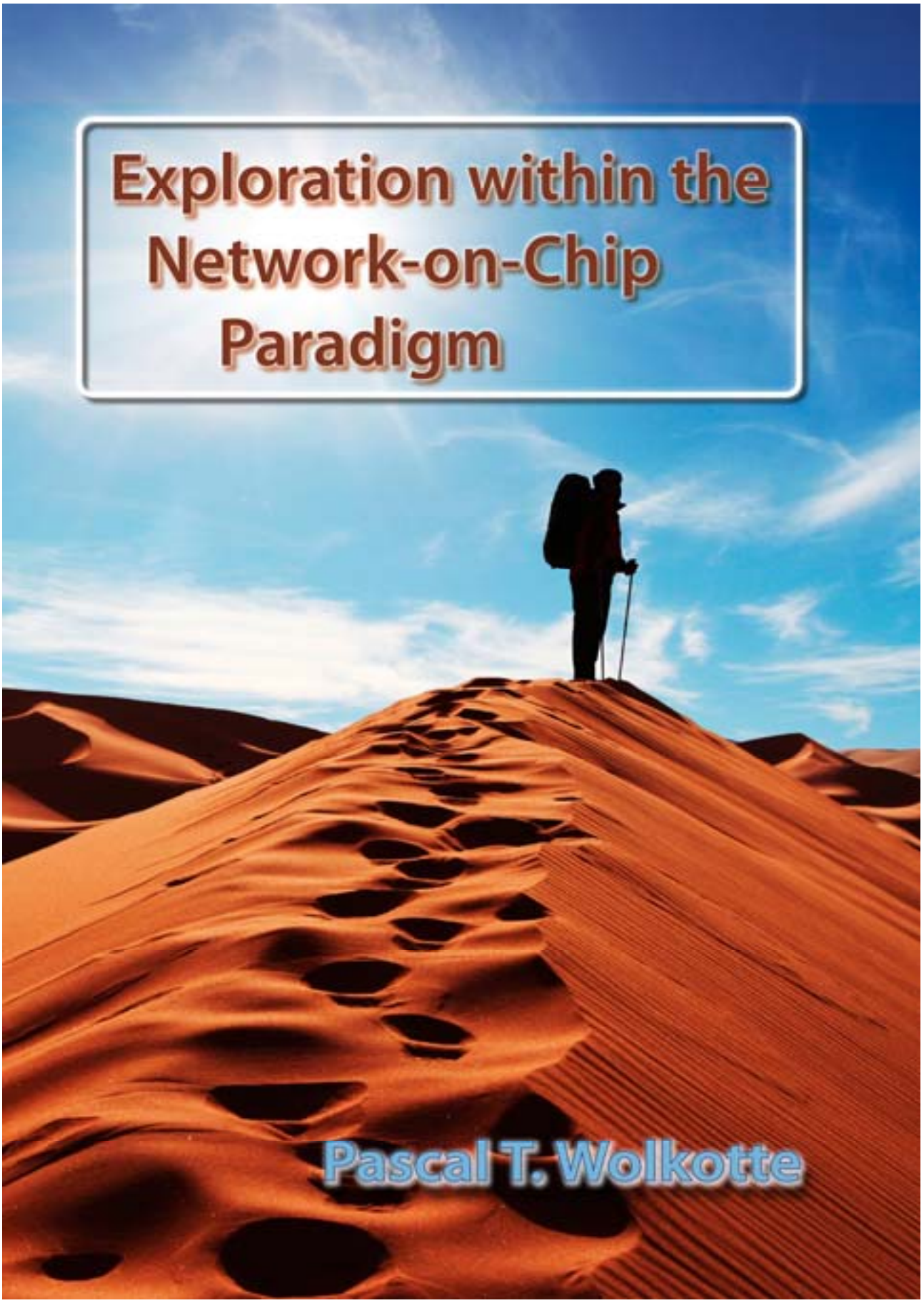
Exploration within the Network-on-Chip Paradigm



Pascal T. Wolkotte

Exploration within the Network-on-Chip Paradigm

Pascal T. Wolkotte

Members of the dissertation committee:

Prof. dr. ir.	G.J.M. Smit	University of Twente (promoter)
Prof. dr. ir.	Th. Krol	University of Twente
Prof. dr. ir.	B. Nauta	University of Twente
Prof. dr.	K.G.W. Goossens	Technical University of Delft NXP Semiconductors, Eindhoven
Prof. Dr.-Ing.	J. Becker	University of Karlsruhe, Germany
Dr.	R.D. Mullins	University of Cambridge, United Kingdom
Prof. dr.	L. Benini	University of Bologna, Italy
Prof. dr. ir.	A.J. Mouthaan	University of Twente (chairman and secretary)

This research is conducted within the Smart Chips for Smart Surroundings project (IST-001908) supported by the Sixth Framework Programme of the European Community.

Computer Architecture for Embedded Systems group
The Faculty of Electrical Engineering,
Mathematics and Computer Science
P.O.Box 217
7500 AE Enschede
The Netherlands

Centre for Telematics and Information Technology
P.O.Box 217
7500 AE Enschede
The Netherlands

This thesis was typeset in Adobe Minion Pro and Adobe Myriad Pro by the author using LaTeX and Ti*k*Z. Credit for the cover photo goes to © Galyna Andrushko – Fotolia.com. The cover was designed by the author. The thesis was printed by Gildeprint, The Netherlands.

ISBN 978-90-365-2757-6
ISSN 1381-3617, CTIT Ph.D.-thesis series No. 09-133
DOI 10.3990/1.9789036527576

Exploration within the Network-on-Chip Paradigm

ter verkrijging van
de graad van doctor aan de Universiteit Twente,
op gezag van de rector magnificus,
prof. dr. H. Brinksma,
volgens besluit van het College voor Promoties
in het openbaar te verdedigen
op donderdag 15 januari 2009 om 16.45 uur

door

Pascal Theodoor Wolkotte

geboren op 16 januari 1979
te Oldenzaal

Dit proefschrift is goedgekeurd door:

Prof. dr. ir. G.J.M. Smit (promotor)

Abstract

A general purpose processor used to consist of a single processing core, which performed and controlled all tasks on the chip. Its functionality and maximum clock frequency grew steadily over the years. Due to the continuous increase of the number of transistors available on-chip and the operational clock frequency, it became impossible to reach every function within the chip in a single clock cycle. Furthermore, centralized control becomes hard with the increase in functionality. This lead to the split of the processing into a set of independent processing cores integrated into a single chip.

These multi-core architectures will rely on a well designed on-chip communication architecture. Global wires and bus-based systems need to be replaced to overcome the problem of wiring and the single point of arbitration. This is introduced as the *Network-on-Chip* (NoC) paradigm. Most of the communication architectures classified as a NoC are a network of routers on-chip, but the paradigm embodies a broader scope. The paradigm enables the sharing of on-chip wiring resources for multiple communication streams to reduce the total wiring required. Furthermore, it enables concurrent communication of concurrently handled data packets. The latter is in contrast to the central arbitration and single communication channel in bus-based systems.

In this thesis we explore the paradigm by implementation and characterization of multiple NoC router architectures. The scope of the communication architecture is the embedding in a heterogeneous multi-core System-on-Chip (SoC) for streaming applications. Six streaming applications, which are used in mobile devices, are analysed. Their common communication characteristics and specific bandwidth requirements are presented. One of the major constraints of these applications is the requirement of Quality of Service (QoS) for the interprocess communication.

Based on application analysis we propose a circuit switched router architecture as opposed to a more flexible packet switched router architecture. The reason for this architecture is the observation that communication patterns in the applications are static. The circuit switched network is integrated in an ARM based heterogeneous reconfigurable multi-core SoC realized in a 0.13 µm CMOS technology.

Besides this architecture, an existing packet switched router architecture, that also offers QoS, is improved and compared with the circuit switched router. Next to the exploration of those two router designs, two other packet switched routers, designed at the University of Cambridge, are included in the in-depth comparison. The four routers are placed and routed in 90 nm CMOS technology. The required

buffering dominates the resource usage of all packet switched routers, which is significantly reduced in a circuit switched architecture. However, the latter pays a penalty by a larger required crossbar and reduced flexibility.

The four routers are also compared for their latency performance and energy consumption. For latency the packet switched networks are simulated with popular synthetic traffic scenarios. The circuit switched router has a deterministic latency, due to the congestion free routes.

The latency analysis shows the higher network utilization for NoCs using virtual channel flow control over wormhole flow control. Furthermore, the allocation mechanisms used in the improved packet switched router, cause a higher latency for randomly distributed packets compared to the router with speculation logic that is tailored for this type of traffic. Despite its higher latency for random traffic, the packet switched network is able to give end-to-end latency guarantees for specific connections, due to deterministic arbitration, as is shown in this thesis.

For the power analysis we compared the four routers using various traffic scenarios. One of the first observations is a high power consumption in idle mode, where no data is transported. The clock-tree and the connected synchronous elements consume the majority of the power. A minor part is the static power, which is directly related to the router's required chip area. Automatic insertion of fine-grain clock gating tremendously reduces this idle dynamic power consumption. With clock-gating, both the static and dynamic component have an equal share in the idle power at a clock frequency of 200 MHz.

The increase in dynamic power consumption is directly related to the number of packets that are transported over the network and the amount of bit flips, i.e. activity, in the payload. Transportation of random payload, i.e. 25% activity, requires almost a factor three more in comparison with a payload of constant values, i.e. all bits inactive. Random activity is observed in the analysed streaming applications for most of the intermediate data. The buffer size has no influence on the packet's dynamic energy consumption, due to the fine-grain clock gating, which makes the packet switched routers as energy efficient as the circuit switched router. Most of the difference in energy consumption between the routers, is caused by the different crossbar dimensions and the extra bits in a packet which are required for routing and allocation. The larger crossbar is required for the circuit switched router to add flexibility, and for the improved packet switched router to enable QoS. A marginal increase in energy consumption is caused by the network congestion.

During the design of the heterogeneous SoC architecture as well as the evaluation of the packet switched routers, we were hampered by the prohibitive simulation times of the architecture's bit and cycle accurate models. Motivated by simulation speed-ups of an FPGA in a Hardware-In-the-Loop (HIL) simulation, we developed a framework to simulate large many-core architectures on a single FPGA. Instead of the instantiation of the whole architecture in parallel in the FPGA, the individual cores are evaluated sequentially. Each core is modified such that the core's internal state and combinational functionality are separated.

As all cores in a homogeneous many-core architecture are identical, we can construct a single hyper core, that embodies all combinational functionality of a

single core. The state of the whole architecture, stored in the FPGA's memory blocks, is updated sequentially by offering a core's old state to the hyper core and store its new state. Using the sequential simulation approach in an FPGA, we are able to simulate two to three orders of magnitude faster compared to cycle and bit-accurate simulations in software.

Samenvatting

Tot voor kort bestond een processor op een chip uit een enkele rekenkern, die alle taken uitvoerde en beheerde. Over de jaren namen de functionaliteit en maximale klok frequentie geleidelijk toe. Door een continue toename van het aantal beschikbare transistors op een chip en de operationele klok frequentie, werd het onmogelijk om alle functies te bereiken binnen een enkele klok periode. Ook zorgde de toename van functionaliteit voor een steeds moeilijkere centrale beheersfunctie. Dit leidde tot de splitsing van een enkele rekenkern in meerdere onafhankelijke rekenkernen, die waren geïntegreerd in een enkele processor chip.

De nieuwe architectuur met meerdere kernen zal moeten vertrouwen op een goed ontworpen communicatiearchitectuur. Lange communicatiedraden en op bussen gebaseerde systemen moeten worden vervangen. Dit is noodzakelijk om problemen bij de bedrading en centrale arbitrage te overkomen. Het paradigma, dat deze systemen vervangt, is geïntroduceerd als *Network-on-Chip* (NoC). Communicatiearchitecturen die worden geclassificeerd als NoCs zijn netwerken van op de chip geïntegreerde sorteerders, maar het paradigma omvat meer. Het maakt de beschikbaarheid van de draden verdeelbaar aan meerdere communicatie stromen, zodat het totaal aantal benodigde draden gereduceerd kan worden. Een NoC staat ook simultane communicatie van berichten toe, die gelijktijdig worden afgehandeld door de beheerder. Het gelijktijdig afhandelen van berichten is niet mogelijk bij centrale arbitrage en in enkele communicatiekanaal, zoals gebruikt in bus gebaseerde systemen.

In dit proefschrift verkennen we het paradigma door de implementatie en karakterisering van verschillende NoC architecturen. De communicatiearchitectuur is toegespitst aan de hand van het gebruik in een heterogene meerkernige System-on-Chip (SoC) voor stroom centrische applicaties. We analyseren zes specifieke applicaties, die typische gebruikt worden in mobiele apparaten. Voor alle applicaties beschrijven we de gemeenschappelijke karakteristieken en eisen en per applicatie presenteren we de vereiste communicatie bandbreedte. Een van de belangrijkste eisen, die alle applicaties stellen aan de communicatie architectuur, is de ondersteuning voor gegarandeerde service voor de onderlinge communicatie tussen de processen van de applicatie.

Gebaseerd op de applicatieanalyse stellen we een sorteerarchitectuur voor die op verbindingen sorteert in plaats van een meer flexibele sorteerder die op berichten sorteert. De belangrijkste reden voor deze architectuur is de statische communicatiepatronen in de geanalyseerde applicaties. Een heterogene herconfigureerbare

meerkernige SoC is gerealiseerd in 0.13 µm CMOS technologie, waarbij een netwerk van verbindingsgeschakelde sorteerders is geïntegreerd als communicatiearchitectuur tussen de processoren.

De verbindingssorteerder wordt ook vergeleken met een bestaande berichtensorteerder. Deze berichtensorteerder is verbeterd en ondersteund communicatie waarbij garanties noodzakelijk zijn. Naast de verkenning van deze twee sorteerders vergelijken we de architecturen ook met twee andere berichtensorteerders, die zijn ontwikkelt aan de Universiteit van Cambridge. De vier sorteerders zijn gerealiseerd in een 90 nm CMOS technologie. De drie berichtensorteerders hebben een relatief groot deel van middelen nodig voor het tijdelijk opslaan van de berichten, wat niet het geval is voor de verbindingssorteerder. Deze laatste heeft daarentegen een groter schakelvlak nodig en biedt minder flexibiliteit.

De vier sorteerders worden ook vergeleken op netwerkvertraging van de berichten en hun energieconsumptie. Voor het bepalen van de berichtvertraging simuleren we drie netwerken van bericht sorteerders, waarbij we middels veel gebruikte kunstmatige scenario's berichten injecteren in het gesimuleerde netwerk. De verbindingssorteerder heeft een vaste vertraging van ingang naar uitgang, doordat conflicten tussen verbindingen niet voorkomen, en wordt dus niet gesimuleerd.

Een hogere netwerkutilisatie wordt waargenomen voor de twee NoCs die virtuele kanalen gebruiken. Door de statische allocatiemechanismen in de verbeterde berichtensorteerder veroorzaakt deze een grotere vertraging van de berichten door het netwerk ten opzichte van de sorteerder die ook virtuele kanalen toepast samen met speculatielogica. Ondanks deze grotere vertraging voor willekeurige berichten is deze sorteerder wél in staat om de vertraging van specifieke berichten door het netwerk tot een vooraf gespecificeerd maximum te beperken. Dit wordt bereikt door middel van voorspelbare arbitrage, zoals ook wordt gedemonstreerd in dit proefschrift.

De energie consumptie van de sorteerders wordt gemeten onder verschillende verkeersdruktescenario's. Een van de eerste observaties is het hoge energieverbruik, zonder dat er berichten worden getransporteerd. Het klokdistributienetwerk van de architectuur en de aangesloten synchrone elementen verbruiken het grootste deel van de verbruikte energie. Een veel kleiner deel zijn de statische lekstromen, die direct gerelateerd zijn aan de benodigde chipoppervlakte van de individuele sorteerders. Automatisch toevoegen van een fijnmazig netwerk van schakelaars, die delen van het kloknetwerk aan- of uitzetten reduceert de verbruikte energie enorm. Na toevoeging van deze schakelaars hebben, bij een klokfrequentie van 200 MHz, de statische en dynamische component van het verbruikte vermogen een gelijk aandeel.

De toename van het dynamische energie verbruik is direct gerelateerd aan de hoeveelheid berichten die worden getransporteerd door de sorteerder in het netwerk. Verder heeft de mate van variabiliteit (activiteit) van bitwaardes in een bericht invloed op het vermogens verbruik. Het transporteren van willekeurige waardes per bericht (een activiteit van 25%) vergt ongeveer drie keer zo veel energie als het transporteren van constante waardes. Een opeenvolging van willekeurige waardes is waargenomen bij veel berichten van de geanalyseerde stroming centrische applica-

ties. De bufferdiepte van de sorteerder heeft geen invloed op de energieconsumptie per bericht, doordat het kloknetwerk fijnmazig wordt geschakeld. Dit maakt de berichtensorteerders even efficiënt als de verbindingssorteerders op het gebied van energieverbruik. Het grootste verschil in verbruik tussen de vier sorteerders wordt veroorzaakt door de dimensies van het schakelvlak en de extra bits in een bericht, die nodig zijn voor routering en toewijzing. Het grotere schakelvlak is bij de verbindingssorteerder nodig, opdat de flexibiliteit van de architectuur toeneemt. Voor de berichtensorteerder wordt het grotere schakelvlak gebruikt om garanties met betrekking tot de vertraging te kunnen geven. Opstopping van berichten in het netwerk heeft maar een kleine invloed op de energiekosten per bericht.

Tijdens het ontwerp van de heterogene SoC architectuur en bij de evaluatie van de berichten sorteerders worden we belemmerd door de ontzettend lange simulatie tijden van de architectuur met behulp van bit- en klokperiodeaccurate modellen. Om deze simulatie tijden te reduceren is er een raamwerk ontwikkelt die het simuleren van grote meerkernige architecturen mogelijk maakt op een enkele FPGA. We zijn hierbij gemotiveerd door de tijd reductie die we eerder behaalden bij een FPGA in een Hardware-In-the-Loop (HIL) simulatie. In tegenstelling tot de instantiatie van de gehele parallelle architectuur, evalueert de FPGA de individuele kernen sequentieel. Per kern scheiden we de combinatorische functionaliteit van elementen die de toestand van de kern beschrijven.

In een homogene meerkernige architectuur zijn alle kernen identiek. Daarom kunnen we een hyperkern maken, die de gecombineerde functionaliteit van de kernen omvat. De complete toestand van de parallelle architectuur, die is opgeslagen in de interne geheugenblokken van de FPGA, wordt sequentieel vernieuwd door de oude toestand van een individuele kern aan te bieden aan de hyperkern en de nieuwe toestand in het geheugen op de slaan. In vergelijking tot bit- en klokperiodeaccurate simulaties in software, is de gerealiseerde sequentiële simulatieaanpak in een FPGA twee tot drie ordegroottes sneller.

Dankwoord

Hier beginnen we dan met de laatste zinnen van dit proefschrift. Bijna klaar. Klaar met het afsluiten van vijf jaar werk. Toch is het proefschrift niet af zonder het bedanken van alle mensen die direct of indirect hebben bijgedragen aan mijn fantastische tijd als promovendus op de universiteit.

Het begon allemaal vlak na mijn afstuderen met een bezoekje aan de Zilverling. Gerard Smit wist erg enthousiast te vertellen over het onderzoek in de groep en zijn idee over mijn inmiddels afgeronde onderzoeksopdracht. Met een erg goed gevoel ging ik naar huis en dat doe ik nog steeds elke dag. Een promovendus in onze groep staat echter niet alleen. Zo wil ik graag de volgende mensen van de groep in het speciaal bedanken voor alle hulp.

Allereerst mijn promotor, Gerard Smit. Hij is een begeleider zoals je hem in het ideale geval zou wensen. Je laat een probleem vallen en de volgende dag heeft hij al weer een suggestie, waar je mee verder kunt, uitgewerkt. Hij voorziet papers sneller van commentaar, dan dat je de papers kunt schrijven. Hij kan ontzettend enthousiast vertellen over al het werk in de groep. Ook is hij niet vies van weddenschappen afsluiten over oplossingen die wel of niet zouden kunnen, met als gevolg dat je weer eens een nachtje slaap mist. Maar ook zeker niet onbelangrijk, hij voorzag onze vele uitstapjes voor het 4S project van de nodige gezelligheid, die ik niet snel vergeet.

In het 4S project werkte ik samen met Lodewijk Smit. Hij hielp me op weg in het prille begin met allerlei zaken: discussies over de de richting van mijn onderzoek, kritische — voor je gevoel soms te kritisch — opmerkingen en vragen over mijn oplossingen. Zelfs na het oprichten van het bedrijf Recore Systems bleef hij geïnteresseerd en behulpzaam.

In heb begin heb ik ook veel hulp gekregen van zijn kompanen bij Recore, Paul Heysters en Gerard Rauwerda. Gerard bedankt voor alle hulp bij het begrijpen van de wireless applicaties. Paul bedankt voor de discussies over architecturen, energie metingen en het leuke excuus voor de vakantie naar Aruba.

Ook wil ik in het speciaal Philip Hölzenspies bedanken. Eén dagje kwam ik gebruik maken van een vrij bureau op je koele kamer en inmiddels zijn we de afgelopen 3 jaar al weer kamergenoten. Philip erg bedankt voor alle subtiele dwang voor en hulp bij het gebruik van Vim en Linux, het beantwoorden van alle vragen over LaTeX, discussies over technische en zeker ook de veel minder technische zaken, hints bij alle illustraties en presentaties, het lezen van mijn papers, en alle ondersteuning bij het schrijven van dit proefschrift.

Ik kon ook nog van veel meer kanten hulp en steun verwachten. Ik wil daarom

bedanken: André, voor het grondig lezen en corrigeren van mijn proefschrift, Bert, voor alle hulp met VHDL en synthese, Pierre, voor een goede introductie in real-time systemen, Nikolay, voor de introductie in het NoC paradigma en de discussies over onze netwerkarchitecturen, Jan, voor het helpen en oplossen van mijn toggle-rate vermoeden, Marcel, voor de leuke samenwerking in het 4S project, Albert en Bas, voor het maken van de MPEG-4 demo op het BCVP, de secretaresses, Marlous, Nicole, Thelma en Tineke, voor het helpen bij al de niet technische zaken, Jochem, voor de discussies over de FPGA simulator en de design flow, Tjerk, voor implementatie van de DDC. Als laatste, maar zeker niet het onbelangrijkste, iedereen van de CAES groep. Bedankt voor de geweldige sfeer zowel tijdens als na werktijd.

Ook buiten de groep wil ik graag een aantal mensen bedanken. Allereerst iedereen die in het 4S project heeft meegeholpen aan de realisatie van de Annabelle Chip. Het is geen ontwerp van een enkel persoon, maar het resultaat van 4 jaar samenwerking. Ik wil ook graag Arnab Banerjee van de Universiteit van Cambridge van harte bedanken. Voor al zijn metingen, de hulp bij mijn energie metingen van de routers en de discussies over de zee van getallen. Daniel Schinkel wil ik graag bedanken voor alle discussies over links en draden in CMOS technologie. Ook wil ik mijn commissieleden bedanken voor de feedback op mijn proefschrift en alle gesprekken tijdens eerdere ontmoetingen op conferenties en bijeenkomsten. Voor de indirecte hulp bij de vormgeving van mijn proefschrift wil ik graag Robert Bringhurst bedanken die me via zijn boek [23] een stuk wijzer heeft gemaakt over typografie. De illustraties in het proefschrift hadden er heel anders uitgezien zonder Till Tantau en Christian Feuersänger met hun werk voor Ti*k*Z.

Tijdens mij onderzoeksperiode kon ik ook voor vier maanden naar Bell Labs in de VS. Het werk is niet opgenomen in dit proefschrift, maar vormde wel een goede opstap naar de simulator. Naast de leuke opdracht heb ik ook een erg goede tijd gehad bij Sape en Connie, waar ik een groot deel van de tijd mocht logeren. Sape en Connie bedankt voor een heerlijke ontspannen tijd en alle kookideeën waar menigeen vandaag nog van profiteert.

Een onderzoek beperkt zich niet alleen tot op het werk. Ook na het werk helpen allerlei ontspannende activiteiten. Zo wil ik iedereen die ik bij Euros heb mogen coachen van harte bedanken. Bedanken voor de geweldige ervaringen op en rond het water. In het bijzonder wil ik bedanken Guus, Martijn, Wabe, Anke, Tessa, Mauro en Sjoerd voor al die uren samen fietsend en coachend langs het kanaal, aan de bar in de kroeg en tijdens de leuke weekenden in het land.

Ook wil ik mijn paranimfen, Jeroen Harmsen en Erik Karstens, van harte bedanken. Bedanken voor alles, voor het zijn van vrienden door dik en dun en al het andere dat te veel is om hier op te noemen. Mijn broer en z'n vriendin wil ik graag bedanken voor al die zaterdagen dat ik kon helpen aan jullie nieuwe huis. Het was heerlijk ontspannend voor mijn gedachten.

Ten slotte wil ik mijn ouders en Antoinette bedanken. Bedanken voor alle hulp bij de niet technische zaken, het trots zijn op wat ik doe en vooral jullie onvoorwaardelijke steun.

Bedankt allemaal, Pascal

Table of Contents

Chapter 1

Introduction

In the every day environment, people are surrounded by a large set of digital devices. The number of these devices is continuously increasing, ranging from small embedded controllers in the car, efficient processors in a mobile device to high performance processors in a laptop and desktop computer. These controllers and processors are ICs realized in CMOS technology. The continuous development of this technology enables more functionality in the same device or smaller devices with the same functionality.

It started with the first working bipolar transistor, realized at Bell Laboratories in 1947 by William Shockley, John Bardeen and Walter Brattain. Jack Kilby, working at Texas Instruments, was the first to combine transistors into an IC in 1958. Gradually more transistors were combined into an IC. This resulted in the present Very-Large-Scale Integration (VLSI) designs, consisting of billions of transistors. The performance and memory capacity of these VLSI designs has increased exponentially over the decades, mainly due to the exponential increase in the amount of transistors.

The increasing amount of transistors was initially dedicated to a single processing core to increase its functionality and the width of its data path. Such a processing core consists of a number of functional blocks. Examples of such blocks are the Arithmetic Logic Unit (ALU), a Floating Point Unit (FPU), Memory Management Unit (MMU) and Multiply Accumulate (MAC) unit. The number of functional blocks of the processing core increased, but they were controlled from a single point. The operational frequency of the processing core could increase due to the shrinking of the transistor size and an increasing number of pipeline stages.

In the last decade, this trend could no longer continue due to, among others, power and memory walls [8]. Instead multiple identical cores were integrated on a single chip, with less focus on increasing the performance of individual cores.

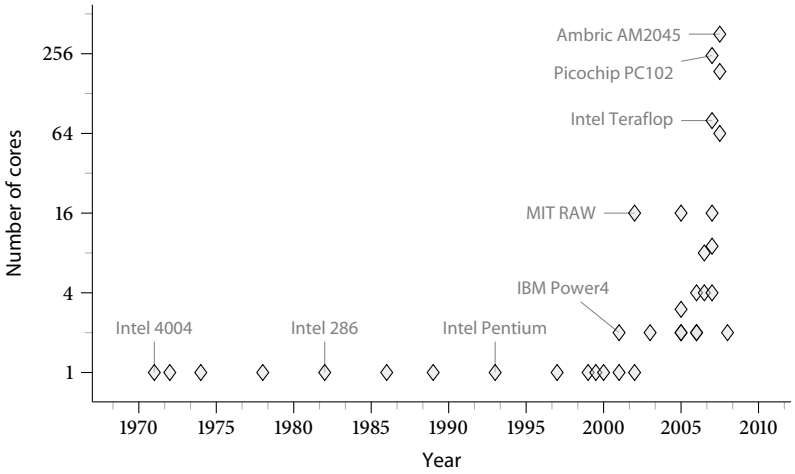


Figure 1.1 – The number of cores on a single chip, adjusted from Amarasinghe [2]

The performance of a single chip increases by adding more cores to the chip with each technology step. This trend in the number of (identical) processing cores per chip over the years is illustrated in figure 1.1 for a selection of chips. We see an exponential increase in the number of cores per chip. Some chips maintained the same functionality per core, which results in a slow increase of the number of cores per chip. Furthermore, designers created chips with a reduced functionality per core, which results in a larger number of cores per chip. With the large number of simple cores the power efficiency and amount of available parallelism on-chip increases, which is suitable for simple but highly parallel tasks.

The trend that is clearly visible with the homogeneous multi-core processors is also common practice for embedded System-on-Chip (SoC) designs. SoCs are used in, for example, mobile devices, digital radio receivers, and TV setup boxes. A SoC does not necessarily consist of a repetition of identical cores, but it may integrate a variety of modules into a single chip. The integration of, for example, some processing cores combined with hardwired modules, AD/DA converters, memory blocks and peripherals reduces the overall size of the system. A single chip solution is often more efficient in resource usage and communication between the blocks compared to multiple components on a PCB. This mix of different cores, or tiles, is often referred to as a heterogeneous multi-processor system.

With the introduction of either homo- or heterogeneous sets of processing cores per chip, extra "glue logic" is required to handle all the communication between the processing cores. The communication between two cores can be solved by direct connections. Interfaces between the cores can be chosen on a pair-by-pair basis. However, this does not scale when the number of cores increases. Furthermore, the cores have to access shared resources like for example external memory, which requires a shared communication medium. A shared communication infrastructure

creates a more flexible system in comparison with application-specific interconnects.

Similar to communication in large-scale systems, initially the on-chip architectures used bus-based communication between multiple cores. All cores are connected to a central bus and a single core can transmit a message via the bus to one or multiple destination cores. Bus arbitration is handled centrally and all nodes, which are connected to the bus, will monitor the bus simultaneously such that they receive the messages destined for them. As recognised by Guerrier and Greiner [60], bus-based architectures suffer from limited scalability and poor performance for large systems. Therefore, bus-based communication is gradually replaced by a network based infrastructure.

The on-chip communication network paradigm is introduced as *Network-on-Chip* (NoC) by Benini and De Micheli [15], Dally and Towles [38], and Sgroi et al. [118]. In the late nineties the first multi-core architectures with an on-chip network were proposed. For example, the MIT RAW architecture that connects its multiple on-chip cores via a programmable switched interconnect [132]. The individual tiles on the chip—processing cores, memory blocks, hardwired modules, etc.—are connected to one or more routers. The interconnection of routers creates a particular network topology. The NoC is a replacement for global interconnect and single bus architectures.

Despite their similarity with off-chip interprocessor communication, the trade-offs on-chip are different. In contrast to off-chip networks, the number of wires available on-chip is large, but the available buffer space is relatively scarce [38]. An IC has an enormous budget of transistors, but only a small fraction can be contributed to the NoC. Therefore, the trade-offs in buffer sizes, routing mechanisms, switch sizes etc. are different from other network routers. Furthermore, all transistors are placed and interconnected in a 2D-plane, which puts constraints on the topology of the network. Other more strict requirements for NoC architectures are the Quality of Service (QoS) demands for communication, which originate from the application constraints. The advantage of an on-chip network is the relatively controlled environment, and fixed organization of the network once the chip is realized. On-chip we also benefit from the huge amount of available wire resources in comparison with the pin limitations for off-chip communication.

In this thesis, we address a detailed exploration of the NoC paradigm. This exploration consists of the implementation, integration and comparison of NoC communication architectures. We limit the scope of the exploration to heterogeneous SoCs that are tailored for streaming wireless and multimedia applications. These applications are often used within mobile systems, where energy is a scarce resource. Research on the energy problem in mobile systems was the main motivation for the *Chameleon* project, which was later succeeded by the *4S project* and other projects. The work described in this thesis was conducted within the 4S project.

The motivation and objectives of this project are outlined in the next section. Within this thesis and the 4S project, energy efficiency is one of the optimization criteria. A short introduction is presented in section 1.2 and more details are provided in appendix A. Section 1.4 gives a more detailed description of the problem

considered within this thesis. The contributions of this thesis are summarized in section 1.5. The last section provides an overview of the structure of the thesis.

1.1 The 4S Project

The overall mission of the 4S (Smart Chips for Smart Surroundings) project is to define and develop efficient (ultra low-power), flexible, reconfigurable core building blocks for future ambient systems, including the supporting tools [120, 121, 123]. As an application, the project has chosen a concrete worldwide broadcast radio application, Digital Radio Mondiale (DRM), and MPEG-4 video that can be used in an ambient system scenario.

The results obtained in the Chameleon project were one of the foundations of the 4S project [24, 122]. In the Chameleon project, the algorithms, architecture, and design of handheld multimedia systems were addressed with emphasis on energy-efficiency. Efficiency is achieved by adapting the entire platform—applications, operating system, and hardware—to the current demands of the system, which is mainly dictated by the dynamic behaviour of the mobile environment.

The projected hardware architecture is a heterogeneous set of processors combined in a SoC. An example of such a SoC is depicted in figure 1.2. The architecture contains a mixture of processing cores, that are each suitable for a selected range of tasks and application domains. Efficiency is obtained by assigning individual tasks of an application to the tiles that can execute the task most efficiently. For example, control oriented tasks are assigned to a General Purpose Processor (GPP) and bit-level operations to an FPGA. For this proposed template a coarse-grain reconfigurable architecture, the MONTIUM, was designed by Heysters [66], which is tailored to the digital signal processing (DSP) application domain.

At the application level, the Chameleon project studied cross-layer optimizations. By monitoring the environmental conditions like Signal-to-Noise Ratio (SNR), the application can change and adapt the settings of the algorithms such that QoS is met with the minimum amount of effort, i.e. be efficient. Being efficient is achieved by Smit [124] for both Universal Mobile Telecommunications System (UMTS) and HiperLAN/2, by reducing the effort spent in error correction and bit error estimation when the channel conditions improve and vice versa.

Whereas the Chameleon project focussed on the design and optimization of individual blocks, the 4S project has a main focus on the integration at both the hardware and software level. The integration of hardware is realized by the design of a flexible SoC platform. On software level the design time flow is studied, such that various tools can be integrated into a single flow that enables a design time cycle of less than eight hours. The process graphs of both the MPEG and DRM applications were implemented and used to verify the design flow. Furthermore, the hardware and software are also integrated by means of run-time software. This included an operating system for the realized SoC and tools to map both MPEG and DRM applications onto this platform at run-time.

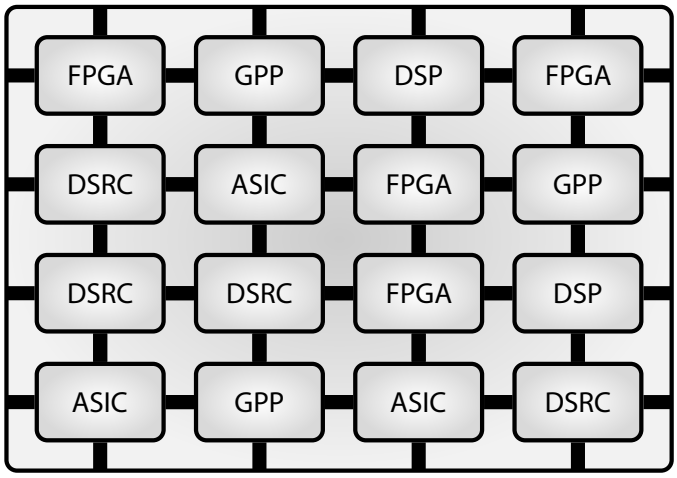


Figure 1.2 – Example of a heterogeneous System-on-Chip

1.1.1 Platform

The hardware platform, realized within the 4S project, is a *heterogeneous dynamically reconfigurable SoC*. Realization of such a platform enables us to verify the expected energy savings by assigning tasks to processors that could execute them most efficiently. Furthermore, integration of reconfigurable processing cores makes the SoC flexible enough to adapt the functionality of the SoC to the continuous evolution and adaptations of standards. In contradiction to ASIC blocks, a reconfigurable processor can adjust its instructions. This will reduce the overall design costs, because the SoC does not require an expensive re-design of the ASIC blocks in case the standard is adjusted.

Because the platform will be a heterogeneous set of processing cores, it requires an interconnection architecture. Traditional on-chip interconnect architectures quite often suffice for a small number of tiles. Within the 4S project a NoC is considered to evaluate its potential increase in performance as well as an increase in efficiency and better support for QoS compared to a bus based infrastructure.

1.1.2 Applications

Multiple applications should be able to run on the heterogeneous SoC platform. A scenario is, for example, a mobile multimedia device that can be used to listen to a radio broadcast, view an MPEG-4 encoded movie and enjoy the owner's personal MP3 music collection. These and other targeted applications within the 4S project and this thesis can be modelled as streaming DSP applications.

In streaming DSP applications, computations can be specified as a data flow graph with streams of data items (the edges) flowing between computation kernels (the nodes). These applications can be naturally expressed in this modelling

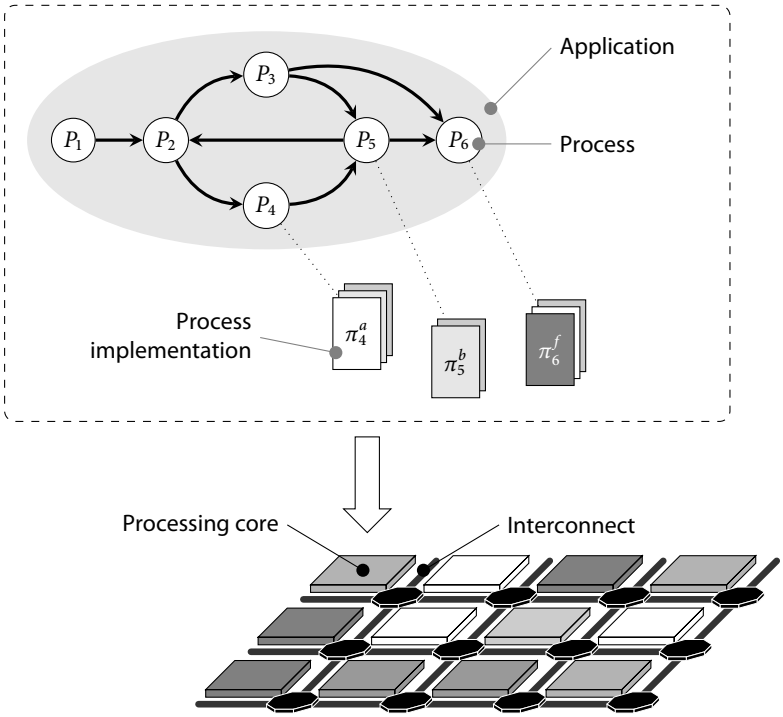


Figure 1.3 – Application specification and targeted platform

style [40]. A typical computational kernel, i.e. *process*, in these graphs is a mathematical algorithm, such as a Fast Fourier Transform (FFT) or a Discrete Cosine Transform (DCT). The top part of Figure 1.3 depicts an example of such a process graph. The processes in these graph will be mapped onto the processing cores and the communication channels between processes are mapped on the interconnect infrastructure.

Typical examples of streaming DSP applications are: wireless baseband processing, multi-media processing, medical image processing and sensor processing e.g. for remote surveillance cameras and phased array radars.

1.1.3 Design Flow

Besides the realized platform, a design flow is necessary that enables the mapping of applications onto this platform. The flow should reduce the development time of an application and provide functions that automatically allocate the processing resources to the multiple processes of an application. The design flow is therefore split into two parts, *design-time* and *run-time*.

Design-Time

The development of an application is done at design-time, where an application is described as a graph of processes, as described in the previous section. Each of the processes (P_i) will have one or more process implementations (π_i^c) for one or more processing cores (c)on the targeted SoC (see figure 1.3). These sets of implementations are required, such that a process can be mapped on multiple processing core types that are present on the heterogeneous SoC. Per implementation, a set of performance figures is determined that is used for the mapping of processes to processing cores at run-time. The edges between the processes are annotated with the communication requirements between the processes. This can only be an average throughput requirement, but may contain extra information like message size or burst characteristics.

Run-Time

The mapping of the application onto the platform, i.e. choosing an implementation and a processing core for every process, is traditionally done at design-time. However, when the set of applications, the required QoS per application and the available processing cores on the platform vary over time, it results in an increasing collection of scenarios where each has its own optimal mapping. For this reason, we propose in the 4S project a design flow that performs the mapping of an application at run-time rather than at compile time. This mapping of the application is referred to as *run-time spatial mapping* and will be performed by the Spatial Mapping Tool (SMIT) [68]. Mapping determines the spatial organization of the set of processes on the set of processing cores and the distribution of the communication on the communication infrastructure. Using the performance figures, QoS and application constraints, and the available resources, the run-time mapping algorithm calculates a (near) optimal mapping. Ideally, the processes of the application will be mapped on the cores that can execute them most efficiently.

A system-wide Operating System (OS), which controls the set of processing cores and their local OS, instantiates the specific mapping on the system. It also adjusts the application's task graph in case of a change in the environment. The reception of a radio signal with a reduced SNR could require additional filtering, for example to suppress an interfering signal. Changing the task graph might lead to an adjustment of parameters, e.g. filter coefficients, or a complete re-mapping of the application(s), due to additional processes, by requesting a new mapping from SMIT. Re-mapping of an application can also be triggered by local failures of the hardware platform. The latter can be used to increase the fault tolerance of the system.

1.1.4 CENTRAL COORDINATING NODE

With the system-wide OS we assume that the SoC is organized as a centralized system. One node in the system, called *Central Coordinating Node* (CCN), performs the system-wide coordination functions. The main task of the CCN is to manage the

system's resources. It performs the run-time spatial mapping of the new applications to suitable processing cores and interprocess communications to the NoC. It also tries to satisfy QoS requirements, to optimize the resource usage and to minimize the energy consumption.

The CCN does not perform run-time scheduling of individual processes and communications while the application is executed. That is performed by the individual tiles with their local OS and the network routers. The CCN only performs the feasibility analysis, spatial mapping, process allocation, and configuration of the tiles and the NoC before an application starts.

Given the fact that the architecture will have a CCN, the CCN has a holistic view of the total problem. Having a global view of the system's parameters and current state makes it possible to take better decisions. An example is QoS for two concurrently running applications. Besides a real-time scheduler on the processor(s), the communication needs to be predictable too. The CCN knows all necessary communication streams and their requirements. This can be combined with the behaviour of the routers, such that QoS can be guaranteed. For example, if a static Time Division Multiplexing (TDM) schedule is used in the routers, the CCN will assign the slots to specific streams. Knowing the slot assignment of all (successive) routers, it can optimize specific streams for latency or any other requirement.

1.2 Energy in CMOS Technology

The energy consumption in a CMOS circuit design can be divided into two major power components. The first, the *dynamic power* of a CMOS circuit, is the energy that is consumed by the components when they are active, i.e. their inputs change. The second major component of the power consumption is the *static power* consumption. This is the nearly constant leakage of the circuit due to non-ideal diodes and transistors. Details on specific components of the power consumption are presented in appendix A. We present a summary in this section.

The largest part of the dynamic power is consumed due to transitions on the outputs of gates. These transitions will cause the output load capacitance of any gate to discharge or charge, which requires energy. This energy is dissipated by the resistance of the wires, but the amount energy that is dissipated is determined by the capacity that has to charged and discharged. This can be described by:

$$P_{switch} = \alpha f C_{eff} V^2 \qquad [\mathrm{W}] \tag{1.1}$$

where α is the activity, f is the frequency on which the circuitry operates, and V is the supply voltage of the circuitry that has an effective capacitance of C_{eff}. This capacitance includes the load of the connected wire (C_{load}), but also includes the internal capacitances of the inverter and input load of the successive gates that have to be charged and discharged as well. Although the complete circuitry has a frequency of f, the individual gates might have a reduced number of transitions. This relative reduction can be corrected via the scaling factor α. A minor part of the dynamic power is consumed by the short circuit power. This part is a result of

Table 1.1 – Predicted development of leakage in CMOS circuitry [65]

Generation	Year	I_{pn}	I_{sub}	I_{gate}
90 nm	2004	25 pA	804 pA	13 pA
50 nm	2010	3 nA	21 nA	52 nA
25 nm	2016	120 nA	260 nA	510 nA

a period, during an input transition, that neither the NMOS nor PMOS transistors are in their cut-off region, which causes a short-circuit current between supply and ground rails.

The second component of the CMOS circuit power consumption is independent of the circuit's activity. In theory the gates should not be conducting when their inputs and outputs voltages are stable, because either the NMOS or PMOS transistors of a gate are within their cut-off region. However, a leakage current (I_{leak}) is present due to the non-ideal transistors. Multiplying this current with the supply voltage (V_{dd}) results in the static power consumption.

When an ASIC is active at its maximum frequency, the dynamic power takes the largest share in the total power consumed. However, when a circuit is inactive, only the leakage current remains unless the supply voltage is switched off. The leakage current will increase due to technology scaling. The increase of leakage current is caused by the quickly reduced oxide thickness (T_{ox}) of the transistors.

A prediction of the three major leakage currents—reverse-bias pn junction leakage (I_{pn}), subthreshold leakage (I_{sub}), and gate leakage (I_{gate})—in future generations CMOS is given in table 1.1. In this table we note a rapid increase of the leakage currents. For technologies of 90nm and above the subthreshold leakage is the main source of leakage. For smaller feature sizes the gate leakage will surpass the I_{sub}.

In this thesis we will focus on techniques that can reduce the dynamic power consumption, but for future technologies it will be worth to consider leakage reduction techniques too. Were reduction of dynamic power will reduce the cost for transportation of bits, the leakage reduction will also reduce the deployment costs of the architecture. Possible techniques to reduce the overall power consumption are presented in appendix A.

1.3 Design-Space Exploration

In this thesis, we study on-chip communication architectures with a primary focus on Network-on-Chip architectures. The NoC communication architectures will be used in multi-core systems.

In the design of the NoC or any other communication infrastructure for multi-core architectures, designers have to make many choices. These choices have an effect on the overall performance of the NoC and on the system as a whole. To achieve a good performance, the designers have to make trade-offs between different performance parameters, e.g. operation bit-widths, memory sizes, packet sizes,

operational frequency, latency, and throughput.

Exploration of communication and processing architectures, to find these trade-offs, can be performed at various levels. In this thesis we have chosen to perform a detailed exploration by means of the implementation, integration and comparison of various router architectures. We want to compare the routers on area requirements in CMOS, communication latency, and energy consumption. Furthermore, within the 4S project, we had to realize a SoC in 0.13 µm technology.

The communication latency of a communication infrastructure can be determined, for example, by a system level simulation. A frequently used language to perform such a simulation is SystemC [102]. An example of SystemC simulation for NoC is the On-Chip Communication Network (OCCN) project, introduced by Coppola et al. [30], which defined a universal Application Programming Interface (API) for specification, modelling, simulation, and design space exploration of NoCs. The general SystemC approach, which supports any design, can also be replaced by simulators dedicated for the communication domain. For example, the OPNET modeler [101] is a domain specific simulator for network systems.

For power/performance analysis the designer can, for example, use Orion. Orion is positioned as a power/performance interconnection network simulator that is capable of providing detailed power characteristics, in addition to performance characteristics, to enable rapid power/performance trade-offs at the architectural-level [133]. A variety of networks can be build using basic building blocks, which have been characterized for power.

All the tools, described above, have incorporated simplifications such that the design is simulated at a higher level of abstraction compared to RTL, which can be synthesized to actual hardware. The level of abstraction determines the speed of simulations. However, it also determines the level of accuracy and it may abstract from not modelled performance bottlenecks.

After an analysis of possible tools for NoC exploration, we have decided to perform the exploration at the RTL level, by designing the router architectures in VHDL. Although building components at this level is considered time-consuming, we had to the build a router module that can be synthesized anyway, because the NoC architecture had to be integrated in the realized SoC. Furthermore, router architectures and other components have to be also modelled in a language for any other exploration tool. Using VHDL, we were not limited by the availability of either power and area models, because a wide-range of accurate values are available for the characterization of standard cell libraries. Furthermore, using a single description of the architecture reduces the risk of design discrepancies.

1.4 Problem Statement

In this thesis, we study on-chip communication architectures with a primary focus on Network-on-Chip architectures. The NoC communication architectures will be used in multi-core systems.

This study consists of a detailed exploration by means of the implementation,

integration and comparison of various router architectures. Exploration of communication and processing architectures can be performed at various levels, as described in the previous section. Higher-level exploration is generally considered to have lower costs, and is therefore often the selected approach. Costs in this exploration consist of, but are not limited to, required knowledge, invested time, and required resources. However, to gain a good understanding of the hardware architecture's full potential and its problems, detailed exploration is required.

Our hypothesis is that the detailed exploration of the NoC will deliver these valuable results against acceptable costs. In this thesis we present the results of the exploration and we discuss frameworks to reduce its costs. We will conclude the thesis with a discussion of the trade-off between obtained results and their required costs.

We limit the scope of the multi-core architectures to heterogeneous SoCs that can be used for streaming applications. As described in section 1.1, we expect the heterogeneous SoC architecture to be a good candidate for energy efficient processing of streaming applications. Efficiency is obtained by assigning each process of an application to the processing core that is able to perform the required computations most efficiently and by minimizing the interprocess communication costs.

Before we propose a suitable communication architecture for these heterogeneous SoCs, we need to examine the communication requirements of the streaming applications. The first objective in this thesis is to quantify the communication requirements of typical streaming applications. Six applications are selected for this analysis. Furthermore, we want to identify their common and different characteristics.

Based on these requirements and characteristics, we can design a communication architecture that is suitable for streaming applications. We want to evaluate and compare implementations that can be directly integrated and realized in silicon. Therefore, we discuss implemented on-chip communication architectures. This is a different level of evaluation compared to a higher level model-based approach, which enables a broad exploration of the router's design space. In this thesis we focus on the design of the communication architecture such that it can be integrated into a SoC that can be realized in silicon. The experience gained with actual implementations of the architecture will give important additional insights which can be used to refine the higher level models. These models can be used by others for their coarse and wide design space explorations.

Besides the actual embedding of the communication architecture in a real system, we also want to measure its performance and compare it with other architectures by objective and simple measurements. We will discuss possible performance measures to evaluate the router architectures. With a selection of possible measurements we compare the proposed communication architectures with router architectures implemented at the same level of detail. This detailed analysis of the router's performance will result in a large number of measurement results. We want to extract a simple model that summarizes the performance of an architecture and can be used in larger system models. These large system models are, for example,

used by design space exploration tools and the spatial mapping tool, see section 1.1.3. The latter tool requires a simple model for the cost of the communication and processing of data to determine where to map each process on the SoC.

During the design of a large SoC, we noticed that, after design space exploration and the selection of blocks to realize the SoC, a considerable simulation effort is required. Primarily, this effort was spent during the integration of the individual blocks into a larger system, which requires the actual architecture description in a Hardware Description Language (HDL). Where design exploration for SoC can be done at various levels of abstraction (and languages), the simulation of the final integrated system requires the same design sources as used for the chip that is realized. This last simulation helps to find errors, overlooked in the previous design stages, and will give extra confidence in the chip just before tape-out. Furthermore, it gives the possibility to extract data traces that can be used for detailed profiling of the architecture's performance and energy consumption. However, due to the large amount of detail, software based simulation results in prohibitive execution times. The benefit of detailed information and disadvantage of long simulation times, resulted in the last topic that is addressed in this thesis.

We examine the possibility to use an FPGA in a Hardware-In-the-Loop (HIL) based simulation approach. We expected that this could result in a considerable decrease in execution times for the cycle and bit-accurate simulations. However, an FPGA has limited hardware resources. We propose and evaluate a framework to overcome these spatial restrictions and simulate large multi-core architectures. We use the original design sources, which are used to synthesize the SoC. These are transformed in such way that a considerable decrease in resource usage is obtained, but still a large increase in simulation speed compared to software based simulations is obtained.

With the answers to the above questions in this thesis, we want to give more detailed insights in the trade-offs that are made when designing the communication architecture for multi-core platforms.

1.5 Contributions of the Thesis

Contributions of this thesis are made in different areas. First, we perform an exploration of a range of streaming applications and their communication requirements. These applications are mapped to heterogeneous SoC architectures, which gives insights in the detailed communication requirements. We present the process graphs of these applications in combination with the required communication bandwidths for the individual edges. Besides the bandwidth requirements we analyse the content of the individual data streams of three applications. The content of the packets gives information on the toggle-rate statistics, which has a direct influence on the power consumption of the communication architecture.

Second, we propose a circuit switched NoC architecture that was motivated by some of the common characteristics found during the application analysis. Together with this circuit switched architecture, an existing packet switched router

is improved and examined in detail.

Third, both NoC architectures are analysed with respect to their performance in area requirements and energy consumption. For the packet switched router, we also measured the network latency under various traffic conditions. These results are compared with the performance of two other NoC architectures. For area and latency comparison, existing metrics are used and we extend the latency measurements with scenarios to test QoS traffic types. For the energy consumption, we propose a number of tests that can be applied to a router, such that its capability to transport various types of traffic can be characterized.

Fourth, the circuit switched router alternative was selected and implemented for a SoC with four processing cores. The heterogeneous SoC is realized in 0.13 µm technology. For a small network of processing cores, the circuit switched alternative is selected and implemented for a SoC that is realized in 0.13 µm technology.

The last contribution is a new framework to perform fast bit and cycle accurate simulations of multi-core architectures. Instead of a software based simulation, the multi-core architecture is mapped onto an FPGA, resulting in a HIL-based simulation. The multi-core architecture is transformed such that the number of required FPGA resources is drastically reduced. The parallel multi-core architecture is simulated sequentially, core-by-core, on this FPGA-based *Sequential Hardware-In-the-Loop Simulator* (SHILS). The SHILS performs bit and cycle accurate simulations of hardware designs without prohibitively long simulation times that are common in software based simulators. The framework is applied to the improved packet switched router architecture, which enables a thorough analysis. Where others present only average results, this framework enables the designer to obtain more detailed characterization of the simulated architecture.

1.6 Structure of the Thesis

In this chapter, we described a trend that is clearly visible in the general purpose processing architectures realized in CMOS technology. This trend is a gradual shift from a chip with a single powerful processing core to a chip with a (large) homogeneous set of processing cores. For SoCs, we also see a shift from a single processing core with some peripherals and small hardwired specialized blocks directly controlled by this single core, to a heterogeneous set of cores which can run autonomously. For both types of architectures, due to the increased parallelism, the communication between the processing cores must be addressed. Network-on-Chip is an approach to improve the performance of this communication and will be the focus of this thesis. A further introduction of the concept and related work to this thesis is described in chapter 2.

We limit the scope of the processing architectures to heterogeneous SoCs for streaming applications. A selection of these applications is discussed in detail in chapter 3. Based on the requirements, we propose a circuit switched router and improve the realization of a packet switched router in chapter 4.

The performance of the routers is evaluated on area requirements, network

latency, and energy consumption in respectively chapters 4, 5, and 6. The evaluation is performed by varying the parameters, e.g. data path width, network load and switch activity. For a selected number of tests the two routers are compared with two other realizations of NoC routers.

Chapter 7 presents the realization of a small on-chip network in a SoC realized in CMOS technology. The long simulation times for the initial network latency analysis and the SoC integration tests were two major motivations for the development of a faster simulation approach for multi-core architectures. The new FPGA-based framework and realized SHILS are presented in chapter 8. In chapter 9, some conclusions and suggestions for future work are given.

Chapter 2

Background and Related Work

Abstract – *The NoC is a specific flavour of interconnection networks. Interconnection networks have been studied for more than two decades and a solid foundation of design techniques is available. We will give a short introduction to the terminology, principles, theory of interconnection networks that are relevant for this thesis. Furthermore, we describe specific characteristics of NoCs in comparison with networks in general. We also present some NoC solutions and present the techniques they employ.*

In this chapter we describe the basic principles of a NoC and present some examples of NoC architectures. The NoC is a specific flavour of interconnection networks. According to the definition given by Dally and Towles [39], an interconnection network is a defined as: "A programmable system that transports data between terminals." A terminal refers to a general block that generates data for or consumes data from other terminals. This definition of an interconnection network occurs at many scales. For example, networks to interconnect on-chip local memories, registers and arithmetic units in a single processor, networks to interconnect processors and memories on PCB and racks, and finally local-area and wide-area networks that connect systems in a building or across the globe. The NoC interconnects the various components of a single VLSI architecture. These components can be small arithmetic units, but most of the NoC studies focus on the interconnection of processing cores and large on-chip memories. The NoC transports data that is communicated between those components and we refer to the combination of the NoC with the components as a *System-on-Chip* (SoC) architecture.

Figure 2.1 depicts an example of a SoC architecture to illustrate the major components for the global on-chip communication. Throughout this thesis we will

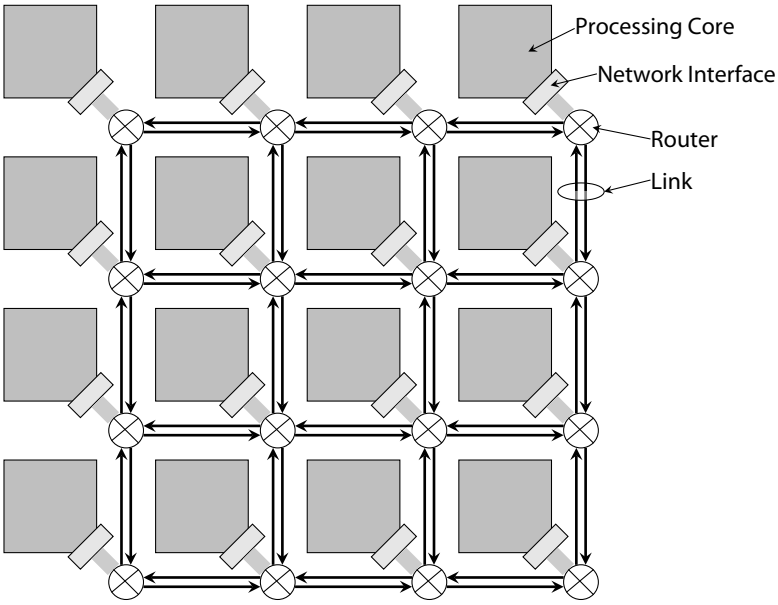


Figure 2.1 – Example of a 2D mesh topology NoC

use the following description of the four basic components:

Cores in a SoC architecture are the producers and consumers of data transported by the NoC. A core contains for example of a processing core, hardwired macro, memory module, or another Intellectual Property (IP) block.

Network Interfaces decouple the computation of the cores from the communication of the network. The Network Interface (NI) implements the interface between network protocol and the protocol(s) supported by the core. Furthermore, it also is a physical interface, which allows different clock and voltage domains for the network and the cores.

Routers handle the network packets according to the chosen network protocol. The packet's items can be buffered in the router and forwarded via the links to the successive router along the packet's route.

Links connect the router nodes and provide the raw bandwidth. They may consist of one or more logical or physical channels. They usually consist of two unidirectional data channels each accompanied with additional control wires both in forward and backward direction. The control wires enable, for example, multiplexing of logic channels onto a physical channel and back pressure to prevent buffer overflow in the successive router. In this thesis we consider all links to be symmetric, i.e. data can flow in both directions via unidirectional channels.

The figure illustrates a specific instance of 16 routers organized in a 4×4 two dimensional mesh, where each router interfaces with a single processing core and its neighbour routers. A wide variety of other organizations, i.e. topologies, is possible, for example, a single router connects to multiple cores or each router connects to exactly two neighbour routers such that a ring structure is created.

In general we consider the cores and network interfaces to be the *terminals* and the routers, i.e. *routers*, together with the links the *interconnection network*. This interconnection network, from now on referred to as network, is primarily defined by its topology and its protocol. The *topology* defines the layout and connections of the routers and the links of the network. The *protocol* specifies the use of the routers and links. The protocol separates two strategies—routing and flow control—to handle the network data. Furthermore, the network protocol can offer traffic specific services. In section 2.1 we describe specific characteristics of NoCs in comparison with networks in general. In the sections 2.2 to 2.5 we present an overview of possible topologies and protocol strategies for NoC architectures. In section 2.6 we present some existing NoC solutions. In this chapter we only provide the information that is relevant for this thesis. For a more detailed overview of possible topologies and protocol strategies we refer to two books by Dally and Towles [39] and Duato et al. [47], and a NoC specific survey by Bjerregaard and Mahadevan [20].

2.1 Network-on-Chip Characteristics

Before we present the basic techniques of an interconnection network we present some specific criteria and requirements for a NoC as depicted in figure 2.1. The NoC is similar to Multi-Processor (MP) networks used to interconnect multi-processor systems. These networks interconnect chips on a PCB or multiple boards in a rack, where as the NoC is integrated on-chip.

The inter-chip and inter-board communications requires signals to go off-chip via pins. Therefore, these networks are heavily bounded by the number of pins of a package, which is limited. Furthermore, an increasing number of pins are grouped in pairs, such that more robust differential signalling can be applied. In contrast with this the NoC can use the large amount of wires available on-chip. Although only a fraction of the wire resources can be used for global communication, the amount of wires is still large. For example, in 65 nm technology the minimum global wire pitch equals 210 nm [74], which enables to bundle thousands of wires crossing an edge of 1 mm length. In comparison, the latest Ball Grid Array (BGA) packages have a pitch of 0.3 mm [77].

One of the limitations of a NoC is the relative small chip area available. The NoC is part of the SoC that is realized on-chip. The largest portion of the chip should be devoted to the computational and memory resources and only a small fraction to the NoC. The NoC's area requirements should be up to 5–10% of the total chip area [20, 38]. For a 0.13 µm processing core the average size is between the 2 mm² and 4 mm² [20, 66]. We assume that a single processing core is accompanied

by a single router, such that a router should have an average size of approximately 0.2 mm^2. Furthermore, due to this small area requirements, the available area for buffering is small.

Another requirement that is different from large MP networks is the layout of the NoC in a 2D plane. The organization of the various modules on-chip is inherently two dimensional, where as the inter-connection and organization of processing boards in a MP system can be three dimensional. Furthermore, the positioning of the individual routers determines the layout of the links between them. Each link will contain a large bundle of wires, which will be positioned in the higher metal layers, because they span over a longer distance. A topology that is inherently regular and can be layout with structured and short links is therefore preferred. It will also reduce the verification costs in comparison with a irregular topology. The layout of those links should be handled with care, such that the signal integrity problems are prevented. The signal integrity problems of structured links can, for example, be solved by shielding, differential signalling and twisted pairs [92].

As described in chapter 1, the number of processing cores increases with the shrinking size of the transistors on a chip. The increasing number of processing cores per technology step will also result in larger network sizes. This continuous scaling requires a scalable network design, such that the communication networks of smaller technology nodes can be implemented with existing router designs.

At the service level of the network a major difference between MP networks and NoC is the need for QoS. Traditional multi-processor systems were focussed on the high performance of the whole system. A NoC is, for example, applied in systems that perform the required processing for time critical streaming applications. For example, applications in mobile phones and TV setup boxes. Those applications require QoS such that the phone call is not interrupted and the frame rate of the video decoding is met.

2.2 Topology

The topology of a network describes the interconnection of the routers by the links. The chosen arrangement determines the possible and efficient protocol strategies implemented by the routers. Furthermore, the organization of the routers influences the positioning of the terminals. In case of a *direct network* this influence is strong, because these types of networks have at least one terminal connected to each router. In contrast, a network where not every router is connected to at least a terminal is called an *indirect network*

The organization of routers can be described as a graph, where routers are the vertices and links the edges of the graph. This model does abstracts away from implementation issues, for example a major constraint for a NoC is the placement of the routers in a 2D plane.

With this graph representation, each topology can be characterized by a few properties. The *degree* of a router is the number of links connecting that node to its neighbour vertices. A topology is considered *regular* when all routers have the same

degree. The *diameter* of the network is the maximum distance between two vertices in the network. Large diameters will result in a higher latency for the messages transported by the network.

A network can grow in multiple dimensions if the number of routers increases. This will result in a faster growth of required wiring resources in comparison to the increase of routers. A measure to quantify these wiring costs is the *bisection width* of a network—the minimum number of wires that must be cut when the network is divided into two equal sets of nodes. We assume that each link in the network has W wires that carry data bits. The *bisection bandwidth* is the collective bandwidth of the wires carrying data on those links. Although a larger bisection bandwidth requires more wiring resources, it offers a higher bandwidth to the terminals interconnected by the routers.

Another measure that is somewhat related to the bisection is the *connectivity*. A network is called k connected if there exist maximally k internally vertex disjoint paths between any pair of vertices. This measure describes the path diversity between the nodes, and therefore can be used to quantify the fault tolerance of a specific topology. With the increase in the number of paths between a pair of nodes, it also is a measure for the performance of the network similar to the bisection bandwidth.

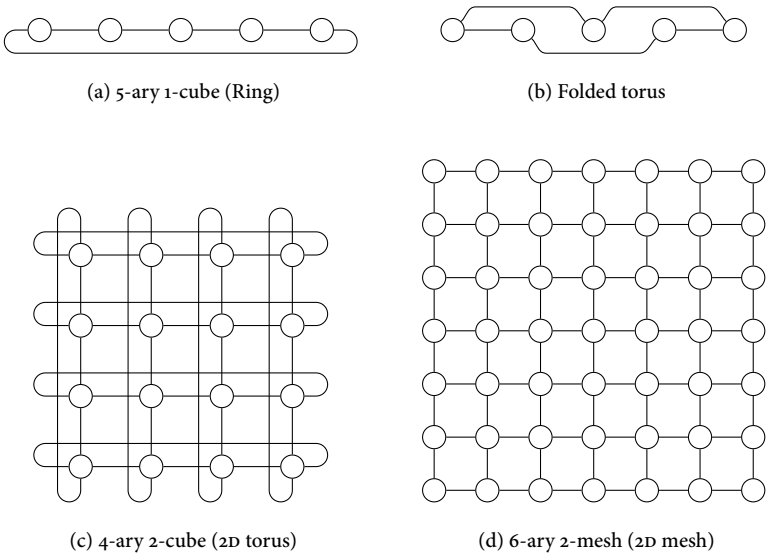
2.2.1 Torus and Mesh Topologies

A simple way to describe a large set of regular network topologies is by a k-ary n-cube, or torus network, where k is the number of nodes per dimension and n the number of dimensions, as introduced by Dally [33]. The k-ary n-cube networks connect $N = k^n$ nodes in a regular grid. The bisection width of a k-ary n-cube equals $2Wk^{n-1}$ and the diameter is $n\lfloor k/2 \rfloor$.

The topologies that are considered for NoC architectures most often are those with low dimensions due to placement of the routers in a 2D plane. Examples of those networks are a ring, with $n = 1$, and the 2D torus, with $n = 2$ as depicted in respectively in figure 2.2(a) and figure 2.2(c). A disadvantage of any torus network is the unequal length of the links when the routers are positioned in a 2D plane. k^{n-1} links wrap around from one edge of the network to the opposite edge. To overcome this problem we can reorganize the positioning of each element, such that a folded torus is created where each link has approximately the same length. Figure 2.2(b) depicts the reorganization of the ring network of figure 2.2(a), which can also be achieved for the 2D torus network of figure 2.2(c).

A variation of the torus networks are mesh networks where the wrap around links are removed as illustrated in figure 2.2(d). For a 2D mesh it simplifies the layout of the links in the 2D plane, but the symmetry of the torus is removed. This can cause load imbalance, as the load on the central links will be in general higher in comparison with the load on links at the edge of the network. Compared to the torus network, the mesh network has the same node degree, but half the bisection bandwidth and the diameter is twice as large.

For mesh and torus networks it is assumed that each router in the network is



(a) 5-ary 1-cube (Ring)

(b) Folded torus

(c) 4-ary 2-cube (2D torus)

(d) 6-ary 2-mesh (2D mesh)

Figure 2.2 – Examples of regular topologies

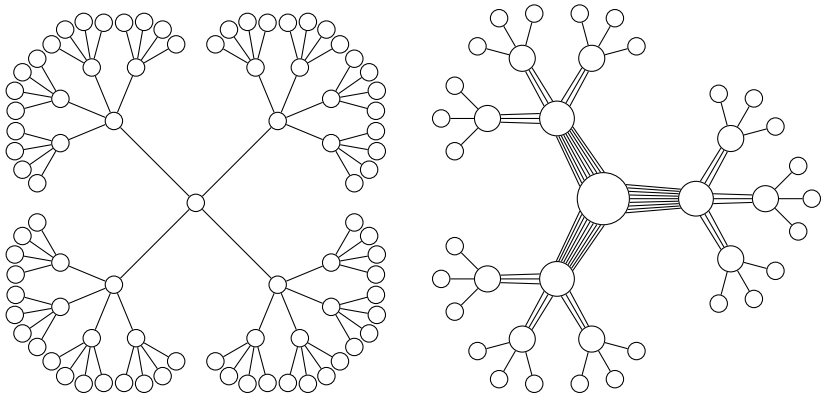
connected to a single processing tile. A concentrated mesh, introduced by Balfour and Dally [10], each router serves four processing tiles. This decreases the average hop count. Additional express channels along the perimeter of the network will restore the reduced bisection bandwidth.

Another special flavour of tori topologies are the 2-ary n-cube networks called hypercubes or binary cubes. In a hypercube network every node is connected to n other nodes, which results in a relatively low diameter for an increasing number of routers. However, with an increase in the number of elements the dimensions and with that the degree of each vertex increases. For large networks this might cause a packaging problem and the layout of the network in a 2D plane will be difficult.

The flattened butterfly topology, introduced by Kim et al. [80], is similar to the hypercube topology. A butterfly network, which has no path diversity, is flattened by combining all routers in each row of the network into a single router. All the connections between routers of different rows are maintained. This increases the radix of each router and the path diversity. In comparison with a hypercube it has a reduced wiring complexity.

2.2.2 Tree Topologies

Two alternative regular topologies are a k-ary tree and the k-ary n-trees, i.e. fat trees as depicted in figure 2.3. The tree topologies interconnect the nodes in form of a tree. Each node, except the root and leaf nodes, has one parent and k children. Tree networks have a small diameter, $log_k(N)$, where N is the total number of vertices



(a) 4-ary tree topology

(b) Fat tree topology

Figure 2.3 – Examples of regular tree topologies

in the network. A major disadvantage of a tree network is its bisection width, which equals W, and low connectivity of 1. This causes a tree to be not fault tolerant and we can expect performance problems at the root of the tree. Tree-based topologies are useful for exploiting locality of traffic.

To overcome the performance bottleneck at the root of the tree, fat trees are introduced. These topologies increase the number of links or bandwidth per link for links that are nearer to the root of the fat tree. This increases the bisection bandwidth, but does not increase the fault tolerance of the network. The major disadvantage of a fat tree is the irregularity of the physical design of the routers due to its variable degree.

2.2.3 Other topologies

We will not discuss a wide range of other topologies in detail. A survey of network topologies is, for example, presented by Balfour and Dally [10], Dally and Towles [39], Duato et al. [47]. The *de Bruijn* networks and *star* graphs try to minimize the network diameter for a fixed number of nodes and node degree [47]. Although the network diameter is low for large networks, the node degree varies with the network size and routing is complex.

The *Octagon* NoC topology was demonstrated by Karim et al. [76]. In its default configuration, eight vertices are organized in a ring and 12 bidirectional links provide a network diameter of 2. Larger networks are created by connecting multiple rings. Irregular topologies are created by combining various topologies in a hierarchical, hybrid or asymmetric organization.

2.3 Routing

A packet will be transported from the source to the destination terminal via the network. The destination can be identified by the use of a unique address for each terminal. However, in most network topologies the number of possible paths between two terminals is more than one. The *routing algorithm* specifies the specific path that a packet will use to reach its destination. The list of algorithms is enormous and we present only the algorithms that are relevant for this thesis. Before presenting some examples we present the taxonomy of routing algorithms to classify a routing algorithm as presented by Duato et al. [47, chapter 4].

2.3.1 Taxonomy of routing algorithms

The algorithms can be first classified by the *number of destinations* a single packet has. We consider only unicast communication, where a packet has a single destination. Having multiple destinations, multicast or broadcast, is not considered in this thesis.

The second classification is based on the *location* of the decisions to be made that are required for the routing algorithm. The path can either be determined by a central controller (centralized routing), the source node (source routing) or by the individual routers while a packet traverses the network (distributed routing). Hybrid schemes are also possible. For centralized routing and source routing the route is determined prior to the injection of the packet, whereas in case of distributed routing the local router can take a decision upon arrival of the packet.

The *implementation* of the routing algorithm is the third classification. The algorithms decisions can be either stored in a routing table (table lookup) or executed in software or hardware using a finite-state machine.

The fourth classification is the *adaptivity* of the algorithm for packets between a given source and destination pair. The path can be either deterministic, oblivious or adaptive. Deterministic routing results in a fixed path for all packets that are communicated between a pair. Identical to the first, oblivious routing chooses a path independent of, i.e. oblivious to, the network state. However, the choice is not necessarily deterministic, e.g. alternates between two output ports of a router. Adaptive routing enables the packet to use a path which is chosen based on information of other network traffic, local congestion, faulty links in the network, etc. However, this increased flexibility makes the individual packet's path non-deterministic.

Adaptivity is often tightly coupled with the location of the routing decision. Local routers often know their and their neighbours current status, such that locally the path can be adapted. However, centrally the coarse global state of a network is often known, which enables other options to adapt paths of specific packets.

Adaptive routing algorithms can be further classified according three criteria. The first criterion is the *progressiveness* of an adaptive path. Progressive routing algorithms do not allow a packet to reconsider a specific routing decision. Upon reservation of a link, the whole packet will traverse over this link to the successive node. In contrast, an algorithm that allows backtracking makes it possible to allow the header of a packet to release previously reserved links and try another path for

that packet.

For both types of progressiveness we can classify the algorithm on its *minimality*. Profitable or minimal routing algorithms will only consider those links that will bring the packet closer to its destination. Non-minimal routing algorithms also allow a packet to use all links of a router.

The last classification for adaptive routing algorithms is the *number of paths* that are considered. Fully adaptive routing algorithms consider all alternatives paths between two terminal pairs, whereas partial adaptive routing algorithms do not.

2.3.2 Deadlock and Starvation

Routing of packets in networks can be sensitive to *deadlock* and *starvation*. Deadlock is a permanent condition in which a system cannot continue to function unless some corrective action is taken. Typical in networks is that multiple packets wait for a condition, e.g. allocation of a link or buffer space, that will never occur. This is caused by a cyclic dependency of network resources used by those packets. The cyclic dependency is created by the routing algorithm.

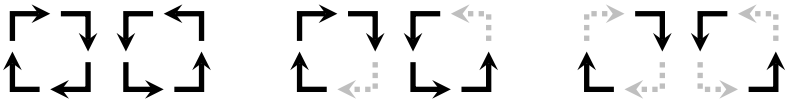
Deadlock can be handled in two ways: *deadlock avoidance* and *deadlock recovery*. A routing algorithm can avoid deadlock by eliminating the cyclic dependencies between resources. These deadlock avoidance techniques put restrictions on the links or buffers a packet is allowed to use. Restrictive link usage is applied in the *turn model* algorithm, which is presented in the next section. Dally and Seitz [36] propose the ordering of Virtual Channel (VC) buffers to prevent cyclic dependencies. Packets are only allowed to use a restrictive set of VCs between two routers based on their source and destination. Because VCs are mutually independent, the cyclic dependency of network resources is prevented.

Deadlock recovery is a method that relies on routing actions that can occur after a deadlock is created. These mechanisms should detect and recover the deadlock, which is assumed to occur in-frequently. Detection is often implemented by monitoring the time between a request and allocation of a resource. If a request is not acknowledge within a certain period, a deadlock situation is assumed. Recovery of the deadlocked packet can be handled by removal of the packet and signalling the source, backtracking the packet and try another route, or store the packet in a special escape buffer and route it with a deadlock free algorithm [46].

Starvation is a situation in which a packet moves through the network without ever reaching its destination. This can occur when a packet uses a non-minimal routing algorithm that misroutes a packet. Algorithms that prevent starvation place an upper bound on the amount of non-minimal steps a packet is allowed to take between source and destination.

2.3.3 Examples of Routing Algorithms

In this section we present some examples of routing algorithms. The simplest routing algorithms are deterministic, because the route is fixed for each source/destination pair.



(a) Cyclic dependency (b) West first routing (c) XY routing

Figure 2.4 – Turn model in a 2D mesh

A well known deterministic routing algorithms is *dimension ordered routing*, which can be applied in k-ary n-cube networks. Those topologies can be decomposed into n orthogonal dimensions.

The distance between a pair of nodes is the sum of all offsets in all the dimensions of the source node with respect to the destination node. For example, in a mesh network this is equal to the Manhattan distance between two nodes. The routing algorithm transports a packet on a 'per dimension' basis. A new dimension is only considered once the offset in the current dimension is reduced to zero. For example, in a 2D mesh (as depicted in figure 2.2(d)) the packet is routed first in the x-direction and then in the y-direction. This algorithm is also known as *XY-routing*. For n-dimensional meshes and hypercubes, it guarantees a deadlock-free routing, but not for torus networks. However, a methodology as proposed by Dally and Seitz [37] enables deadlock-free routing by creating a cyclic free dependency graph for the routing algorithm. Cyclic free graphs in torus networks can be created by, for example, the use of VCs.

A partial adaptive routing algorithm example is the *turn model* introduced by Glass and Ni [55], which applies to the same network topologies as dimension ordered routing. In this routing algorithm a turn is a change in dimension. A change in direction within a dimension is considered two turns, i.e. a 180-degree turn. The model prohibits a minimum amount of turns such that deadlock is avoided. A packet is routed through the network using the turns that are allowed.

To illustrate this we use a 2D mesh topology, which has eight possible turns, as illustrated in figure 2.4(a), that result in two cyclic dependencies. Those cycles can be prevented by prohibiting two non-arbitrary out of the eight turns. An example is the west-first routing algorithm. With this algorithm each packet will follow the path with a priority for the west direction. If that direction is not necessary (any more) than it can choose arbitrarily between the south, east and north direction. Because the west direction has priority, the packet will never make a turn in the west direction which is therefore removed from the possible turns. This is illustrated in figure 2.4(b). Other examples are north-last and negative-first. Compared to XY routing, illustrated in figure 2.4(c), the turn model is less restrictive, because it allows multiple paths between a source and destination terminal. It is up to the designer to make the arbitrary choices dependent on the network state or not.

The *minimal adaptive routing* algorithm, as introduced by Linder and Harden

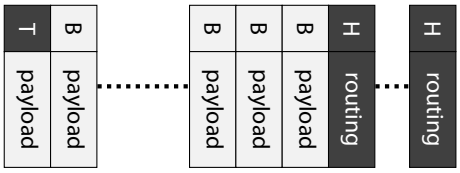


Figure 2.5 – Packet organization in flit-buffer flow control

[87], is a fully adaptive routing algorithm. The algorithm prescribes the use of a minimal path between source and destination in a k-ary n-cube topology. Depending on the source/destination combination a packet is injected into a specific virtual network in which it can be routed along an arbitrary minimal path using local network status. The virtual networks, implemented by virtual channels, avoid deadlock. For an n dimensional topology this requires $n + 1$ virtual channels per channel.

2.4 Flow Control

Data of the terminals is transported in the network by packets. A packet encapsulates the data with extra control information, which is used by the network protocol. A packet encapsulates a portion of the data, i.e. payload, with control information which makes it possible to transport the data over the NoC. Not all data, which has to be transported between two terminals, may fit in a single packet, due to the packet's length limitation. This data has to be divided over several packets and reassembled at the destination terminal.

Packets are transported in the network from one router to the successive router on the packet's path, which is determined by the routing algorithm, via the link that connects the two routers. The *flow control* mechanism handles the transport of the packet by allocation of the network resources, like the physical channel bandwidth and buffer capacity in the router. An efficient flow control mechanism allocates the resources efficiently, such that a high aggregate throughput and low packet latency is obtained.

In general a packet can be broken into smaller fixed size data units called flow control digits or *flits*. A flit is the smallest unit of information recognized by the flow control and a basic unit of bandwidth and storage allocation. Each flit is transported in one or more physical digits or *phits*. A phit is the smallest unit of information that can be transported by a physical channel in a single cycle. Quite often the sizes of a phit and flit are identical. An example of a packet in flit-buffer flow control is depicted in figure 2.5, where the darker blocks contain information for the routers' control logic.

The network is not allowed to split a packet, such that each part uses a different route to the packet's destination terminal. All flits of a packet follow the same route and the order of the flits in a packet can not be re-arranged by the network. The

network packet can be either of a fixed or variable length, which is determined by the type of flow control used.

The flow control can be classified as *buffered* or *bufferless*. The latter does not allocate buffers and only allocates the channel bandwidth and control state. The intermediate routers may buffer flits for a single cycle to reduce the worst-case latency, but the flit in this buffer should be forwarded in the next cycle. Buffered flow control also allocates buffer space in the individual routers, such that a packet can be stored for a longer period in the network until the required channel bandwidth is allocated.

2.4.1 Bufferless Flow Control

Circuit Switching

Circuit switching is a form of bufferless flow control. It establishes a physical path, i.e. *circuit*, from the source to the destination before data is sent. This circuit allocates the channel bandwidth until the circuit is deallocated.

The transmission in circuit switching has four phases. The request phase requests the required links, which are acknowledged upon the allocation of the full circuit in the second phase. When the source terminal receives this acknowledgement the packet's data is transmitted over the circuit. During this transmission idle cycles may be inserted, however, the links can not be used by other packets. The last phase is the deallocation of the links, such that other packets can use the resources again.

Implementation of circuit switching is simple, however, the latency to setup a circuit can be large. Once the desired circuit is reserved, the latency for the data items is deterministic. In chapter 4 we will present more details on circuit switching.

Time Division Multiplexing

A major disadvantage of circuit switching is the under utilization of the available channel bandwidth if the source is not producing data at a rate close the maximum bandwidth of the circuit. Time Division Multiplexing (TDM) of the channels is a way to overcome this disadvantage. Instead of the allocation of a channel to a single circuit at a time, a number of circuits are multiplexed.

It is assumed that all routers in the network have a common notion of time. All circuits are statically scheduled in a global cyclic TDM schedule. The use of channels by different circuits can be made conflict free, such that a flit of a specific circuit can be forwarded in the next cycle at each router. The schedule can either be distributed to all routers or to the terminals that inject the packets in the network. The Æthereal NoC is an architecture that uses TDM to provide guarantees as is described in section 2.6.1.

2.4.2 Buffered Flow Control

In buffered flow control the router may store a full packet or a number of its flits for more than a single cycle in a router. The flits or whole packet are buffered

in the router until the required link to the successive router along the path can be allocated. This in contrast to circuit switching were the payload of the packet is stored at the source until the full circuit is allocated. Buffered flow control is therefore often referred to as *packet switching*[1]. In this class we can distinguish between *packet-buffer* and *flit-buffer flow control*.

In packet-buffer flow control, the buffer of a router should be able to store a complete packet. *Store-and-forward* and *virtual cut-through* flow control belong to this class. The other class of buffered flow control, flit-buffer flow control, consists of *wormhole routing* and *VC flow control*. Despite the suggestion of the name, wormhole routing has nothing to do with routing. Wormhole routers are a sub-class of packet switched routers that, as the name suggests, use wormhole flow control. In flit-buffer flow control, the length of a packet is not coupled with the available buffer size in the router. This makes it possible to increase the packet size to minimize the control overhead or reduce the router's buffer space. Buffers eat a large percentage of the area and energy resources.

Store-and-Forward

The first published description of what we now call packet switching or store-and-forward flow control was described by Baran [13]. With this flow control mechanism, each router along the path of a packet waits until the full packet is received from the preceding router. The header of a received packet is decoded, the specific output port is determined and the router then forwards the packet on to the successive router. A packet is only forwarded to the next router if that router has sufficient storage space to store the full packet. Therefore, each router should provide enough space to store a full packet and the maximum packet length is limited. A single packet will at most allocate a single link at a time.

The latency for this type of flow control is:

$$t_{saf} \geq H\left(t_r + t_w\left\lceil \frac{L}{W} \right\rceil\right) \tag{2.1}$$

where H is the number of hops between the packet's source and destination, t_r is the time to decode the packet's header and wait for allocation of the required link, and t_w is the time to transport a single phit from the router's buffer to the next router's buffer. The total size of the packet, header and payload, is L bits and the link's width is W bits.

Virtual Cut-Through

The major disadvantage of store-and-forward is the assumption that the full packet has to be received entirely before the routing decision is made and forwarded to the next router. Virtual cut-through flow control, introduced by Kermani and Kleinrock [79], reduces the minimum latency of a packet. The first flit(s) of a packet contains

[1] Packet switching also refers to the first flow control in this class, store-and-forward.

the routing information, which can already be examined after this flit is received. If sufficient buffer space is available on the next router along the packet's route, the received flits can already be forwarded, while receiving the remaining flits of the packet.

This results in a reduces latency equal to:

$$t_{vct} \geq H \cdot t_r + t_w \left\lceil \frac{L}{W} \right\rceil \tag{2.2}$$

where we assume a single phit to contain the routing information. In contrast to store-and-forward, a single packet can allocate multiple links.

Wormhole Routing

In wormhole flow control, introduced in the Torus Routing Chip [35], both channel and buffer space are allocated on a flit basis instead on a packet basis. The size of the buffers is now required to be at least a single flit in contrast to at least the maximum packet size. The header flit of a packet is forwarded to the next router if its free buffer space is at least a flit.

A single packet can be distributed over several buffers of successive routers, and the minimum latency is identical to virtual cut-through, because both t_r and t_w have the same minimum value. The packet's size is not limited by the buffer size, and therefore, can be of infinite length. Thus, the procedure of disassembling and reassembly at the terminals is prevented. To indicate the length of a packet, its last flit is marked as tail (see figure 2.5). This tail flit will deallocated the network resources.

As flits do not contain individual routing information, the physical channel between the routers will be owned by a single packet. Other packets, requesting the same link will be blocked until the tail of a packet has deallocated the link. An allocated link can be left idle when the packet's flits are not available in the buffer of the router. These idle time slots can also not be allocated to flits of other packets.

Virtual Channels

Virtual Channel (VC) flow control, introduced by Dally [34], reduces the blocking of multiple links by a single blocked packet. It splits physical channels in a number of virtual channels. Instead of a physical channel, a packet acquires a VC when the head flit allocates the link and buffer resources. Each packet occupies and blocks only a unique VC, where in wormhole flow control each packet occupies and blocks the whole physical channel. The total bandwidth of the physical channel is not increased, but the flits of multiple packets can traverse a physical channel using Time Division Multiple Access (TDMA). Each flit is accompanied by a VC identifier to distinguish the individual flits that belong to different packets. At the input port of a router the flits of different packets are buffered in separate parallel buffers. This prevents head-of-line blocking by flits of other VCs.

2.5 Services

Network design is usually tailored towards the improvement of the average performance. For example, another flow control is chosen to increase the performance or extra prediction logic is added to reduce the minimum delay and latency of a zero loaded network. Despite this increase of the average performance, a situation might occur that the latency of a single packet will be far more than the average latency. Due to congestion, a single packet will have to wait until its requested resources are deallocated by the other packet(s).

If an application is insensitive to large latency variation this is not a major issue, however, there are applications that require specific services. Dependent on their mapping onto the SoC, parts of the network will have to offer different types of services. For example, a part of the traffic may be latency-bounded, while another requires a high throughput. A small part of the packets will have high priority requirements, while other parts even can tolerate data loss.

For these type of packet specific services various traffic classes are introduced. Traffic is divided into a number of *classes*, as introduced by Rijpkema et al. [108]. According to the type of services required, the following types of traffic can be distinguished in the network:

- *Guaranteed Services* (GS) this is the part of the traffic for which the network has to give real-time guarantees (i.e. guaranteed bandwidth, bounded latency). The real-time guarantees are considered hard, such that tight bounds can be given for the communication.
- *Best-effort* (BE) this is the part of the traffic for which the network guarantees only fairness but does not give any bandwidth and timing guarantees. Traffic of this type will in general improve the resource utilization of the network.

Besides these two classes some services have to be provided by the NoC for every packet, because the alternatives are considered to be too expensive. These commitments are: 1) Uncorrupted transport of the packet payload and 2) No loss of packets in the network.

Other GS solutions that offer soft (statistical) bounds, are presented too. For example, Bolotin et al. [22] presented a network with four priorities that distinguishes between signalling messages (highest priority), real-time traffic, read/write traffic and block transfers (lowest priority). Higher priority traffic will pre-empt lower priority traffic, which causes a soft bound on each packet in the network when the traffic in each class is not bounded.

Other types of special services offered by the network can be, for example, broadcast or multicast traffic. This causes a single packet to be transported to multiple destinations, such that the overall network load is reduced.

2.6 Network-on-Chip Solutions

In this section we present a selection of recent NoC solutions that are relevant for this thesis. We distinguish the solutions into two separate sections. First, we

describe two complete NoC solutions, comprising routers, NI, design-flow, etc. Second, we treat a selection of router architectures. Additional NoC surveys are described by Bjerregaard and Mahadevan [20], and Moraes et al. [94].

2.6.1 Integrated NoC solutions

Æthereal

The Æthereal NoC provides both GS and BE services in a single router architecture [57, 108]. For the BE traffic the router uses a conventional input-queue router with wormhole flow control. The packets use source routing and a flit consists of three phits of 32 bit. The input queues of GS and BE traffic are separated and the GS queue buffers have a depth of a single flit.

For the guaranteed services the Æthereal network uses the concept of contention free routing. To guarantee a contention free connection the network uses a global notion of time. Per router a fixed schedule determines the connections within the router and the use of the wires between the routers. In contrast to a parallel circuit switched networks, as described in this thesis, multiple virtual circuit switched networks are scheduled in a pipelined TDM schedule. A communication channel is constructed by reservation of timeslots in the successive routers along the route. The GS flits may not be stalled in a router due to the absence of link-level flow control and the minimum buffer depth. Therefore, a global TDM schedule is determined, in which a GS flit has to reserve timeslot $t + 1$ in a specific router if it passes the preceding router along its route at timeslot t. Unused and unreserved timeslots are filled up by the flits that use the BE services, such that the utilization of the network is maximized.

The initial Æthereal router used a distributed programming model to configure the timeslot table present in each router. The network interfaces determine, in combination with the routers, the suitable timeslots along the connection's route. Each new connection is added to the table via the BE network. This model is scalable, but every router in the network requires a lot of logic and a large lookup table for the timeslot table. In a centralized configuration model the overall TDM schedule is determined by a central entity (similar to the CCN as described in section 1.1.4) in the system. This global schedule and constant flit delay of a single TDM cycle per hop, makes it possible to guarantee contention free routes by controlling the injection times of the flits in the network. In this centralized model the timetable is moved from the routers to the NI and the GS flits are extended with routing information. An approach to determine the TDM schedule is described by Hansson et al. [64]. Hansson and Goossens [62] described the steps and trade-offs to reconfigure the Æthereal network while running multiple applications. Goossens et al. [57] presented a design flow to generate the required communication architecture and validate the performance. The designer can specify the requirements, and the flow outputs the theoretical performance and performance that can be measured with a fast SystemC-based simulation model and a RTL VHDL simulation model for cycle-accurate simulations.

A placed & routed five port router[2] using the distributed programming model, 32 bit links, eight flit deep custom BE buffers and a 256 time slot table occupies 0.24 mm² in 0.13 µm technology [57]. This router operates at a maximum frequency of 500 MHz, which results in a raw bandwidth of 2 GB/s per input and output port. For the centralized model, the timeslot table and its reconfiguration unit can be removed. Furthermore, the 6th crossbar input and output port, which were used internally in the distributed version to configure the timeslot table, can be used as external router port that connects to a link. This simplified six port router has a reduced area of 0.13 mm². When its internal crossbar is changed from N×N to 2N×N, the BE congestion can be reduced, but the area increases to 0.175 mm². When the whole BE part of the router is omitted, the resulting GS only router requires 0.033 mm² and can operate at a maximum frequency of 1 GHz [57].

The NI, which connects the processing cores and other IP modules to the Æthereal network, is introduced by Rădulescu et al. [111]. The NI uses a transaction based model such that backwards compatibility with existing protocols, e.g. Advanced eXtensible Interface (AXI), Device Transaction Level (DTL) and Open Core Protocol (OCP), is guaranteed. The NI consists of multiple NI shells and a single NI kernel. The specific protocols are translated by NI shells to a general peer-to-peer connection that can be translated into network packets by the NI kernel. The kernel's interface offers a configurable number of request and response channels per port, which connects to the individual shells. The area of the NI depends on the specific configuration. An example NI of four ports—one configuration, one slave, and two masters—requires an area of 0.169 mm² before layout and 0.25 mm² after layout in 0.13 µm technology [111].

×*pipes*

The ×pipes architecture is a scalable and high performance NoC architecture introduced by Dall'Osso et al. [32]. The implementation is highly flexible and can be configured upon instantiation of the network. It is build from a SystemC-based library that includes both a router as well as a NI soft macro. The NI provides a standard OCP interface.

Similar to the packet switched architecture described in this thesis the ×pipes router uses source routing and VC flow control, but this router uses an output buffered architecture. Per hop the packet's output port is encoded in the packet's header and this header is stored in the NI using a Lookup Table (LUT). The router architecture is deeply pipelined, seven stage pipeline, to maximize the router's operating frequency. Furthermore, the links can be pipelined too. Each flit is protected by a Cyclic Redundancy Check (CRC) check and an ACK/NACK decision is fed back towards the preceding router. The virtual channels are arbitrated on a flit-by-flit basis.

Due to the error detection circuitry it is possible to detect bit errors for the flits that are communicated between two routers. However, because of the router's and

[2] The crossbar has six ports due to an extra internal input and output port.

link's pipeline it requires multiple cycles before a transmitted flit is acknowledged. The design assumes a link pipeline depth of N and it takes in total M cycles (12 cycles in the described design) in the two routers to generate and interpret the ACK/NACK decision. Both flit and acknowledge have to be transmitted over the link, such that a flit has to be stored $2N + M$ cycles in the buffer after its transmission before it can be discarded or re-transmitted.

Stergiou et al. [127] presented ×pipes Lite, that reimplemented the soft macro library, such that the library modules can be synthesized. Furthermore, the pipeline depth is reduced from seven to two stages. Due to the resulting area reduction, frequency could even increase. Several area figures are reported for a 0.13 µm CMOS technology. For example, a 4×4 router with four buffer stages and a flit width of 64 bit is 0.161 mm² and has a maximum operational frequency of 1 GHz.

With help of the designer, the creation of the communication architecture is controlled by a custom Computer Aided Design (CAD) tool called NETCHIP [18]. NETCHIP incorporates the tools SUNMAP and ×pipesCompiler that respectively perform the topology mapping and selection, and generates the NoC topology using the ×pipes library macros. Angiolini et al. [5] compared two ×pipes NoC topologies, mesh and custom, with two Advanced Microcontroller Bus Architecture (AMBA) bus architectures, AMBA High-performance Bus (AHB) shared bus and AHB Multi-Layer (ML) system. Each fabric connected 30 IP cores of which 15 masters and 15 slaves. The communication fabric is synthesized for a 0.13 µm technology. To obtain realistic wireloads, each core is modelled as a black-box that requires 1 mm² of die area. The ML AMBA is restricted to a 5-layer crossbar, which limits the number of pending transactions to 5. The NoC flit size is configured to either 21 or 38 bit, and the bus architectures have a data width of 32 bit.

The bus architecture resulted in half the achievable frequency (370 MHz) after place & route compared to the NoC architecture (793 MHz). Despite the lower frequency, the ML AMBA outperforms the NoC for single read transactions due the packetisation overhead in the NoC. For write and burst read transactions the NoC shows a lower latency per transaction. The required cell area for the NoC architectures is 1.7 mm² (21 bit) and 2.1 mm² (38 bit) for the mesh topology (15 routers, 30 NIs) and 1.5 mm² for the custom topology (21 bit, 8 routers), compared to only 0.52 mm² for the ML AMBA. The total power consumption of the networks is four to five time as much as the ML AMBA bus. The energy consumption per transported bit to cross a single router is not reported.

2.6.2 NOC ROUTER ARCHITECTURES

Lochside

The Lochside router, presented by Mullins et al. [97, 98], is a router architecture that optimizes routing latency. This router extends a normal input queued VC router with additional control logic. The logic is used to reduce the traditional VC router's control pipeline—1) routing, 2) VC allocation, 3) crossbar allocation, and 4) crossbar and link traversal—for the packet's header flit.

The routing is hidden by performing the routing function in the previous router along the packet's path. This technique is known as look-ahead routing. The result of the routing function is appended to the header flit.

The VC and crossbar allocation is handled in parallel, under the assumption that an unallocated packet is granted a VC. The key idea, which the Lochside router exploits, is the speculation that a newly arrive header flit will not observe contention at its required output port. Therefore, per output port the crossbar allocation is split. For all crossbar requests of flits with already allocated VCs a crossbar allocation is determined and for all newly arrived flits a speculative request of the crossbar allocation is determined. The latter allocation has a lower priority. In case a speculative request is granted, the check for free buffer space in the successive router and the actual allocation of the VC is done in parallel with the crossbar traversal. The crossbar traversal occurs in parallel with all allocation control. This traversal is aborted if two newly arrived flits need the same output port, i.e. the speculation that the flits will not observe contention was not correct.

The router architecture is compared with the architectures presented in this thesis. Its implementation details are discussed in more detail in section 4.3.2. Mullins et al. presented a test chip of 5 mm by 5 mm realized in 0.18 µm CMOS technology [98]. The chip consists of routers in a 4×4 mesh topology. Each router is accompanied by a traffic generator. Each router port consists of four VCs, per VC the buffer depth is four flits, and a flit has a width of 64 bit. The tile—router and traffic generator—is dominated by the router's area, which is more than two thirds of the tile. The reported operational frequency is 250 MHz.

In this thesis we compare our NoC architecture with the Lochside router.

QNoC

The QNoC, as proposed by Bolotin et al. [22], is a router architecture that provides statistical guarantees. The router is an input queued architecture with VC flow control. The four VCs per link are not used to increase the performance compared to wormhole flow control. Similar to the packet switched router described in this thesis, VC are used to provide QoS. However, the QoS is not deterministic.

Each VC is assigned to packets of a specific service type and each flit of a packet can be pre-empted by flits with a higher priority. Packets with the same priority and destined for the same output port are scheduled with a round-robin arbiter. The four priorities distinguish signalling messages (highest priority), real-time traffic, read/write traffic and block transfers (lowest priority). Area and frequency figures are not reported.

SPIN

The SPIN network, introduced by Guerrier et al. [4, 59, 60], is a packet switched network with wormhole flow control and adaptive routing. The indirect network of routers is organized as a fat tree topology. To interconnect a network of 16 routers requires relative long wires. An eight port router, with two extra internal ports, is

implemented in 0.13 µm technology and requires an area of 0.24 mm² and has a maximal operational frequency of 200 MHz [3]. A fat tree network of 16 routers and 32 port for processing cores requires an area of 4.6 mm² after layout.

SoCBUS

SoCBUS is a NoC architecture, proposed by Wiklund and Liu [137], that uses optimistic circuit switching to send the packets between two processing cores. Initially, to setup the circuit, the packet's source sends a request into the network to allocate the links required between source and destination. This request uses a distributed minimum path adaptive routing algorithm. If the destination is reached by the packet an acknowledge is transmitted to the source, which will start with the transmission of packet's payload. The last payload flit will de-allocate all links along the route. If the packet is blocked, because none of the possible output ports at a specific router are free, a negative acknowledgement is sent to the source and the request is dropped. Dropping the request will prevent deadlock.

This setup of the circuit by the packet itself is introduced as a packet connected circuit. Due to the dropping of the packet upon a blocking condition the network is deadlock-free. When the full circuit is acknowledged, latency and throughput guarantees can be given. However, large latencies for setting up a new circuit can be expected, due to the allocation of the complete bandwidth of a link to another circuit. The router has a maximum operational frequency of 1.2 GHz and for a 16 bit five port router an area of 0.06 mm² in a 0.18 µm technology is reported [136].

The circuit switched architecture described in this thesis was inspired by the SoCBUS architecture.

SDM

Another circuit switched solution is proposed by Leroy et al. [86] and is quite similar to the circuit switched architecture in this thesis. In this router architecture every individual wire of an input port can be connected to an arbitrary output port, which makes it possible to tune the bandwidth per circuit at run-time. In contrast to the TDM schedule of the Æthereal network, the authors introduce this concept as Space Division Multiplexing (SDM). Because the lanes are not fixed and a non-blocking crossbar is required, the number of crosspoints of the crossbar inversely grow with the minimum granularity of grouped wires. For example a 5×5 crossbar with 32 bit wide links and a granularity of one wire requires a 160×160 crossbar switch.

MANGO

The MANGO network, proposed by Bjerregaard and Sparso [21], is an asynchronous architecture. The advantages of an asynchronous NoC implementation include low forward latency in pipelined links, zero dynamic idle power consumption and inherent support for Globally-Asynchronous Locally-Synchronous (GALS) systems [19]. Furthermore, irregular channel lengths within the network will not affect the overall network timing.

The MANGO network provides both GS and BE type of services. The GS are connection-oriented by reservation of a VC per hop. Due to a non-blocking input to output crossbar, the variable latency per flit is only influenced by the link arbitration. This variable latency is added to the constant time required to traverse from the VC buffer, via the link and crossbar module to the VC of the successive router. The link arbitration guarantees a minimum bandwidth per VC, such that each connection observes a hard lower bound on the throughput.

The router consists of an output buffered virtual channel architecture. Internal the router is split in a GS and BE router. The BE router occupies a single VC and configures the routing connections for the other seven VC, that are dedicated to the GS traffic. The BE packets have their own routing header flit.

The mechanism to provide GS is identical to the synchronous packet switched router described in this thesis. However, to create a GS connection between two tiles multiple BE packets have to be injected into the network to configure a VC per router that is used for this connection.

The MANGO router is implemented in a 0.12 µm CMOS standard cell technology. This implementation has five ports, 32 bit links and eight VCs per link and uses a four-phase bundled data clock-less control circuit. The pre-layout area is 0.188 mm² and the worst-case timing corresponds to 515 MHz per port. The two largest modules, the crossbar and VC buffers, contribute for respectively 35% and 25% to the total router area.

CHAIN

Another asynchronous NoC architecture is the CHip-Area INterconnect (CHAIN) network, introduced by Bainbridge and Furber [9]. It provides a library of basic NoC building blocks, which can be combined to create a router and larger networks. Each block uses the delay-insensitive asynchronous design style and can handle two data bits per transaction. To support wider links, multiple blocks are placed in parallel. Source routing is used to determine the packet's route through the network. The blocks are relative small, 100–300 transistors per basic block.

2.6.3 SUMMARY

Table 2.1 summarises the protocol strategies for the reviewed NoC solutions. Furthermore, we included a small selection of implementation results for each solution. The area presented in the table is for a single router.

Although the techniques applied in each router architecture vary, some techniques are used quite often. For packet switched architectures the most frequently used protocols are VC flow control and source routing.

The area figures for the individual routers are close to 10% of a processing core in 0.13 µm technology. For example, an area optimised ARM926EJ-S processor with caches at 200 MHz in 0.13 µm technology requires an area of 2.39 mm² [6]. However, the specific configuration per router differs. Some routers have a link width of 32 bit

Table 2.1 – Summary of reviewed NoC solutions

Project	Flow control	Routing	Services	Area [mm^2]	Frequency [MHz]	Techn. [µm]
Æthereal	Wormhole, TDM	Source	BE, GS	0.175[a]	500	0.13
CHAIN	Wormhole	Source	BE	-	-	-
Lochside	VC	XY	BE	< 1[a]	250[b]	0.18
MANGO	VC	Source	BE, GS	0.188	515	0.12
QNoC	VC	Distributed	BE, GS	-	-	-
SDM	Circuit switching	n.a.	GS	-	-	-
SoCBUS	Circuit switching	Minimal adaptive	BE	0.06	1200	0.18
SPIN	Wormhole	Adaptive	BE	0.24[a]	200	0.13
×pipes	VC	Source	BE	0.161[a]	1000	0.13

[a] Area/frequency after layout

[b] Worst-case timing includes link delay

and others of 64 bit and the number of ports also vary from four to six ports of the router. We refer to the sections above and the referenced papers for more details.

Chapter 3

Current and Future Streaming Applications

Abstract – *This chapter gives a description of streaming applications that are mapped to heterogeneous SoC architectures. It provides a selection of applications that can benefit from streaming architectures. The application's standard and mapping of the application gives information on the amount of communication that is required. Besides the required amount of communication the messages themselves are analysed too. The messages will be communicated in packets by the NoC. For the power estimation of the communication, it is important to know the toggling activity behaviour of the messages.*

The mixture of applications analysed in this chapter is characteristic of those running on today's mobile devices. While this is not a comprehensive list of applications, the applications present a representative subset of all applications. For example, in MPEG-4 small video resolution is used, where the number can easily be scaled to other formats as they are for example used in Digital Multimedia Broadcasting (DMB).

For each application we describe the main characteristics and present the process graph of the application as introduced in section 1.1.2. Every process in the

Parts of this chapter have been presented at the 9th International OFDM-Workshop, Dresden, Germany [PW20], the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05) - 12th Reconfigurable Architecture Workshop (RAW 2005), Denver, Colorado, USA [PW16], and the 17th International Conference on Field-Programmable Logic and Applications (FPL 2007), Amsterdam, The Netherlands [PW4].

graph consumes and generates, i.e. communicates, a certain amount of data per second (bit/s) that has to be transported over the NoC to another processing tile, if both processes are not mapped on the same tile. We only determine the required bandwidth of the major communication streams in the process graph. Per communication stream the data, that is required together by the next process, will be grouped in a single message. These messages will be communicated by the NoC by embedding them in network packet. The width of a data item and the number of data items in a message varies between streams.

For energy measurements, as performed in chapter 6, it is also important to know the switch activity within a single message. The activity (α) of a message is defined as the number of rising edges in the message relative to the message length. The α has a direct influence on the dynamic power consumption as described in eq. (A.1). Per bit position[1] (b) the activity ($\alpha_{b,m}$) in a single message (m) is defined as:

$$\alpha_{b,m} = \frac{r_{b,m}}{L_m} \tag{3.1}$$

where $r_{b,m}$ is the number of rising edges in the message, which has length L_m. It is assumed that a message is preceded and ended with all zeros, and not interleaved with other messages. For random data the activity per bit position equals ¼, because the chance of two successive bits not having the same value, i.e. causing a rising or falling edge, equals ½. The activity of individual communication streams is only reported for those applications of which we have data traces available.

The applications are grouped in three categories. Section 3.2 describes three wireless communication standards. Section 3.3 describes digital broadcasting standards that will replace current analogue broadcasting. The third category, in section 3.4, describes a MPEG-4 video decoding, which is widely used in multimedia devices. Many wireless applications analysed are based on Orthogonal Frequency Division Multiplexing (OFDM) based wireless transmission, which is introduced in section 3.1. The chapter concludes with the common characteristics of all the applications.

3.1 OFDM

Orthogonal Frequency Division Multiplexing (OFDM) is used in several communication standards like phone lines, satellite communication, digital radio and television broadcasting, and wireless network systems. It is a special kind of multicarrier modulation. The modulation technique divides the high data rate information in several parallel bit streams and each of those bit streams modulates a separate sub-carrier. Compared to a single carrier modulation, Frequency Division Multiplexing (FDM) gives longer symbol times, which makes it less susceptible to impulse noise, signal reflections and other impairments at the same data rates. Using FDM with a set of orthogonal subcarriers makes it possible to omit the guard bands between the carriers, which increases the spectral efficiency. In OFDM the spectra of subcarriers

[1] The least significant bit has identifier 0.

overlap, but as long as orthogonality is maintained, the subcarriers do not interfere while it is still possible to recover the individual carriers.

In practice the set of subcarriers is generated by an inverse Discrete Fourier Transform (DFT), because its N sinoids form an orthogonal basis set. The phase and amplitude of each sinoid, i.e. source symbols, is determined by one or more bits using for example Quadrature Phase-Shift Keying (QPSK) or -Quadrature Amplitude Modulation (QAM) . At the transmitter side, N source symbols are mapped to the N frequency inputs of an inverse Fast Fourier Transform (IFFT)[2] and transformed to the time domain. The resulting time signal has a duration of T_u seconds and is quite often called an OFDM symbol.

The time signal is transmitted over the wireless channel after frequency transformation to the frequency band of the specific OFDM channel. In wireless communication, the transmitted signal will arrive at the receiver side including multiple delayed versions that are caused by multipath effects due to reflections of the signal on several objects. The delayed versions will distort both the next transmitted OFDM symbol and the current OFDM symbol. The former is called inter symbol interference and can be prevented by inserting an empty guard time (T_g) between two successive symbols. Because the time span of a channel is typically much shorter than the symbol time, the ratio T_g/T_u is typically 25% or less. The latter distortion is called intra-symbol interference, which causes the individual subcarriers to interfere. This interference is a convolution of the time signals. In general, a convolution of signals in the time domain is equivalent to a multiplication in the frequency domain. For discrete time signals this equivalence is only true if the time signal is of infinite length or periodic, which can be achieved by replacing the empty T_g with a cyclic prefix. The cyclic prefix is a replica of the last N_g samples of the OFDM symbol.

An OFDM receiver can be modelled as process graph which can be schematically represented by figure 3.1. The input samples are complex fixed-point numbers that are generally sampled by an AD-converter and are already down converted in the analogue domain or down converted by a Digital Down Converter (DDC) after sampling the analogue signal at a higher rate than the used bandwidth.

The first process, ① → ②, synchronizes the receiver with the transmitter. Either the transmitter sends a continuous stream of OFDM symbols or the transmission is bursty. For a continuous stream, the receiver can synchronize on the cyclic prefix. The channel conditions are estimated using a number of carriers with known values, i.e. pilots. For bursty transmission a group (frame) of data symbols is preceded with one or more known preamble symbols. These preambles contain a known repetitive pattern that can be detected after correlation. Because the whole symbol is known, channel estimation can also be performed. Just a few pilots suffice (for the remaining data symbols) to track the channel conditions.

After synchronization of the receiver, it has sufficient information on how to group samples into useful OFDM symbols and to remove the guard time samples. Because OFDM relies on orthogonality, it is important to correct (minor) frequency offsets, ② → ③, before the symbol is transformed to a frequency domain signal

[2] The IFFT is an efficient implementation of the inverse DFT.

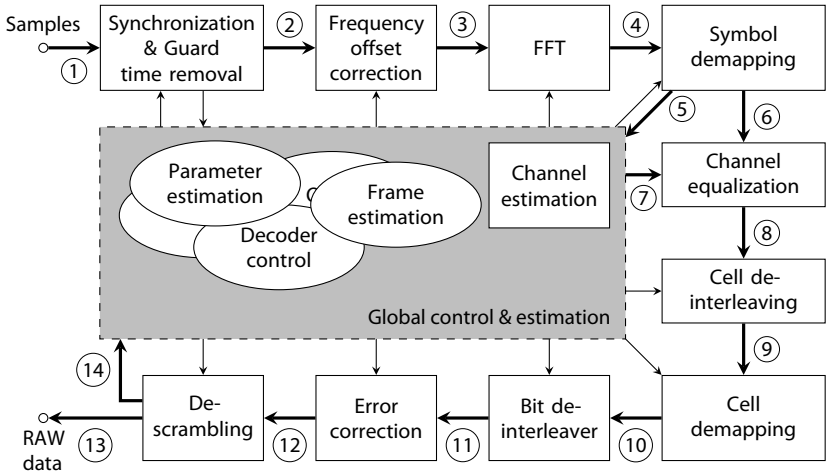


FIGURE 3.1 – A generalized OFDM receiver

with N subcarriers by the FFT, (3)→(4). Each subcarrier is represented by a complex sample that is quite often called a cell. A cell contains modulated data, known pilot information, or is not used. Unused cells are the DC frequency, and multiple cells at the borders of the frequency band to prevent interference with other frequency bands. All the data cells are corrected, (6)→(8), by the inverse of the channel's frequency response, (7), that is estimated by the receiver's control process. Estimation is done using known pilot cells, (5), and preamble symbols when available.

At the transmitter, successive data cells can be interleaved over non adjacent frequency cells and symbols to provide frequency and time diversity. In case of burst errors in either frequency or time, this will spread the errors in the actual data stream. After the cell de-interleaving, (9), the complex samples of each cell can be demodulated to either hard or soft bits, (10). These bits have to be de-interleaved too (10)→(11). Bit interleaving has the same purpose as cell interleaving, but is included in each standard in contrast to cell interleaving. Burst errors have to be prevented as they are hard to correct by convolution and turbo code based error correction schemes.

After error correction, (12), the receiver's physical layer has processed the data which can subsequently be used by the higher layers, like the MAC layer or MPEG-4 source decoding. These higher layers may produce relative large sequences of zero valued bits or other undesired regularities. Therefore, the first step at the transmitter's physical layer and last step of the receiver's is to scramble and descramble respectively the bitstream, (12)→(13), by a modulo-2 addition with a known pseudo-random binary sequence. This operation will not add or remove data from the stream. After the descrambling, a small part of the information stream is not required by the higher layers, but describes the receiver configuration for successive

OFDM symbols that are to be received and decoded, (14).

Before symbol demapping, the data rates are rather fixed because the symbol time, size and number of used cells are fixed for a relative long time. The data rates after symbol demapping and error correction vary quite considerably. This depends on the modulation type that has been chosen. For example, a 64-QAM scheme produces three times more data compared to a QPSK scheme. The error correction handles variable length packets and the rate of correction is varied by specifying a large number of puncturing patterns.

3.2 Wireless Communication

3.2.1 HIPERLAN/2 (802.11)

Wireless Local Area Network (WLAN) networks use radio technologies such as IEEE 802.11a or HiperLAN/2 to provide secure, reliable, fast wireless connectivity. They operate in the unlicensed 2.4 and 5 GHz radio bands, with data rates up to 54 Mbps. The two standards are very much alike and therefore we will concentrate on HiperLAN/2. The physical layer of HiperLAN/2 is described in [53]. The task of the physical layer in HiperLAN/2 is to modulate bits that originate from the data link control layer on the transmitter side and to demodulate them on the receiver side. The frequency spectrum available to HiperLAN/2 is divided into 19 so called channels, which are referred to as radio channels. Each of those radio channels have a frequency bandwidth of 20 MHz.

For modulation per channel, HiperLAN/2 uses OFDM with 64 subcarriers. Of those 64 subcarriers, 48 are used for data, four for pilot cells and the remaining 12 are unused. The signal is extended with a guard time, which causes the OFDM symbol to have a length of 80 samples. Combined with the 20 MHz bandwidth the symbol time becomes 4 µs. Symbols are grouped in frames of 2 ms. A frame starts with a small number of preamble symbols. Compared to the generalized OFDM receiver of figure 3.1 HiperLAN/2 does not have cell interleaving. For error correction it uses Viterbi decoding.

The physical layer of the HiperLAN/2 standard has been mapped on a multi-tile architecture [66, 105]. In this implementation the deinterleaving, Viterbi-decoder and descrambling were omitted. The total processing is divided over multiple processes, such that the designers can map it on multiple processors and benefit from functional pipelining. One should guarantee that each 4 µs a new OFDM symbol can be processed to keep up with the AD-converter. All operations until the demapping are performed on these OFDM symbols. This results in a block-based communication stream between the successive processing tiles.

The communication between each of the processes is analysed for a single frame burst of 2 ms. In this analysis the preamble consisted of two symbols which were succeeded by 498 data symbols. Based on this implementation, we determined the message sizes and number of transported messages per second, which can be translated into required bandwidth per communication stream in Mbit/sec. Table 3.1(a) gives the required communication bandwidth, based on 16 bits quantization for the

TABLE 3.1 – Communication in HiperLAN/2

(a) Bandwidth requirements

#	Edge(s)	Bandwidth [Mbit/s]
1	DDC → Sync.	640
2–4	Sync. → Symbol demapping	512
5	Symbol demap. → Control	32
6–9	Symbol demap. → Cell demap.	416
10–11	Cell demap. → Viterbi	12[a] – 72[b]

[a] BPSK
[b] 64-QAM

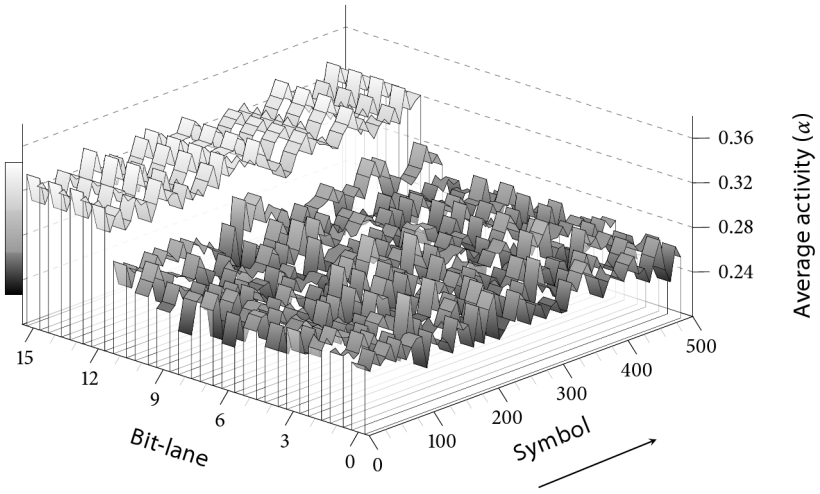
(b) Activity statistics

#	Producing process	Message size	Activity [%]			
		[items]	min	max	mean	std
1	DDC	80	15.0	38.7	27.5	3.8
2	Synchronization	64	12.5	39.1	26.9	3.8
3	Frequency corr.	64	14.1	40.6	27.0	4.0
4	FFT	52	11.5	40.4	25.6	3.7
8	Channel eq.	48	8.3	45.8	25.5	4.1
10	Cell demap. (ASCII)	24	4.2	41.7	20.9	8.0

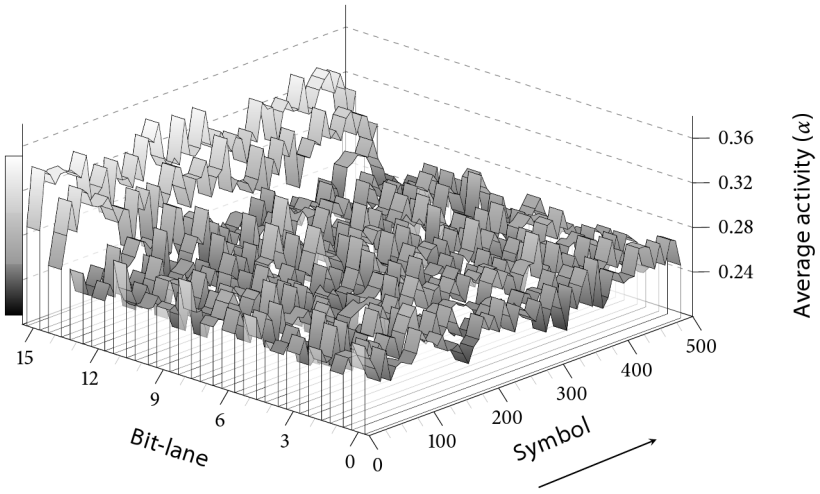
baseband processing. For edge seven the bandwidth requirement is an upper bound, because the channel estimation may update the channel's frequency response not on a per symbol basis.

From earlier projects [105, 115] we used an encoded stream that was transmitted through an AWGN channel model without multipath effects and received with SNR of 11 dB. Per channel between two processes the messages contained a number of items (as given by table 3.1(b)) and each item contained either a 32 bit complex sample or a single byte of raw date, (10). If the data is uniformly distributed over the full number range, we expect an average α of $1/4$ and a small standard deviation. A selection of the α versus time and bit lane is given in figure 3.2. All other α plots are given in section B.1. The statistics of the α is given in table 3.1(b), which is determined over all bits that are transported over a specific edge.

From the α statistics in table 3.1(b) and α versus time and bit lanes in figure 3.2 and figure B.1 we can draw a few conclusions. For the majority of the edges the α is indeed close to the expected value of ¼, especially the least significant bits. However, especially the edges in the time domain (edge one to three), the samples do not use the full dynamic range (caused by signal back-off) of the 16 bit fixed point numbers. This can be seen in the α plots where the three most significant bits and the sign bit have identical α values. In case of HiperLAN/2 the αs for those bits are even above



(a) DDC, real part (16 bit)



(b) Equalization, real part (16 bit)

FIGURE 3.2 – Message's activity versus time and bit lanes in HiperLAN/2

the expected value due to the fact that the time domain signal is fluctuating around zero. This causes more rising and falling edges in a two-complement's notation.

A second observation is the radical change of the α plot before and after specific processes (e.g. the FFT and equalization as depicted in figures B.1(d) and B.1(e)). The last observation, which surprised us, was the typical α plot of the decoded hard bits (see figure B.1(f)). Bit five and seven have a significantly lower α. This is caused by the type of data that is transmitted over the channel. It was an English text in plain ASCII-format and this causes a relatively lower α for the two specific bits.

3.2.2 WIMAX (802.16)

The IEEE 802.16 standard [70] is a new standard for broadband wireless access. It delivers broadband wireless communication for a Metropolitan Area Network (MAN) with a theoretical range of 70 km. The 802.16 standard was designed to evolve as a set of air interfaces based on a common MAC protocol but with physical layer specifications dependent on the spectrum of use and the associated regulation. The physical layer is targeted for two frequency ranges:

10-66 GHz This frequency range is used for line-of-sight (LOS) communication. With the single carrier modulation a point-to-multipoint architecture is designed. The base station allocates time slots for the individual subscriber stations.

2-11 GHz This frequency range is used for non-line-of-sight (NLOS) communication. The current draft of the standard specifies three options for the air interface. They are based on different modulation techniques and subscriber access control.

WiMAX and the 802.16 standard are mentioned quite often both in one breath, but WiMAX focusses, for now, on one of the four air interfaces that is optimized for NLOS using OFDM and is described in this section. The WiMAX air interface is based on a fixed 256 subcarrier OFDM symbol with a variable channel bandwidth between 1.25 and 20 MHz. Of all carriers 192 are allocated for data cells and eight for pilots. The remaining carriers are not used to allow for a guard band and an unused centre frequency. Each of the data carriers is modulated using Binary Phase-Shift Keying (BPSK), QPSK, 16-QAM or 64-QAM . Per burst of symbols the modulation type is fixed. The type of modulation is related to the channel coding rates. The channel coding consists of a combination of convolutional and Reed-Solomon (RS) encoding. The combination of a Viterbi decoder with a RS decoder makes the system robust against both uniformly-distributed errors as well as burst errors. It is optional to include turbo-coding.

The system can either be Time Division Duplex (TDD), Frequency Division Duplex (FDD) or half duplex FDD for the subscriber stations. The OFDM symbols are grouped in a frame of 5, 10 or 20 ms. In a frame the base station prescribes the organization of the frame for both uplink and downlink transmissions. This organization prescribes the specific slots for the individual subscriber stations and modulation and coding combination for the successive symbol bursts.

Table 3.2 – Communication in WiMAX, bandwidth requirements

#	Edge	Bandwidth [Mbit/s]
1	DDC → Sync.	184.32
2–4	Sync. → Symbol demapping	147.456[a]
5	Symbol demap. → Control	4.608
6–9	Symbol demap. → Cell demap.	110.592
10–11	Cell demapping → Viterbi	165.888[b]
11b	Viterbi → Reed-Solomon	17.28
12–13	Reed-Solomon → MAC-layer	15.408

[a] Ratio of guard time versus useful symbol time is ¼

[b] 64-QAM and 8-bit softbits

Due to all the possible frequency bandwidths, possible subchannels[3], modulation types, coding rates, and the exact organization of the frame; it is not easy to give an average value of the required bandwidth for either the base station's or subscriber station's transmitter and receiver. To give a communication bandwidth indication we use a frequency bandwidth of 5 MHz, no sub-channels, and the least protective combination of channel coding (coding rate ¾) and modulation type (64-QAM). This frequency band requires an effective bandwidth of 5.76 MHz that results in a symbol time of 44.44 µs excluding the guard time. The corresponding bandwidth requirements are listed in Table 3.2.

3.2.3 UMTS

The UMTS standard [67] is an example of a Third Generation (3G) mobile communication system. UMTS is based on Wideband Code Division Multiple Access (W-CDMA). In Code Division Multiple Access (CDMA), every transmitted bit is coded by multiplying each bit with a spreading code of a higher rate. Different users use different orthogonal spreading codes. The spreading code consists of a sequence of so-called chips, whose rate is referred to as the chip rate. In this way the information is transmitted at the chip rate, so the spectrum is spread. As a consequence, many correlations with the spreading code have to be performed in the UMTS receiver.

Rauwerda [105] mapped the downlink of a UMTS W-CDMA receiver on a set of (reconfigurable) processors. Figure 3.3 depicts the basic block diagram of the implemented W-CDMA receiver. In contrast to the OFDM based implementations the data processing and communication between the processors is streaming oriented. At a regular short interval a very small message, containing one sample, has to be transported to the successive processor. A streaming oriented implementation reduces the memory requirements per process.

The properties of the communication streams between the processes are listed in table 3.3. The AD-converter has an oversampling rate of four and every sample,

[3] Only a limited set of the 192 data cells is used.

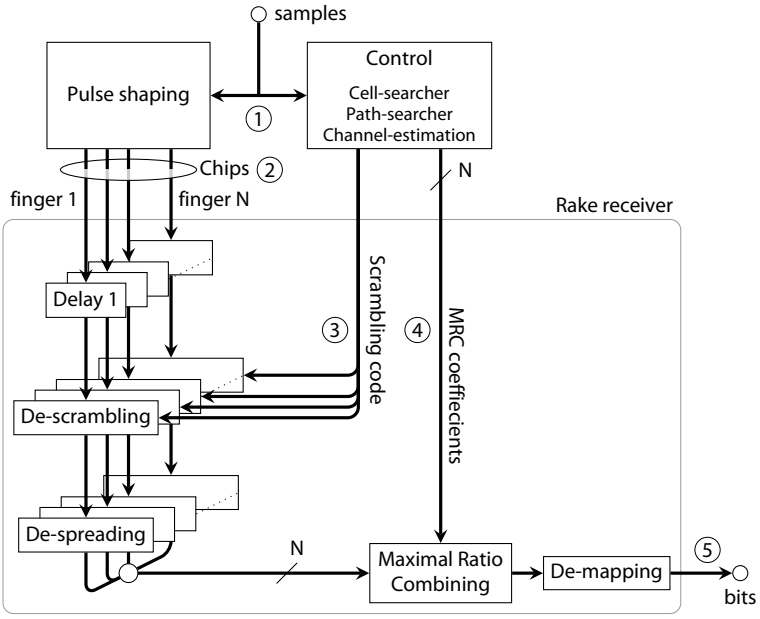


Figure 3.3 – A UMTS W-CDMA receiver with N RAKE fingers

Table 3.3 – Communication in UMTS, bandwidth requirements

#	Edge	Bandwidth [Mbit/s]
1	ADC	245.76
2	Chips (per finger)	61.44
3	Scrambling code	7.68
4	MRC coefficient (per finger)	61.44/SF
5	Received bits	7.68/SF[a] – 15.36/SF[b]

[a] QPSK
[b] 16-QAM

chip or coefficient is represented by an eight bit complex value. For example, the total communication bandwidth for processing four RAKE fingers with a spreading factor (SF) of four is ≈570 Mbit/s.

3.3 Digital Broadcasting

Digital broadcasting systems differ slightly from the communication standards as described in the previous section. The major difference is the one-way communication from the broadcaster (base station in previous section) to the receiver (subscriber station in the previous section). In this section we discuss two audio

broadcasting systems that gain interest as the new standards for radio services: Digital Radio Mondiale and Digital Audio Broadcasting.

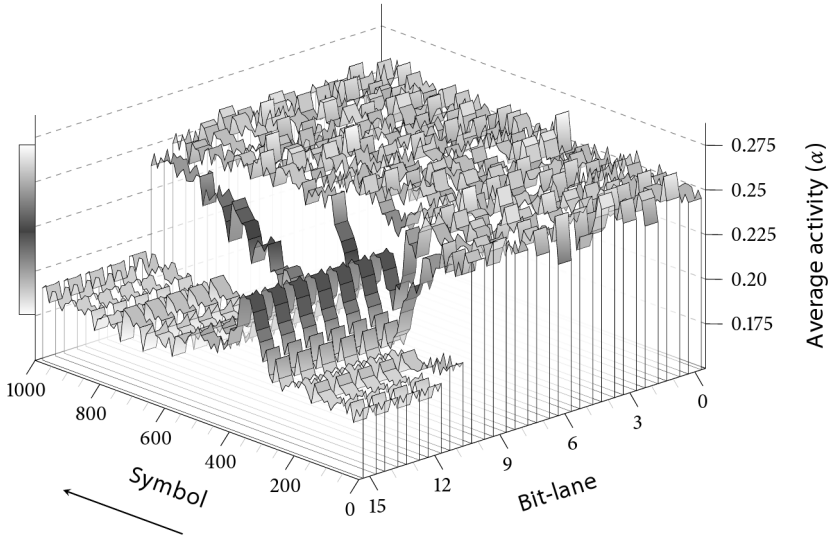
3.3.1 DIGITAL RADIO MONDIALE

Digital Radio Mondiale (DRM) [52] is the upcoming successor of Amplitude Modulation (AM) radio. It provides a flexible and efficient audio and data broadcasting standard. The intention of the DRM standard is to provide Frequency Modulation (FM)-like sound-quality on the AM frequency bands. The AM bands give a large coverage area by a small number of transmitting sites. The application is based on OFDM and MPEG-4 audio coding. The generalized OFDM receiver model (see figure 3.1) is also valid for DRM. DRM is one of the applications used in the 4S project.

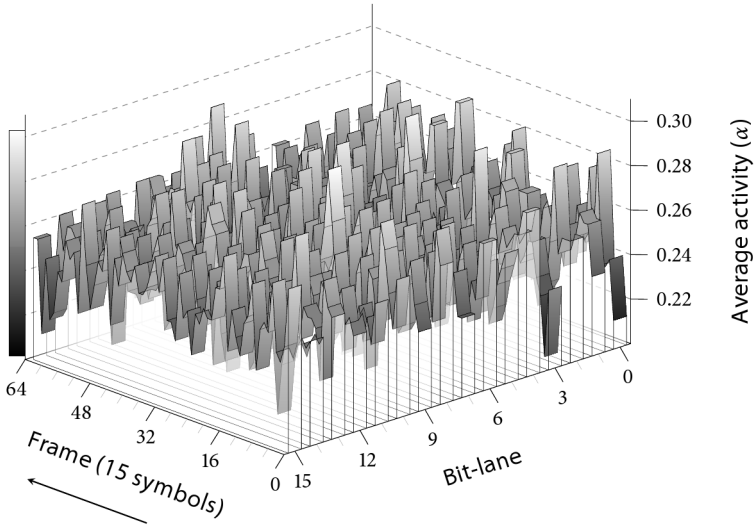
In the baseband processing of a DRM receiver, four demodulation modes (A, B, C and D) have to be supported. The switching between modes is semi-static, because a user will listen to a radio station for seconds or more and the station fixes the mode for a longer time. Each mode has its own characteristics, processing and communication needs. Compared to the other OFDM based standards described in this chapter it has some unique characteristics. First, the DRM signal is a continuous stream of symbols without a known preamble, which requires synchronization on the cyclic prefix and direct channel estimation on the pilots. The pilot versus data cell ratio is therefore much higher compared to the other standards. The ratio ranges from $\approx 1/20$ for mode A to $\approx 1/3$ for mode D. The second unique characteristic is the deviation from 'a power-of-two' number of samples (N_g) in a symbol, which are 288, 256, 176, 114 respectively. The number of samples determines the size of the FFT, such that non-power-of-two implementations are required for this process. Third, DRM uses a multi-level error protection scheme for the transmitted bit stream. After the cell demapping (edge 10 in figure 3.1) the stream splits into multiple protection levels. Each level has its own bit de-interleaver and Viterbi decoder. After the error correction the stream is merged and descrambled. The descrambled bits can be decoded by the MPEG-4 decoder, such that the end-user can listen to the radio broadcast. A small number of the decoded bits contain parameters for the receiver that are sent (edge 14 in figure 3.1) to the global controller of the receiver.

A full DRM implementation [45] is used to perform the activity analysis. This matlab implementation assumes a signal that is sampled with a sampling frequency of 48 ksamples/sec, which corresponds to a maximum bandwidth of 24 kHz. This real signal is downconverted to 12k complex samples/sec. A recorded transmission, which is sampled at 48 ksamples/sec, from the BBC World Service in mode-B with a SNR of 18–25 dB is used [112]. All complex samples are converted to 16 bit fixed point and the decoded source bits (i.e. Main Service Channel (MSC) data on edge 13 in figure 3.1) are grouped in 32 bit words. The message contains either a single OFDM symbol or a full DRM frame (15 symbols) in case of the MSC data. The statistics of the activities α are given in table 3.4(b) and a selection of αs versus time is given in figure 3.4. All other α plots are given in section B.2.

From the α statistics and α versus time we see similar results as obtained with HiperLAN/2. The same average α of approximately 0.25 is present in all edges. The



(a) Frequency correction (edge 3, average over 10 symbols)



(b) MSC hardbits (edge 10)

Figure 3.4 – Message's activity versus time and bit lanes in DRM

Table 3.4 – Communication in DRM

(a) Bandwidth requirements per mode for spectrum occupancy 3 (10 kHz)

#	Edge	Bandwidth [kbit/s]			
		A	B	C	D
1	DDC → Synchronization		750		
2–3	Synchronization → FFT	675	600	550	420
4	FFT → Symbol demapping	268	242	216	162
5	Sym. demap.→ Control	13	40	54	54
6–9	Sym. demap.→ Cell demap.	254	201	162	108
10–11	Cell demap. → Chan. dec.	60	47	39	26
12–13[a]	Chan. dec. → Source dec.	26.6	21	16.6	11

[a] Code rate 0.6, 64-QAM [52, Annex H]

(b) Activity statistics

#	Producing process	Message size	Activity [%]			
		[items]	min	max	mean	std
1	DDC	320	15.6	30.3	23.0	3.1
2	Guard Time removal	256	15.2	31.2	23.0	3.1
3	Frequency correction	256	15.2	30.9	23.0	3.1
4	FFT	256	16.8	32.0	24.2	2.0
7	Channel estimation	207	0.5	33.3	21.4	7.6
8	Equalization	207	18.4	32.4	25.1	1.7
10	Cell demapping (MSC)	2337	19.3	27.0	25.0	0.7
12	Viterbi decoder	263	19.8	30.0	24.7	1.6

edges in the time domain also do not use the full dynamic range, which causes the four most significant bits to have a low and identical α. Due to a DC offset at the AD-converter's input, the most significant bits of the two-complement's values have a lower α compared to the average.

Because a fully encoded MPEG-4 audio stream is used in this analysis, the α of the decoded MSC bits is very close to a random selected bit pattern. The audio decoding will not be analysed in this section. In section 3.4.1 we will discuss the α patterns of a decoded video stream in a more general context (not only DRM).

3.3.2 Digital Audio Broadcasting

The Digital Audio Broadcasting (DAB) standard [49] is positioned to replace the FM radio services. Its coverage is similar to the FM stations, but due to its digital audio

coding it is capable of delivering CD-like quality. Similar to DRM, it is capable of delivering both audio and data services to the end user. The original DAB standard uses the MPEG-1 Layer-2 audio coding scheme for encoding audio and reducing the bitrate.

The standard has several derivatives that use the same physical layer structure but other encoding principles to send more content or other content over the broadcasting channel. The DAB specification version two uses Advanced Audio Coding (AAC) of the MPEG-4 standard to encode the audio [50]. This requires a lower bitrate (64 kbps versus 192 kbps), which enables more audio streams per carrier frequency. The second derivative is DMB, which encodes video streams using the MPEG-4 standard [51]. Frame rates from one to 30 frames per second (FPS) and video formats from QCIF (176×144) to VGA (640×480) are supported. The third derivative is the DAB Enhanced Packet Mode (EPM) which embeds data in Internet Protocol (IP) packets [100]. The IP bits are used instead of encoded audio or video streams as input data for the channel encoding. Using IP packets reduces the effective bandwidth, but increases the possible range of services compared to DMB.

DAB has four transmission modes (I, II, III and IV), each with different characteristics. The main difference between the modes is the number of subcarriers (N) which are respectively 2048, 512, 256 and 1024. In each mode, the OFDM symbols are grouped in a transmission frame, which have a period of respectivily 96, 24, 24, and 48 ms. Each frame starts with a NULL symbol and a reference symbol that are required for the synchronization, and initialization of the frequency and gain corrections. After the reference symbol, three (eight for mode III) symbols, which together form the Fast Information Block (FIB), are required to decode the MSC. The MSC contains 72 (144 for mode III) symbols.

Although DAB has four different modes, the influence on the possible NoC traffic is minimal. Table 3.5 depicts the maximum bandwidths required in the receiver. The bandwidths are determined using the following assumptions: the samples are represented by 16 bit complex values and the de-mapped bits are represented by hardbits. In each mode the sampling rate of the AD-converter is 2.048 MHz, the guard time is $^{31}/_{156}$ of the total symbol time, the symbol time is determined by the number of subcarriers (N) and the number of used carriers is $^{3}/_{4}N$. For each mode the symbol time is proportional to the number of carriers. If the message between the processes are communicated on a symbol basis, the packet size is proportional to N and the frequency of packets is inversely proportional to N. After the de-interleaver block the amount of traffic will become more unpredictable. The transmitter can choose a wide range of channel encoding rates ranging from $^{1}/_{4}$ to $^{8}/_{9}$ and the user can select a specific service (e.g. selecting one or more sub-channels) that is included in a small part of all the transmitted bits. All combinations of channel encodings and selected services will result in another bandwidth requirement.

Table 3.5 – Communication in DAB, bandwidth requirements

#	Edge	Bandwidth [Mbit/s]
1	ADC → Sync.	65.54
2–3	Sync. → FFT	52.51
4–9	FFT → Cell Demapping	39.38
10–11	Cell Demapping → Viterbi	2.46
12	Viterbi CIFs → Source decod.	0.58[a] – 2.09[b]
12b	Reed-Solomon → Source decod.	1.88[c]
14	Viterbi FIBs → Control	0.04[d]

[a] code rate ¼ for all 64 sub-channels
[b] code rate ⅝ for all 64 sub-channels
[c] RS(204,188) code, a sync byte per 187 data items, maximum bandwidth
[d] mode III

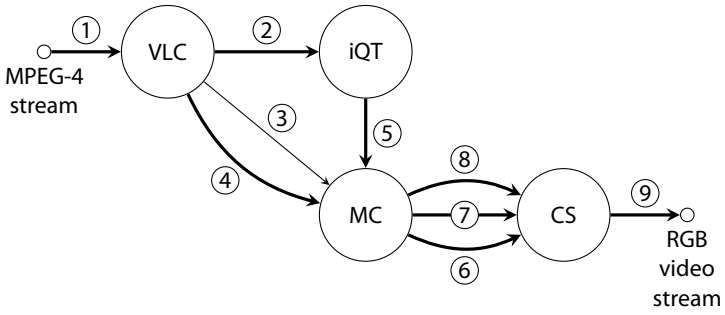


Figure 3.5 – Process graph MPEG-4

3.4 Multimedia

3.4.1 MPEG-4

The Motion Picture Expert Group (MPEG) [28] is a working group of ISO/IEC in charge of the development of standards for coded representation of digital audio and video. It has defined a set of standards that are used for different compression purposes. Each standard consists of multiple parts (e.g. system, video and audio) and each part consists of multiple layers (e.g. to enable encoding at different bitrates).

The first standard (MPEG-1) was used for Video CD coding and included audio layer III, better known as MP3. The MPEG-2 standard is an extension of MPEG-1 and is widely used for Digital Versatile Disc (DVD) and television set up boxes. A much broader standard is MPEG-4 that is suitable for a wide range of application domains. Different profiles are defined in which each selects a subset of the full functionality set of the standard. Due to the large number of profiles, the MPEG-4 standard is suitable for a wide range of demands ranging from profiles for the low bitrates of mobile connections towards profiles that are suitable for studio quality.

The video decoding of MPEG-4 can generally be described by figure 3.5. This graph consists of four processes. The Variable Length Coding (VLC) decodes the Huffman encoded MPEG-4 stream and expands the resulting values based on run-length decoding. The results are the quantized frames in the frequency domain and motion vectors. The quantized frames will be handled by the IQT which performs the inverse quantization and the two dimensional transformation by the inverse DCT. The resulting frames are combined with the motion vectors by the Motion Compensation (MC), which calculates the actual video frames in the YCrCb colour space. The colour space (CS) conversion process converts these values to the RGB color space.

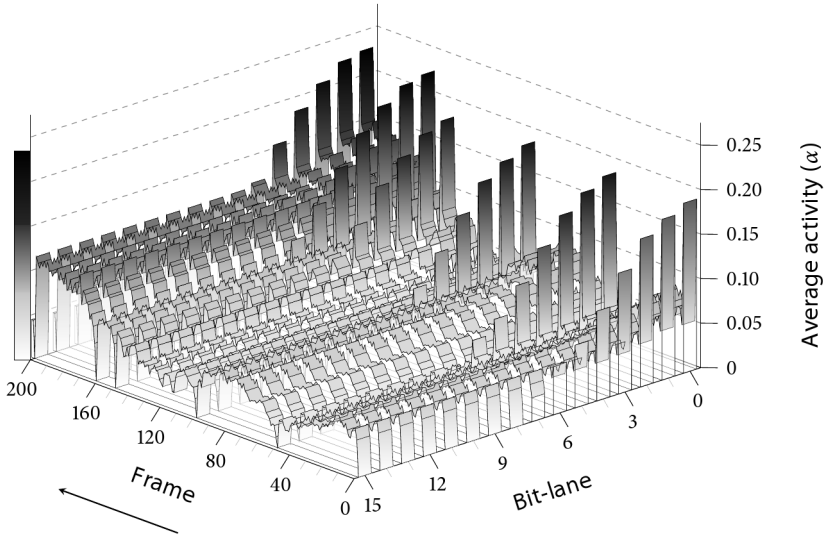
This process graph is mapped onto a heterogeneous architecture for a specific format (the QCIF format, 176×144 pixels) of the simple profile [93, 125], which is suitable for usage on a mobile phone. All processes in the graph perform the processing on a frame by frame basis. Each frame consists of multiple macroblocks, which contain 2×2 blocks of 8×8 pixels. Not all processes need to read and process a whole frame. For example, the IQT and CS can process on a block granularity, and the MC requires at least a whole macroblock (MB) before it can start its processing. The granularity of processing can influence the message and packet size in the network.

Based on the frame format and the frame rate it is possible to determine the required bandwidth for all communication. In case of the QCIF format and a frame rate of 25 frames per second the bandwidth is given in table 3.6. In the implementation of Molderink [93], the edges two to five were realized with 16 bit representation, because the number ranges required more than eight bits. All colour space values and the encoded input use an eight bit value.

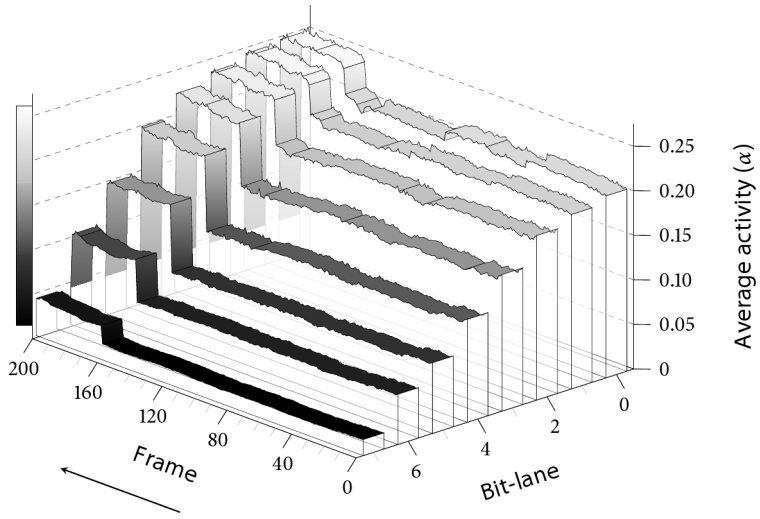
For the activity analysis, a short animated motion picture of 202 frames is used to analyse the activity on the edges. This video fragment has a scene change between frame 162 and 163. It is assumed that a whole frame is communicated in a single message. The statistics of the α values are given in table 3.6 and a selection of α versus time is given in figure 3.6. All other α plots are given in section B.3.

Compared to the activity statistics and the activity over time of the OFDM standards, the MPEG-4 application has a quite different profile. The input has a similar profile as randomly selected values, but all the other edges do not show this profile. The edges to and from the IQT have a very typical profile, which is caused by the I- and P-frames in an MPEG stream. The I-frames have a considerable lower activity factor. The P-frames are characterized by the large number of zeros and small values. This causes a relative low activity of in a single message. The motion vectors (figure B.3(c)) have a higher activity in the bits four, five and seven. The most significant bits have identical activity, which originates from the fact that not the full 16 bit range is used for number representation. This is the case for edges[4] two, four and five. The activity of messages in both colour space domains have a similar profile. Activity decreases with the significance of the bit. Furthermore, the activity is frame content sensitive, as is clearly visible with an overall activity

[4] Bit lanes 8–15 are identical for edge four and five, bit lanes 11–15 are identical for edge two.



(a) IQT output (edge 5)



(b) Luminance messages (edge 6)

FIGURE 3.6 – Message's activity versus time and bit lanes in MPEG-4

Table 3.6 – Communication in MPEG-4, bandwidth requirements and activity statistics

#	Edge	Message size	Bandwidth	Activity [%]			
		[items]	[Mbit/s]	min	max	mean	std
1	IN → VLC	1066.2[a]	0.2	20.7	27.7	24.6	1.9
2	VLC → IQT	39204[b]	15.7	0.2	12.0	1.6	1.2
3	VLC → MC	3168[b]	1.3	0.1	21.4	5.6	6.0
5	IQT → MC	38016[b]	15.2	0.0	22.8	5.7	2.9
6	MC → CS(Y)	25344	5.1	1.5	25.4	13.3	7.1
7	MC → CS(CB)	6336	1.3	2.3	24.7	10.0	6.5
8	MC → CS(CR)	6336	1.3	2.0	21.8	9.2	6.3
9	CS → OUT(RGB)	25344[c]	15.2	1.0	24.3	13.3	7.1

[a] given is average value, minimum: 179, maximum: 7465, standard deviation: 900.0
[b] 16 bits per item (fixed-point)
[c] 3 bytes per item (1 byte per color)

increase after frame 162 (scene change).

3.5 Common Characteristics

Analysing the common characteristics of the communication, broadcasting and multimedia applications we made the following observations:

- All described applications are used on mobile devices, where a human user is involved to start and stop the application. The user will most likely use the application for a time that exceeds a few seconds. This will cause the process graph to be mapped on the system for a relatively long time. The edges of these graphs that are mapped as communication streams on the NoC will be fixed for the same time. Thus the life-time of the communication streams in the system can be considered semi-static.
- The applications consists of manifest loops for the majority of the processes and thus the required computation time per data sample is fixed. This is typically found in wireless baseband processing and audio and video filtering. It does not apply for audio and video (de-)compression and channel decoding, where the required computation time is data dependent (non-manifest). In case of audio and video decompression, it is the first process that decodes the Huffman encoded media stream. Due to the variable length encoding, the time varies to reconstruct the original data that can be used by the IQT and MC processes. After this decoding step, the processing is manifest. In case of channel decoding the processing capacity per data sample depends on the number of errors that have to be corrected. Besides the processing capacity per data sample in channel decoding, the number of data samples varies. This is determined by the coding rate and modulation scheme that are used.

- Input data of most processes arrive at a fixed rate, which causes periodic data transfers between the successive processing blocks. The data is grouped in messages and the message rate is determined by the AD-converter at the input or the DA-converter at the output. The size of the messages is fixed for most communicating edges except for edges around the channel decoding and the input edge of the MPEG-4 decoding.

- The activity of the individual bit-lanes in the messages is analysed in time. We expected a behaviour that is similar to randomly selected values with a uniform distribution. This is true for most of the edges, however not for all applications. In case of the OFDM based applications, it is obvious that the dynamic range of the samples in the time domain has a direct influence on the activity in the most significant bits. In the frequency domain and after demapping to bits, the activity is very close to the expected behaviour. In case of MPEG-4, the activity largely depends on the specific edge in the graph. After the VLC the activity is low due to the large number of zeros in the stream and there is a strong correlation with either the I- or P-frames. After the motion compensation, the activity is related to the spatial redundancy that cause the most significant bits to have a lower activity. The least significant bits have a behaviour that is much closer to random selected values.

We observed that in the described applications the majority of the data streams through the successive processes. This continuous flow of data needs guaranteed throughput, because neither the front-end is allowed to drop data nor the back-end is allowed to stall its output. Beside the main-stream of the communication we foresee a minor part (assumed to be less then 5%) of best effort communication e.g. control, interrupts and configuration data. For example, the DAB receiver, as presented in section 3.3.2, requires multiple reconfigurations of process parameters within a single transmission frame. From an implemented DAB system, the amount reconfiguration data is less than 2 kB per frame [106], which results in a bandwidth of 667 kbit/s. This communication can have more relaxed requirements for the network and hence can use the best effort services.

If we compare the required bandwidth between the processes of the different applications, we observed that this varies widely from several kbit/s (DRM) up to more than 0.5 Gbit/s (HiperLAN/2). Furthermore, the bandwidth requirements change between the successive processes. In case of the communication and broadcasting standards the bandwidth after channel decoding is only a fraction of the initial bandwidth after the AD-converter. In case of audio and video decompression the bandwidth increases through the successive processes.

Each communication edge can be described by the message size and message rate. We analysed some example implementations of applications, where message size (and resulting message rate) were fixed by the designer. The message size is quite often a basic unit that is representative for the application. For example, it is a symbol in the processes before the de-interleaving process in an OFDM system, a variable size MAC message in case of WiMAX starting from the channel decoding processes, and a frame in case of the MPEG-4 implementation. However, it is possible to use

variations of these units, for example a frame can be split in multiple independent MBs.

The overall important characteristic is the life-time of a communication stream. We aim to develop a SoC for a multimedia terminal where we can assume that the data streams are semi-static and have periodic behaviour. This means that for a long period of time subsequent data items of a stream follow the same route. This will last for seconds and more, because a user will listen to its radio or has a phone conversation for a considerable time. However, the control system might change some settings of processes due to changing environmental conditions.

Chapter 4

Router Architectures and Their Realizations

Abstract – *In this chapter we discuss two NoC architectures that are evaluated in this thesis. Both circuit switched and packet switched router architectures are described and the implementations are synthesized and compared in 90 nm CMOS technology. The circuit switched architecture is considerable smaller in comparison with the packet switched architecture. The two routers are also placed and routed, and the resulting area figures are compared with two other packet switched alternatives.*

In chapter 3 we analysed multiple streaming applications. Each of the applications required QoS to communicate the data between the successive processes in the process graph. In a multi-core architecture the edges of the process graph will be mapped onto the NoC that interconnects those cores. In this thesis we explore the NoC paradigm by means of the implementation, integration and comparison of various router architectures.

Unless mentioned otherwise we assume a mesh topology for the NoC architecture. This topology naturally fits in the two dimensional placement of the processing tiles on a chip and provides a simple and regular wire layout. All routers in this topology are identical. Each router has five input and five output ports—four ports

Parts of this chapter have been presented at the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05) - 12th Reconfigurable Architecture Workshop (RAW 2005), Denver, Colorado, USA [PW16], the 15th International Conference on Field Programmable Logic and Applications 2005 (FPL 2005), Tampere, Finland [PW15], and the IEEE Computer Society Annual Symposium on Emerging VLSI Technologies and Architectures (ISVLSI'06), Karlsruhe, Germany [PW11].

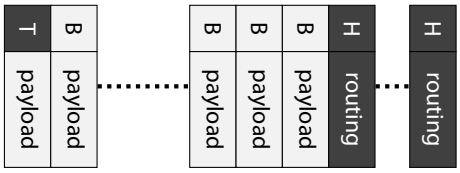


Figure 4.1 – Packet organization in flit-buffer flow control

connect to the neighbouring routers and one router connects to the local processing core. An example architecture is depicted in figure 2.1.

This chapter presents the design and realization of two examples of NoC architectures. Both architectures have a support for QoS for individual communication streams. The first architecture is a packet switched architecture that uses VC flow control and deterministic arbitration to provide QoS. This architecture is presented in section 4.1.

Although packet switching offers more flexibility to handle various traffic patterns, a relative large part of the router consists of buffering resources. These resources are relative expensive on-chip in both area and energy. In contrast circuit switched architectures do not require large amount of buffer capacity. The packet switched architecture is therefore compared with a circuit switched router. Circuit switched connections, by definition, offer fixed QoS. The reduced flexibility, due to the limited number of available circuits, is relaxed by adding serialization of packets and lane division multiplexing. The architecture is presented in section 4.2.

We also compare the two architecture with two packet switched router alternatives. The architecture of those two routers is shortly described in section 4.3. The packet's organization and placed and routed area requirements of the four routers are also presented in this section. In section 4.4 we draw some conclusions.

4.1 Packet-switched NoC Evaluated

In this section we present the details of the packet-switched NoC that is further examined and compared with other packet switched alternatives in the next chapters. The architecture consists of a five port router with VC flow control and that is able to offer QoS. The packet's organization is as depicted in figure 4.1, where the darker blocks contain information for the routers' control logic. The VC flow control belongs to the class of flit-buffer flow control that is described in section 2.4.2. For the remainder of this thesis we refer to the architecture as *GuarVC* router.

The initial architecture was proposed by Kavaldjiev [78]. The major contribution of his work was the change from a traditional symmetric crossbar switch to an asymmetric design, which enables the use of VCs to offer QoS. This increases the crossbar design area, but simplifies the arbitration. However, both control and data structures are mixed in the design's description. This makes it difficult to determine the individual resource costs, area and power, of data and control parts.

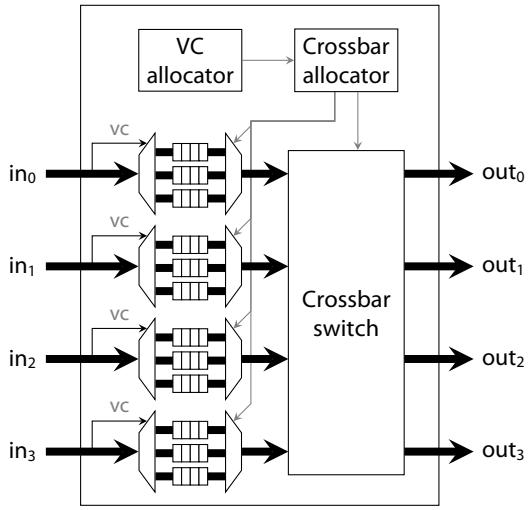


Figure 4.2 – VC router architecture with symmetric crossbar

4.1.1 Traditional virtual channel router

Figure 4.2 depicts one of the frequently used architectures of an input queued router with VC flow control. It consists of independent input queues per port and VC, a fully connected crossbar with the number of input and output ports equal to the number ports of the router and two allocation controllers. The request and acknowledge lines between the queues and two allocators are not included in the figure.

The individual flits of a packet will arrive from one of the input links. Each flit is accompanied with its VC identifier, which is used to store the flits in a specific input queue. The allocation of an output VC and physical channel's bandwidth is performed by respectively the VC allocator and crossbar allocator.

A new packet will request a VC on an output link from the VC allocator using the routing information carried by its head flit. Which output link (and VC) is assigned depends on the type of routing algorithm used. The VC allocator implements the part of the routing algorithm that is performed by the router. The output link can be determined by the destination and a deterministic routing scheme, e.g. XY-routing, encoded in the header flit in case of source routing, or completely adaptive based on, for example, the current load on a specific port.

After output link and VC are assigned, the individual flits of a packet each will request a slot on the output link assigned. These slots and the interconnection configuration of the symmetric crossbar are allocated by the crossbar allocator. The allocation function tries to optimize utilization, throughput and other criteria of both switch and output links. At the granularity of a single flit it will decide: 1) which input VC is assigned to the input of the crossbar and 2) which crossbar input-output connections are valid.

The input queues are of limited depth and buffer overflow should be prevented at all times. Therefore, information on available buffer space is communicated downstream, i.e. the neighbouring routers connected via the input links. We use the general terms *downstream* and *upstream* to denote, respectively, the direction from source to destination and the reverse direction.

Information on the buffer space can be either by indication of status (full or not-full), which is the cheapest as the router downstream does not require any extra administration, or via credits. The latter signals the router downstream when buffers space becomes available. Downstream, the router has to administer the available buffer space at the upstream router and to prevent sending too many flits. This administration requires extra resources, but also enables the allocation algorithm to make better decisions.

4.1.2 GUARVC ARCHITECTURE

The GuarVC router, as originally proposed by Kavaldjiev [78], supports QoS for individual packets. The QoS that is noted by a packet is determined by the network resources that are allocated to the packet given that the packet follows a deterministic route. These resources are the router's *input queues* and the physical channel's *time slots*, i.e. bandwidth. If we increase the available resources or decrease the response time of the arbiters on a request of a resource, the throughput of packets will increase and the packet's latency will decrease.

For most packet switched routers the average packet latency and throughput under certain load conditions are known. For example, on average, the packets observe a latency of 20 cycles, but a specific packet might have a latency of 180 cycles, due to local congestion of the network. In other words the packets receive on average certain resources, but the exact allocation of resources is not known on a packet-by-packet basis.

To offer hard QoS guarantees, the allocation of resources has to be deterministic such that guarantees can be given and are known before packets enter the network. The deterministic allocation of resources also requires a deterministic route of the packet. Deterministic routes can be achieved with a fixed routing algorithm for the whole network, for example XY-routing. The packets will carry the destination address and the individual routers determine the packet's route. Or, as for the GuarVC network, source routing is applied where the packet itself contains its routing information. Source routing is more flexible compared to a fixed routing algorithm, because multiple routing algorithms can be used and adapted at run-time. The routes of packets between specific source- and destination tiles is determined centrally by the CCN, such that local contention is prevented and global optimization is possible. The source tile encodes the route of a packet with information it obtains from the CCN.

Besides deterministic routes, the allocation of the physical channel's bandwidth and buffer allocation has to be deterministic to offer QoS. In the GuarVC router we have chosen to reserve a specific VC per physical channel to packets that have the same end-to-end route. This static allocation of VCs, i.e. virtual circuit buffering, is

determined by the CCN for the whole NoC which results in a deterministic queue allocation by the VC allocator. The reserved VCs are not shared by other packets that follow other routes, which prevents head-of-line blocking for the Guaranteed-throughput (GT) packets.

In eq. (2.2) we presented the minimum latency of any packet that is transported by a VC flow control network. This equation contains two variable delays: t_r and t_w. These delays respectively describe the time required for the packet's header flit to pass a single the router and the time to transport a packet's flit between two routers. We can bound these both delays, given deterministic allocation of both the VCs and physical channel's bandwidth for a GT packet. In the GuarVC router, as presented in this thesis, the bounded delays are respectively $t_{r_{max}} = 4$ cycles and $t_{w_{max}}$ equals the maximum number of allocated VCs per physical channel along the packet's route. The latency of a GT packet is then bounded by:

$$H \cdot t_{r_{min}} + t_{w_{min}} \left\lceil \frac{L}{W} \right\rceil \leq t_{GT} \leq H \cdot t_{r_{max}} + t_{w_{max}} \left\lceil \frac{L}{W} \right\rceil \tag{4.1}$$

where H is the number of hops between the packet's source and destination, L the total length of a single packet in bits and W the width of a link. BE packets will share a VC and therefore have no bounded $t_{r_{max}}$ due to non-deterministic VC allocation.

The deterministic allocation of physical channel's time slots can be achieved with a deterministic behaviour of the crossbar allocator. In the symmetric VC router architecture (see figure 4.2), the crossbar allocator determines the configuration of both the VC multiplexer and the crossbar to transport flits of the input queues to the requested output channels. This require both arbitration per input port over the non-empty VCs and arbitration per output port over all its requests.

These two stages of arbitration are required, because each crossbar input port can transport at most one flit per cycle and the same holds for the crossbar output port. Achieving a 100% throughput and fairness is studied by McKeown [90] for these kind of symmetric input queued architectures, where network items are sorted by the output port in unique queues (i.e. virtual output queueing). He proposes the *iSLIP* allocator, which consists of round-robin arbiters for both stages and an update of priorities in the first stage arbiters based on the grants in the second stage. This and other algorithms are compared by Schoenen on both hardware complexity and latency performance [116]. Although 100% throughput and fairness can be guaranteed for iSLIP and others, the delay and latency of individual packets and flits are not bounded. Deterministic bounds can only be given with a static allocation table, where for dynamic allocators, the delay can only be statistically described, i.e. the percentage of items that violate a specific delay bound [117].

To overcome this two level allocation we removed the multiplexing of input queues and connected the data port of the queues directly to the crossbar. The multiple queues of a single port can now be allocated in parallel to different output ports. The modification is similar to the 2N×N crossbar of the Æthereal router [57]. The modified architecture is depicted in figure 4.3. Similar to the previous figure we left out the request and acknowledge lines between queues and allocator. Details on

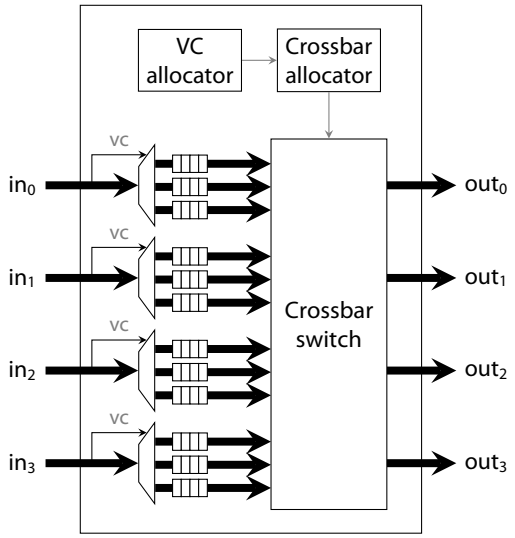


Figure 4.3 – The asymmetric GuarVC router architecture

the specific implementation of the individual blocks and modifications compared to the implementation of Kavaldjiev are described in the next section.

Kavaldjiev performed a comparison of this symmetric versus the asymmetric GuarVC architecture in a 0.13 µm TSMC technology library. The architecture has a configuration with a 16 bit data-path, two flits deep input queues, four VCs per input port and five input and output ports. The resulting area of both implementations are presented in figure 4.4. Despite the larger crossbar, the allocation becomes much simpler, because all individual arbiters of output ports can operate independently. Were in the symmetric architecture the crossbar allocation requires more then 50% of the router's area this is less then 2% in the asymmetric architecture. The crossbar is larger, but consist of both control and data path components. In this thesis we will show the area results of the architecture were those two components are split. The control path part of the crossbar is actually part of the crossbar allocation.

4.1.3 Implementation

In this section we describe the implementation details of the GuarVC router architecture. Compared to the realisation, which is described by Kavaldjiev, we adjusted the architecture at some points to increase its performance, reduce the area and get a better separation of control and data path to perform a more detailed characterisation.

The crossbar designed by Kavaldjiev was a combination of tri-state buffers to transport the flits, and distributed comparators and logic chains of OR and AND gates to handle the request and acknowledge signals between input queues and allocators.

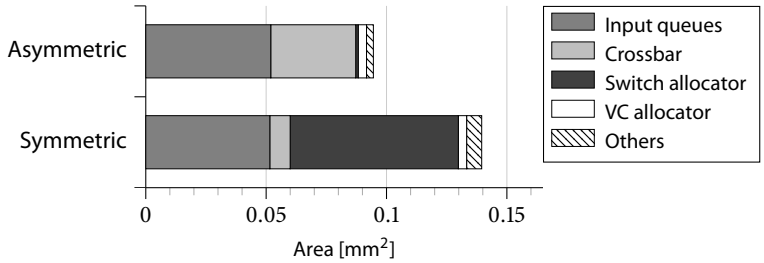
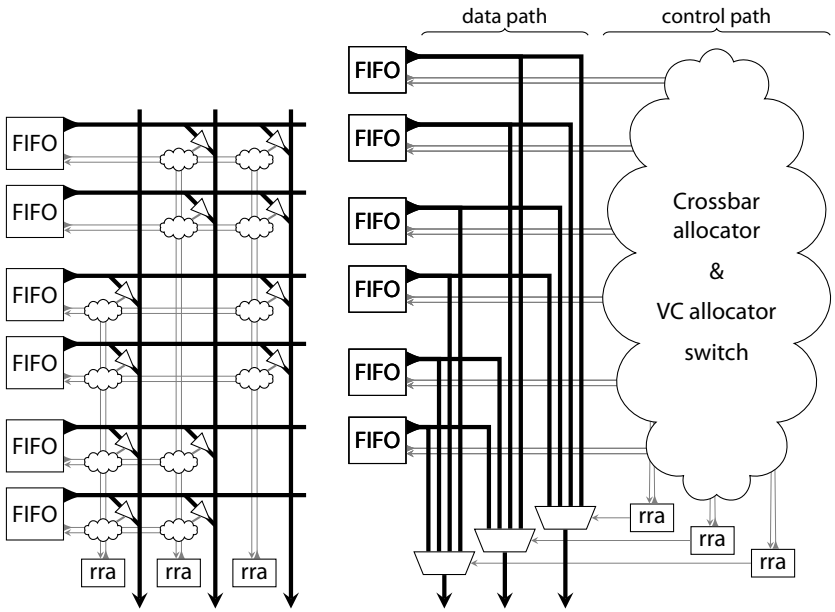


Figure 4.4 – Area figures in 0.13 µm of a symmetric and asymmetric architecture of a virtual channel router, adjusted from [78]



(a) Distributed control and data path

(b) Control and data path separated

Figure 4.5 – Schematic representation of the modification of the GuarVC router's crossbar

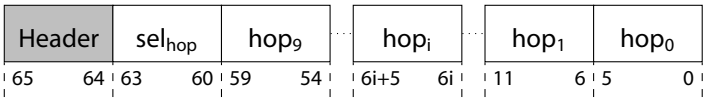


Figure 4.6 – Header flit format for 64 bit data path

Each input VC can be connected to any output port apart from the port on the same side of the router (i.e. no flit U-turns are allowed). A schematic representation of this crossbar is depicted in figure 4.5(a). This crossbar is described as a single entity, which makes it difficult to characterize the control and data path separately.

The data crossbar is replaced by multiplexers, because synthesis tests in 90 nm technology showed a significant smaller area for a multiplexer crossbar compared to a tri-state crossbar. The control part of the crossbar is extracted and combinations of comparators and logic chains are replaced by multiplexers which describe the same functionality. This enabled the synthesis tool to optimize the design, because we use a less restrictive description. A schematic representation of the updated crossbar is depicted in figure 4.5(b).

The synchronous elements in the design are examined too, such that fine-grain clock gating is possible. The effect of this modification on area and power is demonstrated in respectively section 4.1.4 and section 6.4. The last adjustments is the organization of the header flit as described in detail in the following section. It increases the router's performance as will be shown in section 5.1.1 in more detail.

Flit Format

As described in section 2.4, a packet in a virtual channel flow control NoC consists of flits. The flit in the GuarVC network consists of a single data item and is controlled and transported in a single cycle from the input queue to the input queue of the router downstream, when the data path resources are allocated. The data item consists of W bits and is appended with a two bit flit type (*Header*, $Payload_A$, $Payload_B$, and *Tail*). Two types of payload are supported to fully utilize the encoding.

The routing information is encoded in the header flits of the packet and the data flits and tail flit can carry payload for the destination. In each router along the route the first header flit is examined and the required six bit for routing and allocation information is extracted. The $\mathsf{sel}_{\mathsf{hop}}$ field is decremented each hop and the field is used to select the corresponding six bit hop_i field per hop. Due to this six bit field, a single header flit describes at most $\lfloor W/6 \rfloor$ hops of the packet's route. The flit is discarded after this number of hops, i.e. the select field equals zero. This will gradually reduce the packet length until the packet reaches its destination. The header flit format for a network with a 64 bit data path is shown in figure 4.6. In the architecture as described by Kavaldjiev the number of hops per header was limited to one. The content of a hop field is described together with the VC allocation block.

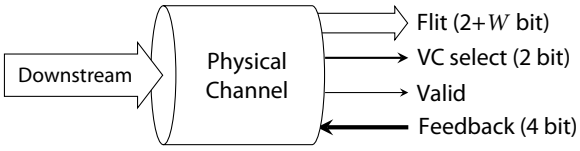


Figure 4.7 – Signals on the physical channel interface

Physical Channel

The physical channel connects an output port of a router with the input port of the successive router. For the GuarVC this physical channel consists of wires without registers, such that the flits can traverse the channel in a single cycle. The configuration of wires is presented in figure 4.7. The majority of the wires is used to transport the flit, consisting of data and type. Some extra wires are required to signal the input port that the data is valid, i.e. a link can be idle, and which of the four VC should store the incoming flit. In the reverse direction, a single wire per VC indicates the feedback information for the flow control.

Input Buffer

Each flit that arrives at the input port is marked with a specific VC identifier. The flit is transported to its corresponding input queue. Although each queue has a limited depth, a flit cannot be dropped, because the flow control will prevent the flit to be transmitted from the router upstream if the queue is full. Each input VC can be either *active* or *inactive*.

The flit at the head of the queue is examined and either:

- a request to the VC allocator is signalled if the input VC is not paired with an output VC, i.e. the VC is not active. When the VC allocation is acknowledged, the VC becomes active. The header flit will be either removed from the queue if all routing information is used (the $\mathsf{hop}_{\mathsf{sel}}$ field equals zero) or a request to the crossbar allocator is signalled if the header flit has to be forwarded downstream. A VC becomes inactive if the crossbar allocator acknowledges the request of the tail flit of a packet.
- a request to the crossbar allocator is signalled, if the input VC is active, to obtain a slot on the output port. If the switch request is acknowledged by the crossbar allocator, the flit is removed from the queue.

VC Allocation

The VC allocator monitors all requests from all inactive VCs. The requests are accompanied with the information of the $\mathsf{hop}_{\mathsf{i}}$ field selected from the header flit. This describes the routing information for this specific router as source routing is used by the network. This field describes: 1) the requested output port and output VC by the packet (four bit) and 2) a unique Identifier (ID) for this port and VC

combination (two bit). A specific port and VC are requested, such that it is known by the CCN which resources can be allocated to which packet. This makes it possible to prevent contention of resources by two competing packets. The input VCs that request an active output VC are masked, i.e. not arbitrated, by the VC allocator. They will be unmasked if a tail flit has traversed that output VC. The not masked requests are arbitrated on a cycle-by-cycle basis.

The CCN manages the individual VC such that packets that require QoS solely can use a specific VC of a specific physical channel. However, BE packet do not require guaranteed services and can share a VC with other packets. For BE packets, the packets competing for the same output VC are tagged by the source with a unique ID. Because the ID field is two bits, at most four packets can compete for the same VC. The VC allocator uses a two bit free-running global counter and acknowledges all not masked requests that have a matching ID at the current clock cycle. The GT packets will obtain an acknowledgement within four cycles, i.e. one cycle of the counter, because they do not compete with other packets for the same VC. The BE packets might observe a longer delay, because other packets use the VC for a short time and therefore no guarantee on the latency of a single packet can be given. Since, at any time, the counter contains an arbitrary value, fairness is provided.

Crossbar Allocation

All active input VCs with non-empty queues will send requests to the crossbar allocator. This block arbitrates which flits can traverse the switch to the output port and connected router downstream. Because all input queues are connected to the switch (see figure 4.3) the crossbar allocator only has to arbitrate over the output VCs of each output port. Due to a fully connected crossbar switch, each output port is independent. Therefore, the crossbar allocator consists of independent arbiters, one per port. In Kavaldjiev's realization of the control that interconnects input queues with the crossbar allocators, i.e. control crossbar, was mixed with the data crossbar that transports the flits. For this thesis we split these two fully independent crossbars, such that both area and power can be analysed in greater detail.

The arbiters have to divide the total bandwidth of the physical channel over the requesting input VCs. The arbiter should have a bounded latency per request which makes it possible to bound the overall latency of a packet that uses a connection of unshared VCs. The most simple arbiter that offers this bounded delay is a round-robin arbiter [61].

A round-robin arbiter assigns the last granted request with the lowest priority. This request will be served again after all other VC with requests are served once. Another simple arbiter that has a bounded delay is a static schedule were each VC has a fixed slot. The disadvantage compared to round-robin are the wasted slots that are assigned but not consumed by idle or inactive VCs.

The maximum delay for a round-robin arbiter between two successive grants of a single VC is therefore bounded to the maximum number of active requests of other VCs of that output port. During this delay, all other active requests will be granted once by the arbiter. The number of active requests is bounded by the

number of VC that can be allocated, which is known by the CCN. Therefore, the CCN is able to divide the physical channels bandwidth to a number of packets that use the channel. The bandwidth can be divided in portions of either 25%, 33%, 50% or 100% depending on the maximum number of VCs that can be active at the same time (respectively four down to one).

Flow Control

Although the latency of individual flits is bounded, it is very well possible that one physical channel is used by multiple packets and others only by one. A single packet observes different bandwidths for different channels, which creates congestion at specific queues. Because a single queue might not be able to store a whole packet, the traversal of flits has to be stalled to prevent buffer overflow. Each queue therefore includes a nearly full signal that is connected to the feedback wires of the physical channel. The nearly full signal of a VC is activated if the free buffer space of its input queue is at most one flit.

The feedback is therefore a number of wires equal to the number of virtual channels. The crossbar allocator of the router upstream uses this feedback to mask the requests for VCs with a full queue. This creates a stop-go flow control and prevents the traversal of flits destined to full queues.

4.1.4 Synthesis Results

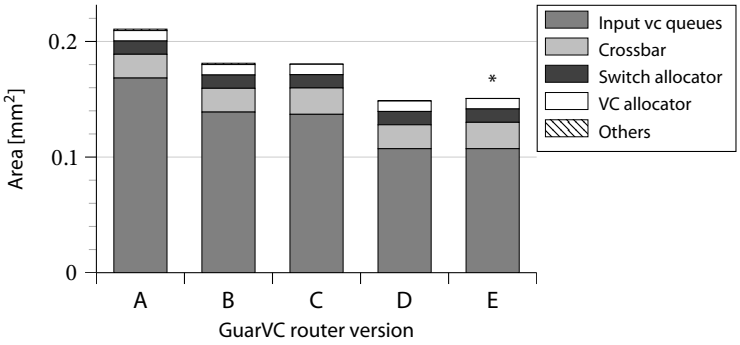
The design of the GuarVC router, as described in the previous section, was described in VHDL and synthesized to ASIC technology. The flit width, number of router ports, and the buffer depth per VC are left as model parameters. Furthermore, the design is synthesized with and without clock gating. The synthesis was performed with Synopsys[1] Design Compiler using a TSMC 90 nm library.

Power Optimizations

Before we present the results of the effects of these parameter values on the area requirements, we first describe some architecture changes that are also analysed during the power evaluation in chapter 6. The resulting area requirements of the five router architectures are depicted in figure 4.8. Each router architecture has five ports, four VCs per port, a VC queue depth of four flits, and a flit width of 64 bit.

We started with an implementation of the GuarVC router that was a modified version of the architecture implemented by Kavaldjiev [78]. The crossbar is split in a control and data path part, header compression is included and the tri-state crossbar is replaced by a multiplexer version. This architecture is synthesized for a flit width of 64 bit and both without (version A) and with (version B) clock gating. Clock gating elements are automatically inserted for the crossbar allocators and the VC input queues. For each VC queue a single gating possibility was detected. The

[1] The software described in this document is furnished under a license from Synopsys International Limited. Synopsys and the Synopsys product names described herein are trademarks of Synopsys, Inc.



Version	Clock gating	VC queue enable	Output latched
A	no	coarse	no
B	yes	coarse	no
C	no	fine	no
D	yes	fine	no
E	yes	fine	yes

Figure 4.8 – GuarVC router optimizations

area reduction of 14% for the gated design is caused by more simple synchronous elements that can be used due to the clock gating element in the clock tree.

As will be shown in chapter 6 additional power can saved by gating the VC per flit position. We modified the input queues VHDL description such that this is detected by the Synopsys Power Compiler tool that inserts the clock gating elements. Both the non-gated (version C) as well as the gated (version D) version of this architecture showed an area reduction of respectively 15% and 18% in comparison with the coarse grained enable of the VC queues in version A and B.

During power analysis, we observed a lot of switch activity at the output port of the routers within a single clock cycle. This was caused by the activity in the first part of the clock cycle of both data and control ports of the crossbar. To reduce this activity on the outgoing link, we included a transparent latch at the output of the crossbar. The latch of an output port is only active during the second half of the clock cycle and if the crossbar allocator has granted a request for this port. This last modification of the router's architecture (version E) resulted only in an overall increase of 1.5% in area in comparison with version D, but a considerable reduction in the power consumption as will be shown in chapter 6.

Parameter Analysis

For the parameter analysis of the router model we synthesized the version E router architecture for a number of parameter values. The reference design is a five port router with four VCs per port, a VC queue depth of four flits, and a flit width of

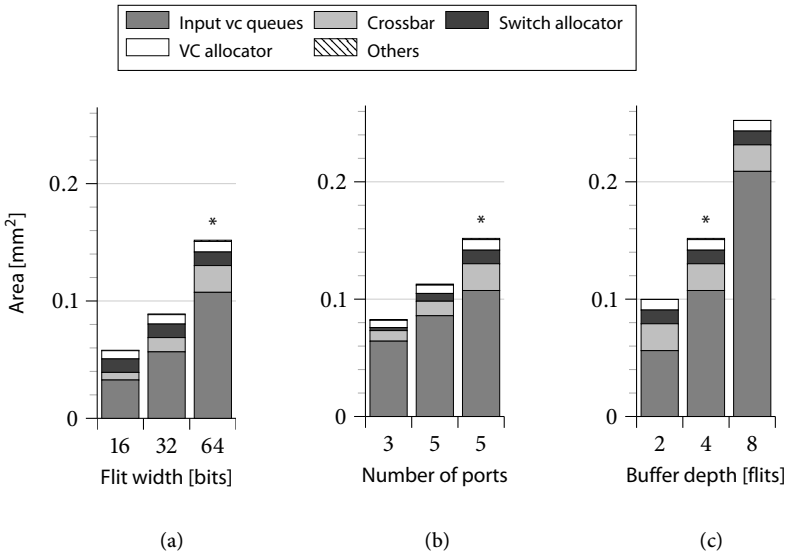


Figure 4.9 – Synthesis results of various GuarVC router designs. Per graph a single model parameter is changed. The values for the reference design (indicated by an asterisk) are five ports, four VCs per port, four flits per VC queue and 64 bit per flit. For all six designs clock gating is enabled.

64 bit. Furthermore, automated clock gating is enabled.

Figure 4.9 depicts the variations of the three model parameters in three different graphs. The reference design is included all three plots and indicated by an asterisk. For the reference design 70% of the router's area is consumed by the input queues of the 20 VCs. The largest part of the queues are the fifo buffers and a small part is required for the control of these buffers. Of the remaining area, 15% is required for the data crossbar, and the crossbar allocation and VC allocation have roughly the same size. The largest part of both switch and VC allocation is required for the control crossbars that exchange status signals (request/acknowledge and occupied/free) between the input ports and the arbiters.

In figure 4.9(a) we reduced the width of the flits to 16 and 32 bit. The data path components—input buffer and crossbar—reduce in size. We can compare the most left bar in the graph with the area figures of the original packet switched router as presented in figure 4.4. In this figure the area for the switch and VC allocators are hidden in the crossbar's area. In the new implementation control and data path are separated. The relative area of allocators and crossbar has decreased due to the change from a tri-state to a multiplexer crossbar. For a flit width of 16 bit, the control crossbars of the allocators are even larger compared to the data crossbar of the flits.

In the middle graph we changed the number of input and output ports of a

single router. These routers with fewer ports can be, for example, instantiated at the boundaries of a mesh topology or to construct a ring topology. The control overhead in routers with a lower number of ports reduces considerable, because the required crossbars scale quadratically with the number of ports. For the three port router almost 80% is required for the input buffers.

In figure 4.9(c) we varied the depth of the individual input buffers. This has only an influence on the area of the VC queues. The area of this component almost scale linear with the depth of the buffer. There is a small additive component, which does not scale with the buffer depth, that stores the input VC's state and some control logic. Even for a minimum buffer depth of two flits the VC queue's area contributes to half the router's area.

4.2 Circuit-switched NoC Evaluated

In the previous section the packet switched router (GuarVC) is described. Although it can provide both BE and GT services and it is initially optimized to minimize buffering, its area is still considerable. A large part of the area is consumed by its input buffers. Therefore, we also consider a NoC based on the idea of circuit switching. The section starts with a motivation for considering this less flexible technique for a NoC after which the architecture, implementation and synthesis results are discussed.

4.2.1 Circuit-switching revisited

The application analysis of section 3.5 shows that the amount of expected GT traffic is much larger compared to the amount of BE traffic. Fundamentally, sharing resources and giving guarantees are conflicting. Efficiently combining guaranteed traffic with best-effort traffic is difficult and load variation or a priority change of one type will influence the other [107]. By using dedicated techniques for both types of traffic we aim at a reduction of the total area and power consumption.

For the GT traffic we propose a reconfigurable circuit switching network that creates dedicated connections between two processing tiles. The reasons for reconsidering circuit switching instead of using packet switching are:

- The flexibility of packet switching is not always required, because in our application domain data streams are fixed for a relative long time. Therefore, a connection between two processing tiles is required for a long period (e.g. seconds or longer). This connection can be configured once by the CCN. Although setting up a circuit might take a longer time compared to a packet switched alternative, this overhead will be negligible when the life time of the specific circuit is long enough.
- Large amounts of the traffic between tiles will need a GT, which can be easier guaranteed in a circuit-switched connection.
- Current SoCs have a large amount of wiring resources that give enough flexibility for streams with different bandwidth demands.

- Circuit switching eases the implementation of asynchronous communication techniques, because data and control can be separated. A control free pipelined asynchronous data stream does not require much design effort. However, asynchronous design is not the focus of this thesis and therefore not discussed in further detail.
- The circuit switching has a minimal amount of control in the data path (e.g. no arbitration) and only a pipeline buffer per router. It is expected that this increases the energy-efficiency per transported bit and the maximum throughput.

Further, we see some benefits when GT traffic has to be scheduled:

- Scheduling communication streams over non-shared channels is easier, because by definition a stream will not have collisions with other communication streams. The scheduler only has to choose a route of free channels that meets the bandwidth constraints. The Æthereal [43, 108] packet switched routers have large interaction between timeslots of two successive routers (both have to have a successive free timeslot for a single GT stream). Determining the static time slots table requires considerable effort. However, Hansson et al. [63] demonstrated that a greedy algorithm can reduce this effort.
- Because data-streams are separated, collisions in the crossbar do not occur. Therefore, we do not need buffering and arbitration in the individual router. An established channel can always be used.

However, circuit switching has some disadvantages:

- An additional control network is required to setup connections. This setup will require more time, because the full circuit from source to destination has to be allocated before any data can be injected by the source.
- Once the circuit is setup the full bandwidth can be used for a specific stream. However, when the stream is not active temporarily the allocated channel bandwidth can not be used by others.

4.2.2 Architecture / Design

Using the benefits of the circuit switching and required application demands we defined a reconfigurable circuit-switched router. For the remainder of this thesis we refer to the architecture as *CS* router. The implementation is described in section 4.2.3. In this section we describe the principles of the architecture.

The idea for a circuit switching architecture is to establish a physical connection between the source and destination. This connection is a set of wires between the routers which is reconfigurable at run-time. By establishing this connection we have a circuit switched network that does not need the buffering and control as needed in a packet switched network, where buffering is needed as contention can occur. Circuit switching will immediately meet the QoS constraints of an application, because no contention can occur. Buffering, i.e. pipelining, is only required to limit the maximum worst-case delay that limits the clock frequency of the NoC. Control

in packet switching is needed, because control information is embedded in each packet. This control information can, for example, describe the route of the packet or it's priority. In our circuit switch approach we completely separated the control from the data-streams.

The main disadvantage of circuit switching is that a reserved physical link can not be used for other connections, where as in packet switching the physical link is time multiplexed among the various packets. If the stream does not use the full bandwidth of the physical link it actually limits the number of active streams in the system. Therefore, we do not want to claim all the available wires between two routers for a single connection. The wires of a link are grouped in smaller channels (e.g. four groups) called lanes. The bi-directional link between two routers consists of multiple uni-directional lanes (e.g. two times four lanes). This increases the flexibility.

The width and number of lanes are adjustable parameters in the design. The width of the lanes is assumed to be uniform over the whole network. They can be adjusted at design-time of the SoC to meet the flexibility and bandwidth requirements of the aimed applications. For example, if more streams or bandwidth is required for the north-south direction their number of lanes can be increased. The tables of chapter 3 can be used to determine the width and number of lanes. At run-time, the assignment of different number of lanes to a stream can overcome different bandwidth requirements of the individual streams.

The control of the individual connections between input and output lanes of each router is not controlled by the packets itself. This would make it a packet switched like network and the intention is to completely separate data and control. Therefore, the specific connections are established and broken down via a separate control interface of the router. This interface can be connected to another communication network, that also handles the BE traffic.

Comparable alternatives

Two similar circuit switched approaches are proposed by Wiklund and Liu [137] and Leroy et al. [86]. Wiklund and Liu proposed the SoCBUS that presents the circuit switching as a packet connected circuit. A single packet reserves a circuit for the duration of the packet and tears down the circuit as well. Because they claim the complete bandwidth between to routers they have large latencies for setting up a new circuit. This latency is caused by the blocking of other existing circuits. Their solution is to calculate a static (predetermined) timeslot table for all the occurring data streams in time to solve the latency problem.

Leroy et al. proposed a nearly similar approach as described this thesis, which is introduced in the paper as SDM. In the circuit switched architecture described in this thesis the number of bits per lane are fixed at design time for all lanes. Leroy et al. did not restrict the number of bits per lane which makes it possible to change the width of a specific lane at run-time. Because the lanes are not fixed and a non-blocking crossbar is required, the number of crosspoints of the switch inversely grows with the minimum granularity of lane width. For example a 5×5 crossbar

with 32 bit wide links and a granularity of one bit lane requires a 160×160 switch. For these size switches it becomes interesting to implement the switch not as a simple crossbar, but use a Multistage Interconnection Network (MIN). Leroy et al. have chosen to use a Beneš switch. This is a rearrangeable non blocking switch that can be updated while active if the switch is not pipelined. For end-to-end flow control they assume a credit-based solution that uses a single bit circuit in the opposite direction. Exact details on the flow control are not given.

4.2.3 IMPLEMENTATION

In this section we describe the implementation details of a specific realization of the circuit switched router. For the first implementation we assume that the router is based on a 16 bit link with four lanes per link. Four lanes of four bits and a tile interface of 16 bits have been chosen to make a fair comparison with the four virtual channel configuration of the packet-switched alternative. The reconfigurable circuit-switched router consists of three major parts: the *data converter*, the *crossbar* and the *crossbar configuration*. Figure 4.10 depicts the block diagram of the router. In a mesh based topology the router has five bidirectional ports where one port is connected to a processing tile and four ports via a bi-directional link to their neighbouring circuit-switched routers.

The router has two major differences compared to a packet switched alternative. Instead of extending the payload with additional routing information, which will request the necessary crossbar configurations for the circuit, the crossbar configuration is controlled via an extra configuration interface. Second, the interface to the tile is different from the interface to the other routers. It is assumed that a tile will produce a single item per clock cycle. The item is serialized and transported by the network over the pre-established circuit.

Data Converter

The intermediate circuit-switched routers do not interpret the information in the data-packets. All items are always forwarded to the successive router on the route and at the last router the data is forwarded to the data converter. Because the network does not have fixed slots for data-packets, the item can arrive at any time in the data converter. Per data item of 16 bit a small four bits header is included to signal the data converter for a new data-packet.

This header is solely used by the receiving data converter and not monitored or adjusted by the intermediate routers. The receiving data converter detects a packet by the valid bit in the header. The header is created in the data converter of the source and combined with a 16 bit data-word of the tile. The result is a packet of 5×4 bits, which can be transported over a lane. The organization of this 20 bit packet is given in figure 4.11.

Combining the delay of a single cycle per router and the delay due to the serialization of the data we can calculate the latency of a packet after a circuit is

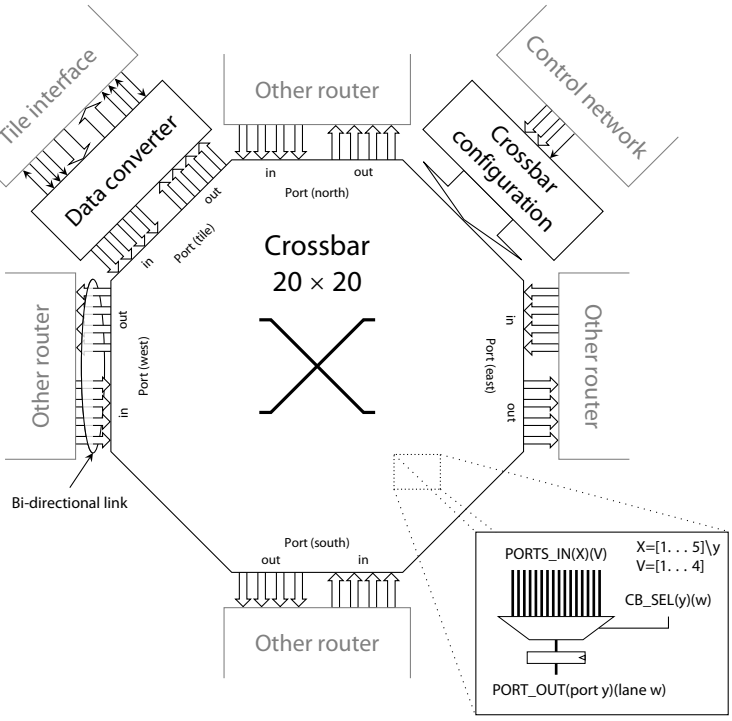


Figure 4.10 – Block diagram of the CS router

established. Due to a congestion free route, this is a deterministic latency:

$$t_{CS} \leq 2 + H + \frac{5}{4}\left\lceil \frac{L}{W} \right\rceil \tag{4.2}$$

where H is the number of hops between the packet's source and destination, L the total number of bits in a packet and W the width of a lane. The two additional cycles are caused by both data converters along the packet's route and the 5/4 ratio represents the header overhead per data-word.

In case the data item width and lane width increases to, for example, respectively 64 and 16 bit the header contains some unused bit positions. Those can be used to add some extra information per data item, or the four bit header can be placed in parallel to the four times 16 bit data of the packet. The parallel version increases the total throughput of the network, but also extends the area of the router.

The small lanes of the network and crossbar are connected to a tile interface via the data-converter. Figure 4.12 depicts a data-converter that converts the 16 bit data to the width of the lanes and visa-versa. The tile interface is compatible with the packet-switched alternative of section 4.1. We define the division in lanes as *lane division multiplexing*. Each lane can be used for a different physical connection between processors.

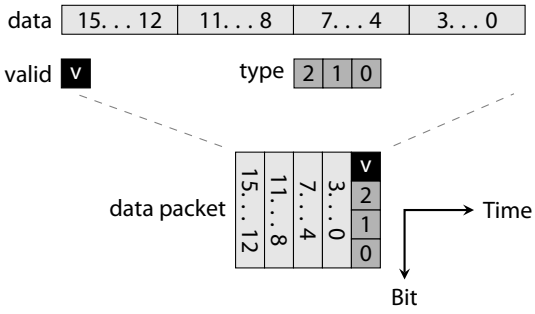


Figure 4.11 – Organization of the header and data

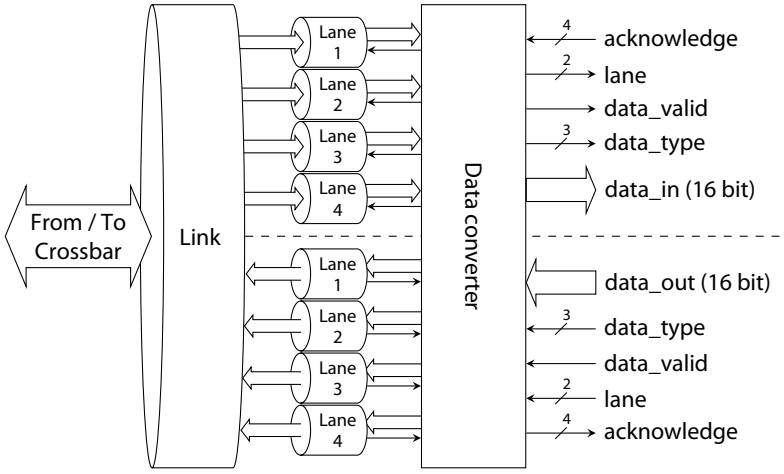


Figure 4.12 – Data converter between tile interface and crossbar

Crossbar

In the router, the crossbar offers the possibility to connection the four lanes of one port to any of the four lanes of all the other ports. This full connectivity is required to not constrain the routing algorithms by the hardware or the necessity to re-route connections in case of incremental mapping of multiple streams. This results in a crossbar with 20 input and 20 output lanes that are interconnected via a fully connected crossbar. Each input lane can be connected to any output lane apart from the lanes of the port on the same side of the router (i.e. no flit U-turns are allowed). U-turns are not necessary, because data does not have to flow back to the previous router. The 20 output lanes of the crossbar are registered. The maximum frequency of the total network will therefore only depend on the maximum delay in a single router plus the maximum wire delay of the link between two routers.

The crossbar both connects the data path of a lane in one direction and the flow

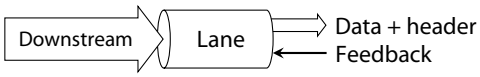


Figure 4.13 – Details of a single lane

control signal in the reverse direction. The fully connected crossbar is implemented with multiplexers, because synthesis tests in 90 nm technology showed a smaller area for a multiplexer crossbar compared to a tri-state crossbar. Furthermore, for test reasons tri-states have to be prevented as much as possible for designs that are realized in silicon.

Flow Control

With only a four bit forward lane from source to destination and no feedback, we have to assume the destination can consume the data. In this case we do not support end-to-end flow control. To prevent buffer overflow at the destination tile, a feedback signal is added in the reverse direction. The new lane consists of four data signals and one acknowledge signal in the reverse direction (see figure 4.13). Depending on the application, one or more lanes and zero, one or more acknowledgements signals can be used.

The acknowledgement signal is used in combination with a credit based mechanism. This mechanism will prevent a buffer overflow at the destination of a connection. Every source has a local window counter of size WC. This local counter indicates how many data-packet the source is allowed to send to the destination. The destination will send an acknowledgement signal when it has read X data-packets, where $X \leq WC$. When the source receives an acknowledge signal, it increases its local counter (WC) by X. By configuring the use of the acknowledgement signal and the sizes of X and WC, we can support both blocking and non-blocking communication.

We have chosen for an end-to-end flow control as this minimizes the required number of buffers in the individual routers. A link-level flow control would require a Moore or Mealy machine to control the buffering of data at each router. A Mealy machine based control of the buffers and acknowledgement signal creates a combinational path along the stream's route. A Moore machine doubles the minimum required buffer depth.

Configuration

To minimize energy consumption the circuit switching has fully separated data and control paths. The configuration of the crossbar, i.e. which input lane is connected with the output lane, is stored in a local configuration memory. Per output lane the connected input lane is stored plus an activation bit. The configuration memory size is $5 \cdot 20 = 100$ bits. We configure the configuration memory via a small additional SRAM-like interface. Configuration of one lane requires 10 bits (five bits address, five

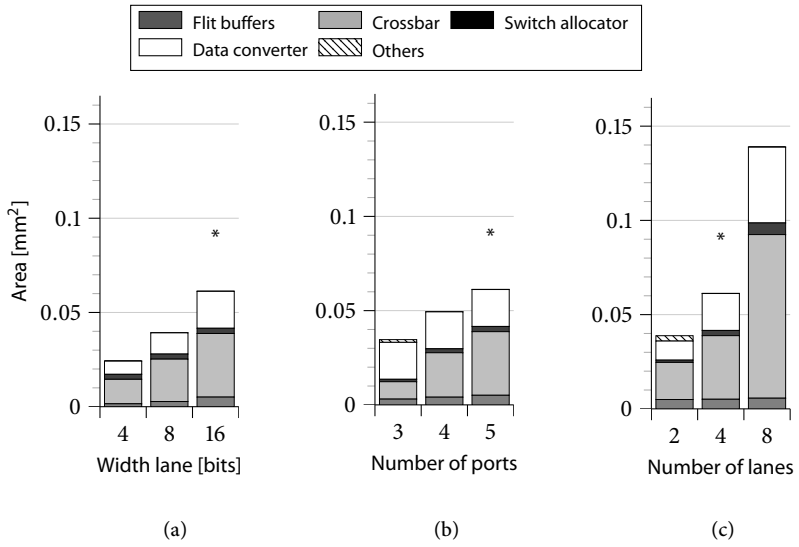


Figure 4.14 – Synthesis results of various CS router designs. Per graph a single model parameter is changed. The values for the reference design (indicated by an asterisk) are five ports, four lanes per port, 16 bits per lane and 64 bit per channel. For all six designs clock gating is enabled.

bits data) that are generated by the CCN. The configuration interface is connected to the separate BE network.

The main disadvantage of the separate control is the inability of individual packets to setup and remove a connection between two tiles. Any fast reuse of lanes between two independent communication streams is therefore not possible. Interaction between CCN, sending and receiving tile is required to know if a specific circuit is empty and ready to be reconfigured. The individual status of the data path of a router is not known. The termination of a circuit has to be signalled by the sending and receiving tile such that the CCN can deallocate the circuit.

4.2.4 Synthesis results

The design of the CS router, as described in the previous section, was described in VHDL and synthesized to ASIC technology. The data path width, number of lanes per port, and number of router ports are left as model parameters. Furthermore, the design is synthesized with and without clock gating.

The router is synthesized for a number of parameter values. The reference design is a five port router with four lanes per port and 16 bits per lane, resulting in a total downstream channel width of 64 bit. Furthermore, automated clock gating is enabled. The synthesis was performed with Synopsys Design Compiler using a TSMC 90 nm library.

Figure 4.14 depicts the variations of the three model parameters in three different graphs. The reference design is included in all three plots and indicated by an asterisk. The data path components—flit buffer, crossbar and data converter—dominate the router's area. Due to the separation of data and control we can almost neglect the area for control. However, an additional configuration network is required, as will be further discussed in section 4.3.4.

In comparison with the reference design of the GuarVC router (see figure 4.9), which also has a channel width of 64 bit, four VC and five ports, the reference CS router is approximately half its area. Furthermore, the amount of area required for buffering is reduced tremendously. However, the area of the crossbar is increased more than three times due to the spatial lane division multiplexing, which requires larger multiplexers. The additional data converter that serializes and de-serializes the tile's data can also not be neglected.

In figure 4.14(a) we reduced the number of bits per lane, which also results in a reduction of the total downstream channel. All data-path components scale linear with the number of bits per lane.

In the middle graph we changed the number of input and output ports of a single router. These routers with fewer ports can be, for example, instantiated at the boundaries of a mesh topology or to construct a ring topology. For the reduced number of ports, the data converter requires an increasing fraction of the total area, because each router has to be connected to a processing core. For a three port router this module even dominates the total resource requirements. The steep reduction of the crossbar's area is caused by the quadratic relation with the number of ports that have to be interconnected. The crossbar allocator and flit buffers scale linear with the number of router ports.

Figure 4.14(c) depicts the area resource for the routers where we changed the number of lanes per port. Per port the total number of downstream channel bits are constant (64 bit). Increasing the number of lanes per port, reduces the width of a single lane. This graphs shows the effect of changing the granularity of the lane division multiplexing. For finer granularity, i.e. more lanes per port, the router's area increase considerable. All components, except the flit buffers, increases considerable due to the increase of lanes.

The reference router design synthesized without the automated insertion of clock gating resulted in an area increase of approximately 25% for both the data converter and crossbar allocator. This area increase is caused by an increasing number of flip-flops with an enable port that are replaced by normal flip-flops and a clock gating element in the gated design. The resulting overall area in the non-gated design is approximately 5% larger

4.3 Comparison with other NoC architectures

In this thesis we compare the two architectures as described in the previous sections with two packet switched alternatives, which are designed by the University of Cambridge. These routers, a simple wormhole router and a virtual channel router

with speculation logic have a 64 bit data path.

4.3.1 WORMHOLE ROUTER

The wormhole router, referred to from now on as *WH* router, uses a conventional input-queued architecture with four flit deep buffers at each input. A two stage pipeline for the flits is provided by the router. The use of look-ahead routing allows crossbar allocation to occur in the first stage with crossbar and link traversal in the second.

In contradiction to the packet format as described by figure 4.1, the routing information is appended to each flit rather than being carried in an additional header flit. This makes the packet length only dependent on the payload size and not the route's length. The routing information consists of a two times two bit destination X and Y address and a deadlock-free XY routing approach is embedded in the network. Additional, a next-port identifier, five bit one-hot encoded, is included to enable the look-ahead routing and a single bit to identify the tail flits. This creates a total flit size of 74 bit.

A pipeline register is included between the input queues and the symmetric crossbar. For the crossbar traversal stage the flit at the head of the input queue is loaded into this register, which drives it across the rest of the data-path. Identical to the GuarVC router, a stop-go flow control is used to stop the flit transmission and prevent buffer overflow of the router downstream.

4.3.2 SPECULATIVE VIRTUAL CHANNEL ROUTER

The second router in this comparison is the Lochside router as introduced in section 2.6.2. This speculative virtual channel router, referred to from now on as *SpecVC* router, is organized as a symmetric VC router architecture as described in section 4.1.1. A conventional input-queued architecture with four VCs per port and four flit deep cyclic buffers for each VC was used. The objective of this architecture is to increase the average performance of the router by using speculation in the control path. It includes look-ahead routing and speculative VC and crossbar allocation, which reduces the flit delay.

In normal VC router designs the routing, VC allocation and crossbar allocation are three sequential stages, i.e. the head flit notices a minimum delay of three cycles per router. Look-ahead routing determines the next-port identifier for the current hop in the preceding router. This makes it possible to do the crossbar allocation of the current hop and routing function of the next hop concurrently. This reduces the delay of the head flit with a single cycle.

Both the VC and crossbar allocators, which respectively allocate VCs and crossbar ports, are based on matrix arbiters. The crossbar allocator is replicated with a speculative allocator for newly arrived flits. The crossbar allocator always prioritizes requests from already waiting flits, i.e. non-speculative requests. The crossbar traversal of newly arrived flits is aborted when multiple newly arrived flits need the same output port. Since both crossbar and link traversal are performed in a single

clock cycle, in the best case, an incoming head flit finds pre-allocated resources and can thus be forwarded to the next hop in a single clock cycle.

The routing information is similar to the WH router, five bit next-port identifier, four bits each for destination X and Y address, a bit to identify tail flits, and is extended with an one hot encoded four bit VC identifier. This results in a total flit size of 82 bits. Identical to the other two packet-switched designs, a stop-go flow control is used to stop the flit transmission and prevent buffer overflow of the router downstream. Further details have been provided by Mullins et al. [98].

4.3.3 PACKET COMPARISON

Figure 4.15 depicts the organization of the extra control information that is required to send a payload of 256 bits over the specific networks. This control information can be divided in routing information and flow control. The latter has both feed forward and feedback signals to control the transport of the payload and routing information from one router to the next router on the route. These packets will be used in both latency and power comparison of the four routers.

Figure 4.16 depicts the overhead of the control information versus the payload size of a packet. The CS packets do not contain routing information, which is controlled by a separate network. Its control information overhead is 33% and that is equal to the overhead for the SpecVC packets. The overhead for the WH packets is 23%, because these packets do not require VC flow control. The overhead for GuarVC BE packets is not a constant factor compared to the other architectures, but will reduce with the increase of the payload per packet. For the GT packets we only have to setup the route once (without the deallocation by a tail flit) and the successive packet will not require routing information as depicted in figure 4.15b_2. These GT packets have a control information overhead of 14%.

4.3.4 AREA COMPARISON

The GuarVC and CS router, used in the comparison with the WH and SpecVC router, are the reference designs of respectively sections 4.1.4 and 4.2.4. The CS router is extended with a small WH router that is directly connected to the configuration interface. This BE router is included to determine the cost of an additional configuration network.

All four routers are described in an HDL, either VHDL or Verilog. Using a normal ASIC tool flow, each router design was synthesized, placed and routed using a TSMC 90nm library, high-performance process with a core voltage of 1.2 V and nominal threshold voltage. The automatic clock gating during synthesis is enabled, which is described in more detail in chapter 6. The area of the designs was obtained after placement and routing and is presented in table 4.1.

In comparison with the figures presented in figures 4.9 and 4.14 for the GuarVC and CS router, the area figures of table 4.1 are approximately 30–40% larger. This is caused by the insertion of the top-level clock tree, some increased cell sizes due the increase wire loads and additional spacing between the gates that is inserted

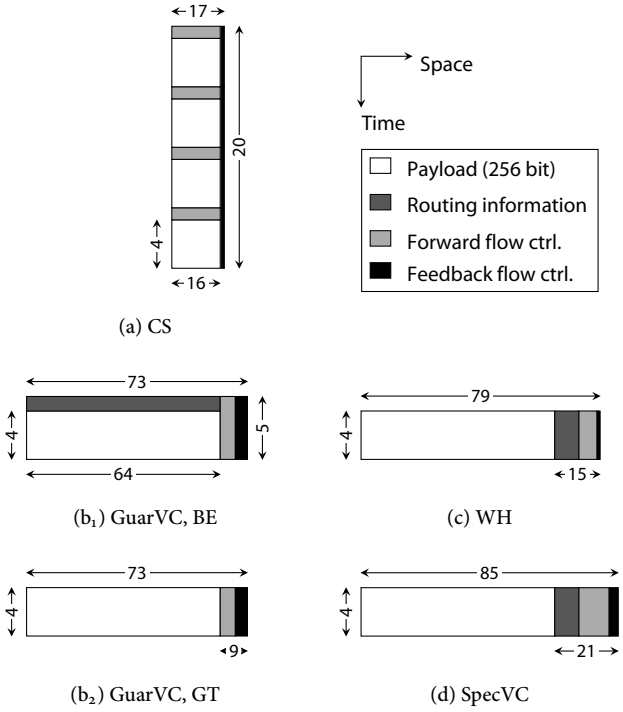


Figure 4.15 – Packet information for a 256 bit payload

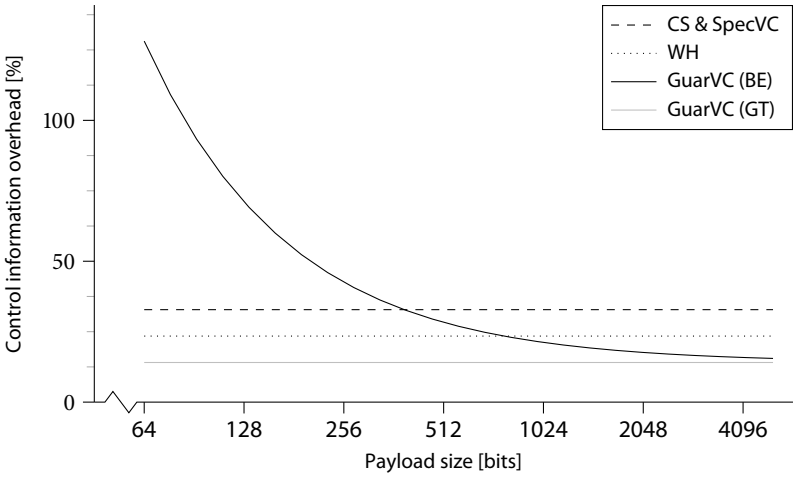


Figure 4.16 – Overhead of control information, including flow control signals, versus the packet's payload size

Table 4.1 – Post place and route area of the different routers

Component	Area [mm^2] | [%]			
	CS	WH	GuarVC	SpecVC
Flit buffers + logic	0.008 | 7.19	0.045 | 57.7	0.160 | 74.7	0.166 | 67.3
Crossbar	0.058 | 53.6	0.016 | 20.1	0.034 | 16.1	0.015 | 6.19
Crossbar allocator	0.004 | 3.34	0.005 | 6.55	0.018 | 8.36	0.015 | 6.24
VC allocator	–	–	–	0.022 | 8.89
Output ports	–	–	–	0.012 | 4.66
BE router	0.009 | 8.76	–	–	–
Data converter	0.026 | 24.3	–	–	–
Other	0.003 | 2.79	0.012 | 15.7	0.002 | 0.87	0.016 | 6.72
Total	0.108 | 100	0.078 | 100	0.214 | 100	0.246 | 100

during placements and routing. The area for the VC allocator for the GuarVC router could not be determined in the placed and routed design. Its area is split over the flit buffers and crossbar allocator.

The breakdown of the area for the CS router showed that the higher order crossbar is its largest component, being approximately 3.5× larger than the WH crossbar. This, along with the serialisation logic, the additional packet switched network and the configuration memory areas of the CS router together outweigh the saving of area from reduced buffering compared to the WH router.

The increase in area for the virtual channel routers are caused by the extra input queues for each VC. Furthermore, the GuarVC has a larger asymmetric crossbar and the SpecVC requires more area for its speculative allocation.

4.4 Conclusion

In this chapter we described the architecture and implementation of both a packet switched and circuit switched router architecture. Both routers are designed with the assumption that a streaming application, which is mapped onto the multi-core architecture, requires QoS for the communication. Furthermore, this multi-core architecture is controlled by a central entity, the CCN.

The GuarVC packet switched architecture is a router with VC flow control and source routing. The VCs are introduced to decouple the packet length from the buffer depth and to offer QoS. QoS is possible via the unique assignment of a VC to a stream that requires this service, and deterministic arbitration per physical channel. This results in an upper bound on the latency of a packet, given its route, packet length, and the number of occupied VCs per physical channel along the route. In comparison with an earlier implementation the router area is reduced and the individual data and control paths are separated such that better characterization is possible. The largest part of the router's area is consumed by the input buffers.

The large buffer requirements for the GuarVC router led to the second router

architecture proposed in this section. This CS router architecture benefits from the observation of the semi-static behaviour observed in the analysed streaming applications. This architecture replaces the VCs, which are time multiplexed, by parallel physical lanes on a link between two routers. To reduce the number of wires required for this link, we applied serialization, such that the total raw bandwidth of the links in the CS network is identical to the GuarVC network. The second characteristic of this circuit switched network is the complete separation of data and control. The router's crossbar configuration, and with that the route of the data, is controlled via a separate control network. The CS router is significant small in comparison with the GuarVC architecture. The largest parts are the crossbar and serialization logic.

The two routers are compared with two other packet switched routers, designed at the University of Cambridge. Their WH router and SpecVC router are tailored to more randomly distributed traffic. In terms of resource usage the WH router requires the least amount of resources, due to a single small input queue per input port. The CS router is slightly bigger compared to this small packet switched router, due to a relative large crossbar and serialization interface. The two virtual channel routers, GuarVC and SpecVC, occupy nearly the same area, of which the largest part is consumed by the input buffers. The GuarVC has a slightly bigger crossbar, due to its asymmetric size that enables a simple and deterministic QoS mechanism. The SpecVC is the largest router in this comparison. Its larger area is caused by the extra speculation logic.

Chapter 5

Timing Evaluation

Abstract – *This chapter describes the timing evaluation of the GuarVC packet switched network. Both the BE and GT packets are simulated, because both types of traffic will observe a variable latency under variable load conditions. The simulations are used to improve the original design to increase its performance. Besides the analysis of various traffic patterns, the router is also compared with two packet switched alternatives. We demonstrate that the GuarVC is the only architecture that can offer QoS, but has a lower performance for randomly distributed BE traffic.*

This chapter describes the timing evaluation of the GuarVC packet switched network. The simulations are performed on the SHILS simulator as described in chapter 8. These simulations are used both to analyse the behaviour of the network and to profile the simulator. The analysis of the NoC is included in this chapter and the profile results of the simulator are described in section 8.4.

Most of the NoC architectures are mainly evaluated using simulations with random traffic uniformly distributed over time and tiles. The average and maximum latency of all packets is calculated and plotted versus the packet injection rate. Therefore, this is also our initial test-case in section 5.1. We start with the router architecture as initially proposed by Kavaldjiev [78]. Based on the initial tests we evaluate possible improvements. The GuarVC router also supports GT traffic, which is simulated in section 5.2.

Parts of this chapter have been presented at the 1st ACM/IEEE International Symposium on Networks-on-Chip, Princeton, NJ, USA [PW5], and in the journal IEEE Transactions on VLSI Systems [PW1].

The GuarVC router is compared with two other packet switched routers designed at the University of Cambridge in section 5.3. These routers, a simple wormhole router (WH) and a virtual channel router with speculation logic (SpecVC) as described in section 4.3, have a 64 bit data path. The 64 bit version of the GuarVC router is used and two BE tests are performed to compare the three implementations. These implementation are also compared on their energy consumption in chapter 6. In section 5.4 we draw some conclusions.

5.1 Best Effort Traffic

In this section we test the GuarVC router on its performance under uniform traffic with various network loads. This initial characterisation of the GuarVC's performance is performed with a 16 bit data path and a 6×6 mesh topology for the NoC, such that the initial results are comparable with the preliminary test of Kavaldjiev [78].

Uniform traffic is normally obtained by selection of a random destination for each packet that is injected by a specific source tile. However, due to the specific VC allocation mechanism, at most four packets can compete for a VC on a specific link. This limits the number of destinations that can be reached by a specific source. To emulate uniform traffic we randomly select pairs of communicating tiles for which a unique VC-ID combination is selected per physical channel that the packets between the pair of tiles will use. Per communicating pair of tiles an XY-route is determined through the mesh-topology. XY-routing is applied, because it is deadlock-free. For a 6×6 network 493 randomly selected pairs can be mapped onto the network. On average, a tile communicates with 14 other tiles, which is close to the maximum of 16 (four VCs and four IDs per VC). All the routes that use a specific link are uniformly distributed over the four virtual channels and four IDs per virtual channel.

Each communicating pair will transport a packet of D data flits at random moments in time. The data flits are preceded by H header flits that contain the routing information and followed by a single tail flit that will free the router's resources for other packets. The number of header flits is equal to the number of hops of the packet's route. Per hop, a header flit is consumed, which reduces the average number of flits/cycle that arrive at a specific tile. Therefore, the latency figures in this section are plotted versus the *gross injection* and *net injection* rate of the tiles. The gross injection rate is defined as the average number of flits per system cycle a tile injects into the network. The net injection rate is defined as the average number of non-header flits per system cycle a tile injects into the network. The difference between the gross injection and net injection rate is the number of header flits consumed by the network. The net injection rate represents the amount of payload data transported by the network.

The latency per packet is measured and grouped depending on the length of its route. The minimum latency per packet is equal to $2H + D + 1$ (the number of hops + the length of the packet).

Figure 5.1 depicts the average and maximum latency of a packet versus the gross

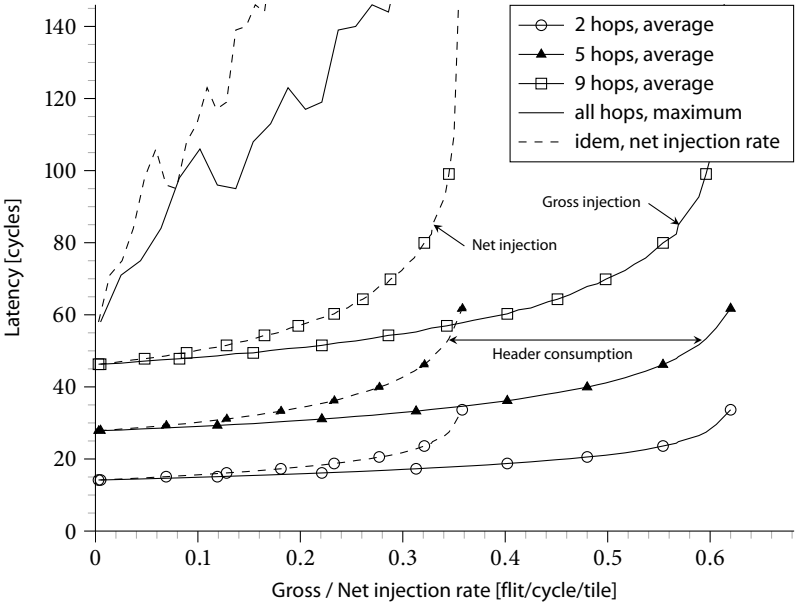


Figure 5.1 – Average and maximum latency for 5 data flits BE packets. *Measurement conditions:* Mesh network of 6×6 routers, uniformly distributed traffic, payload of five flits. Variable between tests: Average inter-packet time between BE packets of each source. Number of tests: 43. Simulated clock cycles per test: 595.200. Packets' latency grouped by the Manhattan distance between source and destination.

and net injection rate when all packets contain five data flits. Furthermore, it is visible that the average latency is influenced by the length of the route. The average latency is given for a selection of route lengths. However, the maximum latency of a single packet is not directly related to the length of the route, but mainly dependent on the packet injection rate.

5.1.1 IMPROVEMENTS

Improved ID selection

The majority of the latency seems to be caused by the VC allocation mechanism, which allocates a waiting packet on its requested output VC. This access mechanism should prevent two or more packets to be granted to access the crossbar and select the same output VC. The packets from different input VCs have an unique ID when they compete for the same output virtual channel. At most one packet per output VC is granted if its header ID matches the global counter. However, this mechanism can also hold up allocation when only one packet requests a specific VC and its ID is not equal to the counter value.

In the reference simulation we have chosen a random ID for each header. Al-

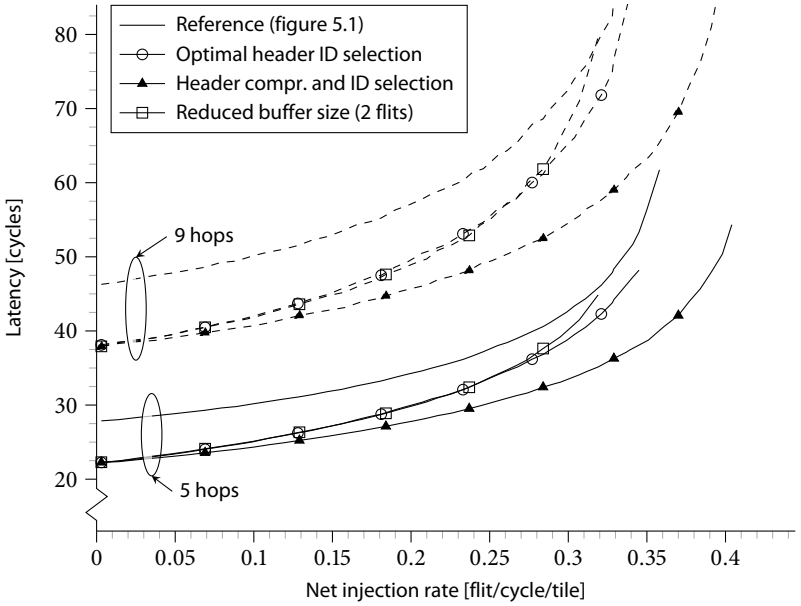


Figure 5.2 – Improved latency for 5 and 9 hops. *Measurement conditions:* Mesh network of 6×6 routers, uniformly distributed traffic, payload of five flits. Variable between tests: Average inter-packet time between BE packets of each source. Number of tests (top to bottom): 41, 47, and 36. Simulated clock cycles per test: 595.200. Packets' latency grouped by the Manhattan distance between source and destination.

though this is fair, it is not fast. As the ID counter value is globally synchronized for the whole NoC, we can predict what the counter value is when the next header of the packet requests crossbar access at the next router, based on the minimum delay of this header flit. In the current design this delay is three cycles. Figure 5.2 depicts the latency if the header ID is optimized to this three cycle delay. Especially for low traffic loads, the latency is reduced by approximately 20%. The network almost saturates at the same net injection rate. In section 5.3 we will see that speculation logic can further improve the delay caused by the VC allocation.

Header Flit Compression

Furthermore, for these small packets of five data flits we notice a relatively large overhead of the header flits as the average route in a 6×6 network is 4.7 hops. The header/data ratio is in this case almost one and for larger topologies or smaller packets, the header overhead will even be larger. Each header uses only six of the available 16 bits, which makes it possible to combine two header flits into one and reduce the average number of header flits to 2.4. The router requires only one extra multiplexer per VC to select the correct bits when it receives its first header of a packet. As depicted in figure 5.2, the packet's have a lower average latency and

the network saturates at higher loads. In this case we combined optimized header ID selection with header compression. Solely header compression did not show a major improvement. Header compression is especially beneficial for higher loads and small packets. The latency at low loads is mainly effected by the ID selection, because the ID selection dominates the packet's delay.

Buffer Reduction

A third test was to reduce the queue size (two flits) of the NoC and see its effect on the performance. Both header compression and ID optimization were included as well. At low traffic loads the buffer size marginally influences the latency. Due to the smaller buffers, the network saturates at lower loads (see figure 5.2).

5.1.2 Other best effort scenarios

In the previous section we tested the GuarVC NoC with a fixed packet size and mesh topology of 6×6 routers. For a second scenario we varied either the network size or the number of data flits in the packet. In figure 5.3, the latency depending on the net injection rate is depicted for both a 6×6 network and an 8×8 network. For the 6×6 network we display the average latency for the scenarios with five or 11 data flits per BE packet. From this figure, it is clear that a larger network will saturate at a lower net injection rate (lower throughput per tile). Similar behaviour is observed by Duato et al. [47, chapter 9]. Larger packets have a higher average latency, but no noticeable change of the saturation rate.

5.2 QoS and Best Effort Traffic

5.2.1 Jitter analysis

As described in section 4.1.2, the GuarVC router can support both BE and GT traffic. Kavaldjiev [78] performed latency measurements to show that a GT stream will never violate its given deadline, independent of other traffic in the NoC. Figure 5.4 depicts the result of these latency simulations, which are performed with a SystemC based description, for a 6×6 mesh network topology. For the GT, packets with a payload of 256 bytes (128 flits) are used, versus 10 bytes (five flits) for BE packets. The BE traffic is uniformly distributed and each tile has one destination for GT packets. The GT packets are transmitted at a fixed interval of 4 µs, which is the symbol period of HiperLAN/2 (see section 3.2.1), with a frequency of 333 MHz. In these tests the amount of GT traffic is constant for all tests and the amount of BE is increased by reducing the average inter-packet time.

The graph shows how the latencies of GT and BE packets depend on the offered BE load. For the GT traffic, the mean and the maximal latency of packets are given. Furthermore, the guaranteed latency for all GT packets is depicted in the graph as a horizontal thick line.

When the offered BE load is low, the latency of the GT packets is smaller than the guaranteed (or allowed) latency. The reason is that the GT traffic utilizes the

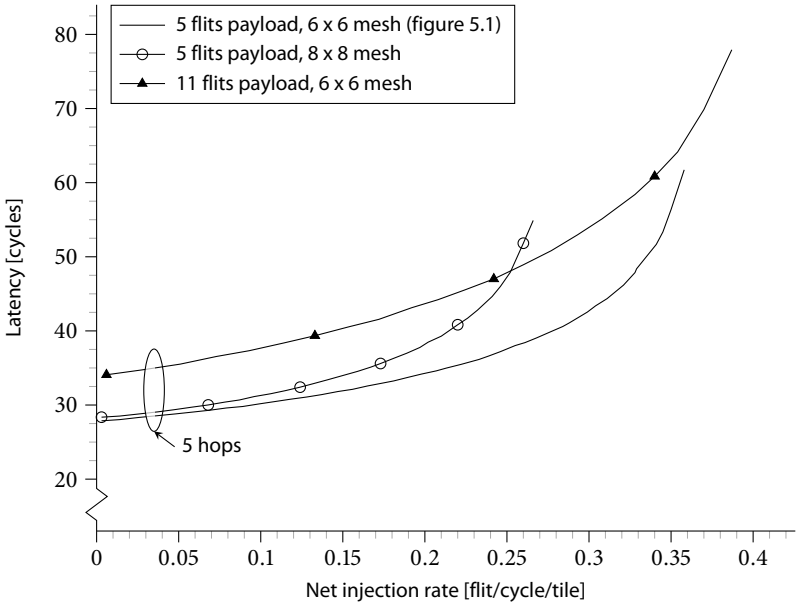


Figure 5.3 – Average latency for 5 and 11 flits BE packets for different topologies. *Measurement conditions:* Mesh network of either 6×6 or 8×8 routers, uniformly distributed traffic, payload either 5 or 11 flits. Variable between tests: Average inter-packet time between BE packets of each source. Number of tests (top to bottom): 32, and 22. Simulated clock cycles per test: 595.200. Packets' latency grouped by the Manhattan distance between source and destination.

bandwidth unused by the BE traffic. The latency of the GT packets is higher than the latency of the BE traffic, because in this experiment the GT packets are larger. With the increase of the BE load, the average and maximum latency of the GT traffic increases, but the maximum GT latency never exceeds the guaranteed latency.

In the following a similar test is conducted, but the inter packet jitter of the various GT streams is studies as well. For our experiments we use the SHILS (see chapter 8). With the SHILS we are able to simulate the NoC for a considerable longer time compared to the simulations using a PC, which was used by Kavaldjiev. We mapped the following three types of streams onto the network:

I GT packets of 128 flits with an average inter-packet time of 1333 cycles (identical to the test of Kavaldjiev).

II GT packets of 256 flits with an average inter-packet time of 4000 cycles.

III BE packets with 5 data flits. The inter-packet time of the BE packets is varied to increase the load in the network. All BE streams are mapped on one single virtual channel. The unique ID will prevent conflicts at the crossbar.

Both GT streams have minimum guarantee of ⅓ of the physical channel's bandwidth (i.e. at most three VCs per physical channel are assigned). The maximum

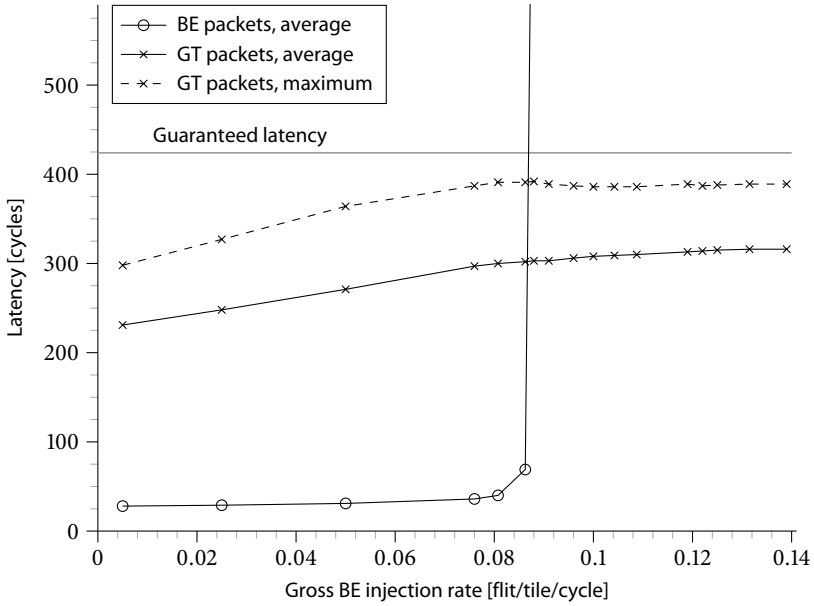


Figure 5.4 – Packet latency of the GT and BE traffic vs. gross BE injection rates for 6×6 mesh network (queue size 2 flits), adjusted from Kavaldjiev [78, figure 3.11, page 60].

latency of a single GT packet equals: $3 \cdot (H + D)$ cycles, where H is the length of the route and D the length of a single packet. The test has 29 streams of type I, 18 streams of type II, and 103 streams of type III, all uniformly distributed over the NoC.

In this test, we are interested in the inter-packet jitter of the GT packets. Of course, it is important that the packets will arrive at the destination in less than the guaranteed latency. Another service is that the packets will also arrive at a very regular or predictable interval (low jitter). A low jitter makes the scheduling of the processes on the individual cores probably more predictable.

In the test, the latencies of the two GT packet types are examined and plotted in a histogram, where the latencies are grouped by the length of the route. The best effort load varies over time and its influence on the latency of the other streams is analysed.

To get an accurate estimate of the variation in the packet latency, the simulation of a very large number of packets is required. Suppose we simulate for 1.5 million cycles. In this period 1125 packets per stream of type I and only 375 packets of type II are transported. The calculation of the average latency is possible, as multiple streams of a type, which have the same length, can be grouped. But, for a histogram a lot more packet latencies have to be collected, as the latencies range from 130 to almost 400 clock cycles. Simulation of 1.5 million cycles in the SystemC simulator required roughly one hour and 45 minutes. The SHILS simulator performs the same

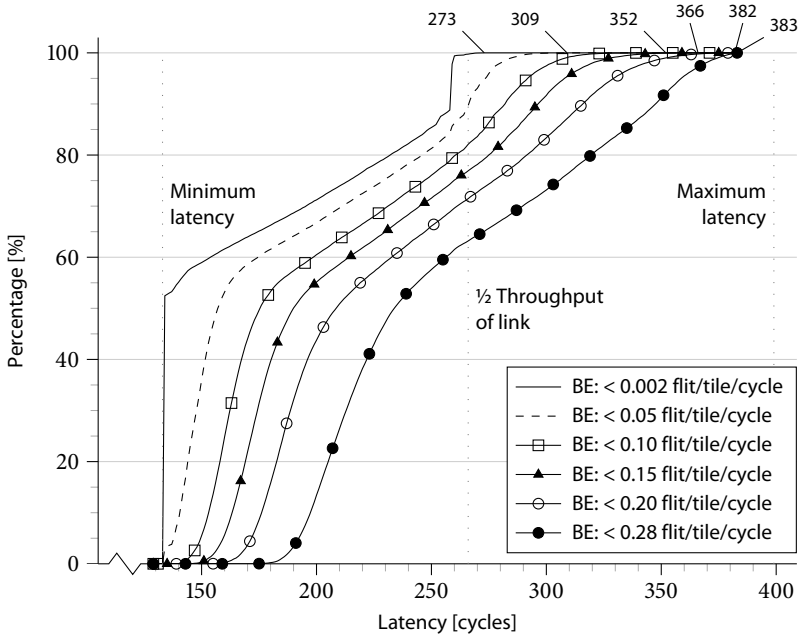


Figure 5.5 – Latency distribution under different BE loads. *Measurement conditions:* Mesh network of 6×6 routers, three traffic types (GT_1, GT_2, and BE), all traffic types uniformly distributed, payload respectively 128, 256 and 5 flits/packet, VC of GT traffic allocated at the start of each test. Variable between tests: Average inter-packet time between BE packets of each source. Number of tests: 30. Simulated clock cycles per test: 14.880.000. Depicted: latency distribution of GT_1 packets for six selected tests.

test in less then a minute. Therefore, we simulated 15 million cycles in approximately 10 minutes and obtained detailed histogram results.

Figure 5.5 depicts the distribution function of the latency (the percentage of packets that notice a specific maximum latency) of all streams of type 1 under different best effort loads. The y-axis gives the percentage of the packets that notice the latency on the x-axis or less. Per line in the graph, we also indicated the maximum latency that is observed for all streams of type 1.

For different BE loads, the distribution function changes and the maximum GT latency increases. At lower loads the amount of slack in the network is large, which is used by the GT packets. From this figure, it is clear that under almost no BE load, 50% of the packets arrive with the minimum latency (length route + length packet) and use the maximum bandwidth of the links. Because there are multiple GT streams in the NoC, a fraction of all the packets has to share bandwidth of the link. Depending on the phase alignment of the packets they influence each other. However, this can be at most a bandwidth reduction to half the link bandwidth under low BE load. A little more than 10% of the packets observe this reduced

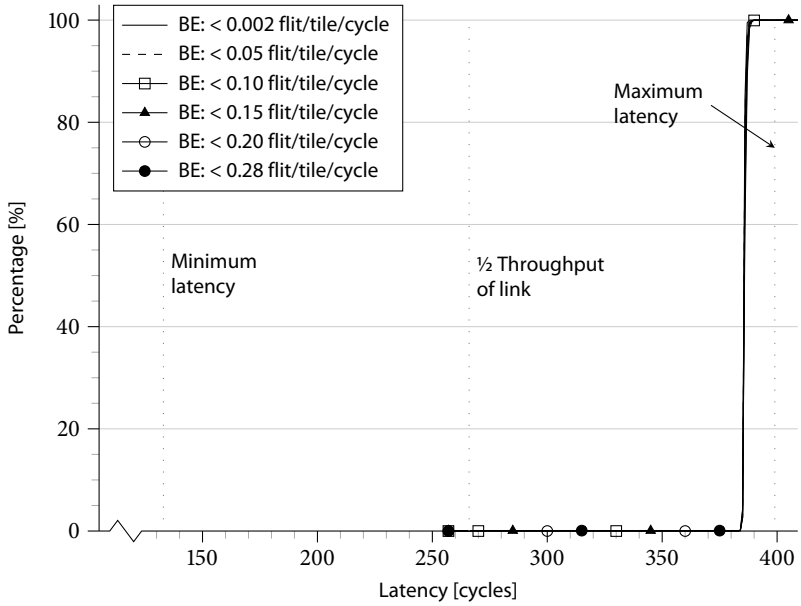


Figure 5.6 – Latency distribution under different BE loads, where the GT streams are traffic shaped at the input. *Measurement conditions:* See figure 5.5.

throughput. Under higher best effort loads the GT packets get influenced more and more. This has a major impact on the latency distribution function. This change in the distribution function will be noticed by the receiver as jitter in the arrival time of packets.

Traffic Shaping

If we now apply traffic shaping by releasing the packets at the source with their given bandwidth of ⅓ of the link bandwidth (i.e. one flit per three clock cycles), we expect them to experience less jitter in the network. This shaping is similar to a cyclic TDM access mechanism as used in the Æthereal network.

The resulting latency distribution of the latency of all streams of type I is depicted in figure 5.6. It is clear that the streams do not get influenced by the BE traffic, but experience a large latency as the packets enter the network at their guaranteed bandwidth. The maximum latency ranges from 388 to 390 cycles and is not indicated in the graph.

5.3 Comparison

The latency measurements reported so far only represent the performance of a specific packet switched router architecture. To obtain a more complete characteri-

sation it is also important to obtain the performance of other router architectures. Two other router architectures are selected that are also used for area and energy comparison. These architectures make it possible to show the benefit of VC flow control versus wormhole routing and the difference between a more static VC allocation and deterministic crossbar allocation versus a VC router with speculative logic for allocation. The architecture details of the routers are described in section 4.3. The GuarVC router is adjusted to a 64 bit data path and has header compression enabled. The data path of 64 bit allows to encode up to 10 hops in a single header (see figure 4.6).

Ideally, the comparison would be performed using full system level simulations with real applications, but given the absence of these, synthetic traffic generators have been used similar to the previous sections. The particular performance tests performed attempt to represent a range of expected realistic scenarios. The first test used uniform random traffic, which can be considered as a base test for performance comparison of different architectures. The importance of locality, which is highlighted by work such as that by Greenfield et al. [58], motivated to compare the architectures for a traffic pattern where the destination of randomly generated communications favoured those nodes closer to the source.

All experiments were carried out with an 8×8 network, with four flit long packets. For all three routers, the same traffic sources were defined entirely in C. The simulations of the GuarVC router were performed by the FPGA based SHILS. The two other routers were simulated using an HDL simulator, because the automatic embedding of the routers' architectures in the SHILS simulator was not possible at the time the tests were performed. The C traffic sources were linked with the HDL network description using the Verilog Programming Language Interface (PLI). Both solutions provided a highly flexible framework, where each tile could be modelled by a separate set of C routines, at any desired level of complexity.

For the SpecVC and WH router the simulated time is determined by the number of transmitted packets. An initial 500 packets transmitted per node were used to initialise the network in the *warm-up* period. The subsequent 3000 packets were the ones used during the measurements in the *sampling* period, with an additional *drain* period used at the end to allow all simulation period packets to be received. For the GuarVC we use a fixed simulation time of 592.000 clock cycles, where we sampled the latencies of packets after 25% of the simulated time. At net injections rates above 0.03 flits/tile/cycle more than 3000 packets are injected per source within the sampling period.

5.3.1 UNIFORM RANDOM TRAFFIC

Figure 5.7 shows the measured average packet latency at varying traffic net injection rates into the network under a uniform random traffic pattern. Each source has an equal probability of transmitting to any other source (apart from itself).

Net injection rate is chosen, because the SpecVC and WH injects the header information in parallel to the payload and the GuarVC before the packet's payload. For the GuarVC we also had detailed latency distributions available. Per measure-

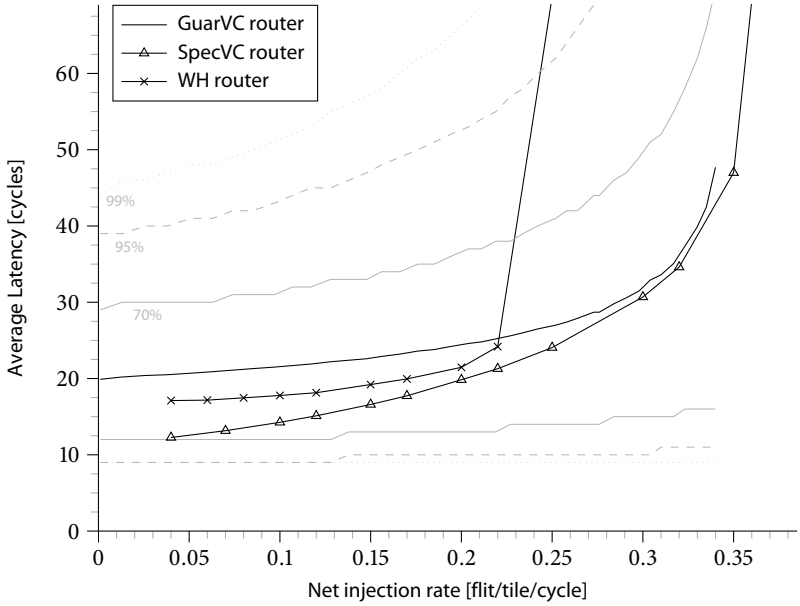


Figure 5.7 – Packet latency for uniform random traffic. *Measurement conditions:* Mesh network of 8×8 routers, uniformly distributed traffic, payload 4 flits/packet. Variable between tests: Average inter-packet time between BE packets of each source. Number of tests: 40 (GuarVC), 13 (SpecVC), 10 (WH). Simulated clock cycles per test: 595.200 (GuarVC), cycles for 3500 packets/tile (SpecVC,WH).

ment, we determined the boundaries between which x % of all packets were present, such that an equal amount of packets are above and below these boundaries. The three pair of gray lines in figure 5.7 indicate the 70%, 95% and 99% boundaries. The lines give an indication of the spreading in packet latencies.

As expected, the VC networks saturate at a higher injection rate than the WH network and the delay optimised SpecVC network achieves a lower delay than the WH network. The increased delay for the GuarVC router is caused by the relatively simple and static VC allocation scheme used. The VCs for the entire path are allocated at the source, before a packet even enters the network. This is necessary as the GuarVC design is primarily optimised for QoS traffic. However, this causes the packets to be halted in the buffers even if the output port is free. A second source of the extra latency (one or two cycles) is the larger length of the packets, because the routing information is encoded in header flits that precede the payload. For the WH and SpecVC router architecture this information is placed in parallel to the data. Despite the larger packets of the GuarVC design the saturation points of both GuarVC and SpecVC are almost identical.

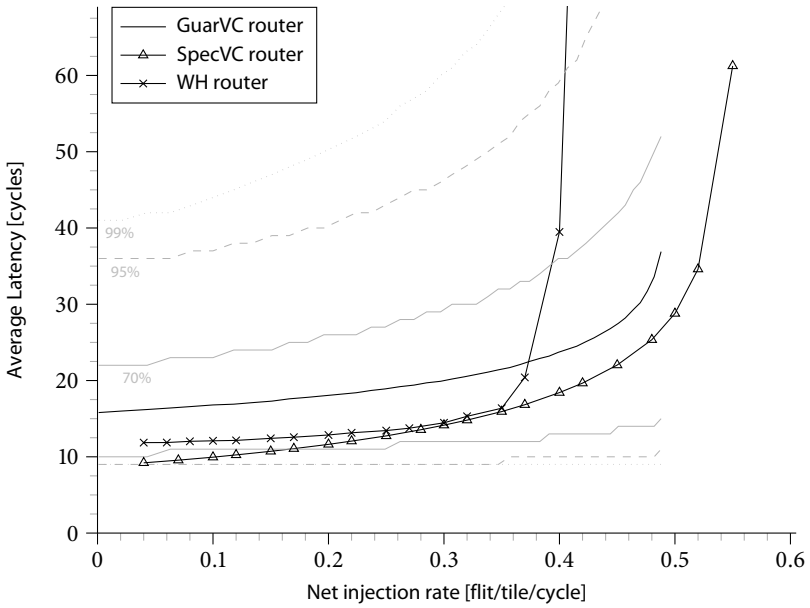


Figure 5.8 – Packet latency for localized random traffic. *Measurement conditions:* Mesh network of 8×8 routers, localized traffic (using Rent's rule), payload 4 flits/packet. Variable between tests: Average inter-packet time between BE packets of each source. Number of tests: 44 (GuarVC), 21 (SpecVC), 17 (WH). Simulated clock cycles per test: 595.200 (GuarVC), cycles for 3500 packets/tile (SpecVC,WH).

5.3.2 LOCALISED TRAFFIC

It has been shown that localised traffic can form an important part of on chip communication traffic [58]. To model this, a roughly exponentially distributed hop count based traffic generator was used at each node, where 40% of all transmitted packets were sent only one hop away, 25% was sent two hops away, 15% was sent three hops away and the rest uniformly distributed across the rest of the network. Figure 5.8 depicts the measured average packet latencies at varying traffic net injection rates into the network for the localised random traffic.

Compared to the uniform random case, all the architectures now show a lower latency and higher saturation point, due to the lower average hop count. Compared to the uniform random distribution the WH and SpecVC router benefit most from localised traffic, as for these architectures the injection delay dominates over the allocation delay. For the GuarVC design the static VC allocation has an increased influence on the packet latency.

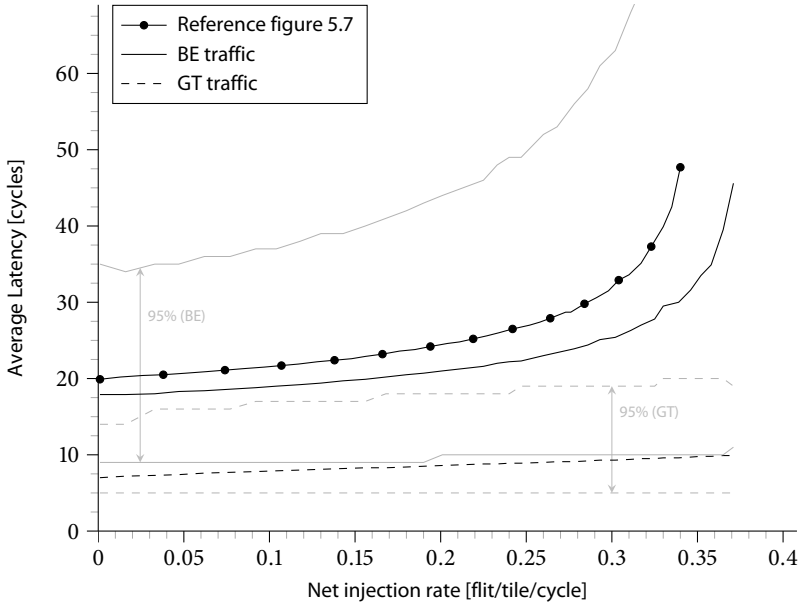


Figure 5.9 – Packet latency for combined streaming and uniform random traffic. *Measurement conditions:* Mesh network of 8×8 routers, 2 traffic types (GT, and BE) per source, 33% of packets GT and 67% BE, distribution respectively localized and uniformly, payload 4 flits/packet. Variable between tests: Average inter-packet time between both GT and BE packets of each source. Number of tests: 37. Simulated clock cycles per test: 595.200.

5.3.3 Streaming traffic

The previous two tests assumed randomised traffic scenarios with equal priority for all packets in the network. In real radio, multimedia or scientific applications, the process graphs consist of a lot of single-in single-out processes that communicate frequently and have QoS demands. Therefore, in this third test we offered both BE and GT traffic to the GuarVC network. The GT packets from a tile are destined to one specific, other tile in the network. The Manhattan distance between the GT pairs has the same distribution as for the localised traffic scenario, 40% of all pairs are one hop apart, 25% two hops, 15% three hops and the rest of the pairs four or more hops. The assumption here is that the mapping of processes to tiles will be optimised for locality. The BE packets are uniformly distributed as in section 5.3.1; 33% of the tile's injected packets are of type GT and the remaining of type BE.

Since the GuarVC design is the only one supporting QoS needs (it has a clear distinction between BE and GT traffic types), it was the only design tested with such traffic.

Figure 5.9 depicts the latency for both BE and GT traffic. For reference the GuarVC latency of the uniform random test is included. Tests with a different ratio between BE and GT packets resulted in comparable results.

The latency for the GT packets is significantly lower compared to the BE traffic. For the GT packets all resources are pre-allocated in the network, which make the latency the sum of only the injection delay and hop distance. The GT latency increase is caused by the flit interleaving of multiple packets on the link. At higher net injection rates the BE part of the traffic saturates, but the GT traffic is guaranteed at least 50% of the link's bandwidth and will therefore never saturate. The saturation point of the BE traffic is higher compared to solely uniform BE traffic, because fewer of the total injected flits are blocked for relative long periods.

5.4 Conclusion

In this chapter we describe the timing evaluation of the GuarVC packet switched network. The CS network has a deterministic latency, due to congestion free routes.

Initially, the original protocol of the GuarVC is improved such that both latency is reduced and the network saturation point increased. The improved version of the architecture is compared with the two packet switched routers of the University of Cambridge.

In this comparison we use a 8×8 network of routers and per router a local traffic generator that injects packets with a fixed payload. The packet's destinations are either randomly distributed or based on Rent's rule. The latter is to simulate localized traffic in the NoC.

For both destination distributions, both virtual channel routers have a higher saturation point in comparison with the wormhole router. The GuarVC has a rather static VC allocation scheme that causes a longer delay for the header flit. This results in a higher average latency compared to the other two packet switched routers. The other penalty for the GuarVC is its organization of the packet, which results in an additional header flit with the packet's routing information. The advantage is the reduction of wires per physical channel and this header overhead is negligible for larger payloads.

The major difference between the GuarVC and the other packet switched alternatives is its QoS support. We measured the effect of various loads of randomly distributed BE packets on the latency of GT packets. The GT packets' latency distribution is plotted versus the additional BE that is injected in the traffic. Despite the increase in BE traffic, a GT packet will always arrive within the bounded latency. The increased network load solely causes a higher jitter in the GT packet latency. Jitter can be prevented by traffic shaping of the input streams, but this also increases the average and maximum packet latency. Depending on the system requirements, one has to decide what is desirable: hardly any jitter with a large latency, or a lower average latency that varies with the network load.

Chapter 6

Power Evaluation

Abstract – *This chapter presents the evaluation of the Network-on-Chip architectures on their power consumption. The two architectures presented in this thesis are analysed in detail and the effect of clock gating is examined. We also compare the routers with two other packet switched router alternatives. The architectures are analysed for both static and dynamic power after placement and routing of the design. Various traffic scenarios are used to obtain a good understanding of their influence on the overall power consumption.*

Evaluating the power efficiency of a NoC router is not a trivial task, because as far as we know no general method has been defined for on-chip networks. In this chapter we define a set of tests to evaluate the routers presented in chapter 4. Several approaches are possible to estimate the overall energy of the NoC. We present a short overview of power estimation techniques in section 6.1.

We performed power measurements at the RTL level, i.e. a detailed power analysis, that gives more insights in the possible bottlenecks and ensures accurate power values. In section 6.2 we describe the flow and setup used to obtain the measurement values. Section 6.4 presents the measurement results for the GuarVC and CS router designs. These results are also compared with the two other packet switched architectures (SpecVC and WH) in a detail in section 6.5. In section 6.6 we draw some conclusions on the obtained results.

Parts of this chapter have been presented at the 7th International Symposium on System-on-Chip (SoC 2005), Tampere, Finland [PW14], and in the journal IEEE Transactions on VLSI Systems [PW1].

6.1 Power Estimation Techniques

To optimize the design for power, the designers have to be able to determine the power consumption of their designs. The most accurate method is measuring an actual IC under typical and worst-case conditions. However, no trade-offs or optimizations can be made after production and it is difficult to examine a specific modules' consumption.

Before the actual production, several ways of energy/power estimation exist. For example, Benini and De Micheli present an overview of power estimation and optimization at the various levels of the system design [16]. A survey of power optimization techniques at the RTL level is presented by Pedram and Abdollahi [103].

Each estimation method has a different trade-off between accuracy, time to estimate and level of detail. A method can rely on either simulation, which uses detailed information of activity over time, or probabilistic techniques, which uses statistical characteristics of the signal. The latter technique is more computationally efficient, but the results are less accurate.

Most accurate power tools can estimate the power of the design at circuit level, where the details of all transistors and other components including their activities in time are known. Examples of tools are HSPICE from Synopsys [128] and Virtuoso Spectre Circuit Simulator of Cadence [129]. Power estimation at this level is based on actual currents and voltages in the circuit. For the gates in a standard cell library this approach is used to characterize the gates. At very high levels, like the application or system level, the activity of signals is not exactly known, but the high level functionality is. The designer can for example count the number of additions, multiplications and memory accesses. Multiplying those numbers with average power values for the basic operations gives a rough estimate on the power consumption. This is rather inaccurate, because most operations have a data dependent power consumption.

The next sections describe a few of the possible estimations techniques for power estimation of a NoC. The first set of tools are Synopsys' software packages that can estimate the power consumption of a synthesized design. The second tool, Orion, is a dedicated tool for NoCs.

6.1.1 Synopsys PrimeTime PX / Power Compiler

Synopsys offers a set of tools that enable the designers to optimize and estimate the power of their designs at various levels. The tools can be used at architectural (RTL) and gate level of the logic design. It consists of an analysis tool (PrimeTime PX) and an optimization tool (Power Compiler) [128].

PrimeTime PX can perform the power analysis using an event-based or a statistical-activity-based method. The former uses the time-based activity of the nets in the circuits. The activity can be determined by running a simulation of the RTL description or gate level description with realistic input data. For the gate level description it is possible to include delay information that is available before and

after placement and routing. The designer is able to observe both average, peak and detailed time-based power of the simulated design.

The latter method uses statistical information of the nets. The statistics consist of the percentage of time a net was either one or zero and the amount of transitions between the two levels. The statistics can be determined by simulation or by manually annotation by the designer. Un-annotated nets are annotated by the tool via probability estimation and known annotations.

In both methods, the tooling combines the activity, with the netlist, design constraints and the power figures given by the standard cell library used. In the standard cell library the dynamic and static power consumption of a specific gate are given for a large number of cases. Each case is a unique combination of, for example, the gate's driving strength, input vector's values and transitions, output transition, and output capacitance. The netlist can either be a gate level design, where wireloads and timing are coarsely estimated by the synthesis tool, or a placed and routed design, where wire parasitics, clock tree and delays are included in the power estimation.

Power Compiler is a tool that can assist the designer to optimize its design for power. It can analyse the RTL description of the design and insert clock gating components where possible. During synthesis it can optimize the design not only for area and timing, but for power as well. Based on the designer's constraints it can choose the optimal gate in terms of power. Furthermore, it makes it possible to support multiple threshold libraries.

Other tools that offer similar capabilities as PrimeTime PX and Power Compiler are PowerTheater [42], which is Sequence Design's power tool, PowerMeter, which is part of Cadence's power analysis solution VoltageStorm [129]. XPower Analyzer is Xilinx's power tool, that enables designers to analyse the power consumption of their FPGA designs [139].

6.1.2 ORION

Orion is positioned as a power/performance interconnection network simulator that is capable of providing detailed power characteristics, in addition to performance characteristics, to enable rapid power/performance trade-offs at the architectural-level [133]. The goal of the simulator is to enable the designer to make detailed performance trade-offs early in the design cycle.

The simulator has a set of basic building blocks (modules) for an interconnection network. Examples of building blocks are buffers, crossbars, arbiters, links and message sources and sinks. With a small number of modules the designer should be able to model a large variety of networks. For the buffers, crossbars and arbiters, an architectural-level parametrised power model is derived. Based on a set of component, technology- and architectural parameters, the switch capacitance equations are determined. These equations are combined with the traced switch activity of a component through network simulation such that the dynamic power can be calculated. A leakage model based on subthreshold leakage (eq. (A.3)) was included at a later stage [27]. The power models can be extended by constructing

hierarchical models from the basic modules.

The Orion estimated power is compared to the preliminary power estimates from the designers of the Alpha 21364 router and Infiniband router. The Orion tool estimated 5.36 W for the Alpha 21364 router core compared to a 7.6 W estimation by the designers. According to Wang et al. [134]: "This is within the ballpark of the designers' estimate." Currently, Orion is extended with area estimates, pipeline delay models and a statistical mode is added to increase performance. This new release is incorporated in Polaris, a system-level roadmap for on-chip networks that guides designers towards a subset of most suitable candidates for on-chip network designs while considering the complex trade-offs between applications, architectures, and technologies [126].

6.1.3 Other Methodologies

Banerjee et al. attempted to increase the accuracy of power estimates results by first extracting a SPICE level netlist from a synthesised design [12]. This was then used to develop power models for NoC router components which then provided power results under random traffic simulations. However, only a limited range of architectures were used which limited the results that could be obtained.

A similar power measurement methodology is provided by Xi and Zhong [138]. High-level power models were again derived and then embedded into a Transaction Level Modelling (TLM)-based simulation framework. The authors also extended the work into future technology nodes using the Berkeley Predictive Technology Models. This allowed key predictions to be made for future technologies, but the questionable accuracy of the high-level models still remains a problem.

6.1.4 Measurements by Others

Dielissen et al. presented a power comparison of the Æthereal NoC and a bus based system [44]. For the NoC various measurements were performed to determine the effect of traffic types, packet lengths, path and packet interaction. The power is measured for a 0.13 µm CMOS technology, nominal process operating at V_{dd}=1.2 V, 25°C and a data path of 32 bits. At a worst-case estimation for random payload (activity of 0.5) with GT traffic, the router consumes 68.25 pJ per flit of which 32 pJ is consumed by the clock tree. For BE traffic an extra 15.7 pJ/flit is required.

Lee et al. reported a detailed power analysis of a high-performance SoC using a NoC for communication between heterogeneous intellectual property blocks such as RISC, SRAM and FPGA units [85]. The 25 mm² chip, realized in 0.18 µm CMOS technology, consumes 160 mW of which 51 mW is consumed by the NoC. For a five port switch with packets consisting of 32 bit data, 32 bit address and 16 header fields, the reported packet energy is 229 pJ per packet of which 86% is consumed by the queues.

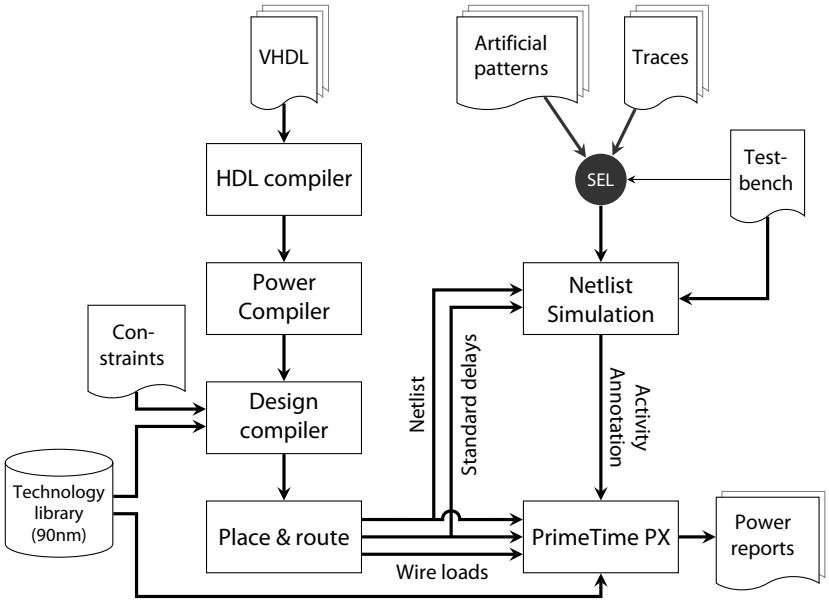


Figure 6.1 – Measurement setup

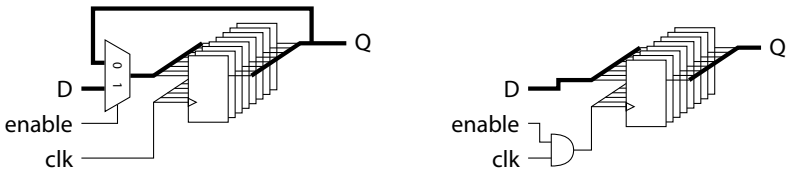
6.2 Measurement Flow

Characterizing the power consumption of an architecture is possible with several methods. As described in section 6.1 each method has a trade-off between time to setup experiments, time for each estimation, accuracy and the level of detail.

We have chosen to use the Synopsys PrimeTime PX flow as depicted in figure 6.1. This specific flow delivers the most accurate results out of the several ways to obtain energy figures using PrimeTime PX. In this flow PrimeTime PX requires five inputs:

1 netlist
2 standard delays of the wires and instantiated cells
3 wire loads
4 activity annotation
5 technology information

The first three are obtained from a standard synthesis design flow including placement and routing of the architecture. In this flow the VHDL sources, or other HDL specifications, are analysed. After analysis, PowerCompiler can be used to insert clock gating elements to reduce the energy consumed by the registers and clock tree. This requires the presence of appropriate load-enable expressions in the original design sources. Figure 6.2(a) depicts an example that is modified by PowerCompiler to figure 6.2(b), where the large multiplexer is replaced by an AND-gate which gates the clock signal. The AND-gate in this example could be replaced by a specific



(a) Original design (b) After clock gating

Figure 6.2 – Modification of design by PowerCompiler

clock gating cell of the technology library used. The clock gated design can then be synthesized using a specific technology library and constraints in Design Compiler. The last step of the synthesis flow is the placement and routing of the gate-level netlist. The resulting netlist, standard delay information and wire load information is given to PrimeTime PX.

The fourth input required by PrimeTime PX is the activity annotation of the netlist. This activity information is obtained by simulating the actual netlist and including its delay information. Simulation of the placed and routed netlist is slow, but gives the most accurate activity information including possible glitches.

The stimuli of the router's netlist are either generated by traffic generators that create artificial traffic patterns or a trace from a specific router's input obtained via a cycle and bit accurate simulation run of the NoC. This cycle and bit accurate NoC simulation can be performed by the simulator as described in chapter 8, where the input traces of a specific router are extracted.

The last input required by PrimeTime PX is the technology information of all standard cells used. Via this separate technology information input, it is possible to synthesize the design for worst-case conditions (high temperature and low voltage), but use the power consumption of each cell at typical-case conditions (room temperature and nominal voltage). Worst-case conditions for timing analysis are not the worst case for power analysis as they dictate a lower power supply voltage than used in the typical-case.

6.2.1 Obtained Output Reports

The reported power consumption results consist of two types: static and dynamic. The static power consumption is the power dissipated by a gate when it is not switching. The dynamic power is the power dissipated when the circuit is active. The dynamic power is composed of two kinds of contributions: switching and internal cell. The switching power of a driving cell is the power dissipated by the charging and discharging of the load capacitance at the output of the cell. The internal cell power is any power dissipated within the boundary of a cell.

Various details can be obtained from the output reports. The most detailed

report contains the power consumption versus time of the individual entities in the design. Based on these figures, the average power consumption during the simulated time interval is presented for all design hierarchies in overview reports. Furthermore, more specific reports can be generated to examine the power consumption of for example: the clock tree, a specific design cell like registers or the scan chain.

6.3 Measurement Setup

The results in this chapter are targeting the 90nm TSMC high-performance technology library (TCBN90GTHP). This library has a supply voltage of 1.2 V and a nominal threshold voltage. The typical-case conditions are selected for all measurements. In contrast with synthesis and static timing analysis the worst case operation conditions do not result in the worst case for power measurements as they dictate a lower supply voltage. All measurements were performed at 250 MHz.

We expected that the power consumption of a single router is at least dependent on the following parameters:

1. The average load of every individual data stream. This varies between 0% and 100% of the available bandwidth of a single lane or VC.
2. The amount of bit-flips in the data stream. This varies from no bit-flips (i.e. transmitting constant values) to continuous bit-flips (i.e. alternating zeros and ones for all bit positions). Specific values are presented in chapter 3 for HiperLAN/2, DRM and MPEG-4. When interleaving two arbitrary bit streams, both with an equal number of 0's and 1's, the amount of bit flips averages to a probability of ½ as derived in appendix C.
3. The number of concurrent data streams through the router, which, in our case, has a maximum equal to the number of virtual channels for the packet switched router or number of lanes for the circuit switched router. For both routers this is equal to 20 streams.
4. The amount of required control and arbitration. For example, the packet switched router can handle both GT and BE traffic concurrently.

Therefore, several tests have been performed. These are grouped in the following sections. The initial power consumption was obtained offering no traffic to the various router designs. This indicates the minimum power consumption of the architecture.

6.3.1 stimuli generation

The stimuli that are applied to the router under test are injected in the router via a VHDL-based testbench which reads files that contain the necessary stimuli information. The content of the files either is generated by a script or the FPGA-based simulator. The script generates artificial traffic patterns that represent a certain load in the network for a specific link. The second option makes use of the logging functionality of the simulator, which can monitor the input links of a specific router

Table 6.1 – GuarVC router versions

Version	Clock gating	VC queue enable	Output latched
A	no	coarse	no
B	yes	coarse	no
C	no	fine	no
D	yes	fine	no
E	yes	fine	yes

in the simulated NoC. Both initial router's state and all bit transitions on the link are extracted from the FPGA into the stimuli files that could be used by the testbench.

6.4 Results

In this section we present the energy measurements performed on the two router architectures—CS and GuarVC—as presented in chapter 4. For both routers we use the 64 bit versions of the architecture.

For the GuarVC router we use the five variations of the router architecture as discussed in section 4.1.4. The configuration of the five router architectures are listed in table 6.1. For each version we either enabled or disabled the automatic clock gating insertion during synthesis. Version A and B have an input queue description for which the Power Compiler tool detected a single gating possibility per VC queue. The other three versions have a queue that can be clock gated per flit position. The E version of the GuarVC router also has a latch at the output ports, which is only active during the second half of the clock cycle and if the crossbar allocator has granted a request for this port. When the version of the GuarVC router is not mentioned, we use this last version of the router architecture.

6.4.1 Idle power consumption

An initial power characterisation of the designs was obtained by offering no traffic to a single router. The measured power is from now on referred to as *standby power*. The clock signal is enabled, reset was activated preceding the measurement and the dissipated power is measured for 5000 clock cycles. This idle test is performed to examine the offset in energy costs of a NoC. This will give information about the constant energy cost to deploy a NoC in a SoC. Besides the costs caused by leakage of the individual gates, some extra dynamic energy is required due to the activity of the clock.

Tables 6.2(a) and 6.2(b) present respectively the dynamic and static power consumption of the routers under the standby condition. For the architectures with no clock gating (GuarVC version A & C), the major component of the idle power is by far the dynamic power. Of the dynamic power a large part is consumed by the top-level clock tree. The major second part is consumed by the local clock tree and

Table 6.2 – Standby power consumption per component of the different routers

(a) Dynamic power [mW]

Router	Top level clock tree	Crossbar	Flit buffers + logic	Arbitration	converter Data	BE router	Other	Total
GuarVC (A)	10.81	0	42.05	0.06	–	–	0.31	53.33
GuarVC (B)	1.14	0	0.68	0.02	–	–	0.06	1.90
GuarVC (C)	14.30	0	46.67	0.06	–	–	0.06	61.09
GuarVC (D)	1.40	0	1.24	0.02	–	–	0.06	2.72
GuarVC (E)	1.90	0.04	1.31	0.02	–	–	0.08	3.34
CS	0.80	0	0	0.14	0.55	0.28	0	1.77

(b) Static power [mW]

Router	Top level clock tree	Crossbar	Flit buffers + logic	Arbitration	converter Data	BE router	Other	Total
GuarVC (A)	0.18	0.48	2.47	0.21	–	–	0.21	3.55
GuarVC (B)	0.02	0.36	2.39	0.21	–	–	0.12	3.10
GuarVC (C)	0.25	0.40	1.73	0.21	–	–	0	2.60
GuarVC (D)	0.02	0.41	1.81	0.20	–	–	0	2.44
GuarVC (E)	0.03	0.36	1.86	0.21	–	–	0.04	2.49
CS	0.02	1.00	0.10	0.06	0.33	0.15	0.10	1.77

the connected registers of the input buffers + logic, and arbiters. For all designs a minor part, less than 0.3 mW, is consumed by the combinational logic.

The static leakage component becomes equally important when clock gating is enabled. This leakage component could be reduced with leakage minimisation techniques as discussed in section A.2 The relative share of the individual blocks is strongly related with the amount of logic, i.e. area, per block (see figures 4.8 and 4.14).

For the GuarVC router the reduction in leakage of the input buffers—versions A & B versus C & D—is caused by other registers[1] used for the input queues, due to a slight modification of the hardware description, as described in section 4.1.4, that is functionally equivalent. Because of this modification, PowerCompiler places a clock gating cell per flit position instead of per queue. This makes the clock gating

[1]The versions A & B of the designs have D-flipflop with an enable port and the other three version of the designs do not have this enable port.

fine grained instead of coarse grained. The increase in the number of clock gating cells causes a slight increase in dynamic power consumption. However, in the next sections we will see that the energy per flit is substantially lower.

6.4.2 Energy consumption under no congestion

A second test is the energy consumption of the router under no congestion. Both routers as described in chapter 4 are tailored for GT traffic. In this test we offer the router multiple streams that do not cause long-term congestion of other streams. Minor congestion is allowed, for example, two VCs can request an output port at the same time, but the VCs will not request more than their guaranteed bandwidth.

Continuous Streams

In the first set of tests the resources for each stream are reserved at the start of the simulation. The header flits and configuration data to setup the streams for respectively the GuarVC and CS router are injected into the network before the energy measurement is started. This visualizes solely the energy required to transport the actual data of the streams. Per test the routers is offered respectively 1, 2, 3, 5, 10, 15 or 20 streams of data. Each stream is mapped on an unique VC or lane in case of the circuit switched NoC. An initial 50 clock cycles were used as a *warm-up* time, with the next 2500 clock cycles forming the *sampling time* for the power analysis.

The average load per stream is equal for all streams in the same test. It varies per stream between 5, 10, 15, 20 and 25% of the maximum link bandwidth for the GuarVC router, and between 20%, 40%, 60%, 75%, and 80% of the maximum lane bandwidth for the CS router. The maximum of 80% of the lane bandwidth is due to the 20% overhead for the serialization. The flits for each stream are randomly distributed in time. The payload of the streams is either a sequence of random values or all zeros. The difference between a random and 'empty' payload will give an impression of the balance between data and the control part of the routers.

In figure 6.3 the total power consumption is presented versus the average traffic load for all five GuarVC router designs. Only the measurements with random payload are depicted. Two groups of marks are clearly visible, those with (white) and those without (grey) the insertion of clock gating.

In figure 6.4 the total power consumption is presented versus the average traffic load per stream. Only the measurements with random payload are depicted. The CS router does not have an enable signal for the pipeline registers at the output ports of the router. Those registers can only be clock gated while the corresponding port is not configured for a stream. Figure 6.4(b) depicts the increase of the CS router's standby power due to the increase of the number of configured streams. Therefore, the representation of the results is different compared to the GuarVC router.

From the numbers of both GuarVC and CS router, we can conclude that the increase in power is caused by the granularity of the clock gating and increase in load. The difference between the same GuarVC design type with and without

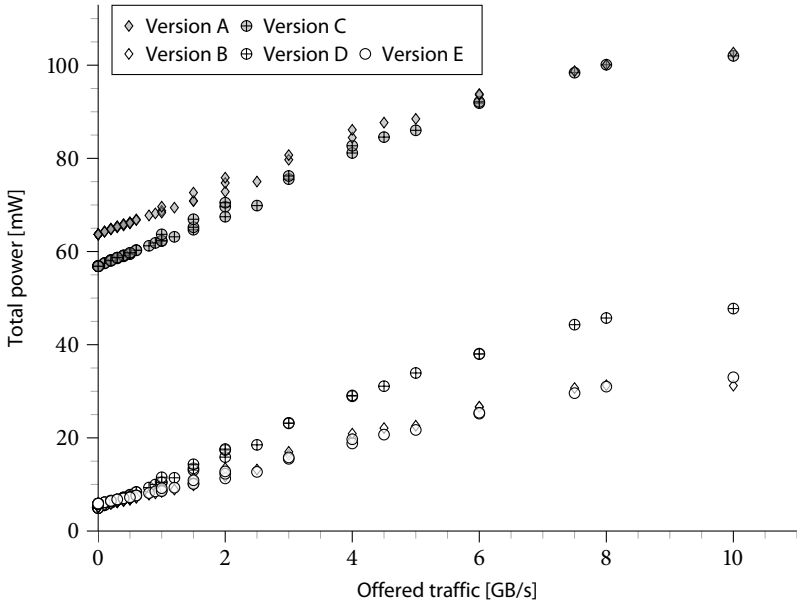
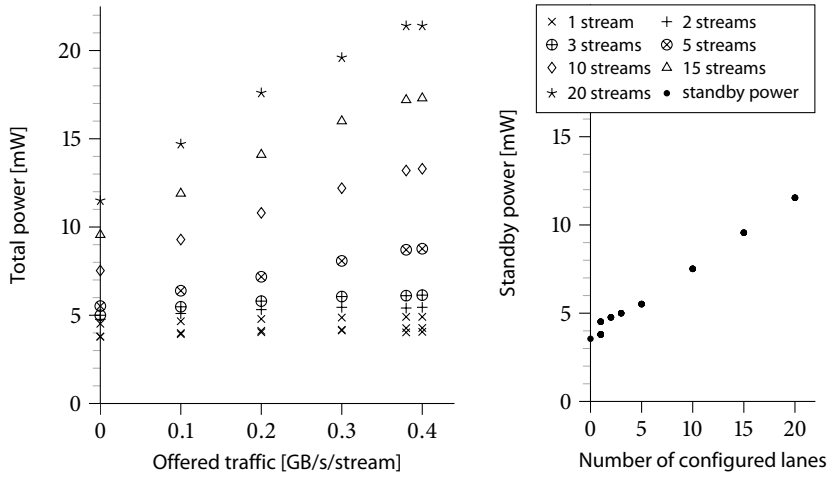


Figure 6.3 – Total power consumption of continuous traffic streams for the five GuarVC router versions at a frequency of 250 MHz and random payload. *Measurement conditions:* Clock frequency: 250 MHz. Voltage: 1.2 V. Library: 90 nm TSMC. Number of cycles: 50 (warm-up), 2500 (sampling). Traffic: artificial flits per stream, randomly distributed in time. Streams allocated at start of simulation. Variable between tests: number of active streams & offered traffic per stream.

clock gating—A versus B & C versus D—is nearly constant for every load. If the design written in a way that enables the PowerCompiler tool to find clock gating possibilities at a fine grain level—A versus C & B versus D—, the increase in power due to a higher load is lower. However, as shown in table 6.2(a), the standby dynamic power is considerable higher in case no automatic clock gating is inserted (version C).

From figure 6.3 and 6.4 we can conclude, that the increase in power is proportional to the increase in traffic load, except for very high loads. A slight increase in power consumption is noted for the packet switched router in case multiple VCs are used per output port. A small extra saving is observed for the GuarVC design with an extra latch (version E) at the output ports of the crossbar to reduce the switching on the outgoing links.

The increase in power for the CS router can be split in the number of streams that are configured and the increase in traffic load per stream. The additional standby power due to the activation of an extra circuit within the CS router can be derived from figure 6.4(b). This power equals 0.24 mW/lane for connections to neighbouring routers and 0.97 mW/lane for connections via the data converter to



(a) Total power

(b) Standby power

Figure 6.4 – Total power consumption of continuous traffic streams for the CS router at a frequency of 250 MHz and random payload. *Measurement conditions:* See figure 6.3

the local tile and has an average value of 0.39 mW/lane.

Because an affine relation between throughput and total power is observed, we plotted the dynamic power excluding the standby power of table 6.2(a) and figure 6.4(b) versus the offered traffic on a logarithmic scale for all four clock gated designs, which is depicted in figure 6.5. This figure clearly shows the linear relation between the increase in traffic load and the extra required power. It also shows the benefit of clock gating per buffer position, which reduces the extra required power by approximately 39%. Furthermore, a slight energy saving is seen for the latched output port, because the router only changes its output if new valid data is scheduled. This latter saving is mainly noticed at low traffic conditions.

In this plot we also depicted the extra required dynamic power for flits with an 'empty' payload for the latched GuarVC design and the CS router. The transport of 'empty' payload flits for the CS router requires almost nothing, because of the total separation of data and control in this router. Its control logic is not active during the tests. The small amount of extra required dynamic power is consumed by the data converter. For the GuarVC router, the transport of 'empty' payload flits requires only 31% of the power compared to the transport of random payload flits. The 'empty' payload results of the other GuarVC designs are left out for clarity, but have a similar ratio between random and 'empty' payload.

Because we observe an affine relation between the throughput and power consumption we can calculate the energy per bit that is required to transport a bit from input to output port of a router. Per test we subtract the standby dynamic power

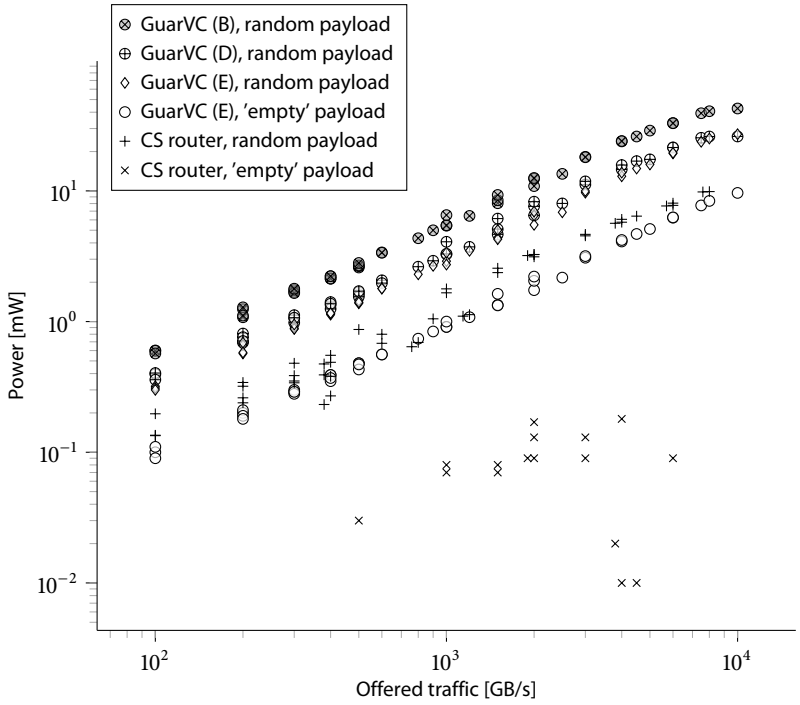


Figure 6.5 – Dynamic power consumption (without constant component of table 6.2(a)) of clock gated router designs. *Measurement conditions:* See figure 6.3.

of table 6.2(a) from the total dynamic power and divide the result by the average throughput of the test. The resulting value is the average energy per bit for the specific test. For the GuarVC (version E) and CS router these values are depicted in figure 6.6. The average energy per bit for all tests is depicted with a grey line.

Using the results as presented above, we can describe a simple power model for both the GuarVC and CS router. Each GuarVC router has a standby power of $2.49 + f \cdot 1.336 \cdot 10^{-2}$ mW, where f is the frequency of the router in MHz. The additional dynamic energy to transport packets with random data equals 0.38 pJ/bit. Each CS router has a standby power of $1.77 + f(0.708 + 0.16l) \cdot 10^{-2}$ mW, where l is the number of configured lanes. The additional dynamic energy to transport packets with random data equals 0.18 pJ/bit.

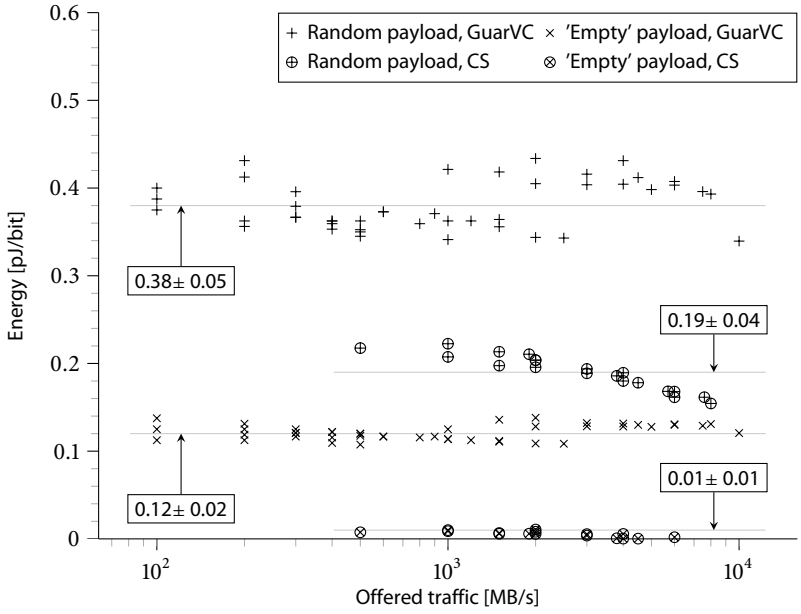


Figure 6.6 – Energy per bit (figure 6.5 divided by the throughput). *Measurement conditions:* See figure 6.3.

Streaming Packets

The second set of tests for uncongested traffic was obtained by streaming data at a fixed rate through a single router and measuring the dissipated power. Four fixed traffic streams were defined, one originating at each of the North, South, East and West ports of the router, with each one transmitting a stream of packets to the opposite router port. Per test one to four traffic streams are selected. In contradiction to the previous tests, where data was grouped in streams of infinite length, the data is grouped into packets with a random payload of 256 bits.

The average data rate of each stream was set to a moderate 30% of the maximum bandwidth of a single router link, with packets being sent at randomised intervals. For each experiment, an initial 500 clock cycles were used as a *warm-up* time, and power analysis was performed for the successive 5000 clock cycles. For simplicity, the CS net was only configured once, at the start of the experiments. This clearly represents a best case scenario for this network. For the GuarVC the latched version with the finest clock gating was selected. The data rate of 30% for the GuarVC equals the net data rate of only the data payload carrying flits of the packet. A single three hop header is added per packet to route it through the router, which results in a gross data rate of 37.5%.

The measurements for the GuarVC were extended with a modified version of the test. In this third set of tests, the routing information was sent with the first

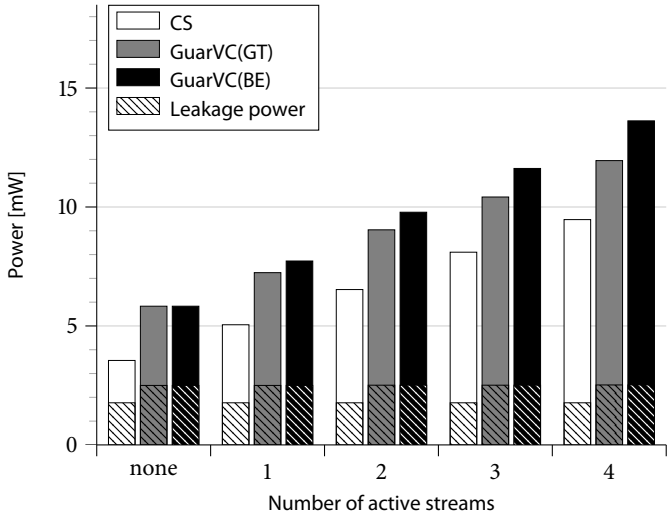


Figure 6.7 – Router power for streaming packets *Measurement conditions:* Clock frequency: 250 MHz. Voltage: 1.2 V. Library: 90 nm TSMC. Number of cycles: 500 (warm-up), 5000 (sampled). Traffic: artificial packets with a payload of 4 flits/packet, randomly distributed in time. Variable between tests: the number of active streams.

packet and the following packets on that VC use exactly the same route, such that no extra packet headers are required. These latter tests represent GT type of traffic.

Figure 6.7 depicts the router power results of these experiments for the two routers. The leakage power is indicated by the shaded areas of the individual bars, which is constant for all tests. Similar to the calculated energy per bit, we can calculate the energy per packet for these tests. For all tests we counted the total number of packets that are transported by the router and the increase in energy demands to transport these packets. This extra required energy during the 5000 cycles divided by the number of packets results in the energy per packet which is respectively 80.3 pJ/packet for the CS, and 80.0 pJ/packet (GT packets) and 102.6 pJ/packet (BE packets) for the GuarVC router.

The energy per packet for the CS router is higher in comparison with the earlier presented energy per bit, because energy per packet also includes the power for the configured number of lanes. The energy per packet for the GuarVC router reduces by 22 pJ if the router handles solely GT traffic due to the absense of a header flit per packet. The contribution of the various components of the router to this packet energy is depicted in table 6.5, where the two routers are compared with two other packet switched routers.

6.4.3 Energy consumption under congestion

The packet dynamic energy cost reported in section 6.4.2 does not account for any network congestion. Clearly, it is of interest to see how congestion of packets

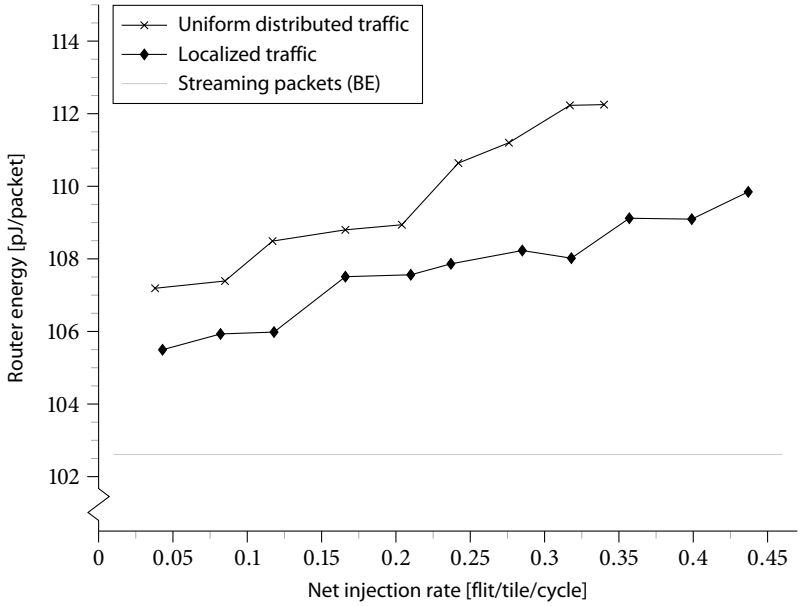


Figure 6.8 – Packet energy under congestion. *Measurement conditions:* Clock frequency: 250 MHz. Voltage: 1.2 V. Library: 90 nm TSMC. Number of cycles: 500 (warm-up), 5000 (sampled). Traffic: traffic traces from latency measurements as depicted in figures 5.7 and 5.8. Variable between tests: Average inter-packet time between BE packets of each source.

will affect the packet energies in a packet switched network. For the GuarVC this was achieved by instantiating an 8 × 8 mesh network. For the CS network, the current lack of dynamic circuit set-up and tear-down support means that this form of congestion energy experiment cannot yet be performed. The latency tests, as described in section 5.3, were performed where the packets each carry a 256 bit (i.e. four flits) random data payload.

A single router, at coordinates $x = 4$, $y = 4$, was considered and the average energy of any packet going through it was calculated in the same fashion as in section 6.4.2. In contradiction to the relatively fast FPGA-based simulations for the latency analysis, the power analysis requires a considerably longer time. Therefore, the trace of the router's input signal was logged for a limited number of clock cycles. An initial 500 clock cycles were used as a *warm-up* time, with the next 5000 clock cycles forming the *sampling time*, all packets transported by the selected router in this period were the only ones considered in the analysis.

Figure 6.8 shows the packet energies of the GuarVC router for various injection rates and two traffic distributions, uniform and localised, of the source-destination pairs. As a reference the grey line indicates the average energy of a streaming packet as described in the previous section. The values represent the energy required to forward one packet through the router. In both distribution cases the energy per

packet only slightly, less than six percent, increases as the total network traffic and congestion increases. The saturation of latency due to the increase in network load is not observed for the energy per packet. Although packets spend more time in the network queues under congestion, this doesn't cause an extra energy consumption of the data path, because of the effective clock-gating. The small increase is mainly caused by the increase in allocation and flow control activity, which increases due to the flit-based arbitration.

The increase in the packet energy at low injection rates compared to the average energy of streaming packets can be explained by the increase in header information for an average packet. In the streaming packets the header contains 3 hop (18 bits) 'random' information and the remaining part of the header is 'empty'. For this congestion test the average header contains more random routing information and as a consequence fewer 'empty' bits. For uniform distributed traffic the average header information in packets through the router under test equals 6.8 hops (40.7 bits) and for localised traffic the average header size is 4.8 hops (28.8 bits). This increase in random bits versus empty bits causes the small increase in energy per packet, because the cost for transporting random data is higher compared to 'empty' data as shown in figure 6.6. This also explains the lower energy per packet for localised traffic, because the average header information is less.

6.5 Comparison

Besides the two architectures described in this thesis the same measurements were preformed for two other packet switched alternatives—the WH and SpecVC router, as described in section 4.3—under the same conditions. These measurements were performed at a frequency of 200 MHz and therefore the numbers given in the previous section are scaled to this 20% lower frequency, which only influences the dynamic energy costs. Besides the comparison of the router power also the link power is included to complete the comparison. In a joint cooperation with the University of Cambridge the total energy of the various router and link configurations was determined.

All tests use packets with a random data payload of 256 bits. Energy is reported per packet in this section, because a packet contains both routing information as well as payload. A coarse estimation of the energy per bit is calculated by dividing the packet's energy by its payload size. However, this neglects the variable ratio between header and payload size.

6.5.1 energy consumption of wires

The performance of logic can be determined by modelling the design in a hardware description language, synthesize and layout the design and using Synopsys PrimeTime PX [128] for power estimation as is described in section 6.1.1. This tool does not include the long wires between the individual analysed logic blocks. This would require the layout of the complete NoC such that wire loads between the routers

are known. For (long) wires between the logic blocks, we use an analytical model of a wire, which is described in this section.

For the power figures of a wire we include the drivers and repeaters that are required in a link between two routing structures. Morgenshtein et al. [95] determine the power of a link between two routers by:

$$P_{link} = \left(P_{drivers} + P_{repeaters} + P_{wire}\right) \cdot N_{wires} \tag{6.1}$$

where N_{wires} is equal to the number of parallel wires of the link.

The power of the three individual parts depends on the configuration (e.g. number of repeaters), environment (e.g. capacitive coupling with other wires and metal layers) and length of the link. The minimum amount of energy required to transfer a bit over a wire of length l_{wire} is determined by the capacity of the wire itself. For 0.13 µm technology this capacity is 240 fF/mm and for 100 nm the capacity is 154 fF/mm [11]. For differential signalling Mensink et al. [91] reported 250 fF/mm and 280 fF/mm for respectively 0.13 µm and 90 nm. The total energy required to fully charge and discharge (i.e. two transitions) this capacity equals CV^2 which is respectively 0.35 pJ/mm and 0.22 pJ/mm for a supply voltage of 1.2 V. In case of random data, each bit has a transition probability of 50%, thus the minimum energy per bit equals for 0.13 µm and 0.10 µm respectively 0.09 pJ/bit and 0.055 pJ/bit for a 1 mm wire. The extra energy is determined by the size of the driver and the extra energy consumed by repeaters.

Banerjee and Mehrotra [11] showed that the dynamic power consumption of a link segment (i.e. the wires between two repeaters including the driver) is equal to:

$$P_{link_{dyn}} = \left\{\alpha \left(s\left(c_p + c_0\right) + c \cdot l_{wire}\right) V_{DD}^2 f_{clk}\right\} \cdot N_{wires} \tag{6.2}$$

where α is equal to the switching factor (or activity factor), l_{wire} the length of the wire segment in mm, c_0, c_p the input and output capacitance of a minimum sized repeater, s the repeater size relative to a minimum sized repeater, and c the capacitance of the wire segment. For various technology nodes the optimal delay per unit length is determined. A segment of length l_{wire} = 2.5 mm and repeater of size s = 151 is optimal in 0.13 µm technology. For a minimum sized global interconnect, c = 240 fF/mm, repeater capacities of c_0 = 1.7 fF and c_p = 3.5 fF and random data (α = 0.25) this results in an energy consumption of 0.50 pJ/bit/segment, of which 57% is consumed by the repeater.

Mullins [96] examined various configurations of 1.5 mm links placed on metal layer three for 90 nm technology. For random data vectors the average energy is reported as energy per transition, per mm, averaged over 500 random input vectors. The energy varies between 0.13 pJ/transition/mm and 0.34 pJ/transition/mm depending on the link's configuration and either optimisation towards a minimum delay or optimal energy x delay product. In random data the number of transitions equals half the number of bits in the vector, which results in energy estimations of 0.06 pJ/bit/mm to 0.17 pJ/bit/mm.

In contrast with repeater based links, Mensink et al. [91] proposes a low-swing differential scheme with a capacitive pre-emphasize transmitter. Differential links

Table 6.3 – Energy consumption per transported bit over a 1 mm long wire with a supply voltage of 1.2 V

Configuration	Technology [nm]	Dynamic energy [pJ/bit/mm]
Full swing, theory	130	0.09
Full swing, theory	100	0.055
Banerjee	130	0.2
Mullins	90	0.06 – 0.17
Mensink & Schinkel	90	0.028
This chapter	90	0.18

with twists are used to prevent crosstalk [114]. The energy consumed for random data with a data-rate of 2 Gb/s over a 10 mm link is 0.28 pJ/bit, which is considerably lower than the theoretical minimum for a full-swing link. Of this energy figure 0.16 pJ/bit scales proportional with the switch activity of data and 0.12 pJ/bit is activity independent. The energy consumption per bit increases at lower data rates.

To estimate the link power, the number of transitions on the output ports of the router designs are counted and multiplied with the average energy per transition per mm. The characterisation of this average energy was performed separately from that of the routers. For this characterisation, we use a similar approach as Mullins [96], and is performed by the University of Cambridge. Links of length 1.5 mm, based on intermediate metal layers (M3–M6), were used. The Quickcap field-solver tool from Magma [88] was then used to extract link capacitance values with an 8-wire model. The energy/delay trade-offs of various link repeater configurations were then analysed with SPICE simulations. Ultimately, instead of using a delay-optimal repeater configuration, a lower energy configuration with 9.7 $FO4$ delay[2] with an associated 0.36 pJ/transition/mm for the links was selected for this comparison of routers.

Other values could have been derived with, for example, equations as discussed above, but SPICE simulations were chosen to get the same level of accuracy. Table 6.3 depicts a summary of the energy costs of transporting a random bit over 1 mm wire.

6.5.2 Standby Power

Identical to measurements in section 6.4.1 the standby power is determined for the SpecVC and WH packet switched routers. The breakdown of the standby power is reported in table 6.4 for the four router designs that are compared. All the routers dissipated a significant amount of power where the ratio between dynamic and static power is larger for the WH and SpecVC designs. For all designs, the leakage power is directly correlated with the area of the router design, because of the lack of any leakage minimisation techniques. The dynamic power is primarily dominated by the activity of clock tree that is connected to the clock pins of the clock gating

[2] One $FO4$ delay is the delay of a single inverter driving four identical inverters.

Table 6.4 – Standby power breakdown

Component	Standby Power [mW] Dynamic | Static			
	CS	WH	GuarVC	SpecVC
Top level clock tree	0.64 | 0.02	1.65 | 0.04	1.52 | 0.03	3.61 | 0.11
Flit buffers + logic	0 | 0.10	1.27 | 0.93	1.05 | 1.86	1.04 | 2.30
Crossbar	0 | 1.00	0.09 | 0.28	0.03 | 0.36	0 | 0.01
Crossbar allocator	0.11 | 0.06	0.05 | 0.09	0.01 | 0.21	0.67 | 0.18
VC allocator	–	–	–	1.05 | 0.23
Output ports	–	–	–	0.20 | 0.20
BE router	0.22 | 0.15	–	–	–
Tile interface	0.44 | 0.33	–	–	–
Other	0 | 0.10	0 | 0.06	0.06 | 0.04	0.02 | 0.12
Total	1.41 | 1.77	3.06 | 1.41	2.67 | 2.50	6.60 | 3.15

elements and non-clock gated synchronous elements. The SpecVC has obviously the largest amount of clock gating and synchronous elements, due to its VC input queues and speculation logic, and the CS router the lowest, due to its small amount of registers in the architecture. Because the data path is both in area and in number of synchronous elements the largest part of all four router designs, it also the major contributor of the standby power consumption.

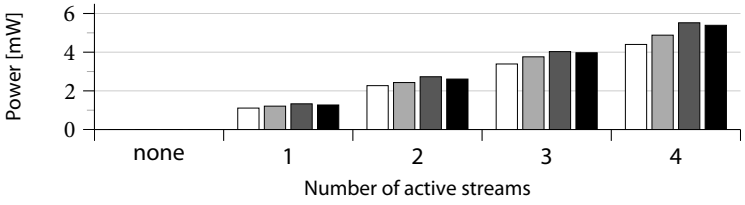
6.5.3 Streaming Packets

For streaming packets the WH and SpecVC routers are offered the same type of packets as described in section 6.4.2. Both the total power, number of packets and the transitions on the output links are measured.

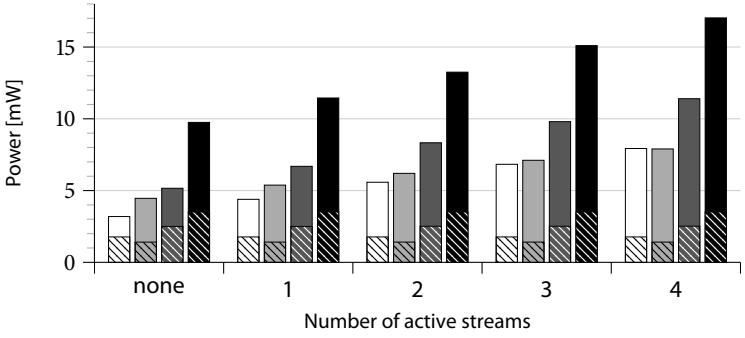
Figure 6.9 depicts the link, router and total power results of these measurements for all of the four routers. The leakage power is indicated by the shaded areas of the individual bars. For all designs the total router power is more than the link power and especially for the SpecVC design. The link power is directly related to the total size of a single packet, which results in the highest power consumption for the GuarVC links due to the extra header flit per packet and the lowest for the CS links.

For all routers we calculated the energy per packet with a payload of 256 bits, as described in section 6.4.2, which is depicted in figure 6.10. A breakdown of the router's energy per packet is reported in table 6.5. The data path components are the flit buffers and the crossbar, and the control path are the other components with the exception of the top level clock tree.

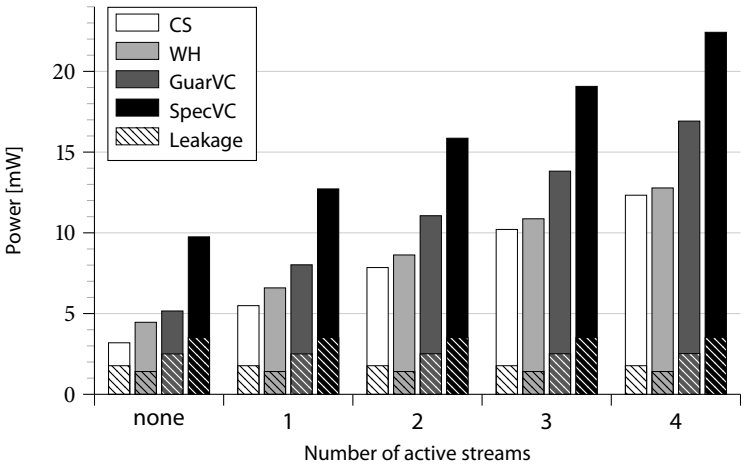
The most important result of the comparison of the packet energy of the different designs is that they are not orders of magnitude different. Initially our hypothesis was that the energy for the CS router would be a lot small due to the smaller number of required buffers. Furthermore, the SpecVC would have the largest consumption due to the extra additional control logic that optimize the average performance.



(a) Link power



(b) Router power



(c) Total power

Figure 6.9 – Link and router power at fixed throughput *Measurement conditions:* see figure 6.7.

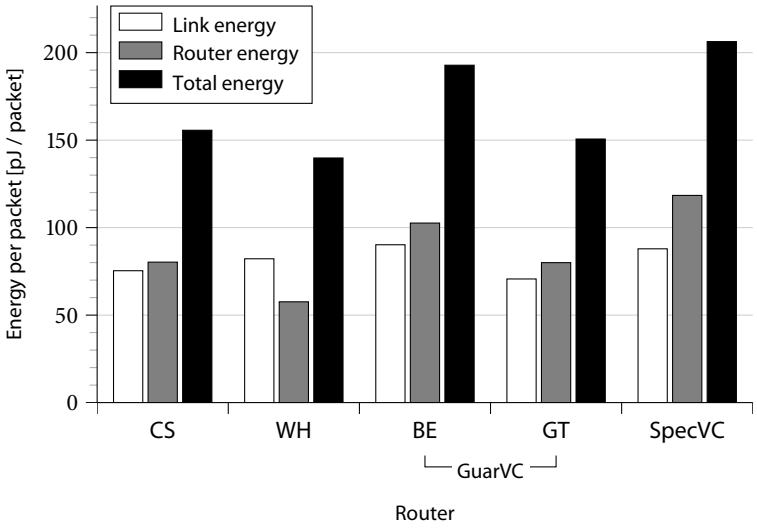


Figure 6.10 – The packet energy for streaming traffic *Measurement conditions:* See figure 6.7.

Table 6.5 – Streaming traffic packet energy breakdown

Component	Packet energy value [pJ] | relative share [%]			
	CS	WH	GuarVC	SpecVC
Top level clock tree	0	0	0.01 | 0.0	5.98 | 5.1
Flit buffers + logic	28.1 | 47.7	45.72 | 79.4	64.10 | 62.5	58.2 | 49.2
Crossbar	37.9 | 35.4	6.62 | 11.5	31.69 | 30.9	15.5 | 13.1
Crossbar allocator	4.65 | 5.9	2.94 | 5.1	2.84 | 2.8	7.27 | 6.1
VC allocator	–	–	–	1.92 | 1.6
Output ports	–	–	–	8.99 | 7.6
BE router	0	–	–	–
Tile interface	0	–	–	–
Other	8.77 | 11.0	2.33 | 4.0	3.98 | 3.9	20.5 | 17.3
Total	79.4 | 100	57.61 | 100	102.61 | 100	118.4 | 100

The small difference in packet's energy is due to the fact that the data path components dominate over the control elements in all designs, especially in the CS and WH nets. Of the data path components, the flit buffers consume a large fraction of the total energy. Moreover, it is interesting to see that the buffer energy is not proportional to the amount of buffering in the designs, but to the communication activity. This is caused by the aggressive low-level clock-gating, such that energy is only required when data is written into or read out of a buffer position. The remainder of the time, clock-gating ensures that very little energy is dissipated. The same explanation also holds true for the rest of the router's computation activity.

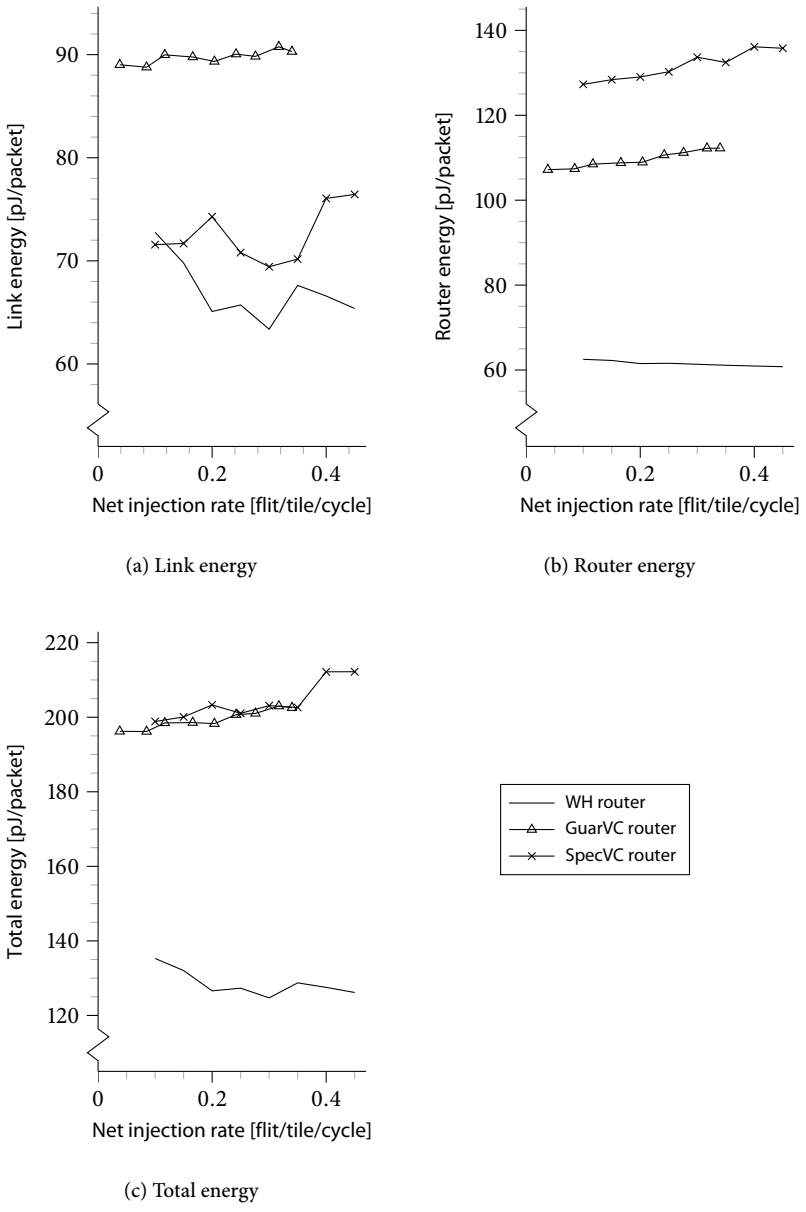
Because the energy breakdown shows an order of magnitude higher energy consumption in the data path compared to the control path, we can justify the use of more complex control to increase the networks' average performance. On the other hand the data path should be kept simple to reduce the overall energy consumption. However, a very simple (and hence low power) data path might also not be feasible from other perspectives. For instance, with the GuarVC router, the QoS specifications demand a higher order crossbar which dissipates extra energy.

6.5.4 CONGESTION

Besides the streaming packets, we compare the packets switched routers also in the scenario of uniform distributed traffic. For the GuarVC we already observed a marginal increase in the packet's energy due to the increase of congestion and header size. For the WH and GuarVC routers a 4×4 mesh network is instantiated for each design. A single router, at co-ordinates $x = 2$, $y = 2$ was considered and the energy of any packets going through it was calculated in the same way as for the GuarVC design.

Figure 6.11 depicts the packet energies for various injection rates and both router and link energy. For all the routers the energy per packet is not increasing drastically due to the increased load that causes congestion in the network. The energy per packet represents the energy required to transport this packet from one router to the next. Compared to the streaming packets traffic, the same differences between the architectures are observed for congested traffic.

The link energy of all three architectures is directly related to the packet size, as also observed with the streaming packets results. Due to the relatively large packet size of the GuarVC, its link energy is higher. The router's part of the energy of the GuarVC is lower, compared to the other VC router, probably due to the more static control path. The GuarVC has a slightly increase of the energy per packet compared to the SpecVC, which results in an almost equal total energy per packet for congested traffic. This is probably due to the fact that the round-robin arbiter of the GuarVC arbitrates on a flit bases, whereas the SpecVC tries to arbitrate on full packets over a link whenever possible. This causes more switching of the crossbar in the GuarVC design. The WH router has overall the lowest energy requirement, due to its smaller data path and lowest control overhead.



(a) Link energy

(b) Router energy

(c) Total energy

Figure 6.11 – Packet energy under congestion. *Measurement conditions:* See figure 6.8.

6.6 Conclusion

In this chapter we described the framework and measurements to analyse the power consumption of a NoC. Whereas performance used to be the main measure to compare hardware designs, nowadays the energy consumption is at least of equal importance. Some more high level frameworks and tools have already been presented by others, but we have chosen to analyse the NoC architectures using detailed simulations of placed and routed designs. It requires more effort to obtain the detailed results, but also gives a very good insight in the parts that require the largest parts of the total energy budget. Furthermore, it does not limit the designer to implement and evaluate a certain functionality, which enabled us to compare both packet and circuit switched solutions. The measurements in the chapter enable other designers to compare their routers against the energy figures of the four architectures that are presented here. The scenarios as presented form a limited but representative set of traffic patterns that can occur in a NoC.

The initial power analysis of both the GuarVC and CS routers showed a major part of the power to be consumed by the clock tree and its directly connected components. Using the PowerCompiler tooling of Synopsys made it possible to automatically include clock gating elements. Comparison of gated and non-gated designs showed an order of magnitude reduction of the standby dynamic power. Including clock gating in a NoC is crucial, because without clock gating the NoC standby power would consume its total energy budget. Furthermore, clock gating enables designers to include functionality that can increase the overall NoC performance, but which is not consuming energy during inactive periods. Without clock gating the choice would directly be a small design with the minimum number of synchronous elements.

Comparing the designs, with the fine-grain clock gating enabled, showed a greatly reduced, but still not negligible, standby power. The total consumed power of the NoC can be split into this constant standby power and a portion that is dependent on the load. The breakdown of the total power into the practically constant quantities of standby power and a dynamic energy cost per packet can now allow simpler functional simulations (to obtain the packet forwarding timestamps at each router) to give a good estimate of total power needs to be made under a wide variety of traffic patterns.

After effective clock gating, both dynamic and static parts of all routers examined have a similar contribution to the total standby power, which can be seen as overhead required by a particular NoC. For future NoCs new reduction techniques will be key to decrease its deployment costs. To reduce leakage power, advanced techniques such as power gating or the use of high-k dielectrics could clearly be applied. Techniques to reduce the dynamic component of the standby power have also been demonstrated, such as the gating of the entire clock-tree demonstrated by Mullins [96]. However, when packets are transported by the NoC some of those techniques will have no effect. For example, the buffers cannot be completely power-gated off while flits are actively stored in them. It is therefore important to strive towards architectures with inherently low standby power needs.

Table 6.6 – Reported energy per packet with random data for various five port routers

Router	Technology [nm]	Voltage [V]	Packet size [bits]	Energy per packet [pJ]
Æthereal BE	130	1.2	96	83.95
Æthereal GT	130	1.2	96	68.25
CS	90	1.2	256	80
GuarVC BE	90	1.2	256	108 ± 4
GuarVC GT	90	1.2	256	80
Lee	180	1.6	80	229
SpecVC	90	1.2	256	131 ± 4
WH	90	1.2	256	61 ± 1

For all architectures, the increase in dynamic power consumption resulted in a nearly constant energy cost per packet. The major influence on the energy consumption of packets was the router architecture itself, the size of the payload and the relative share of 'random' versus 'empty' data. The number of packets in the network and the congestion of traffic had only minor influence on the energy per packet due to the effective use of clock gating. This can be explained by the data path that is dominating the overall energy needs compared to the control path. The minor increase under congestion is mainly caused by the increase of arbitration activity, which causes extra switching between multiple inputs of the crossbar.

Table 6.6 summarizes the energy to transport a single packet from input to output of a single router for all routers described in this chapter. Although it is impossible to present a single figure to characterize the exact routers' energy requirements, this table gives a first impression and coarse estimate of the dynamic energy.

The data reported in this chapter can also be used to estimate the impact of changing some of the design parameters. For example, the impact of using a higher order topology can be seen to have a large impact on a single router's power needs, as shown by the increasing power needs of the larger crossbars in the designs presented here. However, we foresee that the data presented here could be used to better calibrate existing analytical tools, such as Orion, which can themselves be used to predict the effect of these parameter changes.

Chapter 7

Integration of a NoC in the SoC "Annabelle"

Abstract – *One part of the 4S project was the realization of a heterogeneous multi-core architecture. In this chapter we describe the SoC architecture and present the design choices that are made for the on-chip communication infrastructure. The SoC is realized in a 0.13 µm technology, has a total area of 68.5 mm², and an operational frequency between the 25 and 200 MHz for the different cores.*

In the previous chapters we evaluated the performance of four router architectures. In this chapter we will describe the integration of a router architecture in a larger system.

In the 4S project we have developed a prototype chip, called *Annabelle*, for streaming DSP applications [121, 123]. This chip is a *heterogeneous dynamically reconfigurable SoC*. Realization of such a platform enables us to verify the expected energy savings by assigning tasks to processors that could execute them most efficiently. Furthermore, integration of reconfigurable processing cores makes the SoC flexible enough to adapt the functionality of the SoC to the continuous evolution and adaptations of standards. In contrast to ASIC blocks, a reconfigurable processor can adapted by reconfiguring its instruction set. This will reduce the overall design costs, because the SoC does not require an expensive re-design of the ASIC blocks in case the standard is adjusted.

Parts of this chapter have been presented at the 10th International Workshop on System-Level Interconnect Prediction (SLIP'08), Newcastle, UK [PW3].

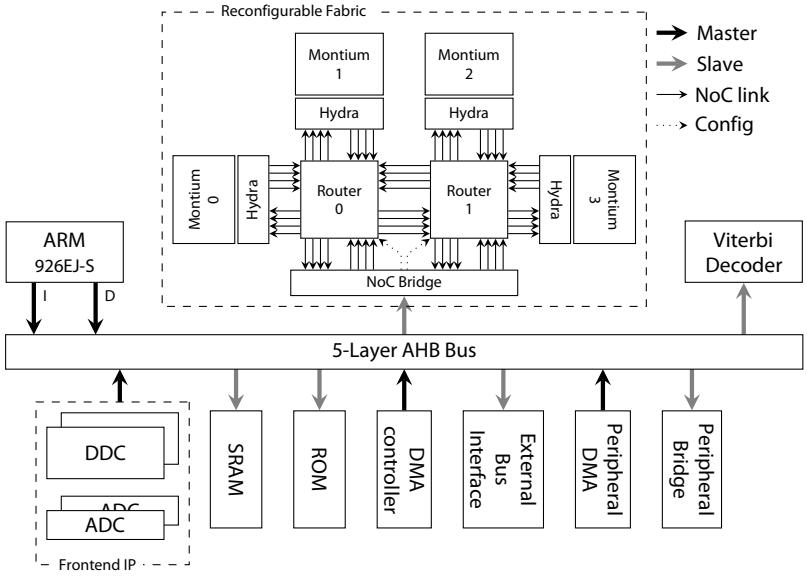


Figure 7.1 – Block diagram of the Annabelle chip

In section 7.1 we describe the architecture of the Annabelle. To interconnect the Montium reconfigurable processing cores we use a small on-chip network. Sections 7.2 and 7.3 describe the design choices for respectively the network interface and the router architecture. In section 7.4 we present the realization of the SoC architecture in a 0.13 µm technology.

7.1 Architecture

Figure 7.1 depicts the schematic block diagram of the Annabelle SoC. The SoC is a conventional ARM926 architecture with a 5-layer AMBA High-performance Bus (AHB) and standard peripherals. It is complemented by custom ASIC blocks, i.e. a Viterbi decoder, two DDCs and two ADCs, and four domain specific coarse-grain reconfigurable Montium cores [66]. The four Montium cores are interconnected by a NoC and they are grouped in a reconfigurable subsystem labelled RECONFIGURABLE FABRIC.

During the 4S project, several alternatives were considered for the prototype chip. An existing prototype SoC, called *Dimitri* [113] and developed by two of the project partners, was used as a reference. This chip is tailored to DRM and analogue radio reception and contains two ARM9 cores, two DDCs, a Viterbi decoder and peripheral I/O.

The initial ideas for the new chip consisted of a heterogeneous SoC with a central GPP and multiple efficient processing cores that can handle all the computational intensive processing parts. In case of DRM these parts are the DDC, (non-)power-of-two

FFTs and Viterbi decoding. The DAB and DMB standard, which is an interesting mix of applications for a single SoC, require the same intensive processing algorithms. For efficient processing, improvements of existing hardware modules (Viterbi and DDC) and two coarse-grain reconfigurable cores, the Montium [66] and XPP [14], were evaluated. They were compared on area, flexibility and their efficiency in processing time and energy to perform certain algorithms. The Montium core was selected as an area and energy efficient solution for algorithms like FFT and DCT [109]. Although this reconfigurable processor is able to perform a large variety of algorithms, the improved Viterbi and DDC modules are selected to handle the very computational tasks of error correction, and down converting the samples of the ADC.

In Dimitri, an AMBA-based communication bus connected all processing units. For the new prototype chip, both a bus and NoC were considered and eventually used. The bus has the advantage of being a ready and known solution which would have a limited impact on other parts of the development like the application software. The main advantage of the NoC are the possibility to offer guarantees for the communication and the ability for any processor in the system to initiate a communication stream. Both solutions can offer concurrent communication streams, because the considered bus is a multi-layer AMBA solution. The multi-layer AMBA protocol restricts this to one stream per layer and each layer is assigned to a specific master. The ARM926 processor, Viterbi and DDC module have existing data and control interfaces for the AHB and Advanced Peripheral Bus (APB) bus. The Montium core had neither an interface for an AMBA-based bus nor an interface for a specific NoC.

As depicted in figure 7.1 the final architecture uses both a AMBA bus and a small NoC to interconnect the reconfigurable cores. This combination is a balance between design cost and risk to modify all cores to a NoC based architecture and the desire to give guarantees for the communication stream between the cores. In the next two sections we describe the *Hydra* NI that connects the Montium with the small NoC and the modified circuit-switched network.

7.2 Hydra: a Network Interface Design

The Hydra is a Network Interface (NI) [130], developed to control the Montium within the Annabelle and to bridge the communication between the Montium and the NoC. The Montium tiles in the Annabelle chip operate independently so they need to be controlled separately. Since the tile processors can be operated at different clock frequencies, the NIs synchronize the data transfers between the tile processors and the NoC. The Hydra uses a light-weight message protocol that includes the functionality to configure the Montium, to manage the local data memories by means of direct memory access (DMA), handle streaming communication, and to start/wait/reset the computation of the configured algorithm.

The Montium processor core has ten local data buses that connect the five ALUs, its ten local memories and the NI [66]. Depending on the communication mode,

the Montium's sequencer or NI controls the interaction between the buses and the NI. In case of the sequencer, the Montium processor has the ability to signal that one or more of these buses have data available for the NI and or that one or more buses expect data from the NI. The sequencer is signalled when data is accepted by the NI or available from the NI. This mechanism implements a blocking write and read for streaming communication. It tightly couples the execution of the actual program with the external communication. In case the NI receives a DMA command, it stalls the Montium processor and uses the buses to directly access the processor's memories and registers. This mechanism is called block-mode communication and decouples computation and communication.

A lot of kernels, that are mapped on the Montium processor so far, work on the granularity of one or two samples, where a sample is quite often represented by one 16 bit value or a 2×16 bit complex value. In case of streaming communication, used for the applications described in chapter 3, one, two, or more values can be communicated in a single clock cycle. Although both described routers in sections 4.1 and 4.2 are able to offer guarantees, a single 16 bit network and interface to the Montium would create a significant bottleneck between the processor and network. Therefore, the ports between NI and routers are extended to handle four 16 bit values concurrently in both directions.

7.3 Modified Circuit Switched NoC

As mentioned in the previous section, a single 16 bit network creates a significant bottleneck compared to the large internal communication bandwidth offered by the Montium tile processor. Therefore, we have chosen to develop a variation on the circuit switched router for the Annabelle.

Circuit switching has been chosen as it simplifies the NI, because it does not have to embed the data in a specific network protocol and include the routing information. This is an advantage for both the sender and receiver, since there is no overhead during the communication for packaging of data (assembly or re-assembly of packets). The routing configuration is controlled via the NoC bridge. The routers' control interfaces are included in the memory map of the bridge. For the Annabelle it was expected that the mapped applications, for example DRM and DAB, would be rather static with fixed information streams. Due to the semi-static behaviour of streaming applications, the connections for the input data and output data remain open during their execution.

The modifications of the circuit switched architecture, which is presented in section 4.2.2, are the omission of the serialization and de-serialization circuitry, i.e. data converter, at the NI port, and the use of Mealy based circuitry for the acknowledgement of the flow control signal. Serialization directly limits the bandwidth of a single lane, which preferably should be prevented as mentioned with the Hydra design. Furthermore, the topology of the NoC is relatively small, which makes the penalty for a lot of wires per port smaller.

The flow control consists of a Mealy based state machine, that blocks the data

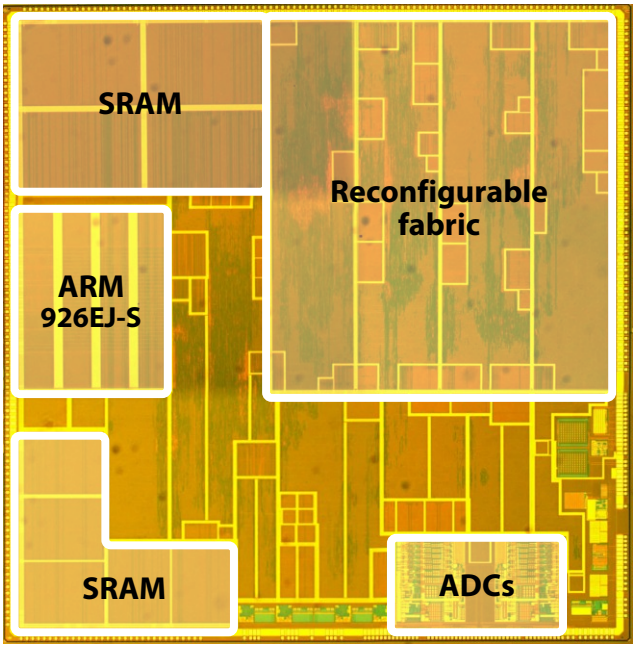


Figure 7.2 – Photo of the Annabelle IC

streams (including the items in the intermediate routers) if the receiver is not able to accept the data. The state machine creates a combinational path between the sending and receiving node, but due to a relatively low frequency (100 MHz) and small network, this is not a critical path in the design. The advantage of a Mealy machine compared to a Moore machine is the minimum buffer depth requirements of one instead of two per port. The credit based mechanism is omitted, because it required a more intelligent NI and AHB bridge.

7.4 Realization of the SoC

The Annabelle has been synthesized and produced in ATMEL's proprietary 0.13 µm process technology. The ARM subsystem is constrained to a clock frequency of 200 MHz. The bus, NoC and rest of the system are constrained to 100 MHz. The longest critical path of the Montium limits the processor's frequency to 25 MHz. Because not all algorithms use this worst-case path, each Montium can select its frequency from 100 MHz to 6.25 MHz.

Figure 7.2 depicts the layout view of the actual chip. The major blocks are indicated by the white rectangles. The chip is packaged in a 400-pin BGA package. Figure 7.3 depicts both sides of the package together with a two euro cent for an impression of the package's size.

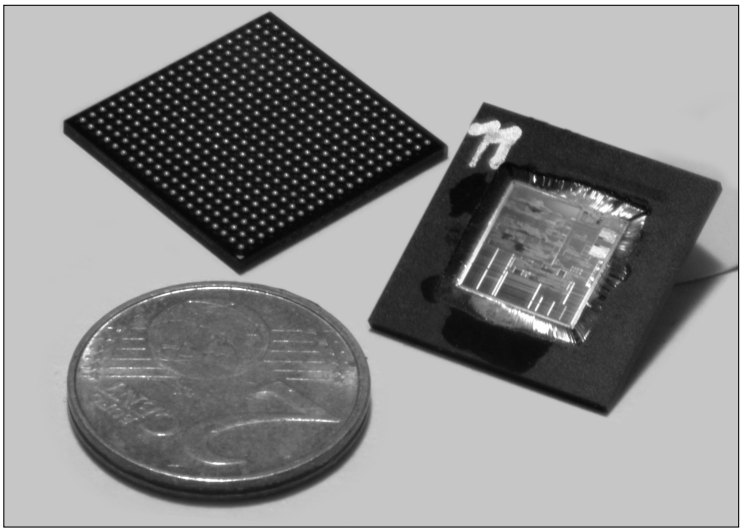


Figure 7.3 – Photo of the Annabelle SoC embedded in a 400-pins BGA package

Table 7.1 – The area of synthesized building blocks in reconfigurable fabric

Component	Amount	Area [mm^2]
AHB-NoC bridge	1	0.18
Circuit-switched router	2	0.09
Hydra NI	4	0.10
Montium TP	4	3.63[a]
Total		15.31

[a] Of which 2.76 mm^2 is logic and 0.87 mm^2 is SRAM

The total area of the chip produced is 68.5 mm^2 of which 21.5 mm^2 is occupied by the reconfigurable fabric. The total available memory within the chip is ≈600 kB of which 416 kB is shared SRAM connected to the AMBA bus and 80 kB is distributed RAM for the data memories in the four Montiums. The area of the individual components in the reconfigurable fabric after layout can not be determined, but the synthesized area is presented in table 7.1.

7.5 Conclusion

In this chapter we described the realization of a heterogeneous multi-core architecture. This SoC consists of an ARM926 processor core complemented by five custom ASIC blocks, four domain specific coarse-grain reconfigurable Montium cores, and standard peripherals.

The individual processing cores are interconnected by a hybrid communication

architecture. The ARM, ASIC blocks and peripherals are connected, as either master or slave, to a 5-layer AHB. In comparison with the CS router architecture, as described in chapter 4, the four reconfigurable cores are interconnected by two slightly simplified circuit switched routers. The routers receive data from the AHB via a slave bridge.

In this SoC the ARM fully controls the configuration of the individual processing cores. The circuit switched network was chosen as the best alternative, because of the small network topology, the necessity for concurrent communication between the reconfigurable cores, and it enables direct control of routers via the AHB bridge by the ARM processor. For this small network and relatively small number of processing cores the serialization is omitted to increase the bandwidth and to reduce the overhead in area.

Chapter 8

Efficient FPGA-Based System Simulation

Abstract – *Experience with cycle-true simulation of hardware designs showed prohibitive simulation times from multiple hours to days. In this chapter we provide a framework to speed-up the simulation of such large homogeneous and heterogeneous systems using an FPGA-based simulator. The simulator is instantiated in an FPGA by manually transforming the hardware design that is simulated. We use the repetitive nature of NoC and SoC architectures to be able to simulate considerably larger designs than would normally, before the transformation, fit in the FPGA. As a case-study we used the GuarVC network to demonstrate the performance gain in simulation time we observed compared with a SystemC based approach. The chapter concludes with a discussion to automate the creation of the simulator in the FPGA.*

The development of homogeneous and heterogeneous platforms introduces problems related to hardware/software co-design. A Multi-Processor System-on-Chip (MPSoC) architect studies all kinds of trade-offs. Examples of trade-offs are operation bit-widths, memory sizes, and performance parameters and bottlenecks, e.g. latency and throughput. Common practice is to perform extensive simulations of the MPSoC architecture before the system can be realized in silicon. In general the approach of simulating large MPSoC designs is to either use (non cycle accurate)

Parts of this chapter have been presented at the 21st IEEE International Parallel and Distributed Processing Symposium (IPDPS'07) - 14th Reconfigurable Architecture Workshop (RAW 2007), Long Beach, CA, USA [PW6], and the 1st ACM/IEEE International Symposium on Networks-on-Chip, Princeton, NJ, USA [PW5].

high level modelling or accept long simulation times with cycle accurate simulations. For systems consisting of several tens or hundreds of tiles, cycle-true simulation leads to prohibitive simulation times from multiple hours to days. For example, in the development process of the Annabelle SoC the simulation of the complete design, which was described in VHDL, took a considerable amount of time.

Despite these excessive simulation times, cycle and bit accurate tests are required. These tests enable the designer to verify the design before manufacturing. Furthermore, it is possible to make accurate performance trade-offs and determine the consequences in area and power consumption of the actual design. Using the same concrete implementation (i.e. the same HDL source code) for simulation as well as synthesis we minimize the risk of errors in the design flow.

In the context of this thesis we studied the consequences of design choices on the performance of the NoC in a system with hundreds of tiles. For determining the performance we cannot just analyse a single router in isolation and determine a local optimal schedule, because a certain router might for instance cause buffering problems in the neighbouring routers. As described in chapter 5 we analysed and compared the performance of the GuarVC with other packet switched alternatives. This analysis is performed with the description in VHDL code, such that the same description is also used to analyse the architecture on its power consumption in chapter 6. Our experience with the performance of FPGAs for Software Defined Radio (SDR) and long software based simulations for hardware designs motivated us to explore the possibilities of an FPGA-based simulator.

For a traditional *Hardware-In-the-Loop* (HIL) simulation of a design, described in any HDL, it is analysed, synthesized to a specific FPGA technology and finally placed and routed onto the targeted FPGA. The FPGA-based simulator, described in this chapter, employs a sequential evaluation of the parallel architecture. We therefore refer to the simulator as *Sequential Hardware-In-the-Loop Simulator* (SHILS).

The rest of the chapter is organized as follows. In section section 8.1 we present several methods to analyse and simulate multi-core architectures. The FPGA-based solutions can either simulate relative small systems or require a large number of FPGAs. Therefore, we propose in section 8.2 a new framework that reduces the resource constraint of an FPGA. The implementation of this sequential simulation framework is presented in section 8.3. The GuarVC router is used to evaluate the performance of the simulator in section 8.4 and we discuss the flexibility of the realized simulator in section 8.5. The creation of the simulator is done manually, but in appendix D we discuss our ideas for an automated design flow and present our first experiences with automated transformation. In section 8.6 we draw some conclusions.

8.1 Related Work

There are several methods to analyse large heterogeneous and homogeneous systems. High level formal analysis methods can be applied [135], where an application of the system is characterized by high level parameters (e.g. latency of a task).

Data dependencies and interactions of processes can only be analysed if their characteristics can be described with the high level model. However, this is only applicable for a restricted number of cases. For example, at design time the worst-case latency and execution time of a task have to be known.

Another method is system level simulation such as SystemC [102] at different levels of abstraction. It can be used to describe systems from the functional level to RTL level. The level of abstraction determines the speed of simulations. An example of SystemC simulation for NoC is the OCCN project, introduced by Coppola et al. [30], which defined a universal API for specification, modelling, simulation, and design space exploration of NoCs. Other examples are the MPARM simulator, which is a multi-processor cycle-accurate architectural simulator [17], or the framework presented by Kogel et al. [83]. In the design flows of Æthereal [56] and ×pipesCompiler [75], SystemC simulation is used for performance validation. The level of detail in the SystemC simulation tremendously influences the speed of simulation. TLM, abstract data types and timed simulations showed a speed-up of almost three orders of magnitude compared to RTL modelling [83]. However, optimizations to increase the simulation performance sacrifice the level of detail in the simulated results.

The general SystemC approach, which supports any design, can also be replaced by simulators dedicated for specific architectures or domains. For example, the OPNET modeler [101] is a domain specific simulator for network systems that has been used by Bolotin et al. to simulate the QNoC architecture [22].

Simulation performance can also be increased by using more processors in parallel or specialized hardware. For very large multiprocessor systems, an FPGA-based emulation platform makes accurate and fast system simulation possible, as proposed in the RAMP project [7]. This approach requires multiple FPGA platforms as for example provided by the Zebu-XL system emulator [48] or the BEE2 [25] multi-FPGA board. Njoroge et al. [99] demonstrated the mapping of ATLAS, a chip-multiprocessor that prototypes the Transactional Coherence and Consistency (TCC) architecture, on the BEE2 platform. The ATLAS design requires 25 to 30% of the Virtex-II Pro LUT resources and runs 100 times faster than the TCC simulator on a 2 GHz PowerPC G5 system. The performance improvement between a SystemC implementation and hardware emulation framework in an FPGA is also demonstrated by Del Valle et al. [41]. With a single Virtex-II Pro FPGA, at 100 MHz, execution speed-ups of two to three orders of magnitude were measured in comparison with the software based implementation on a Pentium 4, at 3 GHz.

The ProtoFlex hybrid simulator is another FPGA-based simulation and emulation platform proposed by Chung et al. [29]. This simulator has two enabling techniques: 1) hybrid transplant simulation and 2) multiple-context emulation engines. The transplant option makes it possible to accelerate simple and/or frequently used operations in the FPGA (e.g. ALU operations). Complex and infrequent behaviours (e.g. disk I/O) are simulated on the simulator host (i.e. PC). The latter technique is similar to the method described in this chapter. A small number of FPGAs host multiple-context emulation engines. Multiple processors are mapped onto the emulation engines. Simple time interleaving of the processors on the

engines decouples the simulated system size from the required hardware resources of the FPGA host system.

Several FPGA-based implementations to validate NoCs are described in literature. Marescaux et al. [89] describe their implementation of interconnection networks on an FPGA. Genko et al. [54] propose a NoC emulation framework implemented on a Virtex-II FPGA. The emulation platform combines traffic generators, network interfaces, routers and traffic receptors. The platform is controlled by the FPGA's PowerPC and can work at 50 MHz. This is respectively 2.5 and 15 thousand times faster in comparison with the SystemC simulator MPARM and a Verilog based simulation [54]. Genko's approach is bounded by the maximum available slices. For example, a six router network required 79% of a Virtex-II Pro VP20. Kumar et al. [84] describe a SoC design on a Virtex-II 6000 FPGA consisting of set of processing cores and peripherals connected via network interfaces to an Æthereal NoC. The simulated systems run at frequency of 12.5 to 25 MHz and the relative small designs are limited by the FPGA resources.

8.2 Simulation Framework

This section describes the simulation approach that is used to simulate large NoCs consisting of tens to hundreds of routers on limited FPGA resources. The approach is developed to simulate a large number of clock cycles more easily and study various design parameters and their effects without sacrificing accuracy or detail. The NoC is used as a test case. The technique can be used for other systems consisting of a homogeneous or heterogeneous set of entities as well.

Simulation of a system at cycle-accurate and bit-level detail requires a continuous update of its exact internal state based on its function, inputs and current state. The network of combinational elements describes the functionality of this system. The state of a synchronous system is stored in registers and updated each clock cycle. The speed of a cycle-accurate simulator of a synchronous system is determined by the time it takes to update all registers of the system, i.e. how much simulation time is required for a cycle?

The attempt to simulate a NoC in an FPGA was inspired by the fact that an FPGA has parallel access to a large amount of distributed internal storage, which enables updating a large number of registers in a single FPGA clock cycle. When the FPGA has enough resources to update all registers in a single cycle, the whole simulated design can be instantiated in the FPGA. A mesh network of GuarVC routers (see section 4.1) was synthesized for a Virtex-II 8000 FPGA. A NoC of fewer than ten routers could be synthesized with a data path of 16 bit. A reduced data path of six bit without network interfaces, traffic generators and simulation controllers increased the network size to a maximum of 24 routers. The two major bottlenecks were the number of CLBs and available number of tri-states in the FPGA.

Sequential update of the design's state in multiple cycles overcomes this resource limitations, because combinational resources can be re-used if multiple parts of the design have an identical functionality (i.e. routers in a homogeneous NoC).

The SHILS makes another trade-off between hardware resource requirements and simulation speed.

One of the problems of this approach is cyclic dependencies. A NoC can have multiple cyclic dependencies that are part of the design of the router or created by the interconnection of routers to create the desired topology of the network. The cyclic dependencies might cause changes of the inputs of a router after its evaluation in the sequential schedule. The change of input requires re-evaluations of the routers that corresponds to that input. We will explain the sequential simulation methodology using a 1-dimensional cyclic dependent example. The method is also applicable for two and higher dimensional systems.

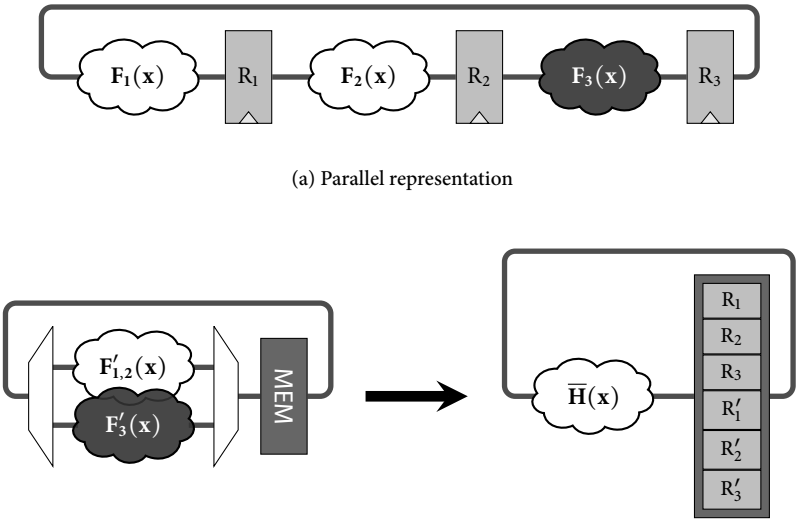
In the next two sections this sequential simulation of a design is explained. We start with a cyclic system that can have an arbitrary evaluation order and no re-evaluation. This method is extended in section 8.2.2 to a partitioned system, where re-evaluation is necessary due to the cyclic dependencies and its partitioning. The method is based on the two level timing model that was introduced in the Consensus Language (CONLAN) [104] and included in the simulation mechanism for VHDL. In the explanation of the methodology we use *system cycles* (i.e. simulated time) and *delta cycles* (i.e. computation steps). A delta cycle is defined as a clock cycle of the SHILS that evaluates one function, but does not advance the simulated time. A system cycle is a clock cycle of the simulated parallel system that advances the simulated time. A system cycle consists of multiple delta cycles.

8.2.1 SEQUENTIAL SIMULATION OF DESIGNS WITH REGISTERED BOUNDARIES

In this section we assume a system that has multiple circuits that are separated by registers. The individual circuits in a system have functions $F_i(x)$ that may be equal or different. In the homogeneous NoC of our test case, all routers have the same functionality. The inputs of all $F_i(x)$ come from registers and the results go to registers.

Figure 8.1(a) depicts a parallel system that consists of three combinational circuits with functionality $F_1(x)$, $F_2(x)$ and $F_3(x)$. To simulate a system cycle of this system, we evaluate the three circuits in an arbitrary sequential order. This increases the required time to simulate the system by a factor three, but reduces required hardware resources. If the identical functions $F_i(x)$, $F_j(x)$ can share the same implementation, this implementation is denoted as $F'_{i,j}(x)$.

The logic of figure 8.1(a) is changed to figure 8.1(b), where all registers are mapped into a single memory. Mapping registers to memory is possible, because not all registers are updated in parallel. The registers of a circuit, which are updated in parallel, are concatenated into a single memory word and stored at a single memory address. This word can have a considerable width due to the large amount of distributed RAM that is available on an FPGA. This makes them accessible in a single read/write cycle of the memory. The functions $F_1(x)$, $F_2(x)$ use implementation $F'_{1,2}(x)$ and F_3 uses $F'_3(x)$. The set of implementations is denoted as $\overline{H}(x)$. The size of this set is limited by the available combinational hardware resources. The number of bits of a memory position limits the number of implementations of the



(a) Parallel representation

(b) Sequential representation

Figure 8.1 – System with registered boundaries

set that can be executed in parallel. The total amount of memory bits limits the state size of the parallel design.

In the memory, both the old and new version of the register values are stored (as depicted in the right part of the figure by $R_{1..3}$ and $R'_{1..3}$). The order in which the implementations are evaluated to calculate new register values can be arbitrary and statically scheduled. For all parts of the system a previously calculated register value is used at the inputs of the combinational circuit to calculate the new register value.

After all functions are evaluated the new state becomes valid and can be copied to the current state of the registers and then a new system cycle can be started. In the system, as will be described in section 8.3, this copy action is not needed and performed by switching the offset pointer of the current state and new state. In the even system cycles the registers $R_{1..3}$ are the current state and $R'_{1..3}$ are the next state. In the odd system cycles, $R'_{1..3}$ are the current state and $R_{1..3}$ are the next state. Figure 8.2 depicts one of the many possible execution examples of simulating three system cycles, where the implementations $F'_{1,2}(x)$ and $F'_3(x)$ are not executed in parallel.

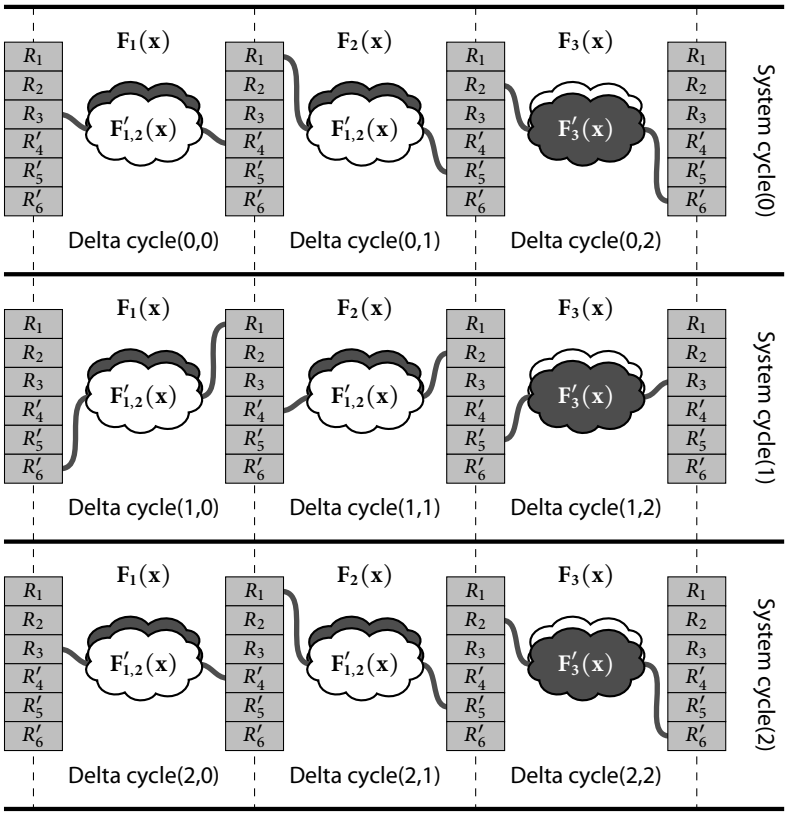


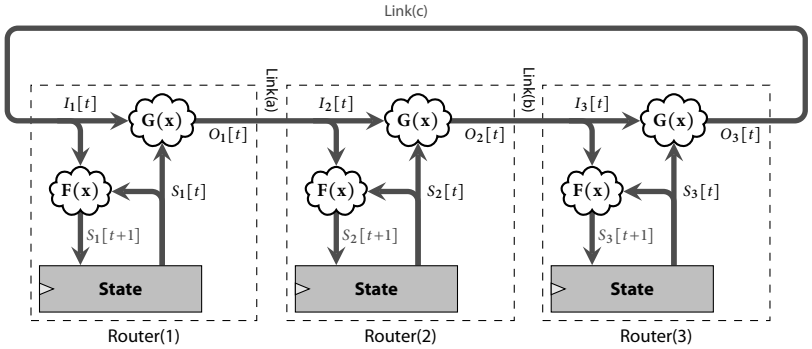
Figure 8.2 – Possible static schedule

8.2.2 Sequential simulation of designs with combinational boundaries

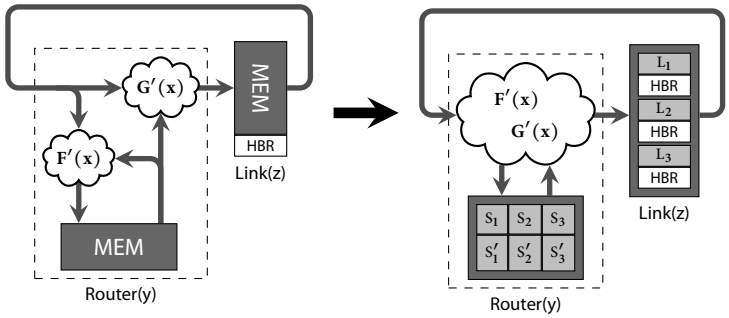
The approach described above is not completely suitable for an arbitrary design, because every output of a block is registered, i.e. independent of the current input value as with a Moore based state machine. In this section we describe the extension to blocks, where outputs can immediately change due to a change of the input signals, similar to a Mealy based state machine.

This approach partitions the design based on the granularity of the available functionalities $F_i(x)$ (i.e. hierarchical blocks) in the design and not the locations of the registers. In case of a homogeneous NoC the basic element and reasonably large functionality is a router and all routers are identical. In the simulator the routers can all share the same implementation $F'_{router}(x)$. The simulation time is proportional to the number of simulated routers.

However, both the router's inputs and outputs are connected to neighbouring routers via links (i.e. wires without registers). The previously described method requires that all connections between functionalities are registered. However, this



(a) Parallel representation



(b) Sequential representation

Figure 8.3 – System with combinational boundaries

is not the case in the NoC design and many other designs. It is undesirable to change a given implementation for testing purposes, because rewriting the code might introduce faults, unwanted pipeline behaviour, and because it is difficult. It requires repositioning of functionality. In a NoC, for example, the functionality might shift to the neighbouring routers. However, in case of the router described in section 4.1, it was not even possible without changing the functional behaviour of the design. A solution is to re-evaluate the router's state and output if one of its inputs has changed after its evaluation in the current system cycle. Re-evaluation is possible, because the router's old state is available during the whole system cycle.

Figure 8.3(a) depicts a system which represents our router architecture. The router cannot be described by a simple function $F_i(x)$, because the router's *outputs* ($O_i[t]$) depend, besides on its *inputs* ($I_i[t]$), on its current *internal state* ($S_i[t]$). This can easily be solved by storing all internal registers that store the state in a memory

as depicted in figure 8.3(b). The functionality of the router is then described by a set of functions, $F(x)$ and $G(x)$, that create the current output signals:

$$O_i[t] = G\left(I_i\left[t\right], S_i\left[t\right]\right) \tag{8.1}$$

and the router's next internal state:

$$S_i[t+1] = F\left(I_i\left[t\right], S_i\left[t\right]\right) \tag{8.2}$$

where t is the simulated time. This set of functions, $F(x)$ and $G(x)$ of a single router, will be evaluated in parallel in the sequential simulation.

A combinational loop seems to be present in the homogeneous parallel design as depicted in figure 8.3(a). However, this is due to the simplified graphical representation. For this approach, we assume that the original parallel system can be synthesized and does not contain a combinational loop that can cause oscillations. The routers evaluated will never create a combinational loop. In general, a combinational loop will be detected during the synthesis of the parallel design.

For the actual router designs only a number of the output signals depend on the values of input signals $I_i[t]$.

Because we have a now Mealy based design, we cannot evaluate these routers in-order as in figure 8.2. Any other given sequence of routers will also give problems, because there is always a link that is read before it is updated by the preceding router. Changing the partitioning (e.g. merge part of $G_i(x)$ with either $F_i(x)$, or move it to the preceding or next router) to create a Moore based design instead of a Mealy based design was examined. However, no fully registered cross-section, which makes the outputs only depended on the internal state, was possible and modifying the design, as previously stated, is generally undesirable. Problems occur if and only if the update of the preceding router changes the values of the link after the link is read. Therefore, data between routers is exchanged via separate link memories that have additional status signals as depicted in figure 8.3(b).

All internal registers use the same mechanism as described in the previous section. The links have a separate memory, where every link has only a single memory position and not two as for the registers. A single position is sufficient, because only the most up to date value is of interest. All bits of a link, which connects two routers, and have the same direction (e.g. from router A to router B) can be grouped into a single memory word on a single memory position. The signals of a router that either do not have the same source and/or destination router have to be accessible via separate memories. All input and output signals for a single router have to be accessible in parallel for write or read access.

Per memory position one additional status bit is stored. This bit indicates whether the last written value *has-been-read* from this link. Per router, we group all has-been-read (HBR) bits that correspond with the links that are connected to its input ports. If not all of these bits are valid, the router is considered to be *non-stable* and has to be evaluated. A scheduler, for example round-robin, can decide which non-stable router to evaluate. If all routers are stable, the network is considered to be completely evaluated and ready for the next system cycle.

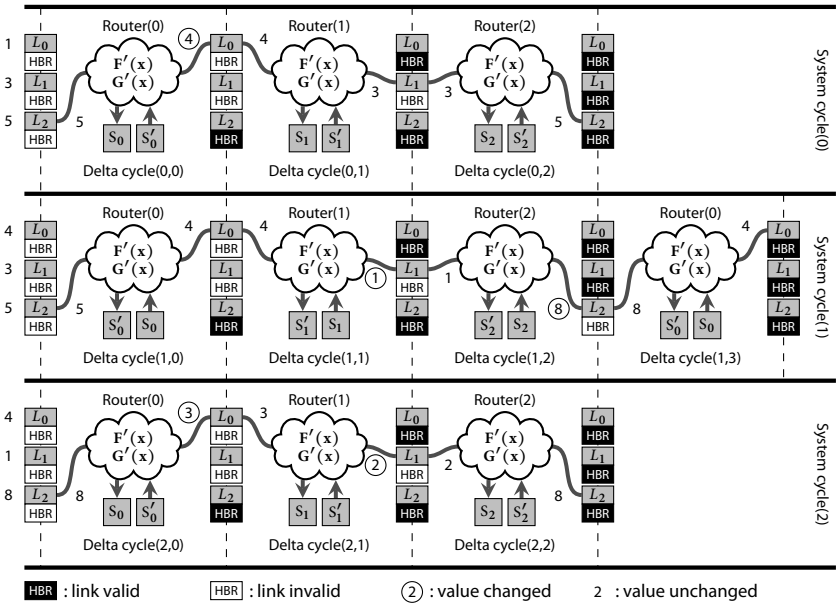


Figure 8.4 – Possible dynamic schedule

An example of three system cycles is given in figure 8.4. Every system cycle is started by resetting all HBR bits to invalid, which makes all routers non-stable. Because all HBR bits are reset to zero at the start of a system cycle, it is guaranteed that all routers are evaluated at least once. This is necessary, as a router might change its outputs independent of its inputs. All links that are connected to the router's input port will make their HBR bit valid after it is evaluated. When the router writes a value to a link, which is not equal to the current value in the link memory, it will invalidate the link's HBR bit. The router that has this link as an input will become non-stable and has to be (re-)evaluated. In figure 8.4, all values that are different from the current value are circled. This happens in delta cycle (1,1), (1,2), (2,0) and (2,1). In case of delta cycle (1,2) link L_2 is updated, but had already been read in delta cycle (1,0). In this specific case, the HBR-bit changes from valid to invalid and Router(0) has to be re-evaluated. In all other cases, the updates of the links do not result in extra evaluation cycles, as the HBR-bit was still invalid and routers had to be evaluated anyway.

8.3 Implementation

This section describes the implementation of the described simulation approach. The implementation of the simulator requires a hardware platform, a design to be simulated and control software. We use the NoC of section 4.1 to describe the implementation and to evaluate the simulator's performance.

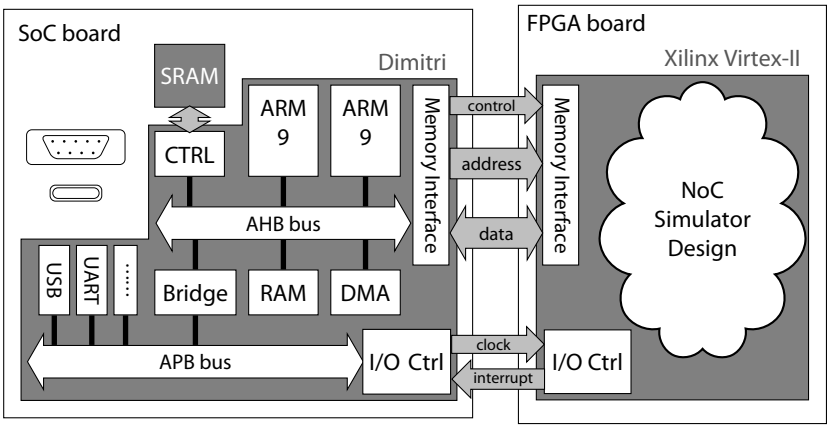


Figure 8.5 – Schematic view of the hardware

First, a hardware platform with a (large) FPGA is required that is able to simulate the network. Section 8.3.1 describes the platform we had at our disposal. Second, the design to be simulated needs to be modified to fit the described method of section 8.2 and needs to be synthesized to the FPGA. Section 8.3.2 describes the architecture and functionality of the FPGA design. Third, the NoC design needs to generate test patterns, i.e. stimuli, and analysis software to evaluate the router's performance. This is described in section 8.3.3.

8.3.1 Platform

Figure 8.5 depicts the general block diagram of the available platform with the most important components. It consists of a SoC board and an FPGA board. The SoC board contains two ARM9 general purpose processors. The two processors run at a frequency of 86 MHz and 192 MHz respectively. With this dual-processor ARM chip we can partition the control software among two processors. The SoC is connected to 1 MB on-board SRAM memory and lots of peripherals and connectors. One of the connectors connects the SoC board with the FPGA board. This connector contains a memory interface with a 32 bit wide data-bus and a 17 bit wide address-bus. Via this memory interface the logic of the FPGA can be controlled and blocks of data are exchanged between the memory of the SoC and the memory instantiated in the FPGA. The FPGA board itself does not have separate off-FPGA memory. The FPGA board contains a Virtex-II 8000 FPGA.

Any other platform that has a large FPGA, general purpose processor, and on-board SRAM memory is suitable for the simulations. The on-board SRAM is required by the general purpose processor that controls the simulation. This control consists of both generation of stimuli vectors as well as analysis of the results. The general purpose processor can either be implemented on the FPGA as soft- or hardcore, or, as in our case, a separate processor on the platform.

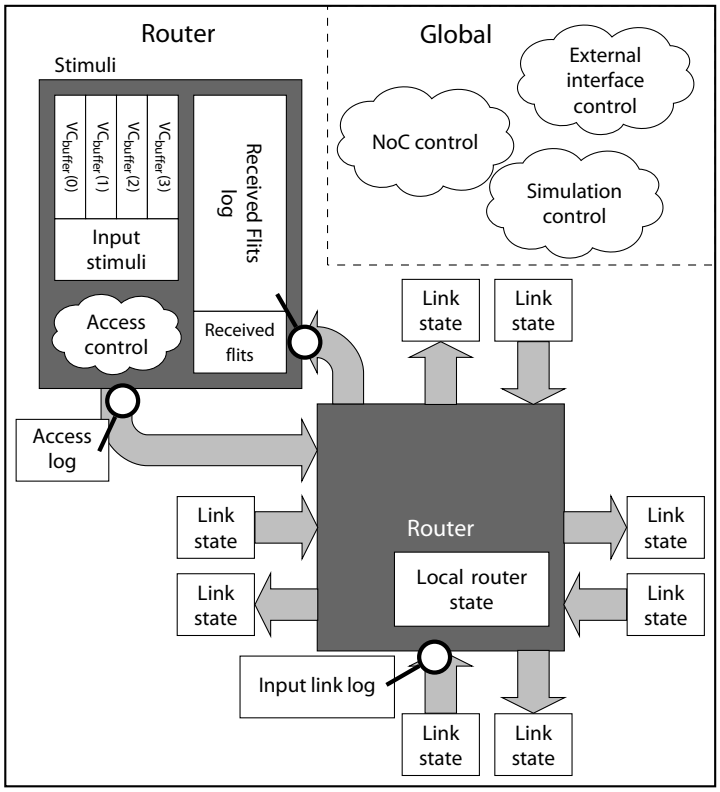


Figure 8.6 – Schematic view of the FPGA design

8.3.2 FPGA implementation

Figure 8.6 depicts the major blocks of the FPGA design. The design can be partitioned into two major parts.

1 The router part, that describes the logic of a *single* router and its stimuli interface.

2 The global part, which controls the FPGA and the NoC that is simulated.

Other parallel systems can be built similarly. If the system would have been heterogeneous, all the unique components needed to be instantiated at least once as depicted in figure 8.1(b).

The simulated router is based on its original design sources, written in VHDL, which are normally used to synthesize for a specific technology. The original sources are modified to separate the combinational logic from the registers. Currently, this is done manually, but in section 8.5.2 we discuss our first experiences with automated transformation.

The input and output signals of all registers are concatenated into two memory

Table 8.1 – Required registers per router

State	Registers
Input queues	1440 bits
Router control and arbitration	292 bits
Links	200 bits
Stimuli interfaces	180 bits
Total	2112 bits

words: *old* and *new*. The old word is the current state of the router and read from the memory at the start of a delta cycle. The new word is the result of the evaluation and has to be written into the memory after a delta cycle. The address of the memory depends on the router that is evaluated. This address is generated by the simulation control in the FPGA. Table 8.1 summarizes the width of the memory word, which is determined by the number of registers in the design. The number of routers in the network determines the depth of the memory.

In the current implementation reading the values (2112 bit in the example) from memory takes one cycle. Evaluation of the combinational logic and writing the result in memory takes another cycle. In total a delta cycle equals two FPGA cycles. For a design with N routers, where the combinational circuit of one router is mapped onto an FPGA, one system cycle takes $2N$ FPGA cycles. This does not include the extra delta cycles due to re-evaluation of non-stable routers.

The stimuli for the design are generated by software in the ARM9. Generation of stimuli in software makes it easier to define new tests and analyse the results. The disadvantage is the large amount of data that has to be copied from the ARM9 to the FPGA and vice versa. This can be solved in two ways:

1. Implement traffic generators in the FPGA, as used by Genko et al. [54]. This restricts the possible tests to the implemented and synthesized generators. Adding new and custom traffic patterns requires the re-synthesis of the simulator.
2. Use an embedded processor in the FPGA. This can either be a soft core like a MicroBlaze or hardcore as for example the PowerPC available in the Virtex-II Pro and Virtex-4 FX.

The stimuli are buffered per VC in cyclic buffers in the FPGA. The output values of the network are stored per router, and not per VC, because a tile will receive at most one flit per system cycle due to the TDM of VCs on a link.

The data in the stimuli buffers has a time stamp and can be read or written by the ARM9. The timestamps make it possible to store only valid data, which requires less storage space and less time to copy data. The cyclic buffers make it possible to run the simulation independently from the copying of data. Of course, buffer under- and overrun has to be prevented, because it will influence the traffic in the NoC and even worse introduces errors. Therefore, the ARM will request the FPGA to simulate a specific number of system cycles. The FPGA will signal the ARM when

it is ready with the requested number of system cycles. During the simulation in the FPGA, the ARM performs other tasks as described in the next section. Two extra cyclic buffers facilitate to log:

1 the traffic (i.e. bit-values in every clock cycle) of a specific link between two routers.

2 the access delay a flit notices before it enters the network, because of the blocking due to a full VC buffer in the first router.

These two buffers cannot influence the traffic in the NoC. Possible buffer overflow of these buffers can be prevented by the ARM by reading the content at regular intervals.

8.3.3 SOFTWARE

The simulation is completely controlled in software by one or two ARM processors. The choice between one or two processors depends on the estimated simulation length and desired simulation speed. The software is partitioned in processes that communicate via cyclic buffers. All the processes can run in parallel and do not have dependencies except for the communication via the buffers.

Figure 8.7 depicts the organization of these processes and what part of the hardware is involved. The top processes require only the ARM processor and the NoC simulation itself only requires the FPGA. The two processes that interchange data between the boards require both ARM and FPGA. Each process that requires an ARM can be mapped on either of the two ARM processors.

Because each process uses its own cyclic buffers, it only needs to be fired when data and free memory are available. The processes that only require the FPGA or ARM run in parallel, which tremendously reduces the simulation time. We also have two ARMs available on the SoC-board that make it possible to run the generate and analysis process in parallel. Running the two processes in parallel requires more effort and time from the simulator user to start a simulation. Therefore, the parallelism with two ARMs is only beneficial if long simulations are performed that outweigh this extra effort. The simplest partitioning is to run the generation and loading of data on one ARM, and to retrieve and analyse the results on the other ARM. This requires the least amount of interprocessor communication and gives a good speed-up.

The simulation is performed in steps. In each simulation we use deterministic source routing. The routes are predetermined for each source-destination pair before the actual simulation. After all routes are determined, a loop is started that has five phases.

1 For each node a stimuli table is generated. Any data pattern can be generated as the generation is done in software. The generation process uses a random number generator on the FPGA. Reading a 32 bit random number from the FPGA is noticeably faster, 50% increase in simulation speed, compared to the standard *rand()* function in C. The generated stimuli table contains stimuli for

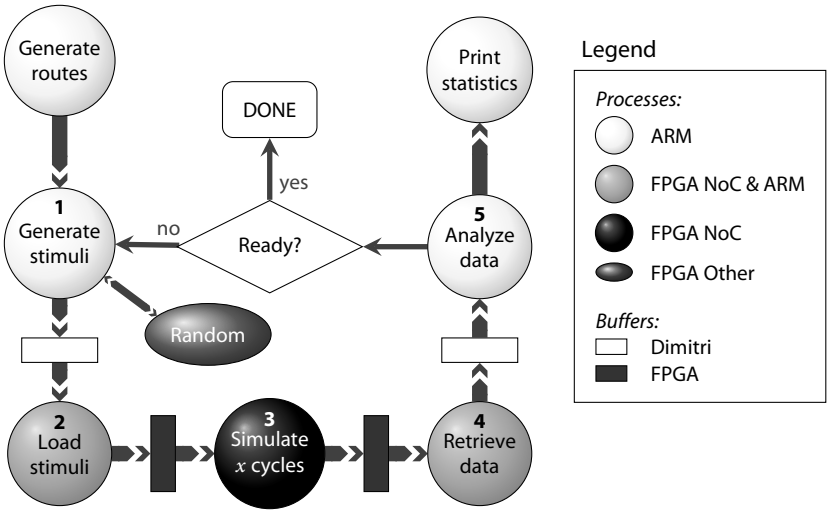


Figure 8.7 – Software processes of the simulation

at least x system cycles, where x is the number cycles the FPGA will simulate without an interrupt by the software.

2 The generated stimuli have to be written into the input buffers of the FPGA. All input buffers are maximally filled unless no data is available.

3 After filling the buffers we start the simulation in the FPGA and evaluate x system cycles. This sequence of x simulated system cycles is called a *simulation period*. To prevent buffer underrun, the simulation period is fixed to the size of the VC stimuli buffers in the FPGA. The simulation in the FPGA needs to be started by software, but runs autonomously.

4 After a single simulation period, we have to empty the output buffers. We retrieve the data from the output buffers that we think are interesting for the analysis. For the buffers that are not interesting we can update the read-pointer, which empties the buffer.

5 After the data is retrieved from the FPGA it is analysed and the desired statistics are stored.

When the simulation is not finished we go to step one and generate extra traffic in the stimuli table. In this way we can simulate an arbitrary number of simulation periods which is neither limited by the software nor hardware. However, due to back-pressure in the network, not all generated data might have been written into the FPGA. To prevent the loss of this data and the potentially resulting undefined state of the stimuli, all unconsumed data will eventually be written into the FPGA. If the network is saturated, due to high traffic loads, both the buffers in the FPGA and in software will saturate too. When this occurs, it is reported to the user and simulation is stopped, because buffer overflow will result in wrong simulation results.

Table 8.2 – Xilinx Virtex-II 8000 FPGA resource usage (256 routers)

Block	CLBs	Block RAMs [2.25 kB]
Router	1762	61
Stimuli interface	540	62
Network	2103	16
Random number generator	2021	0
Global control	627	0
Total	7053 (15%)	139 (82%)

8.4 Results

In this section we describe the performance of the simulator. The simulator is realized using a Xilinx Virtex-II 8000 FPGA and can simulate any size of network from two to 256 routers with four flit deep queues. Table 8.2 shows the resource usage of the simulator in the FPGA for a maximum network of 256 routers. From these results, it is clear that the limiting factor of the design is the number of block RAMs that are used.

The router design is synthesized for a frequency of 6.6 MHz, which gives a delta cycle frequency of 3.3 MHz. This limits the maximum simulation frequency of the simulator to $3.3 \cdot 10^6/36 = 91.6$ kHz for a 6×6 network. No effort was made to increase this frequency, because it was sufficient for the current tests. The interface frequency is equal to the ARM frequency of 86 MHz, which makes it possible to copy data at a higher frequency.

The number of system cycles per second, i.e. simulation frequency, that can be simulated depends on the simulation settings. As a reference we use the simulation frequency of our SystemC simulator that was used to derive figure 5.4. These and other SystemC simulations on a 6×6 mesh network had an average frequency of 0.215 system cycles/msec independent of the applied network load. Figure 8.8 depicts the simulation frequency (in cycles/sec) under different network loads for the series of tests as described in section 5.1 and section 5.2.1. In all tests, we use the slowest of the two ARM processors at 86 MHz. From this figure we can conclude that the FPGA simulator is a factor of 80–300 faster compared to our SystemC simulation. Under low traffic loads, the frequency is close to the theoretical maximum that is limited by the FPGA. Under higher loads the frequency decreases due to extra time required by the software processes of figure 8.7. Furthermore, in figure 8.8 it is visible that the frequency is approximately inversely proportional to the size of the network.

For different tests on the 6×6 topology with identical load, the minor differences are caused by the relation between injection rate and packet rate. For example, the analysis and generation time for the BE tests is almost independent of the packet size. Larger packets (11 flits) will cause a higher injection rate, which results in shorter simulation times at identical injection rates. In other words part of the extra overhead depends on the number of packets and part of the extra overhead depends

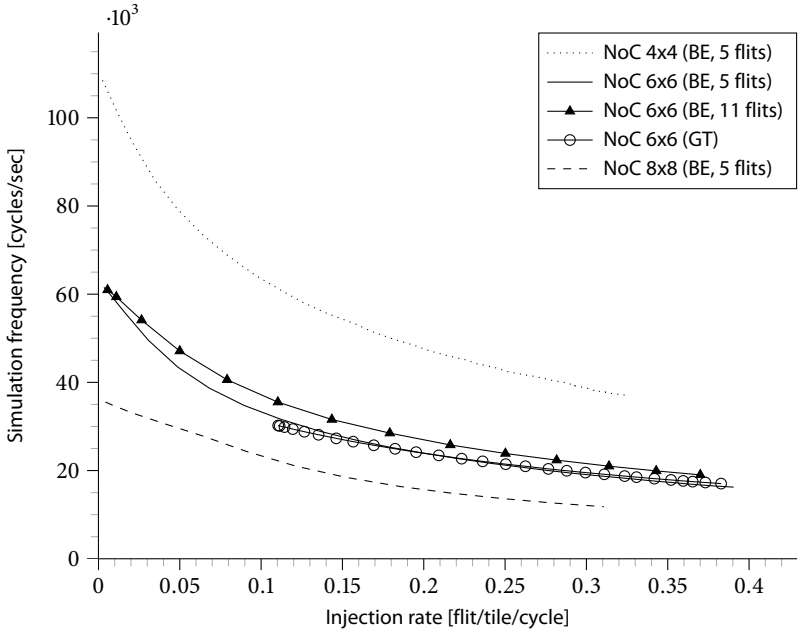


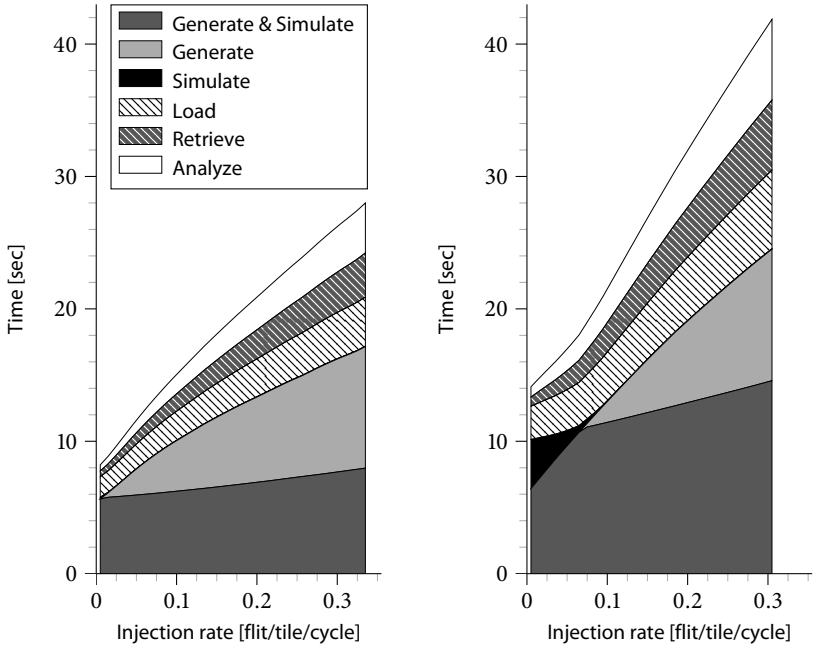
Figure 8.8 – Simulation frequency versus injection rate

on the extra flits that are injected in the network.

The latency distribution presented in figures 5.5 and 5.6 required approximately 15 million clock cycles. Although it is not directly visible in these figure this level of detail is only possible with long simulation times. In the SystemC simulations, this would have taken weeks and with our SHILS the results were obtained in less than five hours for 30 different levels of BE loads.

The time required per process of figure 8.7, to simulate 0.5 million system cycles, is depicted in figure 8.9 for the best-effort tests with five data flits in each packet for two network topologies. In these tests we run the generation and simulate processes in parallel on a single ARM and the Virtex-II 8000 FPGA. For most loads, the simulation process requires less time than the generation process, which completely hides the simulation time spent in the FPGA. Only in the 8×8 topology and under low loads we can achieve lower simulation times if we run more processes in parallel to the simulation process on the FPGA. Although the exact time varies between tests, in general we see that approximately 55% of the time is spent on generation of the stimuli. Loading, retrieving and analysing data require 15% of the time each. The time for analysis increases if complex analysis techniques are included (e.g. determine variance of inter-flit arrival times). As most of the time is spent in software and not by the FPGA, there is no reason to increase the FPGA's delta cycle frequency.

The minimum number of delta cycles per system cycle is equal to the number



(a) 6x6 NoC

(b) 8x8 NoC

Figure 8.9 – Simulation times for 0.5 million cycles

of routers of the NoC, because each router is considered unstable at the start of a system cycle. In the extra delta cycles, unstable routers are re-evaluated as explained in section 8.2.2. The extra number of delta cycles mainly depends on the load that is offered to the network, which causes more activity on the combinational links between the routers. The increase in delta cycles has a proportional relation with the injection rate as visible in figure 8.9.

8.5 Discussion

The sequential simulation method described in this chapter, can be used to simulate a parallel system on any sequential processor. The number of bits that can be updated in parallel in a delta cycle is much larger in an FPGA compared to a processor. It can read and write a large number of bits in the available internal RAM. Furthermore, the FPGA has a large amount of logic, which can evaluate the functionality of different parts of a router in parallel. This also makes it possible to do both the evaluation and update of the registers in parallel. As shown by the case study, a delta cycle can be evaluated in two FPGA clock cycles.

8.5.1 Flexibility of the FPGA simulator

The NoC that is simulated on the FPGA is a homogeneous wormhole switching network with virtual channel flow control with a torus topology. The software on the ARM can change the network size from 1×2 to any two dimensional size with a maximum number of 256 routers. The maximum number of routers is limited by the amount of RAM that is required for the cyclic buffers and the router's state.

In the current simulator we have the same functionality for all the routers. It is possible to select a different router functionality depending on the position in the network. The limiting factor is the number of registers in the router. The topology of a network can either be a torus or a mesh, which is determined by software. Other topologies are possible and only require a change in the addressing function of the link memories in the FPGA.

The technique used for the NoC simulator can also be used for testing other parallel systems on an FPGA, in particular systolic algorithms with many equal parts and a small state space. If the source code for synthesis is available, it is relatively straightforward to modify the code in the sequential framework. In case of the GuarVC router architecture we added approximately 30% extra lines to the original VHDL description. However, manual transformation is sensitive to error and labour intensive. Therefore, in the next section we will describe our ideas to automatically create the whole simulator.

Heterogeneous systems can be supported too, as long as the required extra combinational logic fits in the FPGA. In the NoC case, less than 10% of the logic resources are used for combinational circuitry of the routers. The registers can be mapped in the same memory space.

8.5.2 Automated creation of the simulator

The required SHILS, as described in this chapter, is created manually. However, the process to create the SHILS is error prone, due to the required manual transformations of the design. The automatic transformation of an arbitrary design can be embedded in a general design flow that automatically generates the sequential simulator. The details and ideas for this this design flow are presented appendix D.

The first step to automate the creation is automated detection and extraction of synchronous elements. Their input and output signals have to be re-routed to RAM and some additional logic is to be inserted to emulate the specific behaviour of the extracted synchronous elements. In appendix D we also show the initial results obtained with the automated extraction of synchronous elements. We use the GuarVC NoC, as used in this chapter, to compare the resulting hardware requirements.

The size of the resulting automatically generated simulator is almost equivalent to the manual transformation. The automated flow generates fewer block RAMs and flip-flops, but requires some additional function generators. The development of the other individual steps and tools within this flow and possible optimization steps are left as future work.

8.6 Conclusion

In this chapter we described a simulation method for SoC designs on an FPGA. Where other alternatives simulate only small designs in a single FPGA or require multiple FPGAs for a more realistic design, we are able to use a single FPGA for large designs. We use the regular structure of NoC and SoC architectures and transform the design to simulate it sequentially in the FPGA.

The method is especially suitable for parallel systems where lengthy cycle- and bit accurate simulations are required. These tend to demand a considerable amount of time using software-only simulation on a desktop PC. Using an FPGA, we are able to speed up the simulation with a factor of 80–300 compared to a SystemC simulation without loss of accuracy and a small code difference with the original VHDL source code. Although an FPGA cannot handle high frequencies, it benefits from its large internal memory bandwidth and parallel execution of many combinational circuits. We apply the modifications at RTL-level which makes it possible to run the simulator on any FPGA device.

With the method we simulated our NoC to validate the method and it gave us detailed insights in the behaviour of our GuarVC network. Generation of detailed latency histograms was possible, because millions of system cycles could be simulated in less than a minute. This enabled us to modify and verify the router's circuitry, like, for example, header compression, or adapt the routing function for optimal header ID selection as demonstrated in chapter 5.

We used a homogeneous NoC to test the performance of the simulator. In this case study the largest amount of simulation time was spent on generating stimuli and analysing the results in software. A simple improvement by offloading the random number generation to the FPGA gave an extra 50% simulation speed. An extra factor three to four improvement of the software is necessary before the FPGA simulation itself or I/O with the FPGA becomes a bottleneck.

Initially, the simulator was created manually. We also presented our ideas for a design flow to automatically generate an FPGA-based sequential hardware-in-the-loop simulator to simulate large multi-core architectures. The already developed tools of the design flow enable automated extraction of the hypercell's state. The size of the resulting automatically generated simulator is almost equivalent to the manual transformation. The automated flow generates fewer block RAMs and flip-flops, but requires some additional function generators. The development of the complete design flow is work in progress.

Chapter 9

Conclusion

A general purpose processor used to consist of a single processing core, which performed and controlled all tasks on the chip. Its functionality and maximum clock frequency grew steadily over the years. Due to the continuous increase of the number of transistors available on-chip and the operational clock frequency, it became impossible to reach every function within the chip in a single clock cycle. Furthermore, centralized control becomes hard with the increase in functionality. This leads to architectures with a set of independent processing cores integrated into a single chip.

For these multi-core architectures a Network-on-Chip is required to interconnect the individual cores. The NoC replaces the global wires, it supports sharing of wires for communication, and enables concurrent communication of concurrently handled data packets. The latter is important, because it increases the communication performance in comparison with single channel bus-based architectures.

In this thesis we explored the NoC paradigm by detailed implementation, integration and characterization of multiple NoC router architectures. Compared to a high-level design space exploration study, higher design and development costs were expected for this detailed exploration. Costs in this exploration consist of, but are not limited to, required knowledge, invested time, and required resources.

Our hypothesis is that the extra costs required for this detailed exploration of the NoC concept are acceptable considering the valuable results that can be obtained. Before we discuss the trade-off between the results and the costs to obtain those, we present a summary of results that are presented in this thesis.

Since the introduction of the NoC paradigm by Benini and De Micheli [15], Dally and Towles [38], and Sgroi et al. [118], a wide range of router architectures is proposed in the last decade. We presented a selection of those architectures in chapter 2.

In our detailed exploration we limited the scope of communication architectures to those that are suitable for embedding in a heterogeneous multi-core SoC for streaming applications. Before we propose an architecture, we examine the common characteristics of the streaming applications. Six examples of these type of applications—HiperLAN/2, WiMAX, UMTS, DRM, DAB, and MPEG-4—are described in chapter 3.

We presented the process graph of all applications and determined the bandwidth requirements for the major interprocess communication between the main processes. Furthermore, we had a fully implemented version of the process graph available for three applications. This enabled us to extract typical data packets from the individual edges and extract the packet's switch activity. This switch activity has a direct influence on the architecture's power consumption. The majority of the edges have an activity that shows a high similarity with random data, which has an average switch activity of 25%. However, especially in the MPEG-4 application, much lower activity factors were present, due to the large number of constant values in the packets.

The bandwidth requirements of interprocess communication in different applications varies widely from several kbit/s (DRM) up to more than 0.5 Gbit/s (HiperLAN/2) and changes along the process graph. The input of most processes arrives at a fixed rate and the grouped data in packets have a packet rate that is determined by the AD-converter at the input or the DA-converter at the output. This continuous flow is probably to be handled by a NoC that supports QoS for the interprocess communication.

An overall important characteristic is the life-time of communication streams in a process graph. For all applications a human is involved, who will most likely use an application for a time that exceeds a few seconds. Except during initialization, the process graphs do not change at run-time. We therefore can assume that data streams will have a semi-static and periodic behaviour. This means that for a long period of time subsequent data items of a stream follow the same route.

Due to the observations that streaming applications require QoS, have an expected long life-time, and the assumption that the architecture is controlled by a central entity, the CCN, lead to the first architecture that is used in the exploration. This is a packet switched virtual channel router with QoS support, referred to as GuarVC. QoS is possible via the unique assignment of a VC to a stream that requires this service, and deterministic arbitration per physical channel. This results in an upper bound on the latency of a packet, given its route, packet length, and the number of occupied VCs per physical channel along the route. The original architecture is proposed by Kavaldjiev [78]. In this thesis we improved this architecture, which partly was a result from the detailed latency analysis discussed in this thesis.

Any packet switched architecture has a major disadvantage, which is the large resource, area as well as energy, requirement for the buffering. In combination with the observation that the analysed streaming applications have a semi-static behaviour, we described and implemented a circuit-switched router architecture, referred to as CS. This architecture replaces the VCs, which are time multiplexed, by parallel physical channels on a link between two routers. To reduce the number

of wires required for this link, we applied serialization, such that the total raw bandwidth of the links in the CS network is identical to the GuarVC network. The second characteristic of this circuit switched network is the complete separation of data and control. The router's crossbar configuration, and with that the route of the data, is controlled via a separate control network.

The two routers are compared with two other packet switched routers, designed at the University of Cambridge. Their wormhole router (WH) and speculative virtual channel router (SpecVC) are tailored to more randomly distributed traffic. In terms of resource usage the WH router requires the least amount of resources, due to a single small input queue per input port. The CS router is slightly bigger compared to this small packet switched router, due to a relative large crossbar and serialization interface. The two virtual channel routers, GuarVC and SpecVC, occupy nearly the same area, of which the largest part is consumed by the input buffers. The GuarVC has a slightly bigger crossbar, due to its asymmetric size that enables a simple QoS mechanism. The SpecVC is the largest router in this comparison. Its larger area is caused by the extra speculation logic, which reduces the average packet latency and increases the utilization.

Besides the area occupancy of the four routers, we compared the routers on their packet latency and energy consumption. The latency analysis is solely performed on the three packet switched routers. The circuit switched router has a deterministic latency, due to congestion free routes. We applied synthetic traffic patterns to the GuarVC router. Initially, we used the design as proposed by Kavaldjiev. Adjustments of the protocol, which reduced the header size, resulted in both a reduced latency and an increased saturation point. The improved version of the architecture is compared with the two packet switched routers of the University of Cambridge.

For all three architectures an eight by eight mesh network was created and synthetic traffic sources injected packets with a fixed payload and XY-routing was applied. Packet destinations are either randomly distributed or based on Rent's rule. The latter is to simulate localized traffic in the NoC.

For both destination distributions, both virtual channel routers have a higher saturation point in comparison with the wormhole router. The GuarVC has a rather static allocation scheme that causes a longer delay for the header flit, which results in a higher average latency compared to the other two packet switched routers. The other penalty for the GuarVC is its organization of the packet, which results in an additional header flit with the packet's routing information. The advantage is the reduction of wires per physical channel and this header overhead is negligible for larger payloads.

The major difference between the GuarVC and the other packet switched alternatives is its QoS support. We measured the effect of various loads of randomly distributed BE packets on the latency of GT packets. A preliminary test, as performed by Kavaldjiev, is extended, such that results can be analysed and interpreted in more detail.

Previously, we only were able to present the maximum and average latency, due to the relatively small number of packets that could be transmitted in a single test, which already required hours of simulation. With the use of the FPGA simulator

presented in this thesis, we were able to simulator 15 million clock cycles in a few minutes. This enormous number of clock cycles enabled us to collect the latency behaviour of a sufficient number of packets, and to calculate detailed statistics.

The GT packets' latency distribution is plotted versus the additional BE that is injected in the traffic. Despite the increase in BE traffic, a GT packet will always arrive within the bounded latency. The increased network load solely causes a higher jitter in the GT packet latency. Jitter can be prevented by traffic shaping of the input streams, but this also increases the average and maximum packet latency. Depending on the system requirements, one has to decide what is desirable: hardly any jitter with a large latency, or a lower average latency that varies with the network load.

For the replacement of global wires and on-chip buses with a NoC it is also important to know the energy required to transport the data via the NoC. The power evaluation described in this thesis showed the effects of design choices on the architecture's energy consumption. Evaluation is performed on placed-and-routed designs and with traffic traces obtained in the FPGA simulator. In contrast with the latency simulations, the required number of simulated clock cycles to obtain useful results is low, but the required processing time per clock cycle is considerably more due to the detailed models used.

The initial energy measurements of the GuarVC pointed out that a significant amount of power is consumed in idle mode, when no data is transported by the NoC. Despite the small technology size of 90 nm, the leakage was not the origin of this wasted energy. Up to 96% of the power was consumed by the clock tree and its directly connected synchronous elements.

Automatic insertion of clock gating elements, with help of small adjustments in the HDL code, reduced this standby dynamic power by an order of magnitude, to a few mW per router at a clock frequency of 250 MHz with an equal share for dynamic and static power. A similar standby power consumption of a clock gated CS router is observed.

The extra power consumed by both the routers when data is transported is primarily data dependent and is consumed primarily by the routers' data path—flit buffers, and crossbar. The power increase is only observed in the dynamic power. For the analysis we varied the offered traffic, switch activity of the data, and the GuarVC architecture itself.

The increase in dynamic power is proportional to the number of transported bits from input to output ports, and is sensitive to the switch activity of the transported data. Transmission of packets with an 'empty' payload of constant values requires approximately three times less energy in comparison with random payload for the GuarVC router. We may conclude that one third of the energy is consumed by the switch activity in the control mechanisms to handle the individual flits and two third is contributed to the switch activity of the data. The CS router requires virtually no energy to transport packets with an 'empty' payload. The energy per transported bit is approximately half the energy that is required by the GuarVC router. However, the CS router has a load independent increase of the total standby power when lanes are configured.

The power comparison of the four router architectures showed no large differences, due to the aggressive clock gating included in each architecture. This clock gating reduces the benefit of the lower buffer requirements for the circuit switched router. The CS router energy's benefit is solely observed for the standby power, in which the router consumes at most three times less power in comparison to the other three architectures.

However, due to the increased crossbar size in the CS router, its data dependent power consumption is comparable to the three packet switched routers. Two of the major discriminators in the total energy consumption per packet is the total amount of data transported, which is slightly higher for the GuarVC router, and the size of the crossbar. An increase of congestion has less impact on the average energy per packet. Therefore, future power efficient router designs will benefit from a small investment in the router's control logic, such that the overhead in the transported packet and required crossbar size is minimized. From an energy perspective the use of packet switched routers is preferred over circuit switched routers as they give more flexibility for the same energy budget.

Another part of the detailed exploration was the integration of a NoC into a prototype chip, called Annabelle, which was developed within the 4S project. This SoC consists of an ARM926 processor core complemented by five custom ASIC blocks, four domain specific coarse-grain reconfigurable Montium cores, and standard peripherals.

The individual processing cores are interconnected by a hybrid communication architecture. The ARM, ASIC blocks and peripherals are connected, as either master or slave, to a 5-layer AHB. In comparison with the initial circuit switched router architecture, the four reconfigurable cores are interconnected by two slightly simplified circuit switched routers. The routers receive data from the AHB via a slave bridge.

In this SoC the ARM fully controls the configuration of the individual processing cores. The circuit switched network was chosen as the best alternative, because of the small network topology, the necessity for concurrent communication between the reconfigurable cores, and it enables direct control of routers via the AHB bridge by the ARM processor. For this small network and relatively small number of processing cores the serialization is omitted to increase the bandwidth and to reduce the overhead in area.

The latency analysis performed in this thesis required cycle and bit accurate simulations of the NoC architecture. Cycle and bit accurate simulation of hardware designs require a lot of time using software based simulators. We proposed a HIL-based simulator framework that simulates the hardware design in a single FPGA. The framework requires a transformation of the NoC architecture and instantiates the combinational functionality of a single router. Using delta cycles, similar to the mechanism used in VHDL simulation, the whole NoC architecture's state is updated sequentially.

Compared to a SystemC based simulation of the same network architecture of a 6×6 mesh topology of GuarVC routers, this simulator is a factor 80–300 times faster. The major benefit of an FPGA in comparison with a regular desktop PC is

its enormous memory bandwidth due to the parallel access to its local on-chip memory and the sea of reconfigurable hardware on which the design maps naturally. This parallelism compensates the FPGA's lower operational frequency and makes a quick update of the overall system's state possible in only a small number of clock cycles. The current bottleneck in the simulator, as presented, is the generation of stimuli and analysis of the results on a relatively low performance ARM processor.

The study in this thesis consists of a detailed exploration of the NoC paradigm by means of the implementation, integration and comparison of various router architectures. Our hypothesis was that the detailed exploration of the NoC would deliver valuable results against acceptable costs.

In conclusion to the overall results and the main research question, we conclude that it is useful and required to perform a detailed exploration of on-chip architectures despite its additional costs. The largest part of the extra required costs are in the initial setup of the analysis frameworks and the creation of the designs investigated. Furthermore, with the creation of the FPGA-based simulator framework, we have a new approach to tremendously reduce the time for cycle-accurate simulation. This makes long realistic simulations possible within a relative short time, without the creation and risk of translation errors between the simulation and synthesis model of the hardware design.

Once the framework is ready, an enormous amount of valuable results can be obtained. For example, the detailed energy breakdown showed were architectures can be improved or which part of any architecture requires considerable attention during the design, e.g. to save energy. Furthermore, due to the enormous amount of data that is collected, not only a single estimate can be presented, but also the accuracy and other statistics of this data set. These figures can be used in higher-level models.

For example, to calculate a coarse estimate of the power consumption of the GuarVC and CS router for a fixed frequency of 250 MHz the design can use respectively two and three parameters. Each GuarVC router has a fixed standby power of 5.83 mW, and requires 0.38 pJ/bit to transport both the payload and header flits. Each CS router has a fixed standby power of 3.54 mW and an additional 0.4 mW per lane that is enabled, and the router requires 0.18 pJ/bit to transport the payload.

While designing the two presented router architectures it helped to have a holistic overview of the system in which the NoC is integrated. For example, a CCN made the implementation of QoS relatively easy. However, a holistic control and view of the system will not be efficient without the information of the detailed figures.

9.1 Main Contributions of this Thesis

The contributions of this thesis are made in different areas:

- We performed an exploration of a range of streaming applications and their communication requirements. These applications are mapped to heterogeneous SoC architectures, which gave insights in the detailed communication

requirements. We present the process graphs of these applications together with the required communication bandwidths for the individual edges. Besides the bandwidth requirements we analysed the content of the individual data streams of three applications. The content of the packets gave information on the toggle-rate statistics, which has a direct influence on the power consumption of the communication architecture. (*Chapter 3*)

- We designed and implemented two different NoC router architectures, both tailored to streaming applications. (*Chapter 4*)
  - » A circuit switched NoC architecture that uses some of the common characteristics found during the application analysis.
  - » An existing packet switched router is improved.
- Both NoC architectures are analysed on their area requirements and energy consumption. For the packet switched router we also measured the network latency under various traffic conditions and network sizes. These performance results are compared with two other NoC architectures. For area and latency comparison existing metrics are used and we extended the latency measurements with scenarios to test traffic that require QoS. For the power consumption we propose a number of tests that can be applied to a router, such that its performance to transport various types of traffic can be characterized. (*Chapters 4 to 6*)
- For the Annabelle SoC with a small network of processing tiles, realized in 0.13 µm technology, the circuit switched alternative was selected, implemented and evaluated. (*Chapter 7*)
- We proposed and designed a new simulator framework to perform fast bit and cycle accurate simulations of multi-core architectures. Instead of a software based simulation, the multi-core architecture is mapped onto an FPGA resulting in a HIL-based simulator. The homogeneous or heterogeneous multi-core architecture is transformed, such that the number of required FPGA resources is reduced drastically. The proposed SHILS performs bit and cycle accurate simulations of hardware designs without prohibitive long simulation times that are common in software based simulators. The framework is applied to the improved packet switched NoC architecture, which enabled a thorough analysis. Where others present only average results this framework enables the designer to obtain more detailed characterization of the simulated architecture. (*Chapters 5 and 8*)

9.2 Future Work

The research, described in this thesis, can be extended in various ways.

First, quite a number of NoC architectures are proposed since the beginning of this decade. Various techniques are combined in various architectures. On-chip, the networks can not use a large quantity of resources, but the environment is rather controlled. This makes it relatively easy to include simple QoS mechanisms for

individual streams. When these many-core systems scale in size, and are integrated in larger systems we foresee that the chip packages will be interconnected too, e.g. on a single PCB. This creates a network of cores beyond a single chip. Although the off-chip environment is different from the on-chip connection, for arbitrary cores to communicate in the system it would be beneficial to have protocols that scale beyond a single chip. Such a protocol should offer similar services for both on- and off-chip. The NoC is then extended to the Network-off-Chip. A specific core will not be aware that it is communicating with a core on another chip. This is part of the FP7 Cutting edge Reconfigurable ICs for Streaming Processing (CRISP) project [31].

Second, we have seen an equal share of dynamic and static power if a router is in idle mode at an operational frequency of 250 MHz. This standby power might be a limiting factor to deploy large networks in future systems. Especially, when the leakage power will increase with technology scaling, additional techniques besides clock gating are necessary to reduce the standby power. It is the author's believe, that part of the leakage problems will be solved by the introduction of new types of standard cell libraries and new materials that have a lower leakage contribution. However, application specific information on the activity of parts of the system over time can help to completely power-off the inactive parts. Switching off parts of the routers of the NoC (power gating) will be one of the most challenging tasks as a local router is required for global communication.

Third, the simulation technique described in chapter 8 was evaluated by a case study, in which we manually modified the NoC router. We demonstrated a considerable reduction in simulation time for cycle accurate simulations. However, the process to create the SHILS is error prone, due to the required manual transformations of the design. The first step to automate this creation is automated detection and extraction of synchronous elements. Their input and output signals have to be re-routed to RAM and some additional logic is to be inserted to emulate the specific behaviour of the extracted synchronous elements.

The automatic transformation of an arbitrary design can be embedded in a general design flow that automatically generates the sequential simulator. We presented our ideas for this design flow in appendix D. The development of the individual steps and tools within this flow and possible optimization steps are left as future work.

List of Symbols

Summary of all symbols used in this thesis.

C	capacitance
H	number of hops between the packet's source and destination
I_{GIDL}	gate-induced drain leakage
I_{gate}	gate leakage
I_{pn}	reverse-bias pn junction leakage
I_{punch}	channel punchthrough current
I_{sub}	subthreshold leakage
L	total length of a single packet in bits
P	power
P_{dyn}	dynamic power
P_{stat}	static power
T_g	guard time
T_u	useful symbol time
V	voltage
V_{th}	threshold voltage
W	number of wires of a channel or lane
α	activity
f	frequency
t_r	delay per hop for decoding of the packet's and allocation of resources
t_w	delay to transport a single phit between two routers

List of Acronyms

Summary of all acronyms used in this thesis.

3G	Third Generation
AAC	Advanced Audio Coding
AD	Analog-to-Digital
ADC	Analog-to-Digital Converter
AHB	AMBA High-performance Bus
ALU	Arithmetic Logic Unit
AM	Amplitude Modulation
AMBA	Advanced Microcontroller Bus Architecture
APB	Advanced Peripheral Bus
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
ASIC	Application Specific Integrated Circuit
AXI	Advanced eXtensible Interface
BE	Best-effort
BGA	Ball Grid Array
BPSK	Binary Phase-Shift Keying
BTBT	Band-to-Band Tunnelling
CAD	Computer Aided Design
CCN	Central Coordinating Node
CD	Compact Disc
CDMA	Code Division Multiple Access
CLB	Configurable Logic Block
CMOS	Complimentary Metal Oxide Semiconductor
CONLAN	Consensus Language
CRC	Cyclic Redundancy Check
CRISP	Cutting edge Reconfigurable ICs for Streaming Processing
CS	colour space
DA	Digital-to-Analog
DAB	Digital Audio Broadcasting
DC	Direct Current
DCT	Discrete Cosine Transform
DDC	Digital Down Converter
DFS	Dynamic Frequency Scaling

DFT	Discrete Fourier Transform
DIBL	drain-induced barrier lowering
DMA	direct memory access
DMB	Digital Multimedia Broadcasting
DRM	Digital Radio Mondiale
DSP	digital signal processing
DTL	Device Transaction Level
DVD	Digital Versatile Disc
DVS	Dynamic Voltage Scaling
EPM	Enhanced Packet Mode
FDD	Frequency Division Duplex
FDM	Frequency Division Multiplexing
FFT	Fast Fourier Transform
FIB	Fast Information Block
FM	Frequency Modulation
FPGA	Field Programmable Gate Array
FPS	frames per second
FPU	Floating Point Unit
GALS	Globally-Asynchronous Locally-Synchronous
GPP	General Purpose Processor
GS	Guaranteed Services
GT	Guaranteed-throughput
HBR	has-been-read
HDL	Hardware Description Language
HIL	Hardware-In-the-Loop
IC	Integrated Circuit
ID	Identifier
IEEE	Institute of Electrical and Electronics Engineers, Inc.
IFFT	inverse Fast Fourier Transform
IP	Internet Protocol
IP	Intellectual Property
LAN	Local Area Network
LOS	line-of-sight
LUT	Lookup Table
MAC	Medium Access Control
MAC	Multiply Accumulate
MAN	Metropolitan Area Network
MB	macroblock
MC	Motion Compensation
MIN	Multistage Interconnection Network
ML	Multi-Layer
MMU	Memory Management Unit

MOS	Metal Oxide Semiconductor
MOSFET	Metal Oxide Semiconductor Field Effect Transistor
MP	Multi-Processor
MPEG	Motion Picture Expert Group
MPSoC	Multi-Processor System-on-Chip
MSC	Main Service Channel
NI	Network Interface
NLOS	non-line-of-sight
NoC	Network-on-Chip
OCCN	On-Chip Communication Network
OCP	Open Core Protocol
OFDM	Orthogonal Frequency Division Multiplexing
OS	Operating System
PC	Personal Computer
PCB	Printed Circuit Board
PLI	Programming Language Interface
QAM	Quadrature Amplitude Modulation
QoS	Quality of Service
QPSK	Quadrature Phase-Shift Keying
RAM	Random-Access Memory
RISC	Reduced Instruction Set Computer
RS	Reed-Solomon
RTL	Register Transfer Level
SDM	Space Division Multiplexing
SDR	Software Defined Radio
SF	spreading factor
SHILS	Sequential Hardware-In-the-Loop Simulator
SMIT	Spatial Mapping Tool
SNR	Signal-to-Noise Ratio
SoC	System-on-Chip
SOI	silicon on insulator
SRAM	Static Random-Access Memory
TCC	Transactional Coherence and Consistency
TDD	Time Division Duplex
TDM	Time Division Multiplexing
TDMA	Time Division Multiple Access
TLM	Transaction Level Modelling
UMTS	Universal Mobile Telecommunications System
VC	Virtual Channel

VHDL	Very-High-Speed Integrated Circuits Hardware Description Language
VLC	Variable Length Coding
VLSI	Very-Large-Scale Integration
W-CDMA	Wideband Code Division Multiple Access
WLAN	Wireless Local Area Network

Bibliography

[1] A. Agarwal, Saibal Mukhopadhyay, A. Raychowdhury, and Kaushik Roy. Leakage power analysis and reduction: models, estimation and tools. In *IEE Proceedings-Computers and Digital Techniques*, volume 152, pages 353–368, May 2005. DOI 10.1049/ip-cdt:20045084.

[2] Saman Amarasinghe. The future. Lecture slides of the 6.189 Multicore Programming Primer, 2007. URL `http://cag.csail.mit.edu/ps3/lectures/6.189-lecture18-future.pdf`.

[3] Adrijean Andriahantenaina and Alain Greiner. Micro-network for SoC: Implementation of a 32-port SPIN network. In *Proceedings of the Conference on Design, Automation and Test in Europe*, pages 1128–1129, Washington, Washington, D.C., March 2003. IEEE Computer Society.

[4] Adrijean Andriahantenaina, Hervé Charlery, Alain Greiner, Laurent Mortiez, and Cesar Albenes Zeferino. SPIN: a scalable, packet switched, on-chip micro-network. In *Proceedings of the Conference on Design, Automation and Test in Europe*, pages 70–73, Washington, Washington, D.C., March 2003. IEEE Computer Society.

[5] Federico Angiolini, Paulo Meloni, Salvatore M. Carta, Luigi Raffo, and Luca Benini. A layout-aware analysis of networks-on-chip and traditional interconnects for MPSoCs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 26(3): 421–434, March 2007. ISSN 0278-0070. DOI 10.1109/TCAD.2006.888287.

[6] ARM. ARM926EJ-S overview, 2008. URL `http://www.arm.com`.

[7] Arvind, Krste Asanovic, Derek Chiou, James C. Hoe, Christos Kozyrakis, Shih-Lien Lu, Mark Oskin, David Patterson, Jan Rabaey, and John Wawrynek. RAMP: Research Accelerator for Multiple Processors - community vision for a shared experimental parallel HW/SW platform. Technical report, MIT, 2005.

[8] Krste Asanovic, Ras Bodik, Bryan Christopher Catanzaro, Joseph James Gebis, Parry Husbands, Kurt Keutzer, David A. Patterson, William Lester Plishker, John Shalf, Samuel Webb Williams, and Katherine A. Yelick. The landscape of parallel computing research: A view from berkeley. Technical Report UCB/EECS-2006-183, EECS Department, University of California, Berkeley, December 2006. URL `http://www.eecs.berkeley.edu/Pubs/TechRpts/2006/EECS-2006-183.html`.

[9] John Bainbridge and Steve Furber. Chain: A delay-insensitive chip area interconnect. *IEEE Micro*, 22(5):16–23, 2002. ISSN 0272-1732. DOI 10.1109/MM.2002.1044296.

[10] James Balfour and William James Dally. Design tradeoffs for tiled cmp on-chip networks. In *ACM International Conference on Supercomputing*, pages 187–198, June 2006. ISBN 1-59593-282-8.

[11] Kaustav Banerjee and Amit Mehrotra. A power-optimal repeater insertion methodology for global interconnects in nanometer designs. *IEEE Transactions on Electron Devices*, 49(11):2001–2007, November 2002.

[12] Nilanjan Banerjee, Praveen Vellanki, and Karam S. Chatha. A power and performance model for network-on-chip architectures. In *Proceedings of the Conference on Design, Automation and Test in Europe*, pages 1250–1255, Washington, Washington, D.C., February 2004. IEEE Computer Society. ISBN 0-7695-2085-5.

[13] Paul Baran. *Introduction to Distributed Communications Networks*, volume 1 of *On Distributed Communications Series*. RAND Corporation, 1964.

[14] V. Baumgarte, G. Ehlers, F. May, A. Nückel, M. Vorbach, and M. Weinhardt. PACT XPP-A self-reconfigurable data processing architecture. *The Journal of Supercomputing*, 26(2):167–184, September 2003. ISSN 0920-8542. DOI 10.1023/A:1024499601571.

[15] Luca Benini and Giovanni De Micheli. Networks on chips: A new SoC paradigm. *IEEE Computer*, 35(1):70–78, January 2002.

[16] Luca Benini and Giovanni De Micheli. System-level power optimization: techniques and tools. *ACM Transactions on Design Automation of Electronic Systems*, 5(2):115–192, 2000. ISSN 1084-4309. DOI 10.1145/335043.335044.

[17] Luca Benini, Davide Bertozi, Alessandro Bogliolo, Francesco Menichelli, and Mauro Olivieri. MPARM: Exploring the multi-processor soc design space with systemc. *Journal of VLSI Signal Processing*, 41(2):169–182, September 2005. DOI 10.1007/s11265-005-6648-1.

[18] Davide Bertozi, Antione Jalabert, Srinivasan Murali, Rutuparna Tamhankar, Stergios Stergiou, Luca Benini, and Giovanni De Micheli. NoC synthesis flow for customized domain specific multiprocessor systems-on-chip. *IEEE Transactions on Parallel Distributed Systems*, 16(2):113–129, 2005. ISSN 1045-9219. DOI 10.1109/TPDS.2005.22.

[19] Tobias Bjerregaard. *The MANGO clockless network-on-chip: Concepts and implementation*. PhD thesis, Technical University of Denmark, Lyngby, Denmark, 2005.

[20] Tobias Bjerregaard and Shankar Mahadevan. A survey of research and practices of network-on-chip. *ACM Computing Surveys*, 38(1):1, 2006. ISSN 0360–0300. DOI 10.1145/1132952.1132953.

[21] Tobias Bjerregaard and Sparso. A router architecture for connection-oriented service guarantees in the MANGO clockless network-on-chip. In *Proceedings of the Conference on Design, Automation and Test in Europe*, pages 1226–1231, Washington, Washington, D.C., March 2005. IEEE Computer Society. ISBN 0-7695-2288-2.

[22] Evgeny Bolotin, Israel Cidon, Ran Ginosar, and Avinoam Kolodny. QNoC: QoS architecture and design process for network on chip. *Journal of Systems Architecture*, 50(2–3):105–128, 2004. ISSN 1383-7621. DOI 10.1016/j.sysarc.2003.07.004.

[23] Robert Bringhurst. *The Elements of Typographic Style*. Hartley and Marks Publishers, Vancouver, Canada, 3.1 edition, 2005. ISBN 0-88179-206-3.

[24] Chameleon. Chameleon: Reconfigurable computing in hand-held multimedia computers, December 1999. URL http://www.stw.nl.

[25] Chen Chang, John Wawrynek, and Robert W. Brodersen. BEE2: A high-end reconfigurable computing system. *IEEE Design & Test of Computers*, 22(2):114–125, March-April 2005. DOI 10.1109/MDT.2005.30.

[26] Robert Chau, Justin Brask, Suman Datta, Gilbert Dewey, Mark Doczy, Brian Doyle, Jack Kavalieros, Ben Jin, Matthew Metz, Amlan Majumdar, , and Marko Radosavljevic. Application of high-k gate dielectrics and metal gate electrodes to enable silicon and non-silicon logic nanotechnology. *Microelectronic Engineering*, 80:1–6, June 2005.

[27] Xuning Chen and Li-Shiuan Peh. Leakage power modeling and optimization in interconnection networks. In *Proceedings of the International Symposium on Low Power Electronics and Design*, pages 90–95, August 2003. DOI 10.1109/LPE.2003.1231841.

[28] Leonardo Chiariglione. The MPEG home page, 2007. URL http://www.chiariglione.org/mpeg.

[29] Erik S. Chung, Eriko Nurvitadhi, James C. Hoe, Falsaf, and Ken Mai. Protoflex: FPGA-accelerated hybrid functional simulation. Technical Report 2007-2, Computer Architecture Lab at Carnegie Mellon (CALCM), February 2007.

[30] Marcello Coppola, Stephane Curaba, Miltos D. Grammatikakis, Locatell, Giuseppe Maruccia, and Franscesco Papariello. OCCN: A NoC modeling framework for design exploration. *Journal of Systems Architecture: the EUROMICRO Journal*, 50(2–3): 129–163, 2004.

[31] Cutting edge Reconfigurable ICs for Stream Processing. CRISP project, 2008. URL http://www.crisp-project.eu.

[32] Dall'Osso, Gianluca Biccari, Luca Giovannini, Davide Bertozi, and Luca Benini. xpipes: a latency insensitive parameterized network-on-chip architecture for multi-processor SoCs. In *Proceedings of the International Conference on Computer Design*, page 536, Washington, DC, USA, October 2003. IEEE Computer Society. ISBN 0-7695-2025-1.

[33] William James Dally. Performance analysis of k-ary n-cube interconnection networks. *IEEE Transactions on Computers*, 39(6):775–785, 1990. ISSN 0018–9340. DOI 10.1109/12.53599.

[34] William James Dally. Virtual-channel flow control. *IEEE Transactions on Parallel and Distributed Systems*, 3(2):194–205, 1992. ISSN 1045–9219. DOI 10.1109/71.127260.

[35] William James Dally and Charles L. Seitz. The torus routing chip. *Journal of Distributed Computing*, 1(4):187–196, 1986. DOI 10.1007/BF01660031.

[36] William James Dally and Charles L. Seitz. Deadlock-free message routing in multiprocessor interconnection networks. *Computers, IEEE Transactions on*, C-36(5):547–553, May 1987. ISSN 0018–9340. DOI 10.1109/TC.1987.1676939.

[37] William James Dally and Charles L. Seitz. Deadlock-free message routing in multiprocessor interconnection networks. *IEEE Transactions on Computers*, C-36(5):547–553, May 1987. ISSN 0018–9340. DOI 10.1109/TC.1987.1676939.

[38] William James Dally and Brian Towles. Route packets, not wires: on-chip interconnection networks. *Proceedings of the Conference on Design, Automation and Test in Europe*, pages 684–689, March 2001.

[39] William James Dally and Brian Towles. *Principles and Practices of Interconnection Networks*. Morgan Kaufmann Publishers Inc., San Francisco, California, 2004. ISBN 0-12200-751-4.

[40] William James Dally, Ujval J. Kapasi, Brucek Khailany, Jung Ho Ahn, and Abhishek Das. Stream processors: Programmability and efficiency. *ACM Queue*, 2(1):52–62, 2004. ISSN 1542–7730. DOI 10.1145/984458.984486.

[41] Pablo G. Del Valle, David Atienza, Ivan Magan, Javier G. Flores, Esther A. Perez, Jose M. Mendias, Luca Benini, and Giovanni De Micheli. A complete multi-processor system-on-chip FPGA-based emulation framework. In *IFIP International Conference on Very Large Scale Integration*, pages 140–145, October 2006. DOI 10.1109/VLSISOC.2006.313218.

[42] Sequence Design. Powertheater - low power design & power analysis for nanometer system-on-chip design, 2007. URL `http://www.sequencedesign.com`.

[43] John Dielissen, Andrei Rădulescu, Kees Goossens, and Edwin Rijpkema. Concepts and implementation of the Philips network-on-chip. In *IP-Based SOC Design*, November 2003.

[44] John Dielissen, Andrei Rădulescu, and Kees Goossens. Power measurements and analysis of a network on chip. Technical Note 2005/00282, Philips Research, April 2005.

[45] Andreas Dittrich and Torsten Schorr. Diorama: Real-time DRM receiver for Matlab, February 2006. URL `http://nt.eit.uni-kl.de/forschung/diorama/`. Version 1.1. University of Kaiserslautern, Institute of Telecommunications, D-6766 Kaiserslautern, Germany.

[46] José Duato and Timothy Mark Pinkston. A general theory for deadlock-free adaptive routing using a mixed set of resources. *IEEE Transactions on Parallel and Distributed Systems*, 12(12):1219–1235, 2001. ISSN 1045-9219. DOI 10.1109/71.970556.

[47] José Duato, Sudhakar Yalamanchili, and Lionel M. Ni. *Interconnection Networks: An Engineering Approach*. Morgan Kaufmann Publishers Inc., San Francisco, California, revised printing edition, 2003. ISBN 1-55860-852-4.

[48] Emulation and Verification Engineering. ZeBu-XL system emulator, 2007. URL `http://www.eve-team.com`.

[49] *Digital Audio Broadcasting (DAB) to mobile, portable and fixed receivers*. European Telecommunication Standard Institute (ETSI), Sophia Antipolis, France, ETSI EN 300 401 v1.4.1 edition, January 2006.

[50] *Digital Audio Broadcasting (DAB); Transport of Advanced Audio Coding (AAC) audio*. European Telecommunication Standard Institute (ETSI), Sophia Antipolis, France, ETSI TS 102 563 v1.1.1 edition, February 2007.

[51] *Digital Audio Broadcasting (DAB); DMB video service; User Application Specification*. European Telecommunication Standard Institute (ETSI), Sophia Antipolis, France, ETSI TS 102 428 v1.1.1 edition, June 2005.

[52] *Digital Radio Mondiale (DRM); System Specification*. European Telecommunication Standard Institute (ETSI), Sophia Antipolis, France, ETSI ES 201 980 v2.2.1 edition, October 2005.

[53] *Broadband Radio Access Networks (BRAN); HIPERLAN Type 2; Physical (PHY) layer*. European Telecommunication Standard Institute (ETSI), Sophia Antipolis, France, ETSI TS 101 475 v1.2.2 edition, February 2001.

[54] N. Genko, David Atienza, Giovanni De Micheli, Jose M. Mendias, R. Hermida, and F. Catthoor. A complete network-on-chip emulation framework. In *Proceedings of the Conference on Design, Automation and Test in Europe*, pages 246–251, Washington, Washington, D.C., March 2005. IEEE Computer Society. ISBN 0-7695-2288-2.

[55] Christopher J. Glass and Lionel M. Ni. The turn model for adaptive routing. In *International Symposium on Computer Architecture*, pages 278–287, May 1992.

[56] Kees Goossens, John Dielissen, Om Prakash Gangwal, Santiago González Pestana, Andrei Rădulescu, and Edwin Rijpkema. A design flow for application-specific networks on chip with guaranteed performance to accelerate SOC design and verification. In *Proceedings of the Conference on Design, Automation and Test in Europe*, pages 1182–1187, Washington, Washington, D.C., February 2004. IEEE Computer Society. ISBN 0-7695-2085-5. DOI 10.1109/DATE.2005.11.

[57] Kees Goossens, John Dielissen, and Andrei Rădulescu. The Æthereal network on chip: Concepts, architectures, and implementations. *IEEE Design and Test of Computers*, 22 (5):414–421, Sept-Oct 2005. ISSN 0740-7475. DOI 10.1109/MDT.2005.99.

[58] Daniel Greenfield, Arnab Banerjee, Jeong-Gun Lee, and Simon W. Moore. Implications of Rent's rule for NoC design and its fault-tolerance. In P. Kellenberger, editor, *Proceedings of the ACM/IEEE International Symposium on Networks-on-Chip*, pages 283–294, Los Alamitos, California, May 2007. IEEE Computer Society Press. ISBN 0-7695-2773-6. DOI 10.1109/NOCS.2007.26.

[59] Pierre Guerrier. *Un Réseau d'Interconnexion pour Systèmes Intégrés*. PhD thesis, Université Pierre et Marie Curie, Paris, France, May 2000.

[60] Pierre Guerrier and Alain Greiner. A generic architecture for on-chip packet-switched interconnections. In *Proceedings of the Conference on Design, Automation and Test in Europe*, pages 250–256, Washington, Washington, D.C., 2000. IEEE Computer Society.

[61] Pankaj Gupta and Nick McKeown. Designing and implementing a fast crossbar scheduler. *Micro, IEEE*, 19(1):20–28, Jan/Feb 1999. ISSN 0272-1732. DOI 10.1109/40.748793.

[62] Andreas Hansson and Kees Goossens. Trade-offs in the configuration of a network on chip for multiple use-cases. In P. Kellenberger, editor, *Proceedings of the ACM/IEEE International Symposium on Networks-on-Chip*, pages 233–242, Los Alamitos, California, May 2007. IEEE Computer Society Press. ISBN 0-7695-2773-6. DOI 10.1109/NOCS.2007.45.

[63] Andreas Hansson, Kees Goossens, and Andrei Rădulescu. A unified approach to constrained mapping and routing on network-on-chip architectures. In *Proceedings of the International Conference on Hardware/Software Codesign and System Synthesis*, pages 75–80, New York, New York, September 2005. ACM Press. ISBN 1-59593-161-9. DOI 10.1145/1084834.1084857.

[64] Andreas Hansson, Kees Goossens, and Andrei Rădulescu. A unified approach to mapping and routing on a network on chip for both best-effort and guaranteed service traffic. *VLSI Design*, 2007:Article ID 68432, 16 pages, May 2007. DOI 10.1155/2007/68432. Hindawi Publishing Corporation.

[65] D. Helms, E. Schmidt, and W. Nebel. Leakage in CMOS circuits – an introduction. In *IEEE 14th International Workshop on Power And Timing Modeling, Optimization and Simulation (PATMOS 2004)*, volume 3254/2004 of *Lecture Notes in Computer Science*, pages 17–35, Berlin, Germany, September 2004. Springer. ISBN 978-3-540-23095-3. DOI 10.1007/b100662.

[66] Paul M. Heysters. *Coarse-Grained Reconfigurable Processors*. PhD thesis, University of Twente, Enschede, The Netherlands, September 2004.

[67] A. Holma and A. Toskela. *WCDMA for UMTS: Radio Access for Third Generation Mobile Communications*. John Wiley & Sons, 2001.

[68] Philip K. F. Hölzenspies, Johan L. Hurink, Jan Kuper, and Gerard J. M. Smit. Run-time spatial mapping of streaming applications to a heterogeneous multi-processor system-on-chip (MPSOC). In *Proceedings of the Conference on Design, Automation and Test in Europe*, pages 212–217, Washington, Washington, D.C., March 2008. IEEE Computer Society. ISBN 0-7695-2288-2.

[69] IBM, 2007. URL `http://www.ibm.com`.

[70] *IEEE Standard for Local and metropolitan area networks Part 16: Air Interface for Fixed Broadband Wireless Access Systems*. IEEE, New York, New York, ieee std 802.16-2004 edition, October 2004.

[71] Intel, 2008. URL `http://www.intel.com`.

[72] Randall D. Isaac. The future of CMOS technology. *IBM Journal of Research and Development*, 44(3):369–378, 2000. DOI 10.1147/rd.443.0369.

[73] ITRS. International technology roadmap for semiconductors (ITRS). Technical report, Semiconductor Industry Association, 2005. URL `http://public.itrs.net/`.

[74] ITRS. International technology roadmap for semiconductors (ITRS). Technical report, Semiconductor Industry Association, 2007. URL `http://public.itrs.net/`.

[75] Antione Jalabert, Srinivasan Murali, Luca Benini, and Giovanni De Micheli. xpipesCompiler: A tool for instantiating application specific networks on chip. In *Proceedings of the Conference on Design, Automation and Test in Europe*, Washington, Washington, D.C., February 2004. IEEE Computer Society. ISBN 0-7695-2085-5.

[76] Faraydon Karim, Anh Nguyen, and Sujit Dey. An interconnect architecture for networking systems on chips. *IEEE Micro*, 22(5):36–45, 2002. ISSN 0272–1732. DOI 10.1109/MM.2002.1044298.

[77] Takayoshi Katahira, Scanlan, Jong Wook Park, and Kwang Seok Oh. 0.3mm pitch CSP/BGA development for mobile terminals. In *Proceedings of the International Symposium on Microelectronics*, pages 393–401, November 2007. ISBN 0-930815-82-3.

[78] Nikolay Kavaldjiev. *A run-time reconfigurable Network-on-Chip for streaming DSP applications*. PhD thesis, University of Twente, Enschede, The Netherlands, January 2007.

[79] Parviz Kermani and Leonard Kleinrock. Virtual cut-through: a new computer communication switching technique. *Computer Networks*, 3:267–286, 1979.

[80] John Kim, William James Dally, and Dennis Abts. Flattened butterfly: a cost-efficient topology for high-radix networks. *SIGARCH Comput. Archit. News*, 35(2):126–137, 2007. ISSN 0163-5964. DOI 10.1145/1273440.1250679.

[81] Nam Sung Kim, Todd Austin, David Blaauw, Trevor Mudge, Krisztián Flautner, Jie S. Hu, Mary Jane Irwin, Kandemi, and Vijaykrishnan Narayanan. Leakage current: Moore's law meets static power. *IEEE Computer*, 36(12):68–75, December 2003. DOI 10.1109/MC.2003.1250885.

[82] Uming Ko and P.T. Balsara. Short-circuit power driven gate sizing technique for reducing power dissipation. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 3(3):450–455, September 1995. ISSN 1063-8210. DOI 10.1109/92.407004.

[83] Tim Kogel, Malte Doerper, Andreas Wieferink, Rainer Leupers, Gerd Ascheid, Heinrich Meyr, and Serge Goossens. A modular simulation framework for architectural exploration of on-chip interconnection networks. In *Proceedings of the International Conference on Hardware/Software Codesign and System Synthesis*, pages 7–12, New York, New York, 2003. ACM Press. ISBN 1-58113-742-7. DOI 10.1145/944645.944648.

[84] Akash Kumar, Andreas Hansson, Jos Huisken, and Henk Corporaal. An FPGA design flow for reconfigurable network-based multi-processor systems on chip. In *Proceedings of the Conference on Design, Automation and Test in Europe*, pages 117–122, Washington, Washington, D.C., 2007. IEEE Computer Society.

[85] Kangmin Lee, Se-Joong Lee, and Hoi-Jun Yoo. Low-power network-on-chip for high-performance SoC design. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 14(2):148–160, February 2006. ISSN 1063-8210. DOI 10.1109/TVLSI.2005.863753.

[86] A. Leroy, P. Marchal, A Shickova, F. Catthoor, F Robert, and Dideriek Verkest. Spatial division multiplexing: a novel approach for guaranteed throughput on NoCs. In *Proceedings of the International Conference on Hardware/Software Codesign and System*

Synthesis, pages 81–86, New York, New York, September 2005. ACM Press. ISBN 1-59593-161-9. DOI 10.1145/1084834.1084858.

[87] Daniel H. Linder and Jim C. Harden. An adaptive and fault tolerant wormhole routing strategy for k-ary n-cubes. *IEEE Transactions on Computers*, 40(1):2–12, January 1991.

[88] Magma. Quickcap datasheet, 2007. URL `http://www.magma-da.com`.

[89] Théodore Marescaux, Jean-Yves Mignolet, Andrei Bartic, Moffa, Dideriek Verkest, Serge Vernalde, and Rudy Lauwereins. Networks on chip as hardware components of an OS for reconfigurabl systems. In *Proceedings of the International Conference on Field Programmable Logic and Applications*, volume 2778/2003 of *Lecture Notes in Computer Science*, pages 595–605, Piscataway, New Jersey, 2003. IEEE Circuits and Systems Society. DOI 10.1007/b12007.

[90] McKeown. iSLIP: A scheduling algorithm for input-queued switches. *IEEE Transactions on Networking*, 7(2), April 1999.

[91] Eisse Mensink, Daniël Schinkel, Eric A. M. Klumperink, Ed J.M. van Tuijl, and Bram Nauta. A 0.28pJ/b 2Gb/s/ch transceiver in 90nm CMOS for 10mm on-chip interconnects. In *Proceedings of the IEEE International Solid-State Circuits Conference*, pages 414–415, February 2007. ISBN Softbound 1-4244-0852-0, CD-ROM 1-4244-0853-9. Digest of Technical Papers.

[92] Eisse Mensink, Daniël Schinkel, Eric A. M. Klumperink, Ed J.M. van Tuijl, and Bram Nauta. Optimal positions of twists in global on-chip differential interconnects. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 15(4):438–446, April 2007. ISSN 1063-8210. DOI 10.1109/TVLSI.2007.893661.

[93] Albert Molderink. Feasibility analysis of MPEG decoding on reconfigurable hardware. Master's thesis, University of Twente, April 2007.

[94] Fernando Moraes, Ney Calazans, Aline Mello, Leandro Möller, and Luciano Ost. HERMES: an infrastructure for low area overhead packet-switching networks on chip. *The VLSI Journal of Integration*, 38(1):69–93, 2004. ISSN 0167-9260.

[95] Morgenshtein, I. Cidon, A. Kolodny, and R. Ginosar. Comparative analysis of serial vs. parallel links in networks on chip. In *Proceedings of the International Symposium on System-on-Chip*, Los Alamitos, California, November 2004. IEEE Computer Society Press. ISBN 0-7803-8558-6.

[96] Robert D. Mullins. Minimising dynamic power consumption in on-chip networks. In J. Nurmi and J. Takala, editors, *Proceedings of the International Symposium on System-on-Chip*, pages 1–4, Los Alamitos, California, November 2006. IEEE Computer Society Press. ISBN 1-4244-0621-8. DOI 10.1109/ISSOC.2006.321982.

[97] Robert D. Mullins, Andrew West, and Simon W. Moore. Low-latency virtual-channel routers for on-chip networks. In *Proceedings of the International Symposium on Computer Architecture*, page 188, Washington, DC, USA, 2004. IEEE Computer Society. ISBN 0-7695-2143-6.

[98] Robert D. Mullins, Andrew West, and Simon W. Moore. The design and implementation of a low-latency on-chip network. In *Proceedings of the Conference on Asia South Pacific Design Automation*, pages 164–169, Piscataway, New Jersey, 2006. IEEE Press. ISBN 0-7803-9451-8. DOI 10.1145/1118299.1118348.

[99] Njuguna Njoroge, Jared Casper, Sewook Wee, Yuriy Teslyar, Daxia Ge, Christos Kozyrakis, and Kunle Olukotun. ATLAS: A chip-multiprocessor with transactional memory support. In *Proceedings of the Conference on Design, Automation and Test in Europe*, pages 3–8, Washington, Washington, D.C., 2007. IEEE Computer Society.

[100] Antonio Núñez. Advances in video coding for hand-held device implementation in networked electronic media. *Journal of Real-Time Image Processing*, pages 9–23, September 2006. ISSN 1861–8200. DOI 10.1007/s11554-006-0010-0.

[101] OPNET. OPNET modeler, 2008. URL `http://www.opnet.com`.

[102] Open SystemC Iniative (OSCI). Systemc documentation, 2004. URL `http://www.systemc.org`.

[103] M. Pedram and A. Abdollahi. Low-power RT-level synthesis techniques: a tutorial. *IEE Proceedings - Computers and Digital Techniques*, 152(3):333–343, May 2005. ISSN 1350-2387. DOI 10.1049/ip-cdt:20045111.

[104] R. Piloty and D. Borrione. The Conlan project: Status and future plans. In *Proceedings of the Design Automation Conference*, pages 202–212, New York, New York, June 1982. ACM. ISBN 0-89791-020-6.

[105] Gerard K. Rauwerda. *Multi-Standard Adaptive Wireless Communication Receivers*. PhD thesis, University of Twente, Enschede, The Netherlands, January 2008.

[106] Gerard K. Rauwerda. Discussion / personal communication about dab reference receiver implementation, 2008. Recore Systems.

[107] Jennifer Rexford and Kang G. Shin. Support for multiple classes of traffic in multicomputer routers. In *Parallel Computer Routing and Communication Workshop*, volume 853 of *Lecture Notes in Computer Science*, pages 116–130, Berlin, Germany, May 1994. Springer-Verlag.

[108] Edwin Rijpkema, Kees G.W. Goossens, Andrei Rădulescu, John Dielissen, Jeff van Meerbergen, P. Wielage, and E. Waterlander. Trade offs in the design of a router with both guaranteed and best-effort services for networks on chip. In *Proceedings of the Conference on Design, Automation and Test in Europe*, pages 350–355, Washington, Washington, D.C., March 2003. IEEE Computer Society.

[109] Arnaud Rivaton, Jérôme Quevremont, Qiwei Zhang, Pascal T. Wolkotte, and Gerard J. M. Smit. Implementing non power-of-two FFTs on coarse-grain reconfigurable architectures. In J. Nurmi, J. Takala, and T. D. Hamalainen, editors, *Proceedings of the International Symposium on System-on-Chip*, pages 82–85, Los Alamitos, California, November 2005. IEEE Computer Society Press. ISBN 0-7803-9294-9.

[110] Kaushik Roy, Saibal Mukhopadhyay, and Hamid Mahmoodi-Maimand. Leakage current mechanisms and leakage reduction techniques in deep-submicromete CMOS circuits. *Proceedings of the IEEE*, 91(2):305–327, February 2003. DOI 10.1109/JPROC.2002.808156.

[111] Andrei Rădulescu, John Dielissen, Santiago González Pestana, Om Prakash Gangwal, Edwin Rijpkema, P. Wielage, and Kees Goossens. An efficient on-chip network interface offering guaranteed services, shared-memory abstraction, and flexible network programming. *IEEE Transactions on CAD of Integrated Circuits and Systems*, 24(1):4–17, January 2005. DOI 10.1109/TCAD.2004.839493.

[112] Andrea Russo. Digital radio mondiale examples, 2007. URL `http://www.andrearusso.it/DRMExamples.html`. Used BBCWS7465.zip.

[113] M. Sarlotte, B. Candaele, Jérôme Quevremont, and D. Merel. Embedded software in digital AM-FM chipset. In *Proceedings of the Conference on Design, Automation and Test in Europe*, pages 340–341, Washington, Washington, D.C., March 2003. IEEE Computer Society. DOI 10.1109/DATE.2003.1253631.

[114] Daniël Schinkel, Eisse Mensink, Eric A. M. Klumperink, Ed J.M. van Tuijl, and Bram Nauta. A 3Gb/s/ch transceiver for 10-mm uninterrupted RC-limited global on-chip interconnects. *IEEE Journal of Solid State Circuits*, 41(1):297–306, January 2006. ISSN 0018–9200.

[115] Roel Schiphorst. *Software-Defined Radio for Wireless Local-Area Networks*. PhD thesis, University of Twente, Enschede, The Netherlands, September 2004.

[116] Rainer Schoenen. Weighted arbitration algorithms with priorities for input-queued switches with 100% throughput. In *IEEE International Workshop on Broadband Switching Systems*, May 1999.

[117] Rainer Schoenen. An architecture supporting quality-of-service in virtual-output-queued switches. *IEICE Transactions on Communication*, E83-B(2):171–181, February 2000.

[118] M. Sgroi, M. Sheets, A. Mihal, K. Keutzer, S. Malik, Jan Rabaey, and A. Sangiovanni-Vencentelli. Addressing the system-on-a-chip interconnect woes through communication-based design. In *Proceedings of the Design Automation Conference*, pages 667–672, New York, New York, 2001. ACM. ISBN 1-58113-297-2. DOI 10.1145/378239.379045.

[119] G.H. Shahidi. Soi technology for the ghz era. *IBM Journal of Research and Development*, 46(2/3), 2002.

[120] Smart Chips for Smart Surroundings. Smart chips for smart surroundings, annex I - description of work, November 2003.

[121] Smart Chips for Smart Surroundings. 4S project, 2008. URL `http://www.smart-chips.org`.

[122] Gerard J. M. Smit, Paul M. Heysters, and Egbert Molenkamp. The chameleon project in retrospective. In *Proceedings of the PROGRESS Symposium on Embedded Systems*, pages 181–184, Utrecht, The Netherlands, October 2004. STW.

[123] Gerard J. M. Smit, Eberhard Schüler, Jürgen E. Becker, Jérôme Quevremont, and Werner Brügger. Overview of the 4S project. In J. Nurmi, J. Takala, and T. D. Hamalainen, editors, *Proceedings of the International Symposium on System-on-Chip*, pages 70–73, Los Alamitos, California, November 2005. IEEE Computer Society Press. ISBN 0-7803-9294-9.

[124] Lodewijk T. Smit. *Energy-Efficient Wireless Communcation*. PhD thesis, University of Twente, Enschede, The Netherlands, January 2004.

[125] Lodewijk T. Smit, Gerard K. Rauwerda, Albert Molderink, Pascal T. Wolkotte, and Gerard J. M. Smit. Implementation of a 2-D 8x8 IDCT on the reconfigurable montium core. In *Proceedings of the International Conference on Field Programmable Logic and Applications*, pages 562–566, Piscataway, New Jersey, August 2007. IEEE Circuits and Systems Society. ISBN 1-4244-1060-6.

[126] Vassos Soteriou, Noel Eisley, Hang-Sheng Wang, Bin Li, and Li-Shiuan Peh. Polaris: A system-level roadmap for on-chip interconnection networks. In *Proceedings of the International Conference on Computer Design*, Washington, DC, USA, October 2006. IEEE Computer Society.

[127] Stergios Stergiou, Federico Angiolini, Salvatore M. Carta, Luigi Raffo, Davide Bertozi, and Giovanni De Micheli. ×pipes lite: A synthesis oriented design library for networks on chips. In *Proceedings of the Conference on Design, Automation and Test in Europe*, pages 1188–1193, Washington, Washington, D.C., March 2005. IEEE Computer Society. ISBN 0-7695-2288-2.

[128] Synopsys, 2008. URL `http://www.synopsys.com`.

[129] Cadence Design Systems, 2008. URL `http://www.cadence.com`.

[130] Marcel D. van de Burgwal, Gerard J. M. Smit, Gerard K. Rauwerda, and Paul M. Heysters. Hydra: an energy-efficient and reconfigurable network interface. In *Proceedings of the International Conference on Engineering of Reconfigurable Systems & Algorithms*, pages 171–177, Las Vegas, Nevada, 2006. CSREA Press. ISBN 1-60132-011-6.

[131] H.J.M. Veendrick. Short-circuit dissipation of static cmos circuitry and its impact on the design of buffer circuits. *IEEE Journal of Solid-State Circuits*, 19(4):468–473, August 1984.

[132] Elliot Waingold, Michael Taylor, Devabhaktuni Srikrishna, Vivek Sarkar, Walter Lee, Victor Lee, Jang Kim, Matthew Frank, Peter Finch, Rajeev Barua, Jonathan Babb, Saman Amarasinghe, , and Anant Agarwal. Baring it all to software: Raw machines. *IEEE Computer*, pages 86–93, September 1997.

[133] Hang-Sheng Wang, Xinping Zhu, Li-Shiuan Peh, and S. Malik. Orion: a power-performance simulator for interconnection networks. In *Proceedings of the Annual IEEE/ACM International Symposium on Microarchitecture*, pages 294–305, November 2002. DOI 10.1109/MICRO.2002.1176258.

[134] Hang-Sheng Wang, Li-Shiuan Peh, and S. Malik. A power model for routers: modeling Alpha 21364 and InfiniBand routers. *IEEE Micro*, 23(1):26–35, Jan.-Feb. 2003. DOI 10.1109/MM.2003.1179895.

[135] Maarten Wiggers, Marco Bekooij, Pierre G. Jansen, and Gerard J. M. Smit. Buffer capacities for multi-rate real-time systems with backpressure. In *Proceedings of the International Conference on Hardware/Software Codesign and System Synthesis*, pages 10–15, New York, New York, November 2006. ACM Press. ISBN 1-59593-370-0.

[136] Daniel Wiklund. *Development and Performance Evaluation of Networks on Chip*. PhD thesis, Linköping University, April 2005.

[137] Daniel Wiklund and Dake Liu. SoCBUS: Switched network on chip for hard real time systems. In *Proceedings of the IEEE International Parallel and Distributed Processing Symposium*, Los Alamitos, California, April 2003. IEEE Computer Society.

[138] Jinwen Xi and Peixin Zhong. A transaction-level NoC simulation platform with architecture-level dynamic and leakage energy models. In *Proceedings of the ACM Great Lakes symposium on VLSI*, pages 341–344, New York, New York, 2006. ACM Press. ISBN 1-59593-347-6. DOI 10.1145/1127908.1127986.

[139] Xilinx, 2008. URL `http://www.xilinx.com`.

[140] Bo Zhai, David Blaauw, Dennis Sylvester, and Krisztián Flautner. Theoretical and practical limits of dynamic voltage scaling. In *Proceedings of the Design Automation Conference*, pages 868–873, New York, New York, 2004. ACM. ISBN 1-58113-828-8.

List of Publications

[PW1] Arnab Banerjee, Pascal T. Wolkotte, Robert D. Mullins, Simon W. Moore, and Gerard J. M. Smit. An energy and performance exploration of network-on-chip architectures. *To appear in IEEE Transactions on Very Large Scale Integration (VLSI) Systems Special Section on Networks-on-Chip*, 2009.

[PW2] Pascal T. Wolkotte, Jochem H. Rutgers, Philip K. F. Hölzenspies, Mark Westmijze, Remco Blumink, and Gerard J. M. Smit. An automated design flow for FPGA-based sequential simulation. In *Proceedings of the Annual Workshop on Circuits, Systems and Signal Processing (ProRISC)*, pages 126–132. STW, Utrecht, The Netherlands, November 2008. ISBN 978-90-73461-56-7.

[PW3] Gerard J. M. Smit, André B. J. Kokkeler, Pascal T. Wolkotte, and Marcel D. van de Burgwal. Multi-core architectures and streaming applications. In I. Mandoiu and A. Kennings, editors, *Proceedings of the International Workshop on System-Level Interconnect Prediction*, pages 35–42. ACM, New York, April 2008.

[PW4] Lodewijk T. Smit, Gerard K. Rauwerda, Albert Molderink, Pascal T. Wolkotte, and Gerard J. M. Smit. Implementation of a 2-D 8x8 IDCT on the reconfigurable montium core. In *Proceedings of the International Conference on Field Programmable Logic and Applications*, pages 562–566. IEEE Circuits and Systems Society, Piscataway, New Jersey, August 2007. ISBN 1-4244-1060-6.

[PW5] Pascal T. Wolkotte, Philip K. F. Hölzenspies, and Gerard J. M. Smit. Fast, accurate and detailed NoC simulations. In P. Kellenberger, editor, *Proceedings of the ACM/IEEE International Symposium on Networks-on-Chip*, pages 323–332. IEEE Computer Society Press, Los Alamitos, California, May 2007. ISBN 0-7695-2773-6.

[PW6] Pascal T. Wolkotte, Philip K. F. Hölzenspies, and Gerard J. M. Smit. Using an FPGA for fast bit accurate SoC simulation. In *Proceedings of the Proceedings of the IEEE International Parallel and Distributed Processing Symposium, Reconfigurable Architecture Workshop*, page 167. IEEE Computer Society, Los Alamitos, California, March 2007. ISBN 1-4244-0909-8.

[PW7] Gerard J. M. Smit, André B. J. Kokkeler, Pascal T. Wolkotte, Philip K. F. Hölzenspies, Marcel D. van de Burgwal, and Paul M. Heysters. The chameleon architecture for streaming DSP applications. *EURASIP Journal on Embedded Systems*, 2007:Article ID 78082, 10 pages, 2007. ISSN 1687-3955.

[PW8] Pascal T. Wolkotte, Marcel D. van de Burgwal, and Gerard J. M. Smit. Non-power-of-two FFTs: Exploring the flexibility of the MONTIUM. In J. Nurmi and J. Takala,

editors, *Proceedings of the International Symposium on System-on-Chip*, pages 167–170. IEEE Computer Society Press, Los Alamitos, California, November 2006. ISBN 1-4244-0621-8.

[PW9] Gerard J. M. Smit, André B. J. Kokkeler, Pascal T. Wolkotte, Marcel D. van de Burgwal, and Paul M. Heysters. Efficient architectures for streaming applications. In Peter M. Athanas, Jürgen E. Becker, Gordon Brebner, and Jürgen Teich, editors, *Proceedings of the Dynamically Reconfigurable Architectures*, volume 6141 of *Dagstuhl Seminar Proceedings*, pages 1–7. Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Germany, October 2006. ISSN 1862-4405.

[PW10] Tjerk Bijlsma, Pascal T. Wolkotte, and Gerard J. M. Smit. An optimal architecture for a DDC. In *Proceedings of the Proceedings of the IEEE International Parallel and Distributed Processing Symposium, Reconfigurable Architecture Workshop*, pages 192–200. IEEE Computer Society, Los Alamitos, California, April 2006. ISBN 1-4244-0054-6.

[PW11] Nikolay Kavaldjiev, Gerard J. M. Smit, Pierre G. Jansen, and Pascal T. Wolkotte. A virtual channel network-on-chip for GT and BE traffic. In *Proceedings of the Annual Symposium on Emerging VLSI Technologies and Architectures*, pages 211–216. IEEE Computer Society, Washington, DC, USA, March 2006. ISBN 0-7695-2533-4.

[PW12] Nikolay Kavaldjiev, Gerard J. M. Smit, Pascal T. Wolkotte, and Pierre G. Jansen. Providing QoS guarantees in a NoC by virtual channel reservation. In K. Bertels, J. M. P. Cardoso, and S. Vassiliadis, editors, *Proceedings of the International Workshop on Applied Reconfigurable Computing*, volume 3985 of *Lecture Notes in Computer Science*, pages 299–310. Springer-Verlag, Heidelberg, Germany, March 2006. ISBN 3-5403-6708-X.

[PW13] Arnaud Rivaton, Jérôme Quevremont, Qiwei Zhang, Pascal T. Wolkotte, and Gerard J. M. Smit. Implementing non power-of-two FFTs on coarse-grain reconfigurable architectures. In J. Nurmi, J. Takala, and T. D. Hamalainen, editors, *Proceedings of the International Symposium on System-on-Chip*, pages 82–85. IEEE Computer Society Press, Los Alamitos, California, November 2005. ISBN 0-7803-9294-9.

[PW14] Pascal T. Wolkotte, Gerard J. M. Smit, Nikolay Kavaldjiev, Jürgen E. Becker, and Jens Becker. Energy model of networks-on-chip and a bus. In J. Nurmi, J. Takala, and T. D. Hamalainen, editors, *Proceedings of the International Symposium on System-on-Chip*, pages 82–85. IEEE Computer Society Press, Los Alamitos, California, November 2005. ISBN 0-7803-9294-9.

[PW15] Pascal T. Wolkotte, Gerard J. M. Smit, and Jens Becker. Energy-efficient NoC for best-effort communication. In T. Rissa, S. Wilton, and P. Leong, editors, *Proceedings of the International Conference on Field Programmable Logic and Applications*, pages 197–202. IEEE Circuits and Systems Society, Piscataway, New Jersey, August 2005. ISBN 0-7803-9362-7.

[PW16] Pascal T. Wolkotte, Gerard J. M. Smit, Gerard K. Rauwerda, and Lodewijk T. Smit. An energy-efficient reconfigurable circuit-switched network-on-chip. In *Proceedings of the Proceedings of the IEEE International Parallel and Distributed Processing*

Symposium, Reconfigurable Architecture Workshop, page 155. IEEE Computer Society, Los Alamitos, California, April 2005. ISBN 0-7695-2312-9. ISSN 1530-2075.

[PW17] Lodewijk T. Smit, Gerard J. M. Smit, Johan L. Hurink, Hajo Broersma, Daniel Paulusma, and Pascal T. Wolkotte. Run-time mapping of applications to a heterogeneous reconfigurable tiled system on chip architecture. In *Proceedings of the International Conference on Field Programmable Technology*, pages 421–424. IEEE, Piscataway, NJ, USA, December 2004. ISBN 0-7803-8651-5.

[PW18] Pascal T. Wolkotte, Gerard J. M. Smit, and Lodewijk T. Smit. Complexity analysis for mapping a DRM receiver on a heterogeneous tiled architecture. In *Proceedings of the Annual Workshop on Circuits, Systems and Signal Processing (ProRISC)*, pages 442–446. STW, Utrecht, The Netherlands, November 2004. ISBN 90-73461-43-X.

[PW19] Lodewijk T. Smit, Gerard J. M. Smit, Johan L. Hurink, Hajo Broersma, Daniel Paulusma, and Pascal T. Wolkotte. Run-time assignment of tasks to multiple heterogeneous processors. In *Proceedings of the PROGRESS Symposium on Embedded Systems*, pages 185–192. STW, Utrecht, The Netherlands, October 2004.

[PW20] Pascal T. Wolkotte, Gerard J. M. Smit, and Lodewijk T. Smit. Partitioning of a DRM receiver. In *Proceedings of the International OFDM-Workshop*, pages 299–304. Technical University of Hamburg, Harburg, Germany, September 2004.

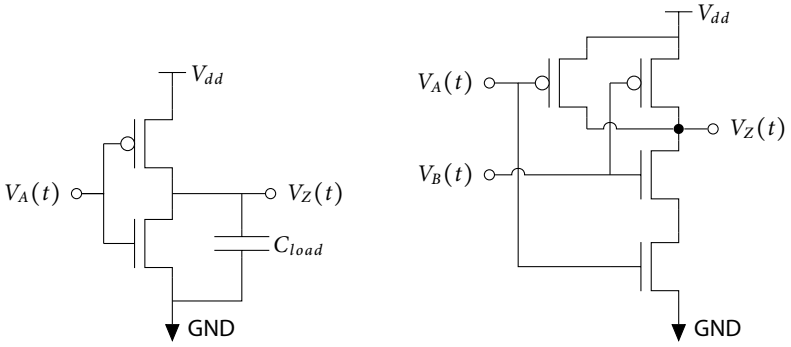
Appendix A

CMOS Power Consumption

Abstract – *This chapter gives a short introduction in the theory of power consumption of IC-designs. We will give the theoretical background of both dynamic (P_{dyn}) and static power (P_{stat}) consumption. Furthermore, we give a set of possible power reduction techniques that can be used. For example, clock-gating, dual threshold, and signal gating.*

The technology of semiconductor devices is a continuous development. It started with the first working bipolar transistor realized at Bell Laboratories in 1947 by William Shockley, John Bardeen and Walter Brattain. Jack Kilby, working at Texas Instruments, was the first to combine transistors into an IC in 1958. Gradually more transistors were combined into an IC resulting in the present VLSI designs consisting of billions of transistors.

Besides the increasing number of transistors on an IC, the type of transistors used changed. From the early bipolar transistor, to Metal Oxide Semiconductor Field Effect Transistor (MOSFET) with initially logic realized in p-channel MOS, later in n-channel MOS, and currently with CMOS technology [72]. The major advantage of CMOS technology was and still is it relatively low static power consumption compared to other technologies in the 1980s. The last two decades it benefited from its shrinking technology size that reduced its silicon area and worst-case gate delay. In this chapter we therefore only focus on the power consumption in CMOS technology.



(a) Inverter and its capacitive load

(b) NAND gate

Figure A.1 – Schematic of CMOS gates

A.1 Power Components

The power consumption in a CMOS circuit design can be divided into two major components. The dynamic power consumption of a CMOS circuit is the energy that is consumed by the components when they are active, i.e. their inputs change. This energy is required to charge (and discharge) the capacitors in the circuitry. The total aggregated effective capacitance is a sum of the wires and gates that are driven by the output of a component. The second part of the dynamic power consumption is caused by the short circuit path between the supply and ground rails of a component when both NMOS and PMOS transistors are conducting. The second major component of the power consumption is the static power consumption. This is the constant leakage of the circuit due to non-ideal diodes and transistors. Each of the power consumption parts is explained using a simple inverter as an example. Figure A.1(a) depicts a CMOS inverter and a NAND gate is depicted in figure A.1(b).

A.1.1 Dynamic power

When $V_{in} = 0V$, the NMOS transistor is in cut-off and the PMOS transistor is conducting. The output load will be charged up to V_{dd}. When V_{in} equals V_{dd} the PMOS is in cut-off and the output load will be discharged down to zero volts. In either case no current flows between the supply rails.

Charging the output load of an inverter or any other gate requires energy. This energy is dissipated by the resistance of the wires, but the amount energy that is dissipated is determined by the capacity that has to charged and discharged. This can be described by:

$$P_{switch} = \alpha f C_{eff} V^2 \qquad [\mathrm{W}] \tag{A.1}$$

where α is the activity, f is the frequency on which the circuitry operates, and V is the supply voltage of the circuitry that has an effective capacitance of C_{eff}. This capacitance includes the load of the connected wire (C_{load})as depicted in figure A.1(a), but also includes the internal capacitances of the inverter and input load of the successive gate that have to be charged and discharged. Although the complete circuitry has a frequency of f, the individual gates might have a reduced number of transitions. This relative reduction can be corrected via the scaling factor α.

Besides the charging of capacitors in the designs an extra amount of energy is consumed during an input transition. Due to a gradual change of the input V_{in} from V_{gnd} to V_{dd} and the capacitive load on its output, the NMOS transistor goes from the cut-off region ($V_{in} < V_{th}$) via saturation to the triode region and the PMOS transistor vice versa. If neither NMOS or PMOS transistor is in its cut-off region a current can flow between the supply and ground rails. This short-circuit dissipation can be modelled and varies with the output load and input signal slope [131]. In a well balanced design with matched rise and fall times of input and output, the short-circuit dissipation will be less than 20% of the total dynamic power consumption. For an unloaded symmetrical inverter the short circuit power can be described by:

$$P_{SC} = \frac{\beta}{12} \left(V_{DD} - 2V_{th} \right)^3 \tau f \qquad \text{[W]} \tag{A.2}$$

where β describes the gain factor of a MOS transistor and τ the rise and fall times of the input. The short-circuit dissipation decreases when the inverter's output load is increased [82]. This is caused by a better match between the rise and fall times of input and output of the inverter.

A.1.2 STATIC POWER

The second component of the CMOS circuit power consumption is independent of the circuit's activity. In theory the gates should not be conducting when the inputs and outputs are stable, because either the NMOS or PMOS transistors of a gate are in cut-off. However, a leakage current (I_{leak}) is present due to the non-ideal transistors. Multiplying this current with the supply voltage (V_{dd}) results in the static power consumption. This static power consumption can be described by five short-channel leakage mechanisms [110] and are depicted in figure A.2:

1. reverse-bias pn junction leakage, I_{pn}
2. subthreshold leakage, I_{sub}
3. gate leakage, I_{gate}, which can be split in a current due to gate-oxide tunnelling and due to hot-carrier injection
4. gate-induced drain leakage, I_{GIDL}
5. channel punchthrough current, I_{punch}

The first three mechanisms cause the largest fraction of the leakage current [1, 65, 81]. The reverse-bias pn junction leakage is a function of area and doping con-

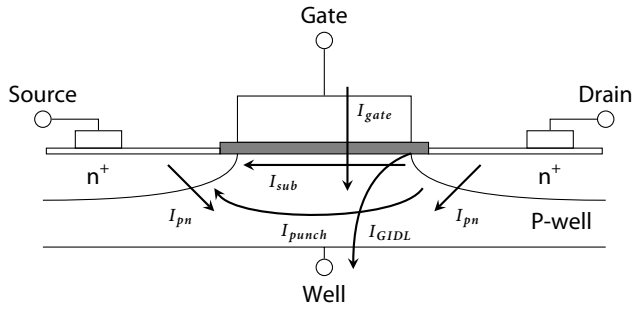


Figure A.2 – Leakage mechanisms in a short-channel transistor [110]

Table A.1 – Predicted development of leakage in cmos circuitry [65]

Generation	Year	I_{pn}	I_{sub}	I_{gate}
90 nm	2004	25 pA	804 pA	13 pA
50 nm	2010	3 nA	21 nA	52 nA
25 nm	2016	120 nA	260 nA	510 nA

centration. If both n and p regions are heavily doped the Band-to-Band Tunnelling (btbt) dominates the pn junction leakage [110].

For technologies above 100nm the I_{sub} is the main source of leakage. The I_{sub} is typically dominated by the weak inversion which can be described by [65]:

$$I_{sub} = KV_T^2 \frac{W}{L} e^{\frac{V_{GS}-Vth}{mV_T}} \left(1 - e^{-\frac{V_{DS}}{V_T}}\right) \tag{A.3}$$

where

$$m = 1 + \frac{C_{dm}}{Cox} \qquad \text{and} \qquad K = \mu_0 \sqrt{\frac{q\epsilon_{si}N_{ch}}{2\Phi_s}} \tag{A.4}$$

are technology parameters, W and L are the channel's width and length. V_{DS} is the drain source voltage V_{GS} is the gate source voltage and V_{th} is the threshold voltage of the transistor. $V_T = \frac{kT}{q}$ is the thermal voltage, which is approximately 25 mV at room temperature. Other effects can be included in the equation to estimate the subthreshold leakage more accurate like the drain-induced barrier lowering (dibl) and the body effect [110].

For smaller feature sizes the I_{gate} will surpass the I_{sub}. A prediction of the leakage in future generations cmos is given in table A.1

The increase of leakage current is caused by the quickly reduced oxide thickness (T_{ox}). The gate leakage is dominated by the gate-oxide tunnelling due to high electric fields that can cause direct tunnelling through the gate or Fowler-Nordheim tunnelling through the oxide bands. Kim et al. [81] reduce the gate current to a simplified formula that depends on the supply voltage (V_{DD}), gate width (W) and

T_{ox}.

$$I_{gate} = KW\left(\frac{V_{DD}}{T_{ox}}\right)^2 e^{-\frac{\alpha T_{ox}}{V_{DD}}} \tag{A.5}$$

where K and α are derived experimentally. From this equation it is clear that reduction of T_{ox} has an exponential effect on the gate leakage. Increasing T_{ox} is not an option, because with technology scaling the oxide thickness must scale proportionally to prevent short channel effects [81]. Usage of high-k dielectric gate insulators can reduce the gate leakage several orders of magnitude.

A.2 Power Reduction Techniques

As described in the previous section the power consumption of CMOS devices depends on a large number of parameters. Due to the continuous process of technology scaling all the parameters have changed and influenced the power consumption. Despite for example the reduced voltage the power increased due to the increasing amount of transistors (and functionality) and higher operating frequencies. Until the beginning of this century the major concern was the dynamic power consumption. For high performance circuits the static power consumption has grown exponentially and might become the dominant factor in a few years [73, 81].

The designer of an IC has many possibilities to reduce the power consumption of a specific IC [103]. Reducing the dynamic power consumption requires a reduction of one or more variables in eq. (A.1). The short circuit power will be optimized by the tooling, by choosing the right transistor size for the desired rise and fall times. Reducing the static power is currently focussing on eq. (A.3), but without paying attention to improved materials, eq. (A.5) will become equally or more important. A few of the optimization possibilities are described in the next paragraphs.

On the lowest level the power optimisation can be achieved by choosing the right technology. For each technology a large set of standard cell libraries are available that optimize for performance, area or power. For example, TSMC offers various libraries per technology size that vary the supply voltage, threshold voltage and that are optimized for lower power or high performance. More optimized libraries will appear. For example, IBM is advocating the silicon on insulator (SOI) technology that, compared to bulk CMOS technology, has a 25% to 30% speed improvement and 1.7 to 3 times lower power consumption [69, 119]. Including MOSFETs with high-k dielectric materials is, for example, researched by INTEL and included in their new 45 nm processor [26, 71].

During synthesis the designer has the possibility to incorporate other techniques to reduce the overall power consumption. Because various libraries are available per technology generation it is possible to select specific gates for parts of the design. This enables the tooling to choose slower (more power efficient) gates for slow parts of the design, for example transistors with a higher threshold voltage as can be seen from eq. (A.3). Reduction of the dynamic power consumption can be achieved by clock gating a synchronous design. In a typical synchronous design not all registers are updated every clock cycle. Disabling the clock tree of those registers in those

cycles has considerable influence on the dynamic power consumption as we have demonstrated in chapter 6. All these techniques require some work of the designer, but he is supported by tooling as for example described in section 6.1.1 where the "power-reduction-tooling" is completely integrated in the tool flow.

More adaptive techniques besides simple clock gating exist. The two most frequently proposed and used techniques are Dynamic Voltage Scaling (DVS) and Dynamic Frequency Scaling (DFS). Lowering the frequency will reduce the power consumption, but not the energy as a task requires more time to finish. Reducing energy consumption is only possible with reducing voltage, which is possible due to lower frequencies. Scaling the voltage is practically limited to approximately 30% of its maximum as has been derived by Zhai [140]. Lowering the voltage more will decrease dynamic power, but it might increase static power due to the lower threshold voltage that is required. Complete switching off, i.e. power gating, inactive parts will also reduce the static power consumption and can be very suitable for energy savings in multi-core architectures.

Combinations of both voltage and frequency scaling techniques has already been implemented in systems. It not only requires silicon that supports DVS and DFS, but high-level system information as well. Changing the performance of the circuit by adapting the voltage and frequency can have major influences, especially in real-time systems where tasks still have to finish before their deadlines. This requires a good interaction between monitoring systems, schedulers and the scaling hardware.

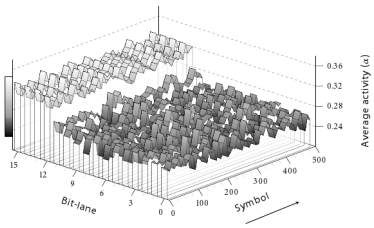
Appendix B

Toggle Rates

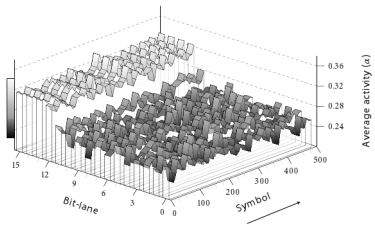
Abstract – *This appendix provides the detailed data activities versus time and bit lane of three streaming applications. Per application a selection of the channels between processes are monitored. Per communicated message on these channels the data activity is determined. It is assumed that a message is preceded and ended with all zeros, and not interleaved with other messages. The resulting activity is plotted versus the time and bit lane.*

B.1 HiperLAN/II

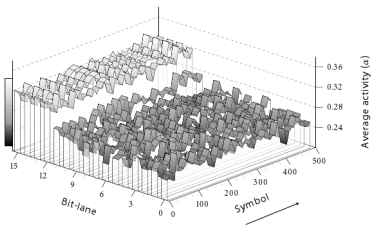
The messages used for the activity analysis of the HiperLAN/2 receiver are obtained from an implementation of earlier projects [105, 115]. We used an encoded stream that was transmitted through an AWGN channel model without multipath effects and received with SNR of 11 dB. Per channel between two processes the messages contained a number of items (as given by table 3.1(b)) and each item contained either a 32 bit complex sample or a single byte of raw date (ASCII text). The resulting α plots are given in figure B.1.



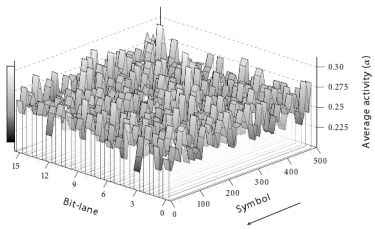
(a) ADC



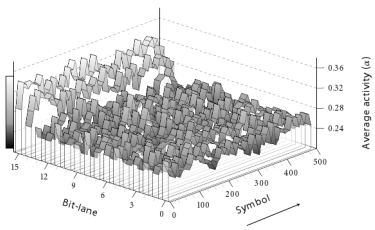
(b) Synchronization



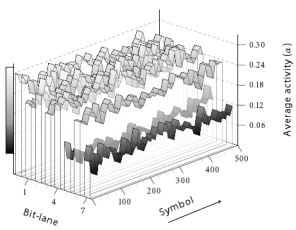
(c) Frequency correction



(d) FFT

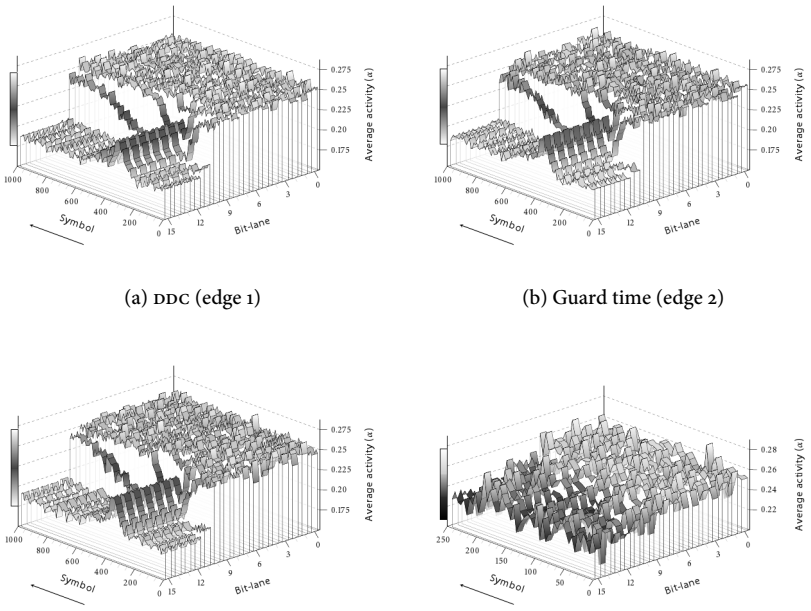


(e) Equalization



(f) ASCII text

FIGURE B.1 – Activity versus time and bit lanes in HiperLAN/2



(a) DDC (edge 1)

(b) Guard time (edge 2)

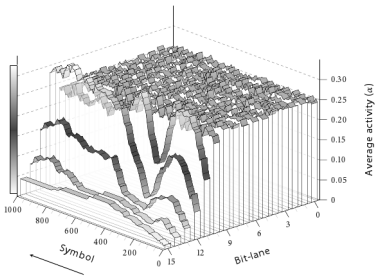
(c) Frequency correction (edge 3)

(d) FFT (edge 4)

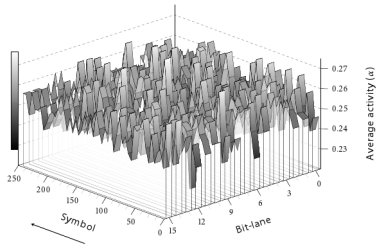
FIGURE B.2 – Activity versus time and bit lanes in DRM

B.2 Digital Radio Mondiale

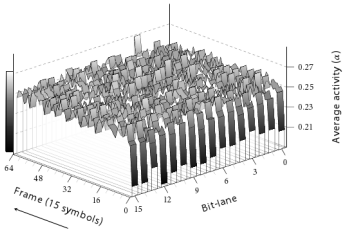
A full DRM implementation [45] is used to perform the activity analysis. We used a recorded transmission, which is sampled at 48 ksamples/sec, from the BBC World Service in mode-B with a SNR of 18–25 dB [112]. All complex samples are converted to 16 bit fixed-point and the decoded source bits, i.e. MSC data, are grouped in 32 bit words. The message contain either a single OFDM symbol or a full DRM frame (15 symbols) in case of the MSC data. The resulting α plots are given in figure B.2 on page 191 and 192.



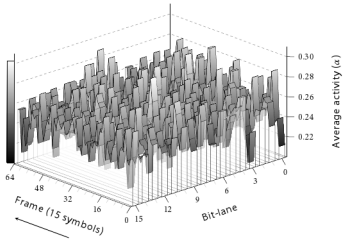
(a) Channel estimation (edge 5)



(b) Equalization (edge 6)

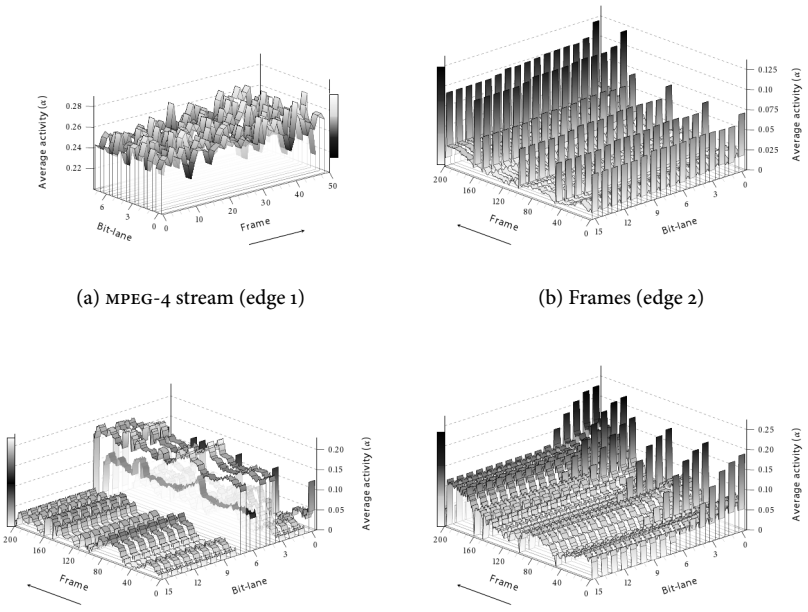


(c) MSC -QAM (edge 9)



(d) MSC hardbits (edge 10)

Figure B.2 – Continued



(a) MPEG-4 stream (edge 1)

(b) Frames (edge 2)

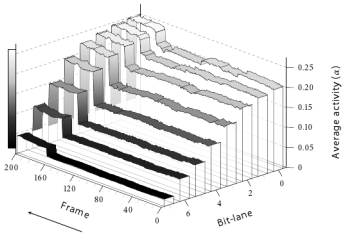
(c) Motion vector (edge 4)

(d) Frames (edge 5)

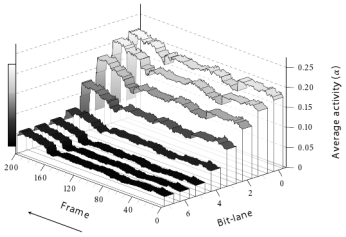
FIGURE B.3 – Activity versus time and bit lanes in MPEG-4

B.3 MPEG-4

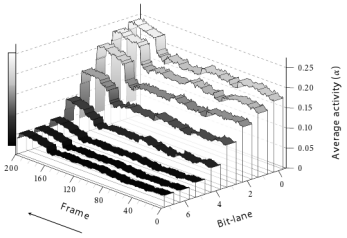
For the activity analysis of the MPEG-4 application, we use the implementation for the QCIF format of the simple profile onto a heterogeneous architecture [93, 125]. A short animated motion picture of 202 frames is used to analyse the activity on the edges. This video fragment has a scene change between frame 162 and 163. It is assumed that a whole frame is communicated in a single message. All samples before the motion compensation are represented by 16 bits and afterwards by eight bits. The resulting α plots are given in figure B.3 on 193 and 194.



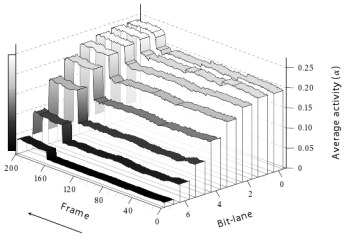
(e) Luminance (edge 6)



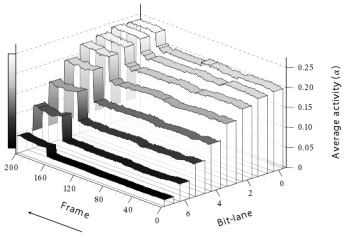
(f) Cr (edge 7)



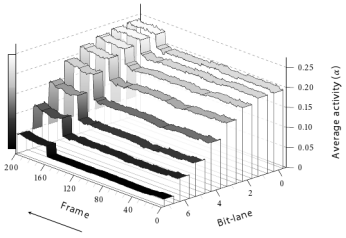
(g) Cb (edge 8)



(h) Red



(i) Green



(j) Blue

FIGURE B.3 – Continued

Appendix C

Interleaving of Packets

In this chapter we determine the probability of a bit flip (changes from zero to one and one to zero) in two interleaved streams. Stream A consists of a sequence of zero's and one's. The probability of value a at position i is given by $p_A(a_i|a_{i-1})$. This stream is merged with a second stream B of equal length. The individual items are interleaved with an interleave pattern of $\{a_i, b_i, a_{i+1}, b_{i+1}, \ldots\}$. We are interested in the possibility of a bit flip in the resulting stream C.

For the probability of the value x, either zero or one, we have the following relations per stream:

$$p(x) + p(\bar{x}) = 1 \tag{C.1}$$

$$p(\bar{x}|x) + p(x|x) = 1 \tag{C.2}$$

$$\begin{aligned} p(x) &= p(x|x) \cdot p(x) + p(x|\bar{x}) \cdot (1 - p(\bar{x})) \\ &= \frac{p(x|\bar{x})}{p(\bar{x}|x) + p(x|\bar{x})} \end{aligned} \tag{C.3}$$

$$E(\#x) = \frac{1}{p(x|\bar{x})} \tag{C.4}$$

where $E(\#x)$ is the expected number of successive identical bit values.

For both streams we define the expected number of successive identical bit values as:

$$z_A = E_A(\#0), \qquad o_A = E_A(\#1)$$
$$z_B = E_B(\#0), \qquad o_B = E_B(\#1)$$

The two streams are interleaved on a per flit bases as presented earlier. From those streams we pick a selections of length N, where N is the smallest number that

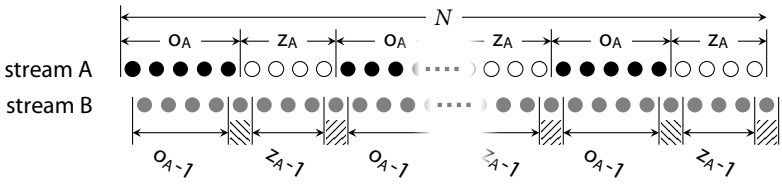


Figure C.1 – Merge of streams A and B

satisfies:

$$N = J_A \cdot (z_A + o_A) = J_B \cdot (z_B + o_B), \quad \text{with } N, J_A, J_B \in \mathbb{N} \tag{C.5}$$

such that the expected number of zero's and one's in the new stream equals:

$$z_C = \frac{z_A + z_B}{2} \qquad o_C = \frac{o_A + o_B}{2} \tag{C.6}$$

Stream A is ordered such that it consists of successive sequences of zero's and one's of length z_A and o_B. This stream is interleaved with an unordered stream B of length N as depicted in figure C.1

The $2 \cdot J_A$ bit flips that are present in stream A will be present in stream C too. These flips are located in the sequence of values $\{c_{2f}, c_{2f+1}, c_{2f+2}\}$, where $a_f \neq a_{f+1}$. The insertion of b_f does not increase nor decrease the number of flips compared to stream A, independent of the value of b_f. Those are indicated by the shaded areas in figure C.1. Furthermore, two extra bit flips are introduced for the all positions where $a_i \neq b_i$ and $a_i = a_{i+1}$. The possibility to introduce two of those bit flips caused by b_i depends on the length of successive identical values in stream A and the possibility of the opposite value in stream B.

The possibility for a bit flip in stream C equals:

$$\begin{aligned}
p_C(flip) &= \frac{E_C(\#flips)}{2 \cdot N} \\
&= \frac{J_A \cdot [2 + p_B(1) \cdot (z_A - 1) \cdot 2 + p_B(0) \cdot (o_A - 1) \cdot 2]}{J_A \cdot (z_A + o_A) + J_B \cdot (z_B + o_B)} \\
&= \frac{2 \cdot J_A \cdot \left[1 + \frac{o_B}{z_B + o_B} \cdot (z_A - 1) + \frac{z_B}{z_B + o_B} \cdot (o_A - 1)\right]}{(J_A + \frac{J_A}{J_B} \cdot J_B) \cdot (z_A + o_A)} \\
&= \frac{\frac{z_B + o_B + z_A \cdot o_B - o_B + o_A \cdot z_B - z_B}{z_B + o_B}}{z_A + o_A} \\
&= \frac{\frac{z_A \cdot o_B + o_A \cdot z_B}{\frac{J_A}{J_B} \cdot (z_A + o_A)}}{z_A + o_A} \\
&= \frac{z_A \cdot o_B + o_A \cdot z_B}{\frac{J_A}{J_B} \cdot (z_A + o_A)^2}
\end{aligned} \tag{C.7}$$

For random data the possibility of zero's and one's in a stream equals ½, i.e. $p_x(0)$=½ and $p_x(0|1) = p_x(1|0)$. This makes $z_x = o_x$, which results in the possibility of a flip in the interleaved stream equals:

$$\begin{aligned} p_C(flip) &= \frac{o_B \cdot z_A + z_B \cdot o_A}{\frac{J_A}{J_B} \cdot (z_A + o_A)^2} \\ &= \frac{z_B \cdot z_A + z_B \cdot z_A}{(z_A + o_A) \cdot (z_B + o_B)} \\ &= \frac{2 \cdot z_B \cdot z_A}{4 \cdot z_A \cdot z_B} \\ &= \frac{1}{2} \end{aligned} \tag{C.8}$$

Appendix D

Automated Creation of the Sequential HIL Simulator

Abstract – *The required sequential HIL simulator, as described in chapter 8, is created manually. In this chapter we describe our ideas for the creation of the simulator via an automated design flow. This flow can support both homo- and heterogeneous multi-core architectures. The implementation of this design flow is work in progress, but we present some initial results.*

For a traditional HIL simulation of a design, described in any HDL, the design is analysed, synthesized to a specific FPGA technology and finally placed and routed onto the targeted FPGA. As described in section 8.2, we use the repetitive structure of many-core architectures. By mapping functionally identical parts of the design onto the same FPGA hardware, we reduce this resource requirement. Instead of the instantiation of the whole architecture in parallel in the FPGA, the individual cells are evaluated sequentially. However, this requires transformation of the original design description.

D.1 Design Flow

To automatically generate the FPGA based simulator we have to transform the designer's parallel architecture. We use the general term *cell* to denote any distinct design element, e.g. a single processor core or a router, in this architecture. We

Parts of this chapter have been presented at the 19th Annual Workshop on Circuits, Systems and Signal Processing (ProRISC 2008), Veldhoven, The Netherlands [PW2].

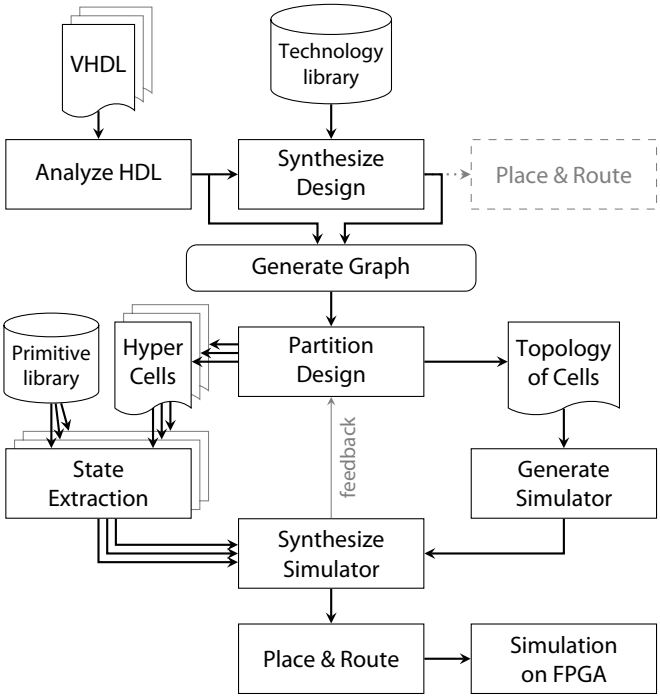


Figure D.1 – Simulator's Design Flow

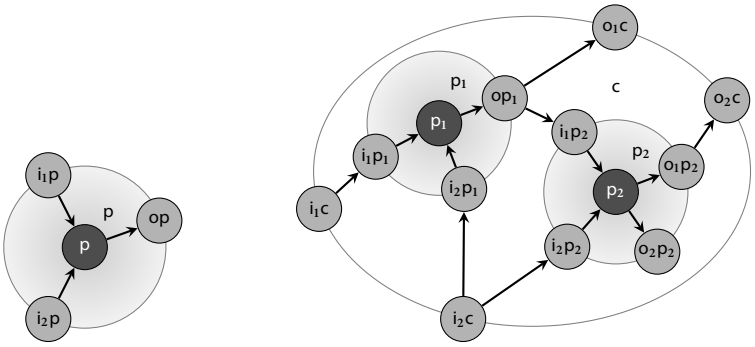
partition the design by identifying the various cells in the design. Hereby we assume that the cells are partitioned via a top-down approach, using the hierarchy of the parallel architecture. We assume each pair of cells to be highly similar or completely different. Therefore, we group cells that are highly similar and for each group of cells we create a single *hypercell*. Per hypercell, we *extract the state*, such that the combinational functionality can be re-used on a delta cycle basis within the FPGA. Small differences between cells within a group are preserved in the hypercell.

In the parallel architecture, the identified cells are interconnected by links. This interconnection of the individual cells is described by the architecture's *topology*. This topology is used to *generate* the simulator, which consists of the link memory, state memory and a central controller. The simulator and transformed hypercells are combined, synthesized, and placed and routed to the targeted FPGA.

Figure D.1 depicts the design flow of the creation of a cycle-accurate SHILS for an arbitrary hardware design. In the next sections we describe the individual parts of the design flow.

D.1.1 Netlist representation

In the intermediate descriptions during a normal synthesis (depicted in the top part of figure D.1, either before or after the technology mapping, the design is described as a *netlist*. This netlist describes the functionality of the design by an



(a) Primitive graph

(b) Netlist graph

Figure D.2 – Graphs representing hardware

interconnection of primitive components. For the sequential simulation, we use one of the two possible intermediate descriptions. We convert the netlist description into a graph, which is used by the tools to apply transformations, partitioning and other operations. After all operations the graph is converted back to a netlist, such that the design can be synthesized for the targeted FPGA.

Every hardware technology, ASIC and FPGA, has a library with standard cells or *primitives*. An AND, OR, LUT, and D-FF are examples of such primitives. Primitives can be instantiated. We represent each primitive by a small graph in which the primitive's function and each port is represented by a vertex. Each vertex of a primitive has a unique label.

The netlist is the hierarchical description of cells interconnected by wires. The instance of a primitive in a netlist is an example of a cell. Larger cells consist of multiple instantiated primitives and other cells. Each cell in the netlist is represented by a graph in which vertices can be uniquely identified by their label and their hierarchical position (denoted with a sequence of numbers) in the corresponding netlist.

Thus, the graph of an instance of a primitive extends its primitive's graph with a sequence of numbers, i.e. hierarchy annotation, on every vertex. The first number of this annotation corresponds to the cell number on the top level of the netlist, the second corresponds to the cell number within the top level cell, etc.

Figure D.2 visualizes a graph representing a primitive and depicts a netlist graph with two primitives. The conversion of a technology primitive into a graph results in multiple vertices. The graph representation of a netlist is therefore large. For example, a GuarVC router, such as used in this thesis, is built of about 15k Xilinx Virtex-II FPGA primitives. Its equivalent graph has 80k vertices and 114k edges.

D.1.2 PARTITIONING OF THE DESIGN

After the conversion to the graph representation we apply transformations to the graph such that a sequential simulator can be generated. The first step in this transformation is a partitioning of the graph. In the partitioning process, we have to identify the identical cells within the graph. As described in section 8.2, we sequentially simulate large cells of the design. The cell can be of any size, for example, a single processor, or the processor's ALU. Large cells update a larger part of the state vector at once. Small cells increase the utilization, due to the event driven simulation, but will impose more simulator overhead, due to the increase in inter-cell communication.

As all cells in a homogeneous many-core architecture are identical, we only have to instantiate a single core's combinational functionality, i.e. the largest replicated cell, in the FPGA. In heterogeneous multi-core designs, the individual cells are different. The difference can range from highly similar to completely different. For example, a NoC with routers that only have a different address describing their position in the topology or a SoC with various types of processing cores.

A simple approach is the instantiation of a transformed cell for each variation of a cell in a design and the selection of the right cell every delta-cycle. However, this might result in a large resource requirement and a low utilization of all cells available in parallel. Therefore, we group cells that are highly similar and for each group of cells we create a single *hypercell*. Differences between cells within a group are preserved in the hypercell. This is done by adding multiplexers that allow selection of the correct parts during simulation every delta cycle.

A group of cells is only merged into a single hypercell, if the overhead of the added multiplexers does not outweigh the resource reduction. This is done by instantiating the similar parts only once. This will reduce the resource requirements and increase the utilization of the simulator.

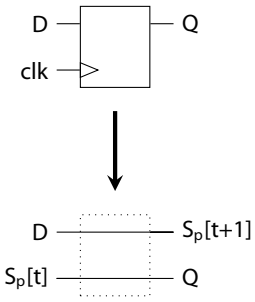
Each unique hypercell will be instantiated at least once in the generated simulator. When memory bandwidth and FPGA resources are still available, additional hypercells are instantiated to increase the performance of the simulator.

D.1.3 STATE EXTRACTION

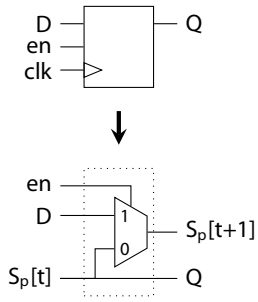
As depicted in figure 8.3(b), we separate the combinational functionality from the synchronous elements within the netlist graph. This extraction of the synchronous elements is performed on a per hypercell basis. Each instantiated synchronous primitive within a hypercell, e.g. D-flipflop, is replaced by its original ports, an old-state input port ($S_p[t]$), a new-state output port ($S_p[t+1]$), and some combinational logic. This combinational logic is required to selectively change the new state, such that the functionality of synchronous element is emulated. For example, a D-flipflop with enable only updates its state, if the enable port is active.

Using such control logic will enable the hypercell to swap states of multiple entities in the original design. That is, the state is not stored within the hypercell itself, but an entity's old state can be read by the hypercell and the updated state of

(a) Memory connections



(b) D-flipflop



(c) D-flipflop with enable

Figure D.3 – Examples of extraction of synchronous elements

this entity can be read from the hypercell.

All the state signals of a hypercell are grouped into two large state vectors that represent the old and new state of a hypercell. The two added state vectors are connected to data ports of the simulator's state memory. The state of the entity E, stored in the state memory, will be updated by:

1 offering two addresses (E_{addr} and E'_{addr}) to the state memory, which respectively read and write the entity's state, and

2 connect the state memory's data ports to the entity's corresponding hypercell.

Those addresses and interconnection are controlled by the simulator's scheduler.

If we assume only rising edge sensitive synchronous elements and a rising edge sensitive state memory, any D-flipflop can be replaced by the interconnection of the D-flipflop's ports with the added external state interface port as depicted in figure D.3(b). More advanced synchronous elements, e.g. D-flipflop with enable or reset, will have to be replaced by additional combinational functionality. An example is depicted in figure D.3(c).

This extraction is automated by transformation of the hypercell's graph. For all synchronous primitives in the technology library used, we have a corresponding combinational replacement primitive. The algorithm will search for all instantiations of the synchronous primitives (using the labelling function) and replace

the primitive's graph by its corresponding combinational replacement primitive. For the extraction we include a parameter for the edge sensitivity of an entity. The combinational replacement primitive is different for just rising edge sensitivity or both rising and falling edge sensitivity.

In the case study, presented in this thesis, we instantiate a single transformed hypercell on the FPGA and the whole state vector of a single cell is available in parallel via the internal RAMs. For every core not all elements of its state vector will be required, because only a part of the state vector is addressed or updated in a single clock cycle, due to the presence of for example register banks and memory blocks. By detecting the relationship between and grouping of individual synchronous elements the width of the state vector used per delta cycle is reduced. This will decrease the required memory bandwidth and makes the instantiation of large cores possible. For cores with a huge state vector, e.g. a local L1 cache, large parts of this vector can even be off-loaded to external memory, because per system cycle only a tiny fraction is addressed and required within the FPGA. Using multiple levels for the storage of the whole system's state will also require a study to the appropriate caching and scheduling mechanisms.

D.1.4 Generation of the Simulator

After the partitioning of the hardware design into multiple unique hypercells and a topology description of the original cells we are able to generate a simulator. The simulator consists of:

- one or multiple instantiations of each unique hypercell;
- a large state memory with both the current and new state of the system;
- a link memory that stores the most up-to-date value of the links between cells;
- two interconnection networks that respectively interconnect:
  - the state ports of the instantiated hypercells with the state memory,
  - the input and output link ports of the instantiated hypercells via the link memory;
- a scheduler and controller.

The scheduler determines, per delta cycle, which original cell's state is updated by a hypercell. The schedule is dynamically adjusted using the stable status of each cell in the current system cycle. The controller generates the right addresses for the state and link memories and controls the interconnection network, such that the determined schedule is executed.

In the current implementation the scheduler is a round-robin arbiter and each cell in unstable at the start of a system cycle. The number of delta cycles can be reduced by a more optimized triggering and scheduling of the evaluation of individual cores. This will reduce the average number of delta cycles per system cycle, which increases the overall performance. For example, parts of a many-core architecture can be in-active for a specific period, such that those cores are not

scheduled for evaluation in the respective system cycles. All these performance gains will rely on an efficient scheduling mechanism of the simulator. Extra help to find the optimal schedule could be found via a pre-analysis of the cores' relations in the parallel design, which is simulated.

The instantiated hypercells and state memory will be generated by the state extraction block and all other components are primarily generated using the topology description of the original parallel design. All blocks can be combined into a single design, which is synthesized to the targeted FPGA.

The FPGA offers also a great opportunity for various SoC architecture explorations without the need to re-synthesize every possible topology. Each core communicates with its neighbours via link memories. With a run-time reconfigurable link configuration, various topologies and configurations can be explored. Furthermore, IP providers could provide a transformed hypercell of their IP, which can be embedded in a standardized simulator framework. This hypercell should solely consist of a synthesized version of the combinational functionality of their IP, its interface ports, and its state interface.

D.1.5 FEEDBACK

We want to optimize the performance of the simulator. The selection of the groups of cells, which create the hypercells, will have a major influence on the performance. Large hypercells will update a large part of the state vector, however they require more resources. Therefore, we include a feedback between the synthesis and partitioning of the design, such that the partitioning can be tuned. With the feedback we can, for example, generate small hypercells that can be instantiate multiple times in the simulator or large hypercells that are instantiated once.

D.1.6 INITIAL RESULTS

In section 8.3.2, we manually transformed the GuarVC NoC architecture, such that it fits within the sequential simulation framework. The hypercell consisted of a single router and a stimuli interface, which enabled the injection of traffic into the network. The modified router was used to evaluate the performance of the NoC.

In this section we present some early results of the automated state extraction process. For this extraction, we use the original VHDL sources of the GuarVC router. This router is initially synthesized using the Virtex-II technology library. The non-transformed router's resource requirements is listed in table D.1.

The synthesized router's netlist is described as a graph and a single router is selected as a hypercell. The state of the hypercell is extracted and replaced by combinational replacement primitives with both rising and falling edge sensitivity. The bottom of table D.1 lists the resource requirements of this transformed router. On average, a flip-flop is replaced by four multiplexers. A single multiplexer is mapped onto a function generator. Compared to the original router, all flip-flops (Dffs) are gone and replaced by function generators.

Table D.1 – Router's resource requirements

Design	Function Generators	Dffs or Latches	Block rams
Non-transformed	3734	1732	0
Automatically transformed	10660	0	0

Table D.2 – Simulator's resource requirements

Design	Function Generators	Dffs or Latches	Block rams
Simulator w.o. hypercell	1050	301	69
Automatically generated simulator	8267	300	69
Manually created simulator	7152	1275	71

The simulator itself is synthesized apart from the router's hypercell. The simulator instantiates a black box hypercell design. In this design, all state and link memories, interconnection logic, the scheduler, overall control unit, interfaces, pipeline overhead, etc. are included. This simulator is capable of simulating a network consisting of 64 routers. The resource usage of this simulator framework, i.e. simulator without hypercell, is presented in table D.2.

The last step is the combination of both hypercell and simulator framework, such that the system is optimized for placement and routing. The resulting resource usage of this full design is also shown in table D.2. The usage of the function generators is significantly less than the sum of the function generators of the non-optimized transformed hypercell table D.1 and the simulator framework after synthesis. During the mapping phase, multiple multiplexers are combined in one function generator.

Timing analysis shows that the system has a maximum operational clock frequency of 23.5 MHz. This is almost twice the maximum operational frequency of 13.2 MHz of the manually created simulator.

In the manual transformation (see section 8.4), there is no clean separation between the router entity and the simulator framework. It would be interesting to compare the change in resources after transformation, but this is not possible. For example, the router entity includes the state memory already. The required resources for the manual transformation of the entity, including the link and state memories and the interconnection logic, excluding the scheduler and overall control unit, are shown in the bottom line of table D.2. As can be seen, the manually transformed simulator, even without the scheduler and control, generates more flip-flops and block rams than the automated one.

Stellingen

behorende bij het proefschrift

Exploration within the Network-on-Chip Paradigm

door

Pascal T. Wolkotte

I – De toename van de onvoorspelbaarheid, waargenomen in vele applicaties, kost meer energie.
(*Hoofdstukken 3 & 6*)

II – De toename van flexibiliteit en performance kosten oppervlakte en energie.
(*Hoofdstukken 4, 5 & 6*)

III – Voor toekomstige digitale systemen zijn de huidige FPGAs een uitermate geschikt simulatie platform.
(*Hoofdstuk 8*)

IV – De niptste overwinning geeft de grootste voldoening.

V – Zoals een promotor meerdere keren de teksten van zijn promovendus leest, hoort een promovendus meerdere keren de ideeën van zijn promotor.

VI – Virtuele busbanen zouden het asfaltgebruik in steden efficiënter maken. Het vereist wel alert gedrag van alle verkeersdeelnemers.

VII – Gelaagdheid zorgt voor begrip en een eerste realisatie. Ook schept het daarna nog onderzoeksmogelijkheden naar performance winst door cross-layer optimalisatie.

VIII – Goed systeembeheer is onzichtbaar. Hetzelfde geldt voor typografie.

IX – Analyse en begrip worden graag vervangen door een rekenmachine en schatting. Dit gaat ten koste van de intuïtie.

X – Schaarste leidt tot creativiteit.
(*Dit proefschrift*)