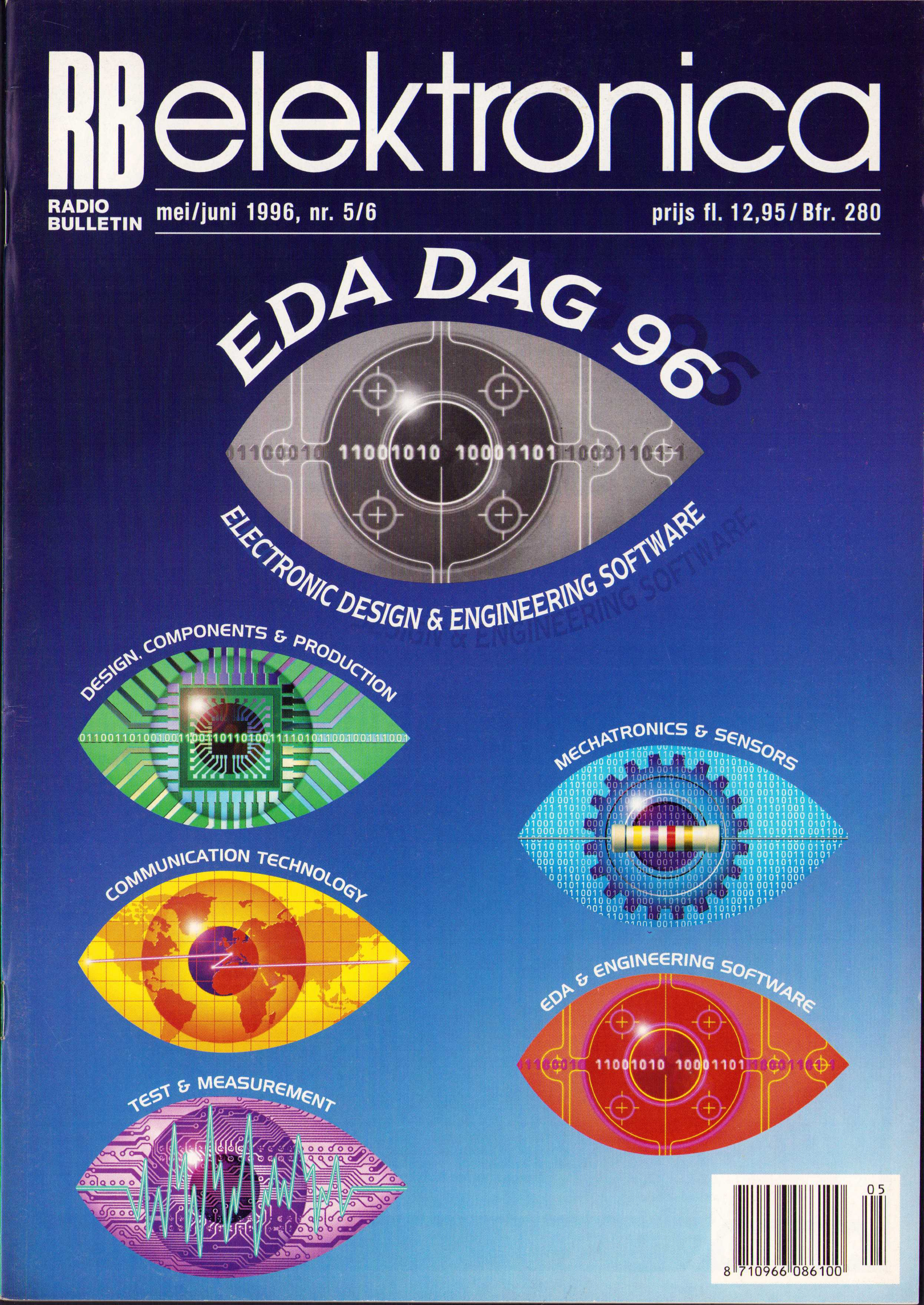
RB elektronica

RADIO BULLETIN

mei/juni 1996, nr. 5/6

prijs fl. 12,95 / Bfr. 280



8 710966 086100 05

AMPLIMO LEVERT NÚ RINGKERNTRAFO'S MET DE BESTE GARANTIE

KEMA KEUR

Het KEMA-KEUR-merk is de beste garantie voor kwaliteit en veiligheid. De AMPLIMO ringkerntrafo's dragen nu dit keurmerk.
AMPLIMO is de eerste in Nederland met KEMA-KEUR voor liefst 170 types van 15 t/m 1000VA.
Alle zijn uit voorraad leverbaar.

Topkwaliteit in kombinatie met een uitstekende veiligheid.
De wikkeling met de gevaarlijke netspanning is volledig omgeven door een drievoudige isolatie, welke liefst 5000V kan weerstaan.

Het ontwerpen en wikkelen geschieden zeer zorgvuldig en de eindcontrole wordt uitgevoerd volgens ISO9003.
Zelfs trafo's met andere wikkelingen in de 12 standaard formaten worden met het beroemde KEMA-KEUR geleverd!

Duidelijk advies over de toe te passen zekering voor optimale veiligheid.
Het voldoen aan de strenge KEMA eisen heeft bij AMPLIMO nauwelijks of geen prijsverhoging tot gevolg.

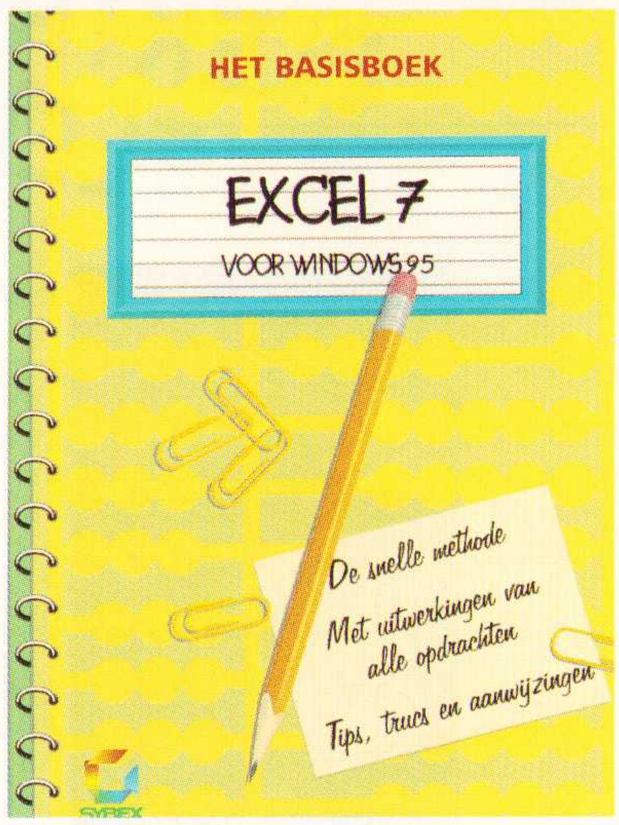
®

AMPLIMO

AMPLIMO b.v.
Vossenbrinkweg 1
7491 DA Delden

Telefoon 074 376 3765
Fax 074 376 3132

LAATSTE BOEKENNIEUWS

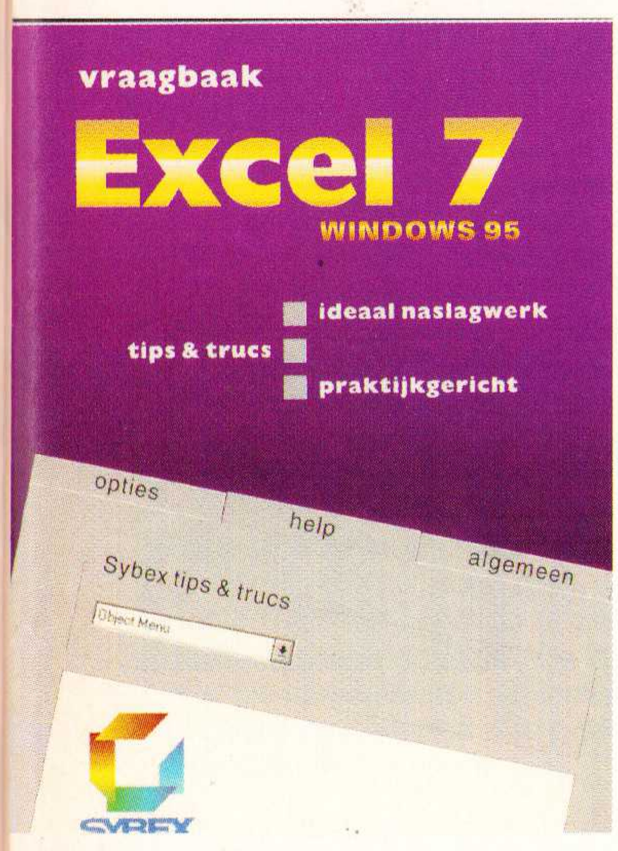


Titel: Het basisboek Excel 7 voor Windows 95
Bestelnummer: 750905
Prijs: ƒ 29,-
Porto: ƒ 6,-
Verkrijgbaar bij: De Muiderkring

De serie het basisboek is speciaal bedoeld voor de enthousiaste en geïnteresseerde PC-gebruiker die zich de nodige kennis van enkele standaard programma's zo snel mogelijk eigen wil maken.

Er is gekozen voor een stapsgewijze opzet in alle hoofdstukken:
- eerst wordt een onderwerp algemeen uitgelegd
- vervolgens kunt u door het uitvoeren van een opdracht zelf aan de slag
- in appendix b wordt in het algemeen de uitwerking van de opdracht beschreven
- het onderwerp wordt afgesloten met enkele tips en trucs en aanwijzingen voor de uitvoering van de opdracht

De eerste zes hoofdstukken behandelen de basisfuncties aan de hand van opdrachten, voorbeelden, tips en trucs. Bovendien komt u alles te weten over de mogelijkheden van het programma om gegevens af te drukken, in het scherm te tonen, te importeren en te exporteren. De laatste hoofdstukken gaan over de specialiteiten van het programma, bijvoorbeeld een ingebouwde macro-taal of extra tekstverwerkingsopties. in de bijlagen vindt u een praktische installatiehandleiding en de uitwerking van de opdrachten uit het boek.



Titel: Vraagbaak Excel 7 Windows 95
Bestelnummer: 750907
Prijs: ƒ 49,-
Porto: ƒ 6,-
Verkrijgbaar bij: De Muiderkring

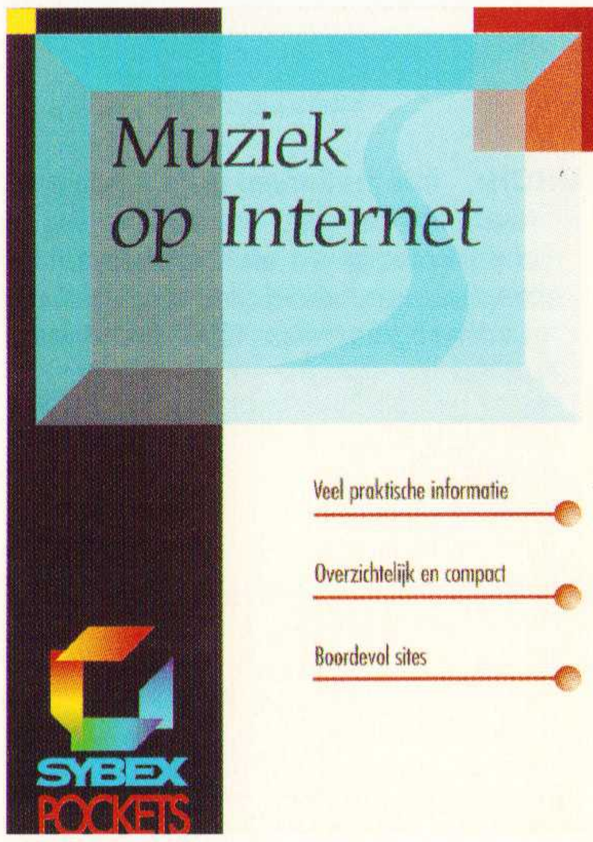
Deze vraagbaak kunt u gebruiken als eerste kennismaking met Excel 7 voor Windows 95, maar het is ook een ideaal naslagwerk. In hoofdstuk 1 leest u alles over de installatie en de grondbeginselen van Excel 7.
In hoofdstuk 2 worden de werkmappen en werkbladen besproken. Hoofdstuk 3 gaat over basisbewerkingen en in hoofdstuk 4 komen formules en functies aan bod.
In hoofdstuk 5 en 6 komen werkbladopmaak en belangrijke functies aan de orde en hoofdstuk 7 behandelt het afdrukken van werkbladen.
In hoofdstuk 8 leert u werkbladen controleren, documenteren en beveiligen. Hoofdstuk 9 gaat over grafieken en objecten, hoofdstuk 10 bespreekt het werken met databases en hoofdstuk 11 draaitabellen.
In hoofdstuk 12 gaat u eigen dialoogvensters maken. Hoofdstuk 13 bespreekt macro's en het laatste hoofdstuk gaat over werken met andere toepassingen.
Handige bijlagen en een index besluiten het boek.

Titel: Muziek op Internet
Bestelnummer: 750847
Prijs: ƒ 19,-
Porto: ƒ 6,-
Verkrijgbaar bij: De Muiderkring

Internet, het wereldwijde communicatienetwerk voor computers, groeit nog dagelijks en de mogelijkheden van het net zijn zeer omvangrijk.

Dit boek richt zich op wat er met Internet mogelijk is in relatie tot muziek. U kunt bijvoorbeeld informatie inwinnen over artiesten, componisten, nieuwe muziekuitgaven, distributeurs, fanclubs en nog veel meer. Of u kunt met andere muziekliefhebbers discussiëren over geliefde of juist verguisde muziekstukken. En dat niet alleen op het gebied van popmuziek, maar ook op het gebied van jazz, klassiek, country and western, en ongeveer alle andere categorieën.

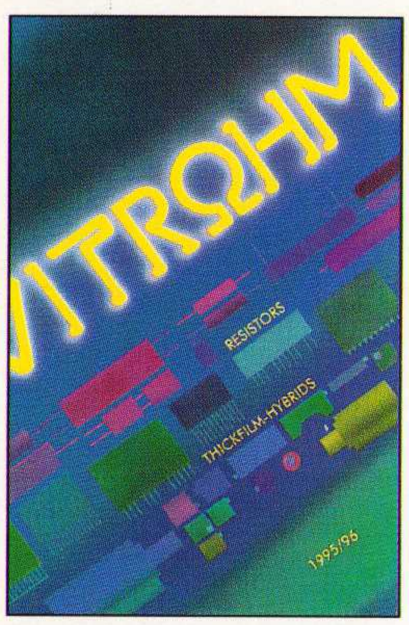
In deel 1 komen praktische zaken aan de orde als benodigde hard- en software, Internet-aanbieders en kosten. Ook wordt uitgelegd hoe u zich bijvoorbeeld abonneert op een nieuwsgroep en hoe u bestanden kunt downloaden.



In deel 2 is een beschrijving van een groot aantal zeer diverse sites, uiteenlopend van klassieke muziek tot rap-muziek. Met dit gedeelte als gids vindt u snel wat u zoekt, of het nu gaat om het downloaden van een muziekfragment van Schubert of het abonneren op een nieuwsgroep voor basgitaristen.

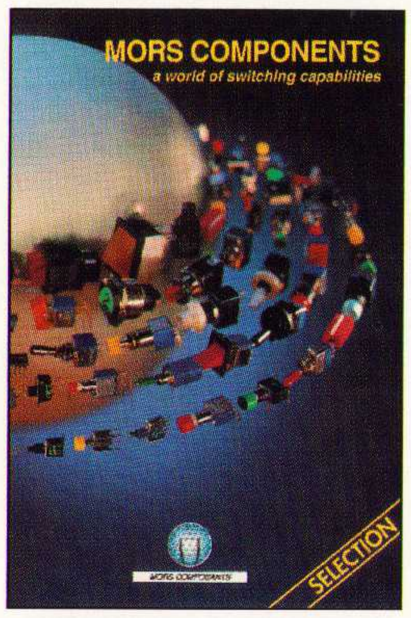
AMROH

ELEKTRONICA- EN ELEKTROTECHNIEK INTERNATIONAAL



VITROHM

Europees marktleider in draadgewonden weerstanden, tevens
- kool- en metaalfilmweerstanden
- netwerken
- hybride-schakelingen

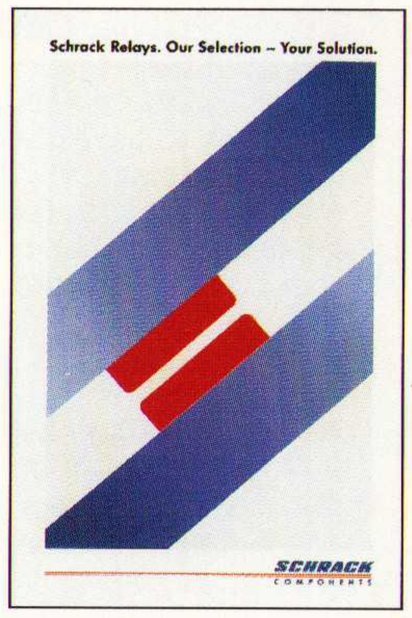


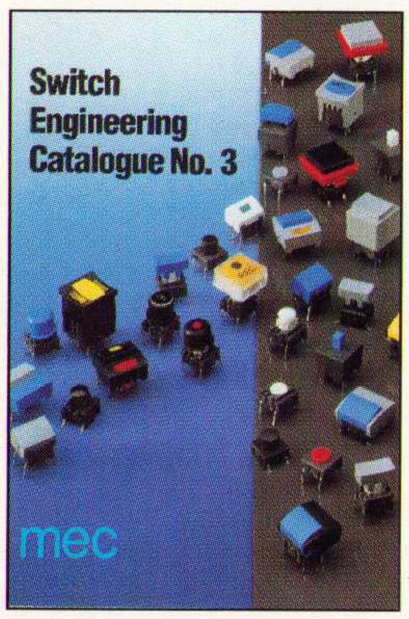
MORS

Een wereld van tuimel-, wiptoets-, drukknop-, schuif- en codeerschakelaars in miniatuur en standaarduitvoering

SCHRACK

Een relaisprogramma met allure:
- vermogens-printrelais van 1 tot 40 Amp.
- insteekrelais tot 30 Amp.
- accessoires, o.a. relaisvoeten met insteekmodules





MEC

Modulaire print-schakelaars
- standaard en SMD-uitvoering
- verlichte versies
- groot aantal accessoires in 7 kleuren

NCC

Toonaangevende fabrikant van elektrolitische condensatoren in axiale, radiale en SMD uitvoering

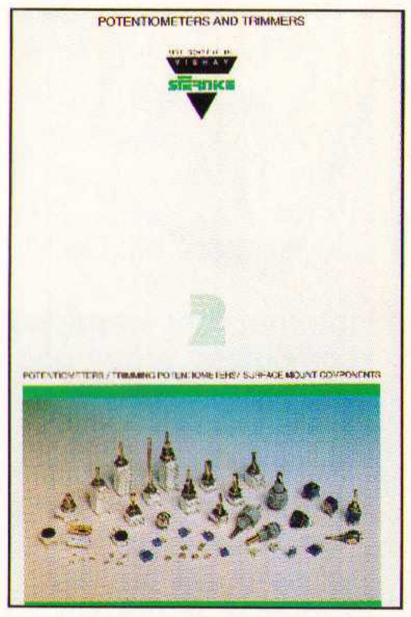
MINIMOTOR

- miniatuur DC motoren van ∅ 10 mm tot ∅ 35 mm
- vertraging tot 1.000.000 : 1
- borstelloze servomotoren
- motor- en tachogeneratoren
- impulsgevers

AMROH: internationaal een gerenommeerde naam als het gaat om de levering van elektronische en elektromechanische componenten; meet- en regelapparatuur en hoogwaardige HI-FI-produkten.

SFERNICE

- cermet enkel- en meerslagen trimmers
- industriële potentiometers in een grote verscheidenheid
- vermogens- en precisie weerstanden



BELLING LEE

- netontstoringsfilters
- zekeringen en houders
- meerpolige ronde connectoren
- DIL-relais
- trekmagneten

AMROH

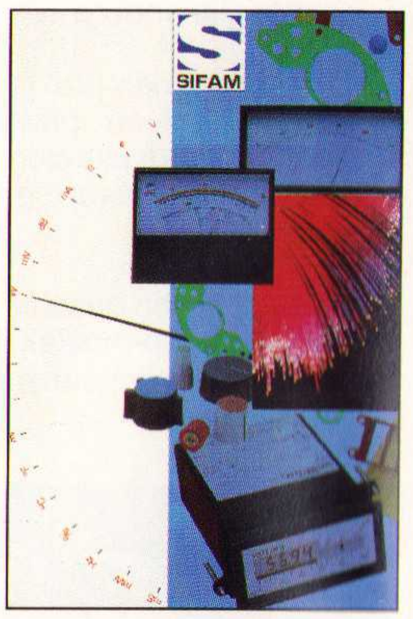
NEDERLAND: Hogeweyselaan 227
1382 JL Weesp
Postbus 370
1380 AJ Weesp
Tel: 02940-15350
Fax: 02940-12782

BELGIE: Amroh Electronics Belgium
Nieuwdreef 7
B-2328 Hoogstraten
Tel/Fax: 03/3150606

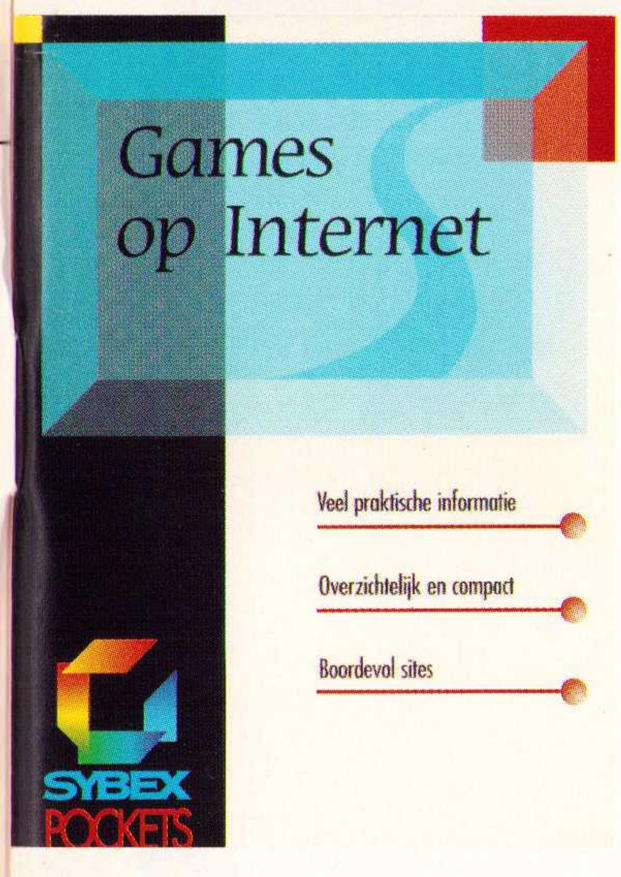
DUITSLAND: Amroh Electronics GmbH
Postfach 460201
D-47856 Willich
Tel: 02154-428461

SIFAM

Europa's grootste producent van:
- kunststof knoppen
- paneelmeters
- proces-indicatoren
- glasvezel-componenten



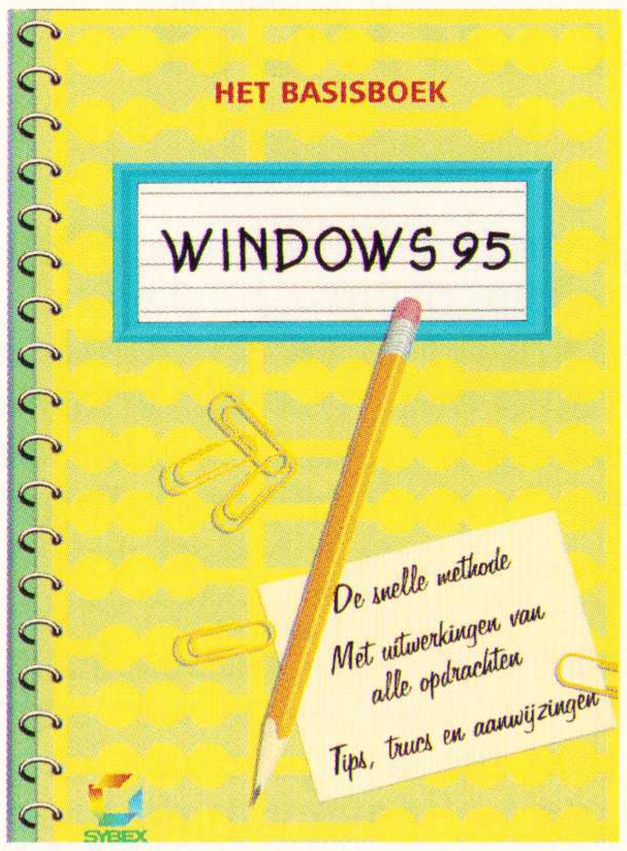
BOEKRECENSIES



Titel: Games op Internet
Bestelnummer: 750848
Prijs: ƒ 19,-
Porto: ƒ 6,-
Verkrijgbaar bij: De Muiderkring

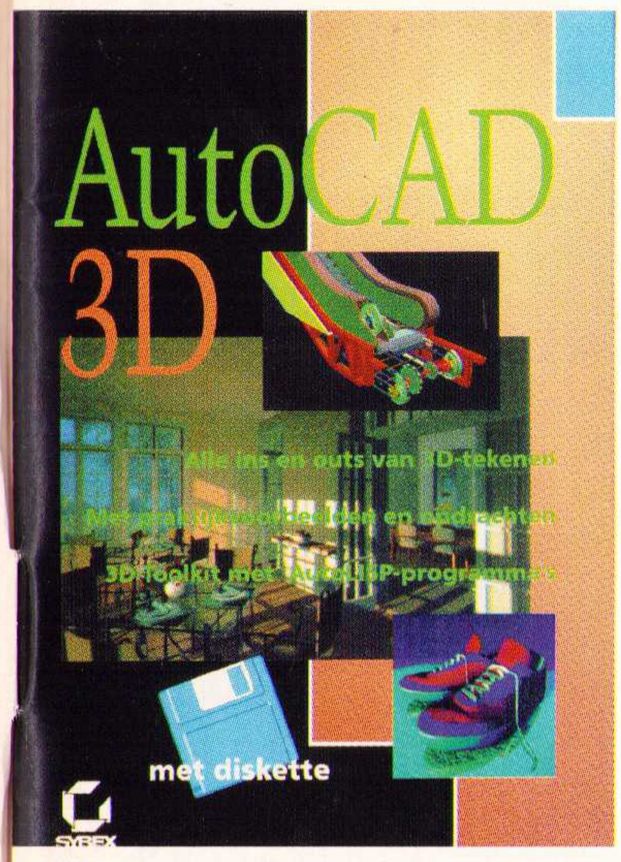
Internet, het wereldwijde communicatienetwerk voor computers, groeit nog dagelijks en de mogelijkheden van het net zijn zeer omvangrijk. Op Internet is veel informatie te vinden over games, maar u kunt er ook spelen. Elke game die u maar kunt bedenken heeft waarschijnlijk wel een nieuwsgroep, een FTP-site of een Web-pagina. U kunt de nieuwste demo's bekijken, games downloaden of via het net spelen met tegenstanders over de hele wereld. In deel 1 van dit boek komen praktische zaken aan de orde als benodigde hard- en software en Internetaanbieders. Ook wordt uitgelegd hoe u bijvoorbeeld bestanden zoekt en overzet naar uw computer. Deel 2 beschrijft een groot aantal games. Met dit gedeelte als gids vindt u snel de beste nieuwsgroepen, Web-sites of discussiegroepen. Er zijn aparte hoofdstukken over spelfabrikanten, controlespellen, arcadespellen en interactieve Internet spellen. Voor liefhebbers van games is er op Internet veel te ontdekken, onder andere dat ze minder slaap nodig hebben dan ze altijd dachten!

Titel: Het basisboek Windows 95
Bestelnummer: 750936
Prijs: ƒ 29,-
Porto: ƒ 6,-
Verkrijgbaar bij: De Muiderkring



De serie het basisboek is speciaal bedoeld voor de enthousiaste en geïnteresseerde PC-gebruiker die zich de nodige kennis van enkele standaard programma's zo snel mogelijk eigen wil maken. Er is gekozen voor een stapsgewijze opzet in alle hoofdstukken:
- eerst wordt een onderwerp algemeen uitgelegd
- vervolgens kunt u door het uitvoeren van een opdracht zelf aan de slag
- in appendix b wordt in het algemeen de uitwerking van de opdracht beschreven
- het onderwerp wordt afgesloten met enkele tips en trucs en aanwijzingen voor de uitvoering van de opdracht

De eerste hoofdstukken behandelen de basisfuncties aan de hand van opdrachten, voorbeelden, tips en trucs. Bovendien komt u alles te weten over de mogelijkheden binnen Windows 95. De laatste hoofdstukken gaan over de specialiteiten en instellingen van Windows 95. In de bijlagen vindt u een praktische installatiehandleiding en de uitwerking van de opdrachten uit het boek.



Titel: AutoCAD 3D
Bestelnummer: 750860
Prijs: ƒ 89,-
Porto: ƒ 6,-
Verkrijgbaar bij: De Muiderkring

Alle ins en outs van 3D-tekenen
Met praktijkvoorbeelden en opdrachten
3D-Toolkit met AutoLISP-programma's

Werken in een 3D-ruimte is inmiddels een integraal deel van het ontwerpproces. Iedereen die meer doet dan één aanzicht maken, moet wel werken in de 3D-ruimte en dat geldt ook voor iedereen die een perfecte presentatie wil maken. Dit boek is ontwikkeld als een praktijkgerichte studiehandleiding. In de hoofstukken 1 tot en met 14 maakt u stap voor stap kennis met de verschillende technieken en opdrachten die voor het 3D-tekenen nodig zijn. Elk hoofdstuk bevat een les waarin één of meer essentiële punten van het 3D-tekenen wordt behandeld. Per hoofdstuk wordt niet alleen geleerd hoe u bepaalde concepten aanpakt, maar krijgt u ook ervaring in het werken met modellen. De hoofdstukken 15 en 16 zijn specifiek voor de vakrichtingen werktuigbouw en architectuur. Hierin wordt een voor die disciplines kenmerkend model ontwikkeld, en krijgt u de gelegenheid twee praktijkvoorbeelden volledig uit te werken met behulp van alle technieken die u in de eerdere 14 hoofdstukken heeft leren hanteren. Hoofdstuk 17 is de zogenaamde 3D Toolkit. Hierin staan 53 kant en klare AutoLISP-programma's waarmee uw werk in 3D aanzienlijk kan worden versneld en uitgebreid, en die u kunt gebruiken alsof het gewone AutoCAD-opdrachten zijn.
Om met dit boek aan de slag te kunnen, moet u de basis-opdrachten van AutoCAD kennen. U hoeft geen AutoCAD-expert te zijn en u hoeft niets te weten van programmeren in AutoLISP. Het boek is ontwikkeld aan de hand van AutoCAD versie 13, maar heeft u deze versie niet, dan kunt u grote delen van het boek gebruiken voor de versies 11 en 12. Kortom, dit boek leert u op een effectieve wijze alle ins en outs van het 3D-tekenen.

RB Elektronica, mei 1996

EDA .... inmiddels een begrip (een must?) binnen de elektronica

De afgelopen jaren is de trend in de ontwikkeling gegaan van hardwarematige proefseries en nulprodukten naar softwarematige analyses en produktontwikkeling op de computer. Vooral is deze trend merkbaar voor produkten waarin de functionaliteit en de verkoopbaarheid gedeeltelijk of geheel worden bepaald door de ingebouwde elektronica.

Onder de noemer EDA, Electronic Design Automation, is de afgelopen jaren een hele range van softwareprodukten op de markt gebracht die aangeschaft kunnen worden als 'gereedschap' ten behoeve van de produktontwikkeling. Bedrijven die op dit terrein 'bijblijven' als doelstelling willen en/of moeten realiseren, kost dat moeite, zowel inhoudelijk als financieel. Met het oog op die moeite organiseerde de brancheorganisatie voor Industriële Elektronica via haar federatie Het Instrument op donderdag 9 mei een zogenoemde EDA Dag in het hotel Mecure in Nieuwegein.

Deze gezamenlijke uitgave van Het Instrument en RB Elektronica bundelt de resultaten van deze dag.

Veel plezier.

Dirk Scheper

RB ELEKTRONICA
(Jaargang 65)

Is een uitgave van
De Muiderkring B.V.,
Hogeweyselaan 227,
Postbus 313,
1380 AH Weesp
telefoon: 0294-450460 (ISDN)
telefoon: 0294-415210
telefax: 0294-412782
bank: 48 49 54 563
giro: 83214

Directie:
Ir. S.M.Th. Kremer

Hoofdredacteur:
Ing. D.J.F. Scheper

Eindredactie:
J.E.E. van der Hoogte

Vaste medewerkers:
J. van Emden, L. Foreman, J.H.M. Goddijn, ir. S.J. Hellings, O.C.A. van Lidth de Jeude, J.W. Richter, drs. ing. C.F. Ruyter, J. Smilde, ing. B. Stuurman, C.G.C. van der Vlies.

Vormgeving:
MK Studio

Advertentieverkoop:
Bosch & Keuning, Postbus 1, 3740 AA Baarn, tel. 02154 - 82340, fax. 02154 - 82344 en/of G. Belecke, tel/fax. 02159-36293

Abonnementen:
Abonnementsprijs per jaar:
f 75,-/Bfr. 1500.
Studenten: f 60,-/Bfr. 1200.
Abonnementen worden automatisch verlengd, tenzij uiterlijk drie maanden voor het einde van de aflooptermijn schriftelijk bericht is ontvangen. Vermeld bij correspondentie altijd uw abonneenummer (zie wikkel).

Typografie:
MK Typopress

Druk:
grafische bedrijven
Bosch & Keuning, Baarn

Distributie:
Betapress

RB in België:
Keesing Uitgevers N.V.
S. van der Rijt
Redactionele bijdrage en correspondentie sturen naar:
Keesing Uitgevers N.V.
Keesinglaan 2-20, B 2100
Antwerpen/Deurne.
Tel.: 03-324.38.90
Fax: 03-324.38.98
Bankrekening: 408-0502011-04

ISSN: 0928-5008

INHOUD

RB Elektronica mei/juni 1996

Cost of Ownership & Return on Investment

van investeringen in software voor elektronica ontwikkelingen

Embedded Software

VHDL, Hard- & Software definitietalen

E-CAD, Hardware Design en Engineering

COVERFOTO: Het nieuwe logo van Het Instrument. Bijzonder is de verschillende ogen voor de uiteenlopende disciplines.
(Coverfoto: Het Instrument, Amersfoort)

INTRODUCTIE:

ONTWERPAUTOMATISERING,

Rendement op investeringen in software-gereedschap ter discussie gesteld

De ontwikkeling van produkten gebeurt steeds meer softwarematig. Dat geldt zeker voor produkten waarin de functionaliteit en de verkoopbaarheid gedeeltelijk of geheel worden bepaald door de ingebouwde elektronica.

Onder de noemer EDA, Electronic Design Automation, is de afgelopen jaren een hele range van softwareprodukten op de markt gebracht die aangeschaft kunnen worden als 'gereedschap' ten behoeve van de produktontwikkeling. Bedrijven die op dit terrein 'bijblijven' als doelstelling willen en/of moeten realiseren, kost dat moeite, zowel inhoudelijk als financieel. Met het oog op die moeite organiseerde de brancheorganisatie voor Industriële Elektronica via haar federatie Het Instrument op donderdag 9 mei een zogenoemde EDA Dag in het hotel Mecure in Nieuwegein. Deze gezamenlijke uitgave van Het Instrument en RB Elektronica bundelt de resultaten van deze dag.

State of the Art & Cost of Ownership

Het concept van de EDA Dag is tamelijk uniek. In de ochtend presenteren de leveranciers de State of the Art, in drie parallelsessies van elk zes lezingen, gratis toegankelijk, gericht op 'de man achter het toetsenbord'. Het middaggedeelte bestaat uit een samenhangend seminarprogramma waarin begrippen als Cost of Ownership en Return on Investment centraal staan. Gebruikers die een investering in de toekomst overwegen kunnen in dit seminar van collega's leren wat 'geloof en werkelijkheid' is bij het nemen en uitvoeren van een beslissing te investeren in softwaretools voor produktontwikkeling.

Het middagprogramma is gericht op het management van bedrijven die zelf produkten ontwikkelen en slechts tegen betaling toegankelijk. De sprekers in dit gedeelte zijn afkomstig van bedrijven met gebruikservaring, zoals Chess Engineering, HSA, Scantech en AGFA Gevaert. Dit deel van het programma staat onder voorzitterschap van de heer Dick Rakhorst, voorzitter van de Development Club een vereniging van vijfendertig onafhankelijke ontwikkelbedrijven.

Het eerste gedeelte van deze uitgave omvat de bijdrage van de sprekers van de middagsessie: Cost of Ownership.

Embedded Software, VHDL en E-CAD

De achttien presentaties in het ochtendgedeelte zijn verdeeld over drie parallelthema's: Embedded Software, VHDL hard- & software definitietalen en E-CAD hardware design en engineering. Deze driedeling geeft aan waarover het gaat als gesproken wordt over de huidige State of the Art van ontwerptechnologie en vormt ook de hoofdstukindeling van het vervolg van deze uitgave.

Embedded Software Design Tools is die software die gebruikt wordt om de software te ontwikkelen die nodig is om moderne elektronicahardware de functionaliteit te geven die de applicatie vraagt; software waarmee software kan worden gemaakt die zorgt dat het apparaat 'werkt'.

Met de groeiende complexiteit van produktontwikkeltrajecten wordt het steeds meer noodzakelijk om vooraf een goede structuur aan te brengen waarbinnen de beoogde ontwikkeling inderdaad softwarematig kan worden gerealiseerd. VHDL, Very High Level Hardware Definition Language, een beschrijvingstaal op een bepaald abstractieniveau, is een software hulpmiddel datdaarbij langzamerhand onmisbaar wordt.

Het onderdeel E-CAD omvat de verhalen die inzicht geven in wat nu beschikbaar is aan tools voor het daadwerkelijk realiseren van een elektronische schakeling die produceerbaar en testbaar is.

Elk jaar een EDA/Engineering software show

De Nederlandse Brancheorganisatie voor Industriële elektronica organiseerde in mei 1995 voor het eerst een EDA/engineering software show. Deze technologieshow was toen onderdeel van de branche vakbeurs Electronics. Met vier andere shows, Test & Measurement, Mechatronics & Automation en Components & Production spraken de leveranciers toen een brede groep aan van (potentiële) ontwikkelaars, fabrikanten en gebruikers van elektronicaprodukten.

In navolging van de in 1994 succesvol georganiseerde separate Test & Measurement Dagen, is nu besloten om in het jaar dat de brancheorganisatie geen brede vakbeurs organiseert, ook voor EDA/engineering software een kleinschalige, laagdrempelige technologieshow te organiseren.
Met de realisatie van deze eerste editie, de EDA Dag '96, wordt dus een traject ingezet van elk jaar een activiteit waarin de leveranciers en gebruikers van elektronica ontwikkelsoftware elkaar ontmoeten om de nieuwste produkten te zien en te analyseren en om ervaringen in het gebruik uit te wisselen.

In deze eerste specifieke nationale EDA Dag staat niet alleen de technologie als zodanig centraal. Nadrukkelijk hebben de organisatoren ervoor gekozen in een strategisch middagseminar het investeringstraject en het rendement daarop ter discussie te stellen.

De organisatie van deze EDA Dag en de vakbeurs, waarin volgend voorjaar weer een EDA/engineering software zal zijn opgenomen, is in handen van het bureau van de Federatie Het Instrument, waarbij de leveranciers en een groot aantal gebruikers zijn aangesloten via hun brancheorganisatie voor Industriële elektronica

drs. J.C. Groeneveld
Het Instrument
Nederlandse Branche organisatie voor Industriële elektronica
Postbus 2099
3800 CB Amersfoort
Tel: 033-4657507

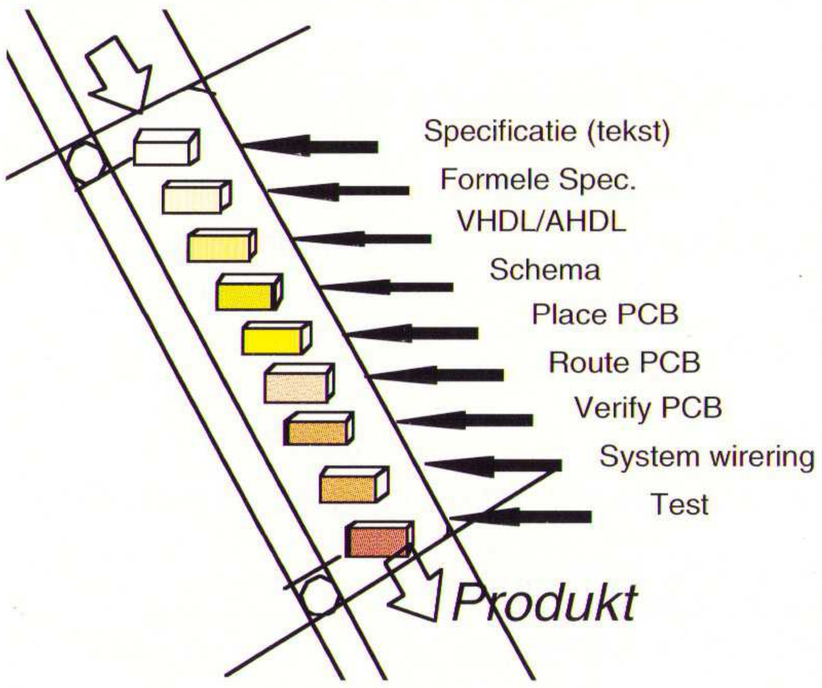
Electronic Design Automation OK? en dan.... Met EDA tooling alleen redt je het niet

1.1 Inleiding

De ontwikkeltijd, kwaliteit en reproduceerbaarheid van de electronica in produkten kan drastisch verbeterd worden door de designflow aan een grondige inspectie te onderwerpen.

Vragen die men zichzelf kan stellen zijn:

Beheers ik het electronica-ontwerpproces?
Is mij bekend wat de kosten van redesign zijn?
Hoe vaak moeten wij ontwerpfouten debuggen?
Hoeveel tijd kost dit?
Is mij duidelijk wat de oorzaak van de fouten is?
Ligt het aan het ontwerpen?
Ligt het aan de specificatie?
Voldoen de componenten in het testsysteem aan hun specs?
Hoeveel tijd stoppen wij in het onderhouden van de ontwerpsystemen?
Hoeveel tijd kost het onderhoud van de bibliotheken?
Weet ik zeker dat bibliotheken of componenten niet op een andere wijze ter beschikking staan?



Figuur 1. Een nadere uitwerking van het electronica ontwikkelproces.

Kan ik goed voorspellen hoeveel tijd een ontwerp zal gaan kosten?
Hoeveel uitval heeft de produktie?
Is bekend wat de oorzaak is van de uitval?
Hoeveel field service wordt veroorzaakt door ontwerpfouten en welk deel door componentenslijtage(MTBF)?

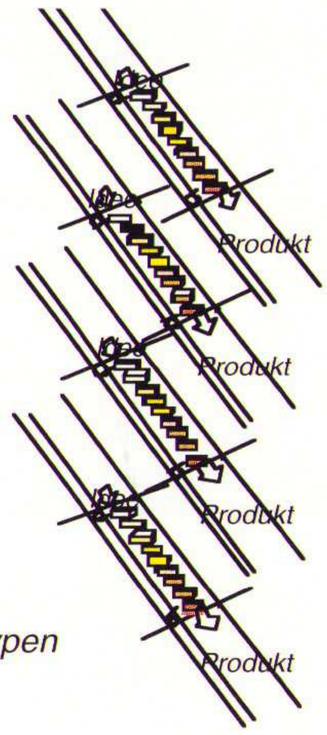
1.2 Electronica-ontwikkelproces

Op het electronica-ontwerpproces kan een metafoor worden loslaten door te kijken alsof het een produktiestraat betreft:

De grote lijnen zijn vaak:
Idee -> Schema -> PCB layout -> Produkt

Een korte analyse van deze ontwikkelstraat leidt al snel naar een verdere uitwerking voor de verschillende technologieën. Er is geen universele ontwikkelstraat voor alle electronica-ontwerpen op te stellen. Een heel groot deel van de electronica-ontwerpen kan worden opgesplitst in de volgende hoofdonderdelen (en daarmee dus ook de verschillende produktiestraten):

- Systeem specificatie
- PCB design
- FPGA design
- ASIC design
- Analog System design
- DSP design
- etc. etc



Figuur 2. Er zijn verschillende ontwikkelstraten voor de verschillende typen electronica.

Deze opsplitsing wordt in hoofdzaak veroorzaakt door de verschillende toegepaste technologieën.
De ontwikkelstraat moet natuurlijk ook worden afgestemd op het eindprodukt. Een ontwikkelproces voor een pacemaker zal anders verlopen dan die voor een electronische lichtregelaar. Bedenk eens welke invloed dit laatste heeft op de designflow!

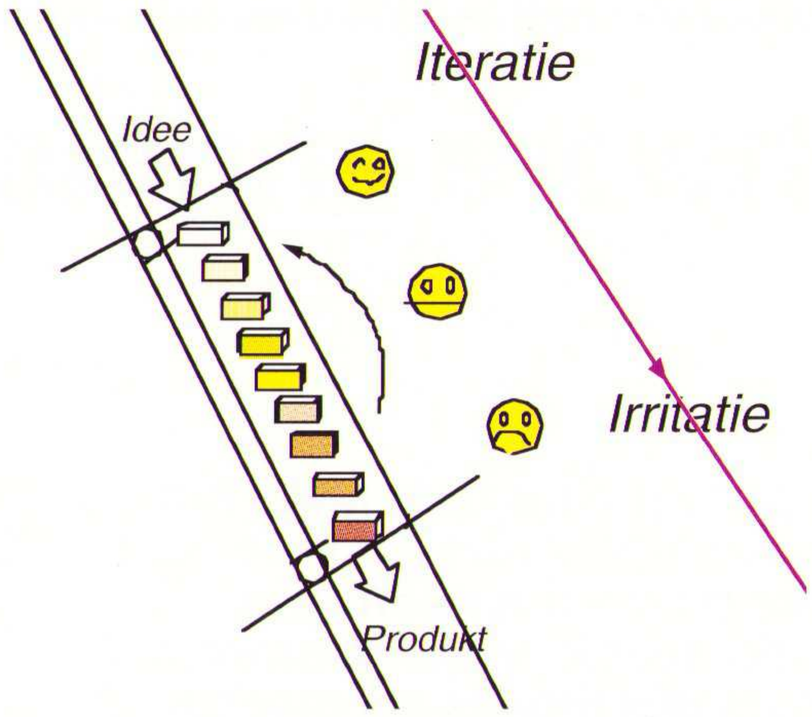
1.3 Design iteraties

Het is zinvol om de ontwikkelstraat te bekijken vanuit het gezichtsveld dat een ontwikkelproces altijd bestaat uit correcties van het reeds uitgevoerde.
De meest ideale produktiestraat is een stap voor stap proces waarbij nooit een stap terug wordt gedaan omdat alle stappen 100% goed zijn. Dit is in de electronica-ontwikkeling mogelijk! Er zijn bedrijven in Nederland die electronicaprodukten ontwerpen, produceren en afleveren zonder de electronica in het eindprodukt getest te hebben! Dit kan alleen door een uitermate strak voorgeschreven ontwerpproces. Bij de analyse van het ontwerpproces moet
normaal gesproken terdege met iteraties in het proces rekening worden gehouden. Enkele adviezen om iteraties niet te laten uitmonden in irritaties:

Zorg voor een goede eenduidige specificatie.
Controleer zoveel mogelijk tussentijdse stappen door het toepassen van:
Specificatie simulatie
Functionele simulatie
Timing verificatie
Board level simulatie
Systeem simulatie
Natuurlijk horen daar ook de testen bij. (Testen doe je op hardware, simuleren met software)
Het electronica-ontwikkelproces (de produktiestraat) is verder sterk afhankelijk van de historie van ontwerpmethoden en gereedschappen die in uw bedrijf zijn toegepast. Het is slechts voor enkelen weggelegd de ontwikkelstraat regelmatig volledig te vernieuwen.

1.4 Bibliotheken

Het is zinvol nog een andere analogie met EDA(Electronic Design Automation) te trekken. Het voorbeeld dat ik wil gebruiken komt uit de



Figuur 3. Naarmate design iteraties langer op zich laten wachten nemen irritaties toe.

binnenhuis architectuur. Er zijn veel systemen om gebouwen met hun inrichting te tekenen. De kwaliteit en toepassing van deze software hangt niet alleen af van de inwerktijd en gebruikersgemak. Een zeer belangrijk onderdeel is de bibliotheek van waaruit geput kan worden voor de inrichting:
Hoe uitgebreid is deze bibliotheek?
Bevatten de componenten alle noodzakelijke informatie zoals materiaal, kosten van het component, levertijd, leverancier, toleranties etc. etc..
Kan ik bibliotheken van andere pakketten gebruiken?
Kan ik informatie er weer uithalen voor de produktie of voor de kostprijsberekening?

Bibliotheken zijn de basisgegevens voor het electronica-ontwerp. Het is zeer aan te bevelen de tijd die daar door uw mensen wordt ingestopt aan een nader onderzoek te onderwerpen:
Worden componenten weleens dubbel aangemaakt?
Zijn er afspraken over de notatiewijze?
Is er een goede link tussen de ontwerpers en hun libraries en de personen die het systeemontwerp doen? Zit daar dubbel werk tussen?
Weet u zeker dat al deze informatie niet op een andere wijze beschikbaar is?
Wat gebeurt er met eigengemaakte symbolen, shapes en simulatiemodellen?
Ik weet zeker dat hier aanmerkelijke kwaliteitsverbeteringen en kostprijsreduktie bij elk bedrijf te realiseren is.

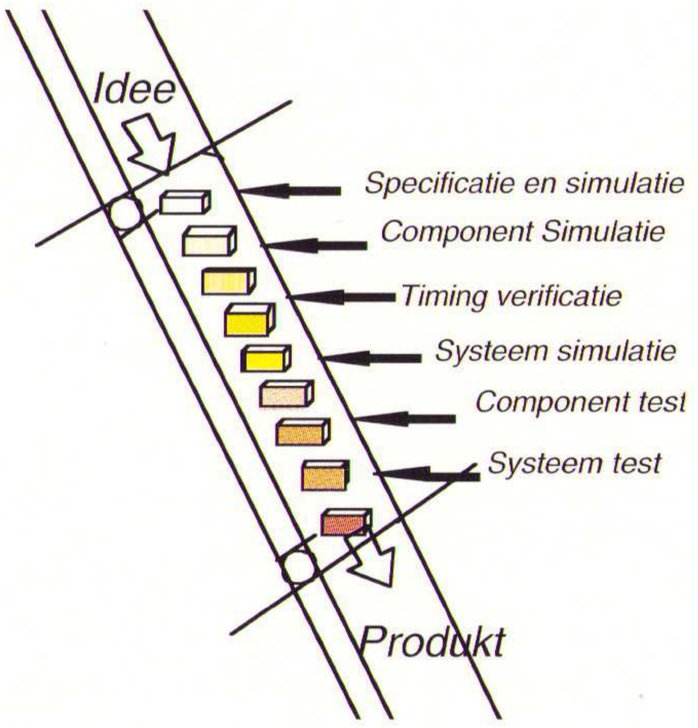
En enkele vragen op het gebied van produkt data management:
En hoe staat het met het beheer van oude ontwerpen?
En correcties daarop?
Wie weet welke versies van de software waar bij horen en hoe ze terug te vinden zijn?

OK, dit zijn veel kritische vragen. Maar geen reden voor ongerustheid. Er is geen enkel bedrijf dat al deze aspecten volledig onder controle heeft. Om deskundig advies te geven heeft TRANSFER Electronic Design Support een nieuwe activiteit in het leven geroepen die bovenstaande aspecten samen met de ontwerper kan doornemen, adviseren en zelfs delen ervan kan uitvoeren voor hem.

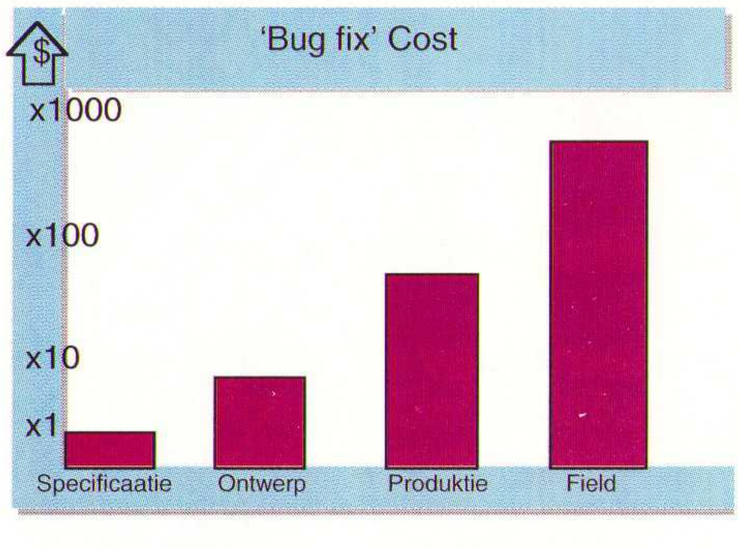
EDAServices Europe verbetert het electronica-ontwerpproces zonder direct aan nieuwe tools te denken. Ze verbetert door de bestaande ontwerpomgeving beter in te zetten. De belangrijkste verbeteringen zijn aan te brengen op de volgende gebieden:

1.5 Conclusies

Een nadere kritische blik op vervolgacties op elk ontwerpproces leidt zeker tot:
Betere voorspelbaarheid
Betere reproduceerbaarheid
Minder redesigns
Kortere designflow



Figuur 4. Design iteraties voorkomen door simulatie en test.



Figuur 5. De kosten voor het opsporen van een ontwerpfout in de verschillende fasen. Let daarbij vooral op de exponentiële schaal!

Te bereiken doelstellingen zijn:
Snellere reactie op marktveranderingen
Meer alternatieven voor uw klant
Betere bereikbaarheid van het TQM

Maar vooral: Denkt alleer gij doende zijt, en al doende denk dan nog.

Ton Zengerink
TRANSFER Electronic Design Support
Auke Vleerstraat 4
7521 PG Enschede
053-4330336
t.zengerink@transfer.nl

Bibliotheken: (Voorkomen dubbel werk, dubbele informatie, betere modellen en beter ontwerp)
- **Generatie: Symbolen**
- **Simulatiemodellen analoog, digitaal, timing, EMC, Cross-talk**
- **Produkt data management:**
- **Componentenbeheer**
- **Projectdata beheer**

EDA tools: (Efficiënter gebruik, betere informatie naar andere systemen)
- **Advies (tools & organisatie)**
- **Interfacing Tools <-> EDA tools**
- **Tools <-> non EDA tools**
- **Customization Tools<-> Ontwerpers**
- **Tools <-> Technologie**
- **Tools <-> Documentatie**

Ontwerpers (Beter gemotiveerd, betere communicatie, efficiënter gebruik mensen en middelen)
- **Begeleiding = in house CAD support**
- **Systeem specificatie trainingen**
- **Communicatie voor technici trainingen**
- **Technologie trainingen**
- **High level design trainingen**

Sharing (Kosten-sharing, kortere inwerktijd, lagere initiële investeringen)
- **Bibliotheken**
- **Ontwerpers**
- **Tools**

Real time software engineering strategieën

Inleiding

Software is een niet meer te stuiten opmars begonnen binnen technische produkten. Een produkt of een systeem met enig vernuft herbergt al snel enkele manjaren software ontwikkeling. Sinds de jaren vijftig neemt software steeds meer functies over van traditionele disciplines als mechanica, elektronica en electrotechniek. Software soupeert tegenwoordig al snel veertig tot vijftig procent van een produktontwikkelbudget. De hoeveelheid geld uitgegeven aan de arbeidsintensieve software ontwikkeling groeit dan ook jaarlijks met gemiddeld 12%. Een ander belangrijk facet waar software steeds meer haar stempel op drukt is de 'time to market'.

Helaas heeft de opmars van software ook haar schaduwzijde. Door de explosieve groei van de vraag en de toenemende belangrijkheid van software, heeft de software industrie te kampen met twee probleemgebieden: een achterblijvende produktiviteit en een slechte kwaliteit. Software staat berucht om het overschrijden van kosten en tijdsplanningen, en het optreden van fouten en tekortkomingen. Ongemak, forse financiële schade of zelfs slachtoffers zijn het gevolg wanneer software het op kritieke momenten laat afweten, of te laat wordt geleverd.

Reeds in de jaren zestig hebben deskundigen de noodklok geluid. Zij voorspelden software een grote toekomst, maar waarschuwden tegelijkertijd voor de grote gevaren van software. Als men er niet in zou slagen om grip te krijgen op de ontwikkeling van software, dan zou software de beperkende factor worden bij het gebruik van informatietechnologie. In 1968 introduceerden zij de term 'software engineering' met als doel de bouw van software doelmatig, voorspelbaar en betrouwbaar te maken door beproefde ingenieursmethodieken toe te passen.

Toch heeft het nog een kwart eeuw geduurd voordat de industrie wakker werd geschud. Vandaag de dag onderkennen steeds meer bedrijven de cruciale rol die software speelt bij de bescherming van hun marktpositie. Men is bereid te investeren in een structurele verbetering van de ontwikkelwerkzaamheden.
Dat niet eerder is gereageerd kan worden toegeschreven aan het ontbreken van softwarekennis en -inzicht op bijna alle managementlagen. Ook was geen sprake van een essentieel concurrentievoordeel omdat alle partijen met vergelijkbare problemen kampten. Verder was er een tekort aan know how en aan professionele hulpmiddelen. Maar ook het kwaliteitsdenken (ISO) en de zwaardere regelgevingen hebben hun steentje bijgedragen aan de bewustwording.

De oorzaak van de problemen waarmee de software industrie te kampen heeft ontspringen voor een groot deel uit wat genoemd mag worden de 'weaknesses' van software ontwikkeling.

2.2 Software 'weaknesses'

We hebben te maken met een:
Achterblijvende produktiviteit.
Matige kwaliteit.
Toenemende complexiteit.

De interactie tussen produktiviteit, welke al vrij snel aan manuren en dus kosten is te koppelen, en de verschillende software kwaliteitsaspecten (betrouwbaarheid, gebruikersgemak, portability, efficiency, etc.) is zeer complex. Zo ook de interacties tussen de verschillende kwaliteitsaspecten onderling.
De produktiviteit kan geen pas houden met de snelle groei van het aantal LOC (Lines of Code) dat men wenst te ontwikkelen. Het uitbreiden van de capaciteit door meer mensen en de verhoging van de produktiviteit per medewerker hebben geen oplossing gebracht.

Door het tekort aan capaciteit en de groeiende complexiteit neemt de kwaliteit af. Directe gevolg: Software levert niet de verwachte voordelen, leveringen worden vertraagd, verlies van resources aan re-work.
Indirect gevolg: Verlies van resources aan extra onderhoud

Professionele software ontwikkeling verdient de aandacht van het top-management. Alleen bedrijven die tijdig en strategisch 'durven' te investeren in de verbetering van de software ontwikkeling, of de software ontwikkelprocessen, hebben in de toekomst in de vorm van software een uitstekend wapen erbij in de concurrentiestrijd.

Voordat we ingaan op de kansen die ons worden geboden om software ontwikkeling te verbeteren gaan we kort in op het software ontwikkelproces en haar omgevingsfactoren.

2.3 Het software-ontwikkelproces

Doel van het ontwikkelproces is het afleveren van een (deel)produkt, een werkend programma compleet met documentatie. Voor een dergelijk produkt afgeleverd kan worden moet een aantal activiteiten uitgevoerd worden. Startpunt voor deze activiteiten is het idee, dat voor een bepaald onderwerp een systeem gebouwd moet worden. Wil men vanuit dit meestal vage idee komen tot een werkend en gedocumenteerd produkt, dan zal een ontwerpproces moeten worden doorlopen. Ontwerpen is een creatieve bezigheid. Dat wil zeggen, dat tijdens het ontwerpproces iets tot stand komt dat er van te voren nog niet was. Ontwerpen is een proces, waarbij afwisselend analyse-stappen en synthese-stappen worden gezet.

In afbeelding 1 zijn de typische activiteiten die aan het ontwikkelproces ten grondslag liggen, weergegeven.

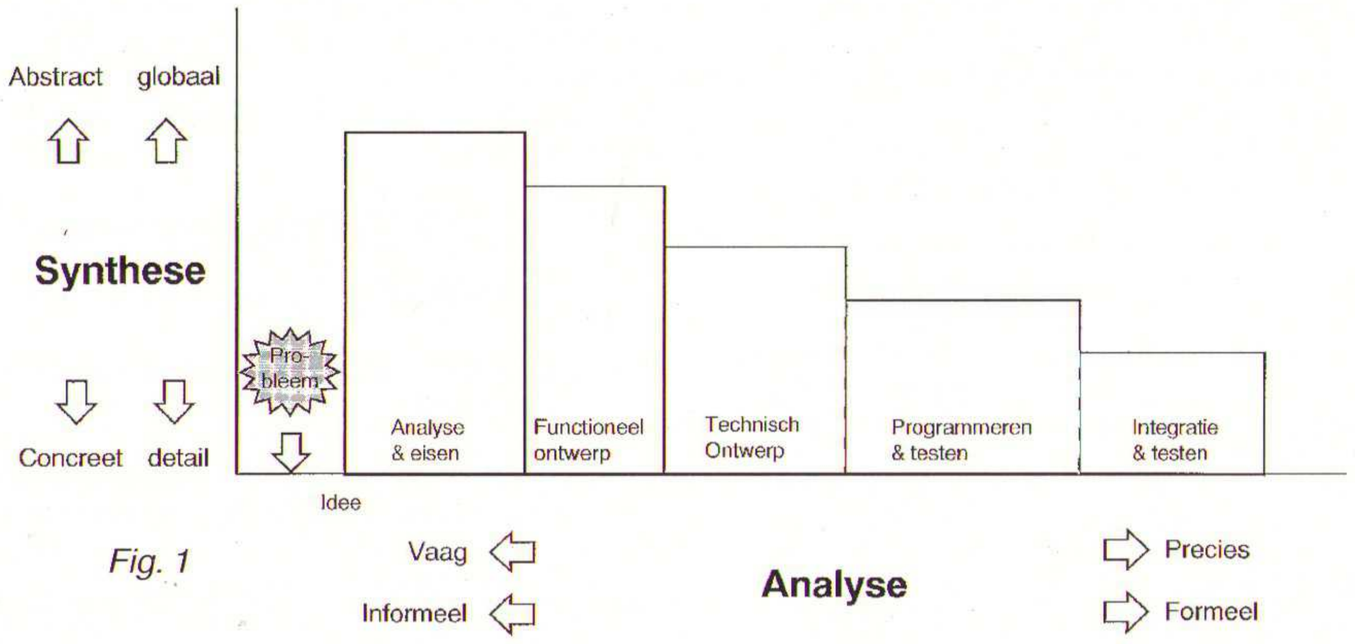


Fig. 1

Omdat men in het begin van het ontwerpproces veel abstracter bezig is dan aan het einde, is het nemen van een ontwerpbeslissing in het begin veel moeilijker dan aan het einde. In de praktijk is de verdeling van tijd, geld en mankracht over te onderscheiden fasen van de ontwikkeling van informatiesystemen hiermee echter in tegenspraak.

2.4 Verdeling van de inspanningen in het omgevingsmodel

In afbeelding 2 is het software-ontwikkelproces gepositioneerd binnen een organisatie. Het is een praktische methode om inzicht te krijgen in de distributie van de totale bestedingen over de afzonderlijke activiteiten waaruit een organisatie kan kiezen om de toegevoegde waarde aan hun produkten mee te geven. De waarden zoals in dit schema vermeld komen voort uit onderzoeken van Porter (Business School Harvard).

Wat direct opvalt is dat 4/5 van alle bestedingen wordt opgesoupeerd door de werkzaamheden op het discipline- en activiteitenniveau. -12% op activiteitenniveau door aandachtsgebieden als Project management, Quality Assurance en Configuration Management en Version Control. -Maar liefst 68% wordt besteed aan de technische effort binnen het software ontwikkeltraject voor het transformeren van inputs naar een afgerond produkt.
Wat hier opvalt is vooral de post 'rework' dat maar liefst 30% van de totale resources vereist.
Hieronder wordt verstaan: alle niet voorziene herstelwerkzaamheden omdat de specificaties, ontwerpen of code foutief zijn of verkeerd geïnterpreteerd of omdat er tussentijds een heroriëntatie heeft plaatsgevonden.

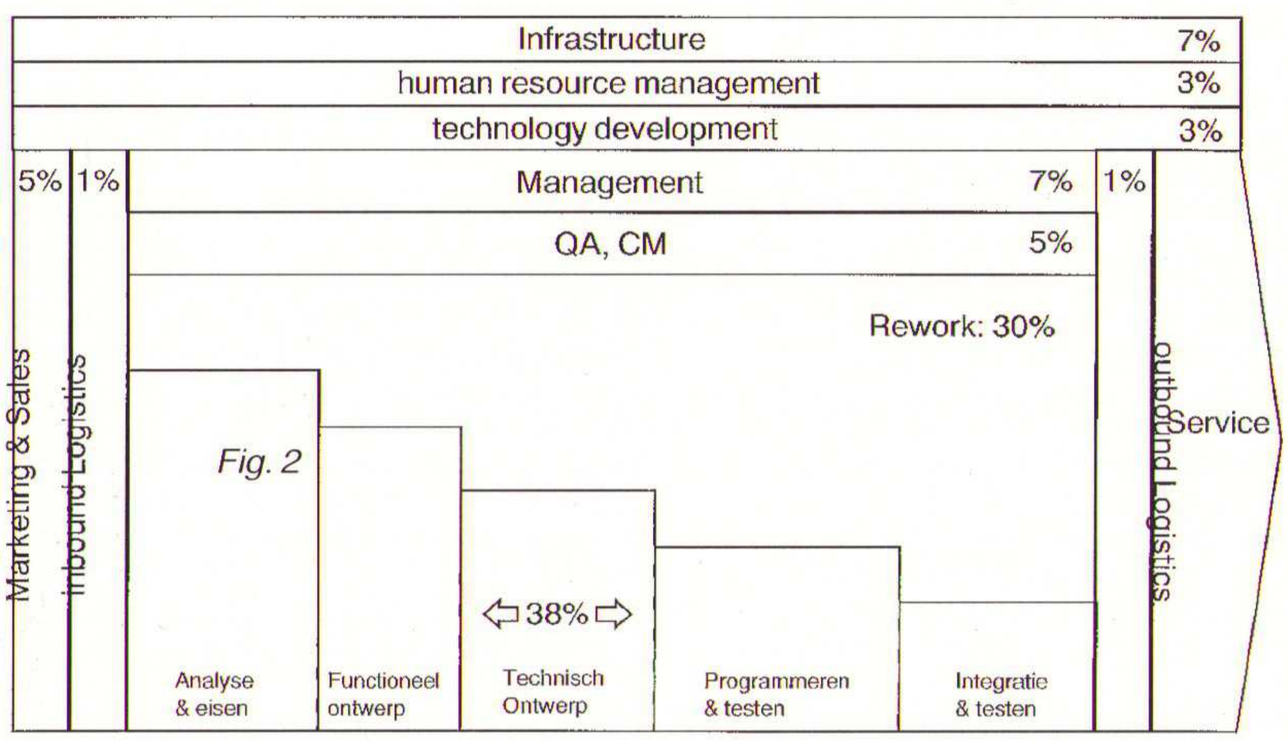


Fig. 2

Een ander gebied dat enige aandacht verdient is in dit schema benoemd als service. Deze activiteit, ook wel onderhoud genoemd, neemt over de gehele lifecycle van een produkt 70% van de totale kosten in beslag. We hebben er in dit schema geen percentage aan toegevoegd omdat de distributie van de kosten over de verschillende activiteiten in de onderhoudsfase nauwelijks afwijkt van de distributie van de weergegeven ontwikkelkosten verdeling.

Aan deze weergave van de verdeling van de inspanningen kunnen we de navolgende gevolgtrekkingen verbinden:
verbeteringen op het discipline niveau zijn van doorslaggevend belang
vrijwel alle componenten zijn zeer arbeidsintensief. Uitstekende kansen dus voor geautomatiseerde hulpmiddelen om de activiteiten efficiënter en kapitaalintensiever te maken. Bovendien wordt de kwaliteit van 'human resources' en het management steeds belangrijker.

In de organisatie zoals weergegeven in afbeelding 2 zijn naast de verdeling van de inspanningen, drie controleniveaus te herkennen. Het discipline-controleniveau, waar het software-ontwikkelproces plaatsvindt, het activiteiten-controleniveau, waar bijvoorbeeld project-management en Quality Assurance thuishoort, en het organisatie-controleniveau. In afbeelding 3 worden deze controleniveaus in een hiërarchische piramide weergegeven.

2.5 Controleniveaus

Discipline controle betreft alle activiteiten die te maken hebben met het vak Software Engineering: de vaardigheid om software (technisch) te kunnen ontwikkelen en te onderhouden. (Zie ook fig. 3) Zou in het model bijvoorbeeld een mechanisch ontwikkeltraject zijn opgenomen, dan zou enkel dit deel van het model er anders uitzien. Op dit controleniveau gaat de aandacht uit naar de Software Development Environment (SDE), moderne SE-methoden en werkprocedures en opleidingen.

Activiteiten controle betreft activiteiten gerelateerd aan het feit dat een project wordt uitge-

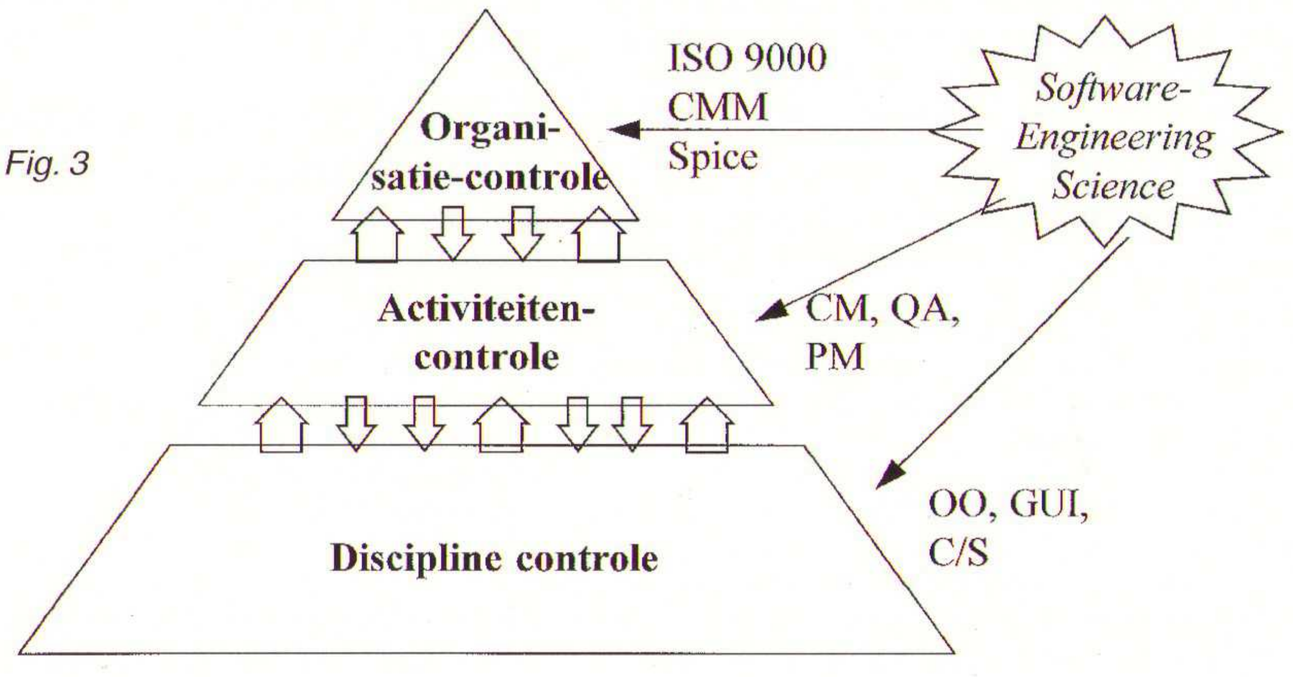


Fig. 3

voerd. Op dit controleniveau gaat de aandacht voornamelijk uit naar de invoer van project controle systemen zoals Project Management, Quality Assurance, Configuration Management en Change Control.

Organisatie controle geeft de voorwaarden aan voor de organisatie om software projecten te kunnen uitvoeren. De organisatie gedraagt zich conform een gedefinieerde manier van werken. Deze komt tegemoet aan de behoefte van de organisatie om gewenste wijzigingen gecontroleerd door te voeren.

SPI (Software Proces Improvement, van het Software Engineering Institute) en SPICE (een Europese variant ontwikkeld bij ISO) zijn werkmethoden die moeten resulteren in een efficiënt software ontwikkelproces met de volgende karakteristieken:
Voorspelbaar, kosten budgettering en planningen worden binnen de geboden marges gehaald en de eindprodukten voldoen aan de kwaliteitseisen van de gebruikers.

Ze vertegenwoordigen een stappenplan waarbij per stap de stand van zaken in kaart wordt gebracht en welke verbeteringsacties men moet nemen (Key Proces Area's) om naar het volgende hogere niveau te komen.

Centraal is het onder controle krijgen van het proces en vervolgens het verbeteren van het softwareontwik
kelproces.

2.6 Benaderingswijzen

We kunnen een onderscheid maken tussen twee gangbare benaderingswijzen om verbeteringen door te voeren:

Top-Down
Het investeren in SPI of soortgenoten is buitengewoon belangrijk. Maar deze veelal Top-Down gerichte benaderingen vergen veel tijd en veel geld. De tijd die SPI in beslag neemt brengt het gevaar met zich mee dat de motivatie afneemt en het initiatief uiteindelijk strandt. We moeten dan ook continu op zoek gaan naar onderdelen die snel resultaat opleveren. Dat is van doorslaggevend belang voor de motivatie van alle betrokkenen.

Bottom-Up
Heeft als nadeel dat initiatieven in de kiem worden gesmoord door het ontbreken van strategische budgetten, beschikbaar gesteld door het top-management. Hierdoor blijven verbeteringen meestal beperkt tot individuele engineers of 'eilandjes' in de organisatie.

Een meer kansrijke benadering is een combinatie van Top-down en Bottom-up: de werknemers op de drie controleniveaus hebben allen vanuit eigen taken, verantwoordelijkheden en bevoegdheden hun eisen en wensen. Bovendien worden alle niveaus vanuit de markt en de software engineering wetenschap gestuurd. De eisen en wensen van de verschillende niveaus dienen steeds dichter bij elkaar te worden gebracht waarbij de nadruk ligt op het discipline en activiteiten controleniveau. Onze belangrijkste gereedschappen hierbij zijn moderne softwaretechnieken en geautomatiseerde hulpmiddelen.

Welke kansen worden ons geboden om vanuit het discipline controleniveau, ook wel de Software Development Environment genoemd, verbeteringen door te voeren ? Voor de identificatie en de toekenning van de prioriteiten aan verbeteringskansen kunnen we gebruik maken van 'The value chain'-methode ontwikkeld door Porter en zijn associaties op Harvard.
Vervolgens zullen we ingaan op de mogelijkheden die beschikbaar zijn om verbeteringen aan te brengen in het software ontwikkelproces.

2.7 Kansen voor verbetering
2.7.1 Teamprestaties

De effectiviteit van een team wordt voornamelijk bepaald door de capaciteiten van het management en demogelijkheden om de werkzaamheden te sturen en te controleren. Belangrijk hierbij zijn:
- Duidelijke doelen voor, functie en taken van het team en ieder individu
- Goede communicatie
- Werkprocedures
- Leereffect

De sturing die aan het ontwikkelproces kan worden gegeven kunnen we afleiden van de eigenschappen van het software ontwikkelproces. Deze eigenschappen zijn o.a.:
- eenmalig karakter;
- vernieuwende elementen;
- duidelijk begin- en eindpunt;
- beperkte middelen (t d, geld);
- gericht op een of meer concrete resultaten;
- verscheidene betrokken deskundigheden;
- een geheel van onderscheidende activiteiten.

Dit zijn eigenschappen die aanduiden dat de meest geschikte wijze van beheersing de projectvorm is. We zien dan ook dat het ontwikkelen van software over het algemeen als project plaatsvindt.

Sturing van het ontwikkelproces
Als we de beheersvorm of de sturing die we aan projectmatig werken vergelijken met andere werkwijzen en hun eigenschappen dan kunnen we o.a. daarvan afleiden:
Sleutelwoord hierbij is: Methoden. Het gebruikmaken van projectmanagement methoden die aansluiten op onderliggende ontwikkelmethoden bieden ons de mogelijkheid het ontwikkelproces te sturen en te controleren.

Werkwijze	Sturing	Kenmerken
Ad hoc	Improviserend	Flexibele -lage kwaliteit
Projectmatig	Methodisch	'Gebonden flexibiliteit' - goede kwaliteit
Routinematig	Procedureel	Star - hoge kwaliteit

Onder methoden wordt verstaan:
Voorschrift dat aangeeft wat, wanneer en waarom gedaan moet worden. Een methode geeft dus richtlijnen die aangeven welke stappen gedaan moeten worden om tot een bepaald doel te komen. Let op, er wordt aangegeven 'welke' stappen gezet moeten worden. Op geen enkele wijze wordt aangegeven hoe men deze stappen moet zetten.

Wat we in dit overzicht ook zien zijn stromingen om van ad hoc richting procedureel werken te gaan. En als we ons voorstellen dat ontwikkelaars al de nodige weerstand bieden om methodisch te werken dan kunnen we ons voorstellen dat de volgende stap nog een graadje moeilijker zal zijn.

Voordelen van methodisch werken zijn:
1. Betere communicatie tussen ontwikkelaars en gebruikers. Voorlopig blijft de meeste programmatuur zo ingewikkeld dat zij nog door mensen moet worden ontwikkeld. Dan zijn goede communicatie en een gemeenschappelijke werkmethode cruciaal voor succes.
2. Projectmatige sturing wordt mogelijk waardoor grip op de voortgang en de kwaliteit van het eindresultaat wordt verkregen. (Daardoor minder rework, minder lines of code, minder onderhoud)
3. Biedt de mogelijkheid om na te gaan hoe het is gedaan en om de werkmethode te verbeteren.
4. Beperken unieke creaties. Het beperkt de mogelijkheid, zoals bij handmatig software ontwikkelen, dat men een unieke creatie krijgt waarvan vaak alleen de geniale ontwikkelaar ervan weet hoe het werkt.
5. Kennis en ervaring niet bij een persoon maar bij de organisatie.

Wel dienen methoden, of liever methode-systemen, aan bepaalde eisen te voldoen om effectief ingezet te kunnen worden.

Eisen gesteld aan methode-systemen:
1. Afdekken gehele Life Cycle
2. Voldoende Algemeen zijn om bij verschillende problemen toegepast te worden
3. Voldoende Specifiek zijn om bij verschillende problemen efficiënt en doelgericht toegepast te worden
4. Waar de ten grondslag liggende basis filosofieën zodanig op elkaar lijken dat tussen de softwareontwikkelfasen het denkproces niet hoeft te worden gewijzigd
5. Door gereedschappen ondersteunt waarmee zoveel mogelijk handelingen worden geautomatiseerd
6. Algemeen geaccepteerd zijn. Voldoende ervaringen, literatuur en opleidingsfaciliteiten.

Samenvattend: om de prestaties van het team te kunnen optimaliseren is het van belang dat er wordt gewerkt volgens een gedegen, en goed toepasbaar methode-systeem. Andere aandachtspunten hierbij zijn:
- Werving, juiste mensen op de juiste plek, zeker de manager, verwijderen van 'misfits'
- Faciliteiten
- Management getraind in praktische management technieken

2.7.2 Efficiency verhogen

In elke stap van het ontwikkelproces

We hebben reeds geconstateerd dat software-ontwikkeling arbeidsintensief is en zich uitstekend leent voor het gebruik van tools. Tools zijn bij uitstek geschikt om bij arbeidsintensieve en vaak routinematige werkzaamheden te ondersteunen. Om tools in een 'Integrated Computer Aided Software Engineering Environment' echt effectief in te kunnen zetten dienen ze aan bepaalde eisen te voldoen.

Eisen gesteld aan een Computer Aided Software Development Environment
Algemeen
1. Voldoen aan de eisen gesteld aan de methoden-systemen binnen de SDE
2. Een onderdeel van een geïntegreerde Software Engineering of Project Support Environment
3. Ondersteunen gedurende de gehele life cycle binnen alle ontwikkelfasen
4. Integratie tussen de verschillende ontwikkelfasen
5. Ondersteunen van multi-user en multi-project behoeften
6. Beschikken over een Centrale (project) Repository waarin alle ontwikkelgegevens centraal worden opgeslagen en beheerd
7. Een uniforme user interface en editors voor resp. binnen alle tools
8. Ondersteunen van meerdere computerplatformen en heterogene netwerken
9. Voorzien van een Application Programmers Interface
10. Ondersteuning Cliënt/Server applicatie ontwikkeling door integratie GUI en Database ontwikkeling
11. Gefaseerd invoeren van de tools

12. Evolutionair pad richting OT.

Technisch
1. Uitgebreide consistentie checks en traceability links
2. Ondersteuning van moderne process models, naast Waterval, evolutionair, incrementeel en Rapid prototyping modellen
3. Ondersteunen van geavanceerde softwaretechnieken (Information Hiding)
4. Ondersteunen van reverse engineering faciliteiten om de code consistent te houden met het ontwerp
5. Krachtige documentatiefaciliteiten. Koppeling tekstverwerkers en DTP hulpmiddelen
6. On line help systeem

Stappen elimineren
Een optimale efficiency is het elimineren van stappen. Voorbeelden uit het verleden hiervan zijn assemblers en compilers. Meer recent zijn software standaard-checkers, quality assurance functies, en analyse en design consistentie-checkers. Of: design-generation vanuit analyse, code (code structure generation), reverse engineering, documentation functions. Ook hierbij zijn tools onmisbaar. Inmiddels gaat ook steeds meer aandacht uit naar de generatie van code vanuit specs.

2.7.3 Elimineer 'Rework'

De belangrijkste post afgeleid van de verdeling van de inspanningen. 30% van alle resources worden hieraan besteed. Over de hele lifecycle zal dit wel oplopen tot 50%. Dit omdat rework kan worden verminderd of geëlimineerd door de kwaliteit van het ontwikkelwerk te verhogen.

Front-end aids
Computer aided requirement & design analyse tools kunnen rework inspanningen drastisch verminderen door betere visualisatie van software specificaties, meer formele en 'slanke' specificaties, geautomatiseerde consistency & completeness checking en geautomatiseerde traceability tussen de verschillende softwareontwikkelfasen.

Door gebruik te maken van moderne software ontwikkeltechnieken of 'bewezen ontwikkeltechnieken', kan rework worden voorkomen. Voorbeelden hiervan zijn: vroegtijdige verificatie en validatie, modulair ontwerp, top-down ontwikkeling, gestructureerd programmeren, walk-throughs of inspections, en software quality standaards. Een aanzienlijke winst in de onderhoudsfase van software kan worden behaald door technieken als 'information hiding' en 'data-encapsulation'.

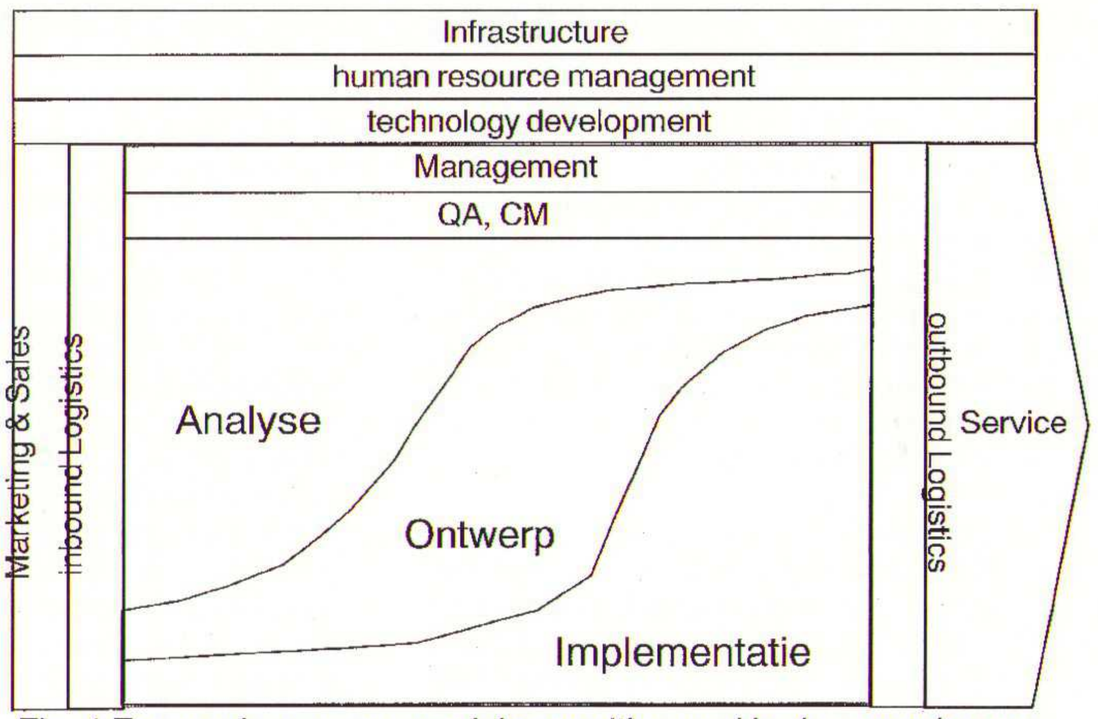


Fig. 4 Een modern procesmodel gepositioneerd in de omgeving van de organisatie.

Rapid Prototyping en Simulatie
Specificatie gebaseerd op een slechte vertaling van de wensen en eisen van de klant veroorzaken in de meeste gevallen het meeste rework. Moderne Rapid Prototyping en simulatietools kunnen deze situatie significant verbeteren.

Moderne procesmodellen
Het meest toegepaste procesmodel is nog steeds het 'Waterfall'-model. Indien toegepast in combinatie met doortimmerde front-end validatie activiteiten, kan deze document georiënteerde methode zeer effectief worden toegepast in de bestrijding van rework. Maar door de toenemende tijdsdruk en de onzekerheden in wat en hoe er ontwikkeld moet worden, wordt steeds vaker gebruik gemaakt van moderne procesmodellen
zoals het Evolutionaire model, en het Spiraal of het incrementele model.

Reduceren van het aantal Lines of Code
De grootste verbetering om het aantal 'Lines of Code' te reduceren kan worden gerealiseerd door het aantal te ontwikkelen programma instructies te verminderen. De twee belangrijkste opties worden geboden door:
- simpelere - niet mooier dan functioneel - produkten te bouwen.
- hergebruik van bestaande software componenten. Eigen en 'third party' bibliotheekroutines.

2.8 Conclusies

Door gebruik te maken van moderne softwaretechnieken en hulpmiddelen (op een kosteneffectieve manier) zijn kwaliteits- en produktiviteitsverbetering mogelijk. Bovendien kan de complexiteit beter in de hand worden gehouden.

De verbetering op het development niveau moet aansluiten op het Software Proces Improvement plan.
Sleutelwoorden:
SPI
Methodisch werken
I-CASE tool
Moderne software technieken

Carl Heskes
Bergson Technology B.V.
Tel: 040-2423414

Visie op de aanschaf van ontwikkel software met het oog op Cost of Ownership en trends in de markt

Inleiding
Om te begrijpen wat ons als hardware ontwerpers allemaal overkomt en hoe we naar de toekomst toe ons beleid moeten bepalen is het verstandig om te kijken of we trends kunnen ontdekken. In de totale globalisering van de industrie kunnen we heel duidelijk zien dat zowel voor de hardware- als de software-fabrikanten het verkrijgen van een groot produktievolume hun belangrijkste doelstelling is.

Aan de hardware kant kunnen we duidelijk zien, dat de silicon manufacturers zich storten op die produkten, die hun een enorm produktievolume kunnen brengen. Denk daarbij in eerste instantie aan geheugenchips, dan CPU's, special chips vanaf een common interface tot en met een PowerFET of een diode.

Ook de software bouwers gaan tegenwoordig voor het volume. In de operating system-markt zien we duidelijk een Microsoft Novell en een Sun Soft. Desktop tools, databases, CAD-systemen en de grafische industrie zijn volumemarkten of volumeprodukten op zich.

3.2 Desktop versus embedded

Qua karakter van de applicatie zouden we een onderscheid kunnen maken tussen open systemen en gesloten systemen. Dezelfde vergelijking is van toepassing of ligt parallel aan desktop versus embedded. Simpel gezegd zou je kunnen stellen dat een open system voorzien kan worden van software update en een gesloten of embedded system niet. Gek genoeg heeft dit niet alleen gevolgen voor de software, zoals we hadden mogen verwachten, maar ook voor het hardware-ontwerp.

Voorbeelden van desktop of open systemen zijn: onze administratieve computers, de PC's, de file servers, de telefooncentrales en andere.
Typische gesloten of embedded systemen zijn: de CD-speler, een inktjet printer, de GSM telefoon, de kopieermachine etc.
Als we qua hardware architectuur kijken, valt op dat we in de desktop of open systemen een grote interne strijd rondom de CPU-standaard zien. Het mag duidelijk zijn dat middels het enorme PC quotum Intel in feite deze strijd voortdurend wint. Echter, we zien een standaard CPU, een standaard chipset daar om heen, komend tot een standaard hardware platform.

In de embedded wereld zien we deze standaardisatie minder. Van de grote bekende chips zijn vele embedded derivaten gemaakt. Vaak zijn die zogenaamde single chip-versies beschikbaar gekomen. Een single chip bevat -naast de CPU-kern- ook alle input en output interfacing. Ook de desktop en de open systemen lijken zich op dit moment te standaardiseren rondom de PCI-bus. Middels deze busstruktuur kan men zijn specifieke computer interfaces eenvoudig toevoegen. In de embedded wereld zal PCI zeker zijn toepassing vinden. Hoewel we daar ook totaal andere strukturen zien, zoals het gebruik van field-bussen, modulaire back plane-systemen en local area networking als interconnectiestruktuur.

3.3 De factor 'Produktie-volume'

Het kiezen van de beste oplossing voor een bepaalde toepassing is afhankelijk, natuurlijk, van vele faktoren. Een essentiele faktor die meetelt is het volume waarop de nieuwe produktie gerealiseerd gaat worden. Om het denken eenvoudig te houden, kan het beste een logarythmische schaal gebruikt worden. Wordt het produkt slechts 1x geproduceerd, 10x, 100x, 1.000x, 10.000x of 100.000x per jaar. Afhankelijk van dit produktievolume zullen we allerlei keuzes moeten maken t.a.v. het gebruik van speciale hardware en speciale software. Het zal duidelijk zijn dat in de lagere volumes het gebruik van "off the shelf"- produkten zowel aan hardware als software kant voordelig zullen zijn. Nadeel is hierbij dat het tamelijk moeilijk is om het produkt een eigen gezicht te geven.
Als we het over EDA-tools hebben, dan zullen we ons moeten richten op de produktievolumes, ongeveer startend bij 50 stuks per jaar en eindigend in het zeer grote volume.

Van groot belang is het dat gebruikers inmiddels van een speciaal produkt een prestatie verwachten die vergeleken wordt met een volumegeproduceerd produkt. Omdat 'ons produkt' zich specifiek richt op een bepaald niche toepassingsgebied, verwacht men vaak zelfs nog een veel betere prestaties dan van het standaard produkt.

3.4 Nieuwe technieken en kostenbeheersing

Doordat het voor de chips-producenten van zo'n groot belang is om te gaan naar volume- produktie, merken we dat wij onze produktietechnieken aan moeten passen aan:
- surface mount PCB-technologie;
- chip-behuizingen als TAB, BGA etc.;
- beschikbaarheid van componenten;
- levensduur en prijs.

We zien steeds meer dat de interessante chips verpakt worden op een manier die uitermate geschikt is voor zeer hoge volume-produktietechnieken. In verband met de complexiteit van de chip en/of de kosten van verpakking besluit de toeleverancier vaak dat een oudere verpakkingstechnologie niet meer van toepassing is. Dit betekent dus voor het midden- en klein-bedrijf dat zij gebruik moeten maken van moderne produktietechnologiën. De meeste componenten leveren inmiddels veel hogere prestaties dan 10 of 20 jaar geleden. Vaak is dit bereikt door een grotere rekennauwkeurigheid, gecombineerd met een veel hogere kloksnelheid. Twee faktoren, die over het algemeen het ontwerp niet eenvoudiger maken. Om al deze steeds maar ingewikkelder wordende chips aan elkaar te kunnen koppelen, moet er ook steeds meer "glue logic" tussen. Gelukkig hebben onze volumejagers hier iets op bedacht in de vorm van de Field Programmable Gate Array. Deze zeer complexe, maar vrij program
meerbare hardware bouwsteen zorgt ervoor dat we in weinig ruimte de ingewikkelde glue logic kunnen maken. Helaas hebben we een aantal extra ontwikkeltools nodig om dit dan ook werkelijk te kunnen doen. De algemene trend die je kan waarnemen voor het midden- en kleinbedrijf en zelfs voor de grote bedrijven die produkten in kleinere series maken is dat de kosten t.b.v. al deze nieuwe technieken hoger uitvallen dan in het verleden. Het maken van een aantal prototypen SMD printen is aanzienlijk kostbaarder dan een dubbelzijdig doorgemetalli
seerde uitvoering.

Het kopen en onderhouden van de FPGA ontwikkeltools kost zeker meer geld dan vroeger. En als we het nodig hebben voor een iets groter volume om een eigen chip te ontwikkelen, dan zijn de kosten nog aanzienlijk hoger. Daarnaast hebben we steeds meer en duurder gereed-schap nodig om de debugging, het meten en het testen uit te voeren. Eigenlijk zou dit alles pleiten om terug te keren tot het maken van de wat simpeler elektronica. Helaas hebben ze dat in het Verre Oosten ook al ontdekt. En door de lagere salarissen aldaar zullen we daarmee hier weinig kansen meer hebben.

Om een goede kans op overleven te hebben en zeker geld te verdienen zullen we ons hier moeten bezighouden met up to date-technologie. We moeten moderne produktietechnieken toepassen en dat alles tegen de laagst mogelijke kosten. Hetgeen weer betekent, dat we met zo weinig mogelijk personeel zoveel mogelijk moeten zien te maken. Wat we maken moet zo ongeveer eeuwig meegaan, van een uiterst hoge kwaliteit zijn en zeker morgen beschikbaar zijn. In een poging deze bijna onmogelijke doelen te bereiken, kunnen we de ontwerpers van de elektronica niet meer alleen laten. In hun ongelijke strijd met deze uitdaging moeten ze geholpen worden door de elektronische design tools. Middels deze CAD-software wordt de ontwerper de nieuwe architect van het PCB. Naast het installeren van design tools is het van groot belang dat een kwaliteitssysteem op zijn plaats gebracht wordt. Belangrijk is hierbij, dat zowel aan de procedure, het proces als ook aan de mensen gedacht wordt. Het eindresultaat van de combinatie van de tools en het kwaliteitssysteem moet zijn dat de klant als allereerste en als allerbeste geholpen wordt.

3.5 De functie van de design tools
De elektronische design tools brengen hulpmiddelen om het ontwerp te modelleren en het ontwerp te simuleren. Als dit op enig abstractieniveau is uitgevoerd, kan vervolgens het geheel op microniveau herhaald worden.
Ten behoeve van de FPGA wordt een bijna vergelijkbaar soort modellering-, simulatie- en ontwerpsynthese uitgevoerd. Bij ASIC's is ditzelfde patroon opnieuw van toepassing, maar moeten we tenslotte ook nog testfaktoren genereren.
Ten behoeve van printed circuit boards hebben we tools nodig voor component placement, het sporenplan rooten, het genereren van produktiebestanden en het eventueel analyseren van hoogfrequent en warmtegedrag.
Tenslotte moet ons design getest worden en testfactorgeneratie is steeds vaker een onderdeel van de complete keten.
Belangrijk is dat de toolset ervoor zorgt, dat al deze losse aktiviteiten een onderlinge samenhang houden. Het beheersen van een complex ontwerp en het meerdere mensen laten samenwerken verschillende mensen aan één design stelt hoge eisen aan de consistentie van alle bovengenoemde aktiviteiten.

3.6 Winst door designtools

Kort samengevat zouden we kunnen stellen dat we voor veel meer geld aan tools ook veel meer kunnen dan 10 jaar terug. Helaas is het zo dat we met de huidige tools maar net mee kunnen voor wat betreft de markteisen, die aan de volumeprodukten gesteld worden. Met een gelijkgebleven designtijd van ongeveer drie tot negen maanden maken we inmiddels wel veel complexere produkten met een hogere snelheid en een lagere produktieprijs. Ondanks deze voordelen brengen de design tools ons zeker niet de hemel op aarde. We zullen het design nog steeds zelf moeten bedenken en we hebben steeds meer training nodig voor pas afgestudeerden.

Eenmaal de design tools op hun plaats, bieden ze voor een langere tijd rust in de organisatie. Door het creëeren van een eigen componenten database en het hergebruik van eerder uitgevoerd deelontwerp kunnen we het nodige geld besparen in de toekomst. Mede door dit hergebruik, maar ook door het algemene gedrag van de tools zelf maken we aanzienlijk minder fouten in onze designs dan vroeger. De toepassing van gestruktureerde ontwerptalen zoals VHDL helpen om de ontwerpdocumentatie op een aanzienlijk hoger plan te brengen. De overdracht van een ontwerp is daarmee op een meer formele basis uit te voeren. Het uitvoeren van onderhoud aan reeds gemaakte ontwerpen is veel eenvoudiger. Het doorvoeren van kleine wijzigingen brengt vrijwel geen risico met zich mee omdat een volledige testomgeving reeds gemodelleerd is. Het gebruik van abstracte be schrijvingstalen, zoals VHDL, zorgt ervoor dat onze glue logic ontwerpen technologie-onafhankelijk worden. Niet dat dezelfde VHDL-source onmiddellijk gesynthetiseerd kan worden naar een andere componentfamilie, maar de overgang is minder desastreus dan met tools die specifiek van speciale chipfabrikant zijn.

Algemeen kunnen we stellen dat de rijpheid en volwassenheid van de ontwerpen en het daarbij passende hergebruik de beste basis vormen voor het geld verdienen middels de design tools.

Siebren de Vries
Chess Engineering
Tel: 023-5317351

Design Automatisering: geloof en werkelijkheid

Inleiding
Ter gelegenheid van de door het Platform EDA/Software-ontwikkeltools van Het Instrument georganiseerde EDA Dag 1996, is aan de sectie Design Technology and Automation van Hollandse Signaalapparaten B.V. (Signaal), in Hengelo, gevraagd om haar visie op design-automatisering te geven.

Het zou te ver gaan om in dit artikel alle aspecten m.b.t. design technologie (d.i. de kennis en kunde die nodig is om het ontwerpproces voor een bepaald produkt, als functie van een bepaalde engineering discipline, adequaat te ondersteunen) en de bijbehorende automatisering, de revue te laten passeren.

Deze tekst zal zich beperken tot een toelichting van de automatiseringsbehoefte voor de ontwikkeling van diverse elektronica produkten, gezien in het licht van de technische- en defensiemarkt specifieke beperkingen, waar een bedrijf als Signaal mee te maken heeft. Deze behoefte staat geenszins model voor de behoeften in het Midden en Klein Bedrijf.

Naast de noodzaak voor het geloof in de toepassing van automatisering, wordt ook ingegaan op een meer kwantitatieve benadering, die mede bij het nemen van investeringsbeslissingen wordt gebruikt.

Besloten wordt met een aantal conclusies aangaande de objectivering van Cost of ownership en de ROI en met een aantal aanbevelingen, gezien vanuit de wensen van een eindgebruiker, om de kwaliteit van de door de EDA-industrie in het algemeen geleverde ondersteuning te verbeteren.

4.2 Hollandse Signaal-apparaten: een introductie

4.2.1 Het bedrijf

De geschiedenis van Hollandse Signaalapparaten B.V. gaat terug tot 1922. Het bedrijf hield zich in eerste instantie bezig met de ontwikkeling en produktie van mechanische vuurleidingssystemen. Via eigen onderzoek en ontwikkeling heeft na de oorlog de ontwikkeling van de radar technologie en van de command en control en elektronische vuurleidingsystemen, een grote vlucht genomen.

Er is zelfs een tijd geweest, dat Signaal haar eigen computerchips ontwierp, tesamen met de daarbij behorende instructiesets, operating systemen en software ontwikkelomgevingen. Daar is inmiddels verandering in gekomen door de beschikbaarheid van adequate processing technologieën die commerciëel verkrijgbaar zijn. Ook tooling en met name software tooling voor de ondersteuning van de verschillende engineering processen wordt al geruime tijd niet meer bij Signaal zelf ontwikkeld.

De elektronica ontwikkelingen bij Signaal omvatten een breed scala aan subdisciplines, lopende van microgolf antenne componenten, microgolf IC's, samengestelde componenten op dunne film, hoogspannings en laagspannings analoge elektronica, digitale multilayer PCB's met daarop DSP's en andere processoren, analoge- en gemengd analoge/digitale PCB's, ASIC's , FPGA's en PLD's, Interconnectie-technologie e.d.

Voor een meerderheid van deze disciplines beschikt Signaal over aansluitende eigen produktie- en montage processen en technieken.

In 1990 is Signaal, dat daarvoor onder de Philips-vlag voer, overgenomen door het Franse elektronica concern Thomson. Signaal behoort nu tot de defensie-tak van dit bedrijf: Thomson-CSF. De positie van Signaal is die van een resultaat verantwoordelijke business unit, met een produktenpakket dat is toegesneden op de Naval Combat Systems markt. Daarnaast is Signaal "Centre of Excellence for Radar systems".

De tijd dat de defensie industrie één van de drijvende krachten achter de verdere ontwikkeling van elektronische technologieën was, is voorbij. Dit betekent dat deze tak van industrie bijna volledig afhankelijk is geworden van wat er op de civiele en commerciële markt verkrijgbaar is.

Voor de situatie rondom ontwerptooling is dat enerzijds een voordeel: schaalvergroting leidt tot kostenreductie, maar anderzijds een nadeel: de meer geavanceerde en geintegreerde processen worden niet of nauwelijks door de bulk van de markt ondersteund en de vraag is dus (te) klein.

4.2.2 De produkten van Signaal

Als "Centre of Excellence for Radar systems" is Signaal een van de belangrijkste ontwikkelcentra van radarsystemen binnen de defensiepoot van Thomson. Radar systemen voor de korte en de lange afstand, voor gebruik als waarschuwingsysteem en voor het doelvolgen, uitgevoerd in diverse technologieën en uitgevoerd als totaal systeem, d.w.z. antenne, zender, ontvanger en de signaal en dataprocessing, dit alles behoort tot het produktenpakket. Ook de combinaties met optische sensoren behoren daartoe.

Daarnaast maakt Signaal vuurleidingssystemen. Dit zijn systemen waarin de sensoren gecombineerd worden met een stuk dreigingsevaluatie en resource-scheduling en met een interface naar één of meer afweersystemen. Die laasten kunnen bestaan uit iedere combinatie van kanonnen, raketten en torpedo's, die overigens niet door Signaal zelf gemaakt en geleverd worden.

Als derde onderdeel van het Naval Combat System, levert Signaal complete Command & Control systemen. Dit kunnen systemen zijn bestaande uit een simpele console met beperkte computerfaciliteiten, b.v. gecombineerd met een simpel richttoestel, tot volledig ingerichte commando centrales van fregatten en andere grote oppervlakte schepen. Ook kustverdedigingssystemen behoren hier overigens toe.

Op het civiele gebied ontplooit Signaal een aantal aktiviteiten in de sector Transport. Ook hier gaat het dan om command & control achtige systemen, gecombineerd met sensoren.

4.2.3 De produkt-karakteristieken vertaald naar procesbeperkingen

De markt van Signaal, die voor het grootste deel een export markt is, bestaat uit landen (overheden dus) met een marine. Het soort van produkten kan het best worden omschreven als zeer complexe professionele systemen, die in alle gevallen bestaan uit elektronica, software en mechanica onderdelen, die op perfecte wijze moeten 'passen'. De systemen worden geacht een hoge graad van betrouwbaarheid en inzetbaarheid te hebben en hebben een levensduur van minimaal 20 jaar. In die tijd moeten ze overigens niet alleen onderhouden kunnen worden, maar ze moeten ook nog wijzigingen kunnen ondergaan om ze aangepast te houden aan de veranderende omstandigheden en omgeving waarin ze gebruikt moeten worden.

Het aantal gelijke of vergelijkbare systemen dat een bedrijf als Signaal afzet is echter bijzonder klein in vergelijking met bijvoorbeeld de automotive, telecommunicatie of computer markten. En van deze genoemde markten heeft alleen de automotive-markt te maken met relatief lange levensduren (telefooncentrales even buiten beschouwing gelaten). De ontwerp- en produktieprocessen van Signaal zijn daarom aan wat stringentere beperkingen onderworpen dan in andere industrieën misschien gebruikelijk is.

Om te beginnen moet het ontwerpproces 'first-time-right' zijn. Dat wil niet zeggen dat elke handeling maar één keer mag worden verricht, maar dat de opeenvolging van handelingen, inclusief de voorziene iteraties, binnen de daarvoor geschatte tijd (en kosten) het gewenste produkt moet opleveren.

Door de relatief kleine aantallen wordt er eigenlijk voortdurend 'nieuw' ontworpen. Door de tijd echter, die er tussen het ontwerp van het ene radarsysteem en het andere zit, is de kans dat reeds eerder geïmplementeerde delen kunnen worden hergebruikt heel klein. In veel gevallen betekent de ontwikkeling van een nieuwe functie of een nieuw systeem, dat ook de te gebruiken implementatie-technologie weer gedeeltelijk vernieuwd moet worden. En vernieuwing van de technologie brengt een procesverandering (en in sommige gevallen een methodologie-verandering) met zich mee. In de nasleep daarvan zijn er ook altijd andere tools nodig.

De kostprijs van produkten is daarom in hoge mate NRE bepaald (d.i. Non Recurrent Engineering kosten) en dat vereist dat de ontwerpprocessen zo'n hoog mogelijke 'toegevoegde waarde' moeten opleveren. Dat kan alleen als ontwerpers zich kunnen concentreren op het vinden en verifiëren van de juiste oplossing en

dat zo snel mogelijk. Dit houdt in dat de ontwerpomgeving een goed geïntegreerde set van tools moet bevatten. De data- en de toolflow moeten zo transparant mogelijk worden gehouden. Ook aan de wijze waarop één en ander in het computernetwerk is geinstalleerd worden hoge eisen gesteld in termen van transparantie, beschikbaarheid, onderhoudbaarheid, gebruikersgemak, etc.

Omdat produktontwikkelingen een relatief lange looptijd hebben (gemiddeld tussen de 0.5 en 2 jaar), is ook de eis dat er twee volledige designomgevingen naast elkaar tegelijkertijd kunnen draaien. Daarbij moeten verschillende mensen tegelijkertijd aan hetzelfde produkt kunnen werken (bijvoorbeeld een ASIC of een board), zonder elkaar in de haren te vliegen. Vanwege de eigen produktiemogelijkheden vraagt de CAD/CAM interface extra aandacht. Maar die is zo ingericht dat zoveel mogelijk gebruik wordt gemaakt van standaardformaten.

Het feit dat produkten een lange levensduur hebben en gedurende die tijd toch wel wat bijstellingen moeten ondergaan, eist dat de essentiele designinformatie wordt opgeslagen in tool-onafhankelijke formaten, liefst werelds-tandaarden (b.v. VHDL).

4.3 Design Automatisering: het proces en de implementatie

4.3.1. Methodologie als uitgangspunt

Het aspect "geloof" dat gebruikt is in de titel van dit verhaal, heeft te maken met de uitgangspunten die ten grondslag liggen aan de automatiseringsinspanningen bij Signaal.

Het begint met het hebben van een "Designmethodologie", d.i. een afspraak over een bepaalde werkwijze, die kan worden gebruikt om zowel de plannen als de resultaten te toetsen. Deze methodologie is niet heilig. Ze kent haar beperkingen en als deze beperkingen teveel negatieve invloed hebben op het uiteindelijke resultaat van het "Product Creation Process" (PCP) dan moet de methodologie worden bijgesteld. Het PCP is overigens gedefinieerd als het Signaal-brede proces waarin kosten en inspanningen worden omgezet in produkten en verdiensten.

De Designmethodologie bij Signaal (welke in de afgelopen 3 jaar onderdeel is gaan uitmaken van een Thomson-CSF wijde werkwijze) kan als volgt in generieke termen worden omschreven:

Gebruik een gefaseerde procesbeschrijving
De fasering dient om een aantal bij elkaar horende activiteiten te groeperen en om enige notie van ordening en sequentie aan te brengen. Het proces is de kapstok voor de beheersbaarheid van de inzet van resources en de kwaliteit van de output.

Gebruik hierarchie
Hierarchie is een middel waarmee overzicht kan worden gecreëerd en dus de beheersbaarheid van het geheel kan worden ondersteund. Het wordt gebruikt om van grote problemen kleinere problemen te maken, die elk afzonderlijk van een oplossing kunnen worden voorzien.

Gebruik abstractie
Abstractie is een middel om van een globale omschrijving van een oplossing, via het gecontroleerd toevoegen van beperkingen op het aantal mogelijke vrijheidsgraden van het voorgestelde model, te komen tot een implementatie. Op de hogere niveau's is de beschrijving technologie-onafhankelijk, op de lagere niveau's wordt de keuze voor een bepaalde technologie een extra beperking in het model van de beschreven oplossing.

De juiste modelering van het designproces is dan afhankelijk van de methodologie en natuurlijk van de te gebruiken technologie. De werkwijze voor het ontwikkelen van een PCB is immers anders dan die voor het ontwikkelen van een ASIC.

Dat niet iedere werkwijze even gemakkelijk te ondersteunen is met de juiste tooling, kan echter niet genegeerd worden. Dit betekent zeker niet dat de toolkeuze voorop staat. Sterker nog, deze komt altijd achteraan. Maar het aantal tools is beperkt en dus is de verkrijgbare functionaliteit wel degelijk een beperking om mee rekening te houden.

4.3.2 De theorie

esign-automatisering kan worden gedefinieerd als het vakgebied dat zich richt op het 'mechaniseren' van handelingen, welke een onderdeel uit maken van ontwerpprocessen. Zoals met alle vormen van automatisering, levert deze mechanisatie alleen iets op als het proces goed bekend is en eenduidig beschreven kan worden.

Bij Signaal wordt onderscheid gemaakt tussen de proces-beschrijving en de beschrijving van de ontwerp-flow. Het proces wordt gedefinieerd als de te onderscheiden sets van aktiviteiten, welke kunnen worden gegroepeerd als opeenvolgende stadia (ook wel fasen genoemd) in de ontwikkeling van een produkt.
De designflow wordt gedefinieerd als de stroom van individuele aktiviteiten en de volgorde waarin ze moeten worden uitgevoerd. Eventuele iteraties tussen (meestal opeenvolgende) aktiviteiten zijn in de flow ook aangegeven.

Het automatiseren in technische zin heeft dan voornamelijk te maken met het ondersteunen, d.m.v. software gereedschappen, van de individuele aktiviteiten in de flow. De relatie met het proces wordt voornamelijk bewerkstelligd door eisen te stellen aan de input- en output formaten waarin de designdata wordt beschreven, met name op de fase-overgangen.

Het automatiseren in administratieve zin heeft voornamelijk te maken met het proces zelf. De automatisering staat hier niet zo zeer ten dienste van de mechanisatie van een designhandeling, maar van de beheersbaarheid van het proces en de controle op de voortgang.

In theorie kan elk goed bekend proces geautomatiseerd worden. De vraag of dat moet gebeuren, hangt af van de gevolgen van de automatisering voor het proces. Het betreft dan niet zozeer de gevolgen in financiële zin, maar veeleer de veranderingen die in het proces optreden als gevolg van de automatisering. Deze veranderingen hebben o.a. te maken met een totaal andere tijdsverdeling over de verschillende fasen en aktiviteiten en de veranderingen in benodigde skills die wordt geintroduceerd. Ook kan het proces buiten de methodologie geraken.

4.3.3 Enkele aspecten van de realiteit

In de industriële praktijk is er altijd sprake van een groot aantal beperkende factoren die de keuze van de middelen en zelfs de keuze van het inzetten van dergelijke middelen, beinvloeden.

Om te beginnen is er een verschil tussen de academische benadering van het ontwerpproces en de industriele werkelijkheid. In het eerste geval wordt het proces opgezet volgens een ideaal model en is de enige beperkende factor het ontwerp zelf. In het tweede geval echter zijn er allerlei beperkingen, al voorafgaande aan de eerste designstap, die beperkingen opleggen aan het proces en de implementatie ervan. Om er een paar te noemen: er is sprake van een beperkte componentkeuze, er zijn beperkte computerresources, de vaardigheid en kennis van de uitvoerende ontwerper laat op diverse gebieden te wensen over, de tools bevatten fouten en die zijn niet altijd overkomelijk (en soms niet eens vindbaar).

Een tweede probleem wordt gevormd door de budgetbeperkingen. Deze gelden voor zowel de aanschaf in kwaliteit en kwantiteit van de te gebruiken tools, alsook voor de eindige designtijd die gegeven is. Hoewel de designinfrastructuur bij Signaal op het proces gericht is en zodanig ingericht dat ze voor een groot gedeelte de behoeften van veelsoortige produktontwikkelingen afdekt, moet voor ieder project opnieuw een strategie worden bepaald om de bestaande mogelijkheden zo optimaal mogelijk te benutten.

De vaststelling dat de meerderheid van de elektronische ontwikkelingen NRE-gedreven is, vereist een 'first-time-right' benadering. Deze benadering vertaalt zich in het zo vroeg mogelijk en zo goed mogelijk elimineren van elk risiko dat een goede oplossing in de weg kan staan. Dit principe is ook van toepassing op de aanschaf en inrichting van de designomgevingen zelf. Anderzijds leidt het tot eisen aan tooling en integreerbaarheid van tooling die zeker niet marktconform te noemen zijn en dus niet standaard verkrijgbaar.

Weer een ander probleem vormt het behoud en toekomstig gebruik van de designdata. Deze data vormt de beschrijving van de oplossing en bevat dus alle toegevoegde waarde. Toegang op elk gewenst moment in de toekomst, tot de kennis (en dus de waarde) die in de oplossing verankerd ligt, is van groot belang. Voor een produktenpakket/markt combinatie zoals die van Signaal is hergebruik van kennis verreweg het meest van belang. Hergebruik van geimplementeerde functies en deelfuncties is, in deze tijd van snelle veranderingen inbijvoorbeeld het componenten-aanbod en de tooling, nauwelijks interessant.

4.4 Cost of Ownership en ROI: enkele voorbeelden

Met een tweetal voorbeeld berekeningen van Cost of Ownership en ROI zal geprobeerd worden het nut van deze berekeningen te bewijzen. Het laatste voorbeeld zal echter ook aantonen dat het ondoordacht accepteren van de berekenigen een gevaar in zich heeft. De genoemde bedragen geven alleen in relatieve zin de werkelijkheid weer; verder zijn ze gefingeerd.

4.4.1 Voorbeeld 1: Introductie Boundary Scan Test Methode

In 1994 ontstond er bij Signaal een grote behoefte om de bestaande printed circuit board (PCB) test methodologie te herzien. Hiervoor waren twee belangrijke technische argumenten te vinden:

Het schrijven van test vectoren op de huidige manier, voor een volledig functionele test via board-connectors en teststekers, wordt steeds moeilijker en tijdrovender, terwijl de kwaliteit van de vectoren steeds moeilijker te controleren is. De verwachting is dat voor toekomstige PCB's geen test vectoren meer met de hand geschreven kunnen worden binnen een redelijke tijd en met een acceptabele kwaliteit.

Het gebruik van componenten met steeds kleinere pitch noopt tot het gebruik van testpads. Deze testpads kosten ruimte op de PCB en kunnen voor distorsie van signalen zorgen. De verwachting is dat voor testpads op toekomstige boards geen ruimte meer is en dat toekomstige high-speed ontwerpen te gevoelig worden voor distorsie door testpads.

Ondanks de harde technische argumenten om Boundary Scan test methoden in te voeren is het van groot belang om een Cost of Ownership en ROI berekening te maken om de financiele consequenties boven water te halen. Dit zal de discussie over het wel of niet investeren duidelijker maken en kan zelfs in het geval van een snelle ROI de discussie versnellen.

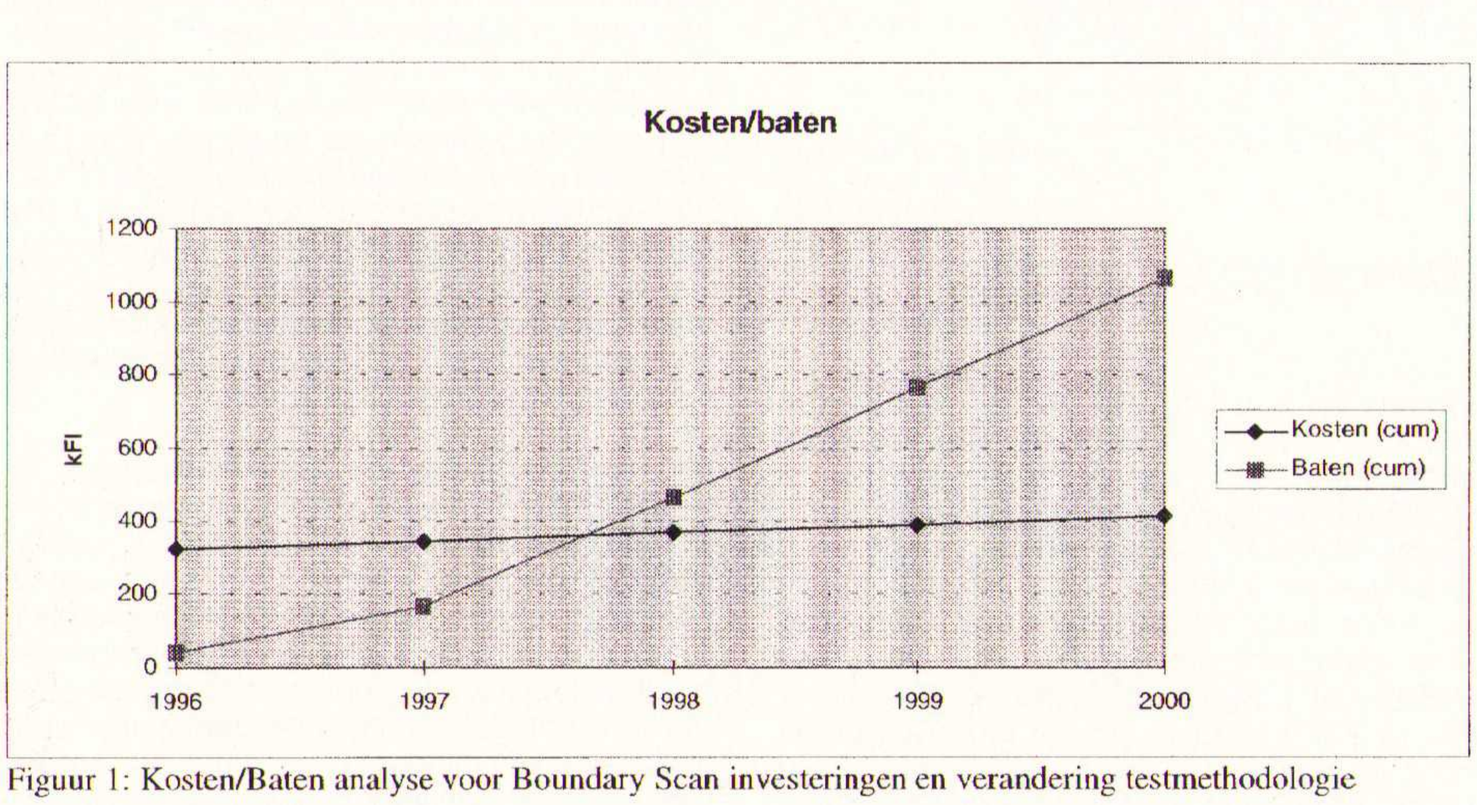
De kosten kunnen gesplitst worden in Initiele Kosten, b.v. evaluatie, aanschaf, en introductie van hardware en software en Maintenance Kosten, b.v. de jaarlijke maintenance kosten van de software. De geschatte kosten, die apart zijn weergegeven voor twee afdelingen binnen Signaal, te weten de Ontwikkelafdeling en de afdeling Produktie&Test, staan in tabel 1.

Tabel 1: Kosten		Initiele kosten	Initiele uren	Maint.kosten	Totaal (Hfl)
Ontwikkelafd.	1995	42.000	250	4.200	71.200
	1996 e.v.			4.200	42.200
Produktie&Test	1995	184.000	470	18.400	249.400
	1996 e.v.			18.400	18.400

De baten zijn geschat op basis van gegevens verstrekt door de ontwerpgroepen. Geschat zijn: het aantal nieuw te ontwikken PCB's met Boundary Scan in de komende jaren het aantal wijzigingen van PCB's met Boundary Scan in de komende jaren de besparing per nieuwe PCB in uren als gevolg van het gebruik van de Boundary Scan methode de besparing per gewijzigde PCB in uren als gevolg van het gebruik van de Boundary Scan methode.

De baten, weer onderverdeeld naar de twee afdelingen Ontwikkeling en Produktie&Test staan in tabel 2.

Tabel 2: Baten		aantal PCB's	Baten per PCB (uren)	Totaal (Hfl)
Ontwikkelafd.	1996	2	80	16.000
	1997	6	80	48.000
	1998 e.v.	13	80	104.000
		5 wijzigingen	50	25.000
Produktie&Test	1996	2	130	26.000
	1997	6	130	78.000
	1998 e.v.	13	130	169.000



Figuur 1: Kosten/Baten analyse voor Boundary Scan investeringen en verandering testmethodologie

Bovengenoemde cijfers zijn in figuur 1 nog eens in een grafiek weergegeven. Het is direkt duidelijk dat het omslagpunt ergens in 1998 ligt. Na het omslagpunt worden de baten hoger dan de kosten. De investeringen verdienen zich dus binnen drie jaar terug. Omdat verwacht wordt dat de komende 5 tot 10 jaren de Boundary Scan testmethode niet wezenlijk zal veranderen, is dit een acceptabele terugverdien periode.

4.4. Voorbeeld 2: Introductie Kwaliteitsanalyse tool

De verificatiekwaliteit van VHDL modellen van ASIC's, FPGA's, en PLD's, hangt sterk af van de kwaliteit van de gebruikte testbench. Zo is het van belang om te weten of alle delen van een VHDL model wel worden gesimuleerd door een testbench. Het valt daarom te verwachten dat een tool die de dekkingsgraad van een testbench voor een VHDL model kan geven, de kwaliteit van testbenches zal verhogen. Dit zal leiden tot kortere simulatie tijden, en dus een kortere design tijd, en minder re-designs als gevolg van niet opgemerkte design fouten.
Voor tot aanschaf kan worden besloten zal eerst een kosten/baten analyse moeten worden gemaakt om te bepalen of de kosten van het tool wel gedekt worden door de verwachte baten. Omdat dit een relatief nieuw automatiserings gebied is en de aangeboden tool nog volop in verandering is moet ook de terugverdien tijd niet te lang zijn. Een ROI van 1 jaar wordt als acceptabel beschouwd.

De kosten van het tool kunnen met de volgende formule worden weergegeven:

$$Kosten = Aanschaf + Introductie + \sum_{m=0}^{Maanden} m \cdot (Onderhoud + Ondersteuning \cdot Uurloon)$$

Waarbij:
Aanschaf = Aanschafprijs in guldens (= Fl. 30.000,-)
Introductie = Introductie kosten (opleiden gebruikers) (= Fl 3000,-)
Onderhoud = Tool maintenance in guldens per maand (= 10%/12 van de aanschafprijs)
Ondersteuning = Gebruikers ondersetuning in uren per maand (= 2 uur per maand)
Uurloon = Loonkosten in guldens per uur (= Fl. 110,-)
Maanden= Gebruiksmaanden van de tool

De baten kunnen met de volgende formule worden beschreven:

$$Kosten = Aanschaf + Introductie + \sum_{m=0}^{Maanden} m \cdot (Onderhoud + Ondersteuning \cdot Uurloon)$$

Waarbij:
Gebruik= Gebruiks percentage van de tool.
FPGAs = Aantal FPGAs per maand (= 2 per maand)
Besparing= Besparing in uren per FPGA (= 16 uur)
Redesign = Aantal uren t.b.v. een redesign van een FPGA (= 16 uur)
Foutkans = Percentage niet gevonden design fouten (= 10%)

Er van uitgaande dat het gebruik van de tool zal oplopen van 10% in de eerste maand naar 100% na 10 maanden, kan een kosten/baten analyse gemaakt worden. Het verloop van kosten en baten staan in een grafiek weergegeven in figuur 2. De ROI wordt bereikt na ongeveer 10 maanden, hetgeen acceptabel is.

Zou de tool echter Fl 60.000,- gulden hebben moeten kosten, dan ziet de Kosten/Baten aalyse eruit als in figuur 3. De ROI ligt nu veel te ver in de toekomst.

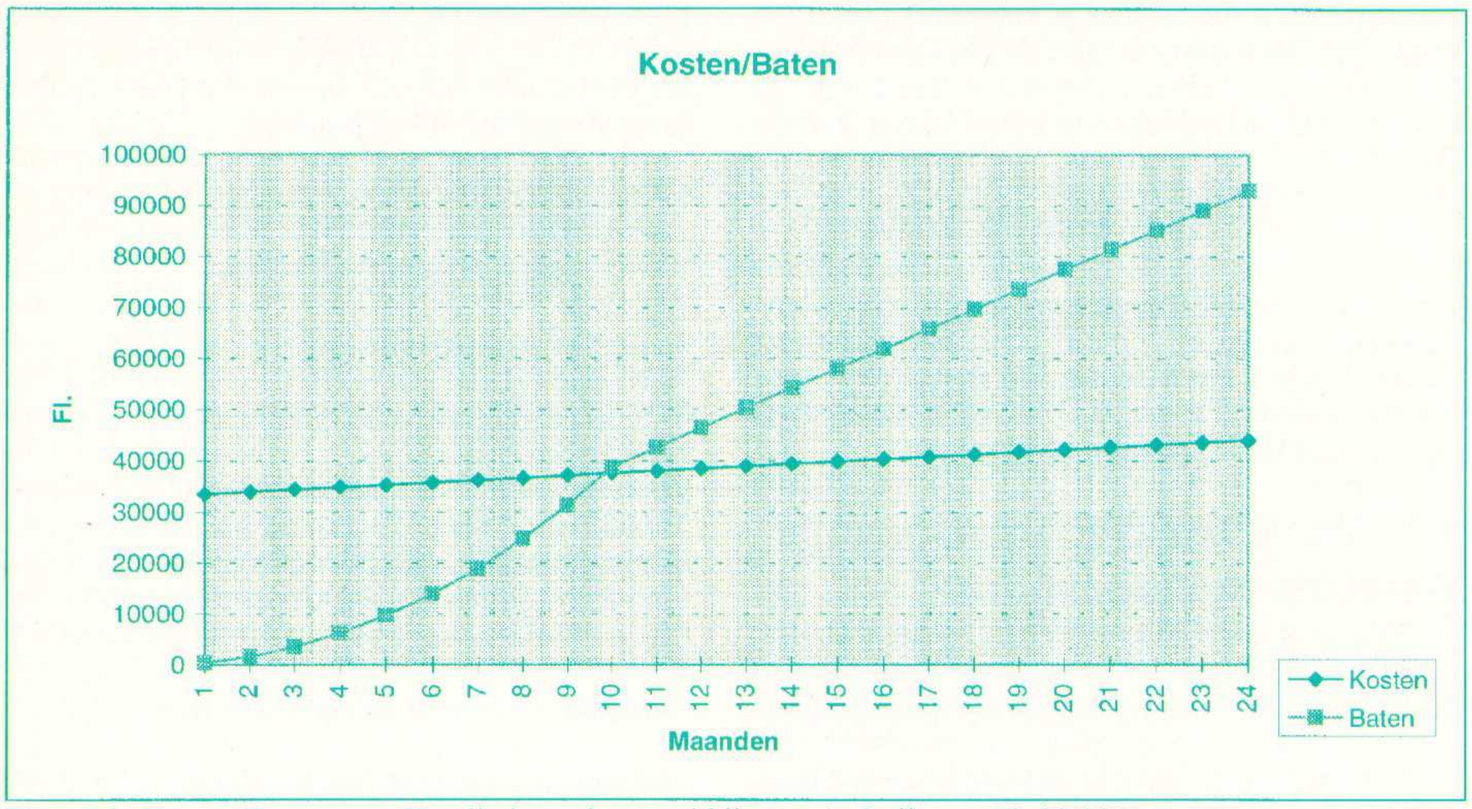
Een gevaar van dit soort Kosten/Baten analyses is dat het resultaat zeer kan afhangen van één of enkele input parameters. Als voorbeeld van dit gevaar is in de bovenstaande analyses de besparing gehalveerd. Het resultaat is zichtbaar in figuur 4. De terugverdientijd is vedubbeld!

4.5 Conclusies en aanbevelingen

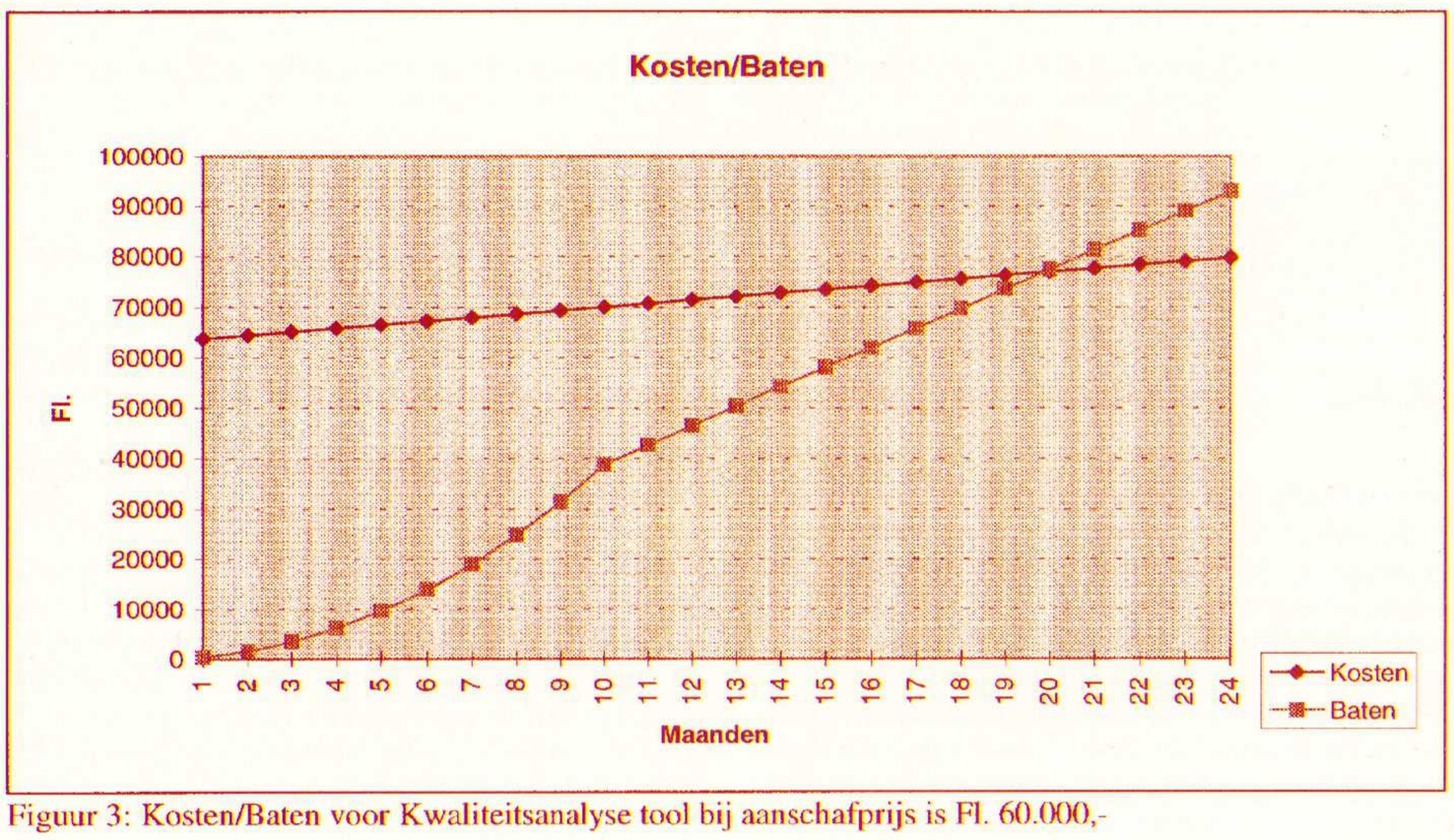
4.5.1 Cost of Ownership en ROI

Met betrekking tot de Cost of Ownership moet worden gesteld dat deze bij Signaal wel degelijk een rol speelt. Er wordt niet in alles geinvesteerd en de automatiserings-infrastructuur is primair proces gericht en niet project gericht. De beperking van het aantal toegestane componenten en technologieën en hergebruik van dezelfde methoden en technieken voor vele verschillende produkten in vele projecten, is noodzakelijk om de kosten i.v.m. investeringen, onderhoud en kennis, beheersbaar te houden.

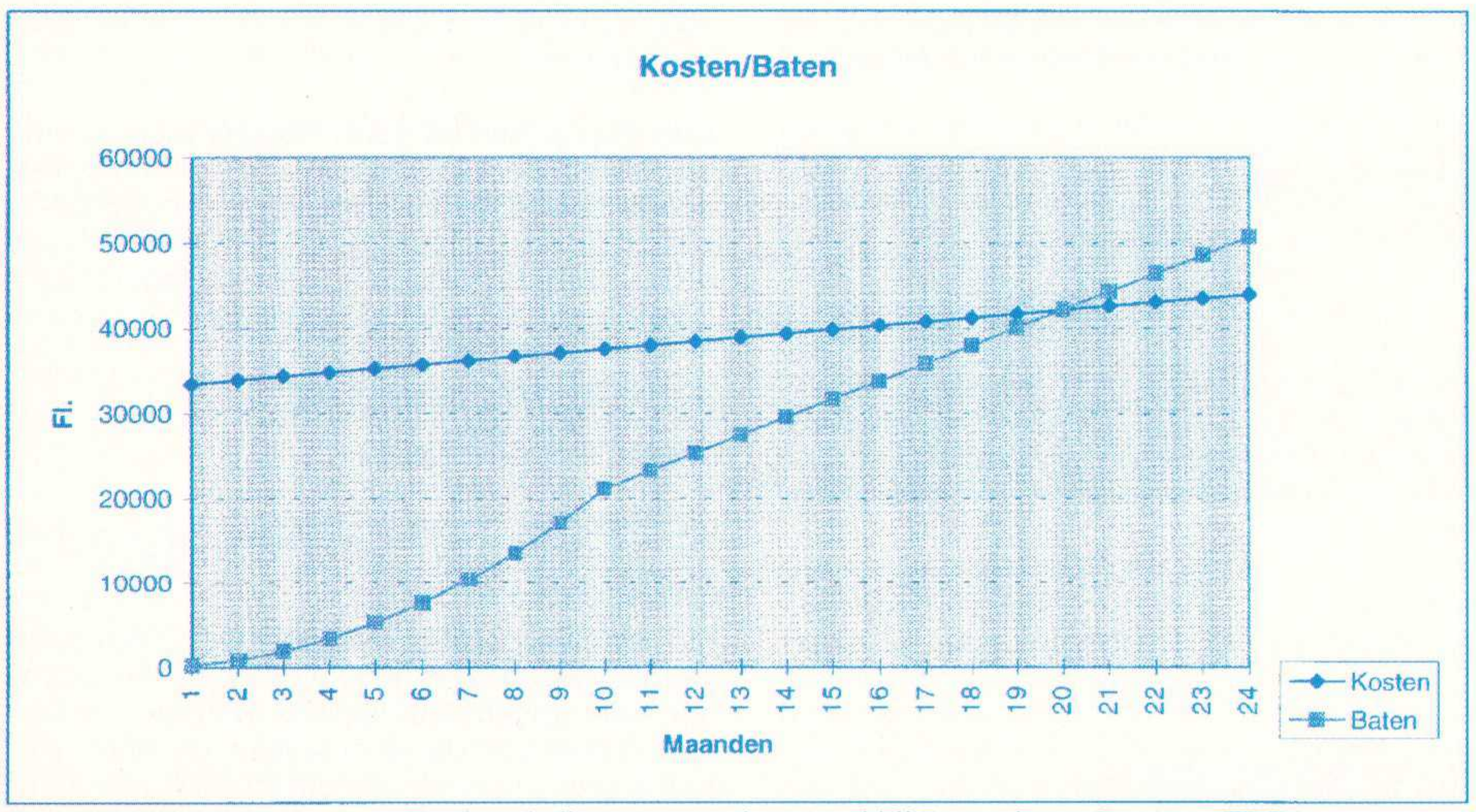
Dat betekent dat de aanschaf van middelen wordt bepaald aan de hand van verwachte of reeds noodzakelijk geachte verbeteringen in de ondersteuning van de ontwerpprocessen. Visie op en kennis van de EDA markt, van de eigen processen en van de gebruikte en te gebruiken technologieën, is daarbij onontbeerlijk. Een kwalitatieve en een kwantitatieve ROI berekening,



Figuur 2: Kosten/Baten voor Kwaliteitsanalyse tool bij aanschafprijs van Fl. 30.000,-



Figuur 3: Kosten/Baten voor Kwaliteitsanalyse tool bij aanschafprijs is Fl. 60.000,-



Figuur 4: Kosten/Baten voor Testbench Coverage Analyses tool bij Besparing = 8 uur per FPGA

ter ondersteuning van de op een jaarlijks bijgesteld automatiseringsplan en technologieplan gebaseerde investeringsvoorstellen, is voor de meerderheid van de studies en investeringsvoorstellen niet meer weg te denken uit het automatiseringsproces.

Het gebruik van bepaalde technologieën is voor de Signaalprodukten van strategisch belang. Toch moeten er keuzen worden gemaakt. Het aantal te ontwerpen produkten of deelprodukten moet groot genoeg zijn om een redelijke kennisopbouw en belading van de aan te schaffen tooling te waarborgen. De kosten moeten terugverdienbaar zijn, d.w.z. dat met nieuw aan te schaffen tooling een verbetering in het ontwerpproces moet kunnen worden ondersteund (t.o.v. de bestaande situatie), die het mogelijk maakt binnen de economische levensduur van het tool de investeringskosten terug te verdienen. Verder moet het mogelijk zijn de kosten die gemoeid zijn met de introductie en het gebruik van het tool over de genoemde periode, tesamen met de gebruikskosten van alle andere tooling, binnen de grenzen van de toegestane R&D deelbudgetten te houden.

Een heel belangrijke technische faktor bij de aanschaf van tooling is verder de mate waarin input- en output formaten van het tool zijn gestandaardiseerd. Het meest de voorkeur verdient het gebruik van internationale standaards. Indien niet voorhanden (of te prematuur) dan mag een de-facto standaard ook. De reden hier achter is simpel: in de data die met het tool 'verwerkt' wordt zit de toegevoegde waarde van Signaal. Opgeslagen in de 'proprietary' database van de geachte tool-leverancier is de retentietijd hiervan nauwelijks meer dan 3 jaar. Voor produkten met een levensduur van 20 jaar of meer is dit nauwelijks acceptabel.

4.5.2 Een oefening in gemeenschapszin

In 1990 is er binnen Thomson-CSF een aantal grote synergie programma's van start gegaan. Een daarvan richt zich op de design-technologie t.b.v. elektronica ontwikkeling. De gedachte achter dit programma is eenvoudig: indien de meer dan 30 Thomson bedrijven die elektronica ontwikkelen een gemeenschappelijke werkwijze en min of meer dezelfde tooling gebruiken, dan kan de produktiviteit omhoog gaan en de investeringen omlaag.

Eén van de principes hier achter is 'economy of size', d.w.z. dat je met 1000 ontwerpers sneller meer kennis kunt opbouwen en een betere marktpositie met scherpere prijzen kunt hanteren, dan met 100. Ook blijf je als bedrijf beter bij: er is altijd wel een project dat de grens van het kunnen en de mogelijkheden verlegt, en de fouten hoeven maar één keer te worden gemaakt.

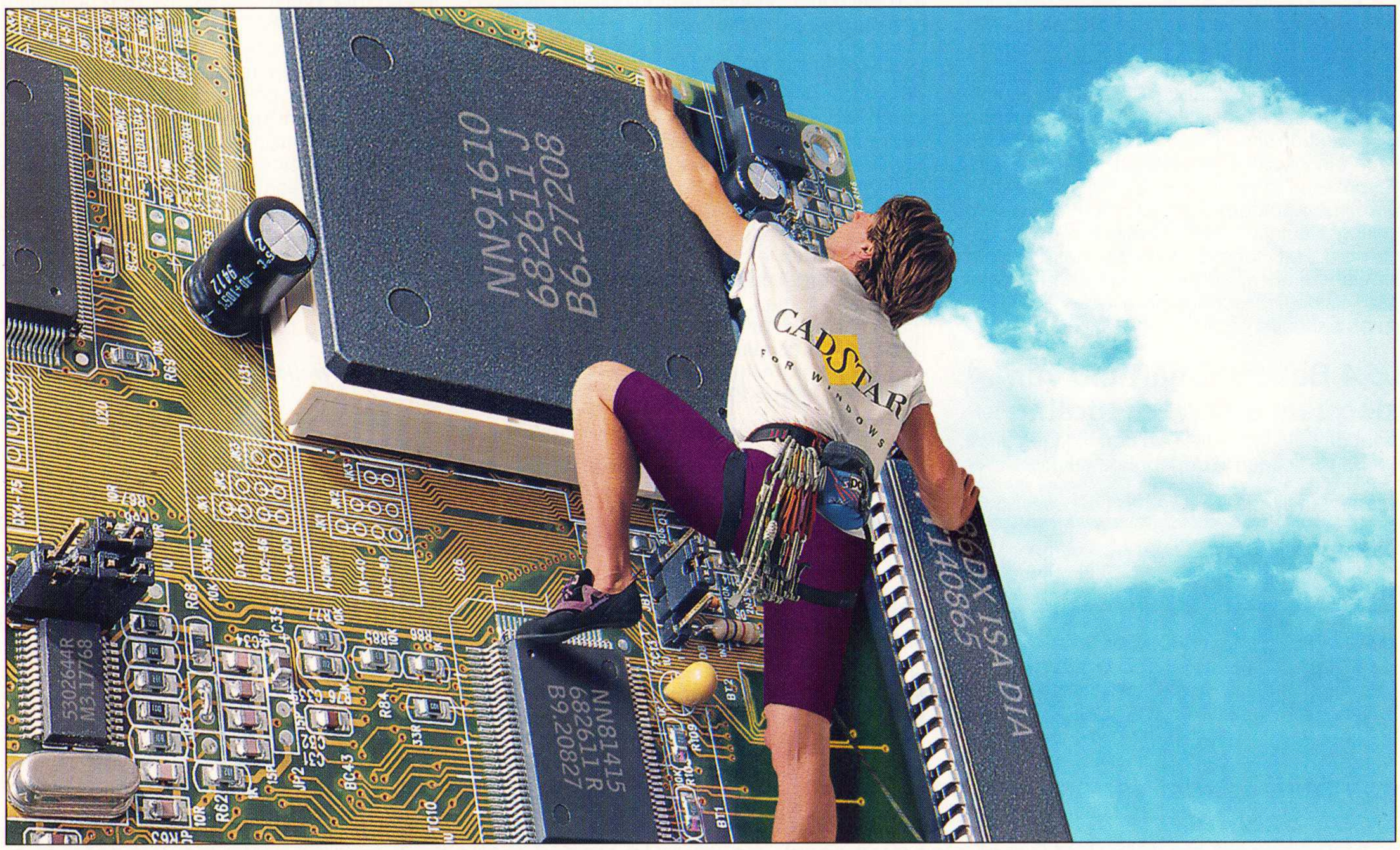
Het is natuurlijk niet gratis. Het vereist een inspanning om het met elkaar over een aantal zaken eens te worden, het vereist dat je verantwoordelijkheid voor elkaar neemt (daar waar dat je eigen belangen niet schaadt) en het vereist dat je je niet blind staart op de problemen van vandaag maar nu gaat werken aan de oplossing voor die van morgen. In een bedrijf dat uiteindelijk de resultaten van de business units voor 100% consolideert en als één naar buiten brengt, is dat al simpel. Toch is het ook voor onafhankelijke bedrijven niet onmogelijk (Het Instrument is daar tenslotte zelf een voorbeeld van).Het model staat ter beschikking (en de expertise van Signaal ook).

4.5.3 Een optimalere ondersteuning: betalen voor gebruik

Op dit moment heeft het begrip 'floating' licentie een tamelijk bekende klank in de EDA industrie. Toch is het nog niet zolang geleden (1992) dat het vragen naar deze mogelijkheid, om als klant een optimalere belading te bewerkstelligen, niet meer dan een verbaasde blik teweeg bracht. Ook nu nog wordt het door de tool-industrie veelal gezien als een mogelijkheid om de licentieprijs wat op te schroeven, dan een middel om samen met een uitgekiende produkt-samenstelling, een klant geld te besparen, dat in nieuwe functies geinvesteerd kan worden.

Een voorstel (gedachten-experiment?) van Signaal, aan één van haar grotere leveranciers, om over te gaan op een puntensysteem, waarbij elke tool eenmaal gekocht wordt en er vervolgens kan worden geinvesteerd in anonieme licentie-punten. Het gebruik van een tool kost dan run-time een x aantal punten uit de punten-pool. Via een inflatie-mechanisme voor verbeterde releases en de minder hoogdrempelige uitnodiging om ook nieuwe functies in de ontwikkel-

VEROVER UW CE MARKERING MET DE EMC DESIGN ADVISER



DEMONSTRATIEDAGEN OP 21 MEI, 29 MEI EN 6 JUNI

CADSTAR For Windows is het toonaangevende PCB ontwerp pakket van Zuken-Redac, de marktleider* in PCB/MCM ontwerp software.

CADSTAR For Windows Platinum biedt een ongelimiteerde en complete PCB ontwerp omgeving, inclusief schema tekenpakket, geavanceerde placement algoritmes en de unieke **gridless auto-interactieve routing mogelijkheden van** Route Editor.

MICROSOFT® WINDOWS™ COMPATIBLE 32-Bit Application

CADSTAR For Windows kan vanaf nu worden geleverd met de unieke **Design Adviser** en het **EMC Rulebook.** Deze software modules, die het resultaat zijn van een Europees onderzoeksproject, hebben hun waarde reeds bewezen in de Unix ontwerp omgeving van Zuken-Redac en zijn nu ook beschikbaar in CADSTAR op een PC platform. Kom kijken hoe EMC problemen in uw PCB ontwerp daadwerkelijk worden opgelost op onze

DEMONSTRATIEDAGEN

georganiseerd door Zuken-Redac en Koning en Hartman te Oosterhout.

DINSDAG 21 MEI

WOENSDAG 29 MEI en

DONDERDAG 6 JUNI

Bel Daphne Ploeg op onze 'Demo Hotline' voor uw gratis inschrijving op **telefoonnummer 0499-392020**, of schrijf u in per **email: Daphne_Ploeg@redac.nl**

Wij sturen u graag een programma en een routebeschrijving toe.

CADSTAR FOR WINDOWS

ZUKEN REDAC

ZUKEN-REDAC

World no.1 in PCB/MCM design

De Ronde 13, 5683 CZ Best. Tel: 0499-392020 Fax: 0499-392902

Windows® and Windows NT™ are trademarks of Microsoft Corporation. CADSTAR is a trademark of Zuken-Redac.

* Zuken-Redac heeft een wereldwijd marktaandeel in PCB/MCM ontwerp software van 21.7% (Bron: Dataquest, September 1995)

omgeving op te nemen, wordt een voor beide partijen goed beheersbare en optimalere situatie bereikt. Natuurlijk werkt dit mechanisme beter voor middelgrote en groot gebruikers, maar voor klein-gebruikers zou het geen bijzondere nadelen hoeven te hebben.

Een vergelijkbare situatie kan worden bereikt door in een bedrijf zoals Thomson de licenties te delen. Het feit dat het hier gaat om een groot aantal fysiek van elkaar verwijderde lokaties is nauwelijks bezwaarlijk als er sprake is van een intern elektronisch netwerk. De licenties worden centraal ingekocht en per maand 'verhuurd'. Vanzelfsprekend is de bron-code (of executable) lokaal geinstalleerd en behoeft alleen de 'floating' sleutel via het netwerk te worden geactiveerd. Het moge duidelijk zijn dat dit alleen kan werken als er sprake is van een redelijke mate van gemeenschappelijke tooling, werkwijzen en behoeften.

4.5.4 Beschikbaarheid van adequate software tooling

Nadat de EDA industrie zich een tijd lang heeft kunnen verheugen in ongebreidelde groei, is er sinds een aantal jaren een ombuiging zichtbaar. Verschillende bedrijven in deze tak van sport hebben al de nodige moeilijkheden achter de rug, al dan niet veroorzaakt door overnames. Maar ook deze bedrijven ondervinden de last van een steeds grotere veranderingssnelheid van de onderliggende technologieen, waarop geanticipeerd moet worden.

Een duidelijke trend is nu reeds zichtbaar: defensiebedrijven en andere grote industrieën zullen hun krachten weer gaan bundelen en een deel van de voor hun procesondersteuning zo noodzakelijke ontwikkelingen weer in eigen beheer gaan nemen. Thomson-CSF is hiervan een goed voorbeeld. In eerste instantie zal daarbij getracht worden de ontwikkeling bij de gereedschap-fabrikant te sturen. In tweede instantie zal men zelf weer een deel van de ontwikkeling ter hand gaan nemen, al dan niet d.m.v. een strategische alliantie met een software fabrikant. Op gebieden als componentenbeheer, modellen en modellibraries, methodologie gerelateerde tools en stukken van de tool-infrastruktuur is dit al gerealiseerd.

Ook over het aanpakken van de relatief grote achterstand die de nu op de markt gebrachte tools hebben op de in de universiteitslaboratoria ontwikkelde algoritmen, wordt binnen de eindgebruikers industrie opnieuw nagedacht. Ook deze industrie heeft immers toegang tot de universiteiten en in tegenstelling tot de EDA industrie ligt het kantelpunt van de ROI berekening daar veel dichterbij.

Het antwoord van de EDA industrie is er ook al: niet meer alleen de tooling, maar de dienstverlening rondom het gehele ontwerpproces wordt 'produkt'. Het valt niet te ontkennen dat met de snelheid waarmee de wereld nu verandert, de EDA industrie een plaatsje dicht bij het vuur inneemt en daarmee een lichte voorsprong kan opbouwen t.o.v. de eindgebruiker. Of ze nu al voldoende jokers in handen heeft, valt echter te betwijfelen.

C. Nieuwenhuis
E. Weening
Hollandse Signaalapparaten B.V.,
Hengelo
Tel: 074-2483288

Erop of eronder: de Scantium, een ASIC doe het gemaakt heeft

Scantech is een Nederlands bedrijf dat barcode scanners ontwikkelt, produceert en verkoopt. Scantech is in 1987 gestart in Heerenveen en is sinds 1988 gevestigd in Amersfoort. In 1990 werd de RIS2010 geïntroduceerd, de eerste vaste barcode scanner in de wereld die gebruik maakt van een laserdiode. Tot dan toe waren er alleen laserscanners met een HeNe-laserbuis op de markt.

Scantech levert zijn produkten aan PoS en computer partners over de hele wereld, die op hun beurt een totaal kassasysteem leveren aan eindgebruikers. Scantech is de enige Europese fabrikant van barcode scanners voor retail, verkoopt de scanners wereldwijd en is in Europa nr. 2. De concurrentie komt uit de Verenigde Staten.

Het huidige produkten pakket bevat diverse soorten barcode scanners:

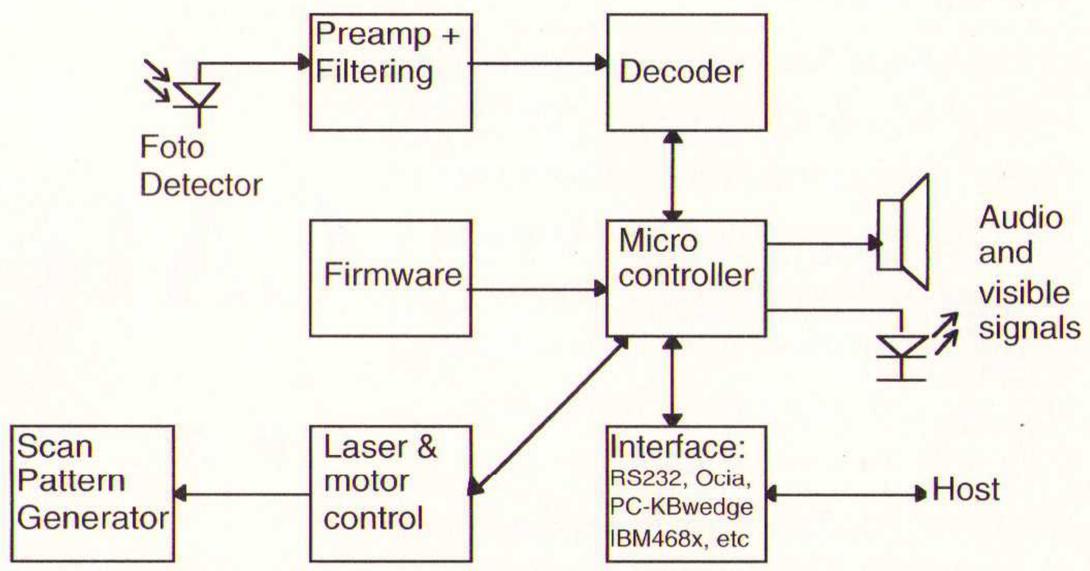
MT60/MT80	: handscanner
C2010	: retailscanner
Hunter	: presentation mode scanner
ScanGuide	: selfscanner/price checker
Pollux	: vertical scanner
Castor	: horizontal scanner
Gemini	: Catcher/Pitcher combination
PolyLine	: industrial linescanner

In de R&D afdeling wordt onderzoek en ontwikkeling uitgevoerd aan nieuwe barcode scanners. Scantech is momenteel ca 90 man groot. Daarvan zijn ca 17 man werkzaam in R&D (elektronica/software en opto/mechanica).

5.2 Hoe ziet een scanner eruit?

Een scanner bestaat uit een scanpatroon generator, fotodetector, analoge voorversterker, digitale decoder en een microcontroller (zie blokschema). Het scanning proces start met de generatie van een laserstraal door een low-power laserdiode(670 nm golflengte). Met een aantal vaste spiegels en een ronddraaiend polygoon (bestaande uit een aantal opgedampte spiegels) wordt een lijnen patroon gemaakt. Als een produkt met een barcode in het scanvolume van de scanner komt, zal de laserstraal de barcode kruisen. Het laserlicht dat wordt diffuus gereflecteerd door de barcode, wordt opgevangen via spiegels en een lens en wordt gefocusseerd op een fotodetector.
De opgewekte stroom in de fotodetector wordt versterkt en gefilterd. De output van de analoge voorversterker gaat naar een hardware decoder, die de data real time analyseert. De microcontroller zorgt voor de besturing, postprocessing (berekening van het controlegetal) en de communicatie naar de kassa.

Deze decoders kunnen parallel gezet worden om meerdere barcode families te decoderen.



5.3 Waarom is het Scantium project gestart?

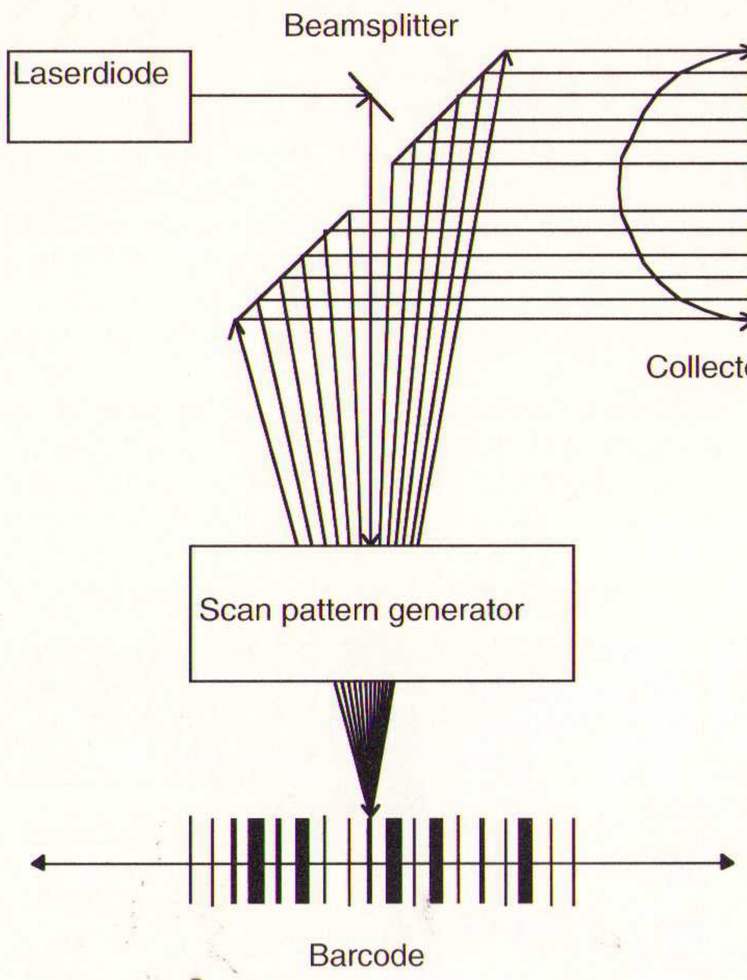
Er bestaan een aantal barcode families, die elk worden toegepast in specifieke markten. Meest bekend is de EAN8/EAN13 code, die in supermarkten wordt gebruikt. TNO-TPD heeft in 1988 voor Scantech 3 digitale decoder ASICs ontworpen:

MC1 : EAN/UPC codes (supermarkten)
MC2 : 2OF5/Interleaved/Matrix codes (industrie)
MC3 : C39/Codabar (apotheken, bibliotheken)

Echter vanuit de markt kwam een sterke behoefte naar C128/EAN128 codes. C128/EAN128 kan de volledige ASCII karakterset bevatten. C128/EAN128 wordt toegepast in detailhandel en distributie, omverpakkingen met daarop bijvoorbeeld gewicht, inhoud en houdbaarheidsdatum gecodeerd.

Het eerste idee was een MC4 te ontwerpen, die parallel aan een MC1/2/3 gezet kon worden. Dat kon relatief snel gerealiseerd worden, omdat het een voortzetting was van het MC1/2/3 concept.

Al snel werd er gedacht aan verdere integratie: 3 decoders simultaan in 1 ASIC: EAN/UPC, C128 en Wide/Narrow codes (C39, Codabar,



2OF5).
Hiervoor waren een aantal redenen:
Kostprijsverlaging (1 ASIC i.p.v. 2 of 3 parallel)
Ruimtewinst op de pcb (scanners worden steeds kleiner)
Mogelijkheid om enkele extra features te implementeren (fragment mode decoding)
Bij/voor blijven op de concurrentie.

Toen is besloten dit project zelf op te pakken, omdat tot dan toe de know how van deze voor Scantech cruciale technologie bij TNO-TPD zat. De behoefte ontstond om deze strategische kennis van het ontwerpen van decoders zelf op te bouwen. Er was echter binnen Scantech geen ervaring op het gebied van ASIC ontwerp en van decoder algoritmen.

Eind '92 is een globale planning gemaakt door Scantech, i.s.m. TNO-TPD, en een schatting van de kosten. Later heeft de ASIC een werknaam gekregen: de Scantium.

5.4 Hoe is de Scantium ontwikkeld?

Begin '93 is er gestart met een specificatie in gewoon Nederlands (10 pagina's). Daarna is er een C-model van de Scantium ontwikkeld. Input was digitale data, zoals die ook uit de analoge voorversterker komt. Output was een string met de barcode.

Er zijn 2 SUN sparc werkstations aangeschaft. Besloten is het ontwerp in VHDL te beschrijven, omdat deze taal een standaard is en de weg opent naar synthese. Door voor VHDL te kiezen dachten we minder afhankelijk te zijn van een specifieke ASIC-vendor. De VHDL bleek uiteindelijk toch vendor-specifieke constructies te bevatten.
VHDL vermindert het aantal fouten, doordat het ontwerp op een formele wijze beschreven wordt en niet in termen van implementatie. Bovendien kon al met het ontwerp gestart worden voordat de laatste onderhandelingen met ASIC-vendors waren afgesloten. De functionele VHDL simulaties zijn gebruikt om de consistentie met het C-model aan te tonen. Als VHDL simulator is Vsystem van Model Technology gebruikt.

Uit een lijst van 6 ASIC-vendors is uiteindelijk gekozen voor LSI LOGIC. Voor logische synthese vanuit VHDL is gekozen voor Locam-V (Vsyn, OMA) van Philips ED&T. Philips zou oorspronkelijk assisteren in het gate-level traject t/m sign-off. Later in het project is door Scantech alsnog een SUN werkstation van Arcobel gehuurd met daarop het CMDE pakket van LSI LOGIC. Het CMDE pakket is gebruikt voor de gate level simulaties, static timing analysis, floorplanning en DRC. Het ontwerp is synchroon en 100 % scantestbaar gemaakt met gebruikmaking van enkele trucs om de power dissipatie te beperken. Voor het automatisch genereren van de testpatronen is AMSAL van Philips ED&T gebruikt.

5.5 Welke proces keuze?

Tijdens het begin van het project is een schatting gemaakt van de benodigde hoeveelheid gates:
logica : 30.000, ram : 16250, totaal : 46250 gate equivalent.
Er is echter meer functionaliteit geïmplementeerd dan gepland.

Uiteindelijk was het volgende nodig:
logica : 42250, ram : 18850, totaal : 61100 gate equivalent.
De afmeting van de on-chip ram is: 512 x 8 bit (static, single port).
Als ASIC-proces is LCA100k van LSI LOGIC gekozen.

De redenen hiervoor waren:
de prijs, support van Arcobel en het feit dat met dit proces de gewenste performance gehaald zou worden.

5.6 Hoe was de projectstructuur?

Om de risico's zoveel mogelijk te beperken is besloten tot een samenwerking tussen Scantech, TNO-TPD, Arcobel en Philips ED&T.
De inbreng van de project partners was als volgt:
- Scantech (2 man) : systeemkennis, bouw van C-model, VHDL entry/simulatie
- TNO-TPD (1 man) : ervaring in ASIC design, decodeeralgoritmen, VHDL entry/simulatie
- Philips ED&T (1 man) : ervaring in ASIC design, support synthese, testbaarheid inbouwen
- Arcobel (1 man) : ervaring in ASIC design, CMDE, sign-off, contact met LSI LOGIC

Gedurende het project is er contact gehouden via vergaderingen, telefoon, fax en email.

5.7 Welke problemen kwamen naar boven?

Het startpunt was: nieuwe (point) tools, nieuwe Unix werkstation omgeving en weinig ervaring.
Het project heeft langer geduurd dan was verwacht. De start was eind '92 met de eerste specs op papier en in maart '95 waren we de eerste samples. De eerste planning - opgesteld door Scantech en TNO-TPD - ging uit van ca 1 jaar tot aan de synthese. Terugkijkend kan gezegd worden dat de complexiteit van het project in het begin onvoldoende is ingeschat. In het begin is door Arcobel een vuistregel genoemd van ca 200 gates/wk. Achteraf blijkt dit een realistisch getal te zijn.
Ten behoeve van het synthese tool moest er nogal wat gewijzigd worden in de stijl van de VHDL. Vervolgens zijn er problemen geweest met de synthese library en het daarin geïmplementeerde delay model. Dit was van directe invloed op het totaal aantal gates.

In de LCA100k technologie waren 2 die-sizes aangeboden. De grootste bleek echter niet in de aangeboden behuizing te passen. En we hadden de grootste wel nodig, aangezien het aantal gates groeide. Een grotere die-size had ook weer invloed op de synthese-library.

Bijna aan het einde van het ontwerp-traject werd door Arcobel het LCA300k proces aangeboden vanwege prijs en dissipatie reductie. We zaten toen al dicht bij sign-off. Toch is besloten om voor de LCA300k technologie te gaan.

Tot slot wilden we een powersimulatie doen om een schatting te doen van het gedissipeerde vermogen in de ASIC. Dit was mogelijk in de LCA100k technologie, maar dit werd helaas niet ondersteund in LCA300k.

Ondanks deze problemen en de toenemende planningsdruk zijn er geen concessies gedaan aan de testbaarheid en de simulaties.

5.8 Welke resultaten?

Het was een race tegen de klok. Het ging erom: Scantium of scant ie um niet? Maar uit de samples bleek: first time right! De performance van de Scantium was een belangrijke factor in dit project. De performance kan direct afgemeten worden aan de maximum klokfrequentie:
- Geschat in LCA100k technologie: 32 MHz
- Geschat in LCA300k technologie: 40 MHz
- Uiteindelijk in LCA300k : 48 MHz

Er bleek een klein foutje in te zitten, wat eenvoudig met 1 or poortje op het pcb op te lossen was. Geen reden voor een redesign dus! De Scantium wordt nu toegepast in alle nieuwe scanners (Hunter, Pollux, Castor, Gemini, PolyLine en Galaxyz.

5.9 Conclusie

Het Scantium project was van vitaal belang voor de concurrentie positie van Scantech.
De investeringen waren groter dan oorspronkelijk begroot (factor 1.7 x).
De ontwikkeltijd bleek ook langer dan gepland. Het is belangrijk om in tijdplanningen de factor 1.7 te introduceren om het enthousiasme te matchen met de realiteit.
Een project van deze omvang is haalbaar voor een klein bedrijf als Scantech, mits
- er een goed opleidingsniveau aanwezig is
- er samengewerkt wordt met anderen.
Er is ervaring opgebouwd in het bedrijf: VHDL, synchroon ontwerp, testbaarheid, planning, decodeeralgorithmen.
De Scantium is ongeveer even duur als een MC1.
De Scantium wordt nu toegepast in alle nieuwe scanners.
Scantech gaat nu starten met de ontwikkeling van een analoge ASIC.

Ronald Kemp
Hoofd R&D
Elektronica/software Scantech BV
tel.: +31 - 33 - 4698400
fax.: +31 - 33 - 4650615
email.: kemp@scantech.nl

Geïntegreerde ontwikkelomgeving voor real-time software

Inleiding

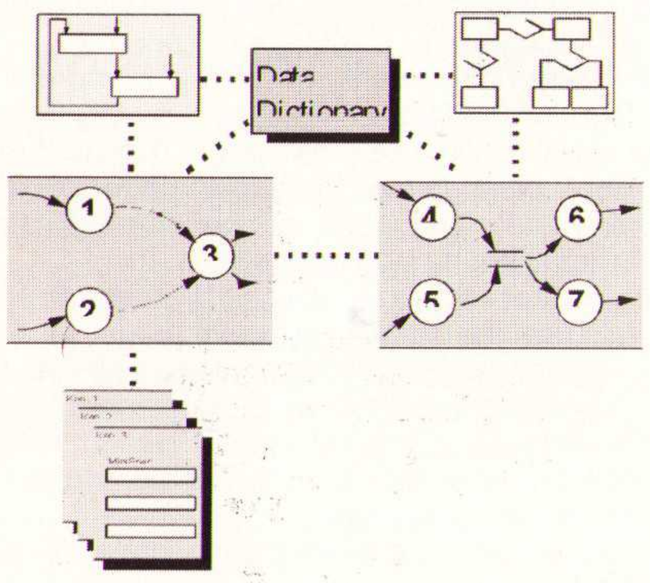
Het bedrijfsproces dat zich bezighoudt met ontwikkeling en onderhoud van software wordt geconfronteerd met een toenemende hoeveelheid werk. Dit is een algemene trend binnen het vakgebied software engineering. Een aantal factoren is hier debet aan:

- steeds meer functionaliteit wordt in software gerealiseerd,
- software wordt complexer,
- de life-cycle van produkten wordt korter, waardoor de productiviteit van software ontwikkelteams moet toenemen,
- de hoeveelheid code waar onderhoud op gepleegd moet worden neemt toe.

Voor het succes van een project zijn beheersing van de complexiteit, toename van de productiviteit en kwaliteitsborging essentieel. ProMod-PLUS is een CASE (Computer Aided Software Engineering)-tool dat is ontwikkeld om een bijdrage te leveren aan het behalen van deze doelstellingen.

Globaal doorloopt een software ontwikkeltraject de fasen analyse, ontwerp en implementatie. ProMod-PLUS ondersteunt algemeen geaccepteerde methoden en technieken voor al deze zijn:

- ondersteuning van alle gangbare platformen,
- software engineers kunnen alleen of in een team ProMod-PLUS gebruiken (single-user/multi-user),
- modulair opgebouwd, niet meer tools aanschaffen dan nodig,
- ondersteuning van zowel gestructureerde als object-georiënteerde methoden en technieken,
- openheid, gegevens uit de repository zijn toegankelijk en er kunnen vanuit ProMod-PLUS koppelingen gelegd worden met externe tools en bestanden.



Afbeelding 1 WI

ontwikkelfasen. Behalve voor de individuele ontwikkelfasen beschikt ProMod-PLUS ook over tools om de overgangen tussen de afzonderlijke fasen te ondersteunen. Er kunnen traceability links tussen modellen uit verschillende ontwikkelfasen worden gelegd. Op deze manier wordt de gehele life-cycle, van analyse tot en met implementatie, afgedekt. Vanwege de traceability links ontstaat een samenhangend model, waarbinnen naar behoefte kan worden genavigeerd. Dit levert een krachtige bijdrage tijdens de onderhoudsfase waarin wijzigingen plaatsvinden in het technisch of functioneel ontwerp. Ook het overdragen van kennis over de applicatie aan anderen wordt op deze manier vereenvoudigd.

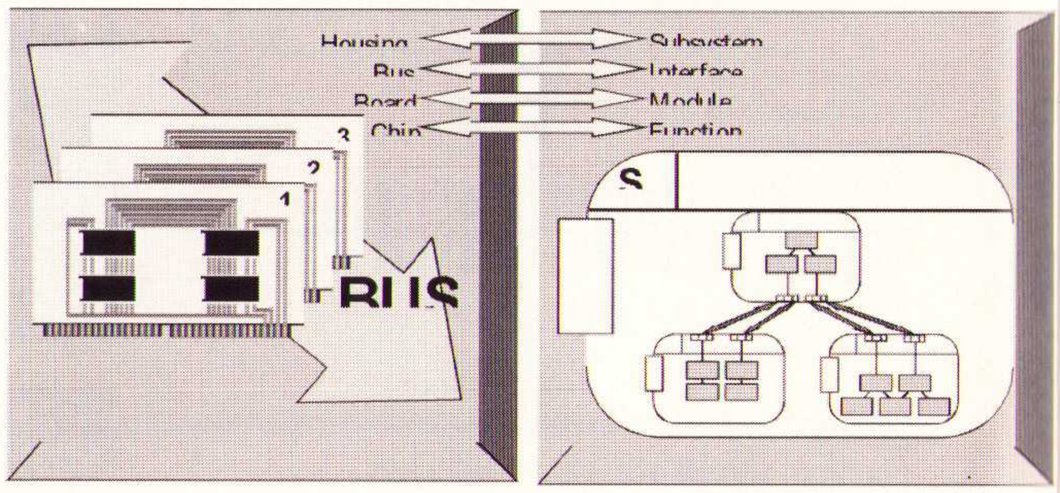
Naast de hierboven beschreven mogelijkheden beschikt ProMod-PLUS over een aantal intrinsieke eigenschappen waardoor investeringen in de Software Development Environment gewaarborgd blijven. Enkele van deze eigenschappen

Aan ProMod-PLUS, wat staat voor procesmodel, ligt het evolutionaire procesmodel ten grondslag. Dit betekent dat het doorlopen van een ontwikkeltraject niet op een voorgeschreven wijze hoeft te gebeuren. In een technische omgeving is deze manier van werken vaak wenselijk. Vanwege restricties op het gebied van tijd, geheugen en/of timing, wordt vaak in een vroeg stadium voor een (risico-)deel van de applicatie al een ontwerp- en implementatieslag uitgevoerd. Dit betekent dat voor delen van de applicatie het gehele traject is doorlopen, terwijl andere delen nog in de analysefase zijn. ProMod-PLUS is een tool dat is ontwikkeld voor en door de technische wereld en zodoende deze manier van werken ondersteunt.

In de volgende paragrafen wordt een globale beschrijving gegeven van enkele ProMod-PLUS tools. Vanwege de beperkte ruimte worden enkel analyse- en ontwerptools behandeld en het tool om consistentie tussen de resulterende analyse- en ontwerpmodellen aan te brengen en te beheersen.

1.2 De analyse-fase

Een informatiesysteem kan op drie wijzen worden benaderd: procesgeörienteerd, gedragsgeörienteerd en datageörienteerd. In het algemeen mag gesteld worden dat in technische omgevingen de gedrags- en procesgeörienteerde benadering van toepassing is (voor bestuurlijke informatiesystemen zijn dit de proces- en datageörienteerde benadering). Voor iedere benadering beschikt ProMod-PLUS over een apart tool. In onderstaande tekening zijn enkele diagrammen weergegeven van de verschillende analysetools. Omdat er tussen de verschillende modellen uit de analysefase overlap kan bestaan, wordt er met een centrale data dictionary gewerkt.

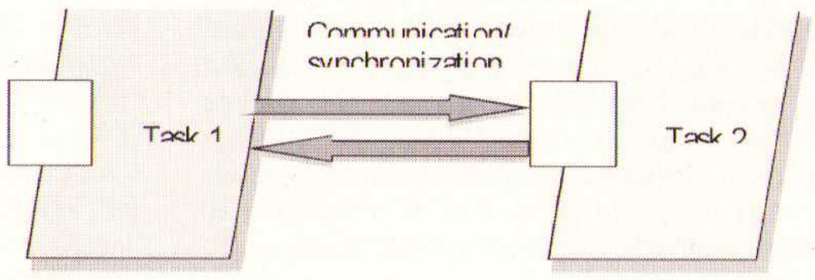


Afbeelding 2 WI

Vanuit een diagram uit het ene model kunnen eenvoudig diagrammen uit de andere modellen worden benaderd.Op deze wijze kan er door het complete analysemodel worden genavigeerd. De ontwikkeling van de afzonderlijke modellen kan door verschillende mensen gelijktijdig gebeuren.

1.3 Het statisch ontwerp

ProMod-PLUS beschikt over een tool om de statische structuur van de software vast te leggen. Op het laagste niveau gebeurt dit d.m.v. function calling trees. Bij grotere projecten met veel functies leidt dit niet tot overzichtelijkheid. Om de overzichtelijkheid, en dus de grip, op de structuur te vergroten heeft ProMod-PLUS hier een tweetal abstractiemechanismen aan toegevoegd, nl. modulen en subsystemen. Vertaald naar een 'C'-omgeving kunnen modulen worden vergeleken met .c files, waarin functions zijn on-



Afbeelding 3 WI

dergebracht. Subsystemen zijn equivalent aan subdirectories, welke uit modulen bestaan.

De nadruk bij het statisch ontwerp ligt op het vastleggen van de interfaces tussen componenten. Men moet hierbij denken aan functies, data, typen en constanten. Binnen het statisch ontwerp worden mechanismen als information hiding en data encapsulation ondersteund. Door goed van deze mechanismen gebruik te maken kan hergebruik, één van de voordelen van de objectgeörienteerde aanpak, ook binnen de gestructureerde manier van werken worden gerealiseerd.

In de elektronica zijn begrippen als information hiding, black boxes en hergebruik al lange tijd gemeengoed. In onderstaande tekening wordt een parallel getrokken tussen componenten uit de elektronica en het ProMod-PLUS tool waarmee het statisch ontwerp wordt vastgelegd.

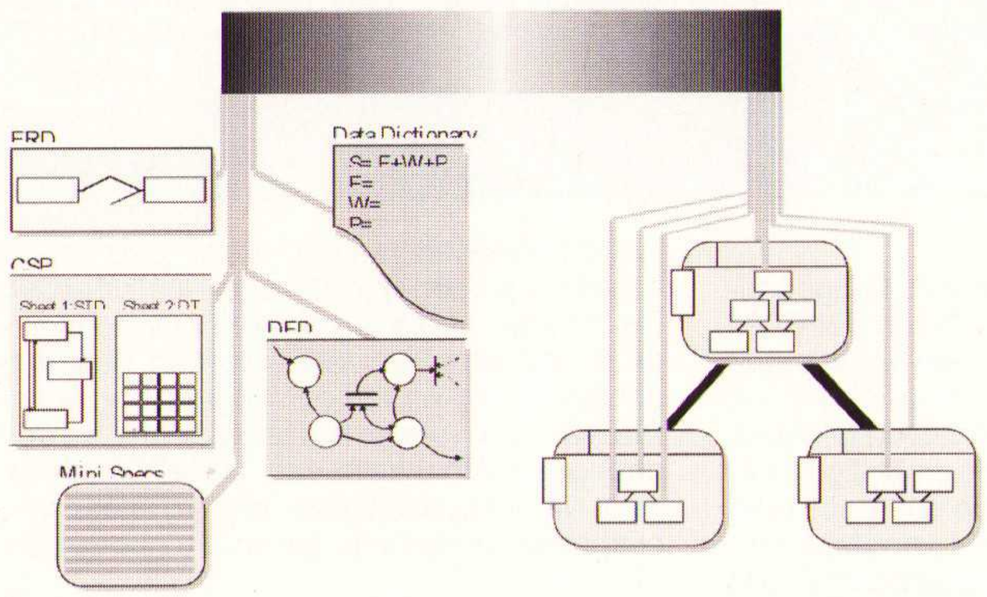
1.4 Het dynamisch ontwerp

Met het dynamisch ontwerp kunnen de run-time aspecten van een systeem worden vastgelegd. De code zoals deze in het statisch ontwerp is vastgelegd zal op een gegeven moment worden uitgevoerd. Het dynamisch ontwerp geeft hier inzicht in.

Binnen het dynamisch ontwerp worden taken (tasks) onderscheiden. Tasks stellen systeemdelen voor die parallel kunnen draaien. Tasks communiceren met elkaar via messages en met de buitenwereld via events. Tasks kunnen worden gebruikt om processen in een multitasking-omgeving weer te geven. De messages tussen de tasks stellen dan interprocescommunicatie mechanismen als messageboxes, queueus, pipes enz. voor. Een andere toepassing is dat tasks verschillende processoren voorstellen. De tasks stellen dan bv. verschillende processorboards, PLC's, PC's enzovoorts voor. De messages stellen weer de communicatie tussen de tasks voor (seriële lijn, VME-bus, ...).

1.5 De consistentie tussen modellen

Met het tool AutoTracer kunnen koppelingen worden gelegd tussen componenten uit de verschillende modellen. Op deze wijze kunnen



Afbeelding 4 WI

afhankelijkheden tussen componenten worden aangelegd, bewaakt en beheerd. Vanuit de AutoTracer kunnen de diverse modellen ook worden benaderd, waardoor het navigeren door de gehele applicatie mogelijk wordt. De configureerbare analyzer van ProMod-PLUS kan nagaan of er een volledige afdekking aanwezig is tussen componenten uit de verschillende ontwikkelfasen.

Voorbeeld: In de onderhoudsfase kan worden nagegaan in welke module (statisch ontwerp) bepaalde functionaliteit (analyse) is gerealiseerd en op welke processor (dynamisch ontwerp) deze uitgevoerd wordt.

Op onderstaande tekening zijn modellen uit de analysefase en de (statische) ontwerpfase te zien. Niet zichtbaar zijn de modellen uit de (dynamisch) ontwerpfase.

Wim Jansen
Bergson Technology BV
Tel: 040-2423414

Embedded Development Life Cycle.

Inleiding, de groei naar applicatiespecifieke processors
Door de steeds snellere veranderingen in de markt en de vraag naar geavanceerde produkten wordt er steeds vaker micro-elektronica toegepast bij het ontwikkelen van nieuwe produkten. Vooral micro-controllers, met sterk geïntegreerde functionaliteit, vinden hun weg in vele applicaties.

Traditioneel werden deze processoren in het verleden geprogrammeerd in 'assembly' doordat compilers te veel overhead genereerden en de daaruit voortvloeiende code bovendien vaak te traag was. Door de sterk verbeterde optimalisatie in de diverse hogere programmeer talen, zoals in eerste instantie PL/M en later C, stappen toch steeds meer bedrijven over naar een hogere programmeer taal. Vooral het gebruik van de taal C is over de laatste jaren sterk gestegen in populariteit en wordt nu niet alleen voor ontwikkelingen voor MS-DOS en Unix (native) maar ook voor embedded applicaties gebruikt. Diverse onderzoeken tonen aan dat ook C++ in populariteit stijgt en bij ontwikkelingen voor 32- en 64-bits processoren reeds vaak wordt toegepast.

Het grote voordeel van het ontwikkelen in C of C++ is dat reeds grote delen van de applicatie op het ontwikkelplatform zelf (PC of Unix) getest kunnen worden. Uiteindelijk moet de applicatie natuurlijk met behulp van een cross-compiler naar de juiste microprocessor code vertaald en daarna getest worden. Een ander voordeel van het gebruik van C/C++ is dat een applicatie eventueel snel naar een andere microprocessor architectuur overgezet kan worden. Vroeger ging men vaak uit van een processor die in voorgaande projecten werd toegepast.

Vandaag de dag wordt er eerst gekeken naar de te ontwikkelen applicatie en wordt er daarna een keuze gemaakt voor de processor die hier het beste op aansluit. De chip-fabrikanten spelen hier op in en introduceren processoren die voor een speciale verticale markt, b.v. automotive of telecommunicatie, bestemd zijn. Op deze manier wordt de investering die men heeft gedaan in het schrijven van software in een hogere programmeertaal niet geheel te niet gedaan en kunnen grote delen hergebruikt worden.

2.2 Debuggen, groei, naar VHDL-simulatie

Ook het debuggen van embedded applicaties is over de jaren sterk veranderd. Vroeger werd vaak de applicatie in een Eprom geprogrammeerd en gekeken of deze naar behoren werkte. Later werden hier Eprom emulatoren voor gebruikt om het steeds terugkerende programmeren van een Eprom te voorkomen. In de jaren 80 kwamen de eerste universele ontwikkelsystemen van o.a. Hewlett-Packard, Tektronix en Philips op de markt. Deze systemen bestonden uit een computer en daaraan gekoppeld een in-circuit emulator waarmee het mogelijk was de processor tijdelijk te vervangen, de applicatie in RAM te laden en de applicatie te executeren.

Door de hoge investerings kosten voor deze systemen ontstond er echter de behoefte aan eenvoudigere systemen die te koppelen waren aan de toen opkomende PC. Diverse fabrikanten brachten in-circuit emulators op de markt die te koppelen waren aan een willekeurige host zoals een PC. In het begin werd er nog op assembly niveau gedebugged maar naarmate de kwaliteit van de gegenereerde code van hogere programmeertalen groeide, werden de eerste Source level debuggers voor o.a. PL/M en C geïntroduceerd.

Met deze debuggers was het mogelijk de applicatie niet alleen op C, maar ook op assembly niveau te testen hierdoor werd het hele ontwikkelprocess sterk versneld. Traditioneel werden erin-circuit emulators gebruikt voor het debuggen van een applicatie maar door o.a. de vele derivaten die er van een bepaalde micro-controller op de markt komen wordt het steeds moeilijker voor de fabrikanten van deze in-circuit emulators al deze derivaten te ondersteunen. Mede ook door de toenemende frequentie waarop deze processoren werken en de steeds kleiner wordende behuizing, is het voor fabrikanten van in-circuit emulators haast onmogelijk een oplossing tegen acceptabele prijzen te bieden.

Omdat het ontwikkelen van software een steeds groter deel uitmaakt van het totale project, ontstaat er de behoefte in een vroeg stadium, als de hardware nog niet klaar is, de applicatie reeds te testen. Simulatoren brachten hier uitkomst en stelde een ontwikkelaar in staat reeds grote delen van de applicatie te testen. Helaas heeft een simulator het nadeel dat de applicatie slechts op een fractie van de uiteindelijke snelheid functioneert en dat problemen die gerelateerd zijn aan de hardware zich niet zullen openbaren. Zonodig moet ook de omgeving zoals I/O en interrupts gesimuleerd worden.

Een oplossing voor dit probleem is het gebruiken van een debugger op basis van het ROM monitor principe. Hierbij wordt een target monitor op het board geplaatst en vanaf de PC via een seriële poort de applicatie in RAM geladen. Op deze manier heeft men de volledige controle over het systeem en kan de applicatie real-time getest worden. Het nadeel van deze methode is dat de monitor enige resources gebruikt van het target board. In de praktijk wordt deze oplossing en die met een simulator dan ook gebruikt naast een in-circuit emulator die op dit punt geen beperkingen oplegt. Ook de chip-fabrikanten hebben deze problemen zien aankomen en hebben debug voorzieningen aangebracht bij het ontwerp van de chip. Het meest bekend is hier de Back ground Debug Mode of kortweg BDM van Motorola. Deze eenvoudige Debug interface werd geïntroduceerd met het op de markt brengen van de nieuwe 68300 familie en maakt het mogelijk via een speciale connector op het board de applicatie te kunnen laden, te steppen en breekpunten te kunnen zetten. Verder kan men via deze BDM interface de applicatie real-time executeren en het gedrag van de applicatie bekijken. Het enige nadeel van deze methode t.o.v. het toepassen van een in-circuit emulator is dat het geen "real-time trace" faciliteit heeft. Daarom werkt Motorola nu samen met enkele fabrikanten aan BDM+ waarbij dit wel mogelijk is. Ook andere chip-fabrikanten werken aan soortgelijke technieken. Zo wordt ook de JTAG interface, oorspronkelijk ontwikkeld voor boundary scan, vaak voor debug doeleinden "misbruikt".

Zeker bij high-end processoren zoals PowerPC wordt de snelheid en de "probing" een probleem en zijn ook fabrikanten van logic analyzers gaan zoeken naar een alternatieve oplossing. Door het mogelijk te maken een absolute file inclusief alle symbolische informatie te laden in een logic analyzer, kan er nu een referentie gelegd worden tussen de informatie die is opgeslagen in het trace geheugen en de oorspronkelijke source files. Op deze manier kan het trace geheugen nu in C worden weergegeven en kan men exacte informatie krijgen over de tijd die nodig is om b.v. een C statement of functie uit te voeren. Let wel dat ook deze oplossing zijn beperking heeft bij o.a. micro-controllers en processoren met caching omdat niet alle informatie naar buiten komt.

Recente ontwikkelingen geven aan dat er zelfs naar een nog hoger abstractie niveau wordt gekeken. Hierbij wordt een source level debugger gekoppeld aan een VHDL simulator. Deze simulator kan de ontworpen chip op hardware niveau volledig simuleren. Op deze manier kan de gegenereerde code geëxecuteerd worden op een VHDL beschrijving van de nieuwe processor.Hierdoor kan reeds in een vroeg stadium gekeken worden naar de performance van de chip en deze zonodig worden bijgesteld. Deze manier van ontwikkelen is door de hoge kosten echter niet voor iedereen weggelegd.

2.2 Conclusie

Het ontwikkelen van een embedded applicatie bestaat voor een steeds groter deel uit software. De tijd die nodig is voor het ontwikkelen van deze software kan sterk worden bekort door gebruik te maken van een hogere programmeer taal en de juiste hulpmiddelen voor het debuggen. Afhankelijk van uw situatie, de applicatie en het beschikbare budget, is één van de besproken mogelijkheden voor u een oplossing.

Kees van der Valk
Tasking Software Nederland BV
Tel: 033-4558584

Software Quality Assurance voor embedded real-time software

Introductie: het streven naar verbetering van de kwaliteit van software
De stimulans voor het verbeteren van de systemen voor Software Quality Assurance heeft verschillende achtergronden. De eindgebruikers van de software verwachten belangrijke verbeteringen in de kwaliteit en de betrouwbaarheid van de produkten die ze inkopen. Kwaliteitsmanagers zoeken naar een hanteerbare referentie waartegen de kwaliteit van software kan worden afgezet en binnen de grotere bedrijven onderzoeken juridische afdelingen wat de gevolgen zijn van de produktaansprakelijkheid ten aanzien van fouten in de toegepaste software. Daarbij zien de R & D managers zich geconfronteerd met een streven naar een kortere 'Time-To-Market' waarbij de software een steeds groter deel van het uiteindelijke produkt vormt.

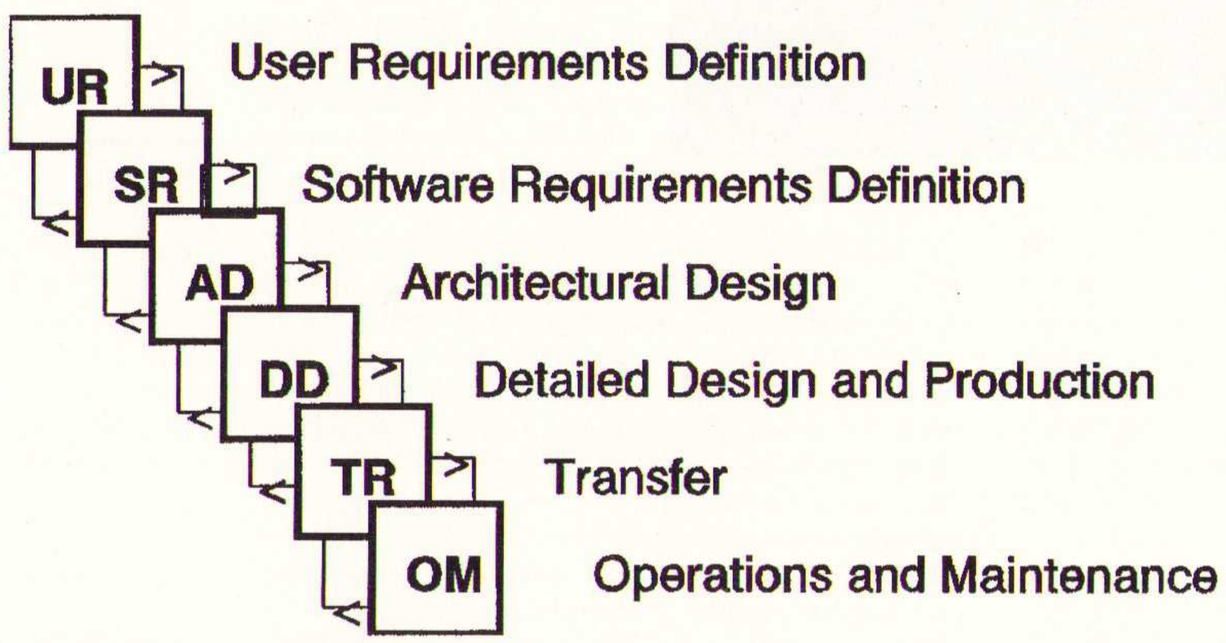
Een groot aantal bedrijven heeft zich de afgelopen jaren ingespannen voor het opzetten van een gedefinieerd en gecontroleerd kwaliteitssysteem als ISO 9000. Ook de software ontwikkelgroepen zullen hiervoor de juiste methoden, technieken en gereedschappen selecteren en implementeren om aan te kunnen tonen dat ook de software adequaat getest is en voldoet aan de standaarden van het bedrijf en de afnemers. In dit artikel willen we een aantal bestaande standaarden en methoden noemen, en ingaan op de problemen die hierbij specifiek spelen in een embedded omgeving waarin stringente eisen aan het real-time gedrag gesteld worden. Tevens zullen enkele systemen geïntroduceerd worden die hierbij hun diensten kunnen bewijzen.

2 Het Software Life-Cycle Model.

Het Software Life-Cycle Model beschrijft in een blokdiagram de verschillende stadia in het proces dat gebruikt wordt voor het definiëren, produceren en onderhouden van software. Alle kwaliteitsstandaarden voor software werken aan de hand van een model dat gebruikt wordt als basis voor het softwareproject, waarvan sommige werken met een specifiek life-cycle model. Hoewel de terminologie niet altijd gelijk is, beschouwen de meeste modellen 6 fasen in de life-cycle van een software produkt. Voor referentie zijn hier de engelstalige termen aangehouden.

User requirements: definitie van het project
Software requirements: analyse van de aanpak van het project
Architectural design: beschrijving van het ontwerp van het project (high-level)
Production: gedetailleerd ontwerp en implementatie
Transfer: in gebruikname van de software
Maintenance: after-sales service

In het begin van de zeventiger jaren was het 'Waterval-model' een van de eerste beschrijvingen van het ontwikkelproces dat algemeen werd gebruikt en vormde aldus de basis voor vele procedures die bij de aanschaf van software door de overheid en industrie gehanteerd worden. Het 'Waterval-model' beschrijft de documenten die opgezet moeten worden ter afsluiting van elk van de ontwikkelfasen, elk document vormt hierbij de basis voor de uitvoering van de opvolgende fase, waarbij het model ook voorziet in de terugkoppeling van de bevindingen en benodigde modificaties naar de voorgaande fase. Deze documenten maken uiteindelijk bij de aflevering deel uit van het totale produkt



Een voorbeeld van een Software Life-cycle model: het 'Waterval Model'

Vandaag de dag is men van mening dat dit model een aantal serieuze tekortkomingen heeft. Vooral de nadruk op de overdracht van de documenten (waarschijnlijk als gevolg van vraag hiernaar bij overheidsprojecten) wekten de suggestie van een filosofie waarbij het produkt 'over de muur gegooid' wordt naar een team van kwaliteitscontroleurs en er weinig terugkoppeling is naar voorgaande projectfasen. Hierdoor is het model ook minder geschikt voor projecten waarbij intensief gebruik gemaakt wordt van terugmeldingen door beta-testers. Het watervalmodel is dan ook aangepast en verbeterd in een evolutionair ontwikkelmodel (waarbij het ontwikkelproces wordt herhaald op basis van informatie uit de praktijk), het 'incremental-model' (waarbij de software in fasen wordt afgeleverd) en het transformatie-model (waarbij een proces voor de automatische regeneratie van de software wordt beschreven op basis van operationele ervaringen).

Het 'Spiral Life-cycle model' van Boehm is een voorbeeld van een moderne zienswijze t.o.v. de Software Life-cycle. Binnen dit model kunnen de eerder genoemde modellen als speciale verschijningsvormen worden opgenomen. [Ref 1].

Bij de keuze van de hulpmiddelen die zullen worden inzet voor Software Quality Assurance (S.Q.A.) is het belangrijk dat deze bijdragen aan een gecoördineerd en beheersbaar proces en inzetbaar zijn in zo veel mogelijk fasen van het life-cycle-model.

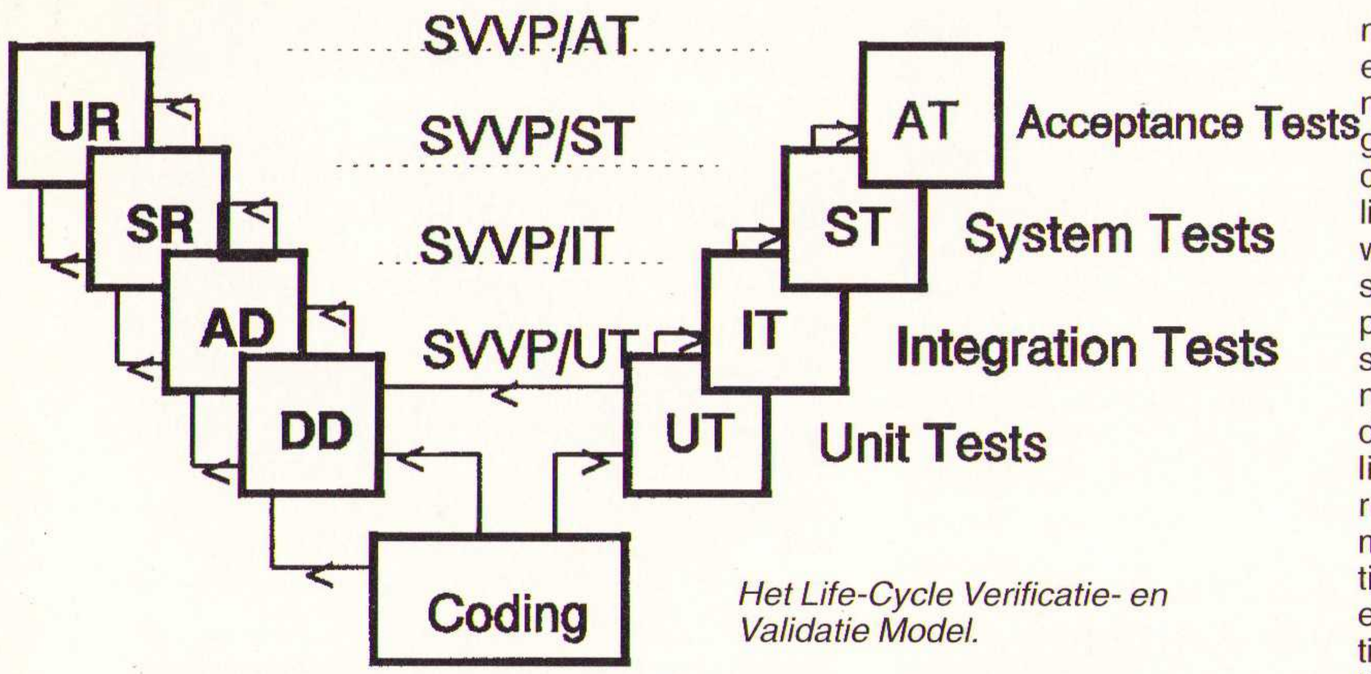
3 Test- en documentatie eisen in standaarden voor software.

De ISO9000 kwaliteitsstandaarden bieden een raamwerk voor het Kwaliteitsmanagement systeem van een bedrijf, een infrastructuur waarin de organisatie kan vastleggen hoe ze de diverse zaken uitvoert. Hierbij wordt tot op het benodigde gedetailleerde nivo beschreven welke procedures gevolgd worden. Op Europees nivo is gekozen voor de ISO9000/EN29000 standaarden, maar in dit artikel willen we ook kort ingaan op de standaarden die gehanteerd worden door de ESA en defensie.

3.1 ISO9000 - Part 3.

Gezien het tot nu toe relatief geringe gebruik van statistische kwaliteitscontrole bij de produktie van software (m.u.v. de overheid en bij militaire projecten) was er een apart document nodig om aan te geven hoe de ISO9001 standaard voor kwaliteitsmanagementsystemen voor ontwikkeling en produktie konden worden toegepast op software. Deel Drie van ISO9000, gepubliceerd in juni 1991 beschrijft deze "Guidelines for the application of ISO9001 to the development, supply and maintenance of software". [Ref. 2]

Bij het identificeren van de relevante aanbevelingen in deze belangrijke standaard moeten we waarschuwen tegen de selective interpretatie van welke standaard dan ook - alleen het



Het Life-Cycle Verificatie- en Validatie Model.

standaarddocument zelf is de geautoriseerde referentie voor de implementatie.

Echter, als het gaat om het uitvoeren van testen en performance-analyse is ISO9000-3 erg duidelijk in de omschrijving van de uitgangspunten. De leverancier draagt alle verantwoordelijkheid voor het vastleggen van de verificatie-eisen van de softwareprojecten en voor het beschikbaar maken van de benodigde voorwaarden om aan deze eisen te voldoen. Verificatie is in elke fase van het gekozen Life-cycle model nodig, vanaf het ontwerp tot en met de 'after-sales' ondersteuning. De testen moeten gebaseerd zijn op een nauwkeurig vastgelegd plan, de Test-Plan documenten. Dit Test-Plan moet alle fasen van de ontwikkelcyclus omvatten met een beschrijving van de test-cases en test-data, met een definitie van de verwachtte resultaten voor module-, integratie- en systeemtest bovendien een functionele- performancetest. Ook de test-omgeving, gereedschappen en testsoftware moeten in dit testplan zijn vastgelegd.

Zoals in de meeste standaarden voor software-engineering en S.Q.A. wordt er veel aandacht besteed aan configuratiemanagement, een methode voor het beheren en beheersen van de versies van elke softwaremodule. ISO9000-3 adviseert dat configuratiemanagement wordt toegepast, niet alleen voor de software specificaties en bestanden, maar ook voor die delen van het ontwikkelsysteem die invloed hebben op de functionele en technische specificaties.

(Een artikel dat verder op dit onderwerp ingaat vindt u onder [Ref 3].)

3.2 European Space Agency (ESA) software enigineering standaard.

Voor een meer gedetailleerde en veeleisende benadering van het management van software-ontwikkeling is het nuttig om de 'ESA Software Engineering Standards issue 2' te bekijken [Ref 4] zoals gepubliceerd in februari 1991. Hoewel de bron van deze standaard erg 'zwaar' klinkt geldt dit niet voor de inhoud van het document. Het is gebaseerd op gezond verstand en ervaring en is dan ook aan te raden voor een ieder die betrokken is bij softwarekwaliteit en een waardevolle referentie bij de opzet vanontwikkel- en S.Q.A.-procedures. Er zullen situaties zijn waarbij het volledige gebruik van deze standaard niet nodig is voor commerciële of industriële software, maar het gebruik van 'hoe-je-het-zou-kunnen-doen' en 'wat-je-moet-doen' elementen maakt het een erg bruikbaar document bij de voorbereiding op certificatie voor ISO9001 of DOD-STD-2167A.

Zoals bij de meeste standaarden, vereist het ESA document dat een project wordt gepland in een aantal afzonderlijke fasen. Hoewel er hierbij een Life-cycle model nodig is, wordt er niet noodzakelijk van een bepaald model uitgegaan. Het vereist dat de ontwikkeling en produktie wordt gebaseerd op drie principes uit de software engineering : Top-down ontwikkeling, gestructureerd programmeren en gelijktijdige produktie en documentatie. Ter verduidelijking van de taken binnen een project stelt het document dat het projectmanagement verantwoordelijk is voor de selectie van de gebruikte methoden en gereedschappen en het opleggen van het gebruik hiervan. Het is de verantwoordelijkheid van S.Q.A. om er op toe te zien dat de juiste gereedschappen, technieken en methoden worden geselecteerd en toegepast.

Net als in ISO9000-3 maakt de ESA specificatie duidelijk dat alle software onderdelen, inclusief de gebruikte ontwikkel- en testsystemen onder het configuratiemangement dienen te vallen. De testsoftware dient volgens de zelfde procedures en standaarden te worden ontwikkeld, gecontroleerd en gedocumenteerd. Dit is een basisvoorwaarde voor regressietesten.

Het Software Verificatie en Validatie plan (SVV) is het basistestplan voor het proces en vereist dat het testplan vastlegt dat de softwaremodule of het systeem voldoet aan de verwachte eisen en dat de verschillen tussen de verwachte en werkelijke resultaten worden vastgelegd.

3.3 Het Software Verificatie en Validatie plan (SVV)

Uitgaande van kwalitatief hoogstaande software is het testen van het software produkt t.o.v. de specificaties van fundamenteel belang. De ESA standaard onderkent dat de testactiviteiten vaak het meest tijdrovende en dure onderdeel van het project zijn. Testen zou zowel verificatie als validatie moeten omvatten. Volgens het IEEE comité gelden de volgende definities :

Verification : "the process of evaluating a system or component to determine whether the products of a given development phase satisfy the conditions imposed at the start of that phase"

Validation : "the process of evaluating a system or component during or at the end of the development process to determine whether it satisfies specified requirements".

Eenvoudiger gezegd, meet verificatie het softwareprodukt t.o.v. de vastgelegde specificaties, terwijl validatie bevestigt of de software alle gevraagde functionaliteit correct uitvoert.

In een effectief en efficiënt testproces moeten deze activiteiten plaats vinden in alle fasen van het ontwikkeltraject. Dit proces kan geïllustreerd worden met behulp van een diagram van het Software Verificatie en Validatie plan dat ook vaak het 'V'-diagram wordt genoemd.
Het Life-Cycle Verificatie- en Validatie Model.

Deze representatie geeft ons enkele belangrijke richtlijnen voor de opzet van een gedetailleerd testplan en voor de keuze van de gereedschappen en methoden voor de implementatie van dit plan:
Binnen elke fase van de Life-cycle moet het testplan de activiteiten die zijn uitgevoerd verifiëren met de eisen die in de voorafgaande fase gesteld zijn.

Nadat een fase van het testpan is geïmplementeerd (unit, module of systeem) moeten de resultaten geverifieerd worden met de overeenkomstige ontwerp-fase.

Bij de opzet van een efficiënt proces kan geen enkele fase in het testplan, noch een testmethode, gezien worden als belangrijker dan een ander. Elke fase draagt logischerwijze bij aan het verzekeren van de kwaliteit van het eindprodukt.

3.4 DOD standaard 2167A

Ter vergelijking hiermee willen we kort ingaan op de methoden en gereedschappen zoals die beschreven staan in een standaard voor militaire software die algemeen wordt toegepast, de US Department of Defense standard 2167A, voor "Defense System Software Development" [Ref. 5].

Hoewel strikt gebaseerd op een door documenten en contoles overheerste sequentiële aanpak, staat de standaard een iteratieve of recursieve uitvoering toe van de belangrijkste fasen, waarin deze niet verschilt van de reeds besproken standaarden. De DOD standaard vereist het gebruik van systematische en goed-gedefinieerde software-ontwikkelmethoden voor alle belangrijke activiteiten waaronder integratie en testen, waarbij deze methoden weer moeten voldoen aan de formele documentatie en controles die bij deze standaard van toepassing zijn.

De testelementen van het ontwikkelproces moeten beschreven zijn in het Software Test Plan, terwijl aan het eind van elke fase de met het eindprodukt meegeleverde documentatie een complete Software Test beschrijving of test-rapport moet bevatten van deze fase.

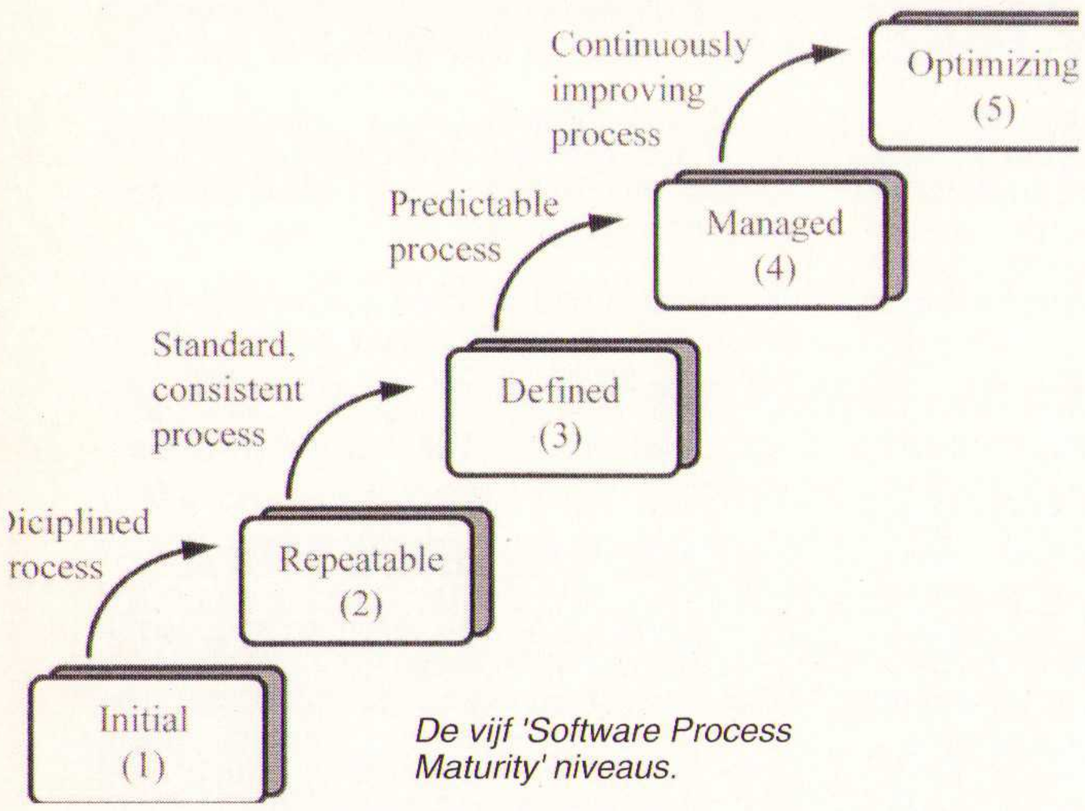
De 2167A standaard beschrijft de criteria die van bijzonder belang zijn bij de selectie en het-gebruik van test- en S.Q.A. gereedschappen. De leverancier is verplicht in het testplan de limieten van de geleverde software te testen en vast te leggen. Hierbij is de noodzaak van Performance Analyse vastgelegd, waarbij niet alleen de prestatie-limieten waarbij getest is van belang zijn, maar ook de reservecapaciteit of 'headroom' van het systeem.

4 De 'TickIT-Guide'

De "Guide to Software Quality Management construction and certification using EN29001", [Ref. 6] zoals gepubliceerd door de U.K. Department of Trade and Industry is uiteraard geen standaard, maar wel een waardevolle referentie voor wat betreft de status van kwaliteits-managementsystemen in de softwareïndustrie. De 'TickIT-Guide' is een bruikbaar 'recept' voor de opzet van een kwaliteitmanagementsysteem ter voorbereiding op ISO9000 certificatie.

4.1 Capability Maturity Model (CMM)

In November 1986, is het Software Engineering Institute (SEI), met assistentie van de Mitre Corporation, begonnen met de ontwikkeling van een raamwerk dat door organisaties gebruikt kon worden ter verbetering van het software ontwikkelproces. In de praktijk werd dit raamwerk echter vaak gebruikt als model ter verbetering van het proces, waarna men besloot dit raamwerk hiertoe daadwerkelijk uit te breiden. Het resultaat was het Capability Maturity Model (CMM) Versie 1.0 dat later uitgebreid is tot Versie 1.1 [Ref 11]. In dit document wordt het raamwerk beschreven, de structurele elementen er-



De vijf 'Software Process Maturity' niveaus.

van en de definitie van de vijf nivo's.
De volgende omschrijvingen karakteriseren deze vijf nivo's en de belangrijkste proceswijzigingen :

1) Initial: Het software proces kan omschreven worden als Ad-Hoc, soms zelf chaotisch. Slechts enkele processen zijn gedefinieerd en het succes is afhankelijk van individuele prestaties.
2) Repeatable: Door toepassing van procesmanagement worden kosten, tijdschema en functionaliteit bewaakt. Er is een procesdisipline bereikt waarin een vergelijkbaar project op dezelfde wijze uitgevoerd kan worden.
3) Defined: Het softwareontwikkelproces voor wat betreft de management en engineering activiteiten is gedocumenteerd, gestandaardiseerd en geïntegreerd in een standaard ontwikkelproces voor de organisatie. Alle projecten maken gebruik van een bewaakt proces voor de ontwikkeling en het onderhoud van de software.
4) Managed: Gedetailleerde informatie van het softwareproces en softwarekwaliteit worden actief gemeten. Zowel het softwareproces en de produkten worden kwantitatief begrepen en bewaakt.
5) Optimizing: Er is een continue programma ter verbetering van het softwareproces door terugkoppeling van ervaringen en de invoering van nieuwe technologiën.

Het CMM wordt tegenwoordig steeds meer toegepast door organisaties als referentie voor waar men is en waar men naar toe wil aangaande de kwaliteit van het software ontwikkelproces. Voor meer informatie en een vergelijking met ISO verwijzen we graag naar het Software Engineering Research Centre in Utrecht of het Software Engineering Institute in de U.S.A. Door het SERC is een aantal voorlichtingsbijeenkomsten georganiseerd, die naast het CMM ook ingaan op ESPITI (European Software Process Improvement Training Initiative) en SPICE (Software Process Improvement and Capability dEtermination).

4.3 Het belang van Software Quality Assurance voor embedded software.

Er zijn in de loop der jaren steeds meer gereedschappen ontwikkeld die behulpzaam zijn bij Software Quality Assurance. Sommige hiervan bestaan al jaren, andere zijn net in de markt of nog in ontwikkeling. Uiteraard kunnen we binnen het ontwikkelproces voor embedded software direct gebruik maken van een groot aantal gereedschappen die ook voor de ontwikkeling van 'host-based' software ingezet worden. Denk hierbij aan CASE (Computer Aided Software Engineering) omgevingen, waarbij al dan niet gebruik gemaakt wordt van een formele taal (bijvoorbeeld SDL) en de versie-controle systemen. Hoewel door de aard van embedded software (of firmware) S.Q.A. hier minstens zo belangrijk is, zo niet belangrijker, valt er op dit gebied nog veel te doen. Door het ontbreken van specifieke oplossingen worden er vaak nog 'klassieke' gereedschappen als Logic analyzers en In-Circuit Emulatoren ingezet voor taken waarvoor ze eigenlijk niet geschikt zijn.

In tegenstelling tot host-software spelen er bij embedded software een groot aantal harde eisen ten aanzien van de responsetijd en het geheugengebruik. Het is bij een administratieve toepassing geen enkel probleem als de software iets trager reageert dan volgens de specificaties gewenst is. Dit is hinderlijk maar zeker niet desastreus, de gevolgen hiervan kunnen eenvoudig opgevangen worden. Bij een embedded toepassing, bijvoorbeeld een elektronische ontsteking, mag echter een meting NOOIT gemist worden of we lopen de kans dat het systeem faalt in het functioneren. Zo is er een groot aantal voorbeelden op te noemen waarbij het systeem volledig deterministisch moet reageren op bepaalde stimuli. Voor het geheugengebruik geldt iets soortgelijks.

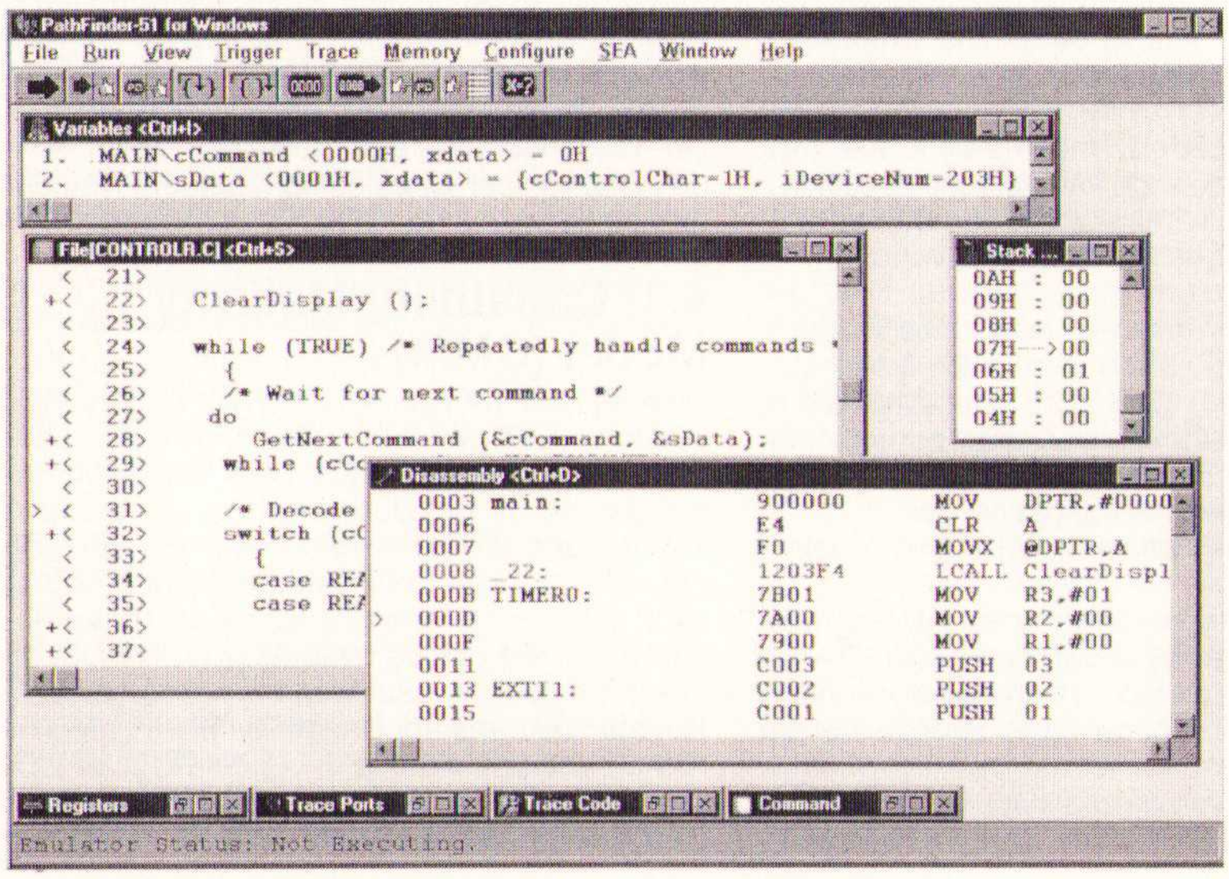
Bij een host-systeem is er vaak reserve-geheugen aanwezig in de vorm van 'swap-space' op een harddisk o.i.d. Bij een embedded Real-Time toepassing is het vaak om economische redenen al belangrijk het beschikbare werkgeheugen zoveel mogelijk te reduceren, het 'overlopen' naar eventueel beschikbaar back-up geheugen is niet mogelijk of in elk geval te traag ten aanzien van de gestelde real-time specificaties. Het gevolg zal zijn dat het systeem niet aan de specificaties voldoet of, waarschijnlijker nog, onderuit gaat. Het falen van een embedded systeem heeft in de regel ernstiger gevolgen dan bij een host-systeem. Het is ronduit slecht voor het imago (denk aan een telefooncentrale) of heeft zelfs levensbedreigende (medische, militaire, luchtvaart-toepassingen) gevolgen. Deze twee gevolgen hebben gemeen dat ze erg veel kosten met zich mee brengen.

Een andere, nog ondergewaardeerde, reden voor S.Q.A. bij embedded systemen is het feit dat het vaak heel moeilijk is om bij de firmware (embedded software) een nieuwe versie te installeren. Stel dat men als fabrikant 100.000 televisietoestellen of GSM-telefoons moet 'upgraden'. Ervaringen in de autoindustrie hebben geleerd dat hiermee vele miljoenen gemoeid kunnen zijn, nog afgezien van de schade aan het imago en de juridische aspecten.

Een symbolische testroutine voor de Ashling in-circuit emulator met de Test-Macrotaal.

```
; ************************************************
; *  File : Test Macrolanguage File for PACKET.C  *
; *  Ver  : 1.3.7                                 *
; *  11:35  December 14 1991                      *
; ************************************************
declare                    ; Declare the following local variables
  LOCAL  int    %no_of_steps
enddeclare                 ; All variables declared
echo on
/autokey:'\ALT-F\lpacket.cso\E\' ; Load the program-under-test
/autokey:'\ALT-D\H'
res d                      ; Reset the processor

wait on
g t main                   ; Execute from RESET to main
  mo  retry_count          ; Monitor key variables in program
  mo  packet_number
  mo  index
  mo  line_error_count[index]
wait off

%no_of_steps=0             ; Initialise the loop variable.
command off

while %no_of_steps < 10    ; Step 10 times.
   step h
   %no_of_steps = %no_of_steps + 1   ; Increment the
                                     ;loop variable
wend

command on
win off
/autokey:'\ALT-A\I'        ; Activate a trigger
g 0 t write_p1_signature    ; Run the program
/autokey:'\ALT-Q\Y'        ; Quit
```

Pathfinder Source-level debugger voor de CTS emulator van Ashling Microsystems.



Bij het opzetten van testprocedures voor embedded software gelden de volgende bijzondere eisen :

a) De software moet volledig in real-time lopen, zonder dat de test hierop ingrijpt.
b) Veiligheids- en performancekritische toepassingen vereisen meting t.o.v. exacte vooraf vastgestelde specificaties. Statistische metingen of 'blind-spots' zijn hierbij niet gewenst.
c) De testen moeten worden uitgevoerd op de uiteindelijke produktieversie van de software.
d) Validatie vereist het testen van alle systeemfuncties, in het bijzonder ook met asynchrone stimuli.
e) Een zeer hoog nivo van softwarekwaliteit is al vereist bij de initiële produktrelease.

Het zal uit het bovenstaande duidelijk zijn dat gereedschappen die alleen in de hostomgeving werken beperkt zijn in hun toepassing bij embedded software.

5 Een goede basis... is het halve werk

Juist bij de ontwikkeling van embedded software is het belangrijk te werken op basis van een kwalitatief hoogwaardig ontwikkelproces en ontwikkelgereedschappen. Steeds meer bedrijven overwegen bijvoorbeeld de toepassing van een CASE omgeving en Top-Down ontwikkeling. In de telecommunicatie waar veel complexe projecten plaatsvinden waaraan niet alleen hoge kwaliteitseisen worden gesteld, maar die ook onder hoge tijdsdruk staan wordt veel gebruik gemaakt van de formele specificatietaal SDL (Specification and Design Language) [Ref 12] waarbij uitgaande van een formele beschrijvingstaal het systeem geverifieerd en gevalideerd kan worden waarna automatisch de C-code gegenereerd kan worden. In combinatie met message-based Real-Time Operating Systemen (R.T.O.S.) als OSE van Enea Data kan zo vrijwel het gehele systeem automatisch gegenereerd worden. Juist bij het R.T.O.S. is ook veel winst te behalen. Zo is er tegenwoordig een gedistribueerd R.T.O.S. beschikbaar dat gecertificeerd is voor veiligheidskritische toepassingen (TUV, IEC1508) zoals in de petrochemische, medische en nucleaire industrie. Het spreekt voor zich dat een dergelijk systeem ook grote voordelen biedt bij iets minder veeleisende toepassingen.

Zo ver hoeven we het echter niet per se te zoeken. Ook het consequent gebruik van een goed versiecontrole systeem en gecertificeerde C-compilers of andere codegeneratoren vormen al een hele belangrijke stap in de goede richting.

3.6 Klassieke S.Q.A. tools.

Zoals gezegd zijn er nog maar weinig gereedschappen voor S.Q.A. in embedded systemen beschikbaar. Dit geldt in het bijzonder voor de testfase. Voor de debugfase vormt de In-Circuit Emulator een gunstige uitzondering. Dit instrument is al vele jaren beschikbaar voor de populaire microprocessors en biedt een vrijwel ongeëvenaarde combinatie van controle over de executie van software en visualisering van wat er in het target gebeurd is. Indien een In-Circuit Emulator gebruikt wordt met een moderne Source- en Assembly level debugger beschikt men over een voldoende krachtige oplossing voor de debugfase.

Voor de testfase heeft een In-Circuit-Emulator (ICE) echter maar een beperkte toepassing. Hij kan uitstekend ingezet worden voor de uitvoering van de testcase door middel van de beschikbare macrotaal. Maar net als de logic-analyzer heeft hij een groot aantal beperkingen als het gaat om metingen in de testfase.

3.6.1 De Test-macrotaal

Voor automatische executie van regressietesten is het noodzakelijk dat op een eenvoudige wijze steeds dezelfde test kan worden uitgevoerd, waarbij een groot aantal teststimuli gegenereerd moet kunnen worden. Door de grote mogelijkheden die een ICE biedt is dit instrument hiervoor uitermate geschikt, indien deze is voorzien van een krachtige Test-macrotaal. Hierbij moet het instrument in staat zijn met de symbolische functie- en procedurenamen te werken, zodat ook nadat een nieuwe softwareversie is gegenereerd nog steeds dezelfde test kan worden gebruikt, doordat de symbolische waarden automatisch verwijzen naar de nieuwe absolute adressen. Ook moet het mogelijk zijn lokale variabelen in de testroutine te gebruiken en resultaten weg te schrijven naar een file voor latere analyse.

6.2 Beperkingen van de klassieke gereedschappen

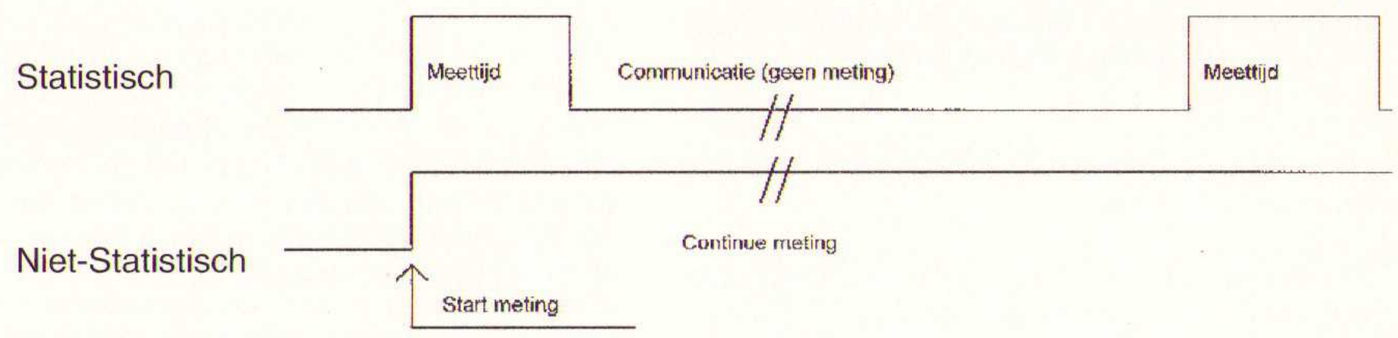
Waarin ligt de beperking van de klassieke tools voor S.Q.A. ? Het probleem is dat deze tools ontwikkeld zijn voor het vinden van problemen (debugging) en niet voor continue meting aan software. Zowel een logic analyzer als ICE worden hiervoor wel ingezet, maar de beperking ligt vooral in het feit dat deze instrumenten alleen statistische metingen kunnen uitvoeren. Beiden programmeren bijvoorbeeld voor performance metingen het trace-systeem (state-analyzer) om aan een aantal functies of modulen van de software te meten. Helaas is de grootte van dit tracegeheugen beperkt, meestal tot maximaal 10.000 'frames'. Als dit geheugen vol is moet de informatie naar een hostcomputer worden verzonden via een communicatiekanaal. Tijdens deze communicatie ligt de meting stil. Hierbij is in de praktijk de 'dode' tijd (zeer) lang ten opzichte van de meettijd.

Het zal duidelijk zijn dat voor een betrouwbare performance-analyse een niet-statistische of continue meting noodzakelijk is. Een ander probleem is dat de logic-analyzer en de ICE maar een beperkt aantal functies in de gaten kan houden, meestal ca. 10 . Dit betekent dat er van te voren bepaald moet worden aan welke kritische functies we willen meten of dat er zeer vele verschillende metingen nodig zijn. Dit laatste maakt niet alleen het statistische element van de meting nog veel groter, maar vergt vooral veel tijd en moeite bij het opzetten van deze verschillende metingen. Nog groter wordt het probleem bij de huidige generatie 16/32 bit CPU's, waarbij door het gebruik van intelligente 'pipelines' en z.g.n. 'caches' de informatie op de bus van de processor niet meer volledig is. Soms worden instructies vanuit de cache uitgevoerd zonder dat dit op de bus te zien is, of worden juist instructies die wel ingelezen zijn vanuit het geheugen niet uitgevoerd. Bij instrumenten die geheel van de externe bus van de processor afhankelijk zijn ontbreekt deze informatie.
Van een 100% betrouwbare meting is in alle bovengenoemde gevallen geen sprake meer !

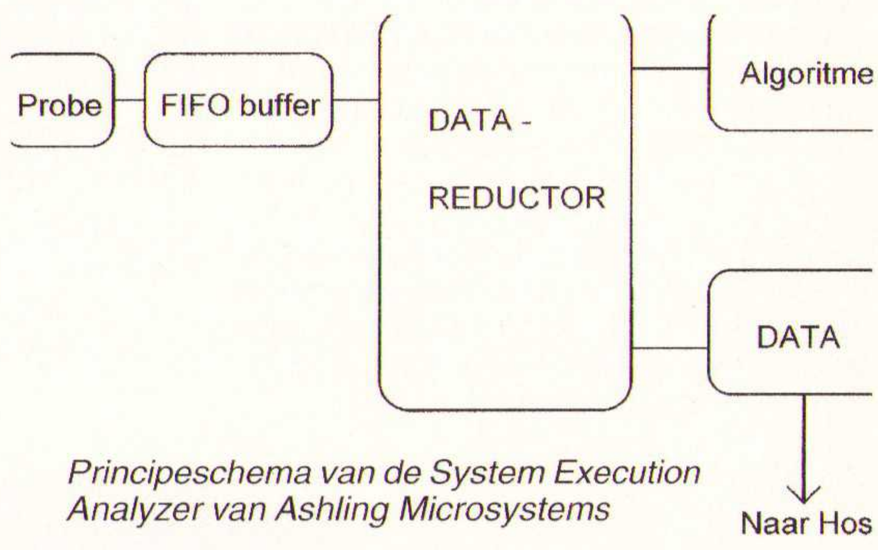
3.7 Een nieuwe generatie S.Q.A. gereedschappen.

Sinds enkele jaren zijn er gereedschappen van een nieuwe generatie beschikbaar gekomen die de eerder genoemde problemen niet hebben. Het geheim hierbij is dat door toepassing van een lokale datareductor in een meetprobe de meetinformatie continue verwerkt wordt en alleen de uiteindelijke meetresultaten voor presentatie en verdere verwerking naar een hostsysteem doorgegeven worden. Dit systeem is vroeger voor het eerst door de firma North-West Instrument Systems (NWIS) toegepast, en wordt nu gebruikt door Ashling Microsystems in haar In-Circuit Emulatoren voor 8/16-bit microcontrollers (System Execution Analyzer optie) en door Applied Microsystems Corp. (CodeTEST) voor de high-end 16/32 bit microprocessors.

Statistische versus NIET-Statistische meting

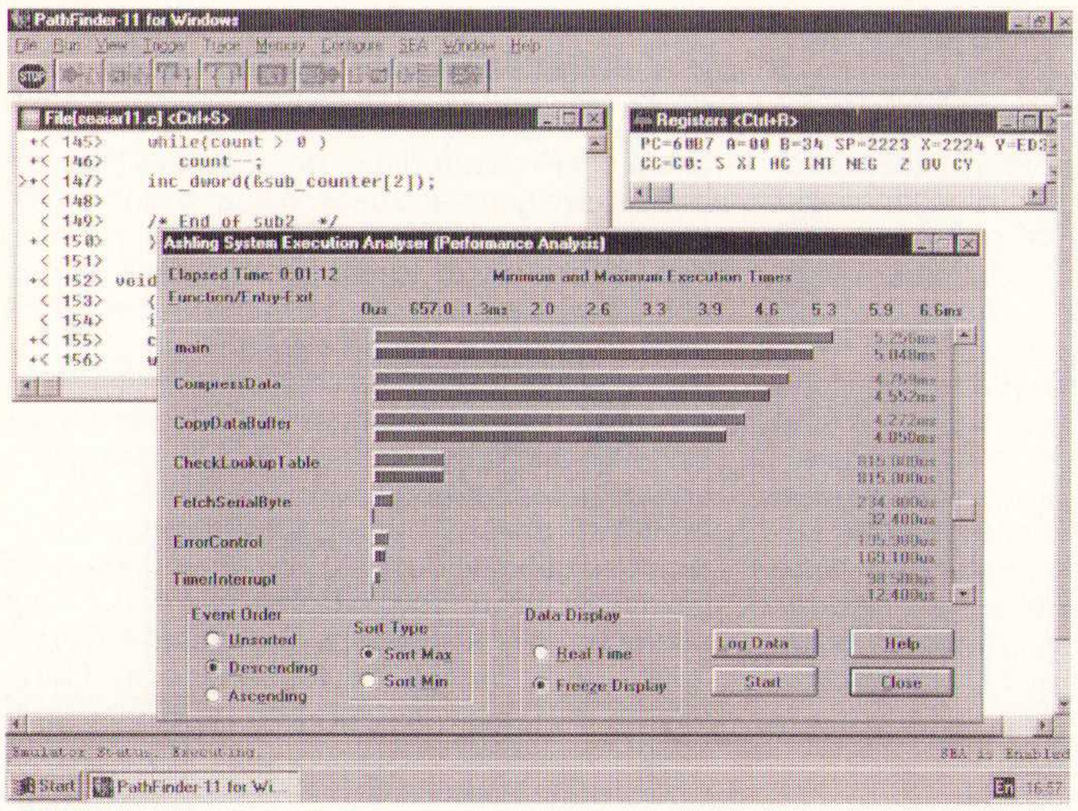


Het bovenstaande principeschema verduidelijkt het een en ander. Via een probe en een FIFO-register komt de informatie van de adres- en databus van de microprocessor bij een datareductor terecht. Deze datareductor is geprogrammeerd om bijvoorbeeld de 'entry-' en 'exit-'adressen van bepaalde functies, modulen of RTOS taken te volgen. Afhankelijk van het algoritme waarvoor de datareductor is geprogrammeerd zal de datareductor de informatie in real-time verwerken. Voor performance-analyse met Min/Max executietijden gebeurt bijvoorbeeld het volgende.



Principeschema van de System Execution Analyzer van Ashling Microsystems

Als het 'entry-adress' van een functie 'testmeting' binnenkomt zal de datareductor het exacte tijdstip waarop dit gebeurt vastleggen in het datageheugen. Als nu enige tijd later het bij deze functie behorende 'exit-adress' verschijnt zal de datareductor het tijdsverschil (is de executietijd van deze functie) vergelijken met de voorgaande minimum- en maximum-executietijd. Als de nieuwe tijd korter is dan het voorgaande minimum, of langer is dan het voorgaande maximum zal de tabel overeenkomstig bijgewerkt worden. De inhoud van deze tabel met absolute min/max executietijden wordt weer doorgegeven aan het host-systeem. In de praktijk blijkt dat we zelfs voor de huidige zeer snelle 32-bit RISC processors op deze manier een 100 % continue meting kunnen halen, waarbij de FIFO dient om een eventuele snelle opeenvolging (burst) van



NIET-Statistische Min/Max executietijd meting; CTS-emulator van Ashling Microsystems.

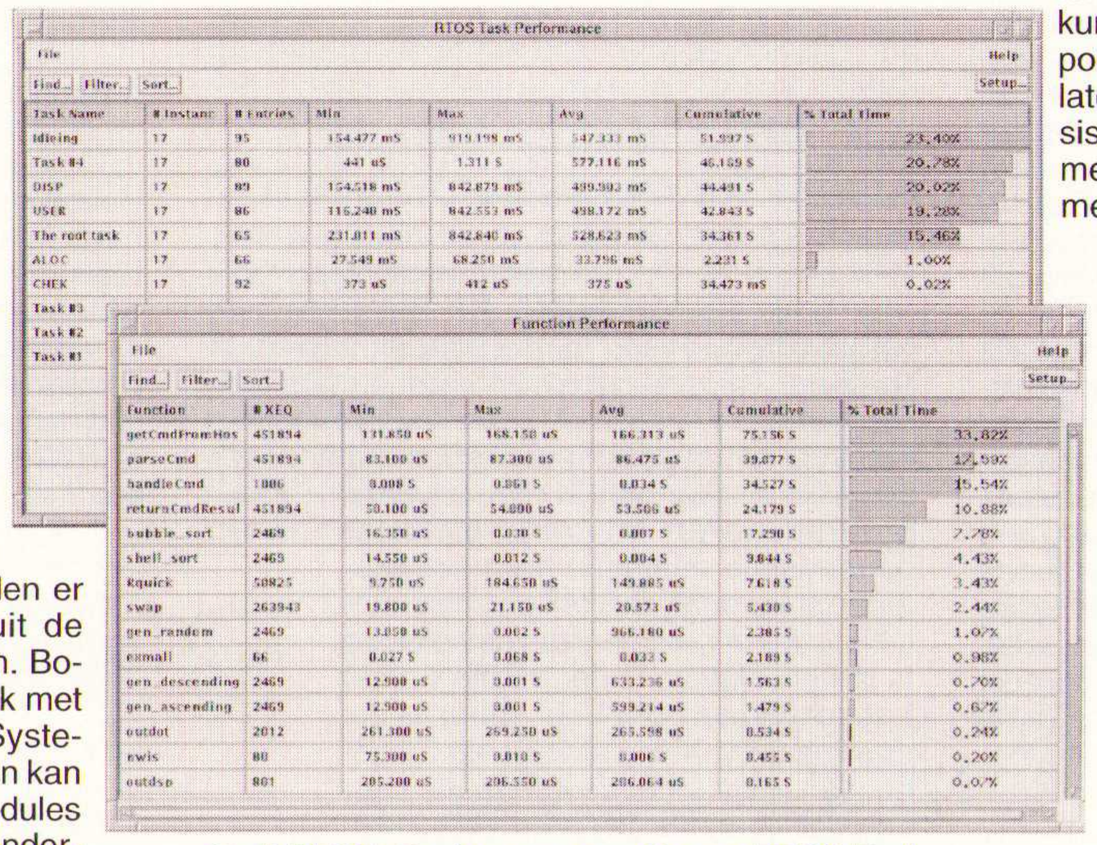
entry's of exit's tijdelijk op te vangen (denk aan recursieve routines)

Door nu het algoritme dat gebruikt wordt door de datareductor aan te passen kunnen verschillende metingen uitgevoerd worden.

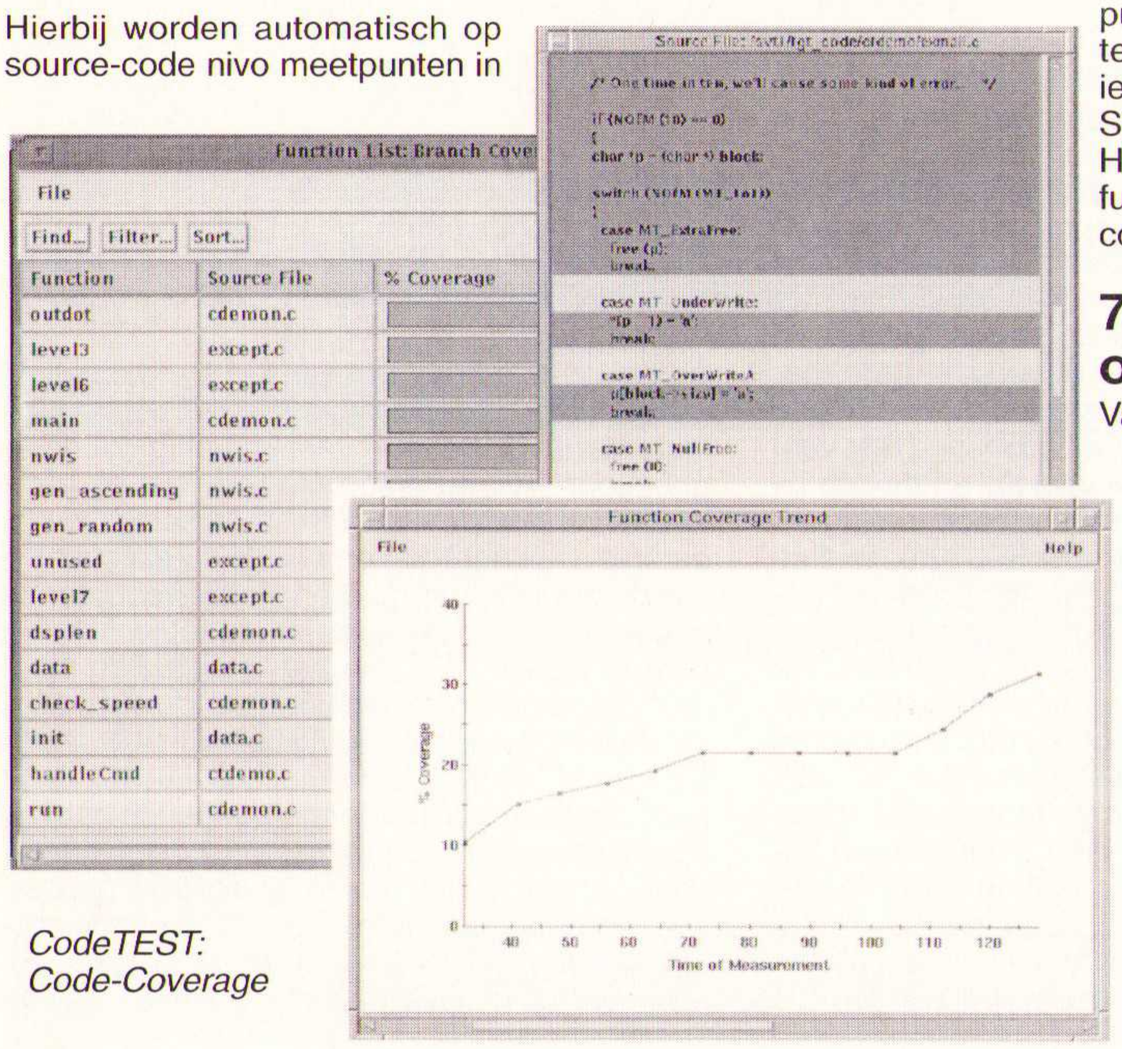
7.1 Oplossingen voor high-end 16/32-bit processors.

Voor de kleinere 8/16-bit processors zijn hiermee alle problemen opgelost, maar voor de krachtige 16/32-bit CPU's is dit maar een deel van de oplossing. Ten eerste maken deze CPU's intensief gebruik van een interne pipeline en/of cache, waardoor er adressen op de bus verschijnen die nooit geëxecuteerd worden. Of worden er juist adressen geëxecuteerd vanuit de cache, die niet op de bus verschijnen. Bovendien werken deze applicaties vaak met dynamische Real-Time Operating Systemen, waarbij een zelfde adres overeen kan komen met verschillende functies, modules of RTOS-taken. Om dit probleem te ondervangen heeft Applied Microsystems gekozen voor een proces van automatische instrumentatie van de source-code.

Hierbij worden automatisch op source-code nivo meetpunten in de software aangebracht die twee vaste adressen gebruiken en zo de datareductor een handje helpen. De datareductor zal nu pas een entry of exit van een functie detecteren als de CPU het meetpunt bereikt, waarna deze informatie op de bus komt. Dit lijkt een drastische aanpak, maar valt in de praktijk erg mee. Bij hostsoftware wordt al heel lang gebruik gemaakt van instrumentatie, en ook een aantal RTOS maken gebruik van debug-codes. Het bijzondere is dat Applied Microsystems er in geslaagd is een robuuste automatische instrumentatie uit te voeren met zeer weinig 'overhead'. Daarbij is deze standaard direct opgepakt door alle belangrijke fabrikanten van R.T.O.S. die de 'meetpunten' al hebben opgenomen in hun produkt. Het is verder nog belangrijk op te merken dat het voor de softwarefabrikant toegestaan is om de 'meetpunten' in de code te laten zitten bij aflevering, iets wat vooral bij DOD-STD-2167A belangrijk is. Het systeem kan 32.000 functies in een meting en continue volgen !



CodeTEST : Performance meting op RTOS-Task en functie nivo.



CodeTEST: Code-Coverage

7.2 Processor ondersteuning

Van de eerder genoemde fabrikanten zijn gereedschappen leverbaar die op een NIET-Statistische basis metingen kunnen verrichten voor vrijwel alle populaire microprocessors en controllers. Ashling ondersteunt o.a. de 8051, 8051-XA, 68HC11, en de 80196 families en Applied Microsystems de Motorola en Intel processors uit de 68xxx, PowerPC, 80x86 en I960 families. Al deze processors worden tot de maximale snelheden ondersteund ! Bovendien is er een universele probe beschikbaar waarmee alle overige processors tot 66 Mhz in principe ondersteund worden.Hiermee ondersteunt men bijvoorbeeld zelfs de PowerQUICC CPU. Om een verbinding met de CPU tot stand te brengen is het ook mogelijk een logic-analyzer adapter te gebruiken. In principe is een verbinding met de bus voldoende.

8 Metingen voor S.Q.A. in een embedded omgeving.

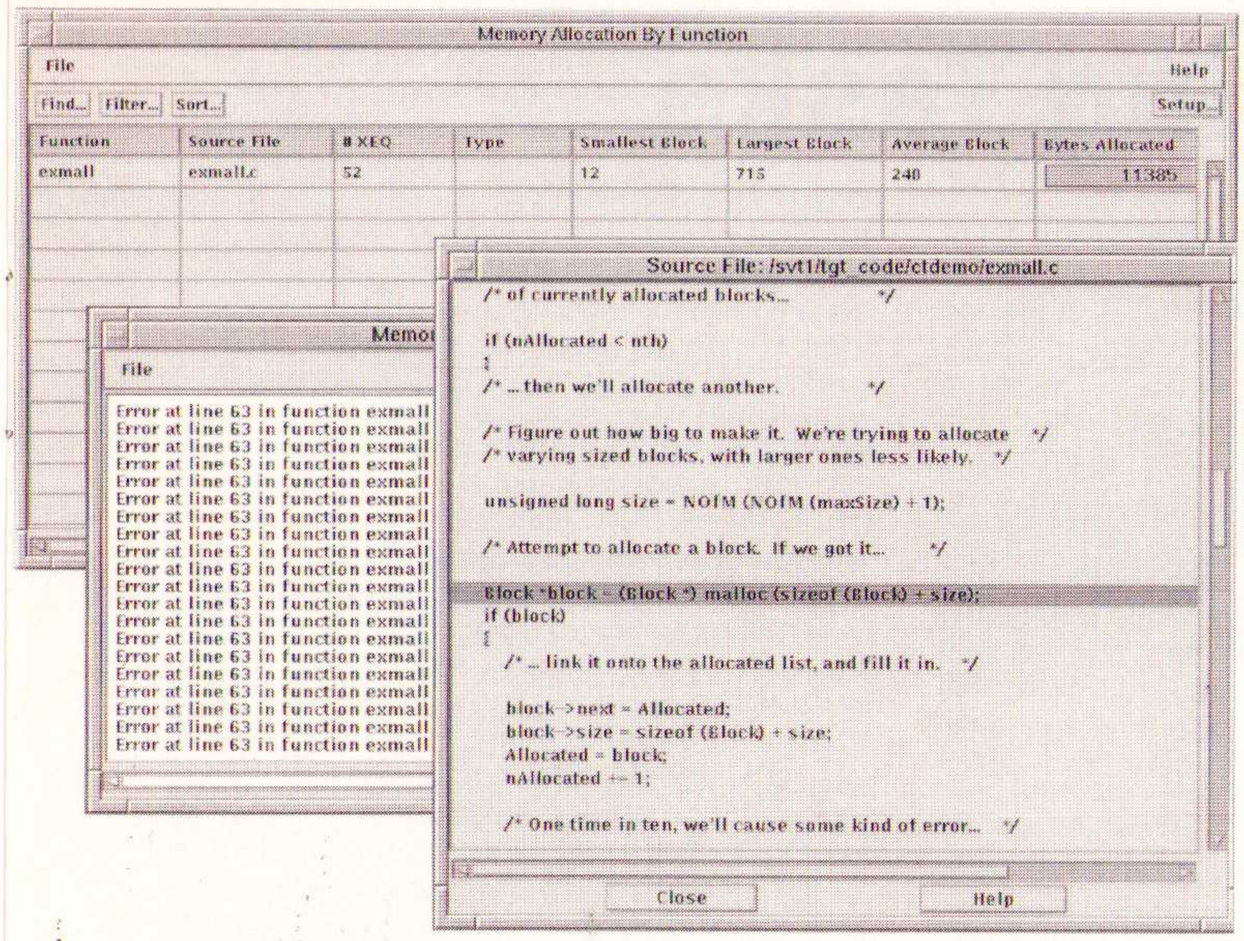
Op dit moment zijn er al diverse metingen mogelijk met de genoemde instrumenten. Er wordt nog gewerkt aan uitbreiding hiervan. Bij al deze metingen wordt, m.u.v. de control-flow trace, gebruik gemaakt van de lokale datareductor. Het opzetten van de meting is hierbij zeer eenvoudig. De gebruiker kiest aan welke RTOS-taken, modulen en functies de meting verricht moet worden of selecteert ze allemaal (tot een maximum van 32.000). Hierna wordt de meting gekozen en enkele seconden later worden de eerste resultaten zichtbaar. Hierbij kan de gebruiker direct zien wat de invloed is van een nieuwe test of bijvoorbeeld een interrupt. Het systeem meet continue zonder een executiecyclus te missen en geeft elke paar seconden de resultaten weer op het hostsysteem. Uiteraard kunnen de resultaten in de vorm van rapporten uitgeprint of bewaard worden voor latere analyse. Het is ook mogelijk op basis van een bestaande meting verder te meten of de resultaten van verschillende metingen te combineren.

8.1 Performance analyse

Met de performance analyzer is het mogelijk executietijden en intervallen te meten van zowel functies als RTOS-taken. De gebruiker kan hierbij continue zien welke taken en functies executeren, hoe vaak ze zijn geëxecuteerd, wat de absolute minimale- en maximale executietijden hierbij geweest zijn, de cumulatieve executietijd gedurende de meting en de executietijd als percentage van de totale executietijd.

Ook is het mogelijk te zien welke RTOS taken de CPU-tijd gebruiken en hoeveel kopieën van dezelfde taak er actief zijn. Door gebruik te maken van krachtige zoek- en sorteer routines kan snel een goed overzicht van de prestaties van zowel het systeem als geheel als op het meer gedetailleerde functie-nivo verkregen worden.

Met deze niet-statistische performance analyse kan zoals eerder opgemerkt in een meting een worst-case executietijd rapport gegenereerd worden voor de gehele systeemsoftware. Hierbij is de meting 100% betrouwbaar, met andere woorden, ook als een keer in de gehele testduur de executie van een functie of module door een onverwachte combinatie van interrupts iets langer duurt dan zien we dit terug in de rapporten. De meettijd is hierbij in principe onbeperkt. Als alternatieve meting biedt de CodeTEST nog het z.g.n. 'Call-Pair Linking'. Hierbij is te zien is hoe vaak bepaalde subroutines worden aangeroepen en door wie... Wanneer bijvoorbeeld blijkt dat een bepaalde korte routine vaak wordt aangeroepen door dezelfde 'caller' kan het efficiënt zijn deze subroutine in de 'calling'-routine op de nemen als in-line code. Dit bespaart ten koste van een klein beetje programmageheugen veel executietijd en stack-ruimte, die eerst gebruikt werd voor de aanroep van de subroutine.



CodeTEST/Memory : Memory allocation error detection

Een performance analyzer wordt ook wel profiler genoemd, omdat er een profiel ontstaat dat aangeeft waar de processor zijn tijd aan besteed. Dit dekt echter niet de gehele lading. Het is bijvoorbeeld ook mogelijk om in te schatten of er nog executietijd 'over' is voor het toevoegen van extra functionaliteit en voor het bepalen van 'worst-case' executie- en responsetijden.

8.2 Code Coverage

Bij Code Coverage wordt bijgehouden welke code wel en welke code nog niet geëxecuteerd is. Deze meting is vereist bij projecten volgens DOD-STD-2167A. Het bewijst namelijk dat de code op zijn minst eenmaal uitgevoerd is. Dit zegt op zich natuurlijk niets over de kwaliteit of de functionaliteit van de code, maar des te meer over de volledigheid van het testplan. Vooral bij de opbouw van dit testplan kan de Code Coverage meting ook goede diensten bewijzen omdat er direct te zien is op Task, module of op Source-code nivo welke code nog niet getest is, zodat het testplan overeenkomstig aangepast kan worden. Bovendien is het mogelijk rapporten te genereren die een overzicht geven van de progressie in de testcoverage die in de verschillende testen wordt bereikt, zodat snel inefficiënte testen geëlimineerd kunnen worden.

8.3 Memory Allocation

In embedded systemen vormen de memory-allocatie problemen misschien wel de meeste gevaarlijke en lastigst op te sporen problemen. Vooral bij het gebruik van een R.T.O.S. is het moeilijk een goede indruk te krijgen van het dynamische geheugen gebruik, terwijl er ook snel een 'geheugen-lek' kan ontstaan, een situatie waarbij door een proces 'vergeten' wordt het gealloceerde geheugen weer vrij te maken. Het CodeTEST systeem is in staat te rapporteren hoeveel geheugen er door welke functies via welke allocatieroutines ge-alloceerd wordt, hoe vaak dit gebeurt en in welke blokgroottes. Bovendien is het totale aantal ge-alloceerde bytes dynamisch te volgen. Als extra is het systeem in staat foutmeldingen te geven voor een aantal veel voorkomende allocatie problemen zoals 'out-of-heap', 'null-pointer', 'invalid-block-size' etc, etc.....
Met deze meetoptie is het mogelijk geheugenlekken al in een vroeg stadium te ontdekken en in te schatten of er nog geheugen 'over' is voor toekomstige opties. Ook wordt hiermee een analyse mogelijk van de software performance in verhouding tot het beschikbare geheugen !

8.4 Control-Flow Trace

De Control-Flow Trace is een uitbreiding op de bekende Assembly- en Sourcelevel trace. Door toepassing van een groot geheugen en alleen de registratie van de beslissingspunten (decision-points) in een trace-buffer kan het CodeTEST systeem een executie-history opslaan van het equivalent van ca. 100.000 source regels. Hierbij is een nauwkeurige time-stamp aanwezig. Zo wordt het mogelijk op het nivo van RTOS-taken en functies de 'Flow' van de software te volgen en te analyseren. De trace kan op Task, Control-Flow en Source-level bekeken worden. Dit vergemakkelijkt het meten van de 'echte' responsetijd van het systeem en verificatie van alle beslissingpunten (decision-points) in de software.

9 Conclusie.

Hoewel Software Quality Assurance voor embedded software geen eenvoudige zaak is, zijn er methoden en gereedschappen beschikbaar waarmee het in toenemende mate mogelijk is het software ontwikkelproces te verbeteren en metingen aan de kwaliteit van de software te verrichten en de resultaten vast te leggen in rapporten. Hiermee wordt het mogelijk bij een eindprodukt een volledige set rapporten af te leveren waarin aantoonbaar is wat de prestaties van het eindprodukt zijn t.o.v. de specificaties. Als gevolg hiervan is het mogelijk het software ontwikkelproces beter te beheersen en de risico's te verkleinen.

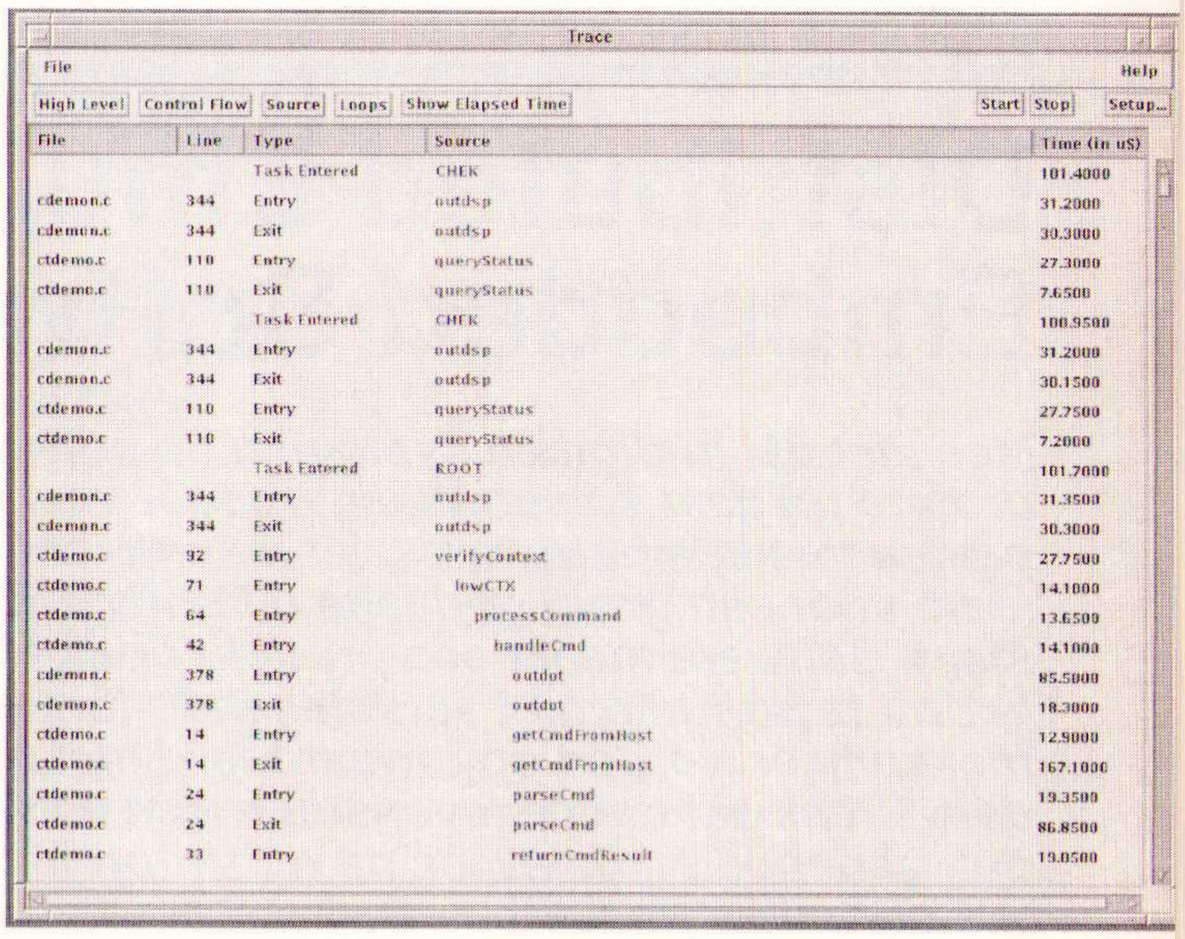
Referenties:

1. "A spiral model of software development", Barry W. Boehm, p.61, IEEE Computer, May 1988.
2. "Guidelines for the application of ISO9001 to the development, supply and maintenance of software", ISO9000-Part 3, June 1991, International Standards Organisation, Case Postale 56, CH-1211 Geneve 20, Switzerland.
3. "Framework for Success: A guide to Quality in Software Development and Support", January 1992, National Centre for Software Engineering, Dublin City University, Dublin 9, Ireland.
4. "ESA Software Engineering Standards, Issue 2", PSS-05-0 Issue 2, European Space Agency, 8-10 rue Mario-Nikis, 75738 Paris Cedex, France.
5. "Military Standard for Defense System Software Development", DOD-STD-2167A, February 1988.
6. "Guide to Software Quality Management System Construction and Certification using EN 29001", September 1990, Department of Trade and Industry, ITD7, Room 840, 66-74 Victoria Street, London SW1E 6SW, U.K.
7. "Engineering Software under Statistical Quality control"; Cobb and Mills, p.44, IEEE Software, November 1990.
8. "Portable Common Tool Environment (PCTE) Abstract Specification", Standard EMCH-149, European Computer Manufacturers Association, 114 Rue du Rhne, CH 2104 Geneva, Switzerland.
9. "IEEE Std. 610.12-1990, Standard Glossary of Software Engineering Terminology", Institute of Electrical and Electronics Engineers, Inc, 345 East 47th St., New York, NY 10017-2394, USA.
10. "ANSI/IEEE Std. 1008-1987, Standard for Software Unit Testing", Institute of Electrical and Electronics Engineers, Inc, 345 East 47th St., New York, NY 10017-2394, USA.
11. Capability Maturity Model for Software, Version 1.1 door Mark C. Paulk, Bill Curtis, Mary Beth, Chrissis Charles V. Weber. FTP : ftp.sei.cmu.edu , februari 1993.
12. SDL, standaard van de ITU-T (CCITT), SDT : Telelogic AB, Zweden
13 Documentatie Ashling microsystems CTS-In-Circuit Emulator
14. Documentatie Applied Microsystems, CodeICE emulatoren
15. Documentatie Applied Microsystems, CodeTEST.

Note : Een belangrijk deel van dit artikel is gebaseerd op een vertaling van de 'YES-guide' van Mr. Michael Healy, managing director van Ashling Microsystems. Het originele document kunt u opvragen bij Air-Parts B.V. Voor fouten of onvolledigheden in de vertaling kan Mr. Healy uiteraard niet instaan.

Nederland :
Air-Parts B.V
Gerard Fianen en Christophe Dumont
Postbus 255
2400 AG Alphen a/d Rijn
Tel : 0172 - 422.455 Fax : 0172 - 421022

Control-Flow Trace



Distributed Debugging Tasks

Effectief ontwikkelen van embedded applicaties wordt steeds belangrijker, en de verwachting is dat het in de nabije toekomst alleen nog maar belangrijker wordt.

Er worden steeds nieuwe manieren ontdekt en ontwikkeld om de verschillende problemen op te lossen. Vaak zien we dat de elektronica ontwikkelgereedschappen uit verschillende onderdelen bestaan zoals PC-debuggers, Target Monitoren, In Circuit Emulators (ICE) en Logic Analysers. Elk instrument wordt dan voor een specifieke taak ingezet. Door hogere snelheden, grotere complexiteit en de snelle opvolging van processor-generaties ontstaan er problemen met In-Circuit Emulators. Leverproblemen, het niet beschikbaar komen van een emulator voor een bepaalde processor omdat de markt te klein wordt geacht, of een bepaalde afgeleide processor wordt niet ondersteund, waardoor de vertrouwde werkwijze wegvalt. Een mogelijke oplossing is het gebruik van een Logic Analyser. Maar de bekende stand-alone analyser biedt geen optimale oplossing. Hij moet voorzien zijn van een voor ontwerpers bekende gebruikers interface en operating systeem. Dit is nodig om eenvoudig met andere gereedschappen te kunnen werken, en zorg te dragen voor een simpele communicatie. Ook moeten de functies van de analyser te besturen zijn. Met de steeds kleiner wordende behuizingen van de processoren en de steeds grotere hoeveelheid pinnen kan de ICE wel beschikbaar zijn, maar fysisch niet aangesloten worden (CQFP, BGA). De beste manier om de Logic Analyser aan te sluiten is met een 'debug pin array'. Die wordt bij het ontwerpen van het target op een goed bereikbare plaats neergezet en voorkomt zo dure en speciale adapters. Als het aansluiten elektrisch of mechanisch niet mogelijk is op de processor dan kan er bij een Logic Analyser ook nog op andere plaatsen verbinding gemaakt worden bijvoorbeeld via databus buffers, adreslijnen etc.
Het grote voordeel van deze opzet is de 'processor onafhankelijkheid'. Wanneer er een andere processor gekozen wordt dan is alleen een andere kabel nodig en moet er natuurlijk andere software in de analyser geladen worden. De debug-tool is zo direct weer gereed voor een ander project. Ook kan de gebruiker een eigen specifieke disassembler of data interpreter schrijven (voor bijvoorbeeld ASICs met een specifieke processor) en zo vertrouwelijke informatie beschermen. Wanneer meer kanalen nodig zijn kan dit door een analyser-board bij te plaatsen.

Voorwaarde voor deze methode is dat de Logic Analyser niet alleen 'Disassembler' en 'Symbolic Adresses' ondersteunt, maar daarnaast ook simultaan High Level Language (HLL). Deze HLL werkt op hetzelfde platform als de Logic Analyser software (Windows 3.11, 95 of NT) en geeft de source code weer die door de processor gebruikt is, gerelateerd aan de real-time data van de Logic Analyser.

De Logic Analyser fungeert als een real-time trace, zonder het target te beïnvloeden. De HLL-manager is modulair zodat er bij processor wijziging alleen een symbol- of data-interpreter herladen hoeft te worden. In de source code kan aangegeven worden waar er getriggerd moet worden. Ook Break-Points worden op die manier ingesteld (zoals gebruikelijk bij Debugging tools).

Extra gebruikersmogelijkheden worden geboden door triggerlijnen met instelbare trigger levels. Met dit triggersignaal kan dan een interrupt op het target gerealiseerd worden. De applicatie wordt dan gestopt en een 'Target Monitor' wordt geactiveerd. De monitor stuurt nu de relevante informatie naar de debugger en de debugger displayed de source code, de assembler listing, symbol informatie, de status van interne registers, globale en lokale variabelen, RAM en ROM status enz. parallel aan de real-time trace van de Logic Analyser. De monitor kan nu gebruikt worden om variabelen te manipuleren, stap voor stap door de source- of assembler-code te lopen, een nieuwe applicatie te laden of de huidige opnieuw te starten. Wanneer een Break-Point gezet wordt op een interne variabele, kan de analyser daar niet direct op triggeren. Om dat te doen wordt er op het Break-Point een sprong gemaakt naar de Target Monitor, wat een triggeradres beschikbaar maakt. In beide hier beschreven gevallen stopt het target op het triggerpunt en beïnvloedt het real-time gedrag van het target. Als de real-time situatie niet door het debugging systeem beïnvloed mag worden is er de mogelijkheid om het Trigger-Out signaal van de Logic Analyser niet aan te sluiten, wat dan resulteert in het weergeven van een real-time trace op het scherm. Interne variabelen en dergelijke blijven onzichtbaar (of er moet een Watch-Point gezet worden). Wanneer een Watch-Point bereikt wordt dan wordt de waarde door de Target Monitor aan external RAM gegeven en is dus beschikbaar voor de Logic Analyser. Dit geeft de analyser een gedefinieerd trigger punt.

Een debugging tool zoals hier beschreven bestaat uit een Logic Analyser, een PC-debugger en een Target Monitor. Deze combinatie geeft vrijwel dezelfde functionaliteit en comfort als een In-Circuit Emulator maar met de voordelen van de aansluitmogelijkheden, flexibiliteit en de onafhankelijkheid van het processor platform.

Rob Strik
Tech 5
Tel: 0184-615551

Distributed Emulation for Real-time Embedded Software Development

Men ziet dat in digitale systemen de software een steeds grotere rol gaat spelen. Waar vroeger veel aandacht werd besteed aan de hardware, ziet men nu dat vaak het design-team voor een groot gedeelte uit software engineers bestaat. Software maakt een produkt flexibel. Het is makkelijker uit te brengen met verschillende functionaliteiten en kan makkelijker worden opgewaardeerd met nieuwe mogelijkheden. Dit maakt dat de levensduur van een produkt kan worden verlengd en daarmee de return-on-investment kan worden zeker gesteld.

Verder zien we ook steeds meer dat de ontwikkeling van hard- en software parallel loopt. Simulatie-paketten maken het testen van delen van een produkt mogelijk en daarmee is het zogenaamde concurrent engineering een feit geworden. Maar nog steeds blijken er problemen te ontstaan bij de integratie van beide. En vooral die fase in de ontwikkeling vraagt om de benodigde hulpmiddelen. Hardware engineers zullen in deze fase waarschijnlijk terugvallen op de logic analyzer. Het instrument is uitermate geschikt om tijdrelaties weer te geven tussen diverse signalen en zodoende probleemgebieden aan te geven. Verder kan gekeken worden naar de code die ge-executeerd wordt. Dit kan in hex of met behulp van microprocessor-ondersteuning in assembler.

De software engineer zal waarschijnlijk niet terugvallen op een logic analyzer, maar eerder een in-circuit emulator kiezen. Dit type van instrument is uitermate geschikt voor het debuggen van real time embedded software en heeft voldoende mogelijkheden om ook tijdens de integratie fase zijn diensten te bewijzen.
Een emulator heeft in het algemeen een logic analyzer funktie, maar voegt daar twee funkties aan toe:

1) Run-control geeft de gebruiker de mogelijkheid om code te stoppen om vervolgens naar de situatie in de processor of controller te kijken. Men kan registers of geheugenwaarden uitlezen. Verder kan men stap voor stap door de code lopen om te zien welke uitwerking de geschreven software heeft.

2) Overlay-memory kan tijdelijk de ROM of RAM van het target bord vervangen. Tijdens het debuggen van de code kunnen uiteraard nog veranderingen worden aangebracht. Met behulp van overlay memory is het niet nodig om in die fase steeds PROM's opnieuw te programmeren. Verder is dit geheugen soms dual-ported. Dit houdt in dat het bekeken kan worden zonder dat de processor daarvan iets merkt. Zo kunnen variabelen bekeken worden tijdens het draaien van de software zonder interupties in de executie ervan. In sommige targets is dit een must omdat het anders crashed.

Een emulator is echter een instrument dat specifiek voor een bepaalde chip ontwikkeld wordt. Door de modulaire opbouw kan het soms zo zijn dat hij kan worden omgebouwd als in een volgend project een nieuwe processor wordt gekozen, maar het blijft toch een minder algemeen inzetbaar instrument dan bijvoorbeeld een oscilloscoop of een logic analyzer.

Een duidelijk aanwezige trend is de steeds groter wordende diversiteit van processoren en controllers. Steeds vaker zien we dat een groot aantal derivaten van een bepaalde architectuur wordt geintroduceerd. Maar ook zien we dat het aantal verschillende architecturen dat gebruikt wordt toeneemt. Zelfs chip-fabrikanten komen met meerdere produktlijnen parallel uit om de verschillende applicatie gebieden te adresseren. In het streven naar steeds meer applicatie gerichte oplossingen wordt ook het gebruik van een processor als onderdeel van een ASIC steeds belangrijker. De grote keuze maakt het voor een ontwerper mogelijk om de meest geschikte chip voor zijn toepassing te kiezen. Voor de emulator-fabrikant wordt het echter een steeds grotere uitdaging om al deze IC's tijdig te ondersteunen.

De grote vrijheid in keuze van technologie heeft tot gevolg dat zowel de hardware- als de software ontwerper voor een gegeven ontwerp uiterst efficient te werk kan gaan. Het heeft echter ook tot gevolg dat een gekozen processor minder lang zal meegaan. Met name core-on-ASIC oplossingen zullen per ontwerp leiden tot andere pen verdelingen of zelfs tot een nieuwe behuizing. Al deze veranderingen zorgen ervoor dat ten behoeve van de processor (of behuizing) specifieke delen van de instrumentatie vervangen moeten worden. Bij een logic analyzer is dit een minimale investering, bij een emulator daarintegen is dit een signifikant deel van de totale kosten van het instrument. Voor die ontwerpen die gebruik maken van processoren met een grote rekencapacitiet, vaak verbonden aan een hoge klok frequentie, wordt de invloed die de instrumentatie uitoefent op het ontwerp belangrijk. Op dit gebied heeft de logic analyzer een duidelijk voordeel. De probing nodig voor een funktionele logic analyzer aansluiting is minderbelastend dan die voor een in circuit emulator.

De bovengenoemde zaken zijn voor Hewlett-Packard reden geweest om te zoeken naar een oplossing met de flexibiliteit van een logic analyzer en de kracht van de in circuit emulator, de Distributed Emulator. Bij een distributed emulator worden de funkties van een in circuit emulator opgesplitst in afzonderlijke, met standaard instrumenten invulbare, componenten. De logic analyzer zorgt voor de real time analyse mogelijkheden. Voor run control wordt gebruik gemaakt van de mogelijkheden van de al regelmatig aanwezige on-chip debugger (N-wire interface, bv. BDM bij de Motorola 68360), of een ROM monitor. Overlay geheugen kan worden ingevuld door het vervangen van de ROM's door RAM, of door een ROM vervanger. Belangrijk bij een Distributed Emulator is de samenwerking tussen de verschillende delen, de manier van aansturing en data presentatie. Bij de distributed emulator oplossing is de logic analyzer zo aangepast dat de gebruiker de data in de gebruikte programmeertaal gepresenteerd krijgt en de instellingen direct vanuit de source gemaakt kunnen worden. De analyzer kan gebruikt worden om, in real time, het target te bekijken en bij het voorkomen van een bepaald verschijnsel de processor te stoppen. Nadat de processor stopt met het uitvoeren van het programma gaat de controle terug naar de gebruikte debugger.
De verschillende onderdelen van de distributed emulator kunnen grotendeels aangepast worden aan de eisen van de gebruiker. De logic analyzer kan varieren in snelheid, geheugendiepte en analyse capacitieten. De run control kan varieren tussen een ROM monitor en het gebruik van on-chip mogelijkheden. De debugger en het gebruikte taalsysteem zijn geheel naar keuze van de gebruiker. Zo kan gekozen worden voor de meest optimale oplossing voor het ontwerp.
Aanpassingen voor een volgend ontwerp kunnen eenvoudig gemaakt worden zonder dat de komplete instrumentatie afgeschreven dient te worden. Een verandering van processor technology kan op deze manier eenvoudig doorgevoerd worden, de gehele bediening en de gebruikers mogelijkheden blijven echter gelijk.

Kortom, de Distributed Emulator maakt het mogelijk om een diversiteit van processoren te ondersteunen. Het geeft een tijdige en flexibele oplossing, naast de traditionele in-circuit emulator en is ook makkelijk in te zetten in volgende projecten.

Hewlett Packard Nederland B.V.
Bert Esser
Tel: 020-5477502

Keuze-criteriums bij het opzetten van real-time en Embedded Software-ontwikkelingen.

Inleiding
De toepassing van micro-elektronica bij de ontwikkeling van een product neemt nog altijd een belangrijke plaats in. In de nog steeds groeiende elektronica-markt zijn er drie zaken vooraf zeker:

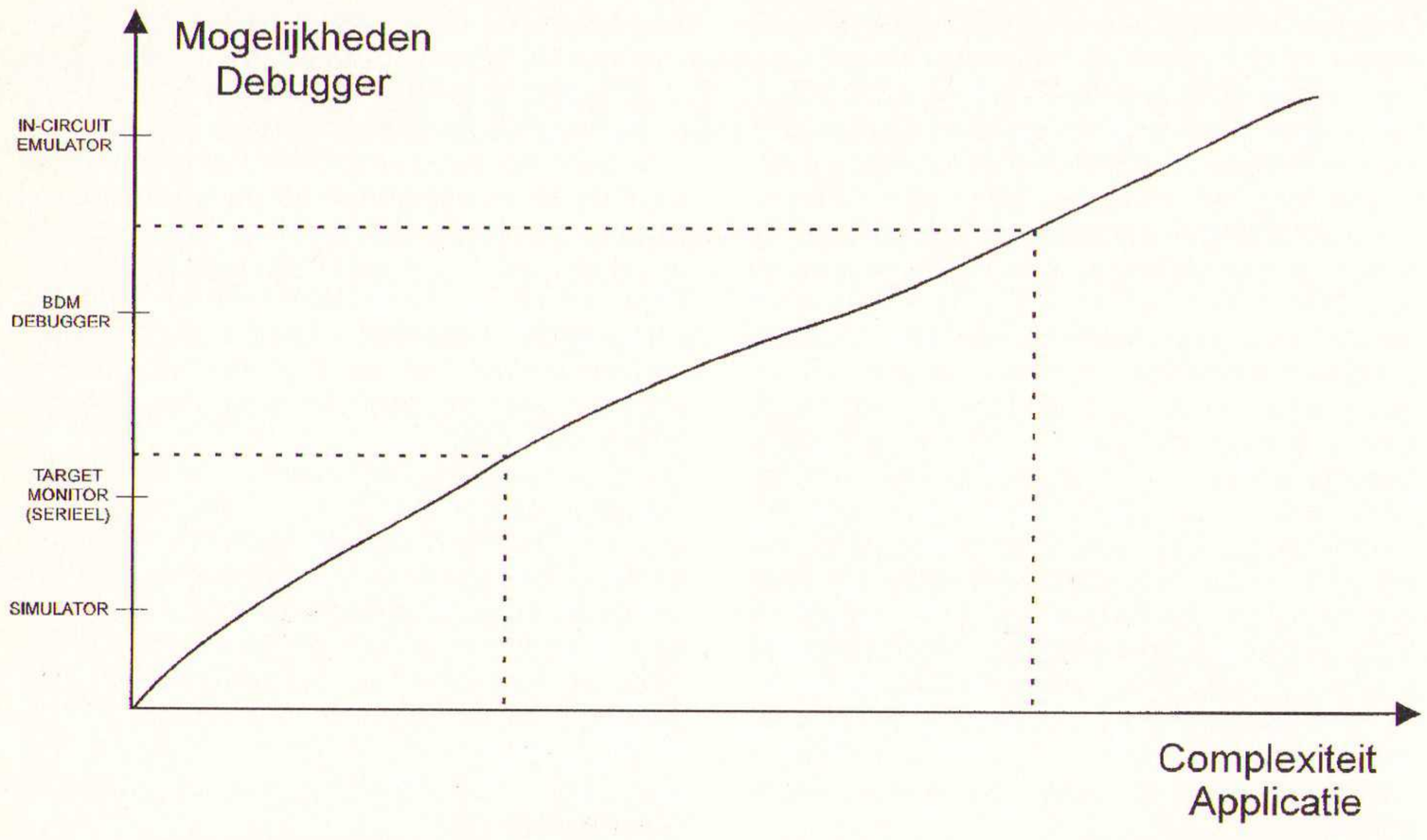
- consumenten worden veeleisender ten aanzien van nieuwe producten
- product life-cycles' worden steeds korter
- (te) laat op de markt komen met een nieuw product kan fataal zijn.

Dit betekent voor de ontwikkeling van zowel voor bestaande als voor nieuwe producten dat er steeds sneller dient te worden ingespeeld op nieuwe technologieën.

Gelukkig komen er steeds meer (nieuwe) ontwerp-hulpmiddelen (tools) beschikbaar welke essentieel zijn om de toenemende complexiteit te kunnen beheersen en een kortere 'time-to-market' van een project te kunnen realiseren. Hierbij wordt de kans op vertragingen zoveel mogelijk gereduceerd. Kortom de productiviteit neemt aanzienlijk toe. In de praktijk blijkt vaak dat de hier bespaarde tijd voor een groot deel wordt geïnvesteerd in de kwaliteit van een product, gebruik makende van de mogelijkheden die de desbetreffende tool te bieden heeft. De afweging welke tool voor bepaalde werkzaamheden nu het beste gebruikt kan worden, blijkt in de praktijk afhankelijk te zijn van een aantal factoren. Zo vallen te noemen, de technische mogelijkheden en beperkingen welke bepaalde tools met zich meebrengen. De financiële consequentie, een tweede factor, kan men als direct hiervan afgeleid beschouwen. Een derde factor die in een groeimarkt altijd een belangrijke rol speelt, is de factor kennis. Ook kennis is, naast technische mogelijkheden en financiële consequenties, een variabele die uiteindelijk bepalend is voor de vraag: zelf doen of uitbesteden ?

6.2 Huidige technische mogelijkheden

In het gebruik van tools, welke een steeds hogere kwaliteit en een steeds groter aantal mogelijkheden uitstralen, is een aantal trends waar te nemen.
Waar vroeger vrijwel direct op een laag niveau in de hardware werd gewerkt, is de trend dat het abstractieniveau vandaag de dag steeds hoger wordt. Microprocessor-onafhankelijkheid, productiviteit en kwaliteit zijn hierbij sleutelbegrippen.
Door het overweldigende aanbod van de bestaande en in een hoog tempo nieuw ontwikkelde en te ontwikkelen microprocessoren en microcontrollers van een groeiend aantal aanbieders anderzijds, ontstaat er een situatie waarvan de product-ontwikkelaar uiteindelijk kan pro-



fiteren. De keuze van de logica kan namelijk steeds beter worden afgestemd op een project.

Uiteraard dienen serieuze tool-leveranciers op beide trends in te spelen.
Beperken we ons hier tot real-time en embedded software ontwikkeling, dan kunnen we stellen dat we te maken hebben met afzonderlijke hulpmiddelen welke gezamenlijk een zogeheten 'toolchain' vormen:
- design-tools of CASE-tools: een methode wordt omgezet in programma-code (bijvoorbeeld ANSI-C)
- compilers/assemblers: high level- en source level programma-code wordt vertaald naar machinecode
- debuggers: voornamelijk bedoeld voor het opsporen van fouten waarbij machine code

via een standaard formaat wordt geladen en vanaf het hoogste niveau (high level language) kan worden geanalyseerd, waarbij kan worden vastgesteld waar een eventuele aanpassing of optimalisatie het meest zinvol is. Afhankelijk van de soort debugger kan de code worden gesimuleerd of worden geëmuleerd.

Naast de verschillende media en eventuele netwerk-omgevingen die worden ondersteund om de tools te kunnen bedienen, kent elk hulpmiddel ook specifieke eigenschappen:
A) design tools:
- koppelingen naar grafische ontwerp tools
- kwaliteit (compactheid, snelheid, etc.) van de te gegenereren (veelal ANSI-C) programmacode
- simulatie voor een logische controle van de code (dead-ends, etc.)
- het genereren van dokumentatie

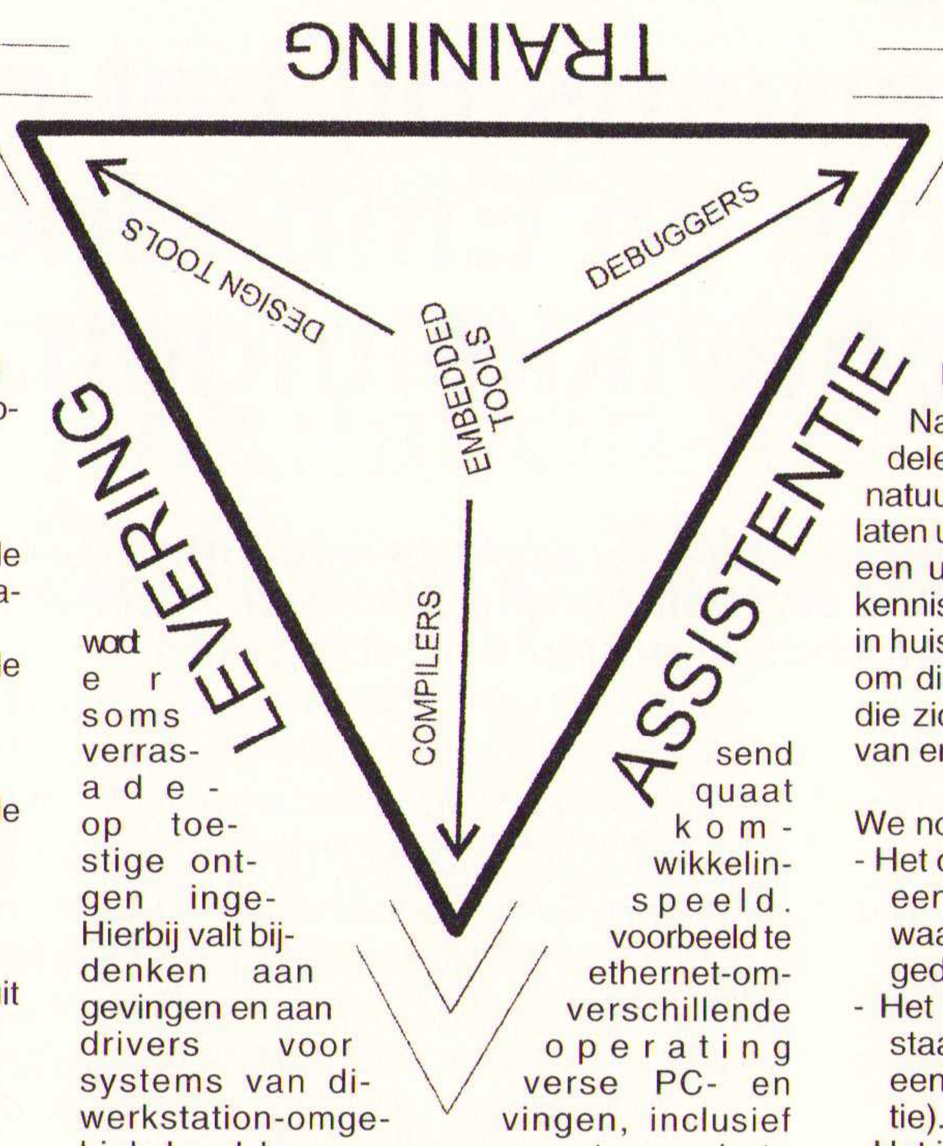
B) compilers/assemblers:
- kwaliteit (compactheid, snelheid, etc.) van de te genereren machinecode
- optimalisatie
- foutcode-generatie
- instelmogelijkheden

C) debuggers (simulators, monitors, in-circuit emulators, etc.)
- high/source level ondersteuning
- real-time gedrag
- up- en downloadtijd
- wel of niet real-time tracing
- aantal ondersteunde devices en de mogelijkheid tot upgrading
- maximaal ondersteunde snelheden van de devices
- mogelijkheden als (real-time) 'code-coverage' analyse en 'program performance' analyse.

De kracht van een goede ontwikkelomgeving zit 'm in feite in de samenbouw van componenten uit A), B) en C) tot een geïntegreerd geheel waarbij alle componenten op het desbetreffende project zijn afgestemd. Verder kent elk project een bepaalde omvang, zodat er bij grotere ontwikkelprojecten ook in een parallelle opzet door verschillende personen, al dan niet in een netwerk-omgeving, optimaal van de tools gebruik gemaakt kan worden.

Op dit moment wordt er door de huidige toonaangevende tool-leveranciers erg veel aandacht besteed aan een verdere integratie van de afzonderlijke tools. De verscheidenheid aan keuzemogelijkheden zorgt ervoor dat er goed op bestaande en nieuwe situaties kan worden ingespeeld.

Afhankelijk van de visie van de tool-fabrikant wordt er soms verrassend op toestige ontgen ingeHierbij valt bijdenken aan gevingen en aan drivers voor systems van diwerkstation-omgehigh level language ondersteunen compilers.
Verder kunnen we hierbij ook denken in de richting van de verscheidenheid aan te ondersteunen microprocessoren/-controllers, waarbij de trend duidelijk is: toenemende integratie, hogere snelheden en kleinere behuizingen.
De opmars van Windows-omgevingen met een gestandaardiseerd 'look and feel' user-interface hebben de basis gelegd voor de verdere integratie tussen de verschillende tools in een toolchain.



6.3 Financiële consequenties

Vaak gaan technologische ontwikkelingen gepaard met een zich voor de eindgebruiker gunstig ontwikkelende prijs. Waar zo'n vijftien jaar geleden een hoge prijs werd betaald voor een stand-alone ontwikkelomgeving met een beperkte functionaliteit, wordt er nu voor een fractie van dit bedrag een geïntegreerde toolset geleverd voor een bepaalde microprocessor- of microcontroller-familie met uitgebreide mogelijkheden. Uiteraard inclusief de hiervoor beschreven state-of-the-art features.
Hierbij is vooral de flexibiliteit toegenomen; de meeste tools zijn upgradable, zowel qua mogelijkheden als qua groei naar andere en/of hogere microprocessor/-controller families.

Er is tevens een taak weggelegd voor de leverancier van de ontwikkeltools om de klant waar nodig een totaaloplossing te bieden. Hierbij wordt vaak een oplossing gezocht in een mogelijke inruil van de apparatuur (of delen hiervan), leasing van apparatuur en huurmogelijkheden.

Omdat de time to market steeds korter wordt en dit ondermeer mogelijk wordt gemaakt door de kwaliteit en de mogelijkheden die de tools ons bieden, wordt er ook gebruik gemaakt van de mogelijkheid om apparatuur voor een korte periode te huren. Door de relatief lage aanschafprijs van de tools is dit meestal alleen interessant voor zeer kortlopende projecten. Gezien het kortstondige gebruik komt hierbij vaak tevens een training om de hoek kijken.

Naast diverse technische afwegingen zien we dus ook dat er financiële afwegingen dienen te worden gemaakt om tot een verantwoorde investering over te kunnen gaan. Uiteraard is iedere situatie verschillend, zodat de uiteindelijke beslissing steeds opnieuw dient te worden gemaakt. In ieder geval blijft de vraag: hoe kan de verhoogde productiviteit voor mijn bedrijf iets opleveren ?

6.4. Zelf doen of uitbesteden.

Naast het zelf direct investeren in hulpmiddelen voor een nieuwe ontwikkeling, zijn er natuurlijk ook mogelijkheden om een project te laten uitvoeren door externe specialisten. Dit kan een uitkomst zijn om bijvoorbeeld capaciteit, kennis en tools al dan niet tegen een 'fixed price' in huis te kunnen halen. Vandaag de dag zijn er om die reden veel aanbieders op de IT-markt die zich meer en meer op de groeiende markt van embedded software richten.

We noemen hier een aantal mogelijkheden.
- Het door middel van een toegesneden cursus een bepaalde technologie eigen maken, waardoor de ontwikkeling zelf kan worden gedaan (training).
- Het op laten zetten van een nieuwe of bestaande ontwikkeling, waarna er na afloop een kennisoverdracht plaatsvindt (assistentie).
- Het inhuren van een tijdelijke embedded software-specialist (soms zelfs inclusief de benodigde apparatuur) voor een vooraf afgesproken termijn en opdracht (detachering).
- Het in zijn geheel uitbesteden van een project aan een extern ontwikkelbureau (project basis; fixed price, fixed date).

Vaak wordt er vooral bij consultancy en engineering, gezocht naar partners met een desbe-

treffend specialisme. Ook hier is een taak weggelegd voor de tool-leverancier, die door een toenemende betrokkenheid met zijn afnemers, vaak zeer goed op de hoogte is van bepaalde kennis in specifieke deelgebieden.

6.5 De rol van een tool-leverancier bij het opzetten van real-time en Embedded Software ontwikkelingen

Zoals gezegd, houdt een leverancier zich vandaag de dag niet alleen bezig met het leveren en ondersteunen van een breed aanbod van goed op elkaar afgestemde hulpmiddelen. Van een goede leverancier mag worden verwacht dat de ondersteuning wordt aangevuld met dienstverlening.

Dienstverlening waaronder in de praktijk mag worden verstaan:
- upgrading door eventuele inruil van apparatuur;
- een uitgebreid trainingsprogramma met naast uiteraard producttrainingen, ook aandacht voor technologietrainingen;
- referenties naar de markt van zowel toeleveranciers van producten (chips, halffabrikaten, etc.), alswel van diensten (engineering, consultancy, etc.).

De productgroep ontwikkelsystemen van TRITEC Benelux B.V. streeft al een kleine tien jaar naar een kortere 'time to market', een hogere kwaliteit en een betere productiviteit bij de ontwikkeling van embedded software, door het gebruik van professionele hulpmiddelen te stimuleren.

TRITEC Benelux B.V.
Antoniuslaan 1, 3341 GA
Postbus 212, 3340 AE
Hendrik Ido Ambacht
Tel. ++31(0)78 - 6816133
Fax. ++31(0)78 - 6820030
e-mail : development-tools@tritec.nl

Robert de Voogt
Tritec Benelux BV
078-6816133
development-tools@tritec.NL

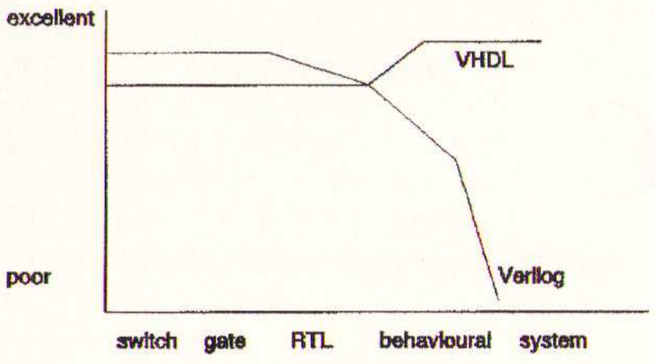
Het hoe en waarom van hogere beschrijvingstalen voor de ontwikkeling van elektronica

1.1 Algemene inleiding

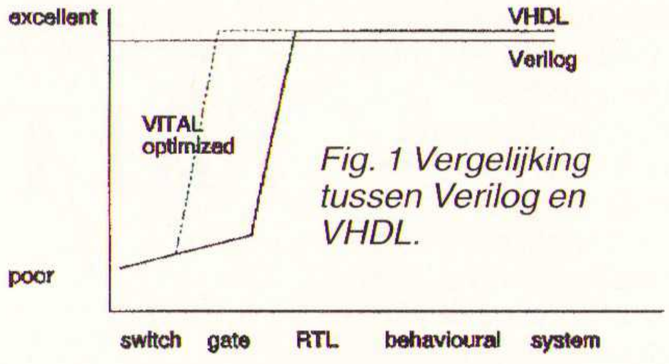
Het gebruik van hogere beschrijvingstalen zoals VHDL wordt steeds populairder. Niet alleen grote multi-nationals gebruiken deze krachtige ontwerptechniek maar ook het midden en klein bedrijf is succesvol met VHDL. Na een korte inleiding van de taal VHDL vindt u een praktijkvoorbeeld van een "midden en klein bedrijf" dat door de invoering van VHDL uiterst succesvol in zeer korte tijd een FPGA realiseerde. De inleiding van de taal VHDL is afkomstig uit het boek "VHDL '87/'93 en voorbeelden" van Bert Molenkamp. In dit boek vindt u een complete beschrijving van de VHDL beschrijvingstaal. Het boek is bedoeld voor ontwerpers van digitale systemen met programmeervaardigheden. Zowel de "oude" VHDL standaard Std. 1076-1987 als de huidige standaard Std. 1076-1993 wordt erin bescheven. De 'oude' standaard wordt nog nadrukkelijk behandeld, omdat veel van de huidige EDA omgevingen nog niet de nieuwe standaard ondersteunen.

1.2 Inleiding VHDL

Na de ratificatie door de IEEE van de hardware beschrijvingstaal VHDL Std 1076-1987 heeft het gebruik ervan een enorme vlucht genomen in de industrie. IEEE standaarden moeten minimaal om de vijf jaar opnieuw geratificeerd worden. Inmiddels is de standaard dan ook vervangen door de Std 1076-1993. Beide worden ook vaak aangeduid met VHDL'87 en VHDL'93, de laatste soms ook wel met VHDL'92. Dit is het oorspronkelijk geplande jaar van de nieuwe standaard. Voor alle duidelijkheid er is slechts één VHDL standaard namelijk de meest recente VHDL 1076-1993. Een subset van de VHDL beschrijvingen is ook synthetiseerbaar. Synthese aspecten maken echter geen deel uit van de VHDL standaard. Helaas beperken nog veel gebruikers van VHDL zich tot de subset van VHDL welke synthetiseerbaar is door haar/zijn synthese-tool en gaan daarbij voorbij aan de kracht van VHDL als specificatietaal van het nog te ontwerpen systeem.
Binnen de IEEE bestaan een aantal werkgroepen, die gecoördineerd worden door de DASC (Design Automations Standards Committee), actief met nauw verwante onderwerpen, o.a. "VHDL Analoge Extention Working Group" en de "VHDL Synthesis Package Working Group". De "VHDL Synthesis Package Working Group" moet een einde maken aan de wildgroei op het gebied van synthese van VHDL. Elk synthese systeem gebruikt zijn eigen interpretatie en typering voor bit_vectoren en de operaties die hierop gebaseerd zijn, waardoor er nauwelijks sprake is van portabiliteit tussen de verschillende synthese systemen. Een tweetal packages zullen worden afgeleverd, één is gebaseerd op het type bit (package numeric_bit) en de andere gaat uit van de IEEE package std_logic_1164 (package numeric_std). Beide packages zullen naar verwachting in het voorjaar van 1996 een IEEE standaard zijn. De "Verilog Analysis and Standardization Working Group" kan in zekere zin gezien worden als een concurrent van VHDL. De basis van Verilog is in de wintermaanden van 1983-1984 gelegd door Phil Moorby en op de EDA markt gebracht in 1985 door Cadence Design Systems. In 1990 zijn de rechten public domain geworden en sindsdien heeft de onafhankelijke *Open Verilog International* (OVI) groep er bekendheid aan gegeven. Nu Verilog ook een IEEE standaard gaat worden ligt het voor de hand dat beide hardware beschrijvingstalen naast elkaar gebruikt zullen gaan worden. Naast elkaar omdat de beide talen niet helemaal hetzelfde doel hebben. Verilog is bedoeld vanaf RTL-niveau (Register Transfer Level) *tot en met* de realisatie terwijl VHDL bedoeld is vanaf specificatie *tot* aan de realisatie. Veel EDA omgevingen zijn al druk bezig met de voorbereidingen voor de co-simulatie van VHDL en Verilog beschrijvingen, immers het ligt immers voor de hand dat op een hoog niveau in VHDL wordt ontworpen terwijl de realisatie in Verilog is beschreven. Janick Bergeron heeft een vergelijking gemaakt tussen VHDL en Verilog met betrekking tot de simulatiesnelheid en het modeleren.
In de VHDL standaard is niet aangegeven op welke wijze ASIC bibliotheken met al hun timingsaspecten beschreven moeten zijn. In mei 1992 werd tijdens de VHDL International User's Forum in Scottsdale besloten dit probleem op te pakken. Uitgangspunt hierbij was het gebruik



Modelering



Simulatie

Fig. 1 Vergelijking tussen Verilog en VHDL.

maken van het SDF (standard delay format), welke ook bij Verilog wordt gebruikt, en het gebruik maken van de al ontwikkelde Std_timing package van de VHDL Technology Group en de technieken van "Ryan & Ryan". Verder moest voor een groot draagvlag worden gezorgd bij de industrie.
Dit initiatief kreeg dan ook de naam VITAL: VHDL Initiative Towards ASIC Libraries. In maart 1994 kwam de "VITAL 2.2b Model Development Specification" uit en momenteel wordt al druk gewerkt aan VITAL 3.0 dit zal een IEEE standaard worden. PAR 1076.4 werkt aan deze standaard. In VITAL 2.2b wordt onder andere een tweetal packages beschreven, *VITAL_Timing* en *VITAL_Primitives*, waarin respectievelijk een groot aantal timing procedures zijn beschreven en primitieve operaties op bit niveau zijn beschreven. Met daaraan voor de EDA tools de uitdaging de package body's zo te implementeren dat de simulaties snel zijn. Verder is er een document beschreven waaraan de gebruiker zich moet houden. Dit document en de packages zijn via ftp te verkrijgen op adres "vhdl.org". Een consequentie van het gebruik van VITAL is ook aangegeven in figuur 1.

Eigenschappen van VHDL

VHDL heeft een groot aantal aantrekkelijke eigenschappen:
Hiërachie wordt ondersteund.
De gebruiker is vrij om haar/zijn eigen ontwerpmethode te gebruiken: top-down, bottom-up, etc.
Technologie onafhankelijk. Dit betekent concreet dat een ontwerp van enkele jaren geleden zonder problemen hergebruikt kan worden in een andere technologie.
Ondersteuning van verschillende beschrijvingsstijlen, zoals een finite state machine, een algoritme, boleaanse vergelijkingen etc.
"Second-sourcing" eenvoudiger. Aangezien de taal EDA Tool onafhankelijk is kan zonder veel problemen worden omgeschakeld naar een andere omgeving. Belangrijk is het dat ook de testomgeving in VHDL is beschreven. Hiervoor kan bijvoorbeeld gebruik worden gemaakt van WAVES.
Verhoging van het abstractieniveau waardoor een betere beheersing van de complexiteit mogelijk is. Verder kunnen details aan de synthese-tools worden overgelaten.
Het vroegtijdig opsporen van fouten doordat de specificatie al te simuleren is en elke ontwerpstap tegen de simulatie is te testen.
Large Scale Integration (LSI) modelering is eenvoudig door het gebruik van 'componenten', 'functies', 'procedures' en 'packages'.
VHDL legt geen beperking op aan de grootte van het ontwerp.
'Test benches' kunnen in dezelfde taal worden beschreven en zijn dus overdraagbaar.
Dankzij de 'generics' en 'attributes' is backannotation eenvoudig. Hiervan wordt ook gebruik gemaakt in VITAL.
VHDL is een sterk getypeerde taal. Nieuwe datatypes zijn door de gebruiker zelf te definiëren.

3.1.4 Inleiding in de beschrijvingstaal VHDL

Hoewel VHDL geen imperatieve taal is lijkt de syntax van de nog te behandelen processen veel op die van sterk getypeerde imperatieve talen, zoals MODULA-2. In de voorbeelden zijn de gereserveerde woorden van de taal dik gedrukt weergegeven en in de ASCII files worden hoofdletters gebruikt. VHDL maakt geen onderscheid tussen kleine en hoofdletters.
Als eerste zal de kern van de taal worden besproken: communicerende processen. Hierbij wordt ook aangegeven hoe deze processen gesimuleerd worden. Vervolgens wordt een summiere inleiding gegeven in de taalconstructies.

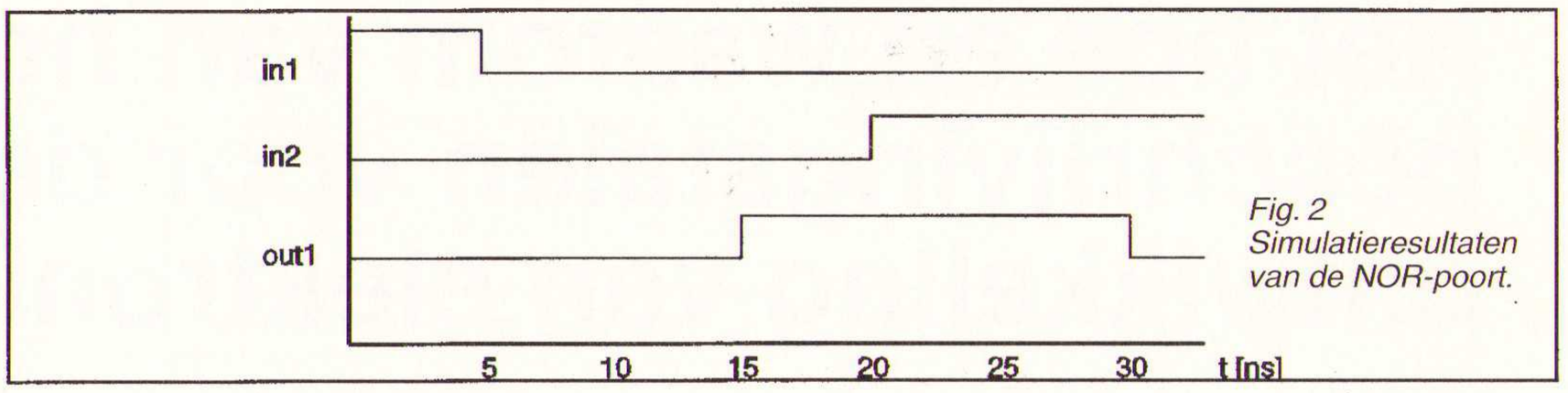


Fig. 2 Simulatieresultaten van de NOR-poort.

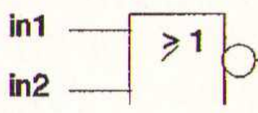
1.4.1 Communicerende processen

Hardware kan worden beschouwd als een verzameling van parallelle processen die met elkaar communiceren. Elk proces berekent continu op basis van zijn ingangssignalen de daarbij behorende uitgangssignalen. In een *nor* poort vindt als het ware continu een berekening plaatst, ook al zijn de ingangssignalen stabiel. Voor het simuleren van een *nor* poort is het echter voldoende om alleen de veranderingen door te rekenen (figuur 3). Dus wanneer een ingangssignaal verandert, moet een nieuw uitgangssignaal worden berekend. Dit simulatiemodel, dat ook in VHDL wordt gebruikt, noemt men *event-driven simulation*.

Figuur 3 geeft een VHDL beschrijving van een nor poort. Deze beschrijving bestaat uit twee delen: de ENTITY en de ARCHITECTURE.
De entity geeft de naam van het component en wat de in- en uitgangssignalen zijn. Bij deze signalen moet ook het type worden opgegeven. In dit voorbeeld zijn beide ingangsignalen en het uitgangsignaal van het type bit. Dit type is een standaard type van VHDL en is gedefinieerd als een enumeratie: *type bit is ('0','1');*. In de entity-beschrijving kan ook nog statische informatie worden opgenomen middels een *generic* statement. In dit voorbeeld geeft het aan dat de poort default een vertraging heeft van 10 ns. Beschouw de generic voorlopig als een constante. De generic en de port statements mogen achterwege blijven.
De architecture bepaalt het gedrag tussen de in- en uitgangssignalen in dit voorbeeld door middel van een procesbeschrijving. Een procesbeschrijving is een sequentiële beschrijving te vergelijken met die van de imperatieve talen. Voor de *nor* poort kan het gedrag door middel van een eenvoudige procesbeschrijving worden bepaald. Deze geeft aan dat het uitgangssignaal OUT1 de logische nor is van IN1 en IN2 en vervolgens wordt er gewacht op een verandering van IN1 en/of IN2. Na een verandering van een ingangssignaal vervolgt de executie van het proces zich weer na de wait statement en aan het einde gekomen van de procesbeschrijving vervolgt de executie zich aan het begin.

Figuur 4 geeft een VHDL beschrijving van een sr-latch. Tevens is in de entity-beschrijving commentaar opgenomen, door gebruik te maken van de '--'. Documenteren is een belangrijk onderdeel van het ontwerpproces. Wat een entity behoord te doen staat altijd als commentaar in de entity en hoe dat wordt verkregen als commentaar in de architecture.

```
entity nor_gate is
  generic (delay : time := 10 ns);
  port (in1, in2 : in bit;
        out1     : out bit);
end nor_gate;

architecture gedrag of nor_gate is
begin
  nor_gate:process
  begin
    out1 <= in1 nor in2 after delay;
    wait on in1, in2;
  end process;
end gedrag;
```

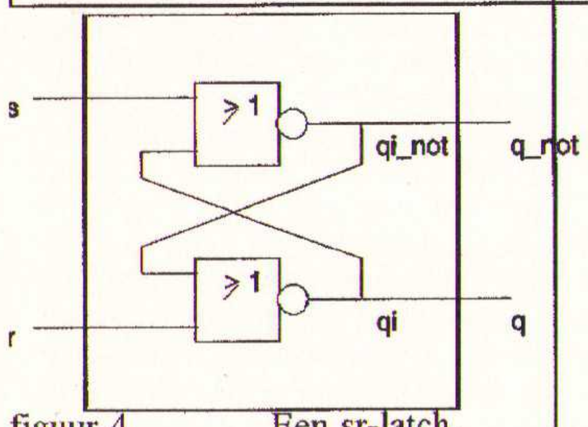


figuur 3 VHDL beschrijving van een nor poort.

Elke nor poort is beschreven met een procesbeschrijving. De volgorde waarin de beide processen zijn beschreven is niet belangrijk. Communicatie tussen beide processen kan alleen plaats vinden door middel van signalen. Hoe lokalesignalen worden gedeclareerd is ook in het voorbeeld aangegeven, deze declaraties bevinden zich tussen de gereserveerde woorden ARCHITECTURE en BEGIN. De signalen QI en QI_NOT zijn in deze beschrijving lokaal. Verder is aangegeven dat het signaal QI_NOT tijdens het initialiseren de waarde '1' krijgt. *Indien een signaal niet expliciet geïnitialiseerd wordt krijgt het de meest linkse waarde van het bereik van het signaal*. Signaal QI wordt niet expliciet geïnitialiseerd, daarom krijgt deze tijdens het initialiseren de waarde '0' (type bit is ('0','1');). Ook de ingangssignalen hebben een initiële waarde. De ingangen S en R zijn beide '0' aangezien ze van het type bit zijn.

Elke VHDL beschrijving wordt intern vertaald naar communicerende processen. In figuur 4 zijn de componenten NOR_GATE1 en NOR_GATE2 expli-

```
entity sr_latch is
  generic (delay : time := 10 ns);
  port (s,r      : in bit;
        q,q_not  : out bit);
  -- description of an sr_latch.
  --      s  r  |  q  q_not
  --      0  0  | unchanged
  --      0  1  |  0  1
  --      1  0  |  1  0
  --      1  1  |  0  0
end sr_latch;

architecture gedrag of sr_latch is
  -- architecture-beschrijving gebruikmakend
  --van twee processen.
  signal qi     : bit;
  signal qi_not : bit := '1';
begin

  nor_gate1:process
  begin
    qi <= r nor qi_not after delay;
    wait on r, qi_not;
  end process;

  nor_gate2:process
  begin
    qi_not <= s nor qi after delay;
    wait on s, qi;
  end process;

  q <= qi;

  q_not <= qi_not;

end gedrag;
```



figuur 4 Een sr-latch.

figuur 4 Een sr-latch.

ciet beschreven. Deze twee processen worden parallel uitgevoerd. Explicieteprocesbeschrijvingen zijn gekoppeld aan de gereserveerde woorden PROCESS en END PROCESS.

Een complex digitaal systeem beschrijven met and's, or's en inverters is niet gebruikelijk. In een dergelijk systeem kan meestal weer onderscheid gemaakt worden tussen de verschillende functionele blokken, zoals alu's (arithmetic logic units), geheugens etc. Niet alleen tijdens het beschrijven maar ook tijdens het ontwerpen van een digitaal systeem is het belangrijk dat geleidelijk aan meer detail kan worden toegevoegd. Door middel van een struktuurbeschrijving kan hiërarchie in het ontwerp worden aangebracht. Een struktuurbeschrijving in VHDL is een opsomming van componenten en hoe deze met elkaar verbonden zijn.

1.4.2 Korte samenvatting beschrijvingstaal VHDL

Een beschrijving van hardware bestaat uit een *entity* en één of meer *architectures*.

De entity legt de communicatie met de omgeving vast, ook kan hierin een generic worden opgenomen. In de entity-beschrijving wordt commentaar opgenomen *wat* de functie van de beschreven hardware is.

De architecture legt het gedrag vast van hardware. Als commentaar wordt globaal aangegeven *hoe* de gewenste functie is vervuld.

VHDL is gebaseerd op communicerende processen. Deze processen zijn impliciet of expliciet aanwezig. De volgorde waarin impliciete of expliciete processen zijn beschreven is niet belangrijk, de volgorde van de statements binnen een expliciete procesbeschrijving wel.

VHDL heeft een drietal objecten: signalen, variabelen en constanten. Communicatie tussen de processen kan alleen plaats vinden door middel van signalen. Alleen binnen een procesbeschrijving, functies en procedures mogen variabelen worden gebruikt.

Entiteiten kunnen als onderdeel van een andere architecture-beschrijving worden gebruikt door componenten te declareren en dit vervolgens door middel van een configuratie-specificatie te koppelen aan de entiteit.

Formele en actuele parameters worden gekoppeld door een positionele koppeling - een actuele parameter staat op de plaats van de formele parameter waarmee het gekoppeld moet worden - of expliciet door de formele en actuele parameters te koppelen m.b.v. de '=>'. Een combinatie is mogelijk, mits eerst de positionele koppeling is aangegeven.

Variabelen en signalen worden default geïnitialiseerd met de meest linkse waarde van het desbetreffende type.

Functies leveren slechts één resultaat op, aan een variabele of signaal. Procedures kunnen meer resultaten opleveren maar default alleen maar aan variabelen. In een package en package body kunnen types, functies, procedures, componenten e.d. worden gedeclareerd waarvan door middel van een 'use-clause' gebruik gemaakt kan worden. Door middel van de mode 'in', 'out' en 'inout' wordt aangegeven of er resp. sprake is van een in, uit of bidirectionele parameter is. Tevens zijn er nog de modes 'buffer' en 'linkage'. De laatste mode wordt zelden gebruikt.

1.4 Van een VHDL beschrijving naar een realisatie

Het ontwerpen van digitale systemen met behulp van VHDL is mogelijk van de specificatie tot en met een implementatie op logisch niveau. De realisatie zelf wordt in het algemeen niet in VHDL beschreven. Hiervoor waren ten tijde van de ontwikkeling van VHDL in 1980 reeds voldoende hulpmiddelen aanwezig. Na een aantal ontwerpstappen uitgevoerd te hebben in VHDL blijft er een beschrijving over die door een commerciële synthese-tool gerealiseerd kan worden. De VHDL beschrijving wordt omgezet in een realisatie (full custom, gate array, programmeerbare logica, etc.). De commerciële synthese-tools zijn grofweg te verdelen in twee groepen:

logische synthese
hoog niveau synthese

Bij een synthese systeem dat uitgaat van logische synthese wordt vaak een één op één afbeelding van VHDL naar een technologie uitgevoerd met een beperkt aantal combinatorische minimalisaties. Dit betekent dat de VHDL beschrijving al zoveel detail moet bevatten dat voor het synthetiseren al in grote lijn bekend is hoeveel registers e.d. gebruikt gaan worden.

Synthese systemen die hoog niveau synthese uitvoeren zijn in staat ook een aantal ontwerpbeslissingen op een hoger abstraktie niveau mee te nemen. Zo wordt bijvoorbeeld nagegaan of meerdere optellers zoals die beschreven zijn in de VHDL beschrijving niet gecombineerd kunnen worden, uiteraard zonder het gewenste gedrag te veranderen. Ook kunnen beschrijvingen waarin de toestanden impliciet aanwezig zijn gerealiseerd worden.

Vaak kan een ontwerp ingedeeld worden in een data-pad en een besturing voor dit data-pad. De besturing kan vaak beschreven worden in de vorm van een toestandsmachine, terwijl in een data-pad vaak weer componenten alsoptellers, multiplexers e.d. voorkomen. Er zijn dan ook synthese systemen die verschillende algoritmen gebruiken voor het synthetiseren van een data-pad en een besturing.

Inmiddels zal het duidelijk zijn dat er veel alternatieve VHDL beschrijvingen mogelijk zijn om hetzelfde gedrag te beschrijven. Synthese systemen ondersteunen echter slechts een subset van de mogelijkheden van VHDL en, helaas, niet dezelfde subset.

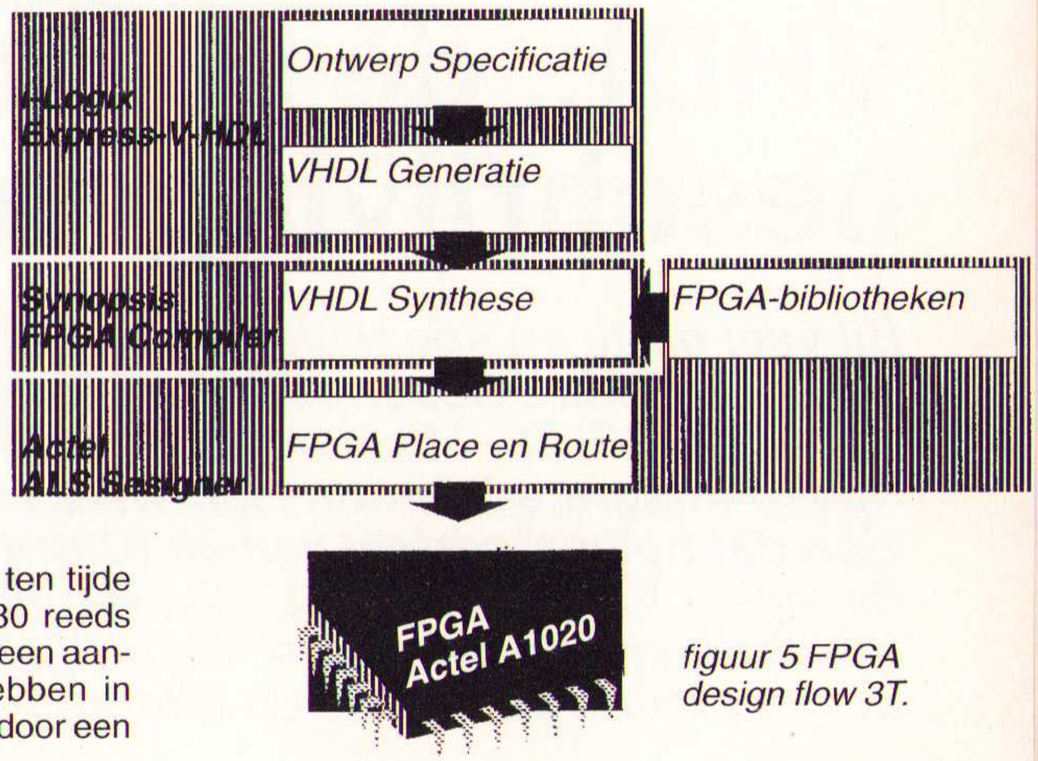
1.5.1 Wat is niet synthetiseerbaar?

In veel hardware beschrijvingstalen zijn alle constructies af te beelden op hardware. Het is de intentie van VHDL dat ook een specificatie moet kunnen worden beschreven. Taalconstructies die in een dergelijke beschrijving worden gebruikt hoeven niet synthetiseerbaar te zijn. Nagenoeg alle synthese systemen hebben geheel of gedeeltelijk problemen met:

expliciet opgegeven vertragingstijden
asynchrone beschrijvingen
wait statements
dynamische strukturen, recursie, loops en files
typering voor signalen
het event-driven simulatiemodel

1.5.2 'Hoog niveau' synthese

In de vorige paragraaf is aangegeven van wat



figuur 5 FPGA design flow 3T.

wel en niet synthetiseerbaar is. Aan de hand hiervan kan bepaald worden tot hoever je een beschrijving in VHDL moet ontwerpen voordat het betrouwbaar gesynthetiseerd kan worden. Er wordt getracht steeds betere synthese systemen te ontwikkelen waardoor de ontwerper steeds minder detail in de VHDL beschrijving hoeft aan te geven, waardoor de beschrijving vaak ook beter te begrijpen blijft.

1.6 Samenvatting

Niet alle VHDL beschrijvingen zijn synthetiseerbaar. In het algemeen geldt dat alle realisaties met simulatietijd niet te realiseren zijn. Dus expliciet opgegeven vertragingstijden worden genegeerd, ook attributen welke gerelateerd zijn aan de tijd worden niet ondersteund. Realisatie van dynamische strukturen en recursieve funkties is eveneens niet mogelijk. Verder wordt het event-driven simulatiemodel niet in alle opzichten gesynthetiseerd.
De meeste synthese systemen gaan uit van synchrone ontwerpen. In dat geval is het resultaat meestal 'correct by construction'. Verificatie, bijvoorbeeld door middel van simulatie, van het gerealiseerde tegen het gewenste gedrag blijft altijd wenselijk omdat de synthese software nooit foutloos is. Afhankelijk van het soort beschrijving (data-pad of besturing) worden soms ook verschillende algoritmen gebruikt om een optimaal resultaat te krijgen. Of snelheid of oppervlak belangrijk is kan de synthese-tool niet weten, daarom kan dit in de tool worden aangegeven.
EDA systemen hebben naast de standaard package vaak een groot aantal extra funkties en procedures waardoor beschrijvingen eenvoudiger worden. Deze funkties en procedures worden meestal ook door een synthese-tool ondersteund.

1.6 Praktijk voorbeeld: 3T uiterst succesvol in hoog niveau FPGA ontwerp

Het bedrijf 3T B.V. in Enschede heeft onlangs een FPGA (Field Programmable Gate Array) ontwerp gerealiseerd binnen 2 maanden, vanaf specificatie tot realisatie. En dat zonder dat men praktische ervaring had van de gebruikte top-down ontwerp methode en de toegepaste tooling!
3T is een ingenieursbureau dat in 1988 is ontstaan uit het Centrum voor Micro Elektronica (CME). 3T heeft momenteel meer dan 30 medewerkers en de hoofdactiviteit is ontwikkeling en produktie van micro-elektronica op klantenspecificatie. Bij de ontwikkeling wordt de structurele specificatie methode Yourdon toegepast.

In december 1995 ontstond de wens om een FPGA te ontwikkelen. Deze zou dienst doen als prototype, in de uiteindelijke produktie zal het ontwerp in een ASIC worden geïmplementeerd. Het was daarom zaak om hierbij de ontwikkeling van de FPGA rekening te houden, zodat het FPGA-ontwerp eenvoudig naar een ASIC kon worden omgezet.
Om aan deze eisen te voldoen, heeft TRANSFER EDS samen met 3T hiervoor het ontwerptraject opgesteld. Het gebruik van een technologie onafhankelijke beschrijvingstaal moest hiervoor de basis zijn. Het uitgangspunt is van het begin af aan een VHDL beschrijving geweest, omdat deze IEEE beschrijvingstaal wereldwijd ondersteund wordt en daardoor voor zowel de ontwikkeling van het FPGA prototype als voor de uiteindelijke ASIC gebruikt kanworden. Het VHDL ontwerp kan dus voor de uiteindelijke produktie worden overgedragen aan de ASIC ontwikkelaar.

Bij 3T was nog weinig praktische ervaring met VHDL aanwezig. Omdat het project onder tijdsdruk stond en een universele aanpak met VHDL een must was, is er uiteindelijk voor gekozen om het complete ontwerp grafisch in te voeren en te simuleren. Vervolgens wordt er automatisch VHDL gegenereerd en deze VHDL file wordt geïmplementeerd in een FPGA. Het complete ontwerp-traject is weergegeven in de figuur 5. Uit deze figuur zijn omwille van de duidelijkheid de verificatie stappen tussen alle fasen weggelaten.
Voor de grafische invoer van de toestandsdiagrammen werd gebruikt gemaakt van de tool ExpressV-HDL van i-Logix. Door deze tool te gebruiken is het mogelijk om direct het ontwerp te simuleren. Daarnaast zorgt ExpressV-HDL voor consistentie van het systeem. Het invoeren en simuleren van het ontwerp heeft voor veel duidelijkheid gezorgd. "Je kunt je zo op het gedrag van je ontwerp richten en niet op hoe de schakeling uiteindelijk gerealiseerd moet worden", aldus Norbert Beltman en Peter Orth van 3T.

Nadat de simulatie resultaten waren geverifieerd en correct bevonden, genereerde de ExpressV-HDL tool van i-Logix automatisch de synthetiseerbare VHDL code. Deze code is met behulp de FPGA Compiler van Synopsys gesynthetiseerd naar een Actel A1020 FPGA (2000 gates, 547 logische modules). De kracht van FPGA Compiler van Synopsys bleek onder andere uit het volgende. Een aantal tellers die in ExpressV-HDL gespecificeerd waren, bleek niet noodzakelijk. De FPGA Compiler optimaliseerde het uiteindelijke resultaat en verwijderde een aantal niet gebruikte logische delen. Dit leverde een aanzienlijke besparing op qua aantal gebruikte logische modules.

Er is voor een Actel FPGA gekozen omdat deze door zijn anti-fuse techniek een voorspelbaar gedrag heeft en te verkrijgen is in vele soorten behuizingen en groottes. De Actel FPGA architectuur heeft nog een voordeel omdat er is geen extra EPROM nodig is om de configuratie op te slaan. Dit was van doorslaggevend belang omdat hiervoor geen ruimte was op het printed circuit board. Een 44-pins PLCC behuizing voldeed.

De eerste synthese slag leverde een te groot ontwerp op. Dit werd met name veroorzaakt door de vele tellers in het ontwerp. Na een extra optimalisatie slag is het uiteindelijk gelukt de A1020 FPGA voor 100% te vullen, alle 547 logische modules worden gebruikt! Dit grote getal baarde ons in eerste instantie nogal wat zorgen. Er werd namelijk gedacht dat een ontwerp met deze bezettingsgraad niet meer in de chip te plaatsen was. De pin-layout van de FPGA was namelijk al vastgesteld omdat andere ontwerpers het uiteindelijke bord al hadden ontworpen. De "place and route" leverde echter totaal geen problemen op: de Actel ALS Designer 3.0 software klaarde de klus in minder dan 5 minuten. Dit onderstreept nog maar eens weer de enorm flexibele "routability" van de Actel architectuur als gevolg van de kleine "grain"-structuur van de Actel anti-fuse technologie. Het ontwerp haalde een maximale kloksnelheidvan ruim 8 MHz hetgeen ruim voldoende was voor deze applicatie. Hierbij kan nog worden opgemerkt dat deze snelheid alleen voor het reset-signaal geldt en dat we verder geen optimalisatie hebben gedaan naar snelheid. Van de A1020 FPGA is de standaard speedgrade gebruikt.

3T heeft hiermee samen met TRANSFER EDS weer bewezen dat een hoog niveau FPGA ontwerp-traject in relatief korte tijd zeer goed is uit te voeren. En dat deze technologie niet alleen is voorbehouden aan grote multi-nationals met hoge R&D budgetten. Ook het midden en klein bedrijf kan uitermate succesvol zijn met top-down ontwerp methodiek. Tevens werd met dit project weer eens aangetoond dat alleen de beschikbaarheid van de juiste tooling niet automatisch tot de gewenste resultaten leidt. Intensieve begeleiding van de leverancier is van groot belang om de inleer-fase zo kort en efficiënt mogelijk te houden.

Erik Nijboer TRANSFER Electronic Design Support
Bert Molenkamp, Universiteit Twente
Auke Vleerstraat, 7521 PG Enschede
Tel. 053-4330336, fax 053-4340336
email info@transfer.nl, http:// www.xxlink.nl/transfer

VHDL, de standaard in hardware beschrijvingen

Bij veel grote en specialistische elektrotechnische bedrijven is een hardware beschrijvingstaal (Hardware Description Language, HDL) goed ingeburgerd. Veel van deze bedrijven gebruiken VHDL, of zullen dit op korte termijn gaan toepassen. VHDL staat voor Very High Speed Integrated Circuit Hardware Description Language. Een Amerikaans onderzoek laat zien dat het merendeel van de IC ontwerpers een HDL gebruikt en op korte termijn zal VHDL de meest toegepaste HDL zijn. Het lijkt erop dat VHDL de nieuwe standaard wordt voor het beschrijven van hardware. Dit artikel gaat nader in op de achtergronden van deze trend en de gevolgen voor kleine en middelgrote bedrijven.

2.2 Beheersen van complexiteit

Complexe elektronische systemen kunnen beter met behulp van een HDL ontwikkeld worden dan met schematische invoer. Bij een HDL wordt de architectuur en het gedrag van een schakeling beschreven op een computerprogrammaachtige manier. Dit in tegenstelling tot poortniveau bij schematische invoer. Grote delen van een elektronisch schema kunnen met HDL statements beschreven worden. De correcte werking van een HDL ontwerp kan met behulp van een simulator worden aangetoond en door middel van synthese kan deze beschrijving automatisch vertaald worden naar een elektronisch circuit op poortniveau. Met behulp van deze ontwerp-methodiek is de complexiteit van grote systemen beter beheersbaar en wordt de ontwerper produktiever en waarschijnlijk ook creatiever.

2.3 Inwerktijd

De elektronica branche is een zeer dynamische wereld. De ontwerpers worden overspoeld met nieuwe realisatietechnieken en ontwikkelgereedschappen met ieder hun eigen HDL invoertaal. Er zijn op het ogenblik enkele tientallen HDL's op de markt. Het overstappen van de ene HDL op de andere is een kostbare en tijdrovende zaak. De ontwerper zal zich ten eerste moeten inwerken in de nieuwe taal en vervolgens zullen de ontwerp-databases moeten worden vertaald. Er wordt veel waarde gehecht aan een standaard taal, die gebruikt kan worden voor verschillende ontwikkelgereedschappen.

2.4 Ontstaan VHDL standaard

Veel van de elektronische standaards zijn afkomstig uit Amerika. Het Amerikaanse Ministerie van Defensie (DOD) besteedt veel geld aan research, ontwikkelingen en standaardisatie. Veel werk wordt daarbij uitbesteed aan commercikle ontwikkelbedrijven. Het nadeel hierbij is dat elk bedrijf zijn eigen ontwikkelmethodiek en ontwikkeltools gebruikt, zodat deelontwerpen moeilijk kunnen worden hergebruikt en worden onderhouden. Dit alles maakte het noodzakelijk om een standaard taal te ontwikkelen. Het VHDL programma werd in 1981 voorgesteld. Met VHDL is het mogelijk om elektronische circuits onafhankelijk van de ontwerptools en technologie te kunnen beschrijven. Het grote voordeel hierbij is dat ontwerpen overdraagbaar zijn en dat deel ontwerpen kunnen worden hergebruikt zodat het relatief eenvoudig is om een meer geavanceerdere realisatie techniek te gaan gebruiken. In 1987 werd VHDL een internationale standaard (IEEE 1076). Het grote voordeel van deze standaard is dat verschillende groepen de taal-onderhouden en verder ontwikkelen met een goede inspraak procedure. De taal wordt onafhankelijk van commerciële producenten verder ontwikkeld. Zo kwam in 1992 een standaard uit die de logische niveaus van een digitaal signaal definieert (Standard Logic Value, IEEE 1164). Verdere ontwikkelingen worden en zijn gedaan op het gebied van analoge technieken, synthese, timing en back-annotation.

2.5 Commerciële VHDL produkten

Commerciële bedrijven kregen grote interesse in de nieuw ontwikkelde VHDL taal. Veel producenten van ontwikkelgereedschappen gingen eigen VHDL tools ontwikkelen en voegde VHDL interfaces toe aan bestaande tools. Momenteel heeft bijna elke tool een VHDL interface en zijn er al speciale VHDL ontwikkelomgevingen voor een personal computer te koop voor enkele duizenden guldens.
Door het grote aanbod van VHDL tools is het niet moeilijk meer om een compleet VHDL ontwerp traject op te zetten. Het gebruik van VHDL biedt flexibiliteit. Nieuwe tools kunnen eenvoudig worden toegevoegd. Een groot deel van het ontwerp-traject is onafhankelijk van de uiteindelijke realisatietechniek. Aan het eind van het ontwikkeltraject bepaalt men met behulp van een synthese tool de uiteindelijke realisatie vorm, zoals PLD's, FPGA'sof een eigen ASIC.

2.6 Overdraagbare specificaties

Grote complexe systemen kunnen door middel van VHDL op architectuur niveau beschreven worden. Vervolgens worden de functies van de subsystemen, de bijbehorende interfaces en de omgeving waarin het elektronische systeem moet werken beschreven. Dit heeft als voordeel dat het hele systeem en de interactie met de omgeving gesimuleerd kan worden. Verder kan de ontwikkeling van een deelsysteem eenvoudig uitbesteed worden.
Met name voor kleine bedrijven heeft dit enorme voordelen. Het bedrijf kan zelf de hele specificatie opstellen van een produkt. Vervolgens kunnen delen van het ontwerp worden uitbesteed aan een specialistisch bedrijf die het omzet in een optimale realisatievorm (bijvoorbeeld een FPGA of ASIC). Kleine bedrijven kunnen hiermee geavanceerde produkten ontwikkelen. Het opstellen van een goede gedetailleerde specificatie is vaak het moeilijkste en meest tijdrovende deel van het hele ontwerp-traject. Dit deel kan een klein bedrijf zelf uitvoeren zonder veel extra investeringen in apparatuur en tools. Alleen het bedrijf weet immers de specificatie van het te ontwikkelen produkt.
Aan de andere kant bestaat er een trend dat een opdrachtgever steeds meer eisen gaat stellen aan het ontwikkeltraject van een uitvoerder. Het Amerikaanse Ministerie van Defensie stelt als eis dat alle extern ontwikkelde elektronica beschreven moet zijn in VHDL. Steeds meer bedrijven nemen deze ontwerp-eis over.

2.7 Zijn er ook nadelen ?

Na alle positieve geluiden moeten we ook onderkennen dat er natuurlijk enkele negatieve kanten zitten aan het gebruik van VHDL. Zo is één van de geclaimde voordelen, dat VHDL modellen kunnen worden uitgewisseld. Maar het probleem is dat een VHDL model verschillende interfaces of inhoud kan hebben, zodat het niet mogelijk is om ze rechtstreeks uit te wisselen. Bij de producenten van ASIC's is dit een van de grootste problemen. Daarom werken verschillende groepen aan een standaard voor de methode van modelleren. Zo werkt bijvoorbeeld de VITAL organisatie (VHDL Initiative Toward ASIC Libraries) aan standaard ASCI bibliotheken.

Sommige ontwerpers vinden VHDL omslachtig in het gebruik en niet echt geschreven voor hardware ontwerpers. Daarbij geeft men vaak aan dat de VHDL simulators relatief langzaam zijn. Deze kritiek zal zeker waar zijn, maar dit weegt niet op tegen de voordelen die het gebruik van VHDL geeft.

2.8 Conclusie

Het belang van VHDL neemt sterk toe. Zoals het zich nu laat aanzien begint VHDL de standaard hardware beschrijvingstaal te worden. Steeds meer opdrachtgevers zullen eisen dat het ontwikkeltraject plaatsvindt in VHDL. Middelgrote en kleine bedrijven zullen hierdoor ook genoodzaakt zijn om VHDL te gaan gebruiken. Voor de ontwerper heeft VHDL veel voordelen. Elektronische systemen kunnen op een hoger niveau worden beschreven. Verder zijn er veel ontwikkelgereedschappen op de markt die een VHDL interface hebben.

Ir. G.J. Kleissen
Ing. S.W. Hulst
Centrum voor Micro-Elektronica
Tel: 0318-580200

EFFICIENCY WITH VHDL

How to improve time to market
VHDL, een andere manier van het omschrijven van programmeerbare logica. U zult wel denken, daar hebben we al zo veel van! Toch is het belangrijk dat u even tijd maakt om dit korte stukje tekst door te lezen. Het gaat namelijk om efficiëntie. Is dat niet iets waar iedereen in geïnteresseerd is?

3.2 Waarom VHDL ?

VHDL is benoemd als de (toekomstige) standaard voor het beschrijven van programmeerbare logica. Het ligt dus in de lijn der verwachting dat de komende tijd iedere designer die op dit gebied actief is in aanraking zal komen met VHDL. Het resultaat hiervan zal zijn dat de communicatie in een projectteam verbetert. Als men dezelfde taal spreekt, begrijpt men elkaar beter. Dit geldt dus ook voor de communicatie tussen designers onderling, ontwikkelaar en gebruiker, contractors en verschillende design tools.
Het maakt overigens niet uit of het design bestemd is voor een Programmable Logic Device (PLD), een Complex Programmable Logic Device (CPLD) of een Field Programmable Gate Array (FPGA). VHDL is hardware onafhankelijk. Het is dus zeer flexibel. Op het laatste moment kunt u nog van device veranderen.

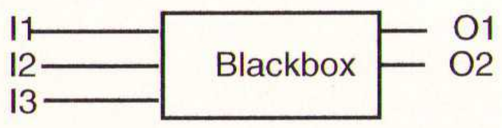
3.3 VHDL.....Wat is dat eigenlijk ?

VHDL staat voor "Very high speed integrated circuit Hardware Development Language". VHDL is ontstaan in de jaren '70 uit een ontwikkeling van de Amerikaanse defensie. Hier werd een taal ontwikkeld om complexe circuits te kunnen documenteren, zodat een design van een contractor door een ander begrepen kon worden. Vandaag de dag gaat het gebruik van VHDL wat verder dan alleen documenteren. Het kan gebruikt worden voor het beschrijven, simuleren en de synthese van logic designs. Het beschrijven van de logica gebeurt in de vorm van een hogere programmeertaal (voorbeelden volgen). Hierdoor is het leesbaar voor mens en machine. Uiteraard gaat men er dan wel van uit dat deze mensen enige kennis hebben van programmeertalen. Bij het simuleren kunnen we zowel functionele- als timingsimulatie toepassen. De timing-simulatie is vooral bij FPGA's van groot belang. Hier heeft de 'routing' van het design in het device namelijk heel veel invloed op.
Onder synthese verstaan we het omzetten van de beschrijving van een design naar de beschikbare hardware logica. Dit is dus eigenlijk de conversieslag van software naar hardware.

3.4 Hoe werkt VHDL?

VHDL gaat altijd uit van twee basisblokken, nl. de ENTITY en de ARCHITECTURE. De entity kunnen we zien als een black-box waar we signalen in zien gaan en uit zien komen. We weten hier niet wat er met deze signalen gebeurt. Kort gezegd praten we hier dus over de I/O definitie van een black-box. In de architecture omschrijven we nu juist het gedrag van de black-box. Hier kunnen we met verschillende opdrachten (statements) definiëren wat er gebeurt met de reeds aangegeven I/O.

De Entity



Hierboven ziet u een voorbeeld van de blackbox. De entity omschrijving die hierbij hoort is als volgt:

```
entity TEST is
  port (I1, I2, I3   : in bit;
        O1, O2       : out bit);
end TEST;
```

U ziet inderdaad een omschrijving van de I/O. Wat betekenen nu de gegevens die tussen haakjes staan? We beginnen met de namen van parameters, hier I1, I2 en I3. Dan volgen 'mode' en het 'type' van het signaal. Voor de 'mode' hebben we de volgende keuzemogelijkheden:

* in: de parameter is alleen een ingang
* out : de parameter is alleen een uitgang
* in/out : de parameter is zowel in als uitgang (bidirectioneel)
* buffer : de parameter is een uitgang, maar kan ook nog intern gebruikt worden

Hieronder zien we een aantal mogelijke type aanduidingen

* bit: de parameter is een bit
* bit_vector(3 downto 0) : de parameter is een 4 bits woord
* x01Z: de parameter kan de waarden don't care, 0, 1 en tri-state hebben
* integer: de parameter is een integer
* boolean: de parameter is true of false
* enumerated: de parameter kan waarden aannemen die door de gebruiker zelf opgegeven zijn

Een voorbeeld van het laatste type zou zijn: type STATUS is (start, slow, fast, stop).
Dit is de basis van de entity. We kunnen uiteraard zoveel I/O regels definiëren als nodig zijn voor het design. De type declaratie die hier gegeven wordt is erg bindend. Dit kan soms lastig zijn omdat je verplicht bent om nauwkeurig en eenduidig te werken, maar is dit eigenlijk niet juist het voordeel?

3.4.1 De Architecture

De Architecture is de omschrijving van het gedrag van de entity. We hebben hier verschillende mogelijkheden om het gedrag te omschrijven, we kunnen gebruik maken van:
Behavioral/Structural descriptions

Bij behavioral descriptions kunnen we gebruik maken van abstracte omschrijvingen of van boolean equations. Bij een abstracte omschrijving kunt u denken aan:
if a = b then state <= state5

Bij boolean equations denken we aan:
x <= (a OR b) AND c

Vooral de abstracte manier van omschrijven is heel begrijpelijk. Deze verdient dan ook veelal de voorkeur. Vooral bij grotere designs heeft dit dan het voordeel dat iemand anders snel inzicht heeft in de werking. Bij structural descriptions wordt gebruik gemaakt van standaard componenten. Deze componenten zijn ook weer opgebouwd uit een entity en architecture structuur. Tevens is het mogelijk om meerdere componenten te combineren. Dit doen we met behulp van een package. In de topfile hoeft alleen kenbaar gemaakt te worden dat dit package gebruikt moet worden. Eigenlijk is het alleen aantrekkelijk deze manier te gebruiken bij grotere designs om een opsplitsing te maken. Hierdoor is men in staat het design hiërarchisch op te zetten. Hieronder een voorbeeld van een twee bits comparator:

```
xor2 portmap (a,b,i);
inv portmap (i,c);
```

De standaard componenten zijn hier 'xor2' en 'inv'. Bij deze omschrijving wordt ervan uitgegaan dat de componenten reeds gedefinieerd zijn. Indien een comparator voor een byte gemaakt moet worden, wordt de omschrijving een stuk uitgebreider. Per bit moet een vergelijk gemaakt worden waarna het resultaat getotaliseerd wordt. In de praktijk zal veelal de combinatie behavioral met structural descriptions voorkomen.

3.4.2 Sequential en Concurrent statements

Binnen een architecture kunnen de opdrachten achtereenvolgens uitgevoerd worden (sequential) of gelijktijdig (concurrent). De voorbeelden die tot nu toe gegeven zijn, zijn allen concurrent. Indien sequential statements gewenst zijn moeten we gebruik gaan maken van de 'PROCESS' structuur.
Deze structuur ziet er als volgt uit:

```
PROCESS(clk, rst)
BEGIN
.....
.....
END PROCESS;
```

Opdrachten in PROCESS worden sequentieel uitgevoerd. Pas bij het END PROCESS statement worden de nieuwe waarden toegekend aan de variabelen. De laatst toegekende waarde telt. De parameters tussen haakjes is de zogenaamde 'sensitivity list'. PROCESS wordt pas actief indien een van deze parameters van waarde verandert. Het is mogelijk om verschillende processen in een architecture op te nemen. Deze worden dan weer concurrent t.o.v. elkaar uitgevoerd. De eerste aanzet tot VHDL is hiermee gegeven. Hieronder een voorbeeld van een design voor een 4- bits counter in VHDL.

```
entity TEL is port(
        count    : buffer bit_vector(3 downto 0);
        clk, rst : in bit);
end TEL;

architecture ARCHTEL of TEL is
process(clk)
begin
        if (clk'event and clk='1') then
          if rst = '1' then count <='0000';
          else count <=  count + '1';
        end if;
end process;
end ARCHTEL;
```

Uiteraard is het niet mogelijk alle details van VHDL in zo'n kort artikel te noemen. Het ging hier om een introductie van de mogelijkheden. Ter afsluiting nog een aantal kernpunten met de voordelen voor de gebruiker:

Eigenschappen VHDL

Voordeel de gebruiker:

* Gestandaardiseerde taal voor hardware
  * Algemene bekendheid, makkelijker te beschrijvingen leren
* Geschikt voor beschrijving, simulatie
* Een tool voor alles, gemak en synthese van logic designs
* Hardware onafhankelijk
  * Flexibiliteit indevice keuze, vermindert afhankelijkheid van specifieke produkten
* Hogere programmeertaal, goede
  * Tijdsbesparing door verbeterde leesbaarheid communicatie in een projectteam
* VHDL is de toekomst
  * Efficiëntie, improved time to market

Michel Smit
Field Application Engineer bij SEI
Sonetech.
Tel: 040-2635635

Hardware ontwerpen met software methodieken

Inleiding

Door de sterk toenemende technische mogelijkheden worden elektronische schakelingen steeds, en steeds sneller, complexer. De geschiedenis van het ontwerpen van deze schakelingen is echter nog maar kort. Slechts twintig jaar geleden werden ASIC's, die nu als zeer eenvoudig gekarakteriseerd zouden worden, ontworpen op lay-out niveau door rechtstreeks de metaal- en andere lagen die aangebracht moesten worden, te tekenen. Dit werk was zeer tijdrovend en foutgevoelig vanwege de toenmalige beperkte technische middelen.

Om de groeiende complexiteit te kunnen bijhouden qua ontwerp productiviteit kwam daarna het schema-tekenen in zwang, wat nog steeds veelvuldig wordt toegepast. Deze vooruitgang hield in dat men op een hoger abstractie niveau ontwierp: in plaats van de werkelijke patronen op silicium, werden nu poorten (gates) getekend die vertaald moesten worden naar de patronen op silicium. Het lay-out abstractie niveau had plaats gemaakt voor het gate-level abstractie niveau.

Thans zijn Hardware Description Languages (HDL's) in vrij korte tijd populair geworden voor het ontwerpen van ASIC's en FPGA's. Daarmee wordt het abstractie niveau van ontwerpen opnieuw opgetild naar Register Transfer Level (RTL) of zelfs algoritmisch niveau, afhankelijk van het niveau waarop de HDL code geschreven wordt. Een zeer ingrijpende verandering is daardoor te zien in de ontwerpmethodieken: software ontwerpmethodieken doen hun intrede bij het ontwerpen van hardware.

In dit artikel worden de parallellen belichten tussen hardware ontwerp met HDL's en software ontwerp en komt tevens een typische HDL gebaseerde hardware ontwerp-flow belichten.

4.2 Overeenkomsten en verschillen

Bij het beschrijven van overeenkomsten en verschillen worden de volgende items belichten:

Overeenkomsten:
Abstractie niveau
- Taal gebaseerd
- Hergebruik van code
- Code profiling
- Prototyping
- CASE tools

Verschillen:
- Simulatie
- Timing

4.2.1 Abstractie niveau

De overgang van schema-tekenen naar het gebruik van HDL's is qua methodiek een overgang in abstractie niveau en niet zozeer één van tekenen naar taal. Immers, bij gebruik van schema-tekenen wordt een netlist gemaakt, die in een taal uitgedrukt wordt. Meestal is dat EDIF (Electronic Design Interchange Format). Aangezien deze netlist echter een beschrijving is van componenten en hun onderlinge verbindingen, leunt deze ontwerp-methode zwaar op grafische invoer. De taal (de EDIF beschrijving) wordt alleen gebruikt als tussenformaat naar andere producten.

Bij het gebruik van HDL's wordt het ontwerp echt *beschreven* in taal, waarbij de taal niet dient als uitwisselingsformaat, maar als uitdrukkingsmiddel zelf. Daarmee verschuift de aandacht naar de *specificatie* van het ontwerp in een HDL. Het doel is nu te beschrijven *wat* een ontwerp functioneel (het gedrag) *moet doen*, in plaats van te beschrijven *hoe* een ontwerp *is opgebouwd* uit componenten.

Bij de overgang van schema-teken technieken naar een HDL based ontwerpmethodologie is dit aspect voor de ontwerper het belangrijkste. Het ontwerp wordt gemanipuleerd door middel van de taalspecificatie, niet meer door het tekenprogramma.

4.2.2 Taal

Behalve het specificatie aspect is de overeenkomst natuurlijk overduidelijk het werken met tekst. Een goede structurering en opzet van het programma met het toepassen van een goede coderingsstijl biedt de hardware ontwerper leesbaarheid en overzichtelijkheid, daarentegen is er ook de mogelijkheid slecht leesbare code te schrijven, zoals dat met software ook kan.

Standaardisatie en daarmee overdraagbaarheid speelt ook een rol. Een ontwerp gespecificeerd in een HDL conformeert zich aan die programmeertaal, niet aan conventies van één bepaald product van één leverancier, zoals dat met schema-tekenen het geval is.
Tenslotte kan taal met een tekst editor worden ingevoerd en is daarom niet veeleisend in het gebruik. Schema-teken pakketten hebben een licentie in gebruik gedurende de tijd dat het schema getekend wordt, bij een op taal gebaseerde aanpak is dat niet het geval. Bijkomend voordeel is platform onafhankelijkheid, tekst editors draaien zowel op PC als op werkstation.

4.2.3 Hergebruik van code

Code hergebruik is een zeer sterke overeenkomst met software ontwikkeling. Niemand zal voor MS-Windows programma's zelf nog de grafische functies programmeren. Deze komen namelijk uit standaard beschikbare bibliotheken. In HDL's is ook de mogelijkheid opgenomen om libraries te bouwen. Run time programmeerbare parameters bieden flexibiliteit voor hergebruik. Dit is een belangrijk voordeel ten opzichte van schema-tekenen waar de vereiste parametrisering niet kan worden uitgevoerd. Ook standaard routines als lezen/schrijven van/naar files kunnen worden opgenomen in bibliotheken.

Bovengenoemde, in onze ogen triviale, mogelijken zijn echter voor hardware ontwerpers pas toegankelijk geworden met de komst van HDL's. Daarvoor was er uiteraard wel hergebruik van ontwerpen, maar gebeurde dat meestal door stukken van een ontwerp te kopiëren en aan te passen aan de nieuwe eisen. Door de HDL's zien we standaard bibliotheken ontstaan die een brede functionaliteit bieden.

4.2.4 Code profiling

"Profiling" is het bepalen welke stukken code hoe vaak worden uitgevoerd. Voor software ontwikkeling is dit al jaren een belangrijk middel voor snelheid optimalisatie. Profiling geeft inzicht in de vraag of alle delen van het ontwerp wel gebruikt worden. Zo niet, dan zijn er twee mogelijkheden: delen worden niet getest of zijn niet nodig, waardoor later redundante hardware gebouwd wordt. In het traditionele schema-tekenen heeft de ontwerper geen inzicht of al zijn componenten wel nodig zijn.

4.2.5 Prototyping

Doordat HDL's simuleerbaar zijn, kan een ontwerp al worden getest op functie voordat er ook maar iets geïmplementeerd wordt. Er hoeft zelfs helemaal geen keuze gemaakt te worden voor een realisatie. Een HDL ontwerp kan later zowel in standaard componenten, als in FPGA's of ASIC's worden geïmplementeerd. In de software-wereld is dit te vergelijken met GUI (General User Interface) builders. Ook hier kan een gebruikers interface worden opgebouwd en getest zonder dat daar een echte applicatie voor nodig is. Daardoor ontstaan er twee voordelen: met behulp van simulatie kan getest worden ofideeën/functies wel werken én de toekomstige gebruiker krijgt zicht op wat de ontwerper nu eigenlijk maakt.

4.2.6 CASE tools

Voor software ontwerp zijn CASE (Computer Aided Software Engineering) tools al jaren bekend. Het hardware ontwerp equivalent hiervan wordt nu ook steeds meer gebruikt door ontwerpers. We komen hierop bij het beschrijven van de ontwerpflow nog terug.

4.3 Verschillen

Er zijn ook een aantal verschillen te noemen. We onderscheiden de volgende twee:

4.3.1 Simulatie

Simulatie is het aanbieden van testvectoren aan het ontwerp en het observeren van de resultaten. In de software ontwikkeling wordt de functionaliteit meestal getest door het ontworpen programma te gebruiken. Deze werkwijze is mogelijk doordat de ontwikkelstappen snel zijn uit te voeren (vooral als deze op één en hetzelfde computer platform worden uitgevoerd) in de volgorde: programma code; compileren; uitvoeren.

Meestal zijn deze fasen in minuten uit te voeren. Voor hardware ontwikkeling kan compilatie uren duren. Het uitvoeren is afhankelijk van het fabriceren van de hardware wat bij een ASIC maanden kan vergen en zeer kostbaar kan zijn. Simulatie bevordert dat deze tijd-consumerende stappen zo min mogelijk worden uitgevoerd. Voor een ASIC is het doel "first time right"!

Een ander belangrijk verschil is dat HDL's zijn ontworpen voor simulatiedoeleinden. Dit heeft nieuwe mogelijkheden met zich meegebracht, die voorheen in hardware simulatie op poortniveau niet mogelijk waren. Ten eerste kunnen test vectoren ook in de HDL's worden beschreven waardoor de testvectoren net als het ontwerp zelf portable worden. Een tweede mogelijkheid is dat HDL simulatoren interactief kunnen reageren op het ontwerp.

4.3.2 Timing

Hardware ontwerpen zijn van nature timing-specifiek. Dit brengt met zich mee dat timing onderdeel is van de specificatie. Tijd-informatie is dan ook onderdeel van HDL's . Dit biedt de mogelijkheid om deze timing in simulaties op te nemen. Simulatie is dan niet meer louter functioneel, zoals bij het testen in software, maar gaat een stap verder.

4.4 Ontwerpflow

4.4.1 Specificatie

Bij het gebruik van een HDL komt specificatie van een elektronisch circuit neer op het schrijven of genereren (of een combinatie hiervan) van een HDL beschrijving. Net zoals in software komt zowel het rechtstreeks schrijven als genereren van de HDL specificatie voor, afhankelijk van de complexiteit van het geheel. In de software-wereld hebben CASE tools opgang gemaakt om de groeiende complexiteit van de code te beheersen. Het equivalent in de hardware wereld is eveneens aanwezig en wordt Electronic System Design Automation (ESDA) genoemd. Een andere benaming hiervoor is ook wel: front-end HDL tools.

Een ESDA tool specificeert een ontwerp. Er wordt HDL code gegenereerd uit een grafische beschrijving. De gegenereerde code is dan syntactisch-correct. De meeste ESDA tools laten toe dat rechtstreeks HDL code ingevoerd wordt op plaatsen waar bepaalde constructies het beste in taal kunnen worden weergegeven, terwijl graphics wordt toegepast waar die vorm duidelijk voordelen biedt. Het grafischedeel van het ESDA tool bevat de volgende functionaliteit: hiërarchische decompositie, formele specificatie, type en andere checking, design management en parameters.

4.4.2 Syntax-directed editors

Evenals in de software-wereld kent een HDL ontwerpomgeving syntax-directed editors. Deze tekst editors vullen automatisch veelgebruikte syntax patronen aan, geven direkte toegang tot de syntax constructies en genereren skeletons waarin de gebruiker zijn/haar HDL beschrijvingen verder kan uitwerken. De HDL code wordt colour-coded weergegeven met regelnummers ervoor wat de leesbaarheid vergemakkelijkt, evenals het opzoeken van regels code waar een bug in wordt gemeld.

4.4.3 Libraries

Er bestaan tal van bibliotheken die qua gebruik te verdelen zijn in een aantal categorieën:

Libraries met beschrijvingen van bijvoorbeeld timing modellen, file i/o routines en geheugenmodellen. Hier kan een parallel getrokken worden met software libraries die meegelinkt kunnen worden. De gebruiker hoeft hierdoor niet alles zelf te definiëren, maar kan een groot aantal basisroutines uit een library gebruiken.

Simulatielibraries. Deze libraries bestaan uit een groot aantal standaardcomponenten waarvan de beschrijving meegesimuleerd kan worden met de beschrijving die men zelf gemaakt heeft.

4.4.4 Simulatie / Source level debug

Een belangrijk onderdeel van het schrijven van software is het debuggen van de code. Daartoe worden voor alle populaire programmeertalen source code debug omgevingen aangeboden. Het gebruikersgemak en de features van deze omgevingen bepalen in grote mate de snelheid van debuggen van de code. Dit alles geldt onverkort ook voor de debug omgevingen van HDL's.

Een eerste vereiste is snelheid van werken. De "loop" die gevormd wordt door simuleren, een fout opsporen in de waveforms, deze fout terugvinden in de broncode door debugging, de broncode aanpassen en opnieuw compileren en simuleren, moet snel te doorlopen zijn.

Hoewel veel simulatieproducten deze funktionaliteit nog gefragmenteerd aanbieden in afzonderlijke producten zoals een simulator, een waveform viewer en een source debugger, zijn er nu ook producten die al deze mogelijkheden in één geïntegreerde omgeving aanbieden.

Verder moeten HDL simulatoren in staat zijn om de volledige taalbeschrijving te kunnen simuleren, wil de gebruiker een efficiënt gebruik van alle taalconstructies kunnen maken.

Pure simulatiesnelheid is ook belangrijk. Omdat de taaldefinitie zich uitstrekt over verschillende abstractie niveaus, dient de simulator zijn snelheid ook op al deze niveaus te behouden, wil het product efficiënt inzetbaar zijn in verschillende typen toepassingen. Over het algemeen geldt dat naarmate het abstractie niveau van de HDL beschrijving afneemt, de simulatiesnelheid eveneens afneemt.

4.4.5 Profilers

De software-wereld kent een schat aan utilities die de programmeur helpen om styling fouten te vinden, code te optimaliseren op runtime snelheid etc. etc. Ook de HDL wereld kent tegenwoordig deze profiler producten die de code analyseren en optimaliseren. Tevens komen er formele verificatie producten op de markt om automatisch, via wiskundige formules, te bewijzen dat HDL beschrijvingen op verschillende abstractie niveaus functioneel gelijk zijn.

4.4.6 Compilers

Veel aandacht wordt in hardware automatisering besteed aan het equivalent van software compilers, synthese producten genoemd.Deze zetten de HDL code om naar de netlist, de assembleer taal van de hardware. Immers, de efficiëntie van deze compilers bepaalt de grootte van de uiteindelijke realisatie en daarmee een belangrijk deel van de kostprijs van de oplossing. Daar komt bij dat de toepassingen divers zijn, met name door de snelle opkomst van FPGA's. In tegenstelling tot ASIC architecturen, die onderling redelijk op elkaar lijken, hebben FPGA's en CPLD's onderling compleet verschillende architecturen. Deze vragen dan ook uiteenlopende algoritmen om een efficiënte mapping naar een netlist te maken. De nieuwste generatie synthese tools is in staat om een zodanig goede mapping uit te voeren dat handkwaliteit wordt benaderd.

De assembler buit de mogelijkheden van de onderliggende hardware uit. Zo biedt een goed syntheseproduct ook de mogelijkheid de architecture van het hardware device, waarop mapping plaatsvindt, uit te buiten om een zo hoog mogelijke dichtheid te bereiken. Eveneens van cruciaal belang is de mogelijkheid om niet alleen de controlelogica goed af te handelen, maar ook het datapad via het gebruik van modulegeneratoren die een rechtstreekse afbeelding maken van het datapad op het device.

4.4.7 Test

Veel software programma's kennen hun eigen testmodules die over het algemeen in dezelfde programmeertaal geschreven zijn als de software zelf. VHDL en Verilog kennen deze mogelijkheid ook. Deze zogenaamde testbenches zijn eveneens in die HDL talen geschreven en kunnen in de simulator worden gebruikt voor de definitie van stimuli en verwachte resultaten. Alle constructies van de taal staan hier ter beschikking van de ontwerper om een goede testset te bouwen. Immers, voor complexe ontwerpen beschreven in een HDL geldt hetzelfde als voor een complex software ontwerp; er treden onvermijdelijk bugs op, waardoor uitputtend testen een vereiste is.

Het schrijven van voldoende testbenches wordt bij hardware ontwerp vaak schromelijk verwaarloosd. Het blijkt echter dat voor complexe ontwerpen meer dan 50% van de ontwerptijd gaat zitten in de testbenches. Daarom loont het om ook op dit vlak automatisering toe te passen. De nieuwste trends wijzen in die richting: er komen tools op de markt waarmee testbench ontwikkeling sneller kan plaatsvinden in een soort spreadsheet-achtige omgeving.

4.5 Conclusie

Het gebruik van HDL technieken biedt grote voordelen. Deze komen voort uit het ontwerpen op een hoger abstractie niveau en de inherente voordelen die gebruik van een programmeertaal bieden. De bijbehorende ontwerpflow wordt tegenwoordig ondersteund door een heel scala aan hulpmiddelen om snel te kunnen specificeren, snel en goed te kunnen debuggen en efficiënte en goed geteste code te schrijven. De compilers zijn van een zodanige kwaliteit dat handkwaliteit dicht benaderd wordt.

Willem J.A. Gruter,
Chris A. van Veenendaal
Translogic BV
Tel: 0318-642076

Cypress introduceert het Ultralogic concept

Open Design Tools
ULTRA LOGIC
High-Performance Programmable Logic

Programmeerbare logica van PAL tot FPGA in één universele VHDL ontwikkelomgeving.

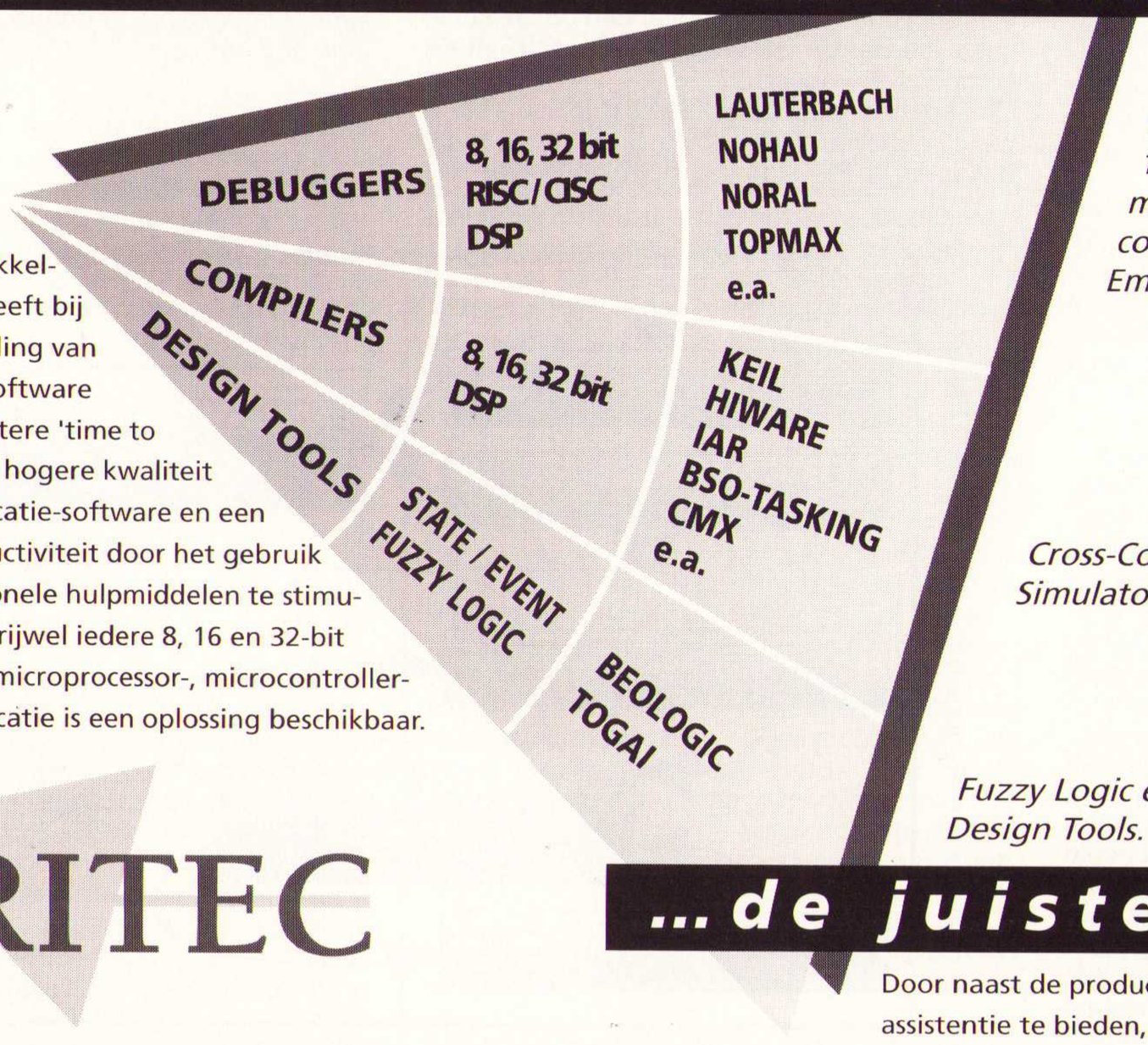
- pASIC: De allersnelste antifuse FPGA's
- FLASH 370: Flash gebaseerde high speed complexe PLD's
- MAX 5000: Complexe reprogrammeerbare PLD's
- High performance PAL's & PLD's
- WARP: Één universeel VHDL gebaseerd ontwikkeltool

Dankzij de Programmeer en Design Service kan Sonetech zeggen ... "Support Included".

SONETECH

Nederland: tel. +31 40 263 56 35, fax +31 40 283 23 00
E-mail:sonetech@euronet.nl
België: tel. +32 2 460 07 07, fax +32 2 460 12 00
E-mail:sonetech@eunet.be

embedded software ontwikkeling...



De productgroep ontwikkelsystemen streeft bij de ontwikkeling van embedded software naar een kortere 'time to market', een hogere kwaliteit van de applicatie-software en een betere productiviteit door het gebruik van professionele hulpmiddelen te stimuleren. Voor vrijwel iedere 8, 16 en 32-bit (RISC / CISC) microprocessor-, microcontroller- en DSP-applicatie is een oplossing beschikbaar.

In-Circuit Emulators voor 8-, 16- en 32-bit (RISC / CISC) microprocessors, microcontrollers en DSP's, EPROM Emulators en BDM debuggers.

Cross-Compilers / Assemblers, Simulators, Real-Time Kernels.

Fuzzy Logic en State / Event Design Tools.

TRITEC

...de juiste partner!

Door naast de producten trainingsprogramma's en assistentie te bieden, verzekert Tritec u van een succesvol verloop in de ontwikkeling van embedded software.

TRITEC Benelux B.V. • Antoniuslaan 1 • 3341 GA • H.I.Ambacht • tel.: 078 - 6816133 • e-mail: development-tools@tritec.nl

VHDL, Hard- & Software definitietalen

Hardware - Software Co-Design

Introduction

The ever increasing cost and complexity of electronic systems, combined with time to market pressures, are forcing design teams to aim for system verification (incorporating both the hardware and software system components) significantly earlier in the design cycle than has traditionally been targeted.

Using conventional simulation techniques, it has proved extremely difficult to execute software programs on a simulated hardware description without increasing program run-times unacceptably. Maintaining compatibility with existing software design and debug tools poses additional problems.

Typically, the hardware and software components of a system design are brought together for the first time when the prototype hardware system has been fabricated. As long as the inevitable bugs can be fixed in software, the system integration is relatively well controlled. However, if a hardware bug is discovered, a board or ASIC iteration can be required to correct the problems. This option can have such serious implications for project costs and schedules that, instead of having to iterate the design, compromises in performance or functionality are often accepted.

The pressing need for solutions to allow hardware and software to be verified before the hardware is built - Virtual Systems Integration - is being exacerbated by several significant industry trends. As systems become more complex, software has become the dominant component in the system (there is a ratio of 4 software engineers for each hardware engineer, and this ratio is increasing). The interaction between the software and the hardware is critical to the correct system function.

Hardware designers are turning to more powerful processor families to handle the increased software requirements. 32-bit processors are increasingly popular for a wide range of applications. However, these processors require more complex tools for the associated software design. Most code development is done using a high level language. It is not practical to write and debug system software at assembly level. As 70% of ASICs design today are designed to work with processors, ASIC manufacturers now provide complex microprocessors as cores on ASICs.. It is possible to build a processor, RAM, ROM and custom hardware onto one ASIC. This poses an interesting question :- Is a software bug on an ASIC ROM a software or a hardware bug?

The ability to verify and debug the software and hardware together before the hardware is built would not only be a significant step forward in detecting and correcting design errors, but would help in realising the first levels of system integration. Figure 1 compares conventional system design flow with Virtual Systems Integration. Almost all the time savings achievable are due to the fact the debug, verification and system integration can begin before the hardware prototype is built. This article reviews existing system simulation strategies and their limitations.

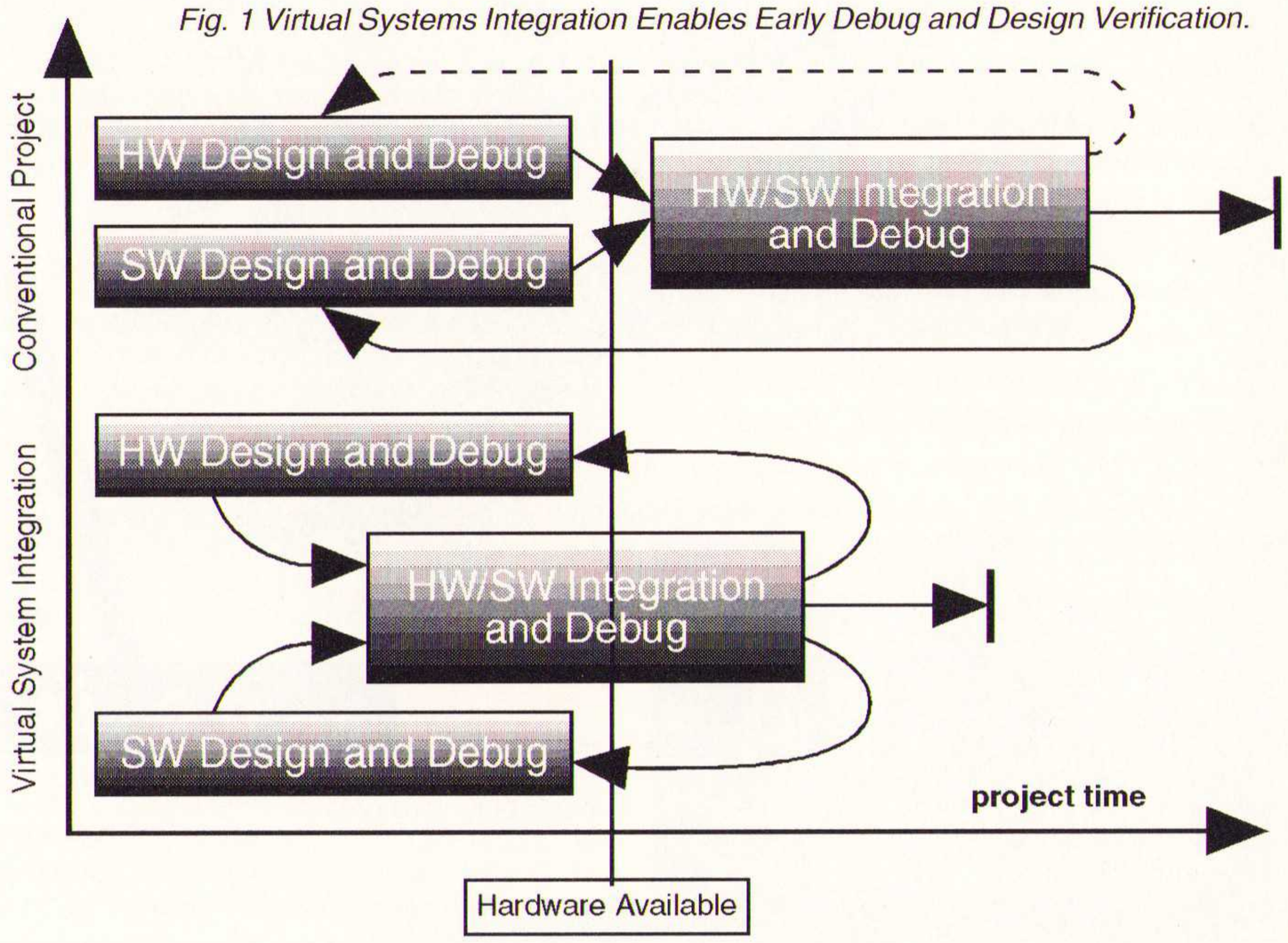
Automation have developed a unique technology that brings together existing hardware and software design toolsets into a unified debug and verification environment capable of executing software programs a thousand times faster than conventional simulations.

Often, the first attempt at system simulation is to cross-compile the software program down to object code. The object code can be loaded into a simulator memory model, together with a full functional model of the system processor and the associated peripherals and ASICs. The object code within the memory can be executed on the processor model.

However, there are some significant drawbacks to this approach. Program execution speed is unlikely to be fast :- a typical rate of execution is 2 to 3 microprocessor instructions per second. Model availability can also be a problem - most processor manufacturers are not keen to release full-functional models of advanced processor in case the design is reverse engineered into a competitive product.
One solution to these problems is to use a hardware modeller. This allows a real chip to be included in the simulation which solves the modelling problem. The speed of simulation also increases but typically only by a factor of two. However, if the chips are not static, or if multiple instances of the processor are used, the hardware modeller is required to store and rerun on all the previous simulation vectors for each simulation cycle. As the simulation progresses, the instruction rate decreases; and the vector memory fills up.

Consequently, most designers set their sights a little lower and use bus functional models of processors. These models are an excellent way of verifying the interaction of each type of bus cycle in the system. Scripts can be used to put together sequences of bus cycles to emulate specific groups of instructions, but true program execution is not possible. All these techniques suffers from one more serious drawback. Even if these methods were powerful enough to run significant amounts of code allowing bugs to be detected, because all the code is at object level it would be extremely difficult and time consuming for a software engineer to diagnose the cause of a problem. There is no access to any of the standard diagnosis tools such as debuggers and code profilers at the object code level. For complex microprocessors, this makes it almost impossible to debug and diagnose the detected problems.

Fig. 1 Virtual Systems Integration Enables Early Debug and Design Verification.



5.3 System Design Methodology Trends

There are a number of trends in both software and hardware design that can be extended to construct a Virtual System Integration environment. System software that utilises hardware functionality tends to be well structured. A general rule is that all the low-level functions that interact directly with the hardware are confined to 'driver' functions. This means that when the hardware changes or a bug is found in the interaction between the software and hardware, the code that has to be rewritten is localised in one place only.

Of course, much of the software is written while the hardware is being designed. Until the hardware is complete there is no target system where the software can be executed. The way around this problem is that, writing in a high level language, the code can be cross-compiled onto another system such as a workstation. This works well until the hardware drivers are executed - there is no custom hardware with which the software prototype can interact. A common technique is to substitute the hardware drivers with 'fake' driver functions. The 'fake' drivers attempt to provide some sort of response back to the main body of the software program that emulates the actions of the projected hardware.

In some cases these fake drivers are minimal functions that provide the most limited responses - but enough to allow testing of the main body of the software. In other cases the 'fake' drivers are extremely complicated functions that emulated many aspects of the projected hardware. In fact, in such cases, there is often a significant amount of work done to verify that the drivers and the hardware description match. This can be an extremely difficult task because the driver functionality is mainly in a 'sequence' domain while the hardware description executes in a time domain. These techniques allow much of the software to be tested independently of the hardware. What is not verifed is the interaction between software and hardware.

For hardware design, the overwhelming trend over the last ten years has been a shift away from gate-level schematic capture towards language based design entry. This has been underlined by the acceptance of standard Hardware Design Languages (HDLs) such as VHDL and Verilog. Within language based design, designers are beginning to realise the benefits of behavioural modelling before the more familiar Register Transfer Level (RTL) descriptions are created. One of the main reasons for higher-level design abstraction is that more testing and verification can be done earlier in the design cycle before the gate-level design is started. This allows design errors introduced at an early stage in the design to be detected and fixed as early as possible.

Putting these two trends together shows that early in the design project there exists prototype software that can be run independently of the ultimate target hardware; and a functional HDL description of the projected hardware.

Eagle Design Automation has put these two components of the system together using unique technology to create a virtual environment in which the HDL description and the prototype software can be executed together.

By : Andre Brandwijk
VIEWlogic
Bruistensingel 126
5232 AC Den Bosch
tel: 073 - 6408468
fax: 073 - 6408469
e-mail: abrandwijk@viewlogic.com

System specification, analysis and implementation

Language Independence, the Future of the EDA Industry

For several years now, the EDA industry has been locked in a battle over which hardware description language (HDL) to support as a standard. After much debate the marketplace has voted and a clear winner selected - both.

This state of affairs may catch even some EDA veterans off guard, but the decision is a logical one. Both Verilog and VHDL, the primary contenders, have their numerous proponents across the EDA industry and among both designers and vendors of ASICs and ICs. Mentor Graphics has long supported standards such as VHDL. This commitment to VHDL is due to the natural strengths of VHDL, and because its IEEE-1076 specification prevents the difficulty of supporting an ambiguous and changing specification.

With the adoption of the IEEE-1364 standard for Verilog, Mentor Graphics has begun the process of supporting functionality that will allow it to provide Verilog point tools and design flows. In addition to supporting VHDL and Verilog separately, Mentor Graphics has developed tools that can support both VHDL & Verilog simultaneously.
This need for mixed language designs as been identified by several key customers and is a requirement for the emerging Systems on Silicon Initiative(tm). Support for both languages, or language independence, means that electrical engineers can bring together any combination of Verilog and VHDL into a single design. In essence, language independence means freedom. Designers can now design, simulate, and synthesize a circuit using either Verilog or VHDL or mixed VHDL and Verilog. By allowing this mixed HDL support, Mentor Graphics adds language independence to the implementation independence already enjoyed by HDL users.

6.2 Removing Barriers

The foremost benefit of language independence is that it opens doors to design reuse by safeguarding the investment a company makes in intellectual property. Such property can take a number of forms. It can be the processor core at the heart of a new IC for the hand held market. Or it can be the stimulus for verification of a VHDL circuit. It can even be the methodology for creating Verilog or VHDL cell libraries - a methodology that's been created by a design team over a period of many months.

Chances are excellent that at some time a company will need to use that intellectual property on a new design. The growing complexity of electronic circuits resembles, in some ways, a snowball rolling downhill. To pack ever-more functionality onto a chip, within a shrinking time-to-market, design teams are finding new ways to re-use previously developed intellectual property. ICs and ASICs that used to be standalone are now being combined into larger systems on silicon (SOS). The design becomes larger and time to market shrinks (the snowball get larger and speeds up) until it reaches a barrier. For most customers, that barrier is encountered early in the process and that barrier is model availability (the VHDL customer who needs a processor model only available in Verilog, for example).

Language independence simply removes the traditional barriers to design reuse and model availability. A piece of code written in Verilog can be applied to VHDL designs and vice versa. Companies are assured that existing intellectual property will remain viable for the foreseeable future.

One area or particular interest to designers is the re-use of stimuli for verification. It has been estimated that the creation of stimuli for a given design module can consume roughly 25% to 75% of the time spent on the design task. With language independence, stimuli developed for a Verilog design can be used on a part developed, synthesized, or re-implemented in VHDL, or vice versa.

6.3 Increasing Choice

Along with the benefits of design re-use, design teams are free to use VHDL, Verilog, or a combination of the two, depending on which is best suited for a particular design. This choice is important to the EDA industry because both languages offer advantages. For instance, VHDL includes constructs that make it better suited for designs at the behavioral level (such as VHDL types, package and library definitions, third party tools, and specialized packages).
Verilog enjoys the widest support among ASIC vendors today and is the predominant verification methodologyfor many designers, although with the IEEE-1076.4 VITAL specification VHDL designers will have significantly increased ASIC library support in the future.

One final advantage of language independence is that designers are essentially free to choose among ASIC vendors and libraries. The laws of marketplace economics say that an increase in vendor selection should help lower costs and improve productivity.

Of course, not everybody will be happy with the new state of affairs. Less efficient ASIC vendors and library suppliers may find their positions further eroded, while their more successful brethren pull further afield. Tool vendors who do not provide the flexibility of mixing VHDL & Verilog will likely see their revenues and customer base shrink as well. But for the industry as a whole, language independence is an opportunity to increase the re-use of intellectual property while improving productivity in the development of new designs. At Mentor Graphics, language independence will be a key feature in the joining of smaller blocks and cores into Systems on Silicon, a trend that will help propel the EDA industry into the twenty first century. In the simulation field, Mentor Graphics' new product QuickHDL(tm) provides the ability to seamlessly mix VHDL and Verilog languages today.

6.4 Simulation Technology

QuickHDL(tm) is the next generation of Mentor Graphics' existing QuickVHDL(tm) product that has allowed Mentor Graphics to meet the needs of VHDL designers. Like QuickVHDL, QuickHDL uses Direct-Compiled Code Technology to reduce HDL descriptions into a compiled database of binary generic RISC instructions. These instructions are then easily mapped to the target workstation instruction set and executed by the new Single Kernel Simulator (SKS). This methodology allows for fast compilation, fast simulation, seamless language mixing, and platform independence for compiled designs.

Most simulators allow the use of different models through the technique of co-simulation. This allows two different simulators to pass results between them to support mixed simulation. Some products today which are not positioned as co-simulation products, but allow model importing are actually co-simulation products since two different simulation kernels are used. The simulation kernel is the part that evaluates results based on the design, stimulus, and pending events. QuickHDL has only one kernel that is capable of executing the instructions compiled from VHDL or Verilog without the performance and memory overhead of a multiple kernel approach. The SKS technology also supports the use of foreign models such as LMG SmartModel libraries, LMG Hardware Modelers, and models written in C.

However, no approach is correct for all situations. While a single HDL simulator is best for mixing the VHDL and Verilog languages, supporting other existing model types such as Mentor Graphics' QuickSim II(r) models or Continuum(tm) analog models would be inefficient within a single simulator. For this reason, QuickHDL supports an optional co-simulation interface to allow the mixing of VHDL, Verilog, schematics, or analog models.

Once a simulation tool provides performance and support for the necessary models, a designer's needs turn to validating the design. Due to the increasing complexity of today's designs, the need for sophisticated and easy to use debugging capabilities can have significant impact on the overall design cycle. QuickHDL supports the ability to view source code, view current values (on signals, variables, registers, and wires), view results in a graphical waveform or tabular list format, view the design structure, view HDL connectivity in a dataflow format, and see pending events in processes. Since these capabilities are being provided in a Single Kernel Simulator, all these services are available for either VHDL, Verilog, or mixed HDL designs without restriction.

Once the designer has done some initial validation, the need often arises to use a scripting language for building routines that check for complex conditions or for use in automated regression tests. QuickHDL uses the industry standard TCL (Tool Command Language) scripting language format for this purpose like it's predecessor QuickVHDL, which was the first commercial simulator to support TCL.

6.5 Links to Design Flows

Of course no matter how efficient a simulator is, it just part of the total design process. In order to make the design process more productive, a simulator must have excellent links to upstream and downstream tools. QuickHDL allows for data input from multiple sources such as text editors (HDL text format), Mentor Graphics' Design Architect(tm) (HDL text format, schematics), and System Architect(tm) (data flow diagrams, state tables). When non-text input mechanisms are used, QuickHDL uses links to the entry tools to allow debugging with graphical representations.

Since many HDL designs will be synthesized, QuickHDL also has the ability to perform synthesis checks for Mentor Graphics AutoLogic II(tm) synthesis tool at compile time. This prevents the designer from spending significant time simulating a design only to find out that it is not synthesizable and requiring design and validation work to be repeated.

After synthesis, verification and timing simulation is the final simulation step. QuickHDL supports the industry standard SDF file format for backannotating delays to either VITAL (VHDL) or Verilog models allowing QuickHDL to fit existing flows and use delay information from ASIC vendor tools, user written tools, or other EDA vendor tools.

6.5 Summary

In order for designers to meet the challenges of tomorrow's designs, they will need to have freedom in model and library choices, efficient ways to validate and verify those designs, and open, flexible design processes. By taking the implementation independence enjoyed by HDL simulators and extending that to include HDL independence, QuickHDL provides a cornerstone technology that will enable designers to solve the problems facing Systems on Silicon.

Ruud Wijtvliet
MGB, authorized distribution Benelux
Enschede
Tel. +31 (0)53 4336665

Optimaliseren van het ontwerp-proces

Inleiding
Dit artikel probeert aan te tonen dat er een trend valt waar te nemen in het optimaliseren van het totale ontwerpproces en dat de fabrikanten van elektronica, die willen overleven, nieuwe ideeën zullen moeten toepassen op het gebied van elektronica ontwikkeling teneinde in de toekomst nog een rol van betekenis te spelen.

Fabrikanten zijn of worden in de komende jaren geconfronteerd met enorme problemen aangaande het plannen en ontwikkelen van nieuwe produkten. Deze problemen worden groter wanneer tegelijkertijd een steeds groter wordend aantal bestaande produkten in de markt gehandhaafd dient te worden. 'Time-to-market' en 'First-time-right' design zullen meer dan ooit van vitaal belang zijn voor het succesvol introduceren van nieuwe produkten.

1.2 Traditionele produkt levenscyclus

Produktlevenscycli worden steeds korter. We zagen vroeger dat de ontwikkeling van het ene produkt alléén de volgende overlapte. Zodra er een produkt uit de markt werd genomen, nam een ander zijn plaats weer in. Kenmerkend van dit model is dat gedurende de introductiefase van een nieuw produkt de ontwikkelkosten worden terugverdiend en dat bovendien in deze fase een hogere prijs berekend kan worden. Klanten zijn namelijk bereid voor de nieuwe mogelijkheden extra te betalen. In de daaropvolgende exploitatiefase zullen derhalve relatieve hoge winsten op een bepaald produkt gemaakt kunnen worden.

Ontwikkeltijd en time-to-market zijn niet echt van belang bij deze traditionele benadering. Een

Boompje opzetten?

Niet meer nodig. Boom staat al. De professionele elektronicus heeft voortaan aan één woord genoeg. Onder de projectnaam Sequoia bied Accel Technologies Inc. de mogelijkheden van **TangoPRO** en **P-CAD**, verenigd in een nieuw krachtig produkt: **ACCEL EDA.**

Dit verrassend veelzijdige programma voor Schema invoer, PCB lay-out, Routing en Bibliotheekbeheer werkt onder Windows 3.11, Windows NT of Windows 95.

De belangrijkste kenmerken:

- Split Power Planes: Intelligent Copper Pour met Ploughing, Thermals, Auto-recalculation, Movability etc.
- Net Classes vanuit Schema instelbaar
- Engineering Change Orders; DXF in en uit; Gerber in en uit; Koppeling met EMI/EMC simulators
- Hot Link Cross Probing; Multi Document Interface
- Mogelijkheden voor onderhouds-contracten, gebruikersgroep en BBS support

Kijk verder dan het bos groot is.
Kies die ene boom:
ACCEL EDA
Staat als een sequoia.
Bel voor documentatie of een demonstratie van **ACCEL EDA** en breng meer efficiency in uw designs.

Tech 5 B.V. Rivierdijk 654
3371 EE Hardinxveld-Giessendam
Tel. 0184-615551 Fax 0184-615451

Dealer België: European Technologies 03-6440169

innovatieve elektronische oplossingen

Titel: Werken met SPSS Base
Uitgeverij: Sybex
Voor Nederland:
De Muiderkring B.V.
Bestelnr. 750.737
Prijs: fl. 69,-

Als er één terrein van onderzoek is waar de computer al snel onmisbaar bleek, dan is het wel statistiek. SPSS is een programma waarmee u statistische analyses uitvoert op meetgegevens. Werken met SPSS gaat er vanuit dat de grondbeginselen en de terminologie van de statistiek u bekend zijn. Van computers hoeft u niets af te weten. Het tweede hoofdstuk geeft onder andere een gedegen opleiding in het programma. Hoofdstuk 3 geeft uitleg over de data-editor, de plaats waar de gegevens worden opgeslagen. U krijgt een antwoord op vragen als: hoe zijn de gegevens gestructureerd, hoe bepaalt u het type ervan en hoe voegt u nieuwe gegevens toe of verwijdert/wijzigt u bestaande? Hoofdstuk 4 behandelt het bestandsbeheer en in hoofdstuk 5 wordt aandacht geschonken aan het 'output'- en 'syntax'-venster. Hoofdstuk 6 laat zien hoe u de gegevens in de data-editor bewerkt. Aan de orde komen zaken als sorteren, samenvoegen/splitsen van een aantal bestanden en het wegen. Hoofdstuk 7 toont hoe u data kunt transformeren om bijvoorbeeld niet relevante gegevens te filteren.

HIOKI

De VEILIGE DMM:
De HIOKI 3256

- Autom. busafsluiter
- 600 V -zekeringen met hoog afschakelvermogen
- Display tot 5610
- Hoge gevoeligheid: 10nA/100µ V
- Freq. meting tot 400kHz
- Snelle Bar-graph (56 segm.)
- "Hold-auto" -functie (max U, max I, min R)

HARTOGS

Ingenieurs Bureau Hartogs B.V.
Postbus 9393, 3007 AJ, Rotterdam, Tel. 010-4795700, Fax 010-4793342
Hartogs Belgium
Kalenbergstraat 31, 1700 Dilbeek, Tel. 02-5695647, Fax 02-5690282

goed produkt is namelijk beschikbaar en de ontwikkelkosten kunnen worden betaald uit de prijsopslag van een nieuw produkt tijdens de introductieperiode.
Echter bedrijven en individuen die nog steeds het enkelvoudige beeld van produktontwikkeling gebruiken, werken met een gevaarlijke misvatting en lopen het risico overgenomen te worden door de concurrentie of met een nieuw produkt de markt volkomen te missen.

1.3 Huidige en/of toekomstige produktlevenscyclus

Als gevolg van het elkaar steeds sneller opvolgen van nieuwe technologiën en invloeden vanuit de markt, is de produktontwikkelperiode veel korter geworden. Dit heeft tot gevolg dat de beschikbare tijd tussen de start van de ontwikkeling en het moment van introductie van vitaal belang is voor het succes van een nieuw produkt en dus voor het succes van de onderneming.

Dit heeft enkele zeer belangrijke effecten tot gevolg:
Verschillende versies van een produkt zullen nu tegelijkertijd op de markt aanwezig zijn, waarbij de complexiteit en de compactheid van de ontwerpen zal toenemen. Daarnaast zal de druk op een concurrerende (kost)prijs en de kwaliteit blijven bestaan of zelfs toenemen;
Produktversies zullen sterk verschillende levensverwachtingen hebben;
Produktontwikkeling zal een continu proces worden.

In plaats van een ontwikkeling te zien als een eenmalige investering en af te schrijven tegen toekomstige winsten, is tegenwoordig ontwikkeling een constante en terugkerende kostenpost en daarom zal hiervoor een budget moeten worden vrijgemaakt.

1.4 Trends in de Electrotechnische Industrie

Zoals reeds in de vorige paragraaf uiteengezet resulteert een kortere produktlevenscyclus in het feit dat het produkt in een kortere tijd zijn investeringen moet hebben terugverdiend waarbij doorgaans de kwaliteit, complexiteit en compactheid gehandhaafd moet blijven of zelfs moet worden verbeterd. Er vindt een verschuiving plaats naar een continue produktontwikkeling in plaats van periodieke ontwikkelingen ten tijde van de noodzaak van een nieuw produkt.

Genoemde verschuiving stelt andere eisen zowel aan de ontwikkelaar als aan het bedrijf. De ontwikkelaar zal een gezonde mix moeten vinden in een combinatie van ontwikkelaar, produktmanager en service engineer. Daar tegenover staat een noodzakelijke verandering in de manier waarop het bedrijf nieuwe ideeën en technologiën implementeert. Indien technici een bredere rol moeten vervullen en dus algemener gericht zullen zijn, kan men niet verwachten dat ze alle technieken kunnen beheersen om een nieuw produkt te ontwikkelen. Uitstekende hulpmiddellen voor de ontwerper zijn hierbij natuurlijk de EDA ontwerp tools.

De vraag werpt zich op voor welke technologie implementaties de eigen technici worden gebruikt en voor welke niche- of speciale technologien partners kunnen worden ingeschakeld.

Kenmerkend zijn dan ook de volgende trends in de elektrotechnische industrie:
Design anywhere / make anywhere. Er ontstaan ontwerpende bedrijven en producerende bedrijven. De keuzemogelijkheid tot uitbesteden speelt hierbij een belangrijke rol.
Increasing clock-speed. De vraag naar meer functionaliteit in minder tijd dwingt de elektronica ontwerpers met een steeds hogere klokfrequentie te gaan werken.
Decreasing time-to-market. De produktlevenscyclus wordt korter, verschillende versies van hetzelfde produkt zullen tegelijkertijd op de markt aanwezig zijn met als gevolg dat het moment van marktintroductie van een produkt van vitaal belang is geworden.
Increasing software content. (Embedded) Software zal bij elektronica ontwikkeling een steeds groter rol spelen.

1.5 Systeemontwerper en zijn gereedschappen

Degenen die als eerste worden geconfronteerd met de gevolgen van een kortere produktlevenscyclus en de trends in de elektronica industrie, zijn de systeemontwerpers. Onder een steeds hogere tijdsdruk moet de ontwerper meer designs produceren en tegelijkertijd onderhouden. Daarnaast moet dezelfde ontwerper in staat zijn op de hoogte te blijven van de technologische ontwikkelingen en deze weten te implementeren in de nieuwe designs.

De vraag die een ieder zich hierbij stelt is: op welke manier kan de ontwerper en dus ook het bedrijf binnen de beschikbare tijd en beschikbare kennis toch de gewenste ontwerpen op tijd en conform specificatie leveren. Het antwoord hierop is niet in enkele woorden te geven, maar een essentiële rol hierbij is wel dat de totale Design Flow en niet de afzonderlijke ontwerpprocessen (specificatie, functioneel ontwerp, detailontwerp, implementatie, verificatie, testen, etc.) geoptimaliseerd moeten worden. Het gaat hierbij om een totale geïntegreerde oplossing die het de ontwerper mogelijk maakt zich bezig te houden met zijn taak; 'het ontwerpen van elektronica' en niet met het op elkaar afstemmen van de afzonderlijke processen en/of gereedschappen.

Veruit het grootste deel van het niet juist functioneren van een bepaald ontwerp ontstaat in de specificatie fase. Het controleren van het gedrag van het totale systeem in een zo vroeg mogelijk stadium, zonder gekozen te hebben voor de uiteindelijke technologie (hardware, software, ASIC, FPGA, etc.), maakt kostbare redesigns, het uitstellen van de introductie van een nieuw produkt en/of extra prototypes overbodig.

Een voorbeeld waaruit de essentie blijkt van een geïntegreerde Design Flow in plaats van geoptimaliseerde afzonderlijke ontwerp-processen is het volgende: Met grote regelmaat worden de noodzakelijke wijzigingen in het lay-out proces niet teruggekoppeld naar het functioneel ontwerp met als gevolg dat bij een re-design van het eerste ontwerp de lay-out designer opnieuw de problemen van het eerste design moet oplossen. Deze vorm van 'over the wall' ontwerpen is funest voor het time-to-market principe en zal derhalve nooit kunnen resulteren in 'first-time-right' ontwerp.

Een tweede kenmerkend voorbeeld om de sleutelfactor 'time-to-market' te reduceren is het tegelijkertijd ontwerpen van hard- en software en de daarbij behorende simulatie. Het is een bekend verhaal: Nieuws over het nieuwe produkt lekt uit naar de markt. Klanten vragen naar het produkt en de verkopers kunnen niet wachten met het onder de aandacht brengen van de nieuwste ontwikkelingen. Praktisch iedereen in het bedrijf kijkt uit naar de introductie van het nieuwe produkt. Iedereen, behalve de ontwerpers aangezien zij wanhopig proberen de software op tijd klaar te stomen.

De vraag die hierbij telkens weer opdoemt is, waarom wordt er nog steeds gewerkt aan de software ontwikkeling? Het simpele antwoord luidt dat de software pas wordt getest en gedebugged nadat de hardware getest en gedebugged is. Het functioneel testen cq simuleren van het gedrag van het totale systeem zou de zekerheid hebben gegeven dat het systeem conform de specificatie wordt ontworpen. Bovendien zouden in deze hardware/software co-design omgeving beide disciplines tegelijkertijd hun werk kunnen verrichten met als gevolg dat de totale ontwerptijd aanzienlijk afneemt.

Samenvattend zijn de meest belangrijke sleutelfactoren ten aanzien elektronica ontwerp methodes dan ook:

Hoog niveau ontwerp-methoden (High Level System Design)
Hardware/Software co-design
Integratie van het gehele ontwerp-proces vanaf specificatie tot aan realisatie en implementatie.

1.6 Conclusie

Belangrijk bij de keuze van een ontwerptool is dat het bedrijf niet alleen de capaciteiten van de afzonderlijk tool in ogenschouw dient te nemen maar juist moet analyseren hoe de totale Design Flow verbeterd kan worden. Uit het voorbeeld van Hardware/Software co-design blijkt heel duidelijk dat het niet alleen gewenst is, maar zelfs noodzakelijk is, dat gedragssimulatie van het gehele systeem in zo vroeg mogelijk stadium moet plaatsvinden. Alleen op die manier kan men voorkomen dat kostbare re-designs, design iteraties en/of prototypen tot een minimum beperkt worden. Dit zal tot gevolg hebben dat alleen bedrijven die deze methodes toepassen als eerste met het nieuwe produkt op de markt zullen verschijnen en derhalve hun concurrentiepositie zeker stellen.

Verder is het van groot belang dat gebruik gemaakt wordt van gereedschappen die het mogelijk maken technologie onafhankelijk te ontwerpen. Te denken valt hierbij aan het grafisch voorstellen van het ontwerp met de mogelijkheid om vanuit die grafische omgeving het systeem te simuleren en code te laten genereren van hogere beschrijvingstalen (VHDL/Verilog). Alleen met deze methodieken wordt het mogelijk de steeds complexere en compacter wordende elektronica ontwerpen te blijven ontwikkelen en te hergebruiken.

Tot slot maken deze methodieken het in de eerste plaats mogelijk de technologie onafhankelijke ontwerpen te hergebruiken in nieuwe designs. En ten tweede bestaat de mogelijkheid het bestaande ontwerp te implementeren in een nieuwe technologie zonder de noodzaak van een kostbaar re-design.

Ing. C. Stemerdink
MGB, authorized distribution Benelux
Enschede
Tel. +31 (0)53 4336665

Gaining Control Over the Escalating Complexity of Design Data

Abstract

Over the past decade substantial improvements have been realized in the electronic design process through the use of computerized tools that automate various tasks in the design process. The impact on product performance, quality and time-to-market has been dramatic. However, along with the use of these tools has come new issues that add complexity to the process and, if not addressed, can have adverse effects on achieving overall design productivity goals. This document describes the issues that arise from the productivity of electronic data and the solutions that are available to address the problems.

2.2 Introduction

The use of EDA tools has resulted in an exponential rise in the numbers of interrelated computer generated files that often must be shared among design team members. Because tools like simulators, design synthesizers and autorouters make it easier to consider different approaches, more design alternatives are evaluated resulting in more design configurations and versions. These versions are maintained in either local or shared file systems, usually without any automated means of design file management. The growing use of concurrent design practices requires that these design files be easily accessed by both the designer and other team members. As the complexity of locating the correct files and versions of files gets out of control, design productivity suffers. Another design productivity issue is the infrequent reuse of older designs. If it is difficult for engineers to search and locate previous designs, they usually resort to starting over from scratch. This practice of "reinventing the wheel", can be avoided by providing easy searches and access to older designs for reuse on new projects.

Design Data Management (DDM) and Product Data Management (PDM) systems are two different but complementary solutions to the problem. Both solutions use "storage managers" or electronic vaults to store and provide access to design files. The first (DDM) addresses the needs of engineering workgroups where the design is rapidly changing. The second (PDM) addresses the broader needs of the organization as the product is readied for manufacture. At this stage, because the various users are located throughout the organization, more formal configuration management revision control, release procedure and engineering change order management is needed.

Design Data Management (DDM) provides high level searching, browsing and access controls that are needed by the designer or workgroup members to make it easy to access the correct versions of a design without the worry of multiple designers simultaneously making changes to the same design files. This type of control system prevents the loss of days or weeks that often results from accessing incorrect versions or redoing work that has already been done.

As designs are released to manufacturing, electrical design data joins other product data that may include the bill of materials, mechanical drawings, manufacturing drawings, manufacturing automation files, manufacturing instructions, product documentation, etc. Design access and control issues that arise at this level are common concerns across the enterprise rather than simply within the electrical design workgroup. As design information is promoted to the enterprise level, the information changes less frequently and a larger number of people need input and access than at the workgroup level. Product Data Management (PDM) systems are used to manage the information that is required to build and support the complete product. The integration of electrical data into these "storage managers" requires a detailed understanding of electronic design tool data structures and is usually the most difficult aspect of PDM system integration.

2.3 What Is Data Management?

Data Management is the management of all the product information flowing through an organization that is required in the development of new products or in updating of current versions of products. This information may be held in files or in levels of abstraction in a database. A data management system should make it easy to understand and control all of the design data such that authorized team members can quickly gain access to stored design information to either view, edit or reuse the design without concern that others are changing the same design.

Today's enterprise PDM solutions do little to address the needs of electronic design workgroups. The reasons for this are that electronic data, in the design phase, changes much more frequently than after the design is released and electronic design data is fundamentally different than the majority of other product data. Some of these differences are as follows:

Electronics data is highly abstract and often has no direct relationship to a physical entity that a person can see, feel or touch. This makes it difficult for a person that is not expert in electronic tool data structures and design practices to characterize electronic design data so that it can be properly and efficiently managed by a PDM system. Schematics, netlists, simulation results, EEPROM, PROM, and PAL data are examples of abstract data entities and the rules that must be obeyed to incorporate design changes are not at all intuitive.

Every EDA system has its own proprietary set of data structures that often are not well documented. Experience in dealing with electronic design data is necessary to understand the data model and is thus a prerequisite in integrating electronic design data in a PDM system.

A large percentage of the information that makes up an electronic design is sourced from component libraries that support the design tools. These libraries undergo constant change. Managing design files without managing the source of the component data can result in incomplete or inaccurate product data. Thus component libraries need to be managed and the version numbers referenced in the design data as well.

Electronic designs contain many levels of hierarchy and often a design block is replicated many times with each instance containing different reference designators. In managing electronic data, it is important to maintain definitions of this hierarchy so that rapid access at any level is possible without encountering time consuming "unpacking" procedures.

ASIC design is often entered behaviorally. This level of abstracfion is twice removed from physical design. Dependencies between pre and post synthesized representations compound the logical/ physical data dependencies found in PCB design.

Most solutions that address the DDM issues have come from EDA companies. While these companies certainly understand electronic data issues, most of their efforts in data management have been too limited in scope. In one approach, dedicated DDM products for electronics data have focused exclusively on the data from a single vendor. This is typical of the wall to wall EDA supplier that assumes all EDA tools will be purchased from a single suplier. Another approach involves the EDA vendor attempting to sell a proprietary storage management system. This approach denies a whole industry outside of EDA which focuses on providing the best storage management technology. In either case, requirements for openness and ease of integration with other storage managers has fallen short of customer needs.

2.4 Why is Data Management Needed?

While there has been enormous improvements in the productivity of individual design tools, the overall sophistication of new products is growing faster. Competitive forces dictate that to survive, companies must strive to obsolete their own products before competitors do it for them. A widely quoted study by McKinsey & Co. found that with a reasonable set of market assumptions, a six month product slippage results in a

33% erosion of net profits.

In industries such as personal computers, printers and peripherals, the total product life cycle is often less than one year. Businesses everywhere are looking for ways to get more with less. While CAE, CAD and CAM solutions are now standard in most engineering departments, procedures for validating, reviewing, circulating and approving changes is still largely done on paper. The review process itself, often takes more time than the implementation. The quantity of data produced is rising exponentially and many companies are finding that just when they thought they were computerized, the largist obstacle to improved productivity is still paper. Companies are taking a broader look at their overall processes and initiating reengineering programs to gain higher productivity. A common goal is to reduce the overall reliance on paper driven processes. Some of the biggest growth markets in business systems today are in automation solutions that eliminate lost time in processing requests and in notifying people of changes. The market for workflow automation systems, automated document imaging systems, and groupware are all experiencing tremendous growth. The equivalent to these office automation solutions in manufacturing is Product Data Management systems. According to CIMdata, the market for these systems is expected to grow at a compound annual rate of greater than 30% over the next three years. The appeal of these systems is that they don't address just one or two bottlenecks, but rather focus on the entire process.

Product quality continues to be one of the most important critical success factors in business today. Through the leadership of the International Standards Organisation (ISO), the ISO 9000 standards have come to be viewed as an important buying criteria. Buyers are looking for ISO certification as an indication of quality. The expectation is that if a supplier has its product process well documented and under process control, then the product will have higher quality. Data Management systems provide significant value in helping organizations document their process and later in demonstrating to auditors that their process is under control.

2.5 Data Management and the Philosophy of Viewlogic Systems, Inc.

The business approach of Viewlogic Systems has always reflected its "best-in-class" philosophy that recognizes the need for electrical engineers to mix and match the best tools for a given job, regardless of who supplies them. The company has long been a leader not only in supporting the adoption of industry standards through its active participation in groups such as CFI, IEEE and EDAC, but also in the implementation of those standards. For example, Viewlogic Systems was the first EDA company to offer both EDIF (IEEE standard for schematic exchange) and SCHEME (CFI scripting language) compatibility in its products. It was the first company to provide a fully CFI compliant design representation. Rather than creating a new behavioral simulation modeling language, Viewlogic Systems was the first EDA company to offer VHDL. The company was also an early adopter of the MOTIF standard for user interface.

Viewlogic Systems believes that the data management technology required to store and retrieve data is outside the realm of EDA. Therefore the technology of any number of external data management companies may be employed within the design process. The aim of Viewlogic Systems is to add value as a best-in-class EDA company. As such, we make no assumptions about where the data is located, how to access it and we provide a totally open environment that supports the management of data, independent of which company tools were used to generate it.

The most recent advancement of Viewlogic Systems in the area of open systems has been with the release of its PowerCODE suite of capabilities. PowerCODE, an acronym for Customizable, Open Design Environment is designed to support the interoperability of disparate design tools. It is the architectural layer on which all tools are developed, and mirrors the best-in-class business model. Keeping vith the spirit of PowerCODE, the data management technology has been designed so that the data to be managed may be used by any tool, whether developed by Viewlogic Systems, its partners, a third party, or an in-house capability.

Peter Bakker
VIEWlogic
Bruistensingel 126
5232 AC Den Bosch
tel: 073 - 6408468
fax: 073 - 6408469
e-mail: pbakker@viewlogic.com

Integrated Design Solutions for Desktop CAD

De valkuilen van een complete ontwikkelomgeving voor Elektronische CAD.

Iedere ontwerper heeft behoefte aan een scala van ontwerphulpmiddelen, maar vaak verschilt de combinatie van die hulpmiddelen. Sommige ontwerpers hebben voldoende functionaliteit met alleen schema-invoer en handmatige lay-out. Wanneer er veel complexe designs gemaakt moeten worden kan een autorouter een efficiënt hulpmiddel zijn. Het vooraf simuleren van allerlei problemen zoals bijvoorbeeld EMI/EMC of temperatuur etc. bekort de ontwikkeltijd en voorkomt redesigns in een later stadium. Het uiteindelijke doel is zo snel en efficiënt mogelijk tot een produkt te komen.

Hier volgt een overzicht van veel gebruikte werkplek gereedschappen:

CAE
Schema-invoer
FPGA / PLD design
Simulatie Analoog / Digitaal
Timing diagrammen

CAD/CAM
AutoPlacement
PCB lay-out
EMI/EMC regels
Temperatuur huishouding
Routability
AutoRouter
CAM

Project Management
ECO van Schema naar PCB
ECO van PCB naar Schema
Versie beheer van tekeningen en PCB lay-outs
Bill of material
Spare-Parts list
Administratief projectbeheer

Er zijn leveranciers met alle functionaliteit in een complete werkomgeving. Keuze voor een dergelijk produkt betekent ook kiezen voor de filosofie van die leverancier en daarmee een grote afhankelijkheid. De filosofie bepaalt in grote mate hoe een ontwikkelproces verloopt en welke functionaliteit nodig is. Vaak zijn de prestaties van die functionaliteit maar voor enkele onderdelen echt optimaal. De rest van de onderdelen van dat pakket zijn vaak minder goed. Soms zijn die onderdelen niet gebruikersvriendelijk of ze bieden niet voldoende functionaliteit. In beide gevallen is er geen sprake van een snelle en efficiënte werkwijze. Het gevolg is dat tijdens de ontwerp-fase slechts enkele onderdelen gebruikt worden van de gehele omgeving. De gebruiker moet door een lange en moeizame leercurve.

Elke ontwerper zou graag de vrijheid hebben, met gereedschappen van gespecialiseerde leveranciers, een optimale omgeving samen te stellen. Voorwaarde is een goede samenwerking tussen de verschillende gereedschappen. Een aantal leveranciers brengen nu gereedschappen op de markt met een open in- en uitgangsstructuur zodat ze makkelijker samenwerken met andere produkten. Zo kan dus de optimale combinatie van specialistische gereedschappen gekozen worden. Een probleem bij toepassing van specialistische gereedschappen kan ontstaan door verschillen in de databases van de ontwerper en de board-ontwerper. Door de beschikbaarheid van nieuwe gereedschappen wordt ook dit op elkaar afgestemd.

De ontwerper kan bij de invoer van het schema al veel informatie invoeren voor de lay-out fase. Zo kunnen er al in het schema de spoorbreedtes, de clearances en door de gebruiker gedefinieerde attributen worden ingevoerd. Clearance regels kunnen nu niet alleen voor de shapes op het pcb ingesteld worden, maar al in het schema kan er bijvoorbeeld al een onderscheid gemaakt worden tussen High Voltage, voeding, Data en Clock signalen, analoge en digitale netten. Dit maakt het voor de lay-out ontwerper duidelijk aan welke eisen het board moet voldoen, terwijl de ontwerper weet dat aan zijn ontwerpvoorwaarden zal worden voldaan. Deze informatie wordt natuurlijk ook gebruikt door de autorouter.

Een grote bron van ergernis is de bibliotheek

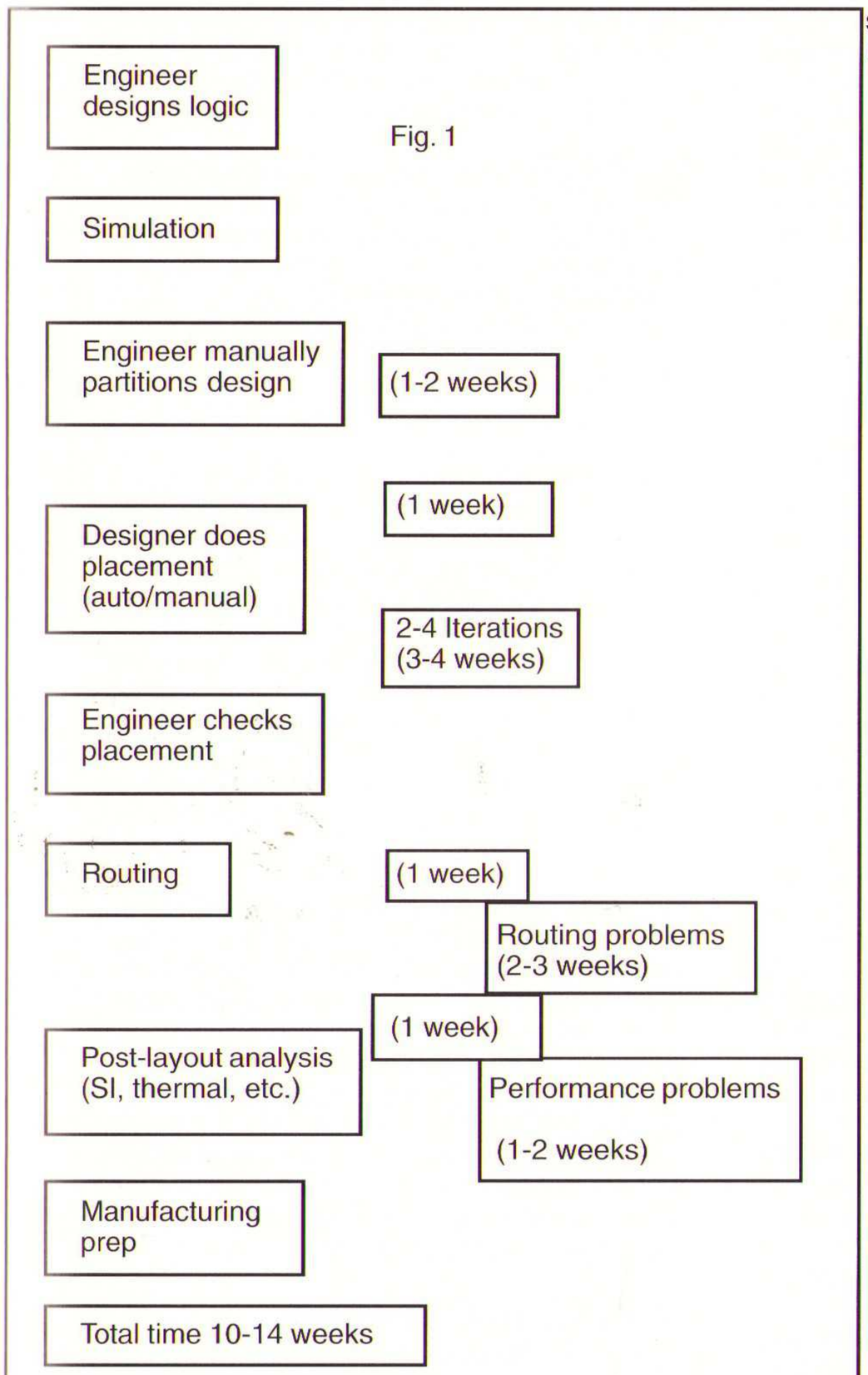


Fig. 1

van componenten en het beheer ervan. Er worden nu mogelijkheden geboden om met slechts één bibliotheek te werken waarin zowel de informatie voor schema-invoer als ook voor lay-out zijn opgeslagen. Dit geeft de ontwerper de mogelijkheid om bijvoorbeeld al bij het invoeren van een schema op te geven welke behuizing gebruikt wordt (DIP, SMD, TSOP, PLCC etc).
Het werken met specialistische gereedschappen van verschillende leveranciers is tegenwoordig zeker een optie. De produkten sluiten goed op elkaar aan en bieden dus de mogelijkheid alleen de voor de eigen werkwijze benodigde gereedschappen aan te schaffen. In de volgende opsomming staan een aantal veel gebruikte traditionele gereedschappen gerubriceerd en een moderne geïntegreerde oplossing. Het traditionele ontwerp-proces ziet er als volgt uit: zie fig. 1. Het moderne ontwerp-proces met physical design syntheses ziet er als volgt uit: zie fig. 2

Analyse van het ontwerp-proces.

Het ontwerp-proces heeft alle valkuilen in zich die, zonder integratie van de juiste gereedschappen, heel snel zichtbaar kunnen worden. De traditionele designmethode, waarbij een ontwerper de logica ontwikkelt en het schema "over de schutting" geeft aan de CAD lay-out specialist voor de routing, is niet adequaat voor de hedendaagse complexe, supersnelle ontwerpen met een hoge dichtheid. Dit resulteert in veel iteratief werk tussen ontwerper en board-ontwerper om een optimaal compromis betreffende alle relevante eisen te verkrijgen. Er dient rekening gehouden te worden met EMI/EMC regels, snelheid, oppervlakte, routability en produceerbaarheid van het board. Deze iteraties verspillen tijd en verlengen de 'Time To Market'.

De juiste combinatie van gereedschappen heft de 'scheiding' tussen CAE en CAD op. Met moderne gereedschappen kan snel een synthese lay-out gemaakt worden van de CAE data. De invloed van fysische factoren op de prestaties van de lay-out kunnen zo bepaald worden. Er kan geoptimaliseerd worden voor: produceerbaarheid, temperatuur, EMI/EMC etc. Er ontstaat zo een betere controle op de voortgang en kosten van een project.
Door de scheiding tussen de design disciplines weg te nemen wordt één van de grote barrières van elektronica ontwikkeling aangepakt. Ontwerp-processen worden zo bekort, de Time To Markt verkort, kosten verlaagd met een kwalitatief beter produkt.
Mocht om mechanische of lay-out technische redenen een andere behuizing voor een component nodig zijn, dan moet die informatie worden teruggecommuniceerd aan de ontwerper zodat het ontwerp ook de juiste informatie bevat. Dit moet dus met gereedschappen die in de omgeving van alle betrokkenen bij het tot stand komen van het ontwerp beschikbaar zijn.
Een goed bibliotheek beheerssysteem biedt de mogelijkheid de grafische presentatie van zowel schemasymbolen als pcb-footprint te bekijken. In deze presentatievorm kunnen zowel pinnen als pads gewijzigd worden, wat de efficiëntie zeer ten goede komt. Ook zijn er verbindingen naar tekstverwerking, DTP en Mechanische CAD pakketten etc. Door een pakket te kiezen dat gebruik maakt van de door Windows geboden faciliteiten wordt de efficiëntie vergroot en het gebruiksgemak verbeterd.

Vaak zijn de gebruikers van schema-invoer en PCB lay-out pakketten ook kundig in het zelf maken van software. Dat heeft er toe geleid dat er een DataBase eXchange (DBX) gereedschap beschikbaar is gekomen om eventuele, zeer specifieke gebruikerseisen te honoreren. Met deze DBX gereedschappen kan de gebruiker informatie uit de design database halen, wijzigen en er zelfs informatie aan toevoegen.

Rob Strik
Tech 5
Tel: 0184-615551

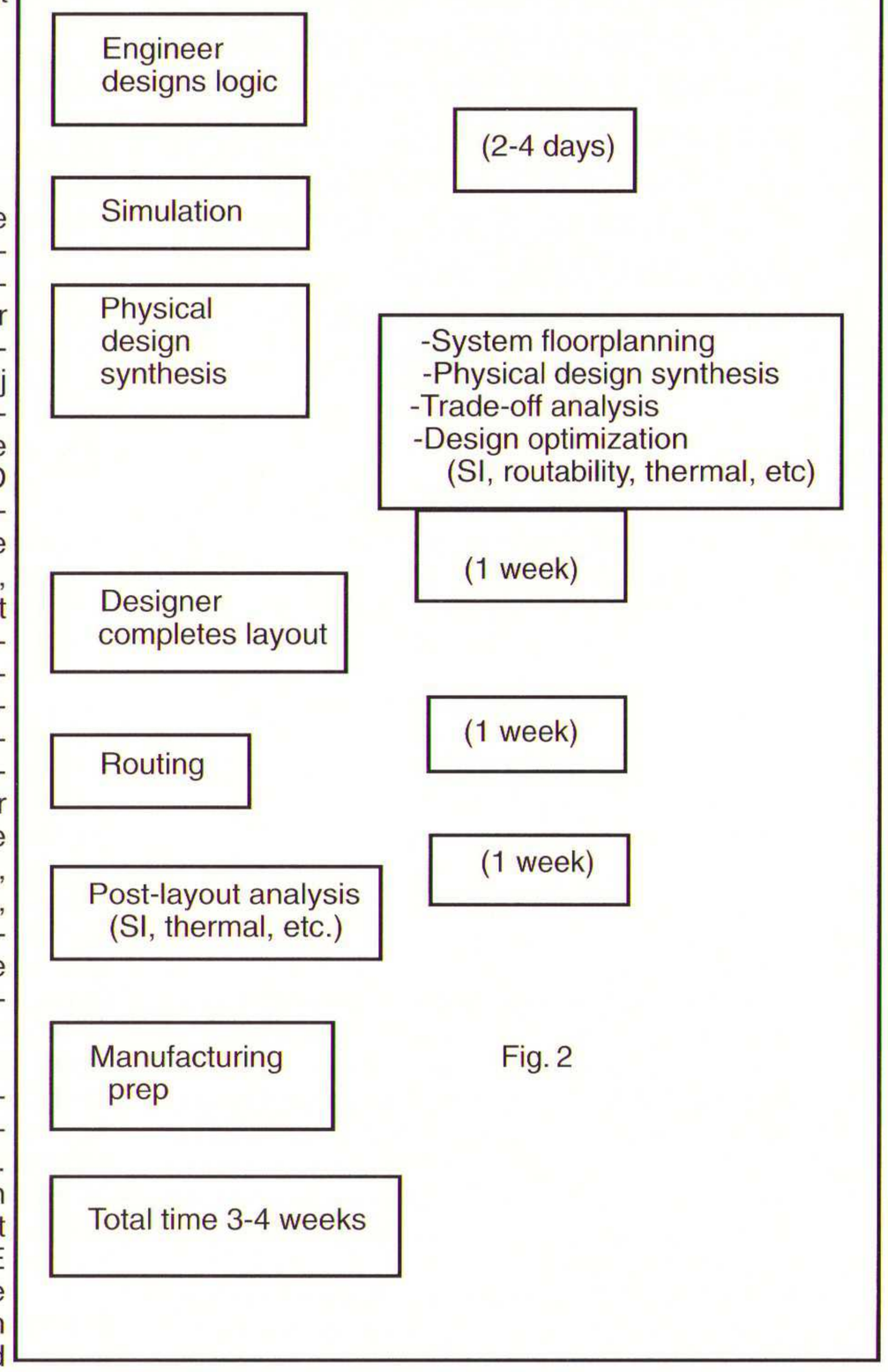
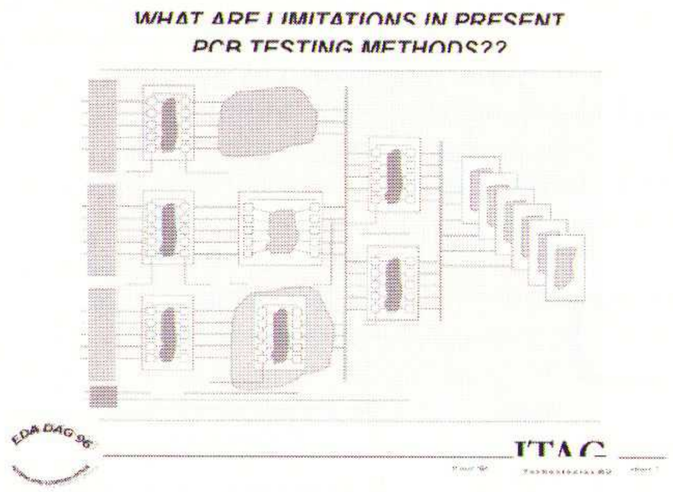


Fig. 2

When does Design-for-testability make Sense?

You are a designer so you don't have much to do with testing. As usual, the production department will take care of test program generation and the testing of the products you have designed. The test methods they use now such as Functional or In-Circuit Testing (ICT) are OK, but there are some clouds appearing on the horizon. More and more test engineering people are chasing you for Design-For-Testability (DFT) provisions in your

design, because nowadays new technologies such as complex devices with high pin-counts and also new types of packages (Ball-Grid-Arrays, SMT, etc.) mean they are faced with enormous test feasibility and access problems.



One of the possible provisions they might have asked for is Boundary-scan, also known as IEEE Standard 1149.1 'Test Access Port (TAP) and Boundary-scan Architecture'. As a designer you have probably read about it in the magazines, but for your designs it is perceived as not applicable due to lack of devices supporting Boundary-scan or it is considered to only add extra costs and effort.
Suddenly, it happens that the devices you are using apparently do have boundary-scan (or JTAG as some IC vendors name it). Examples are the PowerPC's of IBM or Motorola, Programmable Logic Devices from AMD, Altera, Xilinx or Lattice, or DSP's from Texas Instruments, Analog Devices or AT&T, or ATM devices from Fujitsu or Brooktree. And there are many more examples. Now, not only do you have an excellent opportunity to satisfy the testability needs of your production department, but also you can use these same facilities in debugging your first prototypes, resulting in much shorter development times. What's more you may even be able to program your PLD's on the board via the boundary-scan chain.
To become more acquainted with the subject boundary-scan, the next chapters of this booklet will discuss the reasons for introducing boundary-scan, by reviewing Functional and In-Circuit Test, the provisions that are necessary on board level and the benefits boundary-scan can bring to your company.

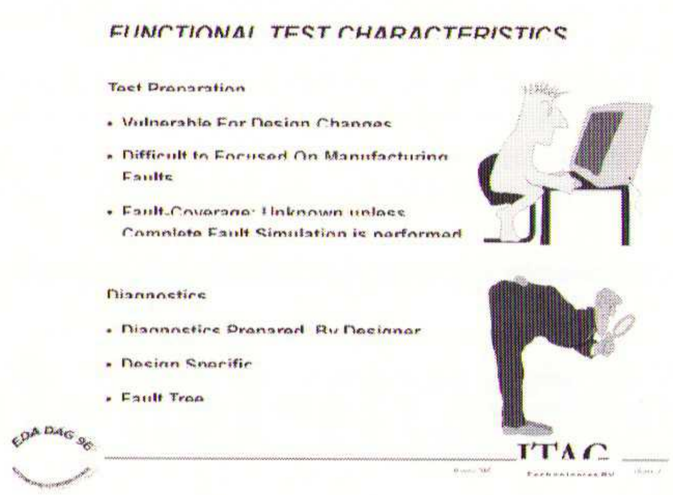
4.2 History of Testing

Functional testing is the "oldest" way of testing electronics. In the early days of the electronic industry many electronic systems were simply assembled, and the power was switched on. By checking the function of the system the "test" was performed. Nowadays some companies still have to work in this way. However growing complexity of systems has made test preparation a lengthy job while the fault coverage of such test programs remains unknown. Functional test diagnostics also requires highly skilled technicians in manufacturing. For this reason testing on Printed Circuit Board (PCB) level evolved. Testing was still performed in a functional way, but by dividing the problem, one could conquer ! But again the quickly increasing complexity of Integrated Circuits (IC) caused the same type problems as encountered at system-level functional testing i.e. that of long test preparation times, uncertain fault-coverage and poor diagnostics.

Next on to be widely adopted was In-Circuit Testing. By providing direct access to the entities to be tested on a PCB by means of a mechanical bed-of-nails fixture it was possible to test for manufacturing faults. This technology was well suitable to Dual-In-Line (DIL) packages and Plated-Through-Hole (PTH) PCB technology. But along with newer fine-line PCB-print technology and more complex IC-packages, such as SMT QFP's or BGA's with higher pin-counts and smaller pitches, the clouds in the test sky appeared nearer. Fixturing technology couldn't keep up with the ever decreasing dimensions of pins and pitches and the higher pin-counts of packages . Finally, as a result of industry co-operation which anticipated many of these problems, the boundary-scan method evolved as IEEE Standard 1149.1 Test Access Port (TAP) and Boundary-Scan Architecture. By
using this technique for testing many of the predicted draw-backs of the other test technologies could and would be overcome.

Following this evolution in testing methods a number of observations can be made:
- Design-For-Testability (DFT) has not always been an issue. It started to become important with functional board testing in order to increase ad-hoc controllability and observability of the function for test. However for testing today's designs DFT is inevitable.

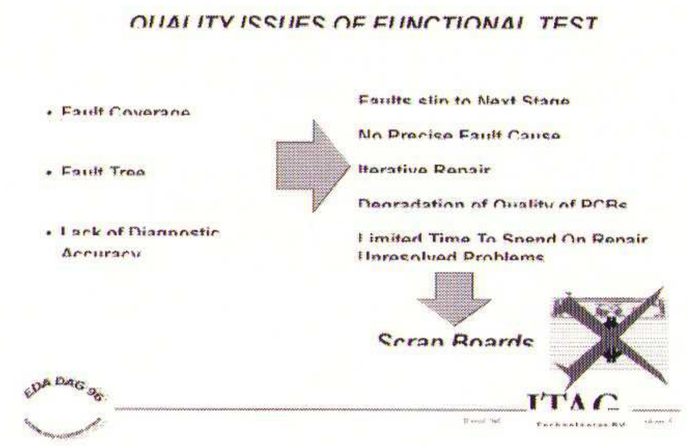
- Initially testing was a mixture of design debug and detecting manufacturing faults. Nowadays separate disciplines exist for design verification and manufacturing testing. Both ICT and Boundary-Scan Testing (BST) are well focused on finding manufacturing faults such as opens solder shorts, stuck at failures or missing/wrong components.



4.3 Functional Test

There are two draw-backs of testing in this way:
Firstly, the preparation of the test program itself is vulnerable to changes in the design. As a consequence of a smalldesign change the whole test development effort might be wasted. Furthermore, a functional test program is not focused on manufacturing type of faults which might end-up giving poor fault coverage. Also the fault coverage of a functional test program is not known, unless cumbersome fault simulators are used.

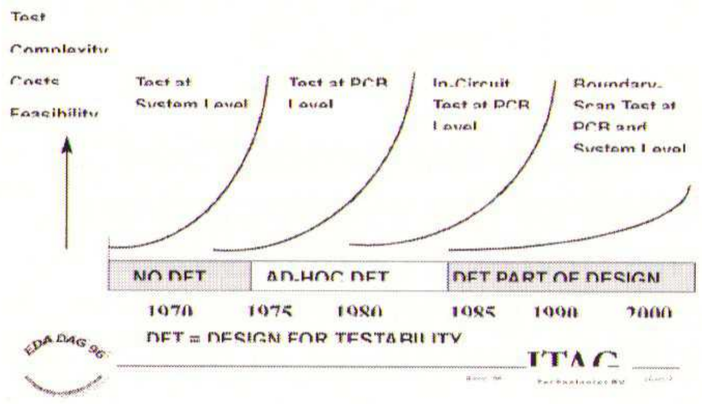
The second drawback lays with the diagnostics. Due to a lack of automated tools diagnostics are generally prepared by the designer (as is the test program), since this person is the most knowledgeable on this design.
However the designers time is a scarce resource and it is not often feasible to generate a precise, down-to-the-pin level, diagnostics routine, since that would take too much time. The result is an imprecise and lengthy diagnosis that will require trial and error repair methods. This will in turn impact in a negative sense on the quality/reliability of the delivered products. Alternatively if time limits are set to the repairs the result could be a pile of scrap-boards with unresolved problems waiting for re-work time capacity that never comes. Ultimately a waste of capital!

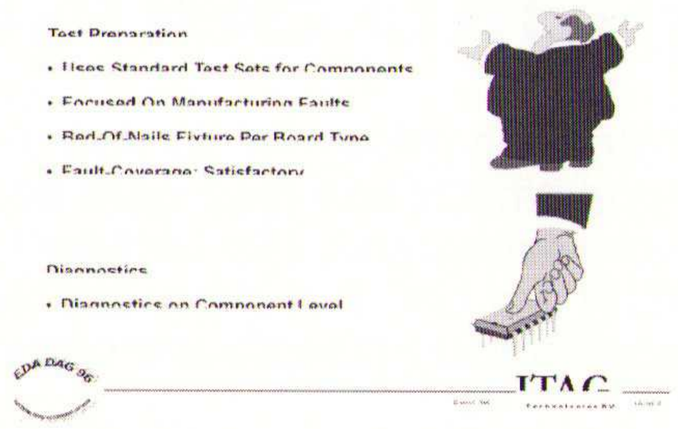


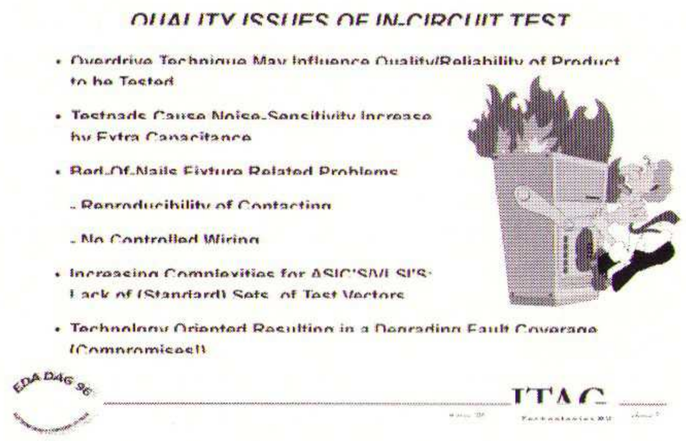
4.4 In-Circuit Testing

In-Circuit Testing evolved as a test method to overcome the functional test problems. By the use of standard sets of test vectors for each component it was easy to generate a test program well focused on the detection of manufacturing faults. Furthermore in ICT diagnostics is achieved directly on the component level. In order to realise the advantages of this test method a so-called bed-of-nails fixture has to be used. This type of fixture also has some negative points and as technology the negative points become drawbacks that make the use of this method impossible. Consider the following.

Firstly, the bed of nails gives mechanical and logical access to internal logical states. This intrusion in the logic, called back-driving, may have impact on the quality/reliability of the PCB, because if done incorrectly devices are used outside their specifications. In the manufacturing process these fixtures are a source of constant irritation for the management due to bad contact reproducibility. An unreliable throughput figure makes planning impossible!
The advent of ASICs and VLSI has also condensed the advantages of the "standard test vector sets". By not having the libraries for these devices available, or by receiving them late, ICT



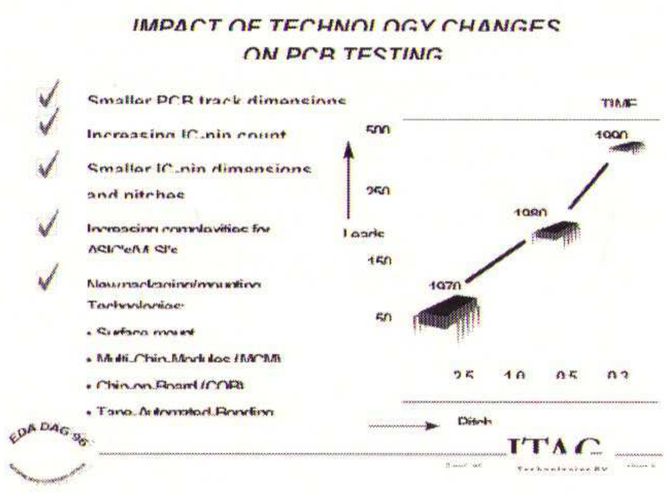




was performed without them by simply using (empty) sockets. However the final blow for ICT within production came with the introduction of new IC packaging technologies such as Surface Mount Technology (SMT), Ball-Grid Arrays (BGA) and new assembly techniques such as Multi-Chip Modules (MCM), Flip-Chip, Chip On Board (COB) or Tape Automated Bonding (TAB). All these new technologies make the mechanical access as required for ICT prohibitive. For this reason the industry invented a new answer to a new problem: Boundary-scan testing.

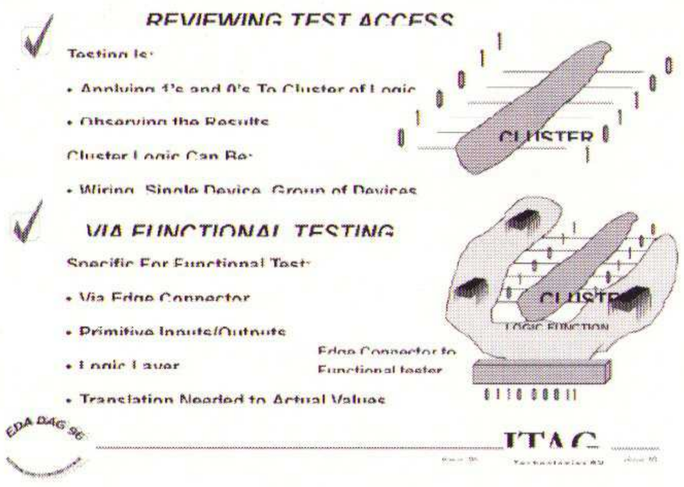
4.5 Test Access Review

To understand the solution offered by boundary-scan, let's first analyze the basics of testing in



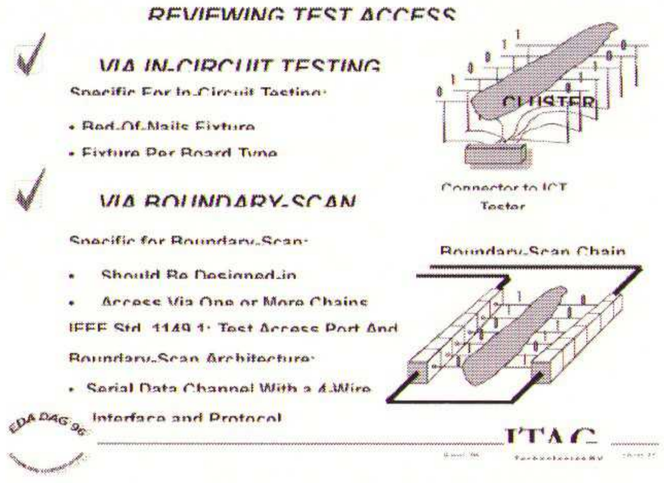
relation to test access. This can then be compared with the test access used by the various other test methods.

For testing digital logic ("cluster") logical values (1's and 0's) are applied to the inputs of a cluster and the response of the cluster in "1"s and "0"s has to be observed (see figure on page 7).



The response is then compared with the expected values at the outputs, in order to judge whether the test has passed or failed. The function of the logic comprised in the cluster could be simply wiring, a single IC, or a group of IC's. The fact remains that the "TEST" of such a cluster is independent of the test access method used. Thus, irrespective of the test methods used, it is only the access to the inputs and outputs of the logic cluster which is provided in differing ways. Let's explain.

Within a functional test the cluster to be tested is embedded in other logic. Suppose then that in the functional test the edge-connector is the only connection to the tester. Therefore, all test vectors have to be applied from the edge con-

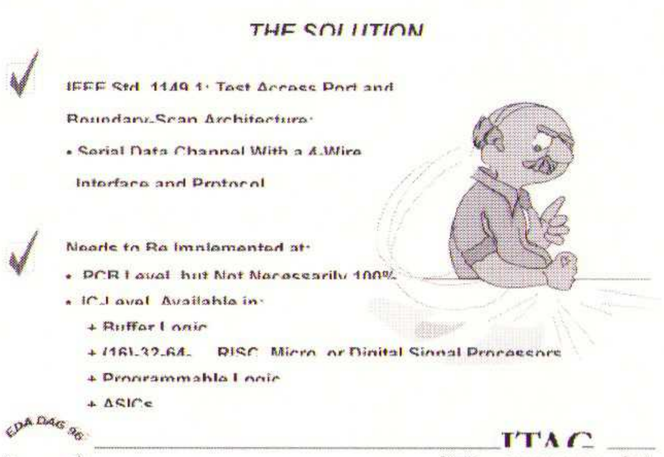


nector such that the intended test patterns appear at the inputs of the "cluster to be tested". It is that transformation which is the principle difficulty in the test preparation and can only be done with the knowledge of designer, or by a (fault)simulator. In the same way it has to be known what the expected output values will be at the edge connector. Again, this job can only be done efficiently by the designer (including the diagnostics in case the fault appearance is wrong), or by a (fault) simulator. Notice however that the "TEST" as such is independent of the surrounding function!

For In-Circuit Testing the test access to the "cluster to be tested" is quite crudely provided by the bed-of-nails fixture. Moreover the foot prints of clusters may vary from board to board type, often requiring specific fixturing per board. From the nails of the fixture wiring then connects the "test-points" to the actual ICT. For testing at high speeds this wiring is subject to the same design rules as must be applied to tracks on the PCB's in order to prevent problems such as cross-talk, ringing, etc. However, in practice use of such design rules in the preparation of pin-beds is often ignored leading to severe test debug problems. Notice though that the "TEST" itself is once again independent of the usage of a pin-bed.

4.6 The Boundary-scan Solution

In cases where the "cluster to be tested" is



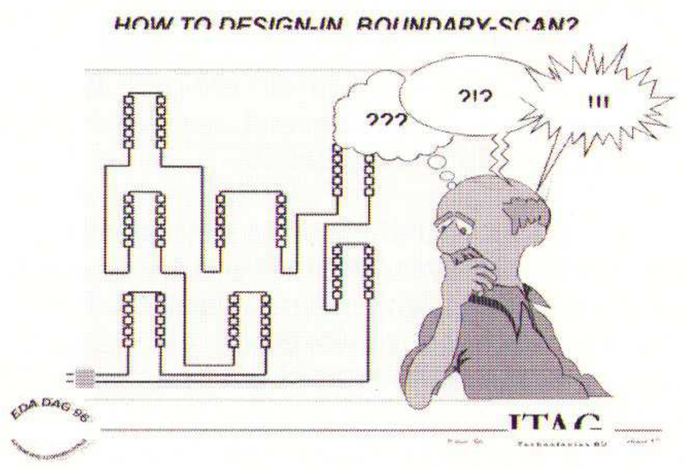
embedded by boundary-scan chains (on the PCB, or external of the PCB), test access can be provided via these chains. The test data is first shifted in serially from the tester, once the serial shift has been completed the test vectors are applied in parallel to the cluster inputs ("UP-DATE-PHASE"). The results of the test at the cluster outputs can then be parallel loaded in the boundary-scan chain ("CAPTURE-PHASE") and serially shifted-out to the tester, a comparison of actual versus expected data will be used to determine whether the test was a PASS or FAIL. Notice that the "TEST" as such is independent of the usage of Boundary-scan as an access method.

What's more the test vectors are still assigned to signal names (nets) in a design, for instance for AD0.....AD5 apply 110010 and at D0...D5 observed values are 001101.

Q:- Is it difficult to perform such a "test" with access via BST-chain ?

A:- NO, not at all, because the test remains described at the (parallel) signal level!

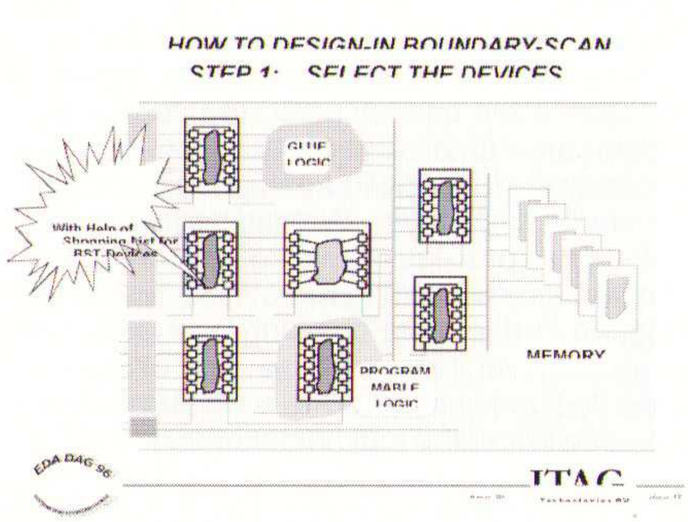
The reason is that the whole "machinery" of shifting-in and shifting-out according to the IEEE 1149.1 Boundary Scan protocol can be automatically performed by the software tools from JTAG Technologies. As a user you don't

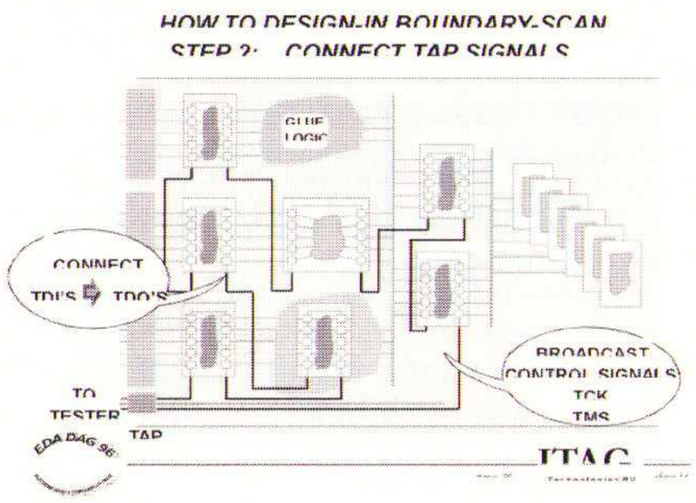


have to know all the in depth details, simply specify at net level the logic level you want to drive or expect to sense. It sounds unbelievably simple, but in our daily life we have similar examples. For instance, compare it to our telecommunications infrastructure. When you call someone to pass a message(10111) you don't want to be bothered with all the ISDN protocols involved just as your receiving party doesn't. The same applies with boundary-scan testing. JTAG Technologies SW takes care of transporting your vectors to the nets in your design and then the Vector Interface Package SW (VIP) reports the response of the test at net level to your computer.

4.7 What's Needed to Get Boundary-scan Access?

It would take too much silicon and cost overhead to specifically design-in a boundary-scan chain by using separate components. Fortunately, that's also not needed! Most IC-vendors have boundary-scan already built-in to their parts. Examples are found at Intel with 486, Pentium, Pentium Pro, 386 EX for embedded application, Motorola and IBM with the various PowerPC versions , Motorola with 68k family, Texas Instruments with DSPs, Sparc and buffers, and bus logic, National Semiconductor with buffers, AMD with 29k, Mach 4/5 , Digital with



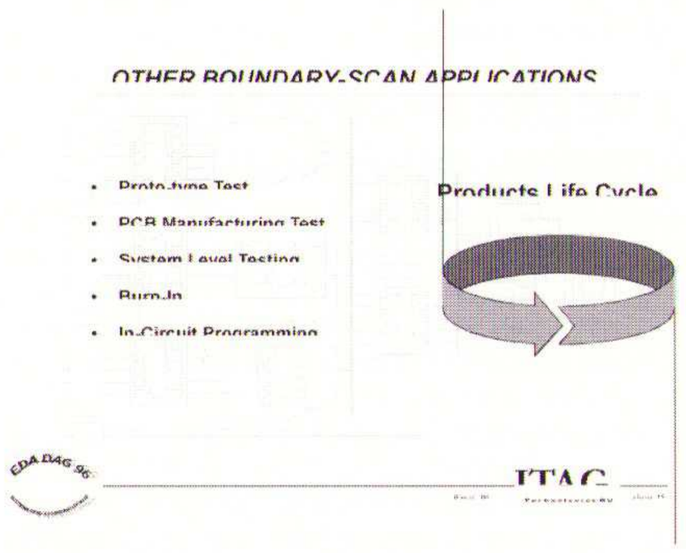


the Alpha chips and many others. Ask your JTAG Technologies' sales representative for the free of charge "Boundary-scan Parts Shopping List". Moreover (and contrary to popular belief), it is not absolutely necessary to have all devices equipped with BST. For example during the review of access to the "cluster to be tested" (see Chapter 4) cluster components could easily be non-boundary-scan parts! What's more, there are several industry examples known, where from just one BST device, an entire PCB could be controlled and observed and thoroughly tested (including memories !!).

To utilise the boundary-scan in your design is also a simple matter. Whether you have selected boundary-scan components intentionally or by accident, all that is required is for the chain(s) to be connected at the start of your design. Firstly each boundary-scan data output pin ("TDO") has to be connected to a next boundary-scan data input in ("TDI") and for control of this 'test machine' which includes the shifting operations etc.. each boundary-scan device has to be connected to the test clock ("TCK") and the test mode select ("TMS") signal. As for the other logic functions, normal design rules apply for the layout!

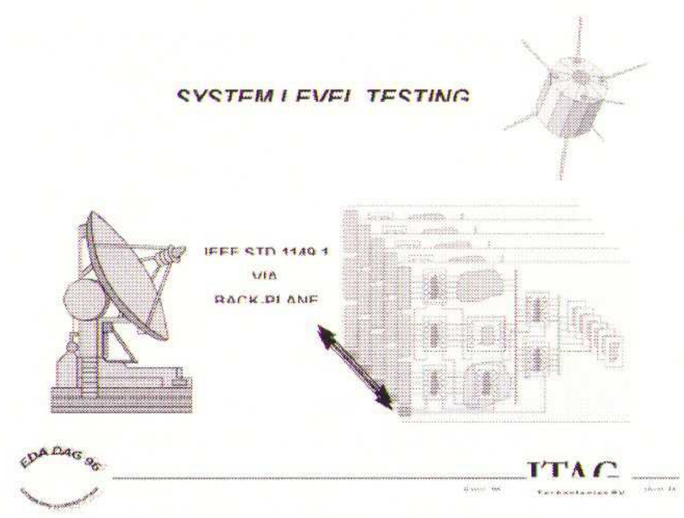
4.8 What Can You Do With Boundary-scan?

Boundary-scan was invented to overcome the manufacturing test problems evolving from In-Circuit Testing and use of SMT devices. The

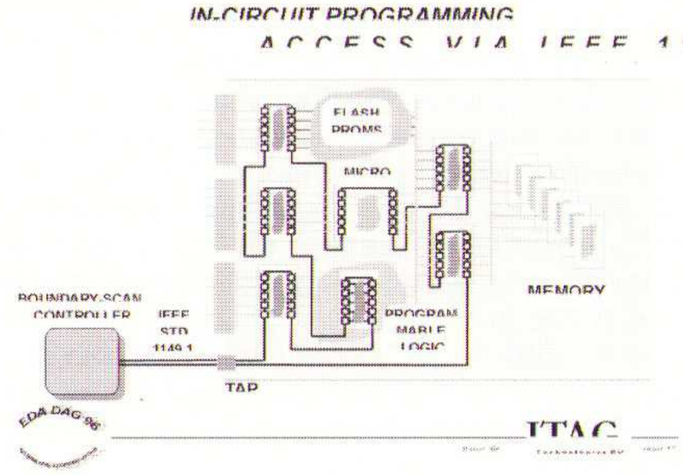


several test steps that can be conducted via boundary-scan will be discussed further in Chapter 8.
However it has turned out that manufacturing testing wasn't the only application for this serial bus. Due to the easy access, the ease of test preparation and the low costs of the tools it is often successfully applied in testing prototypes for manufacturing faults. Without boundary-scan facilities this process may take several days or even weeks, consuming the scarce time of the designer. Furthermore by using boundary-scan the prototype test can be performed by production personal, since no special knowledge is needed about the logic functions of the PCB.

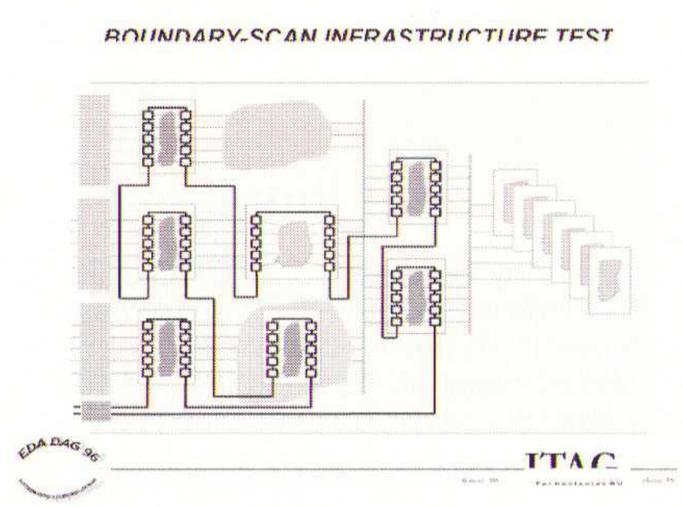
System level testing is another example of a new application area for boundary-scan. By extending the chains to the back-plane test access to the PCBs from system level can be achieved. There are different ways to implement boundary-scan at system level. In order to get a minimum overhead of chains, the ScanBridge of National Semiconductor or the ScanPath Linker and/or the Addressable Scan Port (ASP) of Texas Instruments are often used. System test execution can also be performed from an external tester or from internal system logic as an embedded test. Several solutions are already available in the market to support these applications.



To increase the reliability of electronic systems early life failures can be weeded out by burn-in. At higher temperatures the PCBs are operated in order to accelerate the infant mortality period. During this period early failures should show-up. Common practice is for faults induced during the burn-in to be detected at room temperature



using production test equipment such as In-Circuit Testers. Unfortunately this test will not always reproduce the same fault as detected during accelerated burn-in. The explanation is simple: when cooling down the PCB opens in solder joints will make a connection again at lower temperature or even will be forced to a connection by the bed-of-nails fixture. With the use of boundary-scan the effectiveness of the burn-in can be drastically enhanced by preventing these situations: even at elevated temperature it is possible, due to the simple four

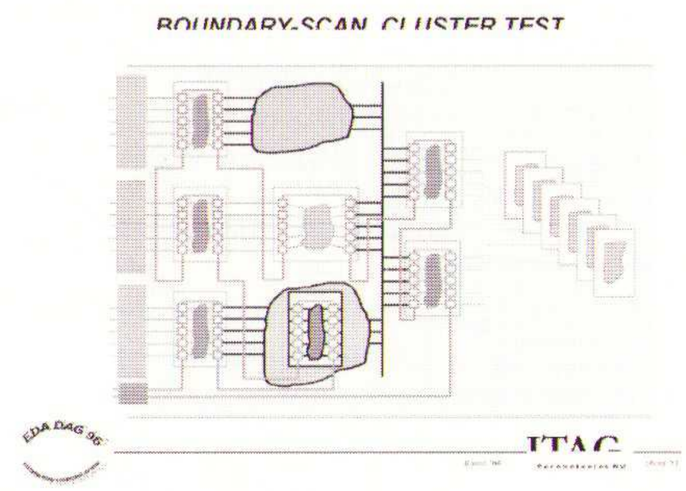


wire (plus power supply) interface, to run a test and to perform full diagnostics!
One of the latest, but not the least (and certainly not the last !) new application of boundary-scan is In-Circuit or In-System programming of PLD's via the boundary-scan chain. The advantages are numerous:
- in engineering easy reprogramming of the PCBs during firmware development and verification
- in the production less logistic overhead and handling
- flexible customizing of the products at the latest possible stage in the production
- testing and device programming as one action using the same (test) equipment via the same connector.

4.9 Test Application Sequence Example

Testing your tester
The first step in the boundary-scan test process is the testing the infrastructure of the boundary-scan chain(s). Clearly before you can conduct any other test on your Printed Circuit Board, you should ensure that the boundary-scan path is working correctly. In the figure you see an example of a boundary-scan infrastructure that can be subjected to testing.

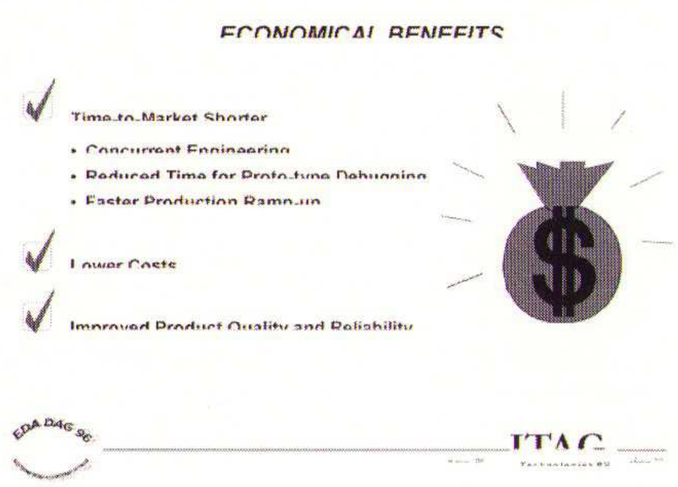
fact, the boundary-scan chain(s) can be seen



as an extension of your tester. For this reason the boundary-scan infrastructure test is always recommended to be performed first.

Interconnection Testing
The next step in the boundary-scan test process is the test for all boundary-scan testable interconnections (nets) of your Printed Circuit Board.
A net is considered to be Boundary-Scan testable if the net can be accessed by boundary-scan cells of devices on the board and/or by the parallel connector pins of the board. With modules like general purpose I/O modules, parallel access to the connector pins of a board is realized.
In this interconnection test a wide variety of net terminators such as drivers, sensors, tri-state outputs, bi-directional pins, differential nets and parallel I/O can be covered.

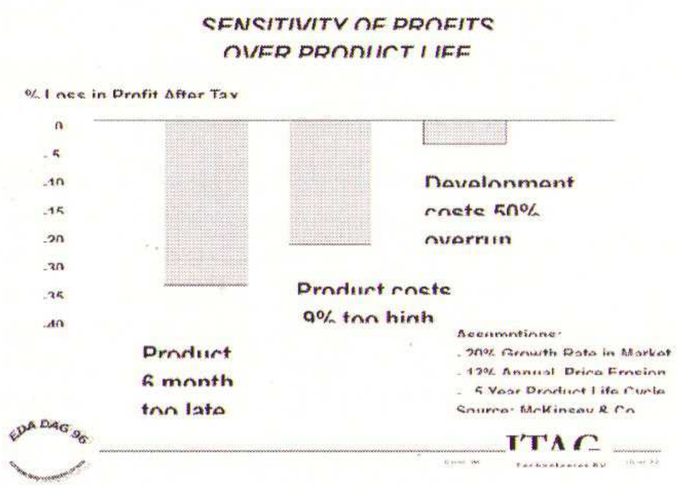
Giving Test Access to Clusters
Until recently the boundary-scan chains on a PCB or MCM could only be used to test interconnections between digital components equipped with boundary-scan cells. However, the boundary-scan chain(s) in a design can be utilized for far more than just interconnect test applications. By re-using existing boundary-scan chains in your design, test access to a group of non-boundary-scannable parts (so-called clusters) can easily be provided. In the figure you will find examples of a Printed Circuit Board with different types of clusters. In this way testing for typical manufacturing faults such as bad solder joints, solder bridges or defective components



on PCBs or defective wire bondings on MCMs inside a cluster is becoming feasible. Typical clusters could be static or dynamic memory devices, FIFO's, PLD's or random logic.

The presence of just one boundary-scan component (e.g. a microprocessor) on your board providing access to those devices, is already enough to make a memory test possible. All this for testing random access memories, for testing clusters of components or for verifying the function of PLDs during prototype debugging or manufacturing.

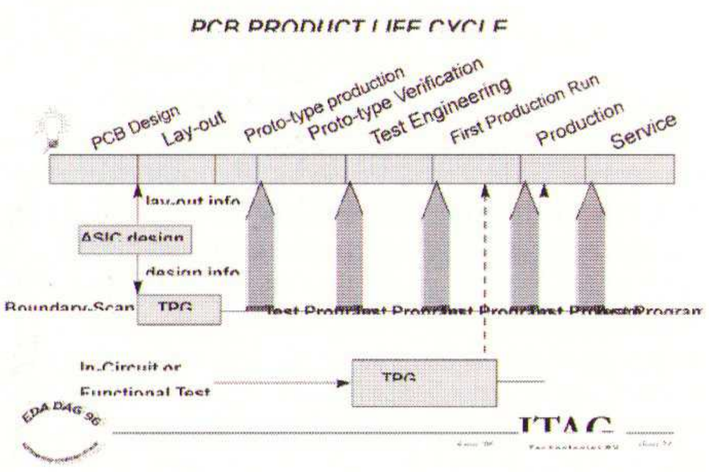
Special Cluster type: Memory Testing
For specific clusters consisting of static or dynamic memory devices, or FIFO's, the tedious task of manually creating functional tests and the associated design specific diagnostics routines has become a thing of the past with JTAG Technologies advanced Test Pattern/Program Generation (TPG) software. All test vectors for memory clusters can be generated automatically. All possible manufacturing defects such as bad solder joints and solder bridges for PCBs will be detected using these test patterns and all address lines, data lines, and control lines



in such RAM clusters are thoroughly tested. In addition a fault dictionary can be generated containing all the necessary information for full diagnostics.

4.10 Commercial Benefits

a) Time-to-market is shorter
The impact of time-to-market has been most

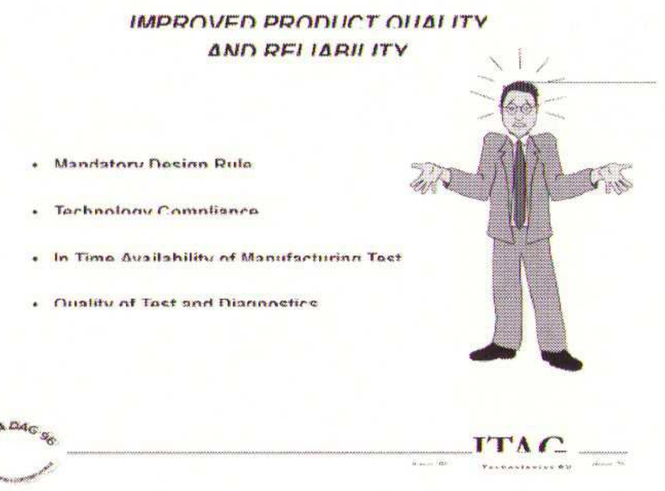


recognized after the publications of the Mackinsey study and other studies. These publications reveal that on-average there is a reduction of 33% in the after-tax profit when products are shipped six months late as compared to a 3.5% reduction of the same profit when the product development expenses are overspent by 50%. Also, the faster a product is introduced into a competitive market, the larger potential life time it will have and hence the greater its return on investments is. Boundary-scan improves, in various ways, the time-to-market of a product:

Concurrent Engineering
The introduction of Boundary-Scan Test (BST) favours the introduction of concurrent engineering. It should be made clear that concurrent

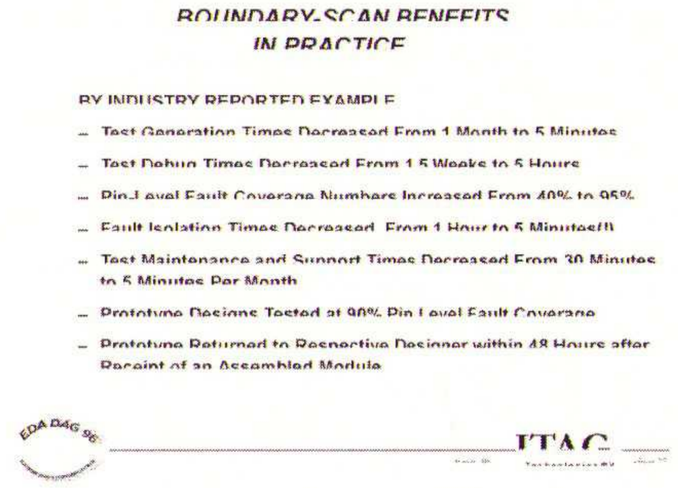
engineering can be introduced without BST, but BST further increases the advantage. In any case, the main reason to introduce concurrent engineering is, of course, to get the right product, at the right time, for the right price, onto the market.

For concurrent engineering a project team is put together with engineers from various disciplines and such a team is continuously supported by the management. Since the team members meet at regular, say weekly, intervals the internal communication will prevent redesigns due to design errors or due to insufficient design verification. As a result, large cost savings are made during production, earning back many fold the extra time spent in the design phase. If a redesign is unavoidable, then it should be done



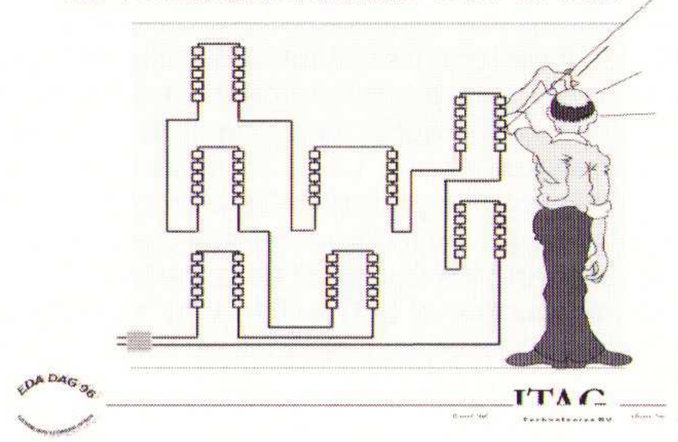
so as to fit the design into the current manufacturing process rather than adjust this process to the new design. In this way unnecessary costs are avoided. Finally, designers are encouraged to cooperate in the new product or project right from the beginning, when the proposals are put forward by the marketing department. In fact, this also gives quality improvements. Due to the encouragement, the designer may be more inclined to added more value to the product.

It should be noted that BST inherently supports concurrent engineering since the designer is



concerned with the whole product life cycle. Furthermore it has been concluded that concurrent engineering shortens the development times and that BST does so even further, through which the shortest possible time to market is obtained.

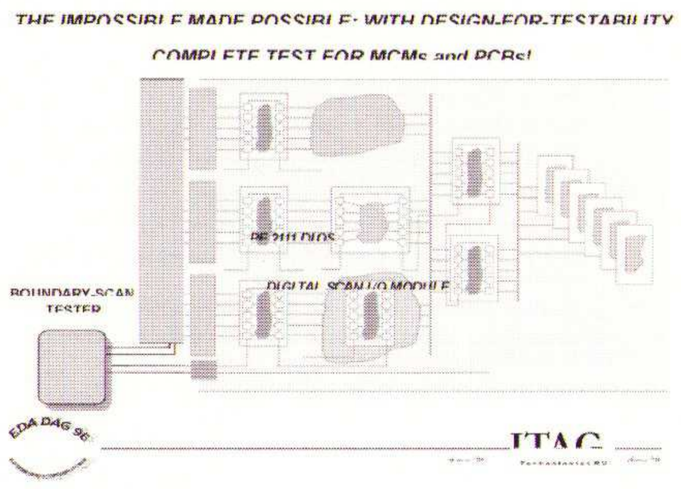
Reduced Time for Prototype Debugging
The types of faults the designer is confronted with during debug are design faults (obviously), and also manufacturing related such shorts and opens. However, in supporting the designer, the investments in specialized or dedicated test equipment are usually kept as low as possible because when the design is ready, part of the equipment becomes obsolete and remains unused. Moreover, the designer can only perform functional tests that are mostly ineffective due to poor fault coverage.



The solution is BST. As shown in the above figure the test programs for manufacturing faults can be easily ready in time to help the designer to debug the PCB prototype. This is especially beneficial in the design of large systems where considerable numbers of prototypes are required for the further development of the system software or the hardware/software integration. This reduction of this critical path length contributes greatly in meeting the time-to-market objectives.

Reduced Production Ramp-up
It frequently occurs that a commercial plan cannot be kept due to start-up problems in the factory. The assistance of the designer is then requested, which is inefficient in at least two ways:
i) the dedicated test equipment of the designer is not suited to support a production line efficiently and
ii) the designer does not like to do work for which he is not trained and certainly not if that work is below his skills.

Moreover, such a working method does not guarantee high quality and it leads to overspent budgets. Once again, the figure on page 18 demonstrates the value of applying BST as the availability of BST test patterns before prototyping starts ensures that prototype production and test preparation are completed



well within time.

b) Lower costs
Lower Capital Investments
It has become clear that the introduction of Boundary-scan Test technology also implies the application of DFT and concurrent engineering. As a consequence the efforts in the design phase have to be increased. Computer power for the software developers also requires considerable investment, in addition to the personnel costs. In order to obtain the highest added value from the designers, they should be equipped with CAD/CAE workstations and supporting software packages.
These computer costs may easily be millions of Dollars, but in any case they are needed for design activities. The point is to exploit these tools to their full extent and not just for the design project. These tools should support both DFT for ICs and/or PCB designs (preferably automated) and test preparation activities for which the automatic inputs of the CAD/CAE stations are very important for fault free test generation. For instance when the boundary-scan test pattern generation SW tools are resident in the design environment then schematics input or the netlist files required (in EDIF) can immediately be used to investigate the testability and to get an first test program. This will also avoid the tedious manual data entries, thus assuring the quality of data transfer.
The introduction of BST will further reduce the investments for testing, particularly in the manufacturing phase of the product life cycle.

This reduction has two causes:
i) The price of the BST tester is much lower than that of a traditional ICT tester: The prices come down from hundreds to tens of thousands of Dollars. One of the reasons for this is the number of test pins (nails) required : a multiple of 5 (for example 25 in a five TAP tester) is used in the BST case and thousands in the ICT case. What's more, since BST testers are highly SW-intensive, they are liable to future price erosions.
ii) For BST less testers are needed due to shorter fault diagnosis times, through which the factory throughput per tester is increased.

due to higher fault coverage achieved during the various test phases.

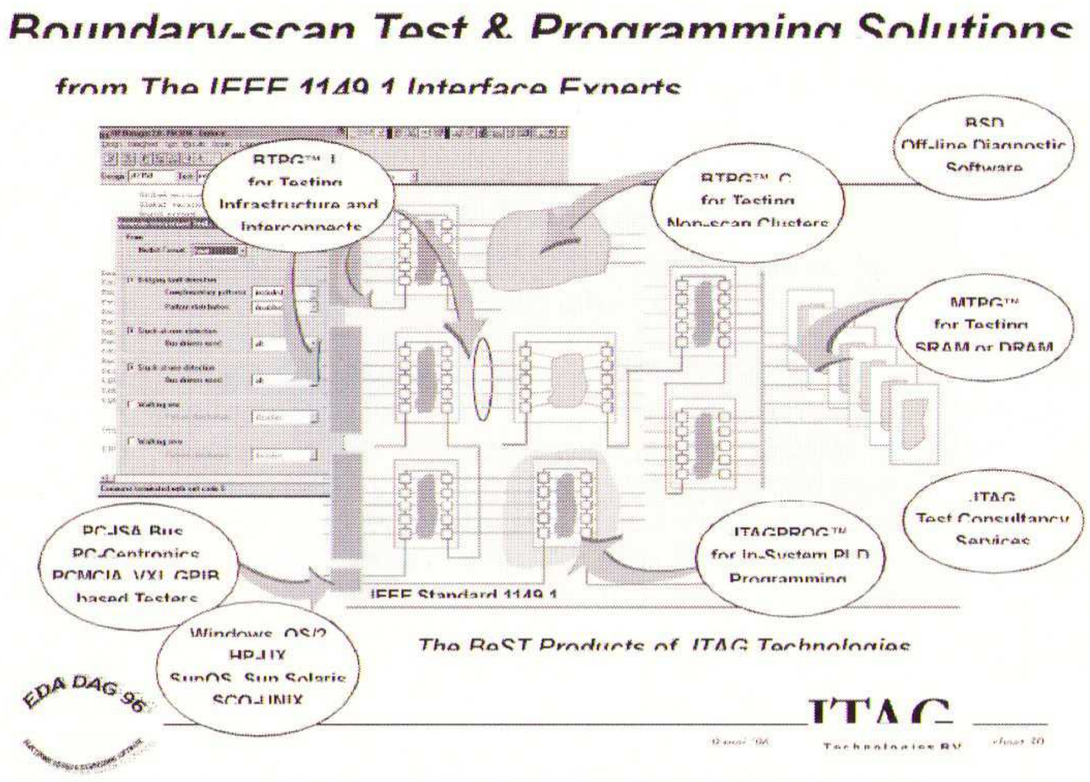
Technology compliance
The products i.e. the Device-Under-Test (DUT) and the test equipment used are by definition in the same technology, because BST is incorporated in the logic of the functional design. There is never a mismatch or divergence of technologies as can happen in the case of bed-of-nails systems where a conflict has occurred between fixture engineering and the shrinking device sizes and novel packaging technologies. These conflicts can clearly influence the quality of testing and consequently the product quality.

In time availability manufacturing tests
There is no ad-hoc testing in the starting-up phase of manufacturing, because with BST test preparation lead times are short and in time due to Concurrent engineering possibilities.
With Functional or In-Circuit Test techniques programs are often not available in time or not at the right level with respect to fault coverage. The impact of these realities on a customer's expectations and your brandname and service costs can be disastrous for the long-term growth of your company!

Quality of test and diagnostics
The very high fault coverage of the BST test and the high degree of diagnostic capabilities lowers the slip through rate of faults during the manufacturing phase resulting in a better product quality and reliability.

4.11 Does Boundary-scan Make Sense to You?

If you don't currently have a problem with testing or programming either from a feasibility or a cost point of view, then please regard the contents of this booklet merely as informative.
However, if some of the problems encountered during testing or programming sound familiar to you and you are looking for a solution, boundary-scan will make sense to you when one or more of the following points apply to your situation:



a) you do have one or more devices in the design available with boundary-scan e.g.
- your own ASIC(s) designed with boundary-scan
-complex PLD's such as AMD MACH 355/445/465, Altera EFX740/780 /880 /8160, MAX7000/9000 series, Xilinx XC4000, XC5200, XC8100, or XC9500, or Lattice isP's
- complex (32-64 bit) processor chips such as Motorola's PowerPC or 68k, or IBM's PowerPC, Intel's 386EX, 486, Pentium (Pro), or Texas Instruments' or Analog Devices' or AT&T's DSP's
-telecomm/datacomm devices such as for ATM from Fujitsu or Brooktree, or as for Ethernet from National Semiconductor or AMD
b) you want to perform In-Circuit or In-System Programming of (C)PLD's or FLASH memory devices in your factory and you want to simplify logistics around the programmable devices
c) you do have problems with physical access (fixtures) with ICT due to fine pitch Surface Mounted Devices or other complex package types
d)you are only using functional testing because the ICT-methodology is too expensive for the (low) number of boards in your activity
e) your costs and development time for fixtures and In-Circuit Testing programs are becoming outrageously high
f) your HW designers hate to spent (too) much time on debugging prototype boards due to production faults
g) your engineers don't have time to spent on designing functional tests with good coverage for production faults
h) you want to re-use the efforts in time and investments spent on testing during the prototype debugging phase for manufacturing testing and for service testing
i)your activity is:
- a small and medium size electronics company
- a (small) department or activity in a large company such as prototype production or a project organization - an engineering/instrumentation department of Institute/University \
- an engineering department of non-electronics (production) company
j) Time_To_Market of your products is vital for your company
k) meeting (international) project deadlines / milestones is very important for your organization.
l) you do have too many scrap boards in your production: boards that can not be repaired due to lack of diagnostics
m)required quality of your products don't allow iterating repair actions.

If one or more of above points are valid for your situation you might consider boundary-scan as an applicable solution for your problems. With just one boundary-scannable device on your board you may already be able to circumvent many of above mentioned problems and easily achieve test or programming applications.

Besides the boundary-scannable devices also static or dynamic memories can be tested, programmable devices such as PLD or FlashProms can be in-system programmed via the chain! For smaller companies or institutes or company department that cannot afford the capital investment of an In-Circuit Tester and still are struggling with functional test, boundary-scan offers a low cost, (typical a factor of 10 lower) test solution for finding manufacturing faults. Also as a test technique suitable for field service, boundary-scan is opening new horizons. Time to market is nowadays becoming more and more important because product life cycles are decreasing continually, even to below one year. If time to market is important to you, boundary-scan will help. Due to its very short test preparation time fixtureless test access and early availability it contributes in the reduction of the prototype debugging phase and is always available in time for production testing.

c) Improved product quality and reliability
Mandatory design rule
Making Boundary-scan mandatory as a design rule to enhance the Design-For-Testability means that the designers have to think beforehand about the problem. The experience teaches us that innovative solutions are created, which would have been impossible if these had to be built-in afterwards. These phenomena will contribute to the improvement of the quality and reliability of the electronic products and systems

4.12 For more information

1) IEEE Standard 1149.1-1990, Test Access Port and Boundary-Scan Architecture, revision I E E E Standard 1149.1a-1993 and Supplement IEEE Standard 1149.1b-1994, published by the Institute of Electrical and Electronics Engineers, Inc., 345 East 47th Streèt, New York, NY 10017, USA.
2) Boundary-Scan Test - A Practical Approach, Bleeker, van den Eijnden, de Jong, published by Kluwer Academic Publishers, P.O.Box 17, 3300 AA Dordrecht, The Netherlands.

4.13 Glossary of Abbreviations

ASIC	Application Specific IC
ASP	Addressable Scan Port
ATM	Asynchronous Transfer Mode
BGA	Ball Grid Array
BSDL	Boundary-Scan Description Language
BST	Boundary-Scan Test
CAD	Computer Aided Design
CAE	Computer Aided Engineering
COB	Chip On Board
DFT	Design For Testability
DIL	Dual In Line
DSP	Digital Signal Processor
DUT	Device Under Test
EDIF	Electronic Design Interchange Format
FIFO	First In First Out
IC	Integrated Circuit
ICT	In Circuit Test(ing)
IEEE	Institute of Electrical and Electronics Engineers
ISDN	Integrated Services Digital Network
JTAG	Joint Test Action Group
MCM	Multi Chip Module
MTBF	Mean Time Between Failures
PCB	Printed Circuit Board
PLD	Programmable Logic Devices
PTH	Plated Trough Hole
QFP	Quad Flat Pack
RAM	Random Access memory
SMT	Surface Mount Technology
TAB	Tape Automated Bonding
TAP	Test Access Port
TCK	Test Clock
TDI	Test Data Input
TDO	Test Data Output
TMS	Test Mode Select
TPG	Test Pattern (or Program) Generation
TRST	Test Reset
VIP	Vector Interface Package
VLSI	Very Large Scale Integration

Testimonial
For Parsytec GmbH Boundary-Scan did make sense: "Since December 1994 we have been working in manufacturing using boundary-scan testers from JTAG Technologies B.V. With the JTAG Technologies' test system, prototype tests for Printed Circuit Board <type> are performed. This has been very simple to implement and has been especially easy to find faults on complex PCBs.
Here is an overview of the results recorded so far:
total of detected faults:.....61
of which
simple faults in memories..... 27
miscellaneous faults.....34
The identification of the 34 miscellaneous faults would have required a considerable effort in man-hours without boundary-scan test facilities. By use of boundary-scan 29 of these faults (i.e. 85%) were detected and removed quickly and efficiently. The remaining 5 faults could be traced back as failing clocks and other easy to find faults.
It must also be mentioned that without boundary-scan some faults couldn't feasibly be detected because of the complex relationship between the PowerPC signals. Using conventional techniques these signals could have only be measured adequately with extremely large [test preparation] efforts.
Also JTAG Technologies' programming software (JTAGPROG) for programming AMD MACH 445 has used. First demonstrations showed that using this software a considerable speed increase in the programming could be achieved. At that time the programming times with the AMD software took about 50 seconds per device (e.g. a board with seven AMD MACH 445 would need 6 minutes). With the software from JTAG Technologies it has been completed within one minute. With further special hardware (VectorBlaster) this programming time can still be drastically reduced."
Parsytec GmbH Germany, May 1995

JTAG Technologies B.V.
Author: Mr. H. Bleeker
P.O. Box 1542,
5602 BM Eindhoven, The Netherlands
fax: +31 40 246 84 71
tel: +31 40 295 08 70, or +31 40 243 32 92
internet: info@jtag.nl

EMC Advisor

Een integrale aanpak van EMC tijdens het routing en layout process

EMC (Electro-Magnetic Compatibility) wordt vaak door ontwerpers nog opgevat als een extra behuizing of een aantal extra onderdelen die moeten worden toegevoegd om het apparaat of systeem aan de gestelde EMC eisen. Dit heeft tot gevolg dat de diegene die het budget van het project bewaken de volgende misvattingen ontstaan:

* Het kost alleen maar meer geld.
* In het gunstigste geval blijft de werking van het apparaat het zelfde.
* De voortgang van het projekt wordt door onduidelijke redenen vertraagd.

Deze misvattingen lijken in eerste instantie terecht maar er zijn ook een aantal voordelen die spreken vóór electromagnetische compatibiliteit van electronische apparaten. Minder ongedefinieerde storingen van apparaten die in gebruik zijn,
dus:
* Lagere service kosten.
* Minder uitval in productie
* Minder redesigns.

Het is belangrijk de EMC problematiek zo vroeg mogelijk aan te pakken, in verband met het feit dat de kosten toenemen en de mogelijkheden tot correctie afnemen naarmate het ontwerp vordert.

5.2 Deskundig advies met een druk op de knop.

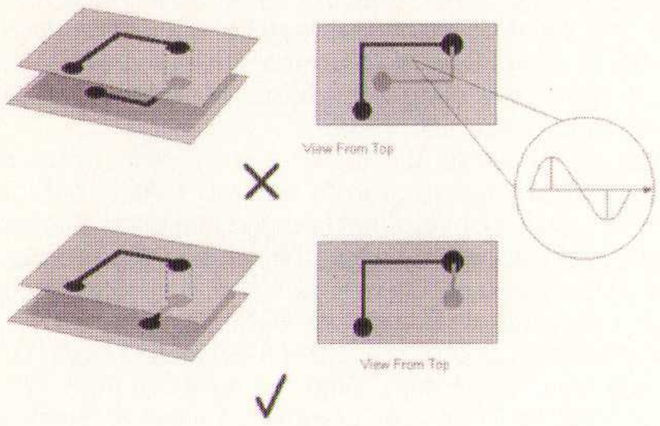
Met EMC Advisor voor Cadstar komt Zuken Redac tegemoet aan de vraag voor verbeterde Electromagnetische comptabiliteit voor printed circuit boards. De EMC Advisor helpt de pcb ontwerper om van te voren te bepalen of een ontwerp voldoet aan EMC eisen. EMC Advisor bestaat uit een flexibele en complete set van regels waarmee het ontwerp gecontroleerd kan worden. het geeft PCB ontwerpers ondersteuning en advies over de te nemen maatregelen tijden het lay-out proces. De advisor laat het resultaat van de analyse zien d.m.v. staafdiagrammen waarna de ontwerper de desbetreffende objecten welke niet aan de gewenste resultaten voldoen kan accentueren. De EMC Advisor bestaat uit een fieldsolver, welke de electromagnetische vergelijkingen oplost en een ruleboek dat de resultaten van de fieldsolver interpreteert en combineert met een aantal praktijk regels. De EMC Advisor werkte geheel geïntegreerd in de Cadstar PCB lay-out omgeving en is al enkele jaren in gebruik op Zuken Redac's Visula Expert systeem.

Bij het ontwerpen van printed circuit boards kunnen we een aantal regels hanteren die het ontwerp minder gevoelig maken voor EMI. Hierbij onderscheiden we een aantal type storingen en regels.

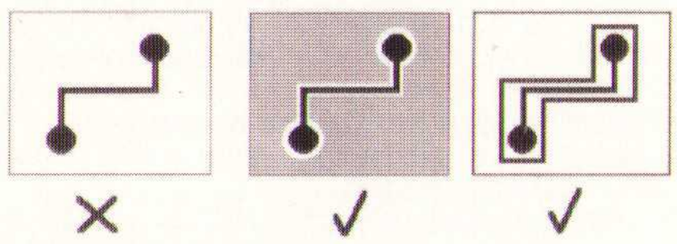
5.3 Differentiële Storingen.

Gesloten lussen.
Een lus van een spoor is in principe een antenne voor EMI welke straling genereert en ontvangt lood recht op het vlak. Deze regel beschouwt alleen lussen welke gevormd worden door het spoor zelf. Gesloten lussen gebruiken altijd meer dan één laag en kruisen een segment van het zelfde net.
Deze regel wordt genegeerd in het geval dat er een voedingsvlak tussen de segmenten ligt. De mate van uitstraling loop kwadratisch op met de frequentie maal het ingesloten oppervlak.

Open lussen.
Een lus van een spoor is in principe een antenne voor EMI welke straling genereert en ontvangt loodrecht op het vlak. Deze regel beschouwd alleen lussen welke gevormd worden door het spoor zelf. Open lussen gebruikenaltijd meer dan een laag en kruizen niet met het zelfde net. De mate van uitstraling is proportioneel met het oppervlak maal de frequentie in het kwadraat Deze regel wordt genegeerd in het geval dat er een voedingsvlak tussen de segmenten ligt

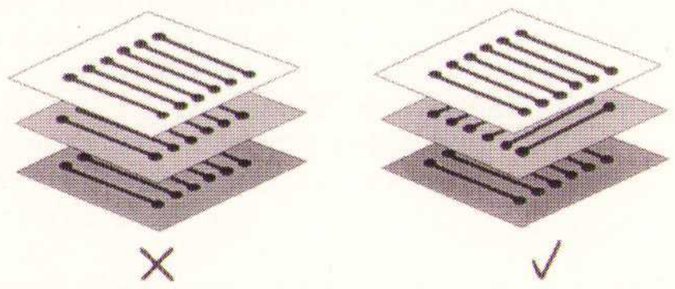


Afscherming van Sporen.
Een afscherming rond een spoor geeft een lokaal retour pad voor EMI. Een scherm rond sporen reduceert ook de karakteristieke impedantie van het spoor.

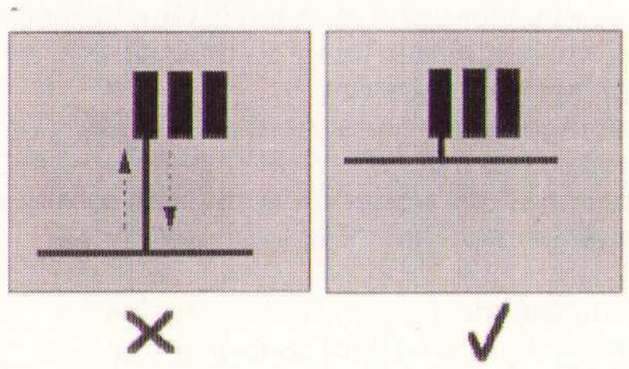


Impedantie profiel.
Gebieden met hogere impedantie geven meer EMI. Het is goed om te proberen de zelfinductie te verlagen en de capaciteit te laten stijgen. Hierdoor neemt de impedantie van het spoor af. Het eenvoudigste is het verhogen van de capaciteit d.m.v. een scherm of grond vlak. Deze regel test sporen waarvan de impedantie hoger is dan het gemiddelde over het board.

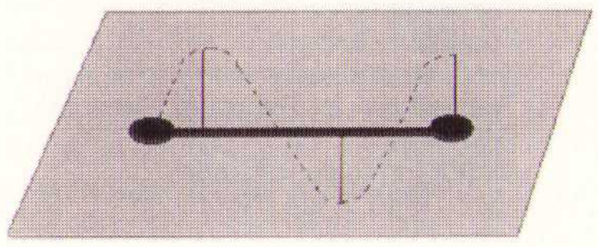
XY richtingen.
Spoor segmenten op naast elkaar liggende lagen moeten haaks op elkaar staan. Dit reduceert de capacitieve koppeling van sporen op elkaar. Deze effekten nemen toe met hogere frequentie en snellere transitie tijden.



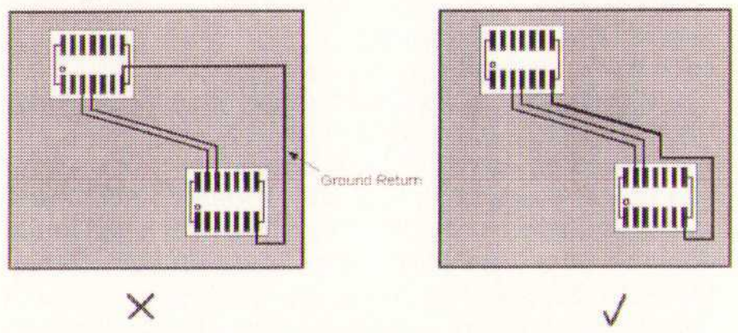
Spoor Stubs.
Stubs (zijtakken van het hoofdspoor) geven ongewenste reflekties als ze een kritische lengte overschrijden. Dit geeft buiten additionele electromagnetische interferentie ook signaal vervorming. De Stubregel gebruikt de fieldsolver om de maximale stubdelay uit te rekenen waarvoor geldt dat deze niet groter dan 10% van de signaal stijgtijd mag zijn.



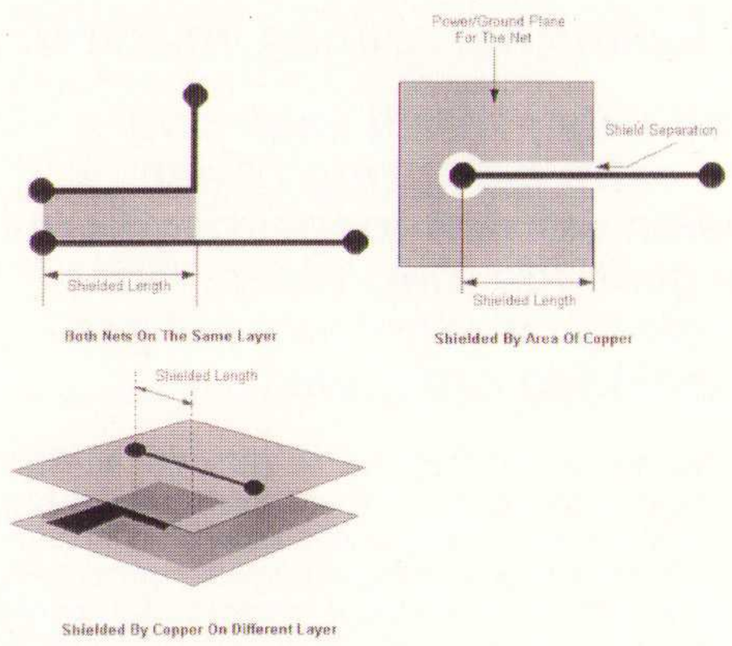
Spoor resonanties.
Als de propagatie vertraging van een spoor in de buurt van een veelvoud van een 1/4 golflengte nadert neemt de efficiëntie als uitstralende antenne toe. Dit geeft een stijging in de electromagnetische uitstraling en maakt het spoor ook meer gevoelig voor instraling.
Spoor resonanties worden berekend aan de hand van de frequentie en de stijgtijd van het signaal om een set van harmonische frequenties te bepalen waarbij de 1/4 golflengte significant is.



Retour sporen.
Retour lussen worden bepaald om zeker te zijn dat een signaal retour pad binnen een minimale afstand van het spoor beschikbaar is. Deze regel is vooral van toepassing op twee laags borden.

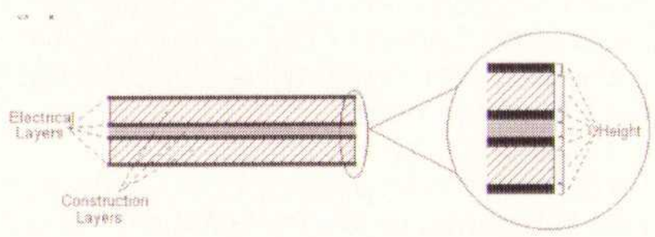


Vanuit EMC oogpunt is het altijd beter een aardvlak op te nemen, waarbij uiteraard de kosten van het board omhoog gaan, maar een betere controle over impedantie en uitstraling kan worden uitgeoefend.
Deze regel detecteert sporen welke geen of te weinig afscherming hebben.



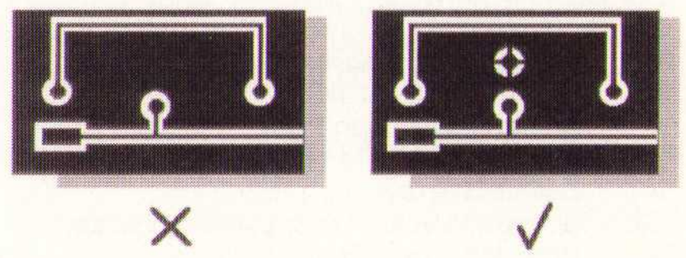
5.4 Common mode Storingen.

Laag opbouw.

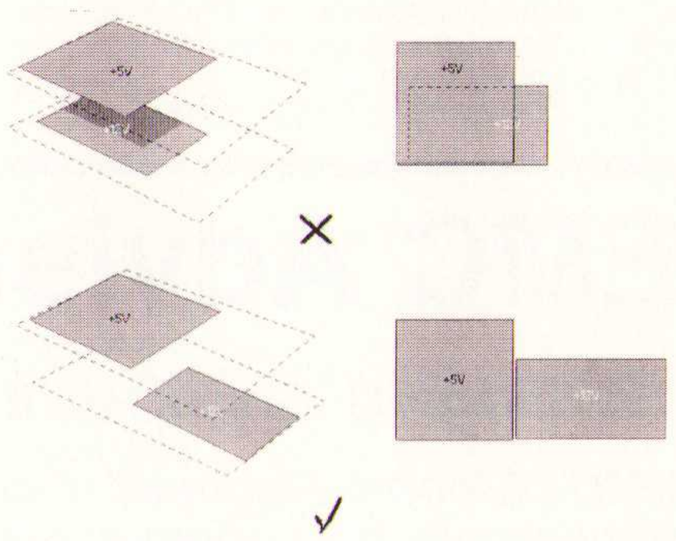


De laag opbouw bepaalt voor een grootdeel de EMC eigenschappen van het bord, waaronder spoorimpedantie e.d. De gebruikte materialen permeabiliteit en dikte van de lagen spelen hierbij een grote rol.

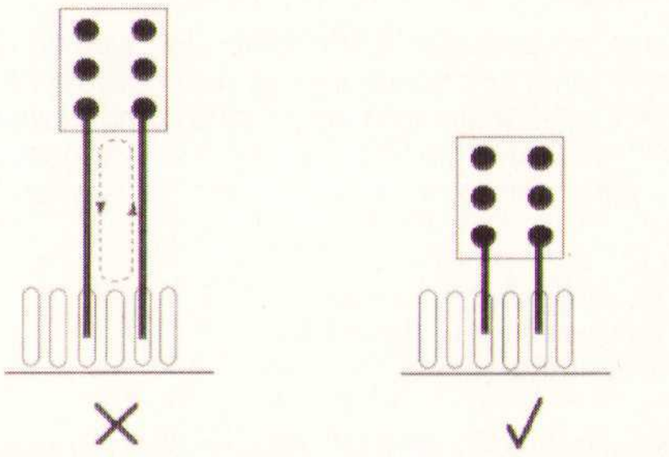
Geïsoleerde gebieden.
Een veel gebruikte methode voor reductie van emissie is het vullen van lege gebieden met kopervlakken hierdoor een effektieve afscherming van de signalen gerealiseerd. Deze kopervlakken kunnen echter als uitstralers gaan fungeren als ze niet aan een voedingsvlak vastzitten. Zwevende vlakken koppelen met sporen doordat ze naast het spoor liggen en kunnen mee resoneren met sporen. Het spoor hoeft zelf niet de correcte dimensies te hebben om een naast liggend vlak dat wel de juiste dimensies (veelvoud van 1/4 golflengte) heeft in resonantie te brengen.



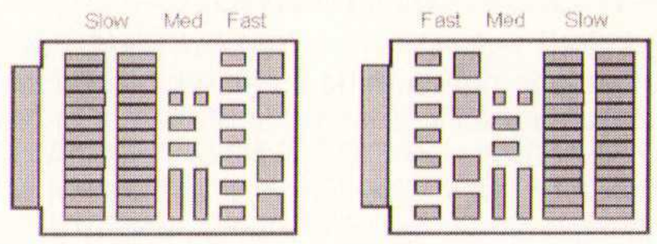
Overlappende vlakken.
Sommige ontwerpen maken gebruik van verschillende voedingsvlakken met verschillende spanningsniveaus. Het is in de praktijk niet gewenst deze vlakken te laten overlappen. Het overlappen van deze vlakken kan leiden tot ongewenste systeemruis. Het overlappen van corresponderende vlakken is echter wel gewenst (bv. +5V en GND) omdat dit de ontkoppelcapaciteit van de vlakken verhoogt.



Component plaatsing.
Snelle componenten moeten zo dicht mogelijk bij de voedingsbron geplaatst worden om transienten te verminderen. Voor borden zonder voedingsvlakken vormen de voedingssporen een signaallus, door snelle componenten zo dicht mogelijk bij de voedingsbron te plaatsen wordt het lusoppervlak kleiner en dus de uitgestraalde energie minder.

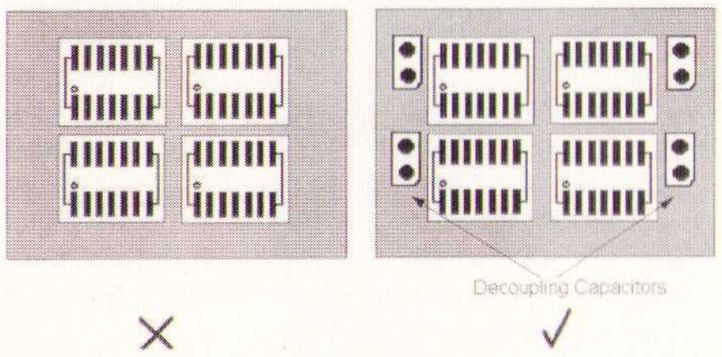


Dit algoritme berekent de ideale afstanden voor ieder component en signaleert componenten die significant van deze plaats afwijken.

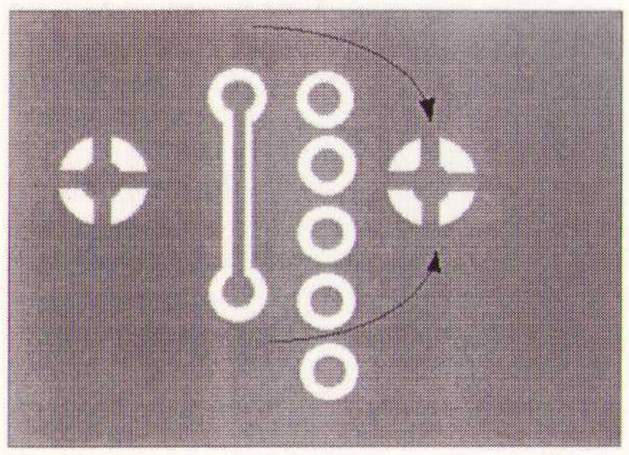


Geïsoleerde gebieden.
Een veel gebruikte methode voor reductie van emissie is het vullen van lege gebieden met kopervlakken hierdoor een effektieve afscherming van de signalen gerealiseerd. Deze

Ontkoppeling.
Goede ontkoppeling van aktieve componenten reduceert component ruis en voedingstransienten. Door voedingstransienten te reduceren wordt de emissie van de signaallus welke door de voedingslijnen wordt gevormd gereduceerd. Alhoewel de keuze van het type ontkoppeling grotendeels door het circuit bepaald worden is de technologie van het van de componenten de belangrijkste factor voor de keuze van het type ontkoppelcondensator.



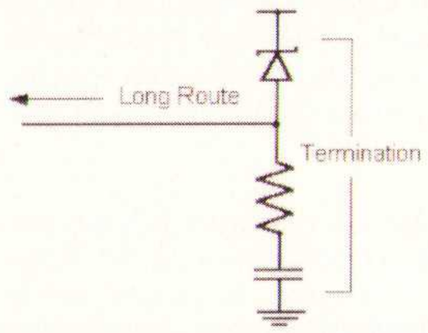
Impedantie van voedingsvlakken.
Een voedingsvlak kan overmatig geperforeerd worden door componentpennen, doormetaliseringen en thermal reliefs. Dit heeft het effect dat de impedantie van het vlak in dit gebied toeneemt, hetgeen er toe leidt dat retourstromen niet meer hun ideale weg zullen volgen welke een toename van de EMI tot gevolg heeft.



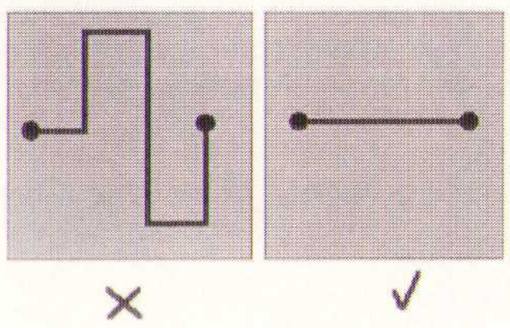
5.5 Snelle Logica.

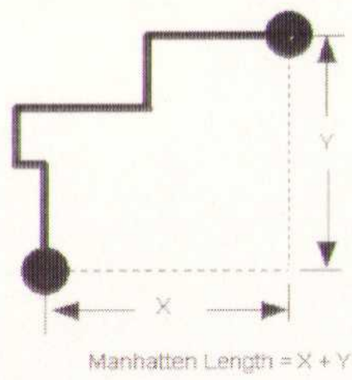
Afsluiting.
Elk spoor heeft een kritieke maximale lengte waarboven reflecties meer effect hebben. Dit gebeurt als de reflectietijd van het spoor de stijgtijd van het signaal nadert. Het is beter om sporen een afsluiting te geven als dit gebeurt. Dit algoritme maakt gebruik van de fieldsolver om de propagatievertraging te berekenen.

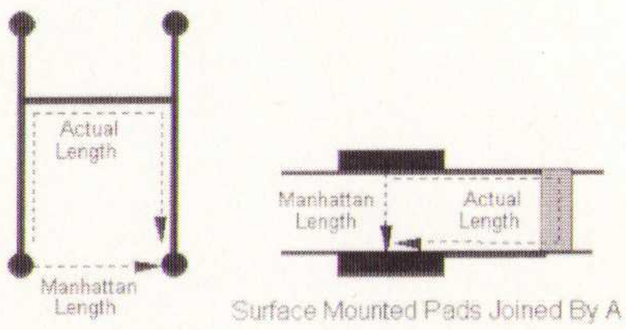
Spoorlengte.
Sporen welke snelle signalen vervoeren moeten zo kort mogelijk gehouden worden. Dit algoritme berekent de verhouding tussen de ideale (manhattan) afstand en de werkelijke spoorlengte en signaleert sporen welke significant afwijken van deze lengte.



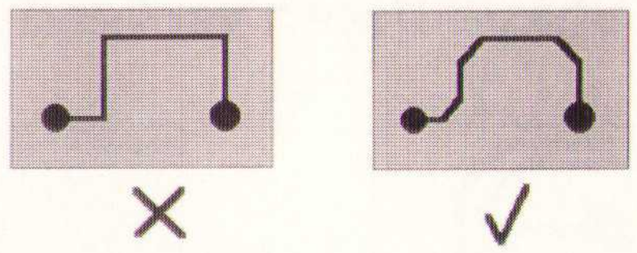
Afschuinen van hoeken.
Electromagnetische emissies zijn speciaal bij snelle signalen meer geconcentreerd op rechte hoeken. Het afschuinen van hoeken helpt het aantal scherpe hoeken te reduceren en geeft daardoor minder emissie.





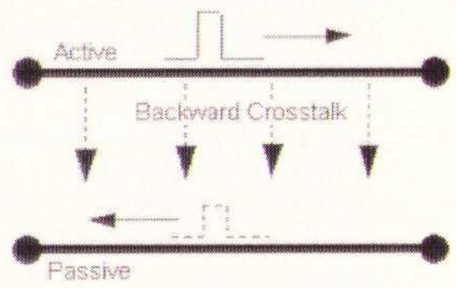


Overspraak.
Overspraak wordt veroorzaakt door parallelle sporen. Een verandering van toestand van een signaal kan interferentie veroorzaken op een ander spoor ten gevolge van wederzijdse inductie of capaciteit. Hierbij zijn twee typen van overspraak mogelijk :



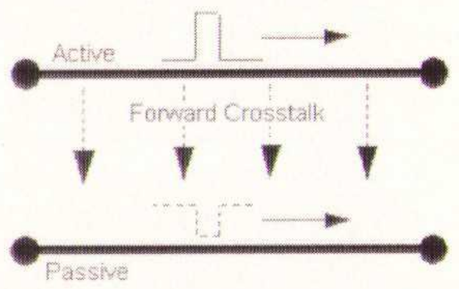
Achterwaartse overspraak
Deze vorm van overspraak is meestal primair. Het geïnduceerde signaal ontstaat op de passieve lijn en loopt in omgekeerde richting over de lijn. Het geïnduceerde signaal heeft altijd de zelfde polariteit als het aktieve signaal.

Voorwaartse overspraak



Dit is een secundair effect welke hoofdzakelijk op de buiten lagen van het bord voorkomt. Bij voorwaartse overspraak loopt het signaal in de zelfde richting als het aktieve signaal en is meestal van tegengestelde polariteit.

Impedantie van sporen.
e impedantie van sporen wordt bepaald door de diameter van het spoor en het omliggende medium. Dit algoritme controleert door middel van de fieldsolver of alle sporen aan de gestelde eisen voldoen. Hiertoe berekent de fieldsolver van alle spoorsegmenten de impedantie.



Stroom.
De stroom die door het ontwerp kan lopen worden bepaald door de spoordiameter van de voedingssporen. Deze sporen moeten voldoende dik zijn om de schakeling de stroom te kunnen geven die het nodig heeft om goed te werken. Dit algoritme bereken de capaciteit van de voedingssporen en vergelijkt dit met de benodigde capaciteit

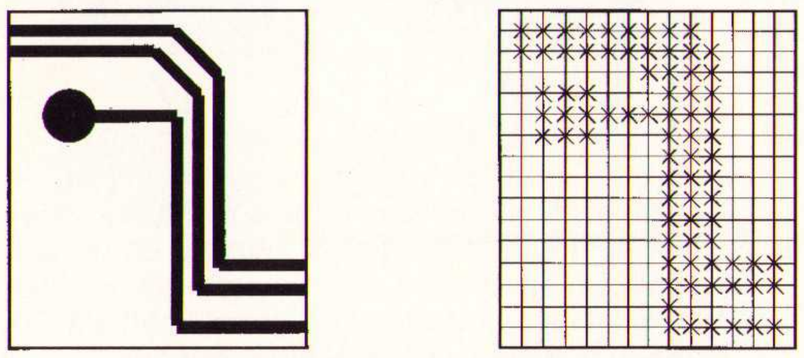
Jos van Heesen
Koning & Hartman Professionele Meet- en Testtechniek
Tel: 0162-480100

Nieuwe ontwikkelingen bij PCB design.

Shape-based databases, rules driven design en integratie.

Door de snelle ontwikkelingen in de hedendaagse ontwerp-methoden en implementatietechnieken zijn ook de makers van CAD systemen gedwongen tot innovatie. In dit korte artikel willen we graag enkele nieuwe trends bespreken en de toepassing hiervan in een CAD systeem voor PCB (Printed Circuit Board) Layout.

Shape-Based vs. Grid Map



Aan de hedendaagse produkten worden steeds meer en steeds zwaardere eisen gesteld. Niet alleen moet alles steeds kleiner, sneller en goedkoper maar er worden ook stringente eisen gesteld aan bijvoorbeeld het EMCgedrag. Alsof dit nog niet genoeg is moet de ontwikkeltijd ook steeds korter zijn. Deze eisen hebben geleid tot geheel nieuwe implementatietechnieken op componentennivo (ASIC's, FPGA's) maar ook op het PCB nivo. Hierbij zijn de CAE- en de CAD wereld steeds meer naar elkaar toegegroeid, er moet bij de fysieke implementatie met steeds meer ontwerp-regels rekening worden gehouden die al in de CAE omgeving vastgelegd zijn. Bovendien moet de ontwikkelaar bij de implementatie steeds meer woekeren met de beschikbare ruimte op het PCB en steeds beter rekening houden met bijvoorbeeld transmissielijn- en EMC effecten. De laatste jaren zijn er enkele trends zichtbaar die hier op inhaken en langzaam hun weg naar de ontwerper vinden.

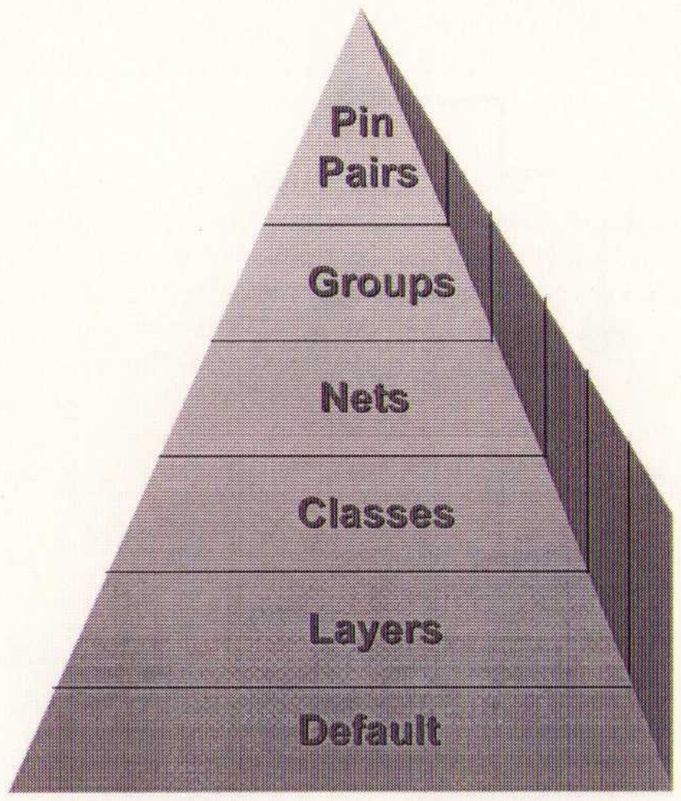
6.2 Shape-Based databases en design.

Als voorlopers in de markt hebben mr. Cooper en Chyan ingezien dat de klassieke database technologie geen oplossingen meer biedt voor de hedendaagse complexe PCB ontwerpen. Zij zijn de grondleggers van het Shape-Based database concept.
Bij het klassieke (grid-based) systeem dat door nog vrijwel alle CAD systemen gebruikt wordt bestaat de database uit een groot 'raster' ook wel de grid-map genoemd. Op elk kruispunt in dit raster kan een database element aanwezig zijn, bijvoorbeeld een via, of een deel van een printspoor. De nadelen van een grid-based database worden het eerst duidelijk bij de toepassing van zowel through-hole als SMD componenten. Doordat deze uitgaan van een verschillend raster (mill's versus metric) is het zeer moeilijk een gemeenschappelijk raster te vinden waar beide componenten op passen. De gebruiker wordt gedwongen om een zeer fijn raster te kiezen dat niet alleen zeer lastig te gebruiken is, maar ook zeer veel geheugen gebruikt.

Als alternatief wordt in een shape-based database de vorm van de complete objecten en hun locatie in de database vastgelegd. In dit model is de Shape het meest primitieve element. Iedere via, pin, segment, ... gebruikt een of meerdere Shapes en iedere Shape krijgt d.m.v. design rules de eigenschappen van het object die hij vertegenwoordigt. Bijvoorbeeld, een net omvat verschillende objecten zoals pinnen, segmenten en via's. Als dit net een bepaalde Clearance Rule heeft, krijgt ieder object van dit net deze Clearance. De Shape-based Database modelleert iedere Shape in zijn meest exacte vorm. Bovendien kan dit model met zeer complexe Design Rules werken. Aan elk object kan een vrijwel onbeperkt aantal design-rules worden gekoppeld, dit maakt een complete hiërarchie van design-rules mogelijk. Ook het gebruik van complexe rules voor bijvoorbeeld overspraak (cross-talk) en gebalanceerde netten (balanced-pairs) past binnen dit concept.

De voordelen van Shape-based Design tonen zich al snel bij de toepassing in Auto- en Dynamische Routers. In plaats van te proberen een route te vinden volgens een grid, zal de Router potentiële routes vergelijken met Shapes die zich in zijn omgeving bevinden. De ruimte om een spoor te leggen tussen twee objecten is alleen gelimiteerd door de afstand tussen de objecten en de clearance samen met de breedte van de baan. Ook bij een Shape-based router worden kostenfactoren vergeleken met als doel de meeste economische oplossing te vinden. Tegelijkertijd worden de bestaande routes continu geëvalueerd en op een strategische manier opnieuw gelegd voor een maximale optimalisatie van het design.

Een ander groot voordeel van Shape-based Design ligt zoals gezegd in een dramatisch gereduceerde hoeveelheid informatie ten opzichte van de Grid-based Designs. Met Shape-based Design, blijft de informatie lineair groeien met het aantal Shapes dat wordt gebruikt in het ontwerp. Dit betekent een optimaal gebruik van het geheugen en van de beschikbare rekenkracht.



PowerPCB Rule Hierarchy

Tot nu toe waren de voordelen van Shape-based design alleen te vinden in auto-routers als bijvoorbeeld de Spectra router van Cooper & Chyan Technology. In 1995 echter heeft Pads Software een PCB-layout systeem geïntroduceerd (PowerPCB) dat geheel opgezet is rond een Shape-based database met alle voordelen die hieruit voortvloeien.

Een extra voordeel is dat het voor de maker van het CAD systeem veel makkelijker is de (object-georiënteerde) ontwerpdatabase uit te breiden met nieuwe functies. Zo werkt men bij Pads software momenteel aan MCM ondersteuning, hoogfrequent ontwerp en EMC simulatie. Uiteraard is ook de koppeling van een Shape-based autorouter met een shape-based layoutpakket een robuustere oplossing dan wanneer alle shape-based informatie weer op een grip ge-'mapped' moet worden.

6.3 Rules Driven Design.

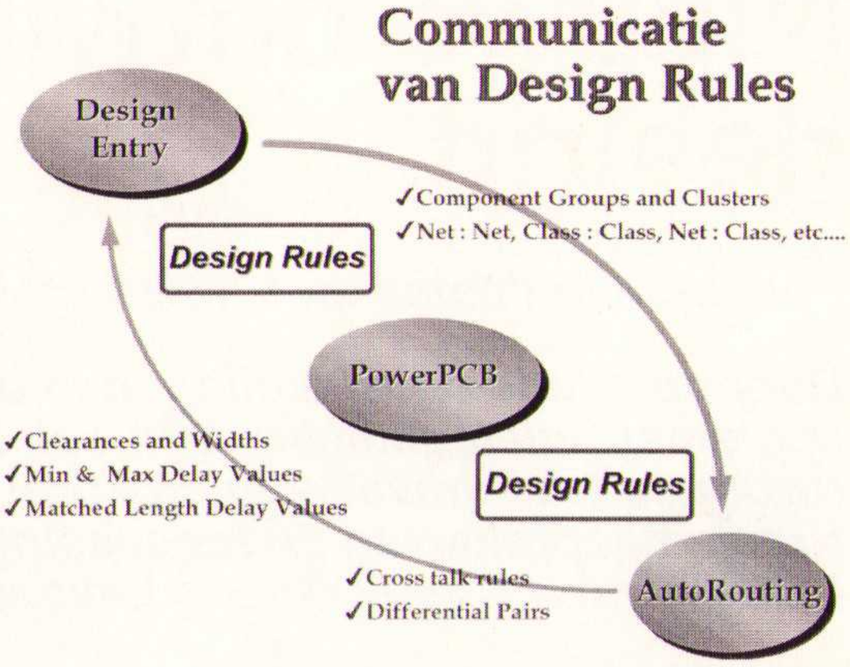
Bij de eisen die gesteld worden aan de hedendaagse ontwerpen is de toepassing van één set design-rules bij lange na niet meer voldoende. Zo wil de ontwerper graag aparte regels kunnen opstellen voor de verschillende deelontwerpen (classes) binnen een systeem, bijvoorbeeld voor de voeding, het analoge deel en het digitale deel. Bovendien is het handig om regels te kunnen koppelen aan bepaalde kritische netten, bijvoorbeeld een voedingsnet dat een hoge stroom voert en dientengevolge een grote dikte moet hebben. Bij de meer geavanceerde ontwerpen zijn er soms regels nodig die forceren dat bepaalde netten een gelijke elektrische lengte hebben, denk hierbij aan een klokdistributie-net, of regels die de EMC eigenschappen beschrijven. Als we al deze regels bij elkaar nemen ontstaat een hele hiërarchie van design-rules die op elk nivo verschillende parameters omvat.

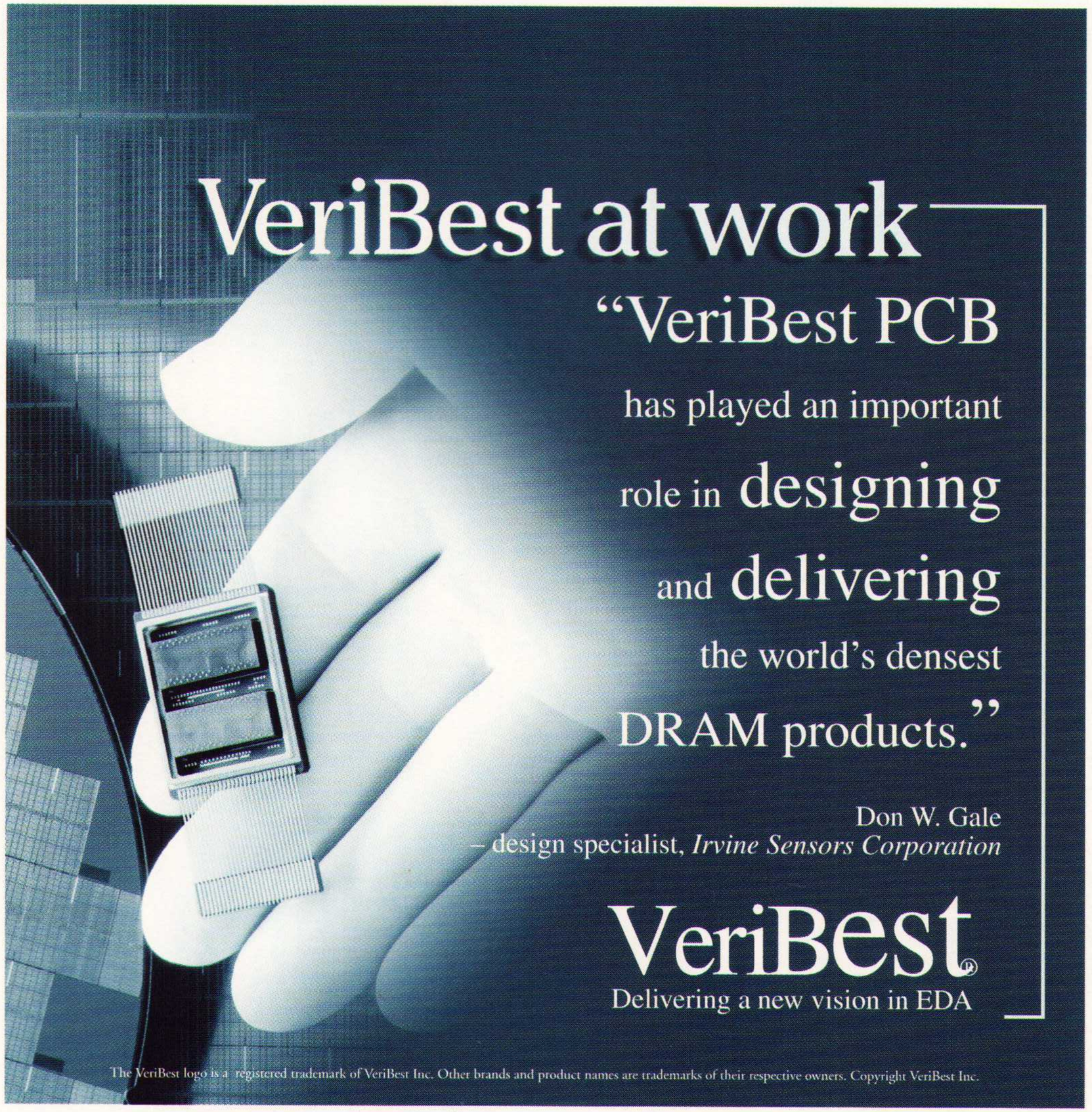
In het geval van het Pads PowerPCB systeem omvat dit op elk nivo regels voor de clearance, lijndikten, autorouting en hoog-frequent regels (cross-talk, max/min delay, route length) . Uiteraard moeten deze regels tijdens het gehele PCB-layout proces toegepast worden. Denk hierbij aan de On-Line Design-Rule Check (DRC), tijdens het handmatig routen, tijdens het autorouten en het automatisch 'gieten' van aardvlakken. De ruleset van het PowerPCB systeem staat tevens toe de complexe regels voor de CCT autorouter in te voeren en door de geven aan deze batch-autorouter wat het grote nadeel van deze krachtige autorouter wegneemt (Veel mogelijkheden zijn namelijk handig, maar zonder goed overzicht raakt men als gebruiker al snel het overzicht kwijt) en de garantie biedt dat gedurende het gehele implementatieproces van de zelfde rule-set gebruik wordt gemaakt.

De eerder genoemde hoogfrequent regels zijn bedoeld om te worden doorgegeven aan de in PowerPCB geïntegreerde EDC (Electro-Dynamic Check) module en aan de EMC en transmissielijnsimulatoren die aan het pakket gekoppeld kunnen worden. Bij de eerstgenoemde verificatie bekijkt het systeem aan de hand van een aantal vuistregels en parameters van het PCB (diëlectrische constante, dikte van de materialen) in combinatie met de designrules of er (en zo ja, waar....) transmissielijn- of overspraakeffecten te verwachten zijn.

6.4 Integratie

Uiteraard is het noodzakelijk dat we de design-rules zo veel mogelijk al in de CAE omgeving (het schema) in kunnen voeren en deze automatisch kunnen meenemen naar het PCB-layout systeem en de eventuele autorouter. Het apart invoeren van de design-rules kost immers niet alleen veel tijd, maar geeft ook aanleiding tot het maken van fouten. De integratie tussen het





Innovation in IC Packaging.
Irvine Sensors Corporation produces the world's densest 3D memory products using a process that stacks multiple ICs. Their products offer memory densities up to 15 times that of a single die, and are intended for use in a variety of military, space, and commercial applications. VeriBest PCB was used to produce their complex PCB designs for test and burn-in, as well as the MCM packaging of their advanced 3D memory products. The combination of bringing together the right people, defining the right process, and using the right tools has played a key role in the success of Irvine Sensors' products.

Put VeriBest to work for you ..
Call: (023) 5666801
E-mail: mmutsaerts@veribest.com
World Wide Web:
http://www.veribest.com

VeriBest International Limited
P.O. Box 333
2130 AH Hoofddorp NL

CAE front-end en de PCB-layout omgeving gaat dan ook verder dan het doorgeven van een zgn. net-list.
Integratie tussen het front-end en de layout omgeving is natuurlijk het eenvoudigst als beide omgevingen van dezelfde fabrikant afkomstig zijn. Dit betekent echter voor de eindgebruiker dat deze zeer beperkt wordt in de flexibiliteit van de ontwikkelomgeving en de vrijheid van zijn keuzen. Bovendien blijkt in de praktijk dat de beste CAE systemen worden geleverd door fabrikanten die geen PCB-layout systeem in het pakket hebben, en dat de fabrikanten van de toonaangevende PCB-layout tools voor de PC geen compleet CAE systeem aan bieden.
Recentelijk zijn de MS-Windows en Windows-NT besturingssystemen echte multi-tasking omgevingen geworden die IPC (Inter-Process Communicatie) mogelijk maken en we zien dan ook dat er hard gewerkt wordt aan een betere integratie tussen de CAE en CAD systemen. Als voorbeeld van wat mogelijk is willen wij de mogelijkheden van de integratie van het VIEWlogic CAE systeem en het Pads PowerPCB pakket noemen. Het PowerPCB systeem is echter ook te integreren met andere CAE front-end's als bijvoorbeeld Synario van Data I/O, OrCAD, Intergraph, MicroSim en Capilano. Een voordeel bij de integratie van de VIEWlogic en de Pads Software produkten was dat beide fabrikanten in het zelfde marktsegment opereren en beiden een duidelijke strategie hebben voor de integratie onder MS-Windows (Windows-95 en -NT).

VIEWlogic levert de Pads PowerPCB interface gebundeld met alle VIEWlogic office pakketten, waarmee niet alleen de 'net-list (interconnects en parts-list) doorgegeven kan worden, maar ook de design-rules en de informatie die nodig is voor het doorvoeren van wijzigingen (Backannotation & forward ECO). Deze design rules omvatten de gehele hiërarchie van rules met daarin de clearance, routing en ook de High-speed design-
rules. Vorige maand is bovendien de nieuwe IPC interface geïntroduceerd die gezamenlijk met VIEWlogic is ontwikkeld waarbij gebruik gemaakt wordt van OLE voor de zgn. Cross-Probing tussen de twee systemen. Hierbij is het mogelijk om bijvoorbeeld in het schema (in VIEWlogic) een component te selecteren, dat vervolgens automatisch in PowerPCB opgepakt kan worden om het een plaats op het PCB te geven. Ook is het mogelijk de om via dit IPC kanaal het doorgeven van de wijzigingen naar het schema (back-annotation) automatisch uit te voeren. Bovendien kunnen de uit de VIEWlogic omgeving afkomstige design-rules direct doorgegeven worden aan de CCT autorouters of de EMC simulatiepakketten van Hyperlynx en Quad Design.

Later dit jaar wil men de tweede fase van de integratie afronden waarbij het doel is om een volledige synchronisatie van de PowerPCB en de VIEWlogic databases te realiseren. Hierbij volgt het systeem de wijzigingen die gemaakt worden (ECO's) en voert deze direct door in de andere omgeving. Hierbij wordt verder ontwikkeld op de reeds ontwikkelde Inter-Process Communicatie via de windows OLE functies.

6.5 Conclusie

Hoewel er aan de ontwikkelaar van complexe systemen steeds zwaardere eisen worden gesteld, zijn er ontwikkelingen gaande die inspelen op zijn behoeften. De ontwikkeling van systemen die top-down design mogelijk maken in een geïntegreerde omgeving en de introductie van nieuwe concepten als de Shape-based database bieden niet alleen oplossingen voor de huidige problemen die de ontwikkelaars tegen komen bij de implementatie van complexe ontwerpen, maar belooft door de vrijwel onbeperkte mogelijkheden van de Design-
rules ook oplossingen voor Design For Manufactoring (DFM) en lastige implementatie technieken als RF-design en Multi-Chip Modulen. Tevens is de integratie tussen standaard front-end's en PCB-layout pakketten op de PC 'volwassen' aan het worden zodat economische alternatieven voor de dure high-end UNIX-systemen nu beschikbaar zijn. Opvallend hierbij is dat grote concurrentiedruk bij de PC systemen geleid heeft tot een sterke innovatie waardoor deze systemen op verschillende gebieden zelfs een voorsprong hebben genomen.

Gerard Fianen en Christophe Dumont.
Air-Parts B.V.
Tel: 0172-443221

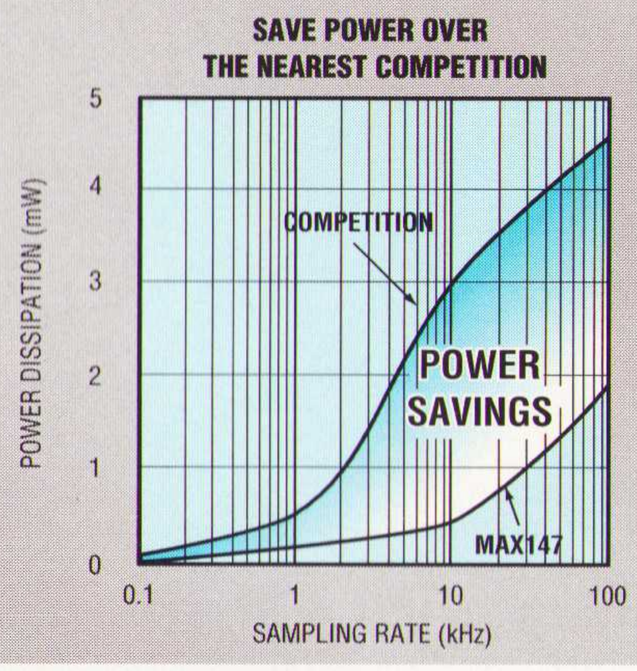
LOWEST POWER 2.7V, 12-BIT ADC EXTENDS BATTERY LIFE 150%

Draws Less than 0.7mA at 100ksps!

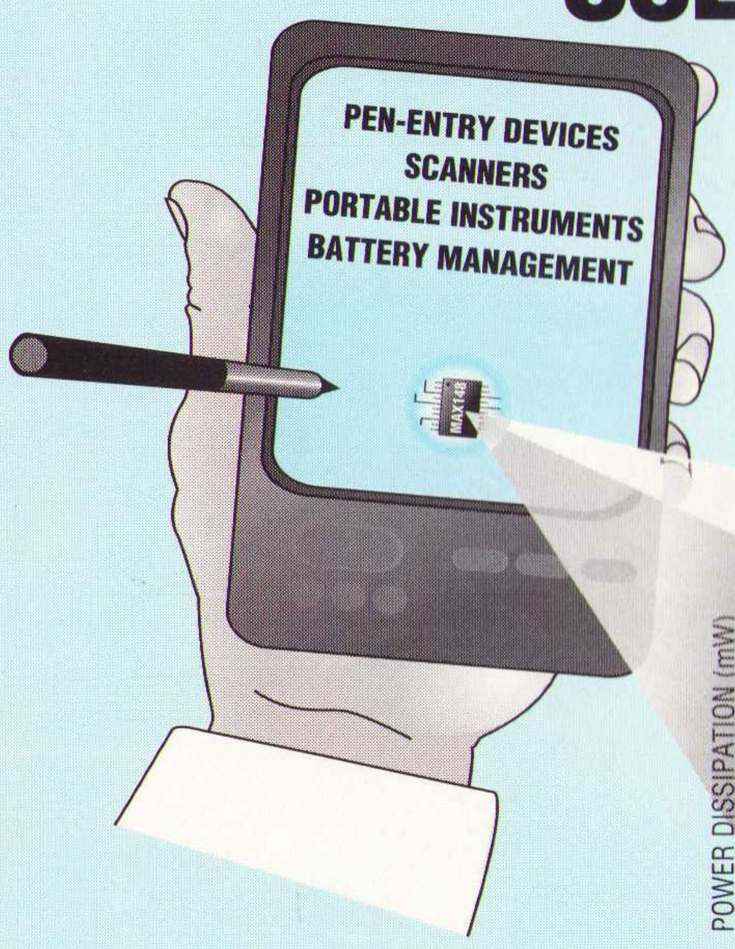
In portable battery-powered applications which demand minimal battery count and maximum battery life, obtaining the required performance using the lowest power is critical. The 12-bit MAX147 works with +2.7V to +5.25V supplies and samples at up to 133ksps while providing the lowest power dissipation of any comparable converter.

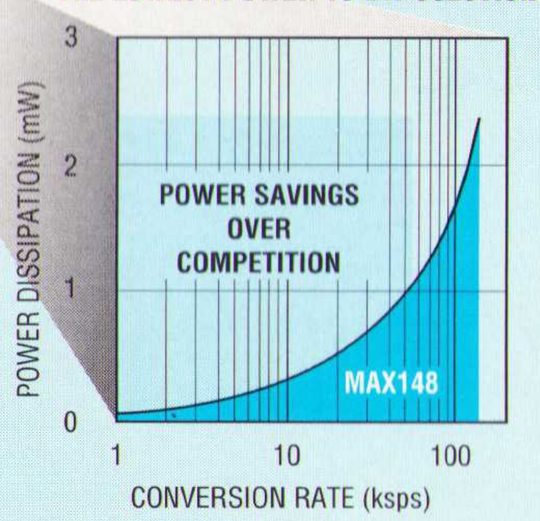
- **Guaranteed 2.7V Operation**
- **1µA Shutdown Mode**
- **Space-Saving 20-Pin SSOP Package**
- **Serial Interface Compatible with SPI™, QSPI™, and Microwire™**
- **8 Single-Ended or 4 Differential Inputs**

SPI and QSPI are trademarks of Motorola. Microwire is a trademark of National Semiconductor.



FIRST 2.7V 10-BIT ADC USES LOWEST POWER

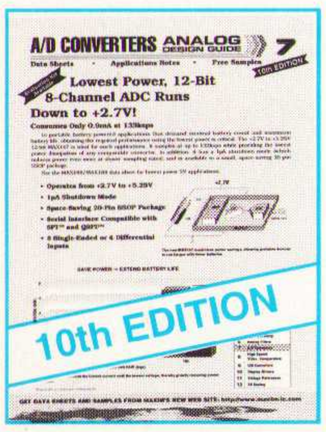




MAX148:

- **+2.7V to +5.25V Operation**
- **Ultra-Low Supply Current:**
  0.9mA @ 133ksps
  100µA @ 10ksps
- **1µA Power-Down Mode**
- **8 Single-Ended/4 Differential Inputs**
- **3-Wire Serial Interface**
- **20-Pin SSOP Package**
- **Pin-Compatible 12-Bit Upgrade (MAX147)**

Gratis A/D Converter Design Guide



Bestel nu de tiende uitgave

Bel 015 - 2 609 906

en wij versturen uw exemplaar binnen 24 uur.

1995 DATAQUEST Best European Niche Vendor

Accredited by the Dutch Council for Certification — ABS Quality Evaluations, Inc. Quality Assurance Certification — ACCREDITED REGISTRAR ANSI-RAB
ISO 9001
Certificate Number: 30337

MAXIM

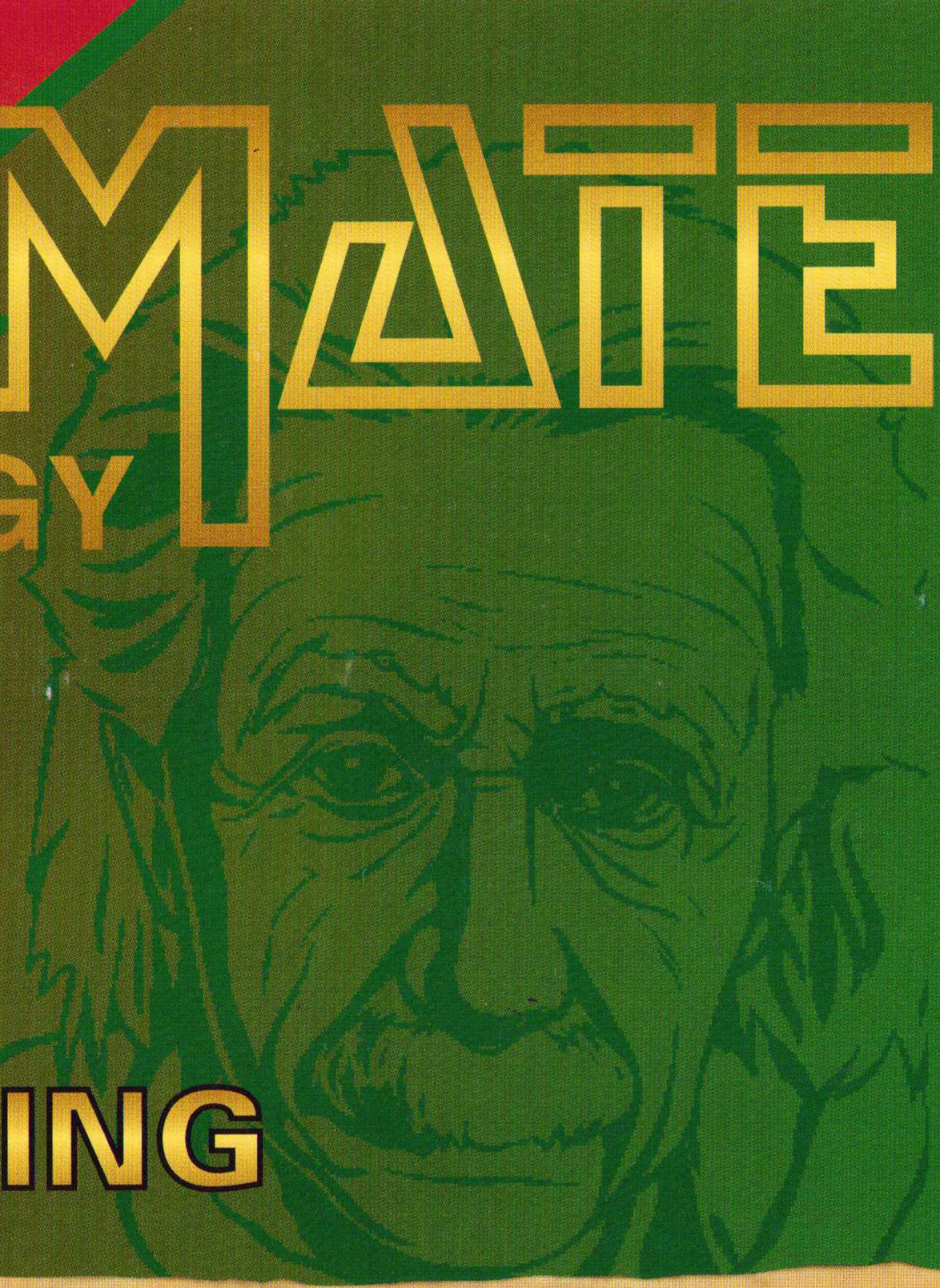
Maxim Integrated Products - U.K.,
phone (01734) 303 388; fax (01734) 305 577

Maxim is een geregistreerd handelsmerk van
Maxim Integrated Products

KONING EN HARTMAN
TELECOMMUNICATIE EN INDUSTRIELE ELEKTRONICA
ENERGIEWEG 1, POSTBUS 125, 2600 AC DELFT, TELEFOON 015-2609906. FAX 015-2619194.

Getronics Group

ULTIMATE TECHNOLOGY



EMC TRAINING

De introduktie van de ULTImate EMC Test Set is een fenomenaal succes gebleken. In de afgelopen maanden is het EMC Competence Centre opgezet, teneinde een optimale support te waarborgen. ULTImate Technology hanteert hierbij als filosofie dat support zich niet mag beperken tot vragen m.b.t. de bediening van de apparatuur: centraal staat het oplossen van een testprobleem. Hiertoe zijn op ULTImate Technology's EMC Competence Centre ervaren specialisten beschikbaar.

Teneinde de behoefte aan training duidelijk in kaart te brengen zijn diepgaande interviews met (potentiële) EMC test set gebruikers. Hieruit bleek dat de benodigde theoretische kennis veelal wel aanwezig is, maar dat de aansluiting naar de praktijk uiterst onduidelijk is. Om deze lancune op te lossen organiseert ULTImate Technology in geheel Europa een sterk op de praktijk gerichte 2-daagse cursus. De doelstelling is dat elke deelnemer na afloop van deze cursus een duidelijk antwoord heeft op de vragen:

1. Welke normen zijn op onze produkten van toepassing?
2. Is pre-compliance testen een alternatief?
3. Welke meetopstellingen zijn nodig voor onze produkten?
4. Afgeschermde ruimte; geleidings- en/of stralingsmethode nodig?

☎ GRATIS
06-022-3444
België: 0800-71937

Deze doelstelling wordt gekonkretiseerd door elke deelnemer één of meerdere cases in te laten brengen, welke in de cursus worden behandeld. Een praktisch cursusboek is bij de prijs van ƒ 925,--/19390 BF (excl. BTW) inbegrepen. De cursus is met name bedoeld voor technici op HBO denk- en werknivo, betrokken bij testen en kwaliteitscontrole. Voorkennis van de normen is niet vereist. Een gedetailleerde agenda is op aanvraag beschikbaar.

2 en 3 mei	Naarden	4 en 5 juni	Drunen	1 en 2 juli	Brussel
14 en 15 mei	Naarden	10 en 11 juni	Brussel	17 en 18 juli	Naarden
21 en 22 mei	Antwerpen	20 en 21 juni	Hoorn		
28 en 29 mei	Naarden	24 en 25 juni	Naarden	*Other countries: please call*	

EMC TEST SET

ULTImate Technology levert voor slechts ƒ 7895,- / 157.900 BF (excl. BTW) een pre-compliance EMC-testset, bestaande uit een 1 GHz software-controlled Spectrum Analyzer met tracking generator, speciale probe, 1 GHz breedbandversterker, LISN/kunstnet (power dummy) en een impulsbegrenzer. Optionele netwerken, ESD test guns en antennes zijn uiterst scherp geprijsd.

Hoofdkantoor: Energiestraat 36 • 1411 AT Naarden • tel. 035-6944444 • fax 035-6943345