HEWLETT® PACKARD
Expanding Possibilities

Rapid Prototyping Methods for Embedded System Development Reduce Time-to-Market

Brian T. Levy
Phone: (360) 212-4219
Fax: (360) 212-3035

Ron Odenheimer
Phone: (360) 212-6126
Fax: (360) 212-3035

Hewlett-Packard Company
1115 SE 164th Avenue
Vancouver, Washington 98683

Practical Techniques to Accelerate Product Development Symposium

Abstract

Time-to-market pressures have created the necessity to identify faster and more reliable design methodologies. Shifting the firmware development phase back in the design cycle to run in parallel with ASIC design has enabled greatly reduced design cycles and increased the probability of first silicon meeting all the system requirements. FPGA and logic synthesis technologies are key enablers of this methodology.

Authors/Speakers

Brian T. Levy

Current Activities:

Brian Levy is a member of the ASIC design team working on designs for future products.

Author Background:

Brian Levy has a MSEE from Stanford and 13 years experience in Analog and Digital IC design.

Ron Odenheimer

Current Activities:
Ron Odenheimer is a member of the ASIC design team at Hewlett-Packard, Vancouver, WA. working on future inkjet printer products.

Author Background:

Ron Odenheimer has a BSEE from California Polytechnic University, San Luis Obispo, Calif. and has 19 years experience in industry and 15 years experience working on digital IC design.

Slide #01

Rapid Prototyping Methods for Embedded System Development Reduce Time-to-Market

Brian Levy, Ron Odenheimer
Hewlett-Packard Company
1115 SE 164th Avenue
Vancouver, Washington 98683

HEWLETT® PACKARD
Expanding Possibilities

The time-to-market of an inkjet printer was significantly reduced through the use of co-design methods employed in ASIC and firmware development. Providing the tools and methods for running these development efforts in parallel allowed early access of a hardware platform for firmware development and reduced risk of unanticipated system interaction and ASIC functional problems. The current technologies for HDL synthesis and FPGAs are key enablers of a co-design methodology.

Slide #02

Major Topics

- **Introduction**
- **Project Development Objectives**
- **Development System Overview**
- **Emulation with FPGAs**
- **System Debug and Verification**
- **Results**
- **Team/Organizational Issues**
- **Areas for Improvement**
- **Summary**

This paper begins by offering some background information to help understand the product objectives and design environment.

An overview of the development system highlights all the elements necessary to achieve the design and verification objectives.

Since the current FPGA technology is a key enabler of this methodology, emulation with FPGAs is discussed in some detail.

The results achieved demonstrate the effectiveness of this methodology.

There is an element of magic or luck in bringing together a number of people to form a high performance team and in creating an environment in which multiple teams work most effectively together. Not professing to have “the answer,” some ideas that worked for these teams are presented.

In an effort to become a truly Learning Organization, we look back to see what was really effective and what we might do differently in the future.

Slide #03

Introduction

- Development of Color InkJet Printer
- Critical Coupling Between Driver, Firmware, Electronics (Software + Hardware)
- Rapid Time-to-Market (Schedule, Schedule, Schedule)
- Reduce Risk through Rapid Prototyping

The product under development was leveraged from a previous product with changes planned to increase the speed and print quality of the printer output and reduce product cost. This required significant changes to some core functional blocks of the ASIC and significant changes to the firmware and driver. While these were the major areas affected there were also additional changes included to achieve the cost reductions.

The schedule, performance and resources were specified and constrained so the risks needed to be reduced as much as possible to give the product design team the best chance of success in meeting the objectives. We had good results using rapid prototyping tools on previous projects and with the size of FPGAs expanding rapidly we expected reduced prototype system complexity, faster turn-around times and higher clock rates.

With this in mind, we set out to identify current tools and technologies that would ease rapid prototyping so we could build firmware development systems earlier in the product development cycle thus reducing the risk of encountering obscured system integration problems and ASIC functionality problems.

Slide #04

Project Development Objectives

- Start Integration of Firmware and ASIC Code before ASIC has been completed
- Run target firmware on target hardware
- Exercise all critical Print Modes
- Print using major modes prior to freezing ASIC
- Run at 1/2 target clock speed of 16 MHz
- Release Alpha Firmware for on-board ROM
- Exercise I/Os at speed
  - IEEE 1284, High Speed Serial (Apple)

Specific objectives were set which would drive the co-design effort and reduce risk. From past projects, we had found that the earlier the firmware and ASIC engineers are working together to prove the functionality of a system element, the earlier design issues and bugs are identified and the easier they are to fix. Design architecture or strategies may even be adapted based on new information at this early point in the project.

To reduce schedule and improve resource utilization, the amount of effort expended on test stubs and maintenance of multiple hardware platforms had to be minimized. Sometimes it seems easier up front to do things differently for emulation than for the final ASIC. There are multiple dangers and inefficiencies along that path. Maintaining multiple (and slightly different) versions of the same block is time consuming and risky. The “real” design is not being exercised in the currently operating prototype! Also, there were certain functions that were required by the chip but not necessary for the operation of the printing emulation system. These elements were placed at a higher level of the ASIC design to avoid unnecessary complexity in modules targeted for FPGA implementation.

One milestone that is very clear and well-defined in printer development is printing legible text and correct images. By setting the objective of printing using major print modes, almost all of the modules of the ASIC and firmware are operated in concert. This exercises the firmware in a manner typically not achieved until prototype systems are built with first silicon ASICs. The functional coverage of ASIC code is also greatly enhanced.

One compromise that was made to allow the use of FPGAs was a reduced system clock rate. This decision was based on lengthy discussions of objectives between management, firmware and hardware engineers. It would have taken an unknown additional time to achieve full speed, and the possibility existed that blocks of HDL would have to have been re-written to achieve the goal. In the end it was decided that most of the ultimate goals would be achieved at a reduced clock rate and still make schedule. Due to EMI costs and risk, the printer system clock was targeted at 16 MHz. Initial studies indicated that this was still too fast to run the emulation system without significant further investigation. Thus the objective of running at 8 MHz or half the target system clock was set. Additional configuration registers were added to the design so that scale factors could be set programmatically, thus allowing real-time operation of areas such as I/O and DC servo.

Slide #05

Development System Overview

- **HP 68000 ICE for Firmware Development**
- **HP 16500C Logic Analyzer**
- **HP PC Running Driver to initiate Print Jobs**
- **ASIC Tools**
  - **coded in Verilog**
  - **Simulated with Verilog-XL**
  - **Synthesized with Synopsys HDL Compiler**
- **HP 735/125 workstations on LAN**
- **Leveraged Previous Generation Mechanism**
- **Single PC Board, all product components plus three ZIF sockets for FPGAs**

The development system was composed of firmware development tools, ASIC development tools, digital system debug tools, prototype printer and PC running driver.

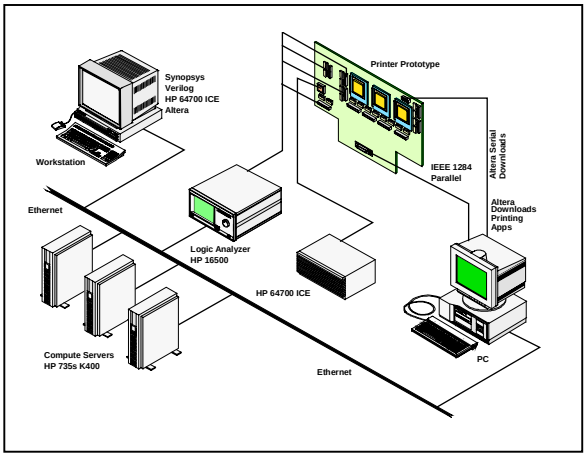
The firmware development environment consisted of an HP 735 workstation running the HP emulator interface and development and debug environment, for the Motorola 60000 processor, and an HP in-circuit emulator for the 68000 connected to the LAN and a 68000 socked on the printer.

The ASIC was coded in Verilog HDL, simulated with Verilog-XL, compiled with Synopsys HDL Compiler. Waveform debug was done with Undertow.

The ASIC was compiled for either the target HP10 process or Altera 10K100 FPGAs.

The printer mechanism was leveraged from the previous product. The PCB was enlarged to include three ZIF sockets for the FPGAs and connectors for logic analyzer pods for signal observation.

Slide #06



Slide #07

Emulation with FPGAs

- Consider emulation requirements and trade-offs early
  - affects ASIC design and architecture
  - affects firmware
- Emulation efforts may be minimized with up front planning
- Emulation benefits both ASIC and firmware
  - firmware gets earlier look at hardware
  - ASIC benefits from additional exercise of design
- Plan chip hierarchy with FPGA capabilities in mind
  - Eliminates duplication efforts for ASIC and testbed

A successful ASIC/firmware project, if it uses emulation, will consider emulation from the beginning. The ASIC design and architecture is affected, as is firmware. It is surprising how a little up front planning can minimize the emulation effort.

Tip: If project pressures allow the time, try to fit your design into two or more vendor's FPGA products. It is usually not necessary to complete the entire process, just place and route to arrive at size and speed metrics.

Both firmware and ASIC teams benefit from an emulator that is as much like the final ASIC as possible. Firmware gets an early look at hardware and a platform on which to develop code. The ASIC team has another avenue beside simulation to test the design.

Slide #08

Structuring for Synthesis

- Common HDL Compiler used for both ASIC and FPGA synthesis.
  - delivers best results when synthesized blocks result in less than 10kgates.
- Keeping the size down also facilitates FPGA partitioning.
  - easier to relocate smaller blocks into another FPGA.
  - FPGAs get harder to partition with huge blocks.
  - results in fewer opportunities to tune partitioning for faster performance.

We used a common HDL compiler for ASIC and FPGA synthesis. The compiler works best when it can synthesize code that results in modules less than ten thousand gates. This size range also facilitates partitioning the design for implementation in multiple FPGAs. They are big enough to get adequate functionality yet small enough to easily move during the partitioning process. FPGAs get harder to partition when modules are huge. If room is needed in an FPGA, it is easier to move a small module than it is to break up one large module into components.

Large modules result in fewer opportunities to tune an emulator design. By tuning we mean moving modules around in order to either make more room in an FPGA for faster compile times or for faster operation. Tuning is one of the few degrees of freedom an emulator designer has.

There usually isn't time, nor is it desirable, to design a separate hierarchy for the emulator and the ASIC. If the ASIC hierarchy is architected at the beginning, with the goal that the bottom levels are emulatable, no extra effort is required. Design the hierarchy such that all of the emulator fits at and below a defined level. Make it a design goal: from the selected level on down will be included in the emulator. This makes the emulation system's header identical to the corresponding level in the ASIC. This not only reduces the number of structural verilog netlist files that must be maintained but it also allows simulations designed for the ASIC to run on the compiled emulator gates.

Emulation is not a replacement for good functional test vectors. Each method has its own place and each has unique capabilities of finding different types of problems.

Slide #09

ASIC Hierarchy Example

- **Testbench**
  - **Clocks defined**
  - **Board level components**
  - **Instantiation of ASIC top level.**
  - **This level is not part of the ASIC to be synthesized.**
- **Top level**
  - **This is the complete chip - top hierarchical level.**
  - **Instantiation of pads.**
  - **Instantiation of Core.**

Once definitions for each hierarchical level are created, the question of where design entities belong falls easily into place. The hierarchical requirements for the emulator will support the ASIC effort almost one for one and vice versa. An example of the suggested hierarchy is outlined in the slides.

Slide #10

ASIC Hierarchy Example

- **Core**
  - **Instantiation of RAM, ROM, CPU, PLL.**
  - **Instantiation of LC module**
- **LC**
  - **Includes test structures that will be synthesized as part of ASIC but are not needed for printing system and cause problems for FPGAs**
  - **Instantiation of SC module**
- **SC**
  - **Everything at this level and below goes into emulator**
  - **Instantiation of all major functional modules**

The last level was designated as being emulatable. Having made that decision, the placement of reset tasks, miscellaneous clock circuits, RAM, ROM, CPU becomes easy to place in the ASIC hierarchy.

Embedded ASIC entities that cannot be emulated are added as external devices.

Slide #11

Simulation Benefits of Following Hierarchy Conventions

- **The same test vectors used for simulation runs on ASIC and FPGA implementations.**
- **Any differences are accommodated through the use of “ifdefs” in the verilog testbench.**
- **This leverage of test vectors leads to greater confidence that emulation system will function properly.**

Our test vectors are used in both environments ASIC and emulator as a result of the above hierarchical rules. Any path differences, such as fewer levels of hierarchy for the emulator, are handled by "ifdefs” in the testbench that is used for both the ASIC and the emulator.

Running simulations on the emulator gates offers earlier detection of any functional problems that may have resulted from compile problems and increases confidence in the functionality of the FPGA download.

Slide #12

Code for Emulation

- **Negative edge flip flops**
  - **Hard for timing constrained FPGA synthesis**
- **Transparent latches in ASIC**
  - **Subject to skew and delay problems in FPGA**
  - **Inconsistent results, differ from one compile to the next**
- **Internal tri-state buffers**
  - **FPGA tool converts to gates anyway**

Coding for emulation is a paper by itself and a topic that is well covered in the literature put out by the FPGA manufacturers. It will not be covered here in detail except for three issues that are problematic.

If possible don’t design with negative edge flip-flops or use inverted clocks. Each time we’ve done this it has resulted in designs that are hard to compile. Those modules that have used negative edged flip-flops have been the slowest in the design.

If possible, don't design with transparent latches. Many FPGA vendors don’t include a latch cell in their library so it becomes routed logic with the potential for glitches and skew problems. Often the compile results will be inconsistent; the circuit works on one compile but not the next.

Try not to use internal tri-state buses. If you can, route separate buses for read and write lines and combine read buses with combinational logic.

Slide #13

Maintaining Identical Source Files for ASIC and Emulation System

- Goal - no branching of source code to support emulation.
  - target firmware run on target hardware.
  - exercise a real working model of ASIC.
- Sometimes this requires the addition of incremental gates
  - An example is the addition of a configuration register to initialize the counter in the 1MHz clock generator.
- Small gate cost delivers significant benefit.

While ASIC area equates directly to cost, the addition of a small number of gates to improve the overall ASIC results often pays real dividends. One example is SCAN, which adds considerably to the area but also greatly enhances testability, thus saving costs due to test escapes. Adding a few additional registers or logic to a design to accommodate emulation is similar in that it enhances the testability of the ASIC up front as it is being designed. Our goal was to have an identical database for the emulator and the ASIC.

We took the tack that if we could add a small amount of hardware and it would help to meet the above goal, it was worth it.

As an example, we added hardware to speed up clocks for real-time activity. Since the emulation system ran at half the targeted system clock, any real-time reference clocks needed to be scaled back up when running in the emulation environment. Hardware was added to insure the 1 MHz derived clock was the same for the ASIC and emulator. The hardware cost was small and the benefit of identical ASIC and firmware source code was significant.

Slide #14

Partitioning SC Level to Multiple FPGAs

- Make it easy to change
  - Automate as much as possible
- Gate Count
  - Leave room for expansion, revision and the unknown.
  - Don't squeeze to fit into one part.
- Interconnections
  - Partition to minimize number of interconnects.
  - Partition to minimize delay of interconnecting signals.

You can ignore this section if your entire design fits into one FPGA. With the higher density FPGAs available today, more designs will fit into a single FPGA, especially with FPGAs approaching 250Kgates. However, many designs will still require multiple FPGAs for emulation and, thus, will also require the design to be partitioned.

We have had experience in the past with automatic partitioning, but have found that the task is very complex for a software tool to perform. This complexity translates into long compute time and frequent partitioning problems. Since the partitioning task is only necessary at the beginning of a design and when a group of modules expands beyond the limits of the FPGA, there is a lot of compute time wasted repeating the task every time another compile is launched. On this project, we decided to manually partition the design. This is really no more than grouping functional blocks together and paying attention to total size and interconnect requirements vs. FPGA resources.

We start with estimated gate counts of functional modules that would go into the emulator. After a number of trials we determined the number of ASIC gates that would fit into one FPGA and partitioned the design estimated to be 50Kgates to fit into three FPGA devices.

Tip: Leave plenty of room for expansion. Most of the FPGA place and route tools work best and compile quicker when parts are not filled to capacity.

There is as much art as science to placing the right group of functional modules together to form an FPGA partition. Although it seems like the partitioning task can be broken down into its components, keep total gate counts low enough in an individual part to allow room for expansion, bug fixes and future revisions. Keep functional blocks together, and keep the frequency of external signals to a minimum, keep interconnecting pins under the maximum for the package. However, partitioning is not always as straightforward as it seems. Circuits take more space than intuitively obvious. Interconnects between functional blocks become pins that add up quickly past maximums. FPGA place and routing efficiencies and tool idiosyncrasies make the partition equation complicated, to the point that we have come to the conclusion that the best way is to hand pick a partition, then try it. Make it easy to change partitions and try new ones. It usually takes a couple of tries.

Usually the ASIC is not complete when the emulator project starts. We left room for expansion, so we did not have to re-partition in the middle of the project.

Try to anticipate some of the future pins necessary and reserve them. Some FPGA vendors utilize unused pins as routing resources, so you can't assume that because your design has not used a particular pin that it is going to be left free. We dedicated some extra pins to spares to alleviate this problem.

Rigorous adherence to naming conventions helps speed the partition problem by promoting faster understanding of unfamiliar code.

Slide #15

Effective Use of Scripts

- **Allows more control over partitioning.**
  - **The designer is in the best position to group modules.**
- **Generate accurate results**
  - **Large number of nets and ports - manual interconnection is error prone.**
  - **Speeds creation of PCA schematic.**
- **Usable in future projects.**
  - **Good scripts formalize a process that can be used as is, or adapted to subsequent projects.**

This project relied on scripts for synthesis and simulation, but it could have greatly benefited from scripts to put partitions together and check them for accuracy. Currently, we are using scripts in the emulator creation process. Scripts are a great tool for helping with the first time accuracy of emulator design. With pin counts of the latest generation of FPGA's approaching 600 and systems involving 3000 pins or more it's all too easy to make a mistake interconnecting multiple FPGA packages.

We wrote scripts to automate the process of FPGA module creation and inter-FPGA wiring.

We still wanted to assist the placing and wiring of all the FPGAs on the board schematic. So another script was created to take the output of the Altera FPGA tool with pin assignments and convert to Mentor Graphics schematics. With this script, a schematic page was created for each FPGA.

Tip: We wrote a script converting the text based pin assignments generated by the Altera tool (Max+PlusII) into Mentor Graphics schematics. The result was a huge saving in time and PCB accuracy.

The script used Perl and Ample to create a single-level-hierarchy schematic of each FPGA that automatically connected the appropriate pins between chips.

The same project structure rules mentioned earlier, plus attention to coding standards, also helps when it comes to scripts.

We are getting better and better at getting everyone on the team to be consistent in the use of file, module and instance names. We call an instance the same as the module name or, in the case of multiple instantiations, we use a suffix to distinguish between the instances. Files are named the same as the modules they contain. For scripts, it means that parsing through the directories, files and modules is easier to accomplish.

Slide #16

Emulator Board Design Issues

- **Single board implementation results in reliable operation.**
  - **solid and robust connections improves reliability.**
  - **No cables running between multiple boards.**
  - **Small footprint on engineers desk.**
- **Emulation PCA mounts directly to mechanism.**
  - **results portable implementation.**
  - **may be easily moved without damage or problems.**
- **All critical signals brought out on LA pod connectors.**
- **ZIF sockets used for FPGAs.**

The board we designed for this project was different from previous emulator designs. Where past emulators were table top boards cabled to the printer mechanism, this emulator board was simply an expansion of the printer PCA to include three ZIF sockets for the FPGAs, many logic analyzer connectors for signal observation, and other emulation support circuitry.

The board, about 5X larger than the production board, contained the larger voltage regulator needed for the FPGAs, as well as external CPU, memory, and analog components.

Our emulators do not seem to stay in one place and take a beating during moves. We tried to make the systems robust enough to withstand the punishment of day-to-day wear. Eliminating many of the cables from previous emulator designs resulted in a highly portable test vehicle with superior reliability.

Tip: Just because it's a prototype doesn't mean that it should not be mechanically sound. Use good mechanical practice to secure test equipment cables and emulator pods. If you can, don't obscure the front of your board by keeping logic analyzer and other cables from draping over it.

Without a doubt, boards are more susceptible to static than in the past.

Tip: Educate firmware and/or software co-workers on proper ESD procedures. This avoids wasted time and money in ESD damaged chips.

The FPGAs become the first component substituted when something fails to work. Without proper ESD precautions, there is a chance of causing catastrophic or partial failure of the FPGA devices. This becomes very expensive, given the cost of the latest generation of dense devices.

Slide #17

Emulator Change Management

- Developed revision documentation and communication process
  - used web browser for easy access.
- Scheduled incremental releases.
- Optimized turn around by trading off performance and packing density for turn around time.

Changes are inevitable. Bugs are identified and fixed and improvements and additional features are added. Sometimes changes are made to bring out an internal signal for debug or timing or to add special hardware for a special development function.

We used some basic rules governing what special change requests were honored and tried to nominally limit ourselves to one hardware update per week.

We tried to limit the constraints placed on the design in both the synthesis and FPGA place and route processes. This turned out to be an iterative process, always compromising between performance and fast turn-around time.

We did try to keep the change process simple, fast, and heavily scripted. The scripts make it painless to kick-off an overnight compile or spawn jobs over multiple machines to speed up the overall turn-around time.

Slide #18

FPGA Programming

- Two programming methods for FPGAs:
  - Download cable; used during emulator debug.
  - Serial PROMS places in sockets on PCA; downloads automatically during power-on cycle. (two per FPGA)
- Future improvements to FPGA Programming:
  - Single Flash for all FPGAs.
  - Takes less time to program and handle.

The FPGAs parts are RAM-based and require an external source for configuration information. We designed two separate configuration methods onto the emulator system.

The download cable proved it's worth during emulator board debug, allowing quick turn-around time. Firmware engineers with special hardware requirements also tended to use this method.

Once the hardware settled, serial EPROMs were used, which automatically download configuration data each power-up cycle.

One of the advantages of programmable FPGAs can also become a burden. Revisions are easily generated. Make sure that revisions between FPGA code releases are well documented and easy to maintain.

The large Altera parts, for example, require two serials ROMs per FPGA. This project used three FPGAs per system. The six ROM's required to configure the devices needed to be well marked as to their relative position on the board as well as indicating what revision of the code they contained.

Every project goes through at least one instance when new firmware compile is coupled with new FPGA code, and the design doesn't behave the way it did "before". At that moment, a clear understanding of what was changed is vital to a quick resolution of the problem. Clear revision control helps maintain and control the inevitable special versions of the hardware.

We used web pages with revision information for availability (UNIX systems and PCs) and easy access.

Each new firmware revision was verified. Our test equipment played a major role in making that job go faster and easily.

The verification process frequently involved people from multiple disciplines and domains. The normal troubleshooting group, consisted of firmware and digital hardware engineers, however, frequently the assistance of mechanical or analog engineering was needed to get at root cause. And then Murphy showed up.

Slide #19

System Debug and Verification

- **Configured FPGAs with additional outputs of critical "internal" nodes to view more meaningful information.**
- **Logic Analyzer was key to capturing hardware state for verification and debug**
- **Created "split-out" board for carriage**

A form of Murphy's law says: two digital signal outputs shall be inadvertently connected together. And those two (or more) points shall always be the same level, except once in a very great while and seemingly in a random fashion. In this project we thought we would be smart to add spare traces between the three FPGAs on the PCB.

Those spares were left unassigned inside the FPGAs. However, the Altera FPGAs use unassigned pins as tie points for internal nodes. So the spare traces inadvertently connected various internal nodes, outputs and inputs together among the three FPGAs.

Slide #20

Verification Examples

- **Debugged PC board connectivity**
  - **identified FPGA connection problems**
  - **missing pull-ups (needed on development board not ASIC)**
  - **discrete analog section for servo feedback processing**
- **Evaluate serial interface to carriage analog ASIC**
- **Look at pen control to carriage interface**
  - **handshake between carriage and pen control**
  - **1200 DPI twice rate of 600 DPI - timing concerns**

Having both an oscilloscope and a logic analyzer together was extremely useful for this problem. The logic analyzer is the only way to find difficult problems, especially when they occur infrequently, like these did. Once a logic analyzer trigger was created, the oscilloscope was used to identify non-logical (analog-like) behavior.

We used this tool combination frequently for finding and solving various specific problems, as well as verifying system performance.

Slide #21

Verification Examples (2)

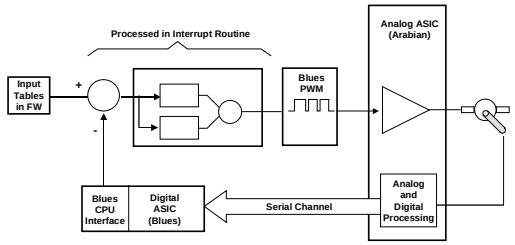
- Carriage position evaluation
  - anticipated as problem area
  - added internal nodes as outputs in FPGA
    - carriage position register after decode and extrapolator
  - easily hooked up to logic analyzer
  - first FPGA compile didn't function properly
    - state transitions were wrong
  - FPGA recompiled with modified constraints
  - verified as functioning properly through state transitions

Tip: Oscilloscope probe clips are available that solder in place on the PCB, they hold an oscilloscope probe directly to the PBC. Locating them at strategic locations on the board allows convenient hands-free-testing for test points that need to monitored frequently.

One instance of the verification problem was verifying accurate paper movement between successive rows of ink on the page. This motion is controlled by a DC motor and servo system. Position performance of this block is specified to be more accurate than 1/10,000 inch and has to cost nearly nothing. This servo has a very highly convoluted, but highly accurate and inexpensive, position feedback system. The feedback consists of two components one digital the other analog. Both components are sampled at the same time but then they are spit up, serialized and delivered to the digital ASIC at different times. The two components are combined by firmware during the interrupt service routine to form the paper position feedback term.

Slide #22

Printer Paper Position Servo System



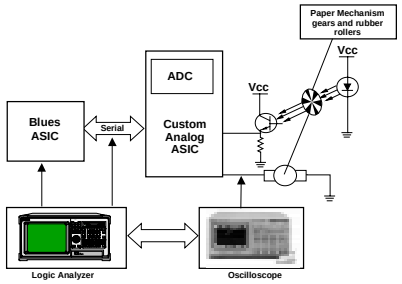
- Uses analog/digital and firmware components

Verifying that the components are being re-combined correctly is a complex task. It is necessary to time correlate analog, serial digital and firmware tasks.

We used the HP 15600C integrated logic analyzer and oscilloscope and 16505A prototype analyzer to verify that the analog and digital components were being re-combined at the digital ASIC (emulator) into a proper working servo system. HP now offers the enhanced HP 16700A system for debug.

Slide #23

System Verification



The system was partially verified as analog with the oscilloscope. The logic analyzer de-serialized the data stream and on the ASIC emulator we were able, with the logic analyzer, to verify the completed feedback path.

Slide #24

Results

- Emulator flexibility and turn around allow for fine tuning of firmware algorithms
- Printed with emulation system 4 weeks prior to netlist release.
- Lab Protos Printing (complete printer with ASIC)
  - three hours after delivery of first chips
  - running from on board ROM
- Identified timing problem w/interface to Carriage Analog ASIC
  - not found earlier due to 1/2 speed system clock

The flexibility and turn around time of the ASIC emulation system was key to meeting the schedule. An incremental turn on one FPGA took as little as 3 hours and a major change involving all three FPGAs was accomplished overnight.

The system was fully integrated and printing four weeks prior to netlist release. The remaining time was used to work through a prioritized list of system exercises, debugging, tweaking code, and verification of code changes.

A snapshot of the firmware was taken just prior to artwork release (about 2 weeks after netlist release) for programming the on-board ROM.

When first silicon was delivered there were some problems with the adapter board for the ICE system, so ASICs were soldered directly to a couple of printer PCAs. The first system was up and running and printing within 3 hours of the arrival of first silicon!!

While the results were very good, it wasn't quite shippable silicon. A timing problem was discovered in an interface to the analog ASIC which didn't exist in the emulation system. This problem did not occur in the emulation system due to the reduced clock speed; ½ target clock speed.

Slide #25

Team/Organizational Issues

- Used one area (Integration Cube) to bring up new code prior to rolling into other emulation systems.
- Co-location of Firmware and Hardware team increased communication.
- Experienced ASIC Engineer responsible for compile emulation process.
- Having a foundry engineer as a member of the team was helpful in delivering high quality netlist.
- Team accommodated change of manager and technical lead.

The ASIC and firmware teams were interspersed, placing hardware and firmware folks that were working on subsystems next to each other. This allowed frequent discussions leading to quick accurate plans without the many assumptions that are sometimes made to keep progressing.

In order to minimize the effort and problems caused by bringing up new ASIC code on the emulation system there was one central area with a complete development system where all new hardware code was exercised before being propagated to other systems. This allowed the firmware team to keep making progress on the current revision while the next revision with additions or changes was debugged. If the change or addition was made at the request of a firmware person this individual would join others in the common area to assist in debugging the latest revision

Taking on the many tasks required for emulation sometimes doesn't get all the respect it deserves. However, having an experienced ASIC designer involved can make a big difference. On our team, the task was divided between a very experienced person and a recent graduate. The experience and ability to look at the effort from a broad perspective allowed problems to be avoided and thoughtful contingencies to be laid in place.

Another benefit that occurred by chance was the addition to the team of an experienced ASIC engineer who worked for the foundry. They took on design responsibilities for modules that improved the testability of the chip and pushed for improvements in the synthesis process to deliver a better netlist; one that needed a minimum of manual adjustments for successful place and route.

It is also interesting to point out that the leadership of this team changed twice. The team accommodated a change in both the project manager and lead engineer. This speaks to the integrity and synergy of the team.

Slide #26

Areas for Improvement

- **Increase system clock speed**
  - **Pen Interface problem slipped past emulation due to reduced emulation vs. target clock speed.**
- **Sync up Firmware and Hardware Schedule**
  - **ended up cutting corners to get system printing**
- **Improving manual partitioning process with scripts.**

As mentioned earlier in the results section, the compromise of reducing clock speed did allow a problem to go undetected in the emulation system. While significant benefits were accrued as a result of this methodology, it would be very nice to run the emulator system at target clock speeds.

Another area that has been a challenge is getting management and the firmware team fully enrolled in this methodology. There were some potential barriers in the experienced person's thought process about when firmware milestones really must be achieved. This methodology is new and different; not like the old tried and true method. Since firmware is not released or “frozen” until much later in the product development it is easy push out firmware milestones. The goal of printing before ASIC netlist release has helped bring this into focus.

Structuring the ASIC with emulation in mind aids manual partitioning to target FPGA sizes. Automating this process further with scripts will save time and increase the reliability of results.

Slide #27

Summary

- Co-Design of ASIC and firmware improves time-to-market.
- The methodology is enabled through the use of current technologies for Logic Synthesis and FPGAs.
- Design for Emulation minimizes the resources and time required, and improves reliability of results.
- Debug with integrated HP 15600C uncovered timing problems.
- Effective design practices, i.e., use of conventions, rules and scripts, is also essential for success.

Slide #28

Current HP Logic Analyzer Products

- HP16600A and 16700A Logic Analysis Systems
  - simultaneous triggering of Scope, Logic Analyzer and Source Code
- HP Serial Analysis Tool Set
- HP 16557D Deep Memory State/Timing Analyzers
- HP 16534A 2GSa/sec, 2 channel Oscilloscope