PPA Printer Controller ASIC Development

As the first Printing Performance Architecture printer, the HP DeskJet 820C needed a completely new digital controller ASIC design. The chip's architecture was optimized for the specific requirements of PPA. Concurrent development of hardware and firmware through the use of hardware emulators and attention to regulatory issues during the design helped the product meet all of its requirements on schedule.

by John L. McWilliams, Leann M. MacMillan, Bimal Pathak, and Harlan A. Talley

The Printing Performance Architecture (PPA) used in the HP DeskJet 820C printer is a significant step forward from any previous HP inkjet printer product in providing the consumer with a high-performance product at an excellent price point. Since PPA redistributes the printing tasks between the host and the printer, a complete redesign of the digital controller ASIC in the printer was required. This redesign effort took into account the overall product constraints of cost and time to market as well as all applicable government regulations. The result is a highly integrated ASIC that implements all digital functions performed by the HP DeskJet 820C on a single chip. This high level of integration significantly decreased the cost of the electronics in the HP DeskJet 820C compared to the previous-generation product while maintaining the printer's performance. This article describes the system considerations, engineering decision trade-offs, and development methodologies that played a role in the development of the digital controller ASIC for the HP DeskJet 820C.

The design of the controller ASIC had to be done under numerous constraints. As in any consumer-oriented product, the foremost consideration during design was the final cost to the buyer. The Performance Printer Architecture, as described in ***Article 1***, was developed to reduce the total cost of the printer. PPA allows several optimizations in the digital architecture. In today's competitive environment, time to market is nearly as critical a constraint as cost. Meeting the time-to-market constraint required concurrent development of hardware and firmware and a bug-free ASIC at netlist release. These needs were addressed by using hardware emulators during development. Finally, the printer had to meet or exceed all government regulations including those pertaining to EMI and ESD. Taking these needs into account during the ASIC design helped the product pass all requirements on schedule.

Digital Architecture

Regardless of their specific type, all printers require several pieces of digital hardware. These pieces include a microprocessor to control the printer, RAM for data, ROM for firmware, and custom digital logic for printer-specific functions. By optimizing each of these pieces, significant cost savings were realized in the digital ASIC.

PPA significantly reduces the cost of the printer by optimally partitioning the printing tasks between the software running on the host and the hardware and firmware running in the printer. The partitioning is done without sacrificing the printer's performance. All tasks that can be done on the host computer without severely affecting application performance are done in the driver. Tasks with real-time constraints are performed by the hardware and firmware in the printer. Because the host performs the majority of the data manipulation, data that is sent to the printer is in a format that is very close to the final form used to fire the printheads. Because of this, the digital architecture was designed with a guiding principle of "the processor does not touch the data." Once this principle was adopted, the ASIC team was able to make several important design decisions.

First, a relatively low-power processor is all that is needed, since the processor does not manipulate the data. After surveying the available microprocessors, the 16-MHz version of the Motorola 68EC000 was chosen as the best fit. Second, since the number of tasks the firmware performs is limited, the code size can be kept small enough that a ROM with all firmware can be integrated on the ASIC, eliminating the need for an external flash memory or ROM. Third, all data manipulations need to be done in hardware, which limits those manipulations to being relatively simple. Finally, the memory requirements are limited—a 1M-bit DRAM is sufficient for the data needs. The DRAM holds all firmware variables and stacks as well as all printing data. Even with the DRAM doing double duty, the memory bandwidth requirements of the architecture are fairly low, and the product is able to use a low-cost, 1M-bit, nibble-wide DRAM.

A block diagram of the HP DeskJet 820C's digital architecture is shown in Fig. 1. The digital electronics consists of three main components: the digital ASIC, a 1M-bit DRAM, and an optional external flash memory or ROM. The digital ASIC consists of a 68EC000 microprocessor, a 64K-byte ROM, a 55,000-gate standard cell block, and a 1K-byte SRAM used as a

data cache. In addition to the external memory components, the digital ASIC is connected to the I/O connector (IEEE 1284), the printer motor ASIC, the printhead ASIC, and an optical encoder which provides carriage position information.

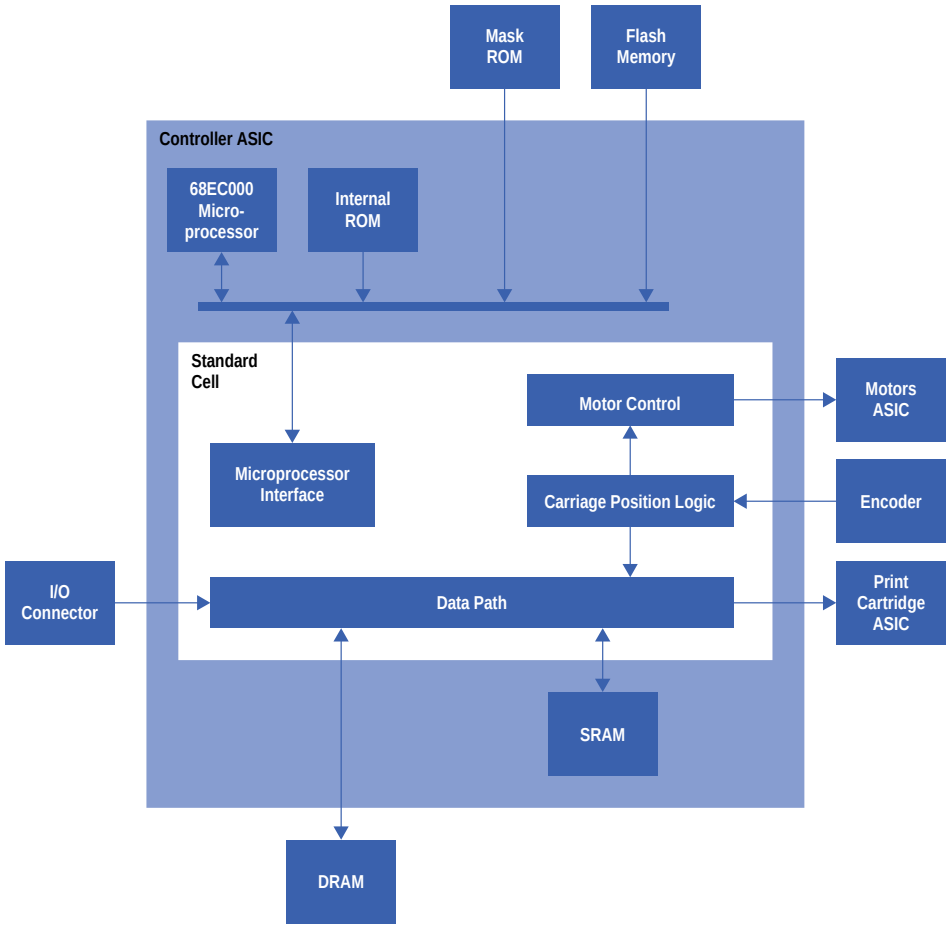

Fig. 1. *HP DeskJet 820C controller ASIC block diagram.*

The majority of the standard cell area is devoted to the data path, which is the path the data follows as it moves from the I/O connector, through the DRAM and SRAM, and up to the pen ASIC. The remaining logic is used for interfacing to the microprocessor, for controlling motors, and for keeping track of the current carriage position. All memories, including registers in the standard cell block, are memory mapped into the 68EC000's standard address space.

Flash or ROM

The ASIC is designed to be able to read code for the processor from one of three sources: a flash memory device, an external mask-programmable ROM (MROM), or the internal ROM. The reason for the three separate sources is to better meet time-to-market constraints. At the beginning of the manufacturing ramp, code was stored in flash memory. That way, final firmware did not need to be released until just before the start of the ramp. As soon as the firmware was stable, it was released to both the MROM vendor and the digital ASIC vendor for programming into the internal ROM. However, MROM lead times are much shorter than general-purpose ASIC lead times, so MROM parts were available much sooner than ASICs with properly programmed internal ROMs. Consequently, printers were built with MROMs for a period of time until ASICs with final firmware were available (MROMs are about half the cost of flash parts).

Motor Control

The HP DeskJet 820C has three motors: a dc motor for moving the carriage across the paper, a stepper motor for picking and advancing the paper, and a second stepper motor for controlling the pen service station. The stepper motors are controlled in an open-loop process by the firmware. The firmware controls a stepper motor move by writing appropriate phase and pulse width data to registers in the ASIC. Hardware then generates the appropriate signals for the motors. The phase and pulse width data determines the direction and speed of the moves.

The carriage motor is controlled by a firmware-based control loop that monitors the carriage position and adjusts the motor control signals appropriately. The carriage position is determined through the use of an optical encoder. The optical encoder consists of a light emitter-detector pair with a plastic encoder strip between them. As the carriage moves across the paper,

the light emitter-detector pair senses that it is moving along the plastic strip, and sends some signals to the ASIC. The hardware in the ASIC takes this information and uses it to keep track of the current carriage position. Using the carriage position, the firmware tracks the carriage's speed and acceleration and adjusts the motor energy appropriately.

PPA I/O Packet Format

The data from the host comes to the printer in a simple packetized format. As shown in Fig. 2, the packets are made up of two pieces: header information and data. The header information consists of a start-of-packet (SOP) byte, a channel byte, and a two-byte data-size field that reflects the number of bytes in the data field (0 to 65K).

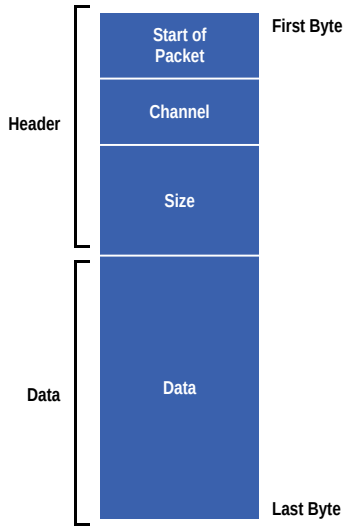

Fig. 2. *PPA data packet format.*

In the HP DeskJet 820C, packets from the host may contain one of two types of information: command data or image data. The channel byte determines which type of data is contained in the packet (hence, in the HP DeskJet 820C, the channel byte will be one of only two distinct values). Command data contains PPA printer control commands, while image data contains information that is to be printed on a page. The image data that is sent to the printer is in a form that resembles a bitmap of the image, and therefore requires a minimum amount of reformatting before being used to fire the printheads. To minimize the amount of data that must be sent over the I/O cable, image data is optionally compressed before being sent to the printer.

Data Path

A block diagram of the data path is shown in Fig. 3. Data enters the ASIC through the I/O cable. Hardware depacketizes it and separates it into the image and command channels. Image data is transferred to one buffer in the DRAM and command data to another, both by DMA. Command data is consumed by the firmware with no hardware interference. Image data is moved by the *servant* hardware from the DRAM to the SRAM. During the move, the image data is decompressed if necessary. From the SRAM, data is moved to a shift register from which it is serially shifted up to the carriage board and is used to fire the pens.

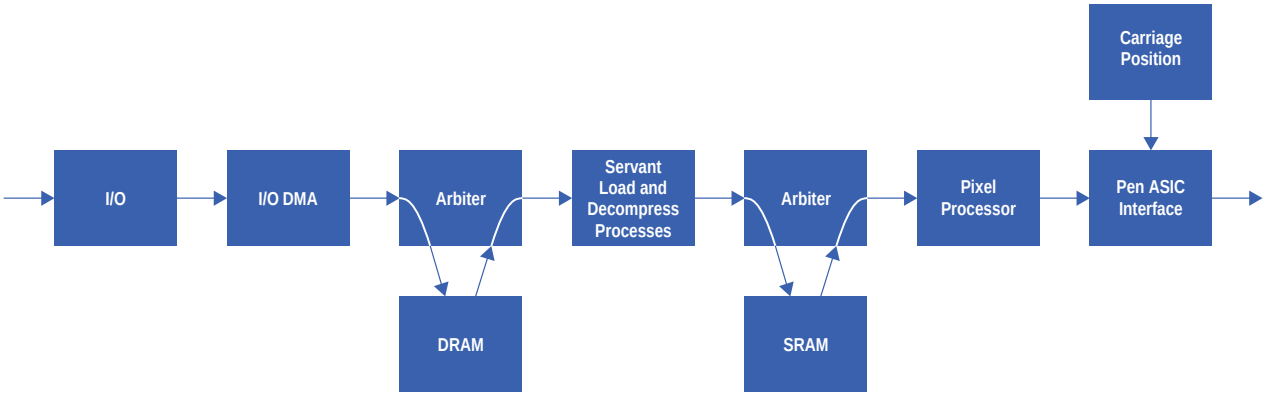

Fig. 3. *Data path.*

Input/Output

The standard cell I/O block implements the low-level hardware that takes in packets of information from the host via the parallel port. It contains hardware support for IEEE 1284 compatibility mode and extended capability port (ECP) in the forward direction from the host to the printer. The hardware also supports, with firmware assist, reverse-channel nibble mode for sending information back to the host computer. The I/O block also contains hardware that strips the data stream of its packet header information, separates the packets into command and image data, and sends them to the I/O DMA block. From the header information, the hardware checks the start-of-packet byte to make sure it is the correct value, uses the channel byte to select the appropriate DMA channel, and uses the size field to determine when to expect a new packet.

The I/O DMA block receives data via the I/O interface and stores it into either the command buffer or the image buffer in the DRAM. These buffers are designed as general-purpose circular buffers that can reside anywhere in the DRAM memory space. The command buffer is emptied by the processor as it executes the commands. Image data is consumed by the servant hardware. As data from the buffers is consumed, the host is notified, via the firmware, of the available buffer space, and more data is sent down. This architecture allows the printer to make optimal use of its limited memory resources.

DRAM Controller/Arbiter

The external DRAM is connected to its own nibble-wide bus. Hardware arbitrates accesses to the DRAM between the I/O DMA hardware, the servant hardware, and the microprocessor. The arbitration method is a combination of priority and round-robin schemes. Both the I/O and the servant hardware processes have real-time constraints that dictate the maximum length of time they can be blocked while waiting for access to the DRAM. Although the microprocessor is less tightly constrained, it is important that it not be completely locked out of the DRAM for extended periods of time. Hence, while each block has a priority for DRAM accesses, the hardware is designed so that no one unit can hold the DRAM bus continuously.

In addition to arbitration, the DRAM controller takes care of the low-level interface to the DRAM. It interfaces the 4-bit DRAM data bus to the 8-bit microprocessor data bus. Using fast page mode accesses, it retrieves two nibbles and concatenates them into a byte. It guarantees that the DRAM refreshes take place at appropriate times. Although only a 1M-bit part is used in the HP DeskJet 820C, the controller also supports 4M-bit DRAMs.

Servant

One of the key contributions of the PPA architecture is that it moves much of the pixel processing into the driver. The image data sent to the printer is in a format nearly ready to be used to fire the pens. The only significant operation that is not done by the driver is the operation of picking out the individual bits (corresponding to dots on the paper), and sending them to the pens in the correct order. The servant logic, so named because it serves the pen by providing it with pixel data, accomplishes this by loading the data into an on-chip cache (the SRAM), and subsequently pulling it out at the correct time and in the correct order. The cache is divided into sets of swing buffer pairs (one pair for each color) such that while data is being taken out of one swing buffer by the pixel processor logic, new data is being loaded into the other swing buffer by the servant load logic. When the pixel processor consumes all the data in one swing buffer and switches to the other swing buffer, the servant load process begins loading new data into the first buffer. This process is depicted in Fig. 4.

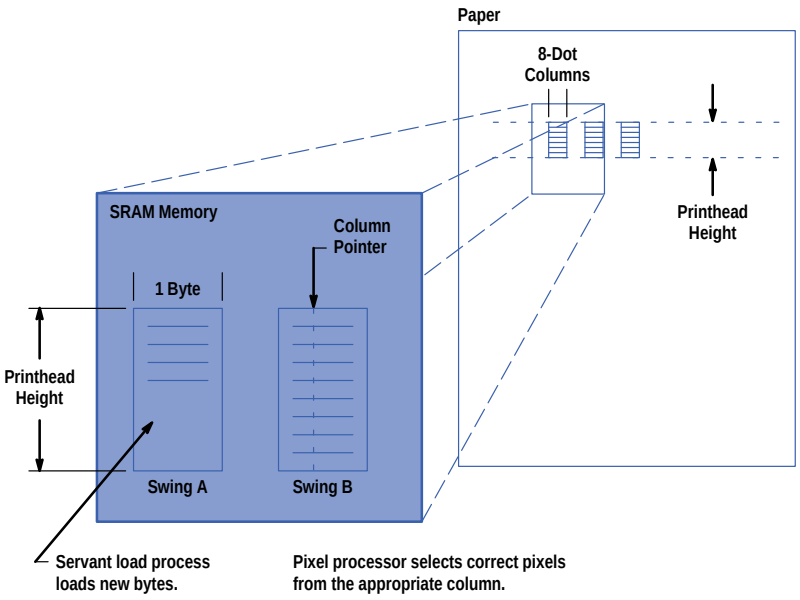

Fig. 4. *Swing buffer operation.*

The PPA driver provides the pixel data in swing buffer loads, which are chunks of a bitmap eight pixels wide and the same height as the printer's pens. The driver provides the swing buffer loads in exactly the order required by the pens. The servant load process transfers the data by DMA from the DRAM to the SRAM as it is needed to fire the pens. During the process of moving data from the DRAM to the SRAM, the data is decompressed if it was sent over the I/O in a compressed format.

SRAM Arbiter

The SRAM arbiter arbitrates memory requests between the servant load process, the pixel processor, and the microprocessor. The arbiter implements a priority-based scheme. Since the microprocessor accesses the SRAM only infrequently, it is given the lowest priority. On the other hand, data for the pen must be immediately available on demand (the carriage cannot be paused for the pen to wait for data). Hence, the pixel processor is given the highest priority. The servant load process has the middle priority.

Pixel Processor

The pixel processor is responsible for placing the bits sent to the pens in exactly the order in which they are needed to fire the pens. Since the nozzles on the pen are staggered, the order in which the bits are needed is not entirely straightforward. As the correct bits are pulled out of SRAM, they are placed in a shift register from which they are serially shifted to the pen driver IC on the carriage board.

Each bit sent to the pens corresponds to a nozzle firing or not firing. Firing data sits in the SRAM in byte-wide chunks. Each byte corresponds to eight columns of dots on the printed page. All dots in a column are fired before beginning to fire the next column. Hence each byte in a swing buffer is accessed eight times, once for each column. After all eight columns are fired, the logic switches to the other swing buffer.

Pen Interface

This block communicates with the analog pen driver IC over a custom serial interface. The pen interface receives data from the pixel processing block and shifts it serially to the pen driver IC. It also generates the timing pulses that the pen driver IC uses to fire the pens and put ink dots on the page. In addition to sending pen firing data, the interface sends setup information to the pen driver IC to adjust various printing parameters that affect print quality. The serial interface is bidirectional, enabling the pen driver IC to send back information about the pens' status. For example, the pen drive IC is able to measure temperatures of the pens, which are important parameters in thermal inkjet printing. This information is sent to the digital IC and read by the firmware. Firmware then uses this information to adjust printing to ensure that the customer will receive optimum print quality.

Because of the staggering of the nozzles on the printhead (see Fig. 1 on page 11), for each column of dots on the page, the pen must be fired multiple times. The pens must be fired every time a set of vertically aligned nozzles is at the correct position on the page. If all nozzles in a dot column are not fired at the same physical position on the page, that column will appear jagged to the customer. Special logic in the chip ensures the proper alignment of the dots and therefore optimum print quality. This logic uses the carriage position as determined directly from the optical encoder, which is at a relatively low resolution, and interpolates it up to the resolution needed to fire the pens. The interpolation is done by phase-locked-loop-like logic that measures the time it takes the carriage to move 1/150 inch, and divides this time down to get the time it takes the carriage to move a distance equal to the nozzle stagger distance. By doing this, the logic is able to issue firing pulses to the pen at the correct time.

Development Methodology

The HP DeskJet 820C was developed under some very tight time-to-market constraints. These constraints dictated that the latest CAE tools be used to speed the development of the ASIC. Additionally, the project team wished to have concurrent design of the hardware and the firmware. This meant that the firmware team needed a platform on which to do development before the ASIC was finished. To meet this need, Aptix hardware emulators were used.

The HP DeskJet 820C ASIC was designed entirely in the Verilog Hardware Description Language (HDL). HDLs are computer languages used to describe digital circuits. They contain constructs that allow designers to describe the function of a circuit rather than the exact gates that are necessary to implement that function. Thus, HDLs allow designers to work at a higher level of abstraction than in the past. Once a designer has written the HDL for a circuit, a compiler program can synthesize the HDL into a gate-level design. By working at a higher level of abstraction, engineers can greatly increase their productivity. The time required to do the design of the HP DeskJet 820C ASIC was significantly decreased over past products.

Since designing an ASIC using an HDL is analogous to writing a piece of software, it is not surprising that many of the practices used by software engineers can be used successfully by hardware teams using HDLs. At the beginning of the project, coding conventions were established. Similar structures in different designers' modules were coded similarly. Designers were encouraged to comment their code liberally. Code reviews were held during the project to find errors and to improve designers' coding practices. These techniques allowed designers to look at each other's code and quickly understand it. In addition to having obvious benefits for the HP DeskJet 820C project, the good coding practices will allow the HP DeskJet 820C hardware to be easily leveraged into future products.

To synthesize the Verilog code into standard cell gates, Synopsys software was used. This software allows the designer to enter information about the design to help the software produce an optimum implementation. Synopsys-specific scripts were used to enter the required information. By using scripts, the designers were able to make changes to the code, and with minimum effort, synthesize the new implementation. Just as for the original Verilog code, conventions and templates for the scripts were developed. As a result of these techniques, in a few special cases engineers were able to modify code written by a different designer and synthesize new hardware very efficiently.

An important part of ASIC design is test development. The HP DeskJet 820C ASIC design team used an HP proprietary technology that allowed the engineers to write test vectors directly in Verilog. These test vectors were used to verify that the functionality of the synthesized design matched the original Verilog. The same vectors were then translated into a format the ASIC tester understood, and used to test the finished silicon. Using this technique, a single set of test vectors was used throughout the project. In addition to the functional test vectors, scan testing was used to achieve the desired fault coverage. Since the insertion of scan hardware and the creation of scan test vectors is done semiautomatically, scan testing was successfully added to the ASIC without incurring a schedule delay. The use of scan testing did increase the cost of the chip because the scan circuitry caused the chip size to grow, but this was deemed acceptable when traded off against the time it would have taken the designers to write functional test vectors with adequate coverage.

Hardware/Software Codesign

To meet the overall project goal of a low-cost product, it was necessary to make careful trade-offs between the hardware and firmware in the product. Functions that are realized in hardware cost money because silicon real estate is used. Functions realized in firmware cost money because they require bits in memory. Since the ROM that holds the firmware in the HP DeskJet 820C is integrated into the system ASIC, its size had a hard upper limit. Also, the HP DeskJet 820C uses a relatively low-power processor, so the processing bandwidth available to perform functions in real time is limited. With the standard cell logic, the processor, and the firmware ROM all integrated onto the same chip, optimal trade-offs between the three were especially important.

To make the correct trade-offs, the ASIC engineers responsible for particular hardware blocks coordinated closely with the firmware engineers responsible for the corresponding firmware blocks. This process allowed the hardware engineers to gain insight into how the firmware would use the block, and at the same time allowed the firmware engineers to have a good understanding of the hardware. This mutual understanding led to better trade-offs. The hardware was designed with just enough functionality to allow the firmware designer to implement the code within the product code size and processor bandwidth constraints, but without a lot of extra hardware thrown in “just in case.” A secondary benefit of this approach was that the firmware engineers were able to write the code for blocks designed this way with few problems. Code for blocks not designed using this process (primarily blocks leveraged from previous products) proved much more problematic to bring up.

ASIC Emulation

To meet the aggressive schedule, it was necessary for the firmware team to begin implementing the code well before ASICs were available to run the code. In fact, firmware implementation began before the ASIC design was even complete. To allow this activity to take place, it was necessary to set up an emulation environment. Traditional methods of doing such emulation include building a custom printed circuit board populated with one-time-programmable devices (anti-fuse devices, laser programmed parts, etc.) and software-based emulation using a previous product. Since the digital architecture for this ASIC was a significant departure from any previous product, emulation on a previous product would have been difficult at best and would have required a significant code port when the ASIC became available. One-time-programmable devices were not flexible enough to support emulation before the design was functionally complete. For these reasons, the product team chose to use full-chip IC emulators from Aptix to support firmware development before ASICs were available.

IC emulators are essentially a large array of SRAM-based FPGAs (field programmable gate arrays) with programmable interconnect between them. Software reads in a gate-level design for an IC, partitions it into the FPGAs, and then creates all the necessary files to program the FPGAs and the interconnect between them. The emulator then is functionally equivalent to the IC that will be fabricated. The emulators are highly flexible since they can be reprogrammed by simply downloading a new pattern into the SRAMs. The main drawback of the IC emulators is that they generally are not able to run at the same speed as the final silicon. For this product, the emulator ran at one fourth of the final clock speed.

Using this approach, the IC team was able to provide the firmware team with usable hardware approximately four months before silicon arrived. Since the ASIC design was not complete at that point, the first hardware provided was only a subset of the full standard cell logic. What was provided was enough for the firmware team to begin writing and testing the operating system, the code that needed to be written first. As more blocks in the IC were completed, they were incorporated into the emulation system.

In addition to providing early hardware to the firmware team for development purposes, the use of IC emulators allowed the hardware to be verified in the full printing system with actual firmware before being committed to silicon. Because the emulators ran at close to the system speed, several orders of magnitude more clocks cycles of verification occured on the emulators than with software simulation. Also, since it was real firmware running, the ASIC was put in states that would have been difficult or impossible to achieve in simulation because of the complexity of getting into that state. Finally, many

unanticipated hardware/firmware interactions were discovered. The team was eventually able to print with the IC emulators, giving very high confidence in the functional correctness of the ASIC.

Thanks to the emulators, two problems in the design were discovered and fixed before committing the design to silicon. Both problems were system interaction issues that would have been very difficult to discover through simulation alone. When silicon arrived, firmware was almost immediately bootable on it. The only things that needed to be changed in the firmware were things that were affected by the difference in clock speed, and these had been deliberately coded to be easy to change.

Regulatory Requirements

Because the HP DeskJet 820C printer is sold in the consumer marketplace it must meet all applicable consumer electronic regulations. Of particular interest to the electronic design of the product are the electromagnetic interference (EMI) and electrostatic discharge (ESD) requirements. EMI occurs when an electronic product creates an electric field and interferes with the correct operation of another electronic product. ESD occurs when an object (generally a human) that has built up a large static charge discharges to a second object (for example, an IC). In addition to government requirements on a product's level of EMI and sensitivity to ESD, HP maintains internal standards, which are generally tougher than the government standards. Meeting or exceeding these internal standards on every product is an important aspect of HP's reputation for high-quality, reliable products. Since the HP DeskJet 820C could not legally be released without meeting government regulations on EMI and ESD, failure to meet them was a significant schedule risk to the product. Therefore, both were addressed early during the design of the digital ASIC.

In general, EMI results from improperly controlled high-frequency signals that travel a long distance, particularly signals that travel over cables. The trick in designing for reduced EMI is to control the signals with high-frequency content to the greatest possible extent. All I/O pads in the HP DeskJet 820C make use of an HP proprietary technology that compensates for process, voltage, and temperature (PVT) variations in the operating environment of the chip. The compensation ensures that the pads have nearly the same slew rate regardless of the PVT environment. (Typically, parts in an environment that causes the chip to run fast have about twice the slew rate of parts in an environment that causes the chip to run slow). Of particular concern in this product were the digital signals that travel between the main logic board and the carriage board. These signals, which are in the megahertz range, travel approximately 19 inches along an unshielded and untwisted flex cable. This was deemed the most EMI-prone piece of the design. The I/O pads that drove these signals were designed to have as slow a slew rate as the signal speed would allow. As a backup system, phase-locked loop hardware and additional logic to dither the system clock or the signals on the flex cable was designed into the ASIC. The result of this design effort was successful passing of the EMI regulations on schedule.

The other big regulatory threat, ESD, was recognized early in the project by both the designers and the ASIC vendor (ICBD, a division of HP). Because ESD events can cause damage that will not immediately destroy the chip but instead will cause it to fail months or even years later, inadequate ESD protection can result in product reliability and customer satisfaction problems down the road. The chip needed to be able to withstand ESD events both before and after being put on a printed circuit board. When the chip is on a printed circuit board, external components can be placed around the chip to help protect it. However, each component adds cost to the product, so integrating protection into the chip results in a cost savings. Also, although the chips will be in a relatively controlled environment before being put onto the printed circuit board, ESD events can and will occur, so the chip needs to be able to withstand them.

When an ESD event occurs at the chip pins, a large current between the discharge point and ground is induced in the ASIC. Most structures internal to the ASIC cannot withstand such a large current without damage. Therefore, a chip designed to withstand ESD has structures that can withstand the high current, and routes the current to these structures rather than to the chip internals. Since an ASIC's only contact to the outside world is through its pads, ESD protection devices are generally located at the pads. During the design of the ASIC, the ASIC vendor assigned a dedicated engineer to evaluate the ESD design. A current-limiting resistor and a reverse-biased diode were used in each pad to limit the current that can reach the chip internals. Additionally, shunt structures between V_{dd} and ground were carefully designed and positioned in the IC. This early involvement by the vendor resulted in a solid design that meets HP ESD requirements.

Conclusion

By taking into account the HP DeskJet 820C's overall project goals of low cost and time to market from the start, a well-optimized digital controller for the printer was delivered on schedule. The chip is specifically designed for HP's Printing Performance Architecture. By integrating all digital functions in the printer on a single piece of silicon, the cost of the electronics in the product was greatly reduced over the previous-generation product while maintaining or increasing performance. The design team used the latest ASIC development tools to deliver a correctly functioning ASIC on a very tight schedule. Through the use of hardware emulators, the firmware team was able to begin coding before the final chip design had been released for manufacturing, further speeding the printer's design. All EMI and ESD requirements for the product were met on schedule. A lithograph of the final chip silicon is shown in Fig. 5. The chip area is approximately 81 mm^2.

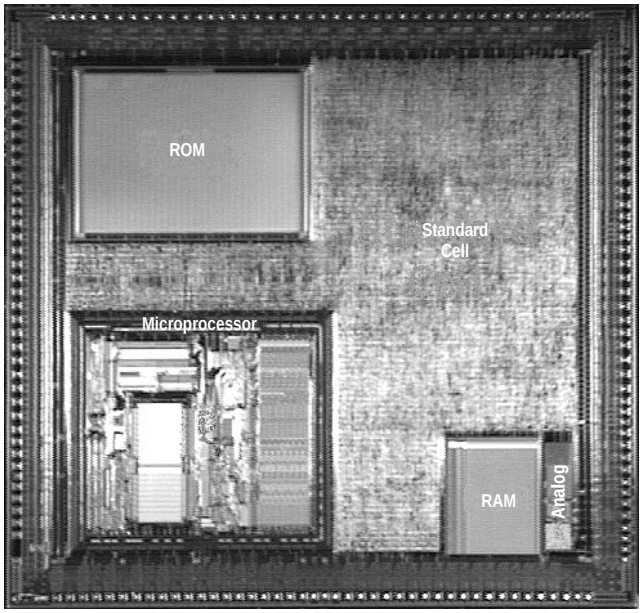

Fig. 5. *HP DeskJet 820C digital controller ASIC.*

Acknowledgments

The authors would like to acknowledge Tom Pritchard and Mark Thackray for their contributions to the ASIC. They would also like to thank the entire team at the HP Integrated Circuit Business Division for their design, development, and manufacturing contributions.

▶ Go to Next Article
▶ Go to Journal Home Page