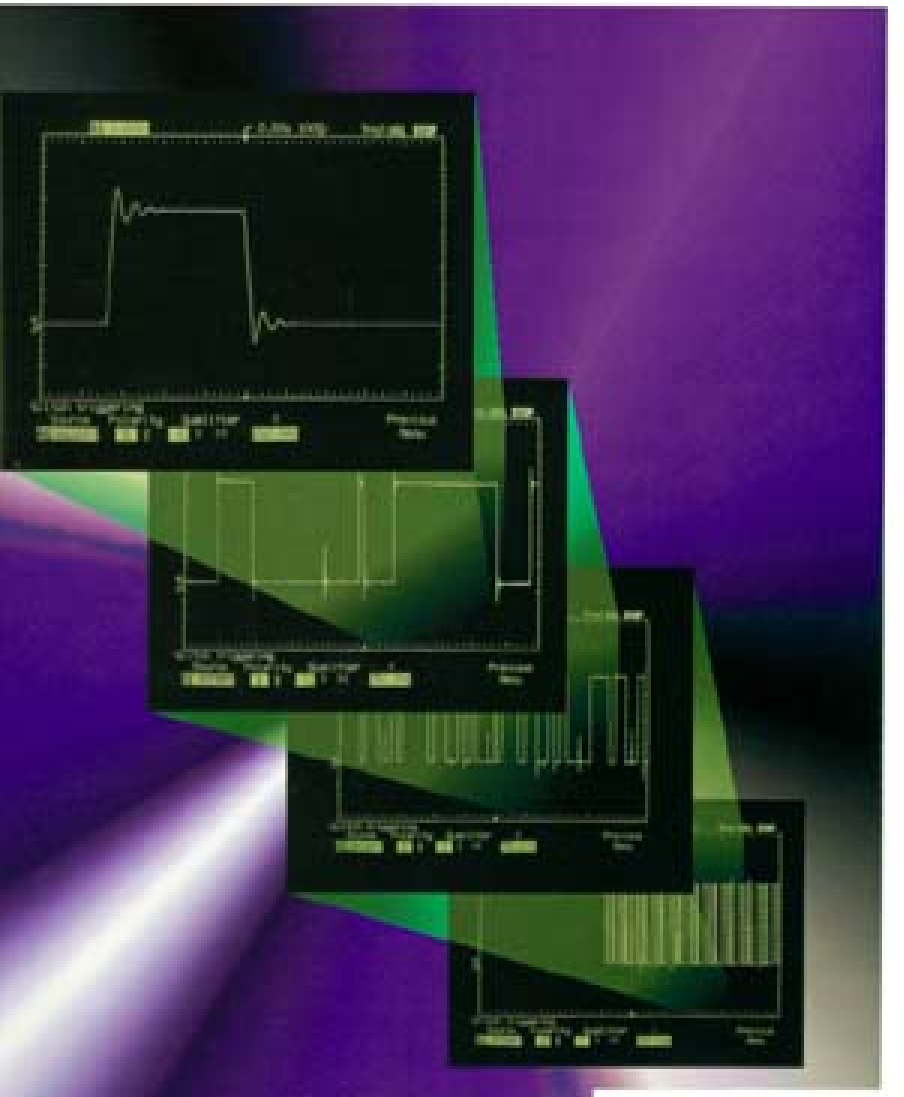
HEWLETT-PACKARD
JOURNAL

April 1987



HEWLETT PACKARD

table of contents

April 1997,
Volume 48, Issue 2

Articles

A Family of Instruments for Testing Mixed-Signal Circuits and Systems

This entirely new product category combines elements of oscilloscopes and logic analyzers, but unlike previous combination products, these are "oscilloscope first" and logic analysis is the add-on.

by Robert A. Witte

Electronic circuits have been a part of modern life for so long that most people take them for granted. Some devices are inherently electronic in nature such as telephones, radio receivers, calculators, and personal computers. Other devices started out as mostly mechanical and have gradually had more electronics incorporated into them over time. Electronics has made its way into automobiles, cameras, water heaters, home appliances, elevators, thermometers and weighing scales. This "electronics everywhere" trend has gradually filled modern life so that it is difficult to imagine what life would be like without all those electrons running around doing their jobs.

Most early electronic systems were analog, especially if they interfaced to real-world physical phenomena. The emergence of digital technology resulted in the gradual diffusion of digital gates into applications that were once exclusively analog. Thus, an "analog moving to digital" trend emerged as the number of digital gates per acre of silicon continued to grow at a fast rate. Analog circuits will never be totally replaced, since for the most part the real world stubbornly retains its analog behavior and circuits that interface to this world must retain at least some analog circuitry. The result is that many electronic systems are mixtures of analog and digital circuitry that have come to be known as "mixed analog and digital" or simply "mixed-signal."

The single-chip microcontroller has emerged as an important component in these mixed-signal designs. Of course, microcontrollers have been around for decades, doing the lowly control tasks in cost-sensitive applications while their more powerful siblings ("real" microprocessors such as an Intel80486 or a Pentium®) got all of the attention, usually because of their critical role in personal computers. Meanwhile, the single-chip microcontroller improved in performance, moving from 4 bits to 8 bits and 16 bits while also improving in cost-effectiveness. Without much fanfare, these single-chip devices found their way into a wide range of designs, causing one author to refer to them as "The Ultimate ASIC."[1] These devices are often used to control or measure a physical system (e.g., antilock braking, camera control, appliance controls, industrial control systems). A generic block diagram for such a system is shown in Fig. 1, and an example of a mixed-signal single-chip microcontroller is presented ***Subarticle 1a***.

This increased use of mixed-signal electronics is showing up in a wide variety of industries and applications. Consumer electronics is an obvious area, with mixed-signal designs being used in CD players, stereo receivers, tape decks, and camera electronics. Similarly, communications devices such as modems, telephone answering machines, and multimedia boards for PCs all use mixed-signal electronics. There are many applications in industrial electronics, including process control, industrial water heater controls and other sensor-based systems. The growing area of mechatronics (the merger of mechanical and electronic technology) is primarily mixed-signal in nature. A large and growing mixed-signal area is automotive electronics, including subsystems such as the previously mentioned antilock braking systems and ignition control. Biomedical applications are another emerging area, with mixed-signal electronics being applied in pacemakers, hearing aids, and other medical devices.

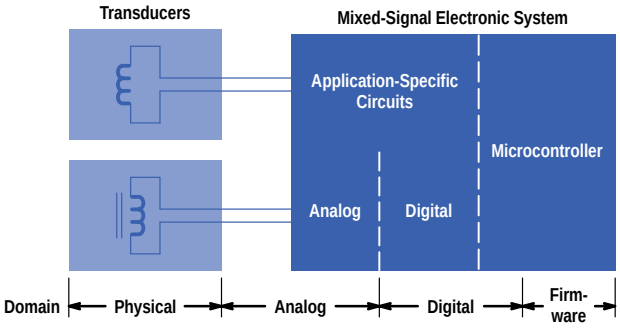


Fig. 1. *Block diagram of a generic mixed-signal system.*

Systems can be totally digital if only on-off control is required. More likely, there is some physical quantity being measured or controlled that requires a portion of the electronic system to be analog. The increased use of sensors allows engineers to create smarter control systems that monitor physical events and do a better job of controlling the system. A good example of this is antilock braking systems in automobiles. In this case, electronics (including sensor technology) is being used to make the braking effectiveness of the automobile far better than would be possible in a purely mechanical system.

Oscilloscopes

For designers of mixed-signal systems, the troubleshooting tool of choice is the oscilloscope. The versatility of the oscilloscope for viewing a wide variety of waveforms allows the design engineer to see what's happening in the device under test. Other test instruments may also be used but the oscilloscope remains the first tool that most users turn to for debugging circuits.

Mixed-signal engineers expect many things from their oscilloscopes. The oscilloscope must have sufficient fundamental performance in terms of bandwidth and sample rate to capture and display signals accurately and reliably. At the same time, the oscilloscope must be easy to use so that the engineer can focus on the operation of the circuit and not on the operation of the oscilloscope. In the heat of the troubleshooting battle, it is very distracting for the user to have to interrupt the debug process to deal with an uncooperative test instrument.

With the increasing digital content in these mixed-signal systems, a two-channel or even a four-channel oscilloscope quickly runs out of inputs. It is just not possible to view the complete operation of even a simple digital circuit with a conventional oscilloscope. Take the trivial example of a 4-bit up/down counter. With four output lines, a clock input, a load input, and an up/down count control, the counter will have at least seven digital signals associated with it (and perhaps more). Since an engineer cannot view all of these signals at once with an oscilloscope, the usual remedy is to apply "mental storage." The engineer applies the oscilloscope probe to a particular signal, views it, and stores the image into mental memory, then moves to other signals of interest and repeats the process. Eventually, a picture of the entire circuit's operation is formed in the mind of the debug artist. Sometimes this picture is precise and meaningful but sometimes it is cloudy and full of uncertainty.

Logic Analyzers

The idea behind a logic analyzer is that many digital signals can be viewed simultaneously so that an engineer can obtain a complete and accurate picture of what is really going on in a circuit. One of the trade-offs made in a logic analyzer is that only digital signals can be viewed and only an idealized reconstruction of the waveform is possible. That is, the waveform is treated as a purely binary signal and is displayed as either a logic high or a logic low, with no detail of the actual waveform shape. This is a reasonable compromise since the major problem in a digital system is determining whether the circuit operation is correct. It is a functional check that does not require the waveform detail. In fact, it can be argued that complete waveform detail is mostly visual noise when checking the functionality of a digital circuit.

As logic analyzers evolved, they tended to be optimized for solving the really tough problems that come with complex, bus-based microprocessor systems. Most modern logic analyzers have a minimum of 32 channels and may have 128 channels or more. They also provide extensive trigger and selective storage features that make them unmatched in debugging and troubleshooting power. This power inherently leads to a certain level of complexity in the analyzer; complexity that can be a barrier to the oscilloscope-oriented engineer. Many designers of mixed-signal systems, even when limited by the number of channels on their oscilloscopes, remain reluctant to adopt logic analyzers.

Mixed-Signal Oscilloscopes

Further investigation into this avoidance of logic analyzers revealed the opportunity for a new kind of test instrument that would expand the channel count of the oscilloscope without losing its inherent appeal. A series of market research activities were launched to determine how to fill this gap between oscilloscope and logic analyzer. Clearly, engineers were limited by the number of channels on their oscilloscopes but they were not always adopting a logic analyzer as the solution. What eventually emerged is an entirely new product category that combines elements of oscilloscopes and logic analyzers.

As a world leader in logic analyzer products, HP had already pioneered the creation of hybrid instruments that combine oscilloscope and logic analyzer technology (HP 1660 Series and HP 16500 Series logic analyzers, for example).[2] However, these products were designed with full-featured logic analysis as their top priority and the oscilloscope capability is an adjunct to the analyzer. The new product line starts with the oscilloscope user in mind. These products are "oscilloscope first" and logic analysis is the add-on.

A family of HP products aimed at mixed-signal testing has been created. The HP 54600 product line now offers the following set of test instruments appropriate for mixed-signal work:

- The HP 54645D mixed-signal oscilloscope (Fig. 2) combines two 100-MHz oscilloscope channels with 16 logic timing channels. This is the ultimate mixed-signal testing tool because it seamlessly integrates oscilloscope channels and logic timing channels. This product includes HP's MegaZoom technology, which delivers the benefits of a million-sample memory for each channel without the agonizingly slow responsiveness normally associated with deep memory.

(a) (b)

Fig. 2. *The HP 54645A two-channel oscilloscope (a) and the HP 54645D mixed-signal oscilloscope (b), both with MegaZoom, are examples of the new product category of mixed-signal oscilloscopes.*

- The HP 54645A two-channel oscilloscope with MegaZoom (Fig. 2), a general-purpose oscilloscope, is the first affordable 100-MHz digital oscilloscope with sufficient sample rate and memory depth to capture a wide range of analog and digital signals. The HP 54645A has the same basic MegaZoom technology as the D version but does not have the logic channels.
- The HP 54620A/C 16-channel logic timing analyzer is an easy-to-use logic analyzer designed to be the perfect companion product to an oscilloscope, perhaps the one that the user already owns. This logic analyzer has an oscilloscope user interface metaphor so that it is familiar and easy-to-use for the typical oscilloscope user.
- The HP 54615B and 54616B/C 500-MHz two-channel oscilloscopes are general-purpose oscilloscopes that provide the bandwidth and sample rate (1 GSa/s and 2 GSa/s, respectively) needed for high-speed digital work while maintaining the responsiveness and trustworthiness required for mixed-signal debugging and troubleshooting. The HP 54616C has a flat-panel color display.

While each of these products represents a particular mix of features and performance, they all share the core values of the HP 54600 product line:

- Direct Access Controls on Main Functions. The volts/division, time/division, position, and trigger level controls are all dedicated knobs for easy access.
- Responsive User Interface. Instant response to front-panel changes is critical to making an oscilloscope easy to use. All of these products provide sufficient processing power to maintain a responsive front panel.
- Fast Update Rate and Low Dead Time. These products use multiprocessor architectures to make sure the display is updated quickly.
- Digital Peak Detection. These products have true digital peak detection, which is available on all sweep speeds. Digital peak detection keeps the sample rate operating at its maximum specification even on very slow time base settings, guaranteeing that short-duration events (such as glitches) are not missed (see ***Article 4***). Peak detection also prevents aliasing. On the HP 54620A/C logic analyzers, the equivalent feature is glitch detection.
- HP Proprietary Alias-Reduction Techniques. All of the HP 54600 Series oscilloscopes employ HP's proprietary alias-reduction techniques to reduce the possibility of an erroneous displayed waveform (see ***Article 5***).
- Compact and Portable. These products pack a lot of measurement power into a compact size and the small footprint helps save precious bench space.
- Optional HP-IB, RS-232, and Parallel Interfaces. These products offer a choice of computer and printer interfaces.
- Optional Measurement/Storage Module. This option provides FFT, advanced math, and pass/fail mask testing (oscilloscope products only).

The accompanying articles in this issue describe some of the key design contributions incorporated in these products.

References

1. “Single-Chip Microcontrollers: The Ultimate ASIC,” *Electronic Engineering Times*, March 20, 1995.
2. *1996 HP Test and Measurement Catalog*, Hewlett-Packard Company.

Mixed-Signal Microcontroller

As explained in the accompanying article, HP's new family of mixed-signal test instruments is designed to meet the needs of designers of products that are partly analog and partly digital, such as antilock braking systems, camera control systems, appliance controls, and industrial control systems. Many of these products are based on single-chip microcontrollers. The producers of these products are demanding simpler and cheaper electronic assemblies from system developers, and this pressure to reduce costs is fed back to microcontroller suppliers in a number of ways, most of which can be summarized as a greater integration of mixed-signal peripherals with the microcontroller core. Thus, the microcontrollers themselves are becoming mixed-signal devices.

For example, Microchip Corporation's PIC14000 microcontroller integrates a number of peripherals that are often requested by system designers. This peripheral set, which is well-suited for slow-moving real-world analog signals, is packaged with Microchip's 14-bit microcontroller core to create the PIC14000. The peripherals are:

- Single-slope analog-to-digital converter (ADC)
  - 16-bit programmable timer with capture register
  - 16-ms maximum conversion time at maximum resolution with 4-MHz clock
  - 4-bit programmable current source
  - 8 external channels, two with selectable level-shift inputs
  - 6 internal channels
- On-chip temperature sensor
- Two comparators with programmable references
- Internal bandgap voltage reference
- Voltage regulator control output
- On-chip low-voltage detector.

Wes Reid
Senior Applications Engineer
Standard Microcontroller & ASSP Division
Microchip Corporation

Testing a Mixed-Signal Design Based on a Single-Chip Microcontroller

The HP 54645D mixed-signal oscilloscope simplifies the testing and debugging of microcontroller-based mixed-signal designs with its integrated analog and digital channels.

by Jerald B. Murphy

This article presents an example that shows how the HP 54645D mixed-signal oscilloscope (see ***Article 1***) might be applied to understand the operation of a device that produces a 60-Hz reference signal that is used by other devices in a system. The reference sine wave is produced by low-pass filtering a pulse width modulated signal generated by a PIC16C74 8-bit CMOS single-chip microcontroller (see block diagram, Fig. 1). This is a fairly typical application of these powerful, low-cost devices. The low cost and availability of single-chip microcontrollers has resulted in their application in areas that previously were nonelectronic, and in some cases even mechanical. Examples include the replacement of clockwork-driven appliance switches and motor control.

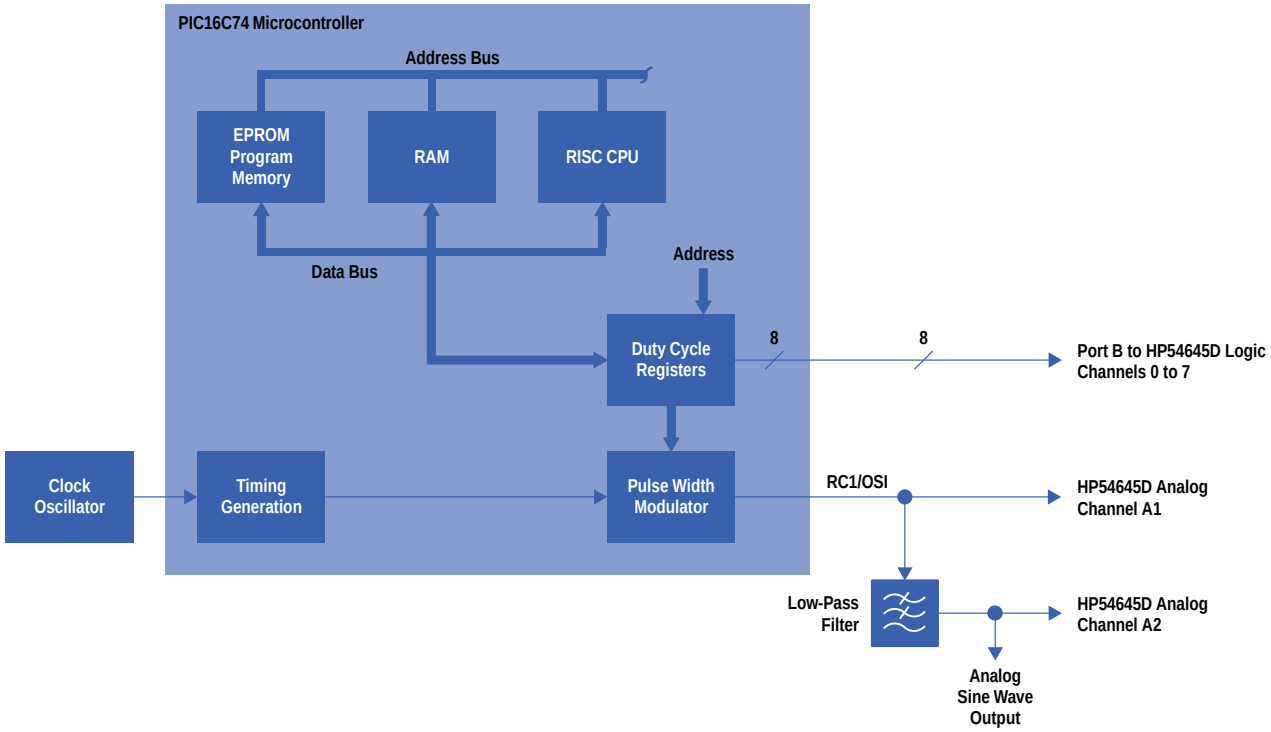


Fig. 1. *Block diagram of the PIC16C74 microcontroller and low-pass filter implementing a reference sine wave generator, showing the hookup to the HP 54645D mixed-signal oscilloscope.*

In this application we have a very stable signal being produced with a simple design. An analog design with the same level of performance would have been much more complex.

The pulse width modulator (PWM) of the PIC16C74 microcontroller is controlled by an 8-bit input. This input is provided by a lookup table stored in the microcontroller's memory. In operation the microcontroller sends the first control input to the PWM from the lookup table, waits a specified time, and then sends the next value in the lookup table to the PWM. When the last value in the lookup table is sent to the PWM, the microcontroller's counter is reset to start the process again at the top of the table. The input to the pulse width modulator is available on the microcontroller's port B RB0-RB7 data lines.

In this application we can view the PWM's output and the 8-bit data that produced that output. This will enable us to verify the correct operation of the PWM.

The first step in testing this design is to verify the operation of the PWM. We want to make sure that the minimum pulse width is correct and the maximum pulse width is less than a period of the system's basic timing. The minimum pulse width is found by using the HP 54645D's glitch triggering mode. We simply use one of the analog channels to probe the PWM and then select the "less than" mode of the glitch trigger to find the minimum pulse width.

The minimum pulse width should be approximately 1 μs, so we set that as the search condition in the glitch trigger. Now, decreasing this value until the mixed-signal oscilloscope stops triggering (which happens immediately) isolates the minimum pulse width (Fig. 2). Applying the mixed-signal oscilloscope's automatic measurement feature gives a measurement of this pulse width as 1.02 μs. The maximum pulse width can be located in the oscilloscope's deep memory by delaying the sweep by 8.3 ms (half the period of the 60-Hz signal). Here we observe a pulse that is 256 μs wide with clean transitions on either side. By probing the microcontroller's port B data lines we can observe the input and output of the PWM. The deep memory of the mixed-signal oscilloscope greatly simplifies the task of observing this operation. Simply pressing the **Autoscale** key results in a display that contains both the analog channel displaying the PWM output and the eight logic channels displaying the PWM's input. The resulting display is not well triggered but the waveform data needed to observe the PWM operation is captured in the oscilloscope's deep memory. Simply pressing the **Run/Stop** key freezes the display (Fig. 3) so that the contents of the oscilloscope's deep memory can be examined by panning and zooming, using the oscilloscope's time base controls.We see that the minimum and maximum values of the pulse width are related to the 00 and FF values of the digital inputs. The display is zoomed in on a leading edge of one of the PWM pulses and the delay from the digital input to the PWM output is measured to be 8.2 μs (Fig. 4). We can conclude that the PWM is operating properly.

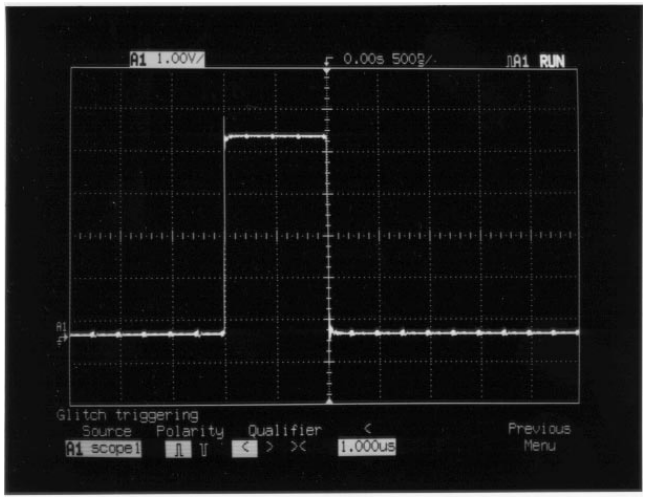


Fig. 2. *Glitch triggering is used to locate the minimum pulse width of the pulse width modulator of the PIC16C74 microcontroller.*

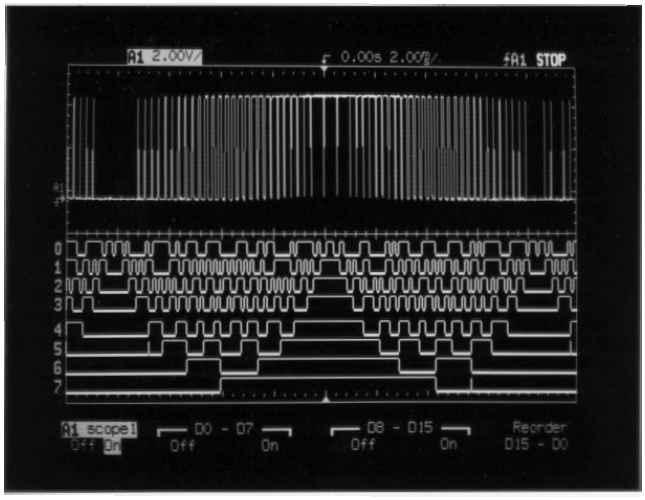


Fig. 3. *A mixed-signal display of the pulse width modulator output on channel A1 and the digital control lines 0-7 (RB0 to RB7 in Fig. 1).*

The next set of questions relate to the low-pass filter. One question might be, "What is the delay from the digital input of the PWM to the output of the low pass filter?" This measurement is easily accomplished with the HP mixed-signal oscilloscope. Connect a second oscilloscope probe to the output of the filter and press **Autoscale**. The resulting mixed-signal display contains the digital input to the PWM, the variable-width pulse output of the PWM, and the output of the low-pass filter. The oscilloscope is now triggered on the rising edge of the sine wave output of the low-pass filter. The total operation of the system under test is now displayed on one screen (Fig. 5).

The delay from the digital input of the PWM to the output of the low-pass filter can be measured by first placing a cursor on a known value of the digital input lines. After selecting the hexadecimal cursor readout mode, the t1 cursor can be placed at the point where the digital lines first reach the value of FF (Fig. 6). The t2 cursor is then placed at the maximum value of the sine wave. Switching the cursor readout to time, this delay is measured to be 2.610 ms (Fig. 7). This measurement was made without having to take into consideration any additional delays that might be associated with the instrumentation. The deep memory of this product allows this measurement to be made on data that was gathered on one trigger. There was no need to be concerned about errors that might be produced by having the display made up of many cycles of system operation.

In this example we have seen how the HP 54645D mixed-signal oscilloscope simplifies the testing and debugging of microcontroller-based mixed-signal designs with its integrated analog and digital channels. The user of the HP mixed-signal oscilloscope has the advantage of MegaZoom technology's high-speed display, instantly responding controls, and deep memory to simplify the testing phase of the project. The integrated analog and digital channels produce an easy-to-understand view of the device under test. The glitch triggering was used to isolate a point of interest. In this example there was little need to use any of the mixed-signal oscilloscope's pattern or advanced pattern triggering capabilities. These

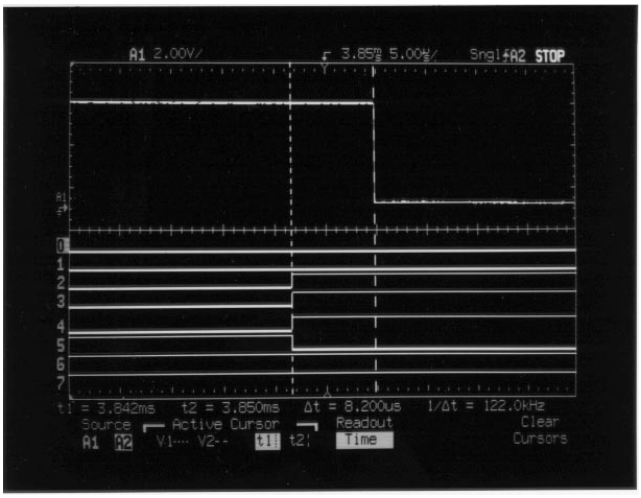


Fig. 4. *A mixed-signal measurement. The delay from digital control to pulse width output is measured to be 8.2 μs.*

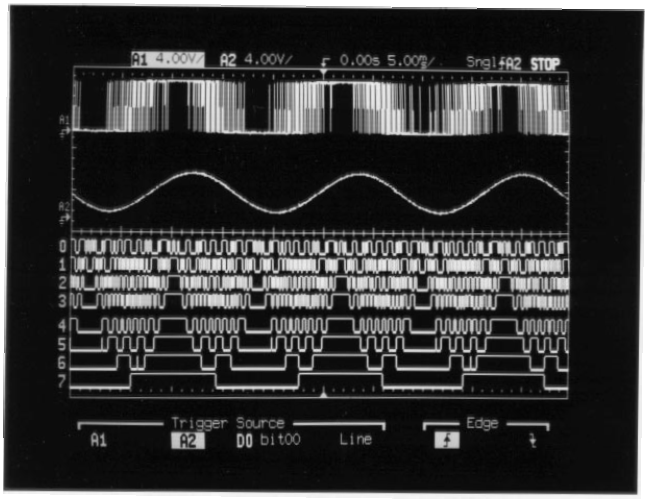


Fig. 5. *A view of the total operation of this mixed-signal system. The pulse width modulator output is channel A1, the output of the low-pass filter is channel A2, and digital channels 0-7 are the pulse width modulator control lines.*

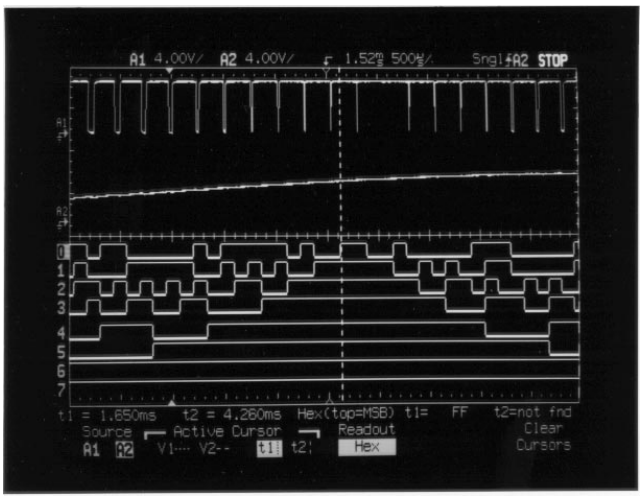


Fig. 6. *Cursor t1 is placed at the point where the digital control lines enter the state FF.*

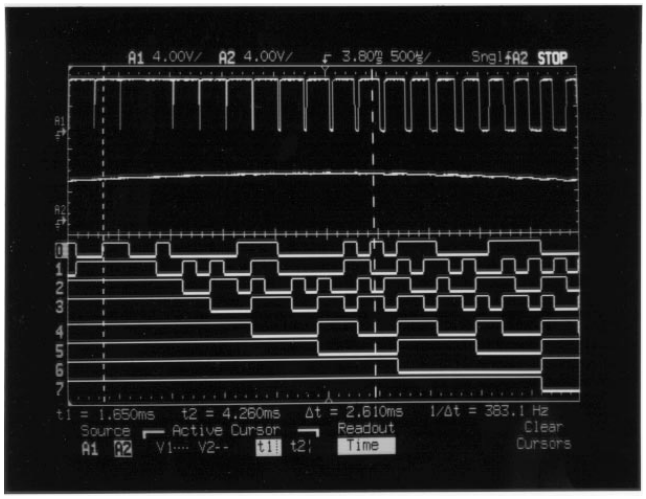


Fig. 7. *With cursor t2 placed on the peak of the low-pass filter output and the cursor readout changed to time, the delay from digital command to output is measured to be 2.610 ms.*

features provide even more debugging power when needed. Both cursor and automatic measurements were used to characterize the device under test.

Acknowledgments

Thanks to Art White, applications consultant for Western Regional Marketing, for developing this application based on the PICDEM-2 model DI163002 demonstration board. The PIC16C741 and PICDEM-2 are products of Microchip Technologies, Inc. 2355 West Chandler Blvd., Chandler, Arizona 85224-6166 USA. Western Regional Marketing, Inc. of Westminster, Colorado is the distributor for Microchip products serving the geographic area where the author is located. Special thanks to Mary Kerr, a former HP employee and now a consultant, for leading the effort to develop this application for Hewlett-Packard. Johnnie Hancock managed the relationship with Microchip. Additional help was received from Steve Warntjes and Bob Witte.

Design of a Mixed-Signal Oscilloscope

This combination of a digital oscilloscope and a logic timing analyzer provides powerful cross-domain triggering capabilities for capturing signals in mixed-signal environments. MegaZoom technology, consisting of advanced acquisition techniques and dedicated signal processing, maintains display responsiveness while making optimal use of deep sample memory.

by Matthew S. Holcomb, Stuart O. Hall, Warren S. Tustin, Patrick J. Burkart, and Steven D. Roach

The design of the HP 54645A/D oscilloscopes introduced in ***Article 1*** began with the HP 54645A, envisioned as a performance upgrade to the HP 54600 Series oscilloscopes.[1] These oscilloscopes, introduced in 1991, have an almost analog look and feel. Their three-processor design yields unprecedented display update rate and responsiveness at an affordable price. The major design goal for the HP 54645A was to improve the sample rate performance by an order of magnitude while maintaining the responsiveness and display update rate of the existing HP 54600 Series products.

Ultimately, two new products were created: the HP 54645A two-channel oscilloscope and the HP 54645D mixed-signal oscilloscope. The mixed-signal oscilloscope is a new product category that adds 16 logic timing analyzer inputs to the two channels of oscilloscope input. In addition to displaying all 16 logic channels, the HP 54645D provides advanced logic triggering functions on patterns that can span all 18 channels.

The HP 54645A and 54645D were designed concurrently. We made every effort to have the oscilloscope-only product (the HP54645A) be simply the combination product (the HP54645D), with an external trigger circuit substituted for the digital channel subsystem. Even the firmware ROM in the two products is identical.

Architecture

We started the design by modifying the architecture. A simplified block diagram of an HP 54600 Series oscilloscope is shown in Fig. 1. Two ICs form the core of the system: the acquisition processor and the display processor. The display system was left alone and only the acquisition circuitry was redesigned. We kept the same display, the same package, the same analog front end, and the same host 68000 processor.

In the original HP 54600 design, the acquisition processor IC was responsible for taking the digitized samples from the analog-to-digital converter (ADC) and placing them into an external memory as time-ordered pairs, which the display processor IC placed on the CRT display. More specifically, the acquisition tasks include:

- Generation of the sample clocks for the ADCs and controlling the decimation of the sample clock and dither algorithms for use at slower time base settings.
- Peak detection of the ADC data. This block calculates the minimum and maximum values of the sampled data.
- Intermediate storage of the ADC data (or minimum/maximum pairs if the peak-detector is used) into an internal circular 2K-sample memory. This memory, known as the *capture RAM*, holds the data until the trigger point is calculated.
- Accepting the analog trigger from one of the channels and measuring the time between the trigger and one of the sample clock edges.
- After the trigger is found, unloading the data from the capture RAM into an external RAM known as the *waveform RAM*. For each sample value, a corresponding x-axis (time) value is calculated.

All of these tasks were integrated into one chip for the HP 54600 oscilloscopes. For the new products, we divided the above functions into separate, discrete components, as shown in Fig. 2.

Clock Generation. Much of the difficulty of the original one-IC design stemmed from unwanted coupling between the sample clocks and the trigger clocks. In a digital oscilloscope, such coupling can severely corrupt the time base fidelity, causing time-axis nonlinearities ("bowing in time"), time-axis discontinuities, and sample bunching. In a higher-frequency instrument, the design of the clocking and trigger systems would have been all the more difficult. Consequently, the new products have a separate, dedicated bipolar IC for handling the clock generation and triggering.

Peak Detection. As before, we needed a handful of digital circuitry that stands between the ADC and the intermediate capture memory. This circuitry takes the 200-MSa/s 8-bit ADC data, calculates the appropriate minimum and maximum values, and stores the results sequentially into memory.

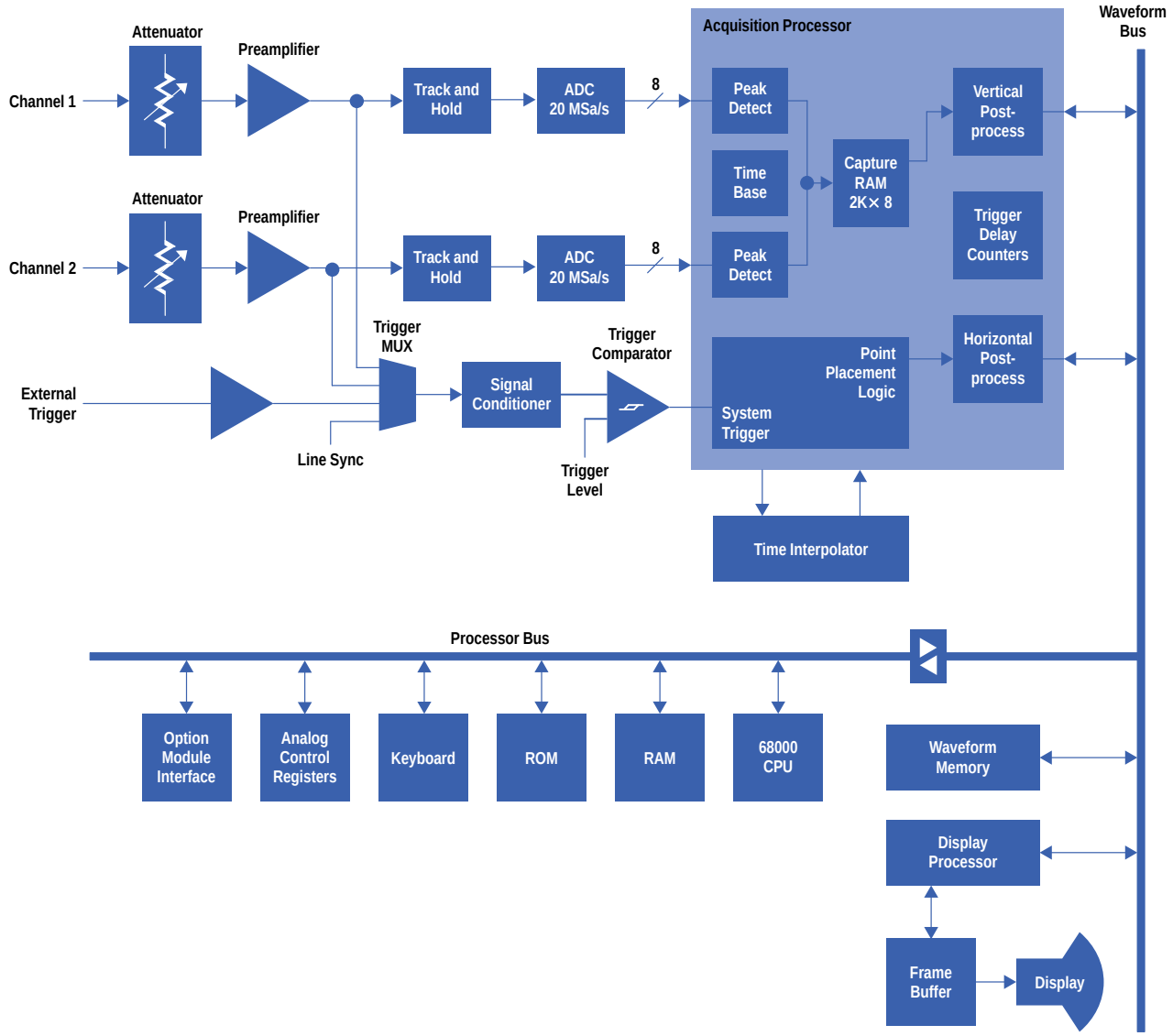


Fig. 1. *Original HP 54600 Series oscilloscope block diagram.*

Additionally, we decided to improve the averaging performance by digitally low-pass filtering the incoming sampled data before storing it into memory. This technique (later named *smoothing*) requires summing 32-bit values at a 200-MHz rate.

We chose to realize these functions in a CMOS gate array known as the *oscilloscope signal processor IC*. In addition to the functions described above, this IC decelerates (or fans out) the ADC data, steering and decimating the sampled data into the capture memory as described next.

Capture Memory. Recall that this memory is used to hold the incoming data until a suitable trigger point has been found and measured. In the HP 54600, the acquisition processor, being a fully custom design, had an internal 2K-byte (16K-bit) memory. Such memories are less available in gate array technology, so combining this function with the oscilloscope signal processor IC wasn't a particularly attractive option.

At 200-MHz rates, or 400-MHz rates for the timing channels, high-speed memories aren't off-the-shelf plug-in items. A traditional ECL design would be power hungry and costly. Our solution was to use commercially available video memories (VRAMs). Since these parts have a 50-MHz 16-bit interface, we could store data at a 100-Mbyte/s rate per memory. Each channel then requires only two devices. Since these memories are dual-ported, we can access the data (for plotting) simultaneously with the acquisition process, an advantage that we capitalized on further, as explained later.

Each VRAM has a capacity of 4M bits, resulting an 8M-bit (or 1M-byte) memory depth per channel. This increased memory depth is a huge improvement over previous designs and delivers value to the customer through increased time capture at a given sample rate (see ***Article 4***).

Time Base and VRAM Controller. Along with the advantages of all of this deep memory came the problems of deep memory. Another CMOS gate array IC was designed for managing the storage and retrieval of data into the VRAM array. Its primary

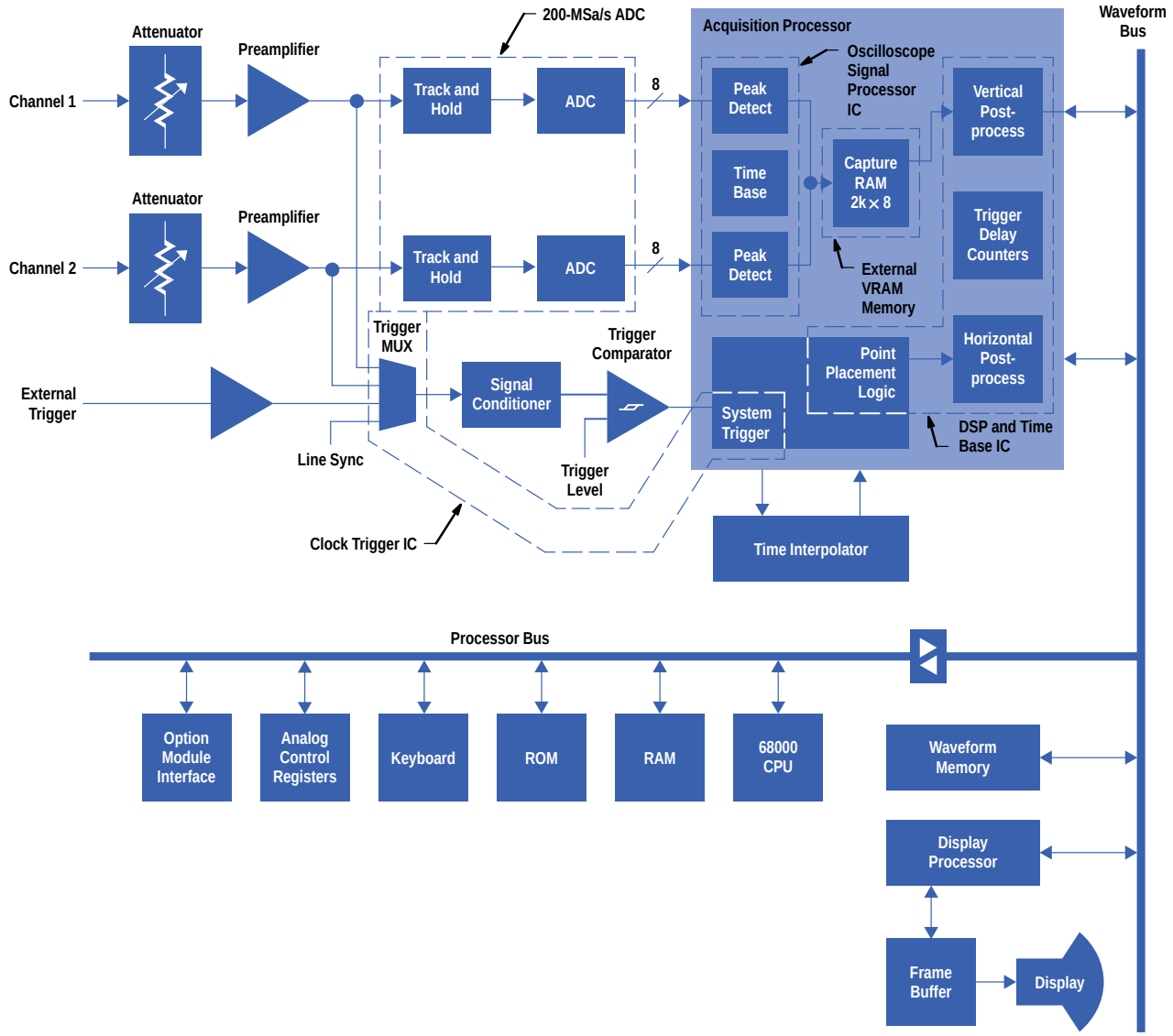


Fig. 2. *HP 54645A oscilloscope architecture compared to the HP 54600. The topology remains relatively unchanged. Only the logic partitioning is different.*

job is to provide the necessary commands to the VRAM to manage the data flowing into and out of its two ports. It also provides a high-speed interface to this memory for the rest of the system at a sustained reading or writing rate of over 10 million bytes per second. It controls the main time base algorithms, the trigger delay and holdoff counters, and a variety of interacquisition and intra-acquisition sample dithering circuits.

Data Processing. In previous products all of the time-axis acquisition algorithms were hard-coded into hardware. Such a system was fast and inexpensive, but more difficult to change. Therefore, for the new oscilloscopes, instead of hard-coding the mathematics into the time base and VRAM controller IC, we opted to use an off-the-shelf digital signal processor, or DSP: Texas Instruments' 25-MHz TMS320C50.

Like its predecessor, the DSP's job is to use the trigger information to determine where each of the samples stored in the VRAMs should be placed on the screen. Additionally, it handles all of the pan-and-zoom operations, allowing the user to redisplay the acquired data at other time base or delay settings after the acquisition is complete.

Mixed-Signal Oscilloscope

The idea behind the HP 54645D mixed-signal oscilloscope is to provide a truly seamless analog and digital waveform viewing instrument. A key goal for the project was to provide a product that was greater than the sum of just the two instruments. By combining an oscilloscope and a timing analyzer, we hoped to provide functionality and ease of use never before available in a laboratory quality oscilloscope.

Careful attention was paid to knitting together the analog and digital halves of the mixed-signal oscilloscope. Anything you can do with the analog channels, you can do with the digital channels: viewing, automatic measurements, triggering, and so on. Both halves share the same time base and the triggering system can operate on all 18 channels.

Where the oscilloscope channels have a *oscilloscope signal processor* CMOS gate array to decelerate and postprocess the ADC data, the digital channels have a similar *logic analyzer signal processor* CMOS gate array. However, the techniques for storing and plotting timing analyzer traces are radically different from oscilloscope traces. More specifically, on the faster time base settings, oscilloscopes use a technique known as *equivalent time sampling* to build up a display record over multiple triggers, resulting in an effective sample period of 50 ps or less (2.5 ps for the HP 54645A). Timing analyzer traces, however, are always single-shot—an entire waveform is reconstructed from each and every trigger, with an effective sample period exactly equal to the actual sample period. We therefore chose to double the maximum sample rate on the logic channels to 400 MSa/s to provide better single-shot time resolution. Using the same 50-MHz 16-bit VRAM memory as the HP 54645A, the HP 54645D has a total memory capacity of two million samples for each of eight digital channels (sampled at 400 MSa/s) or 1 million samples across all 16 digital channels (sampled at 200 MSa/s).

Use of Deep Memory

It has long been recognized that one of the factors slowing the acceptance of the digital storage oscilloscope over the traditional analog oscilloscope has been the superior update rate of the analog oscilloscope. Update rate describes the number of waveforms that the instrument can display per unit time. Since update rate is representative of the percentage of the input waveform that actually gets displayed on the screen, it translates to the amount of information relayed to the user. In the past, digital storage oscilloscope manufacturers used a single-processor architecture to handle all user interface, acquisition, and display tasks. This has resulted in digital storage oscilloscopes gaining a reputation for having far inferior update rate capability compared to analog oscilloscopes.

One of the advantages that digital oscilloscopes hold over analog oscilloscopes is their ability to store the data for closer examination after the acquisition is stopped. Digital oscilloscopes with deep memory have been gaining popularity because of their ability to store a given amount of time with more timing resolution than oscilloscopes with less memory. The deeper an oscilloscope's acquisition memory, the more samples per unit time it can capture. To make full use of the deep memory, digital oscilloscopes provide the capability of *panning* (moving the display window earlier or later in time) and *zooming* (changing the display window to display more or less time). This allows the user to display the portion of the acquisition that is of interest, with the desired amount of detail.

Unfortunately, the update rate of the oscilloscope normally suffers with deep-memory acquisition architectures. The update rate is inversely proportional to the amount of memory captured. Since every data point processed has to be read out of an ADC converter and placed in capture memory by the acquisition system, more points translates to more time to process those points. The deep-memory digital storage oscilloscopes on the market before the HP 54645A/D always place the update rate/memory depth trade-off in the hands of the user by means of a human interface control for the acquisition memory depth. If the user is interested in panning and zooming the acquired data after stopping the acquisition, the memory depth can be increased to gain more timing resolution for this task. If the user is only interested in viewing repeated acquisitions of the signal, the memory depth can be decreased to improve the update rate. This requires that the user have an understanding of the trade-offs involved and a knowledge of the user interface so that the changes can be made.

The HP 54645A/D oscilloscope architecture is designed to optimize the memory depth for the user's requirements. If the user is viewing repeated acquisitions of the signal, the memory depth is decreased if necessary to maintain the maximum update rate. If the user stops the acquisition, the memory depth of the last acquisition is changed to use all of the available acquisition memory. This feature is made possible by the same architectural feature that was designed to maximize the oscilloscope's update rate. During most continuously acquiring configurations, half of the acquisition memory is used to write newly acquired data while the other half is being read into the display. Since the architecture is fundamentally based on a 1-million-point memory system, 500,000 points are dedicated to the next trigger while 500,000 points are being read to the display.

If memory depth has been traded off for update rate, a subset of 500,000 points is used for each trigger during continuous acquisition mode. For example, at 200 μs/div, the acquisition time required to fill the screen with an acquired waveform is 200 μs/div × 10 divisions = 2 ms. The time required to acquire 500,000 points at 200 MSa/s is the sample period times the number of points, or $5.0 \times 10^{-9} \times 500{,}000 = 2.5$ ms. Since the time required to capture 500,000 points is larger than the time required to fill the screen, we choose to reduce the number of acquired points to maintain maximum update rate. In this case, 400,000 points can be acquired in 2 ms ($400{,}000 \times 5.0 \times 10^{-9} = 2$ ms), so we acquire 400,000 points. This maintains the theoretical update limitation of 500 waveforms per second (1/0.002 = 500). A more extreme trade-off occurs at 5 μs/div. Only 10,000 points can be captured in the 50-μs acquisition time ($10{,}000 \times 5.0 \times 10^{-9} = 50.0 \times 10^{-6}$).

If the user presses the Run/Stop key while an acquisition is in progress, the assumption is made that this may have been done to examine the last acquisition in more detail. The last acquisition that occurred before the user's action is stored in one half of the memory system. Since it cannot be assumed that more triggers will follow the user's action, this half of the memory may not be written to any more after the Run/Stop key is pressed. However, since the other half of the memory system now contains defunct data, the acquisition system is reconfigured to use the entire 500,000 points available in this other memory half, and keeps acquiring data, looking for a trigger. Since the primary task to be executed when the user presses the Run/Stop

key is to stop acquisition in what appears to be an instantaneous fashion, the system cannot wait forever for an additional trigger. Therefore, a timeout is initiated that will cause the system to cease looking for additional triggers after 100 ms. For the oscilloscope's acquisition system, this is a large window of opportunity to capture a deep-memory record. However, if no additional triggers occur, the acquisition system is brought to a stopped state, and the shorter acquisition obtained on the last trigger is used for postacquisition pan and zoom. Since most applications involve relatively frequently occurring triggers, this methodology will result in most users viewing deep-memory acquisitions every time they stop their oscilloscopes. For applications in which the rate at which triggers occur is uncertain, or in which there is only one trigger event, the user can press the Single key and guarantee a 1-million-sample acquisition for every trigger.

The Decimation Problem

Even with a million points of sample memory, at the slower time base settings, there will still be more samples across the screen than can be stored. We need to prune (*decimate*) the available data down to the memory size, then further thin the stored data down into 4000-point records for rapid display. In these situations, when the ADCs are sampling faster than the data can be used, the oscilloscope is said to be *oversampling*.

There are a variety of techniques available to perform this decimation process, each portraying a different representation of the input signal. Some methods will tend to highlight the subtle variations in the signal, others show the signal extremes, and others hide the signal extremes. As an example, suppose that the sample rate of the ADC is a factor of 100 greater than the desired store rate. Fig. 3 shows various decimation techniques for 100:1 oversampling.

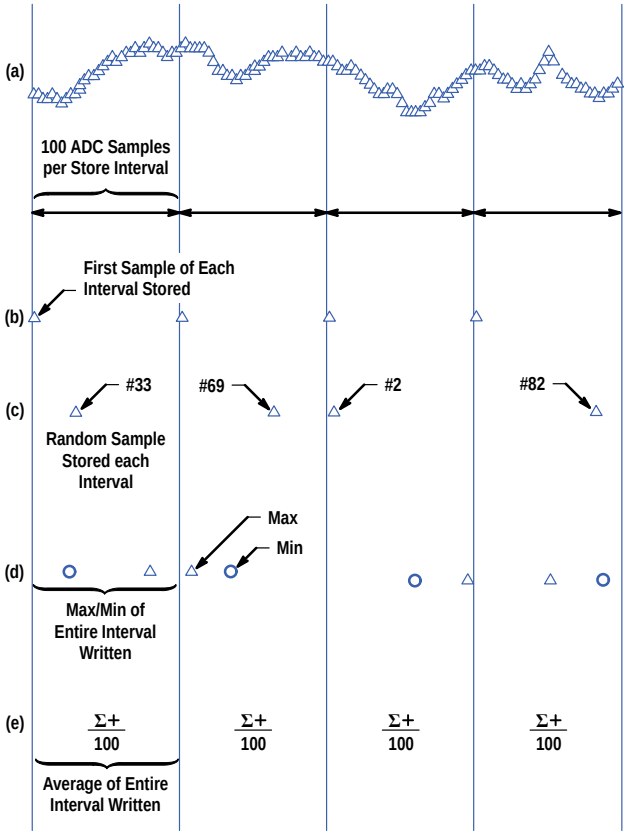


Fig. 3. *Various decimation techniques shown at 100:1 oversampling. (a) ADC samples. (b) Simple decimation. (c) Intra-acquisition dither. (d) Peak detection. (e) Smoothing.*

Simple Decimation. One sample is stored and the next 99 ignored; then the next sample is stored and the next 99 ignored, and so on. This approach is illustrated in Fig. 3b. Since it is the simplest to implement, it is the most common decimation technique in digital oscilloscopes. Unfortunately, because of the very regular spacing of the stored samples, it is exceptionally prone to aliasing. Therefore, the HP 54645A/D rarely use this technique (only when calculating FFTs, when exact sample spacing is called for).

Intra-Acquisition Dithering. Rather than store the first sample during the sample interval, this patented technique stores a randomly selected sample for each interval. In Fig. 3c, sample #33 is stored during the first interval, then #69 during the second, then #2 in the third, and so on. This technique, used in all of the 546xx oscilloscopes, is remarkably effective against aliasing. The stored samples are not evenly spaced, so it becomes much more difficult for the sampled signal to "lock" to the samples. The HP 54645A/D oscilloscopes use this technique with an improved folding pseudorandom number generator.

It is used whenever the instrument is in the normal display mode. For more information on this sampling technique, see *Article 5*.

Peak Detection. Another common data compression technique is to store the minimum and maximum values of all of the samples in the sample interval, as shown in Fig. 3d. Widely used in digital oscilloscopes, this technique is called *peak detection*. When this technique is in use, rare, infrequent excursions are never ignored (lost), as they might be in the preceding two methods. This approach, however, tends to over-emphasize noise on the displayed record. Peaks are exaggerated and baselines become fatter at the expense of signal details. For example, an AM signal, peak detected, would show the modulation envelope quite clearly, but would lose the shape of the carrier wave. Statistical information about where the signal spends most of its time between the minimum and maximum values is lost.

The HP 54645A/D oscilloscopes combine traditional peak detection with intra-acquisition dithering when the instrument is in peak detection mode. They simultaneously acquire and plot both sampled versions of the incoming signal. The peak detected version highlights the signal extremes (one minimum/maximum pair per pixel column), while the denser conventionally sampled record (six points per pixel column) provides statistical information about the signal.

Smoothing. Yet another approach for decimating the incoming samples is shown in Fig. 3e. Here, rather than store the signal extremes for each interval, the average value is stored. This is logically equivalent to a simple low-pass boxcar filter cascaded with a decimating filter. At the slowest sweep speeds, millions of samples are averaged together for every point drawn on the screen, even on single traces. This display mode is useful for pulling the signal out of the noise.

While smoothing reduces the noise in the signal in a manner similar to conventional averaging, there are some differences. Smoothing works on signals acquired with a single trigger, while averaging requires multiple acquisitions to be effective. Smoothing functions as a low-pass filter with the cutoff frequency depending on the time base setting of the oscilloscope. When possible, the HP 54645A/D oscilloscopes use smoothing decimation when conventional averaging is turned on to provide additional noise reduction.

Logic Channel Decimation (Glitch Detection). The acquisition system of the HP 54645D mixed-signal oscilloscope has 16 logic channels. Decimating the series of 1s and 0s for these logic channels provides its own challenges. Simple decimation techniques would lose narrow glitches, so a peak detection technique (known as *glitch detection* in the logic analyzer domain) is always used. Two bits are sufficient to encode any sample interval, one value for each state (high and low). Not suffering from the drawbacks of peak detection on oscilloscope traces, logic waveforms are always acquired and plotted using glitch detection encoding regardless of the display mode selected for the oscilloscope channels.

Trigger System Features

Perhaps the most difficult task of the design was the trigger architecture for the mixed-signal oscilloscope. It needed to be a mixture of both analog and digital trigger systems. The traditional oscilloscope triggering modes (rising or falling edges on any channel, with a variety of coupling and noise-reject selections) needed to be coupled with the logic triggering modes (pattern triggering, sequencing, Boolean AND/OR, and so on). But more significant, all of the cross-domain triggering circuits needed to be defined and developed.

Development of this cross-domain triggering system architecture and choices about which trigger modes to support were guided by two key design objectives. The first of these was to seamlessly integrate the trigger configuration for all channels, whether analog or digital. This allows any channel to be used as a trigger source in any trigger setup. A second objective was to extend this seamless trigger configuration to a full set of trigger features. This feature set extends far beyond traditional single-channel, edge-mode triggering to include functionality essential in a mixed-signal test environment.

In mixed-signal testing, digital signals are often grouped together to form a common data or address word. Consequently, pattern triggering using terms including all channels was an obvious need. Inclusion of analog channels in patterns also provides the unique capability to gate a digital address trigger specification with analog-level conditions. Use of a pattern-qualified edge term (pattern AND edge) is a second key capability, since it allows a typical read or write to be captured by specifying the address in the pattern specification and the edge as the read or write strobe. Specification of trigger sequences in the mixed-signal oscilloscope is a further capability that is especially useful in capturing cause-and-effect relationships between analog and digital signals. Finally, pattern duration and Boolean combinations of trigger terms are included.

Trigger Types

Design of a cross-domain trigger system to meet these objectives was influenced by trigger methods specific to timing analyzers and digital oscilloscopes. Typically, timing (and logic) analyzers triggering on pattern or complex logical terms assume that all trigger events are simultaneous with clock transitions (in our case the ADC sample clock). This is referred to as *synchronous trigger generation*. In the analog world of the oscilloscope, all trigger events are assumed to be asynchronous to clock edges. Since each of these methods has advantages, the final implementation of the HP 54645D trigger hardware uses two independent trigger systems, one for synchronous and one for asynchronous triggering.

Trigger features typical of timing analyzer capability include pattern triggering, sequences of pattern terms, pattern duration, and logical combinations of pattern and edge terms. In a timing analyzer, digital channels are normally grouped together and

trigger conditions are evaluated only at sample clock intervals. There are several advantages to this type of synchronous sample clock triggering. One is that trigger events are always visible on the display because events that occur between sample clocks cannot cause triggers. Also, the signal transition causing the trigger event to occur will not display any jitter at the trigger time because that time occurs at a sample clock edge. Furthermore, complex trigger sequences can be easily implemented in hardware using sequential logic. Finally, it is easy to reject spurious patterns by qualifying pattern detection over multiple samples.

In a mixed-signal oscilloscope, triggering synchronously with the sample clock also has some significant drawbacks. Above all, it degrades equivalent time sampling operation for analog waveforms. This occurs because equivalent time sampling relies on random distribution of trigger events with respect to the sample clock and uses multiple triggered waveforms to enhance the analog display quality. The analog waveform display, when synchronously triggered at fast time base speeds, appears as a broken trace, with bunching of the signal at sample clock intervals (see Fig. 4). Since the HP 54645D samples at 200 MSa/s, degradation of waveforms is only severe at time base speeds of 20 ns/div or less. Furthermore, single-shot trigger captures are unaffected by synchronous triggering of the waveform.

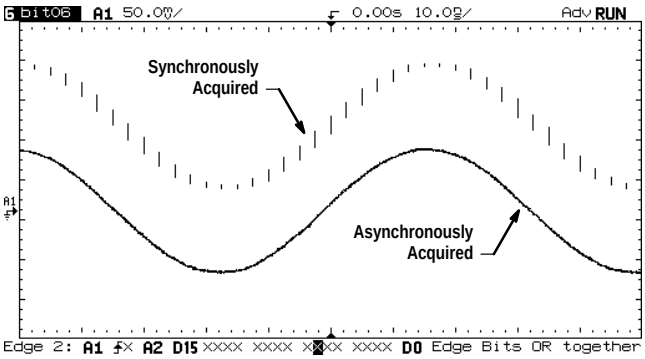


Fig. 4. *Effect of synchronous triggering on the waveform display.*

A second drawback of synchronous triggering is that if the sample rate is too low, trigger events can be totally missed. This is not a problem in the HP 54645D mixed-signal oscilloscope because the sampling interval (5 ns) is smaller than the minimum width of any pulse passed by the 100-MHz analog bandwidth (~7 ns).

Asynchronous trigger generation, traditional in digital oscilloscopes, provides precise trigger timing information to the waveform display. This allows the use of equivalent time waveform reconstruction to combine multiple trigger events and optimize signal display for repetitive signals at fast time base speeds. In addition, trigger time resolution for single-shot captures is limited only by analog channel bandwidth, rather than by sample time granularity. However, asynchronous trigger generation does have one significant drawback. It is possible that the event that generates the trigger may not be visible in the waveform display. This possibility exists because an analog event may cross a trigger threshold and then fall below the threshold again between successive sample clocks. In the HP 54645D, this effect is minimized because the analog bandwidth of the trigger path limits detectable input pulse widths to about 7 ns (depending on threshold level) while the minimum sampling interval is 5 ns.

Mixed-Signal Oscilloscope Trigger System Architecture

To achieve optimal signal display quality and maximize trigger capabilities, the HP 54645D mixed-signal oscilloscope uses both synchronous and asynchronous triggering. The hardware architecture, shown in Fig. 5, includes dual trigger paths that implement this trigger capability. The synchronous trigger path includes a full set of pattern and edge detectors, which allow independent detection of all trigger terms. A pattern duration block is also included to allow triggers based on pattern duration less than target time, greater than target time, and in a target time range. An important feature of this path is the inclusion of high-frequency pattern rejection blocks. These blocks require that patterns be present for a fixed number of sample periods. This enables rejection of patterns that are only present at bus transitions because of channel-to-channel skew. An example of this skew in shown in Fig. 6, where a transition is made between address values of D7 and A4 (hexadecimal). Since the sample clock is not synchronized with the address transition, the sample clock will occasionally occur during transition states (in this case the address B6) and trigger on these patterns unexpectedly. With the high-frequency reject enabled, these triggers are rejected because they are only present for very short times (typically less than 1 ns). The pattern rejection block is automatically programmed by the CPU to require a minimum of two samples for all pattern definitions with more than one defined level. Outputs from the trigger terms detected are then routed through combinatorial logic blocks, sequencing, and holdoff. The design of this path allows for a completely orthogonal set of trigger logic terms.

The asynchronous trigger path provides similar trigger functionality, including edge trigger, pattern detection, pattern-qualified edge detection, pattern duration, and Boolean logic combinations of trigger terms. Since this path lacks

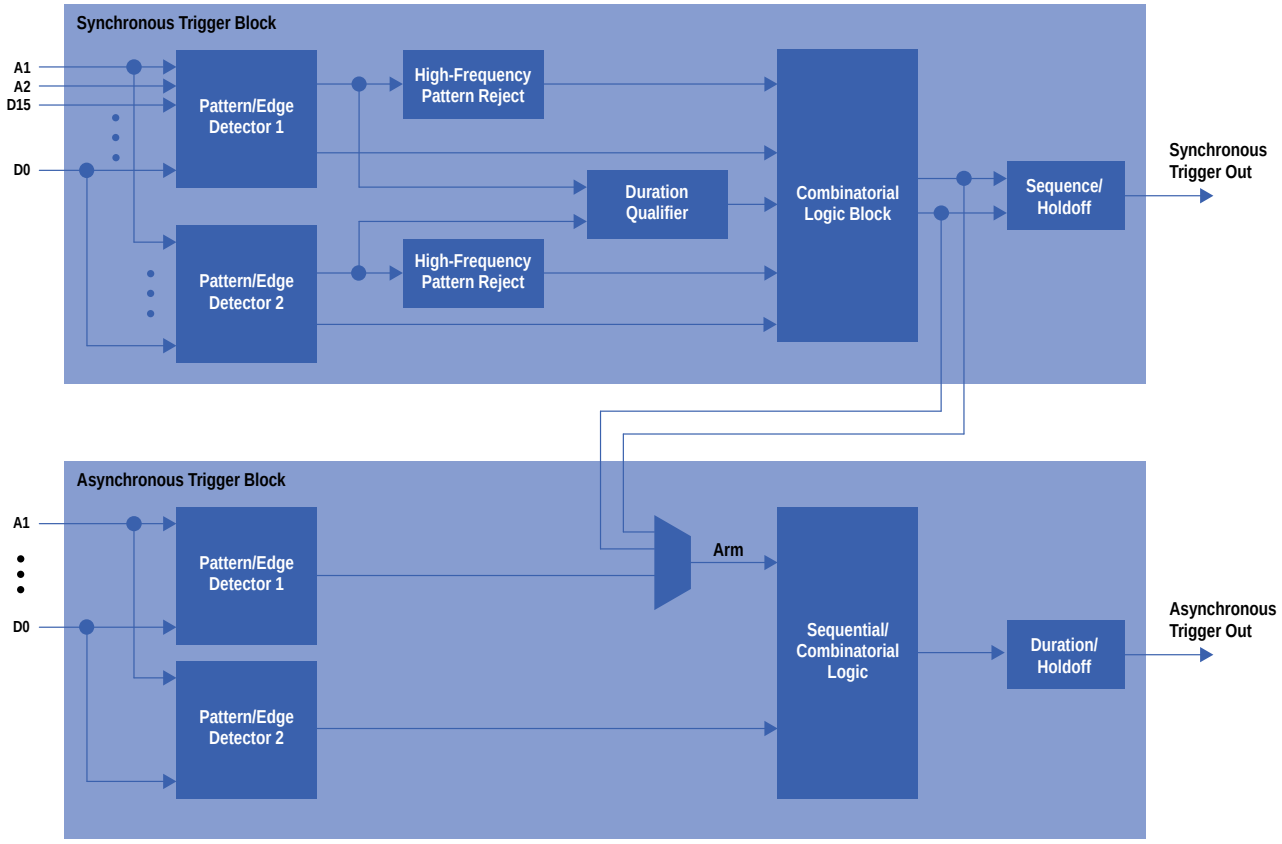


Fig. 5. *HP 54645D trigger system block diagram.*

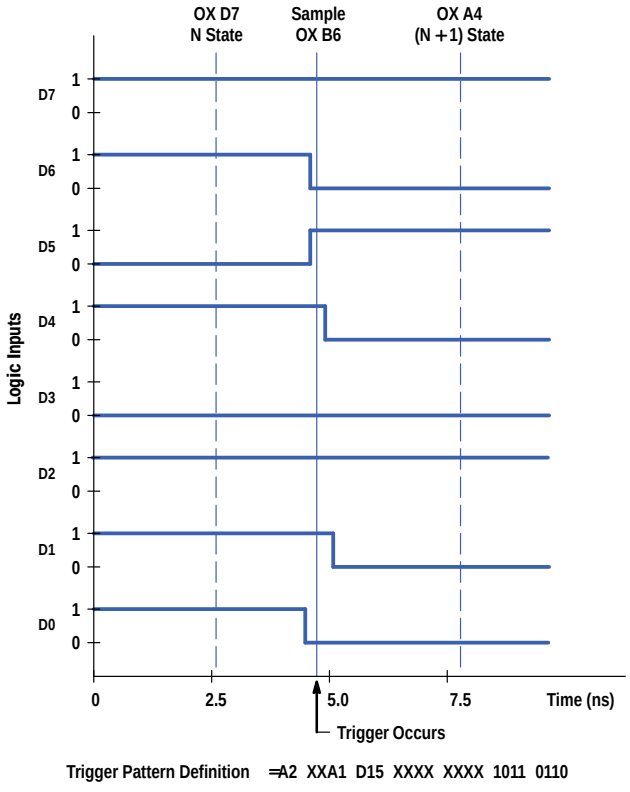


Fig. 6. *Triggering on a channel-to-channel skew state (B6 hexadecimal).*

multiple edge detection blocks, only single-edge trigger terms can be configured. Also, only one pattern-qualified edge detection can be configured. The ability to do cross-domain triggering with synchronously configured pattern terms arming the asynchronous trigger path significantly broadens the configuration choices. For example, an advanced pattern definition of (P1 AND E1) AND (NOT = P2) can be implemented by synchronously detecting P1 AND (NOT = P2) and using this to arm the asynchronous block, which then detects E1 (see Fig. 7).

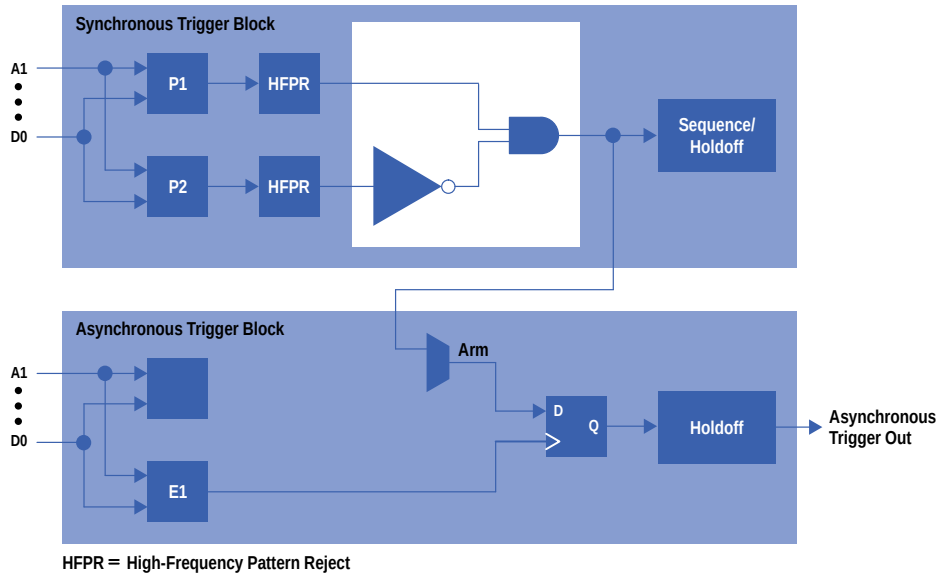


Fig. 7. *Cross-domain trigger configuration.*

Based on the current trigger term definitions and channel display selections, the mixed-signal oscilloscope CPU configures either the synchronous or the asynchronous trigger path as the trigger source for the instrument. To optimize the analog channel display quality, triggers are set up asynchronously, if possible, for all cases in which analog channels are on or analog edges are used in trigger terms. This context-sensitive implementation allows the use of equivalent time sampling to enhance analog waveform display in all possible cases. Consequently, waveform display is optimized without the need for the user to know the details of the trigger configuration.

Mixed-Signal Oscilloscope Trigger Applications

Typical trigger applications for the HP 54645D mixed-signal oscilloscope use the rich set of triggering functions to capture events combining analog and digital criteria. The following paragraphs present specific examples that show the unique capabilities of the mixed-signal oscilloscope trigger system.

ADC Performance Evaluation. Dynamic ADC performance can be evaluated using either pattern triggering to trigger on specific digital outputs or edge triggering with the analog threshold set to the level of interest. By providing an input sawtooth waveform, slew rate effects, missing codes, and nonlinearities can easily be evaluated.

Gated Processor Address Read/Write. An edge-qualified pattern detection can be used to detect a processor read or write to a specific address. By using analog channels in the pattern definition, it is possible to gate the trigger so that only processor reads and writes are displayed that occur when an analog signal is in a specific state. For example, in an alarm system in which a processor normally polls critical sensors to determine if alarm conditions exist, the use of a gated pattern and edge trigger could allow a display of all processor activity starting with (triggering on) the first processor read for which the sensor state indicates an alarm.

Glitch/Pattern Duration. Glitch triggering allows triggering on the duration between the positive and negative edges of a single signal. This allows capture of pulses less than a specific pulse width (glitches), greater than a specific pulse width, and within a specific pulse width range. This is very useful for capturing events that are in or out of specific timing requirements. Pattern duration allows the same duration-triggering choices but combines up to eighteen channels in the pattern definition so that triggering on the timing of combined channels is obtained. A specific example of using pattern duration is in capturing timing violations for a data handshake (see Fig. 8). The time for data acknowledgment after data valid must be less than a maximum time T. Violation of this timing could be found by setting pattern terms DAV low and DTACK high and looking for pattern duration greater than T.

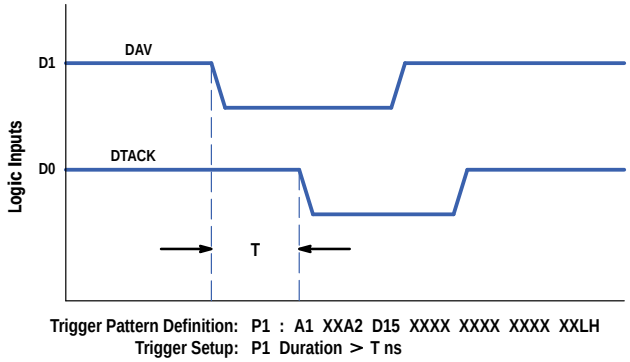


Fig. 8. *Pattern duration triggering to find a timing violation.*

Interrupt Response/Latency. Using a trigger sequence such as E1 THEN P1 AND E2 allows triggering on a processor response to an interrupt by first triggering on a hardware interrupt to a processor and then on a write to a register that is the ultimate response of the processor to the interrupt. The value written out can be read and subsequent events can be traced. Since the mixed-signal oscilloscope captures negative-time (pretrigger) events, this trigger can also be used to measure interrupt processing latency.

Register Write/Analog Response. In control applications, it is useful to be able to trigger on an analog signal that is controlled by the setting of a control register. This can also be achieved by a trigger sequence such as pattern AND edge THEN edge. For example in a motor control application, this trigger sequence can be used to trigger on a write to a control register followed by a motor commutation.

Display Enhancements

In the HP 546xx family of products, waveform update and display have always been high priorities in the design. Traditionally, digital oscilloscopes have had a slow update rate of data to the screen. In the HP 54645A/D the update rate has been increased to approximately three million points per second. This gives a much closer representation of the way an analog oscilloscope displays waveforms. For instance, on an amplitude modulated signal, an analog display would show the upper and lower boundaries with a filled-in area between them. On a traditional digital oscilloscope you would only see a few waveforms inside those boundaries. However, because of the increased update rate on the HP 546xx family, you see the boundaries and the area inside filled with data, a display that matches the traditional analog oscilloscope much more closely. This helps by displaying waveforms more closely to their true nature, but by itself does not address other features of an analog display such as its inherent intensity variation as a function of slew rate.

Analog oscilloscopes generally use a cathode-ray tube (CRT) in which an electron beam is swept across a display at a fixed speed while being deflected in the vertical direction by an analog signal. Because of this, areas of the waveform that have a high slew rate, such as edges, are displayed at a lower intensity than areas such as the top of a waveform, where the slew rate is low. This intensity variation provides information to the user that is not available with a traditional digital oscilloscope. For example, square waves displayed on an analog display are brighter at the top and bottom where the slew rate is low, and quickly changing edges are at a much lower intensity. For another example, consider a waveform that has a lot of baseline noise. On an analog display, the persistence of the phosphor causes the baseline to be very bright and infrequent noise to be dim. On a digital display, the noise is accentuated because all of the data points are displayed with the same relative intensity. This presents the digital oscilloscope designer using a raster display with a real challenge. How to display the waveforms to give a more accurate representation of their time-varying nature?

One thing to note is that the analog oscilloscope doesn't display infrequent signals very well. Because the phosphor has a limited persistence, the waveform will fade after a short time and if not redrawn will not be visible. One attempt to solve this problem is the analog storage oscilloscope, which uses a special plate in the CRT that has a much longer persistence. This is not a perfect solution because the waveform tends to bloom on this type of display and will still fade over time. It is also fairly expensive. A solution in the digital realm has been to use color, displaying data points that are sampled frequently in a bright color that is dimmed at a computed fade rate to simulate a long-persistence phosphor. However, color displays are more expensive, and more memory is needed to keep the counts for each data point. This can also be done with a monochrome display by varying the brightness of a data point in time, but more memory is still required, and in both cases an increased computational burden is placed on the display software. None of these methods addresses the problem of displaying different slew rates at different intensities.

The HP 54645A/D oscilloscopes use a proprietary variable-intensity algorithm to control the intensity of each data point. Adjacent points in the same raster row are used to set the intensity of the current point. Therefore, a point with horizontal neighbors will be brighter than if it has none. In the HP 54645A/D oscilloscopes, two intensity levels are used: full-bright and half-bright. When a point has a neighbor on both sides it is displayed full-bright. When it has zero or one neighbor, it is displayed half-bright. Thus, a high-slew-rate edge is displayed with less intensity because it has no horizontal neighbors.

Looking again at the square wave, the edges will be displayed at a lower intensity than the top and bottom of the signal, which results in a display that looks much more like the analog oscilloscope's representation of the waveform. For the signal with a noisy baseline, the noise away from the baseline is displayed at a lower intensity because the data points do not have horizontal neighbors. The cost to implement this enhancement is greatly reduced because inexpensive printed circuit board hardware is used. The update rate of the waveforms is not affected because this method can be used on the fly as data points are being displayed.

One other enhancement is used when the HP 54645D displays logic waveforms. These traditionally have a high value and a low value with edges drawn between them when the waveform changes. On a raster display, we have a fixed horizontal resolution in which to place the data points. The display is divided into rows and columns of addressable points called pixels. Suppose a point falls at the boundary between two horizontal pixels. On some acquisitions it will be displayed in one pixel, on others in the adjacent pixel. This results in a double-wide full-bright edge. Compared with a point that falls in the middle of a pixel, it will seem brighter. On a display with many logic waveforms this adds a perceptible flicker. The HP 54645D solves this by displaying double-wide pixels half-bright and single-wide pixels full-bright. This has the effect of equalizing the intensity and greatly reducing the perceived flicker.

In all of these methods an attempt was made to give the HP 54645A/D oscilloscopes a display that does its best to represent the waveforms the user might encounter.

Acknowledgments

We would like to acknowledge the technical contributions of Mike Beyers, Stu Hall, Dana Johnson, Pat Burkart, Warren Tustin, Derek Toeppen, Ken Gee, Frank Leri, Marty Grove, and Dan Timm. Jerry Murphy led the marketing effort. Bob Witte and Steve Warntjes managed the whole affair.

Reference

1. R.A. Witte, "Low-Cost, 100-MHz Digitizing Oscilloscopes," *Hewlett-Packard Journal*, Vol. 43, no. 1, February 1992, pp. 6-11.

A Cost-Effective Architecture for a 500-MHz Counter for Glitch Trigger

The HP 54645A and 54645D oscilloscopes can trigger on pulses qualified by their width, a feature known as *glitch trigger.* The time qualification is done with a 500-MHz, 37-bit counter that provides 2-ns timing resolution over an 8-ns-to-100-s range. Operation at 500 MHz (a 2-ns period) is too fast for CMOS logic, but the 37-bit counter is too large for economical integration in bipolar logic. The clock for the counter must be asynchronous with the analog-to-digital sampling clock so that the time at which a trigger is generated is in no way related to the times of the samples. Otherwise, in random repetitive acquisition mode (also known as equivalent time sampling), the samples would not be randomly distributed with respect to the input signal and the oscilloscope waveform would contain missing segments.

The block diagram in Fig. 1 shows how these requirements and constraints were satisfied. The least expensive way to obtain the 500-MHz glitch trigger clock was with a commercial single-chip phase-locked loop locked to the 50-MHz reference clock. The phase-locked loop synthesizes a 250-MHz clock, which is doubled to 500 MHz in the bipolar gate array. Phase locking to the reference clock guarantees that the glitch trigger clock is precisely synchronized with the reference. Since the sample clock is also locked to the reference but phase dithered, the glitch and sample clocks have random phase with respect to each other, preventing missing segments in the waveform.

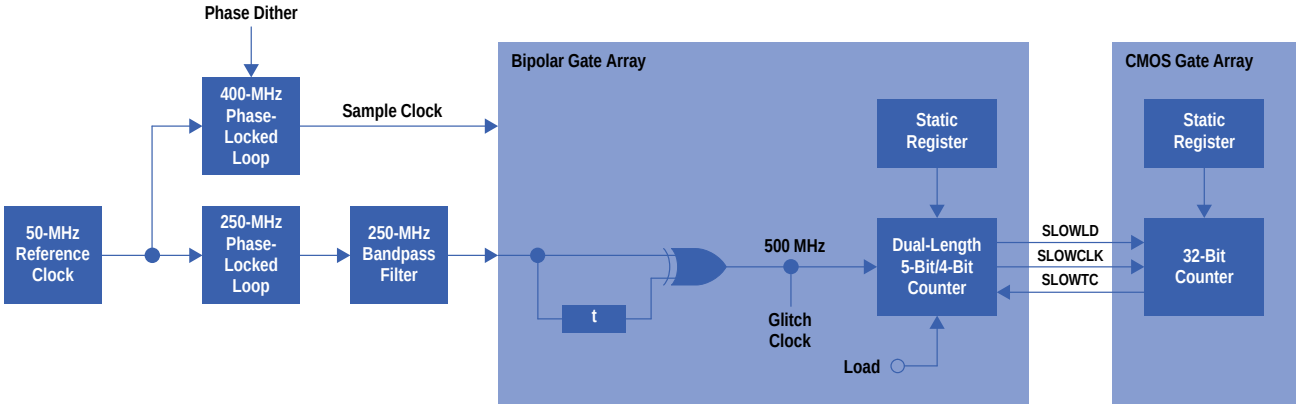


Fig. 1. *Block diagram of the 500-MHz, 37-bit counter.*

The clock doubler generates an active 500-MHz edge on both the rising and falling edges of the 250-MHz clock. The symmetry of the 250-MHz clock affects the uniformity of the periods of the 500-MHz clock. The jitter of the 250-MHz clock causes period jitter in the 500-MHz clock. The delay element τ, formed with gate delays, determines the duty cycle of the 500-MHz clock. The asymmetry and period jitter errors of the 500-MHz clock must both be less than 100 ps to preserve timing margins in the 500-MHz counter. The phase-locked loop IC alone cannot meet these requirements, so a bandpass filter is introduced that removes spurious components in the clock spectrum, improving the symmetry and reducing the jitter from 111 ps to 44 ps.[1]

For reasons of both power and cost, the 37-bit counter is too large to implement entirely in the bipolar gate array. By including only a few of the low-order, high-speed bits in the bipolar IC and the remaining bits in a CMOS gate array we are able to build a very economical, low-power 500-MHz counter. The bipolar portion is called the fast counter and the CMOS portion the slow counter. Together these two operate as a single 37-bit synchronous counter.

The design of the counter is complicated because the CMOS IC must never see pulses narrower than about 10 ns and responds relatively slowly with the terminal count (TC) signal. To overcome these limitations the fast counter operates in either of two modes: as a five-bit counter or as a four-bit counter. For counts from 1 to 31 the fast counter performs all the counting. For counts of 32 and up, the bipolar counter is always loaded with a count between 15 and 31. Since the fast counter is loaded with a minimum count of 15, the slow counter always has at least 15×2 ns or 30 ns to reload itself before it is called upon to begin counting. The fast counter counts down in five-bit mode to zero and then generates a SLOWCLK signal, triggering a count in the slow counter. It then switches to 4-bit mode and counts down, generating a SLOWCLK signal every 15 clocks. Thus, the slow CMOS counter is clocked at only 500 MHz/15 or 33 MHz and has 30 ns in which to generate SLOWTC. These requirements are easily met with modern CMOS gate array technology.

Steven D. Roach
Development Engineer
Electronic Measurements Division

Reference

1. S.D. Roach, "Achieving Ultra-Low Clock Jitter with Commercial High-Speed Clock Generators," *Proceedings of Design Supercon '96*, High-Performance System Design Conference, Santa Clara, California, 1996.

Sustained Sample Rate in Digital Oscilloscopes

At all but a few of the fastest sweep speeds, the acquisition memory depth and not the maximum sample rate determines the oscilloscope's actual sample rate. Peak detection capability, when used correctly, can make up for acquisition memory shortfalls.

by Steven B. Warntjes

One of the most basic specifications in digital oscilloscopes is the maximum sample rate. Oscilloscope users often understand the theory of signal sampling and signal reproduction, but often mistakenly assume that the oscilloscope always samples at its maximum sample rate. In reality, two specifications need to be considered: the maximum sample rate and the acquisition memory behind the signal sampler. At all but a few of the fastest sweep speeds, the acquisition memory depth and not the maximum sample rate determines the oscilloscope's actual sample rate and consequently how accurately the input signal is represented on the oscilloscope display. The deeper the acquisition memory, the longer the oscilloscope can sustain a high sampling frequency on slow time-per-division settings, thus increasing the actual sample rate of the oscilloscope and improving how the input signal looks on the oscilloscope screen.

The digital oscilloscope's peak detection specification is another often overlooked specification. This important feature, when used correctly, can make up for acquisition memory shortfalls. In addition, peak detection can be combined with deep acquisition memory to create unique advantages in digital oscilloscopes.

Digital Oscilloscope Sampling

Basic sampling theory states that for a signal to be faithfully reproduced in sampled form, the sample rate must be greater than twice the highest frequency present in the signal. This sample rate is known as the Nyquist rate.[1] For an oscilloscope, this means that if the user wants to capture a 100-MHz signal, the sample rate should be at least 200 MSa/s. While the theory states that greater than 2× sampling is sufficient, the practicality is that to reproduce the input signal with 2× sampling requires complex mathematical functions performed on many samples. *Reconstruction* is the term commonly given to the complex mathematical functions performed on the sampled data to interpolate points between the samples. In practice, oscilloscopes that do reconstruction typically have less than perfectly reproduced waveforms because of imperfect application of the mathematical theory,* and they may have a slow waveform update rate because of the time it takes to do the necessary computations.

One solution to the reconstruction problem is to sample the waveform at a rate much higher then the Nyquist rate. In the limit as sampling becomes continuous, as it is in an analog oscilloscope, the waveform looks "right" and requires no reconstruction. Consequently, if the digital oscilloscope can sample fast enough, reconstruction is not necessary. In practice, a digital oscilloscope rule of thumb is that 10× oversampling is probably sufficient not to require reconstruction. This rule dictates that a digital oscilloscope with 100 MHz of bandwidth would require an analog-to-digital converter (ADC) to sample the input signal at 1 GSa/s. In addition, the acquisition memory to hold the samples would have to accept a sample every nanosecond from the ADC. While memories and ADCs in this frequency range are available, they are typically expensive. To keep oscilloscope prices reasonable, oscilloscope manufacturers minimize the amount of fast memory in their oscilloscopes. This cost minimization has the side effect of dramatically lowering the real sample rate at all but a few of the fastest time-per-division settings.

The sampling theory presented above assumes that the sampler only gets one chance to "look" at the input signal. Another solution to the reconstruction problem is to use repetitive sampling. In repetitive sampling, the waveform is assumed to be periodic and a few samples are captured each time the oscilloscope sees the waveform. This allows the oscilloscope to use slower, less expensive memories and analog-to-digital converters while still maintaining very high effective sample rates across multiple oscilloscope acquisitions. The obvious disadvantage is that if the waveform is not truly repetitive the oscilloscope waveform may be inaccurately displayed.[2]

Digital Oscilloscope Memory Depth

Now that we understand digital oscilloscope sampling requirements, we can examine how memory depth in oscilloscopes affects the sample rate at a given sweep speed or time-per-division setting. The time window within which a digital oscilloscope can capture a signal is the product of the sample period and the length of the acquisition memory. The

* Errors are attributable to high-frequency components in the captured signal and a finite number of samples.

acquisition memory is measured by the number of analog-to-digital converter samples it can hold. Typically in digital oscilloscopes the length of the acquisition memory is fixed, so to capture a longer time window (adjust the oscilloscope for a slower time-per-division setting) the sample rate must be reduced:

$$t_{CAPTURE} = k_{MEMORY}/f_s,$$

where $t_{CAPTURE}$ is the time window captured by the oscilloscope, k_{MEMORY} is the length of the acquisition memory, and f_s is the sample rate.

In Fig. 1 we have oscilloscope A with a maximum sample rate of 200 MSa/s and an acquisition memory depth of 1K samples. Oscilloscope A samples at its maximum sample rate specification of 200 MSa/s or 5 nanoseconds per sample from 5 ns/div to 500 ns/div. As the time-per-division increases beyond 500 ns/div we see that the sample rate drops. At 1 millisecond per division the oscilloscope's sample rate has dropped to 10 μs per sample or 100 kSa/s. This dramatic drop in sample rate at slower sweep speeds leads to oscilloscope displays of fast signals that don't accurately represent the input signal.

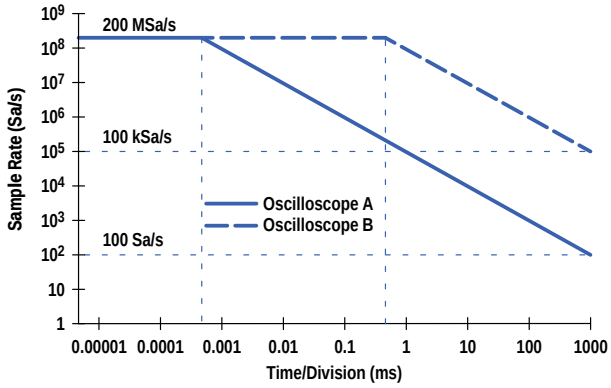


Fig. 1. *Effect of memory depth on the maximum sample rate of a digital oscilloscope. Oscilloscope A has a memory depth of 1000 samples. Oscilloscope B has a 1,000,000-sample memory.*

Fig. 1 also shows the sample rate behavior of oscilloscope B, which differs from oscilloscope A only in that the memory depth has been increased from 1,000 samples to 1,000,000 samples. Oscilloscope B samples at its maximum sample rate of 200 MSa/s from 5 ns/div to 500 μs/div. On fast time base settings, oscilloscopes A and B both sample at 200 MSa/s. An important difference shows up at the slower time base settings. At 1 millisecond per division we see that oscilloscope B samples at 100 MSa/s or 10 nanoseconds per sample while oscilloscope A has dropped to 100 kSa/s. We can clearly see that the deeper memory (×1000) gives us the ability to capture events 1000 times faster at the slower time-per-division settings. The effect of deeper memory is the ability to sustain the maximum sample rate at much slower sweep speeds, in our example from 500 ns/div to 500 μs/div, and sustain faster sample rates at sweep speeds where the sample rate must be reduced because of memory depth limitations. This effect of increased acquisition memory gives the oscilloscope higher sampling performance over a wider range of oscilloscope operation.

Digital Oscilloscope Peak Detection

Peak detection is a powerful feature that compensates for lack of memory depth in digital oscilloscopes. In peak detection mode the oscilloscope looks at many samples but stores only the maximum and minimum voltage levels for a given number of samples. The effective sample rate of the oscilloscope in peak detection is normally the maximum sample rate of the oscilloscope on all sweep speeds.* This enables the oscilloscope to capture fast input signal amplitude variations by sampling at the maximum sample rate, yet save memory space by storing only the significant signal deviations. The value of peak detection is best illustrated with a simple example. In Fig. 2 we have a display of a long time window with short-duration changes in signal amplitude. We are at a time-per-division setting of 20 seconds per division and are showing a 100-nanosecond event. To detect this event without peak detection would require at least a 1/(100 ns) = 10-MSa/s sample rate. Likewise, to capture 200 seconds (20 s/div × 10 div) of information with 100-ns sampling would require a memory depth of 200/(100 ns) or two billion samples. Clearly, peak detection is useful for catching short-duration signal changes over long periods of time without large amounts of oscilloscope acquisition memory. In many instances peak detection can make up for acquisition memory shortfalls.

Traditional digital oscilloscope peak detection has two significant disadvantages. First is the lack of timing resolution of the stored peak detected samples. As described above, the net peak detection sample rate is the maximum oscilloscope sample rate divided by the number of samples searched for minimums and maximums. This leads to a lack of timing resolution because the oscilloscope only stores maximum and minimum values. This 1-out-of-n selection results in a loss of information

* Some oscilloscopes with digital peak detection do not use every possible sample in peak detection mode.

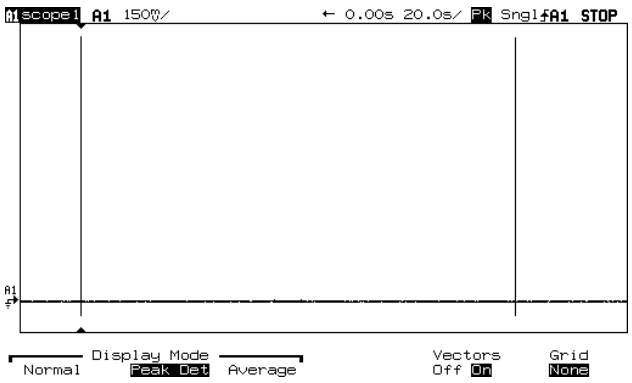


Fig. 2. *A display of a long time window in peak detection mode, showing a 100-nanosecond event at a time-per-division setting of 20 seconds per division. To detect this event without peak detection would require at least a 10-MSa/s sample rate and a memory depth of two billion samples.*

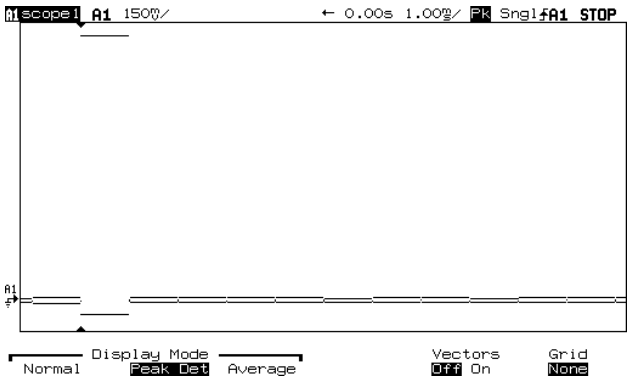


Fig. 3. *The waveform of Fig. 2 with the time scale expanded to display the maximum value. The maximum value is shown as a solid bar, the available timing resolution. The 100-ns pulse captured across 200 seconds shows only approximately 1 ms of timing resolution.*

as to whether the maximum occurred on the first, last, or mth sample of the samples searched for the maximum and minimum voltage values. In Fig. 3 we have taken our Fig. 2 waveform and expanded the time scale to display the maximum value. (The maximum value is shown as a solid bar, the available timing resolution.)

The second peak detection disadvantage is the apparent noise on the oscilloscope display. The storage of only maximum and minimum voltage levels has the effect of making the input waveform appear to contain more noise or signal aberrations then are actually present. This is because the peak detection algorithm stores only these peak values, and not normal signal voltage levels. Traditional digital oscilloscope peak detection gives the user a biased view of the input signal by overemphasizing the signal's infrequent amplitude deviations.

Having considered traditional peak detection advantages and disadvantages, what do deep acquisition memory and sustained sample rate contribute to the peak detection problem? Deep memory has three benefits when applied to peak detection. First is that peak detection is needed less frequently. Deep acquisition memory allows the analog-to-digital converter to sustain a higher sample rate at all sweep speeds. This means that the user is required to switch into peak detection less often to catch short-duration signal variations reliably. The second advantage is increased timing resolution of the peak detected samples, since the benefits of memory depth also apply to the peak detected samples. The deep-memory oscilloscope can store many more minimum and maximum pairs over a given time period, yielding a shorter time interval between the peak detected samples. The third deep memory peak detection advantage consists of the addition of regular (not peak detected) data displayed with the traditional peak detected information. In this case the acquisition memory is segmented and both peak detected samples and regular samples are stored. The HP 54645 deep-memory peak detection display contains regular and peak detected samples. This has the effect of deemphasizing the peak detected samples since they appear as only a subset of the waveform display. While the displayed waveform still differentiates signal amplitude variations by always displaying peaks, it also gives the user a feel for the signal baseline by displaying normal waveform information.

Summary

Conventional digital oscilloscopes achieve their maximum sample rate only on their fastest sweep speeds. We have examined two key features (memory depth and peak detection) that sustain this sample rate over a wider range of oscilloscope operation. Increased amounts of acquisition memory afford the user a much higher sample rate across sweep speeds that are not at the oscilloscope's maximum sample rate and sustain the maximum sample rate of the oscilloscope to a larger number of time-per-division settings. This produces a more accurate waveform representation of the input signal on the digital oscilloscope display across many sweep speed settings. We have also described the advantages and disadvantages of traditional digital oscilloscope peak detection and pointed out some unique sustained sample rate and deep-memory peak detection advantages.

Acknowledgment

I would like to thank Bob Witte for his help in the conception and editing of this article.

References

1. L.C. Ludeman, *Fundamentals of Digital Signal Processing*, Harper & Row, 1986.
2. R.A. Witte, "Sample Rate and Display Rate in Digitizing Oscilloscopes," *Hewlett-Packard Journal*, Vol. 43, no. 1, February 1992, p. 18.

Acquisition Clock Dithering in a Digital Oscilloscope

When a frequency component of the input signal is greater than half the sample rate, aliasing can occur. When the oscilloscope is equivalent time sampling, signals that are subharmonics of the sample clock will be poorly displayed. In the HP 54645A/D oscilloscopes, these effects are greatly reduced by dithering the sample clock during and between acquisitions.

by Derek E. Toeppen

A common concern of digital oscilloscope users is that there are combinations of oscilloscope settings and input signals that will cause the standard digital oscilloscope architecture to display a signal poorly or incorrectly. Since an oscilloscope is a device intended to display a variety of signals, sooner or later one of these combinations of settings and signals will be encountered, leaving the user confused and with diminished confidence in the instrument.

The classic case occurs when the sample rate and input signal violate the Nyquist sampling theorem, or specifically, when a frequency component of the input signal is greater than half the sample rate. When this happens, an aliased waveform will be displayed. A more subtle case occurs when the digital oscilloscope is random repetitive sampling (also known as equivalent time sampling).[1] In this case, signals that are subharmonics of the sample clock will be poorly displayed. This occurs because the repetitive samples are not randomly distributed over the input signal, but rather are bunched together.

There are ways of designing digital oscilloscopes that greatly reduce these effects. These techniques involve dithering the sample clock during and between acquisitions (intra-acquisition and interacquisition dithering). Two such techniques used in the design of the 54645A/D oscilloscopes will be discussed here.

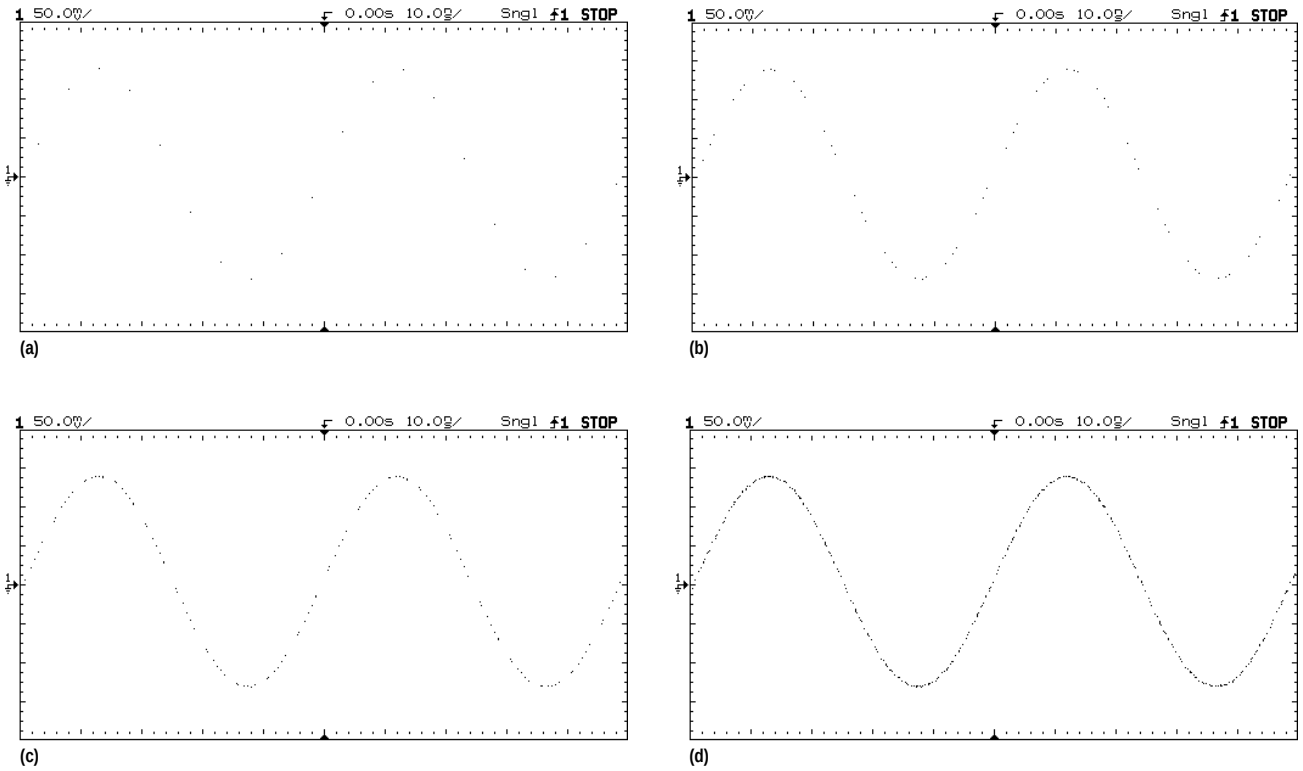


Fig. 1. *The development of a waveform during random repetitive sampling. (a) The result of one acquisition of an input signal. (b) The same signal after three acquisitions. (c) After seven acquisitions. (d) After 20 acquisitions.*

Interacquisition Dithering

The underlying principle of random repetitive sampling is that there is no phase correlation between the sample clock and the signal being sampled. This principle ensures that samples taken of the signal are randomly distributed over the signal, and when accumulated over time, will develop a detailed reconstruction of the input signal. The accumulation of samples is illustrated in Fig. 1. Fig. 1a shows the initial set of samples acquired during the first acquisition of the signal. The sample rate and input frequency meet the Nyquist criterion but that there are, nonetheless, large gaps between samples. Fig. 1b shows the same signal after accumulating three acquisitions. Note the random placement of the second and third sets of samples relative to the first set. The shape of the signal is becoming more defined. Fig. 1c is after seven acquisitions and Fig. 1d after 20. In Fig. 1d, the signal shape is well-defined and accurately represented.

The typical digital oscilloscope relies on the fact that the sample clock (f_s) inside the oscilloscope is derived from a clock separate and independent from the signal being measured (f_i) to satisfy the criterion that there is no phase correlation between the two signals. However, nothing prevents the oscilloscope user from applying a signal that is the same frequency as the sample clock or subharmonically related to it ($f_i = f_s/n$). When this happens, the sample points are no longer randomly distributed across the input waveform as in Fig. 1. This case is illustrated in Fig. 2. In Fig. 2, a sine wave with a frequency equal to precisely 1/10 the sample clock ($f_i = f_s/10$) is applied to a random repetitive sampling digital oscilloscope. The result is that all sets of samples accumulate around the same locations, creating the bunching effect mentioned before. Since the acquisition clock and the input sine wave are not phase-locked to each other, if the waveform is allowed to accumulate long enough, the points will spread out, but the time it takes to do this will depend on the stability of the two sources.

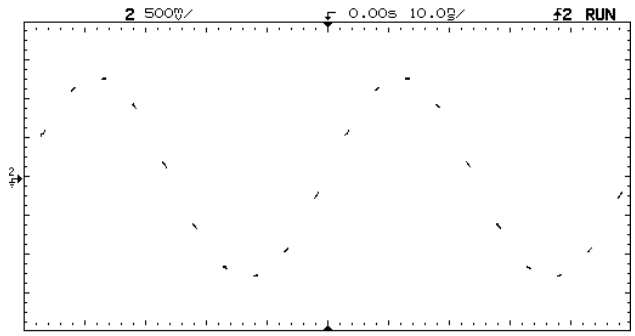


Fig. 2. *The result of applying a sine wave with a frequency equal to precisely 1/10 the sample clock frequency to a random repetitive sampling digital oscilloscope. All sets of samples accumulate around the same locations.*

If the sample set acquired during the first acquisition in Fig. 2 is taken as a reference, then what is needed is a way to shift subsequent sets of samples so that they fall in between the points of the first set (as they did in Fig. 1), or, viewed in terms of the phase of the sample clock, to spread subsequent sets over the 360 degrees of phase between the initial sample points. This can be achieved by shifting the phase of the acquisition clock after each acquisition.

A circuit ideally suited for this task is the phase-locked loop. The basic phase-locked loop is illustrated in Fig. 3a. In this circuit, the phase of the reference clock is compared to the phase of the output of the voltage-controlled oscillator (VCO). The loop filter drives the VCO input to cause the phase difference between the reference clock and the VCO output to be zero. If an error voltage is injected into the filter at the output of the phase comparator, as in Fig. 3b, then the error voltage appears as a phase error to the loop. The loop will adjust the VCO to create a real phase error to cancel the injected error voltage, thereby generating a nonzero phase difference between the reference clock and the VCO output. If the VCO output is used as an acquisition clock, then the error voltage node can be used to create phase shifts in the acquisition clock.

A block diagram of the phase-locked loop circuit used in the HP54645A/D to produce this type of interacquisition clock dither is shown in Fig. 4. In this circuit, the error voltage is generated by a 8-bit digital-to-analog converter (DAC). This provides 2^8 or 256 discrete phase shifts in the acquisition clock. The 8-bit digital word written to the DAC is pseudorandomly generated by one of the oscilloscope's processors to be consistent with the nature of random repetitive sampling. Fig. 5 illustrates the effectiveness of this technique. It is a plot of the sine wave of Fig. 2 using phase-locked-loop-generated interacquisition dither of the acquisition clock. All bunching of the sample points has been eliminated.

Intra-Acquisition Dithering

Real-time sampling is generally considered to be the solution for aliasing in a digital oscilloscope. The reasoning is that if the oscilloscope's sampling rate is always at least twice its bandwidth then aliasing cannot occur. However, what is commonly overlooked is that to overcome finite memory limitations, even real-time sampling oscilloscopes are forced to reduce their sample rates at some point to capture long time periods (see ***Article 4***). When this occurs, the oscilloscope becomes susceptible to aliasing.

A common technique for decreasing the sample rate in a digital oscilloscope is illustrated in the block diagram in Fig. 6. In this circuit, the digitizer always samples at the maximum rate but not all digitized points are stored into memory. The

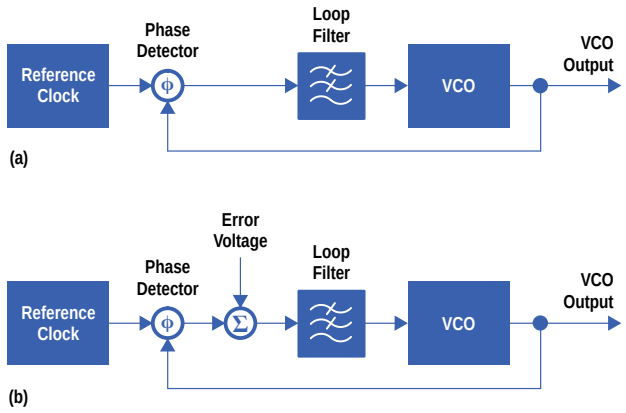


Fig. 3. *(a) Phase-locked loop. (b) Phase error voltage injected into the loop.*

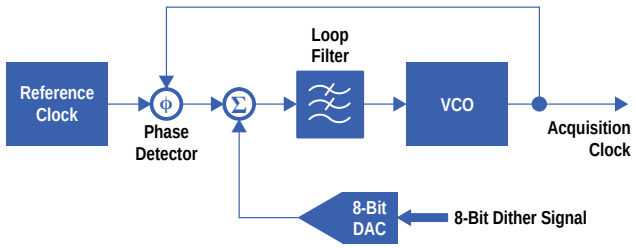


Fig. 4. *Method of introducing acquisition clock dither in the HP 54645A/D oscilloscope phase-locked loop to prevent the bunching effect shown in Fig. 2.*

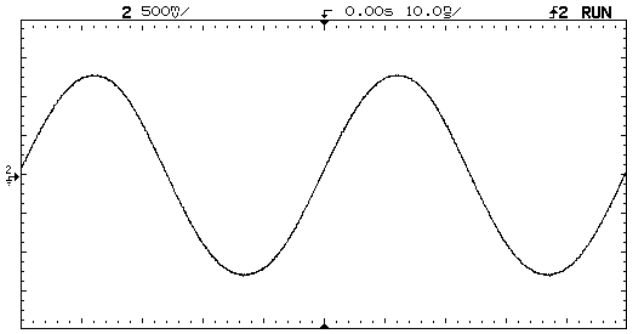


Fig. 5. *Acquiring the signal of Fig. 2 with interacquisition dither eliminates the bunching effect.*

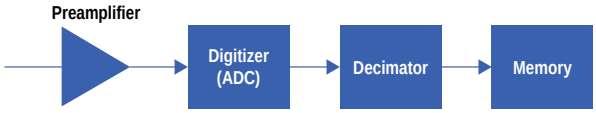


Fig. 6. *Decreasing the sample rate by decimation. The digitizer always samples at the maximum rate but not all samples are stored into memory.*

decimator between the digitizer and memory handles the selection of the desired digitized points. To reduce the sample rate by the factor N, the decimator passes only every Nth point on to the memory. All other digitized points are lost. This technique can be modified slightly in a way that will greatly reduce the likelihood of an aliased waveform developing.

To understand this, first consider how an aliased waveform is generated. If a sine wave with frequency f_i is sampled with frequency f_s, where $f_i > f_s/2$, then an aliased sine wave of frequency equal to $|f_i - f_s|$ will occur. For example, suppose that a 1.01-MHz sine wave is applied to a digital oscilloscope sampling at 1 MSa/s. An aliased frequency (or beat frequency) of 10 kHz will result. What is seen on screen is what appears to be an untriggered 10-kHz sine wave.

This phenomenon repeats itself at each harmonic of the sample clock, so if a frequency of 10.01 MHz, near the tenth harmonic of the sample clock, is input to the same oscilloscope, the same aliased 10-kHz signal is produced. However, between the samples taken at 1-μs intervals (1/1 MSa/s), ten cycles of the input sine wave occur. When the decimator process shown in Fig. 6 is used to reduce the sample rate, samples of those cycles are thrown away. It is these discarded sample points that can be used to prevent the aliased waveform.

Instead of storing every Nth digitized point, the decimator can be designed to randomly select one out of every N points for storage. In the case of the 10.01-MHz input, the points placed in memory are points randomly selected from the ten cycles of the input that occur in every 1-μs interval. This random sample selection technique effectively dithers the acquisition clock during the acquisition and prevents a beat frequency from developing.

This intra-acquisition dithering technique has been used throughout the HP546XX oscilloscope product line and again in the HP54645A/D products. The effect it has on aliasing is dramatic. Fig. 7a shows the aliased 10-kHz sine wave that is produced when a 10.01-MHz sine wave is sampled at 1 MSa/s. Fig. 7b shows the same display using the dithering process just described. The resulting display is a fuzzy band much like what would be seem on an analog oscilloscope, with all signs of an aliased waveform removed.

Acknowledgments

I would like to acknowledge the work of Matt Holcomb, who conceived the intra-acquisition dithering technique, and Allen Montijo and Reggie Kellum who conceived the use of a phase-locked loop for interacquisition dithering.

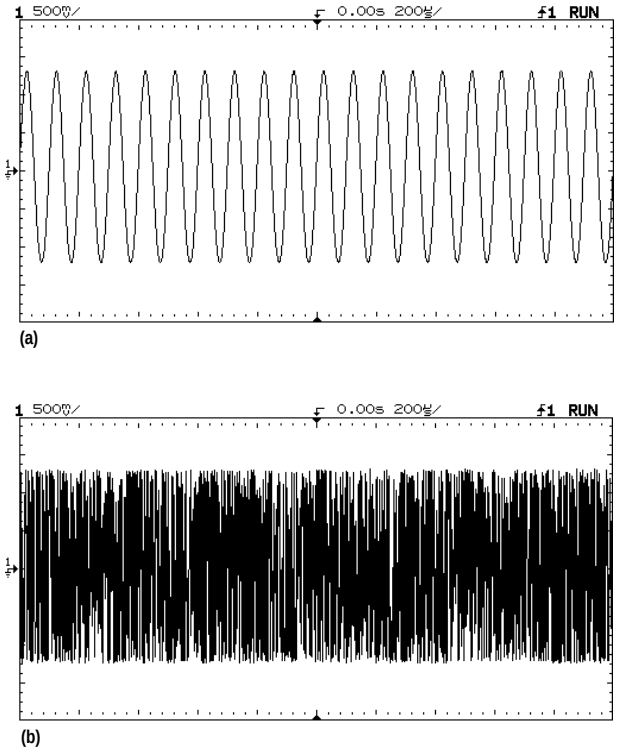


Fig. 7. *(a) An aliased 10-kHz sine wave produced when a 10.01-MHz sine wave is sampled at 1 MHz. (b) The same display using intra-acquisition dithering (random decimation) is a fuzzy band much like what would be seem on an analog oscilloscope, with all signs of an aliased waveform removed.*

Reference

1. K. Rush and D.J. Oldfield, "A Data Acquisition System for a 1-GHz Digitizing Oscilloscope," Hewlett-Packard Journal, Vol. 38, no. 4, April 1986, pp. 4-11.

An Oscilloscope-Like Logic Timing Analyzer

Market research indicated that some customers doing embedded development preferred to work with oscilloscopes instead of standard logic analyzers. The HP 54620 logic timing analyzer offers many oscilloscope features, including direct-access controls, a highly interactive display, computed measurements, delayed sweep, simplified triggering, and a trace labelmaker.

by Steven B. Warntjes

The principal contributions of the HP 54620 logic timing analyzer (Fig. 1) are not in its performance specifications but instead in its user interface, optimized feature set, and price. These are significant contributions, as evidenced by the HP 54620's receiving the *Test & Measurement World* Best in Test Award for 1995. Market research for the HP 54620 indicated that some customers doing embedded development work were frustrated with the cost and complexity of standard logic analyzers and preferred instead to work with oscilloscopes. The HP 54620 bridges the gap between standard logic analyzers and oscilloscopes by providing the functionality of a timing analyzer and the usability advantages of an analog oscilloscope.

Fig. 1. *The HP 54620 16-channel logic timing analyzer bridges the gap between standard logic analyzers and oscilloscopes by providing the functionality of a timing analyzer and the usability advantages of an analog oscilloscope.*

Product Description

The HP 54620 is best described as 16-channel timing analyzer. Each timing analyzer channel has one bit of vertical resolution indicating either a logic high level or a logic low level based on a user-selectable voltage threshold. The HP 54620 leverages HP second-generation logic-analyzer-on-a-chip technology to provide 500-MSa/s or 2-ns timing analysis on all 16 channels simultaneously. This acquisition engine and a custom display processor produce oscilloscope-like, high-throughput displays. Monochrome (HP 54620A) and color (HP 54620C) display versions are available.

The HP 54620 leverages the CPU system and the mechanical design of the HP 54600 Series oscilloscopes for front-panel responsiveness and small form factor. The HP 54620 is positioned as a companion to the user's oscilloscope and therefore has the ability to be triggered from another instrument (trigger in) and to trigger another instrument (trigger out).

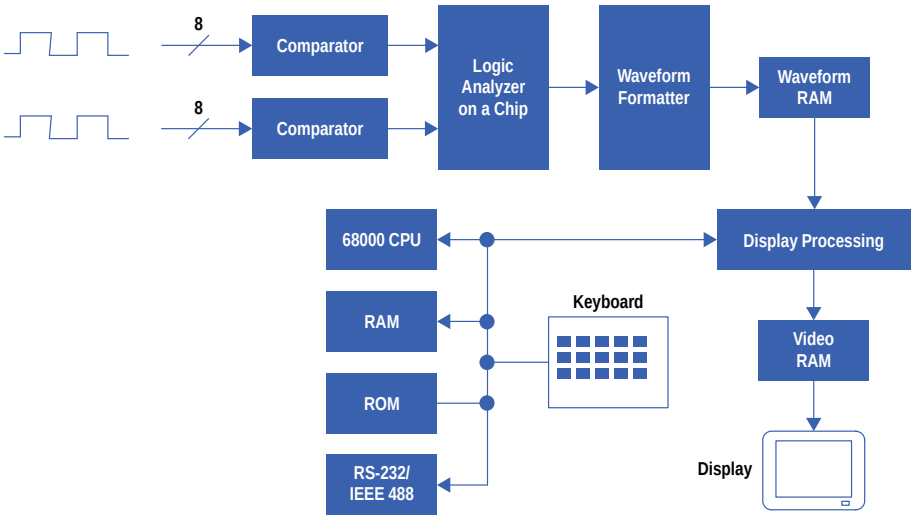


Fig. 2. *The HP 54620's multiprocessor architecture yields a highly interactive display that updates the screen about 16 times per second as a result of parallel processing and acquisition of the waveform data.*

User Interface Oscilloscope Similarities

Most oscilloscope users will tell you that their oscilloscope is relatively easy to use. While some oscilloscopes are more user-friendly than others, the general feeling is that oscilloscopes are easier to use than logic analyzers. The primary reasons are twofold. First, oscilloscopes usually have direct-access controls, something not always true of logic analyzers. Second, oscilloscope displays are highly interactive because of their fast screen update rates. After all, they are primarily waveform viewing tools, and they feel very responsive. The HP 54620 leverages these two oscilloscope ease-of-use attributes and adds some standard oscilloscope features, previously unavailable in logic analyzers, that make the HP 54620 feel and run like an oscilloscope. The project design goal statement was, "Any user who can use an analog oscilloscope should quickly be able to use this new class of logic analyzer."

Direct-access control of instruments means that the most common controls are available on the front panel, through dedicated knobs or buttons. With an oscilloscope, the user often reaches up to expand or contract the time window, often without thinking. This intuitive control is beneficial because it allows the user to concentrate on the problem at hand and not worry about running the test instrument. The front panel of the HP 54620, shown in Fig. 1, has direct controls for time per division, waveform position, and time base delay. Instruments that contain several layers of menus under each front panel key can often confuse the user. The HP 54620 addresses this concern by keeping most menus to a single level under a given front panel key, allowing the user more direct control of the instrument.

The ability to display unexpected or undesired waveform conditions with a highly interactive display often leads the troubleshooter to a quicker solution of the circuit problem. Instrument displays with fast update rates seldom miss infrequent or random events. If the update rate is slow, say less than five updates per second, the instrument can fail to capture important waveform events because it is processing and not acquiring data when they occur. The HP 54620's multiprocessor architecture (Fig. 2) yields a highly interactive display that updates the screen about 16 times per second as a result of parallel processing of the waveform data.

An important element of oscilloscope operation is how responsive it is to control changes. After a control such as the time per division is changed, the instrument should respond quickly to prevent user confusion. The multiprocessor architecture of the HP 54620 helps address instrument responsiveness.

To make the HP 54620 feel and run like a oscilloscope it was necessary to add a number of features that are standard in digital oscilloscopes. One of the advantages that digital oscilloscopes have over analog oscilloscopes and traditional logic

analyzers is the ability to do automatic measurements such as signal frequency, period, duty cycle, and so on. Since the waveforms are stored in memory, the oscilloscope can postprocess the waveform to determine the frequency of the stored signal. Oscilloscope users have come to expect these measurements in digital oscilloscopes. Consequently, automatic measurements for frequency, period, duty cycle, positive width, and negative width are implemented in the HP 54620.

Delayed sweep capability is considered standard in the minds of some digital oscilloscope users and therefore is implemented in the HP 54620. Delayed sweep is the ability to split the instrument screen horizontally to show the waveform at two different time-per-division settings concurrently. Delayed sweep is valuable because of the additional insight that is often gained from the ability to see the waveform at different levels of abstraction. Fig. 3 shows many cycles of a pulse train in the main window with a specific cycle demonstrating an error in the delayed window. In this case the ability to view the big picture along with specific "little picture" detail gives additional information about the waveforms under test.

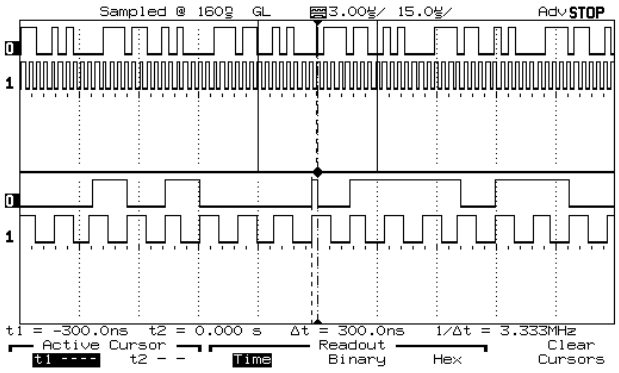


Fig. 3. *The main window shows many cycles of a pulse train while the delayed sweep window shows a specific cycle demonstrating an error.*

Just Enough Features

When deciding on the feature set of an instrument that is supposed to be easy to use, the team faces a dilemma. Too many features make the instrument hard to use because each additional feature adds more controls. On the other hand, too few features dictate that just when progress is being made the instrument's reduced capability becomes an obstacle. The optimum point strikes a balance between features and user interface complexity. The design goal in the HP 54620 was "just enough features—an easy-to-use instrument with the capability I need to do my job the majority of the time."

The best example of just enough features in the HP 54620 is in the area of triggering. The extent of trigger functionality in traditional logic analyzers is broad. Triggering capability ranges from the most basic edge on a single channel to the ability to have up to ten sequence levels in a user-definable state machine using pattern recognizers, range recognizers, timers, and event counters. Conversely, the most basic triggering available and sometimes the only triggering available on oscilloscopes is the ability to trigger on a rising or falling edge at a given voltage on a specific channel. The HP 54620 needed to find a balance between a logic analyzer and an oscilloscope since it is supposed to be an oscilloscope-like logic analyzer. The design team did several rounds of customer visits to understand the target customers, their needs, and their frustrations with traditional logic analyzers before finally arriving at the triggering feature set.

The design team broke the triggering down into three broad categories. First and simplest is edge triggering. This is the logic analyzer version of basic oscilloscope edge triggering. In this category the user selects a channel and an edge polarity. No voltage value is set because it is implied in the logic threshold.

The second category is simple pattern. Simple pattern triggering is defined as highs, lows, and don't cares specified across the input channels. The analyzer triggers when the specified pattern occurs. Optionally, this simple pattern can be combined with a rising or falling edge on a single channel. This optional edge gives the user the ability to qualify the specified input pattern.

The HP 54620 use model called for users to use edge or simple pattern triggering, or both, 80% of the time. Therefore, these trigger categories are accessible through buttons on the front panel. The user switches trigger category by simply pressing the front-panel key.

The third category of triggering is advanced. This category of triggering is in place for the 20% of the time when edge or simple pattern is not powerful enough. In just-enough-features thinking, the added capability of advanced triggering ensures that the instrument is not underpowered. The advanced triggering feature set is best described by the overview screen shown in Fig. 4. Advanced triggering consists of two pattern and two edge resources that can be combined with the logical operators such as AND, OR, and THEN. In addition, pattern duration and edge occurrence triggers are available. While this advanced triggering capability is far below that provided by the most powerful logic analyzers, customer research showed that this feature set would be acceptable in the majority of situations.

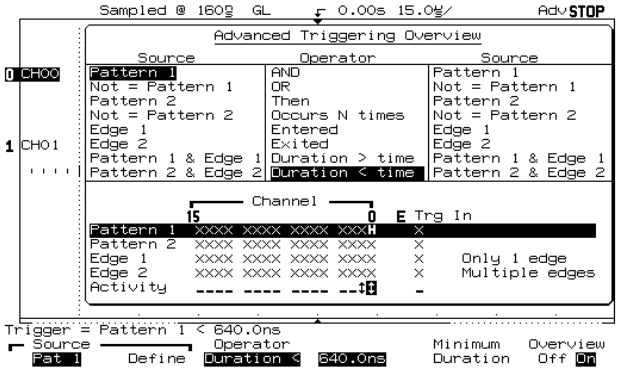


Fig. 4. *The advanced triggering feature set is described by the overview screen.*

Doing the Right Thing

Customer research on waveform viewing tools indicates that these tools are used for bursts of activity separated by weeks or months of inactivity. This leads to customers relearning the tools for each burst of activity. “Doing the right thing” is an attempt at reducing or eliminating the relearn time. Research has shown that when an interface control does what the user expects, it need not be explicitly remembered. Instead, users remember or attempt to remember controls that are exceptions to what they expect. Unfortunately, user interface exceptions are often forgotten and sometimes remembered incorrectly, resulting in frustrated users who can’t get the instrument to do what they want. Doing the right thing means the instrument responds as expected, perhaps even delighting the user by doing exactly what the user had in mind in a given context.

HP 54620 customer research also determined that users need to be able to label the sixteen input channels so as to better understand the onscreen signals. Since the HP 54620 has no computer keyboard this presents a user interface problem. Our solution was to come up with a labelmaker. The labelmaker needed the ability to combine characters to form meaningful signal names. The labelmaker, shown in Fig. 5, provides control of character selection with a knob on the front panel. Softkeys are used to position the cursor and manipulate the label under construction.

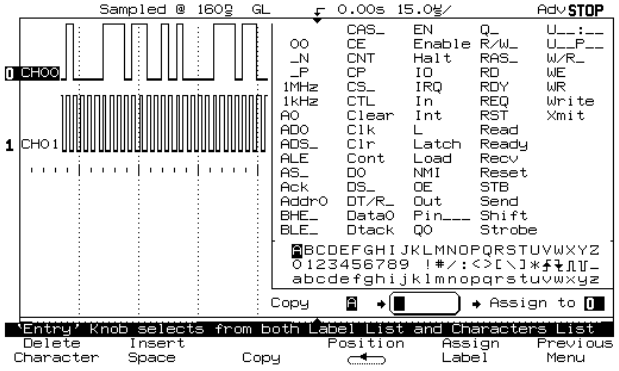


Fig. 5. *The labelmaker provides control of character selection with a knob on the front panel. The label dictionary, a nonvolatile label list of 75 entries, is remembered when the instrument is turned off.*

Even with our best design effort, customers found inputting labels to be tedious. Several changes were then made to the labelmaker to make it do the right thing and delight the customer. The first change was the inclusion of a label dictionary. This nonvolatile label list of 75 entries is remembered when the instrument is turned off and is seeded by the factory with a number of standard signal names. Research indicated that many customers were using the same label names based on which microprocessor or microcontroller they were using. The label dictionary then becomes customized to the user’s labels over time. As the user constructs a new label, it is saved in the label dictionary and the least recently used label is removed. Using the label dictionary dramatically reduces the number of labels that need to be constructed using the labelmaker, making the product much easier to use. This shortens the setup time of the instrument and leads to more understanding of the signals on the screen.

The second change in the labelmaker has been identified as a “delighter” by customers. In defining labels for input signals, the user is often looking at a bus such as an address or data bus. A typical naming convention for buses is ADDR0, ADDR1, ADDR2, and so on. The prefix can change but the number at the end defines the specific bus bit. In the labelmaker the user assigns the constructed label to the selected channel by pressing the Assign Label softkey. In the normal case, Assign Label simply does the assignment to the channel, not changing the edit field or the selected channel. In the case of a bus assignment, that is, when the new label ends in a number, the Assign Label softkey has a different behavior. It not only does

the label assignment, but also increments the selected channel and increments the label in the edit field to the next logical bus bit. For example, if I have just constructed a label named DATA0 the Assign Label softkey will assign that label to channel 0. Since this is the define bus case (the label ends in a number), the new label in the edit label field will be incremented to DATA1 and the selected channel will increment to channel 1. Thus, to define a bus, the user need only press the Assign Label softkey repeatedly to label the entire bus. In addition, the label dictionary is smart: only the first bus label is saved in the dictionary so that the entire dictionary is not filled with bus label entries.

Conclusion

The HP 54620 shows how contributions can be made in user interfaces and optimized feature sets while leveraging existing Hewlett-Packard technology. Customer research and usability testing are key ingredients this process.

Acknowledgments

I would like to acknowledge the work of Scott Sampl, Bob Witte, Jerry Murphy, Dan Timm, Ken Gee, Brian Kerr, Frank Leri, and Marty Grove on the HP 54620 logic analyzer products.

Oscilloscope/Logic Timing Analyzer Synergy

Logic analyzers determine whether an input is high or low based on comparison of the input voltage with a reference or threshold voltage. The logic analyzer looks at the input signals the same way digital hardware in the device under test does. What is displayed on the logic analyzer screen, in theory, is what the digital hardware in the system sees. One technique that can be applied when using a logic analyzer is to take the same measurement several times with slightly different threshold voltages and compare the timing diagrams captured by the logic analyzer. Different integrated circuits see the logic transitions (0 to 1 or 1 to 0) at different voltage levels. For example, a TTL gate may switch anywhere between 0.8 volts and 2.0 volts. By taking the measurement with slightly different logic analyzer thresholds, the user can often identify signals that are not being driven to the correct voltage level. The user can also get a feel for what the system timing would be at different temperatures and across different production runs. This gives some insight into timing variances that may cause problems down the road.

Companion to an Oscilloscope

Users often mistrust the logic analyzer because they feel it doesn't give the "real" picture of their waveforms. In this case, the user is referring to the lack of analog content in the waveform displayed by the logic analyzer. On the other hand, the oscilloscope can show the waveform's analog content but the user can't trigger on the event of interest. Ideally, the solution is logic analyzer triggering with oscilloscope waveform viewing. The HP 54620 logic timing analyzer is optimized to be a companion to the user's oscilloscope, not to replace it. This analyzer has both a trigger input and a trigger output on the front panel. The user can trigger it from an oscilloscope or use the HP 54620 as a trigger source to an analog or digital oscilloscope. Fig. 1 shows the HP 54620 triggering on a complex set of digital signals that combine to form a complex analog signal. This is common in mixed-signal embedded systems that contain either a digital-to-analog converter or an analog-to-digital converter. Often, the analog signal is difficult to trigger on with an oscilloscope. This problem is solved by looking at the digital side of the signal and triggering on a specific pattern. By using the two instruments together, the user can be more productive by getting a view of the situation that neither instrument alone can provide.

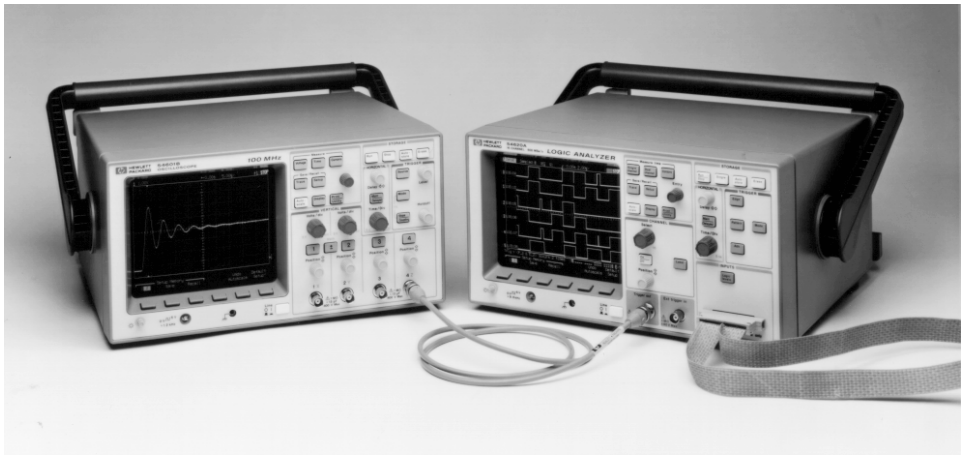


Fig. 1. *The HP 54620 logic timing analyzer can trigger on a complex set of digital signals and provide its trigger signal to an oscilloscope to display a resulting analog signal.*

High-Sample-Rate Multiprocessor-Based Oscilloscopes

The HP 54615B and 54616B oscilloscopes blend proprietary high-speed sampling technology with the power of digital signal processing and a proven user interface to deliver usable advanced characterization capability.

by R. Scott Brunton

The design of the HP 54615B and 54616B oscilloscopes (Fig. 1) focused on adding higher sample rate and extended memory depth to the attributes of the HP 54600 product family.[1] Increasing the sample rate to 1 GSa/s and 2 GSa/s, respectively, broadens the confidence that narrow signal transients will be acquired and, combined with very responsive front-panel controls, presents a visually dense image of the acquired waveform. To provide reliable acquisitions over even the slowest time base settings, special hardware can be engaged to detect and capture peaks as narrow as 1 ns in width.

Successful development depended on retaining the personality of the HP 54600 family by projecting "the feel of analog and the power of digital" along the many dimensions of the customer use model.

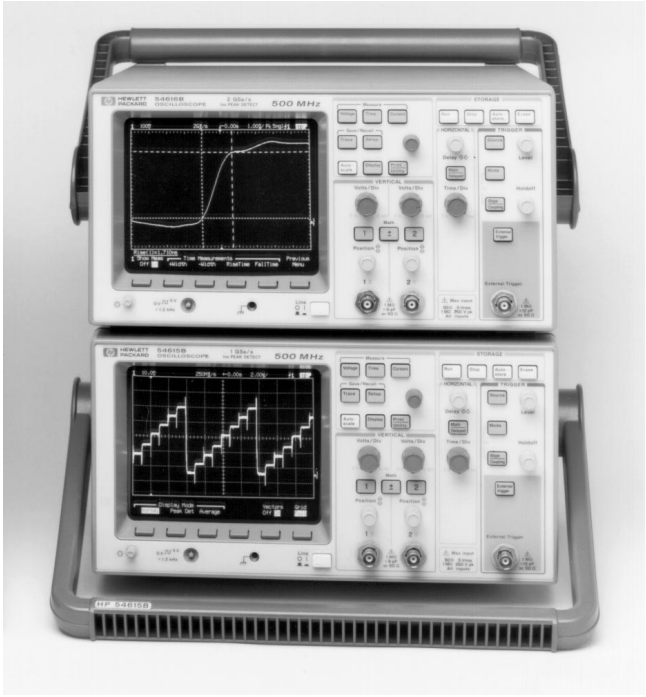


Fig. 1. *The HP 54616B oscilloscope (top) and the HP 54615B oscilloscope (bottom) offer higher sample rates and extended memory depth.*

Product Description

The HP 54515B/16B represent variants on the same software platform. Pursuing this commonality in the underlying software allows efficient delivery of the features and strengths of previous HP 54600 products. To ensure the analog feel of the user interface, fast display update and crisp display quality were cornerstone metrics. Bezel-mounted softkeys facilitate access to advanced control features, and front-panel controls are derived based on the function and operation of classical analog oscilloscopes. Each of the two independent channels delivers a system analog bandwidth of 500 MHz.

Behind the CRT display system and familiar front-panel design is a microprocessor-based acquisition platform that centralizes the overall scheduling of acquisitions and subsequent data filtering and abstraction. Instrument control is through

user front-panel actions or remote control via the HP-IB (IEEE 488, IEC 625) or RS-232. Printers are supported to provide hard-copy imaging of acquisitions and setups.

Optional expansion modules deliver seamless access to additional postprocessing capabilities and full-featured I/O connectivity. Standard 1-ns digital peak detection, advanced measurement functions, and the ability to view events occurring before a trigger permit accurate and detailed characterization of target system designs.

HP 54615B/16B Architecture

Extending the architecture developed for earlier product family members,[1] these new products leverage proprietary technology blocks to deliver added performance. The improved architecture is represented in Fig. 2. Waveform update rate and instrument performance are increased through the delegation of primitive tasks to slave processing elements. Three processors are used. Each has specific tasks and responsibilities that result in a parallel processing boost in responsiveness and a reduction in the "blind" time between acquisitions.

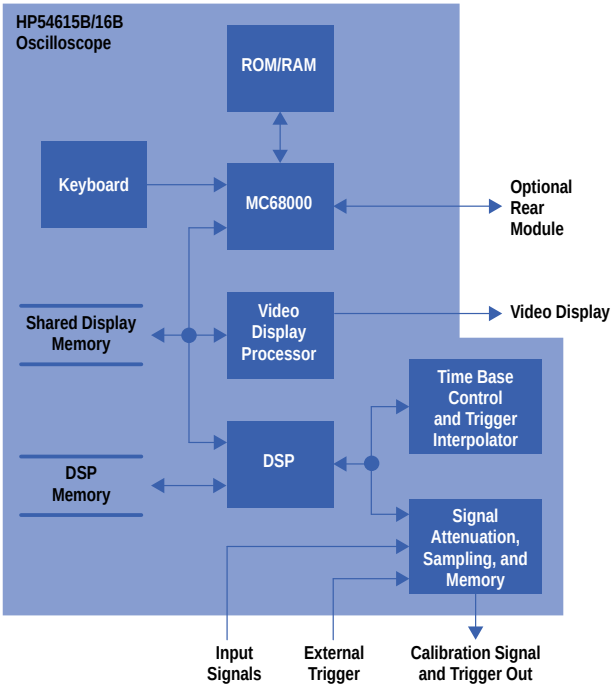


Fig. 2. *Architectural block diagram of the HP 54615B and 54616B oscilloscopes.*

Host Processor

The MC68000 host microprocessor is responsible for the general control and operation of the product. It is the sole provider of service to the front panel, which is of high priority because instrument responsiveness is directly associated with this processor's performance.

In addition to this task, the host maintains communication with and dispatches tasks to the other two processing blocks, where automatic measurements are performed on acquired data and interaction with the optional modules takes place.

During an acquisition, the host processor also tracks the current run state of the instrument and coordinates the subsequent software processing.

Video Display Processor

Once data has been acquired and placed into shared memory for display, the video display processor postprocesses the data record. Data is translated from a time-correlated, sequential organization to one that is appropriate for the display controller. In addition, vector processing on raw data can be performed. This dedicated display processing resource results in a net reduction in the capture-to-display time.

Digital Signal Processor

The TI320C50 digital signal processor (DSP) acts like an acquisition subsystem controller and data processor. It is a slave to the host MC68000 from which it receives instrument state information. The DSP manages the time base and performs sample rate generation. Once a valid trigger is recognized and data is acquired and unloaded from the dedicated front-end acquisition system, the DSP performs time correlation in preparation for postprocessing.

Working at cycle times far shorter than that available on the host MC68000, the DSP, depending on the acquisition mode, performs record averaging and primary vector compression. This processed data record is transferred into a shared memory segment and control is passed back to the host processor. Of significance is the fact that the video display processor accesses this same shared memory segment to transfer the newly acquired record into video RAM without any support from the host. Thus, acquisition-to-display time is markedly superior to methods that require host involvement.

Completion of the acquisition cycle is communicated to the host and, if necessary, the trigger and front-end hardware are reprogrammed and armed for the next acquisition.

Acquisition Front End

The HP 54615B/16B gain many of their capabilities by leveraging hardware technology from other members of the HP 54600 product family. The combined analog attenuation and digital sampling system delivers a signal-input rise time of 700 ps and a sampling rate of 1 or 2 GSa/s. This produces an effective repetitive bandwidth of 500 MHz on both products and a single-shot bandwidth of 250 MHz on the HP 54615B and 500 MHz on the HP 54616B. Through the use of integrated hardware, 1-ns peak detection is achieved over all time base ranges. Following the front-end hardware is a proprietary memory system that is configured to provide 5000-point acquisition records for each channel.

These features permit a high sustained sample rate over a wide number of time base ranges. Engaging the hardware peak detection capability is an effective defense against the possibility of missing fast transient input signals.

Other Capabilities

An integrated internal calibration port enhances the ease of use and reduces the cost of ownership of these products. The CRT, the front-panel keyboard, the internal system memory, and other hardware subsystems are easily tested without the need for additional test equipment. Both vertical and horizontal representations of the acquired data are calibrated for variations in hardware components and the thermal environment.

Additional measurements, enhanced functional capability, mask testing, FFTs, additional nonvolatile memory for setup and trace saving, and expanded I/O connectivity can be provided through the use of attachable option modules.

Software Block Diagram and Theory of Operation

The software environment can be thought of as being assembled from a number of *islands of control*. With the host MC68000 acting as a central dispatcher, tasks are delivered to any of several subcontrollers. Each subcontroller retains independent state information regarding the underlying hardware. In the HP 54615B/16B products, there are three such islands of control.

The first is the MC68000 itself. In addition to running the operating system, the host processor is responsible for dispatching messages related to main and delayed time base sweeps for performing data abstractions on acquired data. In particular, automated measurement calculations on acquired data, mask testing, FFT computation, and function evaluations are performed by the host.

When the state of the host instrument dictates that messages be dispatched to neighboring islands, two methods are employed. The first is a time-sliced access method in which access to a particular island is guaranteed during a given portion of the overall instrument cycle time. During this time, information can be exchanged and island state variables affected. An example of this type of dispatch is the video display processor. When vectors are enabled or disabled, the video display processor is informed of this change of state. In turn, the video display processor effects the necessary changes in the hardware under its control in an autonomous manner.

The second method of message dispatching is based upon a traditional message/interrupt scheme. The host places a message in shared memory, which by protocol is guaranteed not to be corrupted by other islands, and then sends a hardware interrupt to the neighboring island controller. Detection of the message is asynchronous; the host will not initiate subsequent message/interrupt cycles until a handshake from the first is recognized. The DSP is an island that operates on this premise (Fig. 3). For example, when the 1-ns peak detection capability of the instrument is enabled, several message/interrupt packets are built to make the necessary changes to the DSP's internal state variables and tables. Acquisitions are then restarted in the new mode.

Because the host island is running a multiprocess operating system, it can be thought of as several smaller island processes. In addition to the one responsible for communicating with the DSP island, others manage the front-panel keyboard and remote control interfaces. For example, when user interacton is detected by the host, reading of the keyboard state is performed by means of memory mapped hardware located within the address space of the host system RAM.

Each island has specific responsibility for its respective underlying hardware subsystem. Software driver/hardware pairs exist within each island and are unique. For example, the DSP island contains a specific driver to control and operate the integrated time base generator hardware.

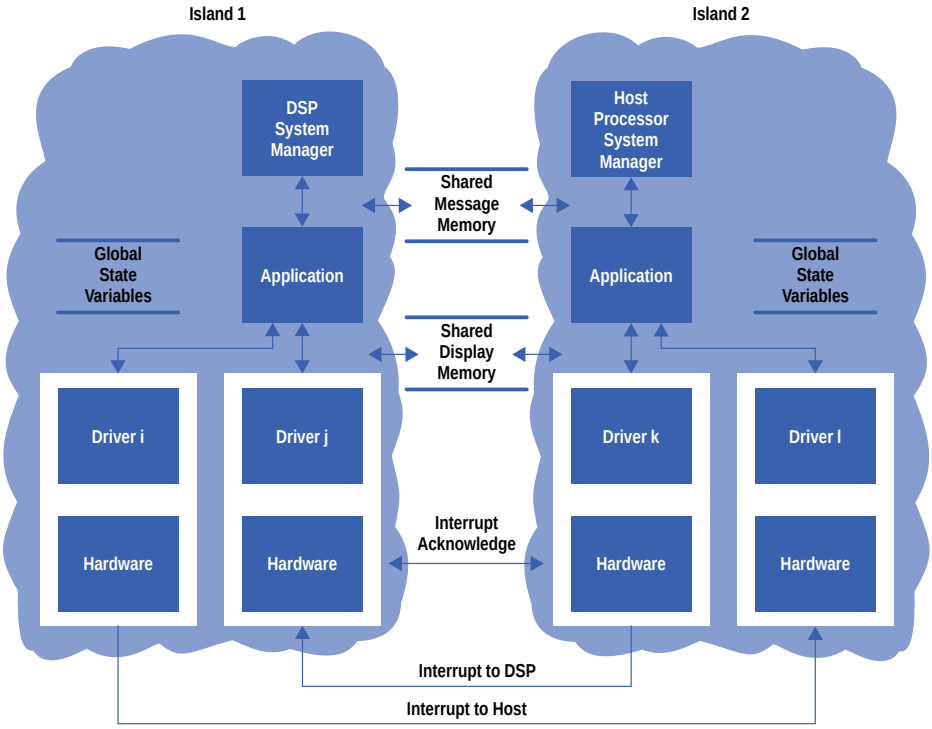


Fig. 3. *Islands of control. Each island consists of a processor with unique applications, hardware, and drivers.*

Code Reuse

An important part of the effectiveness of this development was the the application of software code reuse. An expanded and enhanced printer library was exploited without change to accelerate the development. Taking advantage of software that is ripe for reuse has improved the reliability of this family of instrument products. Consistent with the goal of providing familiar functions throughout the product family, this effort continues.

Summary

Through the application of task-specific hardware and software subsystems, the analog feel of the user interface has been retained while extending the envelope of performance. The notion of islands of control with autonomous operation effectively permits improvements in responsiveness, while an emphasis on software code reuse improves maintainability and the leveraging of existing technology.

Acknowledgments

The author would like to acknowledge the following people for their contributions and support: Dave Allen, Neil Chambers, Kent Hardage, Jerry Kinsley, Mark Lombardi, Lee Mack, Tom Resman, and Tom Schmidt.

Reference

1. *Hewlett-Packard Journal*, Vol. 43, no. 1, February 1992, pp. 6-59.

A Dielectric Spectrometer for Liquid Using the Electromagnetic Induction Method

Key parameters of colloids are often directly related to or can be derived from permittivity or conductivity. Dielectric dispersion analysis (dielectric spectroscopy) yields insights into colloidal properties. A dielectric meter using a new sensing technique has been developed.

by Hideki Wakamatsu

Dielectric spectroscopy is useful for the characterization of colloidal dispersions. In the past, dielectric spectroscopy has been attempted using parallel metal electrodes to measure the permittivity and conductivity of colloids. However, it is difficult in practice to make these measurements in conductive solutions because large measurement errors can be caused by electrode polarization, which is a kind of contact impedance between the electrode and the solution. Electrode polarization behaves like a capacitance and masks the true properties of the solution at low frequencies.

The HP E5050A colloid dielectric probe was developed for colloidal liquid evaluation. Its electromagnetic induction technique eliminates the electrode polarization effect. The probe operates from 75 kHz to 30 MHz with the HP 4285A precision LCR meter and HP VEE software on an HP Vectra or compatible computer (Fig. 1). The HP VEE environment[1] provides easy operation, display, and analysis of the measurement data.

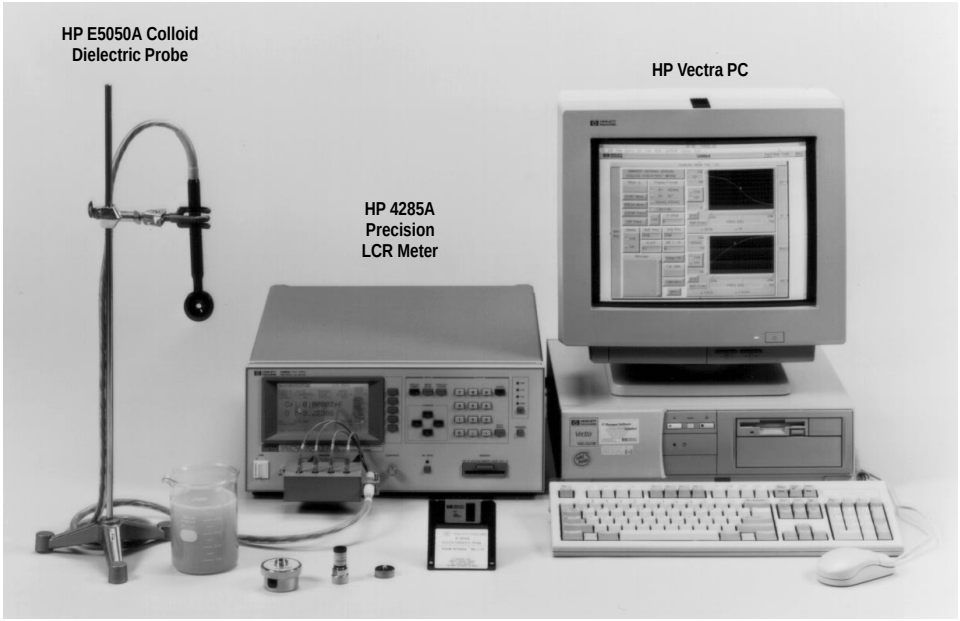


Fig. 1. *HP E5050A colloid dielectric probe and system.*

Background of Dielectric Spectroscopy

Colloids are dispersion systems composed of a dispersed substance in particulate form in a continuous-phase dispersion medium. There are many types of colloid; some familiar examples are listed in Table I.

Table I
Familiar Colloids

Colloid	Dispersion Medium	Dispersed Substance
Shaving Foam	Soapy Water (liquid)	Air, Propane (gas)
Fog, Cloud	Air (gas)	Water (liquid)
Milk, Mayonnaise	Water (liquid)	Fat, Oil (liquid)
Butter, Margarine	Fat, Oil (liquid)	Water (liquid)
Charcoal	Carbon (solid)	Air (gas)
Blood	Serum (liquid)	Erythrocyte (microcapsule)
Black Ink	Water (liquid)	Carbon Black (solid)

Since there are interfaces between the dispersed substance and the surrounding dispersion medium in a colloidal dispersion, there can be appended (nonintrinsic) dielectric relaxations—typically the permittivity decreases and the conductivity increases with increasing frequency—as a result of interfacial polarization caused by charge buildup on the boundaries between the different materials. The analysis of these dielectric relaxations based on an appropriate theory of interfacial polarization provides valuable information on the structural and electrical properties of colloidal particles.[2]

The frequency characteristics of the permittivity and conductivity of colloidal solutions are especially informative. Fig. 2 shows some examples. A practical means of measuring these characteristics—that is, practical dielectric spectroscopy—would be a significant contribution to the study of the stabilization of dispersion systems and product quality control.

Dielectric Measurement of Colloidal Solutions

Traditionally, permittivity is measured with parallel metal electrodes. This technique can be used to measure nonconducting (nonionic) solutions such as oils or alcohols. However, in the case of salty or ionic solutions that have high conductivity, this method suffers from large measurement errors. Fig. 3a shows the mechanism responsible for the errors. Electrode polarization results from the electrical double layer between the metal electrode surface and the ionic solution. This becomes a serious problem at low frequencies because electrode polarization is a capacitance, so the contact impedance is large at low frequencies. If the total impedance consisting of the solution impedance and the electrode polarization impedance is counted as the solution impedance, a large error is incurred at low frequencies (Fig. 3b). In other words, the increase of the contact impedance at low frequencies masks the true properties of the solution.

The second reason why it is difficult to make permittivity measurements is as follows. Often in aqueous colloid spectroscopy, the permittivity is a minor part of the admittance compared to the conductivity. This means that it takes a highly precise measurement to extract the small capacitive permittivity component from the almost entirely conductive admittance. For example, in 0.1% aqueous NaCl ($\varepsilon_r = 80$, $\kappa = 0.2$ S/m), the magnitude of the susceptance (the permittivity part) is only about 1/500 that of the conductance (the conductivity part) at 100 kHz. If 1% accuracy of the permittivity analysis is required, the measurement system must measure the argument of the complex admittance accurately within 20 microradians.

Nonelectrode Electromagnetic Induction Method

If the electrode polarization could be removed, the true impedance of the solution could be measured. To eliminate the electrode polarization, a measurement technique without electrodes has been implemented. In this method, a closed-circuit current is made to flow in the solution by electromagnetic induction. This current is then measured, also by electromagnetic induction.

Fig. 4 illustrates the principle of this measurement technique. When a pair of toroidal coils are immersed in the solution, the solution completes the circuit between the coils. When the primary coil is excited from the ac signal source, a magnetic field in the primary coil induces an electric field around the primary and secondary coils, and the electric field causes a closed-circuit current in the solution. This closed circuit current induces a magnetic flux in the secondary coil and the magnetic flux induces an electromotive force in the secondary coil. If the current flowing in the secondary coil is measured, the current in the solution can be determined. Since the magnitudes of the currents in the solution and the secondary coil are proportional to the closed-circuit solution admittance, the conductivity or permittivity of the solution can be calculated from the measured secondary current and primary voltage.

The induction method avoids the electrode polarization effect and has been used in some conductivity meters suited for high-conductivity solutions. However, while these meters can measure conductivity, they cannot measure permittivity because of the difficulty of measuring the small capacitive part of the admittance, as discussed in the last section. The capacitive part (i.e., permittivity) of the admittance is much smaller than the conductive part for most aqueous solutions.

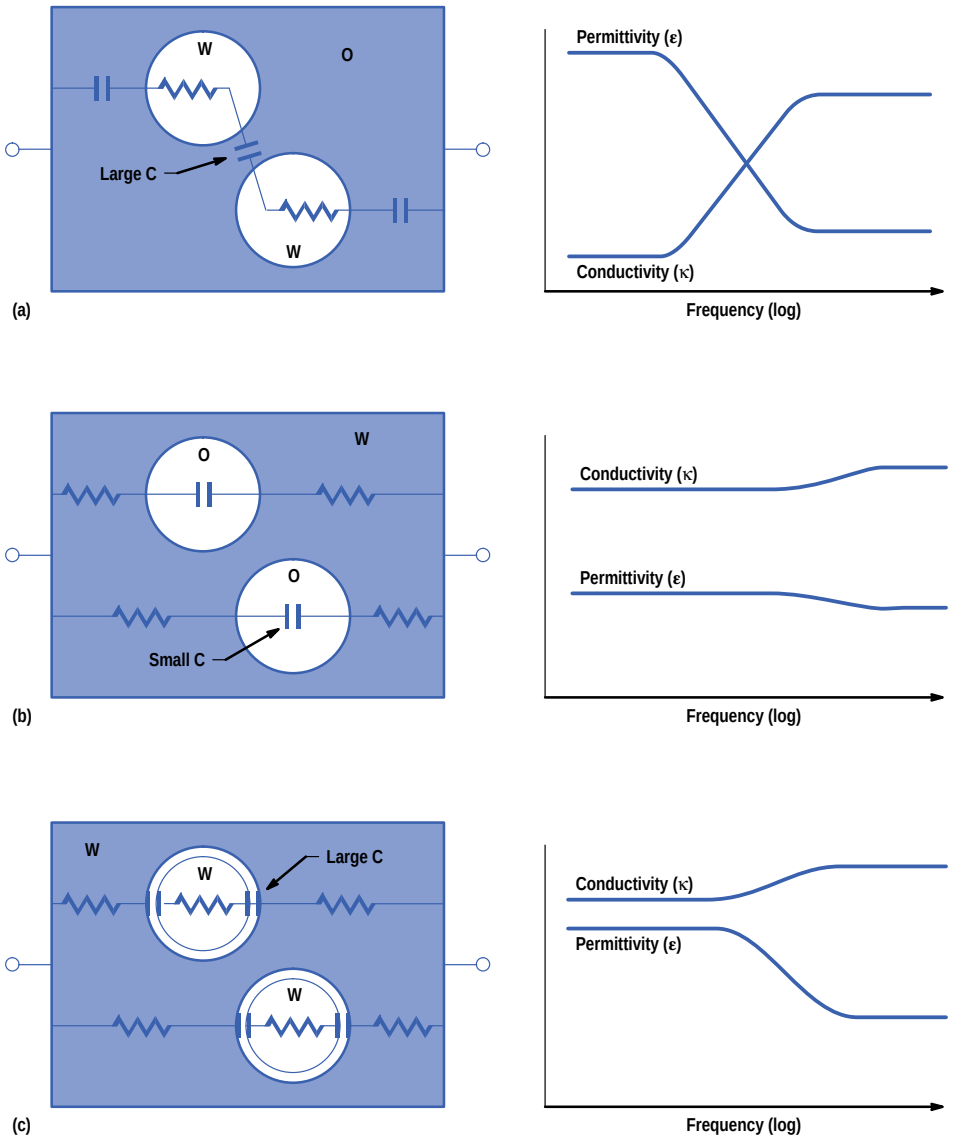


Fig. 2. *Appended dielectric relaxations: mechanisms (left) and frequency characteristics (right). (a) Water-in-oil emulsion. (b) Oil-in-water emulsion. (c) Microcapsule-in-water suspension.*

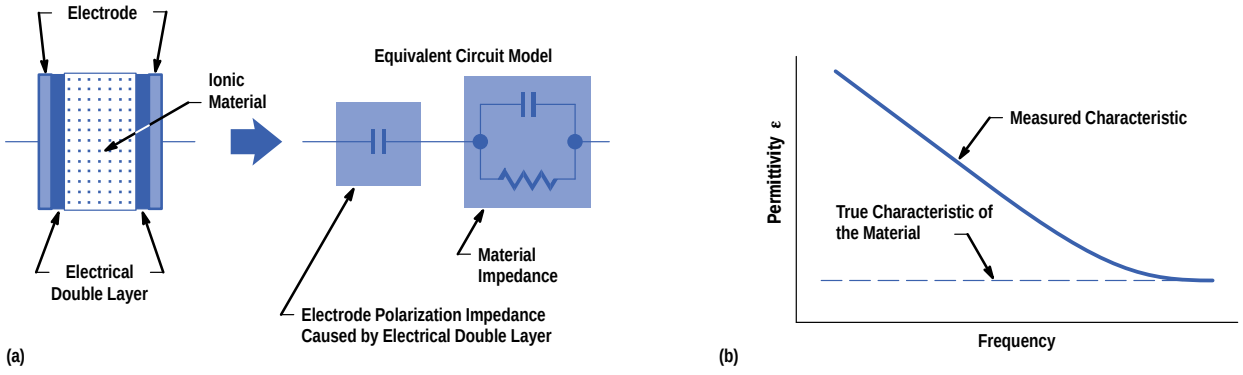


Fig. 3. *Electrode polarization effect. (a) Mechanism. (b) Effect on measured frequency characteristic.*

When a detection device such as shown in Fig. 4 is used to measure a capacitance, measurement errors are produced by undesired coupling between the primary and the secondary. This coupling includes leakage magnetic flux coupling, stray capacitance coupling, and acoustic coupling caused by magnetic distortion of the magnetic cores. All of these are severe problems even if the amount of coupling is small because the secondary current corresponding to the permittivity is

extremely small, even for liquid that has a relatively large permittivity, such as water. This current is often less than one millionth of the primary current at low frequencies (e.g., < 100 kHz).

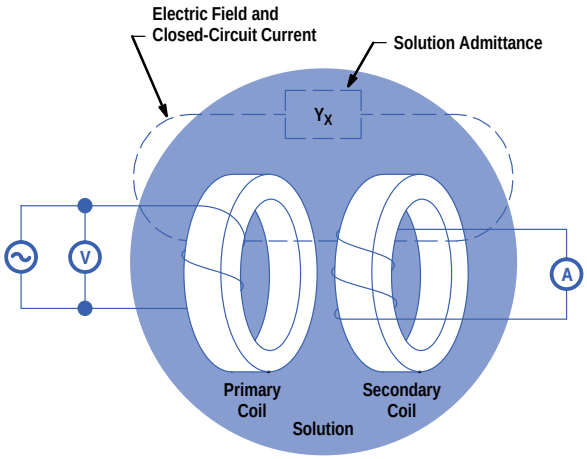


Fig. 4. *Electromagnetic induction method.*

We have developed a new probe and system to solve the problems discussed above. As mentioned earlier, the probe operates from 75 kHz to 30 MHz with the HP 4285A precision LCR meter and HP VEE software on an HP Vectra or compatible computer. The HP 4285A applies 1V to the primary coil, achieves accuracy on the order of 50 microradians or 200 picoamperes in the secondary coil current measurement, and then outputs the vector ratio of the current to the primary voltage, which is measured remotely (Fig. 5). The permittivity and conductivity, which are functions of the vector ratio, are calculated from the measured vector ratio by the software in the computer.

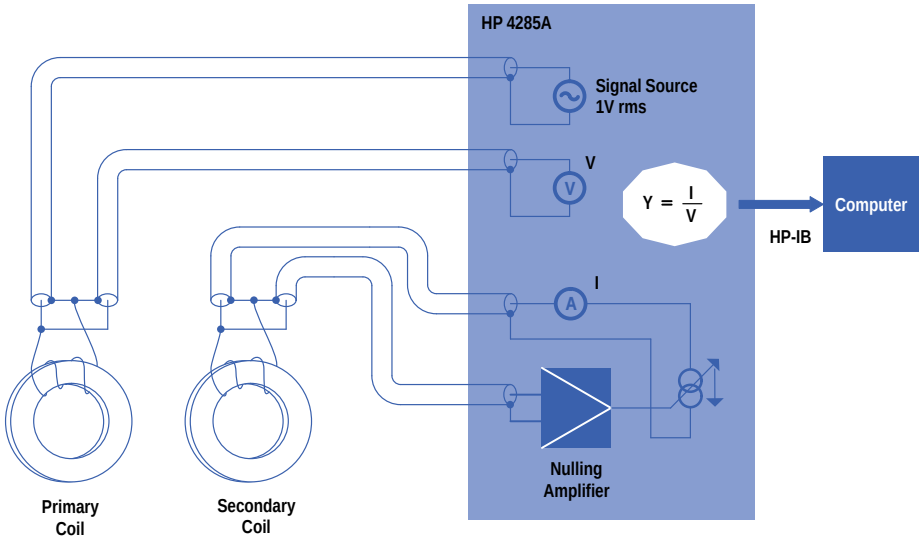


Fig. 5. *Simplified schematic diagram of the probe and LCR meter.*

Probe Architecture

The induction method completely eliminates electrode polarization. However, some improvements to reduce the undesired couplings were necessary for practical use.

The first prototype of the probe had the structure shown in Fig. 6. The interior of this probe consists of two toroidal coils and electrostatic shields, which are covered with epoxy resins. This prototype yielded knowledge in three key areas.

First, although extremely delicate, permittivity measurement is possible. In spite of the tight shield between the primary and secondary coils, the residual admittance from the undesired couplings can be larger than the effective admittance from the permittivity of water (Fig. 7). However, this residual admittance can be canceled out with the assistance of software corrections.

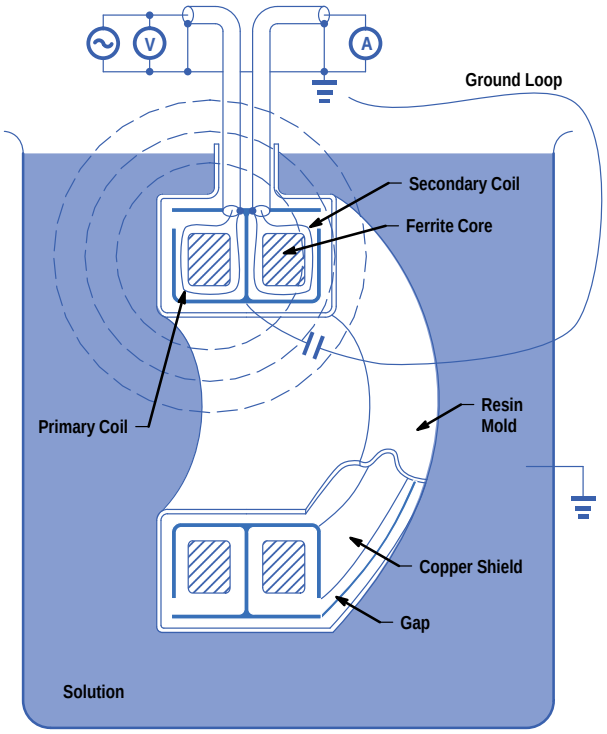


Fig. 6. *Structure of the first prototype probe.*

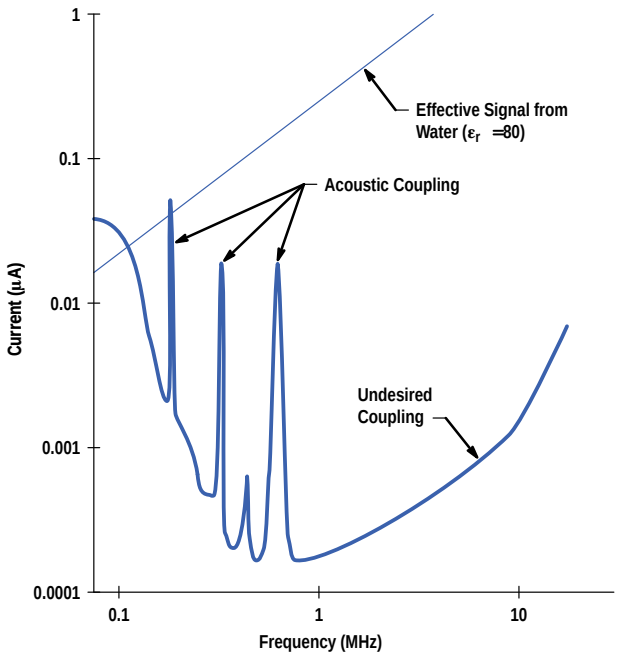


Fig. 7. *Effect of coupling on secondary coil current in water.*

Second, an asymmetrical electric field produces an error if there is a ground loop. When measuring solutions on a beaker scale, where the solution is not grounded, there is no problem using the structure shown in Fig. 6. However, if the solution container is on a tank scale and the solution is nearly grounded, a new problem arises. The asymmetrical field such as the one shown in Fig. 6 produces unbalanced voltages on the symmetrical shield structure. This common-mode voltage produces a ground-loop current, which is measured like an offset capacitance (permittivity). This offset phenomenon is very critical because it is unstable and often larger than the permittivity of water. To eliminate this offset effect, the probe structure must generate a balanced field.

Third, acoustic (microphonic) coupling also produces severe problems. The shape of a ferrite core that has a large permeability is geometrically transformed by a magnetic field. Vibration of the magnetized core produces an electromotive force in the coil wound on the core. When the primary coil is excited from the ac signal source, the primary core vibrates. The secondary core picks up this mechanical vibration and produces an electromotive force in the secondary coil if the secondary core is magnetized. This mechanism spoils the isolation between the primary and the secondary. Since the toroidal cores have many resonant modes of mechanical vibration between 100 kHz and 1 MHz, they produce sharp peaks of residual admittance at specific frequencies (Fig. 7). This residual admittance cannot be canceled out by corrections because it is unstable: its magnitude and frequency depend on the temperature, the magnetization state, and other factors. To prevent this coupling, a suitable buffer material is required between the primary and secondary cores.

On the basis of the above knowledge, a new probe structure was developed. Fig. 8 shows the final structure.

Accuracy Enhancement: Calibration and Compensation

Permittivity measurement is extremely delicate. Corrections are very important to extract the small capacitive component from the almost entirely conductive admittance and to cancel out the residual admittance of the probe.

Open/Short/Load (OSL) Calibration. The equivalent circuit of the probe with the LCR meter is shown in Fig. 9. The solution's loop admittance Y_x and the LCR meter's measurement value Y_m are related by the following equation based on general impedance measurement theory:

$$Y_x = K_1 \frac{K_2 + Y_m}{1 + K_3 Y_m}, \qquad (1)$$

where K_1, K_2, and K_3 are frequency dependent complex constants. The K_1, K_2, and K_3 are parameters related to the circuit of the measurement system and can be derived from the circuit constants. However, it is not necessary to analyze the details of the circuit. The best way to determine K_1, K_2, and K_3 is known as open/short/load (OSL) calibration. The three unknown parameters can be calculated from the relationship between the values (Y_x) of three known admittances and their measured values (Y_m).

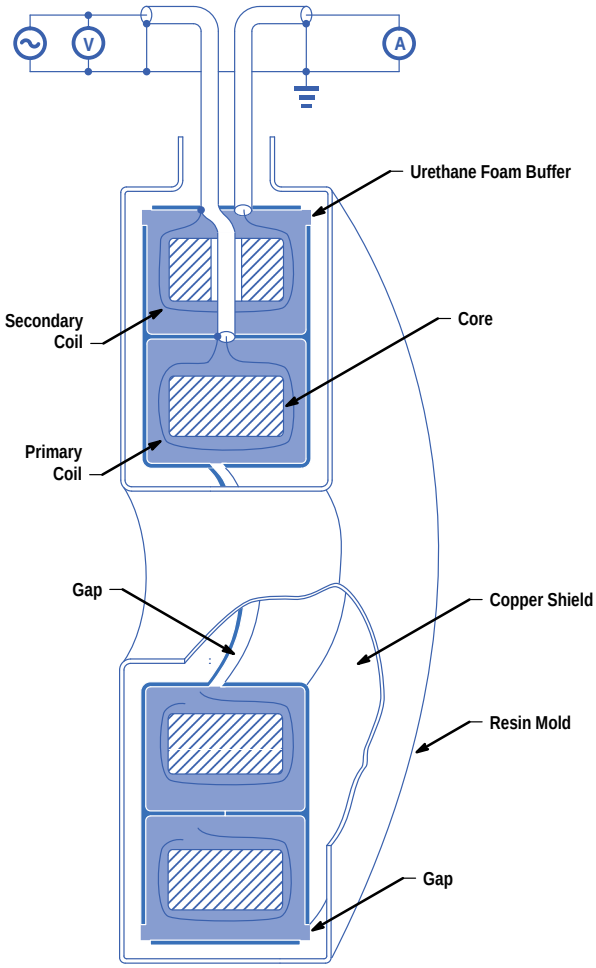


Fig. 8. *Final probe structure.*

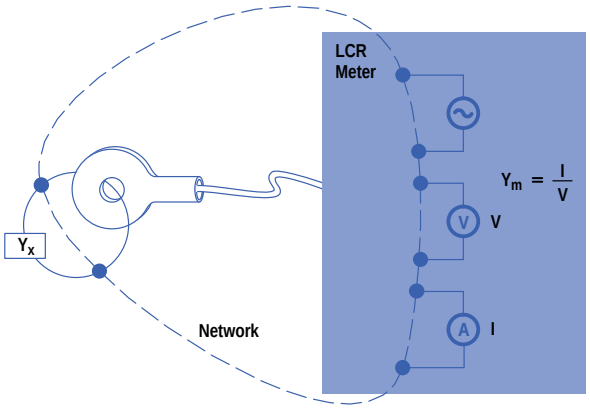
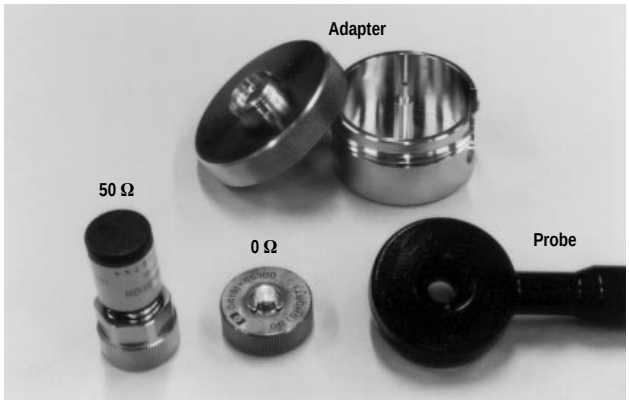
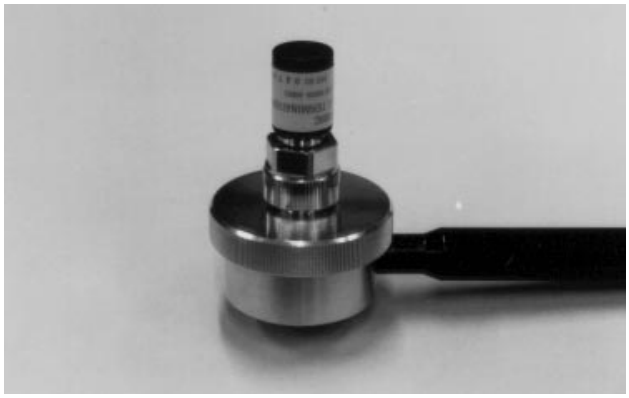


Fig. 9. *Calibration model for probe with LCR meter.*

The three known admittances can be prepared solutions, but a simpler, more accurate, and more stable calibration method is used in our system. The calibration kit shown in Fig. 10 was developed for the OSL calibration of the probe. The kit consists of 50-ohm and 0-ohm one-port standards and an adapter that links the standards with the probe ring. For the load state the probe is placed in the adapter connected to the 50-ohm standard, for the short state the probe is placed in the adapter connected to the 0-ohm standard, and for the open state the probe is placed in air.



(a)



(b)

Fig. 10. *Calibration kit.*

When measuring a solution under test the following relationships exist between the admittance Y_x and the conductivity and permittivity.

$$\text{Conductivity: } \kappa = K_{cell}\text{Re}(Y_x) \quad [\text{S/m}] \tag{2}$$

$$\text{Permittivity: } \varepsilon = K_{cell}\text{Im}(Y_x)/\omega \quad [\text{F/m}] \tag{3}$$

Relative permittivity: $\varepsilon_r = \varepsilon/\varepsilon_o$, $\varepsilon_o = 8.85 \times 10^{-12}$ F/m.

K_{cell} [m^{-1}] is a frequency independent, real-number constant related to the probe dimensions. $\text{Re}(Y_x)$ and $\text{Im}(Y_x)$ are the real and imaginary parts of Y_x, and ω is angular frequency. Measurement of a known solution (e.g., KCl in water) can determine the K_{cell} value, which needs to be obtained only once in the developmental stage of the probe. This is because the probe is molded, and the probe-to-probe variations in its dimensions are insignificant. Differences in the dimensions of the probe do produce errors in the amplitudes of the measured values, but the value of primary importance for measuring permittivity in conductive solutions is the argument of the complex admittance rather than the amplitude.

Compensation. In theory the OSL calibration should provide all the correction that is necessary. However, a problem was discovered when the conductivity and permittivity of a standard KCl solution were measured after calibration. The K_{cell} derived from the real part (conductivity part) of Y_x and another K_{cell} derived from the imaginary part (permittivity part) of Y_x did not coincide. In other words, while the K_{cell} value should be determined uniquely by the probe shape, in practice a K_{cell} value determined this way will give the wrong answer for either conductivity or permittivity. For example, when using the K_{cell} value determined for conductivity, the calculated permittivity of conductive water was usually 30% to 50% smaller than its true value. Furthermore, this error varies depending on the conductivity, permittivity, and measurement frequency. To solve this problem, the error model shown in Fig. 11 was developed.

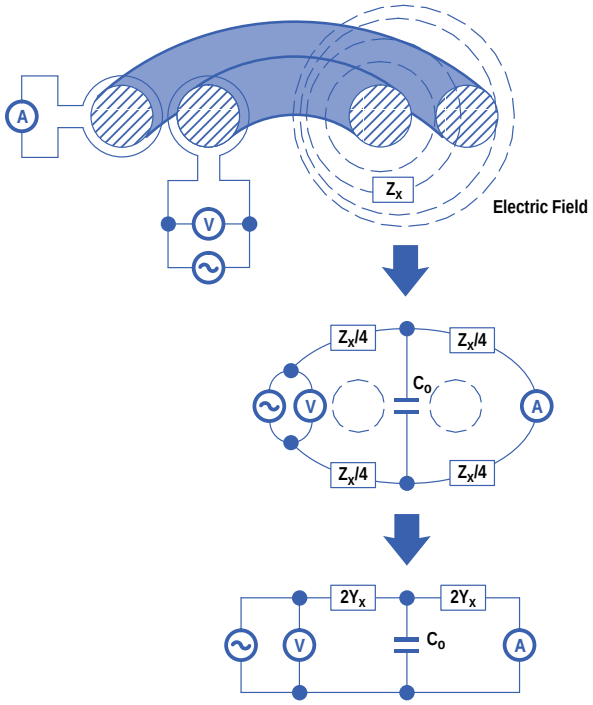


Fig. 11. *Error model involving stray capacitance C_o.*

The compensation equation derived from the model in Fig. 11 is:

$$Y_{x\text{-true}} = \frac{1 + \sqrt{1 + Y_o/Y_x}}{2} Y_x, \tag{4}$$

where $Y_{x\text{-true}}$ is the compensated Y_x, $Y_o = j\omega C_o$, and C_o is a characteristic stray capacitance between the copper shield in the probe mold and the surrounding solution. By using $Y_{x\text{-true}}$ instead of Y_x in equation 2 or 3, the inconsistency between the K_{cell} values for conductivity and permittivity can be improved from 50% to less than 2%. Fig. 12 illustrates the efficacy of the compensation in permittivity.

The value of C_o is determined easily after determining the K_{cell} for conductivity. A solution with a known permittivity (e.g., water: $\varepsilon_r = 80$) is measured, and a C_o value is found that gives the correct permittivity when used in equations 3 and 4.

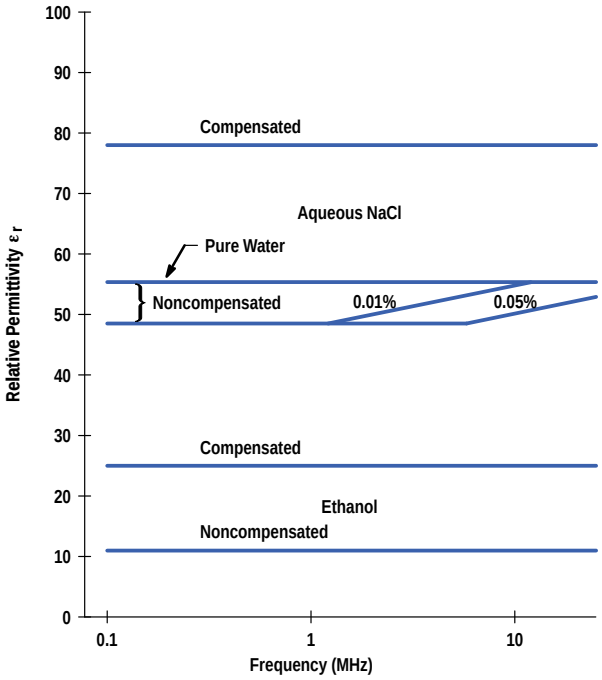


Fig. 12. *Effect of compensation on measured permittivity. (The true values of ε_r at 25°C are 78.3 for water and 24.6 for ethanol.)*

This method would not work if the conductivity and permittivity were to interact, since K_{cell} and C_o would then be difficult to determine independently. However, the effect of C_o on the measured conductivity value is insignificant.

Applications

There are many researchers using dielectric spectroscopy to evaluate colloids. Examples include water-in-oil and oil-in-water emulsions, microcapsule suspensions, biomass monitoring in fermentation, and liposomes. The relations between colloidal structures and dielectric relaxations can be expressed as equations. By measuring the dielectric properties, the researchers hope to measure the structure, geometry, and dimensions of dispersed substances in detail and quantitatively. We expect that the HP E5050 probe will help make such research practical and efficient by eliminating electrode polarization. We will demonstrate two examples of its application.

The first example is liquid car wax. Wax is a water-in-oil colloid that has dielectric relaxation characteristics. Fig. 13 shows the relative permittivity and conductivity for six samples of wax. The samples can be classified into three types of colloidal structures based on their relaxation curves. Noisy data was obtained for low conductivities. It might be more suitable to use

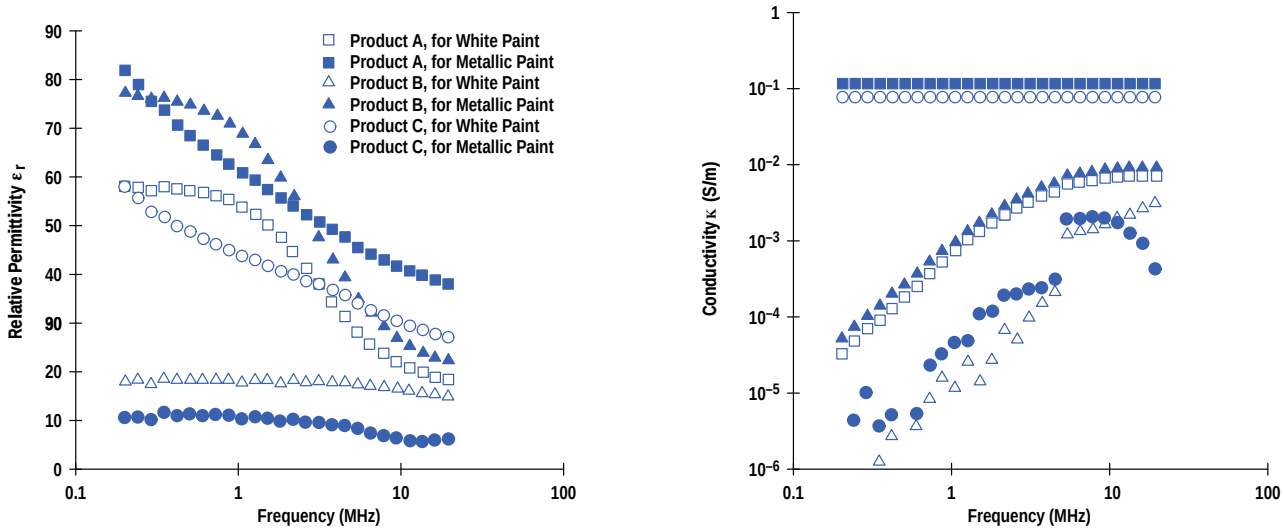


Fig. 13. *Dielectric relaxation of six liquid car waxes.*

the metal electrode method for low conductivities and low permittivities, but the induction method should be used for high conductivities.

The second demonstration is monitoring yeast growth in fermentation. Dielectric spectroscopy is a noninvasive technique suitable for characterizing living biological cells. Biological cells are enveloped in a plasma membrane that is 5 to 10 nm thick. The conductivity of the plasma membrane of a viable cell is negligibly small compared with that of the external medium and the cytoplasm. Since the medium-membrane-cytoplasm structure has the structure of a dielectric material sandwiched between two electrodes, it behaves like a capacitor, and the permittivity of a medium containing cells is larger than that of an empty medium. The increase in the permittivity is proportional to the cell concentration (Fig. 14). Thus, the cell concentration in the cell culture can be monitored by the dielectric method.

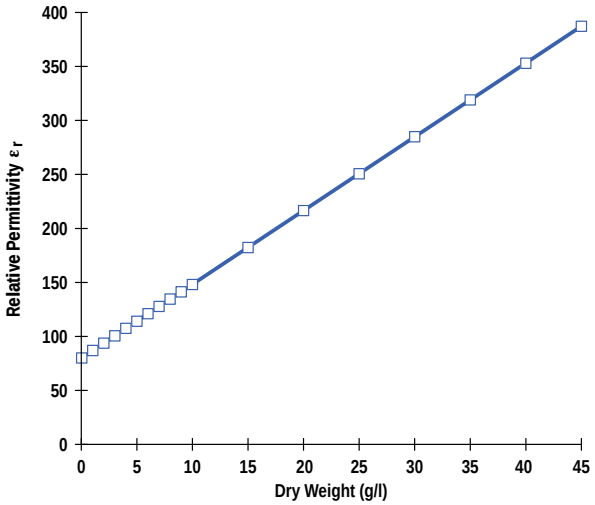


Fig. 14. *Relative permittivity (at 300 kHz) as a function of yeast concentration in fermenting beer.*

Advantages of the HP E5050A system for this application are as follows:

- The technique is nondestructive and requires no sample preparation before analysis. The measurement is quick and simple and online monitoring can be easily realized with computer control.
- Fermentations use ionic culture media, that is, high-conductivity aqueous solutions. The induction probe can eliminate the electrode polarization problem and can measure the permittivity with the required accuracy.
- The dielectric method counts only viable cells. Dead cells with plasma membranes that are leaky to ions are not polarized and are not counted. The method is insensitive to turbid and colored culture media and carbon dioxide bubbles.

Fig. 15 shows the relative permittivity change in beer fermentation at 12°C. The dielectric online monitoring of beer fermentation has succeeded only when the electrodeless method was used. The measurements were made at frequencies between 0.1 MHz and 3.1 MHz. The increase in permittivity up to 80 hours is because of the increase in cell concentration and the increase in cell volume. The undulating curves at low frequencies found at the early stage of the fermentation are caused by highly synchronized yeast growth.[3] The data can reveal not only the cell concentration but also information about cell shape, the plasma, and the membrane that is useful to chemists and biologists.

Summary

Quantitative information about the structure of colloidal dispersions can be obtained from the frequency characteristics of their dielectric properties. The practical realization of dielectric spectroscopy represents a major contribution to the study of the stabilization of dispersion systems and product quality control. Because it is nondestructive, dielectric spectroscopy is also suitable for characterizing living biological cells.

Traditionally, permittivity or conductivity measurements have been made with parallel metal electrodes. However, for conductive colloids, it is difficult to get enough measurement accuracy in practice because this method has large measurement errors caused by electrode polarization.

Electrode polarization can be eliminated by using an induction method, but permittivity measurement is still extremely delicate. In many situations of aqueous colloid spectroscopy, the permittivity is a minor part of the admittance in comparison with the conductivity, so precise measurements are required. We have developed a suitable probe structure and some correction methods for aqueous colloids. The structure realizes the low residual admittance required for accurate permittivity measurements and also generates the balanced field that is necessary to solve ground-loop problems. The

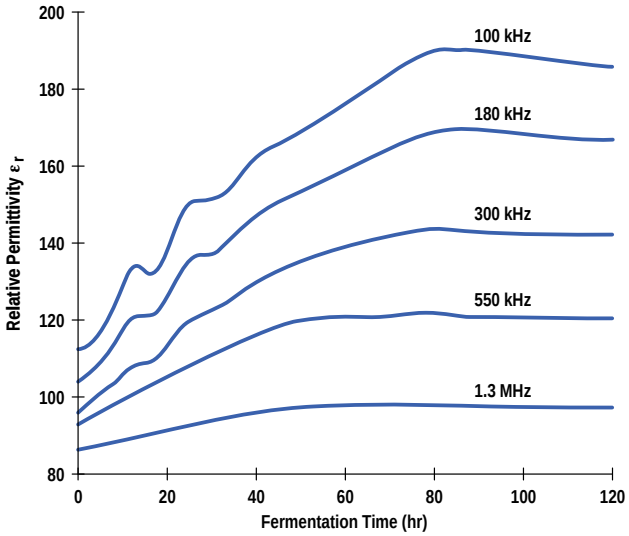


Fig. 15. *Permittivity changes in beer fermentation (courtesy of T. Yonezawa, Suntory Ltd.).*

corrections make it possible to extract the small capacitive component accurately from an almost entirely conductive admittance.

Acknowledgments

The author would like to thank Takeshi Yonezawa, who is the fermentation specialist at Suntory Ltd. His application and advice triggered us to create the probe. He also performed many experiments for us and published papers. The author would also like to thank Dr. Koji Asami, who is an associate professor at Kyoto University and has been studying applications of dielectric spectroscopy for many years. Much information about the applications and some support articles were contributed by him. Thanks to Steve Siano for his evaluation of the probe and application development, Markus Jansons and Naohiko Koyanagi for their efforts on market research and product definition, and the entire management team for their patience and support during development. The author would also like to thank JEOL Creative Co., Ltd. They realized the complex shielding structure of the probe.

References

1. D.C. Beethe and W.L. Hunt, "A Visual Engineering Environment for Test Software Development," *Hewlett-Packard Journal*, Vol. 43, no. 5, October 1992, pp. - .
2. K. Asami, *Evaluation of Colloids by Dielectric Spectroscopy*, HP Application Note 380-3.
3. K. Asami and T. Yonezawa, "Dielectric behavior of nonspherial cells in culture," *Biochimica et Biophysica Acta*, Vol. 1245, 1995, pp. 317-324.

Emulating ATM Network Impairments in the Laboratory

This article discusses a new product for the HP Broadband Series Test System. The HP E4219 ATM network impairment emulator allows telecommunication network and equipment manufacturers to emulate an Asynchronous Transfer Mode network in the laboratory.

by Robert W. Dmitroca, Susan G. Gibson, Trevor R. Hill, Luisa Mola Morales, and Chong Tean Ong

ATM (Asynchronous Transfer Mode) is a new networking technology that is currently being deployed for time-sensitive traffic in high-speed local and wide area networks. Information is encoded into short, fixed-length cells of 53 bytes, consisting of 48 bytes of payload and five bytes of header, as illustrated in Fig. 1.

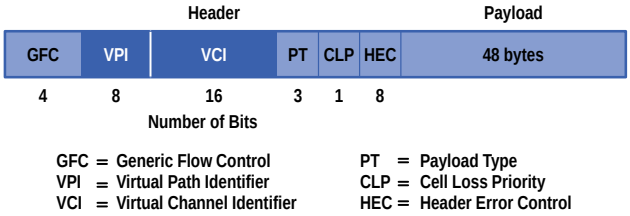


Fig. 1. *ATM cell format (user–network interface).*

An ATM connection can have multiple virtual connections because of a two-part address field in the cell's header—the VPI (virtual path indicator) and the VCI (virtual channel indicator). These fields identify the channel to which the cell belongs and consequently the quality of service it will receive.

ATM has many advantages over other existing networking technologies, such as Frame Relay and SMDS. Its short, fixed-length cell allows switches and multiplexers to route data directly in hardware, as opposed to the slower, software-driven approach required with variable-length packets. ATM's bandwidth-on-demand feature means that a single network can carry all types of traffic—voice, video, image, and data. In summary, ATM is an efficient, flexible technology with the ability to support multimedia traffic at extremely high rates.

Since ATM network implementation is still in its preliminary stages, equipment manufacturers and service providers have a tremendous need for test solutions—in particular, a way to emulate ATM networks in the laboratory so that services and equipment can be tested before deployment. The HP Communications Measurements Division (CMD) has identified three major benefits that ATM network emulation can provide: risk reduction, design verification, and robustness.

Risk Reduction. Full deployment of any network is a very expensive proposition, especially since some planned networks are national in size. Even if a network is available to test, creating maximum delays or errors is difficult since these impairments occur only under congested or abnormal conditions. Network providers can use emulation to test their networks in a well-controlled environment.

Design Verification. Running time-sensitive voice and video data over a packet-switching network such as ATM is an active area of research. Because ATM networks have inherent delays, encoders must be able to handle both variable and fixed delays gracefully. Network emulation allows equipment vendors such as encoder manufacturers to verify their designs in the laboratory in a relatively inexpensive way.

Robustness. Network equipment manufacturers also need to confirm that their algorithms and designs work under stress. For example, some early ATM switches had buffers smaller than a single TCP/IP datagram. When they used AAL-5 (the ATM adaptation layer protocol selected to carry TCP/IP datagrams), the buffers overflowed and dropped cells. Because AAL-5 employs a high-layer, end-to-end error correction scheme, the AAL-5 packets could not be reconstructed and consequently the TCP/IP datagrams were also lost. Almost no data throughput occurred even when error rates were low. Network emulation detects these kinds of unexpected behaviors before they become costly mistakes.

Development Partner

Because CMD develops state-of-the-art test equipment for emerging telecommunication technologies, short development times are crucial. When CMD management made the decision to develop a network emulation module, it became clear that we would need a development partner with more experience in this area if we were going to ship on time.

We were very fortunate to find that Telefónica Investigación y Desarrollo (TI+D) in Madrid, Spain had previously conducted research in network emulation and was willing to cooperate with us to turn their research into a manufacturable product. TI+D is the research and development arm of Spain's national telephone system.

TI+D had developed a network emulator that showed some promise as a commercial product. However, the module's impairment specifications were not quite right for two reasons: the module was designed to interface to a proprietary TI+D line interface, and it was not designed to be manufactured in volume. However, we were confident that we could use the basic technology of this board and redesign it to work in the HP Broadband Series Test System. Development of the HP E4219A network impairment emulator began in the fall of 1994.

Network Impairment Emulator

The HP Broadband Series Test System (BSTS) is a comprehensive platform for testing high-speed local and wide area networks. Designed primarily for laboratory research and early network deployment, the BSTS consists of a VXIbus mainframe, an HP-UX* controller, between one and 12 specialized VXIbus modules for different physical interfaces and convergence layers, and a variety of software applications for testing ATM, SMDS, Frame Relay, and other protocols.

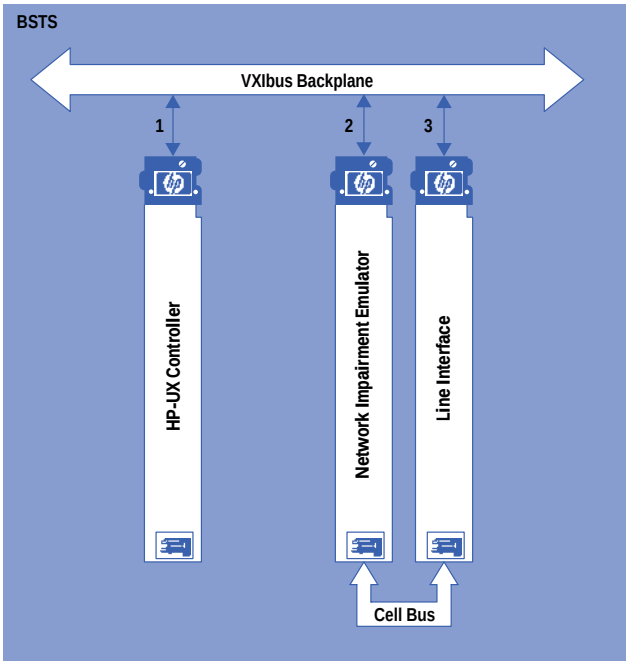


Fig. 2. *Diagram of the HP 75000 Broadband Series Test System with network impairment emulator.*

Fig. 2 is a block diagram of the BSTS with three VXIbus modules—the minimum required when testing with the network impairment emulator. The modules in Fig. 2 are as follows:

- The HP-UX controller, a UNIX®-operating-system-based X Windows controller, provides the instrument interface (GUI and keyboard) and controls all other modules via the VXIbus.
- The network impairment emulator inserts user-specified impairments on selected ATM traffic. The network impairment emulator is a unidirectional product. Two network impairment emulators and two line interfaces are required to insert impairments in both directions.
- The line interface provides the means by which the BSTS connects to a line. In normal operation, the line interface is placed in the middle of a link. ATM traffic is received by the line interface, passed through the network impairment emulator, then placed back on the line by the line interface. A variety of line interfaces work with the network impairment emulator, ranging from T1 to SONET OC-3.

The network impairment emulator connects to a line interface through an HP proprietary cell bus implemented with VXIbus LBUS connections. Using the HP-UX controller, a user can insert into an ATM line a series of impairments that fall into two basic categories: delays and errors.

Delays. Inherent delays are a normal part of the operation of any network, ATM networks being no exception. Although delay is not especially critical for data communications applications, video and voice are highly sensitive to jitter, or variable delay. Since one of the most important commercial uses of ATM technology will be the transport of digital MPEG-2 encoded video, the network impairment emulator enables equipment manufacturers such as encoder and set-top box designers to verify their designs before they are deployed.

Errors. All network transmissions are subject to errors caused by thermal noise, switching equipment, lightning strikes, magnetic fields, interference by other channels, and many other phenomena. The network impairment emulator can inject errors into a line to simulate these kinds of network malfunctions. This enables service providers to ensure that higher-layer transport protocols can detect errors and restart transmissions, to verify that acceptable throughput can be achieved under worst-case error levels, and to determine the worst-case error rate before video and voice degradation become an issue.

Emulator Design

The network impairment emulator offers five impairments: cell delay variation (CDV), constant cell delay, cell error, cell loss, and cell misinsertion.

As illustrated in the emulator block diagram, Fig. 3, cells flow from the cell bus, through the impairments, and back to the cell bus. Impairment modules can impair all or a portion of the ATM cell stream.

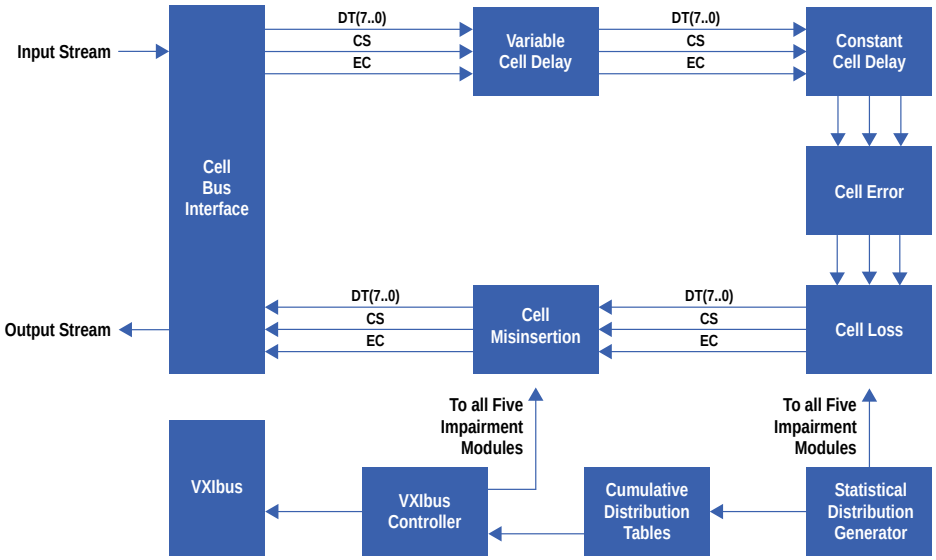


Fig. 3. *Block diagram of the HP E4219A network impairment emulator.*

The cell bus interface takes cells from the "drop" side of the four-bit ECL cell bus and converts them into eight-bit TTL signals. It also receives cells from the last impairment in the chain and places them back onto the "add" side of the cell bus. DT[7..0] are the eight signal lines, CS is a cell start indicator, which goes active at the start of a cell, and EC is the empty cell indicator, which indicates whether the cell contains valid data.

The VXIbus is used to control the logic of each impairment module. The VXIbus controller sets up impairments as requested by the user.

The network impairment emulator mimics various network conditions by inserting impairments onto a line based on a user-defined statistical distribution. The network impairment emulator's statistical information is contained within the *distribution tables*, which hold either normal, binomial, exponential, geometric, deterministic, or user-defined distributions. All distributions are user-selectable. Inside the statistical distribution generator block are pseudorandom number generators that determine the time intervals between impairment events.

Constant Cell Delay Module

The constant cell delay module can delay all or a selected portion of the ATM cell stream by a fixed amount, ranging from one to 79,137 cell times. This means that an OC-3 link with a rate of 155.52 Mbits/s will have a maximum delay of 220 ms, approximately the delay of one satellite hop.

As illustrated in Fig. 4, cells first enter the write block, which determines whether they match the user-specified filter. Delayed (matching) cells are then written to the constant delay RAM, and nondelayed (nonmatching) cells are written to the to the null delay FIFO.

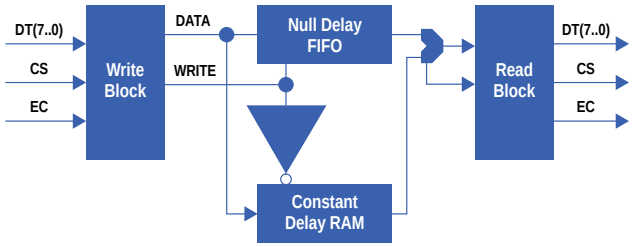


Fig. 4. *Block diagram of the constant cell delay module.*

The constant delay RAM is divided into four 1M-byte blocks using 10-bit bytes. For each byte, eight bits store the ATM cell data (DT[7..0]), one bit stores the empty cell indicator (EC), and the last bit stores the cell start indicator (CS). As delayed cells enter the module they are written in sequential order into RAM.

The constant cell delay module is very straightforward. The write block writes ATM cells (as each is received from the cell bus) to the constant delay RAM. The read block reads from memory locations some increment behind where the write block is writing, and places the cells found there back on the cell bus for transmission. Thus, each ATM cell stays in the constant delay RAM a specified period of time before it is transmitted. This period of time is an integer multiple of the time required to transmit one cell, and is specified by the user.

Cell Delay Variation Module

The cell delay variation module emulates cell buffering, network congestion, and multiplexing delays in ATM networks by varying the amount of delay applied to cells. Depending on the line rate, maximum delays can range from 2.62 ms (with a 155-Mbit/s SONET/SDH line interface) to 220 ms (with a 1.544-Mbit/s T1 line interface). Each cell in the stream is given a different amount of delay. A critical feature is that in an ATM network, cell sequence integrity is guaranteed. This posed a technically interesting problem, since simply delaying each cell to create the desired statistical distribution would put cells out of sequence. This problem was solved by the use of a Markov chain model, which is illustrated in Fig. 5.

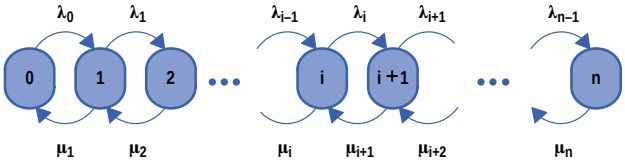


Fig. 5. *Birth-death Markov chain model. The λ_i and μ_i are birth and death rates.*

The network impairment emulator's cell delay variation impairment module is implemented as a birth-death Markov process.[1,2] Each matching cell (a cell whose header values match a user-specified filter) is delayed by a specified amount. This amount is determined by the current state of the Markov chain. For example, a matching cell entering the cell delay variation module when the Markov chain is in state i will undergo a delay of i cell time intervals. To preserve the cell sequence integrity, the state of the Markov chain cannot be changed until the arrival of a nonmatching cell (which includes idle or unassigned cells). On the arrival of each nonmatching cell, random number generators are used to determine whether the Markov chain remains in the same state or changes to a higher or lower state.

Consider the cell sequence $M_1N_1N_2....N_kM_2$ which arrives at the network impairment emulator when the Markov chain is in state n. M_j and N_j denote the jth matching and nonmatching cells in the sequence, respectively. Furthermore, let D_i denote the delay associated with the ith matching cell and T be the cell interval time. Note that the value of D_1 is nT at the arrival time of M_1 and will change to $(n-1)T$ at the arrival time of N_1. The value of D_1 will continue to decrement and will be $(n-k-1)T$ at the arrival time of M_2. Next, consider the possible values of D_2. Since the Markov chain can only change state at the arrival of a nonmatching cell and there are k nonmatching cells between M_1 and M_2, D_2 at the arrival time of cell M_2 is bounded by:

$$(n - k)T \leq D_2 \leq (n + k)T.$$

Therefore, upon the arrival of cell M_2, D_2 is at least one cell interval more than D_1. Hence, cell sequence integrity is preserved.

The implementation of the cell delay variation module using this process does impose some bounds on the delay of two consecutive matching cells. However, this constraint is not expected to be significant since actual traffic conditions are not likely to change greatly. Furthermore, ATM links are typically high-capacity links and the load of each individual virtual channel is expected to be small relative to the overall channel capacity. This implies that for actual network emulation, the percentage of nonmatching cells is expected to be much greater than the percentage of matching cells, thereby reducing the delay dependencies between consecutive matching cells.

The following shows how the steady-state distribution of the Markov chain can be used to implement the cell delay variation distribution. Only the derivation of the binomial distribution with parameters n and p will be shown in detail, where n is the total number of independent trials (or, equivalently, the maximum number of cell-time delays) and p the probability of delay on any single trial. Other distributions can be derived using the same technique by changing the birth rate λ_i and death rate μ_i appropriately.

Let π_i be the steady-state probability of being in state i and let the birth and death rates be defined as:

$$\lambda_i = (n - i)p$$

$$\mu_i = i(1 - p).$$

By flow balance (i.e., the rate of moving from state i to state $i+1$ in the steady state must be equal to the rate of moving from state $i+1$ to state i):

$$\pi_{i+1}(i + 1)(1 - p) = \pi_i(n - i)p.$$

Therefore,

$$\pi_{i+1} = \frac{(n - i)p}{(i + 1)(1 - p)}\pi_i.$$

By recursive substitution, the steady-state probability of each state can be expressed in terms of π_0 as:

$$\pi_i = \binom{n}{i}\left(\frac{p}{1 - p}\right)^i \pi_0. \qquad (1)$$

Since the sum of the total steady-state probabilities must be equal to 1, that is,

$$\sum_{i=0}^{n} \pi_i = 1,$$

using equation (1), we can write the steady-state probability of state zero as:

$$\pi_0 \sum_{i=0}^{n}\binom{n}{i}\left(\frac{p}{1 - p}\right)^i = 1$$

$$\pi_0 = \frac{1}{\left(1 + \frac{p}{1-p}\right)^n} = (1 - p)^n.$$

Using this expression for π_0 in equation 1, we have:

$$\pi_i = \binom{n}{i}p^i(1 - p)^{n-i},$$

which is a binomial distribution with mean np and variance $np(1-p)$.

Implementation of the Cell Delay Variation Module

Given a particular distribution, one of the first processes the network impairment emulator performs is to calculate the state transition probabilities of the Markov chain. These probabilities are determined by the birth and death rates. The state in the Markov chain will increase if there is a birth before a death. Conversely, the state will decrease if a death occurs before a birth. The time between transitions in state is exponentially distributed with mean $1/(\lambda_i + \mu_i)$. Given that there is a change in state, the transition probability is $\lambda_i/(\lambda_i + \mu_i)$ for an increase in state and $\mu_i/(\lambda_i + \mu_i)$ for a decrease in state, respectively. These probabilities, which are written into the table memory, are used together with the random number generator to determine the next state of the Markov chain whenever a nonmatching cell arrives.

Fig. 6 shows a simplified block diagram of the cell delay variation module. Each incoming cell is examined by the write block, then matching and nonmatching cells are written to the matching and nonmatching cell FIFOs respectively. For each nonmatching cell, the random number generator is used together with the table memory to determine the new state of the Markov chain if a state change has occurred. For each matching cell, the state of the Markov chain is sent to mirror memory, which acts as a control to the read block, indicating whether to extract the next outgoing cell from the matching or nonmatching cell FIFO.

Contention between the matching and nonmatching cells may occur at the read block. In such cases, priority is given to the matching cell, and the nonmatching cell is transmitted at the next available slot. This is done to preserve the cell delay variation distribution set by the user as closely as possible. If there is no matching cell to send and the nonmatching cell FIFO is empty, the read block generates an idle cell.

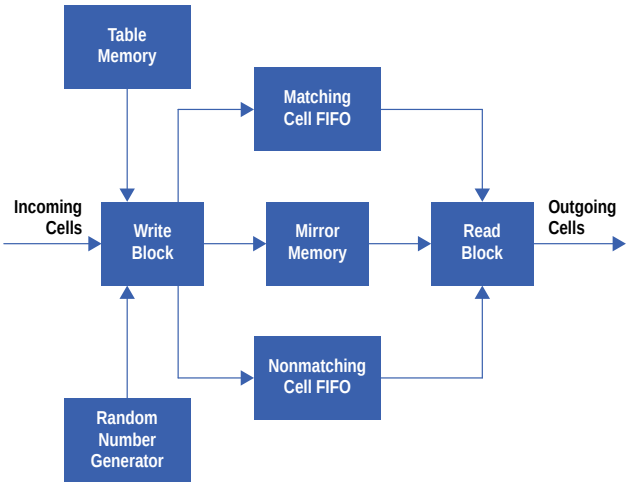


Fig. 6. *Block diagram of the cell delay variation module.*

Cell Error, Loss, and Misinsertion Modules

Because the cell error, loss, and misinsertion modules have similar implementations, they will be described together. All impairments for these modules use a statistical distribution generator to create the specified distribution. This module, illustrated in Fig. 7, contains three pseudorandom generators, all implemented using different polynomials to ensure that the pseudorandom processes are independent.

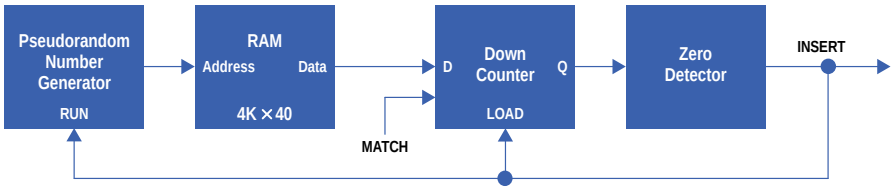


Fig. 7. *Block diagram of a statistical distribution generator.*

The pseudorandom number generator creates 12-bit numbers with a cycle of 2^{17}. The 4K-by-40-bit memory stores the statistical distribution (or more precisely, the cumulative distribution function), with the 40 bits representing the impairment range of 10^{-3} to 10^{-12}. The sequence of events is as follows:

- The statistical distribution is loaded into RAM.
- The pseudorandom number generator generates an address.
- The resulting data is latched and then loaded into the down counter.
- The down counter decrements by one as each cell whose header matches the impairment filter transits the network impairment emulator.
- When the down counter reaches zero, a flag is set to inform the impairment module that the next matching cell should be impaired (cell loss and cell error modules) or a cell should be inserted into the stream (cell misinsertion module).
- The zero detector drives the pseudorandom number generator to create a new number, and the process repeats.

Fig. 8 shows the block diagram of the cell loss, error, and misinsertion modules. Cells enter into a header selector. The MATCH indicator will become active for each cell flowing through the network impairment emulator provided that the user selects all cells to be impaired. Otherwise, it will be active only for selected channels. The MATCH indicator goes to the statistical distribution module.

Cells pass through the impairment module unimpaired if the INSERT signal is not active. If the INSERT and MATCH signals are both active, the cell will be errored, lost, or replaced by an empty cell.

Both the cell error and the cell loss modules have the added capability of introducing bursts of impairments. These impairments are also specified as statistical distributions through the use of a simple probability function. The burst length ranges from one to four bits for the cell error module and from one to eight consecutive cells for the cell loss module. For example, if a user selects a burst length of four with a probability of one for the cell error module, each occurrence of

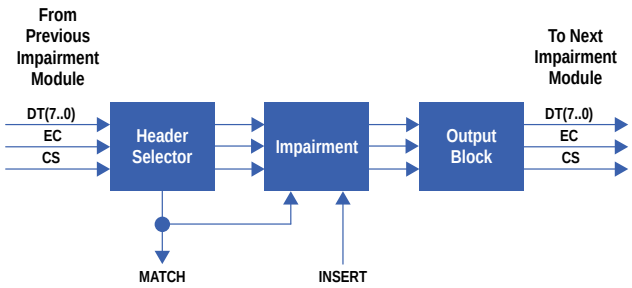


Fig. 8. *Generalized block diagram of the cell loss, error, and misinsertion modules.*

INSERT + MATCH will result in four consecutive errors. By way of further illustration, setting the probability to 0.125 for each of the eight consecutive cell loss events will result in 12.5% of the INSERT + MATCH events being of each length.

The cell misinsertion module inserts cells into the ATM stream by replacing empty cells with nonempty cells whose header (and a part of the payload) are defined by the user. As with the cell error and cell loss modules, the distribution of the inserted cells is defined by the statistical distribution module.

Implementation

With the aid of TI+D engineers, we converted the design to meet HP's manufacturing standards and then added circuity to interface to the BSTS cell bus. The software controlling the TI+D network impairment emulator was rewritten and a new GUI created to make the Hewlett-Packard network impairment emulator control panel appear consistent with the rest of the BSTS products.

The design work was carried out at two sites: Hewlett-Packard's facility in Vancouver, Canada and TI+D's facility in Madrid, Spain. To coordinate the project and to ensure a successful collaboration, we first visited TI+D's site in Spain and outlined the changes they would need to make to their research hardware. We also specified the interface needed between their core ATM impairment logic and our cell bus interface. Once we both had completed our designs, two TI+D engineers traveled to our site in Vancouver to help us debug the design.

Circuit Design

The circuit design had to operate with line interface rates up to 155.52 Mbits/s. Since ATM data is moved in bytes on the circuit board, the resulting system clock is 19.4 MHz. This limitation in clock speed and the need to insert impairments in real time prevented us from considering a microprocessor or DSP-based design. The entire network impairment emulator was implemented using ten large programmable logic parts from Altera and Xilinx and a large number of FIFOs, RAMs, and ECL glue logic. For the Altera designs we used MAX+PLUS II software to create a combination of schematics and text. We used Mentor's Design Architect for the Xilinx designs.

Because of time constraints, the entire design was not simulated as a unit. We used either Altera's MAX+PLUS II or Mentor's QuickSim II to simulate each individual programmable device. The ECL-based cell bus was simulated completely using QuickSim II.

Although this worked reasonably well, a few timing-related issues cropped up concerning timing margins between FIFOs, RAMs, and programmable logic parts that took considerable effort to isolate and correct. In hindsight, a more productive approach would be to simulate the programmable logic and related RAM components together, rather than attempting to debug subtle errors in the lab.

Applications

The following are some typical tests users conduct with the network impairment emulator.

Fig. 9 shows two ATM users communicating through a network that uses a satellite link. Delays from the satellite uplink and downlink are relatively large and fairly constant. Cell delay variation caused by buffering at the ATM switches is also likely, as are bit errors and cell loss. Users can emulate all these impairments simultaneously with the network impairment emulator by setting the appropriate parameters and enabling the constant cell delay, variable cell delay, cell error, and cell loss modules.

Fig. 10 illustrates how the network impairment emulator can determine the acceptable level of quality of service in a video-on-demand application. In this test setup, the video server generates an MPEG-2 video transport stream, which is then transported by the ATM network. At the receiver end, the set-top box decodes the MPEG-2 transport stream and displays the picture on the monitor. One of the most important issues in MPEG-2 technology is the recovery of the system clock. In the above scenario, video service providers or set-top box manufacturers would be interested in knowing how cell delay variation affects the system clock recovery at the receiver end. The network impairment emulator's variable cell delay module can help determine buffer size and other quality of service factors by increasing the standard deviation of the delay distribution until the display visibly degrades.

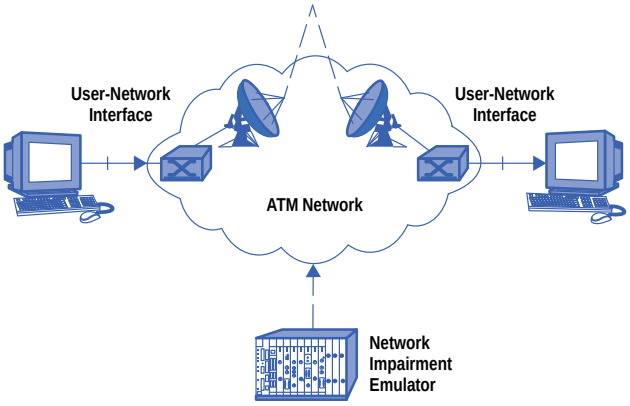


Fig. 9. *Emulation of a satellite link with the HP Broadband Series Test System and the HP E4219A network impairment emulator.*

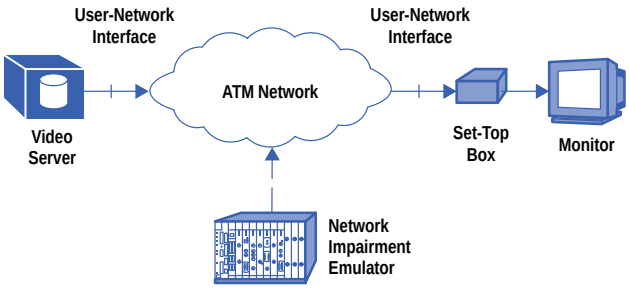


Fig. 10. *Determining acceptable quality of service in a digital video system.*

Conclusion

Hewlett-Packard's third-party collaboration with Telefónica Investigación y Desarrollo of Spain was highly successful. HP's understanding of the ATM market and product commercialization combined with TI+D's technical expertise in ATM networking were merged to produce the world's first 155-Mbit/s ATM network emulator within the targeted market window.

References

1. H.M. Taylor and S. Karlin, *An Introduction to Stochastic Modeling*, Academic Press, 1984.
2. S.M. Ross, *Stochastic Processes*, J. Wiley, 1983.

A Message Handling System for B-ISDN User-Network Interface Signaling Test Software

B-ISDN user-network interface signaling has many different protocol variants and each of them has tens of different types of messages. The message handling system provides a powerful tool for the developer to easily support these variants and messages in the HP Broadband Series Test System (BSTS).

by Satoshi Naganawa and Richard Z. Zuo

Over the past several years, as the "information superhighway" has gained the attention of many countries and communication network service providers, the focus of the Broadband Integrated Services Digital Network (B-ISDN) industry has shifted from providing transmission capabilities via Asynchronous Transfer Mode (ATM) to providing a variety of B-ISDN network services such as private and public B-ISDN, switched virtual connections, and LAN emulation over ATM. Signaling plays a key role in realizing these services. The key functions of signaling are:

- Call setup and disconnection
- Negotiating and agreeing on quality of service (QoS)
- Providing accounting and charging information
- Resolving internetwork and intranetwork routing
- Facilitating basic services and supplementary services
- Identifying the occurrence and cause of any network or service failure.

Signaling consists of user-network interface (UNI) signaling, network-node interface (NNI) signaling for public networks, and private network-node interface (PNNI) signaling for private networks. The UNI is the boundary between a private or public network and the user equipment. User equipment can be a telephone, a computer, or video conference equipment as illustrated in Fig. 1.

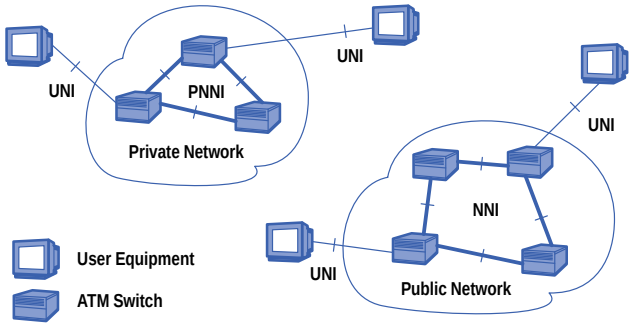


Fig. 1. *User-network interface (UNI), network-node interface (NNI), and private network-node interface (PNNI) signaling.*

The Broadband ISDN UNI signaling test software, HP E4214A, is a new addition to the HP Broadband Series Test System (BSTS). It is designed for R&D engineers and network engineers to develop, troubleshoot, and verify the features and functionality of UNI signaling. Fundamental features of the UNI signaling test software are protocol message decoding and encoding. With decoding, the user can monitor the message transaction between the user equipment and an ATM switch. With encoding, the user can execute a condition test such as constructing a correct or incorrect message and sending it to a switch to analyze the response.

UNI Signaling Standards

As ATM emerges as the mainstream technology in the telecommunication network market, the standardization of UNI signaling has been progressing at a very rapid pace during the last several years. Two major standardization organizations —the International Telecommunications Union Telecommunications Standardization Sector (ITU-T) and the ATM Forum—created their UNI signaling standards in an interlocked way. In May 1993, ITU-T published its first draft recommendation Q.93B[1] to support point-to-point ATM switched virtual connections. The ATM Forum adapted Q.93B to support point-to-multipoint connection control, variable-bit-rate connections, and the ability to signal quality of service (QoS) on a per-call or per-connection basis. These enhancements were published in the ATM Forum UNI 3.0 specification[2] in September 1993. Later, the ITU-T updated Q.93B as Q.2931[3] with some important changes. Following those changes the ATM Forum also updated UNI 3.0 into UNI 3.1[4] in September 1994. In the meantime, the ITU-T defined point-to-multipoint switched virtual connection features and published its standard as Q.2971[5] in June 1995. About half a year later, the ATM Forum published UNI 4.0[6] to incorporate the leaf-initiated join capability and the available bit rate (ABR) service in Q.2931 and Q.2971. In addition to the protocols promoted by the ITU-T and the ATM Forum, some countries and network service providers also created their own UNI signaling protocol variants. From this brief history, it can be understood why there are many UNI signaling protocol variants. Undoubtedly, more variants will come soon to support more new features.

In addition to having many different UNI signaling protocol variants, each variant has many different types of messages. A message is a protocol data unit (PDU) used in a signaling protocol. Each message can contain multiple information elements, called IEs. Some IEs must be contained in a particular message and are referred to as mandatory IEs. Some IEs are optional for a particular message, and therefore will not always appear in that message type. For example, the UNI 3.0 standard[2] has 16 different types of messages and 21 different types of IEs. SETUP, CALL PROCEEDING, CONNECT, and CONNECT ACKNOWLEDGE are the messages for call establishment, as shown in Fig. 2. The other categories of messages are designed for call clearing, point-to-multipoint communication, and other purposes.

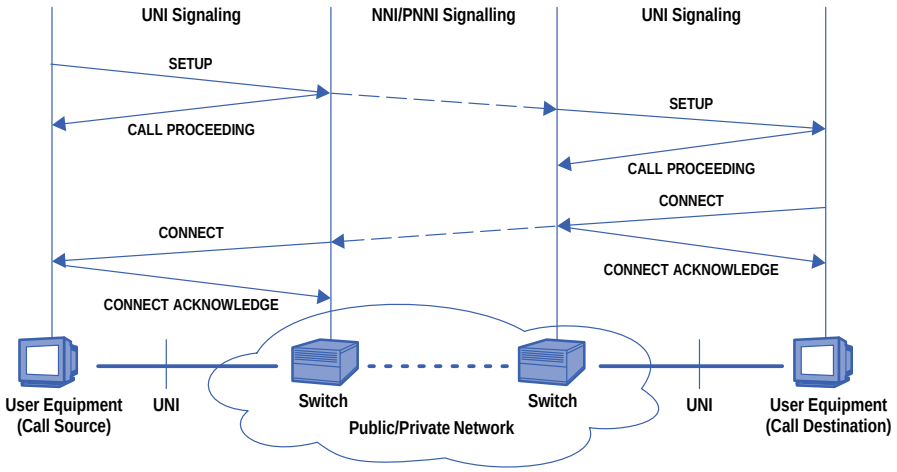


Fig. 2. *Setting up a connection between two user equipments via signaling.*[7]

The situation described above created a challenge for HP's BSTS UNI signaling test software developers. The software needed to support many protocol variants, with many different types of messages and IEs in each variant. Simultaneously, the R&D cycle time was much reduced. We realized that hard-coding each message and IE for each protocol variant in the decoding and encoding engines would make software maintenance a nightmare, would require the programmer to change the decoding and encoding engines whenever adding a new protocol variant, and would be of little help to us in developing NNI and PNNI signaling test software in the future.

Message Handling System

The *message handling system* was developed to solve this problem. The message handling system:

- Separates the structure description of messages and IEs from the remainder of the software
- Specifies the structure descriptions using a protocol-specification-like language in separate script files
- Compiles the script files to generate an object file
- Reads the object file into a database and allows the other parts of the BSTS to access those messages and IEs through library functions.

Fig. 3 depicts the architecture of the message handling system.

The message handling system describes a signaling protocol in one or more text files by means of a script language called the Message Description Language or MDL. A compiler converts the MDL script files into an object file by using the lex and yacc utilities provided by the UNIX® operating system. Like most compilers, the MDL compiler consists of a parser and a

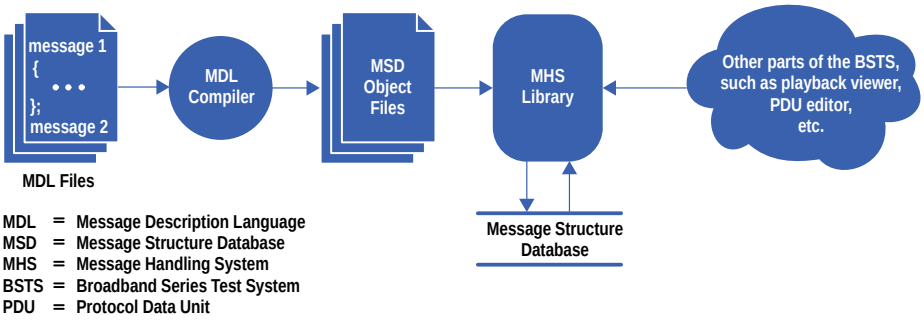
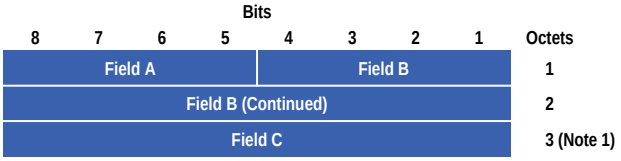


Fig. 3. *Message handling system overview.*

code generator. The parser reads an MDL text file, checks the syntax, and then puts the information in an internal data structure. The code generator generates an object file from the internal data structure.

With ease of use as a top priority, we designed the MDL as a simple protocol-specification-like language that allows a developer to easily translate a UNI signaling protocol specification into a script file. The MDL is also generic enough to handle different types of messages and IEs used in different UNI signaling variants. In the future, such as when using the message handling system for NNI and PNNI signaling, only slight changes are needed in the MDL and its parser.

Fig. 4 shows what the MDL looks like. This is a very simple and artificial example, but one that demonstrates some of the most important common characteristics of all types of messages. This example will also be used later to explain several interesting points in the message handling system structure database. The octet-field map with notations shown in Fig. 4a to describe a message structure is a widely used style in the signaling specifications and is easily translated into an MDL script as shown in Fig. 4b.



(a)

```
message example
{
    name “simpl_example”;

    octet 1 required {
        field Field_A[4]          Field_A_Prm;
        field Field_B[4].[12..9] Field_B_Prm;
    }
    octet 2 required {
        field Field_B[8].[8..1]  Field_B_Prm;
    }

    octet 3 Note_1 {
        field Field_C[8]          Field_C_Prm;}

    rule Note_1 {
        if (decode(Field_A) == 0) {
              return required;
        } else {
              return absent;
        }
    }
}
```

(b)

Fig. 4. *A very simple example of a Message Description Language (MDL) script. (a) The specification. (b) The MDL script.*

The message structure database object file shown in Fig. 3 is designed as a binary disk file version of the message structure database. The disk file format lets us easily separate the MDL compiler from the message handling system library so that a compiler developer doesn't need specific knowledge about the other parts of the test software such as the decoding and encoding engines. Unlike the MDL files, in which the description for a single protocol can be separated into several source files if more convenient, all the information for a single protocol is put into a single message structure database object file to make the run-time loading easier.

The message handling system library in Fig. 3 provides application programming interfaces (APIs) for the message handling system. The other parts of the BSTS can access the protocol-specific information stored in the message structure database through the library functions. Functions are available to:

- Initialize the message handling system library (this includes loading a message structure database object file to generate a message structure database)
- Decode a signaling PDU
- Construct a default signaling PDU
- Encode a signaling PDU.

An example of using the message handling system library is displaying captured PDUs. After a signaling PDU is captured, a message handling system library function is called to extract the values for all fields by using the protocol knowledge stored in the message structure database. Another message handling system library function is called to convert those values into a proper text format. A playback viewer in the other parts of the BSTS can display the formatted decoded result on the screen.

Another example of using the message handling system library is construction of a signaling PDU by the user. A PDU editor in the other parts of the BSTS calls a message handling system library function to create a PDU, which uses the most frequently used values as its default values, and shows the PDU on the screen. After the user makes any modifications to the PDU, the PDU editor calls another library function to put each field value in proper bit and byte order in an octet stream, as described by the protocol specification, to construct a new signaling PDU. During the interaction, the appearance of the user interface can change with the user's input. These dynamic changes are also controlled by message handling system library functions described later.

Message Structure Database

The message structure database shown in Fig. 3 is generated in the application memory space after a message structure database object file is loaded by the message handling system library. The message handling system library can load several different message structure database object files to generate several different message structure databases within the same application. This allows the user to use several different protocols simultaneously.

The message structure database is a set of data structures that define the structure of a PDU and give information about how to decode, encode, and construct a PDU. The message structure database is the core of the message handling system, and it is generic enough to support not only UNI signaling but also NNI and PNNI signaling. The key points in successfully building the message structure database are to correctly analyze the common requirements to express different types of messages and IEs, and to design proper data structures to meet these requirements. Some important requirements are:

- Mapping an octet format of a PDU into a field format or vice versa
- Handling field dependencies
- Handling predefined values for parameters
- Automatic inclusion of mandatory IEs and optional IEs selected by the user
- Automatic calculation of message length.

In the remainder of this section, the mapping and handling field dependencies are used as examples to show in detail what these general requirements entail and how the message structure database meets them.

The bit stream in a PDU can be interpreted into either an octet format or a field format. A single octet can contain part of a field or one or more fields, and in turn, a single field can span part of an octet or one or more octets as shown in Fig. 4a. The octet format is usually used in general computing, such as computing the length of a PDU. The field format is more significant when the user is only concerned with the value of a field but not with the field's bit position. In most cases, such as when decoding or encoding a PDU, both formats are important. Implementation requires three tables—the octet, field, and mapping tables—for each message in the message structure database. Fig. 5 shows how to map the octet format into the

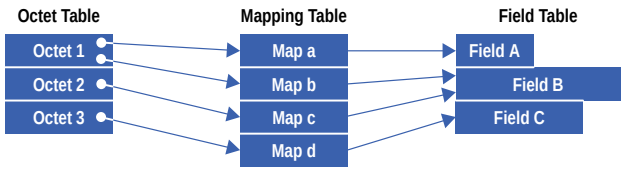
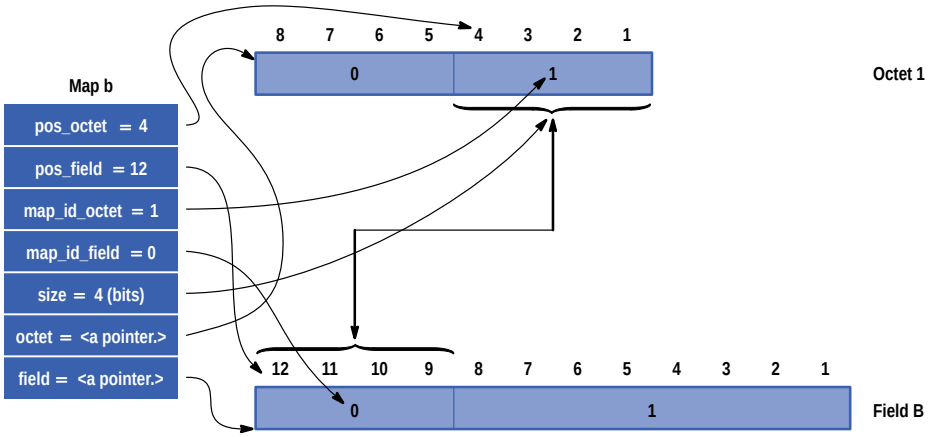


Fig. 5. *Mapping the octet format into the field format.*

field format using these three tables for the example displayed in Fig. 4. Each element in the mapping table is defined by the data structure mhs_msd_map_r shown in Fig. 6a. The definitions of the variables in mhs_msd_map_r are explained in Fig. 6b by showing how map b in Fig. 5 maps the second part of octet 1 with the first part of field B.

```
typedef struct
{
     unsigned int        pos_octet;
     unsigned int        pos_field;
     unsigned int        map_id_octet;
     unsigned int        map_id_field;
     unsigned int        size;
     struct octet_struct *octet;
     struct field_struct *field;
} mhs_msd_map_r;
```

(a)



(b)

Fig. 6. *The mapping data structure and an example. (a) The data structure for a map element. (b) Detail of map b in Fig. 5.*

In a signaling message, modifying the value of a field can change the definitions or values of other fields. This is called a field dependency. Usually the effect of the field dependency needs to be interpreted in real time. For the example shown in Fig. 4, Note 1 indicates that field C is present when the value of field A is equal to 0, and is absent otherwise. Therefore, on the user interface of a PDU editor, the *sensitivity* of field C should be changed with the value of field A in real time as indicated by Fig. 7.

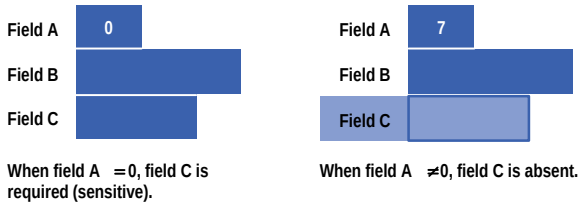


Fig. 7. *An example of field dependency.*

The message handling system internal interpreter was developed to handle field dependencies. The internal interpreter includes two parts: an internal interpreter table for each message in the message structure database to store the field dependency descriptions, and an internal interpreter engine, running a simple and small stack-oriented language, to interpret these descriptions in real time. Fig. 8 shows how the internal interpreter works in this example.

The MDL compiler converts the field dependency description from the MDL script shown in Fig. 8a into an internal interpreter table indicated by Fig. 8b. In this case, the internal interpreter table consists of three blocks of internal interpreter codes. Block[0] indicates how to evaluate the expression in the if(expression) statement. Block[1] will be executed if the expression is true and block[2] will be executed otherwise. Each block contains a sequence of internal interpreter code. Block[0] contains multiple lines of internal interpreter code, while block[1] and block[2] each contain only a single line. A line of internal interpreter code is defined by the structure mhs_mii_code_r, which contains a command and its

```
rule Note_1 {
    if (decode(Field_A) == 0) {
          return required;
    } else {
          return absent;
    }
}
```

(a)

```
typedef struct
{
      commands_e       command;
      unsigned int     value1;
      unsigned int     value2;
} mhs_mii_code_r;
```

```
typedef enum
{
      required = 0,
      optional = 1,
      absent   = 2
} inclusion_type_e;
```

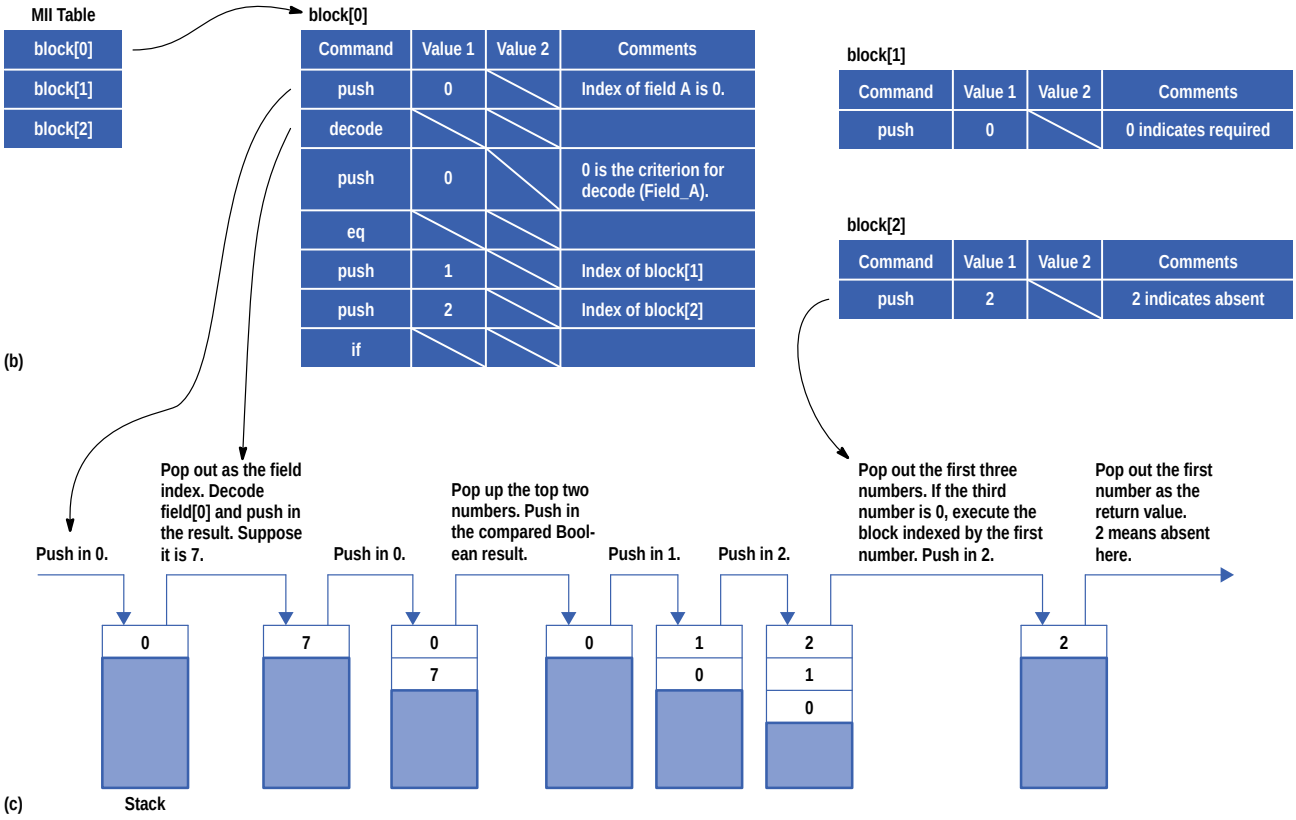


Fig. 8. *An example for the message handling system internal interpreter. (a) The MDL script. (b) The internal interpreter data structure in the structure database. (c) How the internal interpreter engine works.*

corresponding values if applicable. For example, the command push has only a single applicable value located in value 1. The comment columns in each block give the explanation of the values used in each line.

When the user changes the value of field A from 0 to 7, the internal interpreter engine is called to execute the codes started from block[0] to evaluate the expression (decode(Field_A) == 0) as shown in Fig. 8c. The expression is not true in this case, since the value of field A is 7, and therefore block[2] is executed. The absent value, which is defined as 2 in the data structure inclusion_type_e, is returned as the last result of rule Note 1 (Fig. 4a). This makes the PDU editor appropriately change field C from sensitive (required) to insensitive (absent) to meet the real-time interpreting requirement indicated by Fig. 7.

Results

Using the message handling system to add a new UNI signaling protocol variant, the only work that a developer needs to do is to create a script file. In the script files the developer can easily describe the messages in the variant by using the protocol-specification-like language. The message handling system hooks the script file into the other parts of the BSTS automatically, uses the protocol-related knowledge to decode and encode PDUs, and produces the desired results.

In this section, the organization of the general message and one of its instances, the SETUP message, are used as examples to demonstrate the MDL script and the results provided by the BSTS. The SETUP message is one of the most important types of messages used in UNI signaling since it carries all the information for requesting a connection between two user equipments. Fig. 9 shows parts of the protocol specification from UNI 3.0.[2] Fig. 10 is the corresponding MDL script.

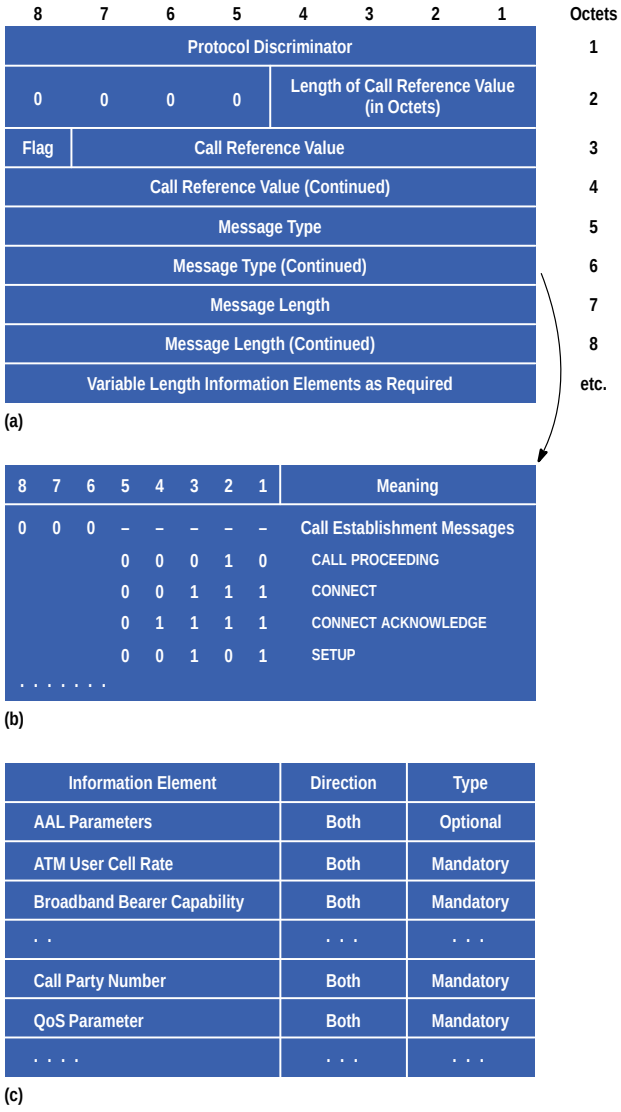


Fig. 9. *General message organization and SETUP message specification.[2] (a) general message organization. (b) Predefined values for message types. (c) IE inclusion of the SETUP message.*

Fig. 9a is the simplified definition of the general message organization. When the value of octet 6 is equal to 5 (binary value 00000101), the message type is defined as SETUP as illustrated in Fig. 9b. Fig. 9c depicts the IE inclusion for the SETUP message. The "Direction" column indicates whether an IE is used in the user-to-network direction, the network-to-user direction, or both.

Fig. 10 shows two segments of the MDL script with C-style comments. Fig. 10a corresponds to the specification of the general message organization shown in Fig. 9a and Fig. 10b is the translation for Fig. 9b and Fig. 9c.

```
generalmessage Message
{
    octet 1
    ....
    octet 6 required {
    /* this octet is required by the
    specification */
        field MessageType[8] MessageType = 5;
        /* MessageType is the variable name.
        [8] indicates the width is 8 bits
        and value 5 (SETUP) is the default
         value */
    }
    ....
    octet 9
    ....
}
```

(a)

```
message SETUP
{
  id   0b00000101;
  /* binary identifier for SETUP message */

  AALParam       both optional;
  /* both directions are optional */
  ATMCellRate    both mandatory;
  /* both directions are mandatory */
  BBearerCap     both mandatory;
  ....
  CalledNum      both mandatory;
  QosParam       both mandatory;
  ....
}
```

(b)

Fig. 10. *MDL scripts. (a) Part of the MDL script for the general message organization. (b) Part of the MDL script for the* SETUP *message.*

Fig. 11 shows the user interface of a UNI signaling PDU editor provided by the BSTS after it hooks into the message handling system. As defined by the specification shown in Fig. 9a and translated by the MDL script shown in Fig. 10a, a message includes the following major parts:

- Protocol discriminator
- Call reference
- Message type
- Message length
- Information elements, as required.

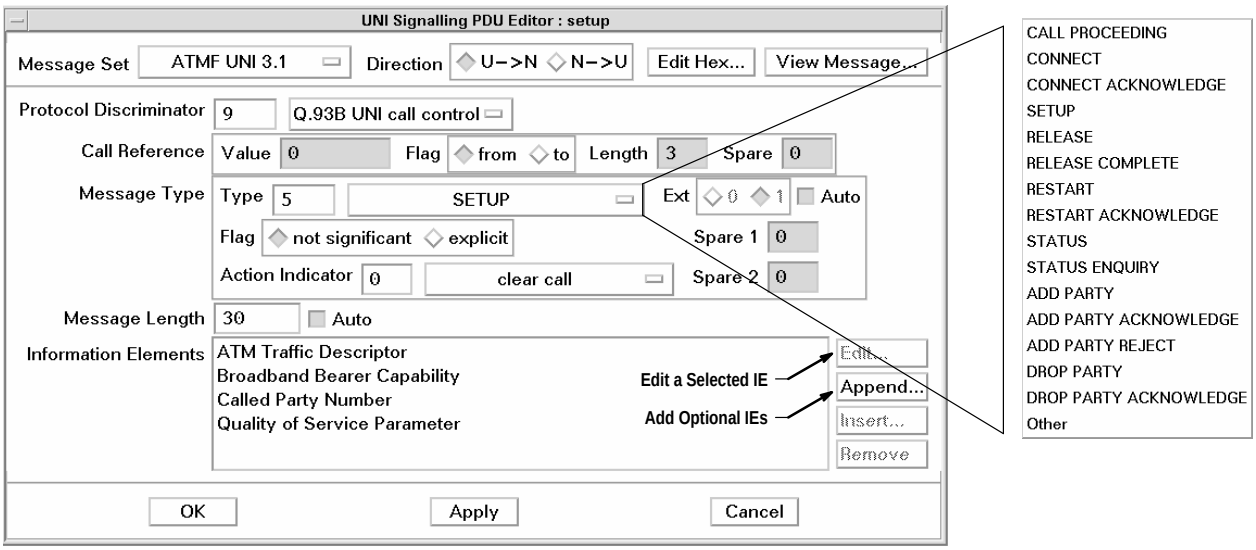


Fig. 11. *Using the PDU editor to construct a* SETUP *message.*

From the option menu in the Message Type item, the user can select any type message to edit. For the selected type, the Type text field automatically displays the corresponding digital value and the Information Elements list shows all the mandatory IEs. In this example, the message has been interpreted as a SETUP message (value 5) and four mandatory IEs have been listed according to the MDL script in Fig. 10b. The user can also edit an IE by clicking the Edit button and add optional IEs by clicking the Append button.

A PDU is constructed by clicking the OK or Apply button. All the field values specified by the user are put in proper bit and byte positions according to the description of the MDL script.

Using the BSTS playback viewer, Fig. 12 shows the decoding result of the constructed message in this example. Fig. 13 illustrates the hexadecimal format of the message. In Fig. 12, the window contents are in two parts: the message header, which includes the first nine octets in the general message organization, and part of the first contained IE, the ATM user cell rate. Notice that the value of octet 6 in the message header is 00000101 in binary, which is interpreted as SETUP according to the MDL script shown in Fig. 10b. With this decoding result, the user can easily monitor the transaction between the user equipment and an ATM switch.

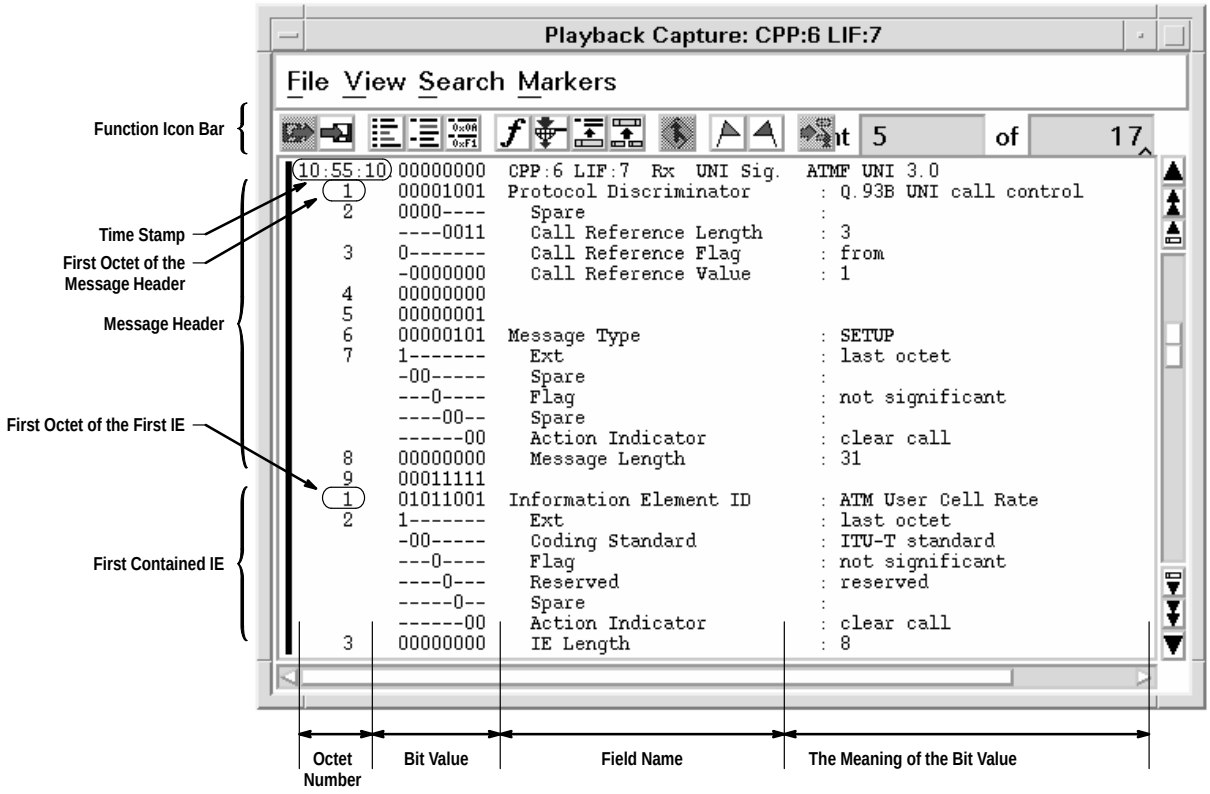


Fig. 12. *Decoding result displayed on the playback viewer.*

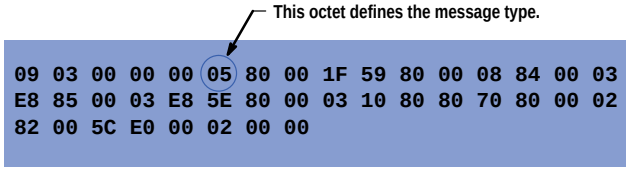


Fig. 13. *The hexadecimal format of a message.*

Summary

The message handling system provides a powerful tool for developers to handle many UNI signaling protocol variants and also many different types of messages and IEs used in each variant. The message handling system significantly improves the maintainability, extensibility, and flexibility of the UNI signaling test software. Because there are a lot of similarities between UNI signaling protocols and NNI/PNNI signaling protocols, it would be possible to reuse the message handling system were we to add NNI and PNNI signaling test software to the BSTS in the future. The message handling system enables HP to keep its competitive position in the signaling test software field, in which old protocols are constantly updated and new protocols are constantly emerging.

Acknowledgments

We would like to thank Mitsuo Matsumoto, who is one of the key developers of the message handling system, and Patrick Siu, who wrote the draft for the introduction and reviewed other parts of this paper. Thanks also to Janet Lomas and Nick Malcolm, who reviewed this paper and gave helpful suggestions for improvement.

References

1. *B-ISDN User Network Interface Layer 3 Specification for Basic Call/Bearer Control*, ITU-T Recommendation Q.93B, May 1993.
2. *User-Network Interface Specification Version 3.0*, ATM Forum, September 1993.
3. *B-ISDN DSS-2 User Network Interface Layer 3 Specification for Basic Call/Connection Control*, ITU-T Recommendation Q.2931, formally approved by ITU in September 1994.
4. *User-Network Interface Specification Version 3.1*, ATM Forum, September 1994.
5. *B-ISDN DSS 2 User Network Interface Layer 3 Specification for Point-to-Multipoint Call/Connection Control*, ITU-T Recommendation Q.2971, June 1995.
6. *User-Network Interface Specification Version 4.0*, ATM Forum, February 1996.
7. A. Scott, *Implementing ATM Signaling: Avoiding the Interoperability Pitfalls*, HP BSTS Solution Note 5963-7514E.

Object-Oriented Network Management Development

As networks continue to proliferate, the need to develop and deploy network management applications has become a critical issue. Two software development tools are described that allow developers to create powerful network management-side applications quickly without necessarily having to be experts on network protocols.

by Peter E. Mellquist and Thomas Murray

The advantages of using an object-oriented design methodology and its associated productivity tools, such as C++,[1] are well entrenched and accepted in today's software engineering practices. One of the areas where the object-oriented paradigm can bring a great deal of productivity is in the development of network management applications.

This article describes two management-side* tools for network management development: SNMP++ and SNMPGen. SNMP++ is a set of C++ classes that provide Simple Network Management Protocol (SNMP) services to a network management application developer.[2] SNMP++ brings the object-oriented advantage to network management programming, and in doing so, makes it much easier to develop powerful, portable, and robust network management applications. SNMPGen is a technology that uses SNMP++ to automatically generate C++ code from the SNMP Management Information Base (MIB) definitions. Fig. 1 shows a conceptual view of these two tools.

SNMP++ was developed by Hewlett-Packard and is a freely available open specification. Anyone can obtain the specification document on the World-Wide Web at URL *http://rosegarden.external.hp.com/snmp++*.

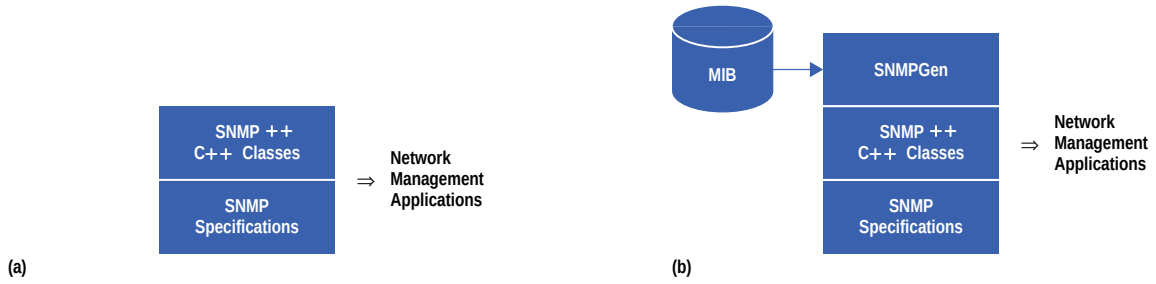


Fig. 1. *A conceptual view of (a) SNMP++ and (b) SNMPGen.*

Benefits of SNMP++

Various SNMP application programming interfaces (APIs) exist that allow the creation of network management applications. The majority of these APIs provide a large library of functions that require the programmer to be familiar with the details of SNMP and SNMP resource management. Most of these APIs are platform-specific, resulting in SNMP code that is specific to a particular operating system or network operating system platform, and thus, not portable.

Application development using a standard set of C++ classes for network management provides many benefits including ease of use, safety, portability, and extensibility.

Ease of Use. SNMP++ is designed so that application programmers do not need to be concerned with low-level SNMP mechanisms. An object-oriented approach to SNMP encapsulates and hides the internal mechanisms of SNMP. SNMP++ provides the following ease-of-use features:

- It provides an interface in which the user does not have to be an SNMP or C++ expert to use its features. For the most part C pointers do not exist in SNMP++, resulting in a simple and straightforward API.
- It provides easy migration to SNMP version 2.[3,4] One of the major goals of SNMP++ was to develop an API that would be scalable to SNMP version 2 with minimal impact on code.

* This is in contrast to agent-side technologies. The manager/agent model, which is used for network management, is analogous to the client/server model for distributed applications.

- It preserves the flexibility of lower-level SNMP programming. A user may want to bypass the object-oriented approach and code directly to low-level SNMP calls.
- It encourages programmers to use the full power of C++.

Programming Safety. Most SNMP APIs require the programmer to manage a variety of resources, such as memory and sockets. Improper allocation or deallocation of these resources can result in corrupted or lost memory. SNMP++ manages these resources automatically. For programming safety SNMP++ addresses the following areas:

- Safe management of SNMP resources. This includes SNMP structures, sessions, and transport layer management. SNMP++ classes are designed as abstract data types providing data hiding and the ability for public member functions to inspect or modify hidden instance variables.
- Built-in error checking and automatic timeout and retry. SNMP++ users do not have to be concerned with providing reliability for an unreliable SNMP transport mechanism. A variety of communications errors can occur, including lost datagrams, duplicated datagrams, and reordered datagrams. SNMP++ addresses each of these possible error conditions and provides the user with transparent reliability.

Portability. Another major goal of SNMP++ was to provide a portable API across a variety of operating systems, network operating systems, and network management platforms (see Fig. 2). Since the internal mechanisms of SNMP++ are hidden, the public interface remains the same across any platform. A programmer who codes to SNMP++ does not have to make changes to move the program to another platform. Another issue in the area of portability is the ability to run across a variety of protocols. SNMP++ currently operates over the Internet Protocol (IP) or Internet Packet Exchange (IPX) protocols.

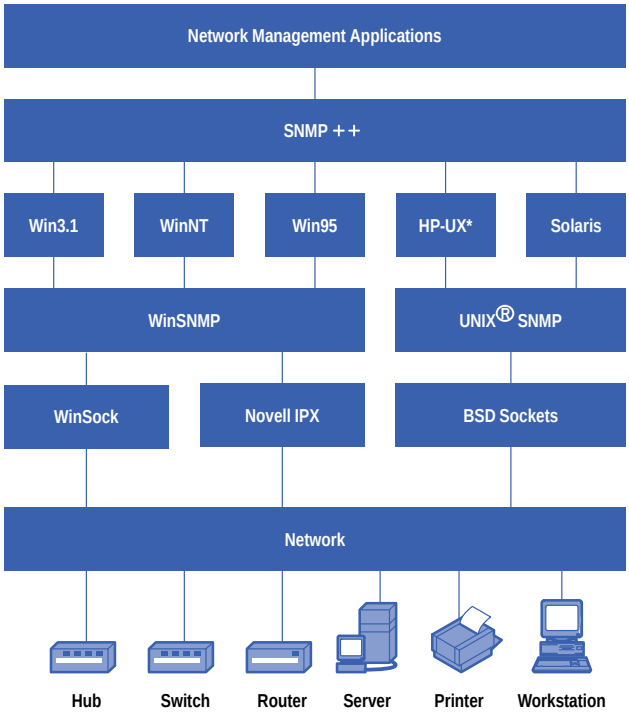


Fig. 2. *An illustration of the different operating systems, network operating systems, and network management platforms to which network management applications based on SNMP++ can be ported.*

Extensibility. Extensibility is not a binary function, but rather one of degree. Extensions to SNMP++ include supporting new operating systems, network operating systems, network management platforms, protocols, SNMP version 2, and other new features. Through C++ class derivation, SNMP++ users can inherit what they like and overload what they wish to redefine.

Overloading SNMP++ Base Classes. The application programmer can subclass the base SNMP++ classes to provide specialized behavior and attributes. The base classes of SNMP++ are meant to be generic and do not contain any vendor-specific data structures or behavior. New attributes can be easily added through C++ subclassing and virtual member function redefinition.

Example

Fig. 3 shows an example of the power and simplicity of SNMP++. This example obtains a system descriptor object from an SNMP Management Information Base (MIB) from the specified managed SNMP agent. Included is all the code needed to create an SNMP++ session, get the system descriptor, and print it out. Retries and timeouts are managed automatically.

```
#include "snmp-pp.h"
#define SYSDESCR "1.3.6.1.2.1.1.1.0"// Object ID for System Descriptor

void get_system_descriptor()
{
  int status                          // return status
  CTarget ctarget((IPAddress) "10.4.8.5"); // SNMP++ v1 target
   Vb vb( SYSDESCR);                  // SNMP++ Variable Binding Object
   Pdu pdu;                           // SNMP++ PDU
// ------ Construct an SNMP++ SNMP Object ------
   Snmp snmp( status);                // Create an SNMP++ session
   if (status != SNMP_CLASS_SUCCESS) {      // check creation status
      cout << snmp.error( status);          // if fail, print error string
      return;  }
//-------- Invoke an SNMP++ Get -----------
   pdu += vb;                         //add the variable binding to the PDU
   if (  (status = snmp.get(  pdu, v1target)) != SNMP_CLASS_SUCCESS)
      cout << snmp.error( status);
   else {
      pdu.get_vb( vb,0);              // extract the variable binding from PDU
                                      // print out the value
      cout << "System Descriptor = "<< vb.get_printable_value(); }
}
```

Fig. 3. *A simple SNMP++ example.*

The SNMP++ calls in this example total only ten lines of code. A CTarget object is created using the IP address of the agent. A variable binding object (Vb) is then created using the object identifier of the MIB object to retrieve the system descriptor (SysDescr). The Vb object is then attached to a PDU (protocol data unit) object. An SNMP object is used to invoke an SNMP Get request. Once retrieved, the response message is printed out. All error handling code is included.

SNMPGen: Machine-Generated SNMP Code

Traditional SNMP development is fairly low-level and redundant. SNMP++ programming requires a thorough understanding of the MIB definition used and the ability to perform manual fabrication of each PDU that is sent and received. When dealing with large amounts of MIB data, manual fabrication becomes tedious and somewhat prone to errors.

Consider, for example, traversing through a column in a MIB table. In SNMP this is performed by loading the object ID of the table column into a PDU and then performing SNMP GETNEXT operations on the table column until the object ID of the variable returned no longer matches the object ID of the table column. Although SNMP++ makes a task like this a little simpler for programmers, there still remains some complexity.

SNMPGen is a set of classes that were created to go beyond the limitations of traditional SNMP development and to further reduce the complexity of developing network management tasks such as the table traversal task mentioned above. These classes allow a higher level of abstraction over SNMP. Rather than deal at the level of PDUs and ASN.1 (Abstract Syntax Notation One) syntaxes, as is typical in SNMP development, SNMPGen offers an abstraction of MIBs and MIB objects. The resulting framework allows the user to write code focused on the manipulation of MIB objects with little concern about the underlying SNMP access details. Additionally, because SNMPGen is based on SNMP++, it can offer all the same advantages as SNMP++.

SNMPGen Classes. A MIB contains managed objects that are represented as tables, scalars, and groups. Each managed object has a name and a unique object identifier and attributes such as syntax and description (see Fig. 4). Tables contain one or more indexes (pointers to other objects) and zero or more managed objects. A scalar represents a single managed object. Groups can contain objects, tables, and other groups.

A MIB also defines the relationships among managed objects. Fig. 5 shows the relationships among SNMPGen classes.

SNMPGen Operations. A set of powerful operations (methods) are available for the various SNMPGen classes. These operations allow the manipulation of objects, columns in tables, entire tables, and groups. Many of the operations are conceptually the same for all of the classes. As a result it takes the same amount of code to get an entire MIB as it does to get a single MIB object. This dramatically reduces the coding effort for writing management applications.

All of the classes have operations to obtain values from an SNMP agent (by performing SNMP GET operations) and operations to send values to an SNMP agent (with SNMP SET operations). Additionally, the classes can have specialized operations that help with manipulating specific types of objects. For example, the CSNMPManagedTable class has operations for manipulating rows and columns in a table.

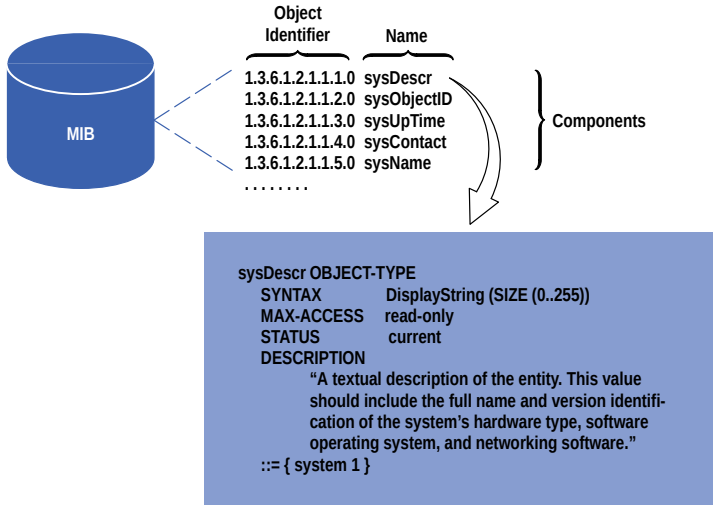


Fig. 4. *Identifiers for the contents of a MIB (Management Information Base) showing object identifiers and the object names. Also shown is the MIB definition of a managed object.*

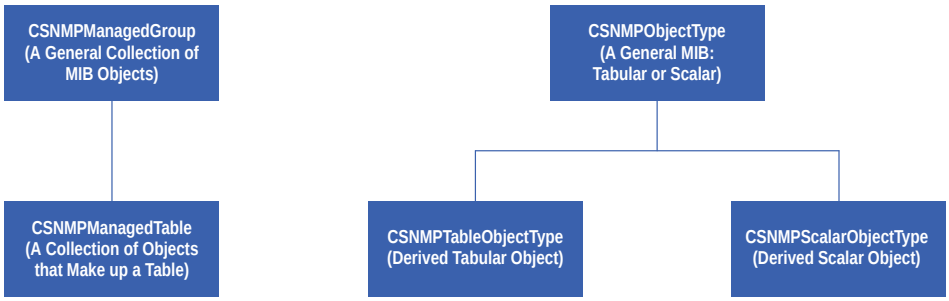


Fig. 5. *The relationships between SNMPGen classes.*

All SNMP considerations are hidden beneath the SNMPGen operations. However, many of the SNMP++ classes are used to manipulate the basic syntax of MIB variables. While nearly eliminating the requirement for detailed SNMP knowledge, this approach provides all the power and flexibility offered by SNMP++.

In addition to the operations provided, the classes can be extended to perform any tasks specialized for a particular application. This can be done simply by deriving new classes from the SNMPGen classes and defining the operations for the new classes.

Mibmgr. SNMPGen classes allow modeling of MIB objects, but they do not give the objects context. For example, instead of having a generic group class, a developer might want a class that represents a group called MIBII* or maybe a system class that represents the system group. A tool called mibmgr was developed to derive classes representing these kinds of specific MIB objects from the classes provided by SNMPGen. The result is a class definition for every object in a MIB. Developers then use these classes in SNMP management applications.

MIB documents tend to be long and somewhat difficult to read. The mibmgr tool was first developed to help the readability of MIBs by displaying them in a simple form. For example, MIBII would be reported as shown in Fig. 6.

This output was created by parsing the output of a MIB compiler tool called MOSY (Managed Object Syntax Compiler yacc-Based), which comes with the SNMP subagent development environment licensed from SNMP Research and maintained by HP's Network System Management Division. MOSY takes a MIB definition as input and produces a compact, easily parsable output file. This output file is then used as input to the mibmgr tool (see Fig. 7).

We quickly realized that this same information could be used in conjunction with the SNMPGen classes to produce C++ header files for SNMP management applications (see Fig. 8). Mibmgr was augmented to produce class definitions for each variable in a MIB. This gives context to the SNMPGen classes by completely defining the object ID space as well as the entire MIB hierarchy.

* MIBII is the second generation of standards for managed SNMP objects as defined by the Internet Engineering Task Force.

```
   Object Name                    Object Type
mgmt
|[1] mib_2
| | [1] system
| | | [1] sysDescr               DisplayString
| | | [2] sysObjectID            ObjectID
| | | [3] sysUpTime              TimeTicks
| | | [4] sysContact             DisplayString
| | | [5] sysName                DisplayString
| | | [6] sysLocation            DisplayString
| | | [7] sysServices            INTEGER
| | [2] interfaces
| | | [1] ifNumber               INTEGER
| | | [2] ifTable                Aggregate
| | | | [1] IfEntry              INDEX "ifIndex"
| | | | | [1] ifIndex            INTEGER
| | | | | [2] ifDesc             DisplayStringr
| | | | | [3] ifType             INTEGER
[...]
```

Fig. 6. *The output produced by the mibmgr for the example MIBII.*

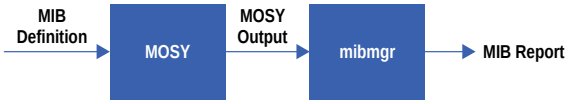


Fig. 7. *The processes involved in creating a MIB report.*

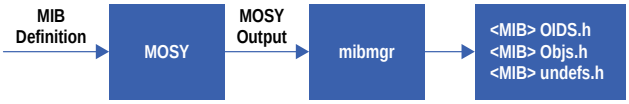


Fig. 8. *The process of automatically generating of C++ header files for network applications using SNMPGen.*

The following code shows an example of a machine-generated scalar object. (Note that mibmgr creates a C++ header file for the entire MIB and only the portion relevant to this example is shown.)

```
#define SsysDescr          "1.3.6.1.2.1.1.1"
 class CsysDescr: public CSNMPScalarObjectType{
   public:
    CsysDescr():
    CSNMPScalarObjectType(SsysDescr,
    sNMP_SYNTAX_OCTETS){};
};
```

Similarly, a machine-generated group would appear as follows:

```
 class Csystem: public CSNMPManagedGroup{
   public:
    Csystem():
    CSNMPManagedGroup(Ssystem){
      AddObject(&sysDescr);
      AddObject(&sysObjectID);
      AddObject(&sysUpTime);
    };
    CsysDescr sysDescr;
    CsysObjectID sysObjectID;
    CsysUpTime sysUpTime; ...
 };
```

This scheme has several benefits. Since the class definitions are produced in C++ header files, any changes to the MIB have little effect on management applications. The C++ header files also allow easy extensions to the classes if needed. Machine generation of class objects greatly reduces the possibility of coding errors and typing mistakes. SNMPGen combined with the machine-generated classes allows the developer to deal only with MIB objects and hierarchies. The use of the machine-generated classes essentially makes a MIB definition the development specification.

```
#include <iostream.h>
#include "mib2objs.h"
main(int argc, char **argv){
int status;                        // result of operation
CsysDescr sysDescr;                // SNMPGen created class from mib2objs.h
int status;                        // result of operation
CsysDescr sysDescr;                // SNMPGen created class from mib2objs.h

if (argc != 2){
cerr << "Usage: " << argv[0] << " <hostname>" << endl;
return(1);   }
IPAddress destination(argv[1]);   // use command line arg as destination
if (!destination.valid()){        // verify we resolved this address
cerr << "Bad hostname: " << argv[1] << endl;
return(1);   }

CTarget target(destination);      // Create an SNMP++ Target
Snmp my_session(status);          // Create an SNMP++ Snmp object

if (status){                      // Ensure the Snmp object is OK
   cerr << "Failed to create SNMP Session" << status << endl;
   return(1);   }

status = sysDescr.SNMPGet(my_session, target);// Access the MIB-II object
   if (status){                                // Check for success
      cerr << "Failed to issue SNMP get: " << my_session.error(status) <<
endl;
      return(1);   }

cout << "SysDescr = " << sysDescr.GetPrintableValue() << endl;  // display
return(0);
}
```

Fig. 9. *A program that uses SNMPGen classes to find a particular host (using a hostname parameter) and returning the MIBII system descriptor.*

Example. Fig. 9 shows a minimal program that takes a hostname as an argument and returns the system descriptor. The example assumes that the mibmgr tool was used to create the header file mib2objs.h from the MIBII specification. A sample output from the example in Fig. 9 is as follows:

```
$ minitest hpisrhx
 SysDescr = HP-UX hpisrhx A.09.01 E 9000/720
 $ minitest hpinddh
 Failed to issue SNMP get: Timed Out While Waiting for Response Pdu
```

The first case worked, and in the second case, no SNMP agent was running on system hpinddh, so the request timed out. The key points to note from this example are:

- Header files include the SNMP++, SNMPGen and its machine-generated files (mib2objs.h includes the SNMP++ and SNMPGen headers), and the standard C++ I/O library.
- SNMP++ target and SNMP classes are required to perform MIB access.
- The machine-generated classes produced by SNMPGen have the same name as defined in the MIB specification, with a C prepended (e.g., CsysDescr).
- Retrieving the object is done using just one operation against the object, which in this case is sysDescr.SNMPGet(). The member function SNMPGet() is inherited from the base class CSNMPObjectType.
- Accessing the retrieved data is also done as a single operation against the object, which in this case is sysDescr.GetPrintableValue(). The member function GetPrintableValue is inherited from the base class CSNMPObjectType.

Practical Uses of SNMPGen

SNMPGen classes combined with the machine-generated classes make it possible for management applications to be developed quickly and efficiently. Engineers writing management applications are frequently not networking experts. The classes allow SNMP management without requiring SNMP expertise. Additionally, when correctly used, the classes can reduce complexity and redundancy in management code.

Several management projects have been based on the SNMPGen classes. The HP ClusterView product, which is used to monitor MC/ServiceGuard high-availability clusters, uses the SNMPGen classes to collect cluster information. The tool was designed and written by engineers with very little networking, SNMP, or even C++ experience. The Web page http://hpgsslhc.cup.hp.com/ClusterView explains this product.

Additional applications include MIB browsing tools and MIB monitoring tools. Since the classes have MIB information embedded within them, they can provide good representations of MIB structures. Additionally, each class can answer questions about MIB object syntax, descriptions, and so on.

Alternatives

Development of network management applications that need to manage network devices or network systems is not an easy task. Often these devices operate in heterogeneous environments where one or more operating systems may be present. This can place additional requirements on a management application. Specifically, an application might need to run on more than one operating system, like the UNIX operating system and Microsoft®Windows.

Before SNMP++, the ability to use a production-quality set of object-oriented SNMP tools did not exist. Without one common set of SNMP libraries, experts are required for each operating system platform and each different SNMP library. Not only does this require additional engineers for development, but testing and maintenance are also adversely affected. SNMP++ allows reused code across any platform that supports ANSI/ISO C++ and supports Berkeley sockets, including all derivatives of the UNIX operating system and Microsoft Windows. Clearly, having to code to and support multiple SNMP APIs is not a viable alternative.

Conclusion

Creating the SNMPGen classes and the mibmgr tool was not a traditional project. It was done as a side job and encountered many obstacles, the foremost being lack of time and resources. The project pioneered the use of C++ development in the division and exposed many issues with the traditional development and integration model. For example, many of the tools had never been used with C++ code before.

The use of SNMP++, SNMPGen, and the machine-generated C++ header files demonstrates the power of object-oriented design and programming. The object-oriented paradigm allows complete freedom for developers. New attributes can be assigned to the class definitions and the classes are fully extensible. The use of these classes greatly reduces development effort. Engineers do not need to know network programming or learn about SNMP to create SNMP-based management applications.

As more and more networked devices and distributed systems are used in today's communication-intensive world, SNMP will play a major role in remote management. SNMP++, SNMPGen, and the classes created with the mibmgr tool will help developers meet the need for network and systems management by enabling rapid development.

References

1. B. Stroustrup, *The C++ Programming Language*, Second Edition, Addison Wesley, 1991.
2. M. Rose, *The Simple Book, An Introduction to Internet Management*, Prentice Hall, 1994.
3. J. Case, K. McCloghrie, M. Rose, and S. Waldbusser, *Coexistence Between Version 1 and 2 of the Internet Standard Network Management Framework*, Internet Engineering Task Force Request for Comment 1452, May 3, 1993.
4. J. Case, K. McCloghrie, M. Rose, and S. Waldbusser, *Structure of Management Information for Version 2 of the Simple Network Management Protocol (SNMPv2)*, Internet Engineering Task Force Request for Comment 1442, May 3, 1993.

SNMP

SNMP (Simple Network Management Protocol) is by far the most popular system and network management protocol.[1] More devices and systems are managed with SNMP than any other management protocol. This is largely because SNMP is quite small and inexpensive to deploy, meaning that it can be implemented in devices with minimal memory and CPU resources. This is in contrast to the Open Systems Interconnect (OSI) management protocols which are complex and more expensive to deploy.

SNMP was developed to provide a basic, easy-to-implement network management tool for the Transport Control Protocol/Internet Protocol (TCP/IP) suite of protocols. This includes a framework of operation and a representation of management information within the framework. The Structure of Management Information (SMI) specification provides the definition of Management Information Bases (MIBs).[2] MIBs are analogous to database schemas. MIB definitions are the basis of interaction between a managed entity and a managing entity (see Fig. 1). The managed entity, called an agent, includes one or more MIBs that define the managed information. This includes a standard set of management information which is part of the SNMP framework, and a set of vendor-specific management information which allows a vendor to instrument specific parts of a device or process. The ability to define custom MIBs allows SNMP management to be extended to meet the needs of a vendor's device. The managing entity, or manager, is responsible for making requests to the agent for management information and handling responses from the agent.

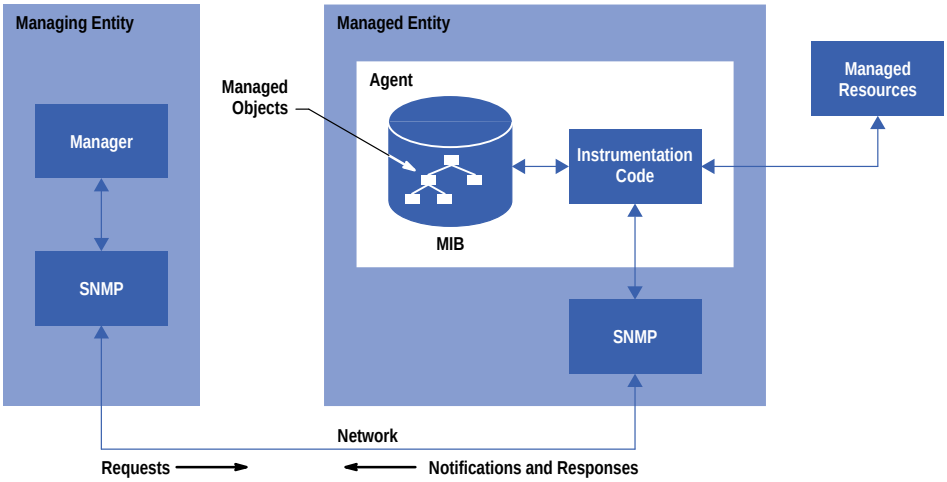


Fig 1. *SNMP and the manager/agent configuration.*

References

1. *Simple Network Management Protocol*, Internet Engineering Task Force Request for Comment 1157, May 1990.
2. *Information Technology—OpenSystem Interconnection—Structure of Management Information*, ITU-Recommendation X.720 (ISO/IEC 10165-1), 1992.

Design of an Enhanced Vector Network Analyzer

A liquid crystal display (LCD) reduces size and weight and has a larger viewing area. TRL (Thru-Reflect-Line) calibration allows measurement of components that do not have coaxial connectors. New software algorithms achieve faster acquisition and frequency tuning of the synthesized source to give faster updates of the measurement data.

by Frank K. David, Frederic W. Woodhull II, Richard R. Barg, Joel P. Dunsmore, Douglas C. Bender, Barry A. Brown, and Stanley E. Jaffe

The development of a new measurement instrument always involves technical challenges. This is also true for the enhancement of an existing family of instruments with the added challenges of getting the enhancements to market in exactly the right form and in a timely manner.

The measurement instrument called a *vector network analyzer* (see Fig. 1) is the integration of a tunable source, a multichannel receiver, and a test set for signal routing and separation, all under the control of an embedded CPU. The block diagram in Fig. 2 shows the interconnection of these components, which for the family of instruments described here are all integrated into one package.

Fig. 1. *HP 8720D vector network analyzer.*

The purpose of a vector network analyzer is to measure the magnitude and phase of the reflection and transmission characteristics of a microwave component (historically called a microwave network) as functions of frequency. The component is inserted between the test ports, the test signal is rapidly tuned over a span of frequency, and portions of this signal are reflected from and transmitted through the component. These reflected and transmitted signals are ratioed with a portion of the test signal that has been tapped off into the reference channel to display the component's reflection and transmission characteristics as functions of frequency. This information is needed by designers to know if their components will work properly when inserted into a microwave system.

The accuracy of the measurement, which depends on separating the component's characteristics from the characteristics of the instrument, is greatly improved by the ratioing process. The accuracy is further enhanced by a calibration process that measures known standards (calibration components) to characterize and mathematically remove the remaining systematic errors in the instrument.

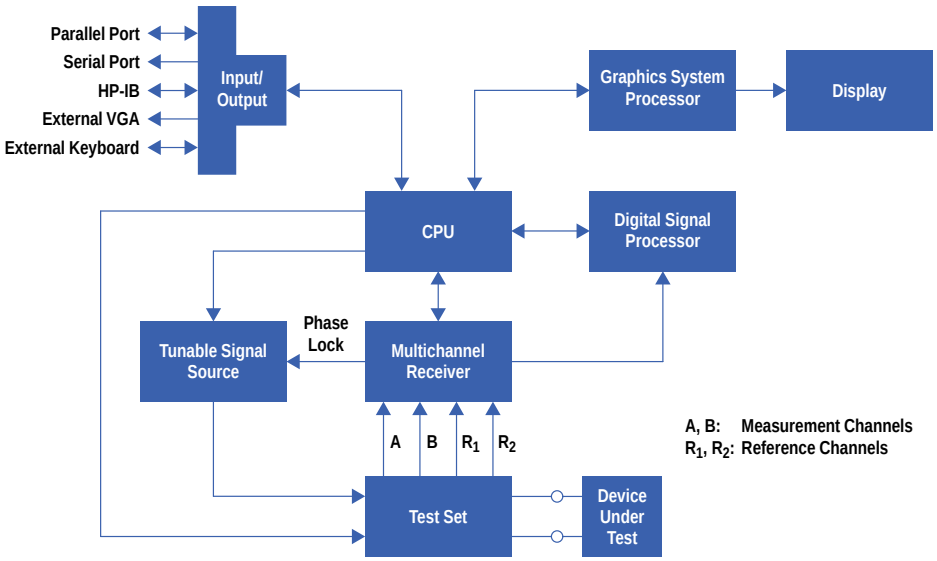


Fig. 2. *HP 8720D vector network analyzer block diagram. Two reference receivers, R_1 and R_2, distinguish the Option 400 model. R_2 is always internal; only R_1 appears (as R) on the front panel.*

The components that are measured with a vector network analyzer are used in a wide variety of commercial and military products: cellular phones, broadcast and cable TV, long-distance telephone transmission, satellite communications, microwave ovens, airplane radar, missile guidance, and so on. Each of these applications uses dozens of components that are developed and produced with the aid of a vector network analyzer. Thus, the accuracy and measurement speed of the vector network analyzer will have an impact on the performance and cost of these products.

The family of HP vector network analyzers known as the HP 8720 family is composed of three members that differ in the frequency range over which they can make their measurements. The HP 8719D, 8720D, and 8722D cover the frequency ranges from 50 MHz to 13.5 GHz, 50 MHz to 20 GHz, and 50 MHz to 40 GHz respectively. The family made its first appearance early in 1988 with the introduction of the HP 8720A and has evolved over the years to the introduction of the D models in May 1996.

The evolution of the HP 8720 family from the A models to the C models was to improve the instruments' hardware performance. The recent evolution from the C to the D models was to respond quickly and directly to a change in users' needs. More and more the instrument was being used in a production measurement role as opposed to the previous R&D role. This change brought with it a need for more measurement speed and expanded I/O capability to allow easier and better integration into a larger production test system. Also, the industry trend of higher levels of integration of components made it necessary to measure components that do not have coaxial connectors—for example, a probe station measuring amplifier chips on an IC wafer. We wanted to respond with something that not only met the new measurement needs, but also had a new look: a state-of-the-art display technology and a smaller, lighter package.

The list of improvements we could have made was long, but we could only choose a few because we wanted to respond quickly. This required careful selection and sorting of the customer inputs that come to us directly or through our field sales force. One difficulty in sorting these inputs was the fact that one fundamental need can manifest itself in several different ways—for example, after thorough investigation, an issue of instrument calibration turned into an issue of measurement speed. Another difficulty was sorting the unique but strongly stated needs from the broadbased but quietly stated needs.

Once the sorting and prioritizing were done, we focused on the most important changes that also allowed us the most leverage from existing designs. When this was done, we proceeded to improve the HP 8720 family with these design changes:

- A liquid crystal display (LCD) gives size and weight reductions plus a larger viewing area. These are important features in a production environment.
- In the HP 8720D Option 400 models, hardware and software additions implement a calibration technique called TRL (Thru-Reflect-Line), which uses four receivers: two reference and two measurement. This allows accurate measurement of components that do not have coaxial connectors. Implementing TRL required some careful multiplexing of the receivers' outputs because the instrument is limited to three data channels.
- New software algorithms achieve faster acquisition and frequency tuning of the synthesized source to give faster updates of the measurement data.

Liquid Crystal Display Electrical Design

There were many advantages to be gained by converting the instrument's display from the traditional color CRT to a color LCD, as can be seen in Table I. In particular, note the significant decrease in volume, weight, and power consumption and the increase in viewing area. However, as beneficial as this conversion is, implementing it in a short time presented some significant challenges.

Table I
LCD versus CRT Display Characteristics

	7.5-in CRT	8.4-in LCD	LCD Advantage
Volume	9745 cm^3	240 cm^3	−97.5%
Weight	5 kg	0.33 kg	−93.4%
Power Consumption	35W	3W	−91.4%
Viewing Area	170.4 cm^2	221.5 cm^2	+30.0%
Horizontal Size	15.6 cm	17.1 cm	
Vertical Size	10.9 cm	12.96 cm	
Aspect Ratio	0.7	0.75	
Pixel Frequency	35.9 MHz	25 MHz	
Horizontal Frequency	25.46 kHz	31.41 kHz	
Vertical Frequency	59.9 Hz	59.8 Hz	
Vertical Resolution	400	480	
Horizontal Resolution	1024	640	

Originally, the display interface was designed to drive a monochromatic, digital, vector CRT display with a state machine controlling stroke and font generators. The state machine executed four basic commands (set condition, plot, graph, character) which were stored in a display list. When the monochrome display was later replaced by a color raster display, the same interface was used, making the transition to color more or less a drop-in replacement (with necessary changes to control the color attributes of lines and text).

With the color raster CRT, the display interface was built around a graphics system processor, which emulates the state machine's stroke and font generators. Supporting blocks for the graphics system processor include DRAM to store the graphics system processor code and display list, VRAM to store the pixel data for the raster display frame and fonts, and a color palette, which uses the pixel data to address a color lookup table which then drives a triple DAC to generate the analog RGB outputs to drive the CRT. The block diagram for this is shown in Fig. 3. The pixel data is 4 bits, which allows 16 simultaneous colors to be displayed. Lookup table registers have 12 bits, which allows selecting one of 4096 colors.

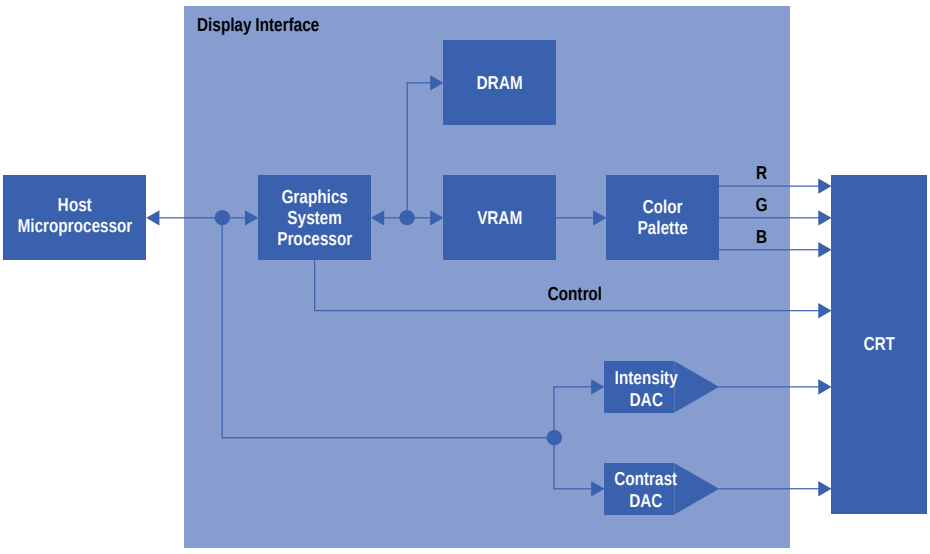


Fig. 3. *CRT graphics system.*

Changing from the color CRT to a color LCD in a short time did not allow creation of a totally new display interface. The focus was to address each of the differences listed in Table I. This resulted in changes of display resolution, timing, aspect ratio, data format, power requirements, and test patterns. The resolution change required changing the x-axis and y-axis scaling in the graphics system processor plot and graph routines and the size of the bitmapped fonts. The display timing change required modifying the values of the graphics system processor timing register value. The change in aspect ratio required changing the host code that generates circles to maintain their roundness. This change affected the aspect ratio of the printer output, which required a modification in the graphics system processor code controlling the horizontal scale factor. The data format change required changing from the CRT's analog RGB data to digital data. The color palette did not provide access to the digital data, so it was replaced with a static RAM for the color lookup table and a triple video DAC, whose output is used for driving an external VGA monitor, as shown in Fig. 4. The power supply had to be changed to the 3.3Vdc required to drive the display, but more important, this voltage had to be withheld from the display until the video synchronization signals were present or the liquid crystal structure would be damaged. The test patterns were changed to allow for testing some of the unique characteristics of an LCD: a black screen checks for pixels that may be stuck on, primary color bars help evaluate the viewing angle, and 16 color bars help evaluate each of the LCD's color bits for troubleshooting the digital interface.

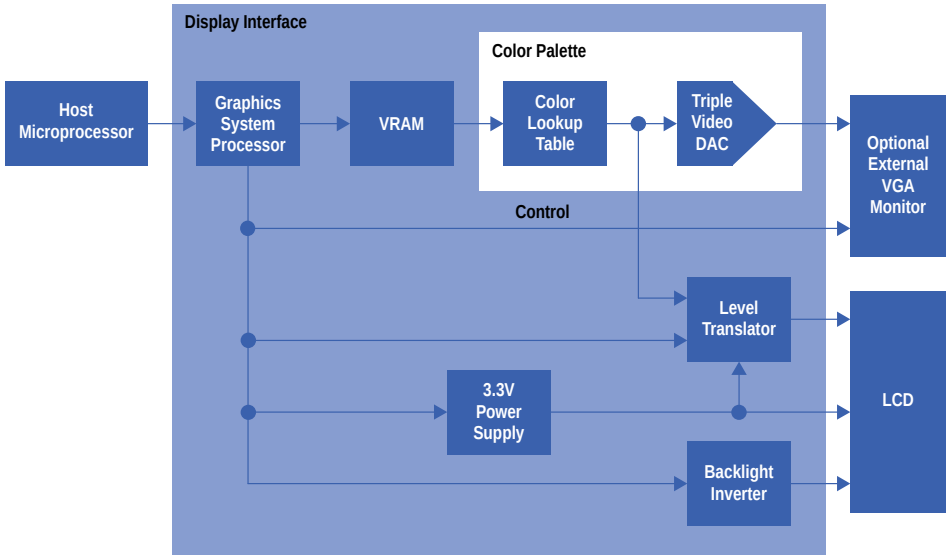


Fig. 4. *LCD graphics system.*

The LCD, which measures 8.4 inches across the diagonal of the active viewing area, was originally designed for use in a laptop computer. Using it in an instrument required several modifications to the display's viewing characteristics to improve general brightness and grayscale contrast reversal (light areas turn dark and dark areas turn light) over a wide viewing angle.

The laptop application optimized its brightness and contrast for a relatively narrow viewing angle (approximately ±20 degrees off the perpendicular in either the horizontal or the vertical direction). Instrument application requires a wider angle, typically ±45 degrees. An instrument display needs to have this wider viewing angle to be free of grayscale contrast reversal. The solution to these problems was to modify the light polarizing and diffusing films that coat the display, and to avoid using colors that are not fully saturated. This restriction on the level of color saturation means that to achieve maximum viewing angle, an LCD that is capable of 4096 colors is limited to a total of eight. This certainly limits the color flexibility, but it is enough for our four-trace display.

One advantage of converting from a CRT to an LCD is that LCDs don't have convergence, focus, geometric distortion, and color purity concerns. This produces a visual image that is sharper and more consistent in color across the entire screen.

Liquid Crystal Display Mechanical Design

The design of the front panel for the HP 8720D vector network analyzer incorporates many features that solve problems associated with the display, including aligning the display with the panel, preventing leakage of electromagnetic fields from the display area, ease of assembly, and space saving.

In past instruments with a CRT display, the display and front panel inherently had poor alignment with each other because of the lack of a common mechanical reference. The LCD design, shown in Fig. 5, solves this problem by integrating into the front panel the display bezel and mounting posts. This allows self-alignment of the display to the bezel. The mounting posts also control the spacing between the LCD and the panel to capture the gaskets and the protective antireflection glass properly.

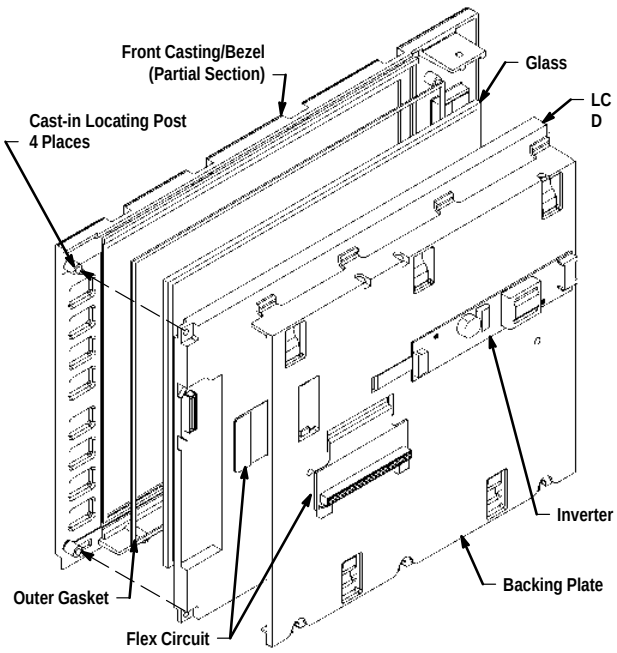


Fig. 5. *The LCD bezel and mounting posts are integrated into the instrument front panel.*

Electromagnetic interference (EMI) from the electrical circuitry in the LCD is a problem because it cannot be shielded by the instrument's sheet-metal enclosure. To solve this, a thin, transparent layer of metal was added to the surface of the glass, and this layer is electrically connected to the metallic front panel by conductive silver strips along all the edges of the glass. To ensure low-resistance, reliable contact, special spring clips attached to the metalized edges of the glass make pressure contact to the front-panel casting. In addition to the electrical connection, the clips also locate and hold the glass in the panel. The net effect is to produce a continuous metal shield over the display without noticeably affecting the display's viewability. To provide additional shielding for the LCD, a special conductive gasket was added to the LCD's metal frame to connect this frame to the edge metallization on the glass. This gasket also hides the LCD's metal frame, which would have been visible in the wide bezel opening necessary to yield the maximum viewing angle of the display's active area (see Fig. 6).

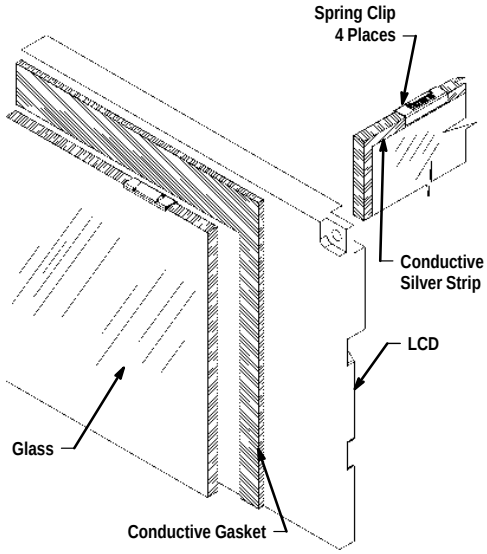


Fig. 6. *To reduce electromagnetic interference, the display has a transparent metal layer on its glass surface, a conductive gasket on its frame, and spring clips and silver strips to connect the glass to the frame.*

The key to the ease of assembly is the LCD backing plate, which has several functions. It holds the LCD to the bezel with five pressure tabs, it provides mounting for the circuit board that drives the display's backlight, it captures the flex circuit that connects the LCD and backlight driver to the graphics processor, and it allows access to the backlight for easy replacement. The assembly of the LCD to the front panel is a simple layering of parts all from one side, and the layered parts are all held in

place by only three screws. This gives a rugged and very compact arrangement. The full depth of the assembly from the front bezel to the back of the mounting plate is less than 21 mm (compared to 317 mm for a standard CRT). This drastic reduction in the depth of the display allowed a 20% volume reduction over previous models and reduced the depth of the instrument, which is very important because it gives the user more working room in front of the instrument.

Three-Receiver Analyzer

To understand the consequences of adding a fourth receiver, it is helpful to review the characteristics of the standard three-receiver version of the network analyzer. The receiver of the network analyzer is synthesized, and the source is phase-locked to the receiver through the reference receiver path, as shown in Fig. 7. In the standard instrument, the reference receiver path is obtained by splitting some of the signal from the main signal path. This same splitting point is used for the automatic level control (ALC) detection point. The main signal path continues to the test set transfer switch, which directs the signal through one of the test port couplers for forward or reverse signal path measurements. The isolation of the switch in the off path creates a cross-talk term, limiting the dynamic range of the transmission measurements.

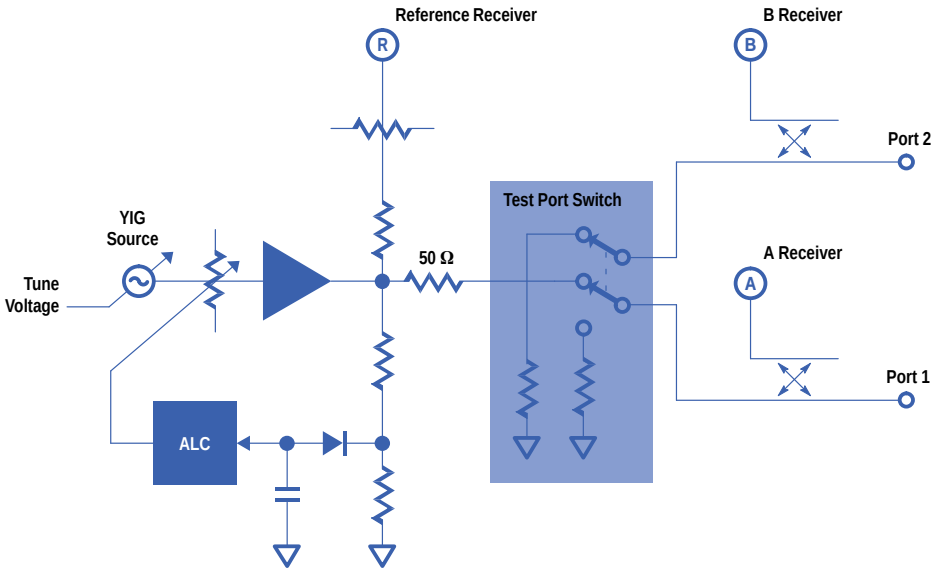


Fig. 7. *Test set for a vector network analyzer with three receiver channels.*

The effects of the system configuration on measurements can be understood in the context of system specifications as seen from the test ports: power source match, ratio source match, load match, test port output power, and test port power flatness.

A key specification for network analyzers is source match, which is a measure of how close the test port impedance is to an ideal 50-ohm real source impedance. If the driving point impedance of a test port is not equal to the ideal reference impedance, measurement errors such as ripples on transmission measurements will become evident. In fact, there are two different source match terms: actual source match, which might be referred to as the *power source match*, and the source match in a ratio measurement, which is referred to as *ratio source match*. Good power source match is important for test port power accuracy, while good ratio source match contributes to accurate measurement of s-parameters. The ratio source match is created by taking the ratio of a test signal to the reference signal. This provides an improvement over the power source match.[1]

For power source match, a sample of the of the source signal is routed to a detector, and the level of the source is controlled to keep the detected signal constant. This has the effect of making the split point a virtual voltage source, thus having a zero impedance. The series 50-ohm impedance in the main path now sets the power source match. The power source match in this case is still not exactly 50 ohms because some signal that does not come from the virtual source node may couple to the detector as a result of high-frequency coupling from the main arm output to the detected output. Also, any physical resistor will have some reactive component, such as series inductance or shunt capacitance. Although minute, these small parasitic elements have substantial effects at high microwave frequencies. For example, 0.1 pF of shunt capacitance has a reactance of 50 ohms at 30 GHz.

Ratio source match is very similar to power source match, in that the sampling of the reference signal makes the node of the sampling point appear to be a virtual source, and the series resistance in the main arm sets the ratio source match. As shown in Fig. 7, the same node is used for both power leveling (ALC) and ratio splitting, and the same series resistor is used for setting both kinds of source match. Because of differing parasitic coupling factors from the main arm to the detector path

and from the main arm to the ratio path, the power source match and the ratio source match will not be identical, especially at high frequencies. For most measurements, the main concern is for good ratio source match.

The *load match* of a network analyzer is the term used for the impedance of the test port that is not the active source. It is easy to see from the block diagram that there is a different path for the load impedance in this state. Therefore, the source match and the load match of a particular port will not be the same.

The insertion loss of the switch reduces the test port signal. The total loss of the switch limits the maximum power available, and the frequency response, or flatness, of the switch determines the test port power flatness. The maximum test port power is determined mainly by the switch loss, but also by the maximum power available from the source amplifier, the loss in the reference and detection splitters, and the loss in the test port coupler. Because the test port switch is positioned between the reference channel and the test channel, any switch repeatability errors, or drift, will be errors in the measurements. Power flatness is determined by how well the detector sensitivity remains constant with frequency. However, because the loss from the power leveling node to the front panel test port increases with frequency, the detector path is designed so that the source power increases with frequency, thus compensating for the loss to the test port.

Incorporating a Fourth Receiver

For the four-receiver version (Option 400), the test port switch needs to come before the reference channel power splitter, thereby creating two reference signals and two test port signals, one for the forward direction and another for the reverse direction (Fig. 8). In this way, the switch repeatability error is "ratioed out" and does not affect measurement accuracy. Current designs use equal-resistance power splitters with a 50-ohm resistor on each leg to obtain the reference signal. The power splitters are integrated into the test port switch. The switch incorporates a shunt p-i-n diode at the splitter node that is forward-biased in the off state such that the node has a low impedance to ground and the off state main-arm impedance is 50 ohms. The reference receiver requires a much smaller signal than the test port, so additional external attenuators are usually added. The loss to the main arm for this configuration is nominally 6 dB. Thus, with no other changes, the maximum test port power and the system dynamic range would need to be degraded by 6 dB. To reduce the power loss in the four-receiver configuration, an unequal splitter was designed to tap off less power for the reference receiver. This decreases the loss through the main arm by about 2 dB, though it does make the power source match a bit worse. The ratio source match in the ratio mode depends upon the design of the splitter circuit (see ***Subarticle 12a***).

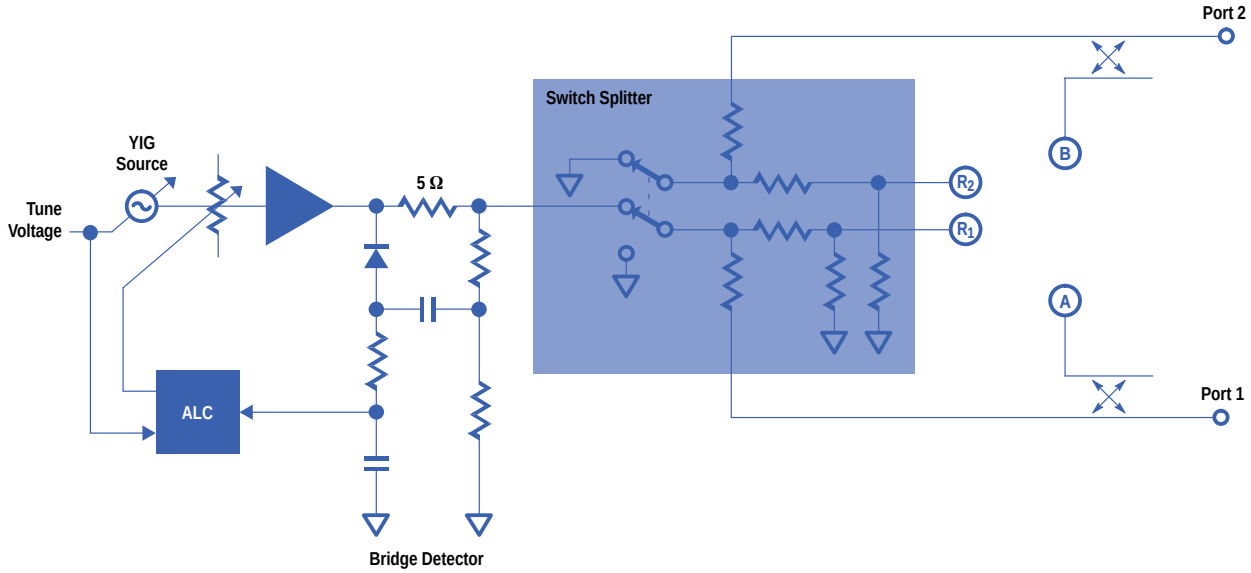


Fig. 8. *Test set for a vector network analyzer with four receiver channels.*

In the three-receiver version of the network analyzer, the splitter for power level detection and the reference receiver shared the same split node and the same series 50-ohm resistor. In the four-receiver version, the power level detection arm is not part of the splitter, so the power loss from this path can be recovered, allowing more power to reach the test port. An alternative method for generating a detected signal was implemented that uses a custom HP GaAs IC.

This chip has an unequal-resistor bridge with a built-in detector diode, which provides the necessary signal to the ALC circuit and has less than 2 dB of loss. The splitter-detector used in the three-receiver version had about 6 dB of loss, so an additional 4 dB of power is recovered by this modification. The detector circuitry is actually integrated into the source amplifier microcircuit, so a new version of this module was introduced with the bridge detector circuit and without the reference RF output. This change, along with the unequal power split in the switch-splitter circuit, makes up for the loss and means that no degradation is needed in the maximum output power or dynamic range.

The integrated bridge detector has a very flat frequency response, so the power out of the amplifier is quite flat. This would normally seem like a good thing, but there is a lot of power loss slope in the switch-splitter and other components on the way to the test port. The detector sensitivity of the IC cannot be sloped, so the power flatness could become quite bad, something like 8 dB of variation from maximum to minimum power at the test port. Because of this, the design of the ALC circuit needed to be modified to accommodate this slope.

The short duration of the project required that the ALC changes not result in any instrument firmware changes to improve the power flatness. This requirement was met by sampling a portion of the tune voltage of the main oscillator, which is linear with frequency, and using it to modify the reference for the ALC level. Some diode shaping and adjustable gain and offset allow a range of adjustments that provide less than 1.5 dB of power variation over the 20-GHz span of the network analyzer.

For the 40-GHz version of the network analyzer, the equal splitter was retained because the greater loss at 40 GHz requires a larger signal to the reference receiver. The 40-GHz source amplifier was modified with the bridge detector circuit to increase the available power by 4 dB, and the final power amplifier chip, again a custom HP GaAs IC, was changed to a newer high-power version that gives 2 dBm more output power. Thus, the output power specification and dynamic range of the 40-GHz analyzer are not degraded for the four-receiver version.

Thru-Reflect-Line Error Correction

One of the classic difficulties in making s-parameter measurements on a device is that a user would like the measurement environment to be as nearly identical as possible to the environment in which the device will ultimately be used. The difficulty is often compounded by a lack of calibration standards for a particular environment, or by the limited ability of the user to create calibration standards that are predictable and reliable for that environment.

The Thru-Reflect-Line (TRL) technique[2,3] has gathered momentum over the past decade because it only requires a known characteristic impedance for a short length of transmission line (the "Line" part of TRL). Furthermore, since an infinitely long transmission line can be represented by terminations or loads of a particular characteristic impedance, the technique can be extended to an entire family of TRL techniques, including Thru-Reflect-Match, Thru-Reflect-Attenuator, Line-Reflect-Line, and so on. Like the HP 8510 network analyzer, the HP 8720D Option 400 instruments incorporate four samplers and a switch-splitter arrangement so that self-calibration using a TRL algorithm can be performed for high-accuracy s-parameter measurements.[4]

A key to the self-calibration technique is the source and load match error removal, also called removal of switching errors. This is accomplished in both the HP 8510 and the HP 8720D Option 400 using the algorithm described by Rytting[4] (see Fig. 9).

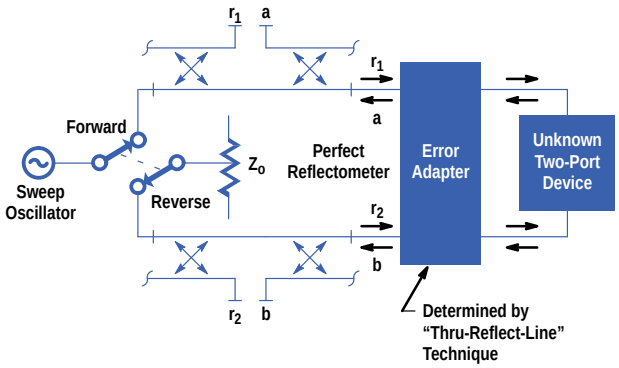


Fig. 9. *Forward and reverse paths and signals measured in the TRL calibration technique.*

Rytting's paper gives equations relating the measured values of the device's s-parameters to the actual values for the generalized condition in which the source and load impedances are not equal to an ideal 50-ohm impedance. Rewriting these equations using notation unique to the HP 8720D family gives:

$$s_{11m} = [(a/r_1) - (a'/r_2')(r_2/r_1)]/d \quad (1)$$

$$s_{21m} = [(b/r_1) - (b'/r_2')(r_2/r_1)]/d \quad (2)$$

$$s_{12m} = [(a'/r_2') - (a/r_1)(r_1'/r_2')]/d \quad (3)$$

$$s_{22m} = [(b'/r_2') - (b/r_1)(r_1'/r_2')]/d \quad (4)$$

$$\text{where } d = 1 - (r_2/r_1)(r_1'/r_2') \quad (5)$$

and the following are complex numbers: s_{knm} is the measured value of s-parameter s_{kn}, a is the A sampler's measurement, b is the B sampler's measurement, r_1 is the R_1 sampler's measurement, and r_2 is the R_2 sampler's measurement.

The prime sign (′) refers to the reverse path as opposed to the forward path of the test set. Ideally, when $r_2 = 0$ and $r_1' = 0$, the switch path has perfect Z_o impedance, $d = 1$, and the s-parameters reduce to the single-term ratios.

In the HP 8720D family, the intermediate frequency (IF) paths for measuring the reference signals r_1 and r_2 are implemented by an IF multiplexing board described later. Because of the architecture of this board, the magnitudes and phase shifts through the various paths are not perfectly balanced. What is important, however, is that all of the ratios are made relative to the same reference IF path.

In the HP 8720D, the ratio a/r_1 is measured using the A IF path and the R IF path, b'/r_2' by the B and R IF paths, and so on. The ratios r_2/r_1 and r_1'/r_2' are measured using only the A and B IF paths.

To establish some notation for this, we will use @ and capital letters to refer to the IF paths used. For example, a@A is the measurement of a using the A IF path. Thus, the five equations above become:

$$s_{11m} = [(a@A/r_1@R) - (a'@A/r_2'@R)(r_2@B/r_1@A)]/d \quad (6)$$

$$s_{21m} = [(b@B/r_1@R) - (b'@B/r_2'@R)(r_2@B/r_1@A)]/d \quad (7)$$

$$s_{12m} = [(a'@A/r_2'@R) - (a@A/r_1@R)(r_1'@A/r_2'@B)]/d \quad (8)$$

$$s_{22m} = [(b'@B/r_2'@R) - (b@B/r_1@R)(r_1'@A/r_2'@B)]/d \quad (9)$$

$$d = 1 - (r_2@B/r_1@A)(r_1'@A/r_2'@B). \quad (10)$$

If you look at these equations carefully, you will note that the measurements of the reference ratios r_2/r_1 and r_1'/r_2' are not made relative to the R IF path, but the other measurements are. Some means for correcting for this difference is needed. The HP 8720D allows measurement of r_2 using both the R and B paths, and measurement of r_1 by both R and A (see Fig. 10). The correction for IF path differences can be reconciled by:

$$(r_2@B/r_1@A)(r_2@R/r_2@B)(r_1@A/r_1@R) = r_2@R/r_1@R \quad (11)$$

$$(r_1'@A/r_2'@B)(r_1'@R/r_1'@A)(r_2'@B/r_2'@R) = r_1'@R/r_2'@R \quad (12)$$

Since r_2 is the same signal to both the R and B paths, the measurement ratios for forward and reverse are equal:

$$r_2@B/r_2@R = r_2'@B/r_2'@R \quad (13)$$

This ratio can be symbolized as $r_2$refcomp for r_2 reference compensation.

Likewise, for r_1:

$$r_1@A/r_1@R = r_1'@A/r_1'@R \quad (14)$$

This can be called $r_1$refcomp.

Rewriting equations 11 and 12, the reference ratios are now compensated for the IF path differences:

$$(r_2@B/r_1@A)(r_1\text{refcomp})/(r_2\text{refcomp}) = r_2@R/r_1@R \quad (15)$$

$$(r_1'@A/r_2'@B)(r_2\text{refcomp})/(r_1\text{refcomp}) = r_1'@R/r_2'@R \quad (16)$$

The original equations 1 through 5 can now be used, as long as the reference ratios are compensated as shown in equations 15 and 16.

10-MHz IF Multiplexing Board

The HP 8720 family of network analyzers was originally designed with three receivers based on the HP 8753 IF processing hardware. The first converter, however, was leveraged from the HP 8510 network analyzer, which has four receivers. Thus, there was a place in the first converter of the HP 8720 where a fourth receiver could be added easily, but there was no fourth channel in the IF processing hardware to connect to its output. Therefore, a four-channel-to-three-channel multiplexing circuit was added to the HP 8720D, between the four first-converter outputs and the three second converters (see Fig. 10). This circuit switches the 10-MHz IF signals so that all four channels can be measured and TRL calibration and error correction can be performed properly in the Option 400 models. In a standard three-channel instrument, this circuit does nothing (the switches are always left in the same positions).

At first glance, it would seem that only the R_1 and R_2 channels need to be switched, and the A and B channels could go straight through. This would allow measurement of all four channels; R_1, A, and B in the forward sweep, and R_2, A, and B in the reverse sweep. Unfortunately, this would not satisfy the requirements of TRL calibration. During the calibration process, it is required to measure the vector ratios R_1/R_2 and R_2/R_1, so the additional switching of Fig. 10 is needed.

Design considerations for the IF multiplexer board were:

- Zero-dB insertion gain
- Harmonic distortion below −50 dBc at −10 dBm (full scale) for good measurement linearity
- Noise low enough to have little effect on the overall instrument noise floor

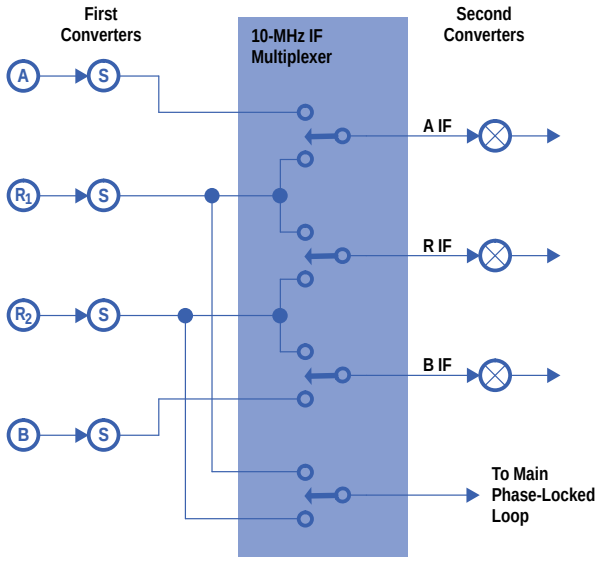


Fig. 10. *Four-channel-to-three-channel IF switching.*

- IF cross talk between channels less than −100 dB.

The circuit design uses a commercial low-noise IC video operational amplifier with a disable input pin. When disabled, the amplifier's output is a high impedance, so two of these ICs can be used as a SPDT switch by simply connecting their outputs together (Fig. 11). This design was preferred over a video switch because it made it easier to achieve high isolation and it has simple, high-impedance input, low-impedance output interfaces. Each amplifier has off-isolation of 60 to 70 dB, so two are cascaded together in each signal path to give greater than 120 dB.

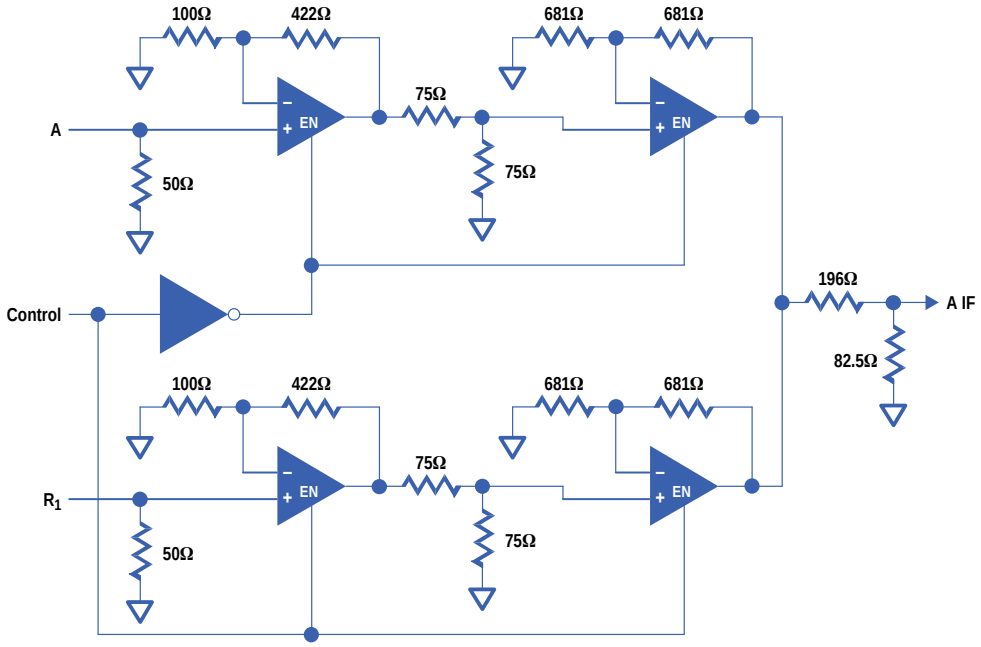


Fig. 11. *10-MHz IF switch.*

The output noise power of the IF switch circuit is −123 dBm in a 3-kHz bandwidth, which is 113 dB below full scale. The overall instrument has typically 100 dB of dynamic range in a 3-kHz bandwidth, so the IF switch contribution is negligible. Second and third harmonics are typically −56 dBc at a −10-dBm output level. Careful consideration of gains and attenuation in the switch circuit was needed to arrive at a good compromise between low noise and low distortion.

The most difficult design goal to meet was the switch off-isolation, which directly affects the instrument's cross-talk specification. Several iterations of the circuit design and board layout were required before the goal was finally met. Careful layout, use of multiple ground planes, and bypassing the power and switch control signals all played a part. But the most vexing problem by far was how the sampler output signals were brought onto the board. These four signals originally

entered the board from the instrument's motherboard through two pin-and-socket connectors, but the best performance that could be achieved with this arrangement was only 90 dB of isolation. It was extremely difficult to attenuate the radiation from the connector pins sufficiently. Eventually, the design team chose to bring the signals onto the board in flexible coax cables, which was deemed a better solution than trying to shield sections of the connectors with elaborate sheet-metal parts. The final design achieves less than −100-dB cross talk between all channels, with typically −120 dB in the most important cross talk paths.

Source Frequency Control Techniques and Improvements

The main components of the HP 8720D network analyzer source are a microwave YIG oscillator, a sampler, a fractional-N synthesizer, a pulse generator, a main phase-locked loop circuit, and a YIG driver, as shown in Fig. 12. Fig. 13 shows the main phase-locked loop circuit.

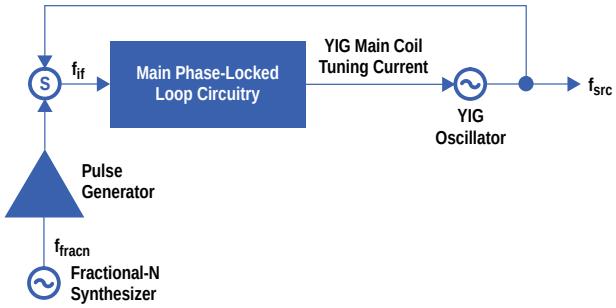


Fig. 12. *Basic HP 8720D source block diagram.*

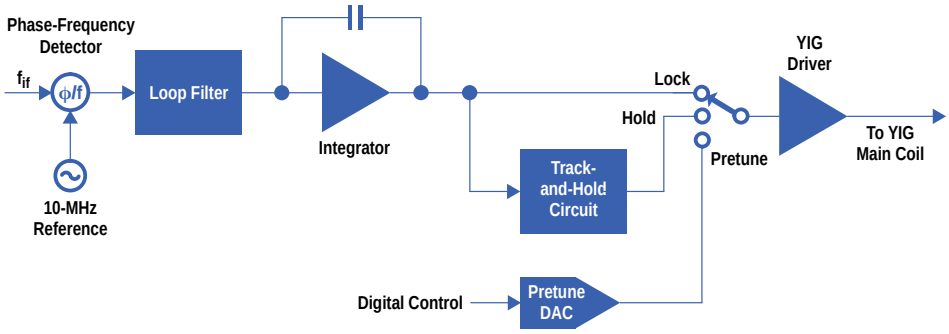


Fig. 13. *HP 8720D main phase-locked loop block diagram.*

The source uses a master-slave technique to control the YIG oscillator frequency based on harmonic multiples of the fractional-N synthesizer output. To set the output frequency, the YIG oscillator is first pretuned to the start frequency by setting a digital-to-analog converter (DAC) to a value that was previously derived during calibration. The output of the DAC is applied to the YIG oscillator driver circuit, establishing the YIG frequency setting. This simple tuning technique is only sufficient for a rough tuning of the oscillator. The output is neither precise enough nor stable enough for network analysis. The fractional-N synthesizer is tuned to a frequency that is determined by the formula:

$$f_{fracn} = \frac{f_{src} + f_{if}}{N},$$

where f_{fracn} is the fractional-N synthesizer frequency, f_{src} is the YIG oscillator frequency, f_{if} is the intermediate frequency (10 MHz), and N is the harmonic number.

For example, for a source frequency of 10 GHz, the fractional-N frequency would be:

$$f_{fracn} = \frac{10^{10} + 10^{7}}{58} = 172.586207 \text{ MHz}.$$

Once the YIG oscillator is pretuned and the fractional-N synthesizer frequency is set, the main phase-locked loop is closed and lock is acquired. The fractional-N frequency is then swept. Since the YIG oscillator is phase-locked to the fractional-N synthesizer, it also sweeps at a rate of N times the fractional-N sweep rate.

One of the limitations of the master-slave tuning technique is the YIG oscillator band crossings. The YIG oscillator can be tuned over a wide frequency range that includes several harmonic bands. The fractional-N synthesizer has a typical range

of 120 to 240 MHz. To sweep the YIG oscillator over its full range requires several band crossings, with relocking of the main phase-locked loop at each band crossing. To avoid pretuning at each band crossing, a sample-and-hold circuit is incorporated as part of the main phase-locked loop. In operation, the fractional-N synthesizer is swept over its range with the YIG oscillator following. Once the fractional-N synthesizer has reached its maximum frequency it must be reset and the YIG oscillator relocked to the next harmonic comb tooth. The YIG oscillator tuning voltage is held constant by the sample-and-hold circuit while the fractional-N synthesizer resets to the start of its range. After resetting, the loop is closed and lock is reestablished at the same frequency, but locked to a new harmonic comb tooth.[5]

The YIG oscillator frequency is determined by the amount of current flowing in its main tuning coil. The transfer function relating main coil current to output frequency is somewhat nonlinear. This makes pretuning the YIG oscillator difficult, since a pretune DAC value must be derived for each pretune frequency. Previous techniques used a multispline, piecewise linear curve fitted to the tuning curve of the oscillator. However, this complicated the pretune calibration process and there was the potential for errors resulting from inaccurate curve fitting.

Another difficulty is YIG oscillator hysteresis. A given amount of current applied to the main tuning coil will result in a different frequency output depending on the previous frequency. For example, if the oscillator had previously been set to 20 GHz, and then 1 mA were applied to the main coil, the resulting output frequency would be several MHz different from the frequency that would result if the previous frequency had been 3 GHz. This hysteresis effect results in inaccuracy when pretuning and potential false locking to the wrong harmonic comb tooth (N).

Improvements in Frequency Tuning and Control

Hysteresis effects of ferromagnetic material are well-understood and were used to advantage in early computers' core memory banks. Hysteresis effects can be minimized by setting the main tuning coil of the YIG oscillator to zero before pretuning. This in effect erases the oscillator's memory of the previous output frequency.

To avoid the nonlinearity of the YIG oscillator tuning curve the technique of pretuning close to the actual start frequency was eliminated. The new technique now pretunes to the YIG oscillator's minimum start frequency regardless of the actual desired start frequency, and the YIG is then swept up to the desired start frequency. This simplifies the pretune calibration process, since only one DAC number is necessary for pretuning. No curve fitting is required, since there is no longer a curve to fit. This pretuning works well with setting the main coil current to zero to minimize hysteresis because the YIG is then already at the minimum of its tuning range.

This tuning technique solves the two difficult problems of tuning curve nonlinearity and YIG tuning hysteresis. However, repeated narrowband sweeps at the upper end of the YIG oscillator's tuning range result in long cycle times, since the YIG must sweep from its minimum start frequency up to its upper frequency limit on each sweep. Setting the main tuning coil current to zero to minimize hysteresis also adds to cycle time, since the time constant of the large-inductance main tuning coil must be taken into consideration. The solution to correct for the long cycle time was to allow for sweeping the YIG oscillator backwards as well as forwards.

To illustrate the technique of sweeping backwards versus pretuning, take the example of a sweep from 19 to 20 GHz. Using the pretuning technique for all sweeps would result in crossing through four bands, with the subsequent delays resulting from four fractional-N frequency resets as well as the long delay required for resetting the main coil current to minimize hysteresis effects. Sweeping backward from 20 to 19 GHz requires no pretuning and no band crossings, and therefore results in a much faster cycle time for network analyzer measurements.

Acknowledgments

The authors would like to acknowledge Dave Sharrit and Bill Pike, the original architects of the instrument platform the HP 8720 has always used. We would like to thank Sue Wood and Dave Blackham for their contributions to the instrument's functionality and accuracy, and Mark Lightner for a heroic and effective marketing introduction. Finally, we would like to thank the production teams that accepted the challenge of turning R&D designs into production reality in such a short time.

References

1. R.A. Johnson, "Understanding Microwave Power Splitters," *Microwave Journal*, December 1975.
2. N.R. Franzen and R.A. Speciale, "A new procedure for system calibration and error removal in automated s-parameter measurements," *Proceedings of the Fifth European Microwave Conference*, September 1975.
3. C.F. Engen and C.A. Hoer, "Thru-Reflect-Line: An improved technique for calibration of the dual six-port automatic network analyzer," *IEEE Transactions on Microwave Theory and Techniques*, December 1979.
4. D.K. Rytting, "An Analysis of Vector Measurement Accuracy Enhancement Techniques," Appendix IX, *Hewlett-Packard RF & Microwave Symposium and Exhibition*, March 1982. Also, Hewlett-Packard Product Note 8720-2, *In-fixture microstrip measurements using TRL calibration*, 1991 and Product Note 8510-8A, *Applying HP 8510 TRL calibration for noncoaxial measurements*, 1992.

5. *Phase-locking Circuit for Swept Synthesized Source Preferably Having Stability Enhancement Circuit*, U.S. Patent #5,130,670.

Optimization of Interconnect Geometry for High-Performance Microprocessors

The goals of the work presented in this paper were to estimate quantitatively the impact of interconnect technology parameters on the performance of high-end microprocessors and to use this information to optimize the interconnect geometry within the constraints imposed by the process. The 64-bit PA 8000 microprocessor was used as a test case.

by Khalid Rahmat and Soo-Young Oh

For the past two decades the driving force for integrated circuits has been scaling of both the devices and the interconnect. This has yielded faster and denser chips with ever increasing functionality. Today's high-performance microprocessors have 4 to 5 million logic transistors[1,2,3] and operate in the frequency range of 200 to 250 MHz. To stay on the trend for CPU performance gains, the next-generation processors will likely have clock speeds of 400 to 500 MHz and have up to 10 million transistors for the logic implementation.[4] The design and manufacture of systems that provide such performance will require careful attention to all process, device, and circuit issues.

In particular, the interconnect system for the next-generation CPUs needs to sustain the burden of connecting 10 million transistors with 5 to 6 metal layers over a 2-cm die with a clock cycle of 2 to 2.5 ns. Because the interconnect system affects the manufacturing yield, reliability, performance, and design complexity of a CPU, it cannot be optimized in isolation but rather must take into account all these aspects.

The goal of the work presented in this paper was twofold. First, we wanted a quantitative estimate of the impact of interconnect technology parameters on the performance of high-end microprocessors. Second, we wanted to use this information to optimize the interconnect geometry within the constraints imposed by the process. It is well-known that interconnect will become an increasingly significant factor in the design of future high-performance processors.[4,5] Therefore, it is important to benchmark the current technology with realistic process parameters and circuit designs. The detailed layout and extracted data for the HP PA 8000 microprocessor[1] provided an ideal testbed to perform both of these tasks.

Two criteria were used for optimization: delay or rise time (they have very similar characteristics) and cross talk noise.[6,7] The optimization was performed only for the global (block-to-block) routing consisting of metal layers 2, 3, and 4. Because the longest signal wires are usually at this level of routing, optimizing these layers has the most direct impact on system performance. For optimization, we only considered nets that had marginal timing based on the expected performance target for the processor.

Optimization Methodology

Our optimization approach is as follows:

1. Determine the timing-critical nets using a global static timing simulator.
2. From this set of nets select a sample that has significant global interconnect.
3. Extract the driver sizes and wire lengths for this interconnect-dominated subset of the global nets.
4. For these global interconnect dominated nets, determine the sensitivity to all interconnect parameters: metal thickness, width, and space and dielectric thickness.
5. For two of the most sensitive parameters perform two-dimensional optimization to determine the best design point for each metal layer considered (metal 2, 3, and 4).

The total wire capacitance and cross talk are calculated assuming a dense geometry with neighboring lines within the plane of the signal line and ground planes above and below, as shown in Fig. 1. The wire capacitance is obtained using the field solver RAPHAEL in 2D, and the cross talk and delay are obtained from SPICE simulations using the coupling and total capacitances generated from RAPHAEL. The cross-talk noise and delay were simulated in worst-case situations. For delay this means that the neighboring lines were switching simultaneously in the direction opposite to the signal line, and for cross talk the victim line was quiescent while the neighboring lines switched simultaneously in the same direction.

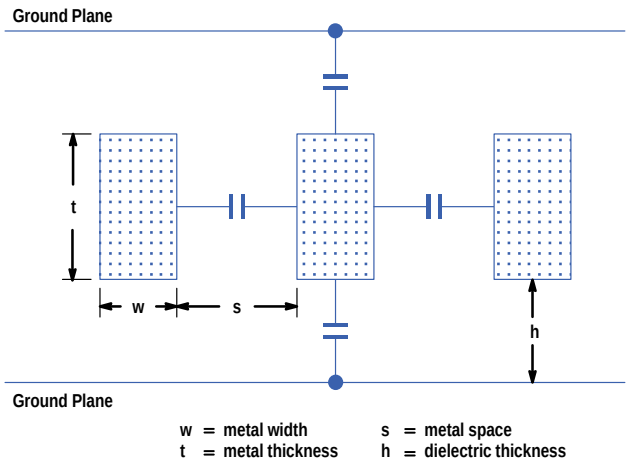


Fig. 1. *Schematic of the interconnect parameters considered for optimization.*

Results

Fig. 2 shows the distribution of metal lengths among the nets that failed the timing criterion at the target frequency. A significant number of critical nets have long metal 2 and metal 4 lines while metal 3 is usually much shorter. This is, of course, a direct consequence of the orthogonal channel routing scheme used by the HP-proprietary automatic router, PA-ROUTE. The other salient feature of Fig. 2 is the large number of nets that have 5 to 6 mm of metal 2. Similarly, there is a secondary peak of 4 to 5 mm for metal 4. Thus, an obvious target for optimization is wires that have this typical long length. We did not select the longest wires for optimization because they were not typical.

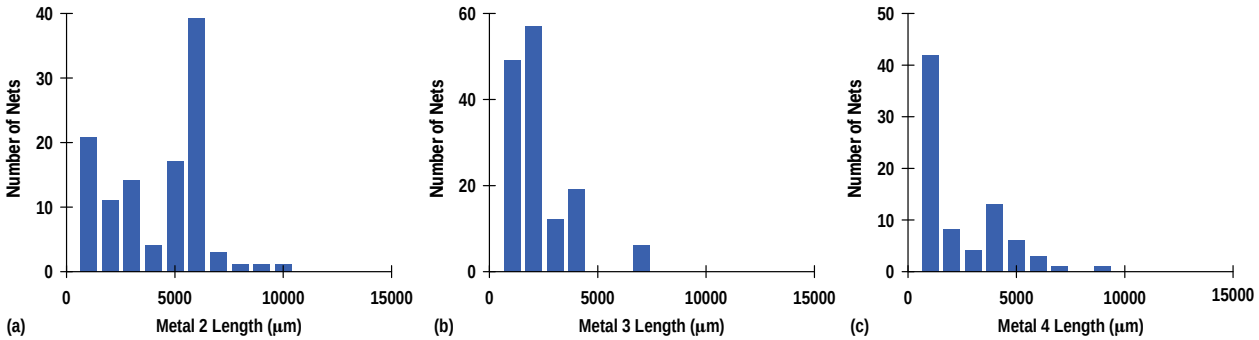


Fig. 2. *Wire length distribution for critical nets. (a) Metal 2. (b) Metal 3. (c) Metal 4.*

The drivers for all of the nets studied were approximately 44 μm wide (W_n = width of n-channel transistor = 44 μm; W_p = width of p-channel transistor = $2W_n$). The drivers were large enough that the delays and rise times were determined primarily by the interconnect and not the driver size.

Metal 2 Optimization. Using the strategy outlined above, we considered net1, which has a length of 6.8 mm in metal 2, 3.6 mm in metal 3 and 0.9 mm in metal 4. Fig. 3 shows the impact on cross-talk noise and rise time (at the load) of varying only one interconnect parameter while keeping all others constant. The nominal values for the metal 2 parameters are:

- Metal width = 1.2 μm
- Metal thickness = 0.7 μm
- Metal space = 1.2 μm
- Dielectric thickness = 0.95 μm.

It is clear from Fig. 3 that the two metal 2 parameters to which the rise time is most sensitive are the metal width and thickness. Thus, the next step in our optimization strategy varies these two parameters simultaneously to generate a two-dimensional contour plot of rise time and cross-talk curves, as shown in Fig. 4. When we vary the metal 2 width we keep the pitch constant, which is a more realistic situation than allowing the pitch to vary, too. To reduce the rise time, either the width or the thickness (or both) of metal 2 lines must be increased. For example, going from width = 1.2 μm, space = 1.2 μm, and thickness = 0.7 μm to width = 1.4 μm, space = 1.0 μm, and thickness = 0.8 μm will yield a reduction in rise time of almost 150 ps (14%). From Fig. 4, it can be estimated that this change will produce an increase in worst-case cross-talk noise of less than 100 mV to about 0.52V which may still be acceptable.

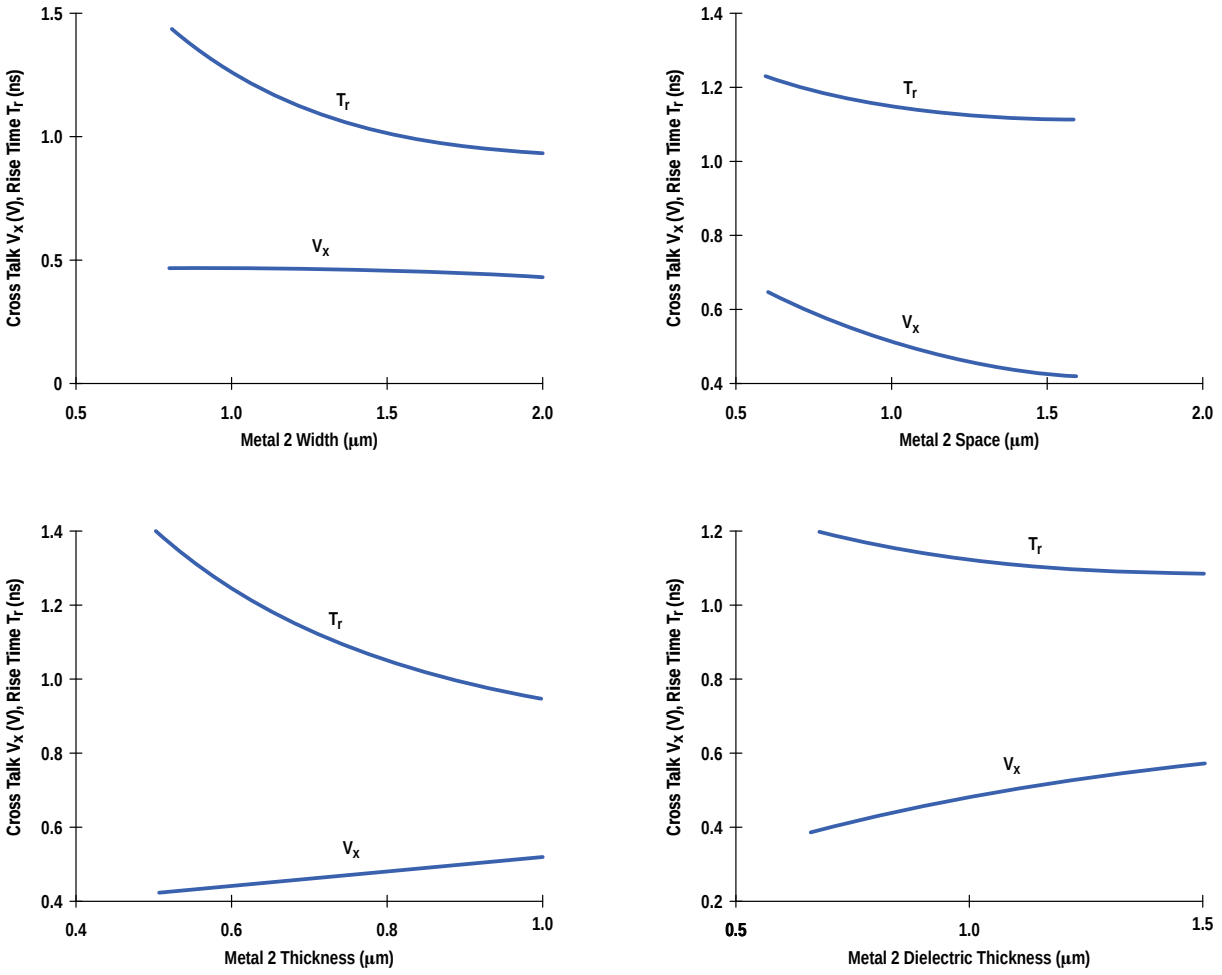


Fig. 3. *Sensitivity of net1 rise time T_r and cross talk V_x to metal 2 parameters.*

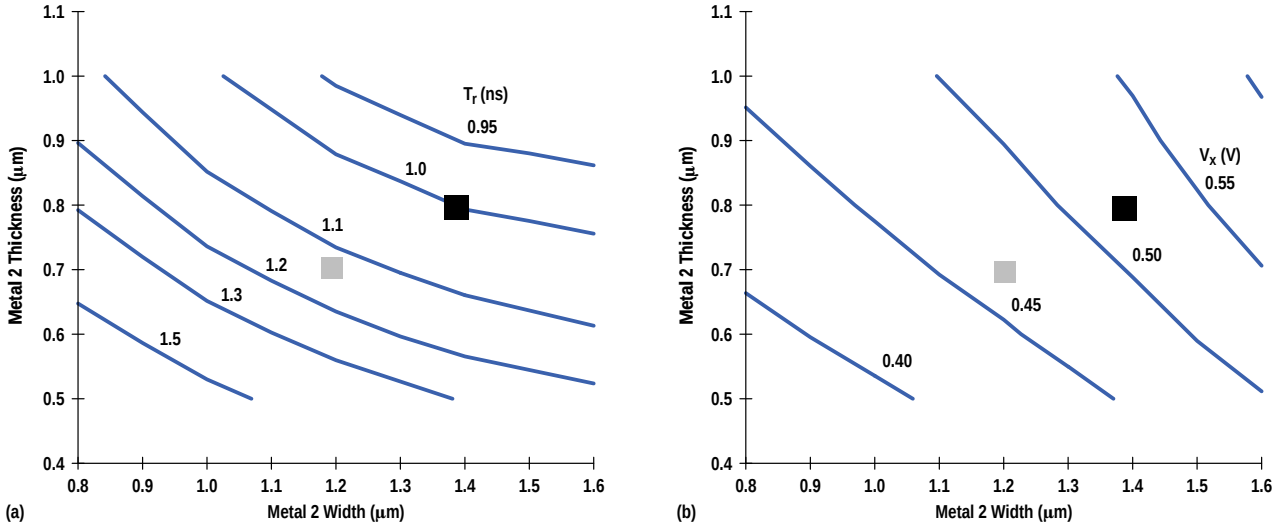


Fig. 4. *Contours of constant rise time (a) and cross talk (b) for the metal 2 dominated net1. The nominal design point is indicated by the gray square while the possible optimized point is shown in black. The pitch (width + space) was kept constant at 2.4 μm.*

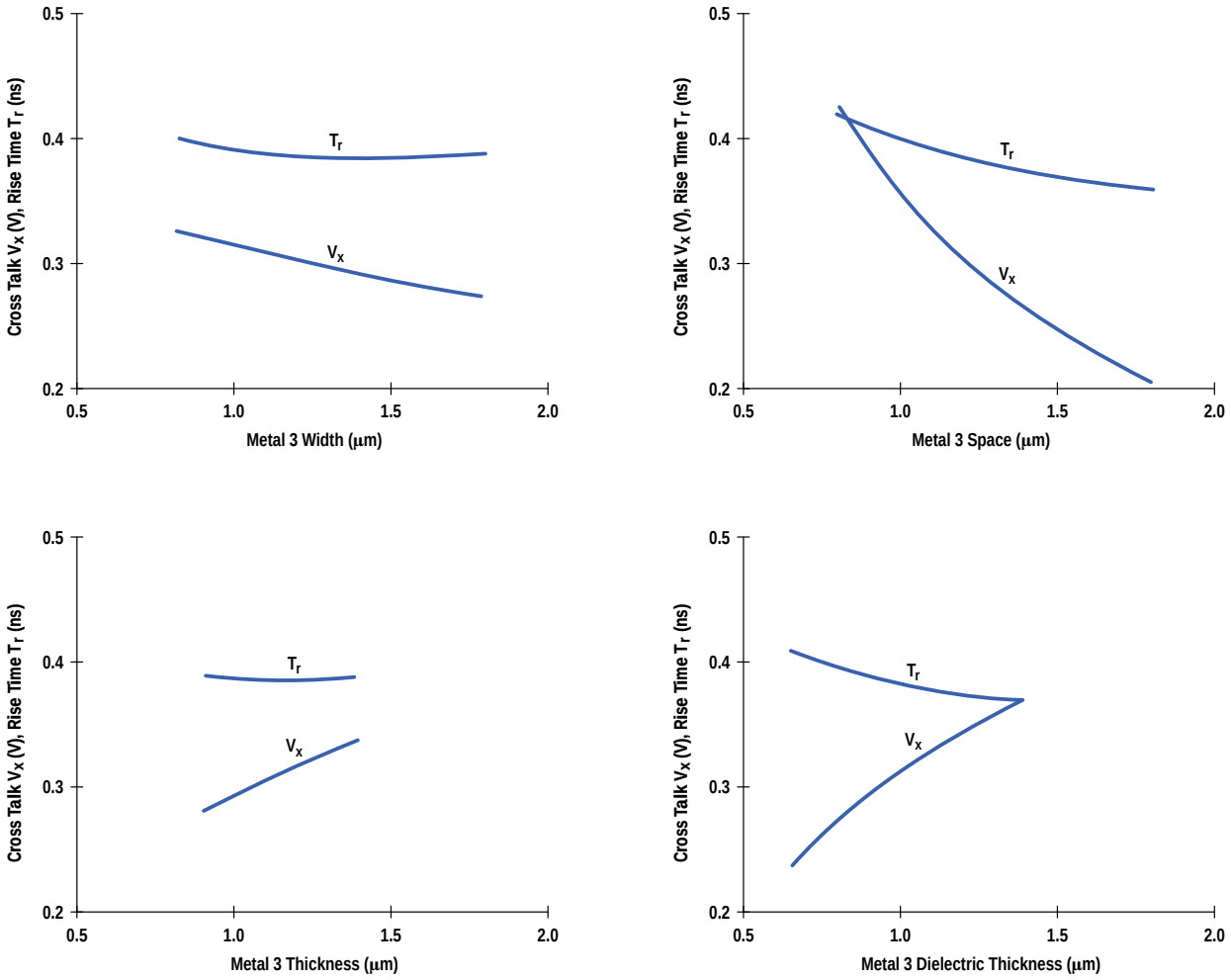


Fig. 5. *Sensitivity of net2 rise time T_r and cross talk V_x to metal 3 parameters.*

Metal 3 Optimization. We proceed here just as in the case of metal 2 by first choosing a net that has significant length in metal 3. In this case this is net2 which has 0.7 mm of metal 2, 3.3 mm of metal 3, and 2.3 mm of metal 4. The nominal metal 3 parameters are:

- Metal width = 1.2 μm
- Metal thickness = 1.1 μm
- Metal space = 1.2 μm
- Dielectric thickness = 0.95 μm.

Because of the relatively short length of the metal 3 lines, the rise time is not very sensitive to metal 3 parameters, as shown in Fig. 5. Of the four parameters, the sensitivity is greatest to the metal 3 spacing and the thickness of the dielectric. As before, we choose these two parameters (while keeping the metal 3 pitch constant) to perform a two dimensional optimization as shown in Fig. 6. The rise time can be improved by using a slightly narrower width (width = 1.0 μm, space = 1.4 μm), but the absolute improvement is small (20 ps).

Metal 4 Optimization. For metal 4 we used net3, which has 1.3 mm of metal 3 and 6.2 mm of metal 4. The nominal metal 4 dimensions are:

- Metal width = 1.6 μm
- Metal thickness = 1.2 μm
- Metal space = 2.4 μm
- Dielectric thickness = 1.05 μm.

For metal 4, the rise time is most sensitive to the metal spacing and the dielectric thickness as seen in Fig. 7. However, for the nominal metal 4 spacing of 2.4 μm, the rise time is a fairly smooth function of spacing, so the spacing between metal 4 lines could be reduced (while keeping the width constant) by up to 0.4 μm without increasing the delay significantly. Thus, the pitch could be reduced without adversely impacting the delay. If the goal is to reduce delays, then two-dimensional

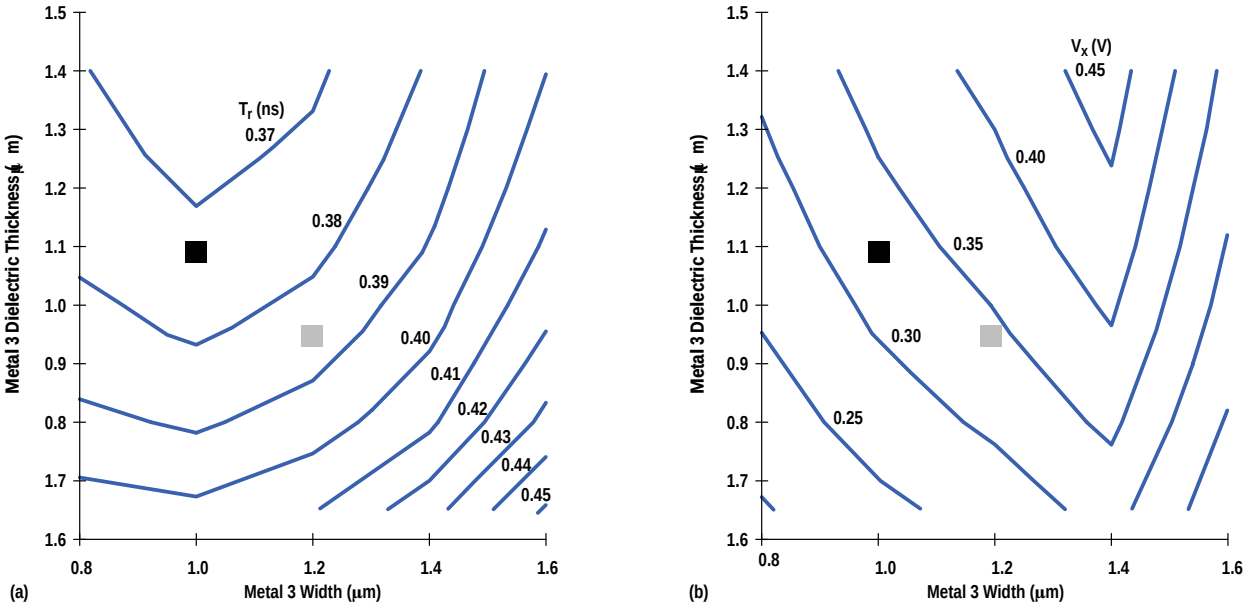


Fig. 6. *Contours of constant rise time (a) and cross talk (b) for the metal 3 dominated net2. The nominal design point is indicated by the gray square while the possible optimized point is shown in black. The pitch (width + space) was kept constant at 2.4 μm.*

optimization reveals that the rise time can only be improved by using a narrower metal 4 and a thicker dielectric as shown in Fig. 8. Cross talk is a much smaller concern here because of the wide spacing between metal lines.

Conclusion

The main results of this study are as follows:

- Global routing in metal 2 wires causes significant delays because of the high resistance associated with this layer. The delays can be reduced by increasing the metal thickness and by increasing the metal 2 width even if the metal 2 pitch is kept constant.
- Typical lengths of global interconnect in metal 3 are relatively short. Therefore, delays incurred because of routing in metal 3 are minimal. Hence, the optimization for metal 3 should be primarily based on performance within functional units.
- For metal 4, the delays are primarily affected by the capacitance of the lines rather than the resistance, since the thickness is substantially greater than for metal 2. Therefore, shorter delays require thicker dielectrics and narrower metal lines when constant pitch is assumed.

In our study the gains in performance of up to 200 ps can be obtained by wire optimization within reasonable process parameters. A simple example will illustrate the impact of an improvement of this magnitude on the cycle time of a microprocessor. Consider a hypothetical processor fabricated in a manufacturing process that yields a nominal cycle time of 4.0 ns (250 MHz), with a standard deviation, assuming a normal distribution around the mean of 0.3 ns. In this case, only 7.6% of the devices will have a speed of 280 MHz or greater. If the nominal design is changed to reduce the cycle time by just 200 ps, then the fraction of the devices yielding above 280 MHz will increase threefold to 22.2%. Thus, a 5.0% reduction in the nominal cycle time produces a 200% increase in the number of the fastest devices, which command a premium because of their high performance.

Acknowledgments

The PA 8000 data to which the optimization method was applied was provided by Clay McDonald and Mark Hammer, whose cooperation and help are gratefully acknowledged.

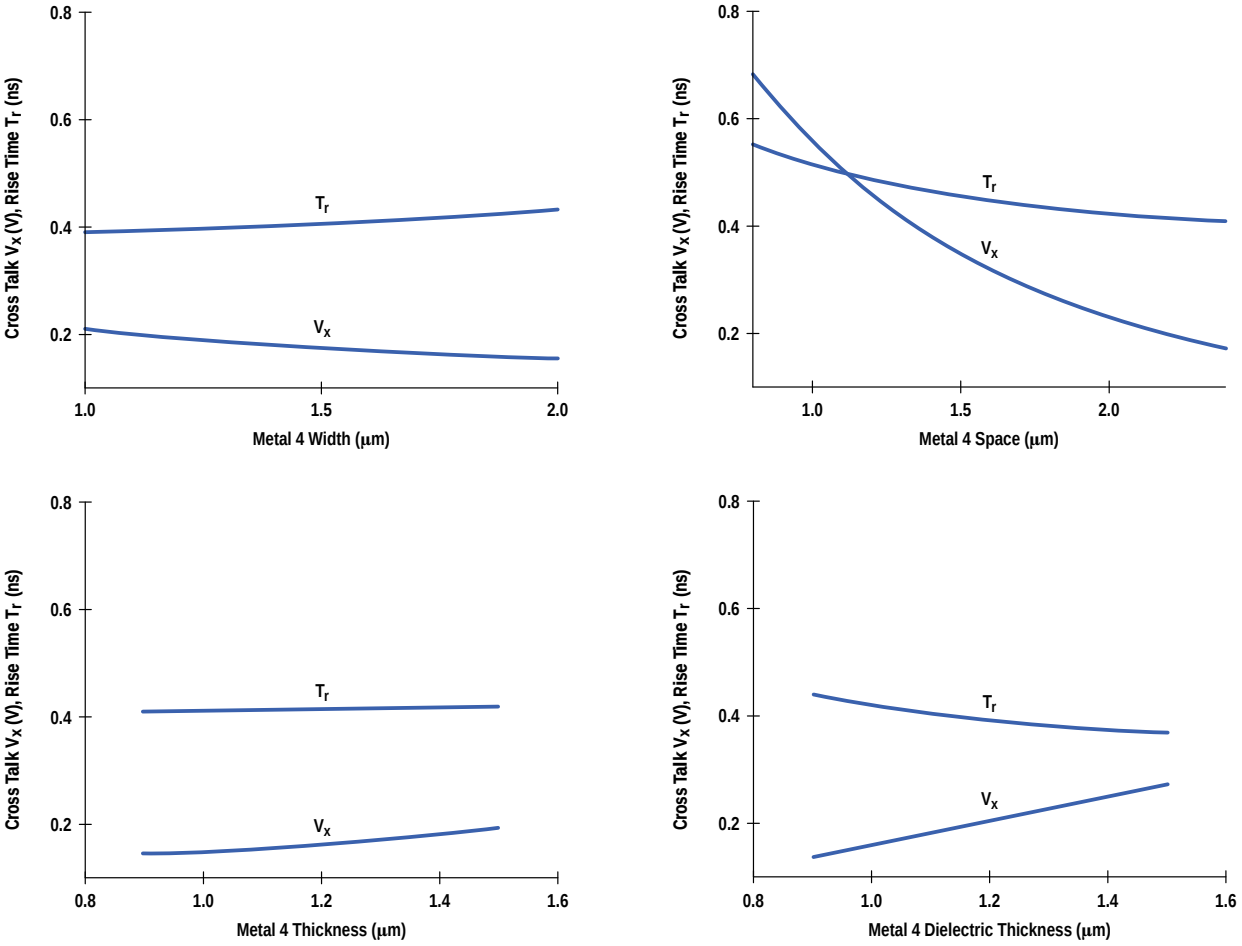


Fig. 7. *Sensitivity of net3 rise time T_r and cross talk V_x to metal 4 parameters.*

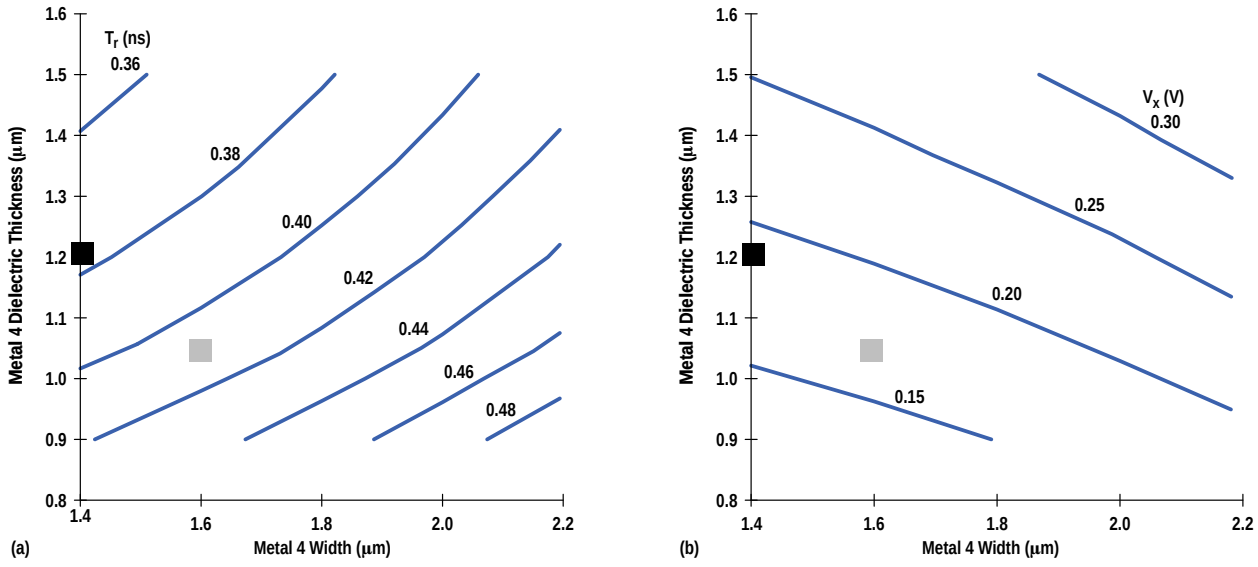


Fig. 8. *Contours of constant rise time (a) and cross talk (b) for the metal 4 dominated net3. The nominal design point is indicated by the gray square while the possible optimized point is shown in black. The pitch (width + space) was kept constant at 4.0 μm.*

References

1. J. Lotz, G. Lesartre, S. Naffziger, and D. Kopp, "A quad-issue out-of-order RISC CPU," *Technical Digest, International Solid-State Circuits Conference*, 1996, p. 210.
2. M. Tremblay, "UltraSparc-I: A 64-bit superscalar processor with multimedia support," *Technical Digest, Hot Chips Conference*, 1995, p. 207.
3. R.P. Colwell and R.L. Steck, "A 0.6-μm BiCMOS processor with dynamic execution," *Technical Digest, International Solid-State Circuits Conference*, 1995, p. 176.
4. *The National Technology Roadmap for Semiconductors (NTRS) 1994*, Semiconductor Industry Association, 1994.
5. S-Y. Oh and K-J. Chang, "2001 needs for multilevel interconnect technology," *IEEE Circuits and Devices Magazine*, January 1995, p. 16.
6. P. Raje, "A Framework for Insight into the Impact of Interconnect on 0.35-μm VLSI Performance," *Hewlett-Packard Journal*, Vol. 46, no. 1, February 1995, p. 97.
7. K. Rahmat, O.S. Nakagawa, S-Y. Oh, J. Moll, and W.T. Lynch, "A scaling scheme for interconnect in deep-submicron processes," *Technical Digest, International Electron Devices Meeting (IEDM)*, 1995, p. 245.

Designing, Simulating, and Testing an Analog Phase-Locked Loop in a Digital Environment

In designing a phase-locked loop for use on several HP ASICs, the digital portion of an existing phase-locked loop was transferred to a behavioral VHDL description and synthesized. A behavioral model was written for the analog section to allow the ASIC designers to run system simulations. A new leakage test was developed that has been very effective in screening out process defects in the filter of the original design.

by Thomas J. Thatcher, Michael M. Oshima, and Cindy Botelho

This paper describes the design and integration process for a phase-locked loop that is being used on several current HP ASICs (application-specific integrated circuits). The design is based on the phase-locked loop for a previous ASIC, but several improvements were made. First, the digital portion of the phase-locked loop was transferred to a behavioral VHDL* description and synthesized. Reusability was a big consideration in writing the code. The portable nature of the VHDL code enabled us to design several phase-locked loops within a very short time. Second, a behavioral model was written for the analog section to allow the ASIC designers to run system simulations. This model, when combined with the model for the digital section, allows the designer to simulate the phase-locked loop as it locks—it does not merely put out an ideal clock waveform. Finally, in a previous ASIC, a large resistor and capacitor in the loop filter were not adequately tested. For the new phase-locked loop, we developed a leakage test that has been very effective in screening out process defects in the filter.

An analog phase-locked loop presents several challenges to designers in an all-digital design environment. Some all-digital simulators, such as Verilog XL, cannot represent analog signals easily. System designers must either use a mixed-mode simulator to represent the analog portions of the phase-locked loop, or use a simplified model of the phase-locked loop. In ASIC production test, limitations of the production test equipment must be taken into account. For example, an analog measurement may take a long time to complete. Also, functional testers cannot measure frequency, so it is difficult to determine that the phase-locked loop is operating properly in production test.

Previous Phase-Locked Loop Design

The previous phase-locked loop design appeared on an ASIC, where its purpose was to accept an input clock (the video clock) and generate the system clock. The clock signal output from the phase-locked loop was modulated so that the output clock frequency alternated back and forth

between two frequencies slightly above and below the target system clock frequency. This frequency modulation of the system clock was required by the system in which the previous design was used.

The block diagram of the previous phase-locked loop is shown in Fig. 1. The loop consists of an input counter (divide-by-M), a feedback counter (divide-by-N), a phase-frequency detector, a charge pump and filter, a voltage-controlled oscillator (VCO), and other digital control logic. The input counter divides the input clock by either M_{high} or M_{low}. It produces an output pulse (FREF) that is low for one clock cycle and high for the remaining time. The feedback counter divides the VCO output by N_{high} or N_{low}. It produces an output pulse (FBAK) that is low for two clock cycles. The phase-frequency detector examines the relative phase of the rising edges of the FREF and FBAK signals and generates pulses on the UP and DOWN signal lines. The charge pump uses these pulses to adjust the control voltage for the VCO. The output signal is generated by dividing the VCO output clock by 4. The resulting phase-locked loop output frequency is given by:

$$f_{out} = \frac{1}{4}\left(\frac{N}{M}\right)f_{in}. \qquad (1)$$

* VHDL stands for Very High-Speed Integrated Circuit Hardware Description Language.

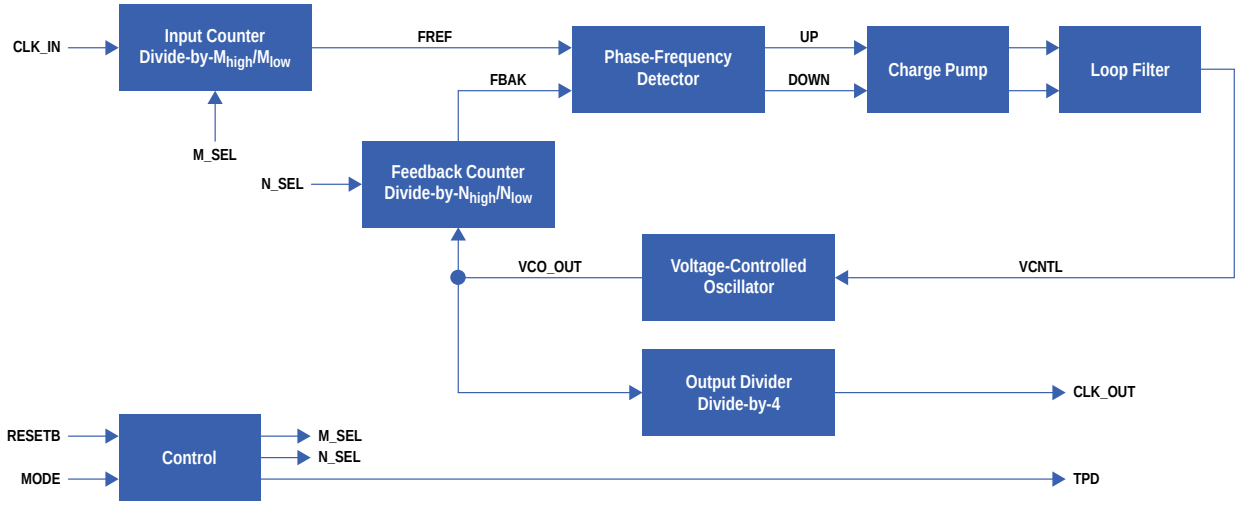


Fig. 1. *Block diagram of the original phase-locked loop design.*

The control block consists of logic that selects one of the operating frequencies. A state machine in this block controls the modulation between the upper and lower frequencies. In normal mode, the control logic sets the phase-locked loop to the upper frequency when it is reset. The loop remains at the upper frequency until the control block receives a signal that indicates that the loop is locked. After this, the loop alternates between the upper and lower frequencies. This is controlled by the N_SEL and M_SEL outputs of the modulator counter.

There were two main problems with the integration of the phase-locked loop. The first problem was that there was no model of the phase-locked loop in VHDL or Verilog that could be used for system simulation. Therefore, no simulations were run with the clock generated by the phase-locked loop. All simulations were run with an external clock, using a different chip mode. This caused the design team to miss a serious bug in the design. When the mode pins were set to enable the phase-locked loop, one block inside the chip was accidently set into scan mode. This problem was not caught until the first prototype parts were placed on a board.

The second problem encountered with this phase-locked loop design was a high production line failure rate. The phase-locked loop tests were not catching all the defective parts. Analysis of the returned parts showed that the failures were caused by defects in the resistor and capacitor in the loop filter, which caused excessive leakage, changing the filter characteristics. The production tester had no way to test for this, so a new test had to be created. Here the lack of a good simulation model for the phase-locked loop was a real handicap. The original tests for the phase-locked loop were debugged on the tester. When trying out new tests, we had no way of simulating them to verify their correctness. Thus, it took two or three iterations before the tests were correct.

New Design Approach

Other ASICs being designed by our design center required phase-locked loops with similar characteristics. These ASICs were designed in a different manufacturing process. Therefore, the phase-locked loop had to be redesigned so that it would be leveragable to many different chips. As a result, it was decided to break up the design into two parts (Fig. 2). The digital logic would be designed using a VHDL synthesis strategy. Behavioral VHDL code would be written and then synthesized into our standard cell library. The code would be written so that it was easy to customize for new projects. Once the synthesis was complete, the netlist for the digital block would be routed and then integrated into the top level of the phase-locked loop. The analog portions of the phase-locked loop would be designed using a full-custom methodology. We would leverage the analog blocks from the previous phase-locked loop, but they might have to be modified and resimulated. Every new phase-locked loop design would undergo a final stability analysis.

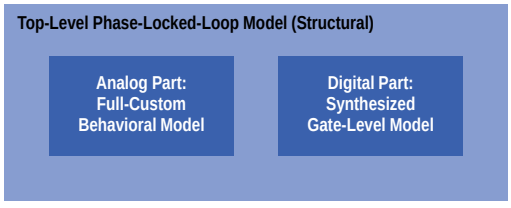


Fig. 2. *Structure of the phase-locked loop model.*

Model Structure

In the current design flow, the customer designs the chip by writing behavioral code in VHDL, synthesizing the design, and running final simulations in Verilog. To have a phase-locked loop model available for system simulations, we needed both a VHDL model and a Verilog model. The structure of the model is the same for both VHDL and Verilog. In the top level of the model are a digital part and an analog part. The model used for the digital part is simply the gate-level netlist for the digital block. Delays were calculated from the routing capacitance and were back-annotated using Standard Delay Format (SDF). The model for the analog part was written separately in VHDL and Verilog. These models were written at the behavioral level.

The logic for the digital section of the new phase-locked loop was highly leveraged from the old design. In most cases, the original phase-locked loop schematics were used as a guide, and the VHDL models for all the blocks inside the digital section were written so that the synthesis tool would produce equivalent logic. Constants were used in the code so that the design could be easily adapted. This made it much easier to change the design when specifications changed.

The analog section of the phase-locked loop was modeled in two parts. The first part modeled was the phase-frequency detector. The phase-frequency detector puts out a pulse on either its UP or DOWN output whose width equals the distance between the rising edges of the divider outputs FREF and FBAK. The state table for the phase-frequency detector is shown in Fig. 3. This was a fairly straightforward model to write, since the inputs and outputs were digital signals.

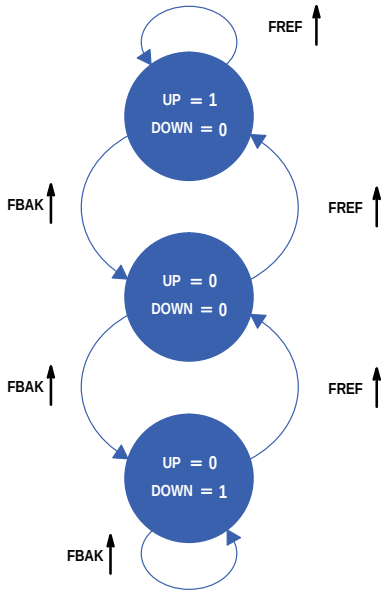


Fig. 3. *State table for the phase-frequency detector.*

Model for the Charge Pump and VCO

Finally, a model needed to be written for the charge pump, filter, and VCO. A simplified model of the charge pump and filter is shown in Fig. 4. A positive pulse on the UP signal will cause the voltage on VCNTL to rise, and a positive pulse on the DOWN signal will cause VCNTL to fall. The filter is added to ensure the stability of the loop.

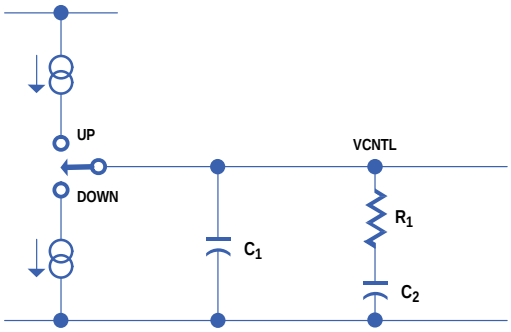


Fig. 4. *Simplified charge pump and filter model.*

A model was needed that would not slow down the simulation too much, and that could be written easily in both Verilog and VHDL. One problem with Verilog is that there is no easy way to define analog signals (although they could be represented with arrays or integers).

This problem was solved by using a variable of type time to represent the period of the VCO output clock. As pulses on the UP and DOWN signals arrive at the charge pump model, the length of the period is adjusted accordingly. The model calculates the width of the pulse on the UP or DOWN signal and divides by the maximum period count to determine the amount of correction to the period.

The model operates much differently than the analog circuit. The operation of the analog circuit is shown in Fig. 5. When the UP signal is high, the charge pump charges C_1 and the voltage VCNTL rises. When UP is low, charge shifts from C_1 to C_2 through the resistor and VCNTL falls.

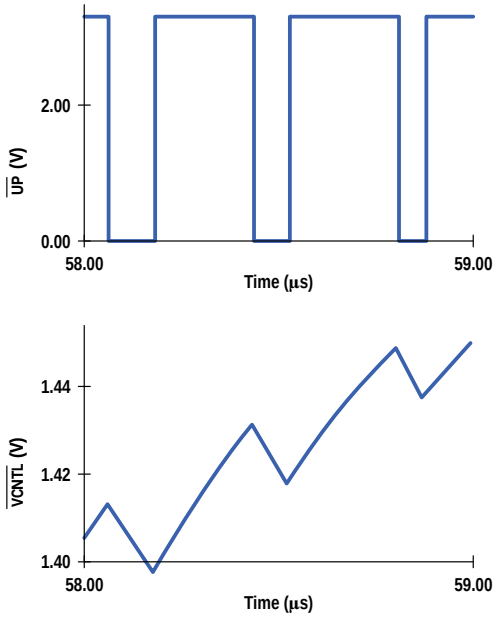


Fig. 5. *Operation of the charge pump.*

In the simulation model of the charge pump and VCO, the model calculates the width of the pulse on the falling edge of the UP signal. It then applies the correction to the period of the VCO. Some time later it will reduce the correction by half. This roughly approximates the rise and fall of the VCNTL voltage in the real circuit. The operation of the model is shown in Fig. 6.

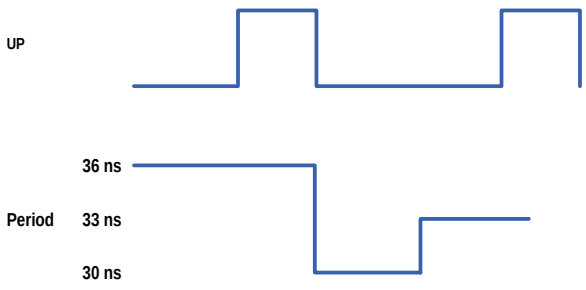


Fig. 6. *Operation of the charge pump model.*

In practice, this model works very well. The model locks to the proper frequency, and if modulation is used, the phase-locked loop will modulate between the proper two frequencies. However, the transition from one frequency to another will be much different from the actual phase-locked loop. Typically, the model locks much faster than the actual circuit.

Process for Developing New Phase-Locked Loop Designs

The intent from the beginning was to create a design that was easy to modify and adapt. Each ASIC has its own requirements for the input and output frequencies. Here is the procedure for developing a new customized version of the phase-locked loop.

1. Copy the VHDL code for the digital block and make changes to customize the code. Usually this involves changing constants in the code for the M counts, the N counts, and the modulation counts.

2. Simulate the entire phase-locked loop in VHDL to make sure that the operation of the digital block is correct.
3. Synthesize the digital logic using Synopsys.
4. Have the resulting netlist placed and routed in a block. We use Cadence's Cell3 to do placement and routing.
5. Calculate circuit delays from the routing capacitance. Calculate RC delays.
6. Do a timing analysis for the digital logic using the calculated delays.
7. Do a logic simulation using the real delays.
8. Make modifications to the phase-locked loop production tests.
9. If necessary, adjust the artwork for the analog blocks according to the specifications for the new phase-locked loop (for example, to adjust the frequency range of the VCO).
10. Redo the stability analysis for the phase-locked loop.
11. Place the routed digital block into the artwork for the top-level phase-locked loop.

Modifications to the digital section of the phase-locked loop can be done in about a week. This was very helpful for some ASIC designs because the system requirements for the phase-locked loop were changed several times. Changes to the analog section usually take much longer.

Problems with the Model

The availability of the complete model for the phase-locked loop was a great advantage. It enabled us to debug production tests for the loop before the chip was released. In addition, it allowed the customer to do system-level simulations that included the full modulating capability of the phase-locked loop. Our customers have used this capability to catch problems in the clocking and to debug corner-case problems. However, there were some minor problems with the model.

The first problem is that the model of the charge pump and VCO is very sensitive to the simulation resolution. The model was developed and runs fairly well with a resolution of 10.0 ps. It does not work very well when simulations are run with 1-ns resolution. This is because the model needs the extra resolution when adjusting the VCO clock period.

The second problem is that it is difficult to initialize the model. Because there is a feedback loop in the model, unknown values present at the beginning of the simulation propagate around the loop. To get the phase-locked loop to initialize, it is necessary to put the loop into its digital test mode or to force the counter reset signal in the phase-locked loop to 0 and then release it. Most people prefer the latter option because it requires less effort and doesn't require changing the system tests.

Once the model is initialized and running, it is still susceptible to unknown values being introduced. Since the phase-locked loop contains two asynchronous clock domains (the input clock and the VCO output clock), occasionally there will be a setup or hold violation on a synchronizer flip-flop. As a result, the output of that flip-flop becomes unknown, and this unknown value propagates around the loop. This problem could be averted if there were a synchronizer flip-flop model that did not go to an unknown state on a setup or hold violation.

Testing and Test Development

Our experience with phase-locked loop designs has showed us the need for more complete testing of the loop. Originally, three main tests were created for the production test of the loop:

- An analog test ran the phase-locked loop in all functional modes, allowed the loop to lock, and then made match assertions for 0 and 1. This test only tested that the phase-locked loop was producing a clock waveform. It did not test the frequency or duty cycle of the loop output.
- A digital test tested the functionality of the digital block through four ad-hoc test points inserted in strategic locations.
- A VCO linearity test applied three different clock frequencies to the clock input of the phase-locked loop. The loop was allowed to lock, and then the control voltage to the VCO was measured with the precision measuring unit of the tester. The test program then checked that the control voltage was higher for higher frequencies. This test was dropped from the production test because of concerns that the large capacitance of the precision measuring unit was altering the voltage measurement when it was connected.

These three tests did not fully cover the phase-locked loop. The digital logic was covered pretty well, and the basic functionality of the analog part was tested, but the production test did not catch any parameter variations that would cause the phase-locked loop to fail to lock. To address this concern, four additional tests were developed for later phase-locked loop designs, and three of these tests have been incorporated into the final production test.

Lock Test. A test was needed that would give some indication that the phase-locked loop was able to lock. To accomplish this, a new test was created. This test checks an internal signal in the phase-locked loop that indicates that the loop is close to being locked. This signal is one of the four test points that appear in the phase-locked loop design. To run this test, the loop is placed in one of its fixed frequency modes and allowed to lock. The internal lock signal is then checked to be sure that the loop has locked.

Clock 1× Test. Other ways of testing that the phase-locked loop was able to lock were also investigated. One test involved setting the loop into a frequency mode in which its output frequency was equal to the input frequency (or some even multiple). Then the loop was allowed to lock. After the loop had locked, the output divider was reset to make the phase predictable, and the clock output was strobed during the clock low time for 100 cycles. Then the output was strobed during the clock high time for 100 cycles. The idea was that if the phase-locked loop was not at the correct frequency, the phase would drift and there would be strobe failures. This test did not work very well in practice. When the loop was put into a chip and simulated, the difference between fast and slow delays made it impossible to find a good time to strobe the output clock. As a result, this test had to be dropped from the production test.

Leakage Test. In one ASIC, the phase-locked loop circuit was suspected of causing an excessive production line failure rate. After some investigation, the filter of the phase-locked loop was identified as the cause of these failures. Manufacturing defects in the large resistors and capacitors caused excessive leakage, changing the characteristics of the filter and preventing proper operation of the phase-locked loop. To screen out these defects, a new test was developed to check the filter for excessive leakage. In this test, the filter's analog signal is multiplexed out to a dedicated test pin and the charge pump circuit is turned off by means of a control bit. A voltage of 3.3 volts is applied to the pin and a current measurement is done with the tester's precision measuring unit. Parts with excessive leakage current are discarded. The implementation of this and other phase-locked loop tests has resulted in a 98% reduction in the line failure rate due to the phase-locked loop.

Charge Pump Tests. Another circuit not adequately tested in the original phase-locked loop was the charge pump. To remedy this a new test was developed to ensure that the charge pump was able to source and sink current within its design specifications. To allow for this test, new circuitry was added to disable either the M or N counter in the digital test mode. In the test, the chip is put into the digital test mode with the N counter disabled. A clock is applied to the M counter and the M counter is allowed to count until three pulses have occurred on its output. At this point, the UP signal will be high and the DOWN signal will be low. The clock is stopped at this point, a voltage of 1.3V is applied to an analog test pin at the output of the loop filter, and the current sourced by the charge pump is measured and checked against the expected current. The same test is repeated, this time disabling the M counter and allowing the N counter to count until three pulses have occurred on its output. At this point, the DOWN signal is high and the UP signal is low. A voltage of 1.3V is applied to the analog test point and the current drawn by the charge pump is measured.

Summary

We have greatly streamlined the procedure for designing, integrating, and testing a new phase-locked loop. In many cases, changes to the phase-locked loop can be made by resynthesizing and rerouting the digital block and then performing a stability analysis. The customer has access to a detailed model of the phase-locked loop in either VHDL or Verilog, and can use this to test the proper connection of the phase-locked loop and the surrounding system. Production tests can be debugged with simulation before they are put onto a tester. Finally, standard phase-locked loop tests have been written that run on our digital IC tester. These tests increase the test coverage for the phase-locked loop, increasing the quality of products shipped with these phase-locked loops in them.

Acknowledgments

The authors would to thank Derek Knee and Terry Huang, who designed the analog section of the original phase-locked loop.

Analog Behavioral Modeling and Mixed-Mode Simulation with SABER and Verilog

A description is given of specific analog behavioral modeling and mixed-mode simulation techniques using SABER and Verilog. Full-channel simulations have been carried out on a class I partial response maximum likelihood (PRML) read/write channel chip. Complex analog circuits such as an adaptive feed-forward equalizer, an automatic gain control block, and a phase-locked loop are modeled in detail with the SABER MAST mixed-signal behavioral modeling language. A simulation speedup of two orders of magnitude has been achieved compared to SPICE.

by Ben B. Sheng, Hugh S.C. Wallace, and James S. Ignowski

For more than two decades, the analog IC design community has been relying on variations of the original Berkeley SPICE, introduced in the 1970s, as the simulation tool for verifying and fine-tuning analog designs. Over the years, many enhancements have been put into these different flavors of SPICE, while increasingly more powerful computers have been used for running these circuit simulations. However, SPICE remains a low-level circuit simulator. It produces accurate results, but is inherently slow. Today's analog and mixed-mode designs are becoming increasingly complex. Functional simulations for larger mixed-signal designs are impractical with SPICE. Meanwhile, as the pressure increases for low-cost, high-integration ASICs ("systems on a chip"), many analog functions are being integrated into largely digital chips. The need for new simulation methodologies is becoming more urgent.

In recent years, benefits from using analog and mixed-mode behavioral modeling languages have received increased recognition. The basic approach is to use a SPICE-like continuous-time simulator, which provides good accuracy in simulations, together with a fast digital simulator to give orders of magnitude faster digital circuit simulations. The modeling language is flexible so that designers can model analog subsystems in different levels of abstraction. The modeling language gives designers control over the trade-off between simulation speed and accuracy.

This paper presents some of the bottom-up modeling techniques and simulation approaches that have been adopted during the process of modeling and simulating the read-write channel chip for an HP DDS-3 DAT drive.

Analog Behavioral Modeling

The idea of behavioral modeling is not new to analog designers. Macro models have been widely used by SPICE users. The newer-generation mixed-mode circuit simulators, such as SABER by Analogy, Inc. and SPECTRE by Cadence Design Systems, Inc., have greatly enhanced designers' ability to model analog and mixed-mode circuits and systems by providing a flexible behavioral modeling language. With this modeling language, a designer can behaviorally describe an analog or mixed-mode device or subsystem at whatever level of abstraction is appropriate for a given simulation accuracy-versus-speed trade-off. One can use this modeling language to write BSIM models for MOS transistors and use these BSIM models to achieve simulation results that are as accurate as those from SPICE simulations. The same modeling language can be used to describe an analog-to-digital converter (ADC) behaviorally, without having to refer to any of its internal circuit elements.

Several modeling approaches are discussed in this section. Based on the scopes of these different approaches and their simulation speed-versus-accuracy trade-offs, they can be categorized as either high-level, medium-level, or low-level modeling. In the following subsections, specific examples are given for high-level and medium-level modeling. Although low-level modeling is a very important part of analog modeling and simulation, the techniques used to do low-level modeling are very similar to those used in higher-level modeling. The only difference is that these techniques are used to model much smaller devices, such as MOSFETs, diodes, and bipolar junction transistors. For brevity, discussion of low-level modeling is omitted.

High-Level Modeling

High-level modeling refers to behavioral models that describe large analog and mixed-mode subsystems in a high level of abstraction. This approach provides the fastest simulation speed but the least detail in the circuits that are modeled.

An ADC can be modeled with a clock input signal that triggers each conversion, an analog input signal, an analog reference signal, and digital outputs. In addition to this basic structure, some realistic behavior can be included in the model. For example, the model can include characteristics such as differential nonlinearity, integral nonlinearity, and metastability characteristics. A behavioral 3-bit ADC model written in the SABER MAST modeling language is shown in Fig. 1.

As can be seen in this example, high-level modeling can be used in describing analog and mixed-mode subsystems, with some detail included. This particular approach is suitable for functional simulations of large systems. A large number of functional simulations can be carried out quickly, but circuit details are often omitted. This type of high-level modeling can speed up simulations by at least three orders of magnitude compared to SPICE, at the cost of not being able to simulate the fine details in the circuits.

For complex mixed-signal designs, chip-level connectivity verification is often a problem, since neither traditional analog simulators such as SPICE nor pure digital simulators such as Verilog can, for example, check voltage reference levels or common-mode levels of differential signals. One of the most important benefits of using high-level models is the ability to verify top-level circuit connectivity when the final chip is composed. One specific example is given in the ADC model of Fig. 1. Line 21 checks that the reference voltage to the ADC is not connected incorrectly (negative reference).

Another key point is the ability to do *analog assertion*. Traditional graphical analog postprocessors work well if there are a manageable number of signals and time windows to look at—in other words, when dealing with simulation results of a relatively small circuit. For system-level simulations, in which multiple complex analog blocks interact with each other and hundreds of signals are changing constantly, it becomes very difficult to track all the important signals to make sure that all the circuits are operating within their specified parameters. The analog modeling language allows designers to put specific assertions in the models to monitor the analog signals and give warning messages or even abort simulations when the model is operating outside of a set of prespecified parameters. An example of such an assertion can be seen in line 22 of Fig. 1, where the input signal is compared to a prespecified maximum level.

The ADC model not only evaluates signals in the analog domain, but also schedules digital events. As mentioned earlier, the newer-generation simulators not only provide SPICE-like analog simulation engines, but also have built-in digital simulators, which give them much improved performance in simulating mixed-mode systems. Built-in digital simulators are orders of magnitude faster in simulating digital circuits than a SPICE-like simulator. We will discuss mixed-mode simulations in more detail later.

Medium-Level Modeling

Medium-level modeling refers to behavioral modeling of smaller building blocks, such as an operational amplifier (op amp), a multiplier, an integrator, or a comparator. These circuits typically contain a few dozen transistors. Models for these circuits can have more physical characteristics, which track SPICE simulation results.

For modeling the read/write channel chip, we used medium-level models most extensively. Three different techniques were used to develop the models, based on the different circuit structures.

The first approach is to create generic building blocks with flexible parameters that can be used to customize these generic models when they are used in different applications. A good example for this approach is the model for an op amp. Characteristics of a generic op amp include dc gain, pole location (if a second pole location is important, one can simply connect two one-pole models in series), zero location, input impedance, input capacitive load, output impedance, output capacitance, output common-mode level, slew rate, output swing, and nonlinear distortion. More comprehensive characteristics can be added to this list when needed. Fig. 2 shows an example of such a generic op amp model. Some of the parameters listed above are omitted for brevity.

This type of generic circuit model can be made flexible enough to satisfy a variety of modeling and simulation needs. Additional characterization can be included, when appropriate, at relatively low simulation speed cost.

In some cases, when the system performance is highly sensitive to the modeling accuracy of a small block, a curve-fitting approach can be used. This approach is suitable for small circuits with few input ports. Outputs of a circuit can be empirically fit to carefully chosen mathematical functions. These models are fully customized for the given circuits. They can be made very accurate in representing a few key characteristics of some small blocks, but they are not reusable for modeling other circuits, and can become very complex when more input/output signals are added to the equations. Designers who take this approach must be careful to produce a model that covers the entire input signal range to prevent the models from producing erroneous simulation results.

The third approach taken in modeling circuits in the read/write channel chip is to use the first-order MOSFET equations to approximate circuit behaviors. A good example is a MOSFET switch. The on-resistance of a MOSFET switch can be approximated with the first-order equations accurately enough for most applications. The switches are effectively modeled as nonlinear resistors. If a more accurate model is needed, the curve-fitting method can be used.

```
1 template adc clk in ref b2 b1 b0 = tau, td, dnl, inl, vmax
  {
    # Capacitive input load of 1 pF
    c.input_load in 0 = 1p
5   # Compute a random integral nonlinearity
    number random_inl = (2*rand()-1)*inl
    when(event_on(clk, clk_last)) {
      # Looking for rising edges in clk.
      if (clk == l4_1 & clk_last == l4_0) {
10      schedule_event(time, b2, l4_x)
        schedule_event(time, b1, l4_x)
        schedule_event(time, b0, l4_x)
        # Tell the analog simulator to step on the clock edge to get accurate
        # analog signal value
15      schedule_next_time(time)
        # Sample input voltage (with some adjustment for correct zeroing)
        # and reference voltage
        vref = v(ref)
        vin = v(in) + vref/8
20      # Error checking
        if (vref < 0) error (”The voltage reference to the ADC is negative”)
        if(abs(vin) > vmax) error (”The ADC input signal is out of range”)
        # Determine the sign bit
        if (vin < 0) d2 = 0
25      else d2 =1
        # Add random differential nonlinearity
        vin = abs(vin) + (2*rand()-1)*dnl
        # Clipping the output
        if (vin > vref) vin = vref
30      # Add random integral nonlinearity
        vin = vin + random_inl*(vref - abs(vin))*abs(vin)*4/vref/vref
        if (vin < vref/2) d1 = 0
        else {
          d1 = 1
35        vin = vin - vref/2
        }
        if (vin < vref/4) d0 = 0
        else {
          d0 = 1
40        vin = vin - vref/4
        }
        # Compute resolution time. If td+t_resolve > 1-clock-cycle, then the ADC is
        # in a metastable state (a conversion error occurs)
        t_resolve = tau*ln(2/(abs(vin)+1u))
45      if (d2 == 1) schedule_event(time+td+t_resolve, b2, l4_1)
        else schedule_event(time+td+t_resolve, b2, l4_0)
        if (d1 == 1) schedule_event(time+td+t_resolve, b1, l4_1)
        else schedule_event(time+td+t_resolve, b1, l4_0)
        if (d0 == 1) schedule_event(time+td+t_resolve, b0, l4_1)
50      else schedule_event(time+td+t_resolve, b0, l4_0)
      }
    }
  }
```

Fig. 1. *A behavioral 3-bit ADC model written in the SABER MAST modeling language.*

```
template opamp inp inm outp outm vcm vdd gnd pd = ci, gm, gm3, vswing, ro, co, rz, imax
  {
    c.cip inp 0 = ci
    c.cim inm 0 = ci
    values {
      # Define the block's inputs
      vint = v(inp,inm)
      vinp = v(inp)
      vinm = v(inm)
      if (v(inm) > v(inp)) {
        vinp = v(inm)
        vinm = v(inp)
      }
      # Gain and 3rd order nonlinear distortion
      idiff = gm*vint + gm3*vint**3
      # Output swing, the output clips at "vswing"
      iclip_up = (vswing - v(outp,outm))/ro/1k
      iclip_dn = (vswing - v(outm,outp))/ro/1k
      if (v(outp,outm) > vswing) {
        iclip_up = iclip_up*10meg
      }
      else if (v(outp,outm) < -vswing) {
        iclip_dn = iclip_dn*10meg
      }
      idiff = idiff + iclip_up - iclip_dn;
      # Slew rate
      if (idiff > imax) idiff = imax
      if (pd ~= l4_0) idiff = 0
    }
    equations {
      # A pole and a zero
      icapp = d_by_dt((v(outp) - icapp*rz)*co)
      icapm = d_by_dt((v(outm) - icapm*rz)*co)
      i(outp) -= ioutp
      ioutp: ioutp = idiff - v(outp,vcm)/ro - icapp
      i(outm) -= ioutm
      ioutm: ioutm = -idiff - v(outm,vcm)/ro - icapm
    }
  }
```

Fig. 2. *An example of a generic op amp model.*

Mixed-Mode Simulations

SPICE is an integration-based simulator. Although it gives good simulation accuracy, it is inherently slow, since it has to set up and solve a nonlinear system matrix for the entire circuit at every integration time point. The integration time step is determined by the largest signal change in the circuit, regardless of whether it is a digital signal or an analog signal. It is impractical to do system-level simulation with such an algorithm when the system has more than a few thousand transistors.

To run system-level simulations on a complex mixed-signal design and get reasonable accuracy, it is necessary to use an event-driven digital simulator to simulate the digital portion of the design, together with an integration-based analog simulator to simulate the analog circuits. The newer-generation mixed-mode simulators provide designers with both digital and analog simulation engines in one simulator. Continuous-time analog signals are computed by the analog simulator, while the digital simulator evaluates the event queue. As illustrated by the ADC example, designers can specify the interactions between the digital and analog domains. These interactions are resolved with the relaxation method.[1] The computational overhead for resolving these interactions is minimal, since the interactions are by nature discrete events. In addition, the built-in digital simulators make it easy for the mixed-mode simulators to cosimulate with widely used external digital simulators, such as Cadence's Verilog-XL. Since the mixed-mode simulators already create digital event queues, they can share their event queues with an external digital simulator's event queue, and vice versa. The ability to cosimulate with well-established digital simulators makes it possible to use all of the available digital libraries, which are often built based on careful characterizations and can provide accurate simulation results.

Fig. 3 shows the functional block diagram of the read/write channel chip. The shaded blocks indicate analog functions that were modeled with a behavioral analog modeling language. The rest of the chip (approximately 40%) is digital, including all of the system timing and control functions, a maximum-likelihood sequence detector (MLSD or Viterbi detector), and much of the phase-locked loop. These digital circuits were designed with Verilog HDL and synthesized with Synopsys using HP CMOS34 (1.0-µm CMOS) standard cells. For the final system-level simulations, the digital circuits were simulated with Verilog and behavioral models of the analog circuits were simulated with SABER. Special interface models were built to establish connections between the two simulators.

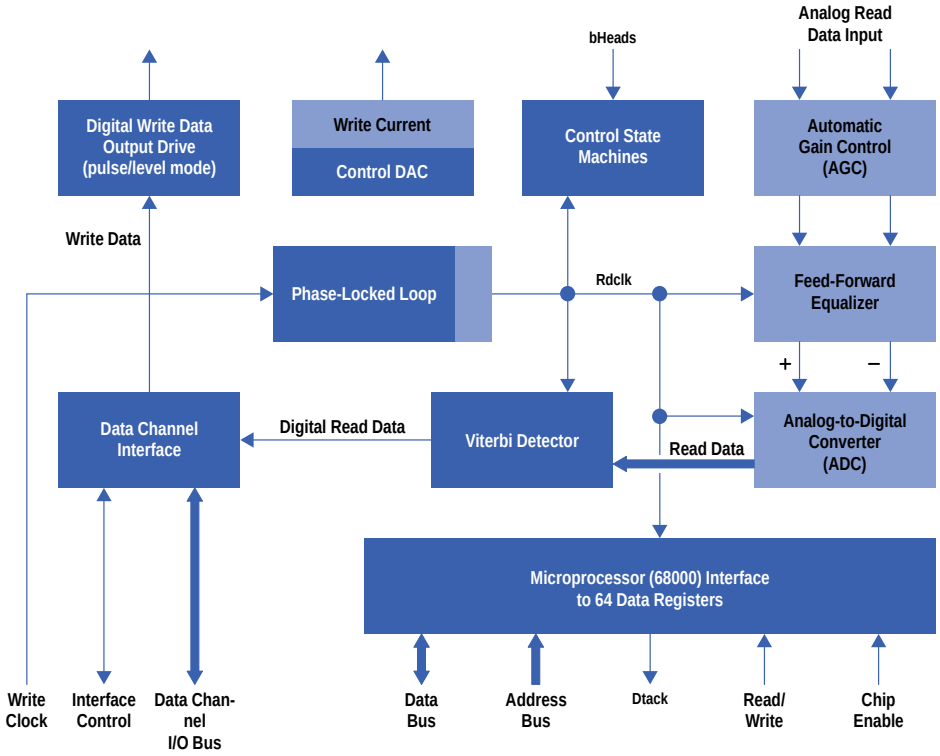


Fig. 3. *Functional block diagram of the read/write channel chip. The shaded blocks indicate analog functions that were modeled with a behavioral analog modeling language.*

System-Level Simulation Example: Read/Write IC

The read/write channel chip is a mixed-signal IC with high analog circuit content. Approximately 60% of the core area is analog in nature. Because of the complexity and size of the analog circuits, many typical design considerations became more challenging. Simulating the interfaces and interactions between analog blocks as well as between the analog blocks and their digital control blocks was critical for a successful design. Translation of system-level specifications into meaningful circuit block design specifications was difficult, and probably most important of all, managing the system-wide budget of circuit nonideal behaviors was a formidable task. Ultimately our IC design team, working closely with the system design team from HP's Computer Peripherals Bristol Division, developed a simulation strategy of linking and correlating the simulation results of several tools to provide a closed loop of circuit and system verification.

In the initial step of the design process, the customer specified circuit performance by building a system model using C and Mathematica that was partitioned in a fashion consistent with the circuit partition. Much of the read/write channel chip's analog signal path was based on an analog PRML (partial response maximum likelihood) read channel chip that was developed for disk drive applications, so there was some previous work that was leveraged in architecting the read/write channel chip. As functional descriptions of circuit operation were put into the system model, specific limits and design goals could be identified. This information drove the design of the analog circuit blocks, which were then simulated in SPICE. The detailed results of the individual block simulations were incorporated back into the system model and real-world, nonideal circuit behaviors, such as offsets, nonlinearity, and PVT (process, voltage, and temperature) dependencies, were introduced into the system model. This increased level of detail in the system model led to very clear and well-defined specifications for the analog circuit blocks. This linking of system performance to circuit block performance allowed us to determine overall channel performance and define criteria for deciding when the design had acceptable margin and was ready for tape release.

In parallel with the system simulations, high-level simulations using SABER and Verilog were being run with analog block models. The high-level simulations performed functional and timing checks of the digital control blocks and their interfaces to the behavioral representations of the analog blocks. In this way, the operation of analog circuitry under the control of

complex state machines was verified. The results of high-level simulations of the analog portion of the system were compared to the results of the Mathematica system model for verification. Analysis and debugging of the analog behavioral simulation results were valuable for modeling and for confirming observations from the test and characterization of first silicon.

Critical Analysis

The read/write channel chip was our first major chip design for which high-level system design and verification were extremely critical to the chip's performance. Many previous mixed-signal designs depended on just the individual functional blocks' meeting the specifications to ensure that the final chip would perform to the system specifications. For the read/write channel chip, the system performance and trade-offs could only be evaluated after much of the actual circuit design was completed. The system performance of the read/write channel chip was extremely sensitive to a number of nonidealities in the signal path, so multiple iterations were carried out between careful circuit characterizations and system performance evaluations.

After the project was finished, and with an increasing need to define formally a methodology for new mixed-mode signal processing chips, the strengths and weaknesses of the read/write channel chip methodology were evaluated.

The high-level SABER models verified that the digital control circuitry functioned correctly with the analog blocks. This contribution cannot be overstated. The number of signals together with the complex interaction between the digital and analog blocks cannot be checked adequately in any other manner. The behavioral modeling of the analog blocks also discovered problems within the analog circuitry. While many of the main related analog blocks were simulated in SPICE together, there were others that were not, because of size and speed limitations in SPICE. The netlist extraction for the top-level SABER simulations was automatically generated from the chip transistor schematic, so the high-level simulations gave good confidence that the chip was correctly connected.

The main weakness in the process was that one engineer was responsible for developing all the analog behavioral models. This was because of manpower constraints in this project and the new introduction of the SABER tools. The investment in the learning curve for the new tools could not be absorbed by this project. The process that was followed to develop more than 100 behavioral models was examination of the schematics and SPICE plots that were available, and communication with the block designers. This worked fairly well in most cases, but there were some instances of incorrect behavioral modeling. The main problem was that human interpretation was needed to create the behavioral models. Characterization tests to compare the circuits with the models could have helped, but with so many blocks, some being continuously modified, a full set of characterization tests was not practical with the given amount of time and resources.

Enhanced Methodology

An enhanced methodology based on the one described above has been developed. This new methodology is intended to eliminate the weaknesses described in the read/write channel chip mixed-mode methodology by leveraging as much as possible from the behavioral synthesis methodology used in digital VLSI designs. The key is to remove as much human interpretation as possible and use the test vector set as a common link from customer definition to final chip simulation. Since not all the specifications in a mixed-mode design can be encoded with traditional digital test vectors, many of the analog specifications can only be evaluated by specific analog tests. The key to these tests being successful is to develop them from the customer's written specifications or the circuit design engineers' knowledge when the behavioral model is developed. The intention is that the block designer will develop the behavioral model. By taking this approach and developing tests that will run on both the real schematic and its behavioral model, a set of characterization tests can be exercised on the behavioral model. By starting a block design with a behavioral model based on written specifications and a top-level customer model, a more specification-driven evaluation of each circuit block can be used to determine whether the circuit will perform its desired function. This makes it possible to delay the exact circuit implementation until the full requirements of the block are understood. It could, for example, make the difference in deciding whether a block should be designed in the analog or the digital domain.

The approach of having the block designers develop the behavioral models still requires human interpretation and complete understanding of the circuit. Although this approach is an improvement of the methodology, the human interpretation is still a weakness. A number of simulators now gaining wide acceptance in VLSI digital design allow the whole chip to be converted into a FET-level simulation. The simulation speed is much faster (10×) than a SPICE simulation, with apparently improved convergence. The improvement in speed is obtained by running many parts of the circuit linearly. This approach compromises accuracy somewhat. The key to this new way to simulate mixed-mode systems is to define which blocks need to be simulated in a SPICE-like mode to ensure the required accuracy. This whole-chip simulation can only be performed when a complete FET schematic is available, so it is seen as a final verification check using the main mixed-mode simulation vector set.

Acknowledgments

The authors would like to thank Derek Knee, Kalwant Singh, and Mike Gilsdorf for reviewing this paper and giving valuable feedback.

Reference

1. *The Cutting Edge*, Analogy Inc., March 1996.

Physical Design of 0.35-µm Gate Arrays for Symmetric Multiprocessing Servers

To meet gate density and system performance requirements for the HP Exemplar S-class and X-class technical servers, a physical design methodology was developed for 1.1-million-raw-basic-cell, 0.35-µm CMOS gate arrays. Commercial and ASIC vendor-supplied tools were augmented with internally developed tools to put together a highly optimized physical chip design process.

by Lionel C. Bening, Tony M. Brewer, Harry D. Foster, Jeffrey S. Quigley, Robert A. Sussman, Paul F. Vogel, and Aaron W. Wells

This article provides a broad overview of the gate design and layout of 0.35-µm gate arrays for the new S-class and X-class members of the Hewlett-Packard Exemplar technical server family. The design process was built around third-party tools, but several internally developed tools were needed because commercial offerings were insufficient or unavailable. Generally, these tools automated the generation of the design or dramatically improved the logistics of the flow. Without these internally developed tools, meeting density and speed objectives (the optimization of the design) would not have been possible in acceptable calendar time with the design staff available.

In-house place-and-route tools played a key role in this design flow. As in previous projects, a close working relationship was developed with the gate array vendor. Among other benefits, this allowed the design staff to use the GARDS placement and routing software from SVR (formerly Silvar Lisco). Most of the internally developed tools written for the project operate directly on place-and-route information. Examples include a floor planner that understands design-specific routing obstructions, custom clock tree generation, and placement-based resynthesis of the gates.

Server Architecture

The target servers are symmetric multiprocessing systems built around the HP PA 8000 processor. S-class machines (also called nodes) connect 16 processors together to form a single system. X-class machines connect multiple nodes together to form a single system. X-class hardware can create a system containing up to 120 nodes (1920 processors). The PA 8000s will initially run at 180 MHz, with the rest of the system running at 120 MHz. Except for the PA 8000 and associated SRAMs and DRAMs, the bulk of the system logic is implemented in Fujitsu CG61 0.35-µm gate arrays, as shown in Table I. One additional gate array is implemented in the much less expensive CG51 0.5-µm process. This chip shared all the same tools and design flow as the 0.35-µm gate arrays.

Table I
Gate Arrays for S-Class and X-Class Servers

Chip Name	All Logic (Basic Cells)	Random Logic (Basic Cells)	Latch Array Bits	Base Type
Processor Interface	550 k	317 k	28 k	0.35 µm
Crossbar	500 k	206 k	29 k	0.35 µm
Memory Interface	570 k	358 k	25 k	0.35 µm
Node-to-Node Interface	300 k	175 k	43 k	0.35 µm
I/O Interface	150 k	106 k	6.4 k	0.5 µm

The characteristics of the Fujitsu CG61 0.35-µm CMOS gate array are as follows:

- 1.1 million raw basic cells. One basic cell can implement one low-power, two-input NOR gate or one medium-power inverter.
- Overall die size: 13.5 by 13.5 mm. Core size: 11.7 by 11.7 mm.
- 4 routing layers.
- 1.75-µm routing pitch on metal layers 1, 2, and 3; 14-µm on metal layer 4.

- 560 I/O signals.
- Flip-chip connection to package. Signals attach at periphery. Power and ground attach on a uniform grid across the die.
- Macros* generally had several functionally equivalent versions with different drive strengths.
- Target 45% utilization of raw basic cells in random logic regions.
- Many paths with 14 macros between registers at 120 MHz.

Custom I/O macros were developed to meet the stringent system electrical and timing requirements. The board designers analyzed all system paths with SPICE. SPICE models of the board nets included gate array driver and receiver macros.

The static flow is depicted in Fig. 1. The static flow does not reflect the actual day-to-day activity of the design process. Many iterations and subiterations were made until a chip could flow cleanly without timing or routing problems from start to finish. In addition to feeding back problems into the flow—for example, to update Synopsys constraints or make behavioral model changes—the design staff refined tools and techniques along the way. Rapid iterations through the place-and-route flow were vital to this process.

Typically, four designers worked on the larger arrays. One person worked almost exclusively on the floor plan, layout, and routing. The other designers, in addition to all other design duties, iterated through Synopsys and floor plan timing. Four full-time CAD engineers built and supported the flow.

RTL Behavioral Model Styles

The S-class and X-class server gate arrays were described in RTL-level Verilog (RTL stands for Register Transfer Language). These models were considered to be the "golden" models and were maintained throughout the life of the project. Design verification used the RTL models exclusively. The Boolean equivalence tool (lover) guaranteed that the RTL Verilog and gate Verilog were equivalent. The RTL Verilog style varied depending on the tools and the tasks for which it was written. Thus, the style was driven by the needs of Synopsys, the Boolean equivalence tool, and high-speed RTL Verilog simulation.

Style Driven by Synopsys. Synopsys was the primary tool to map RTL behavioral models to gates. Anyone familiar with this tool knows the requirements it imposes on the coding style: many small modules with lots of hierarchy, instantiated gates in the RTL, and code almost at the equation level. The project had two methods of instantiating gates in the RTL:

- Macro Group Specification. Based on previous projects' difficulties in synthesizing multiplexer macros, all scan registers and multiplexers were explicitly specified in the RTL. To make the RTL more readable and to improve RTL simulation speed, groups of macros were bundled under one level of hierarchy. For instance, 32 high-drive 2-to-1 multiplexer macros would appear in the RTL as:

```
MUX21_HI_32 muxes (
  .S  (<32 bit port connection>),
  .I0 (<32 bit port connection>),
  .I1 (<32 bit port connection>),
  .O  (<32 bit port connection>)
);
```

An internally developed tool read the RTL Verilog and generated the corresponding modules' definitions based on a vendor-specific mapping of the generic type to the actual type.

- 'ifdef GATES. Frequently, designers wanted to specify explicit macro types for some lines of code in a synthesized RTL module. The module would be written with:

```
'ifdef GATES
  <macro instance specifications>
'else
  <behavioral of the above gates>
'endif
```

For example:

```
'ifdef GATES
  wire t1, t2, t3;
  XOR3 xortree1 (t1, in<0>, in<1>, in<2>);
  XOR3 xortree2 (t2, in<3>, in<4>, in<5>);
  XOR3 xortree3 (t3, in<6>, in<7>, in<8>);
  XOR3 xortree4 (perr, t1, t2, t3);
'else
  perr = ^in;
'end
```

* In this context, a macro is a collection of basic cells that implement a primitive function like a multiplexer or register.

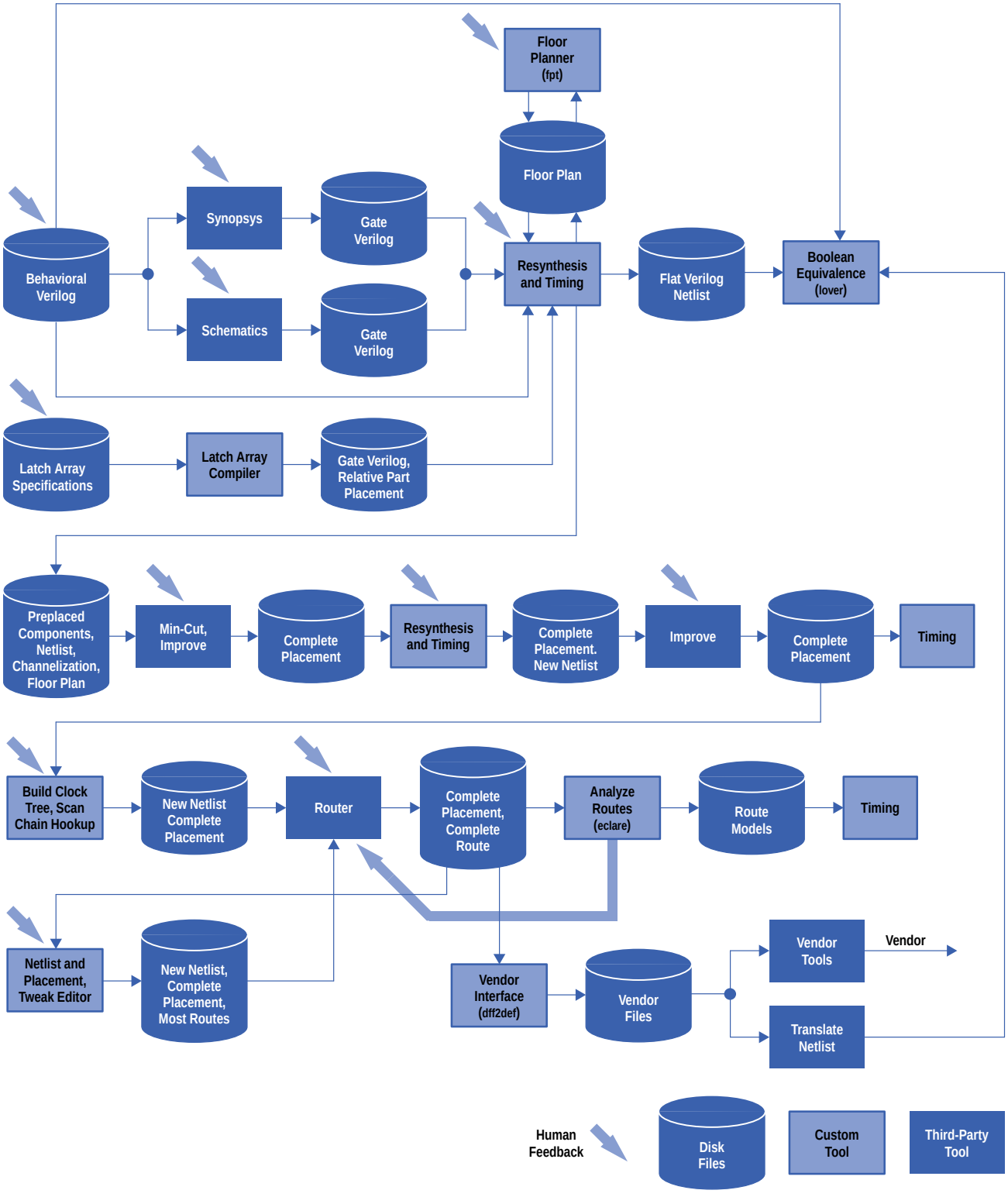


Fig. 1. *Simplified physical design flow.*

The GATES text macro was defined when synthesizing. This preserved clarity and speed of simulation in the RTL while at the same time generating the correct gate design in Synopsys. The Boolean equivalence tool ensured that the gate descriptions matched the behavioral descriptions.

Style Driven by Boolean Equivalence Verification. The Boolean equivalence tool (see "Formal Verification—Boolean Equivalence" below) logically compared the RTL behavioral model and the gate level netlist. This tool partitioned the design into cones of logic based on *equivalence points*, which are essentially net names that exist identically in both models. At a minimum, all register outputs are equivalence points. To keep the comparison tool simple, each equivalence point had to

match 1-to-1 between the models. This simplification required that registers replicated for fanout purposes in the gate netlist be replicated identically in the RTL Verilog.

Style Driven by Simulation Performance. Simulation performance was an important factor in each gate array's design quality and its development schedule. Faster simulation meant that we could get to a functionally correct gate array physical design more quickly. Thus, some of the RTL style methodology was driven by simulation performance.

For all of the hand-instantiated miscellaneous logic prefixed by 'ifdef GATES, the designers included the corresponding behavioral statement(s) bracketed with 'else ... 'endif. Also, guidelines were developed that pointed designers towards behavioral Verilog code constructs that simulate faster, and a linting tool was developed that pointed out where they could use these faster constructs in their Verilog. Examples included:

- Use of a mask plus a unary operator in place of the corresponding binary logic operation on selected bits of a bus, as in error correction bit calculations
- Use of a temporary variable in place of frequently referenced bit positions
- Concatenation in place of subrange assignments and shifts by constants.

In addition to simulating with vendor Verilog simulators, an in-house Verilog-to-C translation tool was developed to generate cycle-based models for each gate array type. The resulting simulations ran five to twelve times faster than the same simulations on vendor simulators.[1]

Because the Verilog language is extensive and expensive to support in its entirety, and because the internally developed tools were never intended for wide use beyond the boundaries of the project, the cycle-based Verilog-to-C translation tools supported only a subset of behavioral Verilog that is sufficient for cycle-based simulation. Where there were several different ways of describing the same design in Verilog, the tools supported the style that was prevalent among designers and less expensive to parse. Time constraints forced tool development to leave out constant expressions, loops, parameters, and variables as bit subscripts from the supported behavioral Verilog language style.

Latch Array Compiler

The latch array compiler allowed the designers to create new configurations of memory blocks easily. The memory blocks were implemented as a row of latches per data bit, with a column of latches selected by a read or write address. All latch arrays had a dedicated write port and either one or two dedicated read ports. All address, data in, and data out ports of the latch array had registers to isolate the internal timing of the latch array from the surrounding logic. Optionally, parity going into or coming out of the latch array could be checked. The placement locations of the write and read address ports were fixed, but the data input and output ports could be independently placed to the left or right of the core area.

The compiler's output consisted of the RTL behavioral and gate Verilog models for the configurations, as well as the relative placement information. This placement information was then given to the floor planner so that the latch arrays could be relocated to any desired site.

There were 16 to 41 latch arrays per design (118 total for four designs), with each latch array storing between 48 and 2112 bits, for an approximate total of 25K to 43K bits per chip, representing 35% to 60% of the basic cells used for each design. Through the use of custom macros designed specifically for the project, the core area of the latch arrays achieved over 87% utilization. For a single-read-port array, the core area consumed about 5.6 basic cells per bit, while the core area for a dual-read-port array used about 9.6 basic cells per bit. The core of the latch array was routed completely in metal 1 and 2, allowing the metal 3 over the core area to be used for normal routing.

Synopsys

Synopsys was used as the primary means to map nonregister and nonmultiplexer RTL to gates. The designers did not rely on Synopsys to generate the final gates. The designer's goal was just to get close to the final timing and logic area and then use the resynthesis step described later to improve the gates. There are two main reasons for this. First, Synopsys cannot read and optimize an entire design in a practical amount of time, and second, Synopsys cannot synthesize fanout trees based on the actual placement of macros.

Constraints for Synopsys were generated by hand.

Formal Verification—Boolean Equivalence

Formal verification using an internally developed Boolean equivalence verification tool (lover) eliminated a tremendous amount of gate level simulation. In fact, the system simulation remained at the RTL level throughout the entire project. Boolean equivalence verification, unlike simulation, guarantees complete results, since it uses mathematical formal proof techniques to verify logical equivalence. lover ran in three fundamental modes: RTL-to-RTL, RTL-to-gate, and gate-to-gate.

RTL-to-RTL comparison was critical when it became necessary to modify modules in the RTL to remove redundancies across module boundaries. Also, RTL-to-RTL comparison allowed the designers to explore different implementations of an equivalent design.

RTL-to-gate comparison took a lot of pain away from doing hand-captured gates. Miscompares could be isolated down to a few gates and fixed quickly. Also, RTL-to-gate comparison was useful as a revision control check by ensuring that any last-minute changes in the RTL source were synthesized into gates, or in other words, that design verification's RTL matched the actual netlist.

Gate-to-gate comparisons ensured that the CAD system (resynthesis in particular) maintained logical correctness throughout the flow. In addition, gate-to-gate comparison was used to ensure that all hand edits performed during the layout process were error-free. This allowed last minute changes to be made in the design with confidence.

Maintaining a consistent Verilog naming convention throughout the CAD flow facilitated the mapping of flat Verilog wire and register names to their hierarchical equivalents. This convention dramatically improved the performance of lover by providing additional *subequivalence points*. Using subequivalence points, large cones of logic whose boundaries consist of ports and registers can be broken up into smaller cones of logic automatically.

A complete gate-to-gate verification of a 550,000-gate design ran on an SPP-1000 (an 8-way symmetric multiprocessing machine with 2G bytes of main memory) in under 20 minutes and required 1.2G bytes of physical memory (see Table II). A complete RTL-to-gate verification of a 550,000-gate design was completed in 1 hour and required 1.2G bytes of memory. Most RTL-to-RTL subhierarchical comparisons completed in under 20 seconds. lover reads and compiles the RTL source files, netlists, and libraries directly. These times represent a combination of the compilation and comparison processes.

Table II
Gate-to-Gate Boolean Equivalence Run-Time Performance and Memory Requirements

Chip Name	Logic Size (Basic Cells)	Minutes	Virtual Memory
Processor Interface	550 k	20	1.2G bytes
Crossbar	500 k	9	0.9G bytes
Memory Interface	570 k	20	1.2G bytes
Node-to-Node Interface	300 k	10	1.0G bytes
I/O Interface	150 k	4	0.3G bytes

Floor Plan

The floor plan tool (fpt) helped the designer generate the placement for the major blocks of logic. The physical hierarchy of the design initially matched the logical hierarchy (a module in the source Verilog became a block for placement), but by editing the placement definition file, the physical hierarchy could be changed to anything the designer desired.

fpt read the same placement obstructions as the gate placement tool, so an accurate determination of the utilization of each block could be made as it was placed. The tool also understood the concept of fixed-size blocks (such as the latch arrays) as opposed to the malleable blocks which were primarily composed of synthesized logic, and allowed the latter block types to be reshaped.

Interconnect lines could be drawn between the block drivers and receivers, with the width of the drawn line representing the number of signals crossing the boundaries of the blocks. Inputs and outputs from a block could be drawn in separate colors to help illustrate the flow through the blocks.

fpt could display a *density map* that quickly identified local hot spots of high placement density. There was also a method to show information from a prior postroute analysis so that the floor plan could be adjusted to provide more room for congested routing areas.

Fig. 2a shows an example of a floor plan for one of the chips and Fig. 2b shows the placement view.

Timing

Timing analysis was performed by an internally developed tool because a third-party tool with the necessary features and performance was not available and resynthesis (see below) required tight coupling of the timing analysis to the automatic netlist editor. Each step in the flow (floor plan, placement, routing) used the same timing algorithm and timing tool. Differences in delays at each step resulted solely from the accuracy of the π-models and each macro input pin's T_{line} (RC delay). (A π-model represents the net + gate load as seen by the macro output.) For different steps in the design flow, the net π-model and RC delays were calculated as follows:

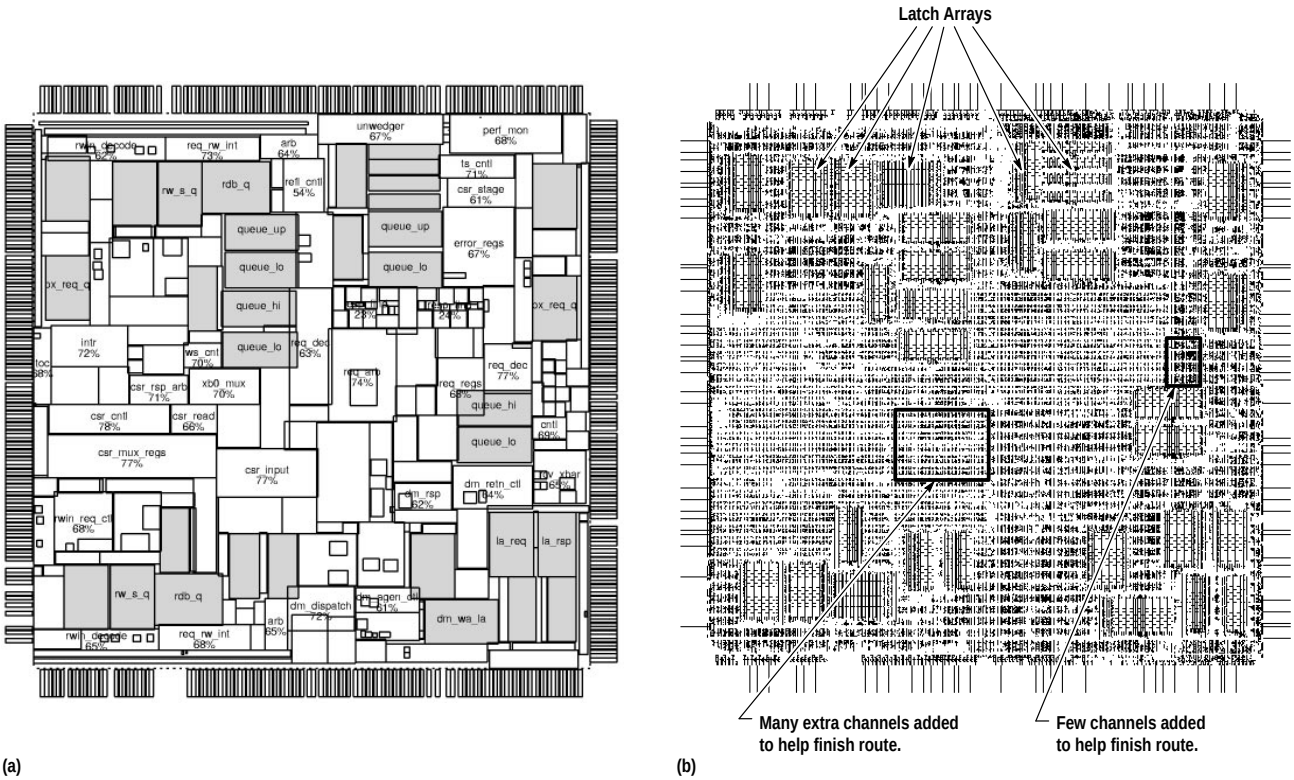


Fig. 2. *Processor interface gate array floor plan and placement views. (a) Floor plan tool (fpt) view. The block name and effective percentage utilization are shown. The effective utilization is based on the underlying obstructions used by the macro placement tools. Obstructions are usually channelization, latch array, or I/O regions. The shaded boxes are latch arrays. Blocks can be overlapped. A density map can be displayed, which shows where overlapped areas result in density that may be too high to place efficiently. Floor plan blocks that show similar effective utilization can actually be placed at different real utilization. For instance, csr_input at 77% (middle of die) and req_dec, also at 77% (right side middle above queue_hi), show similar effective utilization. Compared to the actual placement, csr_input is placed at lower real utilization because of the additional routing channels. (b) Placement view. In the placement view, the regions of custom channelization can be clearly seen. The latch array regions are also clearly visible. The actual size and number of the channels were determined by many iterations through the place-and-route flow, attempting to get a complete route. Register and multiplexer macros clustered around the I/O sites were placed identically to control chip-to-chip skew.*

- Floor plan. Wire table lookup was used for sections of the net that were completely contained within a block. Sections that spanned blocks were pseudorouted. The composite was then reduced to the π-model and T_{line} values.
- Placement. A pseudoroute was constructed for each net and then reduced to the π-model and T_{line} values.
- Routing. A special route analysis program was developed to analyze all metal layers and generate the π-model and T_{line} values.

Delay calculations were nonlinear and edge rate dependent. Presentation of edge rates in the critical path reports was very helpful in fixing slow paths. Slow edge rates were the first thing to look for (and fix) in a broken timing path.

Two delay values and two edge rate values were calculated per macro level:

- T_{gate} is the input-pin-to-output-pin intrinsic macro delay. T_{gate} is a function of the timing arc through the macro, the input pin edge rate (T_{sin}) and the π-model on the macro output. There are several T_{gate} calculations per macro, each corresponding to a different timing arc through the macro.
- T_{sout} is the edge rate seen at the macro output pins. For each T_{gate} calculation there was a matching T_{sout} calculation.
- T_{line} is the RC delay. T_{line} is a function of the net topology only.
- T_{sin} is the edge rate as seen at the macro input pins. T_{sin} is always greater than T_{sout} of the driving pin and represents the degradation of the signal edge as it propagates through the RC tree of the net. The edge rate degradation was a simple function of T_{line} and was not very accurate for complicated net topologies.

Each gate array used the same simple clocking scheme. At most there were three different clock phases, which were generated from the same master clock pin of the chip. The clock phases are 1 × period, 2 × period (2a), and 2 × period with the opposite phase (2b). The simple clock scheme allows a simple forward traversal algorithm to sum the delays T_{gate} and T_{line} to get path timing. Macro output pin edge rates (T_{sout}) were propagated and derated to the input pins (T_{sin}) during the same forward traversal. Typically, 1→1, 2a→1, and 1→2b paths were calculated in one pass, 1→1, 2b→1, 1→2a paths were calculated in another pass, and 2a→2a and 2b→2b paths were calculated in a third pass.

Resynthesis

Resynthesis can be thought of as an elaborate in-place optimization of the entire design. Resynthesis was used to reduce or remove logic, build and place optimal fanout trees, and change macro power levels to the minimum necessary to meet timing requirements. There were six steps in the resynthesis process:

- Flatten. The gate level Verilog modules were read in, linked and flattened so that the entire design could be easily optimized across hierarchical boundaries.
- Rip/join. Buffers and inverter pairs were deleted, simple macros were joined together, logic was reduced through constant propagation, and unused logic was deleted.
- Coarse fix. Each macro output pin in the design was either powered up (i.e., its drive strength was increased) or fanned out such that the output pin edge rate was below an arbitrary value and the input pin edge rates on the net were all below a (different) arbitrary value. This step also ensured that each macro output pin did not drive more than its maximum capacitive load.
- Complete timing analysis.
- Fine pass. Once the coarse fix was made and macros were powered up to meet the edge rate specification, a topological traversal starting from the register inputs and working backwards was made to further power up macros that were in broken critical paths. This step typically did not add more than 2000 basic cells to the design. Timing and pseudoroutes were updated after each macro power-up. The timing update algorithm retimed only those macros affected by the power-up.
- Power down. The coarse fix step and Synopsys can overpower components. A topological traversal was made starting from register inputs and working backwards to power down those components in noncritical paths (i.e., that had positive global slack) as long as the net loading rules were met. Timing and pseudoroutes were updated as the algorithm proceeded.

Resynthesis was run at two different steps in the flow. A first-pass resynthesis was done using floor plan net lengths and floor-plan-based approximate macro locations. This design was then placed and improved. The improved placement and netlist from the first resynthesis were then fed back for a second complete resynthesis. (Fanout trees built during the first resynthesis were ripped out by the rip/join step.) The second resynthesis resulted in about 1000 fewer fanout buffers and 10,000 fewer basic cells used than the initial floor-plan-based resynthesis.

Resynthesis and timing of the processor interface chip (317,000 random logic cells, ~91,000 macros) typically took 16 CPU minutes on an HP 9000 Model 735 workstation and consumed 320M bytes of virtual memory.

Figs. 3 and 4 show the differences in the delays seen at register inputs between timing runs made at different steps in the flow. Fig. 3 shows the differences between floor plan timing after resynthesis and final placement timing. Values less than zero show floor plan timing that was conservative compared to the final placement timing. Values greater than zero show floor plan timing that was optimistic compared to final placement timing.

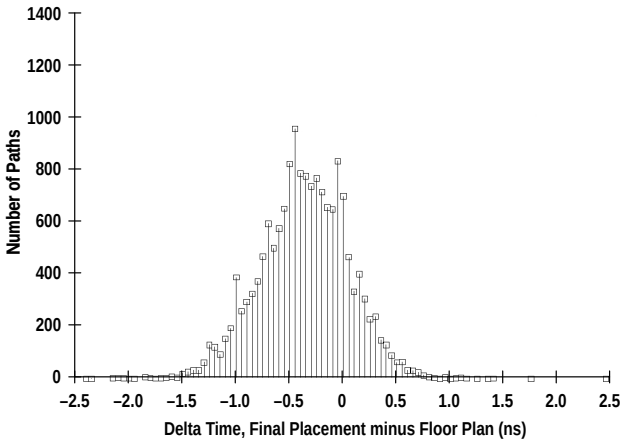


Fig. 3. *Timing comparison: floor plan versus placement.*

Fig. 4 shows good, albeit conservative, correlation between the placement and route timing. Relatively few changes had to be made in the design once the designers had timing working at placement. The histograms clearly show that an attempt to "make timing" at the Synopsys/floor plan step is unnecessarily conservative.

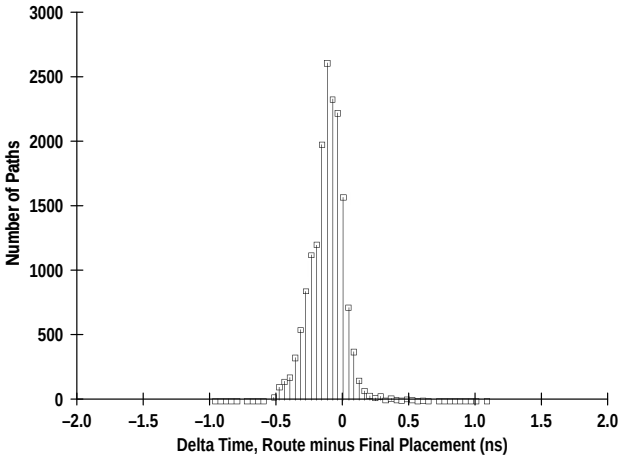


Fig. 4. *Timing comparison: placement versus route.*

Placement

The GARDS gate array suite of tools from Silicon Valley Research (SVR, formerly known as Silvar-Lisco), was chosen as the central set of tools for placement of the server gate arrays. The choice of GARDS was based on a history of successful experience with these tools. A long-held philosophy in our laboratory has been to put the place-and-route tools in the hands of the designers, and to choose the tool that best allows the designer to parlay design knowledge into an intelligent placement. With only minor exceptions, the GARDS tools functioned capably and fulfilled the place-and-route needs for which they were chosen.

Unlike many other design flows for 0.35-µm gate arrays, hierarchy was flattened by the resynthesis step, and the placement tool had free rein to optimize placement across boundaries. A drawback of this approach is that major updates to a current placement (e.g., synthesize one block and feed the changed block into the existing placement) were virtually impossible. In such cases, the placement process was restarted from the beginning.

Complementary internal tools constructed all necessary files in GARDS format, such as placement obstruction files for opening extra routing channels and for inhibiting automatic placement in critical regions such as latch arrays.

A valuable feature of the GARDS tool set is the C programming language database access routines known as GEARS. These made it easy for internal tools to extract design information to customize the integration of GARDS into the tool flow.

The GARDS libraries for placement and routing were created internally by translating Cadence LEF (Library Exchange Format) files provided by the chip vendor. The general placement scheme consisted of the following series of steps:

1. Import of preplaced gates (I/Os, internal macros in the I/O-to-register or register-to-I/O paths, and latch arrays)
2. Import of placement obstructions (for channelization, latch array protection, and clock tree allocation)
3. Min-cut placement of the imported floor plan. Min-cut placement is a recursive subdividing algorithm that seeks to minimize connections across the dividing line. The resulting placement has minimum wiring between macros.
4. Run checkfif (internally developed tool).
5. Unplace errantly placed components.
6. Min-cut placement of floor plan file containing only the errantly placed components.
7. Run a series of walking-window improvement passes.
8. Repeat steps 1 through 7 after a layout-based resynthesis.

Steps 1 through 3 generally took about two hours for the larger designs, and could be automated through scripting. After the initial read of the floor plan, it was necessary to check the integrity of placement adherence to the floor plan. An unfortunate feature of the placement tool was that if components could not be placed within their assigned floor plan bin, they would be placed according to the center of gravity of the connectivity, which usually was not acceptable.

Step 4 involved an internal tool, checkfif, which was developed to unplace errantly placed components and to create a new reduced version of the floor plan bin file that could increase the original dimensions of the bins. Steps 5 through 7 could also be scripted, with step 7 consisting of a mix of net-length-based improvements and congestion-based improvements. Step 7 represents the largest investment in CPU cycles, with a typical set of improvement passes taking about 24 hours.

Clock Tree Builder

Once placement was complete, the clock tree builder was used to generate different gated types of the master clock. The structure of the clock tree, which was identical for all CG61 arrays, was a uniform grid of taps driven from a central point (see Fig. 5). A tap was either an OR gate, which generated some form of gated clock, or a buffer, which generated a free-running clock. Macro positions and most connections were predefined, with the only variables being the tap type (OR gate or buffer), the tap flavor (which gating term drove the OR gate, if any), and the last-level connection to the registers.

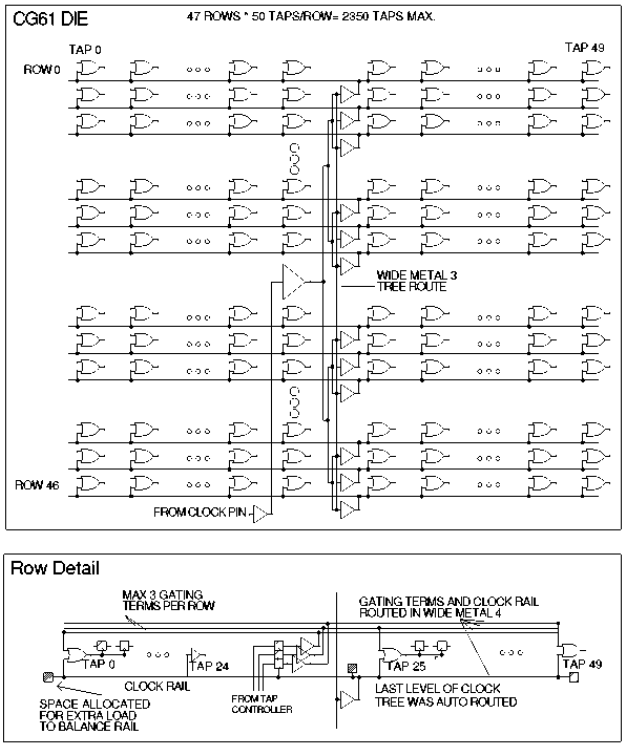


Fig. 5. *Clock distribution tree.*

The clock tree builder removed taps obstructed by latch arrays, figured out a best initial assignment of taps associated with a gating term register, and then minimized the last level of clock tree routing by optimizing the clock-load-to-tap assignments. The last level of clock tree routing was performed by the autorouter. The clock tree builder provided the user with the flexibility of manually selecting tap gating term assignments and tap load assignments. The complete clock tree consumed 1.5% to 2% of the die area.

Skew

Intrachip skew was calculated as the difference in delay between the fastest tap and the slowest tap. The maximum intrachip skew for these gate arrays totaled 400 ps, of which 200 ps was actual metal and gate load variations and 200 ps was intrachip variance.

Metal and gate load variations were mostly due to different numbers of loads and different route lengths in the last level of the tree. Dummy loads were added to the tips and center of the main clock rails after routing to balance the rails precisely. Special control of the last-level loading and routing of I/O registers reduced the metal and gate load skew on those registers to 100 ps.

The variance portion of skew included the voltage, temperature, and process variations possible across the die. Variance also accounted for delay modeling uncertainty between OR gates and buffers in the last level of the tree.

Scan Chain Hookup

Once all the scan elements were placed, the scan chain was automatically connected. Each element had a property that identified the element's scan ring. The beginning and end of each ring had a property that marked these points as special. Buffers and pairs of inverters that were connected to the scan points were marked as equivalent to the scan point and could also be used. Each ring was stitched together using a "greedy" algorithm that chose the best element to add at each step. The best element was determined by assigning a penalty value to each available element and then choosing the one with the minimum penalty. The penalty value was a summation of four individual penalties:

- Distance. The distance between the element and the connection point. Reducing this value decreased the amount of routing needed to connect the ring.
- Clock. A penalty was assessed if there was a difference between the clock for the element and the clock for the connection point, that is, if the clocks came from different taps or rails. The more the clocks were similar, the less the chance for hold errors while scanning.
- Buffering. A penalty was assessed if the connection point was not buffered from the true scan point. Preferring the scan points that were buffered reduced the possibility of hold problems.
- Center crossing. The connection crossed the vertical centerline. The main clock distribution was in this area and horizontal routing was limited. Reducing the horizontal scan connections through this area simplified the routing for the design.

Each of the above individual penalties had an associated weight so the relative importance of each could be adjusted.

For the designs that had routing completion problems, a post-hookup optimization could be applied. This optimization only considered the distance between scan points as the objective to minimize and often resulted in over a meter of wire removed from the scan chain.

After all of the scan rings were connected, the wires added at the gate level could also be inserted into the behavioral model.

Routing

Routing of the chips was accomplished with the GARDS route tools (garout/linesearch, garout/maze, redit/ripup-reroute). Normal signal routing occurred on metal layers 1, 2, and 3, with only clock net routing on metal layer 4. The version of GARDS used by the server project did not include support for fourth-layer routing, so fixed metal 4 segments were specified and connected with internally developed postprocessing tools dff2def and eclare (see below).

The routing process began with certain special nets. These were mostly latch array nets and certain clock nets, which needed to be routed with the maze router (garout/maze). Then the line search router (garout/linesearch) was used to route nearly all of the remaining connections, leaving only about 1% to 1.5% failing to route. These were routed by the maze router, usually to within about 200 disconnects. At this point the design was taken into the redit tool, using the ripup-reroute feature of that tool. This normally completed the remaining connections. The whole routing process generally took one to two days to complete.

Some special techniques were exploited to push the envelope of routing density. It was observed that metal layer 1 was underutilized, and that some of the saturated metal 2 routing could be forced down to metal 1, thus allowing extended use of metal 2. This was done with an internally developed tool that read the routing that had occurred up to a certain stage, determined candidates for moving metal 2 horizontal segments to metal 1 horizontal segments (nonpreferred direction), and finally moving those segments. This technique helped complete the routing in critically dense areas.

Another technique for achieving better routing completion was channelization (see Fig. 2). When a routing attempt failed to make all connections, additional routing channels were created in regions where concentrations of disconnects were observed. The channel obstructions were fed back to the floor planner, where the designer could make sure that the effective utilization of each block was still below the safe limit. If necessary, blocks were enlarged.

Postroute Analysis

After placement and routing were complete, the capacitance and RC delays were extracted by an internally developed tool, eclare. eclare read the GARDS database to obtain the macro and routing information for the design. The Verilog gate level netlist was also read so that the connectivity present in the GARDS database could be verified against the netlist connectivity. eclare then used the GARDS data to construct four arrays of grid points (one per metal level) that described the metal present at every grid location. The information stored at each grid point included metal width, type of metal, directions of connections, an index into a table of pin and net names, and a visited flag.

Storing this much information per grid point is expensive in terms of memory (720M bytes of storage is required to hold the grid data for the current chips), but it greatly simplified the determination of neighbors during the extraction process. The project had access to several machines that had 1G bytes (or more) of physical memory, so the storage requirement was not a serious impediment. This data structure also allowed the run time of the tool to be fairly short. eclare ran in less than 30 minutes, with about half of the time being file input/output.

The extraction process began after all of the grid information was loaded. For every driver on each net, eclare traced all of the metal attached to the driving pin. As the net was traced, a list of nodes and segments was built for use in solving for the equivalent π-model and RC delays.

For each grid point traced, nearby grid locations were examined to determine their contents. When all of the neighbor information was determined for the current grid point, a lookup table was consulted to obtain the capacitance value for this neighborhood configuration. This capacitance and the resistance for the current metal level were added to the appropriate entry in the segment list. The capacitance value was also added to a counter for each of the neighboring nets for use later when determining the coupling factor.

When all of the net had been traced, the list of nodes and segments was used to solve for the RC delays to the input pins and the equivalent π-model at the driver. All neighboring nets were examined to determine the maximum amount of capacitance that was coupled to the target net. If the maximum coupled capacitance to any single neighboring net exceeded 20% of the total capacitance for the current net, the routing had to be adjusted to decrease the coupling. The coupling percentage was used to create a minimum/maximum range of capacitance values in the equivalent π-model that was later used by the timing verifier.

eclare performed a few other tasks in addition to the extraction. The Fujitsu toolset checked for nets that failed their antenna rule (too much unterminated metal connected to the gate of a transistor during fabrication). eclare also made this check, but earlier in the overall design flow. Optionally, eclare could output a distributed RC SPICE model for any of the nets in the design. Finally, eclare produced a file of clock rail loading that was used to determine the number of rail balance loads needed to equalize the total capacitance on each rail.

Last Minute Changes

Given the large latency from RTL behavioral change through Synopsys and resynthesis to placed design (usually three to four days), an interactive *tweak editor* was constructed to allow designers to edit the netlist and placement interactively. This X Windows tool displayed the placement, netlist, and timing information graphically and then allowed the designers to add, change, delete, and move parts and make other changes. Global chip timing information was updated in real time so that the effect of these netlist edits was instantaneously viewable.

The tweak editor gave the designers more than enough latitude to change the logical behavior of the gate design (and sometimes that was the exact intent). The new netlist was logically verified against the RTL behavioral model with lover, the Boolean equivalence verification tool.

Vendor Interface

The vendor interface consisted of a front end, where libraries of various formats were received from the vendor, and a back end, where ultimate tape-out-ready design files were sent to the vendor for go-to-make.

On the front end, the vendor supplied all necessary libraries. Some of these were characterization data files in the vendor's internal format, and the rest were in various standard formats, such as Verilog and LEF. Also provided was an integrated set of programs that were pieces of the vendor's own proprietary ASIC development package. These programs, which included the vendor's LCADFE and GLOSCAD packages, also included library data.

The physical libraries provided by the vendor were in the standard Cadence LEF format. An internally developed tool, lef2gards, was used to translate the LEF description of the base array and the macros into GARDS library formats. The lef2gards translator had been used in previous projects, and only required some minor modifications to work with the vendor's CMOS LEF files.

On the back end, another tool, dff2def, was developed to read the GARDS design file (using GEARS access routines) and produce a logical netlist in the vendor's proprietary format, FLD, as well as a Cadence DEF (Design Exchange Format) file, which contained all of the physical information such as placement and routing. The FLD file was imported into the LCADFE vendor package, where certain netlist expansions and design rule checks were done. Then the GLOSCAD package was used to import the LCADFE logical data and the DEF file physical data. This was followed by a run of the GLOSCAD laychk program, which did a thorough physical design rule check. Finally, the GLOSCAD pdilmake program produced the π-model and RC data, which was folded back into the LCADFE package for postlayout timing simulation. A complete set of vector simulations was then run to verify both timing and logical aspects of the design.

After determination of successful timing and clean LCADFE and GLOSCAD runs, and after passing a stringent tape-out review, the appropriate design files (essentially tar snapshots of the vendor's LCADFE and GLOSCAD user design tree) were transferred via ftp to the vendor's site. A go-to-make release usually followed within a few days of the original transfer.

Conclusions

Commercial synthesis and layout tools alone were not sufficient to achieve the ASIC performance, density, and time-to-market goals of the S- and X-class machines. A significant investment was made in both internally developed and commercial CAD tools to put together a highly optimized physical chip design process. The resynthesis process coupled with in-house place-and-route tools minimized direct human interaction in optimizing the synthesized designs and made it possible to achieve the project design goals, which would otherwise not have been possible.

Acknowledgments

The authors cannot overstate the importance of the work of Harold Dozier in providing the fundamental technical interface with Fujitsu and implementing many of the features of this flow.

Reference

1. L. Bening, *Putting multithread behavioral simulation to work*, 1996, accessible on the World-Wide Web at:
http://www.convex.com/tech_cache/supercon96/index.html

Fast Turnaround of a Structured Custom IC Design Using Advanced Design Tools and Methodology

Through the use of several new tools and methodologies, a small team of engineers was able to design and verify a 1.7-million-FET chip in eight months. The tools and methodologies used included a set of guidelines and timing constraints that were met by the customer, a data path compiler, a highly tuned custom multiplier cell that was used in 87 locations, and an automated top-level power connection scheme.

by Rory L. Fisher, Stephen R. Herbener, John R. Morgan, and John R. Pessetto

The HP IMACC chip was developed to provide image processing capabilities. The initial target application is medical imaging with geological applications as a potential area of expansion. The graphical capabilities of IMACC include spatial filtering, edge detection and enhancement, image pan and zoom, image rotation, and window and level control. IMACC consists of three major components:

- The convolver circuit has a 3×3 programmable kernel* and can perform low-pass or high-pass spatial filtering, edge enhancement, and other functions.
- The interpolator is an implementation of a 4×4 bicubic convolution kernel.* The interpolator can be configured to perform pan, zoom, and rotation.
- A RAM-based lookup table is used for windowing and leveling of image pixel intensities.

In support of the various user-selectable operating modes, any or all three of the functional blocks may be active at a time. The order of operations can be changed as desired, with the single limitation that the convolver must precede the interpolator if both modules are in the chain. When the image visualization accelerator board (IMACC is the heart of this board) is attached to the HP HCRX graphics subsystem, simultaneous convolution, zoom, rotation, and window and level control of 1024-by-1024 pixel, 16-bit medical images at 40 frames per second are possible. The accelerator can process more than 40 million pixels per second independent of the number or order of internal operations.

Customer Interaction

As a result of our experience in designing numerous ICs for various customers, our laboratory has developed some practical, informal guidelines for designing ICs. At the beginning of the IMACC project, we met with the customer (another HP laboratory) and discussed these guidelines along with project goals. The guidelines we provided to our customer are as follows:

- The prime directive: Signal groups such as multistate drivers on a single bus, multiple set signals into a flip-flop, or multiple set signals into a multiplexer, may cause drive fights and therefore need to be completely decoded from the current state of the control machine so that one and only one will fire. This requirement must hold even if the chip comes up in a random state. Exceptions to this have caused significant delays in schedule right before tape release.
- Signals require a consistent naming convention.
- Update flip-flops on the falling edge of the clock (single-edge timing).
- When glitchless values are required (Gray code counters, etc.), they must come directly out of flip-flops.
- Resets are typically heavily loaded and will probably cause timing problems. Have each control block latch its own version of the chip reset, then generate its own local reset. This helps timing at the expense of latency.
- Don't design multistate paths. Complete timing analysis of such a path is not possible in any design tool.
- Don't set and dump the same FIFO or RAM location at the same time.
- Keep Synopsys blocks small.
- Keep large register files in the data path.

* A kernel is a functional unit that can be repeated as needed. A 3×3 programmable kernel performs a programmable function on a 3×3 array of pixels. A 4×4 bicubic convolution kernel performs a bicubic convolution on a 4×4 array of pixels.

- Use no clock uncertainty (skew) in Synopsys. It will be there, but is better allowed for by reducing the period.
- Don't allow Synopsys to try to fix hold problems. There should be none by design.
- When setting your timing constraints, allow some slack for RC delays, the local clock generator, and incremental delays that will be introduced when actual routing capacitances are substituted into the timing model. For example, 15 ns might be a good period constraint at 60 MHz.
- As constraints (timing and loading) become more accurate, make sure to continue to update them in your design. Accurate is better than conservative in Synopsys.
- Simulate at the board level as soon as possible.
- Simulate timing between blocks as soon as possible (schematic simulations are fairly accurate).
- Simulate the chip coming up in random states as soon as possible. A proven way to do this is to make sure the chip can come up with unknown values in all memory elements, including flip-flops and registers.

The highest-priority design goal was to have a working IMACC system to demonstrate at an upcoming conference. We created a schedule consistent with this goal incorporating the necessary checkpoints. Two of the most important checkpoints:

- We were to deliver a top-level, schematic-based Verilog gate model to the customer so they could begin regression tests at the system level. This allowed them to identify design problems early.
- The customer was to freeze the function of major data path blocks by a scheduled date. This allowed us to construct the artwork in a single pass.

Setting a rigorous schedule as the first priority forced a streamlined design. Using a single clock domain made the design less complicated. A large data path block with noncritical functionality was eliminated because it would require too much design time. The die size was determined early and a previously characterized package was used. Timing budgets were kept conservative to ensure that IMACC would run at 45 MHz after parasitic loading was added.

Since all decisions were based on meeting our primary objective, "creeping featurism" was eliminated. The customer was informed of the schedule impact that the addition of a new function implied. In most cases they were not willing to suffer a postponement of the chip release date.

We also shortened the design time by making sure that when we sent them a new gate model of the chip, we had all the latest changes and we had no problems with the syntax of the model. To do this we used several tools (awsim, eval, etc.) to check for connectivity problems. We used an in-house history management system to maintain revision control of these gate models.

Custom Multiplier

One of the key pieces of circuitry on IMACC in terms of design leverage was an integer multiplier. The imaging algorithms that we implemented typically executed a large sum of product terms. As an example, the convolver block in IMACC sweeps a 3×3 matrix across the source image (a 2D array of pixels), multiplies the nine coefficients in the matrix by the corresponding pixels, and adds these nine products to compute a single new pixel value for the target image. Therefore, this block alone required nine multipliers and eight adders. With the customer's help we found that we could consolidate most of the multipliers into one design. The result is that in the IMACC design, a single 18×18 integer multiplier circuit is used 87 times.

It then became very important to make this multiplier as dense as possible. This problem was attacked on two levels. First, an area-efficient architecture was chosen, and secondly, the key multiplier cell was painstakingly designed and laid out to reduce its area as much as possible.

The architecture we chose was a radix-4 (two bits at a time) Booth-encoded array.[1] A standard multiplier array in our case consisted of 18 rows, each row representing the result of one of the multiplier bits times the whole multiplicand value. Since the Booth-encoded array looks at two bits of the multiplier times the whole multiplicand in each step, this cut the number of rows in half to nine. It turned out that the increase in the row height resulting from a more complex unit cell was well below the height of two of the standard rows. (However, there is additional circuitry that needs to be added to perform the Booth encoding of all of the sets of two multiplier bits.) As a side benefit, cutting the multiplier rows in half also increased the circuit's speed.

This speed-up benefit helped with our second task of reducing the area of the critical multiplier cell. Our design requirement was to execute the 18×18 multiply in a single clock state (22 ns). When the circuit was initially built, it executed the multiply in about two thirds of the required time (about 14 ns). Therefore, it was possible to shrink the critical cell by reducing the FET sizes until the cell delay increased the multiply time to the 22-ns limit. Along with some diligent artwork layout, this produced an extremely dense cell.

Our final 18×18 Booth-encoded multiplier is 864.0 μm wide by 307.8 μm high for a total area of 0.266 mm^2. Using the standard multiplier array, this multiplier would have been 648.0 μm wide by 446.7 μm high for a total area of 0.289 mm^2. Considering just the areas of the multipliers, the savings per multiplier is 0.023 mm^2 and the total savings for IMACC is 2.0 mm^2. However, the aspect ratio of the standard multiplier did not fit well with the IMACC data path blocks (18 bits wide

versus 24 bits wide). Taking into consideration the empty spaces that the standard multiplier would have produced, the savings per multiplier increases to 0.123 mm^2 and the total savings for IMACC increases to 10.7 mm^2.

The custom multiplier design cut schedule time for two reasons. Using it 87 times in numerous blocks across the chip saved us the time we would have spent assembling (and possibly designing) multipliers for each specific need in the different blocks. Secondly, the significant area savings we realized made the job of top-level chip routing much easier and consequently faster, since the top-level blocks were smaller.

Data Path Compiler

We were able to save many hours of artwork layout time through the use of Dpc14,[2] a data path artwork generation tool that places and routes data path blocks. A data path is made up of hierarchical horizontal slices, often called macro cells, that are usually, but not always, bit-wise logic repeated as many times as needed. Examples of macro cells are a two-input, 32-bit register, an 18-bit ripple carry adder, another data path block, or a custom data path block. The program dpc_tiler was used to build many of these macro cells. Macro cells can be placed vertically or horizontally with respect to one another. Routing is done over and between macro cells to connect signals within the data path. The user has the ability to control placement and how signals are routed. The user can assign signals to specific data path tracks or defer routing of signals such that the *allow* layers for the signals (layers that can be used later in the design to route signals) are placed on unused tracks.

To use Dpc14, we first generated a Dpc14 input file, which was created from the schematic Block Description Language (BDL) file. The Dpc14 input file could also be generated from a Verilog netlist file. The tool bdl2dp was used to generate the input file from the schematic BDL for most of the IMACC data path blocks that used the schematic BDL. A simple input file is shown in Fig. 1.

```
-block dpc1
–trackassign a 5
–dpwidth 24
–newrow {
  –inst HOLE TOP 24 0 {
    –dpsig A[23:0] a[23:0] 0
    –dpsig B[23:0] b[23:0] 0
  }
}

–newrow {
  –inst REG224 rega 24 0 {
    –dpsig INA[23:0] a[23:0] 0
    –dpsig INB[23:0] b[11:0] 0
    –dpsig OUT[23:0] out[23:0] 0 –metal1
  }
}

–newrow {
  –chanroute { –signal cinput[23:0]
–right }
  –inst MUX224 muxa 24 0 {
    –dpsig INA[23:0] a[23:0] 0
    –dpsig INB[23:0] out[23:0] 0 –metal1
    –dpsig INC[23:0] cinput[23:0] 0
    –dpsig OUT[23:0] muxa[23:0] 0
    –cntlsig SELA sela 0
    –cntlsig SELB selb 0
    –cntlsig SELC selc 0
  }
}

–newrow {
  –inst HOLE BOTTOM 24 0 {
    –dpsig muxa[23:0] muxa[23:0] 0
  }
}
```

Fig. 1. *A simple input file for the Dpc14 data path artwork generation tool.*

Artwork encapsulation information needed to be extracted for each macro cell used in the data path. The Perl script dpEncapInfo was used to extract information about how to connect to ports within the macro cell. It was necessary to specify to dpEncapInfo any wart cells on the left or right side of the macro cell. For instance, a typical DPLIB14 register has two wart cells on the right side of the macro cell. In this case we used dpEncapInfo –r 2 to build the encap_info file correctly. This specified that there were two wart cells on the right side of the register macro cell.

After the Dpc14 input file was created and encapsulation information had been extracted, the Dpc14 program was used to generate an artwork archive that could be read into our artwork editor. The Dpc14 file could then be edited if the artwork needed to be modified, allowing us to make as many iterations as needed to produce the desired result.

Local Toolbox

A number of relatively simple scripts and programs of our own devising were combined into a local toolbox for the project. The more significant of these are described here.

The mkcntl script uses Synopsys, block place-and-route, and other tools to go from a Synopsys netlist through schematic to artwork with parasitics in one command. Of course, one must iterate on the place-and-route portion to ensure workable size, form factor, and port locations. An iteration can be accomplished with mkcntl –b.

We used a connectivity tracer that reads schematic (scip) BDL and reports the connectivity of the specified instance or net. The trace is limited to one level of hierarchy. For net names, regular expression pattern matching is available.

We automated a lot of the top-level power connection of IMACC using two scripts. Several steps were involved in this methodology. First, two symbolic layers were used, one for VDD and one for GND, to define where the metal-4 power buses would go. Next, the getpwrconn script used trantor to find the intersection of the metal-3 power buses with the symbolic layers and dumped them into an artwork editor archive file. Last, the gen_pg_conn script placed contact-4 contacts in the intersection areas and filled the symbolic layers with metal 4.

Our addallow script selects VDD, GND, and CLK metal-3 shapes by size and copies them to metal3.allow and contact4.allow layers, permitting connection to over-the-cell metal 4.

There are two shieldmaker scripts in our toolbox: addshield2 and addshield3. These scripts fill the unused areas of the intermediate (cross-channel) metal layer in routing channels. These areas are then tied to GND or VDD to provide crosstalk shielding for signals running the length of the channel.

Diode placing software was used to eliminate large numbers of charge collectors. This software finds traces longer than 1000 μm in a routing channel and locates areas where diodes can be added without introducing design rule check errors. This technique worked well for our project, which had a tight schedule, lots of charge collectors, and minimal timing problems.

All of the scripts that add shapes to a source block do so through intermediary block pieces to ease modification or rebuilding of the added function.

Results

The IMACC chip was demonstrated in systems at the Radiological Society of North America conference in Chicago on November 27, 1995.

The IMACC chip contains 1.7 million FETs in the HP BiCMOS14QC process operating at a 45-MHz clock frequency. It is predominantly a data path design with 98 integer multipliers performing 4 billion integer multiplies per second on 18-bit or larger operands. A breakdown by design style is as follows:

Style	Percent of Total FET Count	Number of Standard Cell Gate Equivalents	Thousands of FETs per mm^2
Data Path	60%	497,000	22.0
Standard Cell	10%	53,000	10.8
RAM, FIFO	21%	(>48K bytes)	42.7
Pads, Clock, etc.	9%		

Some additional statistics for the IMACC project include: 2342 FETs/day, 8673 FETs/mm^2, and more than 550,000 standard cell equivalent gates.

Acknowledgments

We would like to thank Rich Nash for his time spent developing Dpc14 and for his timely responses to our suggestions.

Reference

1. S. Waser and M.J. Flynn, *Introduction to Arithmetic for Digital Systems Designers*, Holt, Rinehart, and Winston, 1982, pp. 132-137.
2. R. Nash and R. Martin, "Datapath Requirements for Structured Custom Design," *Proceedings of the 1995 HP Design Technology Conference*, pp. 411-418.