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.

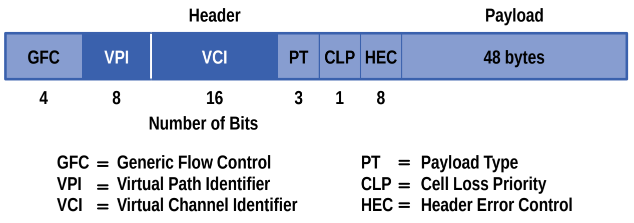
Header
Payload
GFC
VPI
VCI
PT
CLP
HEC
48 bytes
4
8
16
3
1
8
Number of Bits
GFC = Generic Flow Control
VPI = Virtual Path Identifier
VCI = Virtual Channel Identifier
PT = Payload Type
CLP = Cell Loss Priority
HEC = Header Error Control

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.

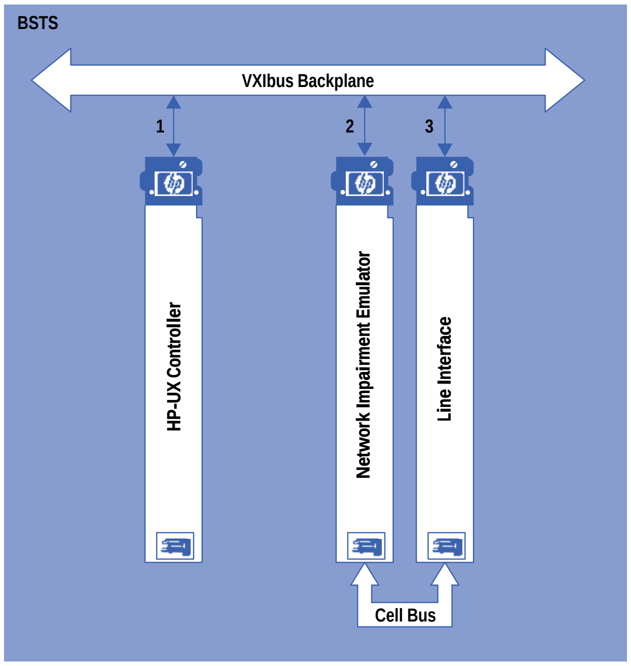

Fig. 2. *Diagram of the HP 75000 Broadband Series Test System with network impairment emulator.*

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.

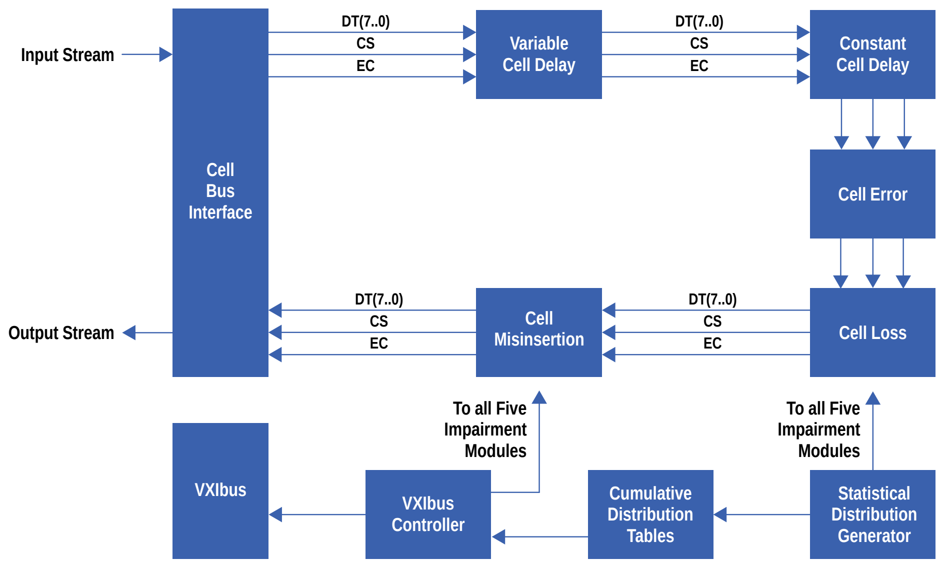

Fig. 3. *Block diagram of the HP E4219A network impairment emulator.*

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.

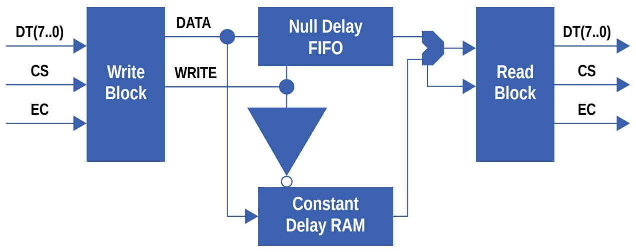

Fig. 4. *Block diagram of the constant cell delay module.*

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.

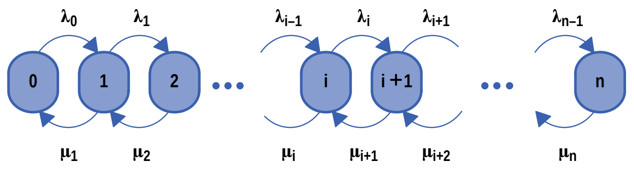

Fig. 5. *Birth-death Markov chain model. The λ_i and μ_i are birth and death rates.*

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:

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

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.

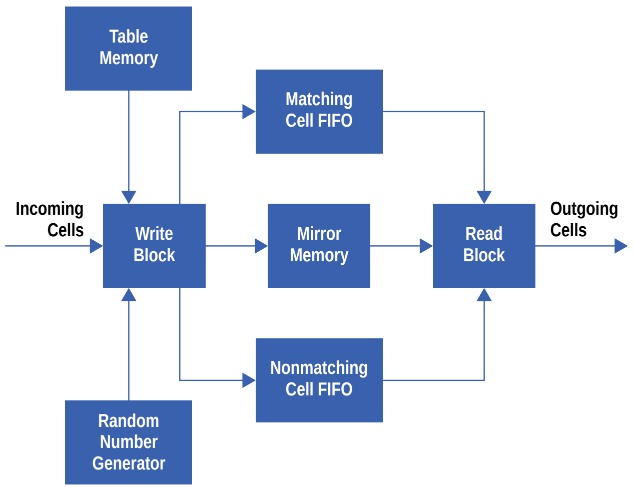

Fig. 6. *Block diagram of the cell delay variation module.*

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.

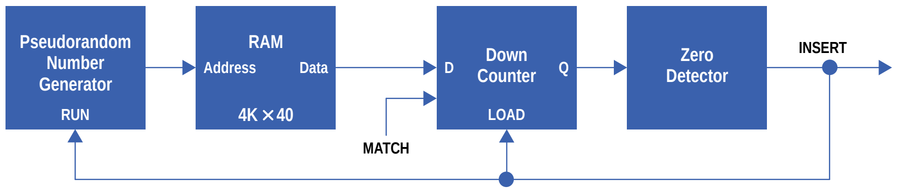

Fig. 7. *Block diagram of a statistical distribution generator.*

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

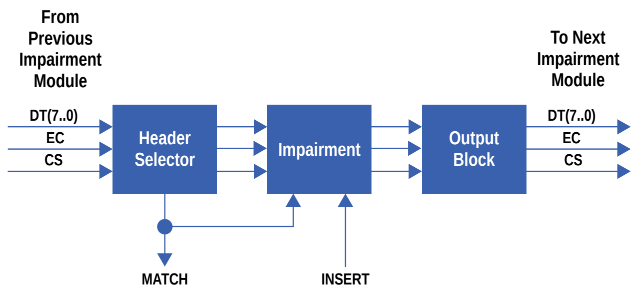

Fig. 8. *Generalized block diagram of the cell loss, error, and misinsertion modules.*

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.

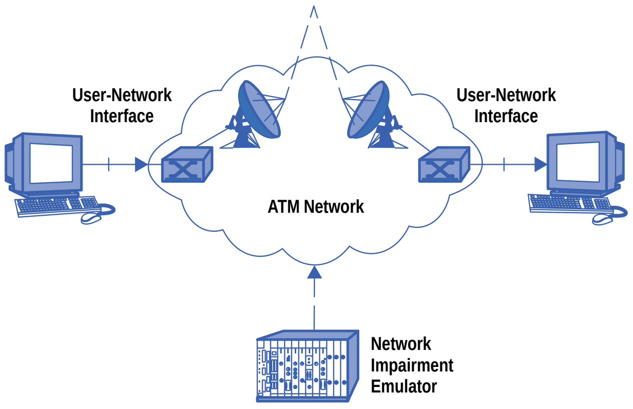

Fig. 9. *Emulation of a satellite link with the HP Broadband Series Test System and the HP E4219A network impairment emulator.*

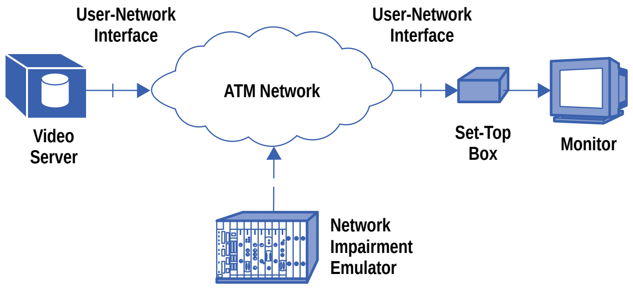

Fig. 10. *Determining acceptable quality of service in a digital video system.*

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.

▶ Go to Next Article
▶ Go to Journal Home Page