Self-timed interface of the input/output subsystem of the IBM eServer z900

by J. M. Hoke
P. W. Bond
R. R. Livolsi
T. C. Lo
F. S. Pidala
G. Steinbrueck

The self-timed interface (STI), used in several generations of IBM's most powerful servers, has evolved to provide a greater I/O subsystem bandwidth than ever before. The STI of the IBM eServer z900 is capable of transferring data between the processor complex and the I/O subsystem at a rate of up to 1 GB/s compared with the 333MB/s rate of the IBM S/390® G3/G4- and G5/G6-class large servers—an improvement of up to a factor of 3. Additionally, in the eServer z900, STI links suitable for transferring data at a rate of 1 GB/s are used for the third iteration of the integrated cluster bus (ICB-3), thus providing direct links to other eServer z900s. Also, the role of the STI has increased in the eServer z900, becoming the interface used on the I/O subsystem backplane. The majority of the STI logic (both physical and logical layers) has remained essentially unchanged, with advances resulting from the use of improved cables and connectors, and from the introduction of a multilevel, pre-distortion differential off-chip "super-driver."

Introduction

The self-timed interface (STI) has been used in IBM's largest servers for several generations, providing successively improved I/O subsystem bandwidth capacities. The eServer z900 contains the latest example in the evolution of the STI, which has the capability of moving data at 1 GB/s. Prior to the introduction of the eServer z900, the STI data rate topped out at 333 MB/s, and STI deployment was limited to interframe applications. In the eServer z900, STI use has been expanded to include the interface used on the backplane of the I/O subsystem. This paper describes the improvements that allow the STI to run up to a factor of 3 faster than in previous large servers.

Figure 1 is a block diagram of two eServer z900s showing STI usage. Primary STI links running at 1 GB/s provide the interface between the memory bus adapters (MBAs) in the processor nest and the MUX/DEMUX chips in the I/O subsystem. The MUX/DEMUXs further provide secondary STI links running at 333 or 500 MB/s on the I/O subsystem backplane, enabling legacy I/O such as ESCON* and intersystem channels (ISC-3) to be used for Parallel Sysplex* configurations. Through the common I/O platform, PCI connectivity is provided to the IBM Fibre

0018-8646/02/$5.00 © 2002 IBM

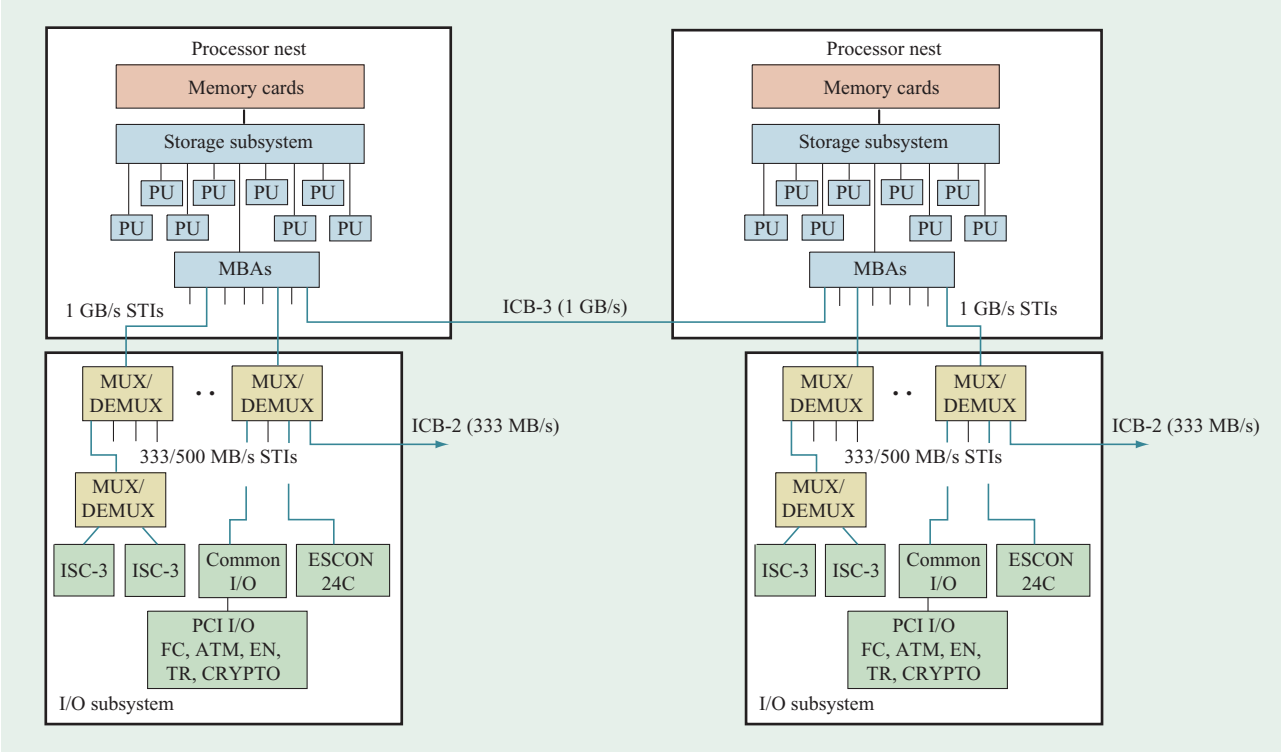

Figure 1

STI of IBM eServer z900.

Channel offering (FICON*), to network-based protocols such as ATM, Ethernet, and Token Ring, as well as to the IBM PCI cryptography processor [1]. New with the eServer z900 is the ability for direct 1GB/s communication links, using STI, between other eServer z900s via the integrated cluster bus (ICB-3) [2]. A secondary 333MB/s STI link may be configured for sysplex connectivity to previous-generation machines, such as S/390* G3/G4- and G5/G6-class servers, using ICB-2 [3].

The challenge confronting the STI designers was how to increase the speed from 333 MB/s to 1 GB/s while driving essentially the same distances and package structure as the previous machines. The signal transmission path would have to include an MCM or SCM, printed-circuit wiring on a card or board, a cable connector, and up to 10 meters of cable, with a similar arrangement at the receiving end of the cable. In some cases there would be additional connectors and cards in the signal path. Other than relatively straightforward improvements such as a reduction in circuit delays realized using the IBM CMOS SA-12[1] technology, the increase in speed was accomplished without appreciably affecting the STI logic design. Innovation and improvements in the cable, connectors, and I/O circuit design allowed the data pulse widths to be reduced from 3 ns to 1 ns.

The STI combines parallel and serialized data running at high speeds. This paper discusses the challenges encountered with the development of the 1GB/s STI physical link and its implementation. The topics presented include the electrical characteristics of the copper cable and the resulting consequences, the problem of jitter and noise on the link performance, the effects of silicon tolerances and the ability of the STI to compensate for skew among the data bits.

Review of the STI physical macro logic

As mentioned earlier, the STI of the eServer z900 was implemented preserving the majority of the physical macro (PM) logic. The PM logic is described at length in Reference [4]; however, a brief review can be helpful in understanding the remainder of the material presented here. STI transmits one byte of data, a combination parity/flag bit, and a half-speed clock signal in each direction. This requires 20 differential signal pairs (10 pairs in each direction) in the cable. The maximum STI

[1] The IBM SA-12 technology: 0.25-μm lithography with $L_{effective}$ 0.18 μm at 1.8 V nominal.

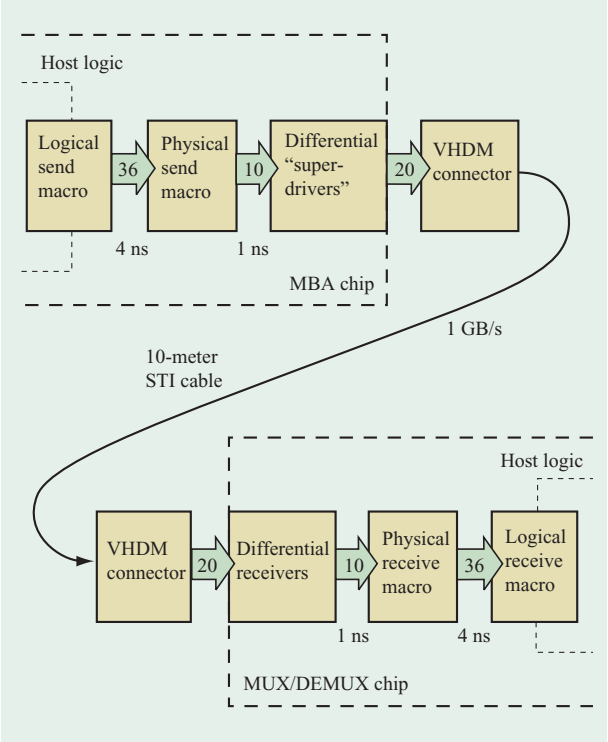

Figure 2

STI link components.

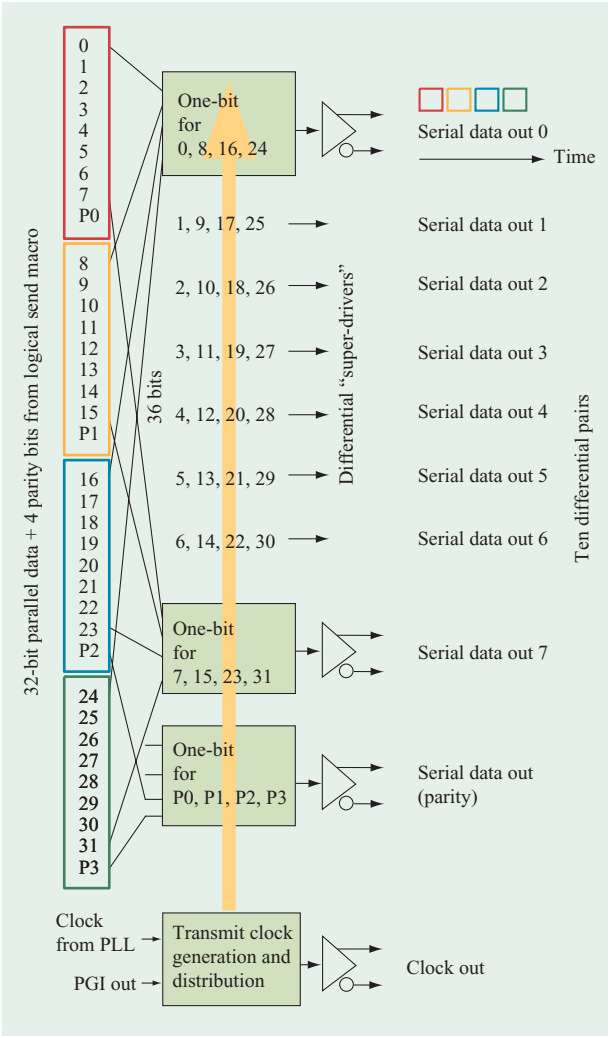

Figure 3

STI physical send macro (PSM).

data rate is 1 GB/s (2 GB/s total), producing 1-ns data pulse widths using a half-speed 500-MHz clock signal. A major problem with operating a copper link at these speeds is that the skew between data signals on the link can easily exceed the bit time.

The physical layer of the STI is partitioned into two semicustom macros, a physical send macro (PSM) and a physical receive macro (PRM). As shown in **Figure 2**, these entities interface with the off-chip differential drivers and receivers on the link side and with the STI logical layer macros [logical send macro (LSM) and logical receive macro (LRM), respectively] on the host side. The STI logical macros handle flow control and packet generation for the STI link and interface with the host chip logic. Additional information on the STI protocol layer can be found in Reference [1].

The primary function of the PSM is to serialize the word-wide data bus from the LSM down to a byte-wide bus for transmission onto the STI cable. A major advance in this implementation over its predecessors is the creation of the differential "super-drivers." More is said about the super-drivers later in this paper. **Figure 3** shows all of the entities that comprise the transmit function of the STI physical layer.

Again referring to Figure 2 on the receiver side of the STI link, the PRM receives the signals from the differential receivers, resynchronizes the data signals to the received clock, and finally deserializes the four bytes of data back into a word and presents the data to the LRM. Similarly, **Figure 4** shows all of the entities that comprise the receive function of the STI physical layer.

Skew aspects of data transmission

Consider the data transmission system shown in **Figure 5**. Data is launched by a clock from a set of latches on the transmit end of the link through a set of off-chip drivers (OCDs) into a set of conductors. As is the case with the STI, the launching clock is sent along with the data. At the receiving end of the conductors, a corresponding set

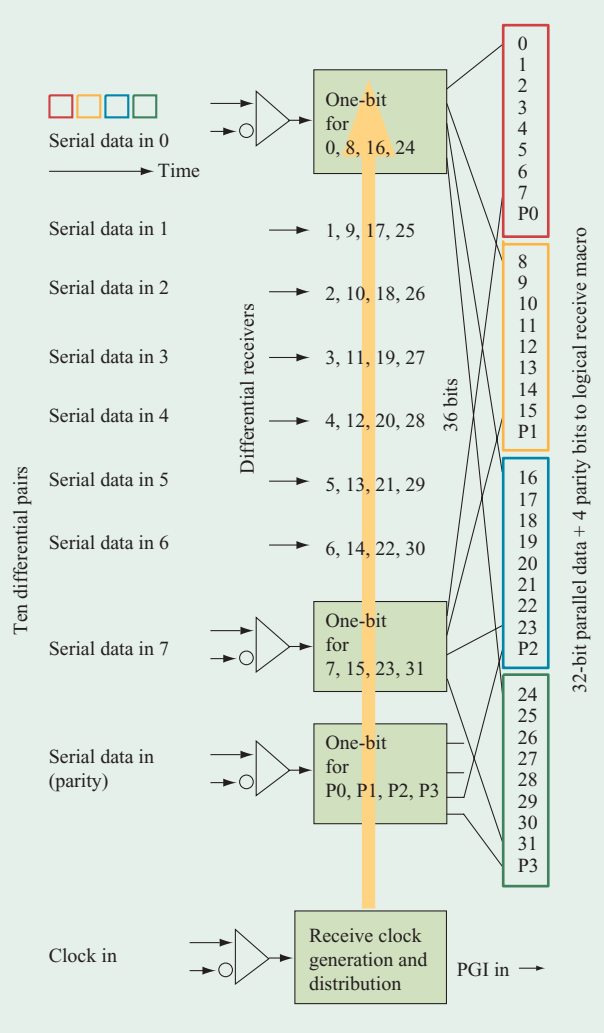

Figure 4

STI physical receive macro (PRM).

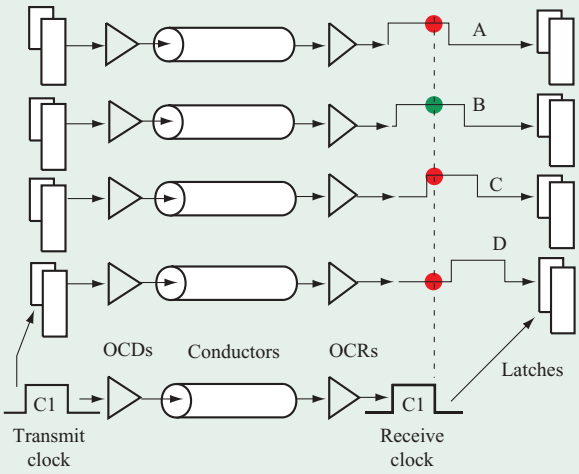

Figure 5

Skew in data transmission.

of off-chip receivers (OCRs) processes the incoming signals, which are then captured by receiving latches using the received clock. Although the data for each conductor is launched at the same time, the arrival times at the receiver are skewed because of the variations of the individual transmission paths. Figure 5 shows that only path B is optimally captured by the falling edge of the C1 clock. The other paths are captured earlier than optimal (path C), later than optimal (path A), or erroneously (path D).

The STI solution

The STI approach is straightforward. Each data input signal to the receiver chip is fed to a delay line with multiple taps, as shown in **Figure 6**. The delay line contains many identical delay elements, with the output of each delay element representing a unique phase of the incoming data signal. At the heart of the STI is the phase selection logic, which is responsible for locating the preferred phase of the data bit to be sampled, i.e., the phase whose center coincides with the capturing clock edge. Once identified, a servomechanism locks onto the preferred data phase, making dynamic adjustments to maintain the lock; hence the name "self-timed interface."

Central to the phase-selection process is centering the data-valid window with respect to the sampling clock edge. The edges of the data bit are found by means of the edge-detection process [4, 5]. As shown in Figure 6, the falling edge of clock C1 aligns with the trailing edge of the data window and identifies the corresponding tap as tap E, or early guard band (EGB). In a similar way, the falling edge of C1 aligns with the leading edge of the data window, which occurs at tap L, or late guard band (LGB). Finding the center of the data window becomes a simple matter of choosing a tap midway between tap E and tap L, which is tap D. Thus, the falling edge of clock C1 falls directly in the center of the data window. This process is known as "bit synchronization."

Figure 7 shows a more detailed block diagram of the STI PRM one-bit logic, with particular attention paid to the phase-selection logic. The serial data is fed into the bulk-delay line and then into the fine-delay line. The serial data is transmitted using double data rate (DDR), which

means that two data bits are sent and received each clock cycle, i.e., one bit on the rising edge of the clock and one bit on the falling edge. The fine-delay line has 32 output taps, which are divided into three groups. The first group comprises delay element 0 through delay element 15 and is assigned to the EGB. The second group, comprising delay element 16 to delay element 31, is assigned to the LGB. The third group comprises delay element 8 through delay element 23 and is assigned to the data tap and shared with the upper half of the EGB and the lower half of the LGB. An adjacent pair of phases is selected from each of the three groups, yielding 15 possible pairs formed by the 16 delay elements in the EGB, LGB, and data groups. The three groups of delay-element outputs are fed to three pair-selector blocks. The EGB pair selector on the left of Figure 7 chooses the EGB tap, or tap E, which is controlled by the EGB address register. The LGB pair selector on the right chooses the LGB tap, or tap L, which is controlled by the LGB address register.

First consider the EGB selection on left side of Figure 7. When the STI logic is initialized, the EGB pair selector selects a pair of predetermined phases near the upper part of the delay-element group (e.g., outputs of element 11 and of element 12). The selected phase pair goes to the EGB sample logic. The EGB sample logic uses L2-STAR latches [4] to capture data on both the rising and falling edges of the clock. Thus, DDR data is converted into two full-cycle data samples. The EGB sample block also serves as a serial-to-parallel converter. Hence, four edge samples are used by the EGB edge detector along with two data samples to determine whether the selected edge pair lies on the data-bit edge. The edge-detector logic generates an *up* or *down* signal to either increment or decrement the EGB address register. The random-walk filter (RWF) is used to filter the edge-detector outputs, thus avoiding instantaneous adjustment in favor of adjustment based on the trend over many cycles. The LGB loop on the right side of Figure 7 operates in a similar manner, with the exception that the LGB pair selector first picks a pair near the lower part of the delay-element group (e.g., outputs of element 18 and element 19). One can see that the initially selected EGB and LGB are pulled toward the middle of the delay line and work their way toward the two ends of the fine-delay line. The bulk-delay line is used to coarsely place the entire data window in the middle of the fine-delay line.

The output of the EGB address register and the output of the LGB address register are fed into the data address register, where the average values of the tap E address and the tap L address are calculated. The result is used by the data-pair selector in the middle of Figure 7 to select a pair of data phases. One of the data-phase pair is selected, and the data-sample logic converts it, again using L2-STAR latches, into a pair of full-cycle data samples,

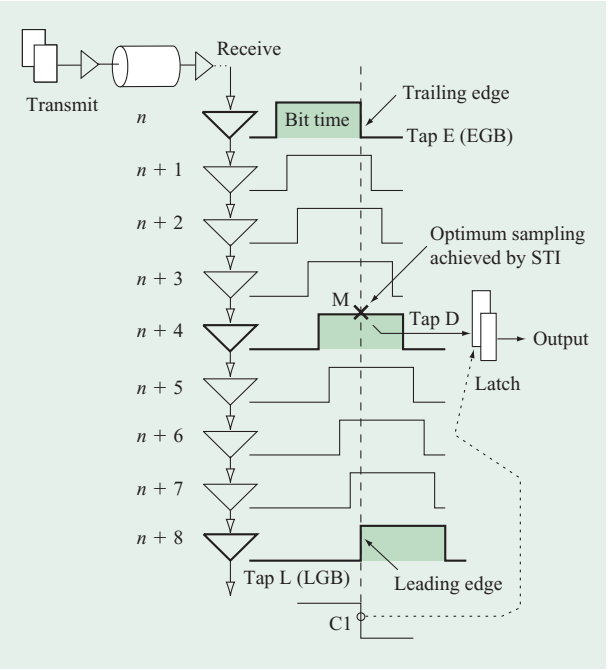

Figure 6

The STI solution generates many phases and samples the center tap between the edges. This is "bit synchronization."

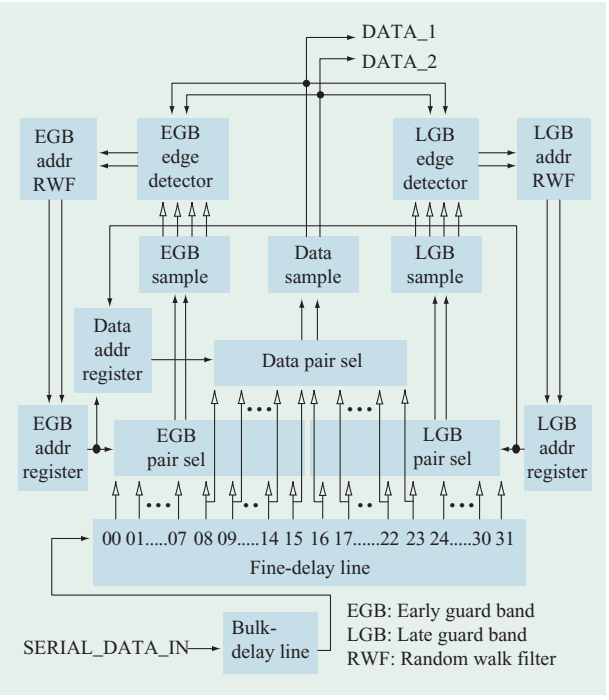

Figure 7

STI PRM one-bit logic.

Table 1 IBM eServer z900 1GB/s STI link jitter budget in picoseconds.

Data-pattern-dependent jitter		192
Terminator (RMS)	38	
Pre-distortion ratio (RMS)	19	
Delay-line resolution		160
Worst case (RMS)	47	
Tracking (RMS)	38	
Delay-line jitter		93
PLL jitter		100
Transmit data duty cycle		25
Worst case (RMS)	25	
Transmit clock duty cycle		25
Worst case (RMS)	25	
Coupled-noise clock jitter		68
Worst case (RMS)	68	
Coupled-noise data jitter		68
Worst case (RMS)	68	
Sum (mean)		731
Sum (RMS)	126	
Overall sum (mean + RMS)		857

DATA_1 and DATA_2. These two data signals are further deserialized into four parallel data signals by the downstream logic. Additionally, the downstream logic achieves "byte synchronization" and ultimately "word synchronization" before presenting the data to the LRM. Again, please refer to Reference [4] for more details.

Jitter aspects of data transmission

The discussion thus far has focused on how the STI de-skews the received data; however, the fundamental concern with the STI link is error-free data transmission. The presence of jitter on the link is what causes the data to be sampled erroneously. There is a distinct difference between data-bit skew and jitter. Although both are caused by similar sources, e.g., differences in the chip tolerances and transmission paths, jitter results from the time-varying component of these sources and applies only to an individual data-bit path. Jitter results primarily from noise sources causing variations in chip delay, and also from variations in transmission-line delay.

Jitter budget

The speed of the link cannot be arbitrarily increased before various physical and design limitations of the link contribute to bit errors. With STI, the primary measure of system performance is the jitter budget. **Table 1** shows the jitter budget used for the eServer z900 1GB/s STI link. Simply, jitter is the cycle-to-cycle variation in the arrival time of the bits relative to the data-sample clock. If the jitter is large enough to move the arrival time of a bit into an adjacent bit time, a bit error will occur. The objective is completely error-free operation of the link. Numerous items contribute to jitter. No single item is large enough to cause an error; rather, a combination of jitter sources causes a bit error. Consequently, most of the items in the jitter budget had to be reduced in order to achieve the desired link performance at the 1Gb/s speed. Since the bit time for the link was reduced, the jitter for the link also had to be reduced. What follows is a brief explanation of the jitter budget and its components. See Reference [4] for a more detailed explanation of the jitter budget.

As stated previously, if the bit is perturbed too far in time relative to the clock, a data-sampling error results. The jitter budget is an attempt to categorize and enumerate the various sources of jitter that can perturb the bit or clock arrival time. As a matter of convention, peak-to-peak numbers are used in the budget; consequently, the combination of the numbers must exceed the bit time in order for an error to occur. The goal is to achieve error-free operation, so a conservative approach is used to combine the various components of the budget. If an item occurs with relative frequency, the nominal peak-to-peak value of the component is summed directly with the other components that occur frequently. Pattern-dependent jitter is an example of such an item, which can cause frequent peak perturbations to the bit arrival; hence, the item is summed. Another example would be duty-cycle variations, in which every bit is affected by the duty cycle of the clock. Finally, the statistical variations or process variations of the items are combined using the root mean square (RMS), and that result is combined with the directly summed items. As can be seen from Table 1, the total from the jitter budget is 857 ps, which is less than the 1-ns bit time.

Referring to Table 1, data-pattern-dependent jitter is caused by frequency and phase distortion of the transmission medium, which in turn causes the arrival time of the data bits to be dependent on previous data content. Variations in terminator impedance and the pre-distortion ratio of the driver contribute to the statistical variation of the pattern-dependent jitter. This is discussed later in more detail. Since the STI uses different taps on the fine-delay line to locate the data-bit edge boundaries and also to select the center tap from which the data bit is sampled, the delay between taps determines the resolution of the delay line. This contributes to the inaccuracy of locating the absolute center of the data bit. The number used in the jitter budget is twice the delay between taps, which has to do with the way in which the STI resolves and determines the bit center.

Another aspect to consider is the time delay of the delay elements as a function of the power-supply voltage, since the power supply is usually increased or decreased

by noise on the chip. Hence, the on-chip noise can vary the arrival time of the data bit and contribute to the data jitter. The time that is affected is the difference between the arrival of the data at the capturing latch input and the delay due to the distribution of the received clock. This time difference can approach two bit times. It is assumed that the data delay and the clock distribution experience the same mid-frequency noise. It is this mid-frequency noise that can do the damage. Generally, the mid-frequency supply noise that has the most effect on the time delay of the clock distribution has a period that can equal or be of the order of several times the period of the STI clock. The period of the noise must be sufficient to affect the clock, but insufficient to prevent the STI from compensating for the delay modulation of the clock distribution.

The STI send clock involves a PLL and a clock-distribution network. The PLL itself has intrinsic jitter. Also, the send-clock distribution has delay and is subject to delay modulation from noise on the power supply, as is the case for the PRM delay line. The clock distribution tree constitutes a delay line for all practical purposes. These components constitute the PLL jitter entry in the budget. Further, it is the cycle-to-cycle jitter that is of interest. Usually the clock cycle that sent the data and the clock cycle that captures the data differ by no more than a few cycles. This leads to another interesting phenomenon brought about by the way STI works. Since the clock cycle that transmitted the data may be different from the one used to capture the data, the data bit and the capture clock are uncorrelated, and the PLL jitter must be doubled as a consequence.

Because STI data is DDR, the duty cycle (the up time versus the down time) of the data and clock are extremely important. The duty cycle can be affected by both the send side and the receive side of the link. Since the asymmetry error occurs every cycle, the duty cycle is added directly into the budget. The number used in the jitter budget is the difference between the up time and the down time of the clock. Finally, coupled noise from the package can also cause jitter. The coupled noise can cause the signals to be displaced in time, resulting in jitter. Coupling can occur on the card, on the module and chip substrate, mostly at the vias and connectors. Again, both the data and clock are affected.

Some of the items in the jitter budget will be reduced simply because of the higher speed. The resolution of the delay line, for example, will naturally be reduced, since the delay elements must have reduced propagation delay to run at the higher speed. Other items such as duty cycle, clock jitter, and coupled noise must be reduced through increased attention to detail during design. Duty cycle is reduced by more careful balancing of the stages in the clock tree, while clock noise is reduced by a reduction in the overall delay of the clock tree and careful isolation of the clock tree to prevent on-chip coupled noise. Reduced delay in the STI clock tree reduces the effect that on-chip mid-frequency noise can have on delay time of the clock trees. Variations in the delay time contribute to clock jitter. Decoupling capacitance on the chip and the chip substrate also help to further reduce the clock noise. Finally, coupled noise in the card and board environment must also be reduced. The coupled noise is controlled by careful wiring of the card and board. Wiring rules that isolate the differential pairs from one another are implemented to reduce the coupling from one pair to another. One card/board wiring ground rule requires the use of a blank channel between pairs to reduce coupling. Further, the MCM and SCM wiring are guided by still other wiring rules to minimize the pair-to-pair coupling.

Perhaps the last source of coupling is the card and cable connectors. The SCSI connectors used for the previous 333MB/s STI design were a major source of coupling. Their use at 1GB/s data rates would be completely unacceptable. A study of the various connectors available from vendors resulted in the selection of the Very High Density Metric (VHDM**) connector from Teradyne as an acceptable solution. This connector considerably reduces coupling and is used for both card and cable connectors throughout the eServer z900 machine.

The only jitter budget term that does not lend itself to improvement through improved wiring or design is pattern-dependent jitter. Basically, pattern-dependent jitter is the variation in the arrival times of the data due to distortion of the transmission medium—in this case, copper cable and card wire. The primary cause of the distortion losses at high frequency is the well-known skin-effect loss of the card and cable wire. Skin effect causes both frequency and phase distortion. The higher the frequency, the greater the losses and distortion become. Pattern-dependent jitter that was acceptable at 333MB/s speeds is totally unacceptable at 1 GB/s over the same link. Not only has pattern-dependent jitter increased because of higher losses, the bit time has also decreased, allowing less margin to sample the data bit.

A number of approaches can be taken to reduce skin-effect losses. The most straightforward action to reduce skin effect is to increase the diameter of the cable conductors and to increase the width of the card wire. To this effect, the cable conductor diameter was increased as much as possible while retaining a cable with manageable physical dimensions. If the overall cable bundle diameter becomes too large, it becomes difficult to plug and route the cable. The cable was changed from 28-gauge twinax to 26-gauge twinax. Unfortunately, increasing the card wire width was out of the question. This would have required the cards and boards to become thicker to maintain the desired impedance, and this was unacceptable. Further,

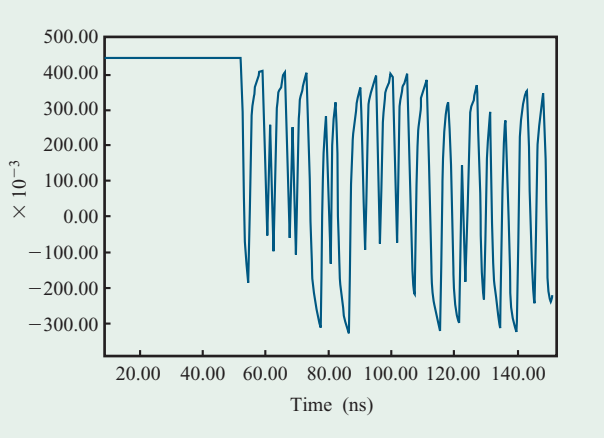

Figure 8

Differential input to receiver with no pre-distortion and random data.

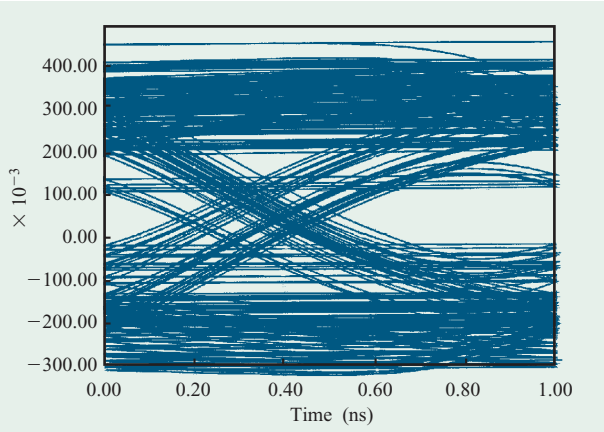

Figure 9

Peak-to-peak jitter for differential input to receiver with no pre-distortion, random data.

wiring density would be affected, and this too was unacceptable.

Figure 8 shows a simulated differential input to the receiver given a random data sequence after traversing the maximum board length and the maximum cable length of 10 meters. Note that some of the signal transitions barely make the differential threshold of 0 volts. With a differential signal, a positive voltage is taken as a "1," while a negative voltage is a "0." **Figure 9** shows the simulated peak-to-peak jitter for the same set of conditions. The jitter is so large that it does not allow enough room for the other items in the jitter budget, and it is unlikely that the link will work. Therefore, more must be done to further reduce the pattern-dependent jitter.

One might think that it would be possible to shape the frequency response of the channel with a filter and basically compensate, to a limited extent, for the high-frequency losses and phase distortion. There are two ways to accomplish this. The first approach employs a filter composed of discrete components, called an equalizer. In this case, specific signal qualities are emphasized to match the characteristics of the transmission channel. This works quite well in most cases and is usually the recommended way to go if the tight space requirements for the passive components can be met. In general, however, the space required for discrete components is not available on MCM and SCM carriers. Moreover, there is currently no means for satisfactorily performing the filter function in CMOS at gigabit speeds. An alternative solution to decrease the jitter using CMOS was necessary.

The second way to shape the frequency response of the channel is to introduce distortion on the drive signal at the transmitter so as to compensate for signal degradation in the transmission channel. This approach pre-distorts the drive signal to enhance those frequency components of interest which are usually corrupted by the transmission media, hence opening the eye at the receiver. Pre-distorting the drive signal is relatively simple to accomplish and is dependent on active components, thus making it very appealing in a CMOS environment. The amount of distortion needed is strongly dependent on the transmission system and must be calculated through modeling or empirically obtained. The result is a ratio between the pre-distorted pulse amplitude and the rest of the pulse train.

The approach that was finally adopted was to incorporate the filter function into the driver. This is done by developing a driver that has multiple output levels whereby the output level of the driver is dependent on the previous data transmitted. In this fashion, the high-frequency content of the data signal can be boosted. The function of the driver becomes very much like that of a finite impulse response filter, where the output of the filter is dependent on the current and previous data samples. Instead of placing the filter at the receiver, it is incorporated into the driver function.

Because of the way the driver functions, discussed shortly, a finite number of levels are selected. It was experimentally determined that three levels for a "1" and three levels for a "0" would be sufficient to produce the desired jitter reduction. Adding additional levels yields decreasing benefits while increasing the driver complexity. The algorithm developed for the driver is very simple. If the present data bit is different from the previous data bit, the level is set to maximum for the present data bit. This is called the high (H) level. If the current data bit remains

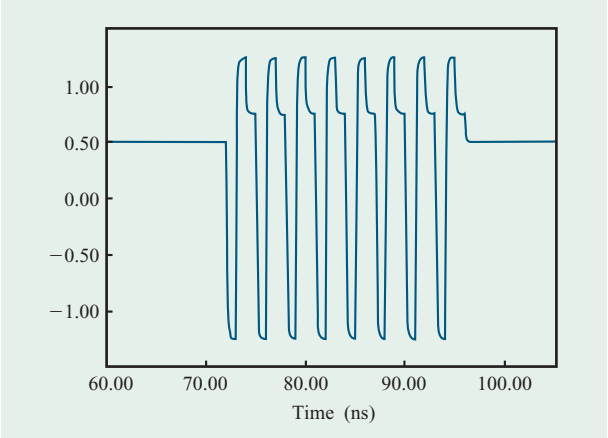

Figure 10

Three-level pre-distortion driver output for the sequence ...1111101101101101101101101111.

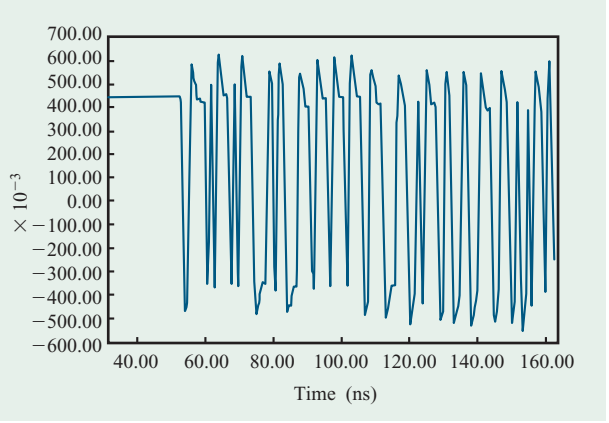

Figure 11

Pre-distortion for differential input to receiver with random data.

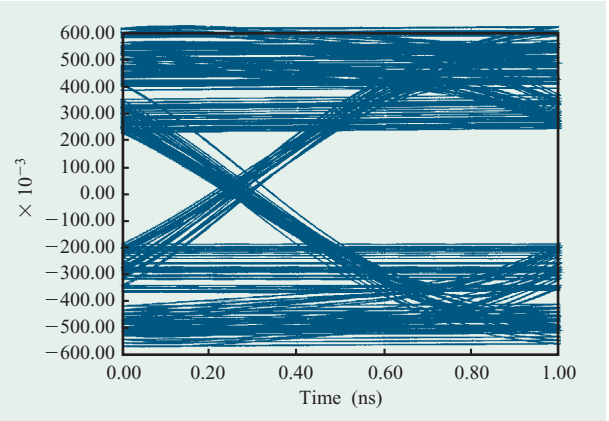

Figure 12

Peak-to-peak jitter for pre-distortion, differential input to receiver with random data.

at the previous data bit after a transition, the level is set to an intermediate level called the medium (M) level. Finally, if the present data bit stays at the previous level for a third bit interval, the level is reduced to a low (L) level, where it remains until a change occurs in the data-bit value. **Figure 10** illustrates the algorithm for the . . . 1111101101101101101101101111 . . . sequence of data bits.

In terms of jitter reduction, the ratios of the levels are important. Through experimentation it was determined that a nominal high-to-low (H/L) ratio of 2.5:1 and a medium-to-low (M/L) ratio of 1.5:1 was optimal for jitter reduction. It should be emphasized that minimizing jitter was the primary criterion for selecting the ratios. Other techniques for selecting the levels may not necessarily minimize the jitter. Also, constraining the output for each phase to three levels puts further restrictions on the optimization process. Consequently, the ratios were varied for a random data sequence until the jitter reached a satisfactory level for the maximum package lengths. **Figure 11** is the simulated differential input to the receiver for the same data sequence and package that was used to create Figure 8. **Figure 12** shows the simulated peak-to-peak jitter diagram for the same received data sequence. Figure 8 and Figure 9 should be compared to Figure 11 and Figure 12 to see the dramatic improvement brought about by pre-distortion.

One final comment about pre-distortion is in order. Even though the ratios have been optimized for the maximum loss and package length, the jitter does not increase for less loss or shorter package lengths. While it is true that jitter may be decreased if a smaller ratio is used for less loss, the jitter never exceeds the maximum-loss case. Other techniques such as using a fixed equalizer filter will actually degrade the jitter, potentially to the point of failure, at the shorter package lengths. This is often solved by making the equalizer "adaptive," i.e., able to change frequency response for different package lengths; however, this is far more complicated than a fixed driver pre-distortion scheme. The fixed pre-distortion scheme described achieved the necessary performance for the 1GB/s STI link.

Pre-distortion driver, or "super-driver"

The 1GB/s STI uses differential signaling on the cable interface with a common-mode voltage of 0.75 V and a 1-V swing about the common mode. A multilevel signaling

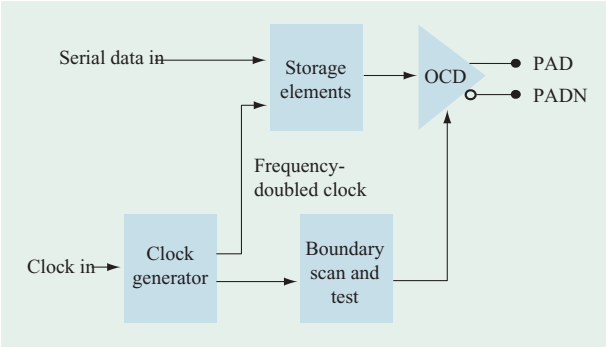

Figure 13

Super-driver block diagram.

scheme has been employed to accomplish the pre-distortion.

To apply the pre-distortion, it is necessary to maintain a history of the data bits previously transmitted by the driver. A pre-distortion algorithm then requires storage elements to be included in the driver. The degree of pre-distortion dictates the number of storage elements needed and determines how much history must be retained. As has been mentioned, three levels of pulse pre-distortion are needed to achieve the desired reduction in jitter. Since differential signaling is employed, it is also required to store the opposite phase of these signals as well. Hence, six latches are used in the driver to keep track of data history for both phases. This function and complexity added to the traditional OCD led to the name "super-driver."

As mentioned previously, the PSM function was unchanged from previous STI designs, in which the PSM produced serial data streams using a half-speed clock which in turn were converted directly into differential serial data streams by the OCDs. Since the super-driver contains latches, the necessary clocking, test, and boundary-scan functions also had to be included. **Figure 13** is a high-level block diagram of the super-driver. Since information must be sent over a maximum of ten meters of cable plus package and connectors, the output driver stage must have sufficient capability. This results in large capacitive loads to the high-speed circuits within the super-driver.

The wiring from the PSM to the super-driver is predetermined so that critical signals are wired in close proximity to one another. Careful attention must be paid to the path of both the half-speed clock and serial data from the physical send macro. Precise alignment must be maintained in order to keep skew and the resulting jitter at a minimum. It is essential that the deviation from a 50% duty cycle of the clock be minimized, since any deviation produces jitter. Further, in order for the super-driver to capture the serial data stream from the PSM using a half-speed clock, the clock must be doubled in frequency to create a full-speed 1-GHz clock.

Storage elements

The storage elements are LSSD-compatible L1/L2 latches that must be capable of driving large capacitive loads. Physical placement of the latches is critical and must be arranged so that dataflow is not affected by wirability. The latches are passgate designs for fast setup time, minimum skew, and maximum speed. These latches also minimize clock-to-data jitter. The latch output rise and fall transitions must be maintained over the range of process variation. Therefore, carefully designed buffer stages are required so that pulse integrity is not lost keeping skew and jitter to a minimum. The data will have narrow pulse widths requiring tight timing margins.

Figure 14 depicts the section of the super-driver that stores the bit-stream history information. The input serial data is exclusive-ORed with the power supplies, ground, and V_{dd} to generate both in-phase and out-of-phase data streams, respectively. Data is shifted in parallel down the register file. Each latch L2 output is sent to the six-input driver stage through buffers for processing. The current bit, *a0*, plus the previous two bits, *a1* and *a2*, and their complemented versions, *an0*, *an1*, and *an2*, are processed. The buffers are designed to minimize pulse-width distortion and are designed specifically for the driver stage inputs. Physical placement of the buffers is bit-stacked to maintain signal integrity.

Clock generator

It is important that the latch B and C clocks be generated within the super-driver, since very narrow critical pulse widths must be maintained. This also avoids transporting high-speed signals around the chip. To generate the necessary clocks, precise control of both edges of the half-speed clock is required for frequency doubling. Any asymmetry results in a jitter component at the output of the OCD.

The clock generator, depicted in **Figure 15**, is another critical element of the super-driver design. The frequency-doubler design begins with standard SA-12 library components, which are then optimized for performance. The half-speed PSM clock is fed to a pair of NAND gates, which produce a delayed and a nondelayed version of the clock. The amount of delay dictates the pulse width of the resulting clock-splitter oscillator input. The delayed version is exclusive-ORed with the nondelayed version, thus producing a stream of very narrow pulses, each about

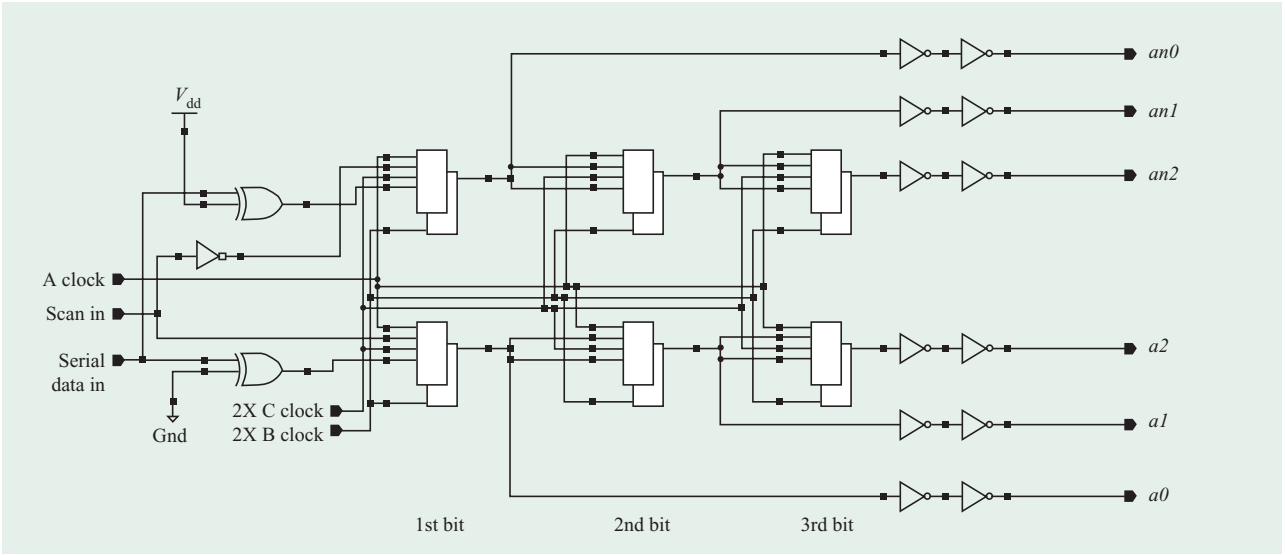

Figure 14

Super-driver storage elements.

250 ps wide, from both the rising and falling edges of the half-speed clock. These pulses are then used to generate the latch B and C clocks using a clock splitter. The result is a frequency-doubled clock. The high-speed signal processing is done entirely within the super-driver.

Off-chip differential driver

The OCD stage, shown in **Figure 16**, is where the final processing takes place. The three levels of pre-distortion are predetermined by empirical data, and the ratios are set by a gain function described below. The data components *a0*, *an1*, and *an2* modulate the gain proportionally in conjunction with feedback controls to generate the prescribed levels. Therefore, the total gain of the output stage is expressed as a Boolean function, A, where $A = a0[w1 + (an1)w2 + (an1 + an2)w3]$. Here, constants *w1*, *w2*, and *w3* are weighting functions prescribed by the pre-distortion ratios. The amplitude is controlled by feedback which provides a mechanism to throttle back on the gate drive of the output devices. This is shown in Figure 16, where signals *a0*, *an1*, and *an2* logically interact with the feedback. A critical parameter is the ratio of device widths of the feedback to the pre-drive devices. The current data bit, *a0*, and the next bit in the sequence are combined logically to switch in the feedback. The current bit presets the condition so that if the next bit in the sequence is of the same polarity, the feedback is switched on. Otherwise it sets up for the opposite phase. The NAND function is used primarily for timing control,

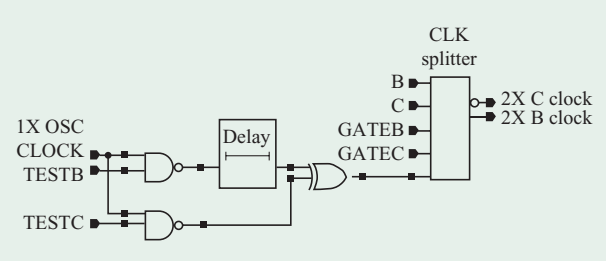

Figure 15

Super-driver clock generator.

since this design is not symmetric and up-level amplitudes are always controlled by the feedback. Otherwise, in a symmetric design, *a1* would interact with *a0* for positive amplitude control. This works quite well and is used in all of the self-timed interface driver circuits [6]. Shown in Figure 16 is the pre-drive stage, which controls the gain on the output drive devices. Essentially, these are specially designed NAND and NOR gates. Unlike standard gates, they effect gain changes based on the input data stream. These circuits have multiple inputs for a single bit. In effect, the next bits in the stream are used to preset conditions within the circuit. Therefore, if the current bit is high-level and the previous bits were of opposite phases, the current bit would have the greatest weighting value. It also sets high, so that if the next bit is of the same

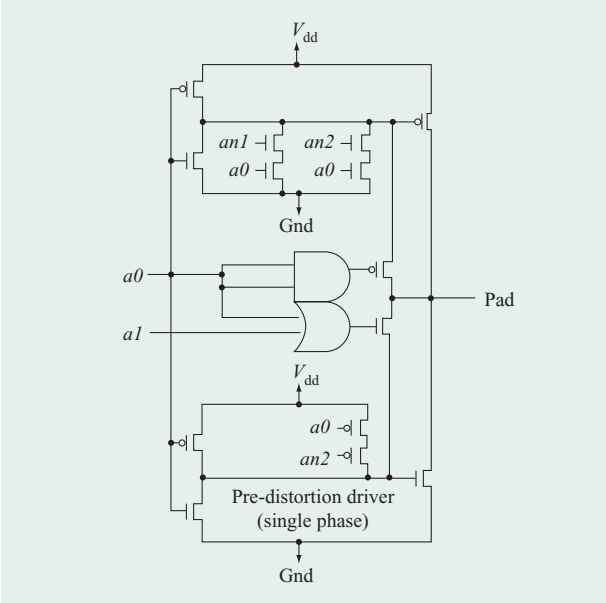

Figure 16

Super-driver OCD.

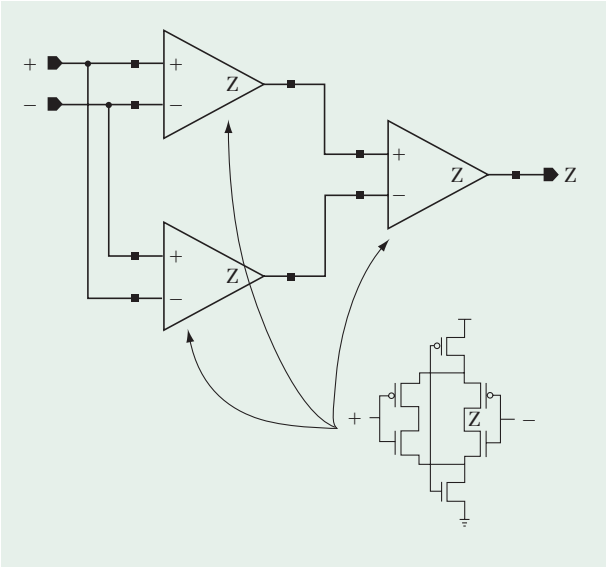

Figure 17

Differential triple-cascade receiver.

polarity, it turns on the next leg in the circuit. Also, a feedback path is turned on, causing the gate drive to be throttled back. The overall effect of the described feedback is to control the signal amplitude. In a similar manner, the same concepts are used for the opposite phase, thus producing differential outputs.

Off-chip differential receiver

The data receiver is a triple-cascade self-biasing design employing three separate cascaded differential operational amplifiers. As shown in **Figure 17**, each operational amplifier is a folded p-to-n-type receiver which provides good common-mode range. The tri-amp configuration enhances common-mode range to the power-supply "rails." The primary cost factor is latency, which is not a major problem in STI. The receiver circuit provides a combination of differential termination with a common-mode terminator at the center tap. The total gain of each amp is approximately ten. This allows for an overall robust receiver for low-level signals on ten meters of cable.

The STI clock receiver has an additional control function to detect when no cable is attached to the port. This is called the power-good indicator, or PGI. When it detects the absence of an input signal to the receiver, it shuts down the clock receiver output. It also provides a signal to quiesce the rest of the STI logic. The PGI detector remains active so that when an input signal is detected, the system may be brought back up. This is a widely used feature in the STI; it is used to start or reset the STI physical and logical protocol layers.

The STI drivers and receivers have the ability to perform an interconnect test. In wire-test mode, the drivers send a dc signal to the receivers via the PAD output. If continuity exists, the signal propagates down the cable and through the differential receiver terminator, and returns to the PADN output of the driver. There the signal is compared against a threshold verifying continuity. This test covers the entire physical net including chip, module, card, connectors, and the differential cable. This is especially useful in large configurations, where it becomes practically impossible to locate a fault in the interconnect.

Summary

The STI links of the eServer z900 achieve a factor of 3 increase in link bandwidth over the links used in previous IBM S/390 G5/G6-class large servers. To realize this increase, a jitter budget was created whereby the various link jitter components were analyzed and optimized until satisfactory jitter performance could be achieved. The results of this work required that the STI cable diameter be increased and the STI cable connectors be upgraded. Finally, to address the unacceptably large pattern-dependent jitter component, a three-level pre-distortion off-chip super-driver was developed. In addition to the increase in link bandwidth, the I/O subsystem infrastructure in the eServer z900 is now STI-based, with either a 333

or a 500MB/s interface supporting each slot on the I/O backplane. As the evolution of the IBM eServers continues, the STI is expected to continue to play an important role.

Acknowledgments

The authors wish to thank Daniel Casper and Jack Yarolin for their helpful comments regarding the manuscript. We also wish to acknowledge the original work done by Frank Ferraiolo and Daniel Casper on the STI. Also contributing to the advancement of the STI were Richard Jordan, Anthony Perri, Mark Fischer, and Jack Yarolin.

References

1. D. J. Stigliani, Jr., T. E. Bubb, D. F. Casper, J. H. Chin, S. G. Glassen, J. M. Hoke, V. A. Minassian, J. H. Quick, and C. H. Whitehead, "IBM eServer z900 I/O Subsystem," *IBM J. Res. & Dev.* **46,** No. 4/5, 421–445 (2002, this issue).
2. T. A. Gregg and R. K. Errickson, "Coupling I/O Channels for the IBM eServer z900: Reengineering Required," *IBM J. Res. & Dev.* **46,** No. 4/5, 461–474 (2002, this issue).
3. T. A. Gregg, K. M. Pandey, and R. K. Errickson, "The Integrated Cluster Bus for the IBM S/390 Parallel Sysplex," *IBM J. Res. & Dev.* **43,** No. 5/6, 795–806 (September/November 1999).
4. J. M. Hoke, P. W. Bond, T. Lo, F. S. Pidala, and G. Steinbrueck, "Self-Timed Interface for S/390 I/O Subsystem Interconnection," *IBM J. Res. & Dev.* **43,** No. 5/6, 829–846 (September/November 1999).
5. R. C. Jordan, R. S. Capowski, D. F. Casper, F. D. Ferraiolo, W. C. Laviola, and P. R. Tomaszewski, "Edge Detector," U.S. Patent 5,577,078, November 19, 1996.
6. R. R. Livolsi, "ECL Compatible CMOS Off-Chip Driver Using Feedback to Set Output Levels," U.S. Patent 5,280,204, January 18, 1994.

Received November 26, 2001; accepted for publication January 21, 2002

Joseph M. Hoke *IBM Server Group, 2455 South Road, Poughkeepsie, New York 12601 (jmhoke@us.ibm.com).* Mr. Hoke is an Advisory Engineer in the eServer I/O Hardware Development group. He received a B.S. degree in electrical engineering from the University of Illinois at Chicago in 1987 and continued his studies under a university fellowship, receiving an M.S. degree in electrical engineering from Northwestern University in 1989. He joined IBM at Poughkeepsie, New York, in 1989 and has held various technical positions in the eServer I/O area. Mr. Hoke holds several patents used in the IBM ESCON and Sysplex products and has received two IBM Invention Achievement Awards. He has received IBM Outstanding Technical Achievement Awards for his work on ESCON, on the G5 server, and for his contributions to the IBM eServer zSeries.

Paul W. Bond *IBM Server Group, 2455 South Road, Poughkeepsie, New York 12601 (pwbond@us.ibm.com).* Mr. Bond is an Advisory Engineer in the eServer I/O Hardware Development group. He received a B.S. degree in 1972 and an M.E. degree in 1973, both in electrical engineering from Rensselaer Polytechnic Institute. He joined IBM in Kingston, New York, in 1973 and is currently involved with the development of high-speed CMOS serial links.

Robert R. Livolsi *IBM Server Group, 2455 South Road, Poughkeepsie, New York 12601 (livolsi@us.ibm.com).* Mr. Livolsi is a Senior Engineer at IBM. In 1978, he received his bachelor's and master's degrees from the Polytechnic Institute of New York. In 1978, he joined the Magnetic Recording Group at AT&T Bell Laboratories in Whippany, New Jersey, where he worked on signal processing for imbedded servo and phase-locked loops for high-capacity tape drives. In 1984 he joined IBM at Kingston, New York, where he worked on the design of high-speed CMOS I/O circuits. He designed and patented the differential drivers and receivers used in the current STI design. Mr. Livolsi coauthored a paper on STI in the proceedings of the IEEE workshop on signal propagation on interconnects. He also holds an IBM second-level patent plateau award as well as IBM Outstanding Achievement and Innovation Awards.

Tin-chee (T. C.) Lo *IBM Server Group, 2455 South Road, Poughkeepsie, New York 12601 (tclo@us.ibm.com).* Mr. Lo received an M.S. degree in electrical engineering from Carnegie Mellon University. Prior to joining IBM, he worked for American Micro-System and Fairchild Semiconductor, both in northern California. Mr. Lo joined IBM at East Fishkill, New York, in 1977 to work on DRAM development projects. He became a Senior Engineer in 1984 and moved to IBM Poughkeepsie in 1985 to work on the design of the IBM S/390. Mr. Lo has worked in different technical areas including bipolar and MOS device modeling, n-MOS and CMOS circuit design, static and dynamic memories, ABIST, high-speed interconnects and signal propagation, and various logic designs for high-end servers. His current interests are in Level-2 cache design in the system-on-a-chip environment. Mr. Lo holds 16 U.S. patents, with another nine patents pending; he has published numerous papers and invention disclosures in several areas in the fields of microelectronics and logic design. He has received many technical awards during his career in IBM.

Frank S. Pidala *IBM Server Group, 2455 South Road, Poughkeepsie, New York 12601 (pidala@us.ibm.com).* Mr. Pidala is a Staff Programmer Analyst in the eServer I/O Hardware Development group. He joined IBM in East Fishkill, New York, in 1969. He is currently responsible for the physical design and release of self-timed interface (STI) macros. Mr. Pidala has received various IBM recognition and informal awards, including an IBM Outstanding Technical Achievement Award for his work on the G5 server.

Gary Steinbrueck *IBM Server Group, 2455 South Road, Poughkeepsie, New York 12601 (steinbru@vnet.ibm.com).* Mr. Steinbrueck is a Senior Engineer in the eServer I/O Hardware Development group. He received a B.S. degree in electrical engineering from the University of Missouri at Rolla in 1968. He joined IBM at East Fishkill, New York, and has held a wide variety of technical positions including thermal engineering; advanced development of power devices, CMOS and bipolar memories, microprocessors, and packaging; and product development of bipolar and CMOS logic circuits. Mr. Steinbrueck is currently responsible for circuit design for high-performance communication links for S/390, power parallel systems, and OEM products. He has received many technical awards, including IBM Outstanding Technical Achievement Awards for his contributions to the S/390 G3 and G5 servers.