Performance Driven Reliable Link Design for Networks on Chips

Rutuparna Ramesh Tamhankar
SUN Microsystems Inc
Sunnyvale, CA-94085
rutu@sun.com

Srinivasan Murali
CSL, Stanford University
Stanford, CA-94305, USA
smurali@stanford.edu

Giovanni De Micheli
CSL, Stanford University
Stanford, CA-94305, USA
nanni@stanford.edu

Abstract—With decreasing feature size of transistors, the interconnect wire delay is becoming a major bottleneck in current *Systems on Chips (SoCs)*. Another effect of shrinking feature size is that the wires are becoming unreliable as they are increasingly susceptible to various noise sources such as cross-talk, coupling noise, soft errors etc. Increasing importance of wire delay and reliability has lead to a communication centric design approach, *Networks on Chip (NoC)*, for building complex SoCs. Current NoC communication design methodologies are based on conservative design approaches and consider worst case operating conditions for link design, resulting in large latency penalty for data transmission. In order to substantially decrease the link delay and thereby increase system performance an aggressive design approach is needed. In this work we present Terror, timing error tolerant communication system, for aggressively designing the links of NoCs. In our methodology, instead of avoiding timing errors by a worst-case design, we do aggressive design by tolerating timing errors. Simulation results show large latency savings (up to 35%) for the Terror based system compared to traditional design methodology.

Keywords: Networks on Chips, Reliability, Performance, Link, Aggressive design

I. Introduction

With continued scaling of transistors the wire delay as a fraction of the total delay is increasing [2]. The delay in crossing a chip diagonally for 50nm technology is around 6 to 10 cycles, with only a small fraction of chip area (0.6% to 1.4 %) being reachable in a single clock cycle [2]. Scaling is accompanied with a decrease in supply voltage and an increase in clock rate. This makes wires unreliable as the effect of various noise sources such as cross-talk, coupling noise, soft errors increase [3], [4], [5]. To effectively design future SoCs, Networks on Chips (NoCs), a communication centric design approach that considers the delay and reliability issues of the wires has been proposed in [1], [6].

As the total system performance increasingly depends upon the communication system, the NoCs need to support high throughput, low latency communication with low power requirements. To effectively tackle the delay of NoC links, the links can be pipelined by adding buffers (i.e. flip-flops) that segment the links into stages (Figure 1) [7]. Link pipelining increases the link throughput and reduces the average packet latency for data transmission. The number of cycles needed for a data bit to traverse a pipelined link is equal to the number of pipeline stages in the link. The number of pipeline stages in the link, in turn, depends upon the distance a data bit can travel on the link in one cycle and the length of the link. As the links are becoming increasingly susceptible to various noise sources such as cross-talk, coupling noise, local temperature variations, process variations, etc., the data bit delay on the link in a cycle is becoming unpredictable as the various noise sources introduce significant delay variations in the transmission of a bit. Link design in current NoC design methodologies are based on a worst-case design approach that consider all the delay variations that can possibly occur due to the various noise sources and environmental effects and target a safe operation of the system under all conditions. Such a conservative design approach targets timing error-free operation of the system. In such a scheme, the distance traveled by a data bit is significantly lesser than the ideal case when no timing variations occur due to noise sources. Thus, a worst-case design methodology leads to poor system performance as the number of pipeline stages in the link and hence the link latency is much higher compared to the ideal case when no timing errors occur. An aggressive design approach (in which the data bit is assumed to travel the maximum distance possible in one cycle) can result in large reduction in latency for communication. However, in such a system, timing errors can occur when the noise sources introduce delay variations in data communication.

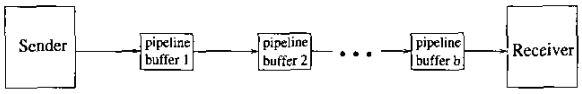

Fig. 1. Pipelined link design, with b pipeline stages in the link

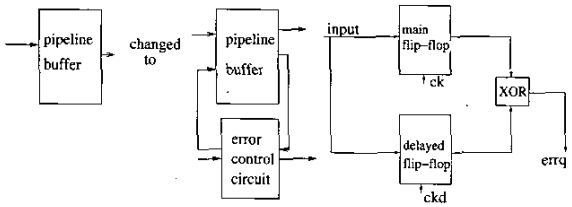

Fig. 2. Proposed Desgin

Fig. 3. Basic Idea

As an example, let us consider an on-chip link that connects all the communicating blocks in an SoC designed in 100nm technology. Assuming a conservative design approach, the number of pipeline stages (and buffers) on the link is 6 (detailed explanation is given in section VIII). On the other hand, if the link is designed aggressively, the distance between successive pipeline buffers can be increased up to 50%, resulting in a 4-stage pipeline. Thus, a 33% reduction in latency is obtained by the aggressive design approach. But, in this aggressive design, timing errors can occur due to delay variations introduced by the environment and wire characteristics. Traditionally, such timing errors are detected by adding redundant bits to the data that is transmitted. Once an error is detected in the receiver, the receiver requests retransmission of the data. The latency and power overheads for retransmission can be quite high. As an example, with 5% error rate, the latency incurred in sending 1000 bits on a 4-stage pipeline link is 1553 cycles, while ideally (when there are no errors) it should take 1003 cycles. Here we have assumed that only the data bits that

0-7803-8736-8/05/$20.00 ©2005 IEEE.

had errors are resent. In most network designs, a *Go Back-N* retransmission strategy is used, where all the data bits following the data with error will be resent [9]. In this case, the latency penalty is much higher. As the power consumption of the communication system in the NoC increases linearly with the amount of data sent, the network power consumption also increases significantly when data is retransmitted.

There is a significant need to mitigate the effect of parasitics on link performance. In this work, we present a timing error tolerant communication system, Terror, that makes the NoC links tolerant to timing errors caused by the unpredictability in environment and wire characteristics. In the Terror based system, the distance between successive pipeline stages is designed such that the latency for data transmission on the link and the number of pipeline stages in the link are much lower than the traditional worst-case design approach. In order to cope with the timing errors that may occur in such an aggressive design approach, the pipeline buffers in the Terror system are augmented with error detection and correction capability, as shown in Figure 2. Any timing error at a pipeline buffer is detected and corrected with a maximum of single cycle penalty. In the Terror system, the latency overhead for error detection and correction is independent of data size and error rate. The system is highly scalable with link (or bus) width and has a latency penalty which is bounded by the number of pipeline stages on the link. For the above example, the Terror based design results in a latency of 1007 cycles, a 35% decrease in latency compared to the traditional design approach in which errors are handled by retransmission of data. The area overhead for the Terror system is less than 0.6% of the total design area of the SoC, which is negligible when compared to the latency savings achieved.

II. Terror Design Principle

In the Terror system, each pipeline buffer (*main flip-flop*) is augmented with a second flip-flop (*delayed flip-flop*) as shown in Figure 3. The *delayed flip-flop* operates on a delayed clock (*ckd*) compared to the *main flip-flop* (clocked at *ck*). The incoming data is sampled twice, once by the *main flip-flop* (at clock edge *ck*) and then by the *delayed flip-flop* (at clock edge *ckd*). The distance between two *main flip-flops* is designed aggressively to improve performance, thereby resulting in an error-prone operation. However, the clock-delay between the *main* and the *delayed flip-flop* (i.e. the delay between the clock edges *ck* and *ckd*) is designed such that even in the presence of unpredictability in the wire characteristics, there is sufficient time for the data bits to reach the delayed flip-flop by clock edge *ckd*. Thus the *delayed flip-flops* are designed to operate in an error-free manner. There are two modes of operation at each pipeline buffer of the Terror: *normal mode* and *delayed mode*. Initially all the buffers are set to the *normal mode* and data transmission begins. In every cycle, at the clock edge *ck*, the *main flip-flop* captures and transmits the incoming data. At clock edge *ckd*, the *delayed flip-flop* captures the incoming data and the error detection control circuit checks whether there is any difference between the main and the *delayed flip-flop* values. If there is a difference, there is an error in the *main flip-flop* value and the data that was transmitted at *ck* is incorrect. The correct data from the *delayed flip-flop* is sent at the next clock edge *ck* and the Terror buffer enters the *delayed mode*. Suitable control signals for the downstream buffers/receiver to recover from the error are also generated and sent. The latency overhead incurred for recovering from the error is one cycle.

Once the Terror buffer has entered the *delayed mode*, all subsequent data is captured and transmitted by the *delayed flip-flop* and the buffer always operates in an error-free manner. Thus after a single cycle penalty, there is no additional penalty for the rest of the data transmitted through this buffer. The same argument applies to all the pipeline buffers of the link. Thus the maximum (worst-case) overhead in sending data

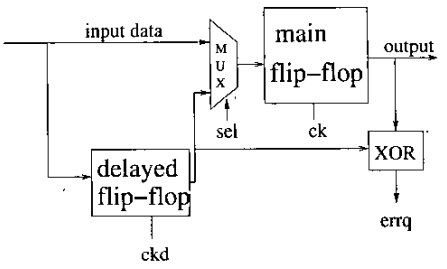

Fig. 4. Logic level implementation of Terror

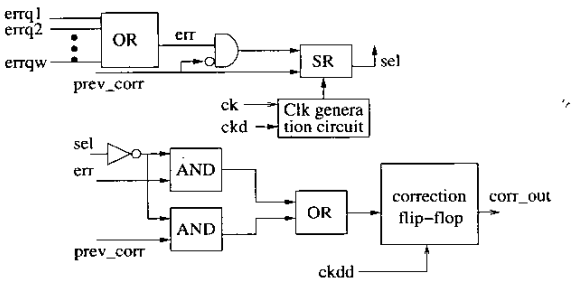

Fig. 5. Error control circuit

through the Terror based link is just the number of pipeline buffers in that link, independent of data size and error-rate. When the data transmission is completed, the Terror buffers that are in the *delayed mode* return back to the *normal mode*.

III. Logic Level Implementation

In the Terror based communication scheme, each pipeline flip-flop is replaced by the Terror buffer, which is composed of two flip-flops (*main flip-flop* and *delayed flip-flop*), a 2:1 multiplexer (MUX) and an XOR gate (refer Figure 4). The delayed clock *ckd* for the *delayed flip-flop* is derived from the main clock *ck* locally at each buffer by using a *delay chain* (a chain of inverters). At each pipeline stage, an error control circuit (Figure 5) is added for generating suitable control signals when an error is detected. Let the number of bit-lines in the link be w lines. The XOR outputs (*errq* signals) generated at all the w bit lines, at each pipeline stage of the link are ORed and fed as an input to the error control circuit. The error control circuit consists of a SR latch, AND, OR gates and a *correction flip-flop*. For proper operation, the *correction flip-flop* is clocked by *ckdd*, which is delayed from the clocks *ck* and *ckd* and locally generated at each pipeline stage. A Terror based link with w bit-lines (width of the link) and b pipeline

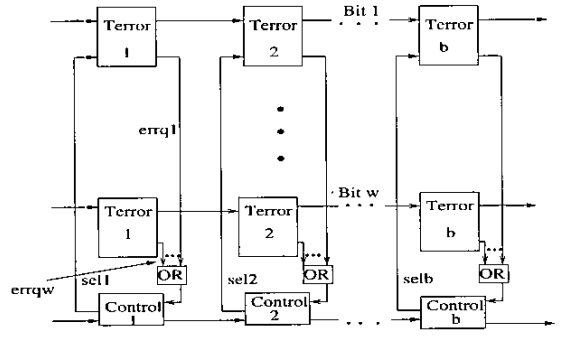

Fig. 6. Link Design using Terror

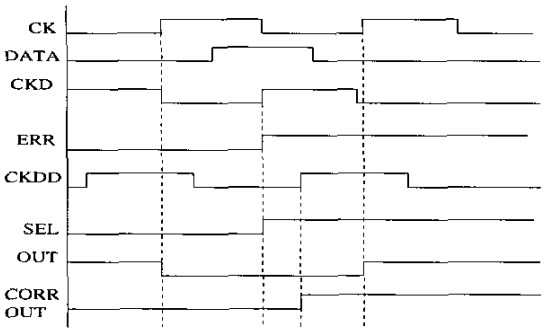

Fig. 7. Normal to delayed mode

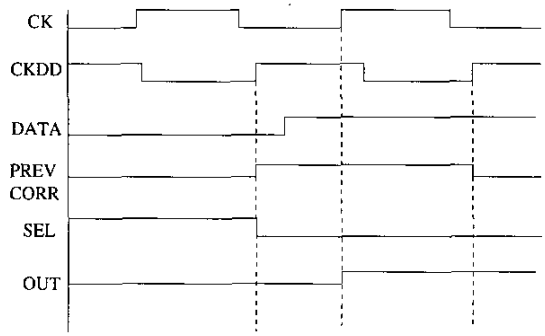

Fig. 8. Delayed to normal mode

buffers (on each bit line) is shown in Figure 6. Note that only one error-correction circuit is used at each pipeline stage for all the w bit-lines of the link, so that all the bit-lines of the link are in synchronization with each other. Moreover, the overhead of the error correction circuitry is also reduced by this design.

When an error is detected by any of the Terror buffer pipeline stages, the *err* signal, which is an input to the SR latch is set. The SR latch output, *sel* is then set to 1, so that the MUX starts sampling the *delayed flip-flop* output and the Terror buffers at this pipeline stage enter the *delayed mode*. Control signal *corr_out* is set to 1 and sent to the next pipeline stage/receiver to indicate that the previously sent data was an error (this *corr_out* signal is received as the *prev_corr* signal by the next pipeline stage/receiver). Once the Terror buffer enters *delayed mode*, no more errors occur as the data sampling is through the *delayed flip-flop* (refer Figure 4). After all the data is transmitted, the MUX control signal is reset to 0 and the Terror buffer returns to *normal mode*. In Figure 7, we show an example where an error is corrected in cycle 2 and the Terror buffer operation transitions from *normal mode* to *delayed mode*.

In the above scheme, we note that if the previous pipeline stage had an error at clock cycle $t-1$ and the current pipeline stage has an error at clock cycle t, then there is no need to resend the data at the current pipeline stage in the next cycle as anyway it is incorrect (because of the error in the previous stage). Instead of correcting and sending the current pipeline stage's data, we can switch back to the *normal mode* and send the new data coming from the previous pipeline stage. This gives a latency savings of one cycle for the case of two contiguous errors. This is shown in Figure 8. To implement this scheme, the SR latch is modified, such that the output is 0 when *prev_corr* is 1 (meaning that the previously received data is wrong and the current data is correct) and the *err* signal is set (meaning that there is an error at this pipeline stage). Similarly, the *corr_out* signal is set to 1 when both the *prev_corr* and *err* are set to 1.

IV. Alternative Approaches

There are many error correcting schemes that correct error without retransmission of data. Teatime [10], tracks the logic delay variation and dynamically changes the clock frequency to eliminate any errors. It tries to avoid timing errors and requires complex analog frequency controller and tracking logic. Also, the correction efficiency depends upon the latency between actual error detection and change in clock frequency. Razor [11] based system has same basic principle as Terror and is used to control power (supply voltage) by monitoring error rate. It detects and corrects error but has one cycle penalty per error occurrence. Also, it gates the global clock to delay the whole pipeline. Clock control may not be always possible, especially in heterogeneous systems. In Terror, only the first error occurrence incurs a single cycle penalty for correction and previous pipeline stages are oblivious to an error occurring at the current stage. The latency savings in Terror comes at the expense of slightly more complex error control circuit when compared to Razor. Favalli et al. [12] assume an encoded data signal which is checked by a small decoder present at input of each flip-flop. In case of error, clock is delayed for one cycle, till the correct value of data settles. Mousetrap [13] is a high speed asynchronous pipeline which ensures correct data availability to consecutive stages, but has a substantial overhead of communication signals (acknowledge and request signals). Eric Dupont et al. [14] suggest to include a latch with delayed clock to detect transient faults due to soft errors. This technique is similar but gives error penalty per occurrence of soft error and involves clock control circuitry. There are many coding techniques for correcting errors such as the Hamming code. But they require extra wires, decoding and encoding circuitry at sender and receiver ends and do not scale well with bus widths. Moreover correcting multiple errors in the data is difficult in such coding schemes. All of these techniques introduce large latency overheads depending on the error rate and have substantial overhead for large bus widths. Terror enabled systems give a bounded penalty irrespective of the data error rate. As an example, in most previous error correcting mechanisms such as Razor [11], if an error occurs at each cycle, then for transmitting N cycles of data, we need $2N$ cycles. This is because, each error occurrence results in a single cycle overhead. Thus the percentage of useful cycles is just $N/2N$=50% and the whole pipeline is delayed by N cycles. In a Terror system, the maximum penalty is limited by the number of Terror elements in the communication link. This is because the one cycle penalty for error correction is incurred only for the first occurrence of error at the input of a Terror element. The operation is such that subsequent data transmissions are error-free. Thus, if the total number of Terror elements between the sender and receiver is b, then the percentage of useful cycles is $N/(N+b)$ and the pipeline is delayed by only b cycles. For $b < N$, which is the usual case in NoC designs, the benefits are substantial.

V. Timing Analysis

In a Terror based system, the reduction in latency when compared to a traditional design approach depends on the delay between the clock edges *ck* and *ckd*. Ideally, the clock *ckd* can be delayed by one cycle from *ck*, so that the number of pipeline stages in the link (and hence the latency) is reduced by 50%. In practice, the delay between the clock edges *ck* and *ckd* is much lower than one cycle as it is bounded by the delay and timing requirements (setup time, hold time) of the logic elements. Note that the effect of the logic delay can be decreased significantly by optimizing the transistor level implementation of the design.

As shown in figure 4, the main flop (clocked by *ck*) has 2:1 MUX at its input. This introduces additional delay in the data

path. Now the data has to arrive earlier at the input pin as compared to a flop without MUX . This increases the setup time of the data input to the Terror element. This increase in setup time is given by t_{mux} (which is the MUX delay). Similarly, we have an AND gate and an OR gate at the input of the *correction flip-flop*, which are in the path of *prev_corr* signal. This increases the setup time requirement of the *prev_corr* signal over the normal set-up time ($t_{setup}(nominal)$). The new set-up time for the *correction flip-flop* is given by:

$$t_{setup} = t_{and} + t_{or} + t_{setup}(nominal). \quad (1)$$

The minimum spacing required between rising edges (assuming all flip-flops are rising edge triggered) of *ckd* and *ckdd* is determined by the total path delay of the *err* signal. The *err* signal has to satisfy the setup time of the *correction flip-flop*. The *err* signal path delay starts from the clock *ckd* to *q* delay of the *delayed flip-flop* , through the XOR and OR gate (ORs *errq* signals) and then through the AND and OR gates. The clock *ckdd* to the *correction flip-flop* should arrive such that it captures the correct value of the *err* signal. This correct value of *err* signal is only available sometime after the rising edge of *ckd* and this time delay is the summation of all the delays in the *err* path and the nominal setup time of the *correction flip-flop*. Thus *ckdd* should be spaced from *ckd* to accommodate the *err* path delay. The minimum spacing t_{ckd} between rising edges of *ckd* and *ckdd* is given by below equation.

$$t_{ckd} = t_{ckq} + t_{xor} + t_{domino-or} + t_{and} + t_{or} + t_{setup} \quad (2)$$

In the case of a bus, the *errq* signals of all bit lines in a link are ORed. This ensures that all the bit lines in the link are in synchronization with each other, simplifying the receiver design. This wide OR can be implemented as a domino gate, instead of a static gate to reduce delay. Also, there is some delay for Terror to go from *delayed* to *normal mode*. Terror goes from *delayed* to *normal mode* when *prev_corr* is set to one. This resets the SR latch output (*sel* signal). This *sel* signal is an input to the 2:1 MUXs. For correct functionality, the *sel* signal should change value before the rising edge of *ck*. The minimum spacing between rising edges of clocks *ckdd* and *ck* is determined by the total path delay of the *sel* signal. The *prev_corr* signal satisfies the setup time of the *correction flip-flop* and hence comes sometime before the rising edge of *ckdd*. Path delay of *sel* signal starts when *prev_corr* arrives, then it goes through the SR latch and then the 2:1 MUX. Minimum spacing (t_{ckdd}) between *ck* and *ckdd* is the total path delay minus the setup time of the *prev_corr* signal.

$$t_{ckdd} = t_{SRlatch} + t_{mux} - t_{setup}(correctionflop) \quad (3)$$

In error correcting circuit, the *err* signal (which is the output of the OR of the *errq* signals) asserts the *sel* signal. The *sel* and *err* signals are also fed as inputs to the *correction flip-flop* of the error control circuitry. Thus *sel* should not change before t_{hold} of the correction flip-flop. We get below hold-time condition:

$$t_{hold} < t_{SRlatch} + t_{and} + t_{or} \quad (4)$$

The timing delays and overheads can be minimized by transistor level optimizations such as including the input MUX into the flip-flop, using a domino OR instead of a static OR, simplifying the latch, combining OR logic into *correction flip-flop*. Figure 9 shows the optimized transistor-level circuit diagram where flip-flops with embedded logic have been used.

Ideally, the clock (*ckd*) to the *delayed flip-flop* can be delayed by one cycle, such that even if the data arrives one cycle late it is captured and sent the next cycle. But due to timing overheads this window is decreased by ($t_{ckdd} + t_{ckd}$). This gives an upper bound on the frequency by which the clock cycle can be decreased beyond specifications.

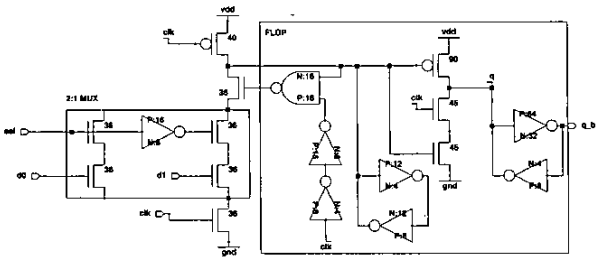

Fig. 9. Semi-Dynamic Flip Flop Design

TABLE I
TIMING OVERHEADS

Parameter	% Overhead
Hold Time	10
32-bit OR delay	8.6
t_{ckdd}	9.7
t_{ckd}	27.0
ckd delay	36.7

VI. TRANSISTOR LEVEL SIMULATION

We designed a transistor level Terror element for a 32 bit bus in 100 nm technology targeted for 1GHz operating frequency, operating at 1.2 V. From SPICE simulations, we obtained values for timing overheads, presented in Table I, where the overheads are expressed as a percentage of cycle. This table is an estimate of the timing constraints and the timing overheads can be reduced substantially by using better transistor sizing techniques, process technology and commercially available CAD tools.

From Table I we see that practically clock *ckd* cannot be delayed beyond 63.3% of cycle time. This is because there is a minimum spacing requirement $t_{ckd} + t_{ckdd} = 36.7\%$ which should be satisfied. Hence only 100-36.7 = 63.3% of cycle is available for *ckd* delay. Also, to satisfy hold time requirement (10%) of the *main flip-flop*, minimum spacing between *ck* and *ckd* should be 10% of the cycle. Due to this, range for variation of *ckd* is limited to 53.3% of the clock cycle.

Errors due to meta-stability of data, as discussed in [11] can occur at the input of the *delayed flip-flop*. These cannot be completely eliminated but can be minimized. Since we use a *delayed flip-flop* instead of a delayed latch used in [11] we significantly minimize short-path constraint problem. Note that for proper Terror operation, the *corr_out* line for signaling the occurrence of an error should be error-free. The *corr_out* line can be made error free by various means such as shielding the line from other bit lines, routing the line in higher metal layer so that it propagates faster, providing parity checker to detect an error in the line, etc. As only a single *corr_out* line is added to the link that typically has multiple bit-lines, the overhead in shielding or conservatively designing the *corr_out* line is low.

VII. ANALYSIS OF PENALTY

The maximum latency penalty in the Terror system is bounded by the number of pipeline stages in the link and is independent of the amount of data sent and the error rate. This is because, a single cycle latency penalty is incurred at a pipeline stage only for the first detection and correction of an error.

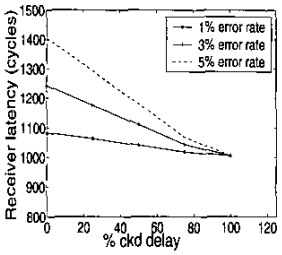

Fig. 10. Receiver latency variation with delay between clocks *ck* and *ckd* for ideal case

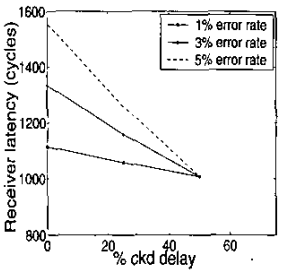

Fig. 11. Receiver latency for practical case

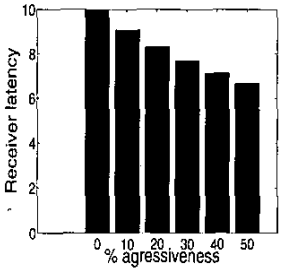

Fig. 12. Latency variation vs.aggressiveness

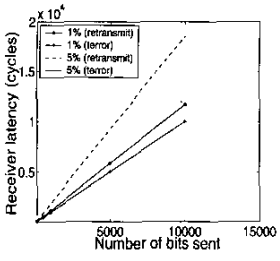

Fig. 13. Comparison of latency for the retransmission and Terror scheme

Once an error occurs at a pipeline stage, the pipeline stage enters the *delayed mode*, so that subsequent data transmission at this pipeline stage is guaranteed to be error free.

The actual latency penalty is a function of temporal and spatial probability of error occurrence. For a Terror link with b pipeline stages, we get

$$1 \leq Penalty \leq b \tag{5}$$

The magnitude of penalty is a function of when and where the timing error occurs in the pipeline. In Figure 1, if a timing error occurs first at the pipeline stage b, followed by an error at pipeline stage $(b-1)$ and so on up to the pipeline stage 1, then the total penalty for error correction is just one cycle. This is because the error in pipeline stage $(b-1)$ is absorbed by Terror at pipeline stage b (since Terror b goes from *delayed* to *normal mode*, incoming error is not propagated by Terror b).

On the other hand, if a timing error occurs first at the pipeline stage 1, followed by an error at pipeline stage 2 and so on up to Terror b, then penalty for error correction is b cycles. This is because at each pipeline stage, a single cycle penalty is incurred for error detection and correction.

Thus, the maximum penalty in the Terror scheme is b cycles, while the actual penalty lies between 1 and b cycles. Also, we note that maximum penalty is independent of the total amount of data sent. This makes Terror design very attractive for high bandwidth data communication of current and future SoCs.

VIII. Simulation Results

We performed several simulation case-studies to quantify the performance (latency) benefits of Terror for different error rates, data sizes and pipeline stages. For the simulations, we consider an SoC with an on-chip link of length 12mm that connects the various components of the SoC. We assume that the link operates at a frequency of 1 GHz. For a conservative design approach that takes into account all the delay variations that can possibly occur due to the unpredictability in the wire and environment characteristics, we assume that the distance between successive pipeline stages is 2mm for safe operation of the links [2]. Thus, for a conservative design the number of pipeline stages on the link is 6.

In an aggressive design approach, the distance between two successive pipeline stages can be increased, so that the total number of pipeline stages in the link and the link latency are reduced. When the data is double sampled (as in Terror), in the ideal case (when we neglect the timing overheads associated with the Terror logic elements), the distance between successive pipeline stages can be doubled when compared to the conservative design. Thus, the number of pipeline stages on the link can be reduced to 3. In this case, the delay between the clocks *ck* and *ckd* should be one cycle to detect all timing errors that can occur due to the aggressive design approach. Figure 10 shows the latency for transmitting 1000 bits of data on the link as a function of the delay between the clocks *ck* and *ckd* of the *main* and *delayed* flip-flops. As the difference between the clocks increases, the number of errors detected and corrected by the Terror scheme starts to increase as the *delayed flip-flop* gets a larger time window to sample the incoming data. In Figure 10, when the difference between the clocks is less than one cycle, we assume that the data bit errors that are not corrected by Terror buffers are retransmitted using an end-to-end flow control mechanism. We have assume that only the data bits that had errors are resent. In most network designs, a *Go Back-N* retransmission strategy is used, where all the data bits following the data with error will be resent [9]. In this case, the latency penalty is much higher. As seen from the plot, there is a significant reduction in the latency as the delay between the clocks *ck* and *ckd* increases.

Though, ideally the distance between successive pipeline stages can be doubled in a Terror scheme, as explained in section VI, practically the distance can only be increased by 50% due to the timing overheads associated with the flip-flops and the logic elements of the Terror system. For the Terror system to detect all the timing errors, the distance between successive pipeline stages for the on chip link needs to be 3mm, so that the above on-chip link is pipelined with 4 stages. Figure 11 shows the latency for transmitting 1000 bits of data for various error rates as a function of the delay between the clocks *ck* and *ckd* for the practical case.

In Figure 12, we plot the variation of the link latency with the percentage aggressiveness. We define the percentage aggressiveness as the percentage by which the distance between successive pipeline stages is increased when compared to the conservative design approach. The plot terminates at 50% aggressiveness, since in our Terror system, the distance between successive pipeline stages can be increased only up to 50% of the conservative design. As expected, the latency of communication decreases significantly as the aggressiveness increases. We obtain a 33% reduction in latency with the Terror based system that utilizes 50% aggressiveness, compared to the conservative design approach.

In Figure 13, the latency for data transmission for two different error rates (1% and 5%) is plotted for various data sizes for the Terror based design and a traditional design where errors are corrected by retransmission. As seen from the figure, the latency for the Terror system for various error rates is almost equal to the latency for an ideal case when there are no timing errors. For a chosen error rate, as the size of data transferred increases, there is significant latency savings in the Terror system when compared to the traditional scheme of retransmission. Moreover, as the error rate starts to increase, there is much larger savings in latency for the Terror based system. For the data size of 1000 bits and error rate of 5%, there is a 35% reduction in latency in the Terror based system when compared to the retransmission scheme.

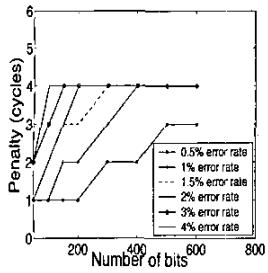

Fig. 14. Terror penalty for different data sizes

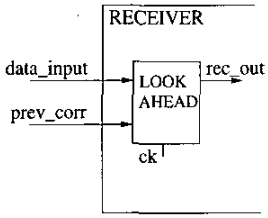

Fig. 15. Receiver interface design

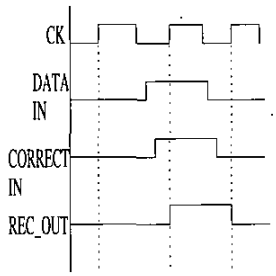

Fig. 16. Look-ahead stage operation

TABLE II
AREA OVERHEAD OF TERROR

Design	Area overhead
1. Merlot	0.12%
2. DSP	0.6%
3. MIT RAW	0.86%
4. Alpha MP	0.9%
Average	0.62%

Figure 14 shows the variation of maximum penalty for error correction in the Terror system, with respect to the total number of bits sent on the link. With increase in error rate and size of data transmitted, the penalty increases until it reaches 4. After the penalty of 4 cycles, which is the number of pipeline stages in the link, the penalty is constant with increasing data size and error rate. Note that a typical error correction mechanism would degrade at higher error rates and large data size. But in Terror enabled system, the latency overhead does not increase with error rate or data size, making it suitable for large bandwidth data transmission of SoCs. Moreover, as explained in [11], the system can be operated at lower voltage levels when errors are permitted to occur in the system. In such a design, significant power savings can be achieved as the error-rate increases and Terror based communication mechanism supports such a design.

We estimated the area overhead when Terror based communication system is added to some example Multiprocessor System on Chips (MPSoCs) available in literature [16], [17], [18], [19]. The area overhead estimation is based on the increase in gate count due to the addition of Terror elements. As seen from Table II, on the average, there is 0.6% increase in area, which is negligible when compared to the large latency savings achieved by the Terror system. The power overhead incurred by the Terror elements during error free operation is negligible when compared to the design power of the MPSoCs (less than 10^{-5}% of total power consumption).

IX. Receiver design

The changes required in the receiver for supporting the Terror based communication system are simple as all the bit-lines of a link are synchronized in the case of an error. As shown in Figure 15, the receiver waits in a *look-ahead* stage when it gets the first data flit. When the *prev_corr* signal that arrives in the next cycle is 0, which means that data received in the previous cycle was correct, the previous data is sent to the subsequent stages on the *rec_out* lines. If *prev_corr* is 1 (refer Figure 16), then it indicates that the data received in the previous cycle had an error and the data is discarded. The receiver now waits on the new data received at this cycle in the *look-ahead* stage. In this scheme, only when the incoming data is known to be correct it is used for further processing. This receiver design introduces a single cycle latency penalty for the first flit of the packet, even if it is error free, due to the *look-ahead* stage. Note that a scheme where the data is processed by the receiver before receiving the *prev_corr* signal would not incur this one cycle penalty, but would require complex recovery mechanism when the data is found to be have an error in the next cycle.

X. Conclusions and Future Work

As *Systems on Chips* become increasingly interconnect limited, efficient design of the interconnects is required for high performance of the overall system. *Networks on Chips (NoCs)*, a communication centric design approach has evolved in the recent years for tackling the delay, throughput and reliability issues of the interconnects. Current NoC design methodologies are based on conservative design approaches for link design that results in poor performance. In this work, we have presented Terror, timing error tolerant communication system, where the NoC links are designed aggressively for high performance operation. Instead of avoiding timing errors in the NoC links by a conservative design approach, the links in the Terror system are designed aggressively by tolerating timing errors. Our aggressive design methodology provides large (up to 35%) latency savings compared to traditional design approaches with very little area overhead. In our future, we plan to enhance the Terror design methodology to incorporate *Quality-of-Service (QoS)* guarantees for the applications.

XI. Acknowledgements

This research is supported by MARCO Gigascale Systems Research Center (GSRC) and NSF (under contract CCR-0305718).

References

[1] L.Benini and G.De Micheli, "Networks on Chips: A New SoC Paradigm", IEEE Computers, pp. 70-78, Jan. 2002.
[2] R. Ho, K.Mai, M. Horowitz, "The Future of Wires", Proceedings of the IEEE, pp. 490-504, April 2001.
[3] K.Aingaran et al., "Coupling Noise Analysis for VLSI and ULSI Circuits", IEEE ISQED 2000, pages 485- 489, March 2000.
[4] K. L. Shepard and V. Narayanan. "Noise in Deep Submicron Digital Design". IEEE/ACM ICCAD-96, pages 524- 531, November 1996.
[5] N. R. Shanbhag, "Reliable and efficient system-on-chip design", IEEE Computer, Vol. 3, Issue: 3, pp. 42-50, March 2004.
[6] W.J.Dally, B.Towles, "Route Packets, not Wires: On-Chip Interconnection Networks", DAC 2001, pp. 684-689, Jun 2001.
[7] L.P.Carloni, K.L.McMillan, A.L.Sangiovanni Vincentelli, "Theory of Latency-Insensitive Design", IEEE Trans. on CAD of ICs and Systems, Vol.20, No.9, pp. 1059-1076, Sep 2001.
[8] Y.Zhao, L.Chen and S.Dey, "Online Testing of Multi-source Noise-induced Errors in the Interconnects and Buese of System-on-Chips", IEEE proceeding of International Test Conference 2002, 10, October, 2002.
[9] J.Warland, P.Varaiya, High-Performance Communication Networks, Morgan Kauffmann Publishers Inc, 1996.
[10] A.K.Uht, "Going Beyond Worst-Case Specs with TEAtime", IEEE Computer, pp. 51-56, March 2004.
[11] D.Ernst e al., "Razor: A Low-Power Pipeline Based on Circuit-Level Timing Speculation", Micro Conference, Dec 2003.
[12] M. Favalli, C.Metra, "Low-level error recovery mechanism for self-checking sequential circuit", DFT 97, pp. 234-242, oct. 1997.
[13] M.Singh, S. M.Nowick, "MOUSETRAP: Ultra-High-Speed Transition-Signaling Asynchronous Pipelines", Proc. ICCD 01, Sept. 2001.
[14] E.Dupont,, M.Nicolaidis, P.Rohr, "Embedded Robustness IPs for Transient-Error-Free ICs", IEEE Design & Test, vol. 19 ,Issue 3,pp. 56-70, May 2002.
[15] F.Klass, "Semi-dynamic and dynamic flip-flops with embedded logic", VLSI Circuits 98, pp. 108-109, June 1998.
[16] S.Matsushita, "Design experience of a chip multiprocessor merlot and expectation to functional verification", ISSS 2002, pp. 103-108, Oct 2002.
[17] B. Ackland et al., "A single-chip, 1.6-billion, 16-b MAC/s multiprocessor DSP", IEEE Journal of Solid-State Circuits, vol. 35, Issue 3, pp. 412-424, Mar 2000.
[18] M.B.Taylor et al., "Evaluation of the Raw Microprocessor: An Exposed-Wire-Delay Architecture for ILP and Streams", ISCA 2004.
[19] A. Jain et al., "A 1.2GHz Alpha Microprocessor with 44.8GB/s. Chip Pin Bandwidth", ISSCC Digest of Technical Papers, pp.240-241, Feb. 2001.