Tolerance to Small Delay Defects by Adaptive Clock Stretching

Swaroop Ghosh, Patrick NDai, Swarup Bhunia*, and Kaushik Roy
School of Electrical and Computer Engineering, Purdue University, IN, USA
*Electrical Engineering and Computer Science, Case Western Reserve University, OH, USA

Abstract—Bridging defects typically manifest themselves as increased path delays instead of stuck-at failures. On the other hand, parametric variations (both inter- and intra-die) increase the spread of the circuit delay. Low power design techniques such as voltage scaling, dual-Vth etc. deteriorate the delay spread further. These mechanisms for delay variations in nanoscaled technologies significantly affect the parametric yield. We propose a new design methodology to tolerate subtle delay failures that arise both due to manufacturing defects and parameter fluctuations. We synthesize the circuit to (a) isolate and predict the critical paths of a circuit; (b) create timing slack between critical and off-critical paths and ensure that they are activated rarely; and, (c) avoid the delay failures in these paths by adaptively stretching the clock period. Since critical paths are the most sensitive section of the circuit in terms of delay defects, we ensure fault-free operation by isolating them and providing extra computation time by predicting their activation. This allows us to achieve the required yield with small performance penalty (due to occasional clock stretching under critical path activation). We present application of the proposed methodology for both linear and non-linear pipeline designs. We also suggest two possible circuit-level implementations of clock stretching using clock gating and handshaking, respectively. Simulations on MCNC benchmark circuits with BPTM 70nm devices show that the proposed technique can achieve good yield by tolerating increased path delays (under variations and bridging defects of various sizes) with small overhead in performance and ~14% die-area compared to the conventional design. For performance analysis, we have implemented the proposed methodology in simple in-order pipeline in SimpleScalar. Simulation results on SPEC2000 benchmarks show less that 2% of IPC (instructions-per-cycle) degradation.

I. INTRODUCTION

Subtle manufacturing defects in nanometer technologies cause increased path delays instead of stuck-at failures. Process parameter variations (both systematic and random) may cause parametric failures in logic circuits due to large delay variations. The increased delay spread causes functional failures with respect to the target frequency leading to yield loss. Conventional wisdom dictates a conservative design approach (e.g., scaling up the Vth or upsizing logic gates) to avoid a large number of chip failures. However, such techniques come at the cost of power and/or die area. Over the past few years, statistical design approach has been widely investigated as an effective method to ensure yield under process variations. Several gate-level sizing and/or Vth assignment techniques [1] have been proposed recently addressing the minimization of total power while maintaining the timing yield. On the other end of the spectrum, design techniques (e.g., adaptive body biasing [2]) have been proposed for post-silicon process compensation and process adaptation to deal with process-related timing failures. Design optimization techniques using gate sizing and dual-Vth assignment to improve power/area typically increase the number of critical paths in a circuit, giving rise to the so-called “wall effect” [3].

In this paper, we present a fault tolerant design technique that attains high yield with respect to delay failures with small area and performance overhead. We achieve this by critical path isolation and adaptive clock stretching [4]. The proposed design methodology isolates the critical paths and makes them predictable and rare even under variations. Activation of delay-critical paths are predicted ahead of time (by decoding few inputs) and are allowed to compute with a stretched clock (e.g., to two cycles assuming all standard operations are single cycle). This lets us tolerate the delay failures (under process variations and manufacturing defects) without large performance penalty or changing the rated clock frequency. We propose an automatic synthesis process based on the above design concept. The synthesis process isolates the critical paths and makes them predictable (based on few primary inputs). The isolated critical paths operate with single cycle at nominal supply voltage for nominal dies. However, for the dies in slower process corners or in case of delay defects, the delay violations for critical paths are tolerated by adaptively stretching the clock. The clock stretching can be either enabled for all dies or can be enabled on die-to-die basis after manufacturing test (to determine if the die fails or passes at rated clock). For example, let us assume that the circuit is designed (using proposed methodology for critical path isolation with associated critical path prediction logic) for nominal frequency of 1GHz. Then after fabrication, the output of the critical path prediction logic of all faster dies (i.e., dies meeting frequencies ≥ 1GHz) can be disabled and they can be operated without any clock stretching operations. However, the slower dies (i.e., dies below 1GHz frequencies) will be operated at rated frequency (1GHz) with adaptive clock stretching operations during the activation of critical paths.

Our design and synthesis process restricts the occurrences of the clock stretching operations by reducing the critical path activation probability. It also increases the delay margin between critical and off-critical paths by both logic synthesis and proper gate sizing so that (a) off-critical paths remain off-critical even under variations and delay defects; (b) aggressive dynamic voltage scaling (DVS) [4] can be applied on slower/defective dies for power saving while maintaining good performance at rated clock frequency.

Although the design technique presented in [4] is based on similar concept, the objective of [4] is to design the circuit for operation at fixed low supply voltage for power saving with occasional two-cycle operations. In this work, the circuit is synthesized and sized in a way that it operates in single-cycle for good dies (and nominal supply) while under delay defects and slower process corners, it can still operate at rated clock frequency with occasional clock stretching operations. Consequently, high yield can be achieved with small performance overhead (for the defective dies). In summary, the circuit designed using [4] always operate with occasional clock...
The schematic of the proposed fault tolerant technique in a pipeline based design is shown in Fig 1. Decoders D1, D2 and D3 predict the activation of critical paths of combinational logic stages by pre-decoding few input states. The combinational logic is designed using the proposed methodology. If the output of one of the decoders is asserted then clock stretching is performed to avoid delay failures. One possible implementation of clock stretching involves stalling the pipeline (by gating the pipeline clock with one-pulse signal generated by decoder outputs as shown in Fig 1). To avoid performance penalty due to clock stretching in good dies, a one time functional test can be performed on the pipeline (after fabrication) to ensure that the delay of critical paths meet the nominal clock frequency. If the test fails, then adaptive clock stretching operation can be enabled for that particular die. In summary, we make following contributions in this paper:

- We propose a fault-tolerant design technique to improve yield and profit in nanometer technologies with minimum impact on performance and area. This is achieved by critical path isolation and adaptive clock stretching to two clock periods. The performance penalty is limited only to the slower/defective dies. The proposed concept of adaptive clock stretching can be extended to more than two clock cycles for further tolerance to process variation and defects.
- The proposed technique provides opportunity to reduce power consumption of slower/defective dies by operating them at lower supply voltage.
- We propose the application of this technique in linear and non-linear pipeline designs and discuss the performance overhead.
- We suggest two possible implementations of adaptive clock stretching that is required for tolerating the delay failures.

The paper is organized as follows. In Section II, we discuss the prior work related to prediction of delay failures. The design flow to synthesize an input netlist for critical path isolation is elaborated in Section III. The experimental results are presented in Section IV. In Section V, we present the application of this technique to a linear and non-linear pipeline designs and proposed two possible implementations of adaptive clock stretching. Conclusions are drawn in Section VII.

II. RELATED WORK

- **Telescopic units**: In [5], authors proposed a technique called “telescopic units” to improve the throughput of the combinational logic blocks. The basic idea is to transform the fixed latency units to variable latency units that can operate at faster speed. Fig 2 shows the schematic of telescopic unit which produces a handshaking signal $f_h$ along with the functional outputs. If the clock period of original logic is $T$, the telescopic unit operates at clock with period $T^* < T$. The handshaking signal ($f_h$) is asserted if the current operation requires more than $T^*$ units of time to finish. Under such circumstances, the combinational logic is allowed to take $2T^*$ units of time. For throughput improvement, the telescopic unit is synthesized in such a way that the activation probability of $f_h$ is very low. The authors also proposed the application of this methodology in pipeline based designs. Note that, the effectiveness of the telescopic unit may be limited by (a) the combinational logic having large number of critical paths (e.g., random logic circuits); (b) the increased complexity of telescopic unit (for handshaking signal generation) and; (c) the increased probability of $f_h$.
- **Critical path isolation technique (CRISTA)**: The approach proposed in [4] designs the circuit for operation at fixed low supply voltage with occasional two-cycle operations (assuming standard operations are single-cycle). The principal idea is to (a) isolate and predict the set of possible paths that may become critical under process variations, (b) ensure that they are activated rarely, and (c) avoid possible delay failures in the critical paths by dynamically switching to two-cycle operation (assuming all standard operations are single cycle), when they are activated. This allows the circuit to operate at reduced supply voltage while achieving the required yield. The above mentioned methodology is implemented using Shannon-based partitioning and selective gate sizing.

III. FAULT TOLERANT DESIGN FLOW

If the critical path a combinational logic is unique with low activation probability, then implementation of the proposed technique becomes easy. Since the delay variations affect only the critical paths, all possible delay failures in the critical path can be tolerated by allowing the critical path to take extra computation time. The activation of the critical path can be predicted by decoding the state of a set of inputs. For example, the critical path of a ripple carry adder (RCA) is unique and its activation can be predicted by decoding carry inputs and intermediate propagate signals. Furthermore, considerable delay slack is present between critical and off-critical paths. This ensures that off-critical paths remain off-critical under
variations and manufacturing defects. Therefore, slower process corners or subtle defects can only cause increased delay of the critical path. If the die fails to meet the nominal frequency, then the predictability of the critical path can be used to avoid the delay errors and prevent yield loss.

The arithmetic units (e.g., RCA) are usually well behaved in terms of critical paths and slack between critical and off-critical paths. The random logic circuits, on the other hand, can have many critical paths and corresponding input conditions for their activations. Furthermore, the slack between critical and off-critical paths can be very small, posing challenge to the application of the proposed methodology. In this section, we present a design flow that enables us to achieve the path delay distribution similar to arithmetic units for random logic circuits.

To achieve fault tolerance in random logic circuits (usually with skewed path distribution as shown in Fig 3(a)), we need to ensure that, (a) the critical paths are confined to a logic section whose activation can be predicted; and, (b) the off-critical paths remain off-critical under process, temperature and supply voltage variations by maintaining a safe timing margin. Fig 3(b) illustrates such a path delay distribution.

To obtain the delay distribution shown in Fig 3(b), the design is partitioned and synthesized in such a way that the paths are divided into several logic blocks so that: (a) these logic blocks are active or remain idle, based on the state of primary inputs; and, (b) the probabilities of activation of the logic blocks containing critical paths (called critical block) are very low. Therefore, it will be possible to predict the activation of a logic block just by decoding the states of certain inputs. Next, gate sizing is performed on the partitioned logic to maximize the slack between critical and off-critical blocks. Note that timing slack critical and off-critical paths (i.e., slack $S_1$ as illustrated in Fig 3(b)) in defective dies can be used in two ways namely, (a) for assignment of lower supply voltage for power saving and, (b) for tolerating delay defects in off-critical paths. Under low voltage or in presence of delay defects, the critical block will operate with stretched clock period while the off-critical ones will operate in single-cycle with little performance degradation (since activation probability of the critical block is very low). This is more clearly illustrated in Fig 3(c) where the dashed curve shows the critical path delay variability under process fluctuations. The critical paths may fail to meet the single-cycle delay target however, since their activation is predicted ahead of time, we may provide an extra clock period for the correct computation and tolerance to delay failure.

The implementation of partitioning the circuit into critical and off-critical sections is similar as described in [4] (i.e., by employing repetitive Shannon expansion [6]) and therefore omitted. However, the gate sizing on the partitioned circuit is performed in such a way that critical cofactor meet the timing slack of $S_1$ with respect to target delay whereas the off-critical cofactors meet slack of $S_2$ as shown in Fig 3(b). We used the Lagrangian-Relaxation based iterative sizing [7] and statistical static timing analysis (SSTA) procedures [8] for this purpose.

IV. SIMULATION RESULTS

The experiments are performed on a set of MCNC benchmark circuits. We used Synopsis Design Compiler [9] for logic optimization in our synthesis flow. For a basis of comparison, the original benchmarks are also optimized for area in Synopsys. The mapping is done to a standard cell library. The circuit delays are computed by using SSTA for BPTM 70nm technology [10] with 1V nominal supply voltage. The delay target ($T_c$) for sizing procedure is chosen by plotting the area-delay curve of the circuit and selecting the knee point delay. Slack targets $S_1$ and $S_2$ for sizing are chosen to be $0.01T_c$ and $0.3T_c$ respectively. The area and delay constraints for Shannon partitioning are kept 40% and 5% more than actual area and delay of the circuit respectively, while the yield targets of original circuit and the cofactors for gate sizing are set to 95%.

Note that partitioning is performed in such a way that the critical cofactor is activated by 4 control variables. The delays of critical and off-critical cofactors (normalized with respect to critical cofactor delay) of the benchmarks are shown in Fig 4(a). It can be observed from this plot that proper delay slacks can be maintained between critical and off-critical paths by using the proposed methodology. The area overhead of the circuit after partitioning and sizing is shown in Fig 4(b). New denotes the proposed circuit whereas Org denotes the original optimized netlist. The average area overhead is approximately 14%. New incurs only 5.8% performance degradation, assuming 50% signal activity of control variables. The penalty reduces to 0.16% if signal activity is 20%. The performance loss is limited only to the slower or defective dies. Note that the performance penalty can be minimized by expanding the critical cofactor further so that the activation probability of critical path is reduced. Since this may increase the area overhead therefore partitioning should be done judiciously.

We also evaluated the tolerance of synthesized circuits with respect to small bridging defects. The defects can occur anywhere in the circuit and it can affect the critical as well as off-critical path delays. We achieve tolerance to such delay failures due to following facts: (a) if the defect affects the off-critical paths, then timing slack between critical and off-critical sections can be used to tolerate a certain defect size and, (b) if the defect affects critical paths, then predictability of critical paths and clock stretching to two-cycles can be used to tolerate it. Note that, the proposed technique achieves tolerance to defects at the cost of small area/performance penalty. A
A. Pipeline design methodology

Our pipeline design methodology is based on a given set of conventional design, on the other hand, can achieve such delay defect tolerance at the cost of huge performance penalty (i.e., 50% if the entire circuit is operated at half of rated clock frequency).

In Fig. 5, we have demonstrated the bridging error tolerance capability of the proposed technique by taking an example MCNC benchmark circuit, set. We have plotted the critical path delay and longest off-critical path delay for different bridging defect sizes. We have considered a single defect that is inserted randomly in the circuit in either critical or longest off-critical paths. The size of the defect is varied from 2Kohms to 40Kohms and path delays are measured using Hspice. The nominal clock period of the circuit is 200ps (which is given by the sum of critical path delay and timing margin to meet 95% yield target under variations). The timing slack of critical and longest off-critical path with respect to one-cycle and two-cycle delay targets is also shown in Fig. 5. The timing slack of off-critical paths is ~80ps (=200ps-120ps) in absence of defects (Fig 5) and can be used to tolerate defects in off-critical paths. The slack of critical path is ~210ps (=400ps-190ps) with respect to stretched clock period. From this figure, we can note that the circuit can tolerate single Bridging defects of size above 4Kohms while maintaining the clock frequency. Below the defects of size less than 4Kohms, the off-critical path delay fails the rated frequency while the critical path fails the stretched clock period.

Note that the defect tolerance capability of different circuits may vary depending on the amount of timing slacks obtained in critical/longest off-critical paths. It is also interesting to observe that the timing slack of critical and off-critical paths also allows us to maintain high yield under process variation without slowing down the clock frequency. For this example, up to 66% variation in delay of off-critical path can be tolerated without any failure. Similarly, ~2X variation in delay of critical path can be tolerated while preserving the correct functionality of the circuit.

V. FAULT TOLERANT PIPELINE DESIGN

In Section III, we presented a design methodology for random logic circuits to achieve tolerance to delay failures. In this section, first we present its application to pipeline-based design where each stage is designed using this methodology and discuss the performance overhead due to pipeline stalls for performing clock stretching operations. We consider both linear and non-linear pipelines in our analysis. Then, we address the implementation details of adaptive clock stretching.

A. Pipeline design methodology

In this section, first we present its application to pipeline-based design where each stage is designed using this methodology and discuss the performance overhead due to pipeline stalls for performing clock stretching operations. We consider both linear and non-linear pipelines in our analysis. Then, we address the implementation details of adaptive clock stretching.

B. Performance analysis of a linear pipeline

Consider an N-stage linear pipeline (Fig. 1) where each stage is designed using the proposed technique. The pipeline is stalled for an extra cycle whenever a clock stretching operation is performed by one or more stages in the current cycle. Probability of pipeline stall ($p_{\text{load}}$) is given by

$$p_{\text{load}} = 1 - (1 - p_i)(1 - p_j) \cdots (1 - p_N)$$

where $p_i$ is the activation probability of critical block of $i^{th}$ stage. If critical block of each stage has same number of control variables ($k$)

$$p_i = p_j = \cdots = p_k = p' = \text{(input switching activity)}$$

Hence,

$$p_{\text{load}} = 1 - (1 - p')^N$$

where $p'$ is the activation probability of critical cofactor of every stage.

If the ideal clock cycle-per instruction (CPI) of the pipeline is given by $\text{CPI}_{\text{ideal}}$, then new CPI is

$$\text{CPI} = \text{CPI}_{\text{ideal}} + p_{\text{load}} \cdot \text{(stall penalty)}$$

where $p_{\text{load}}$ is the probability of pipeline stall due to two-cycle operation is given by

$$p_{\text{load}} = 1 - (1 - p_i)(1 - p_j) \cdots (1 - p_N)$$

Note that the defect tolerance capability of different circuits may vary depending on the amount of timing slacks obtained in critical/longest off-critical paths. It is also interesting to observe that the timing slack of critical and off-critical paths also allows us to maintain high yield under process variation without slowing down the clock frequency. For this example, up to 66% variation in delay of off-critical path can be tolerated without any failure. Similarly, ~2X variation in delay of critical path can be tolerated while preserving the correct functionality of the circuit.
Fig. 6: Performance penalty for different pipeline depths at uniform input switching activity of (a) 20%; (b) 50%

Fig. 7: The DLX pipeline (variable latency units ALU and ADD with decoders D1 and D2 have been highlighted)

D. Implementation of adaptive clock stretching

In a pipeline based design, adaptive clock stretching of a stage indicates stalling of the pipeline for an extra clock cycle. If a clock stretching is required at the input stage of the pipeline then it is important to ensure that the input states of the ith stage do not change so that it can produce correct results in the next cycle. Therefore, a mechanism is required to prevent destruction of input states of the ith stage. Furthermore, the intermediate outputs of the ith stage should not be used for computation in the following stages. In this paper, we describe two possible techniques to achieve this objective:

1. Clock gating of the entire pipeline: This scheme is shown in Fig 9(a). The decoders D1, D2 and D3 predict the activation of critical paths. If the output of any of the decoder is asserted then a single pulse is generated to gate the next rising edge of the clock to the pipeline latches. Clock gating is simple to implement and it satisfies the conditions required for clock stretching. However, the performance penalty can be high because the stalls cannot be hidden.

2. Handshaking protocols: Communication protocols for latency insensitive designs have been presented in [13][14]. Both techniques are aimed at reducing the performance overhead due to variable latency units. In [13], the sender and receiver use an auxiliary storage element in parallel with the pipeline register. If the receiver is not ready to accept the new data, then instead of stalling the pipeline, the data can be pushed to the auxiliary storage element. To reduce the area overhead due to auxiliary storage, an elastic buffer (EB) and handshaking protocol called SELF (i.e., synchronous elastic flow) has been proposed in [14]. Instead of using two storage elements, the EBs efficiently use the master and slave latches of the pipeline flip-flop as main and auxiliary elements respectively. The EB is a simple latch with capability of holding the new data (based on an enable signal) even if the receiver is not ready (Fig 9(b)). The enable is generated by the control logic based on valid and stop signals. The data is being successfully transmitted if valid is ‘1’ and stop is ‘0’. On the other hand, if stop is ‘1’ then data should be transmitted again. The handshaking scheme proposed in [14] can be integrated with our work to partially hide the throughput overhead due to clock stretching operations.

\[
\text{Perf. penalty} = \frac{\text{CPI}_{\text{old}} - \text{CPI}_{\text{new}}}{\text{CPI}_{\text{new}}} \left(1 + \frac{p_{\text{out}}}{p_{\text{in}}} \right) \quad (4)
\]

The performance penalty for different N and k (for input switching activity of 20% and 50%) is shown in Fig 6. It can be observed from this plot that if the control variables have low switching activity (~20%), then the penalty can be restricted within 10% for k=4. The penalty can be large for deep pipeline designs (i.e. large N). We suggest the following techniques for reducing the performance penalty, (a) tune the control variable selection metric to pick low switching inputs as control variables; (b) reduce the activation probability of critical blocks further (i.e., by increasing k); and, (c) use the handshaking protocols to partially hide the performance overhead.

C. Performance analysis of non-linear pipeline

We take example of DLX pipeline [11] to illustrate the performance analysis of non-linear pipeline. Fig 7 shows the five stage non-linear pipeline where two of the stages (ALU and ADD) have been modified to incorporate clock stretching operations. The performance penalty can be computed similar to equations (1)-(4) with N=2 since only two stages may require clock stretching operations in a given clock cycle. We decoded the middle eight bits (bit 29-36) of bit adder in ALU and ADD units for prediction of critical path activation. We simulated SPEC2000 benchmarks in simple in-order pipeline in SimpleScalar [12] to get an estimate of performance loss for actual instructions in microprocessor pipeline. We used ref inputs, fast-forwarded 50 million instructions and simulated 1 billion instructions. The pipeline was stalled on detection of a clock stretching operation. The resulting Instructions per Cycle (IPC) normalized to the nominal design is plotted in Fig 8. Note that the IPC loss due to the scheme is less than 2% in all cases, and for a vast number of benchmarks, there is a negligible performance loss. This results also match well with the theoretical estimate using equation (4) with k=8 and N=2 (0.7% to 3.3% for input signal activities of 20% to 60%).
D1, D2, D3 are Circuit designed using modified CRISTA. D1, D2, D3 are

Fig. 9: Implementation of the pipeline stall: (a) clock gating, (b) SELF protocol [15]

A three stage linear pipeline with the EBs and associated control logic is shown in Fig 9(b). The stop signal can be asserted by pre-decoders D1, D2 and D3 whereas the valid signal can be kept high except when a clock stretching is required. Therefore, the decoder can be used to generate both stop signal (to previous stage) and valid signal (to next stage). The valid signal to the 1st stage of the pipeline can be generated by the issue logic whenever a valid instruction is being issued. The operation of the proposed pipeline with EBs can be elaborated with an example as follows. Let us assume that the 2nd stage of the pipeline (i.e., Comb-2) requires a clock stretching at current cycle (say, c) then a stop signal to 1st stage is asserted (by decoder D2) and a valid signal to 3rd stage is de-asserted. While the stop flag propagates to the master stage M2, the new data from 1st stage can be received by M2 in the next cycle (c+1) while the data from M1 is shifted and held at S2. Hence 1st stage of the pipeline does not require any stalling. Since both master and slave latches M2 and S2 occupy data, a stall in 2nd stage in subsequent cycles will cause latches M1 and S1 to hold the new data. The extra data that has been held in elastic buffers (due to clock stretching) will be processed when the pipeline encounters NOP instructions. Therefore, over a number of cycles (consisting of both active and NOP instructions) the handshaking protocol may cause less throughput loss compared to clock gating technique.

Note that this technique can be extended to non-linear pipelines as well with introduction of fork and join control logic [14]. A five stage DLX pipeline is shown in Fig 10 in order to explain this fact. Few bits of functional units ADD and ALU are decoded (by decoders D1 and D2) to predict the activation of critical path. Each pipeline register can be implemented by master and slave EBs. Furthermore, the valid and stop signals of ID and EX stages are generated by decoders D1 and D2 whereas other stages propagate these values. The control logic for implementation of SELF protocol is also shown in Fig 10.

VI. CONCLUSION

We have proposed a fault-tolerant design technique for improving yield of nanometer circuits. The proposed technique uses a synthesis process to isolate the critical paths and reduce their activation, making the possible delay errors predictable and rare. The delay failures induced by process variations and manufacturing defects (such as bridging defects) can be tolerated by adaptively stretching the clock period (for an extra cycle) at run time. This helps in improving the yield while maintaining the rated clock with small performance and area penalty. We have analyzed linear and non-linear pipeline designs for application of the proposed methodology. We have also suggested two circuit-level implementations of adaptive clock stretching. The proposed design technique appears promising to minimize die failure due to small delay defects and parameter variations.

Acknowledgement: This research was partly supported by Gigascale System Research Center (GSRC’ MARCO) and Semiconductor Research Corp (#1078.002).

REFERENCES