Elastic Timing Scheme for Energy-Efficient and Robust Performance

Rupak Samanta, Ganesh Venkataraman[1], Nimay Shah and Jiang Hu
Department of ECE, Texas A&M University, College Station, TX 77843
[1]Magma Design Automation, San Jose, CA 95110
Email: rupak9@tamu.edu, ganeshv@magma-da.com, {nimay, jianghu}@ece.tamu.edu

Abstract

In nanometer regime, IC designers are struggling between significant variation effects and tight power constraints. The conventional approach - using timing safety margin, consumes power continuously to guard against low probability timing variations. Such power inefficiency is largely eliminated in the Razor technology which detects and corrects variation induced timing errors at runtime. However, the error correction scheme of Razor causes pipeline stalling/flushing and therefore is not preferred in real-time systems or sequential circuits with feedback loops. We propose an elastic timing scheme which can correct timing errors without stalling/flushing pipeline. This is achieved by dynamically boosting circuit speed when timing error occurs. A dynamic clock skew shifting technique is suggested to reduce the boosting cost. An optimization algorithm is also provided to minimize the cost overhead. Compared to conventional safety margin based approach, the elastic timing scheme can reduce power dissipation by $20\% - 27\%$ *on ISCAS89 sequential circuits while retaining similar variation tolerance. After optimization, the boosting is needed for only a small portion of entire circuit. As a result, the area overhead is usually less than* 5%.

1 Introduction

When the VLSI technology scales to $65nm$ and beyond, the endeavor for performance growth is seriously hampered by the fundamental limit on power density. Even worse, variation effects such as process, voltage and temperature variations, are increasingly significant and consequently entail extra timing and power budgets.

Conventionally, the variations are handled by guarding nominal circuit delay with timing safety margins. Because of the uncertainty in variations, designers have to use pessimistically large safety margins such that circuits are guaranteed to work properly under the worst case variations. Such over-design not only makes timing closure difficult but also causes excessive power dissipation. On one hand, the likelihood of the worst case or near-worst-case variations is very small. On the other hand, the power for maintaining the safety margins is consumed almost continuously. Evidently, the efficiency of such power usage is poor. Recently, statistical methods [1, 2, 3, 4] have been developed to reduce the pessimism of safety margins. However, large portions of the safety margins are still retained to guard against at least the near-worst-case variations. Adaptive design techniques [5, 6] can compensate manufacturing process variations, but are not good at handling runtime dynamic variations such as supply voltage fluctuations.

The Razor technology [7] is a breakthrough work that largely eliminates the power inefficiency of safety margin based approaches. Instead of relying on safety margins, Razor achieves variation tolerance through in-situ timing error detection and correction. As a result, power is spent for the worst case or near-worst-case only when it occurs. The power overhead of the error detection and correction is considerably less than the power savings from the safety margin reduction. When correcting a timing error, the Razor pipeline has to be stalled and often flushed [7] via architectural approach. In real-time systems, however, pipeline stalls should be avoided as much as possible. For general sequential circuits with feedback loops, such as finite state machine, it is not obvious how to perform the pipeline flushing.

In this paper, we propose an elastic timing scheme that can correct timing errors without stalling or flushing the pipeline. The key technique of this scheme is dynamic speed boosting. In normal operations, the circuit works with relatively low power consumption. When a timing error is detected, a few parts of the circuit are temporarily switched to a faster speed such that the timing deficit due to the error is compensated. In order to minimize the overhead of the speed boosting, we incorporate dynamic clock skew shifting into the elastic timing scheme. Speed boosting and skew shifting should be applied in such a way that the overall power/cost overhead is minimized. We formulate and solve this problem by mixed integer programming. The management complexity of the elastic timing scheme is moderate and therefore not difficult to handle in practice. This timing scheme can also tolerate multiple simultaneous timing errors. Compared to conventional safety margin based approach, the elastic timing scheme can reduce power dissipation by $20\% - 27\%$ on ISCAS89 sequential circuits. At the same time, our approach retains similar variation tolerance as the conventional approach. Since the boosting technique is applied to only a small portion of each entire circuit, the overall area overhead is usually less than 5%. The short path constraint and metastability problem are handled in the same way as Razor [7].

2 Background on Razor

Our idea of elastic timing scheme is inspired by Razor [7], which is a power-efficient technique for variation tolerance. The

0-7695-3117-2/08 $25.00 © 2008 IEEE
DOI 10.1109/ISQED.2008.82

main idea of Razor is to restrain the power consumption to typical scenarios and handle the low probability timing errors with dynamic corrections instead of relying on safety margins. Since variation-induced timing errors occur with low probability, the power spent on the error correction is much less than that of maintaining safety margins. An important component in implementing this idea is the Razor flip-flop, which is depicted in Figure 1(a). Here, a traditional flip-flop is protected by a shadow latch, which can catch signals missed by the flip-flop. The clock signal arrival time at the shadow latch can be either equal to or later than that to the main flip-flop [7]. The signal miss - or timing error - can be detected by comparing the flip-flop output and the latch output. The shadow latch entails tight short path constraint and requires delay buffers to slow down the short paths. However, the overhead of the delay buffers is limited. It is reported in [7] that the total chip power overhead due to Razor flip-flops and delay buffers is only 2.9%. The metastability issue is also discussed in [7].

There are two error correction schemes suggested in Razor [7]: (i) centralized pipeline recovery, where the error signal stalls the clock network and the pipeline while the corrected logic signal is resent from the shadow latch; (ii) distributed pipeline recovery which flushes and restarts the entire pipeline. The inventors of Razor [7] pointed out that the centralized pipeline recovery is difficult to implement in practice since it is sometimes infeasible to deliver the error (stall) signal from a Razor flip-flop to the clock control gate within one clock cycle. The distributed pipeline recovery is a more practical approach, however, it may cause pipeline stall for several clock cycles and is therefore not ideal for use in real-time systems. Moreover, pipeline flushing is not obviously feasible in circuits with feedback loops, such as finite state machines.

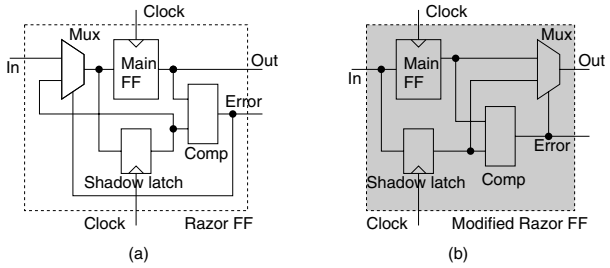

Figure 1. (a) Razor flip-flop. (b) Modified Razor flip-flop.

3 New Timing Error Correction Scheme

Our goal is to overcome the limitation of Razor so that its advantages can be utilized in general pipelined or sequential circuits. The focus of our approach is on a new timing error correction scheme - elastic timing scheme. The timing error detection part is inherited from Razor [7] which has already addressed the issues of delay padding for short paths and metastability. The key idea of the elastic timing is to let the correct signal directly chase after the incorrect signal so that the pipeline does not need to be stalled or flushed. In order to do so, we make a modification to the Razor flip-flop structure as shown in Figure 1(b). After detecting a timing error, we re-send the correct logic signal to the output of the flip-flop instead of its input as was done in Razor. Then, the correct logic signal will be propagated through the combinational logic network in the same clock cycle as the incorrect logic signal which caused the timing error. Our intention is to let the corrected signal eventually overwrite the incorrect signal so that pipeline stall is no longer necessary.

4 Dynamic Speed Boosting

In the proposed timing error correction scheme, the launch time of the corrected logic signal is later than usual because (1) the signal arrives at the shadow latch later than the active clock edge of the main flip-flop and (2) the signal is sent to the multiplexer output after the comparator delay. Therefore, its arrival time to the next flip-flop might be too late to be captured, i.e., the next flip-flop may still capture the incorrect logic signal. We propose to momentarily boost the speed of combinational logic circuit such that the late launch time is compensated. Since we do not use large safety margins, the saved timing budget allows the circuit to run at reduced speed and low power in normal operations. Hence, the circuit speed should have room for acceleration when timing errors occur. There are two options for speed boosting, which are introduced below and illustrated in Figure 2.

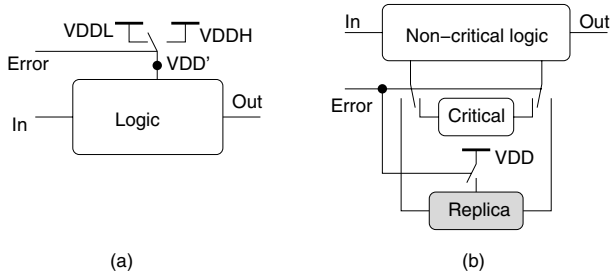

Figure 2. Speed boosting by (a) dynamic dual-V_{DD} and (b) dynamic fast lane.

4.1 Dynamic Dual-V_{DD}

The circuit is supplied by a low V_{DD} in normal operations and switched to a high V_{DD} when timing errors occur. Conventional dual-V_{DD} designs apply different V_{DD} levels to different circuit blocks in a static manner. As long as the low V_{DD} level is significantly higher than the threshold voltage of high V_{DD} blocks, the circuit functionality is not affected when a low V_{DD} block directly drives a high V_{DD} block. However, a low threshold voltage in high V_{DD} blocks may cause high leakage current. Therefore, people use a relatively high threshold voltage for high V_{DD} blocks and insert a level converting circuit between low V_{DD} and high V_{DD} domains to reduce the leakage power [8]. In our case, the level converting circuit is skipped because (1) it causes a delay penalty that degrades the effect of speed boosting, and (2) the power overhead at high V_{DD} is very limited due to low probability of error occurrence. The switches between the power supply network and logic circuits can be implemented by sleep transistors as in power gating [9].

4.2 Dynamic Fast Lane

The timing critical part of the circuit is replicated and the replica can run at significantly faster speed compared to the original circuit. When a timing error is detected, the logic computation is switched from the original circuit to the fast replica. After the error is corrected, the computation is switched back to the original circuit. There are various techniques to obtain high speed for the replica: higher V_{DD}, lower V_{th}, forward body bias, larger and properly sized gates, or a combination of those. In normal operations, the replica idles and does not consume dynamic power. Its leakage power can be reduced by using power gating. However, the wake-up of power gated replica takes some time, often 2 or 3 clock cycles [9]. Therefore, only shallow sleep mode [9] can be applied for the power gating here so that the wake-up time is not too long. In addition, power gated replica should be structurally some distance away from where the error is detected, so that it has sufficient time to wake up before the corrected signal arrives. Although it seems that the area penalty of the replica approach is large, its overall impact to the entire chip area can be very limited if it is judiciously applied at only a few very critical places.

5 Shared Boosting via Dynamic Skew Shifting

We strive for minimizing the area and power delivery overhead of the dynamic speed boosting. A key observation is that we do not need to make every logic stage boostable even if *every* logic stage has small timing slack. A boosting can be shared by multiple stages because an error does not have to be corrected immediately after it is detected. As long as the error can be corrected before it is propagated to the primary output of the entire chip or block, the functionality and timing budget of the chip or block are not affected.

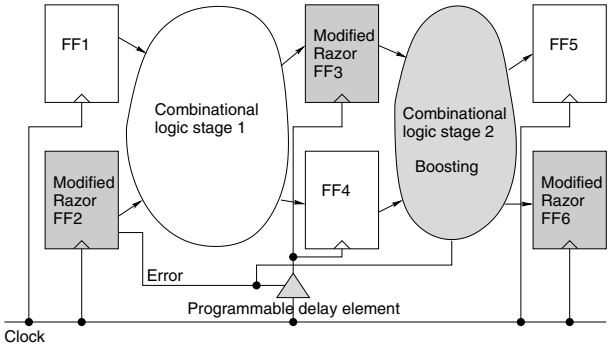

Figure 3. Elastic timing via simultaneous skew shifting and speed boosting.

In order to achieve the boosting sharing, we propose dynamic clock skew shifting such that the timing deficit resulted from a timing error can be transferred among different stages. Consider the example in Figure 3. If a timing error is detected at flip-flop FF2, we temporarily shift the clock signal arrival time at FF3 and FF4 to a later time. Then, the late launch times at the outputs of flip-flop FF3 and FF4 are compensated by boosting at logic stage 2. The skew shifting can be realized using programmable delay elements [10] where certain capacitive load can be dynamically connected or disconnected from the clock signal paths through pass transistors. The pass transistors are controlled by the error signals so that they can be turned on/off at runtime. Although the proposed boosting sharing can reduce boosting cost, it causes skew shifting cost which includes programmable delay elements [10] and the control interconnect. Therefore, we need to find the best tradeoff between the skew shifting cost and the boosting cost.

6 Timing Control

In the elastic timing scheme, we need to manage the timing interaction among logic signals, the error signal and boosting switchings. This timing management should not be more complex than managing the interaction between logic signals and the clock signal in conventional designs.

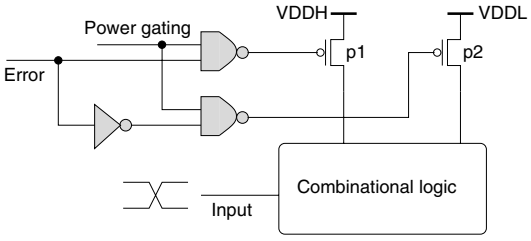

Figure 4. Dynamic dual-V_{DD} with power gating.

We first discuss the case of dynamic dual-V_{DD} which is implemented by two sleep transistors (p1 and p2 in Figure 4). This involves the switchings of p1, p2 and the logic circuit and depends on the direction of mode change: from normal to boosting mode or from boosting to normal mode. The error signal arrival times at p1 and p2 are denoted as d_1 and d_2, respectively. The effects of different scenarios are listed in Table 1. Consider the case that the logic circuit is switching from normal mode to boosting mode. If $d_1 < d_2$, i.e., p1 is turned on before p2 is turned off, there is a short moment when both p1 and p2 are on. Then, there is a short circuit between $V_{DD,H}$ and $V_{DD,L}$. If $d_1 > d_2$, there is a moment that both p1 and p2 are off. If the logic circuit does not switch during this moment, the logic signals retain their levels like in a dynamic circuit. Otherwise, the logic switchings are paused and therefore their delays are increased. The cases when the circuit changes from boosting to normal mode are symmetric.

Table 1. Sleep transistor timing in dynamic dual-V_{DD} boosting. The error signal arrival time to p1 and p2 of Figure 4 are d_1 and d_2, respectively.

Mode change	Logic switches		No logic switching	
	$d_1 < d_2$	$d_1 > d_2$	$d_1 < d_2$	$d_1 > d_2$
Norm $\rightsquigarrow$ boost	V_{DD} fight	Extra delay	V_{DD} fight	No effect
Boost $\rightsquigarrow$ norm	Extra delay	V_{DD} fight	No effect	V_{DD} fight

In order to minimize the short circuit between the two V_{DD}s, the time interval when both p1 and p2 are on should be minimized. This can be achieved by carefully placing p1/p2 and routing the error signal to p1/p2. The two sleep transistors should not be far from each other. The routing of the error signal to

them needs to be performed like clock routing such that the skew $|d_1 - d_2|$ is minimized. If $|d_1 - d_2|$ is less than a few picoseconds and $V_{DD,H} - V_{DD,L}$ is not large, the V_{DD} fight becomes indiscernible. If p1 and p2 are not far from each other, it is not difficult to make the skew within a few picoseconds. If the skew $|d_1 - d_2|$ is small, the time interval when both p1 and p2 are off is also small. Therefore, the resulting extra delay on the logic switchings is small and can be neglected. As a result, the logic signal and the error signal can switch at the same time, and the skew $|d_1 - d_2|$ should be minimized.

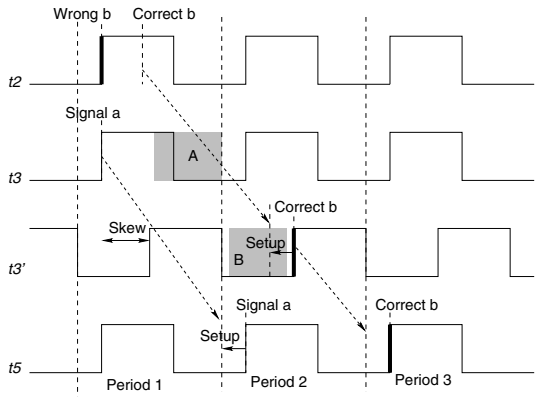

Figure 5. Timing diagram for the pipeline in Figure 3. t_3 is the clock arrival time at FF3 in normal mode. t'_3 is the clock arrival time at FF3 after skew shifting. The error signal should arrive FF3 in shaded interval A and arrive logic stage 2 in shaded interval B.

Now we discuss the requirement to the error signal timing. Consider the example of Figure 3 whose timing diagram is depicted in Figure 5. The clock arrival time to a flip-flop i in normal mode is denoted as t_i. The corresponding time after skew shifting is t'_i. Assume a timing error is detected when FF2 is trying to catch logic signal b. In other words, a wrong signal for b is caught by FF2 at its active clock edge, which is indicated by the thickened rising edge for t_2 in Figure 5. At the same time, logic signal a is correctly captured by FF3. The correct signal b is sent from FF2 at a later time. Then, the error signal should shift the clock signal arrival time t_3 to t'_3. The error signal can arrive FF3 within a time interval indicated by shaded region A in Figure 5. The error signal also switches logic stage 2 to boosting mode. This should happen after logic signal a is captured by the next stage, for example, FF5 in Figure 3. The switching should be triggered before the corrected signal b arrives logic stage 2. Hence, the allowed time interval for the error signal to arrive stage 2 is the shaded region B in Figure 5. In clock period 2 indicated in Figure 5, the error signal disappears at the output of FF2. As a result, t'_3 is switched back to t_3 and logic stage 2 is restored to normal mode. If there is another timing error following signal b, the skew shift and the boosting mode are retained. It can be seen that there are certain constraints to the error signal time, but these constraints are not strict in general.

7 Optimization for Minimizing the Overhead

When designing an elastic timing scheme, we first need to choose where to use modified Razor flip-flops (MRFF). If a combinational path has large timing slack even under low power implementation (low V_{DD}, high V_{th} and small gate sizes), then MRFF is not needed there. Therefore, we use MRFF at the destination end of a combinational path only when its timing slack is relatively small. Next, we need to decide which logic stages should be boostable and where to use skew shifting. In this regard, we consider circuit timing constraints in addition to boosting and skew shifting cost. We wish to minimize the boosting and skew shifting cost subject to timing constraints. This problem can be formulated and solved as a mixed integer linear programming (MILP) which will be elaborated as follows.

We define some notations before describing the MILP formulation. The clock signal arrival time to flip-flop i is denoted as t_i[1]. Each t_i might be shifted by x_i, the value of which is decided by the optimization engine. Due to spatial and fabrication restrictions for programmable delay elements, x_i is bounded in a range $[l_i, u_i]$. According to design rules, x_i can take only discrete values. However, the number of the discrete values is plenty such that we can approximately treat x_i as a continuous variable. We use a linear function $\alpha_i x_i$ to model the cost of skew shifting, although other forms of cost functions can be accommodated in our optimization framework as well. Let $D_{ij,0}$ denote the maximum delay in the logic path between flip-flops i and j in normal mode. Assume there are b_{ij} options for boosting the speed of this path. We use $D_{ij,l} (1 \leq l \leq b_{ij})$ to represent the maximum boosted delay when boosting option l is implemented. The corresponding boosting cost is denoted as $\beta_{ij,l}$. These options are sorted in non-decreasing order of their cost. We also define decision variables $\{y_{ij,0}, y_{ij,1}, ..., y_{ij,b_{ij}}\}$ to choose among the options for boosting. If $y_{ij,0} = 1$, no boosting is implemented. Let T denote the clock period. The delay overhead of error detection at flip-flop i is Δ_i, which includes the lateness of the signal arrival to the flip-flop input and the delay of comparator and multiplexer. The optimization for elastic timing scheme is aimed to minimizing boosting and skew shifting cost subject to timing constraints when timing errors occur. This is formulated as the following MILP problem.

$$\text{Minimize} \quad \sum_i \alpha_i x_i + \sum_{ij} \sum_l \beta_{ij,l} y_{ij,l} \tag{1}$$

$$\text{Subject to} \quad l_i \leq x_i \leq u_i, \forall i \tag{2}$$

$$t_i + \Delta_i + \sum y_{ij,l} D_{ij,l} \leq t_j + T + x_j, \forall\, path\ i \rightsquigarrow j \tag{3}$$

$$t_i + x_i + \sum y_{ij,l} D_{ij,l} \leq t_j + T + x_j, \forall\, path\ i \rightsquigarrow j \tag{4}$$

$$\sum y_{ij,l} = 1, \forall\, path\ i \rightsquigarrow j \tag{5}$$

$$y_{ij,l} \in \{0, 1\}, \forall\, path\ i \rightsquigarrow j, \forall l \tag{6}$$

The objective function (1) is to minimize the overall cost from skew shifting and boosting. The constraints of the above formulation is for the scenario where a timing error has occurred and

[1]For a Razor based flip-flop, the clock signal arrival time at its shadow latch may be different from that at the main flip-flop [7]. Here we assume that the two times are equal to each other for the convenience of presentation without loss of generality.

the error correction procedure is triggered. Constraint (2) enforces the allowed skew shifting range. Inequality (3) is the long path constraint when a timing error occurs at MRFF i. Inequality (4) is the long path constraint when clock skew is shifted at flip-flop i, which can be either a conventional flip-flop or an MRFF. The short path constraints are omitted for the sake of brevity and can be easily added. Constraint (5) ensures that no more than one option of boosting is selected for each logic stage. If power gating is applied with fast lane boosting, the boosting stage cannot be at immediate fanout of an MRFF. This constraint can be realized by prefixing the value of corresponding decision variable $y_{ij,l}$ to be 0. This mixed integer linear programming problem can be solved using existing solvers.

8 Experimental Results

In the experiments, we compare the proposed elastic timing scheme with conventional safety margin based approach[1]. The comparison is focused on power dissipation while the clock periods in both approaches are chosen to be the same for each circuit. We make sure that all timing errors can be recovered by the elastic timing scheme under these clock periods. In the conventional approach, a relatively high V_{DD} is employed to maintain the timing safety margin. In the elastic timing scheme, a relatively low V_{DD} is used in its normal operations.

The experiments are carried out on ISCAS89 sequential circuits which include circuits with feedback loops. The numbers of logic gates and flip-flops for these circuits are listed in the second and the third column of Table 2. The V_{DD} for the conventional timing scheme is $1.05V$. The power and delay of each gate are treated as random variables following Gaussian distribution. The nominal power and delay of each gate are computed based on $90nm$ technology BPTM model [11] and SPICE characterization. The standard deviation (σ) of each random variable is set to be 5% of its nominal value. Each gate delay is characterized by SPICE simulation. The longest path delay of each logic stage is obtained by static timing analysis. The clock period T for each circuit is in column 4 of Table 2. It is decided by adding safety margin (*SM*) to the maximum variational path delay. If the longest path delay from flip-flop i to flip-flop j is D_{ij} and its standard deviation is σ_{ij}, then $T = \max_{\forall \text{paths}}(D_{ij} + 3\sigma_{ij}) + SM$. The fifth column of Table 2 shows the minimum timing safety margin for each circuit in term of T. The average power dissipation is displayed in the rightmost column.

The experimental results from the elastic timing scheme are summarized in Table 3. The second column lists the number of Modified Razor Flip-flips (MRFF) for each circuit. The mixed integer linear programming (MILP) problem (1-6) was solved using a public domain solver GLPK (GNU Linear Programming Kit)

[1]It is difficult to compare our method with Razor for two reasons: (1) our work is targeted to general circuits, especially sequential circuits with loops, where the Razor technology is not directly applicable; (2) our error correction technique is at circuit level and the corresponding benchmarks do not have architectural level infrastructure for pipeline flushing like in Razor. We anticipate that the power savings from our technique is less than that of Razor since Razor is applied together with dynamic voltage scaling. Perhaps this is a price paid for our capability of non-stalling error correction.

Table 2. ISCAS89 benchmark circuits and results from conventional safety margin based timing scheme. T is clock period. Power is in mW. Safety margin is denoted as SM.

Case	#gates	#FF	$T(ps)$	SM	Power
S444	126	21	165	$0.16T$	1.0
S526	163	21	171	$0.15T$	1.5
S526n	159	21	190	$0.15T$	1.3
S820	272	5	192	$0.14T$	1.9
S832	279	5	186	$0.03T$	1.9
S1423	535	74	567	$0.11T$	1.3
S9234	811	135	360	$0.16T$	3.4
S13207	1793	453	525	$0.12T$	4.8
S38584	8182	1285	518	$0.14T$	21.7

downloaded from $http://www.gnu.org/software/glpk/glpk.html$. This solver was run on a Linux machine with 2 dual-core Intel Xeon processors of 3.2GHz and 8G memory, and the corresponding CPU runtimes in seconds are reported in the third column of Table 3. The MILP solution chooses to shift the clock skew for certain flip-flops depending upon the objective function and constraints in (1-6). Column 4 and column 5 of Table 3 show the number of flip-flops with skew shift and the maximum amount of shift.

For each logic stage, there are two boosting options: dynamic dual-V_{DD} or dynamic fast lane. In normal mode, the circuits in the elastic timing scheme operate under V_{DD} of $0.9V$. If a logic stage is implemented with dynamic dual-V_{DD}, its V_{DD} switches to $1.1V$ in boosting mode. For dynamic fast lane, the fast replica is implemented with $V_{DD} = 1.1V$, low V_{th} and forward body bias. Since no gate resizing is performed for the fast replica, its area is approximately the same as its original circuit. Based on the results of MILP, each logic stage is implemented either without boosting, with dynamic dual-V_{DD} boosting or dynamic fast lane boosting. In Table 3, column 6 indicates the number of logic gates with dynamic dual-V_{DD} boosting. Since the size of each logic stage is small, we use two sleep transistors as dual-V_{DD} switches in a logic stage. Column 7 provides the numbers of replicated gates in dynamic fast lane.

The variation tolerance of the elastic timing scheme is tested through Monte Carlo simulation of 1000 times. In Table 3, column 8 shows the number of timing errors occurred in the Monte Carlo simulations. All these timing errors are successfully corrected in the elastic timing scheme. The clock periods T in the elastic timing scheme are the same as those in Table 2. We define **error protection margin** as the timing safety margin that prevents the error correction mechanism from failures in the worst case variations. This is different from the conventional safety margin which is to avoid any timing errors. However, both of them are utilized to ensure that a circuit can function properly under the worst case variations. The error protection margin of the normal mode is the minimum between two cases: (1) a logic path with Modified Razor Flip-flop (MRFF) at its destination, and (2) a logic path with conventional flip-flop at its destination. In case (1), the error protection margin is obtained by $\min_{\forall \text{paths}}(1.5T - D_{ij} - 3\sigma_{ij})$ where

Table 3. Experimental results from the elastic timing scheme. CPU time is in s and power is in mW. The percentages in column 4 and 7 are with respect to the total number of cells.

Case	#MRFF	MILP CPU	Skew shift #shifts	Skew shift Max	Dual-V_{DD} #gates	Fast lane #gates	#errors	Err protect margin Normal	Err protect margin Boosting	Power	Reduction
S444	1	0.03	3 (2%)	$10ps$	41	0	25	$0.12T$	$0.12T$	0.8	20%
S526	6	0.01	10 (5%)	$14ps$	52	0	0	$0.11T$	$0.10T$	1.2	24%
S526n	3	0.04	8 (4%)	$15ps$	91	9 (5%)	28	$0.11T$	$0.13T$	1.0	22%
S820	5	0.01	0 (0%)	$0ps$	86	0	9	$0.48T$	$0.62T$	1.4	27%
S832	4	0.01	1 (0%)	$13ps$	121	0	7	$0.11T$	$0.11T$	1.4	25%
S1423	9	16.00	24 (4%)	$42ps$	316	21 (3%)	8	$0.10T$	$0.08T$	1.0	24%
S9234	12	1.00	13 (1%)	$25ps$	180	0	4	$0.11T$	$0.15T$	2.5	25%
S13207	10	2.00	8 (0%)	$18ps$	107	0	3	$0.19T$	$0.04T$	3.6	26%
S38584	2	41.00	6 (0%)	$25ps$	296	0	0	$0.19T$	$0.19T$	15.8	27%

the $0.5T$ is from the shadow latch protection in MRFF. The error protection margin in case (2) is the same as the conventional approaches. The error protection margin of the boosting mode is the minimum difference between the two sides of inequality (3) and (4). The error protection margins in normal mode and boosting mode are shown in column 9 and column 10, respectively. The rightmost two columns of Table 3 display the average power dissipation, which includes the extra power of speed boosting and skew shifting triggered by timing errors, and the power reduction compared to the conventional safety margin based approach.

Observations and discussions:

- Our approach has a clear advantage on power dissipation which is $20\% - 27\%$ less than the conventional approach.
- The MILP solver tends to choose more dual-V_{DD} boostings than fast lane boostings. Sine fast lane boosting is more effective but more expensive, it is employed only in cases where dual-V_{DD} boosting is not sufficient to correct timing errors.
- The area increase of our approach is limited. This can be observed from column 4 and 7 of Table 3. Roughly speaking, a skew shift induces an extra gate of small size. Thus, the area increase compared to the conventional approach is rarely greater than 5%.
- The options of V_{DD} levels in the elastic timing scheme is significantly less than that in Razor, which is applied with dynamic voltage scaling. Consequently, our control to the error rate is not as refined as in Razor. In some circuits, we did not meet any timing errors in the 1000-run of Monte Carlo simulation. However, the error rate showing in column 8 of Table 3 is never greater than 3%. This error rate is sufficiently low such that the extra power dissipation in boosting is not significant.
- The error protection margins in Table 3 are close to the safety margins in Table 2. This implies that the elastic timing scheme has similar variation tolerance as the conventional safety margin based timing scheme.

9 Conclusions

In this paper, we propose an elastic timing scheme which can correct timing errors induced by variations. The correction is performed at runtime and does not cause pipeline stall. Using this scheme can reduce timing safety margins and corresponding power dissipation without sacrificing robustness. Through judicious usage, the area overhead of our approach can usually be controlled within 5%.

10 Acknowledgment

The authors would like to thank Nikhil Jayakumar at Texas Instrument and Kanupriya Gulati, a senior graduate student at Texas A&M University, for their help in synthesizing ISCAS89 testcases using SIS.

References

[1] H. Chang and S. S. Sapatnekar. Statistical timing analysis under spatial correlations. *TCAD*, 24(9):1467–1482, September 2005.

[2] C. Visweswariah, K. Ravindran, K. Kalafala, S. G. Walker, and S. Narayan. First-order incremental block-based statistical timing analysis. *DAC*, pages 331–336, 2004.

[3] S. Raj, S. B. K. Vrudhula, and J. Wang. A methodology to improve timing yield in the presence of process variations. *DAC*, pages 448–453, 2004.

[4] M. Mani, A. Devgan, and M. Orshansky. An efficient algorithm for statistical minimization of total power under timing yield constraints. *DAC*, pages 309–314, 2005.

[5] J. W. Tschanz, S. G. Narendra, R. Nair, and V. De. Effectiveness of adaptive supply voltage and body bias for reducing impact of parameter variations in low power and high performance microprocessors. *IEEE JSSC*, 38(5):826–829, May 2003.

[6] T. Chen and S. Naffziger. Comparison of adaptive body bias (ABB) and adaptive supply voltage (ASV) for improving delay and leakage under the presence of process variation. *IEEE TVLSI*, 11(5):888–899, October 2003.

[7] S. Das, D. Roberts, S. Lee, S. Pant, D. Blaauw, T. Austin, K. Flautner, and T. Mudge. A self-tuning DVS processor using delay-error detection and correction. *IEEE JSSC*, 41(4):792–804, April 2006.

[8] S. H. Kulkarni, A. N. Srivastava, and D. Sylvester. A new algorithm for improved VDD assignment in low power dual VDD systems. *ISLPED*, pages 200–205, 2004.

[9] K. Agarwal, K. Nowka, H. Deogun, and D. Sylvester. Power gating with multiple sleep modes. *ISQED*, pages 633–637, 2006.

[10] J.-L. Tsai, D. H. Baik, C. C.-P. Chen, and K. K. Saluja. A yield improvement methodology using pre- and post-silicon statistical clock scheduling. *ICCAD*, pages 611–618, 2004.

[11] Y. Cao, T. Sato, D. Sylvester, M. Orshansky, and C. Hu. New paradigm of predictive MOSFET and interconnect modeling for early circuit simulation. *CICC*, pages 201–204, 2000.