Variability mitigation using correction function technique

Conference Paper · December 2013

DOI: 10.1109/ICECS.2013.6815412

CITATIONS
2

READS
254

3 authors:

Shady Agwa
Zewail University of Science and Technology
8 PUBLICATIONS **15** CITATIONS
SEE PROFILE

Eslam Yahya Tawfik
The Ohio State University
40 PUBLICATIONS **196** CITATIONS
SEE PROFILE

Yehea Ismail
The American University in Cairo
454 PUBLICATIONS **4,242** CITATIONS
SEE PROFILE

Variability Mitigation Using Correction Function Technique

Shady Agwa[1], Eslam Yahya[1,2], Yehea Ismail[1]

[1]Center of Nanoelectronics and Devices (CND)
American University in Cairo, Zewail City of Science and Technology
[2]Benha Faculty of Engineering, Benha University
Cairo, Egypt
shady_agwa@aucegypt.edu, eslam.yahya@aucegypt.edu, y.ismail@aucegypt.edu

***Abstract*— AS the fabrication technology migrates towards nanometer scale; the timing constraints of sequential circuits have become more critical. Process, voltage and temperature variations (PVT) increase the unreliability of the sequential circuits. There are different techniques to tolerate the variability and to mitigate the critical timing of the sequential circuits. Traditional techniques, using clock skewing or soft-edge flip-flop, relax the timing conditions by stealing time from adjacent stages. In contrast to traditional techniques, using correction function technique can give an indication about the error rate which is useful to re-adjust the supply voltage or the operating frequency. The correction function technique is able to detect the error of one of the 4x4 bit multiplier's inputs and correct the output data without flushing the pipeline stages with 1.459x overhead of area and 1.427x overhead of power.**

I. INTRODUCTION

The sequential circuit, shown in Figure 1, consists of combinational logic blocks that are surrounded by flip-flops. These flip-flops isolate each stage of combinational logic block and they are used to transfer data between adjacent stages according to a common clock. At the rising or falling edge of the clock signal, every flip-flop captures the output data of its previous stage and transfers the data to the next stage. Clock period of sequential circuit is chosen according to the worst case of the most critical stage.

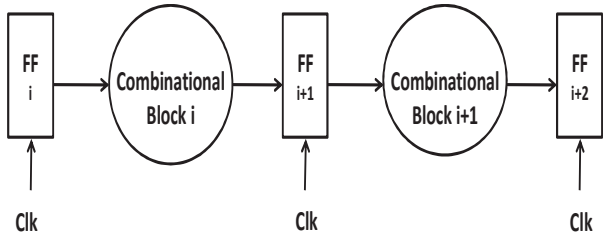


Figure 1. General architecture of sequential circuit [1].

The timing of sequential circuits depends on the common clock. Every combinational logic stage has to finish its computation within the given clock period. Every design should take into consideration the setup time, the hold time and the clock skew to ensure the correctness of transferring data between stages. The timing of sequential circuits should be validated by two constraints to prevent receiving incorrect early data or missing correct late data. The first constraint is the short path constraint, illustrated by (1).

$$T_{i+1} + T_H < T_i + \mathrm{d(i, i+1)} \qquad (1)$$

While T_i is the arrival time of the clock at flip-flop i. T_{i+1} is the arrival time of the clock at the next flip-flop i+1. T_H is the hold time which is the amount of time needed after the clock edge for the data input to be held stable to make sure that the master latch of the flip-flop has been fully disabled. d(i, i+1) is the minimum propagation delay of the combinational logic stage i. Satisfying this constraint guarantees enough time for flip-flop i+1 to save the data before the new data, from flip-flop i, propagates through the short path of the combinational logic stage and damaging the old correct data.

The second constraint is the long path constraint, illustrated by (2).

$$T_{i+1} + T_{CP} > T_i + \mathrm{D(i, i+1)} + T_S \qquad (2)$$

While T_{CP} is the clock period. D(i, i+1) is the maximum propagation delay of the combinational logic stage i. T_S is the setup time which is the amount of time needed before the clock edge for the data input to be stable to make sure that the master latch of the flip-flop has saved the correct data. Satisfying this constraint guarantees enough time for flip-flop i+1 not to save an early incorrect version of the propagated data.

This research was partially funded by Zewail City of Science and Technology, AUC, the STDF, Intel, Mentor Graphics, and MCIT.

Clock signals reach flip-flops through clock distribution networks. These signals propagate through interconnection wires with different delays. Adjacent flip-flops may receive clock signals with different delays, as shown in Figure 2, and this is called clock skew.

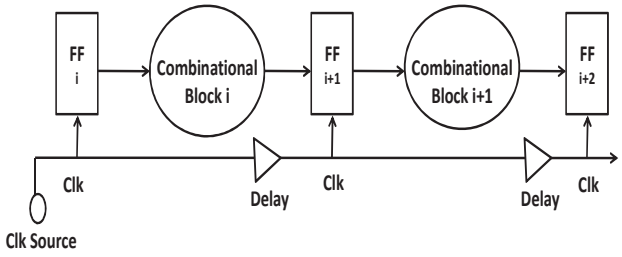


Figure 2. Clock skew of sequential circuits.

Unpredicted skewing may lead to receiving incorrect data. This skew should be taken into consideration while designing the sequential circuits to keep the two constraints satisfied. Process, voltage and temperature variations can lead to unpredicted run-time delays for both data and clock signals. These variations reduce the reliability of the manufactured chips leading to a reduction of the manufactured chips' yield. Conservative designs try to mitigate the negative impacts of variations by increasing the power budget or decreasing the clock frequency of the design to increase the safety margins. To increase the yield, many run-time techniques were introduced to mitigate the impacts of the variations. The target of these techniques is to ensure the correctness of the data captured by flip-flops and some of them can re-adjust the operating frequency and the supply voltage according to the existence of variations. Clock skewing was used intentionally to mitigate the thermal induced delay variations by using Self-Adjusting Clock Tree Architecture SACTA [2].

Soft-edge flip-flops were used to relax the timing among the adjacent combinational stages [3]. Pulsed latches with different pulse widths were used to allow time borrowing by stealing time among adjacent stages [4]. Razor shadow latch was used to check for the correctness of the data received and to flush the faulty stages of the pipeline [5]. Clock skewing was added to the shadow latch approach to allow stealing time among the adjacent stages of the pipeline [6]. The shadow latch approach compares the delayed data of the shadow latch to the data of the main flip-flop to detect the error, which is useful to compute the error rate. The error rate can be used as a feedback to re-adjust dynamically the supply voltage and the operating frequency of the circuit to reduce the error rate. Correction function approach, presented in this paper, is able to detect the error and correct the output without flushing the pipeline.

This paper is organized as the following: In section II, using correction functions to mitigate the potential variations is discussed. A 4x4 bit multiplier is introduced in section III as a case study to verify the validity of the correction function technique. The experimental results are discussed in Section IV. Section V presents the conclusion of the paper.

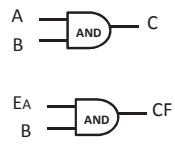
II. VARIABILITY MITIGATION USING CORRECTION FUNCTIONS

The different types of variations have great impacts on critical stages that have no enough slacks. Because of these variations, critical stages are expected to face time violation more than normal stages. Using shadow latches with delayed clocks to detect errors gives an indication about the error rate of the running circuit [5]. The error rate can be used to re-adjust the supply voltage and the operating frequency to avoid producing further errors. According to Razor shadow latch approach [5], if an error is detected the faulty stages of the pipeline will be flushed and the pipeline will be restarted after the faulty instruction. The correct data will be restored from the shadow latch. According to this approach two cycles are wasted to detect the error and restore the data. Flushing the pipeline decreases the overall throughput of the circuit. In case of high error rates, flushing the pipeline is costly. If the circuit has enough time to change its output after detecting an error, the pipeline will be able to continue its work without flushing. Preventing the pipeline flushing increases the overall throughput of the circuit and avoids wasting two clock cycles per every error.

Correction function technique is able to detect the error at the input of the combinational logic stage and correct the output of the stage according to the effect of the propagated incorrect input. The combinational logic stages depend on the binary system which depends on two logic states: high and low. The incorrect input is the inverted logic state of the correct input. The incorrect input may lead to inverting the output of the combinational stage according to its functionality. Inverting the output of any combinational logic stage because of the error at one of its inputs can be determined by what we call: Correction Function.

A Incorrect E_A=1	A Correct E_A=0	B Correct	C Out		
			E_A=0	E_A=1	
1	0	0	0	0	
1	0	1	0	1	Inverted
0	1	0	0	0	
0	1	1	1	0	Inverted

(a)



(b)

Figure 3. AND gate and its correction function: (a) Truth table, (b) Logic gates.

Figure 3 shows an AND gate, as a simple example of a combinational logic, which has an input A from critical unreliable stage. If an error is detected and the propagated value of A was wrong, the output of the AND gate C can be inverted if the output of the correction function CF is high. In case of error detection E_A = 1, the output C of AND gate should be inverted if the other input B =1 and the output will only depend on the value of input A which means that every change in A should result in a change in the output C.

As shown in Figure 4, there are two registers after the critical stage with two different clocks. The main register captures the input data at the rising edge of the normal clock of the circuit

while the shadow register captures a delayed version of the data at the rising edge of a delayed clock. The time interval between the two clocks shows the predicted delay induced by the variations of the critical stage. The data captured by the shadow register is compared to the data from the main register. In case of error detection (E=1), the correction function determines if the output of the current stage, which is a normal stage, will be inverted or not by adjusting the selection signal of the MUX.

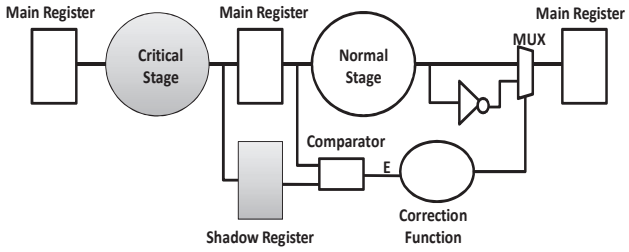


Figure 4. Two stages of a pipeline with error detection and correction function technique.

Only critical stages with potential variations should be treated and the correction function is only needed for the normal stage which is the successor of a faulty critical stage.

$$T_{Delay} + T_{Error\ Detection} +$$

$$T_{Correction\ Function} + T_{Selection} < T_{CP} \qquad (3)$$

As illustrated by (3), the time consumed by the delay variation, detecting the error, correction function and selecting the new output should not exceed the clock period, where T_{Delay} is the induced delay by the variations. $T_{Error\ Detection}$ is the propagation delay of the error detection circuit. $T_{Correction\ Function}$ is the propagation delay of the correction function. $T_{Selection}$ is the time needed by the MUX to choose the output and T_{CP} is the clock period of the system.

III. CASE STUDY: 4X4 BIT MULTIPLIER

To validate the correction function approach, a 4x4 bit multiplier is selected to be a case study. The two stages pipelined multiplier, shown in Figure 5, has two 4-bit input busses and one 8-bit output bus. We assumed that one of the two busses is unreliable because of potential variations, which means that there are 4 critical stages and 4 potential errors. This approach considers a single bit error detection and correction so that only one error can be detected and corrected each cycle.

The main components of the multiplier are: AND gates, Half Adders and Full Adders. Isolating the faulty input within the first stage of the multiplier reduces the cost of the correction functions. Stage 0 is assumed to be the critical stage and the inputs B0, B1, B2 and B3, from Stage 0, form the unreliable bus. The correction functions control the outputs of Stage 1. Only three types of correction functions are used to make decisions about inverting the outputs:

1- The correction function of the AND gate shown in (4).

$$CF_{AND} = E_{Bj}.A_i \qquad (4)$$

For example: i=0 and j=0 in Figure 5, while A_i and B_j are the inputs and E_{Bj} is the error signal of the input B_j.

2- The correction functions of the Half Adder shown in (5) and (6).

$$CF_{Half\ Adder\ Sum} = E_{Bj}.A_i + E_{Bl}.A_k \qquad (5)$$

$$CF_{Half\ Adder\ Carry} = E_{Bj}.A_i.A_k.B_l + E_{Bl}.A_i.B_j.A_k \qquad (6)$$

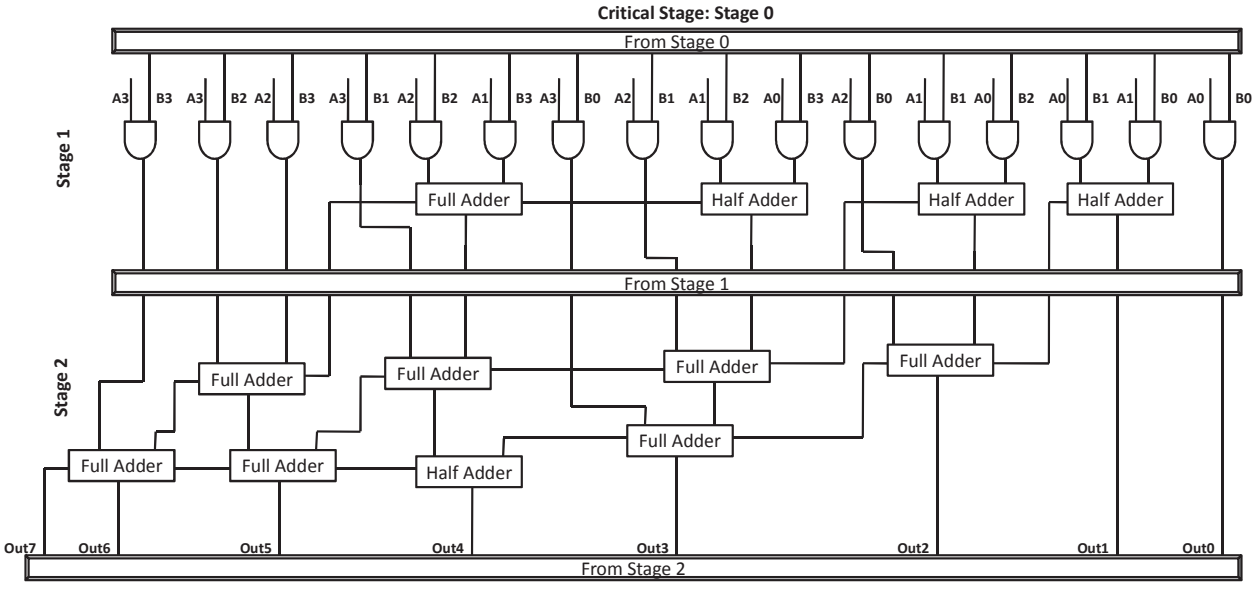


Figure 5. Two stages pipelined 4x4 bit multiplier.

For example: i=1, j=0, k=0 and l=1 in Figure 5, while (A_i.B_j) and (A_k. B_l) are the inputs. E_{Bj} is the error signal of the input B_j and E_{Bl} is the error signal of the input B_l.

3- The correction functions of the cascaded Full and Half Adders shown in (7) and (8).

$$CF_{\text{Full/Half Adder Sum}} =$$

$$E_{Bj}.\ (A_m \text{^} (A_i.\ A_k.\ B_l)) + E_{Bl}.A_k.!(A_i.B_j) \quad (7)$$

$$CF_{\text{Full/Half Adder Carry}} =$$

$$E_{Bj}.(A_i+A_m).(\ A_k.\ B_l) + E_{Bl}.(A_i+A_m).(A_k.B_j) \quad (8)$$

For example: i=0, j=2, k=1, l=3 and m=2 in Figure 5, while A_i, A_k, A_I, B_j and B_l are the inputs. E_{Bj} is the error signal of the input B_j and E_{Bl} is the error signal of the input B_l.

These three types are used to correct the outputs of Stage 1 shown in Figure 5, While the symbol (.) means AND, (+) means OR, (!) means NOT and (^) means XOR.

IV. EXPERIMENTAL RESULTS

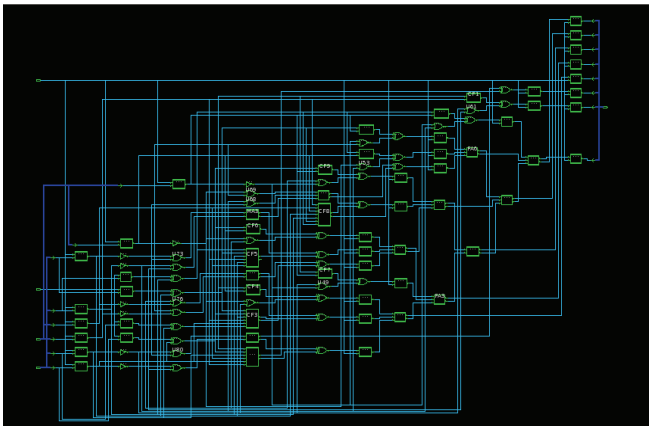


Figure 6. Schematic view of the two stages pipelined 4x4 bit multiplier with error detection circuits and correction functions.

The pipelined 4x4 bit multiplier and its correction functions were designed by Verilog. The functionality of the target design was verified and then the whole design was implemented by Synopsys Tools, using Design Vision with TSMC 90 nm technology as shown in Figure 6.

TABLE I shows that the area of the multiplier with the correction function approach is increased. The overhead of the multiplier with the correction function technique is 1.459x of the normal multiplier. The increase of consumed area is due to the addition of error detection circuits and correction functions. The consumed power is also increased and the overhead of the multiplier with correction function technique is 1.427x of the normal multiplier. The clock period is assumed to be 2 ns and it is selected according to the worst case of the critical stage. The long path of Stage 1 is increased because of the addition of the detection and correction components and the new long path will be 1.28 ns. Stage 1 and Stage 2 do not need the full clock cycle to finish their computations. The increase in the long path of Stage 1 has no negative impacts on the design because it is still satisfying the constraint mentioned in (3).

TABLE I. RESULTS OF APPLYING CORRECTION FUNCTIONS ON 4X4 BIT MULTIPLIER

4x4 bit Multiplier	**Clock Period = 2 nS**		
	Area (um)	***Power (uW)***	***Delay (nS)***
No Correction Functions	1349.227145	132.2491	1^{st} Stage: 1.01 2^{nd} Stage: 1.72
Correction Function	1967.826712	188.6692	1^{st} Stage: 1.28 2^{nd} Stage: 1.72

The long path of Stage 2 has not been changed because there are no correction functions for this stage. The induced delay of the critical stage T_{Delay} can be up to 0.72 ns as the rest of the clock period will be consumed by the error detection circuit and the correction function. The correction function technique has the flexibility to tolerate the induced delay up to 35% of the clock cycle without flushing the pipeline.

V. CONCLUSION

Using correction functions to mitigate the potential negative impacts of variations is an effective technique. If an error is detected, the pipeline will be able to correct the error at run-time. The correction function approach does not have to flush the pipeline and restore the correct data. The correction functions save the clock cycles consumed by flushing the pipeline and restoring the correct data. The overhead of area (1.459x) and power (1.427x) is related to the functionality of the multiplier which is only two pipeline stages. If there is a larger design with many pipeline stages, the overhead of power and area will be reduced as only the outputs of the critical stages will need detection and correction functions. Also full custom design can be used to optimize the area and power according to the needs of the design.

REFERENCES

[1] H. Sathyamurthy, S. S. Sapatnekar, and J. P. Fishburn, "Speeding Up Pipelined Circuits Through a Combination of Gate Sizing and Clock Skew Optimization", 1995 IEEE/ACM International Conference on Computer-Aided Design, 1995. ICCAD-95. Digest of Technical Papers, pp.467-470.

[2] J. Long, J. C. Ku, S. O. Memik, and Y. Ismail, "SACTA: A Self-Adjusting Clock Tree Architecture for Adapting to Thermal-Induced Delay Variation", IEEE Transactions on Very Large Scale Integration (VLSI) Systems, September 2010 ,Vol. 18, No. 9, pp.1323-1336.

[3] M. Wieckowski, Y. M. Park, C. Tokunaga, D. W. Kim, Z. Foo, D. Sylvester, D. Blaauw, "Timing Yield Enhancement Through Soft Edge Flip-Flop Based Design", IEEE Custom Integrated Circuits Conference, 2008, CICC 2008, September 2008, pp.543-546.

[4] S. Paik, S. Lee, and Y. Shin, "Retiming Pulsed-Latch Circuits with Regulating Pulse Width", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 30, No. 8, August 2011, pp.1114-1127.

[5] D. Ernst, N. S. Kim, S. Das, S. Pant, R. Rao, T. Pham, C. Ziesler, D. Blaauw, T. Austin, K. Flautner, and T. Mudge, "Razor: A Low-Power Pipeline Based on Circuit-Level Timing Speculation", In Proc. of the 36th Annual IEEE/ACM International Symposium on Micro-architecture, MICRO-36, December 2003, pp.7-18.

[6] S. Lee, S. Das, T. Pham, T. Austin, D. Blaauw, T. Mudge, "Reducing Pipeline Energy Demands with Local DVS and Dynamic Retiming", In Proc. of the 2004 International Symposium on Low Power Electronics and Design, ISLPED '04, August 2004, pp.319-324.