DESIGN OF AGING-AWARE EFFICIENT BOOTH MULTIPLIER USING ADAPTIVE HOLD LOGIC

K.venugopalarao[1], D.Harihara santosh[2],P.Sirish kumar[3]
[1](Electronics and Communication Engineering Department, AITAM,Tekkali ,India.
Email:venugopalrao8426@gmail.com)
[2] (Electronics and Communication Engineering Department, AITAM,Tekkali,India.
Email: dhhsantosh@gmail.com)
[3](Electronics and Communication Engineering Department, AITAM,Tekkali,India.
Email: sirishdg@gmail.com)

ABSTRACT: High speed and low consumption is one of the most important design objectives in integrated circuits. As multipliers are the most widely used components in such circuits, the multiplier must be designed efficiently. In this project the simple and efficient approach to reduce the maximum power consumption and delay, area in proposed. In this Existing system, negative bias temperature instability effect occurs when a pMOS transistor is under negative bias which increases the threshold voltage of the Pmos transistor and reduces the multiplier speed. Positive bias temperature in stability effect occurs when an nMOS transistor is under positive bias .both effects degrade transistor and in long term the system may fail due to the timing violations. New technique implements serial multiplier architecture with booth algorithm .In proposed system, design of aging –aware efficient booth multiplier using adaptive hold logic circuit is introduced. The AHL circuit achieves reliable operation under the influence of NBTI and PBTI effects with this proposed architecture 4x4 booth multiplier will developed and compare with contemporary architecture.

Keywords: AGING-EFFECT-INDICATOR, BOOTH MULTIPLIER, RAZOR FLIPFLOPS, AHL.

INTRODUCTION

Multiplication operation is one of the area which consuming more arithmetical operations in high performance circuits. As for importance many of the researchers deal with high speed multipliers of low power design. Multiplication operation contains two basic operations, one to generate partial products and another one to generate their sum and this performed using two types of multiplication algorithms parallel and serial. Where the Serial multiplication algorithms use sequential circuits with feedbacks, whereas the inner products are sequentially

Produced and the computed. Parallel multiplication algorithms often use combinational circuits and these never contain any feedback structures.

A multiplier is one of the key hardware blocks in most digital signal processing (DSP) systems. Typical DSP applications where a multiplier plays an important role include digital filtering, digital communications and spectral analysis. Many current DSP applications are targeted at portable, battery-operated systems, so that power dissipation becomes one of the primary design constraints. Since multipliers are rather complex circuits and must typically operate at a high system clock rate, reducing the delay of a multiplier is an essential part of satisfying the overall design. The aging behavior of digital designs is a major topic of research within the integrated circuits. There exists a large body of work that addresses aging effects at the hardware level. For example, Wu and Marculescu presented optimization techniques that allow synthesizing digital circuits that are less prone to aging degradation. In different process variations at a chip level are studied and characterized. Software techniques are presented to cope with problems that are posed by unreliable hardware. These include devising reliability-aware instruction sets, coupled with appropriate instruction scheduling using reliability aware compilation techniques. Recently, in Huang et al. presented an allocation framework based on a heuristic that aims at maximizing the lifetime of a SoC (System-on-Chip).Our work here follows this line of research and also deals with the allocation issues that arise in aging devices.

Aging-aware multiplier

The aging-aware multiplier architecture, which includes two *m*-bit inputs (*m* is a positive number), one 2*m*-bit output, one column- or row-bypassing multiplier, 2*m* 1-bit Razor flip-

flops, and an AHL L circuit.

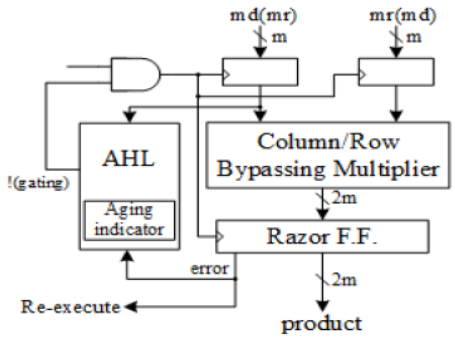


Fig.1 Aging aware architecture (md means multiplicand; mr means multiplicator).

Hence, the two aging-aware multipliers can be implemented using similar architecture, and the difference between the two bypassing multipliers lies in the input signals of the AHL. According to the bypassing selection in the column or row-bypassing multiplier, the input signal of the AHL in the architecture with the column-bypassing multiplier is the multiplicand, whereas of the row-bypassing multiplier is the multiplicator.

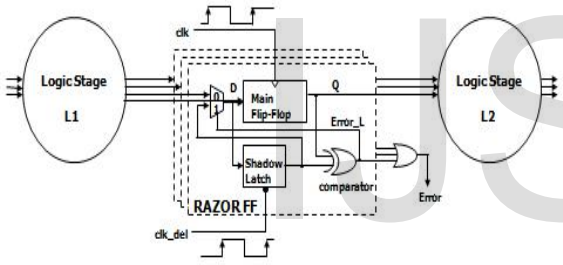


Fig.2 Razor flip flop architecture.

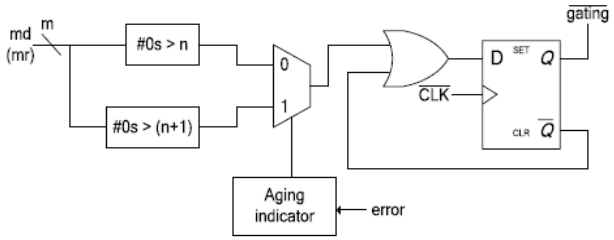


Fig.3 Diagram of AHL (md means multiplicand; mr means multiplication)

If the Razor flip-flops can be used to detect whether timing violations occur before the next input pattern arrives. A 1-bit Razor flip-flop contains a main flip-flop, shadow latch, XOR gate, and mux. The main flip-flop catches the execution result for the combination circuit using a normal clock signal, and the shadow latch catches the execution result using a delayed clock latched bit of the shadow latch is different from that of the main flip-flop, this means the path delay of the current operation exceeds the cycle period, and the main flip-flop catches an incorrect result. If errors occur, the Razor flip-flop will set the error signal to 1 to notify the system to re execute the operation and notify the AHL circuit that an error has occurred.

If not, the operation is re executed with two cycles. Although the re execution may seem costly, the overall cost is low because the re execution frequency is low. The AHL circuit is the key component in the aging-ware variable-latency multiplier. Fig.3 shows the details of the AHL circuit. The AHL circuit contains an aging indicator, two judging blocks, one mux, and one D flip-flop. The aging indicator indicates whether the circuit has suffered significant performance degradation due to the aging effect. The aging indicator is implemented in a simple counter that counts the number of errors over a certain amount of operations and is reset to zero at the end of those operations. If the cycle period is too short, the column- or row-bypassing multiplier is not able to complete these operations successfully, causing timing violations. These timing violations will be caught by the Razor flip-flops, which generate error signals.

Compared with the first judging block, the second judging block allows a smaller number of patterns to become one-cycle patterns because it requires more zeros in the multiplicand (multiplicator). The details of the operation of the AHL circuit are as follows: when an input pattern arrives, both judging blocks will decide whether the pattern requires one cycle or two cycles to complete and pass both results to the multiplexer. The multiplexer selects one of either result based on the output of the aging indicator. Then an OR operation is performed between the result of the multiplexer, and the Q^{-} signal is used to determine the input of the D flip-flop. When the pattern requires one cycle, the output of the multiplexer is 1.

The overall flow of the architecture is as follows: when input patterns arrive, the column- or row-bypassing multiplier, and the AHL circuit execute simultaneously. According to the number of zeros in the multiplicand (multiplicator), the AHL circuit decides if the input patterns require one or two cycles. If the input pattern requires two cycles to complete, the AHL will output 0 to disable the clock signal of the flip-flops. Otherwise, the AHL will output 1 for normal operations. When the column- or row-bypassing multiplier finishes the operation, the result will be passed to the Razor flip-flops. The Razor flip-flops check whether there is the path delay timing violation. If timing violations occur, it means the cycle period is not long enough for the current operation to complete and that the execution result of the multiplier is incorrect

Column-Bypassing Multiplier

The FAs in the AM are always active regardless of input states. In, a low-power column-bypassing multiplier

design is proposed in which the FA operations are disabled if the corresponding bit in the multiplicand is 0. Fig. 4 shows a 4×4 column-bypassing multiplier. Supposing the inputs are 1010_2 * 1111_2, it can be seen that for the FAs in the first and third diagonals, two of the three input bits are 0: the carry bit from its upper right FA and the partial product $a_i b_i$. Therefore, the output of the adders in both diagonals is 0, and the output sumbit is simply equal to the third bit, which is the sum output of its upper FA. Hence, the FA is modified to add two tristate gates and one multiplexer. The multiplicand bit a_i can be used as the selector of the multiplexer to decide the output of the FA, and $a_i$can also be used as the selector of the tristate gate to turn offthe input path of the FA. If $a_i$is 0, the inputs of FA are disabled, and the sum bit of the current FA is equal to the sum bit from its upper FA, thus reducing the power consumption of the multiplier. If $a_i$is 1, the normal sum result is selected.

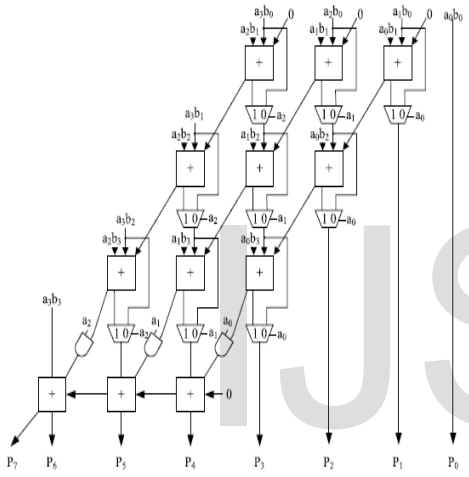


Fig.4 4 × 4 column-bypassing multiplier.

Adaptive hold Logic

The most critical concern in sub threshold circuits is to achieve high level of performance with very tight power constraints. This is evident in the development of mobile phones: in last one decade talk-time per gram of battery has improved by 60x. Challenges that prevent sub-threshold circuits from being widely used are their performances dependency on different Process Voltage and Temperature (PVT) conditions.

That is why the classical guard band methodology for "worst-case" is no more efficient, so some adaptive performance control techniques are required. Initially, the most critical paths of the circuits were replicated to track the correct functionality. Represents an application of adaptive performance control with replica circuit but, original critical path circuit and its replica part can't be identical from manufacturing point of view.To address these issues, different adaptive techniques were proposed.

Booth's multiplication algorithm

Booth's multiplication algorithm is a multiplication algorithm that multiplies two signed binary numbers in two's complement notation. The algorithm was invented by Andrew Donald Booth. Booth used desk calculators that were faster at shifting than adding and created the algorithm to increase their speed. Booth's algorithm is of interest in the study of computer architecture.

Booth's algorithm examines adjacent pairs of bits of the *N*-bit multiplier *Y* in signed two's compslement representation, including an implicit bit below the least significant bit, $y_{-1} = 0$. For each bit y_i, for *i* running from 0 to *N*-1, the bits y_i and y_{i-1} are considered. Where these two bits are equal, the product accumulator *P* remains unchanged. Where $y_i = 0$ and $y_{i-1} = 1$, the multiplicand times 2^i is added to *P*; and where $y_i = 1$ and $y_{i-1} = 0$, the multiplicand times 2^i is subtracted from *P*. The final value of *P* is the signed product.

The representation of the multiplicand and product are not specified; typically, these are both also in two's complement representation, like the multiplier, but any number system that supports addition and subtraction will work as well. As stated here, the order of the steps is not determined. Typically, it proceeds from LSB to MSB, starting at $i = 0$; the multiplication by 2^i is then typically replaced by incremental shifting of the *P* accumulator to the right between steps; low bits can be shifted out, and subsequent additions and subtractions can then be done just on the highest *N* bits of *P*. There are many variations and optimizations on these details.

The algorithm is often described as converting strings of 1's in the multiplier to a high-order +1 and a low-order –1 at the ends of the string. When a string runs through the MSB, there is no high-order +1, and the net effect is interpretation as a negative of the appropriate value.

Booth's algorithm can be implemented by repeatedly adding (with ordinary unsigned binary addition) one of two predetermined values *A* and *S* to a product *P*, then performing a rightward arithmetic shift on *P*. Let **m** and **r** be the multiplicand and multiplier, respectively; and let *x* and *y* represent the number of bits in **m** and **r**.

1. Determine the values of *A* and *S*, and the initial value of *P*. All of these numbers should have a length equal to $(x + y + 1)$.

1. A: Fill the most significant (leftmost) bits with the value of **m**. Fill the remaining (y + 1) bits with zeros.
2. S: Fill the most significant bits with the value of (−**m**) in two's complement notation. Fill the remaining (y + 1) bits with zeros.
3. P: Fill the most significant x bits with zeros. To the right of this, append the value of **r**. Fill the least significant (rightmost) bit with a zero.

2. Determine the two least significant (rightmost) bits of P.
   1. If they are 01, find the value of $P + A$. Ignore any overflow.
   2. If they are 10, find the value of $P + S$. Ignore any overflow.
   3. If they are 00, do nothing. Use P directly in the next step.
   4. If they are 11, do nothing. Use P directly in the next step.
3. Arithmetically shift the value obtained in the 2nd step by a single place to the right. Let P now equal this new value.
4. Repeat steps 2 and 3 until they have been done y times.
5. Drop the least significant (rightmost) bit from P. This is the product of **m** and **r**.

Example

Find 3 × (−4), with **m** = 3 and **r** = −4, and x = 4 and y = 4:

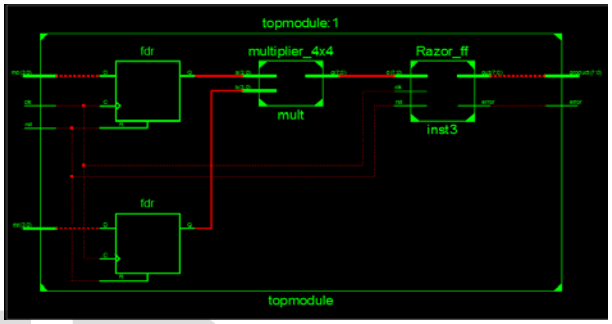
- m = 0011, -m = 1101, r = 1100
- A = 0011 0000 0
- S = 1101 0000 0
- P = 0000 1100 0

- Perform the loop four times :
  1. P = 0000 110**0 0**. The last two bits are 00.
     - P = 0000 0110 0. Arithmetic right shift.
  2. P = 0000 011**0 0**. The last two bits are 00.
     - P = 0000 0011 0. Arithmetic right shift.
  3. P = 0000 001**1 0**. The last two bits are 10.
     - P = 1101 0011 0. P = P + S.
     - P = 1110 1001 1. Arithmetic right shift.
  4. P = 1110 100**1 1**. The last two bits are 11.
     - P = 1111 0100 1. Arithmetic right shift.

- The product is 1111 0100, which is −12.

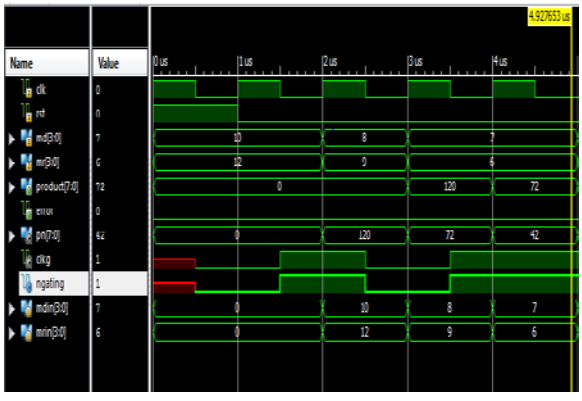
4 RESULTS

RTL SCHEMATIC FOR THE TOP MODULE:



The RTL SCHEMATIC gives the information about the user view of the design.The internal blocks contains the basic gate representaion of the logic.These basic gate realization is purely depend upon the corresponding FPGA selection and the internal database information.

OUTPUT WAVE FORM FOR THE TOP MODULE:



Outputs force In the waveform which is shown above, clk signal represents clock, rst signal represents reset, md represents multiplicand, mr represents multiplier which we are applying as inputs to the design. Similarly product is the output signal for the design. Here clock signal is generated

for the positive edge. Initially the reset signal should be force to logic 1 and after one clock cycle made it to logic 0 for performing the corresponding functional operation. To obtain the required the inputs logic with the required values. The output product gets the multiplicity value of the applied inputs md and mr.

Existed Design summary report:

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	37	4656	0%
Number of Slice Flip Flops	24	9312	0%
Number of 4 input LUTs	72	9312	0%
Number of bonded IOBs	19	232	8%
Number of GCLKs	1	24	4%

Proposed Design summary report:

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	21	4656	0%
Number of Slice Flip Flops	24	9312	0%
Number of 4 input LUTs	41	9312	0%
Number of bonded IOBs	19	232	8%
Number of GCLKs	1	24	4%

From the above tables, the power and area are less for the proposed Aging aware Booth multiplier than the existing Aging aware column multiplier.

5 CONCLUSION

Aging problem of transistors has a significant effect on performance of these systems and in long term, the system may fail due to delay problems, which can be reduced by using over-design approaches**.** This project proposed an aging-aware variable-latency multiplier design with the AHL. The multiplier is able to adjust the AHL to mitigate performance degradation due to increased delay. The experimental results show that our proposed architecture with 4x4 multiplication using Booth mathematical approach results to the area and power efficient design compared to the existing column bypass multiplier. The Verilog language is used for coding. The synthesis and simulation is carried out using Xilinx ISE 12.3i.

REFERENCES

[1] Y. Cao. (2013). *Predictive Technology Model (PTM) and NBTI Model* [Online]. Available: http://www.eas.asu.edu/~ptm

[2] S. Zafar*et al.*, "A comparative study of NBTI and PBTI (charge trapping) in SiO2/HfO2 stacks with FUSI, TiN, Re gates," in *Proc.IEEESymp. VLSI Technol. Dig. Tech. Papers*, 2006, pp. 23–25.

[3] S. Zafar, A. Kumar, E. Gusev, and E. Cartier, "Threshold voltage instabilities in high-k gate dielectric stacks," *IEEE Trans. Device Mater.Rel.*, vol. 5, no. 1, pp. 45–64, Mar. 2005.

[4] H.-I. Yang, S.-C.Yang, W. Hwang, and C.-T. Chuang, "Impacts of NBTI/PBTI on timing control circuits and degradation tolerant design in nanoscale CMOS SRAM," *IEEE Trans. Circuit Syst.*, vol. 58, no. 6, pp. 1239–1251, Jun. 2011.

[5] R. Vattikonda, W. Wang, and Y. Cao, "Modeling and miimization of pMOS NBTI effect for robust naometer design," in *Proc. ACM/IEEEDAC*, Jun. 2004, pp. 1047–1052.

[6] H. Abrishami, S. Hatami, B. Amelifard, and M. Pedram, "NBTI-aware flip-flop characterization and design," in *Proc. 44th ACM GLSVLSI*, 2008, pp. 29–34.

[7] S. V. Kumar, C. H. Kim, and S. S. Sapatnekar, "NBTI-aware synthesis of digital circuits," in *Proc. ACM/IEEE DAC*, Jun. 2007, pp. 370–375.

[8] A. Calimera, E. Macii, and M. Poncino, "Design techniqures for NBTItolerant power-gating architecture," *IEEE Trans. Circuits Syst., Exp.Briefs*, vol. 59, no. 4, pp. 249–253, Apr. 2012.

[9] K.-C. Wu and D. Marculescu, "Joint logic restructuring and pin reordering against NBTI-induced performance degradation," in *Proc. DATE*, 2009, pp. 75–80.

[10] Y. Lee and T. Kim, "A fine-grained technique of NBTI-aware voltage scaling and body biasing for standard cell based designs," in *Proc. ASPDAC*, 2011, pp. 603–608.

About The Authors

Mr.K.venugopalarao is presently pursuing his M.Tech in VLSI system designing in Electronics and Communication Engineering Department,

AITAM, Tekkali. His areas of interest VLSI system designing. He has attended for one international conference. The author may be reached at venugopalrao8426@gmail.com.

D.Harihara santosh is presently working as Associate Professor in the Department of Electronics and Communication Engineering, Aditya Institute of Technology And Managament,Tekkali, Srikakulam District,India.Mr.santosh obtained B.Tech.from JNTU Hyderabad and M.Tech from JNTU Hyderabad in 2005 and 2010 respectively. He is pursuing his Ph.D signal processing J. N. T. U. College of Engineering, Hyderabad, India. He has 11 years teaching experience .He handled various subjects at Undergraduate and Post Graduate levels. He published more than ten technical papers in National and International journals andconferences. He has membership in EITE, IACSIT. The author may be reached at dhhsantosh@gmail.com.

P.Sirish kumar is presently working as Assistant Professor in Electronics and Communication Engineering Department, AITAM, Tekkali. He is pursuing his Ph.D in GITAM University, Visakhapatnam. He completed his M.Tech from JNTU Kakinada in the Dept. of Electronics and Communication Engineering with specialization VLSISD. He has 6 experience years in teaching. The author may be reached at sirishg@gmail.com.