High Speed Multiplier Design with Improved Adaptive Hold Logic in FPGA

Deepika Malviya[1], Prof. Deepa Gianchandani[2]

[1]M-Tech Research Scholar, [2]Research Guide

Deptt. of Electronics & Communication, SISTec, Bhopal

Abstract - The designing of a multiplier with better device utilization will facilitates the various digital device to overcome from the high density of logic on the integrated devices. The integrated device suffers with NBTI and PBTI due to CMOS semiconductor properties and it affects the working of different logic operations and in the same context here we have taken multiplier for consideration and working to develop delay efficient multiplier with aging aware design using adaptive hold logic which is modified in this work to reduce effective delay to speedup circuit logic.

Keywords-AHL, Multiplier, Aging Effect, NBTI, PBTI, Delay Efficient.

I. INTRODUCTION

Multipliers are key segments of numerous high performance systems, for example, FIR filters, chip, digital signal processors, and so forth. Performance of Execution the system is generally determined by evaluation of the performance of the multiplier Execution in light of the fact that the multiplier is generally the slowest
bigger ranges. Thus, an entire range of multipliers with various zone speed imperatives have been outlined with completely parallel. Multipliers toward one side of the range and completely serial multipliers at the flip side. In the middle of are digit serial multipliers where single digits comprising of a few bits are worked on. These multipliers have direct execution in both speed and range. In any case, existing digit serial multipliers have been Plagued by confounded switching frameworks and additionally inconsistencies in outline. Radix 2An multipliers which work on digits in a parallel mold rather than bits convey the pipelining to the digit level and stay away from the vast majority of the above issues.

Many DSP applications request high throughput and continuous response, execution limitations that regularly direct one of kind structures with elevated amounts of simultaneousness. DSP originators require the ability to control and assess complex calculations to remove the important level of simultaneousness. Execution limitations can likewise be tended to by applying elective advancements. A change at the usage level of plan by the addition of another innovation can frequently make reasonable a current peripheral calculation or engineering.

A. Adders

In electronics, an adder is a digital circuit that performs addition of numbers. In modem computers adders reside in the arithmetic logic unit (ALU) where other operations are performed. Although adders can be constructed for many numerical representations, such as Binary-coded decimal or excess-3, the most common adders operate on binary numbers. In cases where two's complement is being used to represent negative numbers it is trivial to modify an adder into an adder-subtracter.

For single bit adders, there are two general types.

A half adder has two inputs, generally labeled A and B, and two outputs, the sum S and carry C. S is the two-bit XOR of A and B, and C is the AND of A and B. Essentially the output of a half adder is the sum of two one-bit numbers, with C being the most significant of these two outputs.

block in the system. Besides, it is fo

The second type of single bit adder is the full adder. The full adder takes into account a carry input such that multiple adders can be used to add larger numbers. To remove ambiguity between the input and output carry lines, the carry in is labeled C_i or C_{in} while the carry out is labeled C_o or C_{out}.

B. Binary Multiplier

A Binary multiplier is an electronic equipment device utilized as a part of digital hardware or a PC or other electronic device to perform fast duplication of two numbers in binary representation. It is constructed utilizing binary adders.

For outlining a multiplier circuit we ought to have hardware to give or do the accompanying three things:

1. It ought to be fit distinguishing whether a bit is 0 or 1.
2. It ought to be fit for moving left partial products.
3. It ought to have the capacity to add all the partial products to give the products as entirety of partial products.

4. It ought to inspect the sign bits. In the event that they are similar, the indication of the product will be a positive, if the sign bits are inverse product will be negative. The sign piece of the product put away with above criteria ought to be shown alongside the product.

From the above dialog we watch that it is not important to hold up until all the partial products have been shaped before summing them. Actually the addition of partial product can be completed when the partial product is shaped.

Notations:

a - Multiplicand

b - Multiplier

p – Product

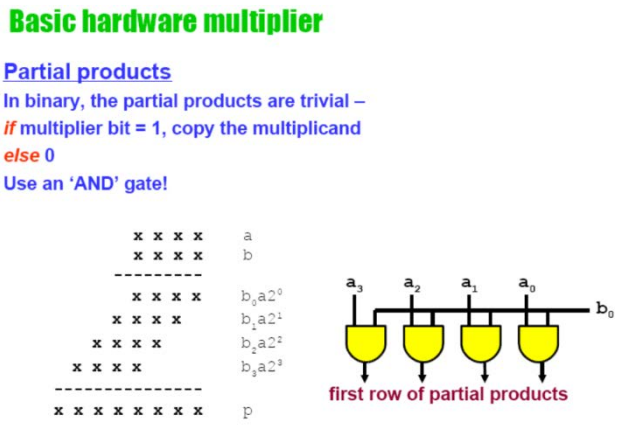

Figure 1.1 Basic Hardware multiplier

II. AGING-AWARE TIMING ANALYSIS

It is very expensive and time consuming to process a chip. Therefore, it is not feasible to iteratively design a chip by processing it, testing it and making design changes. In fact, the IC industry is quite unique by heavily relying on abstract models for designing a product. There are, for instance, transistor models to simulate the voltage and current waveforms on circuit level; or gate models, which provide, in addition to other things, the delay of the standard cells. The objective of models is to give all the data that is fundamental on a specific deliberation level and overlook immaterial data. Only by abstraction it is possible to design state- of-the-art circuits with up to billions of transistors. The models must be as accurate as possible to provide a good prediction for the performance, power and area of a design. Otherwise the final product might not meet the specification. Basically, the gate and wire delays along the longest, the so called critical path, are added up and it is verified whether the resulting circuit delay fulfils the timing specification, or not. When a circuit ages, the gate delays increase and the circuit may violate the timing specification although the specifications were met right after manufacturing .

TA is required in many design flow steps, not just for the final timing sign-off. This enables the consideration of timing at every synthesis step and the synthesis tool can optimize the design until the timing constraints are met. With each synthesis step, the available information is getting more accurate which in turn increases the accuracy of the TA. Only the multi-level logic functions are known at the logic synthesis stage and the circuit delay can only be roughly estimated by the logic depths of those functions. During technology mapping it is first known which gates from the standard cell library are instantiated. From this step on aging can be considered by an aging-aware gate model. The exact net length is available during the place and route synthesis stage, which increases the accuracy of the TA by knowing the parasitic capacitance and resistance of the nets. Finally, the coupling capacitances are available for timing sign-off, which again increases the accuracy of the TA. Hence, an aging-aware TA is beneficial at all synthesis steps from technology mapping on.

III. PROPOSED ARCHITECTURE

In the proposed aging – aware reliable multiplier design. Presented the overall architecture and functioning. the architecture is simulated on result has taken from the Xilinx platform.

The description of proposed architecture is given below.

Figure 3.1 shows the proposed architecture reliable aging-aware multiplier hold logic (AHL).which includes two inputs of m bit (m is a positive integer) and two out puts , in the proposed architecture as illustrated in figure 3.1 the RTL schematic diagram .

There are 4 sections in a proposed architecture having a latch, a shadow latch, a flip flop and two arrays of multipliers 0 and 1 correspondingly. The outcome has taken from a hop(31.0).,a(15.0) and b(15.0)are the inputs

Proposed Architecture

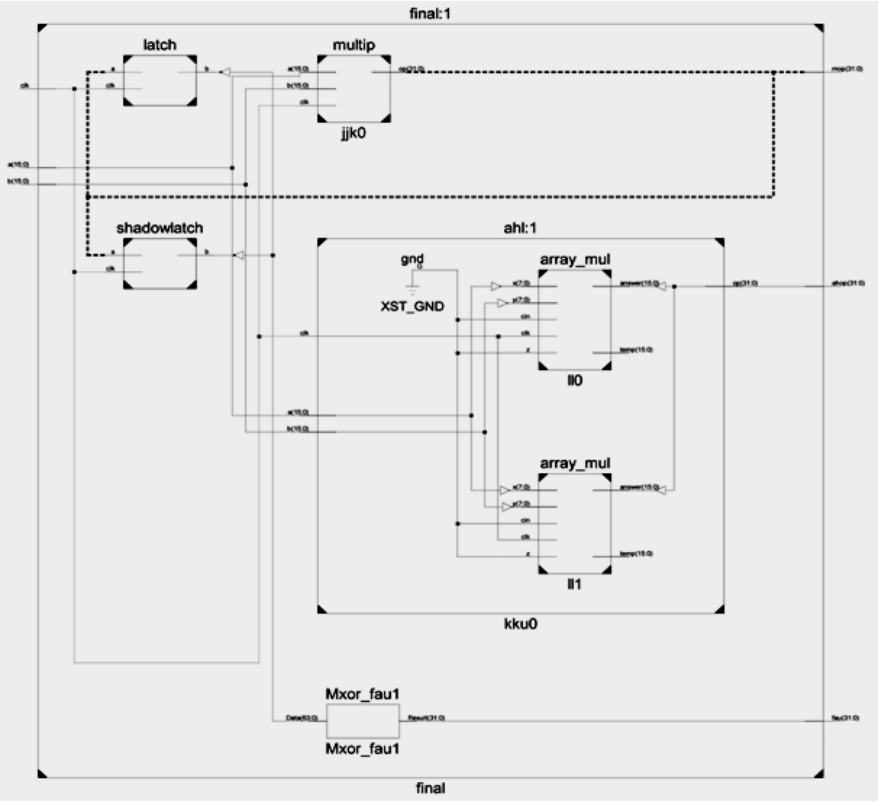

Figure: 3.1 RTL Schematic of Proposed Architecture.

```
Timing Details:
---------------
All values displayed in nanoseconds (ns)
=========================================================================
Timing constraint: Default period analysis for Clock 'clk'
  Clock period: 0.717ns (frequency: 1393.728MHz)
  Total number of paths / destination ports: 640 / 224
-------------------------------------------------------------------------
                                  Gate     Net
    Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
     FDR:C->Q              2   0.239   0.438  kku0/ll1/xx17/x (kku0/ll1/xx17/x)
     LUT3:I0->O            1   0.040   0.000  kku0/ll1/FA2/kku0/ll1/FA2/a_c_OR_1_o)
     FD:D                     -0.003          kku0/ll1/FA2/carry
    ----------------------------------------
    Total                     0.717ns (0.279ns logic, 0.438ns route)
                                      (38.9% logic, 61.1% route)
=========================================================================
  Data Path: a<0> to kku0/ll1/xx1/x
                                  Gate     Net
    Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
     IBUF:I->O             1   0.003   0.279  a_0_IBUF (a_0_IBUF)
     INV:I->O              9   0.053   0.316  kku0/ll1/xx1/kku0/ll1/xx1/a_inv)
     FDR:R                     0.334          kku0/ll1/xx1/x
    ----------------------------------------
    Total                     0.986ns (0.390ns logic, 0.596ns route)
                                      (39.6% logic, 60.4% route)

=========================================================================
                                  Gate     Net
    Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
```

Figure: 3.2 Timing Summaries

IV. SYNTHESIS OUTCOMES

The simulation of experiments are conducted in Xilinx IDE.

Timing summery

In figure 4.1 timing details of device in nanosecond (ns) which shows the time taken to execute logic.

Table 4.1: Performance Comparison of Proposed Architecture with Existing Architecture with SPARTAN 3

Parameters	Proposed Architecture (Improved AHL)	Base Paper Architecture (Using AHL)
Delay	**12.2 ns (Approx 50% improvement)**	24.965 ns
Slices	199 (25%)	167 (21%)
4 input LUTs	350 (22%)	292 (19%)
Bonded IOBs	129 (104%)	66 (53%)

Table 4.2: Performance Comparison of Proposed Architecture with Existing Architecture with SPARTAN 6

Parameters	Proposed Architecture (Improved AHL)	Base Paper Architecture (Using AHL)
Delay	**8.499 ns (Approx 50% improvement)**	17.962 ns
Slices LUTs	240	190
Bonded IOBs	129	66
Power	81 mW	81 mW

The performance evolutions and comparison is shown in table 4.1 and 4.2 with proposed architecture to existing architecture table 5.1 shows comparison with SPARTAN 3 and table 5.2 shows comparison with SPARTAN 6.

V. CONCLUSION AND FUTURE SCOPE

The synthesis of aging aware delay efficient multiplier design using modified adaptive hold logic is explained and synthesized in this work. The results show the delay outcomes of the architecture with existing results and improved results. After synthesis of proposed architecture the overall delay of the device is 12.2ns which is 50% lower than the existing architecture delay i.e. 24.965ns. In the synthesis outcomes it is clear that the modified approach for delay efficient multiplier design is faster than the previous architectures. The comparative analysis of synthesis results clearly concludes that the proposed 16 bit architecture 50% faster than the previous architecture and it is better prone to NBTI and PBTI effects which slow down the calculations of digital circuits. So that for the future up gradation in logic circuits proposed multiplier design will be useful to speed up the calculations and save time. This architecture can also be upgraded using logic and architecture level optimization approaches for future evolution.

REFERENCES

[1] I. C. Lin, Y. H. Cho and Y. M. Yang, "Aging-Aware Reliable Multiplier Design With Adaptive Hold Logic," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 23, no. 3, pp. 544-556, March 2015.

[2] I. C. Wey, C. C. Peng and F. Y. Liao, "Reliable Low-Power Multiplier Design Using Fixed-Width Replica Redundancy Block," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 23, no. 1, pp. 78-87, Jan. 2015.

[3] H. Kükner et al., "Degradation analysis of datapath logic subblocks under NBTI aging in FinFET technology," Fifteenth International Symposium on Quality Electronic Design, Santa Clara, CA, 2014, pp. 473-479.

[4] S. Ganapathy, R. Canal, A. González and A. Rubio, "iRMW: A low-cost technique to reduce NBTI-dependent parametric failures in L1 data caches," 2014 IEEE 32nd International Conference on Computer Design (ICCD), Seoul, 2014, pp. 68-74.

[5] A. Calimera, M. Loghi, E. Macii and M. Poncino, "Dynamic Indexing: Leakage-Aging Co-Optimization for Caches," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 33, no. 2, pp. 251-264, Feb. 2014.

[6] Rahimi, L. Benini and R. K. Gupta, "Aging-aware compiler-directed VLIW assignment for GPGPU architectures," 2013 50th ACM/EDAC/IEEE Design Automation Conference (DAC), Austin, TX, 2013, pp. 1-6.

[7] Y. H. Cho, I. C. Lin and Y. M. Yang, "Aging-aware reliable multiplier design," 2012 IEEE International SOC Conference, Niagara Falls, NY, 2012, pp. 322-327.

[8] K.-C. Wu and D. Marculescu, "Joint logic restructuring and pin reorder¬ing against NBTI-induced performance degradation," in Proc. DATE,2009, pp. 75-80.

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

[10] M. Basoglu, M. Orshansky, and M. Erez, "NBTI-aware DVFS: A new approach to saving energy and increasing processor lifetime," in Proc. ACM/IEEE ISLPED, Aug. 2010, pp. 253-258.

[11] K.-C. Wu and D. Marculescu, "Aging-aware timing analysis and opti¬mization considering path sensitization," in Proc. DATE, 2011, pp. 1-6.

[12] K. Du, P. Varman, and K. Mohanram, "High performance reli¬able variable latency carry select addition," in Proc. DATE, 2012, pp. 1257-1262.

[13] A. K. Verma, P. Brisk, and P. Ienne, "Variable latency speculative addition: A new paradigm for arithmetic circuit design," in Proc. DATE, 2008, pp. 1250-1255.
[14] D. Baneres, J. Cortadella, and M. Kishinevsky, "Variable-latency design by function speculation," in Proc. DATE, 2009, pp. 1704-1709.
[15] Y.-S. Su, D.-C. Wang, S.-C. Chang, and M. Marek-Sadowska, "Per¬formance" optimization using variable-latency design style," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 19, no. 10, pp. 1874-1883, Oct. 2011.