Hybrid Partial Product Based on Low Power And Area Efficient Approximate Multipliers

Aleema.A [1], Mr. G.Prabhakaran [2]
[1, 2] Dept of ECE
[1, 2] Nandha Engineering College

Abstract- *Proposed efficient approximate multipliers, partial products of the multiplier are modified using generate and propagate signals. Approximation is applied using simple OR gate for altered generate partial products. Approximate half-adder, full-adder, and 4-2 compressor are proposed to reduce remaining partial products. Two variants of approximate multipliers are proposed, where approximation is applied in all n bits in Multiplier1 and only in n −1 least significant part in Multiplier2. Multiplier1 and Multiplier2 achieve significant reduction in area and power consumption compared with exact designs. They are also found to have better precision when compared to existing approximate multiplier designs. The proposed multiplier designs can be used in applications with minimal loss in output quality while saving significant power and area.*

Keywords- approximate multipliers, partial products, Approximate half-adder, full-adder, and 4-2 compressor, approximate multiplier designs

I. INTRODUCTION

Multipliers form an important hardware block in DSP and Embedded applications. Multiplication speed determines the processor speed. So high speed multipliers are needed in the processors for many applications. For increase the speed of multiplication different algorithms are used. Multiplication is the most commonly used operation in many computing systems. With the advances in technology , many researchers have tried and are trying to design multipliers which offer either of the following design targets high speed , low power consumption , regularity of layout and hence less area or even combination of them in one multiplier thus making them suitable for the various high speed, low power consumption and compact VLSI implementation. A multiplier can be divided into three stages: Partial products generation stage, partial products addition stage, and the final addition stage. In the first stage, the multiplicand and the multiplier are multiplied bit by bit to generate the partial products. The second stage is the most important, as it is the most complicated and determines the speed of the overall multiplier. The 4-2 and 5-2 compressors have been widely employed in the high speed multipliers to lower the latency of the partial product accumulation stage.

The common multiplication method, add and shift algorithm. In parallel multipliers number of partial products to be added is the main parameter that determines that performance of the multiplier. To reduce the number of partial products to be added, modified Booth algorithms one of the most popular algorithm is one of the most popular algorithm. To achieve speed improvements Wallace Tree algorithm can be used to reduce the number of sequential adding stages. Further by combining both the algorithms we can see advantage of the algorithms in one multiplier .However with increasing parallelism, the amount of shifts between the partial products and the intermediate sums to be added will increase which may result in reduced speed increase in silicon area due to the irregularity of structure and also increased power consumption due to the increase in the interconnect resulting from complex routing. On the other hand serial and parallel multipliers compromise speed to achieve better performance for area and power consumption. The selection of the parallel or serial multiplier actually depends on the nature of the application. In order to satisfy the requirement of small area low power high throughput circuitries compressors are used.

Parallel compressors are a proper subclass of parallel counters. A compressor is distinctively characterized by the set of inputs and outputs that serve as interconnection between different packages in a one-dimensional array of compressors. These extra "side" inputs and outputs are difficult to describe with an arithmetic expression or some formal and easy to understand structure. The following informal definition is introduced for reasons of simplicity. An (n, m, r, s) compressor is a combinatorial circuit capable of compressing n input summands to m output summands, when connected as a 1-dimensional array. It takes n r-bit slices of input summands, and produces m r-bit slices of output summands. There are outputs that are connected to less significant compressors in the one-dimensional array for multiple summand compression. Similarly, each compressor has s inputs supplied from more significant units in the array. The side outputs are divided into groups of r outputs, starting with the most significant output. Naturally, there are such groups.

The least significant group contains exactly output if s is not a multiple of r. The most significant group is connected to the first (most significant) of the less significant compressors in the array. The second most significant group is connected to the second less significant compressor and so on. Large parallel counters having a multitude of inputs are very useful in many arithmetic operations, and particularly in associative processing. Foster and Stockton have described a method for generating large parallel counters with a network of full adders, as have Muller and Preparation using an elegant algorithm. Recently, Kobayashi and Ohara extended the method by using smaller parallel counters for implementing larger ones. The design of large parallel counters using only two types of circuits: a compressor and an added.

Two well known fast multipliers are those presented by Wallace and DADDA. Both of these multipliers consist of three stages. In the first stage, partial product matrix is formed. In the second stage, partial product matrix is reduced to a height of two. In the final stage, these two rows of partial products are combined using carry propagation adder. In the Wallace method the partial products are reduced as soon as possible. But in DADDA multiplier perform the minimum reduction at each level. Wallace and DADDA multipliers use full adders and halfadders in their reduction stage. In the Wallace multiplier partial product reduction is performed as soon as possible. So the number of half adders and full adders required for the partial product reduction in Wallace multiplier is high. But the half adder does not reduce the number of partial product bits. So a modification to the Wallace reduction is presented which reduce the number of half adders. But in DADDA multiplier less number of half adders are required than the Wallace and Modified Wallace multiplier. For an N bit Wallace multiplier the number of half adders required is N1.5 and for an N bit DADDA multiplier the number of half adders required is N-1. For the final carry propagation adder, DADDA multipliers are the refinement of parallel multipliers first presented by Wallace in 1964. In contrast to the Wallace reduction DADDA multiplier perform the least reduction at each stage.

II. LITERATURE SURVEY

Low power is an imperative requirement for portable multimedia devices employing various signal processing algorithms and architectures. In most multimedia applications, human beings can gather useful information from slightly erroneous outputs. Therefore, do not need to produce exactly correct numerical outputs. Previous research in this context exploits error resiliency primarily through voltage over scaling, utilizing algorithmic and architectural techniques to mitigate the resulting errors. Propose logic complexity reduction at the transistor level as an alternative approach to take advantage of the relaxation of numerical accuracy. Demonstrate this concept by proposing various imprecise or approximate full adder cells with reduced complexity at the transistor level, and utilize them to design approximate multi-bit adders. In addition to the inherent reduction in switched capacitance, our techniques result in significantly shorter critical paths, enabling voltage scaling. Design architectures for video and image compression algorithms using the proposed approximate arithmetic units and evaluate them to demonstrate the efficacy of our approach. Also derive simple mathematical models for error and power consumption of these approximate adders. Furthermore, demonstrate the utility of these approximate adders in two digital signal processing architectures with specific quality constraints. Simulation results indicate up to 69% power savings using the proposed approximate adders, when compared to existing implementations using accurate adders.

In signal processing, Multiplication is frequently required. Parallel multipliers typically require a large amount of area, so in some cases, truncation is employed to reduce the area. This introduces error in the product. Recently, schemes have been introduced to add a constant to the truncated product to reduce the expected error. However, these schemes do not take into account the actual data and hence the outputs of the multiplier may have large errors for some input data patterns. This introduces a simple data-dependent method for varying the term added to reduce the actual error. ?he Variable Correction Truncated Multiplier is introduced in this paper. This is a method for minimizing the error of a truncated multiplier. The error is reduced by using information from the partial product bits of the column adjacent to the truncated LSB. This results in a complexity & savings while introducing minimum distortion to the result. In many signal processing applications, the data path is of a fixed size, and often the input data are less accurate than the accuracy of the data path. In this case, the output of a multiplier may not need to be accurate to the LSB of the data path.

The existing scientific and engineering problems are solved using deterministic, precise, and explicit models and algorithms. These rigorous techniques that are known as hard-computing are normally used to solve a category of problems with similar properties such as high precision, predictability, and strict analysis. These well-known and well-developed techniques are widely used to solve some of the existing engineering problems. However; these are in conflict with two frequent observations. The first is that most of the real-world applications and problems are naturally imprecise and with low degrees of certainty while the second one is that using higher precision than the minimum necessary imposes higher implementation cost. The conventional digital hardware computational blocks with different structures are designed to compute the precise results of the assigned calculations. The main contribution of our proposed Bio-inspired Imprecise Computational blocks (BICs) is that they are designed to provide an applicable estimation of the result instead of its precise value at a lower cost. These novel structures are more efficient in terms of area, speed, and power consumption with respect to their precise rivals. Complete descriptions of sample BIC adder and multiplier structures as well as their error behaviors and synthesis results are introduced in this paper. It is then shown that these BIC structures can be exploited to efficiently implement a three-layer face recognition neural network and the hardware defuzzification block of a fuzzy processor.

Computer arithmetic applications are implemented using digital logic circuits, thus operating with a high degree of reliability and precision. However, many applications such as in multimedia and image processing can tolerate errors and imprecision in computation and still produce meaningful and useful results. Accurate and precise models and algorithms are not always suitable or efficient for use in these applications. The paradigm of inexact computation relies on relaxing fully precise and completely deterministic building modules when for example, designing energy-efficient systems. This allows imprecise computation to redirect the existing design process of digital circuits and systems by taking advantage of a decrease in complexity and cost with possibly a potential increase in performance and power efficiency. Approximate computing relies on using this property to design simplified, yet approximate circuits operating at higher performance and/or lower power consumption compared with precise logic circuits. Inexact computing is an attractive paradigm for digital processing at nanometric scales. Inexact computing is particularly interesting for computer arithmetic designs. This paper deals with the analysis and design of two new approximate 4-2 compressors for utilization in a multiplier. These designs rely on different features of compression, such that imprecision in computation can meet with respect to circuit-based figures of merit of a design.

Achieving high energy efficiency has become a key design objective for embedded and mobile computing devices due to their limited battery capacity and power budget. To improve energy efficiency of such computing devices, significant efforts have already been devoted at various levels, from software to architecture, and all the way down to circuit and technology levels. The need to support various digital signal processing (DSP) and classification applications on energy-constrained devices has steadily grown. Such applications often extensively perform matrix multiplications using fixed-point arithmetic while exhibiting tolerance for some computational errors. Hence, improving the energy efficiency of multiplications is critical. In this brief, we propose multiplier architectures that can tradeoff computational accuracy with energy consumption at design time. Compared with a precise multiplier, the proposed multiplier can consume 58% less energy/op with average computational error of 1%. Finally, we demonstrate that such a small computational error does not notably impact the quality of DSP and the accuracy of classification applications.

Approximate computing has received significant attention as a promising strategy to decrease power consumption of inherently error tolerant applications. In this paper, we focus on hardware-level approximation by introducing the partial product perforation technique for designing approximate multiplication circuits. We prove in a mathematically rigorous manner that in partial product perforation, the imposed errors are bounded and predictable, depending only on the input distribution. Through extensive experimental evaluation, we apply the partial product perforation method on different multiplier architectures and expose the optimal architecture perforation configuration pairs for different error constraints. We show that, compared with the respective exact design, the partial product perforation delivers reductions of up to 50% in power consumption, 45% in area, and 35% in critical delay. In addition, the product perforation method is compared with the state-of-the-art approximation techniques, i.e., truncation, voltage overscaling, and logic approximation, showing that it

Approximate circuits have been considered for error-tolerant applications that can tolerate some loss of accuracy with improved performance and energy efficiency. Multipliers are key arithmetic circuits in many such applications such as digital signal processing (DSP). In this paper, a novel approximate multiplier with lower power consumption and a shorter critical path than traditional multipliers is proposed for high-performance DSP applications. This multiplier leverages a newly-designed approximate adder that limits its carry propagation to the nearest neighbors for fast partial product accumulation. Different levels of accuracy can be achieved through a configurable error recovery by using different numbers of most significant bits (MSBs) for error reduction. The approximate multiplier has a low mean error distance, i.e., most of the errors are not significant in magnitude. Compared to the Wallace multiplier, a 16-bit approximate multiplier implemented in a 28nm CMOS process shows a reduction in delay and power of 20% and up to 69%, respectively. It is shown that by utilizing an appropriate error recovery, the proposed approximate multiplier achieves similar processing accuracy as traditional exact multipliers but with significant improvements in power and performance.

III. PROPOSED SYSTEM

Implementation of multiplier comprises three steps: generation of partial products, partial products reduction tree, and finally, a vector merge addition to produce final product from the sum and carry rows generated from the reduction tree. Second step consumes more power. In this brief, approximation is applied in reduction tree stage.

A 8-bit unsigned[1] multiplier is used for illustration to describe the proposed method in approximation of multipliers. Consider two 8-bit unsigned input operands $\alpha = \sum_{m=0}^{7} \alpha_m 2^m$ *and* $\beta = \sum_{n=0}^{7} \beta_n 2^n$. The partial product $a_{m,n} = \alpha_m \cdot \beta_n$ is the result of AND operation between the bits α_m and β_n.

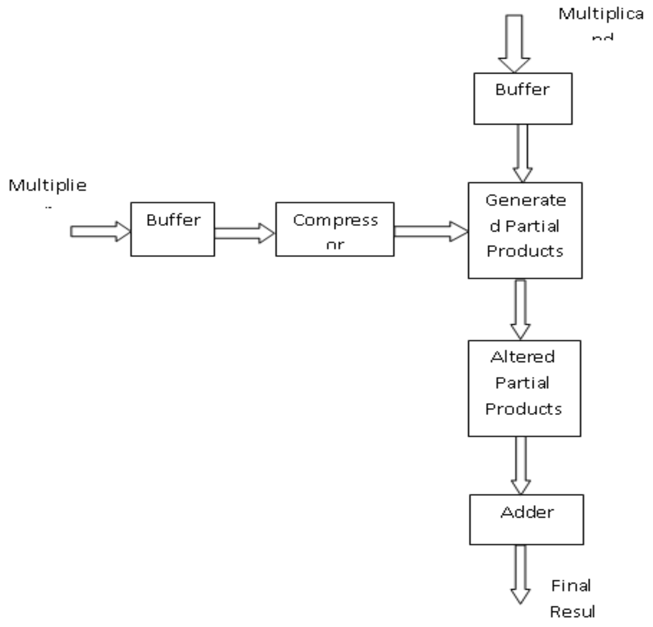


Fig.3.1 Proposed Block Diagram

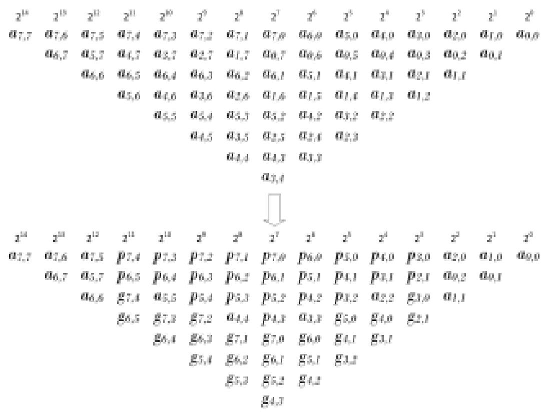


Fig. 3.2 Transformation of generated partial products into altered partial products

.

From statistical point of view, the partial product $a_{m,n}$ has a probability of 1/4 of being 1. In the columns containing more than three partial products, the partial products are combined to form propogate and generate signals as given in (1). The resulting propogate and generate signals form altered partial products $P_{m,n}$ and $g_{m,n}$. From column 3 with weight 2^3 to column 11 with weight 2^{11}, the partial products $a_{m,n}$ and $a_{n,m}$ are replaced by altered partial products $P_{m,n}$ and $g_{m,n}$. The original and transformed partial product matrices are shown in Fig. 3.2

$$p_{m,n} = a_{m,n} + a_{n,m}$$

$$g_{m,n} = a_{m,n} \cdot a_{n,m} \qquad (3.1)$$

The probability of the altered partial product $g_{m,n}$ being one is 1/16, which is significantly lower than 1/4of $a_{m,n}$. The probability of altered partial product $p_{m,n}$ being one is 1/16 + 3/16 + 3/16 = 7/16, which is higher than $g_{m,n}$. These factors are considered, while applying approximation to the altered partial product matrix.

APPROXIMATION OF ALTERED PARTIAL PRODUCTS $g_{m,n}$

The accumulation of generate signals is done columnwise. As each element has a probability of 1/16 of being one, two elements being 1 in the same column even decreases. For example, in a column with 4 generate signals, probability of all numbers being 0 is $(1 - pr)^4$, only one element being one is $4pr(1 - pr)^3$, the probability of two elements being one in the column is $6pr^2(1 - pr)^2$, three ones is $4pr^3(1-pr)$ and probability of all elements being 1 is pr^4, where pr is 1/16.

Inputs		Exact Outputs		Approximate Outputs		Absolute Difference
$x1$	$x2$	$Carry$	Sum	$Carry$	Sum	
0	0	0	0	0 ✓	0 ✓	0
0	1	0	1	0 ✓	1 ✓	0
1	0	0	1	0 ✓	1 ✓	0
1	1	1	0	1 ✓	1 ✗	1

Table 3.1 Table Truth Table of Approximate Half Adder

Inputs			Exact Outputs		Approximate Outputs		Absolute Difference
$x1$	$x2$	$x3$	$Carry$	Sum	$Carry$	Sum	
0	0	0	0	0	0 ✓	0 ✓	0
0	0	1	0	1	0 ✓	1 ✓	0
0	1	0	0	1	0 ✓	1 ✓	0
0	1	1	1	0	1 ✓	0 ✓	0
1	0	0	0	1	0 ✓	1 ✓	0
1	0	1	1	0	1 ✓	0 ✓	0
1	1	0	1	0	0 ✗	1 ✗	1
1	1	1	1	1	1 ✓	0 ✗	1

Table 3.2 Truth Table of Approximate Full Adder

APPROXIMATION OF OTHER PARTIAL PRODUCTS

The accumulation of other partial products with probability ¼ for $a_{m,n}$ and 7/16 for $p_{m,n}$ uses approximate circuits. Approximate half-adder, full-adder, and 4-2 compressor are proposed for their accumulation. Carry and Sum are two outputs of these approximate circuits. Since Carry has higher weight of binary bit, error in Carry bit will contribute more by producing error difference of two in the output. Approximation is handled in such a way that the absolute difference between actual output and approximate output is always maintained as one. Hence Carry outputs are approximated only for the cases, where Sum is approximated.

In adders and compressors, XOR gates tend to contribute to high area and delay. For approximating half-adder, XOR gate of Sum is replaced with OR gate as given in (2). This results in one error in the Sum computation as seen in the truth table of approximate half-adder in Table 3.1. A tick mark denotes that approximate output matches with correct output and cross mark denotes mismatch

$$Sum = x1 + x2$$
$$Carry = x1\ .x2 \qquad (3.2)$$

In the approximation of full-adder, one of the two XOR gates is replaced with OR gate in Sum calculation. This results in error in last two cases out of eight cases. Carry is modified as in (3) introducing one error. This provides more simplification, while maintaining the difference between original and approximate value as one. The truth table of approximate full-adder is given in Table 3.2

$$W = (x1 + x2)$$
$$Sum = W \oplus x3$$
$$Carry = W\ .x3 \qquad (3.3)$$

Two approximate 4-2 compressors produce nonzero output even for the cases where all inputs are zero. This results in high ED and high degree of precision loss especially in cases of zeros in all bits or in most significant parts of the reduction tree. The proposed 4-2 compressor overcomes this drawback.

In 4-2 compressor, three bits are required for the output only when all the four inputs are 1, which happens only once out of 16 cases. This property is taken to eliminate one of the three output bits in 4-2 compressor. To maintain minimal error difference as one, the output “100" (the value of 4) for four inputs being one has to be replaced with outputs “11" (the value of 3). For Sum computation, one out of three XOR gates is replaced with OR gate. Also, to make the Sum corresponding to the case where all inputs are ones as one, an additional circuit x1 · x2 · x3 · x4 is added to the Sum expression. This results in error in five out of 16 cases. Carry is simplified as in (4). The corresponding truth table is given in Table 3.3

Inputs				Approximate outputs		Absolute Difference
$x1$	$x2$	$x3$	$x4$	$Carry$	Sum	
0	0	0	0	0 ✓	0 ✓	0
0	0	0	1	0 ✓	1 ✓	0
0	0	1	0	0 ✓	1 ✓	0
0	0	1	1	1 ✓	0 ✓	0
0	1	0	0	0 ✓	1 ✓	0
0	1	0	1	0 ✗	1 ✗	1
0	1	1	0	0 ✗	1 ✗	1
0	1	1	1	1 ✓	1 ✓	0
1	0	0	0	0 ✓	1 ✓	0
1	0	0	1	0 ✗	1 ✗	1
1	0	1	0	0 ✗	1 ✗	1
1	0	1	1	1 ✓	1 ✓	0
1	1	0	0	1 ✓	0 ✓	0
1	1	0	1	1 ✓	1 ✓	0
1	1	1	0	1 ✓	1 ✓	0
1	1	1	1	1 ✗	1 ✗	1

Table 3.3 Truth Table of Approximate 4-2 Compressor

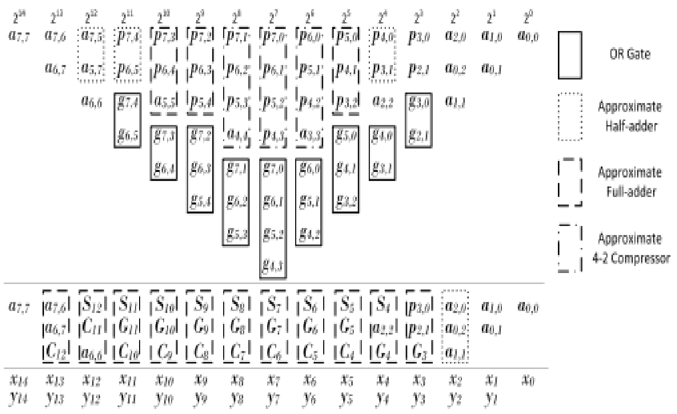


Fig. 3.3 Reduction of altered partial products

$$W1 = x1 \,.\, x2$$
$$W2 = x3 \,.\, x4$$
$$Sum = (x1 \oplus x2) + (x3 \oplus x4) + W1 \,.\, W2$$
$$Carry = W1 + W2 \qquad (4)$$

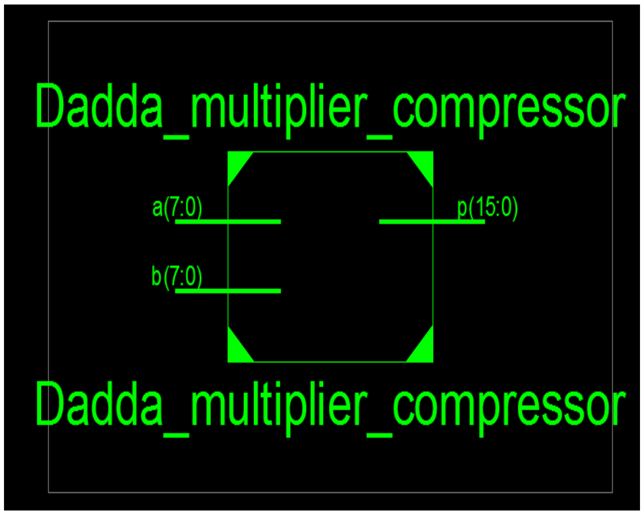
Fig. 3.3 shows the reduction of altered partial product matrix of 8×8 approximate multiplier. It requires two stages to produce sum and carry outputs for vector merge addition step. Four 2-input OR gates, four 3-input OR gates, and one 4-input OR gates are required for the reduction of generate signals from columns 3 to 11.

The resultant signals of OR gates are labeled as G corresponding to the column i with weight 2^i. For reducing other partial products, 3 approximate half-adders, 3 approximate full-adders, and 3 approximate compressors are required in the first stage to produce Sum and Carry signals, S_i and C_i corresponding to column i. The elements in the second stage are reduced using 1 approximate half-adder and 11 approximate full-adders producing final two operands x carry adder for the final computation of the result.

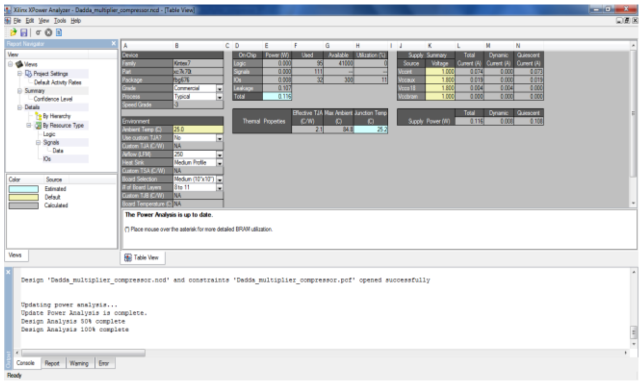
TWO VARIANTS OF MULTIPLIERS

Two variants of multipliers are proposed. In the first case (Multiplier1), approximation is applied in all columns of partial products of n-bit multiplier, whereas in Multiplier2, approximate circuits are used in n − 1 least significant columns.

IV. RESULT AND DISCUSSION



Power analyzer for Dadda multiplier compressor



Device	
Family	Kintex7
Part	xc7k70t
Package	fbg676
Grade	Commercial
Process	Typical
Speed Grade	-3

Environment	
Ambient Temp (C)	25.0
Use custom TJA?	No
Custom TJA (C/W)	NA
Airflow (LFM)	250
Heat Sink	Medium Profile
Custom TSA (C/W)	NA
Board Selection	Medium (10"x10")
# of Board Layers	8 to 11
Custom TJB (C/W)	NA
Board Temperature (C)	NA

On-Chip	Power (W)	Used	Available	Utilization (%)
Logic	0.000	95	41000	0
Signals	0.000	111	---	---
IOs	0.008	32	300	11
Leakage	0.107			
Total	0.116			

Thermal Properties	Effective TJA (C/W)	Max Ambient (C)	Junction Temp (C)
	2.1	84.8	25.2

Supply Summary Source	Voltage	Total Current (A)	Dynamic Current (A)	Quiescent Current (A)
Vccint	1.000	0.074	0.000	0.073
Vccaux	1.800	0.019	0.000	0.019
Vcco18	1.800	0.004	0.004	0.000
Vccbram	1.000	0.000	0.000	0.000

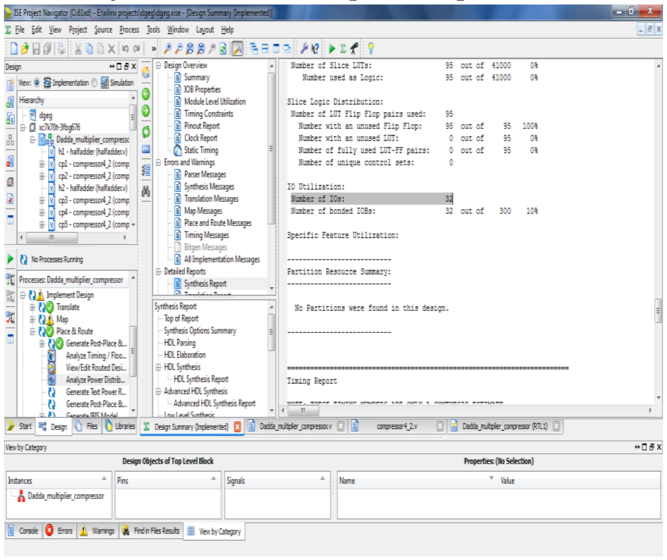
	Total	Dynamic	Quiescent
Supply Power (W)	0.116	0.008	0.108

The Power Analysis is up to date.

(*) Place mouse over the asterisk for more detailed BRAM utilization.

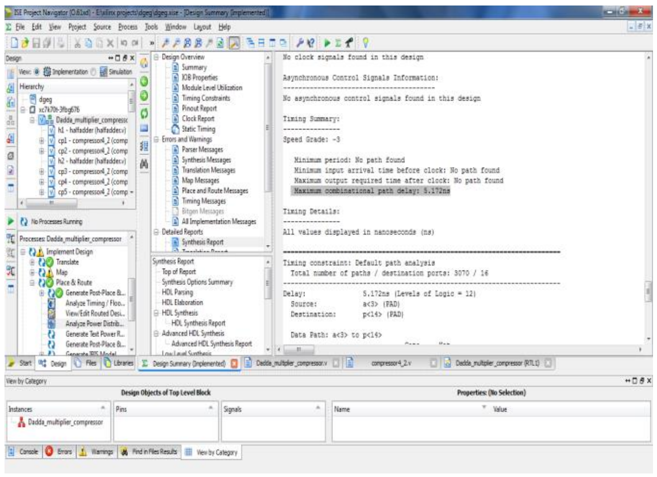
```
Design 'Dadda_multiplier_compressor.ncd' and constraints 'Dadda_multiplier_compressor.pcf' opened successfully


Updating power analysis...
Update Power Analysis is complete.
Design Analysis 50% complete
Design Analysis 100% complete
```

Area analyzer for Dadda multiplier compressor



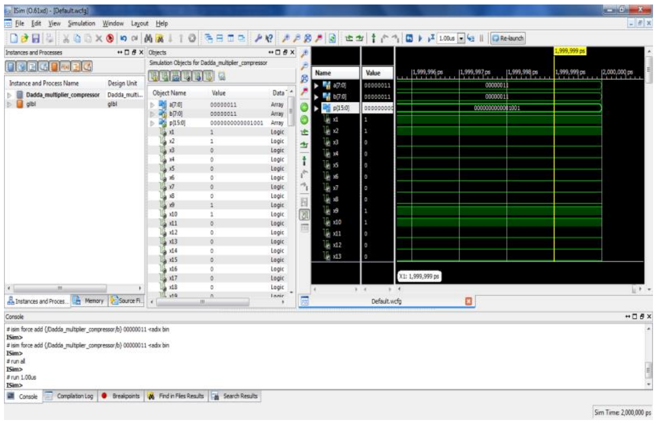
```
 Number of Slice LUTs:                  95  out of  41000     0%
    Number used as Logic:               95  out of  41000     0%

Slice Logic Distribution:
 Number of LUT Flip Flop pairs used:    95
   Number with an unused Flip Flop:     95  out of     95   100%
   Number with an unused LUT:            0  out of     95     0%
   Number of fully used LUT-FF pairs:    0  out of     95     0%
   Number of unique control sets:        0

IO Utilization:
 Number of IOs:                         32
 Number of bonded IOBs:                 32  out of    300    10%

Specific Feature Utilization:

---------------------------
Partition Resource Summary:
---------------------------

  No Partitions were found in this design.

---------------------------



=========================================================================
Timing Report
```

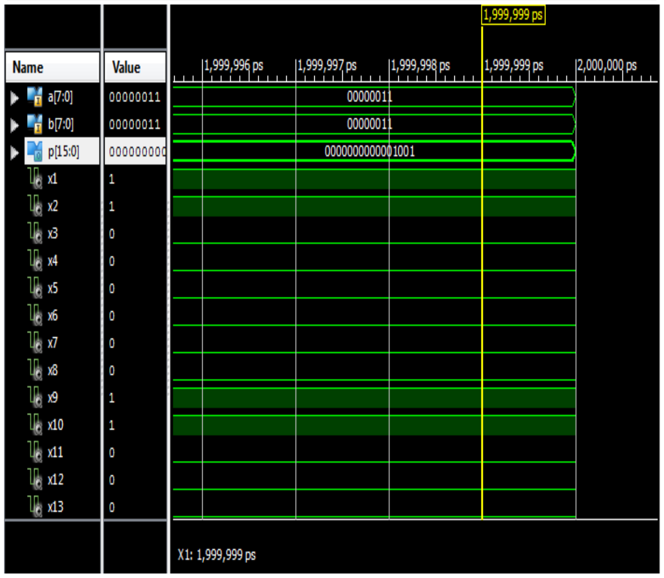
Delay analyzer for Dadda multiplier compressor



```
No clock signals found in this design

Asynchronous Control Signals Information:
----------------------------------------
No asynchronous control signals found in this design

Timing Summary:
---------------
Speed Grade: -3

   Minimum period: No path found
   Minimum input arrival time before clock: No path found
   Maximum output required time after clock: No path found
   Maximum combinational path delay: 5.172ns

Timing Details:
---------------
All values displayed in nanoseconds (ns)

=========================================================================
Timing constraint: Default path analysis
  Total number of paths / destination ports: 3070 / 16
-------------------------------------------------------------------------
Delay:               5.172ns (Levels of Logic = 12)
  Source:            a<3> (PAD)
  Destination:       p<14> (PAD)

  Data Path: a<3> to p<14>
```

Simulation results for Dadda multiplier compressor with Instance, Process Name, Objects and Simulation waveform





V. CONCLUSION

Thus, to proposed efficient approximate multipliers, partial products of the multiplier are modified using generate and propagate signals. Approximation is applied using simple OR gate for altered generate partial products. Approximate half-adder, full-adder, and 4-2 compressor are proposed to reduce remaining partial products. Two variants of approximate multipliers are proposed, where approximation is applied in all n bits in Multiplier1 and only in n −1 least significant part in Multiplier2. Multiplier1 and Multiplier2 achieve significant reduction in area and power consumption compared with exact designs. They are also found to have better precision when compared to existing approximate multiplier designs. The proposed multiplier designs can be used in applications with minimal loss in output quality while saving significant power and area.

REFERENCES

[1] Momeni, J. Han, P. Montuschi, and F. Lombardi, "Design and analysis of approximate compressors for multiplication," IEEE Trans. Comput., vol. 64, no. 4, pp. 984–994, Apr. 2015.

[2] Liu, J. Han, and F. Lombardi, "A low-power, high-performance approximate multiplier with configurable partial error recovery," in Proc. Conf. Exhibit. (DATE), 2014, pp. 1–4.

[3] C.-H. Lin and C. Lin, "High accuracy approximate multiplier with error correction," in Proc. IEEE 31st Int. Conf. Comput. Design, Sep. 2013, pp. 33–38.

[4] E. E. Swartzlander, "Truncated multiplication with approximate rounding," in Proc. Conference Record of the Thirty-Third Asilomar Conference on Signals, Systems, and Computers (Cat. No.CH37020), pp. 1480-1483, 1999.

[5] E. J. King and E. E. Swartzlander, Jr., "Data-dependent truncation scheme for parallel multipliers," in Proc. 31st Asilomar Conf. Signals, Circuits Syst., Nov. 1998, pp. 1178–1182.

[6] G. Zervakis, K. Tsoumanis, S. Xydis, D. Soudris, and K. Pekmestzi, "Design-efficient approximate multiplication circuits through partial product perforation," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 24, no. 10, pp. 3105–3117, Oct. 2016.

[7] H. R. Mahdiani, A. Ahmadi, S. M. Fakhraie, and C. Lucas, "Bio-inspired imprecise computational blocks for efficient VLSI implementation of soft-computing applications," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 57, no. 4, pp. 850–862, Apr. 2010.

[8] H. Waris, C. Wang and W. Liu, "Hybrid Low Radix Encoding based Approximate Booth Multipliers," IEEE

Transactions on Circuits and Systems II: Express Briefs, doi: 10.1109/TCSII.2020.2975094.

[9] K.-J. Cho, K.-C. Lee, J.-G. Chung, and K. K. Parhi, “Design of low-error fixed-width modified booth multiplier,” IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 12, no. 5, pp. 522–531, May 2004.

[10] P. Kulkarni, P. Gupta, and M. D. Ercegovac, “Trading accuracy for power in a multiplier architecture,” J. Low Power Electron., vol. 7, no. 4, pp. 490–501, 2011.

[11] S. Narayanamoorthy, H. A. Moghaddam, Z. Liu, T. Park, and N. S. Kim, “Energy-efficient approximate multiplication for digital signal processing and classification applications,” IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 23, no. 6, pp. 1180–1184, Jun. 2015.

[12] S. Vahdat, M. Kamal, A. Afzali-Kusha and M. Pedram, “TOSAM: An Energy-Efficient Truncation-and Rounding-Based Scalable Approximate Multiplier,” IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 27, no. 5, pp. 1161-1173, 2019.

[13] V. Gupta, D. Mohapatra, A. Raghunathan, and K. Roy, “Low-power digital signal processing using approximate adders,” IEEE Trans. Comput.Aided Design Integr. Circuits Syst., vol. 32, no. 1, pp. 124–137, Jan. 2013.

[14] V. Mrazek, R. Hrbacek, Z. Vasicek, and L. Sekanina, “EvoApprox8b: Library of approximate adders and multipliers for circuit design and benchmarking of approximation methods,” in Proc. Design, Automation & Test in Europe Conference & Exhibition (DATE), pp. 258-261, 2017.

[15] W. Liu, F. Lombardi and M. Shulte, “A Retrospective and Prospective View of Approximate Computing [Point of View],” Proceedings of the IEEE, vol. 108, no. 3, pp. 394–399, 2020.