Vol 05, Article 07387 July 2014
http://ijves.com
International Journal of VLSI and Embedded Systems-IJVES
ISSN: 2249 – 6556

A Novel Architecture for Optimised Parallel Multiplier

SONIA VERMA[1], SAKSHI[2]
[1,2]ECED, Thapar University, Patiala, PUNJAB
[1]soniavlsithapar@gmail.com, [2]sakshi.bajaj@thapar.edu

ABSTRACT

Multipliers are indispensable in modern electronic systems where high speed calculations are required such as Finite Impulse Response filters, digital signal processors and microprocessors etc. Many multimedia and Digital Signal Processing applications are highly multiplication intensive, hence the performance and power consumption of these systems are dominated by multipliers. Also the multiplier is generally the slowest element in the system. Consequently, it is of vital importance to develop power efficient multipliers to develop a high-performance and low-power portable multi-media and Digital Signal Processing systems. This paper presents a power efficient Dynamic Range Detection Based Multiplier. Dynamic Range Detection technique picks the operand with smaller dynamic range for booth encoding to increase the probability of partial products becoming zero and hence reduces the redundant switching activity, saving power. Radix-4 Booth's algorithm is used to generate the partial products and it reduces the number of partial products to half as compared to standard method of multiplication. Compressors are used in partial product accumulation stage of multiplier to compress the partial product rows. Generally full adder (3:2 compressor) and half adder (2:2 compressor) is used. Speed improvement is achieved by using compressors of higher order such as 4:2, 5:2 and 7:2 to sum up the partial products. Delay comparison of the higher order compressors is done with 3:2 compressor. All these multiplier designs are modeled in VHDL and then simulated and synthesized using Xilinx ISE 14.5 targeted on the Spartan 3E FPGA.

Keywords: *DRD, 3:2, 4:2, 5:2 and 7:2 compressors*

1. INTRODUCTION

With the constant growth of computer applications such as computer graphics and signal processing, fast arithmetic units especially multipliers are becoming increasingly important [3]. Multiplication is a widely used and power consuming operation in microprocessor and DSP applications. Multiplier not only consumes more power but also occupies significant area and introduces delay. So it is necessary to develop a power and speed efficient multiplier. Dynamic Range Detection selects the operand that makes more partial product rows zero and hence reduces the switching activity and power consumed. Multiplication operation involves generation of partial products and their accumulation. The improvement in speed of multiplication can be attained by reducing the number of partial products and/or accelerating the accumulation of partial products. There are two basic approaches namely Booth algorithm and Wallace Tree compressors among many methods of implementing high speed parallel multipliers [2]. This paper describes an efficient implementation of a high speed parallel multiplier using both these approaches.

The number of partial products to be added largely determines the speed of multiplier.. One of the most popular algorithms used to reduce the number of partial products is Booth's algorithm. In the conventional multiplier, the number of partial products are determined by the number of bits of the multiplier. The larger the number of bits the multiplier contains, the longer it takes to produce the product [6]. Booth multiplication is used to increase the speed of the multiplier by encoding the numbers that are multiplied. This is a standard technique used in chip design and provides significant improvements over the long multiplication technique. Multipliers require high amount of power and delay during the partial products accumulation. At this stage, carry save adders have been used to add the partial products. Compressors of different orders such as 3:2, 4:2, 5:2 and 7:2 sum up three, four, five and seven rows at a time respectively. These compressors reduce the number of partial product rows to two. Finally a carry propagating adder adds the last two rows to give the final result of multiplication.

2. DYNAMIC RANGE DETECTION

The Dynamic Range Detector breaks Multiplicand A into A_L and A_H and Multiplier B into B_L and B_H where A_L, A_H, B_L and B_H are of 8bits each as shown in fig.2. Firstly the input operands are partitioned into 3-bit groups a_{2i+1}, a_{2i}, a_{2i-1} or b_{2i+1}, b_{2i}, b_{2i-1} and are fed into comparator. If the output of a comparator is 1, it indicates that the input 3-bit group is successive zeroes or ones and hence it's booth encoded product will be zero. The number of booth encoded products with a zero value is denoted as Δ_i [1]. Switching signals SW_{HH} and SW_{LL} are generated by the circuit shown in fig. 1. If SW_{LL} ='1' then the lower bits of multiplicand and multiplier are interchanged else no operation is done. Similarly if SW_{HH}='1' then the higher bits of multiplicand and multiplier are interchanged. So basically SW_{LL} ='1' or SW_{HH}='1' indicates that probability of more number of partial product rows becoming zero is higher, if the multiplicand and multiplier are interchanged. If C_k (three consecutive bits of multiplier) =“000” or “111” then partial product row S_k is 0. So operand which has more number of C_k as “000” or “111” is preferred to be taken as multiplier.

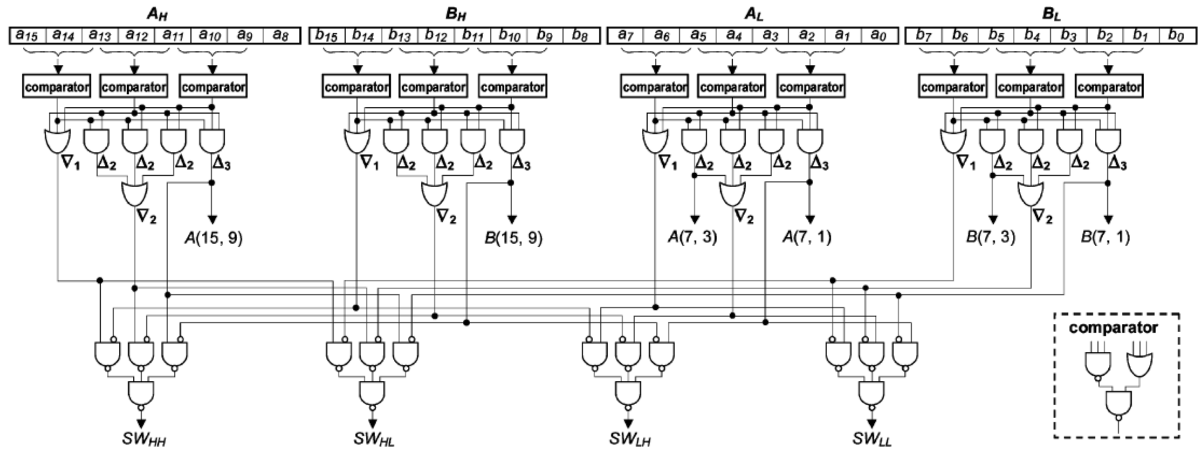

A_H
B_H
A_L
B_L
a_{15} a_{14} a_{13} a_{12} a_{11} a_{10} a_9 a_8
b_{15} b_{14} b_{13} b_{12} b_{11} b_{10} b_9 b_8
a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0
b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0
comparator
∇_1 Δ_2 Δ_2 Δ_2 Δ_3
∇_2
A(15, 9)
B(15, 9)
A(7, 3)
A(7, 1)
B(7, 3)
B(7, 1)
SW_{HH}
SW_{HL}
SW_{LH}
SW_{LL}
comparator

Fig. 1 Dynamic Range Detection Unit

3. ARCHITECTURE

The architecture of multiplier is shown in fig. 2. It consists of five major modules namely DRD unit, Booth encoder, partial product generation, partial product accumulation, Carry Propagate Adder and Final Product. DRD unit chooses the operand which is to be taken as multiplier. Booth encoder performs encoding of the multiplier bits. Based on the multiplicand and the encoded multiplier, partial products are generated by the partial product generator. Partial product accumulation is done by compressors which may take 3, 4, 5 or 7 rows at a time. These compressors reduce the number of partial product rows to two and last two rows of partial product are summed up by a carry propagate adder.

3.1 DRD Unit

Basic multiplication operation involves adding the multiplicand, n number of times where n is the value of multiplier. So the number of addition operations depends on the value of multiplier. Suppose we need to multiply 5 and 3, adding 5 three times (5+5+5) would be faster than adding 3 five times (3+3+3+3+3). And also the first operation will consume less power. So wisely choosing the multiplicand and multiplier also contributes to the performance of multiplier. DRD unit solves the purpose.

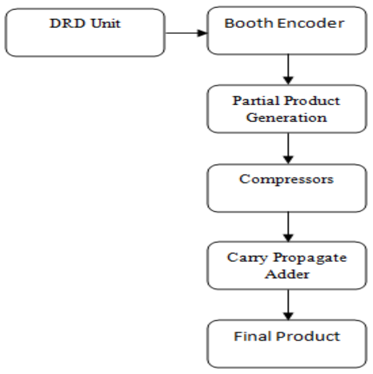


Fig. 2 Block Diagram of proposed multiplier

3.2 Booth Encoder

Booth algorithm is a powerful algorithm for signed number multiplication, which treats both positive and negative numbers uniformly. Since a k-bit binary number can be interpreted as k/2-digit Radix-4 number, a k/3-digit Radix-8 number and so on, it can deal with more than one bit of the multiplier in each cycle by using high radix multiplication [2]. Reduction in the number of partial products relies upon the number of bits encoded. The algorithm of Radix-based Booth Encoding multipliers is explained below. Let A and B be two n-bit two's complement binary numbers where A represents the multiplicand and B represents the multiplier. A with 16 bits is represented as A_{15} A_{14} $A_{13\dots}$ A_2 A_1 A_0 and B with 16 bits is represented as B_{15} B_{14} $B_{13\dots}$ B_2 B_1 B_0 and

$$P = A \times B$$

$$A \times B = \{(S_{n-1} \times 2_{m(n-1)})\rangle + (S_{n-2} \times 2_{m(n-2)}) + (S_{n-3} \times 2_{m(n-3)}) + \dots + (S_1 \times 2_{m(1)}) + (S_0 \times 2_{m(0)})\} \quad \dots\dots..(1)$$

where S_k for n-1<=k<=0, is a value that depends on the value of B and can be determined as shown in Table 1 while n represents the number of bits in the multiplicand or the multiplier and m = 2, 3, 4 and so on. Radix-based Booth Encoding multiplier is encoded using 3-bits which are represented as C_k which gives S_k to determine the partial product. Radix-4 Booth Encoding multiplier is preferred over Radix-2 and other higher Radix encoding designs because it has the best speed advantage. So it is the best-suited for high speed applications. Encoding designs such as Radix-8, Radix-16 and Radix-32 shows a delay smaller than the Radix-2 but the delay increases when compared with the Radix-4 design [6]. Also in terms of area, Radix-4 design gives the best results. Radix-4 multiplication of 16×16 bit multiplier is shown in fig. 3.

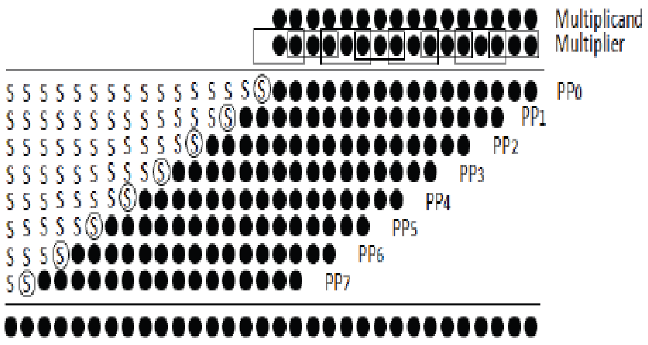


Fig. 3 Dot Diagram of Radix-4 Booth Encoding Multiplier

3.3 Partial Product Accumulation

Compressors are used to accumulate the partial products. These are arithmetic components, similar in principle to parallel counters, but with two distinct differences: (i) they have explicit carry-in and carry-out bits; and (ii) there may be some redundancy among the ranks of the sum and carry-output bits [4].

- 3:2 Compressor

This is the basic compressor which is similar to full adder. It takes three input bits A, B and Cin of rank 0 and produces sum of rank 0 and Cout of rank 1 as shown in fig. 4.

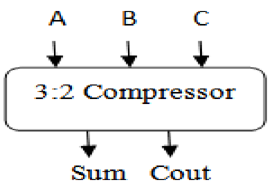


Fig. 4 3:2 Compressor Architecture

- 4:2 Compressor

The 4:2 compressor has 4 input bits and produces 2 output bits (Sum and Cout1), it also has a carry-in (Cin) and a carry-out (Cout2) bit. Thus, the total number of input/output bits are 5 and 3. All input bits including Cin have rank 0 and the two output bits have ranks 0 and 1 respectively, while cout2 has rank 1 as shown in fig. 5. In the architecture of 4:2 compressor two 3:2 compressors are used in series as shown in fig. 5.

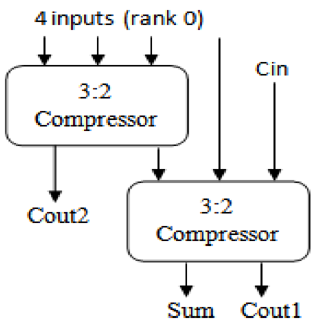


Fig. 5 4:2 Compressor Architecture

- 5:2 Compressor

The 5:2 compressor takes 5 input bits and produces two output bits i.e Sum and Cout1. It also has carry in bits (Cin1 and Cin2) and carry out bits (Cout1, Cout2 and Cout3). Thus the total number of input/output bits are 7/4. All the input bits including Cin1 and Cin2 have rank 0 and the output bits Cout1, Cout2 and Cout3 have rank 1 and Sum bit has rank 0 as shown in fig. 6.

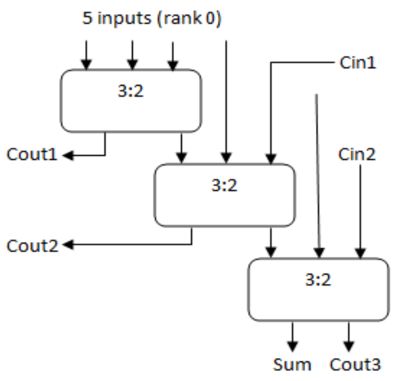


Fig. 6 5:2 Compressor Architecture

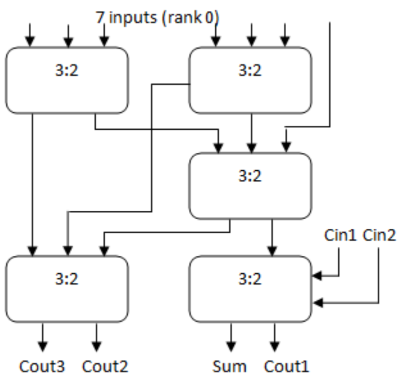
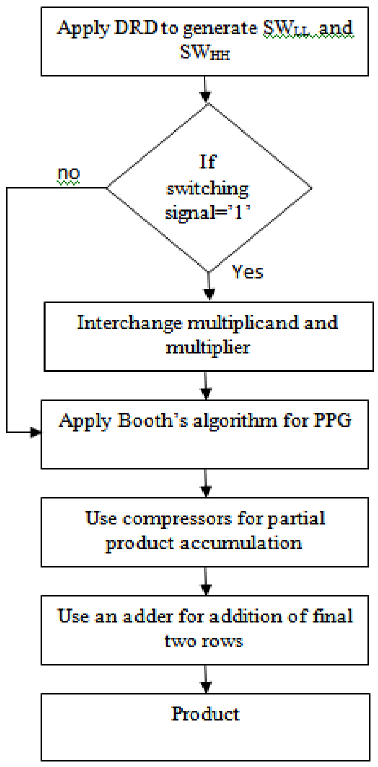


Fig. 7 7:2 Compressor Architecture

- 7:2 Compressor

The 7:2 compressor takes 7 input bits and produces 2 output bits (Sum and Cout1), it also has carry-in input bits (Cin1, Cin2) and carry-out (Cout1, Cout2) bits. Thus, the total number of input/output bits are 9 and 4. All input bits, including Cin1, have rank 0 and Cin2 has rank 1, the two output bits have ranks 0 and 1 respectively, while Cout1 has rank 1 and Cout2 has rank 2 as shown in fig. 7.

4. METHODOLOGY


Apply DRD to generate SW_{LL} and SW_{HH}
If switching signal='1'
no
Yes
Interchange multiplicand and multiplier
Apply Booth's algorithm for PPG
Use compressors for partial product accumulation
Use an adder for addition of final two rows
Product
Fig. 8 Flow Chart of Proposed Multiplier

The flow chart of multiplication process is shown in fig. 11. Firstly DRD is applied on multiplicand and multiplier. DRD unit is shown in figure 1. It consists of comparator which generates delta signals and finally switching signals are generated by the circuit. These switching signals determine whether to interchange the multiplicand and multiplier or not. After choosing multiplier, radix-4 booth's algorithm is applied to generate the partial product rows. In 8×8 bit multiplication, 4 rows are generated and in 16×16 bit multiplication, 8 partial product rows are generated. So number of rows are reduced to half by using radix-4 booth's algorithm. Now compressors of order 3:2, 4:2, 5:2 and 7:2 are used to sum up the partial product rows. Use of higher order compressors reduces multiplication delay. These compressors are more effective for high order multiplication. Finally a carry propagate adder sums up the final two rows to get the final product.

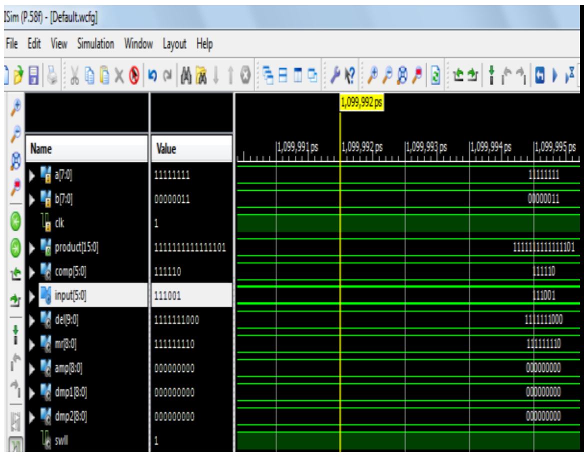
Vol 05, Article 07387 July 2014
http://ijves.com

International Journal of VLSI and Embedded Systems-IJVES
ISSN: 2249 – 6556

5. RESULTS AND DISCUSSIONS

The multipliers have been designed in VHDL, simulated and synthesized using Xilinx ISE 14.5 targeted on the Spartan 3E FPGA. The design entry of 8×8 bit multipliers with 3:2 and 4:2 compressors and 16×16 bit multipliers with 3:2, 4:2, 5:2 and 7:2 compressors is done. Power is calculated using X-Power Estimator 11.1. Functional simulation results and RTL Schematic for the multiplier are shown in figure 9 and figure 10.

Table II shows comparison of 8×8 bit and 16×16 bit multiplier designs without using DRD in terms of area (number of slices) and delay (ns). Table III lists the comparison of 8×8 bit and 16×16 bit multiplier designs using DRD in terms of area(number of slices) and delay(ns). Power results are shown in Table 4. Fig.11 and fig. 12 shows changes in delay and power respectively with changes in order of compressor.



Name	Value
a[7:0]	11111111
b[7:0]	00000011
clk	1
product[15:0]	1111111111111101
comp[5:0]	111110
input[5:0]	111001
del[9:0]	1111111000
mr[8:0]	111111110
amp[8:0]	000000000
dmp1[8:0]	000000000
dmp2[8:0]	000000000
swll	1

Fig. 9 Simulation waveforms of parallel multiplier

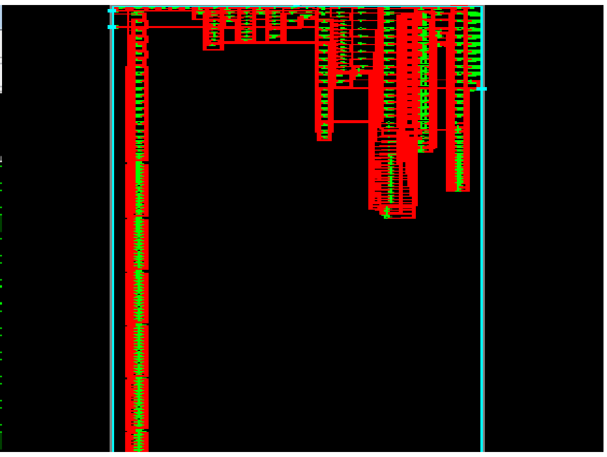


Fig. 10 RTL Schematic of 16×16 bit multiplier

Table II
No. of slices and delay of multiplier designs without using DRD

Multiplier (no DRD)	Compressor	No. of slices	Delay(ns)
8×8 bit	3:2	102/4656	4.365
16×16 bit	3:2	337/4656	5.673
16×16 bit	4:2	340/4656	5.516
16×16 bit	5:2	359/4656	5.326
16×16 bit	7:2	377/4656	5.032

Table III
No. of slices and delay of multiplier designs without using DRD

Multiplier (DRD)	Compressor	No. of slices	Delay(ns)
8×8 bit	3:2	115/4656	4.876
8×8 bit	4:2	119/4656	4.778
16×16 bit	3:2	337/4656	6.514
16×16 bit	4:2	340/4656	6.216
16×16 bit	5:2	359/4656	6.117
16×16 bit	7:2	377/4656	5.089

Table IV
Power consumption of multiplier designs with and without using DRD

Power (mW)	Compressor	DRD	No DRD
16×16 bit	3:2	0.119	0.323
16×16 bit	4:2	0.107	0.312
16×16 bit	5:2	0.101	0.307
16×16 bit	7:2	0.099	0.299

Vol 05, Article 07387 July 2014
http://ijves.com
International Journal of VLSI and Embedded Systems-IJVES
ISSN: 2249 – 6556

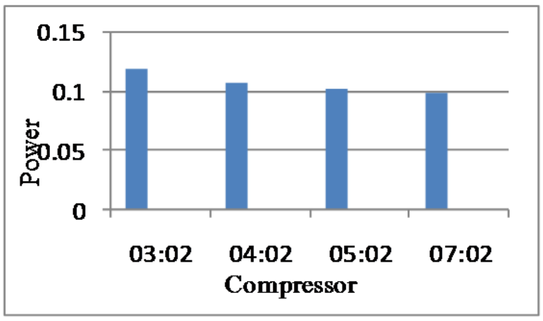
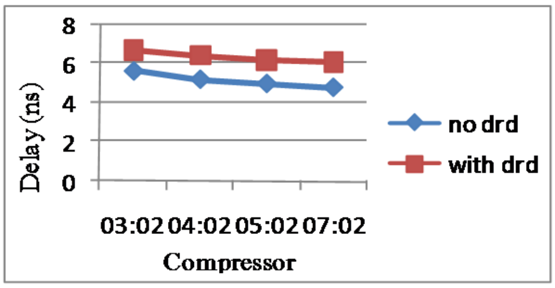


Fig. 11 Changes in Delay as order of compressor increases (16×16 bit)
Fig. 12 Changes in Power(mW) with increase in order of compressor(16×16 bit)



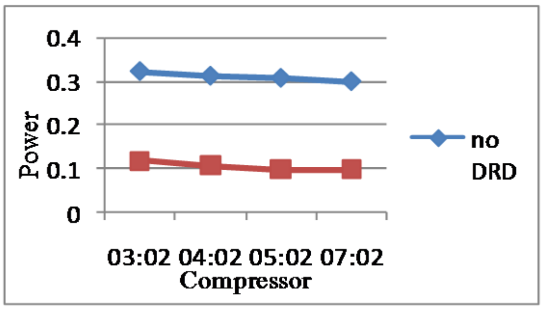


Fig.13 Delay(ns) comparison of 16bit multiplier with and without DRD
Fig. 14 Power (mW) Comparison of 16 bit multiplier with and without DRD

6. CONCLUSION

In this paper, power and speed comparison of parallel multiplier using DRD and without using DRD is done. It is found that by using DRD and compressors, power consumption is reduced by 3.2% for 8×8 bit multiplier and by 5.6% for 16×16 bit multiplier. From compressor 3:2 to 7:2, reduction in power consumption is 7.3% for 16×16 bit multiplier. So we can conclude that as the multiplier size increases, power consumption decreases at a higher rate and as we increase the order of compressor, power is further reduced. Further, delay results show that as order of compressor increases, delay decreases but at the cost of area. But we can neglect the increase in area as we are getting an improvement in both speed and power. Hence power reduction is achieved by DRD and speed is increased by using compressors.

REFERENCES

[1] D. Garg, S. Arya, ”Design of Configurable Booth Multiplier Using Dynamic Range Detector,” *International Journal of Electronics Engineering*, Vol. 4, No. 3, March 2012.
[2] P. R. Aparna, N. Thomas , ”Design and Implementation of a high performance multiplier using HDL,” in *Proc. IEEE Int. Conf. Computing, Communication and Applications*, pp. 1-4, Feb. 2012.
[3] R. Hussin, A. Y. M. Shakaff, N. Idris, Z. Sauli, R. C. Ismail, A. Kamarudin, ”An Efficient Modified Booth Multiplier Architecture,” in *Proc. IEEE Int. Conf. Electronics Design*, pp. 1-4, Dec. 2008.
[4] J. Kaur, N. K. Gahlan, P. Shukla, ”Delay Power Comparison of Array Multiplier in VLSI Design,” *International Journal of Advanced Research in Computer Science and Electronics Engineering*, Vol. 1, No. 3, May 2012.
[5] S. R. Kuang, J. P. Wang, ”Design of Power Efficient Configurable Booth Multiplier,” *IEEE Trans. Circuits & Systems I*, Vol. 57, pp. 568-580, March 2010.
[6] K. L. Suet, L. H. Hiung, ”Performance Comparison Review of Radix Based Multiplier Designs”, in *Proc. IEEE Int. Conf. on Intelligent and Advanced Systems*, Vol. 2, pp. 854-859, June 2012.