Vol 05, Article 09410, September 2014
http://ijves.com
International Journal of VLSI and Embedded Systems-IJVES
ISSN: 2249 – 6556

IMPLEMENTATION OF HIGH PERFORMANCE BINARY SQUARER

PRADEEP M C[1], RAMESH S[2]
[1,2]Department of Electronics and Communication Engineering, Dr. Ambedkar Institute of Technology, Mallathahally, Bangalore, India
[1]pradeepmc02@gmail.com, [2]rameshs.hullepura@gmail.com

ABSTRACT

In mobile driven market to carry-on with present technology high speed application requires faster methods of square architecture. In this work high performance binary number squarer using the concept of Vedic mathematic sutras is presented. Vedic multiplier and binary adder circuits are used for the design of Vedic squaring circuit architecture. Different optimizations are presented to design squarer circuit architecture to have low power and high speed. Optimizations are carried-out by Partial Product (PP) folding method and rearrangement of the PPs. The proposed Vedic squarer circuit is synthesized using Xilinx 13.1 version tool for Field Programmable Gate Array (FPGA) flow and Cadence 12.10 version tool for Application Specific Integrated Circuit (ASIC) flow for the analysis of dynamic power consumption and propagation delay and the design is simulated using Modelsim 6.5 version tool for functional verification.

Keywords*: ASIC flow, Binary Adder, FPGA flow, Vedic mathematics*

1. INTRODUCTION

Square is an arithmetic circuit used in some special processors such as Digital Signal Processing (DSP) [1][2]. Specialized squaring circuits have been proposed for DSP applications such as image compression, pattern recognition etc..., [3][4]. Squaring operations is special multiplication operation which has two equal operands. So a multiplier with two equal inputs can be used as squaring circuit. But in some arithmetic processors due to the increased delay and power which is caused by using the multiplier circuit as squarer, a special squarer circuit can be designed for squaring operation.

In this paper, architecture used to design squaring of a binary number is explored to create a circuit using Vedic sutras. By using Vedic sutras the overall processor performance can be improved for many applications. Therefore the goal is to create a squaring architecture that is comparable in speed and power than a design using standard multiplier.

This paper is organized as follows. In section 2, the overview of related work is briefly reviewed. In section 3, the proposed Vedic squarer architecture is discussed. The performance of proposed Vedic squarer architecture is compared with existing Vedic squarer architecture with results and discussion in section 4. Finally, a brief conclusion is given is section 5.

2. RELATED WORK

Vedic mathematics was rediscovered from the ancient Indian scriptures between 1911 and 1918 by Sri Bharati Krishna Tirthaji (1884-1960), a scholar of Sanskrit, mathematics, history and philosophy [5]. He studied these ancient texts for years and after careful investigation, was able to reconstruct a series of mathematical formulae called sutras. Vedic mathematics is the name given to the ancient system of mathematics, or, to be precise, a unique technique of calculations based on simple rules and principles with which any mathematical problem can be solved – be it arithmetic, algebra, geometry or trigonometry [6]. The system is based on 16 Vedic sutras or aphorisms, which were actually word formulae describing natural ways of solving a whole range of mathematical problems. One of the sutras of Vedic mathematics implied for multiplication is Urdhava Tiryakbhyam (vertical and cross wire) [7] which is also the foundation of the proposed design. It is based on a concept through which the generation of all Partial Products (PP) can be done with the concurrent addition of these PPs. The parallelism in generation of PPs and their summation is obtained by vertical and cross wire multiplication and addition. Various examples and implementation of Urdhava Tiryakbhyam sutra is discussed in [7].

In Vedic multiplier the Partial Product Generation (PPG) and additions are done concurrently. This feature makes it more attractive for binary multiplications. In most of the computations the multiplier unit is used to compute the square of an operand. Since squarer is a special case of multiplication a dedicated squaring hardware will significantly improve the computation time. A comparison between Vedic and conventional multiplier is discussed in [8]. Urdhava Tiryakbhyam sutra of Vedic mathematics is used for the Vedic multiplier design. This referred paper conclude that conventional and Vedic methods are computationally same and difference between the two lies in implementation strategy because of which Vedic multiplier has improved efficiency. Similar work is described in [9]. Multiplier and squarer were designed using Urdhava Tiryakbhyam sutra and duplex property. This design shows that Vedic mathematical methods are computationally faster and

Vol 05, Article 09410, September 2014
http://ijves.com
International Journal of VLSI and Embedded Systems-IJVES
ISSN: 2249 – 6556

easy to perform than conventional method. A binary number squarer is described in [10]. Here, one multiplier and two squaring unit is implemented using Urdhava Tiryakbhyam sutra to design Vedic squarer circuit, to have reduced delay. This squarer is proved to have improved efficiency in terms of speed.
This work attempts to formulate an interactive general strategy for designing and implementation of Vedic squarer based on principles of Vedic mathematics.

3. PROPOSED SQUARER ARCHITECTURE

3.1 Multiplier Architecture

The proposed Vedic squarer uses multiplier module for its computation. The proposed multiplier is designed using Urdhava Tiryakbhyam sutra. The Partial Products (PP) of 4×4 multiplier using Urdhava Tiryakbhyam sutra is shown in Fig.1. As shown in Fig.1 the PPs are grouped into four (n/2) multiplier modules and they are added using Carry Save Adder (CSA) to produce the final multiplier products. The block diagram of Urdhava multiplier is shown in Fig.2. Three input CSA is used in the architecture. First [(n-((n/2) +1)) to 0]-bit resultant product is obtained by taking [n-((n/2) +1) to 0]-bit result of first multiplier module directly. While the remaining resultant bits [(2n-1) to (n-(n/2))] is obtained by the sum produced by CSA. Since only CSA is used in the architecture there is a considerable amount of reduction in power consumption and overall propagation delay than the work proposed in [10].

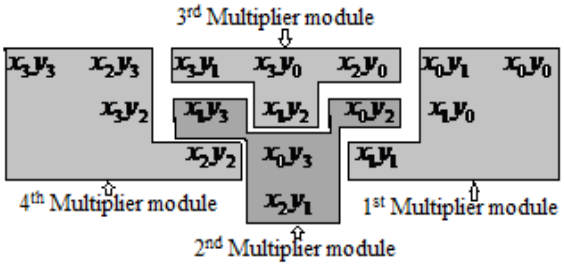


Fig.1. Partial products of 4×4 vedic multiplier using urdhava tiryakbhyam sutra.

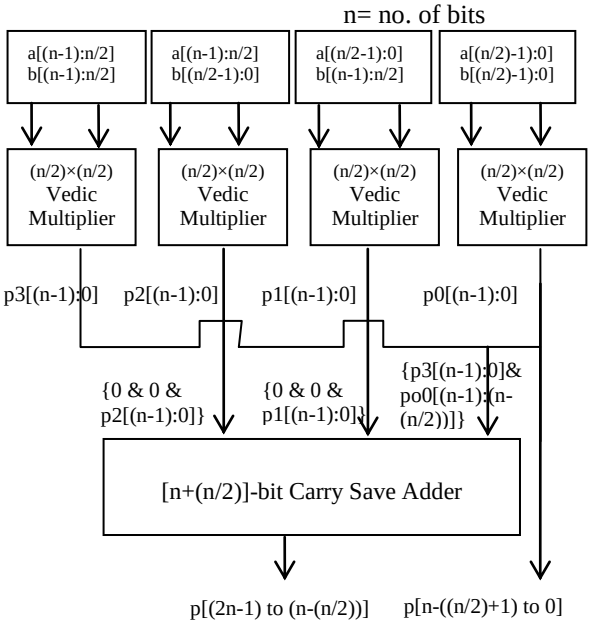


Fig.2. Block diagram of urdhava multiplier

3.2 Squarer Architecture

The squarer architecture presented here consists of three different optimizations. The three optimizations are based on Partial Product [PP] folding technique and PP re-arrangement technique. In each of the optimization the architecture is composed of two (n/2) bit square module and one (n/2) bit multiplier module and the results of these three modules is added using Carry Save Adder (CSA).
In the first optimization using the Urdhava Tiryakbhyam sutra the PPs of squarer is written as shown in Fig.3. The PPs are grouped into two (n/2) square modules and one (n/2) multiplier module. In the multiplier module we observe that PPs appear twice. Instead of using two multiplier modules, only one multiplier module is utilized by appending zero at the Least Significant Bit (LSB) side of the multiplier module result which is equivalent to adding two multiplier modules having similar PPs.

Vol 05, Article 09410, September 2014
http://ijves.com
International Journal of VLSI and Embedded Systems-IJVES
ISSN: 2249 – 6556

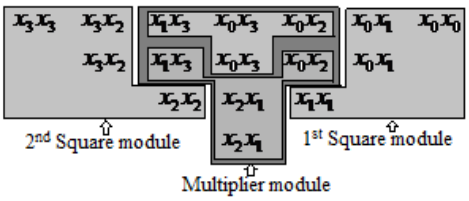


Fig.3. Partial products of 4×4 vedic squarer using urdhava tiryakbhyam sutra.

The results of multiplier and squarer modules are added using Carry-Look-Ahead (CLA) adder. The block diagram of optimization one is shown in Fig.4. As shown in the block diagram first [((n/2)-1) to 0]-bit of final product is obtained by directly taking the [((n/2)-1) to 0]-bit result of first squarer module (Least Significant Bit (LSB)-bits squarer). The result of the second squarer (Most Significant Bit (MSB)-bits squarer) is concatenated with remaining bits of first squarer and it is added with multiplier module results by concatenating ((n/2)-1) zeros at the MSB side and one zero at the LSB side. The sum produced by CLA adder gives the remaining [(2n-1) to (n/2)]-bit product.

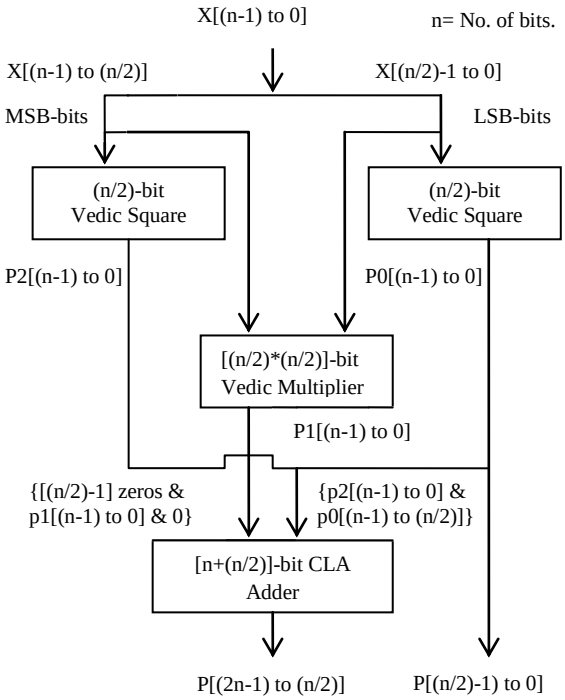


Fig.4. Block diagram of n-bit vedic squarer for optimization one.

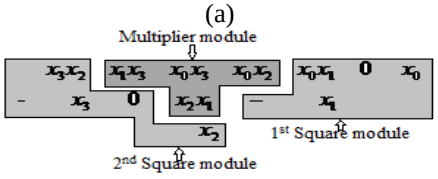
In the second optimization the PPs shown in Fig.3 are reduced as $X_iX_j = X_jX_i$. The PPs having similar identities can be combined as,

$$x_i x_j + x_j x_i = 2x_i x_j \qquad (1)$$

The reduced PPs using Equation (1) is shown in Fig.5. As done in optimization one the reduced PPs are grouped into two (n/2) square module and one (n/2) multiplier module. Since the PPs are reduced only one multiplier module is used and appending of '0' is eliminated as two multiplier module was used in optimization one which was done by appending '0' at the LSB side.

$$x_3 \quad 2x_3x_2 \quad 2x_1x_3 \quad 2x_0x_3 \quad 2x_0x_2 \quad 2x_0x_1 \quad 0 \quad x_0$$
$$x_2 \quad 2x_2x_1 \quad x_1$$

(a)



(b)

Fig.5. Reduced 4×4 vedic squarer partial products using folding technique: (a) before re-arrangement of partial products (b) after re-arrangement of partial products.

Vol 05, Article 09410, September 2014
http://ijves.com
International Journal of VLSI and Embedded Systems-IJVES
ISSN: 2249 – 6556

The block diagram of optimization two is shown in Fig.6. In optimization two [(n/2) to 0] bit of final product is directly taken from first squarer module. As done in optimization one the second square module result is concatenated with remaining bit of first squarer module and it is added with multiplier module result by appending '0' at MSB side.
In optimization two due to reduced PPs and using (n+(n/2)-1)-bit adder there is considerable amount of reduction in power consumption and propagation delay as compared to optimization one as (n+ (n/2)) bit CLA adder is used.

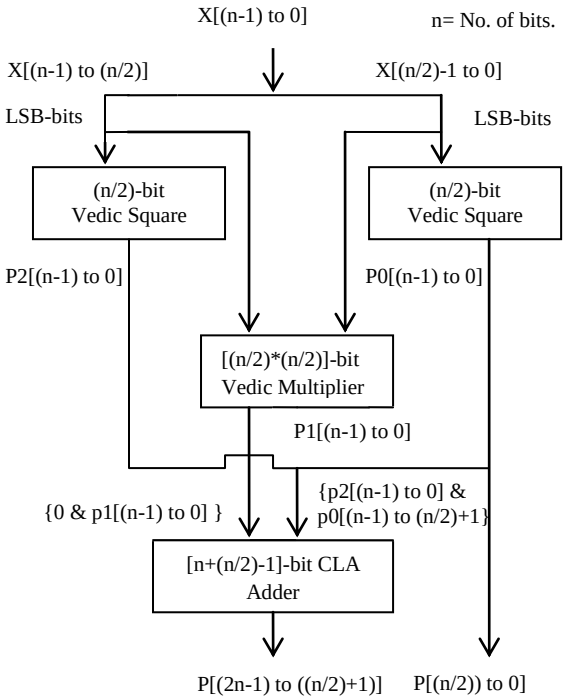


Fig.6. Block diagram of n-bit vedic squarer for optimization two and three.

In optimization three the PPs are further reduced using Equation (2), Equation (3) and Equation (4),

$$x_i x_j + x_i = 2x_i x_j + x_j - x_i x_j \quad (2)$$

$$= 2x_i x_j + x_i (1 - x_j) \quad (3)$$

$$= 2x_i x_j + x_i \overline{x_j} \quad (4)$$

As done in first two optimizations the PPs are grouped into two (n/2) square modules and one (n/2) multiplier module. As done in optimization two the results of squarer and multiplier are added using (n+ ((n/2)-1)-bit CLA adder. Due to further reduction in depth of PPs there is a significant reduction in power consumption as well as propagation delay as compared to first two optimizations. The reduced partial products for 4×4 Vedic squarer using optimization three is shown in Fig.7 and its block diagram is shown in Fig.6.

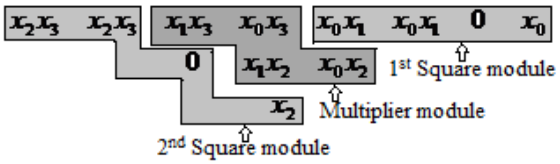


Fig.7. Reduced 4×4 vedic squarer partial products using Equation (4).

4. RESULTS AND DISCUSSION

Squarer for 4-bit, 8-bit and 16-bit were designed for both existing [10] and optimized methods. Three optimizations (optimization 1, optimization 2 and optimization 3) were performed in the optimized method. The designed Vedic squarer were simulated using Modelsim tool of version 6.5 for functional verification and synthesized using Cadence RTL compiler tool of version 12.10 with 180nm standard cell technology library and Xilinx tool of version 13.1 (Vertex 7 family with speed grade of -1) for dynamic power and propagation delay analysis. The simulation result for the proposed 4-bit, 8-bit and 16-bit Vedic squarer is shown in Fig. 8 to 10.

Vol 05, Article 09410, September 2014
http://ijves.com
International Journal of VLSI and Embedded Systems-IJVES
ISSN: 2249 – 6556

Simulation result in Fig. 8 to 10 is shown for various possible input combinations. As shown in Fig.8 'x' is a 4-bit input and 'p' is the output (square of input 'x') which results in 8-bit binary number. Similarly as shown in Fig.9 'x' is an 8-bit input and 'p' is the output which results in 16-bit binary number and in Fig.10 'x' is a 16-bit input and 'p' is the output which results in 32-bit binary number. Block diagram of 4-bit, 8-bit and 16-bit Vedic squarer for optimization three is shown in Fig. 11 to 13. As shown in block diagram 'x' is the input given to Vedic squarer module and 'p' is output of the Vedic squarer module, 'q1' and 'q2' are Vedic squarer modules, and 'm1' is Vedic multiplier module and 'l1', 'l2' are the adder module.

The performance of the proposed Vedic squarer design for 4-bit, 8-bit and 16-bit is shown in Table [1, 2, 3, 4, 5, and 6]. Percentage improvement in the Table [1, 2, 3, 4, 5, and 6] is calculated for optimization three Vedic squarer with respect to existing Vedic squarer [10]. The comparison results in Table [1, 2, 3, 4, 5 and 6] shows that the proposed Vedic squaring architecture not only consumes less power but also performs high speed than Vedic squarer design in [10].

Table 1. Synthesis Result of 4-bit Squarer in ASIC flow

Parameters	Propagation Delay (ns)	Dynamic Power (mw)
Existing [10]	2.770	0.0033
Optimization-1	2.100	0.0022
Optimization-2	1.727	0.0021
Optimization-3	1.442	0.0018
% Improvement	47.94	45.45

Table 2. Synthesis Result of 8-bit Squarer in ASIC flow

Parameters	Propagation Delay (ns)	Dynamic Power (mw)
Existing [10]	7.332	0.0230
Optimization-1	4.218	0.0176
Optimization-2	4.064	0.0170
Optimization-3	3.780	0.0165
% Improvement	48.44	28.26

Table 3. Synthesis Result of 16-bit Squarer in ASIC flow

Parameters	Propagation Delay (ns)	Dynamic Power (mw)
Existing [10]	15.464	0.1287
Optimization-1	8.963	0.1013
Optimization-2	8.946	0.1002
Optimization-3	8.661	0.0994
% Improvement	43.99	22.76

Table 4. Synthesis Result of 4-bit Squarer in FPGA flow

Parameters	Propagation Delay (ns)	Dynamic Power (mw)
Existing [10]	6.529	4.70
Optimization-1	4.016	4.10
Optimization-2	4.010	5.20
Optimization-3	2.959	4.36
% Improvement	54.67	7.23

Table 5. Synthesis Result of 8-bit Squarer in FPGA flow

Parameters	Propagation Delay (ns)	Dynamic Power (mw)
Existing [10]	13.018	13.56
Optimization-1	7.664	13.42
Optimization-2	7.521	13.50
Optimization-3	7.511	13.10
% Improvement	42.30	3.39

Vol 05, Article 09410, September 2014
http://ijves.com
International Journal of VLSI and Embedded Systems-IJVES
ISSN: 2249 – 6556

Table 6. Synthesis Result of 16-bit Squarer in FPGA flow

Parameters	Propagation Delay (ns)	Dynamic Power (mw)
Existing [10]	21.689	31.80
Optimization-1	13.723	28.68
Optimization-2	13.505	31.06
Optimization-3	13.505	28.91
% Improvement	37.73	9.08

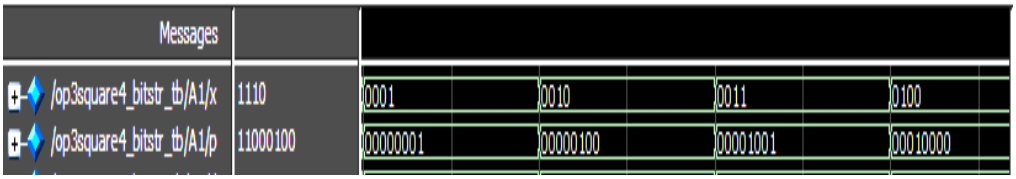


Fig.8. Simulation results of 4-bit squarer

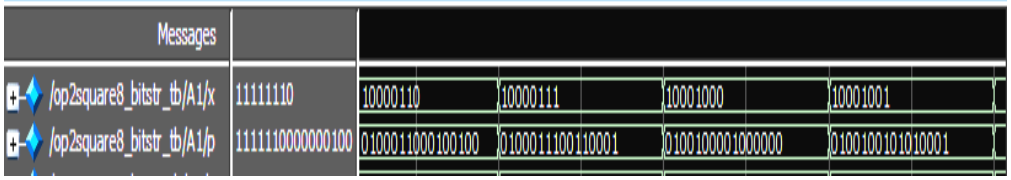


Fig.9. Simulation results of 8-bit squarer

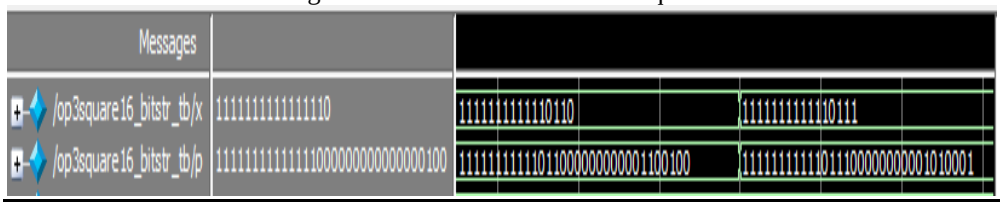


Fig.10. Simulation results of 16-bit squarer

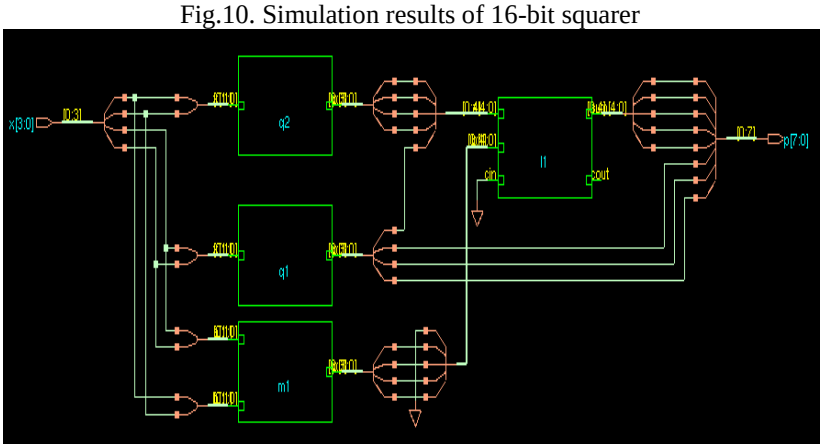


Fig.11. Block diagram of optimization three 4-bit squarer

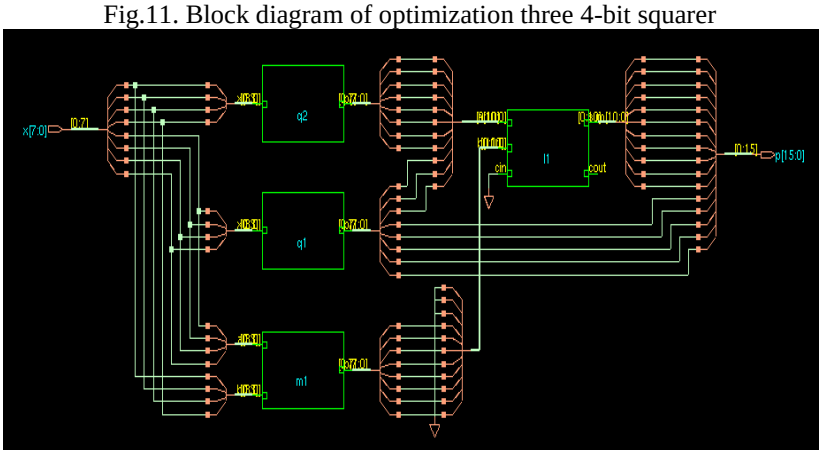


Fig.12. Block diagram of optimization three 8-bit squarer

Vol 05, Article 09410, September 2014
http://ijves.com
International Journal of VLSI and Embedded Systems-IJVES
ISSN: 2249 – 6556

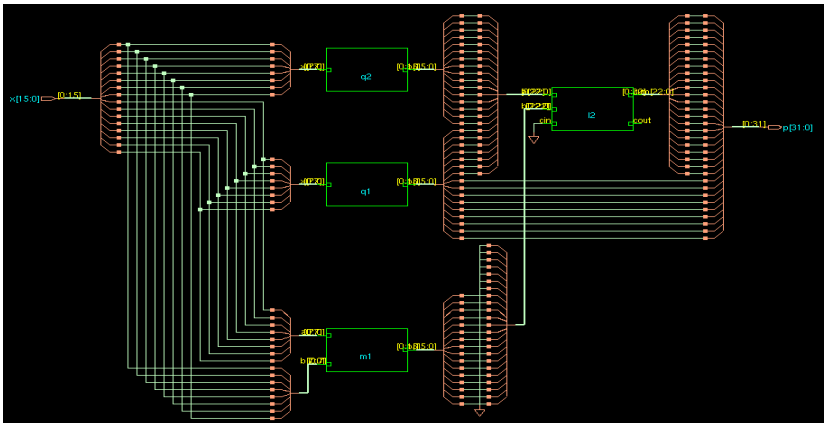


Fig.13. Block diagram of optimization three 16-bit squarer

5. CONCLUSION

The focus of this work is to achieve optimized and realistic squarer architecture. The overall performance of the proposed squarer is proved to exhibit improved efficiency in terms of propagation delay and dynamic power reduction. Due to factors of low power and high speed the proposed squarer can be used for DSP and cryptography applications which involve time consuming processes like squaring. The proposed squarer design is simulated and synthesized for 4-bit, 8-bit and 16-bit and it can be extended for higher number of bits of unsigned numbers. The tabulated result shows that for the optimized 16-bit Vedic squarer the overall propagation delay is reduced by 43.99% and dynamic power by 22.76% for ASIC flow and similarly 37.73% and 9.08% for FPGA flow when compared with existing Vedic squarer design [10].

REFERENCES

[1] Johnny Pihl and Einar J (1996), A multiplier and squarer generator for high performance DSP applications, *IEEE 39th Midwest Symposium on Circuits and System*, Ames, pp. 109-112.

[2] Akhalesh K, Itawadiya, Rajesh Mahle, Vivek Patel and Dadan Kumar (2013), Design a DSP Operations using vedic mathematics, *IEEE International Conference on Communications and Signal Processing (ICCSP)*, Melmaruvathur, pp. 897-902.

[3] Himanshu Thapliyal and M.B Srinivas (2005), An efficient method of elliptic curve encryption using ancient Indian vedic mathematics, *IEEE 48th Midwest Symposium on Circuits and Systems*, Covington, pp. 826-828.

[4] S.Kumaravel and Ramalatha Marimuthu (2007), VLSI implementation of high performance RSA algorithm using vedic mathematics, *IEEE Conference on Computational Intelligence and Multimedia Applications*, Sivakasi, pp. 126-128.

[5] www.vedicmaths.com

[6] A.P Nicholas, J Pickles and K Williams (1982), *Introductory Lectures on Vedic Mathematics*, Polytechnic of North London.

[7] A.P Nicholas, K.R Williams and J Pickles (2010), *Applications of the Vedic mathematics Sutra: Vertically and Crosswire*, Inspiration books, Third revised edition, The Vedic mathematics research group.

[8] Parth Mehta and Dhanashri Gawali (2009), Conventional versus vedic mathematical method for hardware implementation of a multiplier, *IEEE International Conference on Advances in Computing, Control, Telecommunication Technologies*, Trivandrum, pp. 640-642.

[9] Abhijeet Kumar, Dilip Kumar and Siddhi (2012), Hardware implementation of 16*16 bit multiplier and square using vedic mathematics, *International Conference on Signal, Image and Video Processing (ICSIVP)*, pp. 309-314.

[10] kabiraj Sethi and Rutuparna Panda (2012), An improved squaring circuit for binary numbers, *International journal of advanced computer science and applications*, vol.3 , No.2, pp. 111-105.