International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2014): 5.611

Design and VLSI Implementation of N X N Binary Multiplier Using Successive Approximation of (N-1) X (N-1) Binary Multipliers

K. Indumathi[1], M. Nisha Angeline

PG Scholar, Master of Engineering in VLSI Design, Velalar College of Engineering and Technology, Thindal, Erode, India

M.E.,(Ph.D), Assistant Professor (Sl.Gr.), Department of ECE, Velalar College of Engineering and Technology, Thindal, Erode, India

Abstract: ***In VLSI technology, power consumption and delay becomes a major problem in multipliers. To reduce these issues we propose a new multiplier algorithm that combines numerical transformation and shift and add technique. In this design N X N bitmultiplication is done by using successive approximation of (N-1) X (N-1) bit multiplier. The strength of the multiplication is reduced by weight reduction technique.The performance of N-bit multiplier using successive approximation of N-1 bit multiplier is compared with the existing techniques and evaluated through simulation in order to highlight the speed superiority by reducing the number of components and interconnections. N-bit successive approximation is an excellent choice for low area and high speed applications. All the above mentioned multipliers are coded in VHDL and simulated in ModelSim and synthesized in EDA tool Xilinx_ISE 9.2i. This method is suitable for higher order bits. The analysis of power report is also presented here the proposed design is suitable for high speed, low area applications.***

Keywords: weight reduction, numerical tansformation, successive approximation,shift and add method.

1. Introduction

The multiplier is the fundamental component of most of the latest Digital signal Processors (DSP'S). By power investigation, the multipliers consume more power in these chips. The computational speed depends on the critical path of the circuit. The array multiplier is one of the most familiar architectures. But this has the disadvantage of non-uniform path delays in the structure. Other than array multipliers, many authors invented different multiplication algorithms and architectures.

In [1], the author suggested a method for improving the speed of array multiplier by equating delays among carry and sum pay-outs. In [1], delay elements were introduced to compensate the delay imbalance in the structure. In [2], multiplication was done in two steps. First, the complex multiplication operations were decomposed into elementary shift-and-add operations. Second, number transformation was applied to the original system to obtain equivalent architectures with different multiplier coefficients. In [3], the researchers proposed high speed multiply-accumulate structures based on the Baugh-Wooley Algorithm (BWA). The multipliers were implemented with Modified Booth Algorithm (MBA).

In [4], the design of bit parallel GF (2^m) multiplier was proposed. The Mastrovito multiplier was modified and sharing sub-expression was exploited in the computation of the product matrix to reduce the complexity. In [5], the multiplier was designed using separated multiplication technique to use in digital image signal processing with minimum power dissipation. Due to high spatial redundancy in 2-D image data, the multiplication is divided into higher and lower parts. The calculated values of the higher bits are stored in the memory cells and it can be reused when a cache hit occurs.

In [6], radix-4 modular multiplier was proposed to be used in RSA (RIVEST, SHAMIR AND ADLEMAO). The booth algorithm is modified and radix-4 cellular array multiplier was designed in bit level and digit level. Here the low power multipliers were investigated by minimizing the switching activities of partial products according to effective dynamic ranges of input data. Many researches [7],[8],[9] and [10] have been done to design low power, low area multiplier. The conventional shift and add multiplier is modified to reduce its energy consumption. By pass zero during addition was implemented in [11]. This results in an average power reduction by 30%.

Many papers [12], [13], [14] and [15] were focussed on the design of low power array multiplier. They followed signal flow optimization in [3:2] adder array for the linear partial product reduction, left- to – right leapfrog (LRLF) structure and upper / lower splitting structure. In [13] the authors discussed spurious power suppression technique (SPST).

The remainder of the paper is organized as follows. In section II, the existing methods and their problems are discussed. In section III, the redundant calculation and shift and add are discussed. In section IV the proposed architecture with different conditional modes are discussed. Finally, the results and concluding remark are given in section V.

2. Existing Methods

The arithmetic operations addition, subtraction, multiplication and division are used in Digital computers, multi-media applications, Digital Signal Processing, etc. Mainly, multiplication in Digital Signal Processing is used to perform filtering, Convolution and correlation operations. Multiplication can be implemented in two ways either serially or in parallel. The widely used parallel algorithm is

array multiplication. For NxN bit array multiplier, N(N-1) full adders and N^2 AND gates are required. By reducing 1 bit, required FA is (N-1)(N-2) and AND gate is (N-1)(N-1). The percentage reduction is calculated as follows,

Reduction in Full Adder $= \frac{N(N-1)-(N-1)(N-2)}{N(N-1)} X100$

$= \frac{N^2 - N - (N^2 - 3N + 2)}{N(N-1)}$ 100

$= \frac{2N-2}{N(N-1)}$ 100%

Reduction in AND Gate $= \frac{N^2-(N-1)^2}{N^2}$ 100

$= \frac{N^2-(N^2-2N+1)}{N^2}$ 100

$= \frac{2N-1}{N^2}$ 100%

The critical path for NxN bit multiplier is calculated by $[(N-1) + (N-2)]*T_{carry} + (N-1)*T_{sum}+T_{AND}$. By using (N-1)x(N-1) multiplier for N bit multiplication, the critical path is reduced to $[(N-2)+(N-3)]*T_{carry}+(N-2)*T_{sum}+T_{AND.}$

The main advantage is its regular structure. It is having identical cells and generates partial products simultaneously and accumulates same time. But the disadvantage is that it requires large number of logic gates. Another, important widely used multiplier is shift-and-add multiplier. Main problem of this multiplier is that power dissipation is high due to high switching activity. The major sources of switching activities are summarized as below: Shifting of the ‗B' register, activity in the counter, activity in the adder, switching between ‗0' and ‗A' in the multiplexer, activity in the multiplexer select, shifting of the partial product register.

3. Numerical Transformation

Numerical transform is a technique used for manipulating the data into another equivalent form. Normally, number splitting, sharing the sub-expression, constant multiplication etc. can be used. In [8], the number splitting is used in constant multiplication for shift-and-add decomposition. The constant multiplication is used in the applications like matrix multiplication, FIR filter, and IIR filter.

In the proposed paper, multiplication of two N-bit numbers is derived by N-1 bit multiplication using weight reduction technique. Here the strength of the multiplication is reduced by reducing the maximum weight (2^{N-1}) of the number. The numerical transform technique is used here to reduce the weight of the number. Hence the computational complexity of the multiplier can be reduced. The concept is explained through an example. Let X and Y be the 4 bit numbers. X can be represented as,

$X= X_{N-1}\ X_{N-2}\ldots\ldots\ldots\ldots\ X_1\ X_0$

And similarly

$$Y= Y_{N-1}\ Y_{N-2}\ldots\ldots\ldots\ldots Y_1\ Y_0 \quad (1)$$

Here, the weight of the MSB (X_{N-1} and Y_{N-1}) has to be reduced to 2^{N-2}. The weight reduction is achieved by deriving redundant using numerical transform. The weight reduction is done in three ways. Three algorithms are developed based on this. They are, Mode I- Both numbers may be greater than or equal to 2^{N-1}, Mode II – Both may be less than 2^{N-1} and Mode III- one number may be greater than or equal to 2^{N-1} and the other may be less than $2^{N-1.}$ By considering these modes, four architectures are proposed here. The number is split based on 2^{N-1}. The number is divided into two parts, namely the maximum weight of the number and the redundant. The redundant may be positive or negative. If the number is greater than 2^{N-1}, the redundancies will be positive. If the number is less than 2^{N-1}, the redundancies will be negative.

For example, considering the numbers

$$X= 2^{N-1}+(X-2^{N-1}) \quad (2)$$

$$X=2^{N-1} – (2^{N-1} –X) \quad (3)$$

The second term in (2) is knows as positive redundancy. The redundant calculated in (3) is negative. The redundant is derived by subtracting the number from the maximum weight 2^{N-1}. If the number is less than 2^{N-1}, subtracting the number from 2^{N-1} will give the redundant. In this case, the redundant is negative. If the number is greater than 2^{N-1}, the redundancies will be positive and it will be derived by subtracting 2^{N-1} from the number. By calculating redundancy, the N bit number is reduced into N-1 bits. The N-1 bit multiplier is used to multiply these redundant. The redundant calculation and the shift–and-add method are discussed in section III.

4. Proposed Architecture

In this section, the proposed algorithms are presented and their architectures for four different modes are given. The results are proven theoretically in this section. Three modes are discussed in detail below.
Mode I- Both numbers are greater than 2^{N-1}
Mode II- Both numbers are less than 2^{N-1}
Mode III- Only one number is greater than 2^{N-1}

4.1 Algorithm for Mode I

Input : A, B (N bits)
Output: P (2N bits)
Step 1: Given A and B are greater than 2^{N-1}. Subtract 2^{N-1} from A & B to derive redundant. (Consider c1 and c2)
Step2: Multiply the redundant c1 and c2 using successive N-1xN-1 bit multiplier. i.e. s2=c1*c2.
Step3: shift the input A left side by N-1 times. (m2=A<<N-1).
Step4: Shift the redundant c2 by N-1 times (s3=c2<<N-1)
Step5: Add all the components to derive the product P=s2+s3+s4.

The architecture for Mode I is shown in Fig.2. Here both redundant are positive. The product of the numbers is calculated as follows
$P=AB= 2^{N-1}*A+C_1*C_2+2^{N-1}*C_2$

$$= (A<<N-1)+(C1*C2)+(C2<<N-1) \quad (4)$$

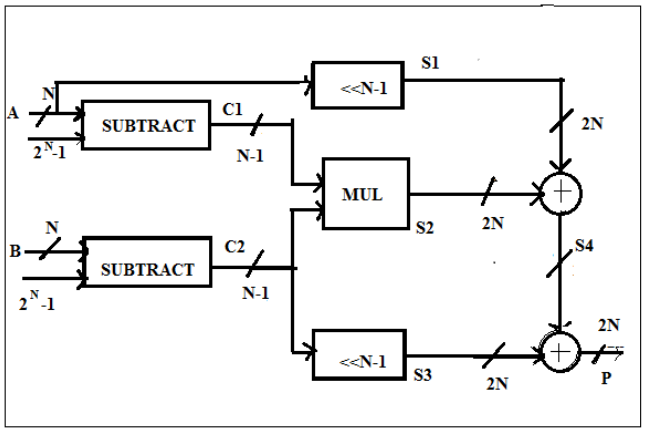

Figure 1: Architecture of multiplier for Mode I

4.2 Algorithm for Mode II

Input : A, B (N bits)
Output: P (2N bits)
Step 1: Given A and B are less than 2^{N-1}. Subtract A& B from $2^{N-1}$to derive redundant. (Consider c1 and c2)
Step2: Multiply the redundant c1 and c2 using successive N-1xN-1 bit multiplier. i.e. s2=c1*c2.
Step3: shift the input A left side by N-1 times. (c2=A<<N-1).
Step4: Shift the redundant r2 by N-1 times (s3=r2<<N-1)
Step5: Add all the components to derive the product
P=s2+s4-s3.

The architecture for Mode II is shown in Fig.3. Here both redundant are negative. The product of the numbers is calculated as follows

$$P=AB = 2^{N-1}*A+c_1*c_2-2^{N-1}*c_2$$

$$= (A<<N-1)+(c1*c2)-(c2<<N-1) \quad (5)$$

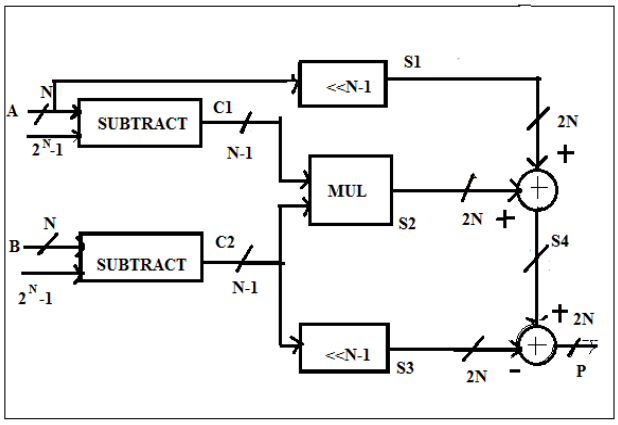

Figure 2: Architecture of multiplier for Mode II

4.3 Algorithm for Mode III

Input : A, B (N bits)
Output: P (2N bits)
Step 1 : Given A or B may be less than 2^{N-1}. A is considered to be less than B. If B>A, interchange A & B.
Step 2: If A_{N-1} = 1, subtract 2^{N-1} from A or subtract A from $2^{N-1}$to derive redundant. (Consider c1). If B_{N-1} = 1, subtract 2^{N-1} from B or subtract B from 2^{N-1} to derive redundant (c2)
Step3: Multiply the redundant c1 and c2 using successive N-1xN-1 bit multiplier. i.e. m1=c1*c2.
Step4: shift the input A left side by N-1 times. (s3=A<<N-1).
Step5: Shift the redundant r2 by N-1 times (s4=r2<<N-1)
Step6: Add all the components to derive the product
P=s2-s4+s3

The architecture for Mode III is shown in Fig.4. Here one redundant is positive and the other is negative. The product of the numbers is calculated as follows

$$P=AB = 2^{N-1}*A-c_1*c_2+2^{N-1}*c_2$$

$$=(A<<N-1)-(c1*c2)+(c2<<N-1) \quad (6)$$

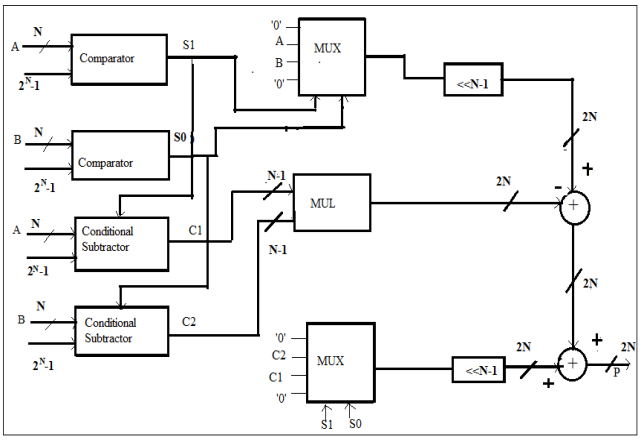

Figure 3: Architecture of multiplier for Mode III

The comparator produces 1 output if the number is greater than or equal to 2^{N-1}. The conditional subtractor is designed to produce the output such that if the control signal is 0, the output is calculated by subtracting the number from 2^{N-1}, otherwise 2^{N-1} is subtracted from input A or B.

The combined structure is shown in Fig.4. Using the combined structure, the number in any mode can be calculated. This structure is similar to the structure shown in Fig.3. Here the control signal to select adder/subtractor is generated by simple logic gate.

The above structure can be simplified by replacing comparator. The comparator produces 1 output if the number is greater than or equal to 8. Instead of designing comparator, it can be compensated by considering MSB bits of the number. If MSB=1,it implies that the number is greater than 2^{N-1}.

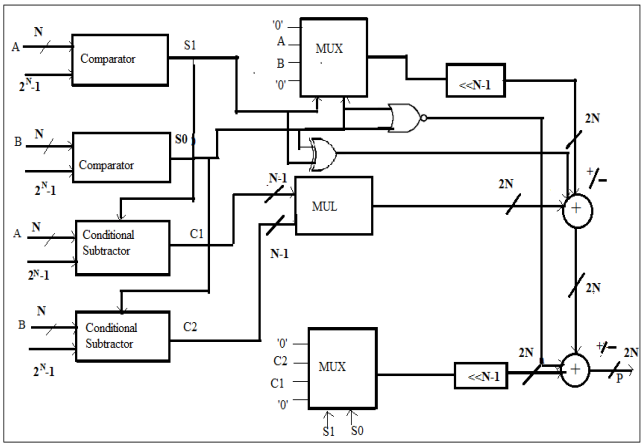

Figure 4: Combined architecture of multiplier.

5. Results and Discussion

For comparison, we have considered both conventional methods and the proposed multiplier for different bit values in VHDL. The VHDL codes are implemented using Xilinx Vertex FPGA. The power analysis of the gate level structure and delay calculations is conducted using Xilinx ISE tool. The array multiplier is widely used due to its linear structure. It is advantageous for minimum number of bits. The results of table 1 and 2 show the values of the proposed multiplier for different bit values. The multiplier unit is the important basic unit in most of the applications. The array multiplier and shift-and-add multipliers are widely used in most of the applications. In the table 2, the proposed multiplier is compared with existing array multiplier and shift-and–add multiplier for the standard bit sizes 4,8,16 and 32. From the result, it is clear that the proposed method is well suited for higher order bits. In the lower bits, the proposed method doesn‘t show advantageous result. For the bit size 2, the delay is high for the proposed method. But for the higher order bits, the delay is reduced. Similarly, the number of LUTs is also increased for different bit values. Comparatively, the proposed method reduces the area to the extent of 71%. The delay is reduced to the extent of 77%. By reducing the delay, the speed can also be increased.

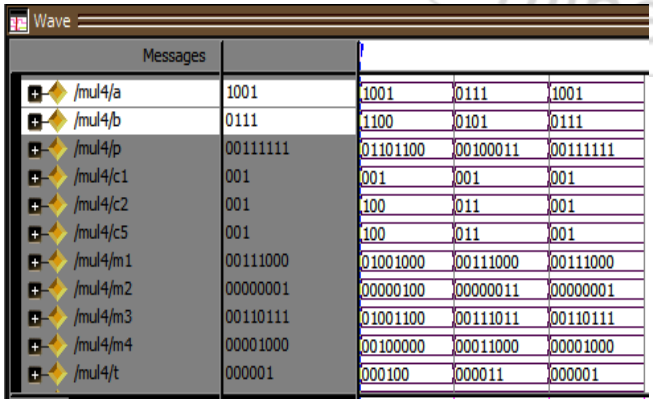

Figure 5: Simulation Results of proposed structure

Table 1: Analysis table for various multipliers comparing Delay

Array Multiplier	15.269	31.111	62.437	123.387
Shift and add multiplier	15.677	33.840	63.089	124.112
Braun Multiplier	13.088	23.331	62.437	127.776
Wallace Tree Multiplier	12.756	22.863	44.258	87.776
Vedic Multiplier	11.752	21.564	41.684	82.289
Proposed combined structure	10.578	10.315	38.617	79.213

Table 2: Analysis table for various multipliers comparing power consumption

Array Multiplier	0.298	0.312	0.368	0.440
Shift and add multiplier	0.310	0.368	0.501	0.636
Braun Multiplier	0.113	0.149	0.406	0.706
Wallace Tree Multiplier	0.113	0.154	0.375	0.706
Vedic Multiplier	0.123	0.142	0.250	0.489
Proposed combined structure	0.109	0.139	0.214	0.397

Table 3: Analysis table for various multipliers comparing No. of LUTs

Array Multiplier	30	125	511	2033
Shift and add multiplier	26	123	508	2044
Braun Multiplier	33	77	294	1194
Wallace Tree Multiplier	27	122	512	2077
Vedic Multiplier	20	79	501	1932
Proposed combined structure	17	75	255	1112

6. Conclusion and Future Work

In this paper, multiplier based on numerical transformations, i.e. number splitting is designed and the same algorithm is converted into hardware. Based on algorithmic concepts, four modules are proposed. All the modules are implemented Xilinx Vertex FPGA. Comparatively, the proposed method occupies less area with minimum delay. The proposed method is not advantageous for minimum number of bits. But the area and delay calculation is very optimal in higher order bits. This method is suitable for higher order bits. The architecture can also be modified to improve the speed further.

References

[1] Shivaling S. Mahant-Shetti, Poras T. Balsara, ―High Performance Low Power Array Multiplier Using Temporal Tiling", IEEE Transactions on Very Large Scale Integration (VLSI) systems, 1999, 7, (1), pp. 121-124.

[2] Huy T. Nguyen and AbhijitChatterjee," Number-Splitting with Shift-and-Add Decomposition for Power and Hardware Optimization in Linear DSP Synthesis", IEEE Transactions on Very Large Scale Integration (VLSI) systems, 2000, 8,(4), pp.419-424.

[3] Fayez Elguibaly "A Fast Parallel Multiplier-Accumulator Using the Modified Booth Algorithm, IEEE Transactions on Circuits and Systems—II: Analog

and Digital Signal Processing, 2000, 47, (9), pp 902-908.

[4] Tong Zhang, Student Member, Keshab K. Parhi, ―Systematic Design of Original and Modified Mastrovito Multipliers for General Irreducible Polynomials", IEEE Transactions on Computers, 2001, 50, (7), pp.734-749.

[5] Chang-Young Han, Hyoung-Joon Park, and Lee-Sup Kim, ―A Low-Power Array Multiplier Using Separated Multiplication Technique", IEEE Transactions on Circuits and Systems—II: Analog and Digital Signal Processing, 2001, 48,(9), pp.866-871.

[6] Bah-HweeGwee , Joseph S. Chang, Yiqiong Shi, Chien-Chung Chua, and Kwen-Siong Chong , ―A Low-Voltage Micropower Asynchronous Multiplier With Shift–Add Multiplication Approach, IEEE Transactions on Circuits and Systems—I: Regular Papers, 2009,56, (7), pp 1349-1359.

[7] Kyung-Ju Cho, Kwang-Chul Lee, Jin-Gyun Chung and Keshab K. Parhi "Design of Low-Error Fixed-Width Modified Booth Multiplier", IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2004, 12, (5), pp. 522-531.

[8] Jiun-Ping Wang, Shiann-RongKuang , and Shish-Chang Liang, ―iJun-Ping Wang, Shiann-RongKuangand Shish-Chang Liang", IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2011, 19, (1), pp. 52-60.

[9] Chip-Hong Chang and Ravi Kumar Satzoda, ―A Low Error and High Performance Multiplexer-Based Truncated Multiplier ―, Transactions on Very Large Scale Integration (Vlsi) Systems,2010, 18, (12), pp.1767-1771.

[10] M. Mottaghi-Dastjerdi, A. Afzali-Kusha, and M. Pedram," BZ-FAD: A Low-Power Low-Area Multiplier Based on Shift-and-Add Architecture", IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2009, 17,(2), pp. 302-306.

[11] Jiun-Ping Wang, Shiann-RongKuang , and Shish-Chang Liang, ―iJun-Ping Wang, Shiann-RongKuangand Shish-Chang Liang", IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2011, 19, (1), pp. 52-60.

[12] Zhijun Huang, Member, IEEE, and Milo_s D. Ercegovac, Fellow, IEEE, ―High-Performance, Low-Power Left-to-Right Array Multiplier Design", IEEE Transactions on Computers, 2005, 54,(3), pp. 272-283.

[13] Kuan-Hung Chen and Yuan-Sun Chu, ―A Low-Power Multiplier With the Spurious Power Suppression Technique", IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2007, 15, (7), pp.846-850.

[14] Shiann-RongKuang , and Jiun-Ping Wang, ―Design of Power-Efficient Configurable Booth Multiplier", IEEE Transactions on Circuits and Systems—I: Regular Papers, 2010, 57,(3), pp.568-580.

[15] Shin-Kai Chen, Chih-Wei Liu , Tsung-Yi Wu, and An-Chi Tsai, ―Design and Implementation of High-Speed and Energy-Efficient Variable-Latency Speculating Booth Multiplier (VLSBM)", IEEE Transactions on Circuits and Systems—I: Regular Papers, 2013, 60, (10), pp.2631-2643.