IOSR Journal of Electronics and Communication Engineering (IOSR-JECE)
e-ISSN: 2278-2834,p- ISSN: 2278-8735.Volume 8, Issue 5 (Nov. - Dec. 2013), PP 17-25
www.iosrjournals.org

Implementation of High Speed Modified Booth Multiplier and Accumulator (Mac) Unit

* Chinababu Vanama[1], M.Sumalatha[2]
[1] *PG Student (M. Tech), Dept. of ECE, Chirala Engineering College, Chirala., A.P, India.*
[2] *Associate Professor, Dept. of ECE, Chirala Engineering College, Chirala., A.P, India.*

Abstract: *The most effective way to increase the speed of a multiplier is to reduce the number of the partial products because multiplication precedes a series of additions for the partial products. To reduce the number of calculation steps for the partial products, MBA algorithm has been applied mostly where CSA has taken the role of increasing the speed to add the partial products. To increase the speed of the MBA algorithm, many parallel multiplication architectures have been researched. A modified booth multiplier has been designed which provides a flexible arithmetic capacity and a tradeoff between output precision and power consumption due to using of SPST architecture. Moreover, the ineffective circuitry can be efficiently deactivated, thereby reducing power consumption and increasing speed of operation. The experimental results have shown that the proposed multiplier outperforms the conventional multiplier in terms of power and speed of operation. In this paper we used Xilinx-ISE tool for logical verification, and further synthesizing it on Xilinx-ISE tool using target technology and performing placing & routing operation for system verification.*

I. Introduction

Fast multipliers are essential parts of digital signal processing systems. The speed of multiply operation is of great importance in digital signal processing as well as in the general purpose processors today, especially since the media processing took off. In the past multiplication was generally implemented via a sequence of addition, Subtraction, and shift operations. Multiplication can be considered as a series of repeated additions. The number to be added is the multiplicand, the number of times that it is added is the multiplier, and the result is the product. Each step of addition generates a partial product. In most computers, the operand usually contains the same number of bits. When the operands are interpreted as integers, the product is generally twice the length of operands in order to preserve the information content. This repeated addition method that is suggested by the arithmetic definition is slow that it is almost always replaced by an algorithm that makes use of positional representation. It is possible to decompose multipliers into two parts. The first part is dedicated to the generation of partial products, and the second one collects and adds them.

The basic multiplication principle is two fold, i.e. evaluation of partial products and accumulation of the shifted partial products. It is performed by the successive Addition's of the columns of the shifted partial product matrix. The 'multiplier' is successfully shifted and gates the appropriate bit of the 'multiplicand'. The delayed, gated instance of the multiplicand must all be in the same column of the shifted partial product matrix. They are then added to form the product bit for the particular form. Multiplication is therefore a multi operand operation. To extend the multiplication to both signed and unsigned numbers, a convenient number system would be the representation of numbers in two's complement format.

Multipliers are key components of many high performance systems such as FIR filters, microprocessors, digital signal processors, etc. A system's performance is generally determined by the performance of the multiplier because the multiplier is generally the slowest clement in the system. Furthermore, it is generally the most area consuming. Hence, optimizing the speed and area of the multiplier is a major design issue. However, area and speed are usually conflicting constraints so that improving speed results mostly in larger areas. As a result, whole spectrums of multipliers with different area-speed constraints are designed with fully parallel processing. In between are digit serial multipliers where single digits consisting of several bits are operated on. These multipliers have moderate performance in both speed and area. However, existing digit serial multipliers have been plagued by complicated switching systems and/or irregularities in design. Radix 2^n multipliers which operate on digits in a parallel fashion instead of bits bring the pipelining to the digit level and avoid most of the above problems. They were introduced by M. K. Ibrahim in 1993. These structures are iterative and modular. The pipelining done at the digit level brings the benefit of constant operation speed irrespective of the size of' the multiplier. The clock speed is only determined by the digit size which is already fixed before the design is implemented.

Booth's multiplication algorithm

Booth's algorithm can be implemented by repeatedly adding (with ordinary unsigned binary addition) one of two predetermined values *A* and *S* to a product *P*, then performing a rightward arithmetic shift on *P*. Let

m and **r** be the multiplicand and multiplier, respectively; and let x and y represent the number of bits in **m** and **r**.

1. Determine the values of A and S, and the initial value of P. All of these numbers should have a length equal to $(x + y + 1)$.
1. A: Fill the most significant (leftmost) bits with the value of **m**. Fill the remaining $(y + 1)$ bits with zeros.
2. S: Fill the most significant bits with the value of $(-\mathbf{m})$ in two's complement notation. Fill the remaining $(y + 1)$ bits with zeros.
3. P: Fill the most significant x bits with zeros. To the right of this, append the value of **r**. Fill the least significant (rightmost) bit with a zero.
2. Determine the two least significant (rightmost) bits of P.
1. If they are 01, find the value of $P + A$. Ignore any overflow.
2. If they are 10, find the value of $P + S$. Ignore any overflow.
3. If they are 00, do nothing. Use P directly in the next step.
4. If they are 11, do nothing. Use P directly in the next step.
3. Arithmetically shift the value obtained in the 2nd step by a single place to the right. Let P now equal this new value.
4. Repeat steps 2 and 3 until they have been done y times.
5. Drop the least significant (rightmost) bit from P. This is the product of **m** and **r**.

Example
Find $3 \times (-4)$, with **m** = 3 and **r** = −4, and $x = 4$ and $y = 4$:

- m = 0011, -m = 1101, r = 1100
- A = 0011 0000 0
- S = 1101 0000 0
- P = 0000 1100 0
- Perform the loop four times :

1. P = 0000 110**0 0**. The last two bits are 00.
   - P = 0000 0110 0. Arithmetic right shift.
2. P = 0000 011**0 0**. The last two bits are 00.
   - P = 0000 0011 0. Arithmetic right shift.
3. P = 0000 001**1 0**. The last two bits are 10.
   - P = 1101 0011 0. P = P + S.
   - P = 1110 1001 1. Arithmetic right shift.
4. P = 1110 100**1 1**. The last two bits are 11.
   - P = 1111 0100 1. Arithmetic right shift.

- The product is 1111 0100, which is −12.

II. Different Multipliers

Binary Multiplication

In the binary number system the digits, called bits, are limited to the set. The result of multiplying any binary number by a single binary bit is either 0, or the original number. This makes forming the intermediate partial-products simple and efficient. Summing these partial-products is the time consuming task for binary multipliers. One logical approach is to form the partial-products one at a time and sum them as they are generated. Often implemented by software on processors that do not have a hardware multiplier, this technique works fine, but is slow because at least one machine cycle is required to sum each additional partial-product. For applications where this approach does not provide enough performance, multipliers can be implemented directly in hardware.

Hardware Multipliers

Direct hardware implementations of shift and add multipliers can increase performance over software synthesis, but are still quite slow. The reason is that as each additional partial-product is summed a carry must be propagated from the least significant bit **(LSB)** to the most significant bit **(MSB).** This carry propagation is time consuming, and must be repeated for each partial product to be summed.

One method to increase multiplier performance is by using encoding techniques to reduce the the number of partial products to be summed. Just such a technique was first proposed by Booth [BOO 511. The original Booth's algorithm ships over contiguous strings of l's by using the property that: 2" + 2(n-1) + 2(n-2) + . . . + 2hm) = 2(n+l) - 2(n-m). Although **Booth's algorithm** produces at most N/2 encoded partial products from an N bit operand, the number of partial products produced varies. This has caused designers to use modified

versions of Booth's algorithm for hardware multipliers. Modified **2-**bit Booth encoding halves the number of partial products to be summed.

Since the resulting encoded partial-products can then be summed using any suitable method, modified 2 bit Booth encoding is used on most modern floating-point chips LU 881, MCA 861. A few designers have even turned to modified 3 bit Booth encoding, which reduces the number of partial products to be summed by a factor of three IBEN 891. The problem with 3 bit encoding is that the carry-propagate addition required to form the 3X multiples often overshadows the potential gains of 3 bit Booth encoding.

To achieve even higher performance advanced hardware multiplier architectures search for faster and more efficient methods for summing the partial-products. Most increase performance by eliminating the time consuming carry propagate additions. To accomplish this, they sum the partial-products in a redundant number representation. The advantage of a redundant representation is that two numbers, or partial-products, can be added together without propagating a carry across the entire width of the number. Many redundant number representations are possible. One commonly used representation is known as carry-save form. In this redundant representation two bits, known as the carry and sum, are used to represent each bit position. When two numbers in carry-save form are added together any carries that result are never propagated more than one bit position. This makes adding two numbers in carry-save form much faster than adding two normal binary numbers where a carry may propagate. One common method that has been developed for summing rows of partial products using a carry-save representation is the array multiplier.

High-Speed Booth Encoded Parallel Multiplier Design:

Fast multipliers are essential parts of digital signal processing systems. The speed of multiply operation is of great importance in digital signal processing as well as in the general purpose processors today, especially since the media processing took off. In the past multiplication was generally implemented via a sequence of addition, subtraction, and shift operations. Multiplication can be considered as a series of repeated additions. The number to be added is the multiplicand, the number of times that it is added is the multiplier, and the result is the product. Each step of addition generates a partial product. In most computers, the operand usually contains the same number of bits. When the operands are interpreted as integers, the product is generally twice the length of operands in order to preserve the information content. This repeated addition method that is suggested by the arithmetic definition is slow that it is almost always replaced by an algorithm that makes use of positional representation. It is possible to decompose multipliers into two parts. The first part is dedicated to the generation of partial products, and the second one collects and adds them.

The basic multiplication principle is two fold i.e. evaluation of partial products and accumulation of the shifted partial products. It is performed by the successive additions of the columns of the shifted partial product matrix. The 'multiplier' is successfully shifted and gates the appropriate bit of the 'multiplicand'. The delayed, gated instance of the multiplicand must all be in the same column of the shifted partial product matrix. They are then added to form the product bit for the particular form. Multiplication is therefore a multi operand operation. To extend the multiplication to both signed and unsigned.

III. Proposed SPST Booth Multiplier

If an operation to multiply two N –bit numbers and accumulates into a 2N -bit number, addition, subtraction, Sum of Absolute Difference (SAD), and Interpolation is considered. The critical path is determined by the 2-bit accumulation operation. If a pipeline scheme is applied for each step in the standard design of Fig. 1, the delay of the last accumulator must be reduced in order to improve the performance of the MAC. The overall performance of the proposed VMFU is improved by eliminating the accumulator itself by combining it with the CSA function. If the accumulator has been eliminated, the critical path is then determined by the final adder in the multiplier. The basic method to improve the performance of the final adder is to decrease the number of input bits. In order to reduce this number of input bits, the multiple partial products are compressed into a sum and a carry by CSA. The number of bits of sums and carries to be transferred to the final adder is reduced by adding the lower bits of sums and carries in advance within the range in which the overall performance will not be degraded. A 2-bit CLA is used to add the lower bits in the CSA. In addition, to increase the output rate when pipelining is applied, the sums and carrys from the CSA are accumulated instead of the outputs from the final adder in the manner that the sum and carry from the CSA in the previous cycle are inputted to CSA. Due to this feedback of both sum and carry, the number of inputs to CSA increases, compared to the standard design. In order to efficiently solve the increase in the amount of data, a CSA architecture is modified to treat the sign bit.

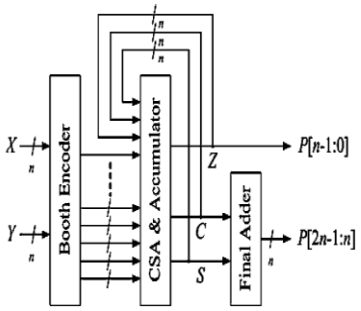

Figure 1 Booth Encoded MAC

A modified Booth's algorithm:

Booth's Algorithm is simple but powerful. Speed of VMFU is dependent on the number of partial products and speed of accumulate partial product. Booth's Algorithm provide us to reduced partial products. We choose radix-4 algorithm because of below reasons.

- Original Booth's algorithm has an inefficient case.

The 17 partial products are generated in 16bit x 16bit signed or unsigned multiplication.

- Modified Booth's radix-4 algorithm has fatal encoding time in 16bit x 16bit multiplication.

Radix-4 Algorithm has a 3x term which means that a partial product cannot be generated by shifting. Therefore, 2x + 1x are needed in encoding processing. One of the solution is handling an additional 1x term in wallace tree. However, large wallace tree has some problems too.

A radix-4 modified Booth's algorithm: Booth's radix-4 algorithm is widely used to reduce the area of multiplier and to increase the speed. Grouping 3 bits of multiplier with overlapping has half partial products which improves the system speed. Radix-4 modified Booth's algorithm is shown below:

- X-1 = 0; Insert 0 on the right side of LSB of multiplier.
- Start grouping each 3bits with overlapping from x-1
- If the number of multiplier bits is odd, add a extra 1 bit on left side of MSB
- generate partial product from truth table
- when new partial product is generated, each partial product is added 2 bit left

shifting in regular sequence.

yi+1	yi	yi-1	partial product
0	0	0	0X (No string)
0	0	1	+ 1X (End of string)
0	1	0	+ 1X (A string)
0	1	1	+ 2X (End of string)
1	0	0	- 2X (Beginning of string)
1	0	1	- 1X (-2X+X)
1	1	0	- 1X (Beginning of string)
1	1	1	- 0X (Center of string)

x: multiplicand y: multiplier

Sign or zero extension

Our MAC supports signed or unsigned multiplication and the produced result is 64bit which are stored in 2 special 32bit register. First MAC receives a multiplicand and multiplier but just 16bit operands are signed number in Booth's radix-4 algorithm. Hence, extension bit is required to express 16bit signed number. The core idea of this is that 16bit unsigned number can be expressed by 33bit signed number. The 17 partial products are generated in 33bit x 33bit case (16 partial products in 32bit x 32bit case). Here is an example of signed and unsigned multiplication. When x(multiplicand) is 3bit 111 and y(multiplier) is 3bit 111, the signed and unsigned multiplication is different. In signed case $x \times y = 1$ (-1 x -1 = 1) and in unsigned case $x \times y = 49$ (7 x 7 = 49).

signed (-1 x -1 = 1)						
x		(1)	1	1	1	
y	×	(1)	1	1	1	(0)
0	0	0	0	0	1	1x
0	0	0	0	0		0x
0	0	0	0	0	1	1_{dec}

unsigned (7 x 7 = 49)						
x		(0)	1	1	1	
y	×	(0)	1	1	1	(0)
1	1	1	0	0	1	-x
1	1	1	0			+2x
1	1	0	0	0	1	49_{dec}

Carry-Save Adder

When three or more operands are to be added simultaneously using two operand adders, the time consuming carry propagation must be repeated several times. If the number of operands is 'k', then carries have to propagate (k-1) times (Weste & Harris, 3rd Ed). In the carry save addition, we let the carry propagate only in the last step, while in all the other steps we generate the partial sum and sequence of carries separately. A CSA is capable of reducing the number of operands to be added from 3 to 2 without any carry propagation. A CSA can be implemented in different ways. In the simplest implementation, the basic element of carry save adder is the combination of two half adders or 1 bit full adder (Weste & Harris, 3rd Ed)

Design Features

One of the most advanced types of MAC for general-purpose digital signal processing has been proposed by Elguibaly . It is an architecture in which accumulation has been combined with the carry save adder (CSA) tree that compresses partial products. In the architecture proposed in, the critical path was reduced by eliminating the adder for accumulation and decreasing the number of input bits in the final adder. While it has a better performance because of the reduced critical path compared to the previous VMFU architectures, there is a need to improve the output rate due to the use of the final adder results for accumulation. The architecture to merge the adder block to the accumulator register in the VMFU operator was proposed to provide the possibility of using two separate N/2-bit adders instead of one-bit adder to accumulate the MAC results. Recently, Zicari proposed an architecture that took a merging technique to fully utilize the 4–2 compressor .It also took this compressor as the basic building blocks for the multiplication circuit.

IV. Algorithm Implementation

In the majority of digital signal processing (DSP) applications the critical operations usually involve many multiplications and/or accumulations. For real-time signal processing, a high speed and high throughput Multiplier-Accumulator (MAC) is always a key to achieve a high performance digital signal processing system and versatile Multimedia functional units. In the last few years, the main consideration of MAC design is to enhance its speed.

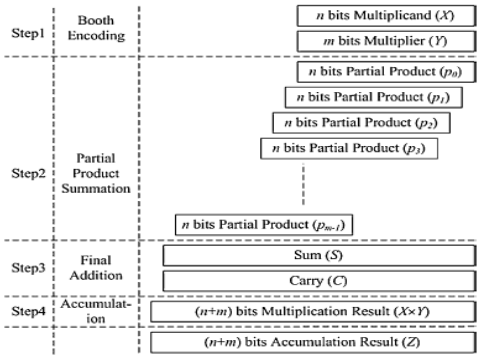

Figure 2 Algorithm for MAC Implementation

This is because; speed and throughput rate is always the concern of block. But for the epoch of personal communication, low power design also becomes another main design consideration. This is because; battery energy available for these portable products limits the power consumption of the system. Therefore, the main motivation of this work is to investigate various Pipelined multiplier/accumulator architectures and circuit design techniques which are suitable for implementing high throughput signal processing algorithms and at the same time achieve low power consumption. A conventional VMFU unit consists of (fast multiplier) multiplier and an accumulator that contains the sum of the previous consecutive products. The function of the VMFU unit is given by the following equation:

$F = \Sigma A i Bi$…… (1)

The main goal of a block design is to enhance the speed of the MAC unit, and at the same time limit the power consumption. In a pipelined MAC circuit, the delay of pipeline stage is the delay of a 1-bit full adder. Estimating this delay will assist in identifying the overall delay of the pipelined MAC. In this work, 1-bit full adder is designed. Area, power and delay are calculated for the full adder, based on which the pipelined MAC unit is designed for low power.

High-Speed Booth Encoded Parallel Multiplier Design:

Fast multipliers are essential parts of digital signal processing systems. The speed of multiply operation is of great importance in digital signal processing as well as in the general purpose processors today, especially since the media processing took off. In the past multiplication was generally implemented via a sequence of addition, subtraction, and shift operations. Multiplication can be considered as a series of repeated additions. The number to be added is the multiplicand, the number of times that it is added is the multiplier, and the result is the product. Each step of addition generates a partial product. In most computers, the operand usually contains the same number of bits. When the operands are interpreted as integers, the product is generally twice the length of operands in order to preserve the information content. This repeated addition method that is suggested by the arithmetic definition is slow that it is almost always replaced by an algorithm that makes use of positional representation. It is possible to decompose multipliers into two parts. The first part is dedicated to the generation of partial products, and the second one collects and adds them.

BLOCK DIAGRAM

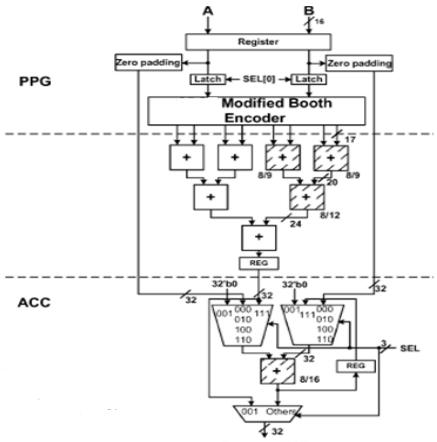

Figure 3 Booth Multiplier

Modified Booth Encoder:

In order to achieve high-speed multiplication, multiplication algorithms using parallel counters, such as the modified Booth algorithm has been proposed, and some multipliers based on the algorithms have been implemented for practical use. This type of multiplier operates much faster than an array multiplier for longer operands because its computation time is proportional to the logarithm of the word length of operands.

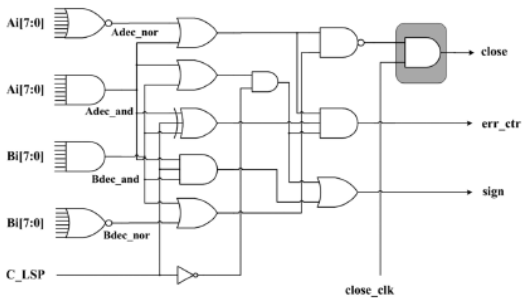

Figure 4 Modified Booth Encoder

Booth multiplication is a technique that allows for smaller, faster multiplication circuits, by recoding the numbers that are multiplied. It is possible to reduce the number of partial products by half, by using the technique of radix-4 Booth recoding. The basic idea is that, instead of shifting and adding for every column of the multiplier term and multiplying by 1 or 0, we only take every second column, and multiply by ±1, ±2, or 0, to obtain the same results. The advantage of this method is the halving of the number of partial products. To

Booth recode the multiplier term, we consider the bits in blocks of three, such that each block overlaps the previous block by one bit. Grouping starts from the LSB, and the first block only uses two bits of the multiplier. Figure 5 shows the grouping of bits from the multiplier term for use in modified booth encoding.

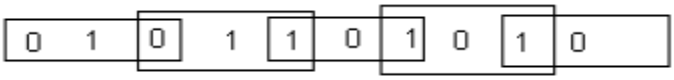

Figure 5 Grouping of bits from the multiplier term

Each block is decoded to generate the correct partial product. The encoding of the multiplier Y, using the modified booth algorithm, generates the following five signed digits, -2, -1, 0, +1, +2. Each encoded digit in the multiplier performs a certain operation on the multiplicand, X, as illustrated in Table 1

Table 1 Operation Encoder

Block	Re - coded digit	Operation on X
000	0	0 X
001	+1	+1 X
010	+1	+1 X
011	+2	+2 X
100	-2	-2 X
101	-1	-1 X
110	-1	-1 X
111	0	0 X

For the partial product generation, we adopt Radix-4 Modified Booth algorithm to reduce the number of partial products for roughly one half. For multiplication of 2's complement numbers, the two-bit encoding using this algorithm scans a triplet of bits. When the multiplier B is divided into groups of two bits, the algorithm is applied to this group of divided bits.

Figure 6, shows a computing example of Booth multiplying two numbers "2AC9" and "006A". The shadow denotes that the numbers in this part of Booth multiplication are all zero so that this part of the computations can be neglected. Saving those computations can significantly reduce the power consumption caused by the transient signals. According to the analysis of the multiplication shown in figure 4, we propose the SPST-equipped modified-Booth encoder, which is controlled by a detection unit. The detection unit has one of the two operands as its input to decide whether the Booth encoder calculates redundant computations. As shown in figure 7.

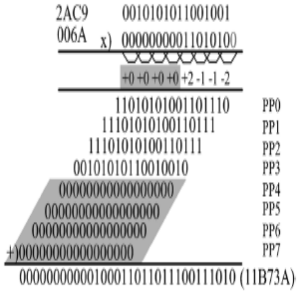

Figure 6 Illustration of multiplication using modified Booth encoding

The latches can, respectively, freeze the inputs of MUX-4 to MUX-7 or only those of MUX-6 to MUX-7 when the PP4 to PP7 or the PP6 to PP7 are zero; to reduce the transition power dissipation. Figure 10, shows the booth partial product generation circuit. It includes AND/OR/EX-OR logic. The PP generator generates five candidates of the partial products, i.e., {-2A,-A, 0, A, 2A}. These are then selected according to the Booth encoding results of the operand B. When the operand besides the Booth encoded one has a small absolute value, there are opportunities to reduce the spurious power dissipated in the compression tree.

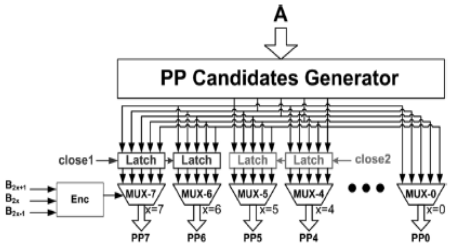

Figure 7 SPST equipped modified Booth encoder

V. Results & Conclusion

This work presents a functional unit which is designed with multiplier-accumulator (MAC), with addition, subtraction, sum of absolute difference. A Modified Booth multiplier circuit is used for MAC architecture. Compared to other circuits, the Booth multiplier has the highest operational speed and less hardware count. The basic building blocks for the unit are identified and each of the blocks is analyzed for its performance.MAC unit is designed with enable to block.Using this block, the MAC unit is constructed and calculated for the MAC unit parameters.

The presented technique explores its applications in multimedia/DSP computations, where the theoretical analysis and the realization issues are fully discussed.In this paper we used Xilinx-ISE tool for logical verification, and further synthesizing it on Xilinx-ISE tool using target technology and performing placing & routing operation for system verification. In future the thesis can be extended to floating point numbers also with the supportive EDA tools. By using transistor level implementation for the carry save logic the design reduces the total area required compared to gate level designs. Total latency required to produce the result is 40.023ns which is less when compare with the Existing MAC units.

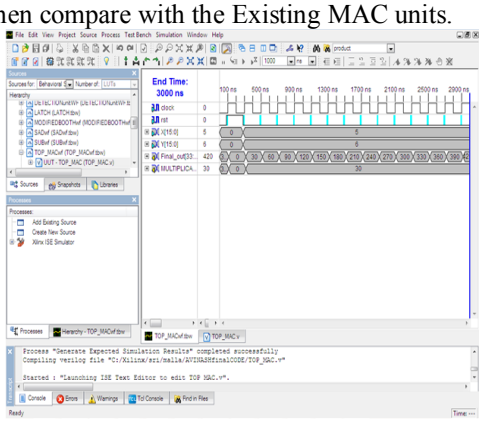

Figure 8 Simulation Results

Table 2 Device Utilization Summary

Device Utilization Summary (estimated values)			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	581	4656	12%
Number of Slice Flip Flops	34	9312	0%
Number of 4 input LUTs	1084	9312	11%
Number of bonded IOBs	100	232	43%
Number of GCLKs	1	24	4%

Acknowledgements

The authors would like to thank the anonymous reviewers for their comments which were very helpful in improving the quality and presentation of this paper.

References

[1] Soojin Kim and Kyeongsoon Cho "Design of High-speed Modified Booth Multipliers Operating at GHz Ranges" World Academy of Science, Engineering and Technology 61 2010.

[2] Magnus Sjalander and Per Larson-Edefors. "The Case for HPM-Based Baugh-Wooley Multipliers," Chalmers University of Technology,Sweden, March 2008.

[3] Z Haung and M D Ercegovac, "High performance Low Power left to right array multiplier design" IEEE rans.Computer, vol 54 no3, page 272-283 Mar 2005.

[4] Aswathy Sudhakar, and D. Gokila, "Run-Time configurable Pipelined Modified Baugh-Wooley Multipliers," Advances in Computational Sciences and Technology ISSN 0973-6107 Volume 3 Number 2 (2010) pp. 223–235.

[5] Myoung-Cheol Shin, Se-Hyeon Kang, and In-Cheol Park, "An Area-Efficient Iterative Modified-Booth Multiplier Based on Self-Timed Clocking," Industry, and Energy through the project System IC 2010, and by IC Design Education Center (IDEC).

[6] Leandro Z. Pieper, Eduardo A. C. da Costa, Sérgio J. M. de Almeida, "Efficient Dedicated Multiplication Blocks for2´s Complement Radix-2*m* Array Multipliers," JOURNAL OF COMPUTERS, VOL. 5, NO.10, OCTOBER 2010.

[7] Wen-Chang Yeh and Chein-Wei Jen, "High-speed Booth encoded parallel multiplier design," IEEE Trans. on Computers, vol. 49, isseu 7,pp. 692-701, July 2000.

[8] Jung-Yup Kang and Jean-Luc Gaudiot, "A simple high-speed multiplier design," IEEE Trans. on Computers, vol. 55, issue 10, Oct. pp.1253-1258, 2006.

Authors Profile:

CHINABABU VANAMA is Pursuing his M. Tech from
Chirala Engineering College, Chirala in the department of Electronics & Communications En-gineering (ECE) with specialization in VLSI & Embedded systems.

M. SUMALATHA is working as an Associate Professor in the department of Electronics & Communication Engin-eering in Chirala Engineering College, Chirala. She completed masters from JNTUK. She has over 7 years of teaching experience.