International Journal of Scientific Engineering and Technology Research

ISSN 2319-8885
Vol.03,Issue.02,
February-2014,
Pages:0239-0244

www.semargroup.org,
www.ijsetr.com

Design and Implementation of High Speed Radix 8 Multiplier using 8:2 Compressors

A.M.SRINIVASA CHARYULU[1], G.SHANMUGA PRIYA[2], E.N.V.PURNA CHANDRA RAO[3]

[1]Research Scholar, Dept of ECE, CMRIT, Hyderabad, Andhrapradesh, India, E-mail: aacharyasrinivas@gmail.com.
[2]Assoc Prof, Dept of ECE, CMRIT, Hyderabad, Andhrapradesh, India, E-mail: spriyagsn@yahoo.com.
[3]HOD, Dept of ECE, CMRIT, Hyderabad, Andhrapradesh, India.

Abstract: This paper presents an area efficient implementation of a high performance parallel multiplier. Radix-4 Booth multiplier with 3:2 compressors and Radix-8 Booth multiplier with 4:2 compressors are presented here. The design for the 8:2 compressors is presented and compared with the 4:2 compressors. The design is structured for m × n multiplication where m and n can reach up to 126 bits. Carry Look ahead Adder is used as the final adder to enhance the speed of operation. Finally the performance improvement of the proposed multipliers is validated by implementing a higher order FIR filter. The design entry is done in VHDL and simulated using Model Sim SE 6.4 design suite from Mentor Graphics. It is then synthesized and implemented using Xilinx ISE 9.2i targeted towards Spartan 3 FPGA.

Keywords: FPGA; HDL; Carry Look ahead Adder; Carry Save Adder; Wallace Tree; Booth Encoding.

I. INTRODUCTION

With the rapid advances in multimedia and communication systems, real-time signal processing and large capacity data processing are increasingly being demanded. The multiplier is an essential element of the digital signal processing such as filtering and convolution. Most digital signal processing methods use nonlinear functions such as discrete cosine transform(DCT) or discrete wavelet transform (DWT). As they are basically accomplished by repetitive application of multiplication and addition, their speed becomes a major factor which determines the performance of the entire calculation. Since the multiplier requires the longest delay among the basic operational blocks in digital system, the critical path is determined more by the multiplier[2]. Furthermore, multiplier consumes much area and dissipates more power. Hence designing multipliers which offer either of the following design targets high speed, low power consumption[3], less area or even a combination of them is of substantial research interest. Multiplication operation involves generation of partial products and their accumulation.

The speed of multiplication can be increased by reducing the number of partial products and/or accelerating the accumulation of partial products. Among the many methods of implementing high speed parallel multipliers, there are two basic approaches namely Booth algorithm and Wallace Tree compressors. This paper describes an efficient implementation of a high speed parallel multiplier using both these approaches. Here two multipliers are proposed. The first multiplier makes use of the Radix-4 Booth Algorithm with 3:2 compressors while the second multiplier uses the Radix-8 Booth algorithm with 4:2 compressors. The design is structured for m x n multiplication where m and n can reach up to 126 bits. The number of partial products is n/2 in Radix-4 Booth algorithm while it gets reduced to n/3 in Radix-8 Booth algorithm. The Wallace tree uses Carry Save Adders (CSA) to accumulate the partial products. This reduces the time as well as the chip area. To further enhance the speed of operation, carry-look-ahead (CLA) adder is used as the final adder [4].

II.MULTIPLER

Multiplication is one of the most Complex Operations within arithmetic processors such as the ALU. Hence it is one of the most complex primitive to be designed in the configurable chip. The selection criteria for various design options. Two Architectures are Configurable serial/parallel Multiplier and Configurable

A. Serial-Parallel Multiplier

Serial multipliers also find applications in system-on-chip(SoC) design. As technology scales, more intellectual property cores and logic blocks will be integrated in a SoC, resulting in larger interconnect area and higher power dissipation. The increase in integration density of the on-chip modules causes the buses connecting these modules to become highly congested. To overcome this problem, new techniques have been evolved recently to have on-chip data transfer in a high speed serial link instead of conventional

bus depict the conventional on-chip bus and alternative on-chip serial-link bus structures, respectively, the serializer at the source module converts the parallel outputs to a bit stream that can be transferred in a simple routing network and at the destination module they are converted back to parallel data by the deserializer.

B. Serial-Serial Multiplier

The on-chip serial-link is capable of transmitting data at Gb/s so that a chunk of parallel data is available when the destination module finishes the previous computation Under the new on-chip communication paradigm for digital signal processing, it is desirable to have a low complexity data processing unit as the destination module that is able to perform partial computation on the incoming data stream at high speed while the data is being buffered illustrates a potential use of a serial-serial multiplier as a destination module in a SoC with serial-link bus architecture. The low complexity pre computation unit forms part of the serial-serial multiplier and could perform partial computation on the high speed serial bit stream.

The unit doubles as a buffer and eliminates the deserializer. As the data has been partially processed and buffered, the completion of the multiplication can be done at a lower speed with a less complex parallel multiplier. The challenge in such a scheme lies in reducing the critical path delay of the pre computation unit to that of the deserializer, which usually has bit rate in the order of several Gb/s. We introduce this new scheme for the design of serial-serial multiplier suitable for SoCs with on-chip serial-link bus architecture. The proposed scheme could also be used as an alternative to embedded multipliers in the future field-programmable gate array (FPGA), where configurable logic blocks (CLBs), embedded multipliers and memory blocks are integrated with serializer / deserializer to facilitate on-chip serial data transfer in order to reduce interconnect complexity.

A serial accumulator developed based on the new design paradigm is proposed to deal with very high-speed data sampling rate of above 4 GHz. The accumulator employs asynchronous counters1 to perform bit accumulation at each bit position of the PP matrix, resulting in low critical path delay and small area, especially for operands with long word length. Asynchronous counter has a low hardware complexity but the outputs are not synchronized with the clock which leads to a timing delay before all output bits of the counter have settled to their final states. The correct output of the counter is read after a timing delay to be analyzed from the timing diagram in Section VI-B. The data dependent counters change states only when the input bit is "1," which leads to low switching power dissipation. The height of the PP matrix after buffering by the asynchronous counters is reduced logarithmically to $[\log_2 n] +1$ before it is further reduced by the CSA tree.

C. Parallel/parallel Multiplier

In serial/parallel multiplier algorithm is one design "serial" components points to reduce silicon chip area. Two unsigned fixed point numbers represented by m, n bits can be

$$a(m) = am-1\ldots\ldots a0$$
$$b(n) = bn-1\ldots\ldots..b0 \quad (1)$$

The double word length product Q (m+n) is

$$Q(m+n) = \sum \sum a^i b^i 2^{i+j} \quad (2)$$

Multipliers play an important role in today's digital signal processing and various other applications. With advances in technology, many researchers have tried and are trying to design multipliers which offer either of the following design targets–high speed, low power consumption, regularity of layout and hence less area or even combination of them in one multiplier thus making them suitable for various high speed, low power and compact VLSI implementation. The common multiplication method is "add and shift "algorithm. In parallel multipliers number of partial products to be added the main parameter that determines the performance of the multiplier. To reduce the number of partial products to be added, Modified Booth algorithm is one of the most popular algorithms. To achieve speed improvements Wallace Tree algorithm can be used to reduce the number of sequential adding stages. Further by combining both modified Booth algorithm and Wallace Tree technique we can see advantage of both algorithms in one multiplier. However with increasing parallelism, the amount of shifts between the partial products and intermediate sums to be added will increase which may result in reduced speed, increase in silicon area due to irregularity of structure and also increased power consumption due to increase in interconnect resulting from complex routing. On the other hand "serial-parallel "multipliers compromise speed to achieve better performance for area and power consumption. The selection of a parallel or serial multiplier actually depends on the nature of application. In this lecture we introduce the multiplication algorithms and architecture and compare them in terms of speed, area, power and combination of these metrics

D. Different multipliers

As we know in multiplication operation there are two operands, one is multiplicand and other is multiplier. In binary number system we do multiplication by using different type of multiplier. A binary multiplier uses the simple shift and adds operation. There are many multipliers introduced in digital electronics. Some of them are

1. Array multiplier

An array multiplier is shown in below Fig.1 is a parallel multiplier which does shift and adds all at once. This multiplier is called an array because it has array of

adders. An array multiplier also uses shift and adds operation as in binary multiplier but it adds the partial products parallel. The following figure shows the 4x4 array multiplier. Digital Multiplication entails a sequence of additions carried out on partial products the method by which this partial product array is summed to give the final product is the key distinguishing factor amongst the numerous multiplication schemes

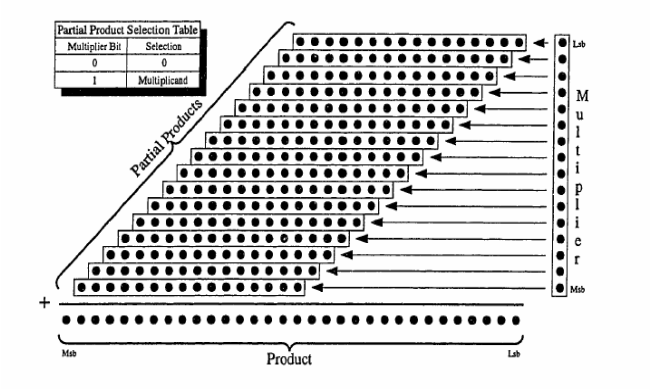

Fig.1. Array Multiplication.

2. Wallace Tree Multiplier

Several popular and well-known schemes, with the objective of improving the speed of the parallel multiplier, have been developed in past. Wallace introduced a very important iterative realization of parallel multiplier. This advantage becomes more pronounced for multipliers of bigger than 16 bits. In Wallace tree architecture, all the bits of all of the partial products in each column are added together by a set of counters in parallel without propagating any carries. Another set of counters then reduces this new matrix and so on, until a two-row matrix is generated. The most common counter used is the 3:2 counters which is a Full Adder. The final results are added using usually carry propagate adder. The advantage of Wallace tree is speed because the addition of partial products is now O (logN). A block diagram of 4 bit Wallace Tree multiplier is shown in below. As seen from the block diagram partial products are added in Wallace tree block. The result of these additions is the final product bits and sum and carry bits which are added in the final fast adder (CRA).

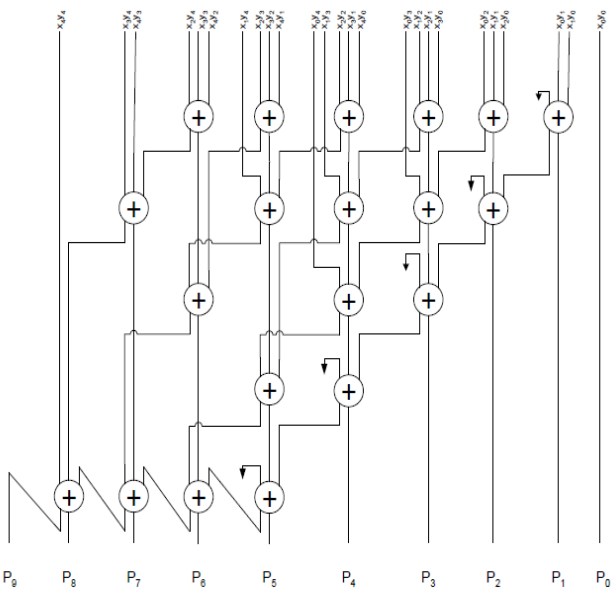

Fig.2. Wallace tree Architecture

Since Wallace Tree is a summation method, it can be used in conjunction with array multiplier of any kind including Booth array. The diagram below shows the implementation of 8 bit squarer using the Wallace tree for compressing the addition process. Above fig.2 is a Wallace tree Architecture

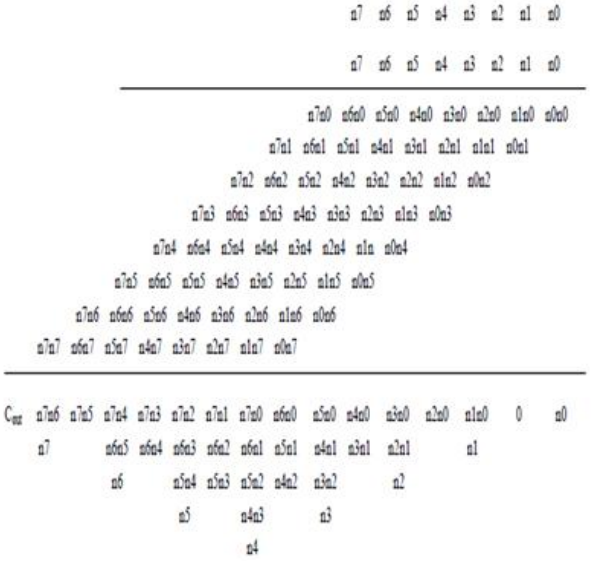

Fig.3. Operation of 8 bit square

Wallace introduced a very important iterative realization of parallel multiplier. This advantage becomes more pronounced for multipliers of bigger than 16 bits. In Wallace tree architecture, all the bits of all of the partial products in each column are added together by a set of counters in parallel without propagating any carries. Hear we see in fig.3. Operation of 8 bit square and in fig.4. Operation of 32 bit Multiplication using Booth and Wallace tree. Another set of counters then reduces this new matrix

and so on, until a two-row matrix is generated. The most common counter used is the 3:2 counters which is a Full Adder. The final results are added using usually carry propagate adder. The advantage of Wallace tree is speed because the addition of partial products is now O (logN). A block diagram of 4 bit Wallace Tree multiplier is shown in below. As seen from the block diagram partial products are added in Wallace tree block fig.2. The result of these additions is the final product bits and sum and carry bits which are added in the final fast adder (CRA).

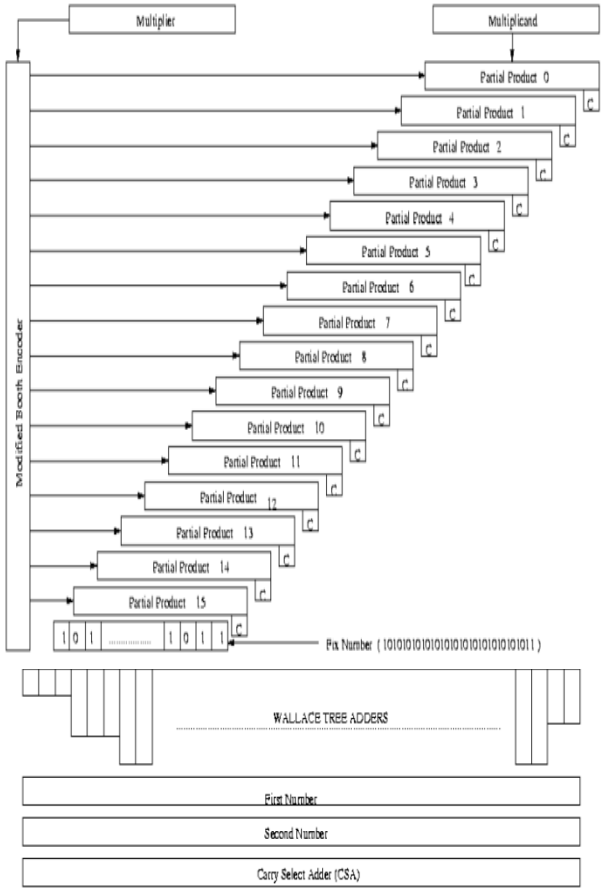

Fig.4. 32 bit Multiplication using Booth and Wallace tree

III.RADIX 2 BOOTH MULTIPLIER

Booth algorithm provides a procedure for multiplying binary integers in signed-2's complement representation. According to the multiplication procedure, strings of 0's in the multiplier require no addition but just shifting and a string of 1's in the multiplier from bit weight 2^k to weight 2^m can be treated as 2^{k+1} - 2^m. Booth algorithm involves recoding the multiplier first. In the recoded format, each bit in the multiplier can take any of the three values: 0, 1 and -1.Suppose we want to multiply a number by 01110 (in decimal 14). This number can be considered as the difference between 10000 (in decimal 16) and 00010 (in decimal 2). The multiplication by 01110 can be achieved by summing up the following products:

- 2^4 times the multiplicand ($2^4 = 16$)
- 2's complement of 2^1 times the multiplicand ($2^1 = 2$).

In a standard multiplication, three additions are required due to the string of three 1's.This can be replaced by one addition and one subtraction. The above requirement is identified by recoding of the multiplier 01110 using the following rules summarized in table 1.

Table 1: Radix 2 recoding rules

Q_n	Q_{n+1}	Recoded bits	Operation performed
0	0	0	Shift
0	1	+1	Add M
1	0	-1	Subtract M
1	1	0	Shift

To generate recoded multiplier for radix-2, following steps are to be performed:

- Append the given multiplier with a zero to the LSB side.
- Make group of two bits in the overlapped way
- Recode the number using the above table.

Consider an example which has the 8 bit multiplicand as 11011001 and multiplier as 011100010.

```
Multiplicand                1 1 0 1 1 0 0 1
Multiplier                  0 1 1 1 0 0 0 1 0
                            ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓
Recoded multiplier          +1 0  0 -1 0 0+1-1
                              0 0 0 1 0 0 1 1 1
                            1 1 1 0 1 1 0 0 1
                          0 0 0 0 0 0 0 0 0
                        0 0 0 0 0 0 0 0 0
                      0 0 0 1 0 0 1 1 1
                    0 0 0 0 0 0 0 0 0
                  0 0 0 0 0 0 0 0 0
                1 1 1 0 1 1 0 0 1
Product     0000001001001001
```

A. Modified Booth Algorithm for Radix 4

One of the solutions of realizing high speed multipliers is to enhance parallelism which helps to decrease the number of subsequent calculation stages. The original version of the Booth algorithm (Radix-2) had two drawbacks. They are:

1. The number of add subtract operations and the number of shift operations becomes variable and becomes inconvenient in designing parallel multipliers.
2. The algorithm becomes inefficient when there are isolated 1's. These problems are overcome by using modified Radix 4.

Booth algorithm which scans strings of three bits is given below:

1. Extend the sign bit 1 position if necessary to ensure that n is even.
2. Append a 0 to the right of the LSB of the multiplier.
3. According to the value of each vector, each Partial Product will be 0, +M,-M, +2M or -2M.

The negative values of B are made by taking the 2's complement and in this paper Carry-look-ahead (CLA) fast adders are used. The multiplication of M is done by shifting M by one bit to the left. Thus, in any case, in designing n-bit parallel multiplier, only n/2 partial products are produced.

The partial products are calculated according to the following rule

$$Z_n = -2 \times B_{n+1} + B_n + B_{n-1} \quad (3)$$

Where B is the multiplier.

Table2. Modified Radix 4 recoding rules

Consider example for radix 4:

Multiplicand 1 0 0 0 0 0 0 1

Multiplier 0 1 1 1 1 1 1 0 0

+2 0 0 -2

0000000011111110

00000000000000

000000000000

1100000010

Product 1100000101111110

Table3. Comparison of normal and modified multiplier

Device parameter	Usage of 8:2 compressor	Usage in % 8:2 compressor	Usage of 8:2 compressor	Usage in % 8:2 compressor	Total available
Number of Slices	2121	45	2181	46	4656
Number of 4 input LUTs	3943	42	4044	43	9312
Number of IOs	128		128		
Number of bonded IOBs	128	55	128	55	232

Device utilization summery for the device 3s500efg320-4

Total delay for modified: 92.458ns (54.205ns logic, 38.253ns route) (58.6% logic, 41.4% route)

Total delay for normal: 115.924ns (64.207ns logic, 51.717ns route) (55.4% logic, 44.6% route)

Table4. In radix-8 recoding insert this table

Multiplier Bits Y_{i+2}	Y_{i+1}	Y_i	Y_{i-1}	Recoded Operation on multiplicand, X
0	0	0	0	0X
0	0	0	1	+X
0	0	1	0	+X
0	0	1	1	+2X
0	1	0	0	+2X
0	1	0	1	+3X
0	1	1	0	+3X
0	1	1	1	+4X
1	0	0	0	-4X
1	0	0	1	-3X
1	0	1	0	-3X
1	0	1	1	-2X
1	1	0	0	-2X
1	1	0	1	-1X
1	1	1	0	-1X
1	1	1	1	0X

IV. SIMULATION RESULTS

For radix 8 with 8:2 compressors simulation results as shown in fig.5,

Multiplicand =00"&x"0000000ABDC45600000000000000569"

Multiplier="00"&x"0000ABCD78000000000000000069544"

Product=0000000000073561249EE650000000003E83085EA34D8000000000239D8CE4

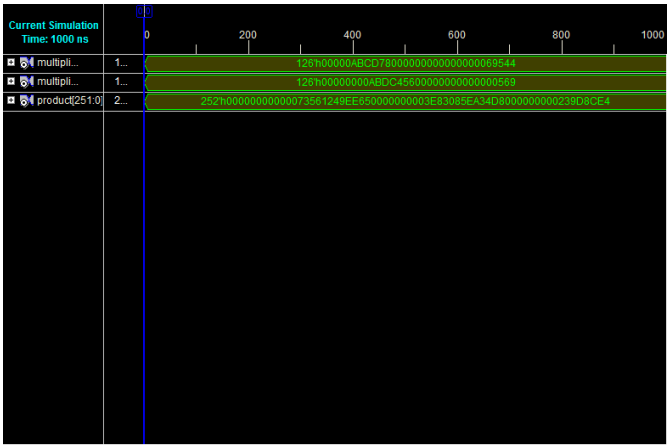

Fig.5. Radix_8with 8:2 compressors simulation result

V. CONCLUSION

In this paper, the design and implementation of two high performance parallel multipliers is proposed. The first multiplier makes use of the Radix-4 Booth Algorithm with 3:2 compressors while the second multiplier uses the Radix-8 Booth algorithm with 4:2 compressors. Both the designs were implemented on Spartan 3 FPGA. The multiplier using Radix- 4 Booth algorithm with 3:2 compressors shows more reduction in device utilization as compared to the multiplier

using Radix-8 Booth algorithm with 4:2 compressors. Meanwhile the multiplier using Radix-8 Booth algorithm with 8:2 compressors are found to be faster than the other. Also the use of Radix- 8 Booth multiplier with 8:2 compressors for a higher order FIR filter showed a dramatic speed improvement than that using Radix-4 Booth multiplier with 4:2 compressors.

VI. REFERENCES

[1] Aparna P R, Nisha Thomas, "Design and Implementation of a High Performance Multiplier using HDL", IEEE Transactions, vol.20, pp.: 401-408, 08 Feb. 2009.

[2] Dong-Wook Kim, Young-Ho Seo, "A New VLSI Architecture of Parallel Multiplier-Accumulator based on Radix-2 Modified Booth Algorithm", Very Large Scale Integration (VLSI) Systems, IEEE Transactions, vol.18, pp.: 201-208, 04 Feb. 2010.

[3] Prasanna Raj P, Rao, Ravi, "VLSI Design and Analysis of Multipliers for Low Power", Intelligent Information Hiding and Multimedia Signal Processing, Fifth International Conference, pp.: 1354-1357, Sept. 2009.

[4] Lakshmanan, Masuri Othman and Mohamad Alauddin Mohd.Ali, "High Performance Parallel Multiplier using Wallace-Booth Algorithm", Semiconductor Electronics, IEEE International Conference , pp.: 433- 436, Dec. 2002.

[5] Jan M Rabaey, "Digital Integrated Circuits, A Design Perspective", Prentice Hall, Dec.1995.

[6] Louis P. Rubin field, "A Proof of the Modified Booth's Algorithm for Multiplication", Computers, IEEE Transactions, vol.24, pp.: 1014-1015, Oct. 1975.

[7] Rajendra Katti, "A Modified Booth Algorithm for High Radix Fixed point Multiplication", Very Large Scale Integration (VLSI) Systems, IEEE Transactions, vol. 2, pp.: 522-524, Dec. 1994.

[8] C. S. Wallace, "A Suggestion for a Fast Multiplier", Electronic Computers, IEEE Transactions, vol.13, Page(s): 14-17, Feb. 1964.

[9] Hussin R et al, "An Efficient Modified Booth Multiplier Architecture", IEEE International Conference, pp.:1-4, 2008.