ISSN (Online): 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Visit: www.ijirset.com

Vol. 7, Issue 1, January 2018

Analysis of Area Performance of Generic Multipliers

Anju Rajput

Assistant Professor, Department of Electronics & Communication Engineering, Arya Institute of Engineering & Technology, Kukas, Jaipur Rajasthan, India

ABSTRACT: Multipliers are the basic element in the Microprocessors and DSPs. Multipliers are the major source of power dissipation. Power consumption of multipliers can be reduce at various levels of design hierarchy.Using different algorithm, power consumption can be reduced. The aim of this paper is to present analysis of generic multipliers on the basis of area and time performance. The generic architecture of all the four multipliers i.e. array, column bypass, wallace tree and booth multiplier has been analysed. Analysis of these multipliers tells that Wallace tree occupy more area but also faster than the other. The simulation and synthesis have been carried out on ISE design suite-14.9. By using this analysis different multipliers can be used for different applications.

KEYWORDS: Array Multiplier,Booth Multiplier, Column Bypass Multiplier, Wallace tree Multiplier

I. INTRODUCTION

For the fabrication of DSP system and high performance systems, low power consumption and small area are most important design criteria. For any multiplier optimizing the speed and area of multiplier is an important design issue. Multipliers are the basic element in the Microprocessors and DSPs. Multipliers are the major source of power dissipation. Power consumption of multipliers can be reduce at various levels of design hierarchy.Using different algorithm, power consumption can be reduced. Multiplication is a process of adding an integer to itself a specified number of times. Multiplicand is added to itself a number of times as specified by Multiplier to form the result that we called product. . Area and speed are important design criteria. By analysing these constraints multiplier which suits best can be used.

The purpose of this paper is to present analysis of four different multipliers i.e. array, column bypass, wallace tree and booth multipliers, on the basis of area and timing performance. Analysis of these multipliers tells that Wallace tree occupy more area but also faster than the other. The growing market for fast floating-point co-processors, digital signal processing chips, and graphics processors has created a demand for high-speed, area-efficient multipliers. Computational performance of a DSP system is limited by its multiplication performance[10]. Multipliers are the main power eating elements of DSP and communication systems. Therefore high speed & low power multiplier is much desired[7].The three important considerations for VLSI design are Power, area and delay. There are many proposed logics of low power dissipation and high speed, each logic style has its own advantages in terms of speed and power.

II. MULTIPLIERS

a. Array Multiplier: It is based on shift and add algorithm. It is used for small circuits. It requires less hardware but takes more time. Each digit of multiplier is multiplied with the multiplicand. We write the result which is called the partial products. Then we take the second digit of the multiplier and multiply with the same multiplicand, write it down from the first partial product, but we do the shifting. We will continue to do that until we exhaust all the digit of the multiplier. Finally we add the whole thing, to get the fellow product. This algorithm is known as shift /add algorithm, because we have to add it after shifting.

ISSN (Online): 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Visit: www.ijirset.com

Vol. 7, Issue 1, January 2018

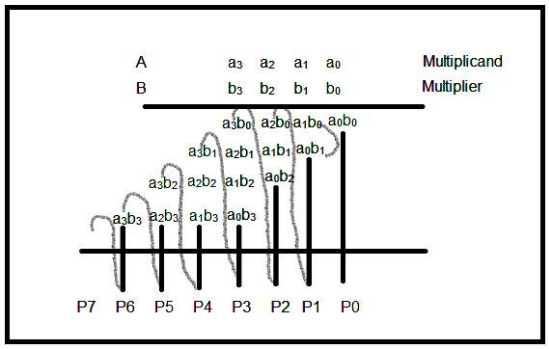

Fig.1 Basic idea of Array Multiplier [1]

All the partial products are added to get the fellow product, but before adding we have to do shifting of the partial product to one bit position in left as shown in **fig.1**[1]. The idea of basic array multiplier is shown below which will give the partial products on each multiplication and results will be shown as P_0 P_1 P_2 P_3 P_4 P_5 P_6& P_7.

b. Column Bypass Multiplier: Column Bypassing in case of multiplier means turning off some columns in the multiplier array in case when certain multiplicand bits are zero. In this technique, during working, the operations in a column can be disabled if the corresponding bit in the multiplicand is 0, to save the power. The modified cell of adder circuit is shown in fig.2 below.

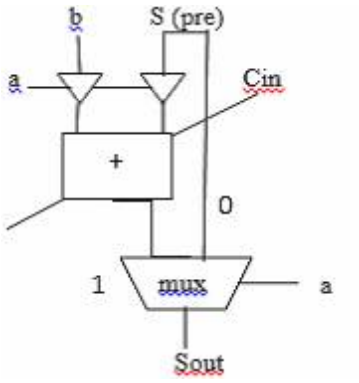

Fig.2 Modified cell of Adder circuit

In **fig.2**, there are two tristate buffers, a and b are the inputs, S(pre) is the previous sum and Cin is the previous carry. When a=0, multiplexers' select line is 0, then b will not get propagated and S (pre) will be the output. When a=1, select line is 1 then a, b and S(pre) will get added by the adder and Sout is the output[2].

c. Booth Multiplier: To enhance the addition among the partial products, requirement of fast adder architectures are required. The Modified- Booth algorithm is mostly used for high speed multiplier circuits. By decreasing the number of generated partial products, we can improve the multiplier performance. For the signednumbers multiplication booth is a powerful algorithm.[8].

ISSN (Online): 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Visit: www.ijirset.com

Vol. 7, Issue 1, January 2018

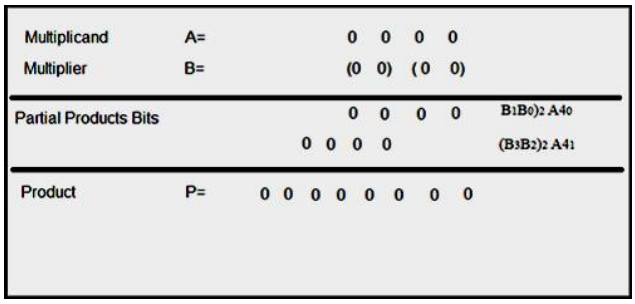

Fig.3 Booth Multiplication

It treats both signed and unsigned numbers. Booth algorithm is a technique which will reduce number of multiplicand multiples. The booth multiplication is shown in **fig 3**[2] in which A_x is multiplicand and B_x is multiplier, P_x is the product of booth multiplication.

d.Wallace Tree Multiplier: For fast multiplication of two numbers, Wallace tree multiplier is used. It is faster than simple array multiplier[11]. It has logarithmic height. But Wallace tree has irregular wiring. Due to this reason, it is often avoided by designer .Wallace tree use log-depth tree network for reduction [6].

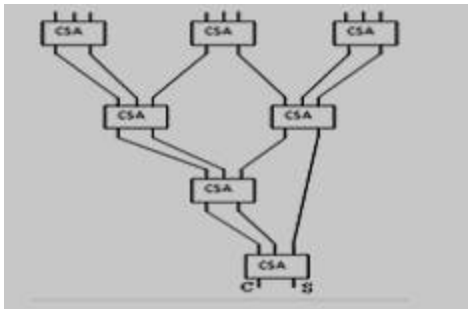

Fig.4 Wallace Tree

This multiplier is not used for low power applications, because excess wiring consumes extra power. But it is faster as it uses carry save adder. It is high speed multiplier. Basic idea of Wallace tree is shown below in **fig.4** [7] in which adders are shown which will help in implementation of Wallace tree.

III.AREA REPORT OF DIFFERENT MULTIPLIERS

1. Area Report of 4-bit Array Multiplier:

Release 6.1i Map G.23
Xilinx Mapping Report File for Design 'array_multiplier'
Command Line : C:/Xilinx/bin/nt/map.exe -intstyle ise -p xc3s400-pq208-4 -cm
area -pr b -k 4 -c 100 -tx off -o array_multiplier_map.ncd array_multiplier.ngd
array_multiplier.pcf
Target Device : x3s400
Target Package : pq208
Target Speed : -4

ISSN (Online): 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Visit: www.ijirset.com

Vol. 7, Issue 1, January 2018

Mapper Version : spartan3 -- $Revision: 1.16 $
Mapped Date : Tue May 06 10:35:23 2014
Design Summary
Number of 4 input LUTs: 33 out of 7,168 1%
Number of occupied Slices: 18 out of 3,584 1%
Number of Slices containing only related logic: 18 out of 18 100%
Number of Slices containing unrelated logic: 0 out of 18 0%
Total Number of 4 input LUTs: 33 out of 7,168 1%
Number of bonded IOBs: 16 out of 141 11%
Total equivalent gate count for design: 198
Additional JTAG gate count for IOBs: 768
Peak Memory Usage: 103 MB

2. Area Report of 16-bit Array Multiplier:

Release 6.1i Map G.23
Xilinx Mapping Report File for Design 'array_multiplier'
Command Line : C:/Xilinx/bin/nt/map.exe -intstyle ise -p xc3s400-pq208-4 -cm area -pr b -k 4 -c 100 -tx off -o array_multiplier_map.ncd array_multiplier.ngd array_multiplier.pcf
Target Device : x3s400
Target Package : pq208
Target Speed : -4
Mapper Version : spartan3 -- $Revision: 1.16 $
Mapped Date : Tue May 06 10:45:00 2014
Design Summary
Number of 4 input LUTs: 525 out of 7,168 7%
Number of occupied Slices: 268 out of 3,584 7%
Number of Slices containing only related logic: 268 out of 268 100%
Number of Slices containing unrelated logic: 0 out of 268 0%
Total Number of 4 input LUTs: 525 out of 7,168 7%
Number of bonded IOBs: 64 out of 141 45%
Total equivalent gate count for design: 3,150
Additional JTAG gate count for IOBs: 3,072
Peak Memory Usage: 105 MB

3. Area Report of 4-bit Booth Multiplier

Release 6.1i Map G.23
Xilinx Mapping Report File for Design 'booth_multiplier'
Command Line : C:/Xilinx/bin/nt/map.exe -intstyle ise -p xc3s400-pq208-4 -cm area -pr b -k 4 -c 100 -tx off -o booth_multiplier_map.ncd booth_multiplier.ngd
Target Device : x3s400
Target Package : pq208
Target Speed : -4
Mapper Version : spartan3 -- $Revision: 1.16 $
Mapped Date : Tue May 06 10:52:43 2014
Design Summary
Number of Slice Latches: 12 out of 7,168 1%

ISSN (Online): 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Visit: www.ijirset.com

Vol. 7, Issue 1, January 2018

Number of 4 input LUTs: 38 out of 7,168 1%
Number of occupied Slices: 23 out of 3,584 1%
Number of Slices containing only related logic: 23 out of 23 100%
Number of Slices containing unrelated logic: 0 out of 23 0%
Total Number 4 input LUTs: 39 out of 7,168 1%
Number used as logic: 38
Number of bonded IOBs: 17 out of 141 12%
Total equivalent gate count for design: 352
Additional JTAG gate count for IOBs: 816
Peak Memory Usage: 104 MB

4. Area Report of 16-bit Booth Multiplier:

Release 6.1i Map G.23
Xilinx Mapping Report File for Design 'booth_multiplier'
Command Line : C:/Xilinx/bin/nt/map.exe -intstyle ise -p xc3s400-pq208-4 -cm area -pr b -k 4 -c 100 -tx off -o booth_multiplier_map.ncd booth_multiplier.ngd booth_multiplier.pcf
Target Device : x3s400
Target Package : pq208
Target Speed : -4
Mapper Version : spartan3 -- $Revision: 1.16 $
Mapped Date : Tue May 06 10:56:00 2014
Design Summary
Number of Slice Latches: 150 out of 7,168 2%
Number of 4 input LUTs: 614 out of 7,168 8%
Number of occupied Slices: 321 out of 3,584 8%
Number of Slices containing only related logic: 321 out of 321 100%
Number of Slices containing unrelated logic: 0 out of 321 0%
Total Number 4 input LUTs: 623 out of 7,168 8%
Number used as logic: 614
Number of bonded IOBs: 65 out of 141 46%
IOB Latches: 2
Total equivalent gate count for design: 5,563
Additional JTAG gate count for IOBs: 3,120
Peak Memory Usage: 106 MB

5. Area Report of 4-bit Column Bypass Multiplier:

Release 6.1i Map G.23
Xilinx Mapping Report File for Design 'coloum_bypass'
Command Line : C:/Xilinx/bin/nt/map.exe -intstyle ise -p xc3s400-pq208-4 -cm area -pr b -k 4 -c 100 -tx off -o coloum_bypass_map.ncd coloum_bypass.ngd coloum_bypass.pcf
Target Device : x3s400
Target Package : pq208
Target Speed : -4
Mapper Version: spartan3 -- $Revision: 1.16 $
Mapped Date : Tue May 06 10:57:59 2014

ISSN (Online): 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Visit: www.ijirset.com

Vol. 7, Issue 1, January 2018

Number of 4 input LUTs: 24 out of 7,168 1%
Number of occupied Slices: 13 out of 3,584 1%
Number of Slices containing only related logic: 13 out of 13 100%
Number of Slices containing unrelated logic: 0 out of 13 0%
Total Number of 4 input LUTs: 24 out of 7,168 1%
Number of bonded IOBs: 16 out of 141 11%
Total equivalent gate count for design: 144
Additional JTAG gate count for IOBs: 768
Peak Memory Usage: 102 MB

6. Area Report of 16-bit Column Bypass Multiplier:

Release 6.1i Map G.23
Xilinx Mapping Report File for Design 'coloum_bypass'
Command Line : C:/Xilinx/bin/nt/map.exe -intstyle ise -p xc3s400-pq208-4 -cm area -pr b -k 4 -c 100 -tx off -o coloum_bypass_map.ncd coloum_bypass.ngd coloum_bypass.pcf
Target Device: x3s400
Target Package: pq208
Target Speed : -4
Mapper Version: spartan3 -- $Revision: 1.16 $
Mapped Date : Tue May 06 11:01:55 2014
Number of 4 input LUTs: 480 out of 7,168 6%
Number of occupied Slices: 241 out of 3,584 6%
Number of Slices containing only related logic: 241 out of 241 100%
Number of Slices containing unrelated logic: 0 out of 241 0%
Total Number of 4 input LUTs: 480 out of 7,168 6%
Number of bonded IOBs: 64 out of 141 45%
Total equivalent gate count for design: 2,880
Additional JTAG gate count for IOBs: 3,072
Peak Memory Usage: 105 MB

7. Area Report of 4-bit Wallace Tree Multiplier:

Release 6.1i Map G.23
Xilinx Mapping Report File for Design 'wallace_mult'
Command Line : C:/Xilinx/bin/nt/map.exe -intstyle ise -p xc3s400-pq208-4 -cm area -pr b -k 4 -c 100 -tx off -o wallace_mult_map.ncd wallace_mult.ngd wallace_mult.pcf
Target Device : x3s400
Target Package : pq208
Target Speed : -4
Mapper Version : spartan3 -- $Revision: 1.16 $
Mapped Date : Thu May 08 09:58:41 2014
Number of Slice Flip Flops: 44 out of 7,168 1%
Number of 4 input LUTs: 46 out of 7,168 1%
Number of occupied Slices: 32 out of 3,584 1%
Number of Slices containing only related logic: 32 out of 32 100%
Number of Slices containing unrelated logic: 0 out of 32 0%

ISSN (Online): 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Visit: www.ijirset.com

Vol. 7, Issue 1, January 2018

Total Number of 4 input LUTs: 46 out of 7,168 1%
Number of bonded IOBs: 17 out of 141 12%
IOB Flip Flops: 8
Number of GCLKs: 1 out of 8 12%
Total equivalent gate count for design: 695
Additional JTAG gate count for IOBs: 816
Peak Memory Usage: 104 MB

8. Area Report of 16-bit Wallace Tree Multiplier:

Release 6.1i Map G.23
Xilinx Mapping Report File for Design 'wallace_mult'
Command Line : C:/Xilinx/bin/nt/map.exe -intstyle ise -p xc3s400-pq208-4 -cm area -pr b -k 4 -c 100 -tx off -o wallace_mult_map.ncd wallace_mult.ngd wallace_mult.pcf
Target Device: x3s400
Target Package: pq208
Target Speed : -4
Mapper Version: spartan3 -- $Revision: 1.16 $
Mapped Date : Thu May 08 10:01:09 2014
Number of Slice Flip Flops: 431 out of 7,168 13%
Number of 4 input LUTs: 438 out of 7,168 13%
Number of occupied Slices: 240 out of 3,584 13%
Number of Slices containing only related logic: 240 out of 140 100%
Number of Slices containing unrelated logic: 0 out of 140 0%
Total Number of 4 input LUTs: 438 out of 7,168 13%
Number of bonded IOBs: 65 out of 141 33%
IOB Flip Flops: 32
Number of GCLKs: 1 out of 8 22%
Total equivalent gate count for design: 6,217
Additional JTAG gate count for IOBs: 2,584
Peak Memory Usage: 105 MB

IV. AREA ANALYSIS

1. Area Analysis of 4-bit multipliers

Table 1.1 Area Analysis of 4-bit multiplier

Multiplier	LUTs	Slices	Gate count
Array	33	18	198
Booth	39	23	352
Column	24	13	144
Wallace tree	46	32	695

The Table 1.1 shows the gate count of 4-bit multipliers, No. of look up tables and number of slices occupied by each multipliers[11].

ISSN (Online): 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Visit: www.ijirset.com

Vol. 7, Issue 1, January 2018

2. Area Analysis of 16-bit multipliers

Table 2.1 Area Analysis of 4-bit multiplier

Multiplier	LUTs	Slices	Gate count
Array	525	268	3150
Booth	623	321	5563
Column	480	241	2880
Wallace tree	438	240	6217

The Table 2.1 shows the number of gate count in four different 16- bit multipliers, no. of look up tables and total no. of slices occupied by 16-bit multipliers with the help of which it is possible to determine larger area among these multipliers.

V. CONCLUSION

From the Table 1.1 and Table 2.1, it can be easily depicted that Wallace tree multiplier occupies more area than the other three multipliers. **Wallace Tree > Booth > Array > Column Bypass**

ACKNOWLEDGEMENT

I would like to acknowledge my mentor Gaurav Jindal Sir who supported me during the period in calculating my results and verifying codes.

REFERENCES

[1]N.Ravi, Dr.T.S.Rao, Dr.T.J.Prasad. “Performance Evaluation of Bypassing Array Multiplier with Optimized Design”, International Journal Of Computer Applications (0975-8887), Vol28 No.5, pp. 3 (2011) .
[2] Mahzad Azarmehr., “Multipliers, Algorithm And Hardware Designs”, Research Center for Integrated Microsystem, 10 (2008).
[3] Nishat Bano, “VLSI Design of low power booth multiplier”, International Journal of Scientific and Engineering Research Vol 3 issue 2, pp. 1-2(feb-2012) .
[4] C.Krishnamacharya ,CH. Sravanthi, K. Avinash, K.V. Uma Maheswaar Rao.,”Design of low power 2-D Multiplier using Bypassing Technique”,International Journal of Innovative Research and Studies ISSN-2319-9725, pp.198 (May 2013).
[5] Partha Sarathi Mohanty, “Design and Implementation of low power multipliers” WILLOW project, 16, pp.26 (2009).
[6] Sumit Vaidya and Deepak Dandekar,“Delay-Power Performance Comparison of Multiplier in VLSI Circuit Design”, International Journal of Computer Networks and Communication Vol2 No.4, pp.49 (July2010).
[7] Kuan –Hung Chen and Yuan-SunChu “A low Power Multiplier with the spst” IEEE , VLSI Vol 15 No.7 July 2007.
[8] H. Lee (A power aware scalable Pipe lines booth multiplier) in proc IEEE int. SOC Conference 2004 PP. 123-12.
[9] Jorn Stohmann Erich Barke, “A Universal Pezaris ArrayMultiplier Generator for SRAM-Based FPGAs” IMS- Institute of MicroelectronicsSystem, University of Hanover Callinstr, 34,D30167 Hanover,Germany.
[10] Morris Mano, “Computer System Architecture”,PP. 346-347, 3rd edition,PHI. 1993.
[11] Pravinkumar Parate, “ASIC Implementation of 4 Bit Multipliers”,,IEEE Computer society. ICETET,2008.25.
[12] 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.