Design, Simulation, Synthesis and Implementation of Wallace Tree Multiplier

By

Taye Girma

Lecturer, SMUC

Abstract

This paper deals with design, synthesis, simulation and implementation of 8x8 Wallace Tree Multiplier. Multipliers form the heart of any DSP operation and determine the performance of general-purpose microprocessors. Wallace tree is an efficient hardware implementation of a circuit that multiplies two integers. It consists of three stages. In the first stage, the partial product matrix is formed or generated. This is to mean multiplying (ANDing) each bit of one of the arguments called multiplier, by each bit of the other arguments called multiplicand. Depending on position of the multiplied bits, the wires carry different weights. Reduce the number of partial products to two by layers of full and half adders. Group the wires in two numbers, and add them with a conventional adder. In the second stage, this partial product matrix is reduced to a height of two through taking any three wires with the same weights and input them into a full adder. In the final stage, these two rows are combined using a carry look ahead adder. Here, if there are two wires of the same weight left, input them into a half adder or if there is just one wire left, connect it to the next layer. The work results in reduction of number of gates that would be used in the design which in turn results in reduction of cost and delay.

Keywords: *Wallace tree multiplier, carry lookahead adders, and multiplier delay*

1. Introduction

Digital circuit design has evolved rapidly over the last 25 years. The earliest digital circuits were designed with vacuum tubes and transistors. Integrated circuits were then invented where logic gates were placed on a single chip. The first Integrated circuit (IC) chips were SSI (small scale Integration) chips where the gate count was very small. As technologies became sophisticated, designers were able to place circuits with hundreds of gates on a chip. These chips were called MSI (Medium Scale Integration) chips. With the advent of LSI (Large Scale Integration), designers could put thousands of gates on a single chip. At that point, design processes started getting very complicated, and designers felt the need to automate these processes. Electronic Design Automation (EDA) techniques began to evolve. Chip designers began to use circuit and logic simulation techniques to verify the functionality of building blocks of the order of about 100 transistors. The circuits were still tested on the breadboard, and the layout was done on

paper or by hand on a graphic computer terminal. With the advent of VLSI (Very Large Scale Integration) technology, designers could design single chips with more than 100,000 transistors. Because of the complexity of these circuits, it was not possible to verify these circuits on a breadboard. Computer aided techniques become critical for verification and design of VLSI digital circuits. Computer programs to do automatic placement and routing of circuit layouts also became popular. The designers were now building gate-level digital circuits manually on graphics terminals. They would build small building blocks and derive higher level blocks from them. This process would continue until they had built the top-level block. Logic simulators came into existence to verify the functionality of these circuits before they were fabricated on chip.

As design requirement become larger and more complex, logic simulation assumed an important role in the design process. Designers could iron out functional bugs in the architecture before the chip was designed further.

2. Review of Related Literature

There are a number of fast multipliers which have already been developed and implemented. Some of the fast multipliers are array multiplier, Dadda multiplier and Wallace tree multiplier.

Array multiplier: Checking the bits of the multiplier one at a time and forming partial products is a sequential operation that requires a sequence of add and shift micro operations. The multiplication of two binary numbers can be done with one micro-operation by means of a combinational circuit that forms the product bits all at once. This is a fast way of multiplying two numbers since all it takes is the time for the signals to propagate through the gates that forms the multiplication array. However, an array multiplier requires a large number of gates, and for this reasons it was not commercial. [7]

To see how an array multiplier can be implemented with a combinational circuit, consider the multiplication of two 2-bit numbers as shown in figure 2.1. The multiplicand bits are b_1 and b_0, the multiplier bits are a_0 and a_1, and the product is $c_3c_2c_1c_0$. The first partial product is formed by multiplying a0 by b_1b_0. The multiplication of two bits such as a_0 and b_0 produces **1** if both bits are **1**; otherwise, it produces a **0**. This is identical to an AND operation and can be implemented with an AND gate. As shown in the diagram, the first partial product is formed by means of two AND gates. The second partial product is formed by multiplying al by b_1b_0 and is shifted one position to the left. The two partial products are added with two half-adder (HA) circuits. Usually, there are more bits in the partial products and it will be necessary to use full-adders to produce the sum. Note that the least significant bit of the product does not have to go through an adder since it is formed by the output of the first AND gate.

A combinational circuit binary multiplier with more bits can be constructed in a similar fashion. A bit of the multiplier is ANDed with each bit of the multiplicand in as many levels as there are bits in the multiplier. The binary output in each level of AND gates are added in parallel with the partial product of the previous level to form a new partial product. The last level produces the product. For j multiplier bits and k multiplicand bits, we need j x k AND gates and (j – 1) k-bit adders to produce a product of j + k bits. [7]

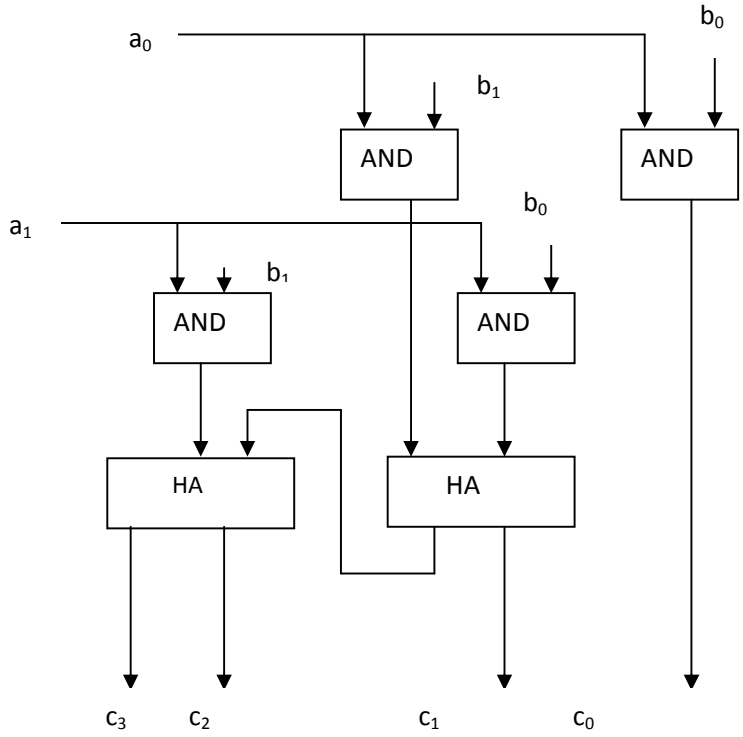

Figure 2.1: 2-bit by 2-bit array multiplier [7]

As a second example, consider a multiplier circuit that multiplies a four bit binary number by four bit binary number.

A_3 A_2 A_1 A_0

B_3 B_2 B_1 B_0

S_{30} S_{20} S_{10} S_{00}

S_{31} S_{21} S_{11} S_{01}

S_{32} S_{22} S_{12} S_{02}

S_{33} S_{23} S_{13} S_{03}

P_7 P_6 P_5 P_4 P_3 P_2 P_1 P_0

Having the above partial products now the design for 4 bit * 4 bit array multiplier will look like the following:

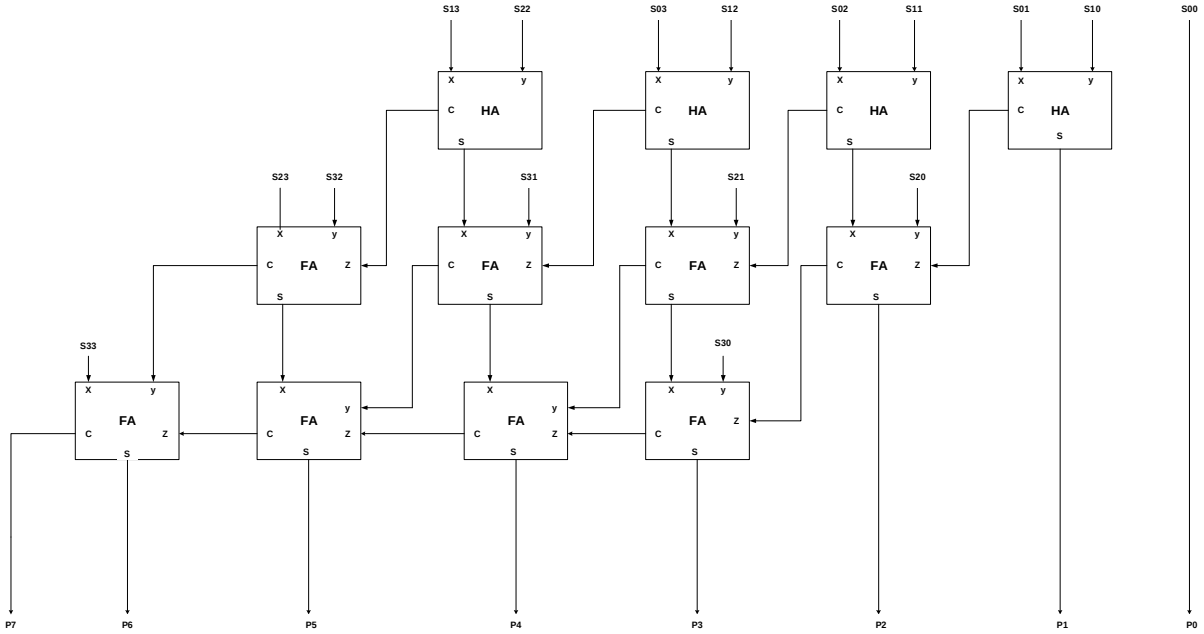

Figure 2.2: 4-bits by 4-bits array multiplier

But Wallace tree and Dadda multipliers are the two well-known fast multipliers [6]. Both consist of three stages. In the first stage, the partial product matrix is formed. In the second stage, this partial product matrix is reduced to a height of two. In the final stage, these two rows are combined. In the Wallace method, the partial products are reduced as soon as possible. In contrast, Dadda's method does the minimum reduction necessary at each level [6]. The Wallace multiplier uses slightly smaller adders than Dadda multiplier [6]. Therefore, even if the Dadda multiplier is also fast multiplier Wallace tree multiplier is selected because it reduces the number of operands, actual partial products, at the earlier stages. However, there a number of algorithms have been implemented for Wallace tree multiplier. For example, figure 2.2 is the algorithm implemented at one of the university in the USA. However, there are four levels in this algorithm which results in significant delay in the addition of the partial products. But, someone improved the algorithm so that the level will be reduced by one, see figure 2.4. [6] [18]

Assuming that there are six partial products y0, y1, y2, y3, y4, and y5 and let us see how these two different algorithms perform the additions of the six partial products. If you look at figure 2.3 there are four levels and there are three levels for figure 2.4. In general, the idea behind this work is to reduce the number of levels/stages so as to reduce the propagation delay.

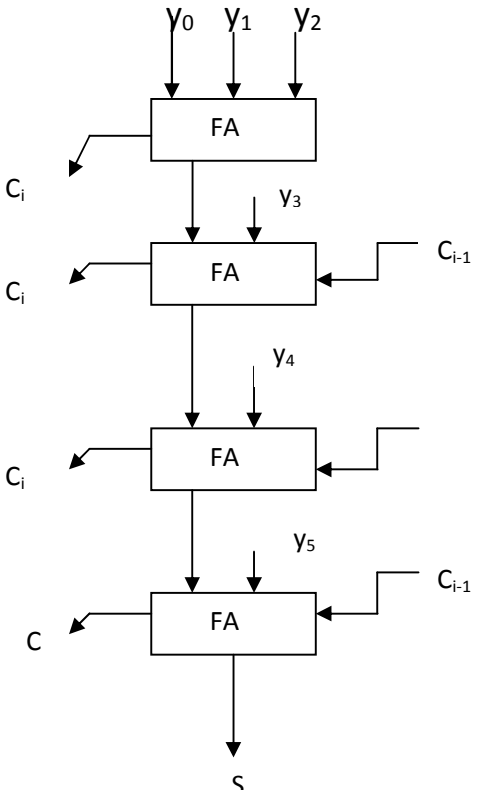

Figure 2.3: Diagram to add six Partial

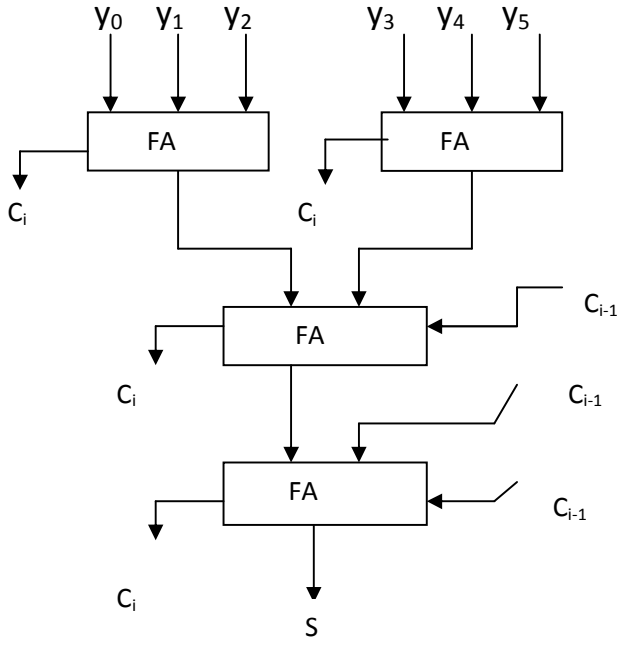

Figure 2.4: Modified diagram of Figure 2.3

Still there is a problem with this algorithm because it takes larger number of adders which results in delay. Let us see how the proposed algorithm is different from the algorithm implemented in figure 2.3.

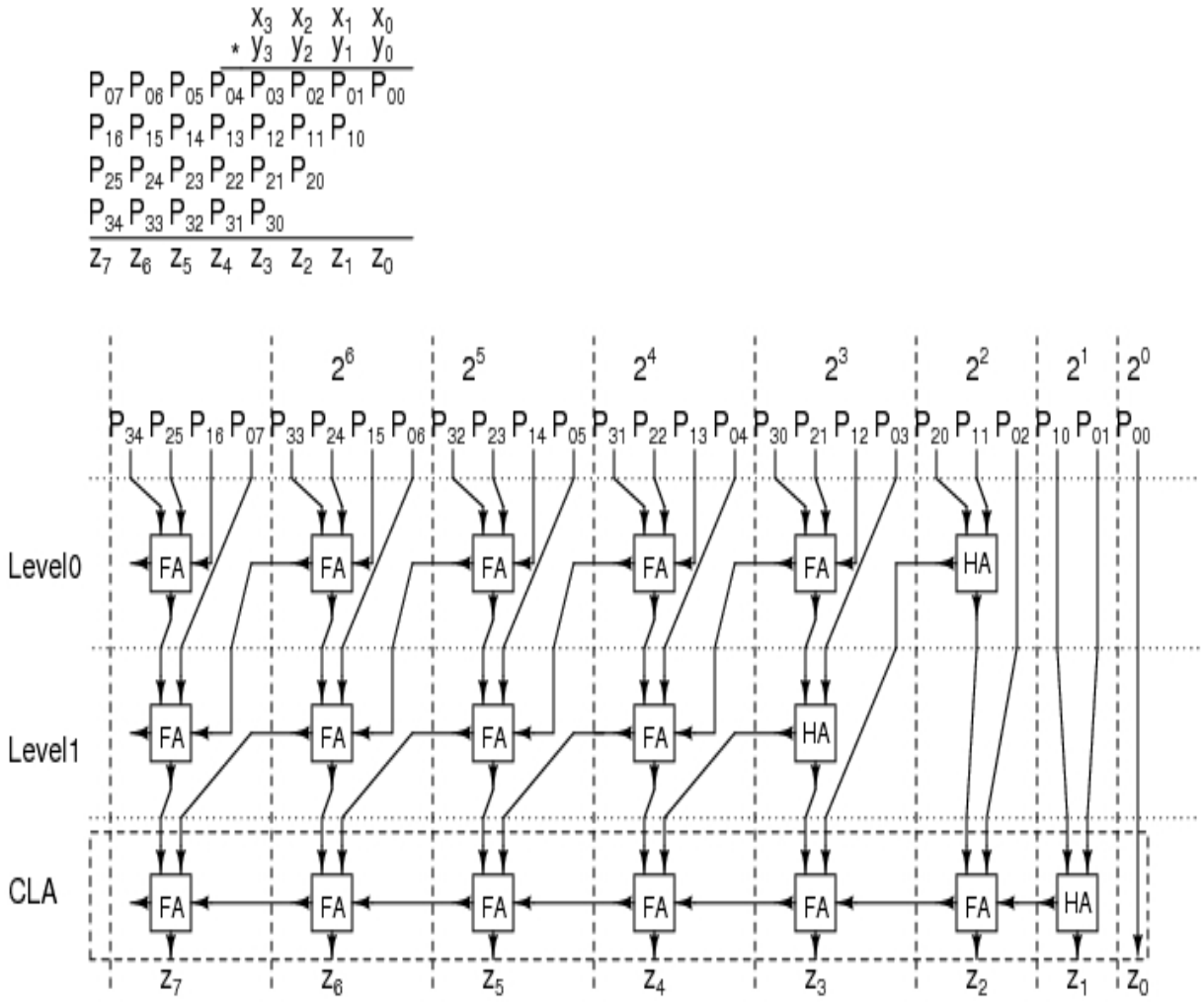

Figure: 2.5: 4-bits*4-bits Wallace tree multiplier expanding algorithm in figure 2.4

According to the figure above, the Wallace tree only need 18 adders (15 Full-adders and 3 Half-adders). However, for the proposed algorithm there will be 12 adders (8 full adders and 4 half adders), see figure 5.2.

Proposed Wallace tree Algorithm

Before designing the proposed algorithm of 8-bits *8-bits Wallace tree multiplier, the design and verilog code of 4-bits*4-bits is given below. A 4bit * 4bits Wallace tree multiplier is implemented in verilog to demonstrate the proposed multiplier. The figure below shows the design of a 4bit * 4bits Wallace tree multiplier.

A_3 A_2 A_1 A_0

*B_3 B_2 B_1 B_0

S_{30} S_{20} S_{10} S_{00}

S_{31} S_{21} S_{11} S_{01}

S_{32} S_{22} S_{12} S_{02}

S_{33} S_{23} S_{13} S_{03}

P_7 P_6 P_5 P_4 P_3 P_2 P_1 P_0

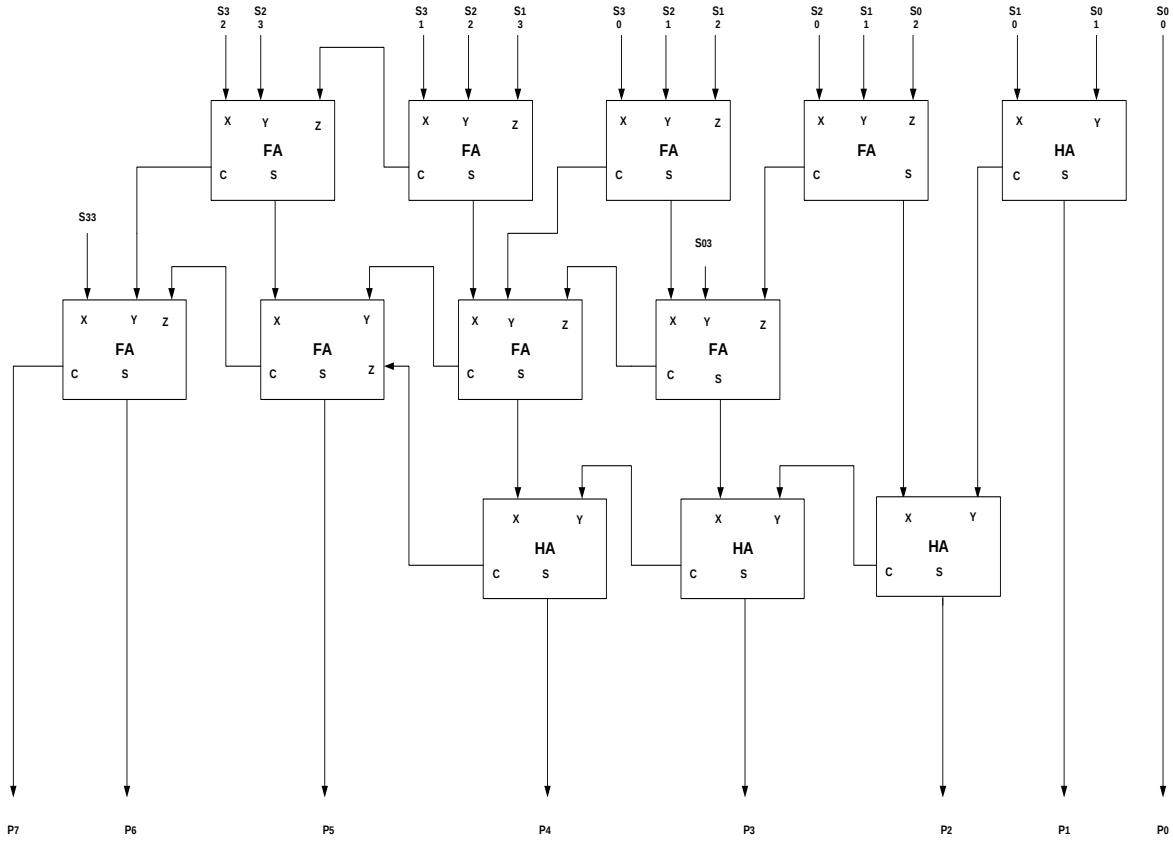

Figure 2.6: 4bits * 4bits Wallace tree multiplier.

According to the figure above, the Wallace tree only need 12 adders (8 Full-adders and 4 Half-adders). Now we precede to the design of 8 bits* 8 bits Wallace tree multiplier. The high level diagram of the proposed algorithm will look like the following:

A7 A6 A5 A4 A3 A2 A1A0

*B7 B6 B5 B4 B3 B2 B1B0

---

s70 s60 s50 s40 s30 s20 s10 s00

s71 s61 s51 s41 s31 s21 s11 s01

s72 s62 s52 s42 s32 s22 s12 s02

s73 s63 s53 s43 s33 s23 s13 s03

s74 s64 s54 s44 s34 s24 s14 s04

s75 s65 s55 s45 s35 s25 s15 s05

s76 s66 s56 s46 s36 s26 s16 s06

s77 s67 s57 s47 s37 s27 s17 s07

---

p15 p14 p13 p12 p11 p10 p9 p8 p7 p6 p5 p4 p3 p2 p1 p0

Discussion

The result of the study made on the delay of Dadda and Wallace tree multiplier by "Computer Engineering Research Center", the University of Texas at Austin is given in the following table 8.1. All values displayed in nanoseconds (ns)

Table 1: Delay for multipliers with RCAs

Multiplier size	**Dadda Delay**	**Wallace Delay**
4 by 4	19(100%)	21(111%)
8 by 8	**37(100%)**	**42(114%)**
16 by 16	69(100%)	77(112%)
32 by 32	133(100%)	145(109%)

Table 2: Delay for multipliers with CLAs

Multiplier size	**Dadda Delay**	**Wallace Delay**
4 by 4	15(100%)	18(120%)
8 by 8	**29(100%)**	**31(107%)**
16 by 16	43(100%)	45(105%)
32 by 32	54(100%)	56(104%)

Table 3: Delay for Proposed Algorithm of Wallace tree multiplier

Multiplier size	Logic gate delay	Route delay	Net delay
8 by 8	12.330(44.7%)	15.239(55.3%)	**27.569 (100%)**

As we can see at table 3, the proposed algorithm of Wallace tree multiplier has less net delay than those in table 1 and table 2. The total hardware used to implement this algorithm is 105 Adders of which 57 are 1-bit adder carry out and 48 are 2-bit adder.

Schematic diagram

The schematic diagram of the result is given in the following diagram.

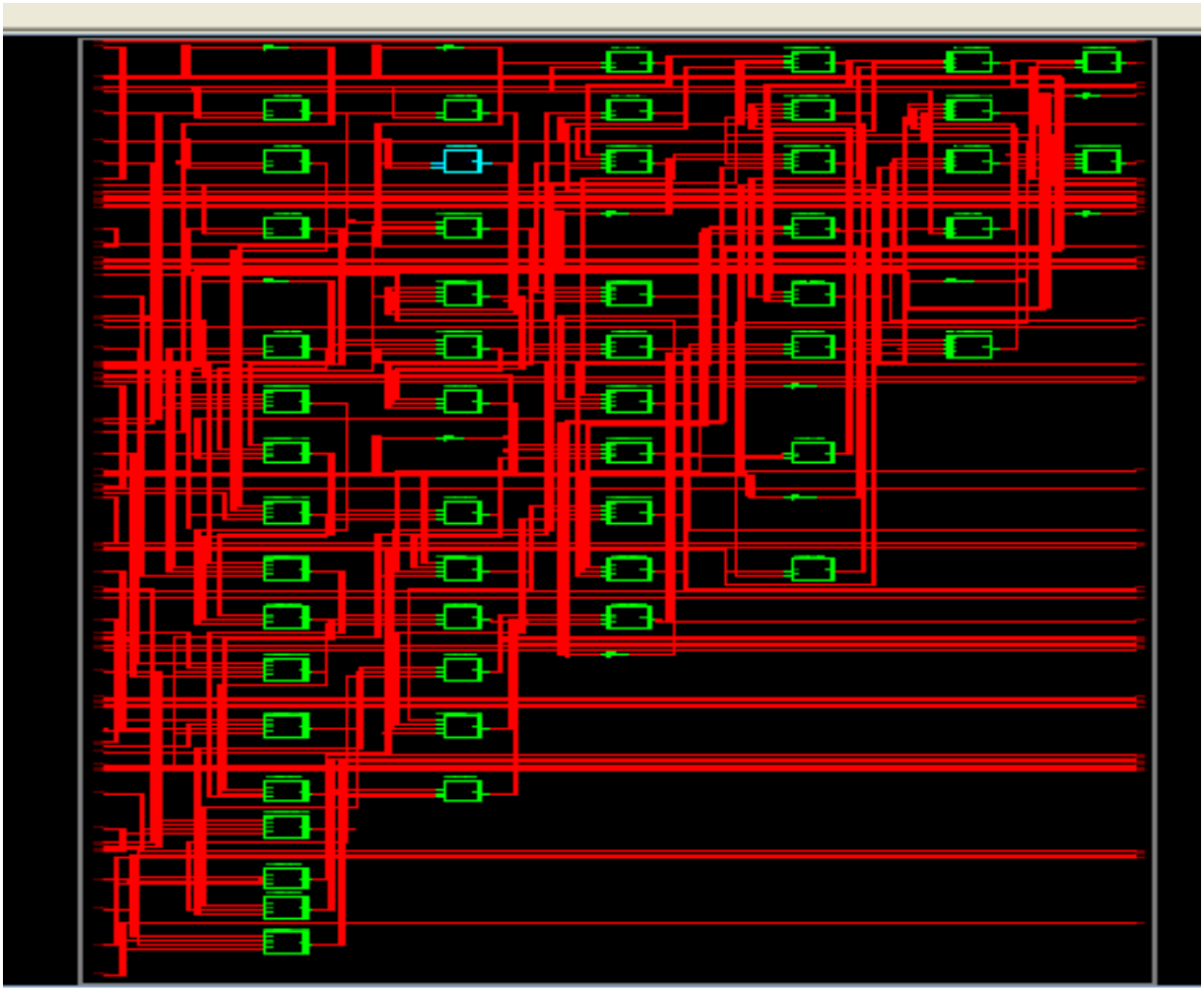

Figure xx: Schematic diagram of the proposed algorithm for 8 bit by 8 bit

3. Conclusion

As shown above this project tried to present three of the most available fast multipliers: Array multiplier, Wallace tree multiplier and Dadda multiplier. Besides, the thesis stated the work result that compares the net delay of Wallace tree and Dadda multipliers at Texas University. After designing, simulating and synthesizing the proposed algorithm of Wallace tree multiplier, it is possible to conclude that the proposed algorithm result has

less net delay than that of work result obtained at Texas University. In general, as multiplier size grows the Wallace tree multiplier requires slightly less hardware (in terms of adders or gates) than the Dadda multiplier.

Future work

In this paper designed, simulated, synthesized and implemented an 8-bit by 8-bit Wallace tree multiplier with improved algorithm only for the unsigned integers. However, the same concept can be used to realize multiplication of signed integers, signed real numbers and FPGU (Floating Point Arithmetic Unit). Further, the proposed algorithm can be applied for higher sizes of multiplier (16 by 16, 32 by 32 and more).

Reference

Brown, Richard, “A Microprocessor Design Project in an Introductory VLSI Course”, IEEE Transactions on Education, Vol. 43, No. 3, August 2000.

Hamblen, James and Furman, Michael, Rapid Prototyping of Digital Systems, 2nd edition, Boston: Kluwer Academic Publishers, 2001.

Hamacher, Vranesic, and Zaky. Computer Organization, 5th edition, New York: McGraw-Hill Companies, 2002.

Liao, and Roberts, “A High-Performance and Low-Power 32-bit Multiply- Accumulate Unit With Single-Instruction-Multiple-Data (SIMD) Feature”, IEEE Journal of Solid-State Circuits, Vol. 37, No. 7, July 2002.

C.S. Wallace, “A Suggestion for a Fast Multiplier," IEEE Trans. Computers, vol. 13, no. 2, pp. 14-17, Feb. 1964.

Whitney J. Townsend, Earl E. Swartzlander, Jr., and Jacob A. Abraham], “SPIE Advanced Signal Processing Algorithms, Architectures, and Implementations XIII,” pp. 552-560, San Diego, CA, August 6-8, 2003

M. Morris Mano, “Computer system architecture”, Third edition, California State University, Los Angeles.1993 by prentice-Hall, inc., New Jersey 07458, USA.

L. Dadda, "Some Schemes for Parallel Multipliers," *Alta Frequenza,* vol. 34, pp. 349-356, Mar. 1965.

B.D. Lee and V.G. Oklobdzija,"Delay Optimization of Carry-Lookahead Adder Structure," *J. VLSI Signal Processing,* vol. 3, no. 4, Nov. 1991.

Vojin G. Oklobdzija, David Villeger, Simon S. Liu, "Method for Speed Optimized Partial Product Reduction and Generation of Fast Parallel Multipliers Using an Algorithmic Approach," March 1996(vol. 45, No.3) pp. 294-306

Verilog HDL: A Guid to Digital Design and Synthesis, Second Edition, published by Pearson Education (Singapore),2003

C.S Wallace, "A suggestion for fast multiplier,"IEEE Trans.on computers," Vol.13, pp14-17, 1964.

L.Dadda,"Some schemes for parallel multipliers," Alta Frequenza, vol.34, pp.349-356, 1965.

B.Parhami, Computer Arithmetic Algorithms and Hardware designs, New York: Oxford University press, 2000.

E.E. Swartzlander Jr.,"Merged arithmetic,"Vol.29, pp.946-950, 1980.

K.A.C.Bickerstaff, M.Schutle, and E.E. Swartzlander Jr., "Reduced area multipliers," Intl.Conf. on Application-Specific Array Processors, pp.478-489, 1993

E.E. Swartzlander Jr. and G.Goto,"Computer arithmetic," The computer Engineering Handbook, V.G. Oklobdzija, ed., Boca Raton, FL: CRC press, 2002

A.Habib and P.A.Wintz,"Fast multipliers," IEEE Trans.on Computers, vol.19, pp.153-157, 1970