Implementation of a CMOS Wallace-tree Multiplier

Xiaoping Li, Xingguo Xiong, Hassan Bajwa, Prabir Patra
Department of Electrical and Computer Engineering,
University of Bridgeport, Bridgeport, CT 06604

Abstract

As slow and expensive operation units, multipliers are often the bottleneck limiting the overall performance of many computational VLSI circuits. Various CMOS multiplier architectures are available, such as the array multiplier, carry-save multiplier, and Wallace-tree multiplier. Wallace-tree multiplier has been a very popular design due to its fast speed, ease for modularization and fabrication. In this paper, the design and simulation of an 8-bit Wallace-tree multiplier with PSPICE is proposed. In order for comparison, an 8-bit CMOS array multiplier is also designed. The worst-case delay of both multiplier architectures are extracted and Wallace-tree multiplier demonstrates significant speed enhancement compared to CMOS array multiplier. Some efforts are made to further improve the performance of Wallace-tree multiplier. The revision in the circuit structure demonstrates effective speed improvement for the Wallace-tree multiplier.

I. Introduction

"CMOS" refers to both a particular style of digital circuitry design, and the family of processes used to implement that circuitry on integrated circuits (chips). CMOS circuitry dissipates less power when static, and is denser than other implementations having the same functionality. As this advantage has grown and become more important, CMOS processes and variants have come to dominate, so that the vast majority of modern integrated circuit manufacturing is on CMOS processes. A Wallace tree is an efficient hardware implementation of a digital circuit that multiplies two integers. A multiplier based on Wallace-tree structure is called Wallace multiplier. It is substantially faster than conventional carry-save structure.

II. CMOS Design

In CMOS (*Complementary Metal-Oxide Semiconductor*) technology, both N-type and P-type transistors are used to realize logic functions. Today, CMOS technology is the dominant semiconductor technology for microprocessors, memories and application specific integrated circuits (ASICs). The main advantage of CMOS over NMOS and bipolar technology is the much smaller power dissipation. Unlike NMOS or bipolar circuits, a CMOS circuit has almost no static power dissipation. Power is only dissipated in case the circuit actually switches. This allows to integrate many more CMOS gates on an IC than in NMOS or bipolar technology, resulting in much better performance.

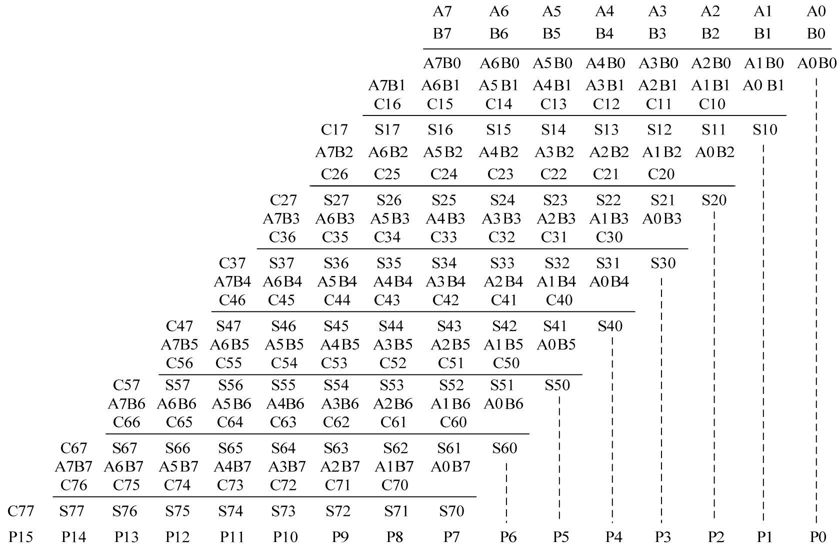

Figure 1. 8-bits multiplier design principle

In general, an n-bit-by-n-bit array multiplier would require n^2 AND gate, n(n-2) full adders, and n half adders.

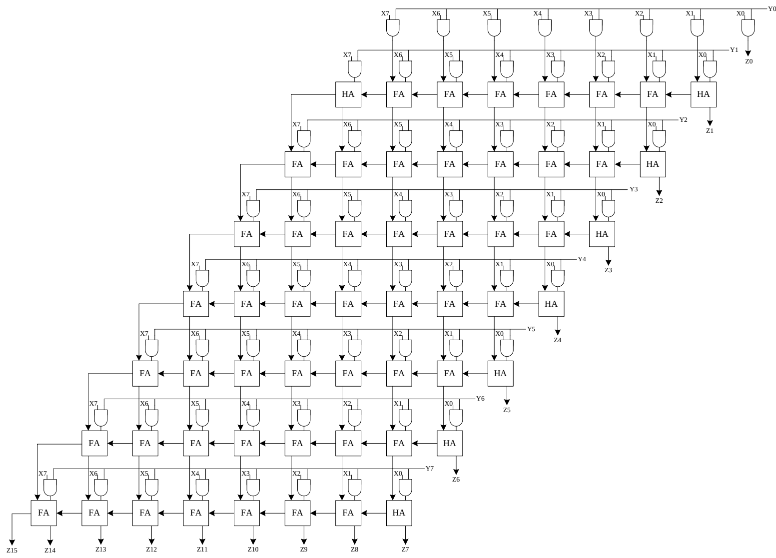

Figure 2. 8-bits multiplier block diagram

In the above figure, we can see we use 64 AND gates, 48 Full adders and 8 half adders. The total delay time is:

$$T_{total} = 13T_{carry} + 7T_{sum} + T_{and} \quad (1)$$

Where Tcarry is the propagation delay between input and output carry, Tsum the delay between the input carry and sum bit of the full adder, and Tand the delay of the AND gate.

In this project, we use PSpice (a kind of software for designing and simulating CMOS circuit) to build the circuit. See figure 3 below. This is the schematic design of 8-bit array multiplier

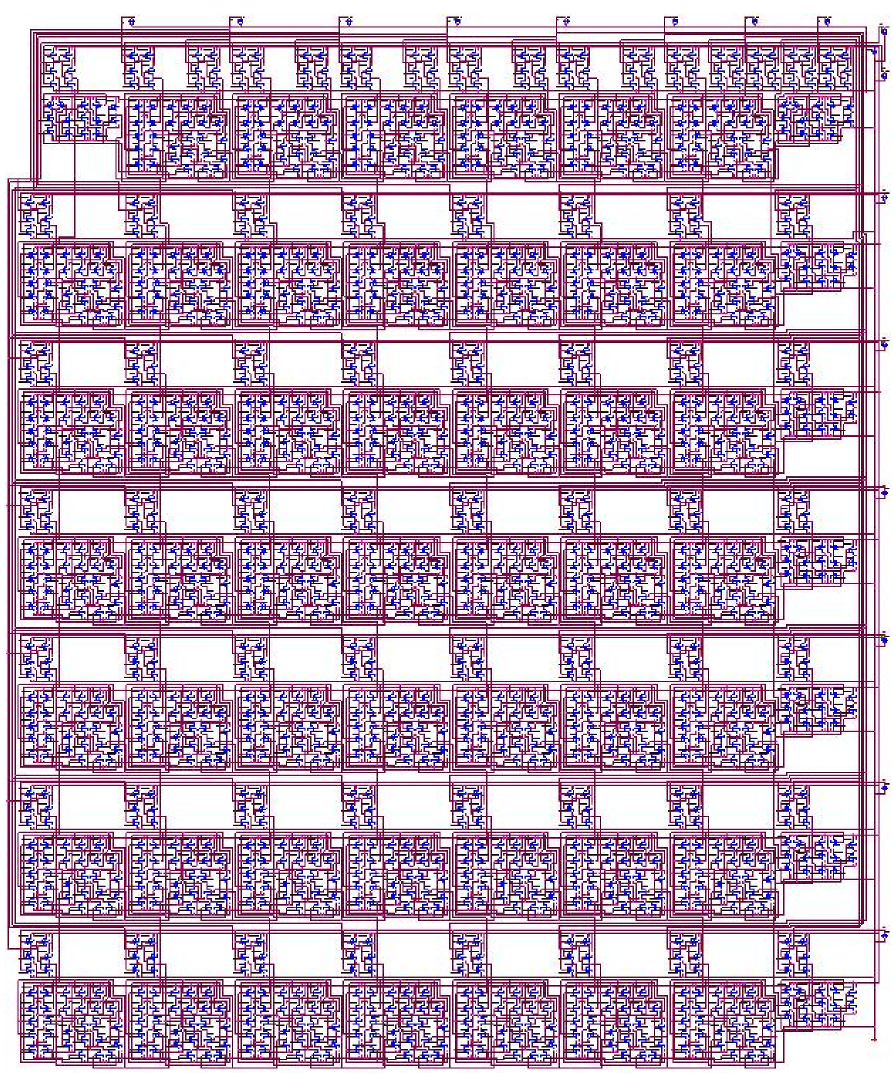

Figure 3. Schematic design of 8-bit array multiplier

In this simulation, I give 3 patterns to both input X and input Y, and both of them are 8 bits, so the answer Z should be 16bits. And I should also get 3 patterns output. See the table above, the first pattern I give is input X7X6X5X4X3X2X1X0 = 11101100, input Y7Y6Y5Y4Y3Y2Y1Y0 = 11010110, so the output Z15Z14Z13Z12Z11Z10Z9Z8Z7Z6Z5Z4Z3Z2Z1Z0 = 1100010101001000. The second pattern I give is input X7X6X5X4X3X2X1X0 = 10001111, input Y7Y6Y5Y4Y3Y2Y1Y0 = 01100111, so the output Z15Z14Z13Z12Z11Z10Z9Z8Z7Z6Z5Z4Z3Z2Z1Z0 = 0011100011111010. The third pattern I give is input X7X6X5X4X3X2X1X0 = 00100111, input Y7Y6Y5Y4Y3Y2Y1Y0 = 00001101, so the output Z15Z14Z13Z12Z11Z10Z9Z8Z7Z6Z5Z4Z3Z2Z1Z0 = 0000000111111011.

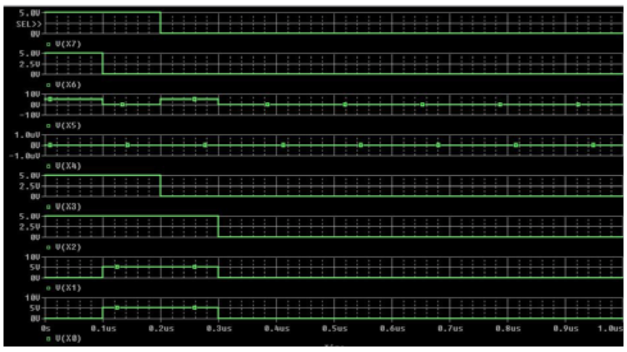

Figure 4. X trace

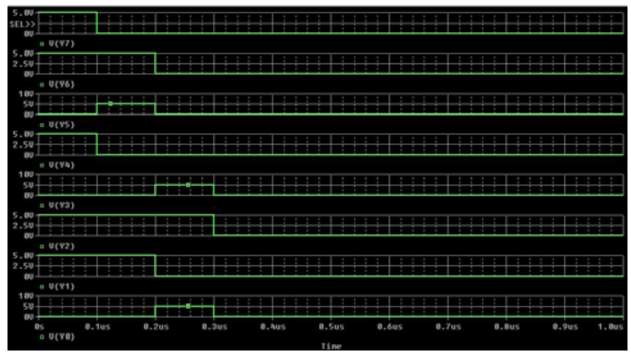

Figure 5. Y trace

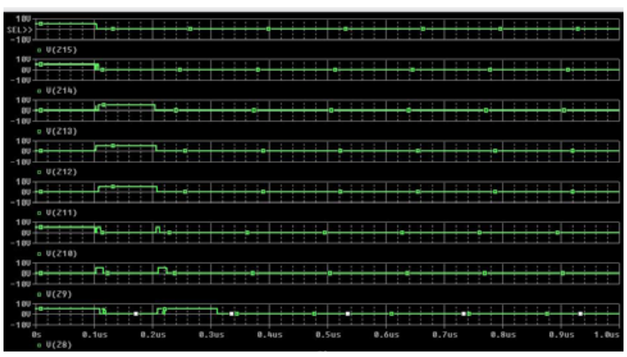

Figure 6. Z15-Z8 trace

Figure 7. Z7-Z0 trace

In the figures above, we can see all of the inputs and outputs trace are correct. However, the output is delayed. According to the logic, Z15 will be changed at latest. So if we can know the changing time between Z15 and any one of the inputs, we can get the delay of this circuit. Following is the trace of X0 and Z15

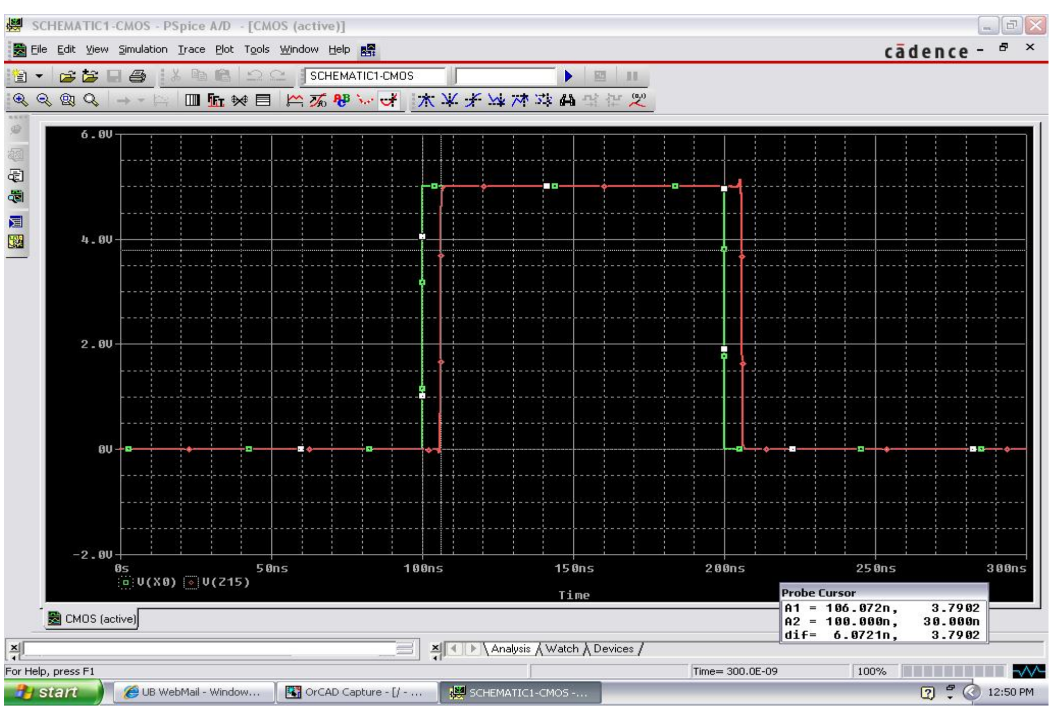

Figure 8. trace for delay calculation

The green line is voltage trace of X0 and red line is voltage trace of Z15. When X0 changes from 0 to 1, Z15 will change later. The delay time which we can read out using cursor is 6.0721ns.

III. Wallace-tree Design

A Wallace tree is an efficient hardware implementation of a digital circuit that multiplies two integers. A multiplier based on Wallace-tree structure is called Wallace multiplier. It is substantially faster than conventional carry-save structure.

The Wallace-tree has four steps:

1. Multiply (that is - AND) each bit of one of the arguments, by each bit of the other, yielding n^2 results. Depending on position of the multiplied bits, the wires carry different weights, for example wire of bit carrying result of a_2b_3 is 5.
2. Reduce the number of partial products to two by layers of full and half adders.
3. Repeat the step 2 till only one wire of each weight left.
4. Array these wires form highest weight to lowest and the result is the product.

Note: If there are three or more wires with the same weight add a following layer:

1. Take any three wires with the same weights and input them into a full adder. The result will be an output wire of the same weight and an output wire with a higher weight for each three input wires.
2. If there are two wires of the same weight left, input them into a half adder.
3. If there is just one wire left, connect it to the next layer

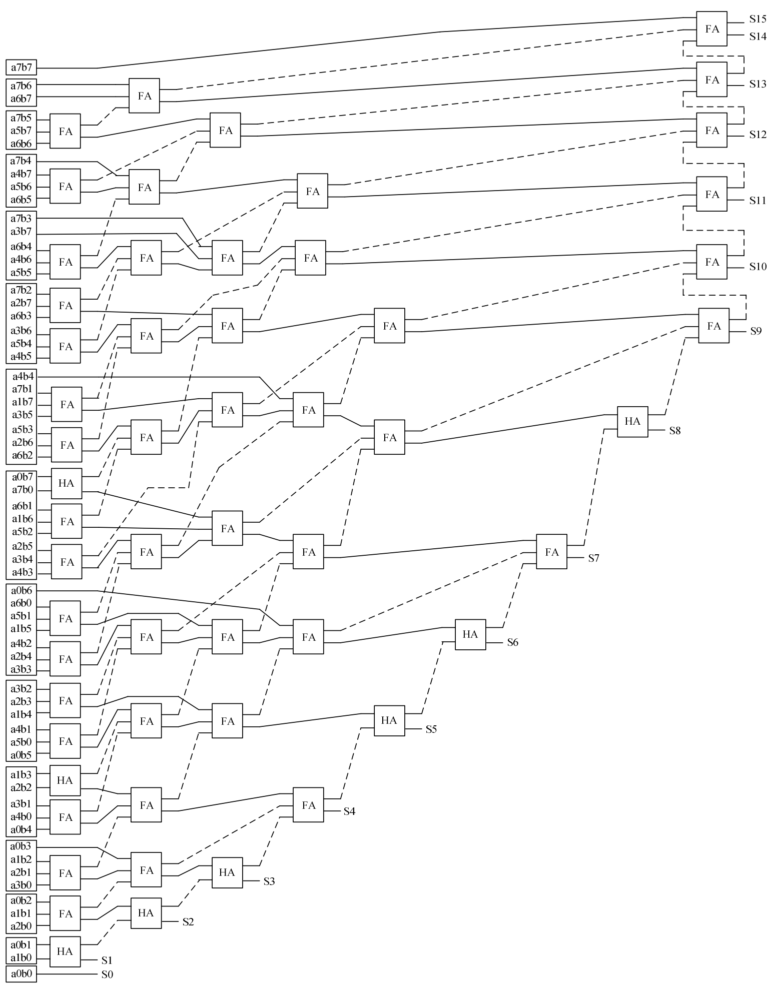

Figure 9. 8-bits Multiplier using Wallace-tree Design

we can get the total delay time with finding the worst line. The worst line pass 6 full adders' sum, 5 full adders' carry, 1 half adder's carry and 1 AND gate. So the total delay time (*not including wire delay*) of this design is:

$$T_{total} = 6T_{sum} + 5T_{carry} + T_{hcarry} + T_{and} \quad (2)$$

In figure 9, we can see in some weight (*for example: weight-13*) there are only few adders and long wires. During the most of time the last full adder must wait the lower weights to finish their calculation. It is not what we want to see. Since the delay time of a half adder is about only half of a full adder. We can divide these last full adders into half adders to promote this logic design. We can also reduce the number of cout to make the whole system work faster. See figures below:

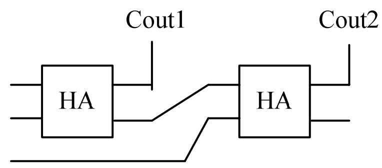

Figure 10. half adder array circuit

There two half adders in this figure. The sum of the former is the input of the latter. And there are 2 carry-outs in this circuit: Cout1 and Cout2. However Cout1 and Cout2 cannot be 1 synchronously, so we can use an OR gate to minimize them. See figure below.

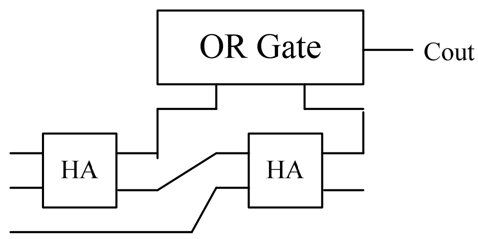

Figure 11. half adder array circuit with an OR gate

After the change, we can get the new block design of 8-bits multiplier

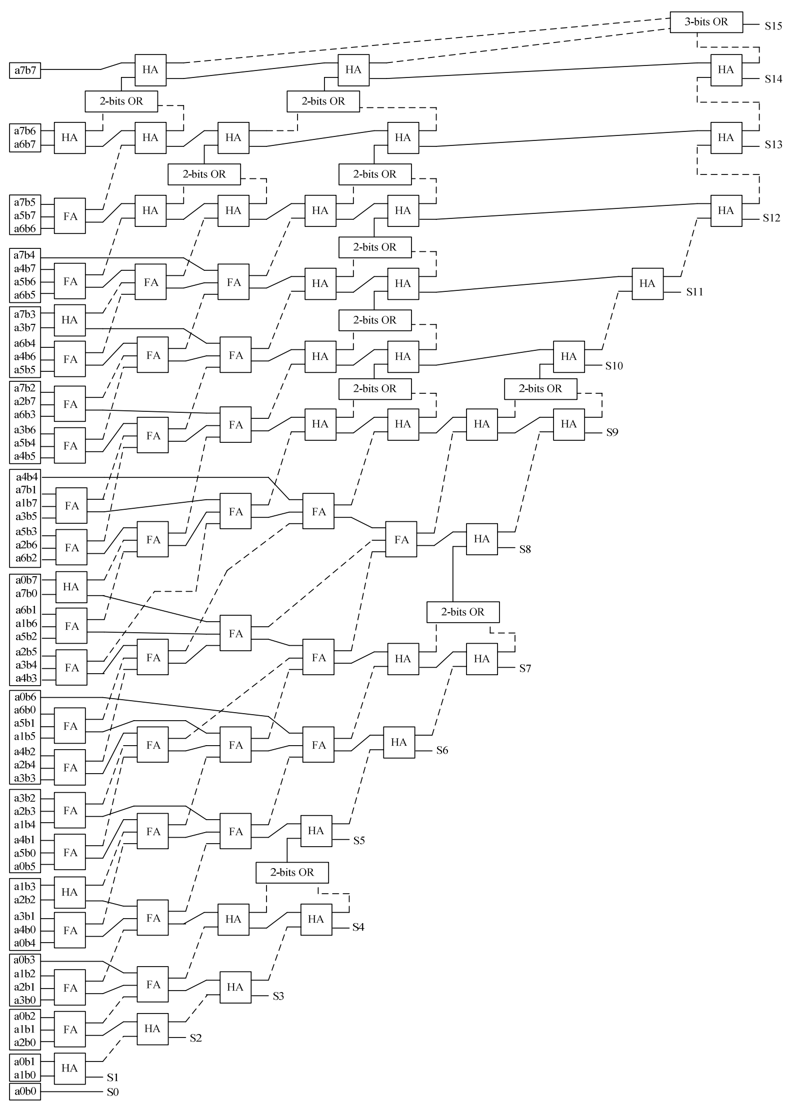

Figure 12. 8-bits Multiplier Design after Change

See the figure above. The worst line pass 5 full adders' sum, 7 half adders' carry, 1 AND gate, 1 3-bits OR gate and 1 2-bits OR gate. Then we can get the total delay time:

$$T_{total} = 5T_{sum} + 7T_{hcarry} + AND + 3bit\text{-}OR + 2bit\text{-}OR \qquad (3)$$

We also use PSpice to build the circuit.

Figure 13. Schematic design of 8-bit Wallace-tree multiplier

In this simulation, I give 3 patterns to both input A and input B, and both of them are 8 bits, so the answer S should be 16bits. And I should also get 3 patterns output. See the table above, the first pattern I give is input A7A6A5A4A3A2A1A0 = 11101100, input B7B6B5B4B3B2B1B0 = 11010110, so the output S15S14S13S12S11S10S9S8S7S6S5S4S3S2S1S0 = 1100010101001000. The second pattern I give is input A7A6A5A4A3A2A1A0 = 10001111, input B7B6B5B4B3B2B1B0 = 01100111, so the output S15S14S13S12S11S10S9S8S7S6S5S4S3S2S1S0 = 0011100011111010. The third pattern I give is input A7A6A5A4A3A2A1A0 = 00100111, input B7B6B5B4B3B2B1B0 = 00001101, so the output S15S14S13S12S11S10S9S8S7S6S5S4S3S2S1S0 = 0000000111111011.

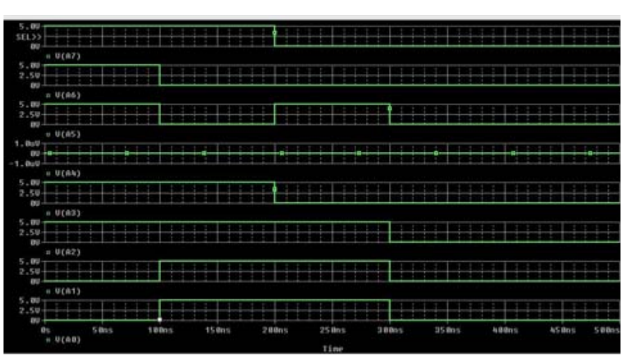

Figure 14. A trace

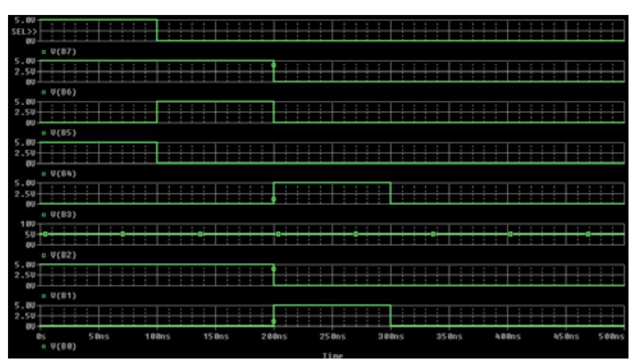

Figure 15. B trace

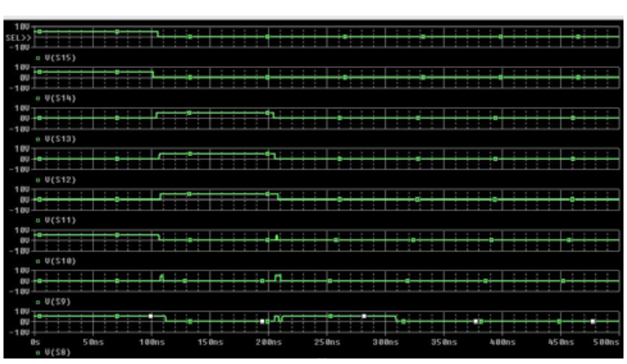

Figure 16. S15-S8 trace

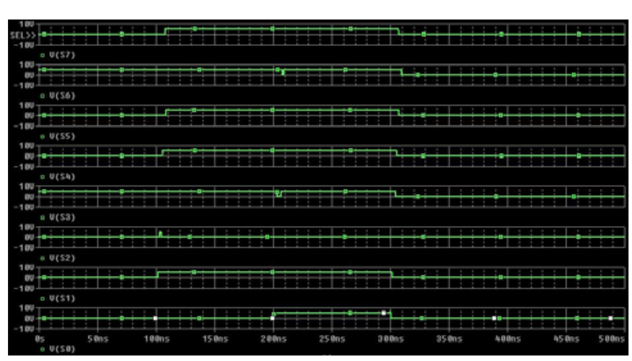

Figure 17. S7-S0 trace

According to the logic, S15 will be changed at latest. So if we can know the changing time between S15 and any one of the inputs, we can get the delay of this circuit. Following is the trace of A0 and S15

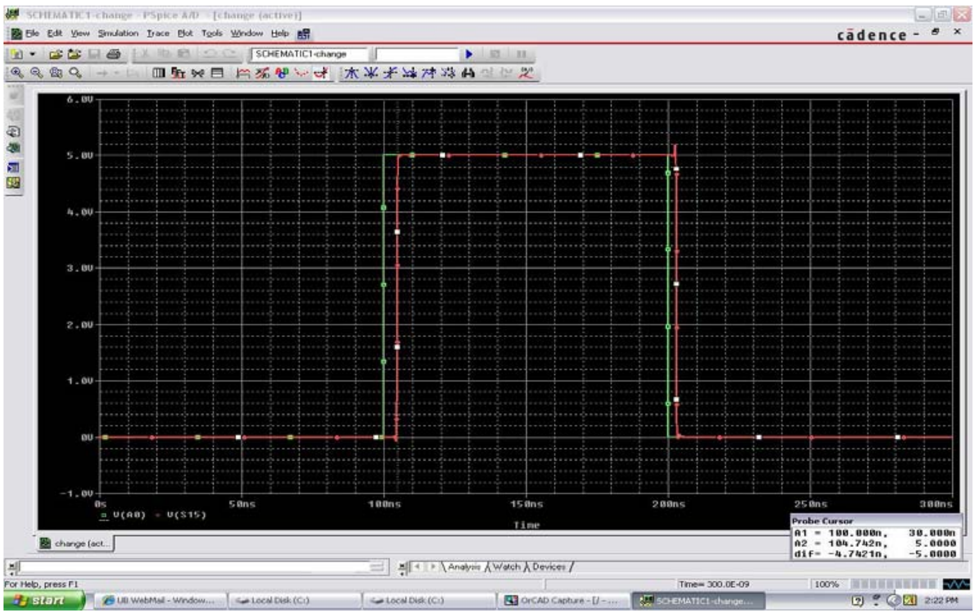

Figure 18. Trace for delay calculation

In the figure above, we can see the green line is the voltage of A0 and the red line is the voltage of S15. When input A0 change from 0 to 1 at the time100ns, the output S15 will also change from 0 to 1 after a short period of time. This time is called ‘delay time’. Using cursor we can see the delay time is 4.742ns.

IV. Conclusions and Future Work

In this paper, the design and simulation of an 8-bit Wallace-tree multiplier with PSPICE is proposed. In order for comparison, an 8-bit CMOS array multiplier is also designed. The worst-case delays of CMOS array multiplier and Wallace-tree multiplier are 6.0721ns and 4.742ns separately. The Wallace-tree multiplier demonstrates significant speed enhancement compared to CMOS array multiplier. From the speed perspective, Wallace-tree multiplier is preferred to CMOS array multiplier. However, CMOS Array Design is much more easy to design and manufacture in industry. In the future, we are going to look into how we could further improve the speed of Wallace-tree multiplier, and reduce the power consumption of the design.

References

[1] Neil H. E. Weste, David Harris, “CMOS VLSI design : a circuits and systems perspective,” *3rd ed.* pp. 10-30.

[2] Charles H. Roth, “Digital systems design using VHDL,” 1998, pp. 130-132.

[3] Jan M. Rabaey, “*Digital Integrated Circuits: A Design Perspective*”, Prentice Hall, 1996, ISBN: 8178089912, pp. 408-414.

[4] “M×N Booth-enclosed multiplier,” IEEE JSSC, vol.1, no.2, June 1993

[5] Townsend, W.; Swartzlander, E.; Abraham , J. (2003-08-06). "A Comparison of Dadda and Wallace Multiplier Delays". *SPIE Advanced Signal Processing Algorithms, Architectures, and Implementations XIII*.

Biographies

Xiaoping Li is a master student in Department of Electrical and Computer Engineering, University of Bridgeport, USA. His research area includes low power VLSI design and testing.

Dr. Xingguo Xiong is an assistant professor in Department of Electrical and Computer Engineering, University of Bridgeport, CT. He received his Ph.D degree in electrical engineering from Shanghai Institute of Microsystem and Information Technology, China, in 1999. He received his second Ph.D degree in computer engineering from University of Cincinnati, OH, USA in 2005. His research interests include microelectromechanical system (MEMS), nanotechnology, as well as VLSI design and testing.

Dr. Hassan Bajwa is an assistant professor of the Department of Electrical Engineering at the University of Bridgeport, CT. He received his Ph.D degree in engineering from City University of New York in 2007. His research interest include Low power VLSI, reconfigurable architectures, flexible electronics, Modeling and simulation of nano-electronic architectures.

Dr. Prabir Patra is an assistant professor of the Department of Mechanical Engineering and the Program director of Biomedical Engineering, University of Bridgeport, CT. He received his Ph.D degree in materials science and engineering engineering from Indian Institute of Technology, Kharagpur India in 1998. He was a postodoctoral research associate at UMASS Dartmouth and then moved to Rice University as a research scientist His research interests include multifunctional nanomaterials, bioinspired nanomaterials, polymer rheology, CNT and Graphene architecture, soft nanoelectronics, tissue engineering, polymer nanocomposite, energy harvesting hybrid nanomaterials, piezoelectric sensors and actuators and smart functional gels etc.