Journal of Theoretical and Applied Information Technology
10th August 2014. Vol. 66 No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

DESIGN OF LOW POWER MULTIPLIER USING COMPOUND CONSTANT DELAY LOGIC STYLE

[1] S. DARWIN, [2] A. BENO, [3] L. VIJAYA LAKSHMI
[1 & 2] Assistant Professor – Electronics & Communication Engineering Department,
Dr. Sivanthi Aditanar College of Engineering, Tamilnadu, INDIA
[3] PG Scholar, ME VLSI Design, Dr.Sivanthi Aditanar College of Engineering, Tamilnadu, INDIA
E-mail: [1]ssdarwin@live.com, [2]benoaustin@gmail.com [3]vijeyalaxmi@gmail.com

ABSTRACT

High performance, energy efficient logic style is a popular research topic in the field of very large scale integrated (VLSI) circuits. A complex constant logic style is used to implement a logic expression to achieve high speed operation. This logic style is well suited for arithmetic circuit where critical path comprises of large cascaded inverting gates. Multiplication is a most utilized arithmetic operator that forms a part of filters, convolvers, and transforms processors in digital signal processing applications. This paper focuses on the design of the Wallace tree multiplier, Baugh wooley and Array multiplier using static logic style, dynamic logic style and compound constant delay logic style .The performance of energy delay product of Wallace tree multiplier, array multiplier and Baugh wooley multiplier using compound constant delay logic style is reduced considerably while compared to static and dynamic logic style.

Keywords: *Wallace Tree multiplier, Array multiplier, Carry Skip adder, Cadence and Baugh Wooley Multiplier*

1. INTRODUCTION

In low power Very Large Scale Integration, different design levels like architectural, layout, circuit level and technology optimization level are addressed [1]. In the circuit design, the proper choices of levels are used to implement combinational logic circuits for power savings. The chosen logic style influences the parameters that govern the power dissipation, switching capacitance, transition activity and short circuit currents. This paper present the multipliers designed using static logic style, dynamic logic style and constant delay logic style. The power dissipation characteristics of various multipliers using static logic, dynamic logic and constant delay logic style are compared qualitatively and quantitatively with actual logic gate implementations.

2. LOGIC STYLE

The logic style used in logic gates influences the speed, size, power dissipation, and the wiring complexity of a circuit. The circuit delay can be determined from the number of inversion levels, number of transistors in series, transistor sizes i.e., channel widths, and intra- and inter-cell wiring capacitances. The circuit *size* is dependent on the number of transistors, its size and the wiring complexity. The switching activity and the node capacitances made up of gate, diffusion, and wire capacitances are used to determine the *power dissipation which* helps to control the circuit size. The *wiring complexity* is estimated by the number of connections, their lengths and type of rail logic used [2]. These characteristics vary from one logic style to another making the proper choice of logic style crucial for circuit performance.

2.1 Requirements of Low Power

The dynamic power is expressed as

$$P_{dynamic} = C V^2 f \quad (1)$$

It is directly proportional to the capacitance C, supply voltage and switching frequency. The power dissipation is reduced by proper selection of these parameters. When the frequency is decreased the dynamic power also decreases with an increase in delay as frequency is inversely proportional to delay.

2.2 Static Logic Style *vs* Dynamic Logic Style

Static Logic styles consist of pull up and pull down network. In dynamic circuit, clock is applied to make it function in two phases, a pre-charge and evaluation phase [3]. In dynamic logic

Journal of Theoretical and Applied Information Technology
10th August 2014. Vol. 66 No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

style, the logic function is implemented using only n-type devices while the clock is applied to both p-type and n-type devices. In pre-charge phase, the output node is pre-charged to high, while the path to ground is turned off. In the evaluation phase, the path to high is turned off when the clock is high and path to ground is turned on. Therefore, the output node will either be high or low depending on the input [4].

An advantage of dynamic logic style is that it reduces the silicon area. Therefore static logic style requires 2n transistors and dynamic logic style requires n+2 transistors. The disadvantage of dynamic logic style is the impossibility of cascading the blocks to implement complex logic and also excessive load on the clock signal that need to be connected to every dynamic gate [5].

2.3 Constant Delay Logic Style

When CLK is high, CD logic enters pre-discharge period and when CLK is low, CD logic enters the evaluation period. Three modes will take place like the contention, C–Q delay, and D–Q delay modes. When CLK is low, the circuit enters into the contention mode while input remains at logic "1." Here, it experiences a temporary glitch when *X* is at a nonzero voltage level. When input make a transition from high to low, C-Q delay modes take place before CLK becomes low. While CLK is at low, *X* rises to logic "1" and *Y* remains at logic "0" for the entire evaluation cycle. D–Q delay mode operates at the pre-evaluated characteristic of CD logic to enable high-performance operations. In this mode, CLK falls from high to low before input transit hence *X* initially raises to a nonzero voltage level [6].

3. DESIGN AND ANALYSIS

The multiplier architectures are studied for high speed signal processing applications, specifications for the multiplier design, modeling the architecture, functional verification, and developing the test bench to verify the design for all possible input combinations using logic style discussed in chapter 2.

3.1 Wallace Tree Multiplier

In 1964 C. S. Wallace introduced the multiplication based on summing the partial product bits in parallel using a tree of carry save adders known as the Wallace tree [7]. This method uses a three step process to multiply two numbers.

Step 1: Assigning two 8 bit numbers as an input.
Step 2: Finding the product of the bit formation.
Step 3: Using compressor technique, the bit product matrix is compressed to a two row matrix by using carry save adders known as Wallace tree.
Step 4: The last 12 bits are added using a ripple carry adder to produce the product.

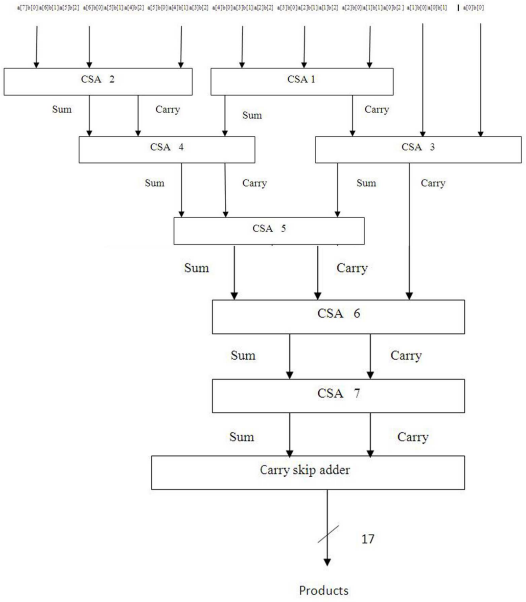

Figure 1: Block Diagram Of 8 Bit Wallace Tree Multiplier Using Static Logic Style

This method yields multipliers with delay proportional to log O(n). The block diagram of 8 bit Wallace tree multiplier is shown in Figure 1. The principle is to achieve partial product by reducing the number of bits in each column using full adders or half adders. From each column of the partial product matrix, three bits are added to produce two output bits, the sum bit in the same column and the carry bit in the next column. Hence the output matrix has been obtained by using the full adder known as the 3:2 compressors. The Wallace tree consists of numerous levels of column compressor structures till it remains with only two full width operands [9]. These two operands can then be added using regular 2N bit adders to obtain the product or 2:2 compressors respectively. Finally the partial product matrix has a depth of only two. The Wallace tree multiplier uses maximum hardware to compress the partial product matrix to obtain the final product.

The 8 bit Wallace tree multiplier using dynamic logic style is shown in Figure 2. The dynamic logic is identified when the clock goes high for which the input signal is applied to the D flip flop and output to the carry save adder. Here

Journal of Theoretical and Applied Information Technology
10th August 2014. Vol. 66 No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

three bits are added to form a partial product and produces two output bits with the sum and the carry. The sum and carry bits are applied to the next stage. The compressed bits is passed to carry propagate adder to generate product.

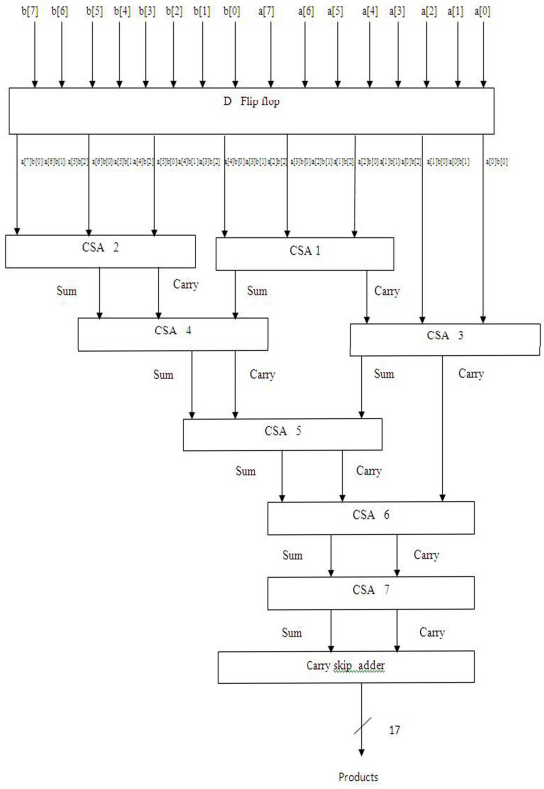

Figure 2: Block Diagram Of 8 Bit Wallace Tree Multiplier Using Dynamic Logic Style

The 8 bit Wallace tree multiplier using compound constant delay logic style is shown in Figure 3.The timing block consists of D flip flop, XOR gate and AND gate. The inputs to the XOR is the input and output of the D flip flop [7]. The clock signal and the output of XOR gate are applied to the AND gate to generate the clock signal. This signal acts as the clock signals for D flip flop. The output of timing block is applied to the carry save adder to form the partial product which is compressed to the sum and carry. Each stage of output is applied to the carry save adder till it propagates through the carry skip adder.

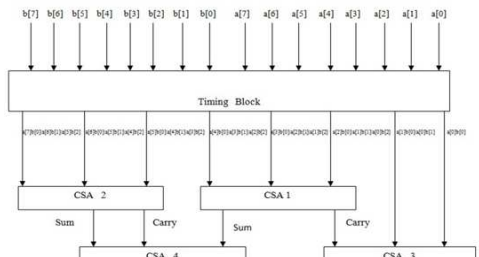

Figure 3: Block Diagram Of 8 Bit Wallace Tree Multiplier Using Compound Constant Delay Logic Style

3.2 Array Multiplier

Array multiplier is based on add and shift algorithm. Each partial product is generated by the multiplication of the multiplicand with one multiplier bit. The partial product are shifted according to their bit orders and then added.

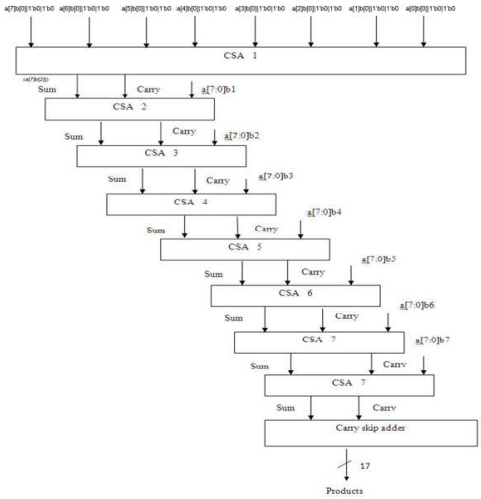

Figure 4: Block Diagram Of 8 Bit Array Multiplier Using Static Logic Style

The 8 bit array multiplier using static logic style is shown in Figure 4. The addition can be performed with normal carry propagate adder. N-1

Journal of Theoretical and Applied Information Technology
10th August 2014. Vol. 66 No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

adders are required where N is the multiplier length.

The 8 bit array multiplier using dynamic logic style is shown in Figure 5. The product of bits is applied to carry save adder at each stage. The sum and carry bits are applied to the next stage. Finally the partial products propagate through carry propagate adder to generate product [10].

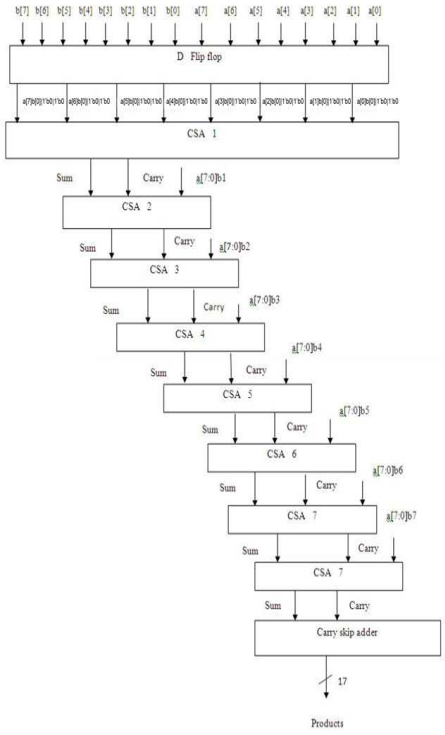

Figure 5: Block Diagram Of 8 Bit Array Multiplier Using Dynamic Logic Style

The 8 bit array multiplier using compound constant delay logic style is given in Figure 6. The data path used for the current input; the path will remain in precharge mode when the circuit switches ON. The timing block consists of D flip flop, XOR gate and AND gate. The input to the XOR is the input and outputs of the D flip flop. The clock signal and the output of XOR gate are applied to the AND gate to generate the clock signal. This signal acts as the clock signals for D flip flop. The output of timing block is applied to the carry save adder to form the partial product which is compressed to the sum and carry. The product of the bits is applied to the carry save adder. The result then propagates through the carry skip adder.

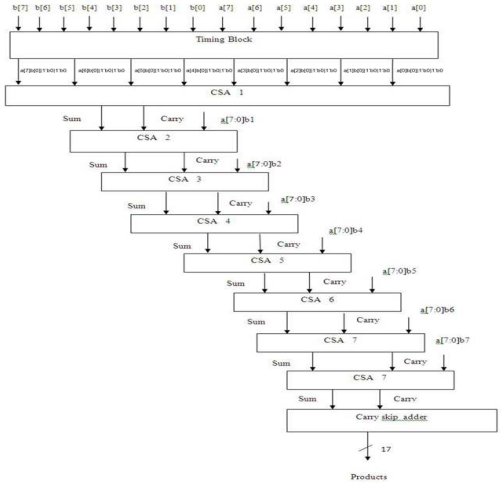

Figure 6: Block Diagram Of 8 Bit Array Multiplier Using Compound Constant Delay Logic Style

3.3 Baugh Wooley Multiplier

The Baugh wooley (BW) algorithm is a straightforward approach of performing signed multiplications. An 8-bit Baugh wooley multiplier using static logic style is shown in Figure 7, where the partial product bits have been reorganized according to Hatamian's scheme.

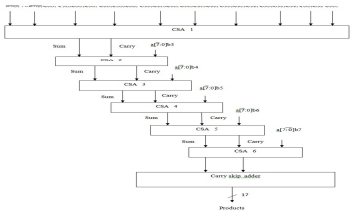

Figure 7: Block Diagram Of 8 Bit Baugh Wooley Multiplier Using Static Logic Style

The creation of the reorganized partial-product array of an *N*-bit wide multiplier comprises three steps: *i)* The most significant bit (MSB) of the first $N-1$ partial-product rows and all bits of the last partial-product row, except its MSB, are

Journal of Theoretical and Applied Information Technology
10th August 2014. Vol. 66 No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

inverted. *ii)* A '1' is added to the N^{th} column. *iii)* The MSB of the final result is inverted.

The 8 bit Baugh wooley multiplier using dynamic logic style is shown in Figure 8. All the input signals are applied to the D flip flop and it is activated when the clock is high [10]. Furthermore, the output is applied to the carry save adder. The sum and carry bits are generated when the three bits are added .The product of bits are applied to carry save adder at each stage. The sum and carry bits are applied to the next stage. Finally the partial products propagate through carry propagate adder to generate products

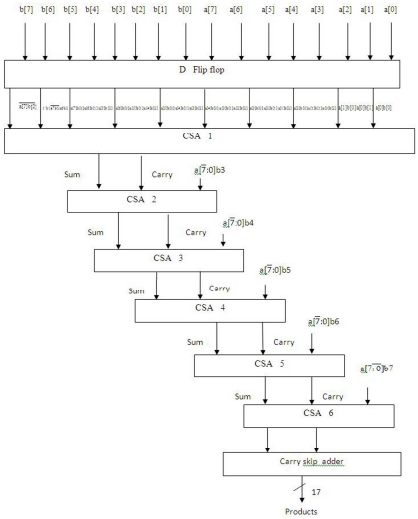

Figure 8: Block Diagram Of 8 Bit Baugh Wooley Multiplier Using Dynamic Logic Style

The 8 bit Baugh wooley multiplier using compound constant delay logic style is shown in Figure 9. The data path used for the current input, remains in the evaluate mode. The timing block consists of D flip flop, XOR gate and AND gate. The input to the XOR is the input and outputs of the D flip flop. The clock signal and the output of XOR gate are applied to the AND gate to generate the clock signal. This signal acts as the clock signals for D flip flop. The output of timing block is applied to the carry save adder to form the partial product which is compressed to the sum and carry. The product of the bits is applied to the carry save adder and gets propagating through the carry skip adder.

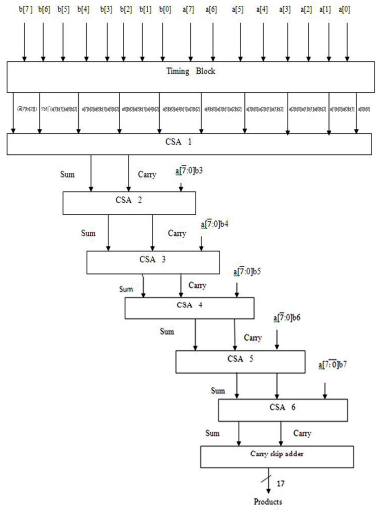

Figure 9: Block Diagram Of 8 Bit Baugh Wooley Multiplier Using Compound Constant Delay Logic Style

4. SIMULATION RESULTS

The integrated software environment (ISE) is Xilinx design software suit that allows taking the design from design entry through Xilinx device programming. The ISE project navigator manages and processes the design through varies steps in the design flow. Xilinx 9.1i provides design entry and synthesis supporting Very High speed integrated Hardware description Language (VHDL)/Verilog, place and route, completed verification and debug. The ISE design suit is the central Electronic Design Automation (EDA) product family sold by Xilinx.ISE controls all aspects of the design flow. Through the project navigator interface, we can access all the various design entry and design implementation tools.

In this paper, delay, power, power delay product and energy delay product are analyzed using Cadence tool. The analyzed data are computed for Wallace tree multiplier, Array multiplier and Baugh wooley multiplier using static logic style, dynamic logic style and constant delay logic style.

Journal of Theoretical and Applied Information Technology
10th August 2014. Vol. 66 No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

4.1 Results of multiplier using static logic style

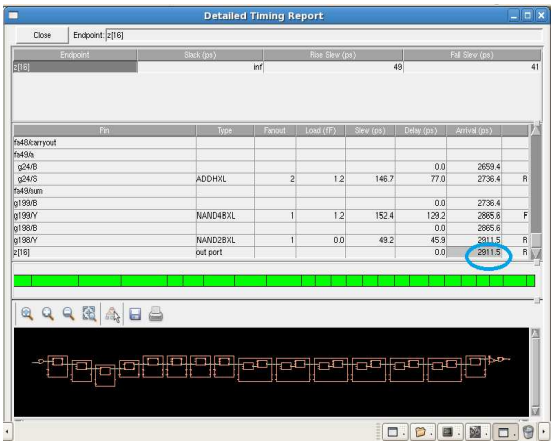

Figure 10: Delay Of Wallace Tree Multiplier

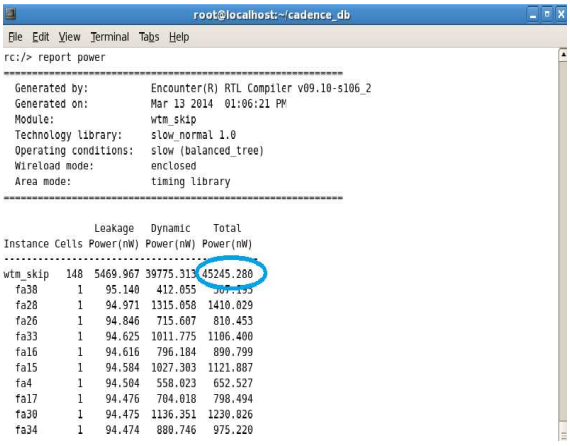

Figure 11: Power Of Wallace Tree Multiplier

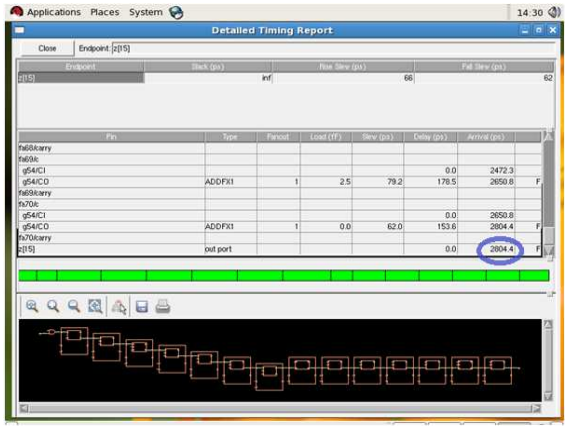

Figure 12: Delay Of Array Multiplier

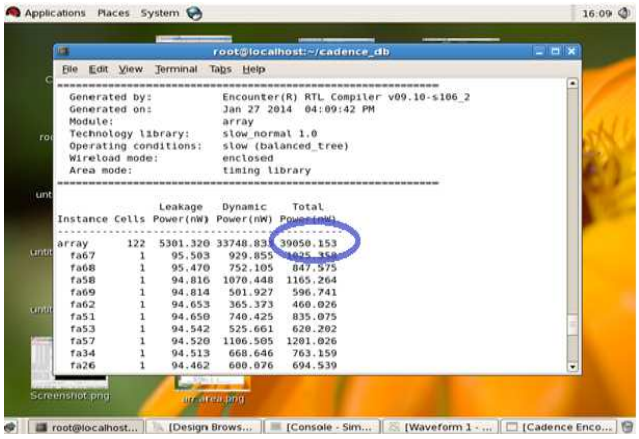

Figure 13: Power Of Array Multiplier

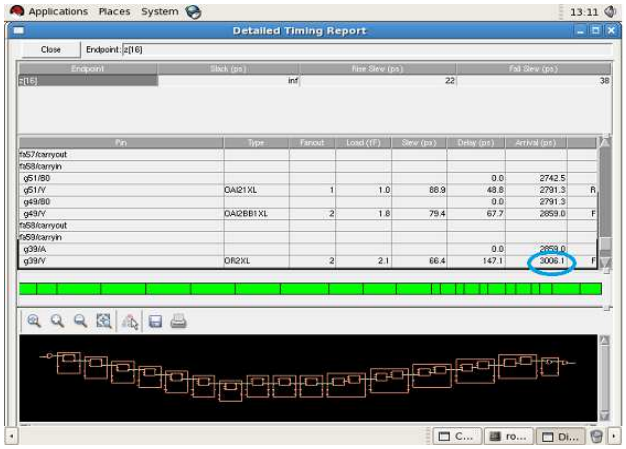

Figure 14: Delay Of Baugh Wooley Multiplier

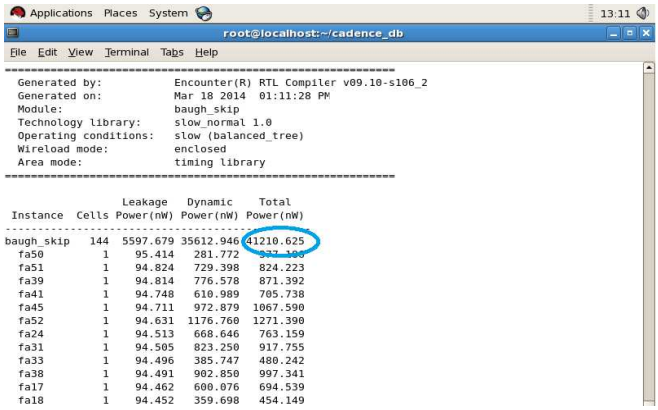

Figure 15: Power Of Baugh Wooley Multiplier

Journal of Theoretical and Applied Information Technology
10th August 2014. Vol. 66 No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

4.2 Results of Multiplier Using Dynamic Logic Style

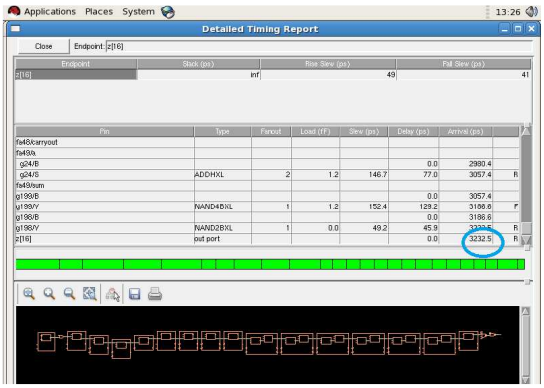

Figure 16: Delay Of Wallace Tree Multiplier

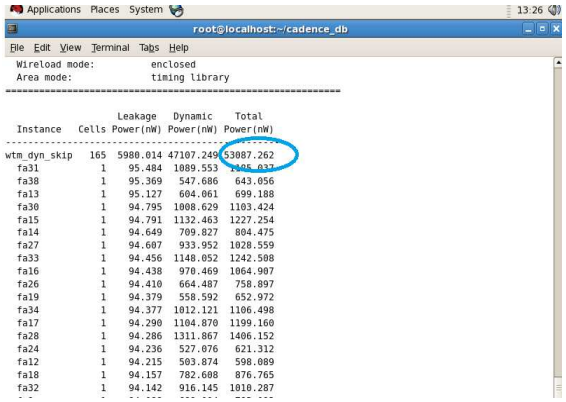

Figure 17: Power Of Wallace Tree Multiplier

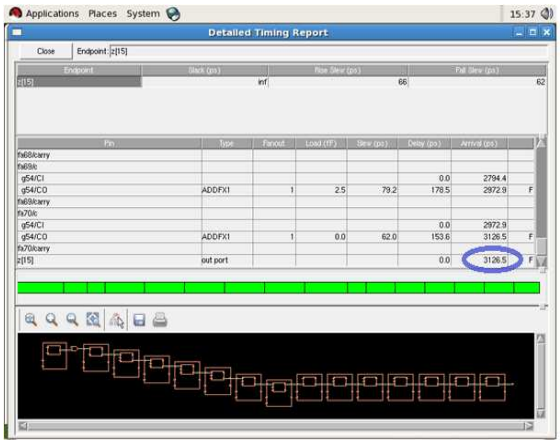

Figure 18: Delay Of Array Multiplier

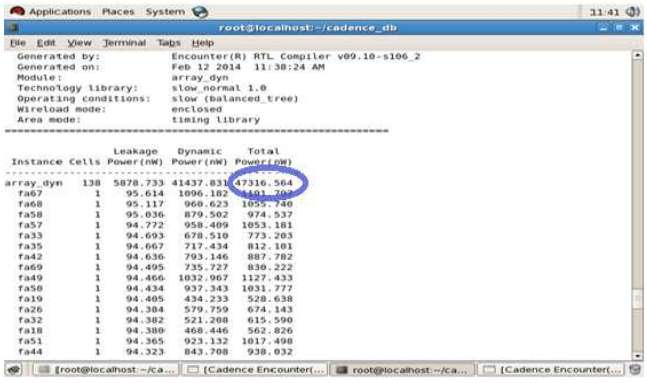

Figure 19: Power Of Array Multiplier

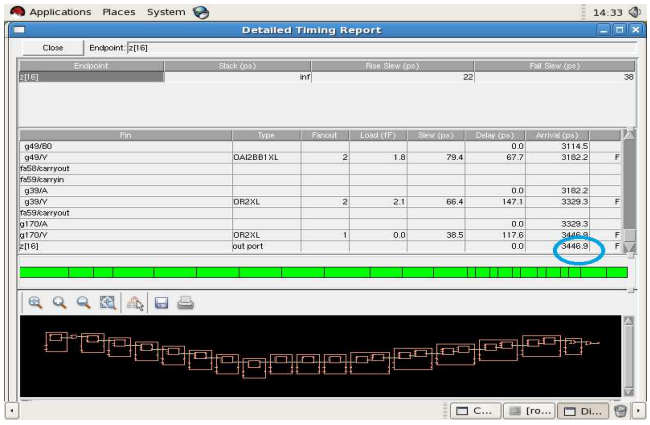

Figure 20: Delay Of Baugh Wooley Multiplier

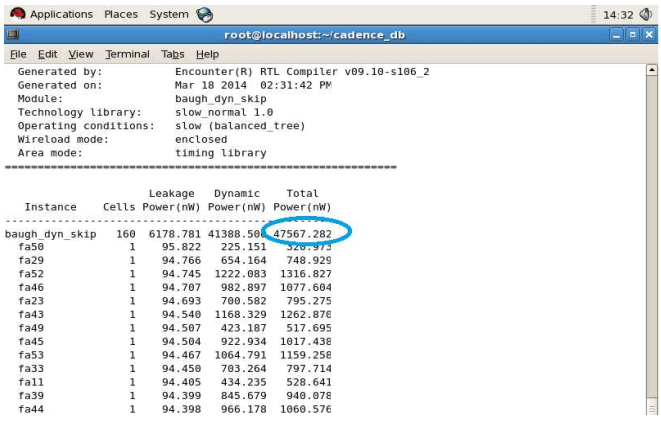

Figure 21: Power Of Baugh Wooley Multiplier

Journal of Theoretical and Applied Information Technology
10th August 2014. Vol. 66 No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

4.3 Results of Multiplier Using Compound Constant Delay Logic Style

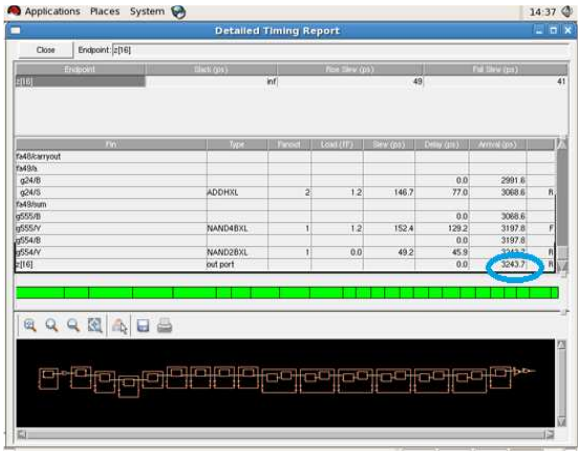

Figure 22: Delay Of Wallace Tree Multiplier

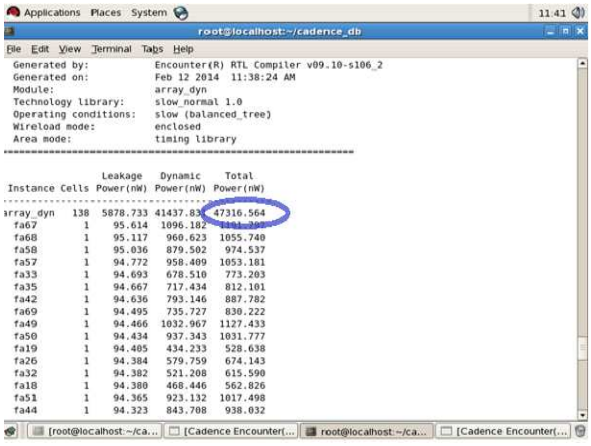

Figure 25: Power Of Array Multiplier

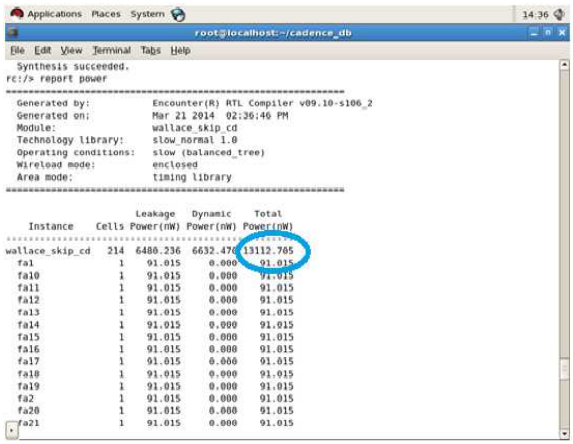

Figure 23: Power Of Wallace Tree Multiplier

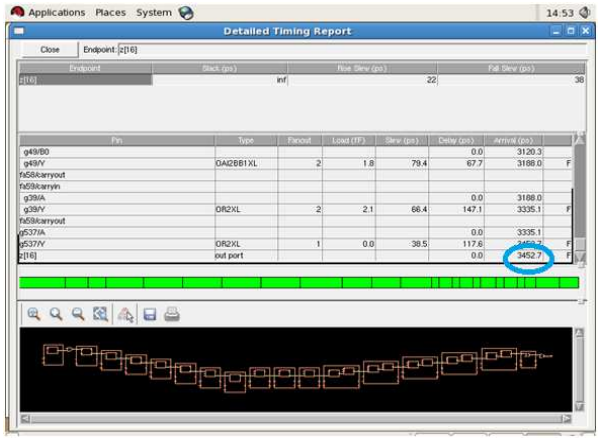

Figure 26: Delay Of Baugh Wooley Multiplier

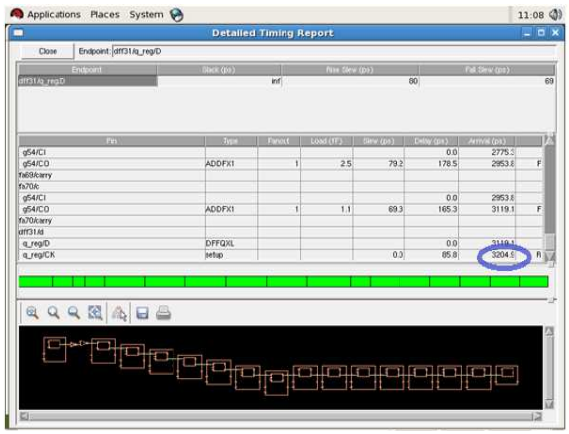

Figure 24: Delay Of Array Multiplier

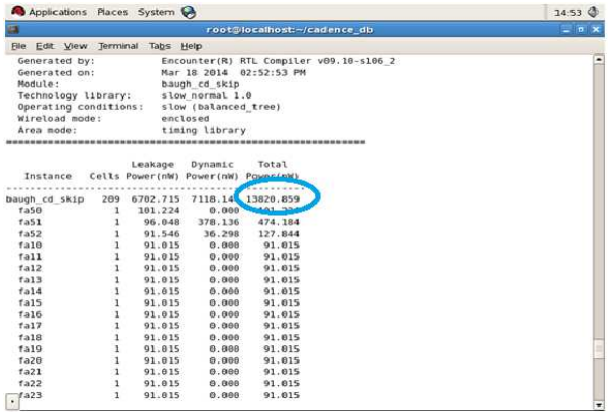

Figure 27: Power Of Baugh Wooley Multiplier

Journal of Theoretical and Applied Information Technology
10th August 2014. Vol. 66 No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

4.4 Performance analysis

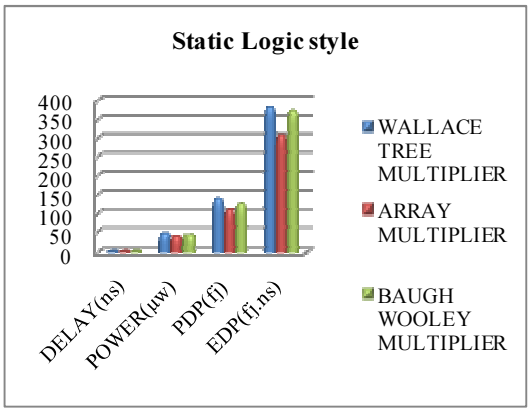

Figure 28: Performance Analysis Of Multipliers Using Static Logic Style

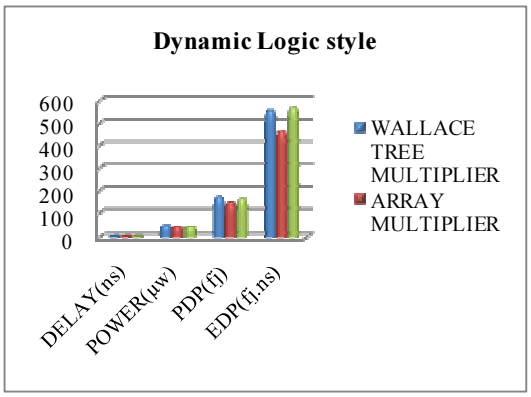

Figure 29: Performance Analysis Of Multipliers Using Dynamic Logic Style

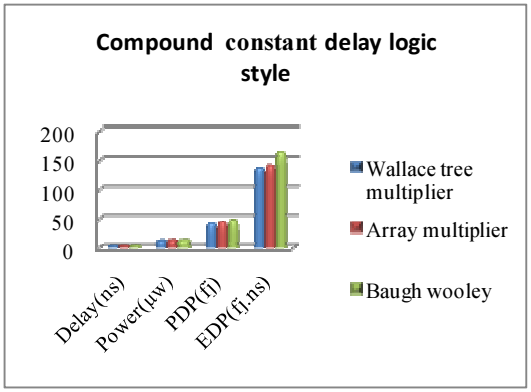

Figure 30: Performance Analysis Of Multipliers Using Compound Constant Delay Logic Style

4.5 Comparison table

Table 1: Static Logic Style

Types of multiplier	Delay (ns)	Power (µw)	PDP (fj)	EDP (fj.ns)
Wallace tree multiplier	2.911	45.245	131.1	381.63
Array multiplier	2.804	39.050	109.4	306.75
Baugh wooley multiplier	3.006	41.210	123.87	372.35

Table 2: Dynamic Logic Style

Types of multiplier	Delay (ns)	Power (µw)	PDP (fj)	EDP (fj.ns)
Wallace tree multiplier	3.232	53.087	171.57	554.51
Array multiplier	3.126	47.316	147.90	462.33
Baugh wooley multiplier	3.446	47.567	163.91	564.83

Table 3: Compound Constant Delay Logic Style

Types of multiplier	Delay (ns)	Power (µw)	PDP (fj)	EDP (fj.ns)
Wallace tree multiplier	3.243	13.112	42.5	137.89
Array multiplier	3.204	13.868	44.43	142.35
Baugh wooley multiplier	3.452	13.823	47.70	164.66

5. CONCLUSION

Multipliers are the major portions in hardware consumption for filters. Carry skip adder is used to speed up the accumulation. Wallace tree multiplier reduces the delay by taking partial product reduction method. Wallace tree multiplier, array multiplier and Baugh wooley multiplier are implemented using static logic style, dynamic logic style and Compound constant delay. From the tabulated results it is clear that the Energy Delay product for a Wallace Tree Multiplier is 137.89 fj.ns, Array multiplier is 142.35 fj.ns and the Baugh Wooley Multiplier is 164.66 fj.ns. The simulation result shows that energy delay product of Wallace tree multiplier, array multiplier and Baugh wooley multiplier using compound constant delay logic style is better than static logic style and dynamic logic style.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

REFERENCES:

[1] M. Aguirre-Hernandez and M. Linares-Aranda, "CMOS full-adders for energy-efficient arithmetic applications," *IEEE Trans. Very Large ScaleIntegr. (VLSI) Syst*., vol. 19, no. 99, Apr. 2010 pp. 1–5.

[2] Douglas A.Pucknell and Eshraghian, *Basic VLSI Design*, 3rd ed. Reading,PHI,2010

[3] N.Goncalves and H.De Man, "NORA: A racefree dynamic cMOS technique for pipelined logic structures," *IEEE J.Solid –State Circuits*,vol.18, no.3, Jun.1983, pp261-266.

[4] A.K.Kumar,D.Somasundareswari,T.S.Pradeepa ,"Design of Low Power Multiplier with Energy Efficient Full adder using DPTAAL,"*Hindawi Publishing Corporation Very Large Scale Integr.(VLSI) Syst*., vol.2013, Feburary 2013 Article ID 157872, pages 9.

[5] C.Lee and E.Szeto, "Zipper CMOS," *IEEE Circuits Syst.Mag.,* vol .2, no. 3, May 1986,pp 10 -16.

[6] V.Navarro-Botello, J.A.Monitel-Nelson, and S.Nooshabadi,"Low Power Arthimetic Circuit in Feedthroughdynamic CMOS logic," in *Proc.IEEE Int. 49th Midw. Symp.Circuits Syst.,*Aug.2006, pp. 709-712.

[7] N.Wesye and D.Harris, *CMOS VLSI Design: A Circuits and Systems Perspective,* 4th ed. Reading, MA;Addison Wesley, Mar. 2010.

[8] L.Raja,B.M.Prabhu and K.Thanushkodi 'Design of Low power digital multiplier using Dual Threshold voltage Adder Module, "*International Conference on Communication and System Design 2011".*

[9] M.Ravindrakumar, G.ParameswaraRao, "Design and Implementation of 32 bit High Level Wallace Tree Multiplier," *International Journal of Technical Research and Applications*, vol. 1 Issue 4 (sept-oct 2013),pp.86-90.

[10] Rafati, S. Fakhraie and K.Smith, " A 16 bit barrel shifter implemented in data driven dynamic logic(D3L)," *IEEE Trans.Circuits Syst.I,Reg.Papers*,vol.53, no.10 , Oct 2006, pp. 2194-2202.

[11] M.Sinangil, N.Verma and A. Chandrasekaran, " A Reconfigurable 8T Ultra Dynamic Voltage Scalable(U-DVS) SRAM in 65 nm cMOS",*IEEE Journal in solid- State Circuits* vol. 44 Nov 2009, pp 3163- 3173.

[12] N. Verma and A. Chnadrakasn, ' A 65 nm 8T Sub-vt SRAM Employing Sense –Amplifier Redundancy," *IEEE International in Solid State Circuits Conference, 2007. ISSCC 2007.Digest of Technical Papers*., 11 -15 2007, pp 328 606.

[13] Zimmermann and W.Fichtner, " Low power logic styles: CMOS versus pass transistor logic," *IEEE Journal of Solid State Circuits* vol 32 July 1997, pp 1079-1090.