International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 05 Issue: 09 | Sept - 2021 ISSN: 2582-3930

COMPARISION OF DIFFERENT IMPLEMENTATIONS OF WALLACE TREE MULTIPLIER AND DADDA MULTIPLIER.

Sumanth A
M.Tech in VLSI Design and Embedded Systems
RV College of Engineering
Bangalore, India

Dr. Kiran V
Professor, Department of ECE
RV College of Engineering
Bangalore, India

***Abstract* - The Wallace Tree Multiplier (WTM) is one of the fastest multipliers used in many data-processing processors to perform fast arithmetic functions. WTM consumes a lot of area as compared to other multiplier implementations. This is because of high number of gates. WTM is still preferred as the multiplier due to its speed. WTM is a complex multiplier. WTM use deferent full and half adders to sum partial products in stages until two numbers are left. Different implementations such as Cadence structural implementation, CMOS gate implementation and Universal gate implementation is done and analyzed.**

Index Terms - Wallace tree multiplier, CMOS implementation, 8x8 multiplier, multiplier performance, CMOS multipliers, fast multiplier

I. INTRODUCTION

Multipliers are one of the basic and most important components of any digital system in present day era. in various applications such as DSP, video processing and Fourier transform multipliers play a very important role. The speed of a arithmetic unit is dependent on MAC and ALU in a system. Multipliers and adders play a very important role in deciding the performance and power consumption in any system. In present day technologies, when billions of components are being integrated into a single chip, low power and speed become a very important parameter.

There are different kind of multipliers used by designers in present day, namely – Array multiplier, Booth Multiplier, Binary multiplier, Dadda multiplier and Wallace tree multiplier. DADDA and Wallace Tree multiplier reduce the number of steps taken to arrive at the final product. Hence power and timing performance of the system is greatly improved. Wallace tree multiplier avoids carry propagation or all the carry is generated at the same time, hence there is no waiting since carry generation is the critical path.

Parallel prefix adders can be used in dadda multipliers to improve their performance. Dadda multipliers using Koge stone adders are found to be better performing than conventional Dadda Multiplier. Wallace also uses parallel prefix adder and compressors to improve the performance. Compressors are used for the purpose of accumulating and reduce multiple number of input signals into a comparatively small number of parallel signals. They are widely used in multipliers due to the presence of large number of partial products and their addition.

In this paper study of the different implementations of Wallace and Dadda multiplier using CMOS libraries of all the basic gates, and universal gate standard cell libraries are done. The power consumption and timing of each of these implementations are compared. Cadence genus tool is used for the synthesis tool is used for the netlist generation for different implementations. CMOS basic gate library contains AND, OR, XOR, XNOR, INVERTER, DFF cells. Universal gate library consists of NAND, NOR, INVERTER, DFF cells.

A. Wallace Tree Multiplier

In Wallace-tree multiplier, forward work is done from multiplier inputs, compressing the number of signals to be added at each stage (Wallace, 1960]. It can be viewed that an FullAdder is a 3 : 2 compressor or (3, 2) counter as it counts the number of 'l's on the inputs. Thus, for example, an input of '101' (two 'l's) results in an output '10' (2). A HalfAdder is a (2 , 2) counter. To form P5 6 sums must be added (Sum0, Sum14, Sum23, Sum32, Sum41. and Sum50) and 4 carries from the Product4 column. These 7 stages are added. The max delay through the CSA array is 6 adder delays. To this delay of the 4-bit (9 inputs) CPA (stage 7) must be added. There are twenty-six adders (six half adders) plus the four adders in the CPA. Fig 1 shows a 6 bit Wallace tree Multiplier.

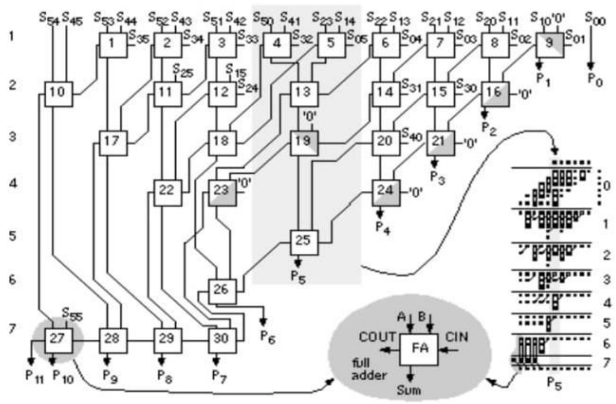

Fig. 1- 6 bit Wallace Tree Multiplier.

B. *Dadda Multiplier*

In a Dadda multiplier, multiplication is worked back from the final product. Every stage contains a max of two, three, four, six, nine, thirteen, ninteen, ... outputs (following stage is one and half times bigger-rounded down to below integer). Thus, for eg of 6 bit multiplier, 3 stages (with 3 adder delays--plus the delay of a 10-bit output CPA) is require for a 6-bit Dadda multiplier. There are nineteen adders (four half adders) in the CSA plus the ten adders (two half adders) in the CPA. A Dadda multiplier is usually having high speed and consumes less area compared to a Wallace-tree multiplier.

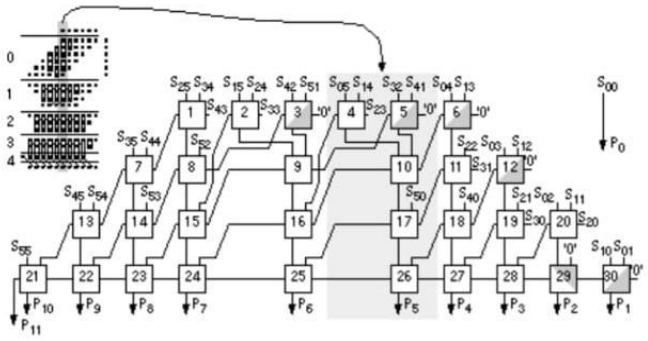

Fig. 2- 6 bit Dadda multiplier.

B. *CMOS Implementation*

While designing IC, the elementary and important element is the transistor. So, the transistor usually used is MOSFET. The formation of this transistor is done by specific procedure. But CMOS has more advantages than NMOS over unipolar and bipolar transistor. In CMOS, static power dissipation is very less and the output will give almost full swing. CMOS has high input impedance and low output impedance. CMOS consists of both NMOS and PMOS.

In CMOS logic circuits, either pull down network or pull network conducts once at a time so there exist no direct path between VDD and GND. Fig 3 shows a typical CMOS circuit. The outputs of the gate gives the Boolean function that is realized for implementation and this effects reduces transient response. Moreover, in traditional logic gates delay will be associated with the gates and by increasing the number of logic gates for complex Boolean expressions redundant logic increases.

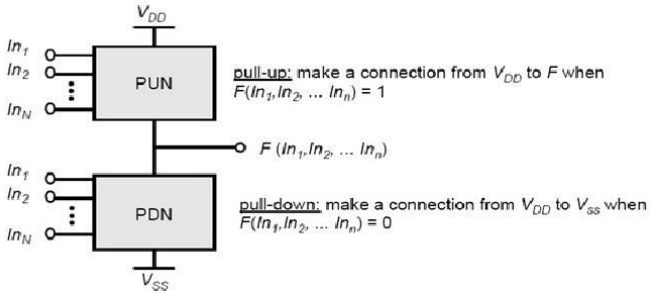

Fig. 3 Typical CMOS logic

II. IMPLEMENTATION

A. *Implementation of 8-bit Wallace Tree Multiplier*

Wallace Tree Multiplier was Implemented using basic CMOS gates AND, OR, XOR, XNOR, INVERTER, DFF cells. The Verilog Model was designed and implemented using genus synthesis tool in Cadence. The simulation was done using NClaunch tool in Cadence. The basic gates designed was 2X1 drive strength. The power report and number of cells was analysed. Fig. 4 shows the simulation results of 8 bit Wallace Tree Multiplication. Fig. 5 shows the generic synthesis of the same using Genus tool.

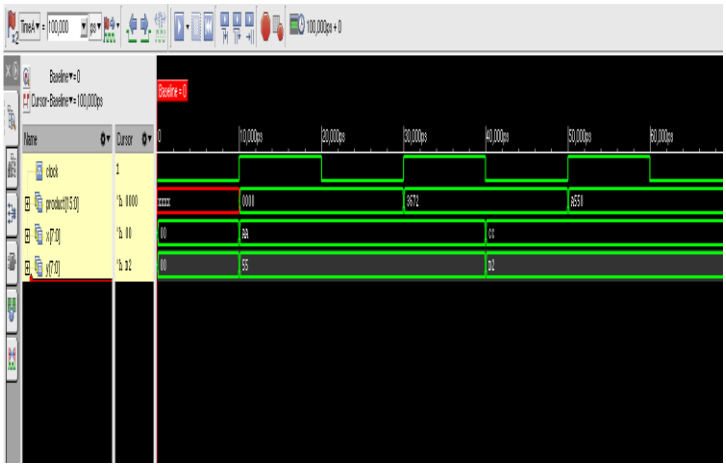

Fig. 4 Simulation on 8 bit WTM in nclaunch tool.

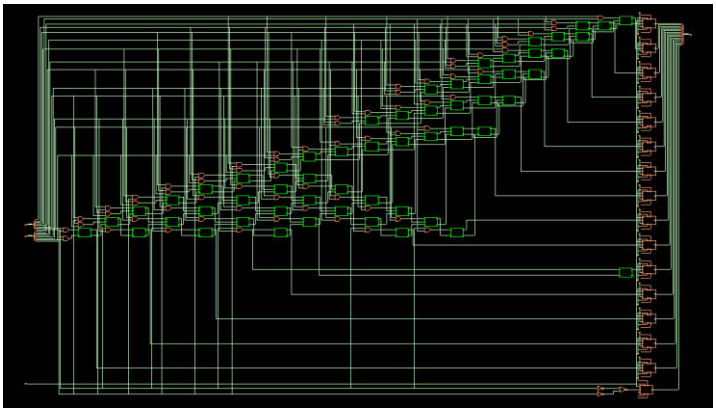

Fig. 5 Generic Synthesis of 8 bit WTM

Synthesis of the circuit was performed using basic gates library. Fig. 6 shows the generates schematic using basic gates.

Power report was generated using the report_power command from the synthesis tool.

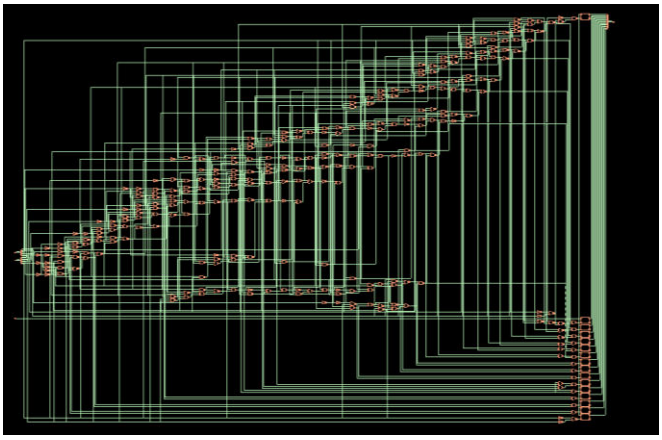

Fig. 6 Map Synthesis of 8 bit WTM using basic gates

Wallace Tree Multiplier was Implemented using Universal CMOS gates NAND, NOR and DFF cells. The Verilog Model was designed and implemented using genus synthesis tool in Cadence. The simulation was done using NClaunch tool in Cadence. The universal gates designed was 2X1 drive strength. The power report and number of cells was analysed. Fig. 7 shows the Universal gate synthesis of the same using Genus tool. Table I shows the comparision of Different parameters in 2 types of Implementations of Wallace Tree Multiplier.

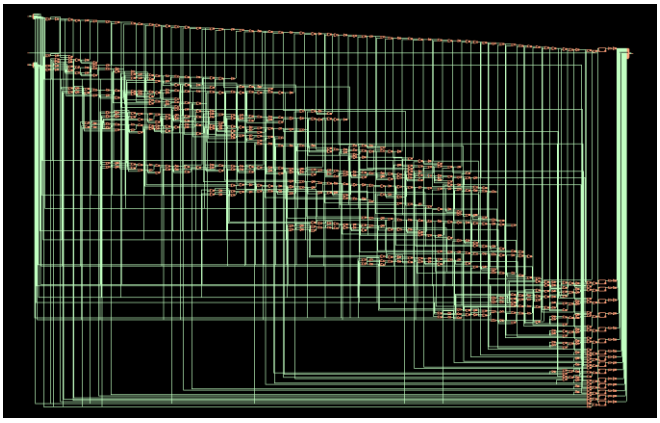

Fig. 7 Map Synthesis of 8 bit WTM using universal gates

TABLE I
PARAMETERS OF WALLACE TREE MULTIPLIER

Type of Implementation of Wallace Tree Multiplier	Basic CMOS Gates	Universal Gates
Number of cells in the design	331	723
Leakage Power (in nW)	15.67	19.42
Dynamic Power (in nW)	1897310	2011723
Total Power (in nW)	1897326.67	2011742.42

B. Implementation of 8-bit Dadda Multiplier

Dadda Multiplier was Implemented using basic CMOS gates AND, OR, XOR, XNOR, INVERTER, DFF cells. The Verilog Model was designed and implemented using genus synthesis tool in Cadence. The simulation was done using NClaunch tool in Cadence. The basic gates designed was 2X1 drive strength. The power report and number of cells was analysed. Fig. 8 shows the simulation results of 8 bit Dadda Multiplication. Fig. 9 shows the generic synthesis of the same using Genus tool.

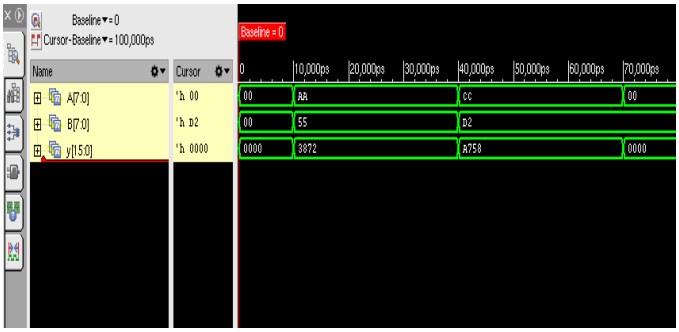

Fig. 8 Simulation on 8 bit Dadda Multiplier in nclaunch tool.

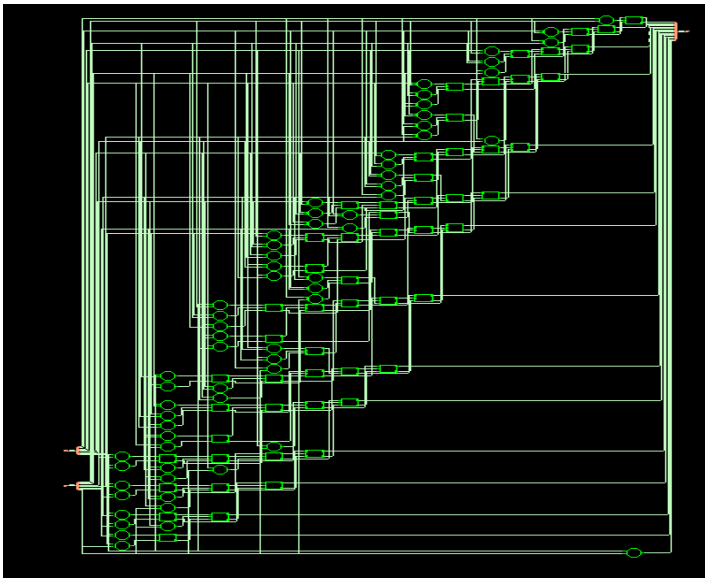

Fig. 9 Generic Synthesis of 8 bit Dadda Multiplier

Synthesis of the circuit was performed using basic gates library. Fig. 10 shows the generates schematic using basic gates.

Dadda Multiplier was Implemented using Universal CMOS gates NAND, NOR and DFF cells. The Verilog Model was designed and implemented using genus synthesis tool in Cadence. The simulation was done using NClaunch tool in Cadence. The universal gates designed was 2X1 drive strength. The power report and number of cells was analysed. Fig. 11 shows the Universal gate synthesis of the same using Genus tool. Table II shows the comparision of Different parameters in 2 types of Implementations of Dadda Multiplier.

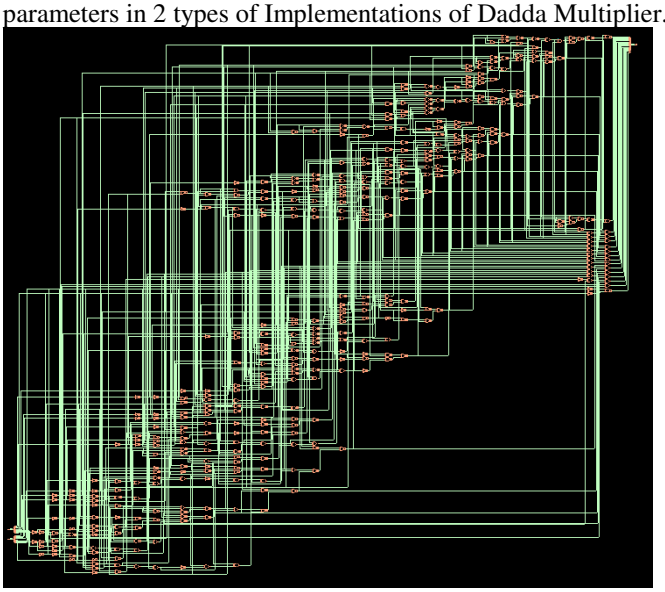

Fig. 10 Map Synthesis of 8 bit Dadda Multiplier using basic gates

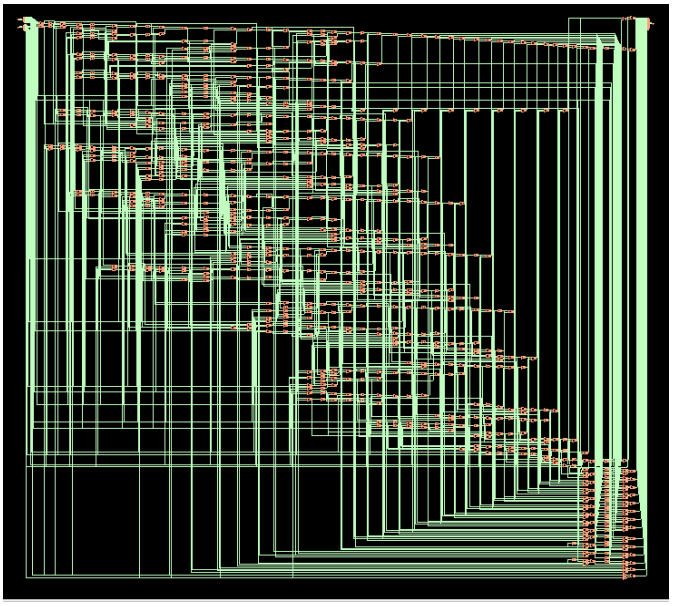

Fig. 11 Map Synthesis of 8 bit Dadda Multiplier using universal gates

TABLE II
PARAMETERS OF DADDA MULTIPLIER

Type of Implementation of Dadda Multiplier	Basic CMOS Gates	Universal Gates
Number of cells in the design	435	784
Leakage Power (in nW)	16.68	19.555
Dynamic Power (in nW)	2247405	2358777
Total Power (in nW)	2247421.68	2358796.55

III. CONCLUSION

Different types of implementations of Wallace Tree Multiplier and Dadda Multiplier is synthesized using genus tool and different parameters like number of cells and power requirements are tabulated. It is found that Universal Gate implementations of the multipliers require more power and area as compared to Basic gate Implementation.

REFERENCES

[1] Mukherjee, B., & Ghosal, A. (2019, March). Counter Based Low Power, Low Latency Wallace Tree Multiplier Using GDI Technique for On-chip Digital Filter Applications. In *2019 Devices for Integrated Circuit (DevIC)* (pp. 151-155). IEEE.

[2] devi Ykuntam, Y., Pavani, K., & Saladi, K. (2020, July). Design and analysis of High speed wallace tree multiplier using parallel prefix adders for VLSI circuit designs. In *2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT)* (pp. 1-6). IEEE.

[3] Gandhi, D. R., & Shah, N. N. (2013, March). Comparative analysis for hardware circuit architecture of Wallace tree multiplier. In *2013*

International Conference on Intelligent Systems and Signal Processing (ISSP) (pp. 1-6). IEEE.

[4] Basiri, M. M. A., Nayak, S. C., & Sk, N. M. (2014, February). Multiplication acceleration through quarter precision Wallace tree multiplier. In *2014 International Conference on Signal Processing and Integrated Networks (SPIN)* (pp. 502-505). IEEE.

[5] Chanda, S., Guha, K., Patra, S., Singh, L. M., Baishnab, K. L., & Paul, P. K. (2020, February). An Energy Efficient 32 Bit Approximate Dadda Multiplier. In *2020 IEEE Calcutta Conference (CALCON)* (pp. 162-165). IEEE.

[6] Bharathi, M., & Shirur, Y. J. M. (2019, November). Optimized Synthesis of Dadda Multiplier Using ParallelPrefix Adders. In *2019 International Conference on Smart Systems and Inventive Technology (ICSSIT)* (pp. 288-292). IEEE.

[7] Abraham, S., Kaur, S., & Singh, S. (2015, January). Study of various high speed multipliers. In *2015 International Conference on Computer Communication and Informatics (ICCCI)* (pp. 1-5). IEEE.

[8] Sebastian, A., Jose, F., Gopakumar, K., & Thiyagarajan, P. (2020, March). Design and Implementation of an Efficient Dadda Multiplier Using Novel Compressors and Fast Adder. In *2020 International Symposium on Devices, Circuits and Systems (ISDCS)* (pp. 1-4). IEEE.