



International Journal of Multidisciplinary Research in Science, Engineering and Technology

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)



Impact Factor: 8.206

Volume 8, Issue 10, October 2025



International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

Sequential and Combinational Multipliers Design and Comparing Constraints

Roopa R

Assistant Professor, Department of Computer Science (Data Science), Vemana Institute of Technology,
Bangalore, India

Email id: roopali.r9@gmail.com

Dr.Vasudeva G

Assistant Professor, Dept. of ECE, DSATM, Bangalore, India

Email id: devan.vasu921@gmail.com

ABSTRACT: For many years VLSI chip designers have been using Metal-oxide semiconductor field effect transistors(MOSFET).Designers have used MOSFET circuits in the designs because MOSFET use little power and they are cheaper to develop or fabricate. To satisfy the demands of making the chip smaller, the designers just simply shrink the dimensions to fit the circuit layouts. The shrinkage is also known as “Scaling”. Unfortunately, the continuous scaling of the circuit design will eventually cause problems in terms of area, timing and power. Signal processing is one of the most power hungry applications. Adders and Multipliers are the main building blocks for signal processing applications. Saving area, power and timing in adders and multipliers would reduce the area, power and timing significantly at the chip level

In this present investigation 3 multiplier structures are selected from existing multiplier types. The selected multipliers are Booth multiplier, Wallace tree multiplier and Dadda multiplier. Benchmarking for these 3 multipliers is done in terms of area, power and timing for 8 bits at 130nm technology. The HDL code for the selected multipliers is written in Verilog HDL and simulated in MODELSIM. To calculate the power and delay for these 3 multipliers we have used Design Compiler

KEYWORDS: MOSFET, MULTIPLIERS, VLSI, CSA, MODELSIM

I. INTRODUCTION

1.1 COMMON FEATURES OF MULTIPLIERS:

The architecture can be easily modified to allow dynamic specification of operand widths, i.e., successive operations of a given multiplier implementation could operate upon different word length[4].The increased complexity of various applications, demands not only faster multiplier chips but also smarter and efficient multiplying algorithms that can be implemented in the chips[8]. It is up to the need of the hour and the application on to which the multiplier is implemented and what tradeoffs need to be considered. Generally, the efficiency of the multipliers is classified based on the variation in speed, area and configuration. Due to rapidly growing system-on-chip industry, not only the faster units but also smaller area and less power has become a major concern for designing very large scale integration (VLSI) circuits[11]. Digital circuits make use of digital arithmetic's. Among various arithmetic operations, multiplication is one of the fundamental operation used and is being performed by an added. There are many ways to build a multiplier each providing trade-off between delays and other characteristics[15], such as area and energy dissipation The objective of a good multiplier and accumulator (MAC)[22] is to provide a physically compact, good speed and low power consuming chip. To save significant power consumption of a VLSI design[16], it is a good direction to reduce its dynamic power that is the major part of total power dissipation.



International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

II. RELATED WORK

The number and variety of products that include some form of digital signal processing has grown dramatically over the last years[22]. DSP has become a key component in many consumers, communications, medical, and industrial products. These products use a variety of hardware approaches[20] to implement DSP, ranging from the use of off-the-shelf microprocessors to field-programmable gate arrays (FPGAs) to custom integrated circuits (ICs). Programmable “DSP processors,” a class of microprocessors optimized for DSP[12], are a popular solution for several reasons. In comparison to fixed-function solutions, they have the advantage of potentially being reprogrammed in the field, allowing product upgrades or fixes. They are often more cost-effective (and less risky) than custom hardware, particularly for low-volume applications, where the development cost of custom ICs[13] may be prohibitive. And in comparison to other types of microprocessors, DSP processors often have an advantage in terms of speed, cost, and energy efficiency [1].

. There are many implementation media available for signal processing. These implementations vary in terms of programmability from fixed-functionality hardware like ASIC’s [9] to fully programmable like general-purpose processors. The emergence of the new architecture[5], which offers the same computational attributes as fixed-functionality architectures in a package that can be customized in the field, is driven by a need for real-time performance within the given operational parameters of a target system[14] and a need to adapt to changing data sets, computing conditions, and execution environments of DSP applications. In this paper we used three main ideas VHDL, architecture pipelining, and implementation of FPGAs. More details on FPGAs can be found in [2] This design has been simulated using the modelsim software then implemented on FPGA.

III. MOTIVATION

Recently, it has been found that area and time are not sufficient metrics for evaluating the system performance. Power consumption is another very important metric. Until recently, Power consumption was only a secondary concern in comparison to area and speed. However this thinking has begun to change in recent years and power is being given comparable weight to area and speed considerations. Many factors contribute to this new trend. Perhaps the most visible factor has been the phenomenal growth of portable electronics. Personal digital assistants, Laptops and cellular phones have enjoyed considerable success among consumers, and the market for these and other portable devices is on the rise. For these applications, the average power consumption has become critical design concern. For example, the power consumption of a portable multimedia terminal when implemented using off the shelf components not optimized for low-power operation is about 40W. With advanced nickel-metal –hydride battery technologies yielding around 65 watt-hour/kilogram, this terminal would require an unacceptable 6 kilograms of batteries for 10 hours of operation.[1] Even with state-of-the-art battery technologies such as lithium ion, which yield about 100 watt- hour/kilogram, the multimedia terminal would require about 4 kilogram of battery cells for 10 hours of operation. Therefore in the absence of low power design techniques, current and future portable devices will suffer from either a very heavy battery pack or a very short battery life. Even in the case of non-portable systems reduction in power consumption plays a very crucial role. All the above factors have motivated and felt the need to identify fast multiplier architectures for low- power applications, and set a stage for problems formulation.

IV MULTIPLIERS

In order to achieve high-speed, multiplication algorithms using parallel counters, such as the modified Booth algorithm[18] has been proposed, and some multipliers based on the algorithms have been implemented for practical use. This type of multiplier operates much faster than an array multiplier for longer operands because its computation time is proportional to the logarithm of the word length of operands[19].

V. ASIC DESIGN FLOW

Any IC other than a general purpose IC which contains the functionality of thousands of gates is usually called an ASIC. An ASIC is a digital or mixed signal circuit designed to meet specifications set by a specific project.

5.1 Design Compiler and Design Vision

The Synopsys Design compiler (DC) and Design Vision (DV) comprise a powerful suite of logic synthesis products, designed to provide an optimal gate-level synthesized netlist based on the design specifications, and timing constraints



International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

5.2 Primetime (Static Timing Analysis tool)

Primetime (PT) is the Synopsys sign-off quality, full chip, gate level static timing analysis tool. It allows comprehensive modeling capabilities often required by large designs. Primetime is faster compared to design compilers internal static timing analysis engine. It provides enhanced analysis capabilities to other Synopsys tools, this tool is based on TCL language, thus providing powerful features of that language to promote the analysis and debugging of the design.

5.3 SDF file generation

The SDF file may be generated for pre-layout or post-layout simulations. The post-layout SDF is generated from DC or PT, after back annotating the extracted RC delay values and parasitic capacitances, to DC or PT. The post-layout values represents the actual delays associated with the design.

The pre-layout numbers contain delay values that are based upon the wire-load models. It does not contain the clock tree. Therefore it is necessary to approximate the post-route clock trees delays while generating the pre-layout SDF. The post-layout design contains the clock tree information. Therefore all the steps that were needed to fix the clock latency, skew and clock transition time, during pre-layout phase are not required for post-layout SDF file generation.

5.4 Synopsys Formats

Here we introduce the various terminology and concepts used by Synopsys. Different Synopsys Formats are discussed along with design entry methods. The advantages and disadvantages of using **read** versus **analyze/elaborate** command are also discussed

Most Synopsys products support and share a common internal structure called the “db” format. The db files are the binary compiled forms representing the text data be it the RTL code, the mapped gate-level designs or the Synopsys library itself. The db files may also contain any constraints that have been applied to the design.

In addition all Synopsys tools understand the following formats of HDL, DC is capable of reading or writing any of the following formats

Verilog VHDL EDIF

For coding a design the 2 main HDL'S in use today are **Verilog** and **VHDL**.

Electronic Design Interchange Format (EDIF) is utilized for porting the gate level netlist from one tool to another.

5.5 Design Entry

Before synthesis the design must be entered in DC in the RTL format. DC provides the following 2 methods of design entry.

“read” command “analyze/elaborate” command

The **read** command is initially introduced by Synopsys which is then followed by **analyze/elaborate** commands. For design entry the later commands provide a fast and powerful method over the **read** command and are recommended for RTL design entry. The **analyze** and **elaborate** commands are two different commands. They allow designers initially to analyze the design for syntax errors and RTL translation before building the generic logic for the design. The generic logic are GTECH Components are part of the Synopsys generic technology independent library. They are unmapped representations of Boolean functions and serve all place holders for the technology dependent library.

The **analyze** command also stores the result of the translation in the specified design library which may be used later. A basic understanding of the concepts of Synopsys technology library and delay calculation is the key for successful synthesis. The popularity of the Synopsys library format is evident from the fact that most place and route tools provide a direct translation of the Synopsys libraries, with almost a one- one mapping between the timing models in Synopsys libraries and the place and route timing models. In order to optimize the design successfully, it is essential for designers to have a clear understanding of the delay calculation method used by DC along with the wire-load modeling and cell descriptions. Hence the intention of this section is to describe Synopsys technology library from the designer's perspective rather than discussing about the structural and functional syntax of the library.



International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

Classification: The Synopsys technology library can be classified in to two broad classes.

1. Logic library
2. Physical library.

Logic library: This library contains information about the Synthesis process and is used by DC for synthesis and optimization of the design. This information may include pin-to-pin timing, area, pin types and power along with other necessary data needed by DC. No physical information is present in the logic library. The logic library is a text file which is compiled using the library compiler (LC) to generate a binary format with “db” extension.

Wire load models : The **wire_load** group contains information that DC utilizes to estimate interconnect wiring delays during the pre layout phase of the design. Usually several models appropriate to different sizes of the logic are included in the technology library. These models define the **capacitance**, **resistance** and **area** factors. Also the **wire_load** group specifies slope and **fanout_length** for the logic under consideration.

The **capacitance**, **resistance** and **area** factors represent the wire resistance, the capacitance and area respectively per unit length of interconnect wire.

The **fanout_length** attribute specifies values for the length of the wire associated with the number of fanouts. This may also contain values for other parameters such as **average_capacitance**, **standard_deviation** and **number_of_nets**.

VI. REVIEW OF MULTIPLIERS

6.1 Booth's Multiplier

Booth's Algorithm is a powerful direct algorithm to perform signed number multiplication.

Booth's algorithm involves repeatedly adding one of two predetermined values A and S to a Product P, then performing a rightward arithmetic shift on P. Let **x** and **y** be the multiplicand and multiplier respectively. Let **x** and **y** represent the number of bits in **x** and **y**.

1) Determine the values of A, S and the initial value of P. All of these numbers should have a length equal to $x+y+1$.

A: Fill the most significant (leftmost) bits with the value of **x**. Fill the remaining $(y+1)$ bits with zeros.

S: Fill the most significant bits with the value of $(-x)$ in two's complement notation. Fill the remaining $(y+1)$ bits with zeros.

P: Fill the most significant **x** bits with zeros. To the right of this append the value of **y**. Fill the least significant (rightmost) bits with a zero

2 Determine the two least significant (rightmost) bits of P. If they are 01, find the value of $P+A$. Ignore any overflow. If they are 10, find the value of $P+S$. Ignore any overflow.

If they are 00 or 11, do nothing. Use P directly in the next step

3. Arithmetically shift the value obtained in the previous step by a single place to the right. Let P now equal to this new value.

6.2 Wallace Tree Multiplier

In 1964 C.S. Wallace introduced a scheme for the multiplication based on summing the partial product bits in parallel using a tree of carry save adders which became generally known as the Wallace tree. This method has a three step process is used to multiply two numbers.

Step 1: The bit products are formed

Step 2: The bit product matrix is reduced to a two row matrix by using carry save adders known as Wallace tree.

Step 3: The remaining two rows are summed using a fast carry –propagate adder to produce the product. Though the process seems to be complex it yields multipliers with delay proportional to the logarithm of the operand size **n**. The Wallace tree multiplier belongs to a family of multipliers called Column Compression Multipliers.

The principle in this family of multipliers is to achieve partial product accumulated by successively reducing the number of bits of information in each column using full adders or half adders. The full adder is known as (3:2) compressor because of its ability to add three bits from a single column of the partial product matrix and output two bits,



International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

one bit in the same column and one bit in the next column of the output matrix. The half adder is known as (2:2) compressor because of its ability to take two bits from a single column of the partial product matrix and output two bits, one bit in the next column of the output matrix. The Wallace tree consists of numerous levels of such column compressor structures until finally only two full width operands remain. These two operands can then be added using regular 2N-bit adders to obtain the product result. The difference between the Wallace tree multiplier. from column compression multiplier is that, in the Wallace tree every possible bit in every column is covered by the (3:2) or (2:2) Compressors respectively. Until finally the partial product matrix has a depth of only two. Thus the Wallace tree multiplier uses as much hardware as possible to compress the partial product matrix as quickly as possible into the final product.

The implementation of 8x8 bit Wallace tree multiplier is shown in figure below.

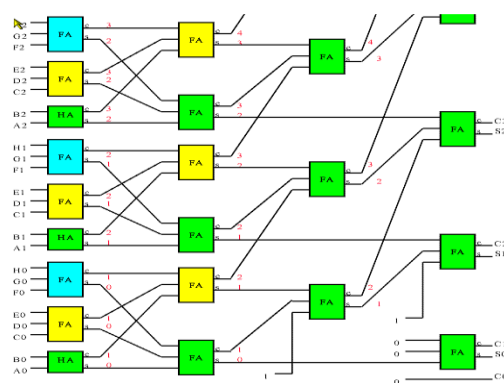


Figure6.1 8x8 bit Wallace tree multiplier

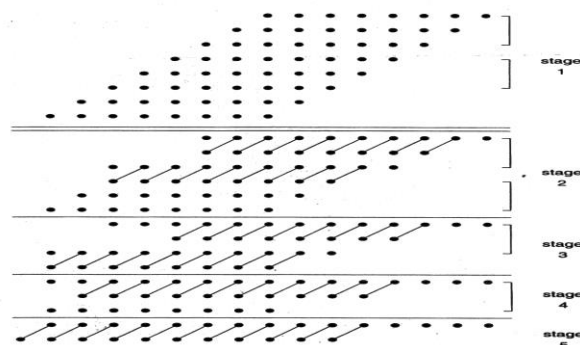


Figure 6.2 Operation of 8x8 – bit Wallace tree Multiplier

6.3 Dadda Multiplier:

Dadda refined Wallace's method by defining a counter placement strategy that required fewer counters in the partial product reduction stage at the cost of a larger carry propagate adder. Dadda has introduced a number of ways to compress the partial product bits using such a counter which later became known as Dadda's Counter. This process is shown for an 8 by 8 Dadda multiplier in figure 3. An input 8 by 8 matrix of dots (each dot represents a bit product) is shown as matrix 0. Columns having more than six dots are reduced by the use of half adders, each half adder takes in two dots and outputs one in the same column and one in the next more significant column and full adders, each full adder takes in three dots and outputs one in the same column and one in the next more significant column so that no column in matrix 1 will have more than six dots.



International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

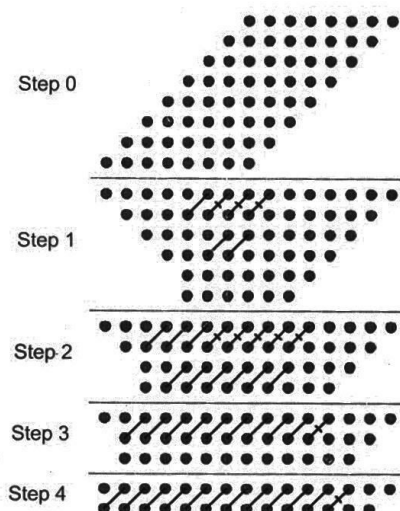


Figure 6.3 Operation of 8x8 – bit Dadda Multiplier

Half adders are shown by a crossed line in the succeeding matrix and full adders are shown by a line in the succeeding matrix. In each case the rightmost dot of the pair that are connected by a line is in the column from which the inputs were taken from the adder. In the succeeding steps reduction to matrix two with no more than four dots per column, matrix three with no more than three dots per column, and finally matrix four with no more than two dots per column is performed. The height of the matrices is determined by working back from the final two row matrix and limiting the height of the each matrix to the largest integer that is no more than 1.5 times the height of its successor. Each matrix is produced from its predecessor in one adder delay. Since the number of bits in the words to be multiplied, the delay of the matrix reduction process that reduces is proportional to $\log(n)$. Since the adder that reduces the final two row matrix can be implemented as a carry look ahead adder which also has logarithmic delay, the total delay for this multiplier is proportional to the logarithm of the word size proportional to the logarithm of the word size.

VII. RESULTS AND ANALYSIS

Comparison between 8x8-bit Dadda and Wallace tree multipliers: Wallace tree multiplier uses 38 full adders and 15 half adders. Dadda multiplier uses 35 full adders and 7 half adders. Wallace tree multiplier requires a carry-propagate adder of 10 bits wide. Dadda multiplier requires a carry propagate adder of 14 bits wide. The other disadvantage of Dadda multiplier is that it is less regular than the Wallace tree multiplier, making it more difficult to layout in VLSI design. Area wise Dadda multiplier consumes less area as compared to Wallace tree and Booth Multiplier. Power wise Booth multiplier consumes less power as compared to Wallace tree and Dadda multiplier.

	Tmin(ns)	Tmax(ns)	Fmin(ns)	Fmax(ns)
Booth Multiplier	6.9114	78.2886	144.688	12.7732
Wallace tree Multiplier	6.8720	78.128	145.5180	12.80
Dadda Multiplier	6.7260	78.2740	148.6767	12.76



International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

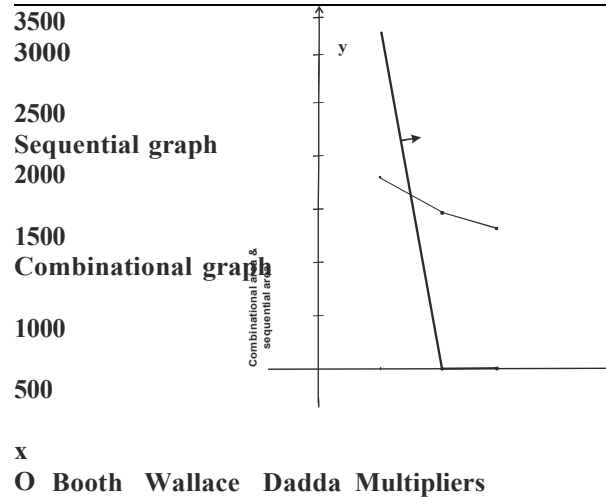


Figure 7.1 Graph of area of multipliers

	BELS	Flip Flops	ROM	Macros
Booth Multiplier	222	88	59	85
Wallace tree Multiplier	149	—	4	85
Dadda Multiplier	151	—	3	66

Table 7.2 FPGA synthesis report

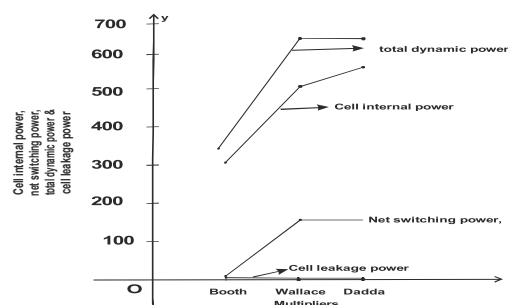


Figure 7.2 Graph of different powers



International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

Table 7.3 comparison of frequencies

	Tmin(ns)	Tmax(ns)	Fmin(ns)	Fmax(ns)
Booth Multiplier	6.911 4	78.288 6	144.6 88	12.7 732
Wallace tree Multiplier	6.872 0	78.128	145.5 180	12.8 0
Dadda Multiplier	6.726 0	78.274 0	148.6 767	12.7 6

All in $\square_w$	Booth	Wallace tree	Dadda
Cell internal power	303.1691	501.8940	562.0922
Net Switching power	21.3611	153.6577	153.7151
Total dynamic power	324.5302	655.5517	655.8073
Cell leakage power	5.6910	5.6263	5.6258

Table 7.3 comparison of power



International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

VIII. IMPLEMENTATION

8.1 FPGA Implementation

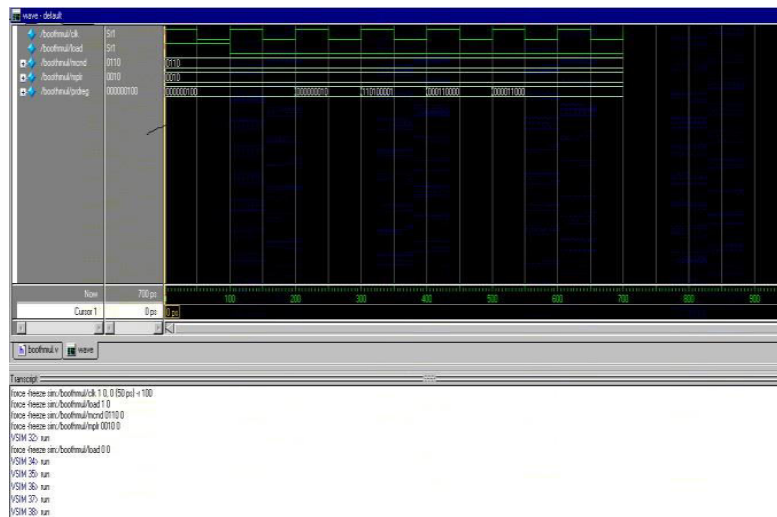


Figure 8.1 Test bench output for multipliers

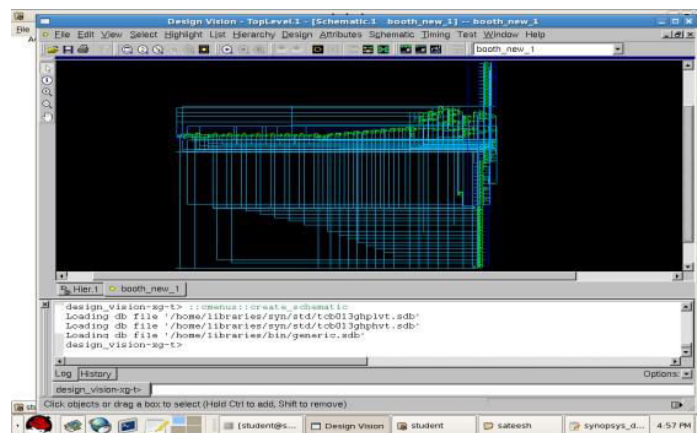


Figure 8.2 Schematic of Booth's multiplier using design compiler

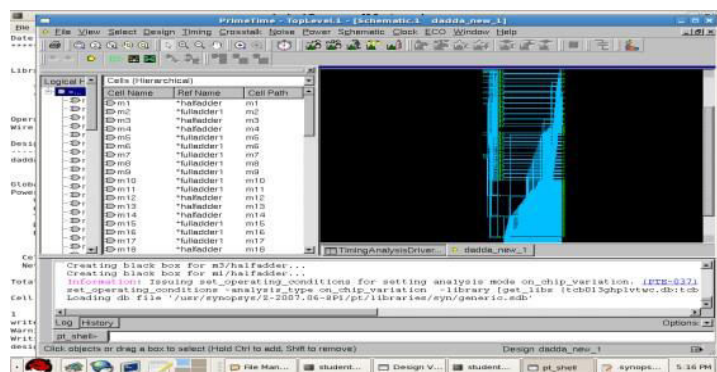


Figure 8.3 Schematic of Wallace tree multiplier using design compiler



International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

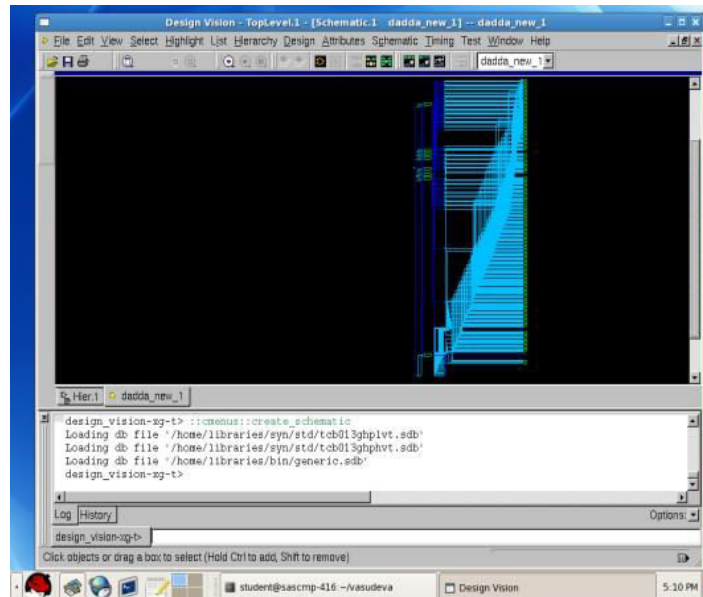


Figure 8.4 Schematic of Dadda's multiplier using Design compiler

After getting the schematic view, we have to give information to the tool about timing, power and area as a constraint. The netlist which we got at the time of synthesis has become the input for the tool. In the physical design flow, we have to do library preparation first which comes under the design setup process. The design setup consists the entire technology file named as technology file, library exchange format, advance library format, cell library file, physical library, design exchange format, top design format, physical design format, and table look up files. All these files contain technology and foundry dependent parameters which we are going to use to get a physically implemented chip. After completing the design setup we have to load the netlist to the library for further process like floor planning, power planning etc. Floor planning is a process of placing the input and output and power and ground pads.

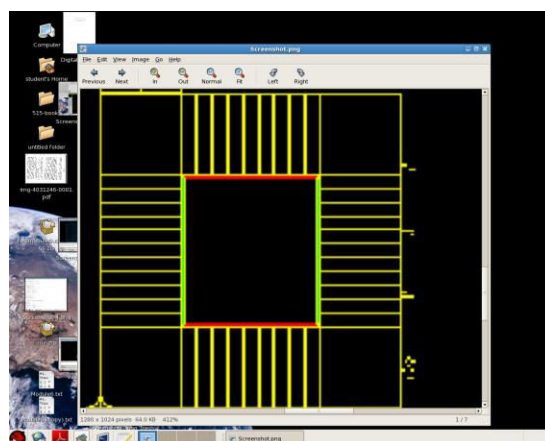


Figure 8.5 Floor planning output



International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

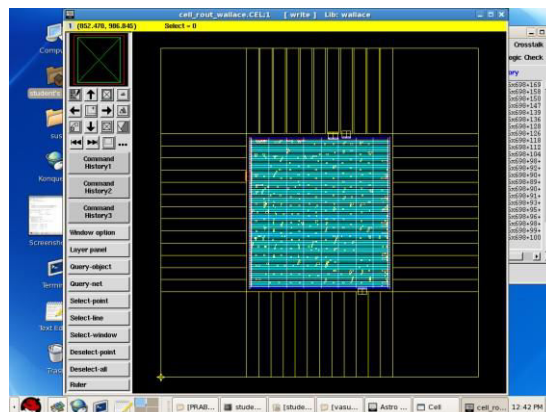


Figure 8.6 Placement output

After loading the TDF file we have to do floor planning of the design. For this process we have to set the aspect ratio of the core i.e. we have to do the power planning of the design by creating the power rings around the core region through which we can connect the straps and trunks for the core region. While doing power planning we have to choose the metal layer which has less RC value. At this stage we use the even metal layer for vertical and odd metal layer for horizontal. We have used metal 4 for the power and metal 3 for the ground. Fig 5.3(b) shows the floor planning and power planning of the design. After creating the rectangular rings around the core region connect the VDD and VSS pads to the power and ground rings, which is known as pre instance in the tool option.

The next step is load the Standard Delay Constraint (SDC) file.SDC file contains the timing parameters of the netlist which was taken at the time of static timing analysis through the primetime tool. At the time of synthesis we do not know which nets are to be used for routing, and the numerical value of the RC is also not known. We have to get the timing analysis output in such a way that the setup and hold time should not be violated

```
create_clock -period 10 [get_ports clk]
set_input_delay -clock clk 4 [get_ports "Instrn* Test* scan_en reset_n"] set_output_delay -clock clk 2 [all_outputs]
set_load 2 [get_ports *] set_clock_latency 1 clk set_propagated_clock clk
```

After loading the SDC file and getting the timing report, placement is to be done. In placement we have to place the standard cells horizontally in the core region. At this step, set some common option like optimization mode, placement constraints, location constraints etc then do pre placement optimization in which we have to set design cleanup, quick placement optimization of high fanout synthesis, ideal optimization and logic remapping. After pre placement do post-placement optimization. Now the stage of placement is over. The next step is the clock tree synthesis. Before going to the clock tree synthesis set the clock common option like the conditions worst, best and buffers, inverters. After setting the common option do the clock tree synthesis with some constraints like optimization, design level etc. After the process of clock tree synthesis we have to do post-place optimization for this set option like setup fixing, hold fixing, maximum capacitance, area recovery and logic remapping

Now the design at this point is DRC free with no violation in the timing. The next step is routing. This is the last but one stage of physical design flow. Before going to the routing we have to set the common option like global routing, Track assign, CTS net, detailed routing, library cells and design rule etc. After giving the common option we have to set the net group. When we set the all option we have to do global routing and after that detailed route. After completing all the routing process do the post route optimization with some constraints like routing phase, optimization effort, optimization target, optimization mode, optimization control, flow control etc. The routing output for the multipliers is as shown in fig 5.3 (c) After completing the routing check the Design rule check. If there is any DRC violation then we have to do search and repair. This option in the tool will detect the DRC violation and it will rectify.



International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

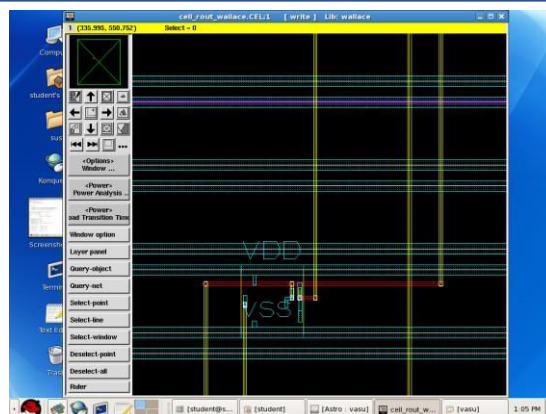


Figure 8.6 Final Routing output

SPEF and GDSII generation for the multiplier:

The following is the message shown by the ASTRO after the successful generation of Standard Parasitic Extraction Format (SPEF) file.

ATAENV: Operating Conditions:

Max: cb13fs120_tsmc_max Min: cb13fs120_tsmc_min EK: max metal layer: 4

EK: read geometry of cell mult EK: Release 1 cache blocks EK: drCoupMap initialized Processed 3368 top level nets.

Total run time: Elapsed = 2.00 seconds; CPU = 2.00 seconds. Peak memory usage = 187 MBytes

VR: Virtual Routing Completed SPEF Out succeeded.

The following is the message shown by the ASTRO after the successful generation of Graphical Data Stream Information Interchange (GDSII) file.

Outputting Cell route_mult.CEL

====> TOTAL CELLS OUTPUT: 1 <====

Outputting Contact \$\$VIA12A Outputting Contact \$\$VIA12B Outputting Contact \$\$VIA23 Outputting Contact \$\$VIA34 Outputting Contact \$\$VIA12f Outputting Contact \$\$VIA23f

Outputting Contact \$\$VIA23f_480_480_1_19 Outputting Contact \$\$VIA34f_940_940_5_10 Outputting Contact \$\$VIA34f_940_940_5_5 Stream out completed successfully!

Power Analysis, Estimation and Optimization:

As technology advances, the metal width is decreasing with the length increasing, making the resistance along the power line increase substantially. The voltage (IR) drop becomes a serious problem in modern VLSI design. To make the integration of the P/G analysis with floor planning feasible, we need a very efficient, accurate analysis method considering the issues like IR drop, electro migration, switching, short circuit and static power.

The power architecture used for the multiplier is the traditional grid architecture. A number of low power architectures are proposed for the efficient distribution of power through out the chip. One of such architectures is discussed below.

DEPOGIT: Dense Power-Ground Interconnect Architecture for Physical Design Integrity: In recent deep submicron VLSI design, signal integrity (SI) and power-ground integrity (PGI) have become very important to design in a short time. Most engineers in the process, chip-design, and EDA areas are acutely aware of a tough challenges emerging because of process variability and physical integrity issues. Process variability is not only a fabrication problem, but also a serious design issue. Similarly, physical integrity problems are not only design and EDA issues, but also process-related architecture problems. As a solution, DEPOGIT is a new dense power-ground interconnect architecture that realizes more robust physical design integrity. The interconnect architecture basically consists of adjoining power and ground lines. This architecture is a method of running both the power and ground wires adjacent to the signal wires. This provides not only the general shielding effect but also explicit decoupling capacitance (decap) by means of the wires. Using this architecture also guarantees regularity, thus reducing manufacturing variations in interconnects. Using this architecture,



International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

1. High quality decap for a small chip areas can be obtained.
2. The resistive IR drop can be less than 20% of that of a conventional power grid.
3. Transient peak noise can be reduced by about 80%.
4. The inductive crosstalk effect of the signal wire can be greatly reduced.

The report of the power analysis is given below. This report gives the information about the following points.

- ❖ Switching and static power
- ❖ Internal & short circuit power

The power report is follows.

Calculated power values for macro cell instances:

IO net switching power = 1.34369 mW Static cell-inst power calculation:

total switching power = 0.930607 mW (1.12573 %) total short-circuit power = 80.4922 mW (97.3693 %) total internal power = 1.16404 mW (1.40811 %) total leakage power = 0.0800637 mW (0.096851 %) total power = 82.6669 mW

Freeing all reference libraries Power calculation succeeded.

Total static cell-inst power and current:

total power and current at power net VDD = 82.6669 mW, 76.5435 mA total power = 82.6669 mW

Power analysis succeeded.

Power Analysis Memory: 271.605 MB CPU: 5 seconds Elapse: 6 seconds

IX. CONCLUSION AND SCOPE FOR FUTURE WORK

The results of the research work carried out are • Reviewed the existing high speed serial and parallel multipliers available, identified the specification requirements for the multipliers. • Modeled the multipliers using HDL and verified the functionality using test vectors. • Implemented the design on FPGA and verified its functionality and identified the hardware requirements. • Carried out ASIC design on the synthesized net list by appropriately providing the constraints based on the first cut information obtained from FPGA synthesis. • Compared the performance of multiplier design and optimized the design for area, speed and power.

REFERENCES

1. Keshab K.Parhi, "VLSI DIGITAL SIGNAL PROCESSING SYSTEMS", Design and implementation John Wiley and sons (ASIA) pte Hd, 1999.
2. Neil H.E.Weste, David Harris, Ayan Banerjee, "CMOS VLSI DESIGN", A circuits and systems perspective. Pearson Education, Third edition 2017, pp345-356.
3. Robert F.Shaw, "Arithmetic operations in a binary computer", Review of scientific instruments; vol 21, pp 687-693, 1950.
4. O.L.Mac Sorley,"High-Speed Arithmetic in Binary Computers", Proceedings of the IRE, Vol 49, pp.67-91, 1961.
5. Bruce Gilchrist, J.H. Pomerene and S.Y.Wong, "Fast Carry logic for Digital Computers", IRE Transactions on Electronic Computers, Vol.EC-4, PP.133-136, 1955.
6. R. De Mori, "Suggestions for an TC Fast Parallel Multiplier", Electronics letters, Vol.5, pp 50-51, 1965.
7. Andrew D.Booth, "A Signed Binary Multiplication Technique", Quarterly Journal of Mechanics and Applied Mathematics, Vol .4, pp.236-240, 2012.
8. Charles R. Baugh and Bruce .A.Wooley, "A Two's Complement Parallel Array Multiplication Algorithm", IEEE Transactions on Computers, Vol. C-22, pp.1045-47, 2018.
9. Behrooz Parhami, Computer Arithmetic: Algorithms and Hardware Designs, Newyork: Oxford University press, 2000.
10. Thomas K.Callaway and Earl E. Swatzlander,JR., "Optimizing Multipliers for WSI", Proceedings of the 1993 International Conference on Wafer Scale Integration, pp.85-94,2021.
11. C.S.Wallace," A suggestion for a Fast Multiplier", IEE Transactions on Electronic Computers, Vol.EC-13, pp.14-17, 1964.
12. Luigi Dadda," Some Schemes for Parallel Multipliers", Alta Frequenza, Vol.34, pp.349-356, August 1965.
13. K'Adrea C.Bickerstaff,Michael J.Schulte,and Earl E. Swartz Lander, Jr., "Reduced Area Multipliers", Proceedings of the 1993 International Conference on Application Specific Array Processors, pp.478-489,2023.



International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

14. Advanced Asic Chip Synthesis by Himanshu Bhatnagar Second Edition pp-183-256, 2002.
15. G. Lakshmi Narayanan and B. Venkataramani, "Optimization Techniques for FPGA-Based Wave Pipelined DSP Blocks", IEEE Trans. Very Large Scale Integr. (VLSI) Syst., **13**, No 7, pp 783-792, July 2023.
16. L.Benini, G.D. Micheli, A. Macii, E. Macii, M. Poncino, and R.Scarsi, "Glitching Power Minimization by Selective Gate Freezing", IEEE Trans. Very Large Scale Integr. (VLSI) Syst., **8**, No. 3, pp. 287-297, June 2020.
17. S.Henzler, G. Georgakos, J. Berthold, and D. Schmitt-Landsiedel, "Fastpower-Efficientcircuit-Block Switch-off Scheme", Electron. Lett., **40**, No. 2, pp. 103-104, Jan. 2024.
18. H. Lee, "A Power-Aware Scalable Pipelined Booth Multiplier", In Proc. IEEE Int. SOC Conf., 2024, pp. 123-126.
19. J. Choi, J. Jeon, and K. Choi, "Power Minimization of Functional units by Partially Guarded Computation", In Proc. IEEE Int. Symp. Low Power Electron. Des., 2020, pp.131-136.
20. Vasudeva, G., Uma, B. V. 22nm FINFET based high gain wide band differential amplifier. Internat. J. Circuits Sys. Sig. Proc., 2021:55–62.
21. Vasudeva, G., Uma, B. V. Low voltage low power and high speed OPAMP design using high-K FinFET device. WSEAS Trans. Circuits Sys., 2021;20:80–87.



INTERNATIONAL
STANDARD
SERIAL
NUMBER
INDIA



INTERNATIONAL JOURNAL OF MULTIDISCIPLINARY RESEARCH IN SCIENCE, ENGINEERING AND TECHNOLOGY

| Mobile No: +91-6381907438 | Whatsapp: +91-6381907438 | ijmrset@gmail.com |

www.ijmrset.com