Study and Analysis of Different Parameters and Power Consumption of Various Multipliers in FIR Filter using VHDL

Sushant Shekhar[1], Ghanshyam Jangid[2]

[1]Master of Technology (Scholar), Suresh Gyan Vihar University, Jaipur, India

[2]Assistant professor, Suresh Gyan Vihar University, Jaipur, India

Abstract: ***Power consumption and speed are the two main performance criterion of any circuit in VLSI system design. A multiplier and adder play an important role in VLSI technology. A circuit performance is usually explained by the performance of adder and multiplier. In digital circuit serial and parallel multiplier are used for the operation. As a result, multipliers with optimized area & speed have been designed with fully parallel algorithms. Low-power design led directly to the portable device uptime. Therefore, low power multiplier design has been in an important part of the low power VLSI system design.***

Keywords: Power consumption, speed, Radix-2 multiplier, Radix-4 multiplier, FIR filter, FFT algorithm.

1. Introduction

As we know that performance and area are the two major design factors, power consumption is a field of great concern in today's VLSI system design. The performance of a system is normally found by the power consumption of the multiplier as multiplier is commonly the slowest element in the system. It is commonly the most area taking. Therefore, the optimization of speed and size of the multiplier is an important design issue. Thus, an important part to play in low power VLSI system design is low power multiplier design. A multiplier is one of the most significant hardware blocks in nearly all digital and high performance systems such as FIR filters, the digital signal processors and the microprocessors. The project proposed high-speed multiplier using an efficient implementation of the shift and add method (serial multiplier), Radix_2, Radix_4 modified Booth multiplier algorithm. With the help of this project, we evaluate the working of these three multipliers by implementing each of them separately in a finite impulse response filter (FIR). Radix_2 and Radix_4 modified Booth multiplier are called parallel multipliers.

Parallel multipliers use lesser adders and lesser iterative steps for its computation. Due to this, they take up less space compared to serial multipliers. In comparing the power utilization of all multipliers in our project, we found that the serial multiplier consume more power. So power is an important factor necessary to be considered while designing any VLSI system. Therefore parallel multipliers like booth multipliers should be preferred to serial multipliers. The preferred choice in designing different circuits is booth multiplier, due to its low power utilization feature.

Here we first designed three different types of multipliers using shift and add method, radix_ 2 and radix_ 4 modified booth multiplier coding. We implemented different types of adders like 16- bit full adder in designing those multipliers. Then we created a 4 tap delay FIR filter and in position of the multiplication and additions we incorporated the mechanism of various multipliers and adders. Then we observed the functioning of different multipliers by comparing the power utilization by each of them. The results of our project help us in choosing a better alternative among serial and parallel multiplier in designing different systems. So by observing the functioning of different multipliers, we determined the better multiplier in terms of lesser area and less power consumption.

2. Research Approach

The basic motive of our project is to research and develop a fast, efficient and low power multiplier. The essential building block of multiplier is an adder circuit. So we put our focus on adder. We studied different delay time and the area consumed by adders and then developed appropriate relationship between them and discovered the time and complexity of the area by considering the adders. We generate a factor delay product area that helps us to correctly understand the area and delay trade off perfectly under appropriate circumstances. Thus the best adder can be chosen.

The next step is to focus on multipliers. We study the multipliers programming, verification waveform and then finally calculate the number of months and the power consumption required for LUT circuit. Knowing it all, we also calculate the delay of different multipliers. This helps us determine the best multiplier. Radix-4 booth multiplier is excellent in less power consumption and proper in all respect. Our future work will be to optimize the power consumption of different multipliers and thus reduce the number of doors and the use of the area they occupy. The flow diagram shown below explains the complete proposed work.

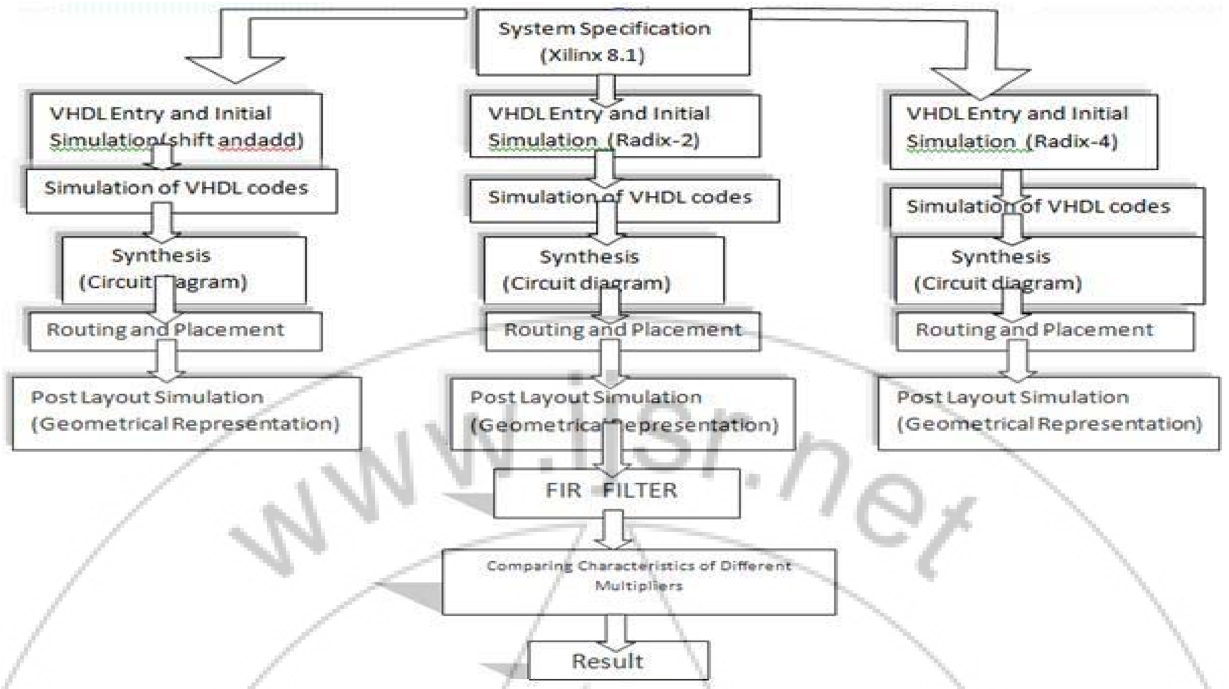

System Specification (Xilinx 8.1)
VHDL Entry and Initial Simulation(shift andadd)
VHDL Entry and Initial Simulation (Radix-2)
VHDL Entry and Initial Simulation (Radix-4)
Simulation of VHDL codes
Synthesis (Circuit diagram)
Routing and Placement
Post Layout Simulation (Geometrical Representation)
FIR FILTER
Comparing Characteristics of Different Multipliers
Result

Figure 1: Flow diagram

3. Simulation Tools.

The two types of simulation tool used in our project work are:
- Xilinx 9.1
- Xilinx Power Estimator (XPE)

3.1. Xilinx 9.1

We use this tool (VHDL language) and simulate various multiplier circuits using VHDL coding. VHDL (Very high speed integrated circuit hardware describe the characteristics of the language) is an electronic design automation to describe digital and mixed-signal systems such as field programmable gate arrays and thus integrate a hardware for the description language. In electronics, hardware description language or HDL is a type of computer language. In any language like formal specification language description, design or modeling language, the most common is the digital logic. HDL has a structure to handle parallelism. VHDL is powerfully typed and isn't case sensitive. Like concurrent programming language, HDL syntax and semantics include a clear sign to express accuracy.

The main advantage of VHDL for designing any system is that it allows the required system's behavior to be described (modeled) and verified (analog) before synthesis tool design it into real hardware (gates and wires). The actual computing languages such as BASIC, C and assembly code run one after the other, execute an instruction at a time but VHDL is a data-flow language. The project completed with the help of VHDL is versatile. Once created, a computing module can be used in many other projects. The code written using VHDL language is portable.

3.2. Xilinx Power Estimator(XPE)

The Xilinx Power Estimator (XPE) spreadsheet is a power estimation tool typically used in the initial design of a project and the early implementation phase. XPE assessment frameworks help to select device and equipment, and help you to choose the right power and thermal management components required for your application. When RTL design is described as incomplete, Xilinx power estimator can be used in early phase in the design cycle as a pre-implementation tool. After the implementation, Xilinx Power Analyzer (XPA) tool (available in ISE Design Suite software) can be used for more accurate estimates and power analysis.

XPE is a spreadsheet, so all the features of Microsoft Excel are writable (unprotected) in the spreadsheet section. XPE can perform quick power estimation for your design. For an accurate XPE estimation, we have to estimate the required amount of device for our project. If you do not know how to make this estimate it may be the best to run examples for designing ISE and we have to estimate the Map Report file, which contains resource usage information.

4. Multipliers

The three different types of multiplier discussed in this paper:

i) Shift and add multiplier
ii) Radix-2 multiplier
iii) Radix -4 multiplier

4.1. Shift and Add Multiplier

Binary multiplier is used in electronic hardware device or a computer or other digital electronic products and can perform quick multiplication of two numbers in binary

depiction. It is created using binary adders.
The block diagram of shift and add multiplier shown in Figure 2.

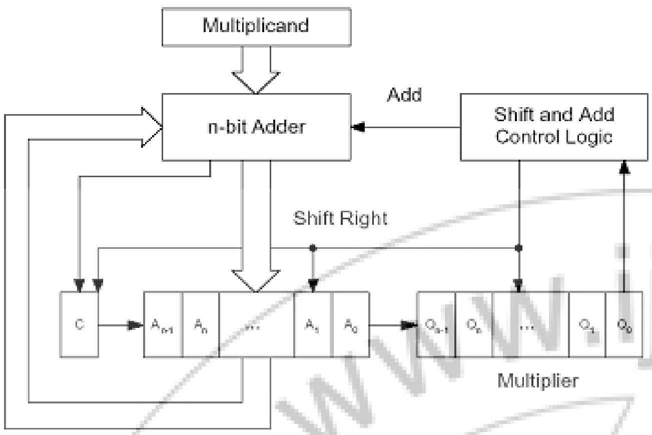


Figure 2: Shift &add multiplier block diagram

The rules for binary multiplication can be declared as follows:

- If the number of bits of the multiplier is 1, multiplicand is to be simply copied and is represented as the product.
- If the multiplier bit is 0 the product is also 0.

We required the following four things for designing a shift and add multiplier circuit:

- It should be able to transfer the left part of the product.
- It should be able to identify the bit is 0 or 1.
- It should check the sign bit. If they are identical, the product will be a positive sign; the sign bit is reversed if the product will be negative. Storing sign bit of the product of the above standard should be displayed together with the product.
- It should be able to add all of the parts of the product to give the product as the sum of some products.

4.2 Radix-2 Multiplier

Radix 2 multiplier architecture is shown in the Figure 3. This block diagram shows two numbers that are being multiplied by four digits each. These numbers are denoted as Q and P. Every circle shown in the figure corresponds to a radix cell. Every radix cell has four digits as input and two digits as output. The corresponding cells are used to supply input digits in this multiplier. The latches for pipelining are demonstrated by the dots. Each dot corresponds to four latches.

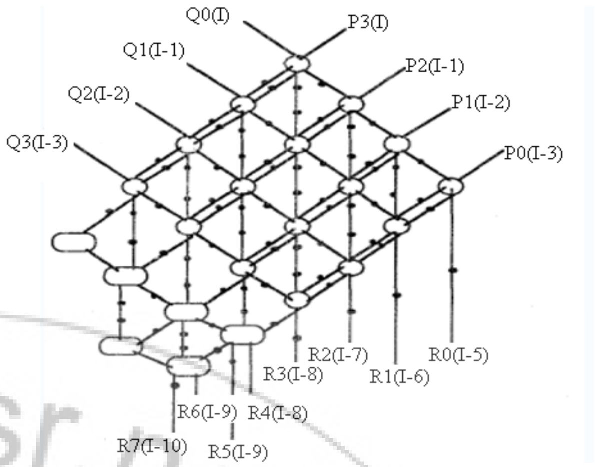


Figure 3: Architecture of a radix 2 multiplier

Booth algorithm illustrates the procedure for multiplying binary integers in signed –2's complement form. We will show the booth algorithm by the subsequent example:

Example: - Multiplying 2 and -4 in decimal and binary form.

In decimal form: $(2)_{10}$ x $(-4)_{10}$
In binary form: $(0010)_2$ x $(1100)_2$

Step 1: Making the Booth table

From the two numerals, pick the numeral with the smallest variation between a series of following numerals, an create it as multiplier, i.e., 0110 -- there will be no change from 0 to 0, there will be one change from 0 to 1, there will be another change from 1 to 0, so there will be two changes on this one 0100 -- there will be no change from 1 to 1, there will be one change from 1 to 0, there will be no change 0 to 0, therefore there will be only one transform on this one.

Thus, multiplication of 2 x (– 4), where (2)10 & (0010)2 is the multiplicand and (- 4)10 & (1100)2 is the multiplier.

- Let M = 1100 (multiplier)

Let N = 0010 (multiplicand)

Take the 2's complement of N and call it –N

–N = 1110

- Load the M value in the table.
- Load 0 for M-1 value and it should be the preceding first least significant bit of M
- Load 0 in P and Q rows which will have the product of M and N at the end of operation.
- As we are multiplying four bits numbers so four rows for each cycle is created.

.

Step 2: Booth Algorithm

Booth algorithm needs inspection of the multiplier bits, and then shifting of the partial product. Before shifting the multiplicand to the partial product they can be added or subtracted from the partial product, or can be changed according to the following rules:

Look at the first least significant bits (LSB) of the multiplier "M", and the previous least significant bits of the multiplier "M - 1".

- 0 0 is shift only
- 1 1 is shift only
- 0 1 is Add N to P, and then shift
- 1 0 is Subtract N from P, and then shift or add (-N) to P and shift.
- Take arithmetic right shift of P & Q together, it preserves the sign bit 2's complement. Thus, a positive number remains positive, negative remains negative.
- In order to prevent us from using two registers of M values, shift M right to the circular shift.

After finishing four cycles, we get the answer in the last row of P and Q which is $(11111000)_2$.

4.3 Radix -4 Multiplier

A solution to improve the high-speed multiplier parallelism, which helps reduce the number of stages of the calculation result, is to boost parallelism. The original version of the Booth multiplier (Radix - 2) has two drawbacks:

- The number of add / subtract operations became uneven and hence became inopportune while designing Parallel multipliers.
- The Algorithm becomes incompetent when there are isolated 1s.

Booth algorithm scanning algorithm string of three bits is given below:

- Extend the sign bit by a position to ensure that n is even.
- Affix a 0 to the right of the LSB of the multiplier.
- Depending on the value of each vector, each part of the product will be 0, +n, -n, +2n or – 2n shown in table 1.
- For product generator, multiply by zero means the multiplicand is multiplied by "0".Multiply by "1" means the product still remains the same as the multiplicand value. Multiply by "-1" means that the product is the two's complement form of the number. Multiply by "-2" is to shift left one bit the two's complement of the multiplicand value and multiply by "2" means just shift left the multiplicand by one place.

Table 1: Radix 4 multiplier

M(i)	*M(i+1)*	*M(i+2)*	*N*
0	0	0	+0
0	0	1	+n
0	1	0	+n
0	1	1	+2n
1	0	0	-2n
1	0	1	-n
1	1	0	-n
1	1	1	+0

5. Filters

A very important part of the digital signal processing is a digital filter. The two uses of filters are parting and the signal reconstruction. When the interference signal is infected or contains noise or other signals, it is necessary to split the signal. Let us imagine a device used to measure the baby's heart (EKG) electrical activity when in the womb. The original signal is possible to be ruined by the mother's breathing and heartbeat. A filter can be used to split these signals so that they can be analyzed separately.

Each filter includes responses like linear, step and frequency response. Each response contains complete information about the filter, but in another format. The other two can be directly calculated if they are preset and if one of three is precise. The easiest way to execute a digital filter is by convolving the input signal with the digital filter's impulse response. All possible liner filters can be made in this manner. When the impulse response is direct away, filters designers give it a special name: the filter kernel. There is one more way to make digital filters, called recursion. Implementing a filter by convolution requires each sample in the output to be calculated by weighting the samples in the input, and adding them together. An expansion of this is recursive filters, using previously calculated values from the output and moreover points from the input. Recursive filters are defined by a set of recursion coefficients as an alternative of using a filter kernel. The impulse response of a recursive filter can be found out by simply feeding in the impulse and check what comes out as output. The impulse response of recursive filters are composed of sinusoids that exponentially decompose in amplitude. This makes their impulse response considerably large. Recursive filters are also called Infinite impulse response or IIR filters, due to these personality. Filters carried out by convolution are called Finite impulse response or FIR filters.

5.1 FIR Filters

The digital filter can be divided into two categories: finite impulse response (FIR) filters and infinite impulse response (IIR) filter. Although FIR filters in general have higher taps than the IIR filter taps to obtain similar frequency personality, FIR filters are usually used because they have a linear phase characteristic to ensure stability and easy to implement with multipliers, adders and delay elements. The number of taps of the digital filter is changed depending on the application. In commercial filter chip having a predetermined number of taps, the zero coefficients can be loaded into an unused register tap and unnecessary calculations can be performed. To alleviate this problem, FIR filter taps of the chip that provides a variable length has been widely used in much application. This paper presents two special features called data multiplexing structure and recurrent factor scheme to provide efficient variable length. Since the proposed architecture requires only a few multiplexers, registers, and a feedback loop, the number of gates can be reduced by more than 20% than the conventional chip.

6. Experiments and Results

After analyzing all the three multipliers by booth algorithm and comparing their characteristics in terms of multiplication speed, no of computations required, no of hardware, we conclude that parallel multipliers are better than series multipliers. This is shown in table 2 below. In this project these multipliers implement with FIR filters to compare the speed, power consumption, computations and hardware requirement of the system.

The coding of all the multipliers is done separately in VHDL & simulated to get the accurate waveforms as output. Then we implement these multipliers separately with FIR filters using computation techniques like FFT and DFT. This coding is also written in VHDL language & simulated it to get the resistor transfer logic (RTL) circuit of each system. Also get the number of look up table (LUT), exact number of input, output and no of slices requirement etc for the system. Power consumption of each multiplier is calculated by Xilinx Power Estimator (XPE).

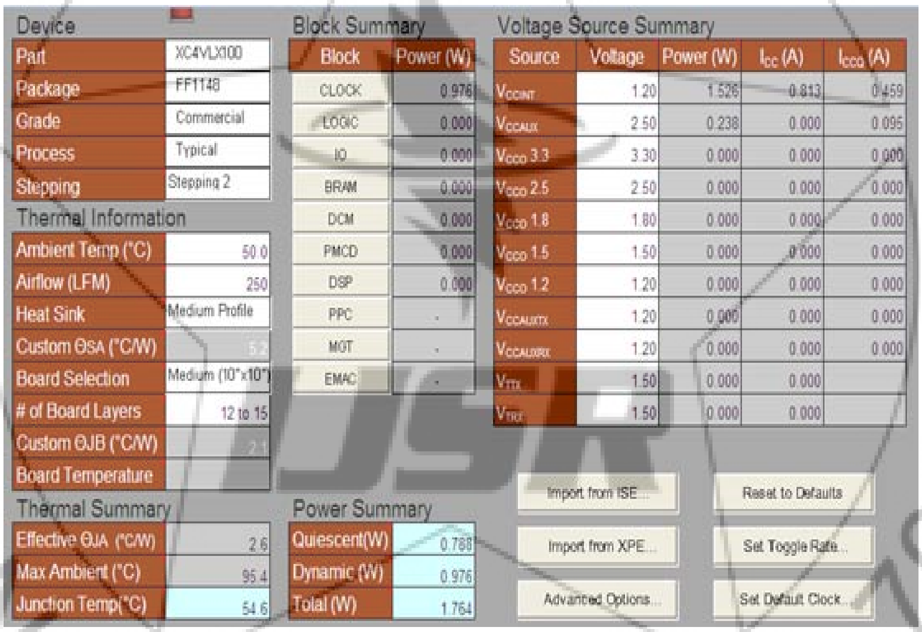
6.1 Results

Some results are obtained by experimental work and these results are shown in table 2.This table shows the comparison of different parameters of shift and add multiplier, radix-2 multiplier and radix-4 multiplier.

Table 2: Comparison of different multipliers

S. No.	*Parameters*	*Shift and Add Multiplier*	*Radix-2 Multiplier*	*Radix-4 Multiplier*
1.	Number of Slices	81	77	97
2.	Number of 4- input LUTs	140	141	171
3.	Number of bonded input	16	16	16
4.	Number of bonded output	16	16	17
5.	Logic Power	1.764	1.133	0.793

A result of power calculation is shown as images in the following figures 4, 5, 6 which are obtained by XPE analysis.



Device

Part	XC4VLX100
Package	FF1148
Grade	Commercial
Process	Typical
Stepping	Stepping 2

Thermal Information

Ambient Temp (°C)	50.0
Airflow (LFM)	250
Heat Sink	Medium Profile
Custom θSA (°C/W)	5.2
Board Selection	Medium (10"x10")
# of Board Layers	12 to 15
Custom θJB (°C/W)	2.1
Board Temperature	

Thermal Summary

Effective θJA (°C/W)	2.6
Max Ambient (°C)	95.4
Junction Temp(°C)	54.6

Block Summary

Block	Power (W)
CLOCK	0.976
LOGIC	0.000
IO	0.000
BRAM	0.000
DCM	0.000
PMCD	0.000
DSP	0.000
PPC	-
MGT	-
EMAC	-

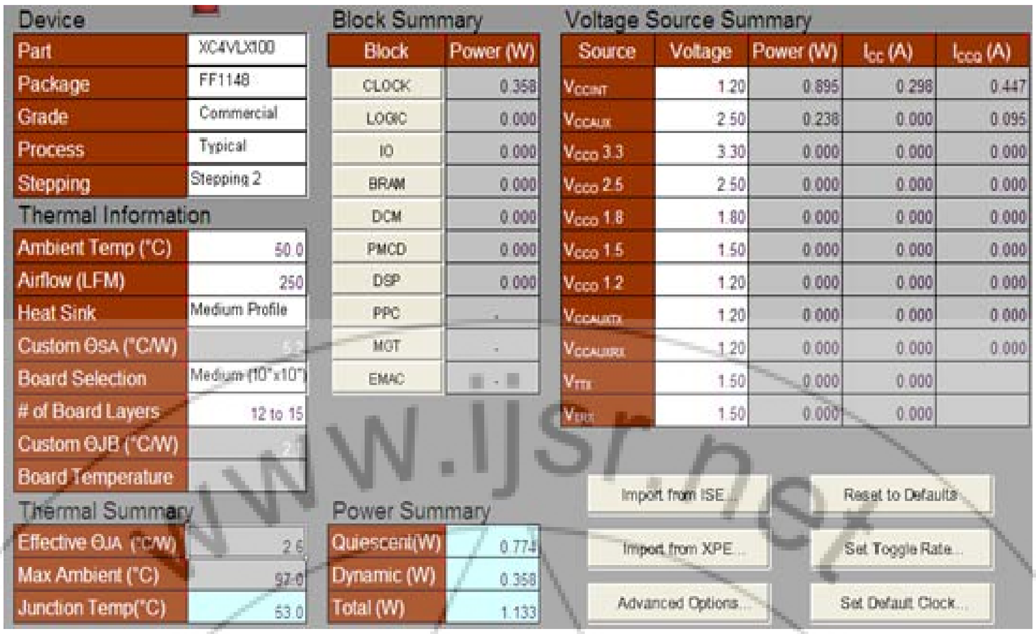
Power Summary

Quiescent(W)	0.788
Dynamic (W)	0.976
Total (W)	1.764

Voltage Source Summary

Source	Voltage	Power (W)	I_{CC} (A)	I_{CCO} (A)
V_{CCINT}	1.20	1.526	0.813	0.459
V_{CCAUX}	2.50	0.238	0.000	0.095
V_{CCO} 3.3	3.30	0.000	0.000	0.000
V_{CCO} 2.5	2.50	0.000	0.000	0.000
V_{CCO} 1.8	1.80	0.000	0.000	0.000
V_{CCO} 1.5	1.50	0.000	0.000	0.000
V_{CCO} 1.2	1.20	0.000	0.000	0.000
$V_{CCAUXTX}$	1.20	0.000	0.000	0.000
$V_{CCAUXRX}$	1.20	0.000	0.000	0.000
V_{TTX}	1.50	0.000	0.000	
V_{TRX}	1.50	0.000	0.000	

Figure 4: Logic power of shift and add multiplier



Device

Part	XC4VLX100
Package	FF1148
Grade	Commercial
Process	Typical
Stepping	Stepping 2

Thermal Information

Ambient Temp (°C)	50.0
Airflow (LFM)	250
Heat Sink	Medium Profile
Custom ΘSA (°C/W)	5.2
Board Selection	Medium (10"x10")
# of Board Layers	12 to 15
Custom ΘJB (°C/W)	2.1
Board Temperature	

Thermal Summary

Effective ΘJA (°C/W)	2.6
Max Ambient (°C)	97.0
Junction Temp(°C)	53.0

Block Summary

Block	Power (W)
CLOCK	0.358
LOGIC	0.000
IO	0.000
BRAM	0.000
DCM	0.000
PMCD	0.000
DSP	0.000
PPC	-
MGT	-
EMAC	-

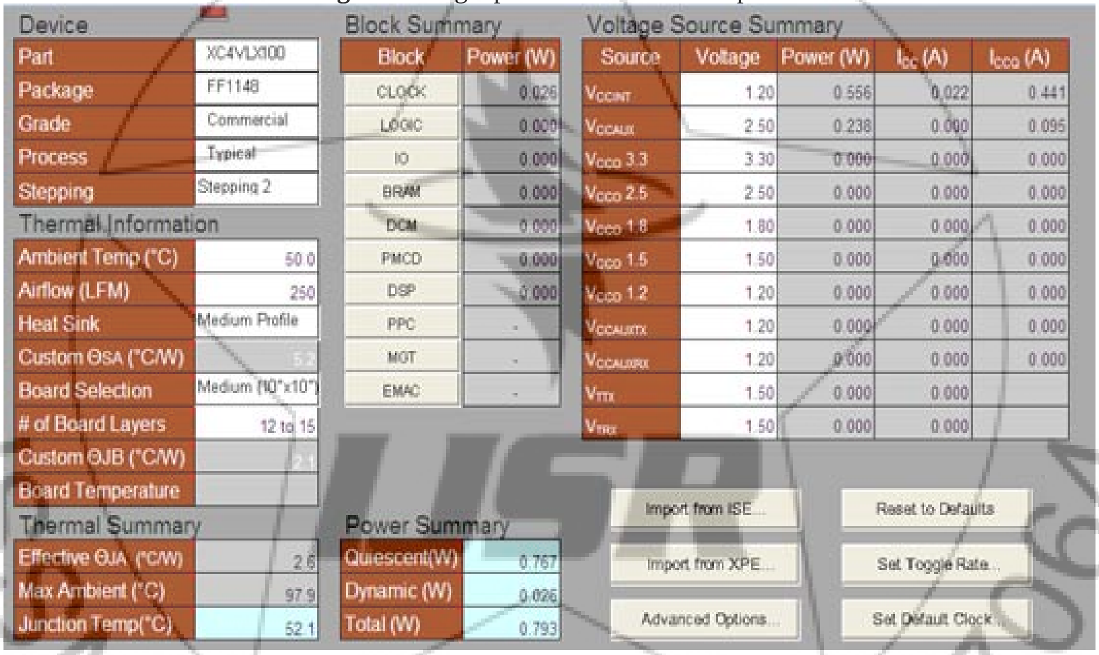
Power Summary

Quiescent(W)	0.774
Dynamic (W)	0.358
Total (W)	1.133

Voltage Source Summary

Source	Voltage	Power (W)	I_{CC} (A)	I_{CCQ} (A)
V_{CCINT}	1.20	0.895	0.298	0.447
V_{CCAUX}	2.50	0.238	0.000	0.095
V_{CCO} 3.3	3.30	0.000	0.000	0.000
V_{CCO} 2.5	2.50	0.000	0.000	0.000
V_{CCO} 1.8	1.80	0.000	0.000	0.000
V_{CCO} 1.5	1.50	0.000	0.000	0.000
V_{CCO} 1.2	1.20	0.000	0.000	0.000
$V_{CCAUXTX}$	1.20	0.000	0.000	0.000
$V_{CCAUXRX}$	1.20	0.000	0.000	0.000
V_{TTX}	1.50	0.000	0.000	
V_{TRX}	1.50	0.000	0.000	

Figure 5: Logic power of radix-2 multiplier



Device

Part	XC4VLX100
Package	FF1148
Grade	Commercial
Process	Typical
Stepping	Stepping 2

Thermal Information

Ambient Temp (°C)	50.0
Airflow (LFM)	250
Heat Sink	Medium Profile
Custom ΘSA (°C/W)	5.2
Board Selection	Medium (10"x10")
# of Board Layers	12 to 15
Custom ΘJB (°C/W)	2.1
Board Temperature	

Thermal Summary

Effective ΘJA (°C/W)	2.6
Max Ambient (°C)	97.9
Junction Temp(°C)	52.1

Block Summary

Block	Power (W)
CLOCK	0.026
LOGIC	0.000
IO	0.000
BRAM	0.000
DCM	0.000
PMCD	0.000
DSP	0.000
PPC	-
MGT	-
EMAC	-

Power Summary

Quiescent(W)	0.767
Dynamic (W)	0.026
Total (W)	0.793

Voltage Source Summary

Source	Voltage	Power (W)	I_{CC} (A)	I_{CCQ} (A)
V_{CCINT}	1.20	0.556	0.022	0.441
V_{CCAUX}	2.50	0.238	0.000	0.095
V_{CCO} 3.3	3.30	0.000	0.000	0.000
V_{CCO} 2.5	2.50	0.000	0.000	0.000
V_{CCO} 1.8	1.80	0.000	0.000	0.000
V_{CCO} 1.5	1.50	0.000	0.000	0.000
V_{CCO} 1.2	1.20	0.000	0.000	0.000
$V_{CCAUXTX}$	1.20	0.000	0.000	0.000
$V_{CCAUXRX}$	1.20	0.000	0.000	0.000
V_{TTX}	1.50	0.000	0.000	
V_{TRX}	1.50	0.000	0.000	

Figure 6: Logic power of radix-4 multiplier

6.2 Output Waveforms

Some output waveforms are obtained by simulating the VHDL coding of different multiplier in Xilinx 9.1.These waveforms are shown in following figures 7, 8, 9.

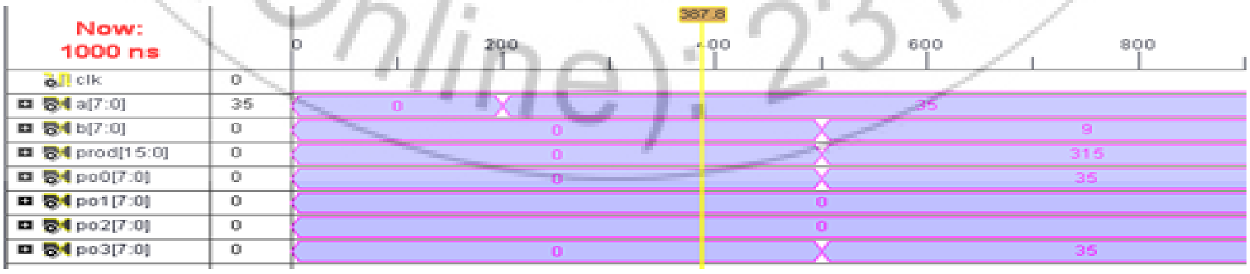


Figure 7: Output waveform of shift and add multiplier

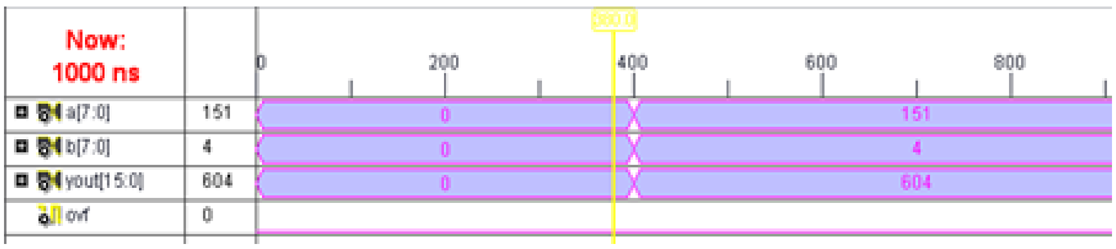


Figure 8: Output waveform of radix-2 multiplier

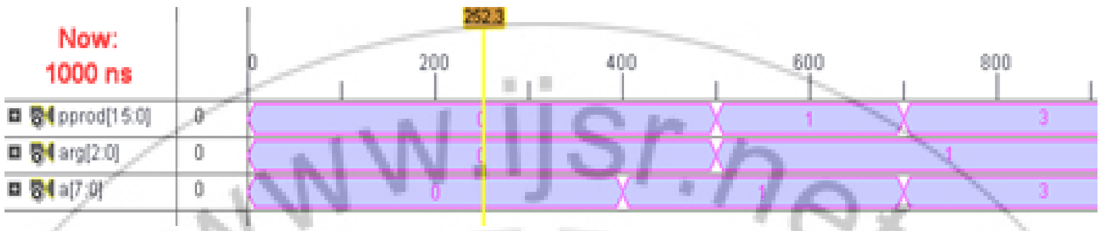


Figure 9: Output waveform of radix-4 multiplier

7. Conclusion and Future Work

7.1 Conclusion

This paper work gives a clear description of different multipliers and their implementation in tap delay finite impulse response (FIR) filter. After simulation, we found that the parallel multiplier (radix-4) are much faster than the serial multiplier (shift and add multiplier). We also found the power consumption of different multiplier using power estimator tool. According to our result parallel multiplier (radix-4) has the low power consumption as compare to serial multiplier (shift and add multiplier).

In comparison of radix- 2 and radix-4 Booth multiplier, we found that the radix- 4 multiplier consumes less power than the radix- 2 multiplier, because it uses almost half the number of iterations and the adder compared to the radix-2. When all three multipliers were compared, we found that array multiplier is the most power consuming, and has the largest area. This is because it uses a lot of adders. Therefore, it will slow down the system because the current system has done a lot of calculations.

Various digital multipliers are one of the most important system components for designing different types of digital systems. Therefore, a better solution for the multiplier is to be found out. We should always ensure that multiplier consumes less power and has small coverage area. So through our dissertation work, we tried to determine which of the three algorithms works best. Finally, we show that the radix-4 multiplier is better in terms of power and speed for the design of digital circuits.

7.2 Future Work

As an attempt to develop arithmetic algorithm and architecture level low power optimization techniques for multiplier design, research presented in this paper has achieved good results, showing a high level of efficiency and optimization techniques. However, there are limitations but some of our work and future research directions are possible. One probable direction is higher than the radix -4. We only took into consideration the radix _ 4 multiplier, because it is a simple and popular choice. To further decrease the number of multiplication and adders higher radix is used, and thus has low power consumption. Another possible direction may be parameters such as a symbol - in the form of amplitude or compliment, which in whichever case prove to be low power consuming, less time consuming and has high speed.

References

[1] K. Likharev and V. Semenov, "RSFQ logichemory family: a new Josephson-junction technology for sub-terahertz-clock-frequency digital systems," IEEE Trans. Appl. Superconductivity, vol. 1, pp. 3-28, March 1992.
[2] L. A. Tawaibeh, "Radix-4 ASIC Design of B Scalable Montgomery Modular Multiplier using Encoding Techniques," M.S. thesis, Oregon %ate University, USA, October 2002.
[3] A. Liakot, S. Roslina, A. Ishak, and M.A.Alauddin, “Design of a Micro-UART for SoC Application,” Computer and Electrical Engineering, Elsevier, Vol. 30, Issue 4, pp. 257-268, June 2004.
[4] L. Petroli, C.A. Lisboa, F.L. Kastensmidt, and L. Carro, “Using Majority Logic to Cope with Long Duration Transient Faults,” Proc. of the 20th annual conference on Integrated circuits and systems design, pp. 354-359, 2007.
[5] http://www.xilinx.com/products/design-tools/ise-design-suite/.
[6] K.H. Tsoi, P.H.W. Leong, "Mullet - a parallel multiplier generator," fpl, pp.691-694, International Conference on Field Programmable Logic and Applications, 2005.
[7] Pedroni Circuit, “Design using VHDL”, Massachussets Institute of Technolgy, 2004.
[8] Christos Meletis, Paul Bougas, George Economakos , Paraskevas Kalivas and Kiamal Pekmestzi, “ High-Speed Pipeline Implementation of Radix-2 DIF Algorithm”, World Academy of Science, Engineering and Technology 2 2005.

Author Profile

Sushant Shekhar is a student of M.Tech (Dual Degree) Electronics & Communication + VLSI engineering at Suresh Gyan Vihar University, Jaipur. He is working on VHDL language.

Ghanshyam Jangid is Assistant Professor in Suresh Gyan Vihar University, Jaipur, Rajasthan. He has done his M.Tech from M.N.I.T, Jaipur, Rajasthan, India