Reconfigurable VLSI architecture for FFT computation

S.Sreenath Kashyap

Abstract— Parallel-prefix adders (also known as carry tree adders) are known to have the best Performance in VLSI designs. The Design of the three types of carry-tree adders namely Kogge-Stone, sparse Kogge-Stone, and spanning carry look ahead adder is done and compares them to the simple Ripple Carry Adder (RCA) . These designs of varied bit-widths were implemented on a Xilinx Spartan 3E FPGA and Dealy measurements were made with XILINX. The carry-tree adders are expected to have a speed advantage over the RCA as bit widths approach in higher bit widths.An Efficient FFT algorithm is designed. The adder which exhibits less delay is used is repaced in the Adder module of FFT.Thsi FFt can be implemented in the high speed DSP applications.

Index Terms— FFT , Kogge stone adder , sparse kogge stone adder , Spanning kogge stone adder , Ripple carry adder , Carry look ahead adder

1 INTRODUCTION

THE saying that if you can count , you can control . Addition is a fundamental operation for a digitsal system, digital signal processing or control system. A fast and accurate operation of a digital system is greatly influenced by the performance of the resident adders. Adders are also very important component in digital systems because of their extensive use in other basic digital operations such as subtraction, multiplication and division. Hence, improving performance of the digital adder would greatly advance the execution of binary operations inside a circuit compromised of such blocks. The performance of a digital circuit block is gauged by analyzing its power dissipation, layout area and its operating speed.

1.1 TYPES OF ADDERS:

The implementation technique of several types of adders and study their characteristics and performance. There are so many types of adders some of them are

- **Ripple carry adder**
- **Carry look-ahead adder**
- **Carry tree adders**

For the same length of binary number, each of the above adders has different performance in terms of Delay, Area, and Power. All designs are assumed to be CMOS static circuits and they are viewed from architectural point of view.

2. BASIC ADDER UNIT

The most basic arithmetic operation is the addition of two binary digits, i.e. bits. A combinational circuit that adds two bits, according the scheme outlined below, is called a half Adder. A full adder is one that adds three bits, the third produced from a previous addition operation. One way of implementing a full adder is to utilizes two half adders in its implementation. The full adder is the basic unit of addition employed in all the adders studied here.

2.1 HALF ADDER:

A half adder is used to add two binary digits together, **A** and **B**. It produces **S**, the sum of A and B, and the corresponding carry out **Co**. Although by itself, a half adder is not extremely useful, it can be used as a building block for larger adding circuits (FA). One possible implementation is using two AND gates, two inverters, and an OR gate instead of a XOR gate as shown below.

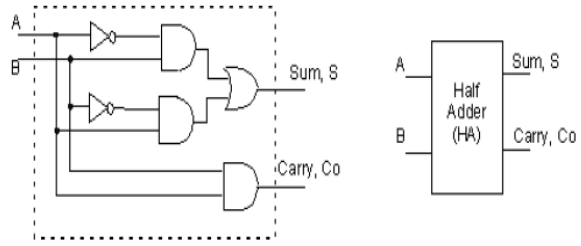


Fig.2.1.Half Adder.logic block

A	B	S	C_o
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Table 2.1: Half adder truth table

- *S.Sreenath kashyap is currently pursuing masters degree program in electric power engineering in SRM University, INDIA,*
- *E-mail: kshyap.foru3@gmail.com*

2.2 FULL ADDER:

A full adder is a combinational circuit that performs the arithmetic sum of three bits: A, B and a carry in, C, from a previous addition. Also, as in the case of the half adder, the full adder produces the corresponding sum, S, and a carry out Co. As mentioned previously a full adder maybe designed by two half adders in series as shown.The sum of A and B are fed to a second half adder, which then adds it to the carry in C (from a previous addition operation) to generate the final sum S. The carry out, Co, is the result of an OR operation taken from the carry outs of both half adders. There are a variety of adders in the literature both at the gate level and transistor level each giving different performances.

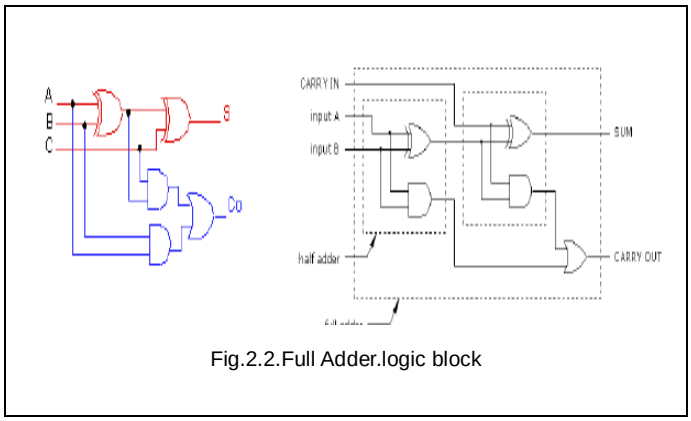


Fig.2.2.Full Adder.logic block

A	B	C	S	Co
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Table 2.2: Full adder truth table

2.3 RIPPLE CARRY ADDER:

The ripple carry adder is constructed by cascading full adders (FA) blocks in series. One full adder is responsible for the addition of two binary digits at any stage of the ripple carry. The carryout of one stage is fed directly to the carry-in of the next stage.

A number of full adders may be added to the ripple carry adder or ripple carry adders of different sizes may be cascaded in order to accommodate binary vector strings of larger sizes. For an n-bit parallel adder, it requires n computational elements (FA). It is composed of four full adders. The augend's bits of x are added to the addend bits of y respectfully of their binary position. Each bit 6 addition creates a sum and a carry out. The carry out is then transmitted to the carry in of the next higher-order bit. The final result creates a sum of four bits plus a carry out (c4). Even though this is a simple adder and can be used to add unrestricted bit length numbers, it is however not very efficient when large bit numbers are used.

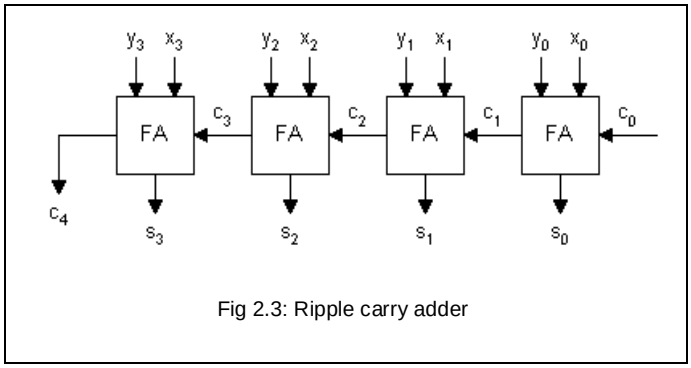


Fig 2.3: Ripple carry adder

One of the most serious drawbacks of this adder is that the delay increases linearly with the bit length. As mentioned before, each full adder has to wait for the carry out of the previous stage to output steady-state result. Therefore even if the adder has a value at its output terminal, it has to wait for the propagation of the carry before the output reaches a correct value. Taking again the example in figure 4, the addition of x4 and y4 cannot reach steady state until c4 becomes available. In turn, c4 has to wait for c3, and so on down to c1. If one full adder takes Tfa seconds to complete its operation, the final result will reach its steady-state value only after 4.Tfa seconds. Its area is n Afa . A (very) small improvement in area consumption can be achieved if it is known in advance that the first carry in (c0) will always be zero. (If so, the first full adder can be replacing by a half adder). In general, assuming all gates has the same delay and area of NAND-2 then this circuit 7 has 3n Tgate delay and 5n Agate.

2.4 CARRY LOOK AHEAD ADDER:

As seen in the ripple-carry adder, its limiting factor is the time it takes to propagate the carry. The carry look-ahead adder solves this problem by calculating the carry signals in advance, based on the input signals. The result is a reduced carry propagation time. To be able to understand how the carry look-ahead adder works, we have to manipulate the Boolean expression dealing with the full adder. The Propagate P and generate G in a full-adder, is given as:

$P = Ai \oplus Bi$ ---------- *Carry propagate*

$Gi = Ai \bullet Bi$---------- *Carry generate*

Notices that both propagate and generate signals depend only on the input bits and thus will be valid after one gate delay. The new expressions for the output sum and the carryout are given by:

$$Si = Pi \oplus Ci\text{-}1$$

$$Ci+1= Gi + PiCi$$

These equations show that a carry signal will be generated in two cases:

1) If both bits Ai and Bi are 1

2) If either Ai or Bi is 1 and the carry-in Ci is 1.

Let's apply these equations for a 4-bit adder: 13

$$C1 = G0 + P0C0$$

$$C2 = G1 + P1C1 = G1 + P1 (G0 + P0C0) = G1 + P1G0 + P1P0C0$$

$$C3 = G2 + P2C2 = G2 + P2G1 + P2P1G0 + P2P1P0C0$$

$$C4 = G3 + P3C3 = G3 + P3G2 + P3P2G1 + P3P2P1G0 + P3P2P1P0C0$$

These expressions show that C2, C3 and C4 do not depend on its previous carry-in. Therefore C4 does not need to wait for C3 to propagate. As soon as C0 is computed, C4 can reach steady state. The same is also true for C2 and C3

The general expression is

$Ci+1= Gi + PiGi-1 + PiPi-1Gi-2 + ……. PiPi-1….P2P1G0 + PiPi-1 ….P1P0C0.$

This is a two level Circuit. In CMOS however the delay of the function is non-linearly dependent on its fan in. Therefore large fan-in gates are not practical.

2.4 SUM GENERATOR:

Carry look-ahead adder's structure can be divided into three parts: the propagate/generate generator, the sum generator, and the carry generator.

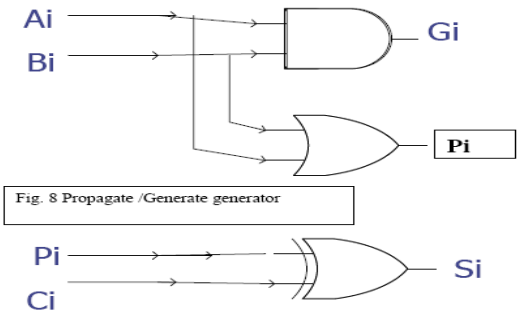


Fig 2.4: Sum Generator

2.5 CARRY GENERATOR:

Below shows the carry generator needed to add four bits numbers. To make the carry generator from 4 bits to n bits, we need only add AND gates and inputs for the OR gate. The largest AND gate in the carry section has always n+1 inputs and the number of AND gates requirements is n. Therefore the design of a 16 bits adder needs the last carry generator section to have 16 AND gates, where the biggest AND gate has 17 inputs. Also the OR gate in this section needs 17 inputs.

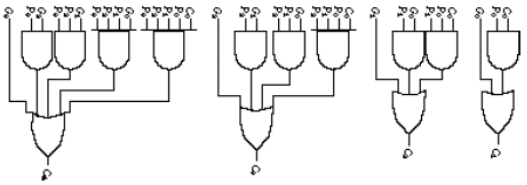


Fig 2.5: Carry Generator

The size and fan-in of the gates needed to implement the Carry-Look-ahead adder is usually limited to four, so 4-bit Carry-Look ahead adder is designed as a block. The 4-bit Carry Look Ahead adder block diagram is shown below

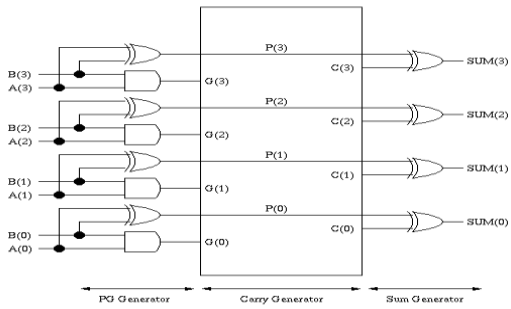
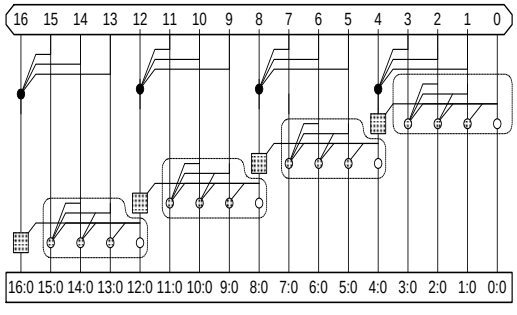


Fig 2.6: Carry look ahead adder with logic blocks

In practice, it is not possible to use the CLA to realize constant delay for the wider-bit adders since there will be a substantial loading capacitance, and hence larger delay and larger power consumption. The CLA has the fastest growing area and power requirements with respect to the bit size. Speed also will drop with increase in bit size. So other techniques may be used. For example a 32-bit Carry-Look ahead adder can be built by using 8 cascaded 4-bit Carry-Look ahead adders (Ripple through between the blocks).



F ig 2.7: Carry look-ahead depth

In this carry look ahead adder the output of the first gp block will be passed to the next and in the same way the entire operation takes palces for every four gp blocks in the form of stages and then the delay will be less when compared to the ripple carry adder.the further improvement of the reduction of delay is being done in the carry tree adders such that low power and area is being consumed.

3. Back ground

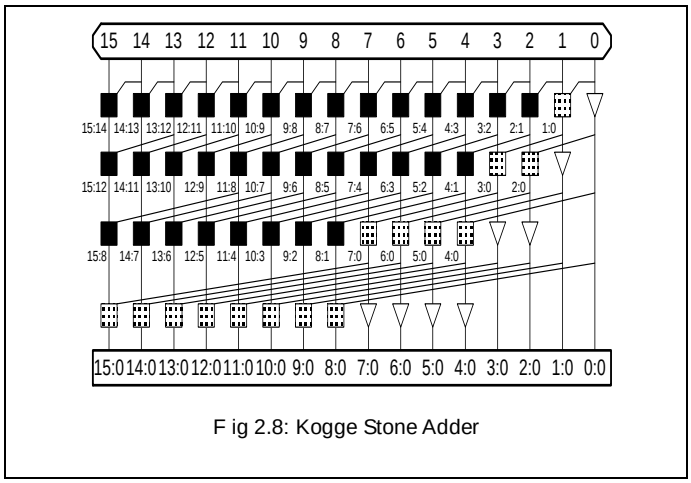
Parallel-prefix adders, also known as carry-tree adders, precompute the propagate and generate signals. . The arrangement of the prefix network gives rise to various families of adders. The different types of carry tree adders are

- Kogge-stone adder
- Sparase kogge-stone adder
- Spanning carry look ahead adder

The above carry tree adders are the adders which are used in thevlsi industry because of there high speed advantage over the large bit widths . the less delay and power is one of the most advantageous factor in which these adders perform .

3.1 KOGGE STONE ADDER:

The Kogge-Stone adder is a parallel prefix form carry look-ahead adder.It is widely considered the fastest adder design possible. It is the common design for high-performance adders in industry.It has high speed performance with reduced delay and occupies less area . Each vertical stage produces a "propagate" and a "generate" bit, as shown. The culminating generate bits (the carries) are produced in the last stage (vertically), and these bits are XOR'd with the initial propagate after the input (the red boxes) to produce the sum bits. E.g., the first (least-significant) sum bit is calculated by XORing the propagate in the farthest-right red box (a "1") with the carry-in (a "0"), producing a "1". The second bit is calculated by XORing the propagate in second box from the right (a "0") with C0 (a "0"), producing a "0".



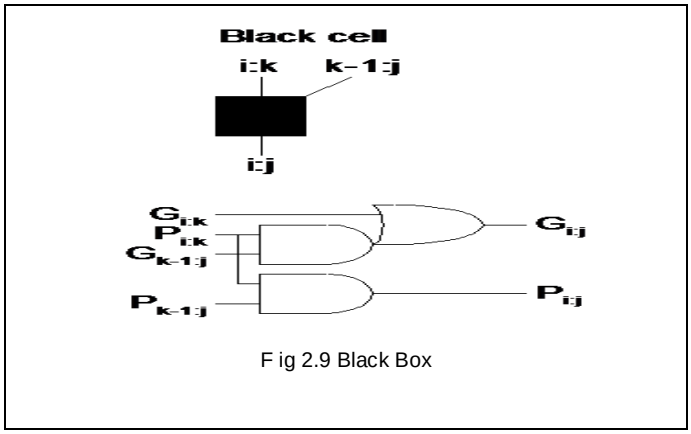
F ig 2.8: Kogge Stone Adder

The Kogge –Stone adder will have three types of blocks. They are mainly

- Black cell
- White box

3.2 Black cell:

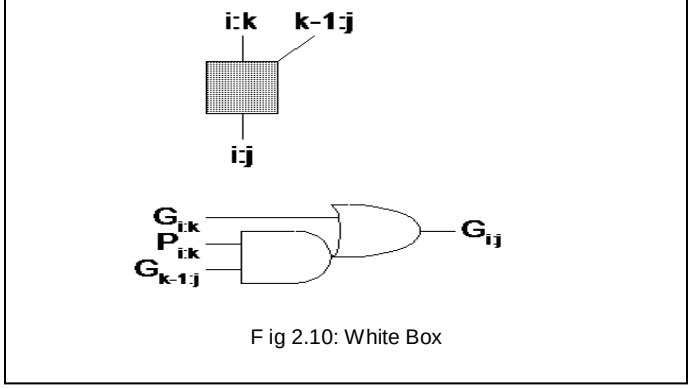
Black cell is mainly a combination of and or gates. the input for this black cell will be the generate and the propagation of the carry of the previous blocks .the output of this black cell will be given to the next blocks such that the generation and propagation of the carry is being done such that the output is being produced .the below figure illustrates the block diagram and the logic diagrammatic representation of the black cell.



F ig 2.9 Black Box

3.3 White box:

White box is mainly a combination of and or gates . the input for this white box will be the input bits .the output of this white box will be the carry generated and such that it is propagated to the next blocks such that the generation and propagation of the carry is being and the output is being produced .the below figure illustrates the block diagram and the logic diagrammatic representation of the white box.



F ig 2.10: White Box

The kogge stone adder the generate and propagate blocks produce the caay and sum such that each block output will act as input to the next block and in the same way such that the final sum is being produced . the below figure represents the schematic how the generation and propogation of carry is done for the production of the final sum

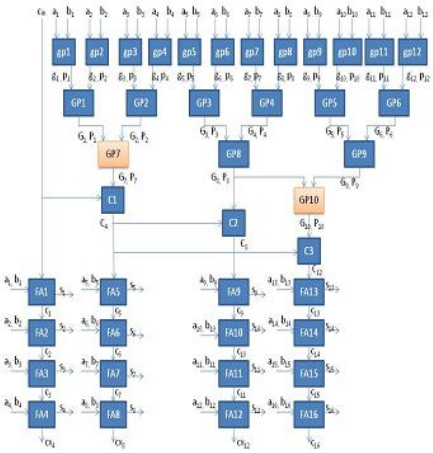
3.4 Sparse kogge stone adder:

Enhancements to the original implementation include increasing the radix and sparsity of the adder. The radix of the adder refers to how many results from the previous level of computation are used to generate the next one. The original implementation uses radix-2, although it's possible to create radix-4 and higher. Doing so increases the power and delay of each stage, but reduces the number of required stages. The sparsity of the adder refers to how many carry bits are generated by the carry-tree. Generating every carry bit is called sparsity-1, whereas generating every other is sparsity-2 and every fourth is sparsity-4. The resulting carries are then used as the carry-in inputs for much shorter ripple carry adders or some other adder design, which generates the final sum bits. Increasing sparsity reduces the total needed computation and can reduce the amount of routing congestion.

F ig 2.11: sparse Kogge Stone Adder

Sparse kogge-stone adder is nothing but the enhancement of the koggestone adder . the block in this sparse kogge stone adder are similar to the kogge stone adder . in this sparse kogge stone a reduction of number of stages is being done by reduceing the genration and propagate units . the ouputs of the previous GP units are being considered such that the combination of consecutive Gp units produces carry once and that one is being given as inout to the next stage . The GP unit blocks will be same such that the generation and propagation of carry is being done such that it will act as inout to the next block and this operations are performed parallely ad stage by stage this is how the reduction of stages is being done and then the final sum is being produced by operations performed by the combination Gp outs given as inouts to the full adders. The delay reduction is done by reducing the number of stages such that the low delay and low power and area is being consumed such that the high speed is being obtained

3.5 Spanning CLA :



F ig 2.12: Spanning CLA

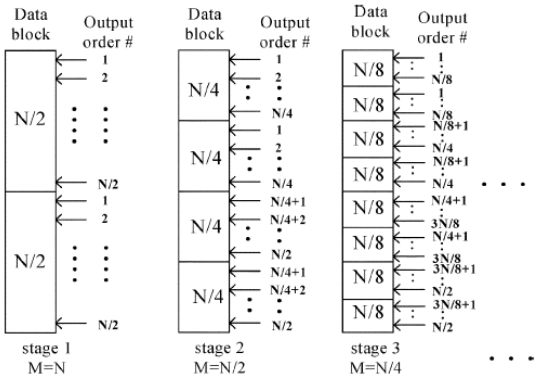
In this spanning CLA a reduction of number of stages is being done by reduceing the genration and propagate units . the ouputs of the previous GP units are being considered such that the combination of consecutive Gp units produces carry once and that one is being given as inout to the next stage .

The GP unit blocks will be same such that the generation and propagation of carry is being done such that it will act as inout to the next block and this operations are performed parallely ad stage by stage this is how the reduction of stages is being done and then the final sum is being produced by operations performed by the combination Gp outs given as inouts to the full adders. The delay reduction is done by reducing the number of stages such that the low delay and low power and area is being consumed such that the high speed is being obtained .

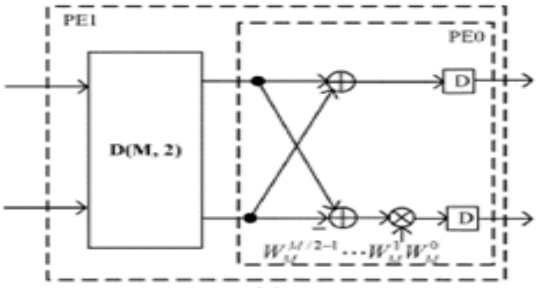
4. FFT DESIGN

DISCRETE Fourier transform (DFT) is one of the fundamental operations in the field of digital signal processing. The DFT, with a transform length equal to a power of 2, isusually implemented with the fast Fourier transform (FFT). In many applications, such as asymmetric digital subscriberline (ADSL) and orthogonal frequency-division multiplexing (OFDM), the transform length is required to belarge and the previous DFT structures with computational complexityof are not practical for VLSI implementation. Inrecent literature, three low-cost

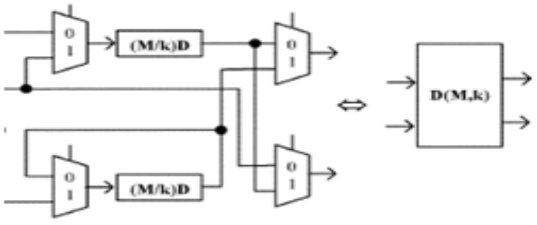
and high-throughput systolicarchitectures have been presented with a computational complexity of , which are regular and are suitable for VLSI implementation. Although these -point FFT algorithms are hardware efficient,they have long latency of clock cycles and their hardwareutilization can be improved further. For example, the datareordering strategy used in [3] is the delay-feedback (DF) architecture with a delay element utilization rate of 100%;but theutilization rate of its multipliers is just 50%.Assume we have a prefetch buffer to ensure concurrent readand write operations; the hardware can be more efficiently used. By applying concurrent computation into the butterfly operations,this brief improves the throughput rate of the previouslyproposed FFT architectures by a factor of 2. High processingspeed leads to reduced number of required delay elements sincefewer intermediate results need to be stored. The latency canalso be reduced by a factor of 2.



F ig 2.13: FFT Algorithm



F ig 2.14: FFT Architecture



F ig 2.15: FFT Internal Blocks

The adders in the FFT design are being replaced with the adders which we are designed namely carry tree adders.
The high speend performance is being achieved by repalceing the adders. There is a reduction in the area delay after using the carry tree adders in the fft design.

5. RESULTS & DISCUSSION

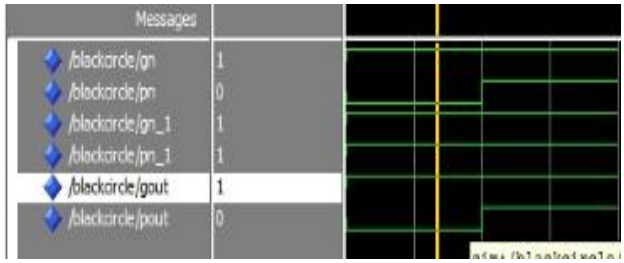


Fig 2.16: Black Cell

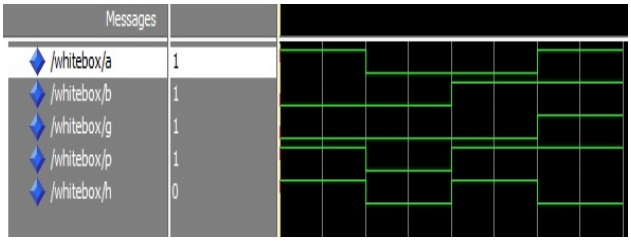


Fig 2.17: White Cell

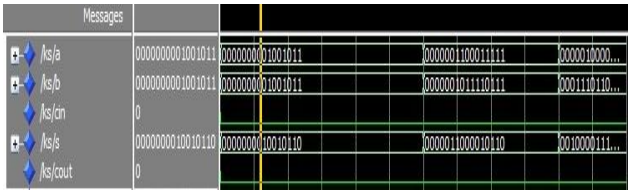


Fig 2.18: Kogge Stone Adder

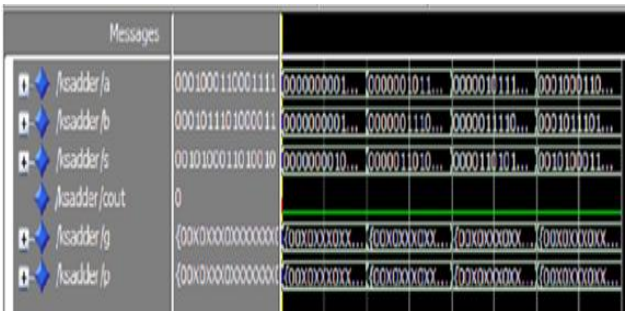


Fig 2.19: Sparse Kogge Stone Adder

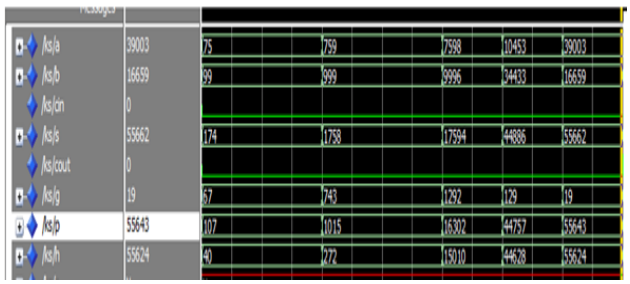


Fig 2.20 : Spanning CLA

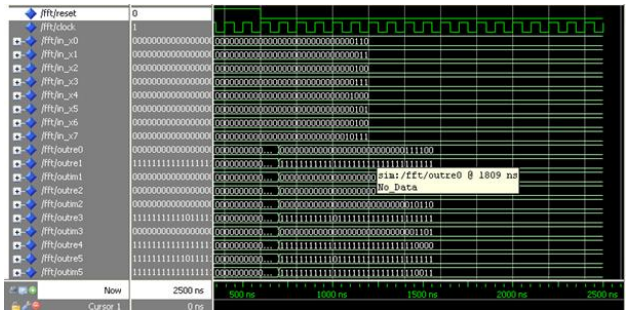


Fig 2.21: FFT OUTPUT

5.1: Delay Analysis:

The synthesis of the above adders is done in Xilinx. The delay variations are being observed in the adders and then the delay variations are being compared in between the above designed adders for different bit widths and then tabulated.

ADDER	BIT WIDTH	DELAY(n SEC)
Kogge stone	16	15.072 n Sec
Sparse Kogge stone	16	16.396 n Sec
Spanning CLA	16	18.247 n Sec

Table : DELAY ANALYSIS BETWEEN ADDERS

6. CONCLUSION AND FUTURE WORK

The Adders namely ripple carry adder, carry look ahead adder, Kogge stone adder , sparse Kogge stone adder , spanning carry look ahead adder are discussed in detail. VHDL code was written for all the modules within the Adder. Each individual module was tested for its correct functionality and then all the modules were integrated to form an entire ripple carry and carry look ahead Kogge stone and sparse Kogge stone adder modules spanning CLA. The adders are designed for 16 bit widths only . Simulation is done in XILINX and the delay is measured.

This project has resulted in the development of Adders Design with reduced delay. The FFT algorithm is designed. The adder module in the FFT is being replaced by the kogge stone adder. These Adders can be used in the place where the adders will have efficient usage with less delay and high speed applications can be implemented. Mostly these types of adders can be used in DSP applications like FIR, IIR filter designs. The FFT module algorithm can be implemented in the design OFDM transmitter and receivers for the generation of OFDM signal by transforming a spectrum (amplitude and phase of each component) into a time domain signal.

REFERENCES

[1] N.H.E.Westte and D.Harris, CMOS VLSI Design ,Pearson-Addison-Wesley, 4 – edition,2011

[2] R.P. Brent and H.T. Kung, "A regular layout for parallel adders," IEEE Trans. Comput., vol. C-31, pp. 260-264, 1982.

[3] D.Harris, "A Taxonomy of Parallel Prefix Networks," in Proc. 37th Asilomar conf. signals systems and computers, PP. 2213-7, 2003.

[4] P.M.Kogge and H.S. Stone, "A Parallel Algorithm for the Efficient Solution of a General Class of Recurrence Equations," IEEE Trans. On Computers, Vol. C-22, No 8. August 1973.

[5] P.Ndai, S. Lu, D. Somesekhar, and K. Roy, "Fine Grained Redundancy in Adders," Int. Symp. On Quality Electronic Design, PP. 317-321, March 2007.

[6] T.Lynch and E. E. Swartzlander, "A Spanning Tree Carry Lookaead Adder," IEEE Trans. On computers, vol. 41 , no . 8,pp. 931-939, Aug 1992.

[7] D. Gizopoulos, M. Psarakis, A. Paschalis, and Y.Zorian, "Easily Testable Cellular Carry Lookahead Adders," Journal of Electronic Testing: Theory and Applications 19,285-293,2003.

[8] S.Xing and W. W. H. Yu, "FPGA Adders: Performance Evaluation And Optimal Design," IEEE Design & Test of Computers ,vol. 15, no.1,ppp.24-29,jan1998.

[9] M.Becvar and P.Stukjunger," fixed Point Arthematic in FPGA," Acta Polytechnicia, vol.45, no.2,pp. 67-72,2005.

[10] K.Virtoroulis and S. J. Al-Khalili, "Perfromance of Parallel Prefix Adders Implemented With FPGA technology ,"