Implementation of A Wallace Tree Approach for Data Aggregation in Wireless Sensor Nodes on ZED board and Send that Aggregated Data to the Sink through WSN Master Development Board(IOT)

[1]Dr. M. KAMARAJU, [2]D. Ramya Krishna, [3]T. Subhashini

[1,2,3] Department of ECE, Gudlavalleru Engineering College, AP, India.

Abstract: A Wireless Sensor Network(WSN) is a combination of sensor nodes(motes) which are used to sense and process the physical and environmental conditions. The WSN collects the data from sensor nodes and send it to the sink. If every node in the WSN can communicate with the sink then it leads to more power consumption. To avoid this problem the sensing data is aggregated at one node in sensor network and that node only can communicate with the sink hence the energy consumption of the WSN is decreased for radio communication. This data aggregation can be done by using different existing tree structures like Binary tree, Folded tree, but these are having more delay for data aggregation process. To decrease the delay for data aggregation process a Wallace tree is proposed. The expected results are Wallace tree is having less delay compared to Binary tree, Folded tree, this Wallace tree is implemented on ZED board to the check its functionality on real time applications and also that aggregated data which at ZED board can be send to the sink through WSN Master Development Board(IOT).

Key Words: Sensor Nodes(Motes), Wallace Tree, ZED board, WSN Master Development Board(IOT).

I. INTRODUCTION

A WSN consisting of small, portable, less complex devices named as sensor nodes [9]. These Sensor nodes are also named as motes, they can sense and monitor the environment. The nodes in the sensor network can communicate with each other and also send that sensing data to the sink. By using wireless link(Internet), WSN is having many applications like weather monitoring, Area monitoring, intelligent building motoring and also used in military for battle field surveillance.

II. CHARACTERISTICS OF WSN

Due to large number of applications [9], the characteristics of WSN are given as follows:

Application Specific: The WSN's are designed for specific application only.

Dynamic Behavior: WSN's are error free to harsh environment because they are placed at the environment, which has been changed its condition dynamically.

Mobility: Due to dynamic behavior, different applications, routing algorithms the nodes in the sensor network can be move to anywhere.

Scale & Density: To design WSN for a large variety of applications the density of the nodes must be high.

Resource Constraints: Sensor nodes are very small and those are working by using a small battery inside it. If battery power is over then sensor node stops it's working, so energy consumption for a sensor node must be very low.

Data Aggregation: It is just a process of aggregating the sensor information through aggregation approaches. This technique was mainly utilized to resolve both overlap and implosion problem in data centric routing.

III. REQUIREMENTS OF WSN

To design a reliable networks, the sensor node hardware designing and implementation should be minimized for three requirements [1]. Those are

Small Size: Mobility of sensor nodes is the most important characteristic in WSN's for that purpose the size of the node must be very small to move.

Low Cost: To cover large areas in WSN's many number of nodes are required, hence to decrease the network designing cost the cost of the each sensor node must be low.

Low Power: The WSN consisting of many number of nodes, replacement of battery in sensor nodes is very difficult, costly, or sometimes it may be impossible. The Nodes must have energy efficient, then only it function for long time without running out of power.

IV. EXISTING WORK

To aggregate the data from sensor nodes, there are different tree structures [1] [5] are existed. Those are

1.Binary Tree
2.Folded Tree

1. Binary tree

In WSN, sensors (nodes or programmable elements) are placed in sensing region. Those sensors are used to sense the data and send that sensing data to sink (central location). All those Programmable Elements(PE's) are connected in different structures. Those structures are known as trees.

In Binary tree 4 PE's (PE_1, PE_2, PE_3, PE_4,)are collected data and that data is added and send it to two PE's (PE_5, PE_6,) then those two data's are aggregated and send it to PE_7.Then PE_7 send that aggregated information to sink.(As shown in Fig. 1)

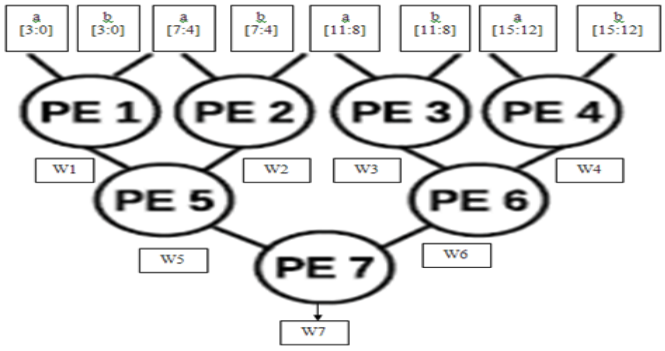


Fig. 1. Binary tree diagram

Working of Binary tree

1. PE_1 takes inputs a[3:0],b[3:0] and PE_2 takes inputs as a[7:4],b[7:4] and produces outputs as w1 and w2.
2. PE_3 takes inputs a[11:8],b[11:8] and PE_4 takes inputs as a[15:9],b[15:9] and produces outputs as w3 and w4.
3. W1 and w2 are applied as inputs for PE_5 then it produces output as w5.
4. W3 and w4 are applied as inputs for PE_6 then it produces output as w6
5. W5 and w6 are applied as inputs for PE_7 then it produces output as w7.
6. That aggregated output w7is send to sink.

2. Folded tree

In normal Binary tree it requires 7 Programmable elements(PE's) to aggregate two 16 bit inputs and that produces one output but in Folded tree structure [1] [3] only 4 PE's are enough to add two 16 bit numbers/data.(As shown in Fig. 2).

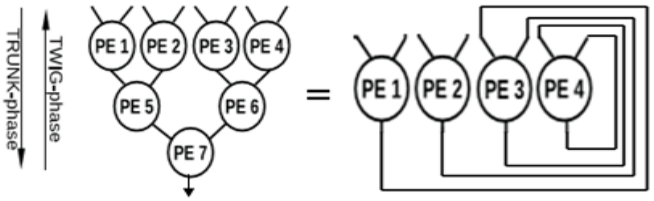


Fig. 2. Binary tree and Folded tree diagram

Phases in Folded tree:

Folded tree is having two phases in it. Those are

i) Trunk Phase
ii) Twig phase

i) Trunk Phase

Trunk phase diagram is as shown in Fig. 3.

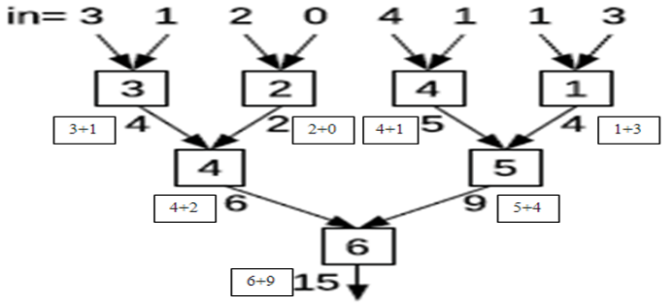


Fig. 3. Trunk phase diagram

In this phase, (As shown in Fig. 3)

1. The left value is stored as Programmable Element(PE).
2. The left and right values are added and pass that output to next stage.

ii) Twig Phase

In this phase, (As shown in Fig. 4)

1. The bottom value is added with Programmable Element (PE) and propagate that value to right side of the PE.
2. At left side of the PE only bottom value (value entered into PE) is only propagated.

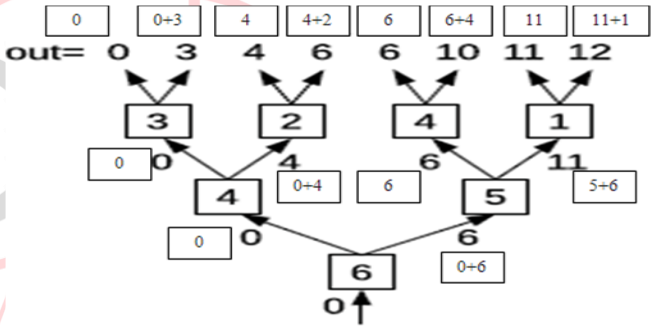


Fig. 4. Twig phase diagram

5. Proposed Work:

To aggregate the data from sensor nodes, in this paper a Wallace Tree was proposed. That tree is dumped on Zink Evaluation Development (ZED) board, to check its functionality practically. At last that aggregated data which is at ZED board is send to the sink through WSN Master Development board.

1. Wallace Tree

Wallace tree [4] is a multiplier, that reduces the number of partial products and those are added by using Sklansky adder structure that is it multiples the data from various nodes and send that multiplied output to the sink, like that it performs data aggregation process.

Multiplication [6] is One of the main operation in aggregation process of wireless Sensor Networks. Here first the partial products are generated after that all those are reduced by using Wallace Tree structure. In that Wallace tree by using 3:2 , 4:2 , 5:2 compressors all those partial products are reduced and at

the last stage all those are added by using Sklansky adder because of it's less delay and area.

Wallace Tree Structure

Wallace tree is a multiplier which is used to reduce partial products greater than 32 bits and at the last stage of this Wallace tree structure a Sklansky adder is used to add the remaining partial products.

The Wallace tree has three steps. Those are i) Partial Product Generation Stage, ii) Partial Product Reduction Stage, iii) Partial Product Addition Stages as shown in Fig.5:

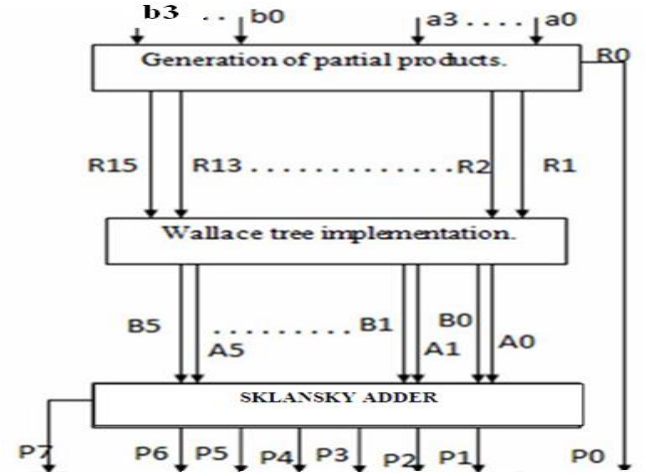


Fig.5. Design flow of proposed work

i) Partial Product Generation Stage

Partial product generation is the very first step in binary multiplier. These are the intermediate terms which are generated based on the value of multiplier. If the multiplier bit is 'Zero(0)', then partial product row is zero, and if it is 'one(1)', then the multiplicand is copied as it is. From the second bit multiplication onwards, each partial product row is shifted one unit to the left. And in signed bit multiplication, the sign bit is also extended to the left. Partial product generators for a conventional multiplier consist of a series of logic AND gates as shown in Fig.6.

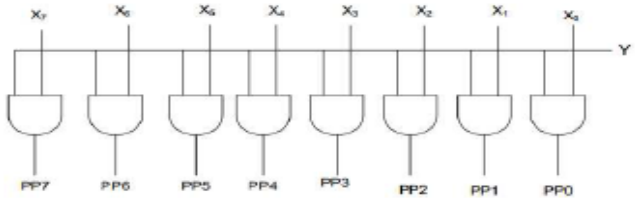


Fig.6. Partial product generation

ii) Partial Product Reduction Stage

In the proposed architecture, partial product reduction is done by the use of 4:2, 5:2 compressors and the final stage of addition is performed by a Sklansky adder. In high-speed designs, the Wallace tree construction process is usually used to reduce the partial products in a tree like fashion, and it produce two rows of partial products that can be added in the last stage. The Wallace tree is fast because the critical path delay is proportional to the logarithm(log) of the number of bits in the multiplier. This method considers all the bits in each column at a time and compresses them into two bits (a sum and a carry). The speed, area and power consumption of the multipliers will be in direct proportion to the compression efficiency of the compressors.

4-2 Compressor:

The 4-2 compressor [4] as shown in Fig.7. has four inputs are represented as X1, X2, X3, X4 and two outputs Sum and Carry along with a Carry-in (Cin) and a Carry-out (Cout) as shown in Fig 5. The input Cin is the output from the previous lower significant stage of the compressor. The Cout is the output to the next significant stage of the compressor block.

5-2 Compressor:

The 5-2 Compressor [4] block has five inputs represented as X1,X2,X3,X4and X5 and two outputs represented as Sum and Carry, along with two input carry bits (Cin1, Cin2) and two output carry bits (Cout1,Cout2) as shown in Fig.8. The input carry bits are from the outputs of the previous lesser significant compressor block and the output carry's are passed on to the next higher significant compressor block.

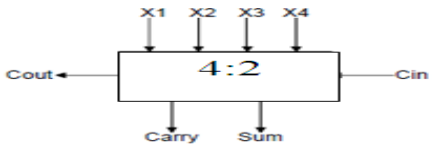


Fig.7. 5:2 Compressor

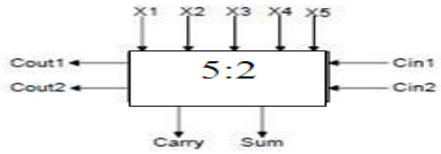


Fig.8. 4:2 Compressor

The following Fig.9. Shows the block diagram for the conventional 8 bits x 8 bits Wallace Tree high speed multiplier. It Reduce the number of partial products by using these 4:2 and 5:2 compressors and form two layers at last stage. Those two layers are added by using Sklansky adder.

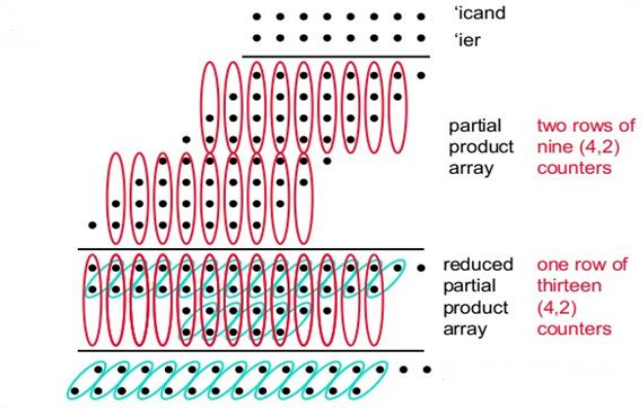


Fig. 9. bit Wallace tree structure

iii) Partial Product Addition Stage

At the final stage of the Wallace Tree [4] the Sklansky adder is used. This Sklansky adder [10] reduces the delay of the addition process to $\log_2 N$ stages. Hence delay of the Wallace Tree with Sklansky adder is very low.

Sklansky Adder

The Sklansky tree as shown in Fig 10. is commonly known as the divide-and-conquer tree because it reduces the delay to $\log_2 N$ stages. It computes the intermediate prefixes along with the large group prefixes. Then the delay of the network also reduces.

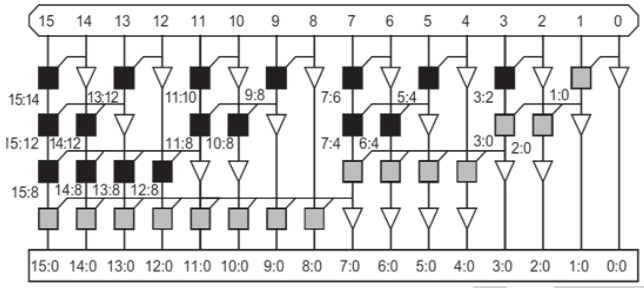


Fig.10. 16-bit Sklansky adder

2. Implementation of Wallace Tree on ZED Board:

The developed 32 bit Wallace Tree is dumped into Zed Board Zynq™-7000 Development Board, to aggregate the data from sensor nodes.

General Description about Zed Board Zynq™-7000 Development Board

Zed Board is a low-cost development board for the Xilinx Zynq-7000. This board contains everything necessary to create a Android, Linux, Windows or other OS/RTOS based design. Additionally, several expansion connectors expose the programmable logic I/Os and processing system for easy user access. To Take advantage of the Zynq-7000 AP SoC's tightly coupled ARM processing system and 7 series programmable logic to create unique and powerful designs with the Zed Board.

Target Applications

1. Video processing
2. Embedded ARM processing
3. General Zynq-7000 AP SoC prototyping

3. Interfacing of WSN Board (IoT) to the ZED Board

The 32 bit Wallace Tree is implemented in ZED board that gives aggregates data from all the nodes (multiplied) and that data is send to the sink by interfacing of Wireless Sensor network Master development board (IoT board) to the ZED board wirelessly.

WSN Master Development board

IoT, or the Internet of Things, can be loosely defined as a network of a very small, low power ,low cost, ubiquitous electronic devices where sensing data and communicating information occur without direct human intervention. Each device in the sensor network functions as a "smart node", by sensing information and performing signal processing to filter signals from noise and to reduce the bandwidth required for node to node communications. The nodes need to communicate with a centralized node in a secure manner to protect, store and process data (data aggregation), and bounce actionable information down to humans.

The WSN board includes a microcontroller, BLE transceiver, multiple environmental sensors and coin-cell holder. The WSN kit also includes a programming adapter compatible with the ARM mbed HDK specification.

The WSN Master Development board Technically Support Ethernet, Zigbee, Wifi, GSM/GPRS ,On Board SMA Antenna (User Can change their Own antenna), BLUETOOTH, GPS.

Applications IoT

- **Wearable devices:** smart wrist and arm bands, watches, fitness and healthcare devices.
- **Smart Home:** smart lighting, appliances, energy-control and home-security devices.
- **Smart City:** smart asset tracking, metering etc.

VI. RESULTS

1. Programmable Element (PE)

Programming Element (PE) is the basic element used for addition process in Binary tree for data aggregation process in Wireless Sensor Nodes. This is developed by using Xilinx 13.1 then it produces a RTL Schematic diagram as shown in Fig.11. it show that PE adds the two 4 bit inputs a and b and it produces a result of 5 bit output y.

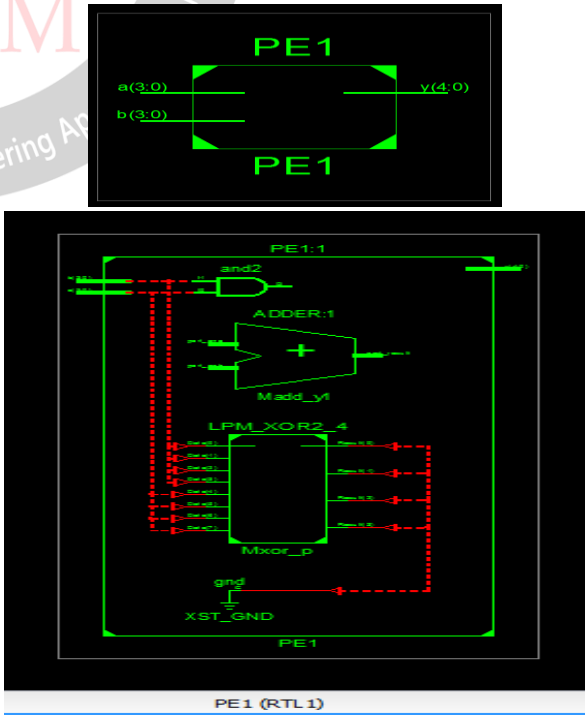


Fig.11. RTL schematic for PE

The PE is synthesized by using Xilinx 13.1 then it produces the synthesis report which gives the delay, memory reports as follows:

Timing Summary:
Maximum combinational path delay: 7.955ns
Total memory usage is 163868 kilobytes

Then this PE is simulated by using Modelsim XE III 6.4b then it adds the two inputs 5 and 5 and it produces a result of sum 10 as shown in Fig.12. mark.

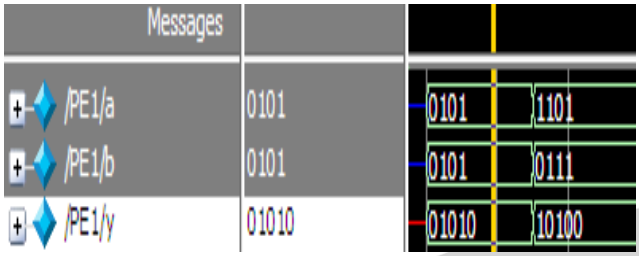


Fig.12. Simulation waveforms for PE

2. Binary Tree Structure

Binary Tree as shown in Fig.13. is one of the existing tree network used in data aggregation process in Wireless Sensor Nodes. This tree uses PE is the basic element to add the data from different nodes. This tree structure is developed in Xilinx 13.1 then it produces RTL diagram for Binary Tree as shown in Fig.14.

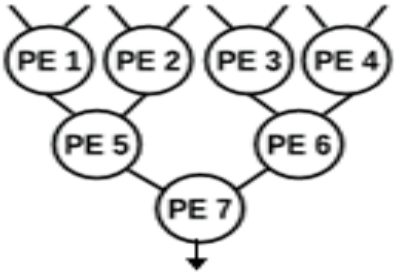


Fig.13. Binary tree

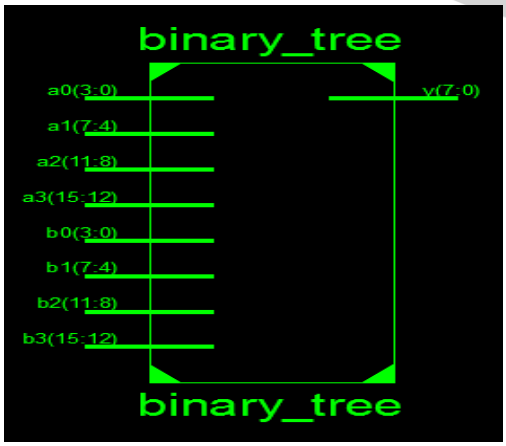


Fig.14. RTL schematic of Binary tree

The internal RTL Schematic of Binary Tree shows how the data flow in tree structure as shown in Fig.15.

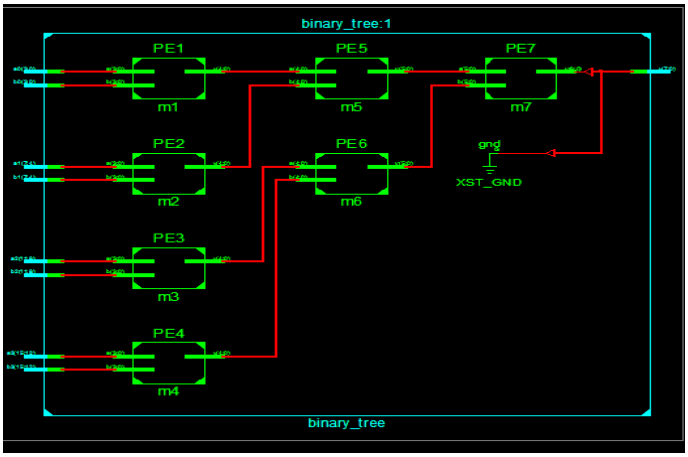


Fig.15. Internal RTL schematic of Binary tree

This Binary Tree is synthesized by using Xilinx 13.1 then it produces the synthesis report which gives the delay, memory reports as follows:

Timing Summary:
Maximum combinational path delay: 14.127ns
Total memory usage is 164828 kilobytes

Then Binary Tree is simulated by using Modelsim XE III 6.4b, it adds the two 16 bit inputs a(3,2,4,1) and b(1,0,1,3) and it produces a result of 7 bit output y(1+3+4+1+2+0+3+1=1111=15) as shown in below Fig.16. mark.

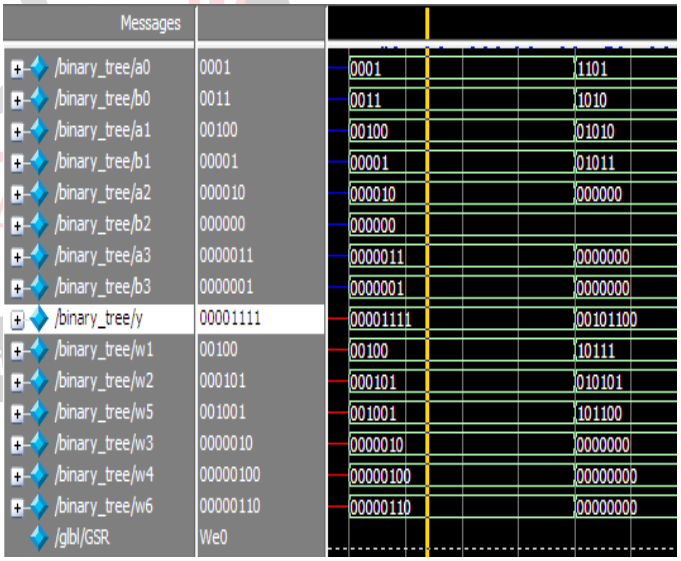


Fig.16. Simulation waveforms of Binary tree

3. Programming Element by Prefix Logic

Programming element (PE) is the basic element developed based upon Carry look ahead prefix logic used for addition process in Folded tree for data aggregation process in Wireless Sensor Nodes. This is developed by using Xilinx 13.1 then it produces a RTL Schematic diagram as shown in Fig.17. it show that PE adds the two 4 bit inputs pe_element and b and it produces a result of 5 bit output y.

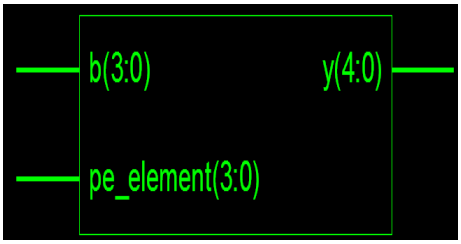


Fig.17. RTL schematic diagram of PE by prefix logic

This PE with prefix logic is synthesized by using Xilinx 13.1 then it produces the synthesis report which gives the delay, memory reports as follows:

Timing Summary:

Maximum combinational path delay: 6.991ns

Total memory usage is 211884 kilobytes

Then PE with prefix logic is simulated by using Modelsim XE III 6.4b, it adds the two inputs 10 and 5 and it produces a result of 15 as shown in below Fig.18. mark.

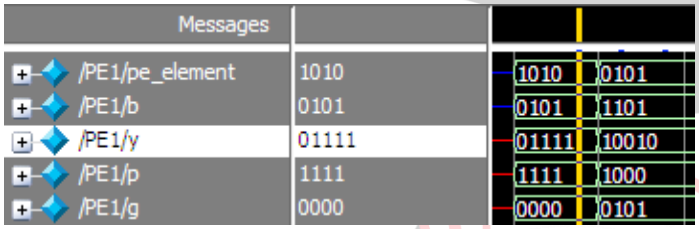


Fig.18. Simulation for PE by prefix logic waveforms

4. Folded Tree

Folded tree as shown in Fig.19. is also another existing tree network in Wireless Sensor Nodes for data aggregation process.

This Folded tree is having two phases [1] [3] in it. Those are

i) Trunk phase

ii) Twig phase

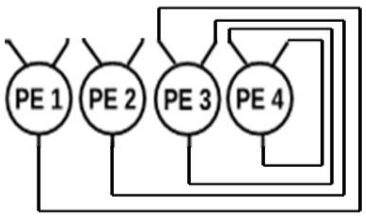


Fig.19. Folded tree

Example of Trunk and Twig phase Process is given as follows:

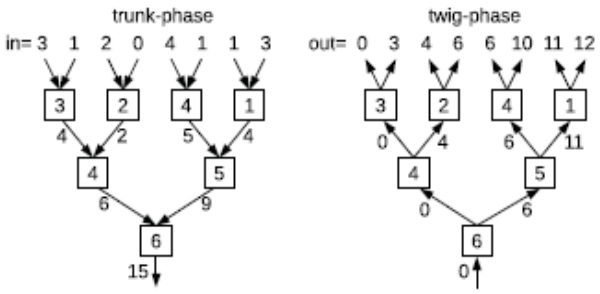


Fig.20. Prefix calculation with sum-operator using Blelloch's generic approach in a trunk and twig phase.

For a simple addition process, if the prefix element of the ordered set [3, 1, 2, 0, 4, 1, 1, 3] is $\sum ai$ = 15. Blelloch's procedure to calculate the prefix-operations [11] [12] on a Folded tree requires two phases as shown in Fig.20. In the trunk phase, the left value L is saved as Lsave and it is added to the right value R, which is passed on to the root. This process is continues until the parallel prefix element fifteen(15) is found at the root. At each time, a store and calculate operation is executed. Then the twig-phase starts, in this phase the data moves in the opposite direction, from the root to the leaves. Now the incoming value, beginning with the sum identity element 0 at the root, and it is passed to the left child, while it is also added to the previously saved Lsave and that is passed to the right child. In the end, the reduced prefix set is found at the leaves.

4.1. PE Element Programming in Folded Tree

To perform data aggregation in Folded tree we have to program the PE elements with prefix logic in two ways. Those are

1. Trunk phase programming
2. Twig phase programming

4.1.1. Trunk Phase Programming

In Trunk phase the Folded tree adds the two 16 bit inputs a, b and finally it produces the result as 7 bit output y as shown in Fig.21. This is developed by using Xilinx 13.1 then it produces a RTL Schematic diagram as shown in Fig.22.

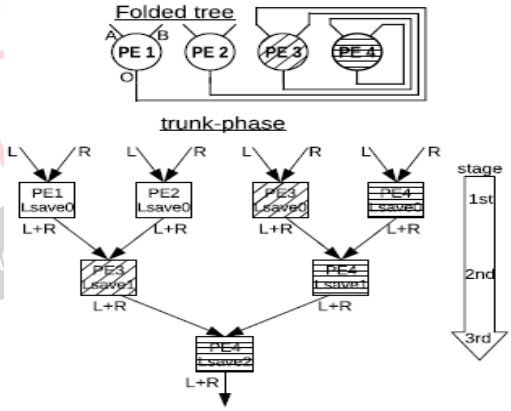


Fig.21.Trunk phase in Folded tree

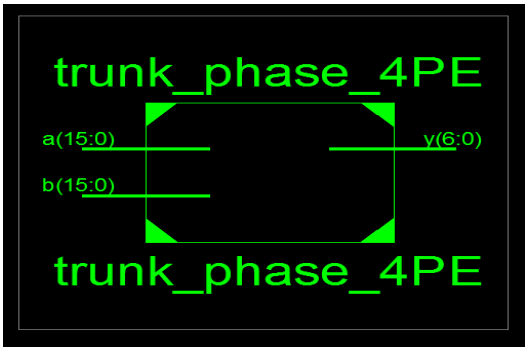


Fig.22. RTL schematic of trunk phase

The internal RTL Schematic of Trunk phase in Folded Tree shows how the data flow in tree structure as shown in Fig.23.

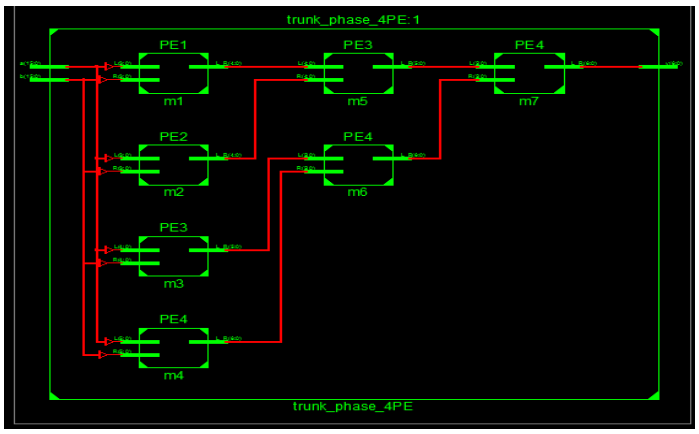


Fig.23. Internal RTL schematic of PE programming in trunk phase

This Trunk phase in Folded Tree is simulated by using Modelsim XE III 6.4b then it adds the two inputs a(3,2,4,1) and b(1,0,1,3) and it produces a result of 7 bit output y(1+3+4+1+2+0+3+1=1111=15) bit by bit as shown in below Fig.24. mark.

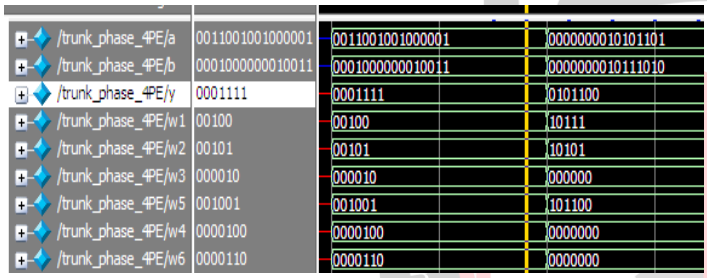


Fig.24. Simulation waveform of PE programming in trunk phase

The Trunk phase in Folded Tree is synthesized by using Xilinx 13.1 then it produces the synthesis report which gives the delay, memory reports as follows:

Timing Summary:

Maximum combinational path delay: 15.186ns

Total memory usage is 164444 kilobytes

4.1.2. Twig Phase Programming

In Twig phase the Folded tree adds the two 16 bit inputs a, b and finally it produces the result as 7 bit output w7 and also it send back that data to inputs also. This process is as shown in Fig.25.This is developed by using Xilinx 13.1 then it produces a RTL Schematic diagram as shown in Fig.26.

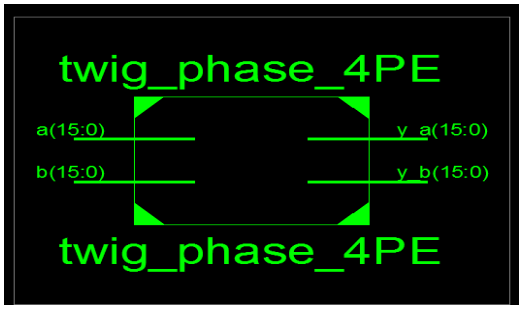


Fig.26. RTL schematic of twig phase

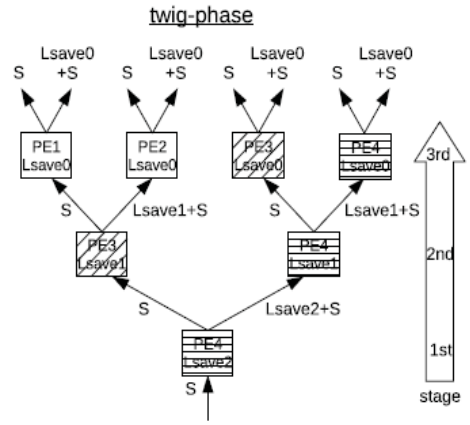


Fig.25. Twig phase in Folded tree

The internal RTL Schematic of Twig phase in Folded Tree shows how the data flow in tree structure as shown in Fig.27.

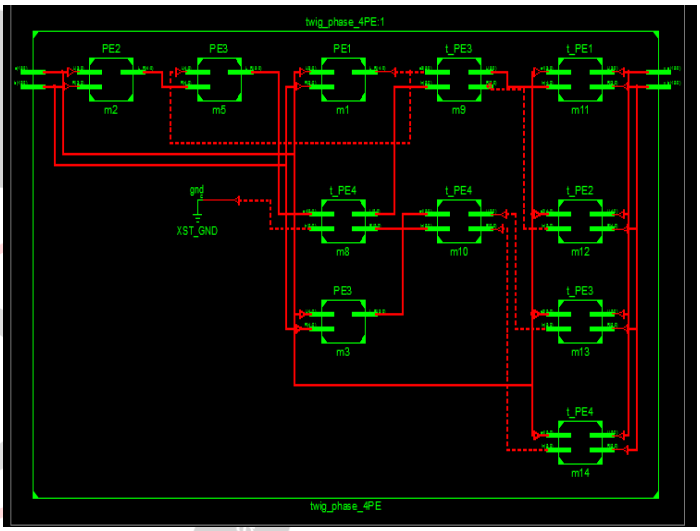


Fig.27. Internal RTL schematic of PE programming in twig phase

This Twig phase in Folded Tree is simulated by using Modelsim XE III 6.4b then it adds the two inputs a(3,2,4,1), b(1,0,1,3) and it produces a output w7(1111=15) and also it send back that data to inputs and produce y_a(0,4,6,11), y_b(3,6,10,12) also. This process is as shown in Fig.28.

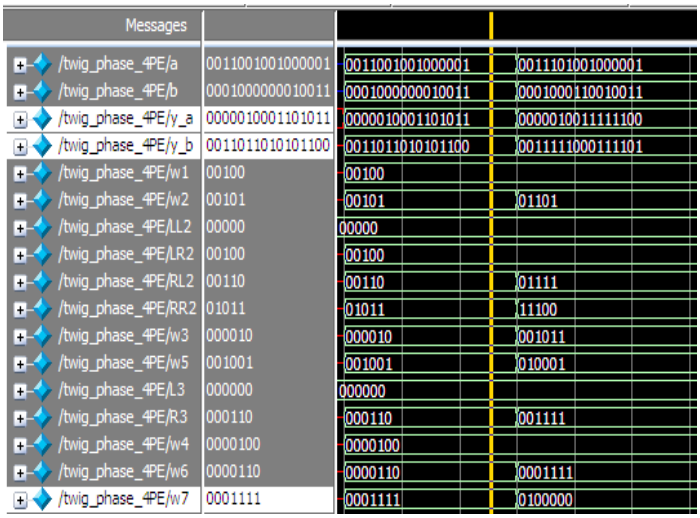


Fig.28. Simulation waveform of PE programming in twig phase

The Twig phase in Folded Tree is synthesized by using Xilinx 13.1 then it produces the synthesis report which gives the delay, memory reports as follows:

Timing Summary:
Maximum combinational path delay: 14.736ns
Total memory usage is 165276 kilobytes

5. Sixteen bit Sklansky Adder

16 bit Sklansky adder is having less delay and memory hence it is developed and used in Wallace tree structure for data aggregation in Wireless Sensor nodes. This adder adds the two sixteen bit inputs and produces a result of sum and carry out (cout) is developed by using Xilinx 13.1 then it gives a RTL schematic diagram as shown in Fig.29.

Fig.29. RTL Schematic Diagram for 16 bit Sklansky adder

This 16 bit Sklansky adder is simulated by using Modelsim XE III 6.4b then it adds the two 16 bit inputs a=725, b=6999 and produces a output of sum=7724 as shown in Fig.30.

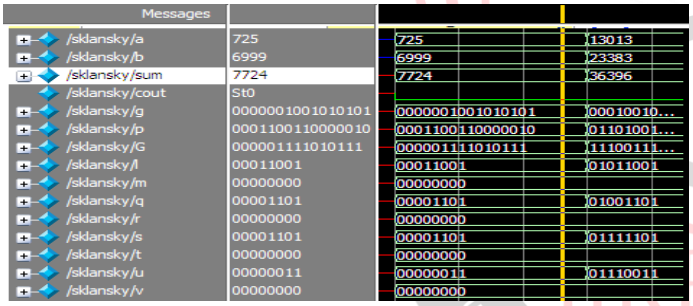


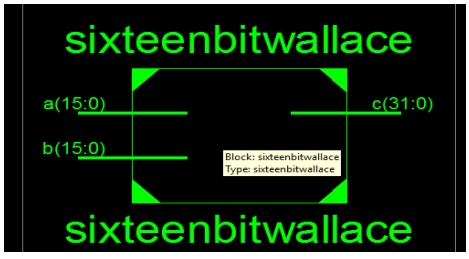
Fig.30 Simulation for 16 bit Sklansky adder

The 16 bit Sklansky adder is synthesized by using Xilinx 13.1 then it produces the synthesis report which gives the delay, memory reports as follows.

Timing Summary:
Maximum combinational path delay: 15.632ns
Total memory usage is 165148 kilobytes

6. Sixteen bit Wallace Tree with Sklansky Adder

16 bit Wallace tree by using Sklansky adder is our proposed tree architecture. This tree is developed by using Xilinx 13.1.It gives the RTL schematic diagram of 16 bit Wallace tree structure as shown in Fig.31.



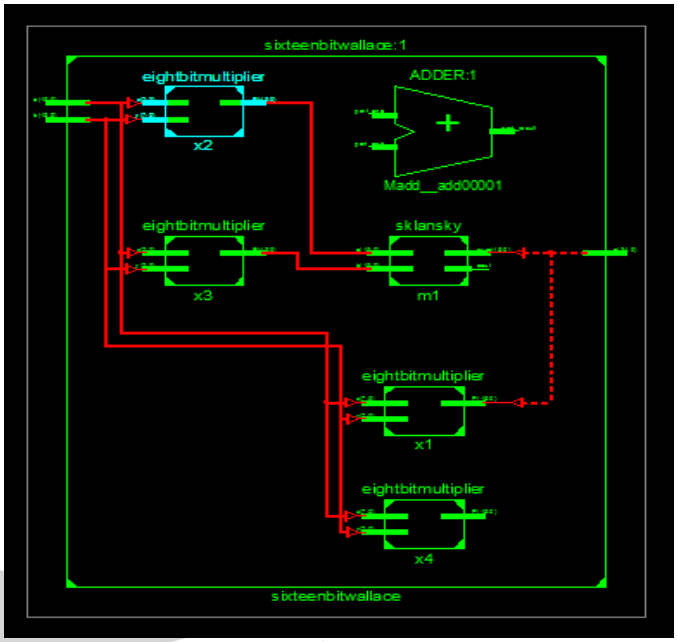


Fig.31: RTL schematic diagram for 16 bit Wallace tree with Sklansky adder

This 16 bit Wallace tree with Sklansky adder is simulated by using Modelsim XE III 6.4b then it multiples the two 16 bit inputs a=40, b=79 and produces a result of y=3160 as shown in below Fig.32.

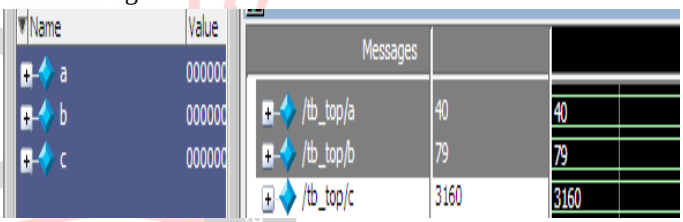


Fig.32. Simulation waveforms for 16 bit Wallace tree with Sklansky adder

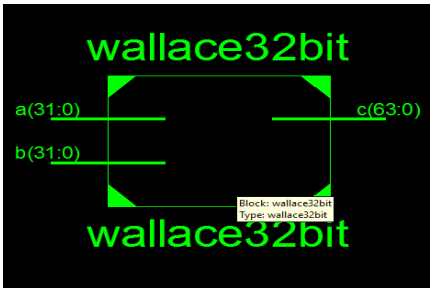
The 16 bit Wallace tree with Sklansky adder is synthesized by using Xilinx 13.1 then it produces the synthesis report which gives the delay, memory reports as follows:

Timing Summary:
Maximum combinational path delay: 23.322ns
Total memory usage is 179228 kilobytes

7. Thirty Two Bit Wallace Tree with Sklansky Adder

32 bit Wallace tree by using Sklansky adder is our proposed tree architecture. This tree is developed by using Xilinx 13.1.

It gives the RTL schematic diagram of 32 bit Wallace tree structure as shown in Fig.33.



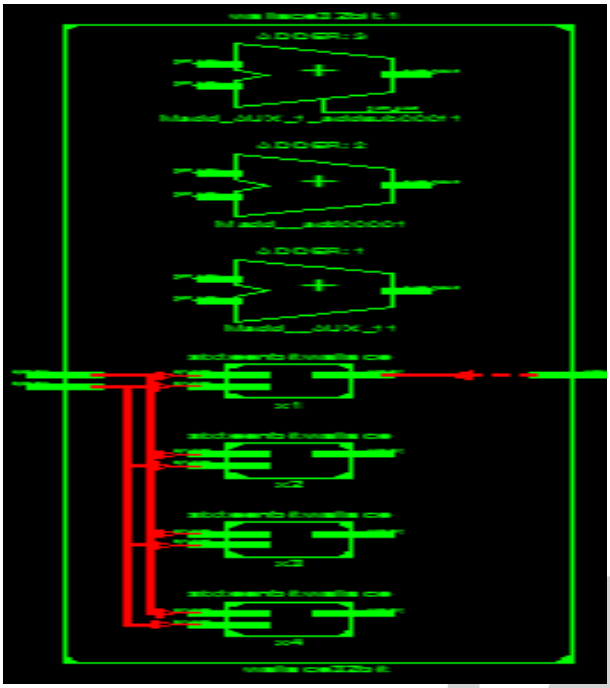


Fig.33. RTL schematic diagram for 32 bit Wallace tree with Sklansky adder

This 32 bit Wallace tree with Sklansky adder is simulated by using Modelsim XE III 6.4b then it multiples the two 16 bit inputs a=393301, b=50331669 and produces a result of y=19795495749369 as shown in below Fig.34.

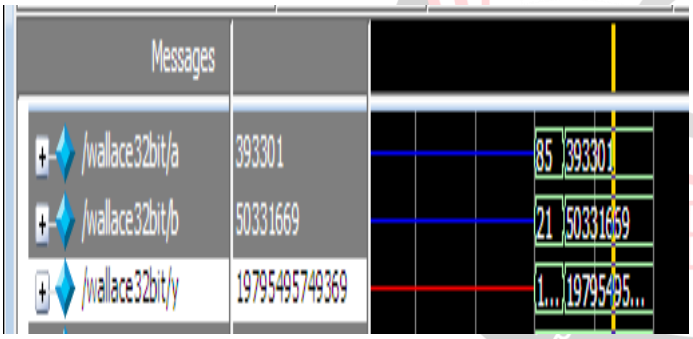


Fig.34. Simulation waveforms for 32 bit Wallace tree with Sklansky adder

The 32 bit Wallace tree with Sklansky adder is synthesized by using Xilinx 13.1 then it produces the synthesis report which gives the delay, memory reports as follows:

Timing Summary:

Maximum combinational path delay: 31.504ns

Total memory usage is 185244 kilobytes

Comparison of Results

The table 1 describes that 16 bit Wallace tree is having less delay and memory compared to 16 bit Folded tree, and also it gives the delay and Memory of a 32 bit Wallace Tree.

Table 1: Comparison of 16 bit Folded tree and 16 bit Wallace tree

Parameter	Trunk Phase in 16 bit Folded Tree	16 bit Wallace Tree in Sklansky adder	32 bit Wallace Tree in Sklansky adder
Delay (ns)	15.186	23.322	31.504
Memory (KB)	164444	179228	185244

Practical Implementation of 32 Bit Wallace tree on ZED Board:

32 bit Wallace Tree by using Sklansky adder is interfaced to ZED board by using Vivado 2014.4 software as shown in Fig.35.

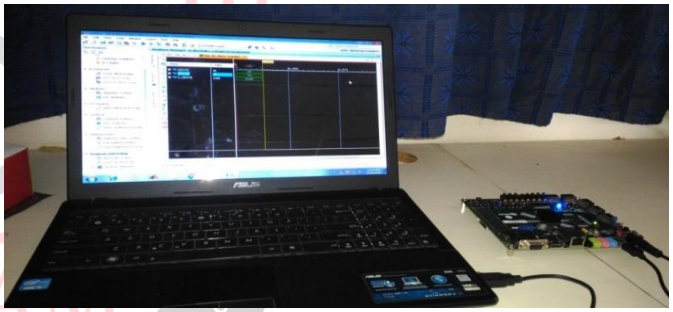


Fig.35. Interfacing of 32 bit Wallace Tree to ZED board

The 32 bit Wallace Tree by using Sklansky adder is implemented on ZED board by using Vivado 2014.4 software.Then it produces a results of this tree is as shown in Fig.36.

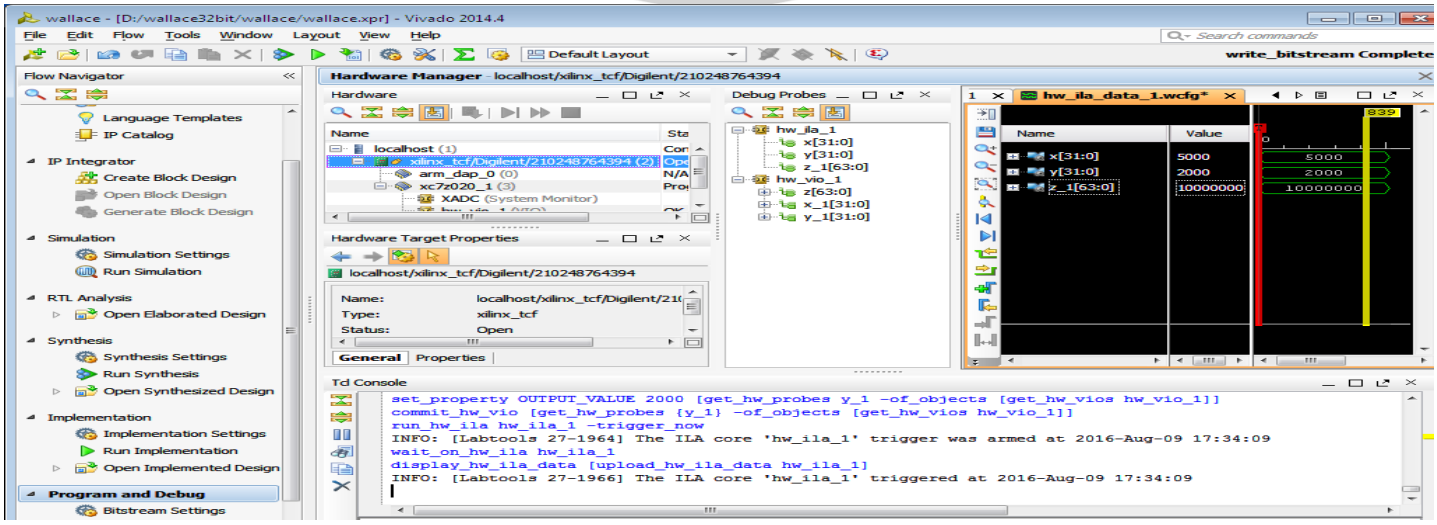


Fig.36. 32 bit Wallace Tree by using Sklansky adder implemententation

The 16 bit Wallace Tree by using Sklansky adder is implemented on ZED board by using Vivado 2014.4 software.Then it produces a output waveform as shown in Fig.37.

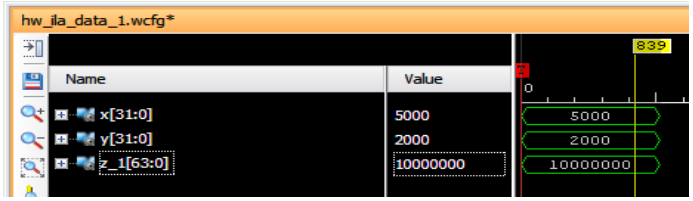


Fig.37. 32 bit Wallace Tree by using Sklansky adder implemented waveforms

Interfacing of WSN Board (IoT) to the ZED Board

A ZED board which is having the aggregated data is interfaced to the WSN Master Development Board

(IOT Board) as shown in below Fig.38. is used for Sending the aggregated information(data) which is located at ZED board to the Sink.

Fig.38. Interfacing of WSN Board (IoT) to the ZED Board

VII. CONCLUSION

A 16 bit Wallace tree structure with Sklansky adder was developed for data aggregation process in Wireless Sensor Nodes. This structure gives 23.322 ns delay and a memory of 179228 KB these values are very less compared to 16 bit Folded Tree with Prefix logic. Further, A 32 bit Wallace Tree is also developed, to increases the data aggregation capacity. It is also having very less delay and memory even though the multiplication capacity is doubled than 16 bit Wallace Tree.

The Sixteen bit and 32 bit Wallace Tree implemented by using Vivado 2014.4 on ZED development board and that aggregated information is send to the sink wirelessly by interfacing that ZED board to the Wireless Sensor Network Master Development board.

REFERENCES

[1] Cedric Walravens and Wim Dehaene, " Low Power Digital Signal Processor Architecture for Wireless Sensor Nodes," IEEE trans., on very large scale integration (VLSI) systems, Vol. 22, No. 2, pp. 313–321,Feb. 2014.

[2] V. Raghunathan et al., "Energyaware wireless microsensor networks," IEEE Signal Process. Mag., Vol. 19, No. 2, pp. 40–50, Mar. 2002.

[3] C. Walravens and W. Dehaene, "Design of a low-energy data processing architecture for wsn nodes," in Proc., Design, Automat. Test Eur. Conf. Exhibit., Vol.25, pp. 570–573,Mar. 2012.

[4] Dakupati.Ravi Sankar and Shaik Ashraf Ali " Design of Wallace Tree Multiplier by Sklansky Adder," International Journal of Engineering Research and Applications (IJERA),Vol. 3, No. 1, pp.1036-1040, Feb. 2013.

[5] N.Vinoth Kumar and S.P.Densingh Xavier "LowPower DSP Folded Tree Architecture for WSN by Using Routing Technique," International Journal of Emerging Technology and Advanced Engineering, Vol. 4, No. 3, pp. 9-16 Feb. 2014.

[6] N. Weste and D. Harris, CMOS VLSI Design: A Circuits and Systems Perspective. Reading, MA, USA, Addison Wesley, 2010.

[7] P. Sanders and J. Traff, "Parallel prefix (scan) algorithms for MPI," in Proc., Recent Adv. Parallel Virtual Mach. Message Pass. Interf., Vol. 25, No.5, 2006, pp. 49–57.

[8] G. Blelloch, "Scans as primitive parallel operations," IEEE Trans. Comput., Vol. 38, No. 11, pp. 1526–1538, Nov. 1989.

[9] H. Karl and A. Willig, Protocols and Architectures for Wireless Sensor Networks, 1st ed. New York: Wiley, 2005.

[10] Deepa Yagain et al., "Design of High-Speed Adders for Efficient Digital Design Blocks," International Scholarly Research Network ISRN Electronics,Article ID 253742,9 pages, 2012.