Multiplier Integrated Circuit Design For Digital Neuron Using 0.13µm Technology

Ahmad Ridhuwan Bin Sudin
Faculty of Electrical Engineering
Universiti Teknologi MARA Malaysia
40450 Shah Alam, Selangor, Malaysia
e-mail: achik_77wan@yahoo.com.my

Abstract - **Neurel Network is artificial intelligence system that consists of multiplication and addition process. The Neuron Network is made from lot of multiplier and adder. The objective is to design an adder and multiplier integrated circuit for neuron architecture. Comparison is done between multiplier architectures, array and booth to neuron performance. The adder that is use in the design is ripple carry adder. For array multiplier, two structure of multiplier that was built 10x5 bit multiplier and 8x8bit multiplier. For Booth multiplier the multiplier is build using verilog code in Quartus software. After that ring structure of neuron was selected to be investigated the effect of the multiplier. The performances are evaluated in terms of number of bit, power, fan-out, and timing analysis. For the data, it was found that Booth multipliers are giving the less time delay, less power, less fan-out and also less logic element. Result shows that ripple carry adder and booth multiplier have almost same power, 196.9mW. For other performances booth is give less value, which is fan-out 467, logic element 103 and timing 19.984ns. For over all study show that booth multiplier performance is better than array multiplier.**

Keywords-component; neuron; multiplier

I. INTRODUCTION

Neurel network has been used in many applications in science and engineering. Neural networks are used for applications where formal analysis would be difficult or impossible, such as pattern recognition and nonlinear system identification and control. Neurel network is a computational model that learns by training on past experience using an algorithm which modifies the interconnection weights as directed by a learning objective for a particular application [1]. The neuron network is made of from lot of multiplier and adder. The great thing about neuron network is its ability to generalize from its training vectors and learn from initially randomly distributed connection.

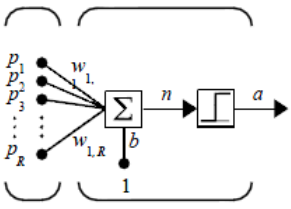

Figure 1:
a simple neuron block diagram.

Figure 1 show a simple neuron block diagram [2]. One neuron is made of from lot of multiplier and adder. P_1, P2, P3, until P_R is input signal that multiple with a weight, W in a multiplier. The output from the multiplier will add in an adder before it go to activation function. The objective is to design a adder and multiplier integrated circuit for neuron architecture for neuron. Compare between the multiplier itself and when the multiplier in neuron is also the objective. There is lots of multiplier architecture that can build. In this project two type of multiplier that is build array multiplier and Booth's multiplier. This project also to compare the multiplier architecture performance and the performance of neuron when it use the multiplier. The performance of neuron will be different if it uses the different of multiplier. The arrangement of the full adder and half adder in build a multiplier also will give different result to the output. After completed design the multiplier, we will use the ring structure neuron to investigate the performances of the neuron in power, time delay, fan-out and logic element that be use.

II. METHODOLOGY

In this project we use verilog language to design the multiplier. Originally a modeling language for a very efficient event-driven digital logic simulator Verilog is a Hardware Description Language; a textual format for describing electronic circuits and systems. Applied to electronic design, Verilog is intended to be used for verification through simulation, for timing analysis, for test analysis (testability analysis and fault grading) and for logic synthesis. In this study, there are few software that be use to make the design. There are Xilinx, Quartus and ModelSim. All the simulations are done in this software. Figure 2 show the flow chart for the multiplier design. Start from design the adder and we simulate the adder until the adder is function before proceed to design multiplier. Then the adder are use to design the multiplier. Three design architecture of multiplier are make and the multiplier be simulate. If the multiplier was function, next proceed to put the multiplier in neuron structure.

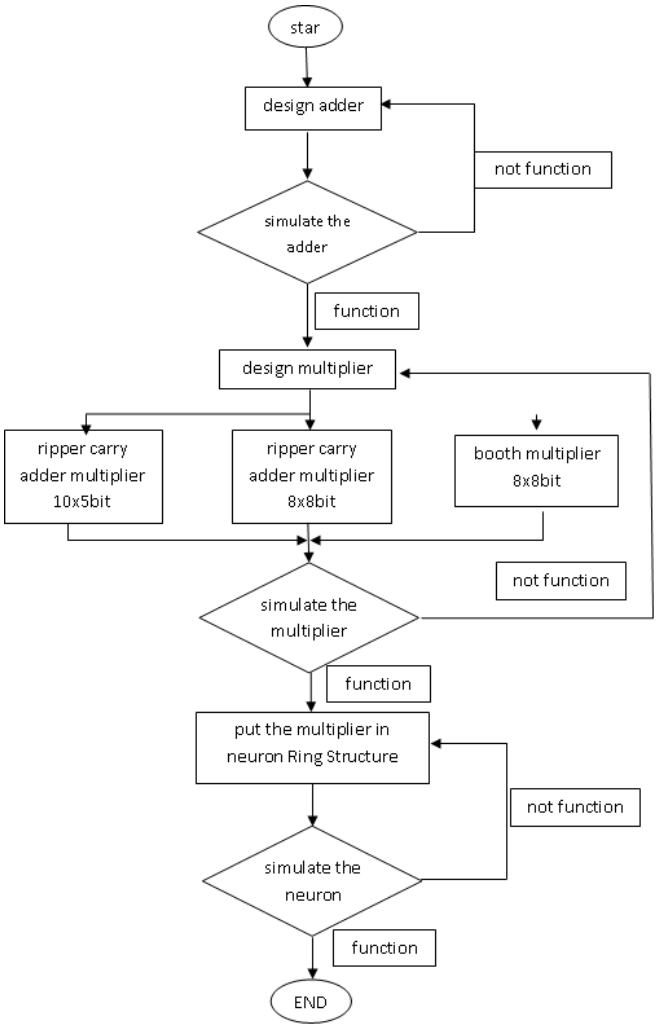

Figure 2: flow chat of multiplier design

A. *Design of Full Adder*

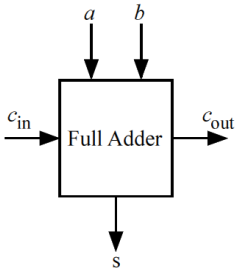

Figure 3: one bit full adder

Adder is one basic component for build the multiplier [13]. The adder that will be use in this design is Ripper adder. Figure 3 show the block diagram of one bit full adder. A one bit full adder is a combination circuit that forms the arithmetic sum of three bits. It consists of three inputs (*a*, *b*, and c_{in}) and two output (*s* and c_{out}) as illustrated in figure 3. The truth table of the full adder is in table 1. The Boolean expression for the full adder is:

$s = (a \oplus b) \oplus c$ (1)

$c = a + b + c$ (2)

Table 1: Truth table for full adder.

input			output	
a	B	Cin	Cout	s
0	0	O	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

For full adder, refer to table 1, if any one of the input are 1, and the s output will become 1. if any two of the input are 1, the Cout is 1. Then if all input is 1, then Cout and S output is 1.

B. *Design Ripple Carry Adder*

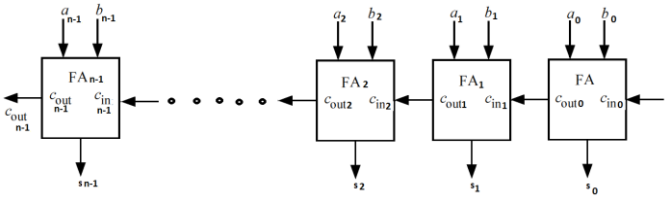

Figure 4: Example of Ripple Carry Adder.

Ripple carry adder are being design using structural modeling [3]. Full adder is instantiated in few times into a structural module to produce *n* bit ripple carry adder as in figure 4. The input *a*, *b* and C_{in} have n bit data so that need n-1 full adder to make n bit ripple carry adder. The C_{out0} of FA0 will add to FA2 as Cin. This will continues until FAn-1. *S* is the output from the adder and the most significant bit for the output is $C_{out\ n-1}$.

.

C. *Design Array Multiplier*

Multiplication involves tow operands: the multiplicand and multiplier. The product of two n-bit numbers can be accommodated in 2n bits. Multiplication of the multiplicand by 1 bit in the multiplier simply copies the multiplicand. If the multiplier bit is a 1, then the multiplicand is entered in the appropriately shifted position as a partial product to be added to other partial products to form the product. If the multiplier bit is 0, then 0s are entered as a partial product.

								a7	a6	a5	a4	a3	a2	a1	a0
								b7	b6	b5	b4	b3	b2	b1	b0
								a7b0	a6b0	a5b0	a4b0	a3b0	a2b0	a1b0	a0b0
							a7b1	a6b1	a5b1	a4b1	a3b1	a2b1	a1b1	b1a0	0
						a7b2	a6b2	a5b2	a4b2	a3b2	a2b2	a1b2	a0b2	0	0
					a7b3	a6b3	a5b3	a4b3	a3b3	a2b3	a1b3	a0b3	0	0	0
				a7b4	a6b4	a5b4	a4b4	a3b4	a2b4	a1b4	a0b4	0	0	0	0
			a7b5	a6b5	a5b5	a4b5	a3b5	a2b5	a1b5	a0b5	0	0	0	0	0
		a7b6	a6b6	a5b6	a4b6	a3b6	a2b6	a1b6	a0b6	0	0	0	0	0	0
	a7b7	a6b7	a5b7	a4b7	a3b7	a2b7	a1b7	a0b7	0	0	0	0	0	0	0
P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0

Figure 5: array multiply algorithm for 8bit operands.

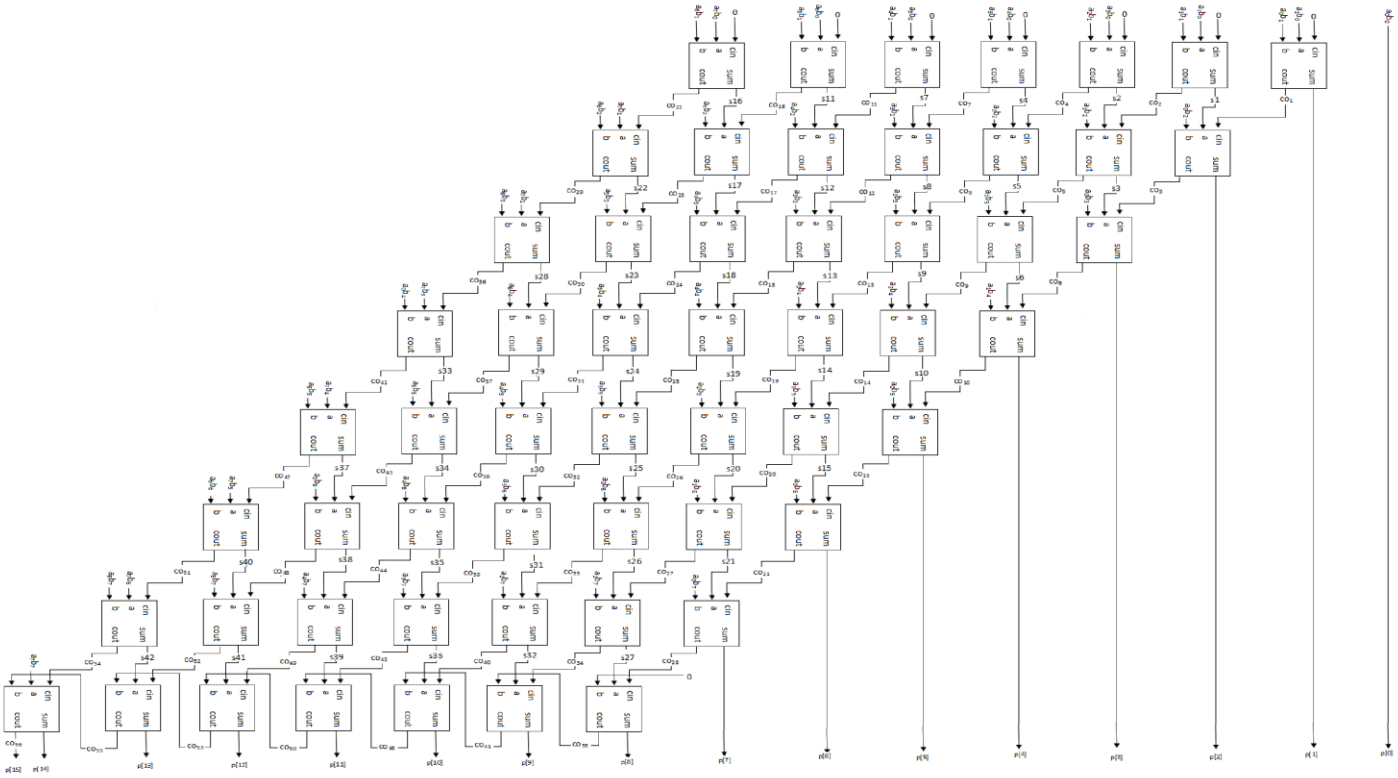

Figure 6: block diagram for 8x8 bit array multiplier.

Array multiplier is suit table to apply in any size of the multiplier operands. Figure 5 show the two 8-bit operands of array multiplier. The multiplicand is A [7:0] = a7, a6, a5, a4, a3, a2, a1, a0 and the multiplier is B [7:0] = b7, b6, b5, b4, b3, b2, b1, b0. The a0 and b0 are the low-order bits of A and B. Each bit in multiplicand is multiplied by the low-order bit b0 of the multiplier. This is same as AND functions and makes the first of 8 partial products. As the bit multiplicand is multiplied to new bit of multiplied b1, the partial product is shifted one bit position to de left. The process is repeated for other of bit multiplier until b7. The partial products are then added together to from the product. A carry-out of any column is added to next higher-order column. The equation for multiplier is same as other multiplier:

$$p = a \times b \qquad (3)$$

Figure 5 show the block diagram for the 8-bit array multiplier. The full adder is used as this multiplier are make 16bit output P [15:0] = P15, P14, until P0. The full adder was arranged of 7 in row and 8 in column as in Figure 6. For Figure 5 and Figure 6 is the multiplier 8x8 bit. The same process is done to 10x5bit array multiplier. Figure 7 and figure 8 show the algorithm and block diagram for 10x5 bit array multiplier. The total full adder was use to design the array multiplier is 52. In the figure 6, the horizontal data S15, S14 until S0 are the output for the array multiplier. Figure 7 and figure 8 showing the array multiplier algorithm and block diagram for 10x5bit array multiplier.

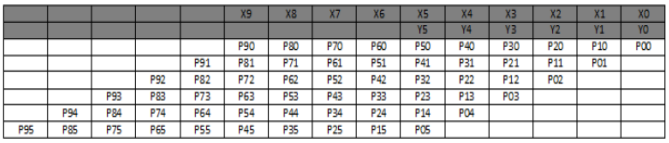

Figure 7: array multiplier algorithm for 10x5 bit.

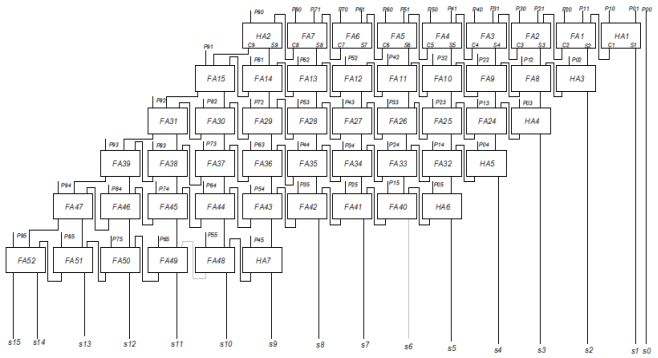

Figure 8: block diagram for 10x5 bit array multiplier.

The 10xbit design to is

D. Booth multiplier

One of the ways to multiply signed number was invented by Booth[4]. The multiplicand M, n=8 bits wide represented as M7, M6 until M0 and a Multiplier R again, n=8 bits wide represented as R7, R6 until R0. Both of these are signed (two's compliment) binary numbers. As per Booth's algorithm,

[M x R = M x $\{(S_7 x\ 2^7) + (S_6 x\ 2^6)$ ..... $(S_2 x\ 2^2) + (S_1 x\ 2^1) + (S_0 x\ 2^0)\}$ ---------- *equation (4)*]

Where each S_k for, 7<=k<=0, is a value which depends upon the value of R, and can be found as explained in the following steps. Steps 1, Append R by a '0' on Lowes significance BIT (LSB), we will called this bit as Z. Step 2 is make collections of 't' bits, where 't' = 2, for booth algorithm, and name each collection Ck, where 7<=k<=0, The rule to make each collection $\mathbf{C_k}$ is such that $\mathbf{C_k}$= (R_k R_{k-1}), if 7<=k<=1, and Ck = (R_k Z) for k= 0. This process will result in 8 collections, such that, C_7 = (R_7,R_6), ...C_1 =(R_1 R_0), C_0 =(R_0 Z).

Steps 3, Ddepending upon the value of $\mathbf{C_k}$, where 7<=k<=0, find out S_k, where the value of 'Sk' is defined in the following table for all possible combinations of values of a pair $\mathbf{C_k}$.

Table 2: the all possible combinations C_k

$\mathbf{C_k}$	s
00	**0**
01	**+1**
10	**-1**
11	**0**

equation(4) can be re-written as:
M x R = (M x p_7)+(M x p_6).....+(M x p_1)+ (M x p_0) where, $p_7 = S_7 x\ 2^7$, $p_6 = S_6 x\ 2n^6$..... $p_1 = S_1 x\ 2^1$, $p_0 = S_0 x\ 2^0$. [M x R = $pp_7 x\ 2^7 + pp_6 x\ 2^6$.....+ $pp_1 x\ 2^1 + pp_0 x\ 2^0$-------*equation (5)*].

Where pp_7 =(M x p_7),pp_6 =(M x p_6),...pp_1 =(M x p_1), pp_0 =(M x p_0) are called partial products. Steps 4, Add these 8 partial products as shown in the equation below to get so
[MxR = $pp_7 x\ 2_7 + pp_6 x\ 2_6$..... +$pp_1 x\ 2_1 + pp_0 x\ 2_0$ ----------equation(6)].

The main bottleneck in the speed of multiplication is the addition of partial products. More the number of bits the multiplier or multiplicand is composed of, more are the number of partial products, longer is the delay in calculating the product. The critical path of the multiplier depends upon the number of partial products. In booth's algorithm, if we are multiplying 2 'n' bits number, we have 'n' partial products to add. booth's multiplication is an answer to reducing the number of partial products. Using booth's multiplier, the number of partial products are reduced to 'n/2' if we are multiplying two 'n' bits numbers, if 'n' is even number, or '(n+1)/2' , if 'n' is an odd number. By reducing the number of partial products, one can effectively speed up the multiplier by a factor roughly equal to 2. Figure 9 show block diagram of booth multiplier that have two input (input a and input b) and one output (product).

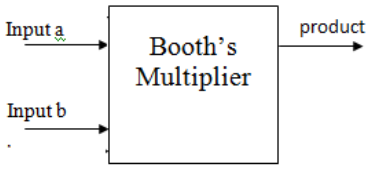

Figure 9: Block Diagram of Booth's Multiplier.

E. Ring Structure of Perceptron

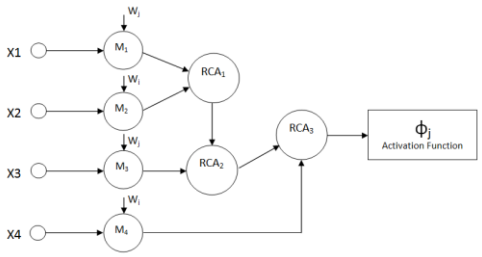

Figure 10: Ring Structure of Neuron.

Neurons have few structures that can be made. For this project, Ring Structure of neuron use to investigate the effect of multiplier to a neuron [5]. Figure 10 show the structure of neuron. In the figure 10, we use 4 multiplier 3

Ripple carry adder and one activation function. X1, X2, X3 and X4 is input signal. Each multiplier has weight to multiply with the input signal. The product from multiplier will be added in ripple carry adder. Product from M1 and M2 is added in ripple carry adder1. Then the sum from ripple carry adder1 we be add in ripple carry adder2 with product M3. Then the sum from ripple carry adder2 will add to ripple carry adder3 with product of X4 and the output from ripple carry adder3 go to Activication Function.

F. Digital Layout Design Flow

The custom design for the neuron will be based on Register Transfer Level in Quartus II software post synthesis simulation. The structural design will implement the hierarchical gates. The purpose of using the same architecture or repetition blocks in IC layout design will give same effect of the layout performance. On the other words, we could reduce the gradient effect and parasitic capacitance while use this technique [6]. Hence this is the best way to come out the layout for a neuron. The NAND gate built from AND gates by adding inverter. Then, Exclusive-OR and NAND connected to form Full Adder. Full Adder, AND and NAND gates combine to form Multiplier. The Ripple Carry Adder is constructed by cascading full adder blocks in series. The combination of Multipliers and ripple carry adder will form Multiplier Accumulator (MAC) [7]. Finally, a single neuron is formed by MAC and activation function. In material state, nMOS operates in electrons base and it faster than pMOS in same width size. Transistor dimensions specified as width, W and length, L. As a base to determine the sizing of design, the minimum size is 4λ /2λ, and this sometimes called 1 unit [8].

$$pMOS = 8/2 \quad (1)$$

$$nMOS = 4/2 \quad (2)$$

Hence, the base ratio for pMOS and nMOS are 2:1 respectively. We use ratio 2:1 for the base size in this neuron custom IC layout design. Base size is the starting size in the determination of the lower hierarchy of the design [9].

III. RESULT AND DISCUSSION

The simulation of Verilog of Single Layer Feed-forward Network is run in Modelsim-Altera software for the functionality test. Then, we run the verilog by using Quartus-Altera software to perform analysis.

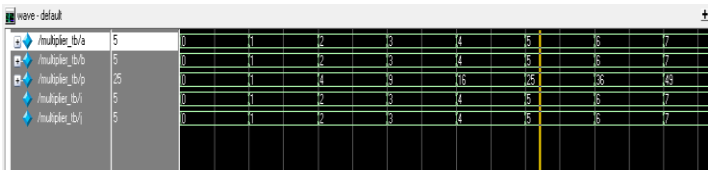

Figure 11: the timing diagram of Array Shifts-add multiplier.

From the timing diagram above in figure 11, the multiplier that we design make same output as the calculation we done. Example, we read data from the yellow line in figure. Input 1 or input a is 5 and input 2 or input b is 5. The product or the output p is 25. So this is evidence that the multiplier we design is function.

A. Data for a single multiplier

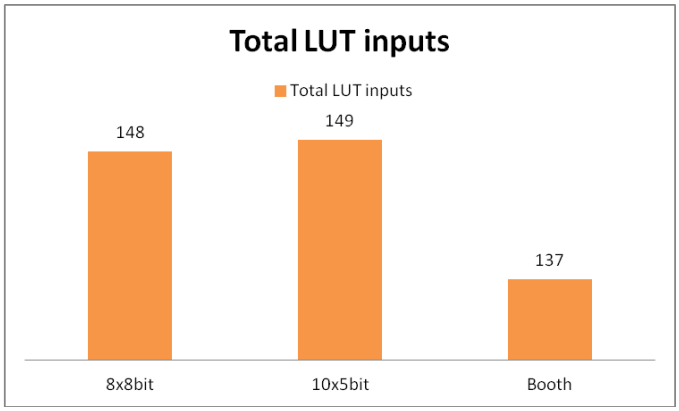

Figure 12: the bar graph for logic elements.

By using Quartus II software simulation, the total LUT input is showing that booth multiplier has the least logic elements in design because the synthesizer minimize the multiplication process. 10x5bit array multiplier shows the most usage of logic element due to the structure consist of many repetition of full adders. As a result it layout will become larger with increasing of bits. On the other word the 8x8bit array multiplier's logic element is smaller than 10x5bit array multiplier because it has smaller bits.

Logic elements by mode	**8x8bit**	**10x5bit**	**Booth**
Total LUT inputs	**582**	**667**	**540**

Table 3: Number of total logic elements used by mode

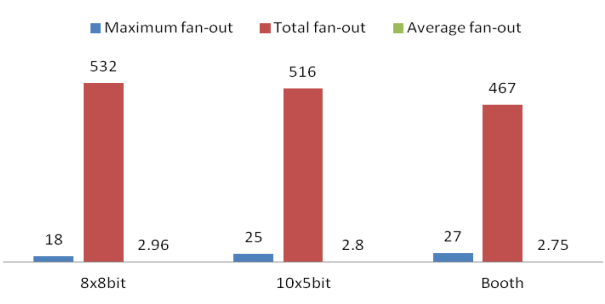

Figure 13: the no of Fan out for the multiplier.

In digital electronics, the **fan-out** of a logic gate output is the number of gate inputs to which it is connected. In most designs, logic gates are connected together to form more complex circuits. While no more than one logic gate output is connected to any single input, it is common for one output to be connected to several inputs. The data in figure 13, booth have the lowest total fan out that is 467fan-out. 8x8bit 10x5bit array multipliers have 532 and 516. It is because booth multiplier has less logic gate at output. Output data Booth multiplier is from 5 adders but for array multiplier, it needs 16unit of full adder to get the output. Fan-out is depending to the number of the logic gate. Table 5 is the data for multiplier in neuron structure. When the multipliers apply in ring structure of neuron the total fan-out for booth multiplier in neuron was increase about 72.56% from 467 up to 1702. For 10x5bit array multiplier, it was 77.30% from 516 to 2274 and for 8x8bit array multiplier is 73.17% that it increases from 532 to 1983.

Fan-out	8x8bit	10x5bit	Booth
Maximum fan-out	66	88	73
Total fan-out	1983	2274	1702

Table 5: Fan-out by each neuron.

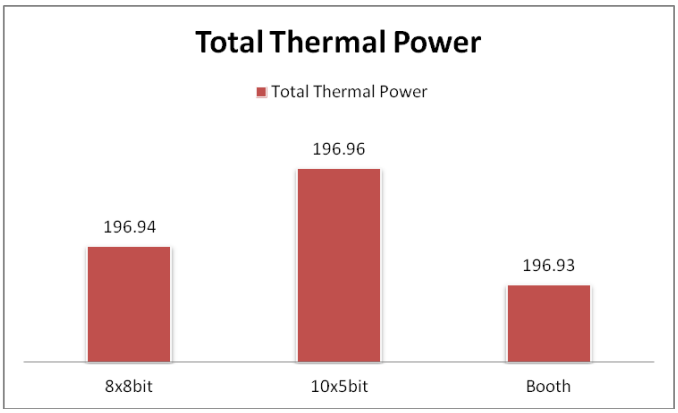

Figure 14: power from multiplier.

Power is needed in all instruments to operate. A good system is when the power is less. From figure 14, the booth multiplier has smaller total thermal power at 196.93mW. This is because the multipliers use same voltage souse and same current to operate. Comparing the data in figure 14 with table 4, it looks show that the power was increases. For booth multiplier the power increases by 2.16%. The 10x5bit and 8x8bit array multiplier also increase about 2.15% and 2.20%. 8x8bit array multiplier is the most increasing power if we compare to other multiplier when it apply in neuron.

Power , mW	8x8bit	10x5bit	Booth
Total Thermal Power	201.38	201.29	201.27

Table 4: Power of neuron.

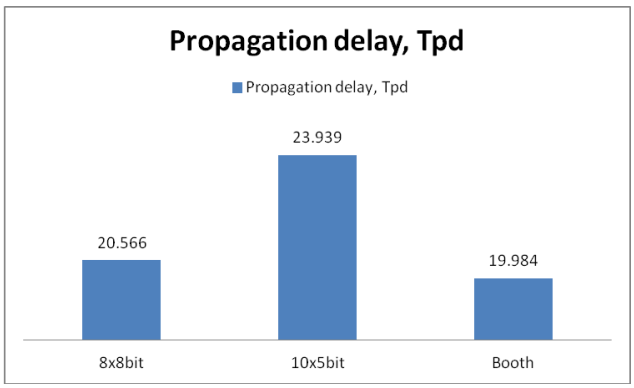

figure15: timing analysis in multiplier.

Structures of multiplier were composed from structural based that combination from lot of basic gates. Timing analysis measures the delay of every design path and report the performance of the design in terms of the maximum clock speed. The timing analysis in Quatus II software was based on structural behavior [10]. Booth multiplier much faster than ripple carry adder. Booth multipliers do not carry the output to nest adder, but the carry out of adder in booth is direct to the output. The table 6 is the timing delay data for a neuron. The time delay is increasing in neuron if compare to multiplier. The most increase in timing delay is booth multiplier, it about 50.32% increasing. The less increase is 10x5bit array multiplier

Timing, ns	8x8bit	10x5bit	Booth
Propagation delay, T_{pd}	38.430	57.261	40.223

Table 6: Timing analysis by each neuron

B. Layout

In this project, the integrated circuit (IC) layout design of array multiplier 8x8 bit was design by using Silterra 0.13 micrometer. This layout design has to pass Design Rule Check (DRC) and have to match the Layout Versus Schematic (LVS). Design layout is the representation of IC a-in terms of planar geometric shape. The array multiplier layout design is like in figure 15. In this layout, there are build from 56 of full adder and 58 of AND gate. As the figure 15, the width of the multiplier is 428µm and the length is 311.50µm. so the area of this multiplier is142147.2µm^2 or 142.15mm^2. At first step, make the basis layout that is nmos and pmos. From nmos and pmos layout, logic AND, XOR, inverter and other logic gate

can be build using same basis gate but need to change the width and length. Then from the logic gate, full adder can build. Last we make more adder and AND gate to make a multiplier. There are many ways to arrange the full adder to make a multiplier. As show in figure 15, in the yellow circle is the full adder. In white circle there are empty spaces between the adders. These empty spaces will affect the performance of multiplier. This will make the layout have parasitic effect that caused the connection between wires and blocks. The empty spaces have to fill with dummy. The different type of layout has different electrical effects [11]. Every different layout has to be separated to minimize the parasitic capacitance and resistance [12].

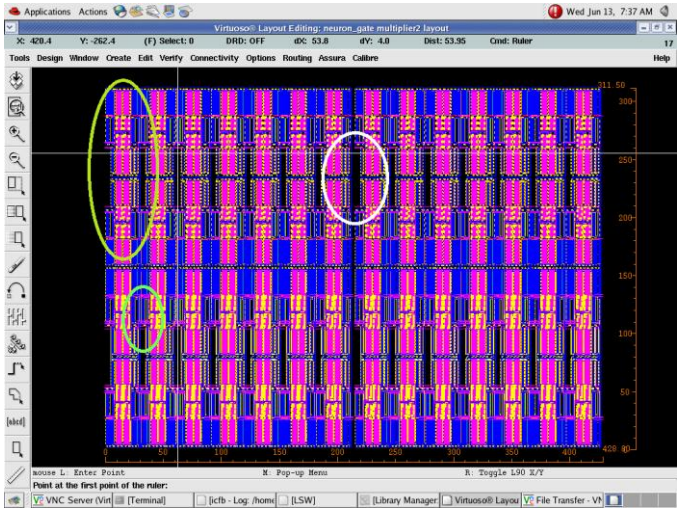

Figure 15: Layout of array multiplier.

IV. CONCLUTION

This paper was presented high reliability of multiplier in any design. The number of resources usage for booth multiplier is 137 and 540 when it applies in neuron. For array multiplier 8x8bit and 10x5bit, the resources be use are 148 and 149. When array multiplier 8x8bit and 10x5bit apply in neuron the resource is 582 and 667. The booth multipliers have small timing propagation 19.984ns, but when it applies in neuron 8x8bit array multiplier are 38.430ns comparing to booth multiplier 40.223ns. For power it give same power value that is 196.96mW as multiplier and 201.38mW when apply to neuron. We were successful in build 8x8bit array multiplier layout with value of area are 142.15mm^2. It was found that booth multiplier have less time delay, lest power, lest fan-out and lest logic element. For the future, carry lookahead adder to build a multiplier. So we can make the multiplier faster.

ACKNOWLEDGMENT

This paper participates in the Innovate Design Malaysia competition 2012 and has been selected for the top five finalists in the Silterra IC design platform. The authors would like to thank Dr. Wan Fazlida Hanim bte Abdullah for being supervisor in this final year project.

REFERENCES

[1] Simon S. Haykin, “Neural Networks: A Comprehensive Foundation”, Prentice-Hall, Englewood Cliffs, New Jersy, (1999), Chapter 1-15, pp 1-889.

[2] Martin T. Hagan, Howard B. Demuth and Mark Hudson Beale, “Neural Network Design”, (1996), PWS Publ. Company, Chapter 1-19, page 1-1 to 19-14.

[3] Khairudin Mohamad, “Design of Neuron Architecture on FPGA for Electrochemical Sensors Signals”, Universiti Technology Mara Malaysia, 2010.

[4] Shi X

[5] Shi Xia, Lu Qi-Shao, “Complete Synchronization of Coupled Hindmarsh-Rose Neurons with Ring Structure”, IEEE, School of science, Beijing University of Aeronautics and Astronautics, Beijing. Jun 2005.

[6] CEDEC’s IC Design Manuals, ”Innovate Malaysia 2012”, February 2012.

[7] Khairudin Mohamad, “Design of Neuron Architecture on FPGA for Electrochemical Sensors Signals”, Universiti Technology Mara Malaysia, 2010.

[8] Neil H.E. Weste, David Money Harris,”Integrated Circuit Design”, Fourth Edition, Chapter 1 Welcome to VLSI.

[9] Neil H.E. Weste, David Money Harris,”Integrated Circuit Design”, Fourth Edition, Chapter 8 Gates.

[10] Z. Abid, H. El-Razouk, D.A. El-Dib, “Low Power Multipliers Based On New Hybrid Full Adders”, Department of Electrical and Computer Engineering, University of Western Ontario, London, Ontario, CA 2008.

[11] CEDEC’s IC Design Manuals, ”Innovate Malaysia 2012”, February 2012.

[12] Neil H.E. Weste, David Money Harris,”Integrated Circuit Design”, Fourth Edition, Chapter 1 Welcome to VLSI.

[13] Joseph Cavanagh, “Verilog HDL Digital Design and Modelling”, Chapter 2 Overview, 2007.