ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 3, March 2017

Effective Improvement of Carry save Adder

K.Nandini[1], A.Padmavathi[1], K.Pavithra[1], M.Selva Priya[1], Dr. P. Nithiyanantham[2]

[1]UG scholars, Department of Electronics and Communication Engineering, Jay Shriram Group of Institutions, Tirupur, Tamilnadu, India

[2]Head of the Department, Department of Electronics and Communication Engineering, Jay Shriram Group of Institutions, Tirupur, Tamilnadu, India

ABSTRACT: CSA (Carry Save Adder) is a type of digital adder used in computer architecture to the computer sum of three or more n-bit number in binary. It differ from other digital adders in that is outputs two numbers of the same dimensions as the inputs one which is a sequence of partial sum bits and another which is a sequence of carry bits. CSA is used to implement the adder circuit. In this paper, Look up Table and Booth Encoder techniques are used. LUT is for speed of calculation, which is an array that replaces runtime computation with a simpler array indexing operation. The savings in terms of processing time can be significant, since retrieving a value from memory is after faster than undergoing an "Expensive" computation operation. Booth Encoder is used to or input, output reduces the number of partial products. These techniques are used to reduce the power, time delay and area by using of modelsim software. We can attain 30% speed improvement and 22% power reduction in Design units of DSP/multimedia Applications.

KEYWORDS*:* Carry Save Adder, Lookup Table, Modified Booth Encoder techniques, Flip-Flop, Multiplier.

I. INTRODUCTION

Power dissipation is recognized as a critical parameter in modern VLSI design field. To satisfy MOORE'S law and to produce consumer electronics goods with more backup and less weight, low power VLSI design is necessary. Dynamic power dissipation which is the major part of total power dissipation is due to the charging and discharging capacitance in the circuit. The golden formula for calculation of dynamic power dissipation is Pd = CLV2f. Power reduction can be achieved by various manners. They are reduction of output Capacitance CL,reduction of power supply voltage V, reduction of switching activity and clock frequency f. In this section we introduced the above three technologies to encounter the unnecessary power dissipation problems CSA is mostly adopted in Multiplier circuits. Modified Booth Encoding is adopted in Multiplier and VMFU. The method of CSA is best understood by applying it to Multipliers. Fast multipliers are essential parts of digital signal processing systems. The speed of multiply operation is of great importance in digital signal processing as well as in the general purpose processors today, especially since the media processing took off. In the past multiplication was generally implemented via a sequence of addition, subtraction, and shift operations. Multiplication can be considered as a series of repeated additions. The number to be added is the multiplicand, the number of times that it is added is the multiplier, and the result is the product. Each step of addition generates a partial product. In most computers, the operand usually contains the same number of bits. When the operands are interpreted as integers, the product is generally twice the length of operands in order to preserve the information content. This repeated addition method that is suggested by the arithmetic definition is slow that it is almost always replaced by an algorithm that makes use of positional representation. It is possible to decompose multipliers into two parts. The first part is dedicated to the generation of partial products, and the second one collects and adds them. The basic multiplication principle is twofold i.e, evaluation of partial products and accumulation of the shifted partial products. It is performed by the successive additions of the columns of the shifted partial product matrix. The multiplier is successfully shifted and gates the appropriate bit of the multiplicand. The delayed, gated instance of the multiplicand must all be in the same column of the shifted partial product matrix. They are then added to form the product bit for the particular form. Multiplication is therefore a multi operand operation.

II. CARRY SAVE ADDER

1. Adding Multiple Numbers

There are many cases where it is desired to add more than two numbers together. The straightforward way of adding together m numbers (all n bits wide) is to add the first two, then add that sum to the next, and so on. This requires a total of $m - 1$ addition, for a total gate delay of $O(m\lg n)$ (assuming look ahead carry adders). Instead, a tree of adders can be formed, taking only $O(\lg m \cdot \lg n)$ gate delays. Using carry save addition, the delay can be reduced further still. The idea is to take 3 numbers that we want to add together, $x + y + z$, and convert it into 2 numbers $c + s$ such that $x + y + z = c + s$, and do this in O(1) time. The reason why addition cannot be performed in $O(1)$ time is because the carry information must be propagated. In carry save addition, we refrain from directly passing on the carry information until the very last step. We will first illustrate the general concept with a base 10 example.

To add three numbers by hand, we typically align the three operands, and then proceed column by column in the Same Fashion that we perform addition with two numbers. The three digits in a row are added, and any overflow goes into the next column. Observe that when there is some non-zero carry, we are really adding four digits (the digits of x,y and z, plus the carry).

```
Carry:      1   1   2   1
   x :      1   2   3   4   5
   y :      3   8   1   7   2
   z : +    2   0   5   8   7
---------------------------------
 sum:       7   1   1   0   4
```

The carry save approach breaks this process down into two steps. The first is to compute the sum ignoring any carries:

```
 x:         1   2   3   4   5
 y:         3   8   1   7   2
 z: +       2   0   5   8   7
---------------------------------
 s:         6   0   9   9   4
```

Each s_i is equal to the sum of $x_i + y_i + z_i$ modulo 10. Now, separately, we can compute the carry on a column by column basis:

```
 x:         1   2   3   4   5
 y:         3   8   1   7   2
 z: +       2   0   5   8   7
---------------------------------
 c          1   0   1   1
```

In this case, each c_i is the sum of the bits from the previous column divided by 10 (ignoring any remainder). Another way to look at it is that any carry over from one column gets put into the next column. Now, we can add together c and s, and we'll verify that it indeed is equal to $x + y + z$:

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 3, March 2017

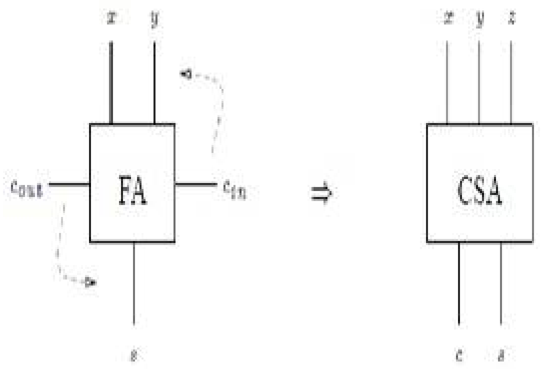
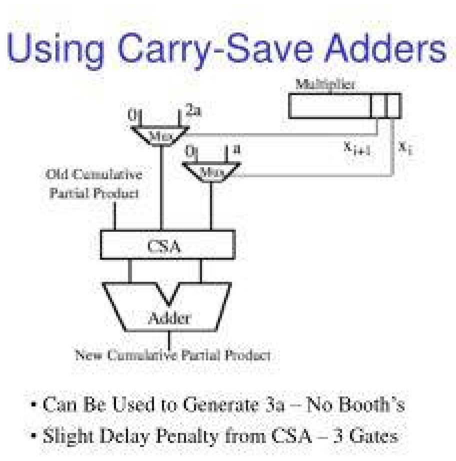


Figure 1: The carry save adder block is the same circuit as the full adder.


Using Carry-Save Adders
Multiplier
0
2a
Mux
0
a
Mux
x_{i+1}
x_i
Old Cumulative Partial Product
CSA
Adder
New Cumulative Partial Product
• Can Be Used to Generate 3a – No Booth's
• Slight Delay Penalty from CSA – 3 Gates

One CSA block is used for each bit. This circuit adds three n = 8 bit numbers together into two new numbers.

s:	6	0	9	9	4
c: +	1	0	1	1	
sum:	7	1	1	0	4

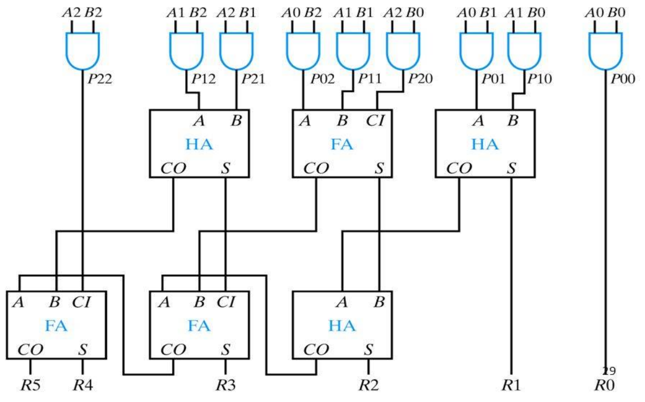
The important point is that c and s can be computed independently, and furthermore, each c_i (and s_i) can be computed independently from all of the other c's (and s's). This achieves our original goal of converting three numbers that we wish to add into two numbers that add up to the same sum, and in O(1) time.

The same concept can be applied to binary numbers. As a quick example:

x :		1	0	0	1	1
y :		1	1	0	0	1
z : +		0	1	0	1	1
s :		*0*	*0*	*0*	*0*	*1*
c : +	1	1	0	1	1	
sum :	1	1	0	1	1	1

What does the circuit to compute s and c look like? It is actually identical to the full adder, but with some of the signals renamed.
Figure 1 shows a full adder and a carry save adder. A carry save adder simply is a full adder with the c_{in} input renamed to z, the z
output (the original "answer" output) renamed to s, and the c_{out} output renamed to c. Figure 2 shows how n carry save adders are arranged to add three n bit numbers x,y and z into two numbers c and s. Note that the CSA block in bit position zero generates c_1, not c_0. Similar to the least significant column when adding numbers by hand (the "blank"), c_0 is equal to zero. Note that all of the CSA blocks are independent, thus the entire circuit takes only O(1) time. To get the final sum, we still need a LCA, which will cost us O(lg n) delay. The asymptotic gate delay to add three n-bit numbers is thus the same as adding only two n-bit numbers.So how long does it take us to add m different n-bit numbers together? The simple approach is just to repeat this trick approximately m times over. This is illustrated in Figure 3. There are m−2 CSA blocks (each block in the figure actually represents many one-bit CSA blocks in parallel) that we have to go through, and then the final LCA. Note that every time we pass through a CSA block, our number increases in size by one bit. Therefore, the numbers that go to the LCA will be at most n + m − 2 bits long. So the final LCA will have a gate delay of O(lg (n + m)). Therefore the total gate delay is O(m + lg (n + m)) Instead of arranging the CSA blocks in a chain, a tree formation can actually be used. This is slightly awkward because of the odd ratio of 3 to 2. Figure 4 shows how to build a tree of CSAs. This circuit is called a Wallace tree. The depth of the tree is $\log_{32} m$.

Multiplier Using Carry Save Adder



ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 3, March 2017

Like before, the width of the numbers will increase as we get deeper in the tree. At the end of the tree, the numbers will be O(n+logm) bits wide, and therefore the LCA will have a O(lg (n + logm)) gate delay. The total gate delay of the Wallace tree is thus O(logm + lg (n + logm)).

III. MODIFIED BOOTH ENCODER

Modified Booth Algorithm

Generally, we perform many mathematical operations in our daily life such as addition, subtraction, multiplication, division, and so on. Let us consider the multiplication process that can be performed in different methods. Different types of algorithms can be used to perform multiplication like grid multiplication method, long multiplication, lattice multiplication, peasant or binary multiplication, and so on. Binary multiplication is usually performed in digital electronics by using an electronic circuit called as binary multiplier. These binary multipliers are implemented using different computer arithmetic techniques. Booth multiplier that works based on booth algorithm is one of the most frequently used binary multipliers.

Booth Algorithm

Booth multiplication algorithm or Booth algorithm was named after the inventor Andrew Donald Booth. It can be defined as an algorithm or method of multiplying binary numbers in two's complement notation. It is a simple method to multiply binary numbers in which multiplication is performed with repeated addition operations by following the booth algorithm. Again this booth algorithm for multiplication operation is further modified and hence, named as modified booth algorithm.

Modified Booth Algorithm

Booth multiplication algorithm consists of three major steps as shown in the structure of booth algorithm figure that includes generation of partial product called as recoding, reducing the partial product in two rows, and addition that gives final product.
For a better understanding of modified booth algorithm & for multiplication, we must know about each block of booth algorithm for multiplication process.

Modified Booth Algorithm Encoder

This modified booth multiplier is used to perform high-speed multiplications using modified booth algorithm. This modified booth multiplier's computation time and the logarithm of the word length of operands are proportional to each other. We can reduce half the number of partial product. Radix-4 booth algorithm used here increases the speed of multiplier and reduces the area of multiplier circuit. In this algorithm, every second column is taken and multiplied by 0 or +1 or +2 or -1 or -2 instead of multiplying with 0 or 1 after shifting and adding of
every column of the booth multiplier. Thus, half of the partial product can be reduced using this booth algorithm. Based on the multiplier bits, the process of encoding the multiplicand is performed by radix-4 booth encoder.
The overlapping is used for comparing three bits at a time. This grouping is started from least significant bit (LSB), in which only two bits of the booth multiplier are used by the first block and a zero is assumed as third bit as shown in the figure.

Bit Pairing as per Booth Recoding

The figure shows the functional operation of the radix-4 booth encoder that consists of eight different types of states. The outcomes or multiplication of multiplicand with 0, -1, and -2 are consecutively obtained during these eight states.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 3, March 2017

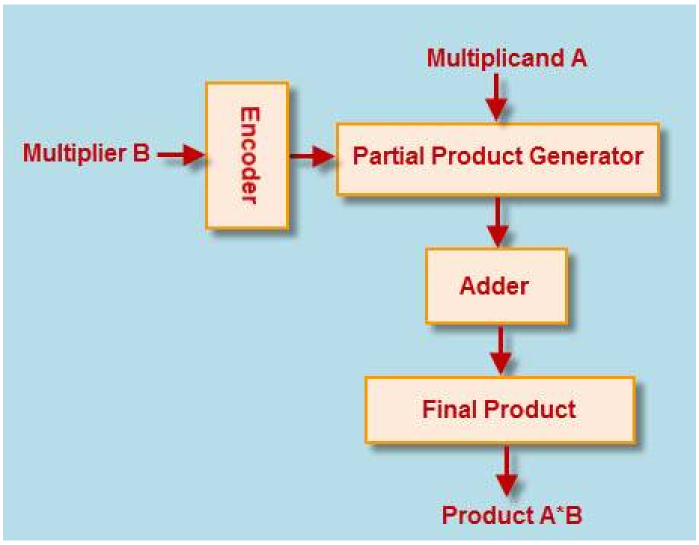


Figure 3.1 Modified booth encoder

Booth Recoding Table for Radix-4

The steps given below represent the radix-4 booth algorithm:

- Extend the sign bit 1 position if necessary to ensure that n is even.
- Append a 0 to the right of the least significant bit of the booth multiplier.
- According to the value of each vector, each partial product will be 0, +y, -y, +2y or -2y.

TABLE 1 Radix-4 Booth encoding

Inputs(bits of M-bit multiplier)			Partial product
x(1)	x(0)	x(-1)	$PP0_i$
0	0	0	0
0	0	1	Y
0	1	0	Y
0	1	1	2Y
1	0	0	-2Y
1	0	1	-Y
1	1	0	-Y
1	1	1	0

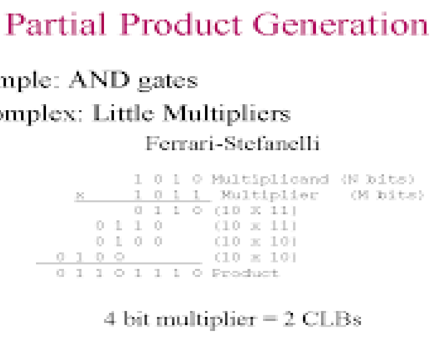
Booth's Encoder

Modified booth multiplier's (Z) digits can be defined with the following equation:
Zj = q2j + q2j-1 -2q2j+1 with q-1 = 0

The figure shows, the modified booth algorithm encoder circuit. Now, the product of any digit of Z with multiplicand Y may be -2y, -y, 0, y,2y. But, by performing left shift operation at partial products generation stage, 2y may be generated. By taking 1's complement to this 2y, negation is done, and then one is added in appropriate 4-2 compressor. One booth encoder shown in the figure generates three output signals by taking three consecutive bit inputs so as to represent all five possibilities -2X, -X, 0, X, 2X.

Partial Product Generator

If we take the partial product as -2y, -y, 0, y, 2y then, we have to modify the general partial product generator. Now, every partial product point consists of two inputs (consecutive bits) from multiplicand and, based on the requirement, the output will be generated and its complements also generated in case if required. The figure shows the partial product generator circuit.



Partial Product Generation

- Simple: AND gates
- Complex: Little Multipliers

Ferrari-Stefanelli

```
              1 0 1 0 Multiplicand (N bits)
   x          1 0 1 1  Multiplier    (M bits)
              0 1 1 0 (10 X 11)
          0 1 1 0     (10 x 11)
          0 1 0 0     (10 x 10)
      0 1 0 0         (10 x 10)
      0 1 1 0 1 1 1 0 Product
```

4 bit multiplier = 2 CLBs

generated and its complements also generated in case if required. The figure shows the partial product generator circuit. The 2's complement is taken for negative values of y. There are different types of adders such as conventional adders, ripple-carry adders, carry-look-ahead adders, and carry select adders. The carry select adders (CSLA) and carry-look-ahead adders are considered as fastest adders and are frequently used. The multiplication of y is done by after performing shift operation on y – that is – y is shifted to the left by one bit.
Hence, to design n-bit parallel multipliers only n2 partial products are generated by using booth algorithm. Thus, the propagation delay to run circuit, complexity of the circuit, and power consumption can be reduced. A simple practical example to understand modified booth algorithm is shown in the figure below.

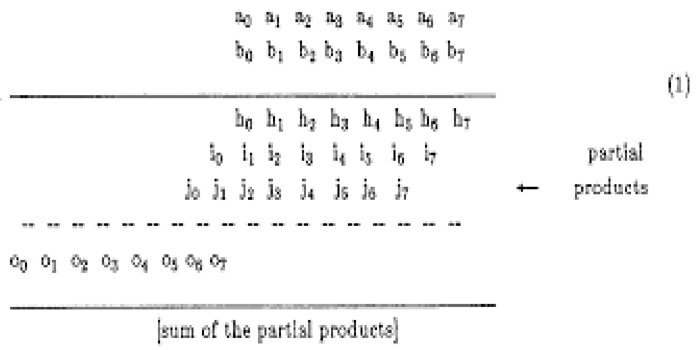
ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

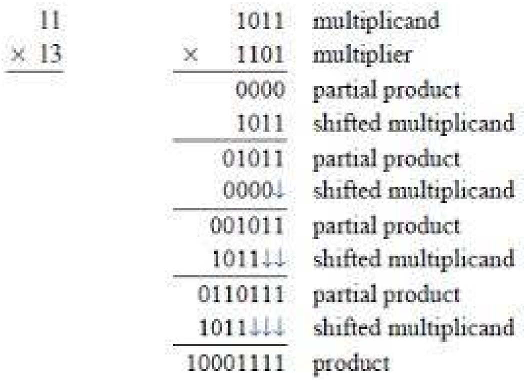
(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 3, March 2017



Practical Multiplication Example using Modified Booth Algorithm. For more technical help regarding modified booth algorithm and also for designing electrical and electronics projects, you can approach us by posting your comments in the comments section below.



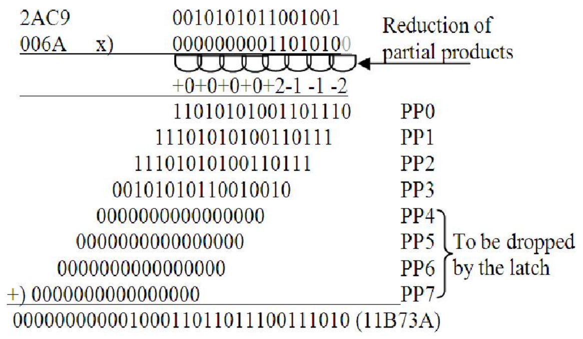


Fig. (6): Illustration of multiplication using modified Booth encoding.

The modified booth encoder is used to generate the partial products and it also used to reduce the number of partial products.

IV. LOOK UP TABLE

Concept:

The Look-up Table (LUT) is a common concept used to reduce processing time for applications that uses complex calculations. Basically, the LUT contains data or results from the complex calculations needed by application, which was done beforehand—once. By keeping the results in the LUT, when the application needs the values, instead of having to do the calculations, it can just refer to the LUT and retrieve the values from it; bypassing the calculations. In complex applications such as signal processing, image processing, device modeling, etc., complex calculations are used repeatedly and using the LUT help tremendously by significantly reducing the processing time.

A simple analogy: when we were in elementary school, we used to memorize multiplications of small numbers. In time, for example, we can say that 4 x 4 is 16, without having to calculate 4+4+4+4 (of course by first learning that to count 4 x 4 is by adding 4 four times). Saves a lot of time, doesn't it?

Application:

Currently, the LUT is intended to be used in a robotic movement control system. Again, the objective here is time; in this case, increasing the robot's response speed.

To illustrate, consider the following scenario. A RoboSoccer robot is able to determine a strategy among several strategies given a condition in the field; such as: enemy position, its current position, the goal position, the ball position and trajectory (if it is moving), etc. Once it evaluates the field condition, then the robot needs to select which is the best strategy to reach its objective (i.e. reach the ball in the most efficient path while avoiding the opponent), then it will formulate its path, do the calculations and sequence of movements, then execute it.

MULTIPLIER

A MUX is a digital switch that has multiple inputs (sources) and a single output (destination).

• The select lines determine which input is connected to the output.

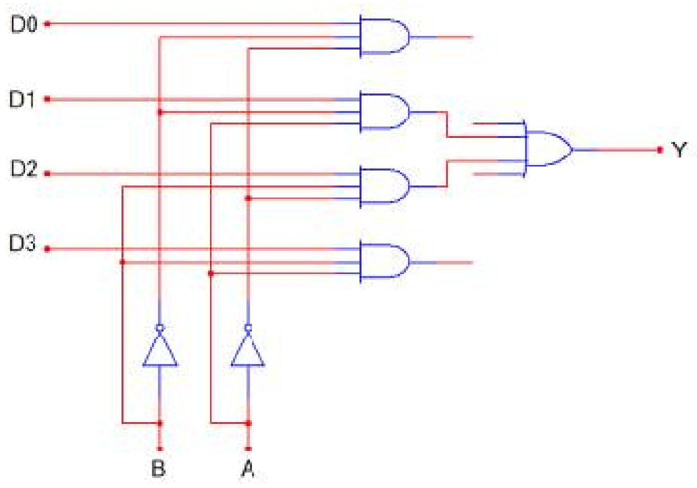


Figure 5.1 4-to-1 multiplexer (mux)

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

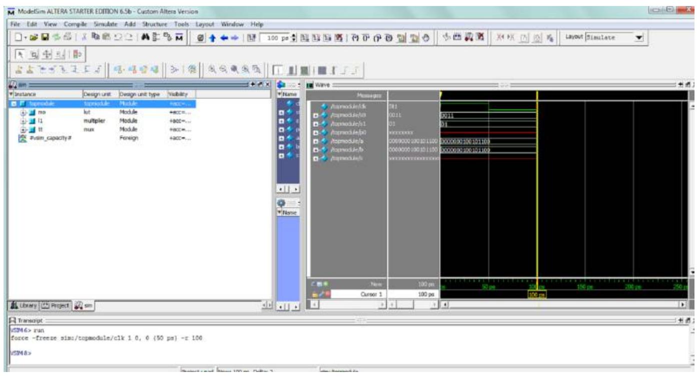
International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 3, March 2017

The result of final multiplication and accumulation process1

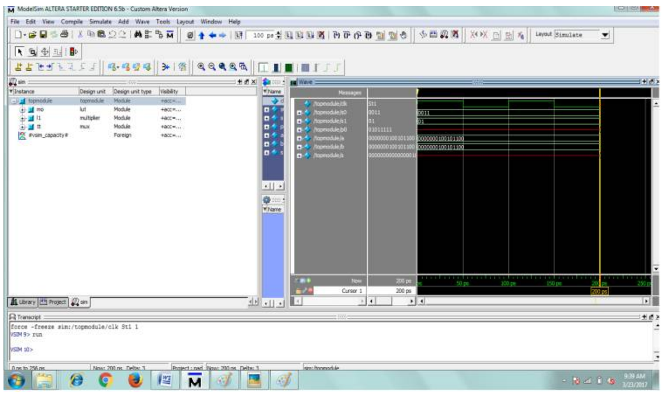


```
VSIM 6> run
force -freeze sim:/topmodule/clk 1 0, 0 {50 ps} -r 100

VSIM 8>
```

Clock= 0
S1 = 0011 ; S2 = 01
Output T1= 100ps

The result of final multiplication and accumulation process2



```
force -freeze sim:/topmodule/clk St1 1
VSIM 9> run

VSIM 10>
```

Clock = 1
S1= 0011 ; S2=01
Output T2=200ps
T = T1+T2
= (100+200) ps
T = 300ps

Power= Energy * Time
= 4 * 300
= 1200 pw

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 3, March 2017

COMPARISION BETWEEN EXISTING AND THE PROPOSED

Parameter	Existing	Proposed
Step1	Booth encoding	Booth encoding
Step 2	Hibird csa	Csa
Step 3	Not implemented	Look up table
Step 4	Not implemented	Mux
Step 5	Final addition	Final addition

VI. CONCLUSION

In this paper we are dissuced the effective improvement of carry save adder using modified booth algorithm. By using**,** Look up Table and Booth Encoder techniques are used. LUT is for speed of calculation, that is an array that replaces runtime computation with a simpler array indexing operation. The savings in terms of processing time can be significant, since retrieving a value from memory is after faster than undergoing an "Expensive" computation operation. We can reduce the power up to 24.4% and time delay is 300ps . By combining CSA and look up table process we can form a CSA, which is the most suitable application.

REFERENCES

[1] New VLSI architecture of parallel multiplier-accumulator based on radix-2 modified booth algorithm Ho Seo and Dong-Wook Kim,IEEE Volume 18,No2 Feb 2010

[2] A Spurious power suppression Technique for multimedia/Dsp applications, IEEE Transactions, Vol 56,No.1, Kuan-Hung Chen,Yuan-Sun Chu.

[3] C. S. Wallace, ―A suggestion for a fast multiplier,‖ *IEEE Trans. Electron Comput.*, vol. EC-13, no. 1, pp. 14–17, Feb. 1964.

[4] A. R. Cooper, ―Parallel architecture modified Booth multiplier,‖ *Proc. Inst. Electr. Eng. G*, vol. 135, pp. 125–128, 1988.

[5] N. R. Shanbag and P. Juneja, ―Parallel implementation of a 44-bit multiplier using modified Booth's algorithm,‖ *IEEE J. Solid-State Circuits*,vol. 23, no. 4, pp. 1010–1013, Aug. 1988. [6] J. Fadavi-Ardekani, ―MN Booth encoded multiplier generator using optimizedWallace trees,‖ *IEEE Trans. Very Large Scale Integr. (VLSI)Syst.*, vol. 1, no. 2, pp. 120–125, Jun. 1993.