DESIGN AND PERFORMANCE ANALYSIS OF MULTIPLIERS USING KOGGE STONE ADDER

Aradhanan Raju
Department of ECE
Silicon Institute of Technology
Bhubaneswar, Odisha, India
aradhana.raju@silicon.ac.in

Sudhir Kumar Sa
Department of ECE
Vijayanjali Institute of Technology
Balasore, Odisha, India
Sudhir007.sa@gmail.com

***Abstract*— Adders are known to have being frequently used in VLSI designs. This work deals with the designing and implementing multipliers using the underlying principle of parallel prefix adders. We are looking for less delay specific multiplier, so among the prominent PPA's like Kogge Stone Adder(KSA), Sparse Kogge Stone Adder(SKSA), Brent Kung Adder(BKA) and Lander Fischer Adder(LFA), we choose to implement the fastest PPA, i.e., KSA to get a comparative idea about the performance of four different multipliers namely; Binary multiplier, Braun Multiplier, Vedic Multiplier and Baugh Wooley Multiplier. Further the synthesis and simulation results reveal better idea about the proposed multipliers by giving an in depth view of their area, delay and power. A brief discussion about the application of the above is done. We have used Cadence Software and TSMC 180 nm technology.**

***Keywords*— Braun Multiplier, Kogge Stone Adder(KSA), Parallel Prefix Adders(PPA), VHDL(VHSIC Hardware Description Language), Vedic Multiplier.**

I. INTRODUCTION

The digital multiplier is a great eventful arithmetic unit being used in digital signal processors, many developing media processors and microprocessors. It also plays as a vital operator in digital filters, video and audio signals, computer graphics, and embedded systems. When compared with other arithmetic operations, it is seen that multiplication is both more time and power consuming. Hence, the designing of high speed multipliers along with low energy utilization, is still a prevailing topic of research. Since multipliers are indeed complex circuits and must preferably operate at a significant high clock rate, so the necessary requirement of our design is the reduction of time delay of the multipliers.

As binary addition continues to be an inevitable operation in digital circuits, researches are going forward in the direction of implying the multipliers using the simpler way of addition to get the same results with less complexity and faster response. There are different types of adders available like ripple carry adders, carry look ahead adders, parallel prefix adders etc. As PPA's are proved to be better among all therefore in this paper we totally focus on how to implement the fastest PPA to design four different multipliers, i.e., Binary multiplier, Braun multiplier, Vedic multiplier and Baugh Wooly Multiplier. We will have a comparative analysis of the parameters like area, time delay and power consumption of the above mentioned.

This paper consists of seven major sections out of which Section I takes us through the introduction part of the project. Section II describes about multipliers, their basic operation and the different types of multipliers are discussed in details. Section III shows the basic design and operation involved in PPA's. The comparison of the PPA's and the multipliers using the synthesis and simulation results and appropriate tabular representation is shown in Section IV. In Section V the applications of our work in MAC is discussed briefly. Section VI concludes our work. Section VII has our necessary references.

II. MULTIPLIERS

The simplest way to perform a multiplication is to use adders and AND gates. For P and Q bits wide inputs, the multiplication undergoes P cycles, using a Q-bit adder. The basic shift-and-add algorithm for multiplication adds P partial products together. Each partial product is generated by multiplying the multiplicand with one bit of the multiplier, which importantly is an AND operation, and then shifting the result on the basis of the multiplier bit's position.

```
MULTIPLICAND A:   1 1 1    = 7
MULTIPLIER    B :    1 1    = 3
                ---------------------------
                    1 1 1              pp  1
                  1 1 1                pp  2
                ---------------------------
                1 0 1 0 1      = 21  final product
```

A. Binary Multiplier

In binary multiplication, the shifted versions of the multiplicand are added, based on the position and value of each bit. Consequently, binary multiplication has a bigger hand over the conventional method decimal multiplication. Binary numbers can only be represented by 0 or 1, thus, depending on the value of the multiplier bit, the partial products can only be a copy of the multiplicand, or 0. This is simply an AND function in digital logic. It is built using binary adders.

A binary adder is illustrated in Fig. 1.

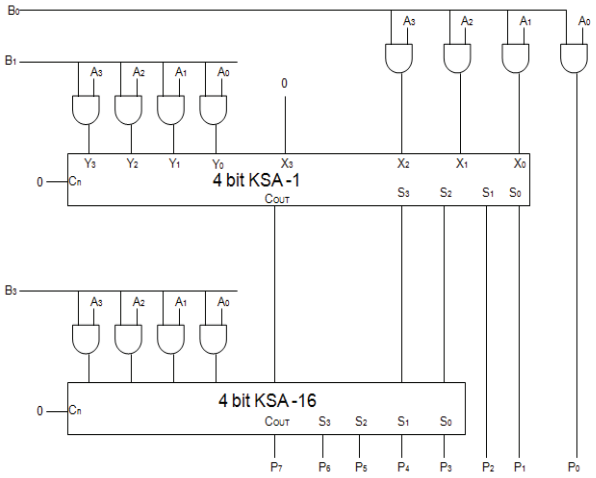


Fig .1. Architecture of a 4 bit Binary multiplier

B. *Vedic Multiplier*

Vedic Mathematics is a representation of arithmetic rules that gives better speed employment. The fact that it reduces the typical calculations used in conventional mathematics, to much uncomplicated ones, makes it a reliable option for usage. It provides some effective algorithms which can be applied to various branches of engineering.

For a simpler and even faster response, the Vedic multiplier is implemented using KSA.Below we have an instance of applying the operation of a KSA to implement a Vedic multiplier. In this structure, for a n*n bit multiplier we use four number of (n/2)*(n/2) Vedic multiplier, three number of Adders.

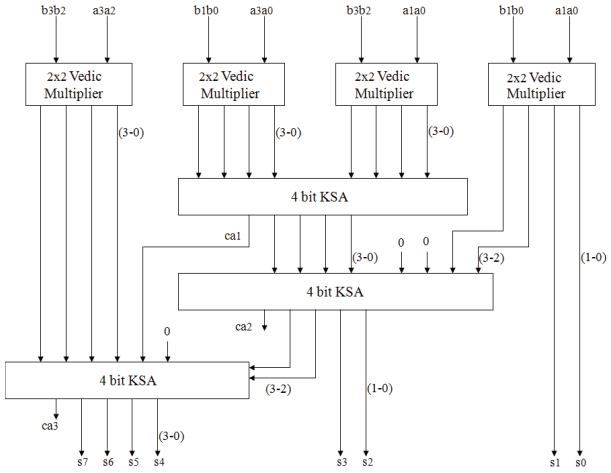


Fig. 2. Architecture of a 4-bit Vedic Multiplier

C. *Braun Multiplier*

Braun Multiplier is one of the multipliers, also referred as carry save array multiplier. Its structure is comprised of an array of adders and AND gates arranged in an iterative manner and there is no requirement of logic registers. An m*m bit Braun multiplier is constructed by using $(m-1)^2$ full adders, one parallel prefix adder and m^2 AND gates.

Each product is generated in parallel with the AND gates. Each partial product is added with the sum of the previously produced partial product by the row of adders. The carry out is shifted one bit either to the left or the right and then added to the sum which was generated by the first adder and the newly generated partial product. The last stage shifting needs the use of an adder, so here we use the Kogge Stone Adder. For designing an n-bit Braun Multiplier, we need (n-1) bit KSA for it.

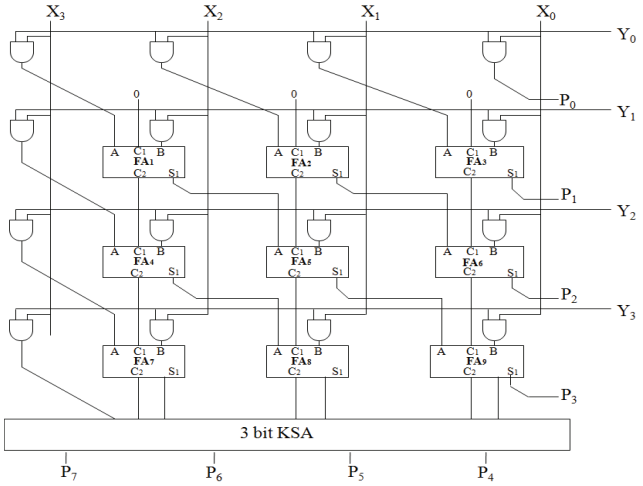


Fig. 3. Architecture of 4-bit Braun Multiplier

D. *Baugh-Wooley Multiplier*

In this two's compliment is most popular method for representing signed integers as it does not need any additional circuitry to represent negative number. Also the two's complement method makes it more efficient for operation of signed multiplication. The use of NAND gates in this multiplier makes it even better as NAND gate is a faster rate than the NOR gate for it uses only one PMOS for working.

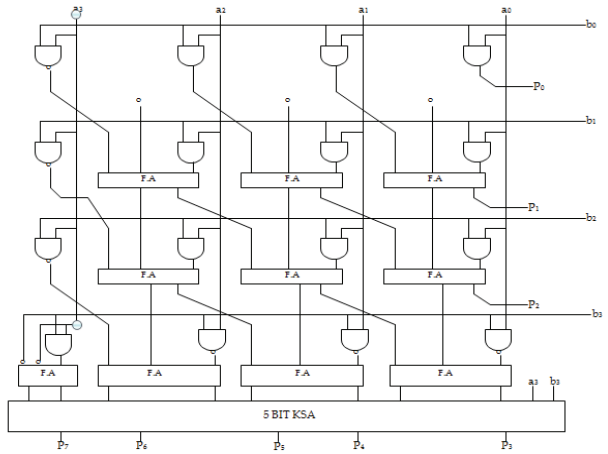


Fig.4. Architecture of 4-bit Baugh Wooley Multiplier

III. Parallel prefix adders

Parallel prefix adders work on the basis of a certain and simple principle. As the name suggests, 'parallel' signifies the execution of an operation in parallel which is done by segmenting it into smaller parts and then and then computing them in parallel fashion. 'Prefix' suggests that the outcome of the operations carried out depends on the previous input.

Parallel prefix adders employ the three stage structure for computing, namely:

1. Pre-computation of Pi and Gi.

This step involves computation of generate and propagate signals corresponding to each pair of bits.

2. Carry generation network

This step forms the deciding factor of all the PPA's as this varies for all PPA's depending upon their structure. It consists of the processing components and the buffer components. The output of the buffer component is the same as that of the input. The output of processing component is

$G_{i:k} = G_{i:j} + P_{i:j} \cdot G_{j-1:k}$

$P_{i:k} = P_{i:j} \cdot P_{j-1:k}$

The illustration is shown in Fig. 5.

3. Post computation

The sum and the final carry are calculated in this stage.

$S_i = P_i \cdot G_{i-1:-1}$

$C_{out} = G_{n:-1}$

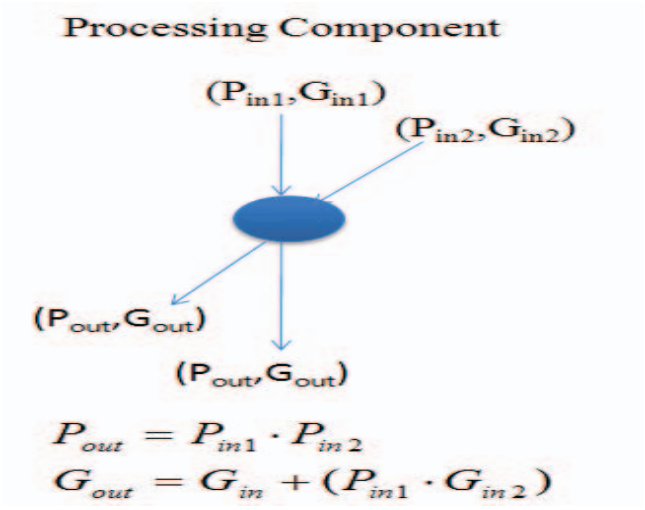


Fig.5. Carry Generation in PPA's

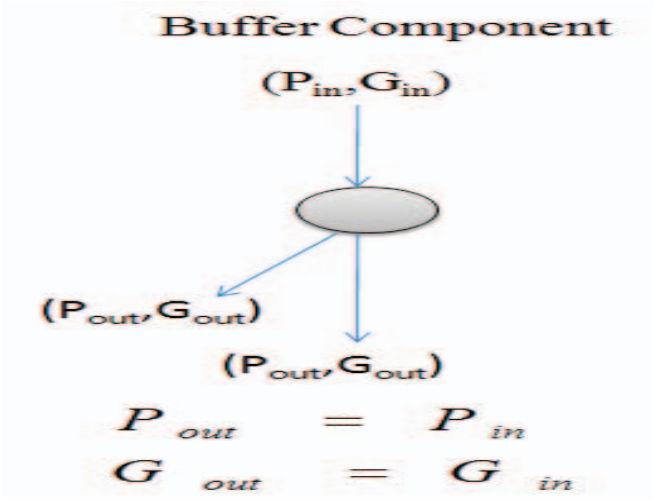


Fig.6. Carry Generation in PPA's

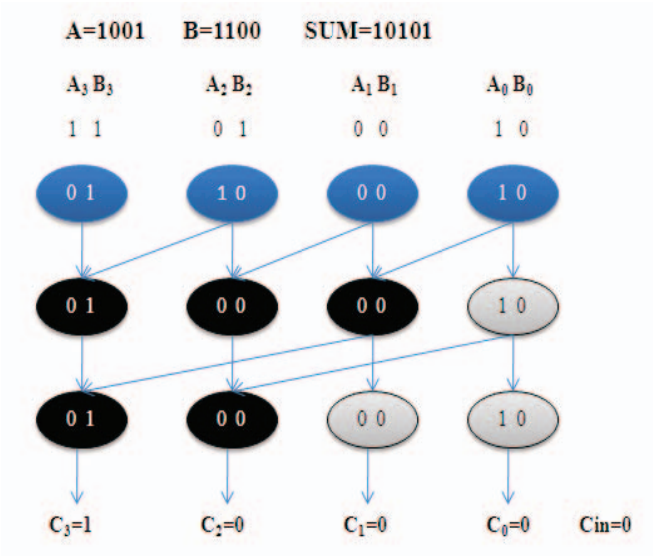


Fig.7. Architecture of a 4-bit KSA

IV. RESULT ANALYSIS

A. Synthesis Results

The synthesis results of the different 32-bit multipliers are as follows.

```
============================================================
  Generated by:           Encounter(R) RTL Compiler v12.10-s012_1
  Generated on:           Mar 14 2017  01:13:18 pm
  Module:                 MUL_KSA32
  Technology library:     tsmc18 1.0
  Operating conditions:   slow (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
============================================================

 Instance  Cells  Cell Area  Net Area  Total Area  Wireload
-----------------------------------------------------------------
MUL_KSA32   6815     108214         0      108214    <none> (D)
============================================================

   Pin         Type      Fanout  Load Slew Delay Arrival
                                 (fF) (ps)  (ps)   (ps)
-------------------------------------------------------------
b[0]          in port          1   3.5    0    +0       0 F
P[63]         out port                       +0   47142 F
-------------------------------------------------------------
Timing slack :  UNCONSTRAINED
Start-point  : b[0]
End-point    : P[63]
============================================================

                  Leakage     Dynamic       Total
 Instance Cells Power(nW)   Power(nW)    Power(nW)
-----------------------------------------------------
MUL_KSA32  6815  3041.831 16724323.061 16727364.892
```

i. Binary Multiplier

```
============================================================
  Generated by:           Encounter(R) RTL Compiler v12.10-s012_1
  Generated on:           Mar 14 2017  01:04:59 pm
  Module:                 BRAUN
  Technology library:     tsmc18 1.0
  Operating conditions:   slow (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
============================================================
Instance  Cells  Cell Area  Net Area  Total Area  Wireload
-----------------------------------------------------------------
BRAUN      1445      45542         0       45542    <none> (D)

============================================================

   Pin         Type      Fanout Load Slew Delay Arrival
                               (fF) (ps)  (ps)   (ps)
-------------------------------------------------------------
x[3]          in port         5 11.1    0    +0       0 F
P[62]         out port                       +0   14175 F
-------------------------------------------------------------
Timing slack :  UNCONSTRAINED
Start-point  : x[3]
End-point    : P[62]
============================================================

                Leakage    Dynamic      Total
Instance Cells Power(nW)  Power(nW)   Power(nW)
-------------------------------------------------
BRAUN     1445  2163.465 6077526.777 6079690.242
```

ii. Braun Multiplier

```
============================================================
  Generated by:           Encounter(R) RTL Compiler RC13.10 - v13.10-p005_1
  Generated on:           Mar 15 2017  02:33:42 pm
  Module:                 vedic_32
  Technology library:     tsmc18 1.0
  Operating conditions:   slow (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
============================================================

   Instance    Cells  Cell Area  Net Area  Total Area  Wireload
-----------------------------------------------------------------------
vedic_32        6362     136928         0      136928    <none> (D)
============================================================

     Pin           Type      Fanout Load Slew Delay Arrival
                                  (fF) (ps)  (ps)   (ps)
-------------------------------------------------------------
b[18]            in port        32 64.0    0    +0       0 F
P32[63]          out port                       +0   16005 R
-------------------------------------------------------------
Timing slack :  UNCONSTRAINED
Start-point  : b[18]
End-point    : P32[63]
============================================================

                    Leakage     Dynamic       Total
   Instance   Cells Power(nW)   Power(nW)    Power(nW)
-----------------------------------------------------------
vedic_32       6362  5000.925 13660334.305 13665335.230
```

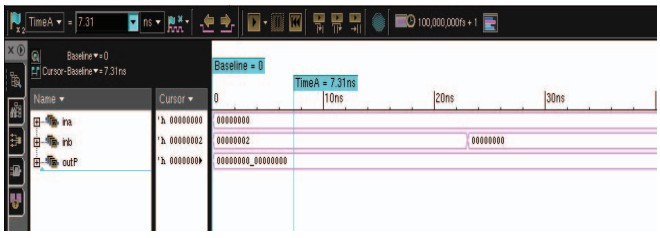
iii. Vedic Multiplier

```
============================================================
  Generated by:           Encounter(R) RTL Compiler v12.10-s012_1
  Generated on:           Mar 14 2017  05:15:25 pm
  Module:                 BW32
  Technology library:     tsmc18 1.0
  Operating conditions:   slow (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
============================================================
 Instance  Cells  Cell Area  Net Area  Total Area  Wireload
-----------------------------------------------------------------
BW32        1465       45695         0        45695    <none> (D)
============================================================

   Pin         Type      Fanout Load Slew Delay Arrival
                               (fF) (ps)  (ps)   (ps)
-------------------------------------------------------------
y[27]         in port         8 18.6    0    +0       0 F
P[58]         out port                       +0    14207 F
-------------------------------------------------------------
Timing slack :  UNCONSTRAINED
Start-point  : y[27]
End-point    : P[58]
============================================================

                  Leakage     Dynamic       Total
 Instance Cells Power(nW)   Power(nW)    Power(nW)
-----------------------------------------------------
 BW32       1465  2164.561 6198529.351 6200693.912
```

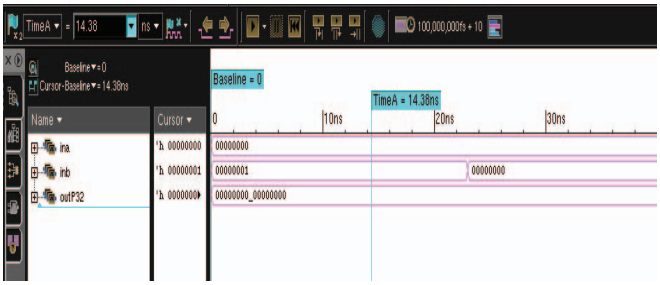
iv. Baugh Wooley Multiplier

B. Simulation Results

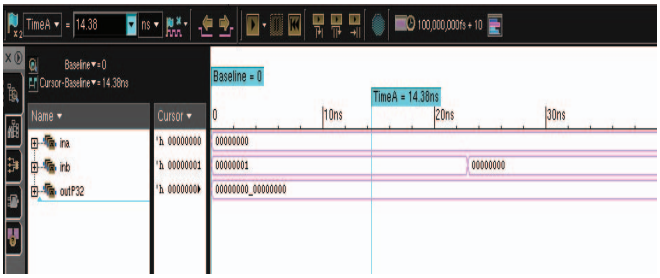
The simulation results of the different 32-bit multipliers are shown below.



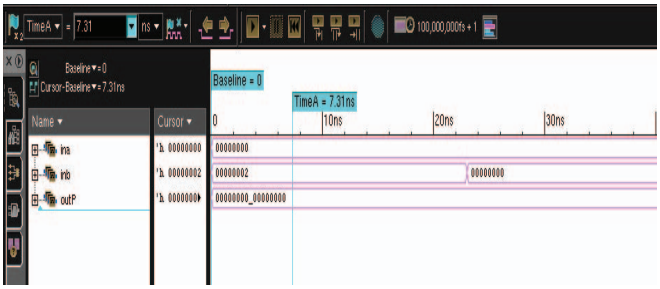
i. Binary Multiplier



ii. Vedic Multiplier



iii. Braun Multiplier



iv. Baugh-Wooley Multiplier

C. Tabular Comparison

Here we have a comparative analysis in the tabular form of the proposed parallel prefix adders and the multipliers stated above.

TABLE I. For 8-bit PPA's

Parameters	KSA	SKSA	BKA	LFA
Area	579	1420	639	579
Delay (ps)	2213	1604	1811	2213
Power (nW)	28795.163	63330.537	31120.45	28795.163

TABLE II. For 16-bit PPA's

Parameters	KSA	SKSA	BKA	LFA
Area	1720	1607	1201	1154
Delay (ps)	2039	2387	4273	4518
Power (nW)	80519.122	84666.383	64682.154	60286.263

TABLE III. For 32-bit PPA's

Parameters	KSA	SKSA	BKA	LFA
Area	4344	2931	2465	1726
Delay (ps)	2283	4734	8557	8834
Power (nW)	216419.27	168978.873	163292.376	91814.081

TABLE IV. For 16-bit Multipliers

Parameters	BINARY	BRAUN	VEDIC	BAUGH WOOLEY
Area	28567	19250	28234	20022
Delay (ps)	21438	10342	10847	11366
Power (nW)	3189651.445	2056470.357	2313120.806	2147772.523

TABLE V. For 32-bit Multipliers

Parameters	BINARY	BRAUN	VEDIC	BAUGH WOOLEY
Area	108214	45542	136928	45695
Delay (ps)	47142	14175	16005	14207
Power (nW)	16727364.892	6079690.242	13665335.230	6200693.912

V. APPLICATIONS

Multiplication being an inevitable operation in modern VLSI circuits has a vast range of applications. All the circuits which have multiplication as their main operation or even if a part of the operation requires a multiplication block, there we can use the application of our results of Braun Multiplier. Out of the many, one very significant application is MAC which is discussed here in very brief.

MAC (Multiplier Accumulator) is a prerequisite block in the DSP system. The vital element required to attain high efficiency in digital signal processing is a high throughput MAC. It extensively uses the basic Multiply-Accumulate operation in all modern DSP's. The unit of MAC allows clear-to-zero functions, very high-speed multiplication, multiplication with cumulative addition and subtraction. MAC is comprised of an adder, a multiplier and an accumulator. The implementation of the multiplier is based on the fastest multiplier and KSA, which is the fastest adder. The layout of this adder is very lucid which allows the fast design time.

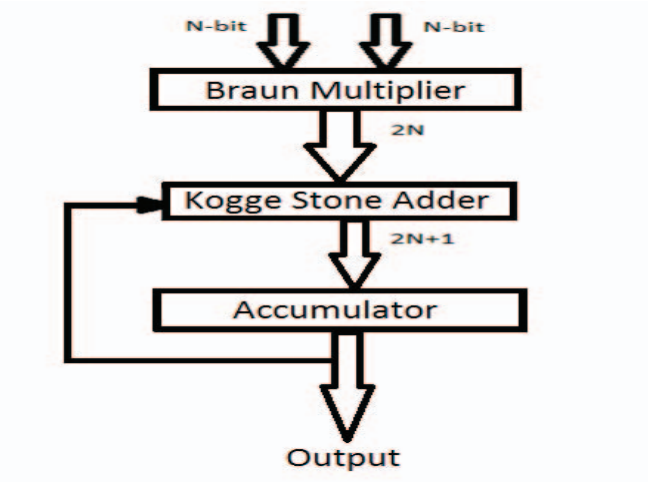


Fig.7. Block Diagram of MAC unit

The equation for MAC operation is given as:

x= x + (y*z) …….(1)

Where x represents an accumulator register, y represents the multiplier & z represents the multiplicand.

The above equation portrays the basic operation of the MAC unit.

VI. CONCLUSION

In this part of our project, we took the necessary assistance from our proved results of the fastest PPA, i.e., KSA to proceed further. Taking the next step towards multipliers, we implement Binary Multiplier, Braun Multiplier, Vedic Multiplier and Baugh Wooley Multiplier using the KSA, for simplifying and speeding up the multiplication with an aim to reduce their time delay. Finally the synthesis and simulation results of different multipliers gave us a clearer picture about the fact that among all of these, Braun Multiplier proved to be the best one in terms of delay performance. Also these results prove to be very useful in showing the power and area usage of all the four multipliers. An efficient implementation of KSA and Braun multiplier is discussed in the MAC unit operation to realize one of the practical implementations of our work.

VII. REFERENCES

[1] Raghumanohar Adusumilli and Vinod Kumar K, "Design and Implementation of a High Speed 64 Bit Kogge-Stone Adder Using Verilog Hdl", International Journal of Electrical and Electronic Engineering and Telecommunication (IJEETC), ISSN-2319-2518, vol-4 No. 1, Jan 2015

[2] Sunil M., Ankith R D, Manjunatha G D and Premananda B S, "Design and Implementation of Faster Parallel Prefix Kogge Stone Adder", International Journal of Electrical and Electronic Engineering & Telecommunications (IJEETC), ISSN: 2319-2518, vol. 3,No. 1, Jan 2014.

[3] V.Krishna Kumari, Y.Sri Chakrapani,"Designing and Characterization of koggestone, Sparse Kogge stone, Spanning tree and Brentkung Adders",International Journal of Modern Engineering Research (IJMER) Vol. 3, Issue. 4, July-august. 2013

[4] U.C. S. Pavan Kumar, A. Saiprasad Goud and A. Radhika, "FPGA Implementation of High Speed 8-bit Vedic Multiplier using Barrei Shifter", 2013International Conference on Energy Efficient Technologies for Technologies, Pages 14 to 17

[5] Pramodini Mohanty, RashmiRanjan, "An Efficient Baugh-Wooley Architecture for both Signed and Unsigned Multiplication.', International Journal of Computer Science & Engineering Technology (IJCSET), ISSN : 2229-3345 Vol. 3 No. 4 April 2012

[6] Anitha R, Bagyaveereswaran V, "Braun's Multiplier Implementation usingFPGA with Bypassing Techniques", International Journal of VLSI design & Communication Systems (VLSICS) Vol.2, No.3, September 2011

[7] Swati Malik and Sangeeta Dhal, 'Implementation of MAC Unit Using Booth Multiplier and Ripple Carry Adder', *International Journal of Applied Engineering and Research,* ISSN 0973-4562 Vol.7 No.11 (2012).

[8] Digital Electronics 4th edition, Morris Mano.

[9] VHDL programming by example 4th edition by Douglas Perry.