ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 2, Issue 11, November 2013

A High Speed and Device Efficient, FPGA Based Squaring Circuit

SHUBHAM MISHRA[1], SHAILENDRA KUMAR DHAKAD[2]

B.E(HONS), Dept. of EI ,BITS PILANI UNIVERSITY ,GOA,INDIA [1]

LECTURER, Dept. of EEE &I, BITS PILANI UNIVERSITY, GOA ,INDIA [2]

Abstract: In this paper a high speed squaring circuit for binary numbers is proposed. The methodology used is inspired from ancient Egyptian method of multiplication. Implementation of this method to binary squaring can increase speed and would improve the device utilization .It was found that the architecture targeted for Xilinx Virtex-4 FPGA(xc4vlx80-12) used 52 , 4-input LUTS with a combinational delay of 14.972 ns for 32 bit squaring.

Keywords: Squaring Algorithm, FPGA(field-programmable gate array), VHDL,VLSI Design, Peasant multiplication .

I. INTRODUCTION

Since squaring is one of the fundamental operations widely used in digital signal processing algorithms, a high-speed and highly efficient squaring methodology is proposed. We present a simple way of squaring which is inspired from Peasants method (old Egyptian technique for multiplication)[6]. the main motivation behind this work is to investigate the VLSI Design and Implementation of Squaring Circuit architecture with reduced delay and improved device utilisation .
In mathematics, ancient Egyptian multiplication(also known as Egyptian multiplication, Ethiopian multiplication, Russian multiplication, or peasant multiplication)[6], one of two multiplication methods used by scribes, was a systematic method for multiplying two numbers that does not require the multiplication table, only the ability to multiply and divide by 2, and to add. It decomposes one of the multiplicands (generally the larger) into a sum of powers of two and creates a table of doubling of the second multiplicand. This method may be called mediation and duplation, where mediation means halving one number and duplation means doubling the other number. It is still used in some areas.

II. THE MULTIPLIER ARCHITECTURE

In this section, we propose an efficient algorithm[6] which can be implemented for fast squaring calculations of binary numbers.
The algorithm used is peasants multiplication method , which is explained below :-

Take two numbers say A and B , to compute A*B through peasants method :

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 2, Issue 11, November 2013

(Rounding the numbers on left and right sections of the table)

1) Step 1 : half the numbers on the left side and double the numbers on the right side till the left side becomes 1.

LEFT	RIGHT
A	B
A/2	2B
A/4	4B
.	.
.	.
.	.
.	.
1	2nB

2) Step 2 : discard the line in which left has an even value .for example say A/2 is even .

LEFT	RIGHT
A	B
~~A/2~~	~~2B~~
A/4	4B
.	.
•	•
•	•
•	•
1	2nB

3) Step 3: form a new table excluding those entries which are discarded after 2nd step and add the entries on Right.

LEFT	RIGHT
A	B
A/4	4B
.	.
.	.
.	.
.	.
1	+ 2nB
	A*B

example : 13 * 238

13	238
~~6~~	~~476~~
3	952
1	+ 1904
	3094

This could be extended to binary numbers ,for example we want to multiply 13(1101) and 238(11101110) :

1101	11101110	
~~110~~	~~111011100~~	
11	1110111000	
1	+ 11101110000	
	110000010110	(3094)

III. MODIFICATION IN METHOD

The method can be modified in order to make it more efficient and hardware implementable .
In order to delete the row with even Left entry ,we can use the following method:-

1)Multiply each bit on the right in a particular row by the last bit (l.s.b) of the number of the same row.

2)Multiplying this bit will automatically cancel rather make all the elements of that row on the right as '0' if the number on left is even and otherwise would leave it as it is.

Example: let us take the same example 1101 * 11101110

LEFT	RIGHT	LAST DIGIT IN LEFT COLUMN
1101	11101110	1
110	111011100	0
11	1110111000	1
1	11101110000	1

Now every bit of the RIGHT row is multiplied by the respective bit mentioned next to the row.
So our transformation in right column would be :

LEFT	RIGHT
1101	(1*1)(1*1)(1*1)(0*1)(1*1)(1*1)(1*1)(0*1)
110	(1*0)(1*0)(1*0)(0*0)(1*0)(1*0)(1*0)(0*0)(0*0)
11	(1*1)(1*1)(1*1)(0*1)(1*1)(1*1)(1*1)(0*1)(0*1)(0*1)
1	(1*1)(1*1)(1*1)(0*1)(1*1)(1*1)(1*1)(0*1)(0*1)(0*1)(0*1) New

table would be:

```
LEFT                RIGHT

1101                 11101110
110                 000000000
11                 1110111000
1          +      11101110000
     ------------------------
                 110000010110
```

IV . HARDWARE IMPLEMENTATION

This method could be extended to square n bit numbers at a high-speed at high efficiency using logic gates.

A) Squaring a two bit number :

Let us take a two bit number msb(a(1) a(0))lsb .Squaring this number by the above method can be described as:

```
LEFT                 RIGHT

   a (1) a(0)                        (a(1) and a(0))   (a(0) and a(0))
 a(1)         (a(1) and a(1))   (a(1) and a(0))          0
   ------------------------------------------------------------
                S(2)                  S(1)                S(0)
```

On performing gate simplification the result is:

S(0) = a(0)
S(1) = 0
S(2) = not (a(0) and a(1)) and a(1)

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 2, Issue 11, November 2013

B) Squaring a four bit number :

Let us take a for bit number msb (a(3) a(2) a(1) a(0))lsb .Squaring this number by the above method can be described as above. The result after gate simplification is:

S(0) = a(0)
S(1) = 0
S(2) = (not (a(0) and a(1)) and a(1))
S(3) = a(0) and a(1)
S(4) = not (a(0) and a(1) and a(2) and a(3)) and a(2)
S(5) = a(1) and a(2) and a(3)
S(6) = not (a(2) and a(3)) and a(3)
S(7) = a(3)

C) Squaring a eight bit number :

Let us consider a eight bit number msb(a(7) a(6) a(5) a(4) a(3) a(2) a(1) a(0))lsb. The result will be :

S(0) = a(0)
S(1) = 0
S(2) = (not (a(0) and a(1)) and a(1))
S(3) = a(0) and a(1)
S(4) = (not (a(0) and a(1) and a(2) and a (3)))and a(2)
S(5) = a(0) and a(1) and a(2) and a (3)
S(6) =not(a(0) and a(1) and a(2) and a (3)and a(4) and a(5)) and a(3)
S(7) = a(0) and a(1) and a(2) and a (3) and a(4) and a(5)
S(8) = not(a(0) and a(1) and a(2) and a (3)and a(4) and a(5) and a(6) and a(7)) and a(4)
S(9) = a(0) and a(1) and a(2) and a (3)and a(4) and a(5) and a(6) and a(7)
S(10) =not(a(2) and a (3)and a(4) and a(5) and a(6) and a(7)) and a(5)
S(11) =a (3)and a(4) and a(5) and a(6) and a(7)
S(12) =not(a(4) and a(5) and a(6) and a(7)) and a(6)
S(13) = a(5) and a(6) and a(7)
S(14) = not(a(6) and a(7)) and a(7)
S(15) = a(7)

Where S(0) is the lsb and S(15) is the msb of the of the square.

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 2, Issue 11, November 2013

D) General algorithm for a ' n ' bit number :

The algorithm can be extended to an 'n' bit number . Where 0 is the lsb and n is its lsb .

To apply the generalised algorithm we need to understand following terms :

- n denotes number of bits in the number which is to be squared.
- j = 2*n – 1
- S denotes the square of the input 'a' .
- S(k) denotes the k th term in S.
- 0 is assigned as the msb(most significant bit) of S.
- round off all the division values : if k= 7 , k/2=3

The terms in its square can be found by using just four generator terms :

Table 1.Generalised algorithm

Limit on 'k'	**Even terms**	**Odd terms**
1< k <= j/2 + 1	not(a(0) and a(1).......and a(k-1)) and a(k/2)	a(0) and a(1)........and a(k-2)
k > j/2 + 1	not(a(j/2) and a(j/2+1) and......and(a(k-j/2-1))and a(k/2)	a(j/2) and a(j/2 +1)…..and a(k – j/2 – 1)

*NOTE : S(0) and S(1) cannot be found using this formula their values are assigned as:

S(0)=a(0) and S(1)= 0

V . RESULTS

The modified high speed squaring algorithm design was simulated in Modelsim 6.6c [2]and synthesised using Xilinx ISE 12.2i [2] through VHDL[1] which was mapped on to Virtex-4 (xc4vlx80 -12) FPGA.
The results for device utilisation are placed in table 2. Table3 reflects the comparison between the proposed algorithm and the existing methodologies .
From the point of view of combinational delay the table 4 provides a comparison between the proposed algorithm and the existing architecture’s .

Table 2. Device Utilisation

Device: xc4vlx80-12	LUT's (4 input)	Slices	Bonded IOB's
4 bits	5	3	12
8 bits	15	9	24
16 bits	30	17	48
32 bits	52	30	96

Table 3. Comparison Of 4 Input LUTS Used

Device: xc4vlx80-12	Modified Booth Multiplier[4]	Prabha etal[4]	Vedic[7]	Ours
4 bits	32	6	6	5
8 bits	186	35	64	15
16 bits	880	294	366	30
32 bits	2760	1034	1267	52

Table4. Comparison Of Combinational Delays(ns)

Device: xc4vlx80-12	Modified Booth Multiplier[4]	Prabha etal[4]	Vedic[7]	Ours
4bits	8.154	4.993	4.993	4.987
8bits	15.718	14.263	12.784	6.299
16bits	36.657	33.391	15.992	8.987
32bits	74.432	68.125	18.272	14.972

VI. CONCLUSION

The modified high speed squaring algorithm was targeted on a Xilinx xc4vlx80-12-ff1148 FPGA.. The design achieved a high device utilisation with only 52 LUTS(4 input) for 32bit squaring **.** The implemented design is also efficient in terms of device utilisation and fast **.**The implementation verified that there was a delay of 14.972ns for 32 bit squaring which is far better than the existing vedic[7] squaring and booth squaring[4]. The idea proposed here may set path for future research in this direction. Future scope of research this is to reduce area requirements and can be extended to various fields of DSP.

REFERENCES

[1] V.A. Pedroni, "Circuit Design with VHDL," 2008.
[2] Xilinx(12.1) ISE User Manual', Xilinx Inc, USA,April 19, 2010. ,
[3] Prabha S., Kasliwal, B.P. Patil and D.K. Gautam, "Performance Evaluation of Squaring Operation by Vedic Mathematics", IETE Journal of Research, vol.57, Issue 1, Jan-Feb 2011
[4] B. Parhami, "Computer Arithmetic Algorithms and Hardware Architectures," 2nd ed, Oxford University Press, New York, 2010.
[5] Kai Hwang, "Computer Arithmetic: Principles, Architecture and Design," New York: John Wiley & Sons, 1979.
[6] http://en.wikipedia.org/wiki/Ancient_Egyptian_multiplication
[7] "An Improved Squaring Circuit for Binary Numbers"(IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 3, No.2, 2012
[8] B. Jeevan,"A High Speed Binary Floating Point Multiplier Using Dadda Algorithm" ,pg.1944 – 1947, vol.3 IEEE International Symposium on Circuits and Systems, 1997. ISCAS '97.