DESIGN OF HIGH SPEED AND AREA EFFICIENT BCD MULTIPLIER WITH MODIFIED CONVERTORS

Ruchira Dixit, M. Tech. scholar.GGITS, Jabalpur
Prof. Abhishek Singh, GGITS,Jabalpur

Abstract: Now days we are living in digital world, where all the operations get performed more reliably and with highest accuracy by digital signal processor. MULTIPLIER is the key element of all these processor like Microprocessor, Microcontroller, DSP processor etc. In this paper we present a new method for implementing BCD multiplication more efficiently than previous proposals in current FPGA devices with 6-input LUTs. In particular, a combinational implementation maps quite well into the slice structure of the Xilinx Virtex-5/Virtex-6 families and it is highly pipeline-able. The synthesis results for a Virtex-6 device indicate that our proposal outperforms the area and latency figures of previous implementations in FPGAs.

Keywords: BCD-Binary Coded Decimal, LUT-Look Up Table, DSP- Digital Signal Processing, RTL-Register Transfer Level, EDA-Electronic Design Automation

I-INTRODUCTION

After through study and deep analysis work we have seen that the existing[1] BCD multiplication hardware have some limitation in terms of area. To overcome these limitations a novel approach has been proposed to design the BCD multiplier with unique addition structure, which is used to add partially generated products. To meet our major concern 'Speed' we need particular high speed MULTIPLIER, the speed of MULTIPLIER greatly depends upon the speed of multiplication unit used in it. There are so many multiplication techniques exist now a days at algorithmic and structural level. Our proposed MULTIPLIER algorithm is already efficient in respect of area and speed, all we need to optimize it further to have it we have plan to use our proposed BCD Wallace multiplication technique on proposed MULTIPLIER module.

Every digital domain based technology depends upon the operations performed by MULTIPLIER either partially or whole. Speed is the most prominent factor of processor and controllers being used recently. Decimal multiplication is one of the most frequent operations used by many financial, business and user-oriented applications but current implementations in FPGAs are very inefficient in terms of both area and latency when compared to binary multipliers.

With growing popularity for decimal computer arithmetic in scientific, commercial, financial & Internet-based applications, hardware realization for decimal arithmetic algorithms is gaining much importance. Hardware decimal arithmetic units now serve as an integral part for few recently commercialized general purpose processors, where complex decimal arithmetic operations, such as multiplication, have been realized by rather slow iterative hardware algorithms. However, with rapid advances in very large scale integration (VLSI) technology[14], semi- & fully parallel hardware decimal multiplication units are expected to evolve soon. dominant representation to decimal digits is binary-coded decimal (BCD) encoding. BCD-digit multiplier may serve as key building block for a decimal multiplier, irrespective for degree for parallelism.

To Design BCD MULTIPLIER module using VHDL, designed module will be synthesized using Xilinx ISE 9.1i [19] Web pack, & verification will be done on ISE simulator, & then to validation be design module will be implemented on Xilinx FPGA Vertex2.

The test bench-waveforms to various parts for BCD-MULTIPLIER verification will be on ISE[20] simulator with standard benchmarks.

To achieve a better speed efficient MULTIPLIER & to compare our work with base references for standard journal papers. To optimized area & speed for our design module as compare to existing design modules.

II- STRUCTURE FOR DESIGN

Figure 1 shows a flow for proposed work as may be seen there are three major modules which has been developed very first a BCE to binary & also a Binary to BCD convertors requires actual modification is been proposed in all these conversion methods also procedure adopted to multiplication also new digit A & B multiplication using vertical-cross method.

Step 1: Multiply numbers in one's place & put product directly under one's. (A(LSB half) *B(LSB half))

Step 2: Cross multiply, we would form fractions by taking top number's tens digit multiplied by

bottom number's ones place. Then take top number's tens place multiplied by bottom number's tens place. Add both products. {(A(LSB half) *B(MSB half))+ (A(MSB half) *B(LSB half))}
Step 3: Multiply again, numbers in tens place & place answer to left for previous step's answer. (A(MSB half) *B(LSB half))
Step 4: Add all partial products with previous step's carry if it is.

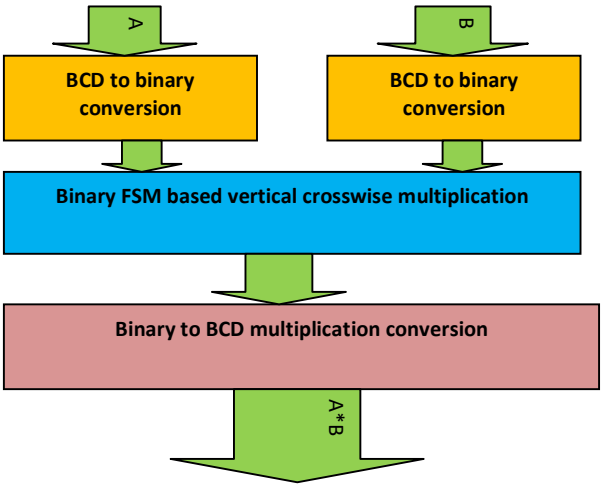


Figure 1: proposed BCD multiplication module
The design has three major modules
16 bit BCD to binary convertor
16 bit binary multiplication
32 bit Binary to BCD convertor
BCD to binary convertor: Proposed new architecture of 16 bit BCD to binary convertor is shown below in figure 2. RS s right shift, WA is Wallace adder, WS is Wallace subtraction

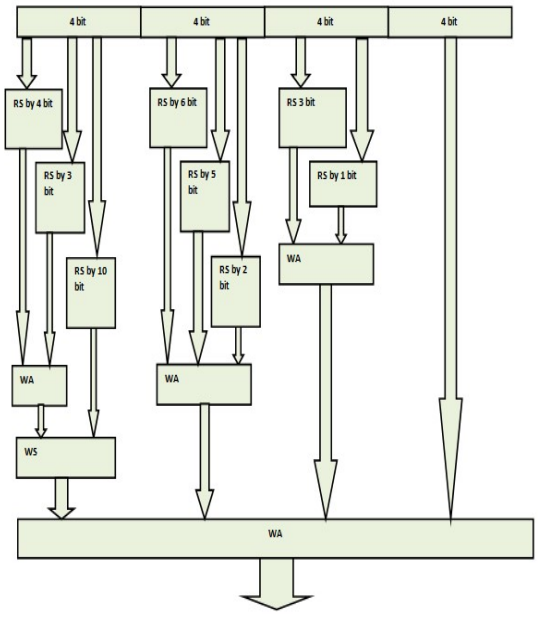


Figure 2 Proposed BCD to binary convertor

The process of proposed design can be explained as
Let 16 bit input BCD is as below
B15 B14 B13 B12 B11 B10 B9 B8 B7 B6 B5 B4 B3 B2 B1 B0
The proposed concept is to perform multiplication using shifting as know left shift by 'n' multiplies a number with 2^n
And for getting 16 bit binary of 16 bit BCD we must do

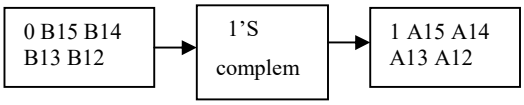
S15S14S13S12S11S10S9S8S7S6S5S4S3S2S1S0

= (B15B14B13B12) x 1000 + (B11B10B9B8) x 100 + (B7B6B5B4) x 10 + (B3B2B1B0)

= (B15B14B13B12) x (1024-16-8) + (B11B10B9B8) x (64+32+4) + (B7B6B5B4) x (8+2) + (B3B2B1B0)

= (B15B14B13B12)x(1024) - (B15B14B13B12)x(16) - (B15B14B13B12)x(8) + (B11B10B9B8)x(64+32+4) + (B7B6B5B4)x(8+2) + (B3B2B1B0)

As we can perform addition instead of subtraction if we change the sign using 2'S complement we perform upper nibble compliment with '0' padding at its MSB first, and 2'S complement is 1'S complement plus '1', proposed work added '1' in its tree adder architecture.



= { (B15B14B13B12)x(1024) } + { (1A15A14A13A12)x(16) + '1' } + { (1A15A14A13A12)x(8) + '1' } + { (B11B10B9B8)x(64+32+4) } + { (B7B6B5B4)x(8+2)} +{ (B3B2B1B0) }

Multiplication by 1024 will be performed by left shifting by 10 bit, Multiplication by 16 will be performed by left shifting by 4 bit, Multiplication by 8 will be performed by left shifting by 3 bit, Multiplication by 64 will be performed by left shifting by 6 bit, Multiplication by 32 will be performed by left shifting by 5 bit, Multiplication by 4 will be performed by left shifting by 2 bit, Multiplication by 2 will be performed by left shifting by 1 bit

								C21	C16	C11					
							C25	C20	C15	C10	C6				
						C28	C24	C19	C14	C9	C5				
					C30	C27	C23	C18	C13	C8	C4	C2			
	C34	C33	C32	C31	C29	C26	C22	C17	C12	C7	C3	C1	C0		
							1	A15	A14	A13	A12				
											1				
								1	A15	A14	A13	A12			
												1			
		B15	B14	B13	B12										
										B11	B10	B9	B8		
							B11	B10	B9	B8					
						B11	B10	B9	B8						
											B7	B6	B5	B4	
									B7	B6	B5	B4			
												B3	B2	B1	B0
S15	S14	S13	S12	S11	S10	S9	S8	S7	S6	S5	S4	S3	S2	S1	S0
0FA 0HA	0FA 1HA	0FA 1HA	0FA 1HA	0FA 1HA	1FA 0HA	1FA 1HA	3FA 0HA	4FA 0HA	4FA 1HA	4FA 1HA	4FA 1HA	3FA 1HA	1FA 1HA	0FA 1HA	0HA 0FA

Figure 3 Proposed new addition structure

Figure 3 above is the proposed addition structure as it has shifting as was requires and also using addition of '1' for 2'S compliment. Total 25 full adder and 11 half adder requires for addition with proposed addition structure and it is very less as compare with the available design of conversion.

The 16 bit Multiplication: Figure 4 shown below is the design of our proposed 4 bit
Multiplier and with the help of 4 bit multiplier a 8 bit multiplier is been developed and with the help of 8 bit multiplier a 16 bit multiplier developed. Let In1 & In2 are the 4 bit numbers, below is the explanation of 4 bit multiplier. IN1 and IN2 are two different inputs.
Proposed 4 bit multiplication: figure 4 and figure 5 shows the proposed work of 4 bit multiplication.

The outputs T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12, T13, T14, T15 and T16 are the outputs after logical AND operation. We required adding this as shown below

Figure 5 above shows the our way of performing addition in this way one can have minimum hardware required as compare with Wallace addition or carry save addition we requires only 8 full adder and 4 half adder through Wallace addition in this scenario requires total 7 full adders and 8 half adders and carry save adder requires 8 full adders and 7 half adders. It shows that proposed method needs minimum area.

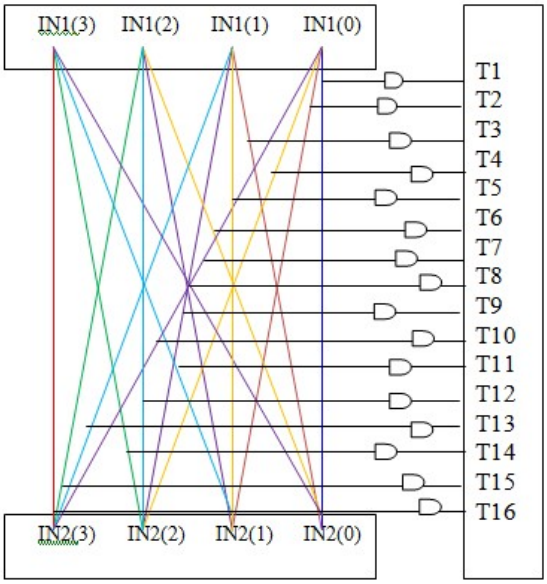


Figure 4: The 4 bit cross multiplication

		C9	C6				
	C11	C8	C5	C3			
C12	C10	C7	C4	C2	C1		
	T16	T14	T11	T7	T4	T2	T1
		T15	T12	T8	T5	T3	
			T13	T9	T6		
				T10			
S7	S6	S5	S4	S3	S2	S1	S0
0FA 0HA	1FA 0 HA	2FA 0HA	2FA 1HA	2FA 1 HA	1FA 1HA	0FA 1HA	0FA 0HA

Figure 5: the proposed addition structure for 4 bit multiplication

8-bit Multiplication: Now for 8 bit multiplication, proposed work have 8 bit inputs x & y and answer will be produce in z which will be of 16 bit. Proposed work performs "4 bit multiplication "on each arrow as shown in figure below.

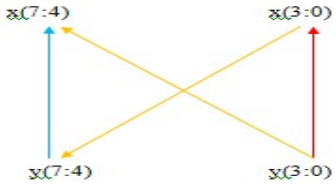


Figure 6: 8 bit cross Multiplication using 4 bit cross multiplication

Red arrow multiplication produces 8 bit ans. P7-P0
Yellow arrow multiplication produces 8 bit ans. R7-R0 & Q7-Q0
Blue arrow multiplication produces 8 bit ans. T7-T0

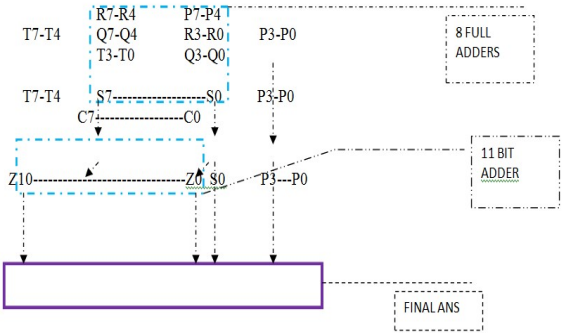


Figure 7 Addition structure for 8 bit multiplication

Figure 7 shown above shows the new addition structure design that we have adopted for the proposed 8 bit multiplication, it requires 8 full adders and one 11 bit adder, instead of old conventional way which requires total three 16 bit adder.
16-bit Multiplication: Now for 16 bit multiplication, thesis work have 16 bit inputs a and b and answer will be produce in z which will be of 32 bit. Thesis work perform "8 bit multiplication "on each arrow as shown in figure below.

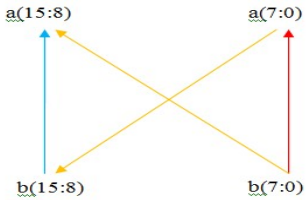


Figure 8: 16 bit cross Multiplication using 8 bit cross multiplication

Red arrow multiplication produces 16 bit ans. P15-P0
Yellow arrow multiplication produces 16 bit ans. R15-R0 & Q15-Q0
Blue arrow multiplication produces 16 bit ans. T15-T0

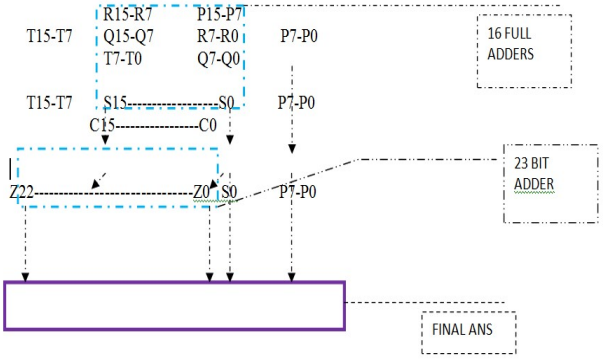


Figure 9 Addition structure for 16 bit multiplication

Figure shown above shows the new addition structure design that we have adopted for the proposed 16 bit multiplication, it requires 16 full adders and one 11 bit adder, instead of old conventional way which requires total three 32 bit adder.
Binary to BCD convertor: Figure 11 below is showing the method of converting a binary number into BCD it shows an example of conversion of 8 binary bit number, it use the concept of left shift and if any number in Units, or Tens or Hundreds become greater than four add '3 (0011)' before the next shifting, the shifting must be done until LSB of binary number not shifted into units.

Operation	Hundreds	Tens	Units	Binary	
HEX				F	F
Start				1 1 1 1	1 1 1 1
Shift 1			1	1 1 1 1	1 1 1
Shift 2			1 1	1 1 1 1	1 1
Shift 3			1 1 1	1 1 1 1	1
Add 3			1 0 1 0	1 1 1 1	1
Shift 4		1	0 1 0 1	1 1 1 1	
Add 3		1	1 0 0 0	1 1 1 1	
Shift 5		1 1	0 0 0 1	1 1 1	
Shift 6		1 1 0	0 0 1 1	1 1	
Add 3		1 0 0 1	0 0 1 1	1 1	
Shift 7	1	0 0 1 0	0 1 1 1	1	
Add 3	1	0 0 1 0	1 0 1 0	1	
Shift 8	1 0	0 1 0 1	0 1 0 1		
BCD	2	5	5		

Figure 10 Binary to BCD conversion

Figure 11 below shows the proposed block diagram for the binary to BCD conversion here proposed method is using the concept of Leading One Detector (LOD) for checking the size of the binary umber and use subtraction by multiple of 10 (i.e 1000, 100 and 10) and count total subtractions this way it find the digits.

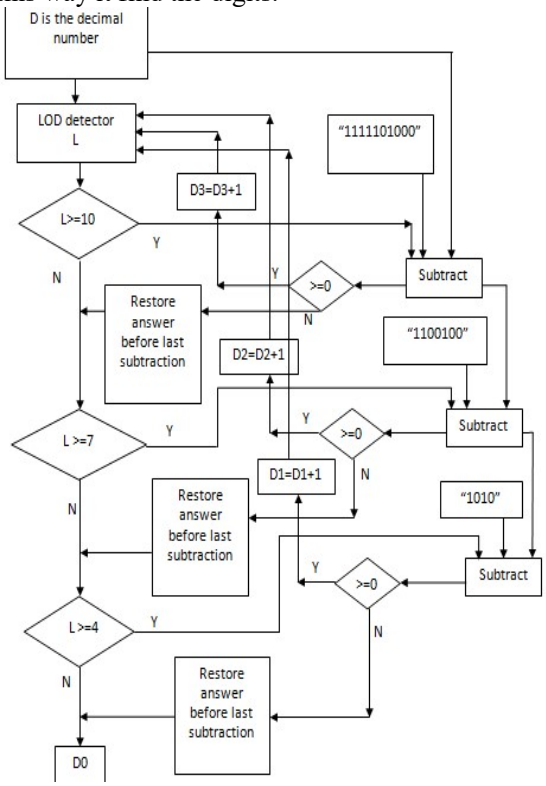


Figure 11 Proposed Binary to BCD conversion

Let's take an example consider our binary number is (3485d)=(D82h)=(110110000010b)
LOD is at 12 hence

	LOD	D3	D2	D1	D0
110110000010-1111101000	12	1			
100110011010-1111101000	12	2			
10110110010-1111101000	11	3			
111001010-1100100	9		1		
101100110-1100100	9		2		
100000010-1100100	9		3		
1100100-1100100	7		4		
111010-1010	6			1	
110000-1010	6			2	
100110-1010	6			3	
11100-1010	5			4	
10010-1010	5			5	
1000	4				8
		3	4	5	8

Figure 12 Proposed Binary to BCD conversion Example

Figure 13 below shows synthesis results of the proposed 16 bit BCD multiplier and figure 4.7 shows the Top view of the proposed 16 bit BCD multiplier and figure 4.8 shows the RTL internal view which shows the modules bcd2binary convertor used two time a 16 bit multiplier used one time and a 32 bit binary2bcd convertor.

top Project Status (09/25/2017 - 00:12:01)			
Project File:	codenew.xise	**Parser Errors:**	No Errors
Module Name:	top	**Implementation State:**	Synthesized
Target Device:	xc4vlx200-11ff1513	**• Errors:**	No Errors
Product Version:	ISE 12.2	**• Warnings:**	22 Warnings (22 new)
Design Goal:	Balanced	**• Routing Results:**	
Design Strategy:	Xilinx Default (unlocked)	**• Timing Constraints:**	
Environment:	System Settings	**• Final Timing Score:**	

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	482	89088	0%
Number of Slice Flip Flops	105	178176	0%
Number of 4 input LUTs	863	178176	0%
Number of bonded IOBs	66	960	6%
Number of GCLKs	1	32	3%

Figure 13 synthesis results of proposed BCD multiplier

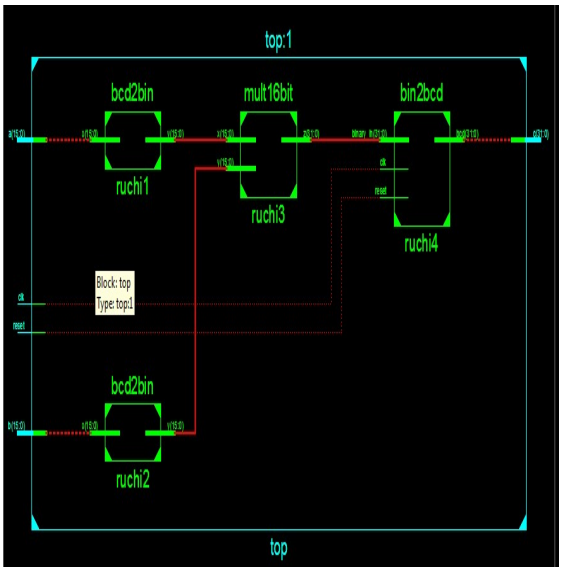


Figure 14 Internal modules used in 16 bit BCD multiplication

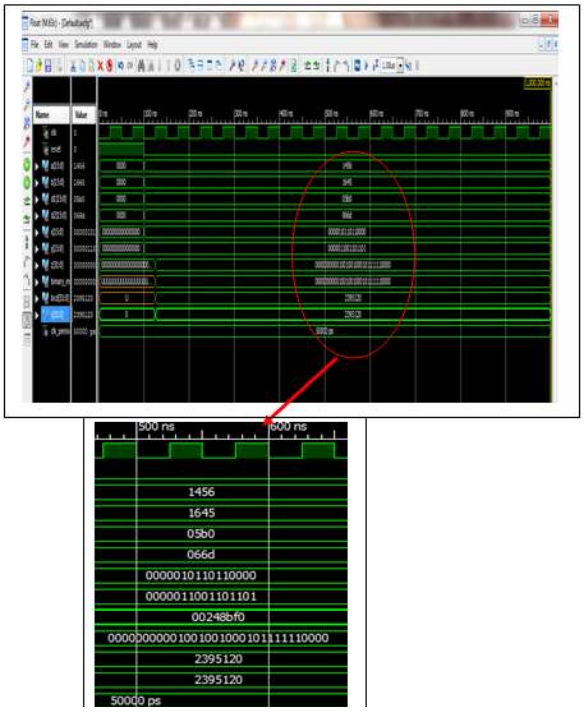


Figure 15 Simulation Results of Proposed design

Figure 15 above shows the result obtain from the simulation of the design, the simulation analysis can be understand as from the table below:-

In1	In2	Bin 1	Bin2	Bin Mult	BCD out	
14 56	16 45	5B 0	66 D	24BF 0	23951 20	OK

Table 1 the simulation observations

Device utilization summary:	
Selected Device : xc4vlx25-11ff668 Vertex 4	
Parameters	Used
Number for Slices:	482
Number for Slice Flip Flops:	105
Number for 4 input LUTs:	863
Number for bonded IOBs:	66
Time Delay	3.863ns
Max Freq.	267.251MHz
Number for GCLKs:	1

Table 2 Synthesis results Observed to proposed multiplier

Table 1 above is the simulation results observed for the multiplication of two numbers and found correct.
Table 2 above is the synthesis results observe for the proposed design it is also found correct.
COMPARATIVE RESULTS

Comparative Results observed to FPGA Platform Vertex family FPGA				
Parameters	Proposed	Xiaoping Cui et al [1]	Sonam Negi et al [2]	Saeid Gorgin et al [3]
Number for Slices:	482 (combinational logic)	536 (combinational logic)	539 (combinational logic)	668 (combinational logic)
Number for Slice Flip Flops:	105			
Number for 4 input LUTs:	863			
Number for bonded IOBs:	66			
Time Delay	3.863 ns	4.18 ns	4.42	6 ns
Max Freq.	267.251 Mhz			
Number for GCLKs:				

Table 3 Comparative Results

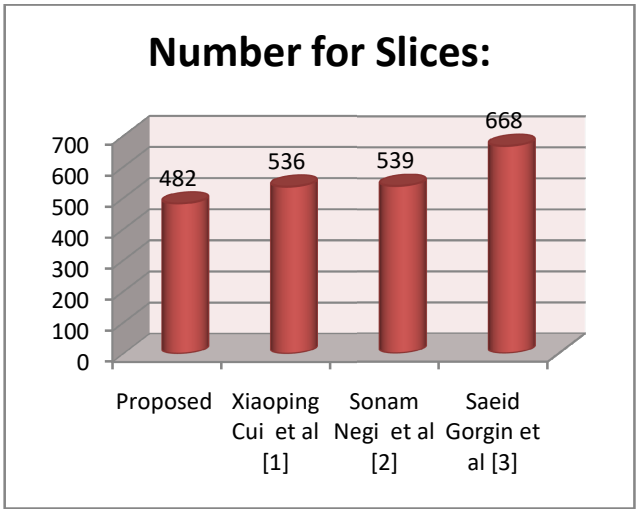


Figure 16 Number of slice used comparison

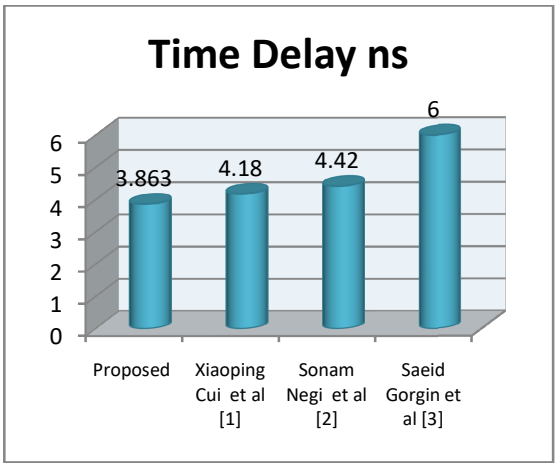


Figure 17 Time delay requires comparison

From table 3 it is clearly observed that observed results to 16x16 BCD multiplication are best among base works in terms for area & speed both, although base work [3] is a combination logic design which we may say is better or not better than our work, however rest for two base works also use clock based sequential logic to produce BCD multiplication result.

IV-CONCLUSION

In this thesis work we have proposed two various algorithms one to BCD adder & another to NxN vertical & crosswise BCD multiplication & synthesized results for 16+16 BCD adder depicts it's efficiency in terms for slices & time for delay. This thesis has reassessed several implementations for NxM-digit multiplications on Xilinx FPGAs. This work presents design for several BCD multipliers & their implementations on Virtex-6 FPGA.

In this work a decimal fully parallel & with FSM based multiplier is presented. Several enhancements are used to improve latency such as use for a new FSM control unit & new BCD adder design multiplier & use for a fast decimal carry propagation adder. multiplier is synthesized in 45 nm technology & simulated at Xilinx ISE. multiplier shows very good performance with respect to delay & area.

REFERENCES

[1] Xiaoping Cui, Weiqiang Liu and Dong Wenwen Fabrizio Lombardi, A Parallel Decimal Multiplier Using Hybrid Binary Coded Decimal (BCD) Codes, 2016 IEEE 23nd Symposium on Computer Arithmetic, 1063-6889/16 $31.00 © 2016 IEEE, DOI 10.1109/ARITH.2016.8

[2] Sonam Negi, Pitchaiah Madduri, Implementation of High Speed Radix-10 Parallel Multiplier using Verilog, 978-1-4799-1743-3/15/$31.00 ©2015 IEEE

[3] Gorgin, Saeid; Jaberipur, Ghassem; Hashemi Asl, Reza, Efficient ASIC & FPGA Implementation for Binary-Coded Decimal Digit Multipliers, Circuits, Academic Journal, Springer link, Systems & Signal Processing;Dec2014, Vol. 33 problem 12, p3883

[4] Arvind Kumar Mehta, Mukesh Gupta, 1Vipin Jain, 2Sudhir kumar, High Performance Vedic BCD Multiplier & Modified Binary to BCD Converter, 2013 Annual IEEE India Conference (INDICON), 978-1-4799-2275-8/13/ ©2015 IEEE

[5] Carlos Eduardo Minchola Guardia, IMPLEMENTATION for A FULLY PIPELINED BCD MULTIPLIER IN FPGA, 978-1-4673-0186-2/14/©2014 IEEE

[6] H. C. Neto & M. P. Vestias. Decimal multiplier on FPGA using embedded binary multipliers. In Proc. Int. Conf. Field Programmable Logic & Applications FPL 2008, pages 197–202, 2008.

[7] E. M. Schwarz, J. S. Kapernick, & M. F. Cowlishaw. Decimal floating-point support on IBM System z10 processor, 2009. IBM Journal for Research & Development.

[8] G. Sutter, E. Todorovich, G. Bioul, M. Vazquez, & J.-P. Deschamps. FPGA Implementations for BCD Multipliers. In Proc. Int. Conf. Reconfigurable Computing & FPGAs ReConFig '09, pages 36–41, 2009.

[9] C. Tsen, S. Gonzalez-Navarro, & M. Schulte. Hardware design for a Binary Integer Decimal-based floating-point adder, 2007. Computer Design, 2007. ICCD 2007. 25th International Conference on.

[10] Liu Han & Seok-Bum, "High-Speed Parallel Decimal Multiplication with Redundant Internal Encodings," IEEE Trans. Computers, Vol. 62, no. 5, pp. 956 – 968, May 2013.

[11] G. Jaberipur & A. Kaivani, "Binary-coded decimal digit multipliers," IET J. Computers & Digital Techniques, Vol. 1 , no. 4, pp. 377 – 381, July 2007.