ISSN2394-3777 (Print)
ISSN2394-3785 (Online)
Available online atwww.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 5, Special Issue 11, April 2018

DESIGN OF FAST DADDA MULTIPLIER USING VEDIC MATHEMATICS

B.Shanmathi[1] PG Scholar
Department of ECE
Sri Venkateswara College of Engineering
Sriperumbudhur-602117
b.shanmathi1994@gmail.com

Dr.G.A.Sathishkumar[2] professor
Department of ECE
Sri Venkateswara College of Engineering
Sriperumbudhur-602117
sathish@svce.ac.in

***Abstract*— In VLSI design, the performance of any system is based on the performance of the multiplier. But multipliers are the most area and power consuming circuits. Improvement in any of these parameters increases the performance of the multiplier. This paper proposes the Dadda multiplier, in which the partial product and adder stage are performed using one of the sutras of the Vedic mathematics. This paper presents a Novel approach towards the reduction of delay in Dadda multiplier by using Urdhva Triyakbhyam and the partial product addition in realized using Ripple Carry Adder. The Vedic multiplier with Ripple carry adder has been designed using Verilog HDL and simulated in Xilinx ISE simulator 14.1 and also synthesized using Cadence EDA tool. The Dadda multiplier are compared with existing literature based on path delay and the result shows that proposed Dadda multiplier with Vedic mathematics are faster with least path delay.**

Keywords-Vedic mathematics, Dadda Multiplier, Delay, Power consumption, Urdhva Triyakbhyam

I. INTRODUCTION

A multiplier is one of the major blocks in digital signal processing, ALU and other logic computations. The speed of the processor determines its performance. High speed processing is the imperative requirement of all the systems. In modern VLSI design, the trade of the high speed multiplication is ever increasing. Multiplier is the essential component in digital signal processing application and thus the speed of the circuit is based on the design of the multiplier. Prior works some multipliers were considered as high speed multiplier example Booth multiplier, Modified Booth multiplier, Array multiplier etc. Although these multipliers involves many intermediate steps, which reduces their speed and consumes more power. The demand of the fast multiplication has given rise to the fast multiplier called Dadda multiplier .

To increase the performance of the multiplier as well as for the high speed multiplication, Vedic mathematics is used. Vedic mathematics is a collection of sutras which consists of 16 sutras and 13 sub sutras. These sutras are used to solve problems such as arithmetic, algebra, geometry, calculus. Indian mathematician Jagadguru Shri Bharathi Krishna Tirthaji discovered the Vedic mathematics. Veda is a Sanskrit word which means Knowledge. Using regular mathematical steps, solving problems sometimes leads to more complex and power consuming. By using Vedic mathematics General Techniques and Specific Techniques, numerical calculation can be done at very fast speed.

A. Dadda multiplier

The scientist Luigi Dadda was designed the hardware multiplier known as Dadda multiplier. The Dadda multiplier is identical to the Wallace multiplier, although it is little faster and the number of gates required is also less compared to Wallace tree multiplier. Both of these multipliers consist of three stages. In the first stage, the product matrix is formed. In the second stage, this product matrix is reduced to a height of two rows. In the final stage, these two rows are combined using an appropriate adder. Even though in the Wallace multiplier, the products are reduced as fast as possible. In contrast, Dadda multiplier does the minimum reduction necessary at each level to perform the reduction of the product in the same number of levels as required by a Wallace multiplier. Generally it is considered that, both the multipliers exhibit similar delay. Because to perform the partial product reduction both the multiplier uses the same number of pseudo adder levels.

B. Vedic Mathematics–Sutra

Vedic mathematics is the technique which can be used for performing multiplication operation with the benefit of several sutras. Urdhva Triyakbhyam (UT) is one of the 16 different sutras based on Vedic mathematical sutras for multiplication. The "Urdhva Triyakbhyam" is derived from the Sanskrit name "Urdhva" means "Vertical" and "Triyakbhyam" means "Crosswise". By applying vertically crosswise technique the generation of partial product and the addition are performed in a single stage. Because of this parallel operation of partial product generation and addition the speed of the multiplier is increased. In this approach which multiplies the digits vertically and crosswise and finally adds them using adder. The advantage of this method is that the partial products needed for the multiplication are already generated and this leads to decrease in delay and power consumption. The Vedic mathematics can be used to compute binary numbers as well as decimal numbers. This method can also be directly applied to trigonometry, spherical geometry, calculus and applied mathematics of various kinds. The different branches of engineering such as Convolution, Cryptography and Digital Signal Processing are practiced based on Vedic sutras.

C. Adder

Inside the Vedic sutra block to combine the partial product Ripple Carry Adder is used. A Ripple Carry Adder is a digital circuit which is used to produce the sum of two numbers. It can be constructed by connecting the full adder in series, with the carry output from each full adder stage can be connected as the carry input to the next stage of full adder. In the ripple carry adder, the output is known after the carry generated from the previous stage. Thus the sum of the most significant bit is only available after the carry signal has rippled from the least significant bit through the adder stages.

II. FAST DADDA MULTIPLIER USING URDHVA TRIYAKBHYAM

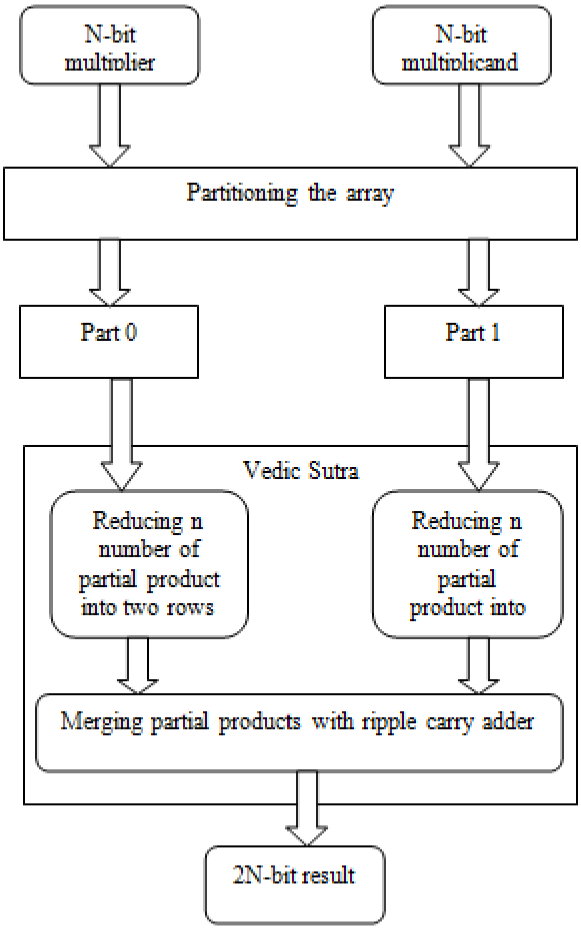

N-bit multiplier
N-bit multiplicand
Partitioning the array
Part 0
Part 1
Vedic Sutra
Reducing n number of partial product into two rows
Reducing n number of partial product into
Merging partial products with ripple carry adder
2N-bit result

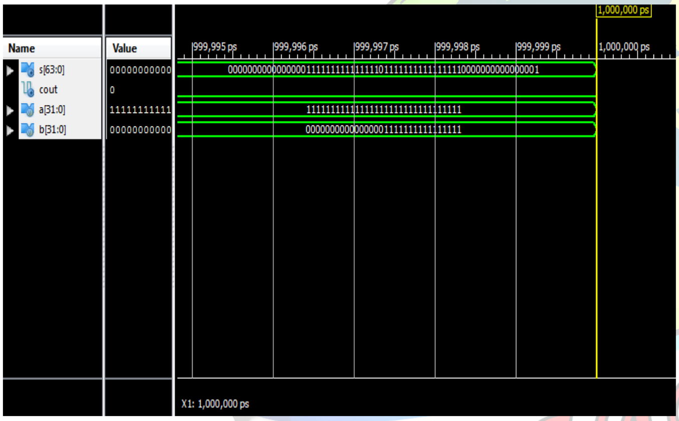
Figure.1.Fast Dadda multiplier with Vedic Sutra

Dadda multiplier proposed a predetermined method for matrix to reduce the number of stages to the minimum matrix height. In the regular Dadda multiplier, the total delay due to the product matrix generation, the product summation stage and finally due to the final adder stage. Among all these three stages, the product summation stage has maximum delay compared to other two stages. In this proposed paper, the maximum delay obtained in the product summation stage can be reduced with the help of one of the Vedic sutras called Urdhva Triyakbhyam. In the fast Dadda multiplier, the array can be partitioned into two parts as part 0 and part 1 at the first stage. At the next stage, the partial product and final adder stage can be performed at the same stage with the help of Vedic sutra. Because of the partial product and adder are performed at the same stage reduce the delay to the minimum compared to the regular Dadda multiplier. The adder used in the Vedic sutra is Ripple Carry Adder. A Ripple Carry Adder is a digital circuit which is used to produce the sum of two numbers. It can be constructed by connecting the full adder in series, with the carry output from each full adder stage can be connected as the carry input to the next stage of full adder.

III. RESULTS AND DISCUSSION

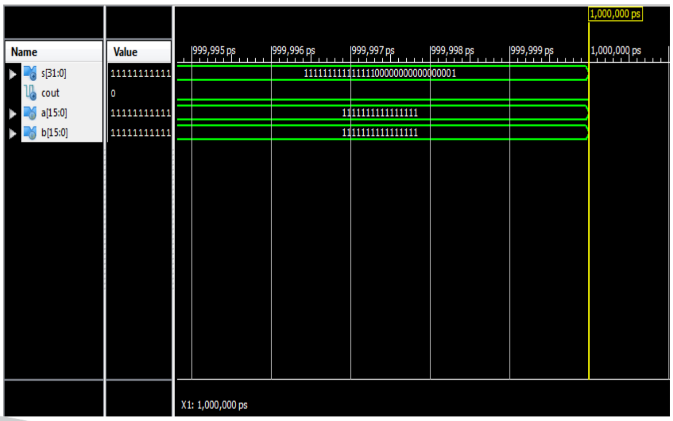
A. SIMULATION RESULT USING XILINX ISE SIMULATOR

In this section the simulation result of the Fast Dadda multiplier using Vedic is described as follows. Figure 2 represent the simulation result of 16-bit Dadda multiplier using Vedic is given. Figure 3 represent the simulation result of 32-bit Dadda multiplier using Vedic is given. Figure 4 represent the simulation result of 64-bit Dadda multiplier using Vedic is given. Figure 5 represent the simulation result of 128-bit Dadda multiplier using Vedic .



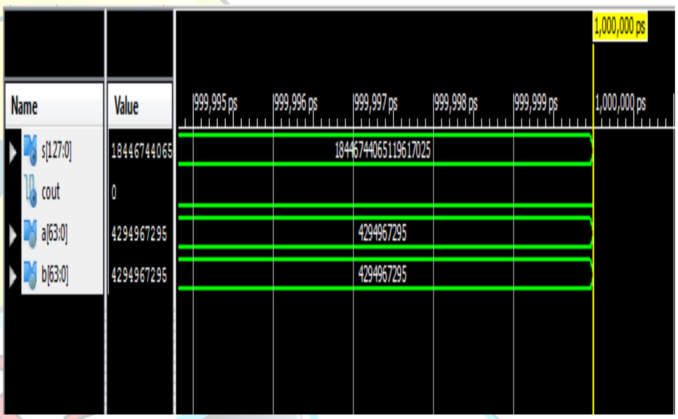
s[63:0] 00000000000 0000000000000000111111111111111011111111111111111000000000000000001
cout 0
a[31:0] 11111111111 11111111111111111111111111111111
b[31:0] 00000000000 00000000000000001111111111111111

Figure.2. Simulation result for 16 bit Fast Dadda multiplier using Vedic



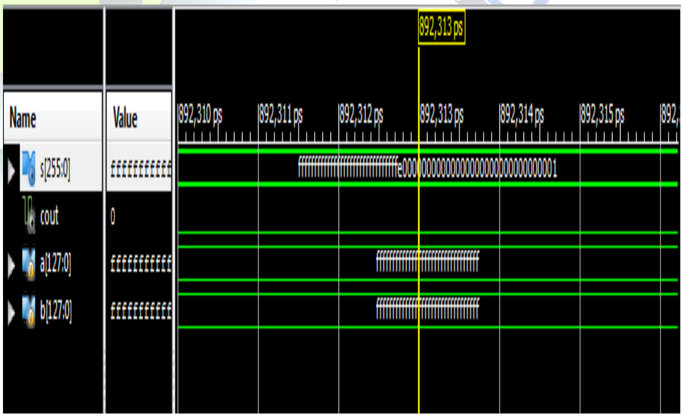
s[31:0] 11111111111 11111111111111100000000000000001
cout 0
a[15:0] 11111111111 1111111111111111
b[15:0] 11111111111 1111111111111111

Figure.3. Simulation result for 32 bit Fast Dadda multiplier using Vedic



s[127:0] 18446744065 18446744065119617025
cout 0
a[63:0] 4294967295 4294967295
b[63:0] 4294967295 4294967295

Figure.4. Simulation result for 64 bit Fast Dadda multiplier using Vedic

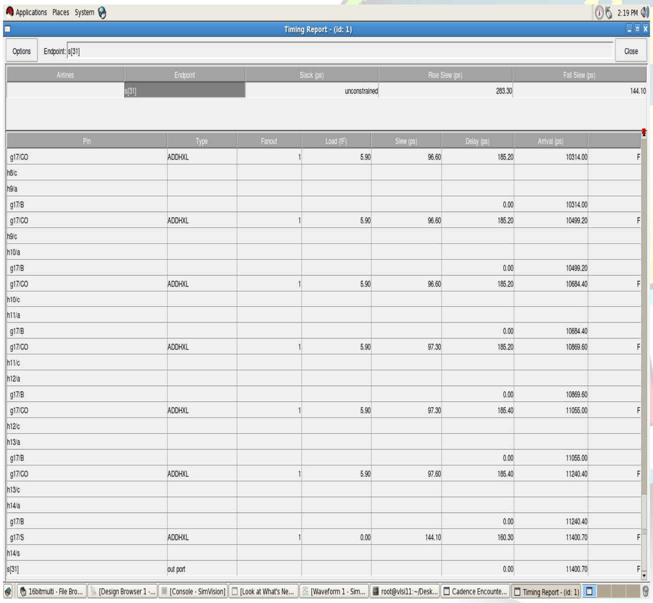


s[255:0] ffffffffff ffffffffffffffffffffffffffffffffe0000000000000000000000000000001
cout 0
a[127:0] ffffffffff ffffffffffffffffffffffffffffffff
b[127:0] ffffffffff ffffffffffffffffffffffffffffffff

Figure.5. Simulation result for 128 bit Fast Dadda multiplier using Vedic

B. SYNTHESIS REPORT USING CADENCE EDA TOOL

In this section the synthesis report for the Fast Dadda multiplier is described as follows. Figure 6 represent the delay report for the 16-bit Fast Dadda multiplier. Figure 7 represent the power report for the 16-bit Fast Dadda multiplier. Figure 8 represent the delay report for the 32-bit Fast Dadda multiplier. Figure 9 represent the power report for the 32-bit Fast Dadda multiplier. Figure 10 represent the delay report for the 64-bit Fast Dadda multiplier. Figure 11 represent the power report for the 64-bit Fast Dadda multiplier. Figure 12 represent the delay report for the 128-bit Fast Dadda multiplier. Figure 13 represent the power report for the 128-bit Fast Dadda multiplier.

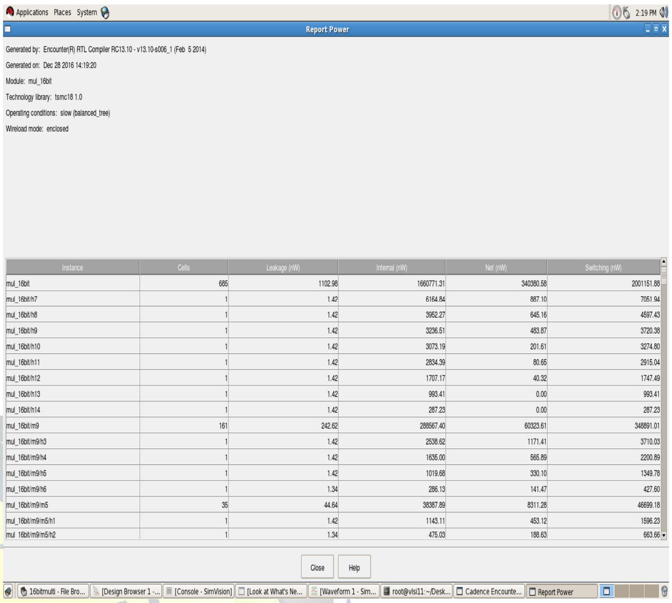


Endpoint: s[31]

Airlines	Endpoint	Slack (ps)	Rise Slew (ps)	Fall Slew (ps)
	s[31]	unconstrained	283.30	144.10

Pin	Type	Fanout	Load (fF)	Slew (ps)	Delay (ps)	Arrival (ps)	
g17/CO	ADDHXL	1	5.90	96.60	185.20	10314.00	F
h8/c							
h9/a							
g17/B					0.00	10314.00	
g17/CO	ADDHXL	1	5.90	96.60	185.20	10499.20	F
h9/c							
h10/a							
g17/B					0.00	10499.20	
g17/CO	ADDHXL	1	5.90	96.60	185.20	10684.40	F
h10/c							
h11/a							
g17/B					0.00	10684.40	
g17/CO	ADDHXL	1	5.90	97.30	185.20	10869.60	F
h11/c							
h12/a							
g17/B					0.00	10869.60	
g17/CO	ADDHXL	1	5.90	97.30	185.40	11055.00	F
h12/c							
h13/a							
g17/B					0.00	11055.00	
g17/CO	ADDHXL	1	5.90	97.60	185.40	11240.40	F
h13/c							
h14/a							
g17/B					0.00	11240.40	
g17/S	ADDHXL	1	0.00	144.10	160.30	11400.70	F
h14/s							
s[31]	out port				0.00	11400.70	F

Figure.6: Delay report for 16-bit Dadda multiplier using Vedic



Generated by: Encounter(R) RTL Compiler RC13.10 - v13.10-s006_1 (Feb 5 2014)

Generated on: Dec 28 2016 14:19:20

Module: mul_16bit

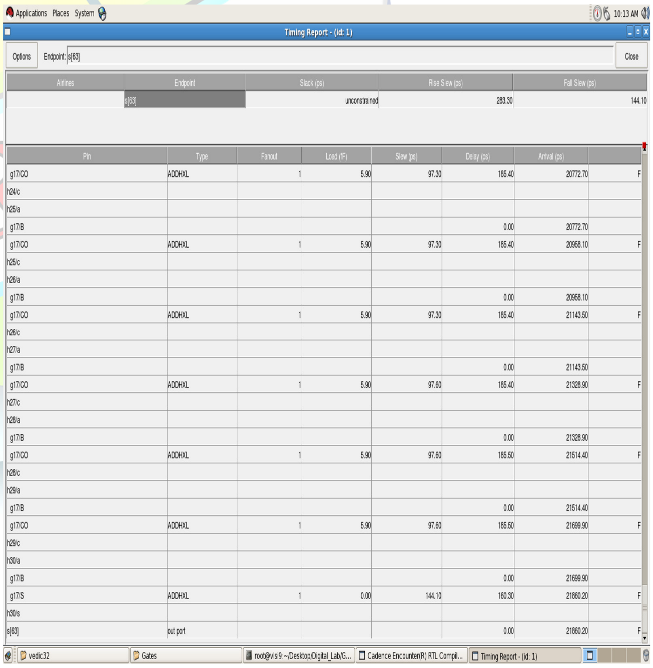
Technology library: tsmc18 1.0

Operating conditions: slow (balanced_tree)

Wireload mode: enclosed

Instance	Cells	Leakage (nW)	Internal (nW)	Net (nW)	Switching (nW)
mul_16bit	685	1102.98	1660771.31	340380.58	2001151.88
mul_16bit/h7	1	1.42	6164.84	887.10	7051.94
mul_16bit/h8	1	1.42	3952.27	645.16	4597.43
mul_16bit/h9	1	1.42	3236.51	483.87	3720.38
mul_16bit/h10	1	1.42	3073.19	201.61	3274.80
mul_16bit/h11	1	1.42	2834.39	80.65	2915.04
mul_16bit/h12	1	1.42	1707.17	40.32	1747.49
mul_16bit/h13	1	1.42	993.41	0.00	993.41
mul_16bit/h14	1	1.42	287.23	0.00	287.23
mul_16bit/m9	161	242.62	288567.40	60323.61	348891.01
mul_16bit/m9/h3	1	1.42	2538.62	1171.41	3710.03
mul_16bit/m9/h4	1	1.42	1635.00	565.89	2200.89
mul_16bit/m9/h5	1	1.42	1019.68	330.10	1349.78
mul_16bit/m9/h6	1	1.34	286.13	141.47	427.60
mul_16bit/m9/m5	35	44.64	38387.89	8311.28	46699.18
mul_16bit/m9/m5/h1	1	1.42	1143.11	453.12	1596.23
mul_16bit/m9/m5/h2	1	1.34	475.03	188.63	663.66

Figure.7: Power report for 16-bit Dadda multiplier using Vedic

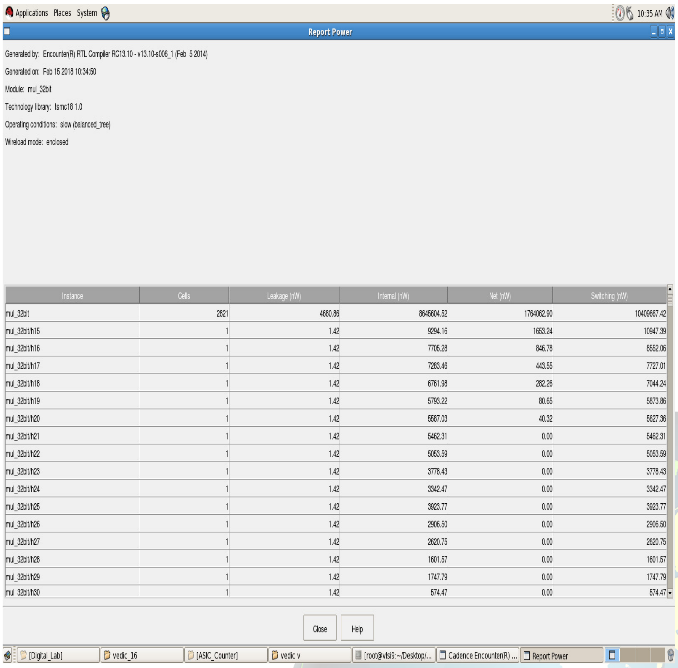

Applications Places System
10:13 AM
Timing Report - (id: 1)
Options
Close
vedic32
Gates
root@vlsi9:~/Desktop/Digital_Lab/G...
Cadence Encounter(R) RTL Compil...
Timing Report - (id: 1)

Endpoint: s[63]

Airlines	Endpoint	Slack (ps)	Rise Slew (ps)	Fall Slew (ps)
	s[63]	unconstrained	283.30	144.10

Pin	Type	Fanout	Load (fF)	Slew (ps)	Delay (ps)	Arrival (ps)	
g17/CO	ADDHXL	1	5.90	97.30	185.40	20772.70	F
h24/c							
h25/a							
g17/B					0.00	20772.70	
g17/CO	ADDHXL	1	5.90	97.30	185.40	20958.10	F
h25/c							
h26/a							
g17/B					0.00	20958.10	
g17/CO	ADDHXL	1	5.90	97.30	185.40	21143.50	F
h26/c							
h27/a							
g17/B					0.00	21143.50	
g17/CO	ADDHXL	1	5.90	97.60	185.40	21328.90	F
h27/c							
h28/a							
g17/B					0.00	21328.90	
g17/CO	ADDHXL	1	5.90	97.60	185.50	21514.40	F
h28/c							
h29/a							
g17/B					0.00	21514.40	
g17/CO	ADDHXL	1	5.90	97.60	185.50	21699.90	F
h29/c							
h30/a							
g17/B					0.00	21699.90	
g17/S	ADDHXL	1	0.00	144.10	160.30	21860.20	F
h30/s							
s[63]	out port				0.00	21860.20	F

Figure.8: Delay report for 32-bit Dadda multiplier using Vedic



Generated by: Encounter(R) RTL Compiler RC13.10 - v13.10-s006_1 (Feb 5 2014)

Generated on: Feb 15 2018 10:34:50

Module: mul_32bit

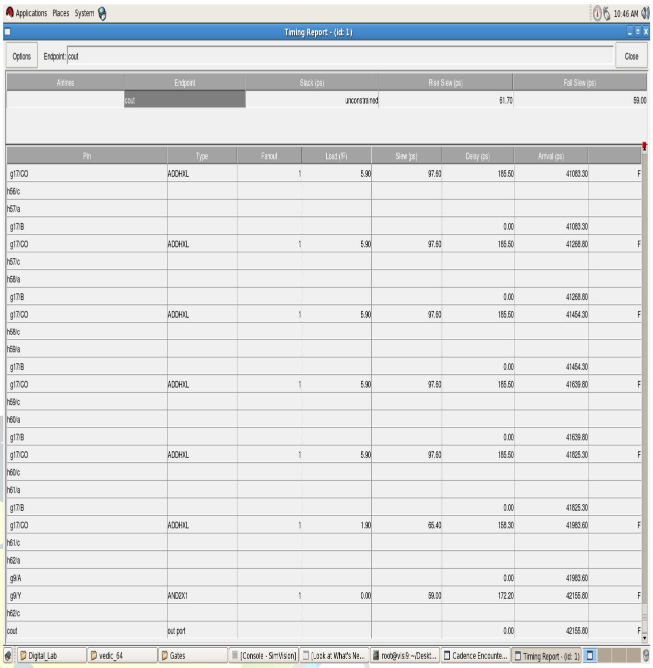
Technology library: tsmc18 1.0

Operating conditions: slow (balanced_tree)

Wireload mode: enclosed

Instance	Cells	Leakage (nW)	Internal (nW)	Net (nW)	Switching (nW)
mul_32bit	2821	4680.86	8645604.52	1764062.90	10409667.42
mul_32bit/h15	1	1.42	9294.16	1653.24	10947.39
mul_32bit/h16	1	1.42	7705.28	846.78	8552.06
mul_32bit/h17	1	1.42	7283.46	443.55	7727.01
mul_32bit/h18	1	1.42	6761.98	282.26	7044.24
mul_32bit/h19	1	1.42	5793.22	80.65	5873.86
mul_32bit/h20	1	1.42	5587.03	40.32	5627.36
mul_32bit/h21	1	1.42	5462.31	0.00	5462.31
mul_32bit/h22	1	1.42	5053.59	0.00	5053.59
mul_32bit/h23	1	1.42	3778.43	0.00	3778.43
mul_32bit/h24	1	1.42	3342.47	0.00	3342.47
mul_32bit/h25	1	1.42	3923.77	0.00	3923.77
mul_32bit/h26	1	1.42	2906.50	0.00	2906.50
mul_32bit/h27	1	1.42	2620.75	0.00	2620.75
mul_32bit/h28	1	1.42	1601.57	0.00	1601.57
mul_32bit/h29	1	1.42	1747.79	0.00	1747.79
mul 32bit/h30	1	1.42	574.47	0.00	574.47

Figure.9: Power report for 32-bit Dadda multiplier using Vedic

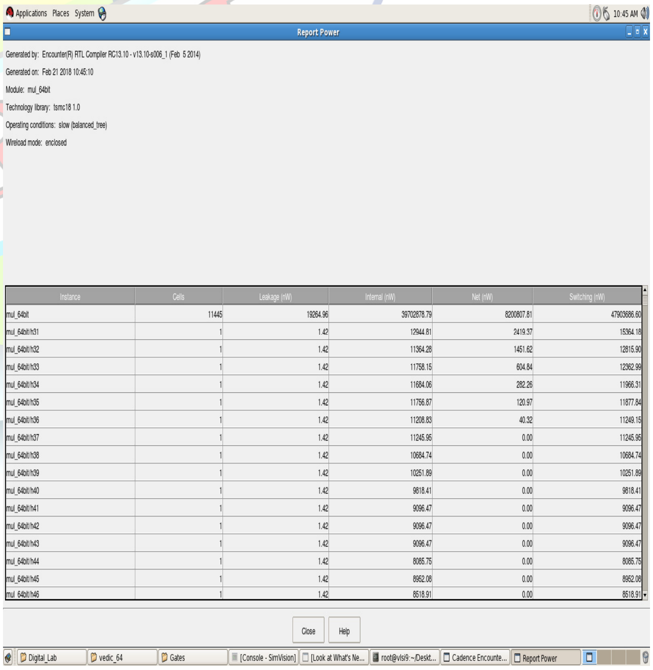


Endpoint: cout

Airlines	Endpoint	Slack (ps)	Rise Slew (ps)	Fall Slew (ps)
	cout	unconstrained	61.70	59.00

Pin	Type	Fanout	Load (fF)	Slew (ps)	Delay (ps)	Arrival (ps)	
g17/CO	ADDHXL	1	5.90	97.60	185.50	41083.30	F
h56/c							
h57/a							
g17/B					0.00	41083.30	
g17/CO	ADDHXL	1	5.90	97.60	185.50	41268.80	F
h57/c							
h58/a							
g17/B					0.00	41268.80	
g17/CO	ADDHXL	1	5.90	97.60	185.50	41454.30	F
h58/c							
h59/a							
g17/B					0.00	41454.30	
g17/CO	ADDHXL	1	5.90	97.60	185.50	41639.80	F
h59/c							
h60/a							
g17/B					0.00	41639.80	
g17/CO	ADDHXL	1	5.90	97.60	185.50	41825.30	F
h60/c							
h61/a							
g17/B					0.00	41825.30	
g17/CO	ADDHXL	1	1.90	65.40	158.30	41983.60	F
h61/c							
h62/a							
g9/A					0.00	41983.60	
g9/Y	AND2X1	1	0.00	59.00	172.20	42155.80	F
h62/c							
cout	out port				0.00	42155.80	F

Figure.10: Delay report for 64-bit Dadda multiplier using Vedic



Generated by: Encounter(R) RTL Compiler RC13.10 - v13.10-s006_1 (Feb 5 2014)

Generated on: Feb 21 2018 10:45:10

Module: mul_64bit

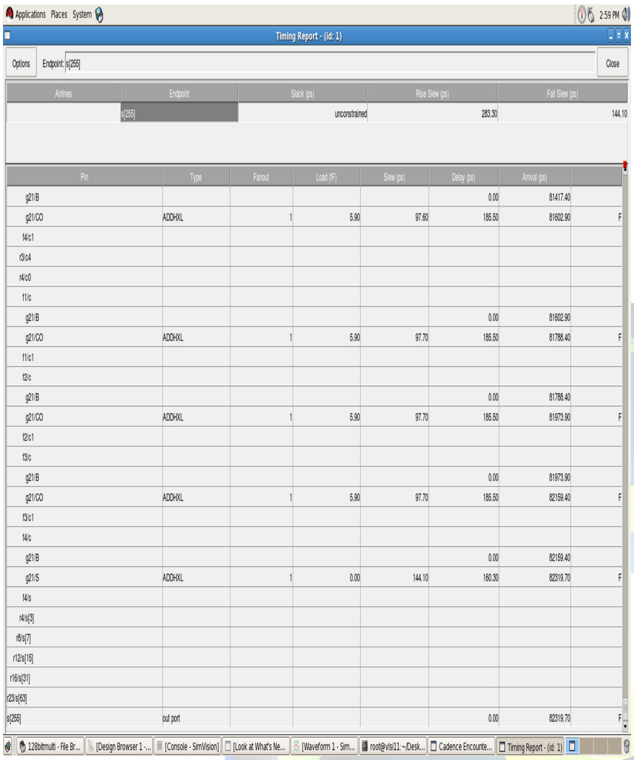
Technology library: tsmc18 1.0

Operating conditions: slow (balanced_tree)

Wireload mode: enclosed

Instance	Cells	Leakage (nW)	Internal (nW)	Net (nW)	Switching (nW)
mul_64bit	11445	19264.96	39702878.79	8200807.81	47903686.60
mul_64bit/h31	1	1.42	12944.81	2419.37	15364.18
mul_64bit/h32	1	1.42	11364.28	1451.62	12815.90
mul_64bit/h33	1	1.42	11758.15	604.84	12362.99
mul_64bit/h34	1	1.42	11684.06	282.26	11966.31
mul_64bit/h35	1	1.42	11756.87	120.97	11877.84
mul_64bit/h36	1	1.42	11208.83	40.32	11249.15
mul_64bit/h37	1	1.42	11245.95	0.00	11245.95
mul_64bit/h38	1	1.42	10684.74	0.00	10684.74
mul_64bit/h39	1	1.42	10251.89	0.00	10251.89
mul_64bit/h40	1	1.42	9818.41	0.00	9818.41
mul_64bit/h41	1	1.42	9096.47	0.00	9096.47
mul_64bit/h42	1	1.42	9096.47	0.00	9096.47
mul_64bit/h43	1	1.42	9096.47	0.00	9096.47
mul_64bit/h44	1	1.42	8085.75	0.00	8085.75
mul_64bit/h45	1	1.42	8952.08	0.00	8952.08
mul 64bit/h46	1	1.42	8518.91	0.00	8518.91

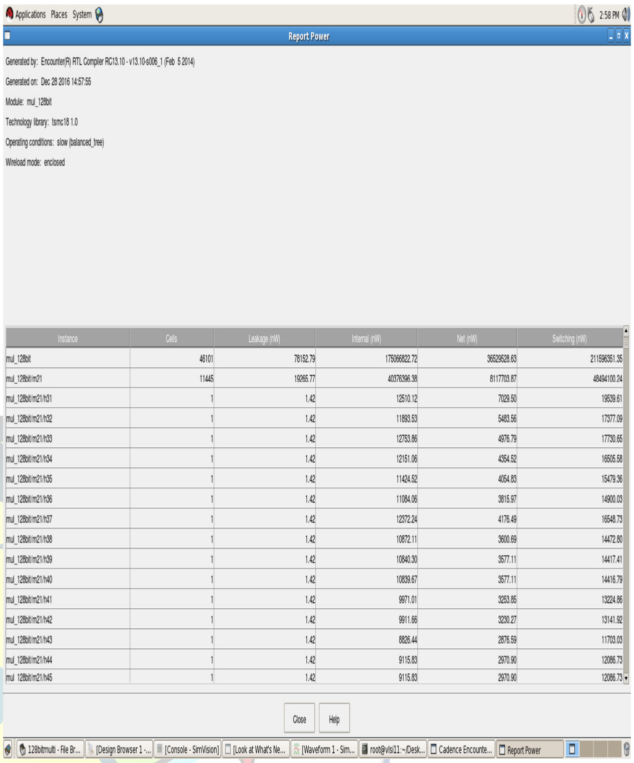
Figure.11: Power report for 64-bit Dadda multiplier using Vedic



Airlines	Endpoint	Slack (ps)	Rise Slew (ps)	Fall Slew (ps)
	s[255]	unconstrained	283.30	144.10

Pin	Type	Fanout	Load (fF)	Slew (ps)	Delay (ps)	Arrival (ps)	
g21/B					0.00	81417.40	
g21/CO	ADDHXL	1	5.90	97.60	185.50	81602.90	F
f4/c1							
r3/c4							
r4/c0							
f1/c							
g21/B					0.00	81602.90	
g21/CO	ADDHXL	1	5.90	97.70	185.50	81788.40	F
f1/c1							
f2/c							
g21/B					0.00	81788.40	
g21/CO	ADDHXL	1	5.90	97.70	185.50	81973.90	F
f2/c1							
f3/c							
g21/B					0.00	81973.90	
g21/CO	ADDHXL	1	5.90	97.70	185.50	82159.40	F
f3/c1							
f4/c							
g21/B					0.00	82159.40	
g21/S	ADDHXL	1	0.00	144.10	160.30	82319.70	F
f4/s							
r4/s[3]							
r8/s[7]							
r12/s[15]							
r16/s[31]							
r23/s[63]							
s[255]	out port				0.00	82319.70	F

Figure.12: Delay report for 128-bit Dadda multiplier using Vedic



Generated by: Encounter(R) RTL Compiler RC13.10 - v13.10-s006_1 (Feb 5 2014)

Generated on: Dec 28 2016 14:57:55

Module: mul_128bit

Technology library: tsmc18 1.0

Operating conditions: slow (balanced_tree)

Wireload mode: enclosed

Instance	Cells	Leakage (nW)	Internal (nW)	Net (nW)	Switching (nW)
mul_128bit	46101	78152.79	175066822.72	36529528.63	211596351.35
mul_128bit/m21	11445	19265.77	40376396.38	8117703.87	48494100.24
mul_128bit/m21/h31	1	1.42	12510.12	7029.50	19539.61
mul_128bit/m21/h32	1	1.42	11893.53	5483.56	17377.09
mul_128bit/m21/h33	1	1.42	12753.86	4976.79	17730.65
mul_128bit/m21/h34	1	1.42	12151.06	4354.52	16505.58
mul_128bit/m21/h35	1	1.42	11424.52	4054.83	15479.36
mul_128bit/m21/h36	1	1.42	11084.06	3815.97	14900.03
mul_128bit/m21/h37	1	1.42	12372.24	4176.49	16548.73
mul_128bit/m21/h38	1	1.42	10872.11	3600.69	14472.80
mul_128bit/m21/h39	1	1.42	10840.30	3577.11	14417.41
mul_128bit/m21/h40	1	1.42	10839.67	3577.11	14416.79
mul_128bit/m21/h41	1	1.42	9971.01	3253.85	13224.86
mul_128bit/m21/h42	1	1.42	9911.66	3230.27	13141.92
mul_128bit/m21/h43	1	1.42	8826.44	2876.59	11703.03
mul_128bit/m21/h44	1	1.42	9115.83	2970.90	12086.73
mul 128bit/m21/h45	1	1.42	9115.83	2970.90	12086.73

Figure.13: Power report for 128-bit Dadda multiplier using Vedic

IV. PERFORMANCE COMPARISON

A. DELAY SUMMARIZATION USING CADENCE EDA TOOL

The maximum combinational path delay obtained for the Dadda multiplier with Vedic sutra and without Vedic sutra using Cadence EDA tool is summarized in Table 1 and their respective comparison representation is shown in figure 14. The proposed Dadda Multiplier is coded using Verilog HDL and simulated using Xilinx ISE simulator 14.1. Synthesis has been performed using Cadence EDA tool.

Table.1: Delay summarization using Cadence EDA tool

DELAY	16-bit	32-bit	64-bit	128-bit
Dadda multiplier with Vedic	11400ps	20772.70ps	41268.20ps	81417.40ps
Dadda multiplier				

without Vedic	20123.50ps	32569.30ps	53789.20ps	98472.90ps

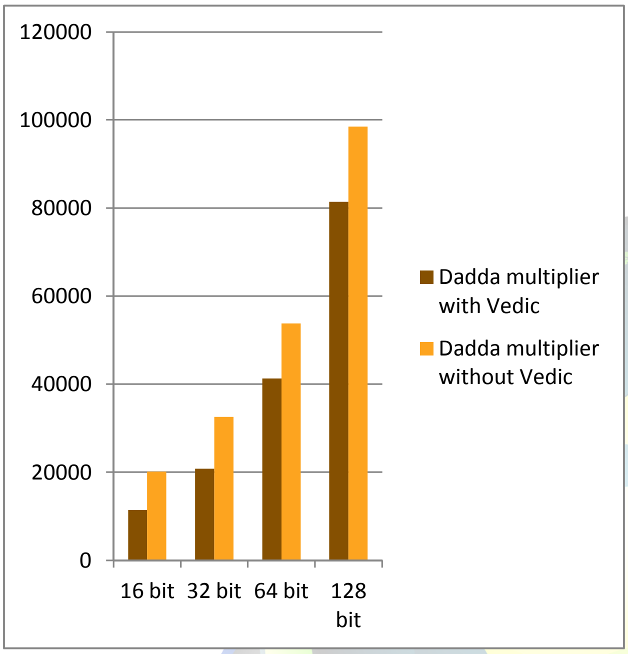


Figure.14: Delay comparison of Dadda multiplier with Vedic and without Vedic

B. POWER SUMMARIZATION USING CADENCE EDA TOOL

The maximum power consumption for the Dadda multiplier with Vedic sutra and without Vedic sutra using Cadence EDA tool is summarized in Table 2 and their respective comparison representation is shown in figure 15. The proposed Dadda Multiplier is coded using Verilog HDL and simulated using Xilinx ISE simulator 14.1. Synthesis has been performed using Cadence EDA tool.

Table.2: Power summarization using Cadence EDA tool

DELAY	16-bit	32-bit	64-bit	128-bit
Dadda multiplier with Vedic	340380.50 nw	1764062.90 nw	8200686.04 nw	36529528.13 Nw
Dadda multiplier without Vedic	575755 nw	2418316.20 nw	11926461.5 nw	43485600.21 Nw

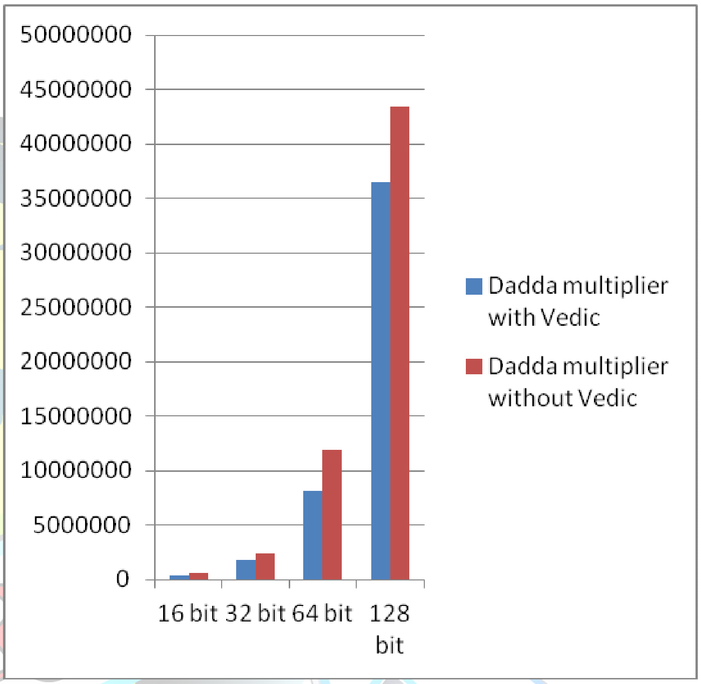


Figure.15: Power comparison of Dadda multiplier with Vedic and without Vedic

V. CONCLUSION

A technique for the multiplication of 128 bit operands with the help of Vedic sutra is described. The Proposed Dadda multiplier is based on Urdhva Triyakbhyam sutra of Vedic mathematics. This sutra makes the parallel generation of partial product and removes unwanted multiplication steps. In this proposed design, the maximum delay obtained in the product summation stage can be reduced with the help of one of the Vedic sutras called Urdhva

Triyakbhyam. This fast Dadda multiplier works in two stages. At the first stage, the array can be partitioned into two parts as part 0 and part 1 . At the next stage, the partial product and final adder stage can be performed at the same stage with the help of Vedic sutra. Because of the partial product and adder are performed at the same stage reduce the delay to the minimum compared to the regular Dadda multiplier. In this paper, the design of the Fast Dadda multiplier with Vedic mathematics has been proposed. The synthesis and simulation result of the Fast Dadda multiplier with and without Vedic sutras are tabulated. The comparison of Dadda multiplier with and without Vedic multiplier showed that Vedic multiplier is more efficient in terms of delay and power consumption.

REFERENCES

[1]. Rajasree Rao and N. Srinivas, "Design and analysis of enhanced dadda multiplier using 5:2 compressor". International Journal on Computer Science and Engineering, ISSN : 0975-3397 Vol. 9 No.06 Jun 2017.

[2]. Yogita Bansal,Charu Madhu,"A novel high speed approach for 16x16 Vedic multiplication with compressor adders", ELSEVIER Computers and Electrical Engineering 4939-49,2016.

[3]. Ritupriya Jha,Ranbir Kumar Paul, "Swift and Approximate Multiply and Accumulate Unit for Embedded DSP Applications", International Journal of Industrial Electronics and Electrical Engineering, ISSN: 2347-6982 Volume-3, Issue-1, Jan.-2015.

[4]. Abhay Sharma, "FPGA Implementation of a High Speed Multiplier Employing Carry Lookahead Adders in Reduction Phase", International Journal of Computer Applications 116(17):27-31, April 2015.

[5]. G.Radhika, D.R.V.A.Sharath Kumar, "Design of Multiplier and Divider Using Reversible Logic Gates with Vedic Mathematical Approach", International Journal for Research in Applied Science and Engineering Technology, ISSN: 2321-965 Vol. 2 Issue IX, September 2014.

[6]. K.N.V.S.Vijaya lakshmi and D.R.Sandeep, LowPower 32-bit Dadda multiplier". International Journal of Computer Trends and Technology (IJCTT), volume 17, November 2014.

[7]. B. Ramkumar, V. Sreedeep and Harish M Kittur, "A Design Technique for Faster Dadda Multiplier", IEEE 2014.

[8]. Ravish Aradhya H.V, Lakshmesha J and Muralidhara K.N, "Reduced Complexity Hybrid Ripple Carry Look ahead Adder", International Journal of Computer Applications (0975 – 8887) Volume 70– No.28, May 2013.

[9]. W. Chu, A.I. Unwala, P. Wu & E.E. Swartzlander, Jr., "Implementation of a High Speed Multiplier Using Carry Lookahead Adders," Asilomar, pp. 400-404, 2013.

[10]. Rajkumar Sarma and Veerati Raju , "Design and performance analysis of hybrid adders for high speed arithmetic circuit", International journal of VLSI design and communication systems Vol.3,No.3,2012.

[11]. Aniruddha Kanhe and Shishir Kumar Das, "Design and Implementation of Low Power Multiplier Using Vedic Multiplication Technique", International Journal of Computer Science and Communication.Vol.3, No.1,June 2012.

[12]. Girish V A and Kerur S S , "Implementation of Vedic Multiplier For Digital Signal Processing", International Conference on VLSI, Communication & Instrumentation (ICVCI) 2011.

[13]. Honey Durga Tiwari and Yong Beom Cho, "Multiplier design based on ancient Indian Vedic Mathematics", Dept of Electronics Engineering, Kongkuk University 2011.

[14]. B.Ramkumar, Harish M Kittur and P.Mahesh Kannan, "ASIC Implementation of Modified Faster Carry Save Adder", European Journal of Scientific Research, Vol. 42, Issue 1, 2010.

[15]. Sabyasachi Das and Sunil P.Khatri,"Generation of the Optimal Bit-Width Topology of the Fast Hybrid Adder in a Parallel Multiplier", International Conference on Integrated Circuit Design and Technology (ICICDT) May, 2007.

[16]. Bickerstaff, K.C."Optimization of Column Compression Multipliers" Doctoral Dissertation, Dept. of Electrical and Computer Engineering, University of Texas at Austin, Austin, Texas, 2007.

[17]. Paul F.Stelling, "Design strategies for optimal hybrid final adders in parallel multiplier", Journal of VLSI signal processing, vol 14, pp,321-331,1996.

[18]. E. J. King and E. E. Swartzlander, "Data-dependent truncation scheme for parallel multipliers," Conf. Rec. 31st Asilomar Conf. Signals, Syst.Comput., 1997.

[19]. V. G. Oklobdzija and D.Villeger, "Improving Multiplier Design by Using Improved Column Compression Tree and Optimized Final Adder in CMOS Technology", IEEE transactions on Very Large Scale Integration (VLSI) systems, Vol. 3, no. 2, June 1995.

[20]. Addanki Purna Ramesh,"Implementation of Dadda and Array MultiplierArchitectures Using Tanner Tool" International Journal of Computer Science Engineering Technology. ISSN : 2229-3345 Vol. 2 No. 2.