

DESIGN OF FAST DADDA MULTIPLIER USING VEDIC MATHEMATICS

B.Shanmathi¹ PG Scholar Department of ECE Sri Venkateswara College of Engineering Sriperumbudhur-602117 <u>b.shanmathi1994@gmail.com</u>

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 . Dr.G.A.Sathishkumar² professor Department of ECE Sri Venkateswara College of Engineering Sriperumbudhur-602117 <u>sathish@svce.ac.in</u>

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



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

999,997 ps



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

u cout d a[15:0]

11111111

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 .

							1,000,000 ps
Name	Value	1999,995 ps	999,996 ps	999,997 ps	999,998 ps	999,999 ps	1,000,000 ps
🕨 📑 s[63:0]	00000000000	00000000	00000000111111111	111111011111111111	11111100000000000	00001	
🗓 cout	0						
🕨 📷 a[31:0]	11111111111		111111111	111111111111111111111111111111111111111	11111		
Þ 📷 b[31:0]	0000000000		000000000	0000001111111111	11111		
		X1: 1 000 000 ps					

									1,000,000 ps
	Name	Value	_	1999,995 ps	999,996 ps	999,997 ps	1999,998 ps	1999,999 ps	1,000,000 ps
	🕨 😽 s[127:0]	18446744065			184	6744065119617025			
	🇓 cout	0							
	▶ 🔰 a[63:0]	4294967295				4294967295			
	🕨 🔰 b(63:0)	4294967295				4294967295			
L									

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

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

					892.313 ps			_
Name	Value	892,310 ps	892,311 ps	892,312 ps	892,313 ps	892,314 ps	892,315 ps	892,
▶ 😽 s[255:0]	tttttttttt			ffffffffffffffffffffe000	000000000000000000000000000000000000000	000000000000000000000000000000000000000		
11 ₆ cout	0							
▶ 闎 a[127:0]	fffffffffff							
▶ 👹 b[127:0]	fffffffffff							

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.

			Tini	ng Report - (id: 1)				
dans Endpoint: s[31]								Glose
Aitires		Endpoint		Stack (ps)	Rise :	Slew (ps)	Fail Slow (pc)	
	s[31]			unconstrain	ved	283.30		144
100		ADDHXL		1 5.90	96.60	185.20	10314.00	
7B						0.00	10314.00	
700		ADDH01.		1 5.90	96.60	185.20	10499.20	
8								
ħΒ						0.00	10499.20	
700		ADDHXL		1 5.90	96.60	185.20	10684.40	
e								
la								
7/B						0.00	10684.40	
700		ADDHXI,		1 5.90	97.30	185.20	10869.60	
ic .								
ia -								
7/B						0.00	10869.60	
700		ADDHXL		1 5.90	97.30	185.40	11055.00	
ic								
la								
21B						0.00	11055.00	
17/00		ADDHX),		1 5.90	97.60	185.40	11240.40	
ic								
3								
7B						0.00	11240.40	
7/S		ADDH01.		1 0.00	144.10	160.30	11400.70	
6								
4		cut port.				0.00	11400.70	

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

Abhreanna nacea ahaem 🖌					100 238
		Report Po	wer		88
Senerated by: Encounter(R) RTL Compiler RC	73.10 - v13.10-s006_1 (Feb 5 2014)				
Generated on: Dec 28 2016 14:19:20					
Module: mul_16bit					
Fechnology library: 1smc18 1.0					
Operating conditions: slow (balanced_tree)					
Wreload mode: enclosed					
Instance	Call.	Lastona MM.	internal (citi)	100 1010	Collection full)
nd 199	000	1923.00	1000771 01	100 (10) 941999 EE	2001151
4 19/07		142.20	1000/71.31	94000.00	2001101.
vi levina		142	9952 27	AIS 16	4507
nd (Rolling		1.42	3236.51	483.87	1720
nd 18kh10		1.42	9079 10	433.67	1274
nd 186611		1.42	2002.13	an ee	0E/4. 5015
N_10001111		1.42	1707.17	41.92	1747
nu_10001112		1.42	1707.11	47.30	1/4/.
10_1000/113		1.46	200.41	0.00	199.
nu_icoluni4		1.42	20/20	0.00	<i>a</i> /.
nu_teotim9	161	242.62	20006/.40	60323.61	348891.
nul_1606/m9/83	1	1.42	2538.62	11/1.41	3/10.
nul_16bitim9ih4	1	1.42	1635.00	565.89	2200.
nul_16bitim9ih5	1	1.42	1019.68	330.10	1349.
nul_16bit/m9/h6	1	1.34	286.13	141.47	427.
nul_16bitim9im5	35	44.64	36367.69	8311.28	46609.
nul_16bitim9im5ih1	1	1.42	1143.11	453.12	1596.
nul 16bitim9im6ih2	1	1.34	475.03	188.63	663.)
		Close	Help		
life II:	1-	1- 1-	12	Y	
r 🔄 🐧 16bitmulti - File Bro 🔍 (Desig	n Browser 1 🛛 Console - SimVision] 🔲 (Look at What's Ne 🧮 (Wavel	form 1 - Sim 📓 root@visi11:~/Desk	. Cadence Encounte	iort Power 🔲 🗖

🜒 🚯 16bitmulti - File Bro... 🐘 (Design B



			Timing Re	port - (id: 1)				
Options	Endpoint: [s[63]							Gase
	Altires	Endpoint	Slack	(25)	Rise Siew (;	6	Fall Siew (ps)	
	s josj			unconstrained		283.30		144.1
-	Pin	Туре	Farout	Load (IF)	Siew (ps)	Delay (ps)	Arival (ps)	
g17/00		ACOHIL	1	5.90	97.30	185.40	20772.70	F
12410								
h25/a								
g17/B						0.00	20772.70	
g17/00		ADDHKL	1	5.90	97.90	185.40	20958.10	F
125/c								
n26/a								
g17/B						0.00	20958.10	
g17/00		ADDHKL	1	5.90	97.30	185.40	21143.50	F
n26ic								
h27/a								
g17/B						0.00	21143.50	
g17/00		ADDHXL	1	5.90	97.60	185.40	21328.90	F
h27/c								
h28/a								
g17/B						0.00	21328.90	
g17/00		ADDHXL	1	5.90	97.60	185.50	21514.40	F
h28/c								
h29/a								
g17/B						0.00	21514.40	
g17/00		ADDHXL	1	5.90	97.60	185.50	21899.90	F
h29/c								
h30/a								
g17/B						0.00	21899.90	
g17/5		ACOHAL	1	0.00	144.10	160.30	21860.20	F
n30/s								
(FBI)		put port				0.00	21860.20	F.

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



		Report Pow	ier		
erated by: Encounter(R) RTL Compiler RC13.14	.0 - v13.10-s006_1 (Feb 5.2014)				
eraled on: Feb 15 2018 10:34:50					
ule: mul_3251					
hnology library: tsmc18 1.0					
eraling conditions: slow (balanced_tree)					
eload mode: enclosed					
Instance	Cets	Leakace (rW)	Internal (riW)	Nel (riW)	Switching (nW)
32bit	2621	4680.85	8645604.52	1764062.90	10409667.4
Chilth15	1	1.42	9294 16	1653.94	10947.3
32bil h16	1	1.42	7705.28	845.78	8552.0
32bit h17	1	1.42	7283.46	443.55	7727.0
32bith18	1	1.42	6761.98	282.26	7044.2
					19442
32bit h19	1	1.42	5793.22	80.65	5873.8
32bit h19 32bit h20	1	1.42	5793.22 5587.03	80.65 40.32	5873.8
32bilh19 32bilh20 32bilh21	1	1.42 1.42 1.42	5793.22 5687.00 5482.31	80.65 40.32 0.00	5873.8 9627.9 5462.9
32bith19 32bith20 32bith21 32bith22	1 1 1 1	1.42 1.42 1.42 1.42	5793.22 5687.05 5482.31 5053.59	80.65 40.32 0.00 0.00	5873.8 5827.3 5462.3 5063.5
3200 1019 3200 120 3200 121 3200 122 3200 123	1 1 1 1 1	1.42 1.42 1.42 1.42 1.42	5793.22 5697.00 5492.31 9053.59 3778.43	80.66 40.32 0.00 0.00 0.00	5873.8 5827.3 5482.3 5063.5 3778.4
38bi h19 38bi h20 38bi h21 38bi h22 38bi h23 38bi h23	1 1 1 1 1 1	1.42 1.42 1.42 1.42 1.42 1.42 1.42	5733.22 5687.00 5482.31 6053.56 3778.43 3342.47	80.65 40.32 0.00 0.00 0.00 0.00	5973.8 5927.3 59422.9 5963.5 3778.4 3342.4 3342.4
328/h19 328/h20 328/h21 328/h22 328/h23 328/h23 328/h23		1.42 1.42 1.42 1.42 1.42 1.42 1.42 1.42	5733.22 5667.00 5442.31 9063.59 3778.43 3342.47 3823.77	80.65 40.32 0.00 0.00 0.00 0.00	5873.8 5827.9 5842.3 5953.5 3778.4 3342.4 3342.4 3342.4
328h119 328h129 328h122 328h122 328h123 328h124 328h124 328h125		1.42 1.42 1.42 1.42 1.42 1.42 1.42 1.42	5783.22 5697.00 5982.31 3083.59 3778.43 3342.47 2006.50	80.65 40.32 0.00 0.00 0.00 0.00 0.00 0.00	9718 95178 9527 3 9515 9515 9515 9778 4 9324 9324 9221 7 2906 5
328/h19 328/h29 328/h22 328/h22 328/h25 328/h25 328/h25 328/h25		1.42 1.42 1.42 1.42 1.42 1.42 1.42 1.42	5783 22 5657 05 5682 31 5585 56 3778 42 3384 47 2828 77 2006 50 2820 75 2806 50	80.65 40.32 0.00 0.00 0.00 0.00 0.00 0.00	5873 8 5873 8 5823 5 5862 3 5863 5 5863 5 5862 3 5862 3 5862 3 5862 3 5862 3 3862 4 2862 5 2862 5 2862 5 2862 5
300/113 300/123 300/123 300/123 300/123 300/124 300/123 300/123		1.42 1.42 1.42 1.42 1.42 1.42 1.42 1.42	5733 22 5597 03 5482 31 3778 45 3382 47 33822 77 2906 50 2820 75 101 57	80,65 40,32 0,00 0,00 0,00 0,00 0,00 0,00 0,00	5873.8 5873.8 5823.3 5863.5 5873.8 5863.5 5873.8 3824.7 2886.5 2886.5 2886.7 1801.5
320m19 20m12 20m12 20m12 20m13 20m13 20m13 20m13 20m13 20m13 20m13 20m13 20m13		1.42 1.42 1.42 1.42 1.42 1.42 1.42 1.42	5783 22 5597 05 54923 19 3774 45 33824 27 2809 50 2800 75 1911 57 1917 74	80.65 40.22 0.00 0.00 0.00 0.00 0.00 0.00 0.0	5873 6 5873 6 5627 3 5642 3 565 5642 3 565 5642 3 5642 3 565 5642 3 565 5642 3 5642 3 5642 3 5642 3 5642 3 5642 3 565 5642 3 565 5642 3 565 565 565 565 565 565 565 565 565 56
320m19 220m20 320m20 320m20 320m20 320m20 320m20 320m20 320m20 320m20 320m20 320m20		1.42 1.42 1.42 1.42 1.42 1.42 1.42 1.42	6730 22 69970 (0) 6942 31 3734 49 3354 49 3354 47 325277 2005 50 280375 101577 1142 78 514 47	80,65 40,22 0,00 0,00 0,00 0,00 0,00 0,00 0,0	6742 6823 6 6823 5 6823 5 6825 5 6825 5 6825 5 6825 5 7878 4 3824 7 2805 5 2820 7 2805 5 2820 7 7897 5 7827 7 7877 7 7877 7 7874 5 784 4
220119 20010 20010 20010 20010 20010 20010 20010 20010 20010 20010		1.42 1.42 1.42 1.42 1.42 1.42 1.42 1.42	5735.22 656710 6482.31 6502.59 3776.40 3922.77 292950 2920575 9001.57 1127.79 574.47	80.65 40.22 9.00 9.00 9.00 9.00 9.00 9.00 9.00	5075 5075 5075 5082 5083 5083 5083 5084 3082 3082 3082 3082 5085 5085 5085 5085 5085 5085 5085 5

-		Timin	g Report - (id: 1)				
Options Endpoint: cout							Gose
Aidines	Endpoint		Siack (ps)	Rise S	iev (ps)	Fall Siew (ps)	
	cost		unconstrain	ed	61.70		56
Pn	Туре	Fanout	Load (F)	Slew (ps)	Delay (ps)	Artival (ps)	
g17/00	ACOHAL	1	5.90	97.60	185.50	41083.50	
hőőic							
h671a							
g17/B					0.00	41083.50	
g17/00	ACOHAL	1	5.90	97.60	185.50	41268.80	
h£Tic							
h58ia							
g17/B					0.00	41268.80	
g17/CO	ACOHOL	1	5.90	97.60	185.50	41454.30	
168ic							
h68/a							
g17/B					0.00	41454.30	
g1700	ACOHAL	1	5.90	97.60	185.50	41639.80	
169/c							
h60/a							
g17/B					0.00	41639.80	
g17/CO	ACOHXL	1	5.90	97.60	185.50	41825.30	
hélic							
n61/a							
g17/B					0.00	41825.30	
g17/00	ACOH(L	1	1.90	65.40	158.30	41983.60	
161/c							
h62'a							
g91A					0.00	41983.60	
g9 Y	ANDEXI	1	0.00	59.00	172.20	42155.80	
1621c							
cout	put port				0.00	42155.80	
			- n	I	Tourse County 1		

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

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

	\land Applications Places System 🚱					0 0 10:45 AM
			Report Power			
	Generated by: Encounter(R) RTL Complex RC13.10 - v13	13.10-s006_1 (Feb 5.2014)				
	Generaled on: Feb 21 2018 10:45:10					
	Module: mul_64bit					
	Technology library: tsmc18 1.0					
	Operating conditions: slow (balanced_tree)					
	Wreisad mode: enclosed					
IJA						
	Instance	Cells	Leokage (1W)	internal (nW)	Net (HW)	Switching (rW)
	instance Trul_\$401	Cels 11445	Leokage (1W) 19264.96	internal (rW) 39702878.79	Net (HW) 8200807.81	Switching (rW) 47903686.6
	instance ral_68it rai_68ith11	Celis 11445 1	Lookage (1W) 19264.96 1.42	internal (114) 39702878.79 129848.81	Net (11V) 8200807.81 9419.37	Switching (1W) 47903686.6 15964.1
	instance Tul_\$40 Tul_\$40h101 Tul_\$40h102	Cels 11445 1	Leskage (1117) 19264-96 1.42 1.42	internal (144) 39702878.79 129441.81 11384.28	Net (1997) 8200807.81 9409.97 1451.62	Switching (1W) 47903686.6 15964.1 12816.9
	Induce mil_5601 mil_6401102 mil_5601102 mil_5601103	Cots 11445 1 1 1	Lookage (111) 19264.96 1.42 1.42 1.42	116anal (144) 39702878.79 17944.81 11384.28 11384.28	Net (1999) 8200607.61 74419.37 1451.62 604.64	Switching (1W) 47900886.6 18944.1 12816.9 12862.9
	Instance mul (444) mul (444)111 mul (444)112 mul (444)112 mul (444)112 mul (444)1124	Colis 11445 1 1 1 1	Lookage (199) 193854.99 1.42 1.42 1.42 1.42	internal (YM) 39702878.79 17944.81 11384.28 11384.28 11384.08	Net (1999) 8200697.81 94193.37 14631.62 604.84 282.26	Switching (rW) 47903686.6 18944.1 12816.9 12862.9 11986.3
	1651x60 mL 64ht mL 64ht2 mL 64	Cells 11445 1 1 1 1 1	Losioge (W) 1988-36 1.0 1.4 1.4 1.4 1.4 1.4 1.4 1.4 1.4 1.4 1.4	144mal (144) 38702878.79 17944.81 11584.28 11758.15 11684.08 11776.07	Net (1999) 8220877.81 9414.93 1451.82 694.84 282.28 120.37	Suticing (1W) 47903866 1894 1 1894 1 1896 1 1786 1 1186 1 1187 8
	ты (Акл. м. (Акл. 11) м. (Акл. 12) м. (Акл. 12) м. (Акл. 13) м. (Акл. 13) м. (Акл. 13) м. (Акл. 13)	Cels 11445	Lossings (111) 19254-99 1.42 1.42 1.42 1.42 1.42 1.42	1166mal (HM) 39702678,75 179044,81 11364,28 11364,28 11368,16 11368,65 11368,65 11368,65	Not (1998) 8200877.81 94143.57 1451.62 644.84 262.26 102.97 41.32	Subching (HV) 4750066,6 1594,1 1896,9 1282,9 1996,9 1977,8 11249,1
	History NJ, 548 NJ, 548 NJ, 548/NJ NJ, 548/N	Cels 1146 1 1 1 1 1 1 1 1 1	Lookup (111) 1999, 4 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0	1164mal (HM) 39702678,75 179044,81 11364,28 113758,15 11684,08 11756,67 11286,85 11286,85 11285,95	Net (1978) 8200877.81 1461.82 654.84 282.26 120.57 40.32 0.00	Sutching (119) 4750366,6 1594,1 13915,9 1396,3 1196,3 1197,7 11246,9 11245,9
	14/460 14/460 14/46010 14/46010 14/46010 14/46010 14/46010	Cels 1146 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	(۲۹۹۵) ۲۶۵۸ (۲۵۰۵) ۱۵۵ (۲۵۰۵) ۱۵۵ (۲۵۰۵) ۱۵۵ (۲۵۰۵) ۱۵۵ (۲۵۰۵) ۱۵۵ (۲۵۰۵) ۱۵۵ (۲۵۰۵)	144mal (HM) 59702878,79 17944,85 117964,85 11796,85 11796,85 11796,85 11726,85 11726,85 11726,85 11726,85 11726,85	741 (1971) 8 200607 28 74 1940 70 74 1940 70	Switching (HW) 47903986 6 18944 5 12862 9 13963 119663 11977,8 112463 9 119663 112463 9 119663
	NEGRO NE (AR NE (ARXIV)	Cels 1146	دهمی (۲۸) ۱۹۵۷ به ۱۹۵۱ به ۱۹۵۹ به ۱۹۹۹ بو ۱۹۹۹ بو ۱۹۹	340md (195) 55702678.79 175641.87 11584.28 11758.15 11758.05 11758.05 11758.05 11758.05 11268	146 (101) 820087 25 9403 27 1451.82 845.84 926.26 102.15 102.15 0.00 0.00 0.00	Suitzing (MV) 4760686 50 1854 51 1286 51 1288 51 1288 51 1288 51 11867 51 11248 51 11248 51 11268 51 1
	Histore NJ, Selt NJ, Selt <td< td=""><td>Cols</td><td>18999(189) 18995 189 182 182 182 182 182 182 182 182 182 182</td><td>74642 (193) 58742878 79 17944 87 17584 58 17584 58 17584 58 17584 58 17524 59 17224 58 17224 58 17224 58 17224 58 17224 58 17224 58 17224 58 17225 58 17255 58 175555 58 17555555 1755555555555555555555555555</td><td>141 (191) 2009 20 2403 20 2403 20 2404 20 2404 20 252 26 20 20 20 20 20 20 20 20 20 20 20 20 20 20 2</td><td>Selding (M) 476066 6 16594 1 18763 1 18773 11985 3 11985 3 11985 4 11985 4 11985 4 11985 4 11985 4</td></td<>	Cols	18999(189) 18995 189 182 182 182 182 182 182 182 182 182 182	74642 (193) 58742878 79 17944 87 17584 58 17584 58 17584 58 17584 58 17524 59 17224 58 17224 58 17224 58 17224 58 17224 58 17224 58 17224 58 17225 58 17255 58 175555 58 17555555 1755555555555555555555555555	141 (191) 2009 20 2403 20 2403 20 2404 20 2404 20 252 26 20 20 20 20 20 20 20 20 20 20 20 20 20 20 2	Selding (M) 476066 6 16594 1 18763 1 18773 11985 3 11985 3 11985 4 11985 4 11985 4 11985 4 11985 4
	14 (464) 14 (464) 14 (464) 14 (464) 15 (464) 15 (464) 15 (464) 16 (464) 17 (464) 17 (464) 17 (464) 18 (464) 19 (464) 10 (46		(10) (10) (10) (10) (10) (10) (10) (10)	919412 (190) SRN22013 79 17644 81 11758 82 11758 82 11758 82 11758 82 11758 82 11758 82 11728 82 11728 11728 82 11728 83	24((rd)) 25(08) 2() 26(08) 2() 46(2) 222 2() 222 2() 2	Selating (197) 4700956: 5 19594 3 19595 3 1955
	National NL (Sel) NL (Sel) <t< td=""><td></td><td>(۲۹) دی در میدود. ۱۹۵۶ - ۲۰۰۶ ۱۹۵۹ - ۲۰۰۶ ۱۹۹۹ - ۲۰۰۶ ۲۹۹۹ - ۲۰۰۶ - ۲۰۰۶ ۲۹۹۹ - ۲۰۰۶ ۲۹۹</td><td>919412 (95) 587422073 79 17944 89 11754 28 11758 25 11758 25 11758 25 11768 25 11768</td><td>146(101) 2003/5</td><td>Selating (KR) 476096 5 15894 1 1286 3 1286 3 1286 3 1186 3</td></t<>		(۲۹) دی در میدود. ۱۹۵۶ - ۲۰۰۶ ۱۹۵۹ - ۲۰۰۶ ۱۹۹۹ - ۲۰۰۶ ۲۹۹۹ - ۲۰۰۶ - ۲۰۰۶ ۲۹۹۹ - ۲۰۰۶ ۲۹۹	919412 (95) 587422073 79 17944 89 11754 28 11758 25 11758 25 11758 25 11768	146(101) 2003/5	Selating (KR) 476096 5 15894 1 1286 3 1286 3 1286 3 1186 3
			189691401 1896 189 182 182 182 182 182 182 182 182 182 182	91930 587122033 75944 81 11984 28 11984 28 11984 28 11984 28 11984 28 11926 85 11926 85 11926 85 11926 85 10968 41 9988 41 9986 47	*xe(rdt) 220897.8 2408.5 446.52 262.8 20.8 20	Settong (M) 476696.6 1954.4 1954.5 1959.5 19
	NU (SAR NJ (SAR		(10) represent 9.1 9.1 9.1 9.1 9.1 9.1 9.1 9.1 9.1 9.1	Printil 1990 SPR02873.2 11954.2 11954.2 11954.2 11954.2 11954.2 11954.2 11954.2 11954.2 11954.2 11954.2 11954.2 11954.2 11954.2 109555.2 100555.2 1005555.2 1005555.2 1005555.2 1005555.2 1005555.2 1005555.2 1005555.2 1005555.2 1005555.2 1005555.2 1005555.2 10055555.2 10055555.2 10055555.2 1005555.2 10055555.2 10055555.2 1005555	241(101) 25083 51 3608 52 46182 2528 46122 46122 4612 4612 4612 4612 4612 46	Settong (M) 4766956 54 19584 3 19584 3 19585 3 19585 3 19585 3 19585 3 19585 4 19585 4 96954 4 96954 3 96954 3 96955 3 969555 3 969555 3 96955555555555555555555555555555555555
	National NL (Sec.10) NL (Sec.10		100 (1996) 1 (1997) 1	910012 (90) 58710283.8 111584.8 111584.8 111584.5 111584.5 111584.5 111584.5 111584.5 111584.5 111584.5 111584.5 111584.5 1126	Ver(rdf) 020087.8 2403.5 1461.8 044.4 022.2 0.0 0.0 0.0 0.0 0.0 0.0 0	Settong (M) 476696.6 1984 3 1984 3 1985 3 1984 3 1985 3 1985 3 1985 4 1986 4 1986 4 1986 4 1986 4 9986 4 9986 4 9986 4 9986 4 9986 4 9986 5 9986 5 9996 5 9996 9996 9996 99966 5 9996 9996

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



		Timing	Report - (ld: 1)				
tions Endpoint: s(255)							Ciose
Arines	Endpoint	S	lack (ps)	Filse S	liew (ps)	Fail Slev (ps)	
ពុខ	δj		unconstrainer		283.30		144
Pa	Туре	Farout	Load (1F)	Siew (ps)	Delay (ps)	Arival (ps)	_
g21/B					0.00	81417.40	
g21/00	ADDHKL	1	5.90	97.60	185.50	81602.90	F
1461							
014							
460							
file							
g211B					0.00	81602.90	
g21/00	ADCHVL	1	5.90	97.70	185.50	81788.40	
1161							
t2it							
g21/B					0.00	81788.40	
g21/00	ADDHKL	1	5.90	97.70	185.50	81973.90	1
12:1							
131:							
g21/B					0.00	81973.90	
g21/00	ADDHKL	1	5.90	97.70	185.50	82159.40	1
131:1							
140							
g218	199100				0.00	82159.40	
g21S	ADDHKL	1	0.00	144.10	160.30	82319.70	
145							
nis[2]							
nelil nelil							
-011							
alan alan							
[64]							

E		6 D - J 14	-1:	17-1:-
Figure 17: Delay re	DOTIVOT 128-DI	i Dadda millin	niter using	venic
ingaronizi zonaj re	010101 120 01	C Ducient mane	prior abiling	, care

\land Applications Places System 😝					05 2581
		Report Po	WEF		
Generaled 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: tamc18 1.0					
Operating conditions: slow (balanced_tree)					
Wreload mode: enclosed					
Instance	Cells	Leakage (rW)	internal (nW)	Net (nM)	Switching (YW)
u_128xit	45101	78152.79	175066822.72	36529528.63	211596
ul_1985itim21	11445	19985 77	40376596 38	8117703.87	49494
ul_128bitim21/h31	1	1.42	12510.12	7029.50	195
u_128bitim211432	1	1.42	11883.53	5483.56	17
ul_128bitim211433	1	1.42	12753.86	4976.79	17
ul_128bitim211534	1	1.42	12151.06	4354.52	16
ul_128bitim211h35	1	1.42	11424.52	4064.83	15
ul_128bitim211h36	1	1.42	11084.06	3815.97	14
u _128bitim211h37	1	1.42	12572.24	4175.49	16
ul_128bitim211h38	1	1.42	10872.11	3600.69	14
u_128bitim211h39	1	1.42	10840.30	3577.11	14
ul_128bitim21140	1	1.42	10839.67	3577.11	14
ul_128bitim21/h41	1	1.42	9971.01	3253.85	13
ul_128bitim211h42	1	1.42	9911.66	3230.27	13
ul_128bitim21/h43	1	1.42	8826.44	2876.59	117
rul_128bitim21/h44	1	1.42	9115.83	2970.90	120
ul 128bitim211h45	1	1.42	9115.83	2970.90	120
		Close	Hép		
🔥 128bitmulti - File Br., 🛝 (Desion Bro	wser 1 🗏 (Console - Sim//ision	i) 🔲 fLook at What's Ne 🕺 (Wave	form 1 - Sim. I root@v/s11:~Desk.	Cadence Encounte	nt Prase

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

IV. PERFORMANCE COMPARISON

A. DELAY SUMMARIZATION USING CADENCE EDA

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				





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				

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.