International Journal of Advances in Engineering and Management (IJAEM)
Volume 3, Issue 9 Sep 2021, pp: 931-937 www.ijaem.net ISSN: 2395-5252

Signed 32-bit Vedic Multiplier Using Urdhva Tiryagbhyam Sutra

Vishal G Sarashetti[1], Dr.Kiran V[2]
[1]*Student, Department of ECE, RV College of Engineering, Bangalore, Karnataka*
[2]*Associate Professor, Department of ECE, RV College of Engineering, Bangalore, Karnataka*

Submitted: 05-09-2021 Revised: 12-09-2021 Accepted: 15-09-2021

ABSTRACT- This design mainly describes the design of 16-bit Vedic multiplier and its performance. Vedic calculations are the olden scheme of Mathematics, which has a procedure of mathematical calculations to compute the multiplication of two 16-bit numbers. In this work Urdhva Tiryagbhyam (vertical and crosswise) Vedic sutra is used for multiplier, design which provides better performance and consumes lesser time for computation. The Urdhva Tiryagbhyam is the finest sutra and universal one among additional sutras and which represents the different multiplication process compared to normal multiplication. In this work, ripple carry adder is used to compute the sum of partially generated products. It reduces the complexity towards the addition of unfinished products. The proposed design is designed and implemented in Verilog HDL. For HDL simulation, modelsim tool is used and for circuit synthesis, Xilinx is used.

Index Terms – Verilog HDL, Urdhva Tiryagbhyam**.**

I. INTRODUCTION

Rapid increase in digital devices the processing of digital data which in the form of text audio video or other form is needed in the much faster way for which the multiplier is used as a basic block, to increase the performance the multiplier delay should be reduced.

The one way to make the faster multiplication, we make use of Vedic multiplier using urdhva triyaghyam sutra. Urdhva Tiryagbhyam (vertical and crosswise) Vedic sutra is used for multiplier design which provides better performance and consumes lesser time for computation. The Urdhva Tiryagbhyam is the finest sutra and universal one among additional sutras and which represents the different multiplication process compared to normal multiplication.

In this work, the multiplier utilizes the Urdhva-Tiryakbhyam sutra for multiplication of binary numbers. The major consideration of the design is to improve the speed of multiplie.

II. LITERATURE SURVEY

Many researchers have developed algorithms for multiplication using Vedic multiplier. Realization of high-speed Vedic multiplier by means of Vedic mathematics sutra was discussed. They used Urdhva Tiryagbhyam sutra for the design of the 8-bit multiplier ripple carry adder is used to add the unfinished products to obtain the resultant product. The result shows multiplier utilizes 1us time to multiply the two 8-bit numbers. Design of area and time delay efficient multiplier to obtain better performance of the multiplier is given. The result shows scheme consumes 44.358 ns to produce the final product of the given two 8-bit input data. In this work, Urdhva Tiryagbhyam Vedic sutra used for multiplication of two 8-bit binary numbers. Ripple carry adder is used to produce the final product by adding unfinished product[1].

A digital computer performs many arithmetic operations which include addition, subtraction, multiplication, division. In these operations multiplication and division are achieved by performing successive addition and subtraction respectively for a specific number of times. Two binary numbers can be multiplied using a circuit called multiplier. It is built using adder circuits. This circuit is also referred as binary multiplier. The speed of a multiplier circuits totally depends on the adders used. Different implementation of adders is used in order to improve the power consumption of the multiplier ripple carry adder is one among them. In this design we have

International Journal of Advances in Engineering and Management (IJAEM)
Volume 3, Issue 9 Sep 2021, pp: 931-937 www.ijaem.net ISSN: 2395-5252

implemented a Vedic multiplier using regular ripple carry adder (RCA)[2].

The ancient Indian mathematics is also known as Vedic mathematics and it is based on sixteen principles or sutras. Vedic mathematics is form of ancient calculation which was discovered by Sri Bharati Krishna Tirthaji Maharaj. There are over sixteen sutras namely Ekadhikina Purvena, Nikhilam Navatashcaramam Dashatah, Urdhva Tiryagbyham, Paaraavartya Yojayet are few of them. Among the sixteen sutras urdhva triyaghyam is the most widely used technique to implement calculations for larger numbers using Vedic mathematics, this technique can also be implemented for digital signal processing applications. Urdhva-tiryagbhyam the name itself suggests that a calculation is performed vertically and crosswise, this technique is used for multiplication and division of large numbers. In this design we have implemented a 4 bit to 16bit Vedic multiplier using RCA[2].

Urdhva Tiryakbhyam This sutra is based on “Vertically and Crosswise” technique. It makes almost all the numeric computations faster and easier. The advantage of multiplier based on this sutra over the others is that with the increase in number of bits, area and delay increase at a smaller rate in comparison to others [3].

III. METHODOLOGY

A binary multiplier can be used in digital electronics as a electronic circuit, such as in computers to find the product of two binary numbers. Carbon-copy of normal multiplication technique is used by binary multiplier, the multiplicand is multiplied with each bit of the multiplier beginning from the least significant bit. Two half adder (HA) modules can be used in order to implement a 2-bit binary multiplier.

In 4 bit Vedic multiplier using urdhva triyaghyam sutra, a multiplier of 2 bit is used to calculate intermediate stage results, and the output is 4 bits.
(A3A2)(B3B2) using 2 bit multiplier generates result: S33S32S31S30
(A3A2)(B1B0) using 2 bit multiplier generates result: S23S22S21S20
(A1A0)(B3B2) using 2 bit multiplier generates result: S13S12S11S10
(A1A0)(B1B0) using 2 bit multiplier generates result: S03S02S01S00.

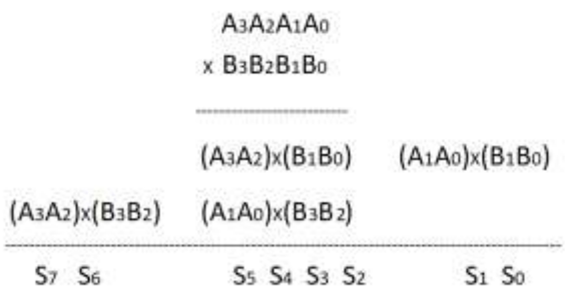

Figure 1. 4x4 Binary Multiplier

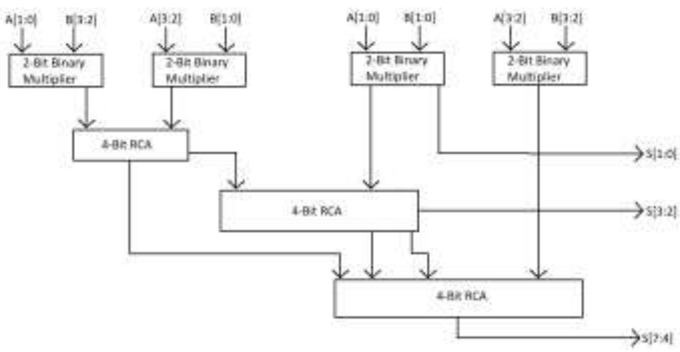

Figure 2. Modified 4 Bit Vedic Multiplier

The above figure represents the block diagram of 4-bit Vedic multiplier, as shown in the fig-2 there are four 2-bit Vedic multiplier and three 4-bit RCA and results will be of 8-bit.

International Journal of Advances in Engineering and Management (IJAEM)
Volume 3, Issue 9 Sep 2021, pp: 931-937 www.ijaem.net ISSN: 2395-5252

Like this we can achieve 32-bit Vedic multiplier using the four 16-bit Vedic multiplier and three 32-bit RCA.

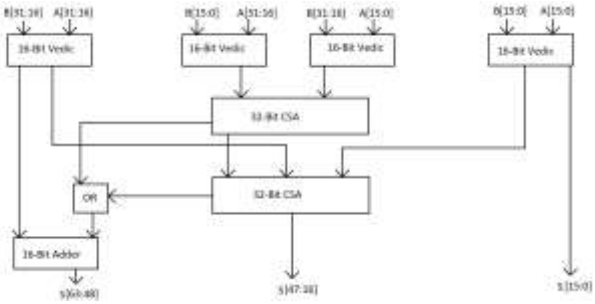

Figure 3. 32 Bit Vedic Multiplier

1) 32bit signed Vedic Multiplier

```
`timescale 1ns / 1ps
module vedic_32bit_mul_signed( p, a, b );

input [31:0] a,b;
output [63:0] p;
wire [31:0] t1,t2,t3,t4;
wire a1=0;

reg signedbit;

vedic_16bit_mul m1( .a(a[15:0]) , .b(b[15:0]) , .p(t1));
vedic_16bit_mul m2( .a({1'b0,a[30:16]}) , .b(b[15:0]) , .p(t2));
vedic_16bit_mul m3( .a(a[15:0]) , .b({1'b0,b[30:16]}) , .p(t3));
vedic_16bit_mul m4( .a({1'b0,a[30:16]}) , .b({1'b0,b[30:16]}) , .p(t4));

assign p[15:0]=t1[15:0];
wire [3:0] cout1;
wire [31:0] temp1,temp2,temp3;

parallel_32bit_adder r1( temp1, cout1[0], {16'b0,t1[31:16]}, t2, 1'b0);
parallel_32bit_adder r2( temp2, cout1[1], temp1, t3, 1'b0);

assign p[31:16] = temp2[15:0];
wire s,c1;

full_adder f1( s ,c1 ,cout1[0] ,cout1[1] , 1'b0);

parallel_32bit_adder r3(temp3,cout1[2],{14'b0,c1,s,temp2[31:16]}, t4 , 1'b0);

assign {a1,p[62:32]} = temp3;
always@(*)
begin
if(a[31]!=b[31])
begin
signedbit=1;
end
else
begin
signedbit=0;
end
end


assign p[63] = signedbit;

endmodule
```

2) 16bit Vedic Multiplier

```
`timescale 1ns / 1ps
module vedic_16bit_mul(p,a,b );

input [15:0] a,b;
output [31:0] p;
wire [15:0] t1,t2,t3,t4;

vedic_8bit_mul m1( .a(a[7:0]) , .b(b[7:0]) , .p(t1));
vedic_8bit_mul m2( .a(a[15:8]) , .b(b[7:0]) , .p(t2));
vedic_8bit_mul m3( .a(a[7:0]) , .b(b[15:8]) , .p(t3));
vedic_8bit_mul m4( .a(a[15:8]) , .b(b[15:8]) , .p(t4));

assign p[7:0]=t1[7:0];
wire [3:0] cout1;
wire [15:0] temp1,temp2,temp3;
```

International Journal of Advances in Engineering and Management (IJAEM)
Volume 3, Issue 9 Sep 2021, pp: 931-937 www.ijaem.net ISSN: 2395-5252

```
parallel_16bit_adder r1( temp1, cout1[0],
{8'b0,t1[15:8]}, t2, 1'b0);
parallel_16bit_adder r2( temp2, cout1[1], temp1,
t3, 1'b0);

assign p[15:8] = temp2[7:0];
wire s,c1;

full_adder f1( s ,c1 ,cout1[0] ,cout1[1] , 1'b0);

parallel_16bit_adder
r3(temp3,cout1[2],{6'b0,c1,s,temp2[15:8]}, t4 ,
1'b0);

assign p[31:16] = temp3;
endmodule
```

3) 8bit Vedic Multiplier

```
`timescale 1ns / 1ps
module vedic_8bit_mul(p , a, b );

    input [7:0] a,b;
    output [15:0] p;
    wire [7:0] t1,t2,t3,t4;

    vedic_4bit_mul m1( .a(a[3:0]) , .b(b[3:0])
, .p(t1));
    vedic_4bit_mul m2( .a(a[7:4]) , .b(b[3:0])
, .p(t2));
    vedic_4bit_mul m3( .a(a[3:0]) , .b(b[7:4])
, .p(t3));
    vedic_4bit_mul m4( .a(a[7:4]) , .b(b[7:4])
, .p(t4));

    assign p[3:0]=t1[3:0];
    wire [3:0] cout1;
    wire [7:0] temp1,temp2,temp3;

    parallel_8bit_adder r1( temp1, cout1[0],
{4'b0,t1[7:4]}, t2, 1'b0);
    parallel_8bit_adder r2( temp2, cout1[1],
temp1, t3, 1'b0);

    assign p[7:4] = temp2[3:0];
    wire s,c1;

    full_adder f1( s ,c1 ,cout1[0] ,cout1[1] ,
1'b0);

    parallel_8bit_adder
r3(temp3,cout1[2],{2'b00,c1,s,temp2[7:4]}, t4 ,
1'b0);

    assign p[15:8] = temp3;
endmodule
```

4) 4bit Vedic Multiplier

```
module vedic_4bit_mul( p, a, b);

    input [3:0] a,b;
    output [7:0] p;
    wire [3:0] t1,t2,t3,t4;

    vedic_2bit_mul m1( .a(a[1:0]) , .b(b[1:0])
, .out(t1));
    vedic_2bit_mul m2( .a(a[3:2]) , .b(b[1:0])
, .out(t2));
    vedic_2bit_mul m3( .a(a[1:0]) , .b(b[3:2])
, .out(t3));
    vedic_2bit_mul m4( .a(a[3:2]) , .b(b[3:2])
, .out(t4));

    assign p[1:0]=t1[1:0];
    wire [3:0] cout1;
    wire [3:0] temp1,temp2,temp3;

    paraller_4bit_adder r1( temp1, cout1[0],
{2'b0,t1[3:2]}, t2, 1'b0);
    paraller_4bit_adder r2( temp2, cout1[1],
temp1, t3, 1'b0);

    assign p[3:2] = temp2[1:0];

    wire s,c1;

    full_adder f1( s ,c1 ,cout1[0] ,cout1[1] ,
1'b0);

    paraller_4bit_adder
r3(temp3,cout1[2],{c1,s,temp2[3:2]}, t4 , 1'b0);

    assign p[7:4] = temp3;
endmodule
```

5) 2bit Vedic Multiplier

```
module vedic_2bit_mul(a , b , out);
    input [1:0] a,b;
    output reg [3:0] out;

    always @( a or b)
    begin
out[0] <= a[0] && b[0] ;
out[1] <= (a[0] && b[1])^( a[1] && b[0]);
out[2] <= (a[1]&& b[1])^((a[0] &&
b[1])&&( a[1] && b[0]));
out[3] <= (a[1]&& b[1])&&((a[0] &&
b[1])&&( a[1] && b[0])) ;
end
endmodule
```

International Journal of Advances in Engineering and Management (IJAEM)
Volume 3, Issue 9 Sep 2021, pp: 931-937 www.ijaem.net ISSN: 2395-5252

6) 32 bit Parallel Adder

```
`timescale 1ns / 1ps
module parallel_32bit_adder(sum , carry ,a , b , cin );
    input cin;
    input [31:0] a,b;
    output [31:0] sum;
    output carry;
    wire temp;

    parallel_16bit_adder p1( sum[15:0], temp , a[15:0] , b[15:0] , cin);
    parallel_16bit_adder p2( sum[31:16], carry , a[31:16] , b[31:16] , temp);

endmodule
```

7) 16 bit Parallel Adder

```
module parallel_16bit_adder(sum, carry, a, b, cin);

    input cin;
    input [15:0] a,b;
    output [15:0] sum;
    output carry;
    wire temp;

    parallel_8bit_adder p1( sum[7:0], temp , a[7:0] , b[7:0] , cin);
    parallel_8bit_adder p2( sum[15:8], carry , a[15:8] , b[15:8] , temp);


endmodule
```

8) 8bit bit Parallel Adder

```
module parallel_8bit_adder(sum, carry, a , b, cin );
    input cin;
    input [7:0] a,b;
    output [7:0] sum;
    output carry;
    wire temp;

    paraller_4bit_adder p1( sum[3:0], temp , a[3:0] , b[3:0] , cin);
    paraller_4bit_adder p2( sum[7:4], carry , a[7:4] , b[7:4] , temp);

endmodule
```

9) 4bit bit Parallel Adder

```
module paraller_4bit_adder(sout , cout , a , b , cin );
    input [3:0] a,b;
    input cin;
    output [3:0] sout;
    output cout;
    wire [2:0]t;

    full_adder f1( sout[0], t[0] , a[0] , b[0] ,cin);
    full_adder f2( sout[1], t[1] , a[1] , b[1] ,t[0]);
    full_adder f3( sout[2], t[2] , a[2] , b[2] ,t[1]);
    full_adder f4( sout[3], cout , a[3] , b[3] ,t[2]);

endmodule
```

10) Full Adder

```
module full_adder(sum, carry , a , b , cin );
    input a,b,cin;
    output sum,carry;
    wire t1,t2;
    wire c1,c2;
    half_adder h1(t1,c1,a,b);
    half_adder h2(sum,c2,t1,cin) ;
    or r1 (carry, c2,c1);

endmodule
```

11) **Half Adder**

```
module half_adder(sum,carry , a , b);

    input a,b;
    output sum,carry;
    xor x1(sum,a,b);
    and a1(carry, a,b);

endmodule
```

International Journal of Advances in Engineering and Management (IJAEM)
Volume 3, Issue 9 Sep 2021, pp: 931-937 www.ijaem.net ISSN: 2395-5252

Figure 4. **32-bit signed multiplication**

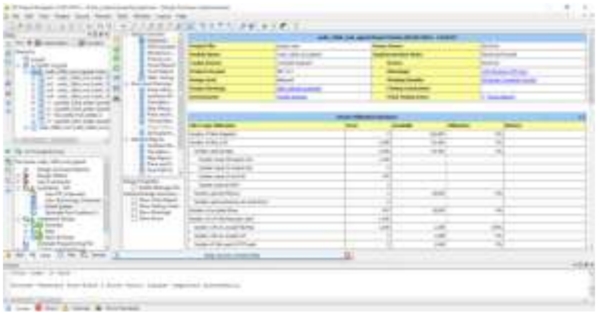

Figure 5. Design summary of 32-bit signed multiplication.

IV. CONCLUSION

The signed 32-bit Vedic Multiplier Vedic Multiplier has been implemented using Urdhva Tiryagbhyam Sutra.

It has observed that the time taken do the multiplication operation of higher bits has been reduced and provides better performance, and consumes lesser power for computation by using the Multiplier which is implemented based on Urdhva Tiryagbhyam Sutra.

REFERENCES

[1]. CHANDRASHEKARA M N, ROHITH S. “Design of 8 Bit Vedic Multiplier Using Urdhva Tiryagbhyam Sutra With Modified Carry Save Adder” 2019 4th International Conference on Recent Trends on Electronics, Information, Communication & Technology (RTEICT-2019), MAY 17th & 18th 2019, Electronic ISBN:978-1-7281-0630-4.

[2]. Y.Harshavardhan, S.Nagaraj, S.Jaahnavi, T.Manasa Reddy. “Analysis of 8-bit Vedic Multiplier using high speed CLA Adder” Proceedings of the Second International Conference on Innovative Mechanisms for Industry Applications (ICIMIA 2020) IEEE Xplore Part Number: CFP20K58-ART; ISBN: 978-1-7281-4167-1.

[3]. Y. Bansal, C. Madhu and P. Kaur, "High speed vedic multiplier designs-A review," 2014 Recent Advances in Engineering and Computational Sciences (RAECS), Chandigarh, India, 2014, pp. 1-6, doi: 10.1109/RAECS.2014.6799502.

[4]. S. Z. H. Naqvi, "Design and simulation of enhanced 64-bit Vedic multiplier," 2017 IEEE Jordan Conference on Applied Electrical Engineering and Computing Technologies (AEECT), Aqaba, Jordan, 2017, pp. 1-4, doi: 10.1109/AEECT.2017.8257751.

[5]. J. Miao and S. Li, "A novel implementation of 4-bit carry look-ahead adder," 2017 International Conference on Electron Devices and Solid-State Circuits (EDSSC), Hsinchu, Taiwan, 2017, pp. 1-2, doi: 10.1109/EDSSC.2017.8126457.

[6]. M. Akila, C. Gowribala and S. M. Shaby, "Implementation of high speed vedic multiplier using modified adder," 2016 International Conference on Communication and Signal Processing (ICCSP), Melmaruvathur, India, 2016, pp. 2244-2248, doi: 10.1109/ICCSP.2016.7754093.

[7]. M. B. Murugesh, S. Nagaraj, J. Jayasree and G. V. K. Reddy, "Modified High Speed 32-bit Vedic Multiplier Design and Implementation," 2020 International Conference on Electronics and Sustainable Communication Systems (ICESC), Coimbatore, India, 2020, pp. 929-932, doi: 10.1109/ICESC48915.2020.9155882.

[8]. G. R. Gokhale and S. R. Gokhale, "Design

of area and delay efficient Vedic multiplier using Carry Select Adder," 2015 International Conference on Information Processing (ICIP), Pune, India, 2015, pp. 295-300, doi: 10.1109/INFOP.2015.7489396.

[9]. G. C. Ram, Y. R. Lakshmanna, D. S. Rani and K. B. Sindhuri, "Area efficient modified vedic multiplier," 2016 International Conference on Circuit, Power and Computing Technologies (ICCPCT), Nagercoil, India, 2016, pp. 1-5, doi: 10.1109/ICCPCT.2016.7530294.

[10]. S. P. Pohokar, R. S. Sisal, K. M. Gaikwad, M. M. Patil and R. Borse, "Design and implementation of 16 × 16 multiplier using Vedic mathematics," 2015 International Conference on Industrial Instrumentation and Control (ICIC), Pune, India, 2015, pp. 1174-1177, doi: 10.1109/IIC.2015.7150925.

[11]. D. K. Kahar and H. Mehta, "High speed vedic multiplier used vedic mathematics," 2017 International Conference on Intelligent Computing and Control Systems (ICICCS), Madurai, India, 2017, pp. 356-359, doi: 10.1109/ICCONS.2017.8250742.