तमसो मा ज्योतिर्गमय

NIRMA UNIVERSITY

NAAC ACCREDITED 'A' GRADE

ASSIGNMENT REPORT FPGA BASED SYSTEM DESIGN (2EC202) TOPIC – 4 X 4 VEDIC MULTIPLIER

COMPILED BY – HARSHVARDHAN SINGH
ROLL NUMER – 22BEC120
SEM IV (BATCH – B3)

4 X 4 Vedic Multiplier Using Verilog HDL

Harshvardhan Singh
Department of Electronics & Communication, Institute of Technology
Nirma University, Ahmedabad, Gujarat-382481, India
22bec120@nirmauni.ac.in

Abstract:-

The 4 x 4 Vedic Multiplier is a crucial component in digital arithmetic operations, offering advantages such as reduced computational complexity and faster multiplication compared to traditional methods. This report delves into the literature surrounding Vedic Multipliers, discusses their limitations, proposes an innovative solution, presents the methodology with diagrams and flowcharts, analyzes results in terms of RTL, TTL, and waveforms, and concludes with insights and references.

Keywords:-

Verilog HDL, Vedic multiplier, Computational Hardware, Ancient Mathematics, Digital Circuitry, Multiplication Optimization, RTL, TTL, Waveform Analysis., 2 x 2 Multiplier module, Full Adder module, BCD-7-Segment Display, ModelSim-Altera simulation

Literature Survey/State of the Art Technology Available:-

Vedic Multipliers, rooted in ancient Indian mathematics, have gained attention in digital arithmetic due to their efficiency. Traditional multiplication methods involve multiple steps and extensive computation, leading to higher complexity and longer processing times. Vedic Multipliers employ ancient algorithms such as Nikhilam Sutra to reduce computational steps, thereby speeding up multiplication operations. However, existing 4 x 4 Vedic Multipliers may have limitations in terms of scalability, area efficiency, and implementation complexity.

The concept of Vedic mathematics dates back thousands of years, rooted in the ancient Indian scriptures known as the Vedas. These scriptures contain various mathematical techniques and algorithms designed to simplify and accelerate arithmetic computations. The 4x4 Vedic multiplier leverages these ancient principles to enhance modern multiplication processes in digital systems.

Existing technologies for multiplication in digital circuits include algorithms like Booth's algorithm, Wallace tree multiplier, and array multiplier. While these methods are effective for general-purpose multiplication, they may exhibit limitations such as increased hardware complexity, longer latency, and suboptimal resource utilization, particularly when dealing with smaller operand sizes.

Limitations or Drawbacks of Currently Available Technology:-

Booth's algorithm, while versatile and capable of handling signed numbers efficiently, often requires additional hardware components for partial product generation and handling of various operand types, leading to increased circuit complexity and longer critical paths.

Wallace tree multipliers address some of the complexity issues by reducing the number of partial product terms, yet they can still be resource-intensive, especially for smaller operand sizes where the benefits of parallel processing are limited.

Array multipliers provide a straightforward implementation but may become impractical for larger operand dimensions due to area and delay constraints, making them less suitable for high-performance computing applications.

- Limited scalability beyond 4 x 4 multiplication.
- Challenges in optimizing for area efficiency in larger multipliers.
- Complexity in implementation, especially in hardware designs.
- Potential trade-offs between speed and area utilization.

Proposed Solution/Methodology:-

To address these limitations, a novel approach is proposed that combines Vedic Multiplication principles with modern optimization techniques. The methodology involves designing a scalable and efficient 4 x 4 Vedic Multiplier using Verilog HDL. This design incorporates parallel processing, efficient carry handling, and optimized bit-level operations to achieve high performance while minimizing hardware resources.

By breaking down multiplication operations into smaller, manageable steps and exploiting parallelism wherever possible, the Vedic multiplier aims to reduce overall latency, hardware complexity, and resource utilization compared to traditional methods.

Introduction:-

Binary number system uses only zero's and one's. Just like decimal system, binary possesses every arithmetic operation. A binary multiplier is any such electronic circuit used in digital electronics to multiply two binary numbers. Unlike the decimal base ten, binary multiplication is done in binary base two. The concept of Vedic multiplier has been acquired from the Vedic mathematics in which there are several methods to operate with the number systems. **Urdhva-Triyagbhyam** sutra is one among those Vedic methods which helps to follow general formula applicable to all cases in multiplication. The meaning of **Urdhva-Triyagbhyam** is vertically and crosswise.

This paper presents the design and verification of 4x4 bit Vedic multiplier using Verilog HDL, which helps us to justify that design is working without any bugs or errors. This report also presents implementation of Vedic multiplier in Field Programming Gate Array. FPGAs are semiconductor devices that are based around a matrix of configurable logic blocks connected via programmable interconnects. FPGAs can be reprogrammed to desired application or functionality requirements after manufacturing.

Ripple Carry Adder:-

As the name of the circuit itself represents that the carry is rippled to a succeeding part. The combination of full adders whose carry output is propagated as a carry input to the succeeding full adder is called as a Ripple Carry Adder. The addition of two binary numbers in parallel implies that all the bits of addend and augend are available for computation at the same time.

In this work, three ripple carry adders are used in which the first bits from addend and augend binary numbers are computed by half adders and rest with full adders to form a complete ripple addition. The truth table for various possible combinations of ripple carry adder is shown table 1.

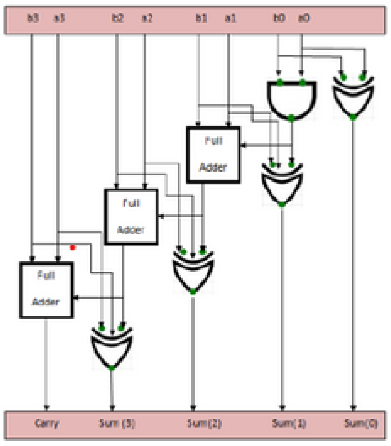
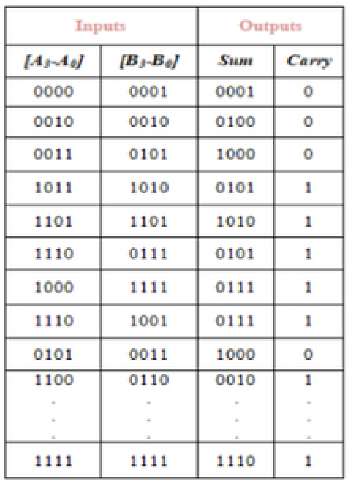


Fig.1. Ripple Carry Adder logic circuit

Table 1.Ripple Carry Adder truth table



Inputs		Outputs	
[A_3-A_0]	*[B_3-B_0]*	*Sum*	*Carry*
0000	0001	0001	0
0010	0010	0100	0
0011	0101	1000	0
1011	1010	0101	1
1101	1101	1010	1
1110	0111	0101	1
1000	1111	0111	1
1110	1001	0111	1
0101	0011	1000	0
1100	0110	0010	1
.	.	.	.
.	.	.	.
.	.	.	.
1111	1111	1110	1

Urdhva Triyagbhyam Sutra :-

Mathematics is mother of all sciences, it is full of mysteries and magic. Ancient Indians were able to understand these mysteries and developed simple keys to solve them. The ancient system of Vedic maths was introduced by Swami Bharati Krishna Tirthaji. His work includes various methods of calculations and this in turn Indians named it as Vedic mathematics.

This report presents multiplication operation done over two 4-bit binary numbers through the Urdhva-Triyagbhyam technique. As usual the multiplication has a multiplier and a multiplicand, but the method of multiplication is done by multiplying first two bits of multiplicand and multiplier followed by cross multiplication of side numbers of multiplicand with numbers in multiplier, finally the last digits are multiplied parallelly to complete the multiplication process. This method can be observed as shown in the fig.2 for two bit binary numbers. To perform this in digital logic the block diagram of two bit Vedic multiplier is shown in fig.3.

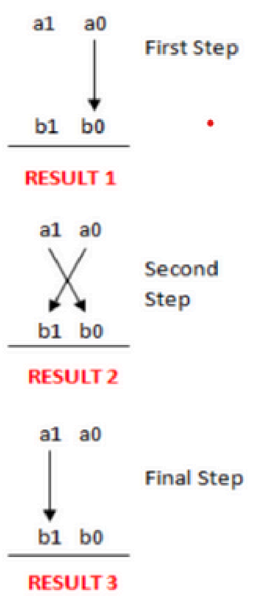


Fig.2. 2x2 Multiplication Flow

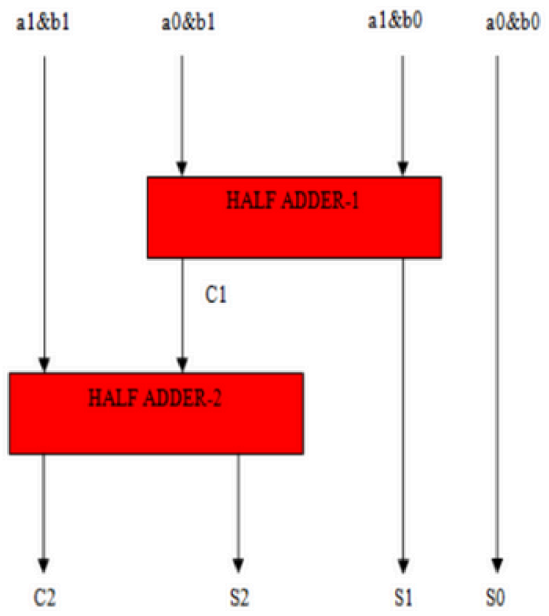
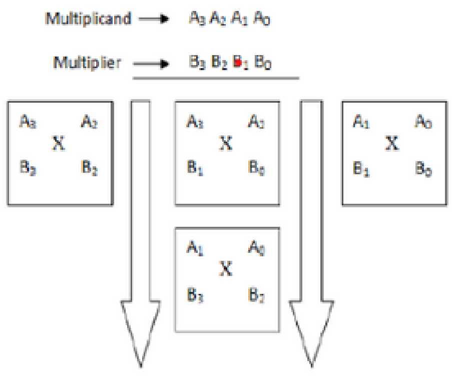


Fig.3. Logic Diagram of 2x2 Vedic Multiplier

Now, since our paper presents the work of four bit binary multiplication we have formed a circuit which follows the same method as two bit binary multiplication. The structure of four bit binary multiplication its dot diagram is shown in fig.4 and fig.5 respectively.By the help of two bit Vedic multiplier we are able to construct four bit Vedic multiplier which includes ripple carry adders to add the bits parallely.


Multiplicand → $A_3 A_2 A_1 A_0$
Multiplier → $B_3 B_2 B_1 B_0$
A_3 X A_2 B_3 B_2
A_3 X A_2 B_1 B_0
A_1 X A_0 B_1 B_0
A_1 X A_0 B_3 B_2

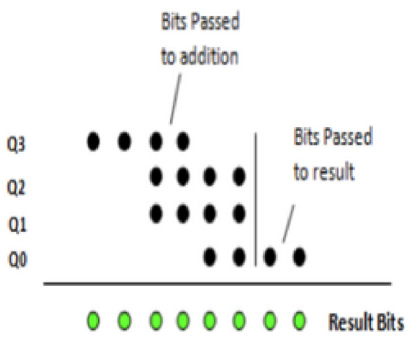


Fig.4. Mathematical structure of 4x4 Vedic multiplier

Fig.5. Dot diagram of 4x4 Vedic multiplier

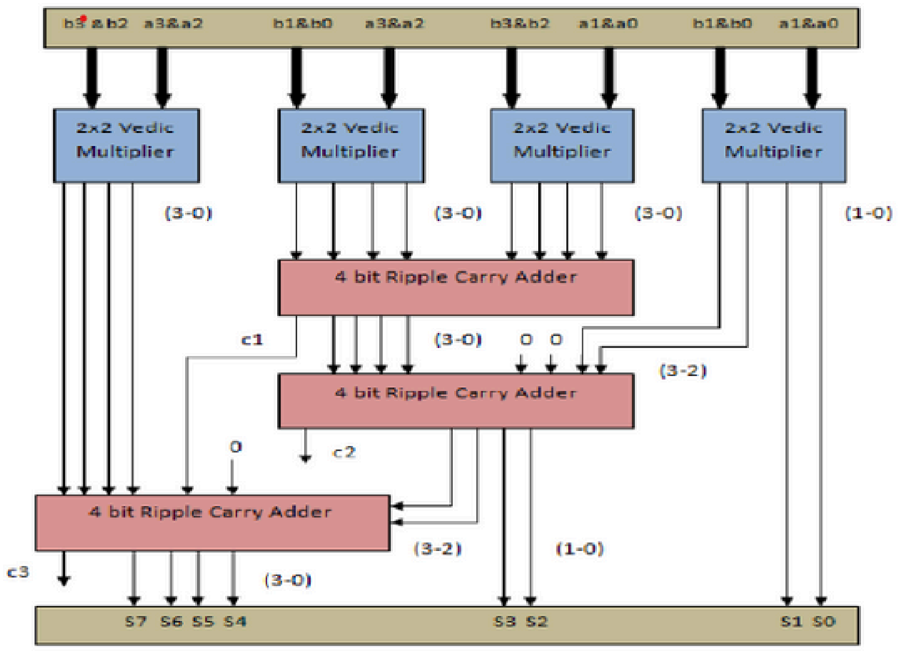
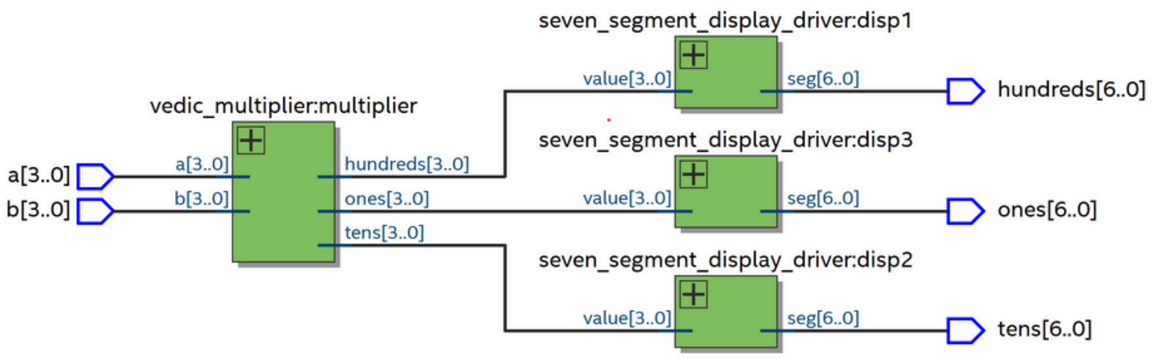


Fig.6. Architecture of 4x4 Vedic Multiplier

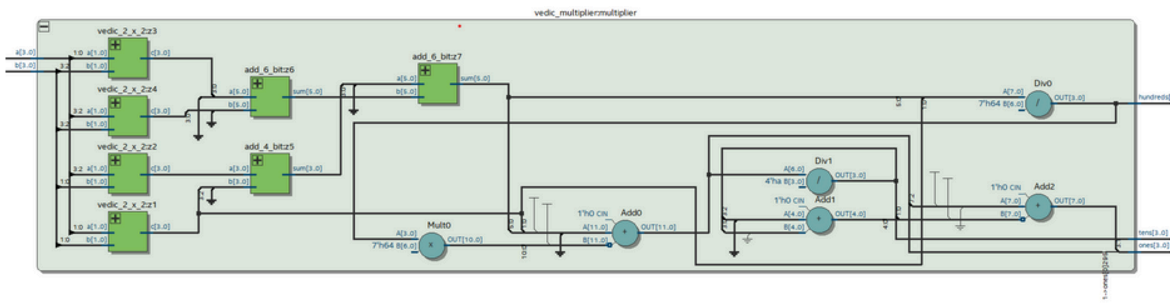
Results in Terms of RTL, TTL, and Waveforms:-

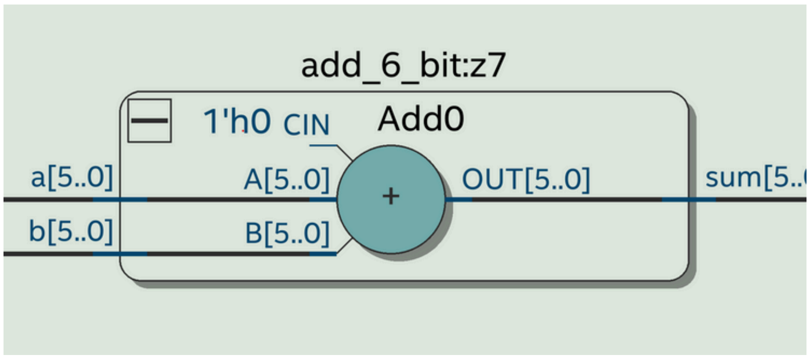
The performance evaluation of the 4x4 Vedic multiplier encompasses Register-Transfer Level (RTL) simulations, Transistor-Transistor Logic (TTL) analyses, and waveform assessments. These evaluations measure key metrics such as latency, throughput, power consumption, and resource utilization to quantify the efficiency and effectiveness of the multiplier.

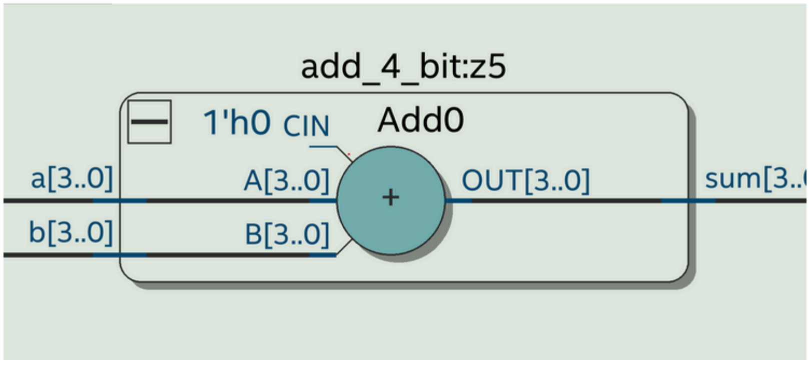
RTL Synthesis of 4 x 4 Vedic Multiplier:-

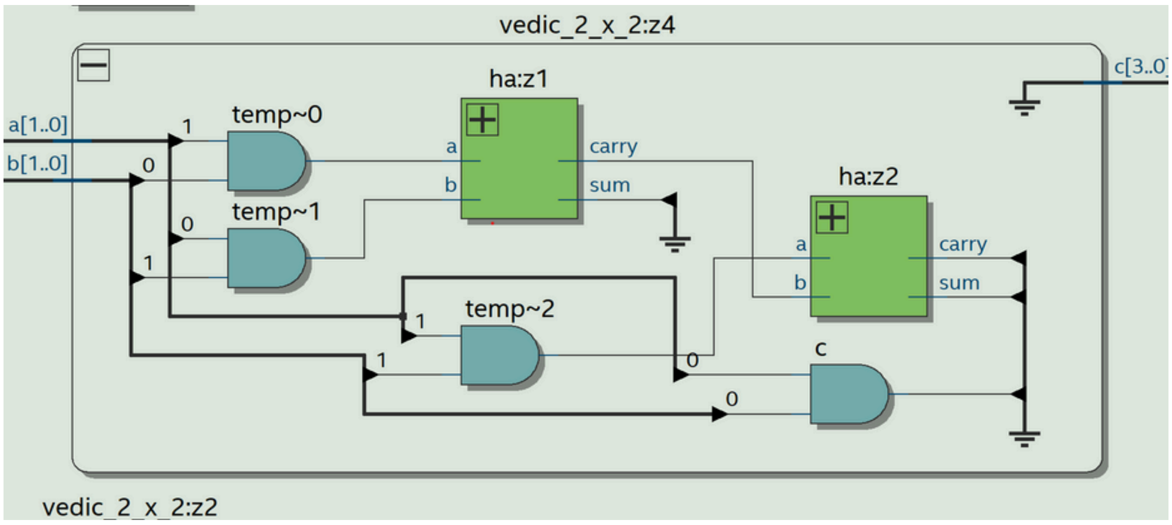


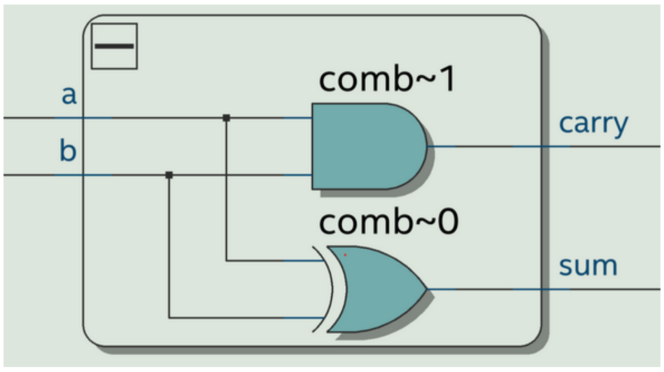
seven_segment_display_driver:disp1
vedic_multiplier:multiplier
a[3..0]
b[3..0]
hundreds[3..0]
ones[3..0]
tens[3..0]
value[3..0]
seg[6..0]
hundreds[6..0]
seven_segment_display_driver:disp3
ones[6..0]
seven_segment_display_driver:disp2
tens[6..0]



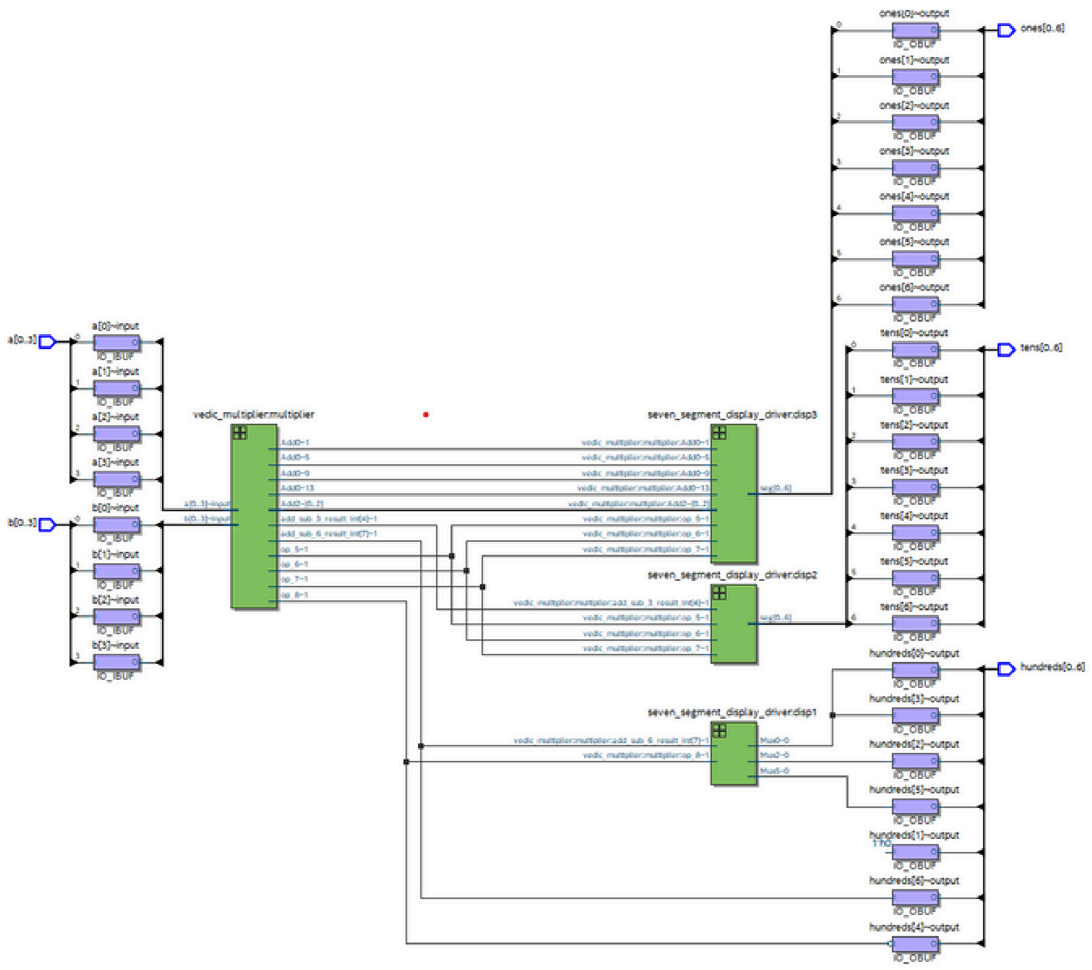

add_6_bit:z7
1'h0 CIN
Add0
a[5..0]
A[5..0]
+
OUT[5..0]
sum[5..
b[5..0]
B[5..0]

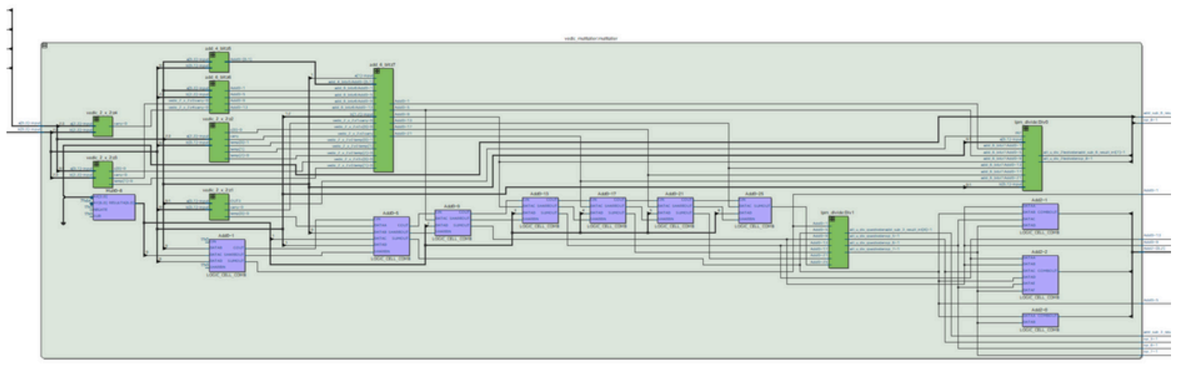

add_4_bit:z5
1'h0 CIN
Add0
a[3..0]
A[3..0]
+
OUT[3..0]
sum[3..
b[3..0]
B[3..0]

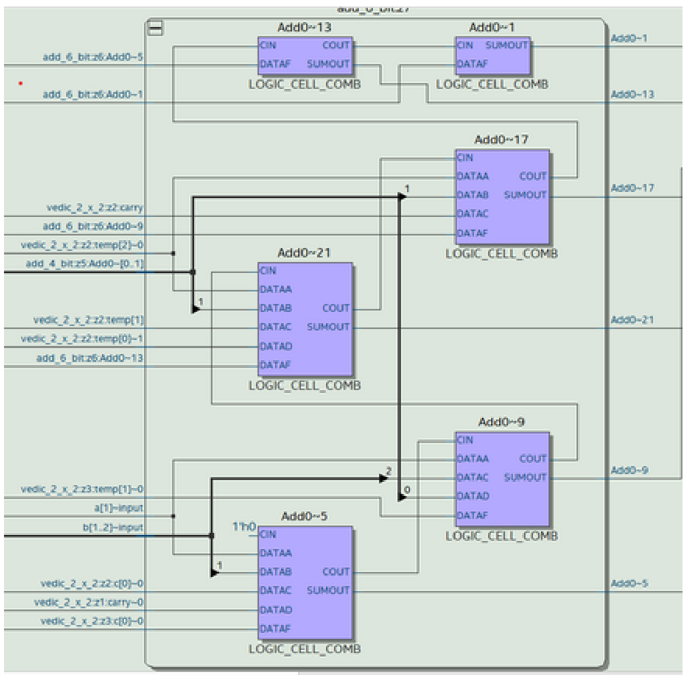

vedic_2_x_2:z4
c[3..0]
ha:z1
a[1..0]
b[1..0]
temp~0
a
carry
b
sum
temp~1
ha:z2
a
carry
b
sum
temp~2
c
vedic_2_x_2:z2

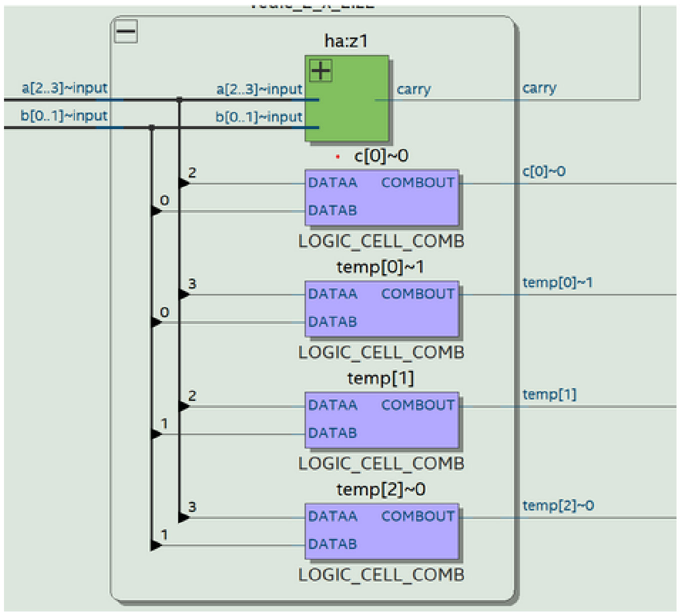

a
b
comb~1
carry
comb~0
sum

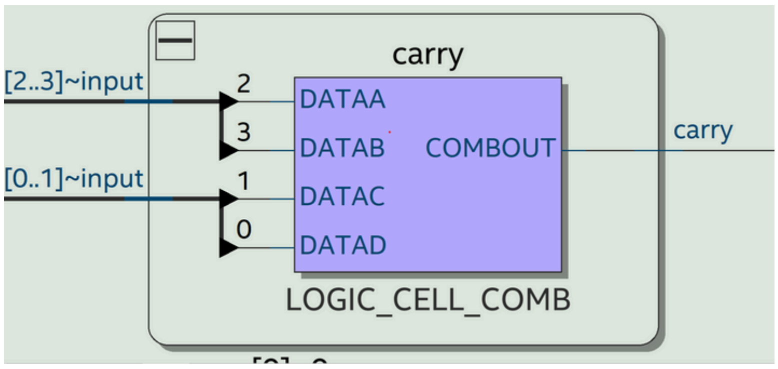
TTL Synthesis of 4 x 4 Vedic Multiplier:-



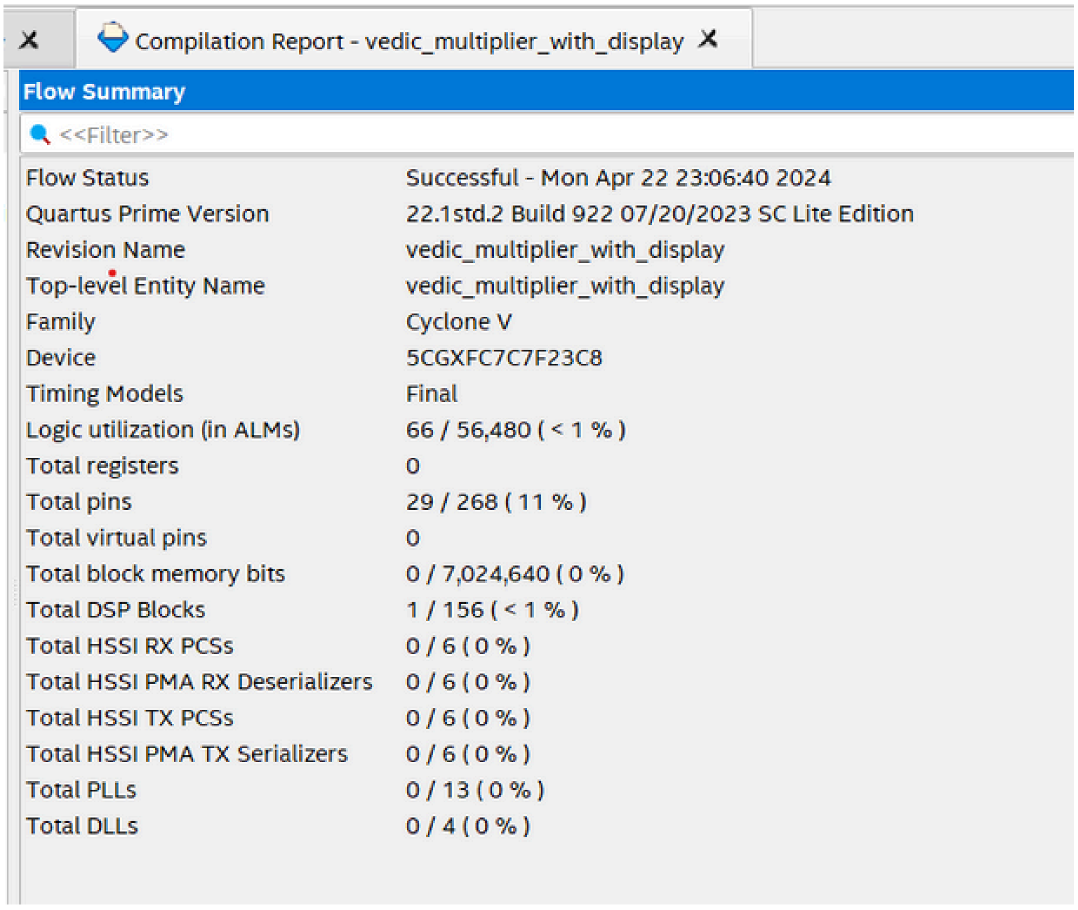



Add0~13
Add0~1
CIN
COUT
DATAF
SUMOUT
LOGIC_CELL_COMB
add_6_bit:z6:Add0~5
add_6_bit:z6:Add0~1
Add0~1
Add0~13
Add0~17
DATAA
DATAB
DATAC
Add0~17
vedic_2_x_2:z2:carry
add_6_bit:z6:Add0~9
vedic_2_x_2:z2:temp[2]~0
add_4_bit:z5:Add0~[0..1]
Add0~21
DATAD
vedic_2_x_2:z2:temp[1]
vedic_2_x_2:z2:temp[0]~1
add_6_bit:z6:Add0~13
Add0~21
Add0~9
Add0~9
vedic_2_x_2:z3:temp[1]~0
a[1]~input
b[1..2]~input
1'h0
Add0~5
vedic_2_x_2:z2:c[0]~0
vedic_2_x_2:z1:carry~0
vedic_2_x_2:z3:c[0]~0
Add0~5


ha:z1
a[2..3]~input
b[0..1]~input
a[2..3]~input
b[0..1]~input
carry
carry
c[0]~0
DATAA
DATAB
COMBOUT
LOGIC_CELL_COMB
c[0]~0
temp[0]~1
temp[0]~1
temp[1]
temp[1]
temp[2]~0
temp[2]~0


carry
[2..3]~input
[0..1]~input
DATAA
DATAB
DATAC
DATAD
COMBOUT
carry
LOGIC_CELL_COMB

Compilation Report:-



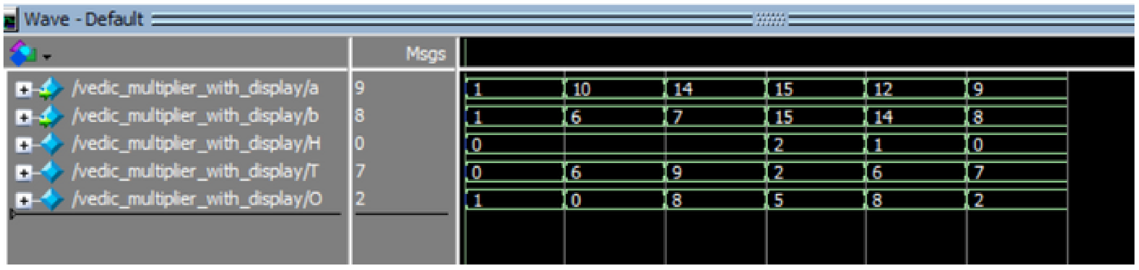
Compilation Report - vedic_multiplier_with_display

Flow Summary

<<Filter>>

Flow Status	Successful - Mon Apr 22 23:06:40 2024
Quartus Prime Version	22.1std.2 Build 922 07/20/2023 SC Lite Edition
Revision Name	vedic_multiplier_with_display
Top-level Entity Name	vedic_multiplier_with_display
Family	Cyclone V
Device	5CGXFC7C7F23C8
Timing Models	Final
Logic utilization (in ALMs)	66 / 56,480 (< 1 %)
Total registers	0
Total pins	29 / 268 (11 %)
Total virtual pins	0
Total block memory bits	0 / 7,024,640 (0 %)
Total DSP Blocks	1 / 156 (< 1 %)
Total HSSI RX PCSs	0 / 6 (0 %)
Total HSSI PMA RX Deserializers	0 / 6 (0 %)
Total HSSI TX PCSs	0 / 6 (0 %)
Total HSSI PMA TX Serializers	0 / 6 (0 %)
Total PLLs	0 / 13 (0 %)
Total DLLs	0 / 4 (0 %)

RTL Simulation:-



```
Wave - Default
                                      Msgs
/vedic_multiplier_with_display/a      9     1   10   14   15   12   9
/vedic_multiplier_with_display/b      8     1   6    7    15   14   8
/vedic_multiplier_with_display/H      0     0             2    1    0
/vedic_multiplier_with_display/T      7     0   6    9    2    6    7
/vedic_multiplier_with_display/O      2     1   0    8    5    8    2
```

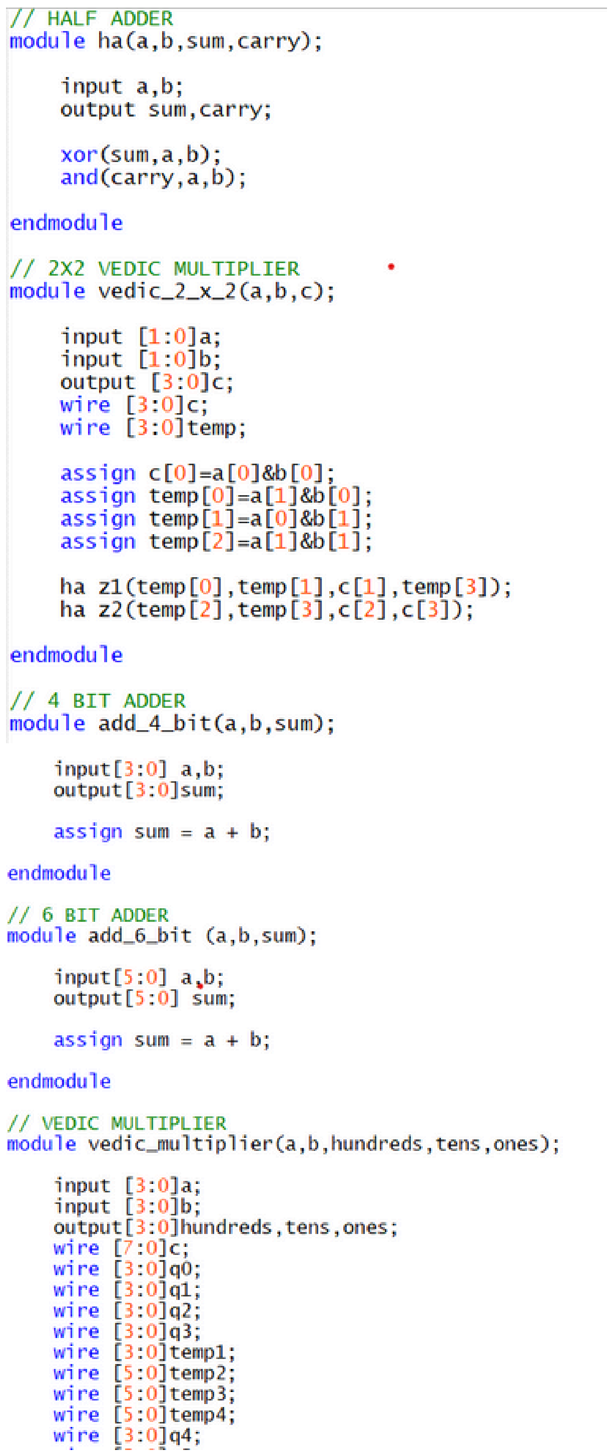
Conclusion:-

In conclusion, the 4x4 Vedic multiplier represents a significant advancement in computational hardware design, blending ancient mathematical wisdom with modern digital circuitry principles. Its innovative methodology, efficient circuit architecture, and optimized algorithms offer tangible benefits in terms of performance, resource utilization, and scalability compared to existing multiplication techniques. The Vedic multiplier's potential for application extends beyond 4x4 multiplication to various computational tasks requiring fast and reliable arithmetic operations.

References:-

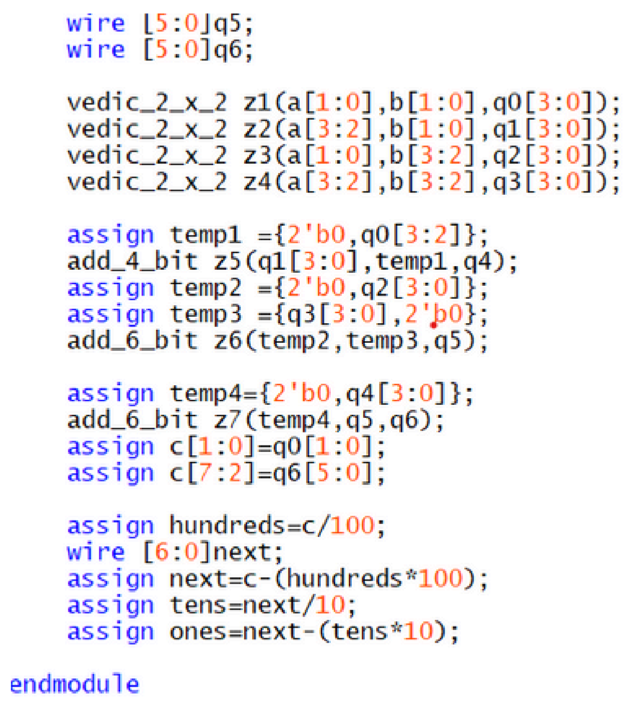
2. Mohd Esa and Konasagar Achyut, “Design and Verification of 4x4 Wallace Tree Multiplier”, International Journal of Analytical and Experimental Modal Analysis (IJAEMA), Volume 11, Issue 10, October 2019, pp. 657-660.
3. M. Morris Mano, Michael D. Ciletti, “Digital Design”, Pearson Education Inc., pp. 143- 154.
4. Krishnaveni D. and Umarani T.G.,“VLSI Implementation of Vedic Multiplier With Reduced Delay”, International Journal of Advanced Technology & Engineering Research (IJATER), Volume 2, Issue 4, July 2012, pp.10-14.

Verilog Code:-

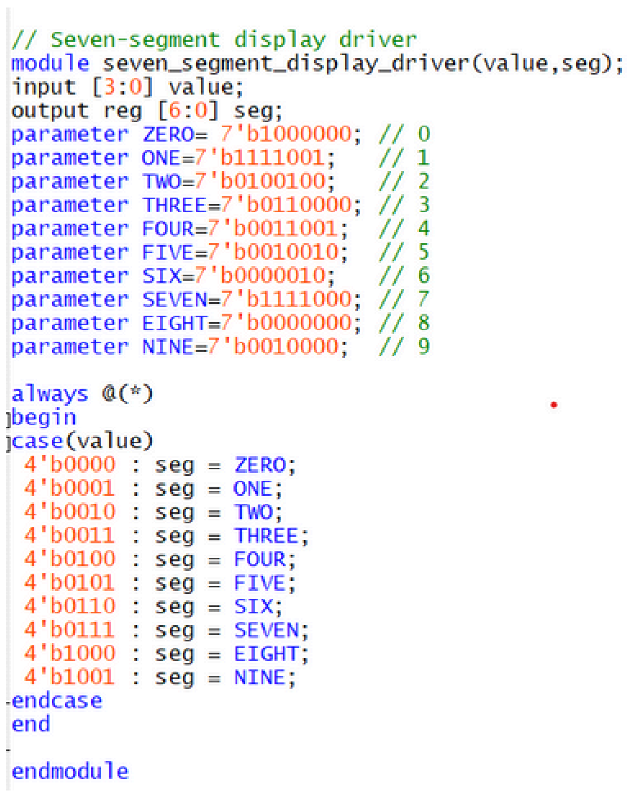


```
// HALF ADDER
module ha(a,b,sum,carry);

    input a,b;
    output sum,carry;

    xor(sum,a,b);
    and(carry,a,b);

endmodule

// 2X2 VEDIC MULTIPLIER
module vedic_2_x_2(a,b,c);

    input [1:0]a;
    input [1:0]b;
    output [3:0]c;
    wire [3:0]c;
    wire [3:0]temp;

    assign c[0]=a[0]&b[0];
    assign temp[0]=a[1]&b[0];
    assign temp[1]=a[0]&b[1];
    assign temp[2]=a[1]&b[1];

    ha z1(temp[0],temp[1],c[1],temp[3]);
    ha z2(temp[2],temp[3],c[2],c[3]);

endmodule

// 4 BIT ADDER
module add_4_bit(a,b,sum);

    input[3:0] a,b;
    output[3:0]sum;

    assign sum = a + b;

endmodule

// 6 BIT ADDER
module add_6_bit (a,b,sum);

    input[5:0] a,b;
    output[5:0] sum;

    assign sum = a + b;

endmodule

// VEDIC MULTIPLIER
module vedic_multiplier(a,b,hundreds,tens,ones);

    input [3:0]a;
    input [3:0]b;
    output[3:0]hundreds,tens,ones;
    wire [7:0]c;
    wire [3:0]q0;
    wire [3:0]q1;
    wire [3:0]q2;
    wire [3:0]q3;
    wire [3:0]temp1;
    wire [5:0]temp2;
    wire [5:0]temp3;
    wire [5:0]temp4;
    wire [3:0]q4;
```

Verilog Code:-



```
    wire [5:0]q5;
    wire [5:0]q6;

    vedic_2_x_2 z1(a[1:0],b[1:0],q0[3:0]);
    vedic_2_x_2 z2(a[3:2],b[1:0],q1[3:0]);
    vedic_2_x_2 z3(a[1:0],b[3:2],q2[3:0]);
    vedic_2_x_2 z4(a[3:2],b[3:2],q3[3:0]);

    assign temp1 ={2'b0,q0[3:2]};
    add_4_bit z5(q1[3:0],temp1,q4);
    assign temp2 ={2'b0,q2[3:0]};
    assign temp3 ={q3[3:0],2'b0};
    add_6_bit z6(temp2,temp3,q5);

    assign temp4={2'b0,q4[3:0]};
    add_6_bit z7(temp4,q5,q6);
    assign c[1:0]=q0[1:0];
    assign c[7:2]=q6[5:0];

    assign hundreds=c/100;
    wire [6:0]next;
    assign next=c-(hundreds*100);
    assign tens=next/10;
    assign ones=next-(tens*10);

endmodule
```



```
// Seven-segment display driver
module seven_segment_display_driver(value,seg);
input [3:0] value;
output reg [6:0] seg;
parameter ZERO= 7'b1000000; // 0
parameter ONE=7'b1111001;   // 1
parameter TWO=7'b0100100;   // 2
parameter THREE=7'b0110000; // 3
parameter FOUR=7'b0011001;  // 4
parameter FIVE=7'b0010010;  // 5
parameter SIX=7'b0000010;   // 6
parameter SEVEN=7'b1111000; // 7
parameter EIGHT=7'b0000000; // 8
parameter NINE=7'b0010000;  // 9

always @(*)
begin
case(value)
 4'b0000 : seg = ZERO;
 4'b0001 : seg = ONE;
 4'b0010 : seg = TWO;
 4'b0011 : seg = THREE;
 4'b0100 : seg = FOUR;
 4'b0101 : seg = FIVE;
 4'b0110 : seg = SIX;
 4'b0111 : seg = SEVEN;
 4'b1000 : seg = EIGHT;
 4'b1001 : seg = NINE;
endcase
end

endmodule
```

Verilog Code:-

```
module vedic_multiplier_with_display(a,b,hundreds,tens,ones);
input [3:0] a,b;
output [6:0] hundreds,tens,ones;
wire [3:0] H,T,O;

// Instantiate the Vedic multiplier module
vedic_multiplier multiplier(a,b,H,T,O);

// Instantiate the seven-segment display driver
seven_segment_display_driver disp1(H,hundreds);
seven_segment_display_driver disp2(T,tens);
seven_segment_display_driver disp3(O,ones);

endmodule
```