PROJECT-1

Title of Project: 8-Bit Vedic Multiplier

Submitted By:

Harshit Sharma (12213044)

Apoorv Mittal (12206778)

Shivam (12218800)

Submitted To

Dr . Abhishek Kumar

8-Bit Vedic Multiplier

1. Abstract

This project involves the design and simulation of an 8-bit Vedic multiplier using the "Urdhva Tiryagbhyam" sutra from Vedic mathematics. Implemented in Cadence Virtuoso, the design utilizes transistor-level modeling to achieve high efficiency in terms of speed and area. The results validate the superiority of the Vedic approach in arithmetic operations, making it suitable for modern digital applications.

2. Introduction

Multipliers are critical components in digital processing systems. Vedic mathematics offers an innovative approach to multiplication that reduces computational complexity. The "Urdhva Tiryagbhyam" method, in particular, enables parallel and hierarchical computation, significantly improving performance. This project aims to design an 8-bit multiplier in Cadence Virtuoso at the transistor level, focusing on optimizing speed and resource usage.

3. Literature Review

Traditional multiplication techniques, such as the array or Wallace tree multipliers, often face challenges in speed and area optimization. Vedic multipliers, however, leverage ancient mathematical principles to provide faster computation with less hardware complexity. Previous works demonstrate their effectiveness in reducing delay and power consumption, making them an excellent choice for modern systems.

4. Methodology

Vedic Sutra

The "Urdhva Tiryagbhyam" sutra performs multiplication by computing partial products in parallel.

Design Process

1. **Transistor-Level Design**:
    - Designed fundamental blocks (AND gates, adders) in Cadence Virtuoso.
    - Hierarchical integration of 2x2, 4x4, and 8x8 multipliers.
2. **Simulation**:
    - DC and transient analysis for functionality.
    - Optimized circuit layout for minimal area and reduced parasitics.

Implementation in Cadence Virtuoso

- Schematic design of each module.
- Layout design to ensure DRC and LVS compliance.

5. Design and Simulation

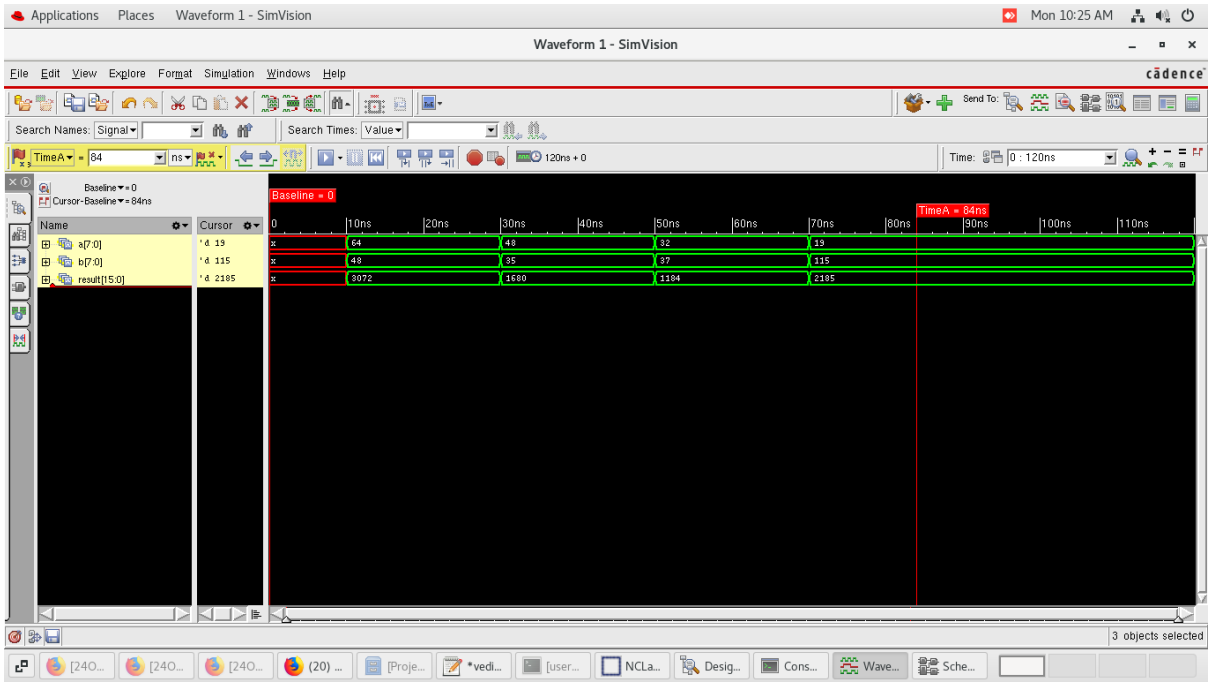
```
input [5:0] a,b;
output [5:0] sum;
wire [5:0] sum;

assign sum = a + b;

endmodule

module adder8(a,b,sum);

input [7:0] a,b;
output [7:0] sum;
wire [7:0] sum;

assign sum = a + b;

endmodule

module adder10(a,b,sum);

input [9:0] a,b;
output [9:0] sum;
wire [9:0] sum;

assign sum = a + b;

endmodule

module vedic8x8tb;
reg [7:0] a,b;
wire [15:0] result;
vedic8x8 V0(a, b, result);
initial begin
$monitor ($time,"<-Time, Productin", result );
```

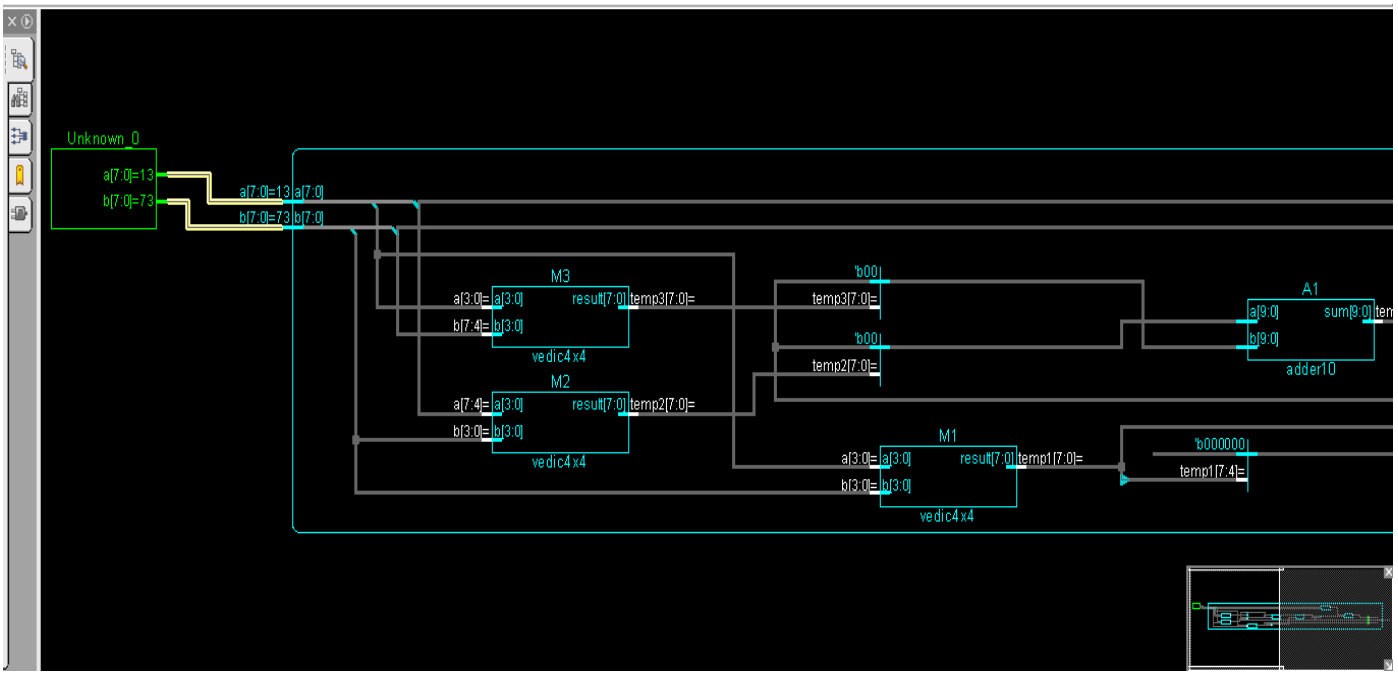
```
    wire [3:0] w;


    assign result[0]= a[0]&b[0];
    assign w[0]     = a[1]&b[0];
    assign w[1]     = a[0]&b[1];
    assign w[2]     = a[1]&b[1];

    halfAdder H0(w[0], w[1], result[1], w[3]);
    halfAdder H1(w[2], w[3], result[2], result[3]);

endmodule

module halfAdder(a,b,sum,carry);
    input a,b;
    output sum, carry;

assign sum   = a ^ b;
assign carry = a & b;

endmodule

module adder4(a,b,sum);

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

assign sum = a + b;

endmodule

module adder6(a,b,sum);
```

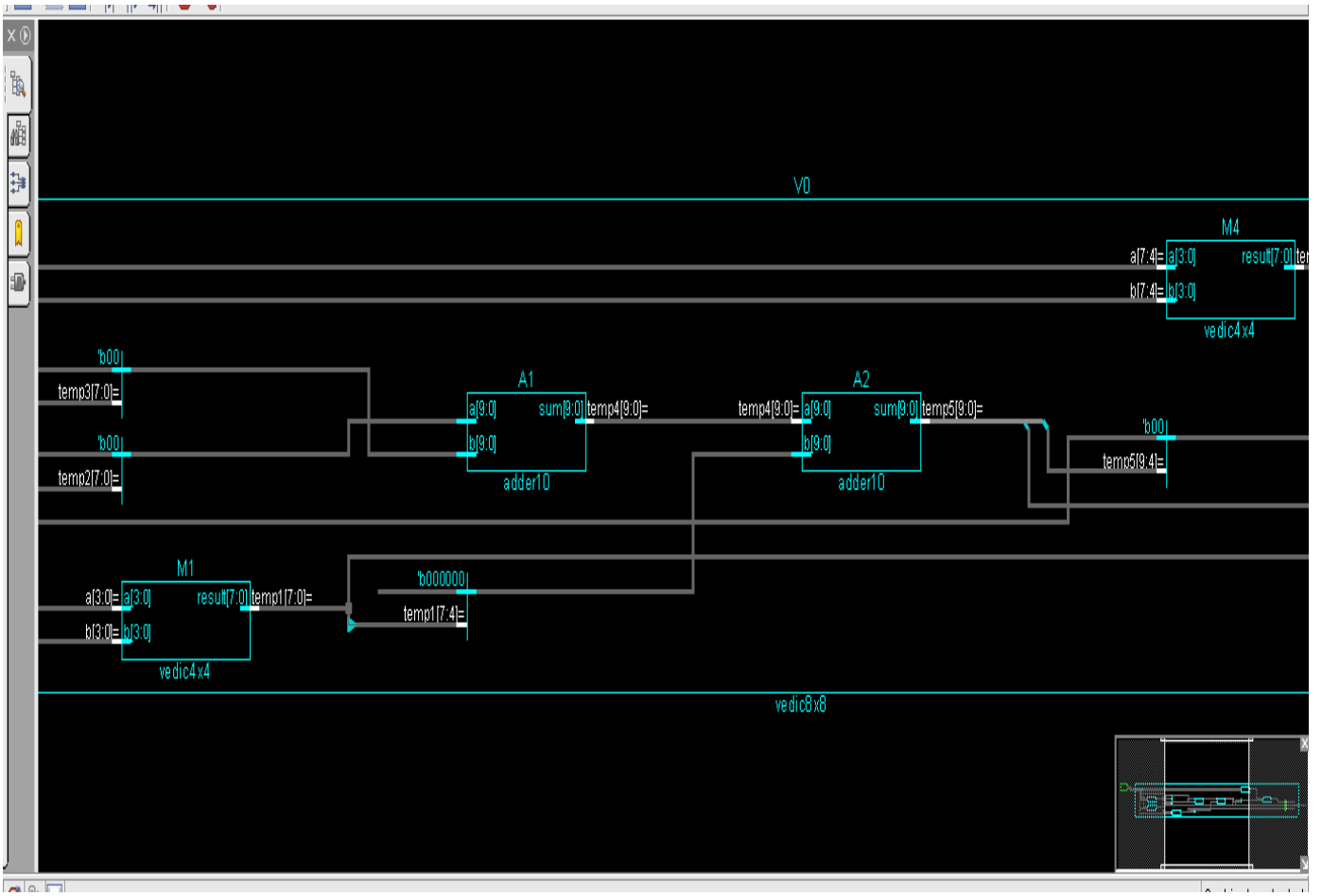
```
wire [3:0] temp1;
wire [3:0] temp2;
wire [3:0] temp3;
wire [5:0] temp4;
wire [5:0] temp5;
wire [3:0] temp6;
wire [3:0] temp7;
wire [5:0] w1;

vedic_2x2 V1(a[1:0], b[1:0], temp1);
assign result[1:0] = temp1[1:0];

vedic_2x2 V2(a[3:2], b[1:0], temp2);
vedic_2x2 V3(a[1:0], b[3:2], temp3);

assign w1 = {4'b0000, temp1[3:2]};

adder6 A1({2'b00, temp3}, {2'b00, temp2}, temp4);
adder6 A2(temp4, w1, temp5);

assign result[3:2] = temp5[1:0];

vedic_2x2 V4(a[3:2], b[3:2], temp6);

adder4 A3(temp6, temp5[5:2], temp7);
assign result[7:4] = temp7;


endmodule

module vedic_2x2 (a, b, result);
    input [1:0] a,b;
    output [3:0] result;
```

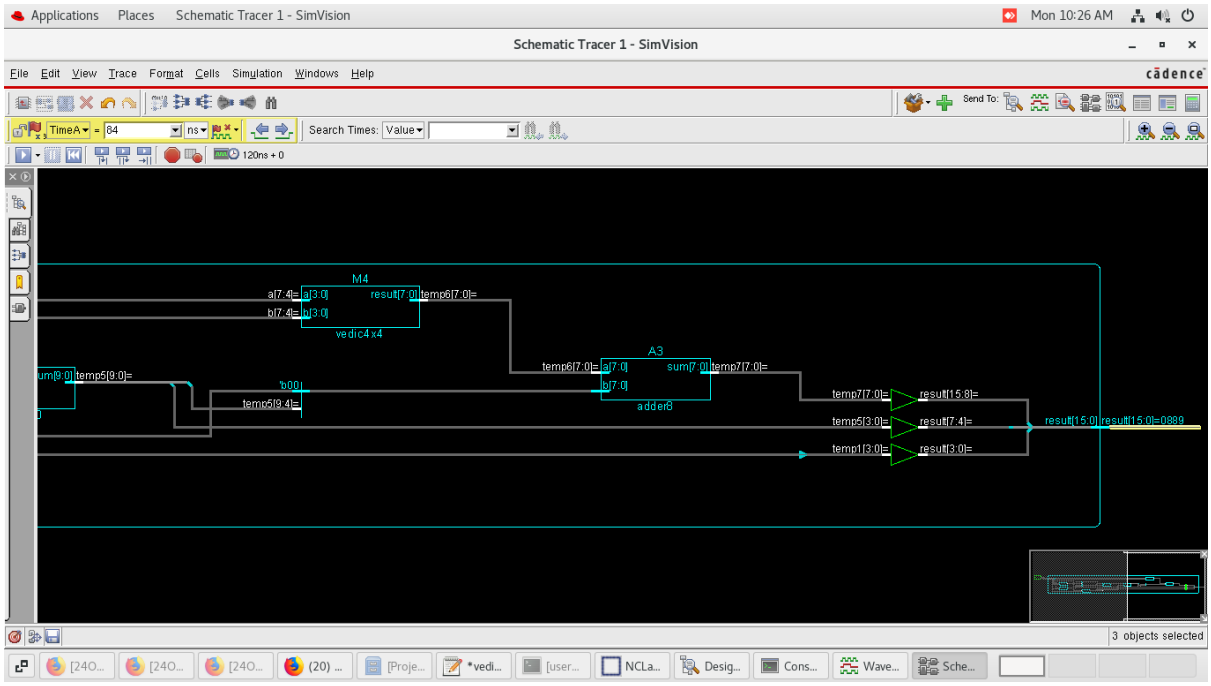
```
module vedic8x8(a, b, result);

    input  [7:0] a,b;
    output [15:0] result;
    wire [15:0] result;

wire [7:0] temp1;
wire [7:0] temp2;
wire [7:0] temp3;
wire [9:0] temp4;
wire [9:0] temp5;
wire [7:0] temp6;
wire [7:0] temp7;

vedic4x4 M1(a[3:0], b[3:0], temp1);
assign result[3:0] = temp1[3:0];

vedic4x4 M2(a[7:4], b[3:0], temp2);
vedic4x4 M3(a[3:0], b[7:4], temp3);

adder10 A1({2'b00, temp2}, {2'b00,temp3}, temp4);
adder10 A2(temp4, {6'b000000, temp1[7:4]}, temp5);
assign result[7:4] = temp5[3:0];

vedic4x4 M4(a[7:4], b[7:4], temp6);
adder8 A3(temp6, {2'b00,temp5[9:4]}, temp7);

assign result[15:8] = temp7;
endmodule

module vedic4x4(a, b, result);
    input  [3:0] a,b;
    output [7:0] result;
    wire [7:0] result;
```

```
input [7:0] a,b;
output [7:0] sum;
wire [7:0] sum;

assign sum = a + b;

endmodule

module adder10(a,b,sum);

input [9:0] a,b;
output [9:0] sum;
wire [9:0] sum;

assign sum = a + b;

endmodule

module vedic8x8tb;
reg [7:0] a,b;
wire [15:0] result;
vedic8x8 V0(a, b, result);
initial begin
$monitor ($time,"<-Time, Productin", result );
#10 a= 8'b01000000; b= 8'b00110000;
#20 a = 8'b00110000; b= 8'b00100011;
#20 a= 8'b00100000; b= 8'b00100101;
#20 a = 8'b00010011; b= 8'b01110011;
#50 $finish;
end
//always #5 clk = ~clk ;
initial begin
$dumpfile("dump.vcd"); $dumpvars;
end
endmodule
```




Unknown_0
a[7:0]=13
b[7:0]=73
a[7:0]=13 a[7:0]
b[7:0]=73 b[7:0]
M3
a[3:0]= a[3:0]
b[7:4]= b[3:0]
result[7:0]
temp3[7:0]=
vedic4x4
M2
a[7:4]= a[3:0]
b[3:0]= b[3:0]
result[7:0]
temp2[7:0]=
vedic4x4
M1
a[3:0]= a[3:0]
b[3:0]= b[3:0]
result[7:0]
temp1[7:0]=
vedic4x4
'b00
temp3[7:0]=
'b00
temp2[7:0]=
'b000000
temp1[7:4]=
A1
a[9:0]
b[9:0]
sum[9:0]
adder10


V0
M4
a[7:4]= a[3:0]
b[7:4]= b[3:0]
result[7:0]
vedic4x4
'b00
temp3[7:0]=
'b00
temp2[7:0]=
A1
a[9:0]
b[9:0]
sum[9:0]
temp4[9:0]=
adder10
A2
temp4[9:0]= a[9:0]
b[9:0]
sum[9:0]
temp5[9:0]=
adder10
'b00
temp5[9:4]=
M1
a[3:0]= a[3:0]
b[3:0]= b[3:0]
result[7:0]
temp1[7:0]=
vedic4x4
'b000000
temp1[7:4]=
vedic8x8



Tools Used

- Cadence NCSIM for schematic and layout design.
- Spectre simulator for waveform analysis.

Simulation Results

- Functional verification of the multiplier for a range of input values.
- Transient response indicating accurate and fast computation.
- Power-delay product analysis.

6. Results and Analysis

- **Performance Metrics**:
    - Low propagation delay.
    - Optimized power consumption.
- **Comparison**:
    - Improved speed compared to traditional transistor-level multipliers.

7. Applications

The designed Vedic multiplier is suitable for applications such as:

- Signal and image processing systems.
- Low-power embedded systems.
- Cryptographic computations.

8. Conclusion and Future Work

The transistor-level design of an 8-bit Vedic multiplier in Cadence Virtuoso successfully demonstrates its efficiency in terms of speed, area, and power consumption. Future enhancements could include:

- Scaling to higher bit-width multipliers.
- Optimization for low-power applications.
- Implementation on ASIC for commercial use.

9. References

1. Tiwari, H. et al., “Design of High-Speed Vedic Multiplier Using Vedic Mathematics Techniques,” IEEE Transactions.
2. Gupta, R. et al., “Analysis of Vedic Multiplier for Digital Signal Processing,” International Journal of Engineering Research.

3. Cadence Virtuoso Design Environment User Guide.
4. Vedic Mathematics, Swami Bharati Krishna Tirthaji Maharaj.