# VLSI LAB Digital Assignment 5

Submitted by:

**Swarup Tripathy — 19BEE0167**


**School of Electrical Engineering**

Faculty: **Professor Balamurugan S**

Course: **EEE-4028**

Course Name: **VLSI Lab**

Lab Slot: **L43 + L44**

This work is submitted in partial fulfilment of the requirement of the award of the degree of Bachelor of Technology in EEE

April 21, 2022

# 4-BIT DADDA TREE MULTIPLIER

## Objectives

1. To provide students with the background needed to design, develop, and test digital arithmetic circuits using IEEE standard Verilog HDL.
2. To provide an understanding complex arithmetic circuit design principles and its architecture design.

## Outcomes

1. After completion of this course the students will be familiar with design and implementation of Digital Arithmetic building blocks using Verilog HDL and Modelsim Software.

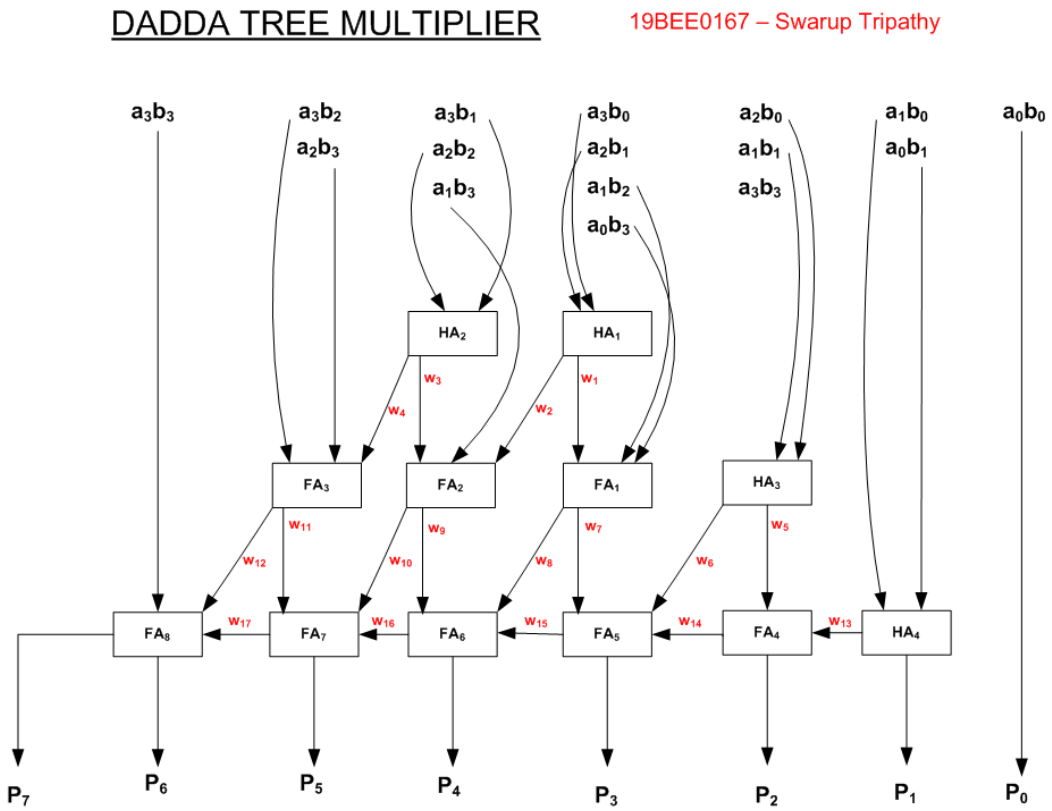
## AIM

1. Design a 4 bit dadda tree multiplier using 8 full adders and 4 half adders.

## REQUIRED SOFTWARE

1. Model sim software for simulation
2. Microsoft Visio for making flowchart
3. Documentation to be done using LaTeX

# CIRCUIT DIAGRAM




## Design Code

1. **Verilog Module** — daddaedesigncode.v

```
module dadda_tree_19BEE0167(input [3:0]a,b, output [7:0]p);
wire [17:1]w;
assign p[0]=a[0]&b[0];

//stage0
halfadder ha1((a[2]&b[1]),(a[3]&b[0]),w[1],w[2]);
halfadder ha2((a[2]&b[2]),(a[3]&b[1]),w[3],w[4]);

//stage1
halfadder ha3((a[1]&b[1]),(a[2]&b[0]),w[5],w[6]);
fulladder fa1((a[0]&b[3]),(a[1]&b[2]),w[1],w[7],w[8]);
fulladder fa2((a[1]&b[3]),w[2],w[3],w[9],w[10]);
fulladder fa3((a[2]&b[3]),(a[3]&b[2]),w[4],w[11],w[12]);


//stage2
halfadder ha4((a[0]&b[1]),(a[1]&b[0]),p[1],w[13]);
```

```
fulladder fa4((a[0]&b[2]),w[5],w[13],p[2],w[14]);
fulladder fa5(w[6],w[7],w[14],p[3],w[15]);
fulladder fa6(w[8],w[9],w[15],p[4],w[16]);
fulladder fa7(w[10],w[11],w[16],p[5],w[17]);
fulladder fa8((a[3]&b[3]),w[12],w[17],p[6],p[7]);


endmodule
```

2. **Fulladder.v**

```
module fa_df_19BEE0167(input a,b,cin,output sum,cout);
assign sum=a^b^cin;
assign cout=(a^b)&cin|(a&b);
endmodule
```

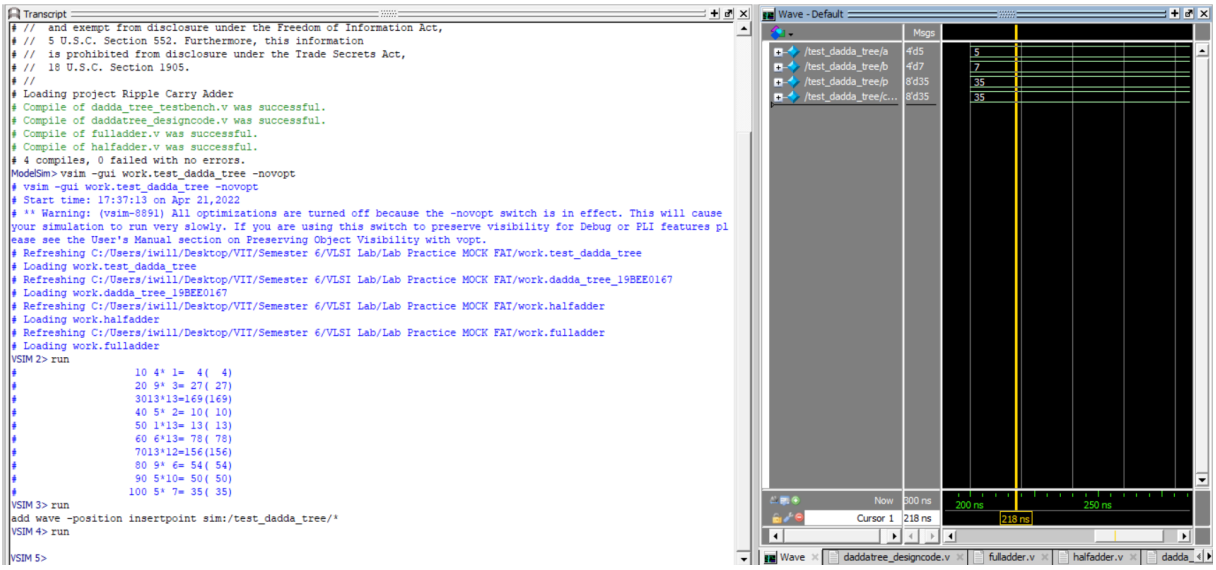
3. **Halfadder.v**

```
module half_adder(input a,b, output sum,cout);
xor x1(sum,a,b);
and a1(cout,a,b);
endmodule
```

4. **Test Fixture** — daddaTest.v

```
module test_dadda_tree();
reg [3:0]a;
reg [3:0]b;
wire [7:0]p;
reg [7:0]check;
dadda_tree_19BEE0167 uut(.a(a),.b(b),.p(p));
initial repeat(10) begin
a = $random;
b = $random;
check = a*b;
#10;
$display($time,"%d*%d=%d(%d)", a, b, p, check);
end
endmodule
```

5. **Output**




6. **Console Output**

```
#                  10 4* 1=  4(  4)
#                  20 9* 3= 27( 27)
#                  3013*13=169(169)
#                  40 5* 2= 10( 10)
#                  50 1*13= 13( 13)
#                  60 6*13= 78( 78)
#                  7013*12=156(156)
#                  80 9* 6= 54( 54)
#                  90 5*10= 50( 50)
#                 100 5* 7= 35( 35)
```

## Result

1. Successfully the 4-bit dadda tree multiplier has been designed and the output was verified.

## Inference

1. In this experiment learnt about how to construct multiple-bit adders.