Report of the 1st Project

Project Name:
Dadda Multiplier

Group Name: Leyla

Name, Surname:
Armin Asgharifard
Student No: 040190912

Name, Surname:
Emrecan Yiğit
Student No: 040190203

Completion Date

13.12.2022

Course title

Digital System Design Applications

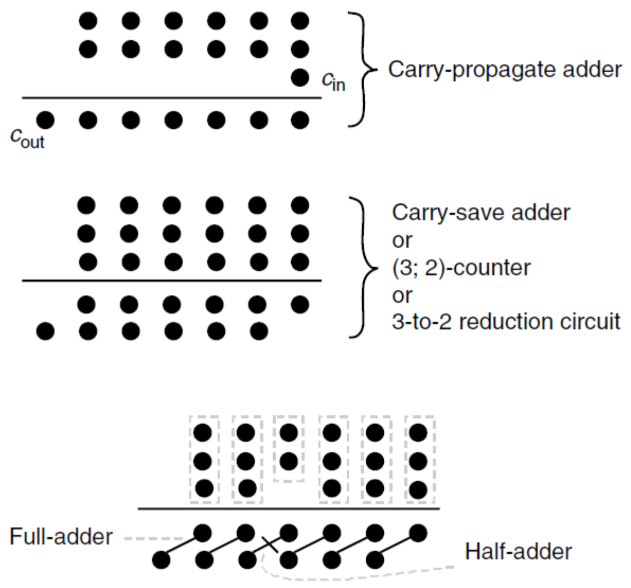
Prof. Dr. Berna Örs Yalçın

Armin Asgharifard, 040190912
Emrecan Yiğit, 040190203

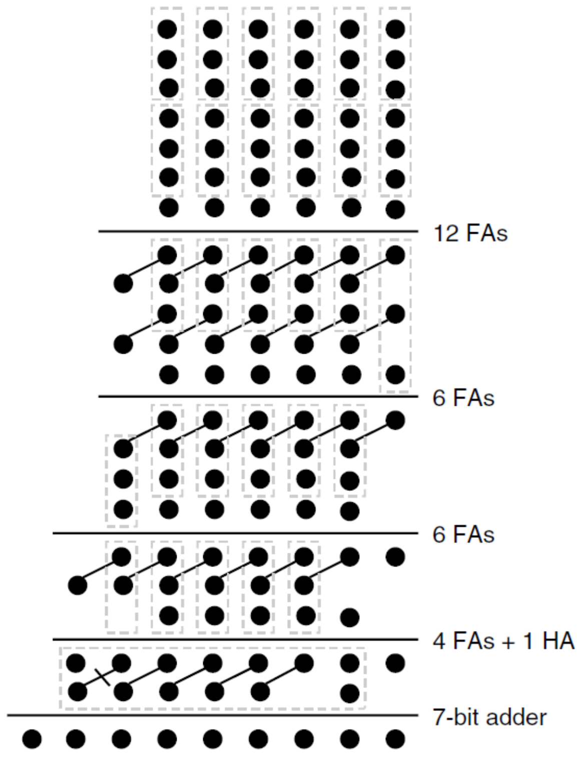
Introduction

Carry-Save Adders

We can view a row of binary FAs as a mechanism to reduce three numbers to two numbers rather than as one to reduce two numbers to their sum. To specify more precisely how the various dots are related or obtained, we agree to enclose any three dots that form the inputs to a FA, or any two dots that form the inputs to a HA, in a dashed box and to connect the sum and carry outputs of an FA, or an HA, by a diagonal line. A CSA tree can reduce *n* binary numbers to two numbers having the same sum in $O(log\, n)$ levels.



An example for adding seven 6-bit numbers is given in the figure below.



Total cost = 7-bit adder + 28 FAs + 1 HA

Dadda tree

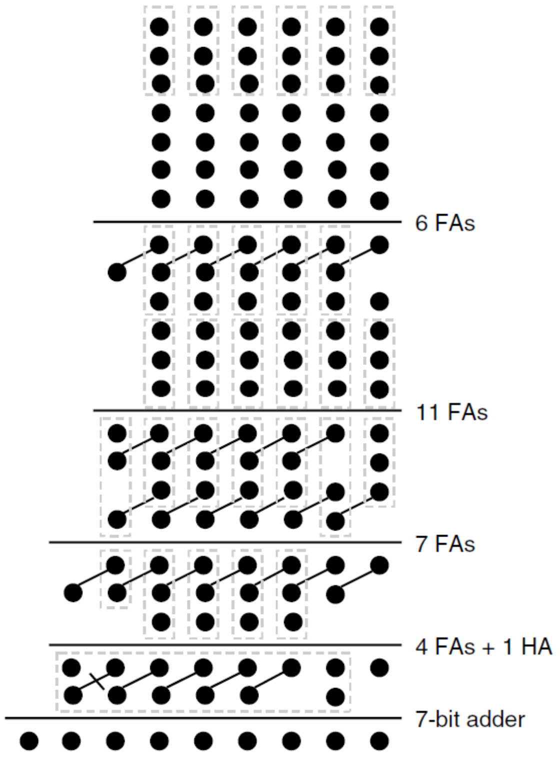
A more specific approach to perform addition is Dadda tree. Dadda algorithm specifies certain numbers that the quantity of inputs to be added shall be reduced to. The table below demonstrates the maximum number of inputs for an *n*-level CSA tree, according to Dadda's strategy.

Table 8.1 The maximum number *n*(*h*) of inputs for an *h*-level CSA tree

h	*n*(*h*)	*h*	*n*(*h*)	*h*	*n*(*h*)
0	2	7	28	14	474
1	3	8	42	15	711
2	4	9	63	16	1066
3	6	10	94	17	1599
4	9	11	141	18	2398
5	13	12	211	19	3597
6	19	13	316	20	5395

If we redo the previous example using Dadda's algorithm, we would obtain the following.

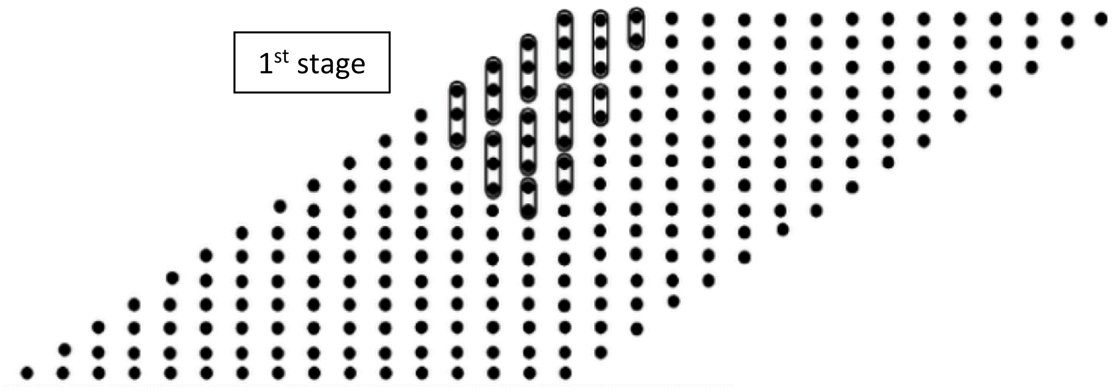
Armin Asgharifard, 040190912
Emrecan Yiğit, 040190203

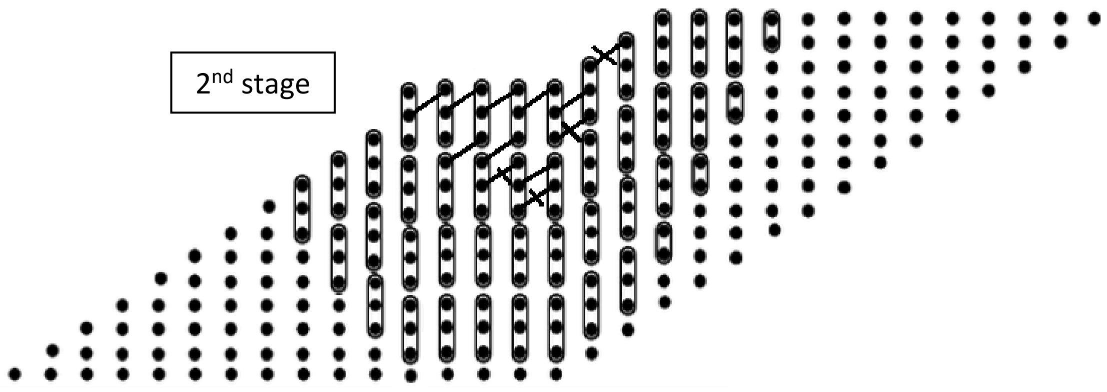


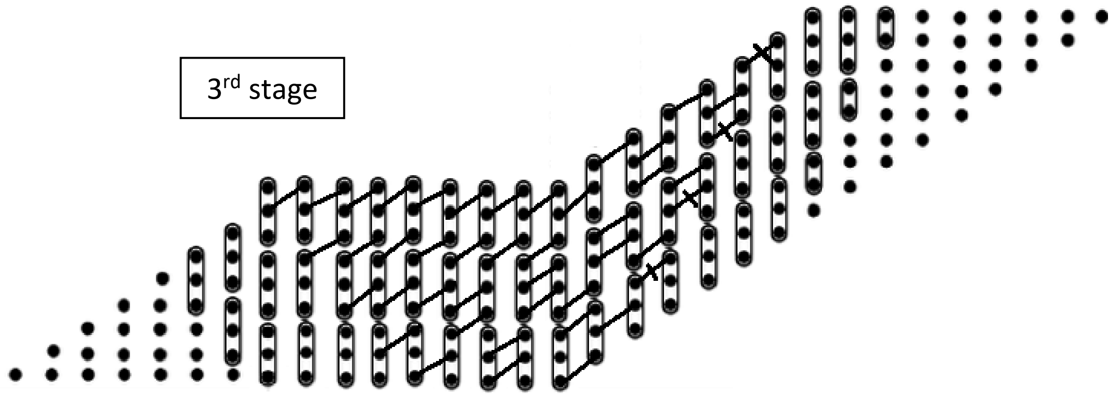
Total cost = 7-bit adder + 28 FAs + 1 HA

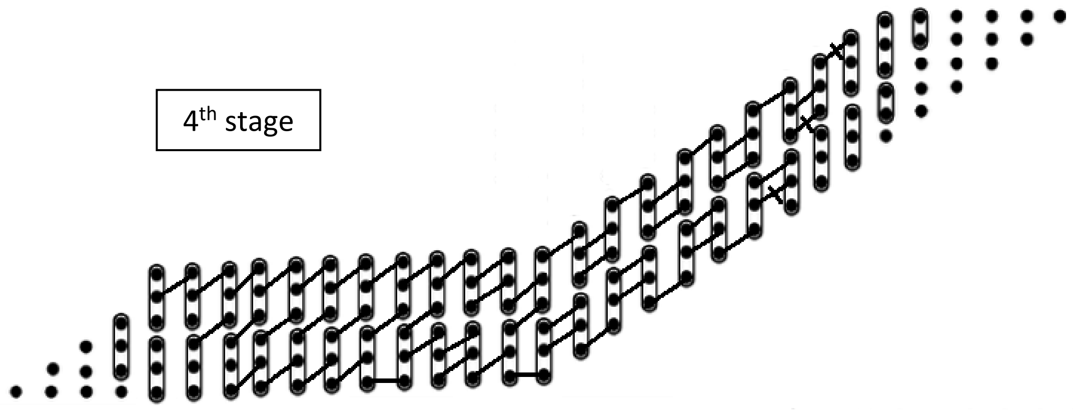
Dadda Multiplier

In this manner, we have designed a digital circuit that multiplies two 16-bit unsigned numbers using Dadda's strategy. Dadda multipliers attempt to minimize the number of gates used, as well as input/output delay.
The calculation steps are given in the following figures. At the beginning, we derive the partial products that are going to be added. Therefore, we will have sixteen 16-bit numbers as the inputs of first stage. Note that, according to Dadda, 16 inputs should be reduced to 13, then to 9, and then to 6, 4, 3, 2, in order. When 2 inputs are obtained, they will be given to a standard Ripple Carry Adder, which, together with the remaining single bit, will give us the actual result of the multiplication.

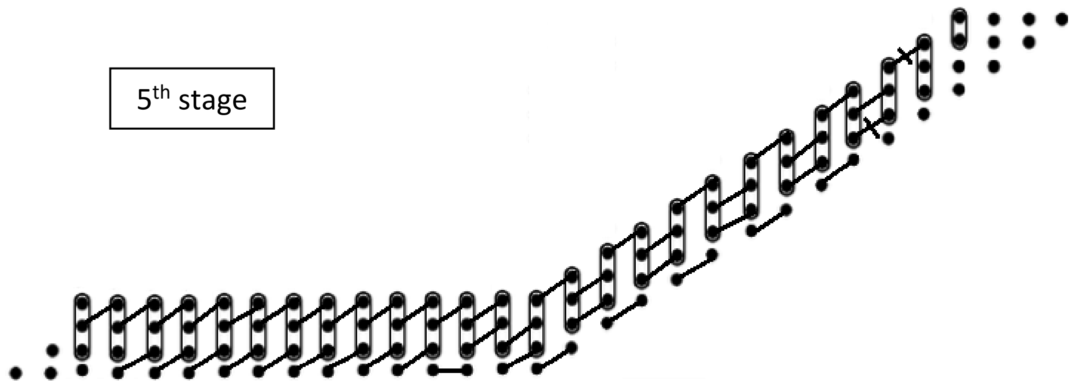



2nd stage

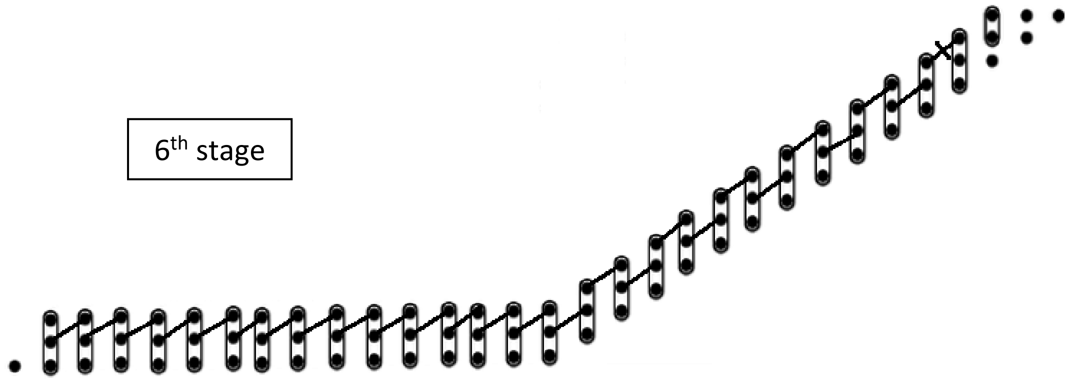

3rd stage


4th stage

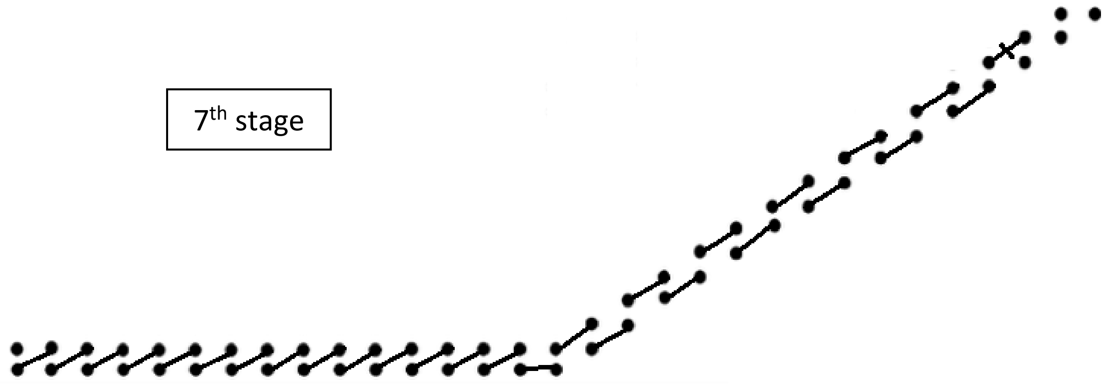
5th stage



6th stage



7th stage



A total of 196 Full Adders, 15 Half Adders, and a 31-bit RCA is generated for this design. In the next section, Verilog code of the design will be provided.

Verilog Design

Building Blocks

Basic building blocks of the design are Full Adder, Half Adder, and 31-bit Ripple Carry Adder. The design code for each of them is given below.

Design Code for Half Adder

```
module HA(
    input x,
    input y,
    output s,
    output cout
    );

    assign cout = x & y;
    assign s = x ^ y;

endmodule
```

Design Code for Full Adder

```
module FA(
    input x,
    input y,
    input ci,
    output s,
    output cout
    );

    wire s1, cout1, cout2;

    HA HA1(.x(x), .y(y), .cout(cout1), .s(s1));
    HA HA2(.x(ci), .y(s1), .cout(cout2), .s(s));
    or(cout, cout1, cout2);
endmodule
```

Design Code for Ripple Carry Adder

```
module parametric_RCA #(
    parameter SIZE = 31)(
    x, y, ci, s, cout
    );

    input [(SIZE-1):0] x, y;
    input ci;
    output [(SIZE-1):0] s;
    output cout;

    wire [SIZE:0] tmp;

    assign tmp[0] = ci;
    assign cout = tmp[SIZE];

    genvar j;
    generate
        for(j = 0; j < SIZE; j = j + 1)
        begin : FA_loop
            FA fa_n(x[j], y[j], tmp[j], s[j], tmp[j+1]);
        end
    endgenerate
endmodule
```

Dadda Multiplier Design

By instantiating the blocks designed in the previous section and mapping their inputs and outputs in accordance with the Dadda stages given in the previous section, the design process of the Dadda Multiplier can be completed. A total of 196 FAs, 15 HAs, and one 31-bit RCA is generated.

Design Code for Dadda Multiplier

```
`timescale 1ns / 1ps

(* DONT_TOUCH = "TRUE" *)
module dadda_mul(
    input [15:0] A,
    input [15:0] B,
    output [31:0] MULT
    );

    wire [30:0] x, y; // inputs to the final adder
    wire [30:0] MUL; // sum output of the final adder
    wire cout; // carry output of the final adder

    //stages of Dadda tree
    wire [16:0] stage1 [0:16];
    wire [16:0] stage2 [0:16];
    wire [16:0] stage3 [0:16];
    wire [16:0] stage4 [0:16];
    wire [16:0] stage5 [0:16];
    wire [16:0] stage6 [0:16];
    wire [16:0] stage7 [0:16];

    genvar i, j;
    for (i = 0; i <= 15; i = i + 1)
    begin
        for (j = 0; j <= 15; j = j + 1)
        begin
            assign stage1[i][j] = B[i] & A[j];
            assign stage1[j][16] = 1'b0;
        end
    end


    //stage1 to stage2 connection
```

```
FA FA1(stage1[3][15], stage1[4][14], stage1[5][13], stage2[5][13], stage2[3][16]);
FA FA2(stage1[5][12], stage1[6][11], stage1[7][10], stage2[7][10], stage2[4][14]);
FA FA3(stage1[2][15], stage1[3][14], stage1[4][13], stage2[6][11], stage2[3][15]);
HA HA1(stage1[7][9] , stage1[8][8] , stage2[8][8] , stage2[5][12]);
FA FA4(stage1[4][12], stage1[5][11], stage1[6][10], stage2[7][9],  stage2[4][13]);
FA FA5(stage1[1][15], stage1[2][14], stage1[3][13], stage2[6][10], stage2[3][14]);
HA HA2(stage1[6][9] , stage1[7][8] , stage2[7][8] , stage2[5][11]);
FA FA6(stage1[3][12], stage1[4][11], stage1[5][10], stage2[6][9],  stage2[4][12]);
FA FA7(stage1[0][15], stage1[1][14], stage1[2][13], stage2[5][10], stage2[3][13]);
HA HA3(stage1[3][11], stage1[4][10], stage2[4][10], stage2[4][11]);
FA FA8(stage1[0][14], stage1[1][13], stage1[2][12], stage2[3][11], stage2[3][12]);
HA HA0(stage1[0][13], stage1[1][12], stage2[1][12], stage2[2][12]);

assign stage2[15] = stage1[15] ;
assign stage2[14] = stage1[14];
assign stage2[13] = stage1[13];
assign stage2[12] = stage1[12];
assign stage2[11] = stage1[11];
assign stage2[10] = stage1[10];
assign stage2[9] = stage1[9];
assign stage2[8][15:9] = stage1[8][15:9] ;
assign stage2[8][7:0] = stage1[8][7:0]  ;
assign stage2[7][15:11] = stage1[7][15:11];
assign stage2[7][7:0] = stage1[7][7:0]  ;
assign stage2[6][15:12] = stage1[6][15:12];
assign stage2[6][8:0] = stage1[6][8:0]  ;
assign stage2[5][15:14] = stage1[5][15:14];
assign stage2[5][9:0] = stage1[5][9:0]  ;
assign stage2[4][15] = stage1[4][15]   ;
assign stage2[4][9:0] = stage1[4][9:0]  ;
assign stage2[3][10:0] = stage1[3][10:0] ;
assign stage2[2][11:0] = stage1[2][11:0] ;
assign stage2[1][11:0] = stage1[1][11:0] ;
assign stage2[0][12:0] = stage1[0][12:0] ;

assign stage2[8][15] = stage1[8][15]  ;
assign stage2[0][8:0] = stage1[0][8:0] ;
assign stage2[1][7:0] = stage1[1][7:0] ;
assign stage2[2][7:0] = stage1[2][7:0] ;
assign stage2[3][6:0] = stage1[3][6:0] ;
assign stage2[4][5:0] = stage1[4][5:0] ;
assign stage2[5][5:0] = stage1[5][5:0] ;
assign stage2[6][4:0] = stage1[6][4:0] ;
assign stage2[7][3:0] = stage1[7][3:0] ;
assign stage2[8][3:0] = stage1[8][3:0] ;
assign stage2[9][2:0] = stage1[9][2:0] ;
assign stage2[10][1:0] = stage1[10][1:0];
assign stage2[11][1:0] = stage1[11][1:0];
assign stage2[12][0] = stage1[12][0]  ;
assign stage2[13][0] = stage1[13][0]  ;
assign stage2[14][0] = stage1[14][0]  ;


// stage2 to stage3 connection
FA FA10(stage2[7][15], stage2[8][14], stage2[9][13], stage3[9][13], stage3[7][16]);
FA FA11(stage2[9][12], stage2[10][11],stage2[11][10],stage3[11][10],stage3[8][14]);
FA FA12(stage2[6][15], stage2[7][14], stage2[8][13], stage3[10][11],stage3[7][15]);
FA FA13(stage2[11][9], stage2[12][8], stage2[13][7], stage3[13][7], stage3[9][12]);
FA FA14(stage2[8][12], stage2[9][11], stage2[10][10],stage3[12][8], stage3[8][13]);
FA FA15(stage2[5][15], stage2[6][14], stage2[7][13], stage3[11][9], stage3[7][14]);
FA FA16(stage2[12][7], stage2[13][6], stage2[14][5], stage3[14][5], stage3[10][10]);
FA FA17(stage2[9][10], stage2[10][9], stage2[11][8], stage3[13][6], stage3[9][11]);
FA FA18(stage2[6][13], stage2[7][12], stage2[8][11], stage3[12][7], stage3[8][12]);
FA FA19(stage2[3][16], stage2[4][15], stage2[5][14], stage3[11][8], stage3[7][13]);
FA FA20(stage2[12][6], stage2[13][5], stage2[14][4], stage3[14][4], stage3[10][9]);
FA FA21(stage2[9][9],  stage2[10][8], stage2[11][7], stage3[13][5], stage3[9][10]);
FA FA22(stage2[6][12], stage2[7][11], stage2[8][10], stage3[12][6], stage3[8][11]);
FA FA23(stage2[3][15], stage2[4][14], stage2[5][13], stage3[11][7], stage3[7][12]);
FA FA24(stage2[12][5], stage2[13][4], stage2[14][3], stage3[14][3], stage3[10][8]);
FA FA25(stage2[9][8],  stage2[10][7], stage2[11][6], stage3[13][4], stage3[9][9]);
FA FA26(stage2[6][11], stage2[7][10], stage2[8][9],  stage3[12][5], stage3[8][10]);
FA FA27(stage2[3][14], stage2[4][13], stage2[5][12], stage3[11][6], stage3[7][11]);
FA FA28(stage2[12][4], stage2[13][3], stage2[14][2], stage3[14][2], stage3[10][7]);
FA FA29(stage2[9][7],  stage2[10][6], stage2[11][5], stage3[13][3], stage3[9][8]);
FA FA30(stage2[6][10], stage2[7][9],  stage2[8][8],  stage3[12][4], stage3[8][9]);
FA FA31(stage2[3][13], stage2[4][12], stage2[5][11], stage3[11][5], stage3[7][10]);
FA FA32(stage2[12][3], stage2[13][2], stage2[14][1], stage3[14][1], stage3[10][6]);
FA FA33(stage2[9][6],  stage2[10][5], stage2[11][4], stage3[13][2], stage3[9][7]);
FA FA34(stage2[6][9],  stage2[7][8],  stage2[8][7],  stage3[12][3], stage3[8][8]);
FA FA35(stage2[3][12], stage2[4][11], stage2[5][10], stage3[11][4], stage3[7][9]);
FA FA36(stage2[11][3], stage2[12][2], stage2[13][1], stage3[13][1], stage3[10][5]);
FA FA37(stage2[8][6],  stage2[9][5],  stage2[10][4], stage3[12][2], stage3[9][6]);
FA FA38(stage2[5][9],  stage2[6][8],  stage2[7][7],  stage3[11][3], stage3[8][7]);
FA FA39(stage2[2][12], stage2[3][11], stage2[4][10], stage3[10][4], stage3[7][8]);
FA FA40(stage2[10][3], stage2[11][2], stage2[12][1], stage3[12][1], stage3[9][5]);
FA FA41(stage2[7][6],  stage2[8][5],  stage2[9][4],  stage3[11][2], stage3[8][6]);
FA FA42(stage2[4][9],  stage2[5][8],  stage2[6][7],  stage3[10][3], stage3[7][7]);
FA FA43(stage2[1][12], stage2[2][11], stage2[3][10], stage3[9][4],  stage3[6][8]);
HA HA4(stage2[9][3],   stage2[10][2], stage3[10][2], stage3[8][5]);
FA FA44(stage2[6][6],  stage2[7][5],  stage2[8][4],  stage3[9][3],  stage3[7][6]);
FA FA45(stage2[3][9],  stage2[4][8],  stage2[5][7],  stage3[8][4],  stage3[6][7]);
FA FA46(stage2[0][12], stage2[1][11], stage2[2][10], stage3[7][5],  stage3[5][8]);
HA HA5 (stage2[6][5],  stage2[7][4],  stage3[7][4],  stage3[6][6]);
FA FA47(stage2[3][8],  stage2[4][7],  stage2[5][6],  stage3[6][5],  stage3[5][7]);
FA FA48(stage2[0][11], stage2[1][10], stage2[2][9],  stage3[5][6],  stage3[4][8]);
HA HA6 (stage2[3][7],  stage2[4][6],  stage3[4][6],  stage3[4][7]);
FA FA49(stage2[0][10], stage2[1][9],  stage2[2][8],  stage3[3][7],  stage3[3][8]);
HA HA7 (stage2[0][9],  stage2[1][8],  stage3[1][8],  stage3[2][8]);

assign stage3[15]       = stage2[15] ;
assign stage3[14][15:6] = stage2[14][15:6] ;
assign stage3[13][15:8] = stage2[13][15:8] ;
assign stage3[12][15:9] = stage2[12][15:9] ;
assign stage3[11][15:11]= stage2[11][15:11];
assign stage3[10][15:12]= stage2[10][15:12];
assign stage3[9][15:14] = stage2[9][15:14] ;
assign stage3[8][15]    = stage2[8][15]    ;

assign stage3[0][8:0] = stage2[0][8:0] ;
assign stage3[1][7:0] = stage2[1][7:0] ;
assign stage3[2][7:0] = stage2[2][7:0] ;
assign stage3[3][6:0] = stage2[3][6:0] ;
assign stage3[4][5:0] = stage2[4][5:0] ;
assign stage3[5][5:0] = stage2[5][5:0] ;
assign stage3[6][4:0] = stage2[6][4:0] ;
```

```
    assign stage3[7][3:0] = stage2[7][3:0] ;
    assign stage3[8][3:0] = stage2[8][3:0] ;
    assign stage3[9][2:0] = stage2[9][2:0] ;
    assign stage3[10][1:0] = stage2[10][1:0];
    assign stage3[11][1:0] = stage2[11][1:0];
    assign stage3[12][0]  = stage2[12][0]  ;
    assign stage3[13][0]  = stage2[13][0]  ;
    assign stage3[14][0]  = stage2[14][0]  ;


    //stage3 to stage4 connection
    FA FA50(stage3[10][15], stage3[11][14], stage3[12][13], stage4[12][13],
stage4[10][16]);
    FA FA51(stage3[12][12], stage3[13][11], stage3[14][10], stage4[14][10],
stage4[11][14]);
    FA FA52(stage3[9][15],  stage3[10][14], stage3[11][13], stage4[13][11],
stage4[10][15]);
    FA FA53(stage3[13][10], stage3[14][9],  stage3[15][8],  stage4[15][8],
stage4[12][12]);
    FA FA54(stage3[10][13], stage3[11][12], stage3[12][11], stage4[14][9],
stage4[11][13]);
    FA FA55(stage3[7][16],  stage3[8][15],  stage3[9][14],  stage4[13][10],
stage4[10][14]);
    FA FA56(stage3[13][9],  stage3[14][8],  stage3[15][7],  stage4[15][7],
stage4[12][11]);
    FA FA57(stage3[10][12], stage3[11][11], stage3[12][10], stage4[14][8],
stage4[11][12]);
    FA FA58(stage3[7][15],  stage3[8][14],  stage3[9][13],  stage4[13][9],
stage4[10][13]);
    FA FA59(stage3[13][8],  stage3[14][7],  stage3[15][6],  stage4[15][6],
stage4[12][10]);
    FA FA60(stage3[10][11], stage3[11][10], stage3[12][9],  stage4[14][7],
stage4[11][11]);
    FA FA61(stage3[7][14],  stage3[8][13],  stage3[9][12],  stage4[13][8],
stage4[10][12]);
    FA FA62(stage3[13][7],  stage3[14][6],  stage3[15][5],  stage4[15][5],
stage4[12][9]);
    FA FA63(stage3[10][10], stage3[11][9],  stage3[12][8],  stage4[14][6],
stage4[11][10]);
    FA FA64(stage3[7][13],  stage3[8][12],  stage3[9][11],  stage4[13][7],
stage4[10][11]);
    FA FA65(stage3[13][6],  stage3[14][5],  stage3[15][4],  stage4[15][4],
stage4[12][8]);
    FA FA66(stage3[10][9],  stage3[11][8],  stage3[12][7],  stage4[14][5],
stage4[11][9]);
    FA FA67(stage3[7][12],  stage3[8][11],  stage3[9][10],  stage4[13][6],
stage4[10][10]);
    FA FA68(stage3[13][5],  stage3[14][4],  stage3[15][3],  stage4[15][3],
stage4[12][7]);
    FA FA69(stage3[10][8],  stage3[11][7],  stage3[12][6],  stage4[14][4],
stage4[11][8]);
    FA FA70(stage3[7][11],  stage3[8][10],  stage3[9][9],   stage4[13][5],
stage4[10][9]);
    FA FA71(stage3[13][4],  stage3[14][3],  stage3[15][2],  stage4[15][2],
stage4[12][6]);
    FA FA72(stage3[10][7],  stage3[11][6],  stage3[12][5],  stage4[14][3],
stage4[11][7]);
```

```
    FA FA73(stage3[7][10],  stage3[8][9],   stage3[9][8],   stage4[13][4],
stage4[10][8]);
    FA FA74(stage3[13][3],  stage3[14][2],  stage3[15][1],  stage4[15][1],
stage4[12][5]);
    FA FA75(stage3[10][6],  stage3[11][5],  stage3[12][4],  stage4[14][2],
stage4[11][6]);
    FA FA76(stage3[7][9],   stage3[8][8],   stage3[9][7],   stage4[13][3],
stage4[10][7]);
    FA FA77(stage3[13][2],  stage3[14][1],  stage3[15][0],  stage4[15][0],
stage4[12][4]);
    FA FA78(stage3[10][5],  stage3[11][4],  stage3[12][3],  stage4[14][1],
stage4[11][5]);
    FA FA79(stage3[7][8],   stage3[8][7],   stage3[9][6],   stage4[13][2],
stage4[10][6]);
    FA FA80(stage3[12][2],  stage3[13][1],  stage3[14][0],  stage4[14][0],
stage4[12][3]);
    FA FA81(stage3[9][5],   stage3[10][4],  stage3[11][3],  stage4[13][1],
stage4[11][4]);
    FA FA82(stage3[6][8],   stage3[7][7],   stage3[8][6],   stage4[12][2],
stage4[10][5]);
    FA FA83(stage3[11][2],  stage3[12][1],  stage3[13][0],  stage4[13][0],
stage4[11][3]);
    FA FA84(stage3[8][5],   stage3[9][4],   stage3[10][3],  stage4[12][1],
stage4[10][4]);
    FA FA85(stage3[5][8],   stage3[6][7],   stage3[7][6],   stage4[11][2],
stage4[9][5]);
    FA FA86(stage3[10][2],  stage3[11][1],  stage3[12][0],  stage4[12][0],
stage4[10][3]);
    FA FA87(stage3[7][5],   stage3[8][4],   stage3[9][3],   stage4[11][1],
stage4[9][4]);
    FA FA88(stage3[4][8],   stage3[5][7],   stage3[6][6],   stage4[10][2],
stage4[8][5]);
    FA FA89(stage3[9][2],   stage3[10][1],  stage3[11][0],  stage4[11][0],
stage4[9][3]);
    FA FA90(stage3[6][5],   stage3[7][4],   stage3[8][3],   stage4[10][1],
stage4[8][4]);
    FA FA91(stage3[3][8],   stage3[4][7],   stage3[5][6],   stage4[9][2],
stage4[7][5]);
    FA FA92(stage3[8][2],   stage3[9][1],   stage3[10][0],  stage4[10][0],
stage4[8][3]);
    FA FA93(stage3[5][5],   stage3[6][4],   stage3[7][3],   stage4[9][1],
stage4[7][4]);
    FA FA94(stage3[2][8],   stage3[3][7],   stage3[4][6],   stage4[8][2],
stage4[6][5]);
    FA FA95(stage3[7][2],   stage3[8][1],   stage3[9][0],   stage4[9][0],
stage4[7][3]);
    FA FA96(stage3[4][5],   stage3[5][4],   stage3[6][3],   stage4[8][1],
stage4[6][4]);
    FA FA97(stage3[1][8],   stage3[2][7],   stage3[3][6],   stage4[7][2],
stage4[5][5]);
    HA HA8(stage3[6][2],    stage3[7][1],   stage4[7][1],   stage4[6][3]);
    FA FA98(stage3[3][5],   stage3[4][4],   stage3[5][3],   stage4[6][2],
stage4[5][4]);
    FA FA99(stage3[0][8],   stage3[1][7],   stage3[2][6],   stage4[5][3],
stage4[4][5]);
    HA HA9(stage3[3][4],    stage3[4][3],   stage4[4][3],   stage4[4][4]);
    FA FA100(stage3[0][7],  stage3[1][6],   stage3[2][5],   stage4[3][4],
stage4[3][5]);
    HA HA10(stage3[0][6],   stage3[1][5],   stage4[1][5],   stage4[2][5]);


    assign stage4[15][15] = stage3[15][15];
    assign stage4[15][14] = stage3[15][14];
    assign stage4[15][13] = stage3[15][13];
    assign stage4[15][12] = stage3[15][12];
    assign stage4[15][11] = stage3[15][11];
    assign stage4[15][10] = stage3[15][10];
    assign stage4[15][9] = stage3[15][9] ;
    assign stage4[14][15] = stage3[14][15];
    assign stage4[14][14] = stage3[14][14];
    assign stage4[14][13] = stage3[14][13];
    assign stage4[14][12] = stage3[14][12];
    assign stage4[14][11] = stage3[14][11];
    assign stage4[13][15] = stage3[13][15];
    assign stage4[13][14] = stage3[13][14];
    assign stage4[13][13] = stage3[13][13];
    assign stage4[13][12] = stage3[13][12];
    assign stage4[12][15] = stage3[12][15];
    assign stage4[12][14] = stage3[12][14];
    assign stage4[11][15] = stage3[11][15];

    assign stage4[8][0] = stage3[8][0];
    assign stage4[7][0] = stage3[7][0];
    assign stage4[6][0] = stage3[6][0];
    assign stage4[5][0] = stage3[5][0];
    assign stage4[4][0] = stage3[4][0];
    assign stage4[3][0] = stage3[3][0];
    assign stage4[2][0] = stage3[2][0];
    assign stage4[1][0] = stage3[1][0];
    assign stage4[0][0] = stage3[0][0];
    assign stage4[6][1] = stage3[6][1];
    assign stage4[5][1] = stage3[5][1];
    assign stage4[4][1] = stage3[4][1];
    assign stage4[3][1] = stage3[3][1];
    assign stage4[2][1] = stage3[2][1];
    assign stage4[1][1] = stage3[1][1];
    assign stage4[0][1] = stage3[0][1];
    assign stage4[5][2] = stage3[5][2];
    assign stage4[4][2] = stage3[4][2];
    assign stage4[3][2] = stage3[3][2];
    assign stage4[2][2] = stage3[2][2];
    assign stage4[1][2] = stage3[1][2];
    assign stage4[0][2] = stage3[0][2];
    assign stage4[3][3] = stage3[3][3];
    assign stage4[2][3] = stage3[2][3];
    assign stage4[1][3] = stage3[1][3];
    assign stage4[0][3] = stage3[0][3];
    assign stage4[2][4] = stage3[2][4];
    assign stage4[1][4] = stage3[1][4];
    assign stage4[0][4] = stage3[0][4];
    assign stage4[0][5] = stage3[0][5];

    //stage4 to stage5 connection
    FA FA101(stage4[12][15], stage4[13][14], stage4[14][13], stage5[14][13],
stage5[12][16]);
    FA FA102(stage4[13][13], stage4[14][12], stage4[15][11], stage5[15][11],
stage5[13][14]);
    FA FA103(stage4[10][16], stage4[11][15], stage4[12][14], stage5[14][12],
stage5[12][15]);
    FA FA104(stage4[13][12], stage4[14][11], stage4[15][10], stage5[15][10],
stage5[13][13]);
    FA FA105(stage4[10][15], stage4[11][14], stage4[12][13], stage5[14][11],
stage5[12][14]);
    FA FA106(stage4[13][11], stage4[14][10], stage4[15][9],  stage5[15][9],
stage5[13][12]);
    FA FA107(stage4[10][14], stage4[11][13], stage4[12][12], stage5[14][10],
stage5[12][13]);
    FA FA108(stage4[13][10], stage4[14][9],  stage4[15][8],  stage5[15][8],
stage5[13][11]);
    FA FA109(stage4[10][13], stage4[11][12], stage4[12][11], stage5[14][9],
stage5[12][12]);
    FA FA110(stage4[13][9],  stage4[14][8],  stage4[15][7],  stage5[15][7],
stage5[13][10]);
    FA FA111(stage4[10][12], stage4[11][11], stage4[12][10], stage5[14][8],
stage5[12][11]);
    FA FA112(stage4[13][8],  stage4[14][7],  stage4[15][6],  stage5[15][6],
stage5[13][9]);
    FA FA113(stage4[10][11], stage4[11][10], stage4[12][9],  stage5[14][7],
stage5[12][10]);
    FA FA114(stage4[13][7],  stage4[14][6],  stage4[15][5],  stage5[15][5],
stage5[13][8]);
    FA FA115(stage4[10][10], stage4[11][9],  stage4[12][8],  stage5[14][6],
stage5[12][9]);
    FA FA116(stage4[13][6],  stage4[14][5],  stage4[15][4],  stage5[15][4],
stage5[13][7]);
    FA FA117(stage4[10][9],  stage4[11][8],  stage4[12][7],  stage5[14][5],
stage5[12][8]);
    FA FA118(stage4[13][5],  stage4[14][4],  stage4[15][3],  stage5[15][3],
stage5[13][6]);
    FA FA119(stage4[10][8],  stage4[11][7],  stage4[12][6],  stage5[14][4],
stage5[12][7]);
    FA FA120(stage4[13][4],  stage4[14][3],  stage4[15][2],  stage5[15][2],
stage5[13][5]);
    FA FA121(stage4[10][7],  stage4[11][6],  stage4[12][5],  stage5[14][3],
stage5[12][6]);
    FA FA122(stage4[13][3],  stage4[14][2],  stage4[15][1],  stage5[15][1],
stage5[13][4]);
    FA FA123(stage4[10][6],  stage4[11][5],  stage4[12][4],  stage5[14][2],
stage5[12][5]);
    FA FA124(stage4[13][2],  stage4[14][1],  stage4[15][0],  stage5[15][0],
stage5[13][3]);
    FA FA125(stage4[10][5],  stage4[11][4],  stage4[12][3],  stage5[14][1],
stage5[12][4]);
    FA FA126(stage4[12][2],  stage4[13][1],  stage4[14][0],  stage5[14][0],
stage5[13][2]);
    FA FA127(stage4[9][5],   stage4[10][4],  stage4[11][3],  stage5[13][1],
stage5[12][3]);
    FA FA128(stage4[11][2],  stage4[12][1],  stage4[13][0],  stage5[13][0],
stage5[12][2]);
```

```
    FA FA129(stage4[8][5],   stage4[9][4],   stage4[10][3],  stage5[12][1],
stage5[11][3]);
    FA FA130(stage4[10][2],  stage4[11][1],  stage4[12][0],  stage5[12][0],
stage5[11][2]);
    FA FA131(stage4[7][5],   stage4[8][4],   stage4[9][3],   stage5[11][1],
stage5[10][3]);
    FA FA132(stage4[9][2],   stage4[10][1],  stage4[11][0],  stage5[11][0],
stage5[10][2]);
    FA FA133(stage4[6][5],   stage4[7][4],   stage4[8][3],   stage5[10][1],
stage5[9][3]);
    FA FA134(stage4[8][2],   stage4[9][1],   stage4[10][0],  stage5[10][0],
stage5[9][2]);
    FA FA135(stage4[5][5],   stage4[6][4],   stage4[7][3],   stage5[9][1],
stage5[8][3]);
    FA FA136(stage4[7][2],   stage4[8][1],   stage4[9][0],   stage5[9][0],
stage5[8][2]);
    FA FA137(stage4[4][5],   stage4[5][4],   stage4[6][3],   stage5[8][1],
stage5[7][3]);
    FA FA138(stage4[6][2],   stage4[7][1],   stage4[8][0],   stage5[8][0],
stage5[7][2]);
    FA FA139(stage4[3][5],   stage4[4][4],   stage4[5][3],   stage5[7][1],
stage5[6][3]);
    FA FA140(stage4[5][2],   stage4[6][1],   stage4[7][0],   stage5[7][0],
stage5[6][2]);
    FA FA141(stage4[2][5],   stage4[3][4],   stage4[4][3],   stage5[6][1],
stage5[5][3]);
    FA FA142(stage4[4][2],   stage4[5][1],   stage4[6][0],   stage5[6][0],
stage5[5][2]);
    FA FA143(stage4[1][5],   stage4[2][4],   stage4[3][3],   stage5[5][1],
stage5[4][3]);
    HA HA11 (stage4[3][2],   stage4[4][1],   stage5[4][1],   stage5[4][2]);
    FA FA144(stage4[0][5],   stage4[1][4],   stage4[2][3],   stage5[3][2],
stage5[3][3]);
    HA HA12 (stage4[0][4],   stage4[1][3],   stage5[1][3],   stage5[2][3]);

    assign stage5[15][15] = stage4[15][15];
    assign stage5[15][14] = stage4[15][14];
    assign stage5[15][13] = stage4[15][13];
    assign stage5[15][12] = stage4[15][12];
    assign stage5[14][15] = stage4[14][15];
    assign stage5[14][14] = stage4[14][14];
    assign stage5[13][15] = stage4[13][15];
    assign stage5[5][0] = stage4[5][0];
    assign stage5[4][0] = stage4[4][0];
    assign stage5[3][0] = stage4[3][0];
    assign stage5[2][0] = stage4[2][0];
    assign stage5[1][0] = stage4[1][0];
    assign stage5[0][0] = stage4[0][0];
    assign stage5[3][1] = stage4[3][1];
    assign stage5[2][1] = stage4[2][1];
    assign stage5[1][1] = stage4[1][1];
    assign stage5[0][1] = stage4[0][1];
    assign stage5[2][2] = stage4[2][2];
    assign stage5[1][2] = stage4[1][2];
    assign stage5[0][2] = stage4[0][2];
    assign stage5[0][3] = stage4[0][3];
    //stage5 to stage6 connection
    FA FA145(stage5[12][16], stage5[13][15], stage5[14][14], stage6[14][14],
stage6[13][16]);
    FA FA146(stage5[12][15], stage5[13][14], stage5[14][13], stage6[14][13],
stage6[13][15]);
    FA FA147(stage5[12][14], stage5[13][13], stage5[14][12], stage6[14][12],
stage6[13][14]);
    FA FA148(stage5[12][13], stage5[13][12], stage5[14][11], stage6[14][11],
stage6[13][13]);
    FA FA149(stage5[12][12], stage5[13][11], stage5[14][10], stage6[14][10],
stage6[13][12]);
    FA FA150(stage5[12][11], stage5[13][10], stage5[14][9],  stage6[14][9],
stage6[13][11]);
    FA FA151(stage5[12][10], stage5[13][9],  stage5[14][8],  stage6[14][8],
stage6[13][10]);
    FA FA152(stage5[12][9],  stage5[13][8],  stage5[14][7],  stage6[14][7],
stage6[13][9]);
    FA FA153(stage5[12][8],  stage5[13][7],  stage5[14][6],  stage6[14][6],
stage6[13][8]);
    FA FA154(stage5[12][7],  stage5[13][6],  stage5[14][5],  stage6[14][5],
stage6[13][7]);
    FA FA155(stage5[12][6],  stage5[13][5],  stage5[14][4],  stage6[14][4],
stage6[13][6]);
    FA FA156(stage5[12][5],  stage5[13][4],  stage5[14][3],  stage6[14][3],
stage6[13][5]);
    FA FA157(stage5[12][4],  stage5[13][3],  stage5[14][2],  stage6[14][2],
stage6[13][4]);
    FA FA158(stage5[12][3],  stage5[13][2],  stage5[14][1],  stage6[14][1],
stage6[13][3]);
    FA FA159(stage5[11][3],  stage5[12][2],  stage5[13][1],  stage6[13][1],
stage6[13][2]);
    FA FA160(stage5[10][3],  stage5[11][2],  stage5[12][1],  stage6[12][1],
stage6[12][2]);
    FA FA161(stage5[9] [3],  stage5[10][2],  stage5[11][1],  stage6[11][1],
stage6[11][2]);
    FA FA162(stage5[8] [3],  stage5[9] [2],  stage5[10][1],  stage6[10][1],
stage6[10][2]);
    FA FA163(stage5[7] [3],  stage5[8] [2],  stage5[9] [1],  stage6[9] [1],  stage6[9]
[2]);
    FA FA164(stage5[6] [3],  stage5[7] [2],  stage5[8] [1],  stage6[8] [1],  stage6[8]
[2]);
    FA FA165(stage5[5] [3],  stage5[6] [2],  stage5[7] [1],  stage6[7] [1],  stage6[7]
[2]);
    FA FA166(stage5[4] [3],  stage5[5] [2],  stage5[6] [1],  stage6[6] [1],  stage6[6]
[2]);
    FA FA167(stage5[3] [3],  stage5[4] [2],  stage5[5] [1],  stage6[5] [1],  stage6[5]
[2]);
    FA FA168(stage5[2] [3],  stage5[3] [2],  stage5[4] [1],  stage6[4] [1],  stage6[4]
[2]);
    FA FA169(stage5[1] [3],  stage5[2] [2],  stage5[3] [1],  stage6[3] [1],  stage6[3]
[2]);
    HA HA13 (stage5[0] [3],  stage5[1] [2],  stage6[1] [2],  stage6[2] [2]);


    assign stage6[15][15] = stage5[15][15];
    assign stage6[15][14] = stage5[15][14];
    assign stage6[15][13] = stage5[15][13];
    assign stage6[15][12] = stage5[15][12];
    assign stage6[15][11] = stage5[15][11];
    assign stage6[15][10] = stage5[15][10];
    assign stage6[15][9] = stage5[15][9] ;
    assign stage6[15][8] = stage5[15][8] ;
    assign stage6[15][7] = stage5[15][7] ;
    assign stage6[15][6] = stage5[15][6] ;
    assign stage6[15][5] = stage5[15][5] ;
    assign stage6[15][4] = stage5[15][4] ;
    assign stage6[15][3] = stage5[15][3] ;
    assign stage6[15][2] = stage5[15][2] ;
    assign stage6[15][1] = stage5[15][1] ;
    assign stage6[15][0] = stage5[15][0] ;
    assign stage6[14][15] = stage5[14][15] ;
    assign stage6[14][0] = stage5[14][0];
    assign stage6[13][0] = stage5[13][0];
    assign stage6[12][0] = stage5[12][0];
    assign stage6[11][0] = stage5[11][0];
    assign stage6[10][0] = stage5[10][0];
    assign stage6[9][0] = stage5[9][0] ;
    assign stage6[8][0] = stage5[8][0] ;
    assign stage6[7][0] = stage5[7][0] ;
    assign stage6[6][0] = stage5[6][0] ;
    assign stage6[5][0] = stage5[5][0] ;
    assign stage6[4][0] = stage5[4][0] ;
    assign stage6[3][0] = stage5[3][0] ;
    assign stage6[2][0] = stage5[2][0] ;
    assign stage6[1][0] = stage5[1][0] ;
    assign stage6[0][0] = stage5[0][0] ;
    assign stage6[0][1] = stage5[0][1];
    assign stage6[0][2] = stage5[0][2];
    assign stage6[1][1] = stage5[1][1];
    assign stage6[2][1] = stage5[2][1];


    //stage6 to stage7 connection
    FA FA170(stage6[13][16], stage6[14][15], stage6[15][14], stage7[15][14],
stage7[14][16]);
    FA FA171(stage6[13][15], stage6[14][14], stage6[15][13], stage7[15][13],
stage7[14][15]);
    FA FA172(stage6[13][14], stage6[14][13], stage6[15][12], stage7[15][12],
stage7[14][14]);
    FA FA173(stage6[13][13], stage6[14][12], stage6[15][11], stage7[15][11],
stage7[14][13]);
    FA FA174(stage6[13][12], stage6[14][11], stage6[15][10], stage7[15][10],
stage7[14][12]);
    FA FA175(stage6[13][11], stage6[14][10], stage6[15][9],  stage7[15][9],
stage7[14][11]);
    FA FA176(stage6[13][10], stage6[14][9],  stage6[15][8],  stage7[15][8],
stage7[14][10]);
    FA FA177(stage6[13][9],  stage6[14][8],  stage6[15][7],  stage7[15][7],
stage7[14][9]);
    FA FA178(stage6[13][8],  stage6[14][7],  stage6[15][6],  stage7[15][6],
stage7[14][8]);
    FA FA179(stage6[13][7],  stage6[14][6],  stage6[15][5],  stage7[15][5],
stage7[14][7]);
    FA FA180(stage6[13][6],  stage6[14][5],  stage6[15][4],  stage7[15][4],
stage7[14][6]);
```

```
    FA FA181(stage6[13][5],  stage6[14][4],  stage6[15][3],  stage7[15][3],
stage7[14][5]);
    FA FA182(stage6[13][4],  stage6[14][3],  stage6[15][2],  stage7[15][2],
stage7[14][4]);
    FA FA183(stage6[13][3],  stage6[14][2],  stage6[15][1],  stage7[15][1],
stage7[14][3]);
    FA FA184(stage6[13][2],  stage6[14][1],  stage6[15][0],  stage7[15][0],
stage7[14][2]);
    FA FA185(stage6[12][2],  stage6[13][1],  stage6[14][0],  stage7[14][0],
stage7[14][1]);
    FA FA186(stage6[11][2],  stage6[12][1],  stage6[13][0],  stage7[13][0],
stage7[13][1]);
    FA FA187(stage6[10][2],  stage6[11][1],  stage6[12][0],  stage7[12][0],
stage7[12][1]);
    FA FA188(stage6[9] [2],  stage6[10][1],  stage6[11][0],  stage7[11][0],
stage7[11][1]);
    FA FA189(stage6[8] [2],  stage6[9] [1],  stage6[10][0],  stage7[10][0],
stage7[10][1]);
    FA FA190(stage6[7] [2],  stage6[8] [1],  stage6[9] [0],  stage7[9] [0],  stage7[9]
[1]);
    FA FA191(stage6[6] [2],  stage6[7] [1],  stage6[8] [0],  stage7[8] [0],  stage7[8]
[1]);
    FA FA192(stage6[5] [2],  stage6[6] [1],  stage6[7] [0],  stage7[7] [0],  stage7[7]
[1]);
    FA FA193(stage6[4] [2],  stage6[5] [1],  stage6[6] [0],  stage7[6] [0],  stage7[6]
[1]);
    FA FA194(stage6[3] [2],  stage6[4] [1],  stage6[5] [0],  stage7[5] [0],  stage7[5]
[1]);
    FA FA195(stage6[2] [2],  stage6[3] [1],  stage6[4] [0],  stage7[4] [0],  stage7[4]
[1]);
    FA FA196(stage6[1] [2],  stage6[2] [1],  stage6[3] [0],  stage7[3] [0],  stage7[3]
[1]);
    HA HA14 (stage6[0] [2],  stage6[1] [1],  stage7[1] [1],  stage7[2] [1]);

    assign stage7[15][15] = stage6[15][15];
    assign stage7[2][0] = stage6[2][0];
    assign stage7[1][0] = stage6[1][0];
    assign stage7[0][1] = stage6[0][1];
    assign stage7[0][0] = stage6[0][0];

    //final addition stage
    assign x[30:15] = stage7[14][16:1];
    assign y[30:15] = stage7[15][15:0];
    assign x[14] =   stage7[13][1]   ;
    assign x[13] =   stage7[12][1]   ;
    assign x[12] =   stage7[11][1]   ;
    assign x[11] =   stage7[10][1]   ;
    assign x[10] =   stage7[9][1]    ;
    assign x[9] =    stage7[8][1]    ;
    assign x[8] =    stage7[7][1]    ;
    assign x[7] =    stage7[6][1]    ;
    assign x[6] =    stage7[5][1]    ;
    assign x[5] =    stage7[4][1]    ;
    assign x[4] =    stage7[3][1]    ;
    assign x[3] =    stage7[2][1]    ;
    assign x[2] =    stage7[1][1]    ;
    assign x[1] =    stage7[0][1]    ;
    assign x[0] = 1'b0              ;
    assign y[14] =   stage7[14][0]   ;
    assign y[13] =   stage7[13][0]   ;
    assign y[12] =   stage7[12][0]   ;
    assign y[11] =   stage7[11][0]   ;
    assign y[10] =   stage7[10][0]   ;
    assign y[9] =    stage7[9][0]    ;
    assign y[8] =    stage7[8][0]    ;
    assign y[7] =    stage7[7][0]    ;
    assign y[6] =    stage7[6][0]    ;
    assign y[5] =    stage7[5][0]    ;
    assign y[4] =    stage7[4][0]    ;
    assign y[3] =    stage7[3][0]    ;
    assign y[2] =    stage7[2][0]    ;
    assign y[1] =    stage7[1][0]    ;
    assign y[0] =    stage7[0][0]    ;

    parametric_RCA add(x, y, 1'b0, MUL, cout);
    assign MULT[31] = cout;
    assign MULT[30:0] = MUL;
endmodule
```

Behavioral Simulation

To ensure that the multiplier works perfectly, testing 100 input pairs is sufficient. In this regard, a Python code is written to generate 200 random 16-bit numbers, and write them to a .txt file, which we call as the stimulus file.
Using Verilog file I/O, we can read the stimulus file, and assign them to inputs of the multiplier in the testbench code. Therefore, a few lines of code will simulate the design with 100 input pairs.
The results of the simulation will be demonstrated in Tcl console, and also, they will be written to a new .txt file. In the console or the text file, you will see each input pair, with the output that the circuit gives and the output that we expect, and whether they match or not.
In the testbench code, we have indicated that the stimulus file path is “D:\random_numbers.txt”, and the results text file path is “D:\results.txt”. Please, keep in mind.

Python Code for Generating 200 Random Numbers

```
import random
f = open("random_numbers.txt", "x")
f = open("random_numbers.txt", "w")
for k in range (200):
    for i in range (16):
        a=(random.randint(0,1))
        f.write(str(a))
    f.write("\n")
```

Testbench Code for Dadda Multiplier

```
`timescale 1ns / 1ps

module dadda_tb();
    // the stimulus txt file must be stored in "D:\"
    // the results of the simulation will also be stored in "D:\"
    reg [15:0] memory [1:200];
    reg [15:0] a;
    reg [15:0] b;
    wire [31:0] result;
    integer i, file;

    dadda_mul dadda(a, b, result);

    wire [31:0] tmp;
    assign tmp = a*b;

    initial
    begin
        $readmemb("D:\random_numbers.txt", memory);
        file = $fopen("D:\results.txt", "a");
        $fmonitor(file, "A = b'%0b' = %0d   B = b'%0b' = %0d    Obtained_Result = b'%0b'
= %0d  Expected_Result = b'%0b' = %0d  Status = %0s", a, a, b, b, result, result, tmp,
tmp, tmp == result ? "TRUE" : "FALSE");
        $monitor(file, "A = b'%0b' = %0d   B = b'%0b' = %0d    Obtained_Result = b'%0b' =
%0d  Expected_Result = b'%0b' = %0d  Status = %0s", a, a, b, b, result, result, tmp, tmp,
tmp == result ? "TRUE" : "FALSE");
        for (i = 1; i <= 200; i = i + 2)
        begin
            a = memory [i];
            b = memory [i + 1];
            #10;
        end
        $finish;
    end
endmodule
```

Stimulus File Example

```
1100101000010101
0100000101010011
1100111110101000
0111000001110111
1000100011101000
0011100000100010
1011010110110111
0100000101101010
1001010000111110
1110110100111100
1110100100110001
0111001110111001
0000010010101101
0111001101000101
1000110101010001
1000001000100011
1110100111011101
1110000101010110
0100111000001110
0110001100011110
0110110101001001
1010001111011100
1110000011010011
1111101010001000
1111011100100111
0111000101111001
0100100100001100
0001111100011101
0101110110101011
0101011001001101
0110100100001001
0111111111110110
1001000101000101
1100101101001111
1010011010100101
1001110010101001
0011001110100001
0010010011001011
1111101110000101
```

```
1001011101111001
1000011101010000
1101110110111001
0001100010001010
1100111000100000
0010010001000010
1110000111111000
0110111010000011
0011011100100011
0011111111100101
0010111001110111
1001010100110000
1001000011010101
0000011001011110
1100110111001111
1100000010011101
1000001010110010
0011100001110000
1110100000011100
0000100010000111
0010101101101110
0000011100001111
0110111001001111
0011001010010001
0110001001100100
0001001111101011
1000101101111011
0001101111011010
0100000111010101
1000111010110100
1100001000001100
1001011101010101
0000001001001011
1011100010001001
0111000001100001
1001110001000001
1010101011111001
1011001101011011
1010011010101101
1110000010001100
1011110001000010
0011100010011101
1000011111111001
0010010000101010
0001011111101100
1011001100100010
1001101011100100
1100010001111000
1000110011111100
0001101011111001
0110110010000100
1011001011111111
0110001010111101
0010101100101011
0110110100010011
0100111110011100
1001100101110010
0010001111111110
0101010010111011
0111101110110011
0100000011111000
0010001110100100
0100011100011010
0110111001011010
1101110110010110
1000001000000000
0011000010110001
0110010010001111
1000011111111000
1010011110101110
1001100010010001
0111001111101010
1011101101110111
1010000110010110
1110101101100101
1011110000001101
0111010010011001
1001011010110111
0111001100000100
1110001001110001
0011010111011011
1101001010101000
0011000011100010
0001011011010001
0111101110100011
0011110001111111
0010001111010100
0000001011111001
1011111001010100
1010100000001101
1011110010011111
0111110011101100
1100010000011000
1100011110001000
1100100011011100
0111000001100000
0000000001111111
```

Armin Asgharifard, 040190912
Emrecan Yiğit, 040190203

```
1101001100110101
1010000110111101
1110011110001001
1011000110100110
0110010110110111
0100000110011111
1010101100011010
0010010110111110
0001111100111010
0101100101001010
1101010100111100
1100011001101011
1000010110010101
1100000101111000
0101110110111010
0110001100001111
0100100100001111
0110010001110111
1110100100001001
0000100111000010
1010011010111000
1101010010101000
0010111001111100
1101100001110110
1001111011000000
1101111101111101
1100010111000000
0011100110000011
0011101101111010
0011000011101010
0001011010110010
1000010010011010
0101110011001000
0110101101011011
0110001111100101
0111001111011010
0011000110000010
0100110011101010
1101011110000000
0100101110110010
0010000110011111
0111111000011111
1001000100010000
1010001000110111
0110110001101101
0110001101101000
1111111101000001
0011001011100110
1110111111010000
0111010111101100
0000000010001011
0101100110010110
1100110110000000
0110101000111010
0111101110100010
0001010001011101
0110010110110111
1111110110110011
0110011010000001
1111111110011001
1010101011000101
0100010000001111
0101001110011111
1000101000111011
```

Results File Example

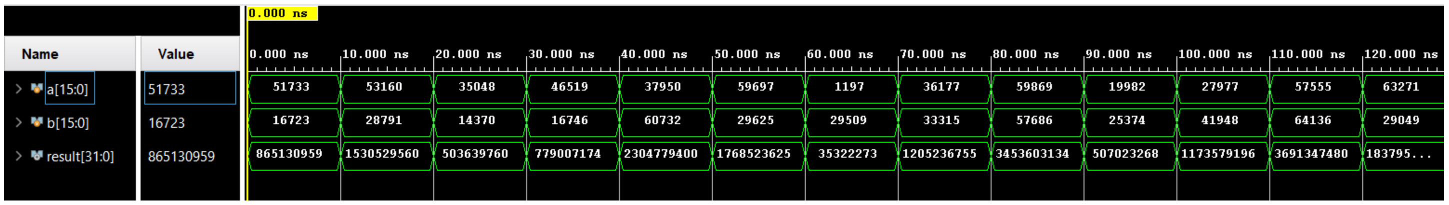
Zoom in to see better.

```
A = b'1100101000010101' = 51733   B = b'100000101010011' = 16723    Obtained_Result = b'110011100100001101100111001111' = 865130959  Expected_Result = b'110011100100001101100111001111' = 865130959  Status = TRUE
A = b'1100111110101000' = 53160   B = b'111000001110111' = 28791    Obtained_Result = b'1011011001110100000011100011000' = 1530529560  Expected_Result = b'1011011001110100000011100011000' = 1530529560  Status = TRUE
A = b'1000100011101000' = 35048   B = b'11100000100010' = 14370    Obtained_Result = b'11110000001001110111011010000' = 503639760  Expected_Result = b'11110000001001110111011010000' = 503639760  Status = TRUE
A = b'1011010110110111' = 46519   B = b'100000101101010' = 16746    Obtained_Result = b'101110011011101011010011000110' = 779007174  Expected_Result = b'101110011011101011010011000110' = 779007174  Status = TRUE
A = b'1001010000111110' = 37950   B = b'1110110100111100' = 60732    Obtained_Result = b'10001001011000000010010010001000' = 2304779400  Expected_Result = b'10001001011000000010010010001000' = 2304779400  Status = TRUE
A = b'1110100100110001' = 59697   B = b'111001110111001' = 29625    Obtained_Result = b'1101001011010011000011101101001' = 1768523625  Expected_Result = b'1101001011010011000011101101001' = 1768523625  Status = TRUE
A = b'10010101101' = 1197   B = b'111001101000101' = 29509    Obtained_Result = b'10000110101111100110100001' = 35322273  Expected_Result = b'10000110101111100110100001' = 35322273  Status = TRUE
A = b'1000110101010001' = 36177   B = b'1000001000100011' = 33315    Obtained_Result = b'1000111110101100111010000010011' = 1205236755  Expected_Result = b'1000111110101100111010000010011' = 1205236755  Status = TRUE
A = b'1110100111011101' = 59869   B = b'1110000101010110' = 57686    Obtained_Result = b'11001101110110011100110100111110' = 3453603134  Expected_Result = b'11001101110110011100110100111110' = 3453603134  Status = TRUE
A = b'100111000001110' = 19982   B = b'110001100011110' = 25374    Obtained_Result = b'11110001110001000111110100100' = 507023268  Expected_Result = b'11110001110001000111110100100' = 507023268  Status = TRUE
A = b'110110101001001' = 27977   B = b'1010001111011100' = 41948    Obtained_Result = b'1000101111100110110010110111100' = 1173579196  Expected_Result = b'1000101111100110110010110111100' = 1173579196  Status = TRUE
A = b'1110000011010011' = 57555   B = b'1111101010001000' = 64136    Obtained_Result = b'11011100000001010111111000011000' = 3691347480  Expected_Result = b'11011100000001010111111000011000' = 3691347480  Status = TRUE
A = b'1111011100100111' = 63271   B = b'111000101111001' = 29049    Obtained_Result = b'1101101100011010000100001101111' = 1837959279  Expected_Result = b'1101101100011010000100001101111' = 1837959279  Status = TRUE
A = b'100100100001100' = 18700   B = b'1111100011101' = 7965    Obtained_Result = b'1000111000001011101001011100' = 148945500  Expected_Result = b'1000111000001011101001011100' = 148945500  Status = TRUE
A = b'101110110101011' = 23979   B = b'101011001001101' = 22093    Obtained_Result = b'11111100100111001111001101111' = 529768047  Expected_Result = b'11111100100111001111001101111' = 529768047  Status = TRUE
A = b'110100100001001' = 26889   B = b'111111111110110' = 32758    Obtained_Result = b'110100100000000110010110100110' = 880829862  Expected_Result = b'110100100000000110010110100110' = 880829862  Status = TRUE
A = b'1001000101000101' = 37189   B = b'1100101101001111' = 52047    Obtained_Result = b'1110011010111101000101101001011' = 1935575883  Expected_Result = b'1110011010111101000101101001011' = 1935575883  Status = TRUE
A = b'1010011010100101' = 42661   B = b'1001110010101001' = 40105    Obtained_Result = b'1100101111110101000111011101101' = 1710919405  Expected_Result = b'1100101111110101000111011101101' = 1710919405  Status = TRUE
A = b'11001110100001' = 13217   B = b'10010011001011' = 9419    Obtained_Result = b'111011010111001010010101011' = 124490923  Expected_Result = b'111011010111001010010101011' = 124490923  Status = TRUE
A = b'1111101110000101' = 64389   B = b'1001011101111001' = 38777    Obtained_Result = b'10010100110100100101010011011101' = 2496812253  Expected_Result = b'10010100110100100101010011011101' = 2496812253  Status = TRUE
A = b'1000011101010000' = 34640   B = b'1101110110111001' = 56761    Obtained_Result = b'1110101001100011101100011010000' = 1966201040  Expected_Result = b'1110101001100011101100011010000' = 1966201040  Status = TRUE
A = b'1100010001010' = 6282   B = b'1100111000100000' = 52768    Obtained_Result = b'10011110000100001110101000000' = 331488576  Expected_Result = b'10011110000100001110101000000' = 331488576  Status = TRUE
A = b'10010001000010' = 9282   B = b'1110000111111000' = 57848    Obtained_Result = b'100000000000010010000111110000' = 536945136  Expected_Result = b'100000000000010010000111110000' = 536945136  Status = TRUE
A = b'110111010000011' = 28291   B = b'11011100100011' = 14115    Obtained_Result = b'10111110011010100000011101001' = 399327465  Expected_Result = b'10111110011010100000011101001' = 399327465  Status = TRUE
A = b'11111111100101' = 16357   B = b'10111001110111' = 11895    Obtained_Result = b'1011100110001101100101110011' = 194566515  Expected_Result = b'1011100110001101100101110011' = 194566515  Status = TRUE
A = b'1001010100110000' = 38192   B = b'1001000011010101' = 37077    Obtained_Result = b'1010100011001110010000011110000' = 1416044784  Expected_Result = b'1010100011001110010000011110000' = 1416044784  Status = TRUE
A = b'11001011110' = 1630   B = b'1100110111001111' = 52687    Obtained_Result = b'101000111100110110000000010' = 85879810  Expected_Result = b'101000111100110110000000010' = 85879810  Status = TRUE
A = b'1100000010011101' = 49309   B = b'1000001010110010' = 33458    Obtained_Result = b'1100010010101011010011100101010' = 1649780522  Expected_Result = b'1100010010101011010011100101010' = 1649780522  Status = TRUE
A = b'11100001110000' = 14448   B = b'1110100000011100' = 59420    Obtained_Result = b'110011001010111010110001000000' = 858500160  Expected_Result = b'110011001010111010110001000000' = 858500160  Status = TRUE
A = b'100010000111' = 2183   B = b'10101101101110' = 11118    Obtained_Result = b'1011100100101011100000010' = 24270594  Expected_Result = b'1011100100101011100000010' = 24270594  Status = TRUE
A = b'11100001111' = 1807   B = b'110111001001111' = 28239    Obtained_Result = b'11000010101001111110100001' = 51027873  Expected_Result = b'11000010101001111110100001' = 51027873  Status = TRUE
A = b'11001010010001' = 12945   B = b'110001001100100' = 25188    Obtained_Result = b'10011011011110100001010100100' = 326058660  Expected_Result = b'10011011011110100001010100100' = 326058660  Status = TRUE
A = b'1001111101011' = 5099   B = b'1000101101111011' = 35707    Obtained_Result = b'1010110110100010101011101001' = 182069993  Expected_Result = b'1010110110100010101011101001' = 182069993  Status = TRUE
```

```
A = b'1101111011010' = 7130   B = b'100000111010101' = 16853    Obtained_Result = b'111001010011000011001100010' = 120161890  Expected_Result = b'111001010011000011001100010' = 120161890  Status = TRUE
A = b'1000111010110100' = 36532   B = b'1100001000001100' = 49676    Obtained_Result = b'1101100001010110001100001110000' = 1814763632  Expected_Result = b'1101100001010110001100001110000' = 1814763632  Status = TRUE
A = b'1001011101010101' = 38741   B = b'1001001011' = 587    Obtained_Result = b'1010110101111111111100111' = 22740967  Expected_Result = b'1010110101111111111100111' = 22740967  Status = TRUE
A = b'1011100010001001' = 47241   B = b'111000001100001' = 28769    Obtained_Result = b'1010001000000001110110111101001' = 1359076329  Expected_Result = b'1010001000000001110110111101001' = 1359076329  Status = TRUE
A = b'1001110001000001' = 40001   B = b'1010101011111001' = 43769    Obtained_Result = b'1101000010110110010010100111001' = 1750803769  Expected_Result = b'1101000010110110010010100111001' = 1750803769  Status = TRUE
A = b'1011001101011011' = 45915   B = b'1010011010101101' = 42669    Obtained_Result = b'1110100110001100011011001111111' = 1959147135  Expected_Result = b'1110100110001100011011001111111' = 1959147135  Status = TRUE
A = b'1110000010001100' = 57484   B = b'1011110001000010' = 48194    Obtained_Result = b'10100101001000001011010000011000' = 2770383896  Expected_Result = b'10100101001000001011010000011000' = 2770383896  Status = TRUE
A = b'11100010011101' = 14493   B = b'1000011111111001' = 34809    Obtained_Result = b'11110000010001110110111010101' = 504486837  Expected_Result = b'11110000010001110110111010101' = 504486837  Status = TRUE
A = b'10010000101010' = 9258   B = b'1011111101100' = 6124    Obtained_Result = b'11011000010001110010111000' = 56695992  Expected_Result = b'11011000010001110010111000' = 56695992  Status = TRUE
A = b'1011001100100010' = 45858   B = b'1001101011100100' = 39652    Obtained_Result = b'1101100011000011111111001001000' = 1818361416  Expected_Result = b'1101100011000011111111001001000' = 1818361416  Status = TRUE
A = b'1100010001111000' = 50296   B = b'1000110011111100' = 36092    Obtained_Result = b'1101100001100110000011000100000' = 1815283232  Expected_Result = b'1101100001100110000011000100000' = 1815283232  Status = TRUE
A = b'1101011111001' = 6905   B = b'110110010000100' = 27780    Obtained_Result = b'1011011011101111010001100100' = 191820900  Expected_Result = b'1011011011101111010001100100' = 191820900  Status = TRUE
A = b'1011001011111111' = 45823   B = b'110001010111101' = 25277    Obtained_Result = b'1000101000010011100010001000011' = 1158267971  Expected_Result = b'1000101000010011100010001000011' = 1158267971  Status = TRUE
A = b'10101100101011' = 11051   B = b'110110100010011' = 27923    Obtained_Result = b'10010011001001000001100110001' = 308577073  Expected_Result = b'10010011001001000001100110001' = 308577073  Status = TRUE
A = b'100111110011100' = 20380   B = b'1001100101110010' = 39282    Obtained_Result = b'101111101101111010111101111000' = 800567160  Expected_Result = b'101111101101111010111101111000' = 800567160  Status = TRUE
A = b'10001111111110' = 9214   B = b'101010010111011' = 21691    Obtained_Result = b'1011111010011010001010001010' = 199860874  Expected_Result = b'1011111010011010001010001010' = 199860874  Status = TRUE
A = b'111101110110011' = 31667   B = b'100000011111000' = 16632    Obtained_Result = b'11111011001001001010101101000' = 526685544  Expected_Result = b'11111011001001001010101101000' = 526685544  Status = TRUE
A = b'10001110100100' = 9124   B = b'100011100011010' = 18202    Obtained_Result = b'1001111001100001101010101000' = 166075048  Expected_Result = b'1001111001100001101010101000' = 166075048  Status = TRUE
A = b'110111001011010' = 28250   B = b'1101110110010110' = 56726    Obtained_Result = b'1011111110000100010110101010111100' = 1602509500  Expected_Result = b'1011111110000100010110101010111100' = 1602509500  Status = TRUE
A = b'1000001000000000' = 33280   B = b'11000010110001' = 12465    Obtained_Result = b'11000101110011110001000000000' = 414835200  Expected_Result = b'11000101110011110001000000000' = 414835200  Status = TRUE
A = b'110010010001111' = 25743   B = b'1000011111111000' = 34808    Obtained_Result = b'110101011010001101001110001000' = 896062344  Expected_Result = b'110101011010001101001110001000' = 896062344  Status = TRUE
A = b'1010011101011110' = 42926   B = b'1001100010010001' = 39057    Obtained_Result = b'1100011111011100100100110001110' = 1676560782  Expected_Result = b'1100011111011100100100110001110' = 1676560782  Status = TRUE
A = b'111001111101010' = 29674   B = b'1011101101110111' = 47991    Obtained_Result = b'1010100111000011100111111000110' = 1424084934  Expected_Result = b'1010100111000011100111111000110' = 1424084934  Status = TRUE
A = b'1010000110010110' = 41366   B = b'1110101101100101' = 60261    Obtained_Result = b'10010100100101000111001000101110' = 2492756526  Expected_Result = b'10010100100101000111001000101110' = 2492756526  Status = TRUE
A = b'1011110000001101' = 48141   B = b'111010010011001' = 29849    Obtained_Result = b'1010101101001100100011111000101' = 1436960709  Expected_Result = b'1010101101001100100011111000101' = 1436960709  Status = TRUE
A = b'1001011010110111' = 38583   B = b'111001100000100' = 29444    Obtained_Result = b'1000011101101101000111111011100' = 1136037852  Expected_Result = b'1000011101101101000111111011100' = 1136037852  Status = TRUE
A = b'1110001001110001' = 57969   B = b'11010111011011' = 13787    Obtained_Result = b'101111101000110001101110101011' = 799218603  Expected_Result = b'101111101000110001101110101011' = 799218603  Status = TRUE
A = b'1101001010101000' = 53928   B = b'11000011100010' = 12514    Obtained_Result = b'101000001110010111100001010000' = 674854992  Expected_Result = b'101000001110010111100001010000' = 674854992  Status = TRUE
A = b'1011011010001' = 5841   B = b'111101110100011' = 31651    Obtained_Result = b'1011000001001111001000010011' = 184873491  Expected_Result = b'1011000001001111001000010011' = 184873491  Status = TRUE
A = b'11110001111111' = 15487   B = b'10001111010100' = 9172    Obtained_Result = b'1000011101110111011000101100' = 142046764  Expected_Result = b'1000011101110111011000101100' = 142046764  Status = TRUE
A = b'1011111001' = 761   B = b'1011111001010100' = 48724    Obtained_Result = b'10001101011100011110110100' = 37078964  Expected_Result = b'10001101011100011110110100' = 37078964  Status = TRUE
A = b'1010100000001101' = 43021   B = b'1011110010011111' = 48287    Obtained_Result = b'1111011110100011110110000010011' = 2077355027  Expected_Result = b'1111011110100011110110000010011' = 2077355027  Status = TRUE
A = b'111110011101100' = 31980   B = b'1100010000011000' = 50200    Obtained_Result = b'1011111101100000110011000100000' = 1605396000  Expected_Result = b'1011111101100000110011000100000' = 1605396000  Status = TRUE
A = b'1100011100011000' = 51080   B = b'1100100011011100' = 51420    Obtained_Result = b'10011100100011011011100011100000' = 2626533600  Expected_Result = b'10011100100011011011100011100000' = 2626533600  Status = TRUE
A = b'111000001100000' = 28768   B = b'1111111' = 127    Obtained_Result = b'1101111011111110100000' = 3653536  Expected_Result = b'1101111011111110100000' = 3653536  Status = TRUE
A = b'1101001100110101' = 54069   B = b'1010000110111101' = 41405    Obtained_Result = b'10000101011100000100001100100001' = 2238726945  Expected_Result = b'10000101011100000100001100100001' = 2238726945  Status = TRUE
A = b'1110011110001001' = 59273   B = b'1011000110100110' = 45478    Obtained_Result = b'10100000101010111011011110101110' = 2695617494  Expected_Result = b'10100000101010111011011110101110' = 2695617494  Status = TRUE
A = b'110010110110111' = 26039   B = b'100000110011111' = 16799    Obtained_Result = b'11010000100101010001110101001' = 437429161  Expected_Result = b'11010000100101010001110101001' = 437429161  Status = TRUE
A = b'1010101100011010' = 43802   B = b'10010110111110' = 9662    Obtained_Result = b'11001001110011011111101001100' = 423214924  Expected_Result = b'11001001110011011111101001100' = 423214924  Status = TRUE
A = b'1111100111010' = 7994   B = b'101100101001010' = 22858    Obtained_Result = b'1010111001000011000011000100' = 182726852  Expected_Result = b'1010111001000011000011000100' = 182726852  Status = TRUE
A = b'1101010100111100' = 54588   B = b'1100011001101011' = 50795    Obtained_Result = b'10100101010001011000100000010100' = 2772797460  Expected_Result = b'10100101010001011000100000010100' = 2772797460  Status = TRUE
A = b'1000010110010101' = 34197   B = b'1100000101111000' = 49528    Obtained_Result = b'1100100111100111111001011011000' = 1693709016  Expected_Result = b'1100100111100111111001011011000' = 1693709016  Status = TRUE
A = b'101110110111010' = 23994   B = b'110001100001111' = 25359    Obtained_Result = b'100100010001000110101111100110' = 608463846  Expected_Result = b'100100010001000110101111100110' = 608463846  Status = TRUE
A = b'100100100001111' = 18703   B = b'110010001110111' = 25719    Obtained_Result = b'11100101010111101000111111001' = 481022457  Expected_Result = b'11100101010111101000111111001' = 481022457  Status = TRUE
A = b'1110100100001001' = 59657   B = b'100111000010' = 2498    Obtained_Result = b'1000111000011110100111010010' = 149023186  Expected_Result = b'1000111000011110100111010010' = 149023186  Status = TRUE
A = b'1010011010111000' = 42680   B = b'1101010010101000' = 54440    Obtained_Result = b'10001010011111011100100011000000' = 2323499200  Expected_Result = b'10001010011111011100100011000000' = 2323499200  Status = TRUE
A = b'10111001111100' = 11900   B = b'1101100001110110' = 55414    Obtained_Result = b'100111010011100000110100101000' = 659426600  Expected_Result = b'100111010011100000110100101000' = 659426600  Status = TRUE
A = b'1001111011000000' = 40640   B = b'1101111101111101' = 57213    Obtained_Result = b'10001010100101011011000011110000000' = 2325136320  Expected_Result = b'10001010100101011011000011110000000' = 2325136320  Status = TRUE
A = b'1100010111000000' = 50624   B = b'11100110000011' = 14723    Obtained_Result = b'101100011011001111000101000000' = 745337152  Expected_Result = b'101100011011001111000101000000' = 745337152  Status = TRUE
A = b'11101101111010' = 15226   B = b'11000011101010' = 12522    Obtained_Result = b'1011010111010011110110000100' = 190659972  Expected_Result = b'1011010111010011110110000100' = 190659972  Status = TRUE
A = b'1011010110010' = 5810   B = b'1000010010011010' = 33946    Obtained_Result = b'1011110000010110111100010100' = 197226260  Expected_Result = b'1011110000010110111100010100' = 197226260  Status = TRUE
A = b'101110011001000' = 23752   B = b'110101101011011' = 27483    Obtained_Result = b'100110111010001001001100011000' = 652776216  Expected_Result = b'100110111010001001001100011000' = 652776216  Status = TRUE
A = b'110001111100101' = 25573   B = b'111001111011010' = 29658    Obtained_Result = b'101101001101001111000000000010' = 758444034  Expected_Result = b'101101001101001111000000000010' = 758444034  Status = TRUE
A = b'11000110000010' = 12674   B = b'100110011101010' = 19690    Obtained_Result = b'1110110111111011000110101000' = 249551060  Expected_Result = b'1110110111111011000110101000' = 249551060  Status = TRUE
A = b'1101011110000000' = 55168   B = b'100101110110010' = 19378    Obtained_Result = b'111111101110000101011100000000' = 1069045504  Expected_Result = b'111111101110000101011100000000' = 1069045504  Status = TRUE
A = b'10000110011111' = 8607   B = b'111111000011111' = 32287    Obtained_Result = b'10000100100000101010001000001' = 277894209  Expected_Result = b'10000100100000101010001000001' = 277894209  Status = TRUE
A = b'1001000100010000' = 37136   B = b'1010001000110111' = 41527    Obtained_Result = b'1011011111010110100101001110000' = 1542146672  Expected_Result = b'1011011111010110100101001110000' = 1542146672  Status = TRUE
A = b'110110001101101' = 27757   B = b'110001101101000' = 25448    Obtained_Result = b'101010000110100011001101001000' = 706360136  Expected_Result = b'101010000110100011001101001000' = 706360136  Status = TRUE
A = b'1111111101000001' = 65345   B = b'11001011100110' = 13030    Obtained_Result = b'110010110000000000011001100110' = 851445350  Expected_Result = b'110010110000000000011001100110' = 851445350  Status = TRUE
A = b'1110111111010000' = 61392   B = b'111010111101100' = 30188    Obtained_Result = b'1101110011101110010001111000000' = 1853301696  Expected_Result = b'1101110011101110010001111000000' = 1853301696  Status = TRUE
A = b'10001011' = 139   B = b'101100110010110' = 22934    Obtained_Result = b'1100001010010001110010' = 3187826  Expected_Result = b'1100001010010001110010' = 3187826  Status = TRUE
A = b'1100110110000000' = 52608   B = b'110101000111010' = 27194    Obtained_Result = b'1010101010001011000111100000000' = 1430621952  Expected_Result = b'1010101010001011000111100000000' = 1430621952  Status = TRUE
A = b'111101110100010' = 31650   B = b'1010001011101' = 5213    Obtained_Result = b'1001110101011001000111011010' = 164991450  Expected_Result = b'1001110101011001000111011010' = 164991450  Status = TRUE
A = b'110010110110111' = 26039   B = b'1111110110110011' = 64947    Obtained_Result = b'1100100110011001111100111110101' = 1691154933  Expected_Result = b'1100100110011001111100111110101' = 1691154933  Status = TRUE
A = b'110011010000001' = 26241   B = b'1111111110011001' = 65433    Obtained_Result = b'1100110010101111100001000011001' = 1717027353  Expected_Result = b'1100110010101111100001000011001' = 1717027353  Status = TRUE
A = b'1010101011000101' = 43717   B = b'100010000001111' = 17423    Obtained_Result = b'101101011001100101010110001011' = 761681291  Expected_Result = b'101101011001100101010110001011' = 761681291  Status = TRUE
A = b'101001110011111' = 21407   B = b'1000101000111011' = 35387    Obtained_Result = b'101101001001101111101110100101' = 757529509  Expected_Result = b'101101001001101111101110100101' = 757529509  Status = TRUE
```

Simulation Waveform

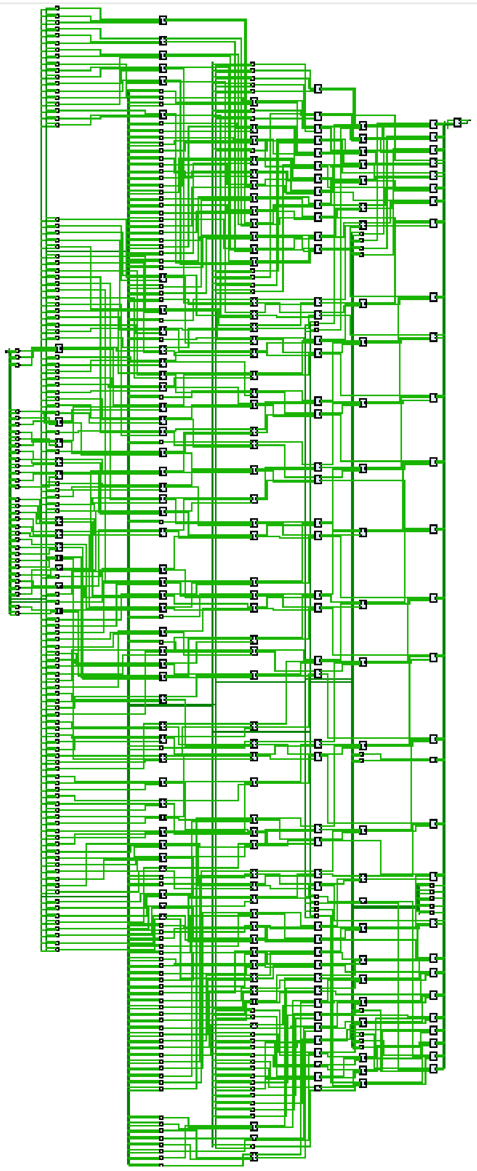
The simulation results are also depicted in the waveform. The design is, once again, verified. A portion of the waveform is given below.



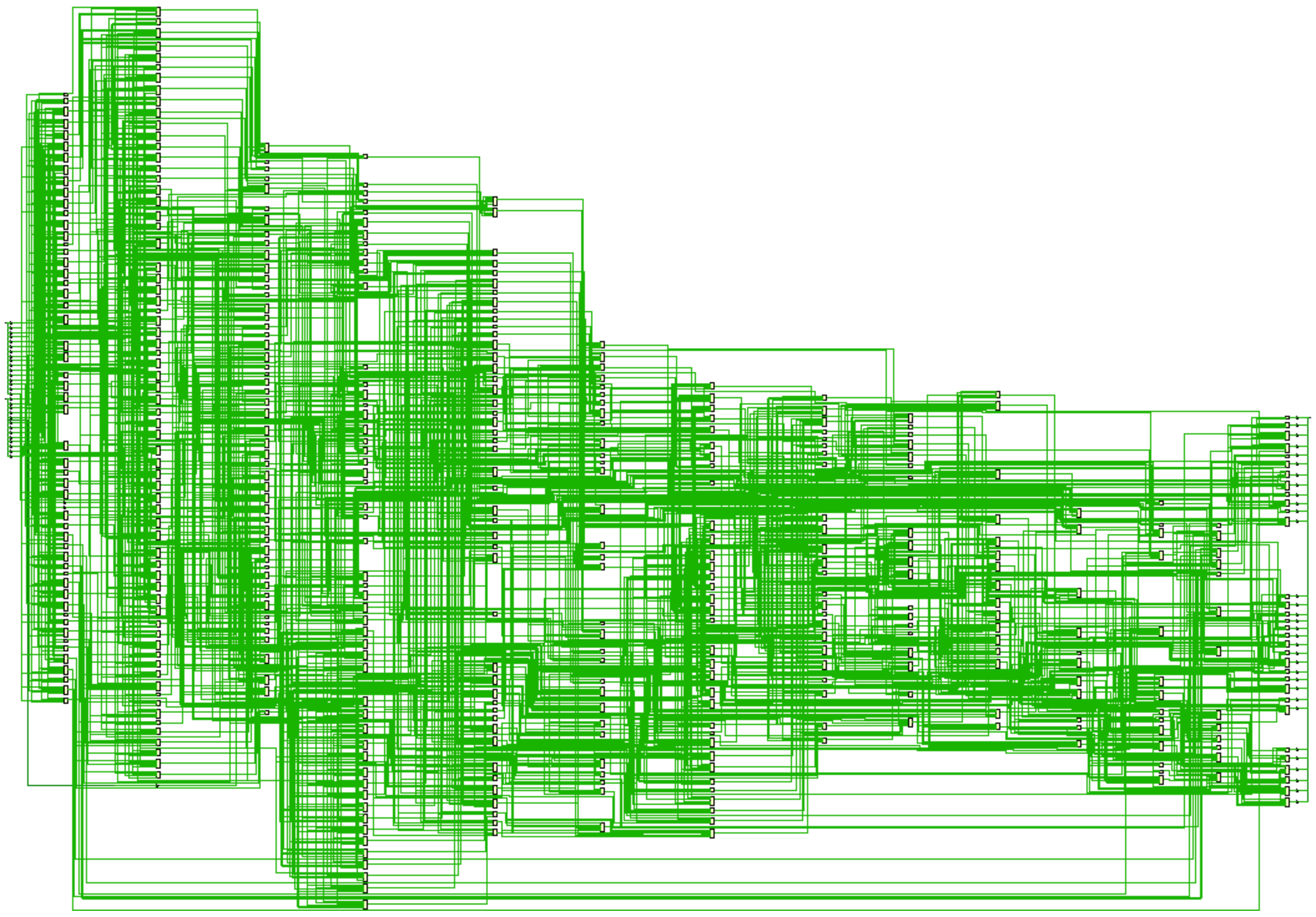
Schematics

RTL Schematic

Armin Asgharifard, 040190912
Emrecan Yiğit, 040190203



Technology Schematic

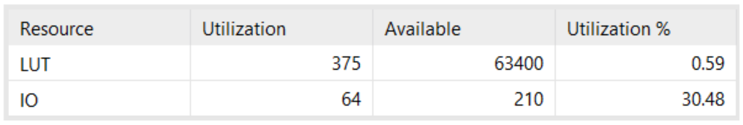


Implementation

After running the implementation step, we look at different aspects of our design.

Resource Utilization

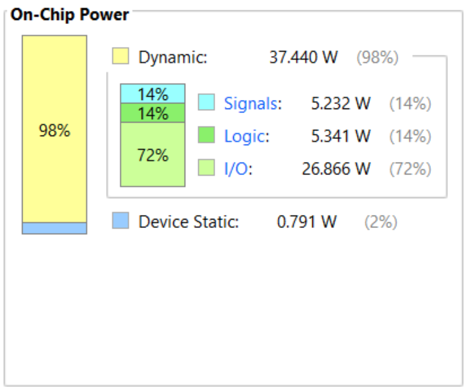
A total of 375 LUTs are utilized in the design.



Resource	Utilization	Available	Utilization %
LUT	375	63400	0.59
IO	64	210	30.48

Average Power Consumption

According to the report, the total on-chip power consumption is 38.23 W, of which the majority belongs to dynamic power consumption, which is 37.44 W.



Dynamic:	37.440 W	(98%)
Signals:	5.232 W	(14%)
Logic:	5.341 W	(14%)
I/O:	26.866 W	(72%)
Device Static:	0.791 W	(2%)

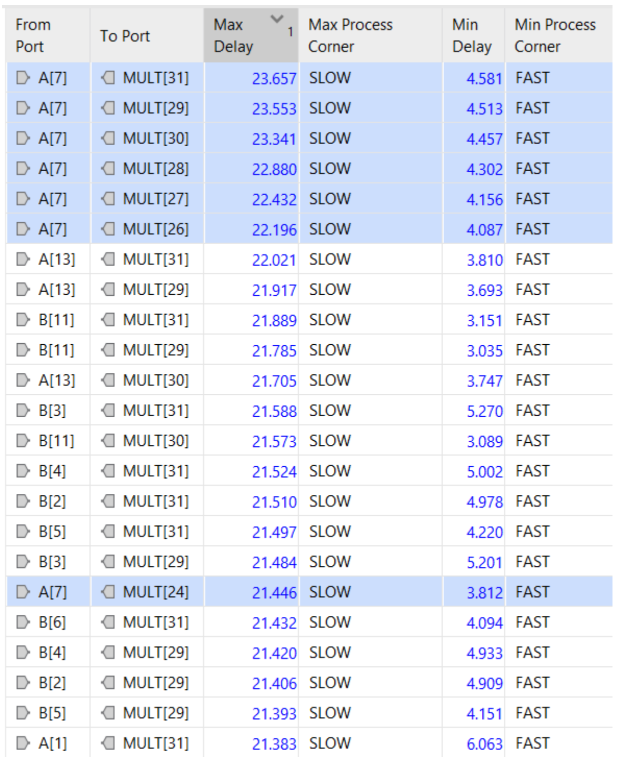
Combinational Path Delays

Combinational path delays datasheet is also generated as a post-implementation report. Since the table is too long, only the top portion of it is included in the following figure.

Critical path is the path that takes the longest time from starting point to ending point. Intuitively, we can assume those bits that are inputs to an adder at each stage, or in other words, those bits that does not remain unchanged at each stage transition, will take the longest time to reach to the last stage.

At each stage, adders are concentrated at the middle regions of the stage diagram. So, by looking at the first stage diagram, we can guess that bits in the middle rows and middle columns may be what we are looking for. If we take the eighth row, for example, it corresponds to ANDing B bits with A[7]. Therefore, one of the paths starting from A[7] can be the critical path.

By looking at the path delays report below, it indeed matches with our guess.

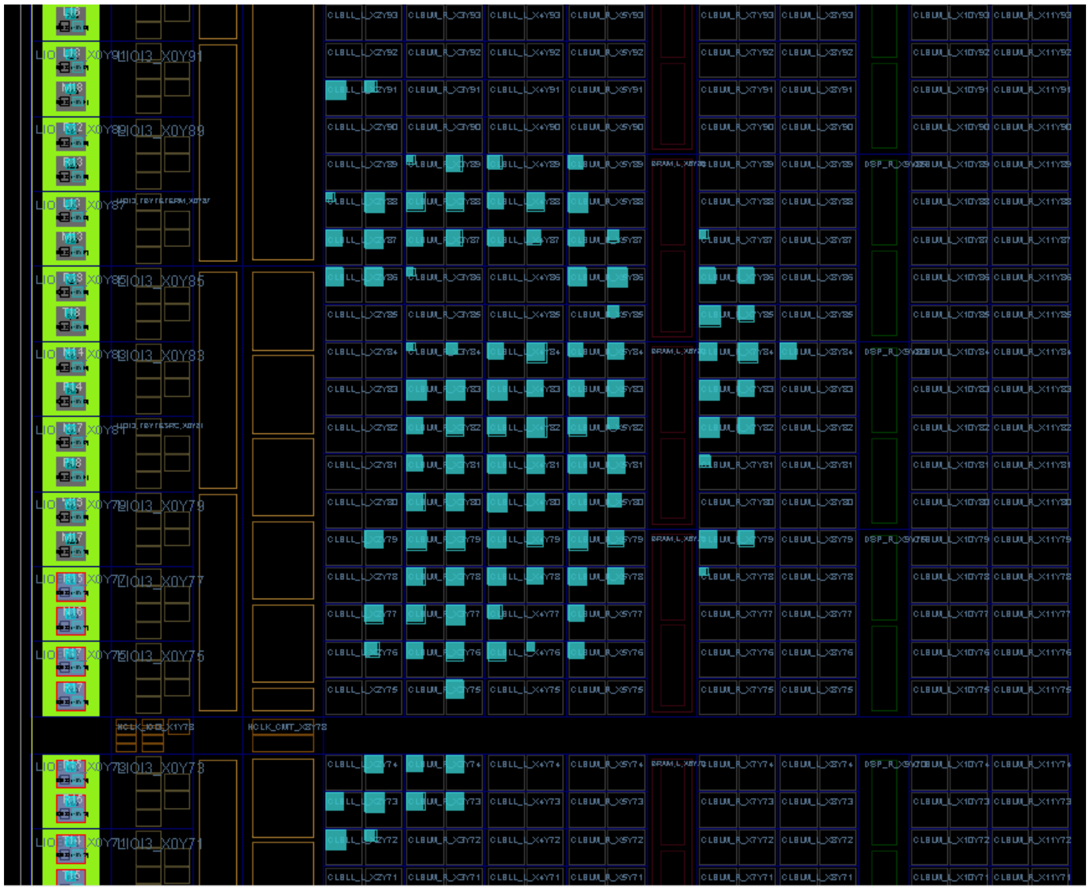


From Port	To Port	Max Delay	Max Process Corner	Min Delay	Min Process Corner
A[7]	MULT[31]	23.657	SLOW	4.581	FAST
A[7]	MULT[29]	23.553	SLOW	4.513	FAST
A[7]	MULT[30]	23.341	SLOW	4.457	FAST
A[7]	MULT[28]	22.880	SLOW	4.302	FAST
A[7]	MULT[27]	22.432	SLOW	4.156	FAST
A[7]	MULT[26]	22.196	SLOW	4.087	FAST
A[13]	MULT[31]	22.021	SLOW	3.810	FAST
A[13]	MULT[29]	21.917	SLOW	3.693	FAST
B[11]	MULT[31]	21.889	SLOW	3.151	FAST
B[11]	MULT[29]	21.785	SLOW	3.035	FAST
A[13]	MULT[30]	21.705	SLOW	3.747	FAST
B[3]	MULT[31]	21.588	SLOW	5.270	FAST
B[11]	MULT[30]	21.573	SLOW	3.089	FAST
B[4]	MULT[31]	21.524	SLOW	5.002	FAST
B[2]	MULT[31]	21.510	SLOW	4.978	FAST
B[5]	MULT[31]	21.497	SLOW	4.220	FAST
B[3]	MULT[29]	21.484	SLOW	5.201	FAST
A[7]	MULT[24]	21.446	SLOW	3.812	FAST
B[6]	MULT[31]	21.432	SLOW	4.094	FAST
B[4]	MULT[29]	21.420	SLOW	4.933	FAST
B[2]	MULT[29]	21.406	SLOW	4.909	FAST
B[5]	MULT[29]	21.393	SLOW	4.151	FAST
A[1]	MULT[31]	21.383	SLOW	6.063	FAST

The longest delay is the path from A[7] to MULT[31], which is 23.657 ns.

Device Layout

In the picture below, a zoomed device layout is shown, on which how LUTs are distributed on the chip can be seen.



Conclusion

Dadda Multiplier is one of the fastest design architectures that are presented for multiplication purposes. To successfully achieve an FPGA design of the Dadda multiplier, several tasks were performed by group members in the following way:

- Armin and Emrecan did research about and learned Dadda's tree algorithm to be used in multiplication and addition.
- Emrecan drew stage diagrams for the desired 16-bit Dadda multiplier design.
- Armin made corrections and optimizations in the diagrams.
- Armin came up with the solution to write the Verilog design code in the most efficient manner.
- Emrecan wrote the Verilog design code.
- Emrecan, also, wrote the Python code to generate random numbers to be stored in a stimulus file.
- Armin wrote the testbench code to read the stimulus file, simulate the design, and write the results in a new text file.
- Armin prepared the report of the project.

References

- *B. Parhami, Computer arithmetic - algorithms and hardware designs, Oxford University Press, (2010)*
- *https://www.javatpoint.com/verilog-file-operations*