Lab 2 report

Name: Oscar Stark Student ID: F11015127

In this lab I implemented a 4x4 adder using 4 full adders and a 4x4 multiplier using 3 4x4 adders.

4x4 adder

4x4 adder code

```
module fourbit_adder(
    input [3:0] X,
    input [3:0] Y,
    input C0,
    output [3:0] S,
    output C4
    );

    //3 wires
    wire C1, C2, C3;

    //4 full adders
    full_adder FA0 (.X(X[0]), .Y(Y[0]), .C_IN(C0), .C_OUT(C1), .S(S[0]));
    full_adder FA1 (.X(X[1]), .Y(Y[1]), .C_IN(C1), .C_OUT(C2), .S(S[1]));
    full_adder FA2 (.X(X[2]), .Y(Y[2]), .C_IN(C2), .C_OUT(C3), .S(S[2]));
    full_adder FA3 (.X(X[3]), .Y(Y[3]), .C_IN(C3), .C_OUT(C4), .S(S[3]));
endmodule
```

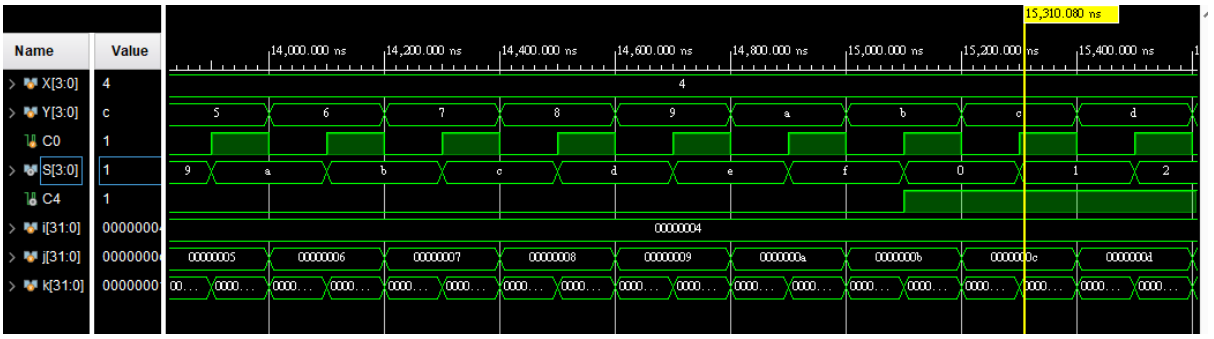
X and Y are the bits being added, C0 is the carry input, C4 is the carry output and S is the sum. 4 full adders are used.

Test bench

```
module fourbit_adder_sim;
    reg [3:0] X;
    reg [3:0] Y;
    reg C0;
    wire [3:0] S;
    wire C4;

    //Unit under testing
    fourbit_adder uut(.X(X[3:0]), .Y(Y[3:0]), .C0(C0), .S(S[3:0]), .C4(C4));

    //For loops used to test all posible inputs
    integer i, j, k;
    initial begin
        for (i=0; i<16; i=i+1)
        begin
            X=i;
            for (j=0; j<16; j=j+1)
            begin
                Y=j;
                for (k=0; k<2; k=k+1)
                begin
                    C0=k;
                    #100;
                end
            end
        end
    end
endmodule
```

All possible inputs are simulated.

Simulation results



X, Y and C0 are the inputs. S and C4 are the outputs.

4x4 multiplier

4x4 multiplier code

```
module fourbitmultiplier(
    input [3:0] X,
    input [3:0] Y,
    output [7:0] P
    );

    //array with 16 and gates
    wire [15:0] A;
     and(A[0], X[0], Y[0]);
     and(A[1], X[1], Y[0]);
     and(A[2], X[2], Y[0]);
     and(A[3], X[3], Y[0]);
     and(A[4], X[0], Y[1]);
     and(A[5], X[1], Y[1]);
     and(A[6], X[2], Y[1]);
     and(A[7], X[3], Y[1]);
     and(A[8], X[0], Y[2]);
     and(A[9], X[1], Y[2]);
     and(A[10], X[2], Y[2]);
     and(A[11], X[3], Y[2]);
     and(A[12], X[0], Y[3]);
     and(A[13], X[1], Y[3]);
     and(A[14], X[2], Y[3]);
     and(A[15], X[3], Y[3]);
```

```
    //arrays with the input for the first FBA
    wire [3:0] X1;
    assign X1[2:0] = A[3:1];
    assign X1[3]=0;
    wire [3:0] Y1;
    assign Y1[3:0] = A[7:4];
    wire [3:0] S1;
    wire C_OUT1;
    //first FBA
    fourbit_adder FBA1 (.X(X1[3:0]), .Y(Y1[3:0]), .C0(0), .C4(C_OUT1), .S(S1));

    //arrays with the input for the second FBA
    wire [3:0] X2;
    assign X2[2:0] = S1[3:1];
    assign X2[3] = C_OUT1;
    wire [3:0] Y2;
    assign Y2[3:0] = A[11:8];
    wire [3:0] S2;
    wire C_OUT2;
    //second FBA
    fourbit_adder FBA2 (.X(X2[3:0]), .Y(Y2[3:0]), .C0(0), .C4(C_OUT2), .S(S2));

    //arrays with the input for the third FBA
    wire [3:0] X3;
    assign X3[2:0] = S2[3:1];
    assign X3[3] = C_OUT2;
    wire [3:0] Y3;
    assign Y3[3:0] = A[15:12];
    wire [3:0] S3;
    wire C_OUT3;
    //third FBA
    fourbit_adder FBA3 (.X(X3[3:0]), .Y(Y3[3:0]), .C0(0), .C4(C_OUT3), .S(S3));

    //P is an array with the result of the multiplication
    wire [7:0] P;
    assign P[0] = A[0];
    assign P[1] = S1[0];
    assign P[2] = S2[0];
    assign P[6:3] = S3[3:0];
    assign P[7] = C_OUT3;
endmodule
```

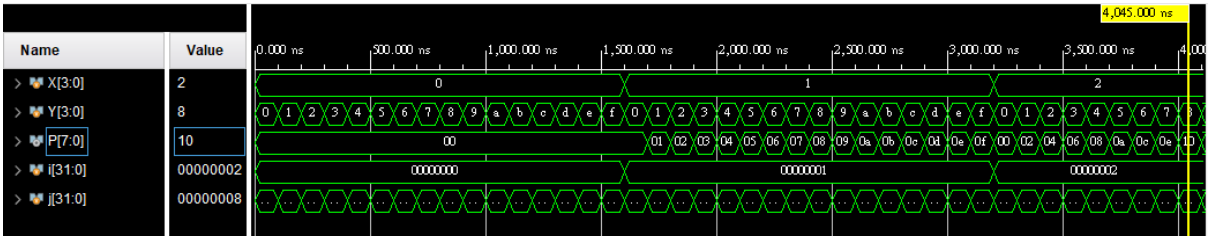
X and Y are the factors, P is the product.The results of 16 and gates are stored in the array A. 3 four bit adders are used.

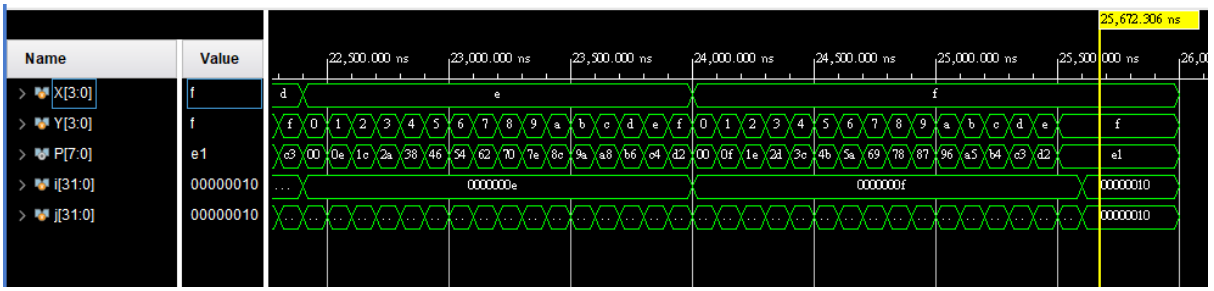
Test bench

```
module fourbitmultiplier_sim;
    reg [3:0] X;
    reg [3:0] Y;
    wire [7:0] P;

    //Unit under test
    fourbitmultiplier uut(.X(X[3:0]), .Y(Y[3:0]), .P(P[7:0]));

    //For loops used to test all posible inputs
    integer i, j;
    initial begin
        for (i=0; i<16; i=i+1)
        begin
            X=i;
            for (j=0; j<16; j=j+1)
            begin
                Y=j;
                #100;
            end
        end
    end
endmodule
```

All possible inputs are simulated.

Simulation results





X, Y are the inputs. P is the output.