|  |
| --- |
| **DIGITAL SYSTEM DESIGN LABORATORY** |
| **LAB 1** |

**IMPLEMENTATION OF BASIC COMBINATION LOGIC CIRCUIT USING VERILOG**

### I. LAB OBJECTIVES

### This Lab experiments are intended to implement Basic Combination Logic and Sequential Circuit in Verilog. Students are require to write test bench to simulate the given example code and Top level module to implement these codes in DE2-115 FPGA Kit.

### a) For each experiment write the Verilog Code in three method ( dataflow, behavior and gate level)

### b) For each model, write the Verilog Code to implement the these modules in DE2-FPGA Kit

### (Show implementation results in Lab report)

### c) For each model, Write the testbench to simulate this module.

### (Do this task at home as a Home work 2 and show simulation results in Lab report)

### d) Analyze the FPGA implementation results and simulation results for these model

### II. PROCEDURE

### II.1 LAB EXPERIMENT 1 : WRITE HDL CODE TO REALIZE ALL LOGIC GATES

##### Refer to the tutorial “Quartus II Introduction Using Verilog Design” to create the first project in Quartus II. implement a simple circuit in DE2-115 Kit.



Referenced Verilog codes in 3 methods ( Dataflow, behavior, gate level):

Dataflow model:

module allgate\_DF ( a, b, yand,yor,ynot,ynand,ynor,yxor,yxnor );

input a,b;

output yand, yor, ynot, ynand, ynor, yxor, yxnor;

assign yand = a & b; // AND Operation

assign yor = a | b; // OR Operation

assign ynot = ~a ; // NOT Operation

assign ynand = ~(a & b); // NAND Operation

assign ynor = ~(a | b); //NOR Operation

assign yxor = a ^ b; //XOR Operation

assign yxnor =~(a^b); //XNOR Operation

endmodule // END of the module

### Write the Top-level verilog module to implement the allgate\_DF module in DE2-FPGA Kit

module lab1\_ex1(SW,LEDG,LEDR);

input[17:0] SW;

output[7:0] LEDG;

output[17:0] LEDR;

assign LEDR=SW;

allgate\_DF DUT(SW[1],SW[2],LEDG[5],LEDG[4],LEDG[3],LEDG[2],LEDG[1],LEDG[0]);

endmodule

Behavior Model :

module allgate\_BH ( a, b, yand,yor,ynot,ynand,ynor,yxor,yxnor );

input a,b;

output yand, yor, ynot, ynand, ynor, yxor, yxnor;

reg yand, yor, ynot, ynand, ynor, yxor, yxnor;

always @(\*)

begin

assign yand = a & b; // AND Operation

assign yor = a | b; // OR Operation

assign ynot = ~a ; // NOT Operation

assign ynand = ~(a & b); // NAND Operation

assign ynor = ~(a | b); //NOR Operation

assign yxor = a ^ b; //XOR Operation

assign yxnor =~(a^b); //XNOR Operation

end

endmodule // END of the module

### Write the Top-level verilog module to implement the allgate\_BH module in DE2-FPGA Kit

module lab1\_ex1(SW,LEDG,LEDR);

input[17:0] SW;

output[7:0] LEDG;

output[17:0] LEDR;

assign LEDR=SW;

allgate\_BH DUT(SW[1],SW[2],LEDG[5],LEDG[4],LEDG[3],LEDG[2],LEDG[1],LEDG[0]);

endmodule

Gate-level (Structural Model) :

module allgate\_GL ( a, b, yand,yor,ynot,ynand,ynor,yxor,yxnor );

input a,b;

output yand, yor, ynot, ynand, ynor, yxor, yxnor;

and G1(yand,a,b); // AND Operation

or G2(yor,a, b); // OR Operation

not G3(ynot,a) ; // NOT Operation

nand G4 (ynand,a,b); // NAND Operation

nor G5(ynor,a,b); //NOR Operation

xor G6(yxor,a,b); //XOR Operation

xnor G7(yxnor,a,b); //XNOR Operation

endmodule // END of the module

### Write the Top-level verilog module to implement the allgate\_GL module in DE2-FPGA Kit

module lab1\_ex1(SW,LEDG,LEDR);

input[17:0] SW;

output[7:0] LEDG;

output[17:0] LEDR;

assign LEDR=SW;

allgate\_GL DUT(SW[1],SW[2],LEDG[5],LEDG[4],LEDG[3],LEDG[2],LEDG[1],LEDG[0]);

endmodule

**II.2 LAB EXPERIMENT 2 : WRITE VERILOG HDL CODES TO SIMULATE AND IMPLEMENT THE HALF ADDER CIRCUIT:**

A black line drawing of a circuit

Description automatically generated A table with numbers and symbols

Description automatically generated

Dataflow model:

module half\_adder\_dataflow(input a, b, output s, Cout);

assign s = a ^ b;

assign Cout = a & b;

endmodule

Behavior Model :

module half\_adder\_behavior(sum,carry,a,b );

output sum,carry;

input a,b;

reg sum,carry;

always @(a,b)

begin

sum <= a ^ b;

carry <= a&b ;

end

endmodule

Gate-level (Structural Model) :

module half\_adder\_structeral(input a, b, output s, Cout);

xor G1(s,a,b);

and G2(Cout,a,b);

endmodule

**II.3 EXPERIMENT 3: WRITE VERILOG HDL CODES TO SIMULATE AND IMPLEMENT THE FULL ADDER CIRCUIT:**

**A diagram of a circuit

Description automatically generated A table of input output

Description automatically generated**

Dataflow model:

module full\_adder\_DF(input a, b, cin, output S, Cout);

assign S = a ^ b ^ cin;

assign Cout = (a & b) | (b & cin) | (a & cin);

endmodule

Behavior Model using case :

full\_adder\_BH1 (input wire A, B, Cin, output reg S, output reg Cout);

always @(A or B or Cin)

begin

case (A | B | Cin)

3'b000: begin S = 0; Cout = 0; end

3'b001: begin S = 1; Cout = 0; end

3'b010: begin S = 1; Cout = 0; end

3'b011: begin S = 0; Cout = 1; end

3'b100: begin S = 1; Cout = 0; end

3'b101: begin S = 0; Cout = 1; end

3'b110: begin S = 0; Cout = 1; end

3'b111: begin S = 1; Cout = 1; end

endcase

end

Behavior Model using if else :

module full\_adder( A, B, Cin, S, Cout);

input wire A, B, Cin;

output reg S, Cout;

always @(A or B or Cin)

begin

if(A==0 && B==0 && Cin==0)

begin

S=0;

Cout=0;

end

else if(A==0 && B==0 && Cin==1)

begin

S=1;

Cout=0;

end

else if(A==0 && B==1 && Cin==0)

begin

S=1;

Cout=0;

end

else if(A==0 && B==1 && Cin==1)

begin

S=0;

Cout=1;

end

else if(A==1 && B==0 && Cin==0)

begin

S=1;

Cout=0;

end

else if(A==1 && B==0 && Cin==1)

begin

S=0;

Cout=1;

end

else if(A==1 && B==1 && Cin==0)

begin

S=0;

Cout=1;

end

else if(A==1 && B==1 && Cin==1)

begin

S=1;

Cout=1;

end

end

endmodule

endmodule

module full\_adder\_BH(a,b,c,sum,carry);

output sum,carry;

input a,b,c;

reg sum,carry;

always @ (a,b,c)

begin

sum <= a^ b^c;

carry <=(a&b) | (b&c) | (c&a);

end

endmodule

Gate-level (Structural Model) :

module full\_adder\_STRU(a,b,cin,Cum,Carry);

output Sum,Carry;

input A,B,Cin;

wire x,y,z;

xor g1(x,A,B);

xor g2(Sum,x,Cin);

and g3(y,x,Cin);

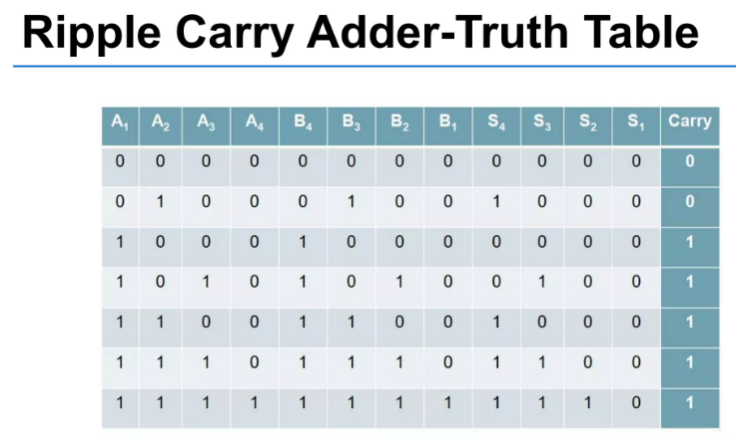
and g4(z,A,B);

or g5(Carry,x,y);

endmodule

**II.4 EXPERIMENT 4 : WRITE VERILOG HDL CODES TO SIMULATE AND IMPLEMENT THE RIPPLE CARRY ADDER CIRCUIT:**

**A diagram of a circuit

Description automatically generated**

A diagram of a computer component

Description automatically generated

Structural Model:

module full\_adder\_STRU(a,b,cin,Cum,Carry);

output Sum,Carry;

input A,B,Cin;

wire x,y,z;

xor g1(x,A,B);

xor g2(Sum,x,Cin);

and g3(y,x,Cin);

and g4(z,A,B);

or g5(Carry,x,y);

endmodule

module four\_bit\_adder\_STRU(cin,a,b,s,cout);

input [3:0] a,b;

input cin;

output [3:0] s;

output cout;

wire [2:0] w\_carry;

full\_adder\_STRU C1(a[0],b[0],cin,s[0], w\_carry[0]);

full\_adder\_STRU C2(a[1],b[1], w\_carry[0],s[1], w\_carry[1]);

full\_adder\_STRU C3(a[2],b[2], w\_carry[1],s[2], w\_carry[2]);

full\_adder\_STRU C4(a[3],b[3], w\_carry[2],s[3], cout);

endmodule

Dataflow model:

module Four\_bit\_Adder\_DF(A,B,Cin,Sum,Cout);

input [3:0] A,B;

input Cin;

output wire [3:0]Sum;

output wire Cout;

wire [4:0]temp;

assign temp=A+B+Cin;

assign Sum=temp[3:0];

assign Cout=temp[4];

endmodule

**II.5 EXPERIMENT 5 :WRITE VERILOG HDL CODES TO SIMULATE AND IMPLEMENT 2:1 MULTIPLEXER CIRCUIT:**

A diagram of a network

Description automatically generated A square with black text

Description automatically generated with medium confidence

Dataflow model:

module mux21\_data\_flow(i0,i1,sel,y);

input i0,i1,sel;

output y;

assign y =(i0&(~sel))|(i1&sel);

endmodule

Behavior Model :

module mux21\_Behavioural (i0,i1,sel,y);

input i0,i1,sel;

output y;

reg y;

always@(\*)

begin

if(sel==0) y=i0;

if(sel==1)y=i1;

end

endmodule

Structural Model:

module mux21\_structural(i0,i1,sel,y);

input i0,i1,sel;

output y;

wire net1,net2,net3;

not g1(net1,sel);

and g2(net2,i1,sel);

and g3(net3,i0,net1);

or g4(y,net3,net2);

endmodule

**II.6 EXPERIMENT 6 : WRITE VERILOG HDL CODES TO SIMULATE AND IMPLEMENT 4:1 MULTIPLEXER CIRCUIT:**

A diagram of a circuit

Description automatically generated A diagram of a diagram and a diagram of a diagram

Description automatically generated

Structural Model:

module mux41\_strutural (i0,i1,i2,i3,s0,s1,y);

input i0,i1,i2,i3,s0,s1;

output y;

wire n\_s0,n\_s1, a0,a1,a2,a3;

not g0(n\_s0,s0);

not g1(n\_s1,s1);

and g2(a0,i0,n\_s0,n\_s1);

and g3(a1,i1,n\_s1,s0);

and g4(a2,i2,s1,n\_s0);

and g5(a3,i3,s1,s0);

or g6(y,a0,a1,a2,a3);

endmodule

Dataflow model:

module mux41\_df (i0,i1,i2,i3,s0,s1,y);

input i0,i1,i2,i3,s0,s1;

output y;

assign y= i0&(~s1)&(~s0) | i1 &(~s1)&s0 | i2&s1&(~s0) | i3&s1&s0;

endmodule

module mux41beh\_v1(in,s,y );

output y ;

input [3:0] in ;

input [1:0] s ;

reg y;

always @ (in,s)

begin

if (s[0]==0&s[1]==0)

y <= in[0];

else if (s[0]==0&s[1]==1)

y <= in[1];

else if (s[0]==1&s[1]==0)

y <= in[2];

else

y <= in[3];

end

endmodule

Behavior Model :

module mux41beh\_v2(in,s,y );

output y ;

input [3:0]in ;

input [1:0]s ;

reg y;

always@(in,s)

begin

case ({s[1],s[0]})

2'b00: y <= in[0];

2'b01: y <= in[1];

2'b10: y <= in[2];

2'b11: y <= in[3];

endcase

end

endmodule

**IV. LAB REPORT GUIDELINES**

Students write up a report on the Verilog HDL implementation experiment projects created in this lab. The lab report should include Circuit Schematics, Truth Table, Verilog Module Codes, Verilog test bench codes, Top level module to implement the required circuit in FPGA KIT and evidences of data output evidences to validate the experiments (The Captured Screens, Photo of FPGA Kit implementation results).