|  |
| --- |
| **DIGITAL SYSTEM DESIGN LABORATORY** |
| **REPORT LAB 1\_Extra** |

**IMPLEMENTATION OF COMBINATION LOGIC CIRCUIT (2) USING VERILOG IN FPGA KIT**

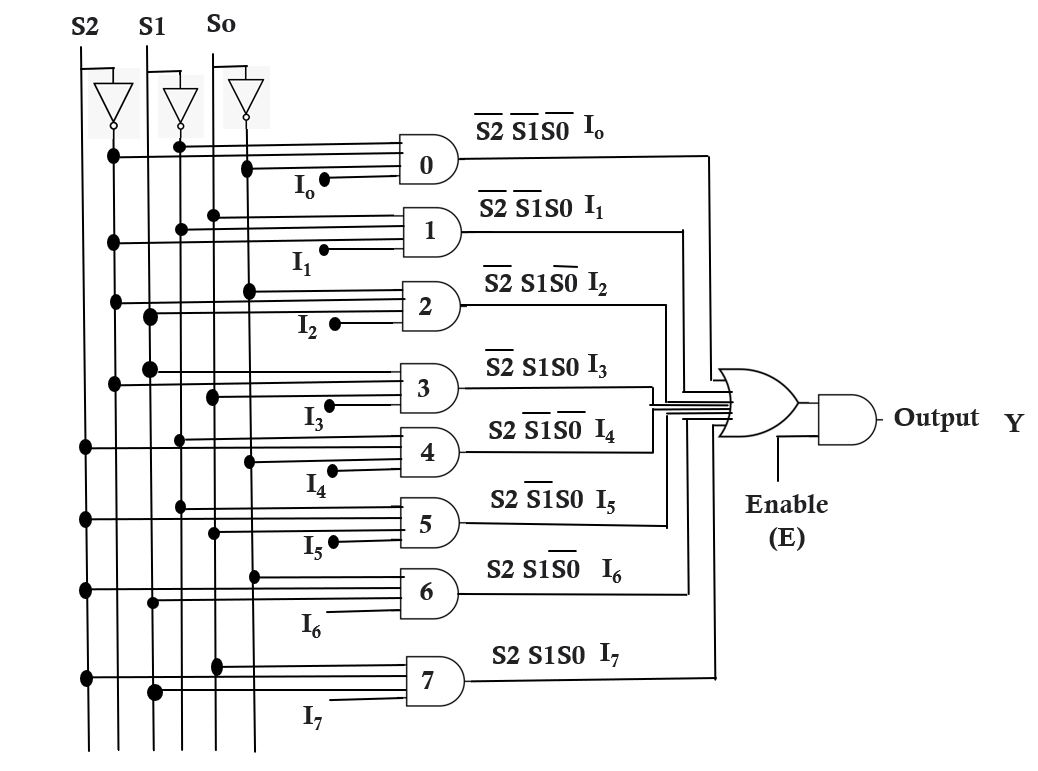
Phạm Quang Vinh ITITIU21347

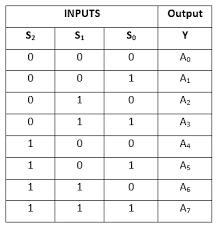
### I. LAB OBJECTIVES

### This Lab experiments are intended to implement Combination Logic Circuits (2) 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.

### II. LAB EXPERIMENT EXERCISES

**AIM: WRITE VERILOG HDL CODES TO SIMULATE AND IMPLEMENT THE FOLLOWING DIGITAL SEQUENTIAL LOGIC CIRCUITS:**

1. **Multiplexer 8 to 1**

**Schematic**

**Truth Table**

**Verilog Code:**

module mux8to1(input [7:0] data\_in, // 8-bit input data

input [2:0] sel, // Selection lines

output reg data\_out); // Output data

always @(\*) begin

case(sel)

3'b000: data\_out = data\_in[0];

3'b001: data\_out = data\_in[1];

3'b010: data\_out = data\_in[2];

3'b011: data\_out = data\_in[3];

3'b100: data\_out = data\_in[4];

3'b101: data\_out = data\_in[5];

3'b110: data\_out = data\_in[6];

3'b111: data\_out = data\_in[7];

default: data\_out = 8'b0; // Default case

endcase

end

endmodule

**Test bench Code:**

`timescale 1ns / 1ps

module testbench\_mux8to1;

// Parameters

parameter CLK\_PERIOD = 10; // Clock period in nanoseconds

// Inputs

reg [7:0] data\_in;

reg [2:0] sel;

reg clk;

// Outputs

wire data\_out;

// Instantiate the mux8to1 module

mux8to1 uut (

.data\_in(data\_in),

.sel(sel),

.data\_out(data\_out)

);

// Clock generation

always #((CLK\_PERIOD/2)) clk = ~clk;

// Test stimulus

initial begin

// Initialize inputs

data\_in = 8'b00000000;

sel = 3'b000;

clk = 0;

// Apply test vectors

#20 data\_in = 8'b10101010; // Input data pattern

#10 sel = 3'b001; // Select input 1

#10 sel = 3'b010; // Select input 2

#10 sel = 3'b011; // Select input 3

#10 sel = 3'b100; // Select input 4

#10 sel = 3'b101; // Select input 5

#10 sel = 3'b110; // Select input 6

#10 sel = 3'b111; // Select input 7

// End simulation

#10 $finish;

end

// Display output

always @(posedge clk) begin

$display("Time: %t, Selected Input: %b, Output: %b", $time, sel, data\_out);

end

endmodule

**Result :**

A screenshot of a computer program

Description automatically generated

**Top-level Verilog Code :**

module Lab1\_task1(

input [17:0] SW,

output [7:0] LEDG,

output [17:0] LEDR

);

wire [7:0] mux\_output; // Output of the 8-to-1 multiplexer

// 8-to-1 multiplexer instantiation

mux8to1 mux\_inst (

.data\_in({SW[17], SW[16], SW[15], SW[14], SW[13], SW[12], SW[11], SW[10]}), // 8-bit input data from switches

.sel({SW[9], SW[8], SW[7]}), // Selection lines from switches

.data\_out(mux\_output)

);

// Connect multiplexer output to LEDs

assign LEDG = mux\_output[7:0];

assign LEDR = SW;

endmodule

module mux8to1(

input [7:0] data\_in,

input [2:0] sel,

output reg data\_out

);

always @(\*) begin

case(sel)

3'b000: data\_out = data\_in[0];

3'b001: data\_out = data\_in[1];

3'b010: data\_out = data\_in[2];

3'b011: data\_out = data\_in[3];

3'b100: data\_out = data\_in[4];

3'b101: data\_out = data\_in[5];

3'b110: data\_out = data\_in[6];

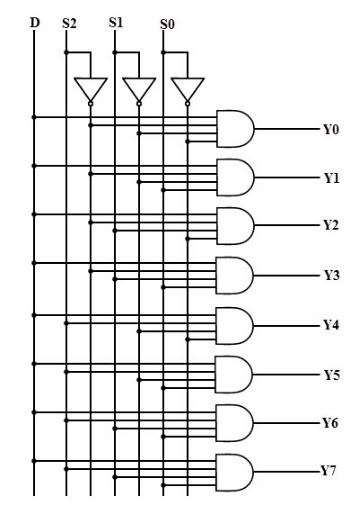
3'b111: data\_out = data\_in[7];

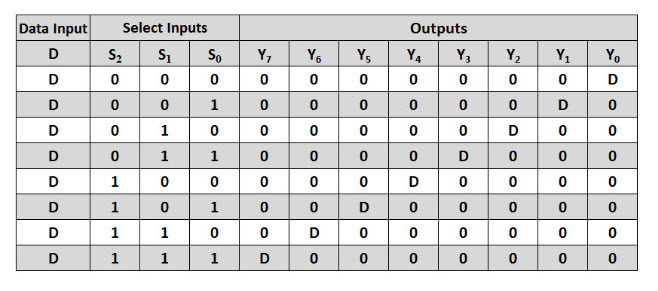
default: data\_out = 8'b0;

endcase

end

endmodule

1. **1:8 Demultiplexer**

**Schematic**

**Truth Table**

**Verilog Code:**

module demux1to8(

input data\_in,

input [2:0] sel,

output reg [7:0] data\_out

);

always @(\*) begin

case(sel)

3'b000: data\_out = (sel == 3'b000) ? data\_in : 8'b00000000;

3'b001: data\_out = (sel == 3'b001) ? data\_in : 8'b00000000;

3'b010: data\_out = (sel == 3'b010) ? data\_in : 8'b00000000;

3'b011: data\_out = (sel == 3'b011) ? data\_in : 8'b00000000;

3'b100: data\_out = (sel == 3'b100) ? data\_in : 8'b00000000;

3'b101: data\_out = (sel == 3'b101) ? data\_in : 8'b00000000;

3'b110: data\_out = (sel == 3'b110) ? data\_in : 8'b00000000;

3'b111: data\_out = (sel == 3'b111) ? data\_in : 8'b00000000;

default: data\_out = 8'b00000000;

endcase

end

endmodule

**Test bench Code:**

`timescale 1ns / 1ps

module testbench\_demux1to8;

// Parameters

parameter CLK\_PERIOD = 10; // Clock period in nanoseconds

// Inputs

reg data\_in;

reg [2:0] sel;

reg clk;

// Outputs

wire [7:0] data\_out;

// Instantiate the demux1to8 module

demux1to8 uut (

.data\_in(data\_in),

.sel(sel),

.data\_out(data\_out)

);

// Clock generation

always #((CLK\_PERIOD/2)) clk = ~clk;

// Test stimulus

initial begin

// Initialize inputs

data\_in = 1'b0;

sel = 3'b000;

clk = 0;

// Apply test vectors

#10 data\_in = 1'b1; // Input data set to 1

#10 sel = 3'b001; // Select output 1

#10 sel = 3'b010; // Select output 2

#10 sel = 3'b011; // Select output 3

#10 sel = 3'b100; // Select output 4

#10 sel = 3'b101; // Select output 5

#10 sel = 3'b110; // Select output 6

#10 sel = 3'b111; // Select output 7

// End simulation

#10 $finish;

end

// Display output

always @(posedge clk) begin

$display("Time: %t, Selected Output: %b, Output Data: %b", $time, sel, data\_out);

end

endmodule

**Result :**

A screenshot of a computer program

Description automatically generated

**Top-level Verilog Code :**

module Lab1\_task2(

input [17:0] SW,

output [7:0] LEDG,

output [17:0] LEDR

);

wire [7:0] demux\_output; // Output of the 1:8 demultiplexer

// 1:8 demultiplexer instantiation

demux1to8 demux\_inst (

.data\_in(SW[17]), // Single input data from switch

.sel({SW[1], SW[0], SW[2]}), // Selection lines from switches

.data\_out(demux\_output)

);

// Connect demultiplexer output to green LEDs

assign LEDG = demux\_output;

// Connect switches directly to red LEDs

assign LEDR = SW;

endmodule

module demux1to8(

input data\_in,

input [2:0] sel,

output reg [7:0] data\_out

);

always @(\*) begin

case(sel)

3'b000: data\_out = (sel == 3'b000) ? data\_in : 8'b00000000;

3'b001: data\_out = (sel == 3'b001) ? data\_in : 8'b00000000;

3'b010: data\_out = (sel == 3'b010) ? data\_in : 8'b00000000;

3'b011: data\_out = (sel == 3'b011) ? data\_in : 8'b00000000;

3'b100: data\_out = (sel == 3'b100) ? data\_in : 8'b00000000;

3'b101: data\_out = (sel == 3'b101) ? data\_in : 8'b00000000;

3'b110: data\_out = (sel == 3'b110) ? data\_in : 8'b00000000;

3'b111: data\_out = (sel == 3'b111) ? data\_in : 8'b00000000;

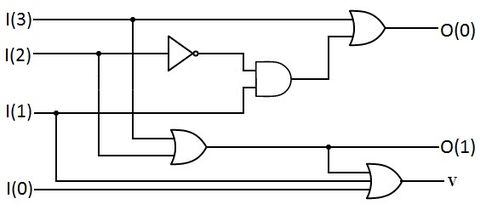
default: data\_out = 8'b00000000;

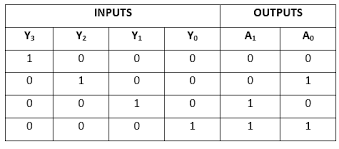
endcase

end

endmodule

1. **4-to-2 bit Encoder**

**Schematic**

**Truth Table**

**Verilog Code:**

module encoder4to2(

input [3:0] data\_in,

output reg [1:0] data\_out

);

always @(\*) begin

case(data\_in)

4'b0001: data\_out = 2'b00;

4'b0010: data\_out = 2'b01;

4'b0100: data\_out = 2'b10;

4'b1000: data\_out = 2'b11;

default: data\_out = 2'b00; // Default case

endcase

end

endmodule

**Test bench Code:**

`timescale 1ns / 1ps

module testbench\_encoder4to2;

// Parameters

parameter CLK\_PERIOD = 10; // Clock period in nanoseconds

// Inputs

reg [3:0] data\_in;

reg clk;

// Outputs

wire [1:0] data\_out;

// Instantiate the encoder4to2 module

encoder4to2 uut (

.data\_in(data\_in),

.data\_out(data\_out)

);

// Clock generation

always #((CLK\_PERIOD/2)) clk = ~clk;

// Test stimulus

initial begin

// Initialize inputs

data\_in = 4'b0000;

clk = 0;

// Apply test vectors

#10 data\_in = 4'b0001; // Input data set to 0001

#10 data\_in = 4'b0010; // Input data set to 0010

#10 data\_in = 4'b0100; // Input data set to 0100

#10 data\_in = 4'b1000; // Input data set to 1000

// End simulation

#10 $finish;

end

// Display output

always @(posedge clk) begin

$display("Time: %t, Input: %b, Output: %b", $time, data\_in, data\_out);

end

endmodule

**Result :**

A screenshot of a computer

Description automatically generated

**Top-level Verilog Code :**

module Lab1\_task3(

input [17:0] SW,

output [7:0] LEDG,

output [17:0] LEDR

);

wire [1:0] encoder\_output; // Output of the 4-to-2 bit encoder

// 4-to-2 bit encoder instantiation

encoder4to2 encoder\_inst (

.data\_in(SW[3:0]), // 4-bit input data from switches

.data\_out(encoder\_output)

);

// Connect encoder output to green LEDs

assign LEDG = encoder\_output;

// Connect switches directly to red LEDs

assign LEDR = SW;

endmodule

module encoder4to2(

input [3:0] data\_in,

output reg [1:0] data\_out

);

always @(\*) begin

case(data\_in)

4'b0001: data\_out = 2'b00;

4'b0010: data\_out = 2'b01;

4'b0100: data\_out = 2'b10;

4'b1000: data\_out = 2'b11;

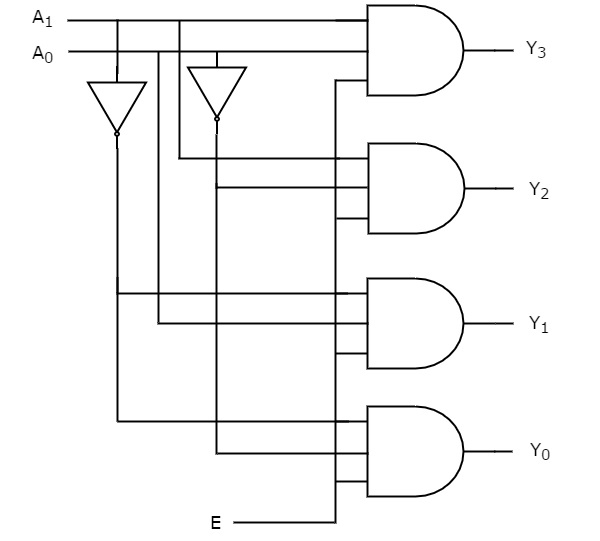
default: data\_out = 2'b00;

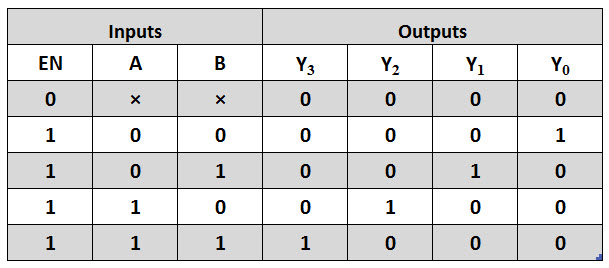
endcase

end

endmodule

1. **2-to-4 Binary Decoders**

**Schematic**

**Truth Table**

**Verilog Code:**

module decoder2to4(

input [1:0] data\_in,

output reg [3:0] data\_out

);

always @(\*) begin

case(data\_in)

2'b00: data\_out = 4'b0001;

2'b01: data\_out = 4'b0010;

2'b10: data\_out = 4'b0100;

2'b11: data\_out = 4'b1000;

default: data\_out = 4'b0000; // Default case

endcase

end

endmodule

**Test bench Code:**

`timescale 1ns / 1ps

module testbench\_decoder2to4;

// Parameters

parameter CLK\_PERIOD = 10; // Clock period in nanoseconds

// Inputs

reg [1:0] data\_in;

reg clk;

// Outputs

wire [3:0] data\_out;

// Instantiate the decoder2to4 module

decoder2to4 uut (

.data\_in(data\_in),

.data\_out(data\_out)

);

// Clock generation

always #((CLK\_PERIOD/2)) clk = ~clk;

// Test stimulus

initial begin

// Initialize inputs

data\_in = 2'b00;

clk = 0;

// Apply test vectors

#10 data\_in = 2'b01; // Input data set to 01

#10 data\_in = 2'b10; // Input data set to 10

#10 data\_in = 2'b11; // Input data set to 11

// End simulation

#10 $finish;

end

// Display output

always @(posedge clk) begin

$display("Time: %t, Input: %b, Output: %b", $time, data\_in, data\_out);

end

endmodule

**Result :**

A black text on a white background

Description automatically generated

**Top-level Verilog Code :**

module Lab1\_task4(

input [17:0] SW,

output [7:0] LEDG,

output [17:0] LEDR

);

wire [3:0] decoder\_output; // Output of the 2-to-4 bit decoder

// 2-to-4 bit decoder instantiation

decoder2to4 decoder\_inst (

.data\_in(SW[1:0]), // 2-bit input data from switches

.data\_out(decoder\_output)

);

// Connect decoder output to green LEDs

assign LEDG = decoder\_output;

// Connect switches directly to red LEDs

assign LEDR = SW;

endmodule

module decoder2to4(

input [1:0] data\_in,

output reg [3:0] data\_out

);

always @(\*) begin

case(data\_in)

2'b00: data\_out = 4'b0001;

2'b01: data\_out = 4'b0010;

2'b10: data\_out = 4'b0100;

2'b11: data\_out = 4'b1000;

default: data\_out = 4'b0000;

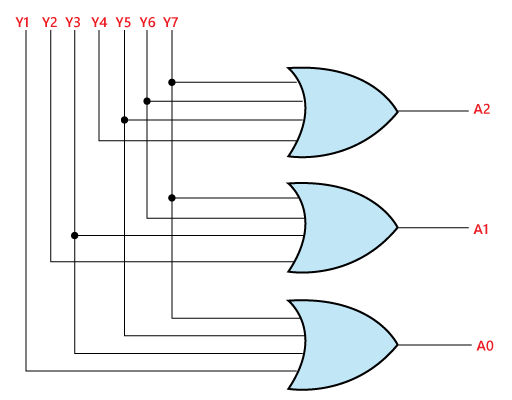
endcase

end

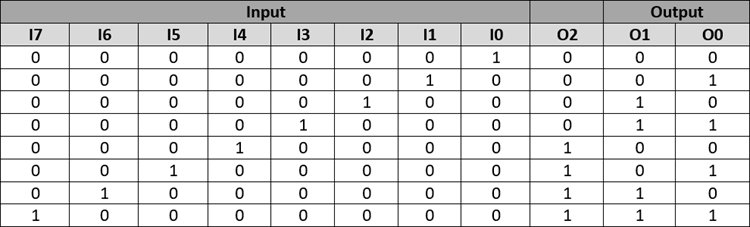
endmodule

1. **8:3 Encoder**

**Schematic**



**Truth Table**



**Verilog Code:**

module encoder8to3(

input [7:0] data\_in,

output reg [2:0] data\_out

);

always @(\*) begin

case(data\_in)

8'b00000001: data\_out = 3'b000;

8'b00000010: data\_out = 3'b001;

8'b00000100: data\_out = 3'b010;

8'b00001000: data\_out = 3'b011;

8'b00010000: data\_out = 3'b100;

8'b00100000: data\_out = 3'b101;

8'b01000000: data\_out = 3'b110;

8'b10000000: data\_out = 3'b111;

default: data\_out = 3'b000; // Default case

endcase

end

endmodule

**Test bench Code:**

`timescale 1ns / 1ps

module testbench\_encoder8to3;

// Parameters

parameter CLK\_PERIOD = 10; // Clock period in nanoseconds

// Inputs

reg [7:0] data\_in;

reg clk;

// Outputs

wire [2:0] data\_out;

// Instantiate the encoder8to3 module

encoder8to3 uut (

.data\_in(data\_in),

.data\_out(data\_out)

);

// Clock generation

always #((CLK\_PERIOD/2)) clk = ~clk;

// Test stimulus

initial begin

// Initialize inputs

data\_in = 8'b00000000;

clk = 0;

// Apply test vectors

#10 data\_in = 8'b00000001; // Input data set to 00000001

#10 data\_in = 8'b00000010; // Input data set to 00000010

#10 data\_in = 8'b00000100; // Input data set to 00000100

#10 data\_in = 8'b00001000; // Input data set to 00001000

#10 data\_in = 8'b00010000; // Input data set to 00010000

#10 data\_in = 8'b00100000; // Input data set to 00100000

#10 data\_in = 8'b01000000; // Input data set to 01000000

#10 data\_in = 8'b10000000; // Input data set to 10000000

// End simulation

#10 $finish;

end

// Display output

always @(posedge clk) begin

$display("Time: %t, Input: %b, Output: %b", $time, data\_in, data\_out);

end

endmodule

**Result :**

**A computer screen shot of a number

Description automatically generated**

**Top-level Verilog Code :**

module Lab1\_task5(

input [17:0] SW,

output [7:0] LEDG,

output [17:0] LEDR

);

wire [2:0] encoder\_output; // Output of the 8:3 encoder

// 8:3 encoder instantiation

encoder8to3 encoder\_inst (

.data\_in(SW),

.data\_out(encoder\_output)

);

// Connect encoder output to green LEDs

assign LEDG = encoder\_output;

// Connect switches directly to red LEDs

assign LEDR = SW;

endmodule

module encoder8to3(

input [7:0] data\_in,

output reg [2:0] data\_out

);

always @(\*) begin

case(data\_in)

8'b00000001: data\_out = 3'b000;

8'b00000010: data\_out = 3'b001;

8'b00000100: data\_out = 3'b010;

8'b00001000: data\_out = 3'b011;

8'b00010000: data\_out = 3'b100;

8'b00100000: data\_out = 3'b101;

8'b01000000: data\_out = 3'b110;

8'b10000000: data\_out = 3'b111;

default: data\_out = 3'b000;

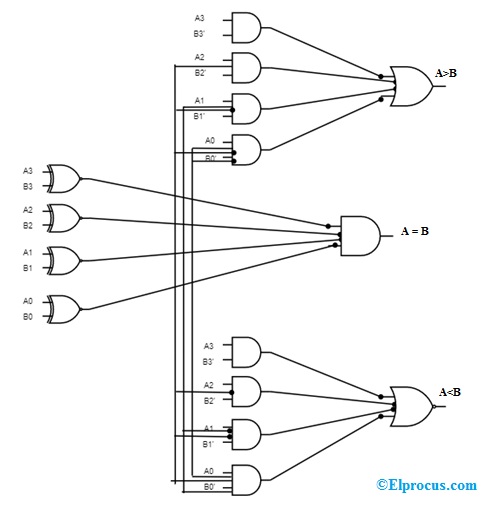
endcase

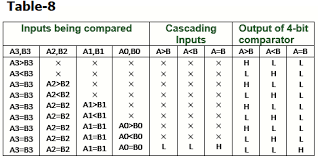
end

endmodule

1. **4 Bit Comparator**

**Schematic**



**Truth Table**

**Verilog Code:**

module comparator4bit(

input [3:0] A,

input [3:0] B,

output equal,

output greater\_than,

output less\_than

);

assign equal = (A == B);

assign greater\_than = (A > B);

assign less\_than = (A < B);

endmodule

**Test bench Code:**

`timescale 1ns / 1ps

module testbench\_comparator4bit;

// Parameters

parameter CLK\_PERIOD = 10; // Clock period in nanoseconds

// Inputs

reg [3:0] A;

reg [3:0] B;

reg clk;

// Outputs

wire equal;

wire greater\_than;

wire less\_than;

// Instantiate the comparator4bit module

comparator4bit uut (

.A(A),

.B(B),

.equal(equal),

.greater\_than(greater\_than),

.less\_than(less\_than)

);

// Clock generation

always #((CLK\_PERIOD/2)) clk = ~clk;

// Test stimulus

initial begin

// Initialize inputs

A = 4'b0000;

B = 4'b0000;

clk = 0;

// Apply test vectors

#10 A = 4'b0001; B = 4'b0000; // A > B

#10 A = 4'b0000; B = 4'b0001; // A < B

#10 A = 4'b0000; B = 4'b0000; // A = B

// End simulation

#10 $finish;

end

// Display output

always @(posedge clk) begin

$display("Time: %t, A: %b, B: %b, Equal: %b, Greater Than: %b, Less Than: %b", $time, A, B, equal, greater\_than, less\_than);

end

endmodule

**Result :**

**A black text on a white background

Description automatically generated**

**Top-level Verilog Code :**

module Lab1\_task6(

input [17:0] SW,

output [7:0] LEDG,

output [17:0] LEDR

);

wire equal;

wire greater\_than;

wire less\_than;

// Instantiate the comparator4bit module

comparator4bit DUT (

.A(SW[7:4]),

.B(SW[3:0]),

.equal(equal),

.greater\_than(greater\_than),

.less\_than(less\_than)

);

// Connect LEDs based on comparator outputs

assign LEDG[0] = equal;

assign LEDG[1] = greater\_than;

assign LEDG[2] = less\_than;

// Connect switches directly to red LEDs

assign LEDR = SW;

endmodule

module comparator4bit(

input [3:0] A,

input [3:0] B,

output equal,

output greater\_than,

output less\_than

);

assign equal = (A == B);

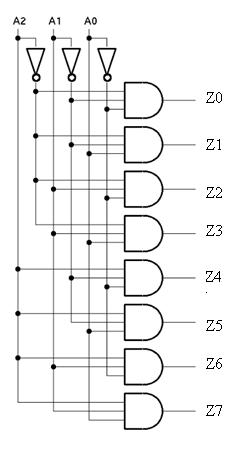
assign greater\_than = (A > B);

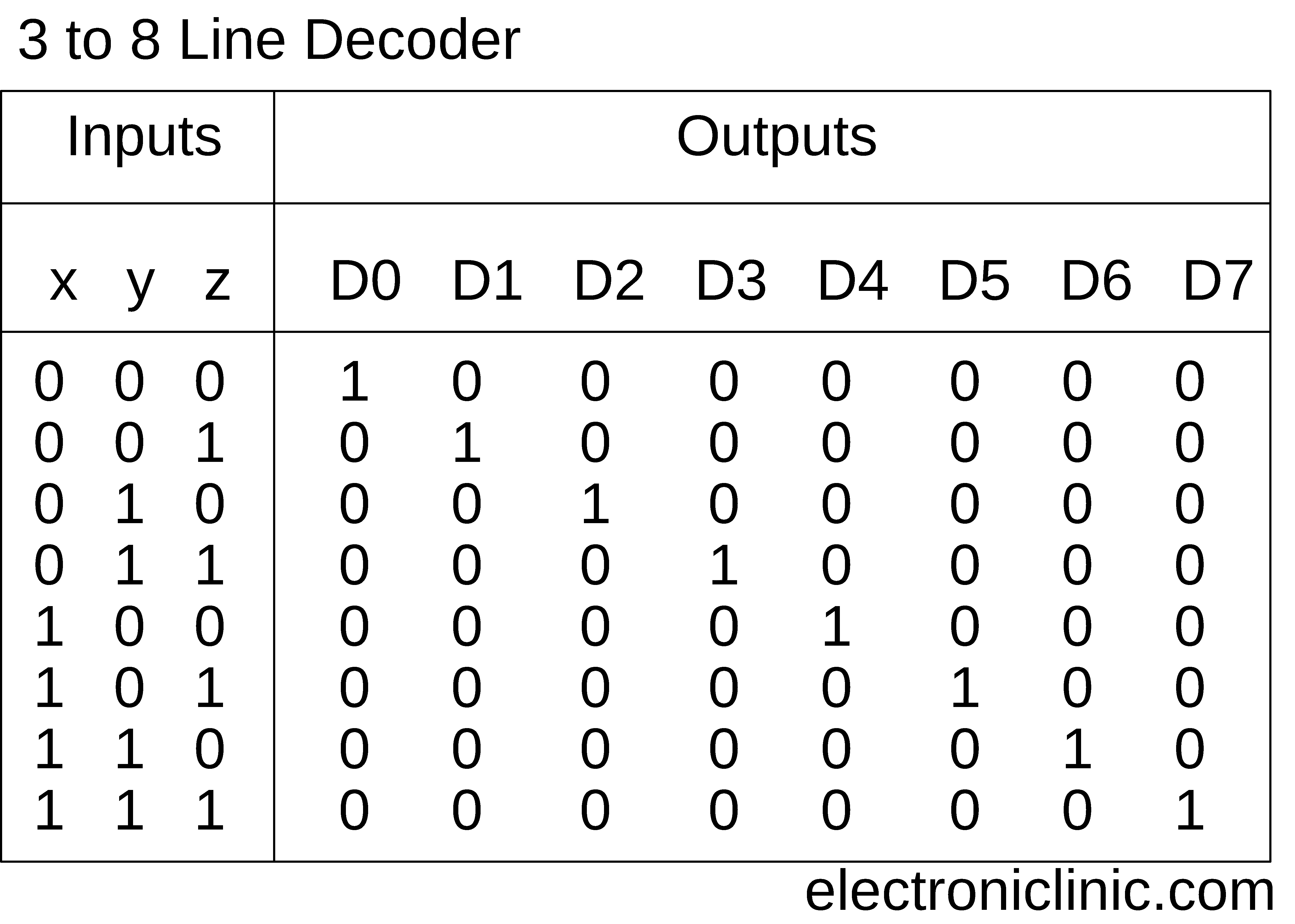
assign less\_than = (A < B);

endmodule

1. **3:8 Decoder**

**Schematic**



**Truth Table**

**Verilog Code:**

module decoder3to8(

input [2:0] data\_in,

output reg [7:0] data\_out

);

always @(\*) begin

case(data\_in)

3'b000: data\_out = 8'b00000001;

3'b001: data\_out = 8'b00000010;

3'b010: data\_out = 8'b00000100;

3'b011: data\_out = 8'b00001000;

3'b100: data\_out = 8'b00010000;

3'b101: data\_out = 8'b00100000;

3'b110: data\_out = 8'b01000000;

3'b111: data\_out = 8'b10000000;

default: data\_out = 8'b00000000; // Default case

endcase

end

endmodule

**Test bench Code:**

`timescale 1ns / 1ps

module testbench\_decoder3to8;

// Parameters

parameter CLK\_PERIOD = 10; // Clock period in nanoseconds

// Inputs

reg [2:0] data\_in;

reg clk;

// Outputs

wire [7:0] data\_out;

// Instantiate the decoder3to8 module

decoder3to8 uut (

.data\_in(data\_in),

.data\_out(data\_out)

);

// Clock generation

always #((CLK\_PERIOD/2)) clk = ~clk;

// Test stimulus

initial begin

// Initialize inputs

data\_in = 3'b000;

clk = 0;

// Apply test vectors

#10 data\_in = 3'b001; // Input data set to 001

#10 data\_in = 3'b010; // Input data set to 010

#10 data\_in = 3'b011; // Input data set to 011

#10 data\_in = 3'b100; // Input data set to 100

#10 data\_in = 3'b101; // Input data set to 101

#10 data\_in = 3'b110; // Input data set to 110

#10 data\_in = 3'b111; // Input data set to 111

// End simulation

#10 $finish;

end

// Display output

always @(posedge clk) begin

$display("Time: %t, Input: %b, Output: %b", $time, data\_in, data\_out);

end

endmodule

**Result :**

**A screenshot of a computer

Description automatically generated**

**Top-level Verilog Code :**

module Lab1\_task7(

input [17:0] SW,

output [7:0] LEDG,

output [17:0] LEDR

);

wire [7:0] decoder\_output; // Output of the 3:8 decoder

// 3:8 decoder instantiation

decoder3to8 decoder\_inst (

.data\_in(SW[1:3]), // 3-bit input data from switches

.data\_out(decoder\_output)

);

// Connect decoder output to green LEDs

assign LEDG = decoder\_output;

// Connect switches directly to red LEDs

assign LEDR = SW;

endmodule

module decoder3to8(

input [2:0] data\_in,

output reg [7:0] data\_out

);

always @(\*) begin

case(data\_in)

3'b000: data\_out = 8'b00000001;

3'b001: data\_out = 8'b00000010;

3'b010: data\_out = 8'b00000100;

3'b011: data\_out = 8'b00001000;

3'b100: data\_out = 8'b00010000;

3'b101: data\_out = 8'b00100000;

3'b110: data\_out = 8'b01000000;

3'b111: data\_out = 8'b10000000;

default: data\_out = 8'b00000000;

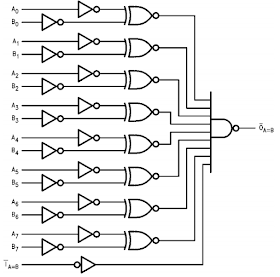
endcase

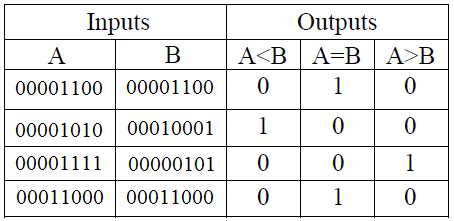
end

endmodule

1. **8 Bit Comparator**

**Schematic**



**Truth Table**

**Verilog Code:**

module comparator8bit(

input [7:0] A,

input [7:0] B,

output equal,

output greater\_than,

output less\_than

);

assign equal = (A == B);

assign greater\_than = (A > B);

assign less\_than = (A < B);

endmodule

**Test bench Code:**

`timescale 1ns / 1ps

module testbench\_comparator8bit;

// Parameters

parameter CLK\_PERIOD = 10; // Clock period in nanoseconds

// Inputs

reg [7:0] A;

reg [7:0] B;

reg clk;

// Outputs

wire equal;

wire greater\_than;

wire less\_than;

// Instantiate the comparator8bit module

comparator8bit uut (

.A(A),

.B(B),

.equal(equal),

.greater\_than(greater\_than),

.less\_than(less\_than)

);

// Clock generation

always #((CLK\_PERIOD/2)) clk = ~clk;

// Test stimulus

initial begin

// Initialize inputs

A = 8'b0;

B = 8'b0;

clk = 0;

// Apply test vectors

#10 A = 8'b00000001; B = 8'b00000000; // A > B

#10 A = 8'b00000000; B = 8'b00000001; // A < B

#10 A = 8'b00000001; B = 8'b00000001; // A = B

// End simulation

#10 $finish;

end

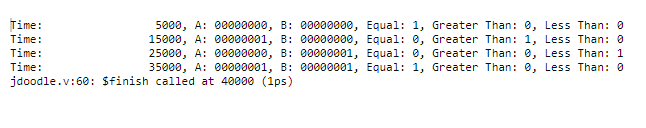
// Display output

always @(posedge clk) begin

$display("Time: %t, A: %b, B: %b, Equal: %b, Greater Than: %b, Less Than: %b", $time, A, B, equal, greater\_than, less\_than);

end

endmodule

**Result :**

**Top-level Verilog Code :**

module Lab1\_task8(

input [17:0] SW,

output [7:0] LEDG,

output [17:0] LEDR

);

wire equal;

wire greater\_than;

wire less\_than;

// Instantiate the comparator8bit module

comparator8bit DUT (

.A(SW[7:0]),

.B(SW[15:8]),

.equal(equal),

.greater\_than(greater\_than),

.less\_than(less\_than)

);

// Connect LEDs based on comparator outputs

assign LEDG[0] = equal;

assign LEDG[1] = greater\_than;

assign LEDG[2] = less\_than;

// Connect switches directly to red LEDs

assign LEDR = SW;

endmodule

module comparator8bit(

input [7:0] A,

input [7:0] B,

output equal,

output greater\_than,

output less\_than

);

assign equal = (A == B);

assign greater\_than = (A > B);

assign less\_than = (A < B);

endmodule

1. **N bit comparator**

**Schematic**

**A diagram of a circuit

Description automatically generated**

**Truth Table**

**A grid of numbers and symbols

Description automatically generated**

**Verilog Code:**

module comparator\_Nbit(

input [N-1:0] A,

input [N-1:0] B,

output equal,

output greater\_than,

output less\_than

);

assign equal = (A == B);

assign greater\_than = (A > B);

assign less\_than = (A < B);

endmodule

**Test bench Code:**

`timescale 1ns / 1ps

module testbench\_comparator\_Nbit;

// Parameters

parameter N = 4; // Number of bits

parameter CLK\_PERIOD = 10; // Clock period in nanoseconds

// Inputs

reg [N-1:0] A;

reg [N-1:0] B;

reg clk;

// Outputs

wire equal;

wire greater\_than;

wire less\_than;

// Instantiate the comparator\_Nbit module

comparator\_Nbit uut (

.A(A),

.B(B),

.equal(equal),

.greater\_than(greater\_than),

.less\_than(less\_than)

);

// Clock generation

always #((CLK\_PERIOD/2)) clk = ~clk;

// Test stimulus

initial begin

// Initialize inputs

A = 4'b0000;

B = 4'b0000;

clk = 0;

// Apply test vectors

#10 A = 4'b0001; B = 4'b0000; // A > B

#10 A = 4'b0000; B = 4'b0001; // A < B

#10 A = 4'b0000; B = 4'b0000; // A = B

// End simulation

#10 $finish;

end

// Display output

always @(posedge clk) begin

$display("Time: %t, A: %b, B: %b, Equal: %b, Greater Than: %b, Less Than: %b", $time, A, B, equal, greater\_than, less\_than);

end

endmodule

**Result :**

**Top-level Verilog Code :**

module Lab1\_task9(

input [17:0] SW,

output [7:0] LEDG,

output [17:0] LEDR

);

parameter N = 18; // Define the number of bits

wire equal;

wire greater\_than;

wire less\_than;

// Instantiate the comparator\_Nbit module

comparator\_Nbit DUT (

.A(SW[N-1:N/2]),

.B(SW[N/2-1:0]),

.equal(equal),

.greater\_than(greater\_than),

.less\_than(less\_than)

);

// Connect LEDs based on comparator outputs

assign LEDG[0] = equal;

assign LEDG[1] = greater\_than;

assign LEDG[2] = less\_than;

// Connect switches directly to red LEDs

assign LEDR = SW;

endmodule

For each circuit, Do the following steps:

Step 1 : Draw the Schematic of this circuit

### (Show Schematic in Lab report)

Step 2 : Write the truth Table for this circuit

### (Show The Truth Table in Lab report)

Step 3 : Write the Verilog Module to implement this circuit ( using structural, data flow, and behavior modeling)

### (Show Verilog codes of this module in Lab report)

Step 4 : Write the testbench to simulate the Verilog modules of this circuit

### (Show simulation results in Lab report)

Step 5 : Write the Top-level Verilog Code to implement the Verilog modules of this circuit in DE2-FPGA Kit

### (Show implementation results in Lab report)

**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, 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).