**B.M.S. College of Engineering**

***(Autonomous Institution affiliated to VTU, Belagavi)***

**Bengaluru - 19**

**Department of Computer Science and Engineering**



**Verilog Program List**

**19CSPC34**

Laboratory Manual

(Autonomous Scheme 2019)

**B.M.S. College of Engineering**

***(Autonomous Institution affiliated to VTU, Belagavi)***

**Bengaluru - 19**

**Department of Computer Science and Engineering**



**Laboratory Certificate**

This is to certify that Mr. / Ms \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_has satisfactorily completed the course of Experiments in Practical \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ prescribed by the Department during the year \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Name of the Candidate: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

USN No.: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Semester: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |
| --- | --- |
| Marks | |
| Max. Marks | Obtained |
| **50** |  |

|  |  |
| --- | --- |
| Marks in Words | |
|  |  |

**Signature of the staff in-charge Head of the Department**

**Date:**

**Verilog Program List**

**19CSPC34**

Laboratory Manual

|  |  |
| --- | --- |
| **Serial No.** | Title |
|  | **CYCLE I**  **Combinational Circuits Designs** |
| 1 | (a) Multiplexer (b) Demultiplexer |
| 2 | (a) Decoder (b) Encoder |
| 3 | (a) BCD to Excess-3code convertor (b) 4 bit Gray code to Binary code convertor |
| 4 | (a)1-Bit comparator (b)4-Bit comparator using 2-bit Comparator |
| 5 | (a) 1- bit Full Adder and Ripple Carry Adder Using behavioral architecture and structural architecture (b) Carry Look Ahead Adder |
|  | **CYCLE II**  **Sequential Circuits Design** |
| 6 | (a) JK-Flipflop (b) D Flipflop |
| 7 | (a) Synchronous counter (b) Asynchronous counter, (c) BCD counters (d) Universal Shift Register |
| 8 | Sequence Detector |
| 9 | Serial Adder |

**Verilog Program List-19CSPC34**

**SCHEME OF CONDUCT AND EVALUATION**

**CLASS: III SEMESTER CIE MARKS: (Max.): 50**

**YEAR: 2019-20 SEE MARKS: (Max): 50**

**EVALUATION SCHEMESEE: 3hrs**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Expt. No.** | **TITLE** | Max. Marks | Marks Obtained | Signature |
| 1. | Design, simulate and implement Multiplexer and Demultiplexer using Verilog code. | 10 |  |  |
| 2. | Design, simulate and implement Encoder and Decoder using Verilog code. | 10 |  |  |
| 3. | Design, simulate and implement Code converter using Verilog code. | 10 |  |  |
| 4. | Design, simulate and implement comparator using Verilog code. | 10 |  |  |
| 5. | Design, simulate and implement adders using Verilog code. | 10 |  |  |
| 6. | Design, simulate and implement Flip flops using Verilog code. | 10 |  |  |
| 7. | Design, simulate and implement counters using Verilog code. | 10 |  |  |
| 8. | Design, simulate and implement Sequence detector using Verilog code. | 10 |  |  |
| 9. | Design, simulate and implement Serial adder using Verilog code. | 10 |  |  |
|  |  |  |  |  |
|  | **TOTAL** | 40 |  |  |
|  | **TEST** | 10 |  |  |
|  | **TOTAL MARKS** | 50 |  |  |
|  |  | |  |  |

**Verilog Program List-19CSPC34**

**Rubrics**

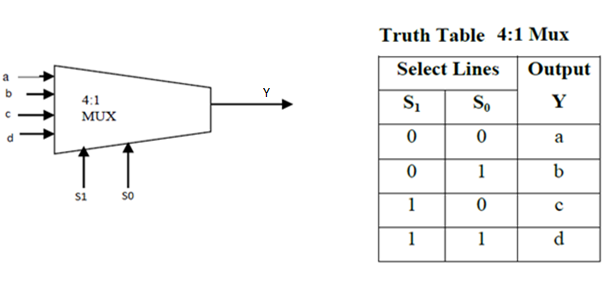
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Sl.No** | **Criteria** | **Excellent** | **Good** | **Average** | **Max Score** |
| **Data sheet** | | | | | |
| A | Problem statement | 9-10 | 6-8 | 1-5 | **10** |
| B | Design & specifications | 9-10 | 6-8 | 1-5 | **10** |
| C | Expected output | 9-10 | 6-8 | 1-5 | **10** |
| **Record** | | | | | |
| D | Simulation/ Conduction of the experiment | 14-15 | 11-13 | 1-10 | **15** |
| E | Analysis of the result. | 14-15 | 11-13 | 1-10 | **15** |
| **Viva** | | | | | **40** |
| **Total** | | | | | **100** |
| **Scale down to 10 marks** | | | | | |

**Experiment 1**

**Multiplexer and Demultiplexer**

**Aim:** Design and write **V**ERILOG code for (a) Multiplexer and (b) Demultiplexers.Perform high-level HDL simulation and verify theexpected output. Synthesize and implement the design on FPGA kit.

# a) 4:1 Multiplexer

****

**Code:**

modulemux41(a,b,c,d, s, y);

inputa,b,c,d;

input [1:0] s;

output y;

reg y;

always@(s,a,b,c,d)

begin

case(s)

2'b00: y = a;

2'b01: y = b;

2'b10: y =c;

2'b11: y =d;

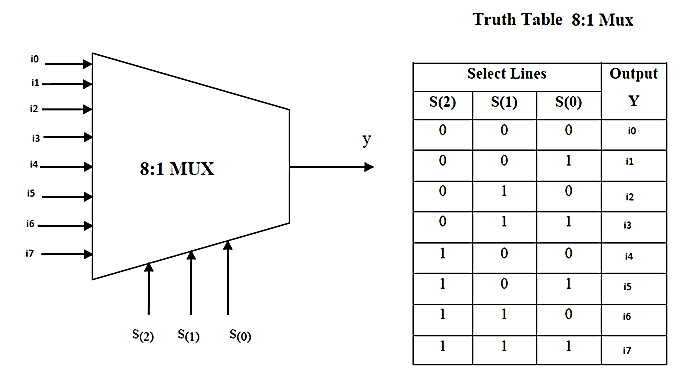
default: y=1’b Z;

endcase

end

endmodule

# b) 8:1 Multiplexer



**Code:**

module mux81(i, s, y);

input [7:0] i;

input [2:0] s;

output y;

reg y;

always @ ( s or i )

begin

case (s)

3' b 000: y = i [0];

3' b 001: y = i [1];

3' b 010: y = i [2];

3' b 011: y = i [3];

3' b 100: y = i [4];

3' b 101: y = i [5];

3' b 110: y = i [6];

3' b 111: y = i [7];

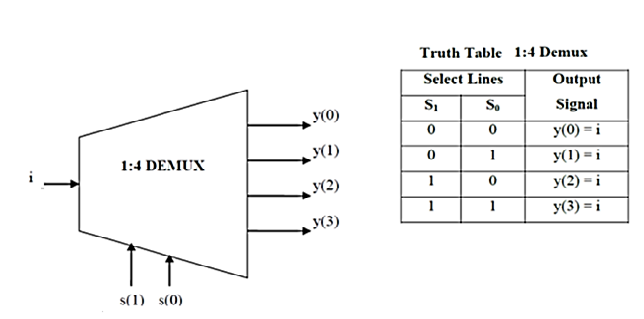
default: y=1’bZ;

endcase

end

endmodule

# c) 1:4 Demultiplexer



**Code:**

moduledemux14(i, s, y);

input [0:0] i;

input [1:0] s;

output [3:0] y;

reg [3:0]y;

always@ (s ,i)

begin

y = 4'b0000;

case (s)

2'b00:y[0] = i;

2'b01:y[1] = i;

2'b10:y[2] = i;

2'b11:y[3] = i;

default: y=4’b0000;

endcase

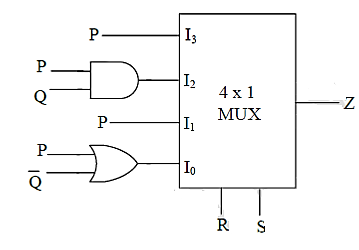
end

endmodule

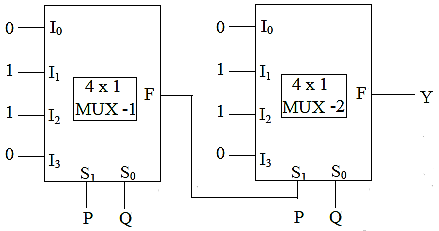
**Result:**

**Practice Questions:**

1. Design and write a Verilog code to implement 1:8 Demultiplexer.
2. Design and write a Verilog code to implement the following circuit



1. Design and write a Verilog code to implement the following circuit



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Sl.No** | | **Criteria** | Max Marks | Marks obtained |  |
| **Data sheet** | | | | |  |
| A | | Problem statement | 10 |  |  |
| B | | Design & specifications | 10 |  |  |
| C | | Expected output | 10 |  |  |
| **Record** | | | | |  |
| D | Simulation/ Conduction of the experiment | | 15 |  |  |
| E | Analysis of the result. | | 15 |  |  |
|  | Viva | | 40 |  |  |
|  | Total | | 100 |  |  |
| **Scale down to 10 marks** | | | | |  |

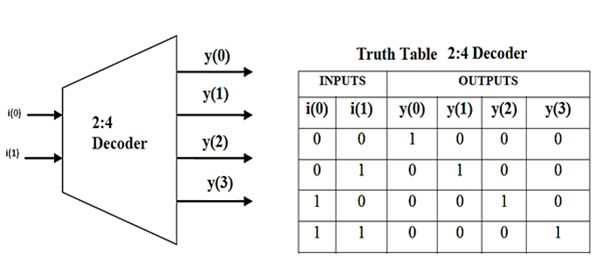
**Signature of the staff in-charge**

**Experiment 2**

**Decoders and Encoders**

**Aim:** Design and write **V**ERILOG code for (a) Decoder and (b) Encoder. Perform high-level HDL simulation and verify theexpected output. Synthesize and implement the design on FPGA kit.

# a) 2:4 Decoder



**Code:**

module decoder24(i, y);

input [1:0]i;

output [3:0]y;

reg [3:0]y;

always@(i)

begin

case (i)

2'b 00 : y = 4'b 0001;

2'b 01 : y = 4'b 0010;

2'b 10 : y = 4'b 0100;

2'b 11 : y = 4'b 1000;

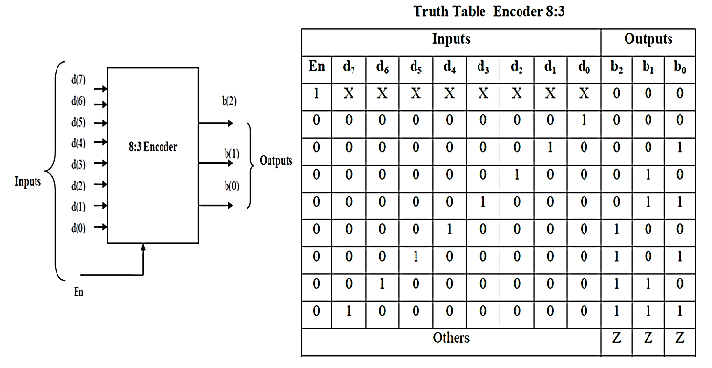
default: y=4’b0000;

endcase

end

endmodule

# b) 8:3 Encoder without priority



**Code:**

moduleenc83(En,d, b);

input En;

input [7:0] d;

output [2:0] b;

reg [2:0]b;

always@ (d, En)

begin

if(En) b = 3'b000;

else

case(d)

8'b00000001: b = 3'b000;

8'b00000010: b= 3'b001;

8'b00000100: b = 3'b010;

8'b00001000: b = 3'b011;

8'b00010000: b = 3'b100;

8'b00100000: b = 3'b101;

8'b01000000: b = 3'b110;

8'b10000000: b = 3'b111;

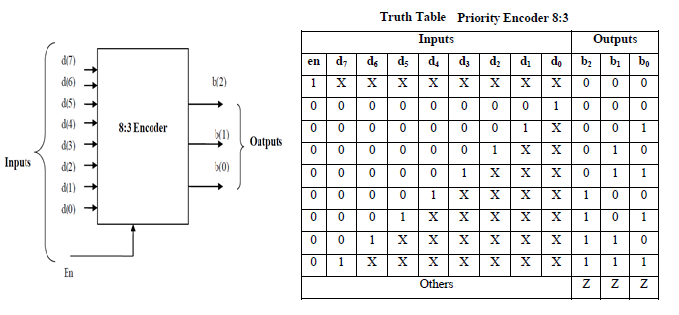
default: b = 3'b ZZZ;

endcase

end

endmodule

# c) 8:3 Encoder with priority



module penc83(en,d, b);

input en;

input [7:0] d;

output [2:0] b;

reg [2:0] b;

always@ (d , en)

if(en) b=3’d0;

else if(d[7]) b = 3'd7;

else if (d[6]) b= 3’d6;

else if (d[5]) b= 3’d5;

else if (d[4]) b= 3’d4;

else if (d[3]) b= 3’d3;

else if (d[2]) b= 3’d2;

else if (d[1]) b= 3’d1;

elseb= 3’d0;

endmodule

**Result:**

**Practice Questions:**

1. Design and write a Verilog code to implement 16:4 encoder (with and without priority).
2. Design and write a Verilog code to design a combinational circuit with three inputs and one outputsuch that the output is ‘1’ when binary value of input is odd, ‘0’ otherwise.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Sl.No** | | **Criteria** | Max Marks | Marks obtained |  |
| **Data sheet** | | | | |  |
| A | | Problem statement | 10 |  |  |
| B | | Design & specifications | 10 |  |  |
| C | | Expected output | 10 |  |  |
| **Record** | | | | |  |
| D | Simulation/ Conduction of the experiment | | 15 |  |  |
| E | Analysis of the result. | | 15 |  |  |
|  | Viva | | 40 |  |  |
|  | Total | | 100 |  |  |
| **Scale down to 10 marks** | | | | |  |

**Signature of the staff in-charge**

**Experiment 3**

**Code converters**

**Aim:** Design and write **V**ERILOG code for (a) BCD to Excess-3code convertor (b) 4 bit Gray code to Binary code convertor.Perform high-level HDL simulation and verify theexpected output. Synthesize and implement the design on FPGA kit.

1. **BCD to EXCESS-3 Code Convertor**

|  |  |
| --- | --- |
| **BCD(b3b2b1b0)** | **Excess-3(e3e2e1e0)** |
| **0000** | **0011**  BCD-to-Excess-3  BCD  Excess-3 |
| **0001** | **0100** |
| **0010** | **0101** |
| **0011** | **0110** |
| **0100** | **0111** |
| **0101** | **1000** |
| **0110** | **1001** |
| **0111** | **1010** |
| **1000** | **1011** |
| **1001** | **1100** |

**Code:**

module bcd\_excess3(b3,b2,b1,b0,e3,e2,e1,e0);

input b3,b2,b1,b0;

output e3,e2,e1,e0;

assign e3=b3|b0&b2|b2&b1;

assign e2=(~b1)&(~b0)&(b2)|(~b2)&(b0)|(~b2)&(b1);

assign e1=(~b1)&(~b0)|b1&b0;

assign e0=~b0;

endmodule

# b)4-bit Gray code to Binary code converter

|  |  |
| --- | --- |
| **Gray** | **Binary** |
| **0000** | **0000** |
| **0001** | **0001** |
| **0011** | **0010** |
| **0010** | **0011** |
| **0100** | **0100** |
| **0101** | **0101** |
| **0111** | **0110** |
| **0110** | **0111** |
| **1100** | **1000** |
| **1101** | **1001** |
| **1111** | **1010** |
| **1110** | **1011** |
| **1000** | **1100** |
| **1001** | **1101** |
| **1011** | **1110** |
| **1010** | **1111** |

Gray-to-Binary

Gray

Binary

**Code:**

moduleGray\_to\_Binary ( gray ,binary );

output [3:0] binary ;

input [3:0] gray;

assign binary[3] = gray[3];

assign binary[2] = gray[3]^gray[2];

assign binary[1] = gray[3]^gray[2]^gray[1];

assign binary[0] = gray[3]^gray[2]^gray[1]^gray[0];

endmodule

**Result:**

**Practice Question:**

1. Design and write a Verilog code Excess-3 to BCD converter

2. Design and write a Verilog code for binary to EXCESS-3.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sl.No** | | **Criteria** | Max Marks | Marks obtained |
| **Data sheet** | | | | |
| A | | Problem statement | 10 |  |
| B | | Design & specifications | 10 |  |
| C | | Expected output | 10 |  |
| **Record** | | | | |
| D | Simulation/ Conduction of the experiment | | 15 |  |
| E | Analysis of the result. | | 15 |  |
|  | Viva | | 40 |  |
|  | Total | | 100 |  |
| **Scale down to 10 marks** | | | | |

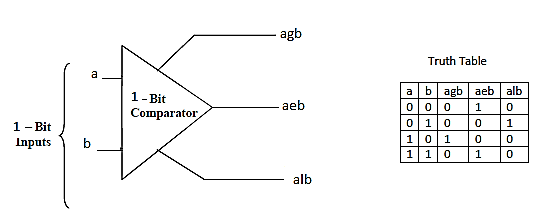
**Signature of the staff in-charge**

**Experiment 4**

**Comparators**

**Aim:** Design and write **V**ERILOG code for (a) 1-Bit comparator (b) 4-Bit comparator using 2-bit Comparator.Perform high-level HDL simulation and verify theexpected output. Synthesize and implement the design on FPGA kit.

1. **1-bit Comparator**



module comparator\_1bit(input a, input b,

outputagb, output aeb, output alb);

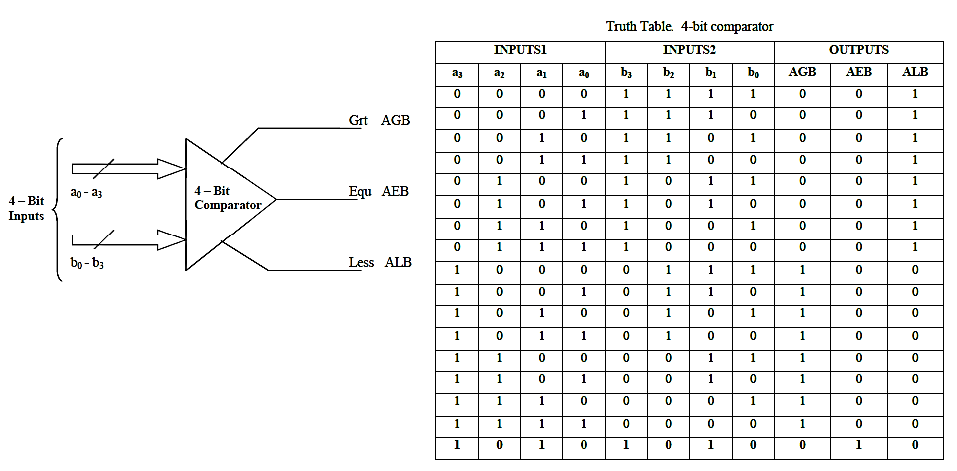
assignagb= a & (~b);

assignaeb= ~(a ^b);

assignalb = (~a )& b;

endmodule

# b) 4-bit Comparator (structural)using 2-bit Comparator



**Code:**

**For 2-bit Comparator (Gate level modeling)**

module comparator\_2bit(input [1:0]a, input [1:0]b,

outputagb, output aeb, output alb);

wire w1,w2,w3,w4,w5,w6,w7,w8,w9,w10;

not (w1,b[1]);

and (w2,w1,a[1]);

not (w3,b[0]);

and (w4,a[0],w1,w3);

and (w5,a[1],a[0],w3);

or (agb,w2,w4,w5);

not (w6,a[1]);

not (w7,a[0]);

and (w8,w6,b[1]);

and (w9,w6,w7,b[0]);

and (w10,w7,b[1],b[0]);

or (alb,w8,w9,w10);

xnor (aeb,alb,agb);

endmodule

**For 4-bit Comparator (Structural)**

module comparator(input [3:0]a, input [3:0]b, output a\_g\_b, output a\_e\_b, output a\_l\_b);

wire l1,e1,g1,l2,e2,g2,d1,d2;

comparator\_2bit f51(a[3:2],b[3:2],g1,e1,l1);

comparator\_2bit f52(a[1:0],b[1:0],g2,e2,l2);

and (d1,e1,g2);

and (d2,e1,l2);

or (a\_g\_b,g1,d1);

or (a\_l\_b,l1,d2);

and (a\_e\_b,e1,e2);

endmodule

**Result:**

**Practice Question:**

1.. Design and write a Verilog code for 2-bit comparator using Data flow modeling.

2. Design and write a Verilog code for 8-bit comparator.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sl.No** | | **Criteria** | Max Marks | Marks obtained |
| **Data sheet** | | | | |
| A | | Problem statement | 10 |  |
| B | | Design & specifications | 10 |  |
| C | | Expected output | 10 |  |
| **Record** | | | | |
| D | Simulation/ Conduction of the experiment | | 15 |  |
| E | Analysis of the result. | | 15 |  |
|  | Viva | | 40 |  |
|  | Total | | 100 |  |
| **Scale down to 10 marks** | | | | |

**Signature of the staff in-charge**

**Experiment5**

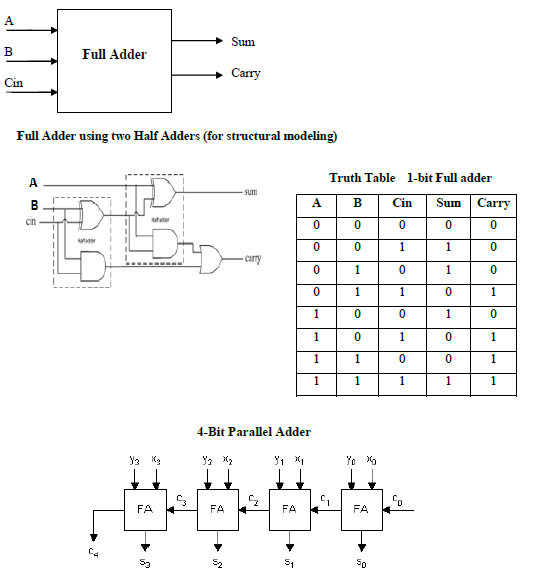
**Binary Adder**

**(Ripple Carry Adder and Carry look ahead adder)**

**Aim: a)** Design and write VERILOG code of 1-bit Full Adder and 4-bit Ripple Carry Adder Using Dataflow Architecture and Structural Architecture.Perform high-level HDL simulation and verify theexpected output. Synthesize and implement the design on FPGA kit.

**b)** Design and write VERILOG code of Carry Look Ahead Adder.Perform high-level HDL simulation and verify theexpected output. Synthesize and implement the design on FPGA kit.

**a)1-bit Full Adder and 4-bit Ripple Carry Adder**

****

**Code: 1-bit Full Adder(Dataflow Modeling)**

modulefulladder(A,B,cin,Sum,Carry);

input A;

input B;

inputcin;

output Sum;

output Carry;

assign Sum = A^B^cin;

assign Carry = (A&B)|(A&cin)|(B&cin);

endmodule

**Code: Half Adder, 1-bit Full Adder and 4-bit Ripple Carry Adder**

**For Half Adder**

modulehalf\_adder( output Sum,Carry,

input A,B );

xor(Sum,A,B);

and(Carry,A,B);

endmodule

**For Full Adder**

modulefull\_adder( output Sum,Cout,

inputA,B,Cin );

wire s1,c1,c2;

half\_adder ha1(s1,c1,A,B);

half\_adder ha2(Sum,c2,s1,Cin);

or or1(Cout,c1,c2);

endmodule

**For4-bit Ripple Carry Adder**

module ripple\_adder\_4bit(output [3:0] Sum,

outputCout,

input [3:0] A,B,

inputCin );

wire c1,c2,c3;

full\_adderFA1(Sum[0],c1,A[0],B[0],Cin),

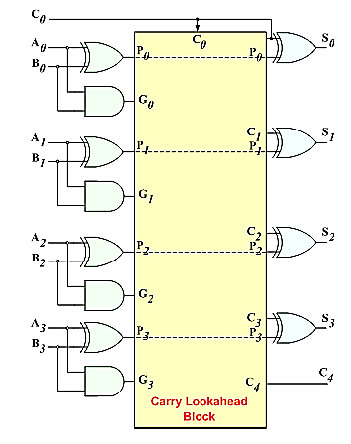
FA2(Sum[1],c2,A[1],B[1],c1),

FA3(Sum[2],c3,A[2],B[2],c2),

FA4(Sum[3],Cout,A[3],B[3],c3);

endmodule

# Carry Look Ahead Adder



**Code:**

# modulecarry\_look\_adder (input [3:0] a, b, input cin, output [3:0] sum, output cout );

# wire g0,p0,g1,p1,g2,p2,g3,p3;

# wire c1,c2,c3;

# assign g0 = a[0] & b[0];

# assign p0 = a[0] ^ b[0];

# assign sum[0] = a[0] ^ b[0] ^ cin;

# assign c1 = g0 | (p0 &cin);

# assign g1 = a[1] & b[1];

# assign p1 = a[1] ^ b[1];

# assign sum[1] = a[1] ^ b[1] ^ c1;

# assign c2 = g1 | (p1 & c1);

# assign g2 = a[2] & b[2];

# assign p2 = a[2] ^ b[2];

# assign sum[2] = a[2] ^ b[2] ^ c2;

# assign c3 = g2 | (p2 & c2);

# assign g3 = a[3] & b[3];

# assign p3 = a[3] ^ b[3];

# assign sum[3] = a[3] ^ b[3] ^ c3;

# assigncout = g3 | (p3 & c3);

# endmodule

**Result:**

**Practice Question:**

1. Design and write a Verilog code for a 1-bit full subtractor using logic equations (Difference = A-B-Bin).

2. Design and write a Verilog code for a 4-bit subtractor using the module defined in question1.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sl.No** | | **Criteria** | Max Marks | Marks obtained |
| **Data sheet** | | | | |
| A | | Problem statement | 10 |  |
| B | | Design & specifications | 10 |  |
| C | | Expected output | 10 |  |
| **Record** | | | | |
| D | Simulation/ Conduction of the experiment | | 15 |  |
| E | Analysis of the result. | | 15 |  |
|  | Viva | | 40 |  |
|  | Total | | 100 |  |
| **Scale down to 10 marks** | | | | |

**Signature of the staff in-charge**

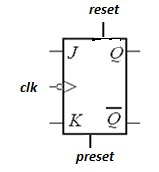
**Experiment6**

1. **JK Flip Flop (b) D Flip Flop**

**Aim:** Design and write VERILOG codes for (a) JK Flip Flop (b) D Flipflop.Perform high-level HDL simulation and verify theexpected output. Synthesize and implement the design on FPGA kit.

1. **JK Flipflop**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Reset** | **Preset** | **J** | **K** | **Qn** | **Qn+1** | **Description** |
| **1** | **0** | **X** | **X** | **X** | **0** | **Clear** |
| **0** | **1** | **X** | **X** | **X** | **1** | **Set** |
| **0** | **0** | **0** | **0** | **0** | **0** | **Hold** |
| **0** | **0** | **0** | **0** | **1** | **1** |
| **0** | **0** | **0** | **1** | **0** | **0** | **Clear** |
| **0** | **0** | **0** | **1** | **1** | **0** |
| **0** | **0** | **1** | **0** | **0** | **1** | **Set** |
| **0** | **0** | **1** | **0** | **1** | **1** |
| **0** | **0** | **1** | **1** | **0** | **1** | **Toggle** |
| **0** | **0** | **1** | **1** | **1** | **0** |

****

**Code:**

# modulejkflop( input clk, input reset, input [1:0] jk, output reg q );

# always @ (negedgeclk)

# if(reset) q<=1'b0;

# else if (preset)q<=1'b1;

# else

# case (jk)

# 2'b00: q<=q;

# 2'b01: q<=1'b0;

# 2'b10: q<=1'b1;

# 2'b11: q<=~q;

# default: q<=1'bZ;

# endcase

# endmodule

1. **D Flipflop**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Reset** | **D** | **Qn** | **Qn+1** | **Description** |
| **1** | **X** | **X** | **0** | **Clear** |
| **0** | **0** | **0** | **0** | **Clear** |
| **0** | **0** | **1** | **0** |
| **0** | **1** | **0** | **1** | **Set** |
| **0** | **1** | **1** | **1** |

**Code:**

# moduled\_ff(q,clk,rst,din);

# output q;

# inputclk,din,rst;

# reg q;

# 

# always @(posedgeclk or posedgerst)

# begin

# if(rst)

# q <= 1'b0;

# else

# q <= din;

# end

# endmodule

**Result:**

**Practice Question:**

1. Design and write a Verilog code to implement a T Flip Flop.
2. Design and write a Verilog code to implement a master slave JK Flip Flop.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sl.No** | | **Criteria** | Max Marks | Marks obtained |
| **Data sheet** | | | | |
| A | | Problem statement | 10 |  |
| B | | Design & specifications | 10 |  |
| C | | Expected output | 10 |  |
| **Record** | | | | |
| D | Simulation/ Conduction of the experiment | | 15 |  |
| E | Analysis of the result. | | 15 |  |
|  | Viva | | 40 |  |
|  | Total | | 100 |  |
| **Scale down to 10 marks** | | | | |

**Signature of the staff in-charge**

**Experiment7**

**(a) Synchronous counter (b) Asynchronous counter,**

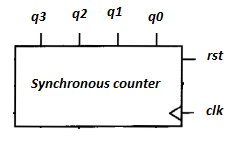
**(c) BCD counter (d) Universal Shift Register**

**Aim:** Design and write VERILOG codes for a) 4 - bit Synchronous binary counter (b) 4 - bit Asynchronous binary counter (c) 4 - bit BCD counter with Asynchronous Reset (d) Universal shift Register.Perform high-level HDL simulation and verify theexpected output. Synthesize and implement the design on FPGA kit.

**a) Binary Synchronous Counter:**It should be a synchronous (4-bit) up counter (0 to 15) with output count that works as follows: All state changes occur on the rising edge of the CLKinput. When reset=1,

the counter resets regardless of the values of the other inputs.

|  |  |
| --- | --- |
| **CLK** | **Count(q3q2q1q0)** |
| ↑ | **0000** |
| ↑ | **0001** |
| ↑ | **0010** |
| ↑ | **0011** |
| ↑ | **0100** |
| ↑ | **0101** |
| ↑ | **0110** |
| ↑ | **0111** |
| ↑ | **1000** |
| ↑ | **1001** |
| ↑ | **1010** |
| ↑ | **1011** |
| ↑ | **1100** |
| ↑ | **1101** |
| ↑ | **1110** |
| ↑ | **1111** |



**Code:**

modulecounter\_behav ( count,rst,clk);

inputrst, clk;

outputreg [3:0] count;

always @(posedgeclk)

if (reset)

count<= 4'b0000;

else

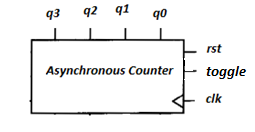
count<= count + 4'b0001;

endmodule

**b) Binary Asynchronous Counter**

Asynchronous counter is where only the first flip-flop is clocked by an external clock. All subsequent flip-flops are clocked by the output of the preceding flip-flop.

|  |  |
| --- | --- |
| **CLK** | **Count(q3q2q1q0)** |
| ↑ | **0000** |
| ↑ | **0001** |
| ↑ | **0010** |
| ↑ | **0011** |
| ↑ | **0100** |
| ↑ | **0101** |
| ↑ | **0110** |
| ↑ | **0111** |
| ↑ | **1000** |
| ↑ | **1001** |
| ↑ | **1010** |
| ↑ | **1011** |
| ↑ | **1100** |
| ↑ | **1101** |
| ↑ | **1110** |
| ↑ | **1111** |

****

**Code:**

moduleripple\_counter (clock, toggle, reset, count);

input clock, toggle, reset;

output [3:0] count;

reg [3:0] count;

wire c0, c1, c2;

assign c0 = count[0], c1 = count[1], c2 = count[2];

always @ (posedge reset or posedge clock)

if (reset == 1'b1) count[0] <= 1'b0;

else if (toggle == 1'b1) count[0] <= ~count[0];

always @ (posedge reset or negedge c0)

if (reset == 1'b1) count[1] <= 1'b0;

else if (toggle == 1'b1) count[1] <= ~count[1];

always @ (posedge reset or negedge c1)

if (reset == 1'b1) count[2] <= 1'b0;

else if (toggle == 1'b1) count[2] <= ~count[2];

always @ (posedge reset or negedge c2)

if (reset == 1'b1) count[3] <= 1'b0;

else if (toggle == 1'b1) count[3] <= ~count[3];

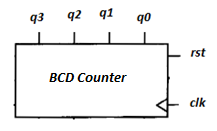
endmodule

# c) BCD Synchronous Counter

It should be a synchronous (4-bit) up decade counter with output count that works as follows: All state changes occur on the rising edge of the CLKinput, except the asynchronous reset (reset). When reset=1,

the counter reset regardless of the values of the other inputs.

|  |  |
| --- | --- |
| **CLK** | **Count(q3q2q1q0)** |
| ↑ | **0000** |
| ↑ | **0001** |
| ↑ | **0010** |
| ↑ | **0011** |
| ↑ | **0100** |
| ↑ | **0101** |
| ↑ | **0110** |
| ↑ | **0111** |
| ↑ | **1000** |
| ↑ | **1001** |



**Code:**

# modulebcd\_count(clk, rst, count);

# inputclk;

# inputrst;

# output [3:0] count;

# reg[3:0] count;

always @ (posedgeclk or posedgerst)

# begin

# if(rst) count = 4'b0000;

# else if (count ==4’b1001) count=4’b0000;

else

# count = count+1;

# end

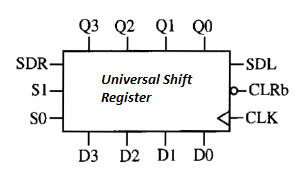
# endmodule

**d) Universal shift register**

The CLRb input is asynchronous and active low and overrides all the other inputs. All other states changes following the rising edge of the clock. If the control inputs S1=S0=1, the register is load in parallel. If S1=1 and S0=0, the register is shifted right and SDR (Serial data right) is shifted into Q3.If S1=0 and S0=1, the register is shifted left and SDL

is shifted into Q0. If S1=S0=0, no action occurs.

|  |  |  |
| --- | --- | --- |
| **S(S1S0)** | **Operation** | **Description** |
| 00 | Q=Q | Hold |
| 01 | Q=Q[2:0],SDL | Shift left |
| 10 | Q=SDR,Q[3:1] | Shift right |
| 11 | Q=D | Parallel Load |



**Code:**

moduleUniversalshiftregister( input clk, input clrb, input [3:0] D, input SDL, input SDR,

input [1:0] S, output reg [3:0] Q );

always @ ( posedgeclk, negedgeclrb)

if (~clrb) Q<=4'b0000;

else

case (S)

2'b00:Q<=Q;

2'b01: Q<={Q[2:0],SDL};

2'b10: Q<={SDR,Q[3:1]};

2'b11: Q<=D;

default: Q<=4'b0000;

endcase

endmodule

**Result:**

**Practice questions:**

1. Design and write a Verilog code for 3-bit arbitrary counter (0, 3, 6, 7, 1, 0).
2. Design and write a Verilog code for 4-bit up/down counter.
3. Add a terminal count (tc) to a Verilog synchronous counter code. Terminal count (tc) goes high when count reaches 15.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sl.No** | | **Criteria** | Max Marks | Marks obtained |
| **Data sheet** | | | | |
| A | | Problem statement | 10 |  |
| B | | Design & specifications | 10 |  |
| C | | Expected output | 10 |  |
| **Record** | | | | |
| D | Simulation/ Conduction of the experiment | | 15 |  |
| E | Analysis of the result. | | 15 |  |
|  | Viva | | 40 |  |
|  | Total | | 100 |  |
| **Scale down to 10 marks** | | | | |

**Signature of the staff in-charge**

**Experiment 8**

**Sequence Detector**

**Aim:** Design and writeVERILOG code for sequence detector “101” using Mealy FSM.Perform high-level HDL simulation and verify theexpected output. Synthesize and implement the design on FPGA kit.

# 

**Code:**

moduleseqdetector( input a, input clk, input reset, output q );

parameter [1:0] S0=2'b00, S1=2'b01, S2=2'b10;

reg [1:0] PS,NS;

reg q;

always @ (PS,a)

case (PS)

S0 : if(a) NS=S1;

else NS=S0;

S1: if(a) NS=S1;

else NS=S10;

S2: if(a) NS=S1;

else NS=S0;

default: NS=S0;

endcase

always @ (posedgeclk)

if(reset) PS<= S0;

else PS<= NS;

always @ (PS,a)

case (PS)

S0 : q=1'b0;

S1 : q=1'b0;

S2: if(a) q=1'b1;

else q=1'b0;

default: q=1'b0;

endcase

endmodule

**Result:**

**Practice Question:**

1. Design and write a Verilog code sequence detector “101” using Moore Machine Model.
2. Design and write a Verilog code sequence detector “1010”.
3. Design and write a Verilog code sequence detector “111000111”.

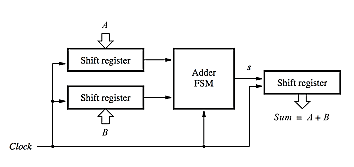
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sl.No** | | **Criteria** | Max Marks | Marks obtained |
| **Data sheet** | | | | |
| A | | Problem statement | 10 |  |
| B | | Design & specifications | 10 |  |
| C | | Expected output | 10 |  |
| **Record** | | | | |
| D | Simulation/ Conduction of the experiment | | 15 |  |
| E | Analysis of the result. | | 15 |  |
|  | Viva | | 40 |  |
|  | Total | | 100 |  |
| **Scale down to 10 marks** | | | | |

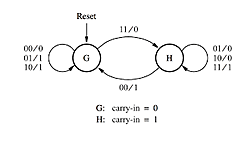
**Signature of the staff in-charge**

**Experiment 9**

**Serial adder**

**Aim:** Design and write VERILOG codes to implement the functionality8-bit serial adder.Perform high-level HDL simulation and verify theexpected output. Synthesize and implement the design on FPGA kit.

****

****

**Code:**

|  |  |
| --- | --- |
| module serial\_adder ( A,B, reset, clock, sum);  input [7:0] A,B;  input reset,clock;  output [7:0] sum;  reg [3:0] count;  regs,y,Y;  wire [7:0] qa,qb,sum;  wire run;  parameter G=0,H=1;  shiftrneshift\_A (A,reset,1'b1,1'b0,clock,qa);  shiftrneshift\_B (B,reset,1'b1,1'b0,clock,qb);  shiftrneshift\_sum (8'b0,reset,run,s,clock,sum);  //adder fsm  //output and next state combinational circuit  always @(qa or qb or y)  case (y)  G: begin  s = qa[0]^qb[0];  if (qa[0] &qb[0])  Y = H;  else  Y = G;  end  H: begin  s = qa[0] ~^qb[0];  if (~qa[0] & ~qb[0])  Y =G;  else  Y = H;  end  default : Y = G;  endcase | //sequential block  always @(posedge clock)  if (reset)  y <= G;  else  y <= Y;  //control the shifting process  always @(posedge clock)  if (reset)  count = 8;  else if (run) count = count - 1;  assign run=|count;  endmodule  // **shift register**  module shiftrne ( R,L,E,w,clock,q);  parameter n=8;  input [n-1:0] R;  input L,E,w,clock;  output [n-1:0] q;  reg [n-1:0] q;  integer k;  always @(posedge clock)  if (L)  q <= R;  else if (E)  begin  for (k=n-1;k>0;k=k-1)  q[k-1] <= q[k];  q[n-1] <= w;  end  endmodule |

**Result:**

**Practice Question:**

1. Design and write a Verilog code for 6-bit ring counter.
2. Design and write a Verilog code for 4-bit Johnson counter.
3. Design and write s Verilog Code for serial subtractor.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sl.No** | | **Criteria** | Max Marks | Marks obtained |
| **Data sheet** | | | | |
| A | | Problem statement | 10 |  |
| B | | Design & specifications | 10 |  |
| C | | Expected output | 10 |  |
| **Record** | | | | |
| D | Simulation/ Conduction of the experiment | | 15 |  |
| E | Analysis of the result. | | 15 |  |
|  | Viva | | 40 |  |
|  | Total | | 100 |  |
| **Scale down to 10 marks** | | | | |

**Signature of the staff in-charge**