EE5193 FPGA and Verilog HDL

Assignment 2

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  | Truth | Table |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| A | B | C | D | o13 | o12 | o11 | o10 | o9 | o8 | o7 | o6 | o5 | o4 | o3 | o2 | o1 | o0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | X | X | X | X | X | X | X | X | X | X | X | X | X | X |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | X | X | X | X | X | X | X | X | X | X | X | X | X | X |
| 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

1a. Generate the truth table for the given inputs and outputs.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| o12 | CD |  |  |  |
| AB | 00 | 01 | 11 | 10 |
| 00 | 0 | 0 | 0 | 0 |
| 01 | 0 | 0 | 0 | 0 |
| 11 | 0 | 1 | 0 | X |
| 10 | 0 | X | 0 | 0 |

1b. Drawing the K-Maps for the obtained truth tables.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| o13 | CD |  |  |  |
| AB | 00 | 01 | 11 | 10 |
| 00 | 0 | 0 | 0 | 0 |
| 01 | 0 | 0 | 0 | 0 |
| 11 | 0 | 0 | 1 | X |
| 10 | 0 | X | 0 | 0 |

o13 = ABC o12 = AC’D

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| o11 | CD |  |  |  |
| AB | 00 | 01 | 11 | 10 |
| 00 | 0 | 0 | 0 | 0 |
| 01 | 0 | 0 | 0 | 0 |
| 11 | 1 | 0 | 0 | X |
| 10 | 0 | X | 0 | 0 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| o10 | CD |  |  |  |
| AB | 00 | 01 | 11 | 10 |
| 00 | 0 | 0 | 0 | 0 |
| 01 | 0 | 0 | 0 | 0 |
| 11 | 0 | 0 | 0 | X |
| 10 | 0 | X | 1 | 0 |

o11 = ABD’ o10 = AB’D

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| o9 | CD |  |  |  |
| AB | 00 | 01 | 11 | 10 |
| 00 | 0 | 0 | 0 | 0 |
| 01 | 0 | 0 | 0 | 0 |
| 11 | 0 | 0 | 0 | X |
| 10 | 0 | X | 0 | 1 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| o8 | CD |  |  |  |
| AB | 00 | 01 | 11 | 10 |
| 00 | 0 | 0 | 0 | 0 |
| 01 | 0 | 0 | 0 | 0 |
| 11 | 0 | 0 | 0 | X |
| 10 | 1 | X | 0 | 0 |

o9 = ACD’ o8 = AB’C’

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| o7 | CD |  |  |  |
| AB | 00 | 01 | 11 | 10 |
| 00 | 0 | 0 | 0 | 0 |
| 01 | 0 | 0 | 1 | 0 |
| 11 | 0 | 0 | 0 | X |
| 10 | 0 | X | 0 | 0 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| o6 | CD |  |  |  |
| AB | 00 | 01 | 11 | 10 |
| 00 | 0 | 0 | 0 | 0 |
| 01 | 0 | 0 | 0 | 1 |
| 11 | 0 | 0 | 0 | X |
| 10 | 0 | X | 0 | 0 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| o4 | CD |  |  |  |
| AB | 00 | 01 | 11 | 10 |
| 00 | 0 | 0 | 0 | 0 |
| 01 | 1 | 0 | 0 | 0 |
| 11 | 0 | 0 | 0 | X |
| 10 | 0 | X | 0 | 0 |

o7 = A’BCD o6 = BCD’

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| o5 | CD |  |  |  |
| AB | 00 | 01 | 11 | 10 |
| 00 | 0 | 0 | 0 | 0 |
| 01 | 0 | 1 | 0 | 0 |
| 11 | 0 | 0 | 0 | X |
| 10 | 0 | X | 0 | 0 |

o5 = A’BC’D o4 = A’BC’D’

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| o3 | CD |  |  |  |
| AB | 00 | 01 | 11 | 10 |
| 00 | 0 | 0 | 1 | 0 |
| 01 | 0 | 0 | 0 | 0 |
| 11 | 0 | 0 | 0 | X |
| 10 | 0 | X | 0 | 0 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| o2 | CD |  |  |  |
| AB | 00 | 01 | 11 | 10 |
| 00 | 0 | 0 | 0 | 1 |
| 01 | 0 | 0 | 0 | 0 |
| 11 | 0 | 0 | 0 | X |
| 10 | 0 | X | 0 | 0 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| o1 | CD |  |  |  |
| AB | 00 | 01 | 11 | 10 |
| 00 | 0 | 1 | 0 | 0 |
| 01 | 0 | 0 | 0 | 0 |
| 11 | 0 | 0 | 0 | X |
| 10 | 0 | X | 0 | 0 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| o0 | CD |  |  |  |  |
| AB | 00 |  | 01 | 11 | 10 |
| 00 | 1 |  | 0 | 0 | 0 |
| 01 | 0 |  | 0 | 0 | 0 |
| 11 | 0 |  | 0 | 0 | X |
| 10 | 0 |  | X | 0 | 0 |

o3 = A’B’CD o2 = A’B’CD’

o1 = B’C’D o0 = A’B’C’D’



1d. Test bench

![A screenshot of a cell phone

Description automatically generated]()

1e. Simulation

![A screenshot of text

Description automatically generated]()

1g. Simulate using the test bench.

A screenshot of a cell phone

Description automatically generated

2. Design an arbiter in Verilog and test using Xilinx simulator.

module arbiter(

req\_0,

req\_1,

req\_2,

clk,

rst,

gnt\_0,

gnt\_1,

gnt\_2,

state,

nstate

);

input req\_0;

input req\_1;

input req\_2;

input clk;

input rst;

output gnt\_0;

output gnt\_1;

output gnt\_2;

output [1:0] state;

output [1:0] nstate;

reg [1:0] state; // current state, output is shown at this state

reg [1:0] nstate; // next state, input is computed at this state

reg gnt\_0; // outputs

reg gnt\_1;

reg gnt\_2;

wire gnt\_0\_out; // To demonstrate the functionality of this circuit

wire gnt\_1\_out; // an equation was created from a truth table.

wire gnt\_2\_out; // Wires are created for assign statements to be used as net data types

// seen in the state machine below.

// Based on the specifications of the design. A truth table was input into

// a program called Logic Friday which can generate a boolean equation

// from the truthtable and minimize the equation. The result is seen below.

// These equations fulfill the described conditions.

// When only req\_0 is asserted, gnt\_0 is asserted.

// "" req\_1 "", gnt\_1 "".

// "" req\_2 "", gnt\_2 "".

// When both req\_0 and req\_1 are asserted, then gnt\_0 is asserted.

// "" req\_1 and req\_2 "", then gnt\_1 "".

// For all other combinations, gnt\_2 is asserted.

assign gnt\_0\_out = ~req\_2 & req\_0;

assign gnt\_1\_out = req\_1 & ~req\_0;

assign gnt\_2\_out = req\_2 & req\_0 + ~req\_1 & ~req\_0;

// reset and update state

always @( posedge rst or posedge clk )

begin

if ( rst )

begin

state <= 2'b00; // For the reset, I left state 0 as an empty

nstate <= 2'b00; // reset state. This is because if you assign

end // gnt\_0 to be associated with state 0, or any output for that matter,

else // then one may run into an issue where gnt\_0 is high in instances where it

state <= nstate; // is not wanted to be.

end

// calculating the next state

always @( state or req\_0 or req\_1 or req\_2 )

begin

case ( state )

2'b01: if ( gnt\_0\_out ) // The assign statements created using the

nstate <= 2'b01; // net data type outputs are used as the conditionals

else if ( gnt\_1\_out ) // in this sequential function.

nstate <= 2'b10;

else if ( gnt\_2\_out )

nstate <= 2'b11;

else

nstate <= 2'b11;

2'b10: if ( gnt\_0\_out )

nstate <= 2'b01;

else if ( gnt\_1\_out )

nstate <= 2'b10;

else if ( gnt\_2\_out )

nstate <= 2'b11;

else

nstate <= 2'b11;

2'b11: if ( gnt\_0\_out )

nstate <= 2'b01;

else if ( gnt\_1\_out )

nstate <= 2'b10;

else if ( gnt\_2\_out )

nstate <= 2'b11;

else

nstate <= 2'b11;

default: nstate <= 2'b11;

endcase

end

// output of the design

always @( state ) // Because this is a sequential design

begin // the output will not be realized until one

case ( state ) // clock cycle after the input is realized.

2'b01: begin // Mentioned above gnt\_0 is associated

gnt\_0 <= 1; // with state 1 and so on to allow for a reset state 0...

gnt\_1 <= 0;

gnt\_2 <= 0;

end

2'b10: begin

gnt\_0 <= 0;

gnt\_1 <= 1;

gnt\_2 <= 0;

end

2'b11: begin

gnt\_0 <= 0;

gnt\_1 <= 0;

gnt\_2 <= 1;

end

default: begin

gnt\_0 <= 0;

gnt\_1 <= 0;

gnt\_2 <= 0;

end

endcase

end

endmodule

module tb\_arbiter();

reg req\_0;

reg req\_1;

reg req\_2;

reg clk;

reg rst;

wire gnt\_0;

wire gnt\_1;

wire gnt\_2;

wire [1:0] state;

wire [1:0] nstate;

arbiter UUT

(

.req\_0 ( req\_0 ),

.req\_1 ( req\_1 ),

.req\_2 ( req\_2 ),

.clk ( clk ),

.rst ( rst ),

.gnt\_0 ( gnt\_0 ),

.gnt\_1 ( gnt\_1 ),

.gnt\_2 ( gnt\_2 ),

.state ( state ),

.nstate ( nstate )

);

initial

begin

rst = 1;

clk = 0;

forever #5 clk = ~clk;

end

initial

begin

#10 rst = 0; // state 0

req\_0 = 0; // next state 0 with no inputs

req\_1 = 0; // default next state 3 with inputs. next output gnt\_2

req\_2 = 0;

#15 req\_0 = 1; // state 3. output gnt\_2

req\_1 = 0; // next state 1. next output gnt\_0

req\_2 = 0;

#10 req\_0 = 0; // state 1. output gnt\_0

req\_1 = 1; // next state 2. next output gnt\_1

req\_2 = 0;

#10 req\_0 = 0; // state 2. output gnt\_1

req\_1 = 0; // next state 3. next output gnt\_2

req\_2 = 1;

#10 req\_0 = 1; // state 3. output gnt\_2

req\_1 = 1; // next state 1. next output gnt\_0

req\_2 = 0;

#10 req\_0 = 0; // state 1. output gnt\_0

req\_1 = 1; // next state 2. next output gnt\_1

req\_2 = 1;

#10 req\_0 = 1; // state 2. output gnt\_1

req\_1 = 1; // next state 3. next output gnt\_2

req\_2 = 1;

#10 // buffer // state 3. output gnt\_2

// next state 3. next output gnt\_2.

#10 rst = 1; // state 0. outputs 0. Reset.

end

endmodule

