|  |  |
| --- | --- |
| **Student**: | Michael Escue |
| **Assignment**: | Assignment 6 |
| **Class**: | ECE 351 |
| **Professor**: | Dr. Garrison Greenwood |
| **Term**: | Spring 2019 |

**Questions:**

1. **Consider the following code:**

*always @(A)*

*if(A < 10)*

*Y = 0;*

*else if (A > 10 & A <= 15)*

*Y = 1;*

**Assume A can assume a value between -2 and +22.Why would a synthesizer believe a latch is necessary? Simply saying there is no “else” clause is not the answer. That causes a latch to be inferred but doesn’t explain why the synthesizer puts one in the design. I want to know why a latch is necessary.**

In the case where A is greater than 15 or equal to 10, updating the value from either of the other conditions has not been defined, and the logic reduces to latches for values of 0 and 1 since there is no defined logic for updating the value in these undefined ranges.

1. **What are three ways of preventing latch inference in case statements?**

1. Preassignment of variables.

2. Default case statement and all case values listed.

3. Using synthesis directive “synthesis full\_case” when not using the other two methods.

1. **Why should module outputs be registered?**

To improve critical paths and better meet timing constraints. Pipelining using registers between logic paths can remove the problem of long net delay and result in speedup, but introduce an initial overhead delay.

1. **Consider the following module:**

*module U1(in1, in2, y ,clk);*

*input in1, in2, clk;*

*output y;*

*reg s;*

*always @(posedge clk)*

*s = in1 ^ in2;*

*assign y = ~s;*

*endmodule*

**Edit the code so the module has a registered output.**

module U1(in1, in2, y ,clk);

input in1, in2, clk;

output y;

reg y;

reg s;

always @(posedge clk)

s <= in1 ^ in2;

y <= ~s;

endmodule

1. **Consider the following Verilog code:**

*if(A)*

*y = 1;*

*else if(B)*

*y = 2;*

*else if(C)*

*y = 3;*

*else if(D)*

*y = 4;*

*else*

*y = 0;*

* **Show what would most likely be synthesized.**

D

C

B

Y

4

3

2

1

0

A

1. **Rewrite the code from the previous problem as a case statement. Show what would be synthesized if the parallel\_case synthesis directive is used.**

0

Case(ABCD) //synthesis parallel\_case

1

A: Y=1;

Y

2

A

B: Y=2;

3

C: Y=3;

4

D: Y=4;

Default: Y=0;

D

C

B