**Course Code: EE461 Lab**

**Lab -06**

**PREPARED BY**

**Khandoker Samiul Hoque**

**Student Id: 19837**

**I. "if" Conditional statements**

- if(a == yyyy) : For "== " or "!=" checking

if a has x or z, then (a == yyyy) = x (unknown) or (a != yyyy) = x (unknown).

Example:

*module IfTest();*

*reg [3:0] X;*

*reg [3:0] Y;*

*initial begin*

*X=4'b101x;*

*Y=4'b101z;*

*if(X==4'b101z)begin*

*$display("Statement 1 has been selected!");*

*end*

*if(X==Y)begin*

*$display("Statement 2 has been selected!");*

*end*

*if(X!=Y)begin*

*$display("Statement 3 has been selected!");*

*end*

*if(X==X)begin*

*$display("Statement 4 has been selected!");*

*end*

*$display("No one has been selected!");*

*end*

*endmodule*

**Ans:**

**Output: No one has been selected!**

**Reason:**

While X has a don't care value ('x) in its third bit position, the first if condition checks if X is equal to 4'b101z.As a result, the first if statement is not performed since the condition is not satisfied. The second if statement determines if X and Y are equal. But, Y is unknown since it has a high-impedance value ('z) in the third bit position. As a result, the second if statement is not carried out since the condition is not satisfied. The third if statement determines if X and Y are not equal. This requirement is likewise not satisfied because the third bit of Y is unknown, hence the third if statement is not carried out. The fourth if statement determines whether X is equal to itself, which is a constant.even though the fourth if statement is executed, nothing is printed. As a result, none of the if statements execute, and the final message "No one has been selected!" is displayed.

- if(a===yyyy): no matter what a’s value is, even *x* or *z* , if LHS exactly matches RHS, (a===yyyy) = 1.

Example:

*module IfTest1();*

*reg [3:0] X;*

*reg [3:0] Y;*

*initial begin*

*X=4'b101x;*

*Y=4'b101z;*

*if(X===4'b101z)begin*

*$display("Statement 1 has been selected!");*

*end*

*if(X===Y)begin*

*$display("Statement 2 has been selected!");*

*end*

*if(X!==Y)begin*

*$display("Statement 3 has been selected!");*

*end*

*if(X===X)begin*

*$display("Statement 4 has been selected!");*

*end*

*$display("No one has been selected!");*

*end*

*endmodule*

**Ans:**

**Statement 3 has been selected!  
Statement 4 has been selected!  
No one has been selected!**

**Reason:**

If X's value matches the binary value "101z" exactly, the first criterion determines if it does.The first if statement is not carried out since the condition is not satisfied because X has an unknown value ('x') in its third bit position. The second test determines if the values of X and Y exactly match each other.The second if statement is not carried out since the condition is not satisfied because Y has an unknown value ('z') in the third bit position. The third condition looks for an exact mismatch between X and Y, even values that are unknown. The third if statement is performed, displaying "Statement 3 has been picked!" as the condition is satisfied since X and Y have distinct unknown values in their third bit places. The fourth test determines if X and themselves are an exact match, which is always true. As a result, the fourth if statement is carried out, and the message "Statement 4 has been picked!" is shown.The code produces "Statement 3 has been picked!" and "Statement 4 has been selected!" in that order since the first two requirements are not satisfied but the third and fourth conditions are.Last but not least, the $display command "No one has been picked!" is also run.

**II. Continuous assignment with conditional operator**

- If a has *x* or *z* in conditional operation, then *a = x* instead of other values in assignment.

Example:

*module CondOp();*

*reg [3:0] X;*

*reg [3:0] Y;*

*wire Z;*

*assign Z=(X==Y)? 1'b1:1'b0;*

*//assign Z=(X===Y)? 1'b1:1'b0;*

*//assign Z=(X!==Y)? 1'b1:1'b0;*

*initial begin*

*X=4'b101x;*

*Y=4'b101z;*

*$monitor("Z's values: %d\n", Z);*

*end*

*endmodule*

**Ans: Z's values: x**

**Reason :**

If X and Y are equal, Z is given the value 1'b1, and if not, 1'b0.

Nevertheless, because X and Y's third bit positions contain unknown values 'x' and 'z', the comparison operation between them likewise yields an unknown value 'x'. Due to the comparison process between X and Y, Z has an unknown value and displays a "x" when printed using $monitor. In other words, the output of the conditional operator "?:" is also uncertain since X and Y have unknown values in their third bit positions, which causes a "x" to be shown by the $monitor statement.

1. **"case" conditional statement**

- "case" is to match exactly for *1, 0 , x* and *z*

Example:

*module caseTest();*

*reg [3:0] X;*

*reg [3:0] Y;*

*initial begin*

*X=4'b101x;*

*Y=4'b101z;*

*case(X)*

*4'b1010: $display("Statement 1 has been selected!");*

*4'b101x: $display("Statement 2 has been selected!");*

*4'b101z: $display("Statement 3 has been selected!");*

*4'bxxxx: $display("Statement 4 has been selected!");*

*4'bzzzz: $display("Statement 5 has been selected!");*

*default: $display("Default has been selected!");*

*endcase*

*end*

*endmodule*

**Ans: Statement 2 has been selected!**

**Reason:**

X should be checked for its value before the matching statement is run. With an unknown value ('x') in its third bit location, the initial value of X is 4'b101x.When X does not match, the case with the value 4'b1010 is checked first, and then it skips it.A second instance, 4'b101x, is run since it matches the value of X.Statement 2 has been chosen, as a result, is the output that is shown. The third example is 4'b101z, however X does not match this situation since X's third bit position has an unknown value ('x').For the same reason, the fourth and fifth examples are not chosen since they only match with unknown ('x') and high-impedance ('z'), respectively, values. Not chosen is the default case

* + 1. **"casex" conditional statement**

- "casex" is to make *x/z* match *1, 0, x* and *z*

Example:

*module casexTest();*

*reg [3:0] X;*

*reg [3:0] Y;*

*initial begin*

*X=4'b101x;*

*Y=4'b101z;*

*casex(X)*

*4'b100z: $display("Statement 1 has been selected!");*

*4'b10xx: $display("Statement 2 has been selected!");*

*4'b11xz: $display("Statement 3 has been selected!");*

*4'bxxxx: $display("Statement 4 has been selected!");*

*4'bzzzz: $display("Statement 5 has been selected!");*

*default: $display("Default has been selected!");*

*endcase*

*end*

*endmodule*

**Ans: Statement 2 has been selected!**

**Reason:**

As the 4th bit of X is unknown (x), the initial instance 4'b100z is not chosen because it matches the first three bits of X. The second instance, 4'b10xx, is chosen since the first two bits of X match it, while the latter two bits are either unknown or unimportant (x).The third example, 4'b11xz, matches the first two bits of X, but is not chosen since the fourth bit, z, is high-impedance and the third bit, x, is unknown. The fourth example, 4'bxxxx, is not chosen since it fits every conceivable value of X. The fifth example, 4'bzzzz, is not chosen since it matches every conceivable value of X with high-impedance (z) bits. The default case is not chosen in this instance since it is only chosen when none of the preceding instances match. As only the second case matches X, its output statement is run and prints "Statement 2 has been picked!" as a result.

- Note: in this case, *x* looks like wildcard (*?=1/0/x/z*). So, we can write as follows:

Example:

*module casex1();*

*reg [3:0] X;*

*reg [3:0] Y;*

*initial begin*

*X=4'b101x;*

*Y=4'b101z;*

*casex(Y)*

*4'b???1: $display("Statement 1 has been selected!");*

*4'b??1?: $display("Statement 2 has been selected!");*

*4'b?1??: $display("Statement 3 has been selected!");*

*4'b1???: $display("Statement 4 has been selected!");*

*default: $display("Default has been selected!");*

*endcase*

*end*

*endmodule*

**Ans: Statement 1 has been selected!**

**Reason:**

First caseThe potential value of Y matching 4'b???1 has the least significant bit set to 1.The second bit of SY has an unknown value (x), hence this instance is chosen. In the second instance, when the third bit is set to 1, the result of Y matches scenario 4'b??1.The third bit of Y has high-impedance (z), hence this instance is not chosen.The third situation, where the second bit is set to 1, corresponds to a potential value of Y. As Y's third bit has a high impedance value (z), this instance is not chosen. In the fourth example, 4'b1???, the most significant bit is set to 1, matching any conceivable value of Y. Y's second bit has an unknown value (x), hence this scenario is not chosen. As a result, only the first case matches Y, and as a result, its output statement is run and prints "Statement 1 has been selected!".

* + 1. **"casez" conditional statement**

- "casez" is to make *z* match *1, 0* and *z*, but not *x*.

Example 1:

*module casezTest();*

*reg [3:0] X;*

*initial begin*

*X=4'b1x00;*

*casez(X)*

*4'b100z: $display("Statement 1 has been selected!");*

*4'b10xx: $display("Statement 2 has been selected!");*

*4'b11xz: $display("Statement 3 has been selected!");*

*4'bxxxx: $display("Statement 4 has been selected!");*

*4'b0zzz: $display("Statement 5 has been selected!");*

*default: $display("Default has been selected!");*

*endcase*

*end*

*endmodule*

**Ans: Default has been selected!**

**Reason:**

To check the value of X in this code, use casez. In the event that z in X matches any bit value in the case item, the casez statement examines each bit of X and compares it to its corresponding bit in each of the case items. The first case item in the provided code is 4'b100z, which denotes that X's initial bit must be 1, its following two bits must be 0, and its final bit can take on any value. The final two bits in X are 00 and not 0, though. The last two bits of X can be any value, but the initial two bits of X must be 10. This is because the second case item is 4'b10xx.Once more, the first two bits in X are 1x rather than 10.The third case item is 4'b11xz, which implies that the final bit must be 0 or 1 and that the first two bits of X must both be 11. The third bit can have any value. The last bit in X is neither x or 1, but rather 0, hence this case item is likewise not picked.4'bxxxx, the fourth case item, matches any value of X. Due to the greater importance of all the earlier case items, this item is not chosen. The fifth case item is 4'b0zzz, which indicates that the final three bits of X can have any value but the initial bit must be 0.The first bit of X, however, is 1, hence this case item is also not chosen. The default case is carried out, resulting in the message "Default has been selected!" since none of the case elements match the value of X.

- In "casez" block, wildcard *? = 1/0/x/z*.

Example 2:

*module casez1();*

*reg [3:0] X;*

*initial begin*

*X=4'b1x00;*

*casez(X)*

*4'b???1: $display("Statement 1 has been selected!");*

*4'b??1?: $display("Statement 2 has been selected!");*

*4'b?1??: $display("Statement 3 has been selected!");*

*4'b1???: $display("Statement 4 has been selected!");*

*default: $display("Default has been selected!");*

*endcase*

*end*

*endmodule*

**Ans: Statement 2 has been selected!**

**Reason:**

To check the value of signal X to different 4-bit patterns, a casez statement is utilized. A pattern with a question mark (?) implies that it can be either 0 or 1, whereas the letters "x" or "z" imply a don't-care value. The first block assigns the value 4'b1x00 to X, indicating that the most significant bit of X is 1 and the remaining bits are don't-care. When the casez statement is run, it checks each pattern in turn until a match is found. The first pattern, 4'b???1, corresponds to the value of X since its three most important bits are don't-care and its least significant bit is 1.As a result, the message "Statement 1 has been selected! “appears. The other patterns do not match with X, thus the default statement is run instead of Statement 4, which displays "Default has been selected!" since no match is found until Statement 4 is executed.