CAD FOR VLSI DESIGN - I
Lecture 28

V. Kamakoti and Shankar Balachandran
Synthesis of casex

- module PriorityEncoder (Select, BitPosition);
- input [5:1] Select;
- output [2:0] BitPosition;
- reg [2:0] BitPosition;
- always @(Select)
  - casex (Select)
    - 5'bxxxx1 : BitPosition = 1;
    - 5'bxxx1x : BitPosition = 2;
    - 5'bxx1xx : BitPosition = 3;
    - 5'bx1xxx : BitPosition = 4;
    - 5'b1xxxx : BitPosition = 5;
    - default : BitPosition = 0;
  - endcase
- endmodule
The Semantics

- if (Select[1]) BitPosition = 1; else
- if (Select[2]) BitPosition = 2; else
- if (Select[3]) BitPosition = 3; else
- if (Select[4]) BitPosition = 4; else
- BitPosition = 0;
Inferring Latches in case statements

• If a variable is assigned a value only in some branches of a ‘case’ statement, and not in all possible branches, then, a latch is inferred for that variable.

• The rules apply equally for casex and casez statements
Example

• module StateUpdate (CurrentState, Zip);
• input [0:1] CurrentState;
• output [0:1] Zip;
• reg [0:1] Zip;
• parameter S0 = 0, S1 = 1, S2 = 2, S3 = 3;
• always @(CurrentState)
  • case (CurrentState)
  •   S0, S3: Zip = 0;
  •   S1: Zip = 3;
  • endcase
• endmodule
To Avoid Latches

• module StateUpdate (CurrentState, Zip);
• input [0:1] CurrentState;
• output [0:1] Zip;
• reg [0:1] Zip;
• parameter S0 = 0, S1 = 1, S2 = 2, S3 = 3;
• always @(CurrentState)
  begin
  Zip = 0;                 //This statement is added – mimics ‘default’
  case (CurrentState)
    S0, S3: Zip = 0;
    S1: Zip = 3;
  endcase
  end
  endmodule
Synthesis Directives

• Full Case: Suppose the designer knows that only a set of values will be taken by a ‘case’ expression, then he need not specify the other cases, but add as a comment to the line in the Verilog file which has the ‘case’ statement,

// synthesis full_case
Synthesis Directives

• The *full_case* directive ensures that no latches are inferred.

• In the following example – Toggle, a 2-bit variable takes the values 2’b01 and 2’b10 only. The other values 2’b00 and 2’b11 are not possible.
Full Case

- module NextStateLogic(NextToggle, Toggle);
- input [1:0] Toggle;
- output [1:0] NextToggle;
- reg [1:0] NextToggle;
- always @(Toggle)
  - case (Toggle)
    - 2'b01: NextToggle = 2'b10;
    - 2'b10: NextToggle = 2'b01;
  - endcase
- endmodule
module NextStateLogicFullCase(NextToggle, Toggle);
  input [1:0] Toggle;
  output [1:0] NextToggle;
  reg [1:0] NextToggle;
  always @(Toggle)
    case (Toggle) //synthesis full_case
      2'b01: NextToggle = 2'b10;
      2'b10: NextToggle = 2'b01;
    endcase
  endmodule
Parallel Case

• With casex and casez statements a priority logic is implied.
• With use of *parallel_case* synthesis directive, the priority logic may be avoided.
• This directive should be used with caution as it might lead to a functional mismatch – to be discussed later.
Example

- module PriorityLogic(NextToggle, Toggle);
- input [2:0] Toggle;
- output [2:0] NextToggle;
- reg [2:0] NextToggle;
- always @(Toggle)
  casex (Toggle)
  • 3'bxx1: NextToggle = 3'b010;
  • 3'bx1x: NextToggle = 3'b110;
  • 3'b1xx: NextToggle = 3'b001;
  • default: NextToggle = 3'b000;
  endcase
- endmodule
The Priority Logic

- if (Toggle[0] == 'b1)  
  NextToggle = 3'b010; else
- if (Toggle[1] == 'b1)  
  NextToggle = 3'b110; else
- if (Toggle[2] == 'b1)  
  NextToggle = 3'b001; else
  NextToggle = 3'b000;
Parallel_Case Directive

• module ParallelCase(NextToggle, Toggle);
• input [2:0] Toggle;
• output [2:0] NextToggle;
• reg [2:0] NextToggle;
• always @(Toggle)
  • casex (Toggle) //synthesis parallel_case
    • 3'bxx1: NextToggle = 3'b010;
    • 3'bx1x: NextToggle = 3'b110;
    • 3'b1xx: NextToggle = 3'b001;
    • default: NextToggle = 3'b000;
  • endcase
• endmodule
The logic

- if (Toggle[0] == 'b1)
  NextToggle = 3'b010;
- if (Toggle[1] == 'b1)
  NextToggle = 3'b110;
- if (Toggle[2] == 'b1)
  NextToggle = 3'b001;
- if ((Toggle[0] != 'b1) && (Toggle[1] != 'b1) && (Toggle[2] != 'b1))
  NextToggle = 3'b000;
Questions and Answers

Thank You