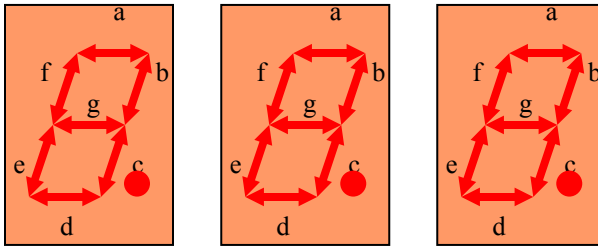


EE 231L

Using Verilog to Implement Functions

For a complex design, it is best to implement well-defined subcircuits and design your circuit by calling these subcircuits. To use a subcircuit in a design, you need to know what it does, and what inputs and outputs it has. You don't really need to know the logic which is used to implement the package.

A seven-segment LED is an array of seven LEDs which can be used to display numbers. A seven-segment LED decoder has four inputs and seven outputs –for an input between 0x0 and 0xF, the outputs will tell which of the segments should be lit to display that number. If you were designing a circuit which used 3 of these decoders, it would be tedious to write this code three times. You could instead use a code that has a subcircuit in a Verilog file:



```

module Altera_Func_Verilog1 (Digits, Lights);
  input [11:0] Digits;
  output [1:21] Lights;

  seg7 digit0 (Digits[3:0], Lights[1:7]);
  seg7 digit1 (Digits[7:4], Lights[8:14]);
  seg7 digit2 (Digits[11:8], Lights[15:21]);

endmodule

module seg7 (bcd, leds);
  input [3:0] bcd;
  output reg [1:7] leds;

  always @ (bcd)
    case (bcd) //abcdefg
      0: leds = 7'b1111110;
      1: leds = 7'b0110000;
      2: leds = 7'b1101101;
      3: leds = 7'b1111001;
      4: leds = 7'b0110011;
      5: leds = 7'b1011011;
      6: leds = 7'b1011111;
      7: leds = 7'b1110000;
      8: leds = 7'b1111111;
      9: leds = 7'b1111011;
      default: leds = 7'bx;
    endcase

endmodule

```

You could also use a function in the Verilog file. Here is a Verilog program which uses a function to implement a design to display a 12-bit binary number on a set of three seven-segment LEDs:

```
module Altera_Func_Verilog1 (Digits, Lights);
    input [11:0] Digits;
    output reg [1:21] Lights;

    function [1:7] leds;
        input [3:0] bcd;
        begin
            case (bcd) // abcdefg
                0: leds = 7'b1111110;
                1: leds = 7'b0110000;
                2: leds = 7'b1101101;
                3: leds = 7'b1111001;
                4: leds = 7'b0110011;
                5: leds = 7'b1011011;
                6: leds = 7'b1011111;
                7: leds = 7'b1110000;
                8: leds = 7'b1111111;
                9: leds = 7'b1111011;
                default: leds = 7'bx;
            endcase
        end
    endfunction

    always @(Digits)
    begin
        Lights[1:7] = leds(Digits[3:0]);
        Lights[8:14] = leds(Digits[7:4]);
        Lights[15:21] = leds(Digits[11:8]);
    end

endmodule
```

You could have Quartus II create your function prototype for you. Open the file `Altera_Func_Verilog1.v` (File – Open), then create an include file for the decoder (File – Create | Update – Create AHDL Include Files for Current File).

You could also use your decoder in a graphics design file. Open the file `Altera_Func_Verilog.v` (File – Open), then create a symbol for the decoder (File – Create Default Symbol). This will make the file `Altera_Func_Verilog.sym` which you can then place in a graphics design file. This is the display as a graphics design file:

