Logic Design using FPGAs
Assignment
BCD and Seven Segment Display

Objective: Behavioral Models in Verilog - Designing combinational circuits that can perform binary-to-decimal number conversion.

Part (1): Seven Segment Display (Book example page 170 & 171)

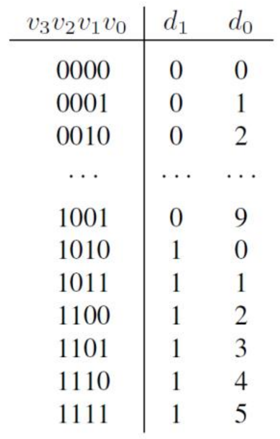
We wish to display on 7-segment display the values set by 4 input switches. Your circuit should be able to display the digits from 0 to 9 and should treat the values 1010 to 1111 as BLANK display (all OFF). The LEDs for the 7-segment display are all active low (common anode), which means that the LED would turn ON when its assigned bit value is a logic ‘0’.

Create a design that assigns the correct output bits to the 7-segments based on the 4- bit input value. Use parameters for the 10 different decimal digits (0 to 9) and the BLANK case.

```
parameter   BLANK   = 7’b_1111_1111;
parameter   ZERO    = 7’b_0000_0001;
```

Part (2): Binary-Coded Decimal

You are to design a circuit that converts a four-bit binary number **V = v_3 v_2 v_1 v_0** into its two-digit decimal equivalent **D = d_1d_0**. Table 1 shows the required output values.



$v_3v_2v_1v_0$	d_1	d_0
0000	0	0
0001	0	1
0010	0	2
...	...	...
1001	0	9
1010	1	0
1011	1	1
1100	1	2
1101	1	3
1110	1	4
1111	1	5

Table 1: Binary-to-decimal conversion values.

Structural Design:

It includes a comparator that checks when the value of V is greater than 9 and uses the output of this comparator in the control of the 7-segment displays. You can use the conditional and comparison operators.

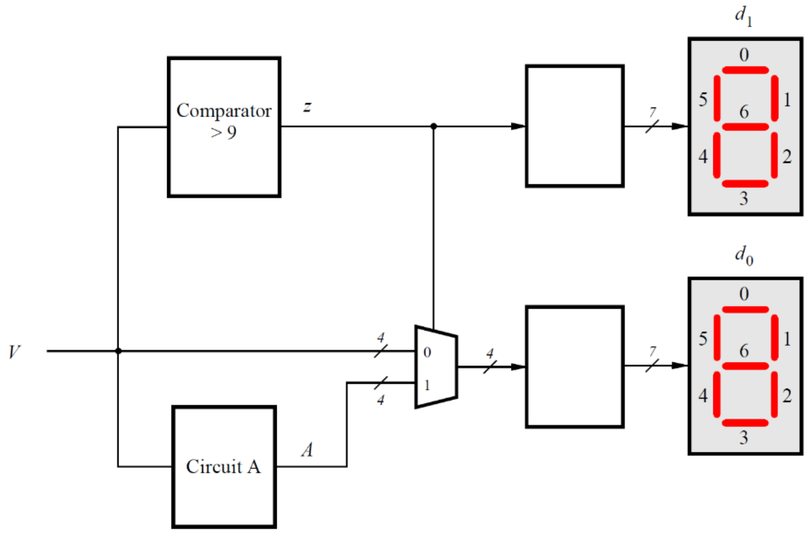


Figure 1: Partial design of the binary-to-decimal conversion circuit.

Notice that the circuit in Figure 1 includes a 4-bit wide 2-to-1 multiplexer. The purpose of this multiplexer is to drive digit d_0 with the value of V when z = 0, and the value of A when z = 1. To design circuit A consider the following. For the input values V < 9, the circuit A does not matter, because the multiplexer in Figure 1 just selects V in these cases. But for the input values V > 9, the multiplexer will select A. Thus, A has to provide output values that properly implement Table 1 when V > 9.
You need to design circuit A so that the input V = 1010 gives an output A = 0000, the input V = 1011 gives the output A = 0001, and the input V = 1111 gives the output A = 0101. Design circuit A using a case structure.

The top code module should instantiate the needed submodules with correct interconnections.

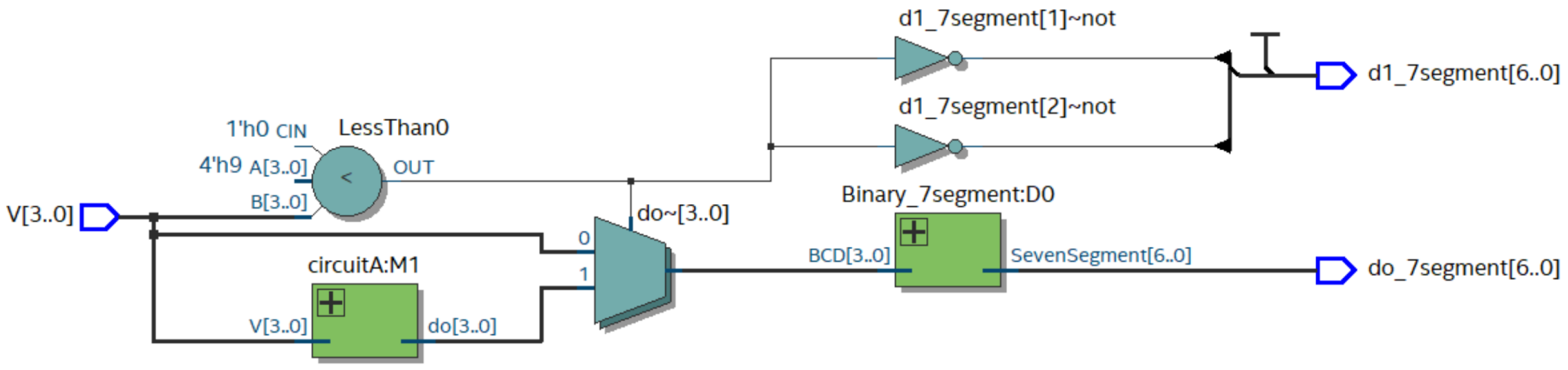
EE 417 - Assignment 3 – Problem 3 – BCD – Seven Segment Display

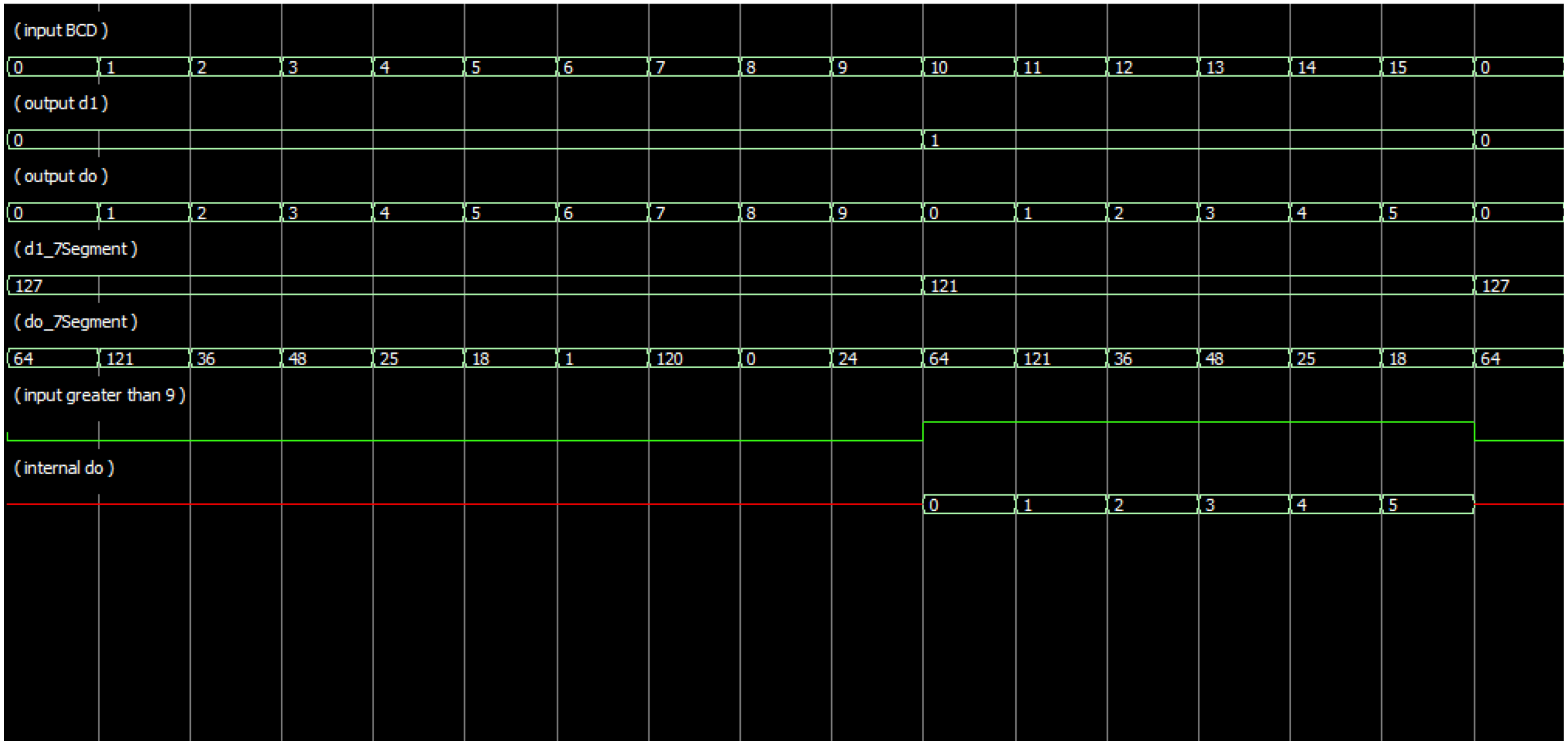
```
module BCD_SevenSegment (V, do_7segment, d1_7segment);

input     [3:0]   V;
output    [6:0]   do_7segment, d1_7segment;
wire      [3:0]   digit0, do, d1;
wire              V_grtr9;

parameter BLANK = 7'b111_1111;  // 127
parameter ONE   = 7'b111_1001;  // 121

assign V_grtr9     = (V > 4'b1001);
assign d1_7segment = V_grtr9 ? ONE     : BLANK;
assign d1          = V_grtr9 ? 4'b0001: 4'b0000;
assign do          = V_grtr9 ? digit0 : V;

circuitA        M1  (V, digit0)  ;
Binary_7segment DO  (do, do_7segment);

endmodule
```

```
module circuitA (V, do) ;
input       [3:0]   V;
output reg [3:0]    do;

always @ *
   case (V)
        4'b1010 : do = 4'b0000;
        4'b1011 : do = 4'b0001;
        4'b1100 : do = 4'b0010;
        4'b1101 : do = 4'b0011;
        4'b1110 : do = 4'b0100;
        4'b1111 : do = 4'b0101;
        default : do = 4'bxxxx;
        endcase
endmodule
```

```
module Binary_7segment (BCD, SevenSegment);
input        [3:0]  BCD;
output reg [6:0]  SevenSegment;

parameter BLANK = 7'b111_1111;  // 127
parameter ZERO  = 7'b100_0000;  //  64
parameter ONE   = 7'b111_1001;  // 121
parameter TWO   = 7'b010_0100;  //  36
parameter THREE = 7'b011_0000;  //  48
parameter FOUR  = 7'b001_1001;  //  25
parameter FIVE  = 7'b001_0010;  //  18
parameter SIX   = 7'b000_0001;  //   1
parameter SEVEN = 7'b111_1000;  // 120
parameter EIGHT = 7'b000_0000;  //   0
parameter NINE  = 7'b001_1000;  //  24

always @ *
case (BCD)
    4'b0000 : SevenSegment = ZERO;
    4'b0001 : SevenSegment = ONE;
    4'b0010 : SevenSegment = TWO;
    4'b0011 : SevenSegment = THREE;
    4'b0100 : SevenSegment = FOUR;
    4'b0101 : SevenSegment = FIVE;
    4'b0110 : SevenSegment = SIX;
    4'b0111 : SevenSegment = SEVEN;
    4'b1000 : SevenSegment = EIGHT;
    4'b1001 : SevenSegment = NINE;
    default : SevenSegment = BLANK;
    endcase

endmodule
```

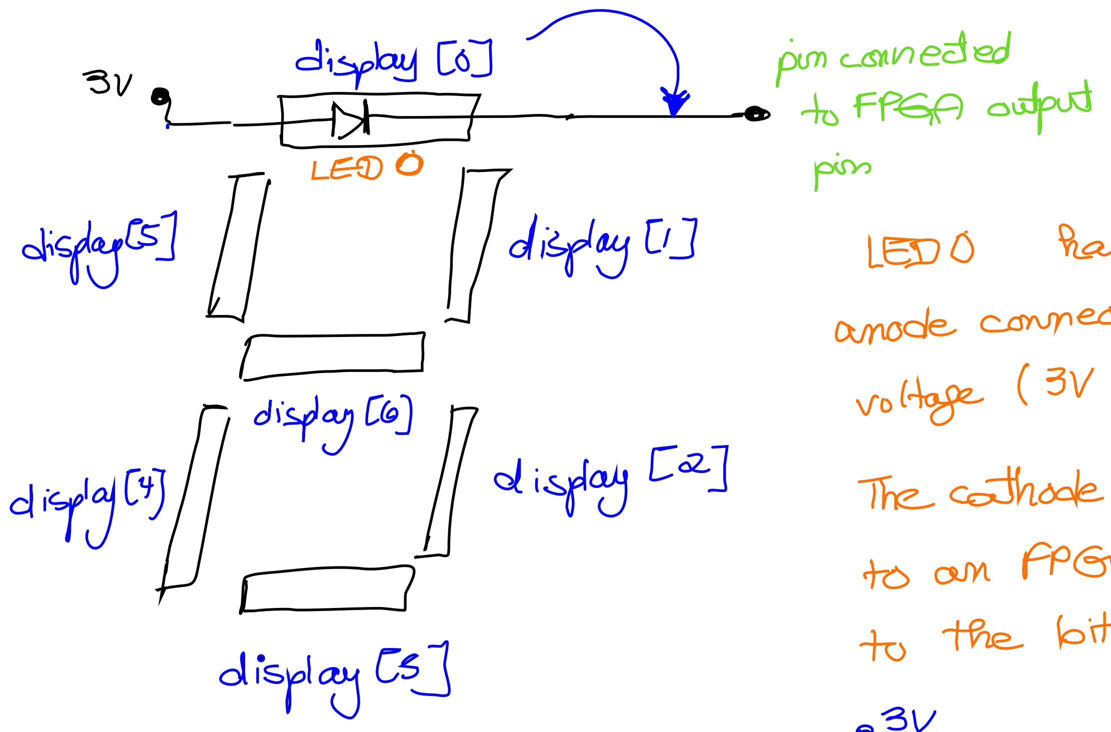
```
module BCD_SevenSegment_tb ();

reg  [3:0]   BCD;
wire [6:0]   do_7segment, d1_7segment;
wire [3:0]   do, d1;
wire         V_grtr9;
wire [3:0]   digit0;

BCD_SevenSegment UUT  (BCD, do_7segment, d1_7segment);

assign V_grtr9 = UUT.V_grtr9;
assign do      = UUT.do;
assign d1      = UUT.d1;
assign digit0  = UUT.digit0;

initial
   begin
   BCD = 4'b0000;
   forever
     begin
     #10 BCD = BCD + 1;
     end
   end

endmodule
```





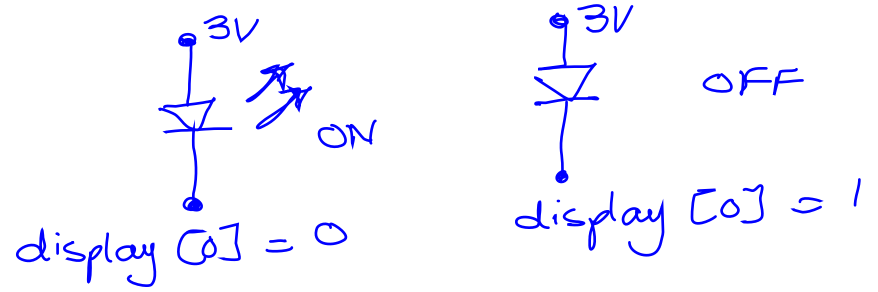
```
BLANK = 7'b111_1111;   // 127
ZERO  = 7'b100_0000;   //  64
ONE   = 7'b111_1001;   // 121
TWO   = 7'b010_0100;   //  36
THREE = 7'b011_0000;   //  48
FOUR  = 7'b001_1001;   //  25
FIVE  = 7'b001_0010;   //  18
SIX   = 7'b000_0001;   //   1
SEVEN = 7'b111_1000;   // 120
EIGHT = 7'b000_0000;   //   0
NINE  = 7'b001_1000;   //  24
```

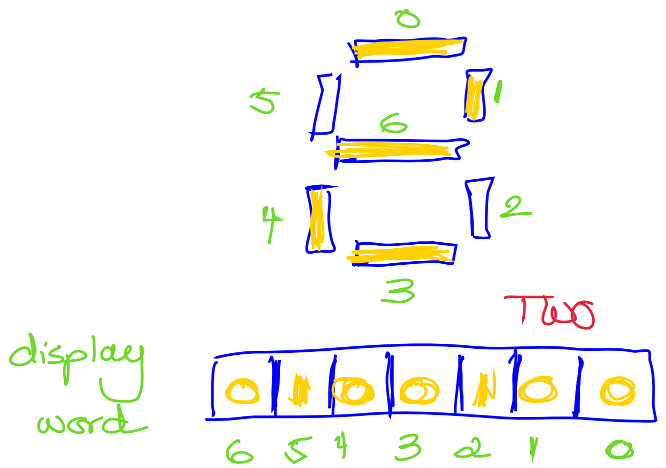
7-segment display

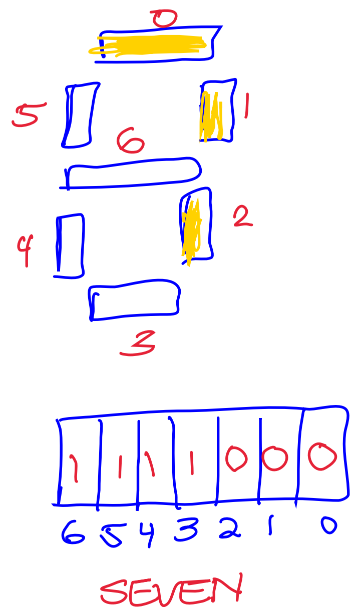


LED 0 has its anode connected to high voltage (3V for example)

The cathode is connected to an FPGA pin assigned to the bit display [0].







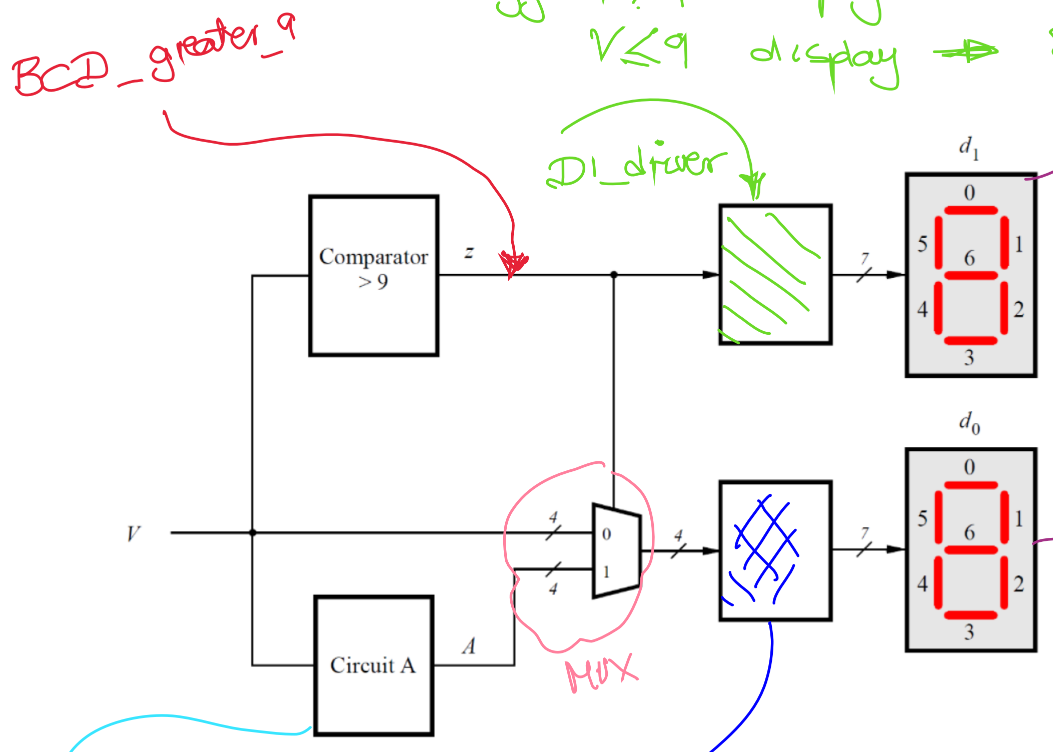
If the 7-segment display is blank and all LEDs will be off, then

display [6:0] = 7'b111_1111

If the 7 segment display should show a digit zero, then all LEDs should be ON except LED6.

display [6:0] = 7'b100_0000

If V > 9 display → ONE
V ≤ 9 display → Blank



Displaying the ten's digit
For 4 bit inputs the values range goes from 0 → 15, hence d1 will be either blank or 1.

Displaying the one's digit

BCD_to_Seven

module that takes 4 bit word representing values from 0 → 9 and finds the corresponding 7 bit display word that will be driving the 7-segment display.

In case V > 9 module A should find the correct value for d0 equivalent binary value

V in decimal	d1
10	0
11	1
12	2
13	3
14	4
15	5

Binary_in	Decimal	display
0000	0	100_0000
0001	1	111_1001
0010	2	010_0100
0011	3	011_0000
⋮	⋮	
1001	9	0011000

```
module D1_driver ( Vgreaterthan9_in , display d1_out ) ;
input    Vgreaterthan9_in ;
output [6:0]   displayd1_out ;
  parameter   Blank  = 7'b111_1111 ;   // Giving special numbers a name to
  parameter   ONE    = 7'b111_1001 ;   // make the code more readable.

  // Conditional operator
  assign displayd1_out = (Vgreaterthan9_in ?) ONE : Blank ;
endmodule
```

```
// Another way to design D1_driver is by using a case structure
module D1_driver ( Vgreaterthan9_in , displayd1_out ) ;
input    Vgreaterthan9_in ;
output reg [6:0] display1_out ;   // If assigned a value within case structure then
                                  // it has to be defined as register.
 always @ ( Vgreaterthan9_in )
       case ( Vgreaterthan9_in )
 1'b0 :   display1_out  = 7'b111_1111 ;   // or use parameters as we
 1'b1 :   displayd1_out = 7'b111_1001 ;   //    did before ↑
 default : displayd1_out = 7'b111_1111 ;   endcase    endmodule
```

Circuit A will have an effect if the input word V is greater than 9.

```
module circuitA ( V, A ) ;
input [3:0] V ;
output reg [3:0] A ;   // Need to define A as a reg to assign it values
                       //    within the always block.

always @ (V)
      case (V)
4'b1010   :  A = 4'b0000 ;      // 10
4'b1011   :  A = 4'b0001 ;      // 11
4'b1100   :  A = 4'b0010 ;      // 12
4'b1101   :  A = 4'b0011 ;      // 13
4'b1110   :  A = 4'b0100 ;      // 14
4'b1111   :  A = 4'b0101 ;      // 15
  default :  A = 4'bxxxx ;
      endcase
endmodule
```

We could have also designed the module circuit using combinational logic with K-map for every binary output bit

For A_2 :

$V_3V_2V_1V_0$	$A_3A_2A_1A_0$
1010	0000
1011	0001
1100	0010
1101	0011
1110	0100
1111	0101

V_1V_0 \ V_3V_2	00	01	11	10
00	X 0	X 4	0 12	X 8
01	X 1	X 5	0 13	X 9
11	X 3	X 7	1 15	0 10
10	X 2	X 6	1 14	0 11

all the values less than or equal to 9 are don't cares.

$A_2 = V_2V_1$

```
assign A[3] = 1'b0 ;
assign A[2] = V[2] & V[1] ;

assign A[1] = V[2] & (~V[1]) ;
assign A[0] = V[0] ;
```

} obtained using K-map for A_1 and another Kmap for A_0.

```
module circuitA_combinational (V, A);

  input [3:0] V;
  output [3:0] A;     // by default outputs are wires

  assign A[3] = 1'b0;
  assign A[2] = V[2] & V[1];
  assign A[1] = V[2] & (~V[1]);
  assign A[0] = V[0];

endmodule
```