ES204 Digital Systems
LAB Assignment - 1

Indian Institute of Technology, Gandhinagar
January 10, 2024
Time & Venue: Tuesday 8:30-9:50am [7/108]

Submission deadline: Jan 13, 2024
Marks : 40

Submission instructions:

(a) Only one student from the team will submit with the word doc name Rollno1_Rollno2.pdf. The PDF will contain the code, testbench and simulation results.
(b) Make a tar-ball / Zip of the project and upload.

For each of the questions, write a Verilog code. You also need to create a **testbench** and show the simulation results.

Smart Math Tutor Hardware

You are to design a hardware that can be used by Kindergarten kids to understand multiples of different numbers.

Write a Verilog code that can implement multiples of 2,3, 4, 5,6, 7, 8, 9. Allow the user to decide the number whose multiple s/he wants to find out. You can use "*select line"* to select the number whose multiple has to be found.

You need to do this for 5-bit input (which decides the range within which all the multiples have to be found.)

Module has 5-bit input, select inputs and ONE output signal. (Don't write separate Verilog codes for each number whose multiples are to be found.)

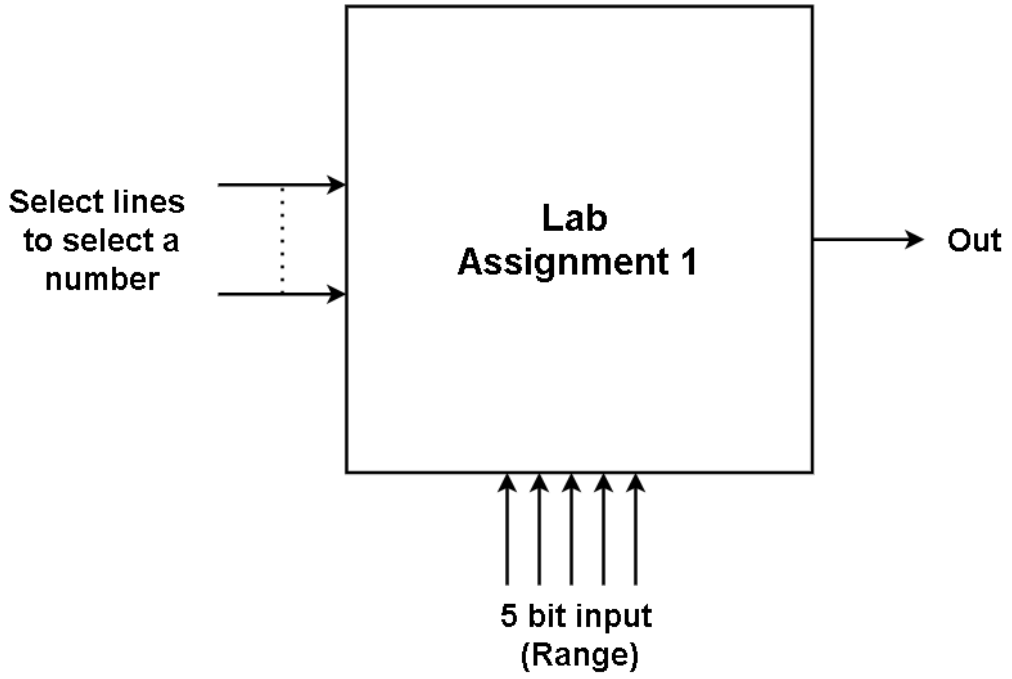
Write the code using **always** block.

Next, assume that you have a large 5x32 decoder. Implement the same question using only decoder and OR gates. You can write the code using any of structural/continuous assignment/procedural coding.

Compare and comment on the simplicity of the two approaches.

ES204 Digital Systems
Appendix for Lab 1 Assignment

- PFA **overview block diagram/schematic** of the module to be designed which executes the functionality dependent on the input signal provided.



Example:
Suppose a number 4 is selected then the “Out” signal goes high for 4, 8, 12 ... etc in corresponding to the range provided by 5-bit input. Similarly if number 6 is selected “Out” goes high for 6,12,18 if the input specified by 5-bit input signal is as 5’b10110 etc.

Lab Assignment - 1

Teammates:
Birudugadda Srivibhav (22110050)
Reddybathuni Venkat (22110220)

Select lines Notation:
3b’000 corresponds to 2
3b’001 corresponds to 3
3b’010 corresponds to 4
3b’011 corresponds to 5
3b’100 corresponds to 6
3b’101 corresponds to 7
3b’110 corresponds to 8
3b’111 corresponds to 9

Question - 1(Using always block)
Code:

```
module question1a(
   input [2:0] select,
   input [0:4] number,
   output reg OUT
   );
always@(*)
begin
   if(select == 3'b000 && number[4] == 0)
   begin
      OUT =1;
   end
   else if(select == 3'b001 && (number == 5'b00000 | number == 5'b11000 |number ==
5'b10010 | number == 5'b01100 | number == 5'b01001 | number == 5'b00110 | number ==
5'b00011 | number == 5'b10101 | number == 5'b11110 | number == 5'b11011 | number ==
5'b01111))
   begin
      OUT =1;
   end
   else if(select == 3'b010 && number[3:4] == 00)
   begin
      OUT =1;
   end
   else if(select == 3'b011 && (number==5'b00000 | number==5'b10100 |
number==5'b01010 | number==5'b00101 | number==5'b11001 | number==5'b11110 |
number==5'b01111))
   begin
      OUT =1;
   end
```

```
   else if( select == 3'b100 && ( number==5'b00000 | number==5'b11000 |
number==number==5'b00110 | number==5'b01100 | 5'b10010 | number==5'b11110 ))
   begin
      OUT =1;
   end
   else if (select== 3'b101 &( number==5'b00000 | number==5'b01110 | number==5'b10101 |
number==5'b11100 | number==5'b00111  ))
   begin
      OUT = 1;
   end
   else if (select == 3'b110 & number[2:4] == 000)
   begin
      OUT = 1;
   end
   else if (select == 3'b111 &( number==5'b00000 | number==5'b10010 | number==5'b01001
| number==5'b11011 ))
   begin
     OUT = 1;
   end
   else
   begin
     OUT = 0;
   end
end
endmodule
```

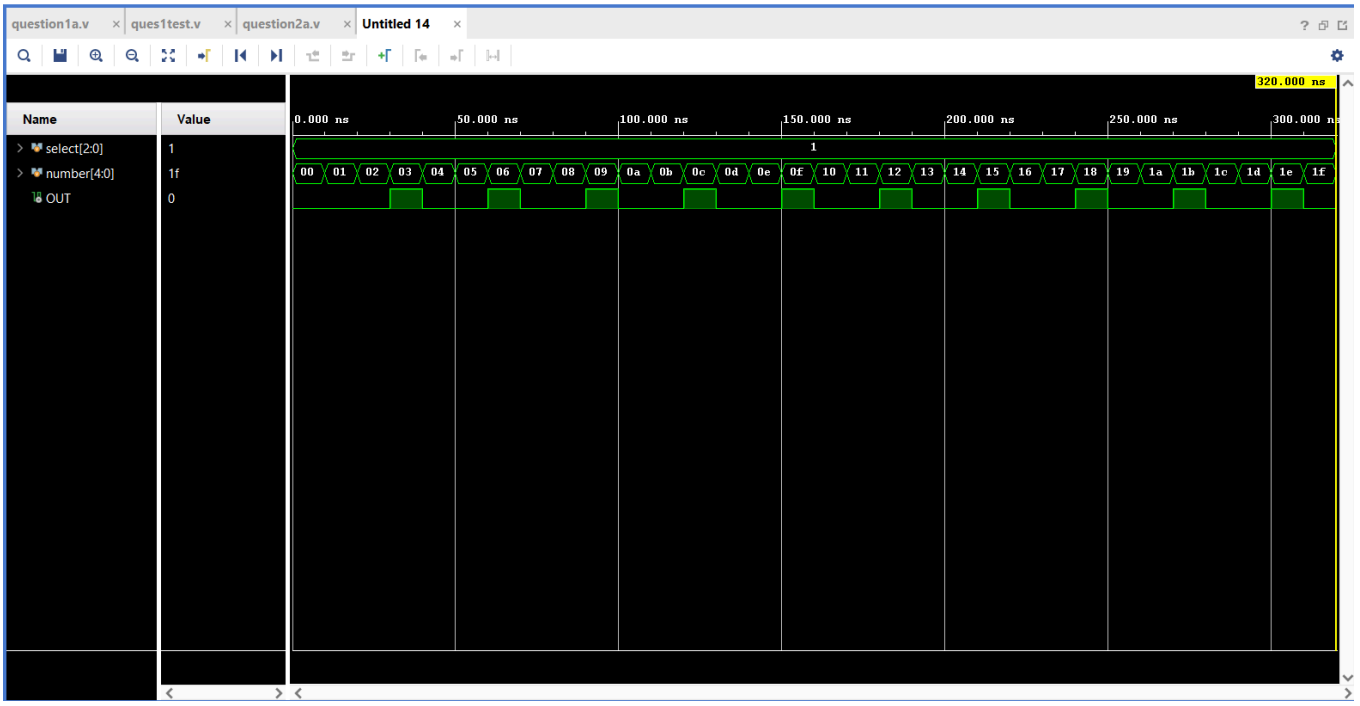
Testbench code (checked for 3 multiples):

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 12.01.2024 13:22:19
// Design Name:
// Module Name: ques1test
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////
```

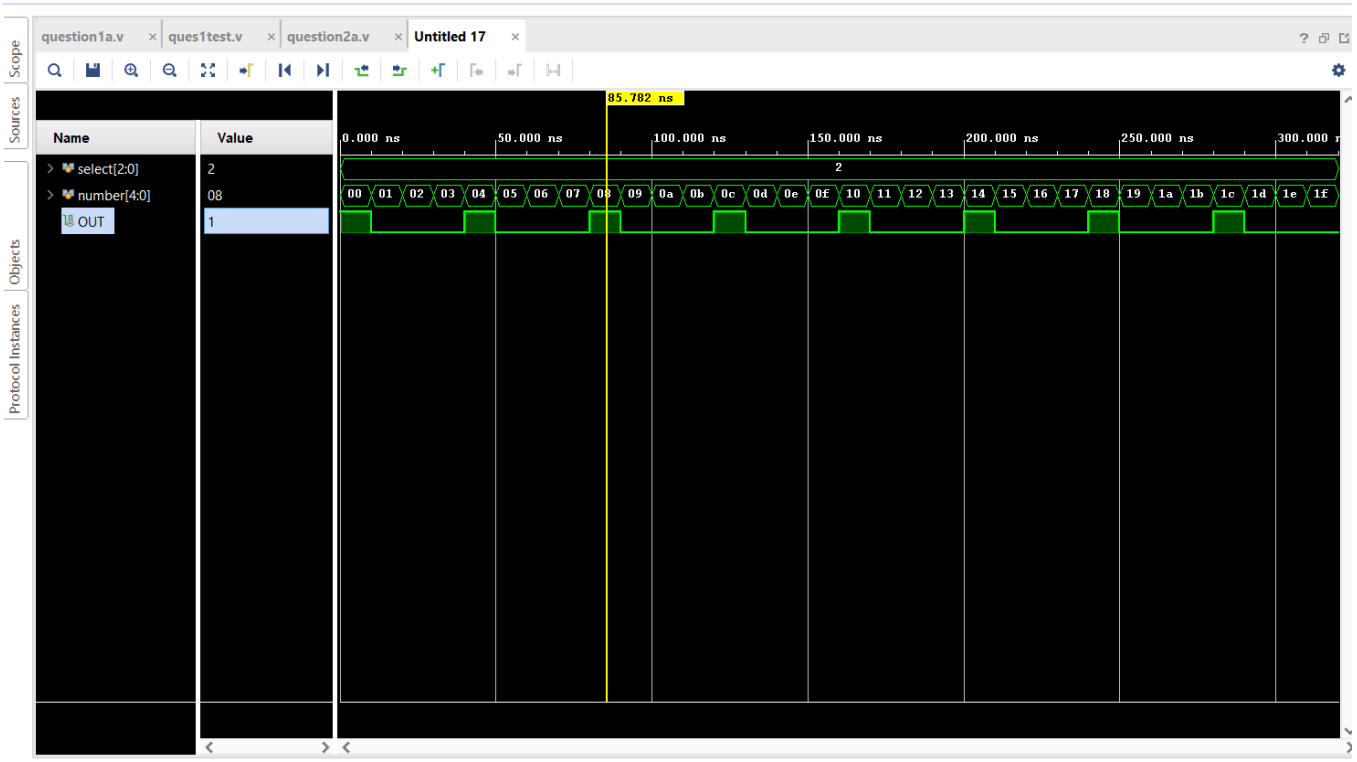
```
module ques1test();
    reg [2:0]select;
    reg [4:0] number;
    wire OUT;
    question1a uut(.select(select),.number(number),.OUT(OUT));
    initial
    begin
    number = 5'b00000; select = 3'b001; #10;
    number = 5'b00001; select = 3'b001; #10;
    number = 5'b00010; select = 3'b001;#10;
    number = 5'b00011; select = 3'b001; #10;
    number = 5'b00100; select = 3'b001; #10;
    number = 5'b00101; select = 3'b001; #10;
    number = 5'b00110; select = 3'b001; #10;
    number = 5'b00111; select = 3'b001; #10;
    number = 5'b01000; select = 3'b001; #10;
    number = 5'b01001; select = 3'b001; #10;
    number = 5'b01010; select = 3'b001; #10;
    number = 5'b01011; select = 3'b001; #10;
    number = 5'b01100; select = 3'b001; #10;
    number = 5'b01101; select = 3'b001; #10;
    number = 5'b01110; select = 3'b001; #10;
    number = 5'b01111; select = 3'b001; #10;
    number = 5'b10000; select = 3'b001; #10;
    number = 5'b10001; select = 3'b001; #10;
    number = 5'b10010; select = 3'b001; #10;
    number = 5'b10011; select = 3'b001; #10;
    number = 5'b10100; select = 3'b001; #10;
    number = 5'b10101; select = 3'b001; #10;
    number = 5'b10110; select = 3'b001; #10;
    number = 5'b10111; select = 3'b001; #10;
    number = 5'b11000; select = 3'b001; #10;
    number = 5'b11001; select = 3'b001; #10;
    number = 5'b11010; select = 3'b001; #10;
    number = 5'b11011; select = 3'b001; #10;
    number = 5'b11100; select = 3'b001; #10;
    number = 5'b11101; select = 3'b001; #10;
    number = 5'b11110; select = 3'b001; #10;
    number = 5'b11111; select = 3'b001; #10;
    $finish();
    end
endmodule
```

Simulation Results:

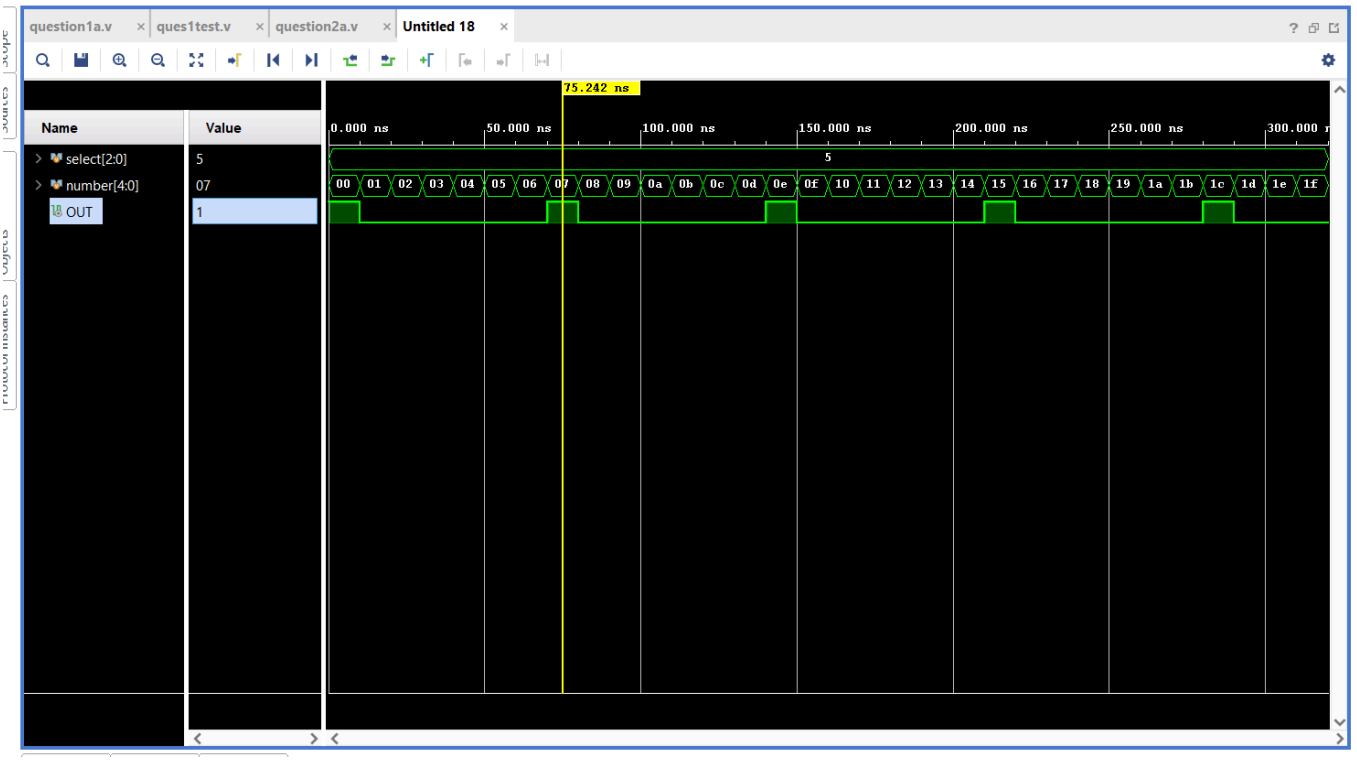
1) Checked for 3 multiples



2) Checked for 4 multiples



3) Checked for 7 multiples



Question - 2 (Using the 5*32 Decoder)
Code:

```
module question2a(
   input [2:0] select,
   input [0:4] number,
   output reg OUT
);

  reg [0:31] O;

  always @* begin
    O[0]  = (~number[0]  & ~number[1]  & ~number[2]  & ~number[3]  & ~number[4]);
    O[1]  = (~number[0]  & ~number[1]  & ~number[2]  & ~number[3]  &  number[4]);
    O[2]  = (~number[0]  & ~number[1]  & ~number[2]  &  number[3]  & ~number[4]);
    O[3]  = (~number[0]  & ~number[1]  & ~number[2]  &  number[3]  &  number[4]);
    O[4]  = (~number[0]  & ~number[1]  &  number[2]  & ~number[3]  & ~number[4]);
    O[5]  = (~number[0]  & ~number[1]  &  number[2]  & ~number[3]  &  number[4]);
    O[6]  = (~number[0]  & ~number[1]  &  number[2]  &  number[3]  & ~number[4]);
    O[7]  = (~number[0]  & ~number[1]  &  number[2]  &  number[3]  &  number[4]);
    O[8]  = (~number[0]  &  number[1]  & ~number[2]  & ~number[3]  & ~number[4]);
    O[9]  = (~number[0]  &  number[1]  & ~number[2]  & ~number[3]  &  number[4]);
    O[10] = (~number[0]  &  number[1]  & ~number[2]  &  number[3]  & ~number[4]);
    O[11] = (~number[0]  &  number[1]  & ~number[2]  &  number[3]  &  number[4]);
    O[12] = (~number[0]  &  number[1]  &  number[2]  & ~number[3]  & ~number[4]);
    O[13] = (~number[0]  &  number[1]  &  number[2]  & ~number[3]  &  number[4]);
    O[14] = (~number[0]  &  number[1]  &  number[2]  &  number[3]  & ~number[4]);
    O[15] = (~number[0]  &  number[1]  &  number[2]  &  number[3]  &  number[4]);
    O[16] = ( number[0]  & ~number[1]  & ~number[2]  & ~number[3]  & ~number[4]);
    O[17] = ( number[0]  & ~number[1]  & ~number[2]  & ~number[3]  &  number[4]);
    O[18] = ( number[0]  & ~number[1]  & ~number[2]  &  number[3]  & ~number[4]);
    O[19] = ( number[0]  & ~number[1]  & ~number[2]  &  number[3]  &  number[4]);
    O[20] = ( number[0]  & ~number[1]  &  number[2]  & ~number[3]  & ~number[4]);
    O[21] = ( number[0]  & ~number[1]  &  number[2]  & ~number[3]  &  number[4]);
    O[22] = ( number[0]  & ~number[1]  &  number[2]  &  number[3]  & ~number[4]);
    O[23] = ( number[0]  & ~number[1]  &  number[2]  &  number[3]  &  number[4]);
    O[24] = ( number[0]  &  number[1]  & ~number[2]  & ~number[3]  & ~number[4]);
    O[25] = ( number[0]  &  number[1]  & ~number[2]  & ~number[3]  &  number[4]);
    O[26] = ( number[0]  &  number[1]  & ~number[2]  &  number[3]  & ~number[4]);
    O[27] = ( number[0]  &  number[1]  & ~number[2]  &  number[3]  &  number[4]);
    O[28] = ( number[0]  &  number[1]  &  number[2]  & ~number[3]  & ~number[4]);
    O[29] = ( number[0]  &  number[1]  &  number[2]  & ~number[3]  &  number[4]);
    O[30] = ( number[0]  &  number[1]  &  number[2]  &  number[3]  & ~number[4]);
    O[31] = ( number[0]  &  number[1]  &  number[2]  &  number[3]  &  number[4]);

    case (select)
```

```
    3'b000: OUT = (O[0] | O[2] | O[4] | O[6] | O[8] | O[10] | O[12] | O[14] | O[16] | O[18] |
O[20] | O[22] | O[24] | O[26] | O[28] | O[30]);
    3'b001: OUT = (O[0] | O[3] | O[6] | O[9] | O[12] | O[15] | O[18] | O[21] | O[24] | O[27] |
O[30]);
    3'b010: OUT = (O[0] | O[4] | O[8] | O[12] | O[16] | O[20] | O[24] | O[28]);
    3'b011: OUT = (O[0] | O[5] | O[10] | O[15] | O[20] | O[25] | O[30]);
    3'b100: OUT = (O[0] | O[6] | O[12] | O[18] | O[24] | O[30]);
    3'b101: OUT = (O[0] | O[7] | O[14] | O[21] | O[28]);
    3'b110: OUT = (O[0] | O[8] | O[16] | O[24]);
    3'b111: OUT = (O[0] | O[9] | O[18] | O[27]);
    default: OUT = 0; // Default case when select is not 3'b000 - 3'b111
  endcase
 end

endmodule
```

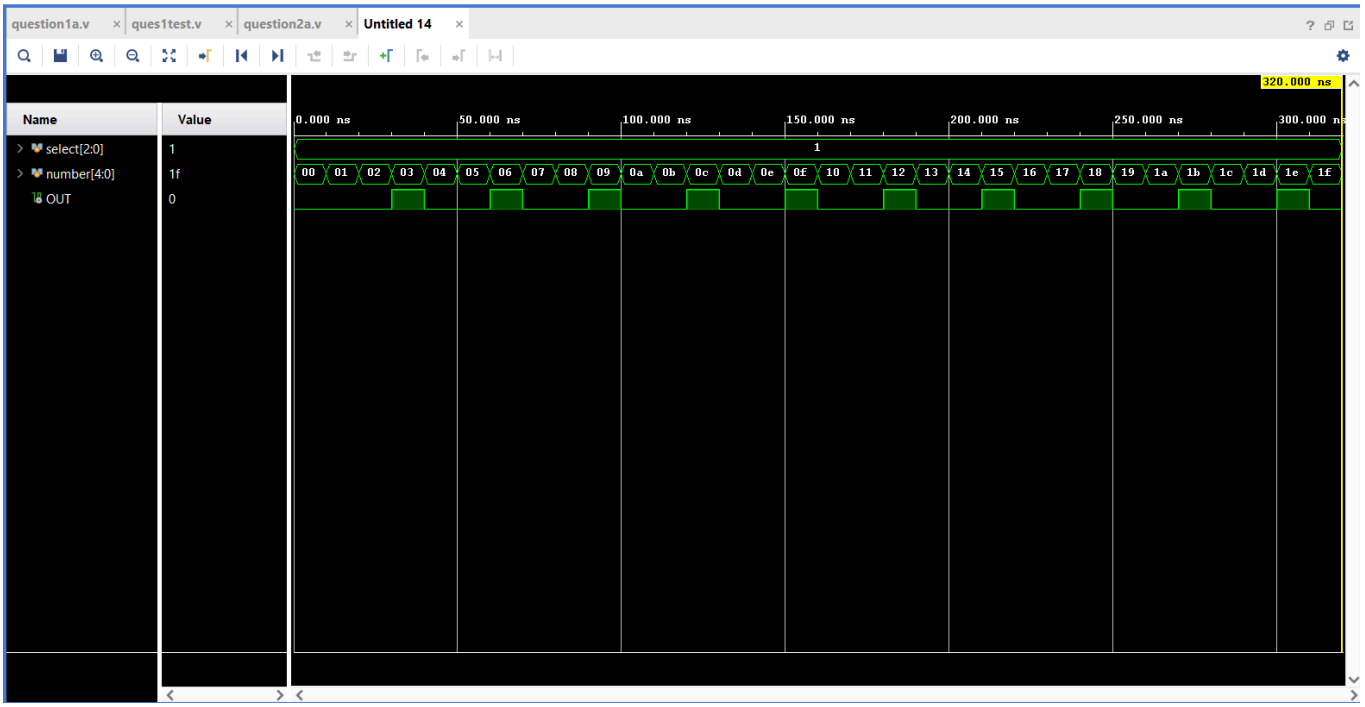
Testbench Code (checked for 3 multiples):

```
module ques1test();
  reg [2:0]select;
  reg [4:0] number;
  wire OUT;
  question1a uut(.select(select),.number(number),.OUT(OUT));
  initial
  begin
  number = 5'b00000; select = 3'b001; #10;
  number = 5'b00001; select = 3'b001; #10;
  number = 5'b00010; select = 3'b001; #10;
  number = 5'b00011; select = 3'b001; #10;
  number = 5'b00100; select = 3'b001; #10;
  number = 5'b00101; select = 3'b001; #10;
  number = 5'b00110; select = 3'b001; #10;
  number = 5'b00111; select = 3'b001; #10;
  number = 5'b01000; select = 3'b001; #10;
  number = 5'b01001; select = 3'b001; #10;
  number = 5'b01010; select = 3'b001; #10;
  number = 5'b01011; select = 3'b001; #10;
  number = 5'b01100; select = 3'b001; #10;
  number = 5'b01101; select = 3'b001; #10;
  number = 5'b01110; select = 3'b001; #10;
  number = 5'b01111; select = 3'b001; #10;
  number = 5'b10000; select = 3'b001; #10;
  number = 5'b10001; select = 3'b001; #10;
  number = 5'b10010; select = 3'b001; #10;
  number = 5'b10011; select = 3'b001; #10;
  number = 5'b10100; select = 3'b001; #10;
  number = 5'b10101; select = 3'b001; #10;
  number = 5'b10110; select = 3'b001; #10;
  number = 5'b10111; select = 3'b001; #10;
```

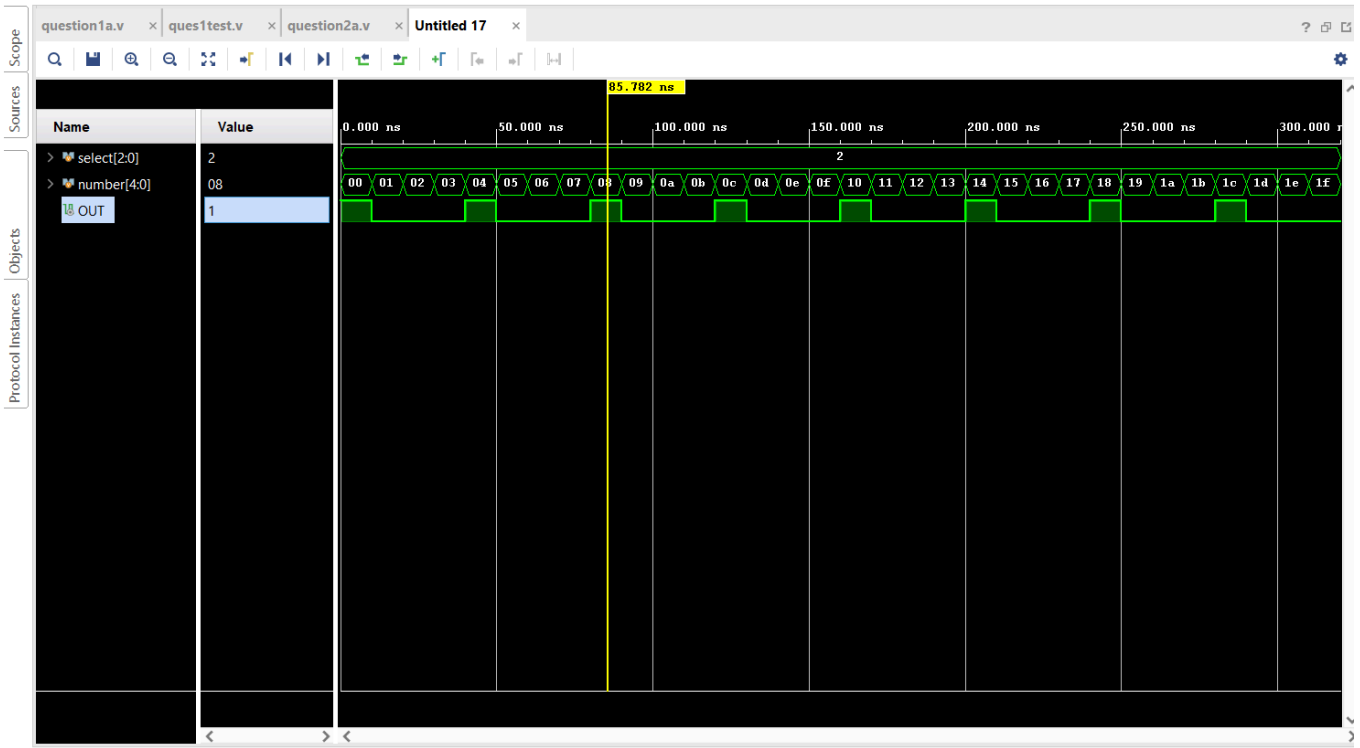
```
    number = 5'b11000; select = 3'b001; #10;
    number = 5'b11001; select = 3'b001; #10;
    number = 5'b11010; select = 3'b001; #10;
    number = 5'b11011; select = 3'b001; #10;
    number = 5'b11100; select = 3'b001; #10;
    number = 5'b11101; select = 3'b001; #10;
    number = 5'b11110; select = 3'b001; #10;
    number = 5'b11111; select = 3'b001; #10;
    $finish();
    end
endmodule
```

Simulation results:

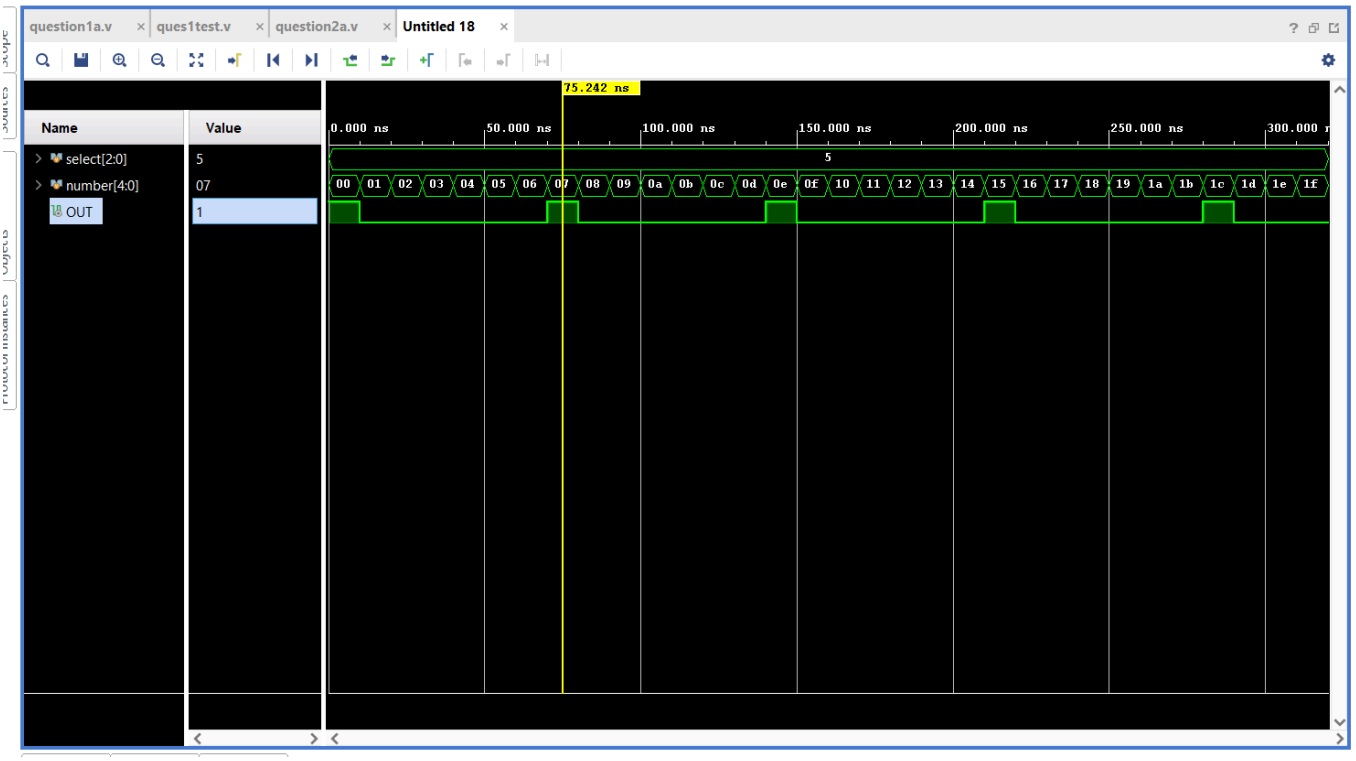
1) **Checked for 3 multiples**



2) Checked for 4 multiples



3) Checked for 7 multiples



Observation:

The 5*32 Decoder implementation is more straightforward than the first one because, in the first implementation, we have to find the k-maps of 5-bit numbers for all select values which is a tedious task.