ASSIGNMENT - 1

DIGITAL SYSTEMS ES204

Guntas Singh Saran
guntassingh.saran@iitgn.ac.in
22110089

Nihar Shah
nihar.shah@iitgn.ac.in
22110237

PART - 1

SEL is a 3-bit input parameter with the following specifications:

```
  SEL   -> MULTIPLES OF
3'b000  ->  2
3'b001  ->  3
3'b010  ->  4
3'b011  ->  5
3'b100  ->  6
3'b101  ->  7
3'b110  ->  8
3'b111  ->  9
```

A is a 5-bit input parameter ranging from `5'b00000` till `5'b11111.`
The `module multiple(A, SEL, OUT)` takes in **A** and **SEL** as input and produces a binary output **OUT** which is **1** if **A** is a multiple of **SEL** and **0** otherwise.

Thus, for a 5-bit number **A** and a given **SEL,** we have to map the 5 bits of **A** such that they correspond to one of the multiples of **SEL**.
This was achieved using K-Maps.
For Example:

- For **SEL** = `3'b000` (Multiples of **2** chosen) and **A** = $ABCDE$, the boolean logic would be:

  OUT = $\overline{E}$.

- Similarly for **SEL** = `3'b011` (Multiples of **5** chosen) and **A** = $ABCDE$, the boolean logic would be:

  OUT = $(\overline{A}\overline{B}C\overline{D}E)+(\overline{A}B\overline{C}D\overline{E})+(\overline{A}BCDE)+(A\overline{B}C\overline{D}\overline{E})+(AB\overline{C}\overline{D}E)+(ABCD\overline{E})$

  Each of these six terms correspond to choosing **either** of the six multiples of 5 within 5-bit range:

```
5'b00101 -> 5
5'b01010 -> 10
5'b01111 -> 15
5'b10100 -> 20
5'b11001 -> 25
5'b11110 -> 30
```

VERILOG CODE

```
module multiple(A, SEL, OUT);

input [4:0] A;
input [2:0] SEL;
output reg OUT;

always @(*) begin
   case(SEL)

       0 : OUT = (~A[0]);

       1 : OUT = (~A[4] & ~A[3] & ~A[2] & A[1] & A[0]) |
                 (~A[4] & ~A[3] & A[2] & A[1] & ~A[0]) |
                 (~A[4] & A[3] & ~A[2] & ~A[1] & A[0]) |
                 (~A[4] & A[3] & A[2] & ~A[1] & ~A[0]) |
                   (~A[4] & A[3] & A[2] & A[1] & A[0]) |
                 (A[4] & ~A[3] & ~A[2] & A[1] & ~A[0]) |
                  (A[4] & ~A[3] & A[2] & ~A[1] & A[0]) |
                 (A[4] & A[3] & ~A[2] & ~A[1] & ~A[0]) |
                   (A[4] & A[3] & ~A[2] & A[1] & A[0]) |
                   (A[4] & A[3] & A[2] & A[1] & ~A[0]);

       2 : OUT = (~A[0] & ~A[1]);

       3 : OUT = (~A[4] & ~A[3] & A[2] & ~A[1] & A[0]) |
                 (~A[4] & A[3] & ~A[2] & A[1] & ~A[0]) |
                   (~A[4] & A[3] & A[2] & A[1] & A[0]) |
                 (A[4] & ~A[3] & A[2] & ~A[1] & ~A[0]) |
                  (A[4] & A[3] & ~A[2] & ~A[1] & A[0]) |
                   (A[4] & A[3] & A[2] & A[1] & ~A[0]);

       4 : OUT = (~A[4] & ~A[3] & A[2] & A[1] & ~A[0]) |
                 (~A[4] & A[3] & A[2] & ~A[1] & ~A[0]) |
                 (A[4] & ~A[3] & ~A[2] & A[1] & ~A[0]) |
                 (A[4] & A[3] & ~A[2] & ~A[1] & ~A[0]) |
                   (A[4] & A[3] & A[2] & A[1] & ~A[0]);

       5 : OUT = (~A[4] & ~A[3] & A[2] & A[1] & A[0]) |
                 (~A[4] & A[3] & A[2] & A[1] & ~A[0]) |
                 (A[4] & ~A[3] & A[2] & ~A[1] & A[0]) |
                 (A[4] & A[3] & A[2] & ~A[1] & ~A[0]);
```

```
        6 : OUT = (~A[0] & ~A[1] & ~A[2]);

        7 : OUT = (~A[4] & A[3] & ~A[2] & ~A[1] & A[0]) |
                  (A[4] & ~A[3] & ~A[2] & A[1] & ~A[0]) |
                    (A[4] & A[3] & ~A[2] & A[1] & A[0]);

    endcase
end

endmodule
```

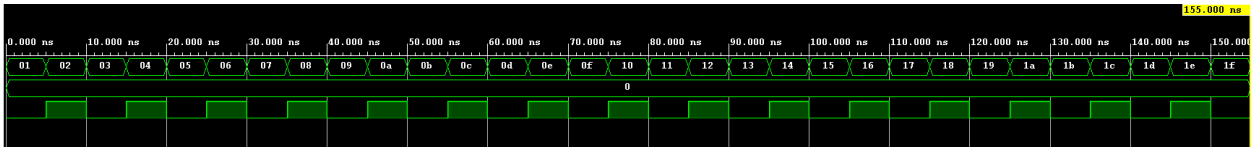
TESTBENCH – 1

```
module testbench;

reg [4:0] A;
reg [2:0] SEL;
wire OUT;

mult1 bloop(.A(A), .SEL(SEL), .OUT(OUT));

initial begin
   $monitor($time, " A = %d, SEL = %d, OUT = %b", A, SEL, OUT);

   // SEL = 2, RANGE = 31
    SEL = 4'b000; A = 5'b00001; // <- 0
   #5 A = 5'b00010; // <- 1
   #5 A = 5'b00011; // <- 0
   #5 A = 5'b00100; // <- 1
   #5 A = 5'b00101; // <- 0
   #5 A = 5'b00110; // <- 1
   #5 A = 5'b00111; // <- 0
   #5 A = 5'b01000; // <- 1
   #5 A = 5'b01001; // <- 0
   #5 A = 5'b01010; // <- 1
   #5 A = 5'b01011; // <- 0
   #5 A = 5'b01100; // <- 1
   #5 A = 5'b01101; // <- 0
   #5 A = 5'b01110; // <- 1
   #5 A = 5'b01111; // <- 0
   #5 A = 5'b10000; // <- 1
   #5 A = 5'b10001; // <- 0
   #5 A = 5'b10010; // <- 1
   #5 A = 5'b10011; // <- 0
   #5 A = 5'b10100; // <- 1
   #5 A = 5'b10101; // <- 0
   #5 A = 5'b10110; // <- 1
   #5 A = 5'b10111; // <- 0
   #5 A = 5'b11000; // <- 1
   #5 A = 5'b11001; // <- 0
```

```
    #5 A = 5'b11010; // <- 1
    #5 A = 5'b11011; // <- 0
    #5 A = 5'b11100; // <- 1
    #5 A = 5'b11101; // <- 0
    #5 A = 5'b11110; // <- 1
    #5 A = 5'b11111; // <- 0

    #5 $finish;
end

endmodule
```

RESULT - 1

The waveform shows all multiples of 2 within the range 31



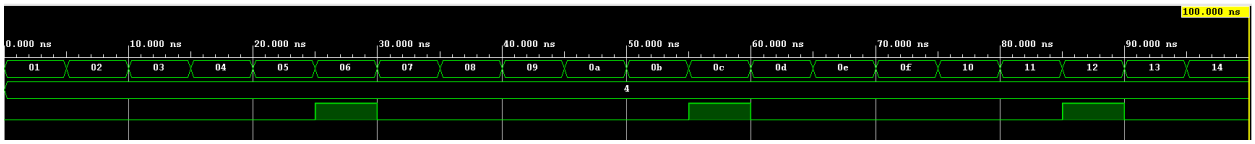
Notice the high output for every **2nd** interval for **15 HIGHS.**
(corresponding to **2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30**)

TESTBENCH – 2

```
module testbench;

reg [4:0] A;
reg [2:0] SEL;
wire OUT;

mult1 bloop(.A(A), .SEL(SEL), .OUT(OUT));

initial begin
   $monitor($time, " A = %d, SEL = %d, OUT = %b", A, SEL, OUT);

   // SEL = 6, RANGE = 20
    SEL = 4'b100; A = 5'b00001; // <- 0
   #5 A = 5'b00010; // <- 1
   #5 A = 5'b00011; // <- 0
   #5 A = 5'b00100; // <- 1
   #5 A = 5'b00101; // <- 0
   #5 A = 5'b00110; // <- 1
   #5 A = 5'b00111; // <- 0
   #5 A = 5'b01000; // <- 0
   #5 A = 5'b01001; // <- 0
   #5 A = 5'b01010; // <- 0
   #5 A = 5'b01011; // <- 0
   #5 A = 5'b01100; // <- 1
   #5 A = 5'b01101; // <- 0
   #5 A = 5'b01110; // <- 0
   #5 A = 5'b01111; // <- 0
   #5 A = 5'b10000; // <- 0
   #5 A = 5'b10001; // <- 0
   #5 A = 5'b10010; // <- 1
   #5 A = 5'b10011; // <- 0
   #5 A = 5'b10100; // <- 0

   #5 $finish;
end

endmodule
```

RESULT – 2

The waveform shows all multiples of 6 within the range 20



Notice the high output for every **6th** interval for **3 HIGHS.**
(corresponding to **6, 12, 18**)

PART - 2

The 5x32 Decoder

The 5x32 decoder takes in 5 inputs and produces 32 outputs. For each 5-bit input, the decoder gives the **minterm** corresponding to that number.

For example, for A = 25 or `5'b11001,` the decoder will light up for the 25th output as this output corresponds to the **minterm** $AB\overline{C}\overline{D}E$.

Therefore, rather than defining the boolean logic for each 5-bit number as done in **PART - 1**, we just select the output index of the 5x32 decoder that corresponds to the 5-bit number.

We just need to add the corresponding multiples of **SEL** to an **OR** gate.

For example, for multiples of 7, we need to do
OUT = decoder[7] OR decoder[14] OR decoder[21] OR decoder[28].

We have created a separate module for the decoder within the same Verilog file and have instantiated its instance in the multiple module.

VERILOG CODE

```
module decoder(input [4:0] a, output reg [31:0] out);

always @(*) begin
   case(a)
       5'd0  :  out = 32'b00000000000000000000000000000001;
       5'd1  :  out = 32'b00000000000000000000000000000010;
       5'd2  :  out = 32'b00000000000000000000000000000100;
       5'd3  :  out = 32'b00000000000000000000000000001000;
       5'd4  :  out = 32'b00000000000000000000000000010000;
       5'd5  :  out = 32'b00000000000000000000000000100000;
       5'd6  :  out = 32'b00000000000000000000000001000000;
       5'd7  :  out = 32'b00000000000000000000000010000000;
       5'd8  :  out = 32'b00000000000000000000000100000000;
       5'd9  :  out = 32'b00000000000000000000001000000000;
       5'd10 :  out = 32'b00000000000000000000010000000000;
       5'd11 :  out = 32'b00000000000000000000100000000000;
       5'd12 :  out = 32'b00000000000000000001000000000000;
       5'd13 :  out = 32'b00000000000000000010000000000000;
       5'd14 :  out = 32'b00000000000000000100000000000000;
       5'd15 :  out = 32'b00000000000000001000000000000000;
       5'd16 :  out = 32'b00000000000000010000000000000000;
       5'd17 :  out = 32'b00000000000000100000000000000000;
       5'd18 :  out = 32'b00000000000001000000000000000000;
       5'd19 :  out = 32'b00000000000010000000000000000000;
       5'd20 :  out = 32'b00000000000100000000000000000000;
       5'd21 :  out = 32'b00000000001000000000000000000000;
       5'd22 :  out = 32'b00000000010000000000000000000000;
       5'd23 :  out = 32'b00000000100000000000000000000000;
       5'd24 :  out = 32'b00000001000000000000000000000000;
       5'd25 :  out = 32'b00000010000000000000000000000000;
       5'd26 :  out = 32'b00000100000000000000000000000000;
       5'd27 :  out = 32'b00001000000000000000000000000000;
       5'd28 :  out = 32'b00010000000000000000000000000000;
       5'd29 :  out = 32'b00100000000000000000000000000000;
       5'd30 :  out = 32'b01000000000000000000000000000000;
       5'd31 :  out = 32'b10000000000000000000000000000000;
     default :  out = 32'b00000000000000000000000000000000;

   endcase
end
```

```
endmodule


module multiple(A, SEL, OUT);

input [4:0] A;
input [2:0] SEL;
output reg OUT;

wire [31:0] decode;

decoder dec(.a(A), .out(decode));

always @(*) begin
   case(SEL)

       0 : OUT = decode[2] |
                 decode[4] |
                 decode[6] |
                 decode[8] |
                decode[10] |
                decode[12] |
                decode[14] |
                decode[16] |
                decode[18] |
                decode[20] |
                decode[22] |
                decode[24] |
                decode[26] |
                decode[28] |
                decode[30];

       1 : OUT = decode[3] |
                 decode[6] |
                 decode[9] |
                decode[12] |
                decode[15] |
                decode[18] |
                decode[21] |
                decode[24] |
                decode[27] |
                decode[30];
```

```
        2 : OUT = decode[4] |
                  decode[8] |
                 decode[12] |
                 decode[16] |
                 decode[20] |
                 decode[24] |
                 decode[28];

        3 : OUT = decode[5] |
                  decode[10] |
                  decode[15] |
                  decode[20] |
                  decode[25] |
                  decode[30];

        4 : OUT = decode[6] |
                  decode[12] |
                  decode[18] |
                  decode[24] |
                  decode[30];

        5 : OUT = decode[7] | decode[14] | decode[21] | decode[28];

        6 : OUT = decode[8] | decode[16] | decode[26];

        7 : OUT = decode[9] | decode[18] | decode[27];

    endcase
end

endmodule
```

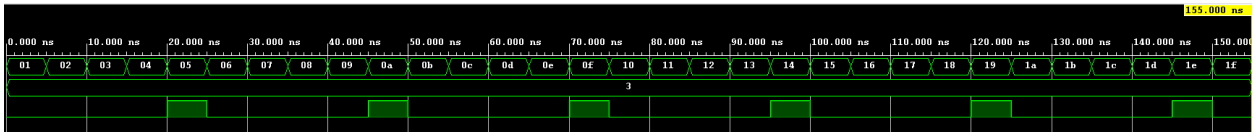
TESTBENCH - 1

```
module testbench;

reg [4:0] A;
reg [2:0] SEL;
wire OUT;

mult2 bloop(.A(A), .SEL(SEL), .OUT(OUT));

initial begin
   $monitor($time, " A = %d, SEL = %d, OUT = %b", A, SEL, OUT);

    // SEL = 5, RANGE = 31
    SEL = 4'b011; A = 5'b00001; // <- 0
   #5 A = 5'b00010; // <- 0
   #5 A = 5'b00011; // <- 0
   #5 A = 5'b00100; // <- 0
   #5 A = 5'b00101; // <- 1
   #5 A = 5'b00110; // <- 0
   #5 A = 5'b00111; // <- 0
   #5 A = 5'b01000; // <- 0
   #5 A = 5'b01001; // <- 0
   #5 A = 5'b01010; // <- 1
   #5 A = 5'b01011; // <- 0
   #5 A = 5'b01100; // <- 0
   #5 A = 5'b01101; // <- 0
   #5 A = 5'b01110; // <- 0
   #5 A = 5'b01111; // <- 1
   #5 A = 5'b10000; // <- 0
   #5 A = 5'b10001; // <- 0
   #5 A = 5'b10010; // <- 0
   #5 A = 5'b10011; // <- 0
   #5 A = 5'b10100; // <- 1
   #5 A = 5'b10101; // <- 0
   #5 A = 5'b10110; // <- 0
   #5 A = 5'b10111; // <- 0
   #5 A = 5'b11000; // <- 0
```

```
    #5 A = 5'b11001; // <- 1
    #5 A = 5'b11010; // <- 0
    #5 A = 5'b11011; // <- 0
    #5 A = 5'b11100; // <- 0
    #5 A = 5'b11101; // <- 0
    #5 A = 5'b11110; // <- 1
    #5 A = 5'b11111; // <- 0

    #5 $finish;
end

endmodule
```

RESULT - 1

The waveform shows all multiples of 5 within the range 31



Notice the high output for every **5th** interval for **6 HIGHS.**
(corresponding to **5, 10, 15, 20, 25, 30**)

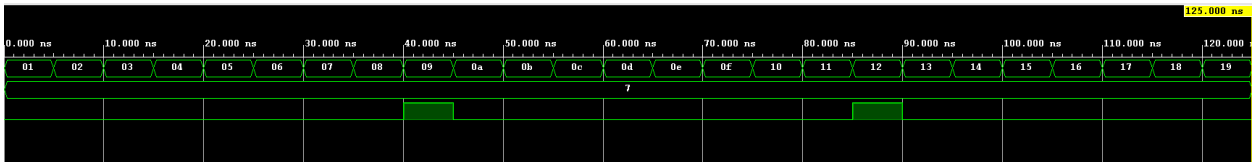
TESTBENCH – 2

```
module testbench;

reg [4:0] A;
reg [2:0] SEL;
wire OUT;

mult2 bloop(.A(A), .SEL(SEL), .OUT(OUT));

initial begin
   $monitor($time, " A = %d, SEL = %d, OUT = %b", A, SEL, OUT);
    // SEL = 9, RANGE = 25
    SEL = 4'b111; A = 5'b00001; // <- 0
   #5 A = 5'b00010; // <- 0
   #5 A = 5'b00011; // <- 0
   #5 A = 5'b00100; // <- 0
   #5 A = 5'b00101; // <- 0
   #5 A = 5'b00110; // <- 0
   #5 A = 5'b00111; // <- 0
   #5 A = 5'b01000; // <- 0
   #5 A = 5'b01001; // <- 1
   #5 A = 5'b01010; // <- 0
   #5 A = 5'b01011; // <- 0
   #5 A = 5'b01100; // <- 0
   #5 A = 5'b01101; // <- 0
   #5 A = 5'b01110; // <- 0
   #5 A = 5'b01111; // <- 0
   #5 A = 5'b10000; // <- 0
   #5 A = 5'b10001; // <- 0
   #5 A = 5'b10010; // <- 1
   #5 A = 5'b10011; // <- 0
   #5 A = 5'b10100; // <- 0
   #5 A = 5'b10101; // <- 0
   #5 A = 5'b10110; // <- 0
   #5 A = 5'b10111; // <- 0
   #5 A = 5'b11000; // <- 0
   #5 A = 5'b11001; // <- 0
   #5 $finish;
```

```
end

endmodule
```

RESULT – 2

The waveform shows all multiples of 9 within the range 25.



Notice the high output for every **9th** interval for **2 HIGHS.**
(corresponding to **9, 18**)