Hacettepe University

Computer Engineering Department

BM233 Logic Design Lab - 2021 Fall

# BBM 233 VERILOG ASSIGNMENT 1

December 5, 2021

*Student name:*
İrem Cengiz

*Student Number:*
b2200356006

## 1 Problem Definition

In first part, I will design a 2x4 decoder and implement it in Verilog HDL.

*2x4 decoder: Two inputs are decoded into four outputs. One of these four outputs will be HIGH for each combination of inputs*

In second part, I will design a 4-to-1 MUX and implement it in Verilog HDL.

*In a 4-to-1 MUX, only one out of four inputs is selected as the output based on a 2-bit select signal*

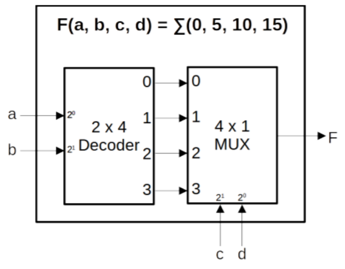


Figure 1: expected circuit

In last part, I will implement above figure using my decoder and mux.( I will use structural design)

## 2 Solution Implementation

### 2.1 2x4 decoder code

I obtained Boolean equations for the outputs D[0],D[1],D[2],D[3] from truth table.
I will use assign statements to implement each output signal to use dataflow design approach

```
module decoder_2x4(
    input[1:0] A,  //declaring 2 bit input
    output[3:0] D  //declaring 4 bit output
);

    assign D[0] = ~A[1] & ~A[0];
    assign D[1] = ~A[1] & A[0];
    assign D[2] = A[1] & ~A[0];
    assign D[3] = A[1] & A[0];

endmodule
```

## 2.2 4x1 multiplexer code

```
module mux_4x1(
    input[3:0] i,    //declaring 4 bit input
    input[1:0] s,    //declaring 2 bit selector
    output F         //declaring 1 bit output
);

    assign F = (~s[1] & ~s[0] & i[0]) | (~s[1] & s[0] & i[1]) |
    (s[1] & ~s[0] & i[2]) | (s[1] & s[0] & i[3]);

    //s[1] ? (s[0] ? i[0] : i[1]) : (s[0] ? i[2] : i[3]);
    //we can use this formula too


endmodule
```

## 2.3 circuit code

```
module circuit(
    input a,
    input b,
    input c,
    input d,
    output F
);

    wire [1:0] selector;          //selector of mux
    wire [1:0] decoder_input;
    wire [3:0] decoder_output;    //this will be input of mux

    assign selector={c,d};        //we need 2 bit selector
    assign decoder_input={a,b};   // 2 bit input

    //we need a decoder to implement expected circuit
    //structural description and explicit association

    decoder_2x4 decoder(.A(decoder_input[1:0]), .D(decoder_output[3:0]));
    mux_4x1 mux(.i(decoder_output[3:0]), .s(selector[1:0]), .F(F));


endmodule
```

## 3 Testbench Implementation

### 3.1 2x4 decoder test bench

```
module decoder_2x4_tb;
    reg[1:0]  A;     //declaring input as 2 bit reg
    wire[3:0] D;     //declaring output as 4 bit net

    //instantiate un t under test and explicit assotation
    decoder_2x4 uut (.A(A), .D(D));

    initial begin
     $dumpfile("decoder_2x4_tb.vcd");
     $dumpvars;

      //initialize possible inputs
      A[1]=0;  A[0]=0;
      #20;

      A[1]=0;  A[0]=1;
      #20;

      A[1]=1;  A[0]=0;
      #20;

      A[1]=1;  A[0]=1;
      #20;

      $finish;

    end
endmodule
```

### 3.2 4x1 mux test bench

```
module mux_4x1_tb();
 reg[3:0] i;   //declaring input 3 bit as reg
 reg[1:0] s;  //declaring selector 2 bit as reg

 wire F;      //declaring 1 bit output as net

 //instantiate un t under test and explicit assotation
 mux_4x1 uut(.i(i), .s(s), .F(F));

 //////    this part shows all 64 inputs and outputs
 // initial begin
 //    $dumpfile("mux_4x1.vcd");
 //    $dumpvars;
 //     i[3] = 1'b0;  i[2] = 1'b0;  i[1] = 1'b0;  i[0] = 1'b0;
 //     s[1]=1'b0;  s[0]=1'b0;
 //     #500
 //    $finish;
 // end
 //   always #40 i[3]=~i[3];
 //   always #20 i[2]=~i[2];
 //   always #10 i[1]=~i[1];
 //   always #5 i[0]=~i[0];
 //   always #80 s[0]=~s[0];
 //   always #160 s[1]=~s[1];
//////

  initial begin
    $dumpfile("mux_4x1.vcd");
    $dumpvars;

    i=4'b0001;  s=2'b00; #5;
    i=4'b1110;  s=2'b00; #5;
    i=4'b0010;  s=2'b01; #5;
    i=4'b1101;  s=2'b01; #5;
    i=4'b0100;  s=2'b10; #5;
    i=4'b1011;  s=2'b10; #5;
    i=4'b1000;  s=2'b11; #5;
    i=4'b0111;  s=2'b11; #5;
    $finish;
    end

endmodule
```

## 3.3 circuit test bench

```
module circuit_tb ();
  reg a,b,c,d;   //declaring inputs as regs
  wire F;       //declaring output as net

  //instantiate un t under test and explicit assotation
  circuit uut (.a(a) , .b(b) , .c(c) , .d(d) , .F(F));

  initial begin

    $dumpfile("circuit_tb.vcd");
    $dumpvars;

  // initialize all possible inputs
   a=0; b=0; c=0; d=0; #5;    a=0; b=0; c=0; d=1; #5;
   a=0; b=0; c=1; d=0; #5;    a=0; b=0; c=1; d=1; #5;
   a=0; b=1; c=0; d=0; #5;    a=0; b=1; c=0; d=1; #5;
   a=0; b=1; c=1; d=0; #5;    a=0; b=1; c=1; d=1; #5;
   a=1; b=0; c=0; d=0; #5;    a=1; b=0; c=0; d=1; #5;
   a=1; b=0; c=1; d=0; #5;    a=1; b=0; c=1; d=1; #5;
   a=1; b=1; c=0; d=0; #5;    a=1; b=1; c=0; d=1; #5;
   a=1; b=1; c=1; d=0; #5;    a=1; b=1; c=1; d=1; #5;
   $finish;

  end
endmodule
```

## 4 Waveforms

### 4.1 waveform for 2x4 decoder

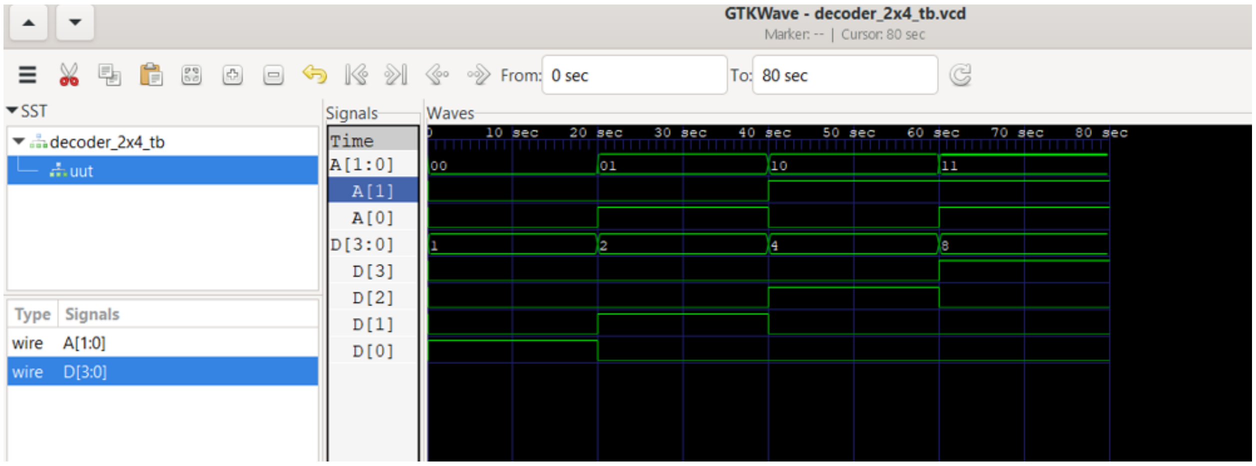
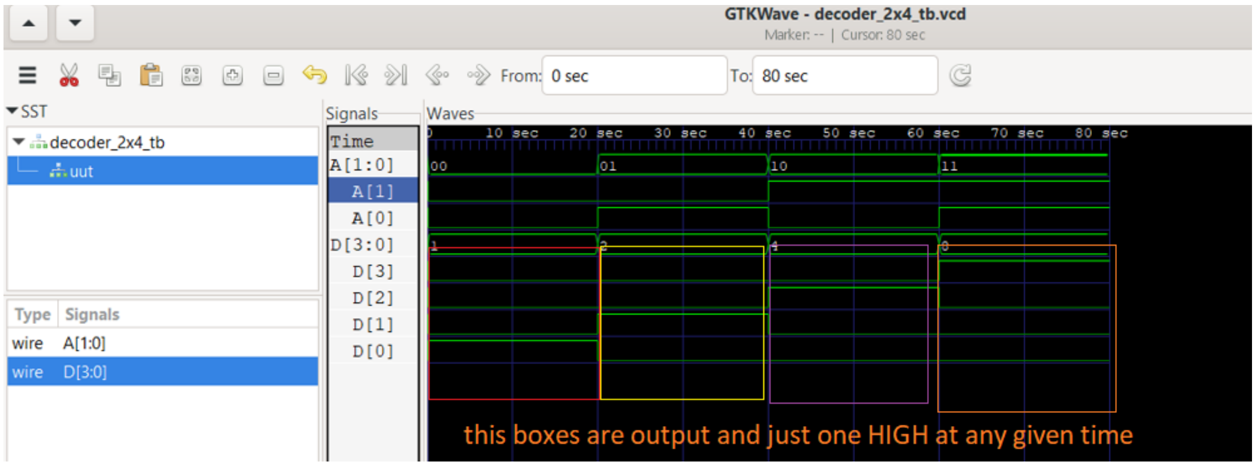


Figure 2: Resulting Waveform



this boxes are output and just one HIGH at any given time

Figure 3: Resulting Waveform with my edit

## 4.2 waveform for 4x1 mux (64 input)

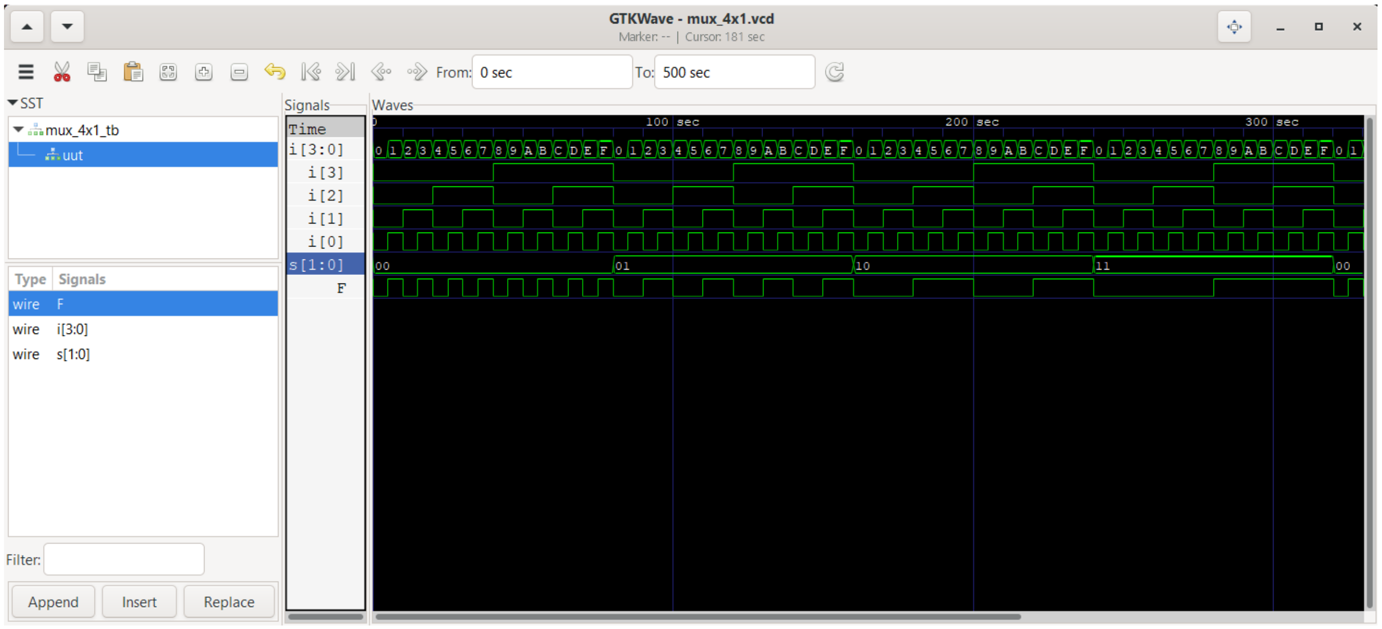


Figure 4: Resulting Waveform

## 4.3 waveform for 4x1 mux (expected 8 input)

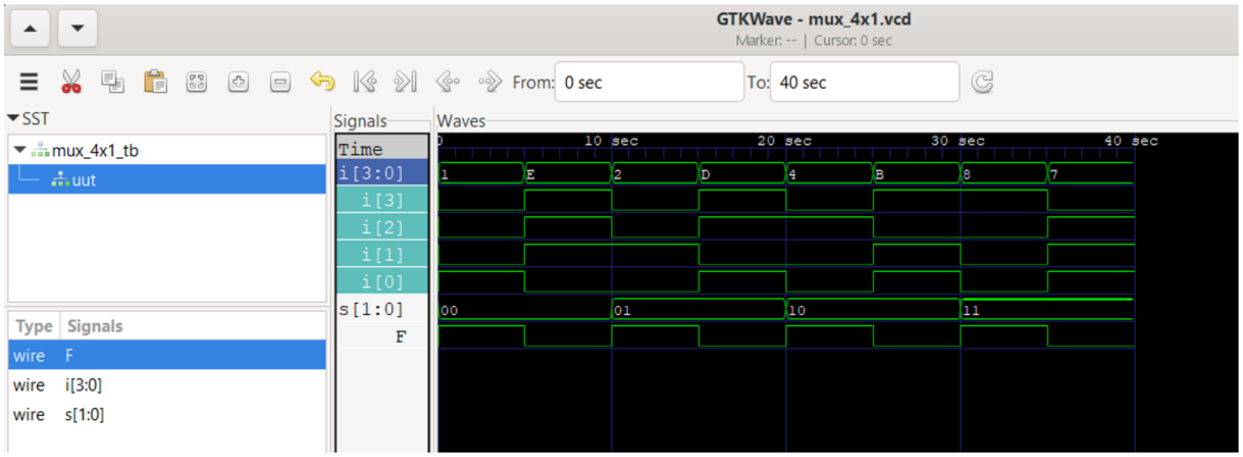


Figure 5: Resulting Waveform

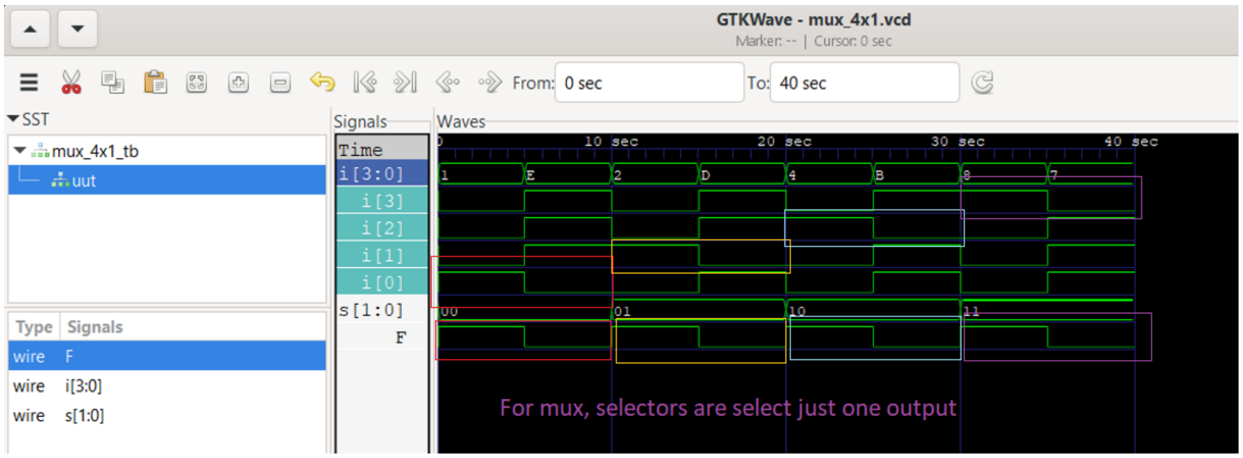


Figure 6: Resulting Waveform with my edit

## 4.4 waveform for circuit

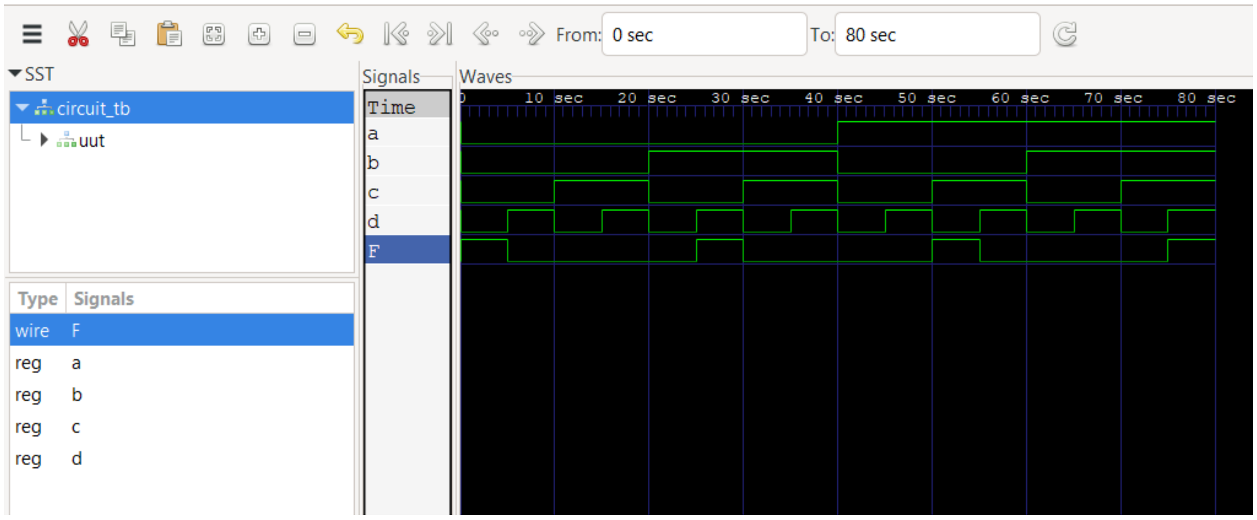


Figure 7: Resulting Waveform

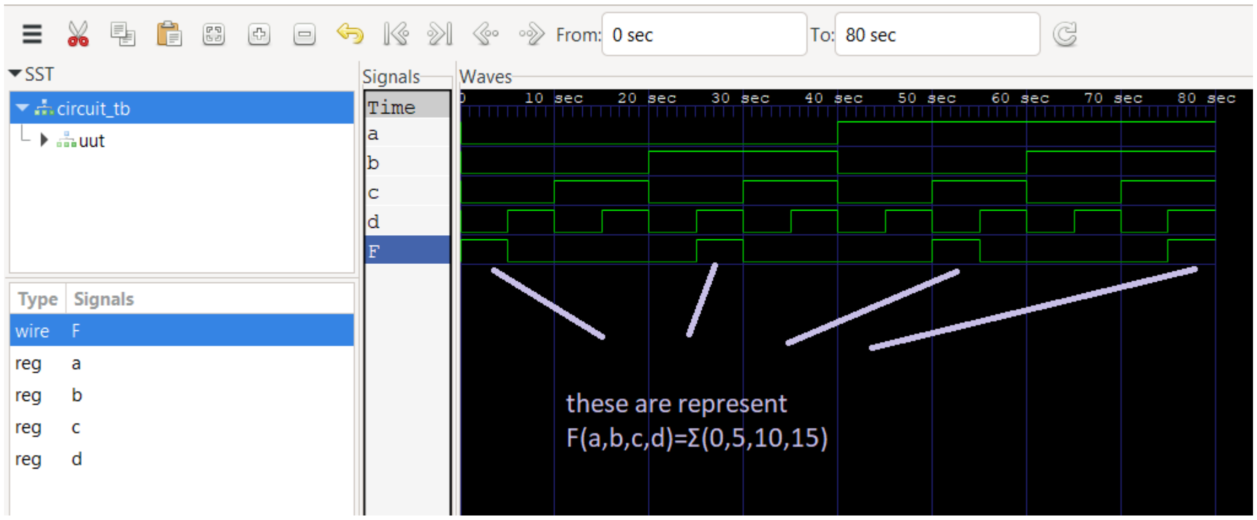


Figure 8: Resulting Waveform with my edit

## 5 Result

I obtained 3 different waveforms from test benchs and all waveforms came as expected waveforms. This shows that my decoder,mux and circuit are working correctly.

## 6 BONUS PART

### 6.1 8x1 mux code

```
module bonus(
    input a,
    input b,
    input c,
    input d,
    output F
);
    //to implement given function I will create 8 input
    //a is selector2
    //b is selector1
    //c is selector0
    //d will be boolean equation for input cases
    wire [7:0]i; //declare i for input cases

    //obtained equation from truth table
    //these inputs are always zero
    assign i[1]=0;     assign i[3]=0;
    assign i[4]=0;     assign i[6]=0;

    //if a,b,c are 0 output will be ~d
    assign i[0]=~a&~b&~c&~d;

    //if a,c are 0 and b is 1 output will be d
    assign i[2]=~a & b & ~c & d;
    //if a,c are 1 b is 0 output will be ~d
    assign i[5]= a & ~b & c & ~d;
    //if a,b,c are 1 output will be d
    assign i[7]= a & b & c & d;

    //the following calculation can be used to find the correct input
    assign F =i[4*a + 2*b +c];

endmodule
```

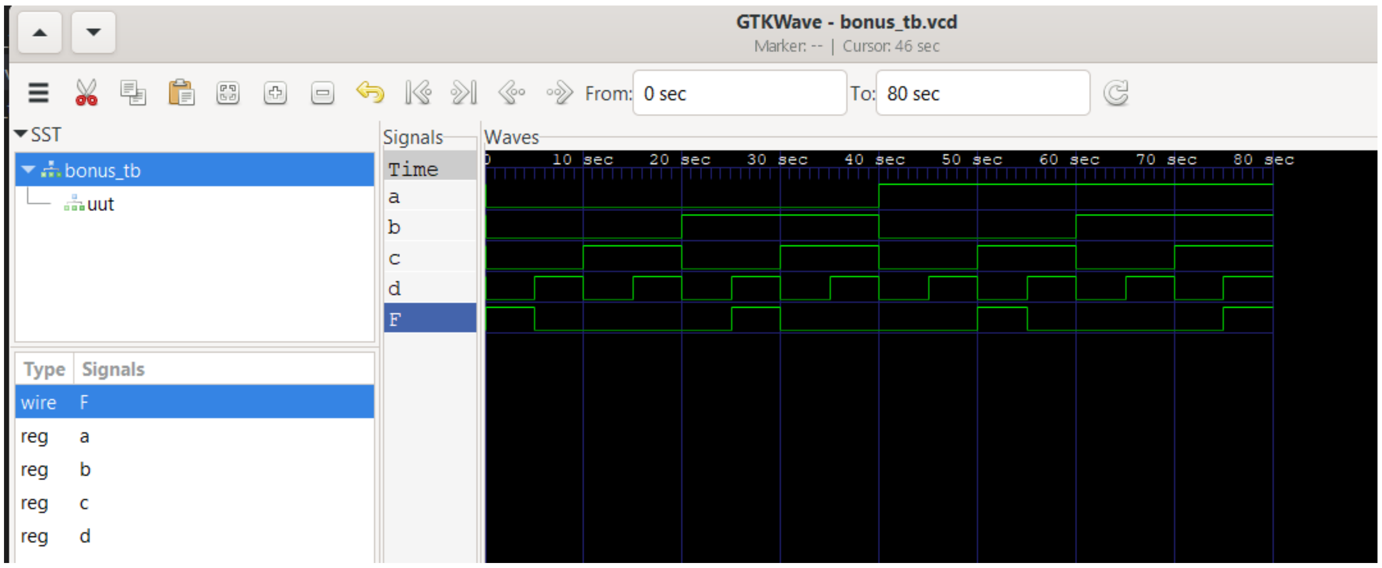
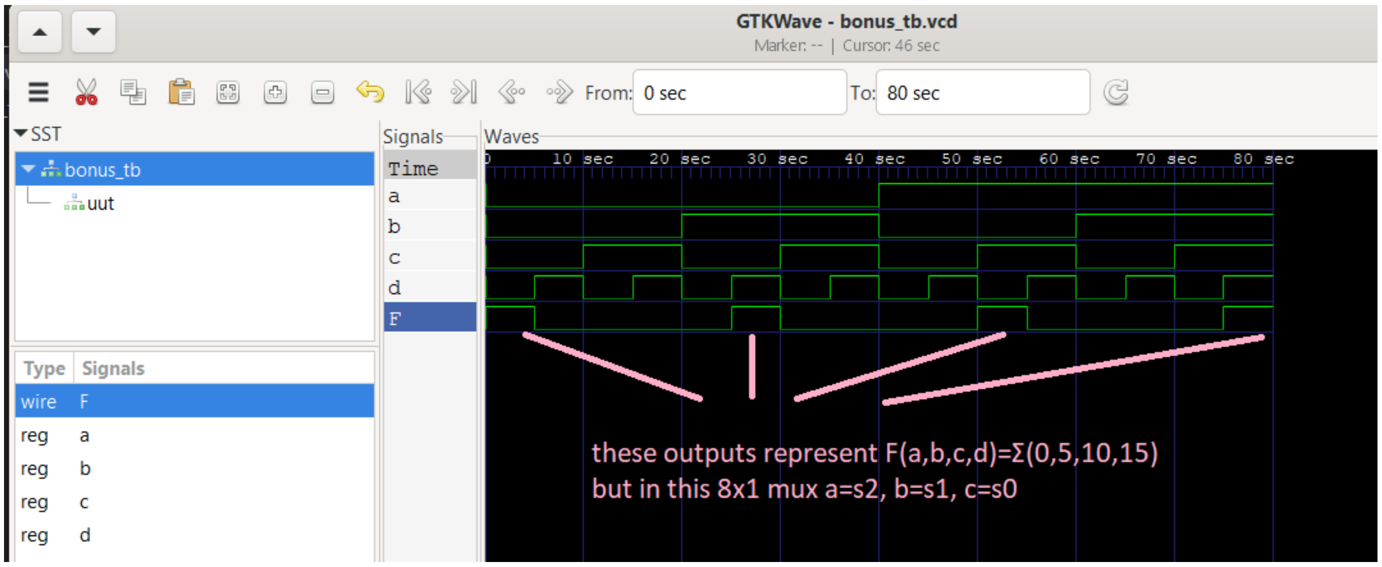


Figure 9: Resulting Waveform



these outputs represent F(a,b,c,d)=Σ(0,5,10,15)
but in this 8x1 mux a=s2, b=s1, c=s0

Figure 10: Resulting Waveform with my edit

## 6.2 8x1 mux waveform

When I test it using my testbench circuit-tb.v I got the expected result. The result for F(A, B, C, D) = (0, 5, 10, 15) function when selectors of 8x1 mux are a, b, c as the select signals s2, s1, s0 respectively.

## References

- Reference 1 https://web.cs.hacettepe.edu.tr/ bbm231/files/VerilogIntro.pdf
- Reference 2 https://technobyte.org/verilog-multiplexer-4x1
- Reference 3 https://cdn-uploads.piazza.com/paste/itmvemweb267cd/ 9d94a54942548e7ff1e9762d5517a65d0
  More$_{V}erilog_{E}xamples_{o}f_{C}ombinational_{c}ircuits.pdf$
- Reference 4 https://www.youtube.com/watch?v=Zk2mCKowUt4
- Reference 5 https://technobyte.org/verilog-multiplexer-4x1/
- Reference 6 https://iverilog.fandom.com/wiki/GTKWave
- Reference 7 http://munjalm.blogspot.com/2019/09/system-verilog-tasks-to-generate-vcd.html
- Reference 8 Examples of last year...