

Embedded Systems

CS429 Lab5

Name: Dipean Dasgupta

ID:202151188

Task1A Solution:

Module code:

```
module adder #(parameter n=4) (  
    input [n-1:0] A, // n-bit input operand A  
    input [n-1:0] B, // n-bit input operand B  
    output [n:0] sum, // (n+1)-bit sum output  
    output carry_out // Carry out bit  
);  
  
    assign {carry_out, sum} = A + B;  
  
endmodule
```

Testbench Code:

```
`include "lab5_1A.v"  
`timescale 1ns / 1ps  
module n_addertb;  
    parameter n = 4;  
    reg [n-1:0] A, B; // Input operands A and B  
    wire [n:0] sum; // Sum output  
    wire carry_out; // Carry out  
  
    adder dut (  
        .A(A),  
        .B(B),  
        .sum(sum),  
        .carry_out(carry_out)  
    );  
  
    initial begin  
        $dumpfile("n_bit_adder.vcd");  
        $dumpvars(0, n_addertb);  
        // Test cases  
        A = 4'b0010; // 2  
        B = 4'b1101; // 13  
        #10;  
    end  
endmodule
```

```

    $display("A = %b", A);
    $display("B = %b", B);
    $display("Sum = %b", sum);
    $display("Carry out = %b", carry_out);

    // Finish simulation
    $finish;
end

endmodule

```

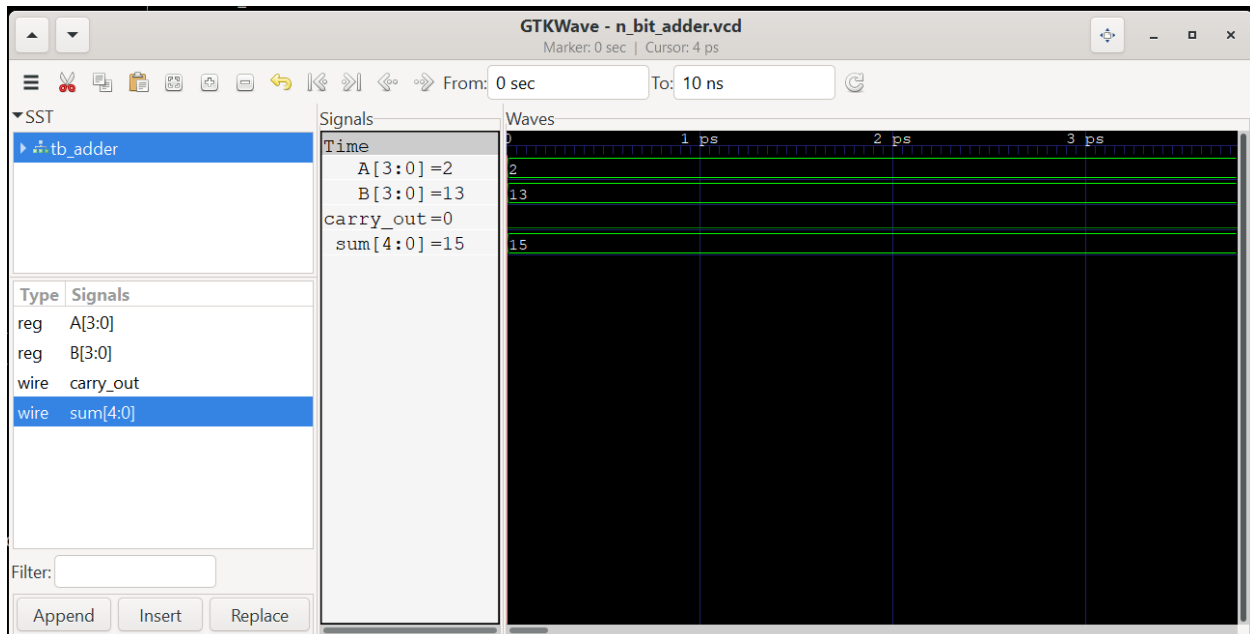
Result:

```

[Running] lab5_1Atb.v
VCD info: dumpfile n_bit_adder.vcd opened for output.
A = 0010
B = 1101
Sum = 01111
Carry out = 0
lab5_1Atb.v:40: $finish called at 10000 (1ps)
[Done] exit with code=0 in 0.236 seconds

```

Ln 1, Col 1 (952 selected)



Task1B Solution:

```
module adder #(parameter N=2) (  
    input [N-1:0] A,  
    input [N-1:0] B,  
    output [N:0] sum,  
    output carry_out  
);  
  
    assign sum = A + B;  
    assign carry_out = sum[N];  
endmodule  
  
module main_module;  
    parameter N3 = 3; // 3-bit adder  
    parameter N2 = 2; // 2-bit adder  
    parameter N4 = 4; // 4-bit adder  
    parameter N5 = 5; // 5-bit adder  
  
    // Instantiate adders with different sizes  
    adder #(N3) adder3bit (.A(A3), .B(B3), .sum(sum3bit),  
        .carry_out(carry_out3bit));  
    adder #(N2) adder2bit (.A(A2), .B(B2), .sum(sum2bit),  
        .carry_out(carry_out2bit));  
    adder #(N4) adder4bit (.A(A4), .B(B4), .sum(sum4bit),  
        .carry_out(carry_out4bit));  
    adder #(N5) adder5bit (.A(A5), .B(B5), .sum(sum5bit),  
        .carry_out(carry_out5bit));  
  
    // Inputs  
    reg [N3-1:0] A3, B3;  
    reg [N2-1:0] A2, B2;  
    reg [N4-1:0] A4, B4;  
    reg [N5-1:0] A5, B5;  
  
    // Outputs  
    wire [N3:0] sum3bit;  
    wire [N2:0] sum2bit;  
    wire [N4:0] sum4bit;  
    wire [N5:0] sum5bit;  
    wire carry_out3bit;  
    wire carry_out2bit;  
    wire carry_out4bit;  
    wire carry_out5bit;
```

```

initial begin
    $dumpfile("test_lab51b.vcd");
    $dumpvars(0, main_module);

    // Assign input values
    A3 = 3'b101; // 5 in binary
    B3 = 3'b011; // 3 in binary

    A2 = 2'b10;  // 2 in binary
    B2 = 2'b11;  // 3 in binary

    A4 = 4'b0101; // 13 in binary
    B4 = 4'b0010; // 10 in binary

    A5 = 5'b11001; // 25 in binary
    B5 = 5'b10110; // 22 in binary

    #10;

    // Display results
    $display("3-bit Adder: %b + %b = %b, Carry Out: %b", A3, B3, sum3bit,
carry_out3bit);
    $display("2-bit Adder: %b + %b = %b, Carry Out: %b", A2, B2, sum2bit,
carry_out2bit);
    $display("4-bit Adder: %b + %b = %b, Carry Out: %b", A4, B4, sum4bit,
carry_out4bit);
    $display("5-bit Adder: %b + %b = %b, Carry Out: %b", A5, B5, sum5bit,
carry_out5bit);

    $finish;
end
endmodule

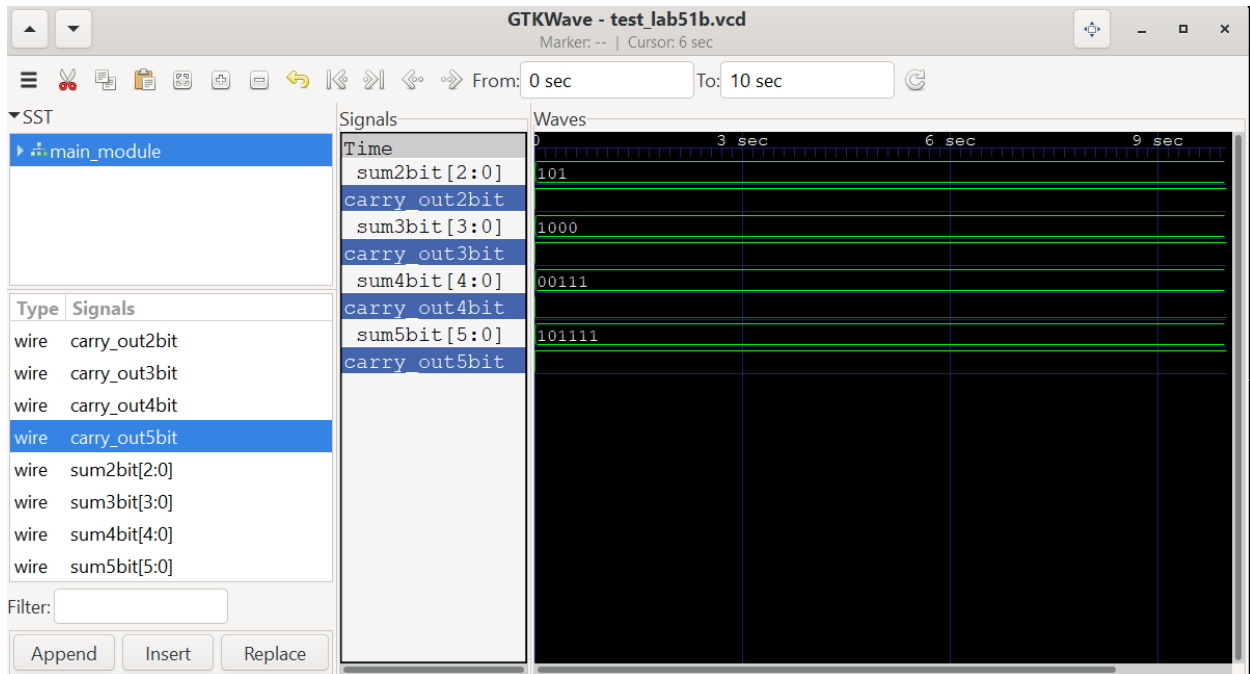
```

OUTPUT:

```

VCD info: dumpfile test_lab51b.vcd opened for output.
3-bit Adder: 101 + 011 = 1000, Carry Out: 1
2-bit Adder: 10 + 11 = 101, Carry Out: 1
4-bit Adder: 0101 + 0010 = 00111, Carry Out: 0
5-bit Adder: 11001 + 10110 = 101111, Carry Out: 1
lab5_1B.v:68: $finish called at 10 (1s)

```



Task2A Solution:

```
module mux_4to1 #(parameter N=4) (
    input [N-1:0] data_in0,
    input [N-1:0] data_in1,
    input [N-1:0] data_in2,
    input [N-1:0] data_in3,
    input [1:0] sel,
    output reg [N-1:0] data_out
);

    localparam WIDTH = 2 ** N;

    always @(sel or data_in0 or data_in1 or data_in2 or data_in3) begin
        case(sel)
            2'b00: data_out = data_in0;
            2'b01: data_out = data_in1;
            2'b10: data_out = data_in2;
            2'b11: data_out = data_in3;
            default: data_out = data_in0; // Default case
        endcase
    end
endmodule
```

```

module tb_mux_4to1;

    reg [3:0] data_in0, data_in1, data_in2, data_in3;
    reg [1:0] sel;
    wire [3:0] data_out;

    mux_4to1 #(4) uut (
        .data_in0(data_in0),
        .data_in1(data_in1),
        .data_in2(data_in2),
        .data_in3(data_in3),
        .sel(sel),
        .data_out(data_out)
    );

    initial begin
        // Test case 1: Select data_in0
        data_in0 = 4'b0000;
        data_in1 = 4'b1111;
        data_in2 = 4'b1010;
        data_in3 = 4'b0101;
        sel = 2'b00;
        #10 $display("Output: %b", data_out); // Expected output: 0000

        // Test case 2: Select data_in1
        data_in0 = 4'b0001;
        data_in1 = 4'b1100;
        data_in2 = 4'b1011;
        data_in3 = 4'b0101;
        sel = 2'b01;
        #10 $display("Output: %b", data_out); // Expected output: 1100

        // Test case 3: Select data_in2
        data_in0 = 4'b1111;
        data_in1 = 4'b0000;
        data_in2 = 4'b1011;
        data_in3 = 4'b1100;
        sel = 2'b10;
        #10 $display("Output: %b", data_out); // Expected output: 1011

        // Test case 4: Select data_in3
        data_in0 = 4'b1111;
        data_in1 = 4'b0000;
        data_in2 = 4'b1010;
        data_in3 = 4'b0101;
    end

```

```

        sel = 2'b11;
        #10 $display("Output: %b", data_out); // Expected output: 0101

        // Finish simulation
        $finish;
    end

endmodule

```

OUTPUT:

```

[Running] lab5_2A.v
Output: 0000
Output: 1100
Output: 1011
Output: 0101
lab5_2A.v:73: $finish called at 40 (1s)
[Done] exit with code=0 in 0.126 seconds

```

Task2B Solution:

Case_I: Parameter passing by Name

```

module mux2to1 #(parameter N = 8) (input [N-1:0] i0, i1, input s, output reg
[N-1:0] y);
always @(*) begin
if(s)
y = i1;
else
y = i0;
end
endmodule

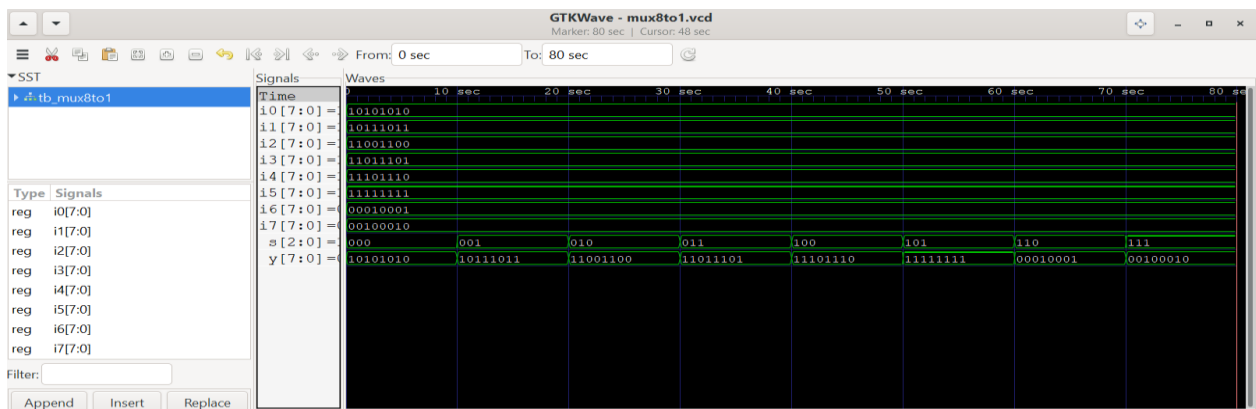
module mux8to1 #(parameter N = 8) (input [N-1:0] i0, i1, i2, i3, i4, i5, i6,
i7, input [2:0] s, output wire [N-1:0] y);
wire [N-1:0] o0, o1;
mux4to1 #(N) m0 (.i0(i0), .i1(i1), .i2(i2), .i3(i3), .s(s[1:0]), .y(o0));
mux4to1 #(N) m1 (.i0(i4), .i1(i5), .i2(i6), .i3(i7), .s(s[1:0]), .y(o1));
mux2to1 #(N) m2 (.i0(o0), .i1(o1), .s(s[2]), .y(y));
endmodule

```

```
// Testbench code
module tb_mux8to1;
reg [7:0] i0, i1, i2, i3, i4, i5, i6, i7;
reg [2:0] s;
wire [7:0] y;
mux8to1 #(8) u1 (.i0(i0), .i1(i1), .i2(i2), .i3(i3), .i4(i4), .i5(i5),
.i6(i6), .i7(i7), .s(s), .y(y));
initial begin
$dumpfile("mux8to1.vcd");
$dumppvars(0, tb_mux8to1);
i0 = 8'hAA; i1 = 8'hBB; i2 = 8'hCC; i3 = 8'hDD; i4 = 8'hEE; i5 =
8'hFF; i6 = 8'h11; i7 = 8'h22; s = 3'b000;
#10 s = 3'b001;
#10 s = 3'b010;
#10 s = 3'b011;
#10 s = 3'b100;
#10 s = 3'b101;
#10 s = 3'b110;
#10 s = 3'b111;
#10 $finish;
end
initial begin
$monitor("i0 = %b, i1 = %b, i2 = %b, i3 = %b, i4 = %b, i5 = %b, i6 = %b, i7 = %b,
s = %b, y = %b", i0, i1, i2, i3, i4, i5, i6, i7, s, y);
end
endmodule
```

OUTPUT:

```
[Running] lab5_2B2A.v
VCD info: dumpfile mux8to1.vcd opened for output.
i0 = 10101010, i1 = 10111011, i2 = 11001100, i3 = 11011101, i4 = 11101110, i5 = 11111111, i6 = 00010001, i7 = 00100010, s = 000, y = 10101010
i0 = 10101010, i1 = 10111011, i2 = 11001100, i3 = 11011101, i4 = 11101110, i5 = 11111111, i6 = 00010001, i7 = 00100010, s = 001, y = 10111011
i0 = 10101010, i1 = 10111011, i2 = 11001100, i3 = 11011101, i4 = 11101110, i5 = 11111111, i6 = 00010001, i7 = 00100010, s = 010, y = 11001100
i0 = 10101010, i1 = 10111011, i2 = 11001100, i3 = 11011101, i4 = 11101110, i5 = 11111111, i6 = 00010001, i7 = 00100010, s = 011, y = 11011101
i0 = 10101010, i1 = 10111011, i2 = 11001100, i3 = 11011101, i4 = 11101110, i5 = 11111111, i6 = 00010001, i7 = 00100010, s = 100, y = 11101110
i0 = 10101010, i1 = 10111011, i2 = 11001100, i3 = 11011101, i4 = 11101110, i5 = 11111111, i6 = 00010001, i7 = 00100010, s = 101, y = 11111111
i0 = 10101010, i1 = 10111011, i2 = 11001100, i3 = 11011101, i4 = 11101110, i5 = 11111111, i6 = 00010001, i7 = 00100010, s = 110, y = 00010001
i0 = 10101010, i1 = 10111011, i2 = 11001100, i3 = 11011101, i4 = 11101110, i5 = 11111111, i6 = 00010001, i7 = 00100010, s = 111, y = 00100010
lab5_2B2A.v:51: $finish called at 80 (1s)
```



Case_II: Parameter passing by Order

```
module mux4to1 #(parameter N = 8) (input [N-1:0] i0, i1, i2, i3, input [1:0]s,
output reg [N-1:0] y);
always @(*) begin
case(s)
2'b00: y = i0;
2'b01: y = i1;
2'b10: y = i2;
2'b11: y = i3;
endcase
end
endmodule

module mux2to1 #(parameter N = 8) (input [N-1:0] i0, i1, input s, output reg[N-
1:0] y);
always @(*) begin
if(s)
y = i1;
else
y = i0;
end
endmodule

module mux8to1 #(parameter N = 8) (input [N-1:0] i0, i1, i2, i3, i4, i5, i6,
i7, input [2:0] s, output wire [N-1:0] y);
wire [N-1:0] o0, o1;
mux4to1 #(N) m0 (.i0(i0), .i1(i1), .i2(i2), .i3(i3), .s(s[1:0]), .y(o0));
mux4to1 #(N) m1 (.i0(i4), .i1(i5), .i2(i6), .i3(i7), .s(s[1:0]), .y(o1));
mux2to1 #(N) m2 (.i0(o0), .i1(o1), .s(s[2]), .y(y));
endmodule

// Testbench code
module tb_mux8to1;
reg [7:0] i0, i1, i2, i3, i4, i5, i6, i7;
reg [2:0] s;
wire [7:0] y;
mux8to1 #(8) u1 (i0, i1, i2, i3, i4, i5, i6, i7, s, y);
initial begin
$dumpfile("mux8to1_order.vcd");
$dumpvars(0, tb_mux8to1);
i0 = 8'hAA; i1 = 8'hBB; i2 = 8'hCC; i3 = 8'hDD; i4 = 8'hEE; i5 =
8'hFF; i6 = 8'h11; i7 = 8'h22; s = 3'b000;
#10 s = 3'b001;
#10 s = 3'b010;
#10 s = 3'b011;
#10 s = 3'b100;
#10 s = 3'b101;
```

```

#10 s = 3'b110;
#10 s = 3'b111;
#10 $finish;
end

initial begin
$monitor("i0 = %b, i1 = %b, i2 = %b, i3 = %b, i4 = %b, i5 = %b, i6 = %b, i7 = %b,
s = %b, y = %b", i0, i1, i2, i3, i4, i5, i6, i7, s, y);
end
endmodule

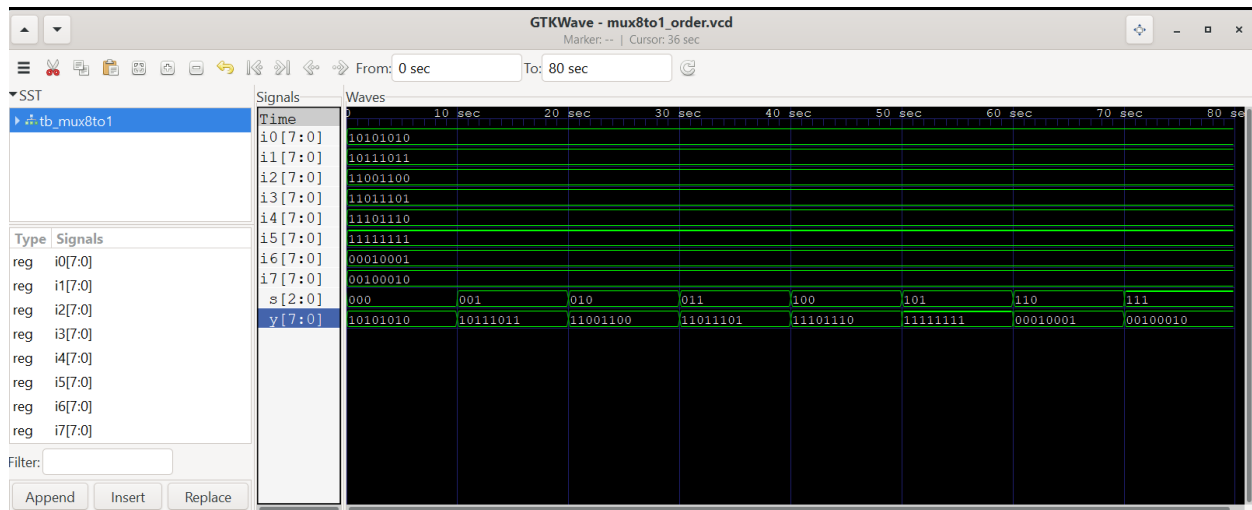
```

OUTPUT:

```

[Running] lab5_2B2B.v
VCD info: dumpfile mux8to1_order.vcd opened for output.
i0 = 10101010, i1 = 10111011, i2 = 11001100, i3 = 11011101, i4 = 11101110, i5 = 11111111, i6 = 00010001, i7 = 00100010, s = 000, y = 10101010
i0 = 10101010, i1 = 10111011, i2 = 11001100, i3 = 11011101, i4 = 11101110, i5 = 11111111, i6 = 00010001, i7 = 00100010, s = 001, y = 10111011
i0 = 10101010, i1 = 10111011, i2 = 11001100, i3 = 11011101, i4 = 11101110, i5 = 11111111, i6 = 00010001, i7 = 00100010, s = 010, y = 11001100
i0 = 10101010, i1 = 10111011, i2 = 11001100, i3 = 11011101, i4 = 11101110, i5 = 11111111, i6 = 00010001, i7 = 00100010, s = 011, y = 11011101
i0 = 10101010, i1 = 10111011, i2 = 11001100, i3 = 11011101, i4 = 11101110, i5 = 11111111, i6 = 00010001, i7 = 00100010, s = 100, y = 11101110
i0 = 10101010, i1 = 10111011, i2 = 11001100, i3 = 11011101, i4 = 11101110, i5 = 11111111, i6 = 00010001, i7 = 00100010, s = 101, y = 11111111
i0 = 10101010, i1 = 10111011, i2 = 11001100, i3 = 11011101, i4 = 11101110, i5 = 11111111, i6 = 00010001, i7 = 00100010, s = 110, y = 00010001
i0 = 10101010, i1 = 10111011, i2 = 11001100, i3 = 11011101, i4 = 11101110, i5 = 11111111, i6 = 00010001, i7 = 00100010, s = 111, y = 00100010
lab5_2B2B.v:44: $finish called at 80 (1s)
[Done] exit with code=0 in 0.122 seconds

```



Case_III: Parameter passing by Namelist:

```

module mux4to1 #(parameter N = 8) (input [N-1:0] i0, i1, i2, i3, input [1:0]
s, output reg [N-1:0] y);
always @(*) begin
case(s)
2'b00: y = i0;
2'b01: y = i1;
2'b10: y = i2;
2'b11: y = i3;
endcase
end

```

```

endmodule
module mux2to1 #(parameter N = 8) (input [N-1:0] i0, i1, input s, output reg
[N-1:0] y);
always @(*) begin
if(s)
y = i1;
else
y = i0;
end
endmodule
module mux8to1 #(parameter N = 8) (input [N-1:0] i0, i1, i2, i3, i4, i5, i6,
i7, input [2:0] s, output wire [N-1:0] y);
wire [N-1:0] o0, o1;
mux4to1 #(N) m0 (.i0(i0), .i1(i1), .i2(i2), .i3(i3), .s(s[1:0]), .y(o0));
mux4to1 #(N) m1 (.i0(i4), .i1(i5), .i2(i6), .i3(i7), .s(s[1:0]), .y(o1));
mux2to1 #(N) m2 (.i0(o0), .i1(o1), .s(s[2]), .y(y));
endmodule

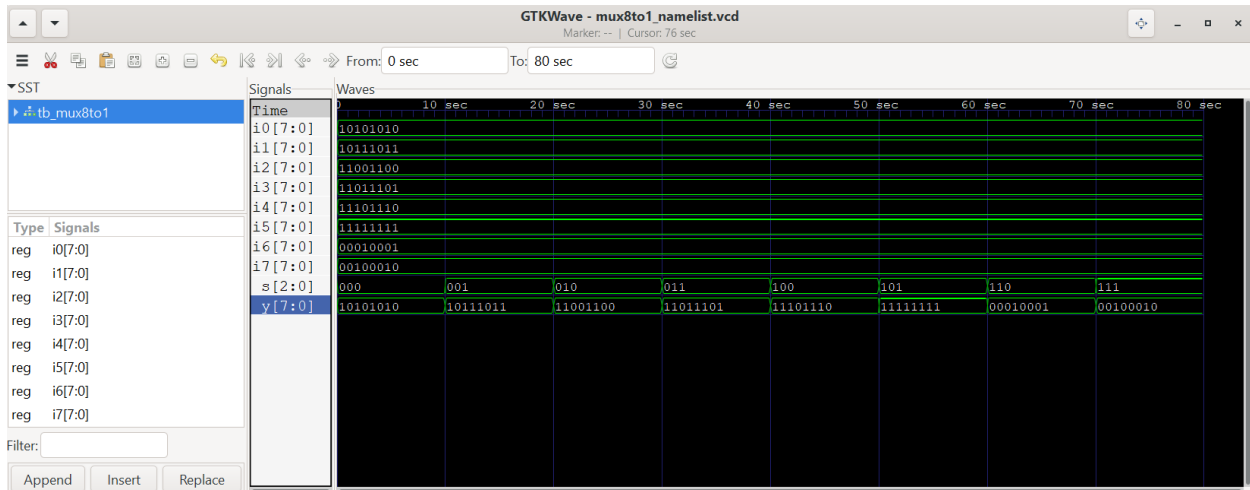
// Testbench code
module tb_mux8to1;
reg [7:0] i0, i1, i2, i3, i4, i5, i6, i7;
reg [2:0] s;
wire [7:0] y;
mux8to1 #(8) u1 (.i0(i0), .i1(i1), .i2(i2), .i3(i3), .i4(i4), .i5(i5),
.i6(i6), .i7(i7), .s(s), .y(y));
initial begin
$dumpfile("mux8to1_namelist.vcd");
$dumpvars(0, tb_mux8to1);
i0 = 8'hAA; i1 = 8'hBB; i2 = 8'hCC; i3 = 8'hDD; i4 = 8'hEE; i5 =
8'hFF; i6 = 8'h11; i7 = 8'h22; s = 3'b000;
#10 s = 3'b001;
#10 s = 3'b010;
#10 s = 3'b011;
#10 s = 3'b100;
#10 s = 3'b101;
#10 s = 3'b110;
#10 s = 3'b111;
#10 $finish;
end

initial begin
$monitor("i0 = %b, i1 = %b, i2 = %b, i3 = %b, i4 = %b, i5 = %b, i6 = %b, i7 = %b,
s = %b, y = %b", i0, i1, i2, i3, i4, i5, i6, i7, s, y);
end
endmodule

```

OUTPUT:

```
[Running] lab5_2B2C.v
VCD info: dumpfile mux8to1_namelist.vcd opened for output.
i0 = 10101010, i1 = 10111011, i2 = 11001100, i3 = 11011101, i4 = 11101110, i5 = 11111111, i6 = 00010001, i7 = 00100010, s = 000, y = 10101010
i0 = 10101010, i1 = 10111011, i2 = 11001100, i3 = 11011101, i4 = 11101110, i5 = 11111111, i6 = 00010001, i7 = 00100010, s = 001, y = 10111011
i0 = 10101010, i1 = 10111011, i2 = 11001100, i3 = 11011101, i4 = 11101110, i5 = 11111111, i6 = 00010001, i7 = 00100010, s = 010, y = 11001100
i0 = 10101010, i1 = 10111011, i2 = 11001100, i3 = 11011101, i4 = 11101110, i5 = 11111111, i6 = 00010001, i7 = 00100010, s = 011, y = 11011101
i0 = 10101010, i1 = 10111011, i2 = 11001100, i3 = 11011101, i4 = 11101110, i5 = 11111111, i6 = 00010001, i7 = 00100010, s = 100, y = 11101110
i0 = 10101010, i1 = 10111011, i2 = 11001100, i3 = 11011101, i4 = 11101110, i5 = 11111111, i6 = 00010001, i7 = 00100010, s = 101, y = 11111111
i0 = 10101010, i1 = 10111011, i2 = 11001100, i3 = 11011101, i4 = 11101110, i5 = 11111111, i6 = 00010001, i7 = 00100010, s = 110, y = 00010001
i0 = 10101010, i1 = 10111011, i2 = 11001100, i3 = 11011101, i4 = 11101110, i5 = 11111111, i6 = 00010001, i7 = 00100010, s = 111, y = 00100010
lab5_2B2C.v:48: $finish called at 80 (1s)
[Done] exit with code=0 in 0.137 seconds
```



Task3 Solution:

```
`timescale 1us/1us

module full_adder(input a, b, cin, output sum, cout);
    assign {cout, sum} = a + b + cin;
endmodule

module tb_full_adder;
    reg a, b, cin;
    wire sum, cout;

    full_adder u1 (.a(a), .b(b), .cin(cin), .sum(sum), .cout(cout));

    initial begin
        $dumpfile("full_adder_wf.vcd");
        $dumpvars(0, tb_full_adder);

        // All possible combinations of 1, 0, x
        #1 a = 1'b0; b = 1'b0; cin = 1'b0;
        #1 a = 1'b0; b = 1'b0; cin = 1'b1;
```

```

#1 a = 1'b0; b = 1'b0; cin = 1'bx;
#1 a = 1'b0; b = 1'b1; cin = 1'b0;
#1 a = 1'b0; b = 1'b1; cin = 1'b1;
#1 a = 1'b0; b = 1'b1; cin = 1'bx;
#1 a = 1'b0; b = 1'bx; cin = 1'b0;
#1 a = 1'b0; b = 1'bx; cin = 1'b1;
#1 a = 1'b0; b = 1'bx; cin = 1'bx;

#1 a = 1'b1; b = 1'b0; cin = 1'b0;
#1 a = 1'b1; b = 1'b0; cin = 1'b1;
#1 a = 1'b1; b = 1'b0; cin = 1'bx;
#1 a = 1'b1; b = 1'b1; cin = 1'b0;
#1 a = 1'b1; b = 1'b1; cin = 1'b1;
#1 a = 1'b1; b = 1'b1; cin = 1'bx;
#1 a = 1'b1; b = 1'bx; cin = 1'b0;
#1 a = 1'b1; b = 1'bx; cin = 1'b1;
#1 a = 1'b1; b = 1'bx; cin = 1'bx;

#10 $finish;
end

// Display values
initial begin
    $monitor("time = %0t us, a = %b, b = %b, cin = %b, sum = %b, cout = %b",
$time, a, b, cin, sum, cout);
end
endmodule

```

OUTPUT:

```

[Running] lab5_3.v
VCD info: dumpfile full_adder_wf.vcd opened for output.
time = 0 us, a = x, b = x, cin = x, sum = x, cout = x
time = 1 us, a = 0, b = 0, cin = 0, sum = 0, cout = 0
time = 2 us, a = 0, b = 0, cin = 1, sum = 1, cout = 0
time = 3 us, a = 0, b = 0, cin = x, sum = x, cout = x
time = 4 us, a = 0, b = 1, cin = 0, sum = 1, cout = 0
time = 5 us, a = 0, b = 1, cin = 1, sum = 0, cout = 1
time = 6 us, a = 0, b = 1, cin = x, sum = x, cout = x
time = 7 us, a = 0, b = x, cin = 0, sum = x, cout = x
time = 8 us, a = 0, b = x, cin = 1, sum = x, cout = x
time = 9 us, a = 0, b = x, cin = x, sum = x, cout = x
time = 10 us, a = 1, b = 0, cin = 0, sum = 1, cout = 0
time = 11 us, a = 1, b = 0, cin = 1, sum = 0, cout = 1
time = 12 us, a = 1, b = 0, cin = x, sum = x, cout = x
time = 13 us, a = 1, b = 1, cin = 0, sum = 0, cout = 1
time = 14 us, a = 1, b = 1, cin = 1, sum = 1, cout = 1
time = 15 us, a = 1, b = 1, cin = x, sum = x, cout = x
time = 16 us, a = 1, b = x, cin = 0, sum = x, cout = x
time = 17 us, a = 1, b = x, cin = 1, sum = x, cout = x
time = 18 us, a = 1, b = x, cin = x, sum = x, cout = x
lab5_3.v:38: $finish called at 28 (1us)
[Done] exit with code=0 in 0.3 seconds

```

Task4 Solution:

```
`timescale 1ps/1ps
module half_adder (
    input wire In1,
    input wire In2,
    output wire Sum,
    output wire Cout
);
    assign {Cout, Sum} = In1 + In2;
endmodule

module tb_half_adder;
    reg In1, In2;
    wire Sum, Cout;
    half_adder uut (
        .In1(In1),
        .In2(In2),
        .Sum(Sum),
        .Cout(Cout)
    );

    initial begin
        // Case 1:
        In1 = 0;
        In2 = 0;
        #0;
        $display("Time: %0t ps In1: %b In2: %b Sum: %b Cout: %b", $time, In1,
In2, Sum, Cout);

        // Case 2:
        In1 = 0;
        In2 = 1;
        #3; // Wait for 3 ps
        $display("Time: %0t ps In1: %b In2: %b Sum: %b Cout: %b", $time, In1,
In2, Sum, Cout);

        // Case 3:
        In1 = 1;
        In2 = 0;
        #67; // Wait for additional 67 ps
        $display("Time: %0t ps In1: %b In2: %b Sum: %b Cout: %b", $time, In1,
In2, Sum, Cout);

        // Case 4:
```

```

        In1 = 1;
        In2 = 1;
        #1; // Wait for additional 1 ps (total 71 ps)
        $display("Time: %0t ps In1: %b In2: %b Sum: %b Cout: %b", $time, In1,
In2, Sum, Cout);
        $finish;
    end
endmodule

```

OUTPUT:

```

[Running] lab5_4.v
Time: 0 ps In1: 0 In2: 0 Sum: 0 Cout: 0
Time: 3 ps In1: 0 In2: 1 Sum: 1 Cout: 0
Time: 70 ps In1: 1 In2: 0 Sum: 1 Cout: 0
Time: 71 ps In1: 1 In2: 1 Sum: 0 Cout: 1
lab5_4.v:48: $finish called at 71 (1ps)
[Done] exit with code=0 in 0.54 seconds

```

