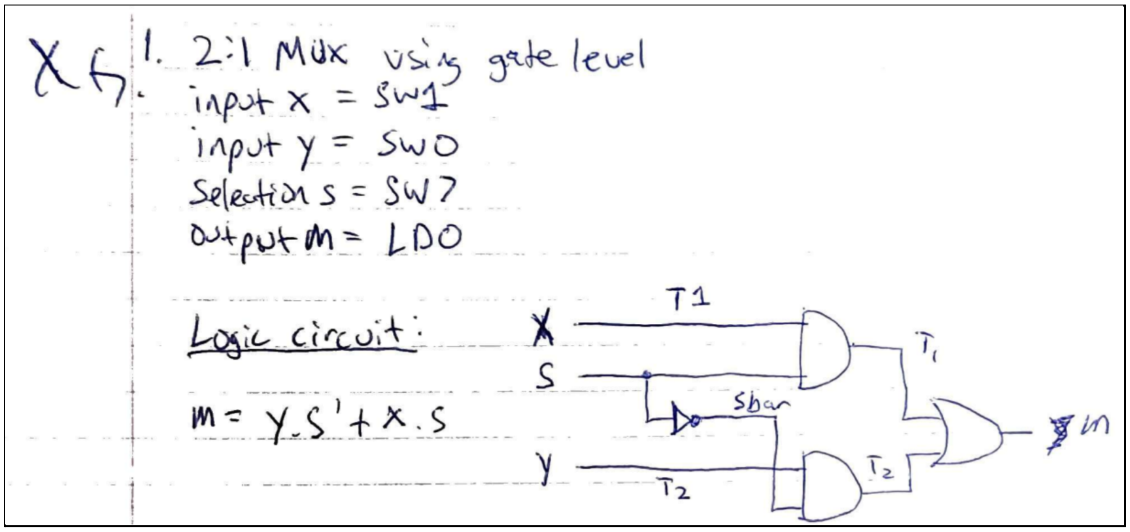
Mark Mitri and Ryan Forelli 8/302021
ECE 128 Lab 1

Part 1: 2-1 Multiplexer using Gate-Level Modeling

Design Diagram:

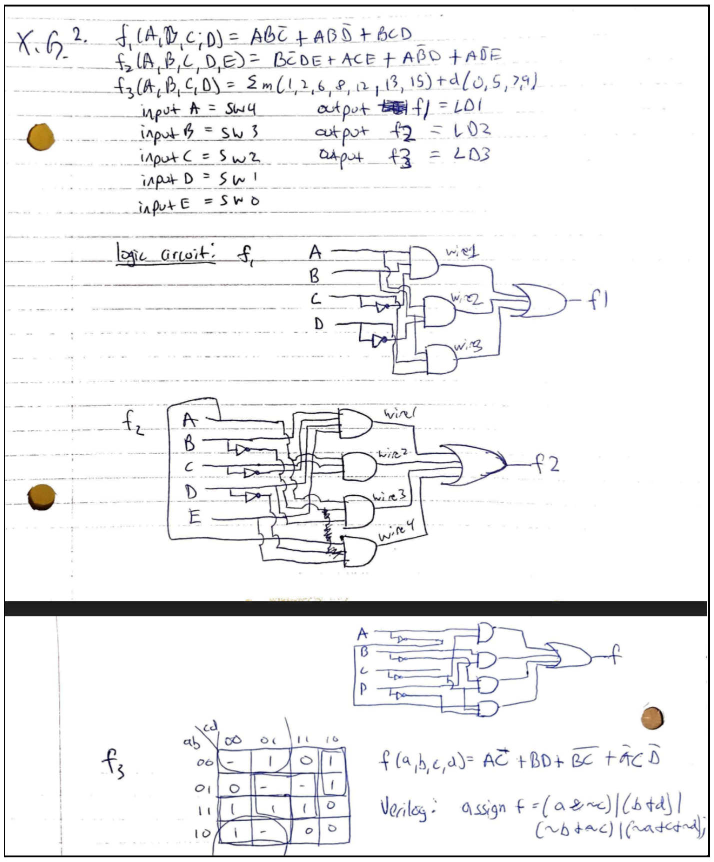


Code:

```
`timescale 1ns / 1ps
module multiplexer2to1(
    input x,
    input y,
    input s,
    output m
    );
    wire not_wire, and1_wire, and2_wire;
    not n1(not_wire, s);
    and a1(and1_wire, not_wire, x);
    and a2(and2_wire, s, y);
    or o1(m, and1_wire, and2_wire);
endmodule
```

Part 2: 3-Function

Design Diagram:



X.G. 2. $f_1(A,B,C,D) = AB\bar{C} + AB\bar{D} + BCD$

$f_2(A,B,C,D,E) = B\bar{C}DE + ACE + A\bar{B}D + A\bar{D}E$

$f_3(A,B,C,D) = \Sigma m(1,2,6,8,12,13,15) + d(0,5,7,9)$

input A = SW4 output ~~[illegible]~~ f1 = LD1

input B = SW3 output f2 = LD2

input C = SW2 output f3 = LD3

input D = SW1

input E = SW0

logic circuit: f_1

f_3

ab \ cd	00	01	11	10
00	-	1	0	1
01	0	-	-	1
11	1	1	1	0
10	1	-	0	0

$f(a,b,c,d) = A\bar{C} + BD + \overline{BC} + \bar{A}C\bar{D}$

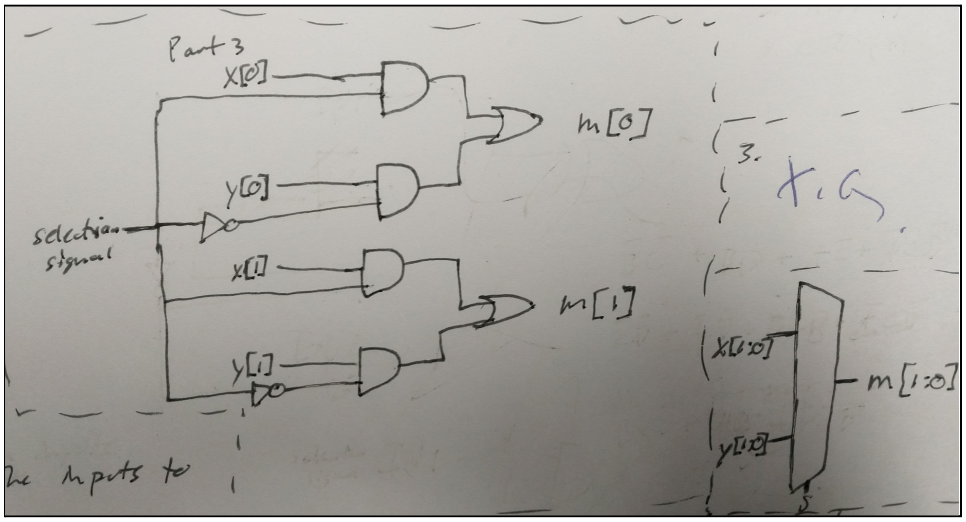
Verilog: assign f = (a & ~c) | (b + d) | (~b + ~c) | (~a + c + ~d);

Code:

```
`timescale 1ns / 1ps
module functions(
    input A,
    input B,
    input C,
    input D,
    input E,
    output F1,
    output F2,
    output F3
    );
    assign F1 = (A & B & ~C) | (A & B & ~D) | (B & C & D);
    assign F2 = (B & ~C & D & E) | (A & C & E) | (A & ~B & D) + (A & ~D & E);
    //Logic minimization with K-map
    assign F3 = (~B & ~C) | (A & ~C) | (B & D) | (~A & C & ~D);
endmodule
```

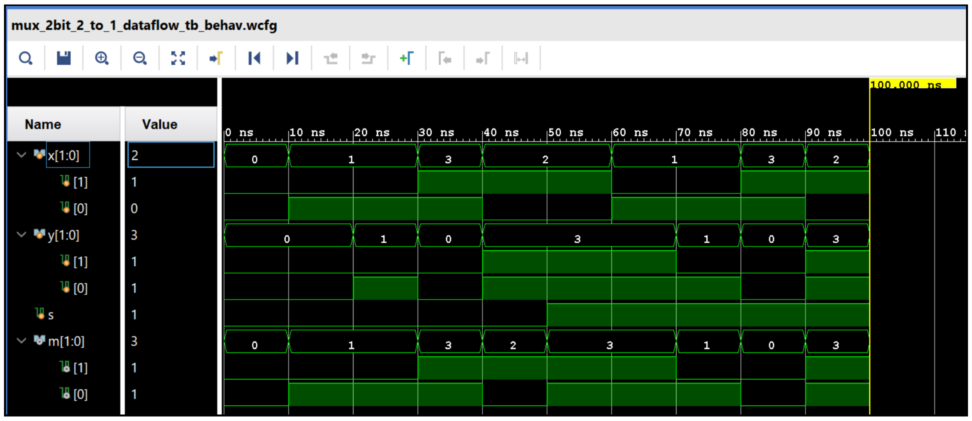
Part 3: 2-to-1 Multiplexer using Dataflow Modeling

Design Diagram:



Code:

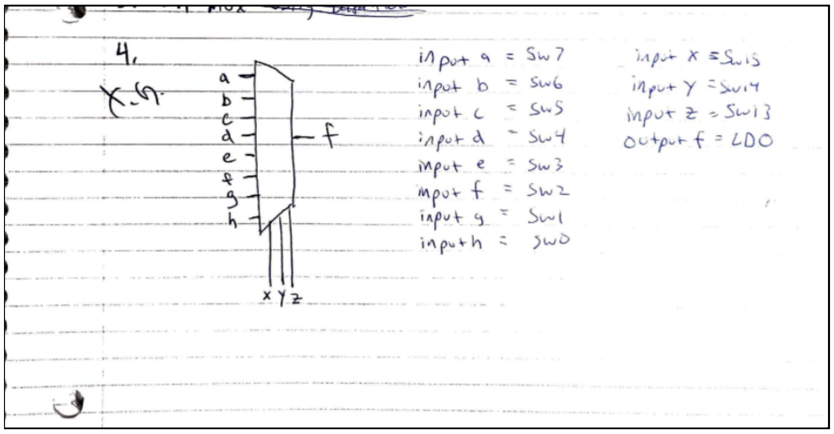
```
`timescale 1ns / 1ps
module mux_2bit_2_to_1_dataflow(
    input [1:0] x,
    input [1:0] y,
    input s,
    output [1:0] m
    );
    assign m[0] = (x[0] & ~s) | (y[0] & s);
    assign m[1] = (x[1] & ~s) | (y[1] & s);
endmodule
```



The waveform is as expected. When the selection signal is low, the inputs from x are outputted to m. As soon as the selection signal is switched high, the inputs from y are outputted to m.

Part 4: 8-1 Multiplexer using Dataflow Modeling

Design Diagram:



Code:

```
`timescale 1ns / 1ps
module multiplexer_8to1_behav(
    input a,
    input b,
    input c,
    input d,
    input e,
    input f,
    input g,
    input h,
    input [2:0] s,
    output reg m
    );
    // * is equivalent to all
    always @ (*)
        begin
        case (s)
            0 : m = a;
            1 : m = b;
            2 : m = c;
            3 : m = d;
            4 : m = e;
            5 : m = f;
            6 : m = g;
            7 : m = h;
        endcase
    end
endmodule
```