Muxes

Presented By: Trevor Abel, Ryan Hinson, Samuel Manos, and Mark Neitzel

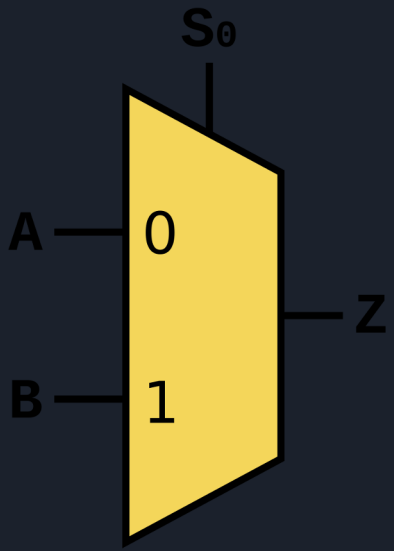
Why Are Muxes Important?

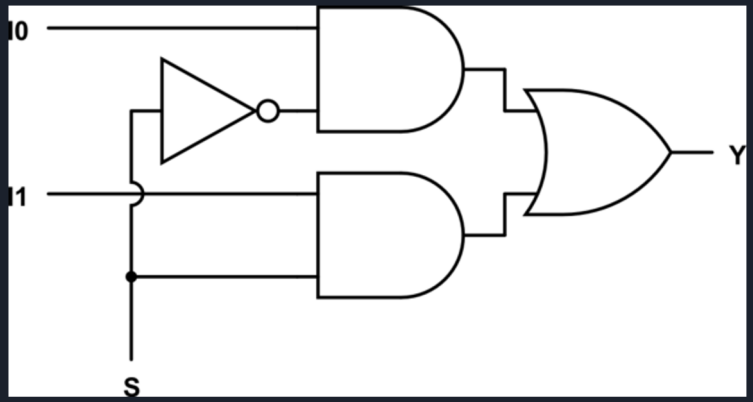
- A multiplexer is a combinational circuit that has many data inputs and a single output that depends on control or select inputs.
- Muxes are mainly used to increase the amount of data that can be sent over the network within a certain amount of time and bandwidth.
- Multiplexers are used to select either a register operand (0) or a constant operand (1).

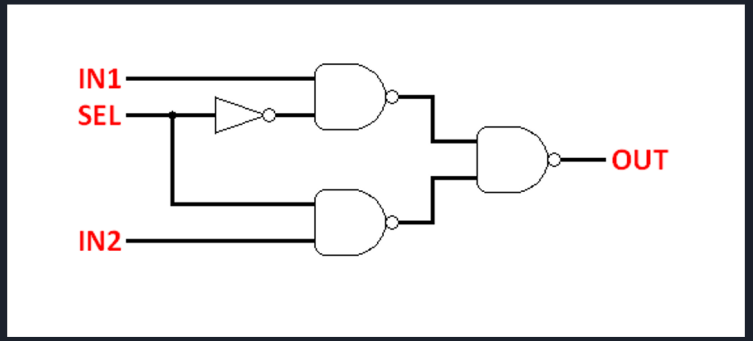
Instructions That Muxes Support

- A multiplexer is a logic gates circuit used to fetch a bit of data from memory at a given memory address. A processor has several multiplexers (MUX) controlling the data and address buses.
- Multiplexers are switches allowing the processor to select data from multiple data sources.
- To select which data source should be used a multiplexer has one or more control lines (a.ka. Selectors).
- The output of a multiplexer is the selected data.

Overview aka how it do?







Test Cases

```
module MUX5TEST();

reg[4:0] A, B;
reg S;
reg clk;
wire O;

MUX5 DUT(O, A, B, S);

always
    #1 clk = ~clk;

initial
begin
    clk = 1'b0;
    A = 5'h00;
    B = 5'h00;
    S = 1'b0;

    @(negedge clk)
        A = 5'h00;
        B = 5'h00;
    @(negedge clk)
        A = 5'h0F;
        B = 5'h00;
    @(negedge clk)
        A = 5'h00;
        B = 5'h0F;
    @(negedge clk)
        A = 5'h10;
        B = 5'h00;
    @(negedge clk)
        A = 5'h00;
        B = 5'h10;
    @(negedge clk)
        A = 5'h1F;
        B = 5'h00;
    @(negedge clk)
        S = 1'b1;
        A = 5'h00;
        B = 5'h00;
    @(negedge clk)
        A = 5'h01;
        B = 5'h00;
    @(negedge clk)
        A = 5'h00;
        B = 5'h01;
    @(negedge clk)
        A = 5'h01;
        B = 5'h01;
    @(negedge clk)
        A = 5'h06;
        B = 5'h00;
    @(negedge clk)
        A = 5'h00;
        B = 5'h06;
    @(negedge clk)
        A = 5'h06;
        B = 5'h06;
    @(negedge clk)
        A = 5'h1A;
        B = 5'h00;
    @(negedge clk)
        A = 5'h00;
        B = 5'h1A;
    @(negedge clk)
        A = 5'h1A;
        B = 5'h1A;
    @(negedge clk)
        A = 5'h1F;
        B = 5'h00;
    @(negedge clk)
        $finish;
end

always @(A or B or S)
    #1 $display("At t=%T", $time,"|", "S=%b A=%b B=%b O=%b", S, A, B, O);

endmodule
```

```
module multiplexer_test();

reg A, B, S;
wire F;

multiplexer DUT(A, B, S, F);

initial
begin
    A <= 0;
    B <= 0;
    S <= 0;
    #11
    $display("F = %d", F);
    A <= 0;
    B <= 0;
    S <= 1;
    #11
    $display("F = %d", F);
    A <= 0;
    B <= 1;
    S <= 0;
    #11
    $display("F = %d", F);
    A <= 0;
    B <= 1;
    S <= 1;
    #11
    $display("F = %d", F);
    A <= 1;
    B <= 0;
    S <= 0;
    #11
    $display("F = %d", F);
    A <= 1;
    B <= 0;
    S <= 1;
    #11
    $display("F = %d", F);
    A <= 1;
    B <= 1;
    S <= 0;
    #11
    $display("F = %d", F);
    A <= 1;
    B <= 1;
    S <= 1;
    #11
    $display("F = %d", F);
end

endmodule
```

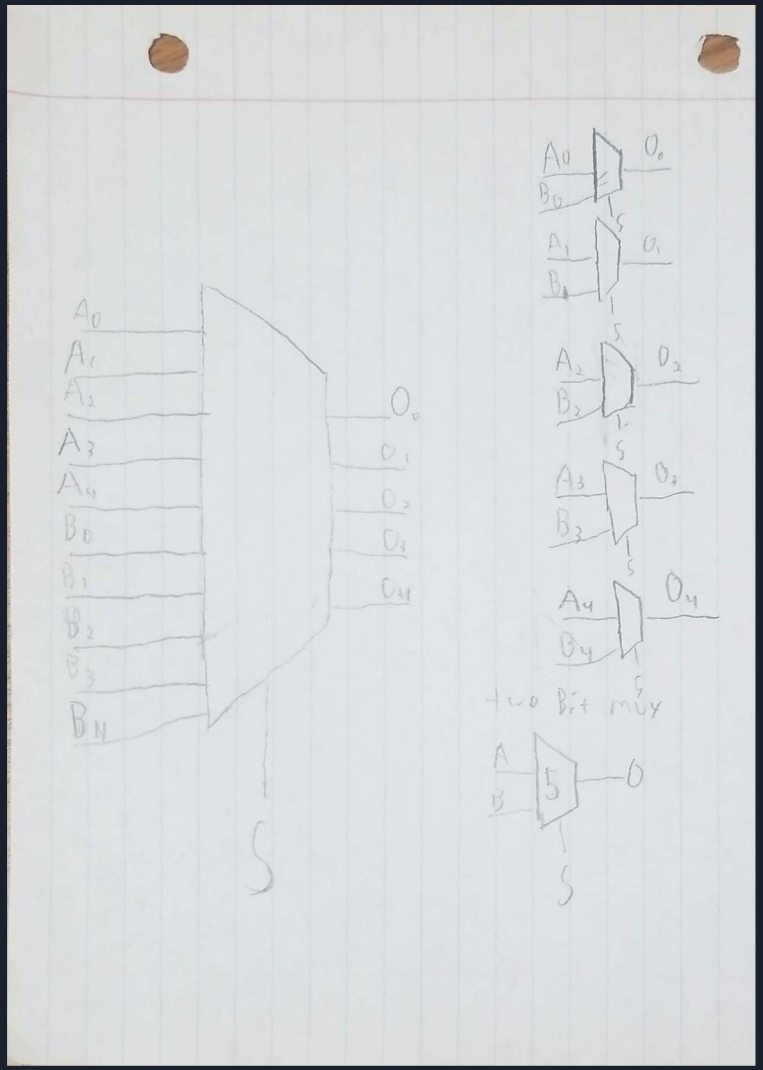
Select	Inputs		Output
0	0	0	0
0	0	1	1
1	1	0	1
1	1	1	1

Issues with Muxes Implementations

- One of the most common implementation problems comes with moving a completed MUX into a full processor. When implementing a MUX into a full processor you can get an unexpected “x” value when it was working by itself.

- Another issue is mislabeling a variable and not listing it as a register then it will produce an output that tells you that the register you are trying to use does not exist.

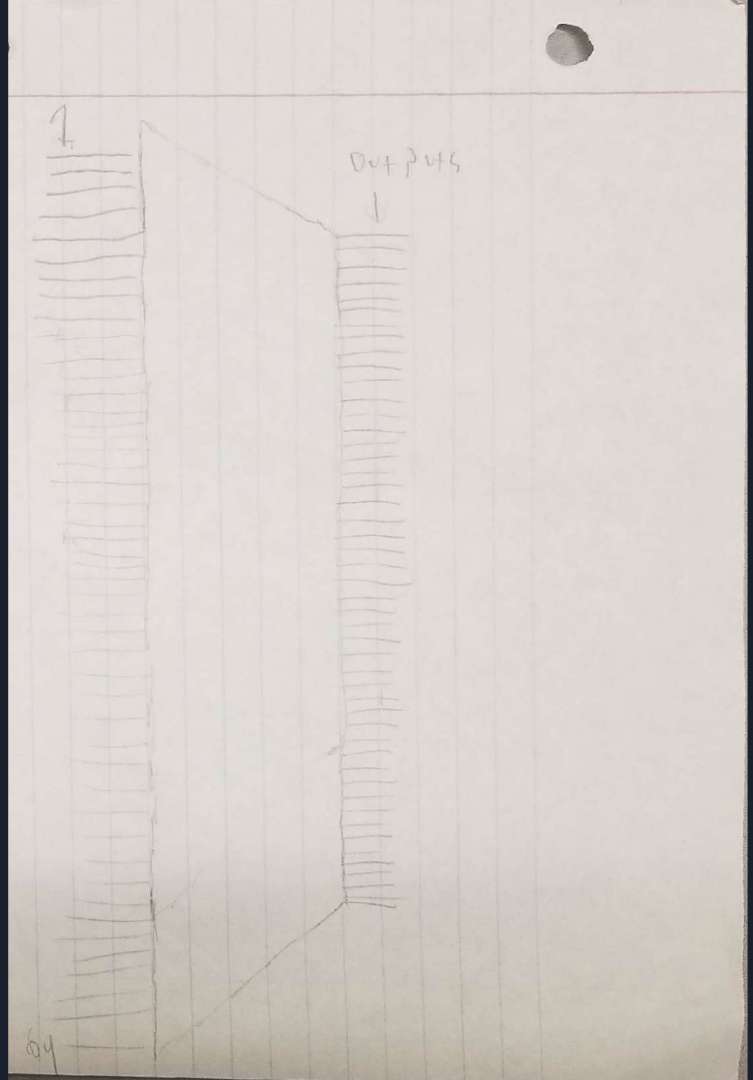
- The other issues can either stem from misspelling or missing a line of code which would make your mux no longer work and integration into a single cycle processor impossible.



Gate Delay - 5 Bit Mux

Because of parallelism, muxes work all at the same time.

This means that the Gate delay is 5, like a 2 bit mux



Gate delay 64 bit mux

The 64 bit mux is the same concept as the 5 bit.

Everything works in parallel, so the gate delay is 5.