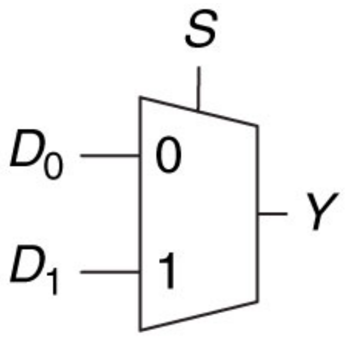
Multiplexors

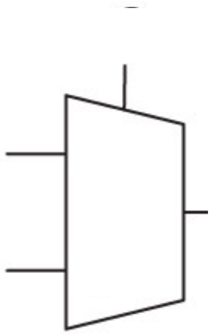
Adapted from slides by Jared Moore

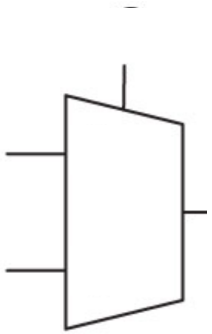
Multiplexer (*mux*)

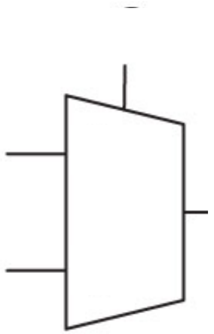
Choose an output from one of several possible inputs based on a select signal

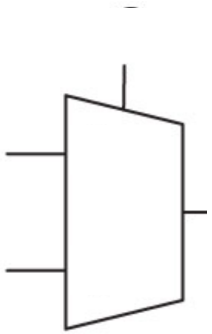


Multiplexor does not perform any operations on the inputs – simply chooses one or the other

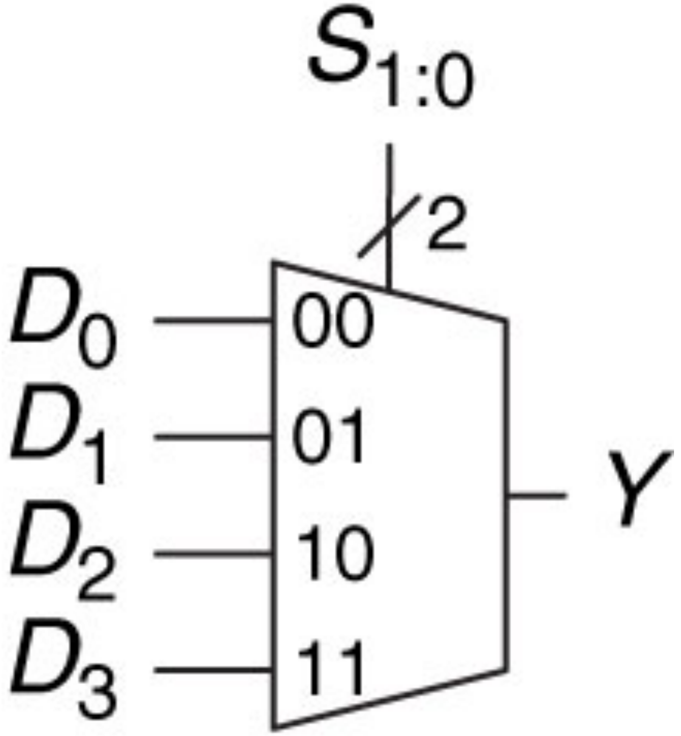








Larger multiplexers



Larger multiplexers

Can make multiplexors as large as we want

For every additional selector bit, number of data inputs doubles

- This should remind you of truth tables, binary numbers, etc.

Choosing with Muxes

Hardware equivalent of “if” statement, except it sometimes appears *after* work is done

Consider how you could implement the following in hardware:

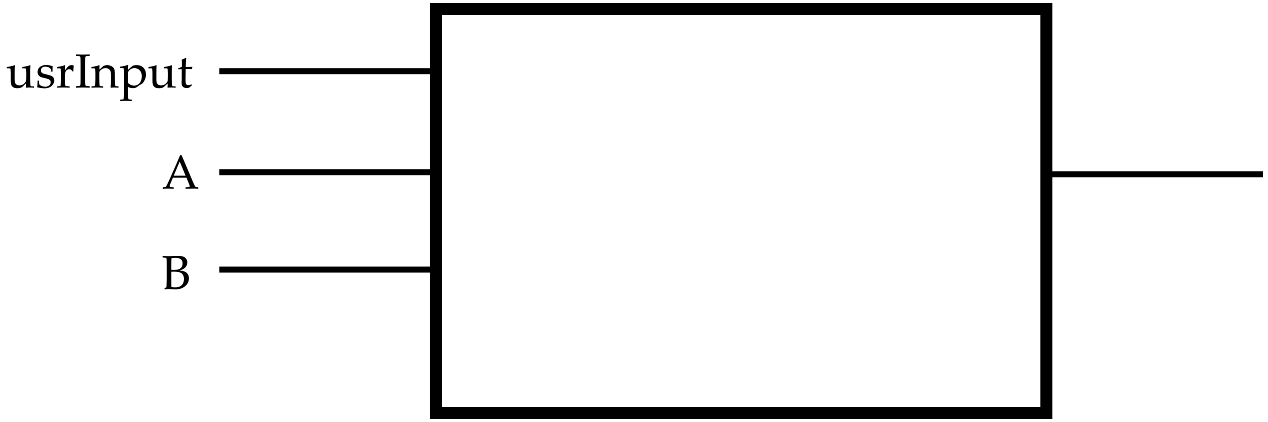
```
if (usrInput == 0)
   return (A and B)
else if (usrInput == 1)
   return (A or B)
```

Choosing with Muxes

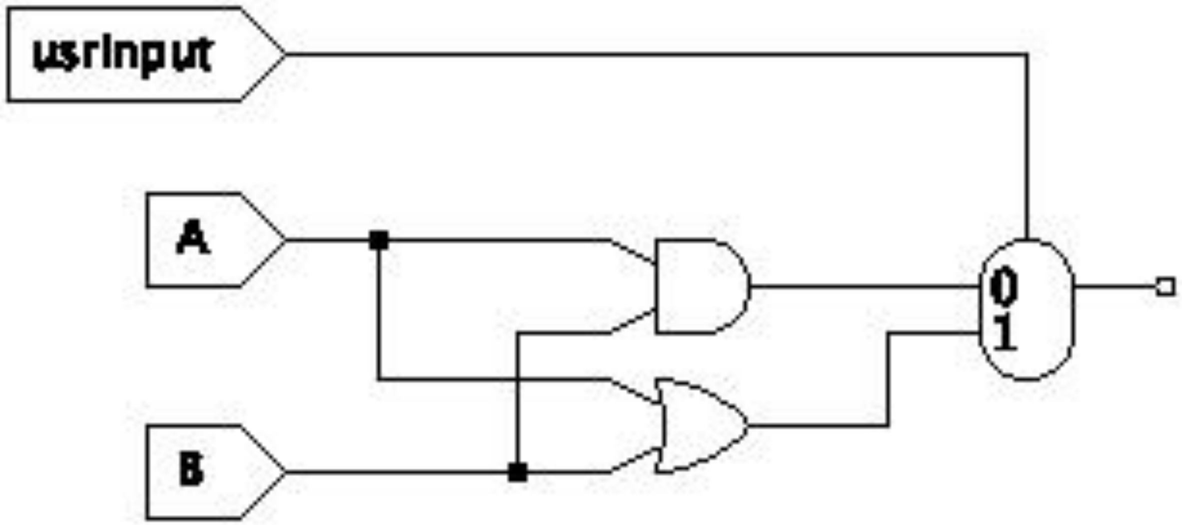
Hardware equivalent of "if" statement, except it sometimes appears *after* work is done

Consider how you could implement the following in hardware:

```
if (usrInput == 0)
   return (A and B)
else if (usrInput == 1)
   return (A or B)
```



```
if (usrInput == 0)
    return (A and B)
else if (usrInput == 1)
    return (A or B)
```



Choosing with Muxes

You can compute everything you might need, then choose between the outputs

Multiplexers to implement logic functions

A second use of multiplexors is implementing logic functions

Essentially, make the mux a “hardware truth table”

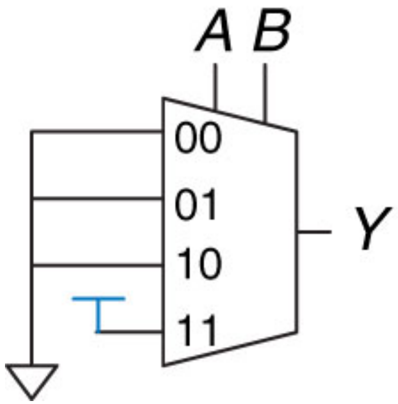
Multiplexers to implement logic functions

Inputs to logic function are *selectors* for mux

For corresponding outputs, tie 0 to ground and 1 to V_{cc}

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

$Y = AB$



Multiplexers to implement logic functions

Using muxes to implement logic functions has same downside as regular truth table – gets very large as number of inputs grows

Muxes as truth tables

To implement $Y = AB' + B'C' + A'BC$ using a mux, how many inputs would be required for

- data?
- selection?

Muxes as truth tables

Implement $Y = AB' + B'C' + A'BC$ with an 8:1 multiplexer.

Hint: it will be helpful to make the truth table first

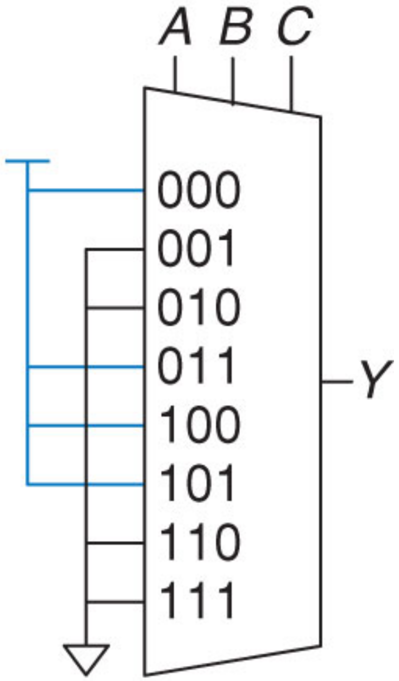
Muxes as truth tables

Implement Y = AB′ + B′C′ + A′BC with an 8:1 multiplexer.

A	B	C	Y
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

$Y = A\overline{B} + \overline{B}\,\overline{C} + \overline{A}BC$

(a)



(b)

How to build a mux?

Multiplexor output depends only on its inputs (D_0, D_1, and S)

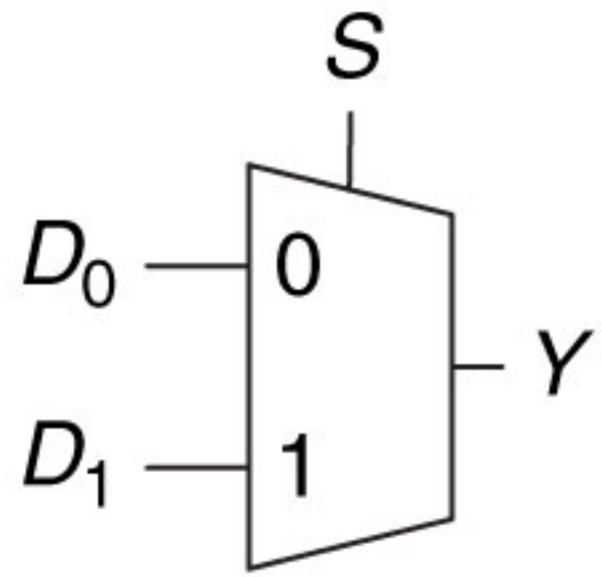
It is therefore a combinational circuit

This means that

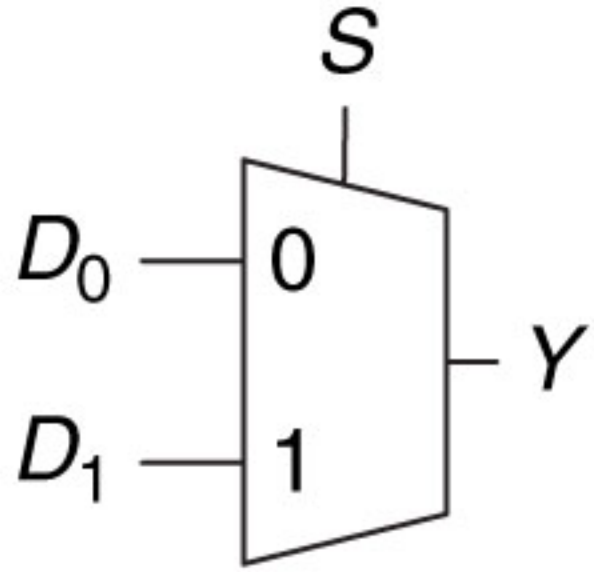
- we can write a truth table for a mux, and
- we can build a mux out of AND/OR/NOT gates

Multiplexer (*mux*) truth table

How many inputs and outputs are there for the truth table?

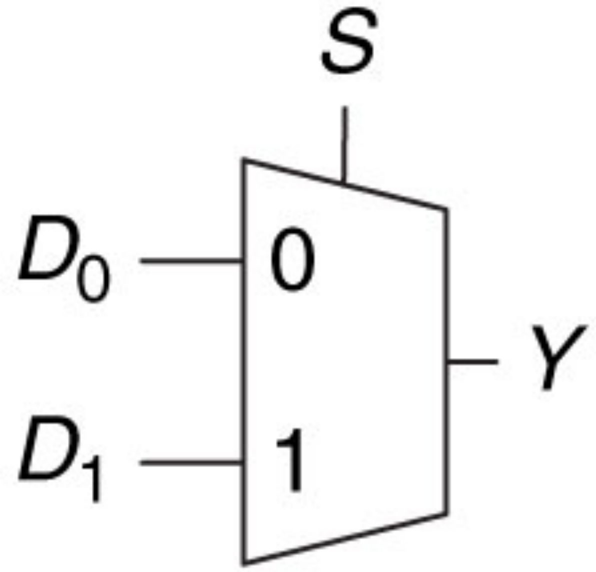


Multiplexer (*mux*) truth table



S	D_1	D_0	Y
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Multiplexer (*mux*) truth table



S	D_1	D_0	Y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

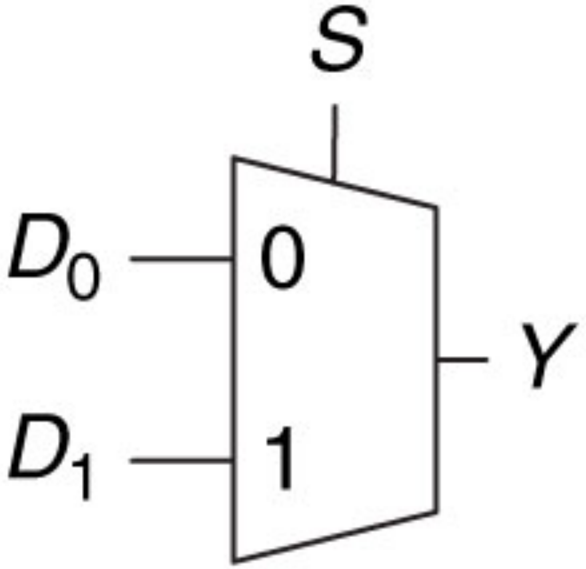
Mux from logic gates

As always, could write the mux in PLA-style

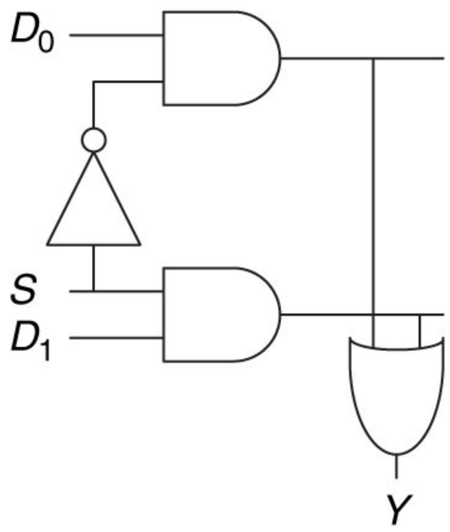
But, as usual, we can do better

S	D_1	D_0	Y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Mux from logic gates

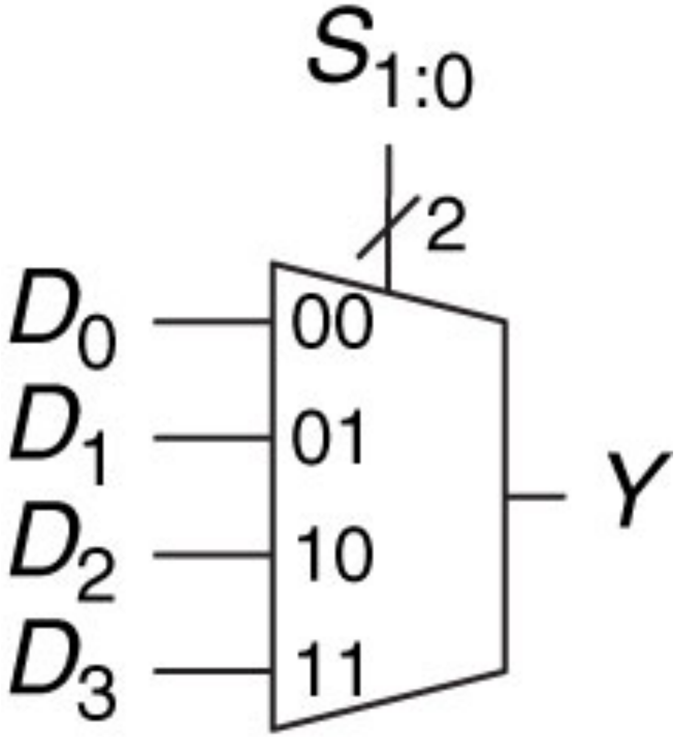


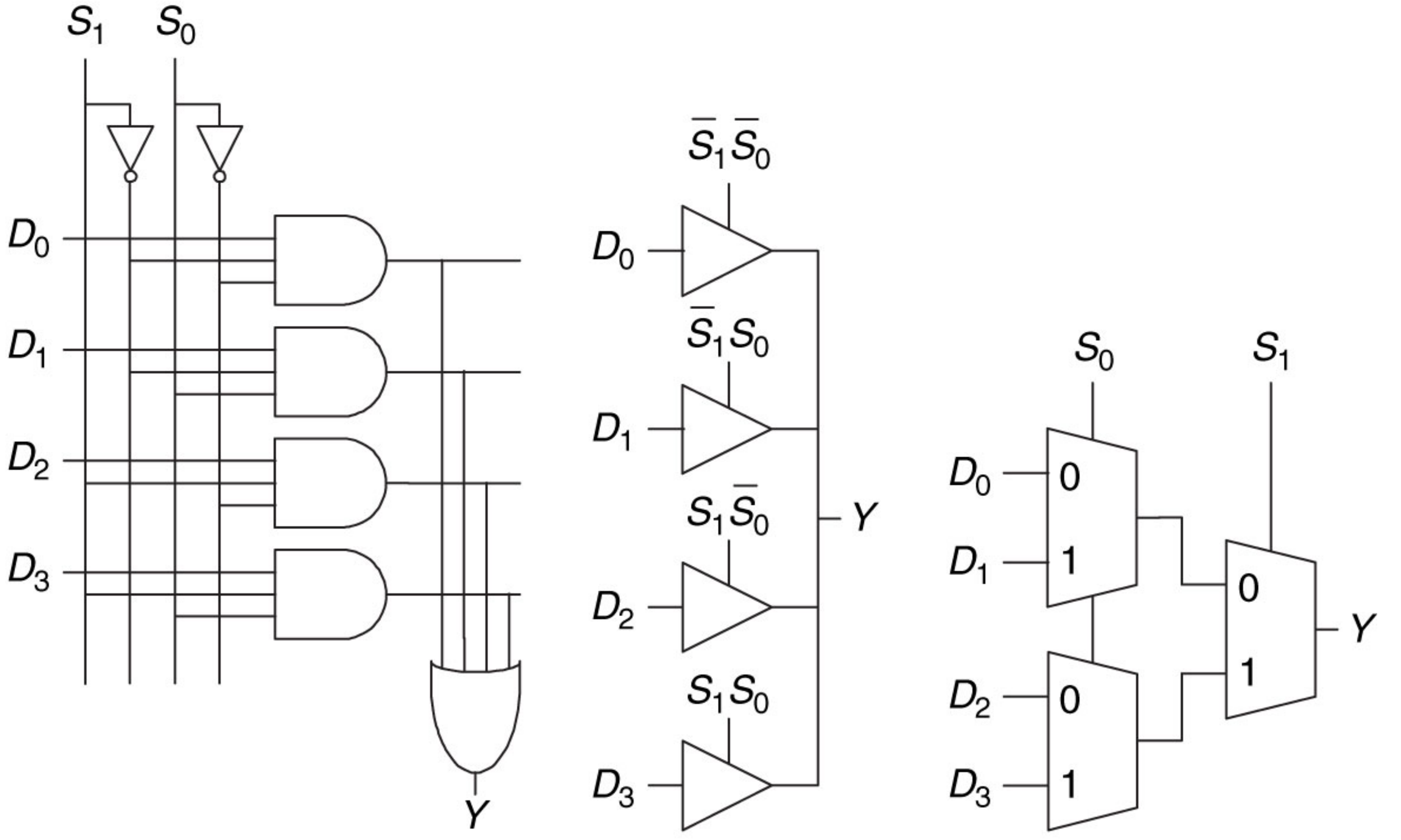
$$Y = D_0\overline{S} + D_1 S$$



S	D_1	D_0	Y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Building larger multiplexers





(a) (b) (c)

Summary

Multiplexors simply choose between their data inputs using a selector input

They are used

- as “if statements” to choose the output of a circuit, or
- as “truth tables” to implement logic functions

Muxes can choose between more than two things if we add more selector bits

We can implement muxes from standard gates