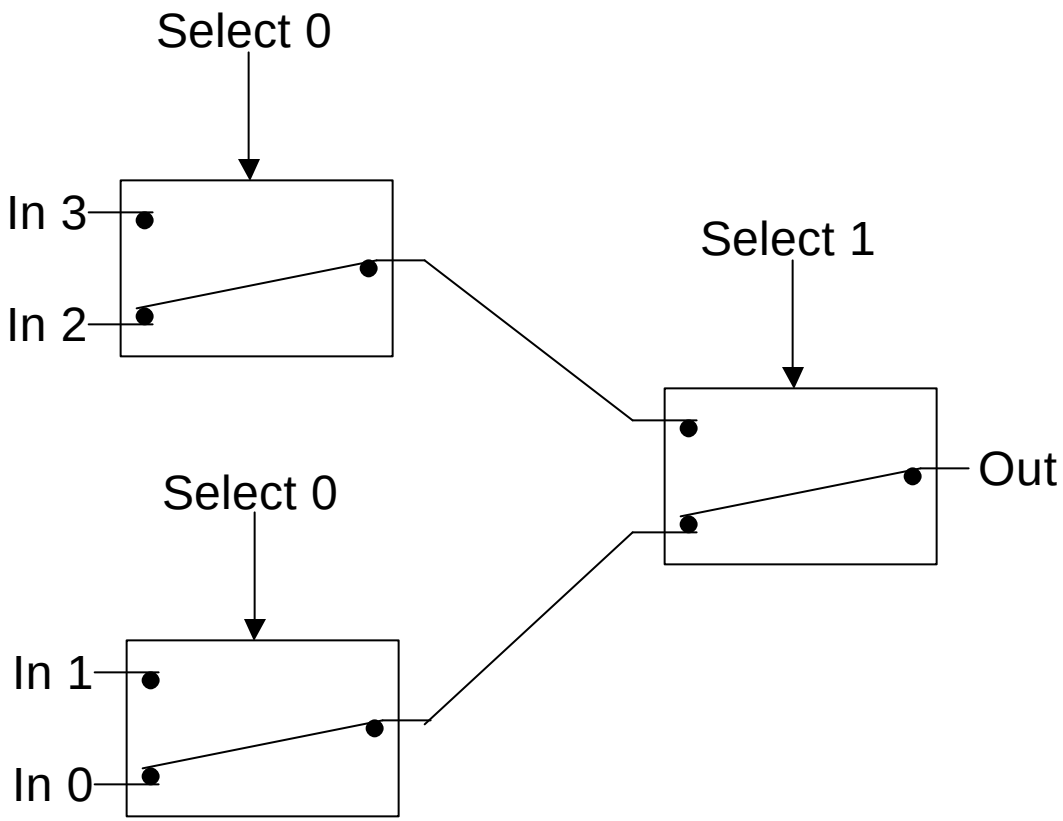
April 3

- Fun with Muxes
- Implementing arbitrary logical functions
- Finite State Machines

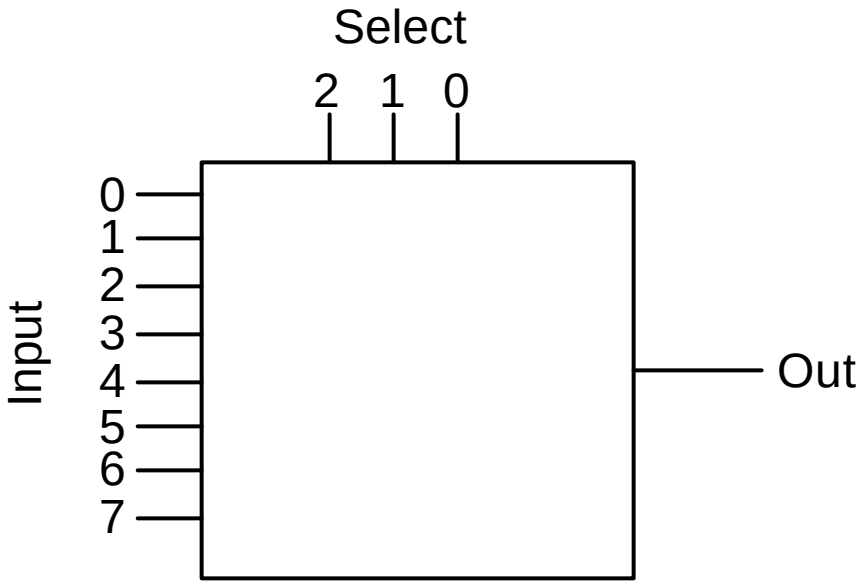
Fun with MUXes

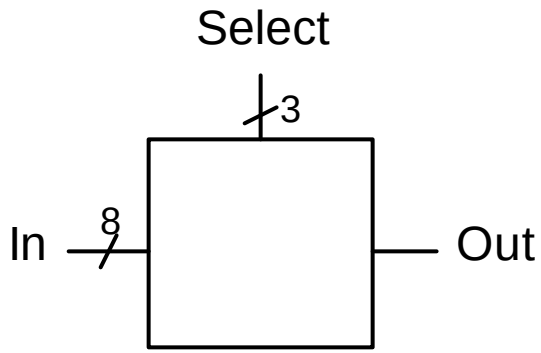


Remember the MUX?

This will route 1 of 4 different 1 bit values to the output.

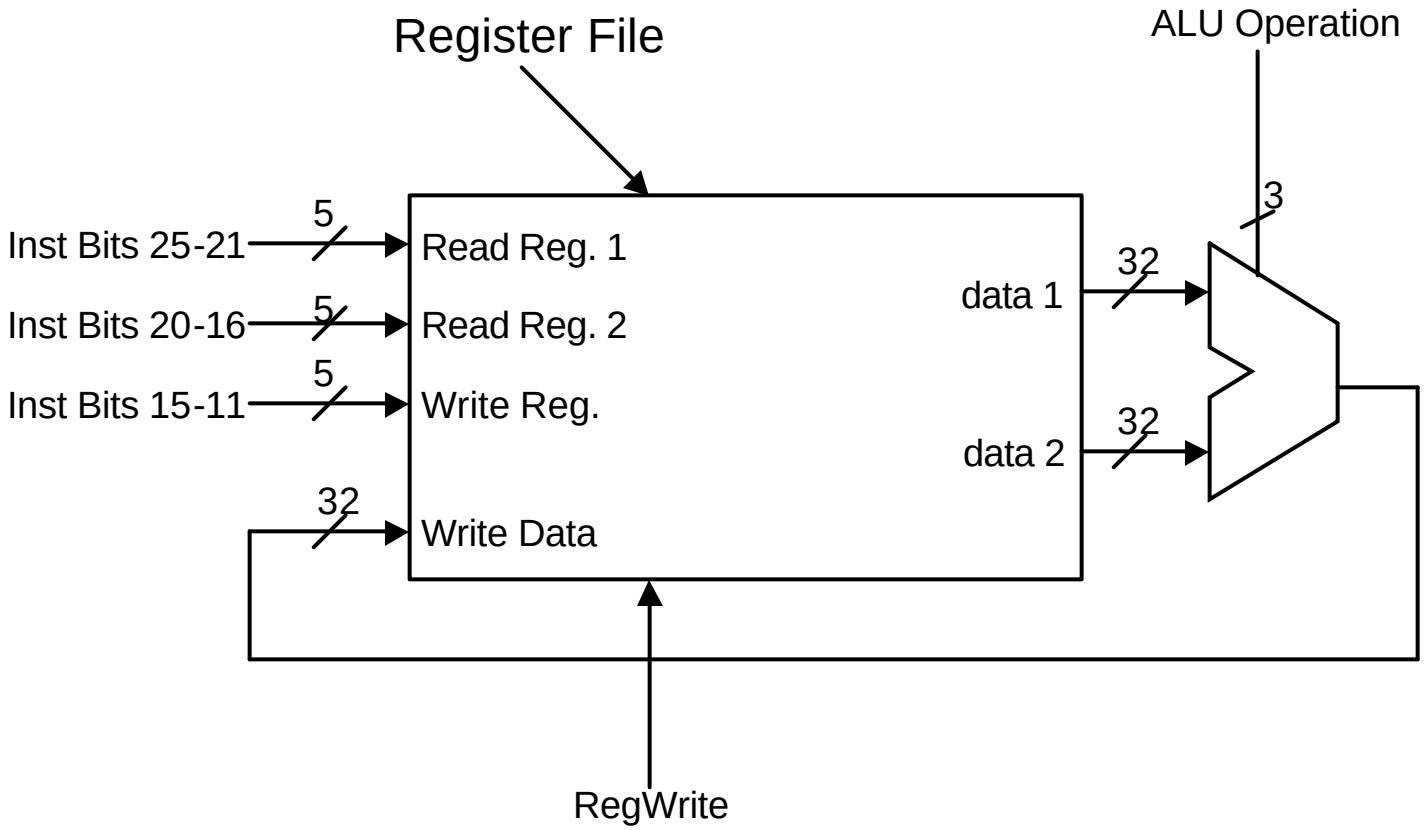
MUX Blocks



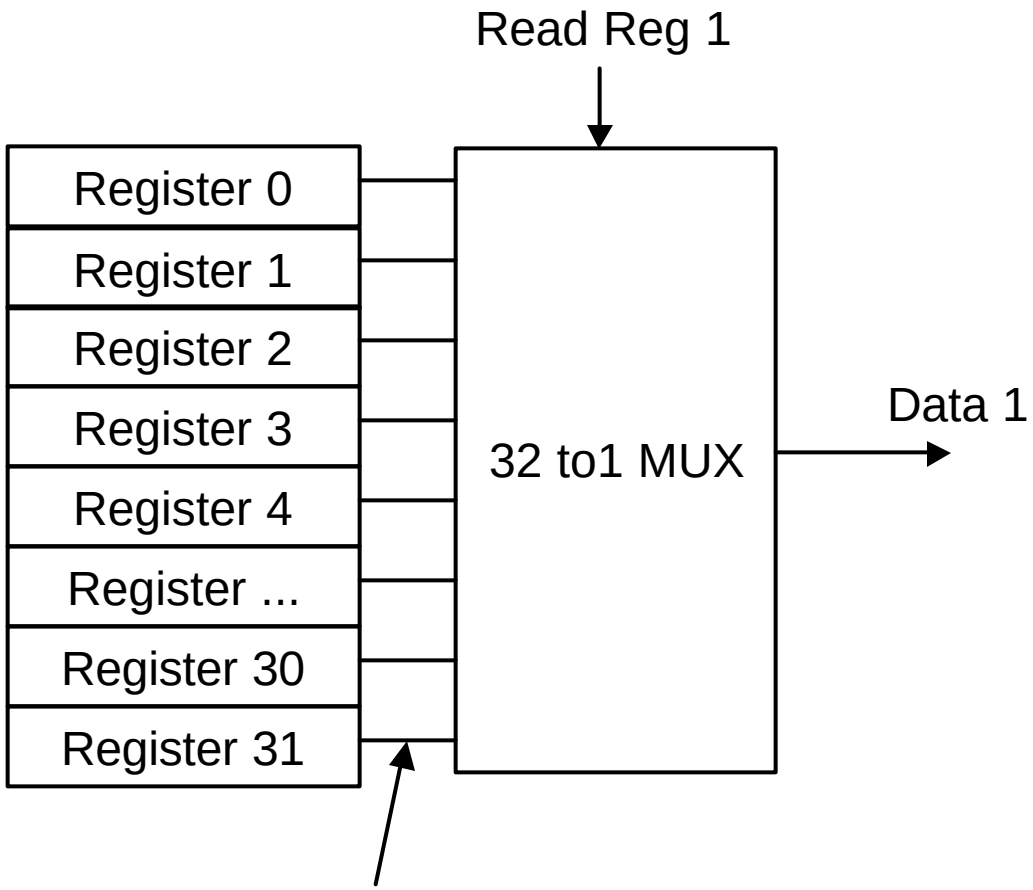


The select signal determines which of the inputs is connected to the output

Remember the Register File?



Inside there is a 32 way MUX per bit



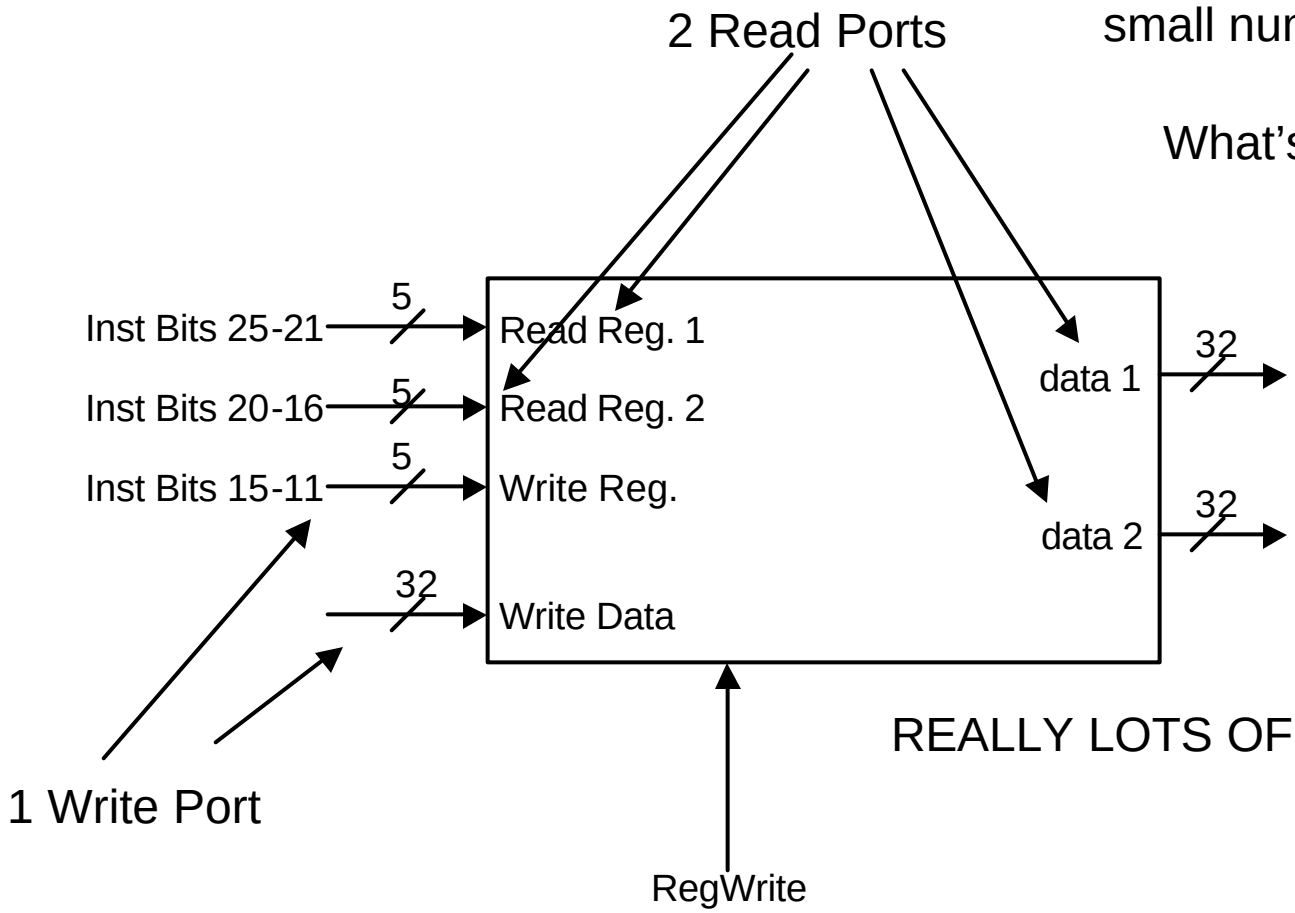
LOT'S OF CONNECTIONS!

And this is just one port!

Our Register File has 3 ports

This is one reason we have only a small number of registers

What's another reason?

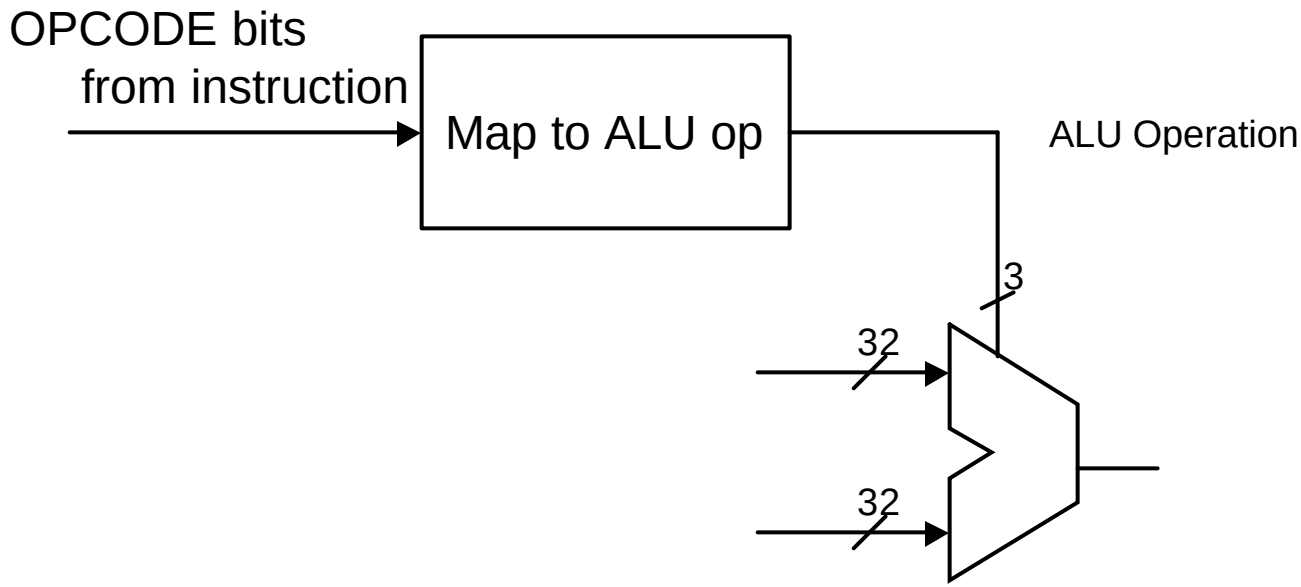


REALLY LOTS OF CONNECTIONS!

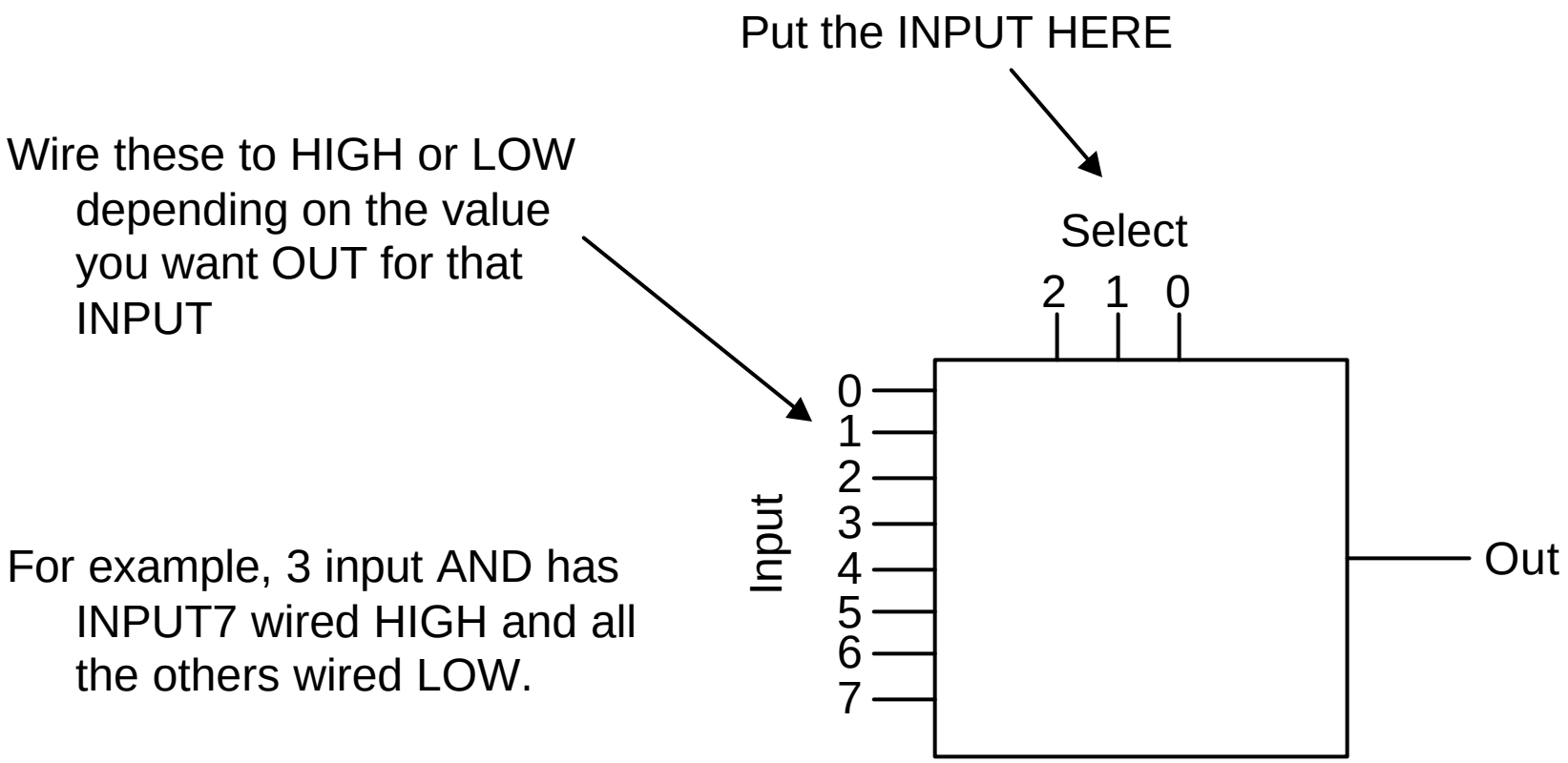
Implementing Logical Functions

Suppose we want to map M input bits to N output bits

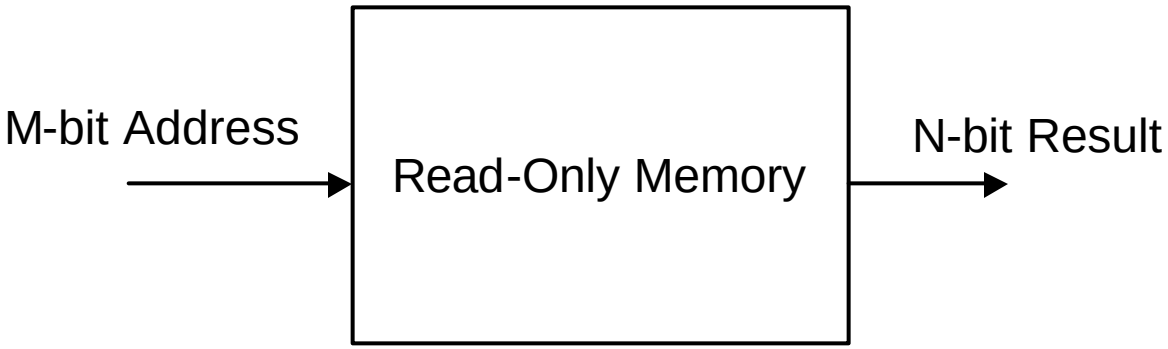
For example, we need to take the OPCODE field from the instruction and determine what OPERATION to send to the ALU.



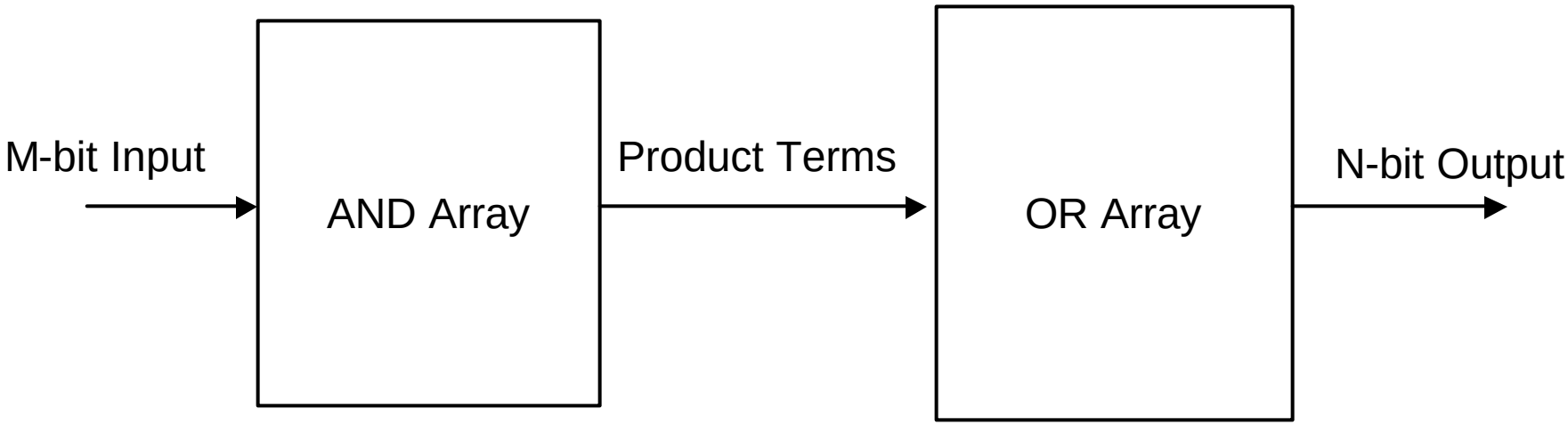
We can get 1 bit out with a MUX



Or use a ROM



Or use a PLA



Think of the SUM of PRODUCTS form.

The AND Array generates the products of various input bits

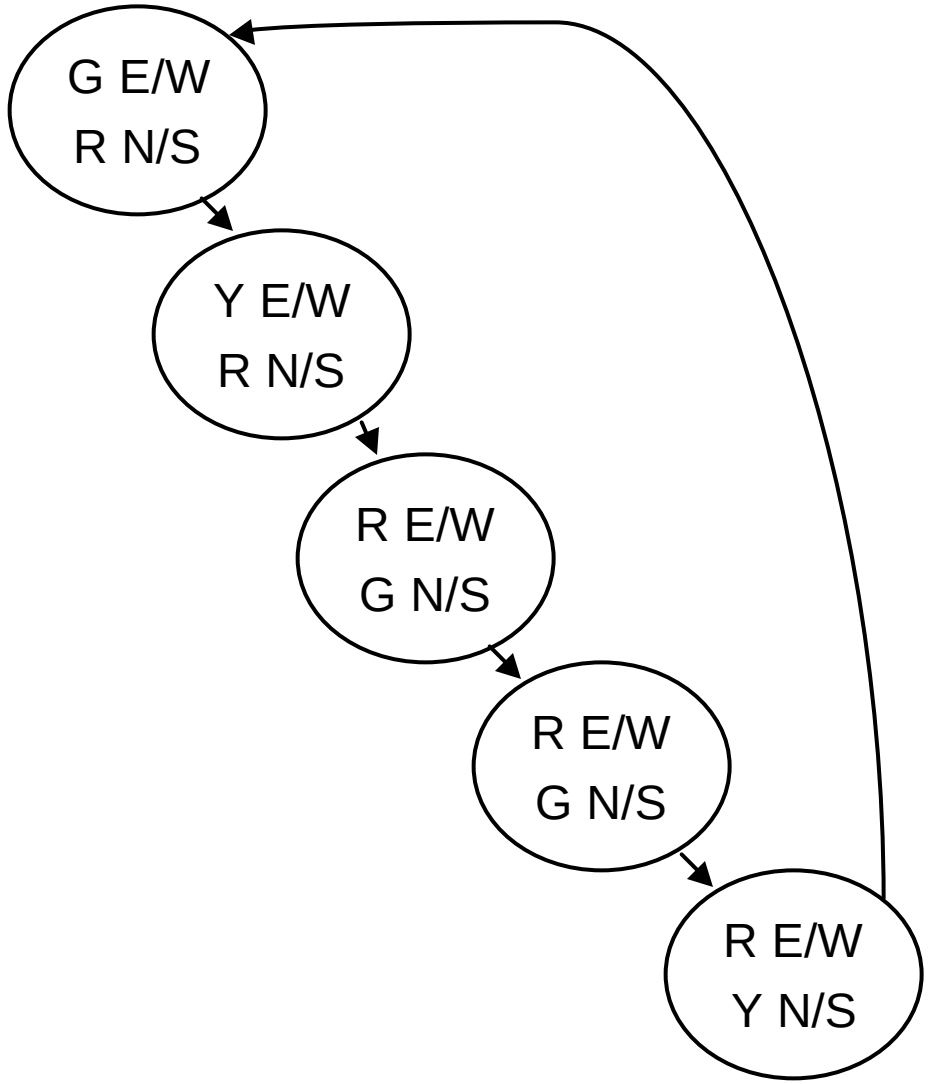
The OR Array combines the products into various outputs

Finite State Machines

- A set of STATES
- A set of INPUTS
- A set of OUTPUTS
- A function to map the STATE and the INPUT into the next STATE and an OUTPUT

Remember “Shoots and Ladders”?

Traffic Light Controller



Implementing a FSM