COS10004 Computer Systems

Lecture 4.3: Let's build a stack!

CRICOS provider 00111D

Dr Chris McCarthy

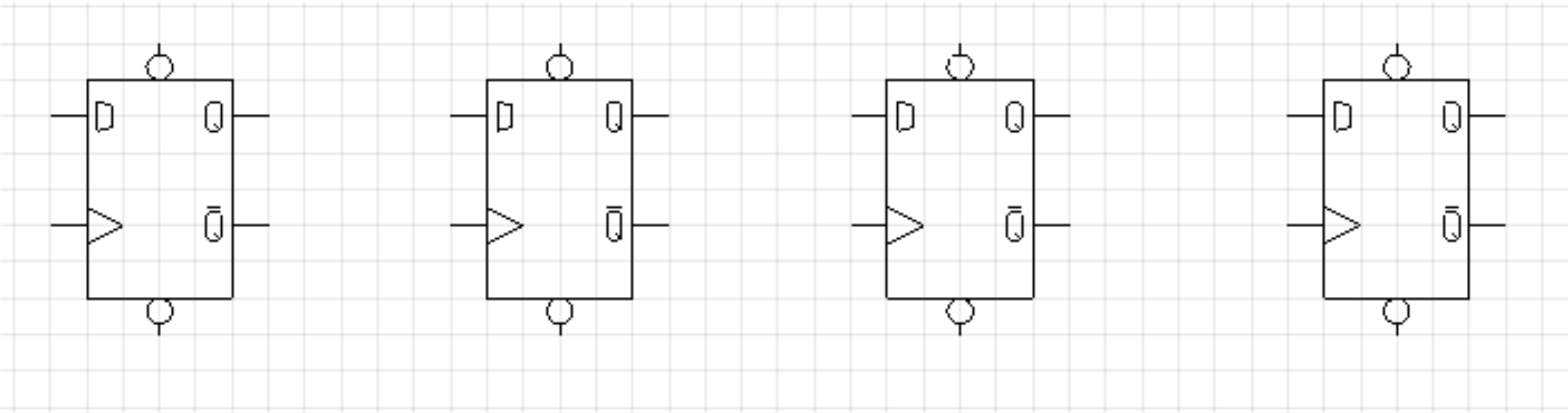
STACKS

- Random access memory requires knowing the address of every byte/word you want to access
- Hardware stacks created out of dedicated shift registers
- So let's build one!

LET'S BUILD A STACK...

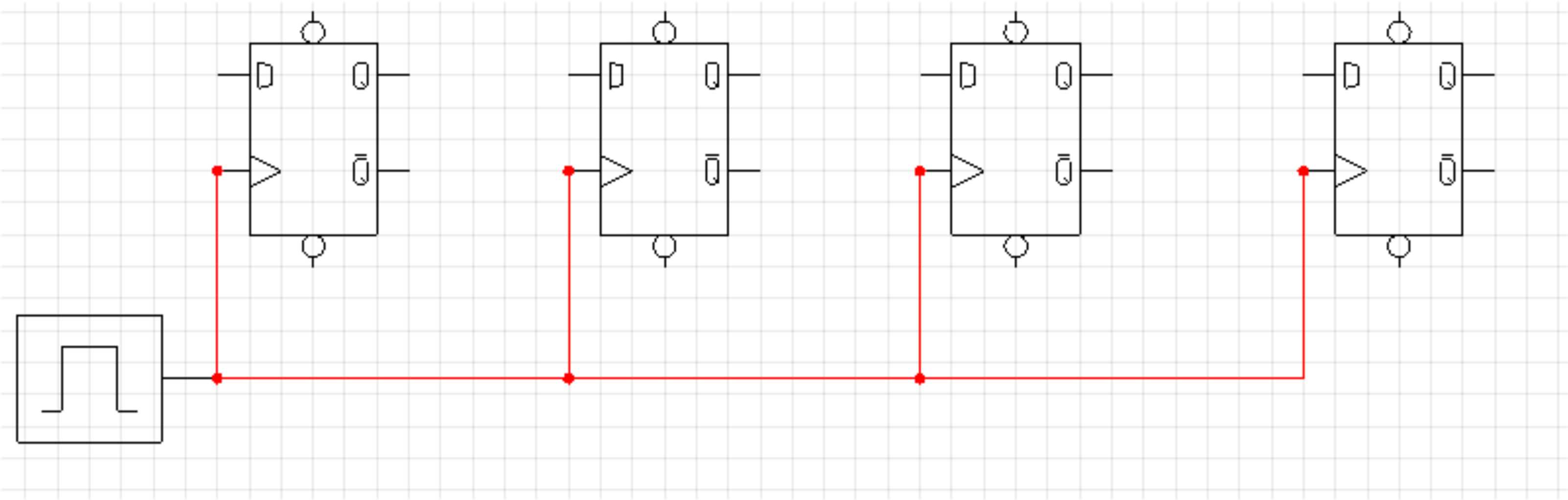
> Start with a bi-directional shift register...

> 1. Start off with 4 D flip-flops:

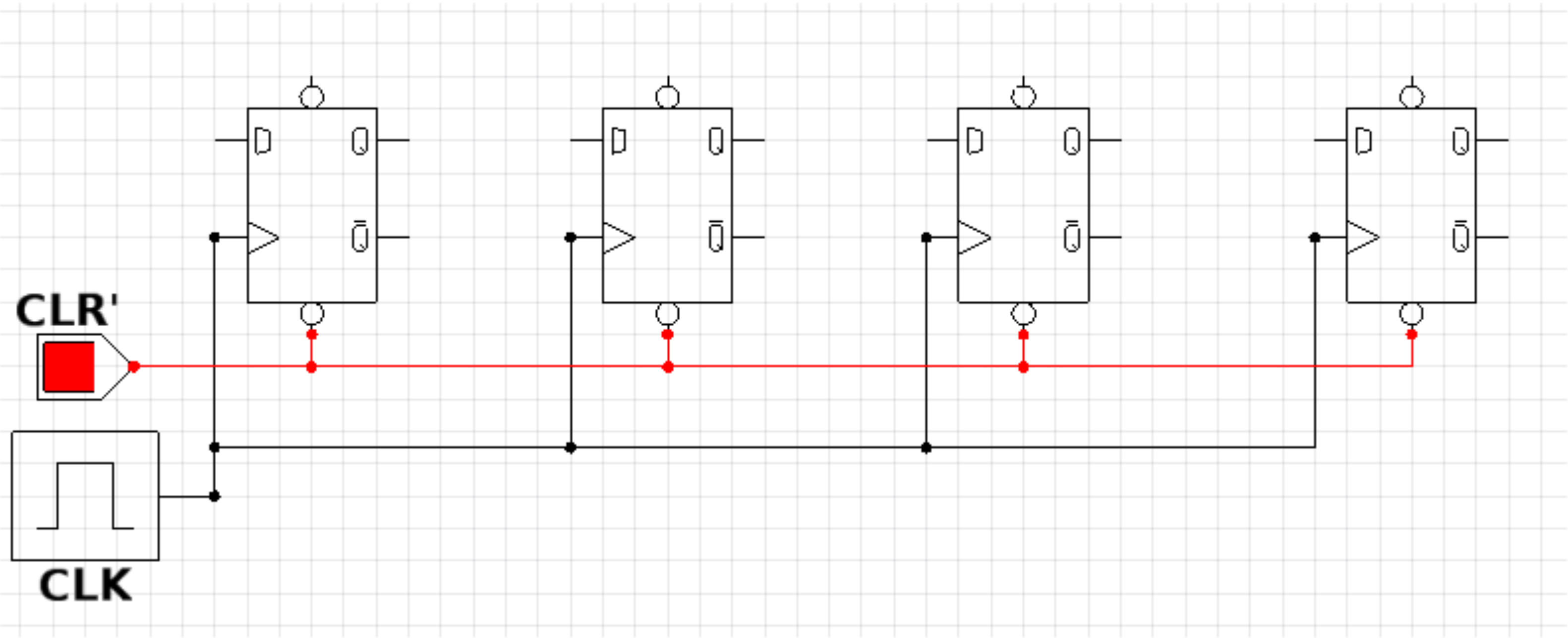


2. ADD THE STANDARD SIGNALS

> Clock

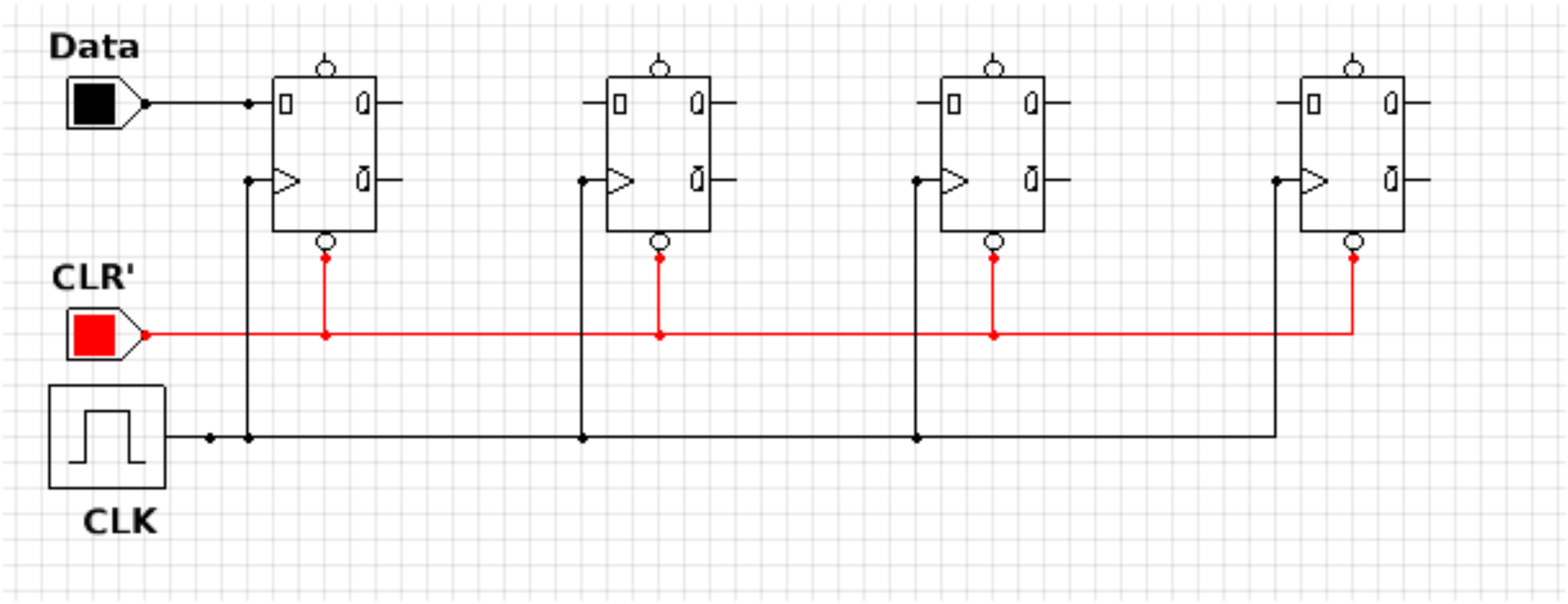


- CLR (resets flip flops when up in Logisim)

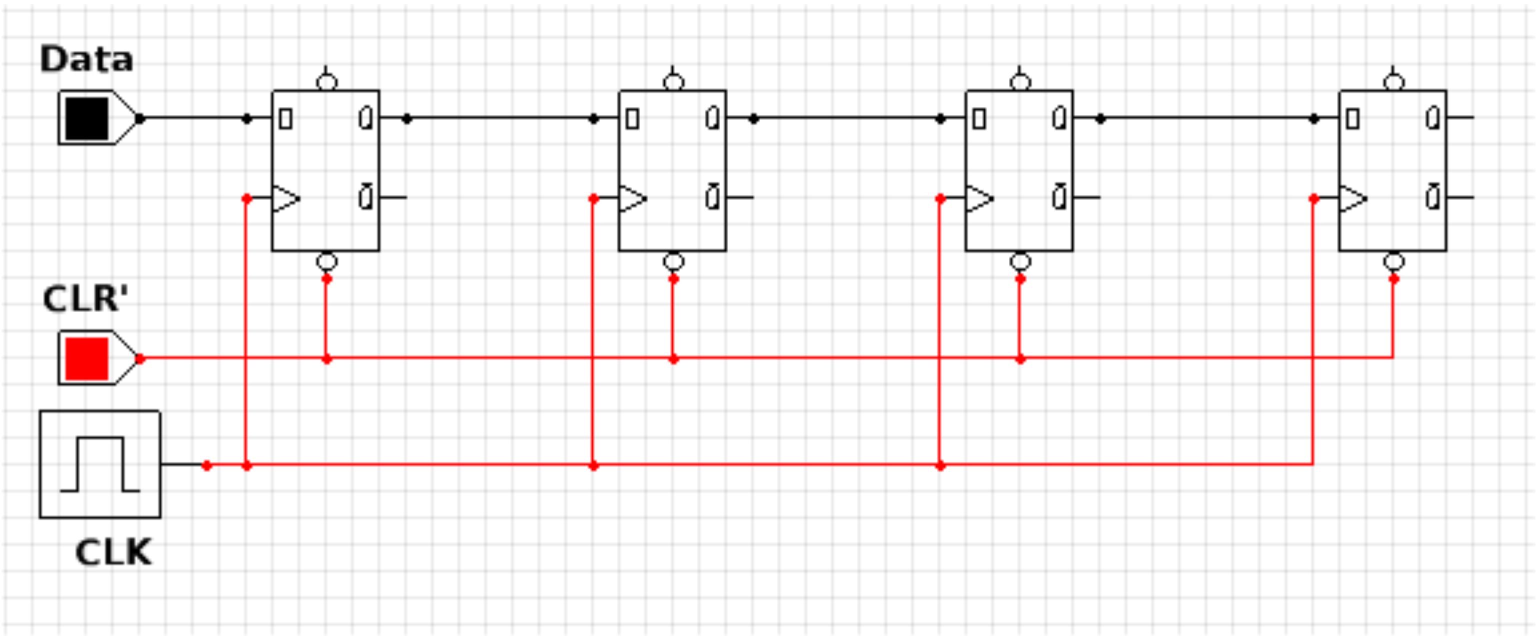


ADD THE SERIAL INPUT (D)

- Data is the serial input (SI)

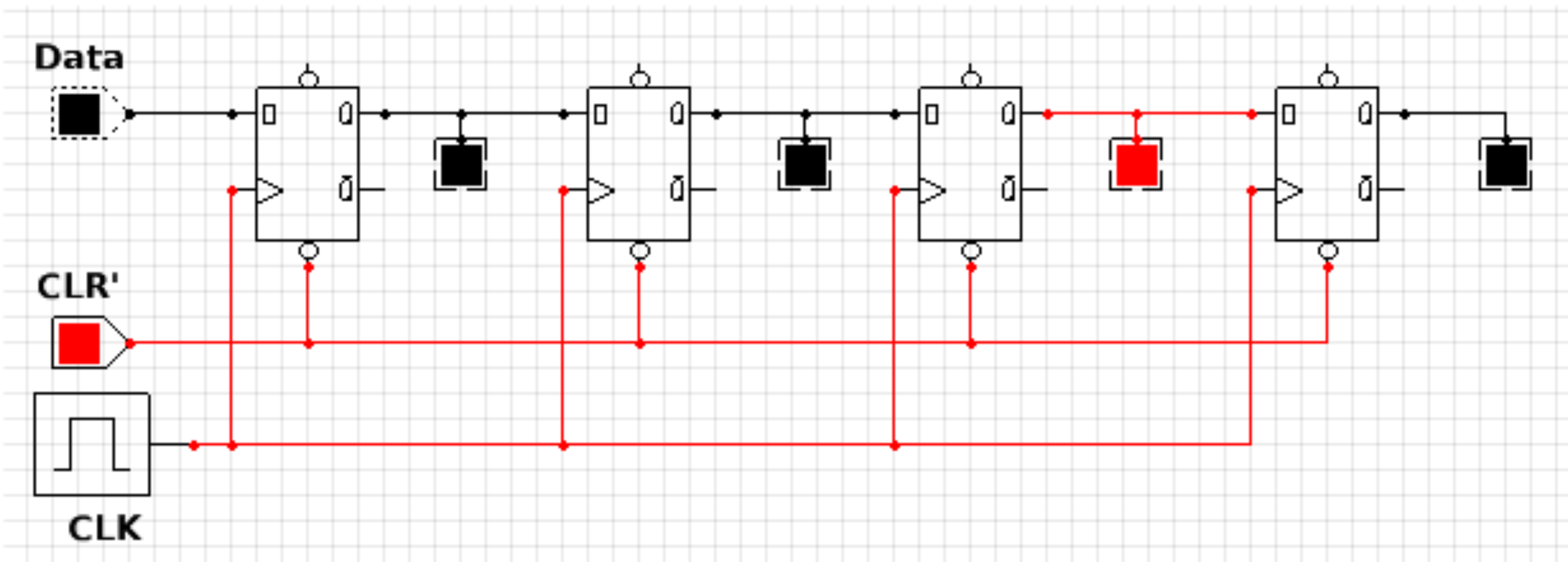


CONNECT EACH Q OUTPUT TO THE NEXT DATA INPUT



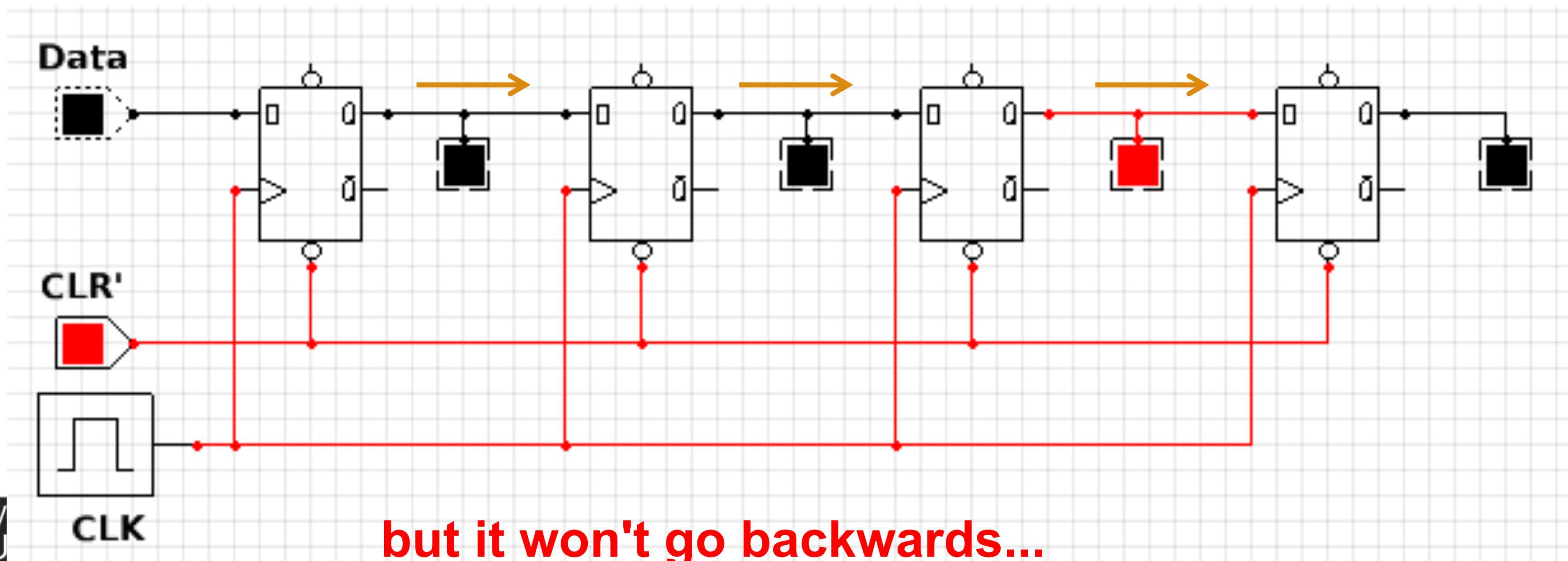
ADD SOME LEDS TO SEE THE PARALLEL OUTPUT / REGISTER STATE

- We can now modulate the Data input and see the **on** state propagate through the register (left to right).



THIS CIRCUIT WILL DO MOST THINGS WE NEED:

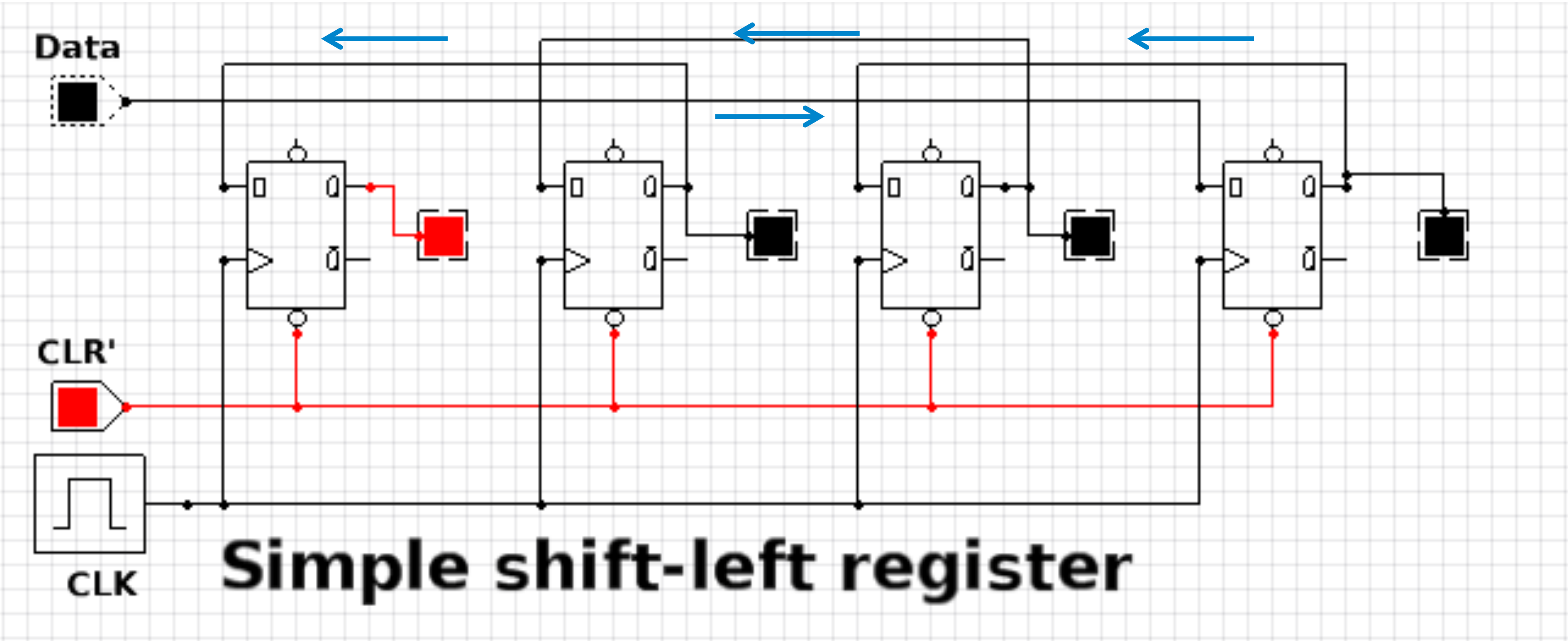
- To get serial out, record the state of the right-most LED.
- To get parallel out, feed each LED into a register (latch) and stop the clock when the conversion is complete.
- To do parallel input, OR each D input with the state of a register and start the clock.



but it won't go backwards...

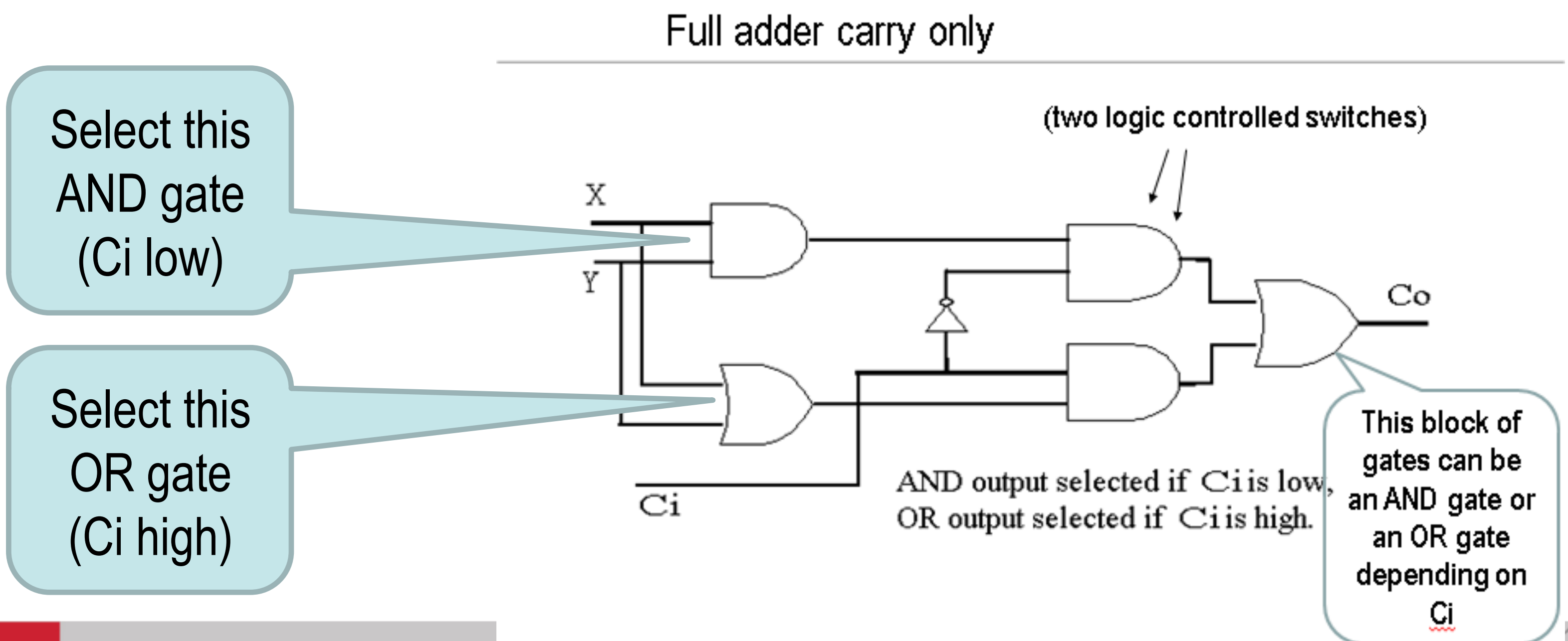
How do we make it go backwards?

- Wire-up the cascade backwards
- Data goes in the far end, each Q outputs to the D input of the previous Flip-Flop.



How do we make the direction selectable?

> Remember those controlled logic gates in Week 2 ?

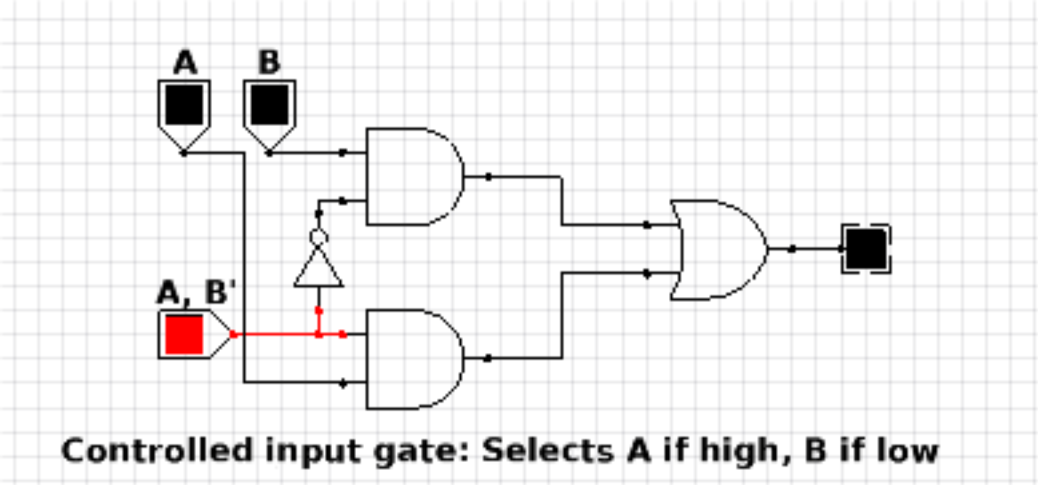


How do we make the direction selectable?

- More specifically, we need to determine from which direction each Flip Flop will receive its input.

How do we make the direction selectable?

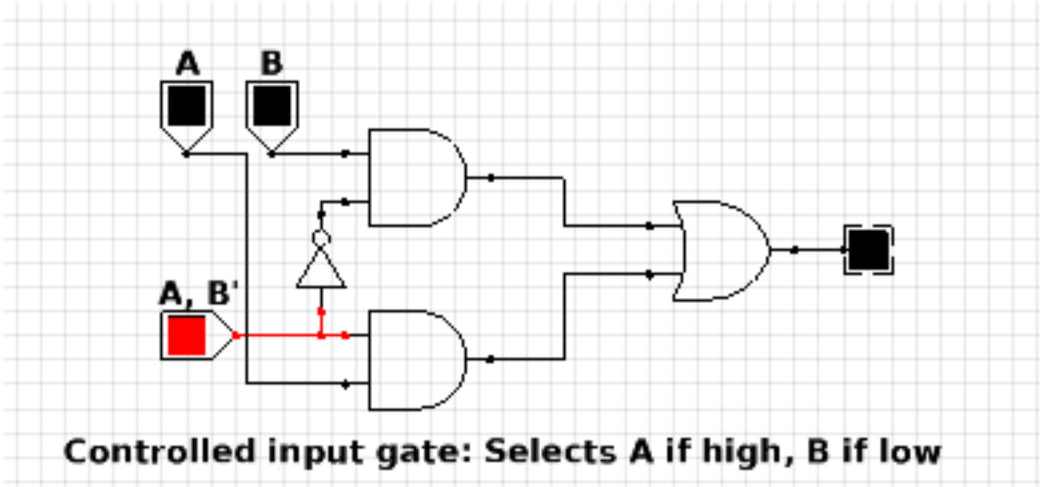
- A controlled gate (remember Week 2!).



PROGRAMMABLE WIRING

> We can use one of these circuits for each input to a Flip-Flop, and use a common control signal to determine direction.

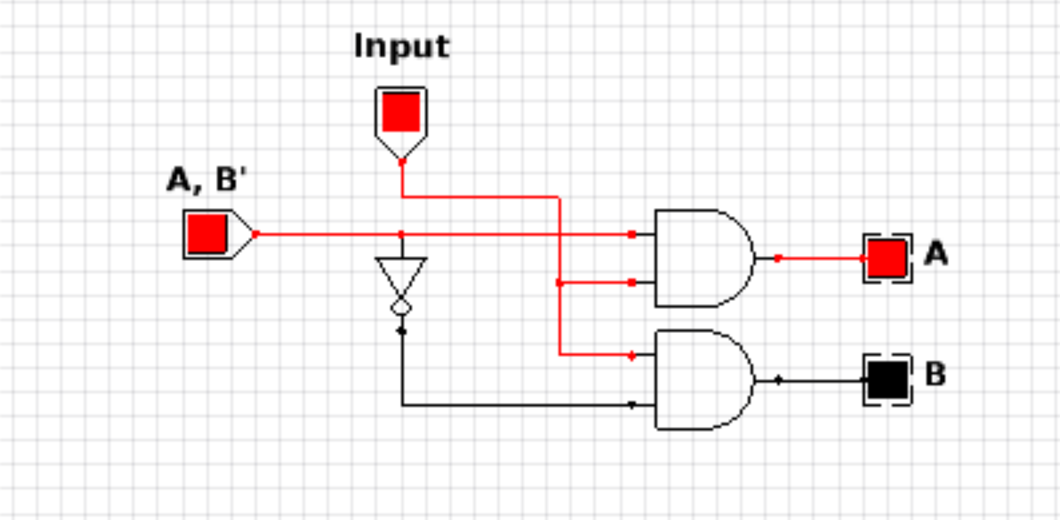
A, B'	Output
1	A
0	B



SELECTING THE OUTPUT WITH GATES

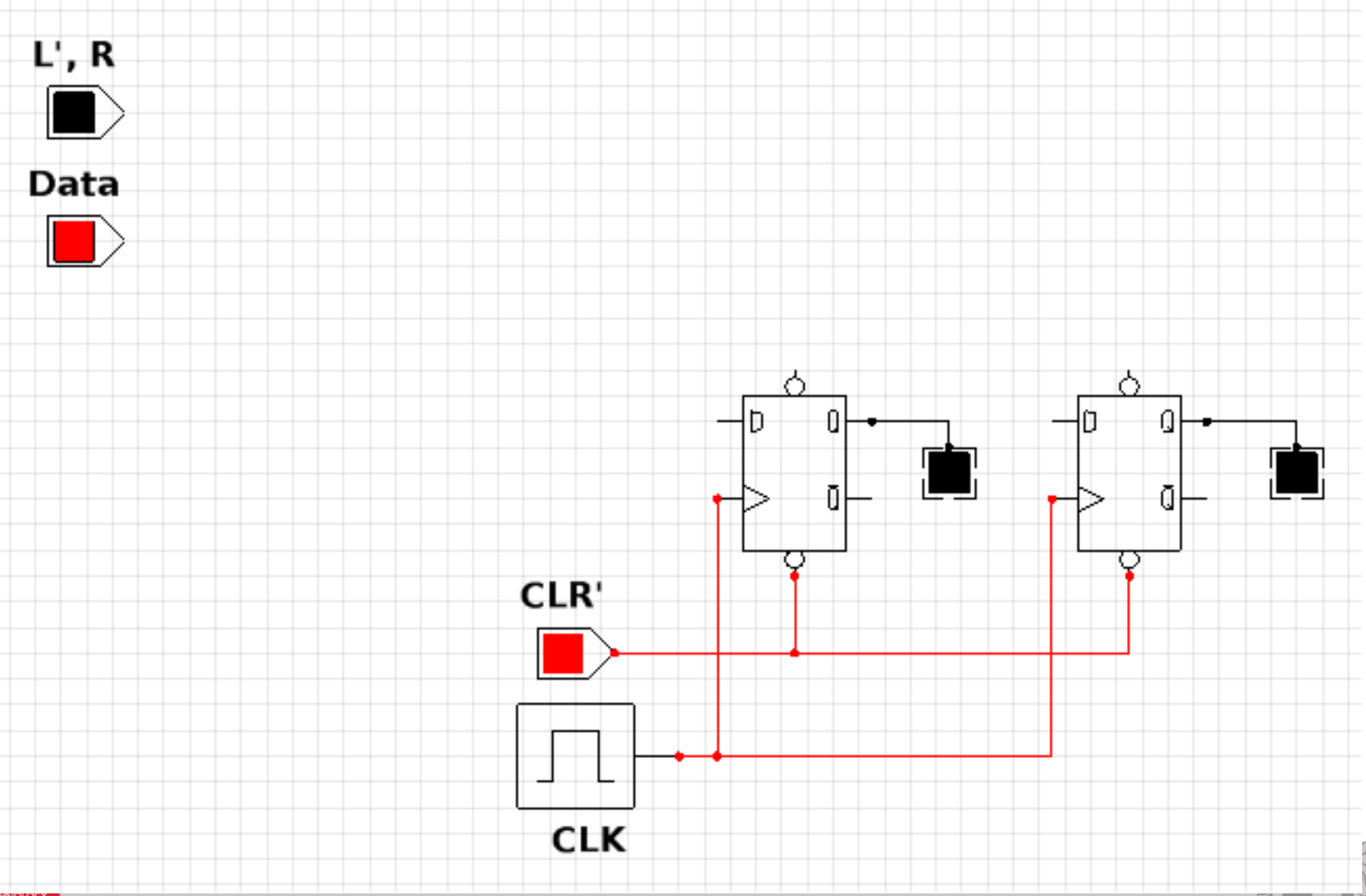
- We also need to determine which direction output from a Flip Flop flows!
- This circuit has a common input, and selectable output.
- We can use one of these for each output from a Flip-Flop, and use a common control signal to determine direction.

A, B'	Input
1	A
0	B

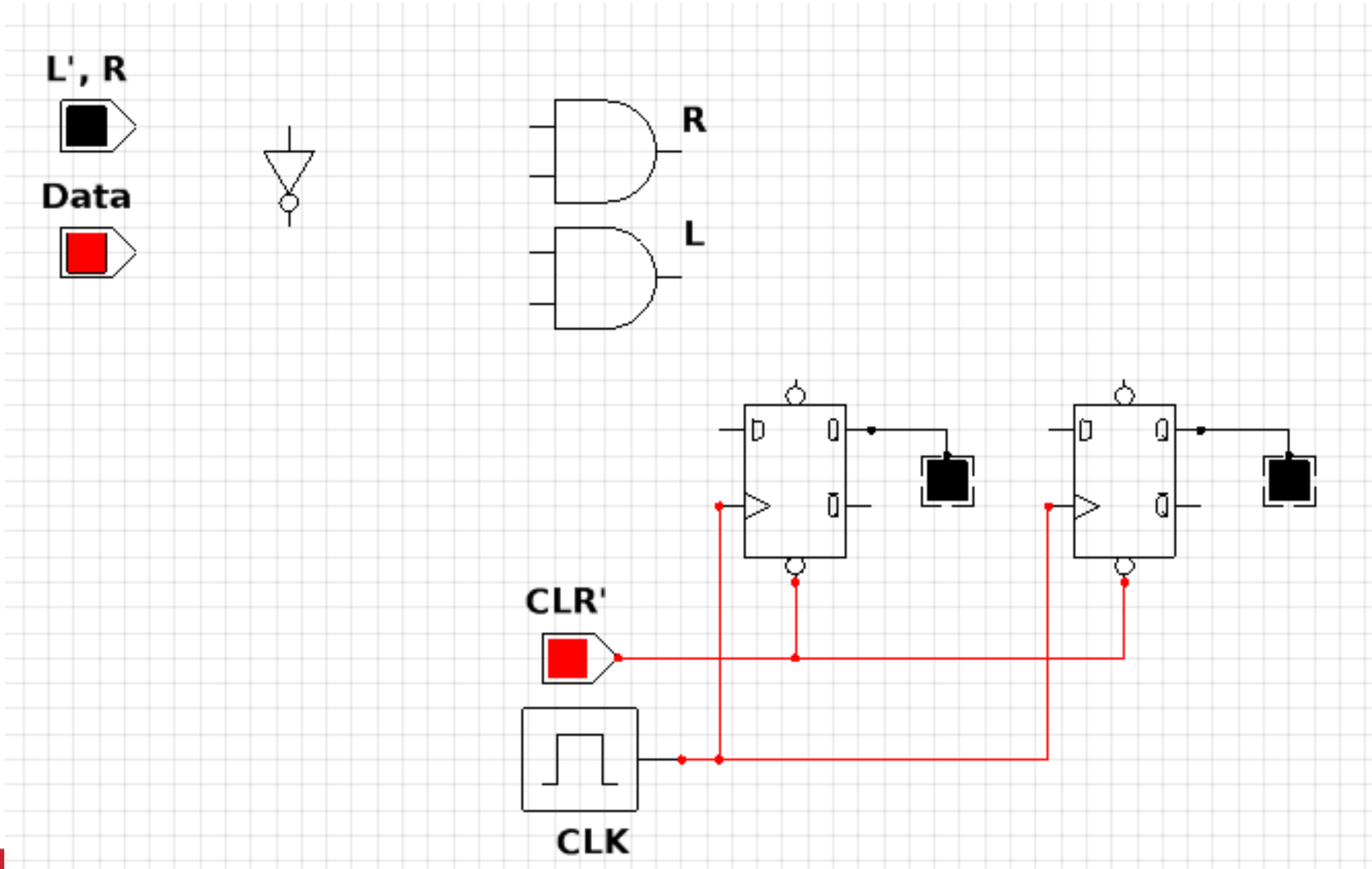


Controlled output gate: Selects A if high, B if low

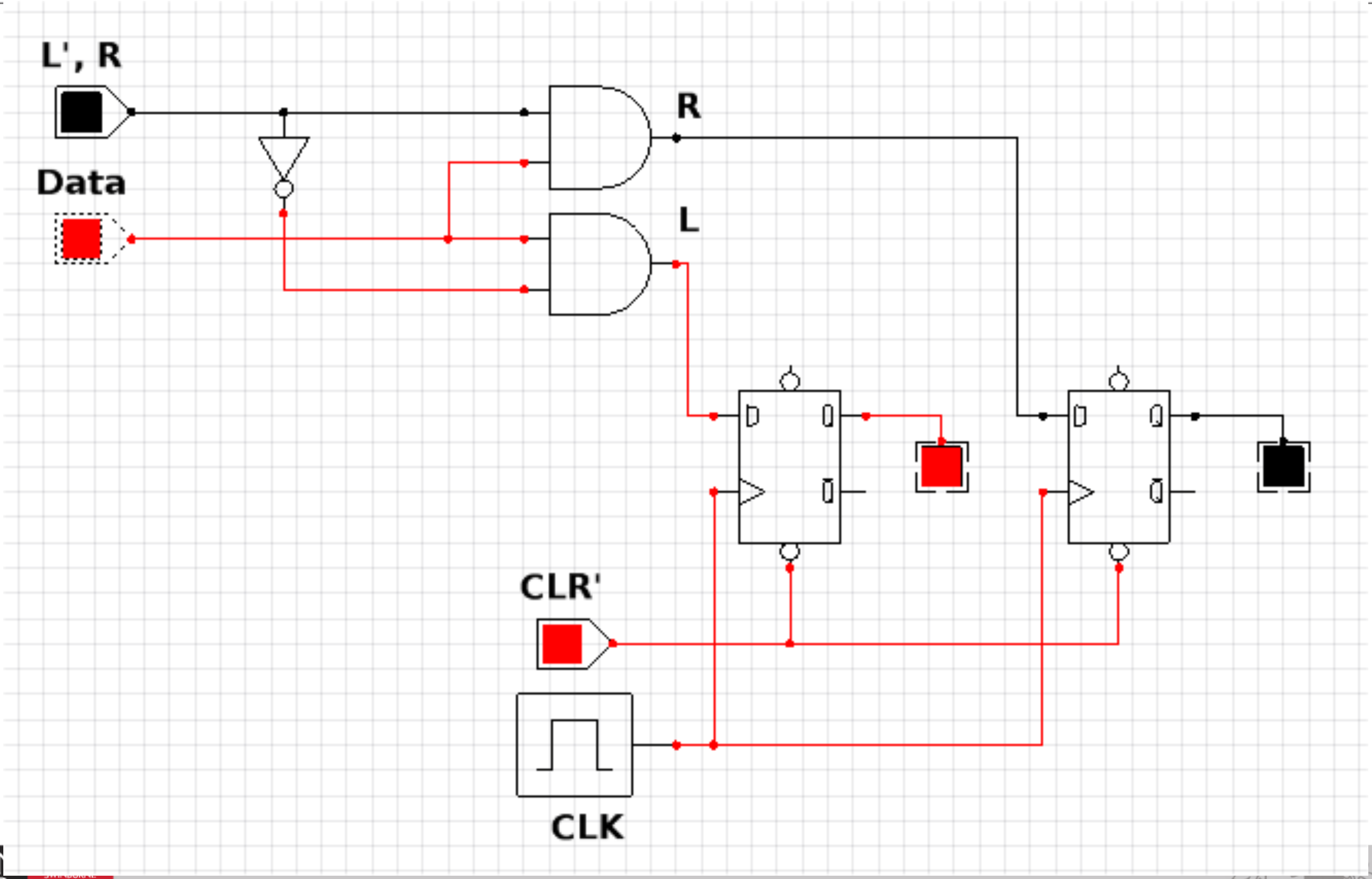
LET'S PUT THIS ALL TOGETHER (JUST 2 FFS TO START WITH)



ADD THE GATES FOR SETTING INPUT

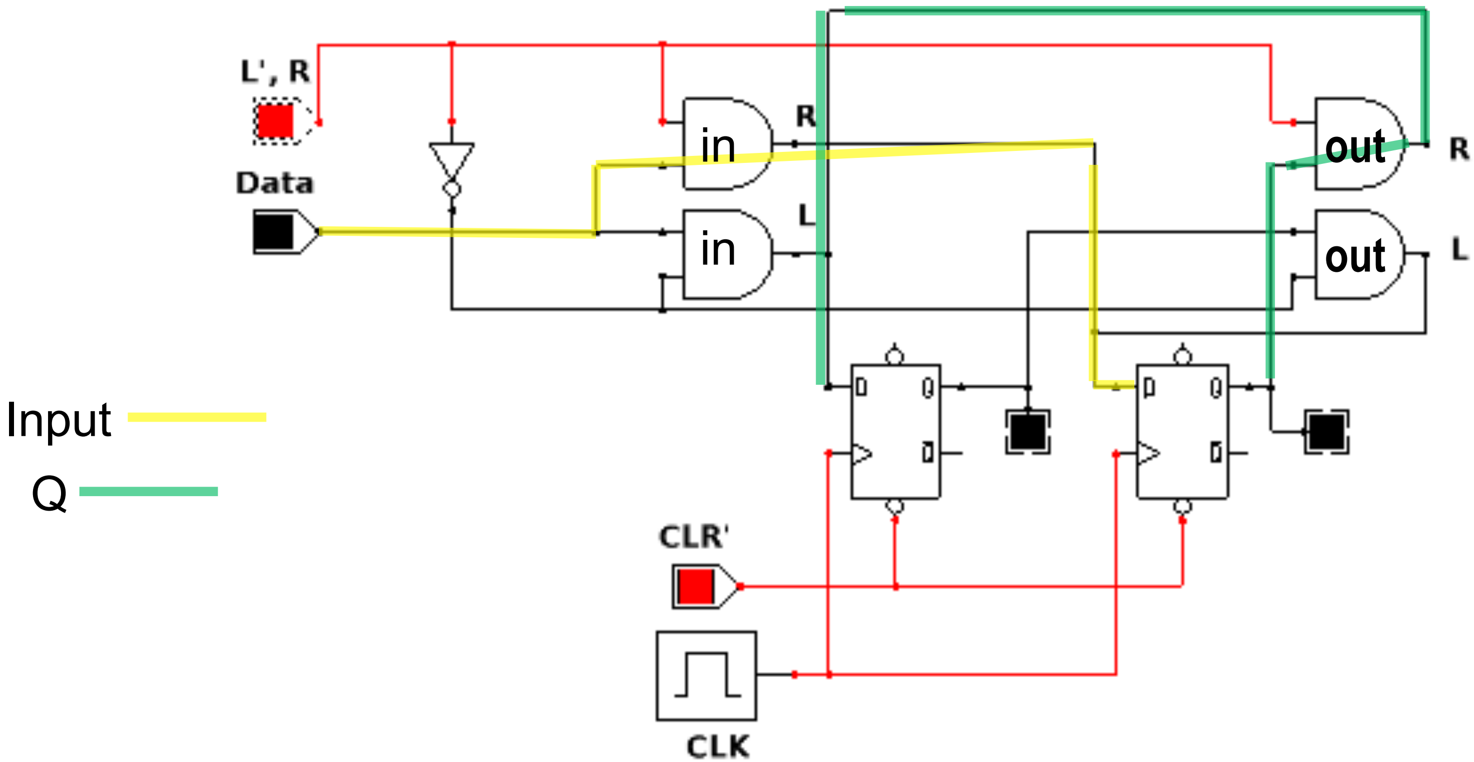


SELECT INPUT

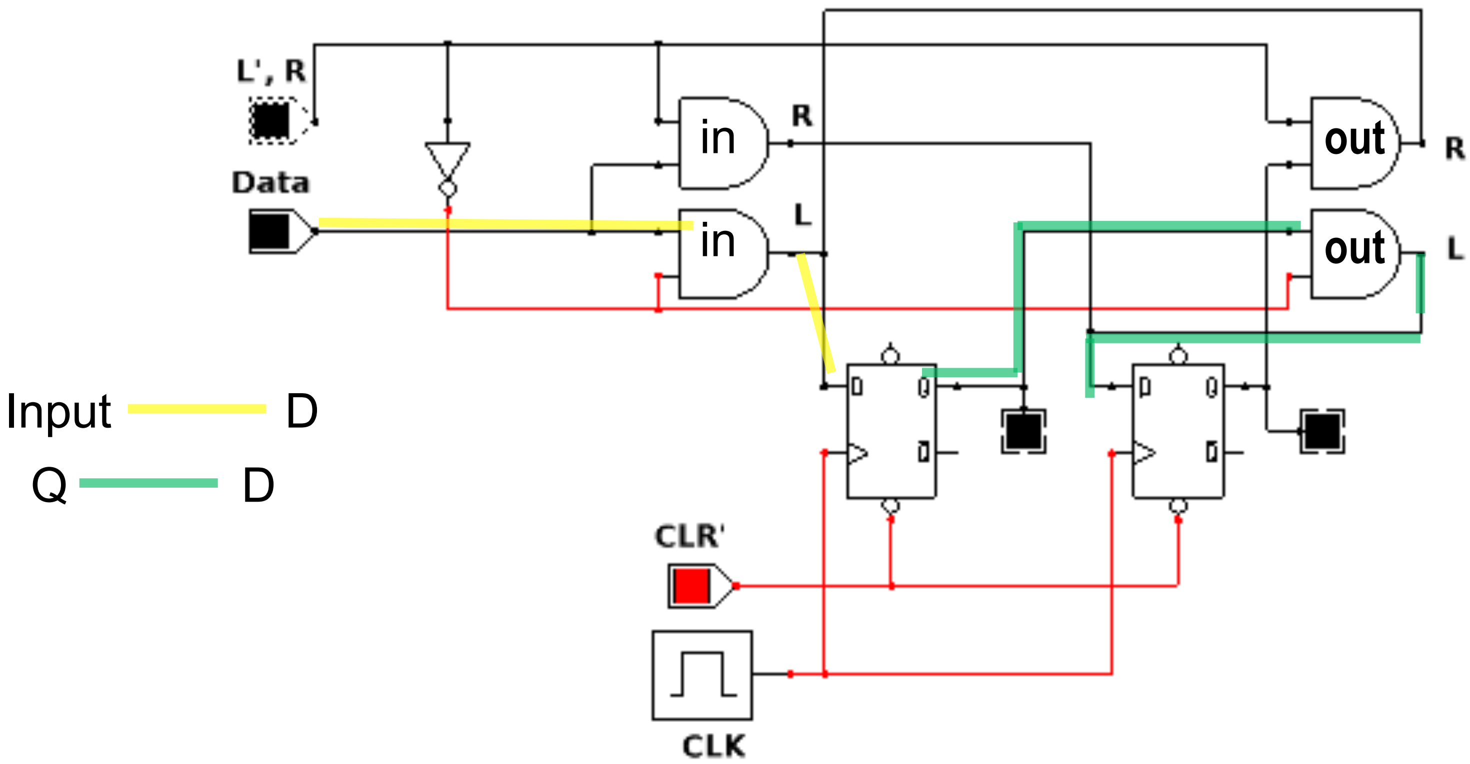


ADD THE OUTPUT DIRECTION SELECTION

- R circuit enabled (red)



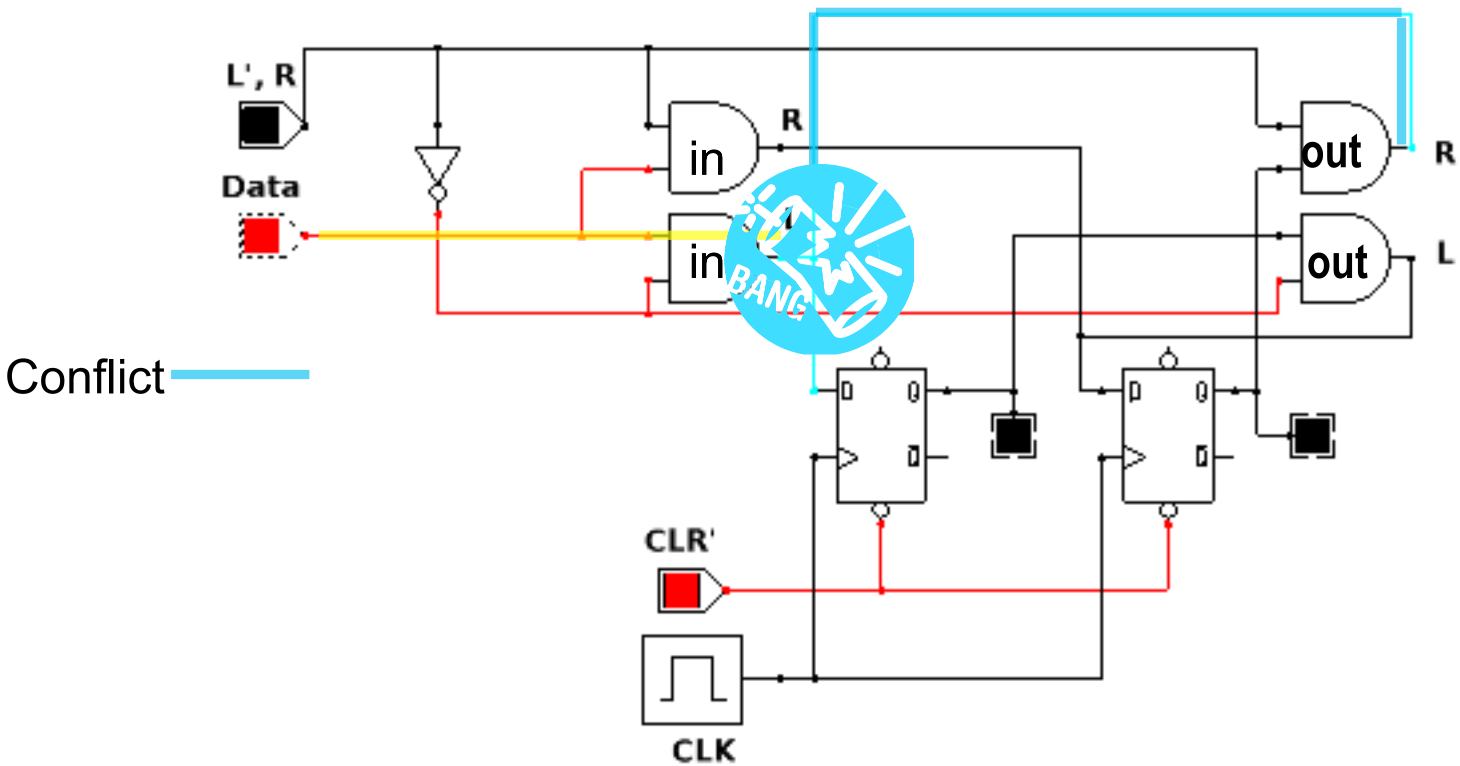
OTHER DIRECTION SELECTED?



WHAT COULD POSSIBLY GO WRONG?

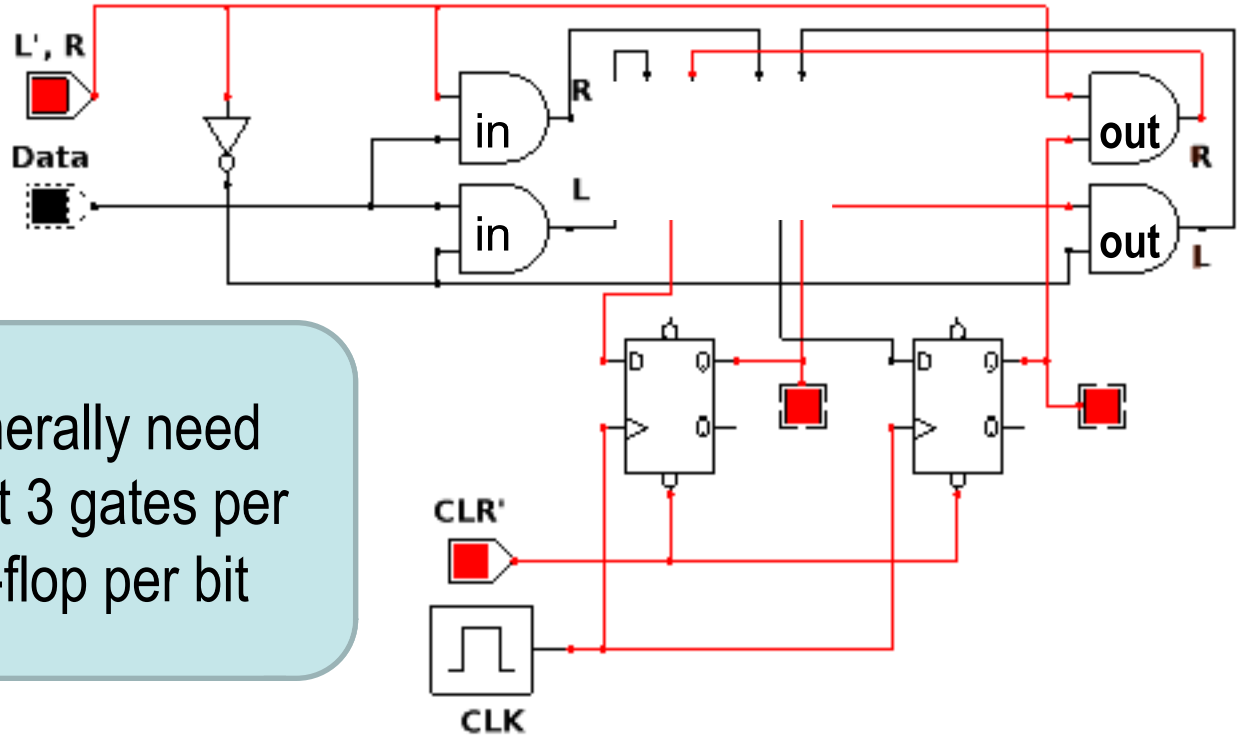
> Can't **short the outputs** of two gates together.

> Have to **Add** with an **OR** gate.



THE FIX

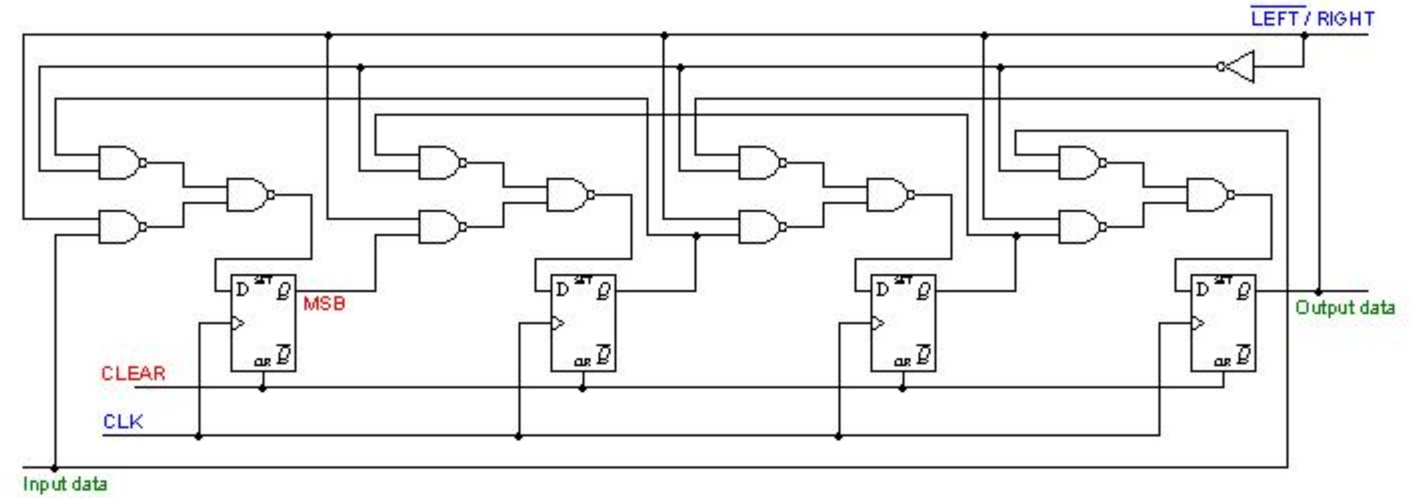
- The OR gates combine the outputs of the controlling AND gates and pass through the signal from whichever one is enabled... to the D inputs on each Flip-Flop.



Generally need about 3 gates per flip-flop per bit

NEED MORE DEPTH?

- This is a 4-deep shift register (http://www.ee.usyd.edu.au/tutorials/digital_tutorial/part2/register06.html).
- As you can see, there are alternative ways of wiring it up, with different logic gates.



TO MAKE A STACK...

- So far we have made a 2-stage bi-directional 1-bit shift register.
- To make a proper stack:
  1. Add depth (flip-flops with associated control logic)
  2. Add width (bits) in parallel (common clock, control signals).
     - identical shift-registers - one for each bit.

SUMMARY

> Hardware stacks formed using banks of shift registers

> Simple input selective circuits allow direction selectability:

  – Programmable gates !

> To make a proper stack:

1. Add depth (flip-flops with associated control logic)
2. Add width (bits) in parallel (common clock, control signals).
   - identical shift-registers - one for each bit.