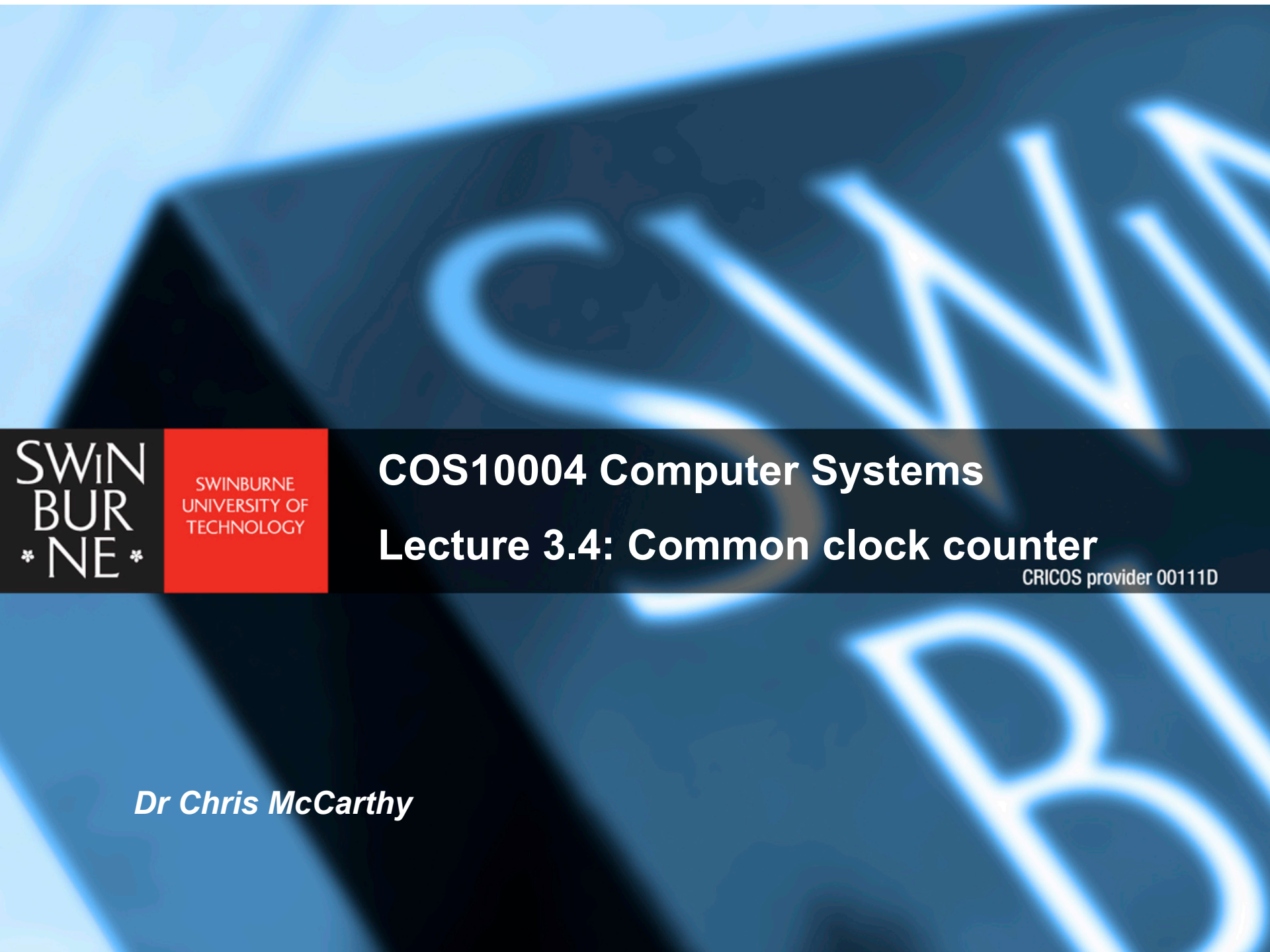


COS10004 Computer Systems

Lecture 3.4: Common clock counter

CRICOS provider 00111D

Dr Chris McCarthy

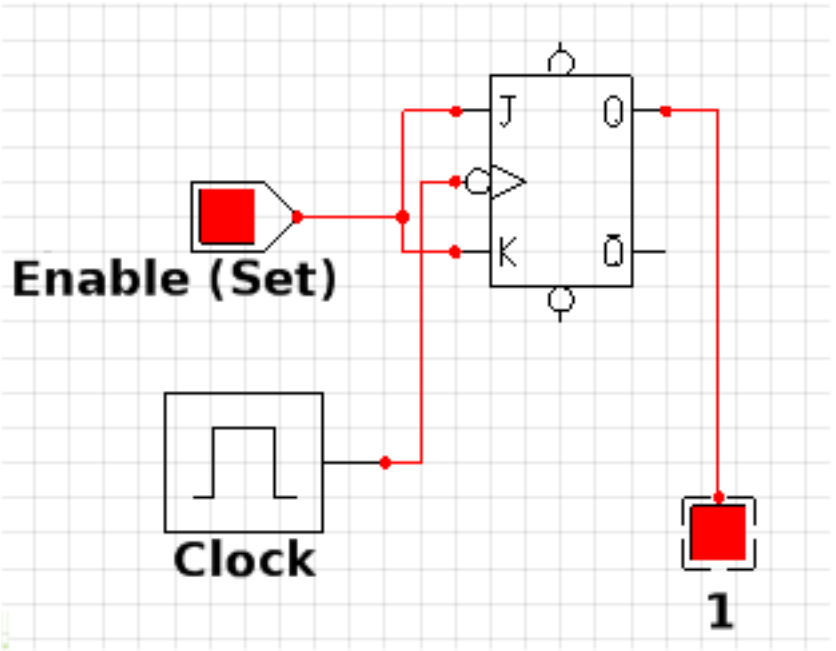
ALTERNATIVE – COUNTER WITH COMMON CLOCK

- We can avoid the illegal state by detecting the last legal state (eg., 5 in a modulo 6 counter), and then set to 0 on next clock pulse.
- This requires a non-cascading counter.
- We need a common clock
- Lets build it up from 1 to 3 bits …..

MOD 6 COUNTER WITH COMMON CLOCK

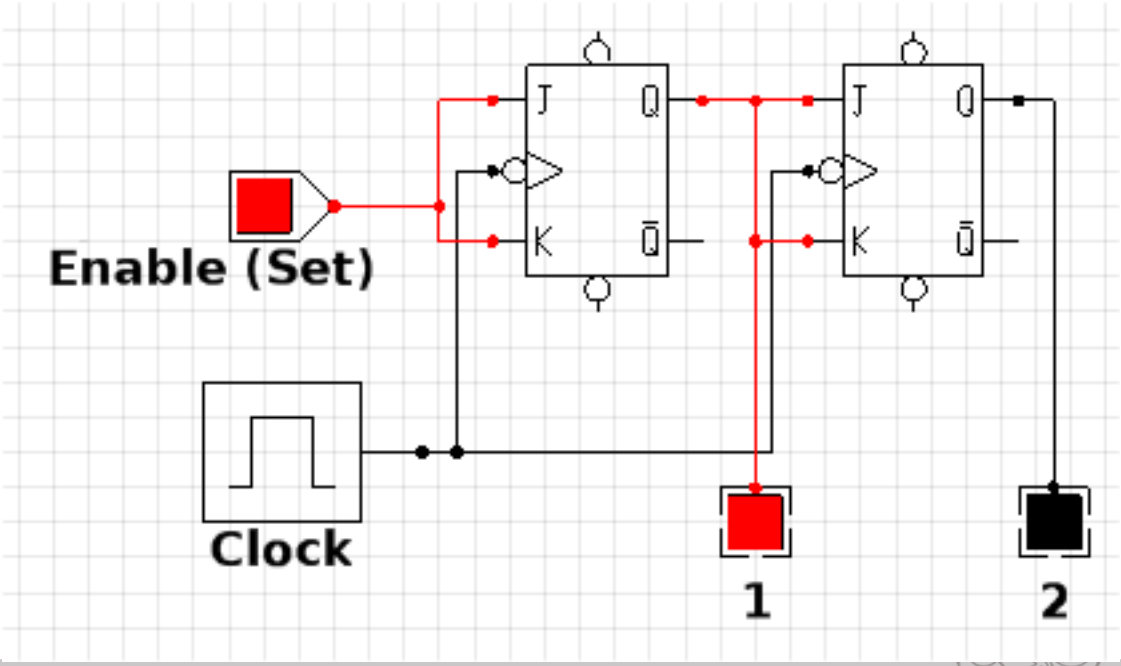
- First a 1 bit counter
- 1-bit (counts 0...1...0...1...)
- Set J and K to make it toggle

J	K	Q_{N+1}	
0	0	Q_N	(No Change)
0	1	0	(Reset)
1	0	1	(Set)
1	1	$\overline{Q}_N$	(Toggle)



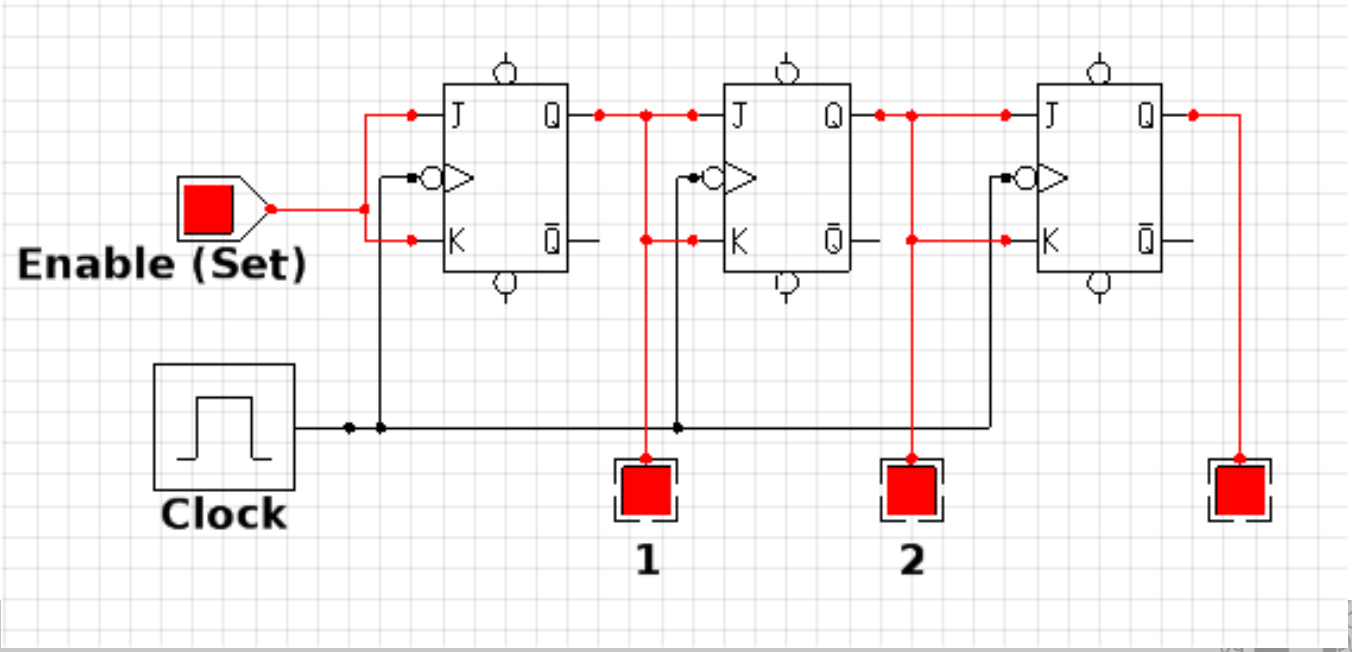
2-BIT COUNTER

- 2-bit (counts 00...01...10...11...00...)
- Connect Q1 to J2 and K2
- Now Q2 only changes state if Q1 is set (halves the frequency).
- but there is no Co (when state = 11)
  - Because we have a **common** clock.



3-BIT COUNTER

- 3-bit? Add another Flip-flop?
- Doesn't work. Q3 is not just supposed to flash at half frequency of Q2
- Think about the truth table.



COUNTING WITH 3 BITS (LITTLE-ENDIAN)

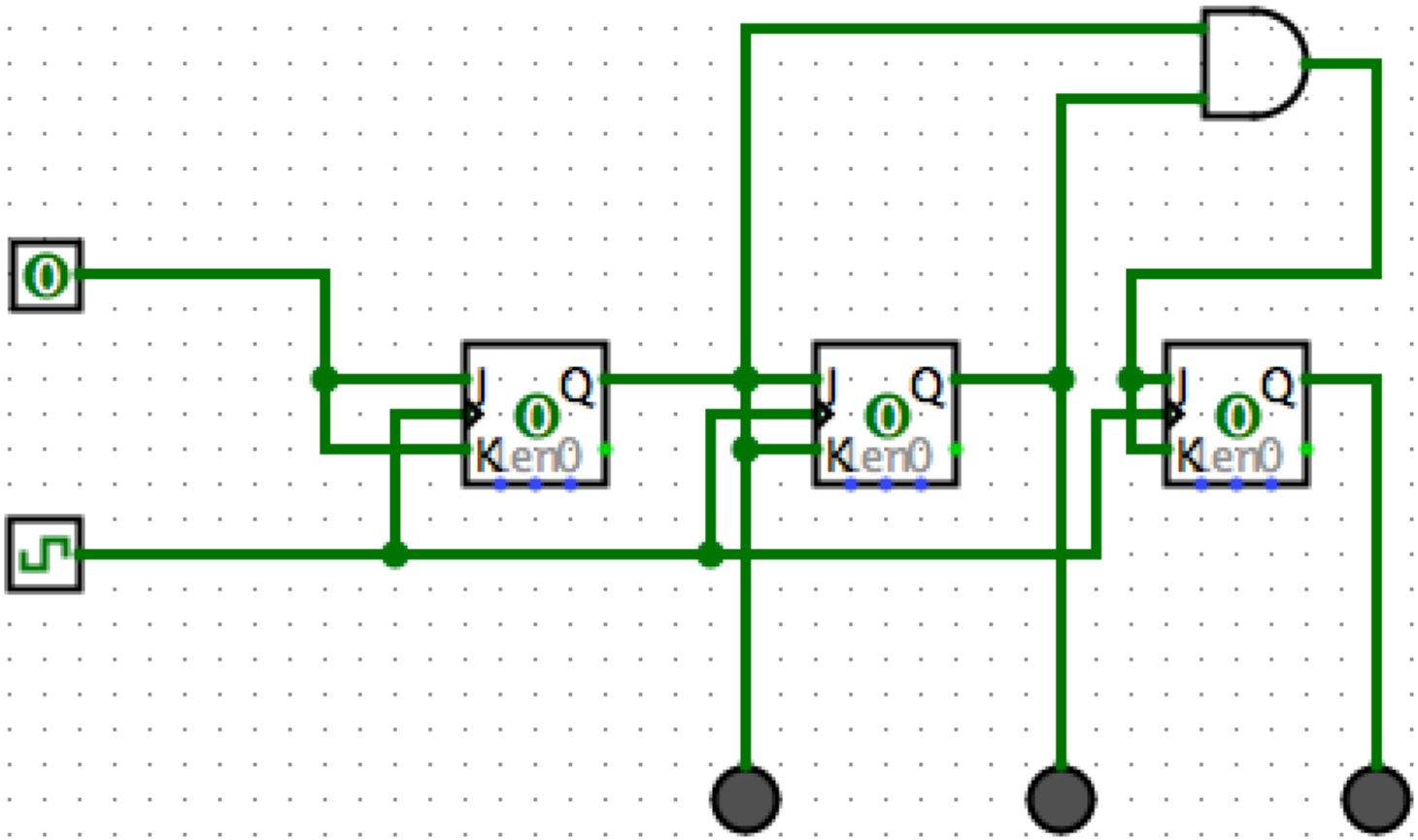
The issue is that the gates are all using the same clock, so Q3 will toggle at 3, and then the counter will reset to 0.
Try it, You will get the number sequence: 0, 1, 2, 7, 0...

Decimal	Q1	Q2	Q3 (expected)	Q3 (actual)
0	0	0	0	0
1	1	0	0	0
2	0	1	0	0
3	1	1	0	**1**
4	0	0	1	**0**
5	1	0	1	**0**
6	0	1	1	**0**
7	1	1	1	**1**

THE FIX... 3-BIT COUNTER

- Add an AND gate to only toggle Q3 when both Q1 and Q2 are set. That's our Carry Out (Co).

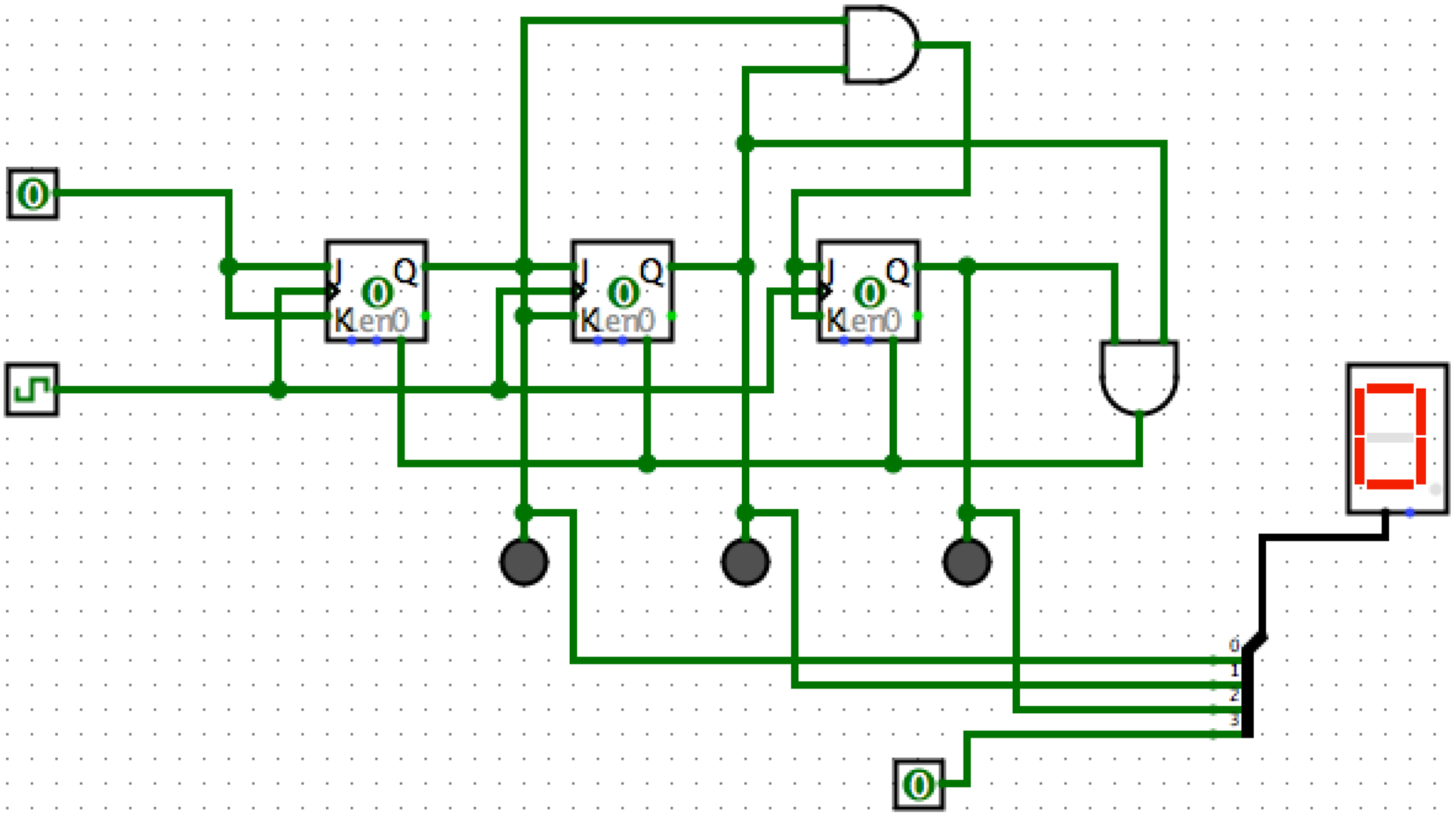
3 bit Common Clock - Little Endian -



NOW TO RESET WHEN IT GETS TO 6 (ILLEGAL STATE)...

- When the output is 110 (Q1=0, Q2=1, Q3=1), use a 2-input AND gate to trigger the Reset on all of the Flip-Flops.

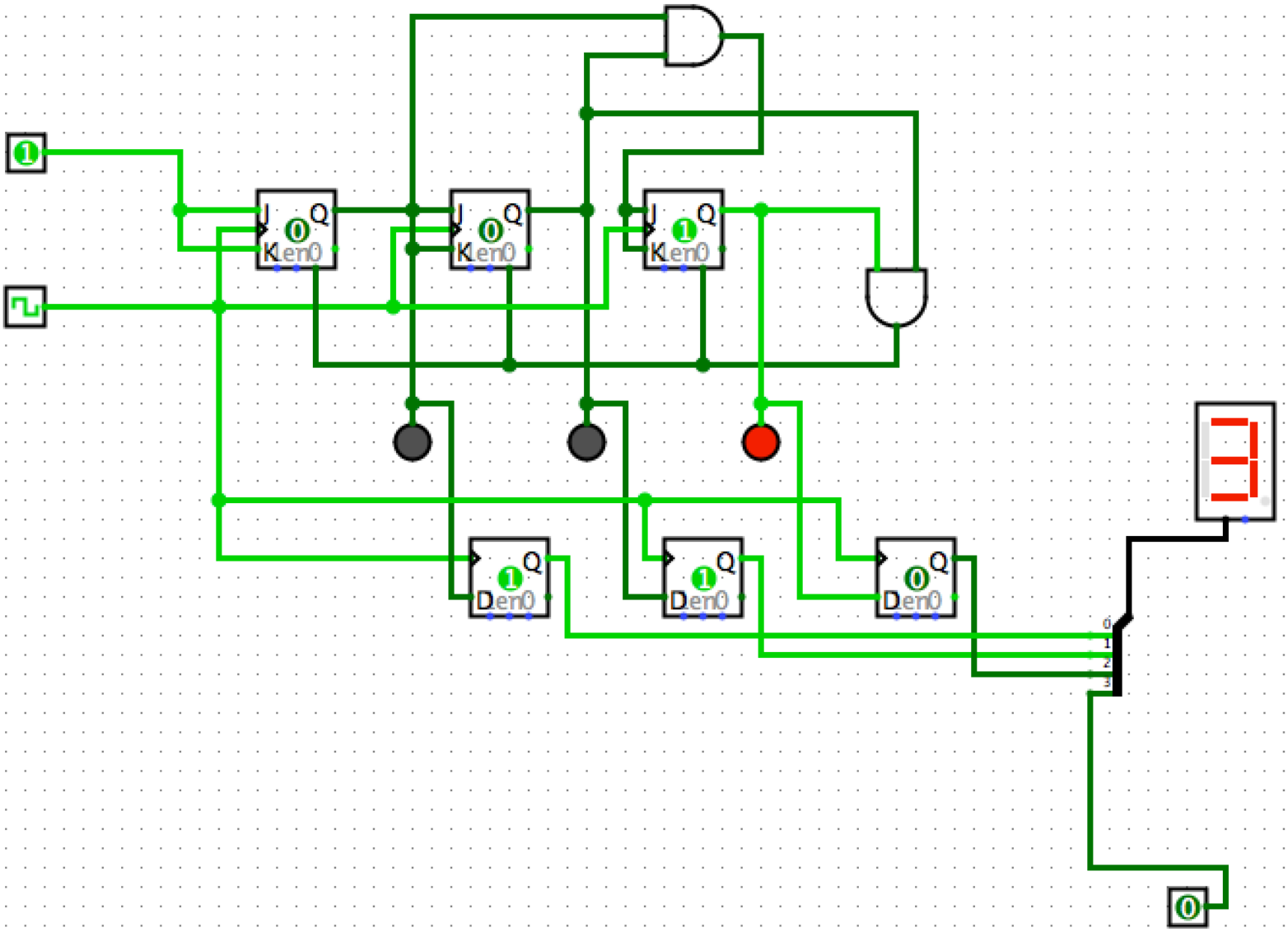
3 bit Common Clock – Little Endian –



ISSUES

- Currently the illegal state (6) still momentarily occurs
- Solution:
  - add D-Flip-Flops as a buffer before displaying
- Adds a delay of 1 clock pulse

3 bit Common Clock – Little Endian – No illegal state



SUMMARY

- Common clock counter synchronises FF state transitions
  - One clock to rule them all!
- Requires explicit logic gates check FF transitions
  - Not quite as elegant as ripple counters!
- D FFs can buffer output for a clock pulse:
  - Allow circuit to stabilise
  - illegal state cannot “escape”

`

push

RESOURCES

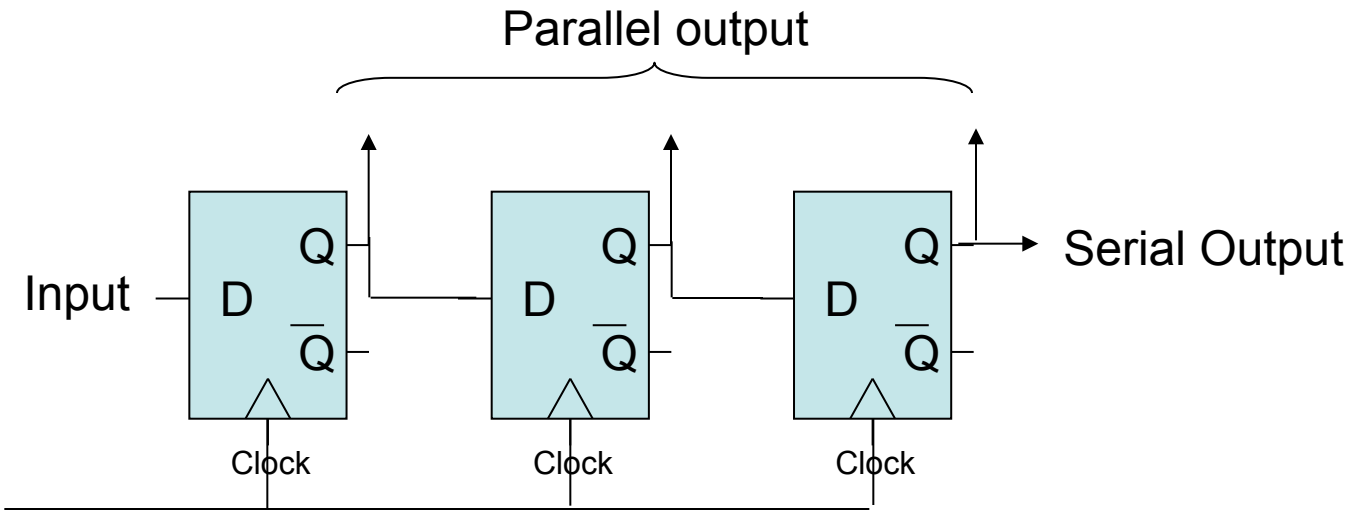
- Video tutorials:
  - A quick reminder that I am progressively making video tutorials for *some* key topics.
  - Not all will be covered, but I try to give practical and quick walk throughs of circuit designs (and later, ASM code)
  - They are linked from the lab sheets.
- Discussion board

pop

SHIFT REGISTERS

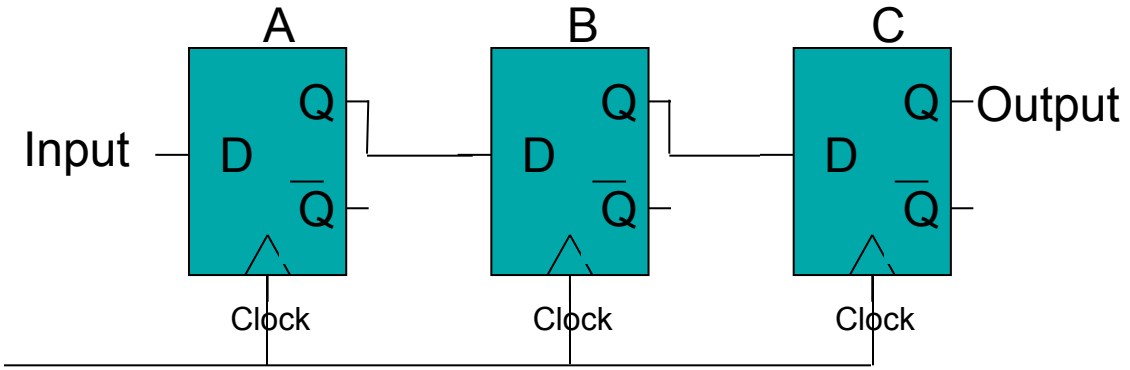
A shift register takes input from one end, and at each clock change this value is moved to the next D-Flip-Flop.

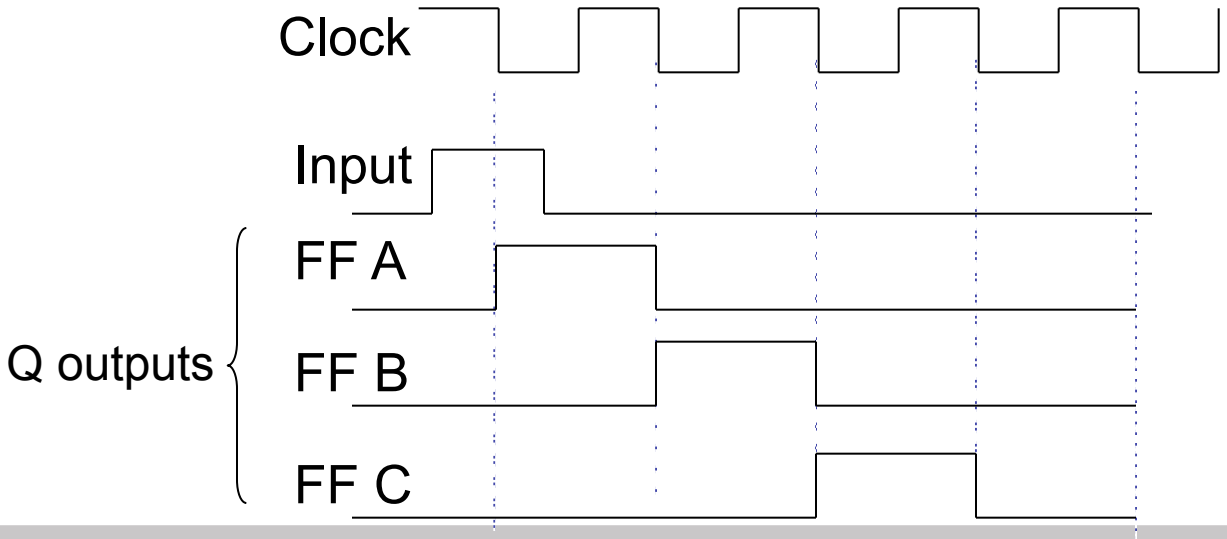
This is used in serial data transfer when a byte (say) of data sent on a cable one bit after another can be collected in a series of D Flip-Flops to rebuild the whole data byte. This is called *serial-to-parallel* conversion.



SHIFT REGISTERS

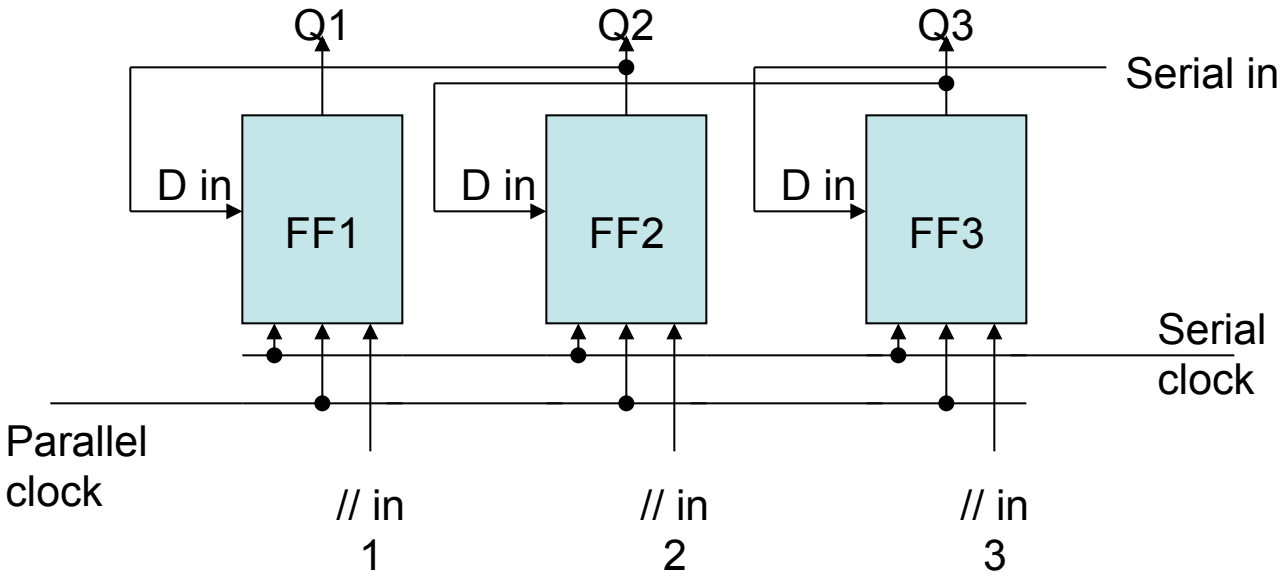
Here is how a *high* travels through a 3 bit shift register. For this example we assume that each of the shift register bits is cleared at the start.





SHIFT LEFT SHIFT REGISTERS

- The shift registers shown so far shift data to the right. A simple rewiring gives a shift register that can move data left. Of course these may also have the ability to parallel load.

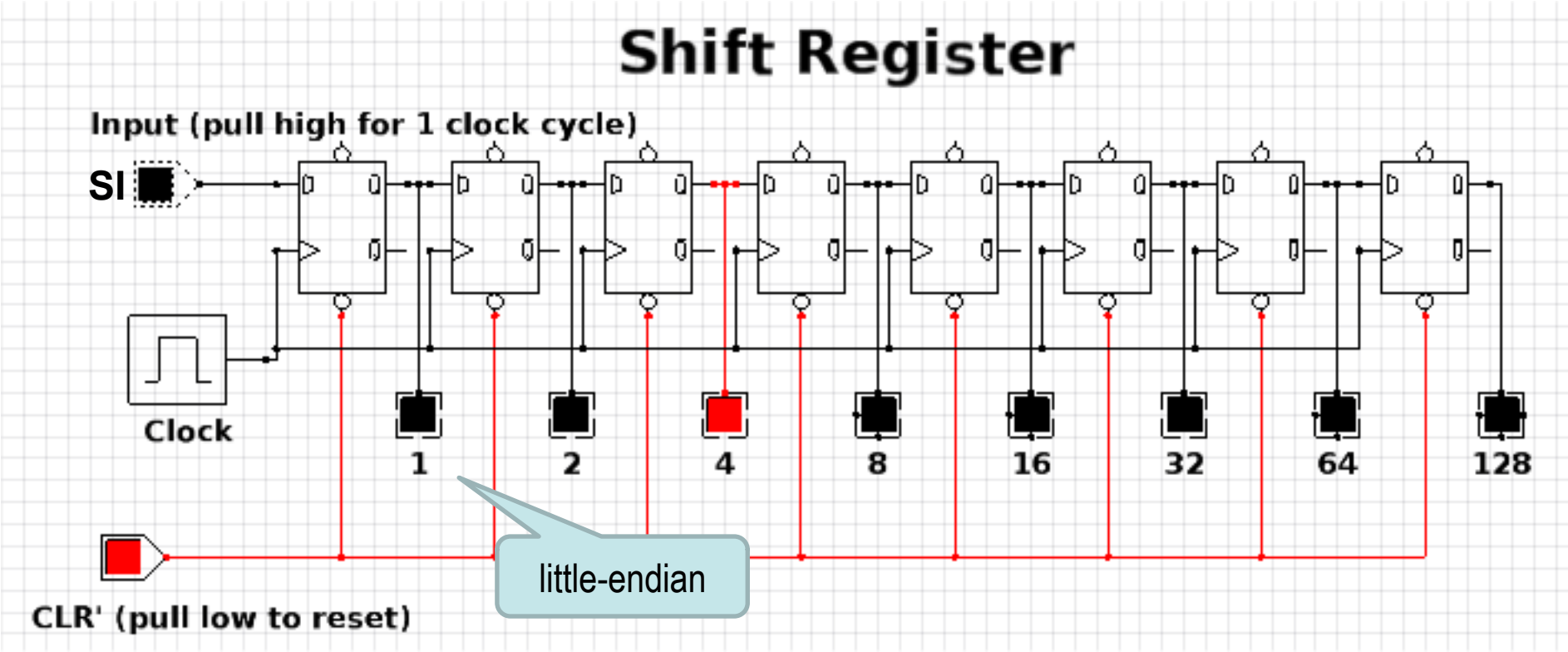


LETS BUILD ONE !

WHAT DOES A SHIFT REGISTER DO?

- Moves a state (number such as 0, 1) from a low order bit to a higher order bit.
- Multiplies a (binary) number by two.
  - Number of "shifts" depends on number of clock cycles.
- Can use a counter to enable/disable clock, thereby programming the amount of shift.
  - Can shift in the other direction - divide numbers by 2.
  - Can have two clocks - one for left-shift, one for right-shift, or use gates to determine the shift direction.
- Can be used as 1-bit of a stack (push/pop).

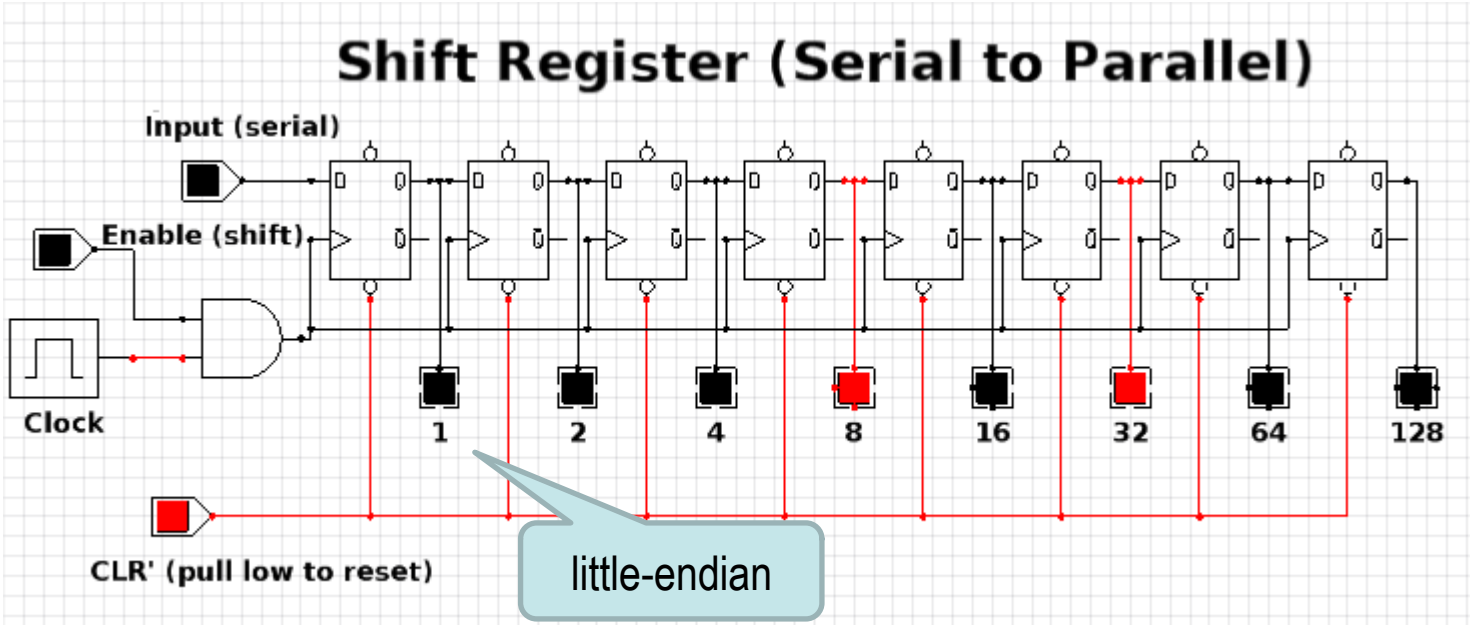
A simple Shift Register from D Flip-Flops



- Same circuit as a decimal counter, but only input 1 pulse instead of holding Input (Data) high.
  Only does serial to parallel.
- To shift *n* bits, input *n* clock cycles.

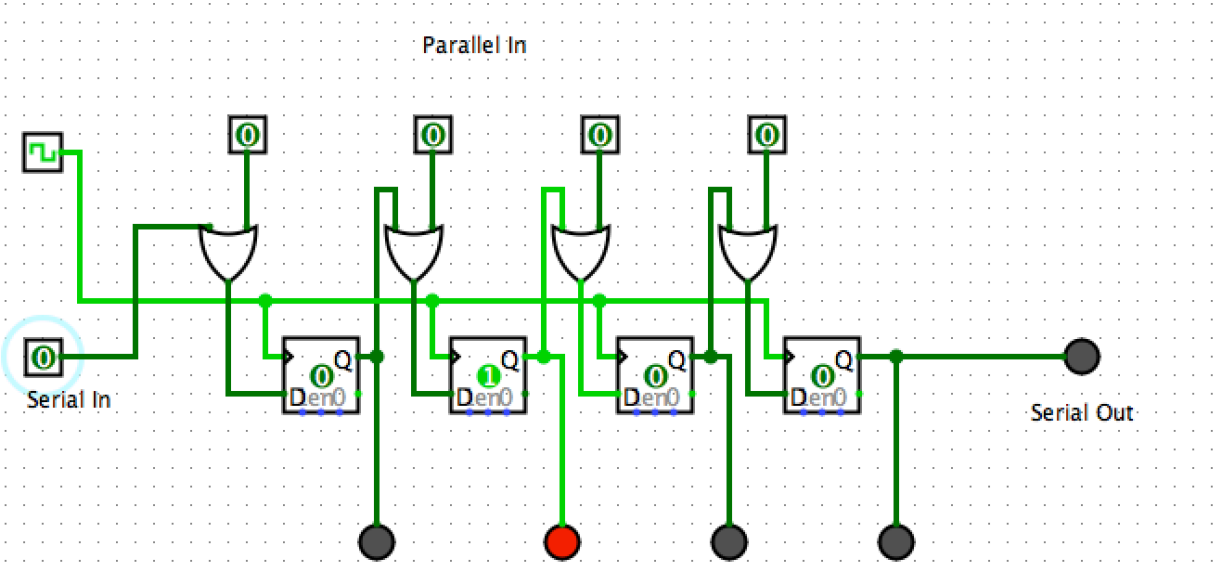
LEFT-SHIFT (LOW BIT TO HIGH BIT) SERIAL TO PARALLEL

- Can modulate the Input (serial input) to load states into a register (serial to parallel conversion)
- Once the "conversion" is complete, disable the clock and store in a latch/register.

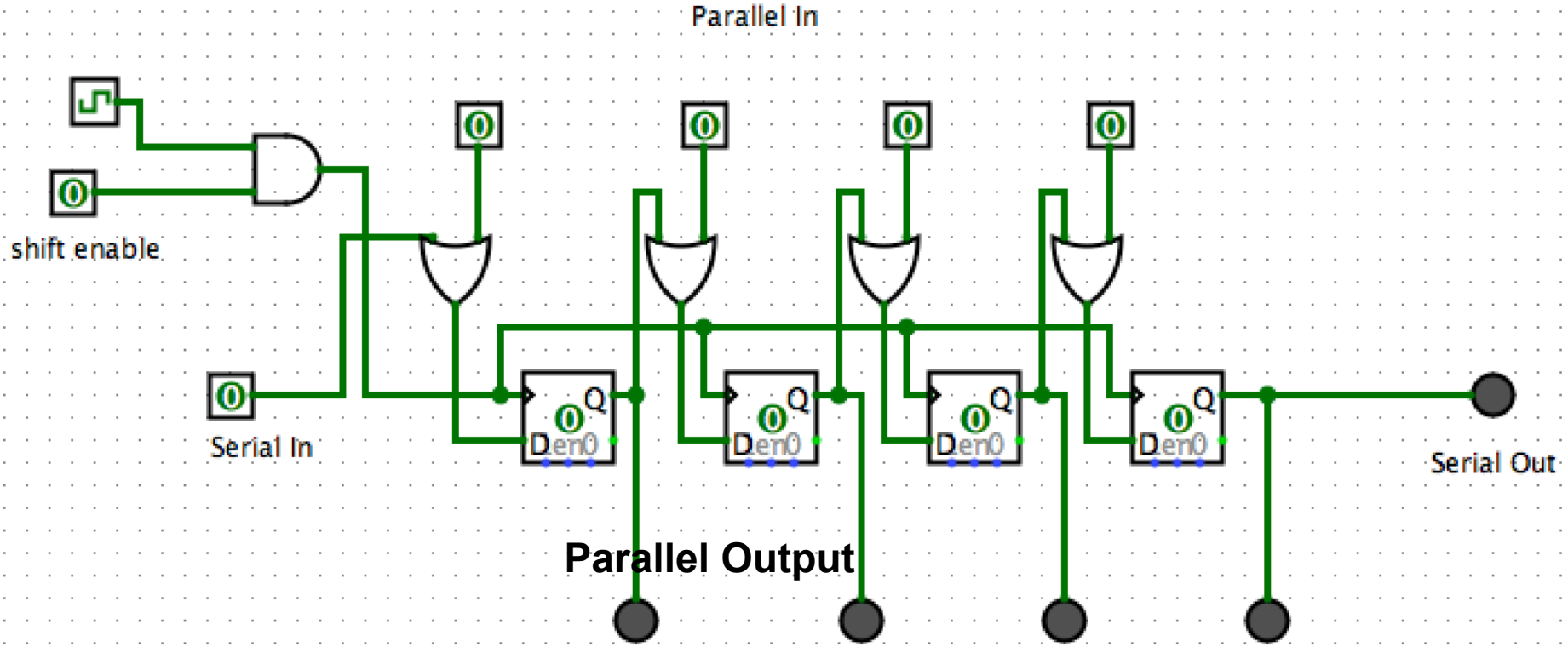


PARALLEL LOAD SHIFT REGISTERS

- Some shift registers allow all flip-flops to load at once,



RIGHT-SHIFT SERIAL OR PARALLEL INPUT: 4-BIT

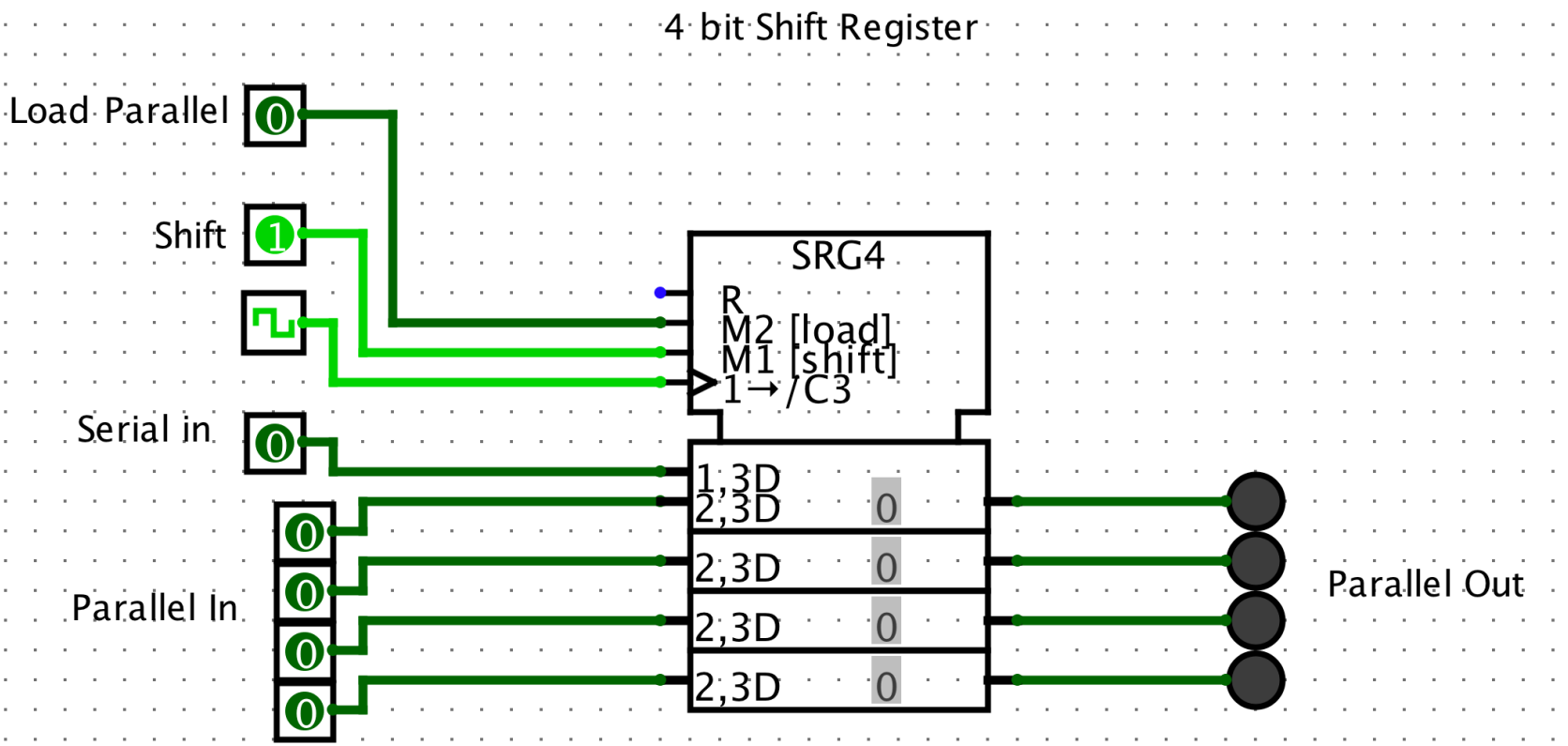


- Flip-flops are connected (output to input) with a common clock to cascade input from high bytes to low bytes.
- Each flip-flop has a parallel bit OR-ed to the input to allow PReset.

What about Bi-directional Shifting (Selectable) ?

- This takes some thinking!
- You need a pin to select which direction
- You need to allow inputs to any D Flip Flop to come from either direction
- Try and design a 2 bit selectable direction Shift Register !
  - Serial input only
  - You will probably need some OR gates ,AND gates, and a NOT gate

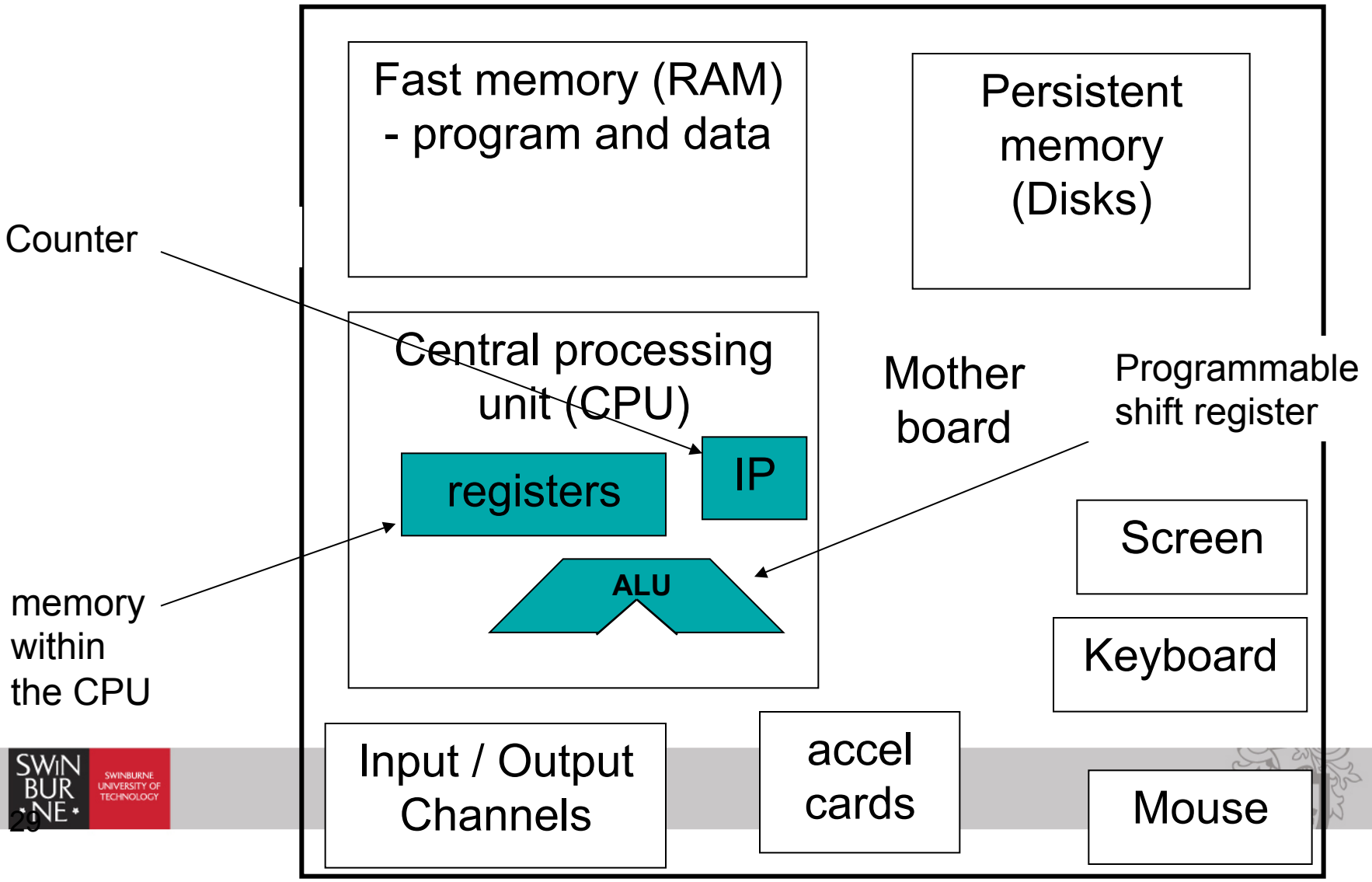
THE LOGISIM SHIFT REGISTER



THINGS TO TRY

- Parallel Load a value and Shift to see it halve.
- Implement both directions and see it double or halve.
- Hold down SI to inject a bit - see it halve or double.

WHERE ARE WE?


Fast memory (RAM)
- program and data
Persistent
memory
(Disks)
Counter
Central processing
unit (CPU)
Mother
board
Programmable
shift register
registers
IP
ALU
Screen
Keyboard
memory
within
the CPU
Input / Output
Channels
accel
cards
Mouse

Things to Remember (Fill this in)

ALU Component	Types of Flip Flops	Inputs	Clock type	Extra Circuit Elements (Gates)
Register (n-bit latch)		Required state	Common clock	
Ripple Counter				none
Decimal Counter	D Flip Flops			
Mod n Counter			Cascading clock	AND gate
Mod 6 Counter			Common clock	
3-bit counter		Enable		AND gate
Shift Register				AND, OR gates to program behaviour

Things to try

- Enable or disable a clock pulse with a switch.
- Convert the little-endian circuits to big endian.
- Convert the big-endian circuits to little endian.
- Wire up counters to count up or count down depending on the wiring (selectable).
- Control the selection with a flip-flop.
- Up-down counters where an up counter triggers an RS (or JK) flip-flop to enable the up wiring, and the down button unsets the RS flip-flop and enables the down wiring.

In the lab...

- Build a register out of D-Flip-Flops
- Build various counters
- Build a shift register