COS10004 Computer Systems

Lecture 3.2: Counters- the ripple counter

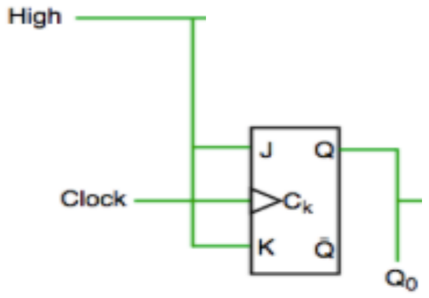
Dr Chris McCarthy

COUNTERS IN DIGITAL LOGIC

- A circuit that stores and increments the number of occurrences of an event:
  - Most commonly clock pulses
- Two types we will consider:
  - Asynchronous (ripple) counter
  - Synchronous (common clock) counter
- J-K Flip Flops central to both

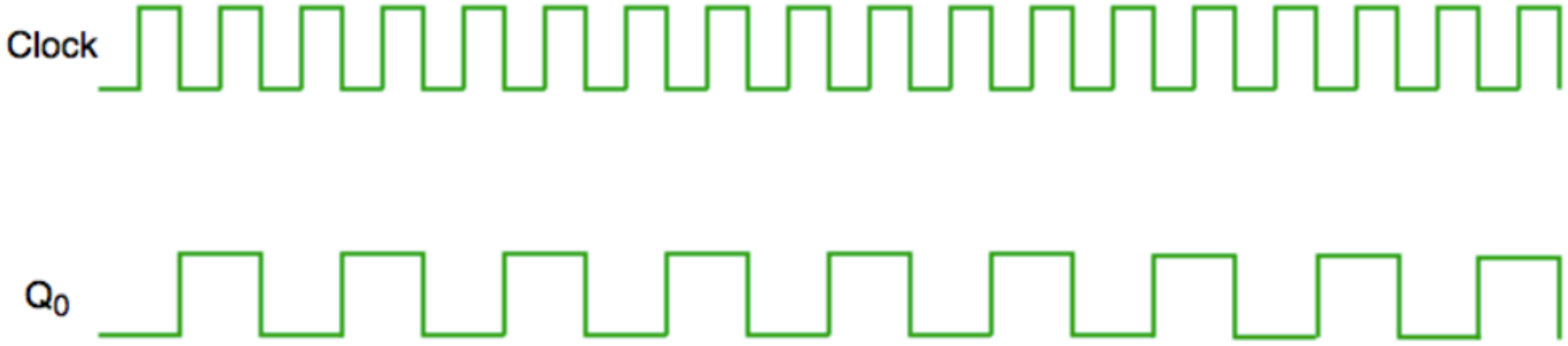
Ripple Counter

- Ripple counters utilise the toggle setting of J-K Flip Flops:
- Consider this circuit
  - What happens to Q_o each clock pulse ?:



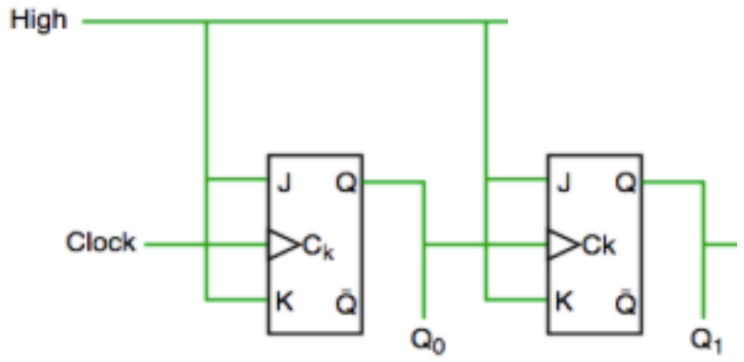
RIPPLE COUNTER

- Ripple counters utilise the toggle setting of J-K Flip Flops:
- Consider this circuit
  - What happens to Q_0 each clock pulse ?:



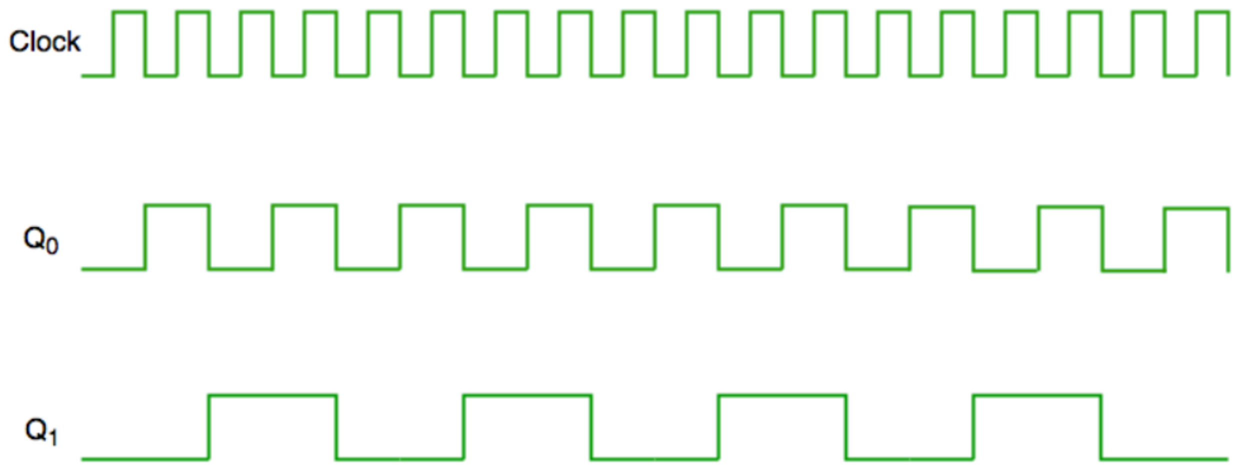
RIPPLE COUNTER

- Ripple counters utilise the toggle setting of J-K Flip Flops:
- Consider two Flip Flops:
  - What happens to Q_0 and Q_1 each clock pulse ?



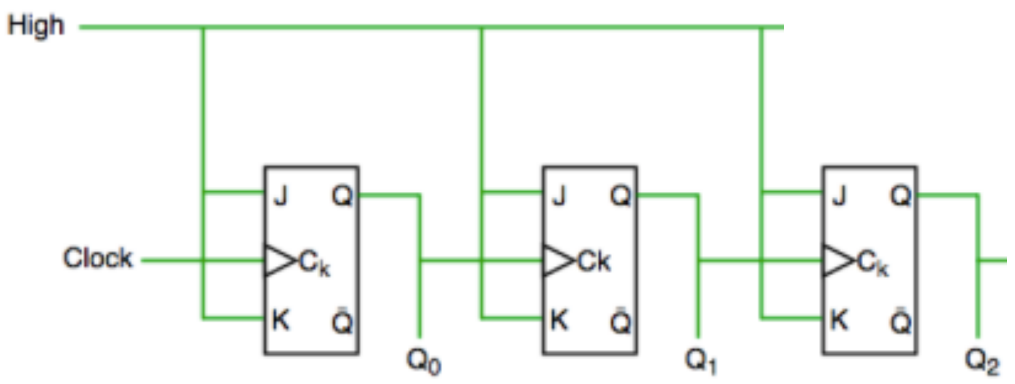
Ripple Counter

- Ripple counters utilise the toggle setting of J-K Flip Flops:
- Consider two Flip Flops :
  - What happens to Q_0 and Q_1 each clock pulse ?



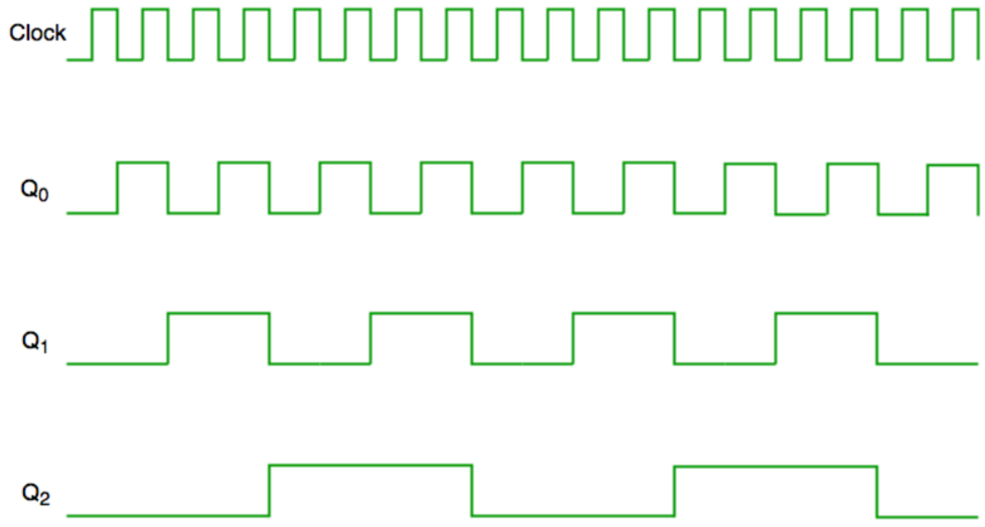
RIPPLE COUNTER

> Ripple counters utilise the toggle setting of J-K Flip Flops:
> What about three FFs ?
  – What happens to Q_0, Q_1, and Q_2 each clock pulse ?



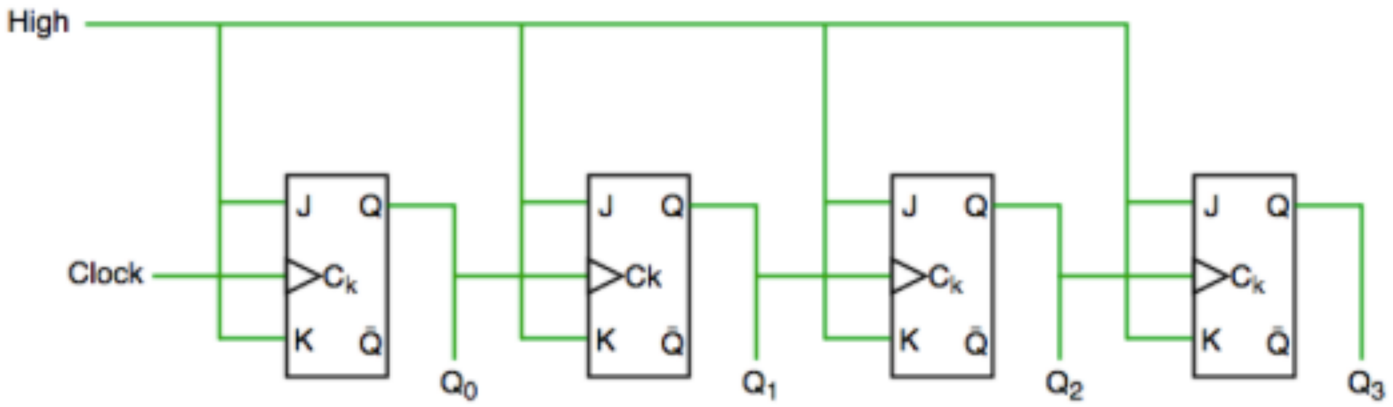
Ripple Counter

- Ripple counters utilise the toggle setting of J-K Flip Flops:
- Consider this circuit:
  - What happens to Q_o , Q_1 and Q_3 each clock pulse ?



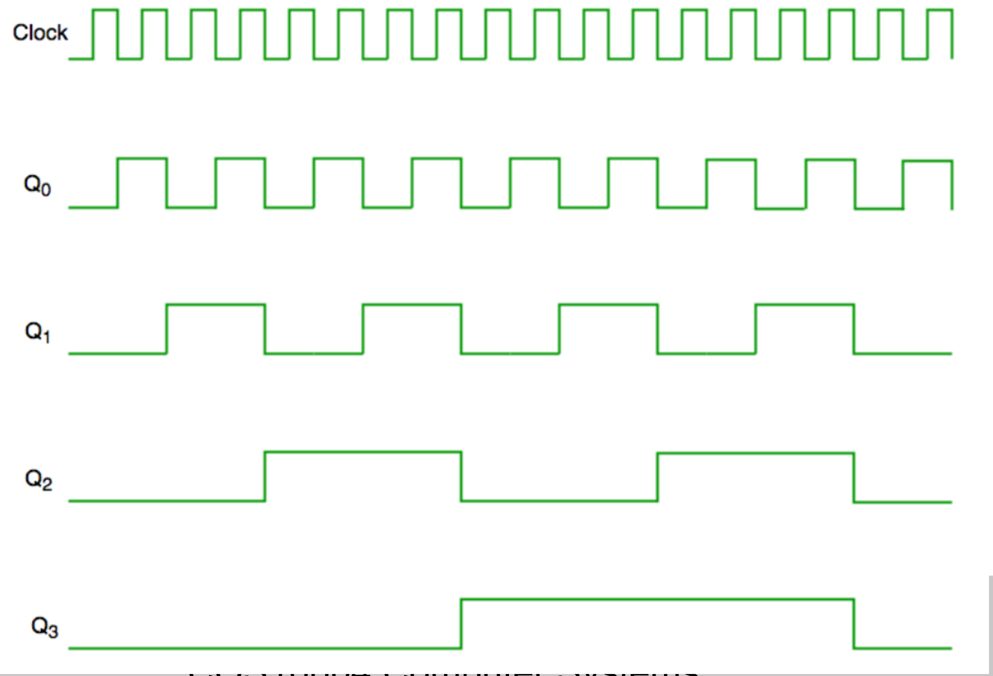
RIPPLE COUNTER

- Ripple counters utilise the toggle setting of J-K Flip Flops:
- And finally, four FFS
  - : Q_0, Q_1, Q_2, Q_3 ?

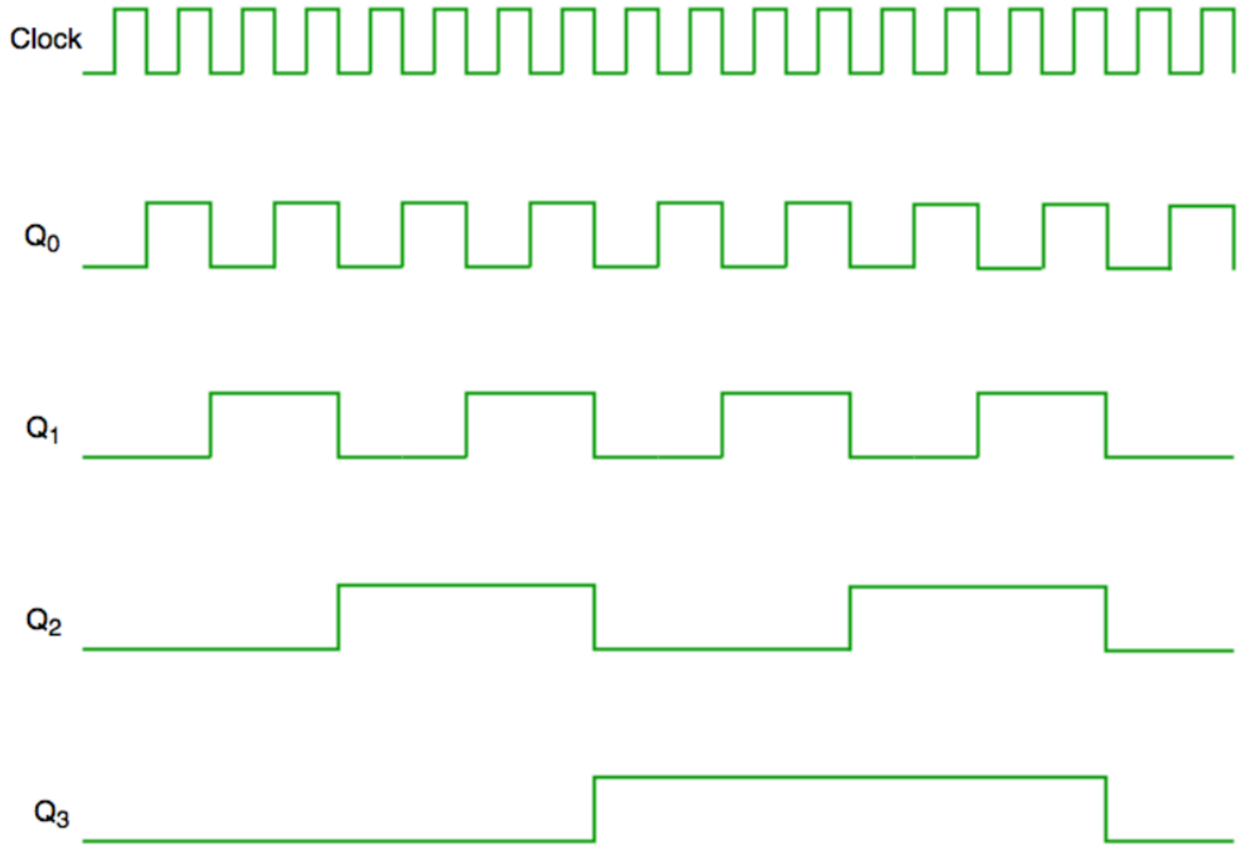


Ripple Counter

- Ripple counters utilise the toggle setting of J-K Flip Flops:
- And finally, four FFS
  - : Q_0, Q_1, Q_2, Q_3 ?



Ripple Counter



One's

Two's

Four's

Eight's

RIPPLE COUNTER

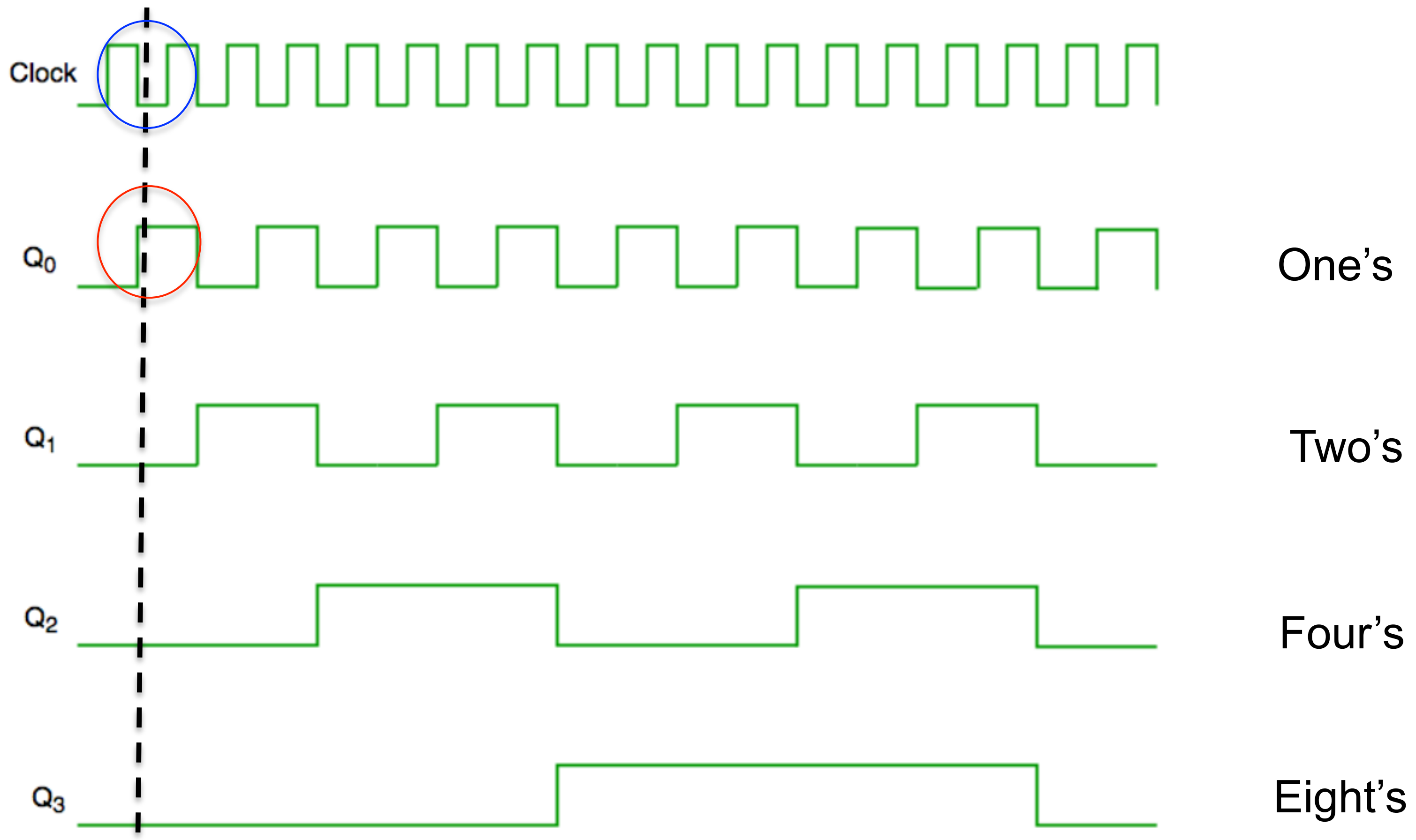
> The ripple effect of FF toggles exactly matches binary counts:

  – each oscillates at half the frequency of the FF before it!

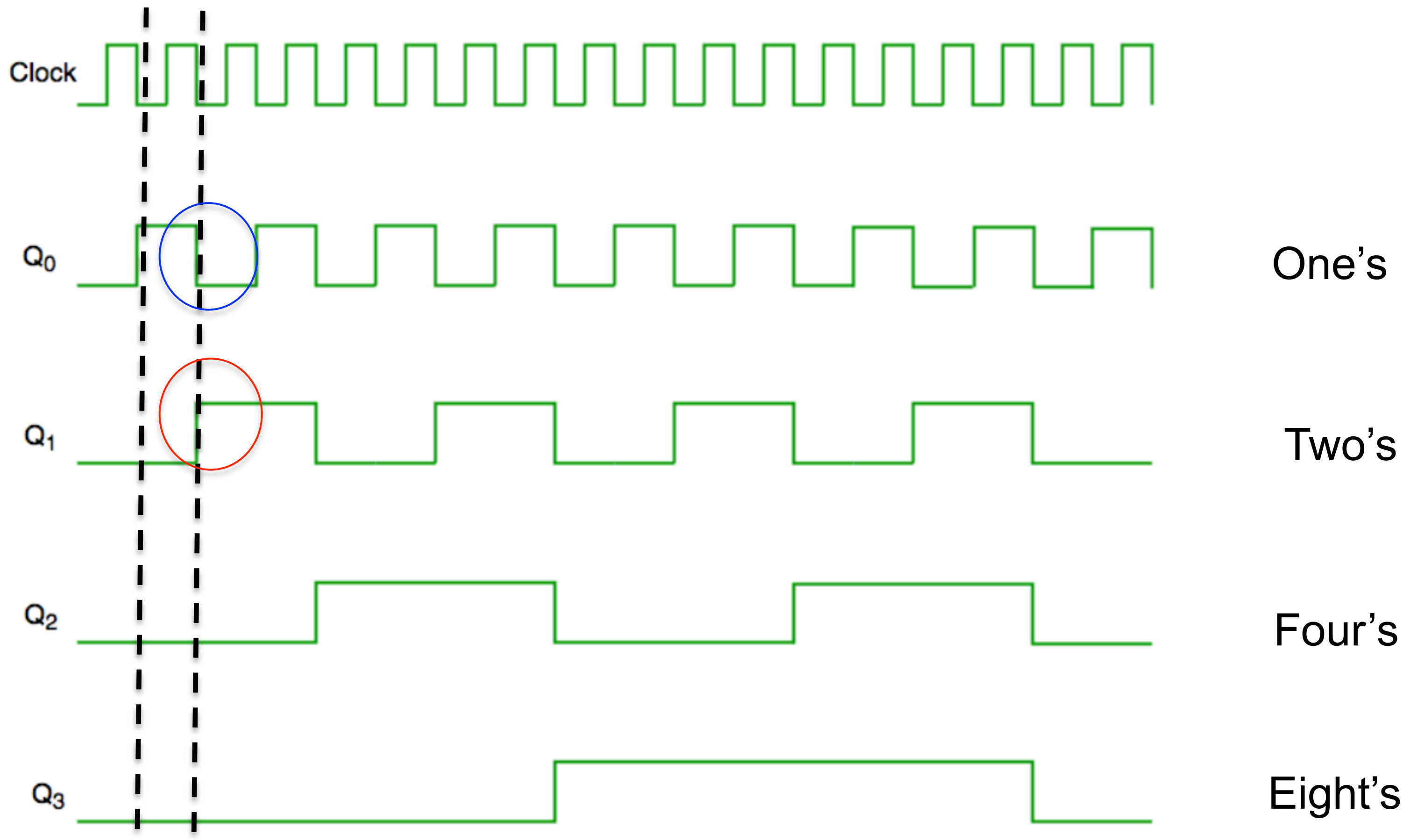
  – The bits stored in the FFs thus represent incrementing binary values

  – BUT - on which edge of the clock pulse ?

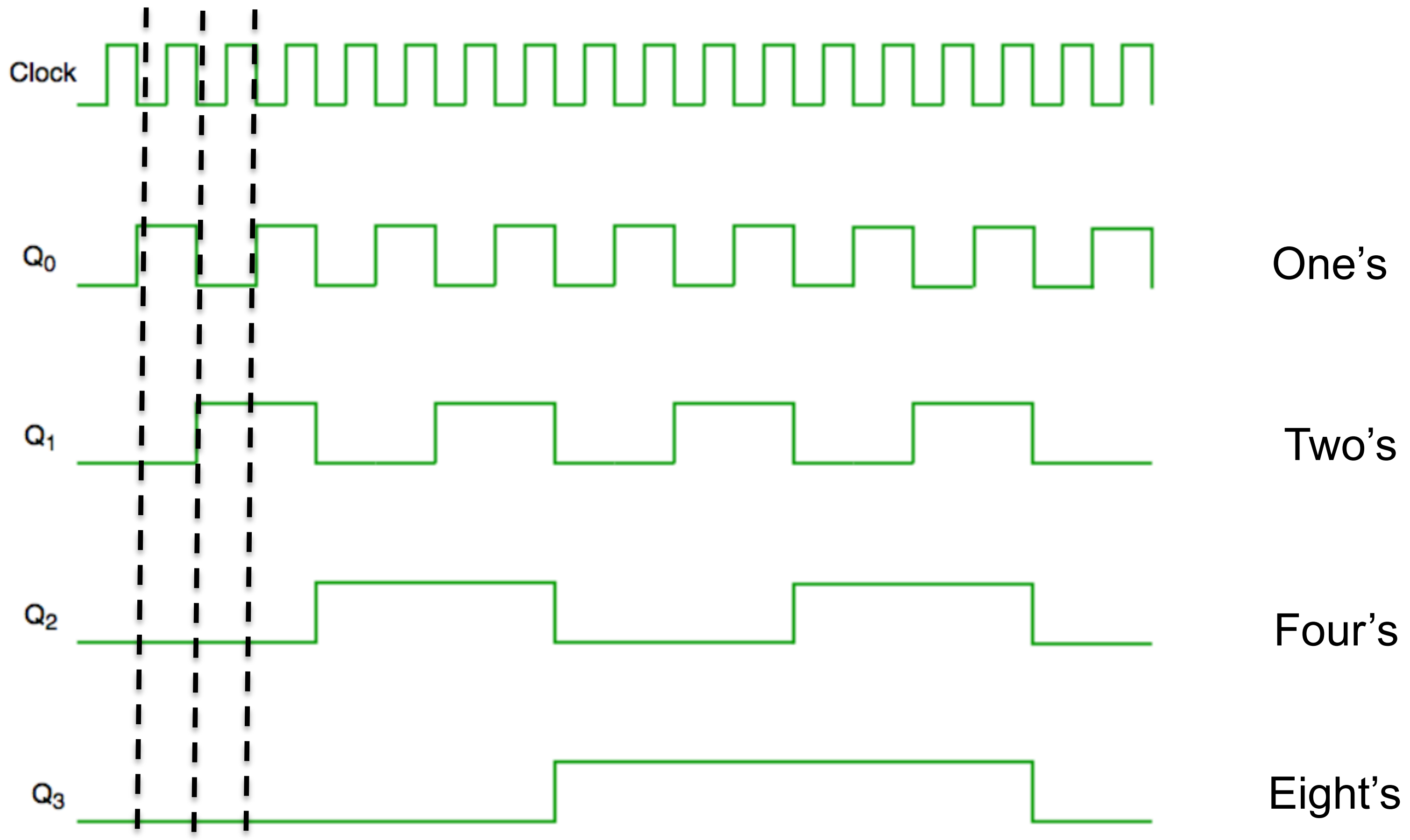
Ripple Counter



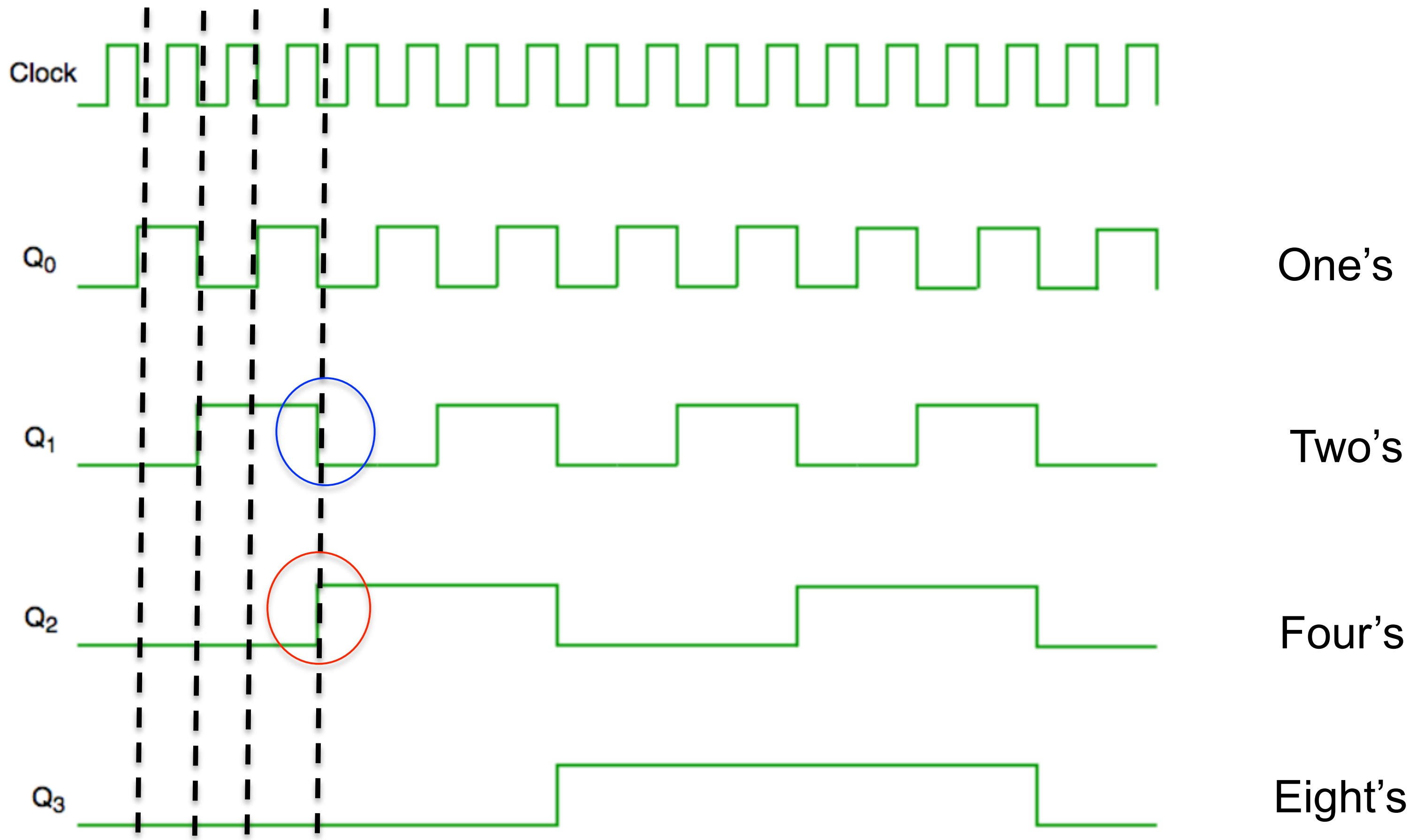
Ripple Counter



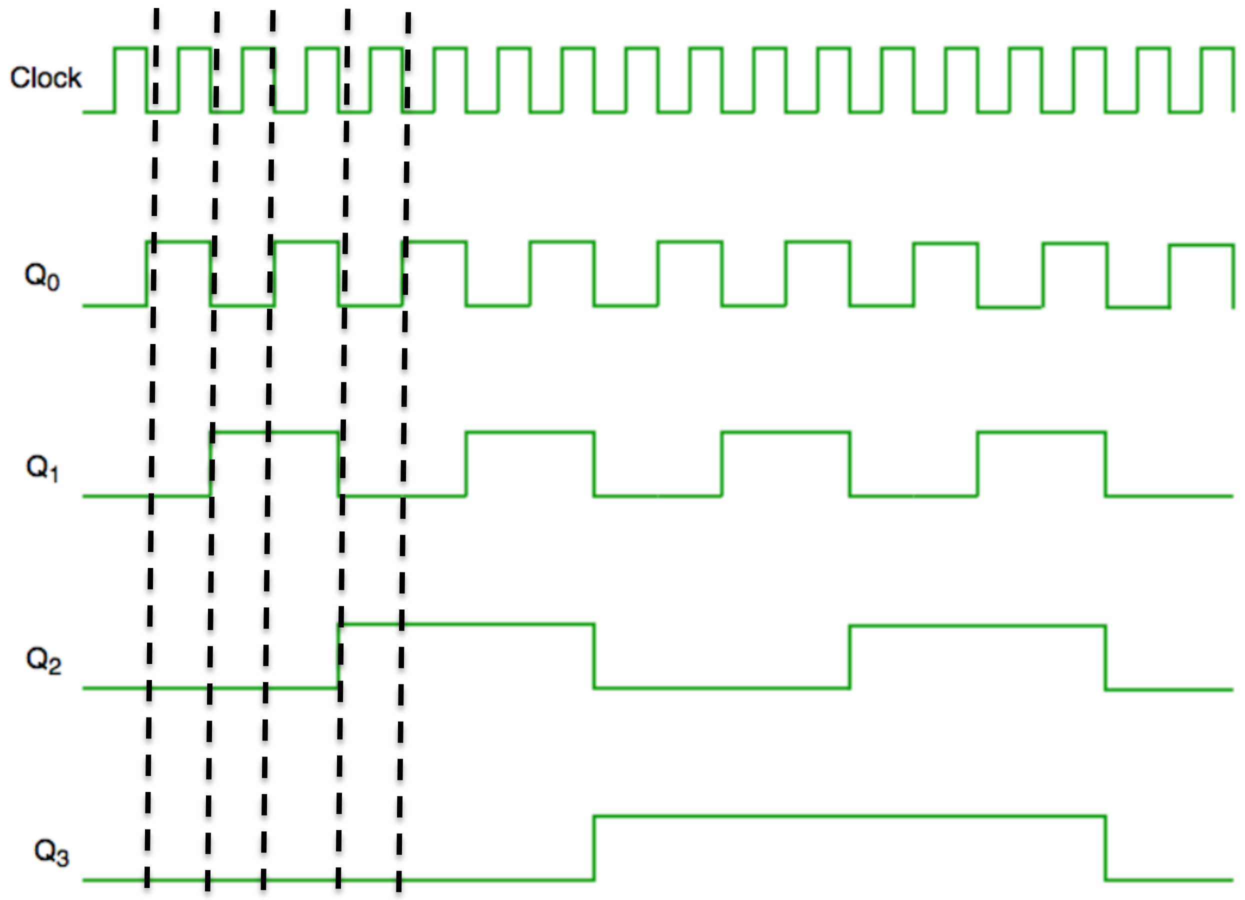
Ripple Counter



Ripple Counter



RIPPLE COUNTER



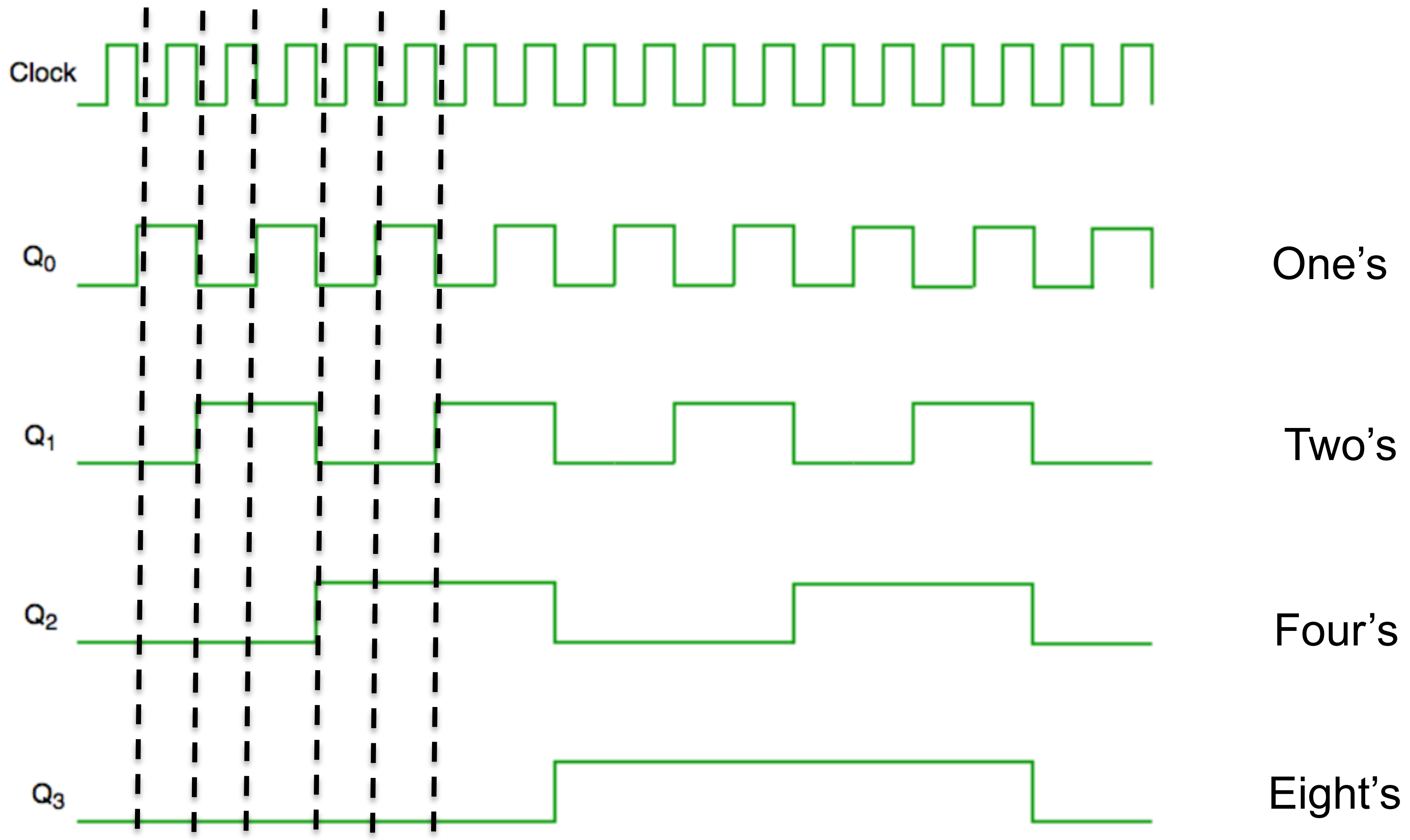
One's

Two's

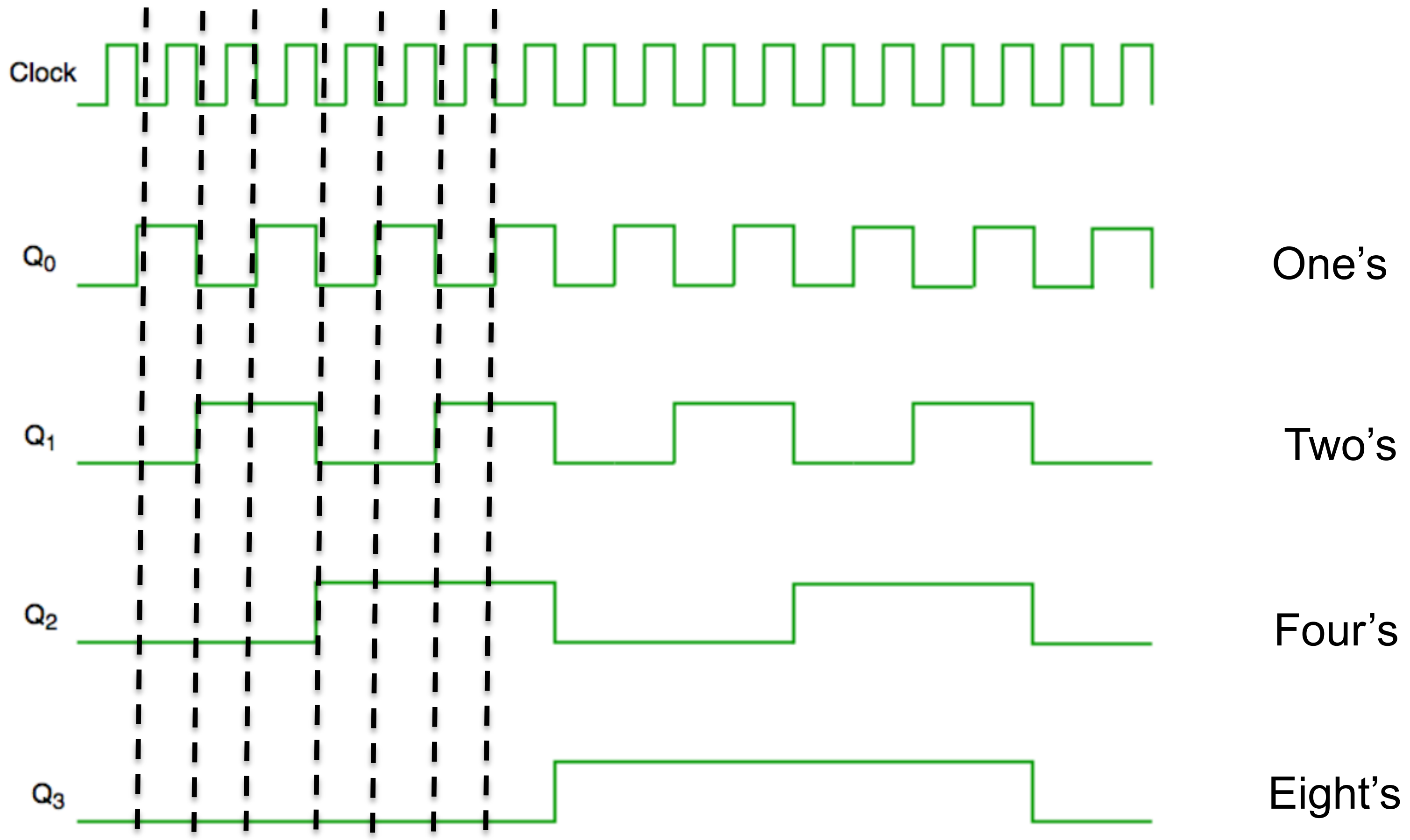
Four's

Eight's

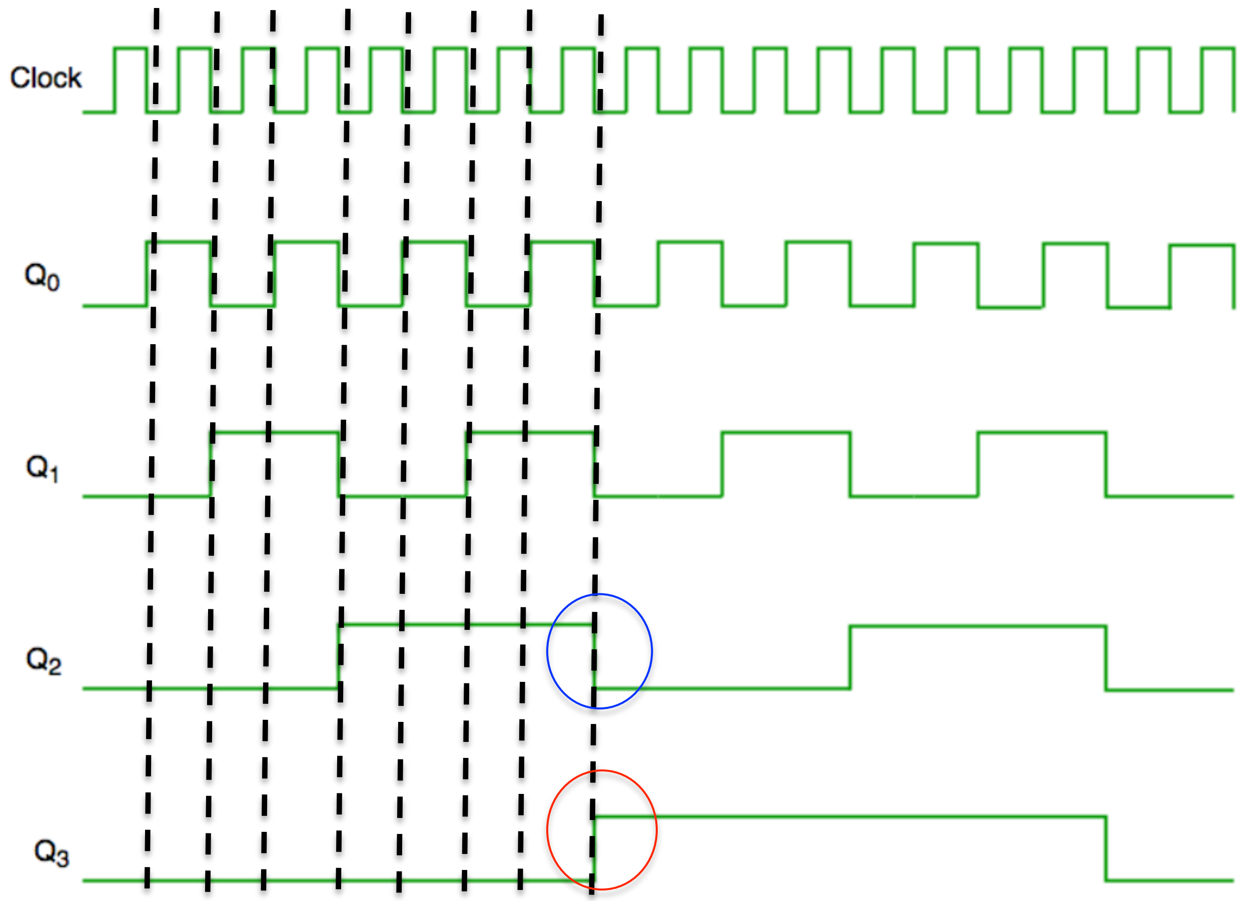
Ripple Counter



Ripple Counter



Ripple Counter



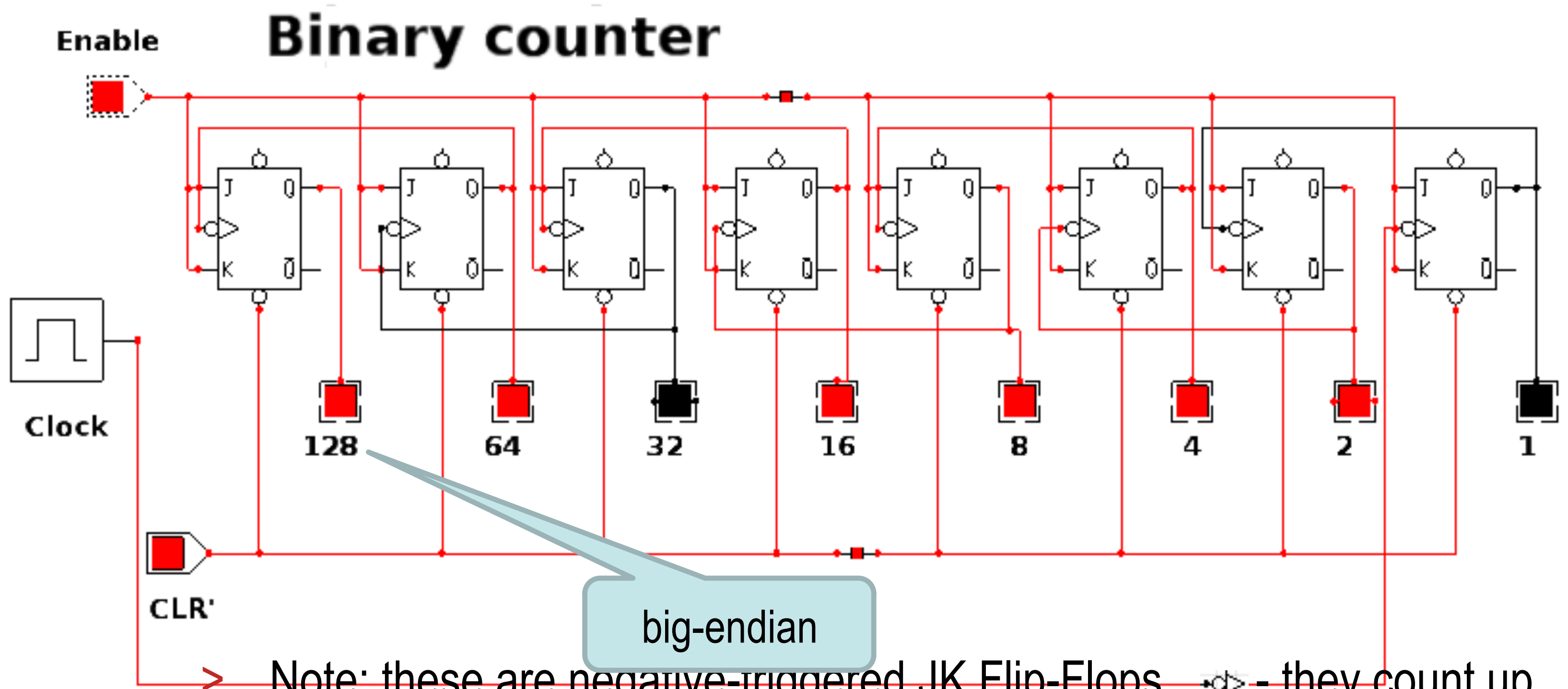
COUNTING DOWN ?

- What if I want to count down instead of up ?
- Requires nothing more than a change of FF trigger:
  - Change FF to trigger on rising edge of clock pulse
- Lets take a look

SUMMARY

- Ripple counter achieves a simple binary counter
- It utilises the J-K toggle setting:
  - J = 1, K = 1,
- It is asynchronous because:
  - No common clock
  - State changes ripple through from least significant FF
- We can change directionality of counter using FF trigger:
  - Falling edge = increment
  - Rising edge = decrement

COUNTERS USING J K FLIP-FLOPS

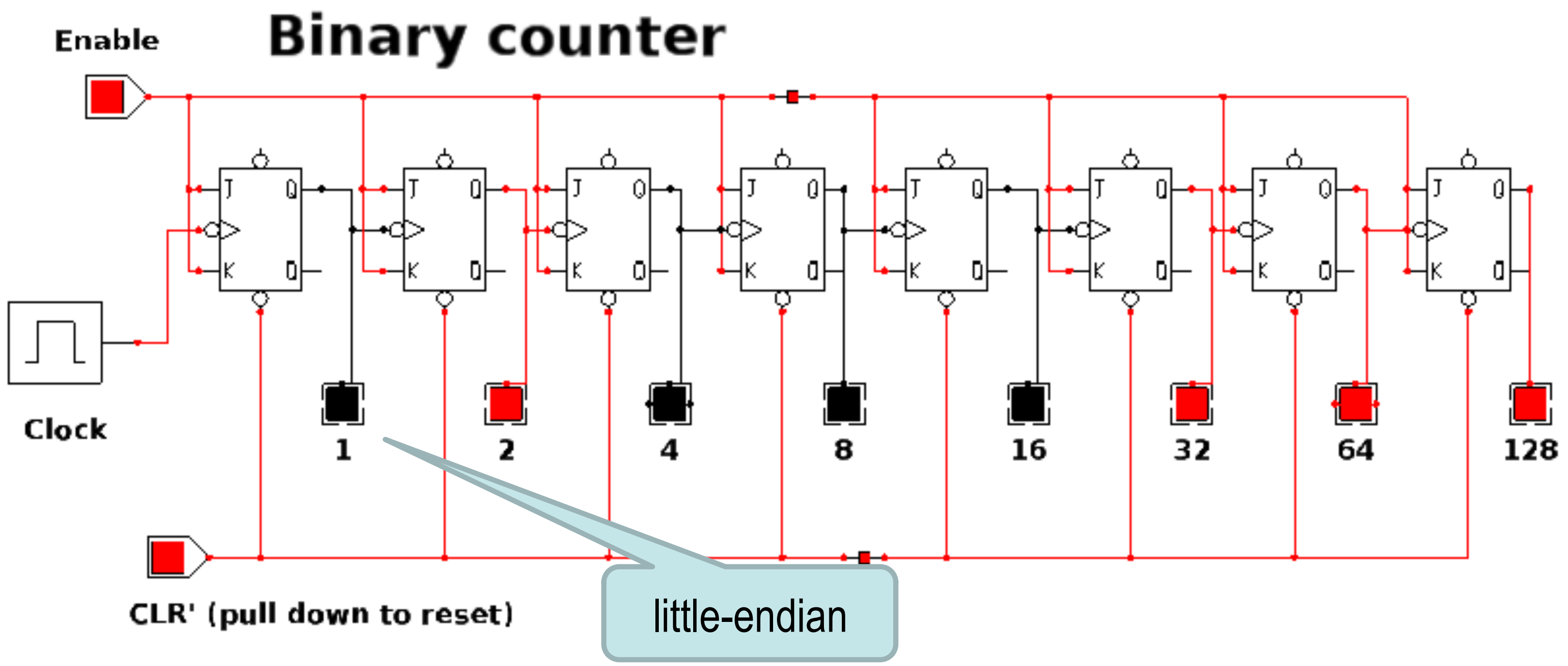


> Note: these are negative-triggered JK Flip-Flops - they count up.

> If you use positive-triggered ones, and connect each clock input to the previous **Q** they count down.

> There are other ways of making a down counter – most rely on using controlled gates to change the wiring between the flip-flops (more later).

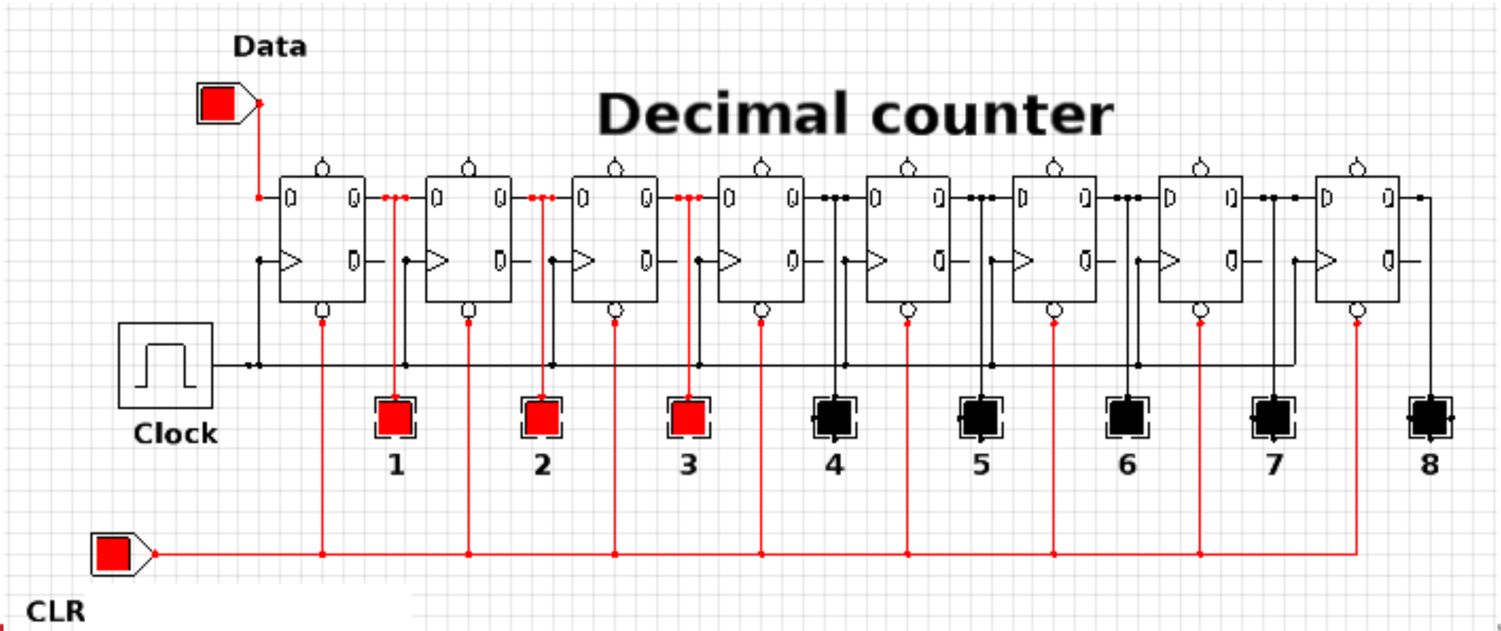
COUNTERS USING J K FLIP-FLOPS



- We can change the interconnections to make a little-endian version
- The Enable and CLR' pins are the same. The connection of each Q to CLK is reversed.

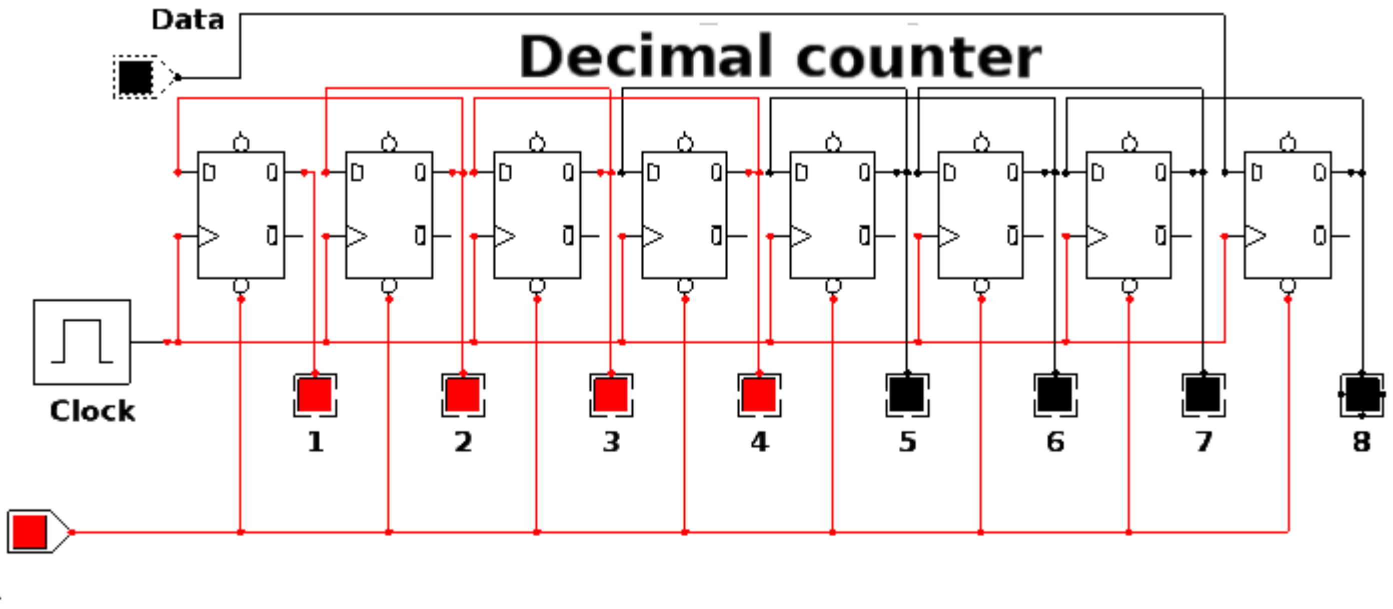
A Level counter?

- Why not? Great for meters, progress displays
- Use D Flip-flops, common clock, Q connected to next D input.
- Increments on each clock pulse as long as D is held high.



DECIMAL COUNTING?

- Count down? Wire up the Q---D connections backwards
- Decrements on each clock pulse as long as D is held low.



HMMM.

> Later, we will need to make a circuit which can go up or down depending on the state of a switch.

> We will use controlled gates to enable or disable individual connections between adjacent flip-flops.

> Alternatively we could use a set of flip-flops with two clocks...

> One for up; one for down.

MORE COUNTERS

> Suppose we want to count in a modulo that is not a neat power of two? For example 6

> Need 3 flip-flops but change the sequence to be 0,1,2,3,4,5,0 etc. (The states 6 and 7 are never used)

We can hard-wire this... with gates!

There are two ways in general to do this:

- **Detect the *first illegal state* (6 in this case)** and immediately reset to 0 (don't wait for the clock) and hope no-one notices.
- **Detect the *last legal value* (5 in this case)** and force the next clock pulse to reset us to 0.

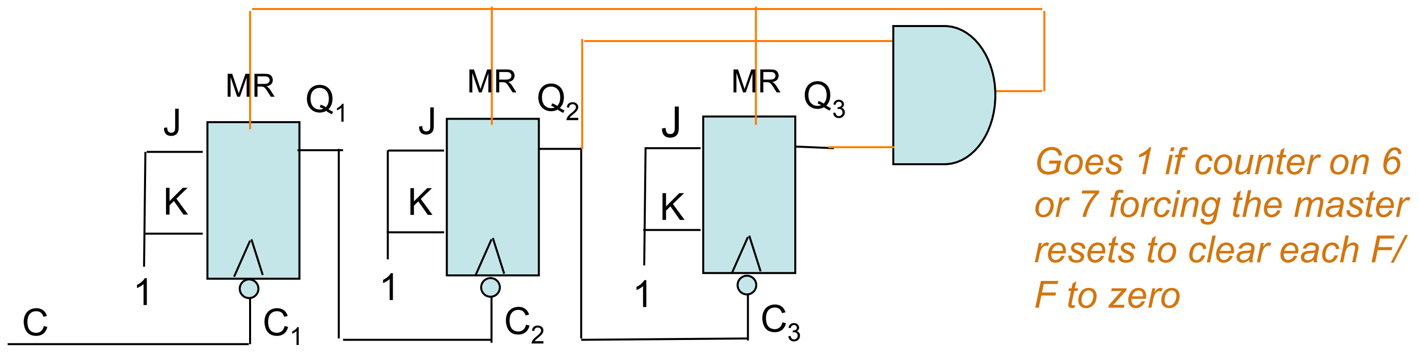
MODULO 6 WITH MOMENTARY ILLEGAL STATE

- Lets build it and see

MODULO 6 COUNTER WITH A MOMENTARY ILLEGAL STATE

> Detecting the first illegal state (6 in this case) and immediately resetting to 0 (don't wait for the clock) by using the asynchronous master reset (MR) or CLR'

  – This circuit uses a *cascading clock*


MR
MR
MR
Q_1
Q_2
Q_3
J
K
J
K
J
K
1
1
1
C
C_1
C_2
C_3
Goes 1 if counter on 6 or 7 forcing the master resets to clear each F/F to zero

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.

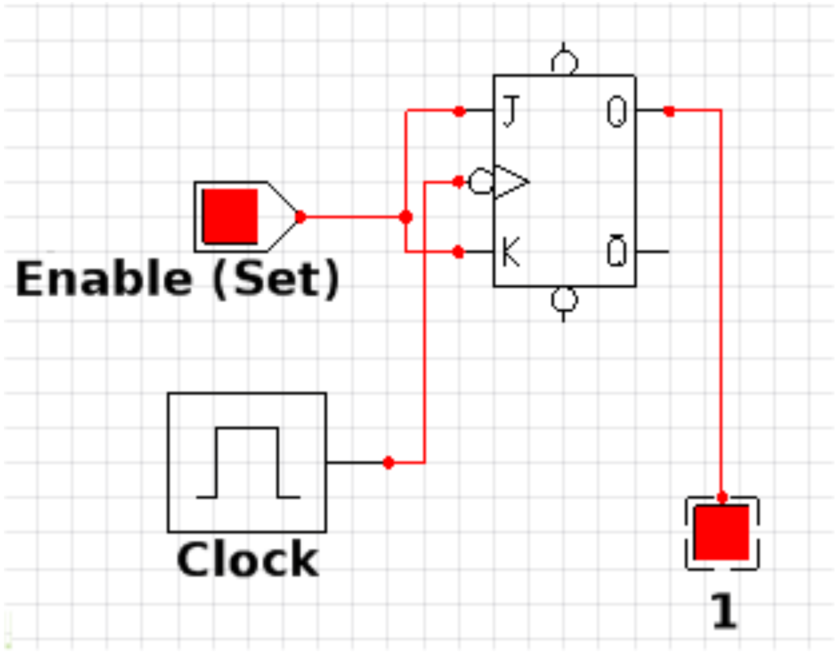
> We need a common clock

> Lets build it up from 1 to 3 bits …..

Mod 6 Counter with common Clock

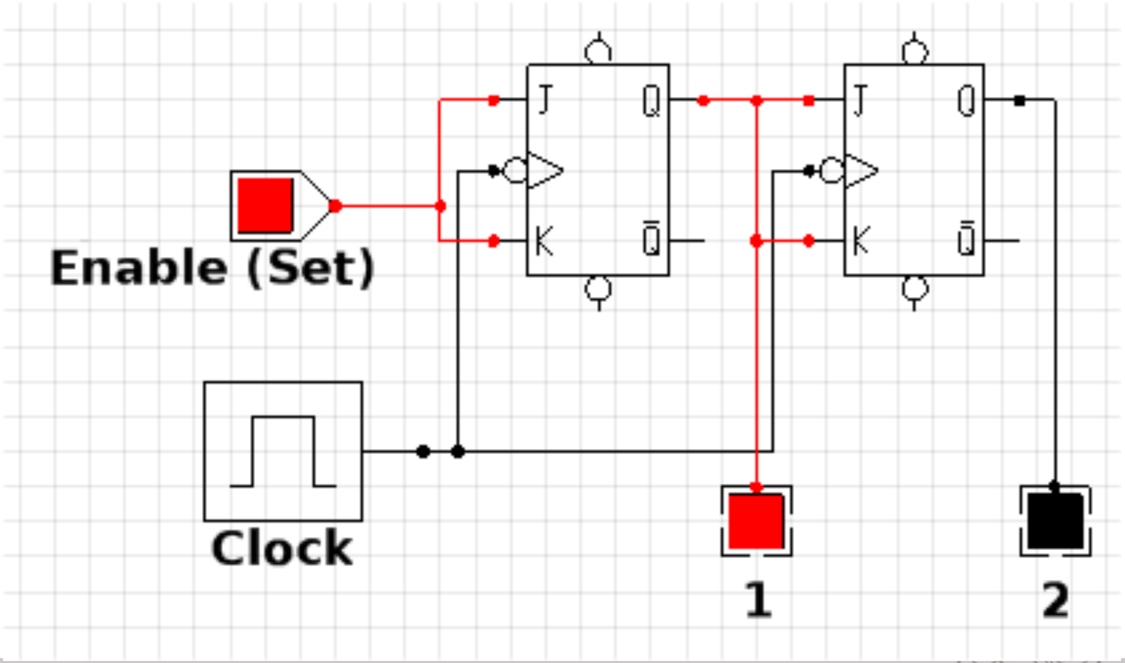
- First a 1 bit counter
- 1-bit (counts 0...1...0...1...)
- Set J and K to make it toggle

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.

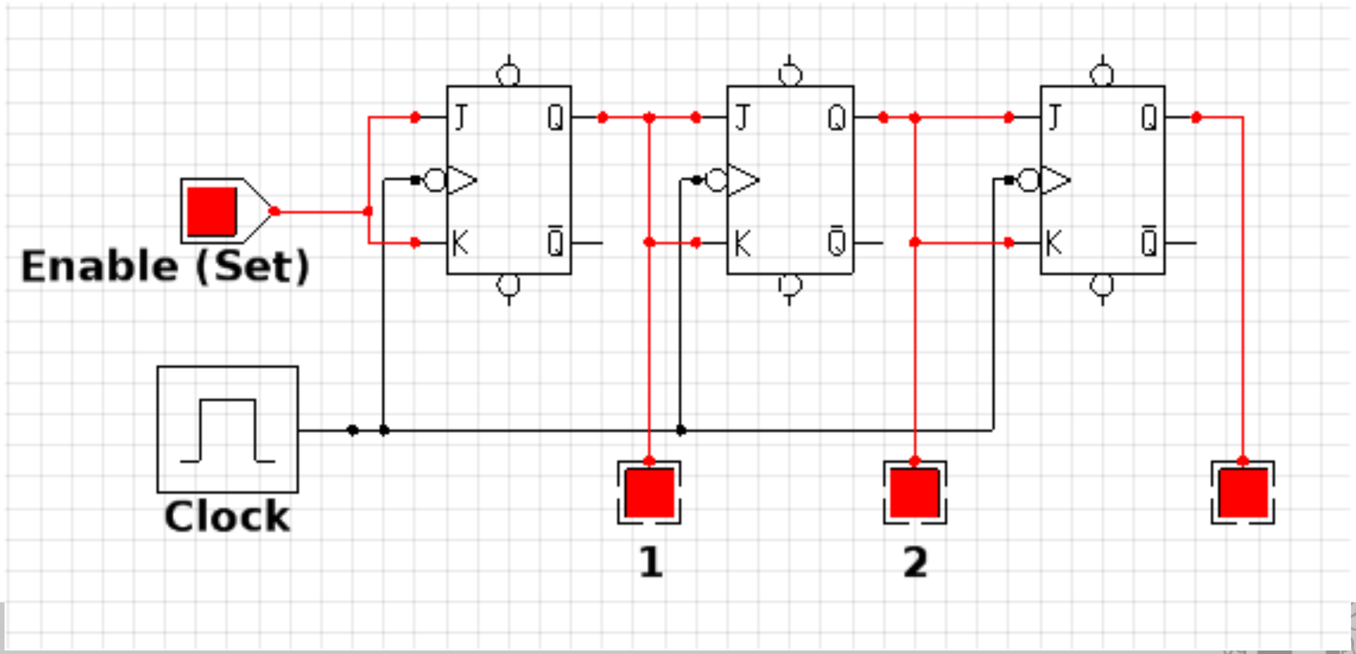


3-BIT COUNTER

> 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)

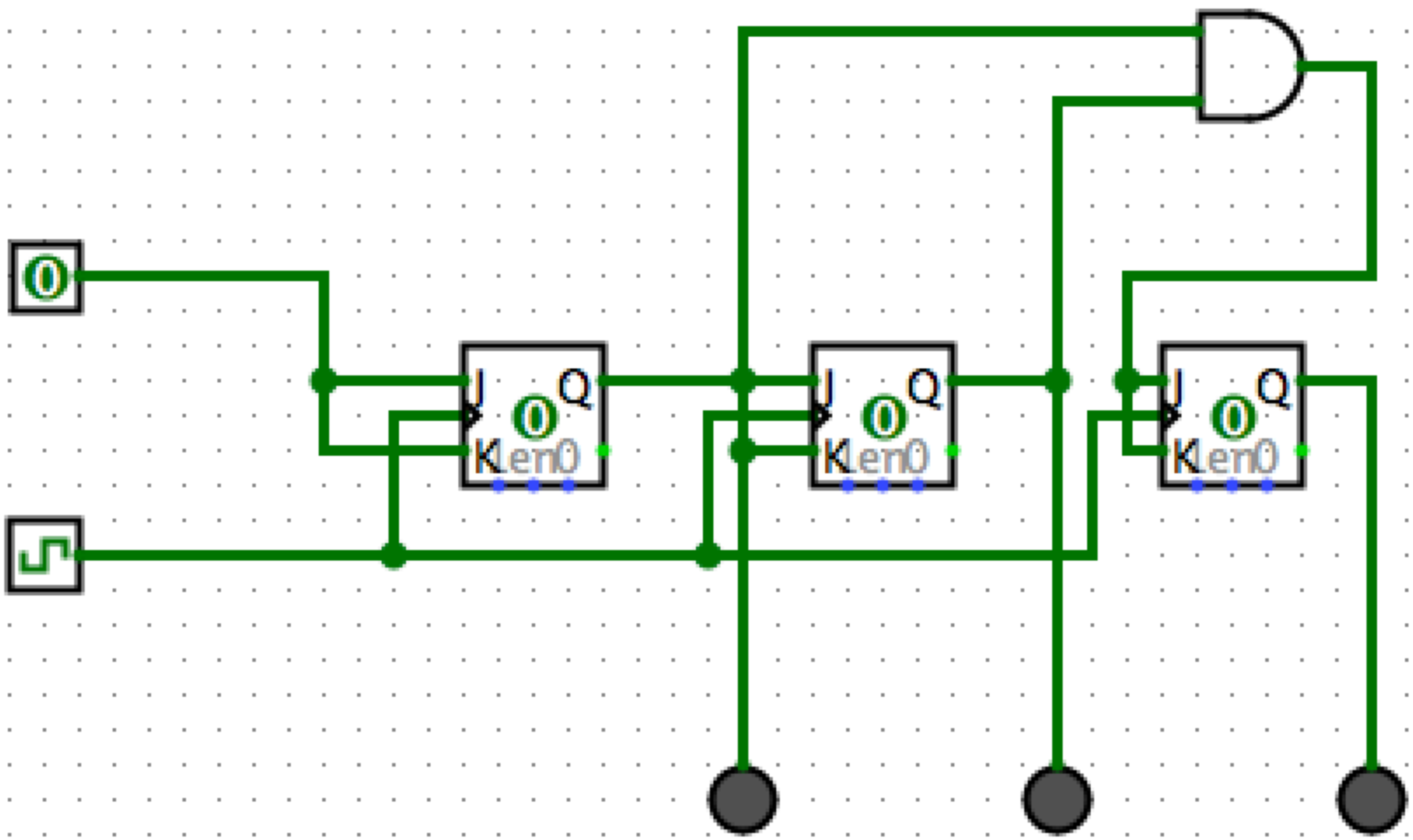
The issue is that the gates are all using the same clock, so Q3 will toggle at 3, and then the counter will reset to 0.
Try it, You will get the number sequence: 0, 1, 2, 7, 0...

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 –

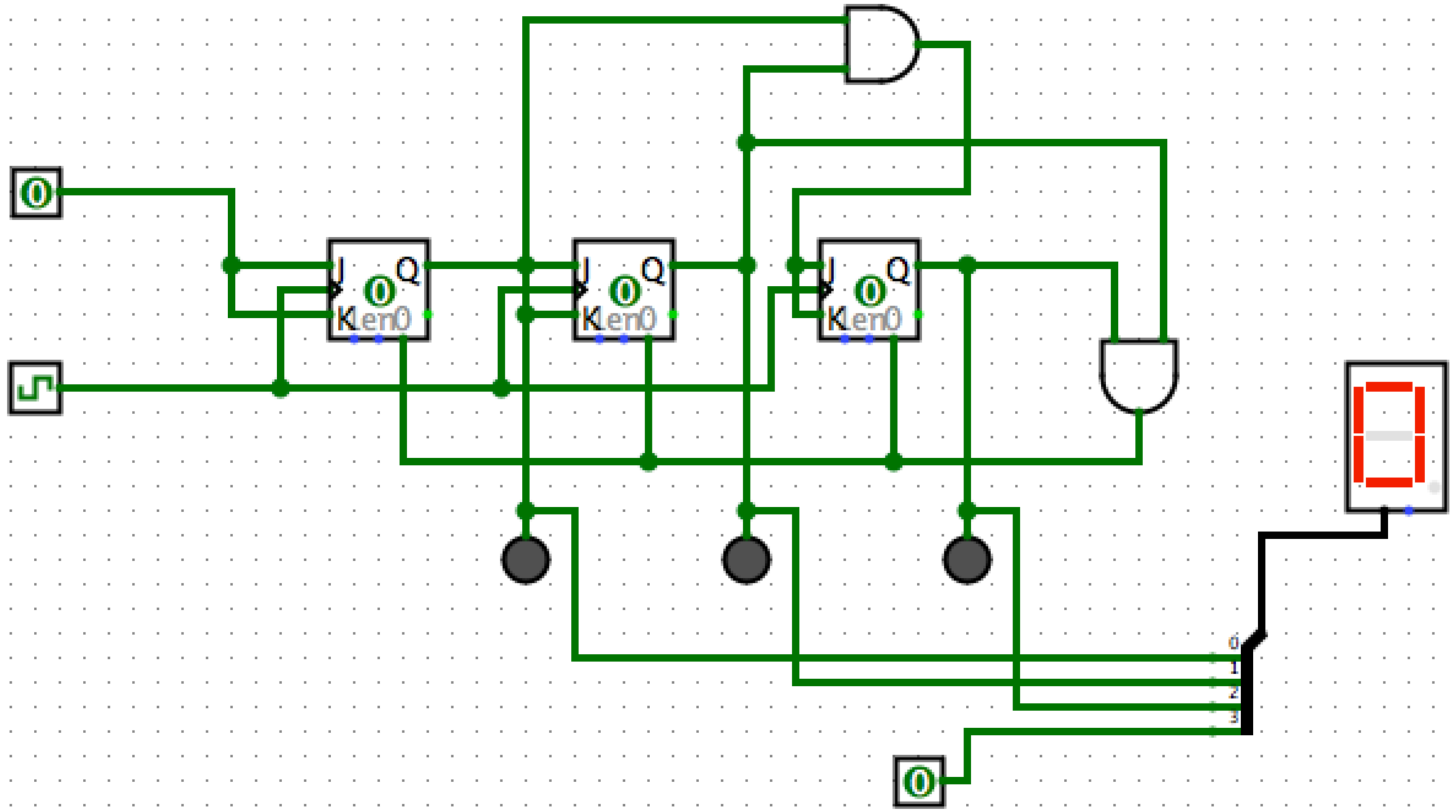


Now To reset when it gets to 6 (illegal state)...

> 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.

Logisim Time - Lets just build it!

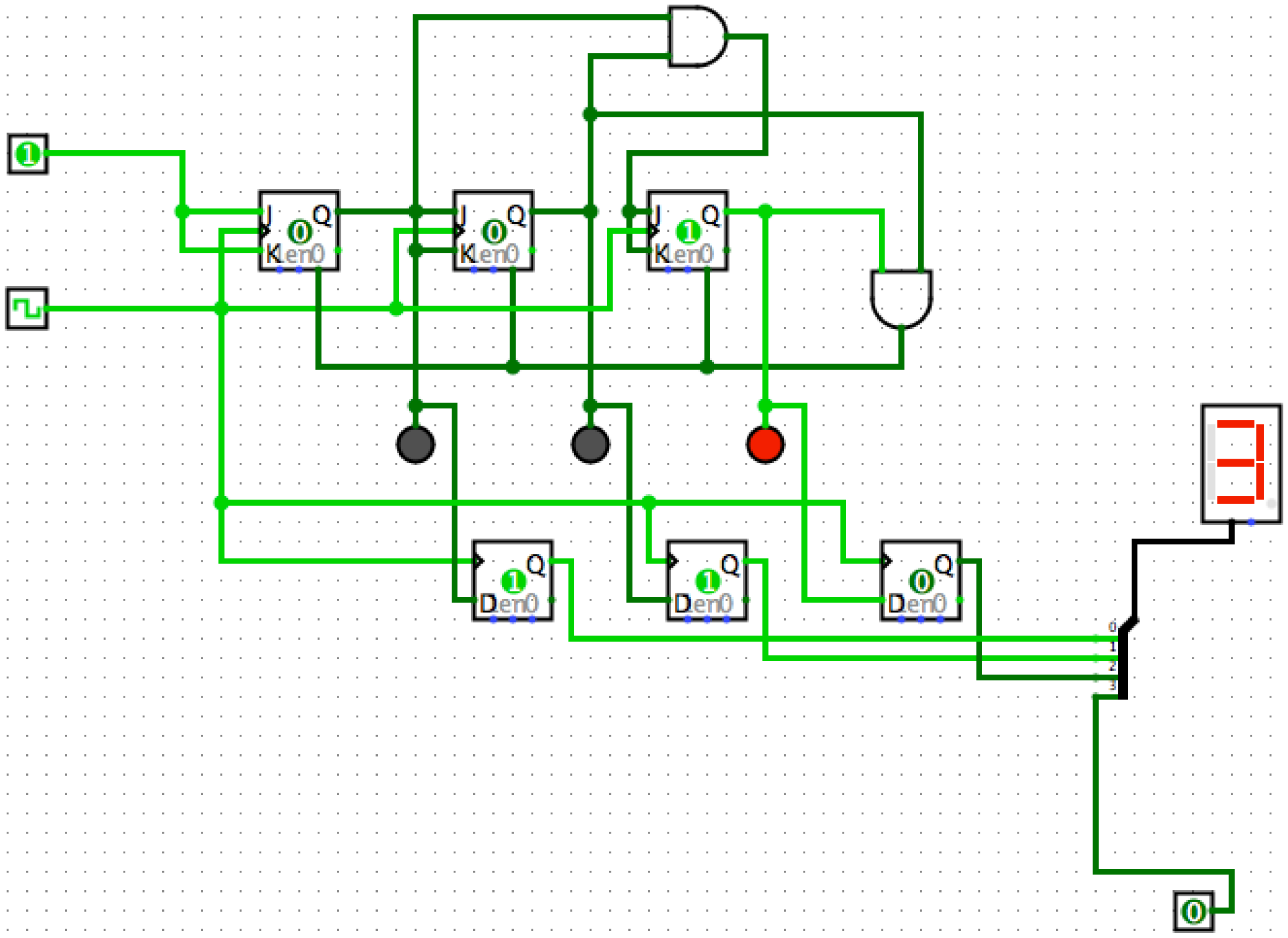
3 bit Common Clock – Little Endian –



Issues

- To prevent illegal state, add D-Flip-Flops as a buffer
- only the valid counter states are available for display
- Adds a delay of
  1 clock pulse.

3 bit Common Clock - Little Endian - No illegal state



`

push

RESOURCES

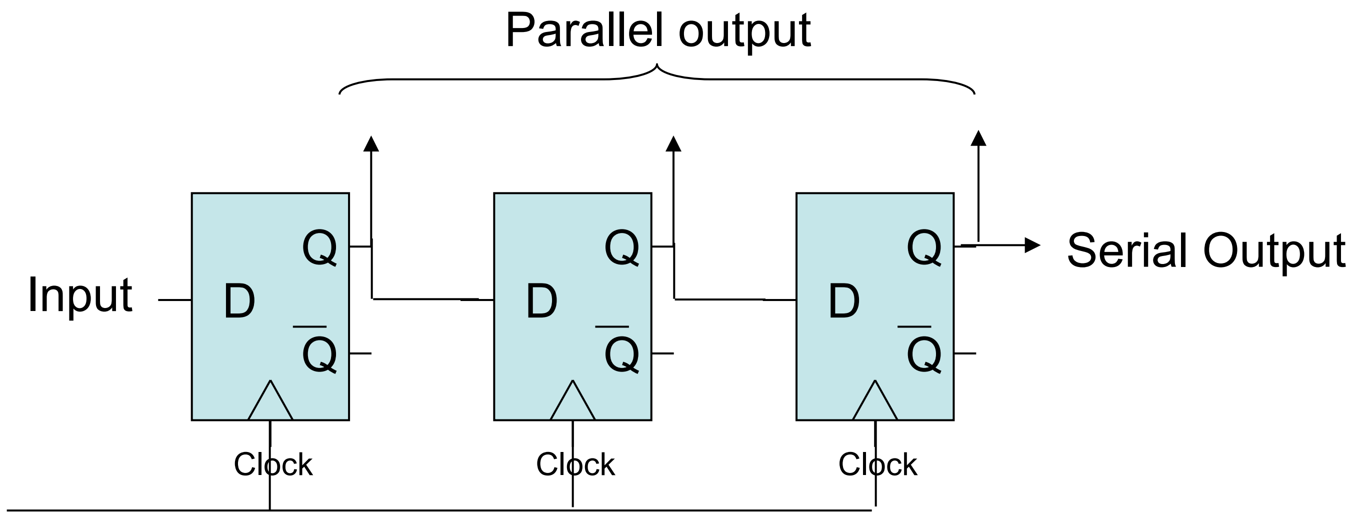
- Video tutorials:
  - A quick reminder that I am progressively making video tutorials for *some* key topics.
  - Not all will be covered, but I try to give practical and quick walk throughs of circuit designs (and later, ASM code)
  - They are linked from the lab sheets.
- Discussion board

pop

Shift Registers

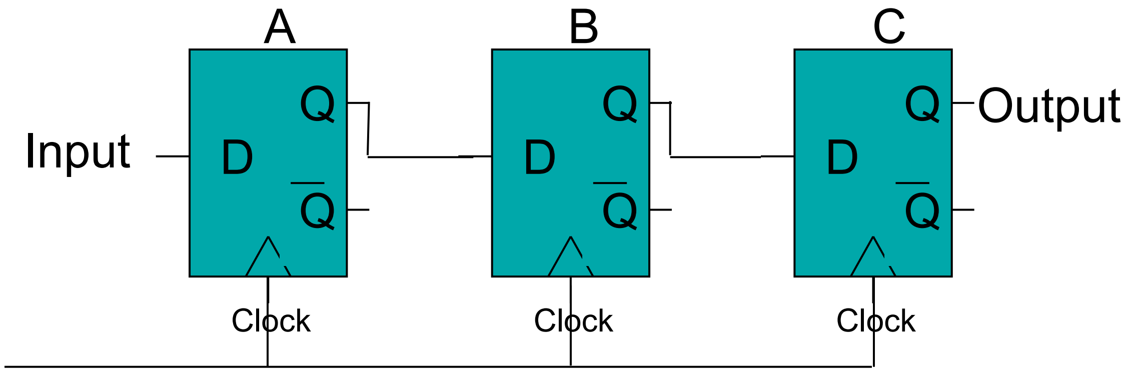
A shift register takes input from one end, and at each clock change this value is moved to the next D-Flip-Flop.

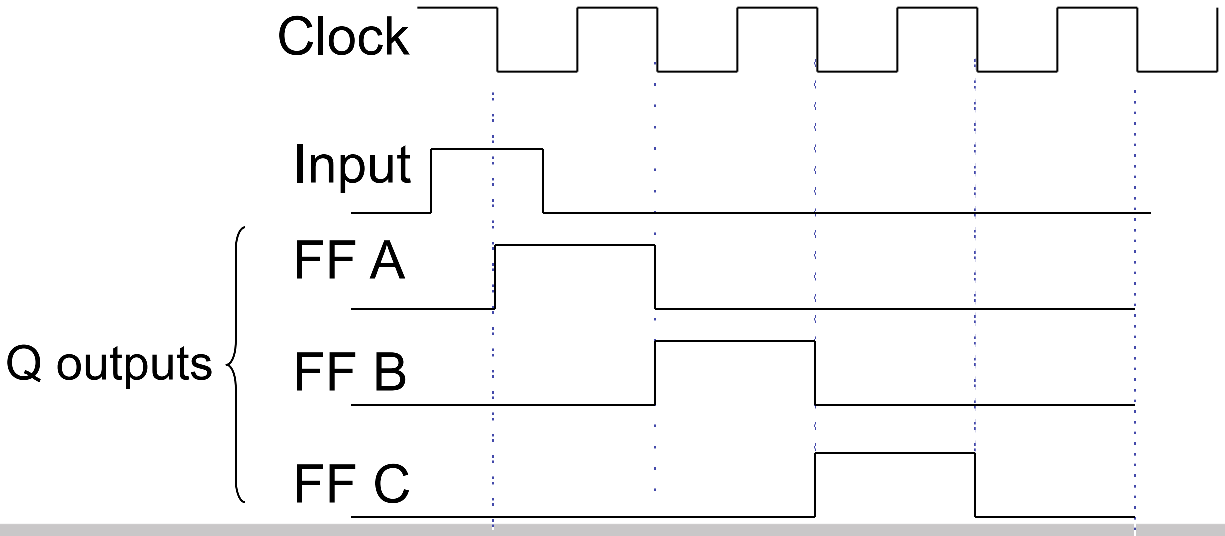
This is used in serial data transfer when a byte (say) of data sent on a cable one bit after another can be collected in a series of D Flip-Flops to rebuild the whole data byte. This is called *serial-to-parallel* conversion.



Shift Registers

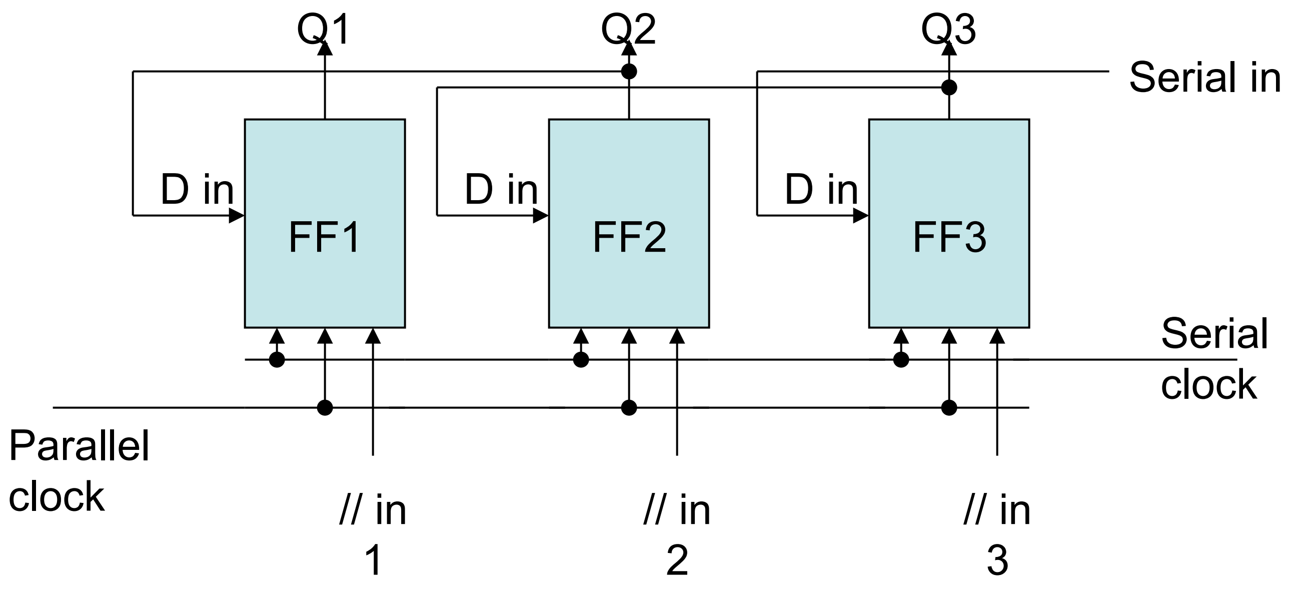
Here is how a *high* travels through a 3 bit shift register. For this example we assume that each of the shift register bits is cleared at the start.





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.

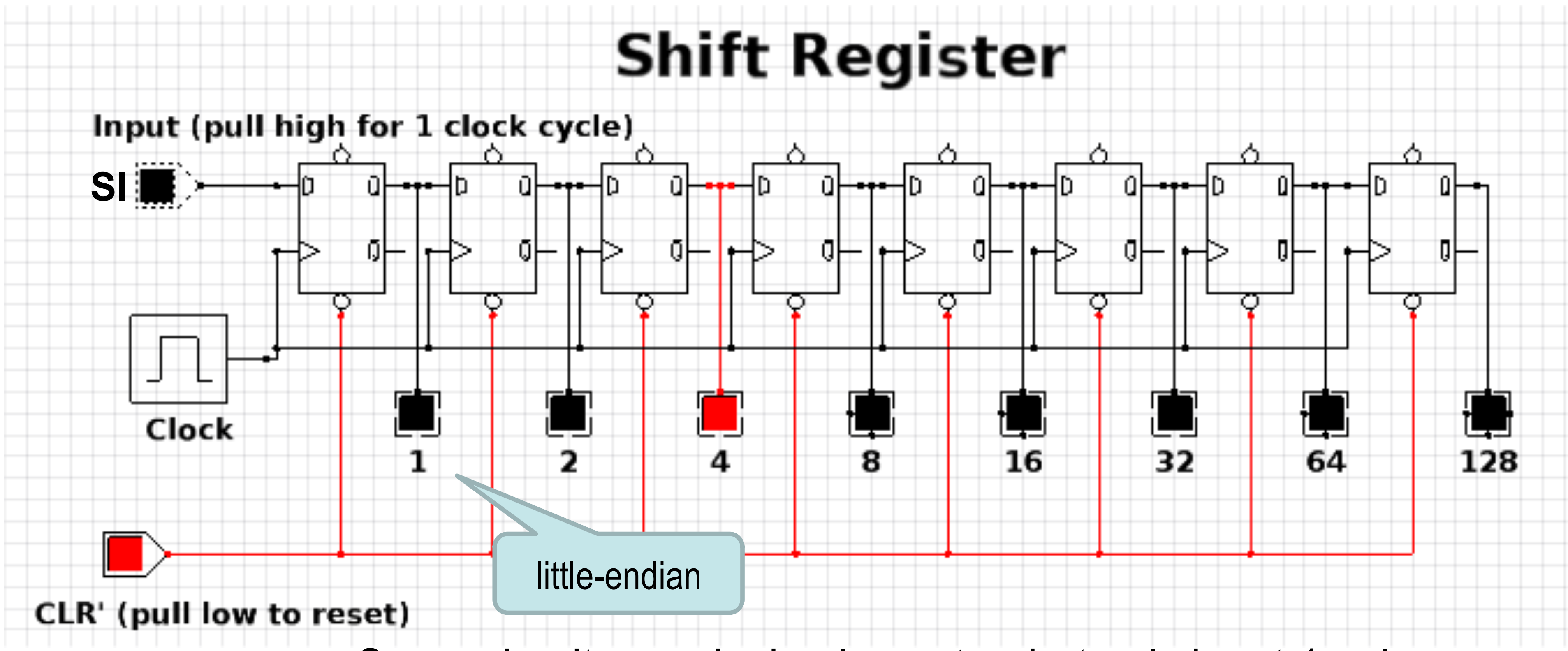


LETS BUILD ONE !

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

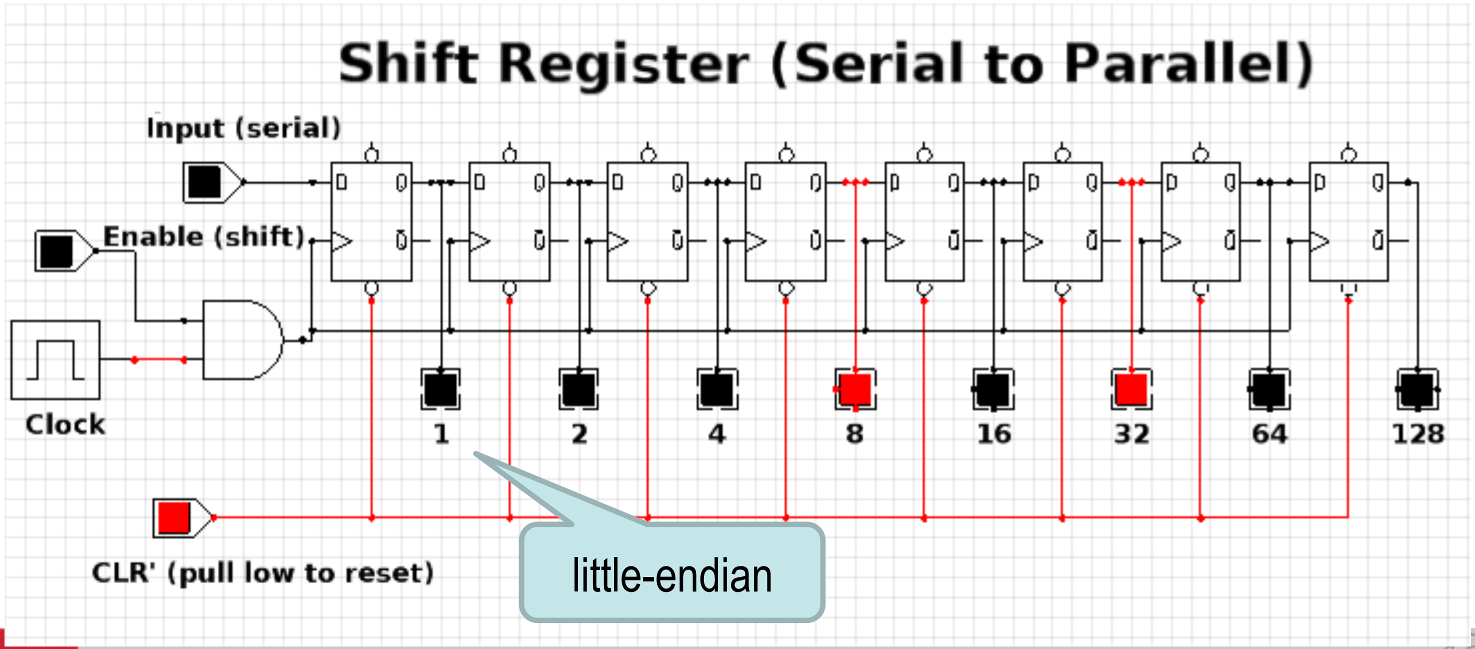


> Same circuit as a decimal counter, but only input 1 pulse instead of holding Input (Data) high.
> Only does serial to parallel.

> To shift *n* bits, input *n* clock cycles.

Left-shift (low bit to high bit) Serial to Parallel

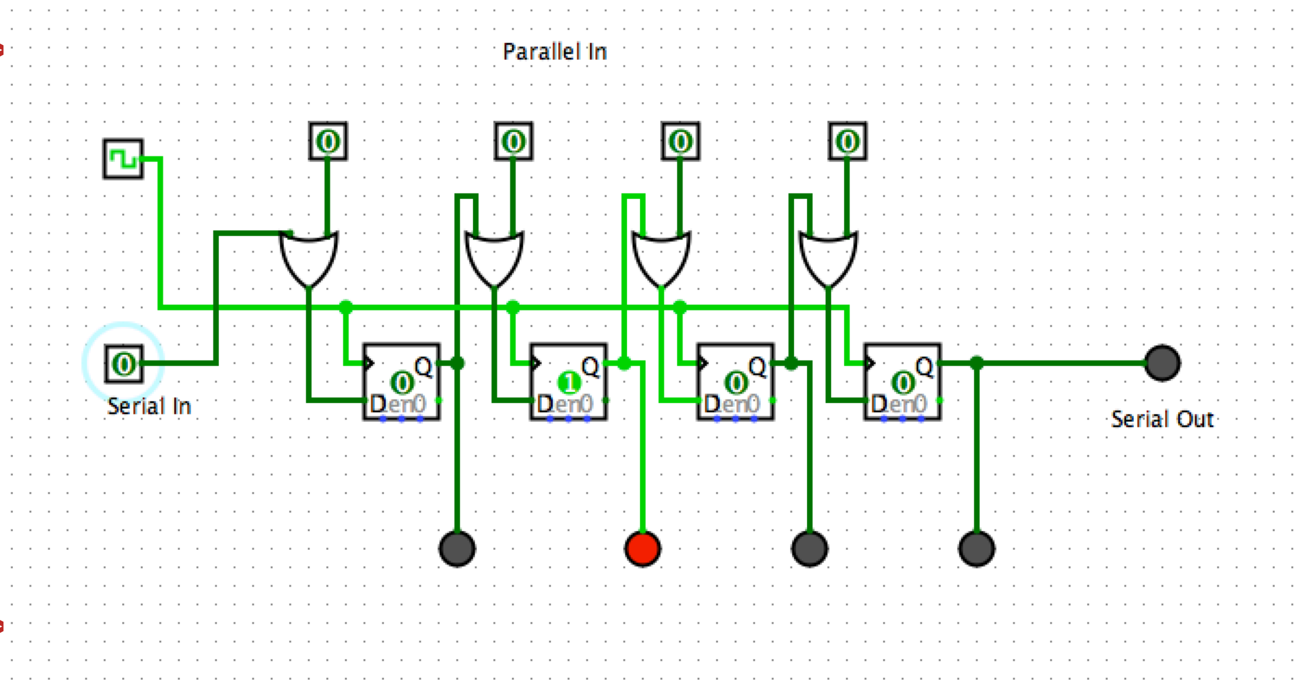
- Can modulate the Input (serial input) to load states into a register (serial to parallel conversion)
- Once the "conversion" is complete, disable the clock and store in a latch/register.



PARALLEL LOAD SHIFT REGISTERS

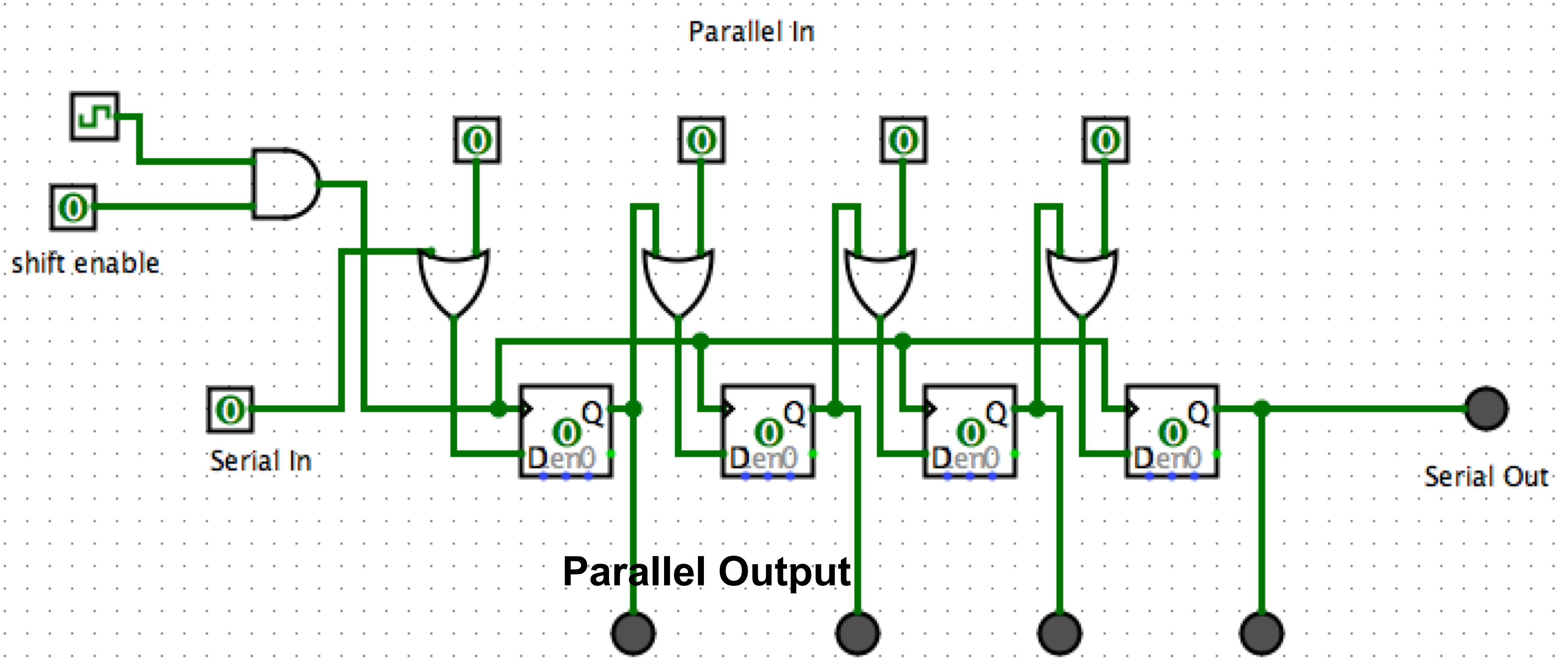
> Some shift registers allow all flip-flops to load at once,

>



>

RIGHT-SHIFT SERIAL OR PARALLEL INPUT: 4-BIT



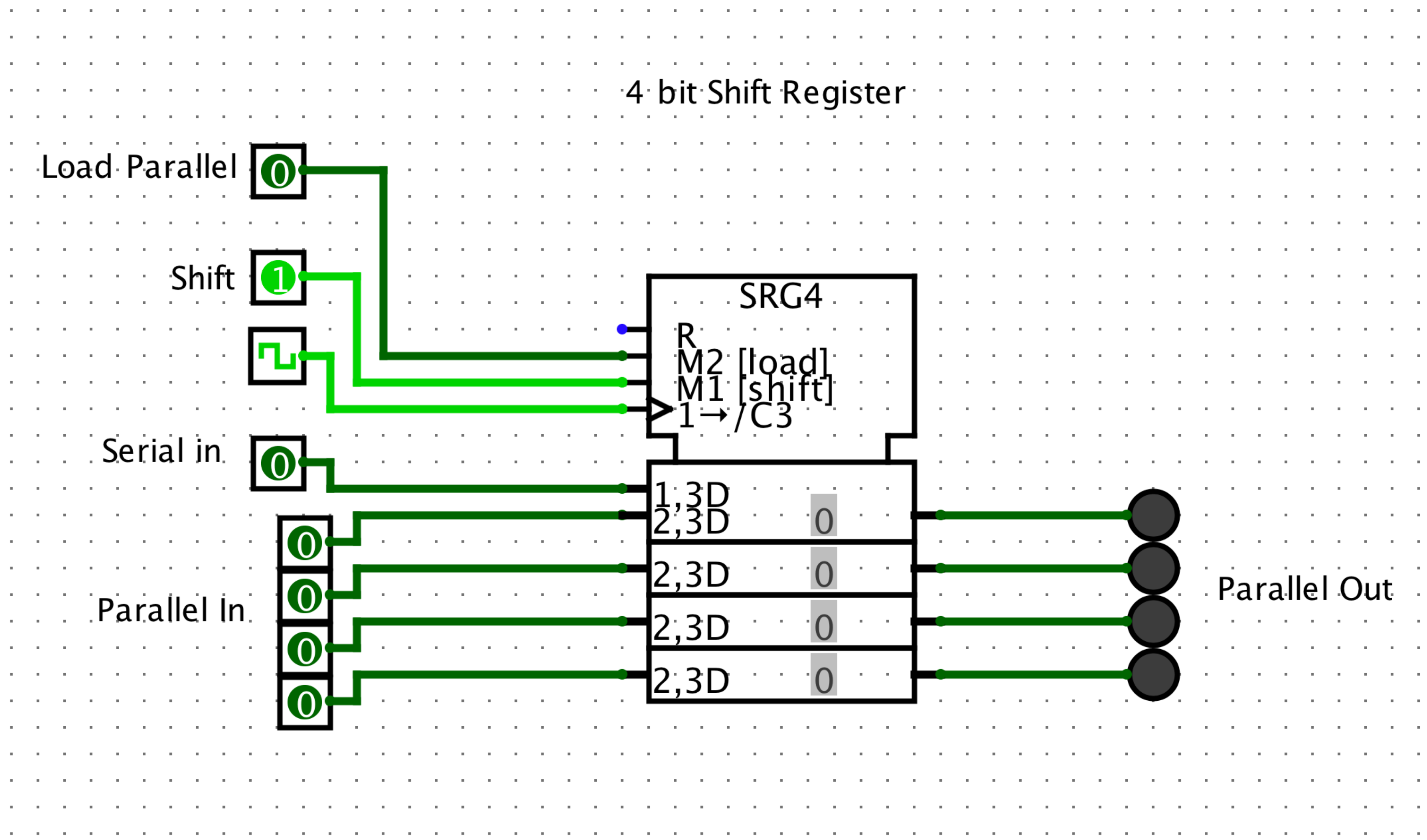
> Flip-flops are connected (output to input) with a common clock to cascade input from high bytes to low bytes.

> 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

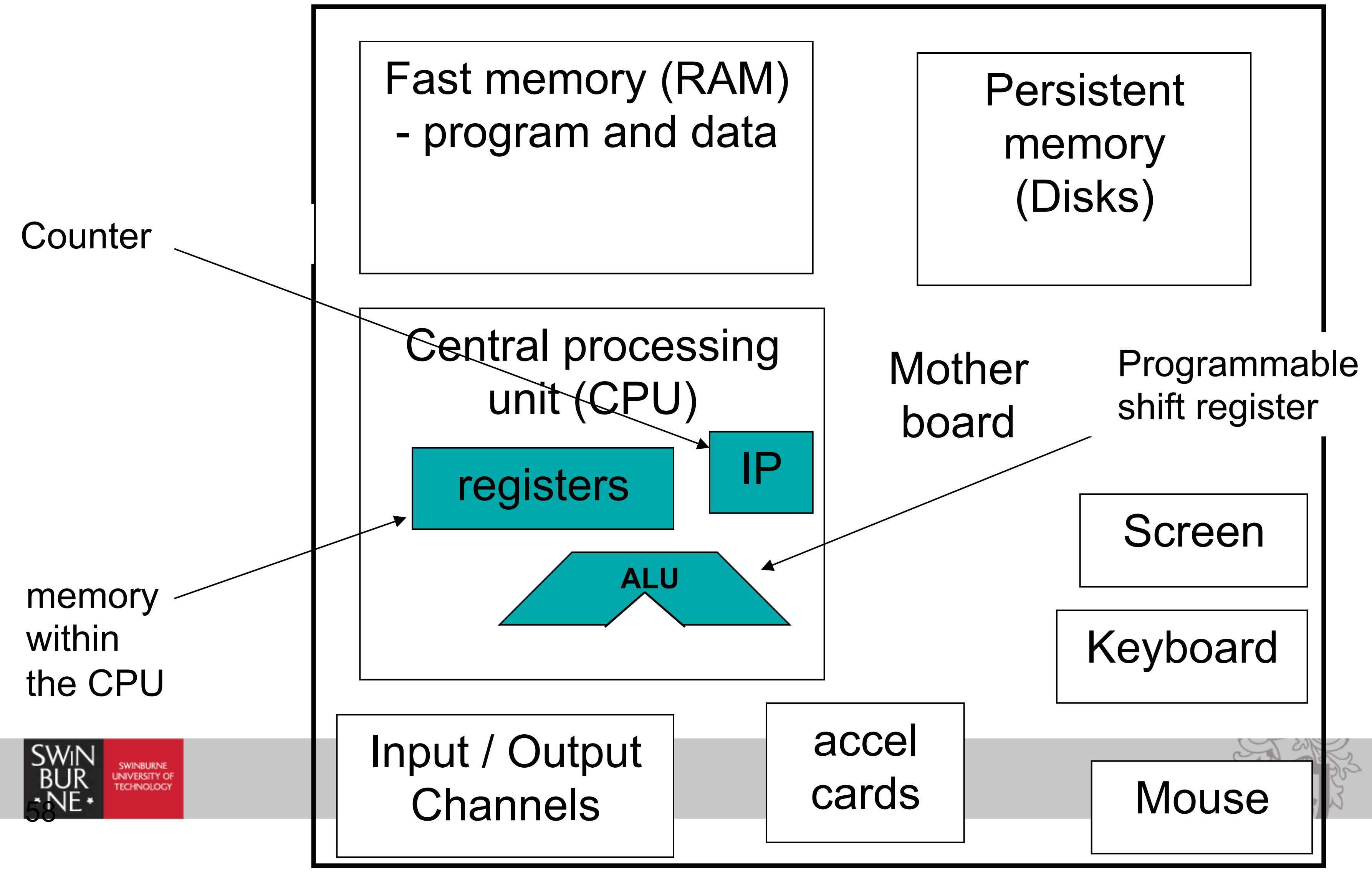
THE LOGISIM SHIFT REGISTER



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)
registers
IP
ALU
Mother board
Programmable shift register
memory within the CPU
Screen
Keyboard
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 <u>up</u> counter triggers an RS (or JK) flip-flop to enable the <u>up</u> wiring, and the <u>down</u> button unsets the RS flip-flop and enables the <u>down</u> wiring.

IN THE LAB...

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