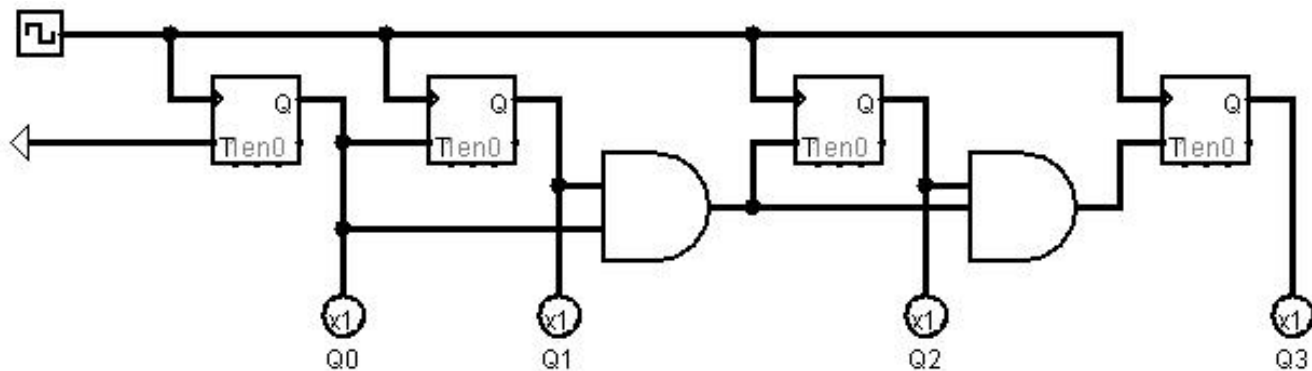


1. Counter

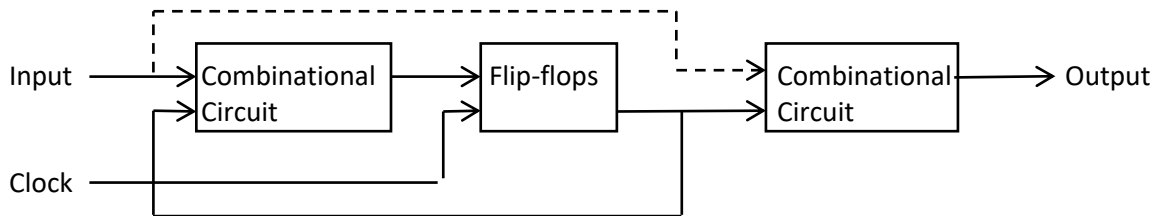
- a. Register whose value is easily incremented by 1
- b. Eventually “saturates” and reaches maximum value
 - i. When this happens, up to designer to decide what happens
 - ii. Can roll over, stay at value, start decrementing...
- c. Ripple (asynchronous) counters
 - i. Each FF waits for the previous FF before changing the signal
- d. Synchronous counter
 - i. Counter that has all FFs change at the same time
 - ii. Far faster than a ripple counter
 - iii. How example below counts
 1. Start with all 0s
 2. When first flip flop (Q0) turns to 1, it's toggled back to 0 on next cycle
 3. Next flip flop (Q1) in sequence turns 1 due to Q0 = 1
 4. When both of those flip flops turn to 1, because of the AND gate, third flip flop (Q2) turns to 1 as well
 5. Repeat for Q3 and further
 - iv. Below: synchronous implementation using T flip flops



2. Finite state machines (FSMs)

- a. Already seen these, actually
 - i. The state transition diagram that we drew for a S-R latch/flip-flop was an FSM
 - ii. Operation behind any latch or flip flop can be described by an FSM
- b. Definition
 - i. A circuit whose behavior can be modeled using the concept of “state” and the transitions between those states

3. FSM general format



- a. Moore model FSMs have outputs only dependent on the current state
 - i. Ripple counters, shift registers
- b. Mealy model FSMs have outputs dependent on the state and input
 - i. This is the dashed line in the diagram
 - ii. If input isn't clocked, output may change at any time in a clock cycle

4. FSM basic design steps

- a. Four step process, from word description to circuit
 - i. Turn a word description into a state diagram
 - 1. Seen this before
 - ii. Turn the state diagram into a transition table
 - 1. Also have seen this before
 - iii. Use transition table to make K-maps to simplify circuit
 - 1. Did K-maps previously, will use them again here
 - iv. Implement simplified circuit