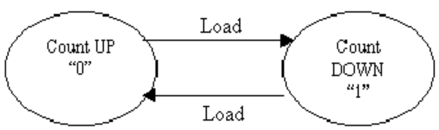
Fall 1998
Homework #4 Solutions

Professor R. Fearing
Solutions by Brian Choi and R. Fearing

1. FSM/Counter Combination

Simple FSM controller

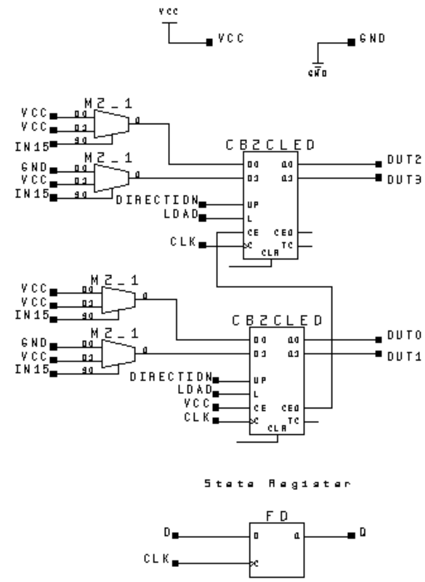
Note: Load is asserted when the counter reads “9” or “12”



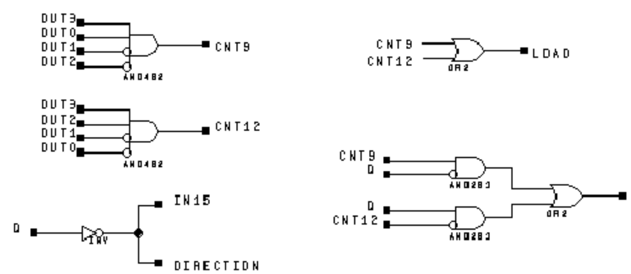
State Table

Load	Q	D
0	0	0
1	0	1
0	1	1
1	1	0

Schematic: Datapath



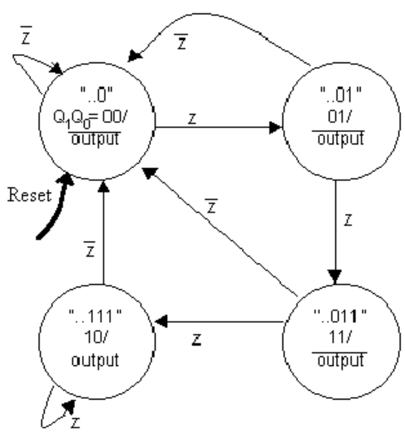
Schematic: Next State and Output Decoder



The outputs of the counters are shown as OUT[3:0]. The control signals CNT9 and CNT12 are just when the counter hits 9 or 12. The multiplexers are used so the FSM can merely state which number (5 or 15) need to be loaded. It is actually better to use this format, rather than combinational logic, since it allows the user to easily change what the loaded values are without changing any gates. The two states of the FSM are self explanitory. The machine is in one state as it counts up, and the other as it counts down.

2. Moore type FSM: Sequence Detector

State Diagram

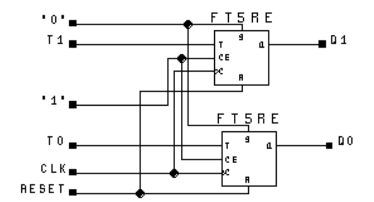


State Table

Reset	Z	Q1	Q0	Q1+	Q0+	T1	T0	Output
1	X	X	X	0	0	X	X	X
0	0	0	0	0	0	0	0	0
0	1	0	0	0	1	0	1	0
0	0	0	1	0	0	0	1	0
0	1	0	1	1	1	1	0	0
0	0	1	1	0	0	1	1	0
0	1	1	1	1	1	0	1	0
0	0	1	0	0	0	1	0	1
0	1	1	0	1	0	0	0	1

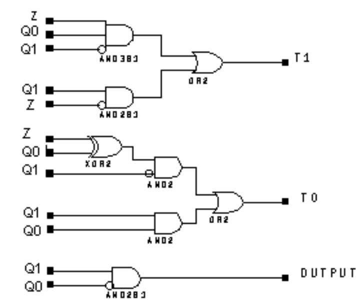
Note: The states go in the order of 00, 01, 11, 10 for detection. Since we are using toggle flip-flops, it is easier to just toggle one bit at a time. Reset will follow directly into the flip-flops, returning them to the zero state at the next clock cycle.

Schemate: Datapath



The clock enable is always set to high, as in, the flip flops are always active. The Set control lines are always low. This makes the flip flops act as generic Toggle Flip Flops.

Schematic: Next State and Output Decoder



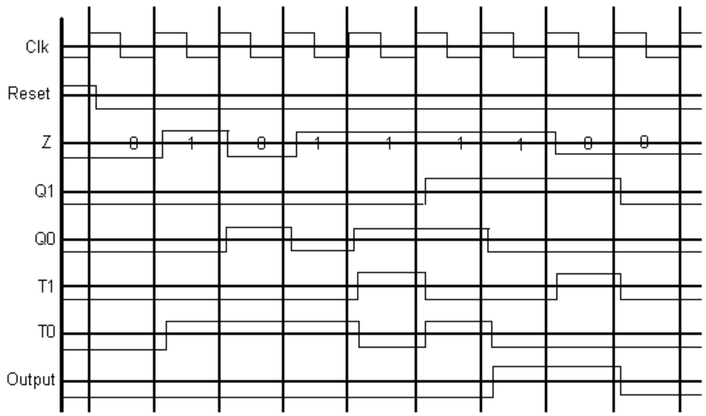
$$T = \overline{Q_1}\cdot(Z\oplus Q_0) + Q_1\cdot Q_0$$

$$T = \overline{Q_1}\cdot Q_0\cdot Z + Q_1\cdot\overline{Z}$$

$$OUTPUT = Q_1\cdot\overline{Q_0}$$

Note that the OUTPUT is only a function of the state. This is a requisite for a Moore Machine.

Timing Diagram for Detector



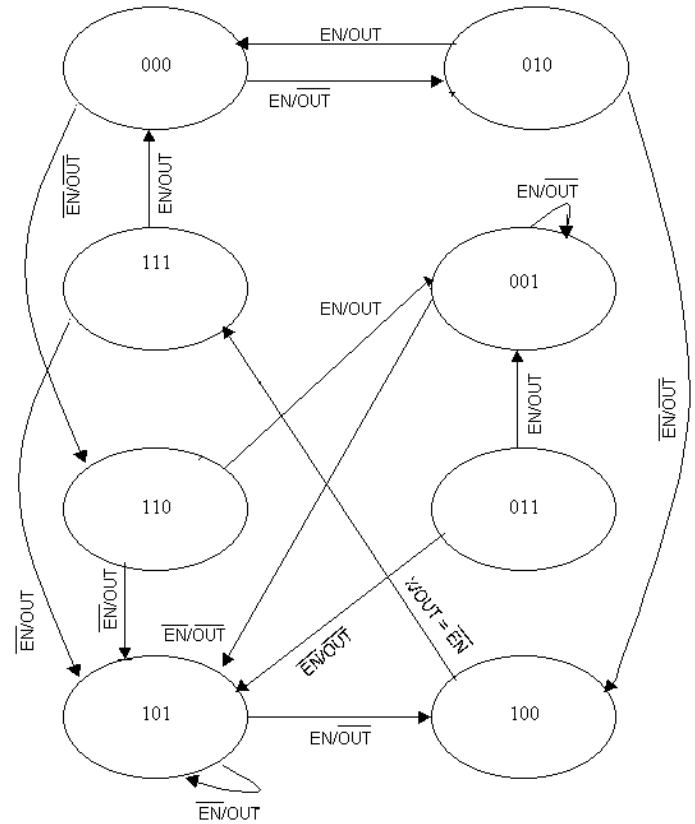
3. FSM Analysis

State and Output Table

EN	Q2	Q1	Q0	Q2+	Q1+	Q0+	T0	J2	K2	OUT
0	0	0	0	1	1	0	0	1	0	0
1	0	0	0	0	1	0	0	0	0	0
0	0	0	1	1	0	1	0	1	0	0
1	0	0	1	0	0	1	0	0	0	0
0	0	1	0	1	0	0	0	1	0	0
1	0	1	0	0	0	0	0	0	1	1
0	0	1	1	1	0	1	0	1	0	0
1	0	1	1	0	0	1	0	0	1	1
0	1	0	0	1	1	1	1	1	0	1
1	1	0	0	1	1	1	1	0	0	0
0	1	0	1	1	0	1	0	1	0	1
1	1	0	1	1	0	0	1	0	0	0
0	1	1	0	1	0	1	1	1	0	1
1	1	1	0	0	0	1	1	0	1	1
0	1	1	1	1	0	1	0	1	0	1
1	1	1	1	0	0	0	1	0	1	1

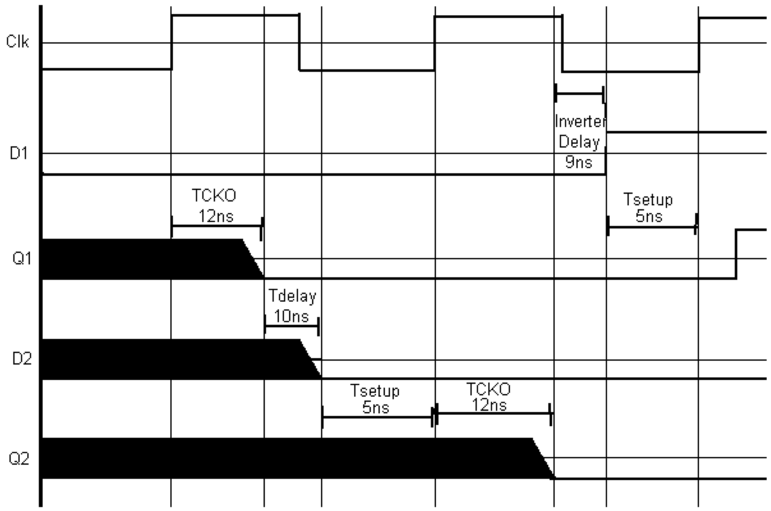
T0, J2 and K2 represent the inputs into the T flipflop and J-K flipflops. The Qx+ represents the values of Q at the next clock cycle.

State Diagram



4. Timing Analysis

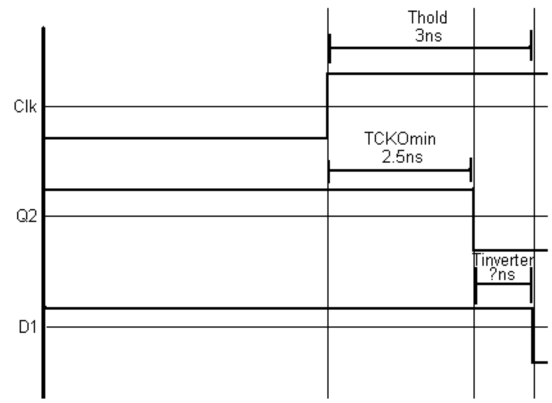
Shortest Clock Cycle



The shortest clock cycle represents the maximum delay for any of the stages in the datapath. Since there are two stages (one for each flip flop in series), we must calculate which stage yields the maximum delay. D1 and Q1 is the input and output of the leftmost flip-flop. D2 and Q2 is the input and output of the rightmost flip-flop. For ease of viewing, we say a signal is first asserted at D1 and must propogate to the system before the inverter can change the value. We will use the largest delay of each element to find the fastest clock period. The largest delay lies between the first and second flip-flop, yielding a total delay at 27ns, while the stage with the inverter yields a maximum delay of 26 ns. This means that the maximum clock speed the system may run, while remaining stable, is **27ns**

Period of clock=TCKOmax+Tdelay+Tsetup=27ns

Fastest Inverter



The point in question deals with the inverter betweeen the second and first flipflop, or between Q2 and D1 to preserve the convention used above. The issue is that the inverter may change too quickly and violate the hold time on the first inverter. The timings line up in the above diagram. It is easy to see that the inverter must have a delay such that the sum of it's delay and the TCKOmin must exceed the hold time on the flip-flop.

Tinverter min $\geq$ Thold-TCKOmin=.5ns

5.

There are 2^5 combinations of inputs for any boolean function. For each input combination, there are 2 values it can evaluate to. As a result, there are 2^{32} discrete boolean functions of 5 variables available. For one function of 5 variables, you would require 2^5 table entries.