Design of an FSM: an up/down counter

Madhav P. Desai

February 15, 2018

Suppose we are given the following building blocks:

- Inverters, NAND2, NOR2 gates each with a delay of 1 unit.
- D-flipflops with $Clock \rightarrow Q$ delay of 2 units, set-up time of 1 unit and hold-time of 1 τ unit.

We illustrate the design process of obtaining a logic circuit implementation of a Mealy FSM starting with an abstract specification.

1 The FSM Specification

- The set of input symbols is $\Sigma = \{reset,\ up,\ down\}$.
- The set of output symbols is $\Sigma = \{Y, N\}$.
- The set of states is $\Sigma = \{A, B, C\}$.
- The initial state is A.

The next state function and output functions are as follows

x(k)	q(k)	q(k+1)	y(k)
reset	_	A	N
up	A	B	N
down	A	C	N
up	B	C	N
down	B	A	N
up	C	A	Y
down	C	B	N

Note that the input symbol *reset* is used to put the machine in the initial state A.

The specification can be visualized by the state transition graph (STG) shown in Figure 1.

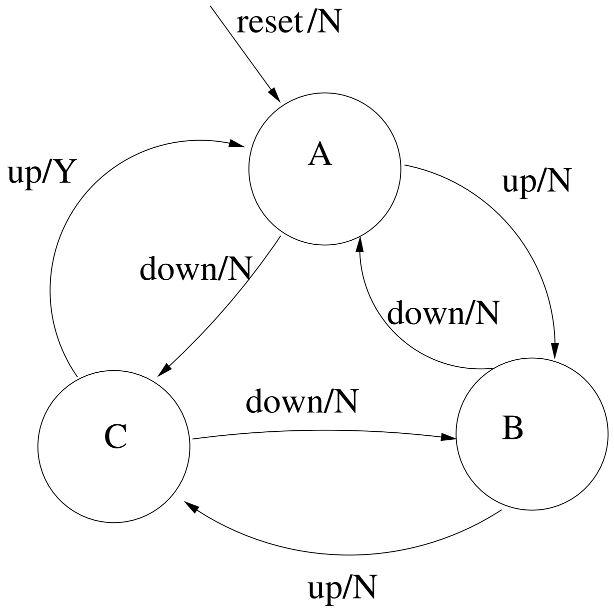

Figure 1: State transition graph of the up/down counter

2 Input encoding and combinational function implementation

First, we need to encode the input, output and state symbols. Let us use the binary encoding:

State	q1	q0
A	0	0
B	0	1
C	1	0

Input	r	x
reset	1	_
up	0	1
down	0	0

Output	y
N	0
Y	1

Thus, the output bit y has the truth-table

```
          00  01  11  10    q1q0
    00              d
```

01	d	1
11	d	
10	d	
rx		

which has the simplest formula $q1.\overline{r}.x$.

The next state variable $nq1$ has the truth-table

nq1

	00	01	11	10	q1q0
00	1		d		
01		1	d		
11			d		
10			d		
rx					

so that $nq1 = \overline{r}.\overline{x}.\overline{q1}.\overline{q0} \; + \; q0.\overline{r}.x$.

The next state variable $nq0$ has the truth-table

nq0

	00	01	11	10	q1q0
00			d	1	
01	1		d		
11			d		
10			d		
rx					

so that $nq0 = \overline{r}.x.\overline{q1}.\overline{q0} \; + \; q1.\overline{r}.\overline{x}$.

To implement these three equations, we can obtain the following simplified set of formulas (we have introduced intermediates u, v, w):

$$\begin{aligned} u &= \overline{q1}.\overline{q0} \\ v &= \overline{r}.x \\ w &= \overline{r}.\overline{x} \\ y &= v.q1 \\ nq1 &= w.u + v.q0 \\ nq0 &= v.u + w.q1 \end{aligned}$$

Implement these using our library of gates to get the logic network in Figure 2.

3 The final timing analysis

- The maximum delay from a circuit input to a flip-flop input is 5 units. r,x to nq
- The maximum delay from launch flip-flop to capture flip-flop (from *clock* to *nq*) is 5 units.

q1 to change in nq1

actually, clock of q's to nq

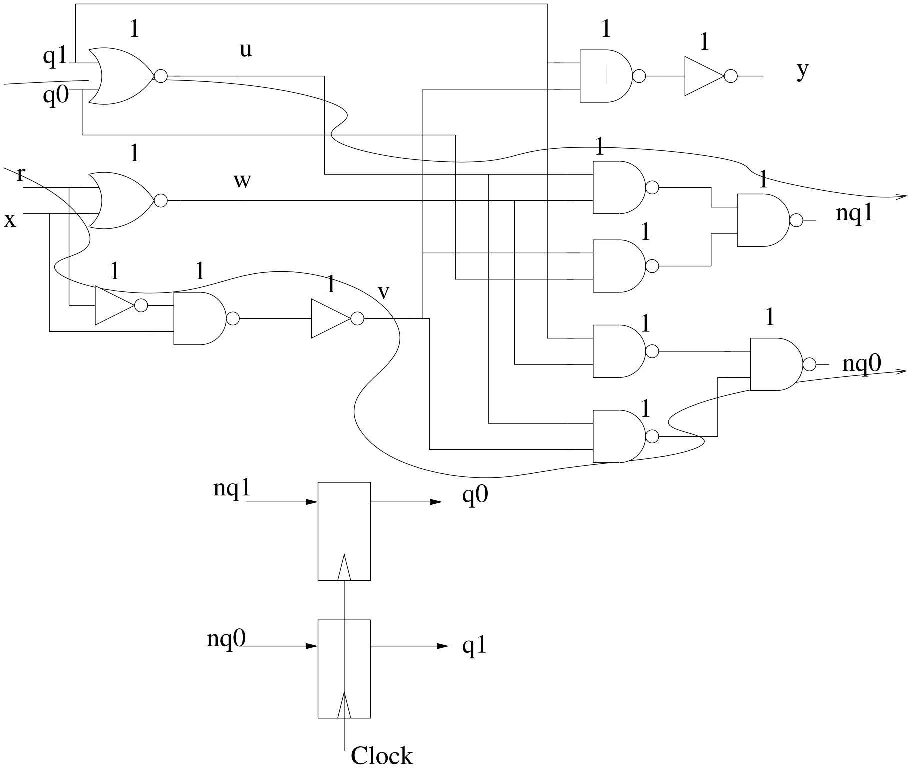

Figure 2: Logic Network

- The maximum delay from clock to the output is 4 units. clock of q's to y
- The maximum delay from circuit input to circuit output is 5 units. r,x to y

If we assume that the clock skew is 0, ~~and the inputs to the circuit change between 2 and 3 units after the rising edge of clock,~~ we observe that the setup constraints are met if the clock period is ≥ 6 units. Further, we observe that the hold-time constraints are also met at any clock period. The circuit operates correctly at a clock period of 6 units.

q's change after 2,
=> nq change btwn 4-5
if r,x change btwn -1 to 0, nq change ends at 5
=>**Tmin =6**

ensuring y can be sampled before inputs change,
y changes 3 after r, 2 after q1 (t=4 < 5) Sample betwn 4 and 5.