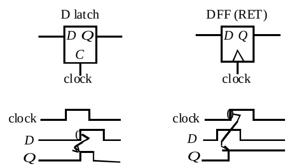
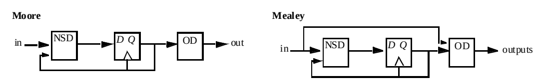


1 Recap

D latch:



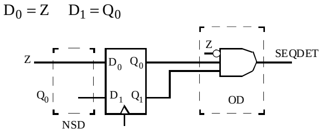
1. Recap (cont.)



Analysis:

$SEQDET = Q_1 Q_0 \cdot \overline{Z}$

$D_0 = Z \quad D_1 = Q_0$

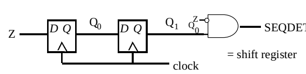


let state $= Q_1 Q_0$

1. Recap (cont.)

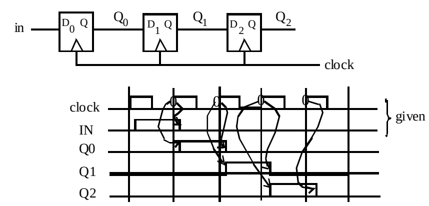
PS Q_1	PS Q_0	IN Z	NS $Q_1 = D_1$	NS $Q_0 = D_0$	SEQDET
0	0	0	0	0	0
	0	1	0	1	0
	1	0	1	0	0
	1	1	1	1	0
1	0	0	0	0	0
	0	1	0	1	0
	1	0	1	0	1
	1	1	1	0	0

Note alternate topology:



2. SHIFT REGISTER

3-bit shift register:



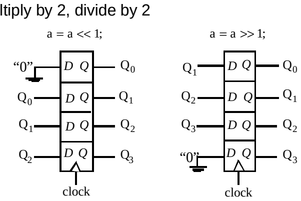
Assumptions:
1 input synchronized
2 clock slow
3 zero initial conditions
4 no skew

If Q_2 is MSB, $Q_2Q_1Q_0$ = 0,1,2,4

Interpretation: multiply by 2, divide by 2, or FIFO

2. Shift Register (cont.)

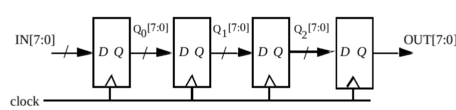
2.1 Multiply by 2, divide by 2



assume initial value already loaded

2. Shift Register (cont.)

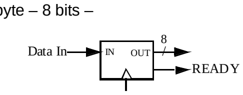
2.2 4-level 8-bit wide FIFO register



2.3 Serial-to-parallel data converter

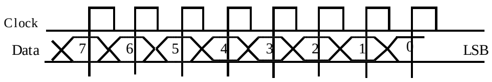
Design a system that outputs 1 byte – 8 bits – after every 8 clocks

MSB first

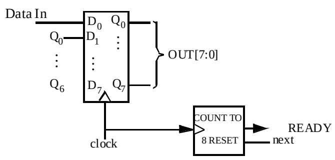


2. Shift Register (cont.)

2.3 Serial-to-parallel data converter (cont.)



Detailed block diagram



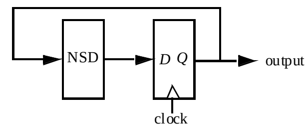
3. COUNTERS

HW equivalent of for (i = 0; i < N; i++) – more general than just numeric sequence

Counter design = FSM design

inputs: reset, starting value, up/down

output: usually just state = Moore/Mealey

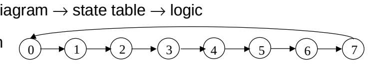


3. Counters (cont.)

3.1 Example: Design count to 8: 0,1,2,3,4,5,6,7,0 …

Method: State diagram → state table → logic

3.1.1 State diagram



⇒ ______ flipflops

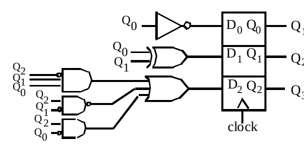
3.1.2 State table

PS	Q_2	Q_1	Q_0	NS	Q_2 (= D_2)	Q_1 (= D_1)	Q_0 (= D_0)
	0	0	0		0	0	1
	0	0	1		0	1	0
	0	1	0		0	1	1
	0	1	1		1	0	0
	1	0	0		1	0	1
	1	0	1		1	1	0
	1	1	0		1	1	1
	1	1	1		0	0	0

3. Counters (cont.)

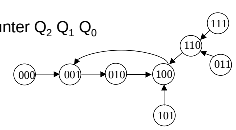
3.1.3 Logic NSD

$$D_0 = \overline{Q}_0$$
$$D_1 = Q_1 \oplus Q_0$$
$$D_2 = \overline{Q}_2 \cdot Q_1 \cdot Q_0 + Q_2 \cdot \overline{Q_1 \cdot Q_0}$$
$$= \overline{Q}_2 \cdot Q_1 + Q_0 + Q_2(\overline{Q}_1 + \overline{Q}_0)$$
$$= Q_2 \oplus (Q_1 \cdot Q_0)$$
$$= Q_2 \oplus (Q_1 \cdot Q_0)$$

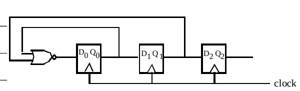


3. Counters (cont.)

3.2 Ring counter Q_2 Q_1 Q_0



PS	Q_2	Q_1	Q_0	NS	Q_2 (= D_2)	Q_1 (= D_1)	Q_0 (= D_0)
	0	0	0		0	0	1
→	0	0	1		0	1	0
→	0	1	0		1	0	0
	0	1	1		1	1	0
→	1	0	0		0	0	1
	1	0	1		0	1	0
	1	1	0		1	0	0
	1	1	1		1	1	0



$$D_0 = \overline{Q_0 + Q_1}$$
$$D_1 = Q_0$$
$$D_2 = Q_1$$

Why? Very fast – minimal NSD