Sequential Circuits

Clock Signals

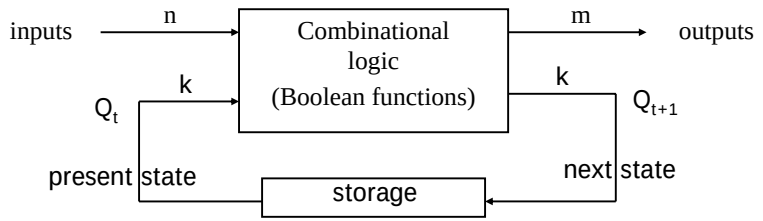
- Clock: a circuit that emits a series of pulses with precise pulse width (how long the pulse lasts) and interval (how long until the next one)
- Number of pulses per second is the clock frequency, commonly between 1 and 2 billion per second (1 and 2 GHz)
- Controlled by crystal oscillator
- Time between edges of 2 consecutive pulses is the clock cycle time
- Clock frequency = 1 / clock cycle time

Metric Prefixes

T	tera	10^{12}	m	milli	10^{-3}
G	giga	10^{9}	μ	micro	10^{-6}
M	mega	10^{6}	n	nano	10^{-9}
K	kilo	10^{3}	p	pico	10^{-12}

Sequential Circuits

- Sequential circuit is a digital logic circuit whose outputs are a function of their current inputs and stored information about previous inputs (i.e., the current state of the circuit).



Latches & Flip-Flops

- In the same way that gates are the building blocks of combinational circuits, *latches* and *flip-flops* are the building blocks of sequential circuits.
- While gates had to be built directly from transistors, latches can be built from gates, and flip-flops can be built from latches.
- Both latches and flip-flops are circuit elements whose output depends not only on the current inputs, but also on previous inputs and outputs.
- The difference between a latch and a flip-flop is that a latch does not have a *clock signal* (level triggered), whereas a flip-flop always does (edge triggered).

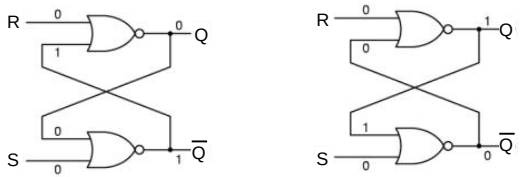
Latches / Flip-Flops

- Exist in one of two states, and in the absence of input, remain in that state
- Can function as a 1-bit memory
- Has two outputs, which are (should be) always complements of each other (generally expressed as Q and $\overline{Q}$).

S-R Latch

- 2 inputs: Set, Reset
- Has feedback so output (Q) not determined by just the 2 inputs
- Has 2 stable states for R = S = 0. 0 or 1 depending on Q.
- S momentarily set to 1, Q = 1
- R momentarily set to 1, Q = 0

2 stable states:



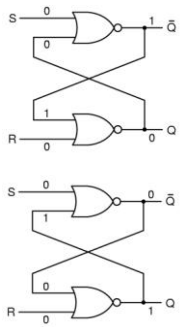
- When S = R = 1, Q = $\overline{Q}$ = 0 (not allowed).
- When R and S become zero at the same time, resulting state becomes undefined (randomly 1 of the 2 stable states).

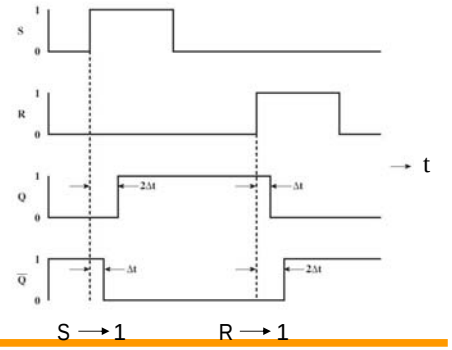
S-R Latch Memory

- Q is the value of the bit
- Setting S=1 (R remains 0) sets the value of Q to 1; state is stable after 2 gate delays even if S is returned to 0.
- Setting R=1 (S remains 0) sets the value of Q to 0; state is stable after 2 gate delays even if R is returned to 0.

Q = 0 state

Q = 1 state





S-R Latch Definition

State Table

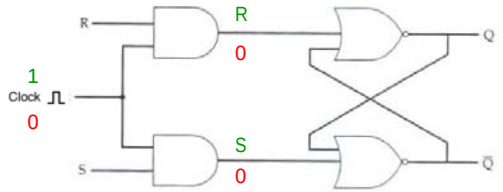
Current Inputs		Current State	Next State
S	R	Q_n	Q_{n+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	-
1	1	1	-

Simplified State Table

S	R	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	-

Clocked S-R Latch (Flip-Flop)

- Events in a computer are typically synchronized to a clock pulse, so that changes occur only when a clock pulse changes (e.g., start & end of pulse = 1).



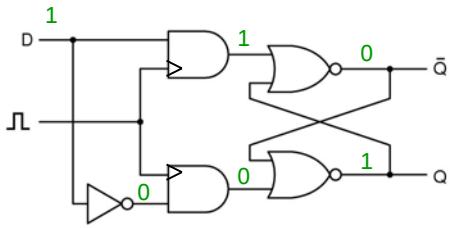
R & S are passed to NOR gate only during the clock pulse.

Timing for Clocked SR Flip-Flop

See Figure A.25 in textbook.

D Flip-Flop

- Like clocked SR Flip-Flop but S = R = 1 not possible.
- Inputs S and R replaced with D and $\overline{D}$ which will never be the same.
- When D is 1, output Q is 1.

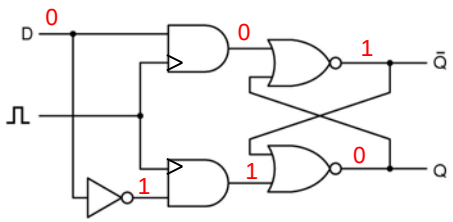


State Table

D	Q_{n+1}
1	1

D Flip-Flop

- Like clocked SR Flip-Flop but S = R = 1 not possible.
- Inputs S and R replaced with D and $\overline{D}$ which will never be the same.
- When D is 1, output Q is 1.
- When D is 0, output Q is 0.



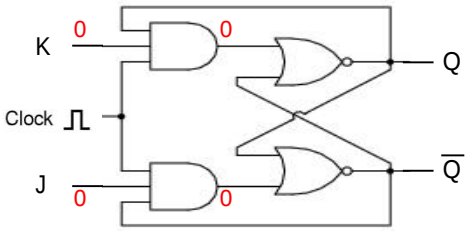
State Table

D	Q_{n+1}
0	0
1	1

J-K Flip-Flop

Sometimes called "Clocked J-K Flip-Flop" or "Clocked J-K Latch"

Similar to S-R Flip-Flop except 11 is a valid input and is a toggle switch between Q and $\overline{Q}$.



State Table

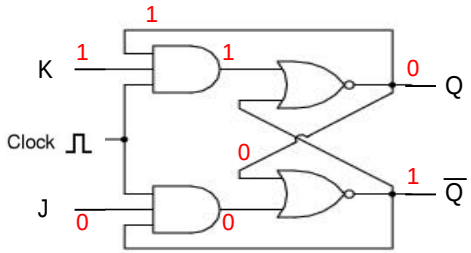
J	K	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	$\overline{Q}_n$

No change

J-K Flip-Flop

Sometimes called "Clocked J-K Flip-Flop" or "Clocked J-K Latch"

Similar to S-R Flip-Flop except 11 is a valid input and is a toggle switch between Q and $\overline{Q}$.



State Table

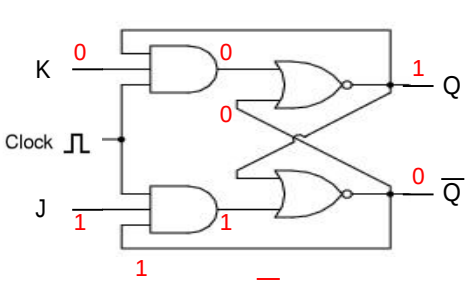
J	K	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	$\overline{Q}_n$

Q set to 0; $\overline{Q}$ set to 1.

J-K Flip-Flop

Sometimes called "Clocked J-K Flip-Flop" or "Clocked J-K Latch"

Similar to S-R Flip-Flop except 11 is a valid input and is a toggle switch between Q and $\overline{Q}$.



State Table

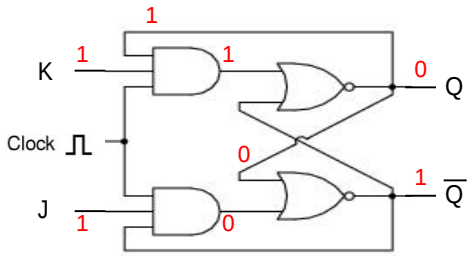
J	K	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	$\overline{Q}_n$

$\overline{Q}$ set to 0; Q set to 1.

J-K Flip-Flop

Sometimes called “Clocked J-K Flip-Flop” or “Clocked J-K Latch”

Similar to S-R Flip-Flop except 11 is a valid input and is a toggle switch between Q and $\overline{Q}$.



State Table

J	K	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	$\overline{Q_n}$

Q set to 0; $\overline{Q}$ set to 1.

Convert D-FF into a JK-FF

Converter
~~JK~~ State Table

J K	Q_n	Q_{n+1}	D
0 0	0	0	0
0 0	1	1	1
0 1	0	0	0
0 1	1	0	0
1 0	0	1	1
1 0	1	1	1
1 1	0	1	1
1 1	1	0	0

D State Table

D	Q_{n+1}
0	0
1	1

$D(J,K,Q_n) = \sum m\,(1, 4, 5, 6)$

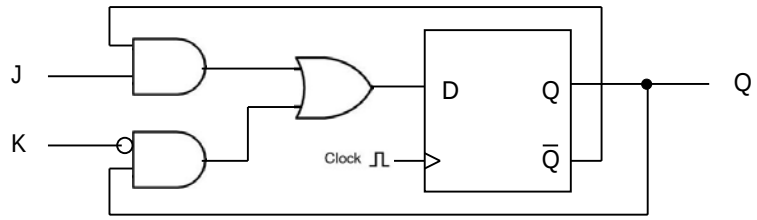
K-map:

J \ K Q	0 0	0 1	1 1	1 0
0		1		
1	1	1		1

$D = J\overline{Q} + \overline{K}Q$

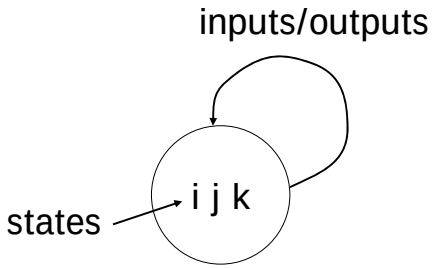
D-FF to JK-FF Circuit

$D = J\overline{Q} + \overline{K}Q$

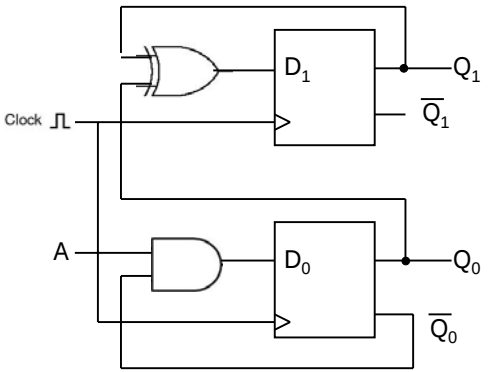


Sequential Circuit Analysis

- Goal is to obtain description of time sequence of signals:
  - Derive FF input equations
  - Fill-in State Table
  - Map to State Diagram



Analysis Example 1 [1]



$D_1 = Q_0 \oplus Q_1$

$\Downarrow$

$Q_1(t+1) = Q_0(t) \oplus Q_1(t)$

$D_0 = A\,\overline{Q_0}$

$\Downarrow$

$Q_0(t+1) = A\,\overline{Q_0}(t)$

Analysis Example 1 [2]

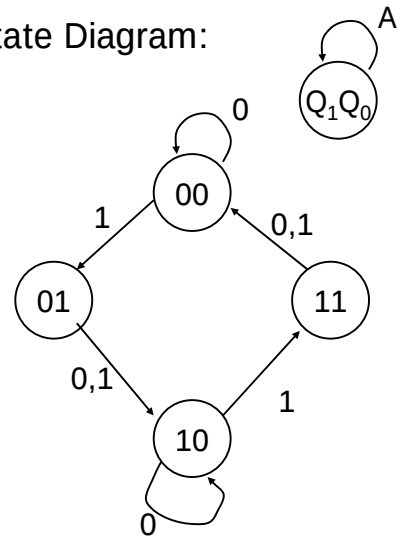
$Q_1(t+1) = Q_0(t) \oplus Q_1(t)$

$Q_0(t+1) = A\,\overline{Q_0}(t)$

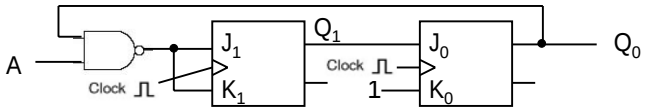
State Table:

	t		t+1	
A	Q_1	Q_0	Q_1	Q_0
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	1
1	1	1	0	0

State Diagram:



Analysis Example 2 [1]



$J_1 = \overline{A\,Q_0}$ $K_1 = J_1$ $J_0 = Q_1$ $K_0 = 1$

State Table:

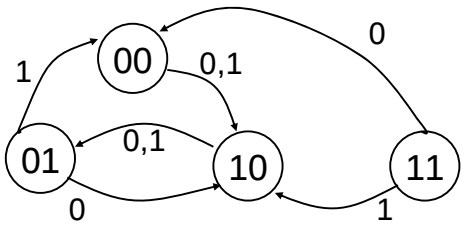
	t		FF input equations				t+1	
A	Q_1	Q_0	J_1	K_1	J_0	K_0	Q_1	Q_0
0	0	0	1	1	0	1	1	0
0	0	1	1	1	0	1	1	0
0	1	0	1	1	1	1	0	1
0	1	1	1	1	1	1	0	0
1	0	0	1	1	0	1	1	0
1	0	1	0	0	0	1	0	0
1	1	0	1	1	1	1	0	1
1	1	1	0	0	1	1	1	0

Analysis Example 2 [2]

State Table:

	t		FF input equations				t+1	
A	Q_1	Q_0	J_1	K_1	J_0	K_0	Q_1	Q_0
0	0	0	1	1	0	1	1	0
0	0	1	1	1	0	1	1	0
0	1	0	1	1	1	1	0	1
0	1	1	1	1	1	1	0	0
1	0	0	1	1	0	1	1	0
1	0	1	0	0	0	1	0	0
1	1	0	1	1	1	1	0	1
1	1	1	0	0	1	1	1	0

State Diagram:



Sequential Circuit Design

- Derive State Table and/or State Diagram
- Number of FF = $\lceil \log_2 \#\ \text{states} \rceil$
  - Choose FF type
- Derive FF input circuitry using K-maps, etc. – same for all output functions

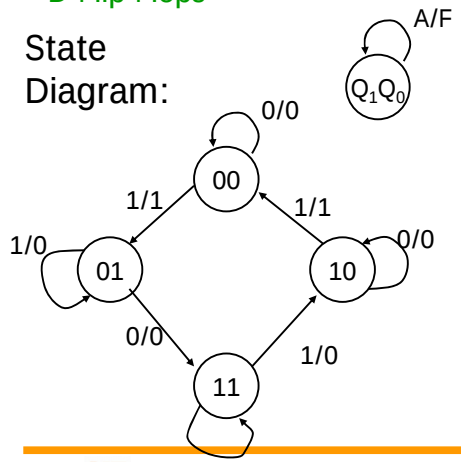
Design Example [1]

4 States ⇒ 2 FFs

1 input, 1 output

D Flip Flops

State Diagram:

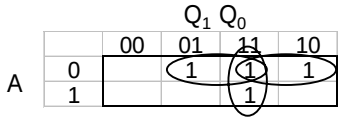


State Table:

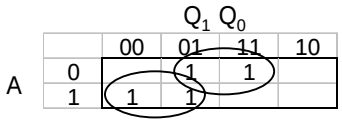
	t		t+1		
A	Q_1	Q_0	Q_1	Q_0	F
0	0	0	0	0	0
0	0	1	1	1	0
0	1	0	1	0	0
0	1	1	1	1	0
1	0	0	0	1	1
1	0	1	0	1	0
1	1	0	0	0	1
1	1	1	1	0	0

Design Example [2]

$Q_1(t+1)$



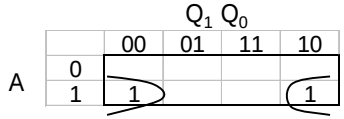
$Q_0(t+1)$



$Q_1(t+1) = \bar{A}Q_1 + \bar{A}Q_0 + Q_1Q_0$

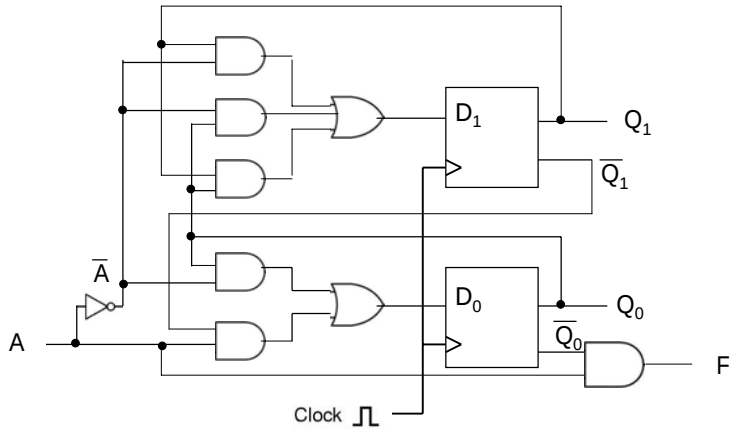
$Q_0(t+1) = \bar{A}Q_0 + A\bar{Q}_1$

F



$F = A\bar{Q}_0$

Design Example [3]



Design Example [4]

4 States ⇒ 2 FFs

1 input, 1 output

JK Flip Flops

State Table:

	t		t+1						
A	Q_1	Q_0	Q_1	Q_0	F	J_1	K_1	J_0	K_0
0	0	0	0	0	0	0	x	0	x
0	0	1	1	1	0	1	x	x	0
0	1	0	1	0	0	x	0	0	x
0	1	1	1	1	0	x	0	x	0
1	0	0	0	1	1	0	x	1	x
1	0	1	0	1	0	0	x	x	0
1	1	0	0	0	1	x	1	0	x
1	1	1	1	0	0	x	0	x	1

Design Example [5]

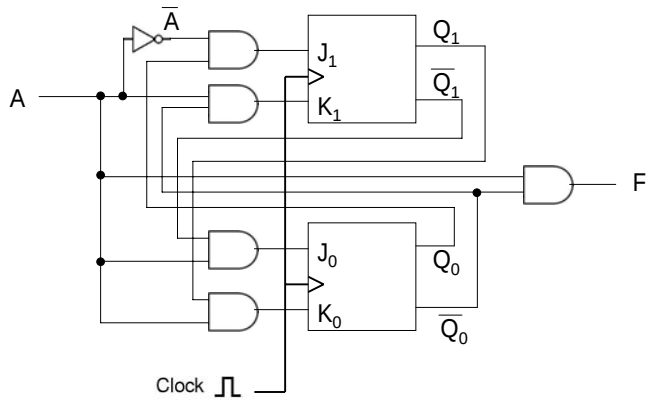
Using K-Maps:

$J_1 = \bar{A}Q_0$ $\quad K_1 = A\bar{Q}_0$

$J_0 = A\bar{Q}_1$ $\quad K_0 = AQ_1$

$F = A\bar{Q}_0$

Design Example [6]



Sequential Circuit Design Flexibility

- If not all states are specified by State Table / Diagram, then use “don’t care” states (K-Map) or tie to specific states (“error recovery”).

Sequence recognizers

- A sequence recognizer is a special kind of sequential circuit that looks for a special bit pattern in some input.
- The recognizer circuit has only one input, X.
  - One bit of input is supplied on every clock cycle. For example, it would take 20 cycles to scan a 20-bit input.
  - This is an easy way to permit arbitrarily long input sequences.
- There is one output, Z, which is 1 when the desired pattern is found.
- Our example will detect the bit pattern "1001":

Inputs: 11100110100100110…
Outputs: 00000100000100100…

Here, one input and one output bit appear every clock cycle.

- This requires a sequential circuit because the circuit has to "remember" the inputs from previous clock cycles, in order to determine whether or not a match was found.

Sequential circuit design procedure

Step 1:
Make a state table based on the problem statement. The table should show the present states, inputs, next states and outputs. (It may be easier to find a state diagram first, and then convert that to a table.)

Step 2:
Assign binary codes to the states in the state table, if you haven't already. If you have n states, your binary codes will have at least $\lceil \log_2 n \rceil$ digits, and your circuit will have at least $\lceil \log_2 n \rceil$ flip-flops.

Step 3:
For each flip-flop and each row of your state table, find the flip-flop input values that are needed to generate the next state from the present state.

Step 4:
Find simplified equations for the flip-flop inputs and the outputs.

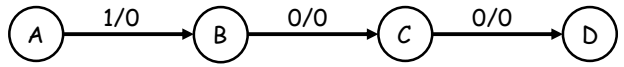
Step 5:
Build the circuit!

Step 1: Making a state table

- The first thing you have to figure out is precisely how the use of state will help you solve the given problem.
  - Make a state table based on the problem statement. The table should show the present states, inputs, next states and outputs.
  - Sometimes it is easier to first find a state diagram and then convert that to a table.
- This is usually the most difficult step. Once you have the state table, the rest of the design procedure is the same for all sequential circuits.

A basic state diagram

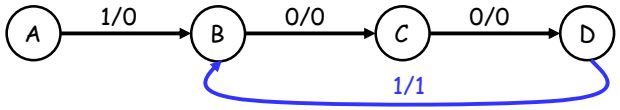
- What state do we need for the sequence recognizer?
  - We have to "remember" inputs from previous clock cycles.
  - For example, if the previous three inputs were 100 and the current input is 1, then the output should be 1.
  - In general, we will have to remember occurrences of parts of the desired pattern—in this case, 1, 10, and 100.
- We'll start with a basic state diagram:



State	Meaning
A	None of the desired pattern (1001) has been input yet.
B	We've already seen the first bit (1) of the desired pattern.
C	We've already seen the first two bits (10) of the desired pattern.
D	We've already seen the first three bits (100) of the desired pattern.

Overlapping occurrences of the pattern

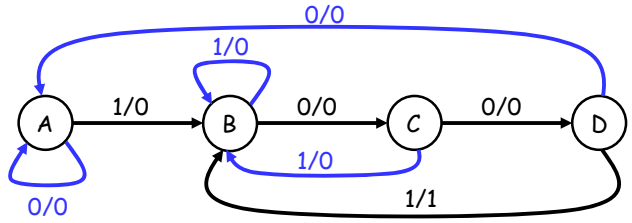
- What happens if we're in state D (the last three inputs were 100), and the current input is 1?
  - The output should be a 1, because we've found the desired pattern.
  - But this last 1 could also be the start of another occurrence of the pattern! For example, 1001001 contains *two* occurrences of 1001.
  - To detect overlapping occurrences of the pattern, the next state should be B.



State	Meaning
A	None of the desired pattern (1001) has been input yet.
B	We've already seen the first bit (1) of the desired pattern.
C	We've already seen the first two bits (10) of the desired pattern.
D	We've already seen the first three bits (100) of the desired pattern.

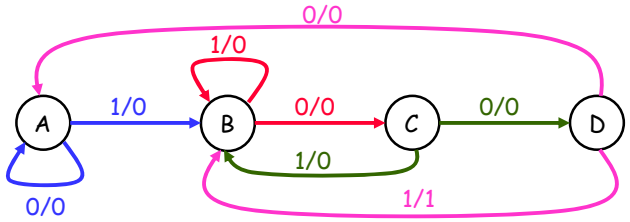
Filling in the other arrows

- Remember that we need *two* outgoing arrows for each node, to account for the possibilities of X=0 and X=1.
- The remaining arrows we need are shown in blue. They also allow for the correct detection of overlapping occurrences of 1001.

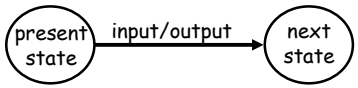


State	Meaning
A	None of the desired pattern (1001) has been input yet.
B	We've already seen the first bit (1) of the desired pattern.
C	We've already seen the first two bits (10) of the desired pattern.
D	We've already seen the first three bits (100) of the desired pattern.

Finally, making the state table



Remember how the state diagram arrows correspond to rows of the state table:



Present State	Input	Next State	Output
A	0	A	0
A	1	B	0
B	0	C	0
B	1	B	0
C	0	D	0
C	1	B	0
D	0	A	0
D	1	B	1

Step 2: Assigning binary codes to states

- We have four states ABCD, so we need at least two flip-flops Q_1Q_0.
- The easiest thing to do is represent state A with Q_1Q_0 = 00, B with 01, C with 10, and D with 11.
  - The state assignment can have a big impact on circuit complexity, but we won't worry about that for this example.

Present State	Input	Next State	Output
A	0	A	0
A	1	B	0
B	0	C	0
B	1	B	0
C	0	D	0
C	1	B	0
D	0	A	0
D	1	B	1

Present State Q_1	Present State Q_0	Input X	Next State Q_1	Next State Q_0	Output Z
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	1	0	0
0	1	1	0	1	0
1	0	0	1	1	0
1	0	1	0	1	0
1	1	0	0	0	0
1	1	1	0	1	1

Building the circuit with D flip-flops

- We have the state table and state assignments, we now must find the flip-flop input values.
- D flip-flops have only one input, so our table only needs two columns for D_1 and D_0.

Present State Q_1	Present State Q_0	Input X	Next State Q_1	Next State Q_0	Flip-flop inputs D_1	Flip-flop inputs D_0	Output Z
0	0	0	0	0			0
0	0	1	0	1			0
0	1	0	1	0			0
0	1	1	0	1			0
1	0	0	1	1			0
1	0	1	0	1			0
1	1	0	0	0			0
1	1	1	0	1			1

D flip-flop input values (Step 3)

- The D excitation table is pretty boring; set the D input to whatever the next state should be.
- You don't even need to show separate columns for D_1 and D_0; you can just use the Next State columns.

Q(t)	Q(t+1)	D	Operation
0	0	0	Reset
0	1	1	Set
1	0	0	Reset
1	1	1	Set

Present State Q_1	Present State Q_0	Input X	Next State Q_1	Next State Q_0	Flip flop inputs D_1	Flip flop inputs D_0	Output Z
0	0	0	0	0	0	0	0
0	0	1	0	1	0	1	0
0	1	0	1	0	1	0	0
0	1	1	0	1	0	1	0
1	0	0	1	1	1	1	0
1	0	1	0	1	0	1	0
1	1	0	0	0	0	0	0
1	1	1	0	1	0	1	1

Finding equations (Step 4)

- You can do K-maps again, to find:

$$D_1 = Q_1\bar{Q}_0\bar{X} + \bar{Q}_1Q_0\bar{X}$$

$$D_0 = X + Q_1\bar{Q}_0$$

$$Z = Q_1Q_0X$$

Present State Q_1	Present State Q_0	Input X	Next State Q_1	Next State Q_0	Flip flop inputs D_1	Flip flop inputs D_0	Output Z
0	0	0	0	0	0	0	0
0	0	1	0	1	0	1	0
0	1	0	1	0	1	0	0
0	1	1	0	1	0	1	0
1	0	0	1	1	1	1	0
1	0	1	0	1	0	1	0
1	1	0	0	0	0	0	0
1	1	1	0	1	0	1	1

Building the circuit (Step 5)