ME 311: Microprocessors and Automatic Control

Application of RS, D-FlipFlops
Counters: finite state machines

P.S. Gandhi
Mechanical Engineering
IIT Bombay

Acknowledgment: Aniruddha M Pol, Mtech 10

PRASANNA S GANDHI
gandhi@me.iitb.ac.in

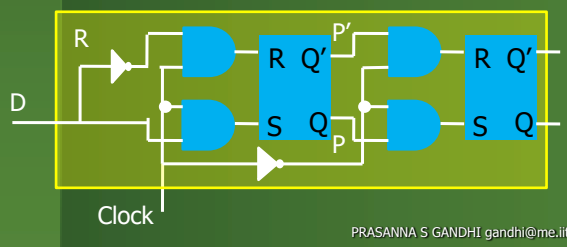
Life Skill

Understanding mind!

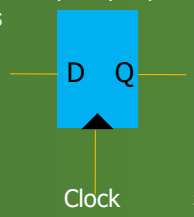
- Story
- Laws work at mind level are different than those work at body level

Recap

- D-negative edge-triggered flip-flop
- Many other variants of flip-flop are available. Ex JK flip-flop
- Cascading flipflops



The entire block can be compactly represented as

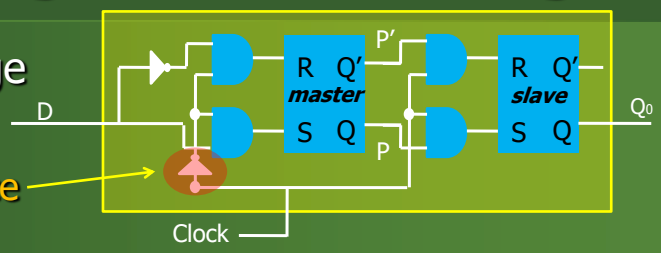


D-negative edge-triggered flip-flop

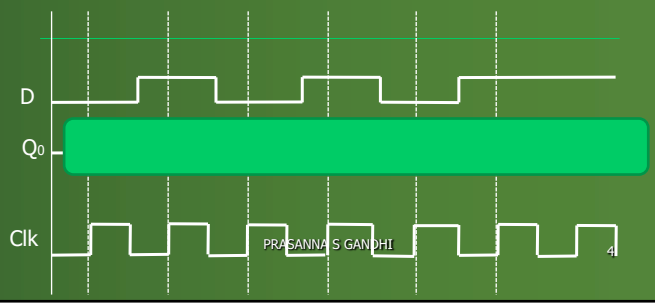
D-Flipflop Timing Diagram and Working

- Positive edge Triggered

Notice difference



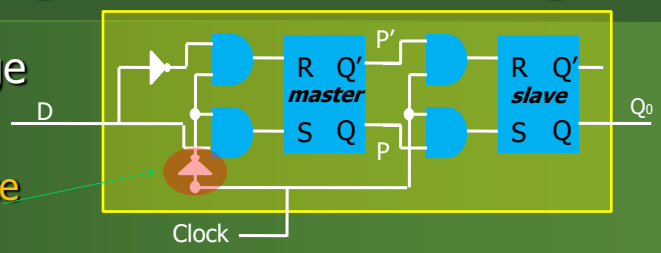
Given input D, can you complete timing diagram for output Q_0



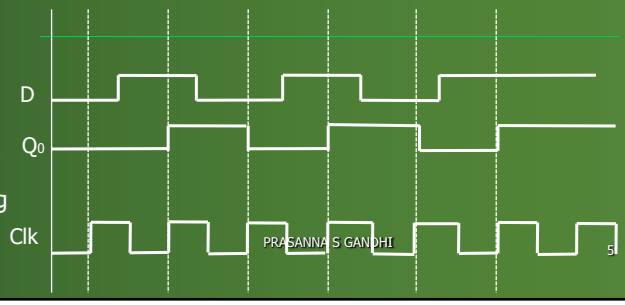
D-Flipflop Timing Diagram and Working

- Positive edge Triggered

Notice difference



Given input D, can you complete timing diagram for output Q_0

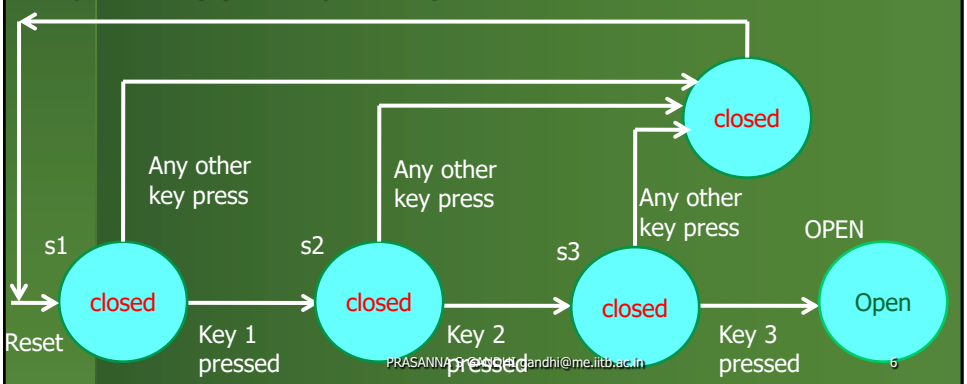


The flipflop acts as Synchronizer synchronizing Changes to occur only at Clock cycle!!!

5

Sequential logic design: digital lock

- Recall LOGIC DEVELOPMENT: Lock open sequence is key1 followed by key2 followed by key3 (These keys remain pressed (1) once pressed)



6

Sequential logic design: digital lock

Key1	Key2	Key3	STATE	Next STATE		LOCK
0	-	-	S1	S1		Closed
1	0	0	S1	S2		Closed
1	1	0	S2	S3		Closed
1	1	1	S3	OPEN		Open

- For all other combinations of keys and whatever be the state the next state should be s1 then lock will remain closed (this part of the logic is NOT presented above)
- Memory block will make previous state available for given clock cycle to decide the next state AND change the state to the next state every clock cycle
- Now states (4) can be represented further as binary number to get the logic circuit with a memory block

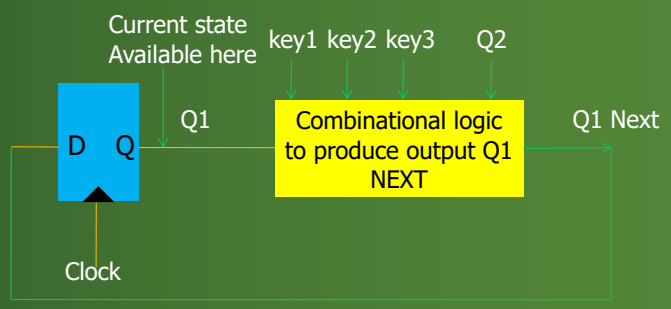
Sequential logic design: digital lock

Key1	Key2	Key3	STATE Q1 Q2	Next STATE Q1 Q2		LOCK
0	-	-	0 0	0 0		Closed
1	0	0	0 0	0 1		Closed
1	1	0	0 1	1 0		Closed
1	1	1	1 0	1 1		Open

- Once everything is binary now a combinational logic process can be used to get the expressions and circuit block represented by later.
- Think how D flip flop can be used to generate the next state given all keys and current state Q1 Q2??

Sequential logic design: digital lock

- D flip flop can be used in the following way say for Q1



- Notice the way of implementation using fundamental understanding (though the formula for D-FF says $Q^{+} = D$

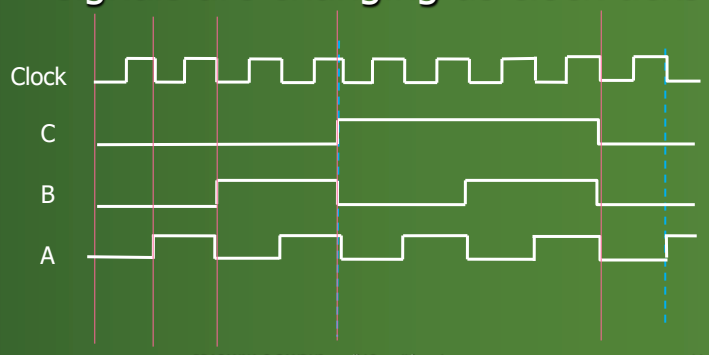
3 bit up counter: Design Procedure

- Problem: We would like to design digital circuit that on every clock tick, can count numbers 0, 1, 2, so on.. Say upto 8 each one would be with corresponding binary representation
- Q: how we can we develop such counter? Multiple ideas possible
  - Use DFF and feed its output in clock of next DFF with or without modifications using other gates
  - Formal way: Develop state diagram about what you want to do and represent states with binary numbers

3Bit up counter

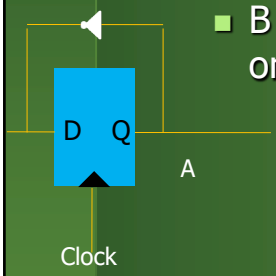
Present State			
	C	B	A
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

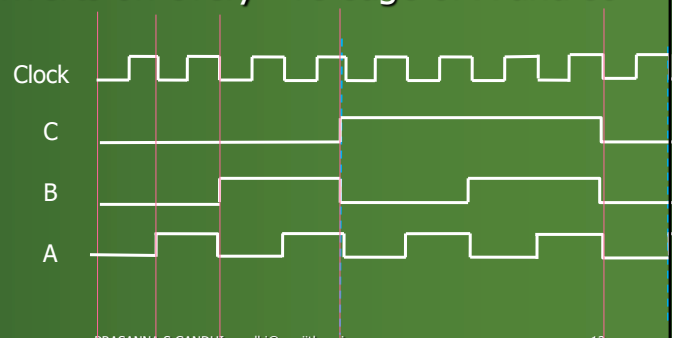
- Close observation: see how the signals are changing as clock ticks



3Bit up counter

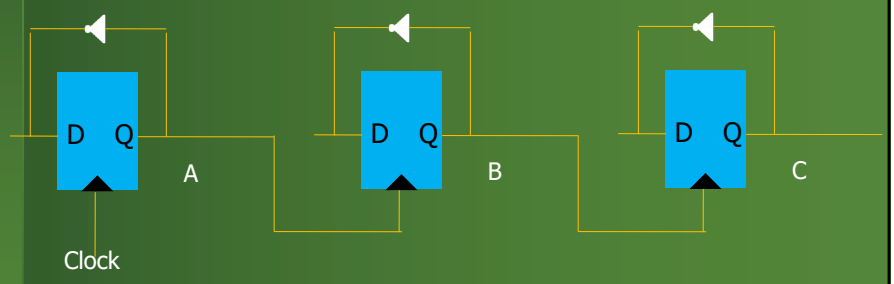
- A for example inverts on every –ve clock edge. So use Dff with inverter as shown
- B inverts on every –ve edge of A and so on



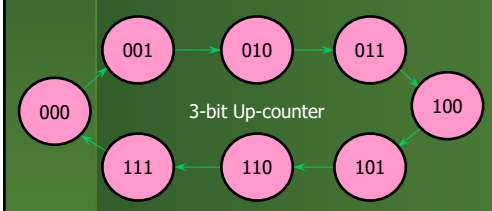


3Bit up counter

- Thus use A as a clock signal for Dff representing B as shown and so on.

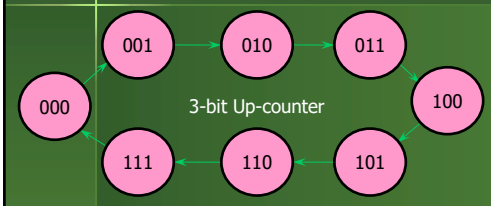


3 bit up counter: Formal way to design



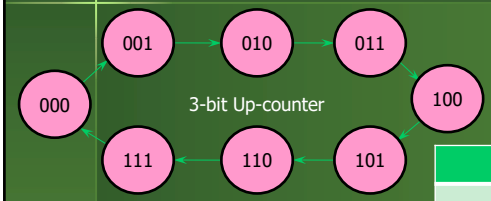
- Get the the state diagram:
  - there are no external condition for transition from one state to other
  - No external inputs
- Can you next develop truth table with previous and next states?

3 bit up counter: Design Procedure



- Represent number by CBA
- What would be entries in truth table
  - Inputs: Current values of C B A
  - Outputs: Next values C^+, B^+, A^+ respectively
- Truth table?

3 bit up counter: Design Procedure

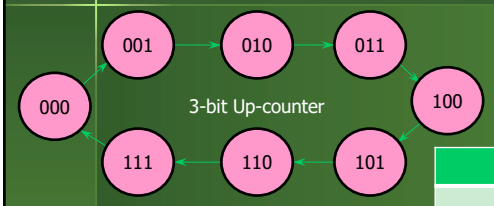


- Truth table:

	Present State			Next State			
	C	B	A	C+	B+	A+	
0	0	0	0				1
1	0	0	1				2
2	0	1	0				3
3	0	1	1				4
4	1	0	0				5
5	1	0	1				6
6	1	1	0				7
7	1	1	1				8

Q: Once you lay down various possible values of inputs, how will you generate Outputs?

3 bit up counter: Design Procedure



- Truth table:

	Present State			Next State			
	C	B	A	C+	B+	A+	
0	0	0	0	0	0	1	1
1	0	0	1	0	1	0	2
2	0	1	0	0	1	1	3
3	0	1	1	1	0	0	4
4	1	0	0	1	0	1	5
5	1	0	1	1	1	0	6
6	1	1	0	1	1	1	7
7	1	1	1	0	0	0	8

PRASAN

17

3 bit up counter

Recall Template for K map ➔

C \ AB	00	01	11	10
0	0	2	6	4
1	1	3	7	5

A, B, C

C+

A \ CB	00	01	11	10
0	0	0	1	1
1	0	1	0	1

B+

A \ CB	00	01	11	10
0	0	1	1	0
1	1	0	0	1

A+

A \ CB	00	01	11	10
0	1	1	1	1
1	0	0	0	0

- K map: Develop expressions

	Present State			Next State			
	C	B	A	C+	B+	A+	
0	0	0	0	0	0	1	1
1	0	0	1	0	1	0	2
2	0	1	0	0	1	1	3
3	0	1	1	1	0	0	4
4	1	0	0	1	0	1	5
5	1	0	1	1	1	0	6
6	1	1	0	1	1	1	7
7	1	1	1	0	0	0	8

PRASANNA S

18

3 bit up counter

C+

A \ CB	00	01	11	10
0	0	0	1	1
1	0	1	0	1

- K map: Develop expressions

$$C^{+} = CA' + CB' + C'BA$$
$$= C(A' + B') + C'(AB)$$
$$= C(AB)' + C'(AB)$$
$$\boxed{C^{+} = C \oplus (AB)}$$

3 bit up counter

B+

A \ CB	00	01	11	10
0	0	1	1	0
1	1	0	0	1

- K map: Develop expressions

$$B^{+} = BA' + B'A$$
$$= A \oplus B$$

3 bit up counter

- K map: Develop expressions

$$A^{+} = A'$$

$$= A'.1 + A.0$$

$$= A \oplus 1$$

A+

A \ CB	00	01	11	10
0	1	1	1	1
1	0	0	0	0

C

B

Converted to have Expression with XOR Gate only!! Since other expressions Had XOR!!

3 bit up counter

- With all expressions available as below can you now develop circuit for implementation?

$$C^{+} = C \oplus (AB)$$

$$B^{+} = A \oplus B$$

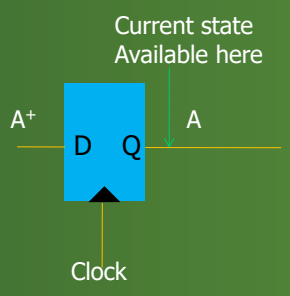
$$A^{+} = A' = A \oplus 1$$

You can put Inverter here As well (its same thing)

In market usually same XOR chip will have multiple XOR gates. Thus the implementation may reduce number of chips on PCB

3 bit up counter

- Recall the way to implement previous and current state using D Flip Flop



23

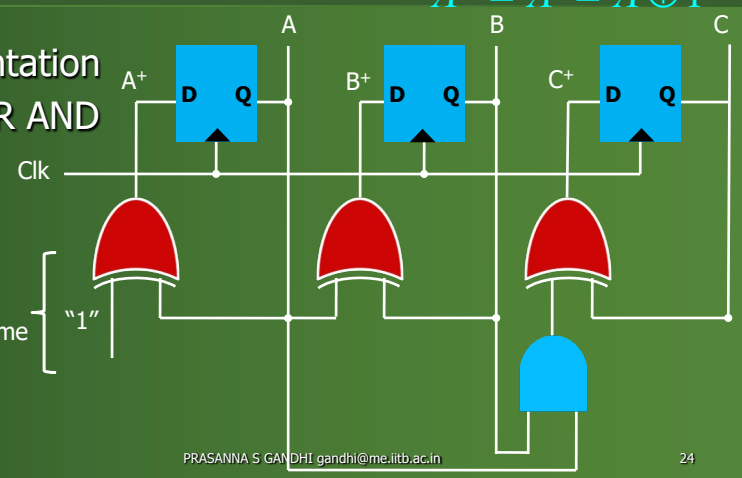
3 bit up counter

$$C^+ = C \oplus (AB)$$

$$B^+ = A \oplus B$$

$$A^+ = A' = A \oplus 1$$

- Circuit: implementation using XOR AND



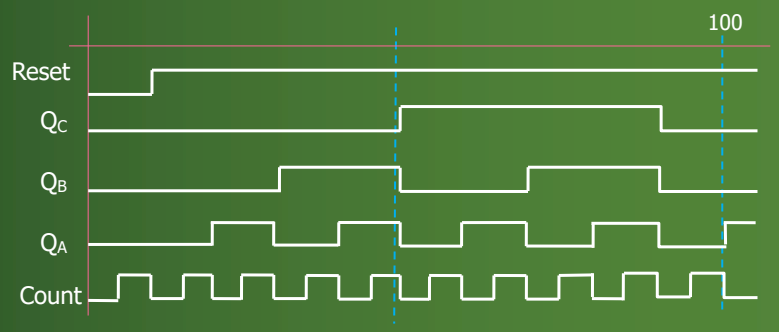
You can put Inverter here As well (its same thing)

In market usually same XOR chip will have multiple XOR gates. Thus the implementation may reduce number of chips on PCB

24

3 bit up counter

- Timing diagram



Counter

DESIGN PROCEDURE: Can be generalized to
Other finite state machines

- STEP 1: written information: understanding ➔ draw state transition diagram
- STEP 2: Derive state transition table tabulating the present state and corresponding next states
- STEP 3: Express each next state bit as a combinational logic function of the current state bits
- STEP 4: Implement corresponding logic circuit

THANK YOU