Unit 12

Registers and Counters

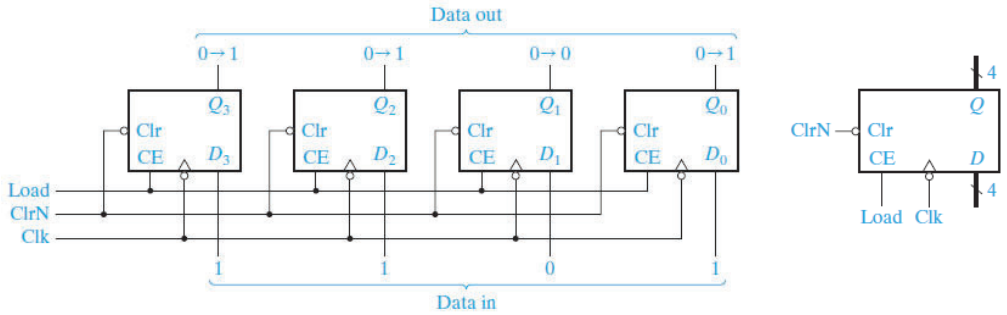
Logic Circuits (Spring 2022)

Registers

- *Register*
  - A group of flip-flops with a common clock input
  - Commonly used to store and shift binary data
- *Shift register*
  - A bit is shifted out one end, and it may be lost or shifted back in the other end
  - The serial input is loaded into the first or the last flip-flop
- *Accumulator*
- (*Binary*) *Counter*
  - A circuit that cycles through a fixed sequence of states
  - Usually constructed from two or more flip-flops which change states in a prescribed sequence when input pulses are received

4-Bit D Flip-Flop Register

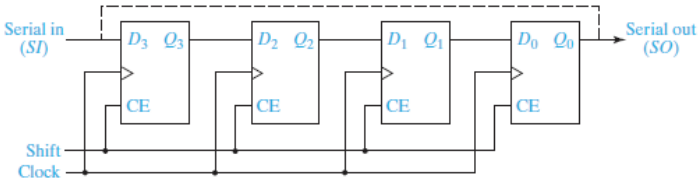
- 4-bit register
  - Each flip-flop stores one bit of information
  - 4-bit register stores 4 bits of information



- Signals and operations
  - When asynchronous clear signal(ClrN) = 0, the Q output will become 0
  - When Load = 1, the Clock Enable(CE) becomes 1 and thus the data applied to D inputs will be loaded in the flip-flops on falling edge
  - When Load = 0, the clock is disabled and the register holds its data

Shift Register

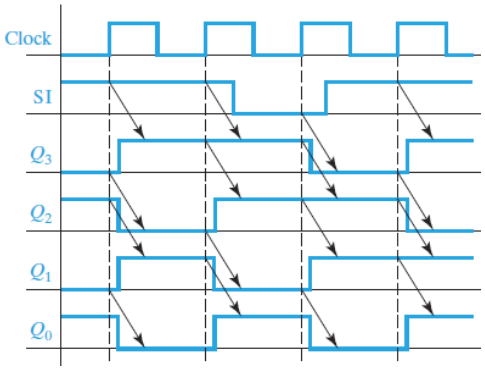
- *Shift register*
  - A register to store binary data
  - The data stored in a shift register can be shifted to the left or right when a shift signal is applied
  - Bits shifted out one end may be lost or shifted back in the other end



- Signals
  - Shifting occurs on the rising clock edge when Shift = 1
  - No shifting occurs and the data in unchanged when Shift = 0

Shift Register

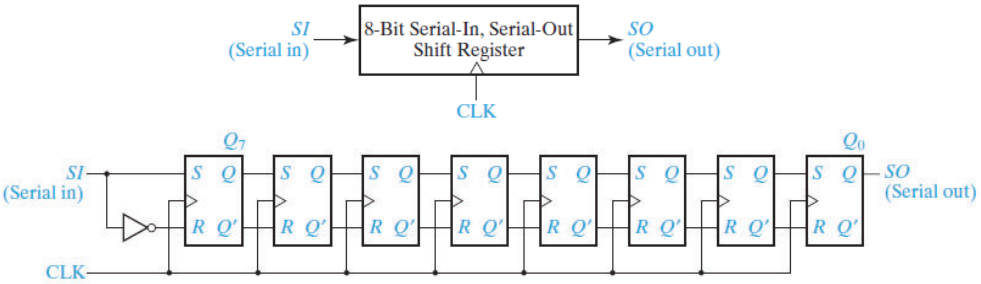
- Timing diagram



  - Serial input(SI) is loaded into the first flip-flop(Q_3)
  - The output of the first flip-flop(Q_3) is loaded into the second flip-flop
  - … …

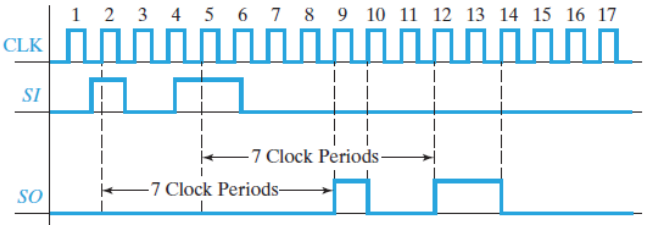
(Serial-In, Serial-Out) Shift Register

- 8-bit *serial-in, serial-out* shift registers
  - Serial in: data is shifted into the first flip-flop one bit at a time,
  - Serial out: data can only be read out of the last flip-flop and the outputs from the other flip-flops are not connected to terminals
  - Cannot load data in parallel



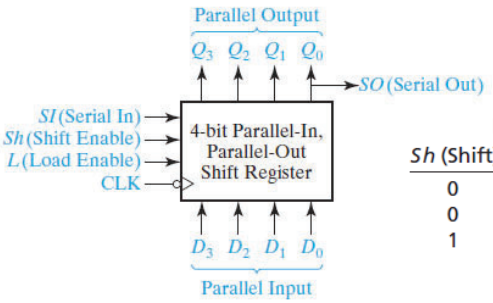
(Serial-In, Serial-Out) Shift Register

- Timing diagram

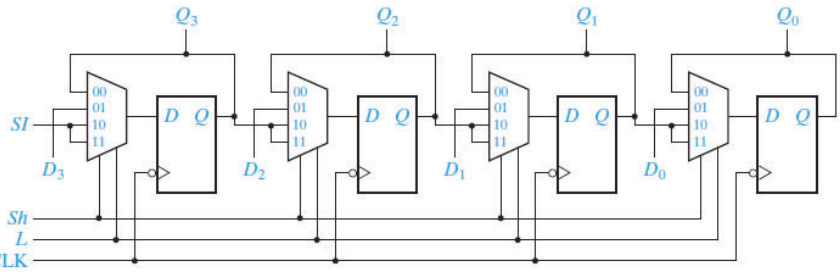


Universal Shift Register

- 4-bit *parallel-in, parallel-out* shift registers

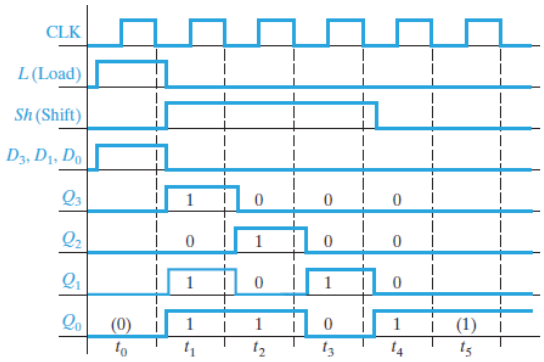


Inputs		Next State				
Sh (Shift)	*L* (Load)	Q_3^+	Q_2^+	Q_1^+	Q_0^+	Action
0	0	Q_3	Q_2	Q_1	Q_0	No change
0	1	D_3	D_2	D_1	D_0	Load
1	X	*SI*	Q_3	Q_2	Q_1	Right shift



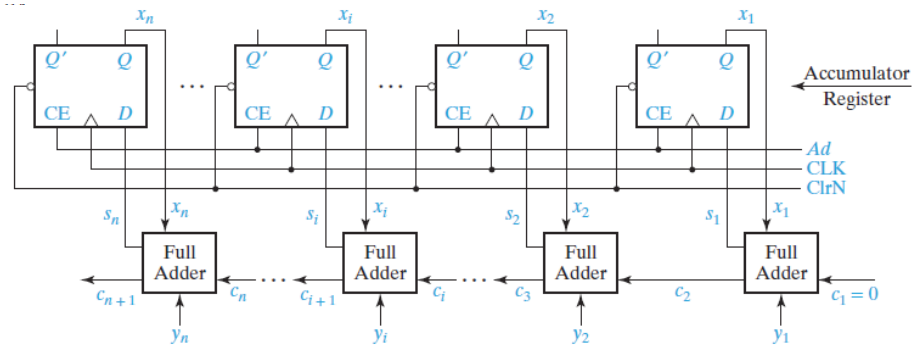
Universal Shift Register

- Timing diagram



Parallel Adder with Accumulator

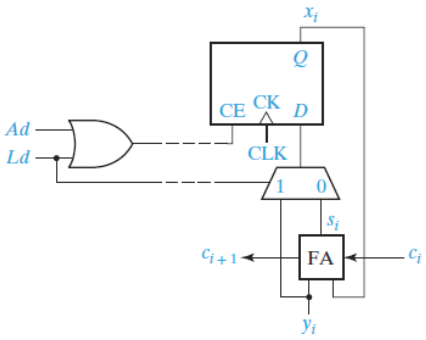
- *Accumulator*
  - Register in which intermediate arithmetic and logic results are stored
  - Example: a number stored in an accumulator is added to a second number in another register



- *n*-bit accumulator can be combined with *n* full adders
  - $s_i \leftarrow x_i + y_i + c_i$

Parallel Adder with Accumulator

- Typical adder cell with multiplexer



- Two operations
  - Accumulator could be loaded with the given number (when $Ld = 1$)
  - Accumulator could store the sum with the given number (when $Ad = 1$, and $Ld = 0$)

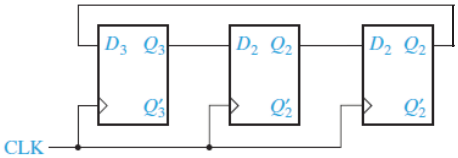
Binary Counter

- *Synchronous counters*
  - The operation of the flip-flops is synchronized by a common clock pulse
  - When several flip-flops must change state, the state changes occur simultaneously.

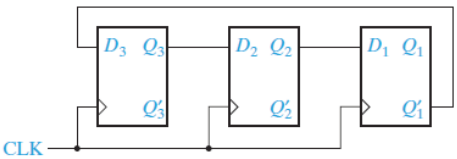
- *Ripple counters*
  - The state change of one flip-flop triggers another flip-flop

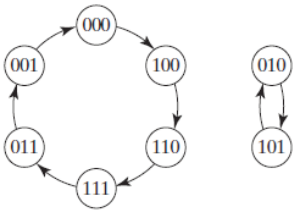
Johnson Counters

- *Ring counter*
  - A shift register with feedback



- *Johnson counter*
  - A shift register with inverted feedback
  - Also called as a *twisted ring counter*





Binary (Up) Counter with D Flip-flops

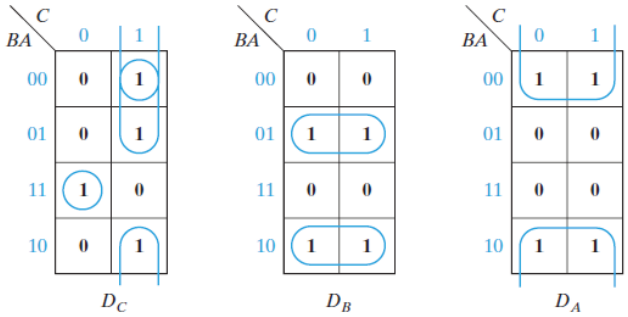
- Transition of 3-bit counter
  - 000 → 001 → 010 → 011 → 100 → 101 → 110 → 111 → 000

- Transition table (Next state table)

Present State			Next State		
C	B	A	C^+	B^+	A^+
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

Binary (Up) Counter with D Flip-flops

- Karnaugh maps for D_C, D_B, and D_A

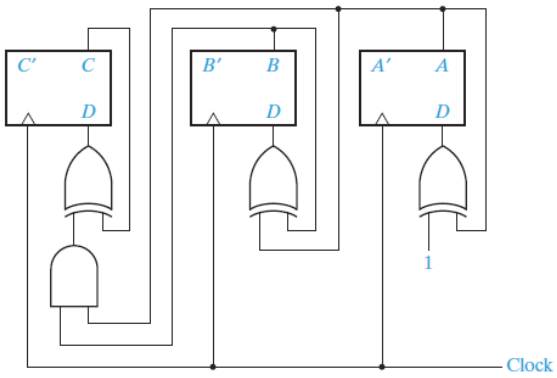


- Flip-flop input equations

$$D_A = A^+ = A'$$
$$D_B = B^+ = BA' + B'A = B \oplus A$$
$$D_C = C^+ = C'BA + CB' + CA' = C'BA + C(BA)' = C \oplus BA$$

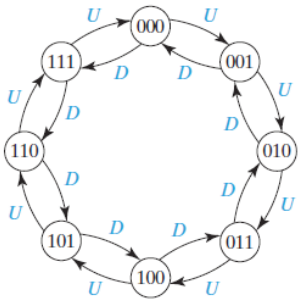
Binary (Up) Counter with D Flip-flops

- Logic diagram



Binary Up-Down Counter

- Transition graph and table



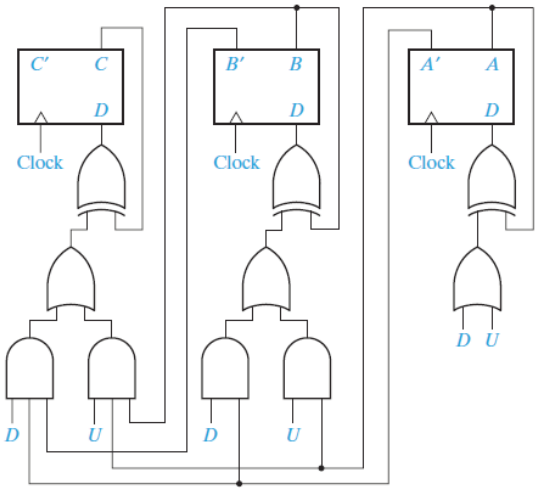
	$C^+B^+A^+$	
CBA	U	D
000	001	111
001	010	000
010	011	001
011	100	010
100	101	011
101	110	100
110	111	101
111	000	110

- Flip-flop input equations

$$D_A = A^+ = A \oplus (U + D)$$
$$D_B = B^+ = B \oplus (UA + DA')$$
$$D_C = C^+ = C \oplus (UBA + DB'A')$$

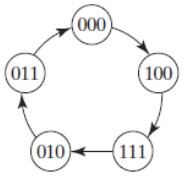
Binary Up-Down Counter

- Logic diagram



Binary Counter for Other Sequences

- An example counter
  - 000 → 100 → 111 → 010 → 011
- Transition graph and table

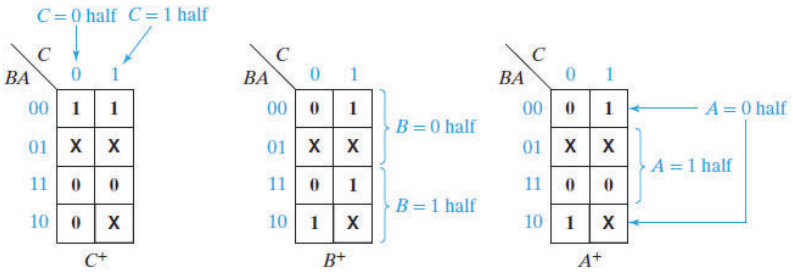


C	B	A	C^+	B^+	A^+
0	0	0	1	0	0
0	0	1	–	–	–
0	1	0	0	1	1
0	1	1	0	0	0
1	0	0	1	1	1
1	0	1	–	–	–
1	1	0	–	–	–
1	1	1	0	1	0

  - Incompletely specified transition table

Binary Counter for Other Sequences

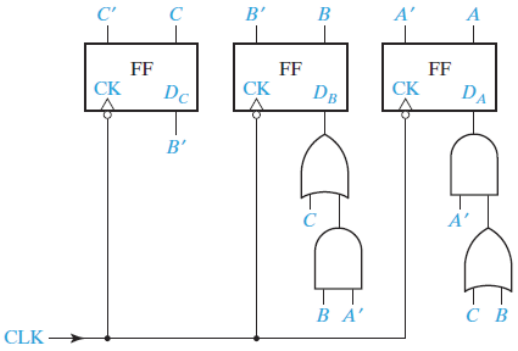
- Karnaugh maps for D flip-flop implementation



$$D_C = C^+ = B' \qquad D_B = B^+ = C + BA'$$
$$D_A = A^+ = CA' + BA' = A'(C + B)$$

Binary Counter for Other Sequences

- Logic diagram for D flip-flop implementation



Self-Starting Counter

- Problems with incompletely specified transition graph
  - The initial states may be unpredictable when the power is turned on
  - What happens if the state becomes unspecified
  - Example: state 001 (C = 0, B = 0, A = 1)

- *Self-starting counter*
  - Don't care states eventually lead into the main counting sequence
  - Example: state 001 goes to 111