Lecture 10: Registers and counters

Chapter 6.1, 6.6

Objectives

- Analysis of *registers* and *counters* as an integration of sequential circuits
  - Sequential functional blocks

Sequential functional blocks

- Sequential circuits that perform certain specific functions
- Structured circuits, with multiple stages or cells that are (close to) identical
- Expansion of circuits is simple
- Registers and counters
  - Registers: store information during data processing
  - Counters: assist in sequencing data processing

Outline

- Registers
- Counters

Registers (1)

- Composed of flip-flops
  - n-bit register: n flip-flops, with one flip-flop storing 1 bit binary information
- Combined with combinational logic gates
  - Flip-flops hold data
  - Combinational logic gates determine the new or transformed data to be transferred into flip-flops
- Useful to store and manipulate data information

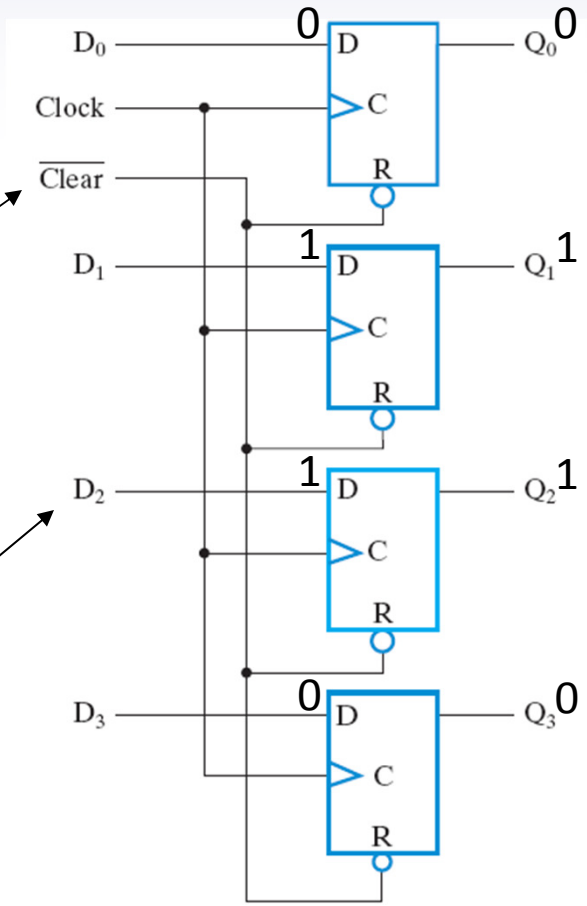
Registers (2)

Example: a 4-bit reigster

4 D flip flops, with clock signal and clear input

$\overline{Clear}$ input:
Clear = 1 to clear flip-flops' values to 0;
Clear = 0 during normal operations

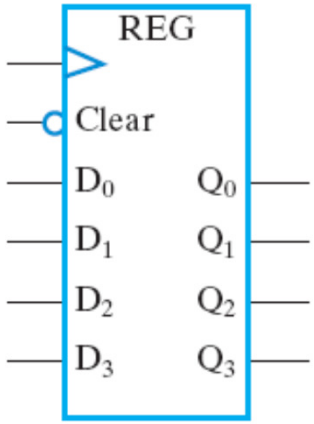
D inputs:
Data (binary values) to be stored



(a) Logic diagram

Positive clock edge:
Transfer 4-bit binary information into the register (a.k.a. loading a register)

Parallel loading: all bits of registers are loaded with a common clock pulse

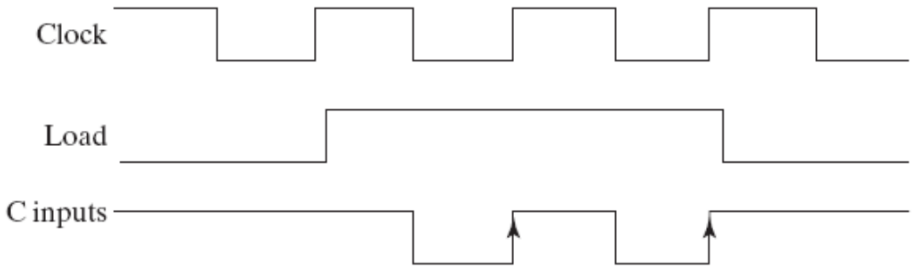


Graphical symbol

Registers with load control

- We want to keep the contents of register unchanged
- Method: prevent clock signal entering the circuit
- Load control input: a load control (Load) combined with the clock signal
- Also known as "clock gating"

(c) Load control input



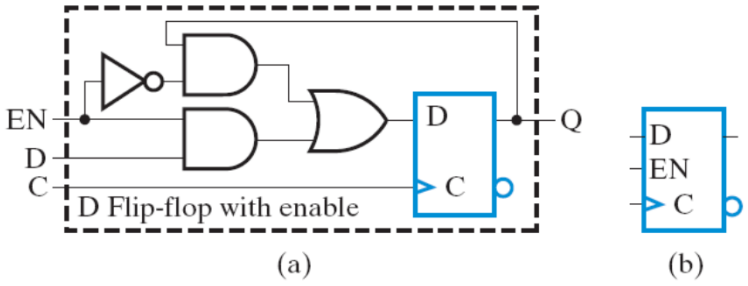
(d) Timing diagram

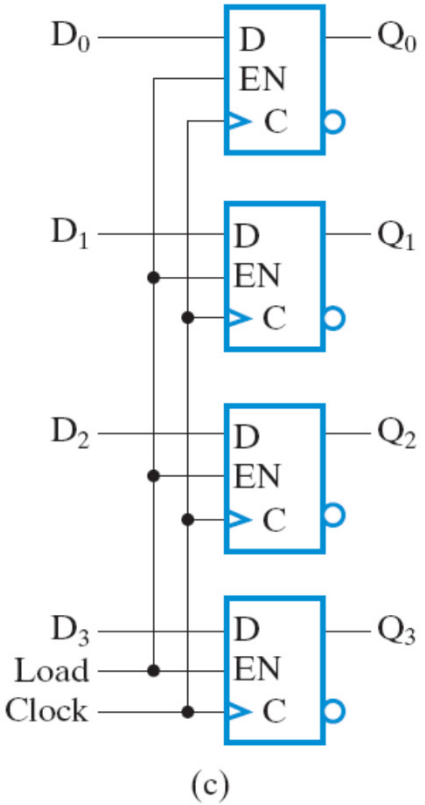
Clock skewing effect

- Clock gating may produce *clock skew*
  - Clock skewing effect: clock signals arrive at different flip-flops or registers at different time
- Flip-flops with *clock gating* receive clock signal later than others without clock gating
  - Gate delays exist on logic gates, and hence on the clock signals
- Not desirable
  - Should ensure all flip-flops are triggered at the same time in a truly synchronous system

D flip-flop with enable

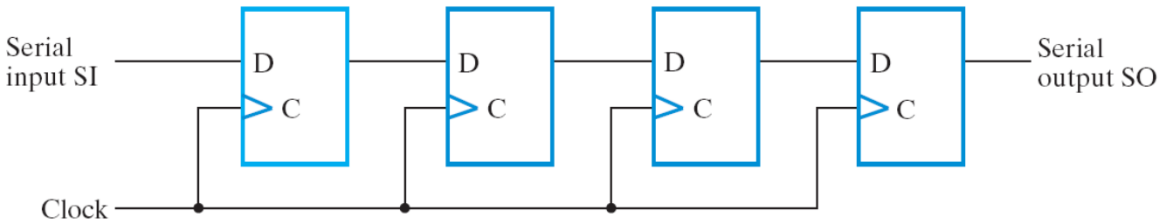
- Use 4 *D flip-flops with enable* to build a 4-bit register
- During a positive clock edge
  - Load = 1: data in D_0 to D_3 are transferred into the register
  - Load = 0: current value remains in the register
- Avoids clock skewing effect
  - Clock pulses are applied continuously to input C of flip-flops





Shift registers

- Operations: shift the stored bits laterally in one or both directions
- Implementation
  - Chain of flip-flops, output of one flip-flop connects to input of the next flip-flop
  - Common clock pulse to all flip-flops

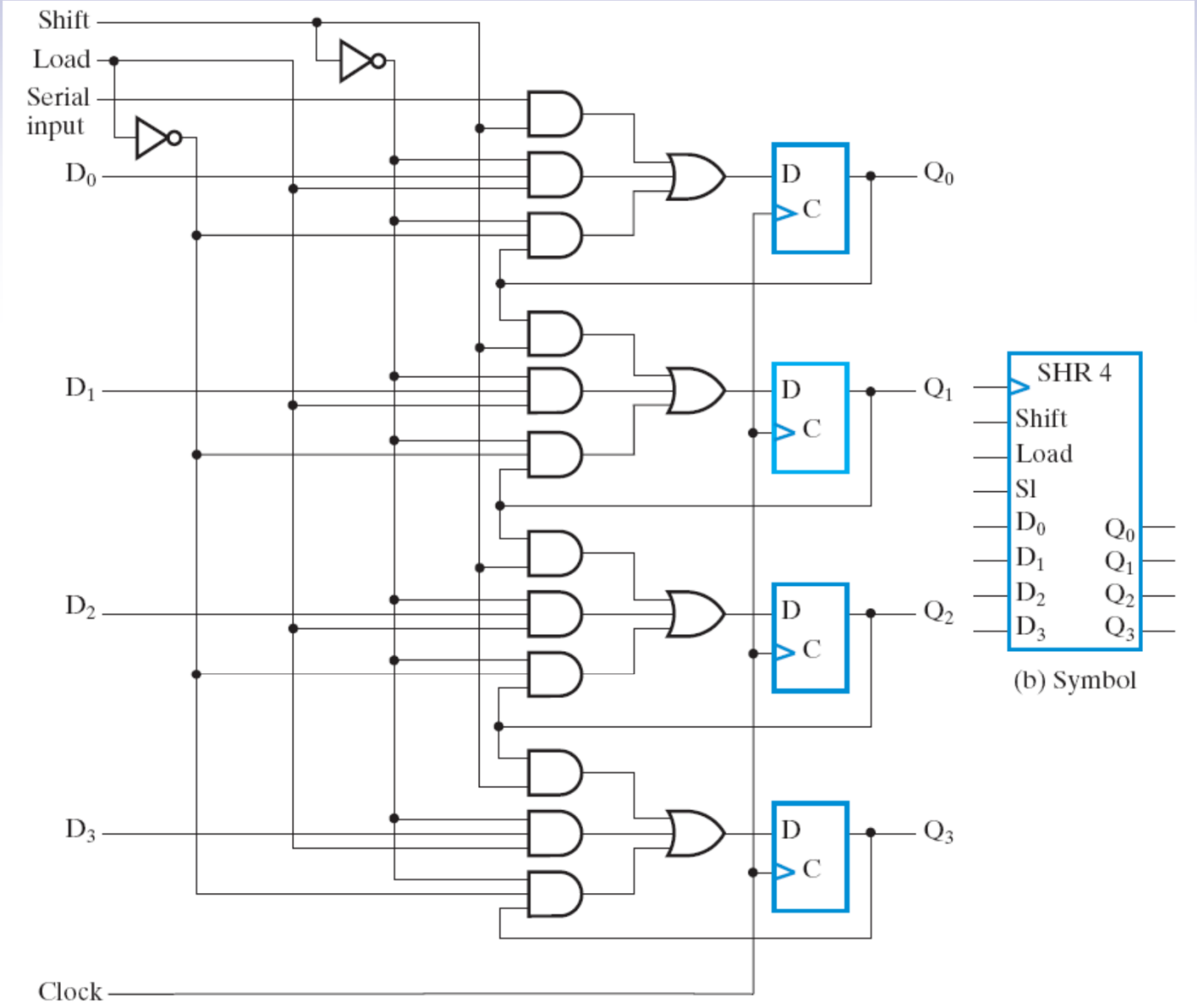


(a) Logic diagram

One bit as output in one clock pulse

(b) Symbol

Shift register with parallel load (1)

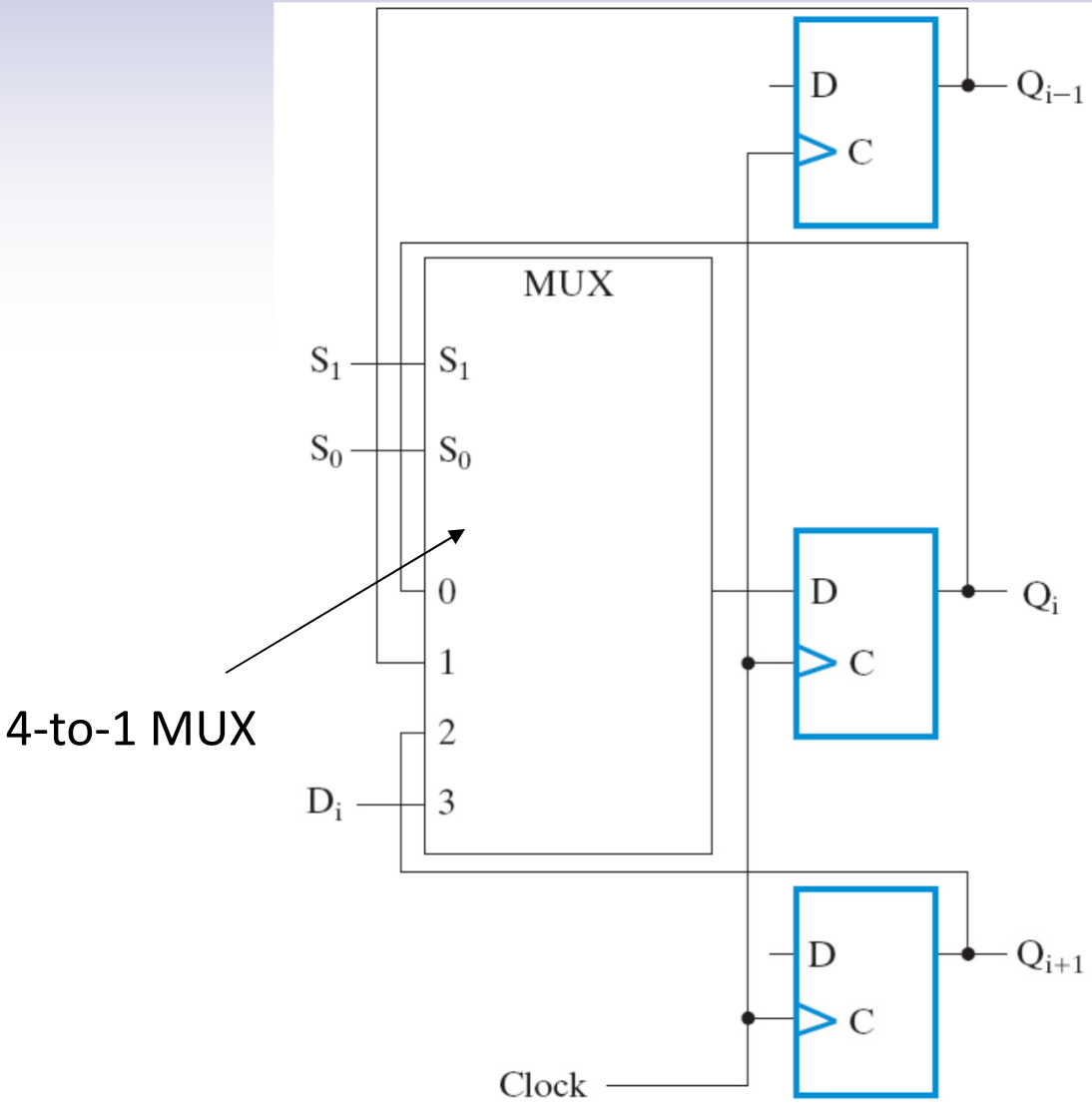


Shift register with parallel load (2)

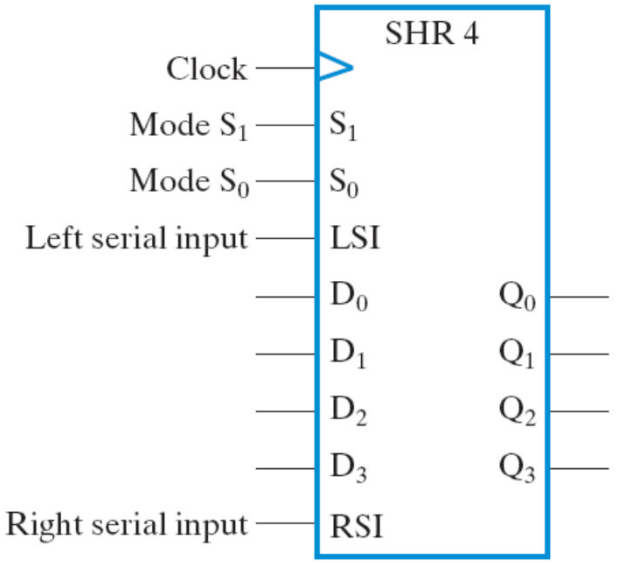
- Shift = 1 → enable first AND gate
  - Bit shifts down from Q_0 to Q_3 (Q_0 = serial input)
- Shift = 0 & Load = 1 → enable second AND gate
  - Load parallel data D_0 to D_3 into registers (Q_0 to Q_3)
- Shift = 0 & Load = 0 → enable third AND gate
  - Q_i connected back to flip-flop i (no change)

Shift	Load	Operation
0	0	No change
0	1	Load parallel data
1	X	Shift down from Q_0 to Q_3

Bidirectional shift register (1)



(a) Logic diagram of one typical stage



(b) Symbol

Bidirectional shift register (2)

- Shifting data in both directions
- A multiplexer at each stage i
  - Selection inputs: control operation of the register at stage i

Mode control		Register Operation
S_1	S_0	
0	0	No change
0	1	Shift down
1	0	Shift up
1	1	Parallel load

Outline

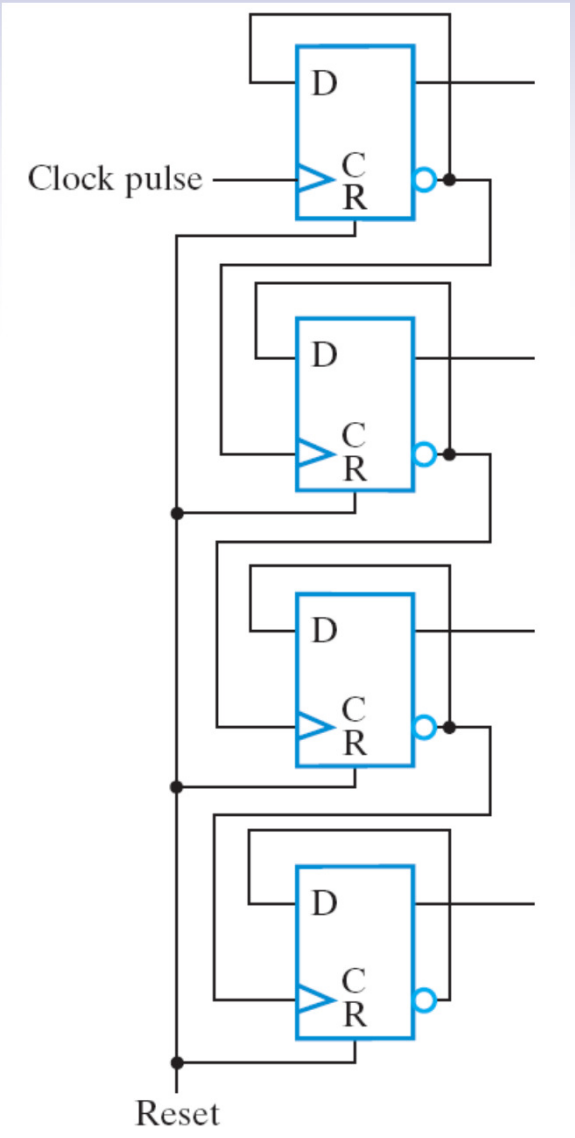
- Registers
- Counters

Counters

- A register that goes through a predetermined sequence of binary states upon application of clock pulse
  - States are the "counting values"
  - Combinational logic gates are connected in a way to produce the state sequence
- Binary counter
  - Follows binary number sequence
  - n-bit binary counter: *n* flip-flops, counts from 0 to $2^n - 1$
- Function
  - To sequence and control operations in a digital system
- Two categories
  - Synchronous vs. asychronous (ripple counters)

4-bit binary ripple counter (1)

- Example: built by positive edge-triggered D flip-flops
- Behaviour
  - Output of a flip-flop is complemented upon a positive edge to C input
  - D flip-flops are triggered at different times
- Implementation
  - **D input**: complemented output of itself
  - **C input of first flip-flop**: clock pulse
  - **C input of other flip-flops**: complemented output of the previous (less significant) flip-flop



4-bit binary ripple counter (2)

D input value?

Q_i: 1 → 0, Q_{i+1} is complemented

Hence, D input is complement of own output

C input value?

Flip-flop in bit position i+1 should be triggered at a time when Q_i changes from 1 to 0

Flip-flops are positive edge triggered (i.e., triggered when C changes from 0 to 1)

Hence, C input of flip-flop i+1 = complemented Q_i

How about the implementation of a downward counter?

Upward Counting Sequence				Downward Counting Sequence			
Q_3	Q_2	Q_1	Q_0	Q_3	Q_2	Q_1	Q_0
0	0	0	0	1	1	1	1
0	0	0	1	1	1	1	0
0	0	1	0	1	1	0	1
0	0	1	1	1	1	0	0
0	1	0	0	1	0	1	1
0	1	0	1	1	0	1	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	0	0
1	0	0	0	0	1	1	1
1	0	0	1	0	1	1	0
1	0	1	0	0	1	0	1
1	0	1	1	0	1	0	0
1	1	0	0	0	0	1	1
1	1	0	1	0	0	1	0
1	1	1	0	0	0	0	1
1	1	1	1	0	0	0	0

4-bit binary ripple counter (3)

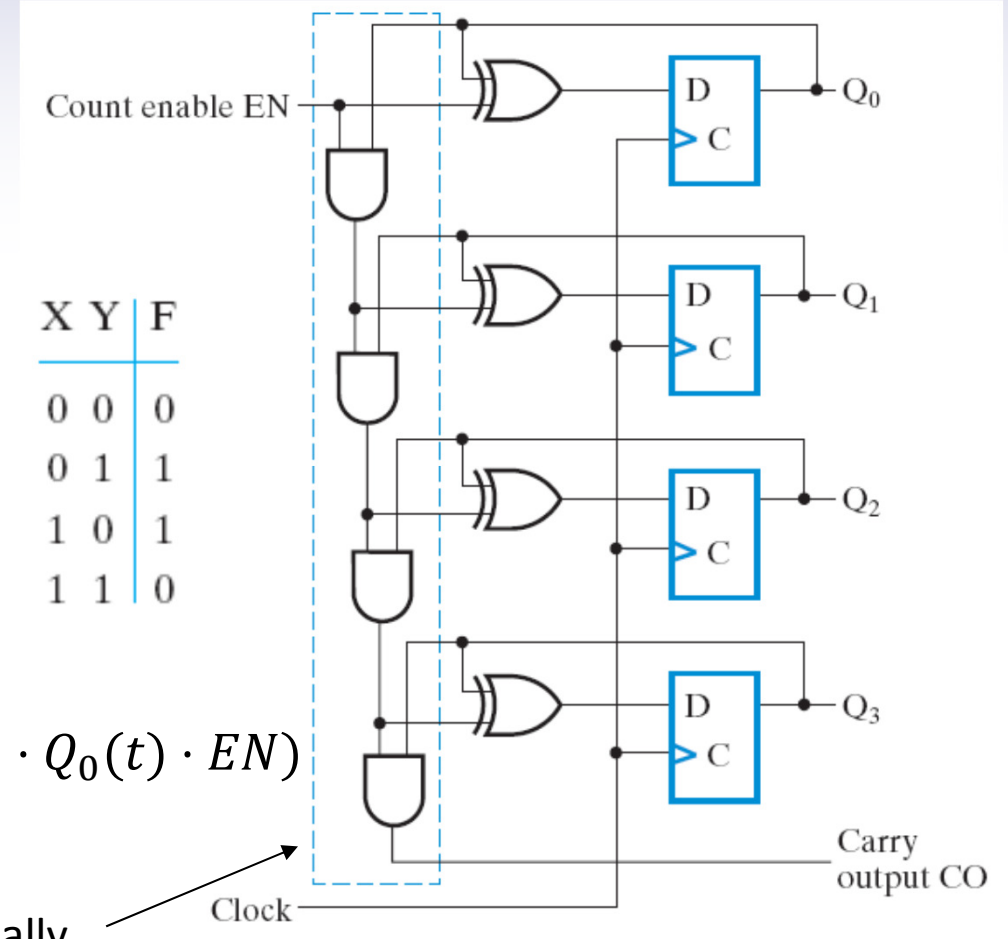
- Advantages
  - Simple hardware
- Disadvantages
  - Asynchronous, hence is delay dependent and unreliable
    - Clock applies to C input of one flip-flop only
  - Slow in counting one state: changes in each bit can occur only after the less significant bit has changed

- Synchronous binary counter
  - One common clock pulse triggers all flip-flops at the same time
  - Serial and parallel counters

Synchronous serial counter

- Upon a positive clock edge
  - Q_i is complemented only when $Q_{i-1}, Q_{i-2}, \ldots, Q_1, Q_0$ are all 1s
  - Otherwise, Q_i is unchanged
- Implementation: XOR gate
  - Take complement if less significant Qs are 1

X	Y	F
0	0	0
0	1	1
1	0	1
1	1	0



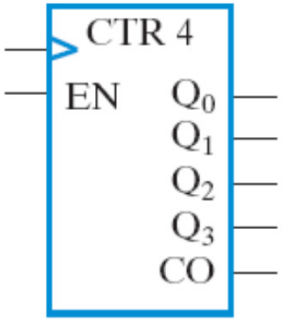
$$Q_i(t+1) = D_i(t)$$
$$= Q_i(t) \oplus (Q_{i-1}(t) \cdot Q_{i-2}(t) \cdot \ldots \cdot Q_1(t) \cdot Q_0(t) \cdot EN)$$

AND operation is performed serially

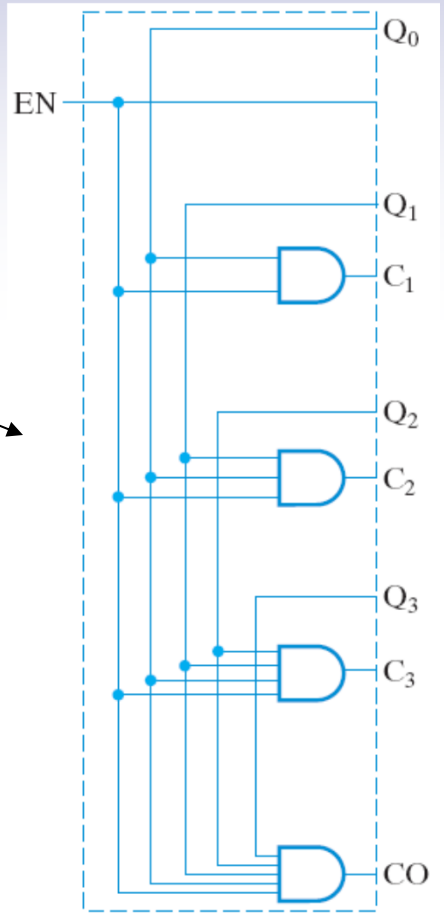
Synchronous parallel counter

Another implementation for the logic:

$Q_{i-1}(t) \cdot Q_{i-2}(t) \cdot \ldots \cdot Q_1(t) \cdot Q_0(t) \cdot EN$



Logic diagram of 4-bit synchronous binary counter



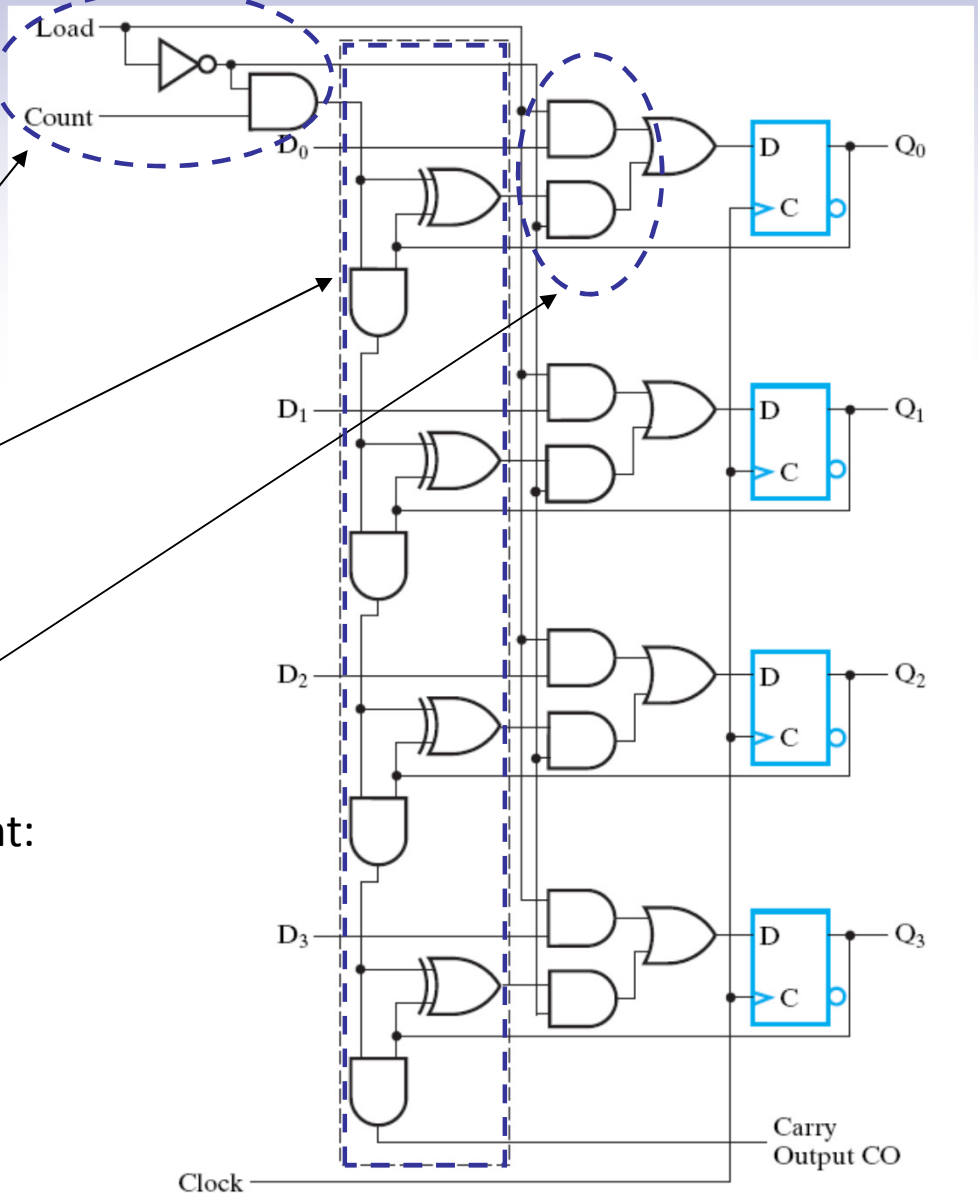
Binary counter with parallel load

Allows an initial value (binary number), e.g., D_3:D_0 to be loaded prior to counting

Selection input logic

Gates for counting

Gates for selection logic



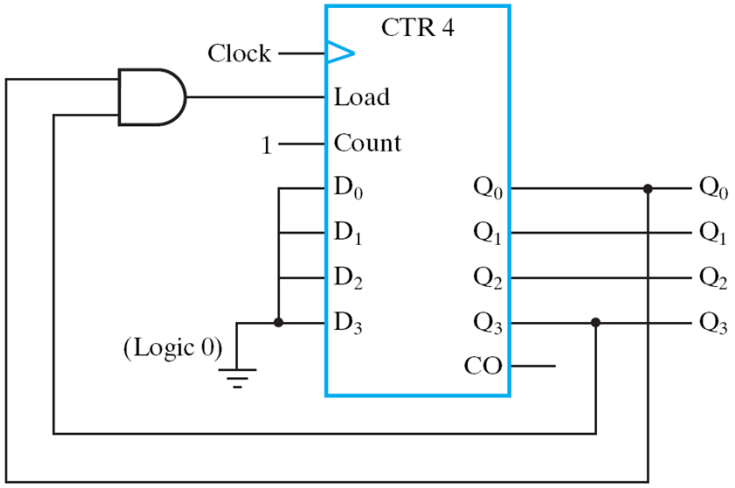
Selection based on (1) Load, (2) Count:
Load = 1: D_3:D_0 loaded into counter
Load = 0 & Count = 1: count
Load = 0 & Count = 0: hold

Divide-by-N counter

- Counters that go through a repeated sequence of *N* states
- Examples
  - BCD counters
  - Counters with arbitrary sequence of states

BCD counter

- Implementation
  - Use a binary counter with parallel load
  - Set Load = 1 when Q = 1001
    - Hence, count again from 0000 when 1001 is reached



Counter with arbitrary count (1)

- Study Guide Worked Example 10.2

Counter with arbitrary count (2)

- Example: counter that counts the sequence: 000, 001, 010, 100, 101, 110, and reset to 000

Present State			Next State		
A	B	C	DA = A(t+1)	DB = B(t+1)	DC= C(t+1)
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	1	0	0
1	0	0	1	0	1
1	0	1	1	1	0
1	1	0	0	0	0

Flip-flop input equations:

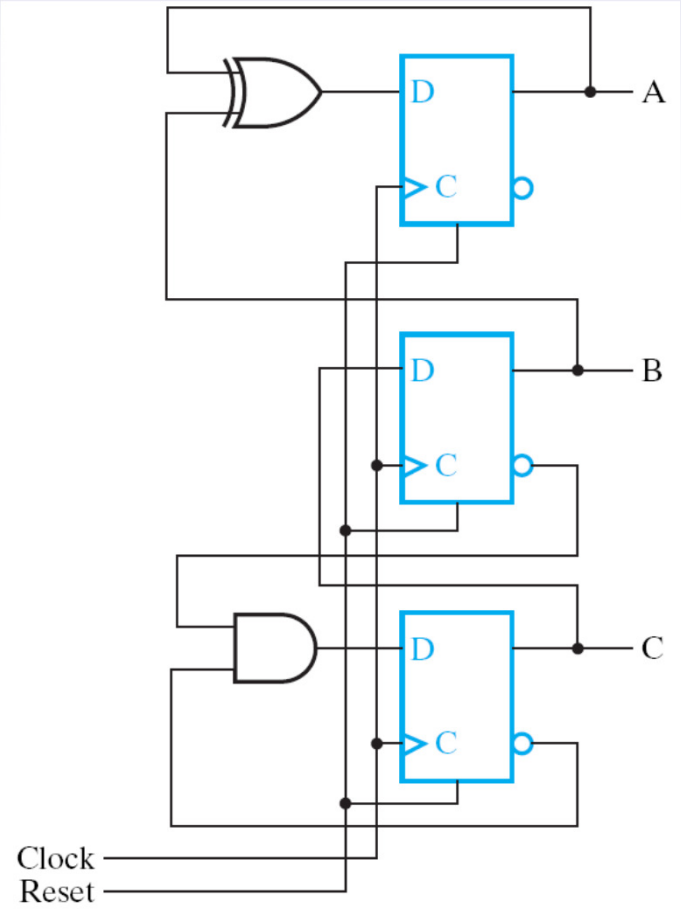
$$D_A = A \oplus B$$

$$D_B = C$$

$$D_C = \overline{B}\,\overline{C}$$

Counter with arbitrary count (2)

Logic diagram of a counter with arbitrary count



Summary and acknowledgment

- Registers are set of interconnected flip-flops with combinational logic
- Counters go through a predetermined sequence of binary states
- Implementation of different registers and counters are studied

- Reference
  - Logic and Computer Design Fundamentals (5/e)