UNIVERSITY OF NEW BRUNSWICK

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

UNB

ECE2214

Digital Logic Design

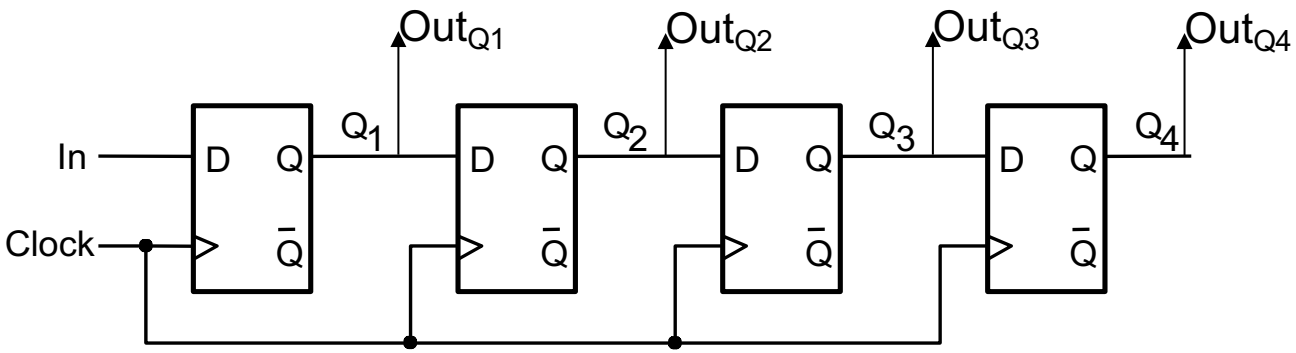
L-23 More on counters

Outline

- Synchronous counters with D flip-flops
- D-type FF counters with parallel loads
- Synchronous reset counters
- Other types of counters
  - BCD counters
  - Ring counters
  - Johnson counters

Summary of previous lecture

- Registers are a set of ***n*** flip-flops used to store n bits of information
  - Shift registers enable shifting the content based on a clock signal



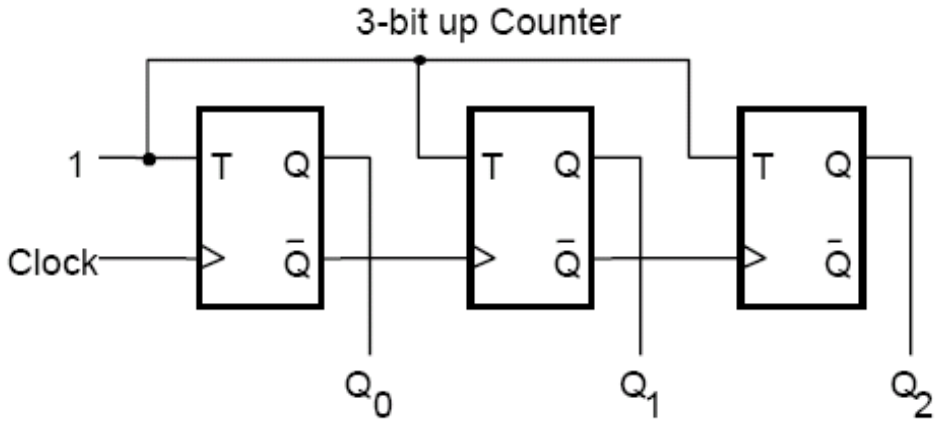
Circuit

- Counters are special case of adders
  - Asynchronous and synchronous
  - Can be implemented with T, D and J-K FF

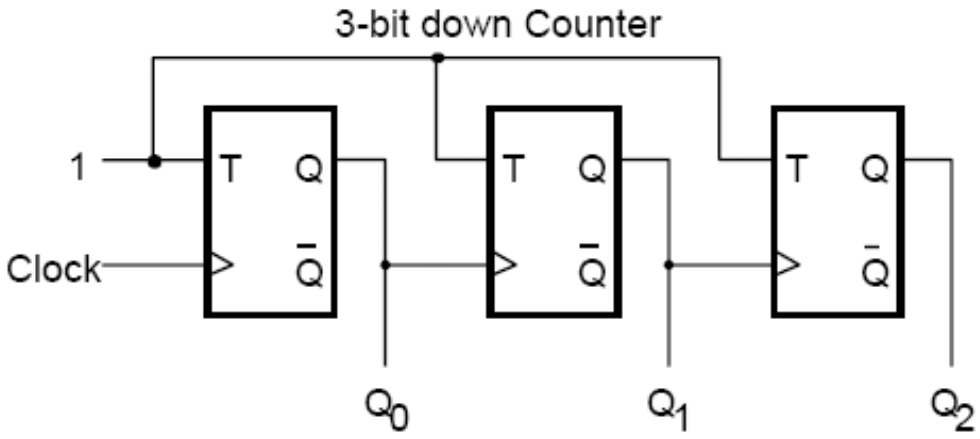
Summary of previous lecture

- Asynchronous counters with T FF
  - Inputs T = 1, CLK are cascaded to allow consecutive counting

Up-counter



Down-counter

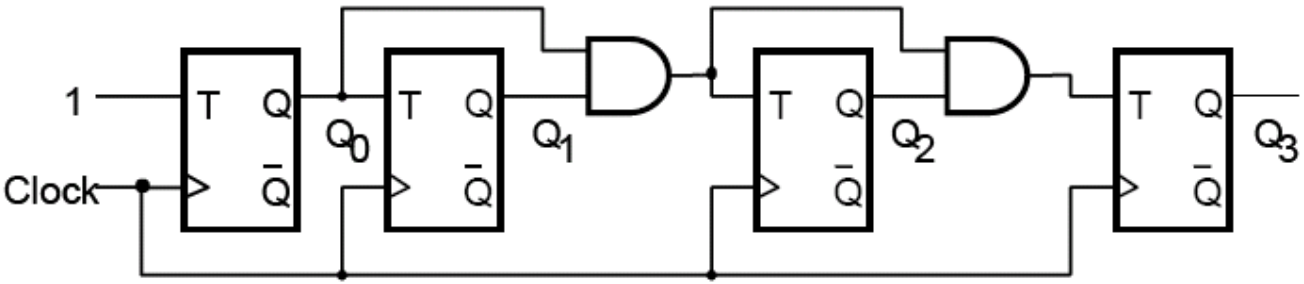


Summary of previous lecture

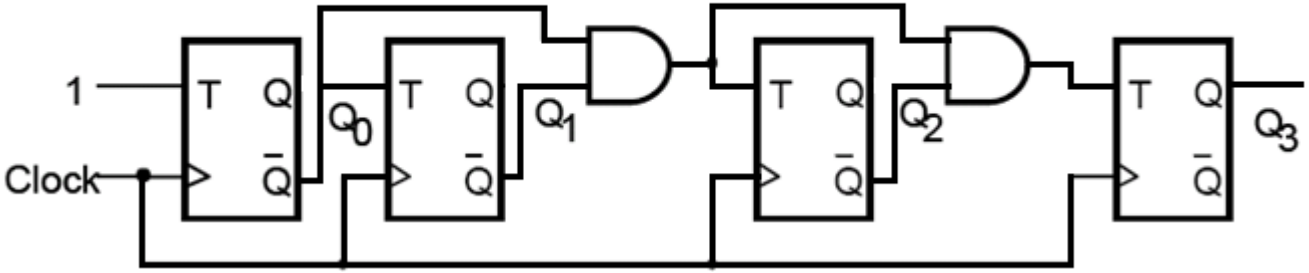
- Synchronous counters with T FF
  - All clocks are set together to obtain synchronization
  - Inputs must to be combined to obtain counting sequence

Q_3	Q_2	Q_1	Q_0
1	1	1	1
1	1	1	0
1	1	0	1
1	1	0	0
1	0	1	1
1	0	1	0
1	0	0	1
…			
…			
0	1	0	0
0	0	1	1
0	0	1	0
0	0	0	1
0	0	0	0

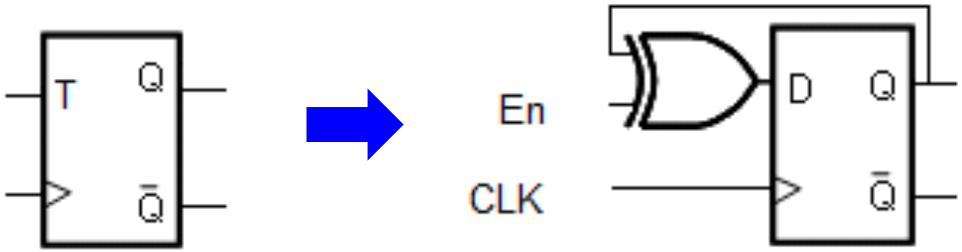
4-bit synchronous up counter



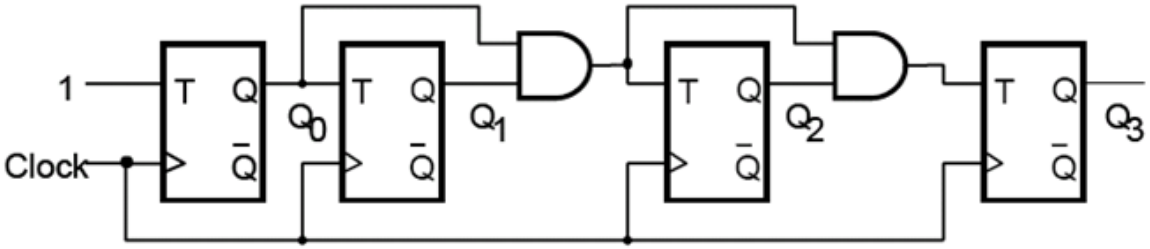
4-bit synchronous down counter

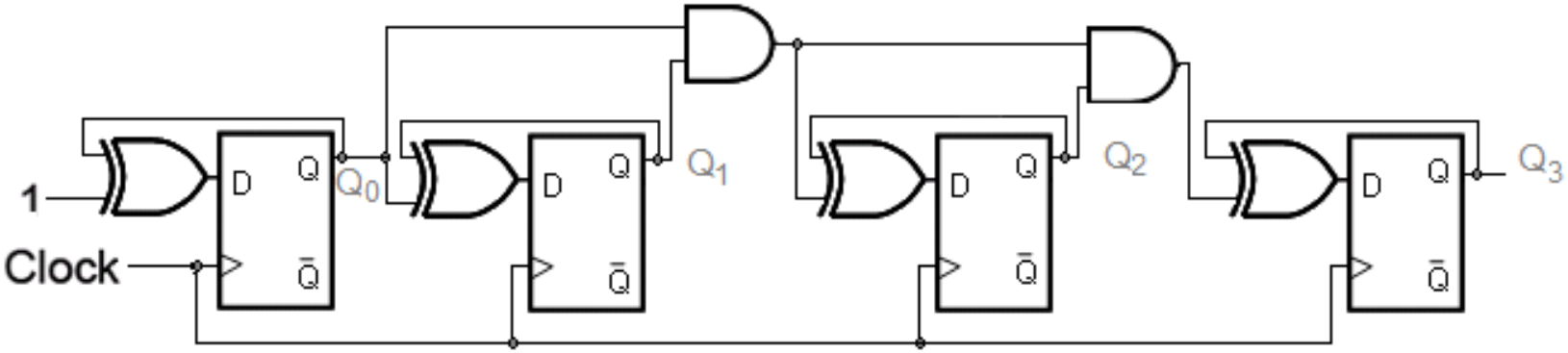


Replacing T flip-flops by D flip-flops



En	*Q*	D = *En* ⊕ *Q*
0	0	0
0	1	1
1	0	1
1	1	0

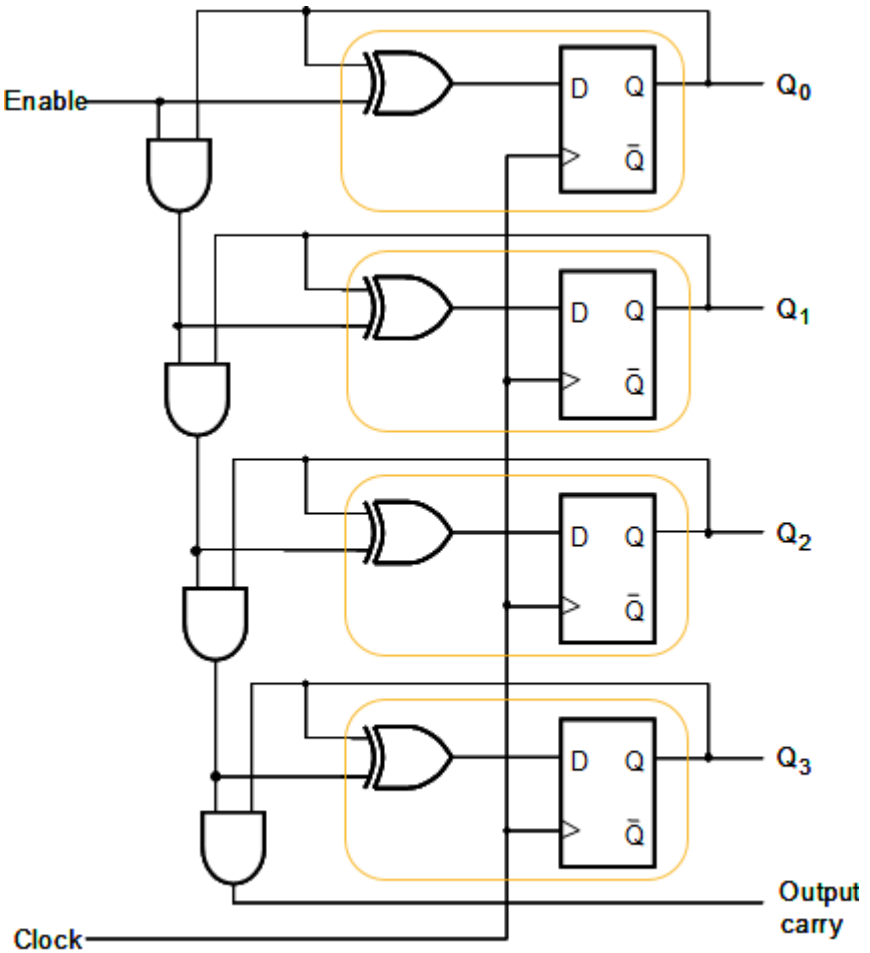




Synchronous counters with D flip-flops

- D F-F need to set D=Q to hold/memorize the current state (when CLK ↑, Q = D)
- Enable/disable properties must be implemented with logic
  - E_N = 0, the counting is disabled (output of AND gate always 0 and Q is fed back to the D input holding the current state)
  - E_N = 1, counting is enabled (output of AND gate will be Q from previous F-F and D will be toggle when previous Q_{N-1} differs from Q_N)

Q_3	Q_2	Q_1	Q_0
1	1	1	1
1	1	1	0
1	1	0	1
1	1	0	0
1	0	1	1
1	0	1	0
1	0	0	1
1	0	0	0
0	1	1	1
0	1	1	0
0	1	0	1
0	1	0	0
0	0	1	1
0	0	1	0
0	0	0	1
0	0	0	0



A four-bit counter with D flip-flops

Generalization of synchronous counters implementation with D flip-flops

- The equations that describe this implementation for an n bit counter are:

$D_0 = Q_0 \oplus E_N$

$D_1 = Q_1 \oplus Q_0E_N$

$D_2 = Q_2 \oplus Q_1Q_0E_N$

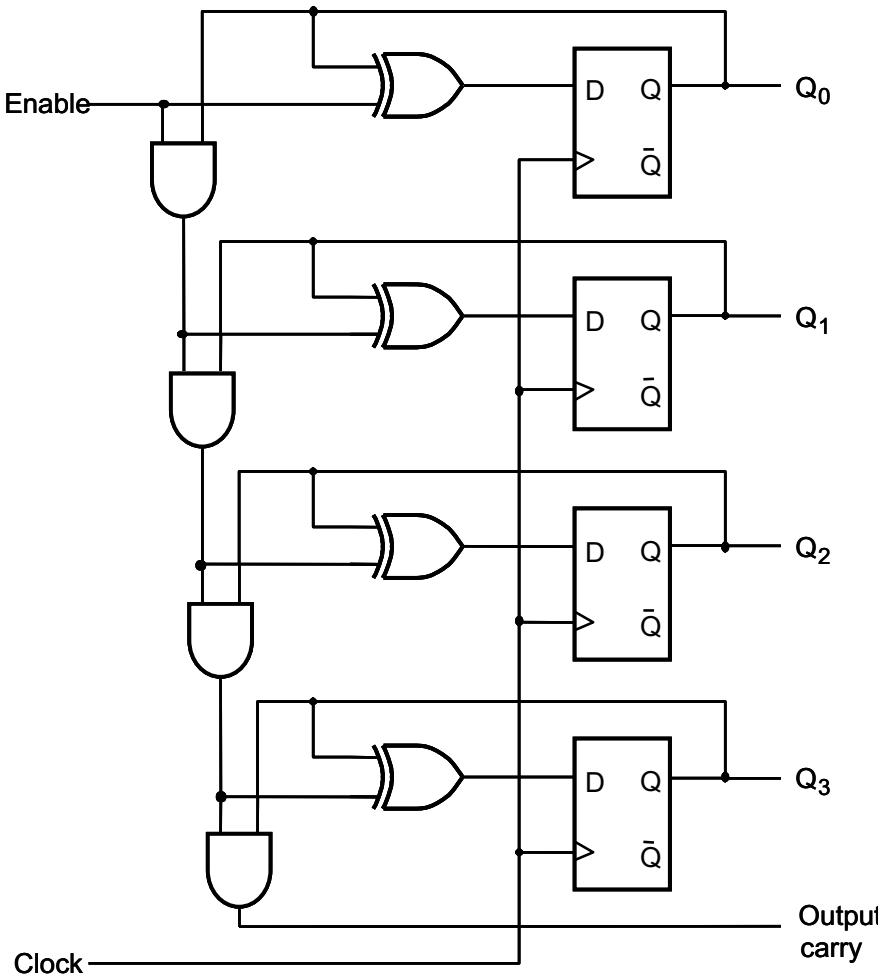
$D_3 = Q_3 \oplus Q_2Q_1Q_0E_N$

…

…

…

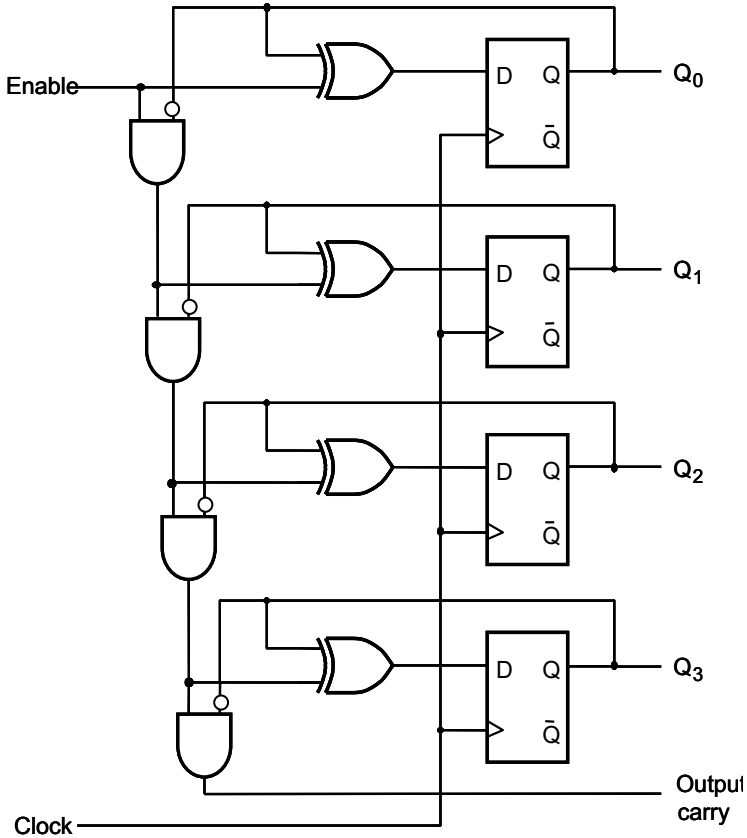
$D_n = Q_n \oplus Q_{n-1}. . .Q_1Q_0E_N$



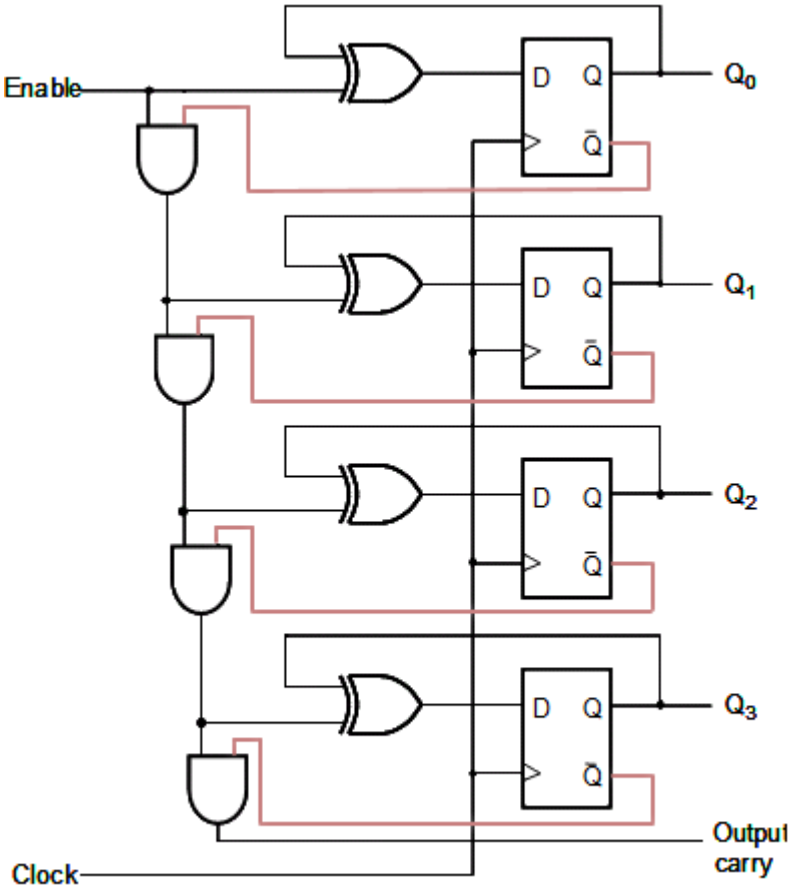
A four-bit counter with D flip-flops

Synchronous down counter with D flip-flops

- Q_0 complemented is fed back to the AND gate that enable other bits



A four-bit counter with D flip-flops

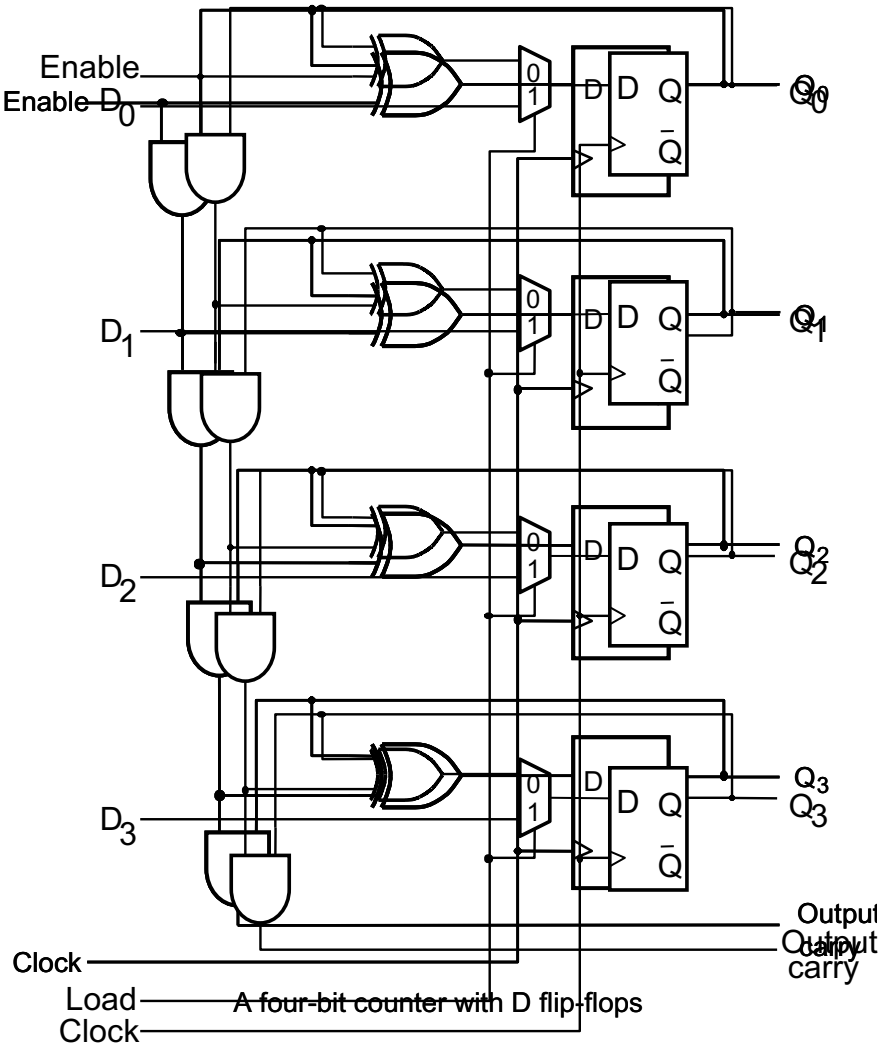


Q_3	Q_2	Q_1	Q_0
1	1	1	1
1	1	1	0
1	1	0	1
1	1	0	0
1	0	1	1
1	0	1	0
1	0	0	1
1	0	0	0
0	1	1	1
0	1	1	0
0	1	0	1
0	1	0	0
0	0	1	1
0	0	1	0
0	0	0	1
0	0	0	0

Synchronous up counter with D flip-flops and parallel loads

- A multiplexer is added to select the external bit or cascade the previous stage
- External bit $D_3D_2D_1D_0$ are loaded with

  Load = 1 and counts when load = 0
- Note that complexity of this implementation is remarkable compared to the same implementation using T F-Fs



A counter with parallel-load capability.

Synchronous reset

- Synchronous reset is necessary when the application requires counting up to a number that is not a power of 2
- An N-bit binary counter is reset automatically after reaching the number 2^N-1
  - A higher number will not fit in the available number of bits (N)
  - The output automatically are reset to 0
- Example
  - Suppose a 3-bit counter
    - CLK0 → 000
    - CLK1 → 001
    - CLK2 → 010
    - CLK3 → 011
    - CLK4 → 100
    - CLK5 → 101
    - CLK6 → 110
    - CLK7 → 111
    - CLK8 → 000

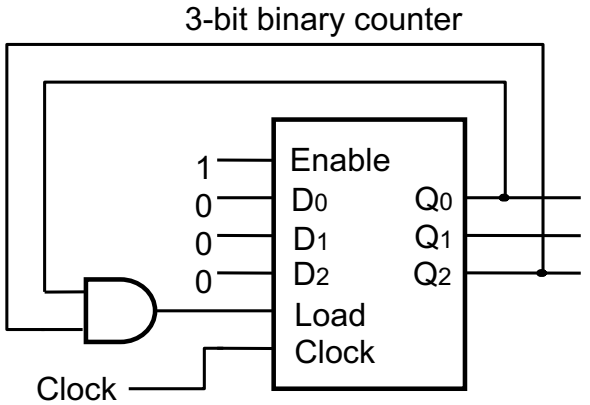
- **What if we don't want to count until the maximum value possible for N bits?**
  - Counting from 0 to 5 (module 6)
    - **CLK0 → 000**
    - **CLK1 → 001**
    - **CLK2 → 010**
    - **CLK3 → 011**
    - **CLK4 → 100**
    - **CLK5 → 101**
    - CLK6 → 000
    - CLK7 → 001
    - CLK8 → 010

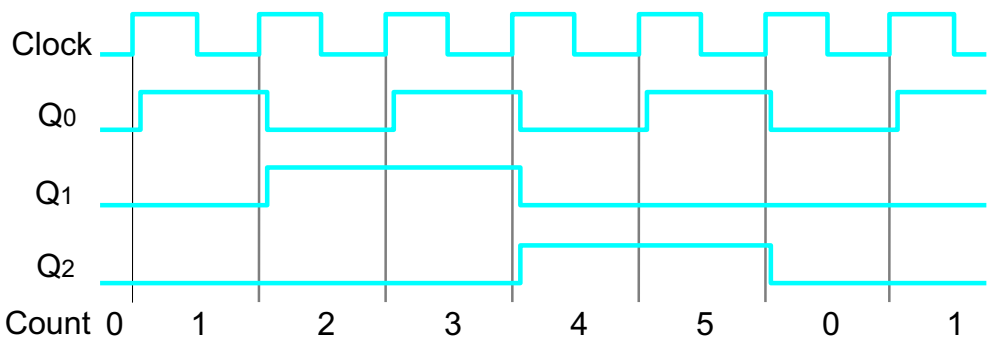
- **Need to add extra logic to reset the counter at CLK5**

Synchronous reset

- Suppose a counter Module 6 (C_6) :
- The sequence of number are: 0,1,2,3,4,5,0,1 …

- Need 3 bits to count from 0-5, but…
- Sequence 6 and 7 can't occur
- A logic circuit must be designed to:
  - Detect sequence 5 (101)
  - Set outputs to 000 with the next active CLK edge
- Reset of outputs can be generated
  - Using the load feature
  - Using the reset feature if F-F have synchronous reset input
- Notice from sequence 5
  - $D_2 = 1$ & $D_0 = 1$ occurs only for 5
  - No other previous sequence has that unique combination



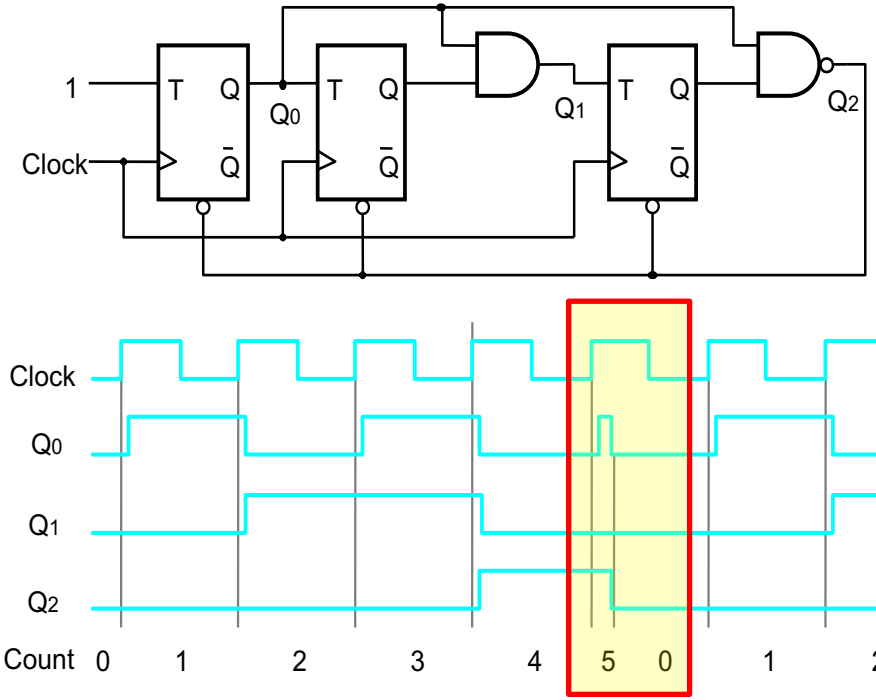


Timing diagram

Asynchronous reset

- Achieved when the signal that resets the counter is not clock synchronized
- Same example of the module 6 counter:

- Clear input is active low
- A NAND used to detect $D_2 = 1$ & $D_0 = 1$
- When the 5 occurs, then the CL inputs are activated…
- However, the number five will not be held for the whole clock cycle
  - Only during the time delay of NAND gate and the reset of the flip-flops
- Can be solved counting up to six and neglecting the glitch
  - NAND connected to $D_2 = 1$ & $D_1 = 1$



NOTE: This reset form is not always desirable since the delay introduced by the asynchronous reset could cause undesirable effects

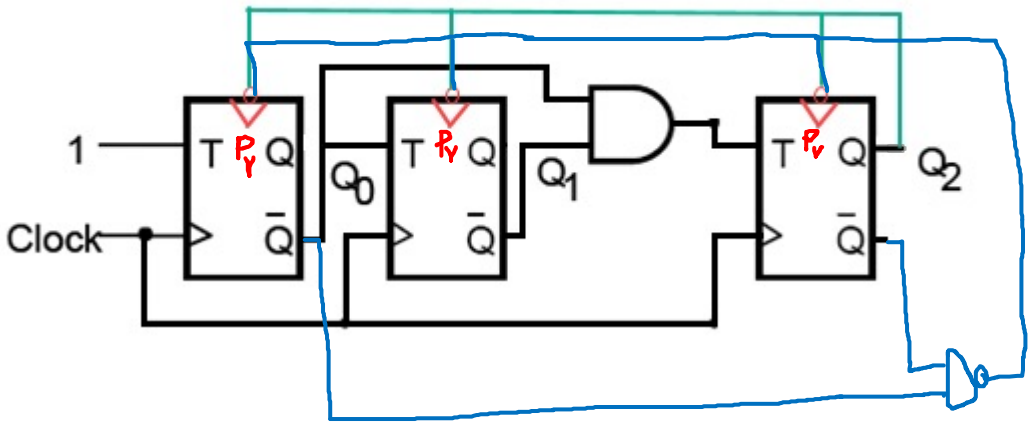
Design of up counters base N (non a power of 2)

- Synchronous reset for counters base N
  1. Implement an up counter with base 2^M where 2^M next power of 2 after N
  2. Determine the last state of the desired counting sequence **(N-1)**
  3. Analyze a unique combination of bits of the **N-1** sequence
  4. Design a logic that activate the parallel load synchronously or activate the synchronous reset
- Asynchronous reset for counters base N
  1. Implement an up counter with base where 2^M next power of 2 after N
  2. Determine the last state of the desired counting sequence **(N)**
  3. Analyze a unique combination of bits of the **N** sequence
  4. Design a logic circuit that activate the asynchronous **clear /preset** inputs of flip-flops
- What if the counter counts downward?

Design of synchronous down counters

- Similar to up counter but counter must be reset to the desired counting stage
- Best understood with an example
  - Design a 3-bit synchronous down counter with T FF that counts from **111** down to **011**
- Solution:
  - The last counting sequence is 011
  - $D_0 = 0$ is unique
  - Activate Pr (active low)

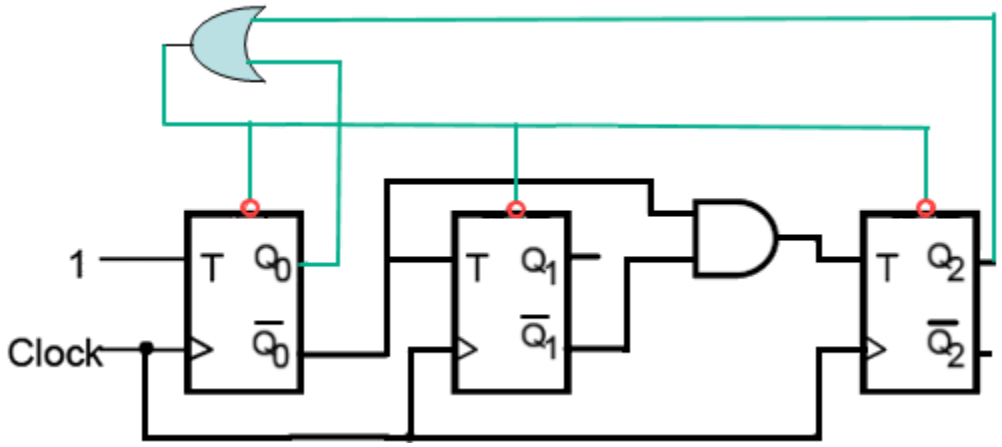
111
110
101
100
011
010
001
000



Design of asynchronous down counters

- Design a 3-bit asynchronous down counter with T FF that counts from **111** down to **011**

- Solution:
  - The last counting sequence is 010
  - $D_2 = 0$ & $D_0 = 0$ are unique
  - Activate Pr (active low) with an OR gate



111
110
101
100
011
010
001
000

Design of down counters base N (non a power of 2)

UNB

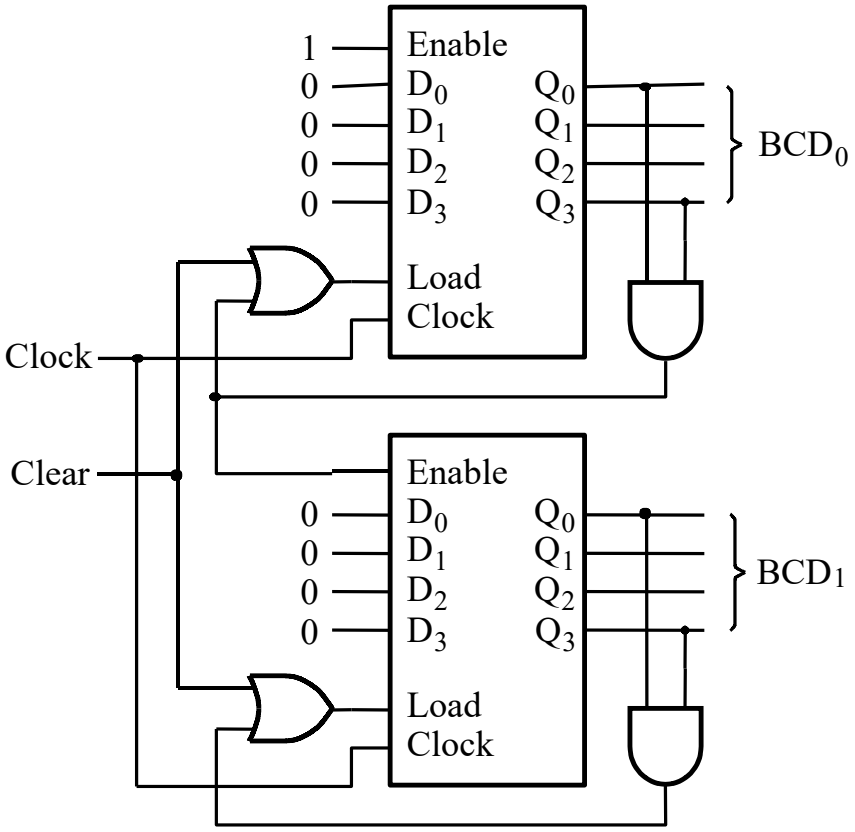
- Synchronous reset for counters base N
  1. Implement a down counter with base 2^M where 2^M next power of 2 after N
  2. Determine the last state of the desired counting sequence
  3. Design the logic that activate the parallel load or reset/preset synchronously when 2^M – N is reached
  4. Activate the parallel load with the initial counting sequence (default 2^M)

- Asynchronous reset for counters base N
  1. Implement a down counter with base 2^M where 2^M next power of 2 after N
  2. Determine last state of the desired counting sequence
  3. Design the logic that activate the parallel load synchronously when 2^M – (N+1) is reached
  4. Activate the **preset/reset** inputs to set the initial counting sequence (default 2^M)

Other types of counters

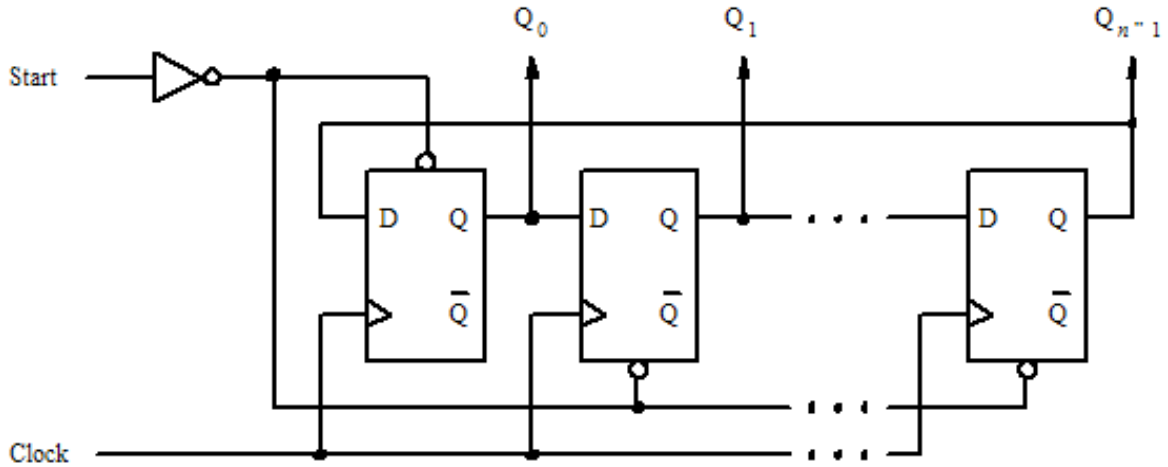
- Counters of many different base or module can be implemented by resetting the counting sequence when the desired number is met
- One example is a 2-bit BCD counter

- **BCD counter**
  - A binary-coded-decimal counter is based on a modulus 10 counter
  - Last sequence is **1001**
  - An AND gate is used to load 0000
  - Same signal activate the second digit
  - An external clear was added
- BCD counter is widely used to perform arithmetic operation in decimal system and provide the number to display in a 7-segment format



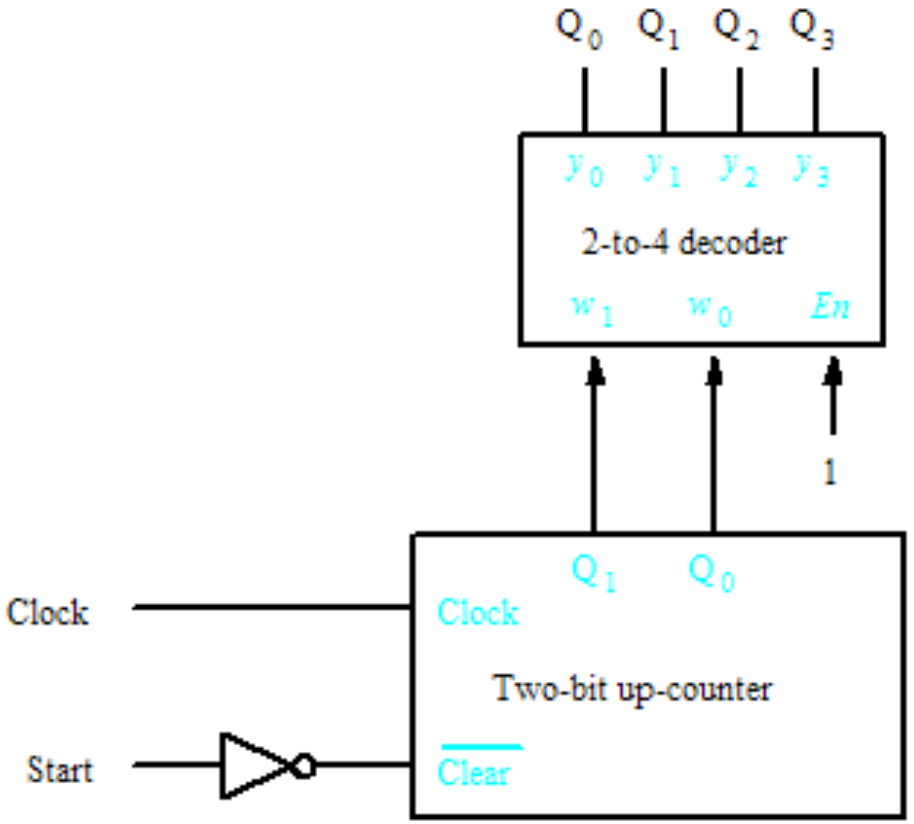
Other types of counters

- **Ring counters**
  - Ring counters are special type of counter that the output is not a binary number
  - Activate only one output in each clock cycle
  - The output of the last bit is fed back to the input
  - The output pattern is repeated after the last output is activated creating a type of ring
  - The best implementation of a ring counter is by injecting a ‘1’ to a shift register and shifting it to one of the sides



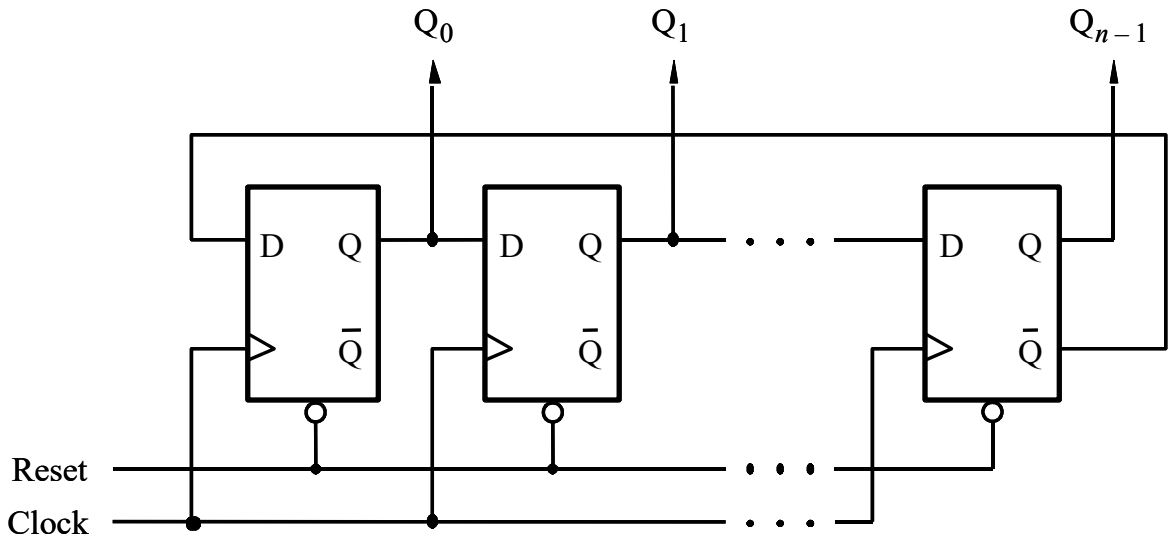
Other types of counters

- **Ring counter**
  - Same counter can be implemented by combining a 2-bit binary counter and a decoder
  - Example:



Other types of counters

- **Johnson counters**
  - This is a counter that is obtained from a slight modification to the ring counter
  - Feeds back the complemented output of the last digit instead of Q
  - This makes that the pattern:



Example:

0000
0001
0011
0111
1111
1110
1100
1000
0000
0001
0011
0111
1111
…

Summary

- Synchronous counters with D flip-flops
- D-type FF counters with parallel loads
- Synchronous reset counters
- Design of an arbitrary base down and up counters
- Other types of counters : BCD, ring counters and Johnson counters
- Reference
  - Fundamentals of Digital Logic with VHDL Design 3/e, Stephen Brown, Zvonko Vranesic; McGraw-Hill, 2009. Chapter 7, pp. 409-418
  - Digital Design; Principles and Practices. Fourth Edition. John F. Wakerly, Prentice Hall, 2006. ISBN 0-13-186389-4. Chapter 7, pp. 710-727

- Recommended exercises from the book
  1. Problems of pp. 473 of textbook
     - 7.23 - 7.31