ShanghaiTech University

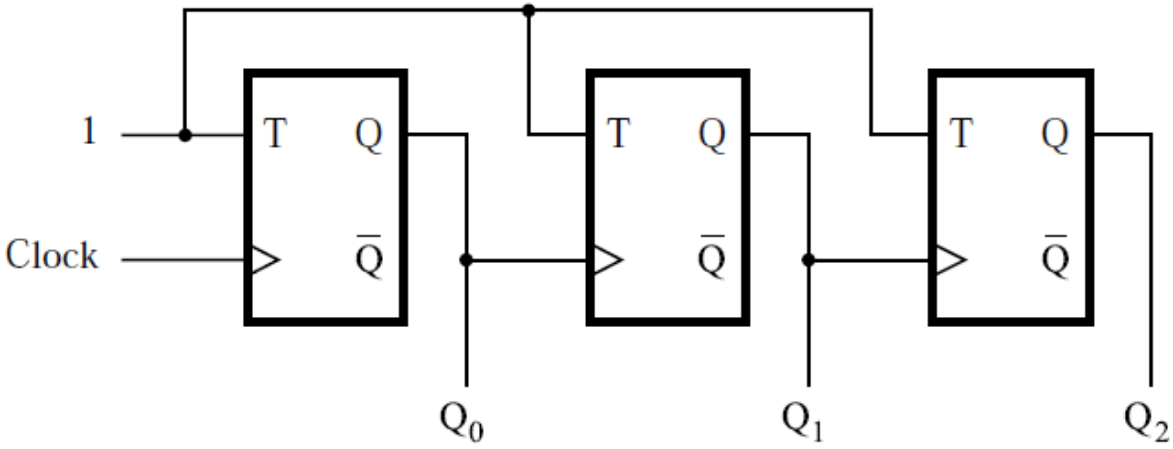
EE 115B: Digital Circuits

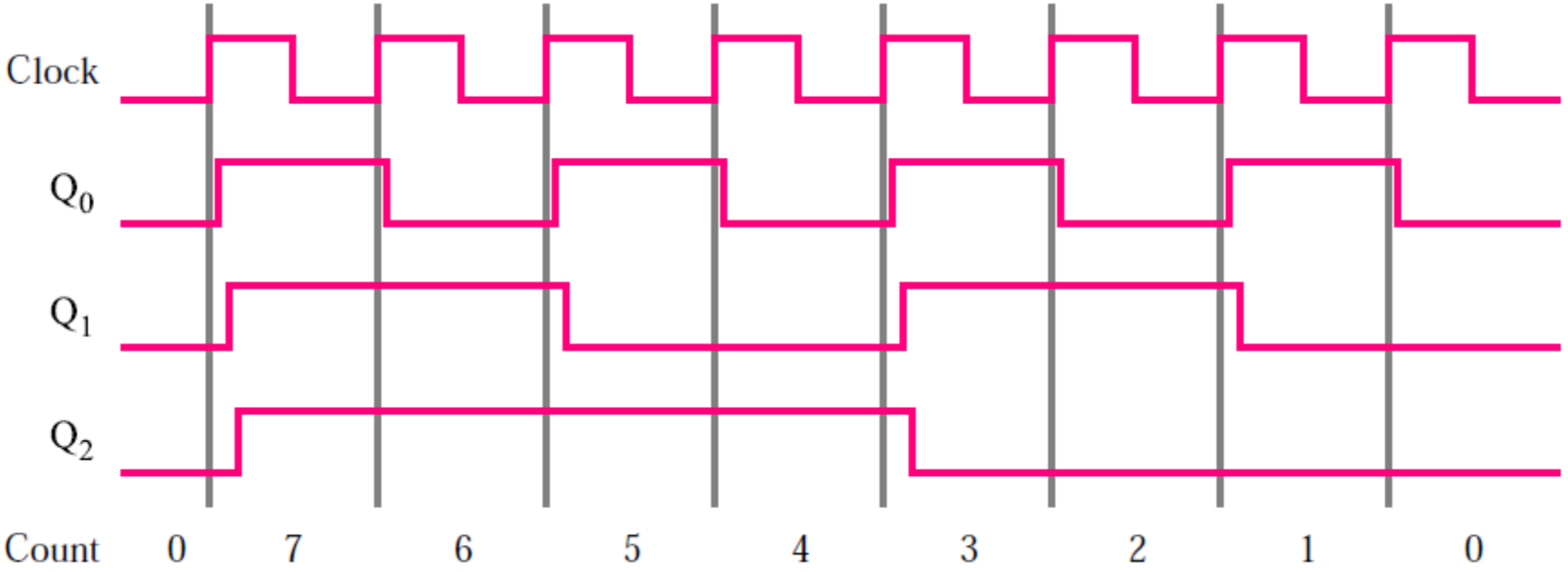
Fall 2022

Lecture 19

Hengzhao Yang
December 13, 2022

Asynchronous down-counter with T flip-flops



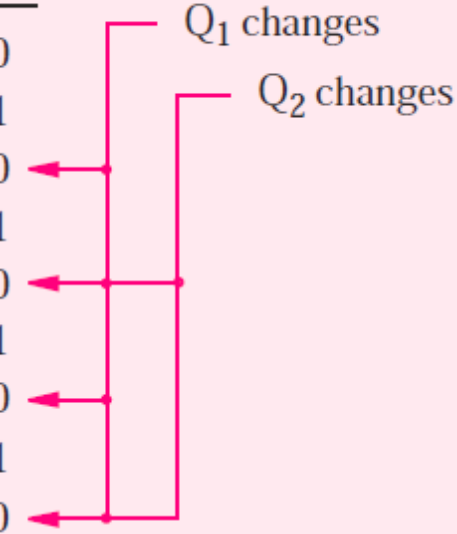


Synchronous up-counter with T flip-flops

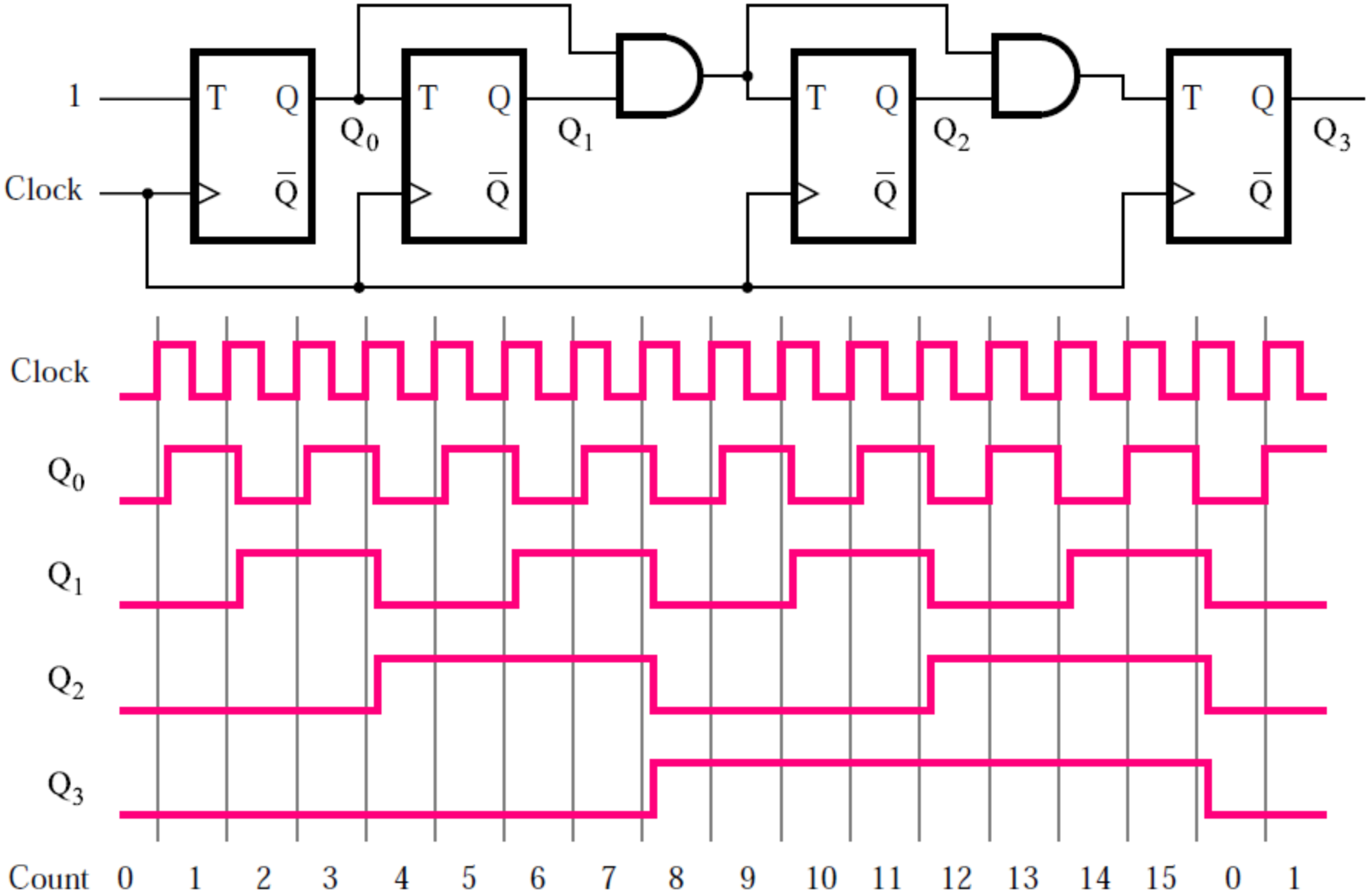
- Bit pattern: a given stage changes only when all the preceding stages satisfy Q=1
  - Q_0 changes at each clock positive edge
  - Q_1 changes only when Q_0=1
  - Q_2 changes only when Q_0=1 AND Q_1=1
- T input

$$T_0 = 1$$
$$T_1 = Q_0$$
$$T_2 = Q_0 Q_1$$
$$T_3 = Q_0 Q_1 Q_2$$
$$\vdots$$
$$T_n = Q_0 Q_1 \cdots Q_{n-1}$$

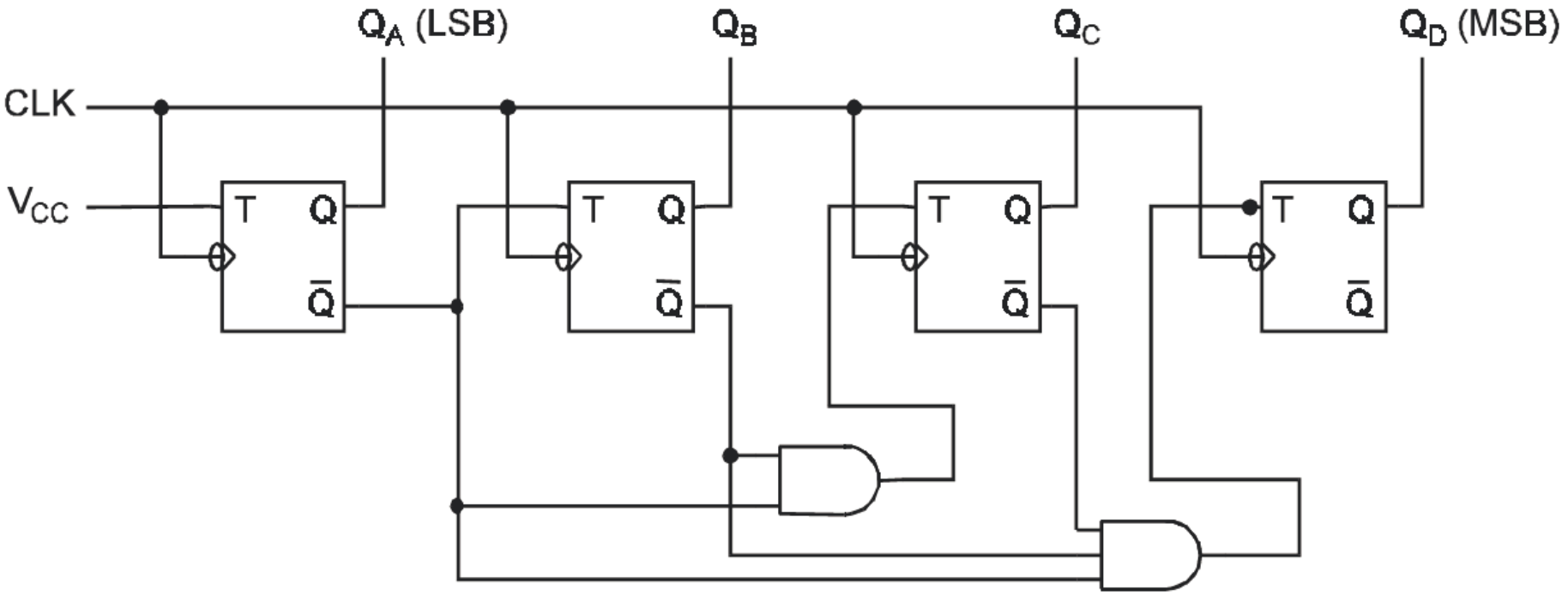
Clock cycle	Q_2	Q_1	Q_0
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1
8	0	0	0



4-bit synchronous up-counter with T flip-flops

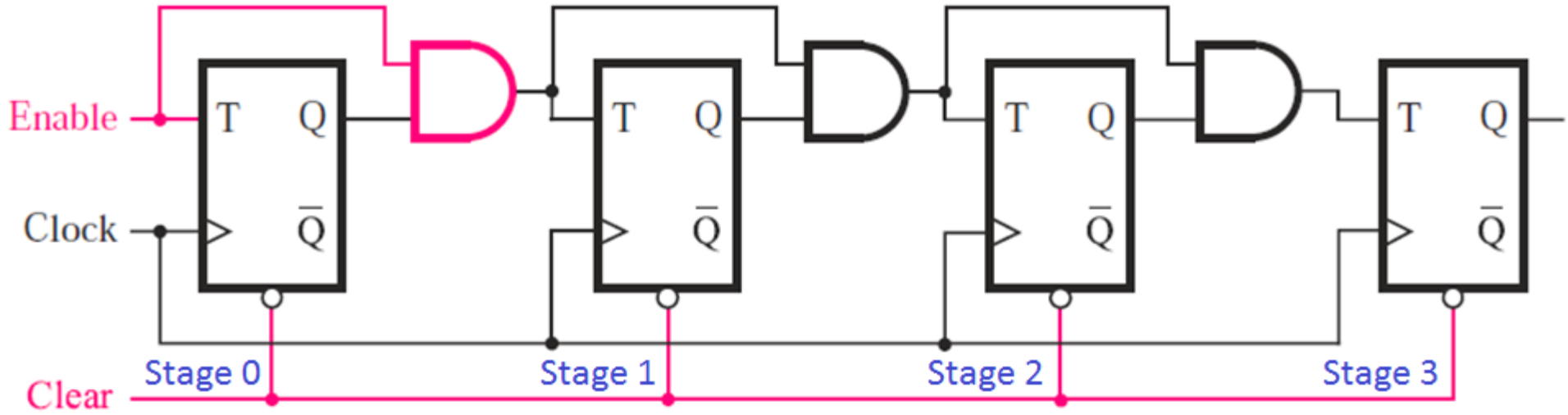


4-bit synchronous down-counter with T flip-flops



Enable and clear capabilities

- 4-bit synchronous up-counter with T flip-flops
  - Clear=0: reset counter output to 0 (clear is asynchronous)
  - T inputs: T_0=Enable, $T_1=Q_0 \cdot$Enable, $T_2=Q_1 \cdot Q_0 \cdot$Enable, $T_3=Q_2 \cdot Q_1 \cdot Q_0 \cdot$Enable
  - Enable=1: normal operation as counter
  - Enable=0: hold current state (all T inputs are 0)



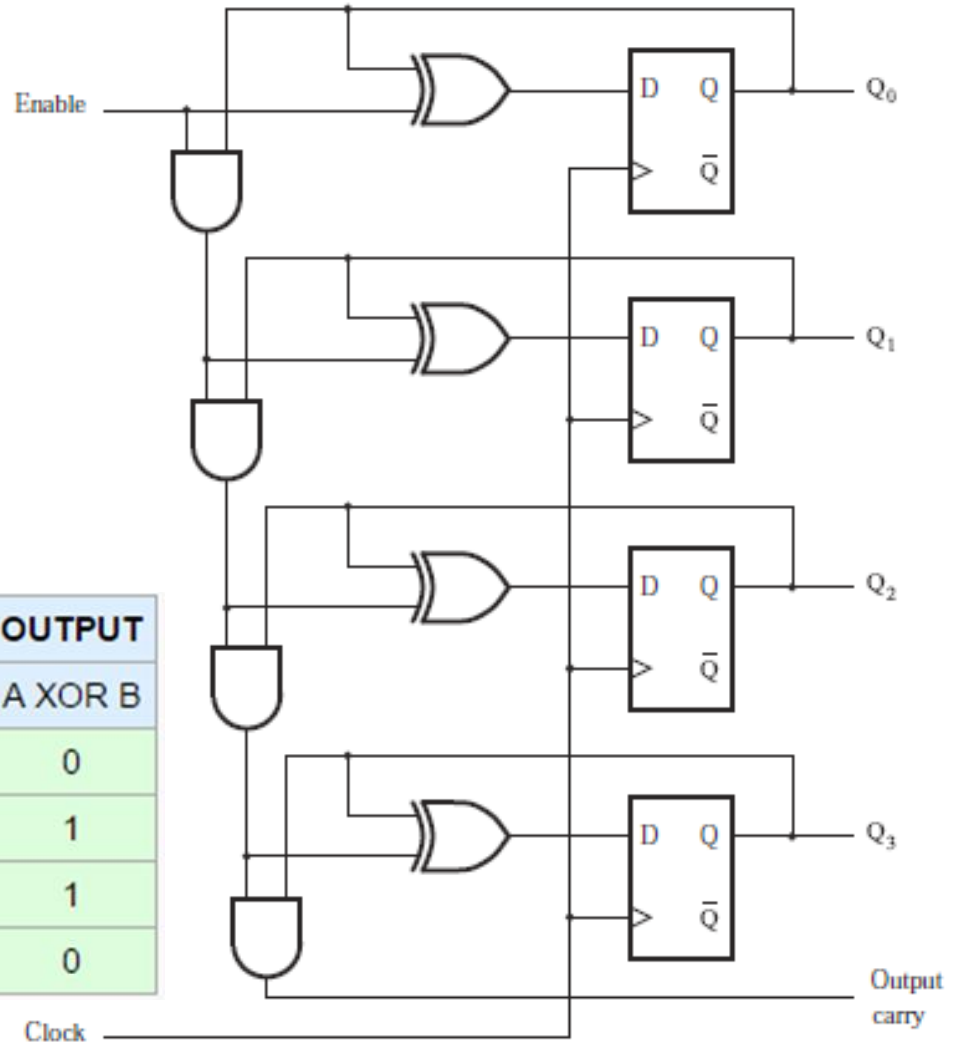
Synchronous counter with D flip-flop

- D input

$$D_0 = Q_0 \oplus Enable$$
$$D_1 = Q_1 \oplus Q_0 \cdot Enable$$
$$D_2 = Q_2 \oplus Q_1 \cdot Q_0 \cdot Enable$$
$$D_3 = Q_3 \oplus Q_2 \cdot Q_1 \cdot Q_0 \cdot Enable$$

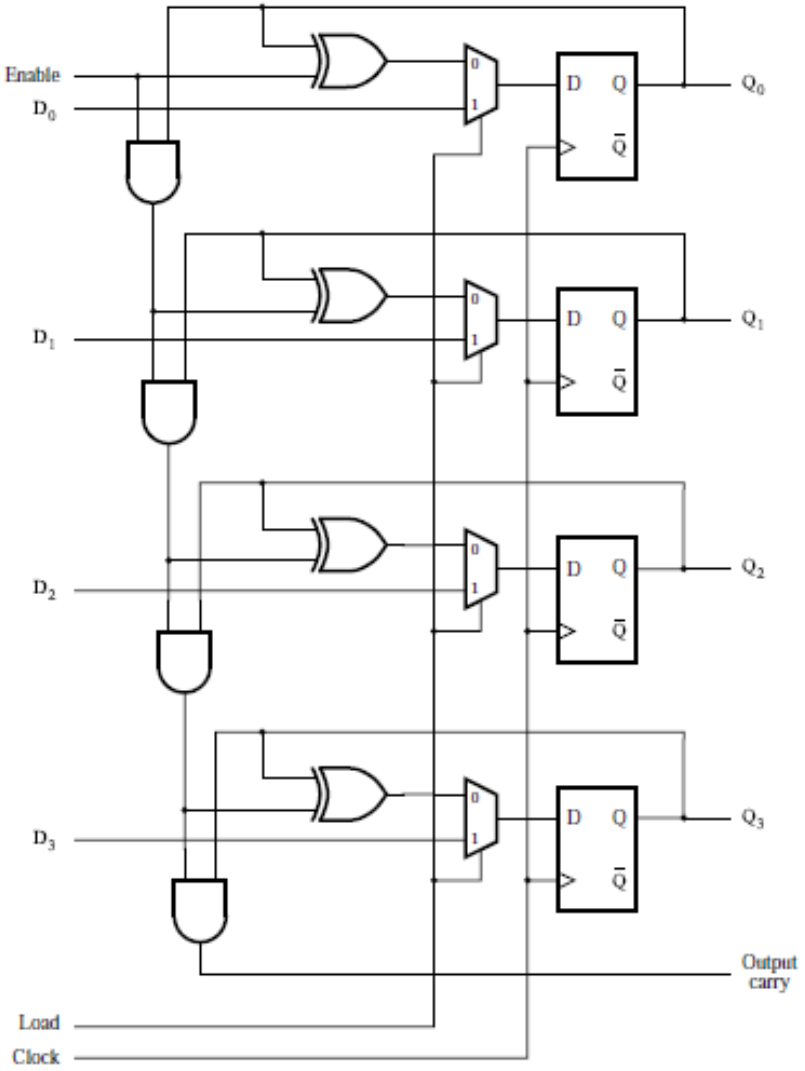
- Operation (stage 0)
  - A=Enable, B=Q_0
  - Enable=1
    - $D_0=Q_0'$, Q_0 flips
  - Enable=0
    - $D_0=Q_0$, Q_0 holds

INPUT		OUTPUT
A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0



Synchronous counter with parallel loading

- 2-to-1 mux
  - Load: mode selection signal
  - Load=0: normal counter
  - Load=1: load $D_3D_2D_1D_0$

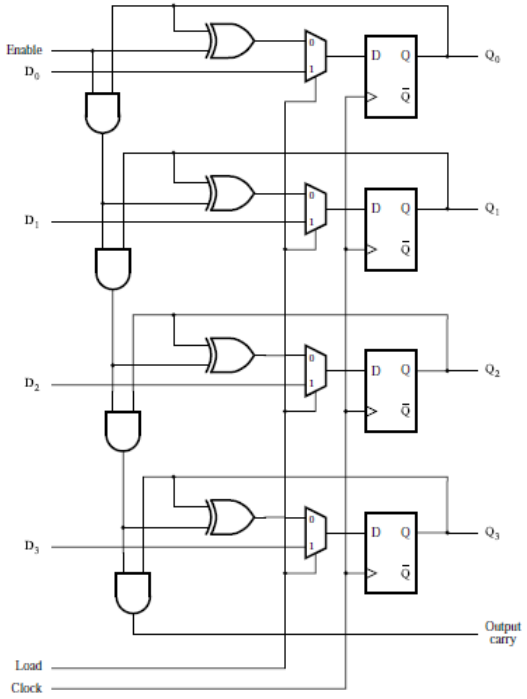


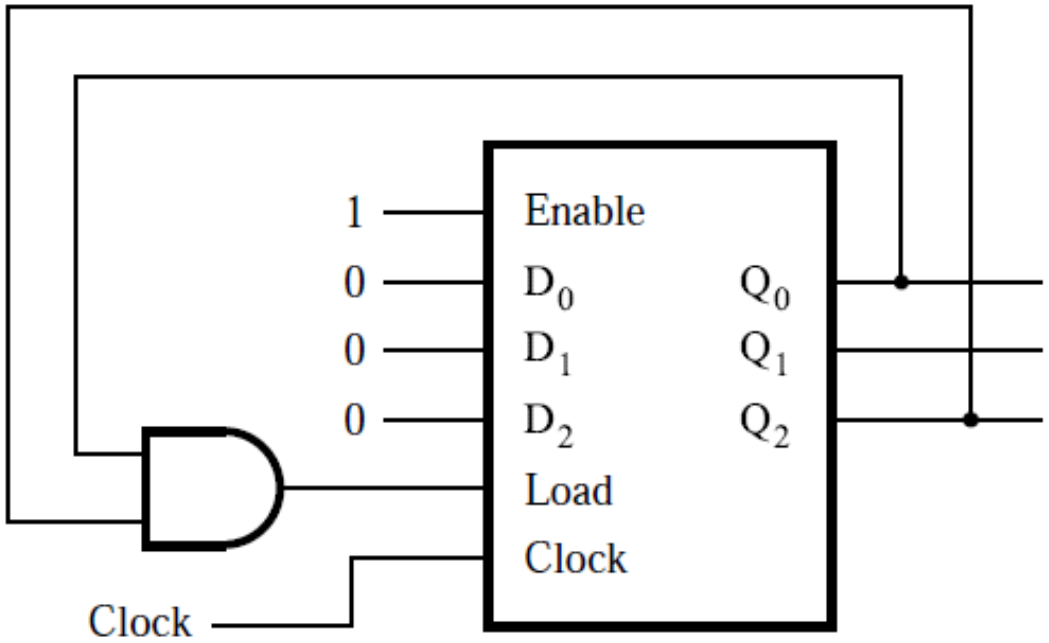
Reset synchronization

- Reset counter during its normal operation
- Modulus (MOD number) of an N-bit counter is NOT always 2^N
- A modulo-6 (MOD-6) counter
  - States (counts): 0, 1, 2, 3, 4, 5
  - Reset is needed when count reaches 5
  - Synchronous reset
  - Asynchronous reset

Synchronous reset

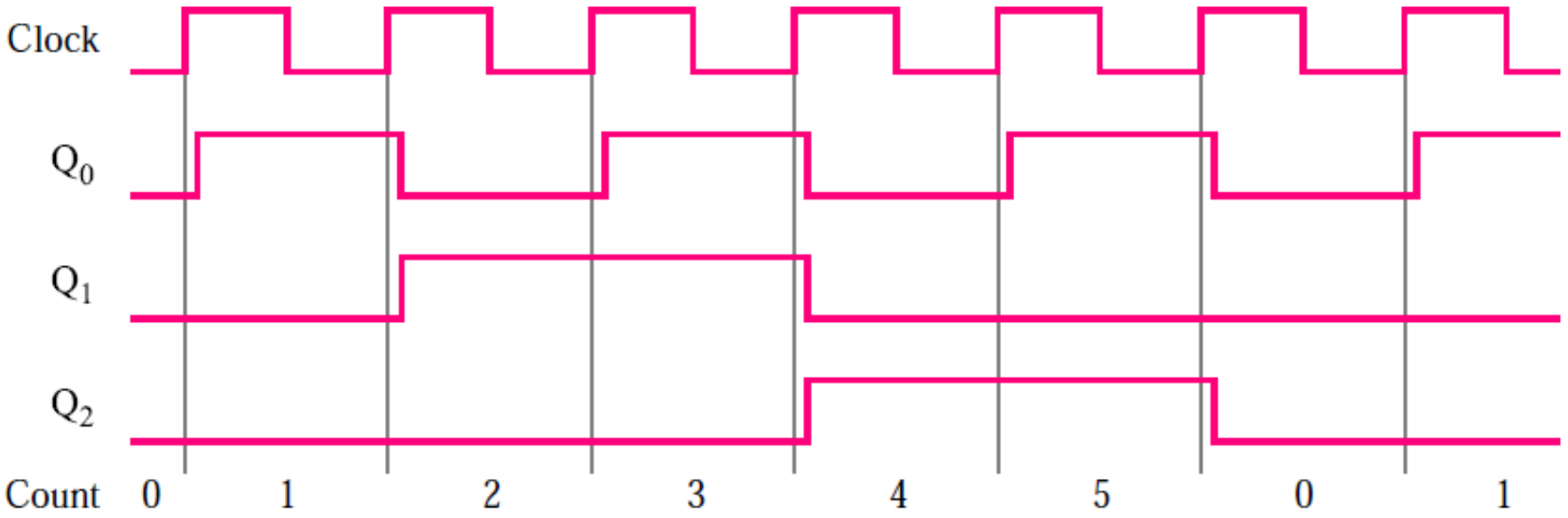
- Reset: implemented using parallel loading mode
- Reset condition: $Q_2Q_1Q_0$=101 (count 5)
- Load=1 when Q_2=Q_0=1: load $D_2D_1D_0$=000 (reset counter)
- Reset is synchronized by clock





Synchronous reset

- Timing diagram

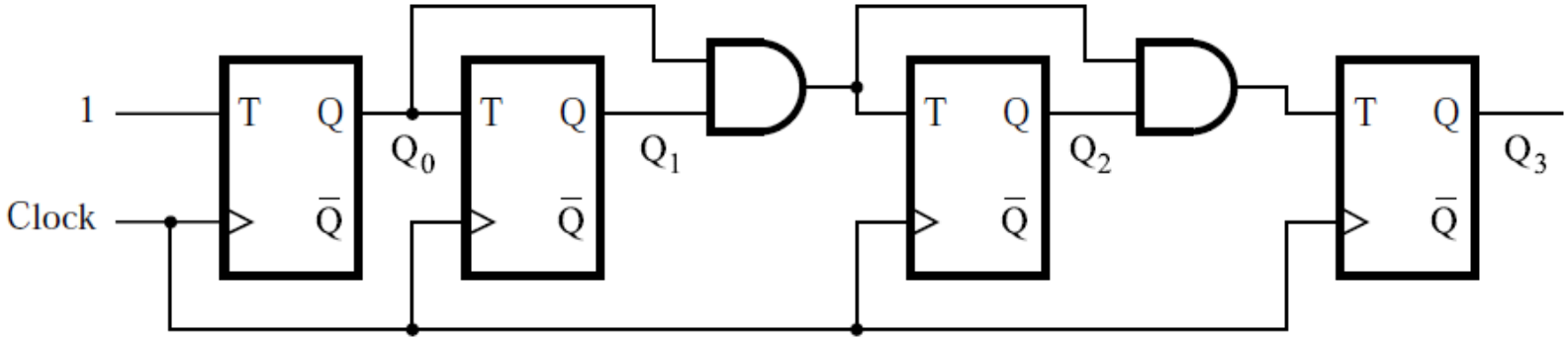


Asynchronous reset

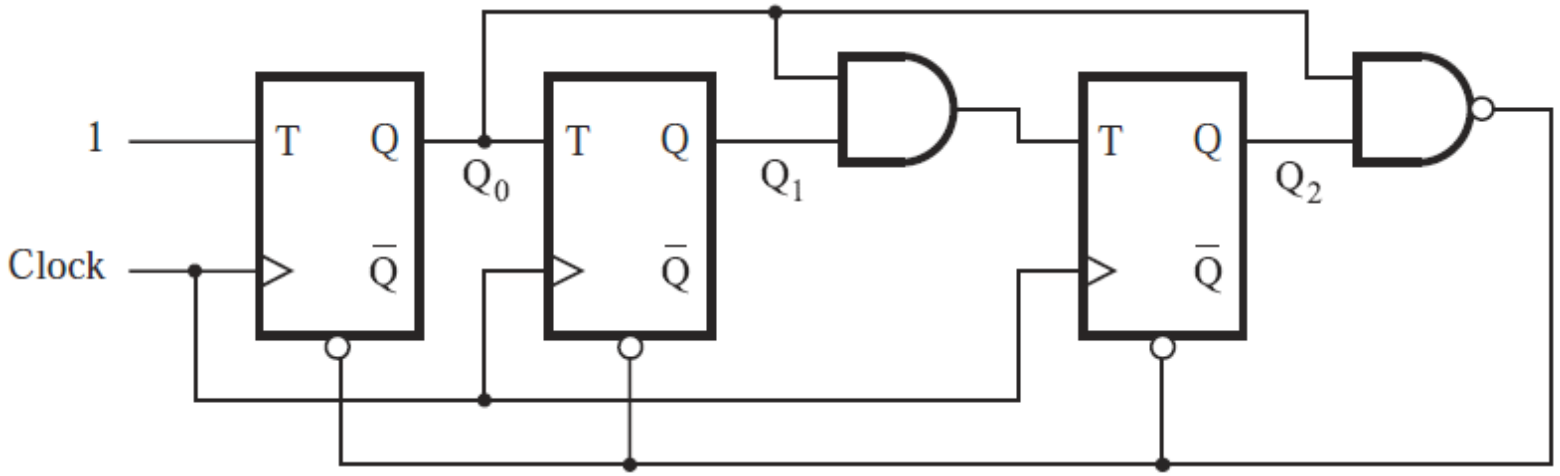
- Reset: implemented using asynchronous clear signal generated by NAND gate
- Reset condition: $Q_2Q_1Q_0$=101 (count 5)
- Circuit modifications
  - NAND gate: use Q_2=Q_0=1 to generate asynchronous clear signal (active low)
  - Asynchronous clear: reset flip-flops
- Reset is NOT synchronized by clock

Asynchronous reset

- Original counter (4-bit synchronous)

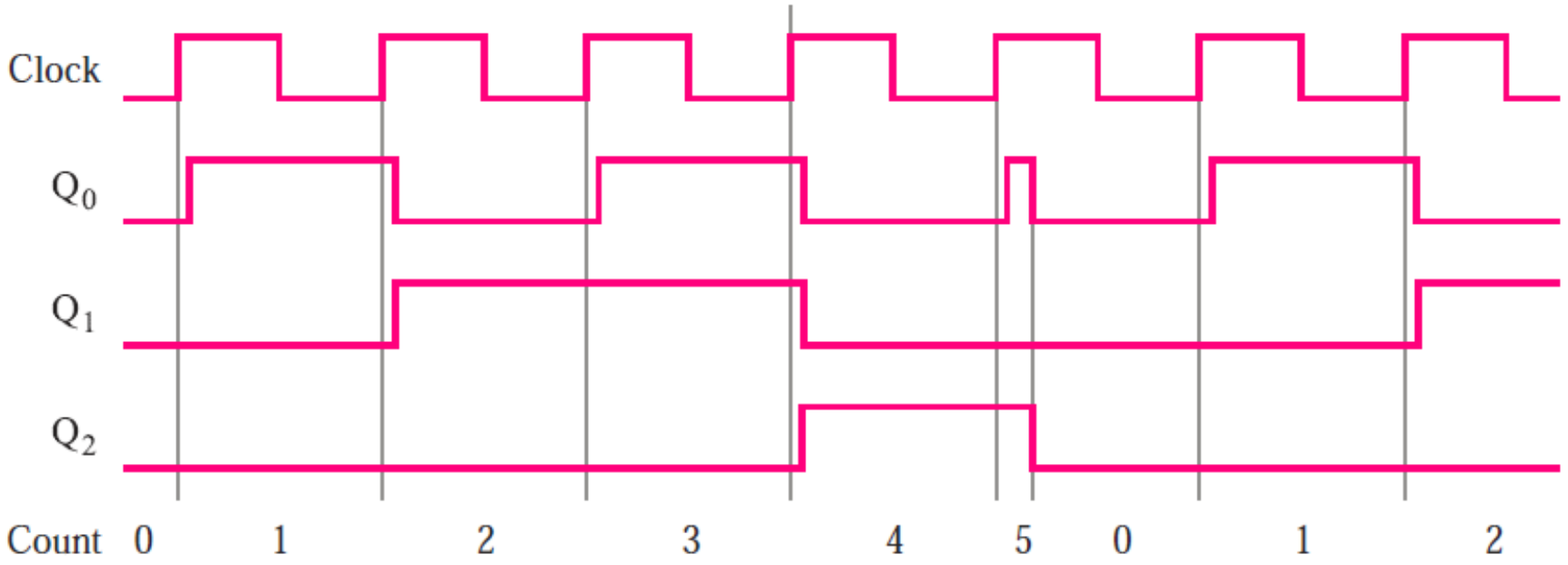


- Modified counter with asynchronous reset (MOD-6)



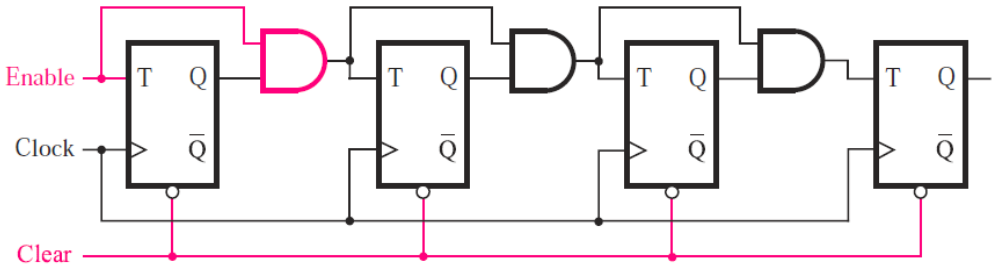
Asynchronous reset

- Timing diagram
  - State “5” duration is shorter than a clock cycle



Code for 4-bit synchronous up-counter with asynchronous reset (clear) and enable

- Ports
  - Resetn: clear
  - E: Enable
- Signal
  - Count: flip-flop outputs




```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;
USE ieee.std_logic_unsigned.all ;

ENTITY upcount IS
    PORT ( Clock, Resetn, E  : IN   STD_LOGIC ;
           Q                 : OUT  STD_LOGIC_VECTOR (3 DOWNTO 0)) ;
END upcount ;

ARCHITECTURE Behavior OF upcount IS
    SIGNAL Count : STD_LOGIC_VECTOR (3 DOWNTO 0) ;
BEGIN
    PROCESS ( Clock, Resetn )
    BEGIN
        IF Resetn = '0' THEN
            Count <= "0000" ;
        ELSIF (Clock'EVENT AND Clock = '1') THEN
            IF E = '1' THEN
                Count <= Count + 1 ;
            ELSE
                Count <= Count ;
            END IF ;
        END IF ;
    END PROCESS ;
    Q <= Count ;
END Behavior ;
```

Code for 4-bit synchronous up-counter with asynchronous reset and parallel loading

- L: load
- R: parallel data
- Integer range: 4-bit

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;
ENTITY upcount IS
    PORT ( R              : IN     INTEGER RANGE 0 TO 15 ;
           Clock, Resetn, L : IN     STD_LOGIC ;
           Q              : BUFFER INTEGER RANGE 0 TO 15 ) ;
END upcount ;

ARCHITECTURE Behavior OF upcount IS
BEGIN
    PROCESS ( Clock, Resetn )
    BEGIN
        IF Resetn = '0' THEN
            Q <= 0 ;
        ELSIF (Clock'EVENT AND Clock = '1') THEN
            IF L = '1' THEN
                Q <= R ;
            ELSE
                Q <= Q + 1 ;
            END IF;
        END IF;
    END PROCESS;
END Behavior;
```

Code for 3-bit down-counter

- Modulus: generic
- E: enable
- L: load
- Initial count: modulus-1

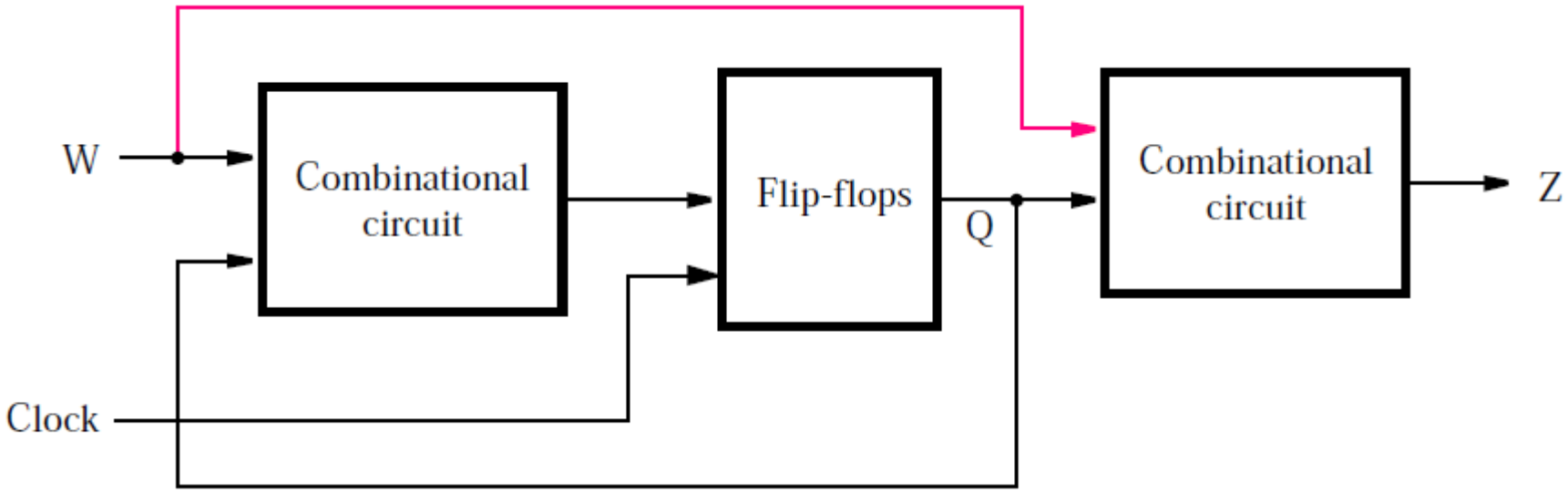
```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

ENTITY downcnt IS
    GENERIC ( modulus : INTEGER := 8 ) ;
    PORT ( Clock, L, E  : IN     STD_LOGIC ;
           Q            : OUT  INTEGER RANGE 0 TO modulus-1 ) ;
END downcnt ;

ARCHITECTURE Behavior OF downcnt IS
    SIGNAL Count : INTEGER RANGE 0 TO modulus-1 ;
BEGIN
    PROCESS
    BEGIN
        WAIT UNTIL (Clock'EVENT AND Clock = '1') ;
        IF L = '1' THEN
            Count <= modulus-1 ;
        ELSE
            IF E = '1' THEN
                Count <= Count-1 ;
            END IF ;
        END IF ;
    END PROCESS;
    Q <= Count ;
END Behavior ;
```

Chapter 8 Synchronous sequential circuits

- Sequential circuits
  - Also called finite state machines (FSMs)
  - Output depends on past behavior and present input

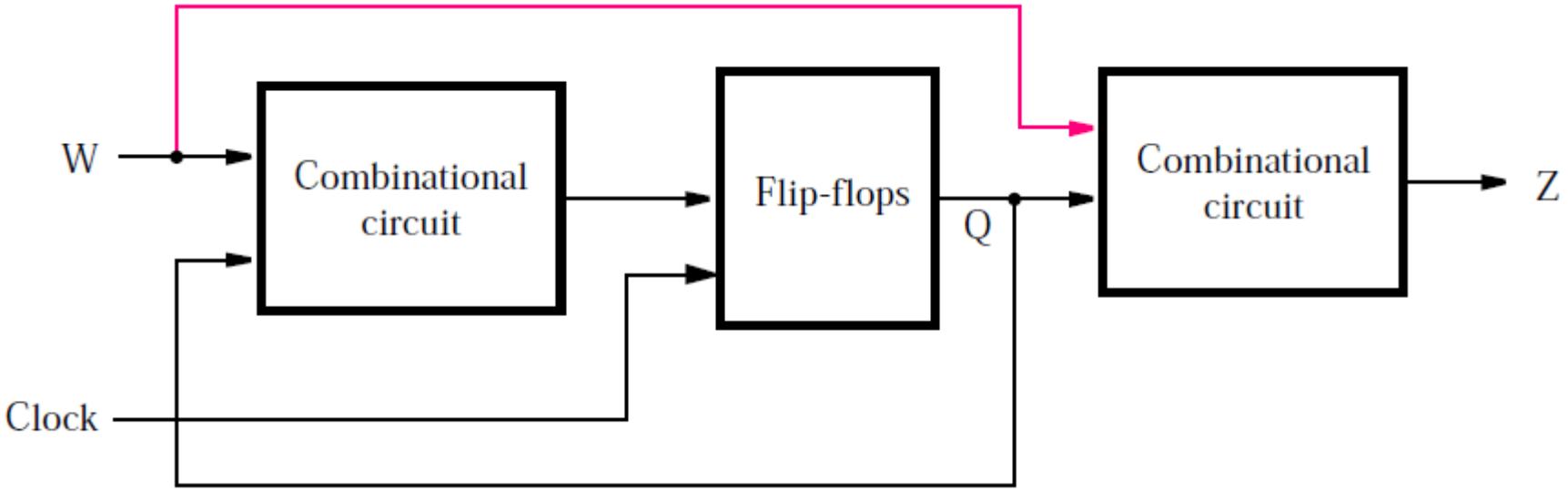
Sequential circuits

- General form
  - Using combinational circuits and flip-flops
  - Input: W, output: Z, state: Q
  - Active clock edge: edge causing state change



Sequential circuits

- Two types
  - Moore: output depends only on state (without red line)
  - Mealy: output depends on both state and input (with red line)



Synchronous sequential circuits

- Design flow
  - Example: sequence detector
- Moore FSM
- Mealy FSM