ECE 2214

Digital Logic Design

L-22 Registers and Counters

Fall 2022

Outline

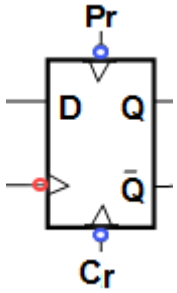
UNB

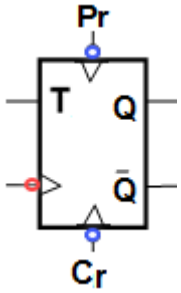
- Shift registers
- Counters and applications
- Asynchronous counter
- Synchronous counter

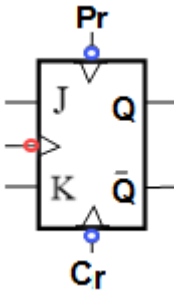
- Information
  - Q4 will take place tomorrow
  - L18 - L22

Summary of previous lecture

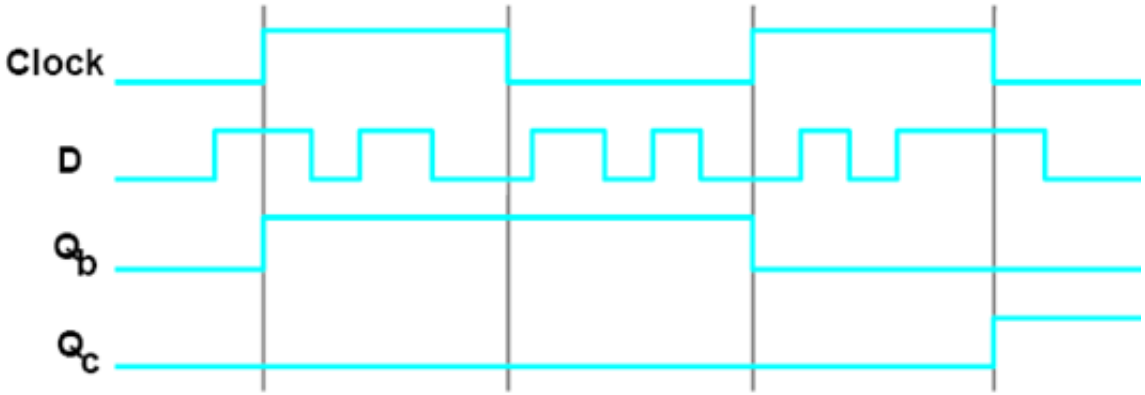
- Introduced edge sensitive flip-flops
  - **Negative edge (D, T, JK)**
  - **Positive edge (D, T, JK)**







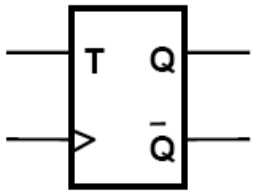
D	CL	Cr	Pr	Q_n	$\overline{Q}_n$
0	↑	1	1	0	1
1	↑	1	1	1	0
x	D	1	1	Q_{n-}	$\overline{Q}_{n-1}$
x	D	1	1	Q_{n-}	$\overline{Q}_{n-1}$
x	↑	0	1	0	1
x	↑	1	0	1	0



From Fundamentals of Digital Logic with VHDL Design 2nd Ed. Stephen Brown, Zvonko Vranesic; McGraw-Hill, 2005

Summary of previous lecture

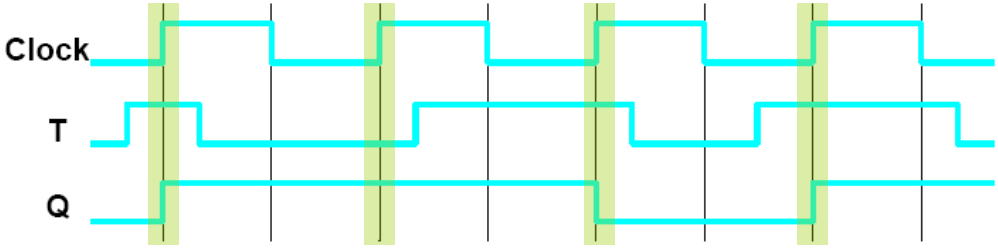
- T flip-flop:
  - Special type of flip-flop that toggles the outputs T



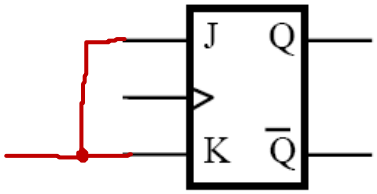
Graphical symbol

T	Q (t + 1)
0	Q (t)
1	Q̄ (t)

Truth table



- J-K flip-flops :
  - Synchronous device
  - General type of flip-flop that can implement T and D flip-flops



CLK	J	K	Q_n	$\overline{Q}_n$
↑	0	0	Q_{n-1}	$\overline{Q}_{n-1}$
↑	0	1	0	1
↑	1	0	1	0
↑	1	1	$\overline{Q}_{n-1}$	Q_{n-1}
0	x	x	Q_{n-1}	$\overline{Q}_{n-1}$
1	x	x	Q_{n-1}	$\overline{Q}_{n-1}$

⇨ D-FF

⇨ T-FF

Summary of previous lecture

UNB

- **Flip-flops in VHDL** : The implementation of flip-flops in VHDL are simple and can be implemented readily with conditional assignment
  - **Use IF statement to check the edge of clock variable**
  - **Use rising_edge(clk) or falling_edge(clock) to assign variables**

Code for a D flip-flop

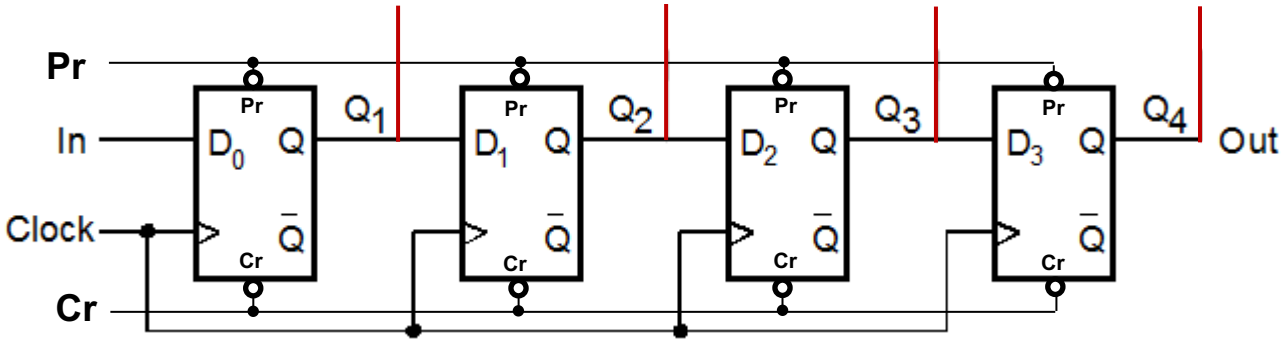
```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

ENTITY flipflop IS
      PORT (   D, Clock  : IN      STD_LOGIC ;
               Q         : OUT     STD_LOGIC) ;
END flipflop ;

ARCHITECTURE Behavior OF flipflop IS
BEGIN
     PROCESS ( Clock,D )
      BEGIN
           IF rising_edge(Clock) THEN
             Q <=D ;
           END IF ;
      END PROCESS ;
END Behavior ;
```

Registers

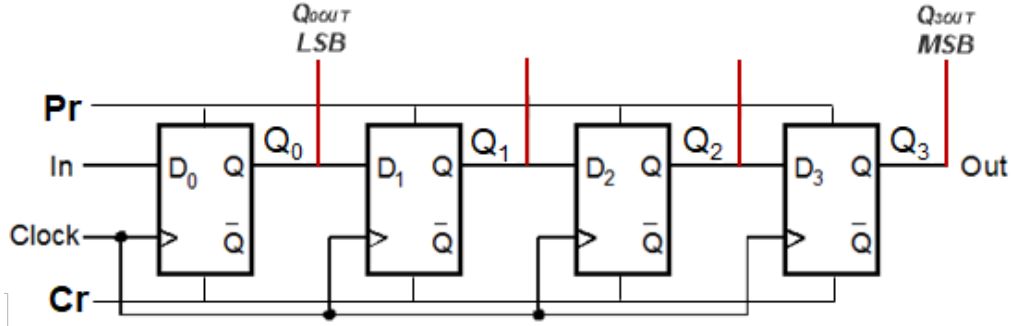
- Registers are a set of **n** flip-flops used to store n bits of information
  - Each flip-flop stores one bit of data
  - Generally, the same clock drives all flip-flops



- Use **Pr='0'** inputs to set bits to "1"
- Use **Cr='0'** inputs to set bits to "0"
- Use **clock** to memorize bits in the flip-flops
- Input **In** used to enter data serially to the register
- Use Cr & Pr independently to load a particular 4-bit number

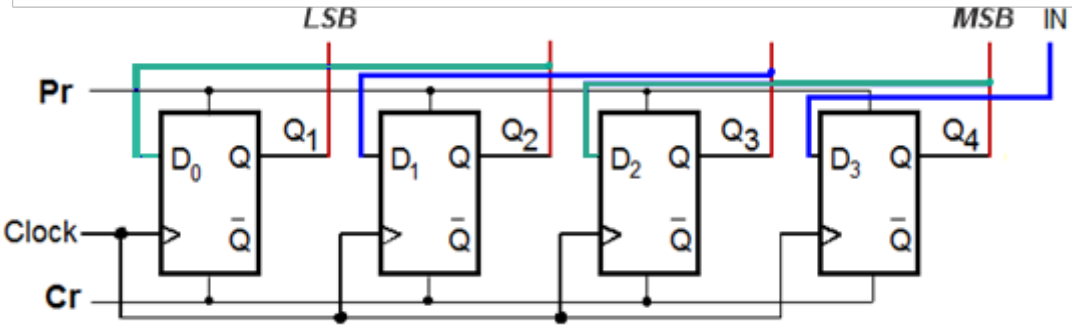
Implementation of shift registers

- **Shift register:** a register that has the ability to shift stored bits based on a clock signal
  - Shift registers are very important for arithmetic operations, especially for multiplication and division.
  - For instance:
    - A binary number is multiplied by a power of 2 if it is shifted to the left, inserting zeros at the LSB
    - A binary number is divided by a power of 2 if it is shifted to the right inserting '0' at the MSB



Shift 1 bit to the left

0000

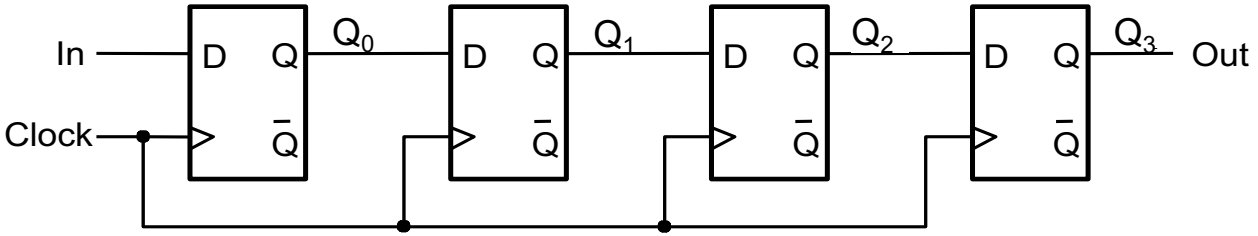


Shift 1 bit to the right

0000

Registers

- **Serial input/serial output register:**
  - Inputs are connected to output of previous FF to allow bit shifting
  - Clocks are connected together to synchronize shifts
  - Q_3 provides a serial output of the shifted bits

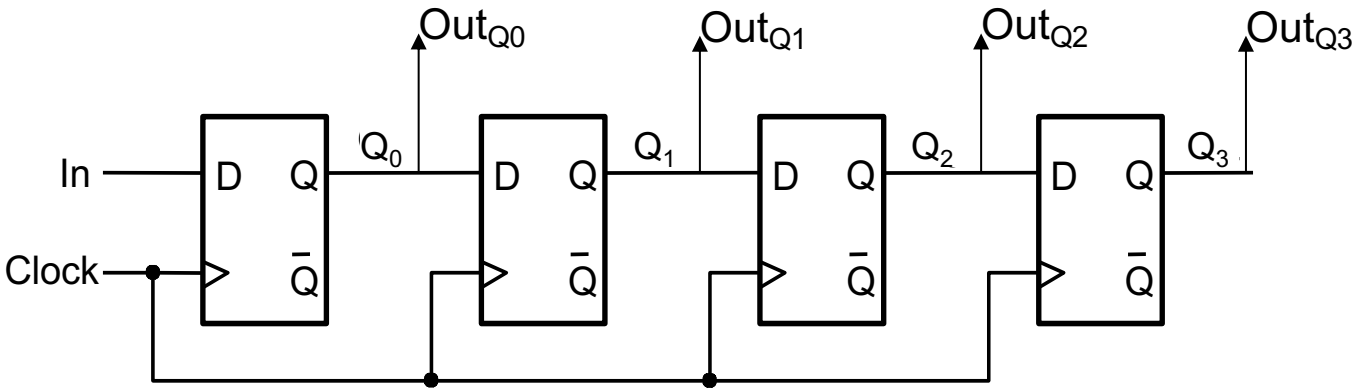


Circuit

	In	Q_0	Q_1	Q_2	Q_3 = Out
t_0	1	0	0	0	0
t_1	0	1	0	0	0
t_2	1	0	1	0	0
t_3	1	1	0	1	0
t_4	1	1	1	0	1
t_5	0	1	1	1	0
t_6	0	0	1	1	1
t_7	0	0	0	1	1

But… Computer need to access results at once (in parallel)

Serial input/parallel output shift register



Circuit

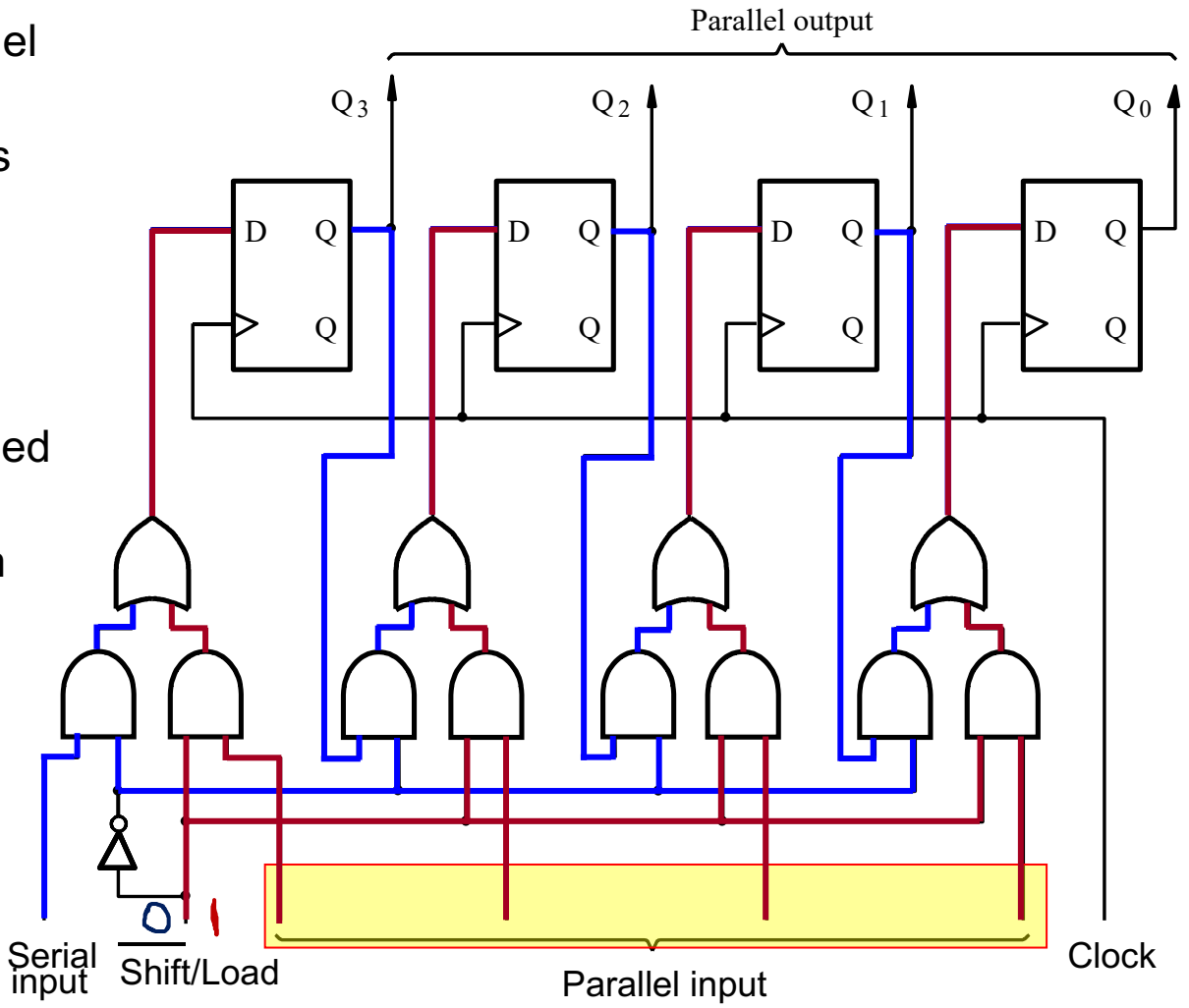
	In	Q_0	Q_1	Q_2	Q_3
t_0	1	0	0	0	0
t_1	0	1	0	0	0
t_2	1	0	1	0	0
t_3	1	1	0	1	0
t_4	1	1	1	0	1
t_5	0	1	1	1	0
t_6	0	0	1	1	1
t_7	0	0	0	1	1

NOTE:

- Similar configuration to the serial version but outputs are obtained at once
- All clocks are connected together to synchronize shifts
- Computer can access values stored in the register in parallel

Shift register with parallel load

- External circuit allows parallel load of external bits
- Parallel load used to set bits individually
- Control signal (Shift/Load) controls shift or load operations
- Output of each bit is accessed directly in parallel form
- A serial out is obtained from output Q_0



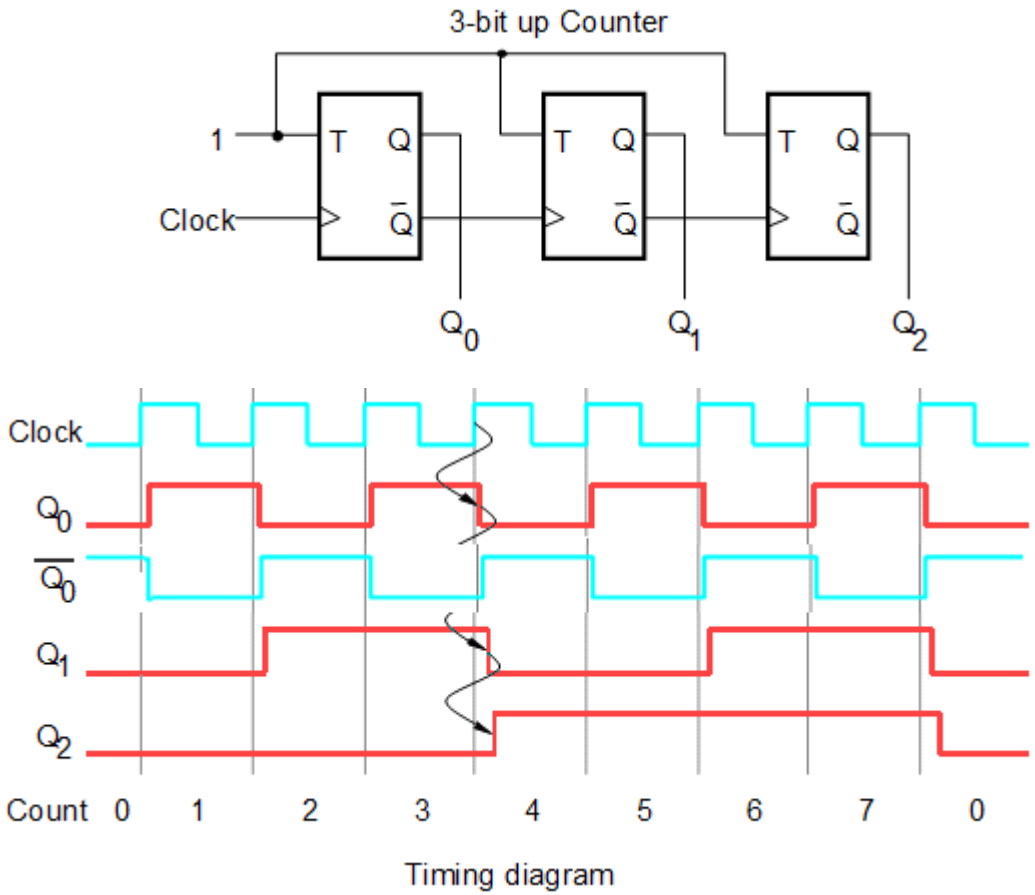
Counters

UNB

- Important to perform arithmetic operations such as addition and subtraction
  - Counters perform a special type of addition and subtraction used for counting: counts by one or a fixed magnitude
- Applications of counters
  - Many applications such as:
    - Count the number of occurrences of certain event
    - Generate timing signals to control different system tasks
    - Keep track of time elapsed between events, etc.
- Implementation of counters
  - Counters can be asynchronous and synchronous
  - The simplest ways of implementing a counter with F-Fs is:
    1. Using T flip-flops to toggle the output of the FF when the input variable meet the design condition
    2. Using D flip-flops to detect clock edge events

Asynchronous counter

- Asynchronous counter: different clock signal for each bit (FF) so…
  - Each output do not change at the same time
- **Up-counter with T flip-flops**

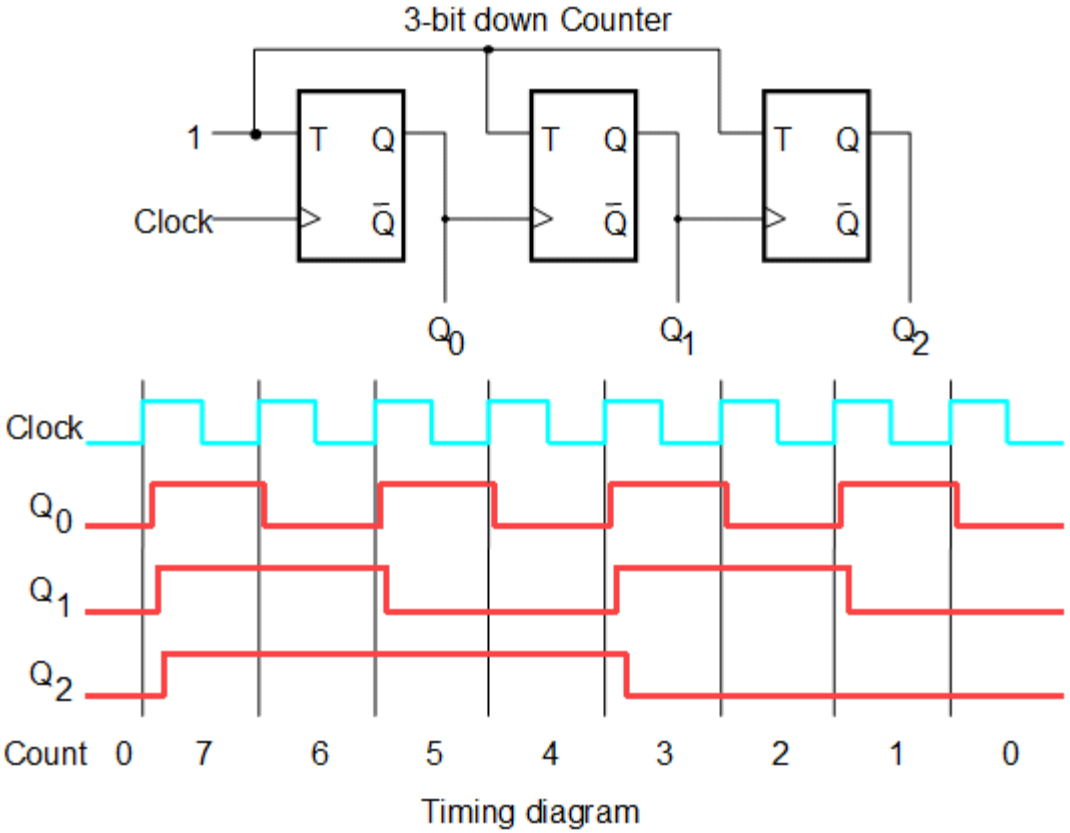


Timing diagram

- Objective is to count the pulses of the input clock
- F-F are connected in cascade to provide consecutive bits
- T = 1 toggled FF output at CLK↑ That's why CLK_1 = clock
- $CLK_2 = CLK_3 = \overline{Q}$ enable next-digit count event after previous digit counted the event and count up consecutively
- Q_1 and Q_2 will change with CLK ↓
- Q_0 is the LSB and Q_2 is the MSB
- Binary number is $\mathbf{Q_2Q_1\ Q_0}$
- Clock cascade introduces a propag. delay. **Drawback of asynchronous counting**

Asynchronous counter

- **Down-counter with T flip-flops**



- Same configuration as the up counter
- $CLK_2 = CLK_3 = Q$ now to enable down counting
- Q_1 and Q_2 will change with CLK ↑ since are connected to Q of previous digit
- First state is “111” (max counting number)
- Suffer from the same drawback of asynchronous counting
- It is possible to combine this circuit with previous to form a up/down counter!

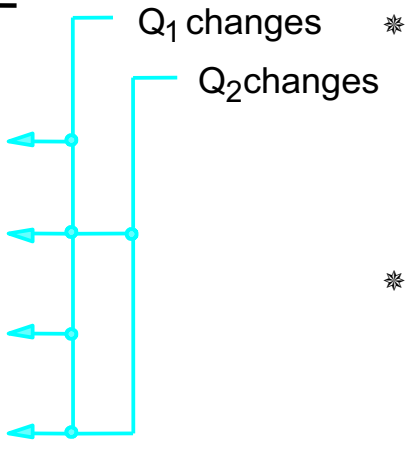
Limitation:

- Delay increase with the number of bits
- Not suitable for strictly time-constrained applications

Synchronous counter

- A more time-constrained implementation
- **Synchronous up-counter with T flip-flops:**
  - All clocks are set together to obtain synchronization
  - The inputs of the FFs have to be combined to obtain the counting sequence
- Derivation of counter equations :

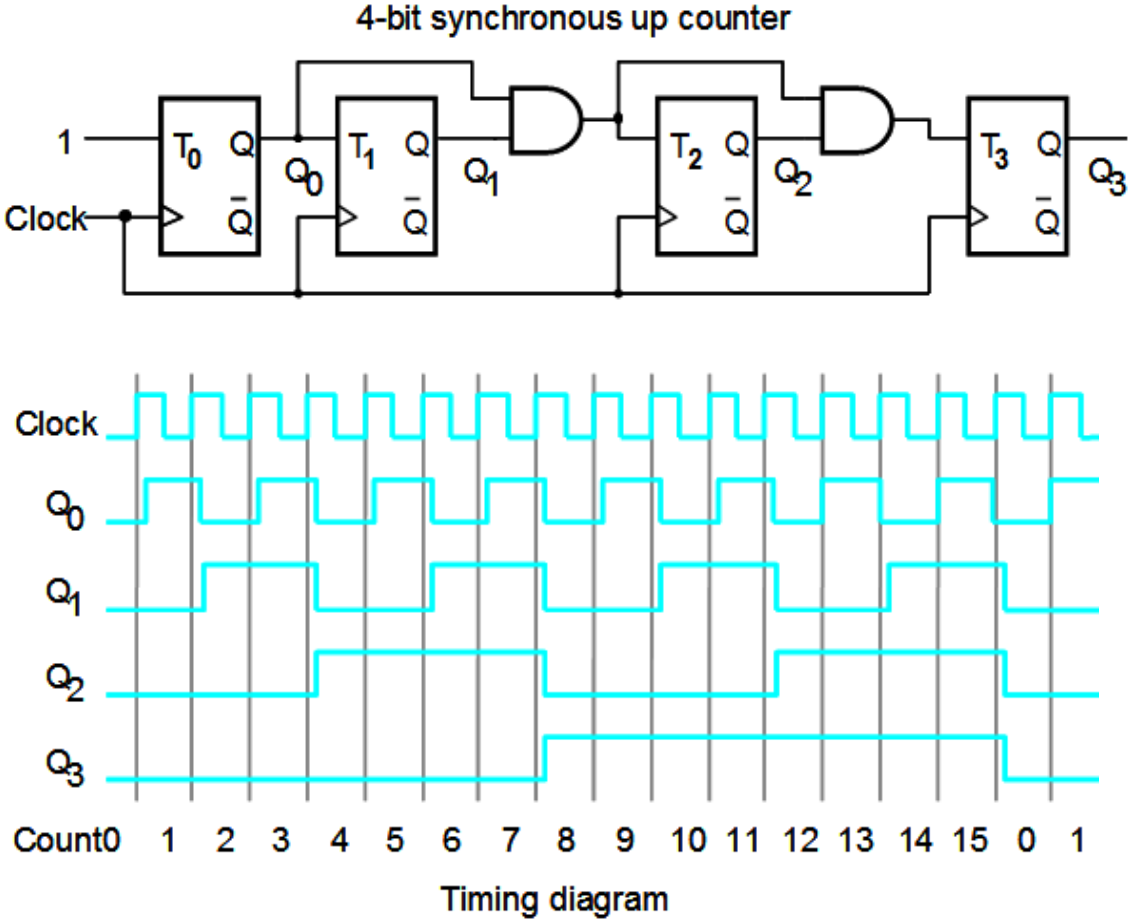
Clock cycle	Q_2	Q_1	Q_0
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1
8	0	0	0



- The table represents the output of a 3-bit counter in the first 8 clock cycles
- Assume a starting state at '000':
  - Q_0 changes in all states
  - Q_1 will only change after $Q_0 = 1$
  - Q_2 will only change after $Q_1 = 1$ and $Q_0 = 1$
- Implementing counter with T F-Fs require the following equations for the T inputs:
  - $T_0 = 1$ (assumes initial T=1)
  - $T_1 = Q_0$ (green column, out of FF_0)
  - $T_2 = Q_0Q_1$ (yellow column , out of FF_1)

- Equations imply that all output of FFs have to be combined to reach counting sequence

4-bit counter implementation synchronous counter



- All clocks are connected to the same signal
- Note that:
  - $T_0 = 1$
  - $T_1 = Q_0$
  - $T_2 = Q_0Q_1$
  - $T_3 = Q_0Q_1Q_2$
- The delay introduced at any stage is constant after the active edge of the clock signal
- The delays introduced do not depend on the number of bits but it is determined by the delay introduced by the F-F used

- **Notice that the output of a counter divide the frequency of the clock by a power of 2!**

Generalizing to n-bit up synchronous counters

UNB

- Generalizing for n-bit up counters:
  - A given F-F changes its states when all previous outputs are in state 1
  - Using T F-Fs to implement the counter, the following equations apply:

$T_0 = 1$ (assumes initial T=1)

$T_1 = Q_0$

$T_2 = Q_0Q_1$

$T_3 = Q_0Q_1Q_2$

…

…

$T_n = Q_0Q_1 \ldots Q_{n-1}$

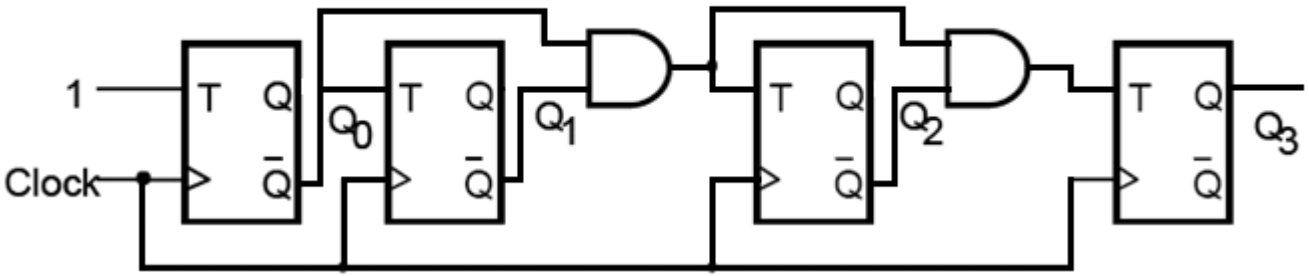
- All F-Fs outputs have to be combined to enable the following bit

4-bit down counter implementation synchronous counter

- The complemented outputs of the FF used to enable the next bit

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	0	0	1	
0	0	0	0	

4-bit synchronous down counter



- Implementing down counter with T F-Fs require the following equations for the T inputs:
  - $T_0 = 1$ (assumes initial T=1)
  - $T_1 = \overline{Q_0}$
  - $T_2 = \overline{Q_0}\,\overline{Q_1}$
  - $T_3 = \overline{Q_0}\,\overline{Q_1}\,\overline{Q_2}$

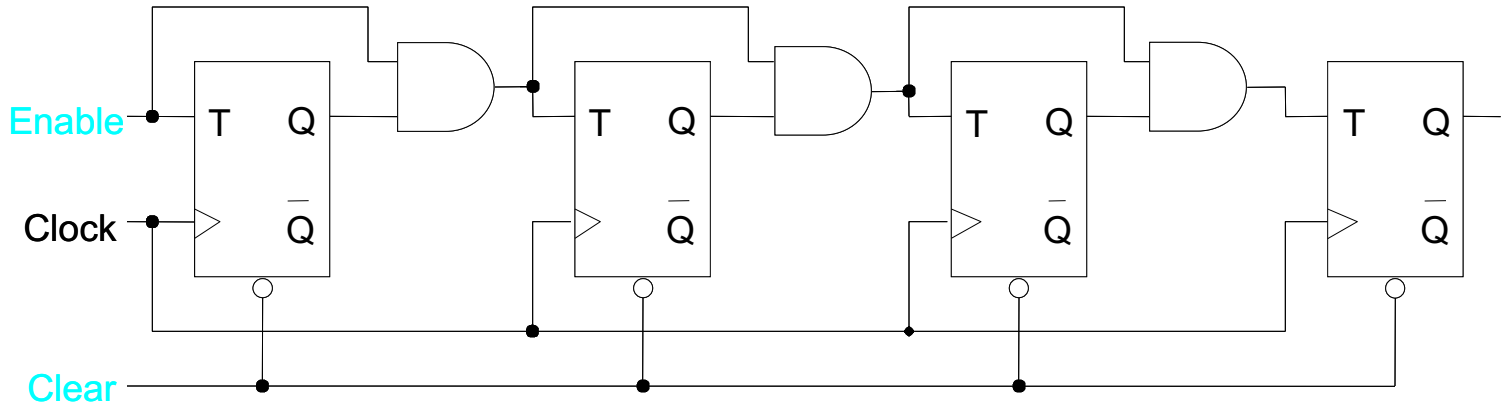
Enable and clear capabilities on counters

UNB

- **Enable:** Allows inhibiting the counter and holding the counting sequence
  - Easily achieved by setting the T inputs of the F-Fs to ‘0’
  - Freeze the counter at certain value until the data is read or the application performs a decision

- **Clear:** Allows resetting the counter where all bits will be set to 0
  - Achieved activating the clear terminal of all F-Fs
  - Allows changing the maximum counting value:
    - If additional logic is added to monitor the occurrence of a given number sequence
    - The logic activate the clear and resets counter outputs to 0

Enable and clear capabilities on counters



- Enable = 0, set all T inputs to 0 and current counting sequence is frozen regardless any clock change
- Clear = 0, all Q outputs are set to 0 and current counting sequence is restarted
  - If FF has asynchronous reset
    - the change takes affect immediately
  - If FF has synchronous reset
    - the change won't take affect until next positive edge of the clock
- Preset = 0, preset to the maximum counting sequence

Summary/references

UNB

- Shift registers
- Counters and applications
- Asynchronous and Synchronous counter
- Enable and reset features of counters

- Recommended exercises from the book
  1. Problems of pp. 473 of textbook (7.13-7.22)

- References
  - Fundamentals of Digital Logic with VHDL Design 2nd Edition Stephen Brown, Zvonko Vranesic; McGraw-Hill, 2005. Chapter 7, pp. 398-406.
  - Digital Design; Principles and Practices. Fourth Edition. John F. Wakerly, Prentice Hall, 2006. ISBN 0-13-186389-4. Chapter 7, pp. 710-720, 727-752.

- Next Lecture
  - More on counters
  - Fundamentals of Digital Logic with VHDL Design 3/e, Stephen Brown, Zvonko Vranesic; McGraw-Hill, 2009. Chapter 7, pp. 406-415.
  - Digital Design; Principles and Practices. Fourth Edition. John F. Wakerly, Prentice Hall, 2006. ISBN 0-13-186389-4. Chapter 7, pp. 710-727.