Lecture 11: Serial Adder & Counter

CS207: Digital Logic

Jialin Liu

Department of Computer Science and Engineering (CSE)
Southern University of Science and Technology (SUSTech)

16 December 2022

Acknowledgement

These slides were prepared based on the slides by Dr. Jianqiao Yu and the ones by Prof. Georgios Theodoropoulos of the Department of CSE at the SUSTech, as well as the contents of the following book:

M. M. Mano and M. Ciletti, *Digital design: with an introduction to the Verilog HDL*. Pearson, 2013

A. Saha and N. Manna, *Digital principles and logic design*. Jones & Bartlett Learning, 2009

Outline of This Lecture

Serial Addition

- Serial Adder
- Design A Serial Operation

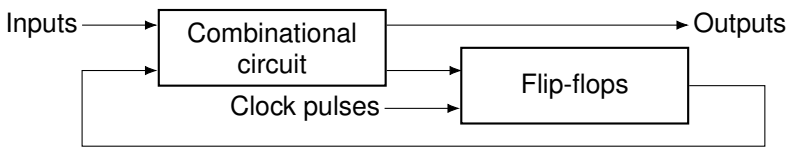
Counters

- Asynchronous Counters
- Synchronous Counters

Summary

Recap: Clocked Sequential Circuits

- Clocked sequential circuits have flip-flops (FF) and combinational gates.
  - Flip-flops are essential, otherwise reduce to combinational.
  - Circuits that include flip-flops are usually classified by the function they perform rather than by the name of the sequential circuit.
  - Two such circuits are **registers** (寄存器) and **counters** (计数器).



Recap: Commonly Used Circuits That Include Flip-flops

- **Register**:
  - A group of flip-flops, each one of which
    - shares a common clock and
    - is capable of storing one bit of binary information.
  - A flip-flop stores one bit of binary information.
    → An n-bit register consists of a group of n flip-flops capable of storing n bits of binary information.
  - In addition to the flip-flops, a register may have combinational gates that perform certain data-processing tasks.

- **Counter**:
  - A special type of register that goes through a predetermined sequence of binary states.

Outline of This Lecture

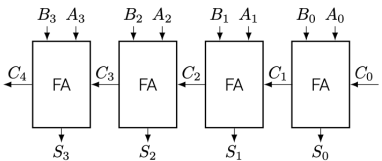
- Operations in digital computers are usually done in parallel because that is a faster mode of operation.
- But serial operations have the advantage of requiring fewer hardware components.

  → As an example, let's look at the design of a **serial adder** (串行加法器) using
  - two shift registers,
  - a full adder, and
  - a D flip-flop.

Outline

Recap: Binary Adder

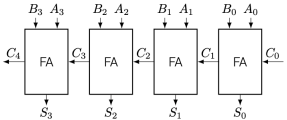
- A **binary adder** is a digital circuit that produces the arithmetic sum of two binary numbers.
- It can be constructed with full adders connected in cascade, i.e., **ripple-carry-adder** (进位传送加法器/串行进位加法器), with the output carry from each full adder connected to the input carry of the next full adder in the chain.
- Addition of n-bit numbers requires a chain of n full adders or a chain of 1 half adder and $n-1$ full adders.
  - Below shows the interconnection of four 4 adder (FA) circuits to provide a 4-bit binary ripple-carry-adder.



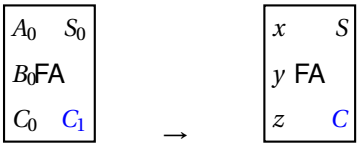
Serial Addition (串行加法器)

- Design a **serial adder** (串行加法器) using
  - The two binary numbers to be added serially are stored in two shift registers.
  - Beginning with the least significant pair of bits, the circuit adds one pair of bits at a time through a single full-adder (FA) circuit.
  - The carry out of the full adder is transferred to a D flip-flop, the output of which is then used as the carry input for the next pair of significant bits.

Ripple-carry-adder:

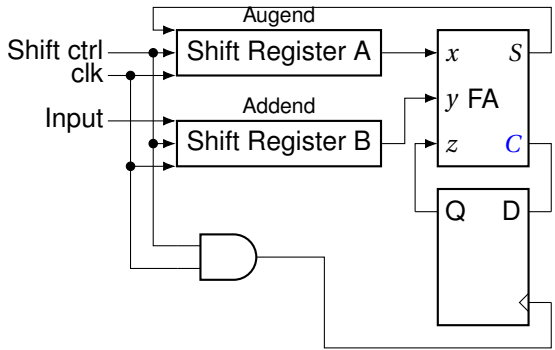


Let's change the notation:



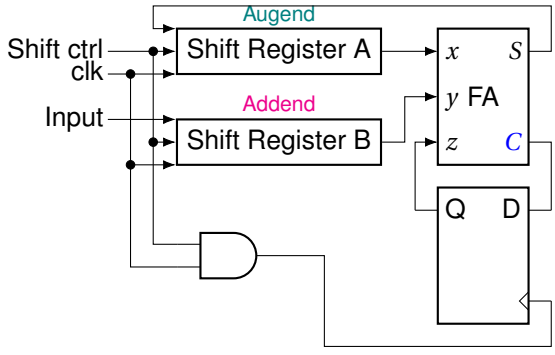
Serial Addition (串行加法器)

- The two binary numbers to be added serially are stored in two shift registers.
- Beginning with the least significant pair of bits, the circuit adds one pair of bits at a time through a single full-adder (FA) circuit.
- The carry out of the full adder is transferred to a D flip-flop, the output of which is then used as the carry input for the next pair of significant bits.



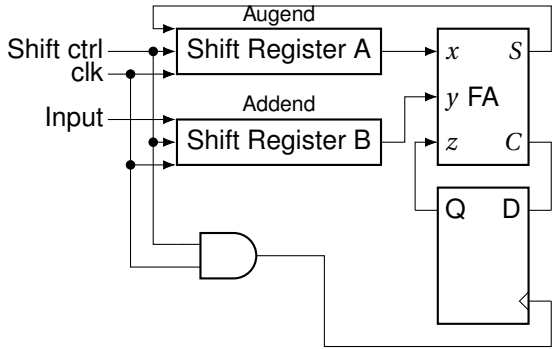
Serial Addition

- Initially, register A holds the augend, register B holds the addend, and the carry flip-flop is cleared to 0.
  - The outputs of A and B provide a pair of least significant bits for the full adder at x and y.
  - Output Q of the flip-flop provides the input carry at z.
- At the next clock pulse, both registers are shifted once to the right, the sum bit from S enters the leftmost flip-flop of A, and the output carry is transferred into flip-flop Q.



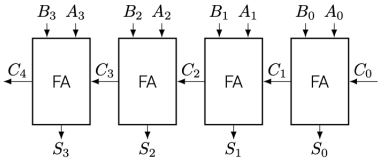
Serial Addition

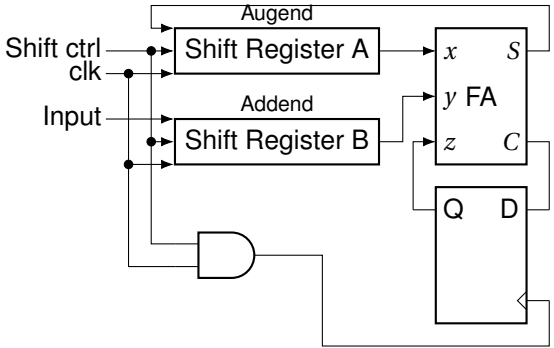
- For each succeeding clock pulse, a new sum bit is transferred to A, a new carry is transferred to Q, and both registers are shifted once to the right.
- This process continues until the shift control is disabled.
- The addition is accomplished by passing each pair of bits together with the previous carry through a single full-adder circuit and transferring the sum, one bit at a time, into register A.



Serial Addition

- Difference between the serial adder and the ripple-carry-adder:
  - The number of full-adder circuits in the ripple-carry-adder is equal to the number of bits in the binary numbers, whereas the serial adder requires only one full-adder circuit and a carry flip-flop.





Outline

Design A Serial Operation

- We will redesign the serial adder with the use of a state table.
- We assume that two shift registers are available to store the binary numbers to be added serially.
  - The serial outputs from the registers are designated by x and y.
- The sequential circuit has
  - two inputs, x and y, that provide a pair of significant bits, ← two registers
  - an output S that generates the sum bit, ← a binary adder
  - and flip-flop Q for storing the carry. ← a memory element

Design A Serial Operation

- The sequential circuit has the two inputs, x and y, that provide a pair of significant bits, an output S that generates the sum bit, and flip-flop Q for storing the carry.

Present State (Carry) Q_t	Inputs x	y	Next State (Carry) Q_{t+1}	Output S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Recap: Characteristic Equations of Flip-flops (Lecture 7)

- A characteristic equation describes the logical properties of a flip-flop by describing its Boolean function.
- DFF: $Q_{t+1} = D$.
- JKFF: $Q_{t+1} = JQ'_t + K'Q_t$.
- TFF: $Q_{t+1} = T \oplus Q_t$.

J	K	Q_{t+1}	
0	0	Q_t	No change
0	1	0	Reset
1	0	1	Set
1	1	Q'_t	Complement

D	Q_{t+1}	
0	0	Reset
1	1	Set

T	Q_{t+1}	
0	Q_t	No change
1	Q'_t	Complement

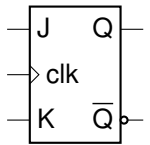
Design A Serial Operation

- The sequential circuit has the two inputs, x and y, that provide a pair of significant bits, an output S that generates the sum bit, and flip-flop Q for storing the carry.
  - If a D flip-flop is used for Q, the circuit reduces to the previous example.

Present State (Carry)	Inputs		Next State (Carry)	Output	FF Input
Q_t	x	y	Q_{t+1}	S	D_Q
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	1	0
0	1	1	1	0	1
1	0	0	0	1	0
1	0	1	1	0	1
1	1	0	1	0	1
1	1	1	1	1	1

Design A Serial Operation

- The sequential circuit has the two inputs, x and y, that provide a pair of significant bits, an output S that generates the sum bit, and flip-flop Q for storing the carry.
  - What if using a JK flip-flop?



J	K	Q_{t+1}	
0	0	Q_t	No change
0	1	0	Reset
1	0	1	Set
1	1	Q_t'	Complement

- JKFF: $Q_{t+1} = JQ_t' + K'Q_t$.

Design A Serial Operation

- JKFF: $Q_{t+1} = JQ_t' + K'Q_t$.

Present State (Carry)	Inputs		Next State (Carry)	Output	FF Inputs	
Q_t	x	y	Q_{t+1}	S	J_Q	K_Q
0	0	0	0	0	0	X
0	0	1	0	1	0	X
0	1	0	0	1	0	X
0	1	1	1	0	1	X
1	0	0	0	1	X	1
1	0	1	1	0	X	0
1	1	0	1	0	X	0
1	1	1	1	1	X	0

Design A Serial Operation

Present State (Carry)	Inputs		Next State (Carry)	Output	FF Inputs	
Q_t	x	y	Q_{t+1}	S	J_Q	K_Q
0	0	0	0	0	0	X
0	0	1	0	1	0	X
0	1	0	0	1	0	X
0	1	1	1	0	1	X
1	0	0	0	1	X	1
1	0	1	1	0	X	0
1	1	0	1	0	X	0
1	1	1	1	1	X	0

$$J_Q = xy,$$
$$K_Q = x'y' = (x+y)'$$
$$S = x \oplus y \oplus Q$$

Outline of This Lecture

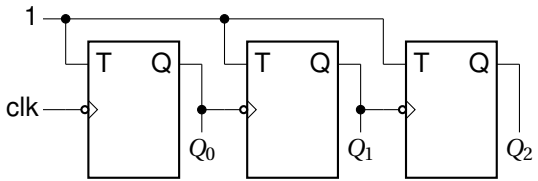
Counters

- Counters are one of the simplest types of sequential networks.
- A counter is usually constructed from one or more flip-flops that change state in a prescribed sequence when input pulses are received.
- Since the clock pulses occur at known intervals, the counter can be used as `an instrument for measuring time` and therefore period of frequency.
  - Asynchronous and synchronous counters.
  - Single and multimode counters.
  - Modulus counters.
    - **Modulus**: the number of predefined states.

Outline

Asynchronous Counters

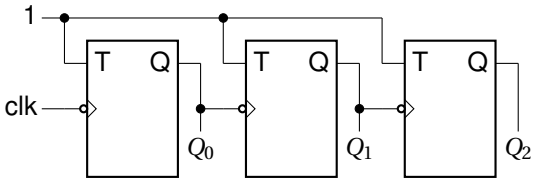
- A.k.a. **Serial** or **ripple counters** (脉冲型计数器).
- The simplest counter circuit can be built using TFF because the toggle feature is naturally suited for the implementation of the counting operation.
- All the flip-flops are not driven by the same clock pulse.
  - The successive flip-flop is triggered by the output of the previous flip-flop.
  - Hence the counter has cumulative settling time, which limits its speed of operation.



T	Q_{t+1}	
0	Q_t	No change
1	Q'_t	Complement

Asynchronous Counters

- The clock inputs of the three flip-flops are connected in cascade.
- The T input of each flip-flop is connected to a constant 1, which means that the state of the flip-flop will toggle (reverse) at each negative edge of its clock.
  → 下降沿触发
- We are assuming that the purpose of this circuit is to count the number of pulses that occur on the primary input CLK (Clock).
- Thus the clock input of the first flip-flop is connected to the CLK line.



Asynchronous Counters

- The counter has 8 different states.
  - It is a MOD-8 asynchronous counter.
- The **Modulus (or MOD-number)** of a counter is the total number of unique states it passes through in each of the complete cycles.
- The maximum binary number that can be counted by the counter is $2^n - 1$.

State	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

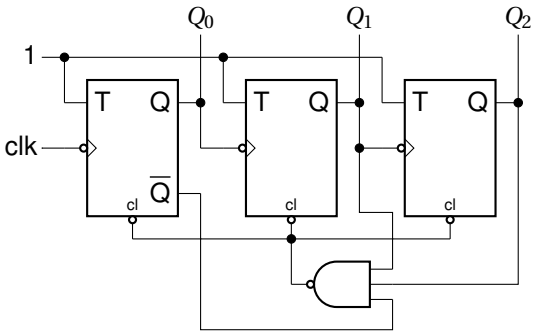
Asynchronous Counters

- The input consists of a sequence of pulses of frequency f.
- Q_0 changes only when the clock makes a transition from 1 to 0.
  - Two input pulses will result in a single pulse in Q_0.
  - The frequency of Q_0 is $f/2$.
- Similarly, the frequency of Q_1 signal will be half that of Q_0 signal, i.e., $f/4$.
- The frequency of Q_2 signal will be half that of Q_1 signal, i.e., $f/8$.
- **Frequency divider**.
  - If there are n flip-flops used in the circuit then the frequency will be divided by 2^n.

State	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

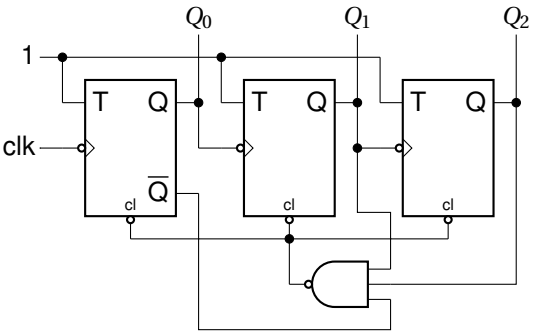
Asynchronous Counter with Modulus $< 2^n$

- In practice, it is often required to have a counter which has a MOD-number less than 2^n.
- In such cases, it is required that the counter will skip states that are normally a part of the counting sequences.



Asynchronous Counter with Modulus $< 2^n$

- Although the counter goes to the 110 state, it remains there only for a few nanoseconds before it recycles to the 000 state.
- Hence we may say that the counter counts from 000 to 101, it skips the states 110 and 111
- Works as a MOD-6 counter.

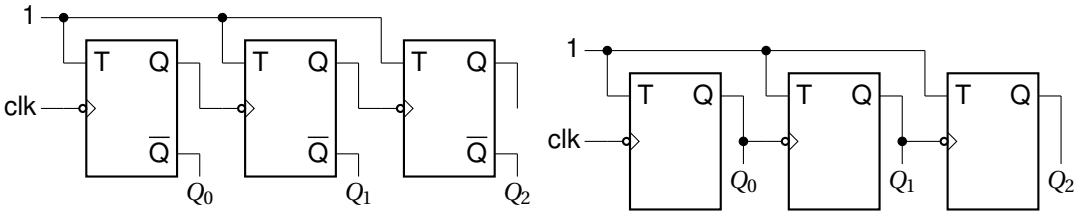


Design An Asynchronous Counter with Modulus $< 2^n$

- To construct any MOD-N counter, the following general steps are to be followed.
  - Find the number of flip-flops n required for the desired MOD-number using the equation $2^{n-1} < N < 2^n$.
  - Then connect all the n flip-flops as a ripple counter.
  - Find the binary number for N.
  - Connect all the flip-flop outputs, for which $Q = 1$, as well as $Q' = 1$, when the count is N, as inputs to the NAND gate.
  - Connect the NAND gate output to the clear input of each flip-flop.
- When the counter reaches the N-th state, the output of the NAND gate goes low, resetting all flip-flops to 0.

Asynchronous Down-counter

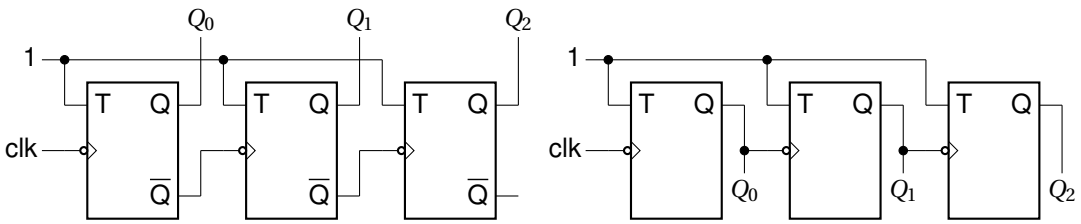
- A **down-counter** using n flip-flops counts downward starting from a maximum count of 2^n–1 to zero.
- Such a down-counter may be designed in three different ways as follows.



(down counter v.s. up-counter)

Asynchronous Down-counter

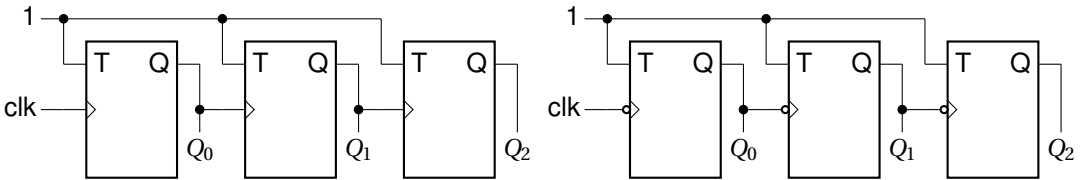
- A **down-counter** using n flip-flops counts downward starting from a maximum count of 2^n–1 to zero.
- Such a down-counter may be designed in three different ways as follows.



(down counter v.s. up-counter)

Asynchronous Down-counter

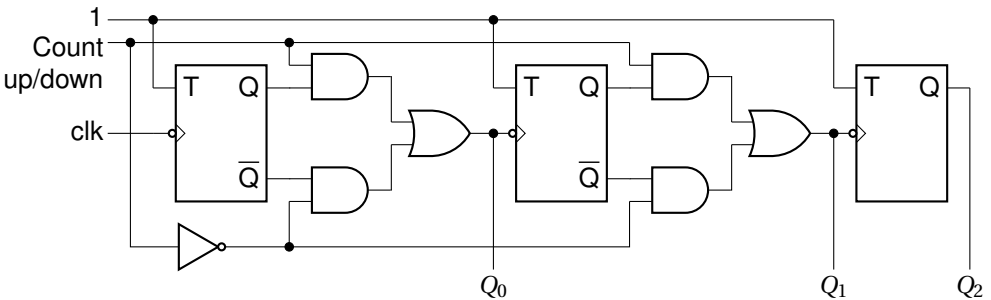
- A **down-counter** using n flip-flops counts downward starting from a maximum count of $2^n - 1$ to zero.
- Such a down-counter may be designed in three different ways as follows.



(down counter v.s. up-counter)

Asynchronous Up-down Counter

- We have already considered up-counters and down-counters separately.
- But both of the units can be combined in a single **up-down counter**.
  - Such a counter is also called a **multimode counter**.



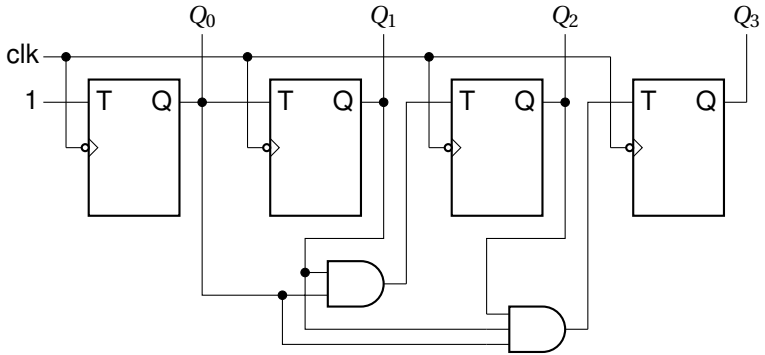
Outline

Synchronous Counters

- The ripple or asynchronous counter is the simplest to build, but its highest operating frequency is `limited because of ripple action`.
  - Each flip-flop has a delay time.
  - In ripple counters these delay times are additive and the total “settling time” (安定时间) for the counter is approximately the product of the delay time of a single flip-flop and the total number of flip-flops.
  - There is the possibility of glitches (毛刺) occurring at the output of decoding gates used with a ripple counter.
- Both of these problems can be overcome, if all the flip-flops are clocked synchronously.
- The resulting circuit is known as a **synchronous counter**.

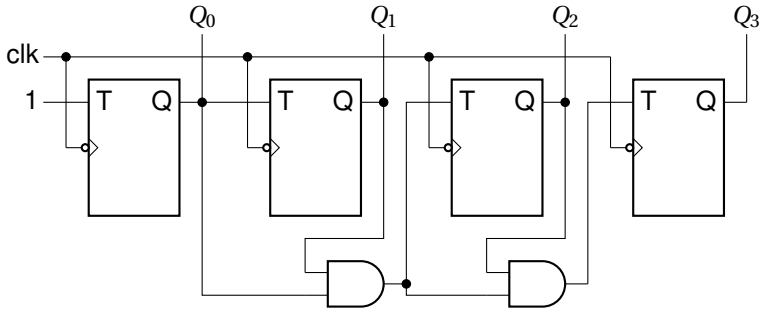
Synchronous Counters

- **Synchronous counter with parallel carry**.



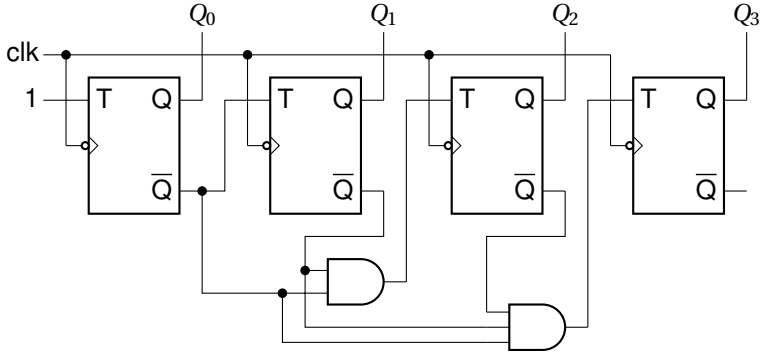
Synchronous Counters

- **Synchronous counter with ripple carry**.
  - As the number of stages increases, the number of AND gates also increases, along with the number of inputs for each of those AND gates.



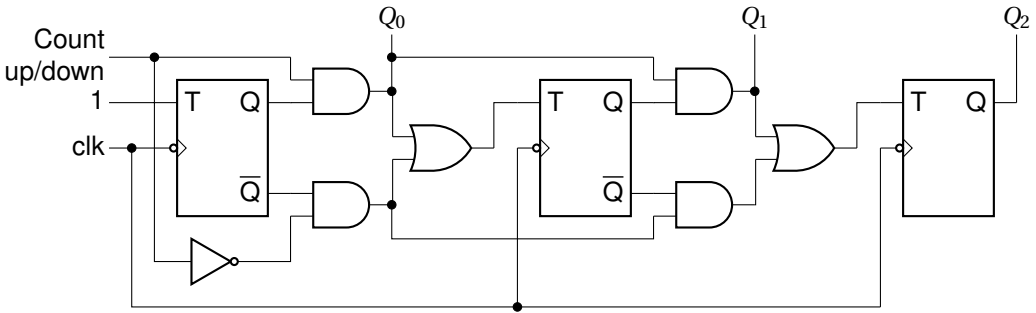
Synchronous Down-counter

- A parallel down-counter can be made to count down by using the inverted outputs of flip-flops to feed the various logic gates.



Synchronous Up-down Counter

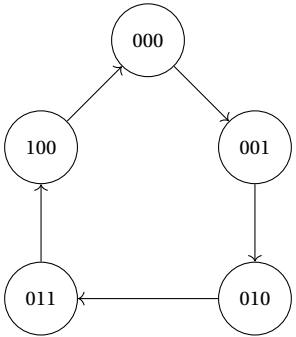
- Combining both the functions of up- and down-counting in a single counter, we can make a **synchronous up-down counter**.



Design A Synchronous counter

- Following certain general steps, synchronous counters of any given count sequence and modulus can be designed. The steps are listed below:
  - From the given word description of the problem, draw a state diagram that describes the operation of the counter.
  - From the state table, write the count sequences in the form of a table.
  - Find the number of flip-flops required.
  - Decide the type of flip-flop to be used for the design of the counter. Then determine the flip-flop inputs that must be present for the desired next state from the present state using the excitation table of the flip-flops.
  - Prepare K-maps for each flip-flop input in terms of flip-flop outputs as the input variables. Simplify the K-maps and obtain the minimised expressions.
  - Connect the circuit using flip-flops and other gates corresponding to the minimised expressions.

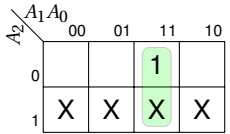
Design A MOD-5 Counter



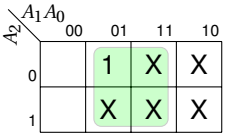
Design A MOD-5 Counter

Present			Next			Flip-flip Inputs					
A_2	A_1	A_0	A_2	A_1	A_0	J_{A2}	K_{A2}	J_{A1}	K_{A1}	J_{A0}	K_{A0}
0	0	0	0	0	1	0	X	0	X	1	X
0	0	1	0	1	0	0	X	1	X	X	1
0	1	0	0	1	1	0	X	X	0	1	X
0	1	1	1	0	0	1	X	X	1	X	1
1	0	0	0	0	0	X	1	0	X	0	X

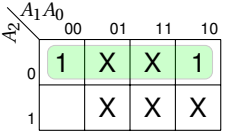
Design A MOD-5 Counter



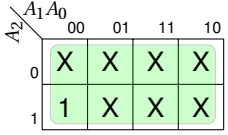
$$J_{A2} = A_1A_0$$



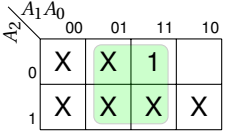
$$J_{A1} = A_0$$



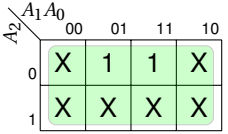
$$J_{A0} = A_2'$$



$$K_{A2} = 1$$



$$K_{A1} = A_0$$



$$K_{A0} = 1$$

Lock Out

- In the counters with modulus less than 2^n, it may happen that the counter by chance finds itself in any one of the unused states.
  - See `101`, `110`, `111` in the example above.
- If by chance the counter enters into any one of these unused states, its next state will not be known.
- It may be possible that the counter might go from one unused state to another and never arrive at a used state.
- A counter whose unused states have this feature is said to suffer from **lock out**.
- To ensure that lock out does not occur, we design the counter `assuming the next state to be the initial state, from each of the unused states`.

Outline of This Lecture

Summary

- In this lecture ...
  - Design of serial adders using FFs and combinational circuits.
  - Design of asynchronous / synchronous counters using FFs and combinational circuits.
- Remarks:
  - Operations in digital computers are usually done in parallel because that is a faster mode of operation.
  - But serial operations have the advantage of requiring fewer hardware components.

Essential Reading

- Essential reading for this lecture: pages 266-280 of the textbook.
- Essential reading for next lecture: pages 299-324 of the textbook.

[1] M. M. Mano and M. Ciletti, *Digital design: with an introduction to the Verilog HDL*. Pearson, 2013