Registers

CS207 Chapter 9

James YU
yujq3@sustech.edu.cn

Department of Computer Science and Engineering
Southern University of Science and Technology

Jul. 15, 2021

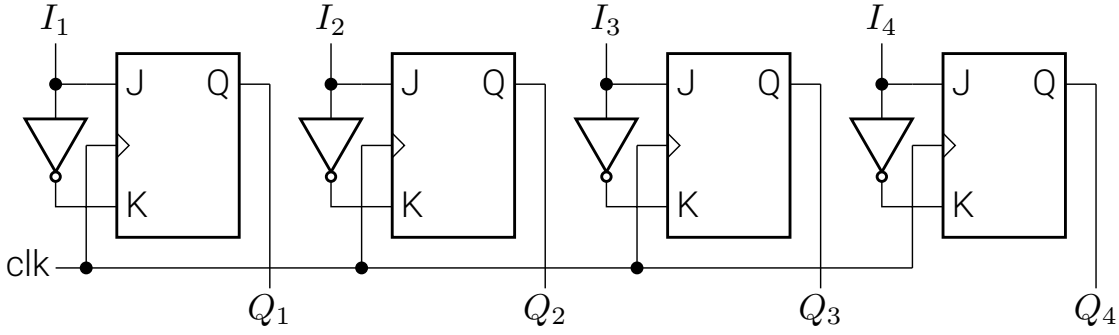
南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

Registers

- Clocked sequential circuits have flip-flops and combinational gates.
  - FFs 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 and counters.
- *Register*:
  - A group of flip-flops, each one of which shares a common clock and is capable of storing one bit of 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*:
  - Essentially a register that goes through a predetermined sequence of binary states.

Registers

- Various types of registers are available.
  - The simplest possible register is one that contains no external gates, and is constructed of only flip-flops.
- The clock pulse enables all the flip-flops at the same instant so that the information available at the four inputs can be transferred into the 4-bit register.
- All the flip-flops in a register should respond to the clock pulse transition.

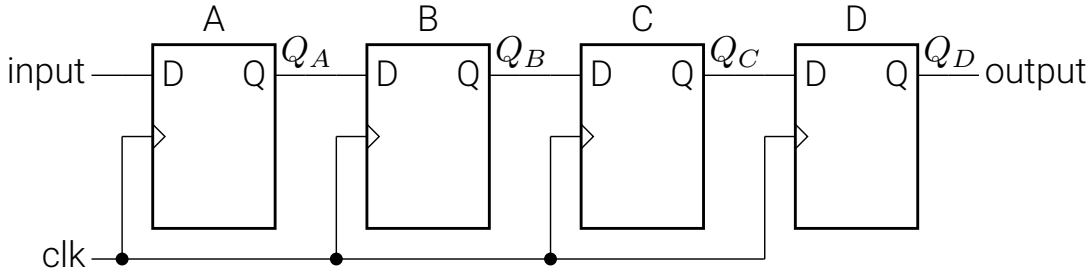


Shift register

- A register capable of shifting its binary contents either to the left or to the right is called a *shift register*.
  - The shift register permits the stored data to move from a particular location to some other location within the register.
- The data in a shift register can be shifted in two possible ways:
  - *serial shifting* shifts one bit at a time for each clock pulse in a serial manner, beginning with either LSB or MSB, and
  - *parallel shifting* shifts all the data (input or output) simultaneously during a single clock pulse.
- Parallel shifting operation is much faster than serial shifting operation.
  - What is the drawback?

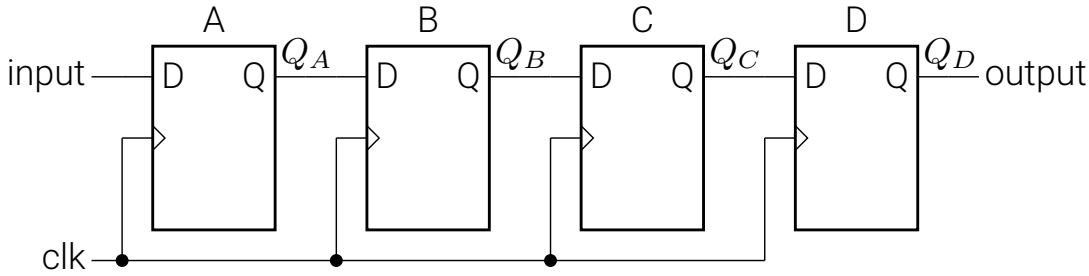
Serial-in serial-out shift register

- From the name itself it is obvious that this type of register accepts data serially, i.e., one bit at a time at the single input line.
- The output is also obtained on a single output line in a serial fashion.
- The data within the register may be shifted from left to right using *shift-left* register, or may be shifted from right to left using *shift-right* register.



- For each clock pulse, data stored in the register is shifted to the right by one stage.
- New data is entered into stage A, whereas the data present in stage D are shifted out (to the right).

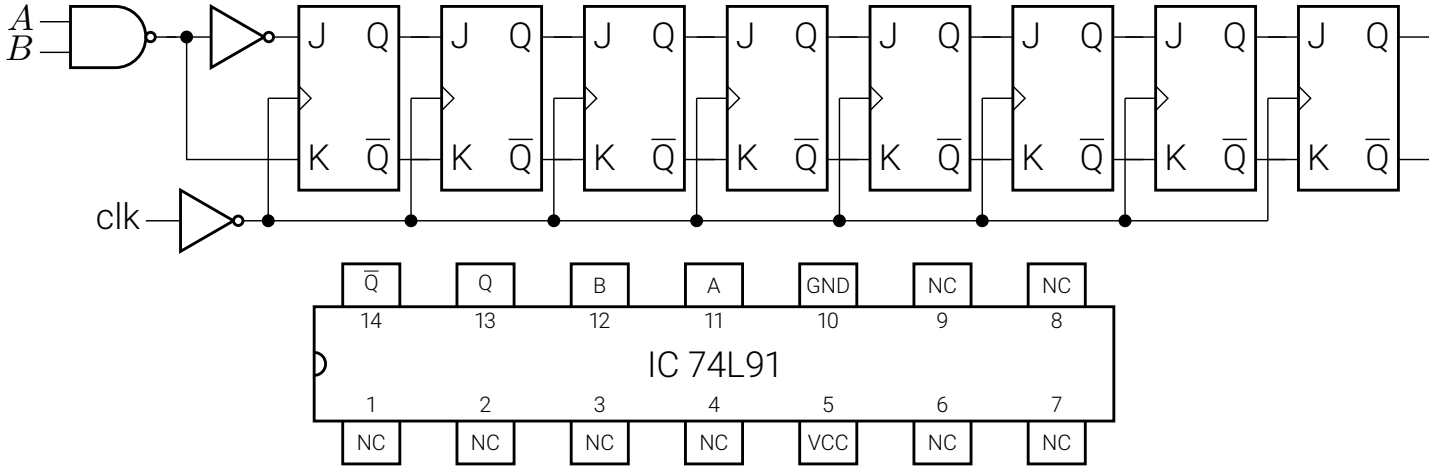
Serial-in serial-out shift register



- An example of 1011 into the register.

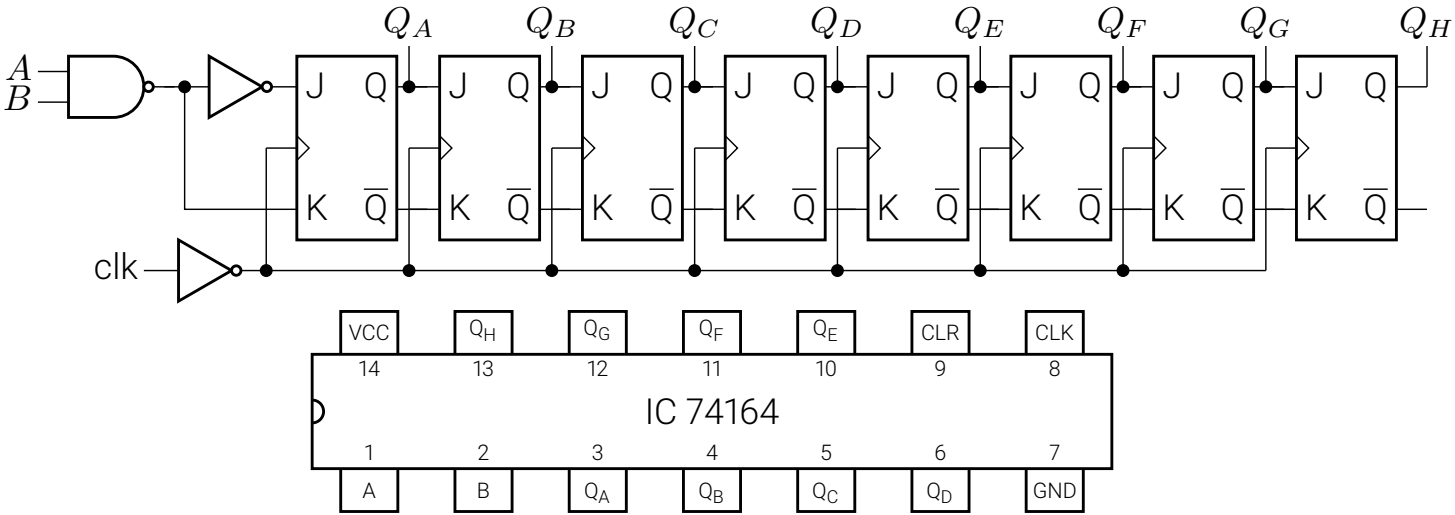
	Q_A	Q_B	Q_C	Q_D	Serial output
Initial value	0	0	0	0	0
After 1st clock pulse	1	0	0	0	0
After 2nd clock pulse	1	1	0	0	0
After 3rd clock pulse	0	1	1	0	0
After 4th clock pulse	1	0	1	1	1

Serial-in serial-out shift register



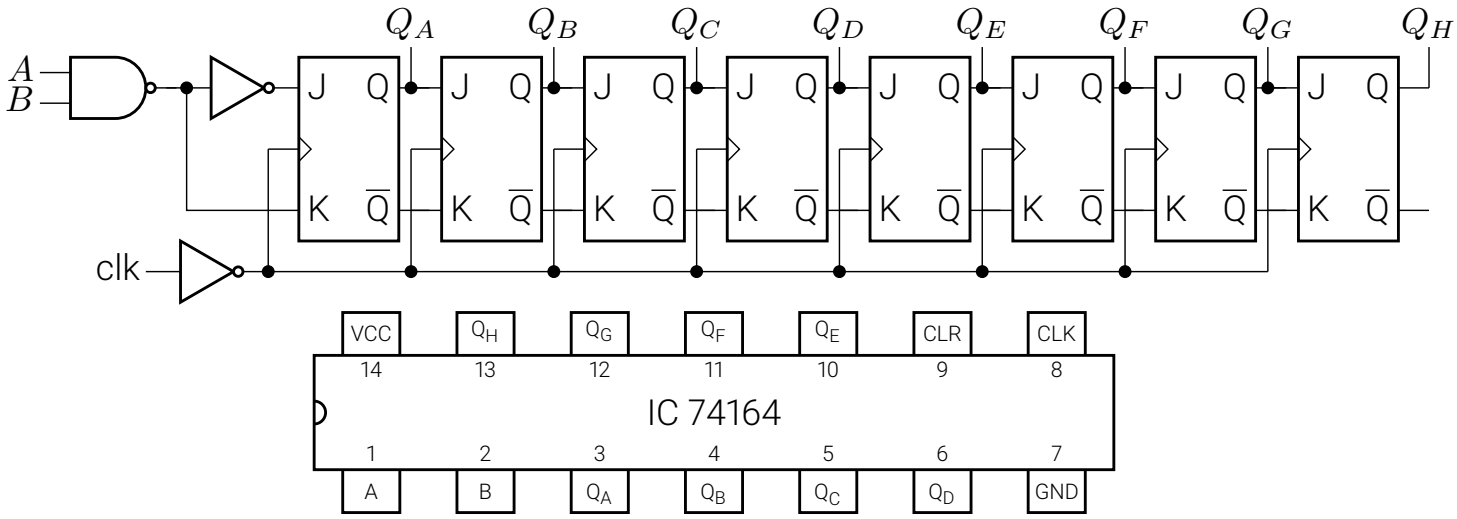
- NC: not connected
- VCC: power supply
- GND: ground

Serial-in parallel-out register



- In this type of register, the data is shifted in serially, but shifted out in parallel.
- To obtain the output data in parallel, it is required that all the output bits are available at the same time.
  - This can be accomplished by connecting the output of each flip-flop to an output pin.

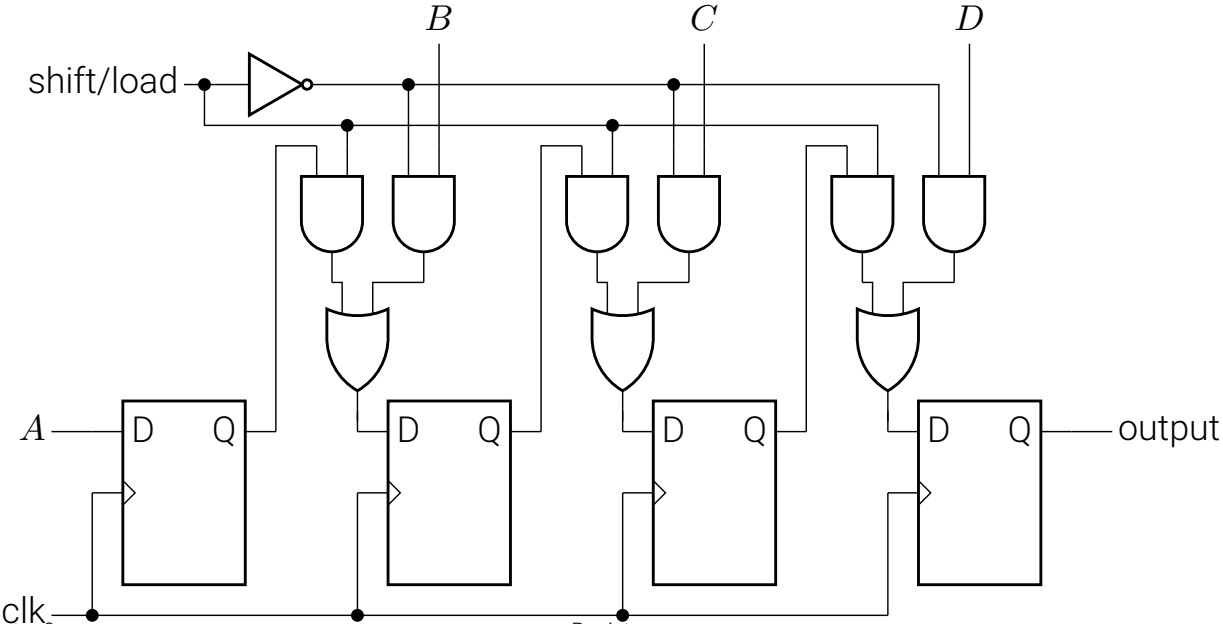
Serial-in parallel-out register



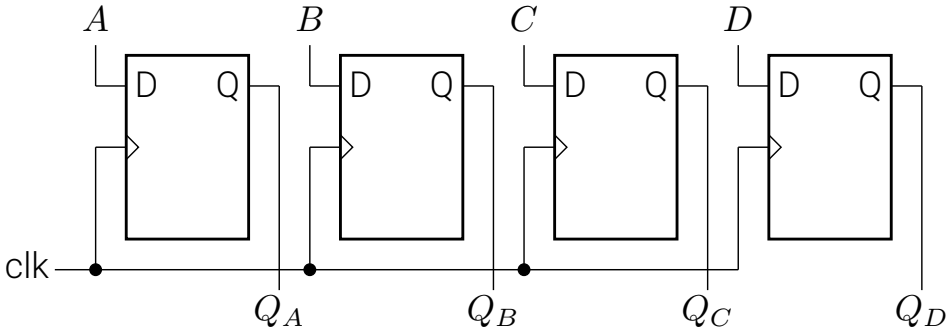
- One question: How long will it take to shift an 8-bit number into a 74164 shift register if the clock is set at 1 MHz?
  - A minimum of eight clock pulses will be required since the data is entered serially. One clock pulse period is 1000 ns, so it will require 8000 ns minimum.

Parallel-in serial-out register

- In the preceding two cases the data was shifted into the registers in a serial manner.
- We now can develop an idea for the parallel entry of data into the register.



Parallel-in parallel-out register



Universal shift register

- If the flip-flop outputs of a shift register are accessible, then information entered serially by shifting can be taken out in parallel from the outputs of the flip-flops.
- If a parallel load capability is added to a shift register, then data entered in parallel can be taken out in serial fashion by shifting the data stored in the register.
- Some shift registers provide the necessary input and output terminals for parallel transfer.
    - They may also have both shift-right and shift-left capabilities.

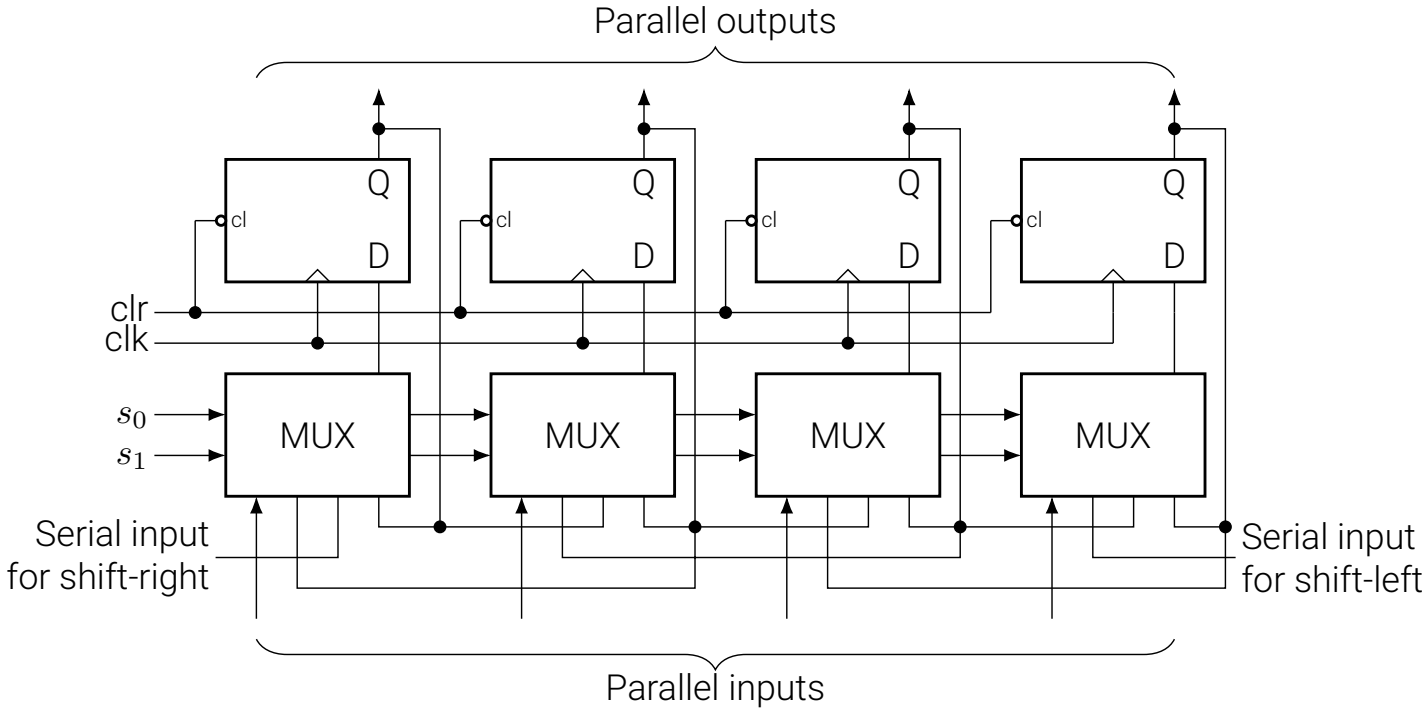
Universal shift register

- A clear control to clear the register to `0`.
- A clock input to synchronize the operations.
- A shift-right control to enable the shift-right operation and the serial input and output lines associated with the shift right.
- A shift-left control to enable the shift-left operation and the serial input and output lines associated with the shift left.
- A parallel-load control to enable a parallel transfer and the n input lines associated with the parallel transfer.
- n parallel output lines.
- A control state that leaves the information in the register unchanged in response to the clock.

Universal shift register

- A register capable of shifting in one direction only is a *unidirectional shift register*.
- One that can shift in both directions is a *bidirectional shift register*.
- If the register has both shifts and parallel-load capabilities, it is referred to as a *universal shift register*.

Universal shift register



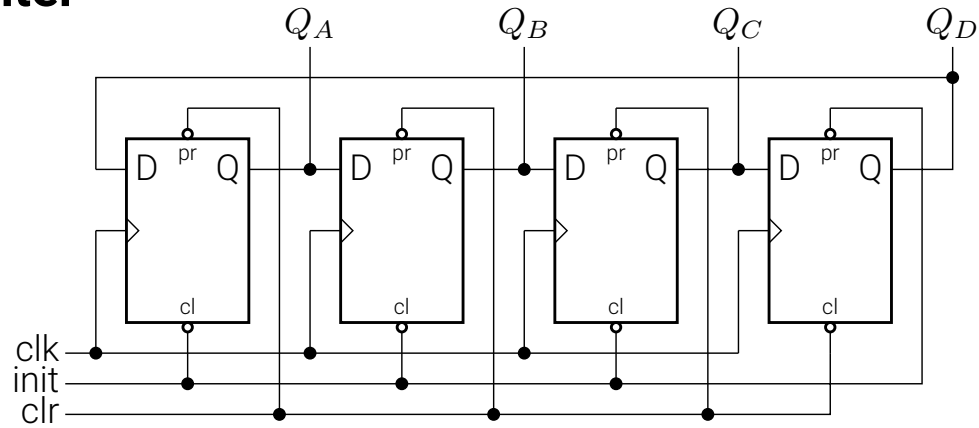
Shift register counters

- Shift registers may be arranged to form different types of counters.
- These shift registers use feedback, where the output of the last flip-flop in the shift register is fed back to the first flip-flop.
- Based on the type of this feedback connection, the shift register counters are classified as
  - *ring counter*, and
  - *twisted ring* or *Johnson* or *Shift counter*.

Ring counter

- It is possible to devise a counter-like circuit in which each flip-flop reaches the state $Q = 1$ for exactly one count, while for all other counts $Q = 0$.
  - Then Q indicates directly an occurrence of the corresponding count.
  - Since this does not represent binary numbers, it is better to say that the outputs of the flip-flops represent a code.
- A *ring counter* is a circular shift register with only one flip-flop being set at any particular time and all others being cleared.
- The single bit is shifted from one flip-flop to the other to produce the sequence of timing signals.

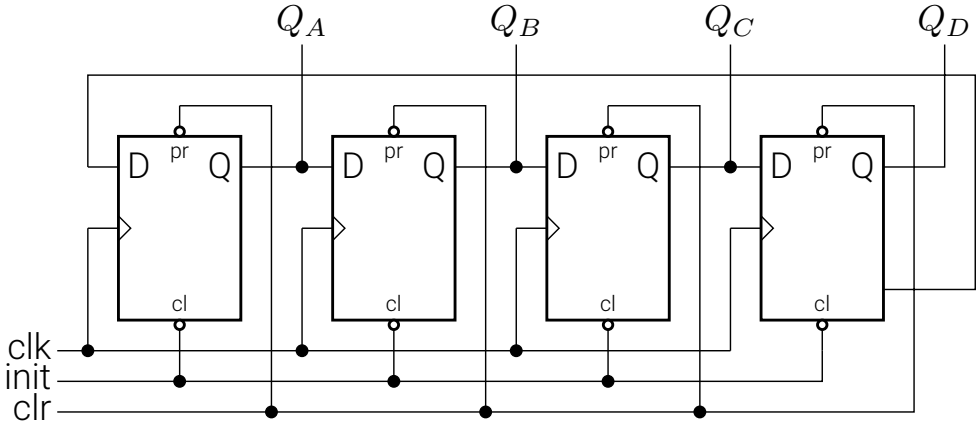
Ring counter



init	clk	Q_A	Q_B	Q_C	Q_D
L	X	0	0	0	1
H	↑	1	0	0	0
H	↑	0	1	0	0
H	↑	0	0	1	0
H	↑	0	0	0	1

Johnson counter

- A k-bit ring counter circulates a single bit among the flip-flops to provide k distinguishable states.
- The number of states can be doubled if the shift register is connected as a **switch-tail ring** counter.
  - A circular shift register with the complement of the last flip-flop being connected to the input of the first flip-flop.



Johnson counter

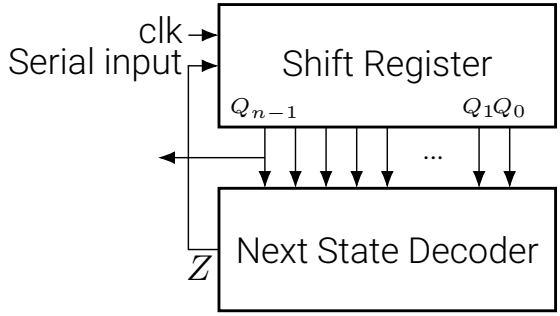
init	clk	Q_A	Q_B	Q_C	Q_D
L	X	0	0	0	0
H	↑	1	0	0	0
H	↑	1	1	0	0
H	↑	1	1	1	0
H	↑	1	1	1	1
H	↑	0	1	1	1
H	↑	0	0	1	1
H	↑	0	0	0	1
H	↑	0	0	0	0

Johnson counter

- One disadvantage of the circuit is that, if it finds itself in an unused state, it will persist in moving from one invalid state to another and never find its way to a valid state.
  - The difficulty can be corrected by modifying the circuit to avoid this undesirable condition.
  - **How?**

Sequence generator

- A *sequence generator* is a circuit that generates a desired sequence of bits in synchronization with a clock.
  - Can be used as a random bit generator, code generator, and prescribed period generator.



- The output of the next state decoder is a function of the shift register state
- and is connected to the serial input of the shift register.
  - This sequence generator is similar to a ring counter or a Johnson counter.

Design a 4-bit sequence generator

- We consider the design of a sequence generator to generate a sequence of 1001.
- The minimum number of flip-flops required to generate a sequence of length N is given by

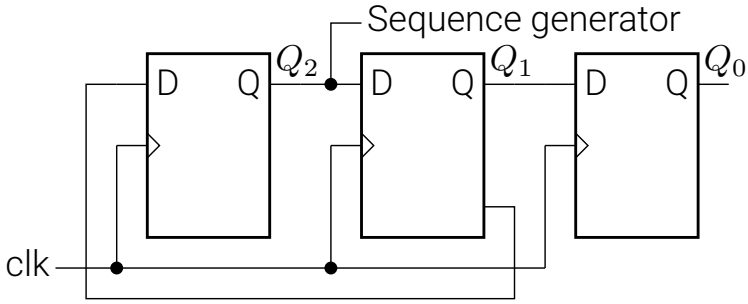
$$N \leq 2^n - 1$$

  - The minimum value of n to satisfy the above condition is 3.

clk	Q_2	Q_1	Q_0	Z
↑	1	1	0	0
↑	0	1	1	0
↑	0	0	1	1
↑	1	0	0	1
↑	1	1	0	0
↑	0	1	1	0

Design a 4-bit sequence generator

Q_2 \ Q_1Q_0	00	01	11	10
0	X	1		X
1	1	X	X	



Design a 5-bit sequence generator

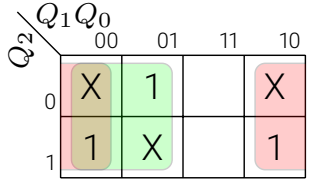
- We consider the design of a sequence generator to generate a sequence of 10011.
- The minimum number of flip-flops required to generate a sequence of length N is given by

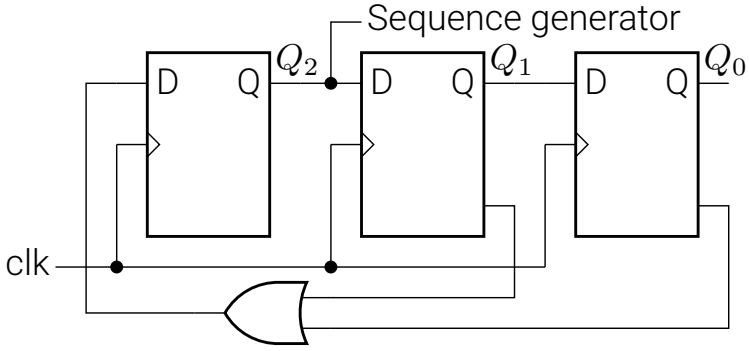
$$N \leq 2^n - 1$$

  - The minimum value of n to satisfy the above condition is 3.

clk	Q_2	Q_1	Q_0	Z
↑	1	1	1	0
↑	0	1	1	0
↑	0	0	1	1
↑	1	0	0	1
↑	1	1	0	1
↑	1	1	1	0

Design a 5-bit sequence generator





Design a 6-bit sequence generator

- We consider the design of a sequence generator to generate a sequence of 110101.
- The minimum number of flip-flops required to generate a sequence of length N is given by

$$N \leq 2^n - 1$$

  - The minimum value of n to satisfy the above condition is 3.
- **Have a try!**

Design a 6-bit sequence generator

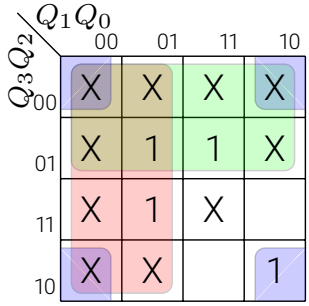
clk	Q_2	Q_1	Q_0
↑	1	1	0
↑	1	1	1
↑	0	1	1
↑	1	0	1
↑	0	1	0
↑	1	0	1

- **State 101 occurs twice!**
- Three flip-flops are not sufficient to generate the given sequence.

Design a 6-bit sequence generator

clk	Q_3	Q_2	Q_1	Q_0	Z
↑	1	1	0	1	1
↑	1	1	1	0	0
↑	0	1	1	1	1
↑	1	0	1	1	0
↑	0	1	0	1	1
↑	1	0	1	0	1

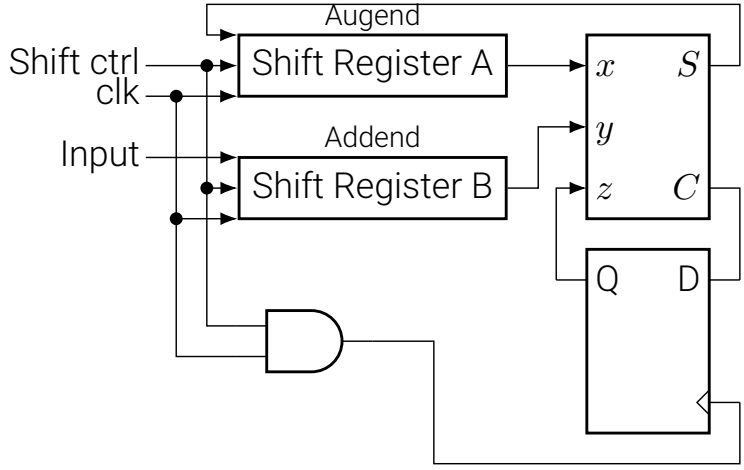
Design a 6-bit sequence generator



$$Z = Q_3' + Q_1' + Q_2'Q_0'$$

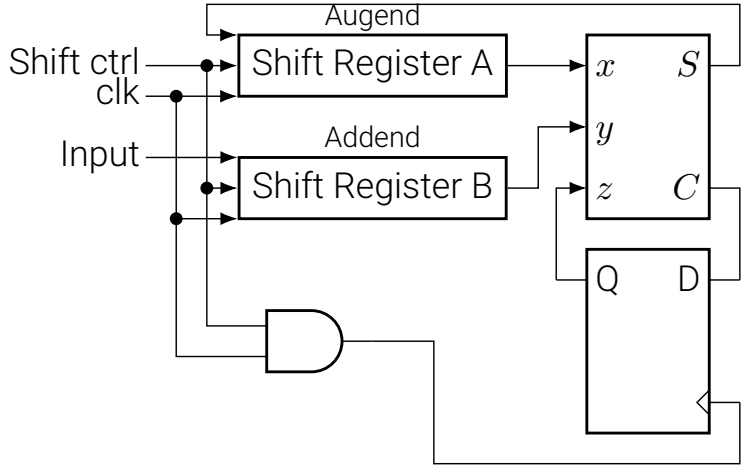
Serial addition

- 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.



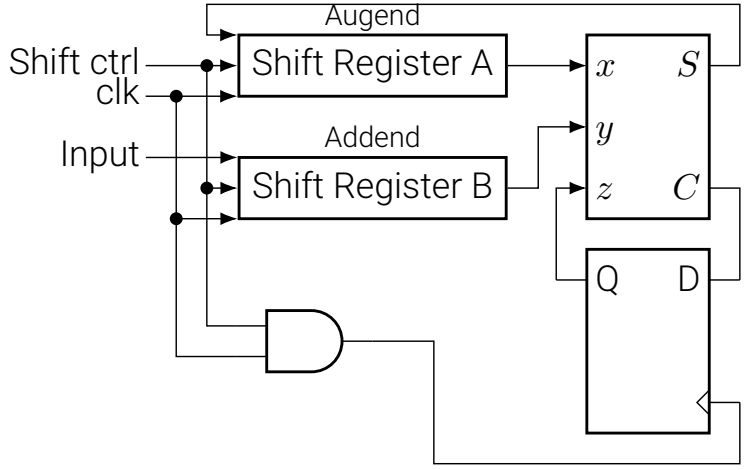
Serial addition

- 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 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.



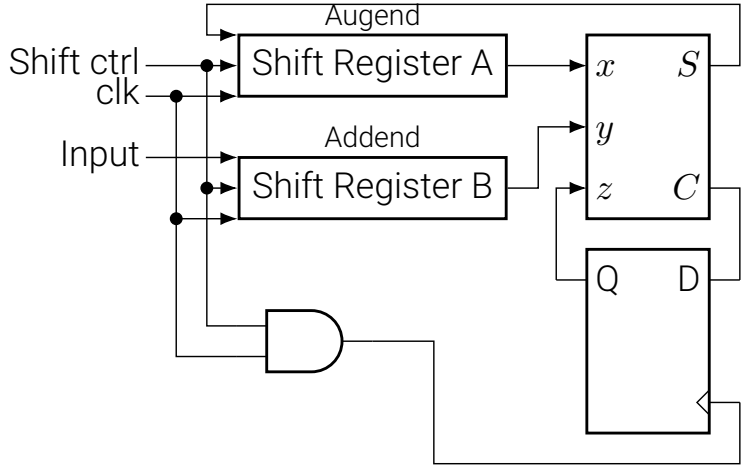
Serial addition

- 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.



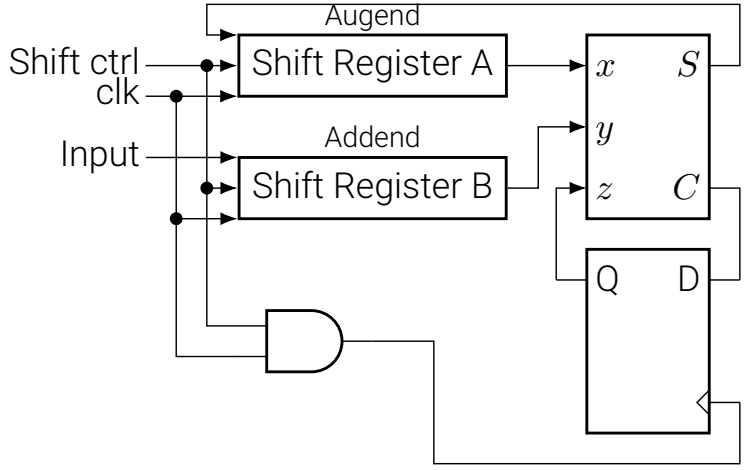
Serial addition

- 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

- We note several differences on the serial adder with the parallel adder.
  - The parallel adder uses registers with a parallel load, whereas the serial adder uses shift registers.
  - The number of full-adder circuits in the parallel 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.



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 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 one.

Design a serial operation

Present State Q	Inputs x	y	Next State Q	Output S	FF Inputs 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

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