Registers and Counters

CS207 Lecture 13

James YU

May 14, 2020

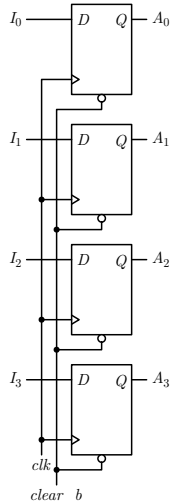
南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

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.

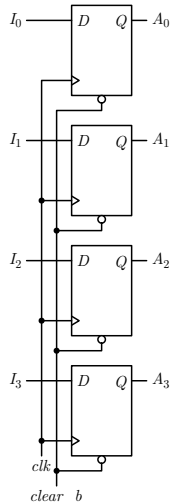
Register

- The simplest register has only FFs.
  - E.g., four DFFs, a common clock, and an input `clear_b` for reset.
  - The common clock input triggers all flip-flops on the positive edge of each pulse, and the binary data available at the four inputs are transferred into the register.
  - The outputs can be sampled at any time to obtain the binary information stored in the register.
  - When `clear_b` goes to 0, all flip-flops are reset asynchronously.
- Transfer information into a register is called *loading* or *updating* the register.
- Loading at the same time is called load *in parallel*.



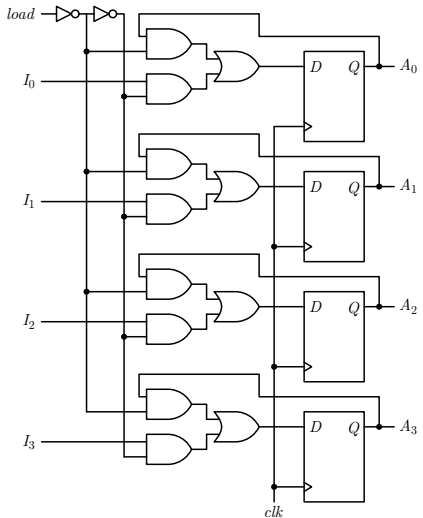
Register with parallel load

- Registers with parallel load are a fundamental building block in digital systems.
- A clock edge applied to the clk inputs of the register will load all four inputs in parallel.
  - If the contents of the register must be left unchanged, the inputs must be held constant,
    - The data bus driving the register would be unavailable for other traffic.
  - or the clock must be inhibited from the circuit.
    - Inserting gates into the clock path is ill advised because the insertion of logic gates produces **uneven propagation delays** between the master clock and the inputs of flip-flops.



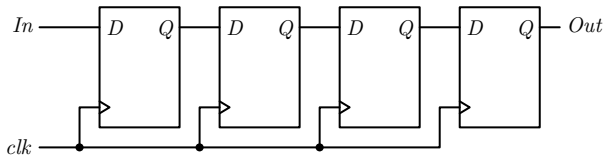
Register with parallel load

- For this reason, it is advisable to control the operation of the register with the D inputs, rather than controlling the clock in the C inputs of the flip-flops.
- The additional gates implement a two-channel MUX whose output drives the input to the register with either the data bus or the output of the register.
- The load input to the register determines the action to be taken with each clock pulse.



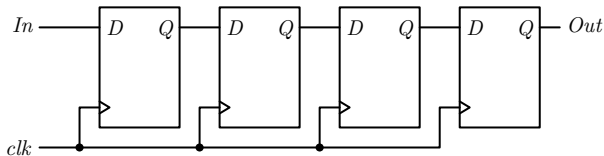
Shift registers

- A register capable of shifting the binary information held in each cell to its neighboring cell, in a selected direction, is called a *shift register*.
  - The logical configuration of a shift register consists of a chain of flip-flops in cascade, with the output of one flip-flop connected to the input of the next flip-flop.
  - All flip-flops receive common clock pulses, which activate the shift of data from one stage to the next.
  - Each clock pulse shifts the contents of the register one bit position to the right.



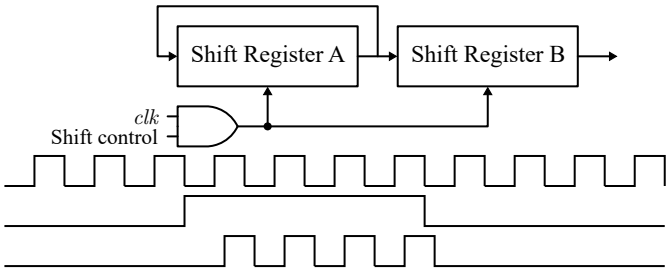
Shift registers

- Sometimes it is necessary to control the shift so that it occurs only with certain pulses, but not with others.
  - As with the data register discussed in the previous section, the clock's **action** can be suppressed by recirculating the output of each register cell back through a two-channel MUX whose output is connected to the input of the cell.
- Note that the simplified schematics do not show a reset signal, but such a signal is required in practical designs.



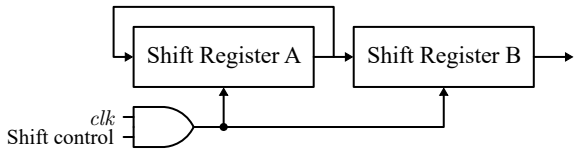
Serial transfer

- The datapath of a digital system is said to operate in serial mode when information is transferred and manipulated one bit at a time.
  - Information is transferred one bit at a time by shifting the bits out of the source register and into the destination register.
  - This type of transfer is in contrast to parallel transfer, whereby all the bits of the register are transferred at the same time.



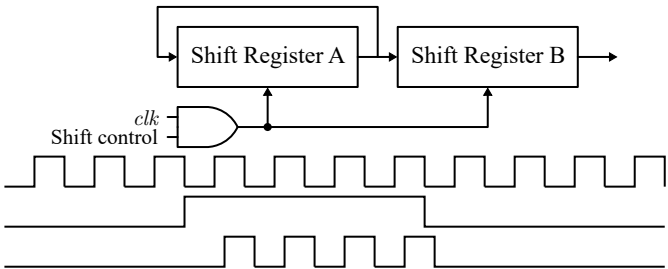
Serial transfer

- The serial transfer of information from register A to register B is done with shift registers.
- The serial output (SO) of register A is connected to the serial input (SI) of register B.
  - To prevent the loss of information stored in the source register, the information in register A is made to circulate by connecting the serial output to its serial input.
  - The initial content of register B is shifted out through its serial output and is lost unless it is transferred to a third shift register.
  - The shift control input determines when and how many times the registers are shifted.



Serial transfer

- Suppose the shift registers have four bits each.
- Then the control unit that supervises the transfer of data must be designed in such a way that it enables the shift registers for a fixed time of four clock pulses in order to pass an entire word.



Serial transfer

- Assume that the binary content of A before the shift is `1011` and that of B is `0010`.
- The serial transfer from A to B occurs in four steps.

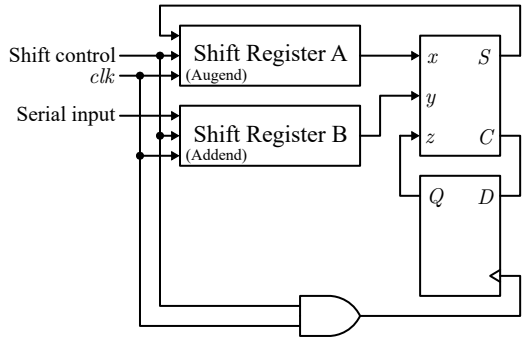
Timing Pulse	Shift Register A				Shift Register B			
Initial	1	0	1	1	0	0	1	0
After 1	1	1	0	1	1	0	0	1
After 2	1	1	1	0	1	1	0	0
After 3	0	1	1	1	0	1	1	0
After 4	1	0	1	1	1	0	1	1

Serial transfer

- The difference between the serial and the parallel mode of operation should be apparent from this example:
  - In the parallel mode, information is available from all bits of a register and all bits can be transferred simultaneously during one clock pulse.
  - In the serial mode, the registers have a single serial input and a single serial output. The information is transferred one bit at a time while the registers are shifted in the same direction.

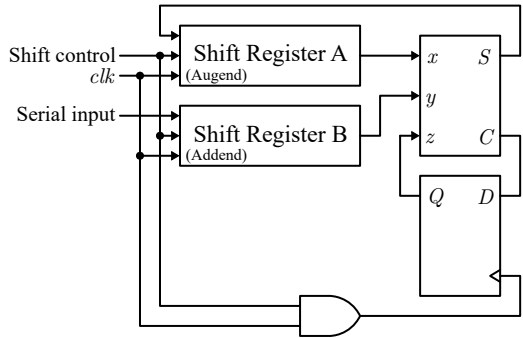
Serial addition

- 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.



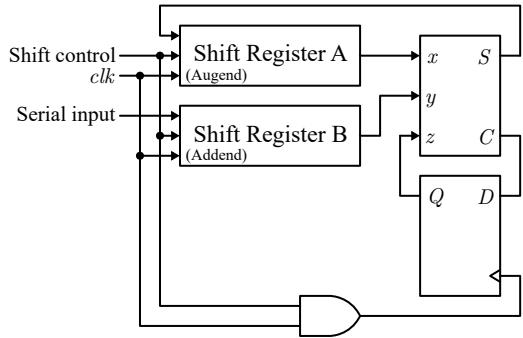
Serial addition

- 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.



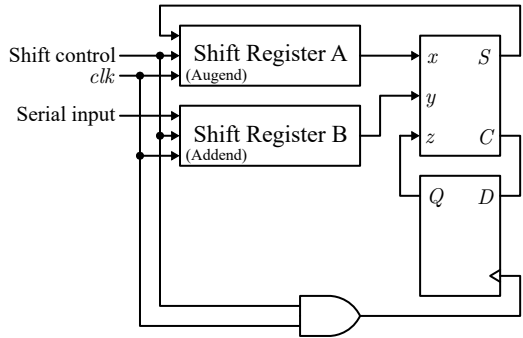
Serial addition

- 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.



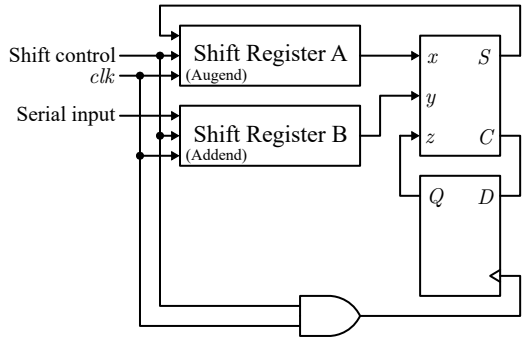
Serial addition

- 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	Inputs		Next State	Output	FF Inputs	
Q	x	y	Q	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

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

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.

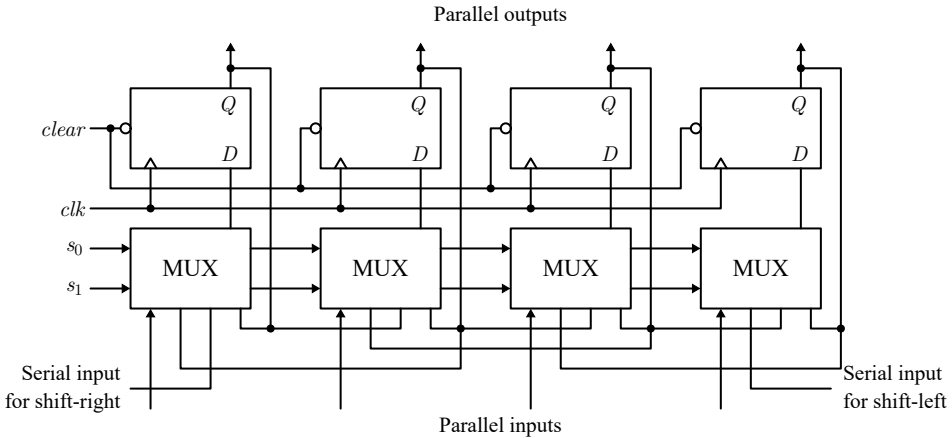
Universal shift register

- 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



Counter

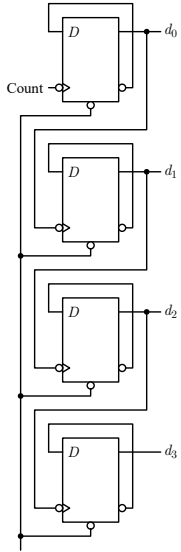
- A register that goes through a prescribed sequence of states upon the application of input pulses is called a *counter*.
- The input pulses may be clock pulses, or they may originate from some external source and may occur at a fixed interval of time or at random.
- A counter that follows the binary number sequence is called a *binary counter*.
  - An n-bit binary counter consists of n flip-flops and can count in binary from 0 through $2^n - 1$.
- Counters are available in two categories: *ripple counters* and *synchronous counters*.

Counter

- In a ripple counter, a flip-flop output transition serves as a source for triggering other flip-flops.
  - the clk input of some or all flip-flops are triggered, not by the common clock pulses, but rather by the transition that occurs in other flip-flop outputs.
- In a synchronous counter, the clk inputs of all flip-flops receive the common clock.

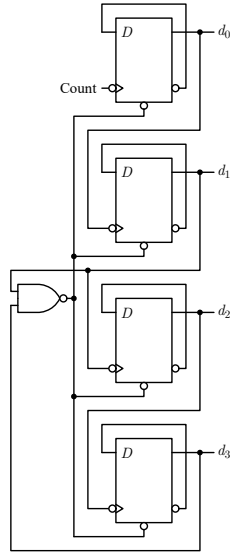
Binary ripple counter

- A binary ripple counter consists of a series connection of complementing flip-flops.
  - With the output of each flip-flop connected to the clk input of the next higher order flip-flop.
  - The flip-flop holding the least significant bit receives the incoming count pulses.
- A binary counter with a reverse count is called a *binary countdown counter*.
  - All flip-flops trigger on the positive edge of the clock.



BCD ripple counter

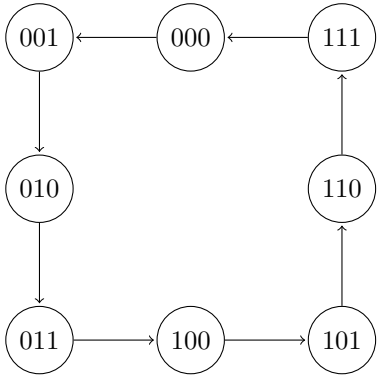
- A decimal counter follows a sequence of 10 states and returns to 0 after the count of 9.
- Such a counter must have at least four flip-flops to represent each decimal digit, since a decimal digit is represented by a binary code with at least four bits.
  - A decimal counter is similar to a binary counter, except that the state after `1001` (the code for decimal digit 9) is `0000` (the code for decimal digit 0).



Synchronous counter

- Synchronous counters are different from ripple counters in that clock pulses are applied to the inputs of all flip-flops.
  - A common clock triggers all flip-flops simultaneously, rather than one at a time in succession as in a ripple counter.
  - We have already designed one previously.

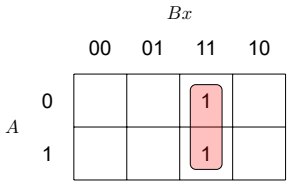
Synchronous counter

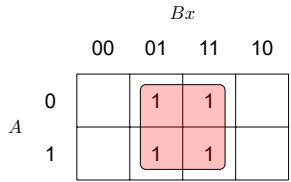


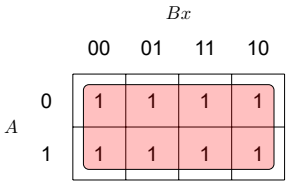
Synchronous counter

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

Synchronous counter







$$T_{A2} = A_1 A_0,$$
$$T_{A1} = A_0,$$
$$T_{A0} = 1.$$

Counter with unused states

- A circuit with n flip-flops has 2^n binary states.
  - There are occasions when a sequential circuit uses fewer than this maximum possible number of states.
  - In simplifying the input equations, the unused states may be treated as don't-care conditions or may be assigned specific next states.
- It is important to realize that once the circuit is designed and constructed, outside interference during its operation may cause the circuit to enter one of the unused states.
  - It is necessary to ensure that the circuit eventually goes into one of the valid states so that it can resume normal operation.
- If the unused states are treated as don't-care conditions, then once the circuit is designed, **it must be investigated to determine the effect of the unused states**.