Registers

In this lesson, you will learn about

- Registers
- Registers with Parallel load
- Shift registers
- Shift registers with Parallel Load
- Bi-directional Shift Registers

Register

A register is a circuit capable of storing data. In general, an n-bit register consists of n FFs, together with some other logic that allows simple processing of the stored data.

All FFs are triggered *synchronously* by the same clock signal. In other words, new data are latched into all FFs at the same time.

Figure 1 shows a 4-bit register constructed with four D-type FFs. In this figure we have:

- Inputs D_0 to D_3
- Clock
- Clear'
- Outputs Q_0 to Q_3

The Clock input is common to all the four D FFs. It triggers all FFs on the rising edge of each clock pulse, and the binary data available at the four D inputs are latched into the register.

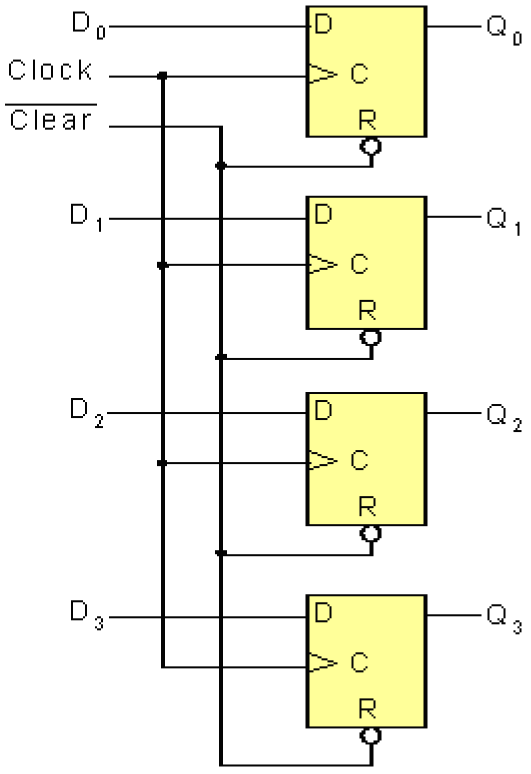


Figure 1: 4-bit register with D flip-flops

- The Clear/ input is an *active-low asynchronous* input which clears the register content to all 0's when asserted low, independent of the clock pulse.
- During normal operation, Clear/ must be maintained at logic 1.
- The transfer of new information into a register is referred to as Loading
- The term Parallel Loading is used if all the input bits are transferred into the register simultaneously, with the common clock pulse.

In most digital systems, a master clock generator supplies clock pulses to all parts of the system, just as the heart that supplies a constant beat to all parts in the human system. Because of this fact, the input values in the register are loaded when a clock pulse arrives. This implies that, whenever a clock pulse arrives, it would load the register with new values, thus overwriting the previously stored register data.

Because of this, a problem arises:
Problem: *What if the contents of the register are to be left unchanged?*
A Solution: The Clock may be prevented from reaching the clock input of the FFs of the register.
$\Rightarrow$ A separate control signal is used
Another Solution: Inputs D_0 to D_3 may be prevented from changing their values.
$\Rightarrow$ A control signal is needed for this

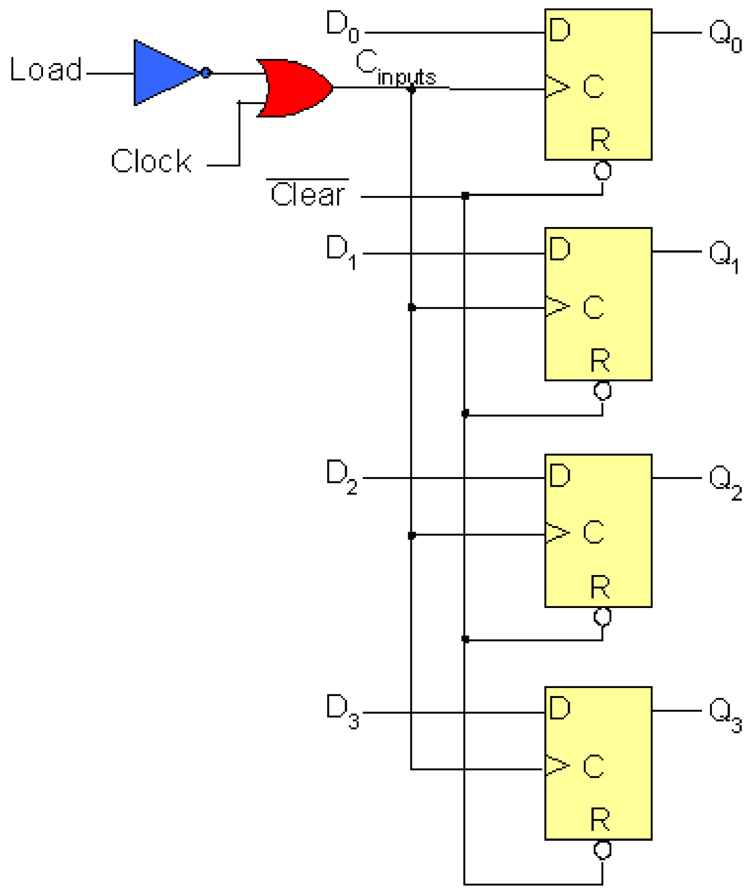


Figure 2: Clock gating

This control can be provided by implementing the following function:

$$C_{inputs} = Load^{/} + Clock$$

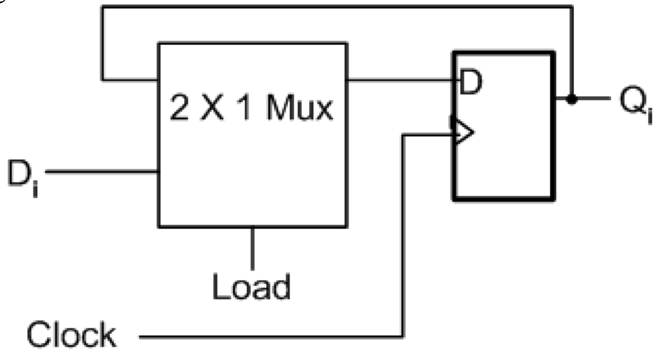
When Load = 0 $\Rightarrow$ $C_{inputs} = 1$, causing no positive transitions to occur on C_{inputs}. Thus contents of the register remain unchanged.

When Load = 1 $\Rightarrow$ $C_{inputs} = Clock$, thus the register is clocked normally.

The above phenomenon is known as **Clock gating** (Figure 2).
Problem: Different stages of the register will be gated at different time. This may cause loading of wrong information (known as **clock skew**).
Solution: Clock gating should be avoided.

Use register with *Parallel Load*. (Figure 3)

- No clock gating is used.



$$D = \overline{\text{Load}}.Q_i + \text{Load}.D_i$$

Figure 3: One stage of Register with Parallel Load

When Load = 1, the data on the input D_i is transferred into the D flip-flop with the next positive transition of clock pulse.
When Load = 0, the data input is blocked, and output Q_i gets a path to the D input of the flip-flop.

Why do we need feedback connection from output to input of D-FF?

- Because D-FF does not have a "no change" input condition. Having the feedback will cause the next state of the FF (D input) to be equal to present state of the FF, i.e. ***no change in state***.

Note that there is no *Clock gating*. Load determines whether to accept new information in the next clock pulse or not.

A 4-bit register with parallel load is shown.

Q: Can clocked latches be used instead of FFs to implement parallel-load registers?

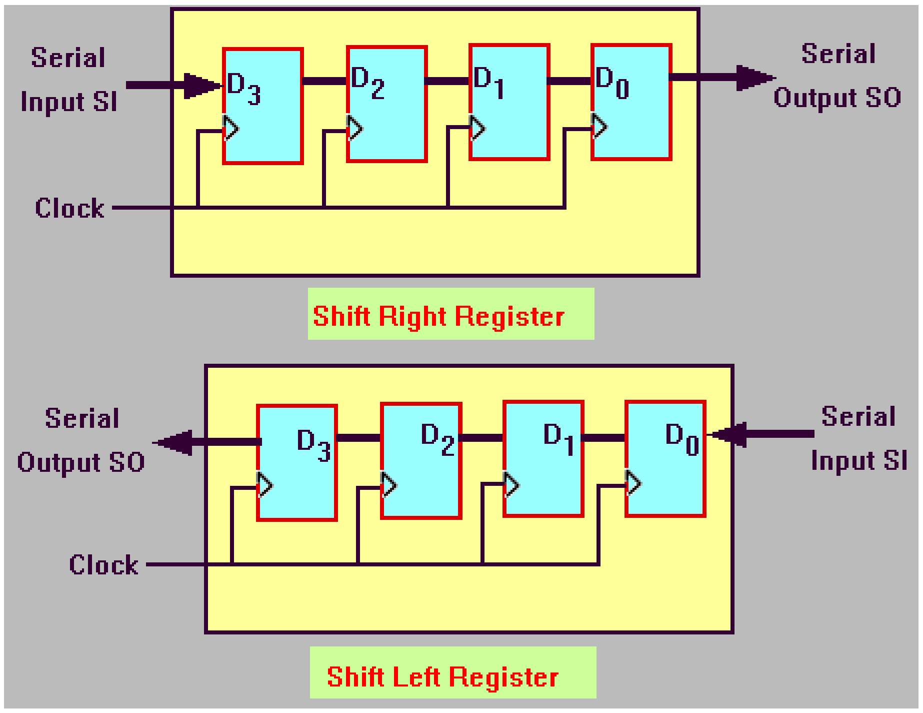


Figure 4: Shift Registers

A shift register (Figure 4) is capable of transferring data from each FF to the next in one or the other direction. Each clock pulse causes data shift from one FF to its immediate neighbor.

The configuration consists of a chain of FFs in cascade, with the output of one FF connected to the input of the next one.

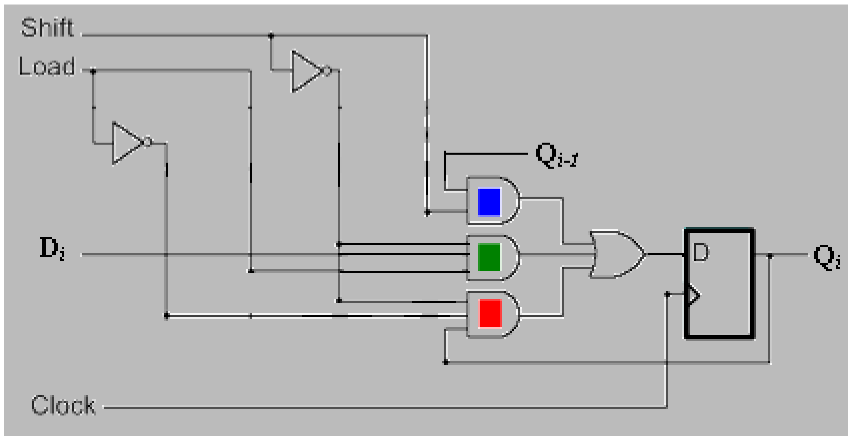
All FFs receive a common pulse, which activates the shift operation from each stage to the next.

The serial input SI is the input to the first (leftmost) FF of the chain.

The serial output SO is the output of the last (rightmost) FF of the chain.

The register discussed above is a "shift right" (MS to LS shifting) register. There is also a "shift left" (LS to MS shifting) register. (Figure 4)

Q: Can clocked latches be used instead of FFs to implement shift registers?



$$D = (Shift \cdot Q_{i-1}) + (\overline{Shift} \cdot Load \cdot D_i) + (\overline{Shift} \cdot \overline{Load} \cdot Q_i)$$

Shift	Load	Operation
0	0	No change
0	1	Load parallel data
1	X	Shift left (from Q_{i-1} to Q_i)

Figure 5: One stage of a shift register with parallel load

A shift register with parallel load capability can be used to input the data bits in parallel into the shift register and then take the data out in a serial fashion by applying the shift operation.

Such a register can act as a parallel-to-serial converter, where data can be loaded in parallel and shifted *out* serially (bit-by-bit)

It can also act as a serial-to-parallel converter, where data can be shifted *in* serially (bit-by-bit) and the output made available in parallel after shifting is complete.

Figure 5 shows a typical stage of a shift register with parallel load. There are two control signals: Shift and Load. A table showing the operation of the register with respect to the Shift and Load inputs is also shown in the figure.

If Shift = 0 and Load = 0, red AND gate is enabled, causing the output of the flip-flop to feed back to its D input.
➢A positive transition in the clock loads this input value into the FF⇒ **No Change** state.

Load condition: If Shift = 0 and Load = 1, the green AND gate is enabled. This causes the D_i input to propagate to its input of the D flip-flop.
➢A positive transition of the clock pulse transfers the input data into the FFs;

Shift condition: If Shift = 1 while Load = 0 or 1, the blue AND gate is enabled, while the other two AND gates are disabled.

- A positive transition of the clock pulse causes the shift operation. That is, the blue AND gate takes the input from output Q_{i-1} of previous flip-flop. This is true for all stages except for the first stage, where, instead, serial input SI is provided

Bidirectional Shift Register (BDSR)

Design a 3-bit shift register which has 4 operating modes. The operating modes are defined by the status of two select lines S_1 and S_0. The given table specifies the values of S_1 and S_0 and its corresponding operating mode.

Mode Control		Register Operation
S1	S0	
0	0	No change
0	1	Shift down
1	0	Shift up
1	1	Parallel load

Table 1: Modes of BDSR

The design uses 3 stages, where each stage consists of a single multiplexer and a single D-FF.

The output of each MUX is connected to the input of corresponding D FF.

The MUX select inputs are connected to S_1 and S_0 to pass the proper signal to the D-FF input depending on the mode of operation.

When $S_1\ S_0 = 00$, input 0 of the MUX is selected. This forms a path from the output of the FF into its own input, which causes the same value to be loaded in the D FF when a clock pulse is applied. This results in the NO CHANGE operation.

When $S_1\ S_0 = 01$, input 1 of the MUX is selected. This forms a path from the lower significant to higher significant bit, resulting in the SHIFT LEFT (i.e. LSB to MSB) operation.

- The serial input SI is transferred into the rightmost bit in this case.

When $S_1\ S_0 = 10$, input 2 of the MUX is selected. This forms a path from the higher significant bit to lower significant bit, resulting in SHIFT RIGHT (i.e. MSB to LSB) operation.

- The serial input SI is transferred into the leftmost bit (i.e. MSB) in this case.

Finally, when $S_1\ S_0 = 11$, input 3 of the MUX is selected. On this input, the binary information on the parallel input line D_i is transferred into the FF, resulting in PARALLEL LOAD operation.

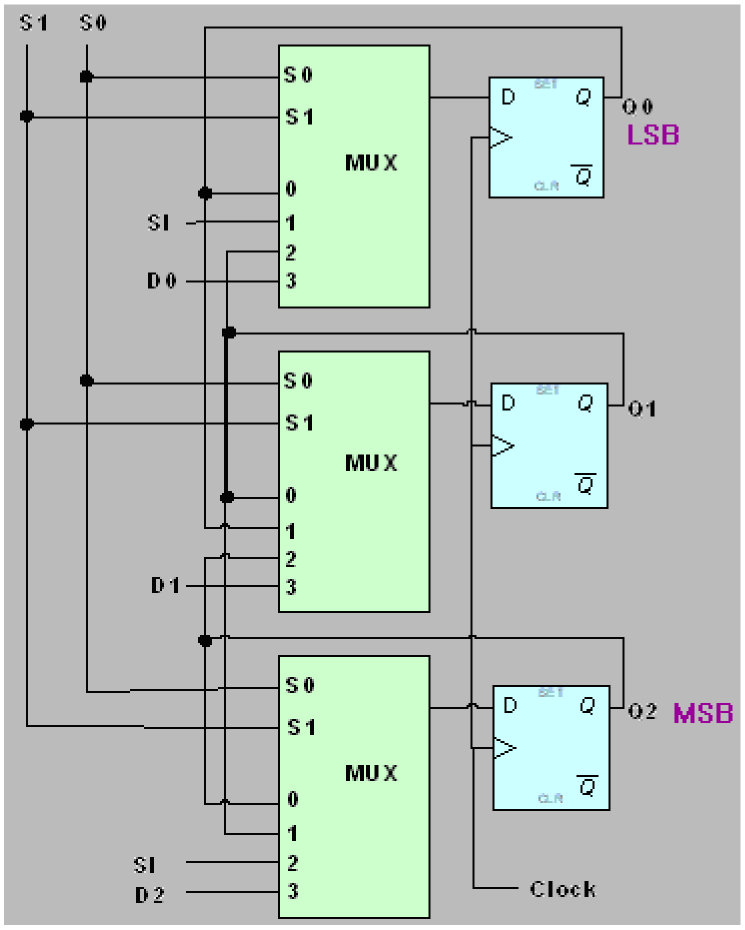


Figure 6: Bi-directional shift register