Lecture 18 In-Class Worksheet

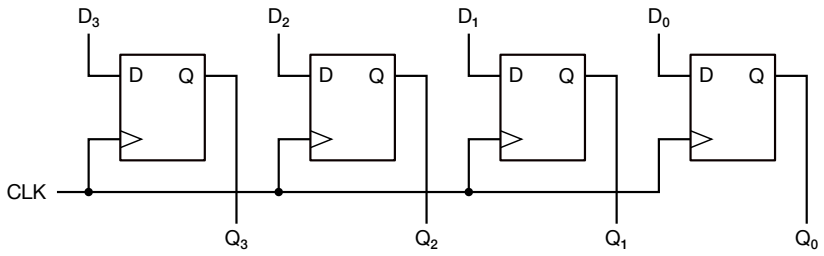
1 Learning Objectives

This worksheet is based on Lumetta course notes section 2.7. After completing this lesson, you will know how to:

- [S18-1] Construct a register that supports any combintion of: load enable, parallel load, serial load, and bit shifts.

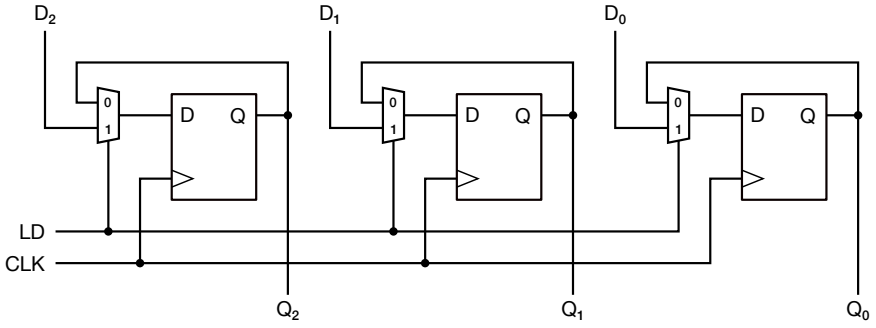
2 Registers

A *register* is a set of one or more flip-flops with a common clock. The simplest kind of register is just a set of n flip-flops with a common clock. A 4-bit register of this type is shown below. This register supports *parallel load*, meaning that all 4 bits can be loaded in one cycle.



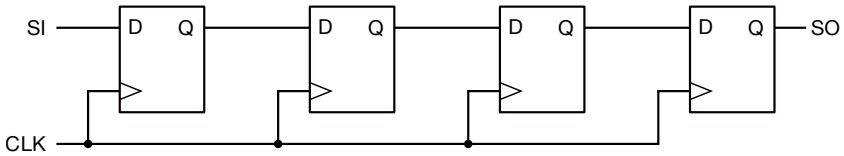
3 Load Control

The simple register above loads a new set of bits on every clock cycle. We can add a control signal to allow us to keep the value currently stored in the register ($LD = 0$) or update it using the provided input ($LD = 1$) on the rising edge of the clock. A 3-bit register of this type is shown below.

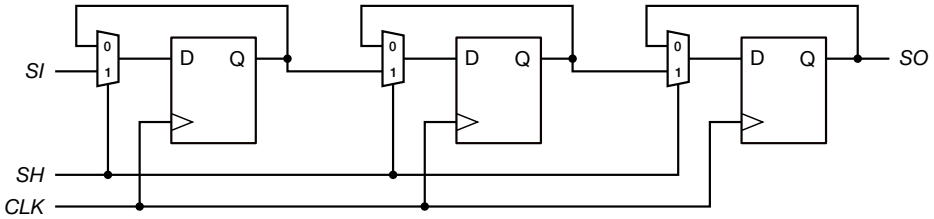


4 Shift Registers

A *shift register* receives its input 1 bit at a time, shifting bits internally on each cycle. A 4-bit register of this type is shown below. On each rising edge of the clock, all values are shifted one bit to the right and *SI* is shifted into the left-most register bit.



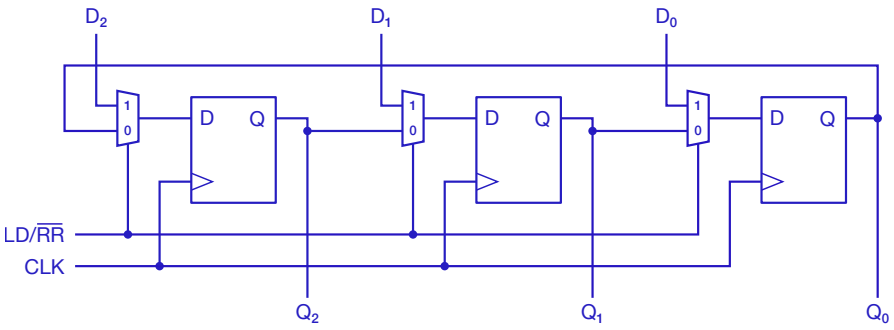
We can also add a control signal to allow us to keep the value curretly stored in the register ($SH = 0$) or to shift the values one bit to the right ($SH = 1$).



5 Registers with Combinations of Features

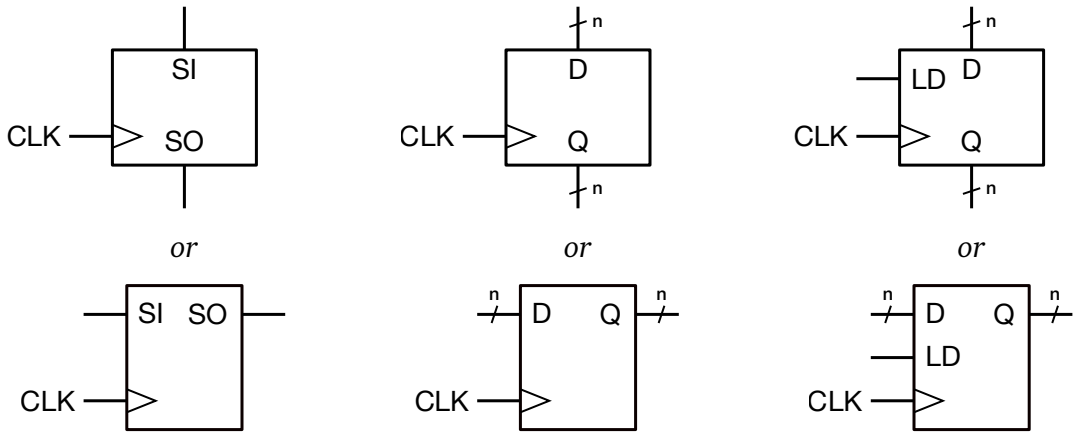
We can combine the mechanisms above to create any combination of features, such as the registers shown in the Lumetta course notes, section 2.7.

> **Q1.** Create a register with parallel inputs and outputs and a control signal $LD/\overline{RR}$ where $LD/\overline{RR} = 1$ causes the register to be loaded in parallel and $LD/\overline{RR} = 0$ causes the bits to rotate right. (In a *rotate* operation, the bit shifted out one side is shifted in at the other. For example, rotating the 4-bit value `1011` right one bit gives `1101`.)


D2
D1
D0
1
0
D
Q
LD/RR
CLK
Q2
Q1
Q0

6 Diagram Symbols

As with other devices we have built so sar, we represent a register using a box with inputs an outputs, hiding its exact implementation. It is usually possible to tell what a register does from the names of the input and output signals it has. Shown below are several such symbols.

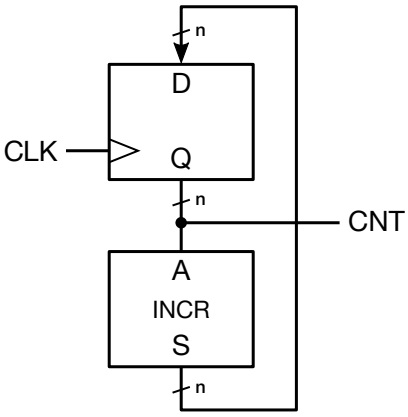


Q2. Match the symbols shown above to the registers described in Sections 2–4.

7 Building Counters Using Registers

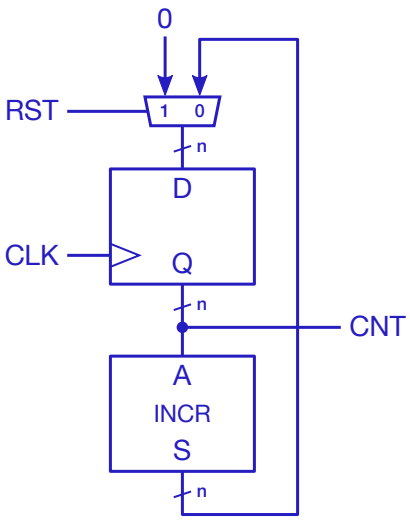
We will use registers extensively throughout the remainder of the class to build complex devices. However, we can already build some useful devices combining a register with some additional logic.

Shown on the right is a n-bit *counter*, a device that counts the number of clock cycles. The device labeled INCR is an incrementer: it adds one to the input. As discussed in Question 3 on Worksheet 13, we can build an n-bit incrementer using an n-bit adder.



Unfortunately, while the counter above does count, it is not very useful: we can't control its starting value or when it starts counting.

Q3. Add a reset input signal to the counter above that resets to counter to zero. When RST $= 1$, the counter value should reset to zero on the next rising edge of the clock. When RST $= 0$, the counter value should increment on the rising edge of the clock.



Q4. Add a input signal to the counter in Q3 above that pauses the counter. When RUN = 1, the counter should count. When RUN = 0, the counter value should not change on the rising edge of the clock.

The figures below show two different possible implementations.