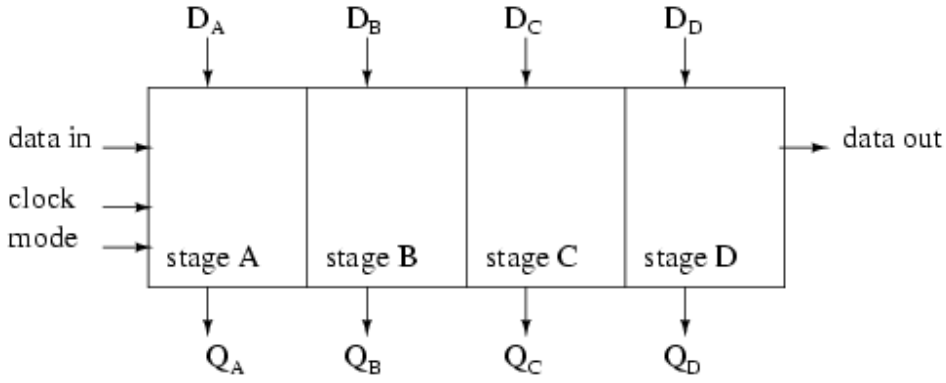
PARALLEL-IN, PARALLEL-OUT, UNIVERSAL SHIFT REGISTER

The purpose of the parallel-in/ parallel-out shift register is to take in parallel data, shift it, then output it as shown below. A universal shift register is a do-everything device in addition to the parallel-in/ parallel-out function.



Parallel-in, parallel-out shift register with 4-stages

Above we apply four bit of data to a parallel-in/ parallel-out shift register at D_A D_B D_C D_D. The mode control, which may be multiple inputs, controls parallel loading vs shifting. The mode control may also control the direction of shifting in some real devices. The data will be shifted one bit position for each clock pulse. The shifted data is available at the outputs Q_A Q_B Q_C Q_D . The "data in" and "data out" are provided for cascading of multiple stages. Though, above, we can only cascade data for right shifting. We could accommodate cascading of left-shift data by adding a pair of left pointing signals, "data in" and "data out", above.

	D_A	D_B	D_C	D_D
data	1	1	0	1
	Q_A	Q_B	Q_C	Q_D
load	1	1	0	1
shift	X	1	1	0
→				

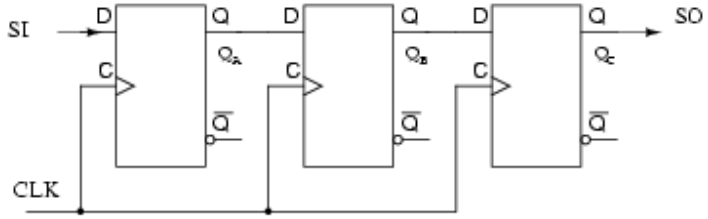
Load and shift

	D_A	D_B	D_C	D_D
data	1	1	0	1
	Q_A	Q_B	Q_C	Q_D
load	1	1	0	1
shift	X	1	1	0
shift	X	X	1	1
→				

Load and 2-shifts

Parallel-in/ parallel-out shift register

Above, a data pattern is presented to inputs D_A D_B D_C D_D. The pattern is loaded to Q_A Q_B Q_C Q_D . Then it is shifted one bit to the right. The incoming data is indicated by X, meaning the we do no know what it is. If the input (SER) were grounded, for example, we would know what data (0) was shifted in. Also shown, is right shifting by two positions, requiring two clocks.



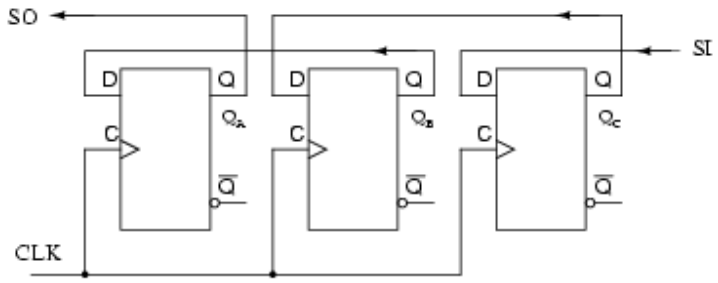
Shift right

The above figure serves as a reference for the hardware involved in right shifting of data. It is too simple to even bother with this figure, except for comparison to more complex figures to follow.

	Q_A	Q_B	Q_C
load	1	1	0
shift →	X	1	1

Load and right shift

Right shifting of data is provided above for reference to the previous right shifter.



Shift left

If we need to shift left, the FFs need to be rewired. Compare to the previous right shifter. Also, SI and SO have been reversed. SI shifts to Q_C. Q_C shifts to Q_B. Q_B shifts to Q_A. Q_A leaves on the SO connection, where it could cascade to another shifter SI. This left shift sequence is backwards from the right shift sequence.

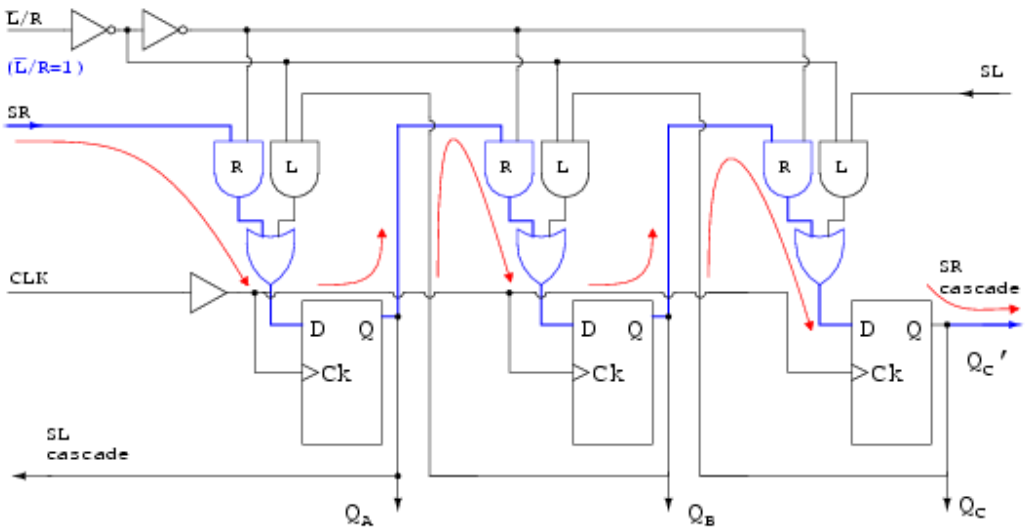
	Q_A	Q_B	Q_C
load	1	1	0
shift	1	0	X

←

Load and left shift

Above we shift the same data pattern left by one bit.

There is one problem with the "shift left" figure above. There is no market for it. Nobody manufactures a shift-left part. A "real device" which shifts one direction can be wired externally to shift the other direction. Or, should we say there is no left or right in the context of a device which shifts in only one direction. However, there is a market for a device which will shift left or right on command by a control line. Of course, left and right are valid in that context.

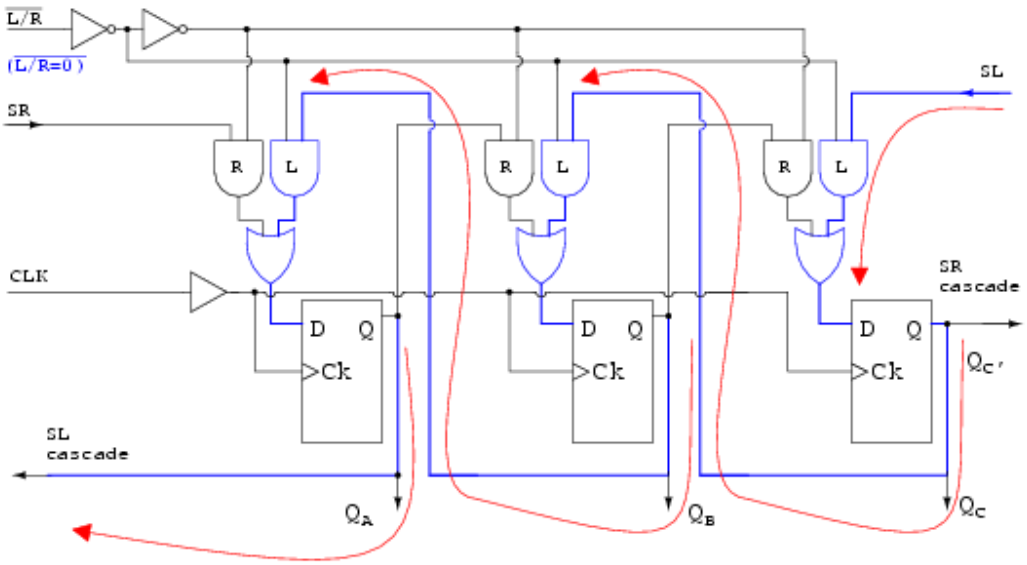


Shift left/ right, right action

What we have above is a hypothetical shift register capable of shifting either direction under the control of L'/R. It is setup with L'/R=1 to shift the normal direction, right. L'/R=1 enables the multiplexer AND gates labeled R. This allows data to follow the path illustrated by the arrows, when a clock is applied. The connection path is the same as the"too simple" "shift right" figure above.

Data shifts in at SR, to Q_A, to Q_B, to Q_C, where it leaves at SR cascade. This pin could drive SR of another device to the right.

What if we change L'/R to L'/R=0?



Shift left/ right register, left action

With L'/R=0, the multiplexer AND gates labeled L are enabled, yielding a path, shown by the arrows, the same as the above "shift left" figure. Data shifts in at SL, to Q_C, to Q_B, to Q_A, where it leaves at SL cascade. This pin could drive SL of another device to the left.

The prime virtue of the above two figures illustrating the "shift left/ right register" is simplicity. The operation of the left right control L'/R=0 is easy to follow.