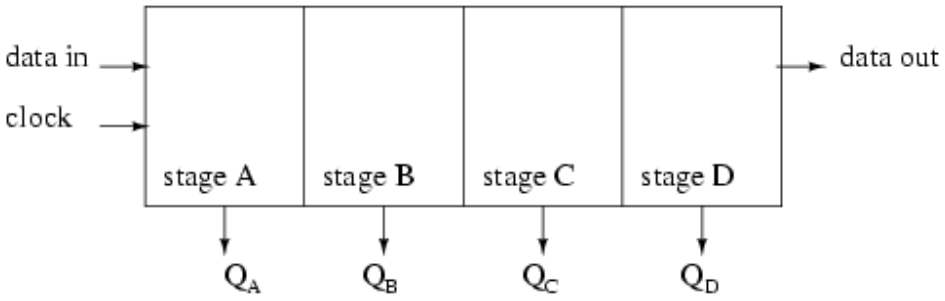
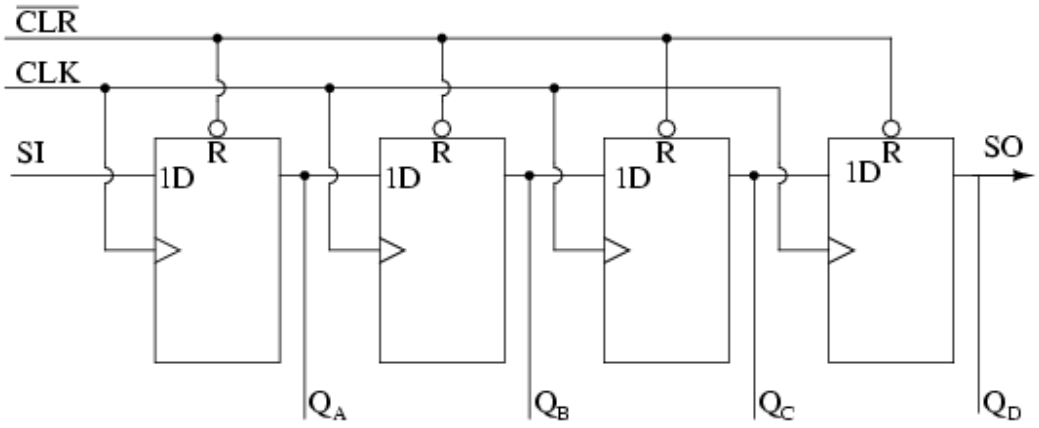
SERIAL-IN, PARALLEL-OUT SHIFT REGISTER

A serial-in/parallel-out shift register is similar to the serial-in/ serial-out shift register in that it shifts data into internal storage elements and shifts data out at the serial-out, data-out, pin. It is different in that it makes all the internal stages available as outputs. Therefore, a serial-in/parallel-out shift register converts data from serial format to parallel format. If four data bits are shifted in by four clock pulses via a single wire at data-in, below, the data becomes available simultaneously on the four Outputs Q_A to Q_D after the fourth clock pulse.



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

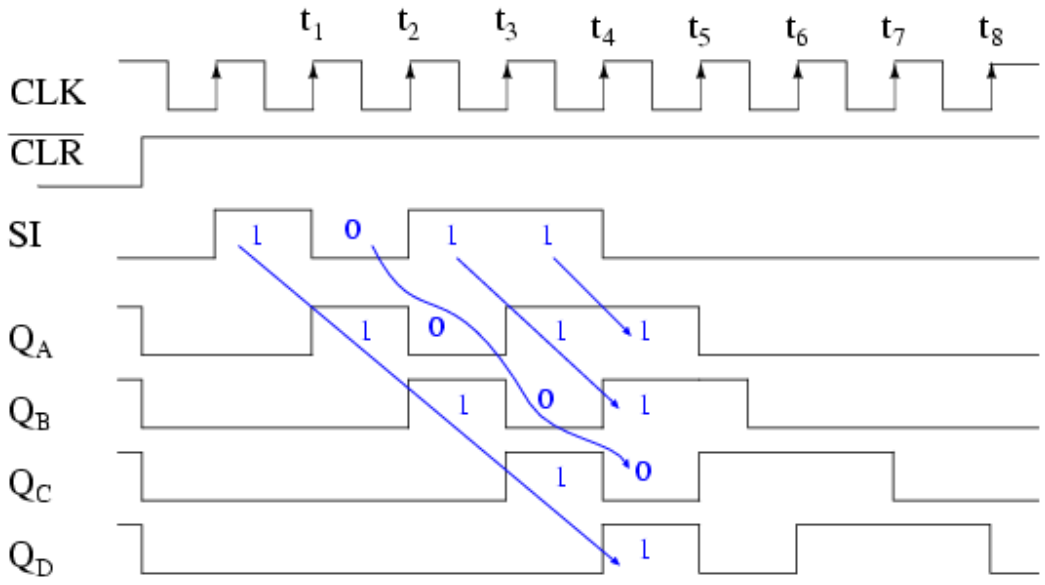
The practical application of the serial-in/parallel-out shift register is to convert data from serial format on a single wire to parallel format on multiple wires. Perhaps, we will illuminate four LEDs (Light Emitting Diodes) with the four outputs (**Q_A Q_B Q_C Q_D**).



Serial-in/ Parallel out shift register details

The above details of the serial-in/parallel-out shift register are fairly simple. It looks like a serial-in/ serial-out shift register with taps added to each stage output. Serial data shifts in at **SI** (Serial Input). After a number of clocks equal to the number of stages, the first data bit in appears at SO (Q_D) in the above figure. In general, there is no SO pin. The last stage (Q_D above) serves as SO and is cascaded to the next package if it exists.

If a serial-in/parallel-out shift register is so similar to a serial-in/ serial-out shift register, why do manufacturers bother to offer both types? Why not just offer the serial-in/parallel-out shift register? They actually only offer the serial-in/parallel-out shift register, as long as it has no more than 8-bits. Note that serial-in/ serial-out shift registers come in bigger than 8-bit lengths of 18 to to 64-bits. It is not practical to offer a 64-bit serial-in/parallel-out shift register requiring that many output pins. See waveforms below for above shift register.



Serial-in/ parallel-out shift register waveforms

The shift register has been cleared prior to any data by **CLR'**, an active low signal, which clears all type D Flip-Flops within the shift register. Note the serial data **1011** pattern presented at the **SI** input. This data is synchronized with the clock **CLK**. This would be the case if it is

being shifted in from something like another shift register, for example, a parallel-in/ serial-out shift register (not shown here). On the first clock at **t1**, the data **1** at **SI** is shifted from **D** to **Q** of the first shift register stage. After **t2** this first data bit is at $\mathbf{Q_B}$. After **t3** it is at $\mathbf{Q_C}$. After **t4** it is at $\mathbf{Q_D}$. Four clock pulses have shifted the first data bit all the way to the last stage $\mathbf{Q_D}$. The second data bit a **0** is at $\mathbf{Q_C}$ after the 4th clock. The third data bit a **1** is at $\mathbf{Q_B}$. The fourth data bit another **1** is at $\mathbf{Q_A}$. Thus, the serial data input pattern **1011** is contained in ($\mathbf{Q_D}$ $\mathbf{Q_C}$ $\mathbf{Q_B}$ $\mathbf{Q_A}$). It is now available on the four outputs.

It will available on the four outputs from just after clock $\mathbf{t_4}$ to just before $\mathbf{t_5}$. This parallel data must be used or stored between these two times, or it will be lost due to shifting out the Q_D stage on following clocks $\mathbf{t_5}$ to $\mathbf{t_8}$ as shown above.