Multiplexor

A multiplexor is a device that takes a number of data inputs and selects one of them to pass through as its output.

The interface of a multiplexor provides means to control which data input value is selected.

If there are K data input signals, then at least log K bits are needed to specify which input signal is to be passed through.

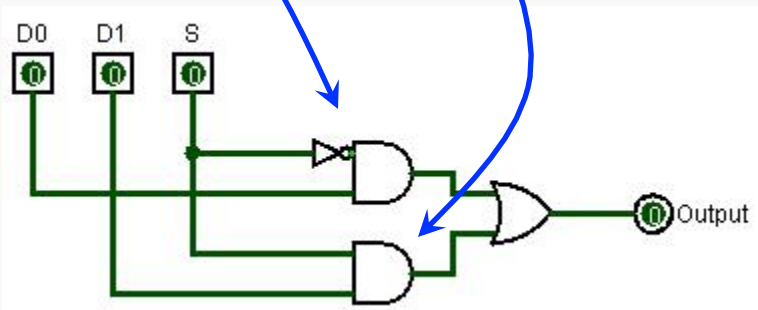
So, in most cases, multiplexors take 2^n data input signals and n control signals.

Designing a Multiplexor

Consider a 2^1 x 1 multiplexor; it takes two data inputs D0 and D1 and a single select bit S:

D0	D1	S	Output
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

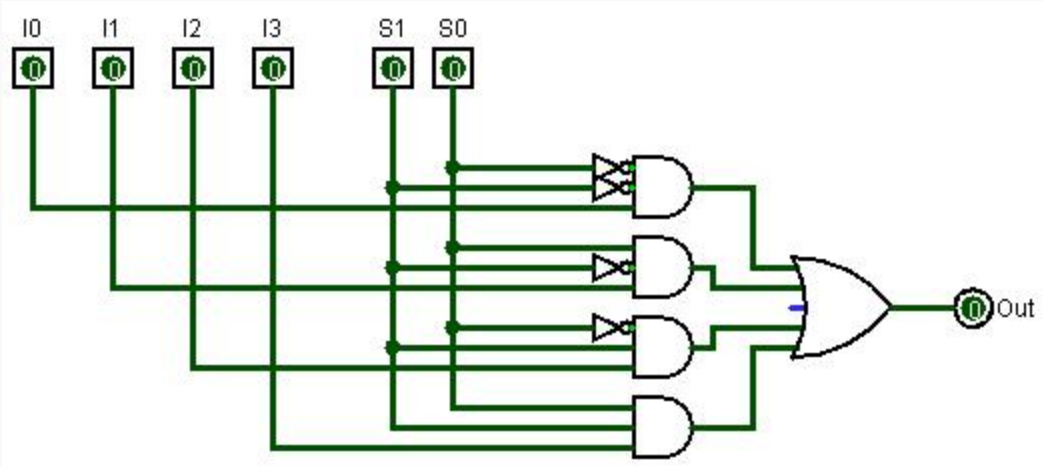
$$
\begin{aligned}
Output &= \overline{D_0} \cdot D_1 \cdot S + D_0 \cdot \overline{D_1} \cdot \overline{S} + D_0 \cdot D_1 \cdot \overline{S} + D_0 \cdot D_1 \cdot S \\
&= (\overline{D_0} + D_0) \cdot D_1 \cdot S + (\overline{D_1} + D_1) \cdot D_0 \cdot \overline{S} \\
&= D_0 \cdot \overline{S} + D_1 \cdot S
\end{aligned}
$$

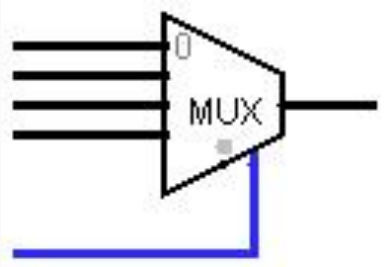


2 x 1 multiplexor

A 4x2 Multiplexor

$$Out = I_0 \cdot \overline{S_1} \cdot \overline{S_0} + I_1 \cdot \overline{S_1} \cdot S_0 + I_2 \cdot S_1 \cdot \overline{S_0} + I_3 \cdot S_1 \cdot S_0$$

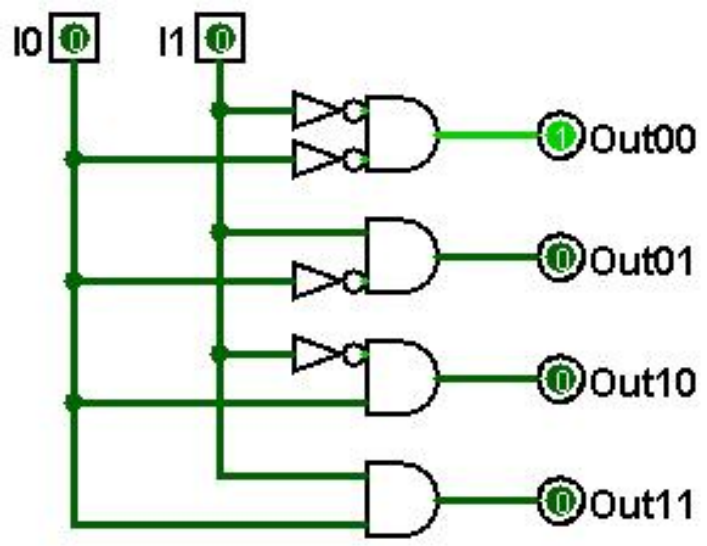


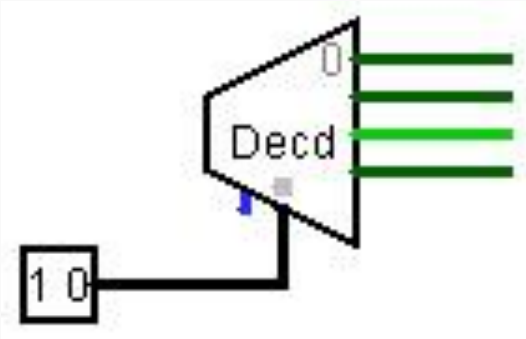


Decoders

A *decoder* selects a single data output line to set high.

Typically, there are 2^n possible destinations and, therefore, n bits to specify the destination.

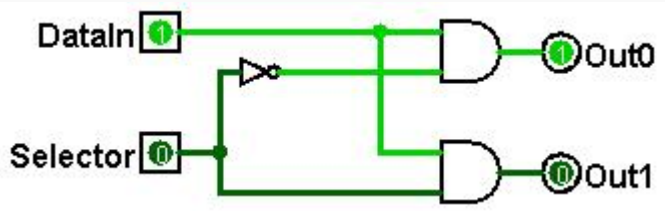


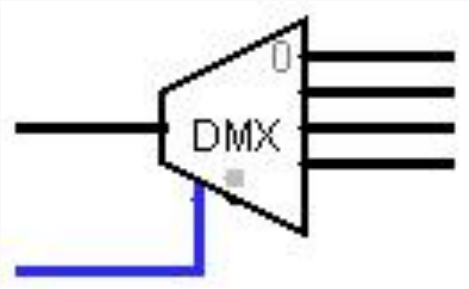


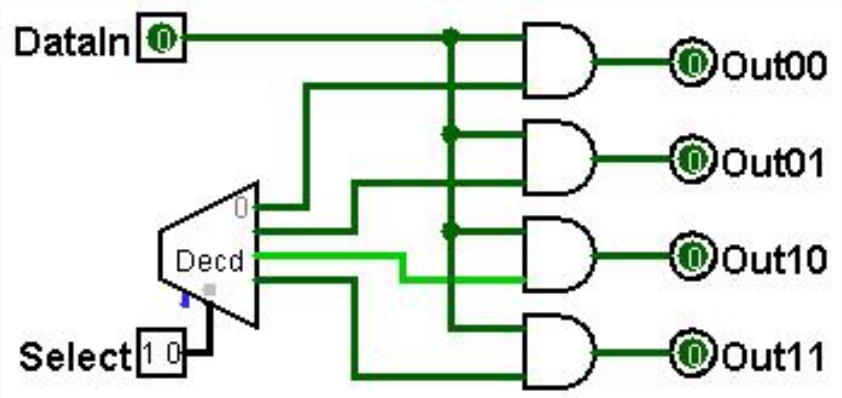
Demultiplexor

A *demultiplexor* takes a single data input and passes that input through to a single, selectable destination.

Typically, there are 2^n possible destinations and, therefore, n bits to specify the destination.







Application of Plexors

Demultplexor applied to clock signal supports selection of register to write to

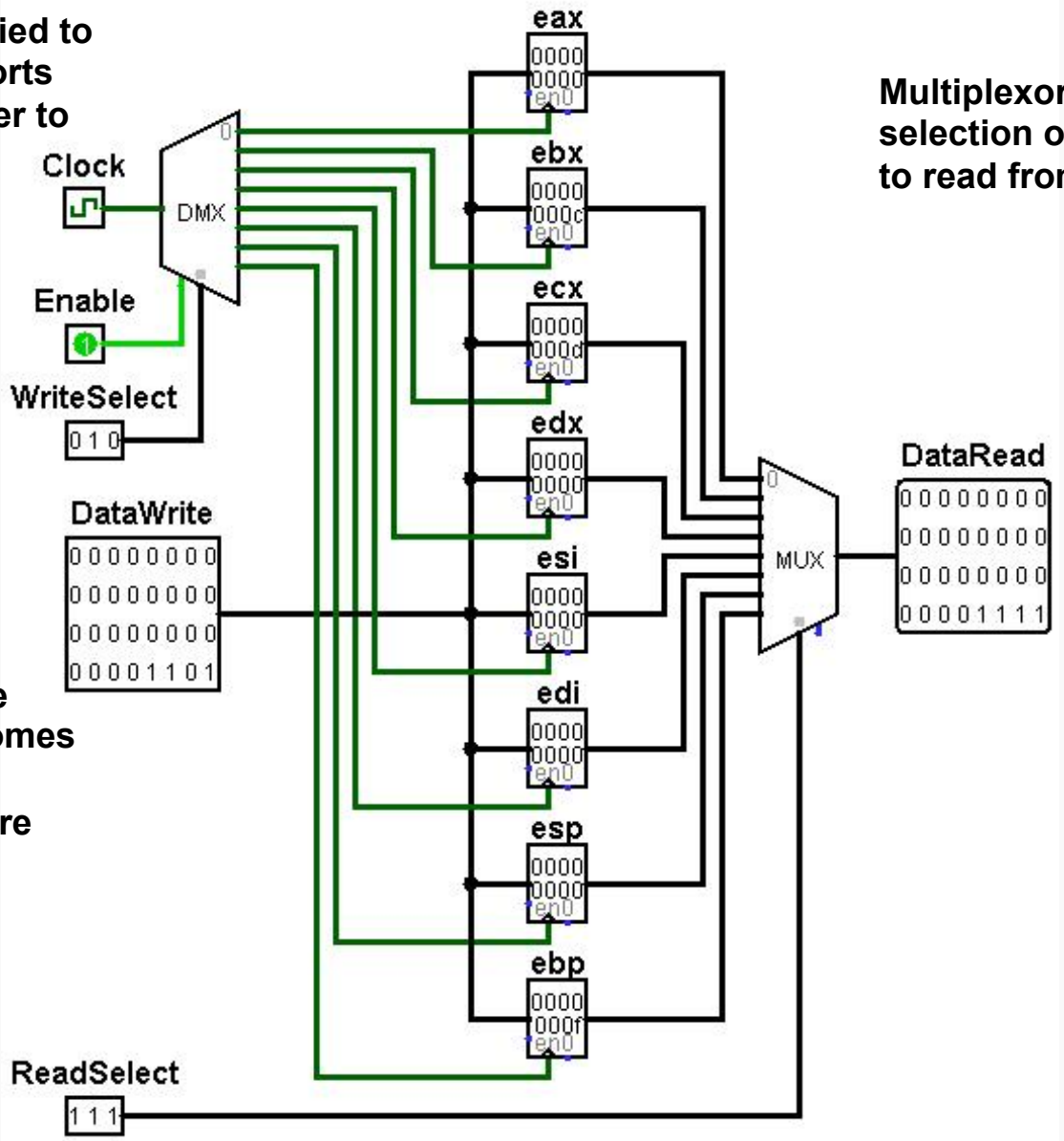
Write register number comes from machine instruction

Data to be written comes from… somewhere

Read register number comes from machine instruction

Multiplexor supports selection of register to read from

Data just read goes … somewhere

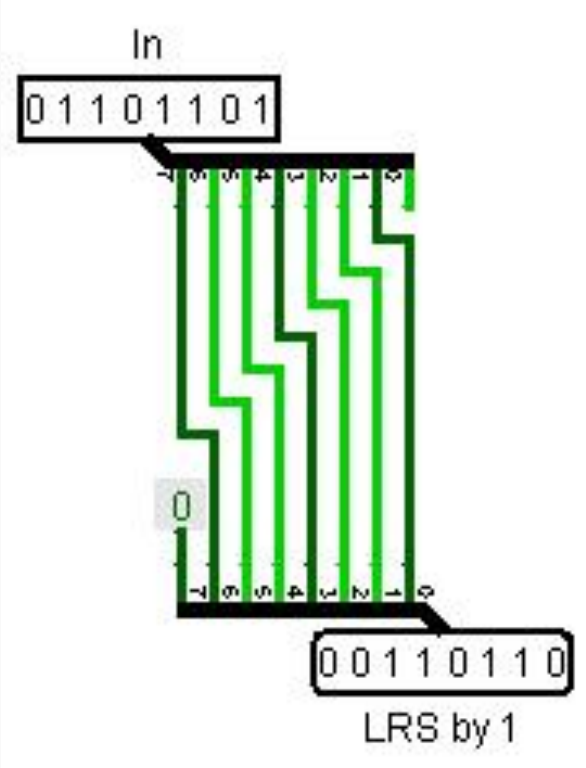


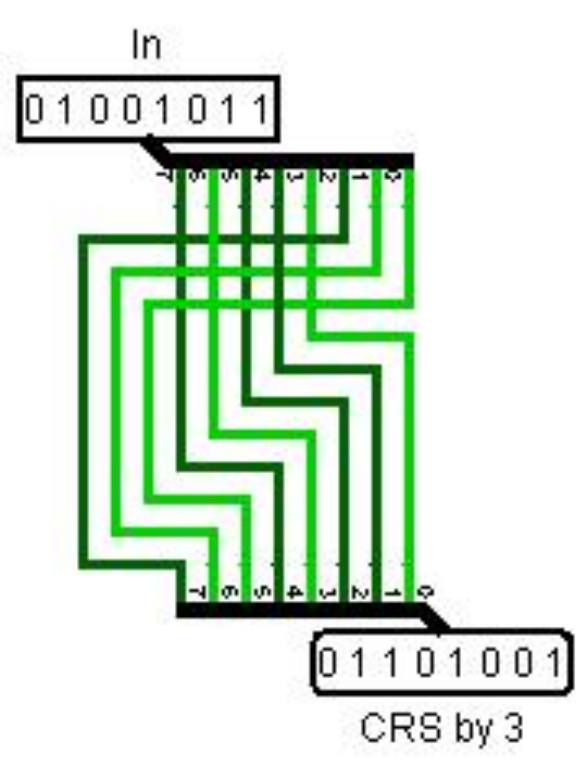
Bit Shifts

We have seen that efficient bit shifting is important because:

- bit shifts provide a simple way to perform multiplication/division
- bit shifts are often needed when applying masks to a data value

Fixed shifts are easily implemented in hardware:





Barrel Shifter

Selectable shifts can be made time-efficient by careful use of 2x1 multiplexors:

This is a 4-bit barrel shifter that supports right logical shifts of a 4-bit operand.

The operand can be shifted 0, 1, 2, or 3 positions to the right.

It requires two levels of 4 multiplexors each, with a total of 4 gate delays (ignoring inverters).

Note how the inputs to the multiplexors are arranged…