CS 240 Lab 2
More Digital Logic
and
Combinational Circuits

- **Binary and Hex Numbers/Binary Counter**
- **Multiplexer**
- **Decoder**
- **Adder**

Binary and Hexadecimal Numbers

Hex	Binary			
	QD	QC	QB	QA
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
A	1	0	1	0
B	1	0	1	1
C	1	1	0	0
D	1	1	0	1
E	1	1	1	0
F	1	1	1	1

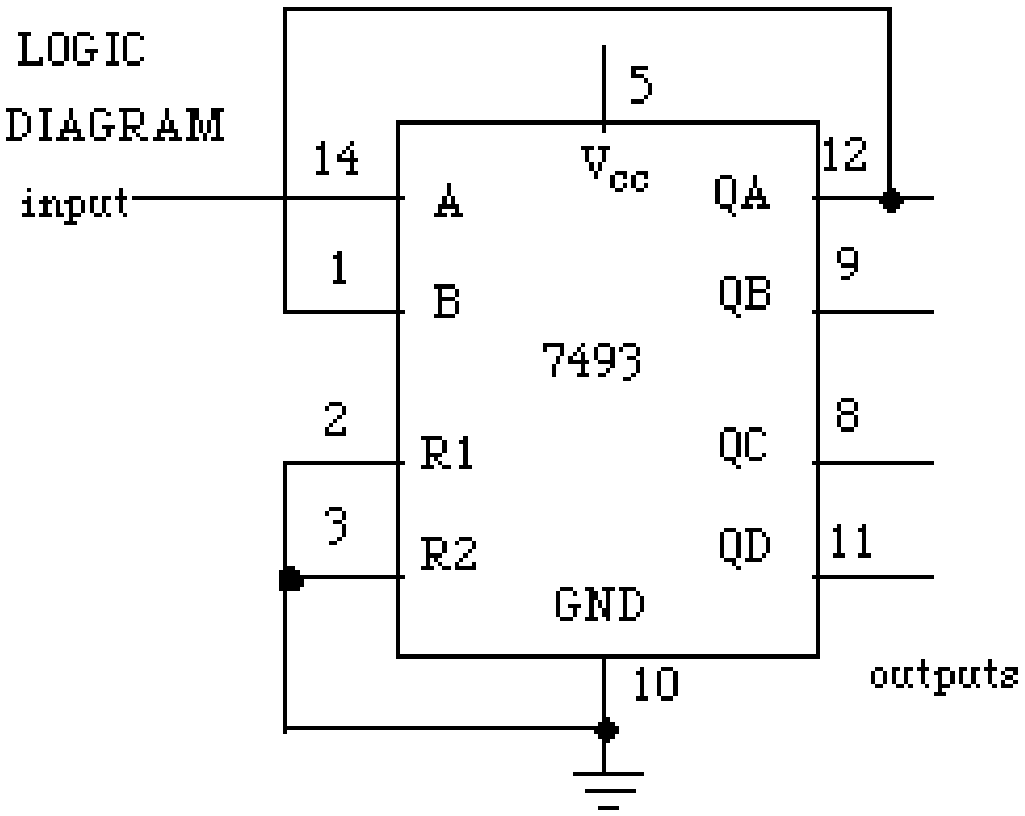
Hex can be converted to binary and vice versa by grouping into 4 bits.

$11110101_2 = F5_{16}$ $37_{16} = 00110111_2$

Binary Counter

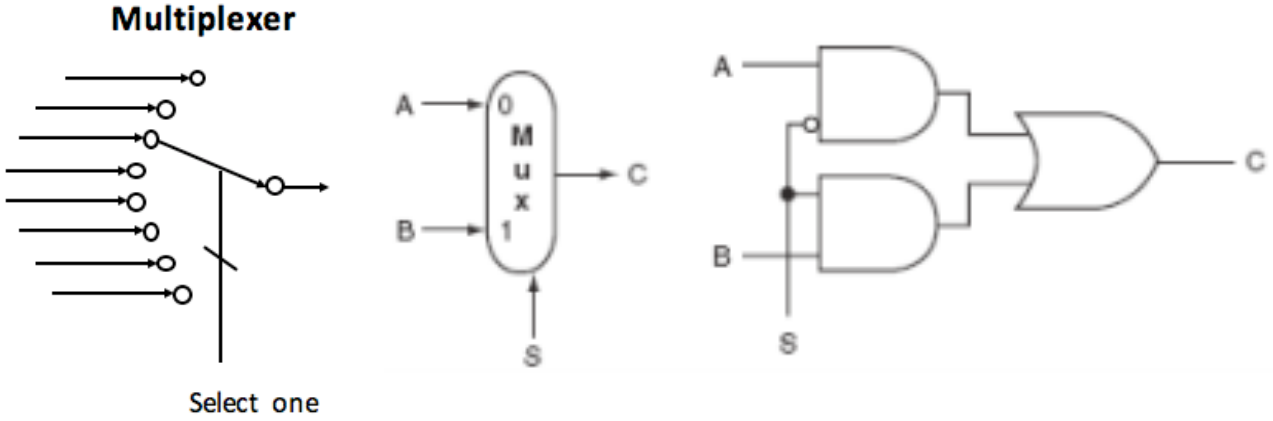
NOTE: logic diagram is not the same as pin-outs! Shows information about the logical operation of the device.

- Inputs on left side of diagram
- Outputs on right
- Voltage shown on top
- Ground shown on bottom



Multiplexer

A multiplexer can be thought of as a **selection circuit**, which steers a single input from a set of inputs through to the output, based on the select line.



- n select lines
- 2^n input lines
- 1 output

One of the possible 2^n inputs is chosen by the n select lines, and gated through to the output of a multiplexer. The truth table for an 8x1 MUX is:

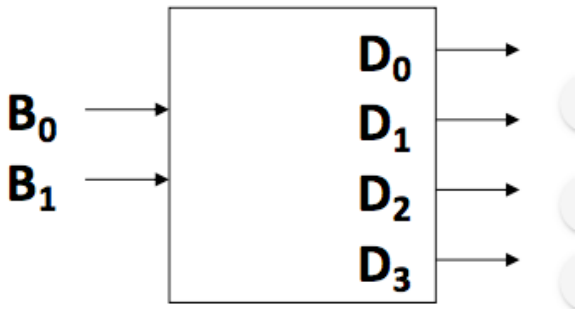
S2	S1	S0	Q
0	0	0	D0
0	0	1	D1
0	1	0	D2
0	1	1	D3
1	0	0	D4
1	0	1	D5
1	1	0	D6
1	1	1	D7

Decoder

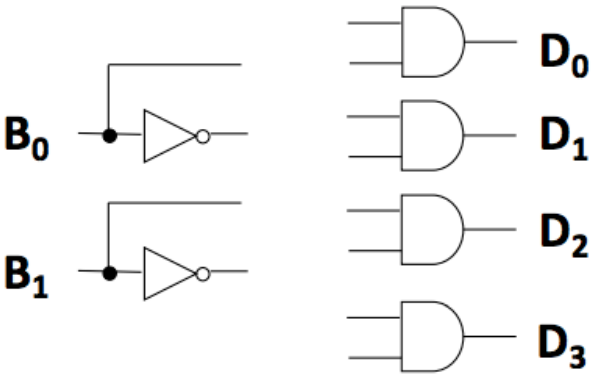
A decoder takes an n-bit binary number as an input, and asserts the corresponding numbered output from the set of 2^n outputs.

- n input/select lines
- 2^n outputs
- only one of the outputs is active at any given time, based on the value of the n select lines.

2x4 Decoder



Built with code detectors:



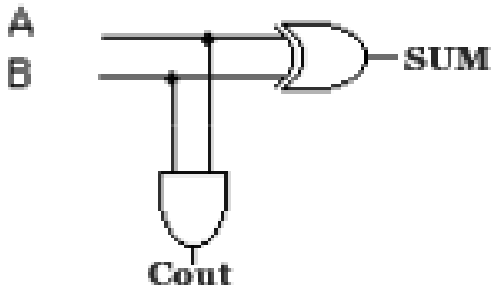
Truth table for an 3x8 decoder

S2	S1	S0		Q0	Q1	Q2	Q3	Q4	Q5	Q6	Q7
0	0	0		1	0	0	0	0	0	0	0
0	0	1		0	1	0	0	0	0	0	0
0	1	0		0	0	1	0	0	0	0	0
0	1	1		0	0	0	1	0	0	0	0
1	0	0		0	0	0	0	1	0	0	0
1	0	1		0	0	0	0	0	1	0	0
1	1	0		0	0	0	0	0	0	1	0
1	1	1		0	0	0	0	0	0	0	1

Adder

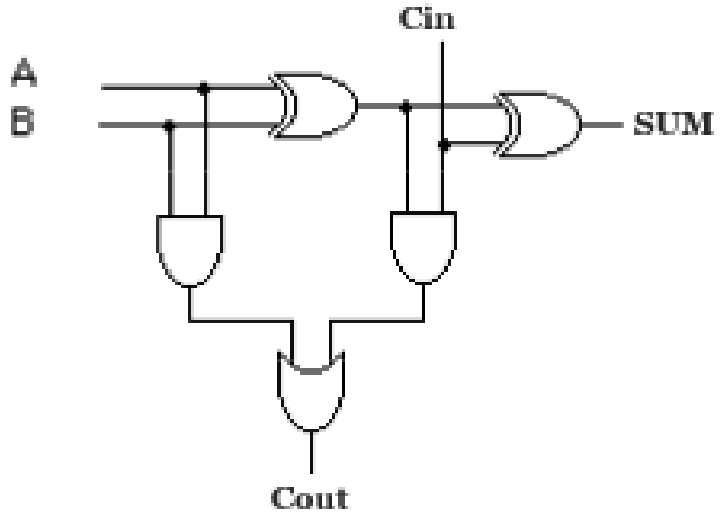
Addition is a very important arithmetic operation, and uses the Exclusive OR gate.

Half-Adder – adds two one-bit values



A	B	Cout	Sum
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Full Adder – uses two half-adders and incorporates a carry-in



Cin	A	B	Cout	Sum	
0	0	0	0	0	Sum = A⊕B⊕Cin
0	0	1	0	1	
0	1	0	0	1	
0	1	1	1	0	
1	0	0	0	1	Cout = AB+(A⊕B)Cin
1	0	1	1	0	
1	1	0	1	0	
1	1	1	1	1	