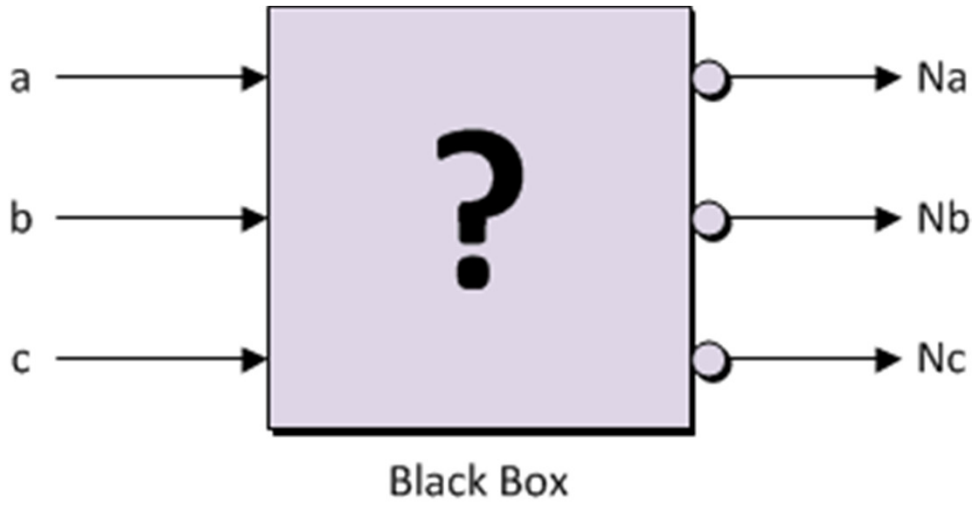
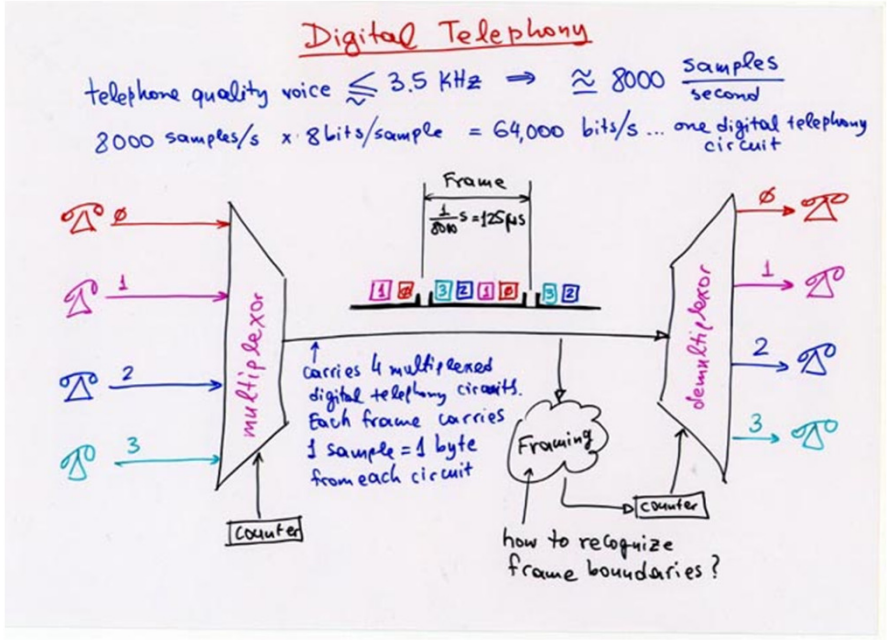
Carpe Gates!!!!...
& Diem Data
CSCI 255



http://img.deusm.com/programmableplanet/2012/06/246072/173837_783569.gif

- Previous notes (and class), we have covered the BIG 3
  - AND, OR, NOT

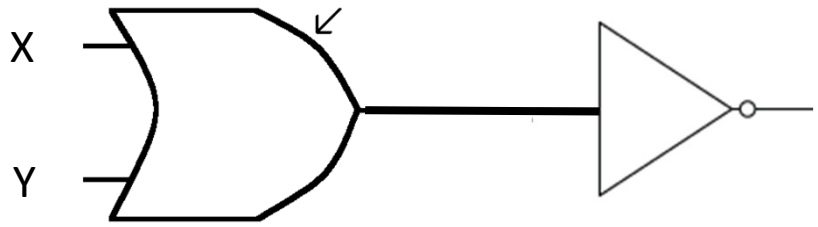
- We will look in how to derive other digital gates
  - NOR,
  - NAND,
  - XOR,
  - MUXes,
  - Decoders….



http://www.csd.uoc.gr/~hy534/05a/s23_ts.html

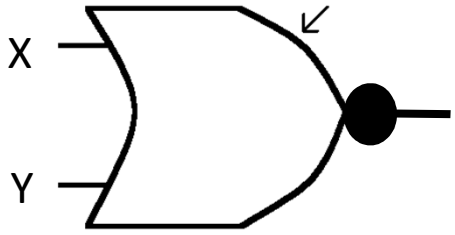
NOR-GATE

- NOR Gate
- Gate representation of an OR-gate with a complemented output
- Gate diagram & Truth table:



$$Output = \overline{(X + Y)} = X'Y'$$

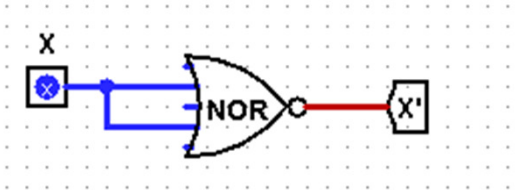
INPUT 1 (X-value)	INPUT 2 (Y-value)	OUTPUT (x NOR y)
0	0	1
0	1	0
1	0	0
1	1	0



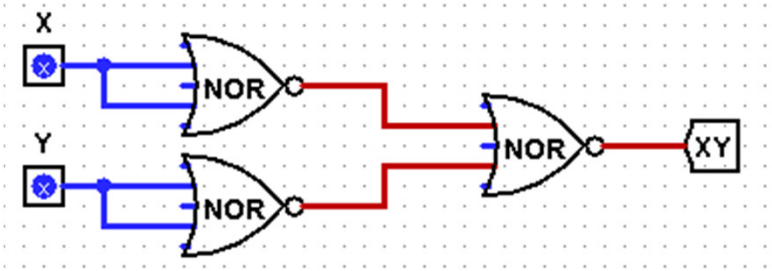
NOR-GATE

- NOR Gate(s) as:

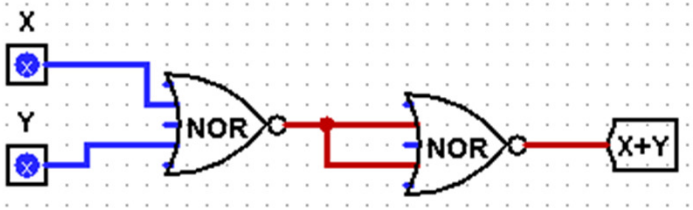
❑ NOT Gate



❑ AND Gate

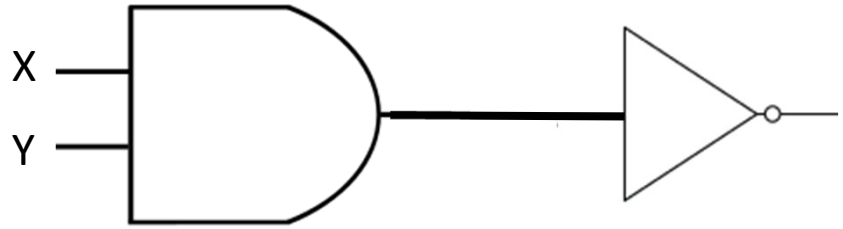


❑ OR Gate



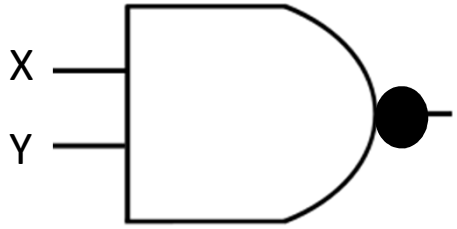
NAND-GATE

- NAND Gate
- Gate representation of an AND-gate with a complemented output
- Gate diagram & Truth table:



$$Output = \overline{(X \bullet Y)} = X' + Y'$$

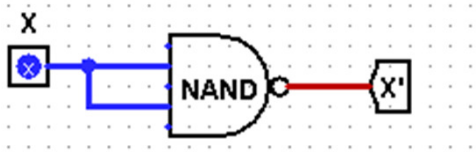
INPUT 1 (X-value)	INPUT 2 (Y-value)	OUTPUT (x NAND y)
0	0	1
0	1	1
1	0	1
1	1	0



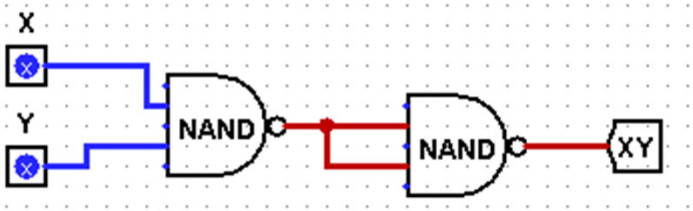
NAND-GATE

- NAND Gate(s) as:

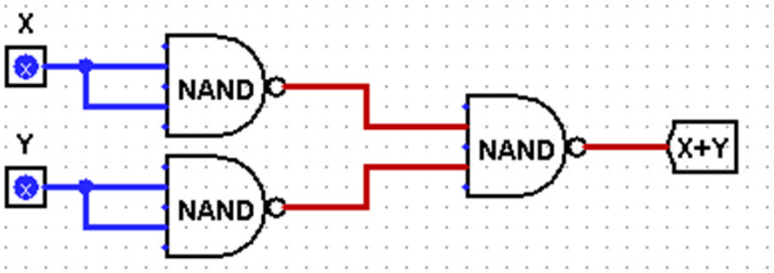
❑ NOT Gate



❑ AND Gate

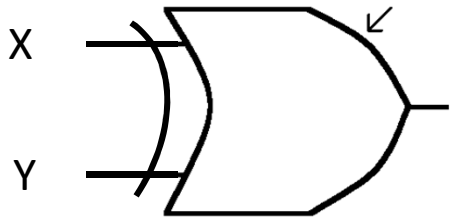


❑ OR Gate



XOR-GATE

- Exclusive-OR Gate (aka XOR)
- XOR is true when the inputs are exclusively different, false when the same
- Gate diagram:



$$Output = X \oplus Y = XY' + X'Y$$

INPUT 1 (X-value)	INPUT 2 (Y-value)	OUTPUT (x XOR y)
0	0	0
0	1	1
1	0	1
1	1	0

XOR-GATE

- What happens when applying DeMorgan's to the XOR???

$$\overline{X \oplus Y} = \overline{XY' + X'Y}$$

Complement variables and the operation in relation to them

Complement : OR -> AND

$$\overline{X \oplus Y} = (X' + Y)(X + Y')$$

"AND"-it out, (Distribute)

$$\overline{X \oplus Y} = X'X + X'Y' + YX + YY'$$

Apply Theorem 5D

$$\overline{X \oplus Y} = X'Y' + YX$$

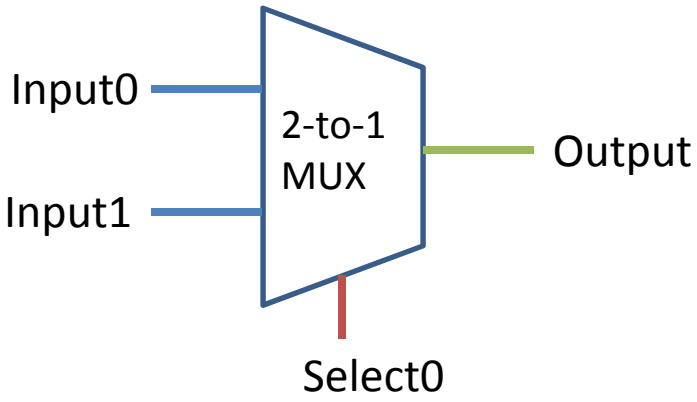
...this can be useful...

MUXes

- Multiplexers – (aka MUXes): *"Select the input you want" -- me again*
- MUXes come in different sizes, with three major components
  - Input lines
  - Single Output line
  - Selection lines
- The selection lines must satisfy:

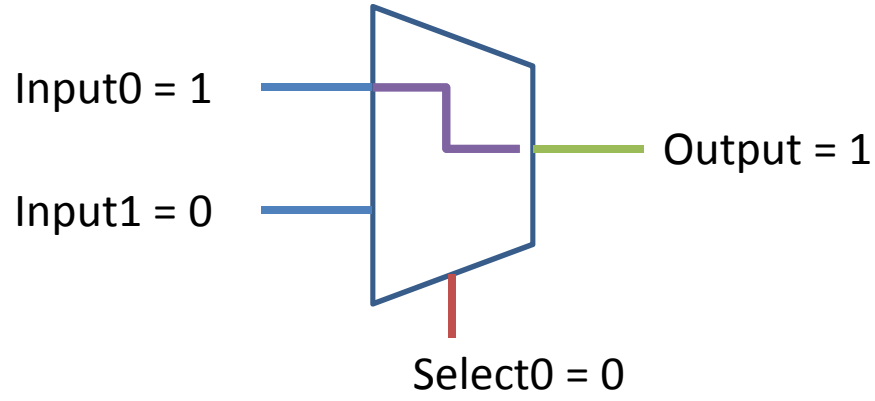
$$number\ of\ Input\ Lines = 2^{number\ of\ Selection\ Lines}$$

- Simple 2-to-1 MUX:

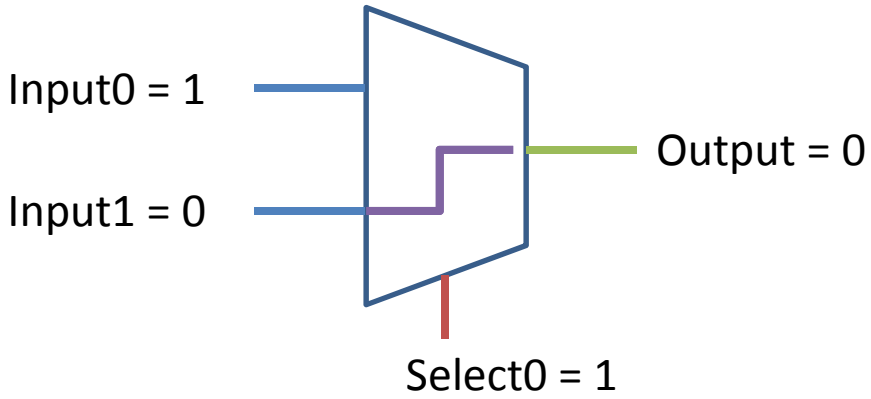


MUXes

- How does it work??



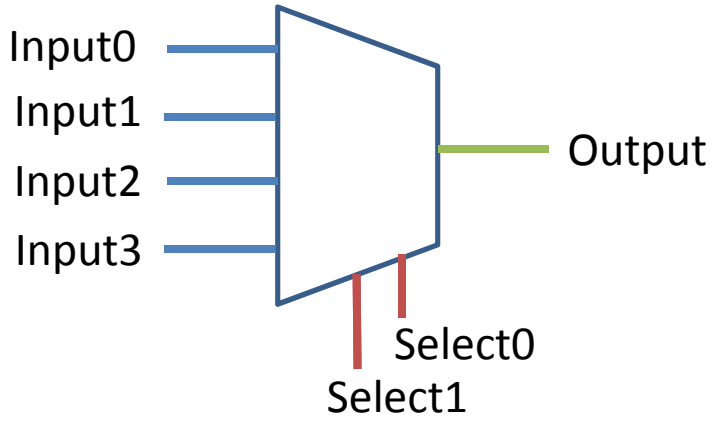
When Select line = 0; Value of Input0 with go through the Output line



When Select line = 1; Value of Input1 with go through the Output line

MUXes

- 4-to-1 MUX:



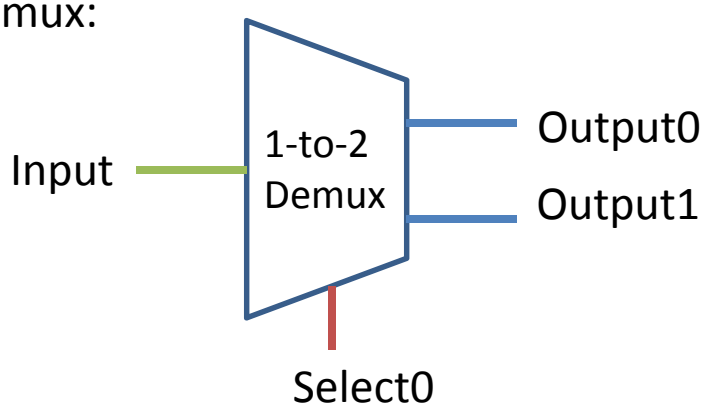
Select (Sel1:Sel0)	OUTPUT
00	Input0
01	Input1
10	Input2
11	Input3

Demux

- Demultiplexers – (aka Demuxes): *"Select the output you want" -- me again*
- Three major components
  - Input line
  - Output lines
  - Selection lines
- The selection lines must satisfy:

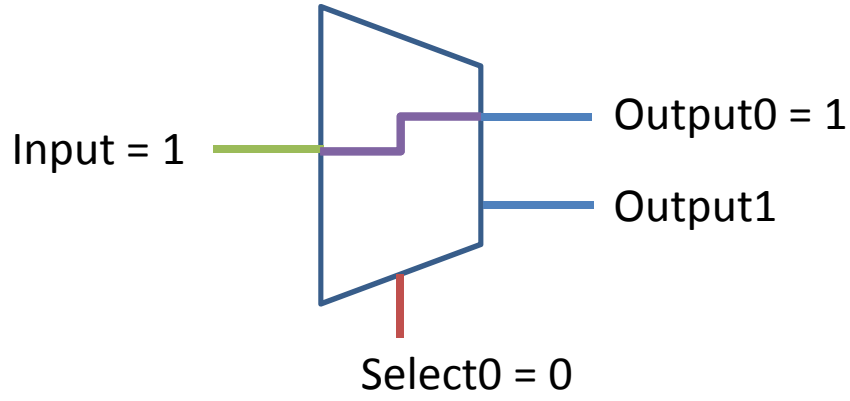
$$number\ of\ Output\ Lines = 2^{number\ of\ Selection\ Lines}$$

- Simple 1-to-2 Demux:

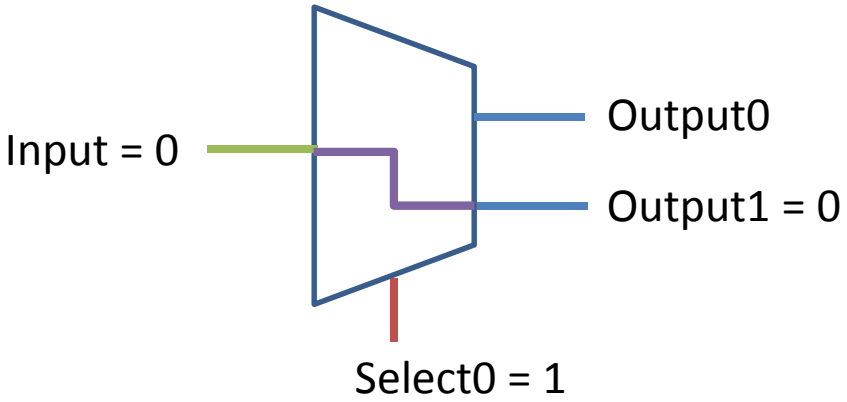


Demux

- How does it work??



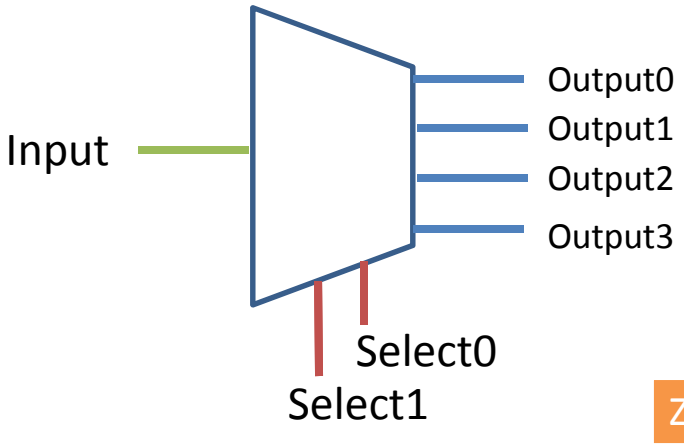
When Select line = 0; Value of Input with go through the Output0 line; other output lines do not have values



When Select line = 1; Value of Input with go through the Output1 line; other output lines do not have values

Demux

- 1-to-4 Demux:



Z-value: High Impedance

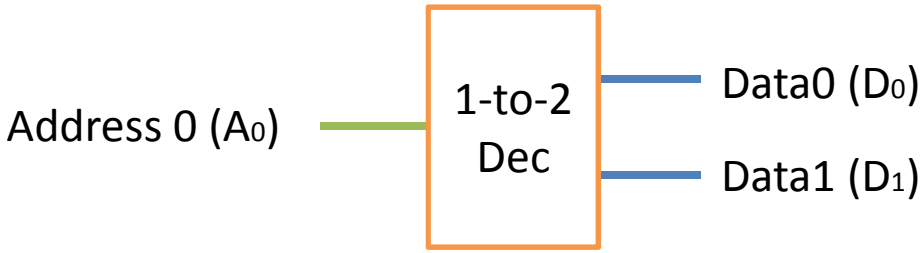
Select (Sel1:Sel0)	OUTPUT0	OUTPUT1	OUTPUT2	OUTPUT3
00	Input	Z	Z	Z
01	Z	Input	Z	Z
10	Z	Z	Input	Z
11	Z	Z	Z	Input

Decoder

- Decoder : *"Decode the incoming information in order to provide the whole message" -- me again*
- Decoders are mainly designed & utilized for RAM memory management
- Address & Data lines
- Decoders have a design that satisfies:

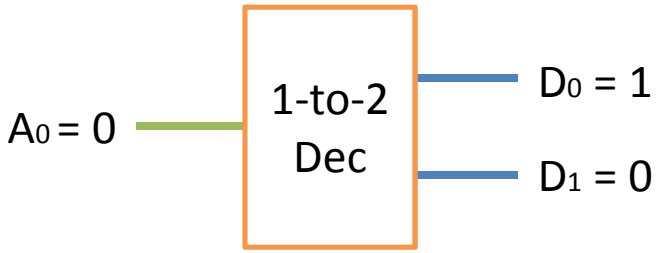
$$number\ of\ Data\ Lines = 2^{number\ of\ Address\ Lines}$$

- Simple 1-to-2 Decoder:

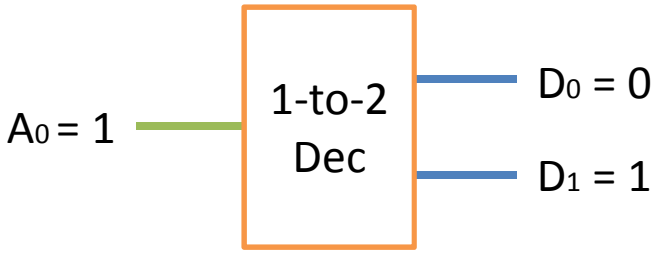


Decoder

- How does it work??



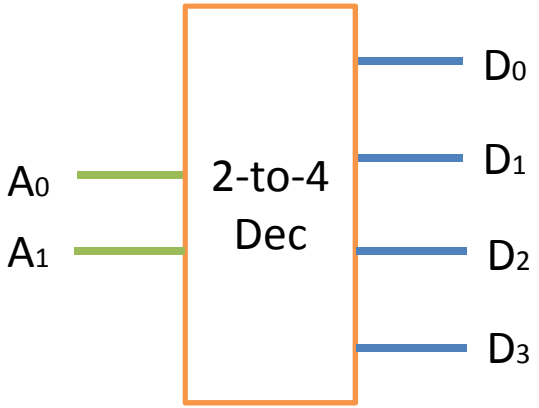
When address line = 0; First data line is true; while all other data lines are false



When address line = 1; Second data line is true; while all other data lines are false

Decoder

- 2-to-4 Demux:



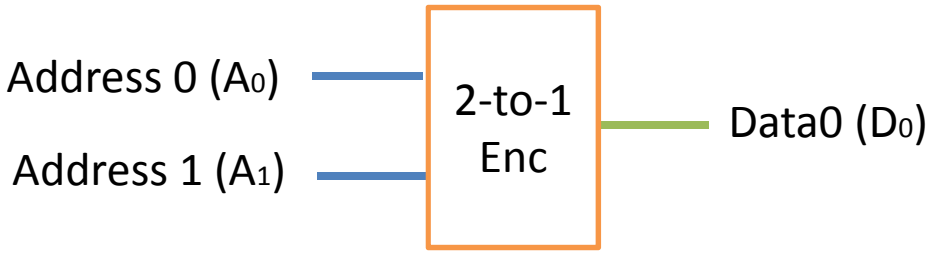
A_1	A_0	D_3	D_2	D_1	D_0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

Encoder

- Encoder : *"Duh, the opposite of the decoder…." -- me again*
- Also, mainly designed & utilized for RAM memory management
- Address & Data lines
- Encoders have a design that satisfies:

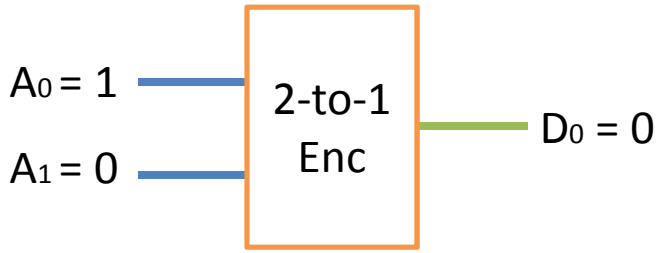
$$number\ of\ Address\ Lines = 2^{number\ of\ Data\ Lines}$$

- Simple 2-to-1 Decoder:

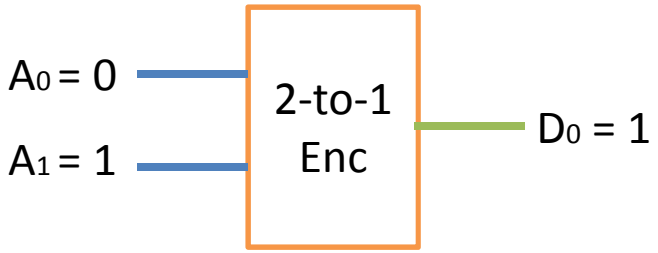


Encoder

- How does it work??



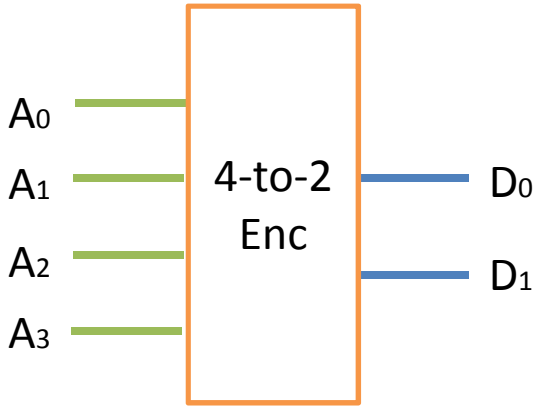
The data line value (zero) points to the address line which is true (A_0); other address lines are false



The data line value (one) points to the address line which is true (A_1); other address lines are false

Encoder

- 2-to-4 Demux:



A_3	A_2	A_1	A_0	D_1	D_0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

Gates OUT!!!!!

http://static4.businessinsider.com/image/517700336bb3f7394f000008-480/psy-bill-gates.jpg