# Lecture 5: More Gates and their Applications

Professor Peter Cheung
Department of EEE, Imperial College London

(Floyd 3.6, 6.8-6.10)
(Tocci 3.13-3.15, 4.6-4.8, 9.6-9.8)





## Points Addressed in this Lecture

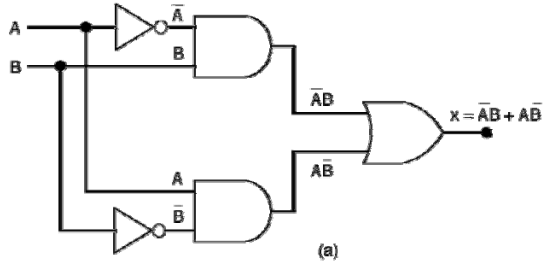
- Exclusive-OR & Exclusive NOR gates
- Usefulness of Logic Gates (as functions)
- Parity Circuits using XOR gates
- Multiplexer and Demultiplexer circuits
- Alternative logic symbols – IEEE Standard





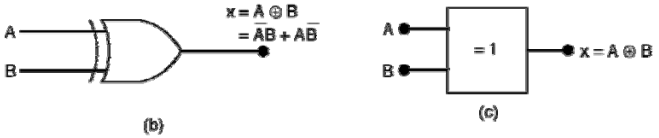
## Exclusive-OR

**Exclusive-OR (XOR) produces a HIGH output whenever the two inputs are at opposite levels.**




| A | B | x |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

XOR gate symbols

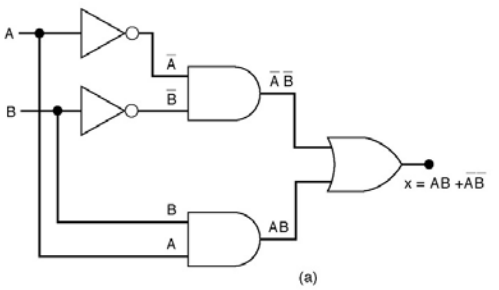






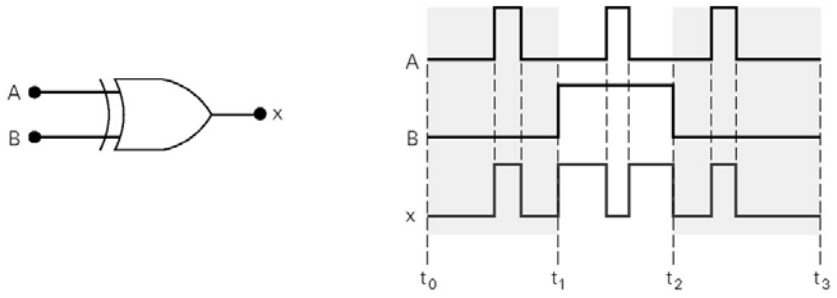

## Exclusive-XNOR

**Exclusive-NOR (XNOR) produces a HIGH output whenever the two inputs are at the same level.**




| A | B | x |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

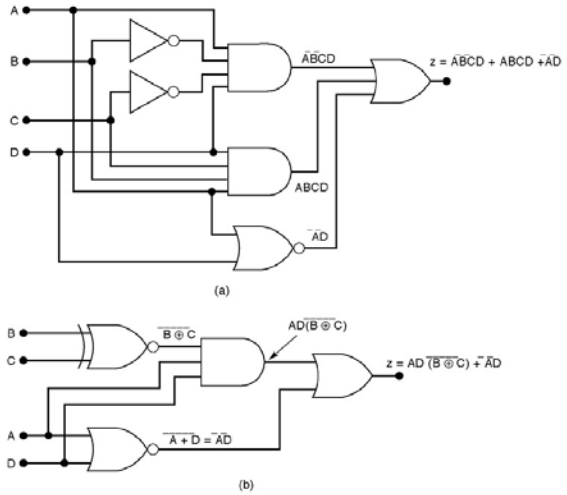
XNOR gate symbols







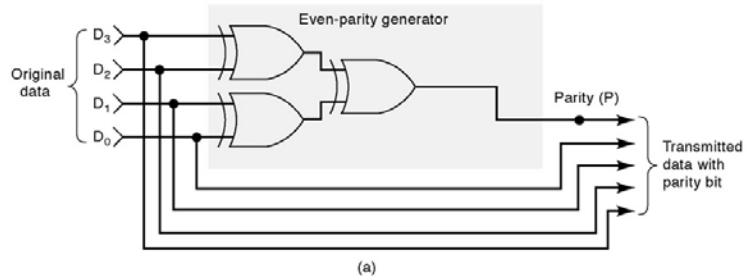

## XNOR gate may be used to simplify circuit implementation.








## Parity Generator and Checker

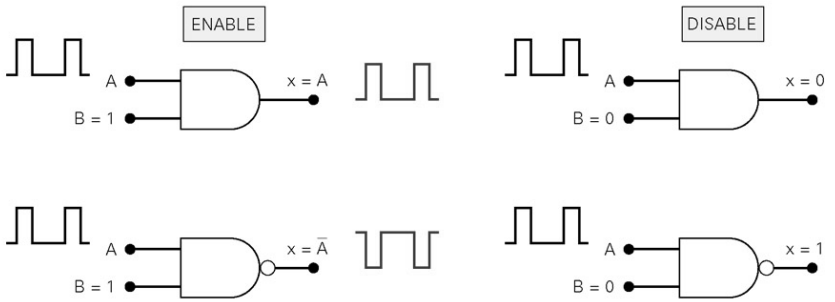






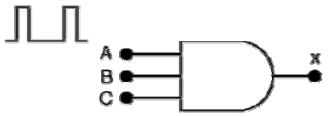

## Enable/Disable Circuits

**AND gate function act as enable/disable circuits:-**

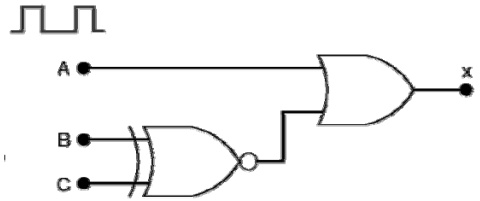


# Enable/Disable Circuits cont.

**Design a logic circuit that will allow a signal to pass to the output only when control inputs B and C are both HIGH; otherwise, the output will stay LOW.**




**Design a logic circuit that will allow a signal to pass to the output only when one, but not both, of the control inputs are HIGH; otherwise, the output will stay HIGH.**

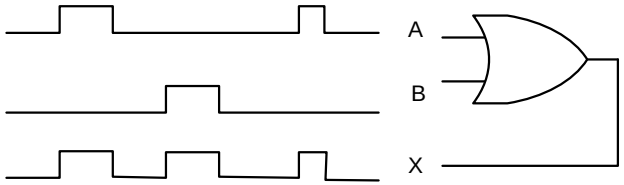




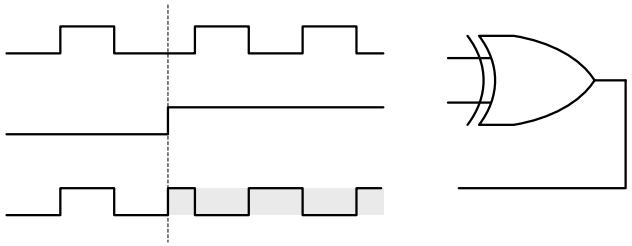



# Merging & Inversion Circuits

**OR gate performs signal merging function:-**



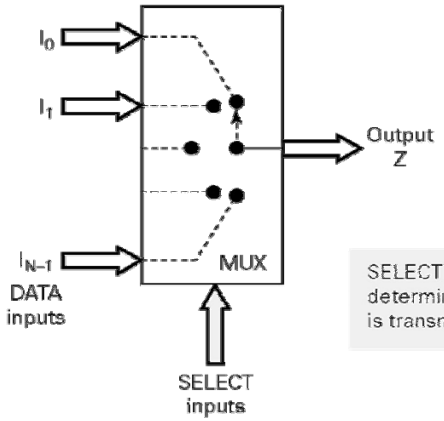

**XOR gate performs selectable inversion function:-**







# Multiplexers (Data Selectors)

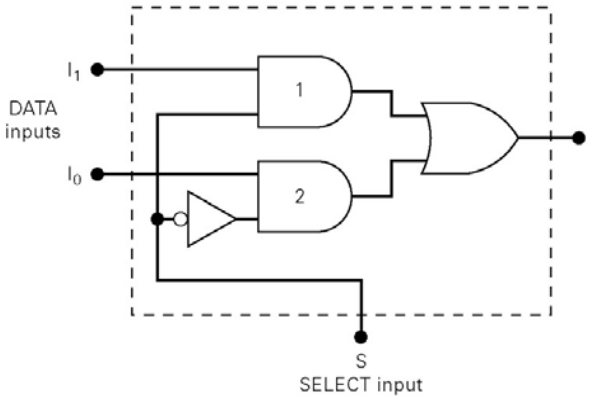



SELECT input code determines which input is transmitted to output Z.





# Multiplexers cont.

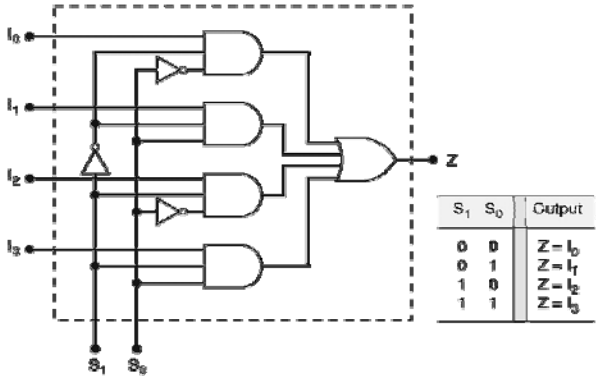



$Z = I_0 \cdot \bar{S} + I_1 \cdot S$

| S | Output |
|---|---|
| 0 | $Z = I_0$ |
| 1 | $Z = I_1$ |

Two-input multiplexer.

# Multiplexers cont.

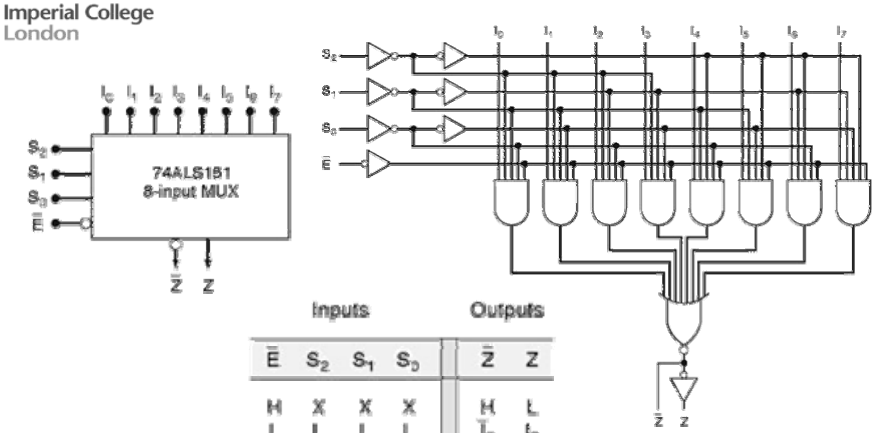



| $S_1$ | $S_0$ | Output |
|---|---|---|
| 0 | 0 | $Z = I_0$ |
| 0 | 1 | $Z = I_1$ |
| 1 | 0 | $Z = I_2$ |
| 1 | 1 | $Z = I_3$ |

**Four-input multiplexer.**







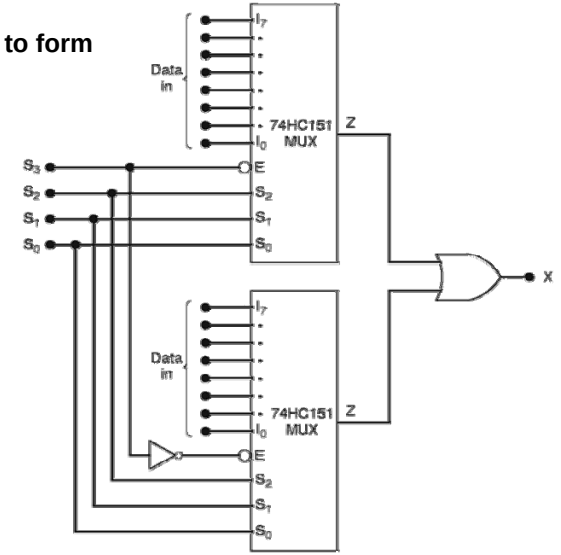

| Inputs | | | | Outputs | |
|---|---|---|---|---|---|
| $\bar{E}$ | $S_2$ | $S_1$ | $S_0$ | $\bar{Z}$ | Z |
| H | X | X | X | H | L |
| L | L | L | L | $\bar{I}_0$ | $I_0$ |
| L | L | L | H | $\bar{I}_1$ | $I_1$ |
| L | L | H | L | $\bar{I}_2$ | $I_2$ |
| L | L | H | H | $\bar{I}_3$ | $I_3$ |
| L | H | L | L | $\bar{I}_4$ | $I_4$ |
| L | H | L | H | $\bar{I}_5$ | $I_5$ |
| L | H | H | L | $\bar{I}_6$ | $I_6$ |
| L | H | H | H | $\bar{I}_7$ | $I_7$ |





# Multiplexers cont.

**two 74HC151s combined to form a 16-input multiplexer.**





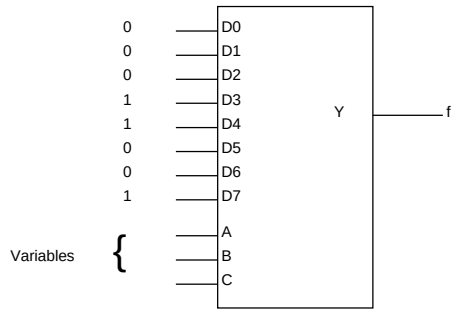



# Mux used to Implement Logic Function

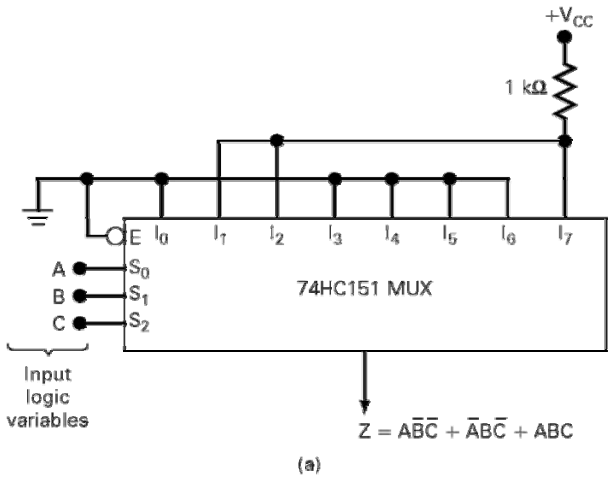
$$f = \bar{A}BC + A\bar{B}\bar{C} + ABC$$

$$f = \sum(3,4,7)$$

| A | B | C | f |
|---|---|---|---|
| 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 | 0 |
| 1 | 1 | 1 | 1 |



## Another example



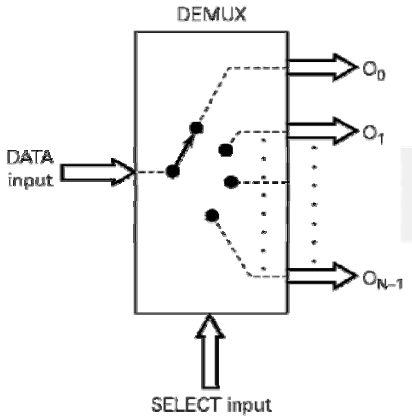

| C | B | A | Z |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |





## Demultiplexers (Data Distributors)

**A DEMUX takes a single input and distributes it over several outputs.**




DATA input is transmitted to only one of the outputs as determined by select input code

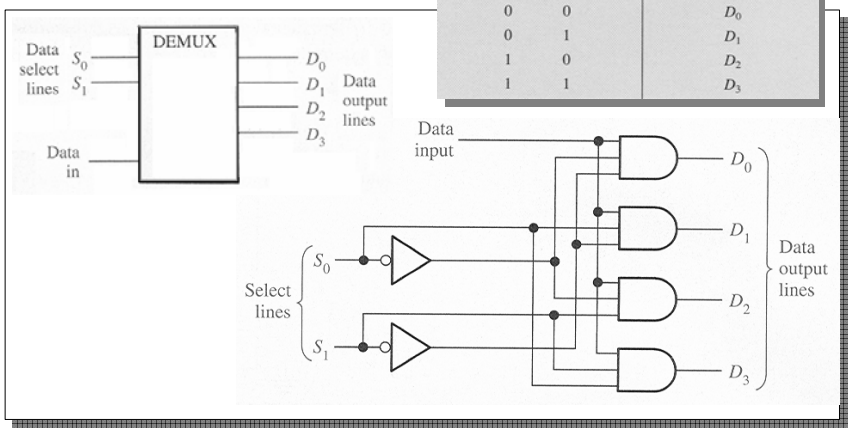




## Demultiplexers

- 2-line-to 4-line demux

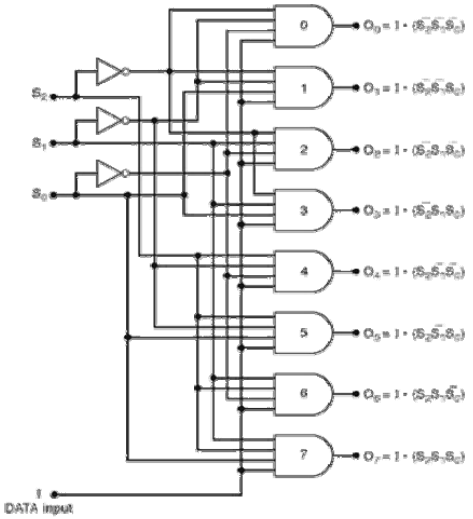
| DATA-SELECT INPUTS $S_1$ | $S_0$ | OUTPUT SELECTED |
|---|---|---|
| 0 | 0 | $D_0$ |
| 0 | 1 | $D_1$ |
| 1 | 0 | $D_2$ |
| 1 | 1 | $D_3$ |








## Demultiplexers (Data Distributors) cont.

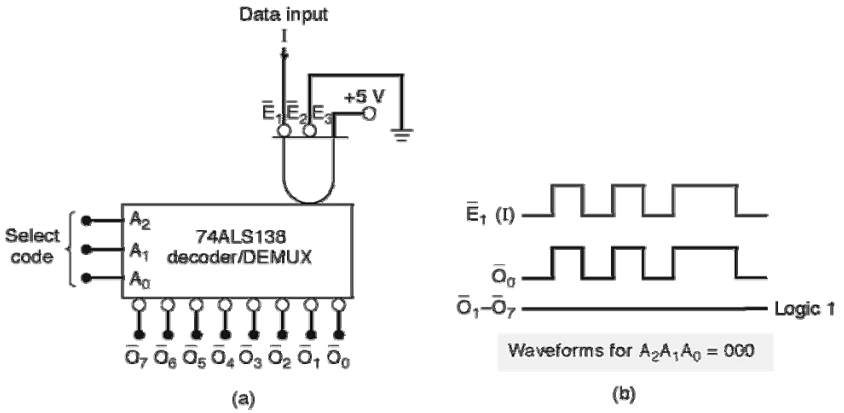



| SELECT code $S_2$ | $S_1$ | $S_0$ | OUTPUTS $O_7$ | $O_6$ | $O_5$ | $O_4$ | $O_3$ | $O_2$ | $O_1$ | $O_0$ |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**1-line-to-8-line demux**

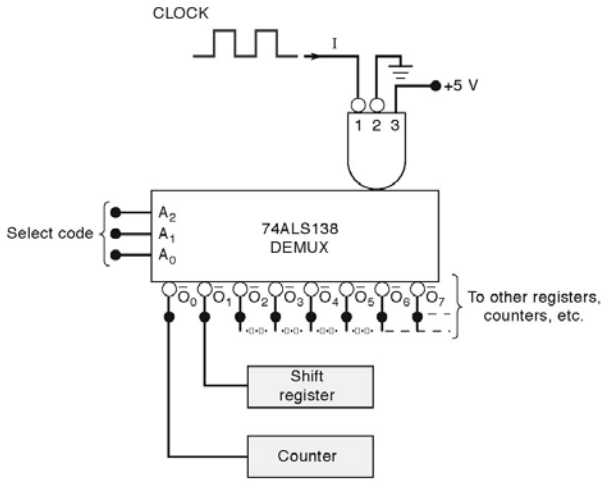
## Demultiplexers (Data Distributors) cont.

**The 74ALS138 demultiplexer with $E_1$ used as the data input. (b) Typical waveforms for a select code of $A_2 A_1 A_0 = 000$ show that $O_0$ is identical to the data input I on $E_1$.**







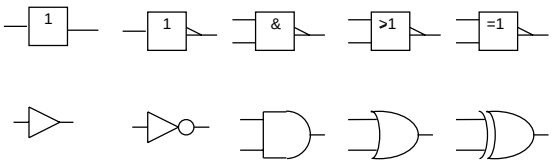

A clock demultiplexer transmits the clock signal to a destination determined by the select code inputs.





## Alternative Symbols for Gates

- The symbols presented so far are International Standards of ANSI and IEEE
- Other (older) symbols are still widely used



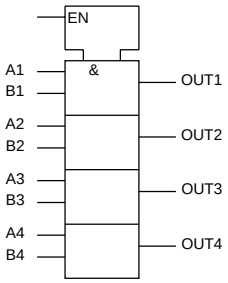

– inputs are on the left, outputs on the right
– qualifying symbol top centre:

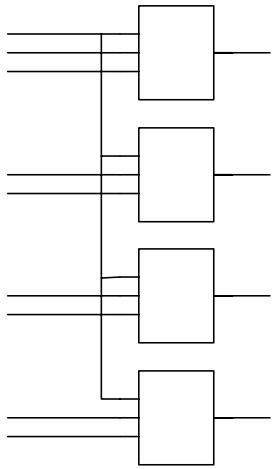
| | | | |
|---|---|---|---|
| **1** | **Straight through (buffer)** | | |
| **&** | **AND** | **≥1** | **OR** |
| **=1** | **XOR** | **Σ** | **Adder** |
| **P** | **Multiplier** | **MUX** | **Multiplexer** |





– identical elements can be grouped as an array with common control signals
– Here is a 4 identical AND gates sharing a single enable signal



EN
A1
B1

- Other Dependencies

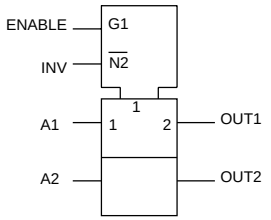
| Label | Name | On assertion ... | On de-assertion ... |
|---|---|---|---|
| EN | Enable | permits action | prevents action |
| G | AND (Gate) | permits action | forces output low |
| V | OR | forces output high | permits action |
| N | NOT (Invert) | Inverts output | No effect |
| S | Set | forces output high | No effect |
| R | Reset | forces output low | No effect |





## Numbered Dependencies

- Data inputs and outputs can be uniquely numbered
- Control input dependency labels can be followed by a number
  - indicates which inputs or outputs they affect
  - E.g.



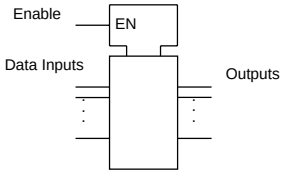

  - An array of buffers with
    - inputs (1) are ANDed with the ENABLE signal
    - outputs (2) are inverted if INV is asserted (active low)





## Active High and Active Low Inputs

- Consider a device with an additional "enable" input




- The chip is enabled if the Enable input is "asserted".
- What does asserted mean?
  - If the input is labelled EN, asserted means set Enable to 1
    - Active High
  - If the input is labelled $\overline{EN}$ , asserted means set Enable to 0
    - Active Low





## IEEE Standard symbol for 4-input MUX