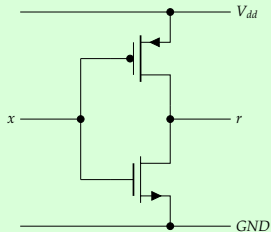


- ▶ Designing large, complex components using transistors alone will quickly give you a headache; transistors are too low-level.
- ▶ An obvious next step is to combine common transistor organisations into easier to use, *higher*-level components.
- ▶ The aim is to build **logic gates** from
  1. two **power rails** connected to  $GND$  and  $V_{dd}$ ,
  2. a **pull-up network** of P-MOSFET transistors linked to  $V_{dd}$ , and
  3. a **pull-down network** of N-MOSFET transistors linked to  $GND$and with familiar functional characteristics.

# Logic Gates (1) – NOT

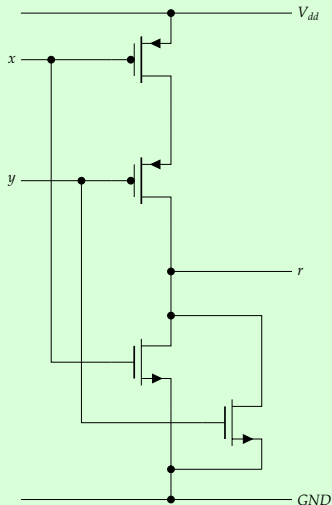
## Circuit (NOT)



- ▶ Connect  $x$  to  $GND$ :
  1. the top P-MOSFET will be connected,
  2. the bottom N-MOSFET will be disconnected,
  3.  $r$  will be connected to  $V_{dd}$ .
- ▶ Connect  $x$  to  $V_{dd}$ :
  1. the top P-MOSFET will be disconnected,
  2. the bottom N-MOSFET will be connected,
  3.  $r$  will be connected to  $GND$ .

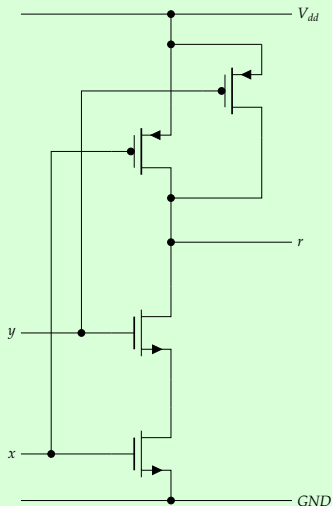
## Logic Gates (2) – NOR

### Circuit (NOR)



- ▶ Connect both  $x$  and  $y$  to  $GND$ :
  1. both top P-MOSFETs will be disconnected,
  2. both bottom N-MOSFETs will be disconnected,
  3.  $r$  will be connected to  $V_{dd}$ .
- ▶ Connect  $x$  to  $V_{dd}$  and  $y$  to  $GND$ :
  1. the upper-most P-MOSFET will be disconnected,
  2. the left-most N-MOSFET will be connected,
  3.  $r$  will be connected to  $GND$ .
- ▶ Connect  $x$  to  $GND$  and  $y$  to  $V_{dd}$ :
  1. the lower-most P-MOSFET will be disconnected,
  2. the right-most N-MOSFET will be connected,
  3.  $r$  will be connected to  $GND$ .
- ▶ Connect both  $x$  and  $y$  to  $V_{dd}$ :
  1. both top P-MOSFETs will be disconnected,
  2. both bottom N-MOSFETs will be connected,
  3.  $r$  will be connected to  $GND$ .

### Circuit (NAND)



- ▶ Connect both  $x$  and  $y$  to  $GND$ :
  1. both top P-MOSFETs will be connected,
  2. both bottom N-MOSFETs will be disconnected,
  3.  $r$  will be connected to  $V_{dd}$ .
- ▶ Connect  $x$  to  $V_{dd}$  and  $y$  to  $GND$ :
  1. the right-most P-MOSFET will be connected,
  2. the upper-most N-MOSFET will be disconnected,
  3.  $r$  will be connected to  $V_{dd}$ .
- ▶ Connect  $x$  to  $GND$  and  $y$  to  $V_{dd}$ :
  1. the left-most P-MOSFET will be connected,
  2. the lower-most N-MOSFET will be disconnected,
  3.  $r$  will be connected to  $V_{dd}$ .
- ▶ Connect both  $x$  and  $y$  to  $V_{dd}$ :
  1. both top P-MOSFETs will be disconnected,
  2. both bottom N-MOSFETs will be connected,
  3.  $r$  will be connected to  $GND$ .

Hang on! This sounds very familiar; if we

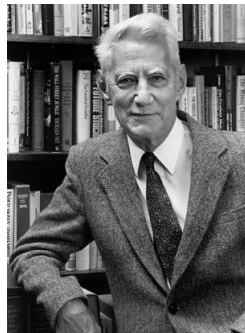
1. assume the power rails are “everywhere”,
2. relabel  $GND$  and  $V_{dd}$  using 0 and 1, and
3. describe the operation of each logic gate using a truth table

then we get a **standard cell library** (i.e., a set of standard gates) which is *very* close to the underlying theory, i.e.,

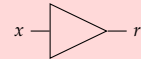
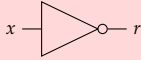





NOT	
$x$	$r$
0	1
1	0

NAND		
$x$	$y$	$r$
0	0	1
0	1	1
1	0	1
1	1	0

NOR		
$x$	$y$	$r$
0	0	1
0	1	0
1	0	0
1	1	0

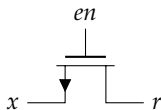


## Definition (Boolean algebra vs. logic gates)

$r \text{ is } x$	$\equiv$	$r = x$	$\equiv$	
$r \text{ is NOT } x$	$\equiv$	$r = \neg x$	$\equiv$	
$r \text{ is } x \text{ NAND } y$	$\equiv$	$r = x \bar{\wedge} y$	$\equiv$	
$r \text{ is } x \text{ NOR } y$	$\equiv$	$r = x \bar{\vee} y$	$\equiv$	
$r \text{ is } x \text{ AND } y$	$\equiv$	$r = x \wedge y$	$\equiv$	
$r \text{ is } x \text{ OR } y$	$\equiv$	$r = x \vee y$	$\equiv$	
$r \text{ is } x \text{ XOR } y$	$\equiv$	$r = x \oplus y$	$\equiv$	

## Physical Limitations (1) – 3-state Logic

- ▶ You can (more or less) connect any number of inputs to any output but you can't connect two outputs together ...
- ▶ ... to cope, we can use **3-state logic** by introducing an “extra” logic value, i.e.,
  1. 0 represents **false**,
  2. 1 represents **true**, and
  3. Z represents **high impedance**.
- ▶ Think of high impedance as being the null value; the idea is to allow part of a circuit to be “disconnected”, e.g., to allow two parts to share the same wire ...



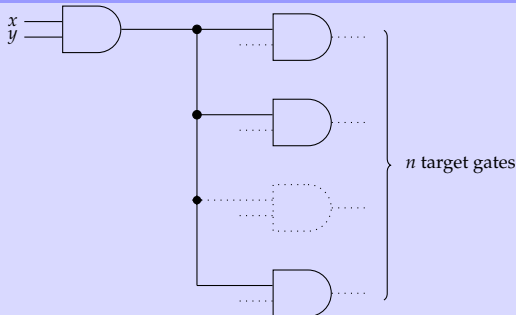
ENABLE		
$x$	$en$	$r$
?	0	Z
0	1	0
1	1	1
Z	1	Z

- ▶ ... so the ENABLE “gate” is basically just a switch.

## Physical Limitations (2) – Fan-in and Fan-out

- ▶ Logic gates have (fairly) strict operating parameters inherited from the transistors they are built from ...
- ▶ ... the terms **fan-in** and **fan-out** roughly capture the number of logic gates which are connected to a given input or output:

### Example (fan-out)



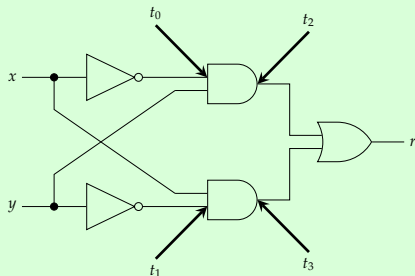


- ▶ This issue of **time** is important:
  1. **wire delay**, where it takes some time for a signal to travel from one end of a wire to the other, and
  2. **gate delay**, where it takes some time for transistors within a gate to switch, i.e., it takes some time for gate to produce an output from given inputs,combine to determine **propagation delay**.
- ▶ An important feature of a circuit is the **critical path**:
  - ▶ This is the longest sequential path through the circuit; it acts as a constraint on the time taken to generate a result.
  - ▶ Minimising the critical path (e.g., by redesigning the circuit) is an important optimisation step wrt. improved performance.

## Physical Limitations (4) – Delay

- ▶ We can implement XOR by using other gate types ...

### Circuit (XOR)



### Example

Consider setting  $x = 0$  and  $y = 1$ ; the circuit computes

$x$	=		=	0
$y$	=		=	1
$t_0$	=	$\neg x$	=	1
$t_1$	=	$\neg y$	=	0
$t_2$	=	$t_0 \wedge y$	=	1
$t_3$	=	$t_1 \wedge x$	=	0
$r$	=	$t_2 \vee t_3$	=	1

but this is a *static* view of computation in the sense that we assume all signals eventually just propagate through the circuit.

- ▶ ... but note that this isn't the only way to construct an XOR gate!

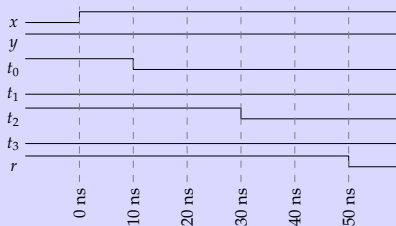
## Physical Limitations (5) – Delay

### Example

Including gate delay gives a *dynamic* view computation; imagine the delay of

1. a NOT gate is 10 ns,
2. an AND gate is 20 ns, and
3. an OR gate is 20 ns

and then flip  $x = 0, y = 1$  to  $x = 1, y = 1$ :



Note that

1. the circuit takes some time matching the **critical path** (i.e, 50 ns through a NOT, an AND and an OR gate) to settle into the right state, and as a result
2. at some points in time, the output doesn't match the inputs.

## ► Take away points:

1. We've now got logic gates where it's clear
  - 1.1 what their *behavioural* properties are since they relate directly to transistors, **and**
  - 1.2 what their *functional* properties are since they relate directly to Boolean algebra,  
i.e., there is no “magic” going on behind the scenes.
2. The tough bit is design and optimisation of circuits; that is, how do we now design a circuit to
  - 2.1 match some specification, e.g., “implement some function  $f$ ”, **and**
  - 2.2 match some design goals, e.g., “use less than  $X$  gates (or  $Y$  transistors)” or “ensure a delay less than  $Z$  ns”.

## References and Further Reading

- [1] Wikipedia: Boolean logic.  
[http://en.wikipedia.org/wiki/Boolean\\_logic](http://en.wikipedia.org/wiki/Boolean_logic).
- [2] Wikipedia: Logic gate.  
[http://en.wikipedia.org/wiki/Logic\\_gate](http://en.wikipedia.org/wiki/Logic_gate).
- [3] D. Page.  
[Chapter 2: Basics of digital logic](#).  
In *A Practical Introduction to Computer Architecture*. Springer-Verlag, 1st edition, 2009.
- [4] W. Stallings.  
[Chapter 11: Digital logic](#).  
In *Computer Organisation and Architecture*. Prentice-Hall, 9th edition, 2013.
- [5] A.S. Tanenbaum.  
[Section 3.1: Gates and Boolean algebra](#).  
In *Structured Computer Organisation* [7].
- [6] A.S. Tanenbaum.  
[Section 3.2: Basic digital logic circuits](#).  
In *Structured Computer Organisation* [7].
- [7] A.S. Tanenbaum.  
[Structured Computer Organisation](#).  
Prentice-Hall, 6th edition, 2012.