Gate Networks

course	
area	ucla
created	@January 17, 2023 2:15 PM
updated	@January 17, 2023 3:49 PM
✅ tasks	
📕 courses	
URL	
tags	cs

Definitions

▼

Big Ideas

▼ Combinational Module (Gate Networks)

- ▼ created with Gate Networks
- ▼ eg. set {AND,OR} gate network

Resources

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/4a2e7da1-0bfa-476f-b5fe-56eee9bf2fa1/ch4.pdf

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/338429c3-6e7f-4eed-9458-cd59d1a6bff4/cs51a_CriticalPath.pdf

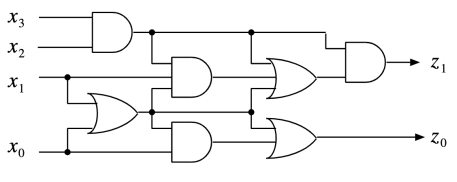


Figure 4.2: A gate network

▼ logic diagram (graphical representation)

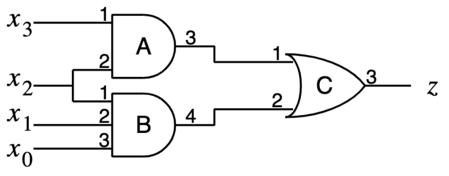
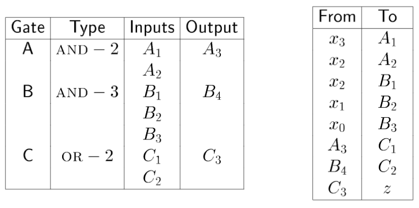


Figure 4.4: a) Graphical representation (logic diagram)

▼ net list (tabular representation)



Gate	Type	Inputs	Output
A	AND − 2	A_1	A_3
		A_2	
B	AND − 3	B_1	B_4
		B_2	
		B_3	
C	OR − 2	C_1	C_3
		C_2	

Gates

From	To
x_3	A_1
x_2	A_2
x_2	B_1
x_1	B_2
x_0	B_3
A_3	C_1
B_4	C_2
C_3	z

Connections

(b)

▼ Hardware Description Language (HDL) (program)

```
A_3 <= x3 and x2;
B_4 <= x2 and x1 and x0;
C_3 <= A_3 or B_4;
z   <= C_3;
```

(c)

▼ Universal Gates: {NAND, NOR}

▼ Universal Sets

- ▼ you can show a set of gates of a combinatorial module (gate network) is universal if it can be represented using only universal gates {NAND, NOR}
- ▼ e.g. NOT (inverter) and AND using universal gates

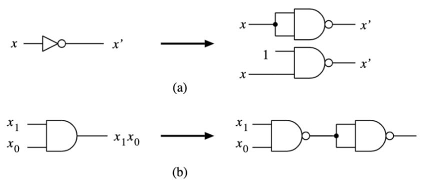
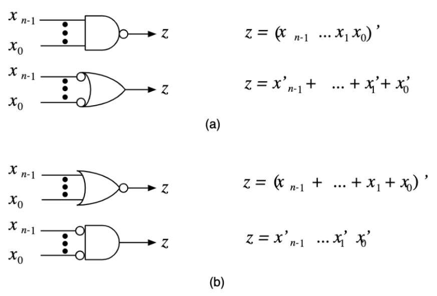


Figure 4.7: Implementations with NAND gates: a) NOT; b) AND

▼ Mixed-logic Notation and Complex gate structures

- ▼ e.g. showing gates as re-implemented unique gates



x_{n-1}
x_0
z
$z = (x_{n-1} \dots x_1 x_0)'$
$z = x'_{n-1} + \dots + x'_1 + x'_0$
(a)
$z = (x_{n-1} + \dots + x_1 + x_0)'$
$z = x'_{n-1} \dots x'_1 x'_0$
(b)

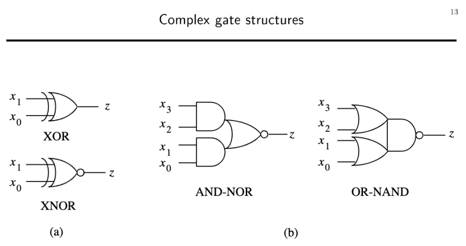
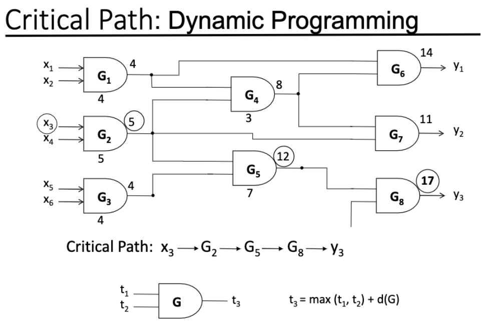


Figure 4.9: Additional gates in CMOS a) XOR and XNOR, b) Complex gate structures: AND-OR and OR-AND

▼ Critical Path: Dynamic Programming - Longest Path

▼ gates have delay time in execution → finding the largest latency in the gate network tells us how fast the network will execute (the slowest chain)

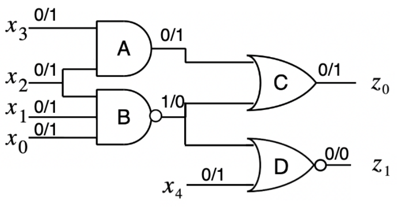
▼ the latency can be written above the gate


Critical Path: Dynamic Programming
x_1
x_2
x_3
x_4
x_5
x_6
G_1
G_2
G_3
G_4
G_5
G_6
G_7
G_8
4
5
12
14
11
17
8
3
7
y_1
y_2
y_3
Critical Path: $x_3 \to G_2 \to G_5 \to G_8 \to y_3$
t_1
t_2
G
t_3
$t_3 = \max(t_1, t_2) + d(G)$

$$t_{\text{tot}} = t_n = \max(t_1, ..., t_{n-1}) + d(G)$$

▼ Calculate Critical Path with Delay Table

▼ note how each gate changes its values as shown in the example



    - the gate values are determined considering if inputs are all low (usually 0) then if inputs are all high (usually 1)
    - e.g. gate A has low = 0 (bc with low inputs AND give 0) and has high = 1 (bc AND w high inputs gives 1) and o on
- use the delay table s.t. $L = \text{load factor} = \text{fan-out}$ in the propagation delays

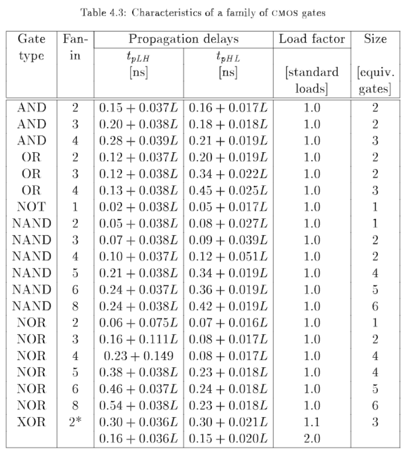


Table 4.3: Characteristics of a family of CMOS gates

Gate type	Fan-in	Propagation delays t_{pLH} [ns]	Propagation delays t_{pHL} [ns]	Load factor [standard loads]	Size [equiv. gates]
AND	2	$0.15 + 0.037L$	$0.16 + 0.017L$	1.0	2
AND	3	$0.20 + 0.038L$	$0.18 + 0.018L$	1.0	2
AND	4	$0.28 + 0.039L$	$0.21 + 0.019L$	1.0	3
OR	2	$0.12 + 0.037L$	$0.20 + 0.019L$	1.0	2
OR	3	$0.12 + 0.038L$	$0.34 + 0.022L$	1.0	2
OR	4	$0.13 + 0.038L$	$0.45 + 0.025L$	1.0	3
NOT	1	$0.02 + 0.038L$	$0.05 + 0.017L$	1.0	1
NAND	2	$0.05 + 0.038L$	$0.08 + 0.027L$	1.0	1
NAND	3	$0.07 + 0.038L$	$0.09 + 0.039L$	1.0	2
NAND	4	$0.10 + 0.037L$	$0.12 + 0.051L$	1.0	2
NAND	5	$0.21 + 0.038L$	$0.34 + 0.019L$	1.0	4
NAND	6	$0.24 + 0.037L$	$0.36 + 0.019L$	1.0	5
NAND	8	$0.24 + 0.038L$	$0.42 + 0.019L$	1.0	6
NOR	2	$0.06 + 0.075L$	$0.07 + 0.016L$	1.0	1
NOR	3	$0.16 + 0.111L$	$0.08 + 0.017L$	1.0	2
NOR	4	$0.23 + 0.149$	$0.08 + 0.017L$	1.0	4
NOR	5	$0.38 + 0.038L$	$0.23 + 0.018L$	1.0	4
NOR	6	$0.46 + 0.037L$	$0.24 + 0.018L$	1.0	5
NOR	8	$0.54 + 0.038L$	$0.23 + 0.018L$	1.0	6
XOR	2*	$0.30 + 0.036L$	$0.30 + 0.021L$	1.1	3
		$0.16 + 0.036L$	$0.15 + 0.020L$	2.0	

- delay prop. Low→High if gate has (0/1) and High→Low if (1/0) otherwise (0/0) and (1/1) have t=0 delay)
- fan-in in is the number of input variables
- fan-out is the number of output variables

- ▼ load factor is a reference to the energy/cost it takes to use the gate
- ▼ size is arbitrary definition based on power consumption and delay prop.
  - ▼ size is now obsolete as transistors can be manufacture in billions to trillions per die

▼ Analysis of Gate Networks

▼ Functional Analysis

- ▼ define I/O switching expressions (typically letters with subscript)
- ▼ create a net list (tabular representation) of the binary function
- ▼ define high-level I/O variables → use codes to represent as bit-vectors
- ▼ create a high-level spec of the system

▼ Network characteristic

- ▼ input load factors
- ▼ fan-out factors
- ▼ delays

▼ e.g. logic diagram with SEs

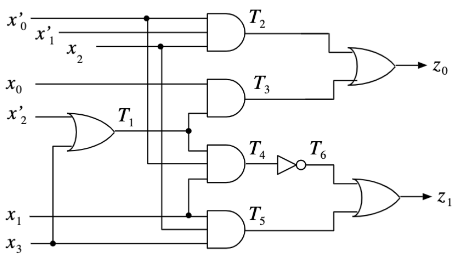


Figure 4.10: Gate network for analysis

- e.g. binary function

Output expressions:

$$\begin{aligned}
z_0 &= T_2 + T_3 \\
&= x_0'x_1'x_2 + x_0T_1 \\
&= x_0'x_1'x_2 + x_0(x_2' + x_3) \\
&= x_0'x_1'x_2 + x_0x_2' + x_0x_3
\end{aligned}$$

$$\begin{aligned}
z_1 &= T_5 + T_6 \\
&= x_1x_2x_3 + T_4' \\
&= x_1x_2x_3 + (T_1x_0'x_1)' \\
&= x_1x_2x_3 + T_1' + x_0 + x_1' \\
&= x_1x_2x_3 + x_2x_3' + x_0 + x_1'
\end{aligned}$$

Reduced expressions

$$\begin{aligned}
z_0 &= x_0'x_1'x_2 + x_0x_2' + x_0x_3 \qquad \text{(no reduction possible)} \\
z_1 &= x_0 + x_1' + x_2
\end{aligned}$$

Hierarchical approach

- decompose the network in subnetworks (modules)
- analyze subnetworks separately
- substitute into gate network function
- e.g. creating subnetworks

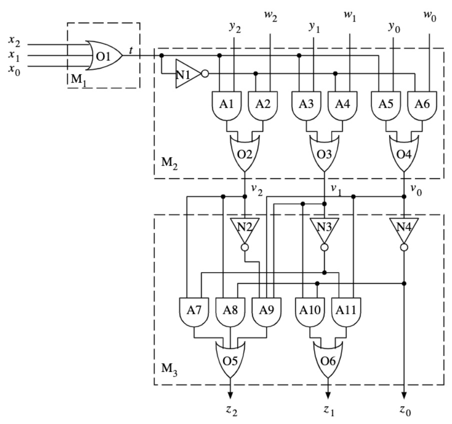


Figure 4.11: Network for hierarchical analysis

- verify subnetwork satisfies binary function
  - binary function (spec)

Inputs: $x, y, w \in \{0, 1, \ldots, 7\}$
Output: $z \in \{0, 1, \ldots, 7\}$

Function: $z = \begin{cases} (y+1) \bmod 8 & \textbf{if} \quad x \neq 0 \\ (w+1) \bmod 8 & \textbf{if} \quad x = 0 \end{cases}$

  - subnetwork functions

M_1:

$$t = x_2 + x_1 + x_0$$

$$t = \begin{cases} 1 & \textbf{if} \quad x \neq 0 \\ 0 & \textbf{otherwise} \end{cases}$$

M_2:

$$v_i = y_i t + w_i t' \quad (i = 0, 1, 2)$$

$$\underline{v} = \begin{cases} \underline{y} & \textbf{if} \quad t = 1 \\ \underline{w} & \textbf{if} \quad t = 0 \end{cases}$$

$$v = \begin{cases} y & \textbf{if} \quad t = 1 \\ w & \textbf{if} \quad t = 0 \end{cases}$$

M_3:

$$\begin{aligned} z_2 &= v_2' v_1 v_0 + v_2 v_1' + v_2 v_0' \\ z_1 &= v_1 v_0' + v_1' v_0 \\ z_0 &= v_0' \end{aligned}$$

- create high level spec (net list, table)

High-level specification:

v_2	v_1	v_0	z_2	z_1	z_0
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	1	0	1
1	0	1	1	1	0
1	1	0	1	1	1
1	1	1	0	0	0

$\rightarrow$

v	z
0	1
1	2
2	3
3	4
4	5
5	6
6	7
7	0

▼ derive network function

From table, we get

$$z = (v + 1) \bmod 8$$

📌 SUMMARY