Chapter 5
CMOS Logic Gate Design

Section 5.2

-To achieve correct operation of integrated logic gates, we need to satisfy
1. Functional specification
2. Temporal (timing) constraint.

(1) In CMOS, incorrect functions are caused by
1. Incorrect or insufficient power supplies or power supply noise.
2. Noise on gate input (so we need noise margin)
3. Faulty transistors
4. Faulty connections to transistors
5. Incorrect ratio in ratioed logic
6. Charge sharing or incorrect clocking dynamic gates

-In general, CMOS is tolerable to noise -> safeness
⇒ good choice for modern system - level Ic designs

(2) A fair portion of the design cycle may be spent in optimizing the speed of the design.

\[ tf(tr, td) \propto k \cdot \frac{C_{Load}}{\beta_{eff} \cdot V_{DD}} \]

where \( k \) is a constant (1.5~2)

\( \beta_{eff} \propto \) no. of transistors in parallel or serial

\( e.g., \) in 3-input NAND gate

\[ \beta_{eff,n} = \beta_n / 3, \beta_{eff,p} = \beta_p \]
$- C_{load}$:
Loading capacitance: the performance factor

(1) Size of transistors in the gate (self-loading)
(2) Size and number of transistors to which the gate is connected.
(3) The routing capacitance own the gate and other the gates it drives

Also, speed of gate will be affected by $tr/\tau f$ of input waveform.

* Check speed -> Find the critical paths of your design: We can use “timing analyzer” to find the total delays along the critical paths.
* Design Skills:
  (1) Algorithm level ((Power-of-two coefficients instead of real multiplier)
  (2) Architecture level (e.g., Carry look-ahead adder design \textit{v.s.} Ripple adders)
  (3) RTL/logic gates (check pipelining, fan-in, fan-out, \textit{etc}).
  (4) Circuit-level level approaches (Logic AND/OR gates \textit{v.s.} CMOS compound gates)
  (5) Good floorplanning (relationship between blocks and I/O pins)
  (6) Layout-level design skills (Full-custom designs and automatic Placement and Routing (P&R)).

* Most leveraged way is achieved by
  $\Rightarrow$ Design a good algorithm (\textit{e.g.}, Fast Discrete Cosine Transform in JPEG and MPEG)
  $\Rightarrow$ Novel architectural designs (minimize the critical paths)
  $\Rightarrow$ Register-transfer Level (RTL) designs: Describe your designs in terms of Gates, Basic operators ($\textit{adders, multipliers, division units, etc.}$), and Delay elements. For example, MIPS CPU architecture and FIR filter architectures.
5.2.1 Fan-in and Fan-out of Logic gates

(a) **Fan-in:** Number of inputs *e.g.*, 4-input NAND gate has a fan-in of 4; 2-input NAND gate has a fan-in of 2 (it is known in advance.)

(b) **Fan-out:** Total number of gate inputs that are driven by a gate output. *Default gate size=minimum sized inverter as unity.*

(c) Fan-in and Fan-out will be affected by “stage ratio” and “transistors in parallel or serial” (effective Beta values for NMOS and PMOS).
\[
tr = \frac{R_p}{n} \left( n \cdot m \cdot C_d + C_r + K \cdot C_g \right)
\]

*Tr* (rise time) for an \( m \)-input NAND gate:

1. \( R_p \): effective resistance of a p-device in a unit inverter (size: \( L/W = 1/n \))
2. \( n \): width multiplier of PMOS
3. \( k \): fan-out
4. \( m \): fan-in
5. \( C_g \): Gate capacitance of an unit inverter.
6. \( C_d \): Drain capacitance
7. \( C_r \): Routing capacitance.
\[
Tr = \frac{R_p}{n} \left( m \cdot n \cdot C_R \cdot C_g + q(k) \cdot C_g + k \cdot C_g \right)
\]

*Drain cap  Routing cap  Fan-out*

\[
\begin{cases}
R: Cd/C_g: \text{ ratio of the intrinsic drain capacitance of an inverter to the gate capacitance:} \\
q(k): \text{ Represent routing cap in terms of } C_g
\end{cases}
\]

where
Example: (SPICE simulation)

- $W_n = 6\mu, L_n = 1\mu$
- $W_p = 12.3\mu, L_p = 1\mu$
- $T_{\text{input-rise/fall}} = 0.1\text{ns}$
- $C_L = 0 \sim 1\text{pF}$

=> NAND gate is a better choice than NOR gate

=> Fan-in vs. delay
=> $C_L$ vs. Delay

Example of an 8-input NAND gate construction

- Approach 1: An 8-input NAND + an Inverter.
- Approach 2: Two 4-input NAND + 2-input NOR
- Approach 3: see Figure
Trade off between Area, speed (and power?)

*Transistor sizing: “stage ratio” to drive large $C_l$ (such as Clock Tree and Global Reset)*

*Guideline: Start with minimum sized devices then optimize paths from a critical-path-timing analysis. Optimizing paths can be done at different levels*

- Use NAND structures where possible
- Place (big) inverters at high fan-out nodes if possible
- Avoid the use of NOR structures in high-speed circuits (fan-in > 4 or fan-out is large)
- Use a fan-out below 5-10
- Use minimum-sized gate on high fan-out nodes to minimize $C_l$ presented to the driving gate.
- Keep rising and falling edges sharp

---

### TABLE 5.3 Comparison of Approaches to Designing an 8-input AND Gate

<table>
<thead>
<tr>
<th>APPROACH</th>
<th>DELAY STAGE 1 ns</th>
<th>DELAY STAGE 2 ns</th>
<th>DELAY STAGE 3 ns</th>
<th>DELAY STAGE 4 ns</th>
<th>TOTAL DELAY (SPICE) ns</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.82</td>
<td>3.37</td>
<td></td>
<td></td>
<td>6.2</td>
</tr>
<tr>
<td>ND8-&gt;INV</td>
<td>ND8 INV</td>
<td>falling rising</td>
<td></td>
<td></td>
<td>(6.5)</td>
</tr>
<tr>
<td>2</td>
<td>88</td>
<td>4.36</td>
<td></td>
<td></td>
<td>5.24</td>
</tr>
<tr>
<td>ND4-&gt;NR2</td>
<td>ND4 NR2</td>
<td>falling rising</td>
<td></td>
<td></td>
<td>(5.26)</td>
</tr>
<tr>
<td>3</td>
<td>31</td>
<td>.4</td>
<td>.31</td>
<td>2.17</td>
<td>3.19</td>
</tr>
<tr>
<td>ND2-&gt;NR2</td>
<td>ND2 NR2</td>
<td>falling rising</td>
<td>falling rising</td>
<td>INV</td>
<td>(3.46)</td>
</tr>
<tr>
<td>ND2-&gt;INV</td>
<td>falling</td>
<td>rising</td>
<td>rising</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

### TABLE 5.4 Areas for 8-input AND Gate Implementations

<table>
<thead>
<tr>
<th>APPROACH</th>
<th>AREA</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>216</td>
</tr>
<tr>
<td>2</td>
<td>216</td>
</tr>
<tr>
<td>3</td>
<td>150</td>
</tr>
</tbody>
</table>

(Note: In a real row decoder, some of the gates may be shared between different row decoders.)
5.4 CMOS Logic structures

5.4.1 CMOS complementary logic

Two function determining blocks: 
* N-block and P-block

* 2n transistors for an n-input logic gate.

5.4.3 Pseudo-NMOS logic

*need (n+1) transistors
5.4.4 Dynamic CMOS Logic

\[
Z = \begin{cases} 
0, & \text{n-block short} \\
1, & \text{n-block open} \\
clk=0, Z=1 & (C_L \text{ is charged to vdd}) \\
clk=1, & \text{Z is conditionally evaluated}
\end{cases}
\]

- CLK is a single-phase clock
- Pull-up time is improved.
- Pull-down time is increased due to the ground switch.

Problems:

(a) Inputs can only change during the precharge phase and must be stable during the precharge phase => charge sharing may corrupt the output node voltage.

(b) Simple single-phase dynamic CMOS gates cannot be cascaded (some delay between \(N1&N2\))
5.4.5 Clocked CMOS Logic

5.4.6 Pass-Transistor Logic

\[ F = \sum_i P_i V_i \]

\[ P_i = \text{Control signals} \]

\[ V_i = \text{Pass signals} \]

\[ V_i \in \{0, 1, X, \neg X, Z\} \]
Example 1:
Use pass-transistor logic to design a 2-input XNOR gate

(a) Truth table

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>A @ B</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

(b) Pass-network Karnaugh map

Use: A as the control signals
B as the passed signals

(c) Logic function

\[
F = \neg A \cdot (\neg B) + A \cdot B
\]

(d) Implementation

(a) Complementary
(b) NMOS
(c) Cross-coupled

Example 2: Use pass-transistor logic to construct Boolean function

(a) Truth Table

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>P₁</th>
<th>P₂</th>
<th>P₃</th>
<th>P₄</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>P₄</td>
<td>P₃</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>P₂</td>
<td>P₁</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(b) Some Functions Implemented by the Boolean Function Unit

<table>
<thead>
<tr>
<th>OPERATION</th>
<th>P₁</th>
<th>P₂</th>
<th>P₃</th>
<th>P₄</th>
</tr>
</thead>
<tbody>
<tr>
<td>AND (A, B)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>XOR (A, B)</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>OR (A, B)</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>NOR (A, B)</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>NAND (A, B)</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
The apparent advantages of pass-transistor networks in CMOS should be studied carefully (e.g., how to achieve good logic levels?)

5.4.7 CMOS Domino Logic

(1) During precharge (clk=0), PZ=1, and inverter o/p=0

(2) Transistors in subsequent logic blocks will be turned off during the precharge phase

(3) Each gate in sequence can make at most one transition (1 to 0) => Can be used in cascaded logic gates
Limitations:

1. Each gate must be buffered (an advantage, too)
2. Only non-inverting structures are possible
3. Common in dynamic CMOS—“charge sharing”

Charging sharing problem in Dynamic CMOS

\[ C_2 - C_7 = \text{low} \]
\[ A_0 = \text{low} \]
\[ A_{1-5} = \text{high} \]
when clk = 1
Q in C_1 is dumped into C_2 - C_7
Q in C_1

\[ V_{nl} = \frac{C_1 \cdot V_{DD}}{\left( \sum_{i=2}^{7} C_i \right) + C_1} \]

IF \( C_1 = 3 \times C_2 \) & \( C_2 = C_3 = C_4 = C_5 = C_6 = C_7 \)

then \( V_{nl} = \frac{3C_2}{6C_2 + 3C_2} \cdot V_{DD} = 0.33V_{DD} = 1.65V \Rightarrow \) turn on the inverter

Solution:

(a) Place clocked NMOS at the bottom
(b) Provide immediate nodes with precharging transistors

13
5.4.8 NP domino logic (Zipper CMOS)

Will turn off next stages during pre-charging

Advantage of D. CMOS
(1) Smaller area
(2) $C \downarrow$, $speed \uparrow$
(3) Glitch free if design carefully

5.5 Clocking strategies

(a) FSM (Finite State Machine)
(b) Pipelined system

FIGURE 5.43 Clocked systems: (a) a simple finite state machine (FSM); (b) a pipelined system

FIGURE 5.44 A single-phase clock showing parameters of interest
(a) Setup time: the time before the clock edge that the D input has to be stable
(b) Hold time: the time after the clock edge that the D input has to maintain stable
(c) Clock-to-Q delay ($T_q$): the delay from the positive clock input to the new value of the Q output.

Latches

(a) Negative level-sensitive latch
(b) Positive level-sensitive latch
(c) Positive edge-triggered register (master-slave)
(d) Operation of the master-slave register
(e) CMOS circuit implementation.