EE 414 – Introduction to VLSI Design

Module #6 – Combinational Logic

- Agenda
  1. Combinational Logic
     - n-Input Gates & Equivalent Inverter
     - CMOS Logic Synthesis
     - Transmission Gates
     - Layout of Complex Logic

- Announcements
  1. Read Chapter 7

CMOS Combinational Logic

- CMOS Combinational Logic:
  - combinational logic refers to circuits with 2 or more inputs and 1 output
  - the output depends on the combination of the input values
  - we describe the logic operation of a circuit using truth tables
  - from this, we can find a minimal Sum of Products expression using K-maps
  - once we have a logic expression, we then move into the transistor level implementation stage

CMOS Combinational Logic

- CMOS Basic Gates
  - in CMOS, we always have:
    - a pull-up network using PMOS transistors
    - a pull-down network using NMOS transistors
  - the pull-up and pull-down networks are configured in a complementary topology
  - the complementary topology guarantees that when driving a ‘1’, the pull-up network is ON and the pull-down network is OFF
  - it also guarantees that when driving a ‘0’, the pull-up network is OFF and the pull-down network is ON
  - circuit topology gives us the ability to design logic functionality where transistors:
    - in series = an AND’ing function
    - in parallel = an OR’ing function

CMOS Combinational Logic

- CMOS 2-Input NOR Gate
  - the truth table for a 2-input NOR gate is:

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Y_{out}</th>
</tr>
</thead>
<tbody>
<tr>
<td>low</td>
<td>low</td>
<td>high</td>
</tr>
<tr>
<td>low</td>
<td>high</td>
<td>low</td>
</tr>
<tr>
<td>high</td>
<td>low</td>
<td>high</td>
</tr>
<tr>
<td>high</td>
<td>high</td>
<td>low</td>
</tr>
</tbody>
</table>

CMOS Combinational Logic

- CMOS 2-Input NOR Gate
  - the transistor level implementation for the NOR gate is:
CMOS Combinational Logic

- CMOS 2-input NOR Gate
  - we can model a 2-input NOR gate as an equivalent inverter as follows:
  
  \[
  \begin{align*}
  V_{\text{out}} &= V_{DD} - V_{\text{in}} - V_{DD} \\
  &= -V_{DD} - V_{\text{in}}
  \end{align*}
  \]

  - let's use representative voltages of \(V_{DD}=5\) and \(V_{\text{in}}=2.5\) to illustrate the derivation

CMOS Combinational Logic

- CMOS 2-input NOR Gate
  - we can model a 2-input NOR gate as an equivalent inverter as follows:
  
  \[
  \begin{align*}
  V_{\text{out}} &= V_{DD} - V_{\text{in}} - V_{DD} \\
  &= -V_{DD} - V_{\text{in}}
  \end{align*}
  \]

  - let's use representative voltages of \(V_{DD}=5\) and \(V_{\text{in}}=2.5\) to illustrate the derivation

CMOS Combinational Logic

- CMOS 2-input NOR Gate
  - we can model a 2-input NOR gate as an equivalent inverter as follows:
  
  \[
  \begin{align*}
  V_{\text{out}} &= V_{DD} - V_{\text{in}} - V_{DD} \\
  &= -V_{DD} - V_{\text{in}}
  \end{align*}
  \]

  - let's use representative voltages of \(V_{DD}=5\) and \(V_{\text{in}}=2.5\) to illustrate the derivation

CMOS Combinational Logic

- CMOS 2-input NOR Gate
  - we can model a 2-input NOR gate as an equivalent inverter as follows:
  
  \[
  \begin{align*}
  V_{\text{out}} &= V_{DD} - V_{\text{in}} - V_{DD} \\
  &= -V_{DD} - V_{\text{in}}
  \end{align*}
  \]

  - let's use representative voltages of \(V_{DD}=5\) and \(V_{\text{in}}=2.5\) to illustrate the derivation

CMOS Combinational Logic

- CMOS 2-input NOR Gate
  - we can model a 2-input NOR gate as an equivalent inverter as follows:
  
  \[
  \begin{align*}
  V_{\text{out}} &= V_{DD} - V_{\text{in}} - V_{DD} \\
  &= -V_{DD} - V_{\text{in}}
  \end{align*}
  \]

  - let's use representative voltages of \(V_{DD}=5\) and \(V_{\text{in}}=2.5\) to illustrate the derivation

CMOS Combinational Logic

- CMOS 2-input NOR Gate
  - we can model a 2-input NOR gate as an equivalent inverter as follows:
  
  \[
  \begin{align*}
  V_{\text{out}} &= V_{DD} - V_{\text{in}} - V_{DD} \\
  &= -V_{DD} - V_{\text{in}}
  \end{align*}
  \]

  - let's use representative voltages of \(V_{DD}=5\) and \(V_{\text{in}}=2.5\) to illustrate the derivation

CMOS Combinational Logic

- CMOS 2-input NOR Gate
  - we can model a 2-input NOR gate as an equivalent inverter as follows:
  
  \[
  \begin{align*}
  V_{\text{out}} &= V_{DD} - V_{\text{in}} - V_{DD} \\
  &= -V_{DD} - V_{\text{in}}
  \end{align*}
  \]

  - let's use representative voltages of \(V_{DD}=5\) and \(V_{\text{in}}=2.5\) to illustrate the derivation

CMOS Combinational Logic

- CMOS 2-input NOR Gate
  - we can model a 2-input NOR gate as an equivalent inverter as follows:
  
  \[
  \begin{align*}
  V_{\text{out}} &= V_{DD} - V_{\text{in}} - V_{DD} \\
  &= -V_{DD} - V_{\text{in}}
  \end{align*}
  \]

  - let's use representative voltages of \(V_{DD}=5\) and \(V_{\text{in}}=2.5\) to illustrate the derivation
CMOS Combinational Logic

- CMOS 2-Input NOR Gate

Now we can relate the drain currents knowing that $I_{D3} = I_{D4} = I_{D,n}$ giving a 2nd equation relating $V_{th}$ to $I_D$:

$$V_{th} = \frac{1}{2} \frac{V_{DD}}{I_{D,n}}$$

Combining this with our previous expression we get:

$$V_{NOR} = \frac{1}{2} \frac{V_{DD}}{I_{D,n}} (|V_A| + |V_B|)$$

or...

- CMOS N-Input NOR Gate

To expand the NOR gate to N-inputs, we can use:

$$V_{NOR} = \frac{1}{N} \frac{V_{DD}}{I_{D,n}} (\sum |V_A|)$$

or our rule of thumb for an ideal symmetric equivalent inverter becomes:

$$k_P = k_n = \frac{1}{\sqrt{N}}$$

- An equivalent network for series-connected transistors with the same sizes:

$$\sum \frac{1}{LW} = \sum \frac{1}{LW}$$

- An equivalent network for parallel-connected transistors with the same sizes:

$$\sum \frac{1}{L} = \sum \frac{1}{W}$$

CMOS 2-Input NOR Gate

- In an equivalent model, to get $V_{th} = V_{DD}/2$, we can use:

$$I_{D,n} = \frac{1}{2} \frac{V_{DD}}{k_n}$$

- Note that the PMOS series network has to be sized larger in order to overcome the voltage drop across each stage.
**CMOS Combinational Logic**

- **CMOS 2-Input NAND Gate**
  - The truth table for a 2-input NAND gate is:

<table>
<thead>
<tr>
<th>( V_A )</th>
<th>( V_B )</th>
<th>( V_{out} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>low</td>
<td>low</td>
<td>high</td>
</tr>
<tr>
<td>low</td>
<td>high</td>
<td>low</td>
</tr>
<tr>
<td>high</td>
<td>low</td>
<td>high</td>
</tr>
<tr>
<td>high</td>
<td>high</td>
<td>low</td>
</tr>
</tbody>
</table>

- **PMOS Pull-Up Network**
  - The pull-up network is continually driving the output unless \( V_A \) AND \( V_B \) are 1.
  - Since the pull-up network uses PMOS transistors (0=ON), we can say that the pull-up network is conducting if \( V_A \) OR \( V_B \) are 0.
  - This implies a parallel configuration in the pull-up (PMOS) network.

- **NMOS Pull-Down Network**
  - The only time the pull-down network drives the output is when we have two 1’s on the inputs.
  - This means the pull-down network is conducting only when \( V_A \) AND \( V_B \) are 1.
  - This implies a series configuration in the pull-down (NMOS) network.

- **Transistor Level Implementation** for the NAND gate is:

- **Switching Threshold** for the 2-input NAND gate is:

\[
V_{th(\text{NAND})} = \frac{V_{DD} \cdot R_{pp}}{R_{nn}} \cdot \left( 1 + \frac{1}{V_{DD}} \right)
\]

- In an equivalent inverter model, to get \( V_{th} = V_{DD}/2 \), we can use:

\[
4 \cdot R_{pp} = R_{nn}
\]

- Note that the NMOS series network has to be sized larger in order to overcome the voltage drop across each series stage.

**EELE 414 – Introduction to VLSI Design**

Module #6

Page 19

Page 20

Page 21

Page 22

Page 23

Page 24
CMOS Combinational Logic

- **CMOS N-Input NAND Gate**
  - To expand the NAND gate, we:
    - add more PMOS transistors in parallel in the pull-up network
    - add more NMOS transistors in series in the pull-down network

- **CMOS Combinational Logic**

  - **CMOS 2-Input NAND Gate Layout**

  - **Complex CMOS Logic Circuits**
    - We can implement any logic functions using NANDs, NORs, and INV's.
    - However, the timing and area of the standard SOP approach can be improved further by creating the entire logic function in one circuit.
    - We can design a complex function by:
      1. Creating the NMOS pull-down network of \( \overline{F} \) where:
         - An OR is performed using parallel connected NMOS's
         - An AND is performed using series connected NMOS's
      2. Creating the PMOS pull-up network to be the complement of the pull-down network

  - **Complex CMOS Logic Graphs**
    - The complementary PMOS network is created with a technique called a dual pull-up/down graph.
    - This is a graphical way to create the PMOS network for a given NMOS pull-down circuit.
    - We first create the pull-down graph by representing:
      1. Each NMOS transistor as an edge (i.e., a line), and
      2. Each node as a vertex (i.e., a dot)
    - We orient the pull-down graph in the same orientation as the NMOS circuit
      - \( \overline{V_{DD}} \) on top, \( V_{SS} \) on bottom.
CMOS Combinational Logic

- Complex CMOS Logic Graphs
  - We create the dual pull-up graph on top of the pull-down graph.
  - We orient the pull-up graph with $V_{DD}$ on the left and $V_{out}$ on the right.
  - We create the pull-up graph using the rules:
    1) A new vertex is created within each confined area of the pull-down graph.
    2) Each vertex is connected by an edge which crosses each edge of the pull-down graph.

- AOI / OAI CMOS Logic
  - We classify the common types of logic expression forms as:
    AOI - AND-OR-INVERT
    - This corresponds to a Sum-of-Products logic expression form:
      $$f = x \cdot (y' + z')$$
    OAI - OR-AND-INVERT
    - This corresponds to a Product-of-Sums logic expression form:
      $$f = (x + y) \cdot (z' + w')$$
  - Note that we have the Invert portion in these forms so that we can directly synthesize the NMOS pull-down network.
  - We can create this form of a logic expression using De Morgan’s Theorem.

- Complex CMOS Logic Graphs (XOR)
  - Let's design a 2-input XOR gate.
  1) We first find the traditional SOP expression
  2) We then manipulate it into an OAI or AOI form so that the pull-down network can be directly synthesized.

- Complex CMOS Logic Graphs (XOR)
  - We directly synthesize the pull-down network and create its equivalent pull-down graph.
  - We then create the dual pull-up graph on top of the pull-down graph.
  - Insert a vertex in each enclosed region of the pull-down graph (vertex = node).
  - Connect each vertex such that each pull-down is crossed by one pull-up edge (edge = PMOS).
Transmission Gate (Pass Gate)

A Transmission Gate (T-gate or TG or pass gate) is a bi-directional switch made up of an NMOS and PMOS in parallel.

- A control signal is connected to the gate of the NMOS (C) and its complement is sent to the gate of the PMOS (C‘).
- The T-gate is a bidirectional switch between A and B which is controlled by C.

When the control signal is HIGH (VCC):
- both transistors are turned on
- a low resistance path exists between A and B

When the control signal is LOW (0v):
- both transistors are off
- the T-gate looks like an open circuit

This type of operation is commonly used in bus situations where only one gate can drive the bus line at the same time.

T-gates are put on the output of each gate on the bus. The circuit that drives will use a T-gate to connect to the bus with a low impedance path. All other circuits that aren’t driving will switch their T-gates to a high impedance.

CMOS Combinational Logic

Complex CMOS Logic Graphs (XOR)

1) we then unfold the dual pull-up/down graph and directly synthesize the pull-up network

CMOS Combinational Logic

Complex CMOS Logic Equivalent Inverters

- To calculate the performance of a complex Logic Circuit, we need create an equivalent inverter

- we then use our equations for VDD, Vss, and Vout on the equivalent transconductance values

- we know that:

- an equivalent network for series-connected transistors with the same sizes:

- an equivalent network for parallel-connected transistors with the same sizes:

When the control signal is HIGH (VCC):

- diffusion regions as rectangles
- polysilicon inputs are columns
- metal traces are lines

Complex CMOS Logic Graphs (XOR)

To calculate the performance of a complex Logic Circuit, we need create an equivalent network for parallel

- the T-gate looks like an open circuit

- we don’t put any size information in this diagram

- diffusion regions as rectangles
- polysilicon inputs are columns
- metal traces are lines

The intent of an atlas diagram is to figure out how the inputs and outputs are connected to the diffusion regions.

- the intent of a atlas diagram is to figure out how the diffusion layer contacts are placed.

When the control signal is LOW (0v):

- contacts are circles

- we don’t put any size information in this diagram

- diffusion regions as rectangles
- polysilicon inputs are columns
- metal traces are lines

Planning ahead can save a lot of time in the tool.

When the control signal is LOW (0v):

- drawing the general layout configuration that will be used in order to figure out how the diffusion layer contacts are placed.

- a stick diagram:

- diffusion regions as rectangles
- polysilicon inputs are columns
- metal traces are lines

- contacts are circles

- we don’t put any size information in this diagram

- we typically sketch this out with pencil and paper prior to going into the tool

- planning ahead can save a lot of time in the tool.
CMOS Combinational Logic

- Transmission Gate (Pass Gate)
  - When the T-gate is on, the regions of operation of the transistors will depend on \( V_{in} \) and \( V_{out} \).
  - Let’s say we drive \( V_{in} = V_{DD} \) and initially \( V_{out} = 0 \).
  - As \( V_{out} \) moves from 0 to \( V_{DD} \), the regions of operation for the transistors are as follows:

\[
R_{eq} = \frac{V_{DD} - V_{in}}{I_{in}} \\
R_{on} = \frac{V_{out} - V_{in}}{I_{in}} \\
R_{off} = R_{eq} \cdot R_{on}
\]

CMOS Combinational Logic

- Transmission Gate (Pass Gate)
  - We typically model the T-gate as a switch with an equivalent resistance.
  - At any point, the equivalent resistance is given by:

\[
R_{eq} = \frac{V_{DD} - V_{in}}{I_{in}} \\
R_{on} = \frac{V_{out} - V_{in}}{I_{in}} \\
R_{off} = R_{eq} \cdot R_{on}
\]

CMOS Combinational Logic

- Transmission Gate (Pass Gate)
  - Transmission gates can be used to create combinational logic, allowing for complex logic using T-gates, which has an advantage when it comes to layout because of the symmetry of the PMOS and NMOS. One N-well can be used for all PMOS's.