EE 414 – Introduction to VLSI Design

Module #6 – Combinational Logic

- **Agenda**

  1. Combinational Logic

     - n-Input Gates & Equivalent Inverter
     - AOI/OAI Logic Synthesis
     - Transmission Gates
     - Layout of Complex Logic

- **Announcements**

  1. Read Chapter 7

CMOS Combinational Logic

- **CMOS Combinational Logic**

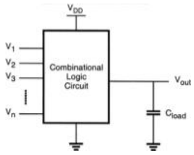
  - combinational logic refers to circuits with 2 or more inputs and 1 output

  - 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

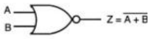
  - circuit topology gives us the ability to design logic functionality where transistors:

    - 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:



V_A	V_B	V_{out}
low	low	high
low	high	low
high	low	low
high	high	low

CMOS Combinational Logic

- **CMOS 2-Input NOR Gate**

  PMOS Pull-Up Network

  - The only time the pull-up network drives the output is when we have two 0’s on the inputs.

  - Since the pull-up network uses PMOS transistors (0=ON), we can say that the pull-up network is conducting if V_A **AND** V_B are 0.

  - This implies a **series** configuration in the pull-up (PMOS) network.

  NMOS Pull-Down Network

  - The pull-down network is continually driving the output unless V_A AND V_B are 0.

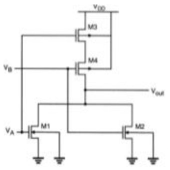
  - Since the pull-down network uses NMOS transistors (1=ON), we can say that the pull-down network is conducting if V_A **OR** V_B are 1.

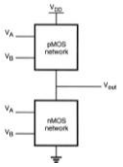
  - This implies a **parallel** configuration in the pull-down (NMOS) network.

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 have already derived expressions for how to find the critical voltages for an inverter

- of specific interest is V_{th}, which we can adjust with the sizing of the transistors

- in order to analyze a more complex logic gate, we convert it into an *equivalent inverter*

Transistors in Series

- conceptually, the current flowing in series transistors needs to go through *two* channels, each with an equivalent resistance (or transconductance *k*)

- with the effective resistance doubling, we can say that the transconductance (or the ability to drive a current given an input voltage) is divided by 2

- transistors in series with the same size can be modeled as an equivalent transistor with $\mathbf{k_{eq}=k/2}$

Transistors in Parallel

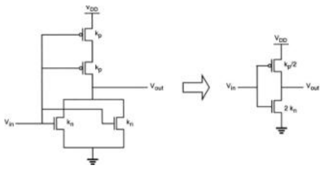
- conceptually, the current flowing in parallel transistors can conduct twice the amount of current compared to a a single transistor with the same gate voltage.

- we can model this behavior with an equivalent transistor with $\mathbf{k_{eq}=2\cdot k}$

CMOS Combinational Logic

- **CMOS 2-Input NOR Gate**

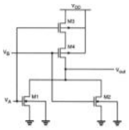
- we can model a 2-Input NOR gate as an equivalent inverter as follows:



- let's use representative voltages of V_{DD}=5v and V_{th}=2.5 to illustrate the derivation

CMOS Combinational Logic

- **CMOS 2-Input NOR Gate**



- we can derive the switching threshold by stating that:

$$V_{in} = V_{out} = V_A = V_B = V_{th}$$

- we can begin by writing the KCL equation at the V_{out} node:

$$\left|I_{D,n-network}\right| = \left|I_{D,p-network}\right|$$

- for the NMOS, since $V_{GS,n}$=$V_{DS,n}$, we know what both transistors are in *saturation*

- in addition, we can state that the current at the V_{out} node is the combination of both NMOS currents

$$I_{D,n-network} = 2 \cdot I_{D,n|sat}$$
$$I_{D,n-network} = 2 \cdot \left[\frac{1}{2} \cdot k_n \cdot \left(V_{th} - V_{T,n}\right)^2\right] = k_n \cdot \left(V_{th} - V_{T,n}\right)^2$$

- rearranging this equation to get in terms of V_{th}, we have:

$$V_{th} = V_{T,n} + \sqrt{\frac{I_{D,n-network}}{k_n}}$$

CMOS Combinational Logic

- **CMOS 2-Input NOR Gate**

- now we look at the PMOS network, which has a +2.5v drop across it.

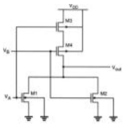
- since both transistors are ON, we can estimate that ~1.25v drops across each transistor:

$$V_{DS,M3} = V_{DS,M4} = -1.25v$$

- looking at M3, we know that $V_{GS,p|M3}$= -2.5v and $V_{DS,p|M3}$= -1.25v so M3 is in the **linear region.**

- looking at M4, the node between M3 and M4 is estimated to be at ~3.75 (i.e., 5v – 1.25v)

- this puts $V_{DS,p|M4}$= -1.25v and $V_{GS,p|M4}$= -1.25v, which means M4 is in the ***saturation region.***



CMOS Combinational Logic

- **CMOS 2-Input NOR Gate**

- since we know the regions of operation for M3 and M4, we can write:

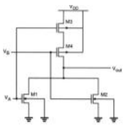
$$I_{DS,3|lin} = \frac{1}{2} \cdot k_p \cdot \left[2 \cdot \left(V_{GS,p} - V_{T,p}\right) \cdot V_{DS,p} - V_{DS,p}^2\right]$$

- Since the PMOS current is expressed terms of I_{DS}, we can rewrite this as:

$$I_{DS,n} = I_{SD,p} = -I_{DS,p}$$

- we know that for M3, $V_{GS,p}$ = V_{th}-V_{DD}. substituting this in and carrying the (-) through, in we get:

$$I_{SD,3|lin} = -I_{DS,3|lin} = -\frac{1}{2} \cdot k_p \cdot \left[2 \cdot \left(V_{th} - V_{DD} - V_{T,p}\right) \cdot V_{DS,p} - V_{DS,p}^2\right]$$
$$I_{SD,3|lin} = \frac{1}{2} \cdot k_p \cdot \left[2 \cdot \left(V_{DD} - V_{th} - \left|V_{T,p}\right|\right) \cdot V_{SD,p} - V_{SD,p}^2\right]$$



CMOS Combinational Logic

- **CMOS 2-Input NOR Gate**

- M4 is in the *saturation region* so we can write the current as:

$$I_{DS,4|sat} = \frac{1}{2} \cdot k_p \cdot \left[\left(V_{GS,p} - V_{T,p}\right)\ \right]$$

- Again, the PMOS current can be rewritten as:

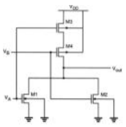
$$I_{DS,n} = I_{SD,p} = -I_{DS,p}$$

- we know that for M4:

$$V_{GS,p|M4} = V_{th} - (V_{DD} - V_{SD3})$$
$$V_{DS,p|M4} = V_{th} - (V_{DD} - V_{SD3})$$

- which gives the M4 current as:

$$I_{SD,4|sat} = -I_{DS,4|sat} = -\frac{1}{2} \cdot k_p \cdot \left(V_{th} - V_{DD} + V_{SD3} - V_{T,p}\right)^2$$
$$I_{SD,4|sat} = \frac{1}{2} \cdot k_p \cdot \left(V_{DD} - V_{th} - \left|V_{T,p}\right| - V_{SD3}\right)^2$$



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 a 2nd equation relating V_{th} to $I_{D:}$

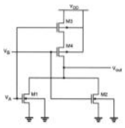
$$V_{DD}-V_{th}-\left|V_{T,p}\right|=2\sqrt{\frac{I_D}{k_p}}$$

- combining this with our previous expression we get:

$$V_{th}(NOR2)=\frac{V_{T,n}+\frac{1}{2}\sqrt{\frac{1}{k_R}}\left(V_{DD}-|V_{T,p}|\right)}{1+\frac{1}{2}\sqrt{\frac{1}{k_R}}}$$

- or…

$$V_{th}(NOR2)=\frac{V_{T,n}+\sqrt{\frac{1}{4\cdot k_R}}\left(V_{DD}-|V_{T,p}|\right)}{1+\sqrt{\frac{1}{4\cdot k_R}}}$$



CMOS Combinational Logic

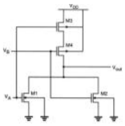
- **CMOS 2-Input NOR Gate**

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

$$k_R=\frac{k_n}{k_p}=1=\frac{2\cdot k_n}{k_p/2}$$

$$k_p=4\cdot k_n$$

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

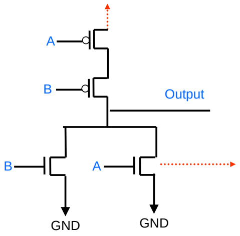


CMOS Combinational Logic

- **CMOS N-Input NOR Gate**

- to expand the NOR gate to N-inputs,

  - add more PMOS transistors in series in the Pull-up Network
  - add more NMOS transistors in parallel in the Pull-down Network



CMOS Combinational Logic

- **CMOS N-Input NOR Gate**

- our V_{th} expression for an N-input NOR gate becomes:

$$V_{th}(NOR)=\frac{V_{T,n}+\frac{1}{N}\sqrt{\frac{1}{k_R}}\left(V_{DD}-V_{T,p}\right)}{1+\frac{1}{N}\sqrt{\frac{1}{k_R}}}$$

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

$$k_R=\frac{k_n}{k_p}=1=\frac{N\cdot k_n}{k_p/N}$$

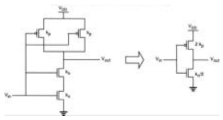
$$k_p=N^2\cdot k_n$$

CMOS Combinational Logic

- **CMOS N-Input NOR Gate**

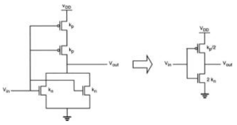
- an equivalent network for ***series-connected*** transistors with the same sizes:

$$\left(\frac{W}{L}\right)_{equivalent}=\frac{1}{\sum\frac{1}{\left(\frac{W}{L}\right)_N}}$$



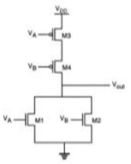
- an equivalent network for ***parallel-connected*** transistors with the same sizes:

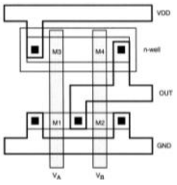
$$\left(\frac{W}{L}\right)_{equivalent}=\sum\left(\frac{W}{L}\right)_N$$



CMOS Combinational Logic

- **CMOS 2-Input NOR Gate Layout**

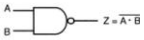




CMOS Combinational Logic

- **CMOS 2-Input NAND Gate**

  - the truth table for a 2-input NAND gate is:



V_A	V_B	V_{out}
low	low	high
low	high	high
high	low	high
high	high	low

CMOS Combinational Logic

- **CMOS 2-Input NAND Gate**

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

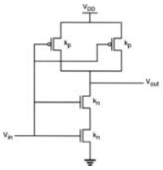
- This means the pull-down network is conducting only when V_A **AND** V_B are 1.

- This implies a **series** configuration in the pull-down (NMOS) network.

CMOS Combinational Logic

- **CMOS 2-Input NAND Gate**

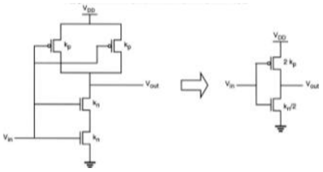
  - the transistor level implementation for the NAND gate is:



CMOS Combinational Logic

- **CMOS 2-Input NAND Gate**

  - we can model the 2-Input NAND gate as an equivalent invert as follows:

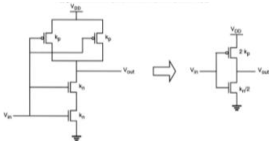


CMOS Combinational Logic

- **CMOS 2-Input NAND Gate**

  - the switching threshold for the 2-input NAND gate is:

$$V_{th}(NAND) = \frac{V_{T,n} + 2\sqrt{\frac{1}{k_R}}\left(V_{DD} - |V_{T,p}|\right)}{1 + 2\sqrt{\frac{1}{k_R}}}$$



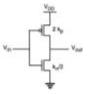
CMOS Combinational Logic

- **CMOS 2-Input NAND Gate**

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

$$k_R = \frac{k_n}{k_p} = 1 = \frac{k_n/2}{2 \cdot k_p}$$

$$4 \cdot k_p = k_n$$



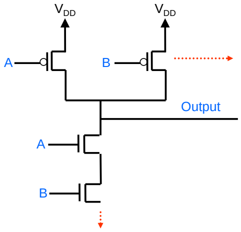
- note that the NMOS series network has to be sized larger in order to overcome the voltage drop across each series stage.

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 N-Input NAND Gate**

  - our V_{th} expression for an N-input NAND gate becomes:

$$V_{th}(NAND) = \frac{V_{T,n} + N\sqrt{\frac{1}{k_R}}\left(V_{DD} - |V_{T,p}|\right)}{1 + N\sqrt{\frac{1}{k_R}}}$$

$$V_{th}(NAND) = \frac{V_{T,n} + \sqrt{\frac{N^2}{k_R}}\left(V_{DD} - |V_{T,p}|\right)}{1 + \sqrt{\frac{N^2}{k_R}}}$$

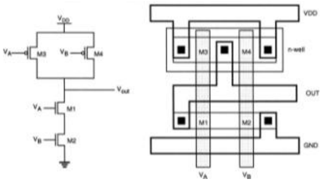
  - and our rule of thumb for an ideal symmetric equivalent inverter becomes:

$$k_R = \frac{k_n}{k_p} = 1 = \frac{k_n/N}{N \cdot k_p}$$

$$N^2 \cdot k_p = k_n$$

CMOS Combinational Logic

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



CMOS Combinational Logic

- **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 F' where:

      - an OR is performed using parallel connected NMOS's
      - an AND is performed using series connected NMOS's

      NOTE: this is an NMOS pull-down network so the logic function must be in an F' form (or a *dual* of F).

      NOTE: these rules can be used in a *nested* configuration to form any logic function

    2) Creating the PMOS pull-up network to be the *complement* of the pull-down network

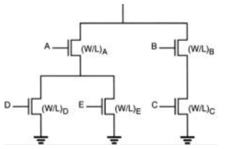
CMOS Combinational Logic

- **Complex CMOS Logic Circuits**

  - an example of the NMOS pull-down network synthesis is:

$$F = \overline{A \cdot (D+E) + B \cdot C}$$

    - D+E is created with two NMOS's in parallel
    - A(D+E) puts an NMOS in series with the (D+E) network
    - the entire A(D+E) network is in parallel with the BC network
    - the BC network is created with two series NMOS's



CMOS Combinational Logic

- **Complex CMOS Logic Graphs**

  - the *complementary* PMOS network is created with a technique called a ***dual pull-up/down graph***

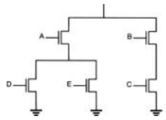
  - this is a graphical way to create the PMOS network for a given NMOS pull-down circuit.

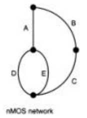
  - we first create the ***pull-down** graph* by representing:

    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 (V_{out} on top, V_{SS} on bottom)





CMOS Combinational Logic

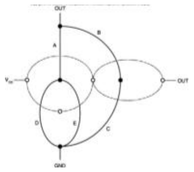
- **Complex CMOS Logic Graphs**

- next, 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*

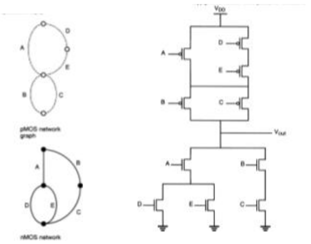


CMOS Combinational Logic

- **Complex CMOS Logic Graphs**

- now we can separate the graphs and synthesize the PMOS ***pull-up*** network

Remember that: an **edge** (a line) = a transistor
a **vertex** (a dot) = a node



CMOS Combinational Logic

- **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:

ex) $F = \overline{A\cdot B + B'\cdot C + C'D}$

OAI - OR-AND-INVERT

- this corresponds to a Products-of-Sums logic expression form:

ex) $F = \overline{(A+B)\cdot(B'+C)\cdot(C'+D)}$

- Note that we have the *Invert* portion in these forms so that we can directly synthesize the NMOS pull-down network.

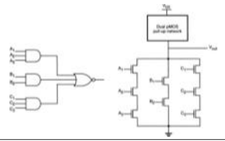
- we can create this form of a logic expression using *DeMorgan's Theorem*

CMOS Combinational Logic

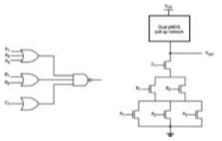
- **AOI / OAI CMOS Logic**

- AOI and OAI logic expressions can create standardized layouts due to their symmetry

AOI



OAI



CMOS Combinational Logic

- **Complex CMOS Logic Graphs (XOR)**

- let's design a 2-input XOR gate.

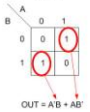
1) we first find the traditional SOP expression

2) we then manipulate it into an OAI or AOI form so that the pull-down network can be directly synthesized

Truth Table

AB	OUT
00	0
01	1
10	1
11	0

K-Map



Logic Manipulation

$$Out = A'B + AB'$$
$$Out = \overline{\overline{A'B + AB'}}$$
$$Out = \overline{\overline{(A'B)}\cdot\overline{(AB')}}$$
$$Out = \overline{(\overline{A'}+\overline{B})\cdot(\overline{A}+\overline{B'})}$$
$$Out = \overline{(A+B')\cdot(A'+B)}$$

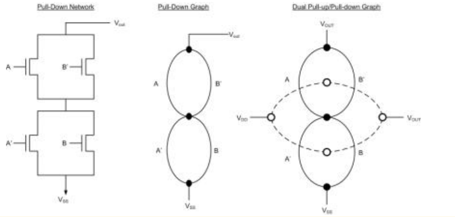
CMOS Combinational Logic

- **Complex CMOS Logic Graphs (XOR)**

3) We directly synthesize the pull-down network and create its equivalent *pull-down graph*

4) We then create the *dual pull-up graph* on top of the *pull-down* graph

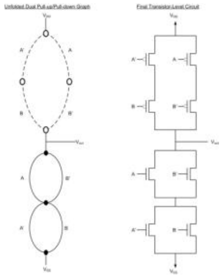
- insert a vertex in each enclosed region of the pull-down graph (vertex=node)
- connect each vertex such that each pull-down is crossed by one pull-up edge (edge=PMOS)



CMOS Combinational Logic

- **Complex CMOS Logic Graphs (XOR)**

5) 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 V_{th}, τ_{PHL}, and τ_{LHL} on the equivalent transconductance values

- we know that

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

$$\left(\frac{W}{L}\right)_{equivalent} = \frac{1}{\sum \frac{1}{\left(\frac{W}{L}\right)_N}}$$

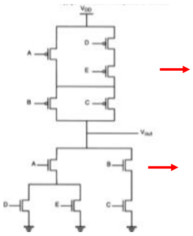
- an equivalent network for parallel-connected transistors with the same sizes:

$$\left(\frac{W}{L}\right)_{equivalent} = \sum \left(\frac{W}{L}\right)_N$$

CMOS Combinational Logic

- **Complex CMOS Logic Equivalent Inverters**

- for example:



$$\left(\frac{W}{L}\right)_{P-equivalent} = \frac{1}{\frac{1}{\left(\frac{W}{L}\right)_A + \frac{1}{\frac{1}{\left(\frac{W}{L}\right)_D} + \frac{1}{\left(\frac{W}{L}\right)_E}}} + \frac{1}{\left(\frac{W}{L}\right)_B + \left(\frac{W}{L}\right)_C}}$$

$$\left(\frac{W}{L}\right)_{N-equivalent} = \frac{1}{\frac{1}{\left(\frac{W}{L}\right)_A} + \frac{1}{\left(\frac{W}{L}\right)_D + \left(\frac{W}{L}\right)_E}} + \frac{1}{\frac{1}{\left(\frac{W}{L}\right)_B} + \frac{1}{\left(\frac{W}{L}\right)_C}}$$

CMOS Combinational Logic

- **Complex CMOS Logic *Stick-Diagrams***

- a *stick diagram* is a graphical way to map a complex CMOS circuit into a layout diagram

- in a stick diagram, draw the general layout configuration that will be used in order to figure out how the diffusion layer contacts are placed.

- a stick diagram has:

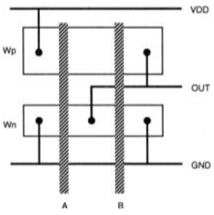
- diffusion regions as rectangles
- metal traces are lines
- contacts are circles
- polysilicon inputs are columns

- the intent of a *stick diagram* is to figure out how the inputs and outputs are connected to the diffusion regions

- 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



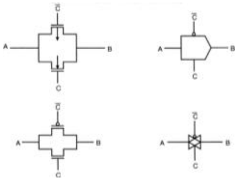
CMOS Combinational Logic

- **Transmission Gate (Pass Gate)**

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



CMOS Combinational Logic

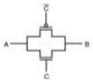
- **Transmission Gate (Pass Gate)**

- When the control signal is HIGH (V_{DD}):

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 be a high-impedance.

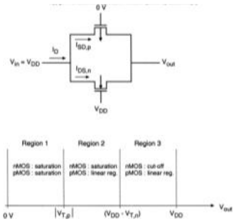
CMOS Combinational Logic

- **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 $V_{out}=0v$

  - As V_{out} moves from 0v to V_{DD}, the regions of operation for the transistors are as follows:



CMOS Combinational Logic

- **Transmission Gate (Pass Gate)**

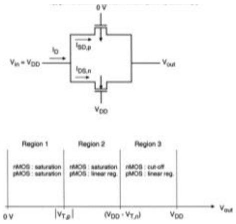
  - we typically model the T-gate as a switch with an equivalent resistance

  - at any point, the *equivalent resistance* is given by:

$$R_{eq,n} = \frac{V_{DD} - V_{out}}{I_{DS,n}}$$

$$R_{eq,p} = \frac{V_{DD} - V_{out}}{I_{SD,p}}$$

$$R_{eq,TG} = R_{eq,n} \,//\, R_{eq,p}$$



CMOS Combinational Logic

- **Transmission Gate (Pass Gate)**

<u>Region 1</u>

$$R_{eq,n1} = \frac{(V_{DD} - V_{out})}{\frac{1}{2}k_n(V_{DD} - V_{out} - V_{T,n})^2}$$

$$R_{eq,p1} = \frac{(V_{DD} - V_{out})}{\frac{1}{2}k_p(V_{DD} - |V_{T,p}|)^2}$$

<u>Region 2</u>

$$R_{eq,n2} = \frac{(V_{DD} - V_{out})}{\frac{1}{2}k_n(V_{DD} - V_{out} - V_{T,n})^2}$$

$$R_{eq,p2} = \frac{(V_{DD} - V_{out})}{\frac{1}{2}k_p\left[2(V_{DD} - |V_{T,p}|)(V_{DD} - V_{out}) - (V_{DD} - V_{out})^2\right]}$$

<u>Region 3</u>

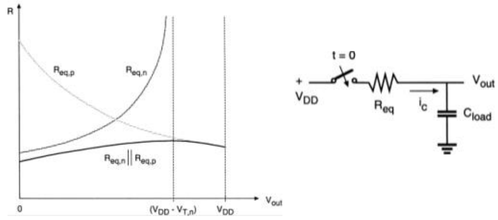
$$R_{eq,n3} = HIGH$$

$$R_{eq,p3} = \frac{(V_{DD} - V_{out})}{\frac{1}{2}k_p\left[2(V_{DD} - |V_{T,p}|)(V_{DD} - V_{out}) - (V_{DD} - V_{out})^2\right]}$$

CMOS Combinational Logic

- **Transmission Gate (Pass Gate)**

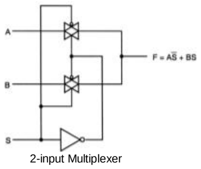
  - as the PMOS and NMOS transistors turn on/off, they keep the overall T-gate resistance LOW



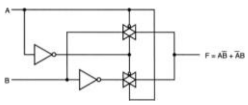
CMOS Combinational Logic

- **Transmission Gate (Pass Gate)**

  - T-gates can be used to create combinational logic



2-input Multiplexer

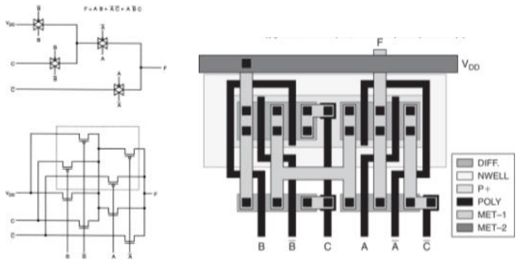


Exclusive OR (XOR)

CMOS Combinational Logic

- **Transmission Gate (Pass Gate)**

  - complex logic using T-gates 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


$F = A\cdot B + \overline{A}\cdot\overline{C} + A\cdot\overline{B}\cdot C$
F
V_{DD}
DIFF.
NWELL
P+
POLY
MET-1
MET-2
B
$\overline{B}$
C
A
$\overline{A}$
$\overline{C}$