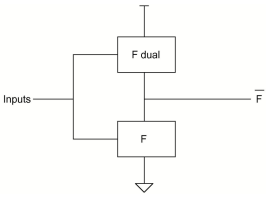
Topics

- Multi-input delay analysis
- CMOS Design
  - Transmission gate
  - latch

CMOS Logic Implementations

- General CMOS combinational logic



- Out = $\overline{F}$
- "OR" operations are performed by parallel-connected nMOS
- "AND" operations are performed by series-connected nMOS
- Inversion is provided by the nature of MOS circuit operation
- pMOS network must be the dual function of nMOS network

CMOS Logic Gate Design

- How do you characterize delay for a gate more complicated than an inverter?

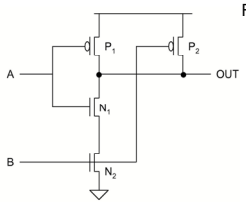
$$t_d \propto \frac{C_L}{k_{eff}}$$

- k_{eff} is determined by the structure of the logic gate

$$R_{eq} \propto \frac{1}{k_{eff}}$$

Delay time analysis

- Multi-input gates



For pull down (falling delay time)

$$\frac{1}{k_{neff}} = \frac{1}{k_{n1}} + \frac{1}{k_{n2}}$$

$$k_{neff} = \frac{k_n}{2}$$

falling delay has doubled

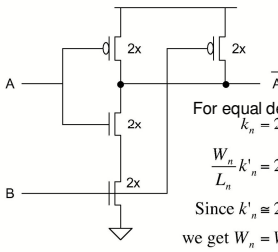
For pull up (rising delay time)

worst case $k_{peff} = k_p$

best case $k_{peff} = 2k_p$

CMOS Logic Implementations

- NAND



For equal delay times

$k_n = 2k_p$

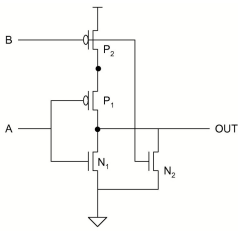
$$\frac{W_n}{L_n} k'_n = 2\frac{W_p}{L_p} k'_p$$

Since $k'_n \cong 2k'_p$ and assume $L_n = L_p$

we get $W_n = W_p$

Delay time analysis

NOR



•For pull down (falling delay time)

$k_{n-eff} = k_n$ (worst case)

•For pull up (rising delay time)

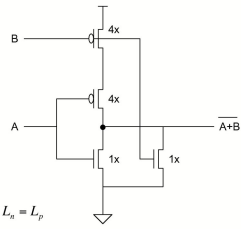
$$\frac{1}{k_{peff}} = \frac{1}{k_{p1}} + \frac{1}{k_{p2}}$$

$$= \frac{k_p}{2}$$

rising delay has doubled

CMOS Logic Implementations

- NOR



•For equal delay times

$$2k_n = k_p$$

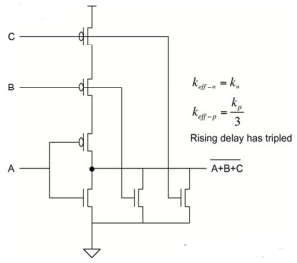
$$2\frac{W_n}{L_n}k'_n = \frac{W_p}{L_p}k'_p$$

Since $k'_n \cong 2k'_p$ and assume $L_n = L_p$

we get $4W_n = W_p$

CMOS Logic Implementations

- Multi-input NOR



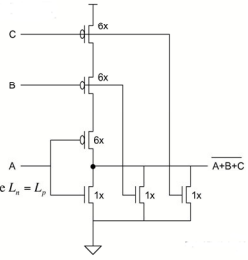
$k_{eff-n} = k_n$

$k_{eff-p} = \frac{k_p}{3}$

Rising delay has tripled

CMOS Logic Implementations

- Multi-input NOR



•For equal delay times

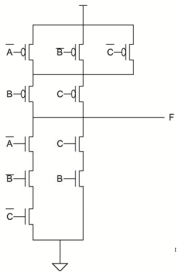
$$3k_n = k_p$$

$$3\frac{W_n}{L_n}k'_n = \frac{W_p}{L_p}k'_p$$

Since $k'_n \cong 2k'_p$ and assume $L_n = L_p$

we get $6W_n = W_p$

CMOS Logic Implementations



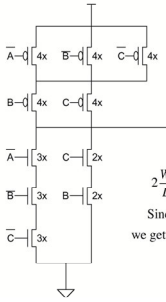
$$k_{eff-n} = \frac{k_n}{3}$$

$$k_{eff-p} = \frac{k_p}{2}$$

Falling delay has tripled

Rising delay has doubled

CMOS Logic Implementations



$$2k_n = 3k_p$$

$$2\frac{W_n}{L_n}k'_n = 3\frac{W_p}{L_p}k'_p$$

Since $k'_n \cong 2k'_p$ and assume $L_n = L_p$

we get $4W_n = 3W_p$

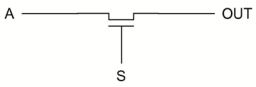
CMOS Logic Gate Delays

- Increasing transistor sizes can reduce the delays, but it
  - Increases area
  - Increases load capacitance for driving gates
- Multi-input gates may not always be good

CMOS Logic Implementations

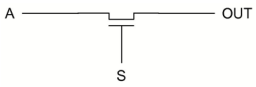
- Transmission Gate

A	S	OUT
0	0	Z
0	1	0
1	0	Z
1	1	1



CMOS Logic Implementations

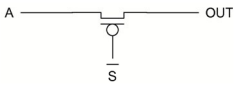
- Transmission Gate



  - When S is low, the output is at high impedance
  - When S is high, the output follows A
    - However, OUT can only rise to V_{DD}-V_T, at which point the transistor will shut off

CMOS Logic Implementations

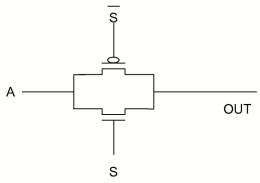
- Transmission Gate



  - When S is low, the output is at high impedance
  - When S is high, the output follows A
    - However, OUT can only fall to V_T, at which point the transistor will shut off

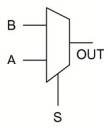
CMOS Logic Implementations

- Transmission Gate
  - Requires both nMOS and pMOS transistors
  - Single nMOS or single pMOS will cause signal degradation



CMOS Logic Implementations

- Multiplexer



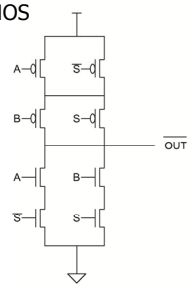
A	B	S	OUT
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

$$OUT = A\overline{S} + BS$$

CMOS Logic Implementations

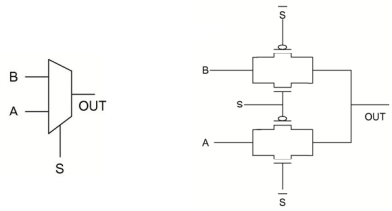
- Multiplexer in CMOS

$$OUT = A\overline{S} + BS$$



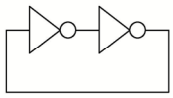
CMOS Logic Implementations

- Multiplexer in TG



CMOS Logic Implementations

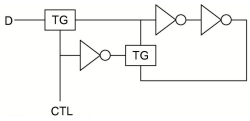
- Memory
  - Use feedback loops to store bits



  - How do you get the bit in the loop?

CMOS Logic Implementations

- Memory
  - Use transmission gates to control entry to the loop



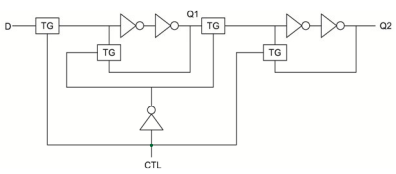
  - Level sensitive D-latch

CMOS Logic Implementations

- Latches
  - Level sensitive latches do not allow you to isolate output from input when control is high.
  - Solution is to use a master-slave setup where master latches input and then slave latches the output

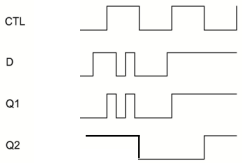
CMOS Logic Implementations

- Master-slave latch



CMOS Logic Implementations

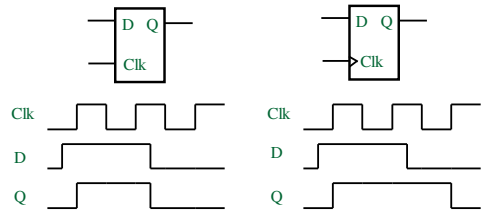
- Master-slave latch



- Negative edge-triggered flip-flop

CMOS Logic Implementations

- Latch- stores if clock is low transparent if clock is high
- Register- stores if clock rises, never transparent (mostly)



CMOS Logic Implementations

- Setup time is how much time **before** the clock edge that the data should be ready
  - If setup time is not met, the data will not have time to get through transmission gate and into the feedback loop
- Hold time is the time that the data is required to be stable **after** the clock edge.
  - If hold time is not met, invalid data may get past the transmission gate and into the feedback loop (race-through problem).

Timing Definitions