ECE 27000 | Chpt 1

Intro. to Digital Design

- Analog - Time varying signals that have a continuous range of voltages/currents. (Reality, variable V/I, ↓ stability/accuracy)
- Digital - Discrete signal of 0/1, Low/High, True/False. (Hides pitfalls of analog by using discrete signals of 0/1's called bits)
  - 0-1 is algebraicly low/high voltages.

Logic Circuits/Gates

- Buffer:  , $x \Rightarrow x$
- NOT Gate:  , $x \Rightarrow \bar{x}$ / x'
- AND Gate:  , $A, B \Rightarrow A \cdot B$
- OR Gate:  , $A, B \Rightarrow A + B$
- NAND Gate: 
- NOR Gate: 
- XOR Gate:  , $A, B \Rightarrow A \oplus B$ ★ Multi-input XOR = 1 if odd # of 1s are inputted
- XNOR Gate:

CMOS

- ↑ V, CMOS runs faster + noise immunity is better, but consume more power.
- $P \propto CV^2f$, C - elec. C of signals, V - ps voltage, f - signal switching freq.

Programmable Devices

- ROMs (Read-only memory) - stores 2D array of 2^n rows x b columns, of bits
  - ex. n=16 + b=8 ⇒ 64 Kbytes.
  - Used for complex + non-time critical functions
- PLAs / PALs (Programmable logic arrays) - can connect data with logic gates
  - Generally called PLDs (Programmable logic devices)
  - CPLDs are multi PLDs connected.
- FPGA (Field-programmable gate array)
  - Lots of small configurable logic blocks (CLBs)

More common nowadays than CPLDs ★

ECE 27000 | Chpt 2

Binary, Decimal, Hexadecimal

- Groups of 8 bits = bytes, 4bits = nibble.
- Hexadecimal is used to shorten long binary w/ 2 hexadecimal digits = 1 byte. 0x can be used as a hex prefix.
- Binary ⟷ Hex

0100101101110010_2

2 5 B 2 $\Rightarrow 25B2_{16}$

Negative Nums

- Signed-Magnitude Rep: Has sign bit @ MSB to det. sign.
  - Range of $-(2^n-1)$ to 2^n-1 w/ two rep. for 0.
- Complement Representation: All bits are swapped + 1 is added to get negative value of org. #.

ASCII

Uses 7 bits for characters, giving 128 characters.

BCD

Each digit gets 4 bits, to represent 0-9 (Any other 4 bit val is invalid)

ECE 27000 | Chpt 3

Boolean Algebra

- Or Switching Algebra
- Axioms: $X=0$ iff $X\neq 1$, $X=1$ iff $X\neq 0$.
- NOT Gate: X', AND Gate: $X\cdot Y$, OR Gate: $X+Y$.
- Single Var. Theorems:
  - Identities: $X+0=X$, $X\cdot 1=X$
  - Null Elem.: $X+1=1$, $X\cdot 0=0$
  - Idempot.: $X+X=X\cdot X=X$
  - Involution: $(X')'=X$
  - Complements: $X+X'=1$, $X\cdot X'=0$
- Two + Three Var. Theorems
  - Commutativity: $X+Y=Y+X$, $X\cdot Y=YX$
  - Associativity: $(X+Y)+Z=X+(Y+Z)$, $(XY)Z=X(YZ)$
  - Distributivity: $X\cdot Y+X\cdot Z=X(Y+Z)$, $(X+Y)(X+Z)=X+YZ$
  - ~~Combining: $X\cdot Y+X\cdot Y'=X$, $X\cdot(X+Y)=X$~~
  - Covering - $X+X\cdot Y=X$, $X\cdot(X+Y)=X$
  - Combining - $X\cdot Y+X\cdot Y'=X$, $(X+Y)\cdot(X+Y')=X$
  - Consensus - $XY+X'Z+YZ=XY+X'Z$

    $(X+Y)(X'+Z)(Y+Z)=(X+Y)(X+Z')$
- N-Var Theorems
  - Demorgan's Law: $(X_1\cdot X_2'\cdots X_n)'=(X_1'+X_2'+\cdots+X_n')$

    $(X_1+X_2+\cdots X_n)'=(X_1'X_2'\cdots X_n')$
  - Shannon's exp. Theorems:

$$F(X_1,X_2,\cdots X_N)=X_1F(1,X_2,\cdots,X_N)+X_1'F(0,X_2,\cdots,X_N)$$
$$=(X_1+F(0,X_2,\cdots,X_N))(X_0+F(1,X_2,\cdots X_N))$$

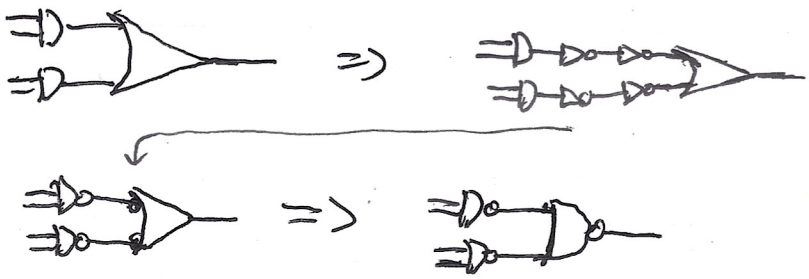
- Duality
  - Swap All 0/1 and +/·, Eq is true.
- Compliment: $F(X)\Rightarrow\overline{F(X)}$

ECE 27000 | Chpt 3 cont'd

Combinational-Circuit Synthesis

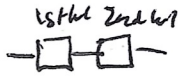
- We can simplify circuits by having inputs that will be calculated from more complex circuits ([redacted] (clock/lock signal)). We can also use bool algs w/ them.
- NAND + NOR gates are usually faster in CMOS.
- To utilize them insert two theoretical not gates between gates and update circuits, w/ pushing inputs through etc.

ex.

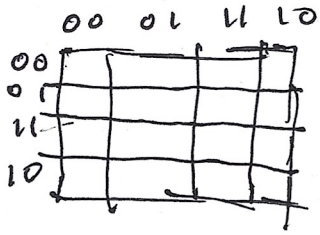


- Any splits should happen before the not gate to each input.

- Minimization
  - Minimize # of 1st lvl gates
  - Minimize # inputs on each 1st lvl gate
  - Minimize # inputs on the 2nd lvl gate.



- Karnaugh Maps - Used like truth tables to minimize.



(I will expand on later)

- Timing Hazards - When a false output is triggered b/c prop. delay, or when multiple changes happen that also cause this.
  - Static 1 Hazard - Should be 1 but flips to 0 for a sec (usually SoP)
  - Static 0 Hazard - Should be 0 but flips to 1 for a sec (usually PoS)
  - Dynamic Hazard - Changes multiple times in the output.

ECE 27000 | Chpt 3 cont'd

Representations of Logic Funcs

- Definitions:
  - Literal: Var/Complement of var (X/X')
  - Product Term: Single Literal or Product of multiple literals (X/XY/XY')
  - Sum-of-products expression (SoP): Logical sum of product terms (XY+YZ)
  - Sum Term: Single Literal or Sum of multiple literals (X/X+Y)
  - Normal Term: Prod/Sum Term which each variable only appears once. (XYZ)
  - n-var minterm: Product Term with n literals (XYZ - 3-var min.)
  - n-var maxterm: Sum Term with n literals (X+Y+Z - 3-var max.)

- Minterm/Maxterm Table (Max = opp of literal value, min = same as literal value)

	X	Y	Z	Min.	Max.
#0	0	0	0	X'Y'Z'	X+Y+Z
1	0	0	1	X'Y'Z	X+Y+Z'
2	0	1	0	X'YZ'	X+Y'+Z
3	0	1	1	X'YZ	X+Y'+Z'
4	1	0	0	XY'Z'	X'+Y+Z
5	1	0	1	XY'Z	X'+Y+Z'
6	1	1	0	XYZ'	X'+Y'+Z
7	1	1	1	XYZ	X'+Y'+Z'

$F = \Sigma_{(X,Y,Z)}(0,1,2)$ = SoP of minterms 0,1,2
↗ on-set = F is 1 for 0,1,2 only

$F = \Pi_{(X,Y,Z)}(0,1,2)$ = PoS of maxterms
↗ off-set = F is 0 for 0,1,2 only

$F = \Sigma_{XYZ}(\,) = \Pi_{(X,Y,Z)}(\overline{\,})$ where $\overline{\,}$ is the opposite set of maxterms

Karnaugh Maps

(minterms listed)

YZ \ WX	00	01	11	10
00	0	4	12	8
01	1	5	13	9
11	3	7	15	11
10	2	6	14	10

- To simplify, Circle all adjacent 1s/0s if/for doing SoP/PoS respectively. in powers of 2 groups.
  - The variables that don't change in the group will be represented in the term in the final product expression.
  - Repeat for all groups

- Can be used to prevent errors
  - Adjacent groupings w/ no overlap cause static errors.
  - Add a group overlapping the adjacency to solve.

ECE 27000 | Chpt 5

Verilog

- Verilog is a HDL (Hardware Description Lang.) where you can design, simulate, and synthesize almost any digital circuits.
- Modules build verilog (functions) - usually 1 a file w/ .v prefix
  - ex.
```
module Example (X, Y, Z);
    input X, Y;
    Output Z;

    assign Z = X & ~Y;
endmodule
```
- Module structure:
  - module module-name (port-name, port-name, ..., port-name);
    - input decl.
    - Output decl.
    - inout decl.
    - Net decl.
    - var decl.
    - Param decl.
    - Func. decl.
    - task decl.

    - Concurr. statements

    endmodule
  - input [msb:lsb] ident, ident, ..., ident;
    Output [msb:lsb] ident, ...; // (only read by other mod.)
    inout // (Can be used for in/out)

    [msb:lsb] is optional, but indicates index of most sig bit / least sig bit. This rep. vector 8bit inputs (range must be 8 to 0)
- All vars have to be 0, 1, X, Z (X = unknown, Z = high imp., used in 3-state logic)
- Bitwise op: &, |, ^, (~^, ^~), ~ (^ XOR, ~^ ^~ XNOR)
- Nets - enable connectivity, default is wire
- reg - store bits for use in module
- integer - signed 32 bit # (used for for loops)
- Literals: L'B N... where L is length of the literal in bits, B is the base of the literal, and N is the literal numbers to translate. (b/d/h/o) etc.
- Parameters: Constants / used to store literals.
  - [...:...] ... = vector, ... [...:...] = array ∈ can be combined as vector is like bits.
  - Vector uses {}, N{i} produces N len vector of 1s.
- Struct mod, busser is out => M (out, in1, in2, ... inN)

~~Can go more than 1 deep.~~

ECE 27000 | Chpt 5 cont'd

Logical op. + Expressions

- If any bit is 1, the value is true.
- Expressions return 1'b1 or 1'b0 (T/F) with 0-left padding if assigned to larger bit size
- Operators: &&, ||, !, ==, !=, >, >=, <, <=
- Conditional operator: X ? Y : Z - evaluates as Y or Z depending on X.
  - Same as if X:
    ret Y
    else
    ret Z

Compiler Directives

- `include filename, `define identifier text
@ - Like C

Verilog Models

- Structural Model - Uses inbuilt gates + nets/wires for functions.
  - Use instatiators that are unique, aka: not U1 (A, B);
- DataFlow Model - Uses bitwise operators + assign for functions.
- Behavioral Model - Uses traditional coding logic and always for functions
  - always @ (sig1 or sig2...), always @ (sig1, sig2...), always @ (*), always @ (posedge sig1), always @ (negedge sig1), always
    - # 1 + #2 trigger when those signals change, #3 trigger when any signal mentioned changes in body, #4/#5 = negedge + edge
    #5 is always triggered. The body will between a begin/end.
  - begins can be named tracked (begin: name)
  - if (cond) exp; else if (cond) exp; else exp;
  - Case (C)
    case1, case2: expA;
    case3: expB;
    default: expC;
    endcase
  - for loops work the same, begin/end instead of brackets.
  - ~~funcs/tasks, are mini modules that don't use {} functions; testbench~~

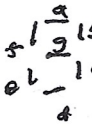
ECE 22000 | Chpt. 6-7

Read-only Memories (ROMs)

- A ROM is a memory with 2^n data entries of size b.
  - To access, there are n address pins and b output pins
- FPGAs use small ROMs and Lookup Tables (LUTs)

Decoding + Selecting

- Uses n inputs + 2^n outputs + enable signal (turn on enable to use)
- Translates the binary input and activates the corresponding output pin (0 - 2^n-1)
- Any pin can be active low/high
- Seven Segment Displays use Decoders:



7-bit number w/ MSB being g.

- Binary Encoders do the opposite, 2^n inputs → n outputs
  decimal → binary. (Pins 0 - 2^n-1)

Multiplexing

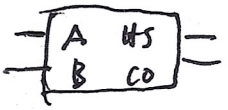
- Uses a decoder + Data lines
- n inputs + 2^n datalines + 1 output + enable signal
- Links the correct dataline to output depending on the n-len. address.
- Demultiplexers do the opposite n inputs + 1 data line + 2^n output lines.
  - Directs data to specific wire.

Priority Encoders

- Fixes issues of basic encoder which needs only one input high
- Encodes the highest significant bit that is active and encodes it into binary
  0011 → 00

Addings + Subtractions

- Half adder

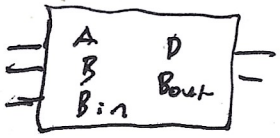


$HS = A \oplus B$ ← Half Sum
$CO = A\,B$ ← Carry Out

- Full adder

$S = A \oplus B \oplus Cin$ ← Sum
$CO = AB + A\,Cin + B\,Cin$ ← Carry Out

- We can chain full adders to add more bits and make a ripple adder. (But ineff.)
- Full subtractor



$D = A \oplus B \oplus B_{in}$ ← Difference
$B_{out} = A'B + A'B_{in} + B \cdot B_{in}$ ← Borrow out

- Carry Look ahead adder
  - $S_i = a_i \oplus b_i \oplus c_i$
  - Uses next-level logic to generate the carries for each bit.

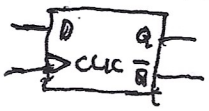
~~Binary Codes for Decimals~~

- ~~Regular Binary~~
- Gray Code

EEE 22000 | Chpt 9

State Machine

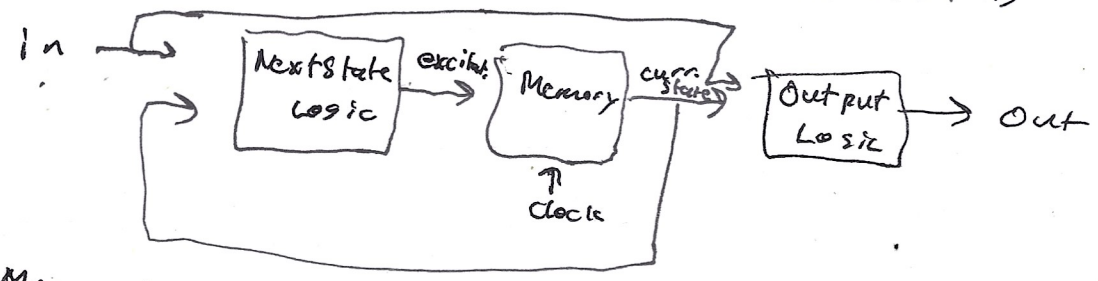
- Clock Signals are used for State Machines/Sequential Circuits
  - Active high = triggered on rising edge
  - Active low = triggered on falling edge.
- D flipflops commonly used to store data in state machines



D is data line, Q is data, $\bar{Q}$ is data complement
Clk is clock, D is stored to S on clock rising edge.

State Machine Structure

- Mealy state-machine - Output depends on state + inputs



- Moore state - machine output depends on state only.
  - (Removes connection from in to output logic.

State Machine Design

1. Make a state table (State/output)
2. Use state minimization
3. Choose state variables and assign combos to the named states
4. Sub state-var combos to create transition/out table that shows the next state.
5. Choose a flipflop to store the state (usually D)
6. Construct excitation table (what vals needed for next state)
7. Derive excitation equations
8. Derive output eqs.
9. Draw logic diagram showing state-var storage, reg. excitation, and output eqs.

- State Diagram ex:

Moore Mealy

State Machine cont'd Design

- Excitation Logic = Next State Logic
- Transition Table

Q1	Q0	IN 0	IN 1
0	0	0 0	0 1
0	1	0 1	1 0
1	0	1 0	1 1
1	1	1 1	0 0

$Q_1^* Q_0^*$

- State Table

S	IN 0	IN 1
A	A	B
B	B	C
C	C	D
D	D	A

- State Output table
  - Mealy:

S	IN 0	IN 1
A	A,0	B,0
B	B,0	C,0
C	C,0	D,0
D	D,0	A,1

  - Moore:

S	IN 0	IN 1	Out
A	A	B	0
B	B	C	0
C	C	D	0
D	D	A	1

- Excitation Equations calc $Q_1^* Q_0^*$.
- Output Equations calc Out.

State Minimization

1. Partition States based on output
2. Keep Partitionins based on nextstate partition locations disserences when in=0 or in=1.

ex AB or CD ⇒ AB $\xrightarrow{0}$ AB, AB $\xrightarrow{1}$ CA

0,1 s.

- # FF = $\lceil \log_2(\#S) \rceil$

Verilog

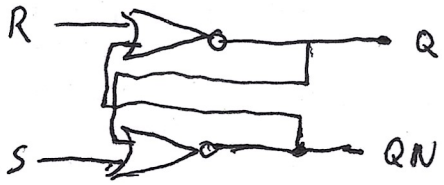
- Next state Logic : always_ff + cases (use =)
- State Mem + Output Logic : always_comb (use <=)

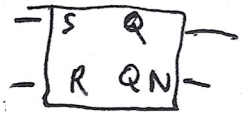
Bistable Elements

- Bistable - Two stable states
- Metastable - A third stable point inbetween logic levels (not truely stable).

Latches + FlipFlops

- S-R Latch

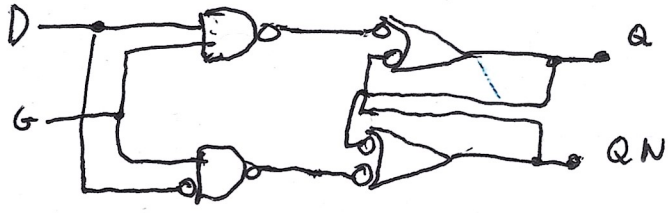




S	R	Q	QN
0	0	Q	QN
0	1	0	1
1	0	1	0
1	1	0	0

Latches onto the set/reset.

- D Latch



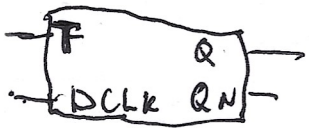
G	D	Q	QN
1	0	0	1
1	1	1	0
0	X	Q	QN

Latches Data when enabled

- D Flip Flop

Updates Data Value @ Q from D on edge of clock signal (usually pos edge)
Sometimes has enable

- T Flip Flop



Toggles Data value @ Q when T=1 on edge of clock signal (usually pos edge)
Sometimes has enable

Counters

- Ripple Counter - links bunch of TFFs so that whenever a bit updated from 1→0 it ripples the bit forward in QN. (Very Slow)
- Synchronous Counter - Links TFFs with one clock signal and a AND gate w/ master enable + the previous bit. (All bits update at the same time)

Shift Registers

- Data line + Clock
- Shifts all bits left/right and inserts new bit from data line.
- Can use serial i/o or parallel i/o (One data line vs multiple)

More Counters

- Ring Counter - Link Out back to in to loop the bits making a ring of repeated numbers
- Johnson Counter - Ring Counter but QN is linked to in instead, leading to twice as many States.
- Linear Feedback Shift-Reg Counter - Uses xor gates to Loop through all 2^n-1 possible states.

Iterative vs Sequential

- Iterative - Uses no Storage or clock
- Sequential - Uses FFs and requires clock pulses.

ECE 27000 | Chpt 12 [redacted]

Verilog Test Benches

- Steps:
  1. Declaration of the test-bench module itself.
     - No inputs or outputs
  2. Instantiation of the component to be test (CUUT/DUT)
  3. An always block to create a free running clock.
  4. Statements to init. DUT apply multiple test inputs and check output.
- Construction Methods:
  - Try to visit every normal state and take all normal transitions.
  - Try to test every "feature" of the DUT.

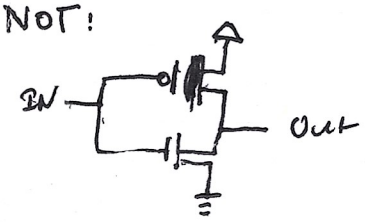
Synchronizer Failure + Metastability

- Synchronizer Failure - using synch. output while still in metastable state
  - Can be avoided by waiting long enough before using output.
- To get out of this state, force FF into a valid state using the published specifications.
- Or wait long enough for metastability to be resolved.
- t_r - metastability resolution time or how much time before metastability causes a synch. fail.
  - Usually $t_r = t_{clk} - t_{comb} - t_{setup}$
    - t_{clk}: clock period
    - t_{comb}: comb logic prop delay
    - t_{setup}: FF setup time
  - Fast clocks often are used, so minimize $t_{comb} + t_{setup}$ to maximize t_r for more reliability.
- t_s - setup time, t_h - hold time (published) bracket the decision window
  - D changes outside of d.w. = output will settle in t_{pd}
  - else metastability is possible and can persist past t_r.
- $MTBF(t_r) = \frac{e^{t_r/\tau}}{T_0 \cdot f \cdot a}$ - mean time betw sync fails.
  - f: clock, a: # changes per sec
  - T_0 + τ are electrical characteristics
- Setup viol = Misses data, Hold viol = Gets wrong data
  - setup viol: data arrives too late
  - hold viol: data leaves too early.
- t_{skew} - time gap between identical clock signals
- Setup time constraint: $T_c >= t_{pd} + t_{setup} + t_{pcg} + t_{skew}$
  - t_{pcg}: time to stable @ FF pd (sometimes)
- Hold time constraint: $t_{hold} < t_{ccg} + t_{cd} - t_{skew}$
  - t_{ccg}: cd of g
  - t_{cd}: time to change (like t_{pd} but not stable yet just when out starts changing)
- t_s is before clock pulse, t_h is after.

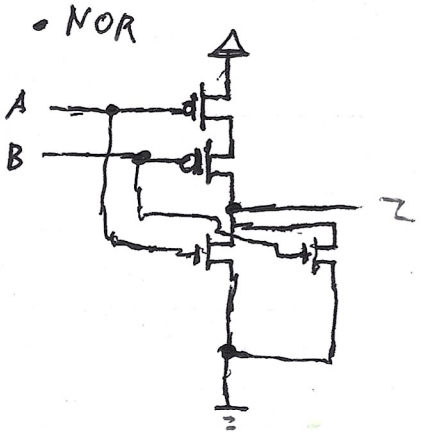
CMOS Logic Circuits

- I/O is certain voltage + noises. Anything outside Logic 1 / Logic 0 ranges is invalid.
- Typical CMOS operates at 5 V, but some circuits use lower voltage to save power.
  - 0-30% of V_{DS} is Logic 0, 70%-100% of V_{DS} is Logic 1
  - 30%-70% is invalid
- $R_{DS} \approx 1M\Omega$ when off, ($V_{GS} = 0$) $R_{DS} \approx 10\Omega$ when on. ($V_{GS} \approx 5V$ (N), $-5V$ (P))
- NMOS & PMOS are combined in CMOS circuits, w/ PMOS being wired to V_{DS} at the top, NMOS being wired to GND at the bottom and output in the middle
  - PMOS is wired as the complement of the logic function, while NMOS is wired normally for inverting gates (ignoring the wire NAND)
  - Non-inverting gates have the output of the complementary inverting gate linked to a simple CMOS not.

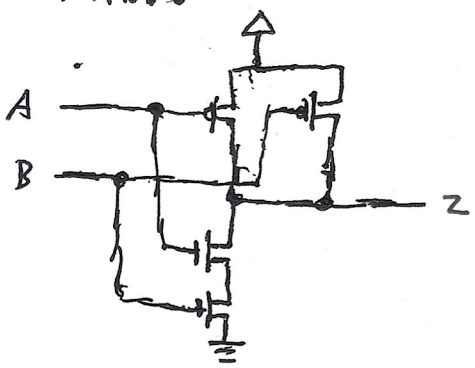
- NOT:



- NOR



- NAND:



Buss/AND/OR can be acheived by attaching Z from one of these gates to a NOT Gate.

Electrical Behavior of CMOS Circuits

- Datasheet for CMOS Devices

V_{IH}/V_{IHmin} = Min. Logic High Level for input (usually 70% V_{CC})

V_{IL}/V_{ILmax} = Max Logic Low Level for input (usually 30% V_{CC})

V_{OH}/V_{OHmin} = Min Logic High for Output (usually $V_{CC}-0.1 \approx V_{CC}$)

V_{OL}/V_{OLmax} = Max Logic Low for Output (usually GND $+0.1 \approx V_{GND}$)

↑ More error when out current ↑ ↓

(V_{CC} has error and V_{OH}/V_{OL} are calc. taken that into acc.)

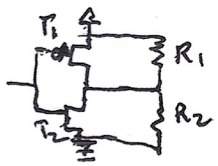
$I_H/I_L \approx \pm 1\,\mu A$ = Max current in input to/through (Cause FET tech.)

- DC Noise Margin -

- Low Noise Margin (NML) = $|V_{ILmax} - V_{OLmax}|$
- High Noise Margin (NMH) = $|V_{OHmin} - V_{IHmin}|$

- When attached to res. load, behaves unideally.

Ex.



- Convert to thevenin form (the R load)
- Replace transistors w/ Resistors (1MΩ for off, 100Ω for on) (Do not use as it should act like OC)
- Calculate voltage division w/ low res/on transistor resistance (200Ω for on P, N)

Ex. (w/ nmos R)

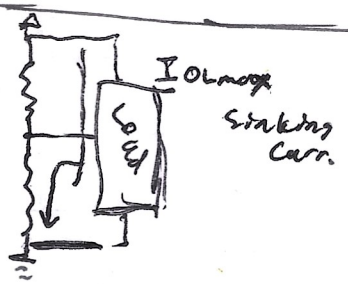
$V_{out} = V_{thev} \left(100/(100+R_{thev})\right)$ (For Vin High)

$V_{out} = V_{thev} + (V_{CC} - V_{thev})\left(R_{thev}/(200+R_{thev})\right)$ (For Vin Low)

↑ Pmos R

- Hard to calc (not given parameters)

- I_{OLmax} - Max sink current when Out Low
- I_{OHmax} - Max source current when Out High



TTL load ↓ precise V ↑ more curr. vs CMOS load

$R_{p(on)} \approx \dfrac{V_{CC} - V_{OHmin}}{|I_{OHmax}|}$

$R_{n(on)} \approx \dfrac{V_{OLmax}}{I_{OLmax}}$

Electric Behavior of CMOS Circuits cont'd

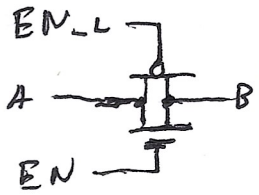
- Non-ideal inputs,
  - When inputs to logic gates aren't ideal, this causes unideal output voltages which worsen with loads, due to the transistors both not having high resistances.
  - This also causes current wastage + power wastage
- Loading beyond rated fanout (max inputs gate can drive) causes
  - $V_{OL max}$ to be exceed
  - $V_{OH min}$ to not be met
  - Prop delay ↑, Rise/Fall times ↑, Op temp ↑ = ↓ reliability + faster failure.
- Don't leave floating inputs, tie them to other outputs or logic 1/0, according to switching logic rules.

CMOS Dynamic Behavior

- Transition time
  - Rise time: t_r - time for signal to rise from 10% to 90%.
  - Fall time: t_f - time for signal to fall from 90% to 10%
- Propagation Delay
  - Time for Output to Change from Input
  - t_{pHL}: time from input change to output change H→L (measured @ 50%) for in + out
  - t_{pLH}: time from input change to output change L→H
- Power Dissapation
  - $P_T = C_{PD} \cdot V_{CC}^2 \cdot f$
    - C_{PD} - Power Disp. Capacitance, rep. current flow changes
    - f - freq of transitions / output changes

Other CMOS IN/Out Structs

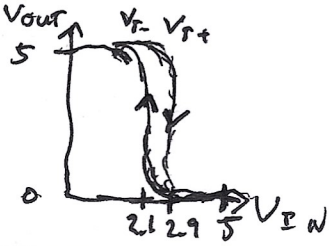
- Transmission Gate



EN_L & EN always opposite
EN↑ = 1-5Ω of imp.
EN↓ = A ⇏ B is d.c.

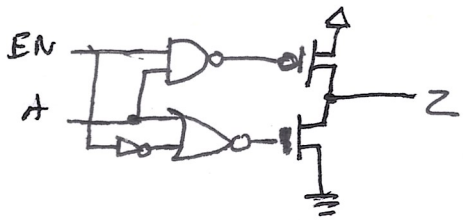
→ A → B prop delay ↓↓ so used in larger scale CMOS devices
* (Multiplexers / Flip Flops)

- Schmitt Trigger Inputs

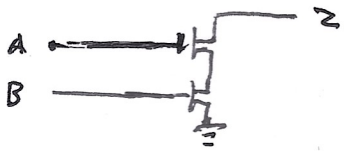


Low → High EN threshold is V_{T+}
High → Low IN threshold is V_{T-}
(Regular Input has one threshold)

- Three State Output
  - Hi, Lo, Hi-Z (High Impedence)
  - Hi-Z ≈ Not connected.
  - Add enable pin as such →



- Open Drain - No connection to Vcc, Only low out is outputted, Otherwise its open. Used w/ a external pull up resistor to achieve higher V.



High → Low is fast
Low → High v/ slow.