EECS 281

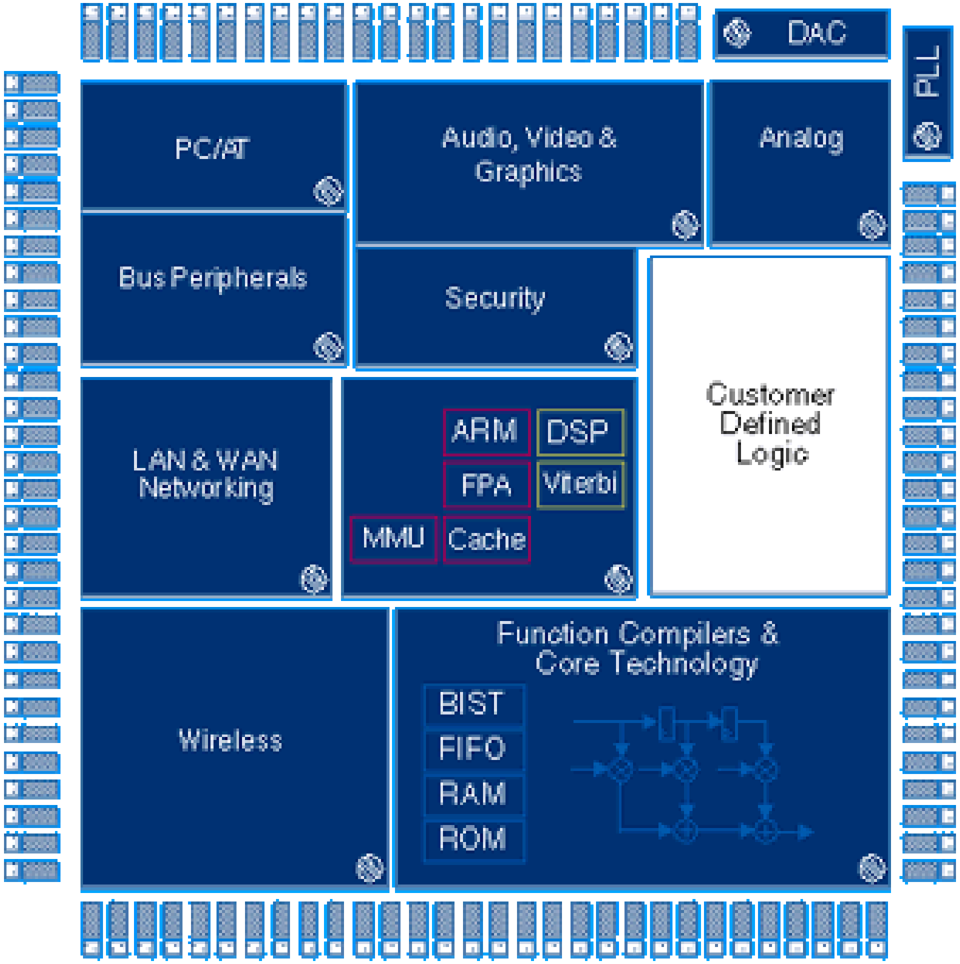
Logic Design

LECTURE 1: Digital Modeling

Instructor: Francis G. Wolff
fxw12@case.edu
Case Western Reserve University

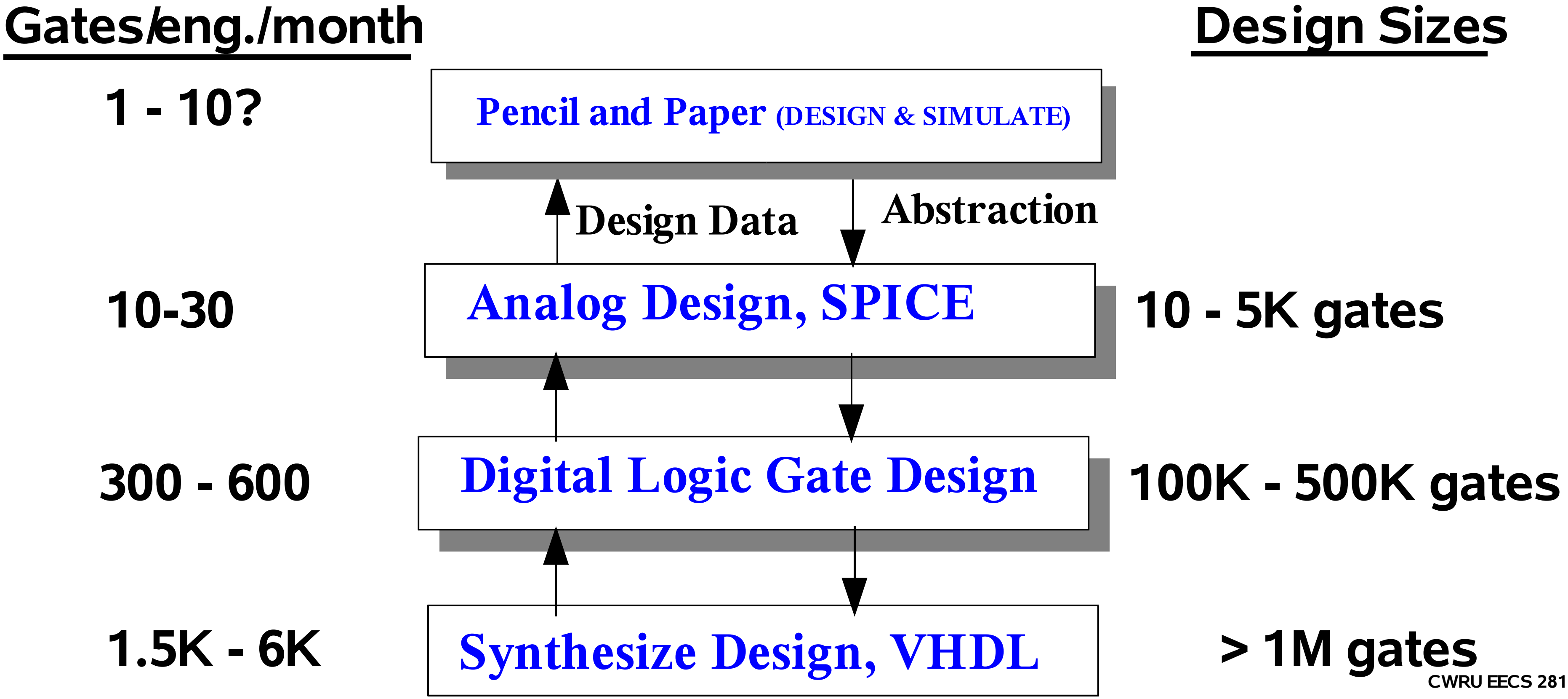
SoC: System on a chip (beyond Processor)

- The 2007 prediction: SoC's will be > 1B transistors



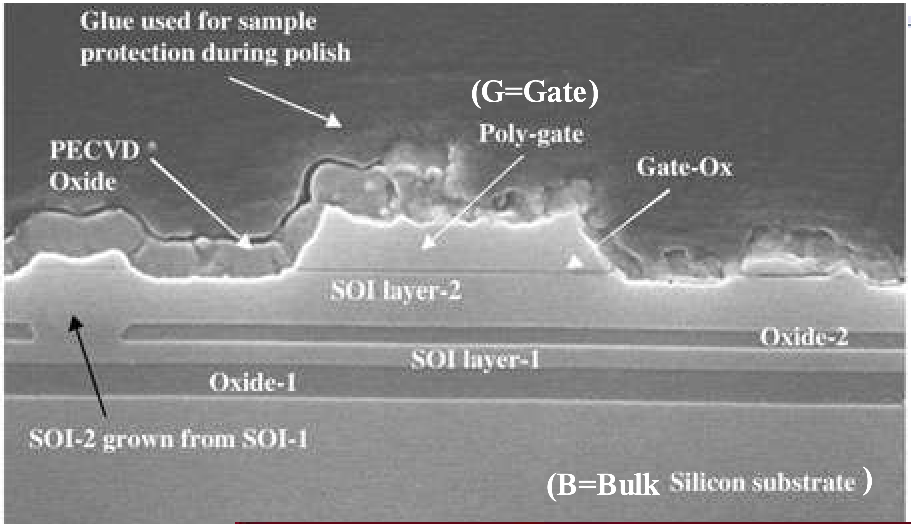
Design approaches

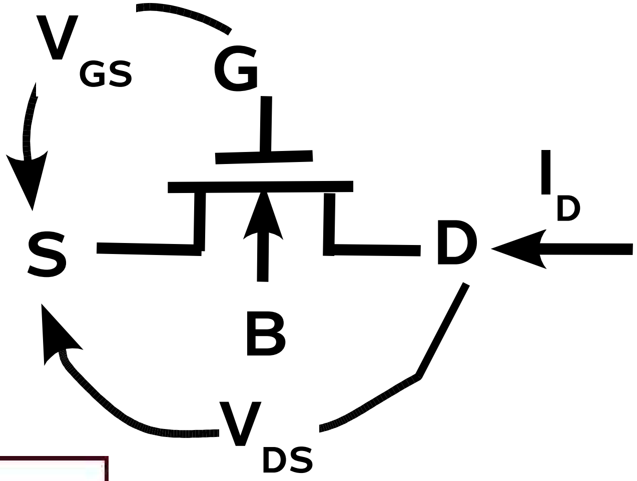
- Goal of each design flow methodology is to increase productivity of the design engineer
- Increasing the abstraction level of the design methodology and tools is one approach:

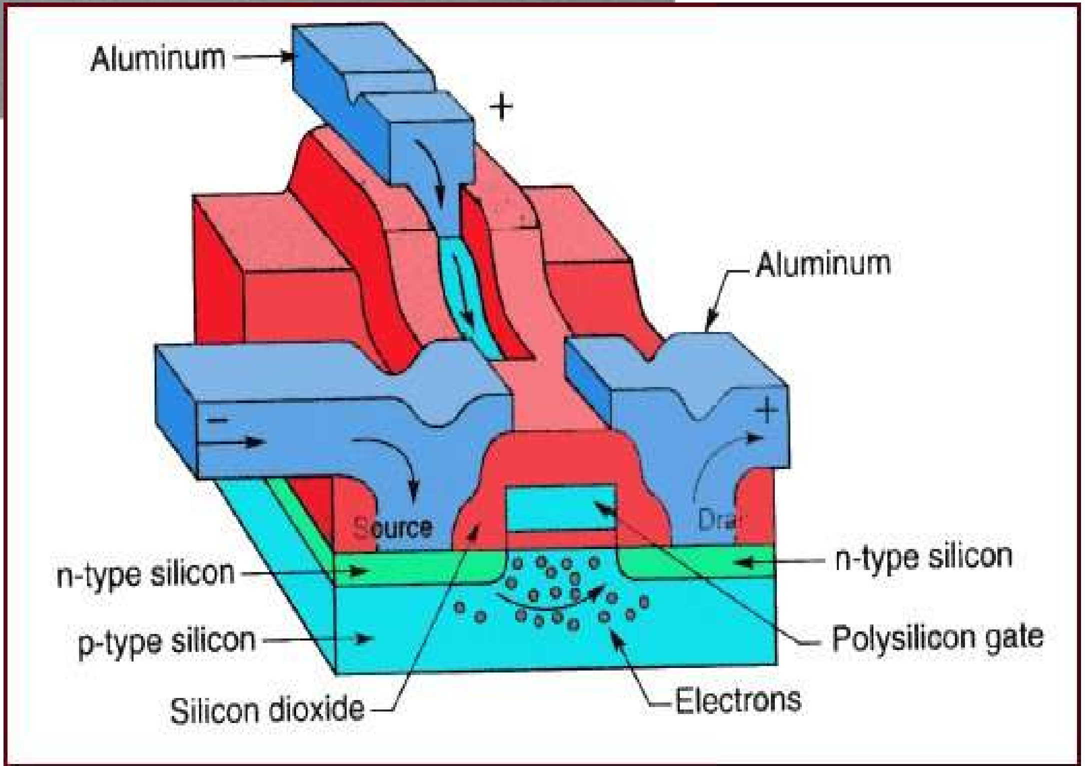


Gates/eng./month		Design Sizes
1 - 10?	Pencil and Paper (DESIGN & SIMULATE)	
10-30	Analog Design, SPICE	10 - 5K gates
300 - 600	Digital Logic Gate Design	100K - 500K gates
1.5K - 6K	Synthesize Design, VHDL	> 1M gates

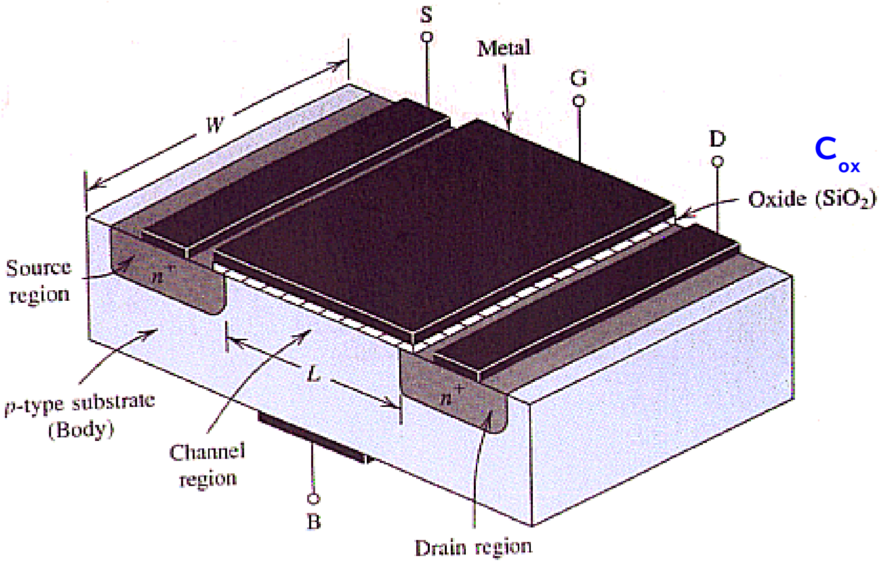
Levels of Abstraction: NMOS Transistor

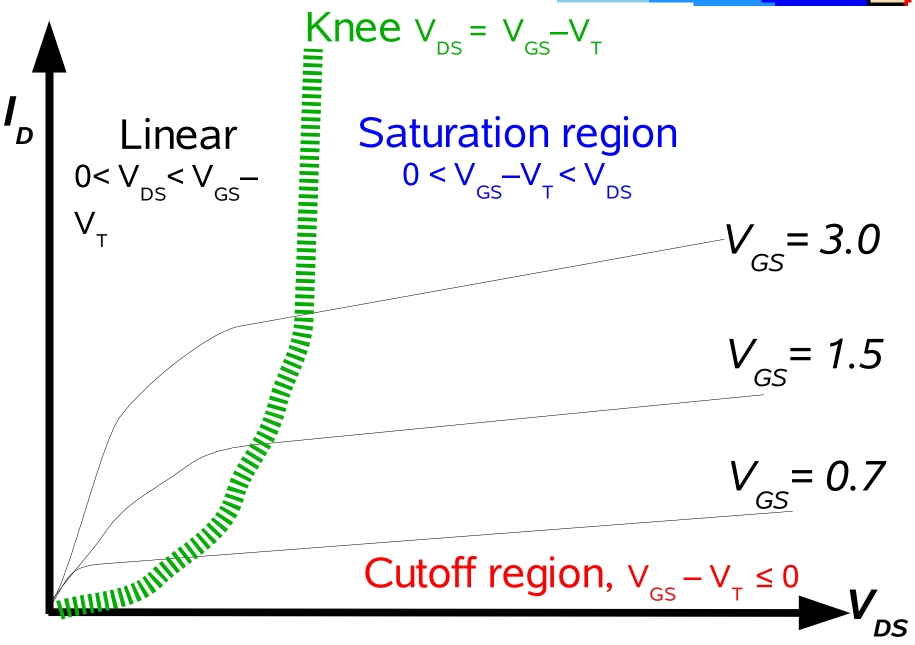






Shichman-Hodges model (spice level 1)





Knee $V_{DS} = V_{GS}–V_T$

Linear
$0< V_{DS}< V_{GS}– V_T$

Saturation region
$0 < V_{GS}–V_T < V_{DS}$

Cutoff region, $V_{GS} – V_T \leq 0$

$$I_D = \begin{cases} 0, & V_{GS} – V_T < 0, \text{ cutoff region} \\ k'(W/L)((V_{GS} – V_T)V_{DS} – ½V^2_{DS})(1\text{-}\lambda V_{DS}), & 0 < V_{DS} < V_{GS}–V_T, \text{ linear (triode) region} \\ ½k'(W/L)(V_{GS} – V_T)^2(1\text{-}\lambda V_{DS}), & 0 < V_{GS}–V_T < V_{DS}, \text{ saturation region} \end{cases}$$

H. Shichman and D. A. Hodges. Modeling and simulation of insulated-gate field-effect transistor switching circuits. Solid-State Circuits, IEEE Journal of ,Vol. 3, Issue 3, Sep 1968, pages 285-289.

Shichman-Hodges model: I_D parameters (~8)

$$I_D = \begin{cases} 0, & V_{GS} - V_T < 0, & \text{cutoff region} \\ k'(W/L)((V_{GS} - V_T)V_{DS} - \frac{1}{2}V^2_{DS})(1\text{-}\lambda V_{DS}), & 0 < V_{DS} < V_{GS} - V_T, & \text{linear region} \\ \frac{1}{2}k'(W/L)(V_{GS} - V_T)^2(1\text{-}\lambda V_{DS}), & 0 < V_{GS} - V_T < V_{DS}, & \text{saturation region} \end{cases}$$

k' = KP = μC_{ox} = (n-channel surface mobility)(gate oxide)
= process transconductance

W, L = effective channel width, length

λ = LAMBA = channel length modulation (i.e. slope)

V_T $= V_{T0} + \gamma(\sqrt{(V_{SB}+\psi)} + \sqrt{\psi})$ = threshold voltage

V_{T0} = VT0 = threshold voltage without substrate bias

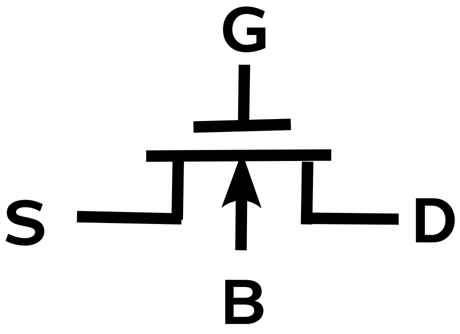
γ = GAMMA = body bias coefficient

ψ = PHI = bulk fermi potential = 2 | (k TEMP / q) ln(NSUB/n_i) |

$$t_{min} = \text{switching speed} = \frac{2\,L^2\,C_{ox}\,V_{dd}}{k'\,(V_{dd} - V_T)^2}$$

Shichman-Hodges model (spice level 1)

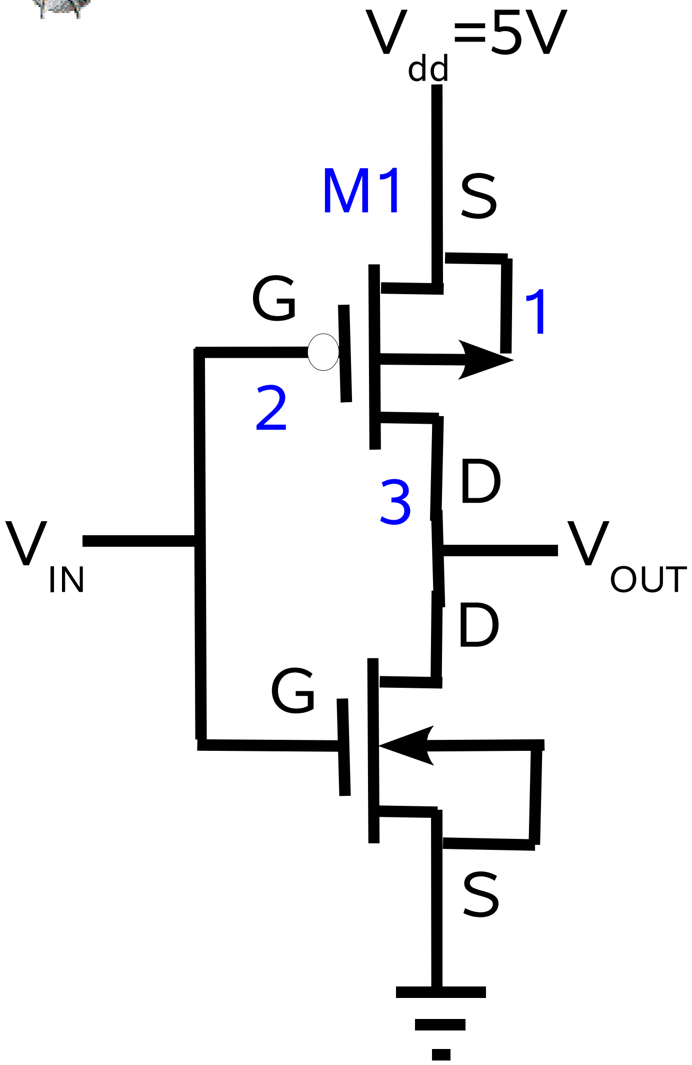
$$
I_D = \begin{cases} 0, & V_{GS} - V_T < 0, & \text{cutoff region} \\ k'(W/L)((V_{GS} - V_T)V_{DS} - \frac{1}{2}V^2_{DS})(1\text{-}\lambda V_{DS}), & 0 < V_{DS} < V_{GS} - V_T, & \text{linear region} \\ \frac{1}{2}k'(W/L)(V_{GS} - V_T)^2(1\text{-}\lambda V_{DS}), & 0 < V_{GS} - V_T < V_{DS}, & \text{saturation region} \end{cases}
$$



Process(N/P)	2 micron	1.2µ	0.8µ
k'	48/16	80/27	110/50
λ	0.032/0.044	0.37/0.61	0.04/0.05
V_{T0}	0.88/-0.85	0.74/-.74	0.7/-.7
γ	0.66/0.69	0.54/.58	0.4/0.57
ψ	0.7/0.7	0.6/0.6	0.7/0.8

```
SPICE:
 .model NCH nmos LEVEL=1 KP=48E-6 LAMBDA=0.032
                   VTO=0.88 GAMMA=0.66 PHI=0.7
 .model PCH pmos LEVEL=1 KP=16E-6 LAMBDA=0.044
                   VTO=-0.85 GAMMA=0.69 PHI=0.7
```

Analog abstraction model: SPICE netlist



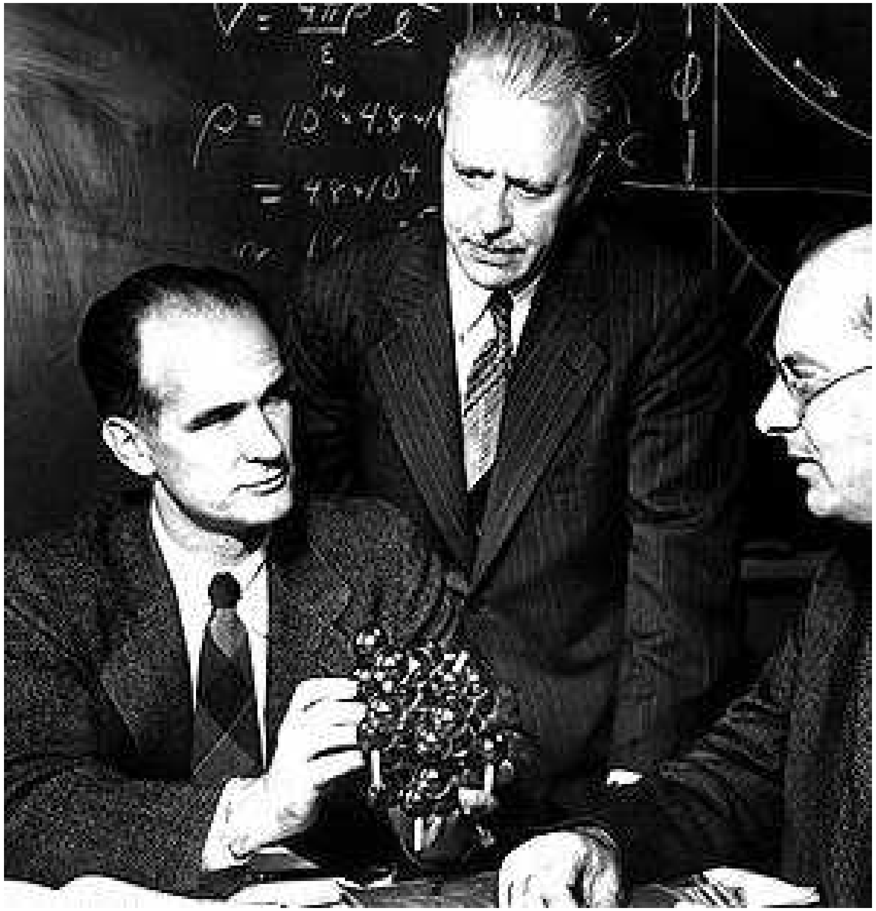
```
*Mname DRAIN GATE SOURCE SUBSTRATE MODEL WIDTH   LENGTH
*      NODE  NODE NODE   NODE      NAME  MICRONS MICRONS
*----- ----- ---- ------ --------- ----  ------- ------
M1     3     2    1      1         PCH   W=3.6U  L=1.2U

M2     3     2    0      0         NCH   W=3.6U  L=1.2U
*
*CAPACITOR
*Cname +NODE -NODE VALUE(Picofarads)
*----- ----- ----- -----------------
C3     3     0     0.1P
*
* INDEPENDANT VOLTAGE SOURCE
*
*Vname +NODE -NODE VALUE
*----- ----- ----- -----
VDD    1     0     DC=5.0


* The following two line are for TRANSIENT analysis
*
*Vname +Node -Node Option T1  V1   T2  V2   T3   V3  T4  V4  T5   V5
*----- ----- ----- ------ --  --   --  --   ---- --  --  --  ---- --
Vin     2     0    PWL(    0   0    4N  0    4.1N 3   8N  3   8.1N 0   )

*     TSTEP TSTOP
*     ----- -----
.TRAN 0.1N  12N
* TEMPERATURE SETTING
.OPTIONS TEMP=25 METHOD=GEAR
*MODEL NAME TYPE
*----- ---- ----
.model NCH nmos LEVEL=1 KP=48E-6 LAMBDA=0.032
                                      VTO=0.88 GAMMA=0.66 PHI=0.7
.model PCH pmos LEVEL=1 KP=16E-6 LAMBDA=0.044
                                      VTO=-0.85 GAMMA=0.69 PHI=0.7

* http://bear.ces.cwru.edu/eecs_cad/tut_spice3_invertor.html
```

Process Technology

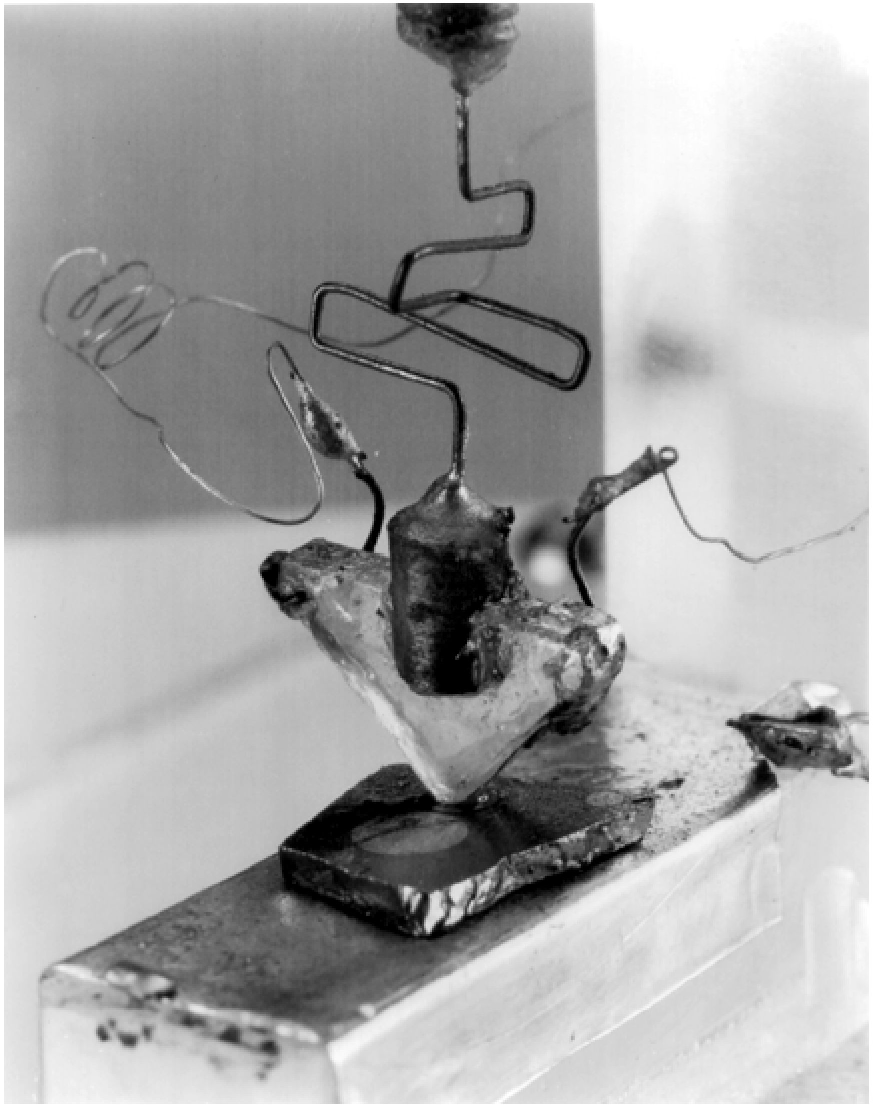
Level	Process	I_D parameters	Year
1	≥ 4μ	8	1968
2	≥ 2μ	23	1980
3	≥ 2μ	21	1980
4	≥ 1μ	67	1985

# Transistors	Technology	Year
1	Bell Labs	1947
10	SSI: Logic, Flip Flops	1961
100-1000	MSI: Adders, counters	1966
1K-20K	LSI: 8-bit uP,ROM,RAM	1971
20K-	VLSI: 16/32-bit uP	1980

First Transistor: 1947

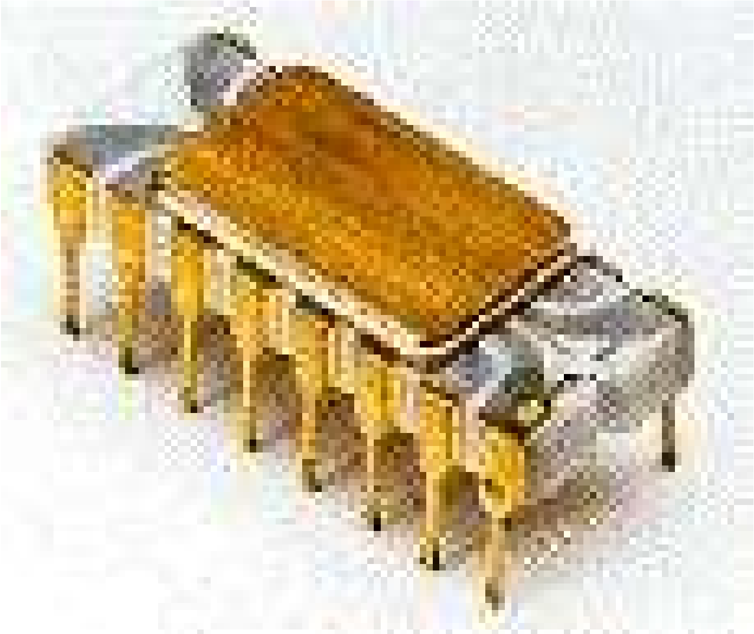


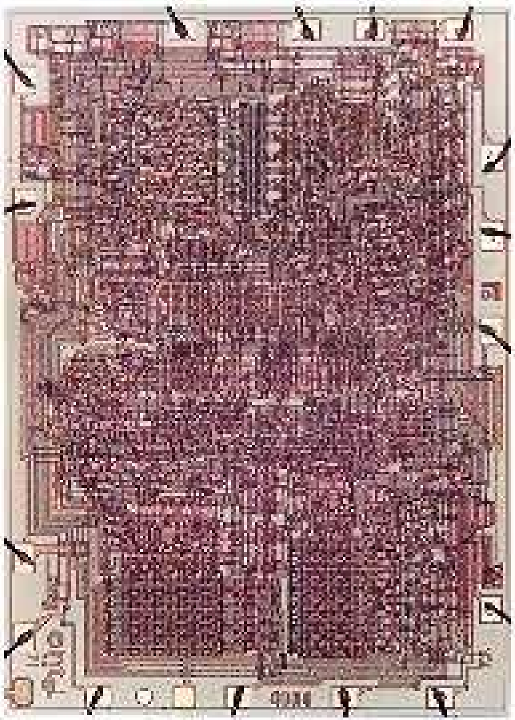
Shockley, Bardeen and Brattain



LSI: Intel Microprocessor History: 4004

- 1971 Intel 4004, 4-bit, 0.74 Mhz, 16 pins, 2250 Transistors



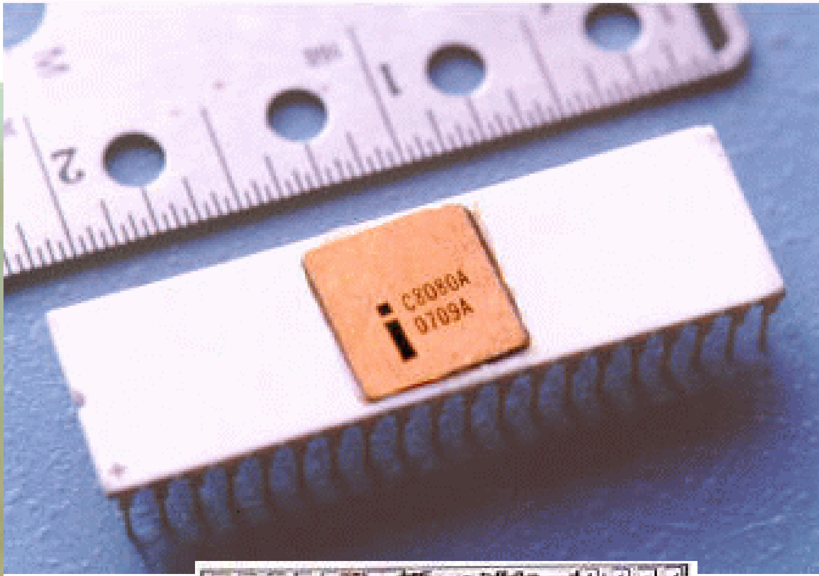


- Intel publicly introduced the world's first single chip microprocessor: U. S. Patent #3,821,715.
- Intel took the integrated circuit one step further, by placing CPU, memory, I/O on a single chip

VLSI: Intel Microprocessor History: 8080

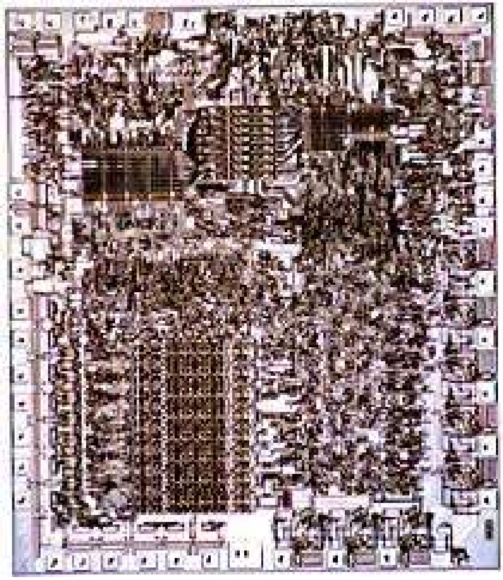
- 1974 Intel 8080, 8-bit, 2 Mhz, 40 pins, 4500 Transistors

Altair 8800 Computer



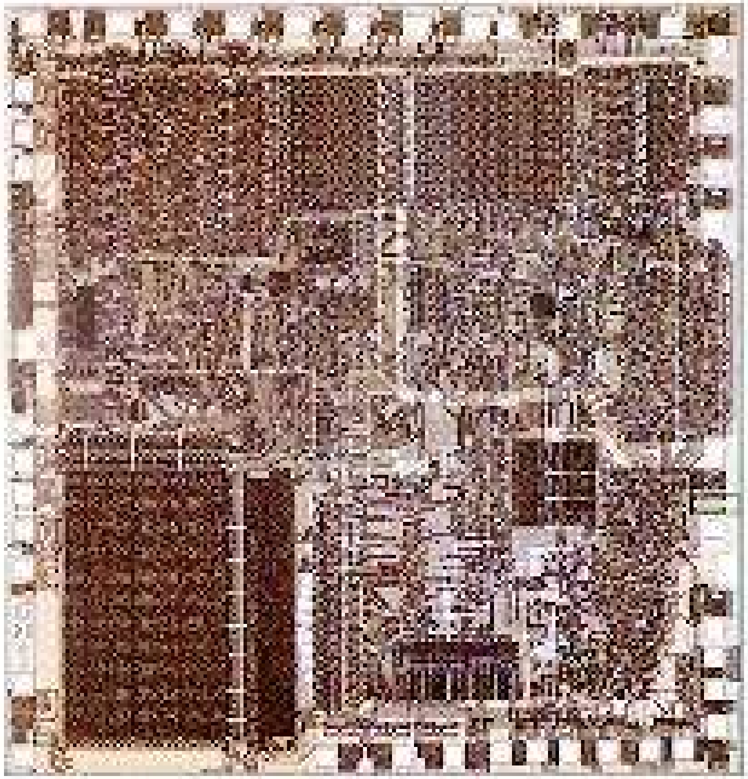
Bill Gates & Paul Allen

write their first Microsoft software product: Basic



VLSI: Intel Microrocessor History: 8088

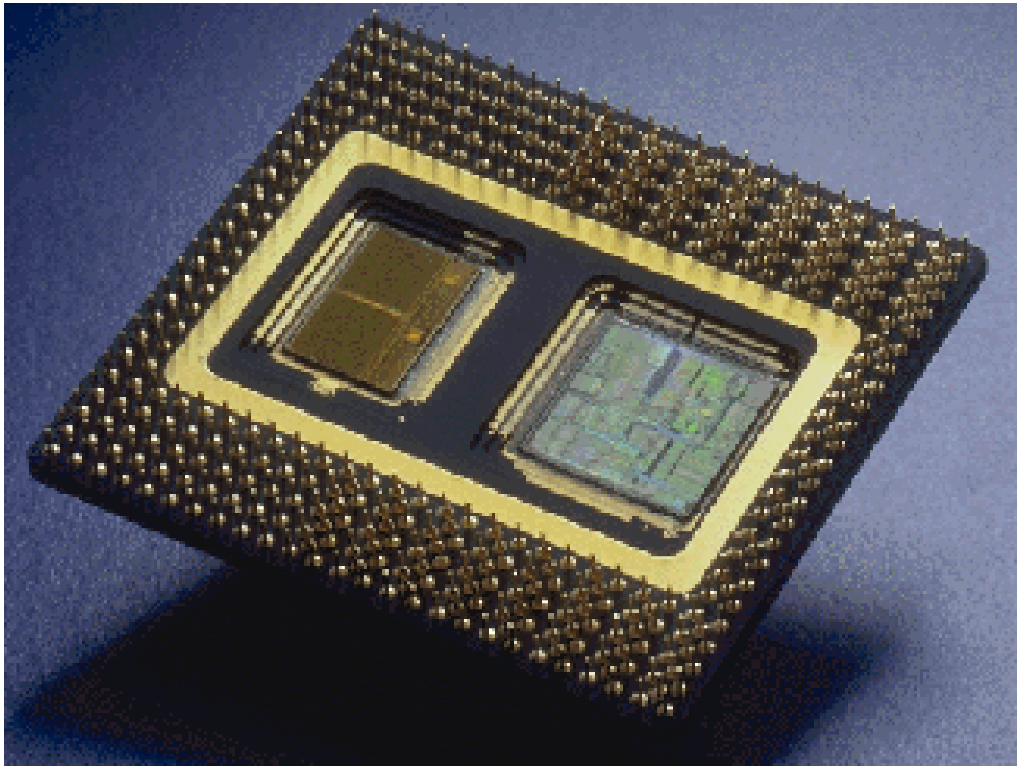
- 1979 Intel 8088, 16-bit internal, 8-bit external, 4.77 Mhz, 40 pins, 29000 Transistors

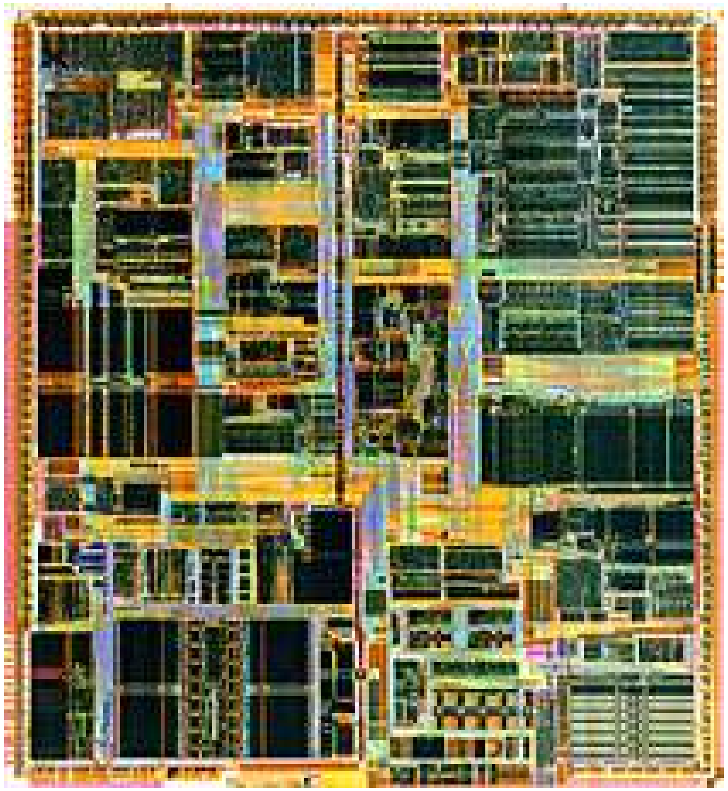


- 0.128M - 0.640M RAM
- 0.360Kb, 5.25” Floppy
- 10M Hard Disk

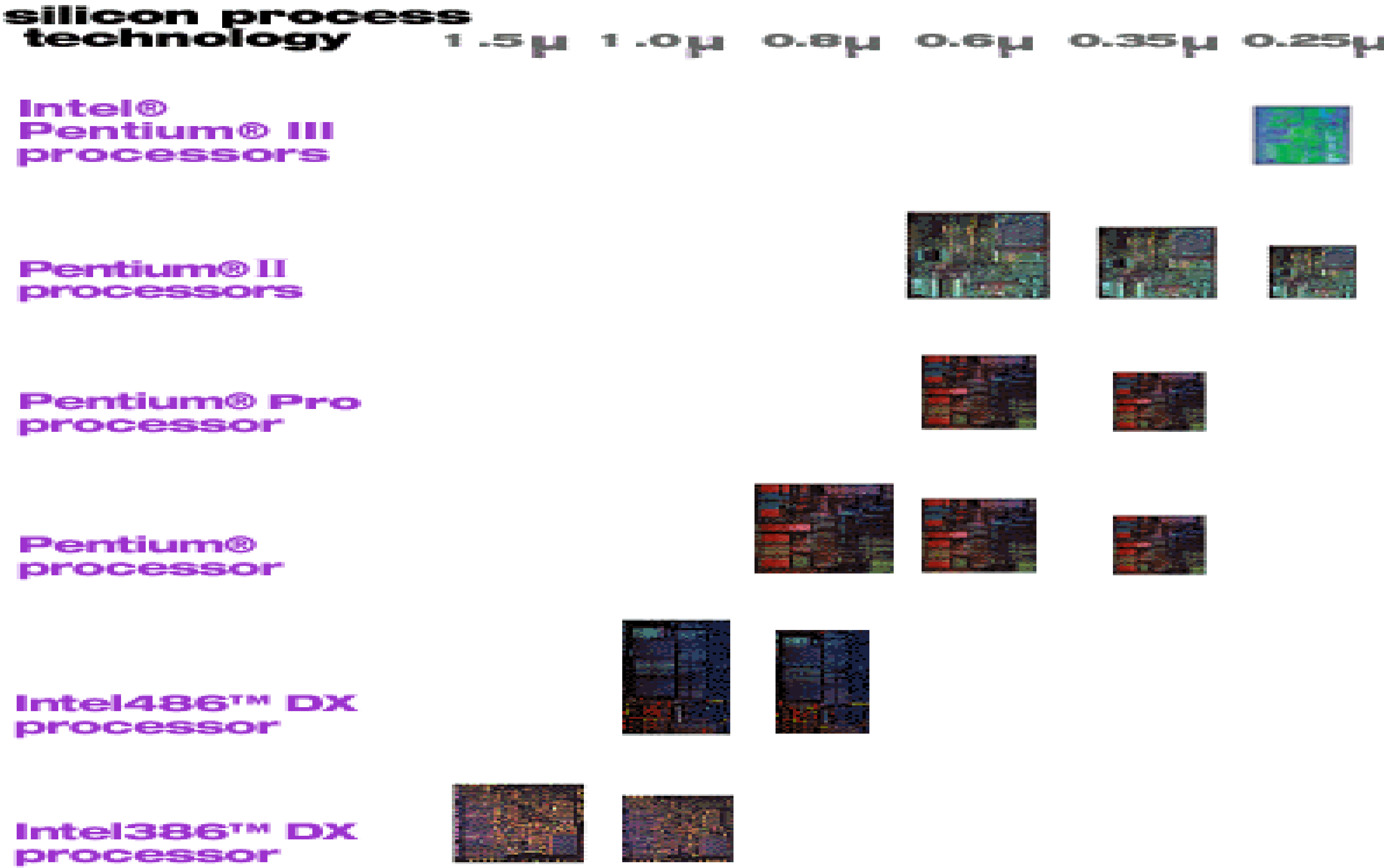
VLSI: Intel Processor History: Penitum Pro

- 1995 Intel Pentium Pro, 32-bit ,200 Mhz internal clock, 66 Mhz external, Superpipelining, 16Kb L1 cache, 256Kb L2 cache, 387 pins, 5.5 Million Transistors





Intel's microprocessor evolution



silicon process technology

1.5µ 1.0µ 0.8µ 0.6µ 0.35µ 0.25µ

Intel® Pentium® III processors

Pentium® II processors

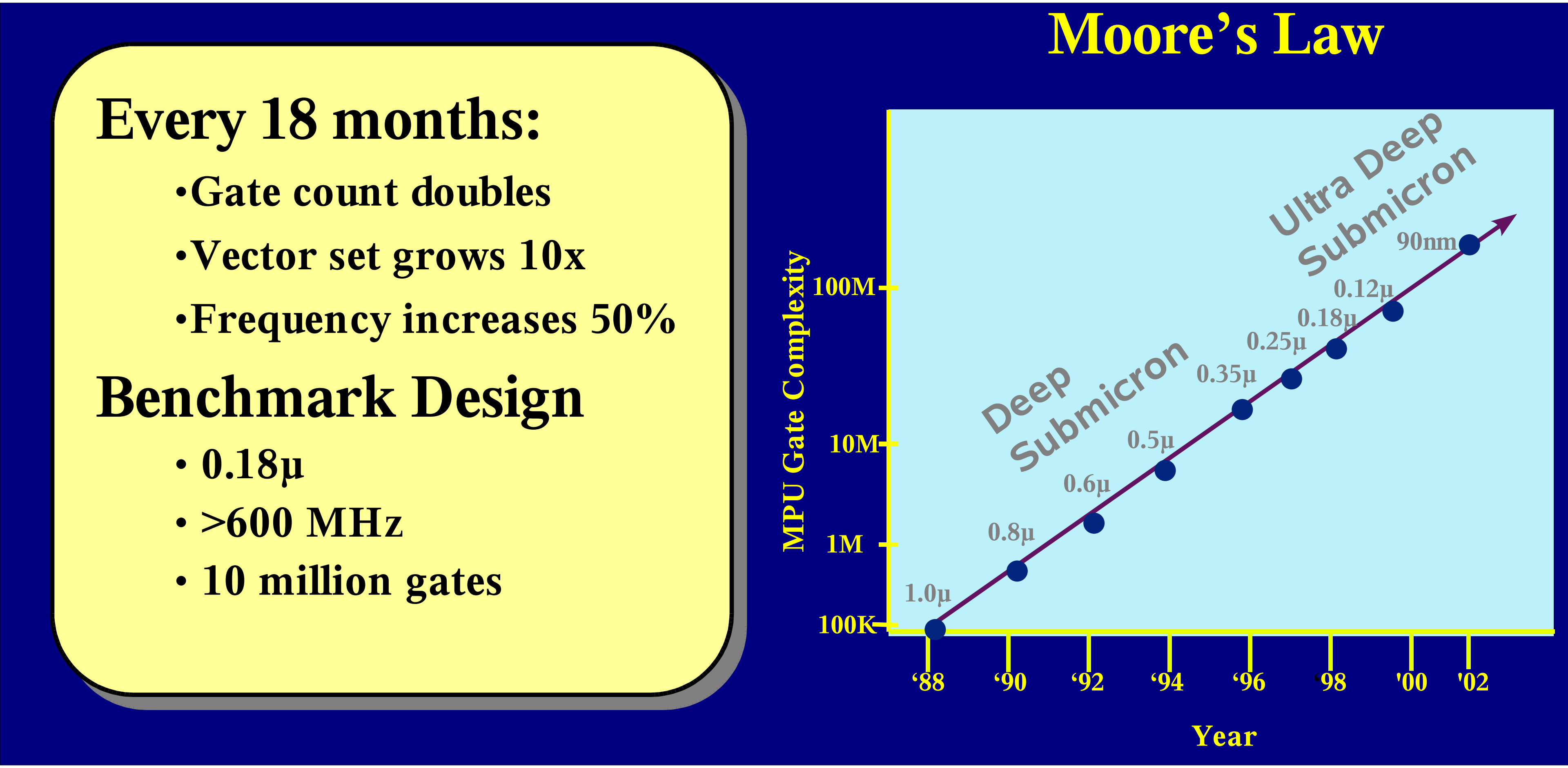
Pentium® Pro processor

Pentium® processor

Intel486™ DX processor

Intel386™ DX processor

Background: Moore’s Law



Moore’s Law

Every 18 months:

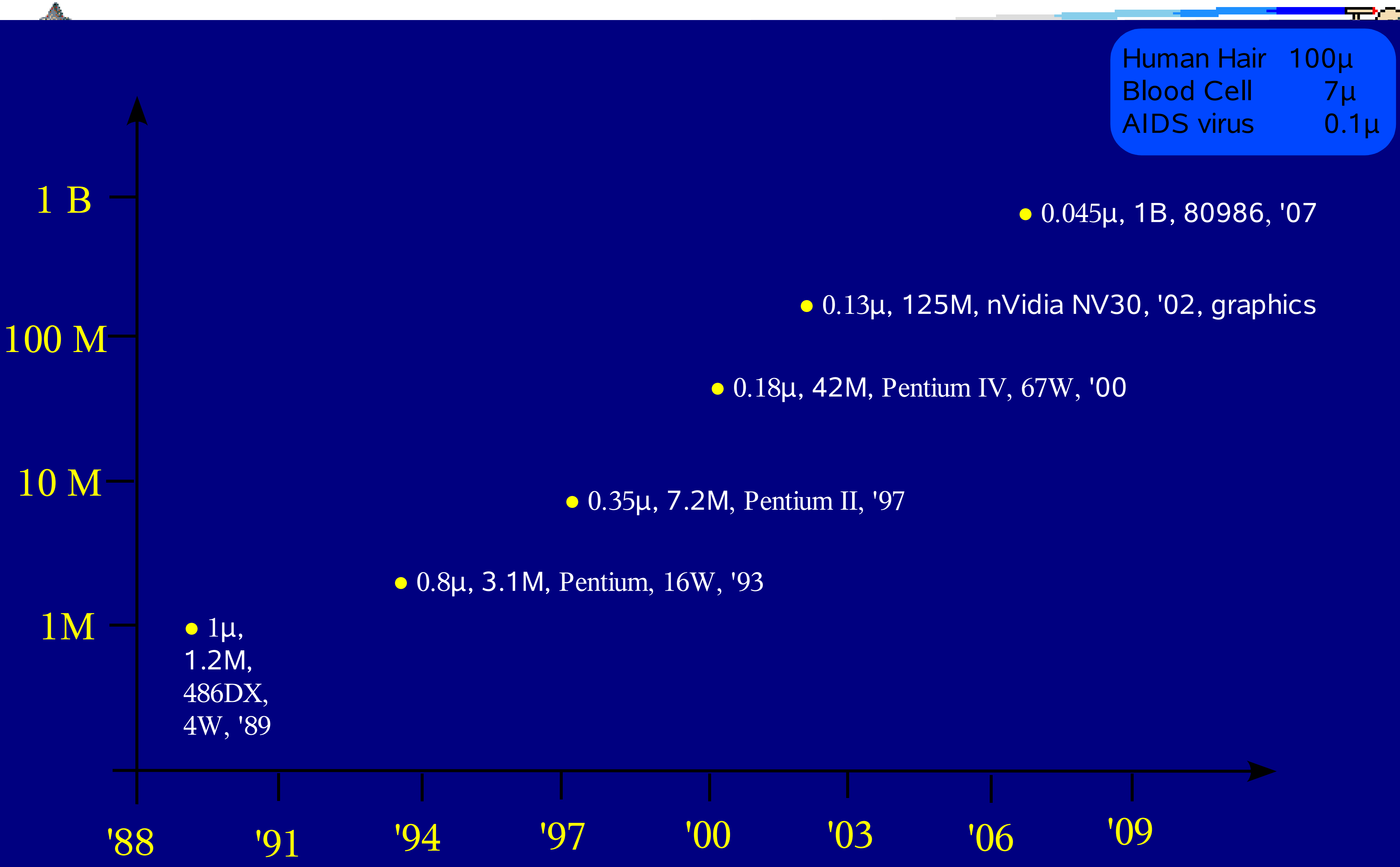
- Gate count doubles
- Vector set grows 10x
- Frequency increases 50%

Benchmark Design

- 0.18µ
- >600 MHz
- 10 million gates

- SoCs by the year 2007, predicts that the state of the art ICs will exceed 1 billion transistors.

Moore's law in perspective



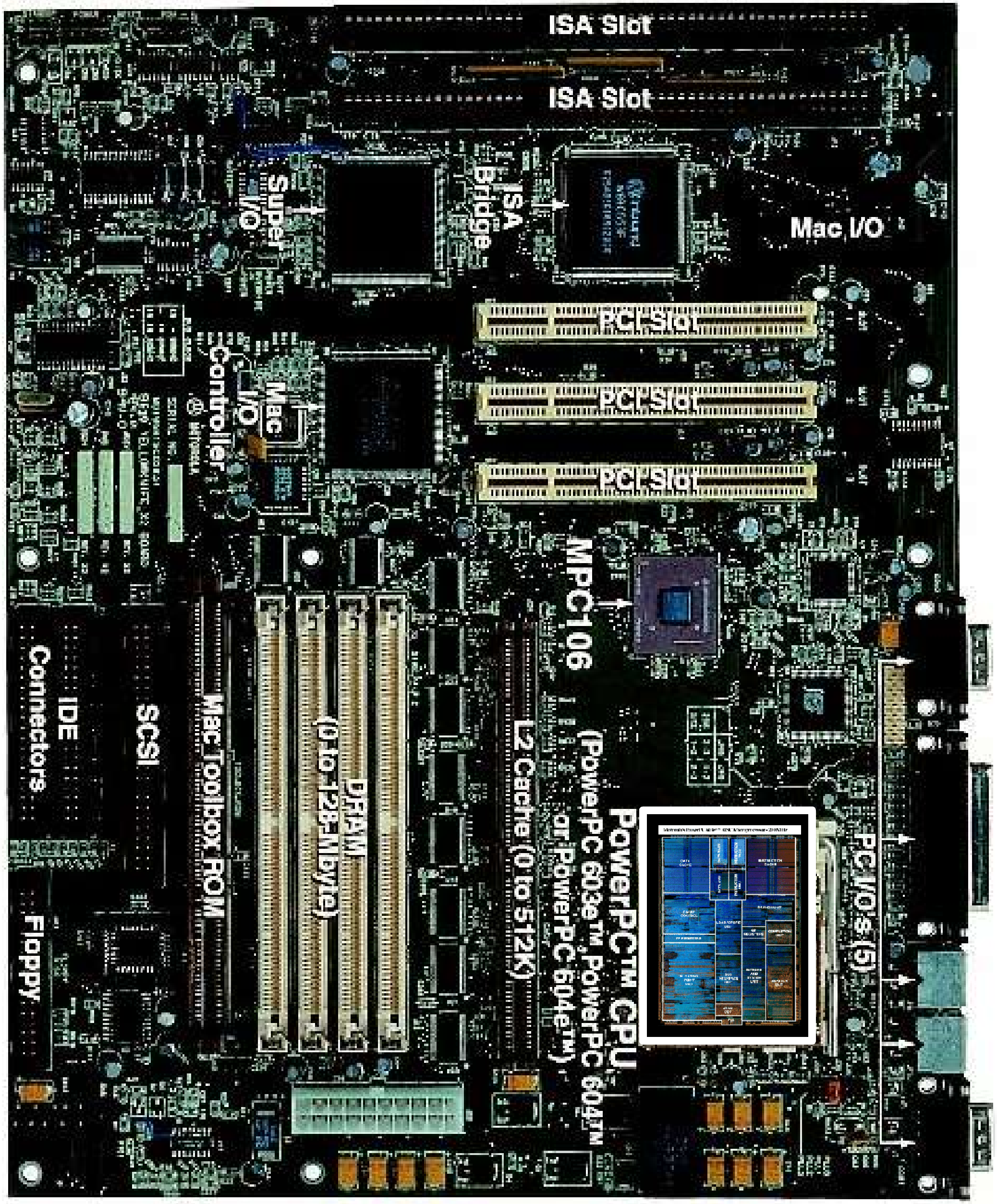
Human Hair	100µ
Blood Cell	7µ
AIDS virus	0.1µ

4M 16M 64M 256M → DRAM capacities (manufacured by year)

Conventional Systems

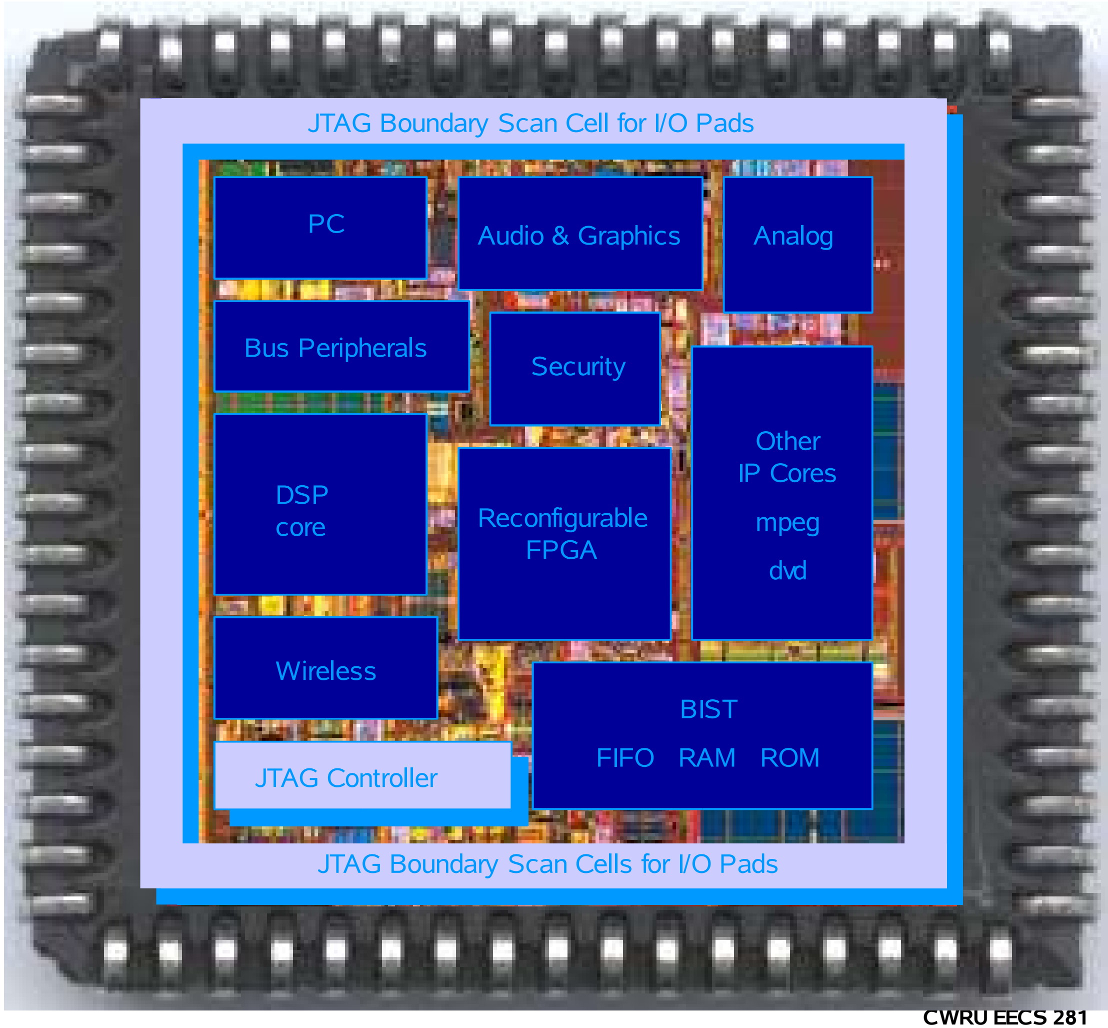
- *Systems* are traditionally composed of many separate chips: microprocessor, RAM, audio chips, …

- New tech trends favor integration of all these components as **embedded cores** on a single Sytem-on-Chip (SoC).

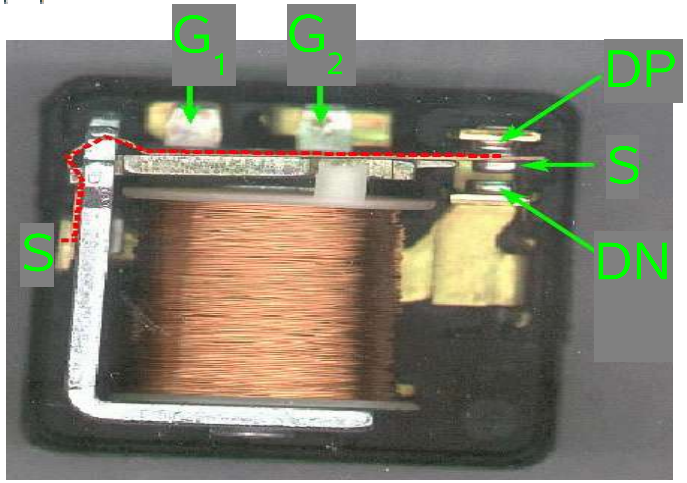


SoC: System on a Chip

- Modern VLSI technology enables the integration of a multitude of predefined core modules into a Systems-on-a-Chip (SoC).
- SoC is an enabling technology for embedded systems.
- Embedded systems handle and manipulate large volumes of data in real-time.
- Some Examples: Internet Appliances, PDAs, cell phones, MP3 players, …



Digital abstraction model: Relays

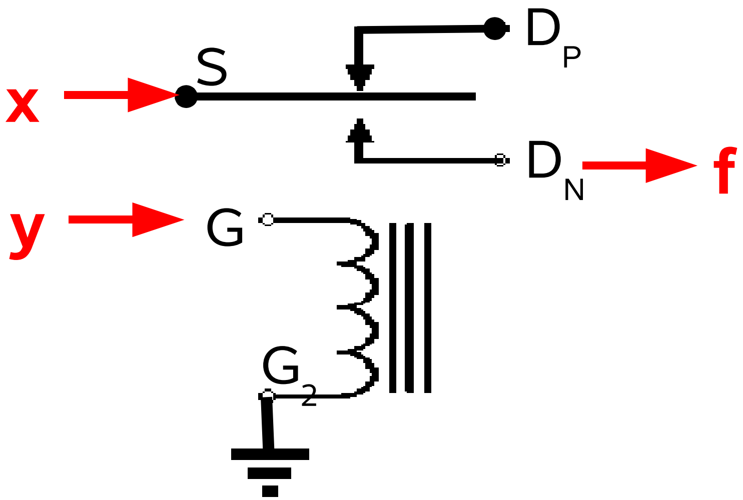


Relay function table

G	DP	DN
0	S	Z
1	Z	S

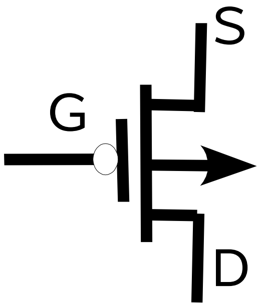
Z = No connection = tristate

What is the function f(x,y) ?

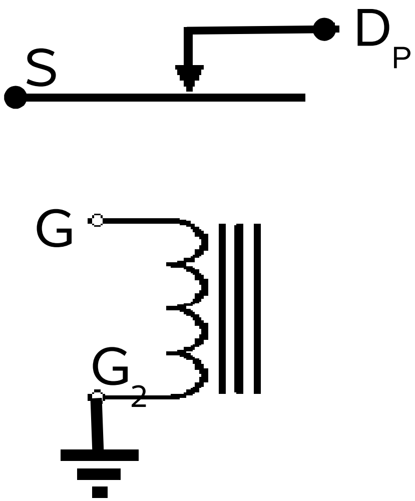


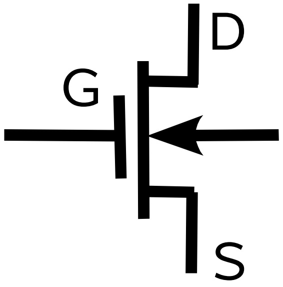
x	y	f
0	0	
0	1	
1	0	
1	1	

Digital abstraction model: Relays

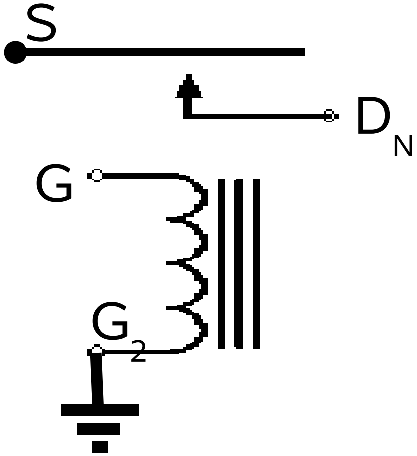


$$I_D = \begin{cases} 1, & V_{GS} = 0, \quad \text{cutoff} \\ 0, & V_{GS} = 1, \quad \text{saturation} \end{cases}$$

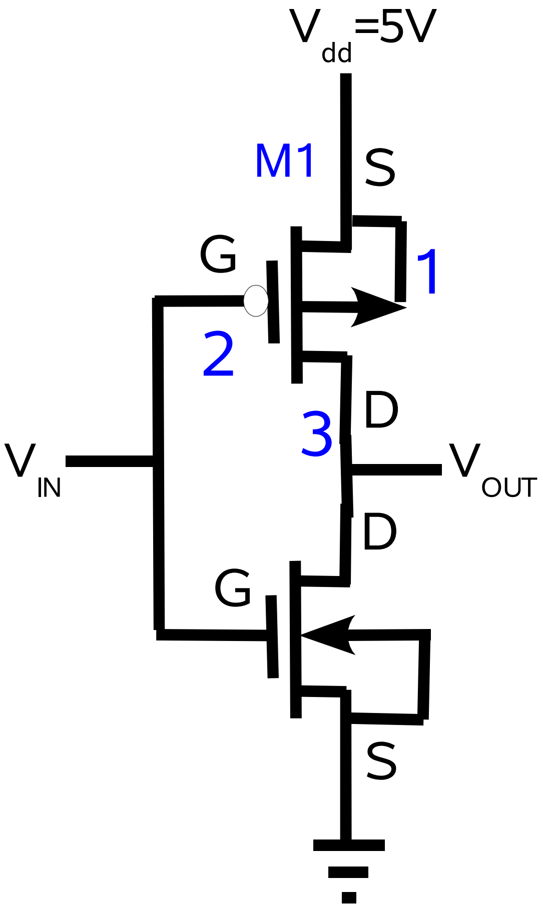


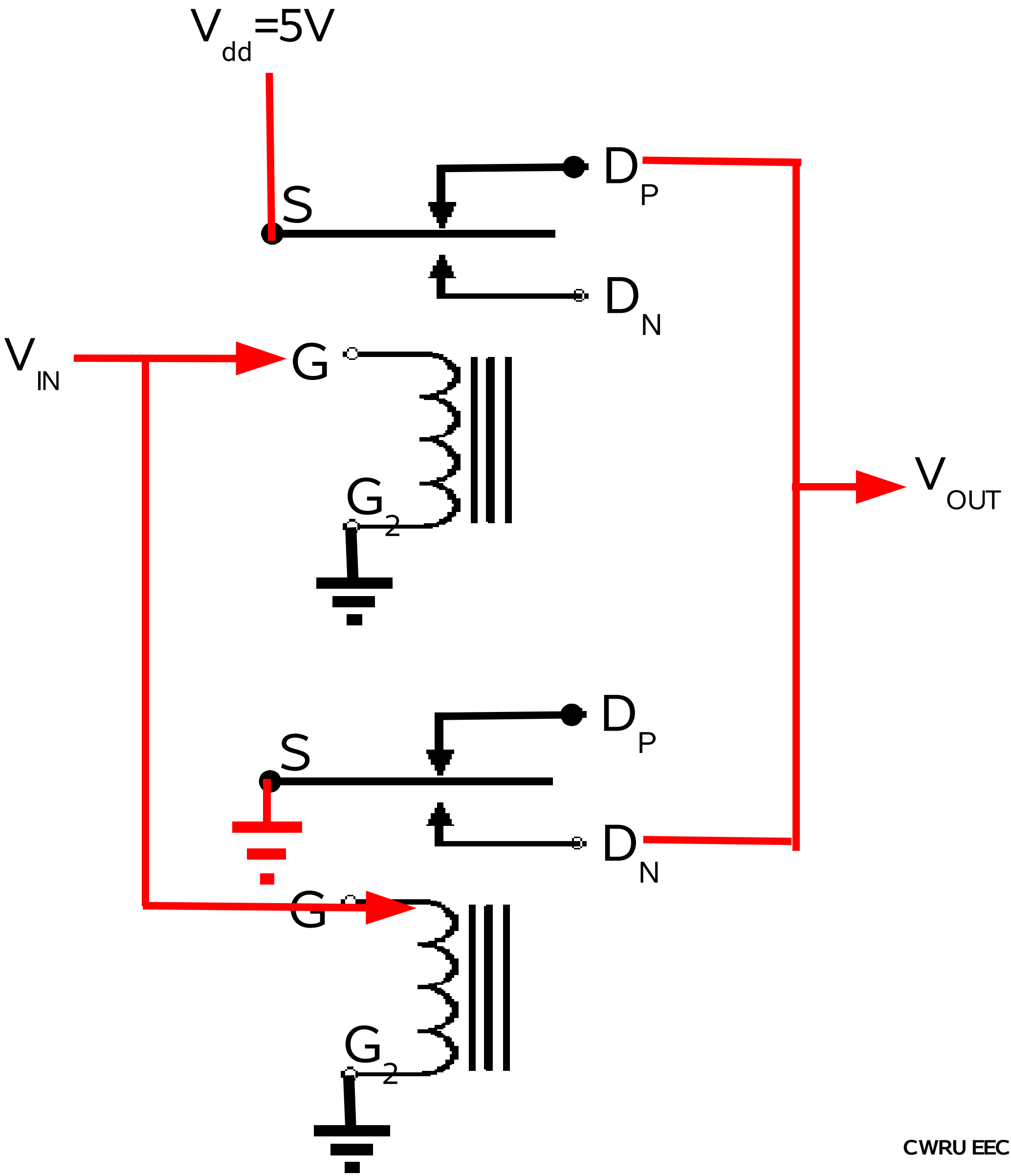


$$I_D = \begin{cases} 0, & V_{GS} = 0, \quad \text{cutoff} \\ 1, & V_{GS} = 1, \quad \text{saturation} \end{cases}$$

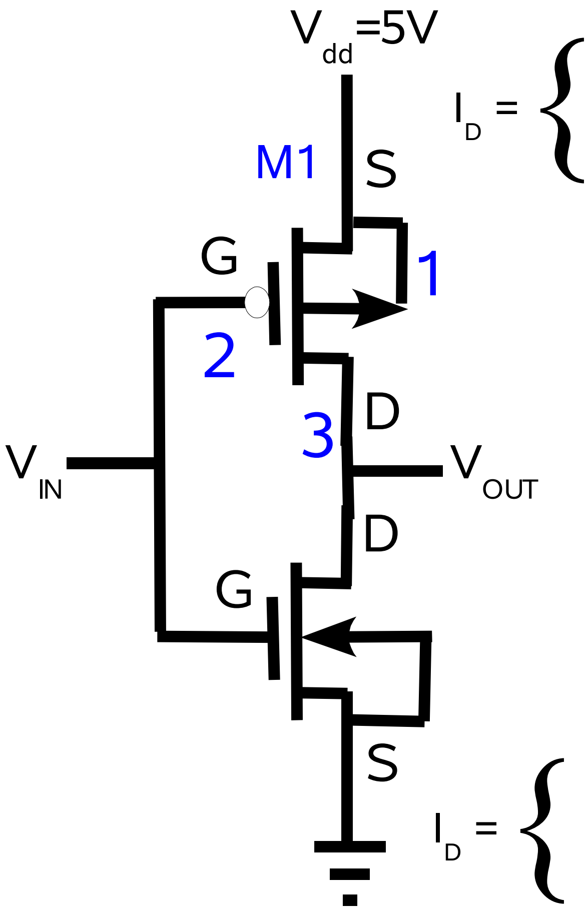


Digital abstraction model: Relays



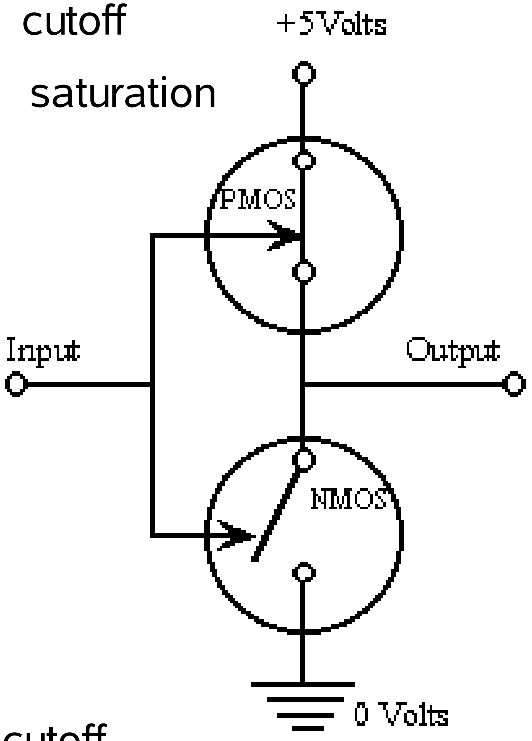


Digital abstraction model (VHDL, Verilog)



$$I_D = \begin{cases} 1, & V_{GS} = 0, \quad \text{cutoff} \\ 0, & V_{GS} = 1, \quad \text{saturation} \end{cases}$$

$$I_D = \begin{cases} 0, & V_{GS} = 0, \quad \text{cutoff} \\ 1, & V_{GS} = 1, \quad \text{saturation} \end{cases}$$

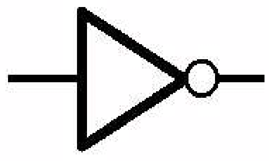


TRUTH TABLE

V_{IN}	V_{OUT}
0	1
1	0

C++

```
VOUT = ~VIN;
```



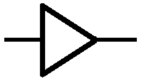
VHDL

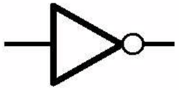
```
VOUT <= NOT(VIN);
```

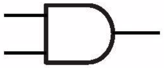
Logic: Symbolic notation and definitions

1 True: T.

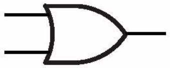
0 False: F.


Assertion, buffer, p: p is true.

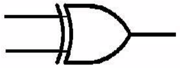

Negation: ~p, NOT(p): p is false.


Conjunction, p /\ q, p AND q, p & q:

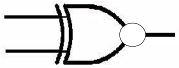
both p and q are true.


Disjunction, p \/ q, p OR q, p | q:

either p is true, or q is true, or both.


Exclusive Or, p$\oplus$q, p XOR q, p^q:

either p is true or q is true, but not both.

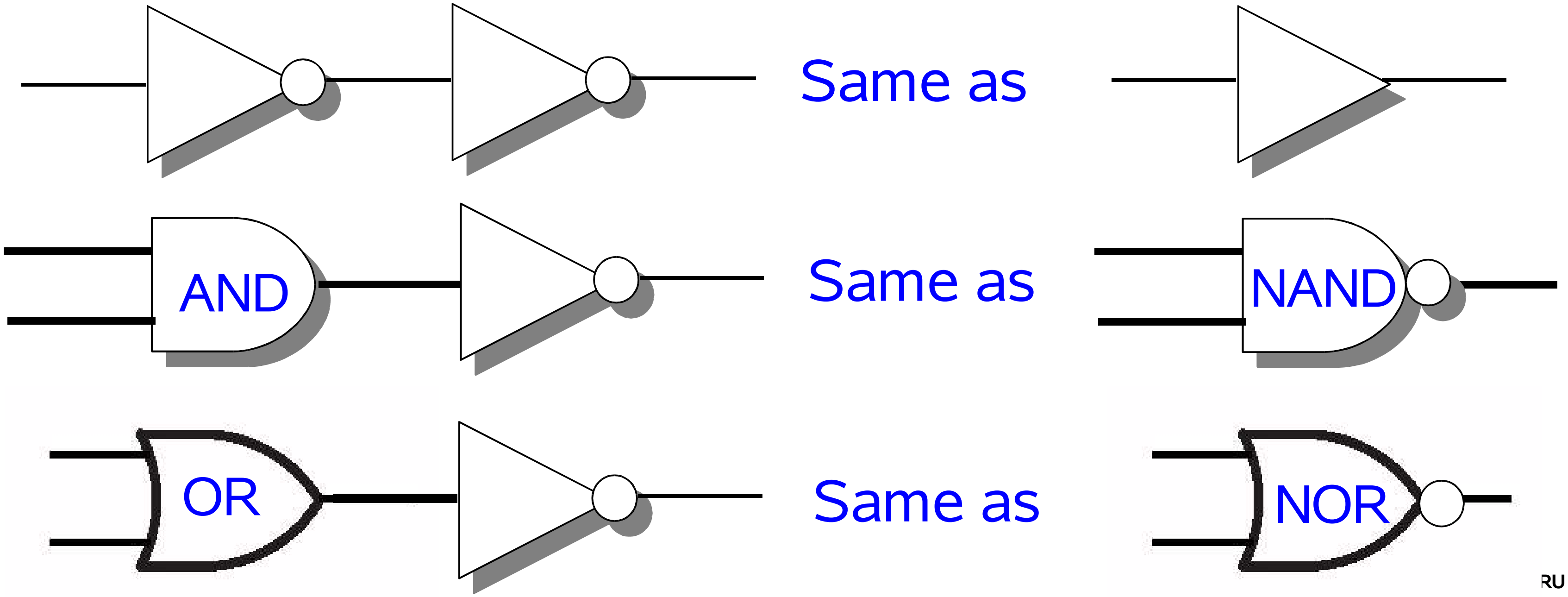

Equivalence, p$\Leftrightarrow$q, p XNOR q, ~(p ^ q):

p and q are either both true or both false .

Implication, p$\Rightarrow$q: if p is true, then q is true.

Logic: Truth Tables

p	q	p/\q	p \/ q	p⇒q	~p	~q	p^q
F	F	F	F	T	T	T	F
F	T	F	T	T	T	F	T
T	F	F	T	F	F	T	T
T	T	T	T	T	F	F	F

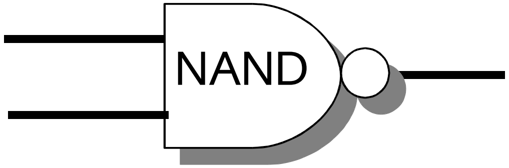


Same as

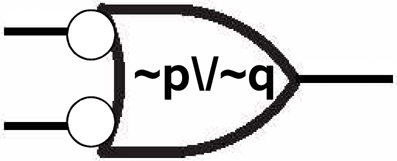
Same as

Same as

Logic: DeMorgan's Theroem



Same as

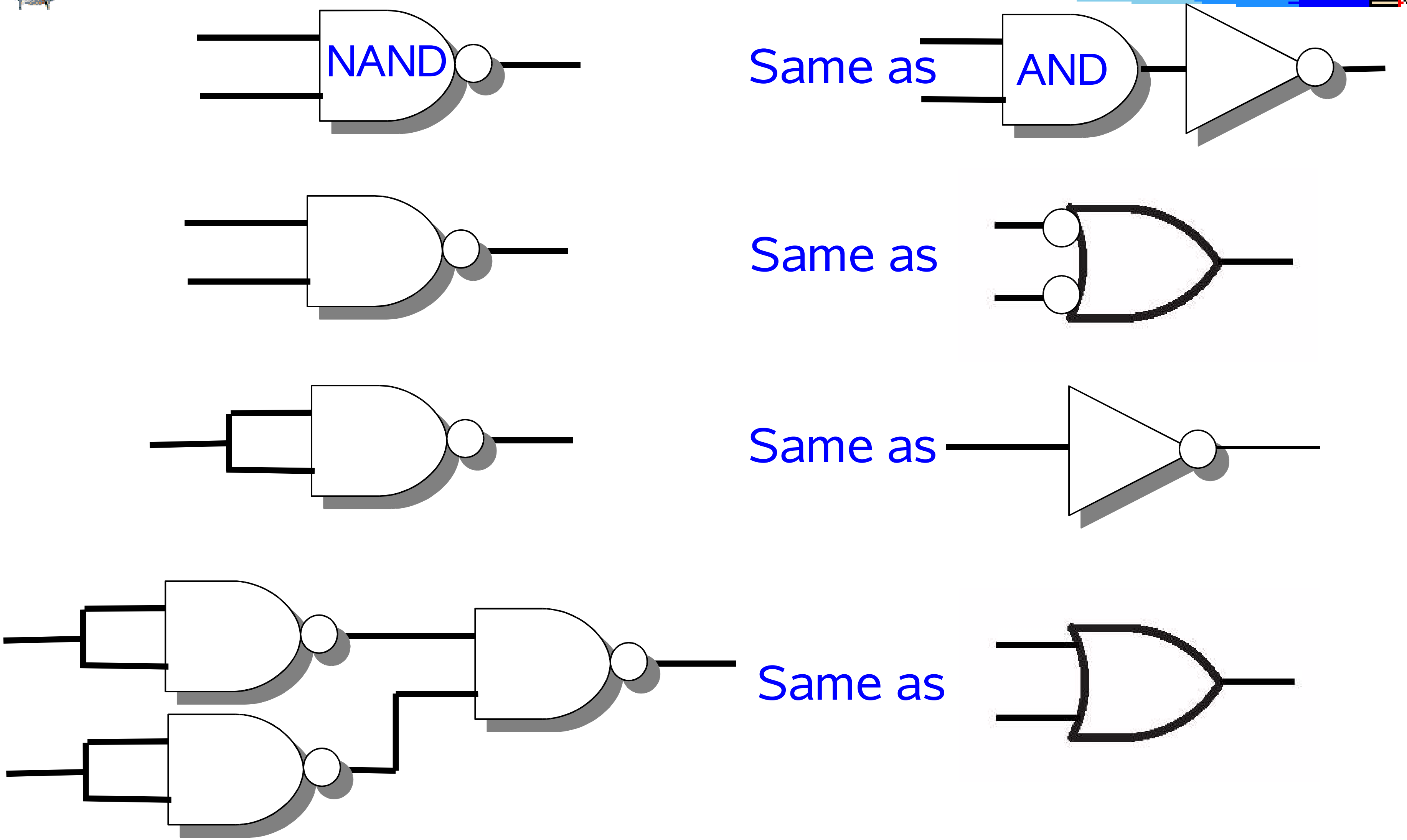


Rule of Thumb:

(1) Complement each “dot” (2) Flip “AND” to “OR”

p	q	NAND	~p	~q	~p\/~q
F	F	T	T	T	T
F	T	T	T	F	T
T	F	T	F	T	T
T	T	F	F	F	F

Logic: NANDs: equivalent forms



NAND

Same as

AND

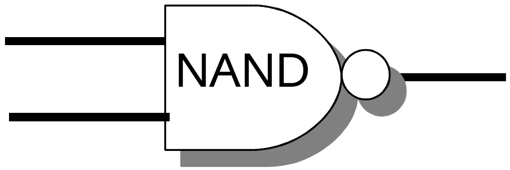
Same as

Same as

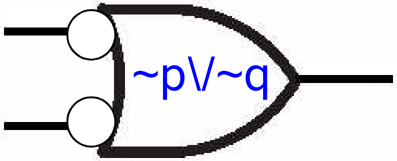
Same as

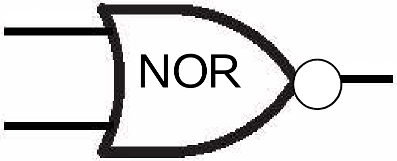
Can every logic function be defined just by using “only” NAND gates?

Logic: DeMorgan's Theroem

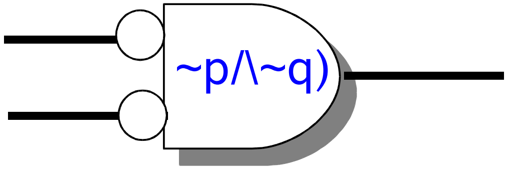


Same as





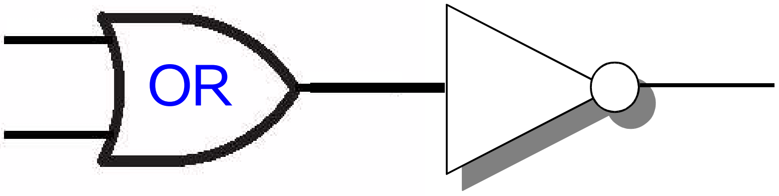
Same as



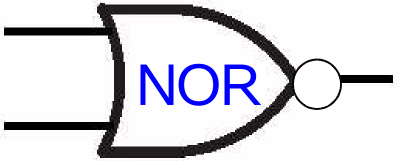
Rule of Thumb:

(1) Complement each “dot” (2) Flip “AND” to “OR”

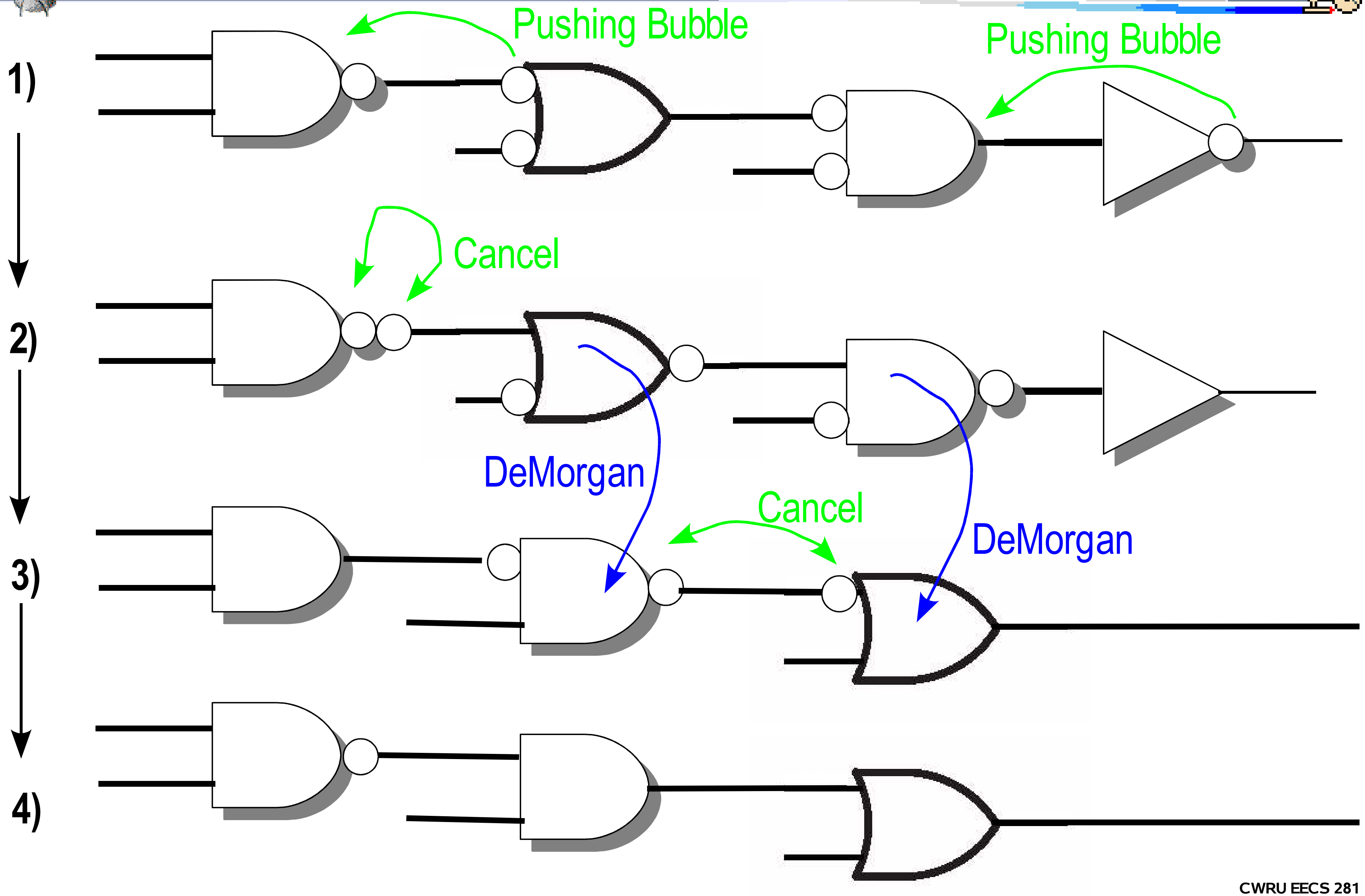
Can every logic function be defined just by using “only” NOR gates?



Same as

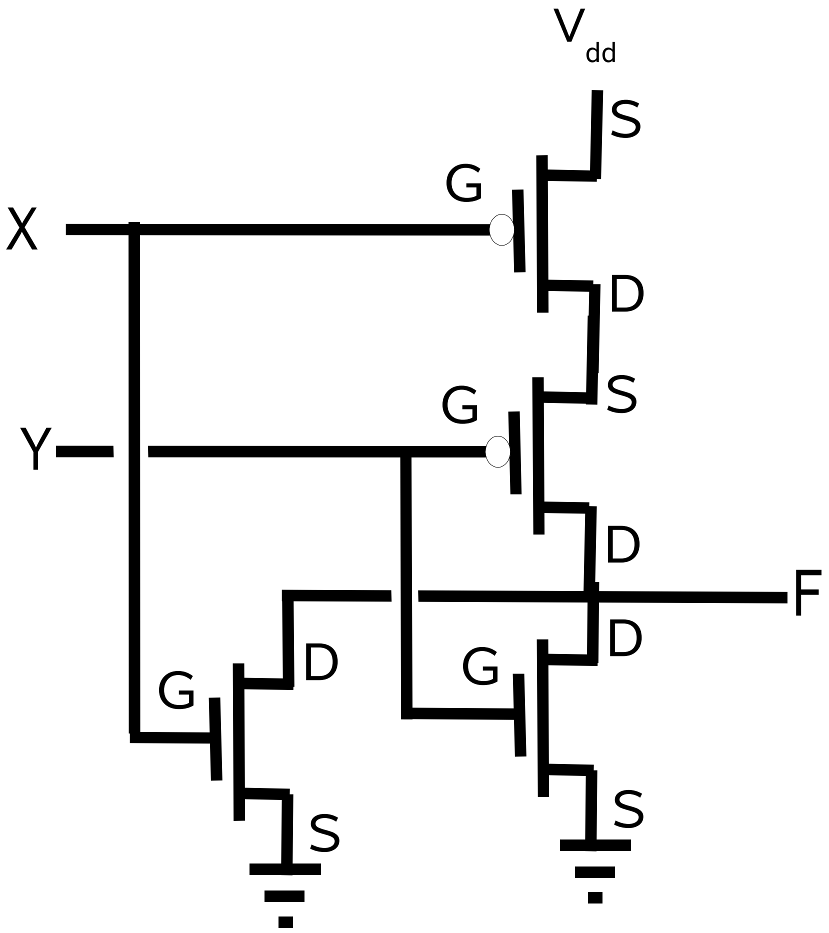


Logic: Bubble pushing



Logic: CMOS “switch model” example

1. Given CMOS circuit:



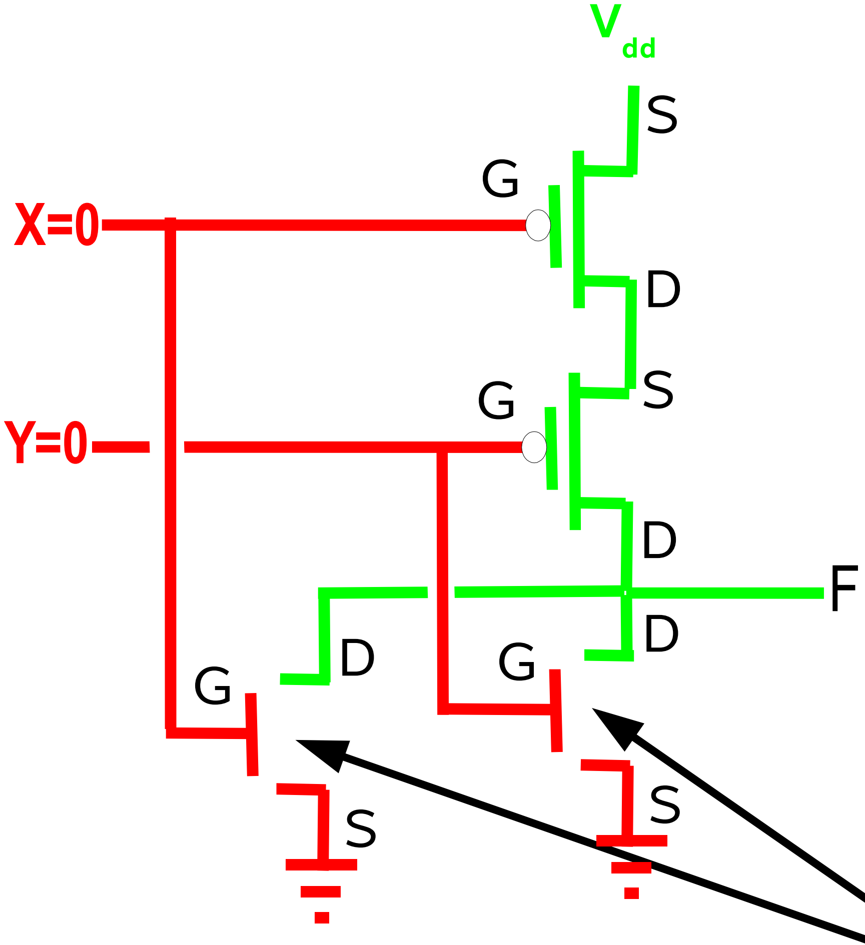
2. Build truth table:

x	y	f
F	F	
F	T	
T	F	
T	T	

3. Deduce Logic gate(s):

Logic: CMOS example: X<=0; Y<=0;

1. Given CMOS circuit:



1. Build truth table:

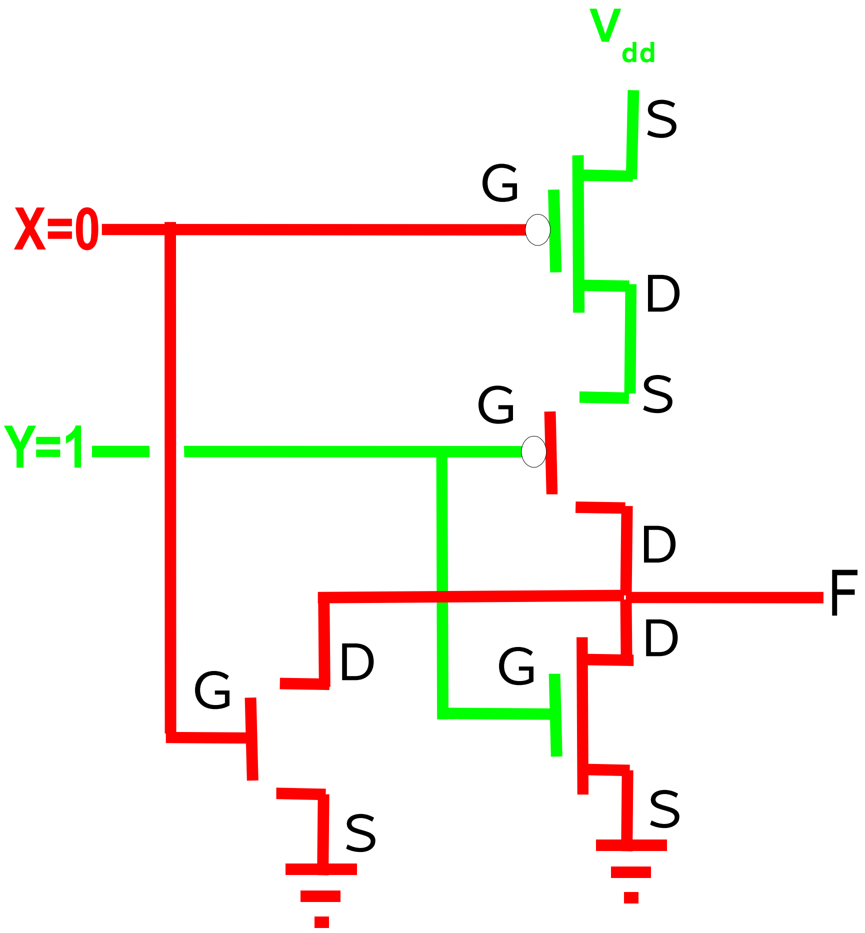
X	Y	F
0	0	1

3. Determine logic type:

Treat CMOS as ON-OFF switches:
Turn these off

Logic: CMOS example: X<=0; Y<=1;

1. Given CMOS circuit:



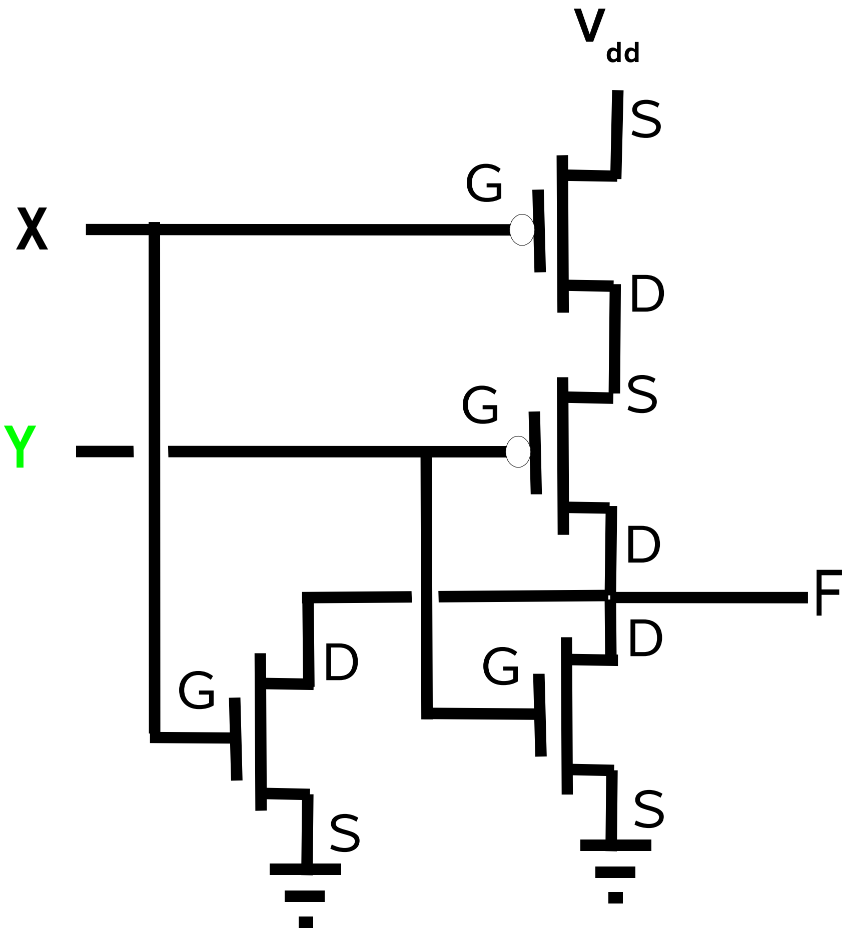
2. Build truth table:

X	Y	F
0	0	1
0	1	0

3. Determine logic type:

Logic: CMOS example: X<=0; Y<=1;

1. Given CMOS circuit:

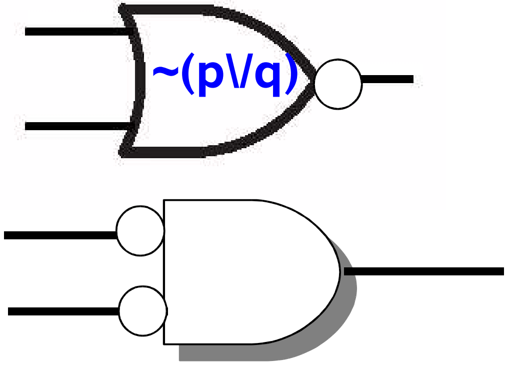


2. Build truth table:

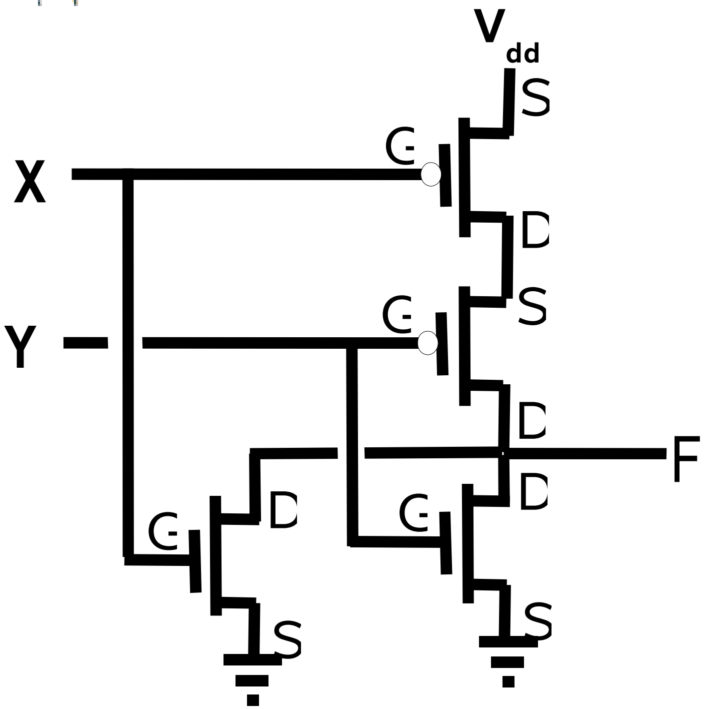
X	Y	F
0	0	1
0	1	0
1	0	0
1	1	0

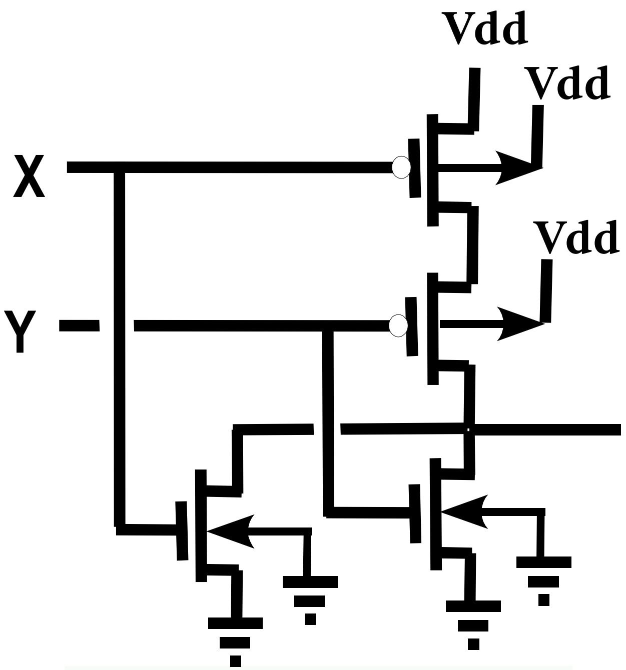
3. Determine logic type:

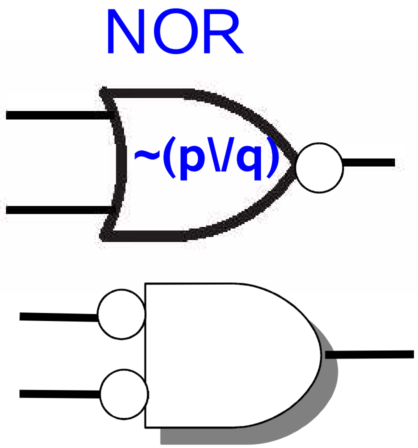
NOR

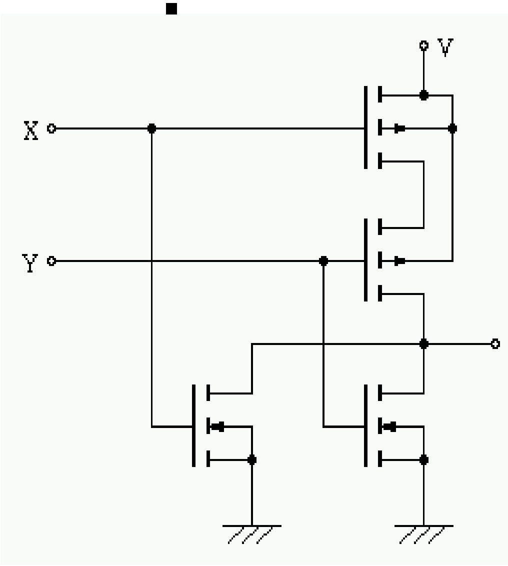


Logic: NOR equivalent circuit

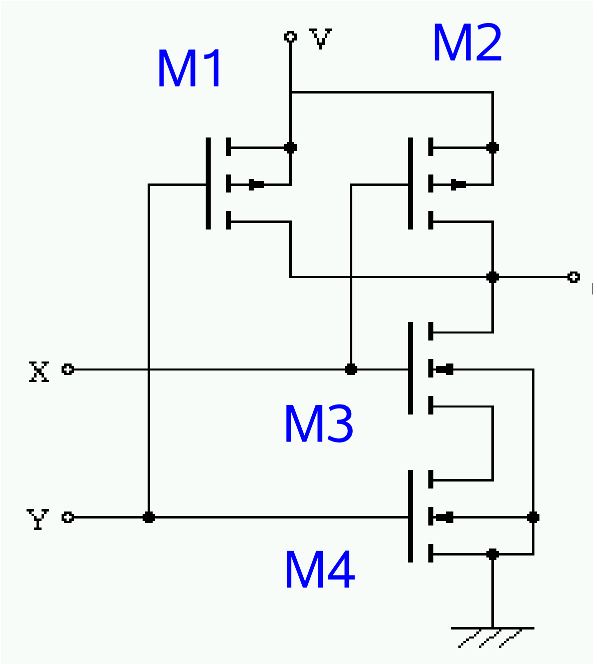




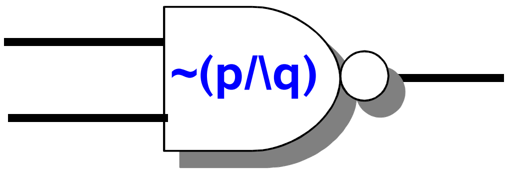


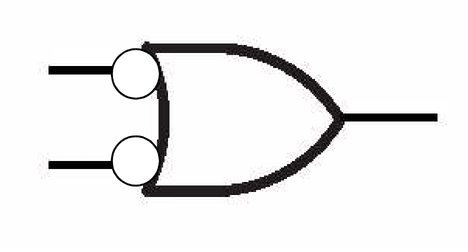


Logic: NAND sub-circuit



NAND

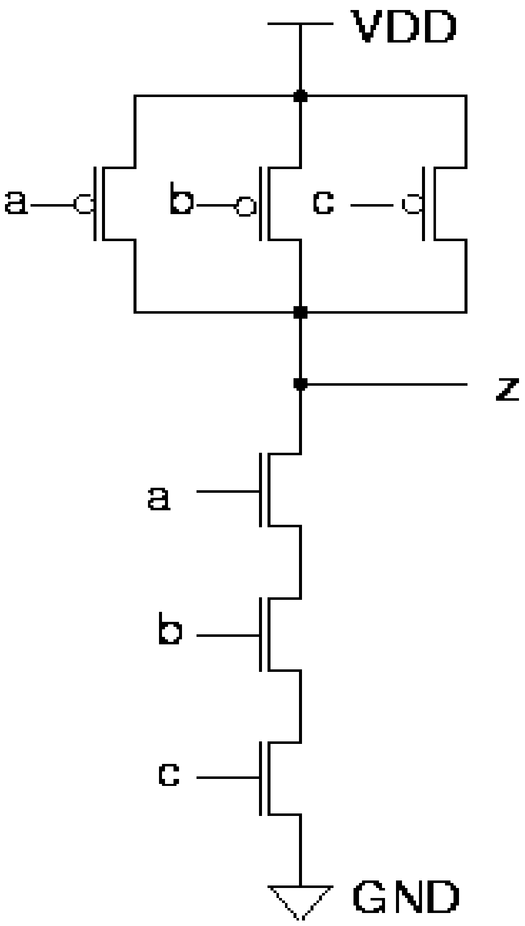




```
.SUBCKT NAND1X 1 2 3 4
* NAND(X:1, Y:2) => F:3, Vdd:4;
*
*Mname DRAIN GATE SOURCE SUBSTRATE MODEL WIDTH   LENGTH
*      NODE  NODE NODE   NODE      NAME  MICRONS MICRONS
*----- ----- ---- ------ --------- ----  ------- ------
M1     3     1    4      4         PCH   W=9U    L=2U
M2     3     2    4      4         PCH   W=9U    L=2U
M3     3     1    5      0         NCH   W=3U    L=2U
M4     5     2    0      0         NCH   W=3U    L=2U
.ENDS NAND1X



*MODEL NAME TYPE; 2 Micron process technology
*----- ---- ----
.model NCH nmos LEVEL=1 KP=48E-6 LAMBDA=0.032
         VTO=0.88 GAMMA=0.66 PHI=0.7
.model PCH pmos LEVEL=1 KP=16E-6 LAMBDA=0.044
         VTO=-0.85 GAMMA=0.69 PHI=0.7
```

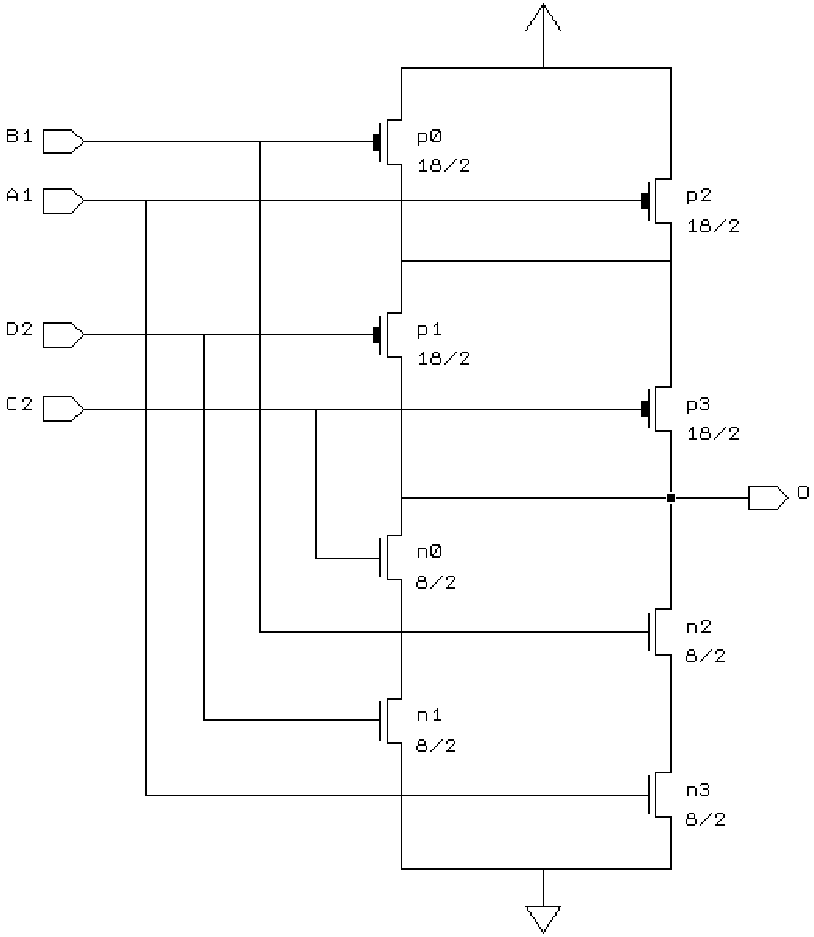
Logic: NOR

Gate?



Logic: Example #1

Convert the following schematic to (a) SPICE, (b) truth table, (c) logic gates and (b) logic expression.



Logic: Example #2

Convert the following equation: s=(~a&b)|(a&~b)
to (a) logic gates and (b) NAND only;

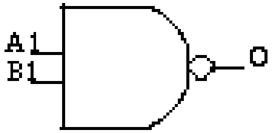
Modelling types

- Behavioral model
  - Explicit definition of mathematical relationship between input and output
  - No implementation information
  - It can exist at multiple levels of abstraction
    - Dataflow, procedural, state machines, …

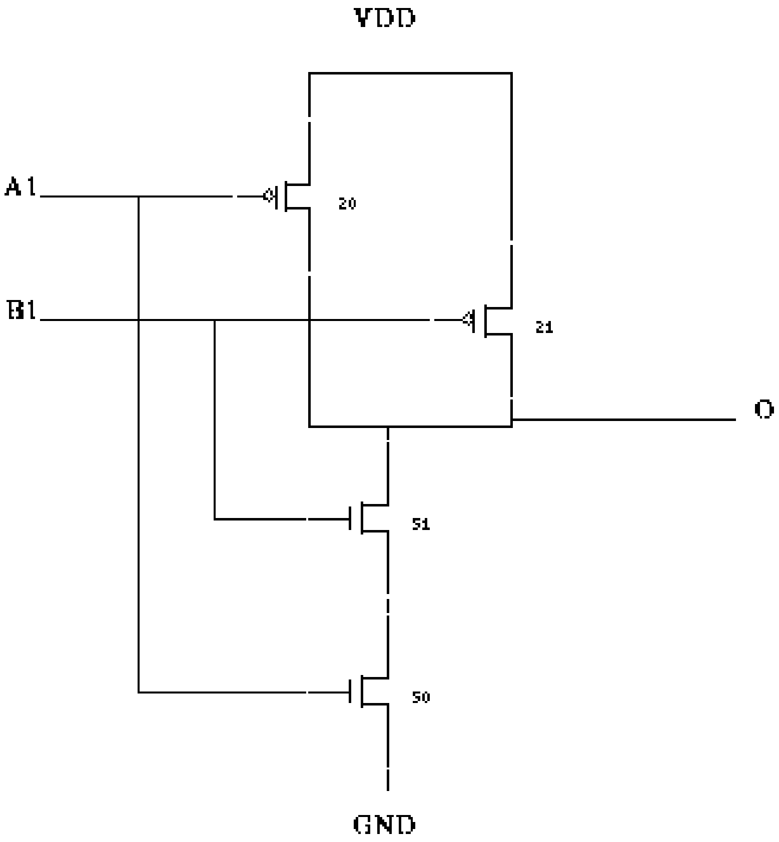
- Structural model
  - A representation of a system in terms of interconnections (netlist) of a set of defined component
  - Components can be described structurally or behaviorally

Nand gate: behaviorial, transistor, layout

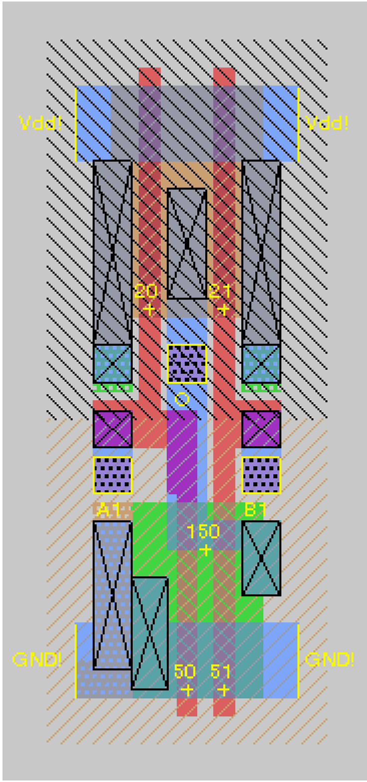
```
O <= NOT ( A1 AND B1);
```



Boolean Equation



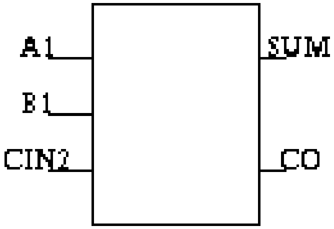
Transistor



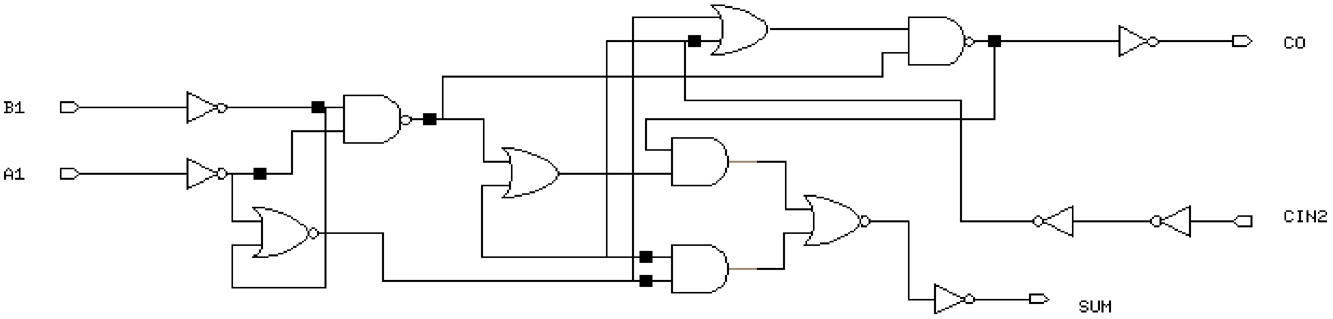
Mask

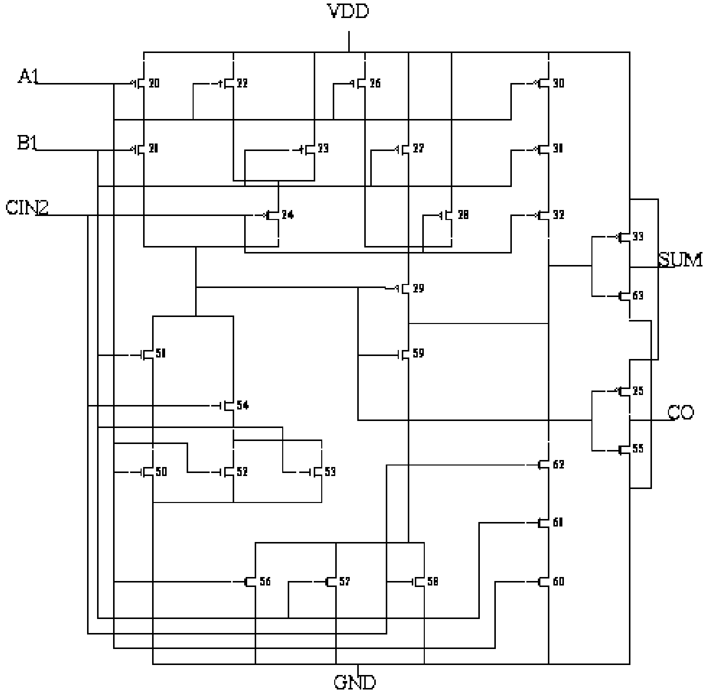
Adder: behavior, netlist, transistor, layout

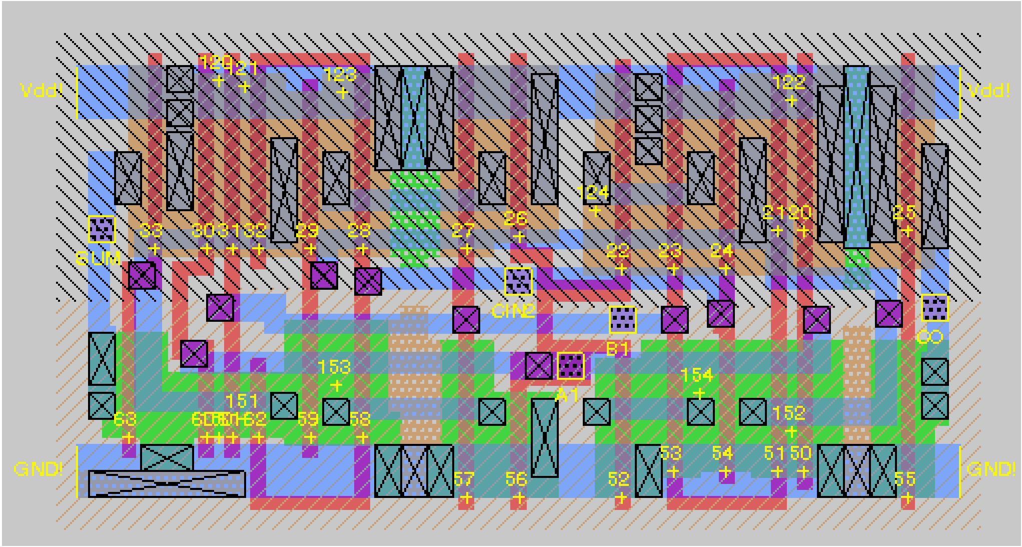
Behavioral model



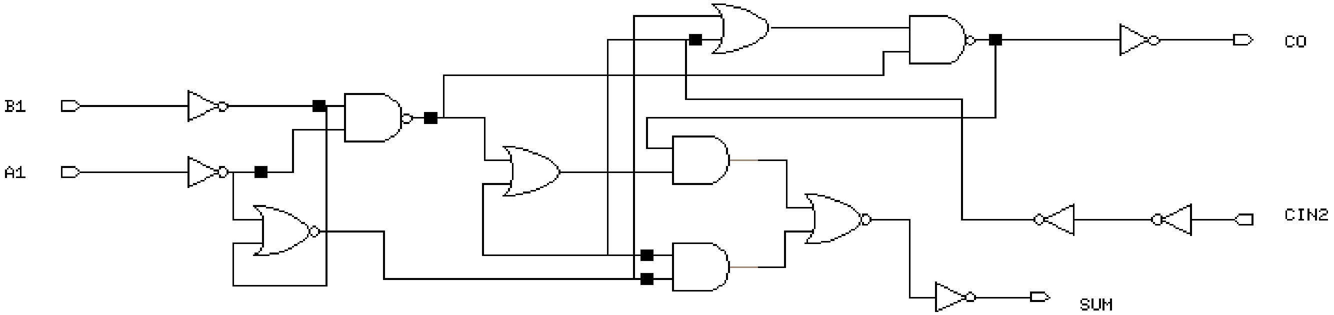
Structural model

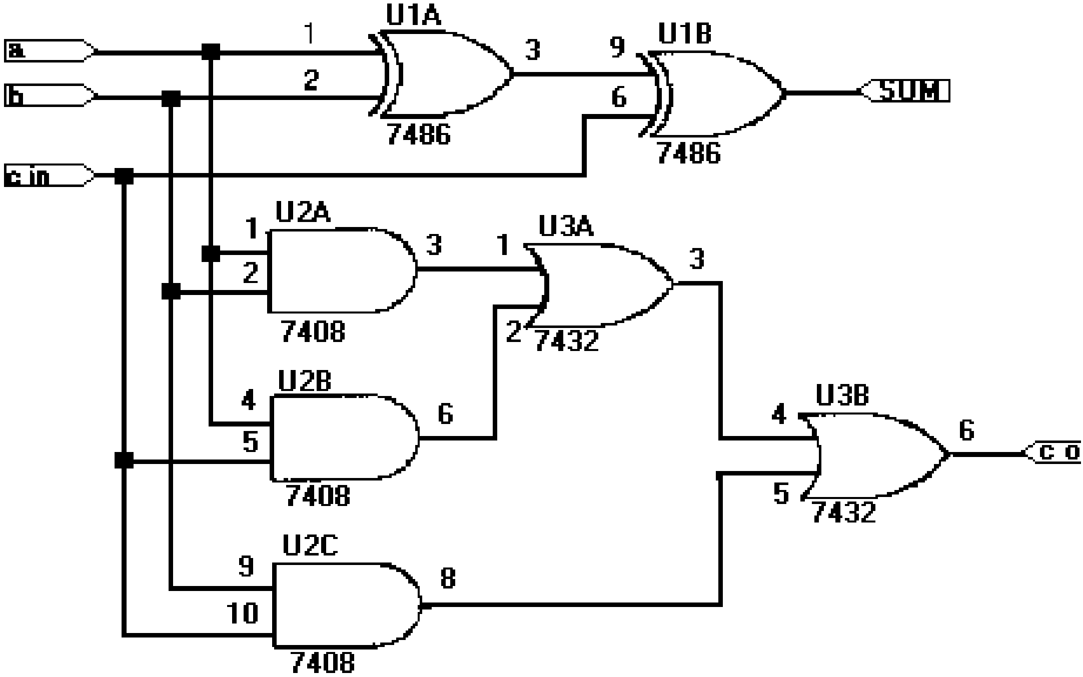


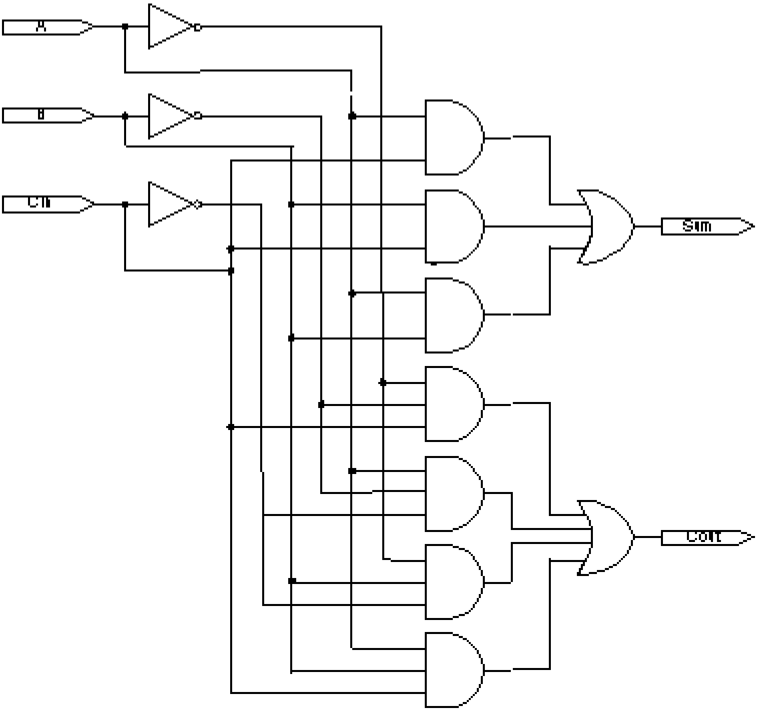




Full Adder: alternative structural models

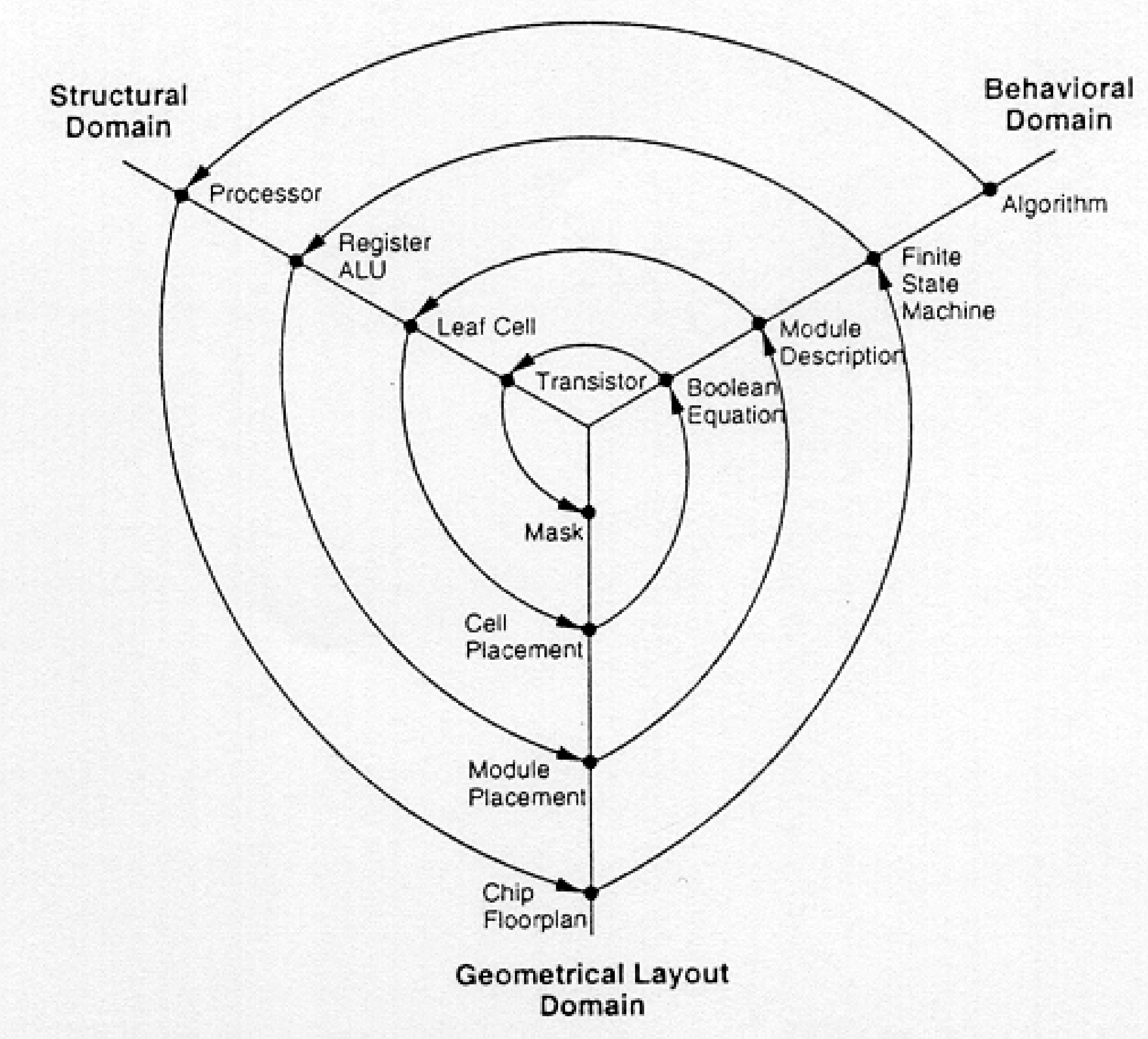






Are the behavioral models the same?

Logic Design flow



Half Adder

- A **Half-adder** is a Combinatorial circuit that performs the arithmetic sum of two bits.
- It consists of two inputs (x, y) and two outputs (Sum, Carry) as shown.

X	Y	Carry	Sum
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

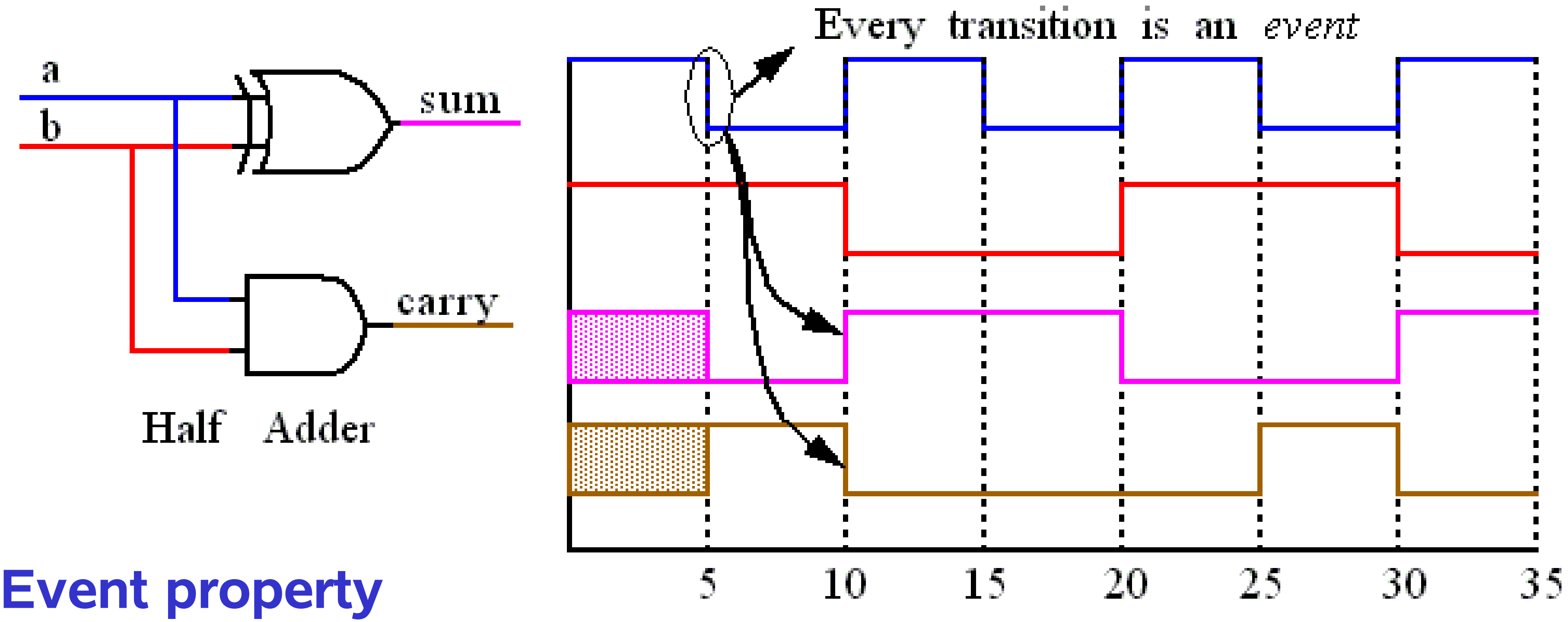
Behavioral Truth Table

Carry <= X AND Y;

Sum <= X XOR Y;

Half Adder: behavioral properties

What are the behavioral properties of the half-adder ciruit?



- **Event property**

  The event on a, from 1 to 0, changes the output

- **Propagation delay property**

  The output changes after 5ns propagation delay

- **Concurrency property:** Both XOR & AND gates compute new output values concurrently when an input changes state

Half Adder: Design Entity

- Design entity

  A component of a system whose behavior is to be described and simulated

- Components to the description

  - entity declaration

    The interface to the design

    There can only be one interface declared

  - architecture construct

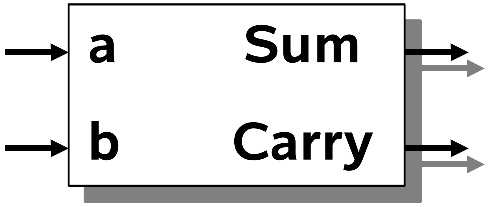
    The internal behavior or structure of the design

    There can be many different architectures

  - configuration

    bind a component instance to an entity-architecture pair

Half Adder: Entity



```
ENTITY half_adder IS
        PORT (
                a, b:          IN std_logic;
                sum, carry:    OUT std_logic
        );
END half_adder;
```

- All keyword in capitals by convention
- VHDL is case insensitive for keywords as well as variables
- The semicolon is a statement separator not a terminator
- std_logic is data type which denotes a logic bit
  (U, X, 0, 1, Z, W, L, H, -)
- BIT could be used instead of std_logic but it is only (0, 1)

Half Adder: Architecture

```
ENTITY half_adder IS
        PORT (
                a, b:           IN std_logic;
                Sum, Carry:     OUT std_logic
        );
END half_adder;
```

must refer to entity name

```
ARCHITECTURE half_adder_arch_1 OF half_adder IS

BEGIN

                Sum <= a XOR b;

                Carry <= a AND b;

END half_adder_arch_1;
```

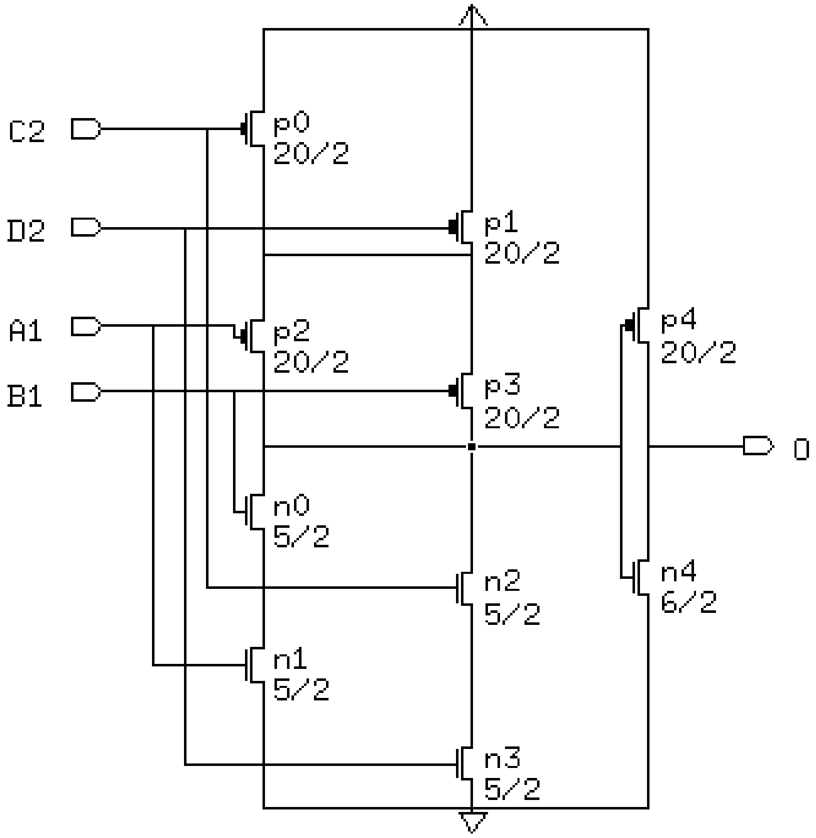
Half Adder: Architecture with Delay

```
ENTITY half_adder IS
        PORT (
                a, b:           IN std_logic;
                Sum, Carry:  OUT std_logic
        );
END half_adder;
```

```
ARCHITECTURE half_adder_arch_2 OF  half_adder  IS

BEGIN

                Sum <= ( a XOR b );

                Carry <= ( a AND b );

END half_adder_arch_2;
```

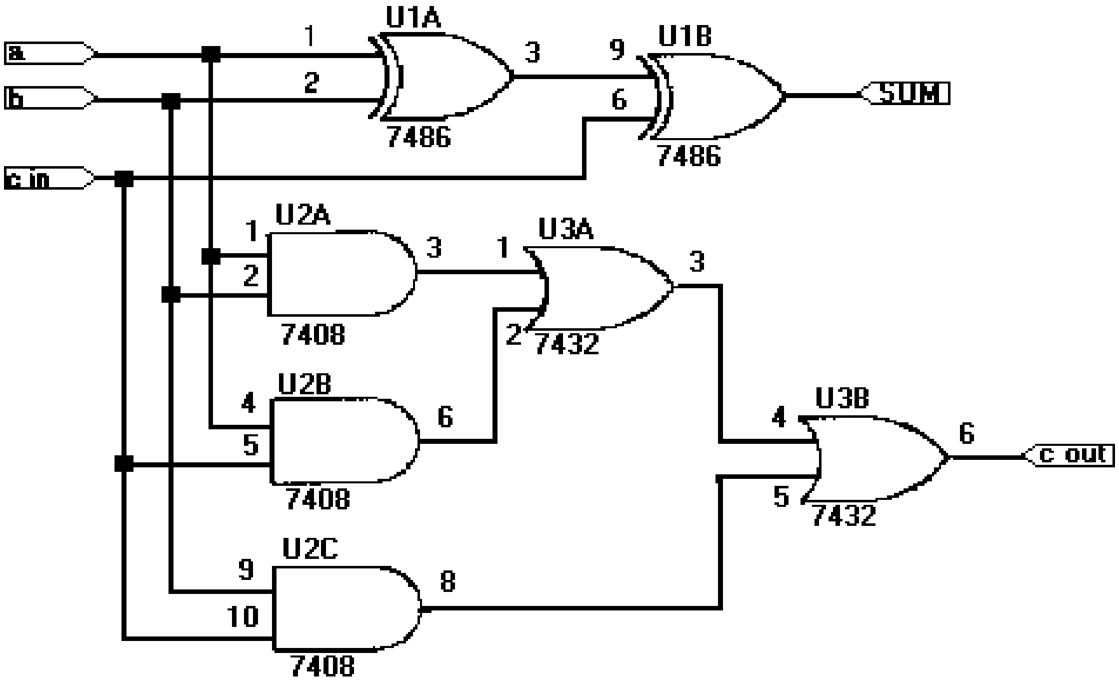
Homework #1: Problem 1

Convert the following schematic to (a) SPICE, (b) truth table, (c) logic gates (e) logic expression.



Homework #1: Problem #2

Re-write the following schematic as two logic expressions, sum=? And cout=? (b) as VHDL (c) and convert schematic using only NORs.


a
b
c in
U1A
1
2
3
7486
U1B
9
6
7486
SUM
U2A
1
2
3
7408
U3A
1
2
3
7432
U2B
4
5
6
7408
U3B
4
5
6
7432
c out
U2C
9
10
8
7408