Digital System Design

CE 325 L1
SPRING 2024

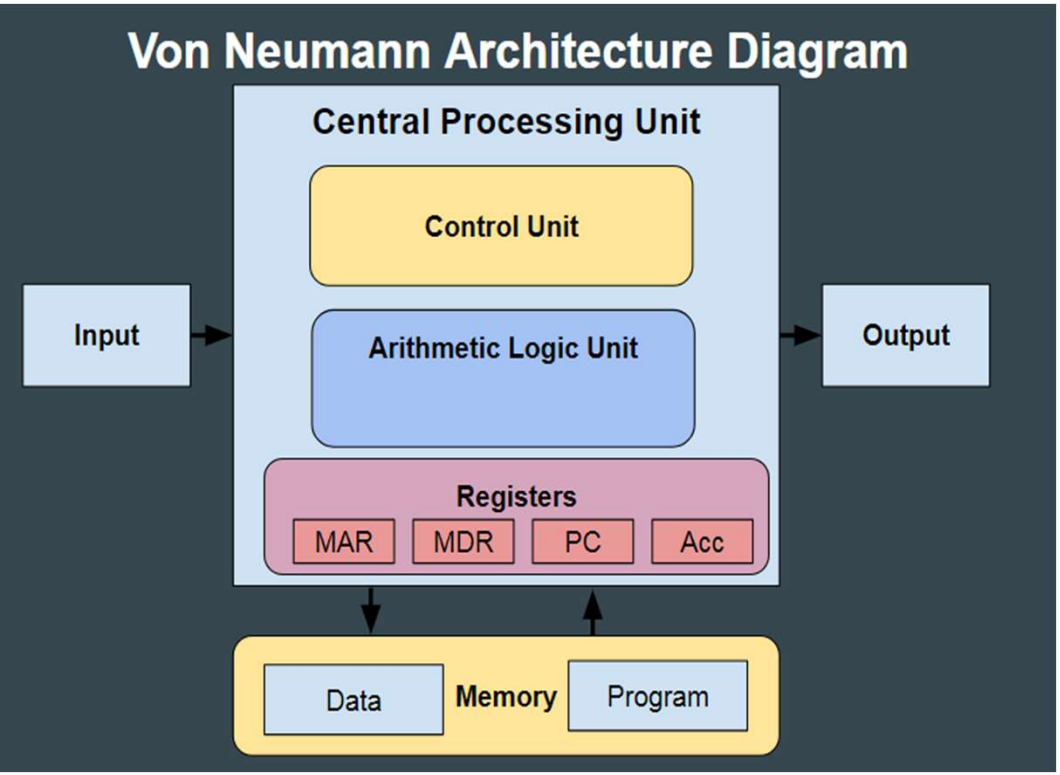
Instructor : Dr. Syed Arsalan Jawed
Associate Professor of Practice
arsalan.jawed@sse.habib.edu.pk
Room : W-301
Office Hours : Monday 0930-1030, Wednesday 1430-1530, Friday 0930-1030

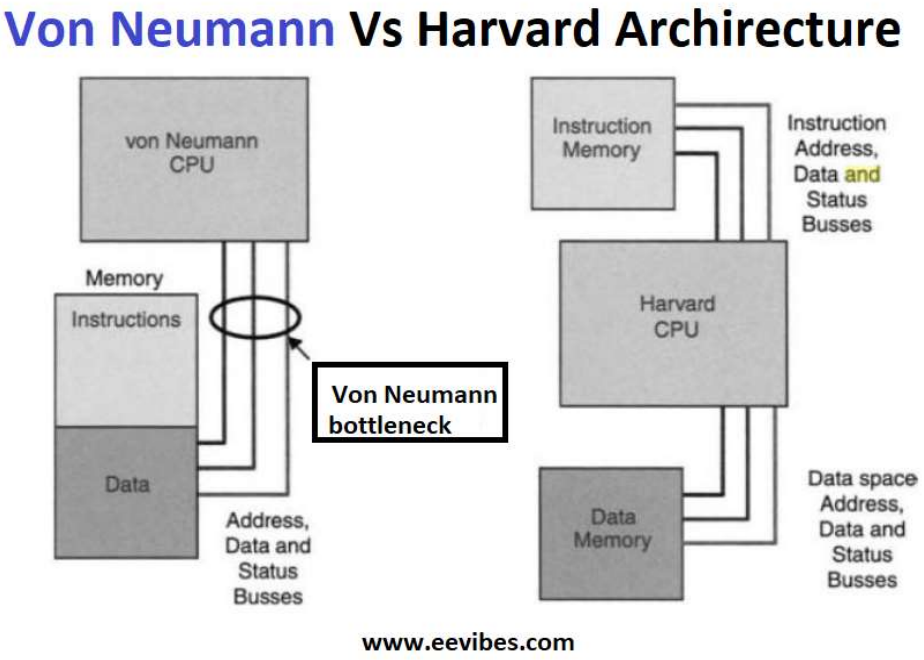
Complement these with the hand-written notes and explanations during the class on the white board!

Contents of this Lecture

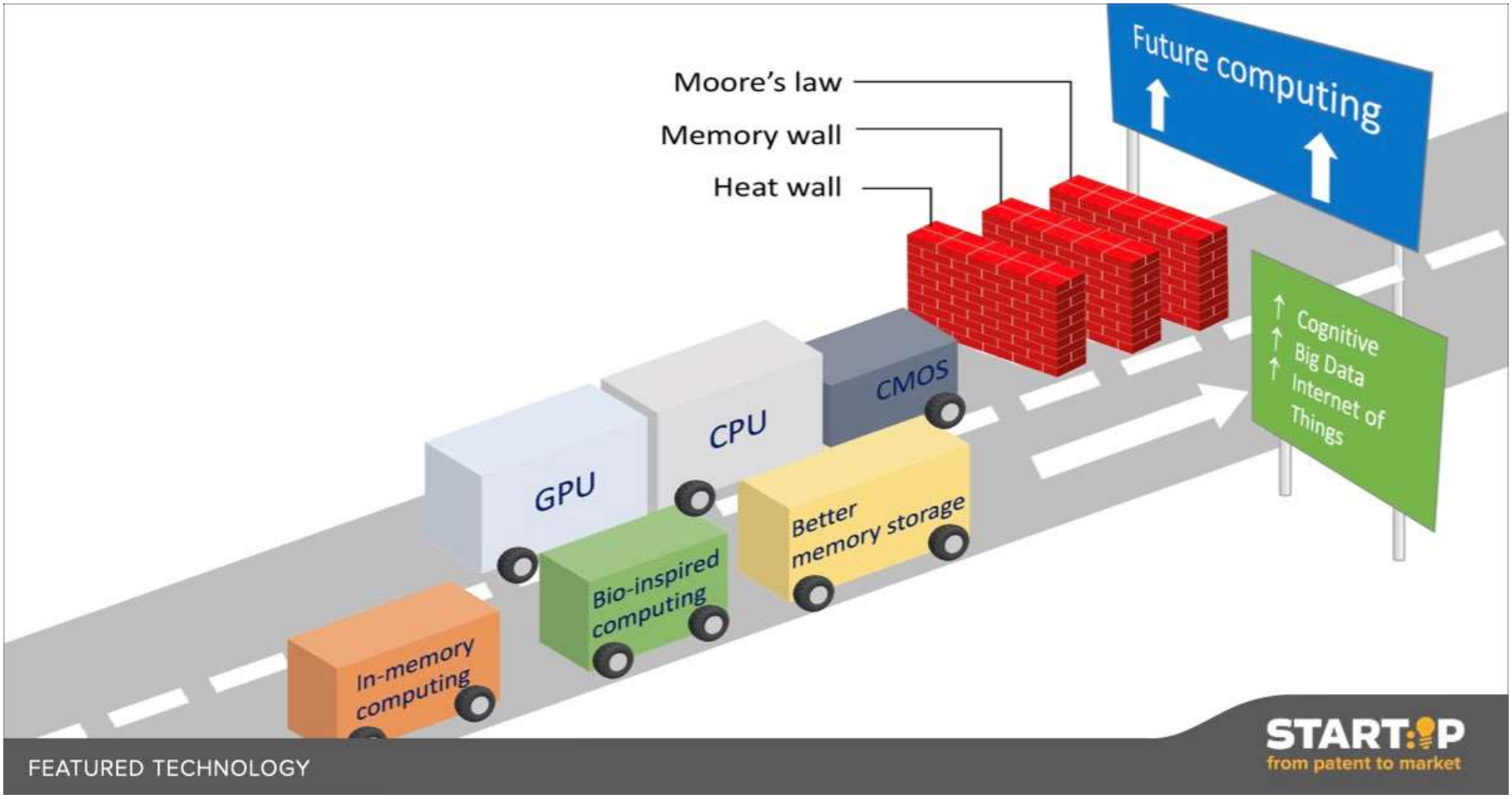
- Digital Design Architecting and Partitioning
  - Computer Architectures and Organization Evolution - briefly
  - Processing Elements / Storage Elements / Interconnection Topologies
  - Upcoming architecture of In-Memory Compute
  - Functional Components inside the Processing Element
    - Datapaths
    - Controllers
- The concept of Register-Transfer-Logic
  - Going from one DFF to the other – through a certain logic cloud
  - Partitioning between Combinational and Sequential logic
- Combinational Logic main functional components and methodologies
  - Static CMOS Logic Family for Gate Design
  - Timing, Area and Power Concepts

Conventional Computer Architecture

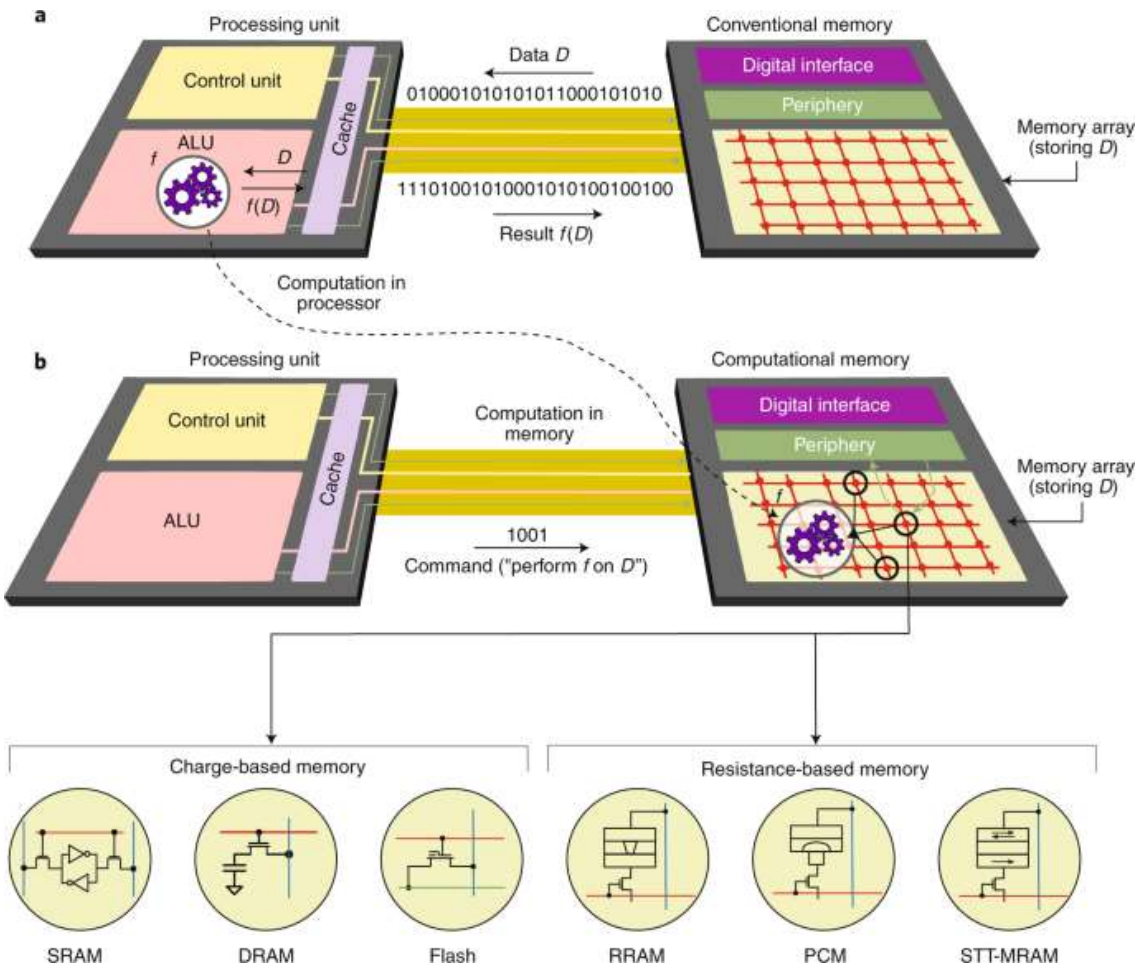




Emerging Dynamic of Digital Computing Systems

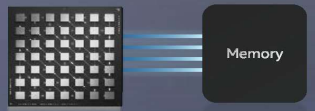
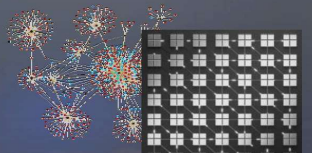
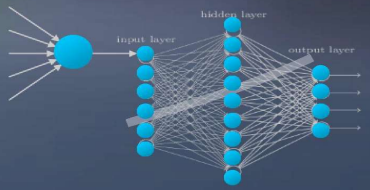
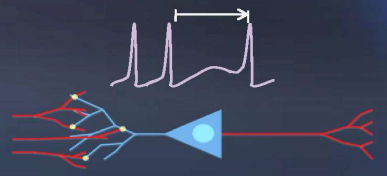


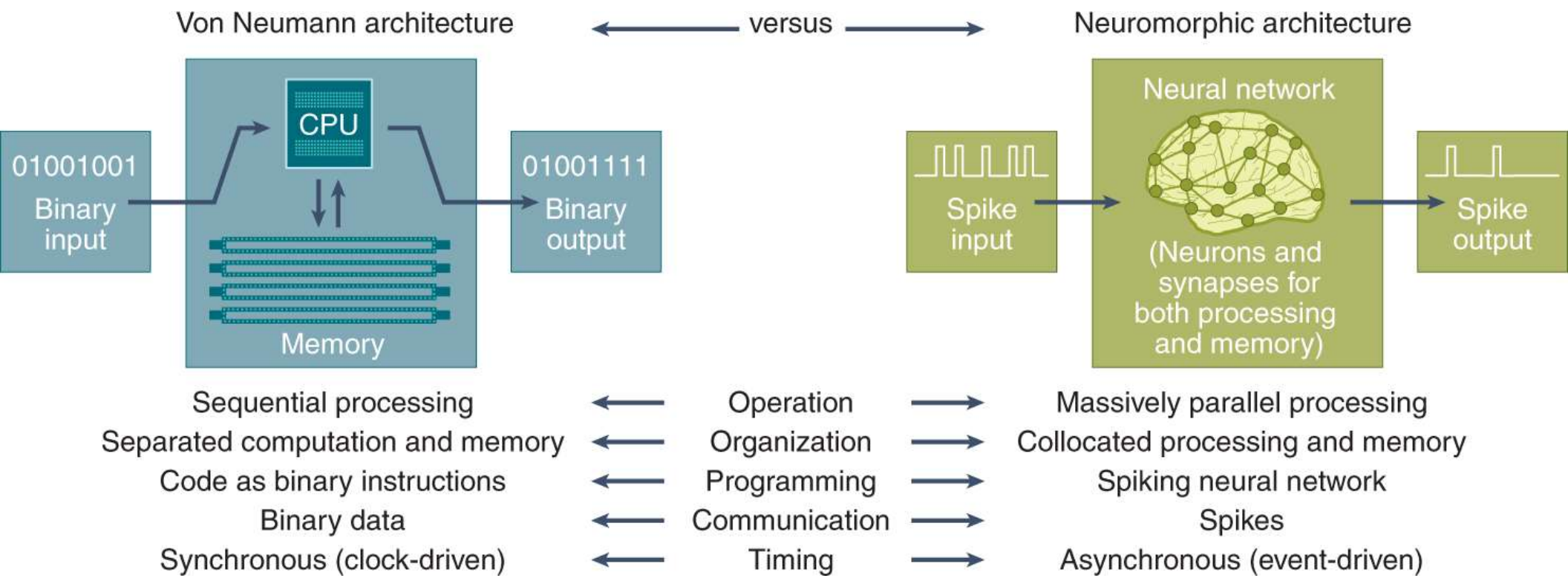
Upcoming In-Memory Compute



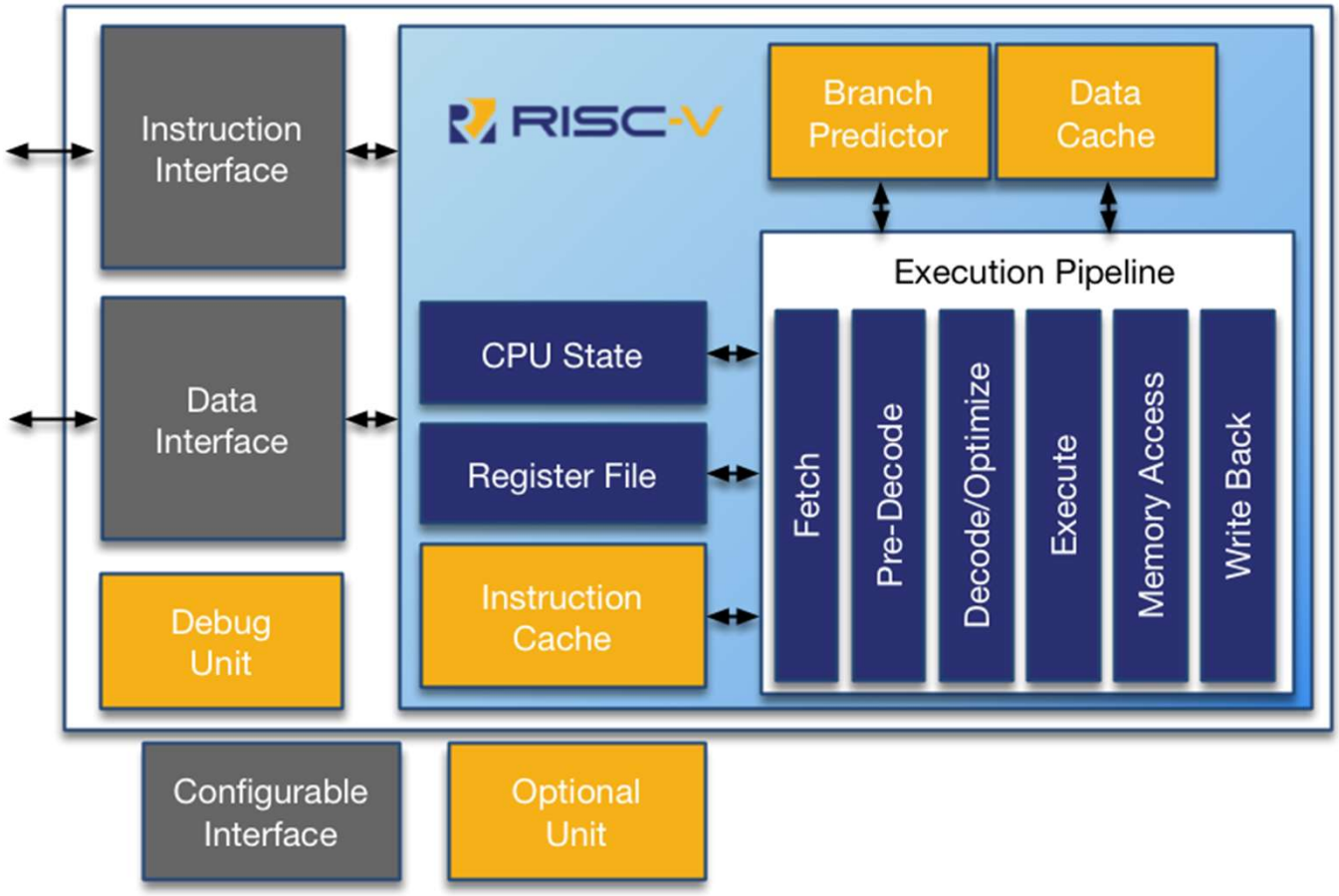
Neuromorphic Computing

Motivates a New Kind of Computer Architecture

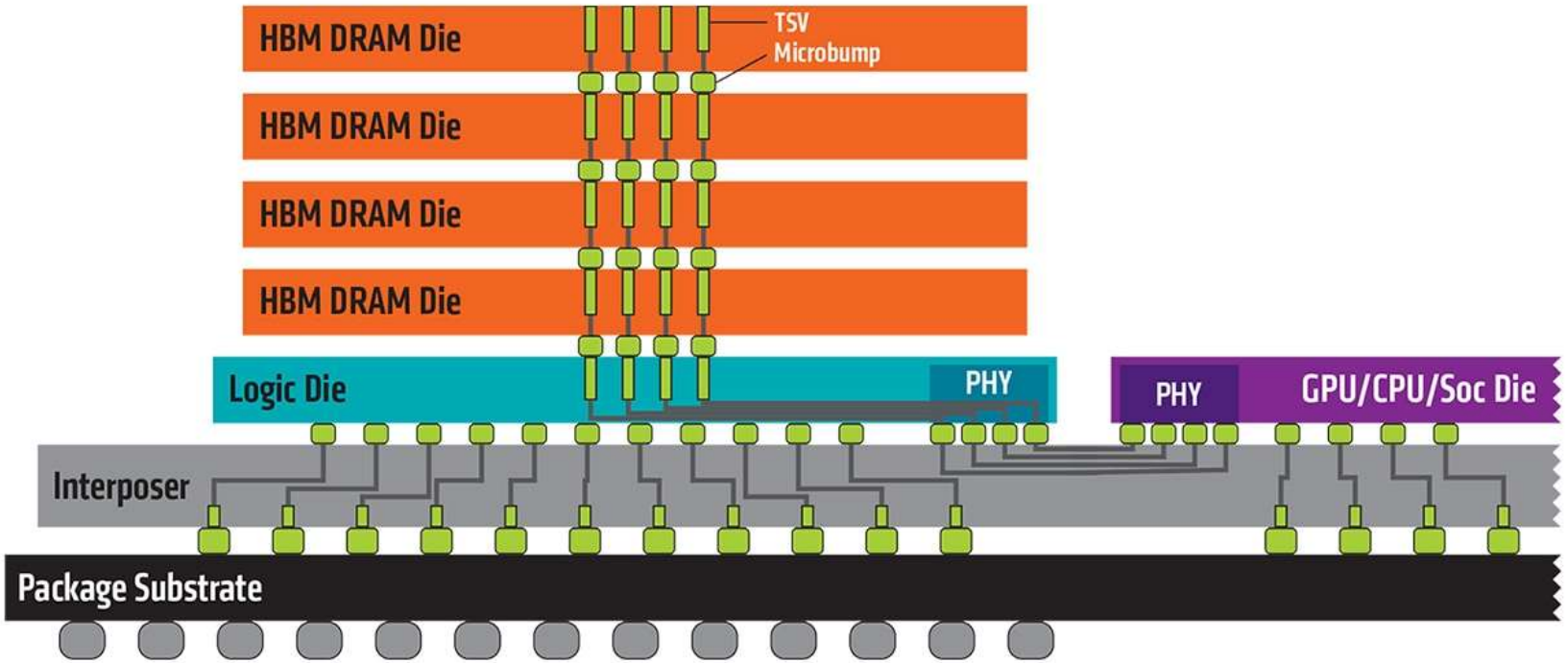
Standard Computing	Parallel Computing	Neuromorphic Computing
		
PROGRAMMING BY ENCODING ALGORITHMS	OFFLINE TRAINING USING LABELED DATASETS	LEARN ON THE FLY THROUGH NEURON FIRING RULES
SYNCHRONOUS CLOCKING	SYNCHRONOUS CLOCKING	ASYNCHRONOUS EVENT-BASED SPIKES
SEQUENTIAL THREADS OF CONTROL	PARALLEL DENSE COMPUTE	PARALLEL SPARSE COMPUTE
		


Von Neumann architecture
versus
Neuromorphic architecture
CPU
01001001
Binary input
01001111
Binary output
Memory
Neural network
Spike input
(Neurons and synapses for both processing and memory)
Spike output
Sequential processing
Operation
Massively parallel processing
Separated computation and memory
Organization
Collocated processing and memory
Code as binary instructions
Programming
Spiking neural network
Binary data
Communication
Spikes
Synchronous (clock-driven)
Timing
Asynchronous (event-driven)

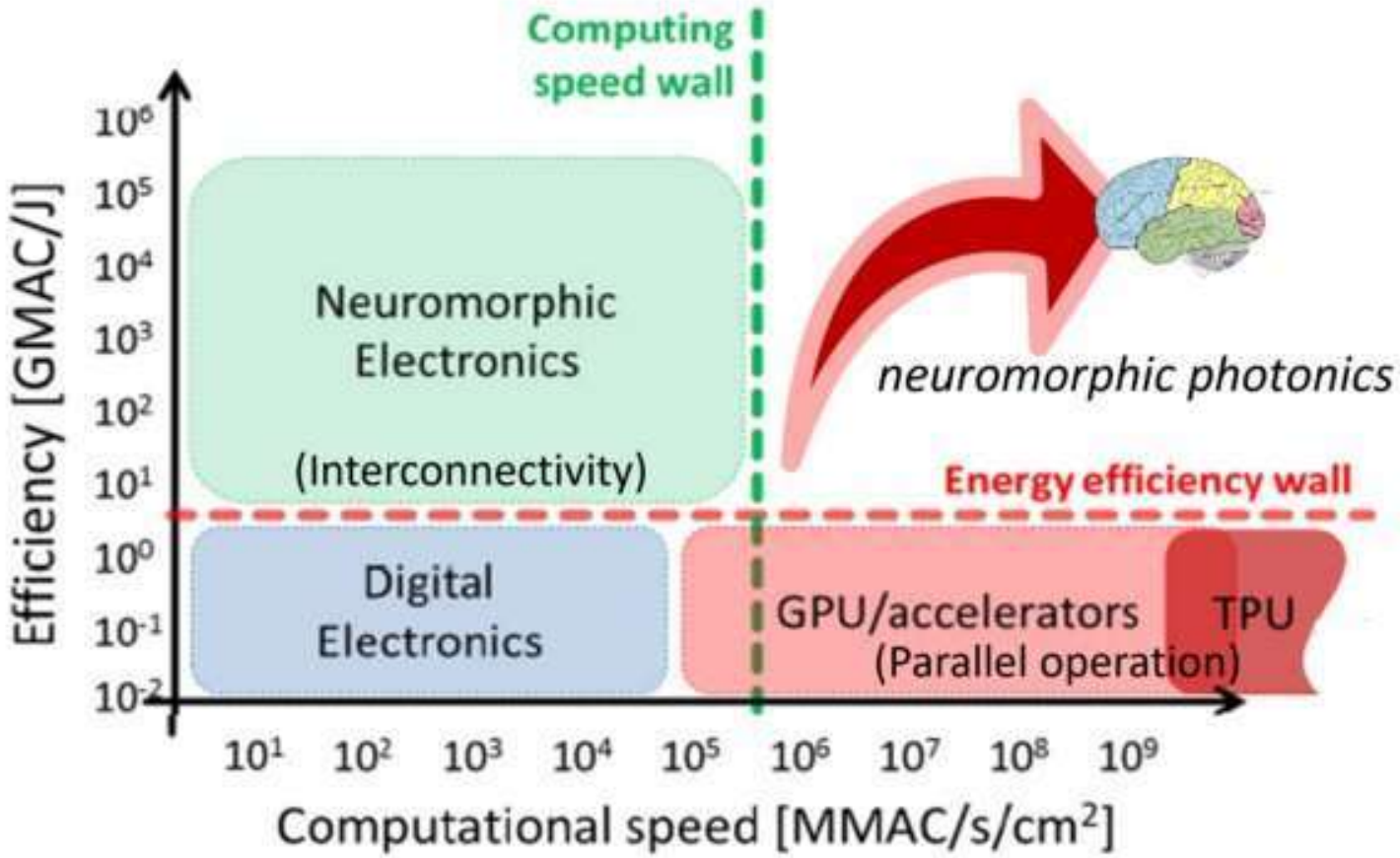
RISC-V Open Source Architecture



High Bandwidth Memory

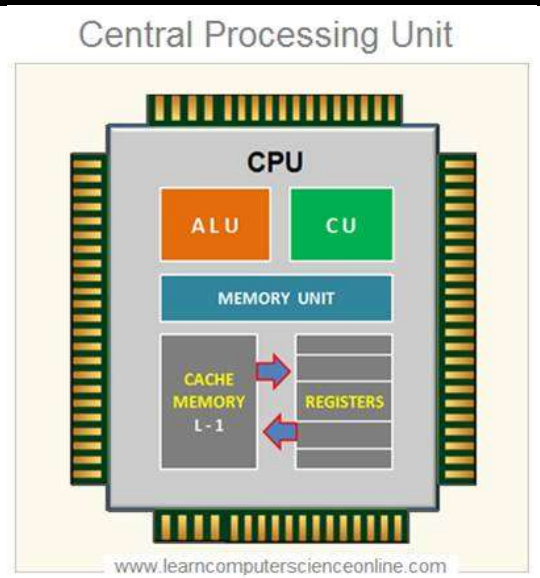


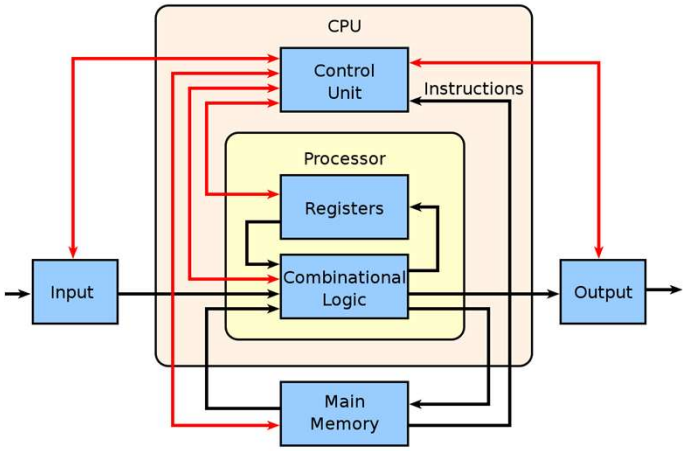
Changing Architectures, Organizations, Interconnections



Main Components inside a Processing Element

- ALU → Specially designed datapath units
- Control Unit → responsible for timing all the events
- Registers / Internal Memory → Program Counter, Data Reg, Inst. Reg,
- Cache
- Buses

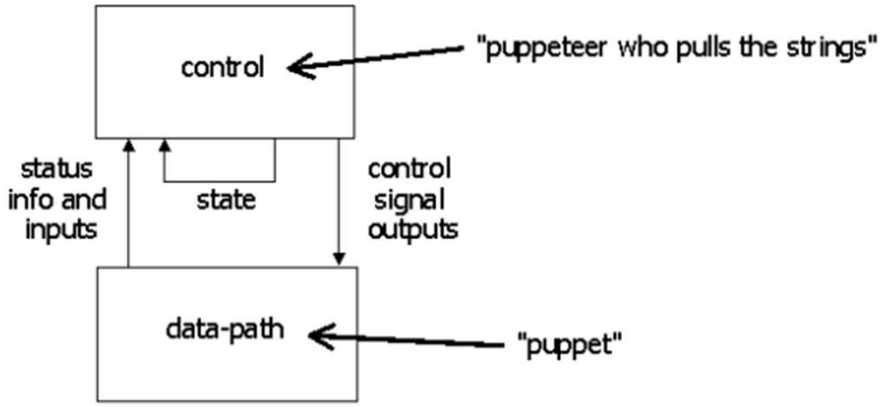


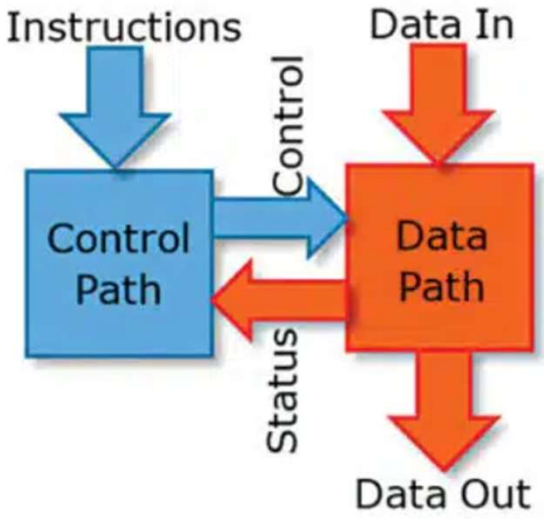


DataPath and ControlPath

Data-path and control

- Digital hardware systems = data-path + control
  - datapath: registers, counters, combinational functional units (e.g., ALU), communication (e.g., busses)
  - control: FSM generating sequences of control signals that instructs datapath what to do next

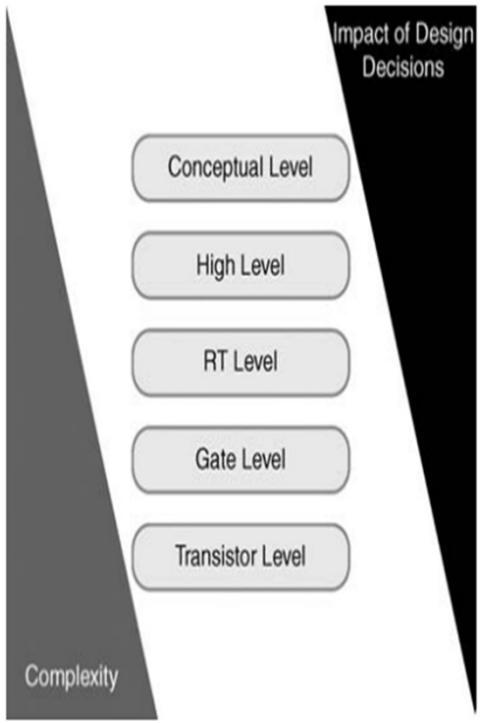




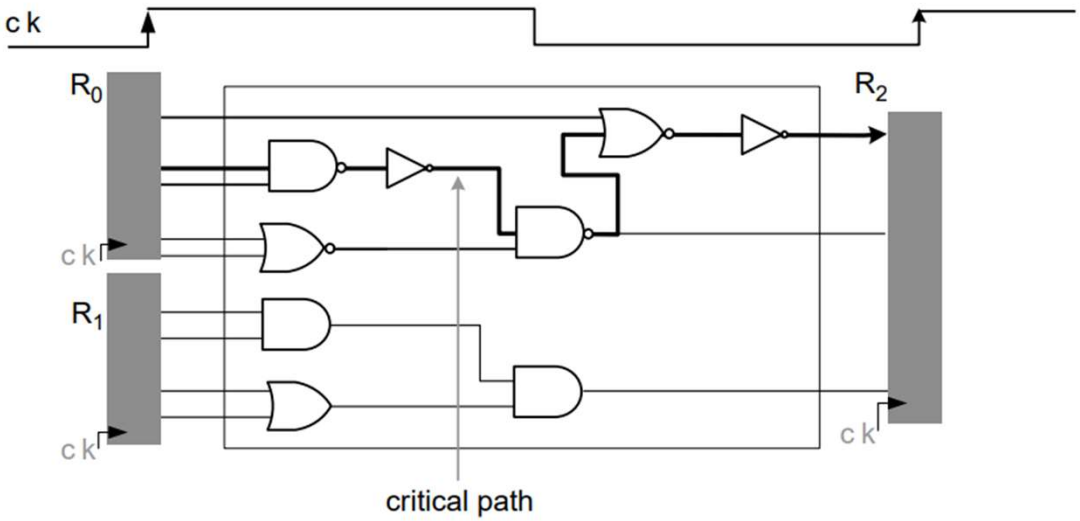
Four Abstraction Levels

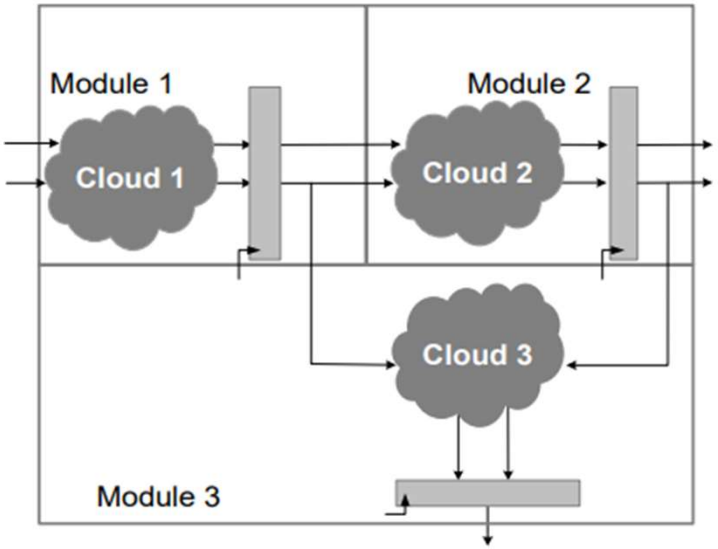
- Switch/Transistor in Triode Level
- Gate Level
- Data Flow Level or RTL Level
  - Expressions, operands and operators characterize this level. Most of the operators used in dataflow modeling are common to software programmers, but there are a few others that are specific to HW design.
- Behavioral or Algo Level
  - The behavioral level is the highest level of abstraction in Verilog.
  - Provides high level language constructs like for, while, repeat, if else and case.
  - Designers with a software programming background already know these constructs.
  - Although the constructs are handy and very powerful, the programmer must know that each construct in RTL Verilog infers hardware.
  - High level constructs are very tempting to use, but the HW consequence of their inclusion must be well understood.

  - For example, for loop to a software programmer suggests a construct that simply repeats a block of code a number of times, but if used in RTL Verilog the code infers multiple copies of the logic in the loop



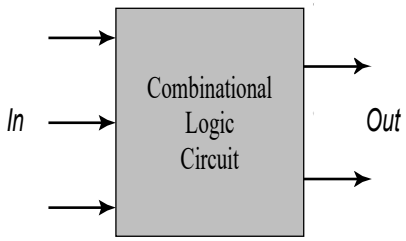
Register Transfer Level / Partitioning between Combination and Sequential Logic





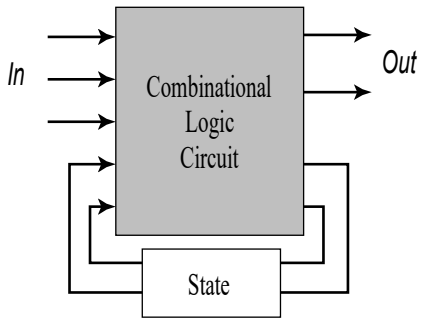
Combinational Logic

- Output at time t depends only at input at time t



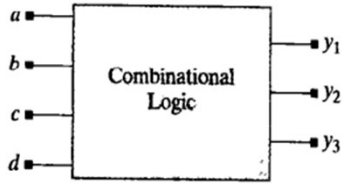
Combinational

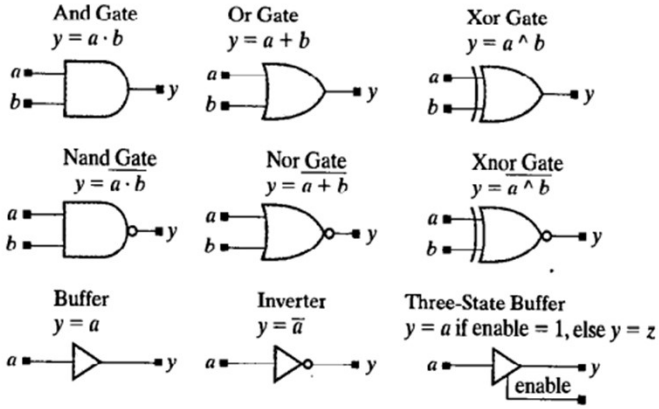
Output = *f(In)*



Sequential

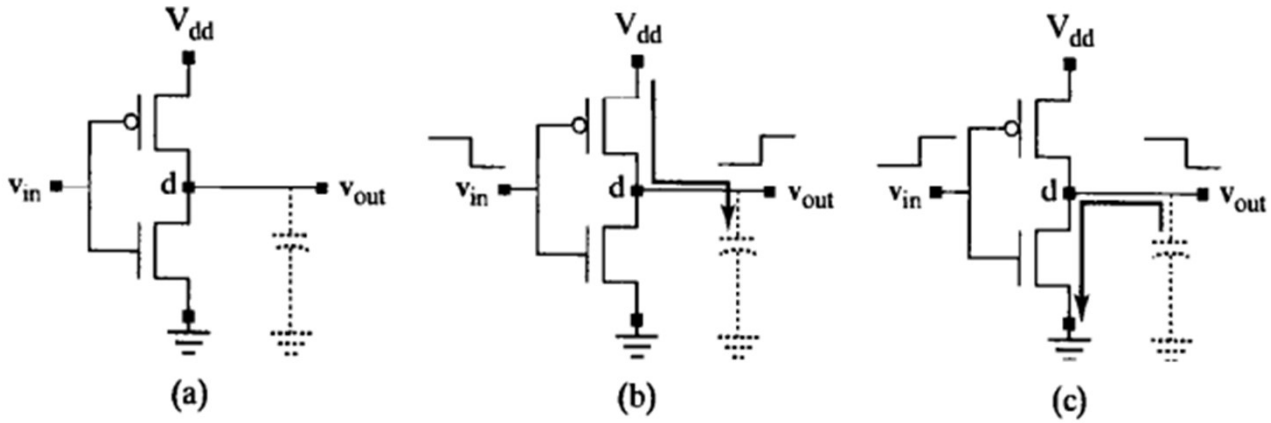
Output = *f(In, Previous In)*





Symbol	Logic Operation
+	Logic "or"
·	Logical "and"
⊕	Exclusive "or"
∧	Exclusive "or"
′	Logical negation
¯	Logical negation (overbar)

Looking into the most basic block : Inverter



- Explain:
  - Why Inversion to start with?
  - Why cant NMOS deliver a 1?
  - Concept of Ron
  - Where are capacitors coming from?
  - Equation of propagation delay

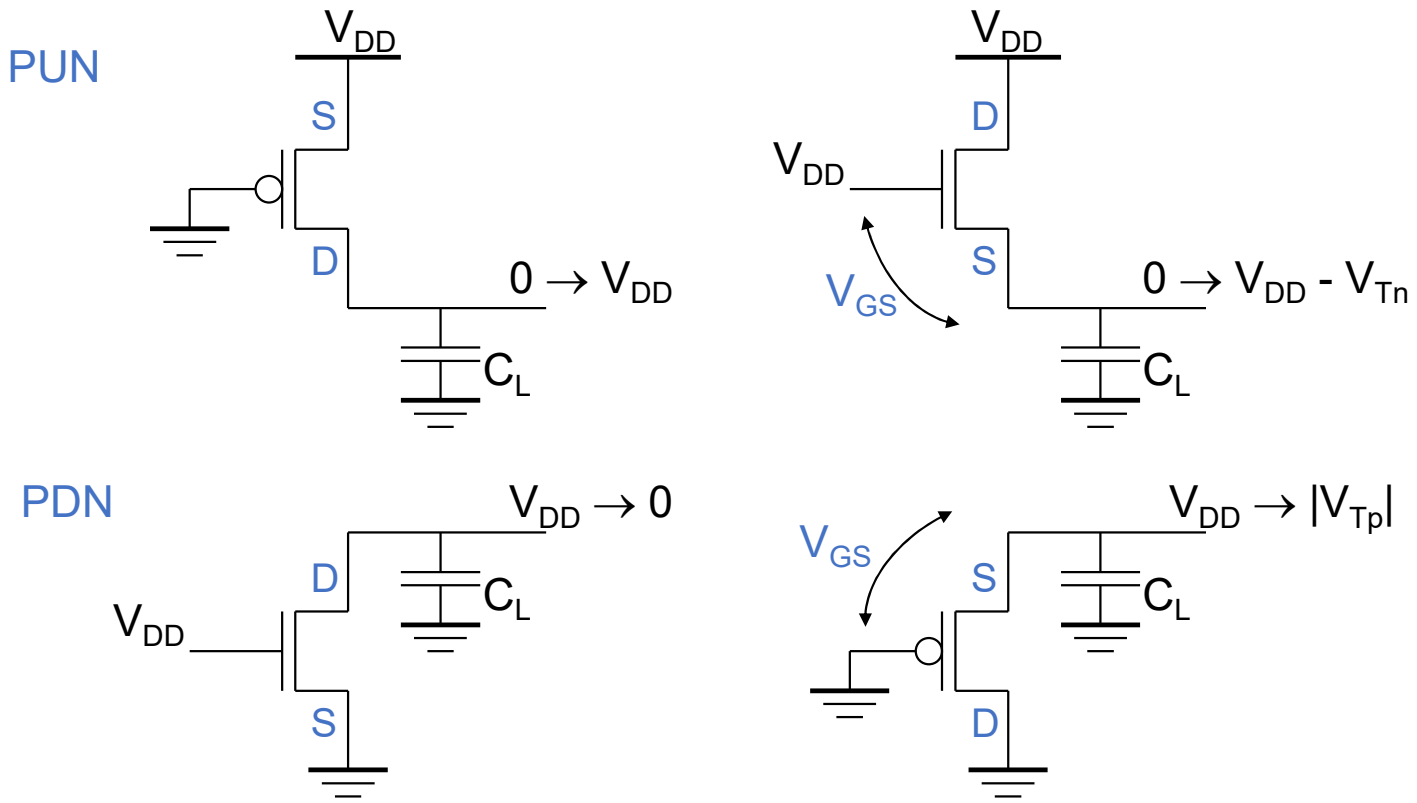
- Explain:
  - Draw input and output and show 50%-50% for tpLH and also Ron and C
  - Assume starting time is zero.

$$v_{out}(t) = (1 - e^{-t/\tau})\,V \qquad (1.13)$$

The time to reach the 50% point is easily computed as $t = \ln(2)\tau = 0.69\tau$. Similarly, it takes $t = \ln(9)\tau = 2.2\tau$ to get to the 90% point. It is worth memorizing these numbers, as they are extensively used in the rest of the text.

$$t_p = \frac{t_{pLH} + t_{pHL}}{2}$$

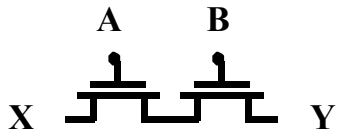
Threshold Drops



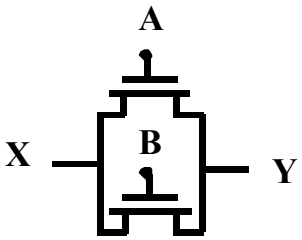
NMOS Transistors in Series/Parallel Connection

Transistors can be thought as a switch controlled by its gate signal

NMOS switch closes when switch control input is high



Y = X if A and B

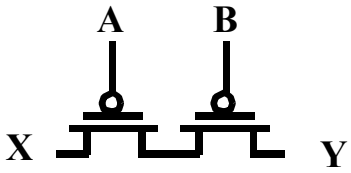


Y = X if A OR B

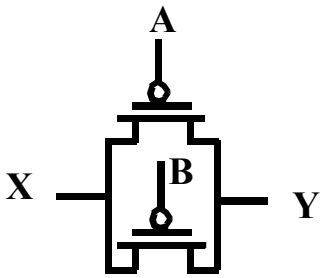
NMOS Transistors pass a "strong" 0 but a "weak" 1

PMOS Transistors in Series/Parallel Connection

PMOS switch closes when switch control input is low



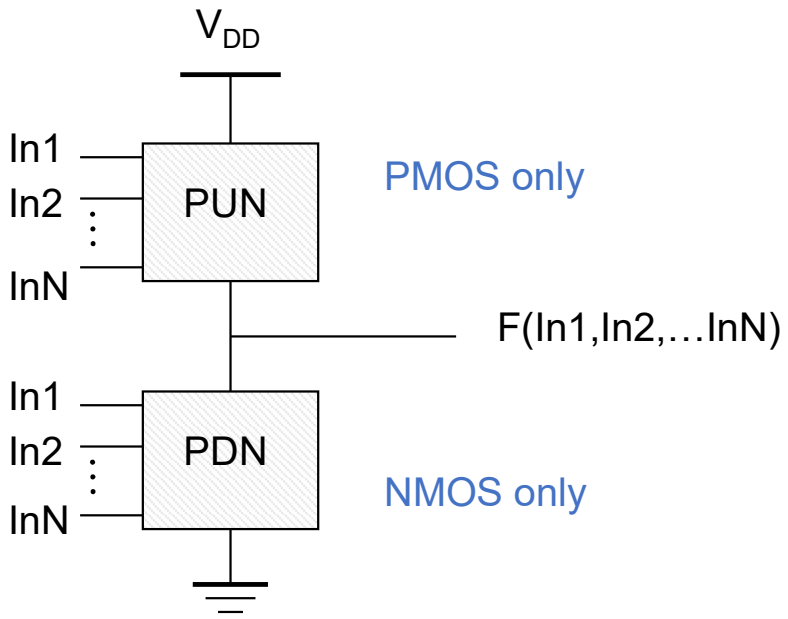
$Y = X$ if $\overline{A}$ AND $\overline{B} = \overline{A + B}$



$Y = X$ if $\overline{A}$ OR $\overline{B} = \overline{AB}$

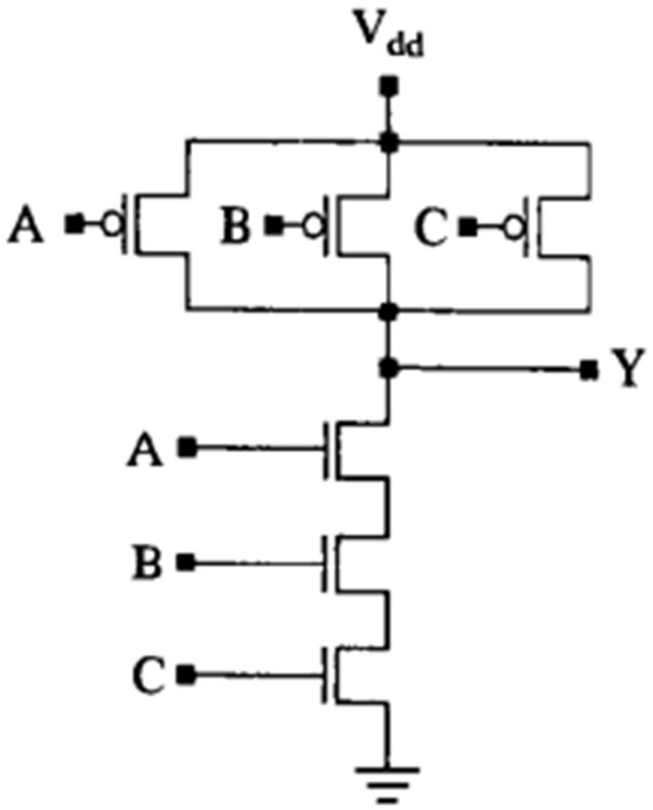
PMOS Transistors pass a "strong" 1 but a "weak" 0

Looking into NAND Gate



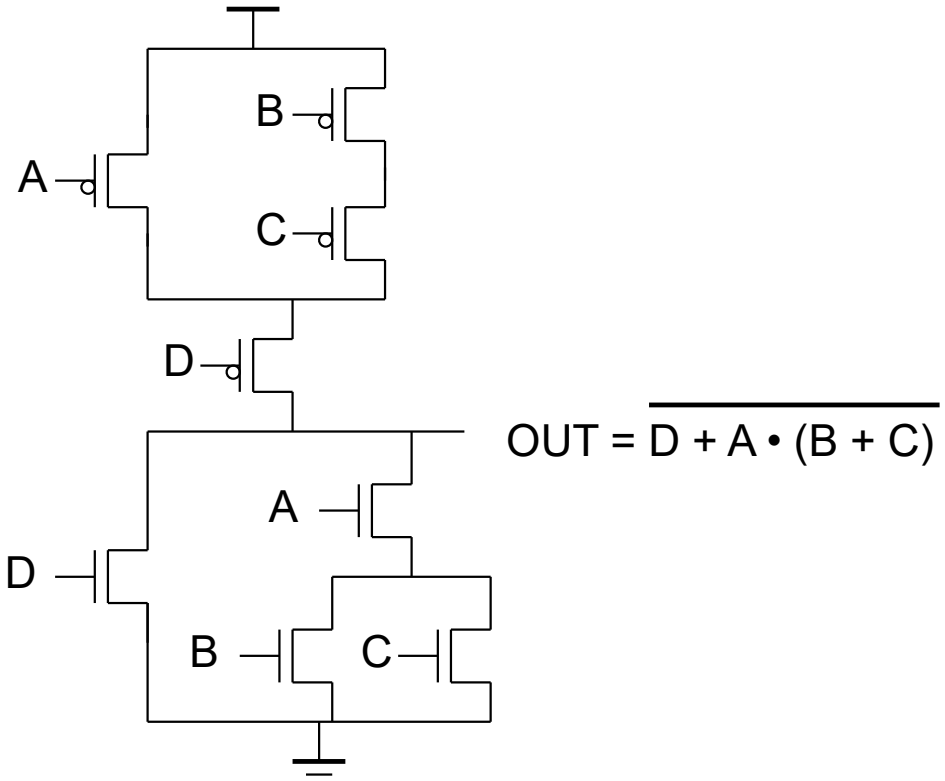
PUN and PDN are dual logic networks

NAND



- Explain
  - How Anding is happening
    - Explain we are implementing complement of the required function in the pull down network.
    - We are implementing dual (double bar) with the inverted input in the pull up network
  - Three Ron in series so sizing is critical
    - Relate it to different strengths of gate available for RTL logic

Complex CMOS Gate



$$OUT = \overline{D + A \bullet (B + C)}$$

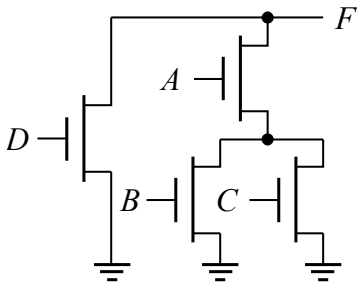
Using complementary CMOS logic, consider the synthesis of a complex CMOS gate whose function is F = D + A· (B +C).

The first step in the synthesis of the logic gate is to derive the pull-down network. by using the fact that NMOS devices in series implements the AND function and parallel device implements the OR function.

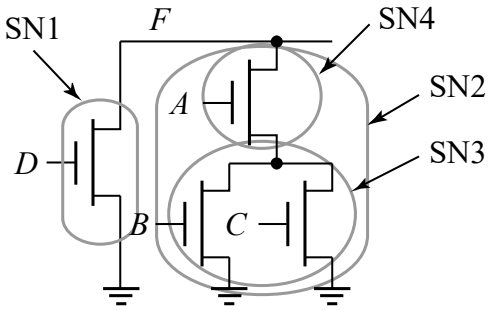
The next step is to use duality to derive the PUN in a hierarchical fashion

Break network into sub-nets.

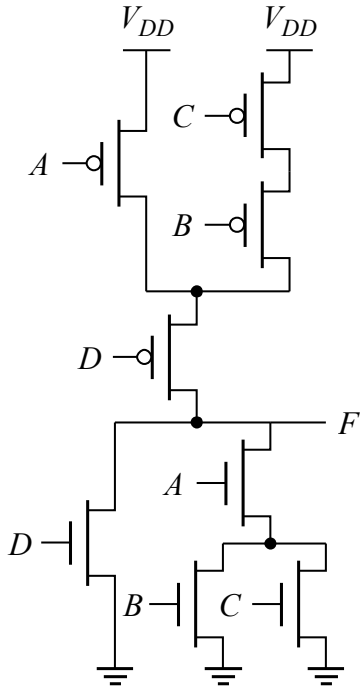
Constructing a Complex Gate



(a) pull-down network



(b) Deriving the pull-up network hierarchically by identifying sub-nets



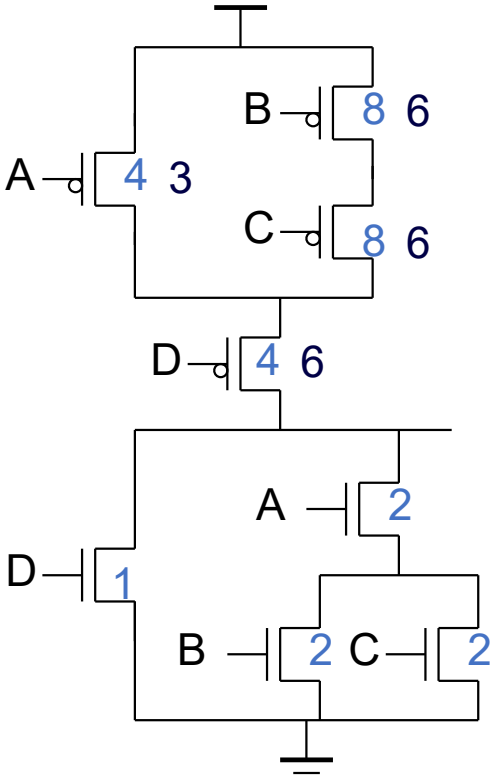
(c) complete gate

Transistor Sizing a Complex CMOS Gate

Synthesizer will do it for you using your Verilog RTL code and using the timing constraints file!

Must appreciate the automation.

But must appreciate the wrong guidance will lead to wrong design.



$$OUT = \overline{D + A \bullet (B + C)}$$

Where Does Power Go in CMOS?

- **Dynamic Power Consumption**

  Charging and Discharging Capacitors

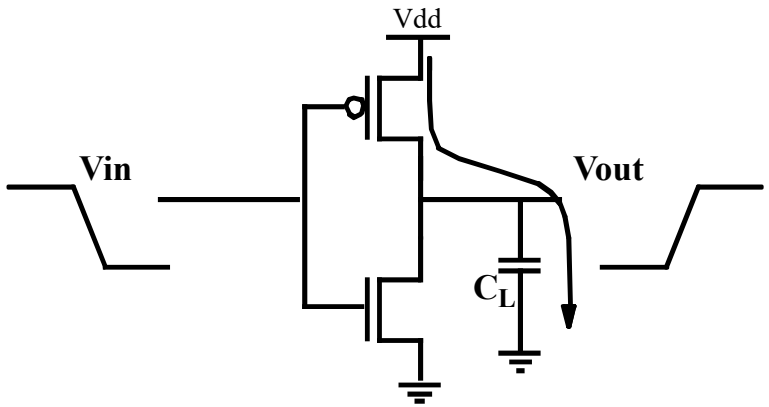
- **Short Circuit Currents**

  Short Circuit Path between Supply Rails during Switching

- **Leakage**

  Leaking diodes and transistors

Dynamic Power Dissipation

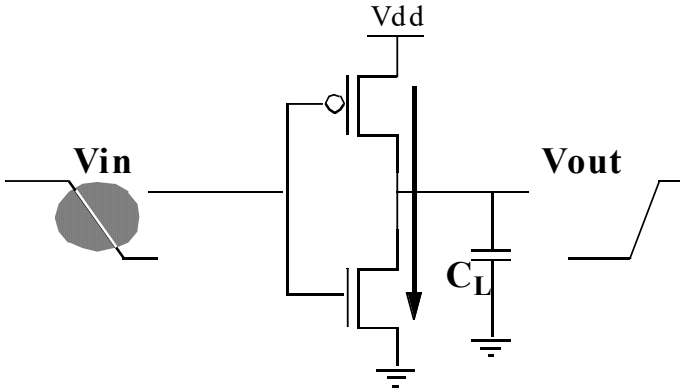


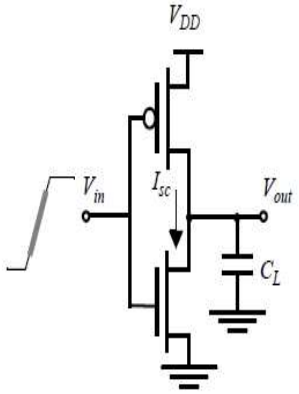
$$\text{Energy/transition} = C_L * V_{dd}^2$$

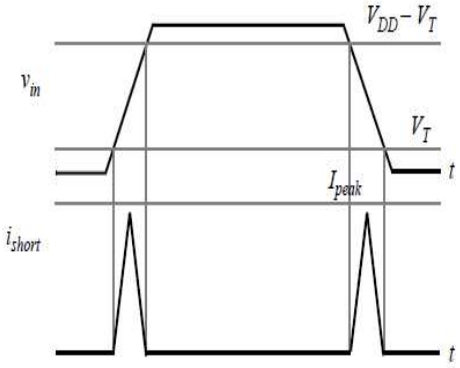
$$\text{Power} = \text{Energy/transition} * f = C_L * V_{dd}^2 * f$$

- **Not a function of transistor sizes if internal loading ignored!**
- **Need to reduce C_L, V_{dd}, and f to reduce power.**

Short Circuit Currents







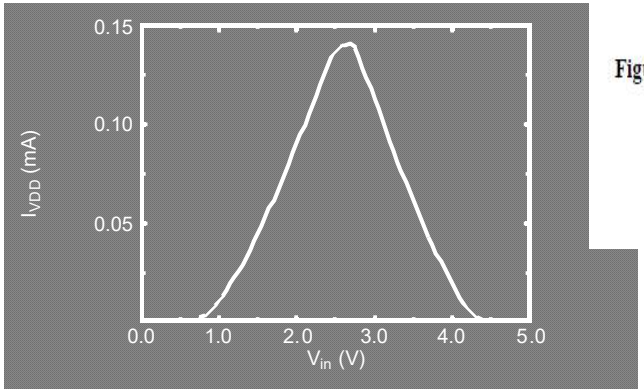
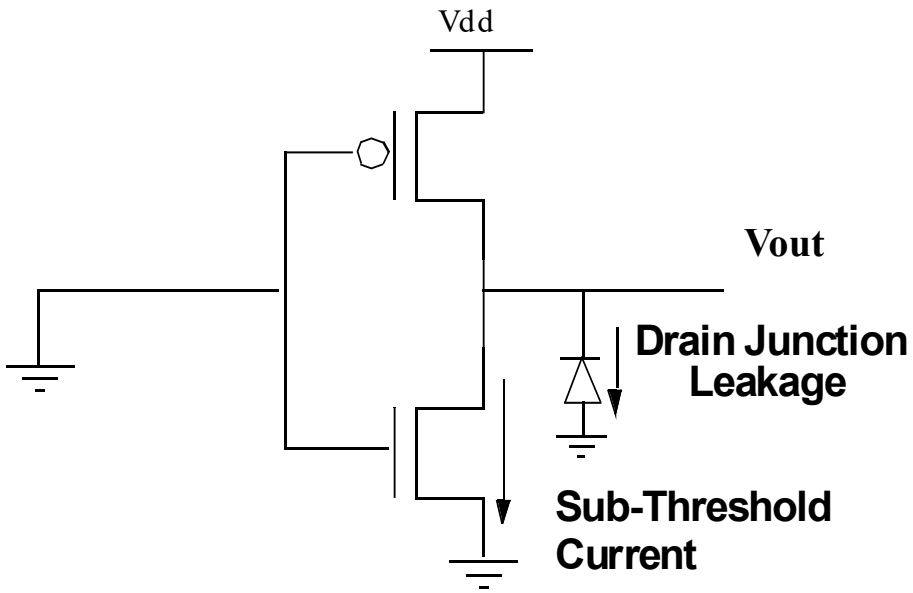


Figure 5.30 Short-circuit currents during transients.

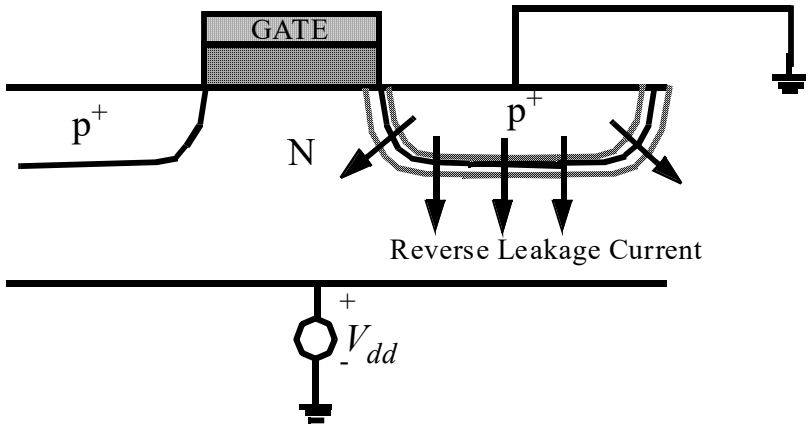
$$t_{sc} = \frac{V_{DD}-2V_T}{V_{DD}} t_s \approx \frac{V_{DD}-2V_T}{V_{DD}} \times \frac{t_{r(f)}}{0.8} \qquad (5.5($$

Leakage



Sub-threshold current one of most compelling issues in low-energy circuit design!

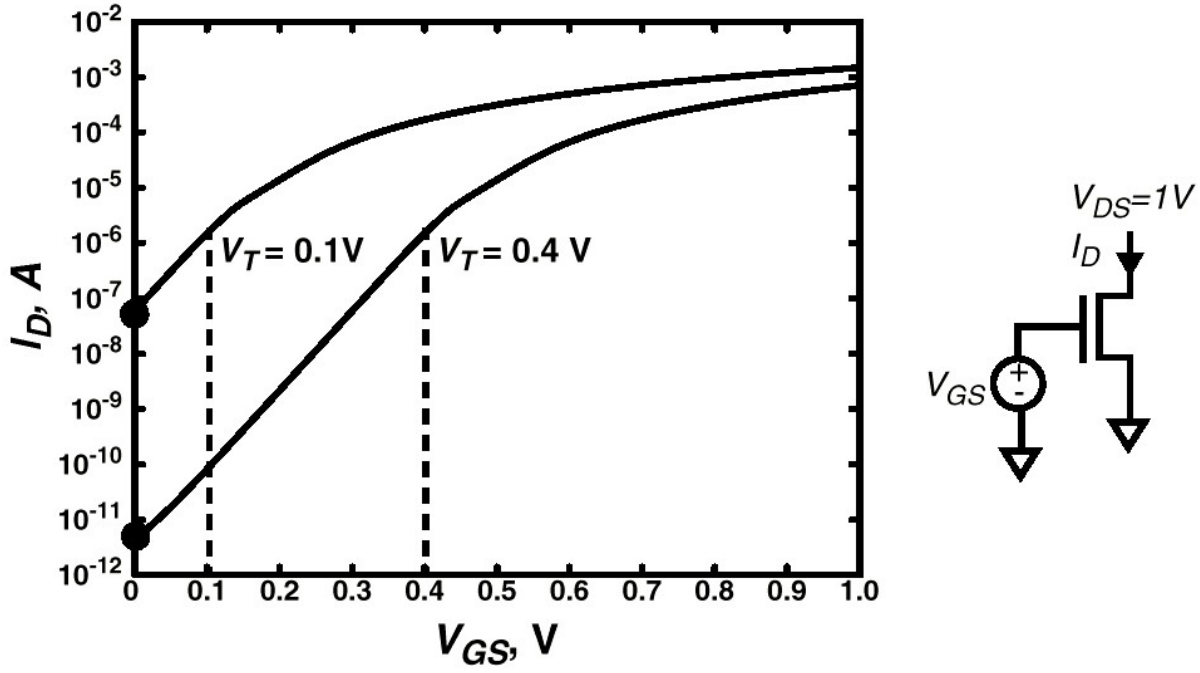
Reverse-Biased Diode Leakage



$$I_{DL} = J_S \times A$$

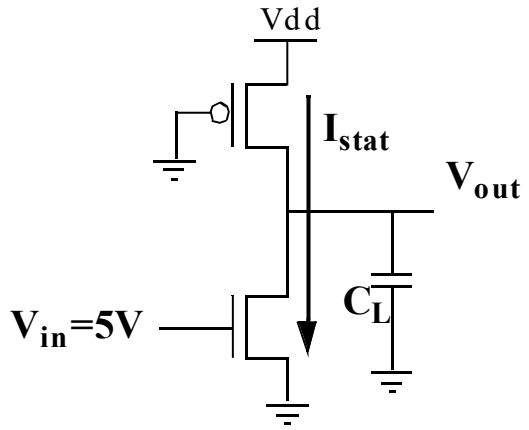
JS = 10-100 pA/µm2 at 25 deg C for 0.25µm CMOS
JS doubles for every 9 deg C!

Subthreshold Leakage Component



- Leakage control is critical for low-voltage operation

Static Power Consumption



$$P_{stat} = P_{(In=1)} \cdot V_{dd} \cdot I_{stat}$$

Wasted energy …
Should be avoided in almost all cases,
but could help reducing energy in others (e.g. sense amps)

Total Power

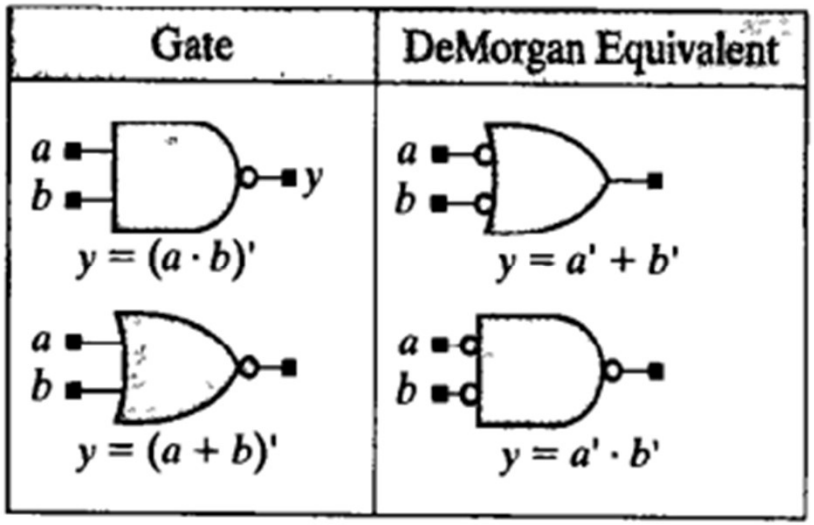
$$P_{tot} = P_{dyn} + P_{dp} + P_{stat} = (C_L V_{DD}^2 + V_{DD} I_{peak} t_s) f_{0 \to 1} + V_{DD} I_{leak}$$

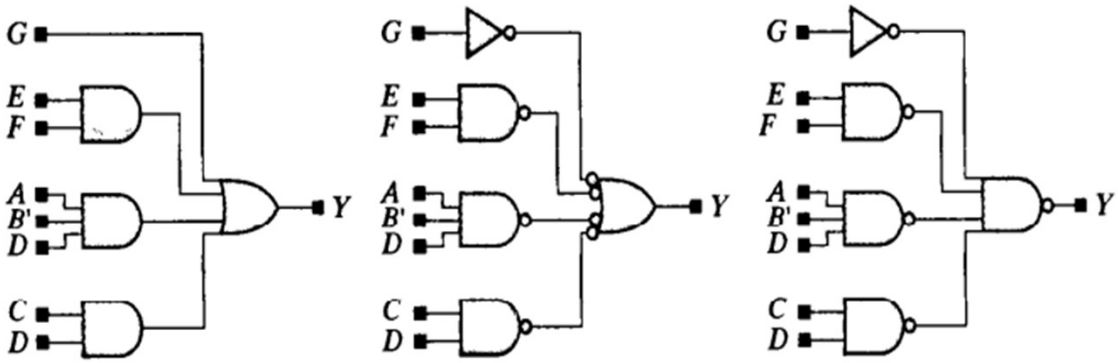
Building Blocks for Logic Design

NAND – NOR Structures

AS we saw, AND and OR are not implemented well in CMOS.

Better to implement SOP using NAND or NOR.





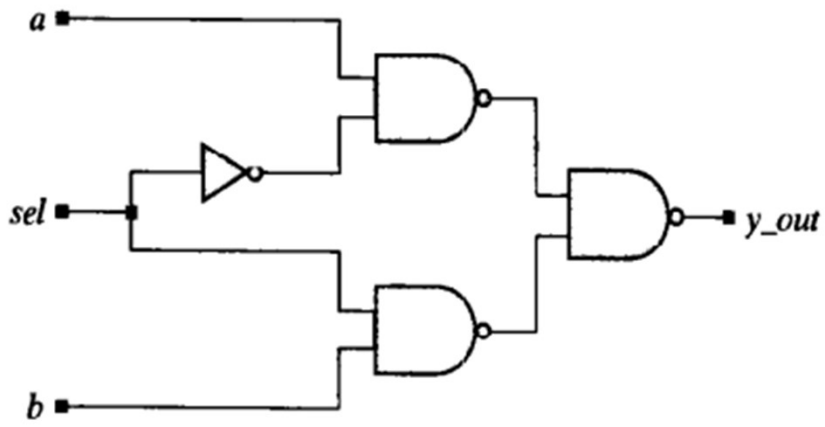
Multiplexers & Demux

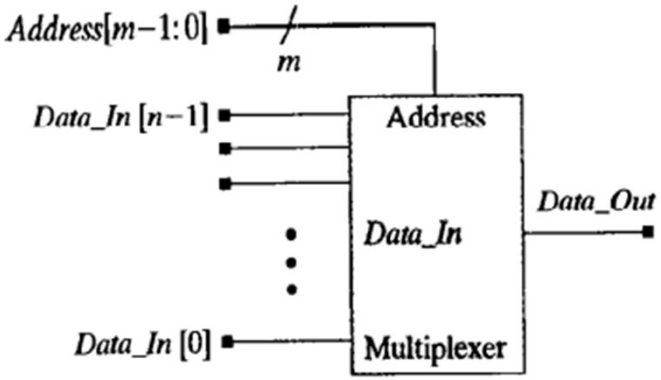
Multiplexers

Y_out = a.sel’ + b.sel

Can be used to implement combinational logic

Values of Boolean function assigned to input. Select lines used for decoding.





Encoders and Decoders

Mux and Demux just connect input and output dynamically. Do not change data pattern.

Encoders and Decoders change data-patterns.

Encoders → input word is wider → n inputs and m output , n=2^m
Decoder → output word is wider

Input	Output
00000	000
00001	001
00010	001
00011	010
00100	001
00101	010
00110	010
00111	011
01000	100
01001	010
01010	010
01011	011
01100	010
01101	011
01110	011
01111	100

Input	Output
10000	001
10001	010
10010	010
10011	011
10100	010
10101	011
10110	011
10111	100
11000	010
11001	011
11010	011
11011	100
11100	011
11101	100
11110	100
11111	101

Priority Encoder

Input Word	Output Word
1xxxxxxx	000
01xxxxxx	001
001xxxxx	010
0001xxxx	011
00001xxx	100
000001xx	101
0000001x	110
00000001	111