Traineeships in Advanced Computing for High Energy Physics (TAC-HEP)

FPGA module training

Week-3

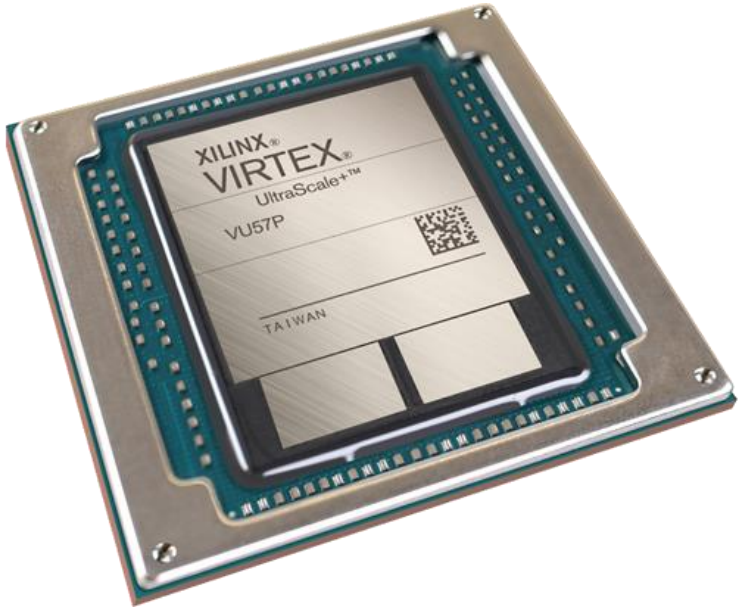
Lecture-6: February 13th 2025

Varun Sharma

University of Wisconsin – Madison, USA

Content

- Some concepts of hardware design
  - Clock Frequency, Latency, Pipelining
- Hardware Description Languages
  - Digital Gates, Verilog, VHDL



TAC-HEP 2025

Basic concepts of Hardware Design

FPGA Processing

FPGA: blank slate with a box of building blocks

- ***Vivado @ HLS compiler*** create processing architecture from the box of building blocks that best fits the software program

- The process of guiding the HLS compiler to create the best processing architecture requires fundamental knowledge about hardware design concepts

- In contrast, with a **processor**, computation architecture is fixed, and the job of compiler is to determine how to best fit the software application in the available processing structures

Clock Frequency

- **Important metric to determine the choice of processor**
- *In general*: High clock frequency means higher performance execution rate
  - Can be misleading

Maximum clock frequency:

FPGA	500 MHz
Processor	2 GHz

Which is better?

Stages		Description
IF	Instructions Fetch	Get the instruction from program memory
ID	Instruction decode	Decode the instruction to determine the operation and the operators
EXE	Execute	Execute the instruction on the available hardware
MEM	Memory Operation	Fetch data for the next instruction using memory operations
WB	Write Back	Write the results of the instruction to local registers/global memory

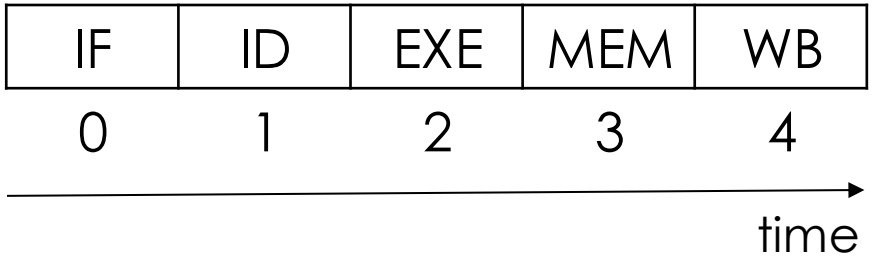
Regardless of processor type: Execution instructions

Clock Frequency

FPGA	500 MHz
Processor	2 GHz

4 times better?

- A **processor** is able to execute any program on a common hardware platform
- The compiler, which has a built-in understanding of the processor architecture, compiles the user software into a set of instructions

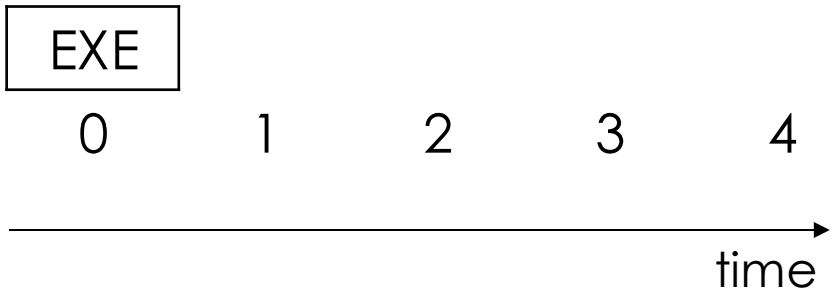


Processor instruction execution stages

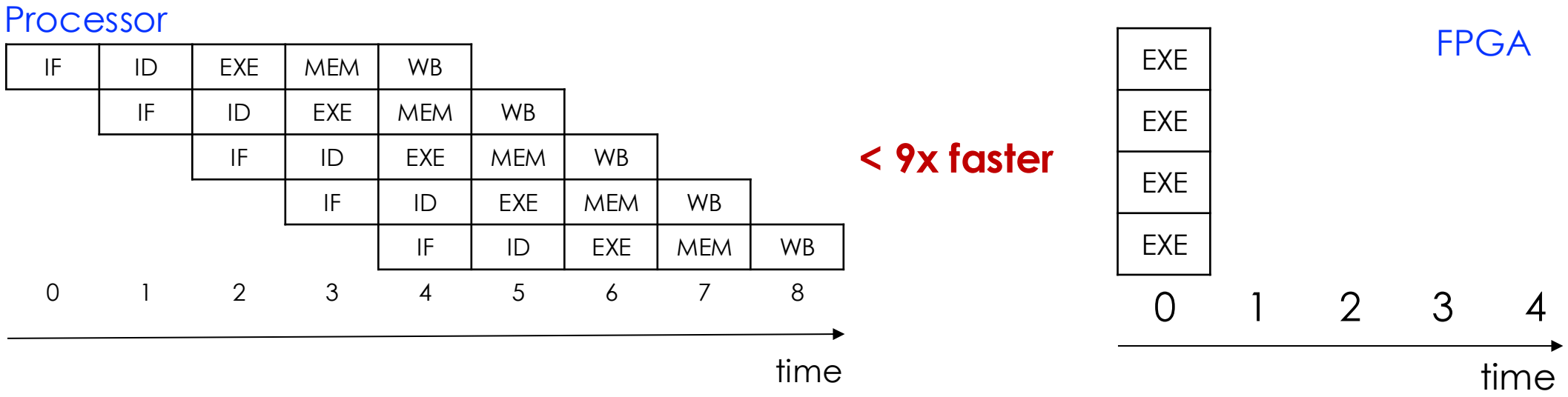
Clock Frequency: FPGA

- **FPGA** does not execute all software on a common computation platform.
- BUT executes on custom circuit for that program
  - Therefore, any modification to program changes the circuit in the FPGA.

- Vivado HLS compiler does not need to account for overhead stages in the platform
  - Can find ways of maximizing instruction parallelism.



Clock Frequency



Multiple instruction execution stages

FPGAs generally demonstrate at least 10x the performance

Approximate Power consumption = $\frac{1}{2}$ cF.V^2

FPGA is able to run at a lower clock frequency with maximum parallelism

- Thus lower power for same computational workload

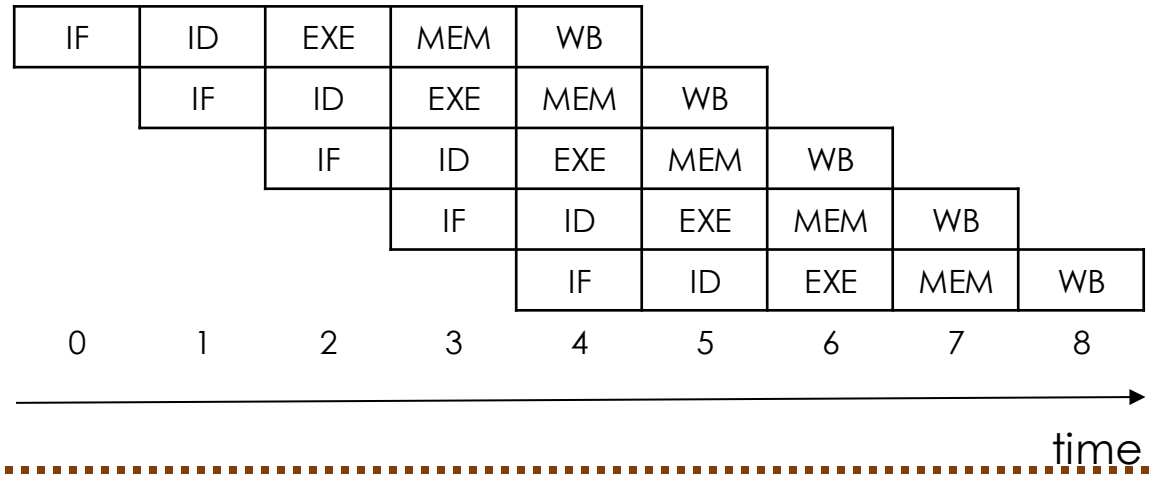
Latency and Pipelining

Latency: *number of clock cycles it takes to complete an instruction or set of instructions to generate an application result value*

Latency: **5 clock cycle**
For 5 set of instructions: **25 clock cycles**

Latency is another important key performance metric

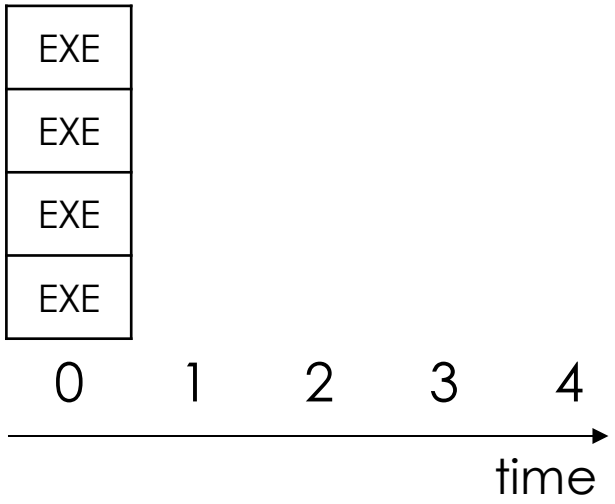
Latency can be improved via pipelining



Latency: **5 clock cycle**
For 5 set of instructions with pipelining
9 clock cycle

Latency and Pipelining

Parallelism also plays an important role in reducing latency



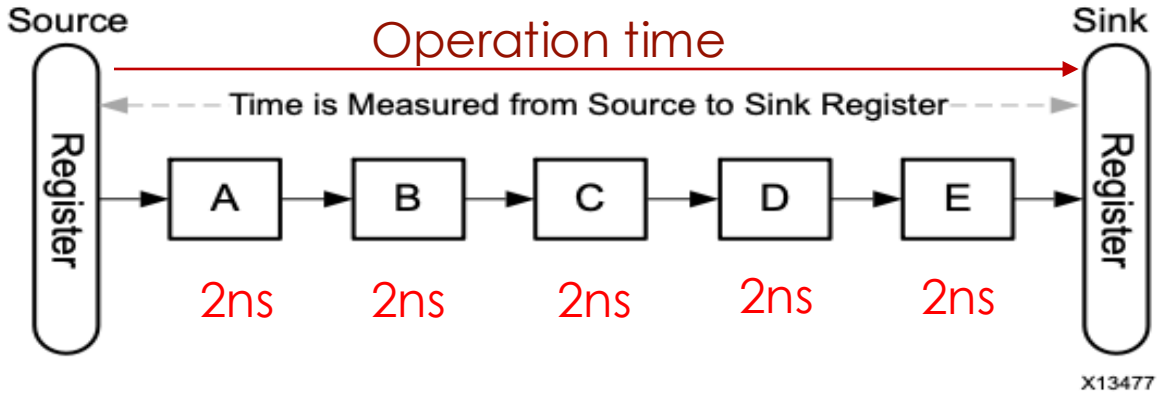
5 set of instructions
FPGA latency: 1 clock cycle

Do we need pipelining in One clock cycle latency of the FPGA?

- The reason for pipelining in an FPGA is to improve application performance

Latency and Pipelining

Reminder: FPGA is a blank slate with building blocks that must be connected to implement an application



FPGA implementation without pipelining

Example:

- Each block takes 2 ns to execute
- Current design (5 stages of implementation): 10 ns
- Latency: 1 clock cycle
- Clock frequency: $\frac{1}{5 \times 2\ ns} = \frac{1}{10\ ns}$ =100 MHz

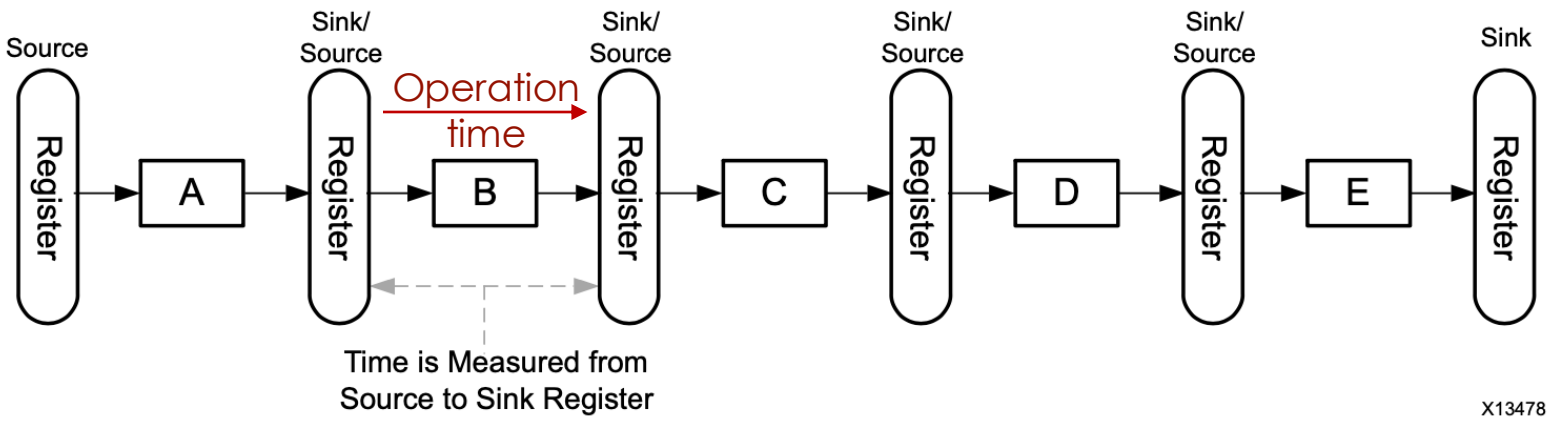
Clock Frequency: longest signal travel time between source and sink registers

Latency and Pipelining

Technique to avoid data dependencies and increase the level of parallelism

- Pipelining in an FPGA is the process of inserting more registers to break up large computation blocks into smaller segments.

- Partitioning of the computation increases the latency in absolute number of clock cycles but increases performance by allowing the custom circuit to run at a higher clock frequency



FPGA implementation with pipelining

Latency and Pipelining

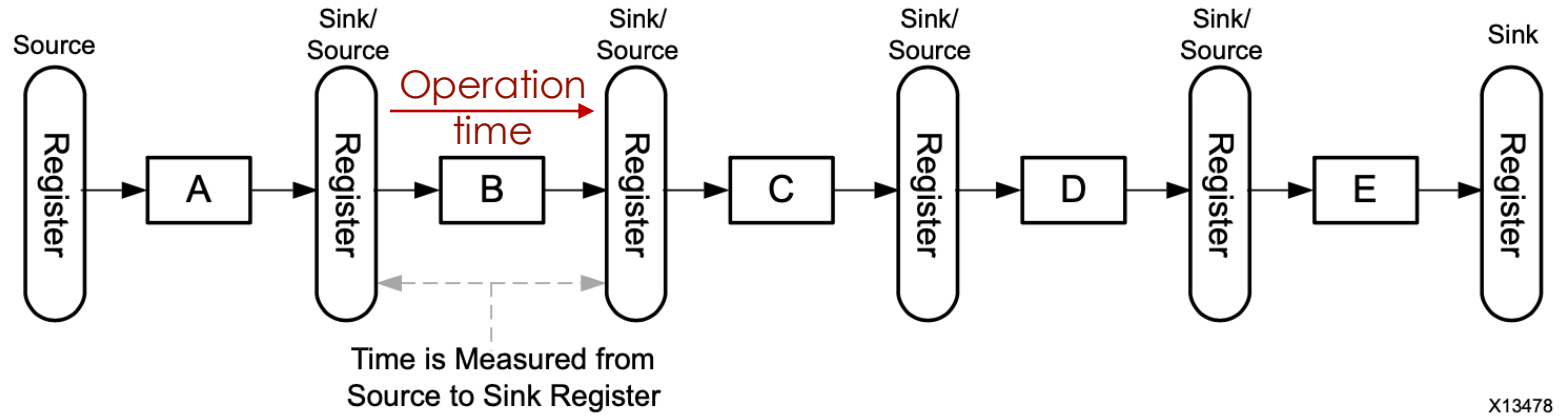


Fig. 12

FPGA implementation with pipelining

- Addition of registers reduces the timing requirement of the circuit from 10 ns to 2 ns,
- Results in a maximum clock frequency of 500 MHz.

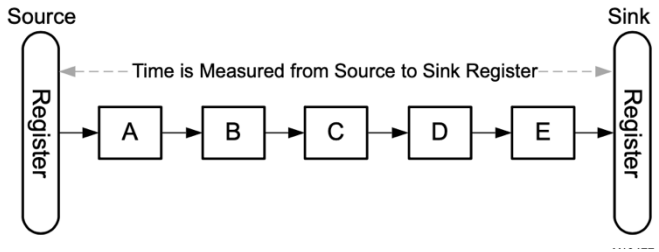
- In addition, by separating the computation into separate register-bounded regions, each block is allowed to always be busy, which positively impacts the application **throughput**

The latency caused by pipelining is one of the trade-offs to consider during FPGA design

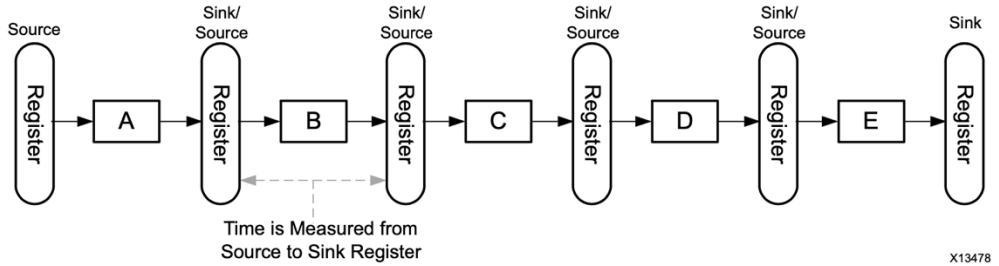
Throughput

- Another another metric used to determine overall performance of an implementation

- Number of clock cycles it takes for the processing logic to accept the next input data sample
- Throughput changes with clock frequency



10 ns between input samples



2 ns between input samples

Second implementation has higher performance, because it can accept a higher input data rate

TAC-HEP 2025

Digital Gates

What are FPGAs made up of?

- FPGAs contain **millions to billions of transistors**

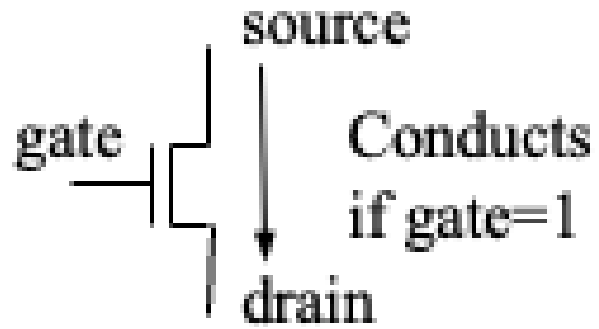
- Primarily used in three key components:
  - CLBs
    - LUTs use **transistors** to store truth tables and implement logic functions
  - Programmable interconnects
    - **Transistors** act as **switches** to route signals between logic blocks
  - I/O blocks
    - Use **transistors** to drive external signals into and out of the FPGA

- Xilinx's VU13P FPGA (16nm speed grade): ~35 B transistors

What is a transistor?

- **A fundamental electronic switch that controls the flow of electrical signals**
  - Basic electrical component in digital systems
  - Acts as an ON/OFF switch

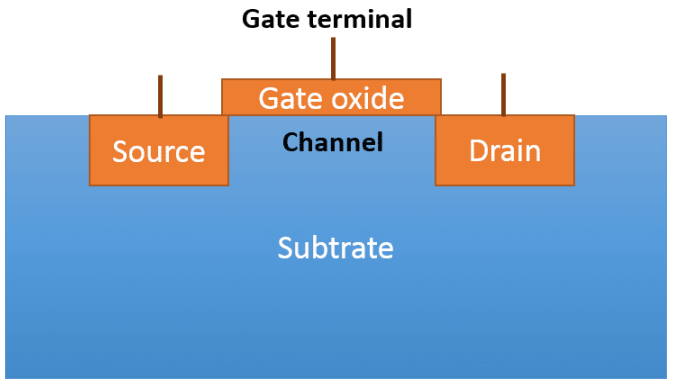
- **Different types of transistors:**
  - Bipolar Junction Transistor (BJT)
  - Field-Effect Transistor (FET)
    - Junction FET (JFET)
    - **Metal Oxide Semiconductor FET (MOSFET)**



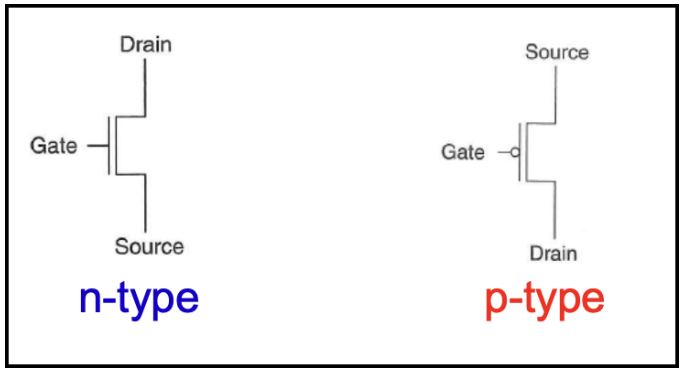
- Intel Core i9-10900k (10th Gen): ~2.5billion

MOS Transistor implementation

- **MOS transistors:**
  - Made by combining
    - Conductors (**M**etal)
    - Insulators (**O**xide)
    - **S**emiconductors



- **Two types of MOS transistor**
  - n-Type (nMOS) conducts if gate**=1**
  - p-Type (pMOS) conducts if gate**=0**
  - Hence "*Complementary MOS*"

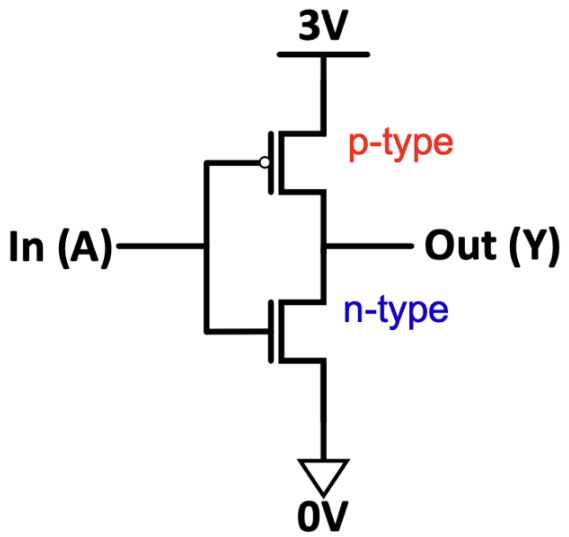


The two are combined to realize simple logic gates

Simplest Logic Structure

Modern computers use both n-type & p-type transistors:
CMOS technology: nMOS + pMOS

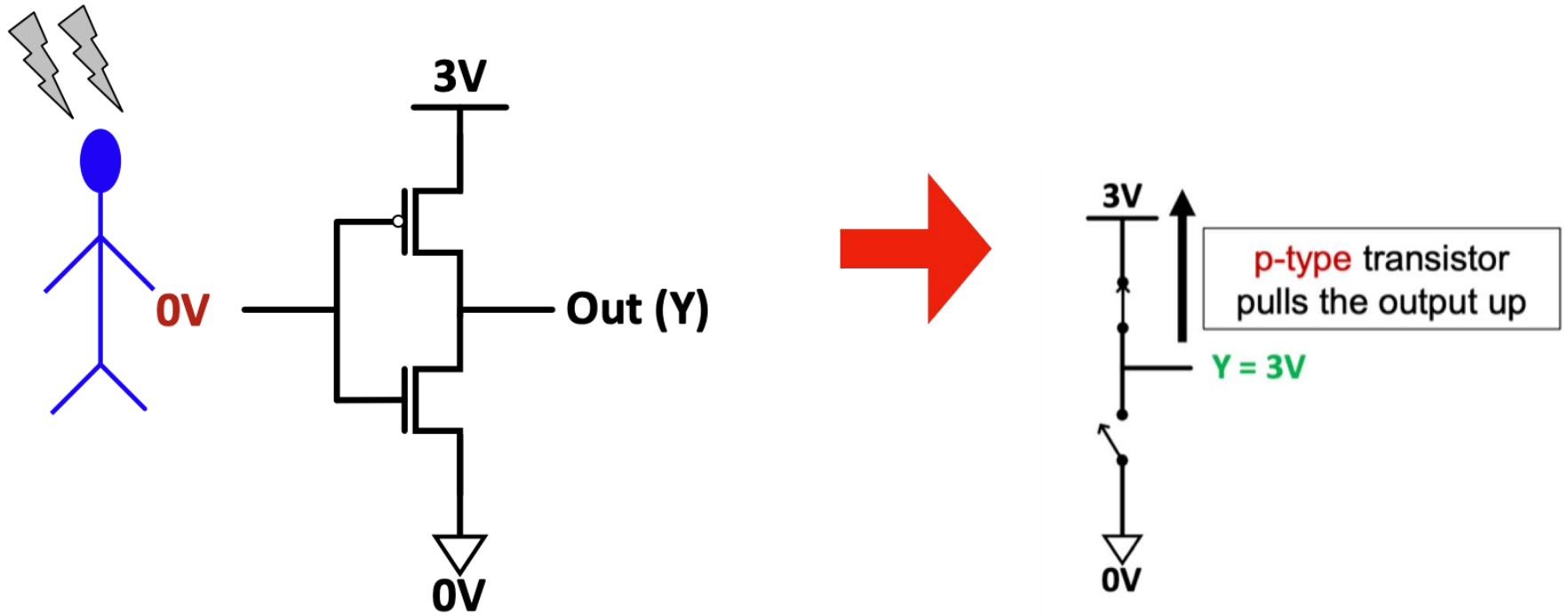
The simplest logic structure that exists:



What does this circuit do?

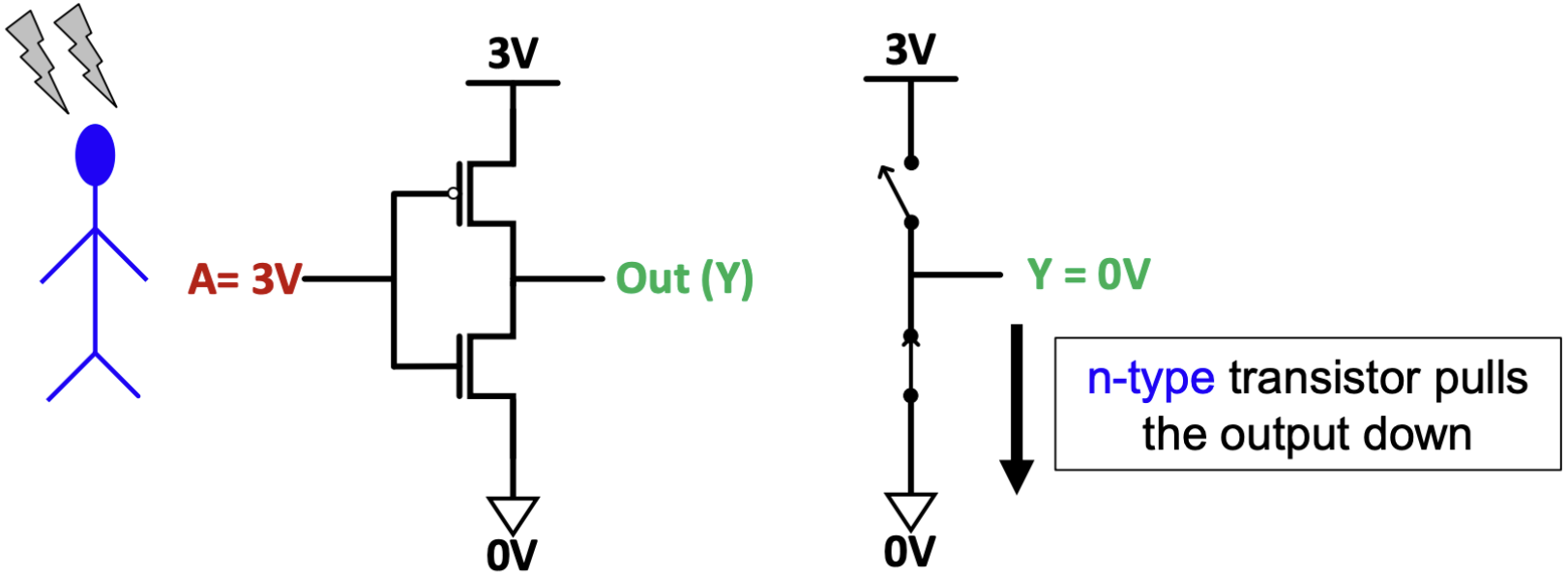
CMOS Functionality

What happens when the input is connected to 0V



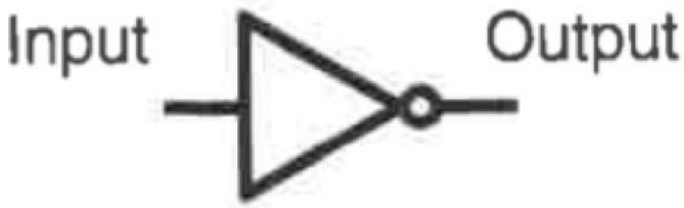
CMOS Functionality

What happens when the input is connected to 3V



CMOS NOT Gate

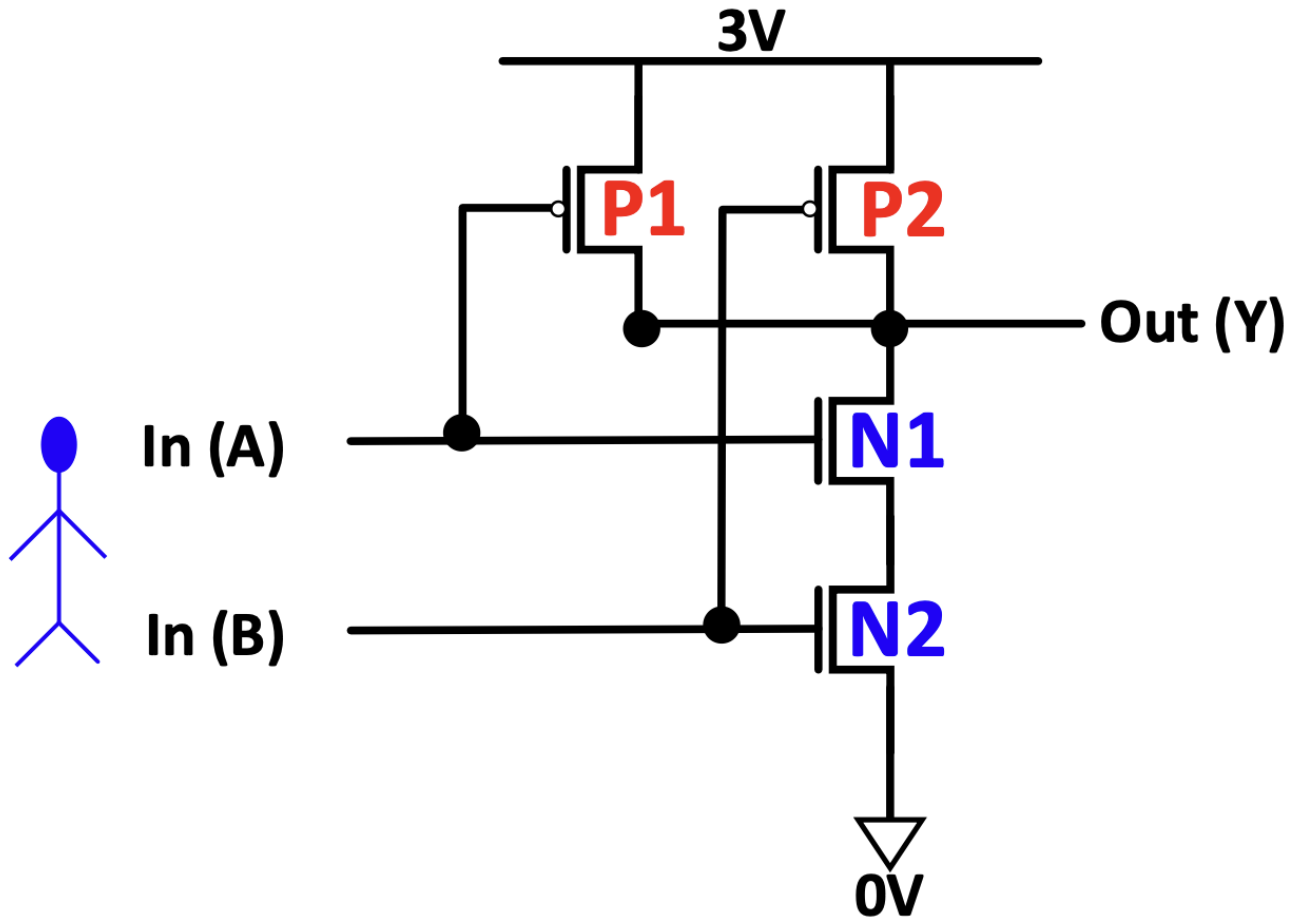
Not gate



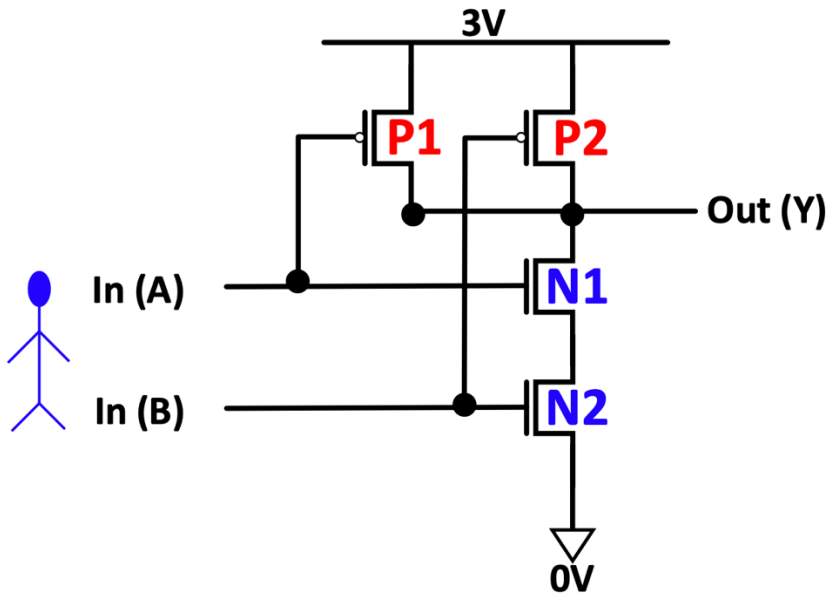
Truth table

Input	Output
L (0)	H (1)
H (1)	L (0)

A more complex gate



CMOS NAND Gate

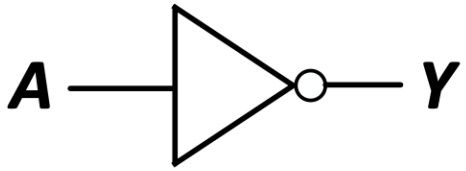


$$Y = \overline{A \cdot B} = \overline{AB}$$

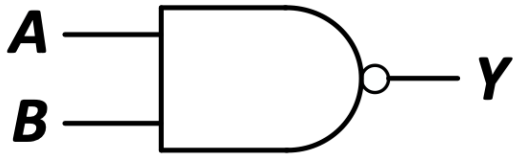
A	B	P1	P2	N1	N2	Y
0	0	ON	ON	OFF	OFF	1
0	1	ON	OFF	OFF	ON	1
1	0	OFF	ON	ON	OFF	1
1	1	OFF	OFF	ON	ON	0

- P1 and P2 are in parallel; only one must be ON to pull the output up to 3V
- N1 and N2 are connected in series; both must be ON to pull the output to 0V

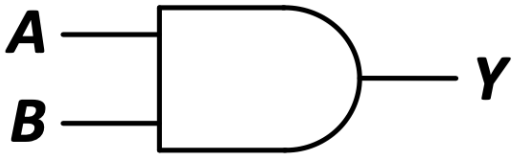
CMOS NOT, NAND, AND Gates



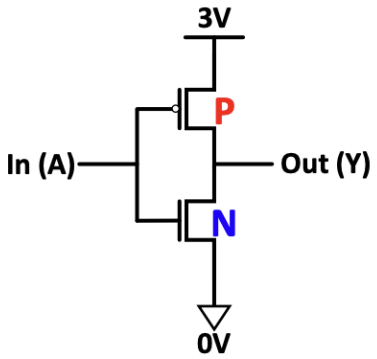
A	Y
0	1
1	0

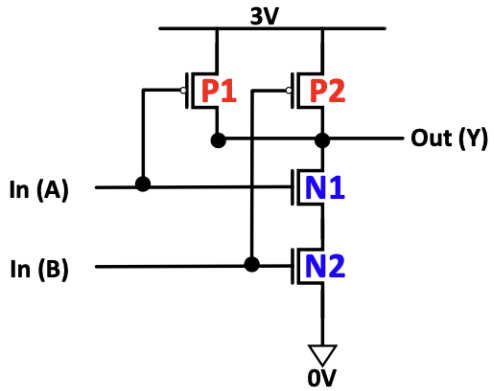


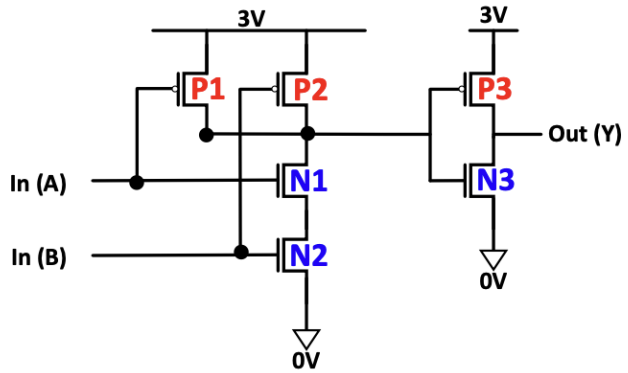
A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0



A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1



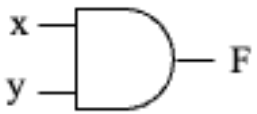




Summary of Logic Gates

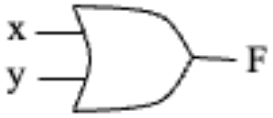
x	F
0	0
1	1

F = x
Driver



x	y	F
0	0	0
0	1	0
1	0	0
1	1	1

F = x y
AND



x	y	F
0	0	0
0	1	1
1	0	1
1	1	1

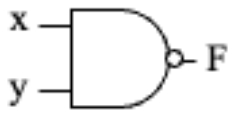
F = x + y
OR

x	y	F
0	0	0
0	1	1
1	0	1
1	1	0

F = x ⊕ y
XOR

x	F
0	1
1	0

F = x'
Inverter

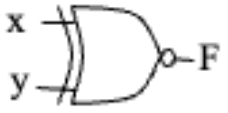


x	y	F
0	0	1
0	1	1
1	0	1
1	1	0

F = (x y)'
NAND

x	y	F
0	0	1
0	1	0
1	0	0
1	1	0

F = (x+y)'
NOR



x	y	F
0	0	1
0	1	0
1	0	0
1	1	1

F = x ⊙ y
XNOR

TAC-HEP 2025

Hardware Description Languages

Hardware Descriptive Language

- Specialized programming languages used to model, design, and simulate digital circuits
- **HDLs describe hardware behavior and its structure in textual form rather than sequential software execution**
  - To describe the circuits by syntax and sentences
  - As opposed to a circuit described by schematics

Different levels of description

- Behavioral level
- Register Transfer Level (RTL)
- Gate level
- Transistor level:

Widely used HDLs

- Verilog – Similar to C
- SystemVerlog – Similar to C++
- VHDL – Similar to Pascal

VHDL: VHSIC **H**ardware **D**escription **L**anguage
VHSIC: **V**ery **H**igh **S**peed **I**ntegrated **C**ircuit

Verilog HDL History

What is Verilog

- IEEE Industry Standard Hardware descriptive language
  - Used to describe a digital system
- Used in both hardware simulation & and synthesis

History

- Introduced in 1984 by **Phil Moorby** at **Gateway Design Automation**
- Purchased by Cadence in 1989
- In 1995 IEEE accepted OVI Verilog as standard
  - OVI: Open Verilog International
- Merged with SystemVerilog becoming IEEE standard 1800-2009
  - Includes object-oriented verification standard

VHDL

- Is an industry standard language used to describe hardware from the abstract to the concrete level
  - Standardized as IEEE standards 1076—1987, 1076-1993 & 1076-1164 (standard logic data type)
  - Specify the behaviour and structure of a digital circuit
  - Concurrent and sequential statements

- Powerful language with numerous language constructs capable of describing very complex behaviour
  - VHDL enforces strict typing rules, ensuring greater design accuracy.

Assignment-1

1. Compare FPGA with microcontrollers and ASICs and their real world applications
2. What are Look-Up Tables (LUTs) and how do they work in FPGAs?
3. Describe the use of Flip-Flops and Registers in an FPGA.
4. Compare different memories available on FPGA: SRAM, BRAM and URAM
5. Make CMOS NOT, NOR and OR gates, with corresponding truth table.

Uploaded to cernbox: https://cernbox.cern.ch/s/gmUqRDHTxDLqx4M

Submit in 2 weeks from now

TAC-HEP 2025

Questions?

Jargons

- **ICs - Integrated chip:** assembly of hundreds of millions of transistors on a minor chip
- **PCB:** Printed Circuit Board
- **LUT - Look Up Table aka 'logic'** - generic functions on small bitwidth inputs. Combine many to build the algorithm
- **FF - Flip Flops** - control the flow of data with the clock pulse. Used to build the pipeline and achieve high throughput
- **DSP - Digital Signal Processor** - performs multiplication and other arithmetic in the FPGA
- **BRAM - Block RAM** - hardened RAM resource. More efficient memories than using LUTs for more than a few elements
- **PCIe or PCI-E - Peripheral Component Interconnect Express**: is a serial expansion bus standard for connecting a computer to one or more peripheral devices
- **InfiniBand** is a computer networking communications standard used in high-performance computing that features very high throughput and very low latency
- **HLS** - High Level Synthesis - compiler for C, C++, SystemC into FPGA IP cores
- **HDL** - Hardware Description Language - low level language for describing circuits
- **RTL** - Register Transfer Level - the very low level description of the function and connection of logic gates
- **FIFO** – First In First Out memory
- **Latency** - time between starting processing and receiving the result
  - Measured in clock cycles or seconds
- **II - Initiation Interval** - time from accepting first input to accepting next input