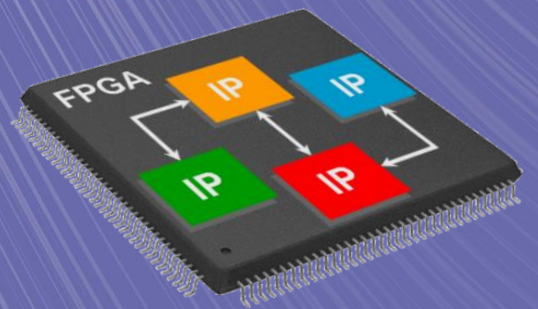
FPGAs

What are they?
How do they work?
What do we do with them today?
Where will they go tomorrow?



THOMAS FARMER, PHD
SCHOOL OF ENGINEERING AND APPLIED SCIENCE
UNIVERSITY OF PENNSYLVANIA
APRIL 9, 2025

OVERVIEW OF TALK

About the Author

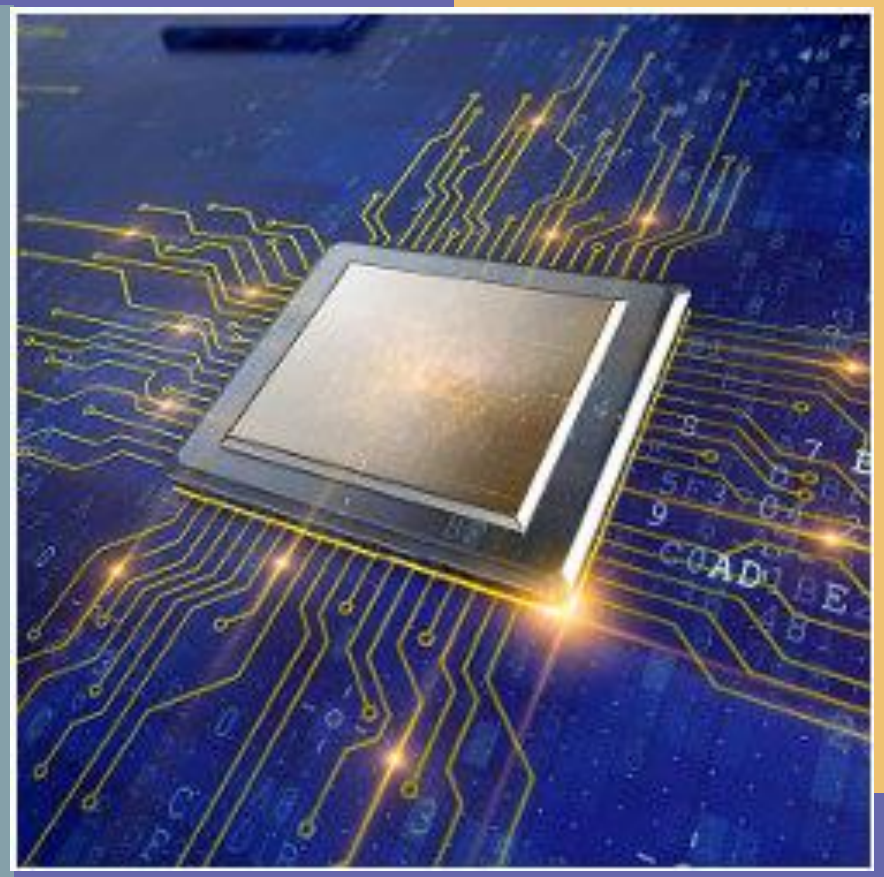
What is an FPGA?

What's inside an FPGA?

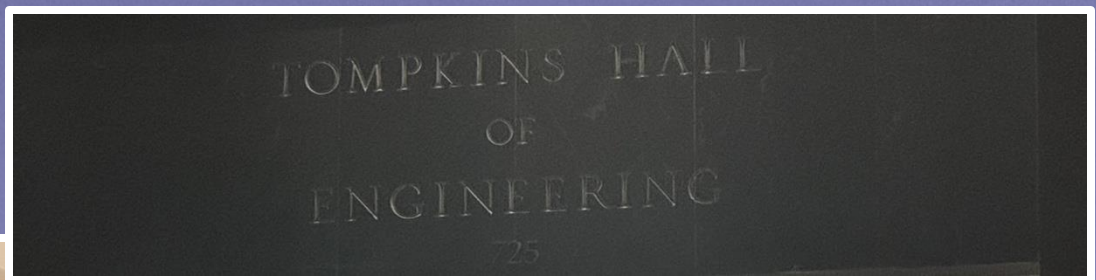
- *Transistors 101*
- *Digital Logic Overview*
- *LUTs and RAM*

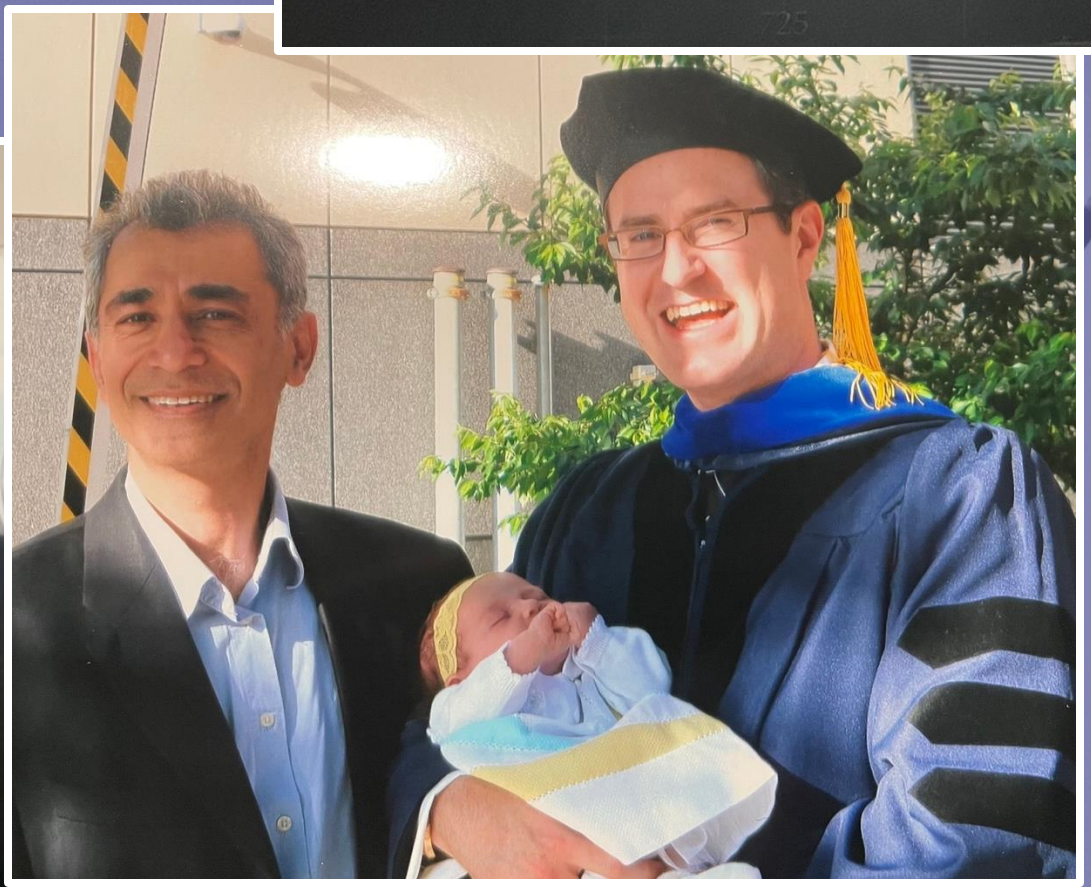
Programming an FPGA (HDLs)

FPGA Applications and Alternatives



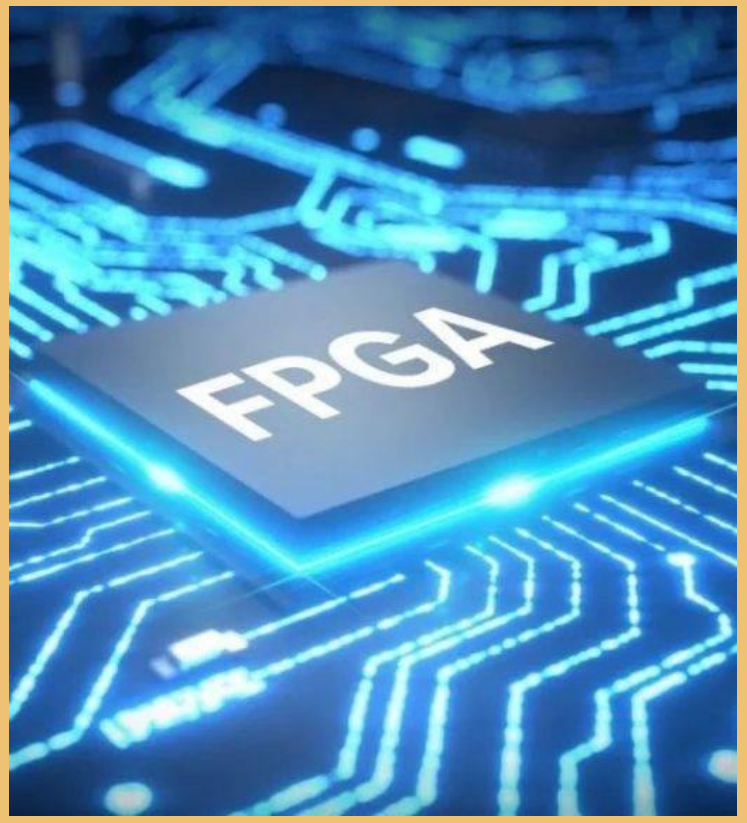
ABOUT THE AUTHOR





- *I LOVE GWU!*
- *Dr. Kartik Bulusu was our senior (and best) TA!*
- *Graduated in 2010, PhD in Comp. Engineering!*
- *Teaching Professor at Upenn since 2012*

WHAT IS AN FPGA?



- **FPGA – Field Programmable Gate Array**
  - Belong to class of devices known as: programmable logic
    - *AKA – programmable hardware*
- **Can be configured to be just about any digital circuit**
  - Simple things like: AND/OR gates/ Adders/Subtractors
  - Control Units – Dishwashers/Washing Machines/etc.
  - Microprocessor / CPU
    - *You can 'purchase' CPU IP and install on your FPGA*
  - Microcontroller
    - *A self contained microprocessor on a chip (Arduino)*
  - SoC (System on a Chip)
    - *Full blown mini-computer (Rasberry Pi)*
- **They can even be reprogrammed and their entire functionality changed!**

FPGA HISTORY

- An evolution from Programmable Logic Devices (PLDs) and programmable read-only memory (PROM)
  - *Could be factory/field programmed!*

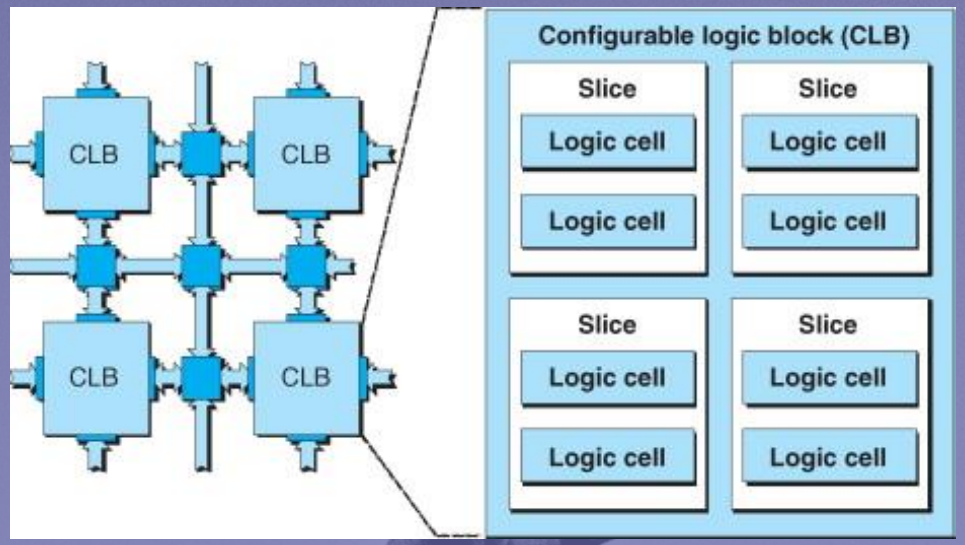
- Company: Altera, founded in 1983, released industry's first field re-programmable device in 1984 (now part of Intel)
- Company: Xilinx, first FPGA in 1985 (now part of AMD)
- Naval Surface Warfare Center, patented similar device in 1987
- Altera & Xilinx grew unchallenged until mid 90's
  - Actel/Microchip, Lattice Semi, Achronix
  - Current leaders: Intel and AMD

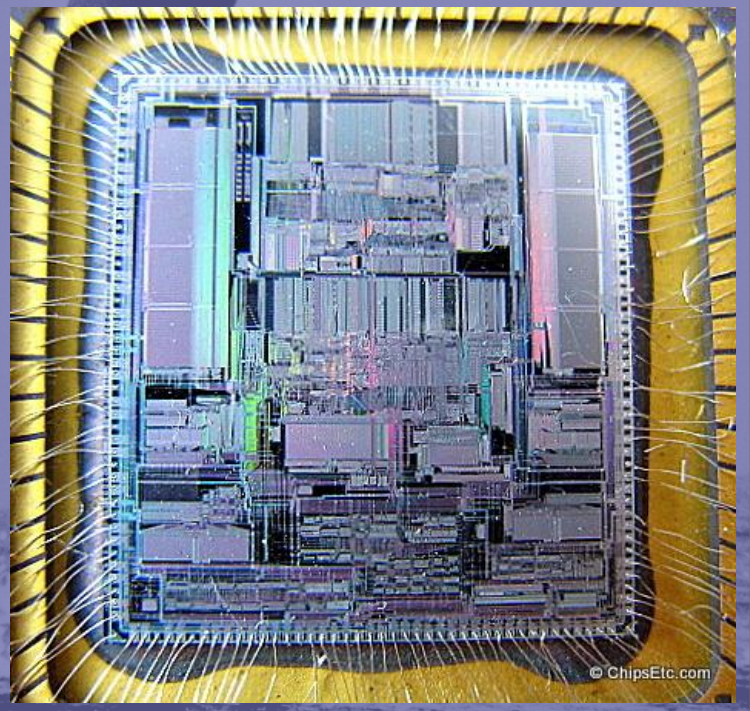
- Market Size:
  - 1987: $14 million,
  - 2030: est. $23.24 billion!
- Number of "gates" inside an FPGA:
  - 1987: 9k, 1992: 600k, 2013: 50 million

WHAT'S INSIDE AN FPGA?

A matrix of:

- Configurable logic blocks (CLBs) / Logic Block Arrays (LABs)
- Interconnects
- Input/Output (I/O) Blocks
- Digital Signal Processing Blocks
- Embedded Memory
- Hardblocks
- Transceivers



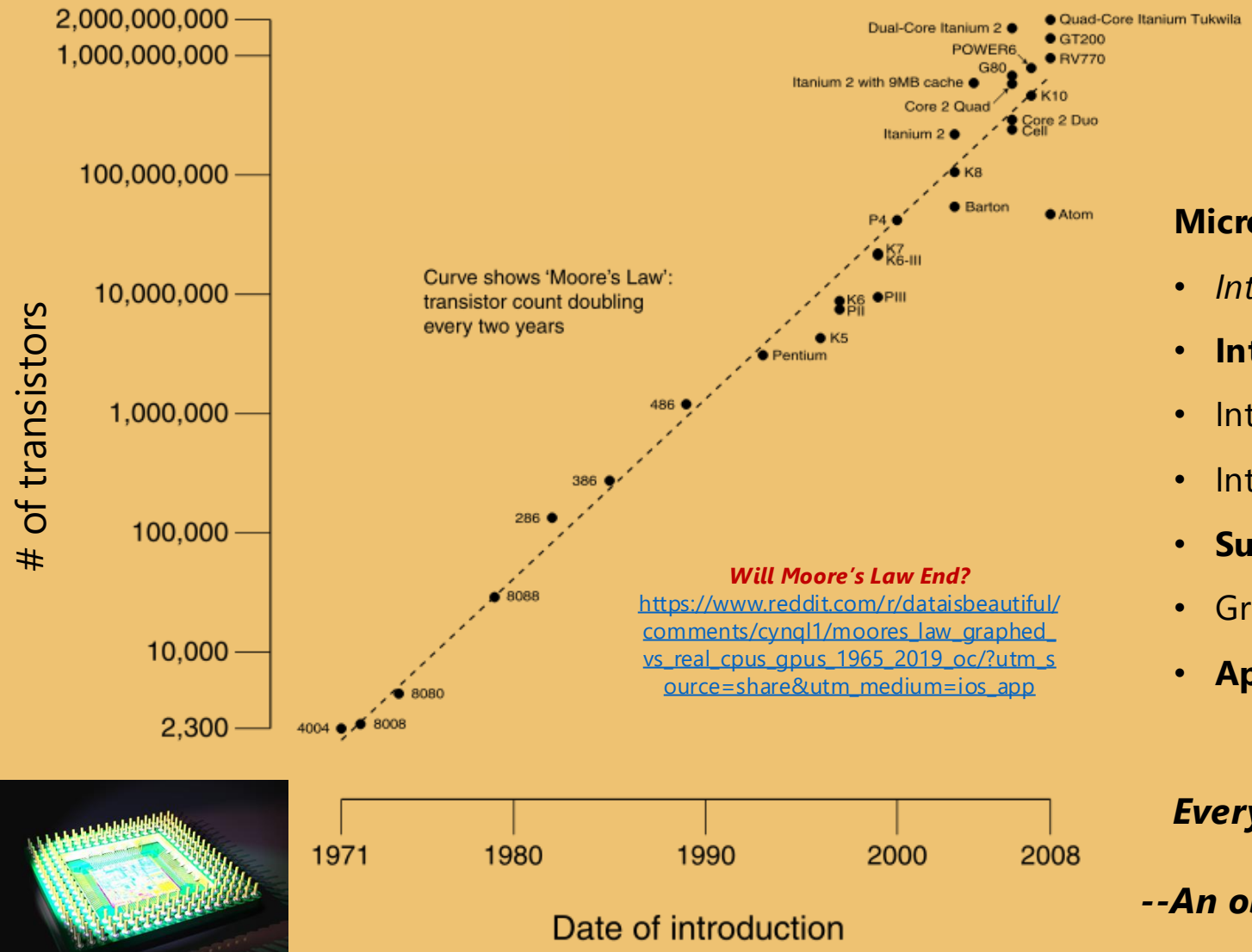


At its heart an FPGA is a massive Integrated Circuit (IC):

Intel's latest Stratix FPGA: 43.3 Billion **TRANSISTORS!**

So what exactly is a transistor? What is digital logic?

TRANSISTORS:
BUILDING BLOCKS OF COMPUTERS



Will Moore's Law End?
https://www.reddit.com/r/dataisbeautiful/comments/cynql1/moores_law_graphed_vs_real_cpus_gpus_1965_2019_oc/?utm_source=share&utm_medium=ios_app

1947 – AT&T invents transistor
Shockley

1958 – TI invents Integrated Circuit (IC)
Jack Kilby

1965 – Moore's Law is Coined!
Gordon Moore (1st CEO Intel)

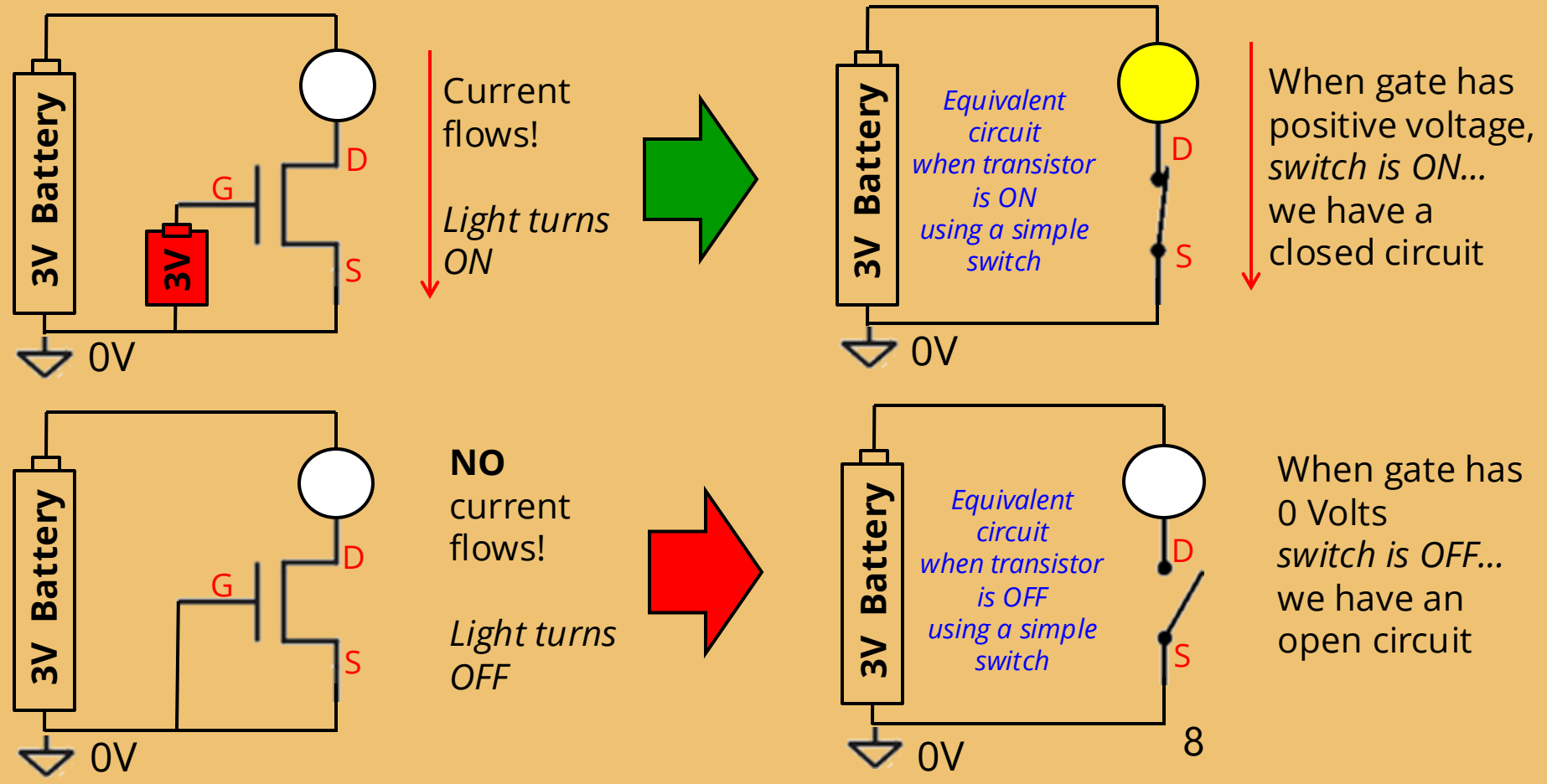
Microprocessors contain billions of *transistors*

- *Intel 4004 (1971): 2,300 transistors*
- **Intel 80486 (1989): 1,180,235 transistors**
- Intel Pentium 4 (2000): 48 million
- Intel Core Duo 2 (2006): 291 million
- **Sun's Sparc T3 16-core (2010): 1 billion**
- Graphcore GC2 IPU (CPU): 23.6 billion
- **Apple's M3 (2025): 184 billion**

Moore's Law:
Every 18 months, # of transistors fit into an IC will double.
--An observation made by Gordon Moore – co-founder of Intel

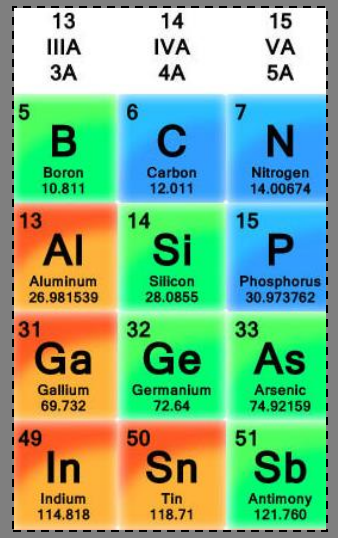
WHAT IS A TRANSISTOR?

- An electrical device that acts as an **electrical switch**; it's typically made from Silicon
  - 3 electrical contacts/terminals: Gate, Drain, Source
  - *Gate* controls flow of current between *Drain* and *Source* terminals; this style transistor is called a MOSFET

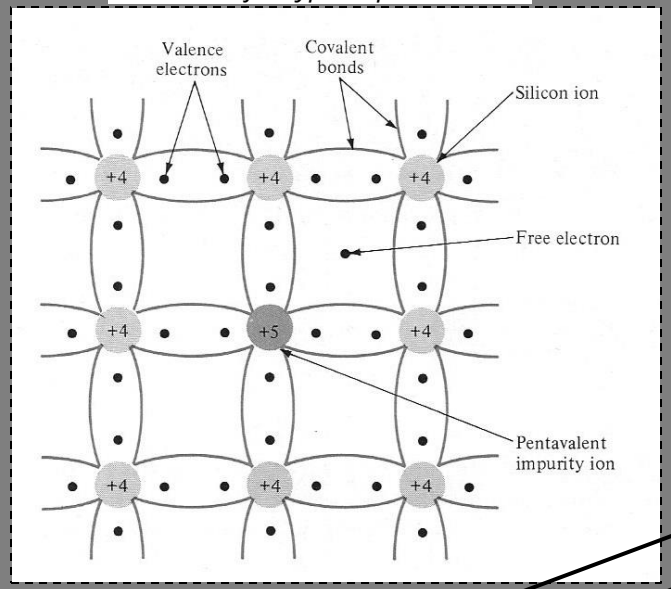


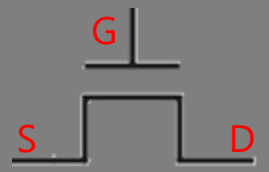
HOW TRANSISTORS ARE BUILT

- Transistors are built from Silicon. Make it a better conductor by "doping" it
  - N-type Silicon: "doped" with atoms from column V *(P, As, Sb)*
    - *Gives it an extra electron!*
  - P-type Silicon: "doped" with atoms from column III *(Al, Ga, In)*
    - *Removes electrons – creates "holes" in lattice*
- Transistors are formed from p & n-type "doped" Silicon

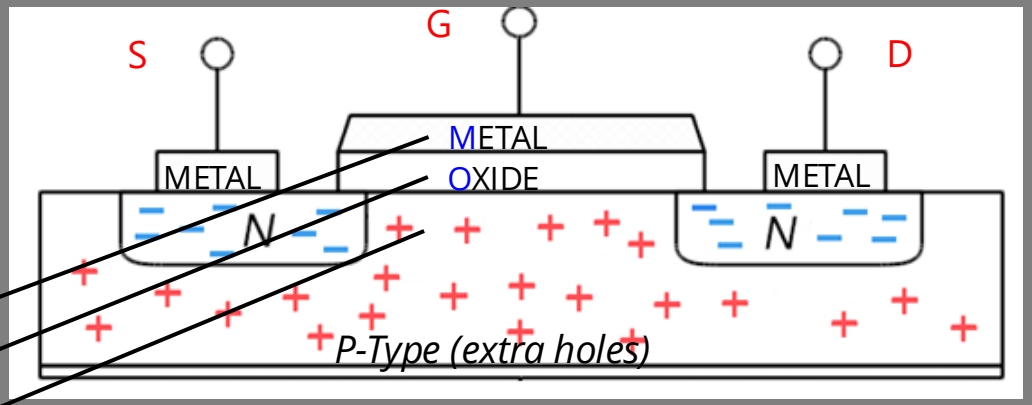


A lattice of n-type doped silicon





Symbol for an n-type MOSFET Transistor



Internal Structure of an n-type MOSFET Transistor

MOS – Metal Oxide Semiconductor

HOW DOES A TRANSISTOR WORK?
(When It's On)

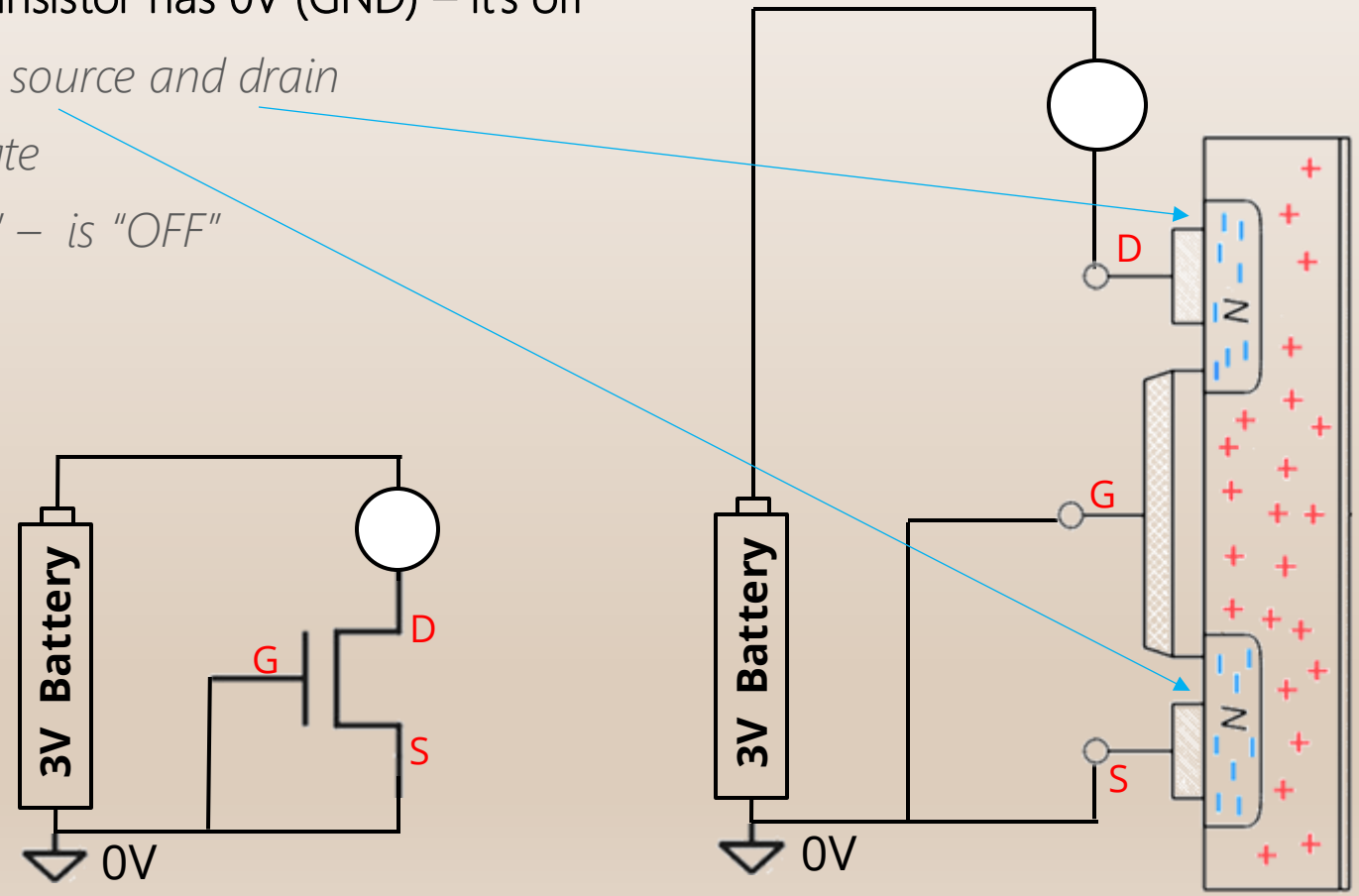
When the gate on an nMOS transistor has 0V (GND) – it's off

Electrons cannot move between source and drain

no path exists under the gate

The transistor – aka "the switch" – is "OFF"

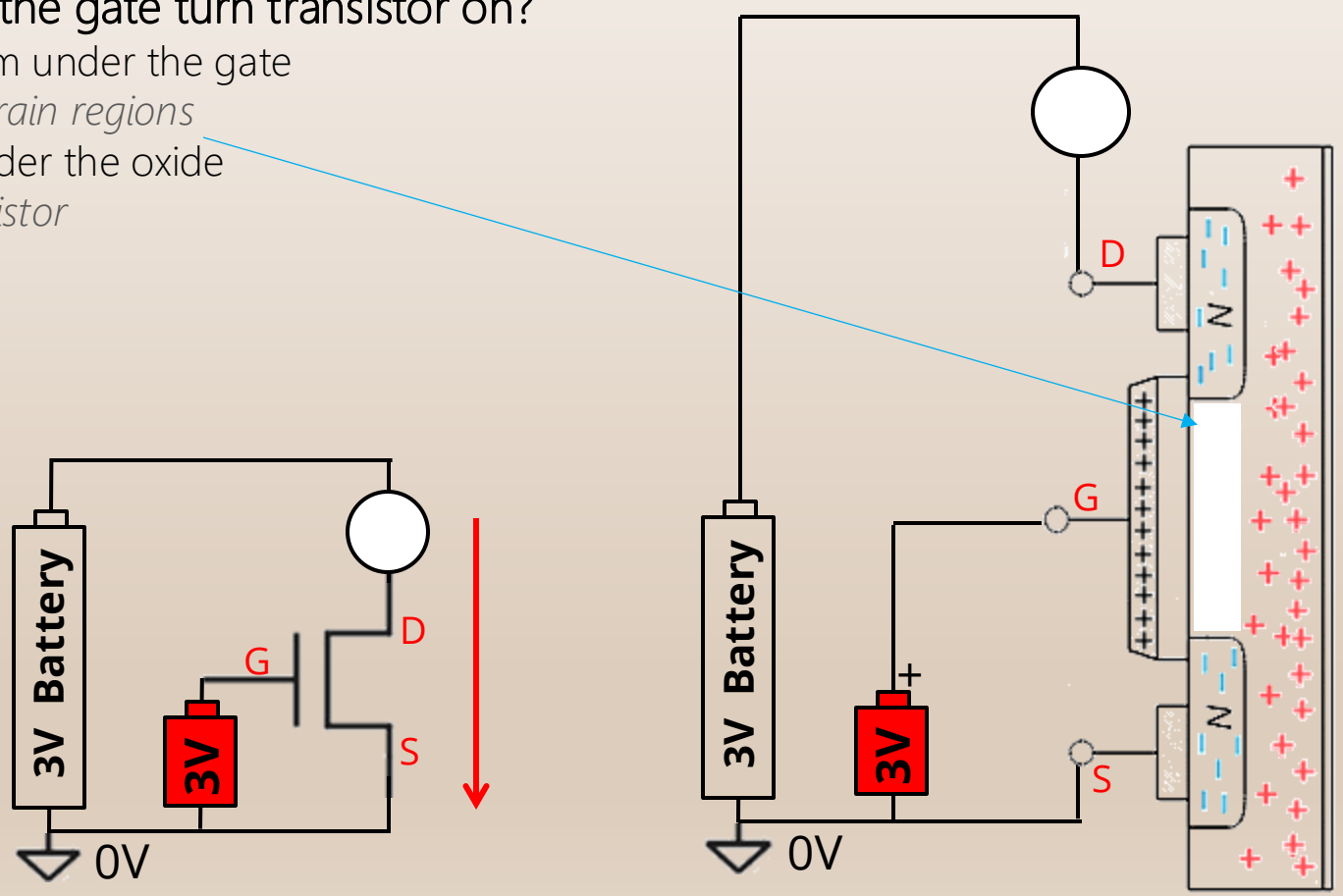
(the light stays off)



HOW DOES A TRANSISTOR WORK?
(When It's On)

Why does a positive charge on the gate turn transistor on?

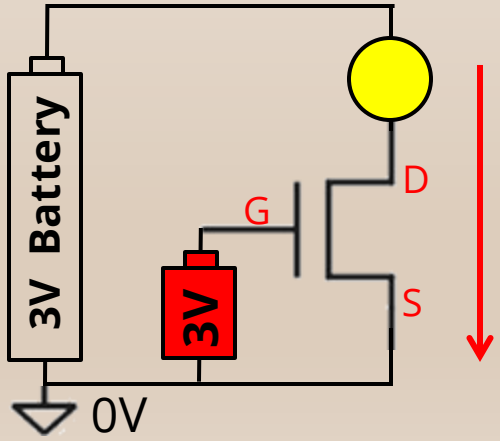
1) Positive charge repels holes from under the gate
   attracts electrons from source/drain regions
2) Creates an "n-type" channel under the oxide
   why we call this an nMOS transistor

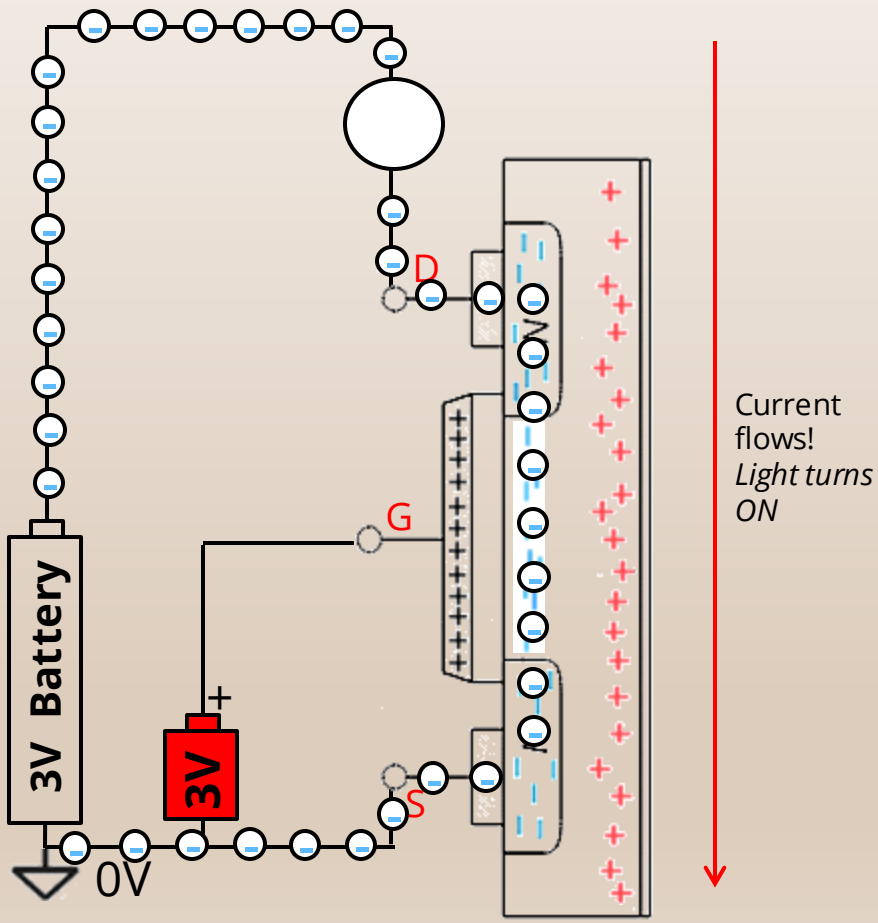


HOW DOES A TRANSISTOR WORK?
(When It's On)

Why does a positive charge on the gate turn transistor on?

1) Positive charge repels holes from under the gate
   attracts electrons from source/drain regions
2) Creates an "n-type" channel under the oxide
   why we call this an nMOS transistor
3) Current can flow from drain to source
   electrons enter source, exit drain
4) Electrons cannot penetrate oxide
   electric field forms across oxide; hence "FET"





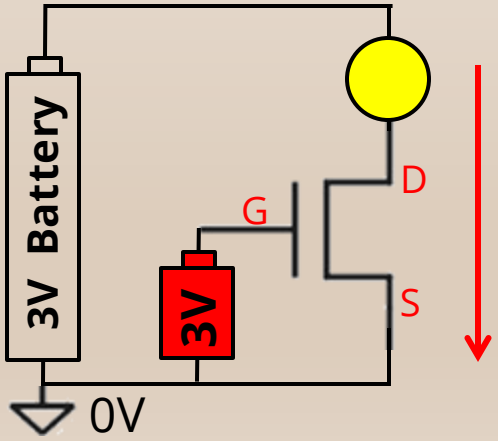
HOW DOES A TRANSISTOR WORK?
(When It's On/Off)

When gate on nMOS transistor has "positive" charge
(aka "High Voltage") on gate

A "channel" is formed between source and drain,
electrons flow (so current flows!)

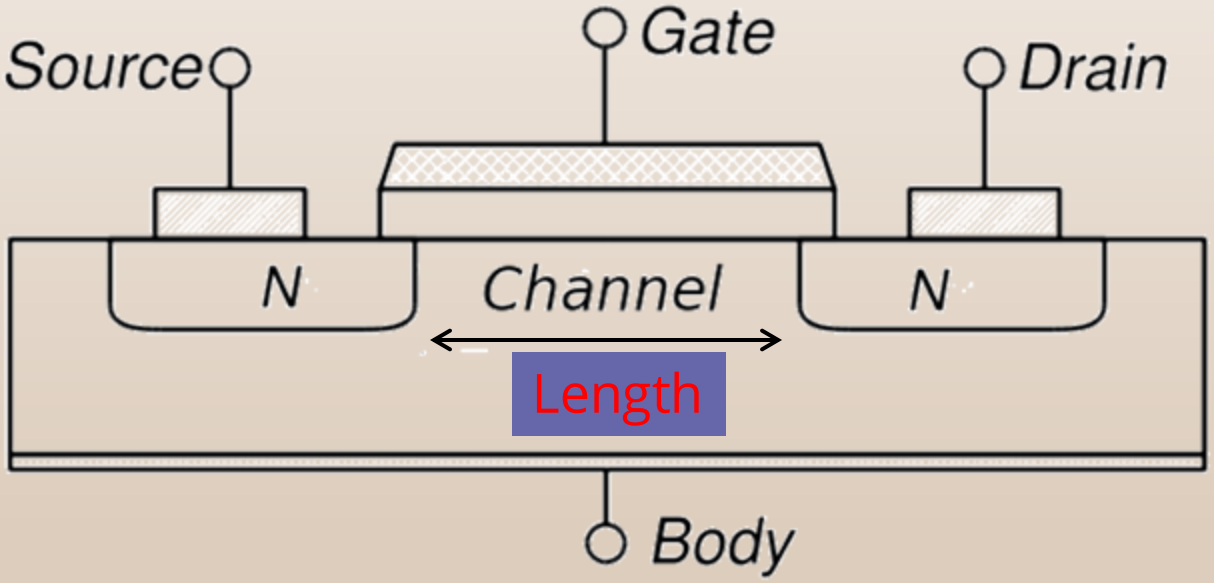
The transistor – aka "The Switch" – is "ON", and so is the light!

When gate on nMOS transistor has no charge (0V) – It's off



SPEED OF TRASISTOR *(MOSFET TYPE)*

- Dependent on many factors, 1 crucial factor: Length of Channel
  - ➢ Why? Electron takes less time to travel across smaller distance!
    - Currently, channels have lengths < 5 nm! *Note: Human hair is ~80,000 nm wide!*

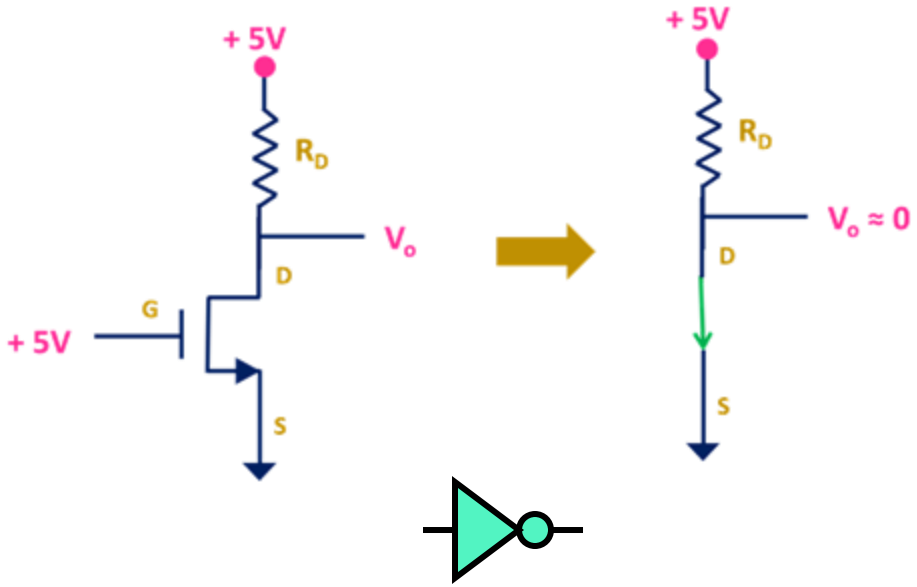


Moore's Law: Every 18 months, # of transistors fit into an IC will double

BECAUSE length of the channel will halve every 18 months

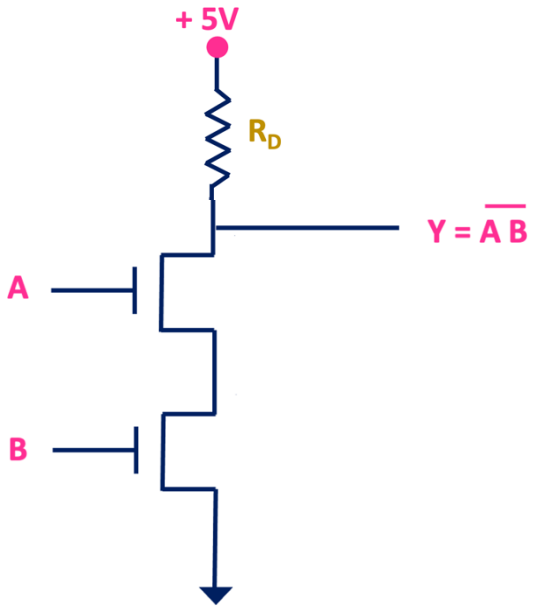
DIGITAL LOGIC 101: TRANSISTORS TO GATES

NOT/INV GATE

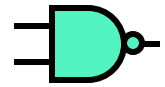


Digital Inverter: Truth Table

IN	OUT
0 (0 V)	1 (+5V)
1 (+5V)	0 (0 V)

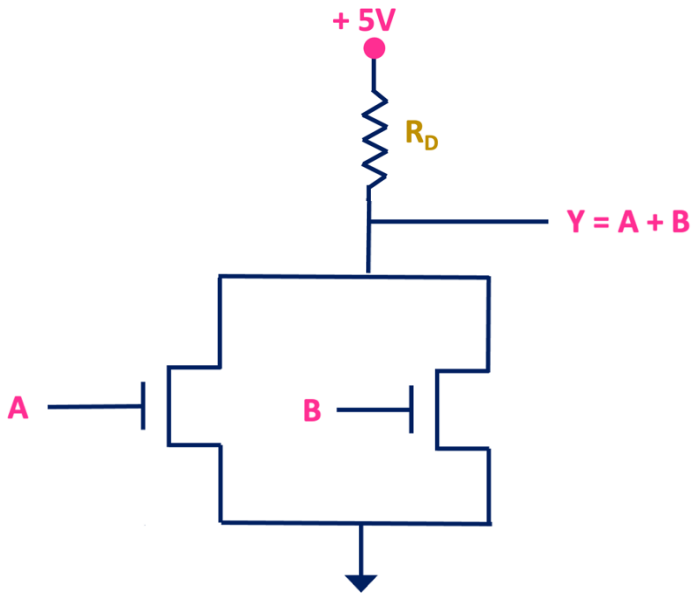


NAND GATE

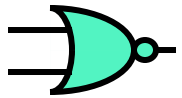


Truth Table

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



NOR GATE

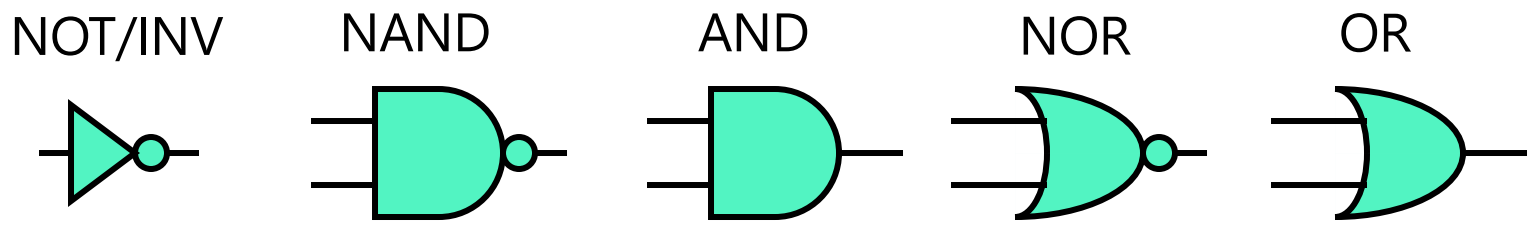


Truth Table

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

BASIC DIGITAL LOGIC GATES

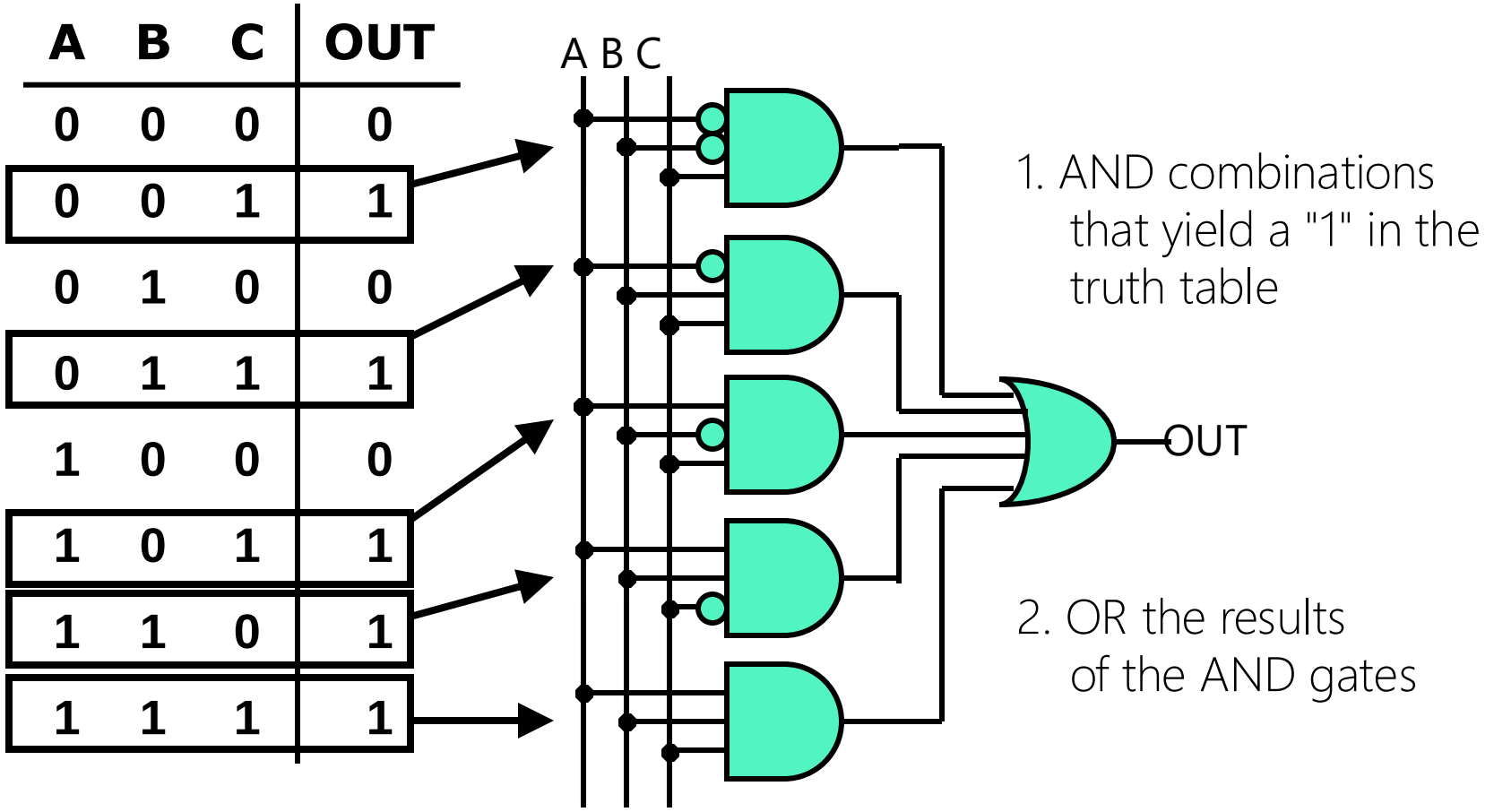
Basic Logic Gates:



- We can put these together to make any complex Boolean function
  - *Using only an INV / AND / OR we can make any other Boolean function (logically complete!)*
- We could design only at the “logic gate level;” not always the most efficient
- When you hear an FPGA has 50 Million Gates, these are the things we’re talking about
  - *Gates are made of transistors, # of transistors per gate varies for different functions*

GATES TO ARRAYS – PROGRAMMABLE LOGIC ARRAY (PLA)

- AND, OR, NOT can implement ANY truth table

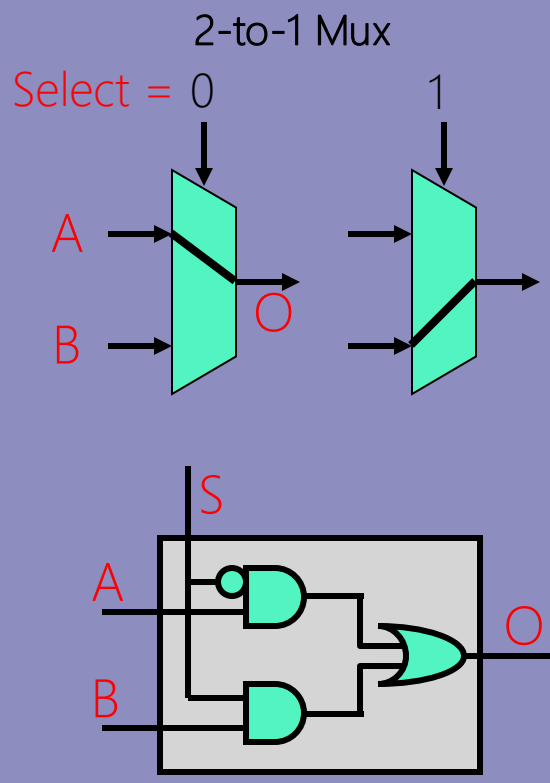


PLAs are the predecessors of FPGAs!
Represent the universal nature of GATEs to implement any digital logic function

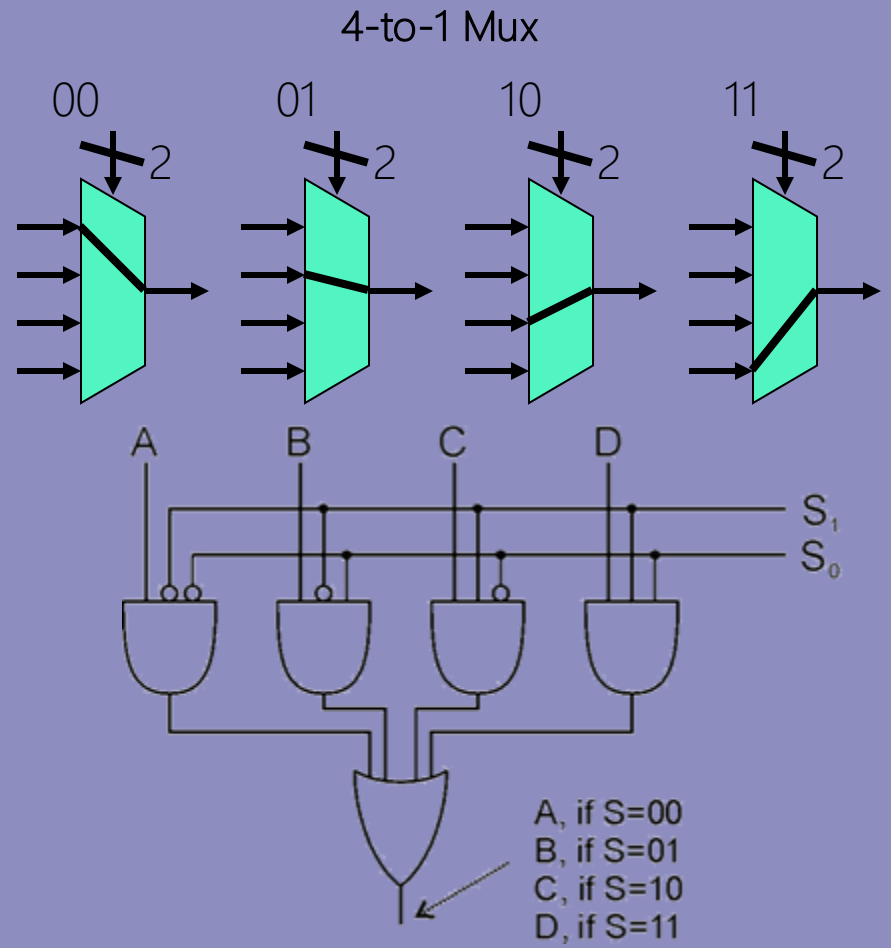
THE MULTIPLEXER (MUX) – "THE SELECTOR"

The representation of "choice" in hardware…*if (condition) then (action)*

- They are like a railroad switch, allows us to choose different signals to "route" around an FPGA



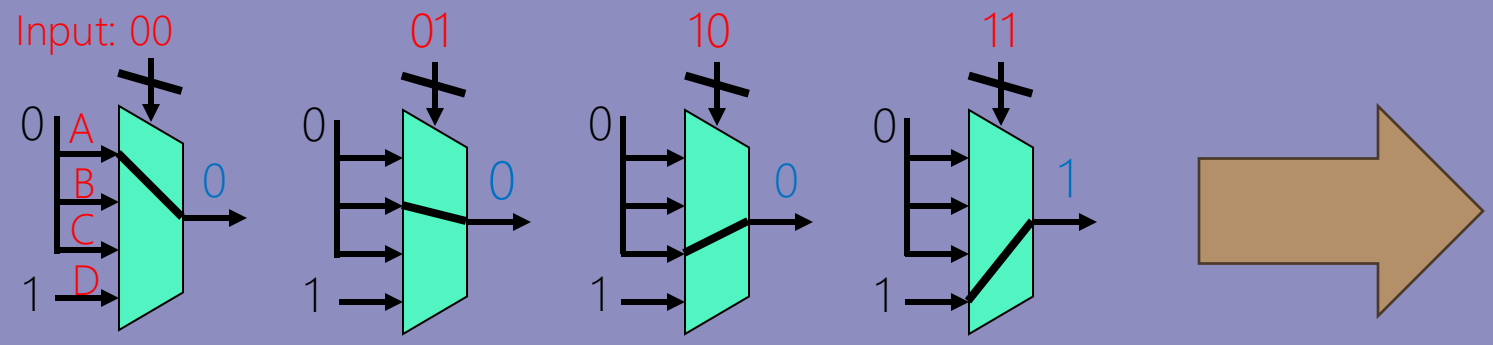
Input "S" *selects* A or B to attach to "O" *output*
Acts like an "IF/ELSE" statement

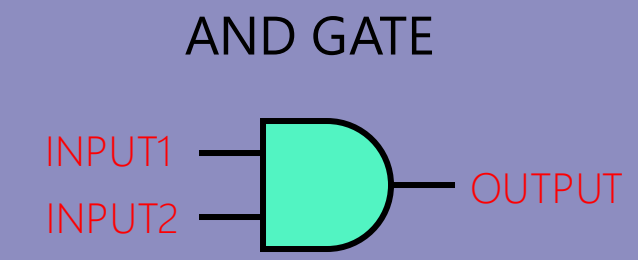


THE MULTIPLEXER (MUX) – THE UNIVERSAL GATE

FPGAs use MUXs to implement "any" digital logic gate: AND/OR/NOT

4-to-1 Mux – Acting as an AND Gate





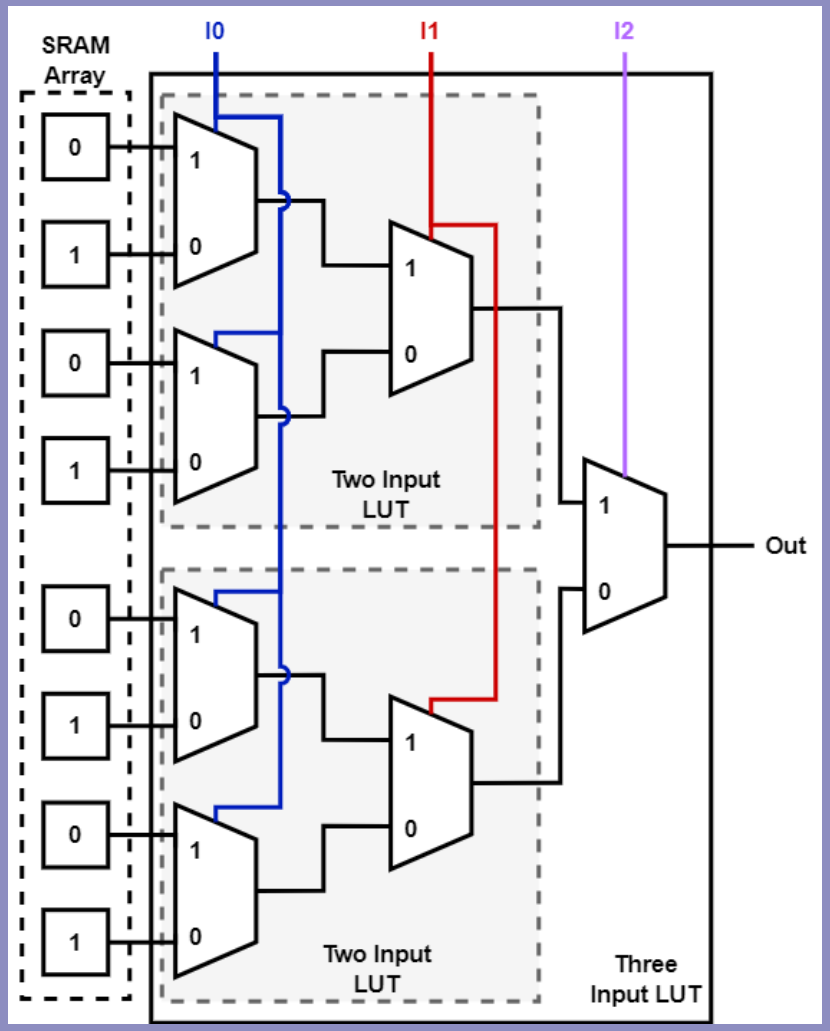
INPUT1	INPUT2	OUTPUT
0	0	0
0	1	0
1	0	0
1	1	1

- By setting inputs A, B, C, and D to "0" or ground, and D to "1" or 5V
  - The "select" line behaves as the input line to a typical AND gate
- Notice the outputs correspond to the rows of an AND gate truth table!
- We can set inputs ABCD to different values to obtain OR/NAND/NOR!

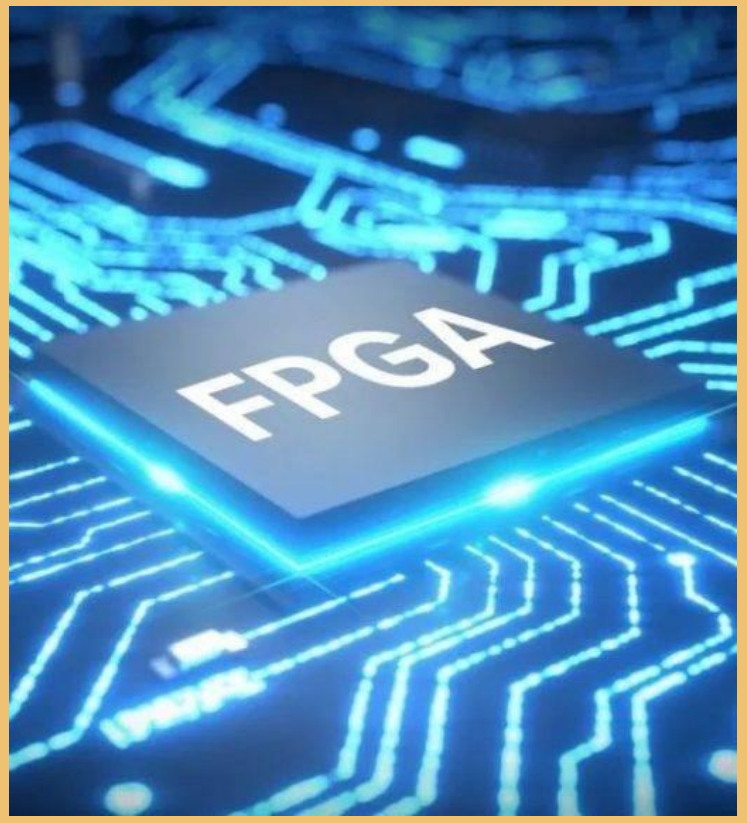
SRAM & MUX – CREATING A LOOK UP TABLE (LUT)

Attach SRAM to MUX's and we can "program" the MUX to behave as any gate

- All depends on the value in SRAM's
  - We can then create any digital logic circuit
- We can even use other MUX's to 'connect' …
  - …the output of one LUT to another LUT
- By setting the values in SRAM, we can setup any gate
- As well as connect or "route" LUTs together to create more complex logic circuits!
- When we say an FPGA can be "programmed" we are talking about populating SRAM with the right values to implement our digital logic circuit
- Luckily we don't work at the "bit level" we have a special language that translates into the SRAM values; it's called an "HDL" Hardware Definition Language



WHAT IS AN HDL?

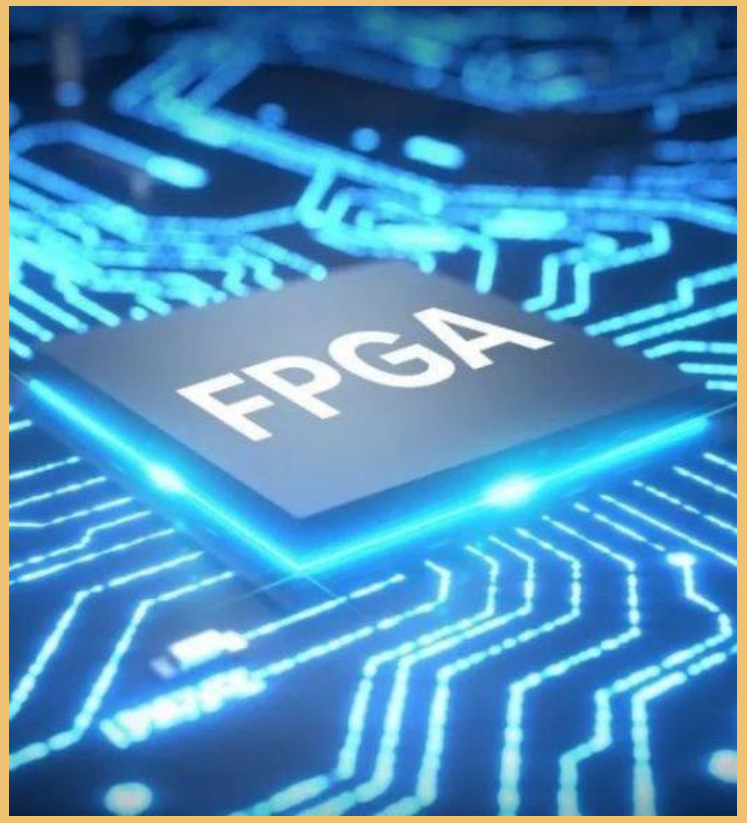


- **Hardware Definition Language (HDL)**
  - A special language that we can use to describe hardware
    - *We can "compile" it (aka – translate it), into the*
    - *1s and 0s that will populate the SRAM on an FPGA*
  - *Two popular versions: Verilog and VHDL*
- **Example of HDL to make an AND gate:**

```
module and_gate (
          input a,
          input b,
          output c );

   assign c = a & b; // we describe the behavior

endmodule
```

- **Modules are like "functions" in traditional languages**
- **But we think of them as hardware modules**

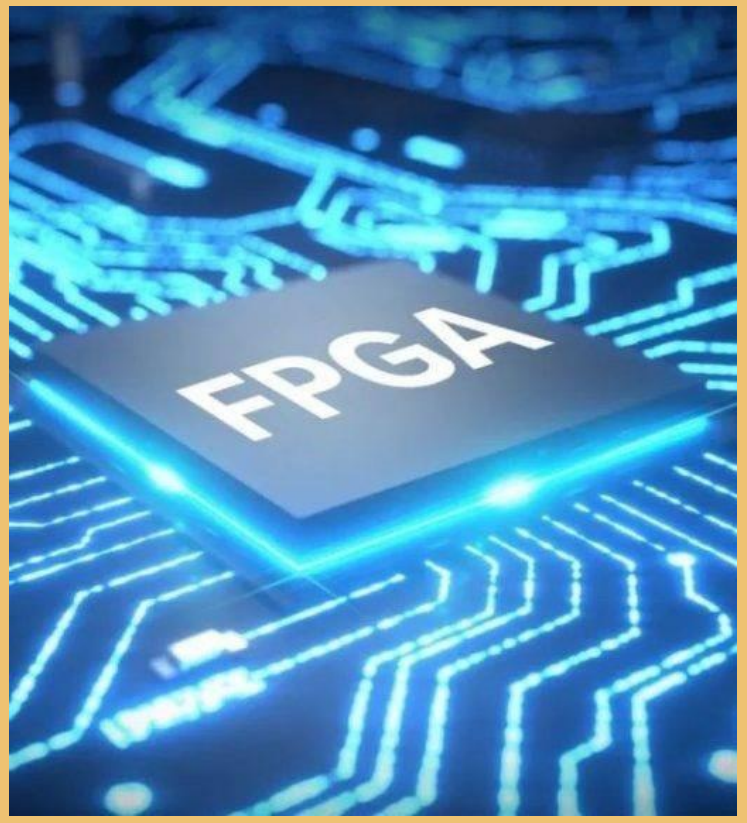
HDL EXAMPLE 2



- **Example of making a counter in HDL:**

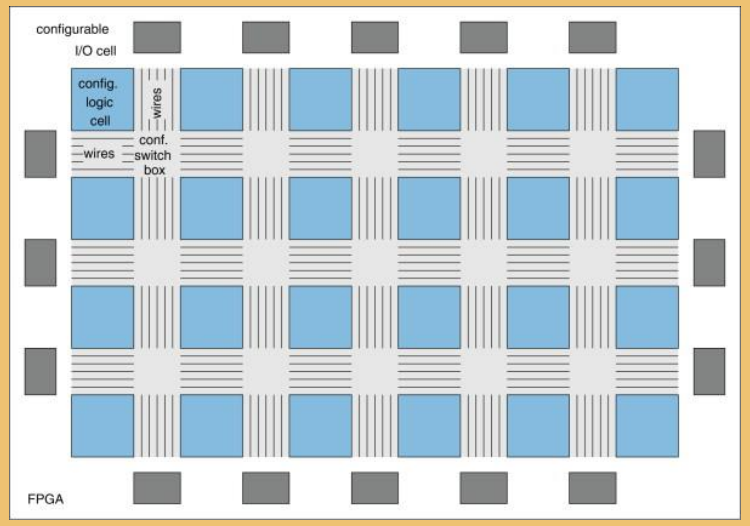
```
module counter (
input clk,
input rst,
output reg [3:0] count
);

    always @(posedge clk or posedge rst) begin
       // Triggered by clock or reset

       if (rst) begin
          count <= 4'b0; // reset count to 0000
       end else begin
          count <= count + 1; // increment counter
       end

    end


endmodule
```

LET'S COMPILE!

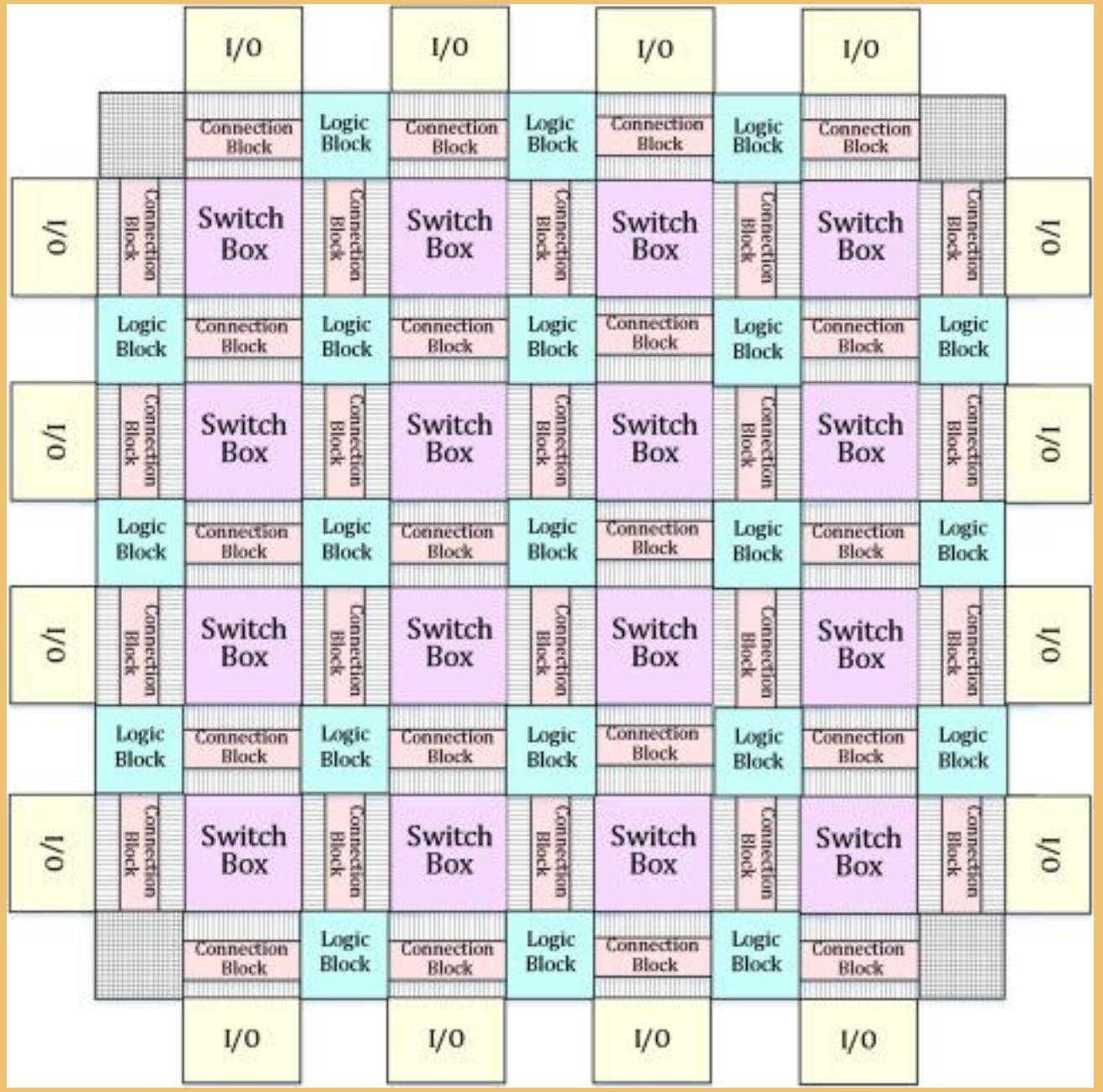


- **How does our HDL code get transferred to FPGA?**
  - **Step 1**: Design your system and implement in HDL
  - **Step 2**: Synthesis
    - HDL compiler translates code into circuit's logical elements (AND/OR/NOT gates/ registers /etc) and their connections
  - **Step 3**: Mapping:
    - Mapper program maps the circuit's logical elements from synthesis to the FPGA's specific hardware: LUTs, routing MUXs, etc.
  - **Step 4**: Place and Route:
    - This step determines the physical placement of the logic elements and the best way to route the signals around the FPGA from LUT to LUT
  - **Step 5**: Bitstream Generation
    - Compiler generates the exact 1 and 0's (bitstream) to load into the FPGA's SRAM cells
  - **Step 6**: Configuration:
    - Bitstream is then downloaded into FPGAs SRAM

THE "ARRAY" IN FPGA



- FPGAs have their overall organization patterned after that of gate-arrays
- Many **configurable logic cells** are arranged in a two-dimensional array with bundles of parallel wires in between
- A mux is present wherever two wiring channels intersect
- Depending on the product, each logic cell can be configured such as to carry out some not-too-complex combinational operation, to store a bit or two, or both



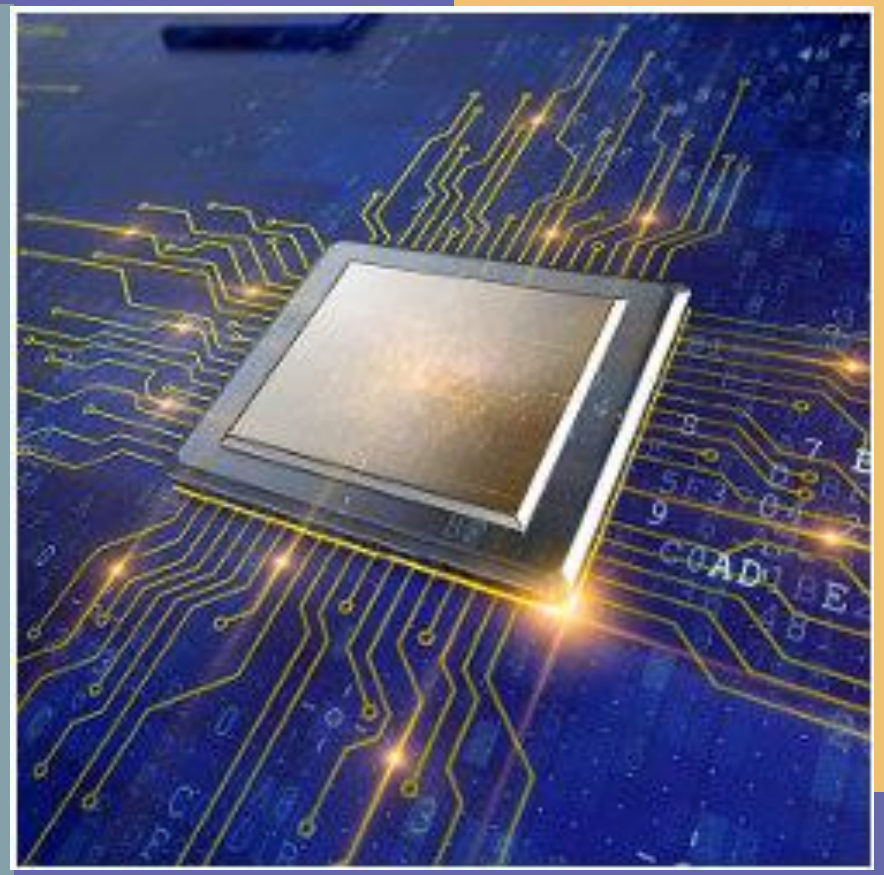
WHEN TO USE AN FPGA

- Should I use it for everything?
  - While it is a universal tool, it can be overkill
    - FPGAs tend to be expensive:
    - Entry-level: $$20-$100
    - Mid-Range: $100-$500
    - High Performance: $500-$10,000!
- Another cost is "power"
  - There are billions of transistors on an FPGA, not all are used in every design; they can be turned off…
  - However, 100 mA (at rest) to 1000 mA (when you push the chip)
- End of the day…how do I decide?
  - Creating custom circuits can be difficult; ask yourself if there is a better solution
  - Many capable processors with many peripherals can handle many of the problems you wish to solve
  - Trying to do design an FPGA to do something like *sending and receiving data over WiFi* would be a daunting task with an but easily accomplished with a few dollar microcontroller.

ALTERNATIVES TO FPGAs

- **Breadboards / PCBs**
  - Quick prototypes of a small scale digital circuit
- Microprocessors
  - General Purpose Computing Tasks with high performance and flexibility
- Microcontrollers
  - Great for embedded systems requiring lower power consumption, and specific predictable tasks
- FPGAs
  - Complex Custom Digital Logic
  - Serve as a prototype before on creates an ASIC (reprogrammable, so you can fix mistakes!)
- ASICs
  - Application Specific Integrated Circuit
  - This is a custom digital circuit made in Silicon that can never be reprogrammed (at the hardware level) once it is created

QUESTIONS??



REFERENCES

- https://learn.sparkfun.com/tutorials/how-does-an-fpga-work/all
- https://en.wikipedia.org/wiki/Field-programmable_gate_array
- https://www.sciencedirect.com/topics/computer-science/configurable-logic-block
- https://community.fs.com/article/a-brief-introduction-to-cpu-gpu-asic-and-fpga.html
- https://en.wikipedia.org/wiki/Field-programmable_gate_array