Digital Design with FPGAs

A TECHNICAL TALK FOR THE IEEE CONSULTANTS NETWORK OF LONG ISLAND (LICN)

Presenter and Affiliation

Presented by: Smitha Kaje

Embedded Systems Engineer

Affiliation: Spellman High Voltage Electronics Corporation, NY

What we'll Cover Today

Part I: Introduction and History

What are FPGAs?

A Brief History

Major Vendors

Types of FPGAs

FPGA versus ASIC

FPGA Boards

Part II: The Internal Architecture

Early Digital Design

Inside a Modern FPGA

Core Components

High-Speed Transceivers

FPGA DSP

Clock in FPGA

FPGA Timing

What we'll Cover Today

Part III: The Design Methodology

- The FPGA Design Flow
- Hardware Description Languages (HDLs)
- FPGA Verification & Simulation
- Behavioral Simulation
- Synthesis
- Functional Simulation
- Timing Simulation

Part IV: Applications

- AI/Machine Learning
- Medical Imaging

Part V:

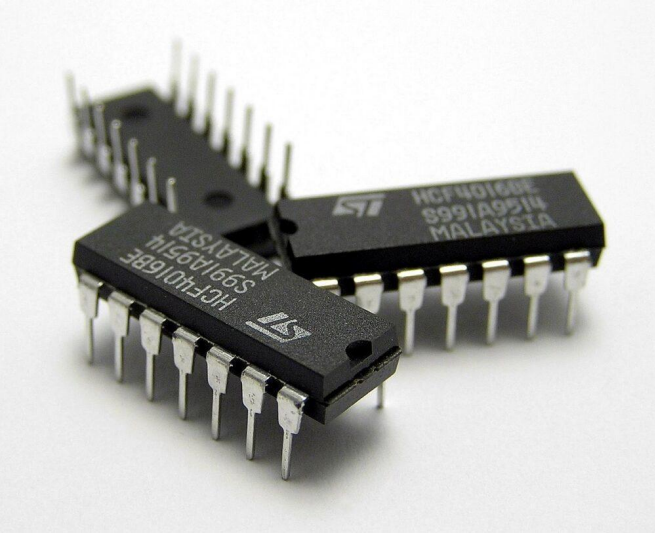
- Tools and Courses
- Q & A

Part I

INTRODUCTION AND HISTORY

What are FPGAs? **Field Programmable Gate Arrays**

An **Integrated Circuit (IC)**, or "chip," is a miniature electronic circuit on a single piece of semiconductor material



FPGAs are r**econfigurable integrated circuits** that act as b**lank digital logic chips**, allowing users to define and implement custom digital circuits by programming the logic blocks and their interconnections

A Stratix IV FPGA from Altera

CREDIT TO ORIGINAL SOURCE
WIKIPEDIA

The History of the FPGAs:

Pre-FPGA programmable logic (1950s – 1970s)

1957: Programmable Read-Only Memory (PROM)
Wen Tsing Chow invented PROM for the Atlas E/F airborne digital computer, patenting the "storage matrix" technology

1970: Erasable Programmable Read-Only Memory (EPROM)
Dov Frohman of Intel filed a patent for EPROM, which could be erased with ultraviolet light

1977: Programmable Array Logic (PAL)
John Birkner and Hua-Thye Chua filed a patent for PAL, featuring a programmable AND plane and a fixed OR plane.

The History of the FPGA:

The Age of Invention (1984 – 1991)

- **1983 – 1984: The first reprogrammable logic device Altera**
  Founded in 1983, delivered the first reprogrammable logic device in 1984, the EP300
  It was based on EPROM technology, requiring a UV lamp for erasure

- **1984 – 1985: The first commercial FPGA**
  Former Zilog engineers Ross Freeman, Bernard Vonderschmitt, and James V. Barnett II founded Xilinx in 1984
  In 1985, they released the XC2064, the first commercially viable field-programmable gate array.
  The term "FPGA" was later popularized by competitor Actel in the late 1980s

- **1987:**
  US Naval Surface Warfare Center project Steve Casselman secured funding for a reconfigurable computer with 600,000 gates
  The project was successful, and a patent was issued in 1992

The History of the FPGA:

The Age of Expansion (1992 – 1999)

- **Increased competition and market share growth:**
  Competitors like Actel (later Microsemi, then Microchip) entered the market.
  By 1993, the FPGA market had grown significantly, with Actel holding 18% market share

- **Market diversification:**
  By the late 1990s, FPGAs had expanded into consumer, automotive, and industrial electronics

- **Enhanced architecture:**
  The introduction of dedicated multipliers into FPGA architectures in the late 1990s enabled FPGAs to compete with digital signal processors (DSPs) in many applications.

The History of the FPGA:

The Age of Accumulation (2000 – 2007)

- **System-on-a-Chip (SoC) integration:**
  Xilinx introduced its first SoC in 2012, which combined a processor system with programmable logic

- **Early data center usage:**
  Companies began to explore using FPGAs for hardware acceleration in data centers

The History of the FPGA:

Recent modern applications

- **2014: Microsoft's Project Catapult**
  Microsoft began deploying FPGAs in its data centers to accelerate the **Bing search** engine, demonstrating the technology's performance-per-watt benefits for computationally intensive tasks
- **2015: Acquisition of Altera by Intel**
  Intel acquired Altera for $16.7 billion, a major move by a CPU giant to acquire FPGA technology
- **2019: AI engine integration**
  FPGAs evolved to incorporate specialized hardware for artificial intelligence (AI) and machine learning workloads
- **2022: Acquisition of Xilinx by AMD**
  AMD completed a $50 billion acquisition of Xilinx, signaling the growing strategic importance of FPGAs in the semiconductor industry
- **Continued hardware acceleration** across various fields

Major Vendors & Manufacturers of FPGAs

AMD (Advanced Micro Devices)

- **Acquired:** Xilinx (the pioneer of the FPGA) in 2022
- **Focus:** High-performance FPGAs, Adaptive SoCs (Versal ACAP), and System-on-Chip (SoC) FPGAs for Data Center, AI/ML, 5G, and Aerospace/Defense
- **Key Products:** Virtex, Kintex, Artix, Spartan, and Zynq/Versal ACAP families

- **Intel Corporation**
  - **Acquired:** Altera in 2015
  - **Focus:** High-performance, high-density FPGAs, primarily targeting Data Center, Networking, and embedded applications
  - **Key Products:** Agilex, Stratix, Arria, Cyclone, and MAX families

Prices range from **$10 (Lattice iCE40) to several thousand dollars (Stratix 10 TX FPGA Device - $55K)**

FPGA versus ASIC

Programmable ICs:

FPGAs

Flexible, reconfigurable internal structure

Functionality is not set until after manufacturing, when the user "programs" them

Fixed-Function ICs:

ASICs (Application-Specific Integrated Circuits)

Internal wiring is permanent and cannot be altered

Examples:
A CPU, a simple logic gate chip

CREDIT TO ORIGINAL SOURCE
HTTPS://WWW.UTMEL.COM/BLOG/CATEGORIES/INTEGRATED%20CIRCUIT/FPGA-VS-ASIC-COMPREHENSIVE-COMPARISON-GUIDE

Comparison between FPGA and ASIC

Feature	FPGA	ASIC
Flexibility	High (reprogrammable)	Low (not reprogrammable)
Performance	Lower than ASIC	Higher Performance for specific tasks
Power Consumption	Higher compared to ASIC	Lower (optimized for efficiency)
Development Cost	Low (no NRE cost)	High (high NRE cost)
Production Cost per Unit	Higher compared to ASIC	Lower (optimized for volume)
Time to Market	Shorter (reprogrammable, adaptable)	Longer (due to design and fabrication)
Reprogrammability	Yes (can change algorithms post-production)	No (fixed design)
Suitable Production Cycle	Small to medium scale	High volume (to offset NRE costs)
Design Cycle	Shorter	Longer
Examples	Custom chips in smartphones, game consoles, smart TVs	Flexible Control Systems, Data Center Acceleration

CREDIT TO ORIGINAL SOURCE
HTTPS://WWW.CANDTSOLUTION.COM/NEWS_EVENTS-DETAIL/WHAT-IS-FPGA-FIELD-PROGRAMMABLE-GATE-ARRAY-AND-HOW-DOES-IT-WORK/

Types of FPGAs

1. **SRAM-based FPGAs**
   - Configuration: Stores its configuration in volatile Static Random-Access Memory (SRAM)
   - Programming: Highly reprogrammable
   - Operation: Requires an external, non-volatile memory (like flash) to load the bitstream when the device powers on
   - Characteristics: Offers a high degree of flexibility for frequent updates or dynamic configurations
   - Example Vendors: Lattice Semiconductor

2. **Antifuse-based FPGAs**
   - Configuration: Uses antifuses, which are one-time programmable elements
   - Programming: Once programmed, the antifuses create a permanent connection that cannot be changed
   - Characteristics: Non-volatile, small routing delays, and good for security as they are difficult to read
   - Example Vendors: Microchip

3. **Flash-based FPGAs**
   - Configuration: Stores its configuration in non-volatile flash memory
   - Programming: Reprogrammable and non-volatile, combining the benefits of both SRAM and antifuse FPGAs
   - Characteristics: Offers a balance of non-volatility for faster boot times and the ability to be reprogrammed
   - Example Vendors: Microchip, Lattice

Types of FPGAs

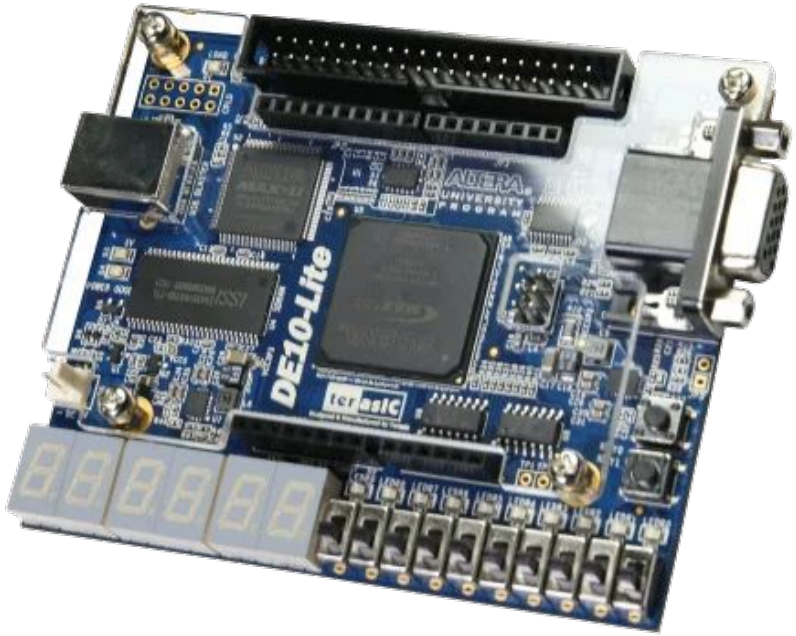
1. **EEPROM-based FPGAs**
   - Configuration: Uses Electrically Erasable Programmable Read-Only Memory (EEPROM) to store its configuration
   - Programming: Highly reprogrammable
   - Characteristics: Non-volatile and reprogrammable, similar to flash-based FPGAs
   - Example Vendors: AMD, Intel

2. **Hybrid FPGAs**
   - Configuration: Integrates various types of programmable elements (e.g., SRAM and flash logic)
   - Programming: Once programmed, the antifuses create a permanent connection that cannot be changed
   - Characteristics: Provides a combination of performance, low-power operation, and versatility
   - Example Vendors: AMD, Microchip, Intel

3. **System-On-Chip (SoC) FPGAs**
   - Configuration: Integrates traditional FPGA logic with hard processor cores (like CPUs and GPUs) on a single chip
   - Programming: Reprogrammable and non-volatile, combining the benefits of both SRAM and antifuse FPGAs
   - Characteristics: Combines the flexibility of FPGAs with the advanced processing capabilities of a traditional SoC for complex, heterogeneous workloads
   - Example Vendors: AMD, Intel

FPGA Boards

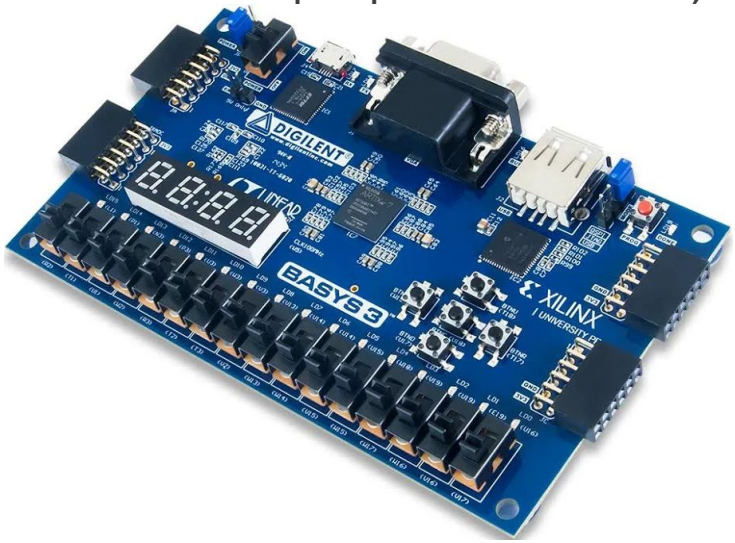
FPGA development boards come equipped with essential peripherals such as switches, LEDs, Analog-to-Digital Converters (ADCs), and Digital-to-Analog Converters (DACs)



Types of FPGA Boards

1. General-purpose development boards

- Best for: Beginners and students learning digital logic design and FPGA programming
- Features: Standard peripherals like LEDs, buttons, switches, and basic I/O interfaces



Digilent Basys 3: Has a Xilinx Artix-7 FPGA, with onboard I/O

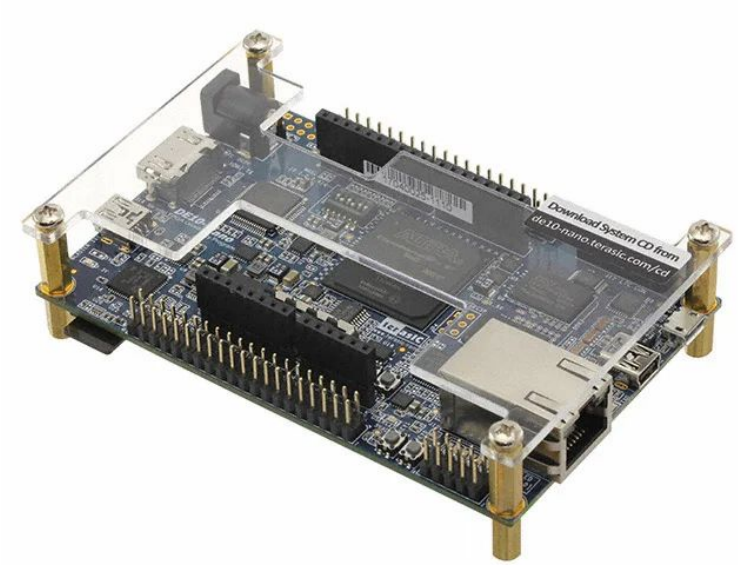
From Terasic with an Intel Cyclone 10 FPGA, including 7-segment displays

Types of FPGA Boards

2. System-On-Chip (SoC) FPGA boards

- Has a powerful hard-core processor (like an ARM Cortex-A9) with the FPGA's programmable logic on a single chip

Digilent Zybo Z7 Zynq-7000 ARM/FPGA SoC Development Board



Programmable Logic IC Development Tools DE10-Nano Dev Kit

Types of FPGA Boards

3. High-performance computing (HPC) boards

- These include high-speed transceivers and large-capacity FPGAs.

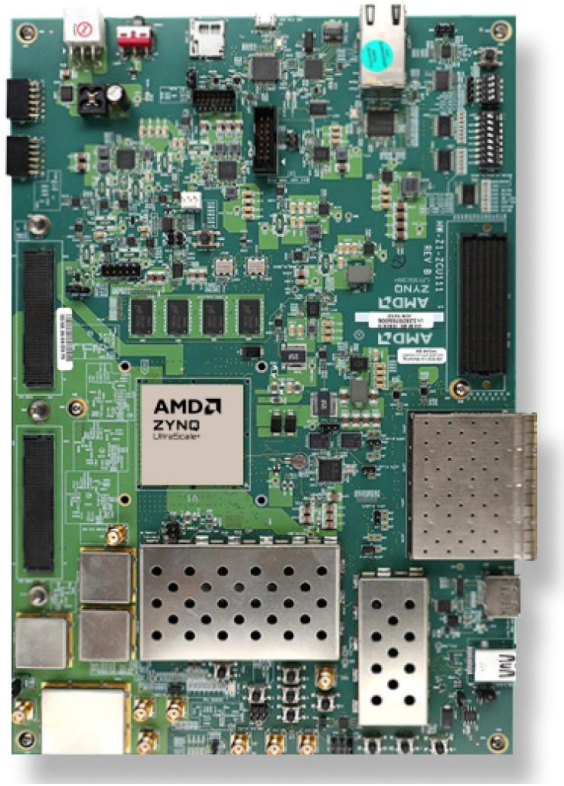
Agilex™ 7 FPGA F-Series FPGA Development Kit

Types of FPGA Boards

4. Application-specific boards

- Optimized for a particular purpose or industry and feature specialized peripherals and configurations

  Example: Aerospace, Medical Imaging



AMD Zynq™ UltraScale+™ RFSoC ZCU111

Types of FPGA Boards

5. Custom FPGA boards

- For applications with highly specific requirements that cannot be met by off-the-shelf options

Part II

THE INTERNAL ARCHITECTURE

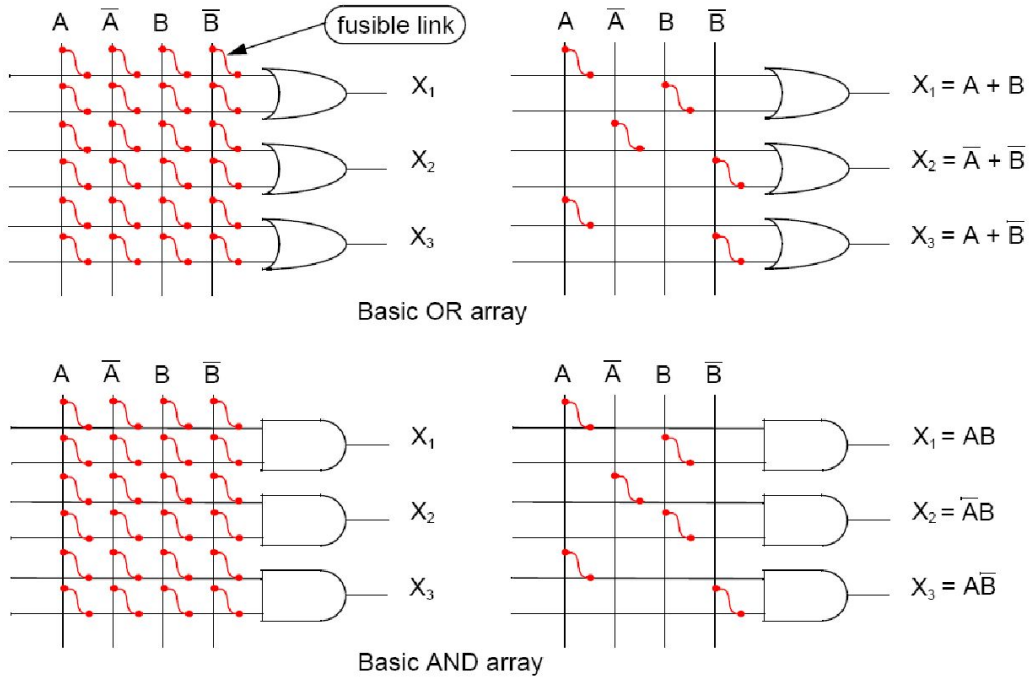
Early Digital Design

The first programmable chips were **PLAs (Programmable Logic Arrays)**

Two-level structures of AND and OR gates with user-programmable connections

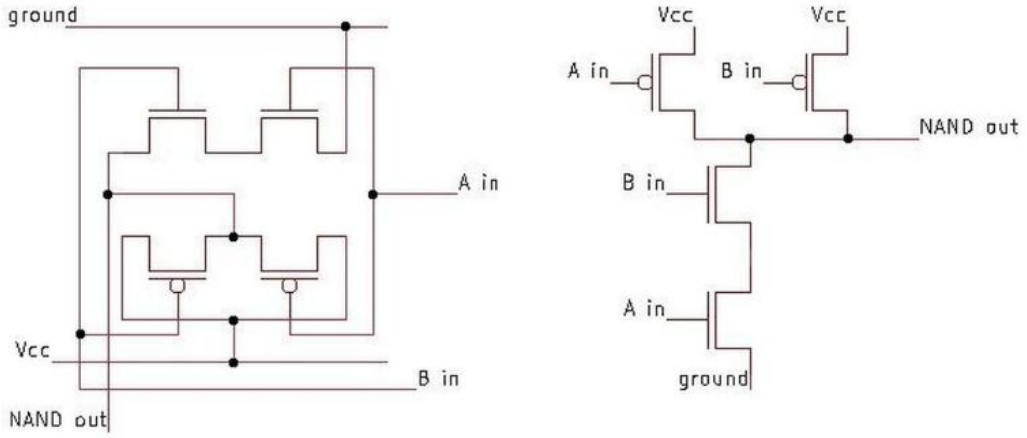
Generically called Programmable Logic

NAND Gate Schematic

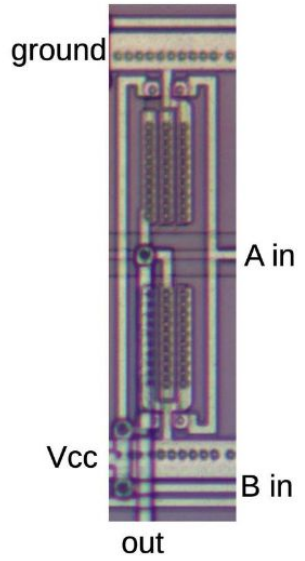


Early Digital Design

Some would design chips on their own using an array of NAND Gates



NAND Gate Schematic



NAND Gate on the die

CREDIT TO ORIGINAL SOURCE
HTTPS://WWW.RIGHTO.COM/2024/03/IDT-GATE-ARRAY.HTML

Early Digital Design

The earliest logic integrated circuits used resistors and transistors internally, so were called **RTL (Resistor Transistor Logic)**

RTL was replaced by **Diode Transistor Logic (DTL)** and then by **Transistor Transistor Logic (TTL)**

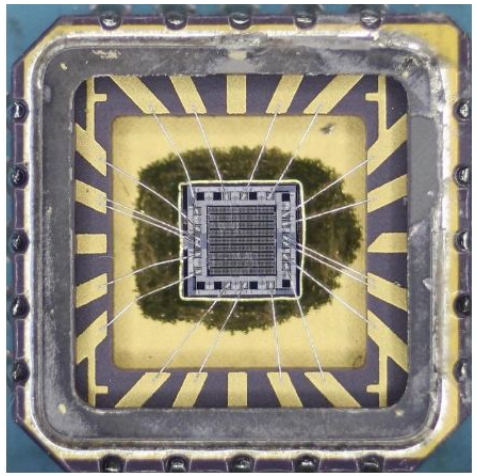
Engineers used discrete **7400-series transistor-transistor logic (TTL)** integrated circuits (ICs) as modular building blocks

Texas Instruments (TI) first introduced the 7400-series

Inexpensive, fast, and easy to use

Started with simple logic circuits such as four NAND gates on a chip, and moved into more complex chips such as counters, shift registers, and ALUs

Example: was used in **Apple II**



A tiny silicon die in its package
IDT 54FCT139ALB dual 1-of-4 decoder

CREDIT TO ORIGINAL SOURCE
HTTPS://WWW.RIGHTO.COM/2024/03/IDT-GATE-ARRAY.HTML

Inside a Modern FPGA

A matrix of reconfigurable logic blocks (CLBs)

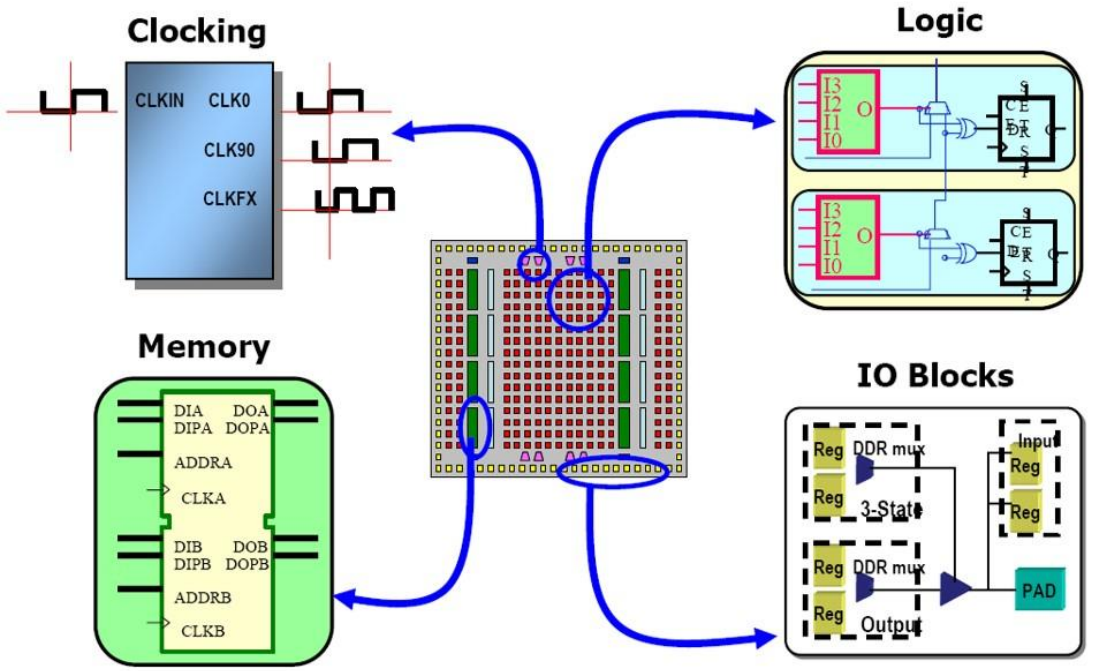
Interconnected by a programmable routing fabric

Input/Output Blocks

Clocking Architecture

High-Speed Transceivers

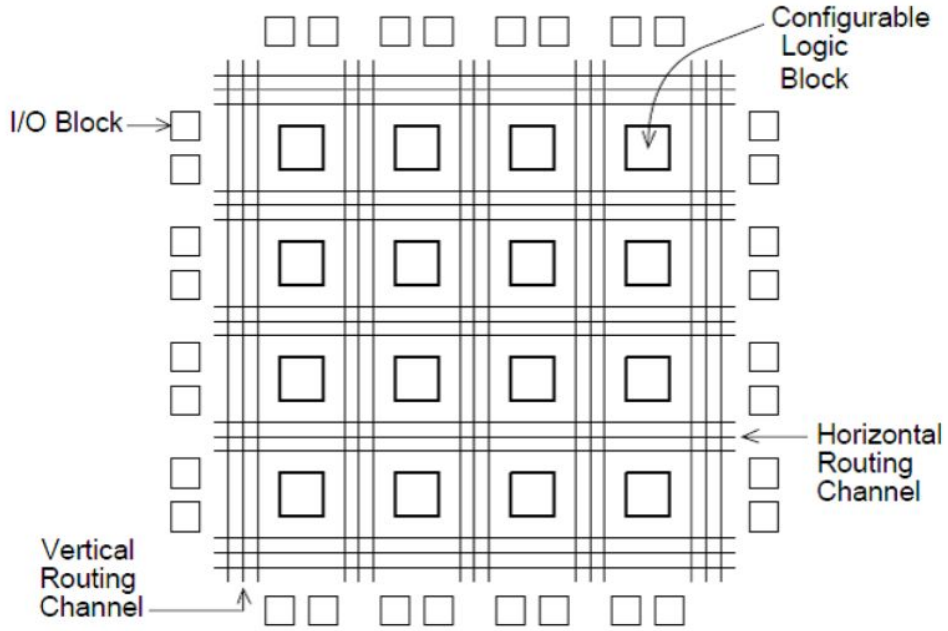
FPGA DSP



https://www.researchgate.net/figure/linx-structure-of-FPGA-123_fig15_257675533

Core Components of an FPGA

- **Configurable Logic Blocks (CLB) with programmable logic (LUT)** and few registers - implement logic functions
- **Input Output Blocks** - make off-chip connections
- **Programmable interconnects** - connect I/O blocks and CLBs

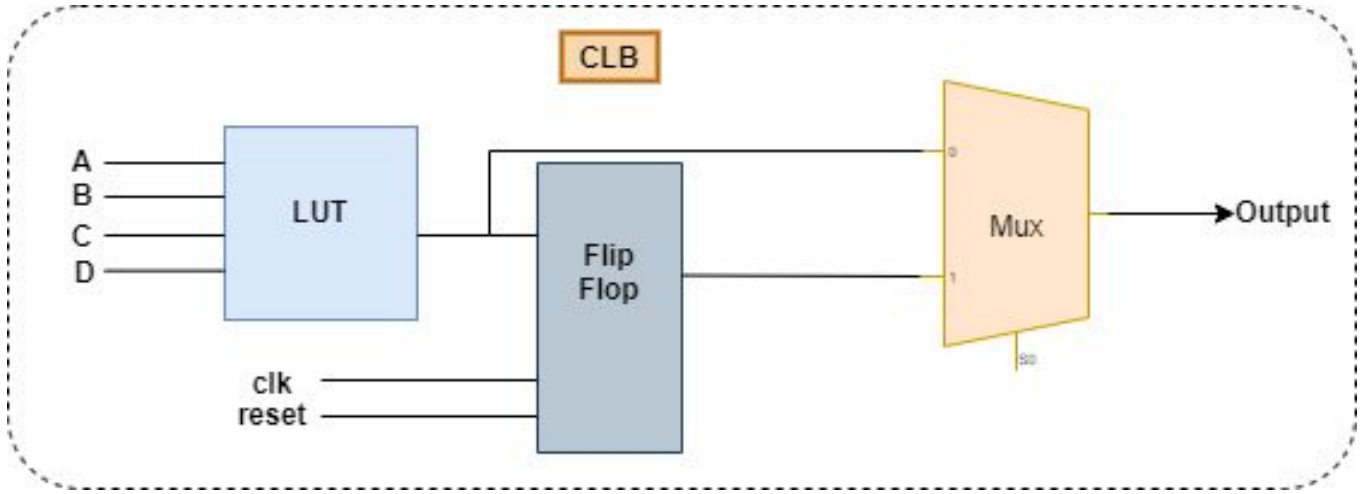


CREDIT TO ORIGINAL SOURCE
HTTPS://DIGITALSYSTEMDESIGN.IN/FPGA-BASIC-OVERVIEW/

Configurable Logic Blocks (CLBs)

Any logic circuit can be realized in terms of these CLBs

Generally, consists of a LUT, a multiplexer and flip-flops



Input A	Input B	Output C
0	0	**0**
0	1	**0**
1	0	**0**
1	1	**1**

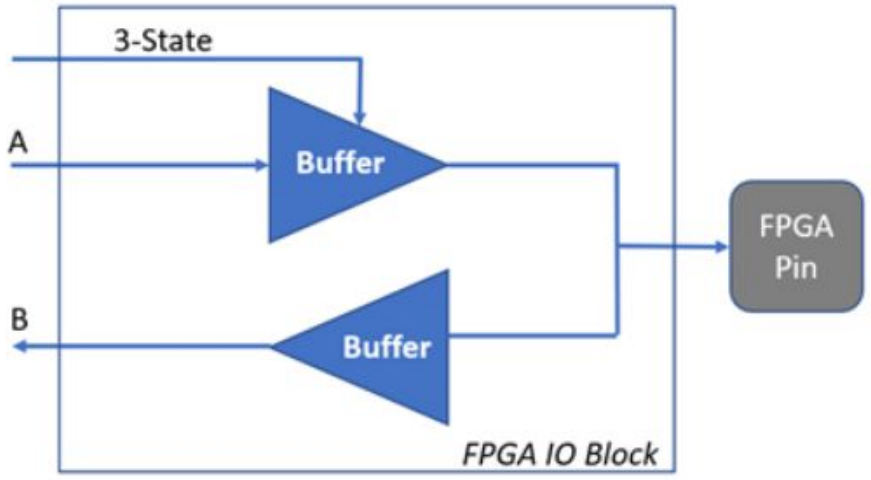
Lookup table that is implementing the function of an AND gate

CREDIT TO ORIGINAL SOURCE
HTTPS://HARDWAREBEE.COM/WHAT-IS-FPGA/
HTTPS://HARDWAREBEE.COM/THE-ULTIMATE-GUIDE-TO-FPGA-ARCHITECTURE/
HTTPS://HARDWAREBEE.COM/OVERVIEW-OF-LOOKUP-TABLES-IN-FPGA-DESIGN/

Input/Output Blocks

The interface between an FPGA and other external devices is enabled by input/output (I/O) blocks

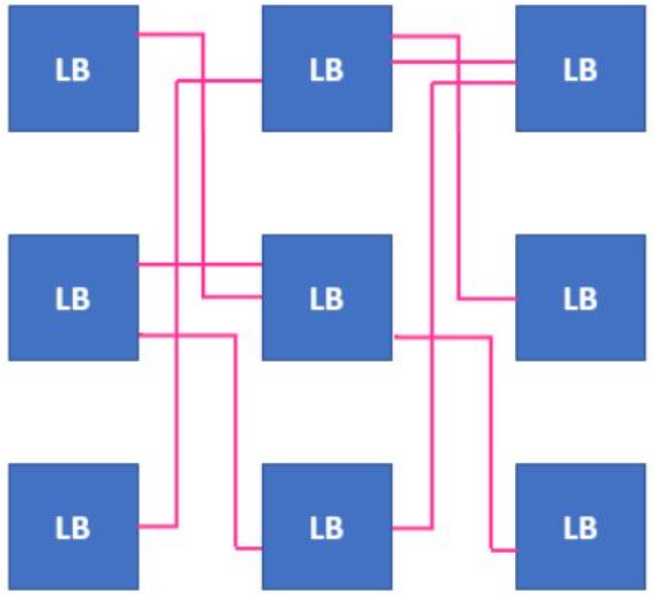
All signals entering or leaving the FPGA do so through device pins and associated IOBs



CREDIT TO ORIGINAL SOURCE
HTTPS://HARDWAREBEE.COM/WHAT-IS-FPGA/

Programmable Interconnects

The various logic blocks are connected with the help of this internal routing



CREDIT TO ORIGINAL SOURCE
HTTPS://HARDWAREBEE.COM/WHAT-IS-FPGA/

High-Speed Transceivers

To establish communication between the FPGA device and the external interfaces or devices

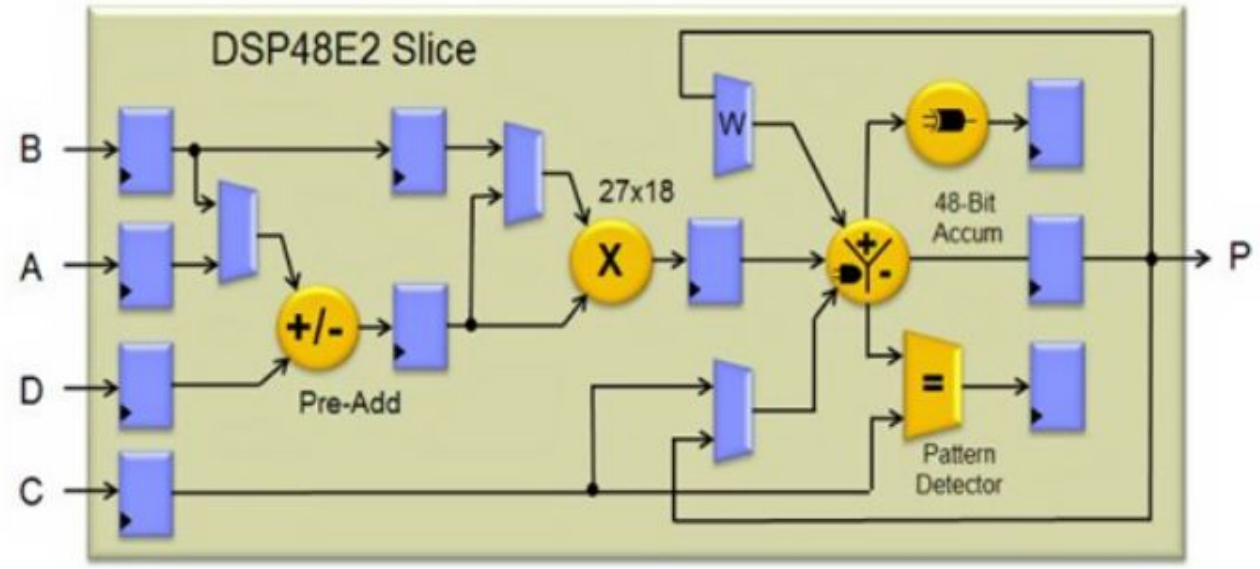
Many standard protocols that can be used for this purpose: **Ethernet, PCI, HDMI**

Ethernet: Intel's Agilex series

Intel Agilex FPGAs support up to PCIe 4.0 x8, while AMD Versal and older 7 Series FPGAs offer up to PCIe 3.0

FPGA DSP

The DSP48E2 was first introduced in Virtex–5 FPGA to make the multiplication and accumulation operations fast



Courtesy: Xilinx

Clock in FPGA

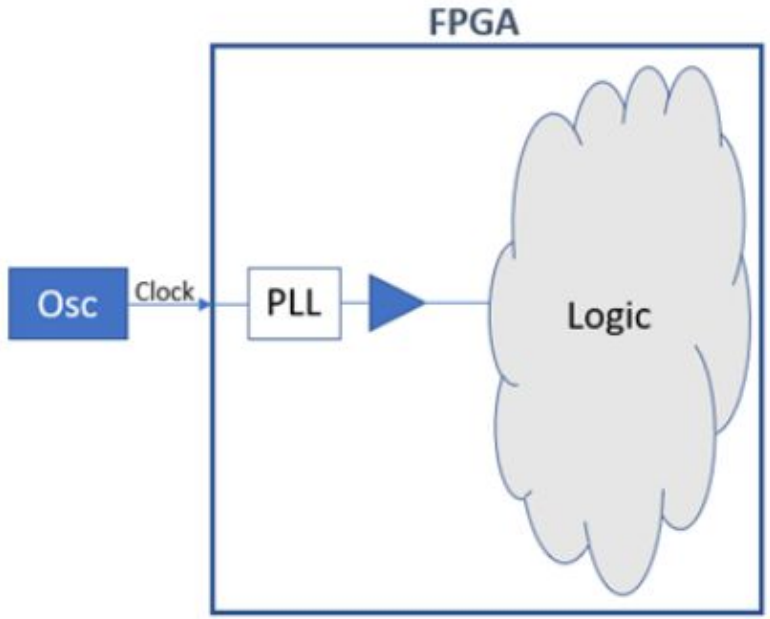
A clock **is a signal inside any digital circuit** that determines **how fast a flip-flop runs**
The **faster** the **clock**, the **faster** the **design will run**

Responsible for driving the FPGA design and determines how fast it can run an process data (maximum of upwards of 1GHz)

The clock signal is connected to all flip-flops and RAM blocks and activates ther according to the clock frequency

A typical FPGA consists of **several clock signals** and thus allows **different areas across the FPGA to operate in various speeds**

FPGA systems contain internal **phase-locked loops (PLLs)** that help generate various frequencies of signal waves

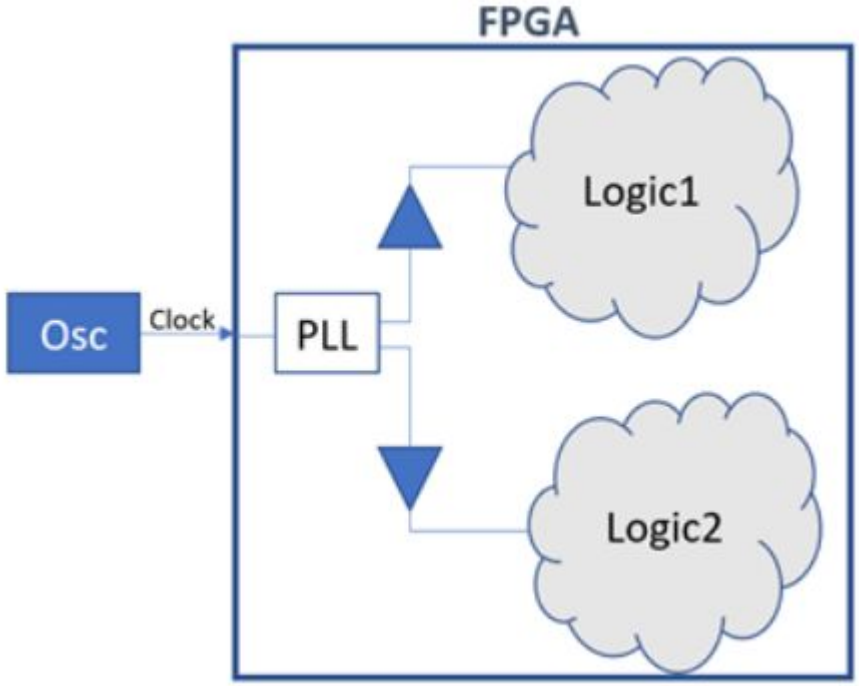


Clock in FPGA

A single FPGA system will employ the **use of at least one clock** that will generate a wave at a certain frequency, which will then be **distributed across the FPGA to produce a synchronized response** from all the flip-flops involved in the design

An **external oscillator** placed on the circuit board is what generates the square wave or clock signal with a certain frequency and enters the FPGA system through a **single physical connecting pin**

The clock signals are distributed along interconnected wires (called global routing) so that the signal is distributed and received at the same time by each flip-flop



FPGA with 2 clock domains running 2 separate clock speeds

CREDIT TO ORIGINAL SOURCE
HTTPS://HARDWAREBEE.COM/ULTIMATE-GUIDE-FPGA-CLOCK

FPGA Timing

Timing refers to **when it takes a signal to propagate from one flip-flop**, through some combinational logic, to the next flip-flop

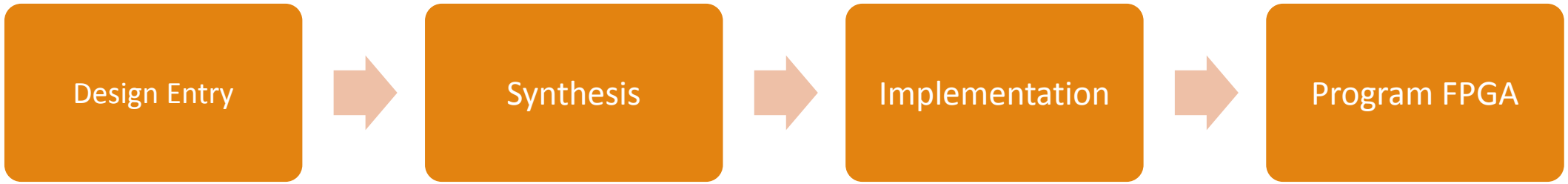
It takes some time for the signal to propagate to the output

The more transistors we need to turn-on and turn-off, the longer it takes

Part III

THE DESIGN METHODOLOGY

FPGA Design Flow



CREDIT TO ORIGINAL SOURCE
HTTPS://WWW.LOGIC-FRUIT.COM/BLOG/FPGA/FPGA-DESIGN-AN-ULTIMATE-GUIDE-FOR-FPGA-ENTHUSIASTS

FPGA Design Flow

Design entry
(Schematics or HDL)

HDL: Hardware Description Language

```
// ----------------------------------------------
// MODULE DEFINITION: 2-input OR Gate
// ----------------------------------------------
module or_gate_2_input (
   input  A,
   input B,
   output Q
);

   // Output Q is TRUE if A OR B is TRUE.
   assign Q = A | B;

endmodule
```

FPGA Design Flow

Synthesized OR Gate

Steps:

- **Parsing:** Syntax check on the HDL-based design
- **Optimize**: By reducing logic, eliminating redundant logic, and reducing the size of the design
- **Generate a techno-independent output:** The output of synthesis is a generic netlist (not yet specific to any particular FPGA family or device)
- Performed by dedicated synthesis tools
- Cadence, Synopsys and Mentor Graphics are EDA companies that develop, sell, and market FPGA synthesis tools.

CREDIT TO ORIGINAL SOURCE
HTTPS://HARDWAREBEE.COM/ULTIMATE-GUIDE-FPGA-DESIGN-FLOW/

FPGA Design Flow

- The layout of our design will be determined
- Three steps: translate, map, and place & route

- **Placement:** The first step for the tools is to gather all the constraints that are set by the user, together with the netlist files
- The logic elements (like the logic gates for the OR function) are assigned to specific physical locations (logic cells) within the FPGA's grid.
- **Routing:** Electrical connections (wires) are programmed to connect the placed logic cells, creating the actual circuit paths within the FPGA.

CREDIT TO ORIGINAL SOURCE
HTTPS://HARDWAREBEE.COM/ULTIMATE-GUIDE-FPGA-DESIGN-FLOW/
HTTPS://WWW.TENCENTCLOUD.COM/TECHPEDIA/101380

FPGA Design Flow

- The last step in the process is to finally load the mapped-out and completely routed design into the FPGA
- For this, we will need to generate a BitSteam file
- Example: a **.bit or .rpt file** generated by vendor tools like **Intel Quartus Prime or Xilinx Vivado**
- Bitstreams can **either full or partial** **(**entire or partial configuration memory of a given device)

```
Xilinx ASCII Bitstream Created by
Bitstream 2018.3 SW Build 2405991 on Thu
Dec 6 23:36:41 MST 2018 Design name:
base_mb_wrapper;UserID=0XFFFFFFFF;Version
=2018.3 Architecture: virtexuplus Part:
xcvu9p-flga2104-2L-e Date: Wed Nov 20
04:13:05 2019 Bits: 641272864
11111111111111111111111111111111
11111111111111111111111111111111
11111111111111111111111111111111
```

Hardware Description Languages (HDLs)

A specialized language used to design and describe the **behavior and structure of digital circuits**

Not a Traditional Programming Language:
Unlike Python or C++, HDLs **DO NOT execute sequential instructions**.
Instead, they act as a blueprint for digital logic

Synthesizable Code:
Key characteristic - The HDL code must be synthesizable to be translated into actual hardware components (registers, gates) that the FPGA can implement.

VHDL: Developed by the **United States Department of Defense** in the 1980s
VHDL stands for VHSIC Hardware Description Language. "VHSIC" is an acronym for Very High-Speed Integrated Circuit

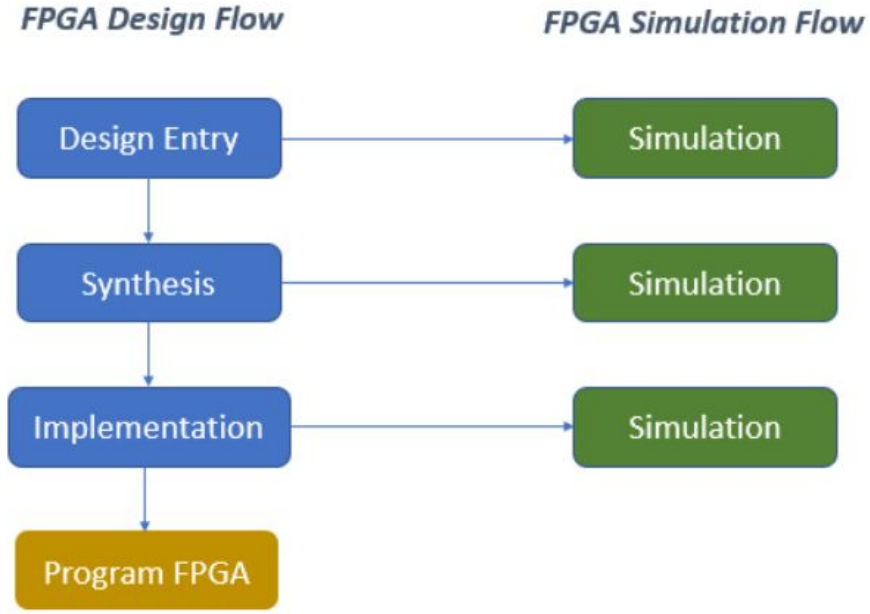
Verilog: Created by **Gateway Design** Automation (Prabhu Goel, Phil Moorby, and Chi-Lai Huang) in the mid-1980s

SystemVerilog: An extension of Verilog that adds features for writing sophisticated test benches
Developed by the Co-Design Automation startup.

FPGA Verification & Simulation

We have the opportunity to simulate and test the design:

- At **design entry**
- At **post synthesis**
- At **post implementation**



CREDIT TO ORIGINAL SOURCE
HTTPS://HARDWAREBEE.COM/ULTIMATE-GUIDE-FPGA-DESIGN-FLOW/

Behavioral Simulation (At Design Entry)

Called RTL Simulation

To test the code and find the logic errors

Design Under Test (DUT):
2-to-1 Multiplexer

C:\altera\13.1\mux\mux2to1.v - Notepad++

File Edit Search View Encoding Language Settings Tools M

mux2to1.v

```
1   // mux2to1.v
2   module mux2to1 (
3       input a,
4       input b,
5       input sel,
6       output reg out
7   );
8
9   always @(*) begin
10      if (sel == 1'b0) begin
11          out = a;
12      end else begin
13          out = b;
14      end
15  end
16
17  endmodule
```

Behavioral Simulation

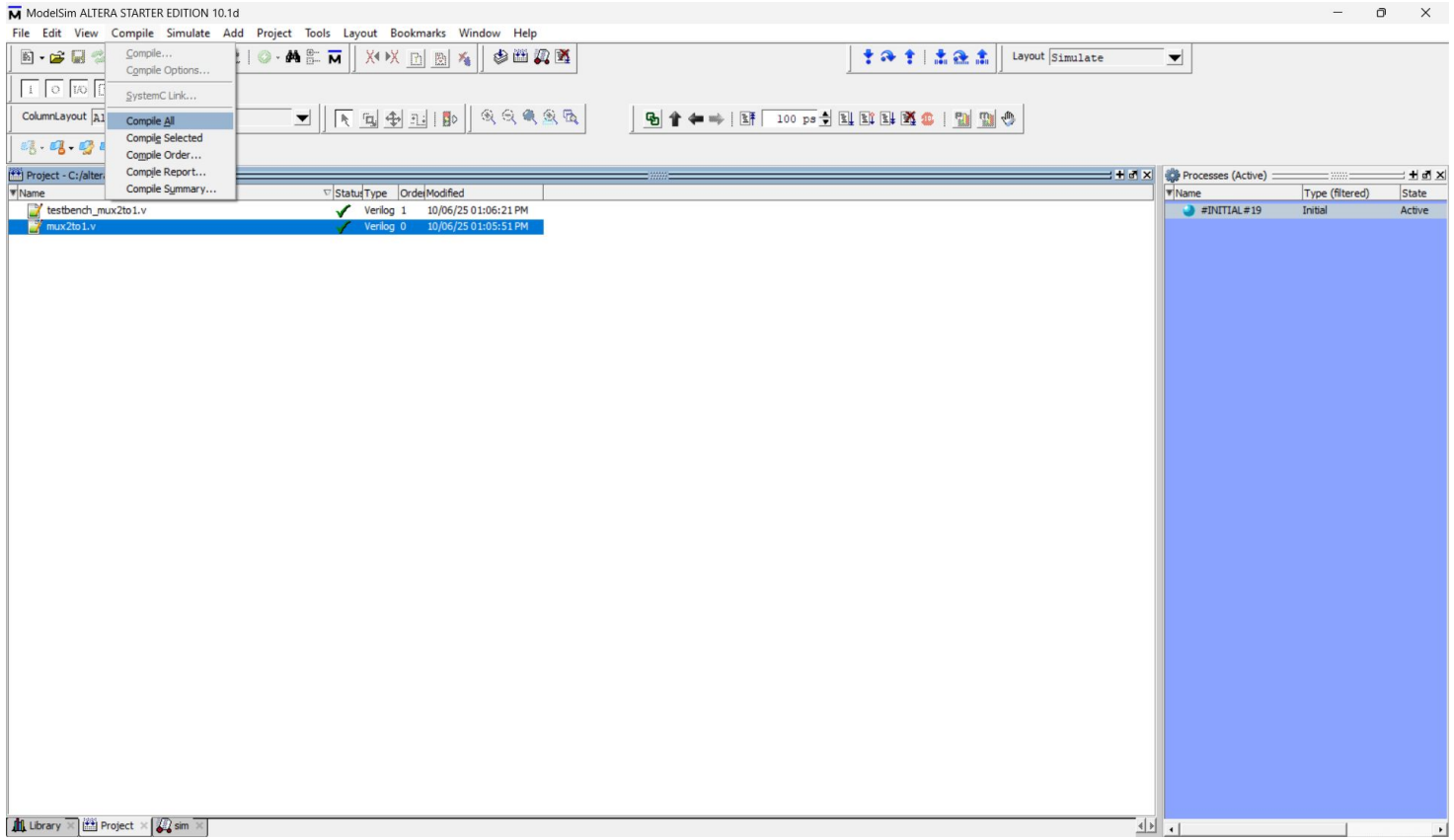
Behavioral Testbench

testbench_mux2to1.v

```
// testbench_mux2to1.v
module testbench_mux2to1;

    // Declare signals for connecting to the DUT
    reg tb_a;
    reg tb_b;
    reg tb_sel;
    wire tb_out;

    // Instantiate the DUT
    mux2to1 uut (
        .a(tb_a),
        .b(tb_b),
        .sel(tb_sel),
        .out(tb_out)
    );

    // Initial block for generating stimulus
    initial begin
        // Initialize inputs
        tb_a = 0;
        tb_b = 0;
        tb_sel = 0;

        // Display initial state
        $display("Time = %0t, a = %b, b = %b, sel = %b, out = %b", $time, tb_a, tb_b, tb_sel, tb_out);

        // Test case 1: sel = 0, a = 1, b = 0
        #10 tb_a = 1; tb_b = 0; tb_sel = 0;
        $display("Time = %0t, a = %b, b = %b, sel = %b, out = %b", $time, tb_a, tb_b, tb_sel, tb_out);

        // Test case 2: sel = 1, a = 0, b = 1
        #10 tb_a = 0; tb_b = 1; tb_sel = 1;
        $display("Time = %0t, a = %b, b = %b, sel = %b, out = %b", $time, tb_a, tb_b, tb_sel, tb_out);

        // Test case 3: sel = 0, a = 0, b = 0
        #10 tb_a = 0; tb_b = 0; tb_sel = 0;
        $display("Time = %0t, a = %b, b = %b, sel = %b, out = %b", $time, tb_a, tb_b, tb_sel, tb_out);

        // End simulation
        #10 $finish;
    end

    // Monitor changes (optional, for detailed logging)
    initial begin
        $monitor("MONITOR: Time = %0t, a = %b, b = %b, sel = %b, out = %b", $time, tb_a, tb_b, tb_sel, tb_out);
    end

endmodule
```

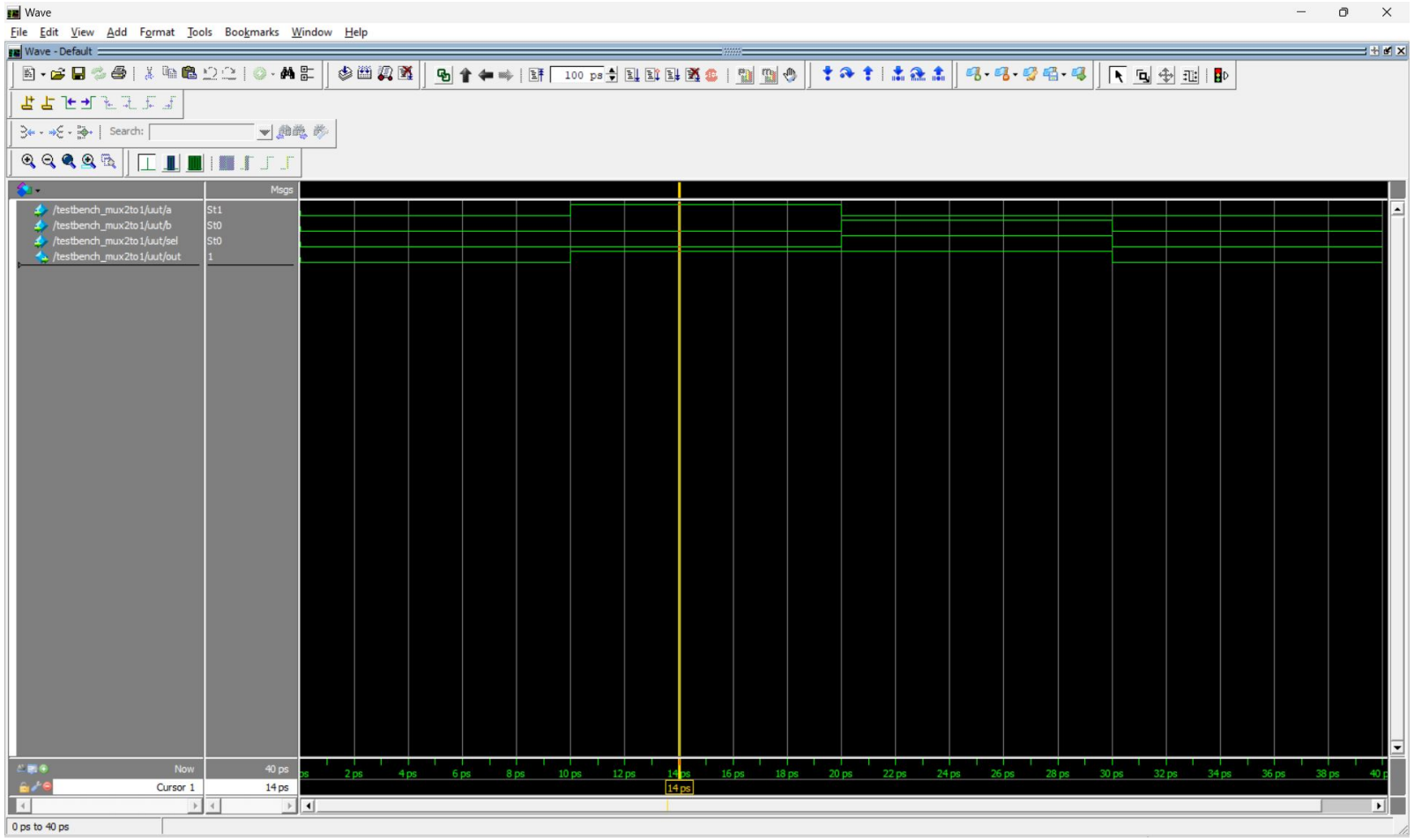
Behavioral Simulation

Verilog simulator: ModelSim

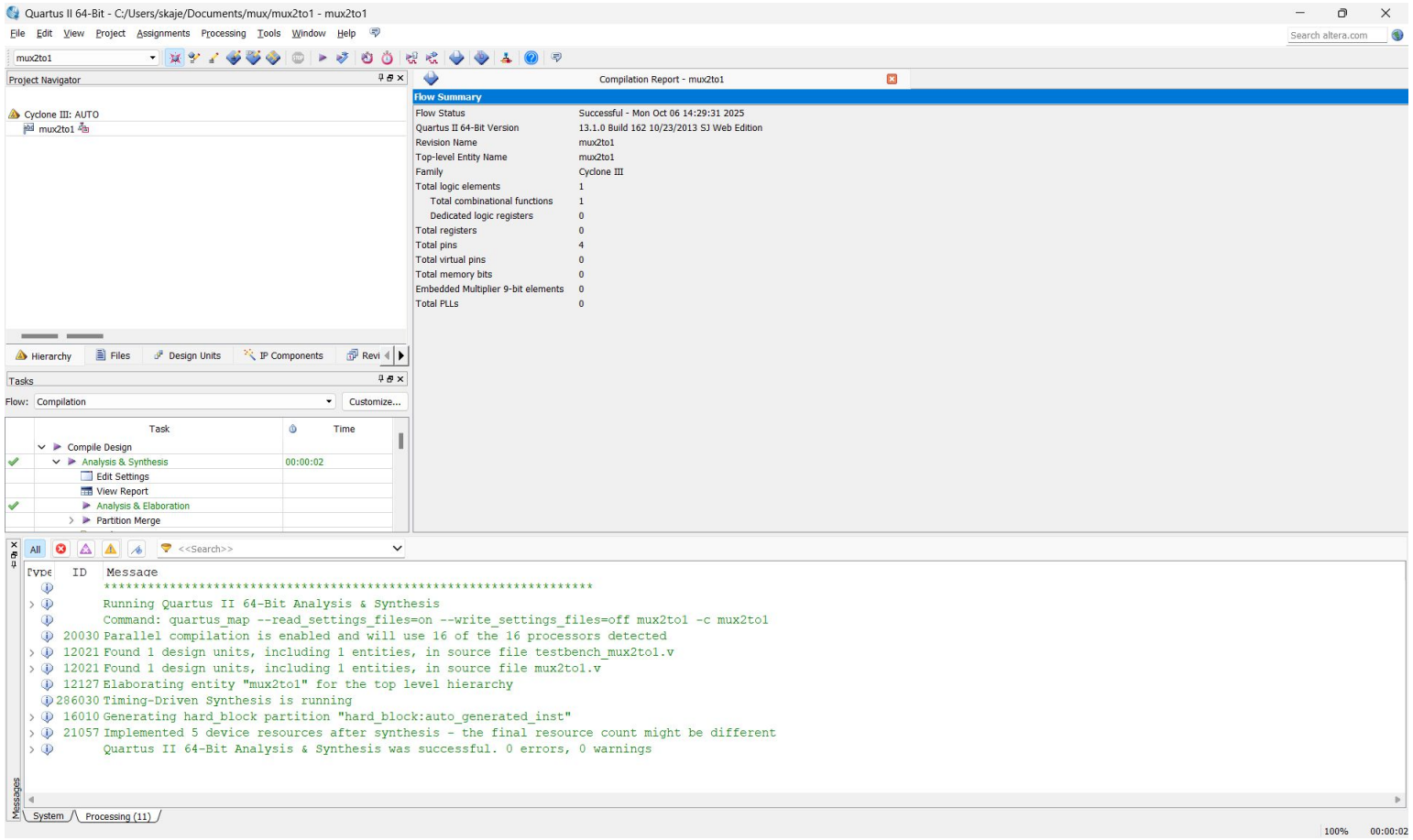


Behavioral Simulation

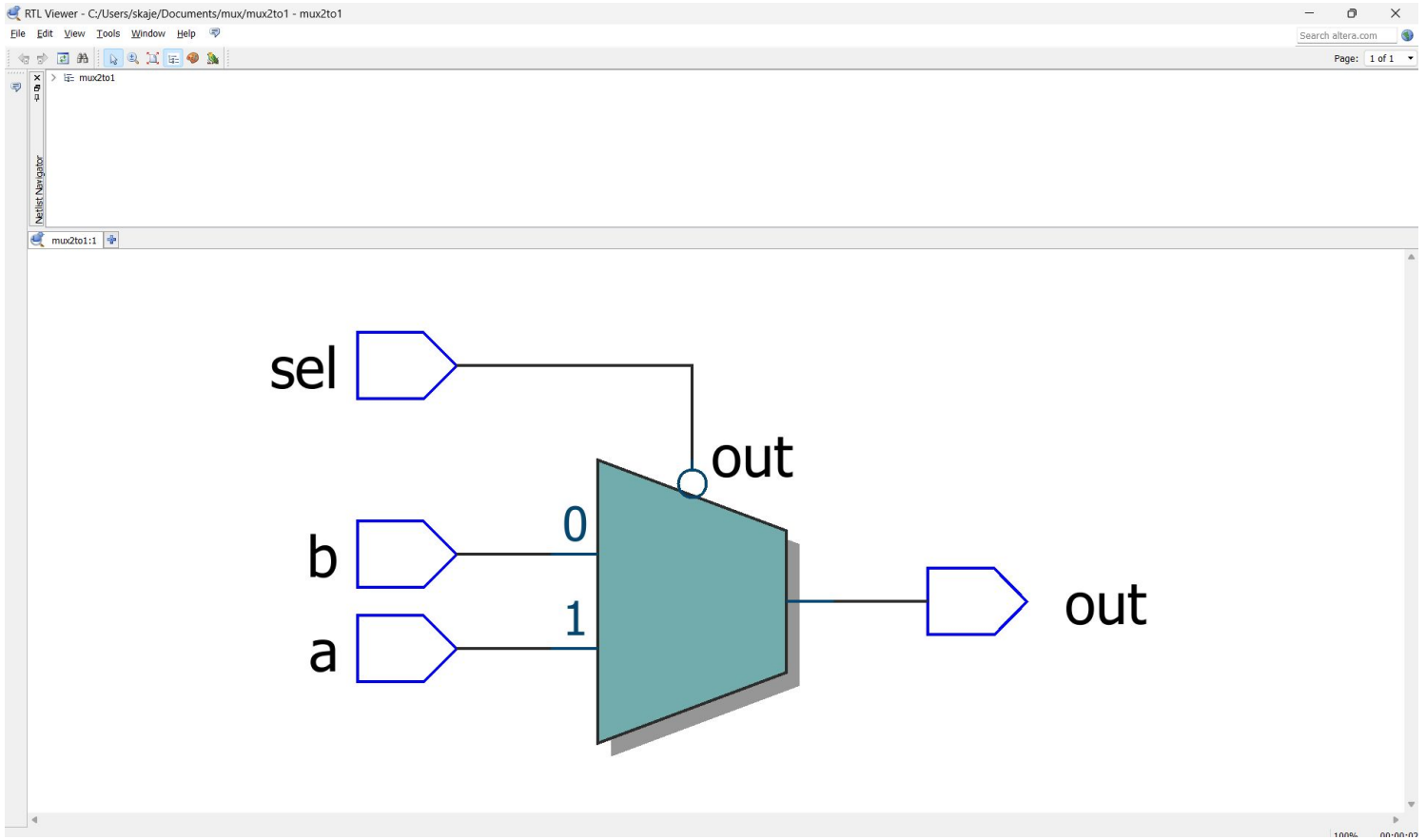
ModelSim: Output Signals



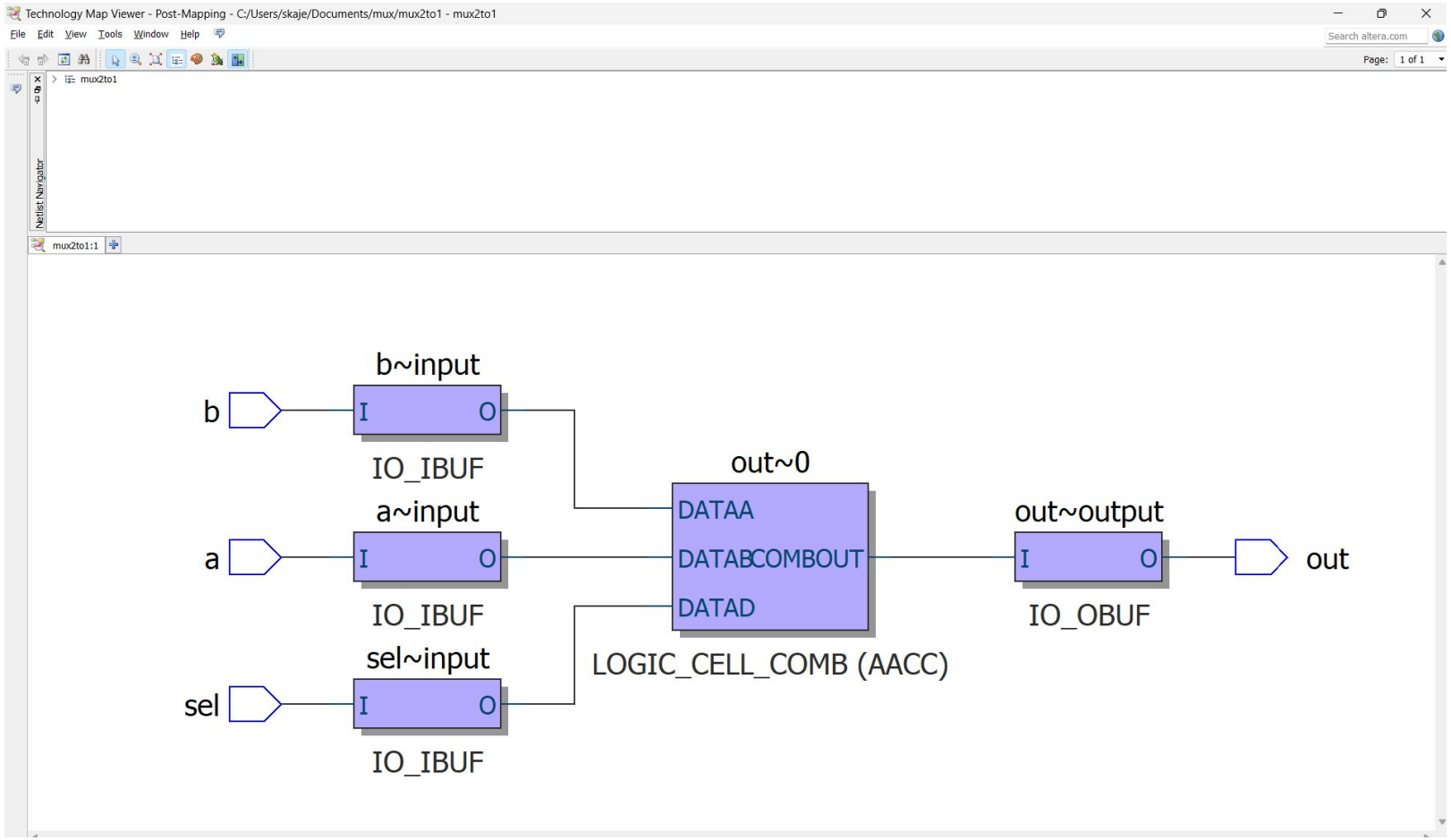
Synthesis: Using Quartus



Synthesis Results: RTL Viewer



Synthesis Results: Technology Map Viewer



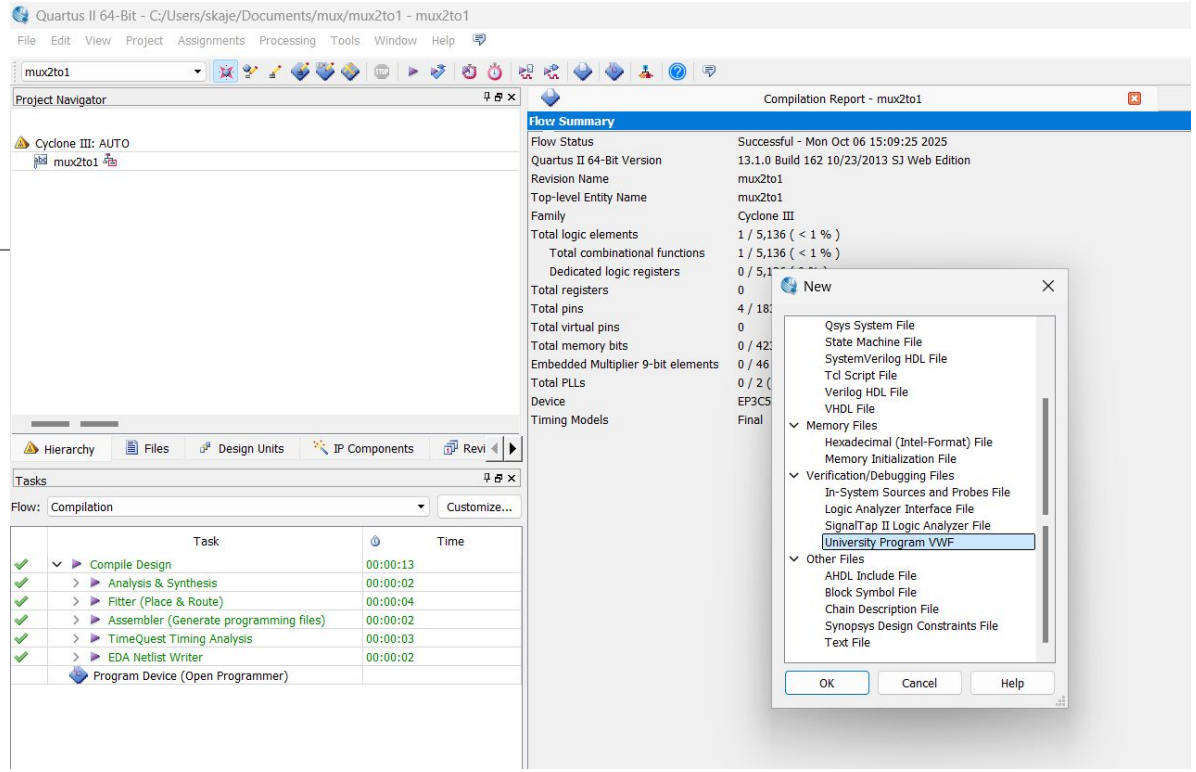
Functional Simulation (Post Synthesis)

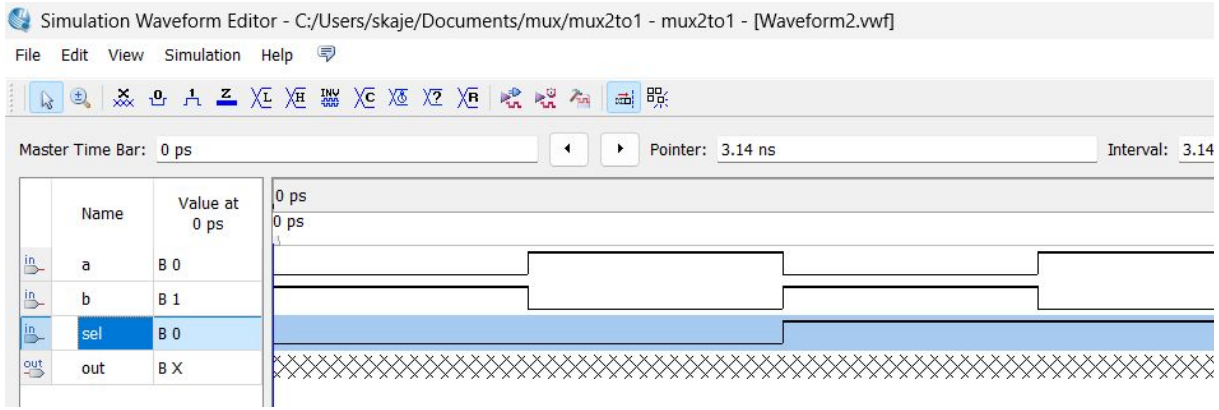
To verify the design functionality after the synthesis process has been completed

Ignores timing-related issues

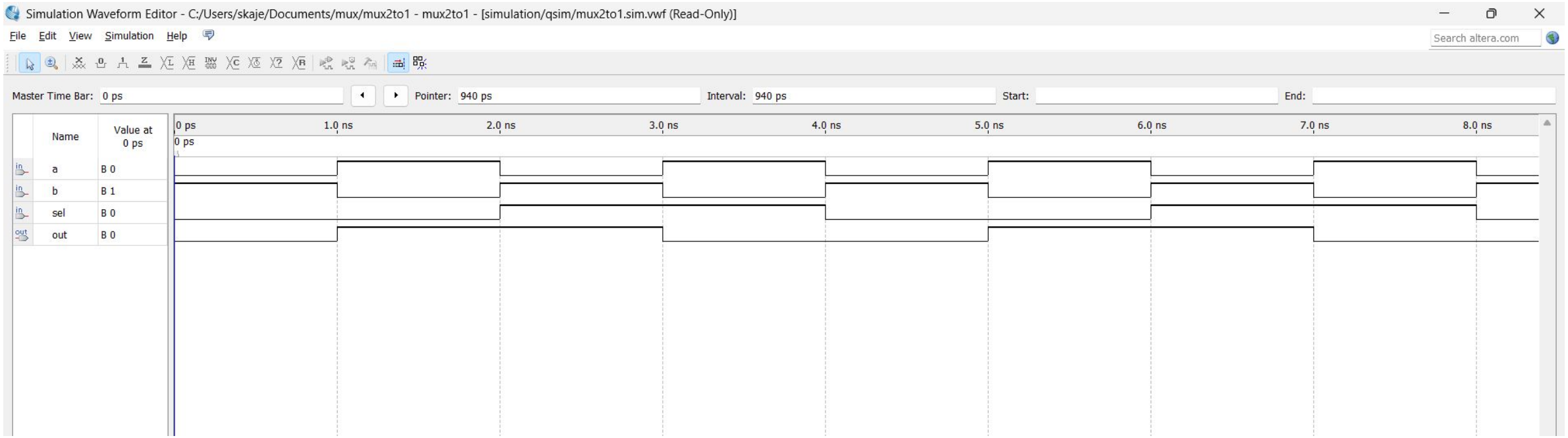
Functional Simulation

- Create **Vector Waveform File (.vwf):**
  In Quartus, create a new Vector Waveform File (File > New > Verification/Programming Files > Vector Waveform File)
- Use the Node Finder to add our mux2to1's input and output pins
- Define test vectors (waveforms) for our inputs





Functional Simulation



Timing or Gate-level Simulation (At Implementation)

Verifies that a design **still works correctly** after it has been placed and routed onto the target FPGA, factoring in **real-world delays** from the logic elements and physical wiring

Verifies the design functionality after the synthesis process has been completed

Gives the most accurate picture of our design behavior

Takes into account the target FPGA chip and all the logic blocks functionality, wiring, and delays

Timing Simulation

1. Full Compilation and Netlist Generation:

Run a **Full Compilation** in Quartus

Configure **Assignments > Settings > EDA Tool Settings > Simulation**

Quartus will generate a **post-fit netlist** (e.g., .vo or .vho) and an **SDF** (Standard Delay Format, .sdo) file in the simulation/modelsim directory.
The **SDF file contains the precise timing delays**

mux2to1_6_1200mv_85c_v_slow.sdo

```
19
20  //
21  // This file contains Slow Corner delays for the design using part EP3C5F256C6,
22  // with speed grade 6, core voltage 1.2V, and temperature 85 Celsius
23  //
24
25  //
26  // This SDF file should be used for ModelSim-Altera (Verilog) only
27  //
28
29  (DELAYFILE
30    (SDFVERSION "2.1")
31    (DESIGN "mux2to1")
32    (DATE "10/06/2025 14:59:04")
33    (VENDOR "Altera")
34    (PROGRAM "Quartus II 64-Bit")
35    (VERSION "Version 13.1.0 Build 162 10/23/2013 SJ Web Edition")
36    (DIVIDER .)
37    (TIMESCALE 1 ps)
38
39    (CELL
40      (CELLTYPE "cycloneiii_io_obuf")
41      (INSTANCE out\~output)
42      (DELAY
43        (ABSOLUTE
44          (PORT i (1379:1379:1379) (1436:1436:1436))
45          (IOPATH i o (1969:1969:1969) (1953:1953:1953))
46        )
```

mux2to1_6_1200mv_85c_slow.vo

```
22  // Device: Altera EP3C5F256C6 Package FBGA256
23  //
24
25  //
26  // This Verilog file should be used for ModelSim-Altera (Verilog) only
27  //
28
29  `timescale 1 ps/ 1 ps
30
31  module mux2to1 (
32      a,
33      b,
34      sel,
35      out);
36  input   a;
37  input   b;
38  input   sel;
39  output  out;
40
41  // Design Ports Information
42  // out  =>  Location: PIN_D3,    I/O Standard: 2.5 V,    Current Strength: Default
43  // b    =>  Location: PIN_R1,    I/O Standard: 2.5 V,    Current Strength: Default
44  // a    =>  Location: PIN_L8,    I/O Standard: 2.5 V,    Current Strength: Default
45  // sel  =>  Location: PIN_P2,    I/O Standard: 2.5 V,    Current Strength: Default
46
47
48  wire gnd;
49  wire vcc;
50  wire unknown;
51
52  assign gnd = 1'b0;
53  assign vcc = 1'b1;
54  assign unknown = 1'bx;
55
56  tri1 devclrn;
57  tri1 devpor;
58  tri1 devoe;
59  // synopsys translate_off
60  initial $sdf_annotate("mux2to1_6_1200mv_85c_v_slow.sdo");
61  // synopsys translate_on
62
63  wire \out~output_o ;
64  wire \sel~input_o ;
65  wire \b~input_o ;
66  wire \a~input_o ;
67  wire \out~0_combout ;
68
69
70  // Location: IOOBUF_X1_Y24_N9
71  cycloneiii_io_obuf \out~output (
72      .i(\out~0_combout ),
73      .oe(vcc),
74      .seriesterminationcontrol(16'b0000000000000000),
75      .devoe(devoe),
76      .o(\out~output_o ),
```

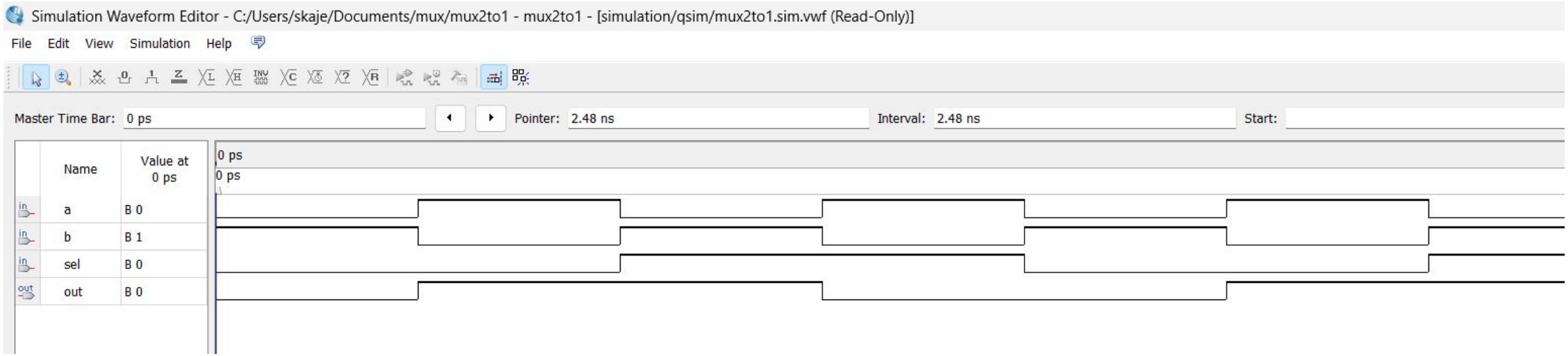
Timing Simulation

2. Create **Vector Waveform File (.vwf)**

---

3. Run Timing Simulation

4. In the resulting waveform, will see a measurable time delay between the input changes and the corresponding output change, which represents the **propagation delay** of the implemented mux



Part III

APPLICATIONS AND FUTURE

FPGAs Applications

CREDIT TO ORIGINAL SOURCE
HTTPS://HARDWAREBEE.COM/FPGA-COMMON-APPLICATIONS/

FPGAs Applications

Aerospace & Defense: Used for radar, communication systems, and flight control in UAVs due to their high-speed signal processing.

Medical Devices: Integrated into patient monitoring systems, imaging equipment (MRI, CT), and therapy delivery devices for their low-power, high-precision control and data processing capabilities.

Automotive: Found in embedded systems for motor control, sensor management, and flexible connectivity solutions as cars become more digital.

Consumer Electronics: Accelerate image and video processing in cameras and projectors, enabling features like HDR and 4K/8K video.

Data Centers: Used in servers and networking infrastructure to provide low-latency, high-bandwidth performance for cloud services and storage.

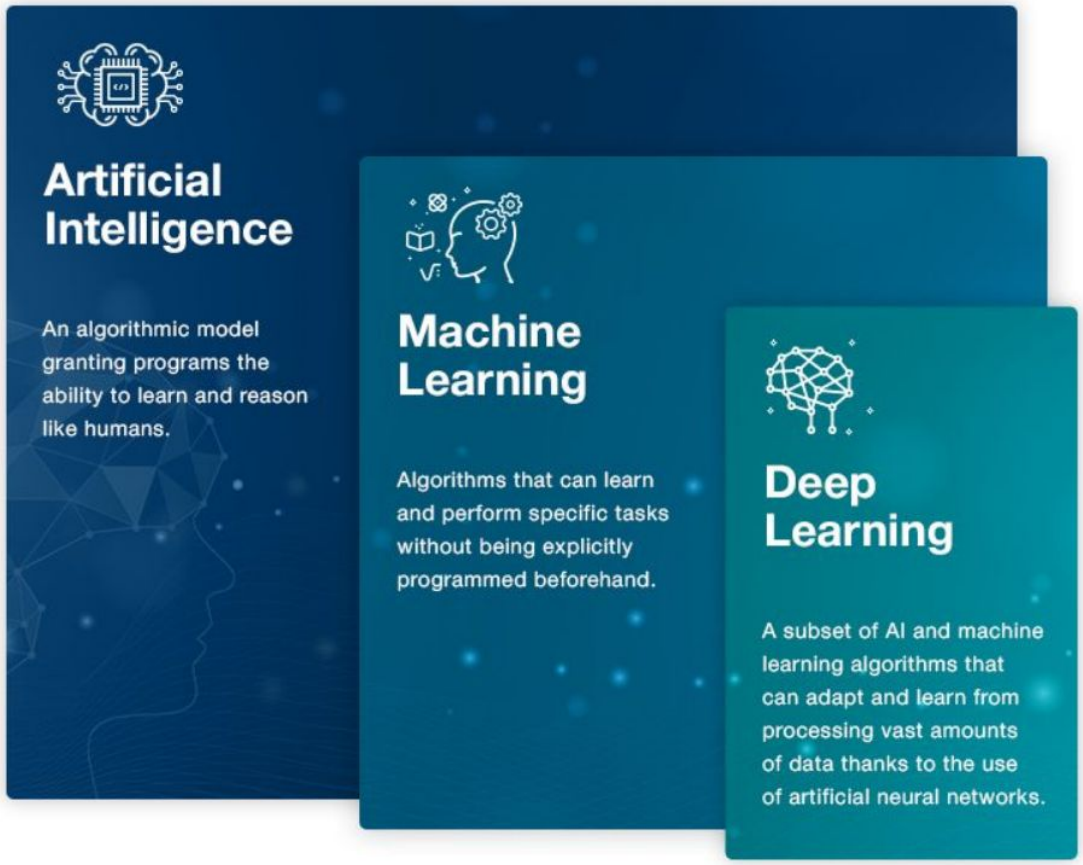
AI and Machine Learning: FPGAs can be customized to accelerate AI tasks and machine learning inference, providing a flexible and efficient solution for edge AI and data centers.

Robotics: Incorporate technologies for multi-axis motor control, machine learning, and functional safety, essential for modern smart robotics.

Telecommunications: Implement high-speed network interfaces and digital signal processing for network infrastructure and satellite communications.

Prototyping: FPGAs serve as valuable platforms for developing and prototyping custom integrated circuits (ASICs) before manufacturing them.

AI/Machine Learning



CREDIT TO ORIGINAL SOURCE
HTTPS://WWW.APRIORIT.COM/DEV-BLOG/586-FPGAS-FOR-AI

Neural Networks

The inspiration for neural networks comes from the **biological neurons** in the human brain, which communicate through electrical signals

Convolutional Neural Networks (CNNs) in AI are for image processing tasks
They use convolution layers to detect patterns in images

Recurrent Neural Networks (RNNs) are designed to handle sequential data
Example usages: Speech recognition, Automated text generation

Transformers and autoencoders for transforming and reconstructing data
Example usage: **NLP** (Natural Language Processing) – used in translation tools, chatbots

CREDIT TO ORIGINAL SOURCE
HTTPS://WWW.IBM.COM/THINK/TOPICS/NEURAL-NETWORKS
HTTPS://HARDWAREBEE.COM/FPGA-FOR-AI-ARTIFICIAL-INTELLIGENCE-ULTIMATE-GUIDE/

AI/Machine Learning acceleration

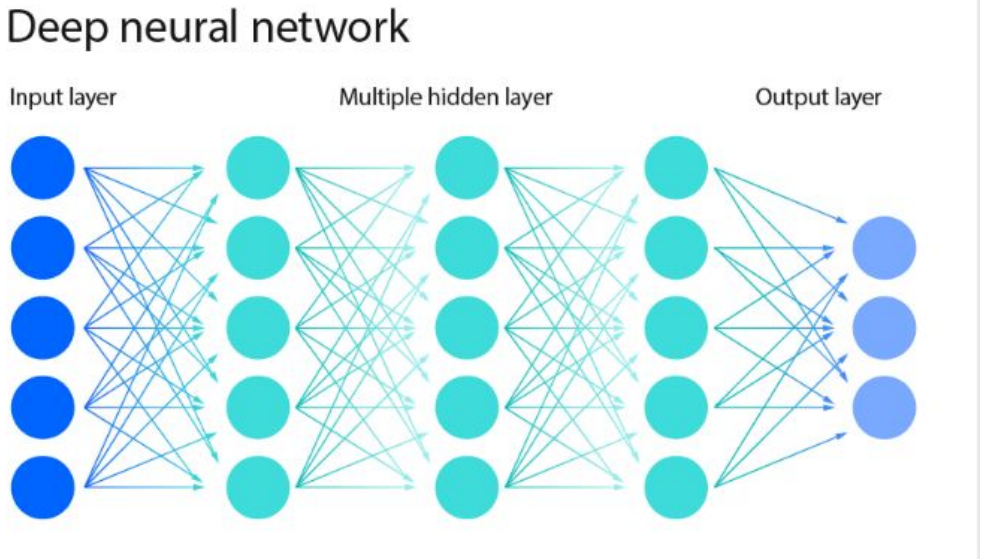
Convolutional Neural Networks (CNN)

Input Layer: holds the raw features (X1, X2, X3…)

Hidden layers: consist of artificial neurons (or nodes) that transform inputs into new representations

Input features, **multiplied** by their associated **weights and added bias** to pass from one layer to the next layer, eventually arriving at the final output layer. Called **“Linear Transformation”**

Output layer: Further, a nonlinear activation function (tanh, sigmoid, ReLU) is added to produce the final prediction



CREDIT TO ORIGINAL SOURCE
HTTPS://WWW.IBM.COM/THINK/TOPICS/NEURAL-NETWORKS
HTTPS://HARDWAREBEE.COM/FPGA-FOR-AI-ARTIFICIAL-INTELLIGENCE-ULTIMATE-GUIDE/

AI/Machine Learning acceleration

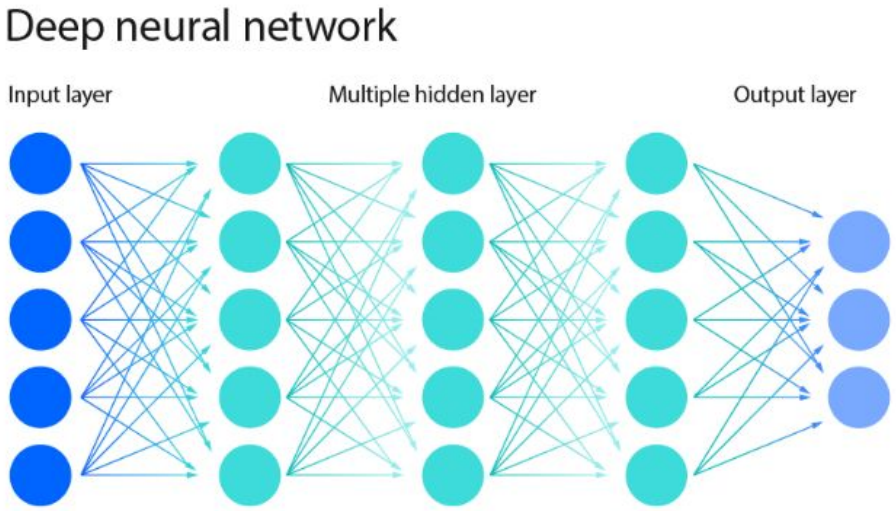
A simple Example: **Spam Detection**

An **email is fed into the network**, and features such as words or phrases like "prize," "money" are used as **inputs**

The early neurons process the importance of each signal, while later layers combine this information into higher-level cues that capture context and tone

Weights act like dials that control how strongly each input feature influences the decision, ex: “prize” has more weight than “hello”

The **final layer** computes a probability of **whether the email is spam**, and if that **probability is high enough**, **the email is flagged**



CREDIT TO ORIGINAL SOURCE
HTTPS://WWW.IBM.COM/THINK/TOPICS/NEURAL-NETWORKS
HTTPS://HARDWAREBEE.COM/FPGA-FOR-AI-ARTIFICIAL-INTELLIGENCE-ULTIMATE-GUIDE/

FPGAs in AI/Machine Learning

Flexibility and Reconfigurability: Tailored configurations that meet the specific demands of AI workloads

Parallel Processing Capabilities: Handling multiple operations simultaneously, essential for the **matrix and vector computations** fundamental to machine learning and neural networks

Real-Time Processing

Energy Efficiency: The CLBs of an FPGA can be optimized for **convolution operations**, improving computational power while maintaining **power efficiency**

CREDIT TO ORIGINAL SOURCE
HTTPS://FIDUS.COM/BLOG/THE-ROLE-OF-FPGAS-IN-AI-ACCELERATION/

FPGAs in AI/Machine Learning

Enhanced Security and Data Privacy: FPGAs provide data security by enabling **local, on-device processing** that minimizes data transmission to external servers or the cloud, and can incorporate encryption

Scalability Across AI Applications : From low-power IoT deployments to high-performance data center systems

Long-Term Cost Efficiency

CREDIT TO ORIGINAL SOURCE
HTTPS://FIDUS.COM/BLOG/THE-ROLE-OF-FPGAS-IN-AI-ACCELERATION/

Medical Imaging Device (CT/IGT)

Main components

- Operating Console
- X-ray tube
- High voltage Generator

- Produces a controlled electron flow to generate an X-ray beam
- Operates at high voltages between **50 to 150 kV**
- Tube currents range from **10 to 1200 mA**

CREDIT TO ORIGINAL SOURCE
HTTPS://WWW.GEHEALTHCARE.COM/PRODUCTS/COMPUTED-TOMOGRAPHY

Medical Imaging

Can be efficiently controlled by the FPGAs on the Control Board (along with other ICs such as Microcontroller, Memory, DSP)

Adjusts tube voltage (kV), current (mA), and exposure time (s) to ensure proper X-ray quality and quantity

Adjusting these parameters **optimizes image quality**

Proper settings **reduce patient radiation exposure**

Tools and Courses

HDL Simulator:
https://edaplayground.com/

https://quicksilicon.in/course/rtl-design/module/sequence-generator

Courses and Webinars:

Coursera

https://www.doulos.com/events/webinars/deep-learning-with-fpgas/

Q & A