DVA494

Programming of Reliable Embedded Systems

Obed Mogaka | Hamid Mousavi | Masoud Daneshtalab

IDT, Malardalens University

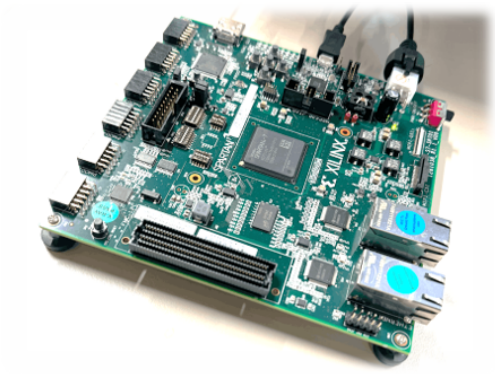
January 12, 2026

Today's Agenda

- What are FPGAs?
- Forms of FPGAs
- FPGA Architecture.
- Configuring FPGAs.
- Applications of the FPGAs.
- Mini-Project.

What is an FPGA?

- An FPGA is an **Integrated Circuit (IC)** that primarily deals with digital electronics.
- It is **Field-Programmable**:
  - Configurable *outside the factory* by the user.
  - Design is specified using HDLs (e.g., VHDL) or schematics.
  - Software tools generate a programming file (bitstream).
  - This bitstream configures the FPGA to implement your custom digital circuit.
- **Key Benefit: Agile Hardware Development**
  - Build custom electronic circuits without breadboards or soldering.
  - Avoid the high cost and long lead times of custom chip fabrication.
  - Quickly prototype and test complex digital designs.



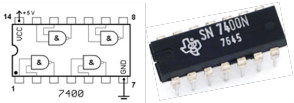
Historical Context: The Pre-FPGA Era

Discrete Logic Implementation

- Digital systems were originally built using individual logic chips (e.g., 7400 series TTL).
- Each chip contained only a few gates (e.g., 4 NAND gates) or a single flip-flop.

Why was this painful?

- **High Complexity:** A simple function required dozens of chips.
- **Inflexible:** "Debugging" or changing logic meant physically cutting wires and soldering new connections.
- **Reliability Issues:** Thousands of physical connections (soldered joints, wire wraps) created points of failure.
- **Size & Power:** Large PCBs consuming significant power and space.



A 7400 Quad NAND Gate

The System: Prone to wiring errors and huge

The Landscape of Digital Logic Technologies

1. Standard Logic (Fixed)

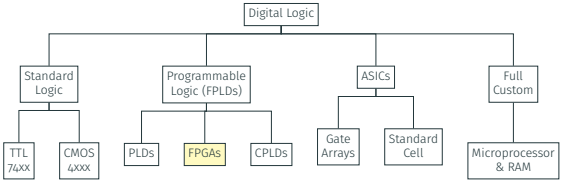
- Off-the-shelf components (e.g., 7400 TTL).
- Good for glue logic, bad for complex systems.

2. ASICs (Application-Specific ICs)

- Custom chips designed for a specific purpose.
- **Pros:** Highest performance, lowest power.
- **Cons:** Extremely expensive design cost (NRE), cannot be changed once manufactured.

3. Programmable Logic (FPLDs)

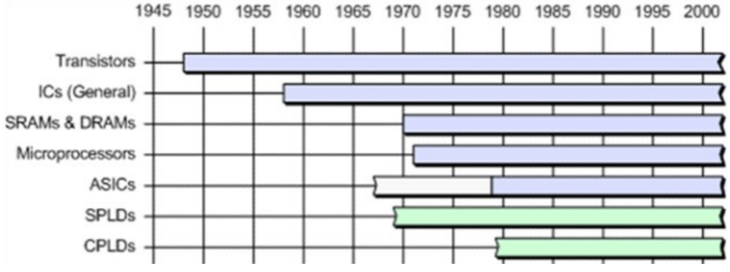
- **Where FPGAs live.**
- Hardware that can be reconfigured after manufacturing.
- Bridges the gap between the flexibility of software and the speed of hardware.



Classification of Digital Logic

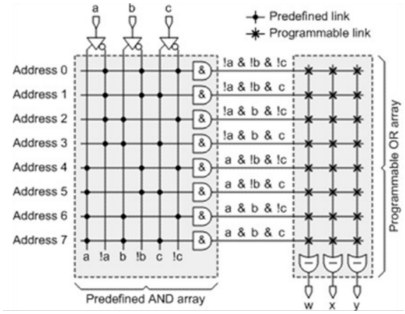
The Evolution of Programmable Electronics

- **1940s – 1960s: The Foundation**
  - **Transistors** (1947) replace vacuum tubes.
  - **Integrated Circuits (ICs)** combine transistors into single chips.
- **1970s: The Rise of Programmability**
  - **Microprocessors** introduced software programmability.
  - **SPLDs** (Simple PLDs) emerged as the first *hardware* programmable devices (e.g., PALs, PLAs) to replace discrete glue logic.
- **1980s: Increasing Density**
  - **CPLDs** (Complex PLDs) integrated multiple SPLD blocks to handle larger logic functions, bridging the gap toward modern FPGAs.



A Brief History: Simple PLDs (sPLDs)

(a) Programmable Read Only Memory (PROM)



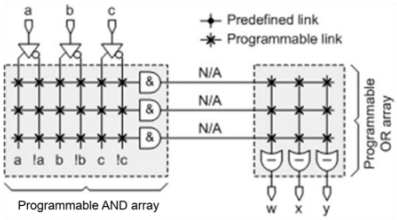
Architecture:

- **Fixed AND Plane:** A full address decoder. Every possible input combination activates exactly one product term (one row).
- **Programmable OR Plane:** The memory content. The user programs which product terms sum together to form the output.

Drawbacks for Logic Design:

- **Inefficient:** Requires a full decoder (2^N product terms for N inputs). Most logic functions only need a few terms, wasting vast amounts of the array.
- **Limited Inputs:** Scaling the inputs exponentially increases the size (e.g., 20 inputs = 1 million rows).

(b) Programmable Logic Arrays (PLAs)



PLA Structure: Two Programmable Planes

Architecture:

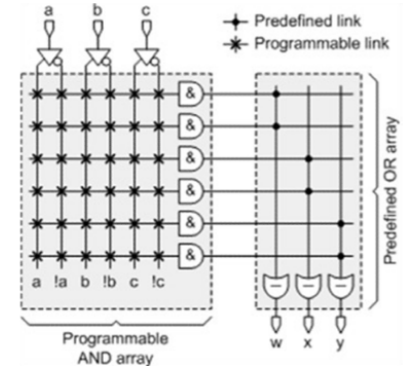
- **Two Levels of Programmability:** Unlike PROMs, *both* the AND plane and the OR plane are programmable.
- Implements Sum-of-Products (SOP) logic efficiently by allowing users to define *only* the product terms they need.

Trade-offs:

- **Pro:** Much more flexible than PROMs; can handle larger inputs.
- **Con:** Two programmable planes significantly increased signal delay (**poor speed**).
- **Con:** More complex and expensive to manufacture than devices with fixed planes.

A Brief History: Simple PLDs (sPLDs)

(c) Programmable Array Logic (PAL)



PAL Structure: Programmable AND, Fixed OR

Architecture: The "Goldilocks" Solution

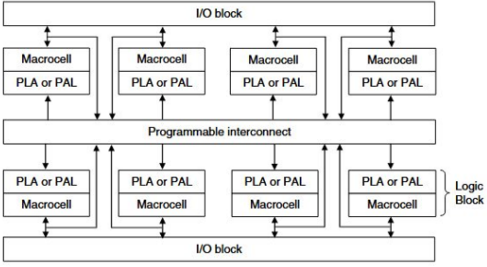
- **Programmable AND Plane:** Users define their product terms.
- **Fixed OR Plane:** Faster and cheaper than PLAs because it removes a level of programmable delay.

Key Innovation: Registered Outputs

Many PALs included Flip-Flops at the output. This allowed them to implement **Sequential Logic** (Finite State Machines) on a single chip, revolutionizing digital design.

Scaling Up: Complex Programmable Logic Devices (CPLDs)

- **The Motivation:** Single PALs were too small for complex systems. Designers needed to integrate multiple PALs into one package.
- **The Solution:** The CPLD architecture.



CPLD Internal Structure

Architecture Overview:

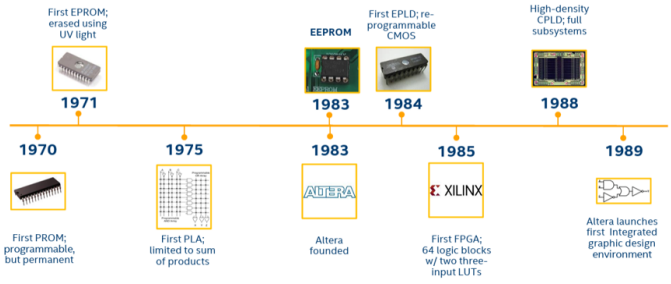
- **Macrocells/Logic Blocks:** Essentially multiple SPLDs (PAL-like structures) on a single chip.
- **Programmable Interconnect:** A central switch matrix that connects signals between different Logic Blocks and I/O pins.

Characteristics:

- **Non-Volatile:** Configuration is stored in on-chip EEPROM/Flash (instant-on, no external memory needed).
- **Predictable Timing:** Uniform delays due to the fixed interconnect structure.
- The direct forerunner to the FPGA.

The Birth of the FPGA

- By the mid-80s, designers needed more than just CPLDs. They needed higher density, more flexibility, and the ability to prototype complex systems without custom ASICs.
- **The Solution:** The Field Programmable Gate Array.



1985: A Pivot Point: Xilinx introduces the **XC2064**, the world's first commercial FPGA.

XILINX

- **Innovation:** Instead of AND/OR planes (like PALs/CPLDs), it used **Look-Up Tables (LUTs)** and a flexible interconnect fabric.
- **Specs:** Only 64 Logic Blocks (CLBs) – tiny by today's standards, but revolutionary at the time.

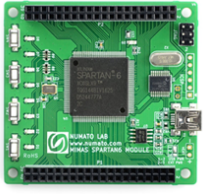
Formal Definition: Field Programmable Gate Array (FPGA)

What is an FPGA?

An FPGA is a **software-reconfigurable hardware substrate**. Unlike a processor that executes instructions, an FPGA configures physical digital circuits to perform a task.

Key Pillars of Reconfigurability:

1. **Reconfigurable Functions:** The ability to implement any logic gate or combinatorial function (via Look-Up Tables).
2. **Reconfigurable Interconnect:** A programmable wiring network that connects functions together.
3. **Reconfigurable I/O:** Flexible pins that can support various voltage standards and protocols.

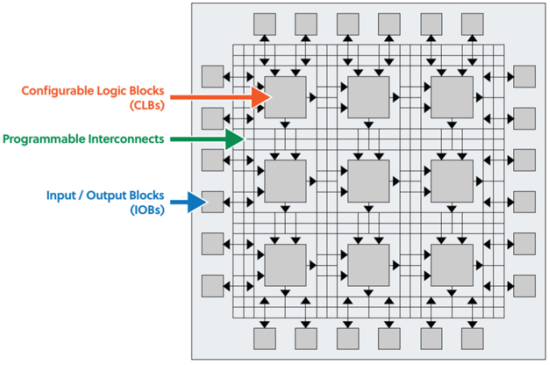


A High-Level Overview of FPGA Architecture

- An FPGA is composed of a "sea" of programmable resources.
- We create hardware circuits by configuring these blocks and the routing between them.

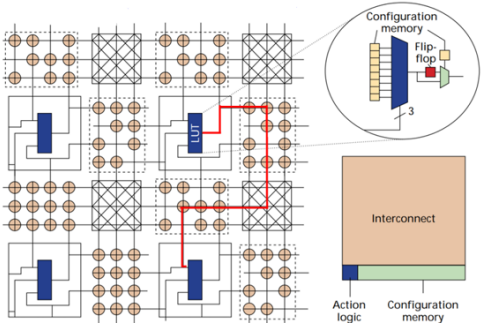
The Three Pillars of FPGA Fabric:

1. **Configurable Logic Blocks (CLBs):**
   - The "brain" of the FPGA. Contains LUTs (for logic) and Flip-Flops (for memory).
2. **Programmable Interconnects:**
   - The "nervous system." A grid of switch matrices that route signals between CLBs.
3. **Input/Output Blocks (IOBs):**
   - The "interface." Programmable pads that connect internal logic to the outside world pins.



FPGA Architecture: Inside the Logic Block

- The FPGA fabric consists of repeating tiles of logic and routing.
- **The Fundamental Building Blocks:**
  1. **Look-Up Tables (LUTs):** These implement the combinatorial logic (truth tables).
  2. **Flip-Flops (Registers):** These provide sequential memory elements for state.
  3. **Programmable Switches:** These route signals between blocks.



How it works:

- **Configuration Memory:** SRAM bits that hold your design.
- **Action Logic:** The actual hardware resource (LUT + Register) that performs the task.
- By changing the configuration bits, we change the truth table of the LUT and the routing path, effectively "rewiring" the chip.

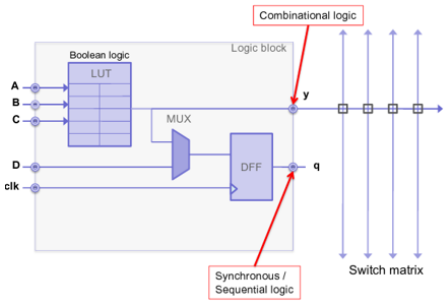
FPGA Architecture: The Basic Logic Element (BLE)

The Atomic Unit

- The BLE is the smallest functional unit in an FPGA.
- **Components:**
  1. **K-LUT:** Implements any boolean logic function of K inputs.
  2. **D Flip-Flop (DFF):** Stores state for sequential logic.
  3. **2:1 Multiplexer:** The "Bypass" switch.

Operating Modes

- **Combinational Mode:** The MUX selects the direct output of the LUT (bypassing the Flip-Flop).
- **Registered Mode:** The MUX selects the output of the Flip-Flop (synchronous logic).

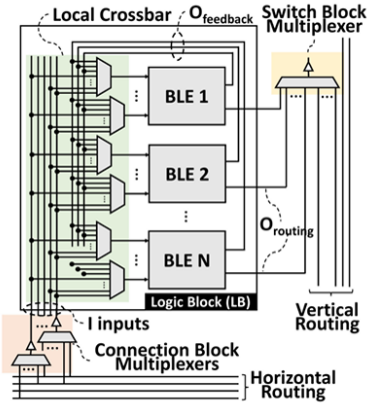


Hierarchy: BLEs are rarely solitary; they are typically clustered together (e.g., N=2, N=4) to form a single **Logic Block (LB)** or CLB.

Historical Note: *The first Xilinx FPGA (XC2000 series, 1984) used a cluster size of $N = 2$ with $K = 3$ (3-input LUTs).*

FPGA Architecture: The Logic Block (LB)

- **Hierarchy:** Single BLEs are too granular for efficient routing. FPGAs group multiple BLEs (e.g., N=4 to 10) into a single cluster called a **Logic Block (LB)** or **Configurable Logic Block (CLB)**.



Inside the Logic Block:

- **BLE Array (1 to N):** The computational core.
- **Local Crossbar :** A fast, dedicated internal network. It allows the output of *any* BLE in the cluster to feed back into the input of *any other* BLE in the same cluster without touching the slower global routing fabric.

Connections to the World:

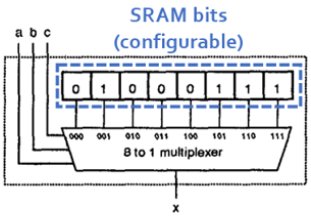
- **Switch Block MUX:** Routes outputs to the global fabric.
- **Connection Block MUX:** Selects inputs from the global fabric.

FPGA Architecture: The Look-Up Table (LUT)

- **What is it?** A small, fast memory (SRAM) used to implement combinational logic.
- **How it works:** Instead of using logic gates (AND, OR), the LUT stores the **Truth Table** of the function.
- **Configuration:** A K-input LUT requires 2^K bits of SRAM.

a	b	c	$x = ab + \bar{b}c$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Truth table



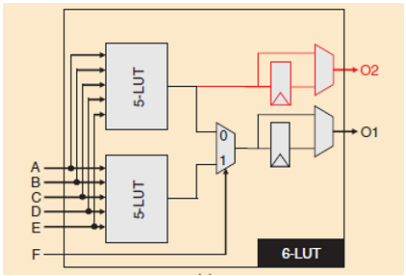
3-Input LUT

Key Advantages:

- **Universal:** Can implement *any* logic function of K inputs.
- **Constant Delay:** The propagation delay through the LUT is fixed, regardless of whether the function is a simple AND gate or a complex XOR chain.

Advanced Architecture: The Fracturable LUT

- **The Trade-off:**
  - **Large LUTs (e.g., 6-input):** Faster (fewer logic levels) but waste area if you only need a 2-input AND gate.
  - **Small LUTs (e.g., 4-input):** Area efficient but slower for complex functions (more routing delay).
- The Solution: **Fracturable LUTs** (Adaptive Logic Modules).



How it works:

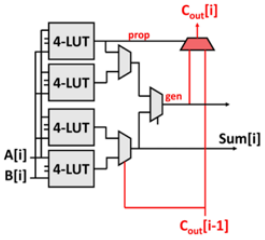
- A single large 6-LUT is built from smaller 5-LUTs.
- **Mode 1 (Performance):** Acts as one big 6-input LUT for complex logic.
- **Mode 2 (Efficiency):** "Fractures" into two independent smaller 5-input LUTs to implement two simple functions simultaneously.

Impact: This architecture (introduced by Altera Stratix II) achieves 14% higher performance with only 17% area cost compared to purely small LUTs.

FPGA Architecture: Hardened Arithmetic Circuitry

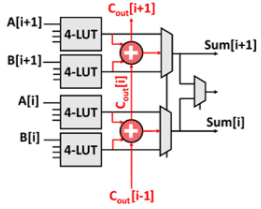
- **The Challenge:** Implementing adders using only LUTs is inefficient. The carry signal must ripple through multiple LUTs and routing switches, creating a very slow critical path.
- Modern FPGAs embed **Hardened Arithmetic Logic** (Carry Chains) directly next to the LUTs.

Xilinx/AMD Approach



Dedicated "Fast Carry" logic (red path) bypasses general routing for speed.

Intel/Altera Approach



Full hardened adders embedded in the logic block.

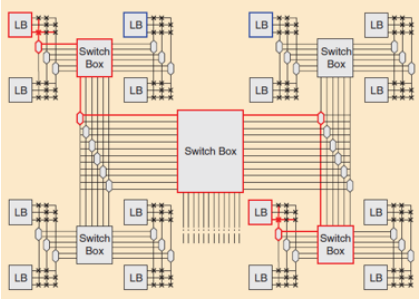
Impact on Design:

- Allows high-performance arithmetic (adders, counters, comparators) without using up valuable LUT resources.
- Synthesis tools automatically infer these structures when you write `A + B` in VHDL.

FPGA Architecture: Programmable Routing

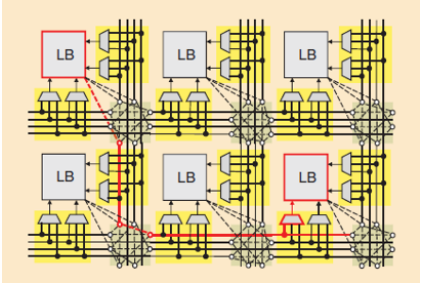
- **The Hidden Cost:** Routing wires and programmable switches account for **>50%** of the FPGA's silicon area.
- **Function:** Connects the inputs and outputs of Logic Blocks (LBs) to each other and to the I/O pins.

(a) Hierarchical Routing



Signals connect within local groups first, then move up to higher levels. Used in smaller FPGAs and CPLDs.

(b) Island-Style Routing

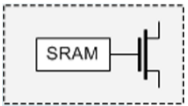


Logic Blocks (Islands) float in a sea of interconnects. Uses **Switch Boxes** to route signals North, South, East, or West. Standard for modern, large FPGAs.

FPGA Architecture: Switch Technologies

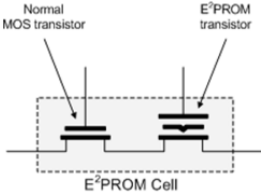
- The "programmable" part of an FPGA relies on a physical switch technology to configure interconnects and logic.

(a) SRAM



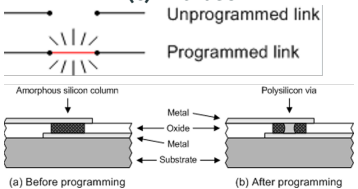
- **Mechanism:** A pass transistor controlled by a static memory bit.
- **Pros:** Fast, infinitely reprogrammable, uses standard CMOS process.
- **Cons:** Volatile (needs power to hold config), larger area.
- *Dominant in modern FPGAs.*

(b) Flash/EEPROM



- **Mechanism:** Floating-gate transistor stores charge to switch.
- **Pros:** Non-volatile (keeps config without power).
- **Cons:** Slower programming, requires special manufacturing steps.

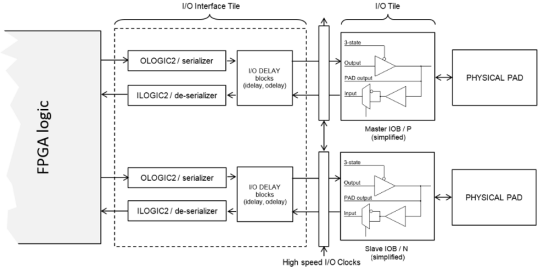
(c) Antifuse



- **Mechanism:** High voltage creates a permanent physical link.
- **Pros:** Highest density, extremely fast, radiation hard.
- **Cons:** One-time programmable (OTP) - cannot be changed once burned.

FPGA Architecture: Input/Output Blocks (IOBs)

- **The Interface:** IOBs are the programmable pads located around the periphery of the FPGA die.
- **Function:** They bridge the internal logic (running at low voltage, e.g., 1.0V) with external devices (running at 3.3V, 2.5V, 1.8V, etc.).



Configurable Features:

- **Voltage Standards:** LVCMOS, LVTTL, HSTL, SSTL.
- **Drive Strength:** Current output capability (e.g., 4mA to 24mA).
- **Slew Rate:** Control how fast the signal transitions (Fast/Slow) to manage noise.
- **Termination:** On-die termination resistors to prevent signal reflections.

Advanced I/O: LVDS
Low-Voltage Differential Signaling

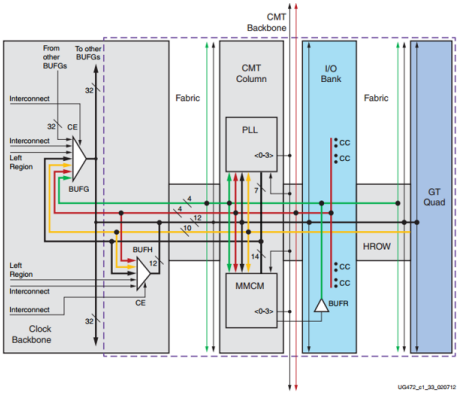
- Uses two pins per signal (***P*** and ***N***).
- High speed, high noise immunity.
- Essential for video, PCIe, and high-speed ADC/DACs.

FPGA Architecture: Clock Management

- **The Challenge:** Distributing a clock signal to 100,000+ flip-flops synchronously is physically difficult due to propagation delays (skew).
- **The Solution:** Dedicated **Global Clock Networks** (Clock Trees).

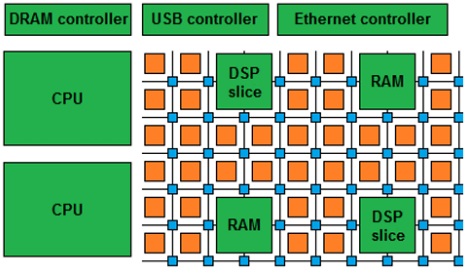
Key Components:

1. **BUFG (Global Buffer):** High-drive buffers that drive the clock tree to reach every flip-flop with minimal skew.
2. **MMCM / PLL (Mixed-Mode Clock Manager):** Hardened analog blocks that can:
   - **Synthesize** new frequencies (e.g., Input 100MHz → Output 200MHz).
   - **De-skew** clocks.
   - **Phase shift** clocks (e.g., 90-degree shift for DDR).



Modern FPGA Architecture: Beyond Just Logic

- Early FPGAs were homogeneous (just LUTs + Registers).
- **Modern FPGAs are Heterogeneous:** They integrate dedicated "hard" blocks to improve performance, area efficiency, and power.



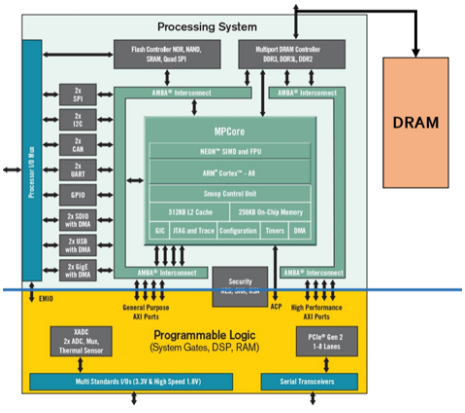
Common Hardened Blocks:

- **Block RAM (BRAM):** Dedicated memory blocks (e.g., 18Kb/36Kb) for storing data buffers.
- **DSP Slices:** Hardened multipliers and accumulators for high-speed signal processing.
- **Hard Processors:** Embedded CPUs (e.g., ARM cores).
- **Memory Controllers:** Hardened DDR/HBM PHYs.

Benefit: Using a hard DSP block is 10x faster and smaller than building a multiplier out of LUTs.

System-on-Chip (SoC): The Best of Both Worlds

- **Concept:** Combine a powerful standard processor (PS) with flexible FPGA fabric (PL) on the same silicon die.
- **Example:** Xilinx Zynq-7000 (ARM Cortex-A9 + Artix-7 Fabric).



1. Processing System (PS)

- **Role:** Runs the OS (Linux), manages peripherals (USB, Ethernet), and controls the application flow.
- **Fixed Hardware:** Fast, efficient, but not flexible.

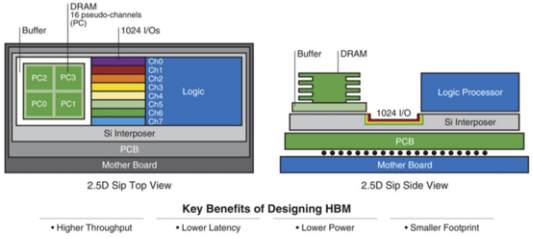
2. Programmable Logic (PL)

- **Role:** Hardware acceleration, custom interfaces, real-time signal processing.
- **Flexible:** Configured by the PS at boot time.

Connection: High-bandwidth AXI Interconnects link PS and PL.

Advanced Architecture: High Bandwidth Memory (HBM)

- **The Bottleneck:** Modern FPGAs can process data faster than external DDR4 memory can provide it.
- **The Solution: HBM (High Bandwidth Memory)**. Stack DRAM dies directly inside the FPGA package using 2.5D Silicon Interposer technology.



Why HBM?

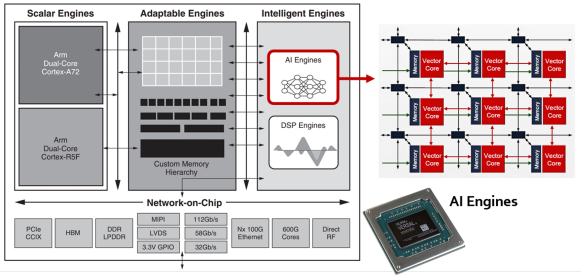
- **Massive Bandwidth:** Up to **460 GB/s** (vs 25 GB/s for DDR4).
- **Wide Interface:** 1024-bit wide bus per stack (vs 64-bit for DDR).
- **Lower Power:** Shorter traces (microns vs inches) mean less energy per bit transferred.

Calculation:

$$2 \text{ Stacks} \times 16 \text{ Channels} \times 64 \text{ bits} \times 1.8 \text{ Gbps} \approx 3.7 \text{ Tb/s}$$

Advanced Architecture: ACAP (Adaptive Compute Acceleration Platform)

- **The Next Evolution:** Xilinx Versal (ACAP) goes beyond the traditional FPGA/SoC model.
- **Key Innovation:** It integrates three distinct compute engines connected by a hardened **Network-on-Chip (NoC)**.



1. Scalar Engines (PS)

- ARM Cortex-A72 CPUs for OS and control.

2. Adaptable Engines (PL)

- Traditional FPGA fabric for custom logic.

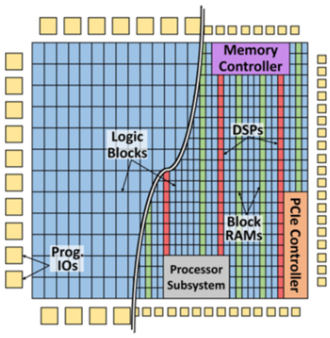
3. Intelligent Engines (AI/DSP)

- **Vector Cores:** Massively parallel VLIW processors designed for Machine Learning (AI) and DSP workloads.

Network-on-Chip (NoC):

- A hardened high-speed highway that replaces programmable routing for moving data across the chip.

Why FPGAs?



1. Massive Fine-Grained Parallelism

- Unlike a CPU (serial execution), an FPGA can execute thousands of operations simultaneously.
- Ideal for pipelined data processing.

2. Hardware Reconfigurability

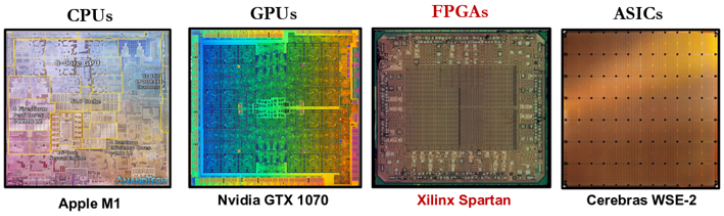
- **Numerical Accuracy:** Use exactly the bit-width you need (e.g., 5-bit math instead of standard 32-bit), saving resources.
- **Custom Memory Hierarchy:** Build caches and buffers exactly where the data is.

3. Performance Characteristics

- **Deterministic Latency:** No OS interrupts or cache misses to cause jitter.
- **Energy Efficiency:** Computes more operations per watt than general-purpose CPUs.

The Spectrum: FPGAs vs. Other ICs

- Every digital choice involves a trade-off between **Flexibility** (ease of programming) and **Efficiency** (performance/power).



FPGAs Come in Many Forms: Deployment Models

1. Data Center Acceleration

- **PCIe-Attached:** Similar to GPUs. Used for offloading heavy compute tasks (AI, Video Transcoding).
- **In-Network (SmartNICs):** FPGAs sitting directly on the network card to process packets at line rate (firewalls, routing).
- **In-Storage:** "Computational Storage." Processing data directly on the SSD to reduce data movement.

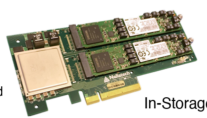
2. Deep Integration

- **CPU Integrated:** FPGA fabric inside the same package as the CPU (e.g., Intel Xeon + Arria 10). Ultra-low latency communication.

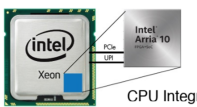
3. Embedded & Prototyping

- **Standalone Boards:** The classic development board (e.g., Basys 3) used for education, prototyping, and industrial controllers.

PCIe-Attached



In-Storage



CPU Integrated

In-Network

The FPGA Landscape: Major Manufacturers

The FPGA market is consolidated into two main tiers based on performance and application focus.

1. The High-Performance Duopoly

- **AMD (formerly Xilinx):** Market leader. Inventors of the FPGA. Famous for Virtex, Kintex, Zynq, and Versal (ACAP).
- **Intel (formerly Altera):** The main competitor. Famous for Stratix, Arria, Cyclone, and Agilex.

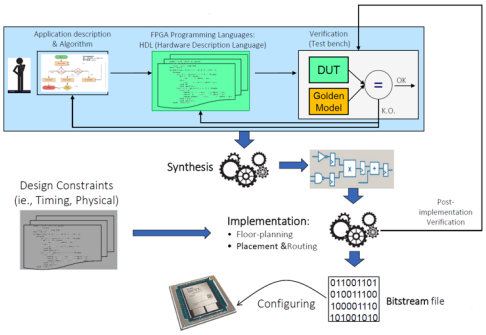
2. Low Power & Specialized

- **Lattice Semiconductor:** Specialists in low-power, small-footprint FPGAs (e.g., iCE40, MachXO) for mobile and control applications.
- **Microchip (formerly Microsemi/Actel):** Focus on high-reliability, radiation-tolerant, and mixed-signal FPGAs (e.g., PolarFire) for space and aviation.

***Note**: Other emerging players include Achronix (embedded FPGA IP) and Gowin (budget FPGAs).*

The FPGA Design Flow: From Concept to Bitstream

- The workflow is a multi-step translation process managed by vendor tools (e.g., Vivado, Quartus).



1. Synthesis

- Translates HDL (Behavioral) into a Netlist of generic gates and registers.

2. Implementation

- **Place:** Maps generic gates to specific physical LUTs/FFs on the die.
- **Route:** Configures the interconnects to wire them together.

3. Verification

- **Simulation:** Validating logic before synthesis.
- **Timing Analysis:** Ensuring the routed design meets clock constraints.

4. Bitstream Generation

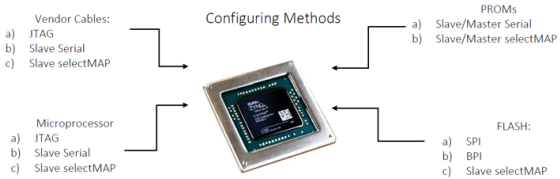
- Creating the binary file to load into the FPGA's configuration memory.

Configuring the FPGA: Loading the Bitstream

- Most FPGAs use **SRAM** to store their configuration (LUTs + Routing).
- But SRAM is **volatile**. When power is lost, the FPGA becomes a blank slate.
- **The Solution:** Store the bitstream in non-volatile external memory (Flash) and load it at power-up.

Configuration Methods:

- **JTAG:** Used for debugging/development. Connects PC to FPGA directly via cable (USB). volatile.
- **Master Serial/SPI:** The FPGA automatically reads from an external Flash chip at boot. Standard for production.
- **Slave SelectMAP/Parallel:** A CPU pushes the data into the FPGA (fastest method).



Note: Some FPGAs (e.g., Microchip PolarFire, Lattice MachXO) have internal Flash and are "Instant-On".

FPGA Application Domains

Defense

Space

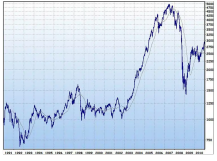
Aerospace

Telecom

Automotive

Microsoft
Google amazon
facebook

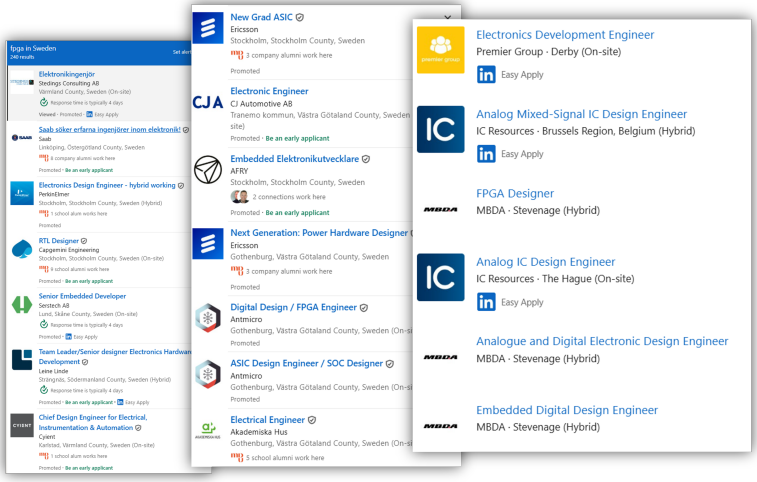
Data Centers



High-frequency trading

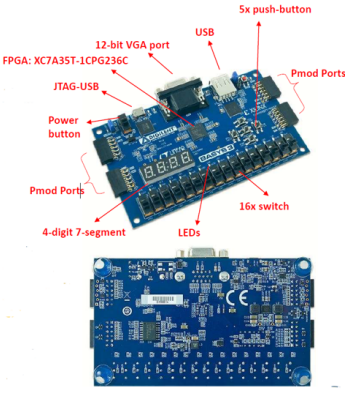
Cryptocurrency mining

Demand for Experts



The Course Hardware: Basys 3 Development Board

- We use the Digilent **Basys 3**, built around the Xilinx **Artix-7** FPGA.
- **Why Artix-7?** A modern, low-power FPGA perfect for learning digital logic, FSMs, and embedded processors.

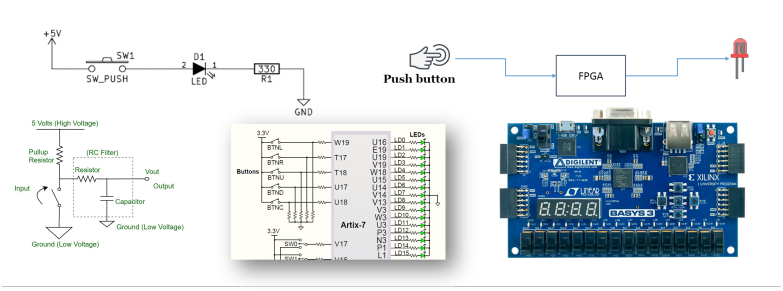


Key Features for Labs:

- **Logic:** 33,280 Logic Cells (enough for complex CPUs).
- **Inputs:** 16 Switches, 5 Push-buttons.
- **Outputs:** 16 LEDs, 4-Digit 7-Segment Display.
- **Clock:** Internal 100 MHz oscillator (adjustable).
- **Connectivity:** USB-UART, VGA, Pmod ports.

Mini-Project 1: Turn on the Lights!

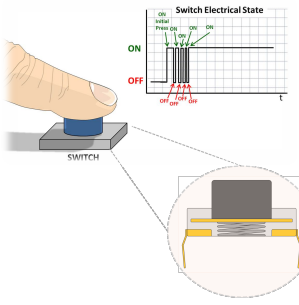
- **Objective:** Press the middle Push Button to turn on ALL 16 LEDs.
- **Why this matters:** It introduces the core workflow: Input → Processing → Output on real hardware.

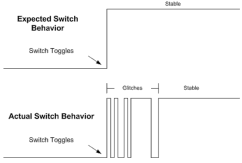


Challenge: Mechanical switches are noisy.
We cannot just connect the wire directly if we want reliable behavior.

The Physics of Switches: Bouncing

- When you press a button, metal contacts physically slam together.
- They don't make a perfect connection instantly; they **bounce** (vibrate) for milliseconds.
- **The Result:** A single press looks like dozens of rapid ON/OFF pulses to the high-speed FPGA.



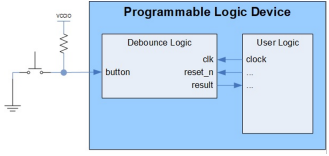


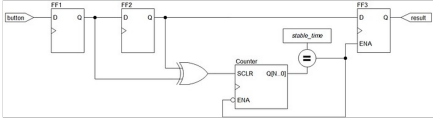
Consequence: If this feeds a counter, one press could count as 10 or 20 inputs!

The Solution: Digital Debouncing

Strategy: Ignore the chaos.

- We wait for the signal to remain **stable** for a minimum time (e.g., 10ms) before accepting it as a valid "High" or "Low".
- **Implementation:** A counter.
  - If Input = 1, start counting.
  - If Input drops to 0 (glitch), reset the counter.
  - Only when Counter reaches MAX (10ms) do we toggle the output.





Implementation: The Debouncer Entity

We create a reusable module `btn_debouncer`.

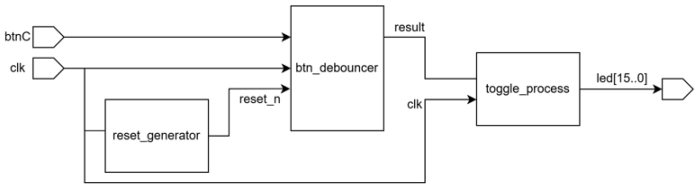
- **Generics:** Allow us to configure the clock speed and stability time.

```
entity btn_debouncer is
    generic (
        clk_freq    : INTEGER := 100_000_000; -- 100 MHz
        stable_time : INTEGER := 10;          -- 10 ms
    );
    port (
        clk     : in  STD_LOGIC;
        reset_n : in  STD_LOGIC;
        button  : in  STD_LOGIC; -- Noisy Input
        result  : out STD_LOGIC  -- Clean Output
    );
end btn_debouncer;
```

Note: We calculate the required clock cycles: $10ms \times 100MHz = 1,000,000$ *cycles.*

System Integration: The Top-Level Design

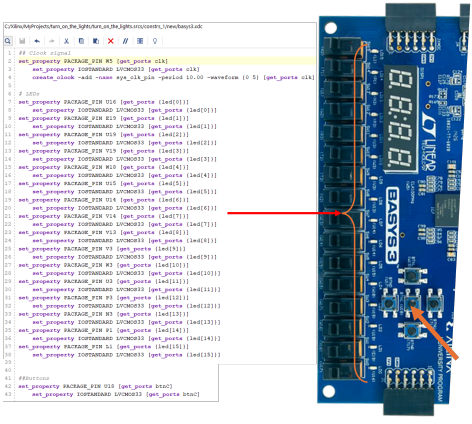
We connect our components in the top-level architecture.



- **reset_generator:** Ensures a clean startup state.
- **btn_debouncer:** Cleans the button signal.
- **toggle_process:** The logic that turns LEDs ON/OFF.

Connecting to Reality: The Constraints File (.xdc)

- The synthesis tool doesn't know that "led[0]" means "Pin U16".
- We use a **Xilinx Design Constraints (XDC)** file to map ports to physical package pins.



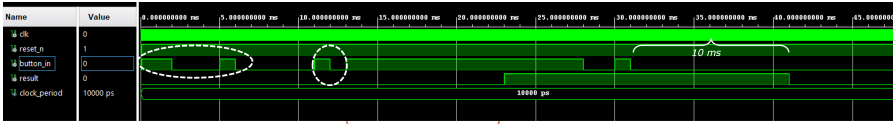
Key Commands:

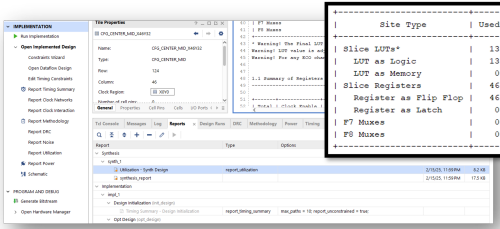
- `PACKAGE_PIN` : The physical location.
- `IOSTANDARD` : The voltage level (e.g., LVCMOS33 = 3.3V).

Example: *Mapping the middle button to Pin U18.*

Verification: Simulation Waveform

Before testing on hardware, we verify the timing in simulation.



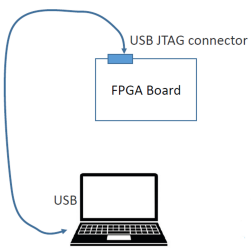


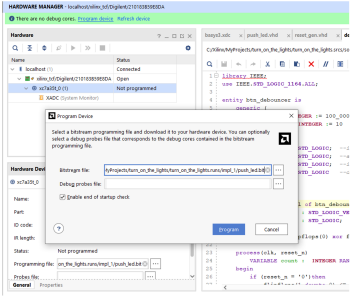
Site Type	Used	Fixed	Prohibited	Available	Util%
Slice LUTs*	13	0	0	20800	0.06
LUT as Logic	13	0	0	20800	0.06
LUT as Memory	0	0	0	9600	0.00
Slice Registers	46	0	0	41600	0.11
Register as Flip Flop	46	0	0	41600	0.11
Register as Latch	0	0	0	41600	0.00
F7 Muxes	0	0	0	16300	0.00
F8 Muxes	0	0	0	8150	0.00

- Notice the delay: The `result` signal goes high exactly 10ms *after* the noisy `button_in` stabilizes.

Final Step: Program the FPGA

1. **Synthesize & Implement:** Run the Vivado flow to generate the `.bit` file.
2. **Connect:** Plug in the Basys 3 via USB (JTAG).
3. **Program:** Use the Hardware Manager to load the bitstream.





Vivado Demo!