EE-334
Digital System Design

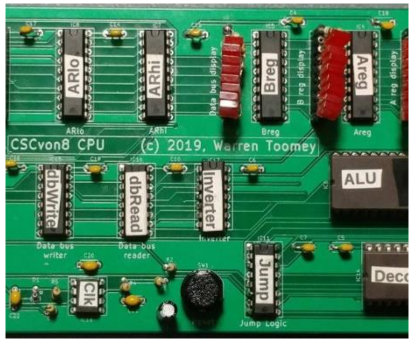
Custom Digital Circuits

Field Programmable Gate Arrays
(FPGAs)

Andreas Burg

Implementation Options for Digital Circuits

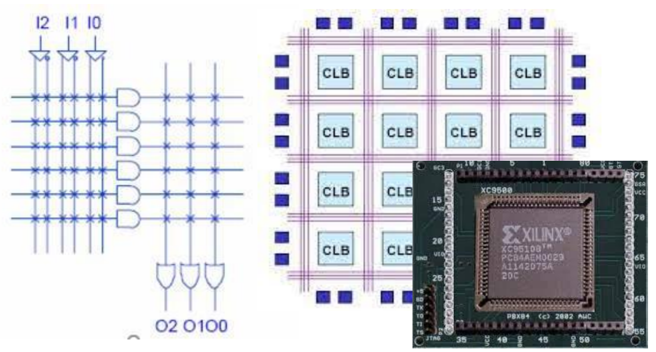
- Digital **systems** often **require** different **digital functions** which are **not** readily **available as standard** commercial off-the-shelf (COTS) **integrated circuits.**
- Options to realize those functions:



https://hackaday.com/2019/06/23/a-ttl-cpu-minimising-its-chip-count/

Composition of discrete Components

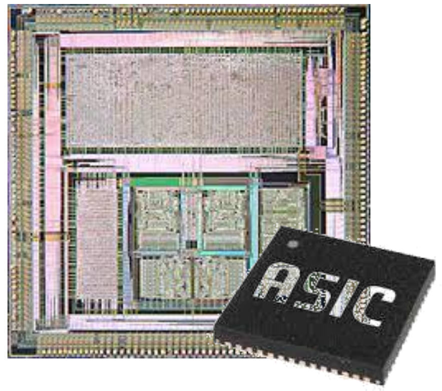
- Poor performance
- High unit cost
- Manageable NREs & R&D



Programmable Logic ICs

- Good performance
- Manageable unit cost
- Low NREs & R&D

Option of choice for medium volume



Application specific integrated circuit

- Great performance
- Very low unit cost
- Very high NREs & R&D

Basic Idea of Programmable Logic

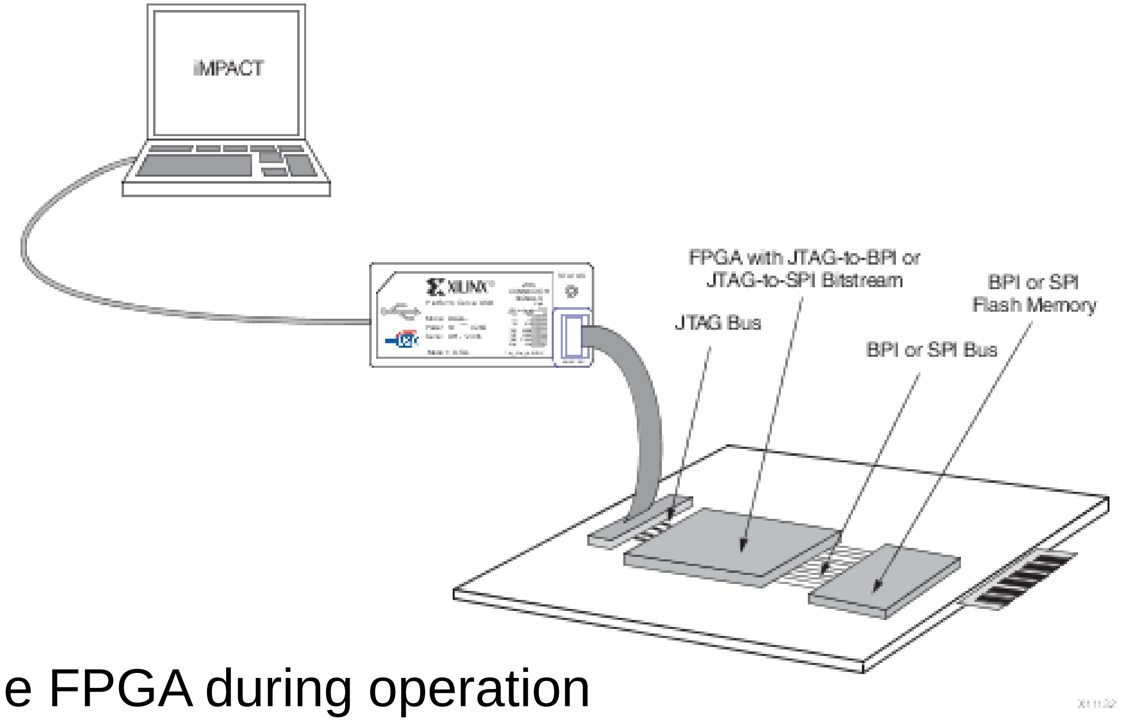
- **Objectives**:
  - **avoid high NREs** of a dedicated ASIC
  - **speed up** the **development process** for each application
  - provide **flexibility to update a system** in the field

- **Programmable logic device**: a **fixed** application specific IC (**ASIC**) **that**
  - **Integrates** all the **basic building blocks (logic & memory) to realize** different **digital circuits**
  - **Allows to change connections** between components after manufacturing (**at run time**)

- **After programming**, the **chip behaves** similar to a dedicated ASIC **as custom digital logic**, ideally with no visible difference

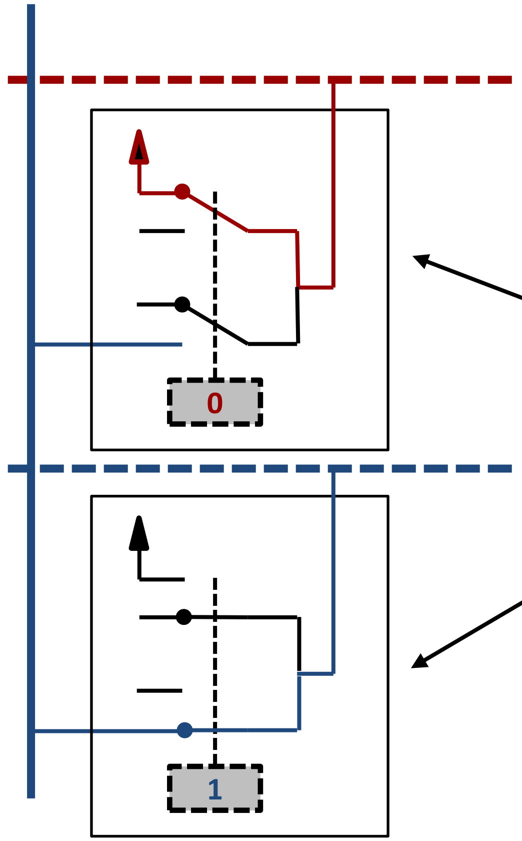
Programmable Logic Application

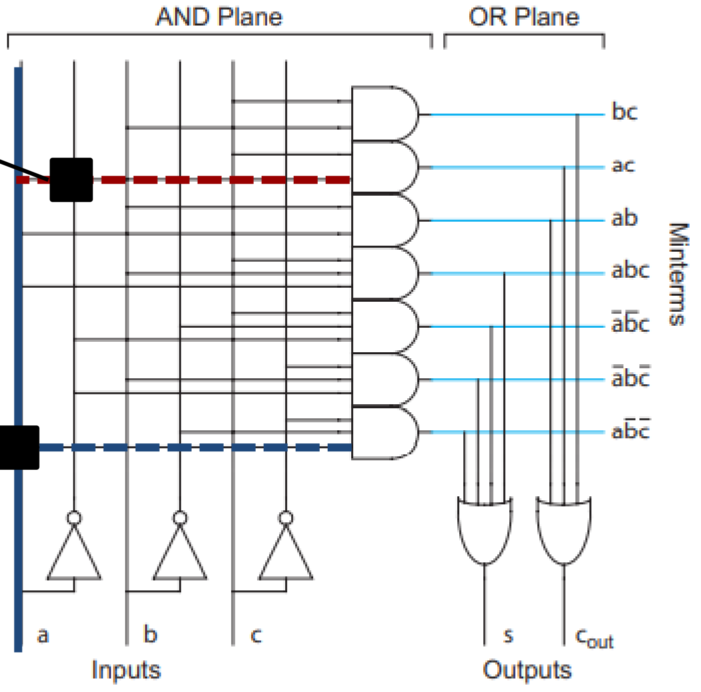
- **Development** of custom digital logic **based on same principles as for an ASIC**
  - **Synchronous RTL design**
  - **Similar** design tools **to** those for **semicustom ASIC design**
  - Device limitations impose **additional restrictions on complexity and abstraction**
- **Programming configures the device**
  - **from a PC** or **from** a separate **non-volatile** memory (EEPROM or FLASH)
  - **Some devices** can **store the configuration** in **on-chip** non-volatile memory
  - Dedicated interface (parallel or serial) or through JTAG (standardized test-access port)
  - **In-system programming**: programming while the chip is already integrated
  - **Run-time reconfiguration**: reconfigure parts of the FPGA during operation

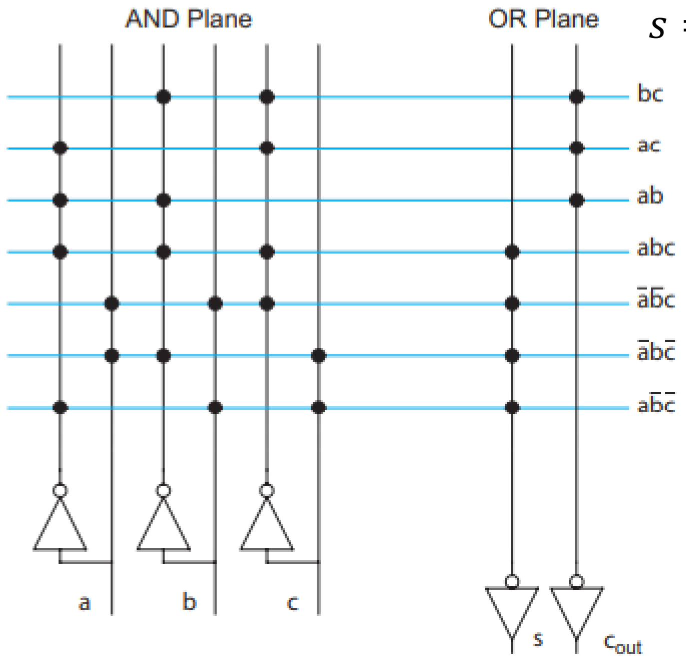


Principle of Programmable Logic

- Remember: **any logic function can be realized** with 2 logic stages **as**
  - **Sum-of-products**: 2-stage AND-OR function of inputs and inverted inputs
  - Product-of-sums: 2-stage OR-AND function
  - Programmable logic arrays: fixed **matrix of AND and OR gates**
    - **Configurable** (programmable) **switches connect** or dis-connect **them** to inputs





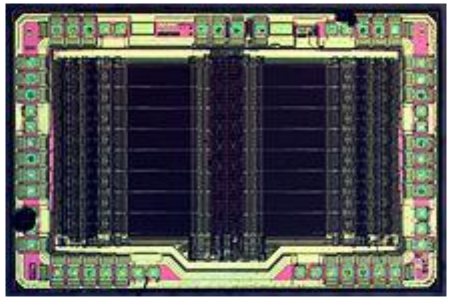


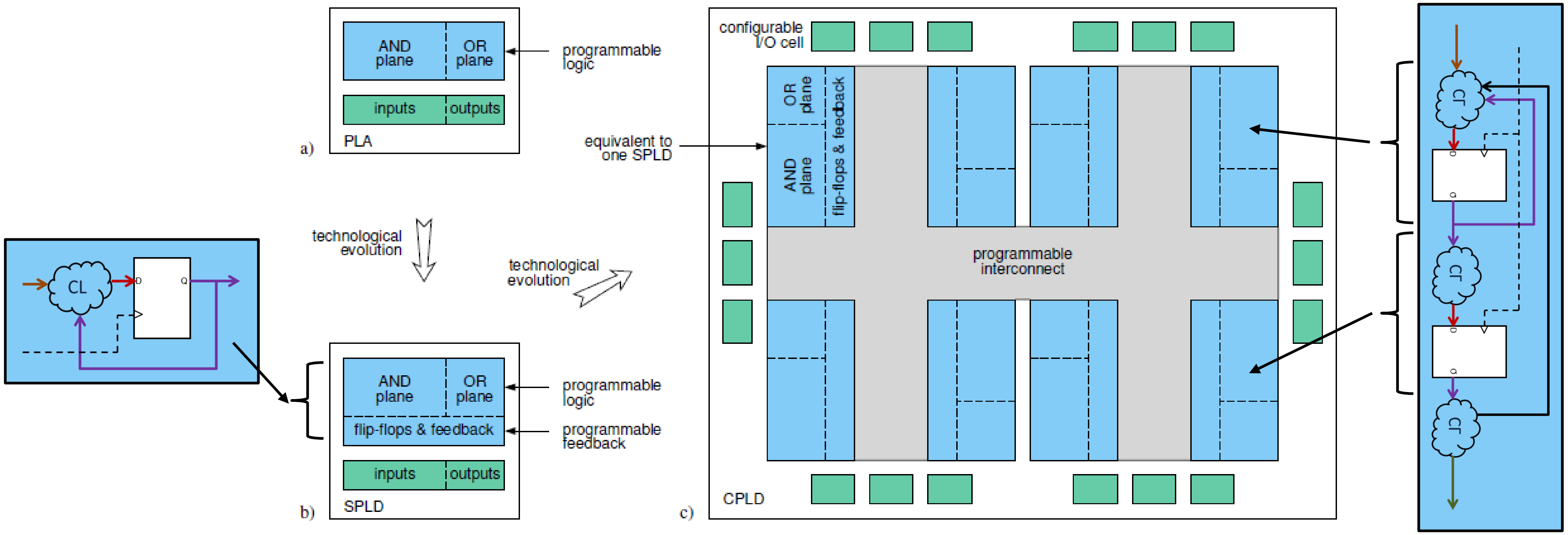
$$s = a\bar{b}\bar{c} + \bar{a}b\bar{c} + \bar{a}\bar{b}c + abc$$
$$c_{out} = ab + bc + ac$$

a / cb	0	1
00	0	0
01	0	1
11	1	1
10	0	1

Early Complex Programmable Logic

- **Complex Programmable Logic Devices** (CPLDs): **multiple stages of logic and registers** in a single device
- **Some** programmable **interconnect between large logic stages**





Issues of AND-OR Based Programmable Logic

- The **AND-OR** based **2-stage logic style** of PLAs/CPLDs **has several disadvantages**

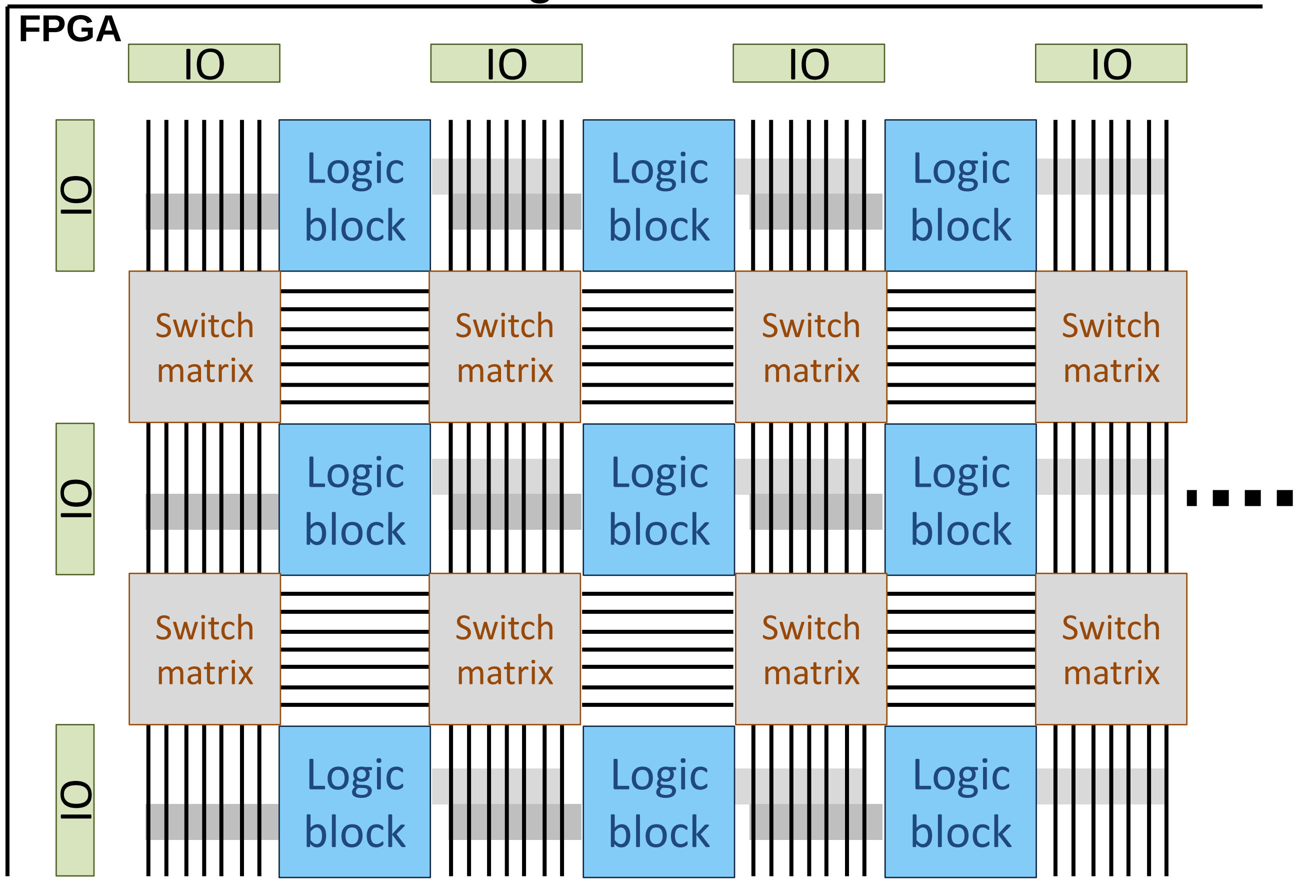
  - **From an application perspective**
    - 2-stage logic **limits the fan-in** (maximum number of inputs per output)
    - 2-stage logic also **limits resource/logic sharing** between multiple outputs
    - Modern synthesis tools, developed for ASIC design, rely on multi-stage logic

  - **From a device design perspective**
    - PLAs, especially with larger fan-in have a **high power consumption**
    - Area-**efficient implementation** of PLA circuit **is** sometimes **tricky**
      - **Not well suited for** the highly efficient **CMOS** logic style
        (**favours** more tricky logic styles such as **dynamic logic** or **pseudo-nMOS logic**)
    - Few large **interconnects** between complex logic stages **involve large overhead**

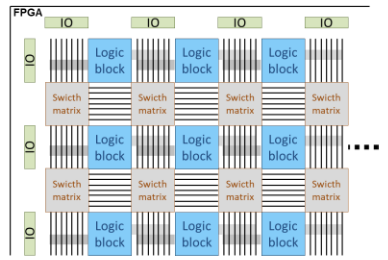
Field Programmable Gate Arrays (FPGAs)

- **FPGAs**: Similar idea to CPLDs, but **built for multi-stage logic** (with logic sharing)
  - Collection of **many small** programmable **logic/register blocks**
  - Focus on **massive, but highly modular interconnect** between logic blocks

  - **Configurable logic blocks (CLBs)**:
    - Registers
    - Programmable logic with 4-6 inputs
  - **Interconnect wires** and **switch matrices**:
    - 2D grid of short and long wires
    - Local **switch matrices** to connect
      - Wires to wires
      - Wires to logic blocks
  - Programmable IOs (pads)



Implementation of Logic Operations in FPGAs

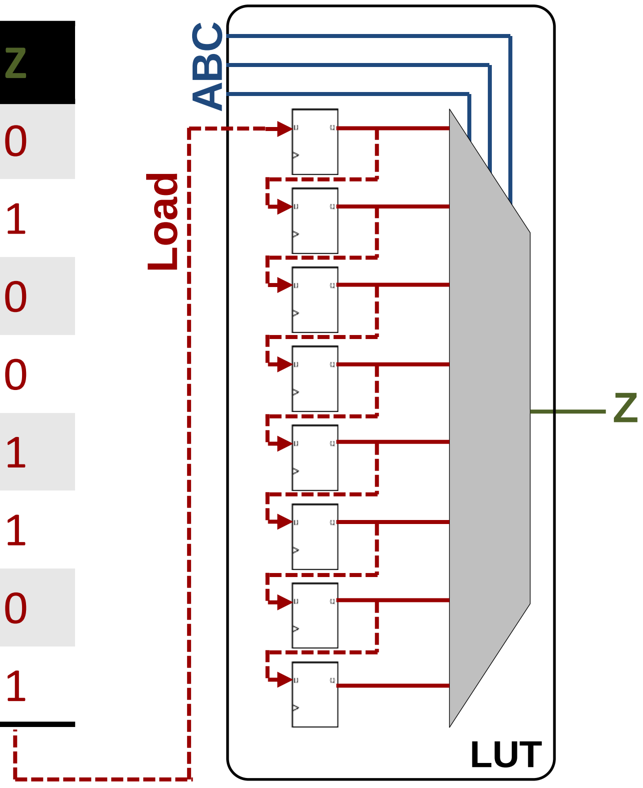


- **Logic is implemented** based **on programmable look-up tables**

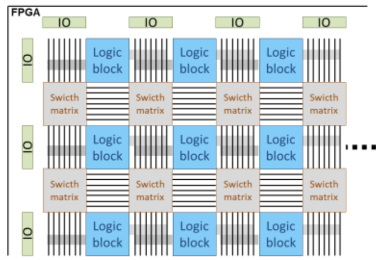
- **Basic principle:** start from truth table representation
  - **Programming:**
    - Registers connected as a shift-register
    - **Truth table** content (logic result) **shifted into the LUT configuration registers** during programming
  - **Operation (after programming):**
    - **Truth table** content stored **in configuration registers**
    - A **multiplexer chooses the output** from the registers **based on the logic input**

- **Note:** LUT is basically a memory that is pre-loaded during configuration

ABC	Z
000	0
001	1
010	0
011	0
100	1
101	1
110	0
111	1

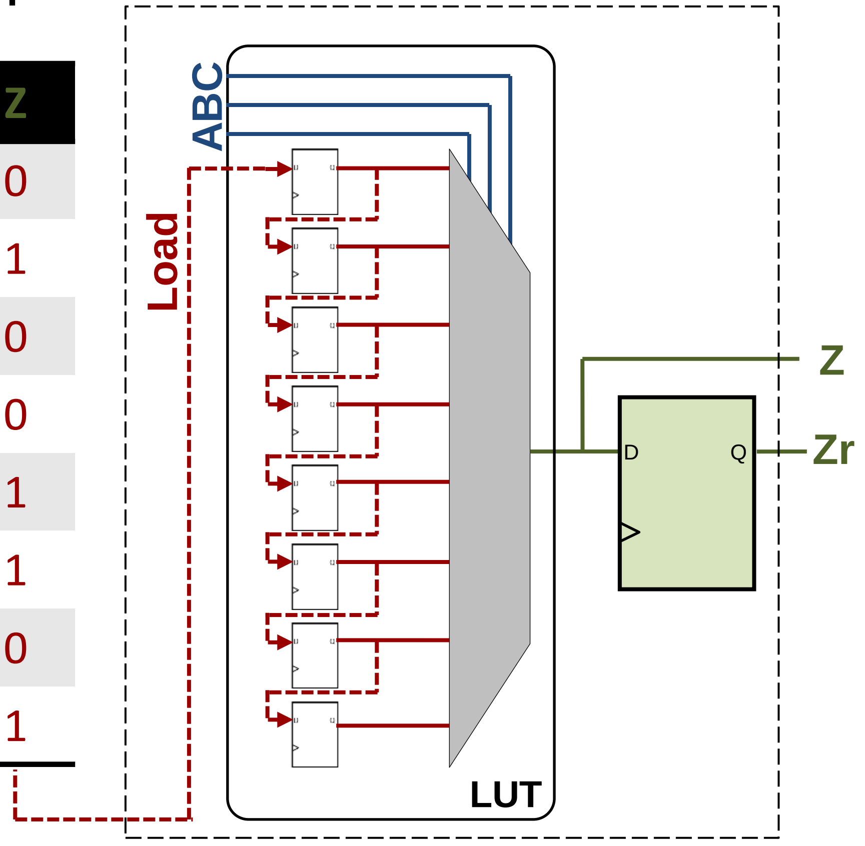


Implementation of Logic Operations in FPGAs



- **Logic is implemented** based **on programmable look-up tables**
  - **Functional registers** (with bypass) added to LUT outputs
- **Basic principle:** start from truth table representation
  - **Programming:**
    - Registers connected as a shift-register
    - **Truth table** content (logic result) **shifted into the LUT configuration registers** during programming
  - **Operation (after programming):**
    - **Truth table** content stored **in configuration registers**
    - A **multiplexer chooses the output** from the registers **based on the logic input**
- **Note:** LUT is basically a memory that is pre-loaded during configuration

ABC	Z
000	0
001	1
010	0
011	0
100	1
101	1
110	0
111	1

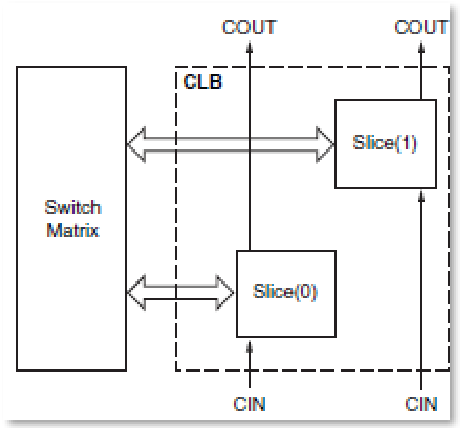


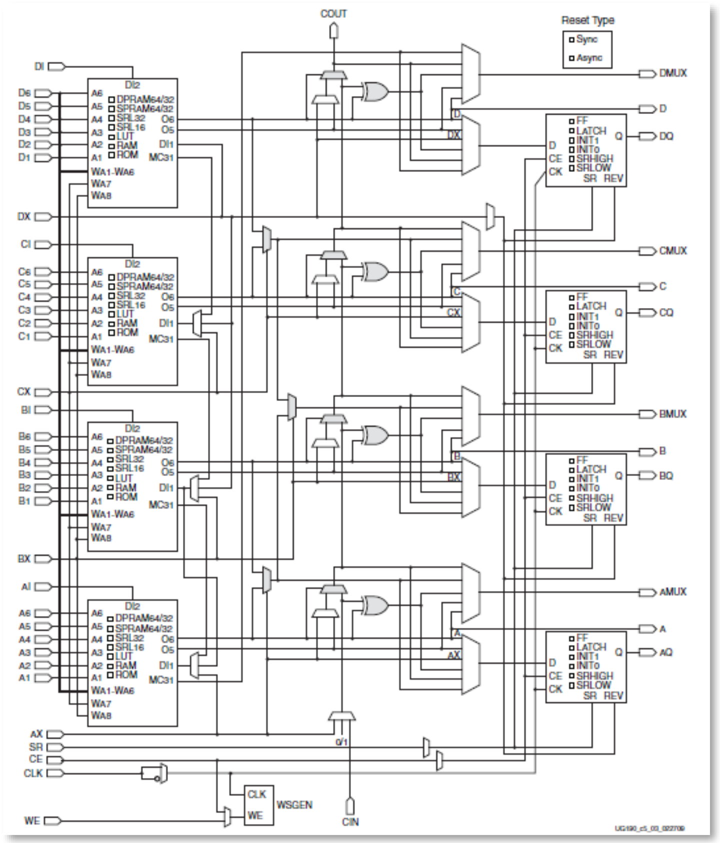
Logic Blocks in Modern FPGAs

- **Modern FPGAs all build on the LUT concept**

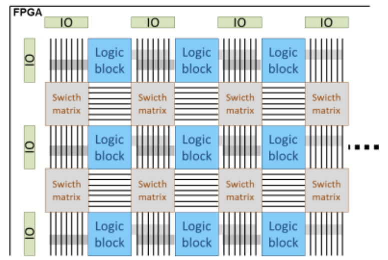
- **Trend: cluster more logic/registers into each configurable logic block** to minimize use of the slow global interconnect
  - **Additional logic** (e.g., MUXes after LUTs or dedicated connections) **supports** important and frequent **high fan-in logic** operations
  - **Implement larger LUTs** with multiple outputs
    - XILINX Virtex-7: supports either 5-input-1-output OR 6-input-2-output
  - Advanced **LUTs can be re-purposed for special functions**, e.g., memory or shift registers

Example: XILINX Virtex-5 Slice

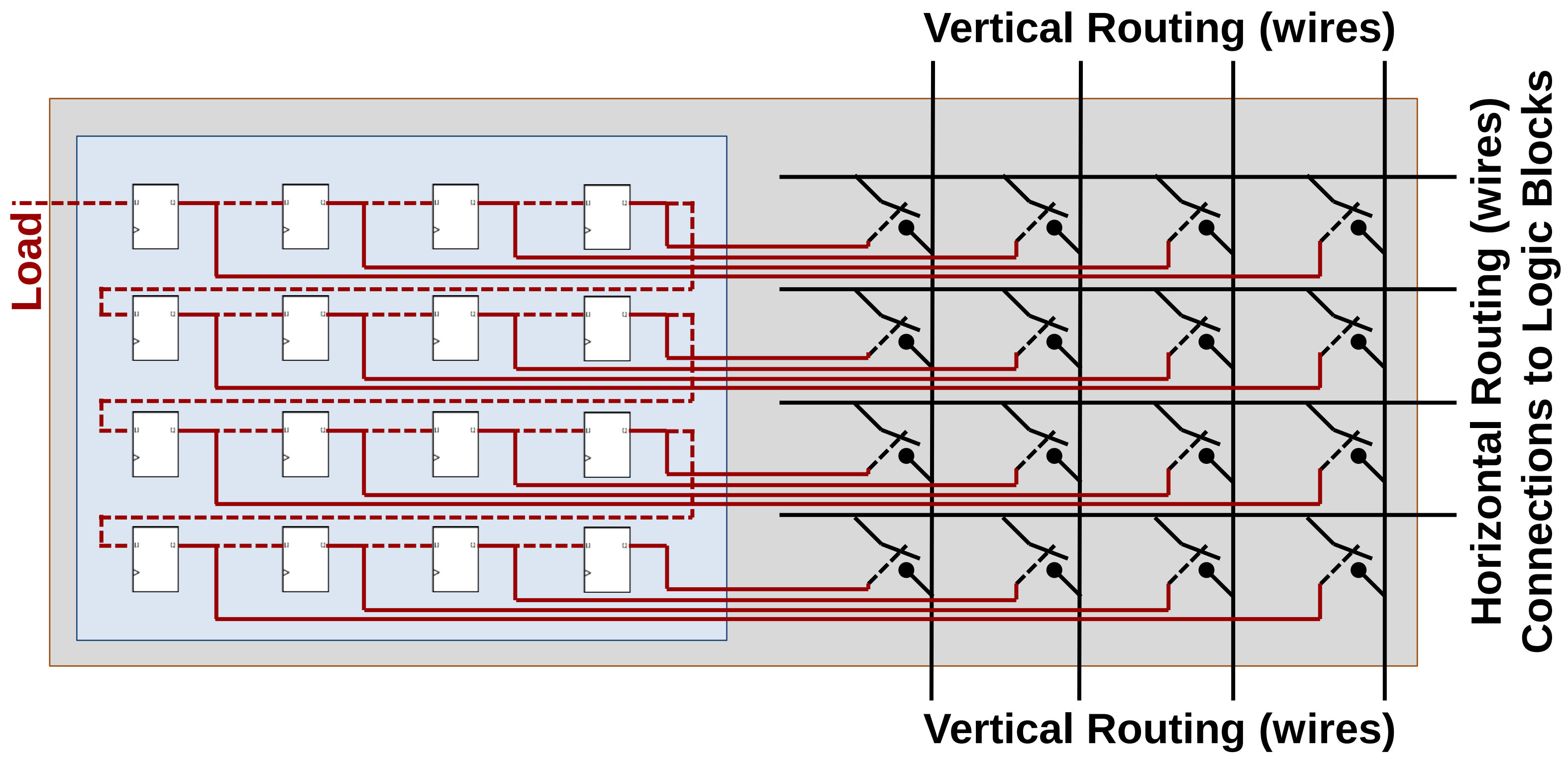




Connecting CLBs with Configurable Global Routing

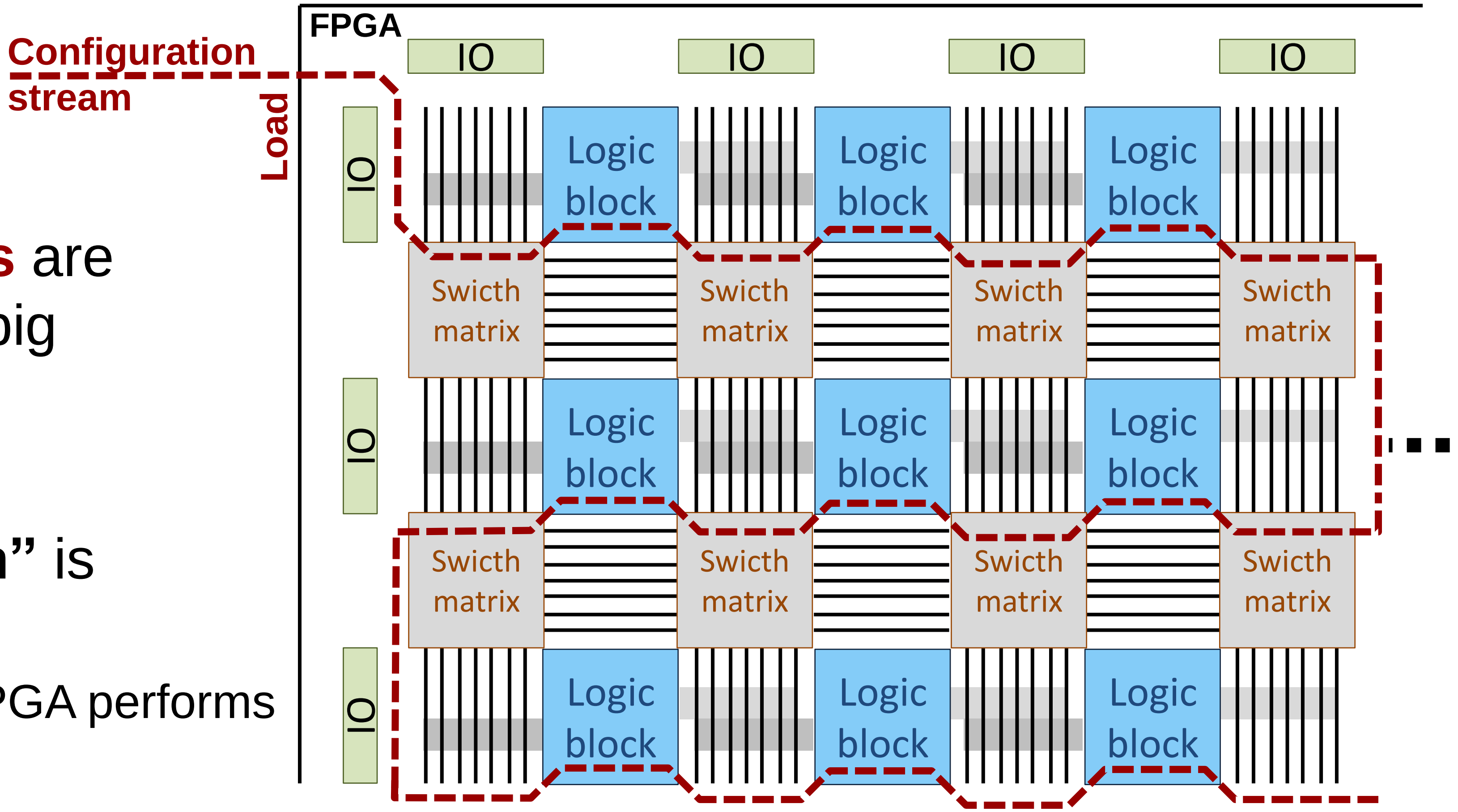


- **Routing resources on** the **FPGA devices** comprise
  - **Wires (**short (local) and long (global)) **that run** both **horizontally and vertically**
  - **Switch boxes to connect**
    - **wires to logic blocks** and
    - **horizontal** and **vertical wires**
    - **Configuration in registers**
    - **Programming: through shift register configuration**



Programming FPGAs

- **Programming loads** the **configuration registers & LUTs** of the FPGA logic and the programmable routing (switch boxes).

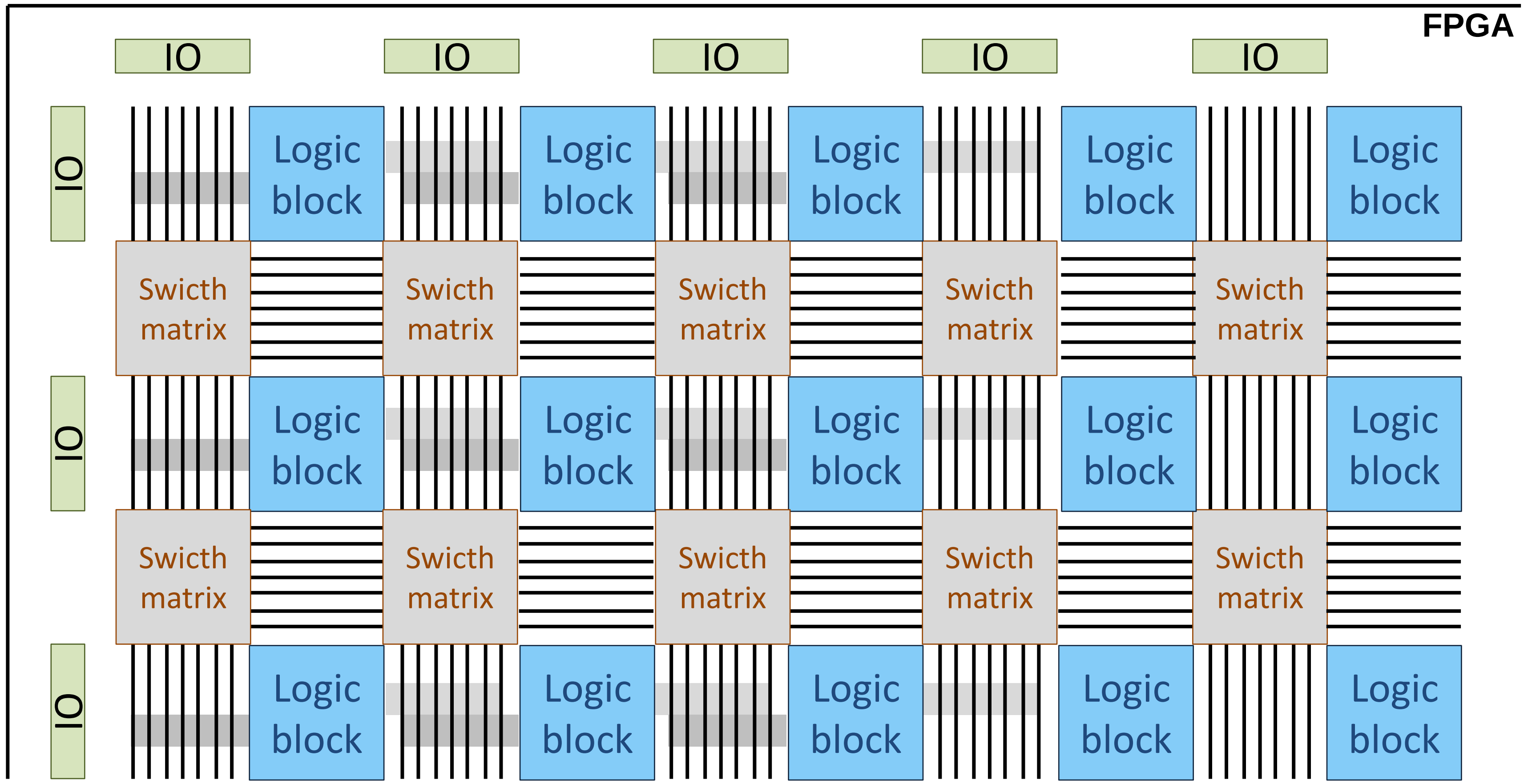


- All **configuration registers** are **daisy chained** in a single big shift register

- **Configuration "bit-stream"** is shifted in bit-by bit
  - After this configuration, the FPGA performs the configured function

Modern FPGAs also Include Complex Macros

- **Some common functions** (e.g., arithmetic, memory) **are costly** and slow **when realized with** general purpose **configurable logic** blocks

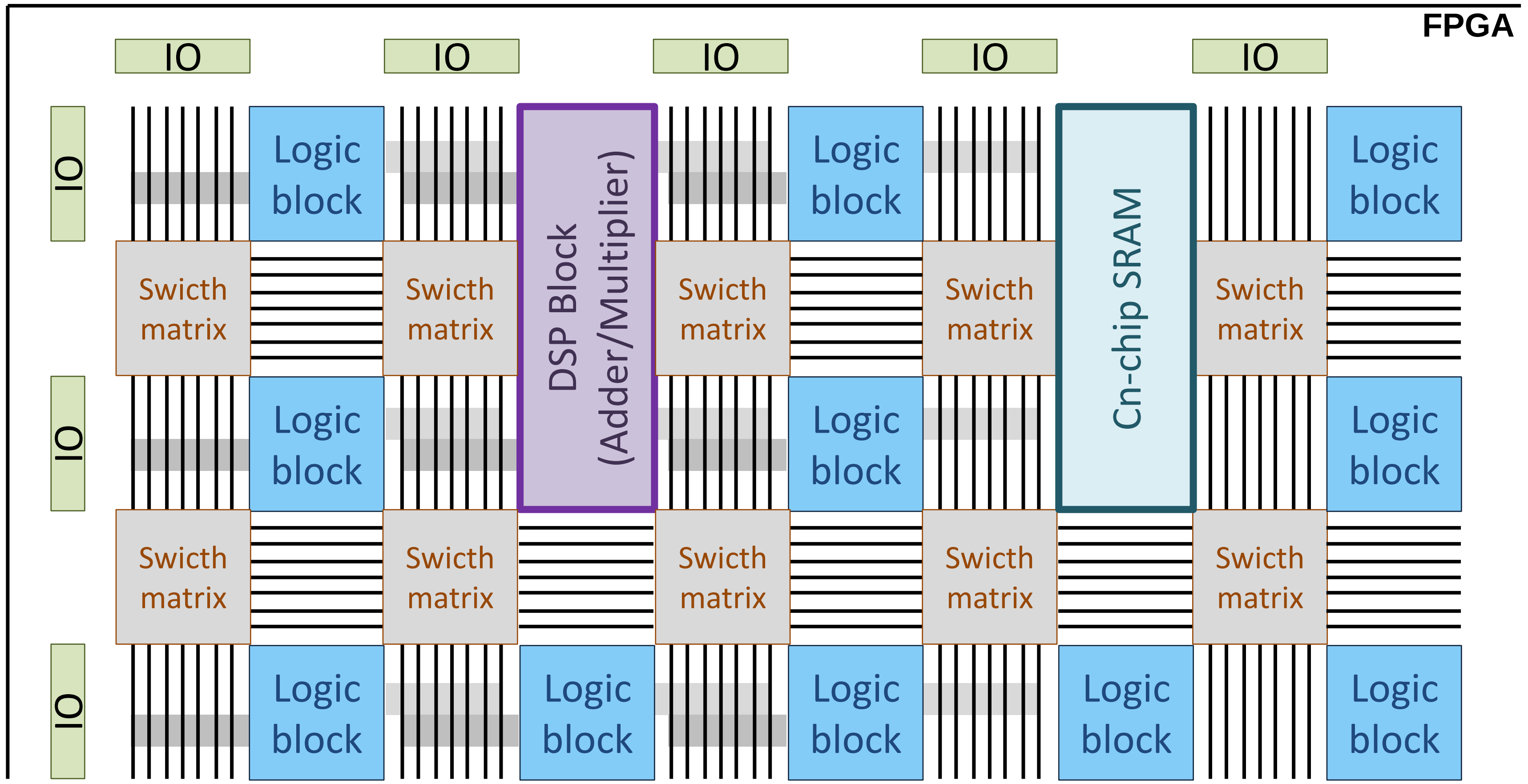


Modern FPGAs also Include Complex Macros

- **Some common functions** (e.g., arithmetic, memory) **are costly** and slow **when realized with** general purpose **configurable logic** blocks

- **Solution**: integrate **common complex functions as fixed,** optimized **macros**

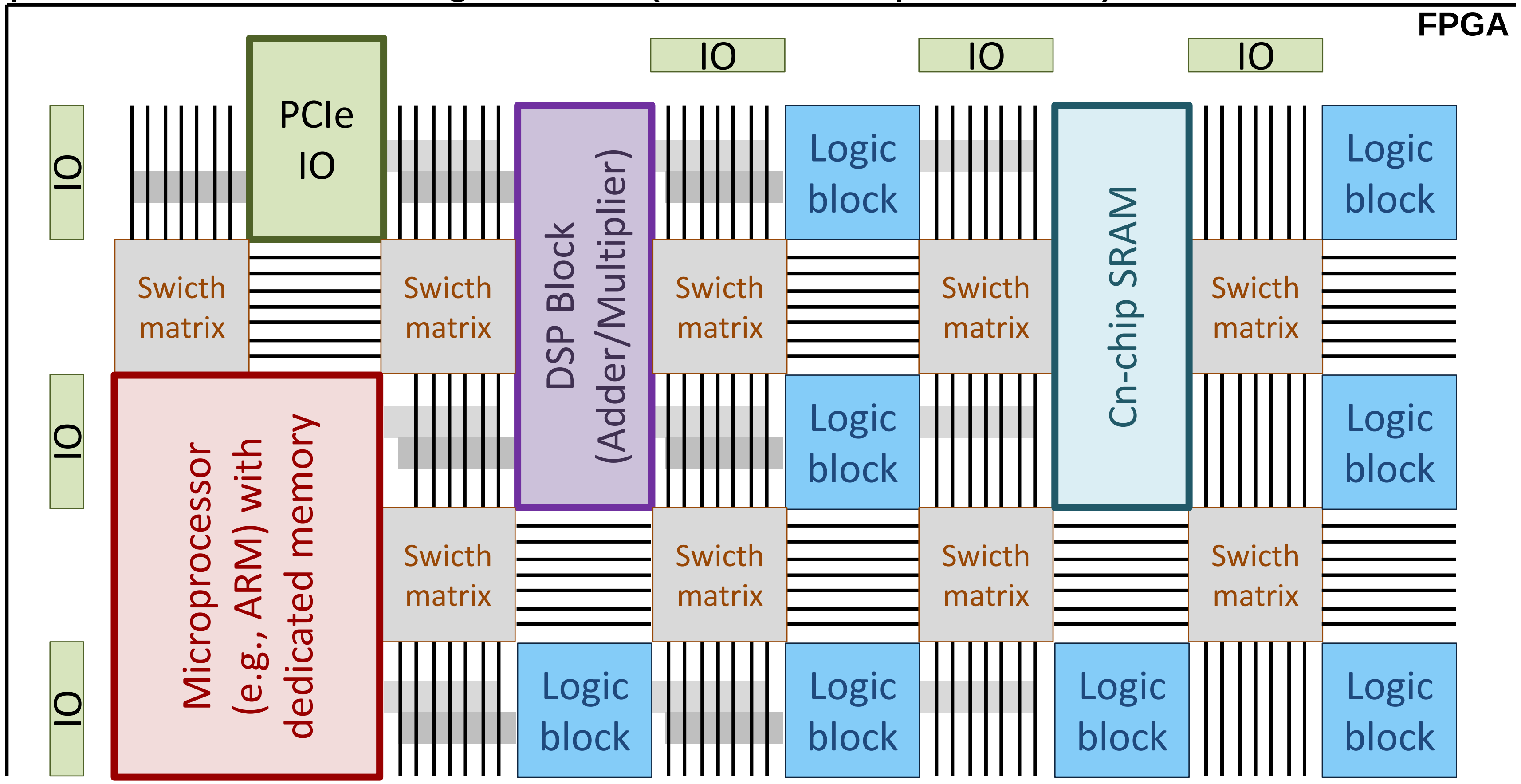
- Examples:
  - **DSP blocks**
  - **On-chip SRAM**



Modern FPGAs as SoC Platform

- Complex **systems often include microprocessors** and high-speed interfaces
  - Microprocessor sub-systems are generic and can be very efficient in dedicated logic
  - High-speed interfaces require dedicated analog circuits (can not map to CLBs)

- **FPGA SoCs include Micro-processors that have access to the programmable logic (e.g., for accelerators)**
- **High-speed IOs** are added for common standards (e.g., PCIe)


FPGA
IO
IO
IO
IO
IO
IO
PCIe IO
DSP Block (Adder/Multiplier)
Cn-chip SRAM
Microprocessor (e.g., ARM) with dedicated memory
Logic block
Swicth matrix

The FPGA Implementation Flow

- **FPGA design** typically **starts from HDLs**

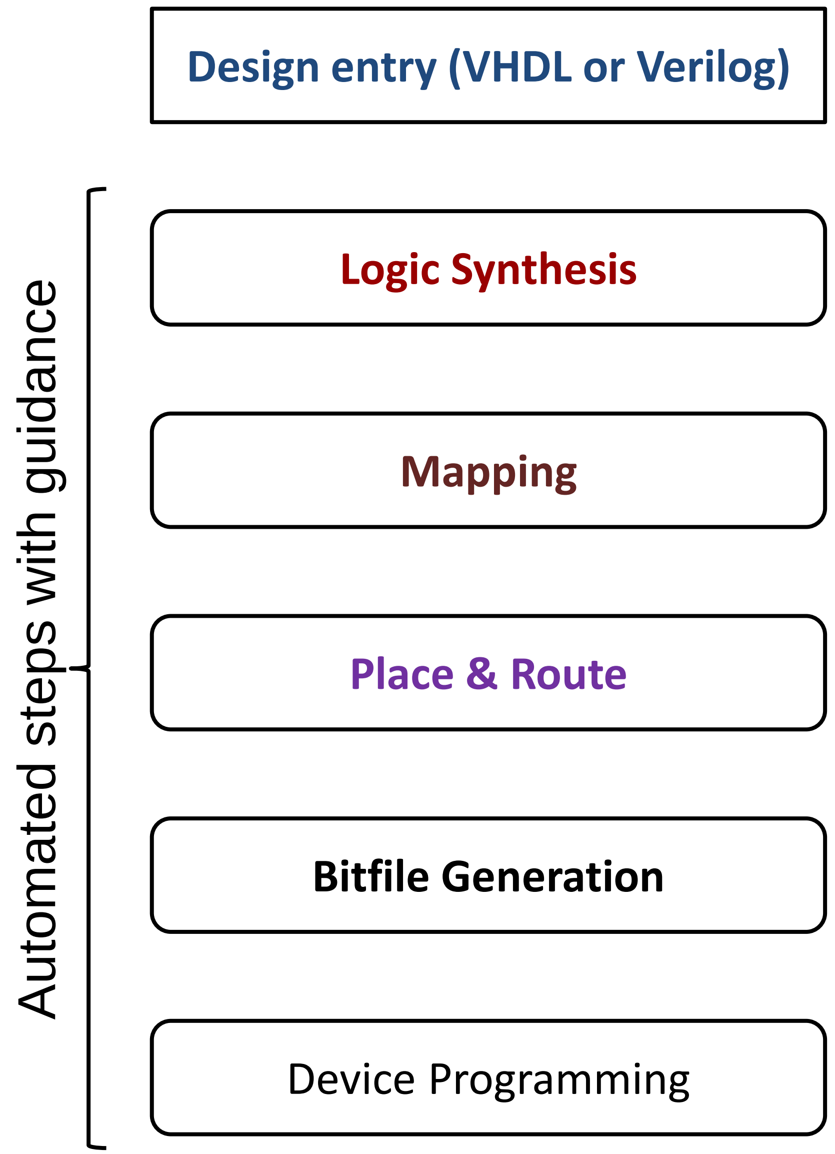
- **Synthesis produces a netlist** of logic gates

- The **netlist is partitioned** into groups of gates that are **mapped into** individual **LUTs**

- **Place**: assign LUTs to logic blocks
- **Route**: determine switch box configuration

- **Bitfile generation**: write out the configuration bits to be loaded into the FPGA

- Load bit stream into FPGA


Design entry (VHDL or Verilog)
Automated steps with guidance
Logic Synthesis
Mapping
Place & Route
Bitfile Generation
Device Programming