INTRODUCTION TO FPGA PROGRAMMING

Lesson 01: Digital Systems, FPGAs and HDL

Dr. Davide Cieri[1]

[1] Max-Planck-Institut für Physik, Munich

September 2024

COURSE LOGISTICS

- Agenda available at this link.
  - Slides will be directly published on the Agenda.
- Lab Exercises are available at this Gitlab.com repository
  - https://gitlab.com/davide.cieri89/fpga-course-labs
- Every afternoon, I will post the solution to the lab exercise in this other repository
  - https://gitlab.com/davide.cieri89/solutions-fpga-course-labs
- Lab exercises can be done on the provided laptop.
  - If you want to exercise at home, you need a laptop with Linux and install Vivado 2022.2
  - Download it here.
  - Instructions on how to install Vivado are available in the README of the lab gitlab repository.

COURSE PROGRAMME

1. Digital Systems, FPGAs and HDL
2. VHDL Fundamentals
3. Boolean, Algebra, Look-up Tables and IOs
4. Sequential Logic and Flip-Flops
5. VHDL Types, Arrays and Arithmetic Functions
6. VHDL Simulation
7. Storing Data on FPGAs
8. VHDL Packages, Libraries and Parameterisation
9. Finite State Machines
10. IP Blocks
11. Timing on FPGAs
12. Hardware Debugging
13. Advance Vivado Flow
14. External Interfaces
15. Processors on FPGA

REFERENCES

Books:

- Russel Merrick, Getting Started with FPGAs
- Free Range VHDL: link
- S. Churiwala, Designing with Xilinx FPGAs

Online Resources:

- AMD Documentation Hub
- FPGA4Fun.com
- NandLand.com
- VHDLWhiz.com
- Digilent Basys3 Reference Manual

ANALOGUE AND DIGITAL

Analogue Systems:

- Continuous in both time and voltage.
- Continuous range of current $i(t)$ or voltage $V(t)$ as a function of time.
- Full range of information available.

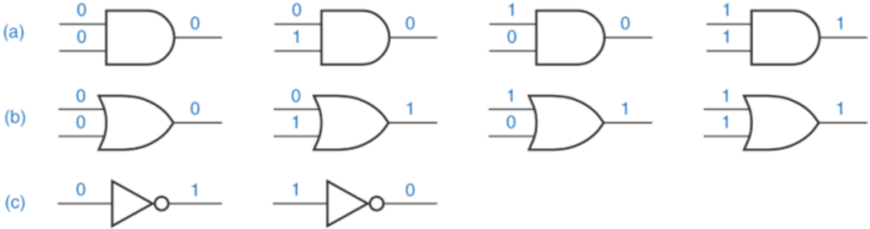
Digital Systems:

- Continuous in time, discrete in amplitude.
- Modelled as taking on, at any time, only one of two discrete values (0 or 1).
- Less information, but more robust against noise

CLASSIFICATION OF DIGITAL DEVICES

Combinational Digital Circuits

- Outputs of the circuit depends only on the current inputs (e.g. Gates).
- Three main types of gates: AND (a), OR (b) and NOT (c).



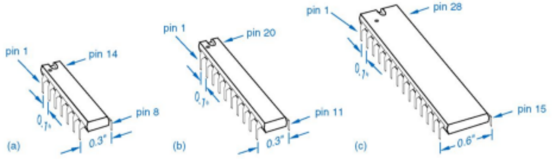
Sequential Digital Circuits

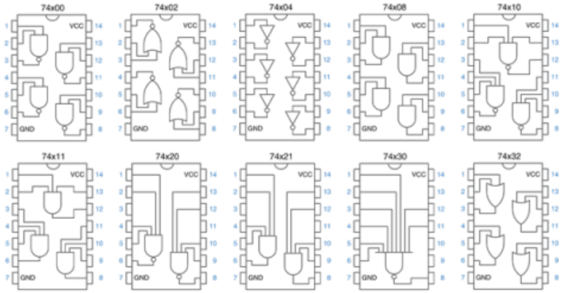
- Outputs of the circuit depends upon its past state (e.g. Flip-Flops).

INTEGRATED CIRCUITS

- *Integrated Circuit (IC)*: Collection of gates fabricated on a single silicon chip.
- Any IC is initially part of a *wafer* containing many replicas of the IC.
- ICs historically divided into: small, medium, large scale, and very large
  - SSI (up to 20 gates)
  - MSI (20-200 gates)
  - LSI (200 1M gate)
  - VLSI (over 1M gates)

74 series pin diagrams are shown on the bottom

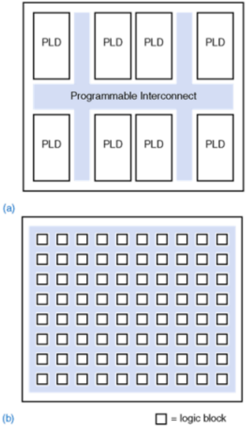




PROGRAMMABLE LOGIC DEVICES

ICs that can have their logic function programmed

- Programmable logic array: PLA
  - Historically the first programmable ICs
- Programmable logic device: PLD
- Complex PLD: CPLD (a)
  - A mere collection of PLDs
  - The inter-PLD connection is also programmable
- Field-programmable gate array: FPGA (b)
  - Much larger number of small individual blocks
  - Distributed interconnection structure that dominates the entire chip



PLD-based products can be programmed by means of Hardware Description Languages: HDL

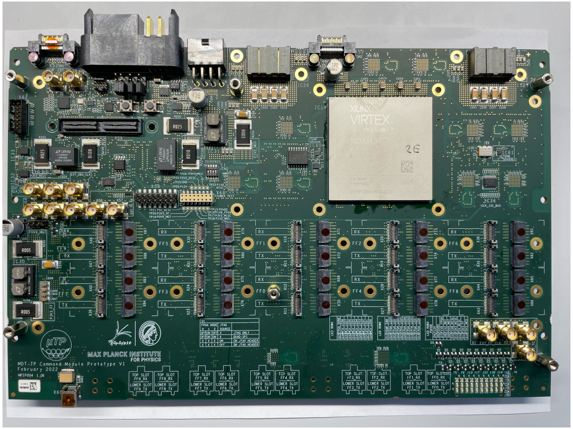
FIELD-PROGRAMMABLE GATE ARRAYS (FPGAS)

- *Field-Programmable*: Can be reprogrammed in the *field* (not returning to manufacturer).
- *Gate-Array*: 2D grid featuring a large number of interconnected gates.[1]
- Born in 1985 as evolution of CPLD
- Same concept as PLDs, but with vastly more complex functions and extra features

[1] FPGAs are nowadays more complex than an array of simple gates. Some of them cannot be reprogrammed.

COMMON FPGA APPLICATIONS

- Digital signal processing
- Telecommunications
- Aerospace and defense
- Research
- Prototyping of ASICs
- Novel: Neural-Network and Machine Learning Algorithm implementations (Hardware Accelerators)



The ATLAS MDTTP board with a Xilinx Virtex Ultrascale+ FPGA

FPGA VS. MICROCONTROLLER VS. ASIC

Feature	FPGA	Microcontroller	ASIC
Definition	Field-Programmable Gate Array	Integrated CPU with peripherals	Application-Specific IC
Flexibility	High	Low	None
Performance	High, parallel processing	Moderate, sequential processing	High, optimized for tasks
Development Time	Moderate	Short	Long
Cost (low quantities)	Moderate	Cheap	Expensive
Cost (high quantities)	Moderate	Cheap	Cheap
Use Cases	Prototyping, custom computing	Embedded systems, control applications	High-volume, performance-critical
Power Consumption	Moderate	Low	Low

MAJOR FPGA VENDORS

- AMD (formerly Xilinx), our option for the course. Largest vendor worldwide.
- Altera (Intel)
- Microchip (formerly Microsemi / Actel)
- Lattice Semiconductor

AMD FPGA FAMILIES

- Three available series on the market (7, UltraScale, UltraScale+)
- Available families in (almost) each series:
  - **Virtex**: Largest devices, highest speeds
  - **Kintex**: Best Price/Performance balance
  - **Artix**: Low-end, Cost and Transceiver Optimized
  - **Zynq/Versal**: FPGA fabric with built-in ARM/AI processor systems
- More infos on AMD website.

DIGILENT BASYS3 BOARD

AMD Artix-7 FPGA Trainer Board

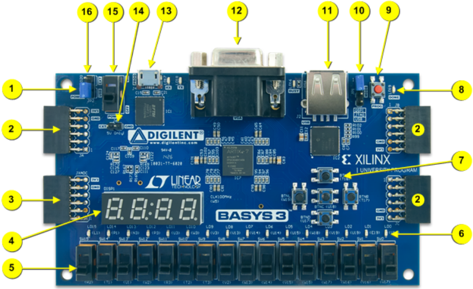


Figure 1. Basys3 board features

https://digilent.com/reference/programmable-logic/basys-3/start

MAIN COMPONENTS OF AN FPGA

- **Configurable Logic Blocks (CLBs)**
  - Basic logic units
  - Configurable to perform a variety of logic functions
- **Interconnects**
  - Network of wiring connecting logic blocks
- **Input/Output Blocks (IOBs)**
  - Interfaces for connecting the FPGA to external devices
- **Block RAM**
  - Dedicated memory blocks
  - Used for storing data and instructions
- **Digital Signal Processing (DSP) Blocks**
  - Specialized blocks for performing complex mathematical calculations
- **Clock Management Tiles (CMTs)**
  - Include phase-locked loops (PLLs) and clock distribution networks
- **Configuration Memory**
  - Stores the configuration data that defines the behavior of the FPGA

CONFIGURABLE LOGIC BLOCKS (CLBS)

- Consist of Look-Up Tables (LUTs), Flip-Flops, and Multiplexers
- LUTs implement logic functions
- Flip-Flops store state information
- Multiplexers are used for routing within the CLB

HARDWARE DESCRIPTION LANGUAGES (HDL)

HDLs are not programming languages

- Used to describe the behavior of digital circuits, to be implemented on FPGAs or ASICs.
- The modelling of digital circuits in HDL is called register-transfer level (RTL).
- Also used to verify digital designs with simulations
- Only a subset of the language can actually be "synthesised"

Major available languages:

- VHDL (our choice for this course)
- Verilog

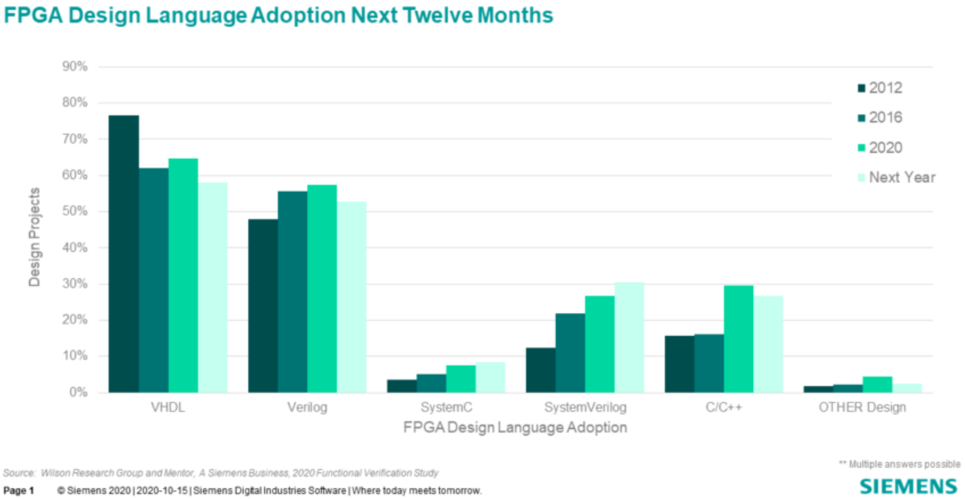
Other options:

- High-Level-Synthesis (HLS): C-like language, which is translated into RTL by FPGA vendor tools
- SystemVerilog (Verilog extension with dedicated functionalities for verification).

HDLS VS PROGRAMMING LANGUAGES

	General purpose programming language	Hardware Description Language
Sequential Execution	✓	✓
Concurrent Execution	✗	✓
Model Function	✓	✓
Capture timing	✗	✓
Structure of hardware	✗	✓

HDL MARKET SHARE



FPGA Design Language Adoption Next Twelve Months

** Multiple answers possible

Source: Wilson Research Group and Mentor, A Siemens Business, 2020 Functional Verification Study

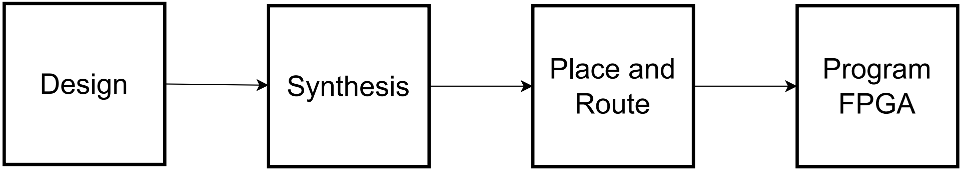
Page 1 © Siemens 2020 | 2020-10-15 | Siemens Digital Industries Software | Where today meets tomorrow.

SIEMENS

WHAT IS VHDL?

- VHSIC Hardware Description Language
  - VHSIC: Very High Speed Integrated Circuit
  - Designed originally from the US Government, based on ADA
- #1 Language for FPGA Design and Verification (especially in Europe)
- 1987: First original standard IEEE-1076
- 1993: First major revision (still widely supported version)
- 2008: Second major revision (our option for the course)
- 2019: Third major revision (still not widely supported by FPGA vendors)

FPGA BUILD WORKFLOW

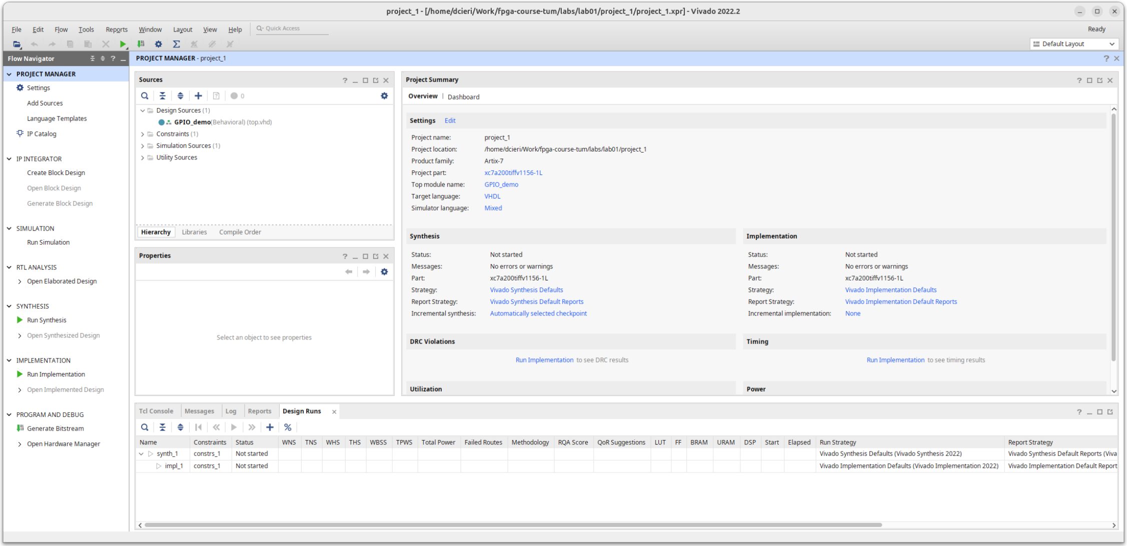


1. **Design.** Write HDL code describing the functionalities to implement on FPGA (FPGA gateware).
2. **Synthesis.** Translates HDL code to low-level components.
3. **Place and Route (Implementation).** Map the synthesised design to the physical layout of the FPGA (place), and wire the connection between the components (Route).
4. **Programming.** Load the output (bitstream) of P&R step onto the physical FPGA.

VIVADO

- Vivado is a design suite developed by AMD/Xilinx for FPGA and SoC design.
- Key Features:
  - **Integrated Development Environment (IDE)**
    - □ User-friendly interface for design, simulation, and synthesis.
  - **IP Integrator**
    - □ Facilitates easy integration of various IP (Intellectual Property) cores.
  - **Comprehensive Debugging Tools**
    - □ Includes tools like Integrated Logic Analyzer (ILA).
  - **Support for Multiple Languages**
    - □ Supports VHDL, Verilog, and SystemVerilog.

VIVADO



project_1 - [/home/dcieri/Work/fpga-course-tum/labs/lab01/project_1/project_1.xpr] - Vivado 2022.2

Settings Edit

Project name: project_1
Project location: /home/dcieri/Work/fpga-course-tum/labs/lab01/project_1
Product family: Artix-7
Project part: xc7a200tffv1156-1L
Top module name: GPIO_demo
Target language: VHDL
Simulator language: Mixed

Synthesis

Status: Not started
Messages: No errors or warnings
Part: xc7a200tffv1156-1L
Strategy: Vivado Synthesis Defaults
Report Strategy: Vivado Synthesis Default Reports
Incremental synthesis: Automatically selected checkpoint

Implementation

Status: Not started
Messages: No errors or warnings
Part: xc7a200tffv1156-1L
Strategy: Vivado Implementation Defaults
Report Strategy: Vivado Implementation Default Reports
Incremental implementation: None

DRC Violations

Run Implementation to see DRC results

Timing

Run Implementation to see timing results

Utilization

Power

Name	Constraints	Status	WNS	TNS	WHS	THS	WBSS	TPWS	Total Power	Failed Routes	Methodology	RQA Score	QoR Suggestions	LUT	FF	BRAM	URAM	DSP	Start	Elapsed	Run Strategy	Report Strategy
synth_1	constrs_1	Not started																			Vivado Synthesis Defaults (Vivado Synthesis 2022)	Vivado Synthesis Default Reports (Viva
impl_1	constrs_1	Not started																			Vivado Implementation Defaults (Vivado Implementation 2022)	Vivado Implementation Default Report

HDL SIMULATION

- **Definition**: HDL Simulation is a process of verifying the functionality of digital circuits described in HDLs like VHDL or Verilog.
- **Importance**:
  - Ensures the correctness of the hardware design before implementation.
  - Allows for testing under different scenarios and inputs.
  - Helps in debugging logic and timing issues.
- **Types of Simulation**:
  - **Behavioral and Timing Simulation**: Tests the functionality of the design including timing information to validate the design against clock cycles and delays.
  - **Post-Synthesis Simulation**: Validates the design after synthesis to ensure it matches the intended logic.

SIMULATOR SOFTWARE

Many digital simulator tools available on the market

- QuestSim by Siemens
- Riviera-Pro by Aldec
- XCelium by Cadence
- XSim, integrated in Vivado (limited support to VHDL-2008)
- GHDL (open-source, VHDL-only)

In this course, we will use mainly Xsim and GHDL.

LAB 01: FIRST LOOK AT VIVADO

LAB 02: SIMULATING AN HDL DESIGN

The figures in these slides are taken from:

- Digital Design: Principles and Practices, Fourth Edition, John F. Wakerly, ISBN 0-13- 186389-4.

- siemens.com - digilent.com