VHDL

5051158-xxxx

Lecture 1

Introduction to VHDL, FPGA

V 1.1 17.9.2020
Jarno Tuominen

TURKU AMK
TURKU UNIVERSITY OF
APPLIED SCIENCES

Contents of this lecture

- Intro to
  - This course - practicalities
  - VHDL background
  - Programmable logic devices
    - PAL, CPLD, FPGA

Course organization/practices

- On Monday (PTIVIS18) or Wednesday (PTIVI17) mornings, starting at 08.30
  - A short lecture (and not even always), followed by a lab session (9- ~12)
- 12 lectures, 11 lab sessions + an exam
- See Teams for lectures schedule, materials etc
- Grading is based on
  - Completed exercises
  - (Optional) Project work
  - Exam

What is...

- … micro chip?
- … an IC?
- An analog IC? Digital IC? Mixed signal IC?

- Where are they used?
- How do you design them? <- Focus of this course!
- How they are manufactured?

VHDL – What and Why?

- **V**ery High Speed Integrated Circuit **H**ardware **D**escription **L**anguage
- Starting 1983, developed by U.S. DoD (Department of Defense) in order to *document the behavioral* of ASICs (**A**pplication **S**pecific **I**ntegrated **C**ircuits)
  - Why not simulate ASICs based on that documentation? -> VHDL logic simulators were developed
  - Next, why not “create” the logic based on this document? -> Logic synthesis
- Became an IEEE standard (IEEE 1076-1987) – known as VHDL’87
- Later, 1076-{1993, 2000, 2002, 2008} introduced, bringing some new features
  - VHDL’93 is the most widely used (and has the best tool support)

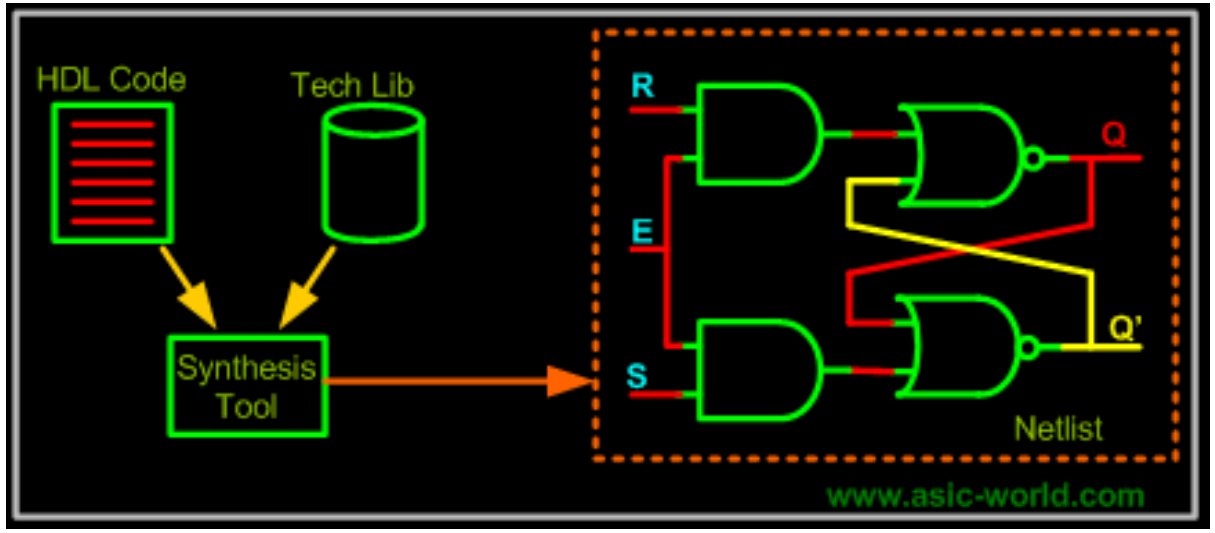
https://en.wikipedia.org/wiki/VHDL

Are there other HDLs?

- Yes (unfortunately)
- The world has been divided to two camps
  - In Europe, the dominating language for programmable logic and ASIC design is VHDL
  - In U.S. **Verilog** is used
  - And of course these are wildly mixed…
- VHDL and Verilog are for RTL synthesis
  - https://en.wikipedia.org/wiki/Register-transfer_level
- Also, higher level languages (for behavioral synthesis) exist, like System C
  - https://en.wikipedia.org/wiki/High-level_synthesis

What is logic synthesis?

- Logic synthesis is the process of converting a high-level description of design into an optimized gate-level representation.
- Logic synthesis for ASIC designs uses a standard cell library which have simple cells, such as basic logic gates like and, or, and nor, or macro cells, such as adder, muxes, memory, and flip-flops.
  - Standard cells put together are called technology library. Normally the technology library is known by the transistor size (0.18u, 90nm,65nm … 12nm end even below)



What is logic synthesis? (2)

- The synthesis tool will first translate (or **elaborate**) the “textual” design to a netlist of technology primitives
  - ANDs, ORs, Multiplexers… and in case of FPGAs, Look-Up-Tables (LUTs)
- Then, the logic **optimization** takes place
  - Synthesis/optimization is driven by **constraints** : time, area (usage of resources) or power
- Finally, we have a **netlist** – the list of these elementary building blocks, which are **implemented** on a silicon
  - Implementation = **place and route**
- We will talk about the full design flow in the coming lectures

What are FPGAs?

- ASICs are integrated circuits (digital, analogue or mixed-signal), which functionality is fully defined by the designer, to fit the application needs, FPGA can be thought as a “canvas”, where the designer can put his/her own design, based on the application needs
- ASICs are manufactured to their “final form”, whereas FPGAs are “field programmable” after the manufacturing -> **F**ield **P**rogrammable **G**ate **A**rray

https://en.wikipedia.org/wiki/Field-programmable_gate_array

https://en.wikipedia.org/wiki/Application-specific_integrated_circuit

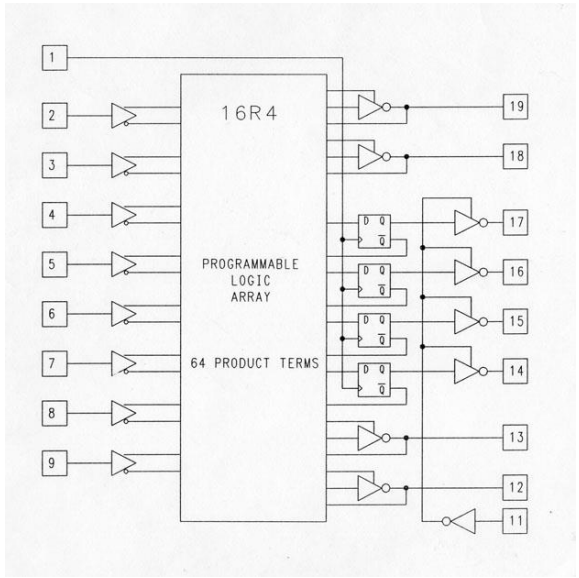
ASICs vs FPGAs

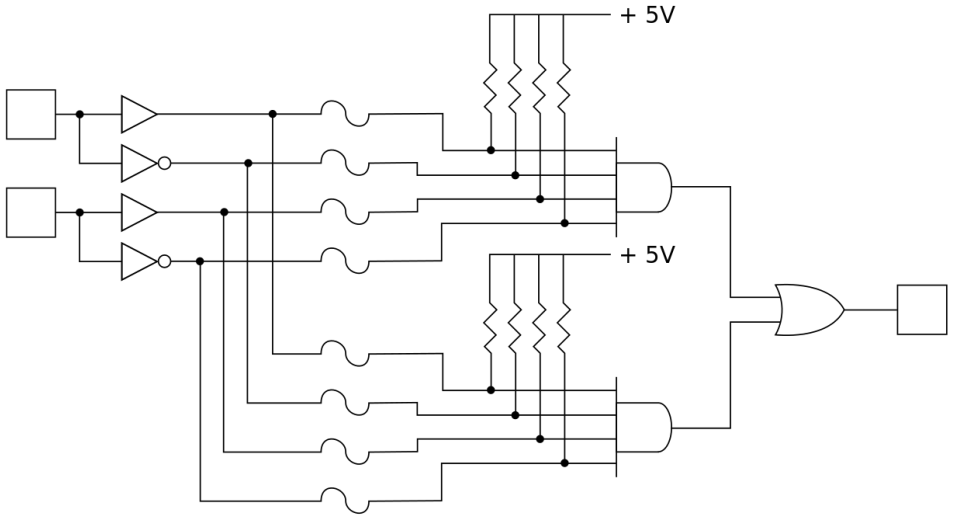
- FPGA
  - A ready made “canvas” with logic gates, registers, memory elements etc and the it is “just missing the interconnections”. Programming an FPGA means programming the interconnections (a bit simplified view…)
  - Quite expensive, especially the big and fast ones
  - There is always extra, unused logic, so utilization is not 100%
  - Higher power consumption than ASICs
  - Due to programmability, very flexible: you can download hardware updates
-
- ASIC
  - Design starts with an empty silicon, where everything needs to be designed
  - Typically, standard cell libraries are used
  - No need to put anything extra, so it is fully optimized ans can be designed for extreme performance, low power or low price (or even all of them at the same time)
  - Design work is _really_ expensive to design (Tools, NRE costs), but cheap in high volumes

FPGA history and trends (1)

- First programmable logic parts were called PALs (Programmable Array Logic), early 80's
  - https://en.wikipedia.org/wiki/Programmable_Array_Logic
- Used to replace "glue logic", i.e. a small set of standard logic gates (recall: 74HC00, CD4011B etc), just to save PCB (Printed Circuit Board) space
- Programmed with language called PALASM
- For example PAL16R4

```
PAL16R4 PAL                PAL DESIGN SPECIFICATION
CNT4SC
4 bit counter with synchronous clear
Michael Holley and Dave Pellerin
Clk  Clear  NC  NC  NC  NC  NC  NC  NC  GND
 OE  NC     NC /Q3 /Q2 /Q1 /Q0  NC  NC  VCC

  Q3 :=  Clear
       + /Q3 * /Q2 * /Q1 * /Q0
       +  Q3 *  Q0
       +  Q3 *  Q1
       +  Q3 *  Q2

  Q2 :=  Clear
       + /Q2 * /Q1 * /Q0
       +  Q2 *  Q0
       +  Q2 *  Q1

  Q1 :=  Clear
       + /Q1 * /Q0
       +  Q1 *  Q0

  Q0 :=  Clear
       + /Q0

FUNCTION TABLE
OE Clear Clk    /Q0 /Q1 /Q2 /Q3
------------------------------
 L   H    C     L   L   L   L
 L   L    C     H   L   L   L
 L   L    C     L   H   L   L
 L   L    C     H   H   L   L
 L   L    C     L   L   H   L
 L   H    C     L   L   L   L
------------------------------
```





Simplified programmable logic device

FPGA history and trends (2)

- Next, as devices grew in size, they were called CPLD's (Complex Programmable Logic Devices)
  - https://en.wikipedia.org/wiki/Complex_programmable_logic_device
  - Complexity somewhere between PALs ad FPGAs
- The basic architecture:
  - Macrocells, containing a programmable input matrix (the input product terms), followed by a flip-flop (aka register) and an output multiplexer
  - Routing channels (interconnections) between the macro cells

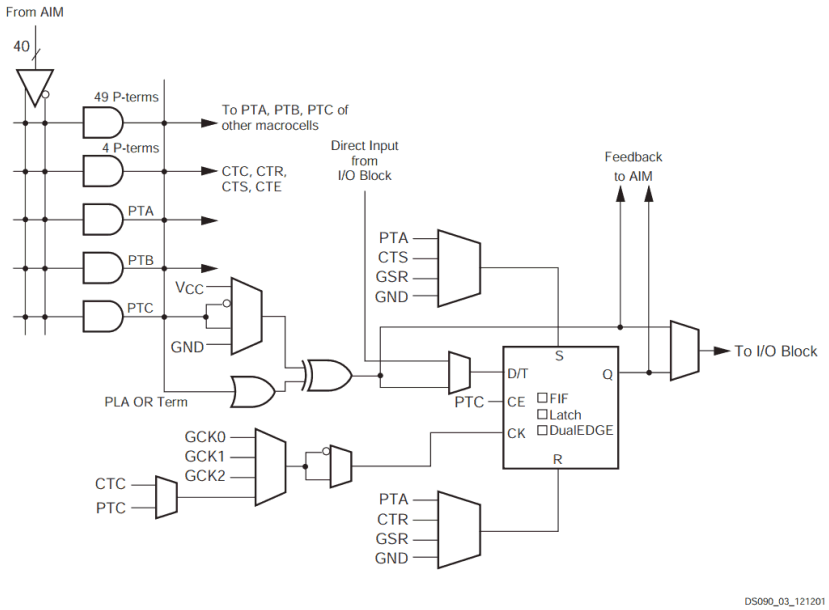


Figure 3: **CoolRunner-II CPLD Macrocell**

FPGA history and trends (3)

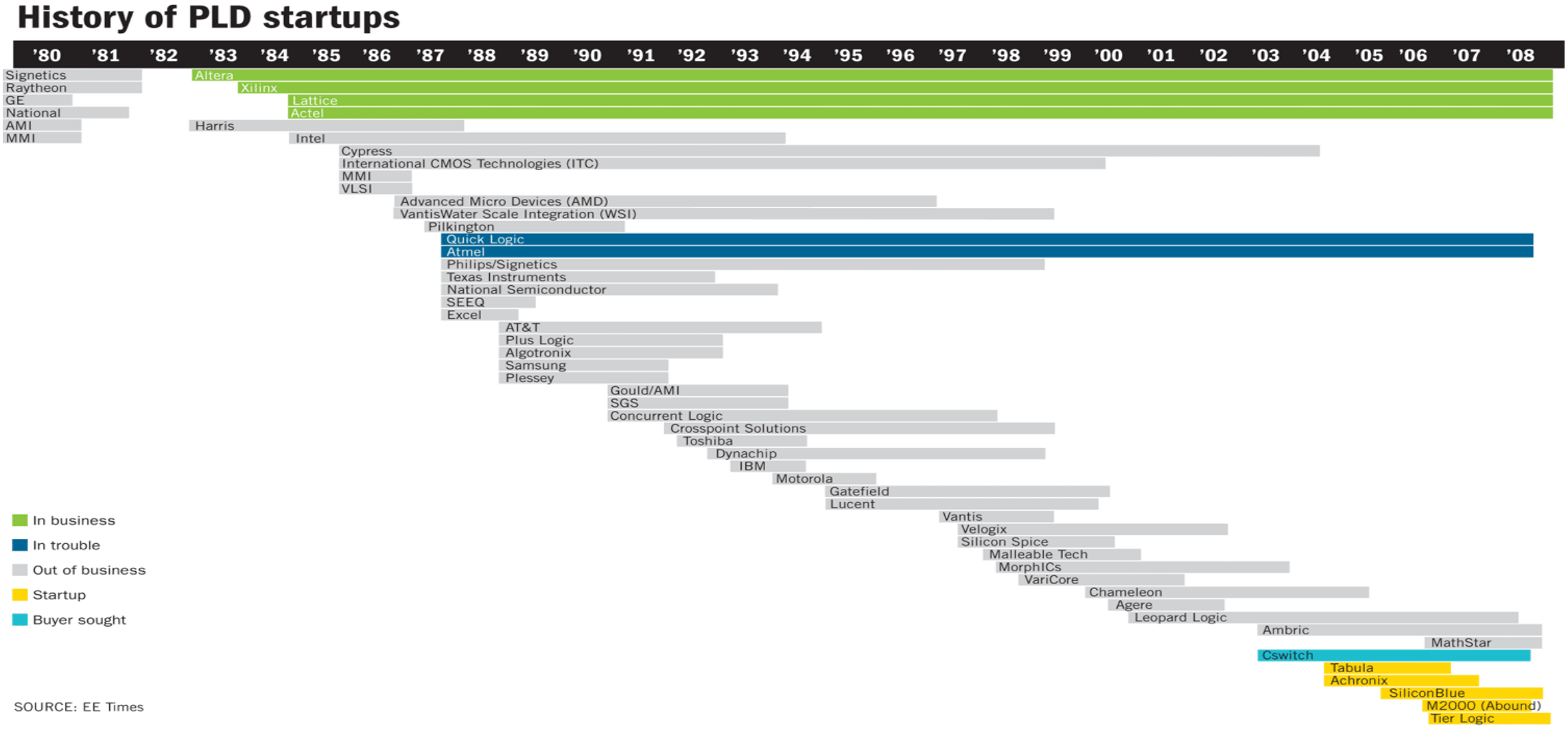
- Drawing the line between large CPLD and FPGAs in not easy – maybe even impossible
- FPGAs are in essence really BIG CPLDs, which a bit more fine-grained architecture compared to CPLDs (i.e. the macro cells are smaller/simpler, but there's a LOT of them)
- Typically includes specialized blocks in addition to CLBs (Complex Logic Blocks) like
  - Block RAMs, FIFOs (all SRAM, DRAM not used at all)
  - DSP "slices"
  - Even complete processor subsystems
- The size of FPGAs is often measured as number of (equivalent) logic gates (or logic cells), gate being a simple NAND-gate (4 transistors)
  - Easily counted in millions
- The basic architecture is still the same:
  - A set of input product terms (implemented with LUTs)
  - One or more flip-flops
  - An output multiplexer
  - Interconnections between these "macrocells" or logic slices...or CLBs, naming varies by manufacturer
- Have a look at Xilinx 7-series FPGAs
  - -> Slideset: 7-series architecture overview

Xilinx Virtex-7 series maximum resources:

- 2 million logic cells
- 6.8 billion transistors
- 3600 DSP slices
  - 5.3 TMACS combined peak performance
- 96 serial transceivers at 28.05 Gb/s each
  - Combined 2.78 Tb/s

(And these figures are probably outdated, but you get the idea)

The “must-have” 10 years old picture

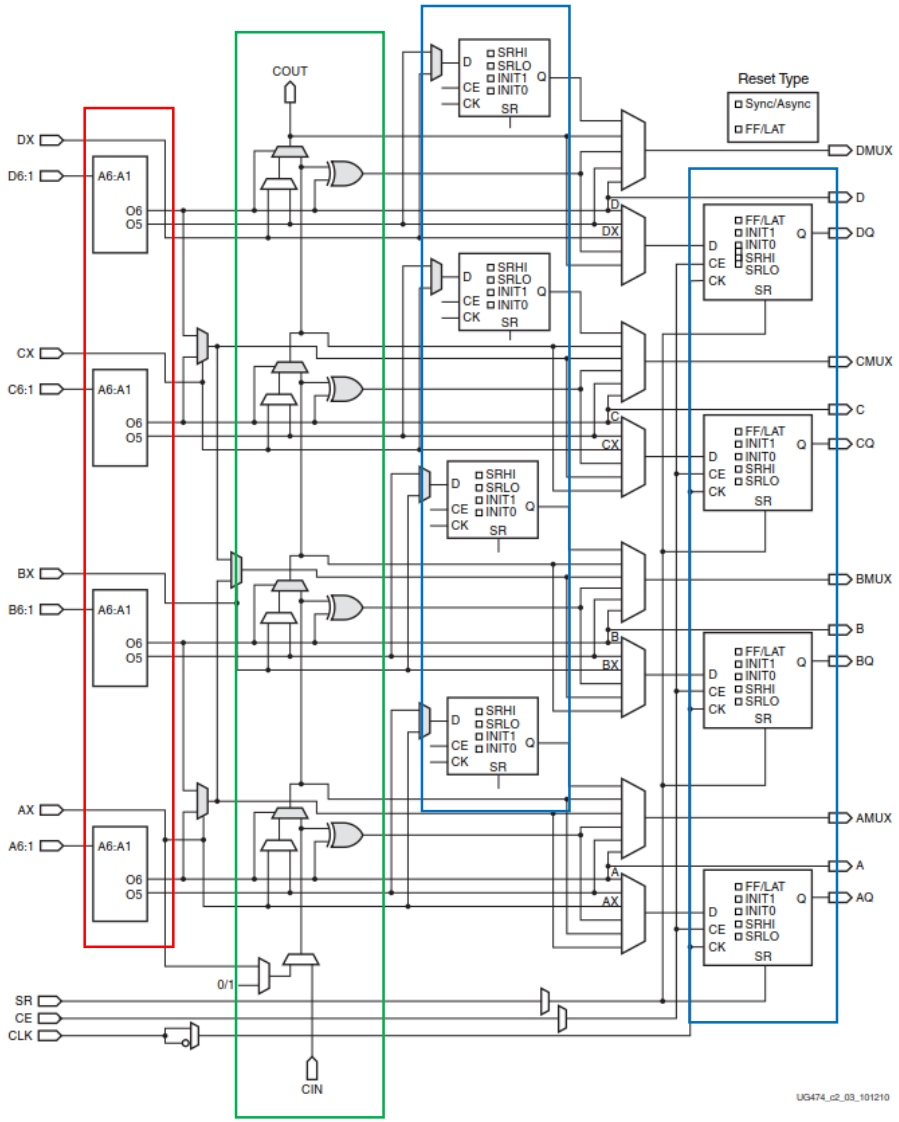


What is Xilinx Zynq-7000?

- A family of SoC-FPGAs, integrating one or more ARM A9 processor cores and Artix-7 (or Kintek-7) programmable FPGA logic fabric
  - https://www.xilinx.com/products/silicon-devices/soc/zynq-7000.html
- Popular throughout universities for educational purposes, because of:
  - Lots of accessories and expansion modules available from 3rd parties due to compatible “industry standard” interfaces, like PMOD and Arduino “shield” interface
  - Several evaluation boards alternatives available, which are reasonably priced
  - Runs Linux, but also good support for bare-metal programming.
  - An open-source project for embedded Python/FPGA-development exists (http://www.pynq.io/)
  - Free design tools (with some limitations), university program exists (https://www.xilinx.com/support/university.html)
  - Good vendor and community support, lots of demo material, tutorials

Xilinx 7-series CLB structure

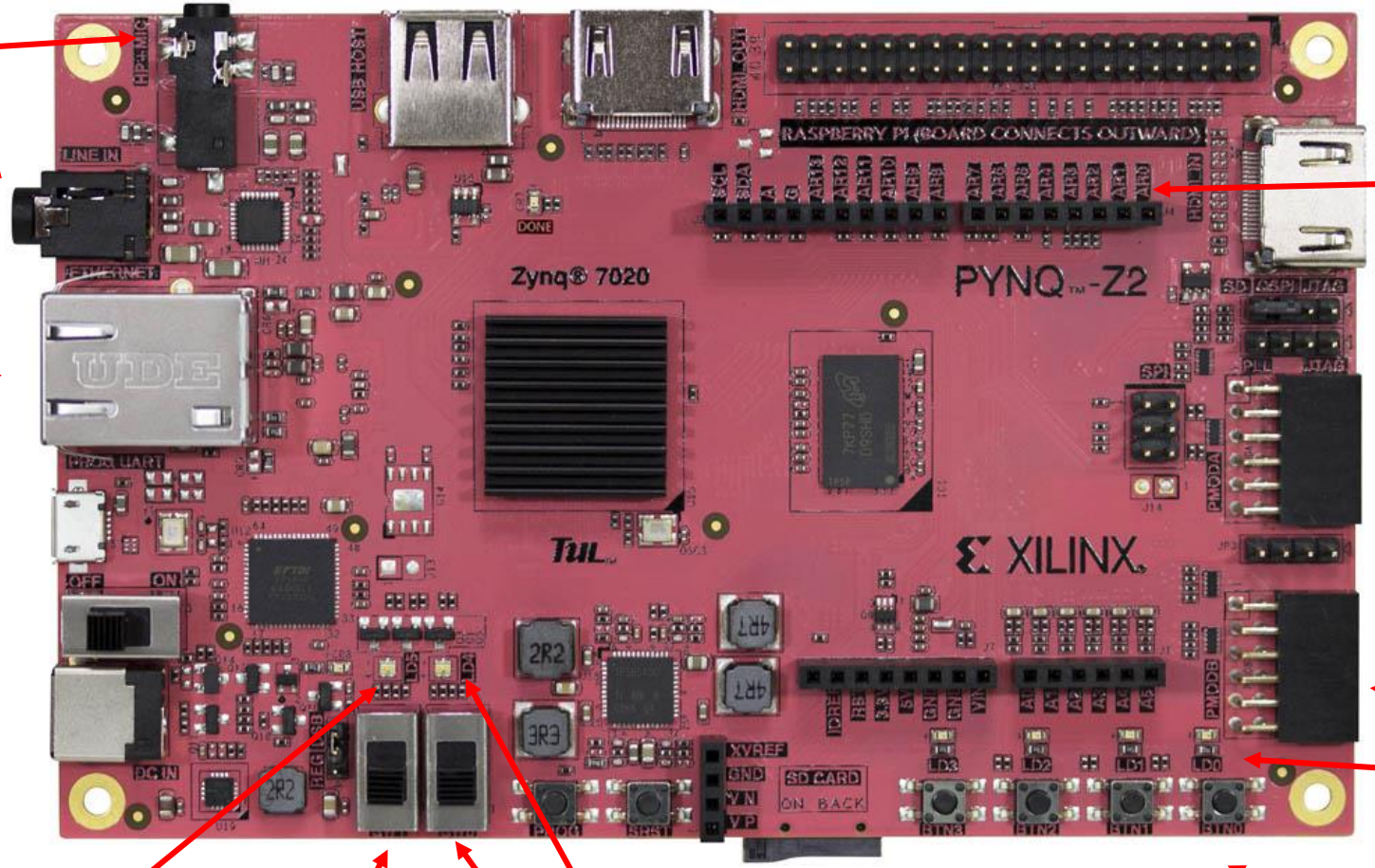
- Zynq7000 (7z020) has 13300 logic slices, each containing
  - 4 6-input LUTs (with 64x1-bit memory capability)
  - 8 flip-flops (Register and shift register functionality)
  - Cascadable adders
- Synthesis will figure out how the logic is mapped to this structure in a most optimal way



A question

- Would you consider programming Xilinx 7-series FPGA with PALASM? Or by drawing a schematics of the logic gates?
- => Hence, VHDL ☺

PYNQ-Z2



HDMI In/Out

RasPi connector

Audio In/Out

Arduino connector

HDMI In/Out

Ethernet

PMOD A ja[0:7]

PMOD B jb[0:7]

LEDs led[0-3]

RGB LED 4 ld4_[r.g.b]

RGB LED 4 ld4_[r.g.b]

Switch 1 sw[1]

Switch 0 sw[0]

Buttons btn[0-3]