What are FPGAs and how do they work

Andre Zibell
Julius-Maximilians-Universität Würzburg

RD51 electronics school
CERN, 5.2.2014

Images: Wikipedia

Motivation

- FPGA : Field Programmable Gate Array
  - -> can be reprogrammed at any time (unlike ASICs)
  - -> Array of logic gates
- A gate implements a basic logic function, like OR, AND, NAND, etc...
- Any logic function, complex or not, can be built from a number of interconnected „gates“.
- With enough gates (even of a single type) it is possible to build up any type of digital circuitry (Even processors)

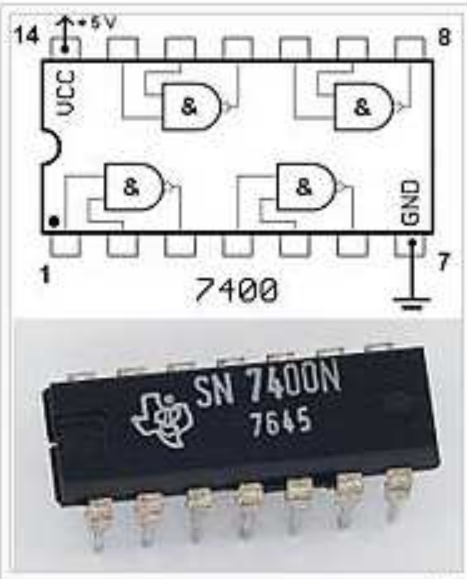
INPUT	OUTPUT
A	NOT A
0	1
1	0

INPUT		OUTPUT
A	B	A NAND B
0	0	1
0	1	1
1	0	1
1	1	0

INPUT		OUTPUT
A	B	A NOR B
0	0	1
0	1	0
1	0	0
1	1	0

INPUT		OUTPUT
A	B	A OR B
0	0	0
0	1	1
1	0	1
1	1	1

INPUT		OUTPUT
A	B	A AND B
0	0	0
0	1	0
1	0	0
1	1	1

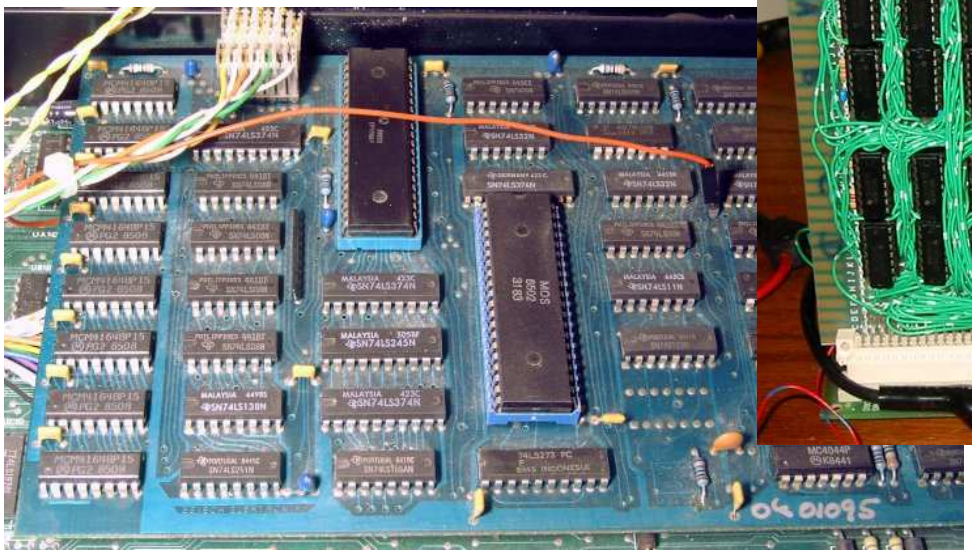


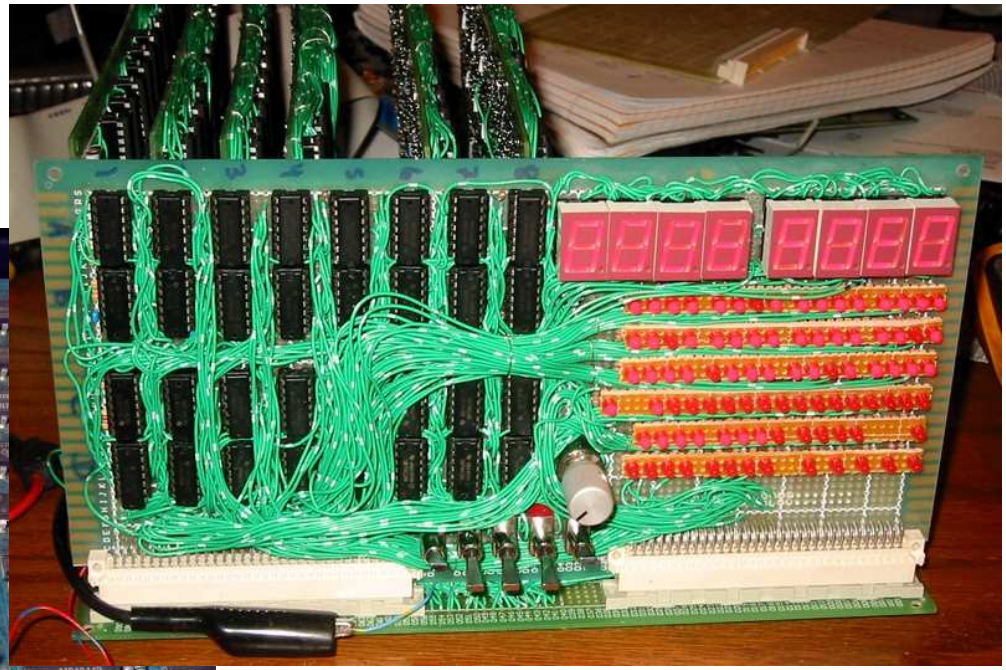
The 7400 chip, containing four NANDs. The two additional pins supply power (+5 V) and connect the ground.

Images: cbmhardware.de, hackedgadgets.com

Digital electronics in the 70’s

- Use of may logic gate chips, to perform complex operations
- Wire routing difficult and prone to errors
- Limited functionality due to available board space

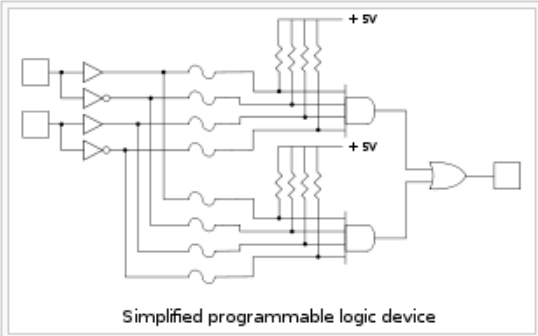




Images: wikipedia

Evolution of logic density

- Development of chips, that contain several (different) logic gates that were cascadable
- Programmable Array Logic (PAL, late 70‘s):
One-time-programmable interconnection of „dozens of logic gates“ by burning of fuses
- Generic Array Logic (GAL, 1985)
Replaces several PALs, re-programmable
- Complex Programmable Logic Device (CPLD)
Contain hundreds of logic gates, devices with hundreds of pins available
- -> FPGAs
Contain not only many thousand (or million) logic gates, but much more...



Simplified programmable logic device

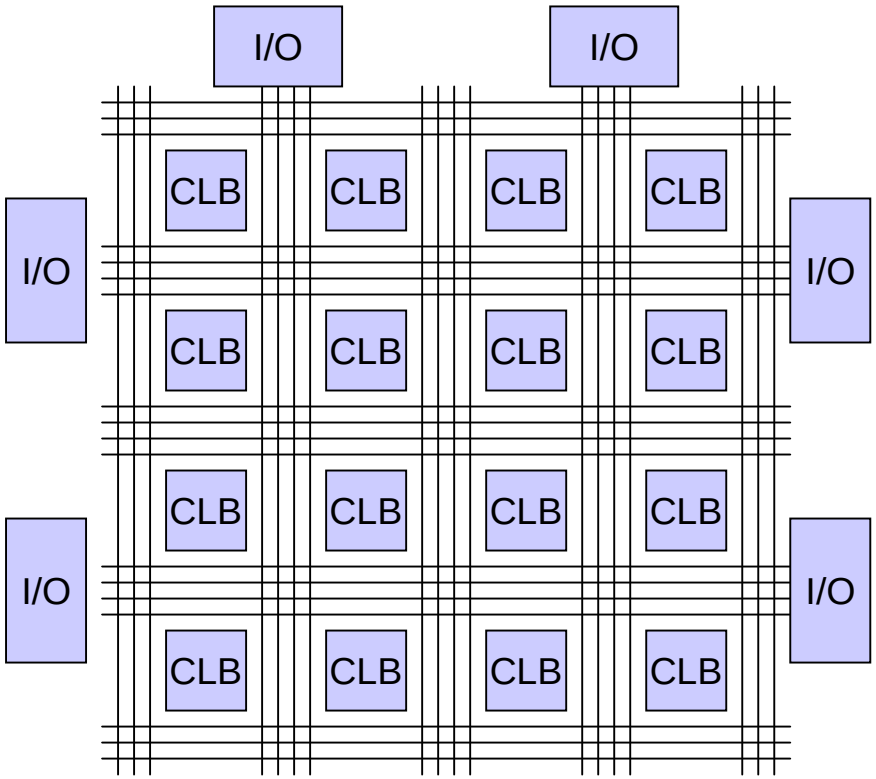
A simplified PAL device. The programmable elements (shown as a fuse) connect both the true and complemented inputs to the AND gates. These AND gates, also known as product terms, are ORed together to form a sum-of-products logic array.


Lattice GAL 16V8 and 20V8

An Altera MAX 7000-series CPLD with 2500 gates.

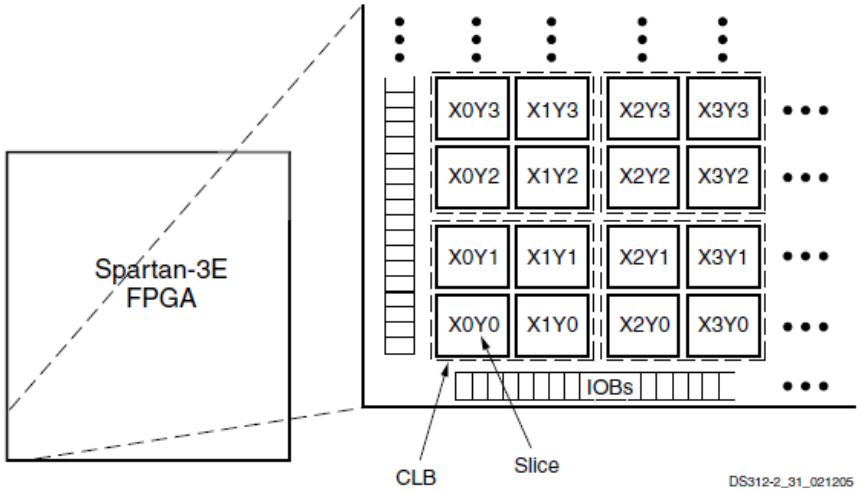
FPGA Layout

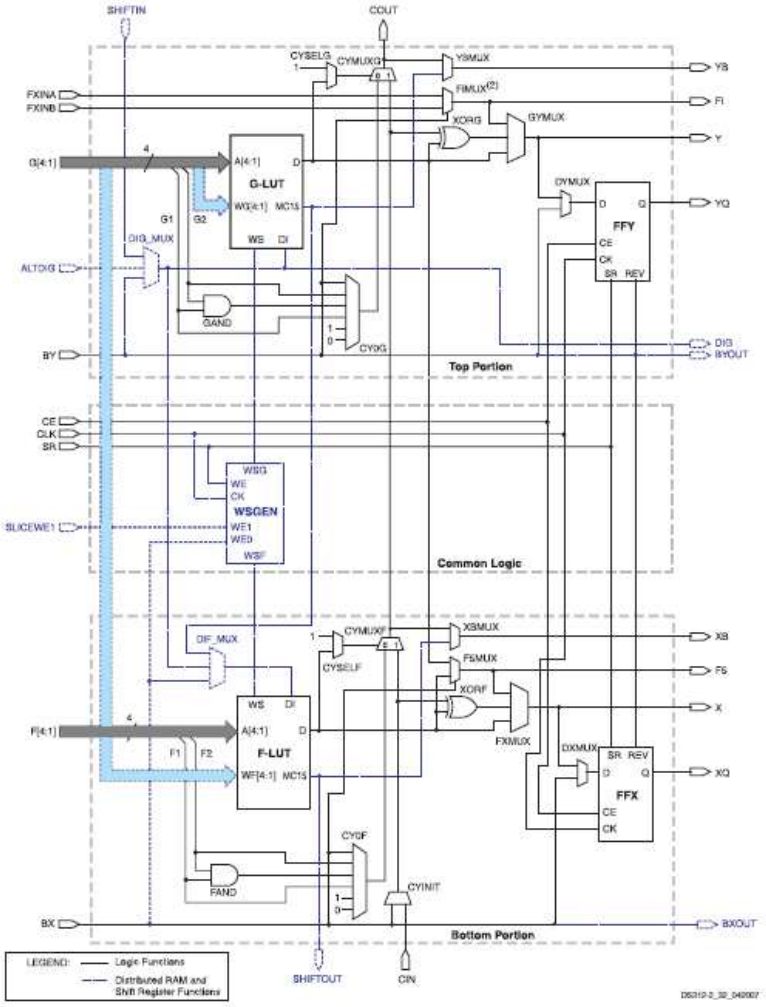
- An FPGA contains a very (!) large number of „Configurable Logic Blocks“ (CLB)
- A programmable interconnection matrix propagates signals in between blocks
- I/O blocks connect to the world outside of the FPGA
- This is a simplified model – modern FPGAs contain much more powerful components!



Images: Xilinx Spartan 3E Datasheet

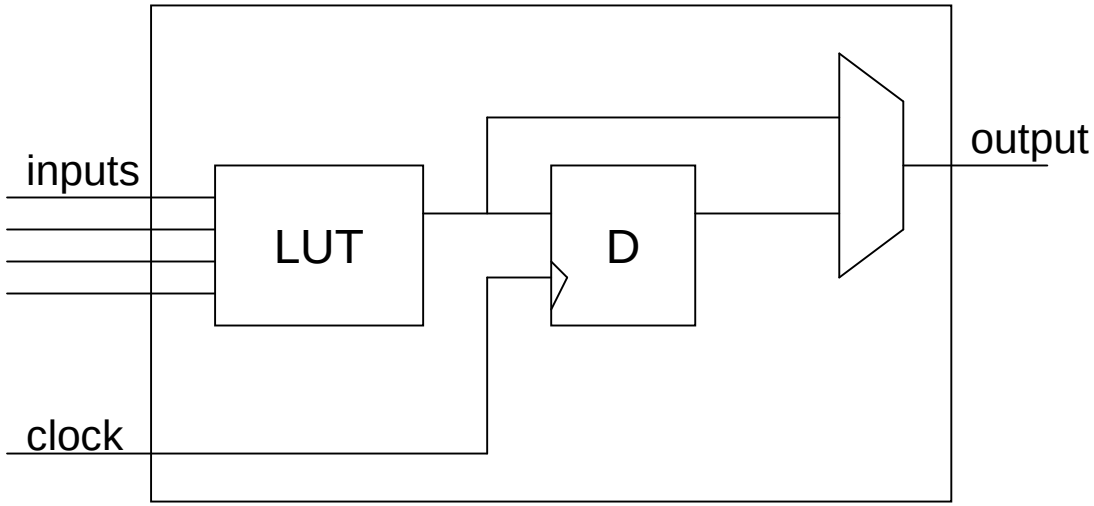
Configurable logic blocks





Logic Elements

- CLB contain „logic cells“ like below (simplified!)
- A reprogrammable LookUp Table (LUT) with several inputs forms the logic function or stores data
- Additional flip-flops serve as storage element, or synchronize the processes to a clock input



Additional blocks

- A modern FPGA contains many additional specialized blocks
  - Internal Memory
  - DDR Memory controllers
  - Clock Multiplicators / Dividers („Phase Locked Loop“)
  - Digital Signal Processing (DSP)
  - High-speed transceivers (multi-Gigabit/s)
  - PCI-express interfaces
  - Gigabit-Ethernet controllers
  - Processors (ARM, ...)
  - ...

What you can get today

- Only few vendors for FPGAs
- Market dominated by Xilinx and Altera (80%)
- Wide range of FPGA complexity (and cost): January 2014 examples.

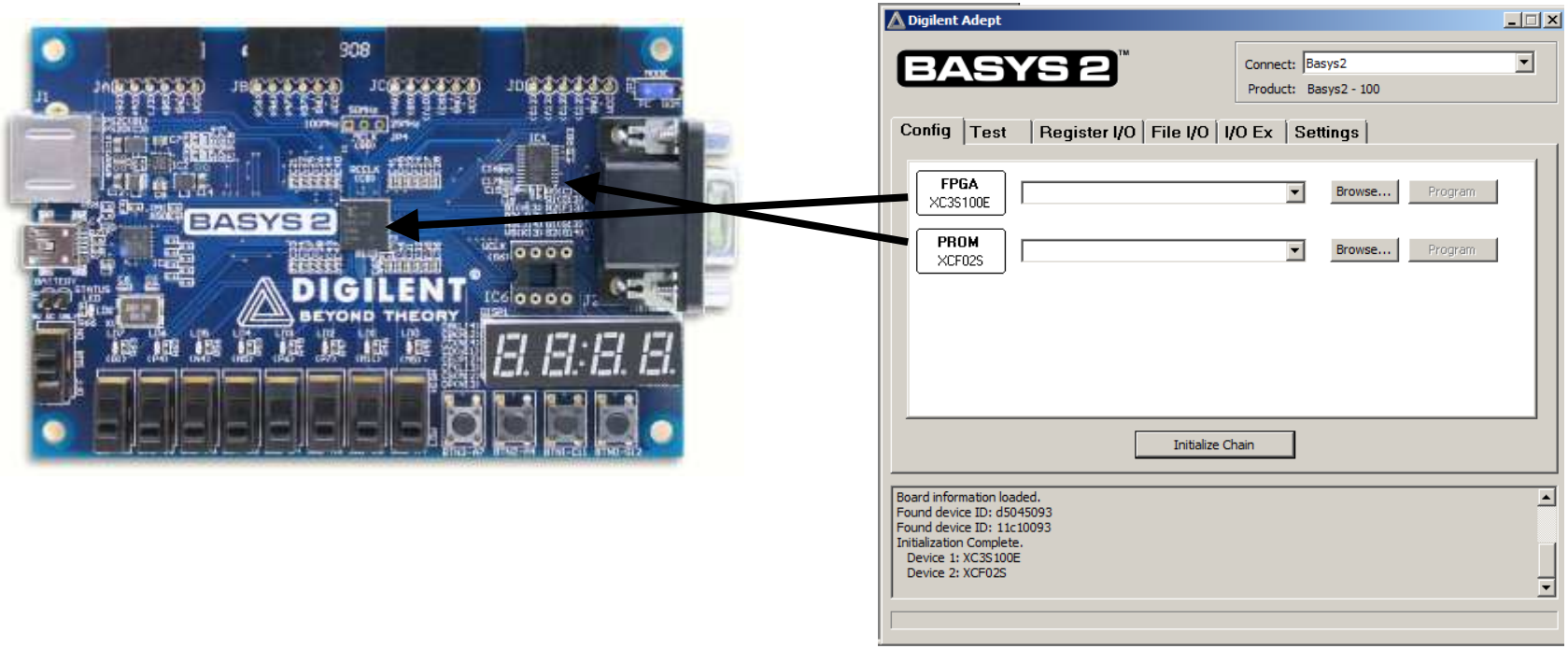
FPGA	Xilinx Spartan-3E XC3S100E (Basys2-Board)	Xilinx Virtex-6 XC6VLX130T (SRU Board)	Xilinx Virtex-7 XC7V2000T	Xilinx Virtex „UltraScale“ XCVU440
I/O lines	83	600	1200	1456
Logic cells	2,160 (4-input LUT)	128,000 (6-input LUT)	1,954,560	4,407,480
Internal Block RAM memory	72 kBits	9,500 kBits	46,512 kBits	88,000 kBits
Price (@1 pcs) www.digikey.com	8.47€	817.39€	16,286.94€	???

FPGA configuration

• On powerup, a FPGA is completely unconfigured („empty“) (there are exceptions...)

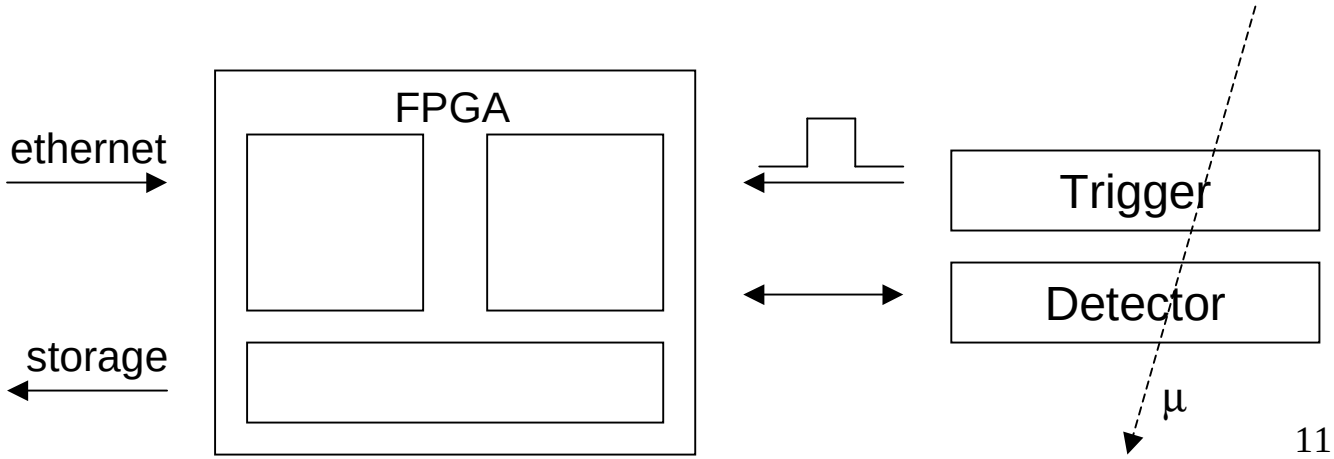
• External memory is needed to hold the configuration. When the FPGA powers up, it reads its configuration from the attached non-volatile memory

• When programming the board, you can decide to write either to the FPGA itself (temporary) or to the permanent memory (valid until you overwrite it)



Parallelism

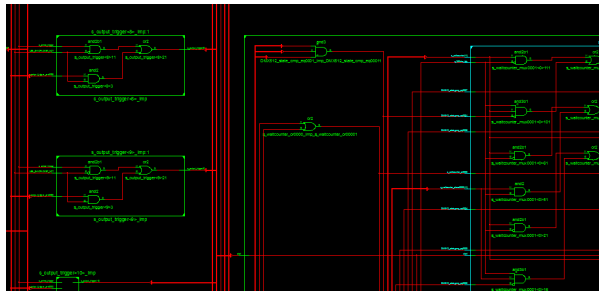
- A (sigle core) **processor** can only execute one command at a time. -> **Software**
- Even worse: (most) Computer operating systems switch in between tasks very „slowly“
- **FPGAs** do not have a „program memory“ they go through line by line.
Instead every functionality you configure inside the FPGA gets executed permanently and at the same time! -> **Hardware**
- At every point in time you can predict the behaviour of your design (->simulation)
on a 1-clockcycle-basis (e.g. 10 ns at 100 MHz)
- Very useful (not only!) in Physics data acquisition systems, where precise timing is absolutely important!

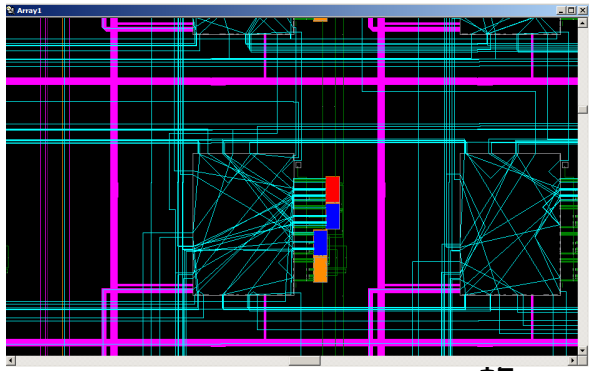


Workflow

- Design („describe“) your circuit,
using one of several „languages“:
    - Hardware Description Language (HDL)
    like **VHDL** or Verilog
    - SystemC, SystemVerilog, ...
    - Labview, ...
    - ...
- Use a Synthesis tool, to transfer your desired functionality onto basic logic elements
- ...
- Vendor-specific tools then try to fit these elements into the available FPGA-ressources, including their interconnection („Place and Route“)
- The result of this configuration (.bit file) can then be programmed into the FPGA

```
--set default gateway
if s_initcounter = 1 then
   s_addr <= "000" & x"001";
   s_dout <= s_wiz_GW(31 downto 24);
elsif s_initcounter = 2 then
   s_addr <= "000" & x"002";
   s_dout <= s_wiz_GW(23 downto 16);
elsif s_initcounter = 3 then
   s_addr <= "000" & x"003";
   s_dout <= s_wiz_GW(15 downto 8);
elsif s_initcounter = 4 then
```





Why use FPGAs?

+ Implement any logic functionality
+ Change the design at any time
+ parallel processing of data
+ many I/O pins to interface other electronics

- expensive (per piece)
- need lots of power / different voltages
- tricky board design (BGA chips, Multilayer, ...)
- complex tools

Microcontrollers?

+ cheap
+ easy to programm
+ simple board layout

- slow
- no parallel execution of tasks
- limited to internal execution set

ASICs?

+ Implement any logic functionality (plus analog stuff)
+ parallel processing of data
+ even faster and larger than FPGAs
+ cheap (per piece)

- expensive (one time)
- non-reprogrammable
- Months from design to chip

Recommended ressources

- http://www.eevblog.com/2013/07/20/eevblog-496-what-is-an-fpga/
- http://www.fpga4fun.com/
- http://embeddedmicro.com/tutorials/mojo
- http://www.digilentinc.com/Products/Detail.cfm?Prod=BASYS2