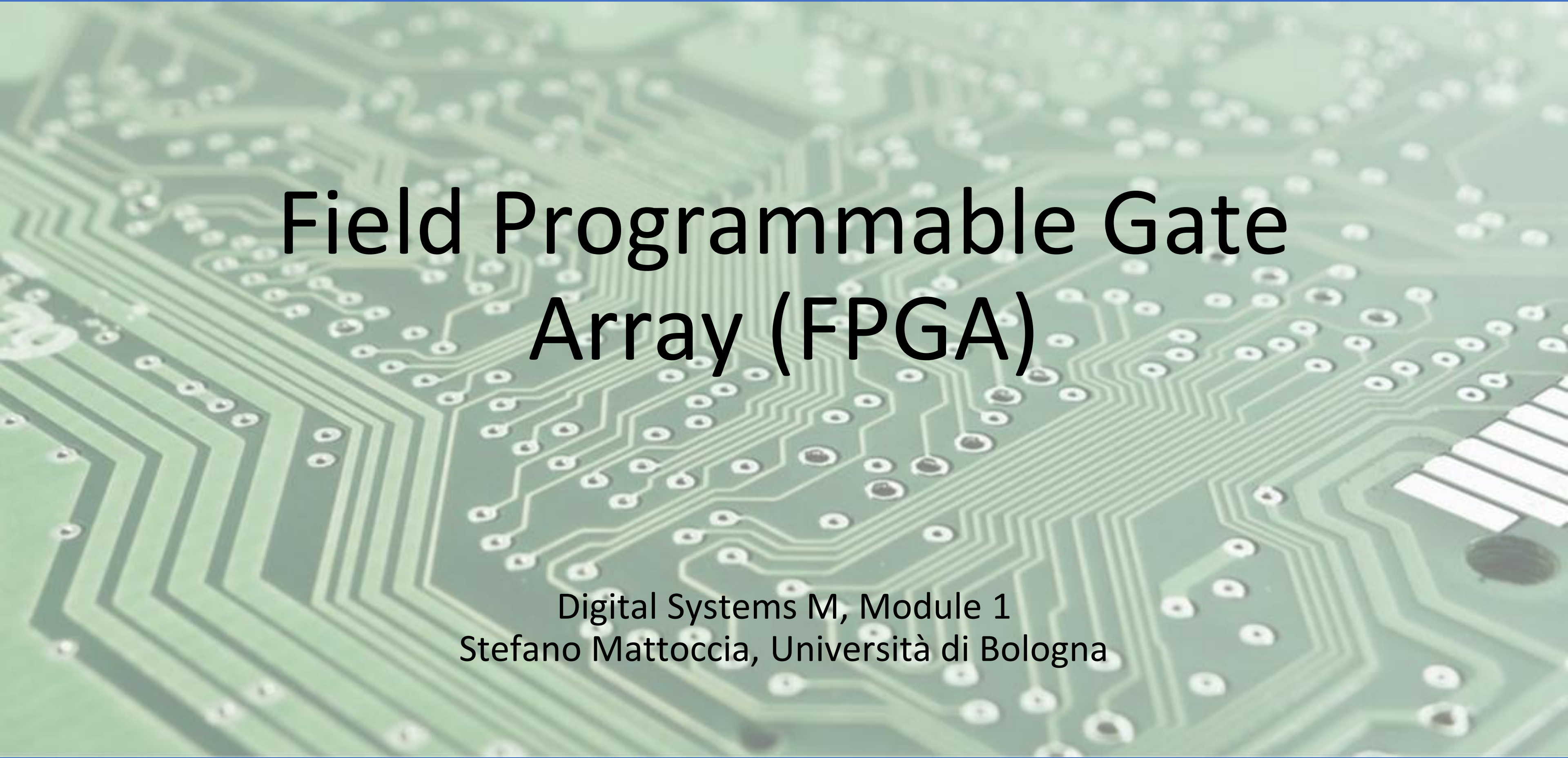
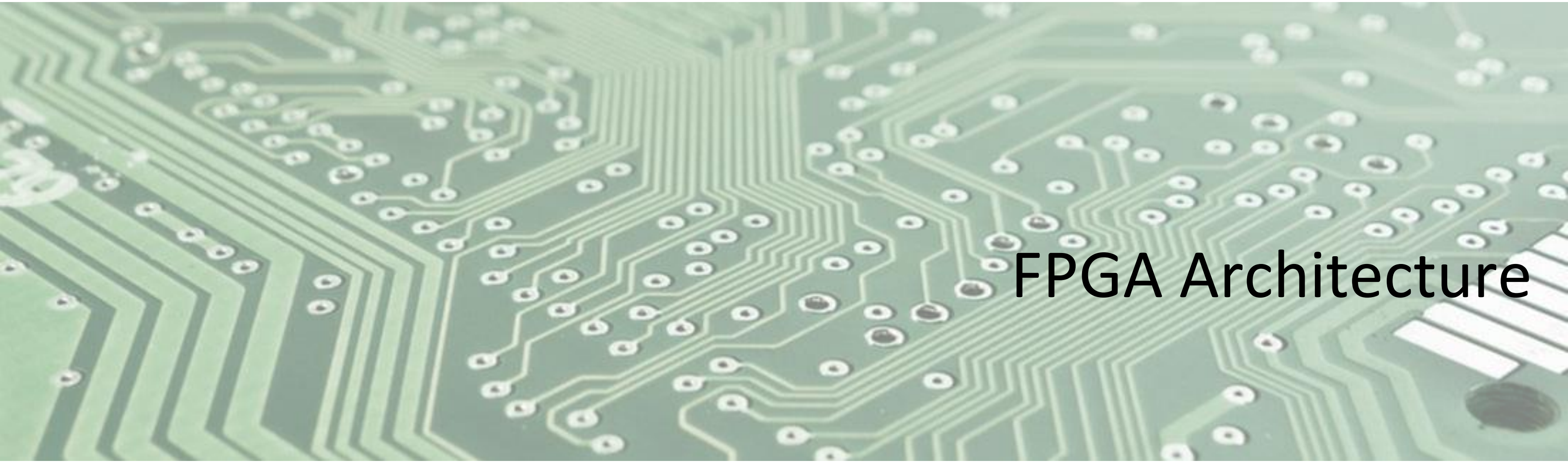


Field Programmable Gate Array (FPGA)

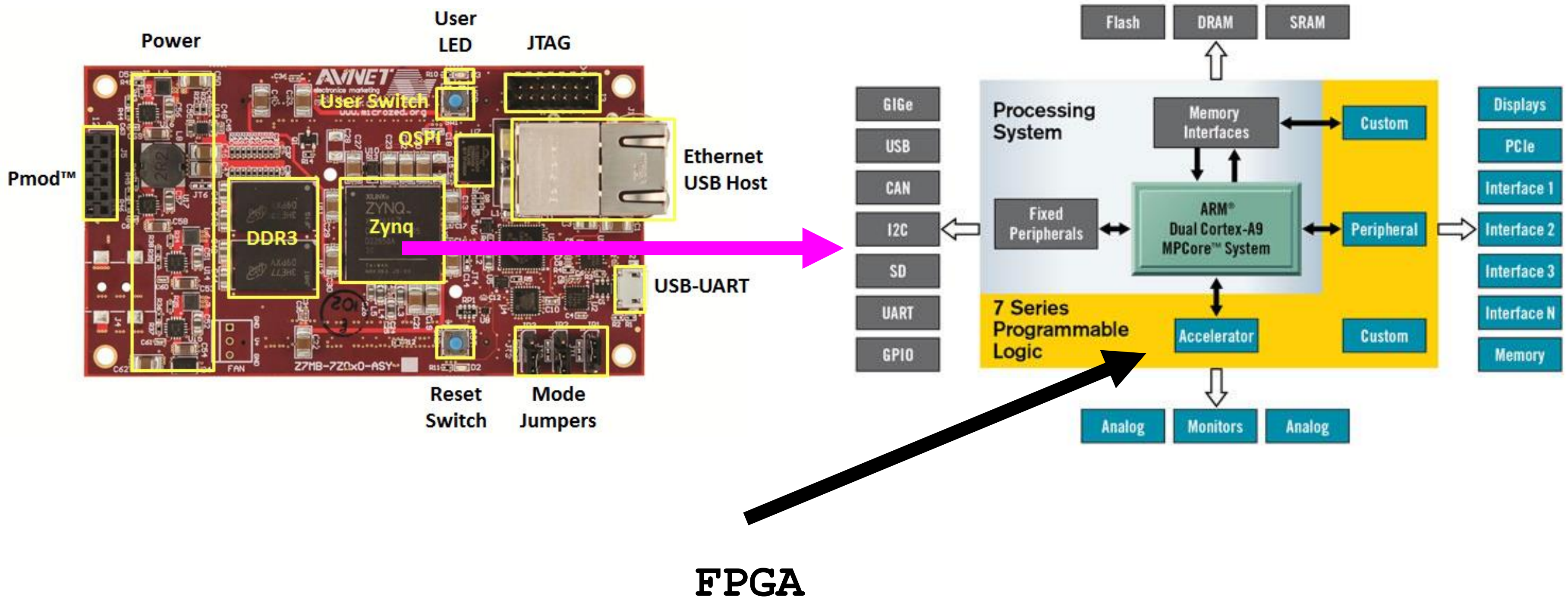
Digital Systems M, Module 1
Stefano Mattoccia, Università di Bologna



FPGA Architecture

FPGA Architecture

Although a design methodology based on C/C++ will be adopted, a deep knowledge about how an FPGA architecture is organized allows to achieve optimal results



FPGA Architecture

Background

- ✓ 70s: gates, SSI, MSI, LSI
- ✓ Then: Programmable devices (logic functions, pins and connections) even *on field* such as EPROM, PAL, PLA, PLD (studied in Reti Logiche T)
- ✓ 80s/90s: deployment of large-scale integration devices FPGA (e CPLD), programmable *on field* through HDL language

FPGA Altera

FPGA Xilinx

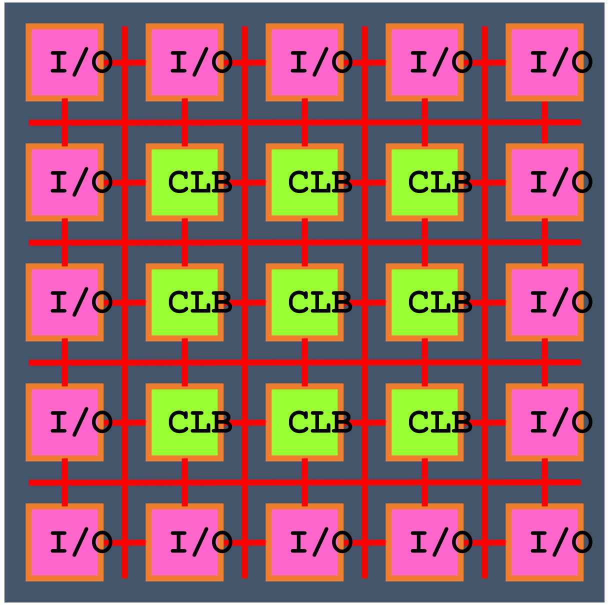
FPGA Architecture

- ✓ FPGA: made of Configurable Logic Blocks (CLB) that can be trasparently connected through *high-level* languages (HDL or HLS)
- ✓ *Programmable* (and re-programmable) *on field*
- ✓ Low-cost, low *time to market*, supported by high-level languages (C/C++, **Python**) or Hardware Description Languages (VHDL or Verilog)
- ✓ Ideal for rapid prototyping (e.g. CPU design)
- ✓ Ideal for low-powered devices
- ✓ Allows for high abstraction -> hardware implementation of traditional logical networks as well as of **algorithms**
- ✓ Allows for core intergration (both *soft* and *hard*)
- ✓ FPGA vs ASIC ($$) – for both, HDL languages can be used

FPGA Architecture

FPGA Architecture

An FPGA is made by several *Configurable Logic Blocks* (CLB) that can be connected.
Each CLB function and connections is designed by the developer by means of on-field programming, that can be repeated multiple time in order to develop novel architectures (possibly, *infinite* times).



- ✓ Many CLBs (thousands)
- ✓ Usually, internal RAM is available (Block RAM, few hundreds KB)
- ✓ Some blocks are devoted to I/O
- ✓ Several adders, multipliers, etc. available

FPGA Architecture

- ✓ Several FPGA producers and technologies exist. FPGA developed by different producers are different by means of two aspects:

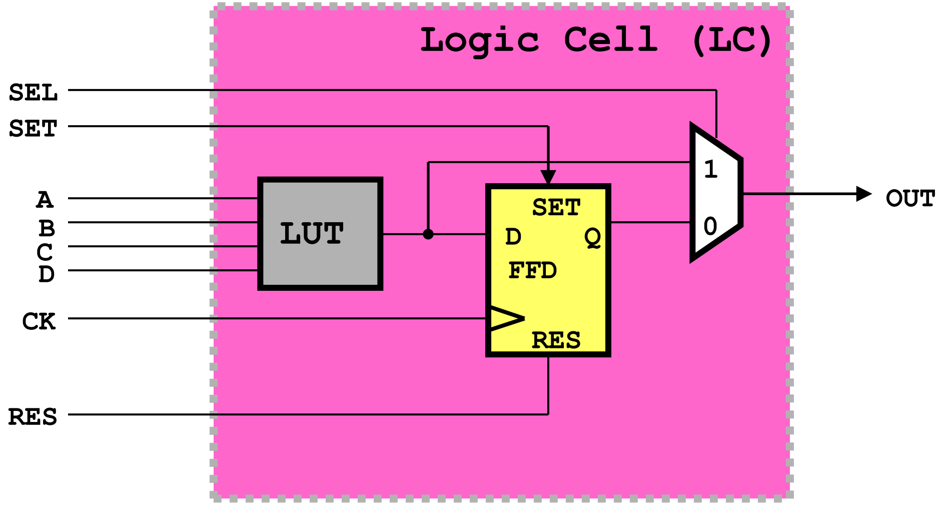
  - ✓ Technology used for connections
    - - *Fusibili*
    - - Flash memories
    - - SRAM memories

  - ✓ An introduction (1996) to technologies for programmable devices (CPLD, FPGA, etc) : www.eecg.toronto.edu/~jayar/pubs/brown/survey.pdf

  - ✓ CLB structure
    - ✓ A detailed analysis of CLBs is beyond the scope of this course. Anyway, it is interesting to fully understand how CLB are orgenized

FPGA Architecture

The figure shows the logical network, Logic Cell (LC), implementing a CLB

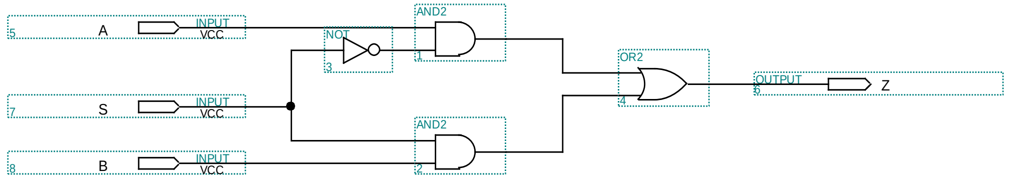


The LUT block (*Look-up-table*) is a programmable, combinatorial network.

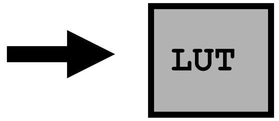
It can be re-programmed to act like a shift-register or a memory (distributed RAM)

FPGA Architecture

Combinatorial functions & Look-up-table



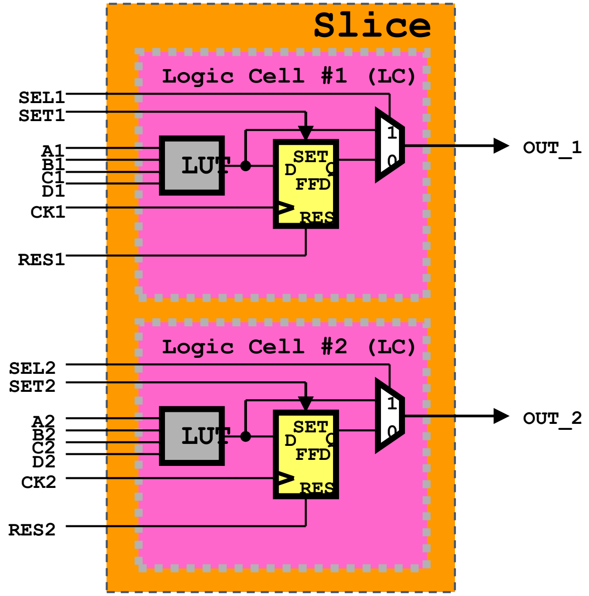
S	A	B	Z
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1



The LUT can be re-programmed to act like a shift-register or a memory (distributed RAM)

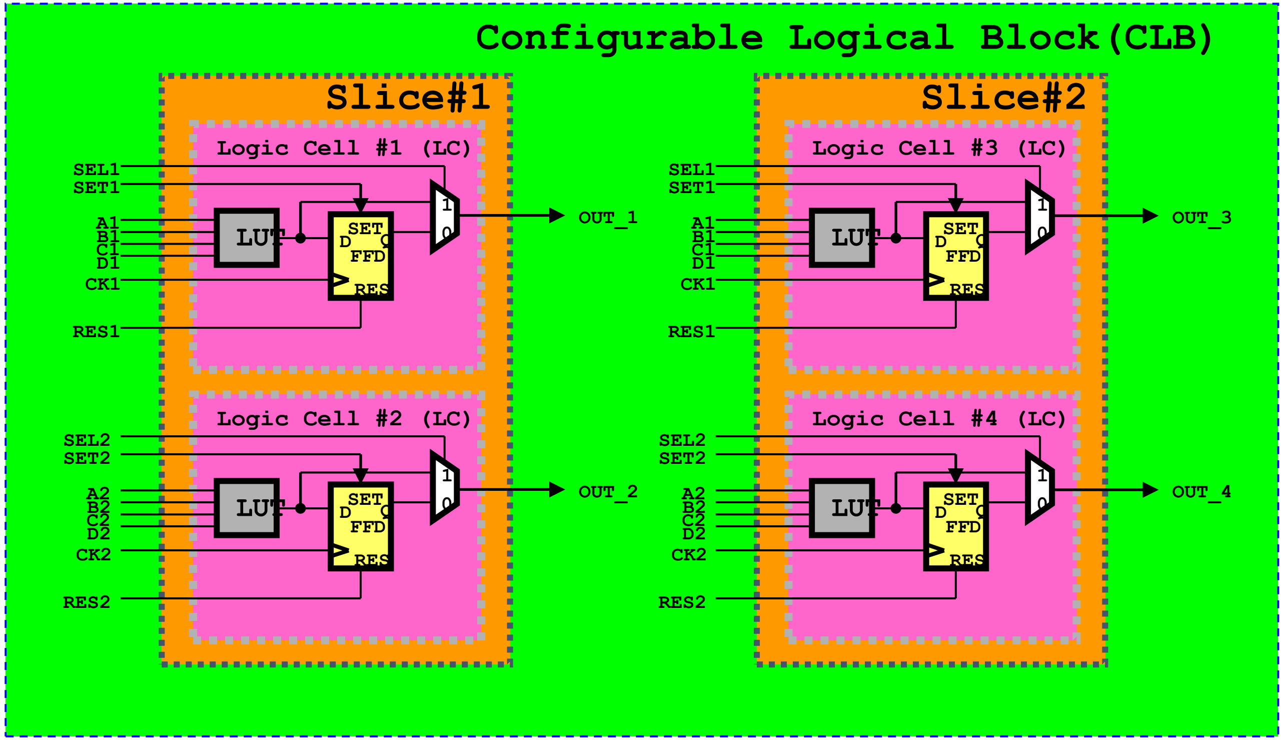
FPGA Architecture

Logic Cells are typically grouped in slices, each one containing some (2,4,…) Logic Cells



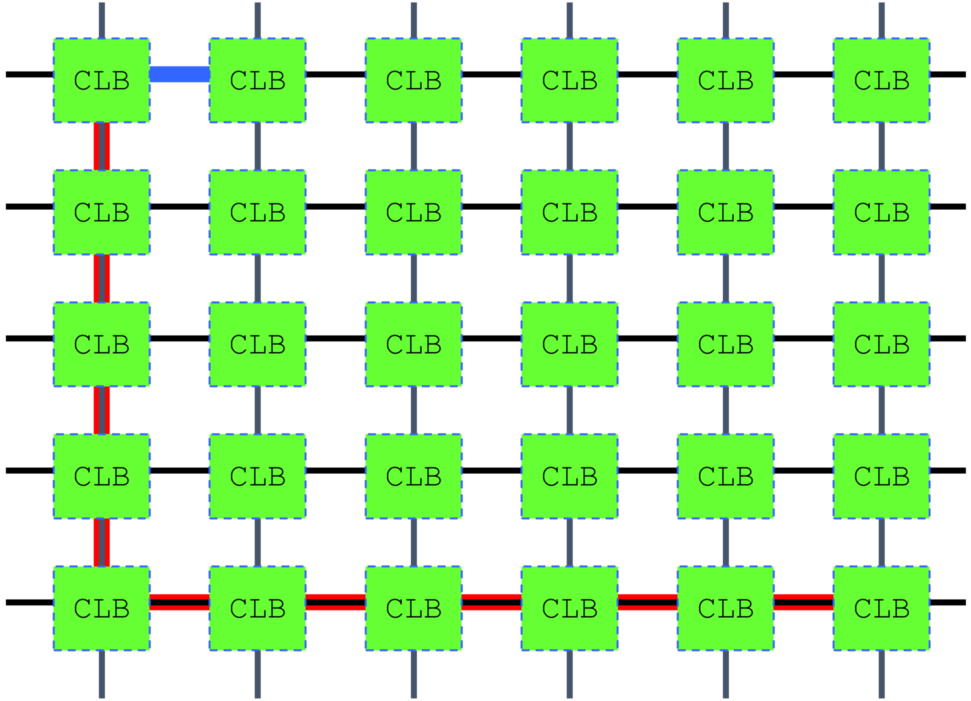
FPGA Architecture

Finally, CLBs are obtained by grouping some slices (2,4,...) inter-connected (details aabout possible connections are not shown in the figure)



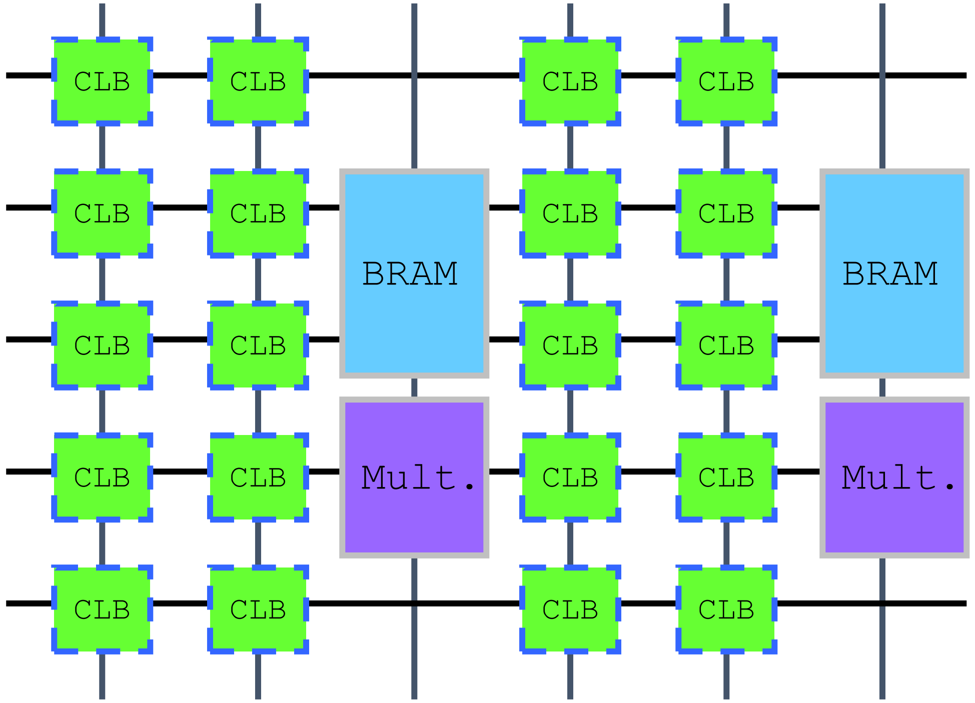
FPGA Architecture

Inside the FPGA there are the possible connections between CLBs, clocks, etc



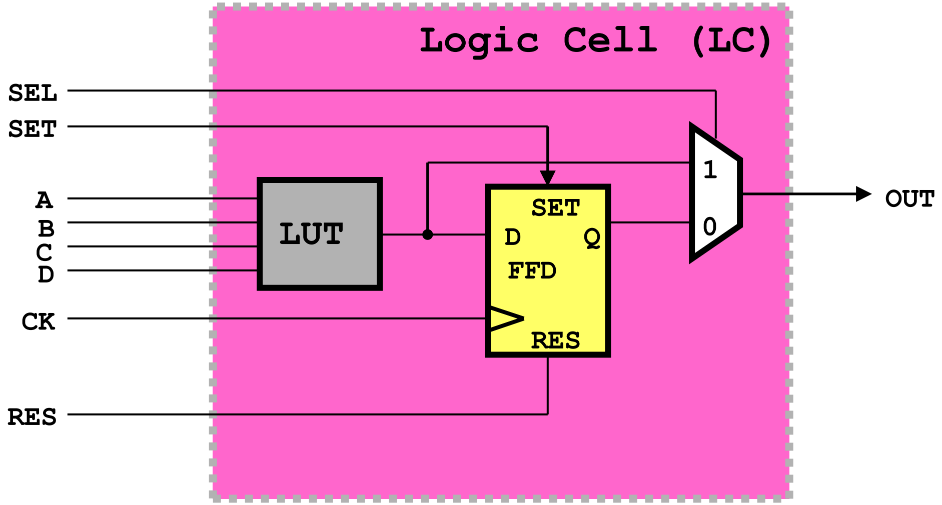
FPGA Architecture

Actually, there are other function units between the different CLBs (Block RAM, multipliers,...)



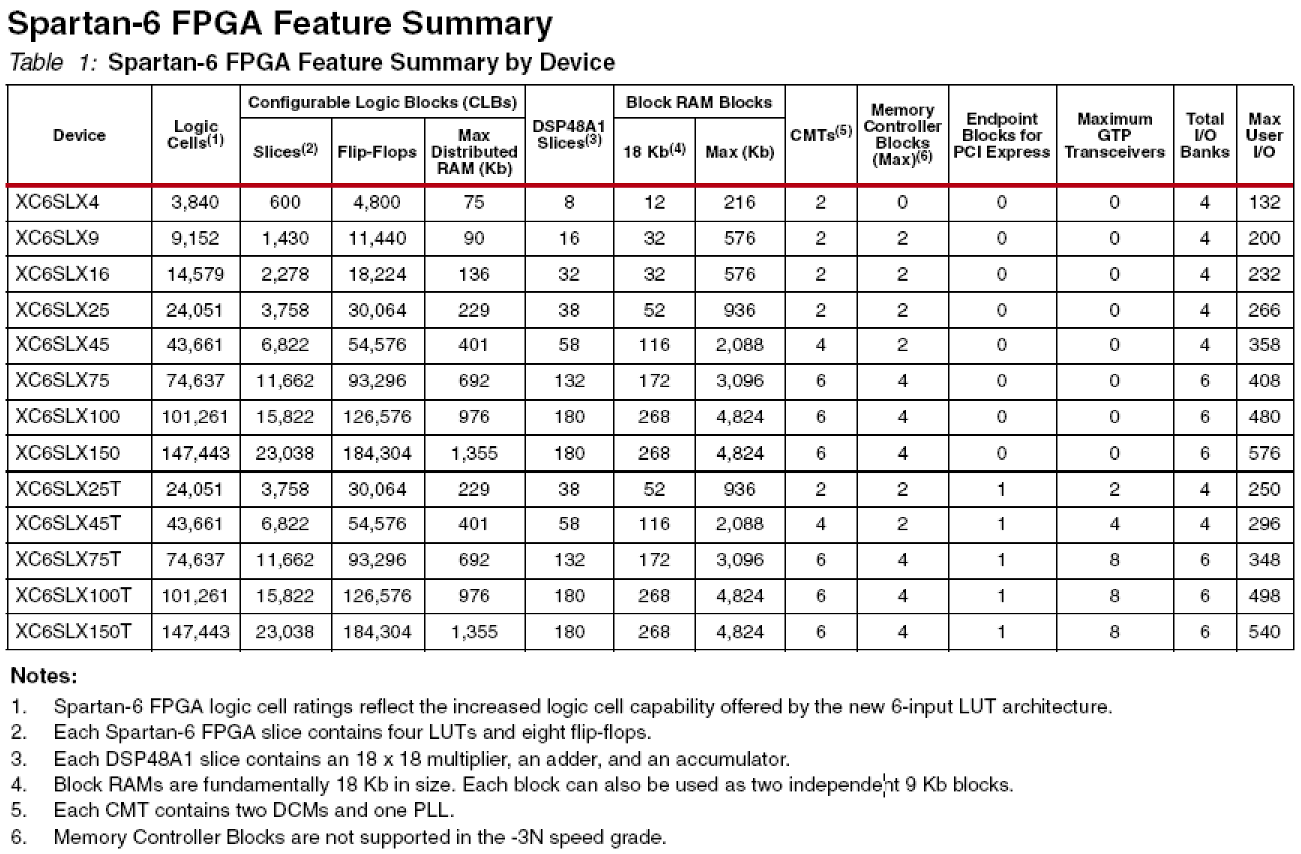
FPGA Architecture

Logic Cell



In the next slide, Xilinx Spartan 6 features. Complete features list available at:
http://www.xilinx.com/support/documentation/data_sheets/ds160.pdf

FPGA Architecture



Spartan-6 FPGA Feature Summary

Table 1: **Spartan-6 FPGA Feature Summary by Device**

Device	Logic Cells[1]	Configurable Logic Blocks (CLBs)			DSP48A1 Slices[3]	Block RAM Blocks		CMTs[5]	Memory Controller Blocks (Max)[6]	Endpoint Blocks for PCI Express	Maximum GTP Transceivers	Total I/O Banks	Max User I/O
		Slices[2]	Flip-Flops	Max Distributed RAM (Kb)		18 Kb[4]	Max (Kb)						
XC6SLX4	3,840	600	4,800	75	8	12	216	2	0	0	0	4	132
XC6SLX9	9,152	1,430	11,440	90	16	32	576	2	2	0	0	4	200
XC6SLX16	14,579	2,278	18,224	136	32	32	576	2	2	0	0	4	232
XC6SLX25	24,051	3,758	30,064	229	38	52	936	2	2	0	0	4	266
XC6SLX45	43,661	6,822	54,576	401	58	116	2,088	4	2	0	0	4	358
XC6SLX75	74,637	11,662	93,296	692	132	172	3,096	6	4	0	0	6	408
XC6SLX100	101,261	15,822	126,576	976	180	268	4,824	6	4	0	0	6	480
XC6SLX150	147,443	23,038	184,304	1,355	180	268	4,824	6	4	0	0	6	576
XC6SLX25T	24,051	3,758	30,064	229	38	52	936	2	2	1	2	4	250
XC6SLX45T	43,661	6,822	54,576	401	58	116	2,088	4	2	1	4	4	296
XC6SLX75T	74,637	11,662	93,296	692	132	172	3,096	6	4	1	8	6	348
XC6SLX100T	101,261	15,822	126,576	976	180	268	4,824	6	4	1	8	6	498
XC6SLX150T	147,443	23,038	184,304	1,355	180	268	4,824	6	4	1	8	6	540

Notes:

1. Spartan-6 FPGA logic cell ratings reflect the increased logic cell capability offered by the new 6-input LUT architecture.
2. Each Spartan-6 FPGA slice contains four LUTs and eight flip-flops.
3. Each DSP48A1 slice contains an 18 x 18 multiplier, an adder, and an accumulator.
4. Block RAMs are fundamentally 18 Kb in size. Each block can also be used as two independent 9 Kb blocks.
5. Each CMT contains two DCMs and one PLL.
6. Memory Controller Blocks are not supported in the -3N speed grade.

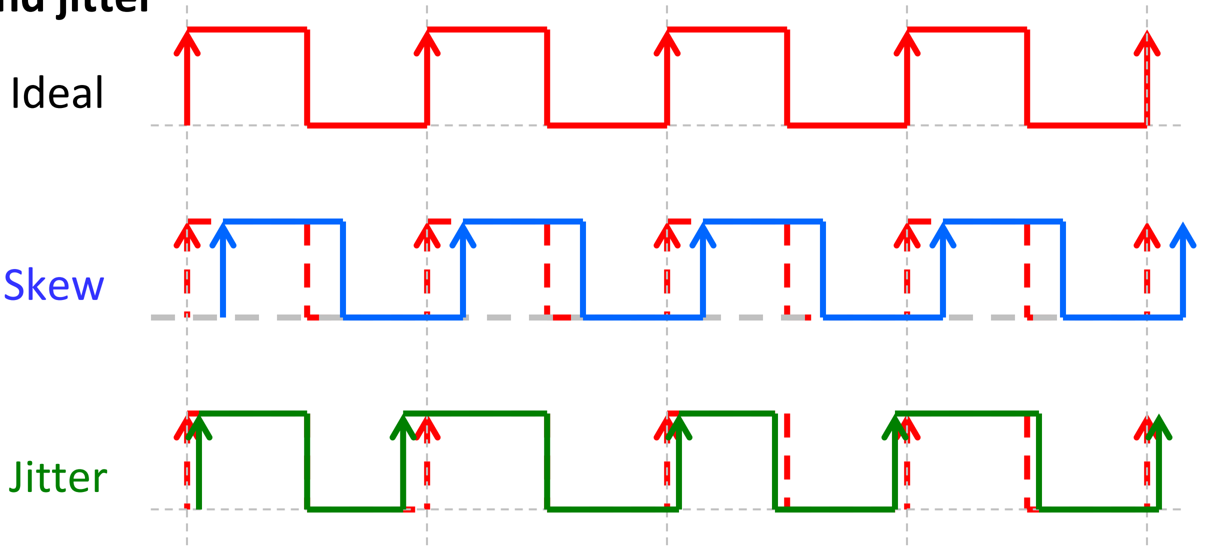
FPGA Architecture

FPGA and clock

- ✓ FPGA are usually configured to implement synchronous networks (e.g. with direct synthesis to implement algorithms).
- ✓ Thus, atleast one clock signal is required, used for SSR (Sequential Synhronous networks) building up the project deployed on the FPGA
- ✓ Usually, several clock domains coexist in complex projects, i.e. different modules use different clocks (in both frequency and duty-cycle). Thus, there are problems due to data transfer and comunication **across clock domains**
- ✓ How to generate a stable clock signal at a desired frequncy? FPGA produces make available specific components, called DCM and PLL.

FPGA Architecture

Clock: skew and jitter



- Skew – the signal is delayed or anticipated with respect to the reference clock. Phase shift (but same frequency)
- Jitter – the signal changes its frequency (dynamically)
- Unfortunately, in real scenarios, *Skew* and *Jitter* are not mutually exclusive

FPGA Architecture

Digital Clock Manager (DCM) e Phase-Locked Loop (PLL)

- ✓ In order to generate stable clock signals, *clock-managers* are typically available (for Xilinx, DCM and PLL)
- ✓ DCM and PLL are frequency multiplier/divider used for many purposes:
  - ✓ To generate stable signals, starting from an external periodical signal (it is possible to multiply or divide frequency)
  - ✓ To reduce *skew* and *jitter*
  - ✓ To generate signals with a certain shift with respect to a given reference signal

FPGA Architecture

- ✓ In FPGAs we can use common devices by means of *IP-Core* (the hardware counterpart of software libraries)
- ✓ Sometimes, for the sake of efficiency, IP-Core are based on logic functions wired in the FPGA circuits by the producer. In addition to DCM/PLL, some popular IP-Cores are:
  - ✓ Memory controllers
    Allows for transfers with external memory devices (e.g., DDR, DDR2, DDR3)
  - ✓ Serializers/Deserializers (SERDES)
    Devices for series/parallel conversion of high-frequency signals
  - ✓ Communication controllers
    Devices for handling high-bandwidth transfers (e.g., 10/100/1000 Gb Ethernet)

FPGA Architecture

IP-cores and protection

- ✓ An IP-Core configures an FPGA (or part of it) to implement a specific function
  - ✓ as for software, IP-Cores can be sold or made freely available (e.g. www.opencores.org)
  - ✓ as for software, it can be copied (SRAM based approach is the most vulnerable)
  - ✓ As for software, protection mechanisms exist against illicit uses

- ✓ Producers specialized in development and sale of IP-cores exist. With the obvious differences, these producers are similar to software producers

FPGA Architecture

FPGA: frequency and performance

- ✓ FPGA frequencies are not high compared to common CPUs (few hundreds MHz, often lower)
- ✓ How can be performance competitive with such low frequencies?
- ✓ By configuring FPGA for the highest parallelism!
- ✓ On the positive side, by reducing frequency we can greatly reduce power requirements…

FPGA Architecture

Programming languages

- ✓ How to configure an FPGA by connecting CLBs, etc to implement a specific logical network?
- ✓ For this purpose (synthesis), HDL or high-level tools (HLS) and compilers are used, that generate configuration data (*bitstream*)
  - ✓ The *bitstream* is a «software» configuring the FPGA for a given purpose
- ✓ It is generated from HDL code (or schematics) or from high-level synthesis tools (almost) without intervention from the developer (unless specifically desired)

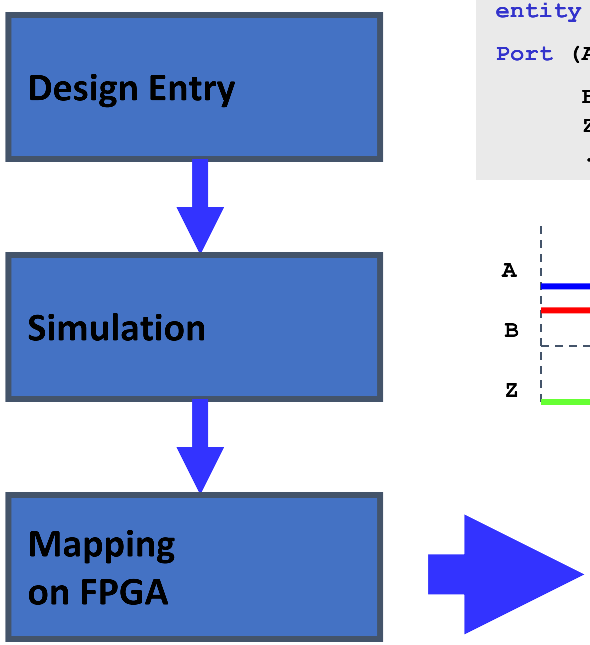
FPGA Architecture

FPGA development with HDL (1/2)

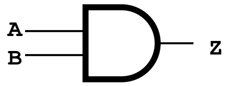
- ✓ High-level languages for digital circuits simulation, becoming popular for synthesis as well
- ✓ Goal: to reduce development costs and time budget, easing the managment of large projects
- ✓ HDL (e.g. VHDL) allows for modelling hardware behavior, i.e. parallelism and delays
- ✓ Typically, used to program FPGAs
- ✓ Also used to design ASIC, processors, etc
- ✓ IEEE Standard

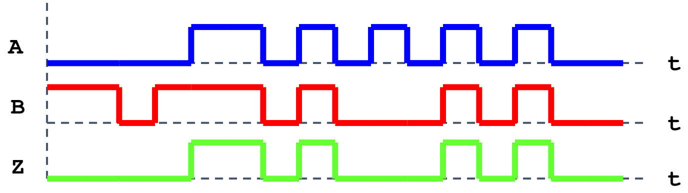
FPGA Architecture

FPGA development with HDL (2/2)



```
entity my_AND is
Port (A : in   BIT;
      B : in   BIT;
      Z : out   BIT);
      . . .
```





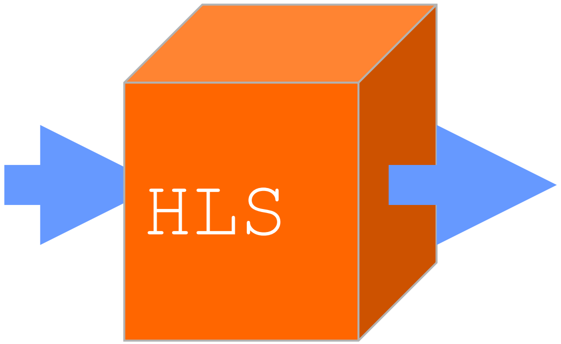
Each step is performed by the CPU!

FPGA Architecture

FPGA development with HSL tools:

```
#include <math.h>

int function()
{
 int i;
 int r = 0;

 for (i=0;i<100;i++)
 {
  . . . . .
  . . . . .
 }

 return r;

}
```

C, C++, etc



```
entity function is
Port(x : in  type;
     Y : in  type
     r : out type );
end function;


architecture hls of
function is
 . . . . .
 . . . . .

end hls;
```

RTL (VHDL, etc)

bitstream