RECHNER-
ARCHITEKTUR
INFIII

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG
FACULTY OF ENGINEERING

5. Fundamentals of FPGAs

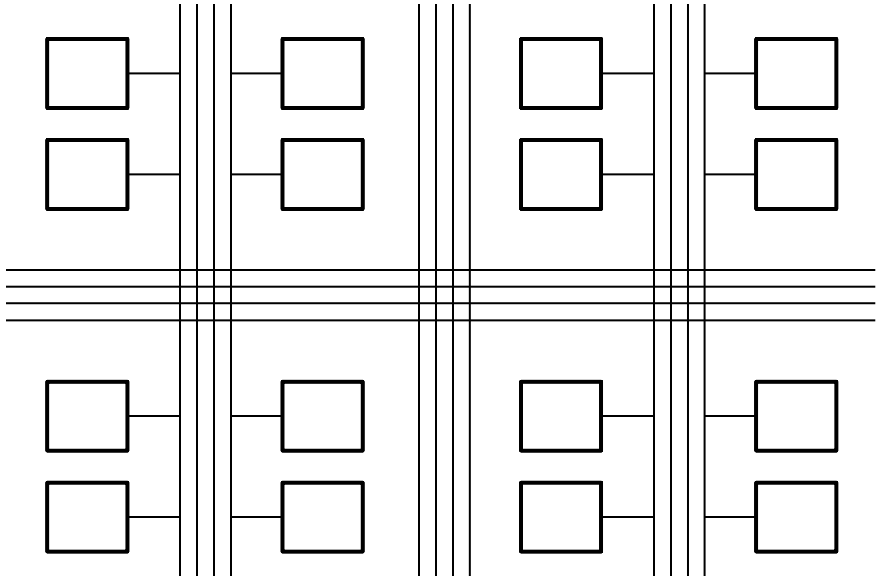
Holzinger Philipp

5.1 Basic FPGA Architecture

5.1.1 FPGA Architecture

Field-Programmable Gate Array (FPGA):

- Consists of an interconnected set of basic logic cells (“Gate Array”)



- Can be reconfigured after manufacturing of the chip (“Field Programmable”)

5.1.1 FPGA Architecture

Main differences of an FPGA to a CPU:

- Not instruction or program counter based (**dataflow architecture**)
- All functional units can work completely independently (**very high parallelism**)
- Data can be directly passed from one functional unit to the next (**local data movement**)
- No predefined, fixed data widths (**arbitrary precision**)
- Highly predictable application latency/timing (**cycle accuracy**)
- Lower clock frequencies (typically < 500 MHz)

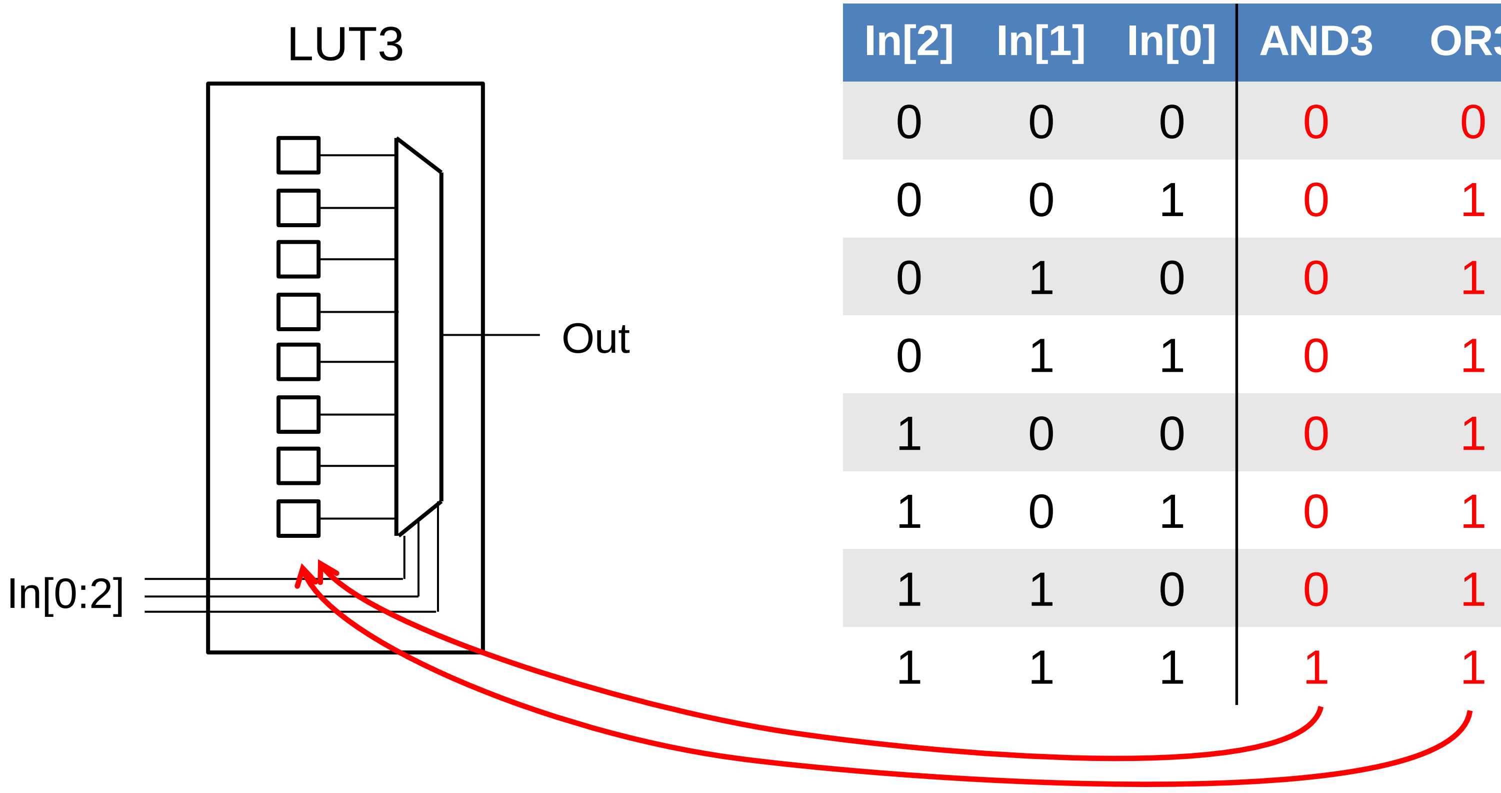
5.1.1 FPGA Architecture

Basic components of an FPGA:

- Lookup Table (**LUT**)
- Flip-Flop (**FF**)
- Block Static Random-Access Memory (**BRAM**)
- Digital Signal Processing Unit (**DSP**)
- High Speed Serial Transceiver (**GT-SERDES**)

5.1.1 FPGA Architecture

Operating principle of a LUT:

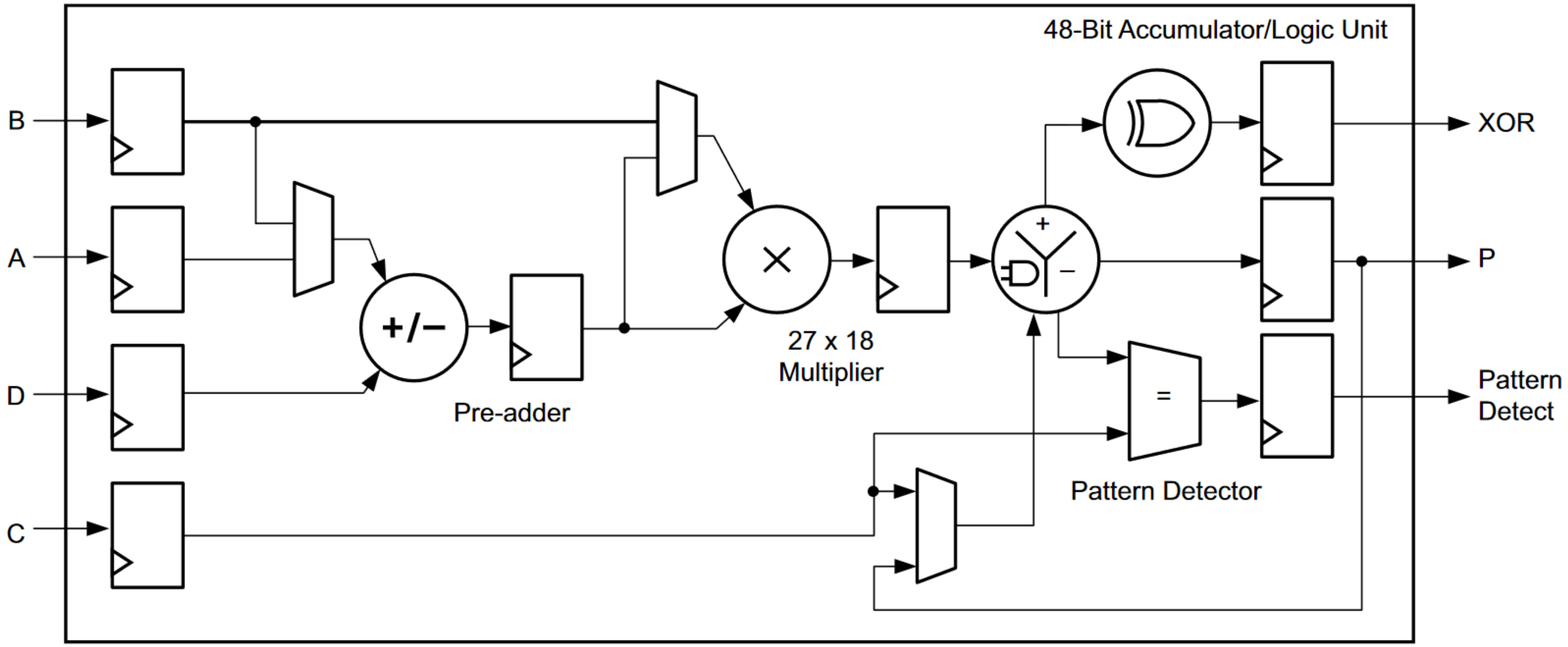


In[2]	In[1]	In[0]	AND3	OR3
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	0	1
1	0	0	0	1
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

All logic functions can be build with a cascade of LUTs!

5.1.1 FPGA Architecture

Example Xilinx DSP architecture:

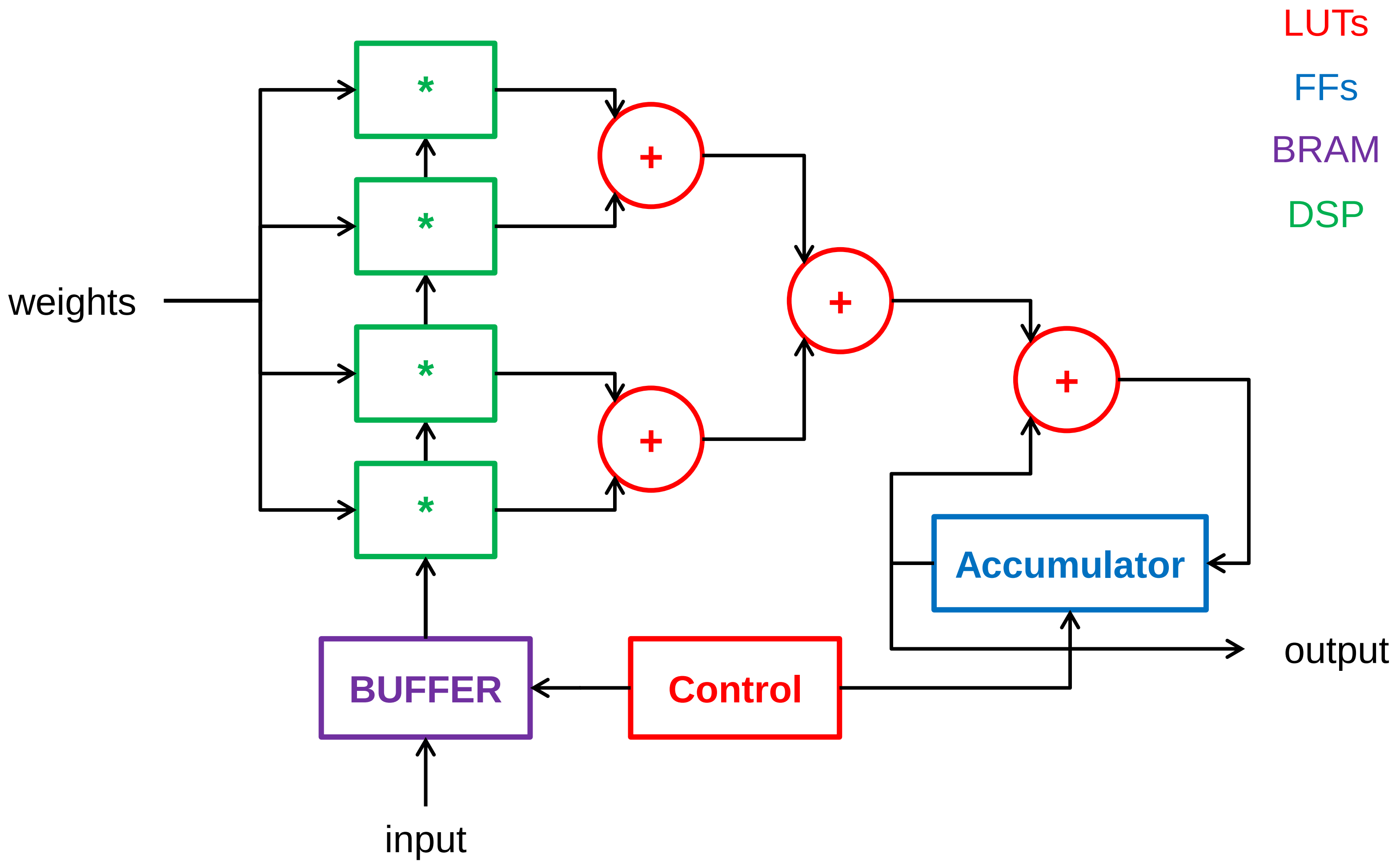


➢ Newer FPGAs also with floating point DSPs

- www.xilinx.com/support/documentation/user_guides/ug579-ultrascale-dsp.pdf

5.1.1 FPGA Architecture

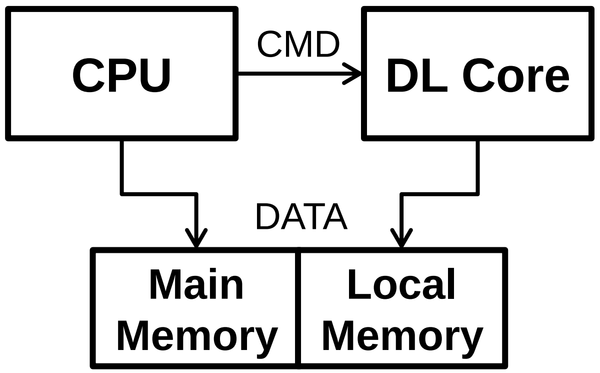
Accelerators can be composed of these basic blocks:



5.1.2 System Integration

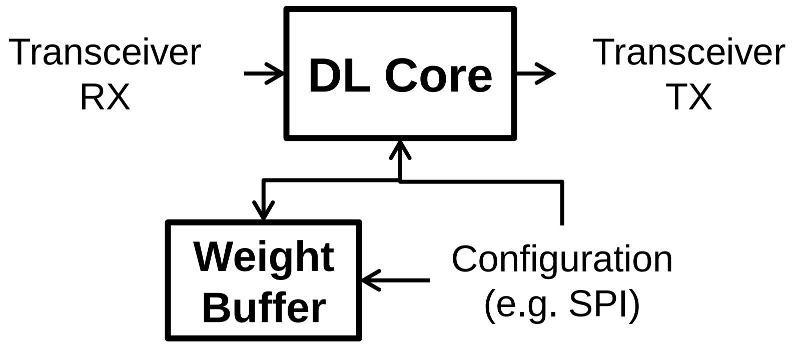
Two main approaches to use FPGAs:

Accelerator



➢ CPU offloads tasks to dedicated logic

Standalone



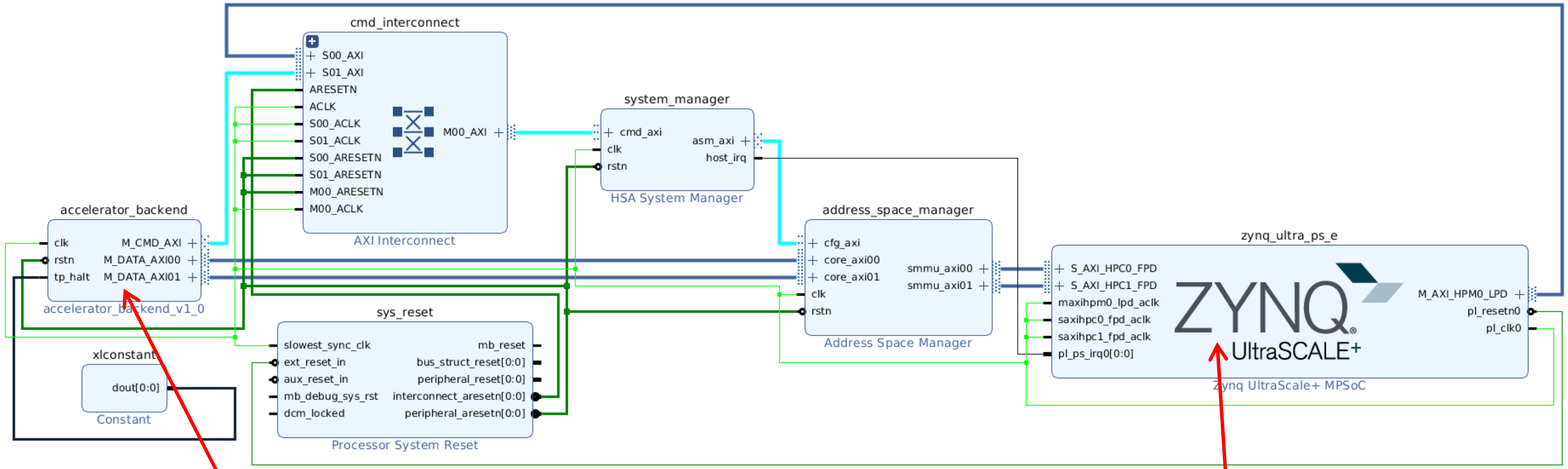
➢ Data is continuously streamed to and from the chip

The so called IP Cores are usually interconnected via standardized bus protocols like ARM AMBA AXI

5.1.2 System Integration

Xilinx Vivado block design example:

clock reset AXI4 Lite AXI4 Full



accelerator cores

Processing System with Cortex A53

5.2 FPGA Programming Model

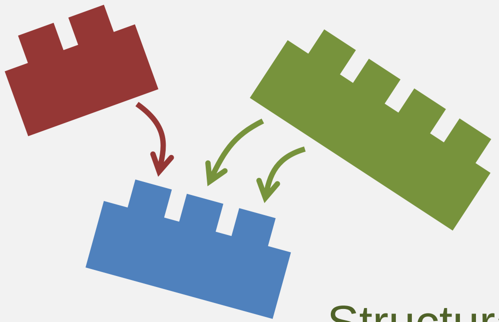
5.2.1 Hardware Description Languages (HDL)

Hardware Description Language (HDL)	**Software Programming Language**
e.g. Verilog, VHDL	e.g. C/C++, Python
describes structure and behavior of circuits	sequence of commands
fine grained control of every bit flip	limited by available operations and data types
inherently concurrent	usually procedural
explicit and intrinsic notion of relative time (clock cycle)	series of instructions without time dependency
user interaction must be explicitly modeled by e.g. USB, HDMI cores	simple user interaction at runtime with standardized I/O

5.2.1 Hardware Description Languages (HDL)

VHDL example:

```
// this the I/O definition of the module also seen by other components
entity adder is
port(
    clk: in  std_logic;
    rst: in  std_logic;
    a  : in  std_logic_vector(31 downto 0);
    b  : in  std_logic_vector(31 downto 0);
    c  : out std_logic_vector(31 downto 0)
);
end entity;
// the architecture describes the behavior of the component
architecture behav of adder is
begin
    add: process(clk)
    begin
        if(rising_edge(clk)) then
            if(rst = ‘1’) then
                c <= (others => ‘0’);
            else
                c <= std_logic_vector(signed(a) + signed(b));
            end if;
        end if;
end architecture;
```



Structural description of the hardware

Every clock cycle the value of c is updated (→ Flip-Flops)

Combinatorial evaluation within a single clock cycle (→ LUTs)

5.2.2 High-Level Synthesis (HLS)

High-Level Synthesis (HLS) refers to the act of generating an HDL description from a high-level language as C/C++:

Advantages	Disadvantages
Much faster hardware development	Less control of the resulting hardware
Better accessibility for software developers	Usually bigger and less energy efficient than HDL cores
Interoperability to other device classes as CPUs and GPUs	Very poor results for functions written completely in software style (e.g. file parser)

5.2.2 High-Level Synthesis (HLS)

Simple OpenCL Vector Add kernel:

```
// kernel function which describes the behavior of an accelerator IP core
// arguments are passed to the bus slave of the core
__kernel void vadd(__global float *a, __global float *b, __global const float *c){
    const unsigned int idx = get_global_id(0);

    // loads and stores are translated to bus master accesses
    c[idx] = a[idx] + b[idx];
}
```

General (shortened) HLS procedure:

- Extract control and data flow graphs from kernel
- Find all needed DFG operations (**Allocation**)
- Insert as many pipeline stages as needed (**Scheduling**)
- Assign operations to available hardware components (**Binding**)
- Derive hardware control FSM from CFG, scheduling, and binding

5.2.3 Synthesis Workflow

Hardware workflow:

$$\begin{tikzcd}\text{HDL Description} \arrow[d, "\text{Elaboration / Synthesis}"'] \\ \text{Netlist} \arrow[d, "\text{Place \& Route}"'] \\ \text{Configuration Bitstream}\end{tikzcd}$$

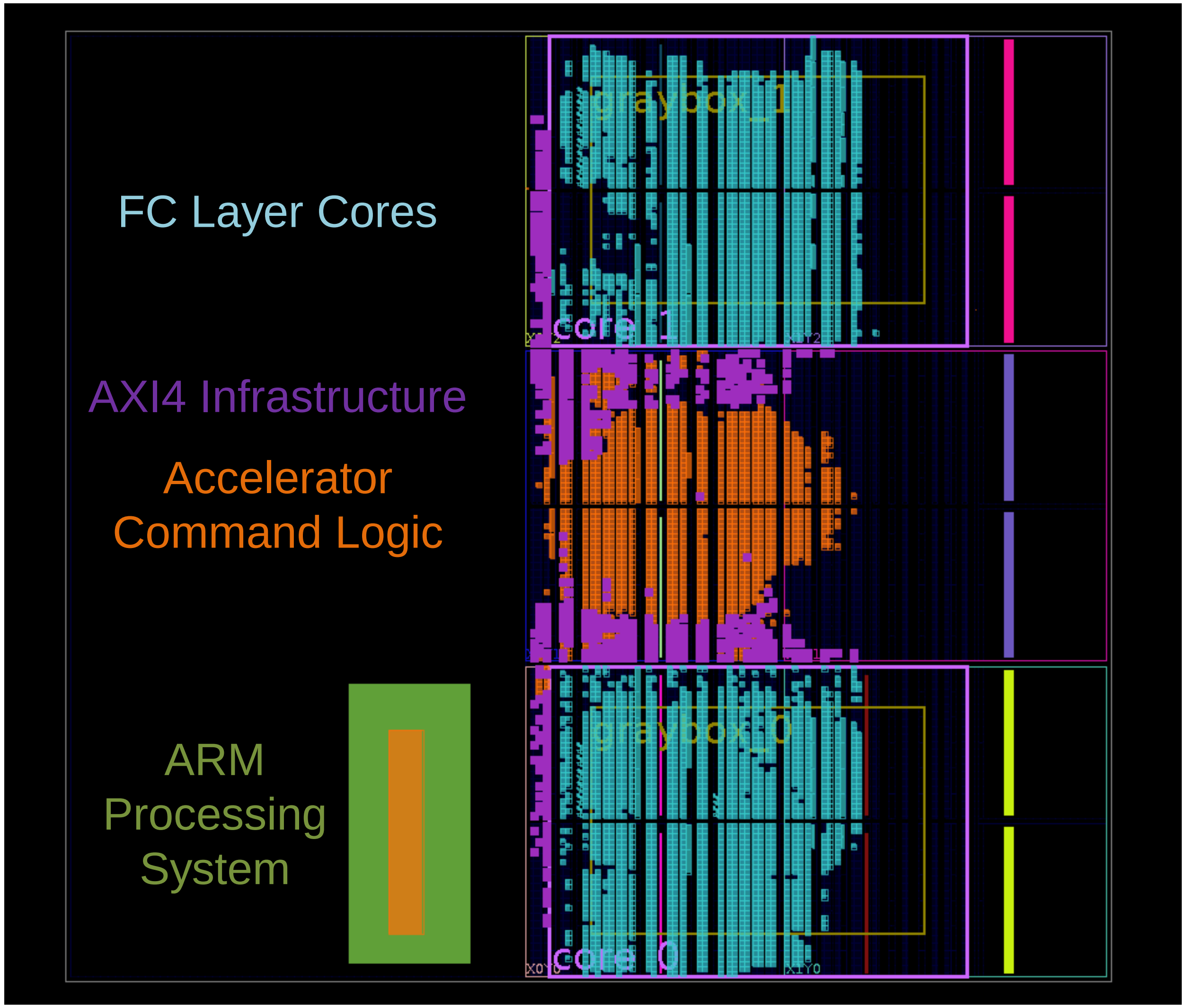
“interpretation” of the circuit description

Graph of basic component instances (e.g. LUTs, FFs) and their connections (wires)

where on the chip to meet frequency requirements

Series of bits ordered in configuration frames which describe i.e. LUT contents, route interconnection

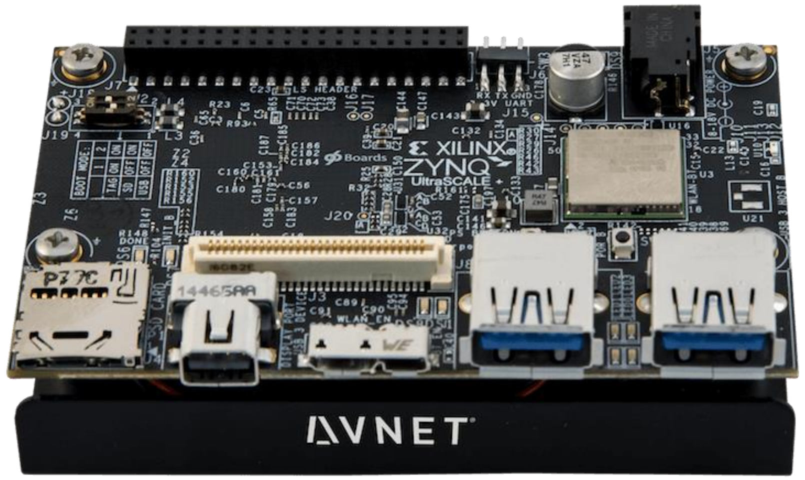
Final design floorplan (Ultra96):



5.3 Example FPGA Boards

5.3 FPGA Boards

Avnet Ultra96:



- Xilinx Zynq UltraScale+
- 4x ARM Cortex A53 Cores @ 1,2 GHz
- 71K LUTs, 141K FFs, 950KB BRAM, 360 DSPs
- 2 GB LPDDR4 (4.264 GB/s) (shared with CPU cores)

Alpha Data ADM PCIe 9H7:

- Xilinx Virtex UltraScale+ HBM
- PCIe Gen 3 x16 or PCIe Gen 4 x8, 96 Transceiver (32.75 Gb/s)
- 1304K LUTs, 2607K FFs, 42.6 MB BRAM, 9024 DSPs
- 2x 4 GB HBM2 (460 GB/s)