Introduction to Field Programmable Gate Arrays

Hannes Sakulin
CERN / EP-CMD

What is a **F**ield **P**rogrammable **G**ate **A**rray ?
.. a quick answer for the impatient

- An FPGA is an integrated circuit
  - Mostly digital electronics

- An FPGA is programmable in the in the field (=outside the factory), hence the name "field programmable"
  - Design is specified by schematics or with a hardware description language
  - Tools compute a programming file for the FPGA
  - The FPGA is configured with the design (gateware / firmware)
  - Your electronic circuit is ready to use

With an FPGA you can build electronic circuits …
… without using a bread board or soldering iron
… without plugging together NIM modules
… without having a chip produced at a factory

Outline

- Quick look at digital electronics

- Short history of programmable logic devices

- FPGAs and their features

- Programming techniques

- Design flow

- Example Applications in the Trigger and DAQ domain

Acknowledgement

- Parts of this lecture are based on material by Clive Maxfield, author of several books on FPGAs. Many thanks for his kind permission to use his material!
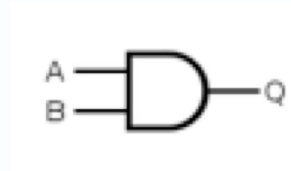
Re-use

- Re-use of the material is permitted only with the written authorization of both Hannes Sakulin (Hannes.Sakulin@cern.ch) and Clive Maxfield.

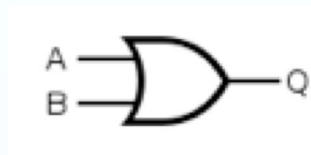# Digital electronics

The building blocks: logic gates

Truth table

C equivalent

AND gate

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

q = a && b;

OR gate

INPUT		OUTPUT
A	B	A + B
0	0	0
0	1	1
1	0	1
1	1	1

q = a || b;

Exclusive OR gate
XOR gate

INPUT		OUTPUT
A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

q = a != b;

:

Combinatorial logic (asynchronous)
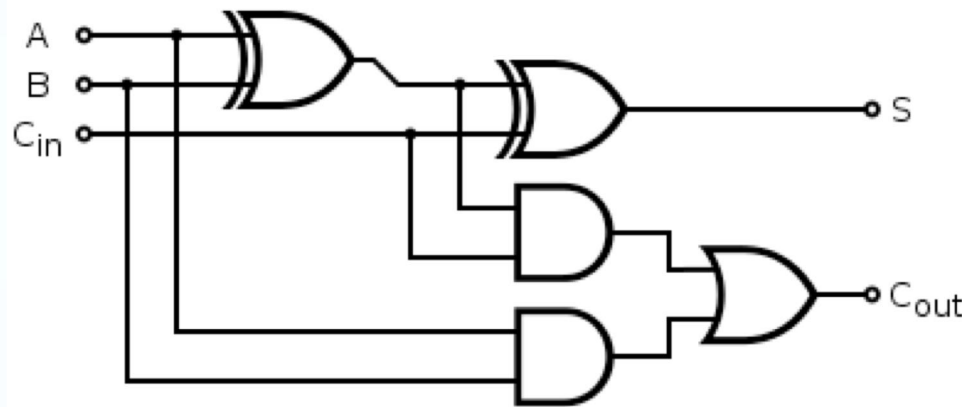
Example: Full adder with carry-in, carry-out

Outputs are determined by Inputs, only

A	B	C_{in}	S	C_{out}
0	0	0	0	0
1	0	0	1	0
0	1	0	1	0
1	1	0	0	1
0	0	1	1	0
1	0	1	0	1
0	1	1	0	1
1	1	1	1	1

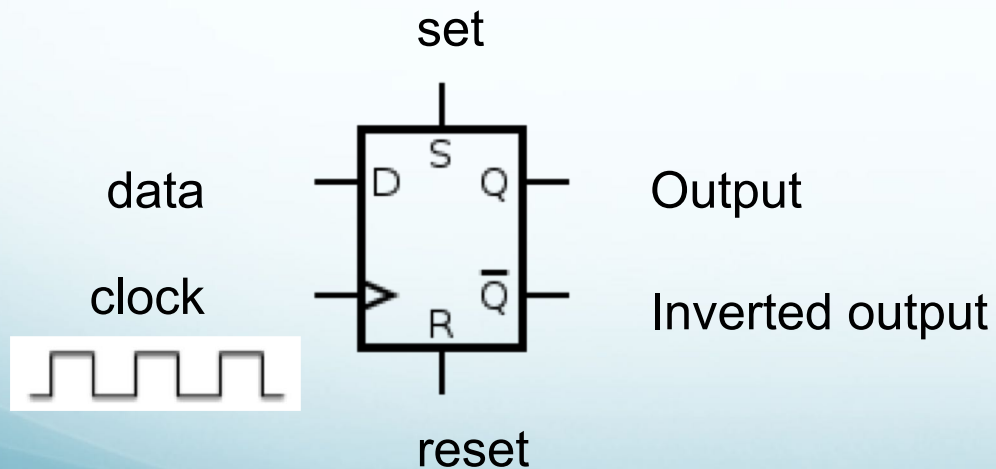Combinatorial logic may be implemented using Look-Up Tables (LUTs)

LUT = small memory

(Synchronous) sequential logic

2-bit binary counter
https://www.zeepedia.com/read.php?b=9&c=32&d_flip-flop_based_implementation_digital_logic_design
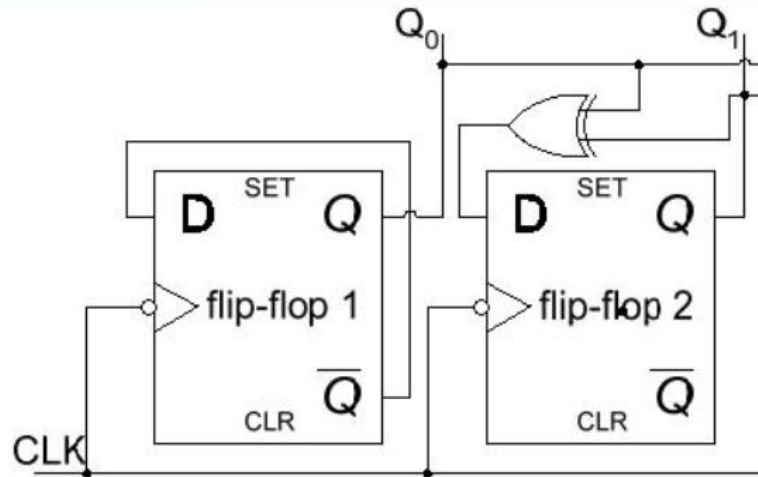
Outputs are determined
by Inputs and their History
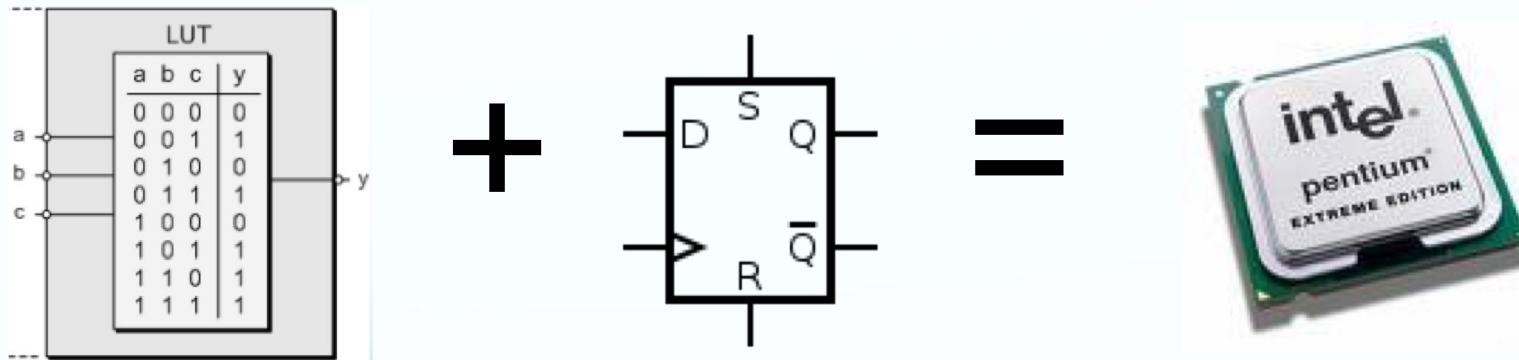(Sequence)
The logic has an internal state

D Flip-flop:
samples the data at the rising
(or falling) edge of the clock

The output will be equal to
the last sampled input until the
next rising (or falling) clock edge

D Flip-flop (D=data, delay)
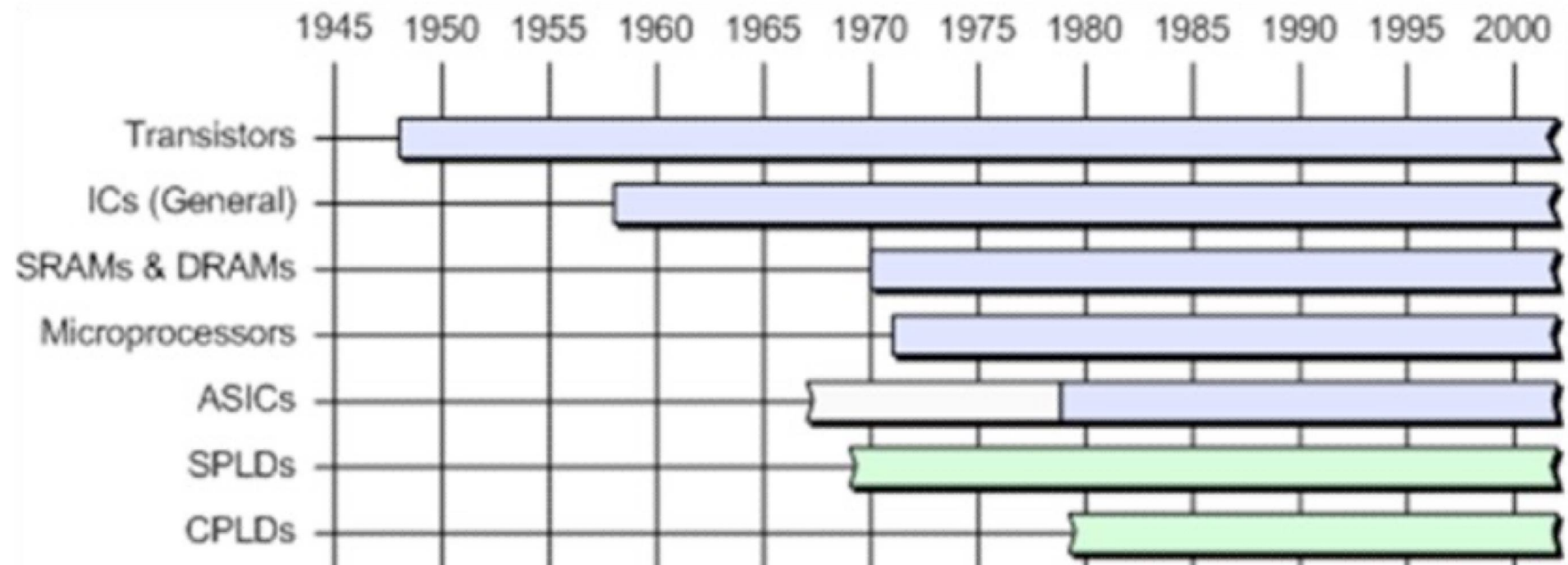
Synchronous sequential logic

Using Look-Up-Tables and Flip-Flops
any kind of digital electronics may be implemented

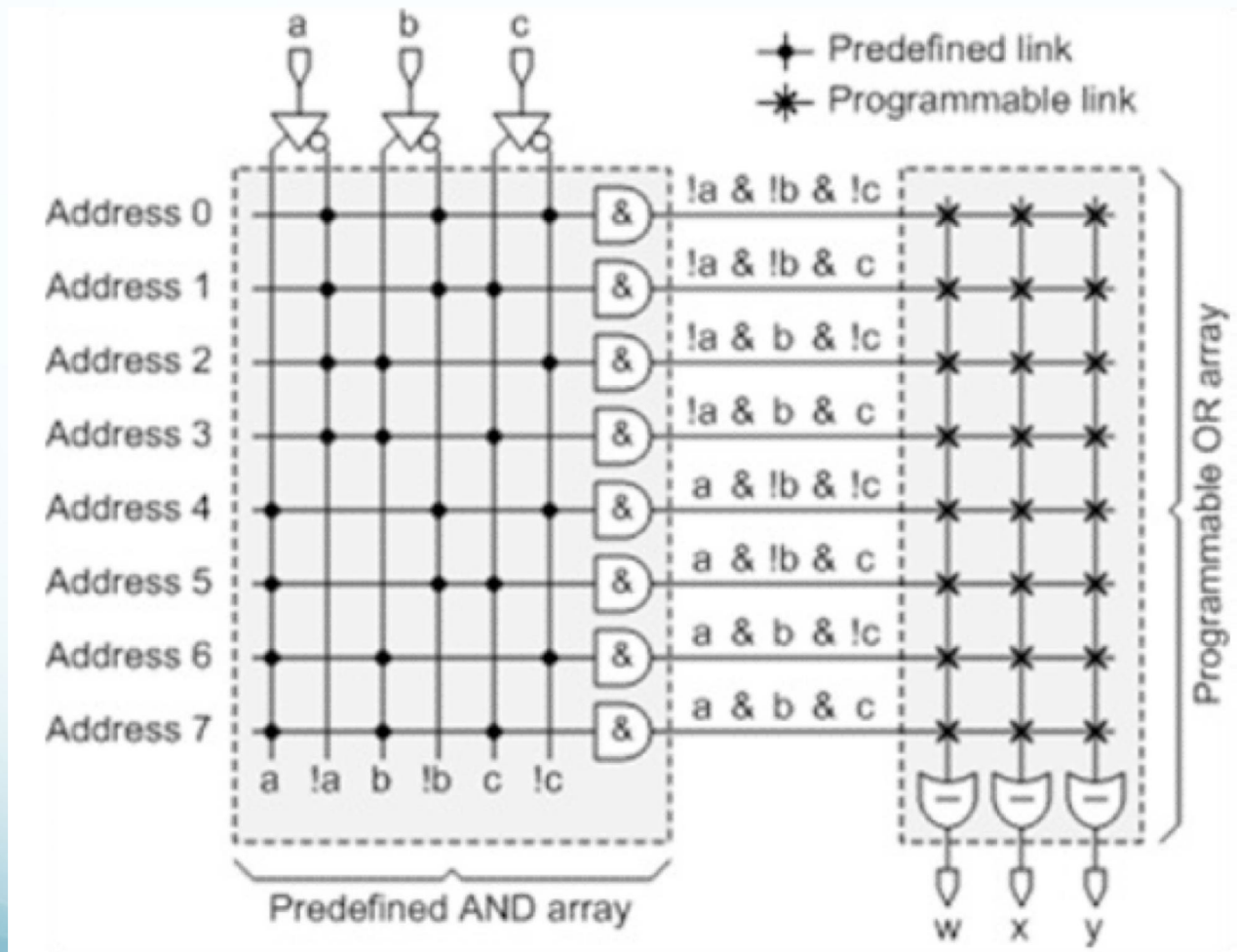Of course there are some details
to be learnt about electronics design …

Programmable digital electronics

Long long time ago …

Simple Programmable Logic Devices (sPLDs)
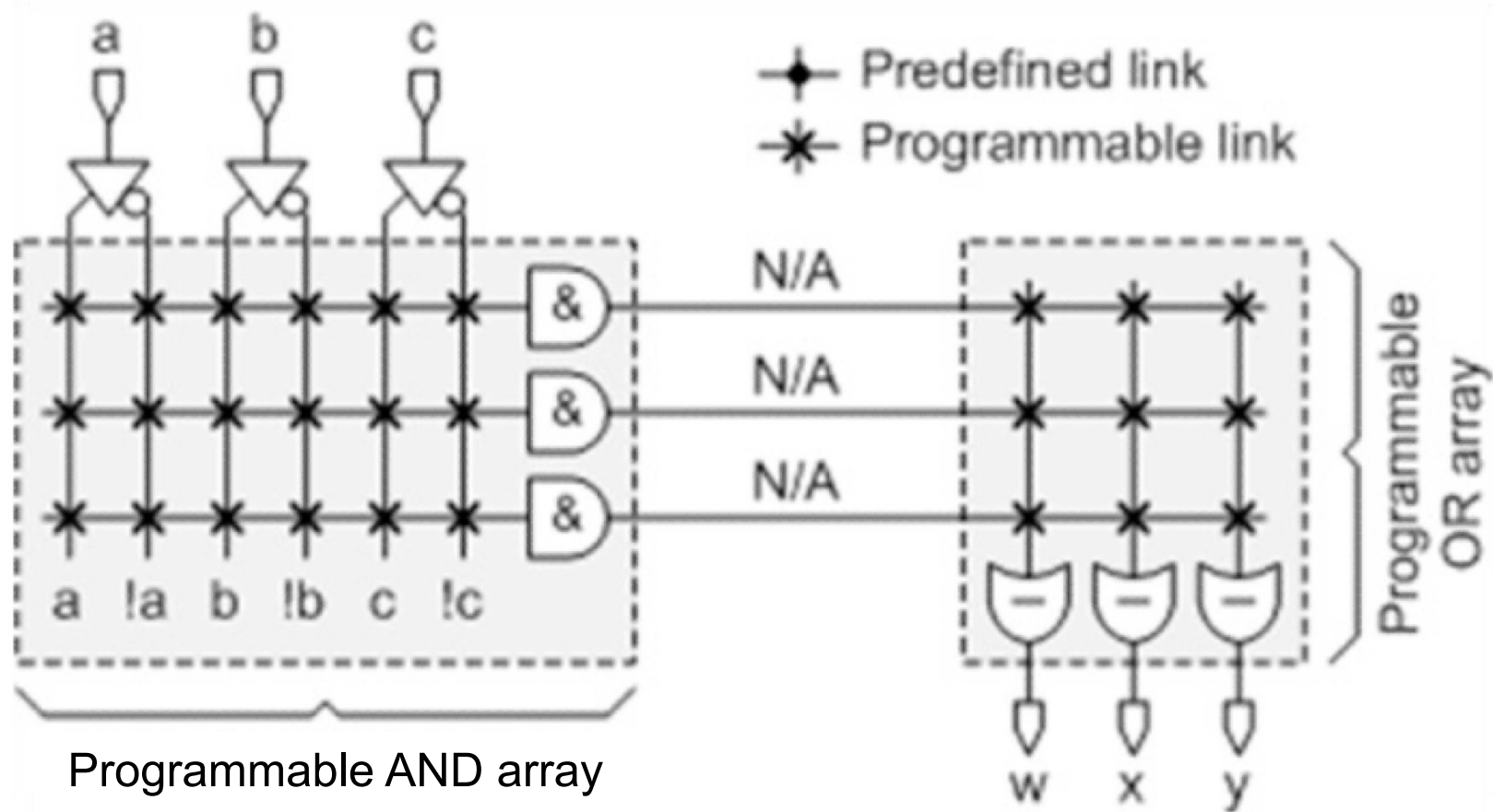## a) Programmable Read Only Memory (PROMs)

Late 60's

Unprogrammed PROM (Fixed AND Array, Programmable OR Array)

Simple Programmable Logic Devices (sPLDs)
b) Programmable Logic Arrays (PLAs)
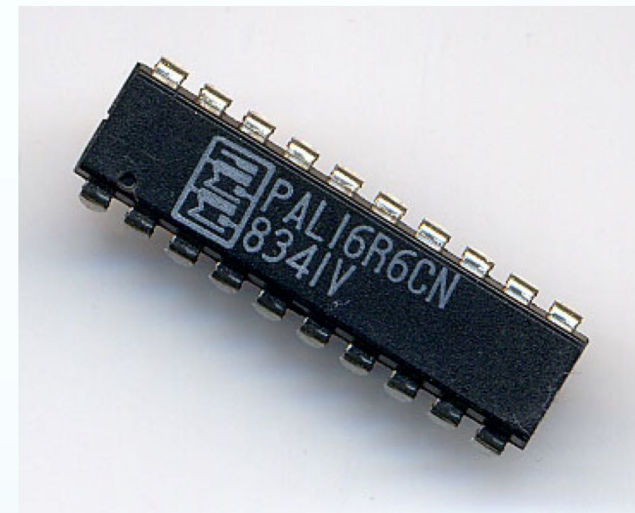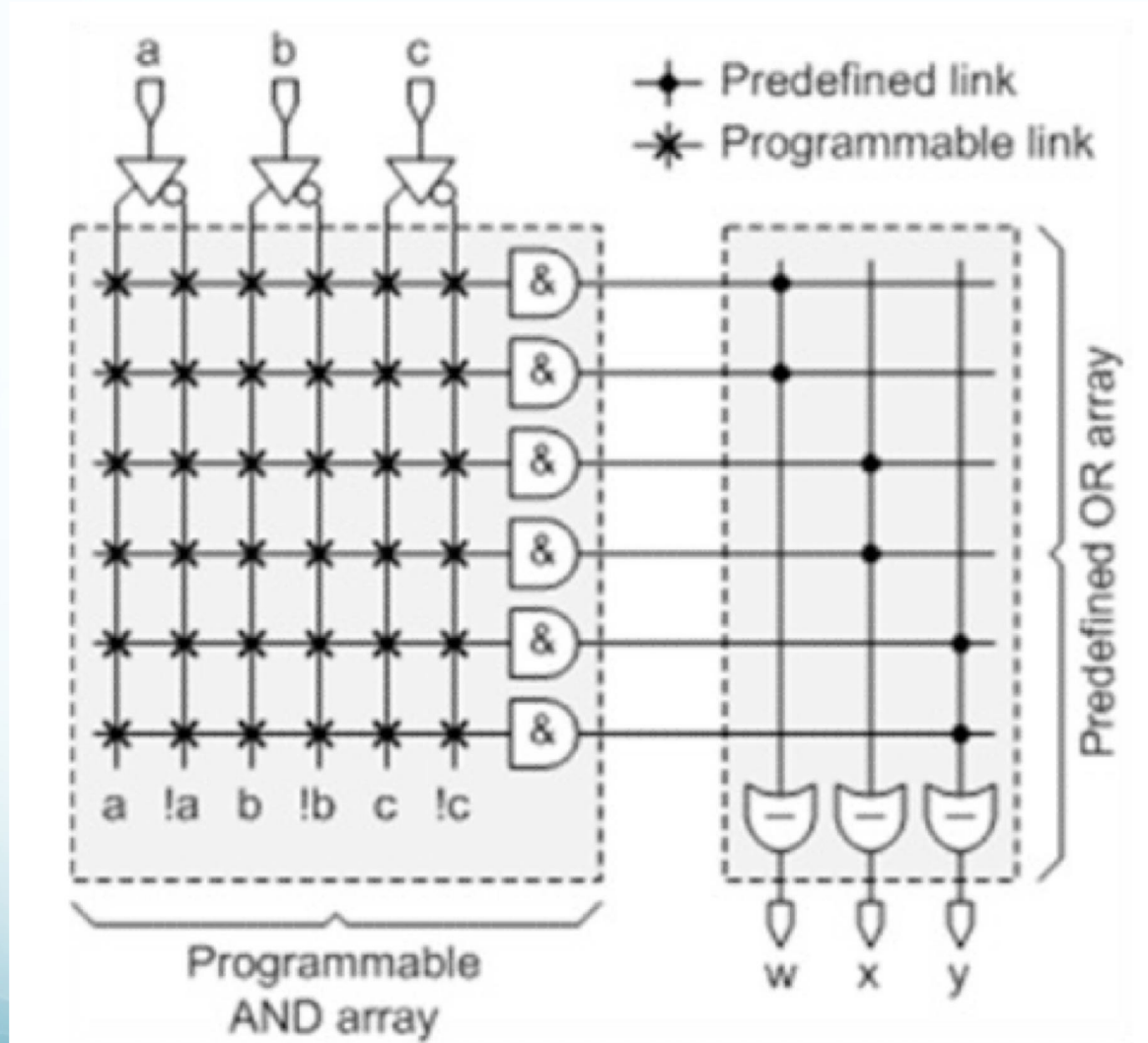

Programmable AND array

Unprogrammed PLA (Programmable AND and OR Arrays)
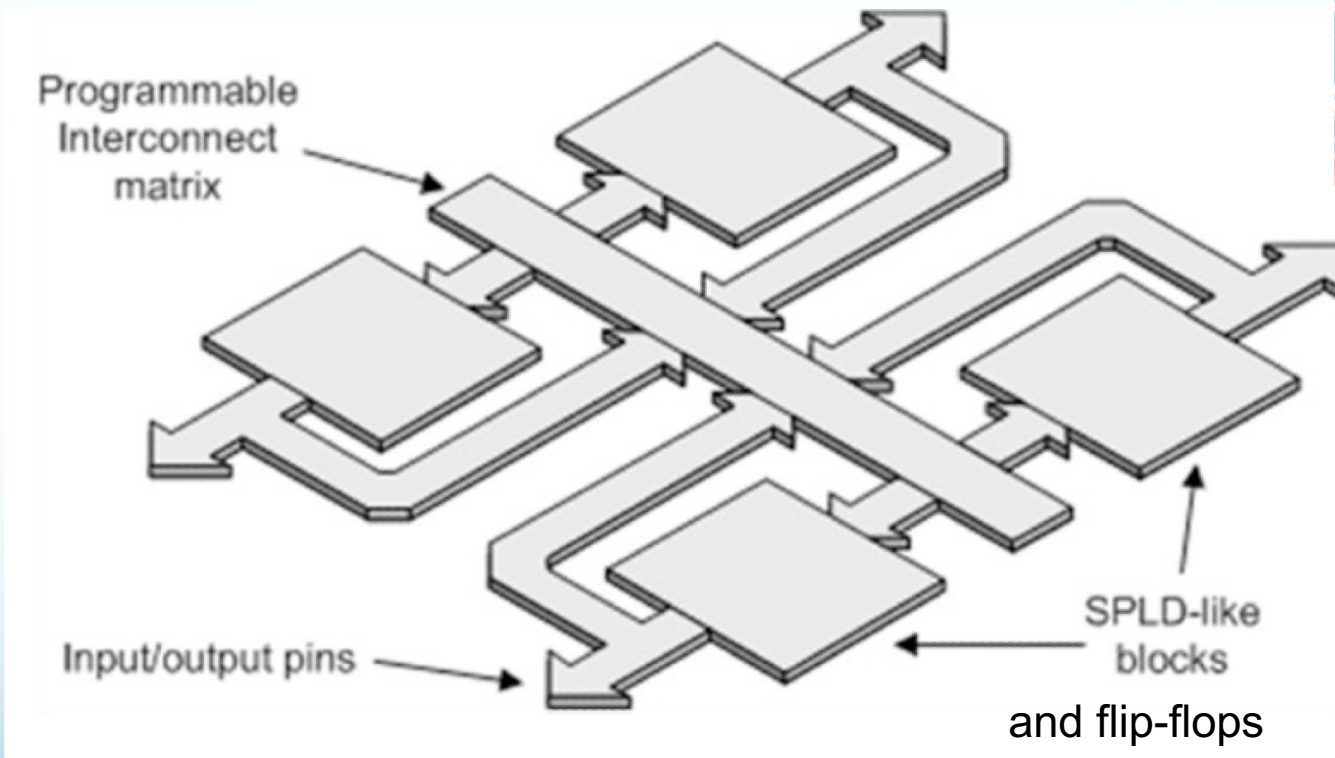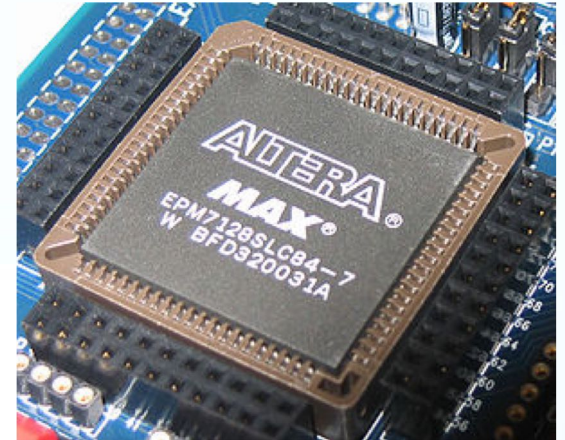
Most flexible
but slower

1975

Simple Programmable Logic Devices (sPLDs)
c) Programmable Array Logic (PAL)
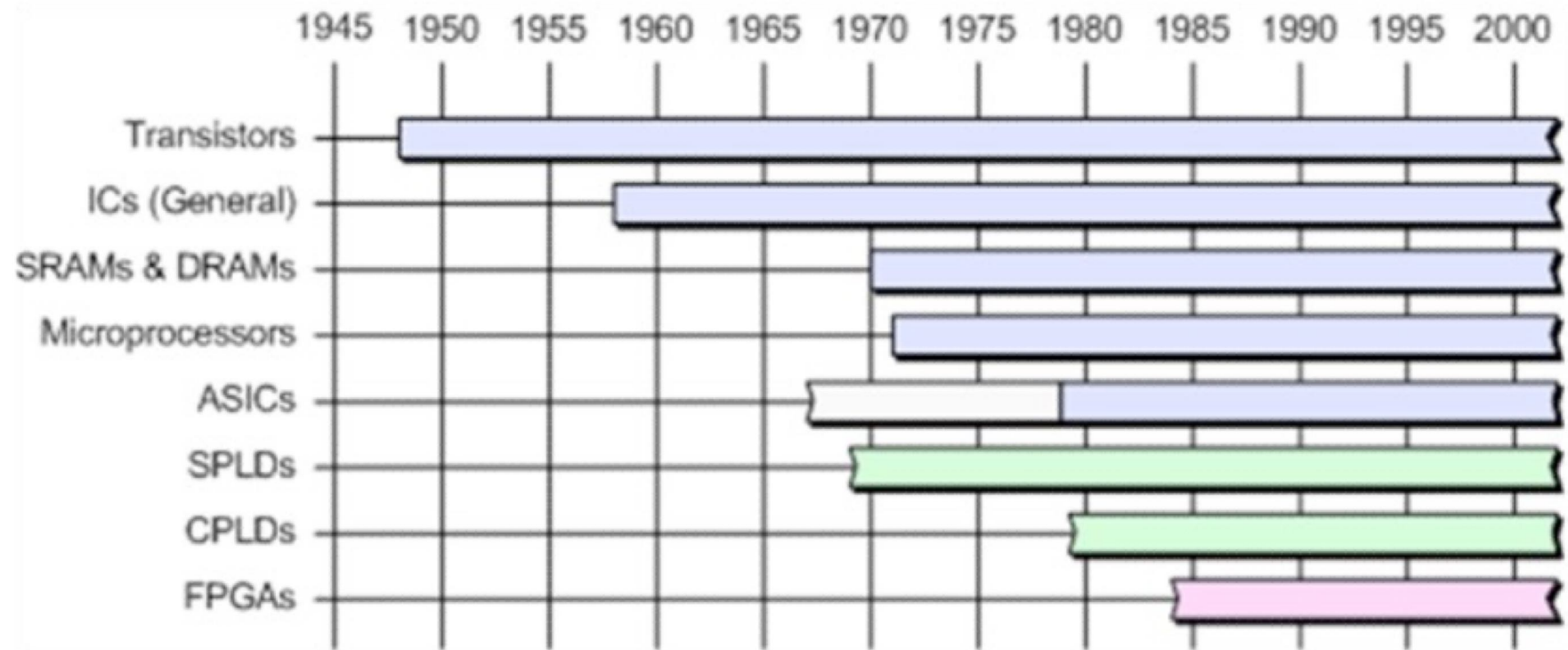

Unprogrammed PAL (Programmable AND Array, Fixed OR Array)
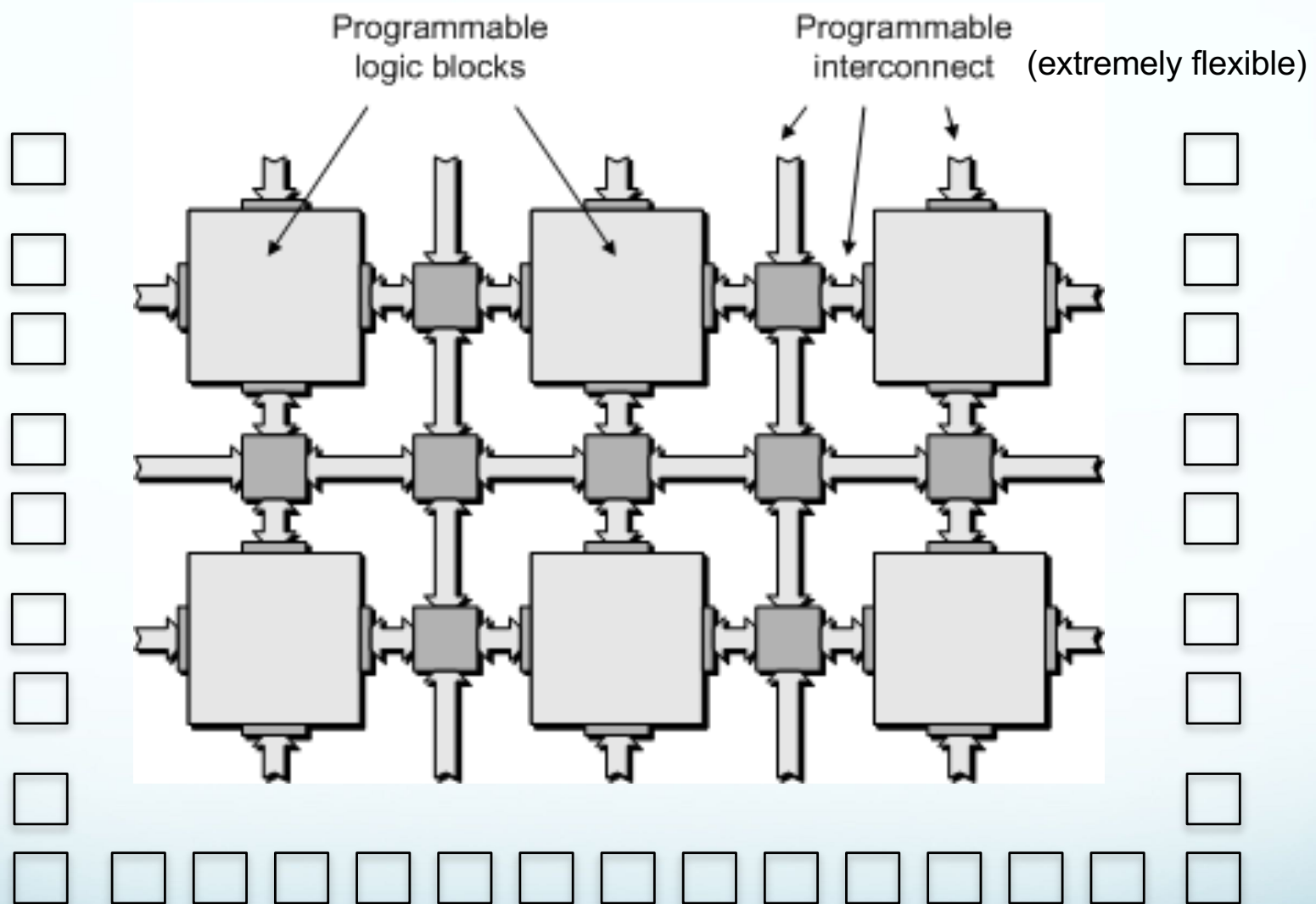
Complex PLDs (CPLDs)

Programmable Interconnect matrix

Input/output pins

SPLD-like blocks

and flip-flops

Coarse grained
100's of blocks, restrictive structure
(EE)PROM based

FPGAs ...

FPGAs

Programmable logic blocks

Programmable interconnect (extremely flexible)

Programmable Input / Output pins
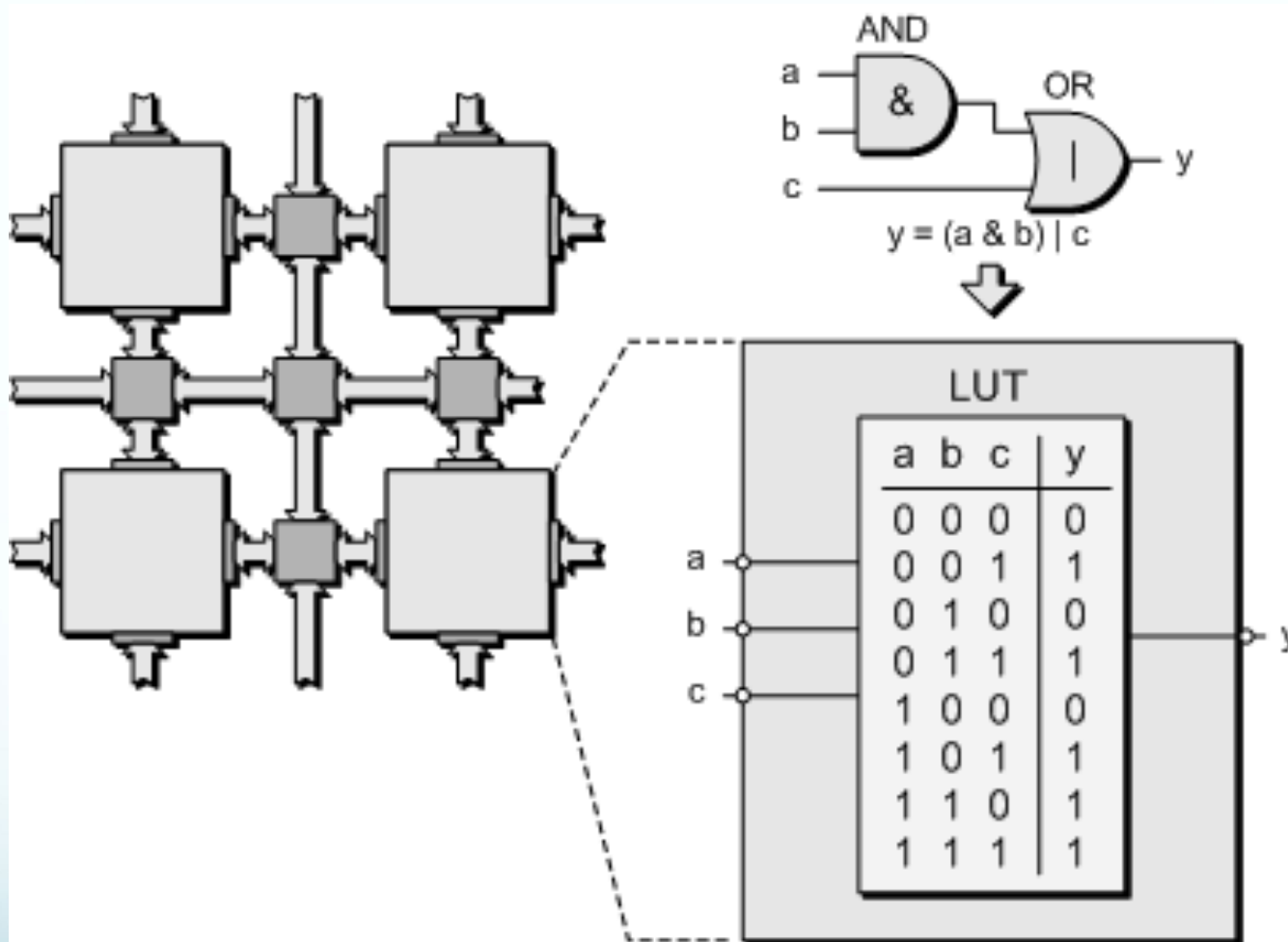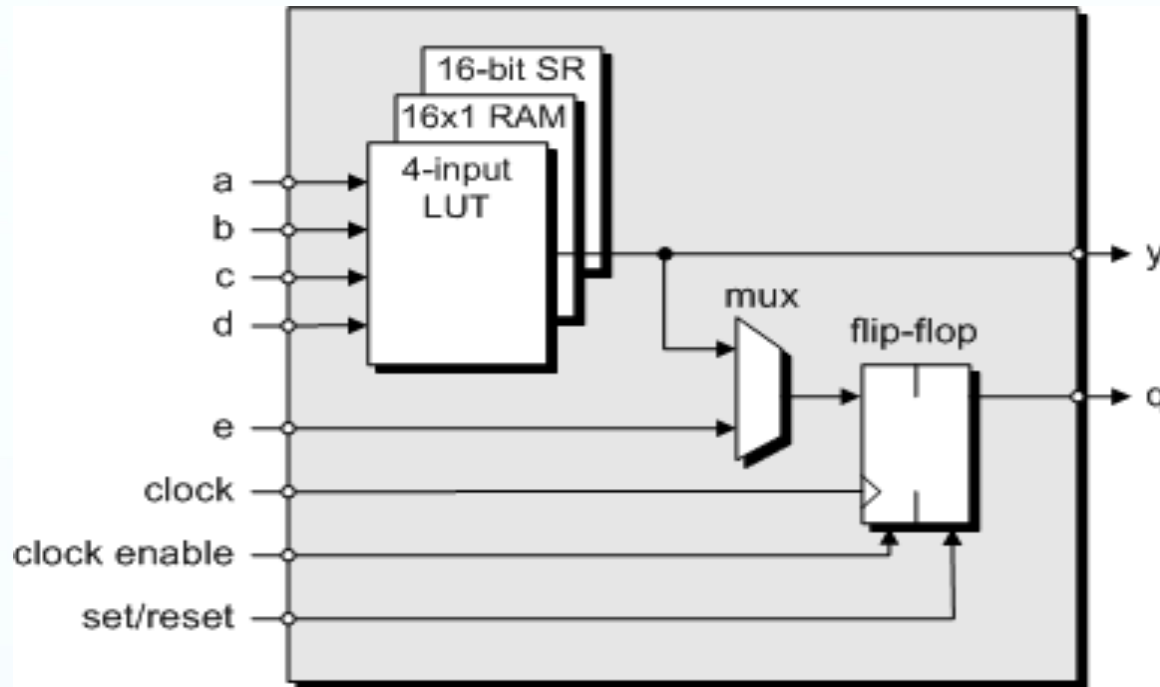
Fine-grained: 100.000's of blocks
today: up to 5 million logic blocks

LUT-based Fabrics

Typical LUT-based Logic Cell
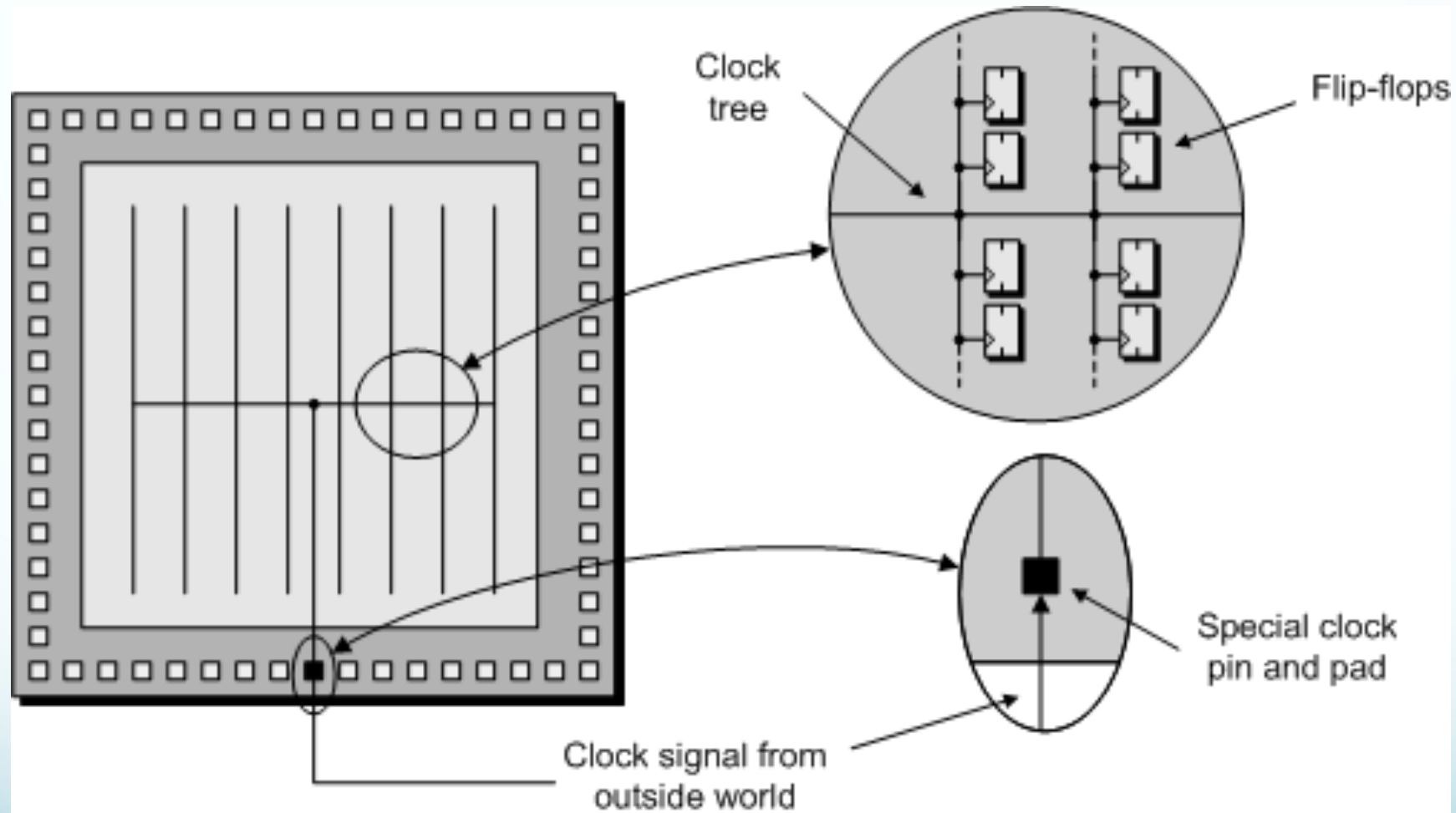

Xilinx: logic cell,
Altera: logic element

- LUT may implement any function of the inputs
- Flip-Flop registers the LUT output
- May use only the LUT or only the Flip-flop
- LUT may alternatively be configured a shift register
- Additional elements (not shown): fast carry logic

Clock Trees

Clock trees guarantee that the clock arrives at the same time at all flip-flops

Clock Managers

Embedded RAM blocks

Columns of embedded RAM blocks
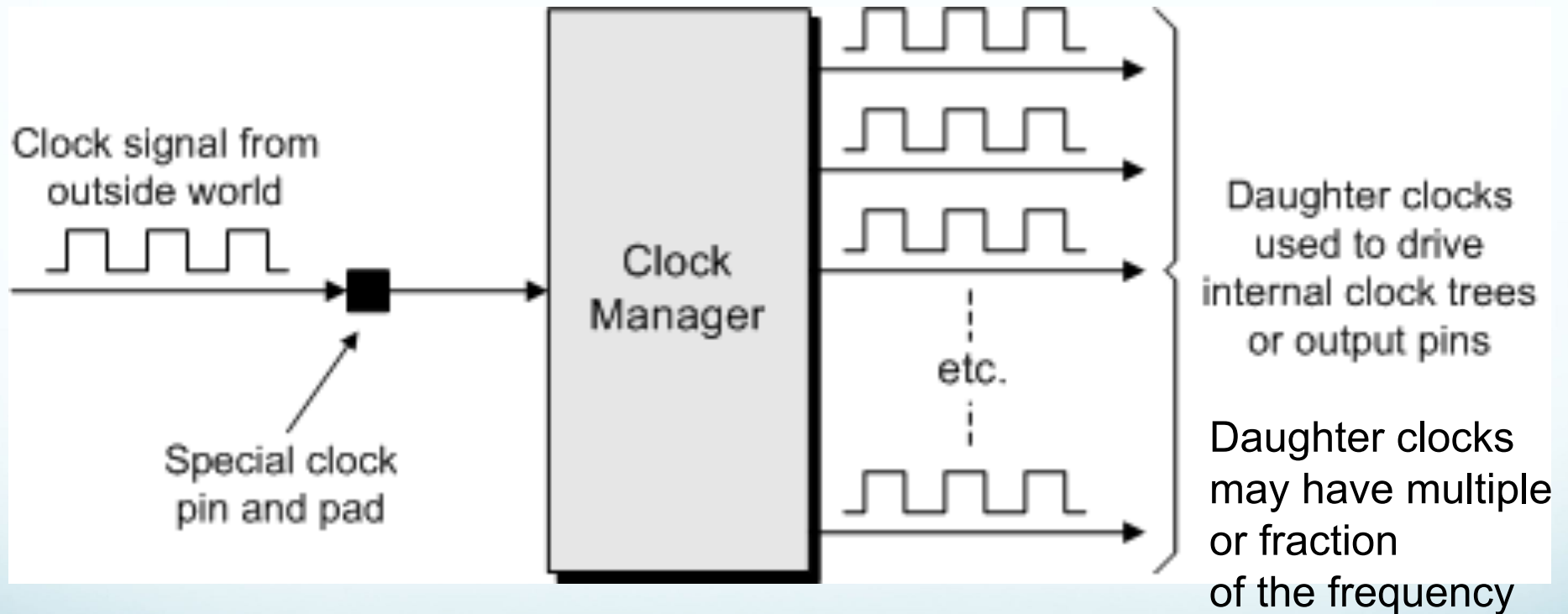
Arrays of programmable logic blocks

Today: Up to ~500 Mbit of RAM

Embedded Multipliers & DSPs

Digital Signal Processor (DSP)

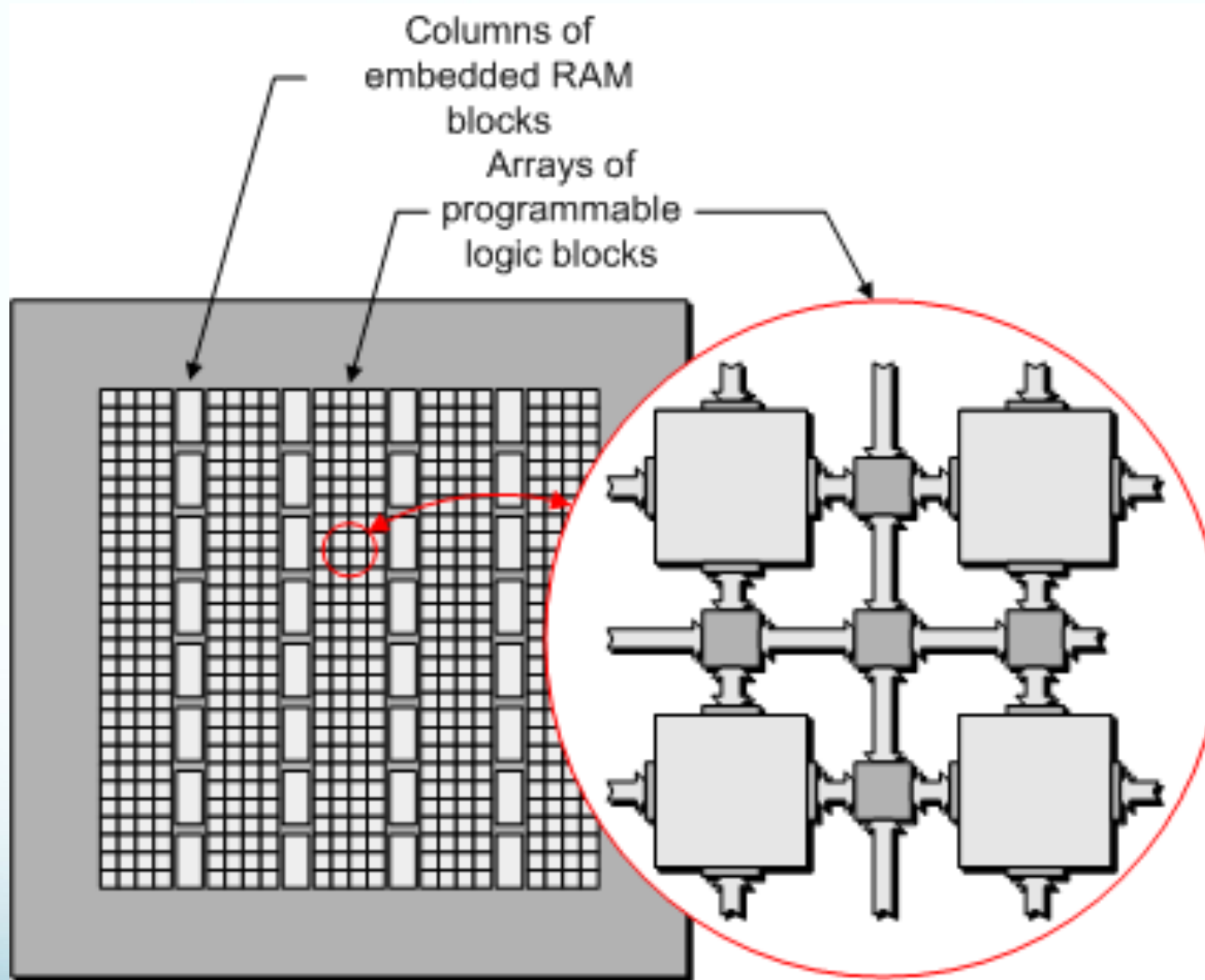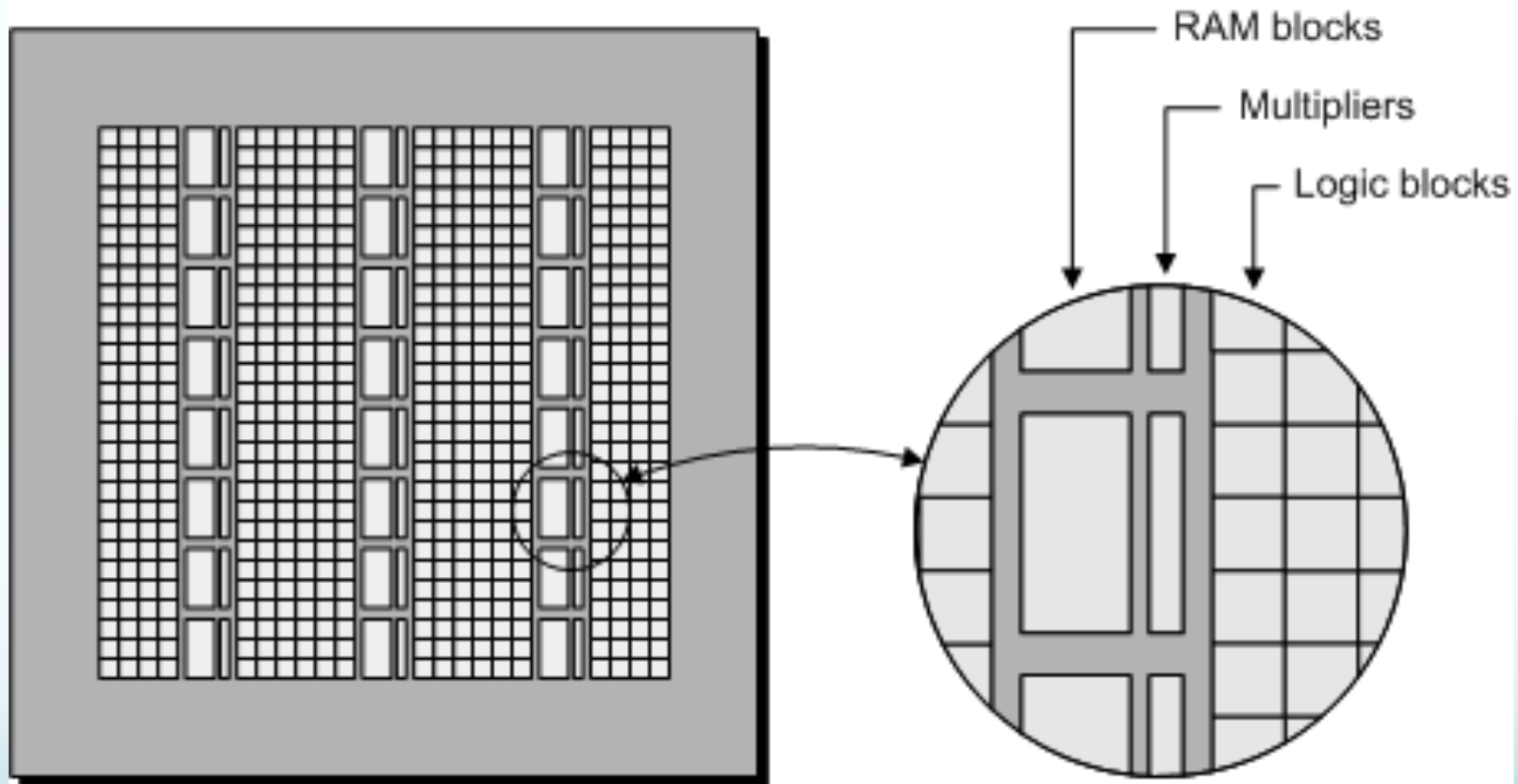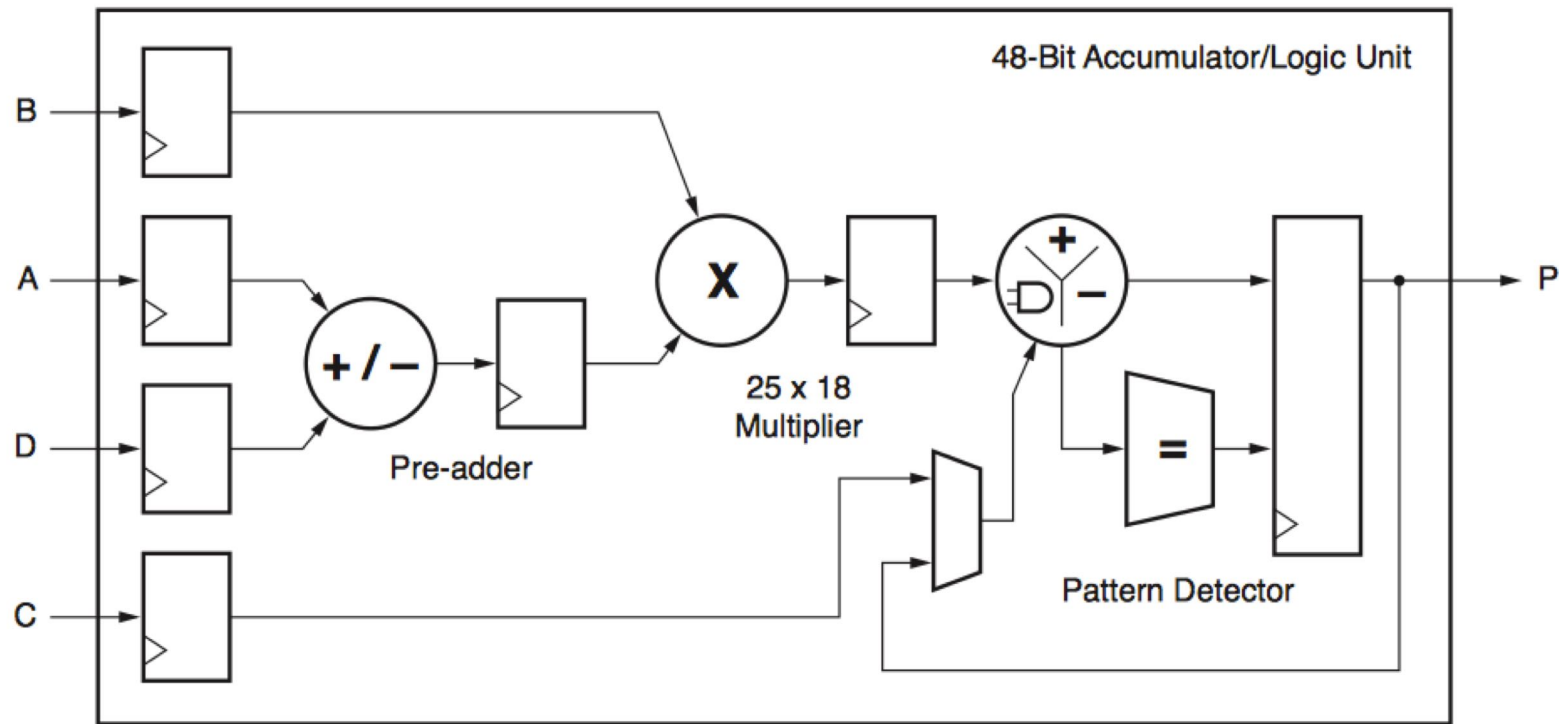DSP block (Xilinx 7-series)
Up to several 1000 per chip

Soft and Hard Processor Cores

- Soft core
  - Design implemented with the programmable resources (logic cells) in the chip

- Hard core
  - Processor core that is available in addition to the programmable resources
  - E.g.: Power PC, ARM
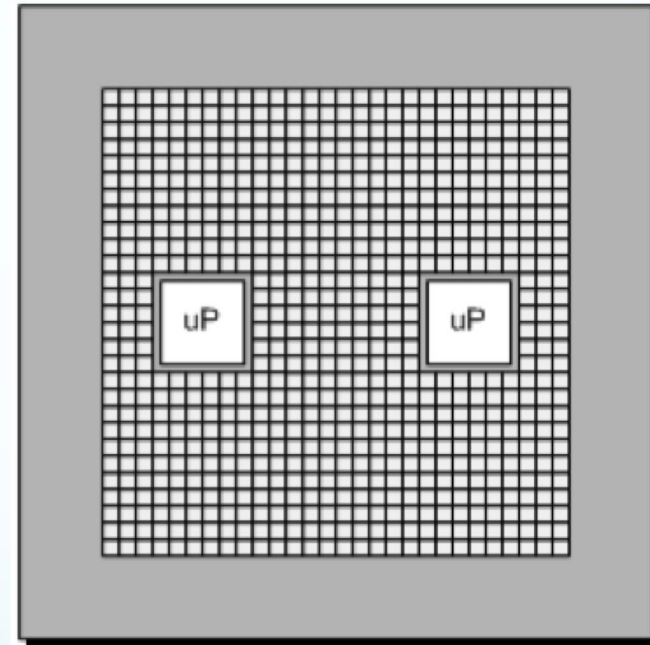
General-Purpose Input/Output (GPIO)

General-purpose I/O banks 0 through 7
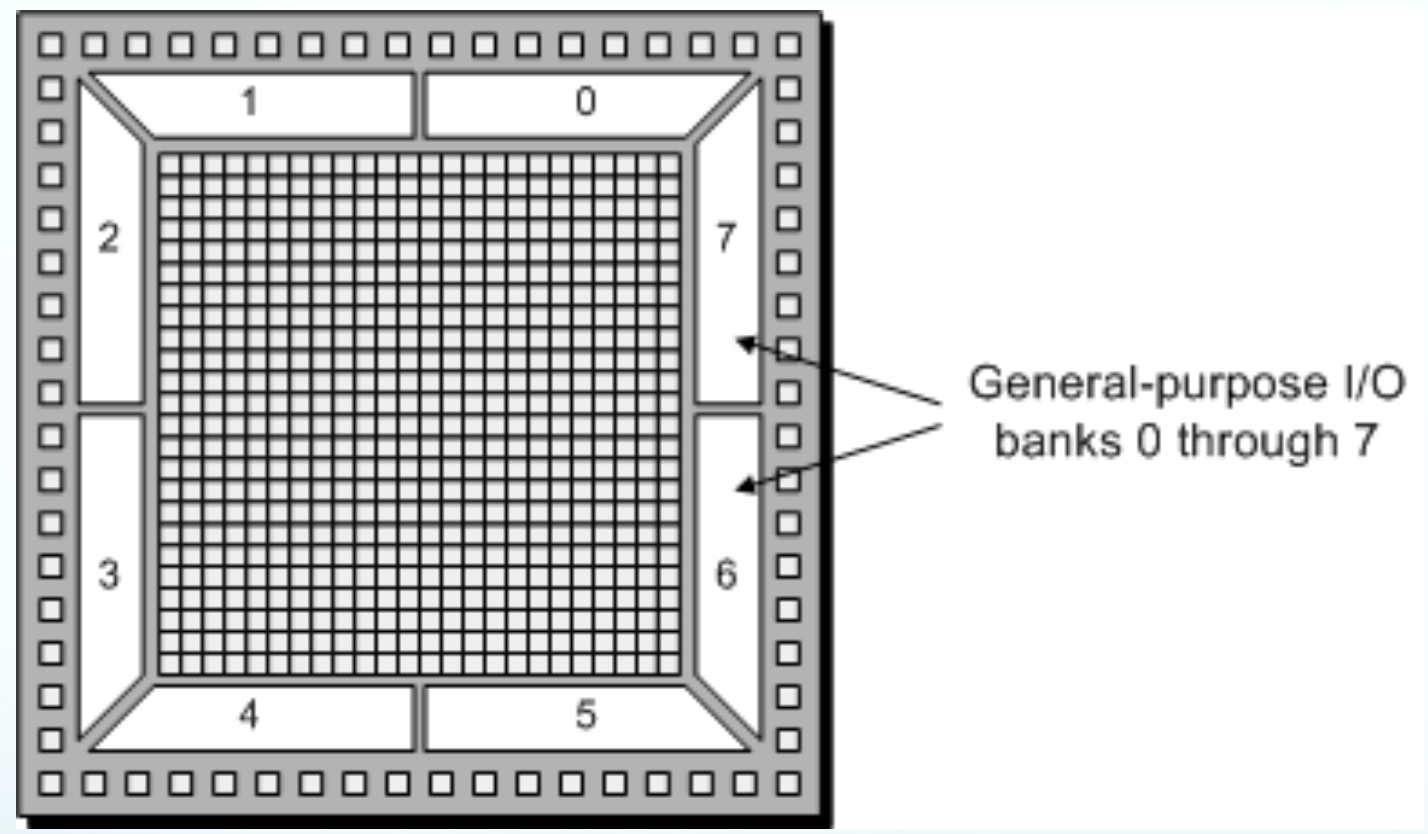
Today: Up to 1200 user I/O pins
Input and / or output
Voltages from (1.0), 1.2 .. 3.3 V
Many IO standards
Single-ended: LVTTL, LVCMOS, …
Differential pairs: LVDS, …

High-Speed Serial Interconnect

- Using differential pairs

- Standard I/O pins limited to about 1 Gbit/s

- Latest serial transceivers: typically 10 Gb/s, 13.1 Gb/s,
  - up to 32.75 Gb/s
  - up to 56 Gb/s with Pulse Amplitude Modulation (PAM)

- FPGAs with multi-Tbit/s IO bandwidth
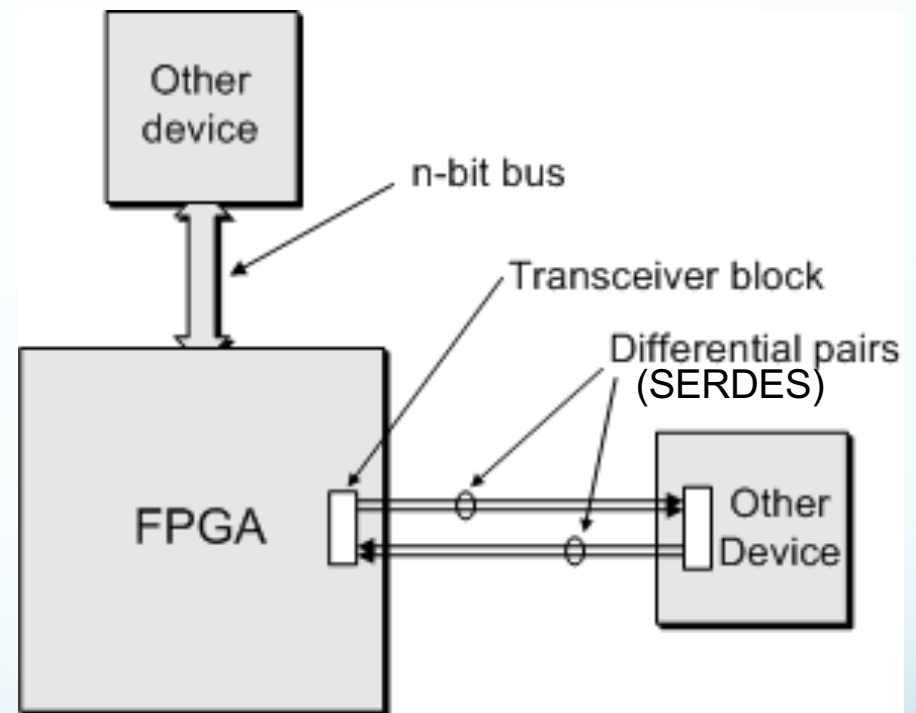
Components in a modern FPGA

- ☐ Logic block
- ▮ SRAM block
- ▯ Multiplier
- ▮ DSP block
- ▮ PLL
- ▯ Clock Manager
- ▯ I/O Bank
- ▮ SERDES block
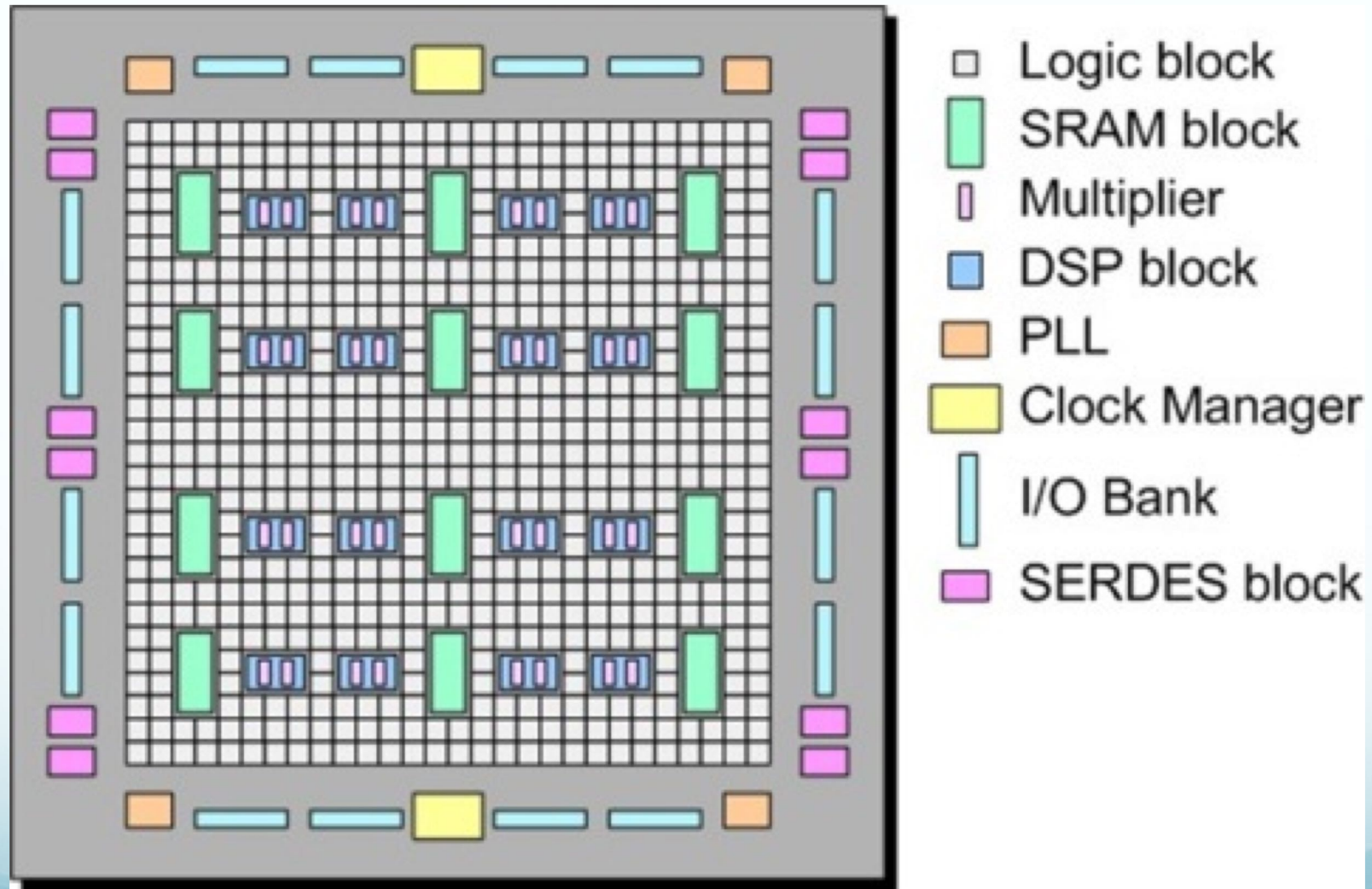
Programming techniques

Fusible Links (not used in FPGAs)

Antifuse Technology

Unprogrammed link

Programmed link

Amorphous silicon column

Metal

Oxide

Metal

Substrate

Polysilicon via

(a) Before programming

(b) After programming
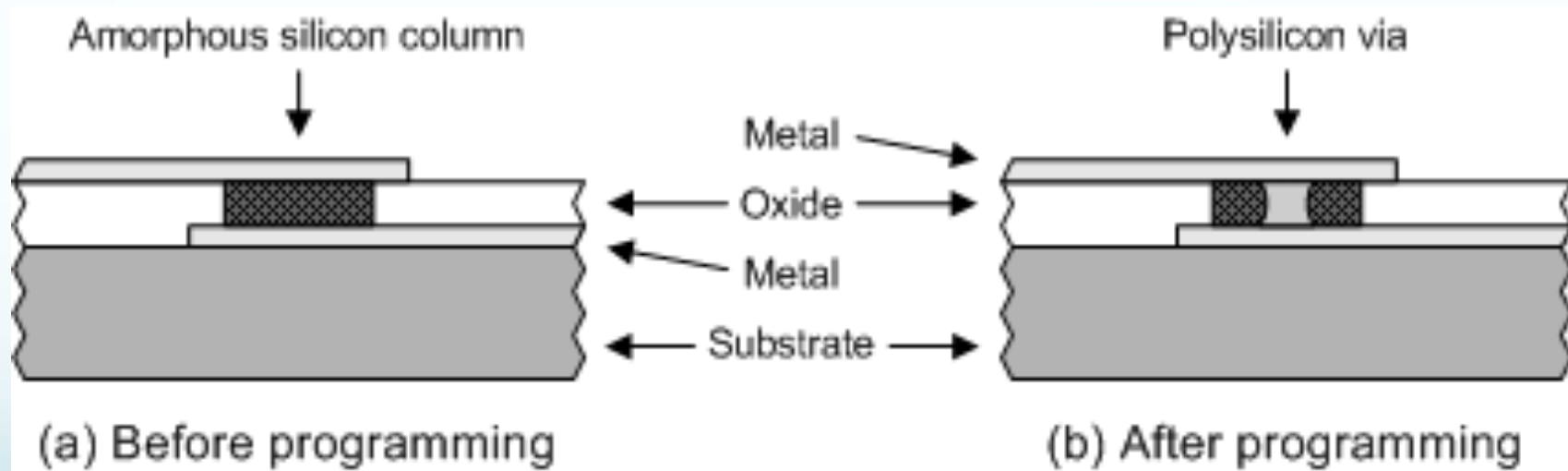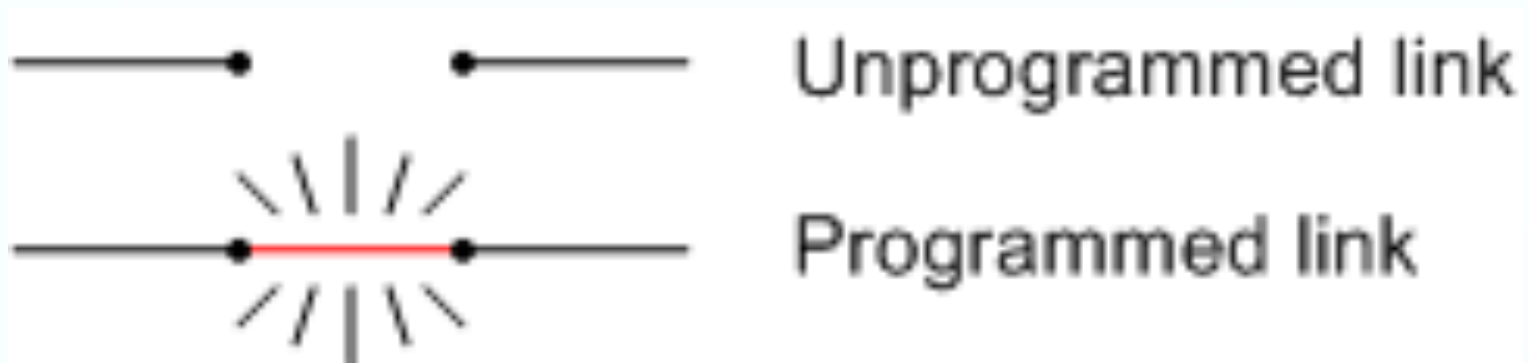
EPROM Technology

Erasable Programmable Read Only Memory

(a) Standard MOS transistor
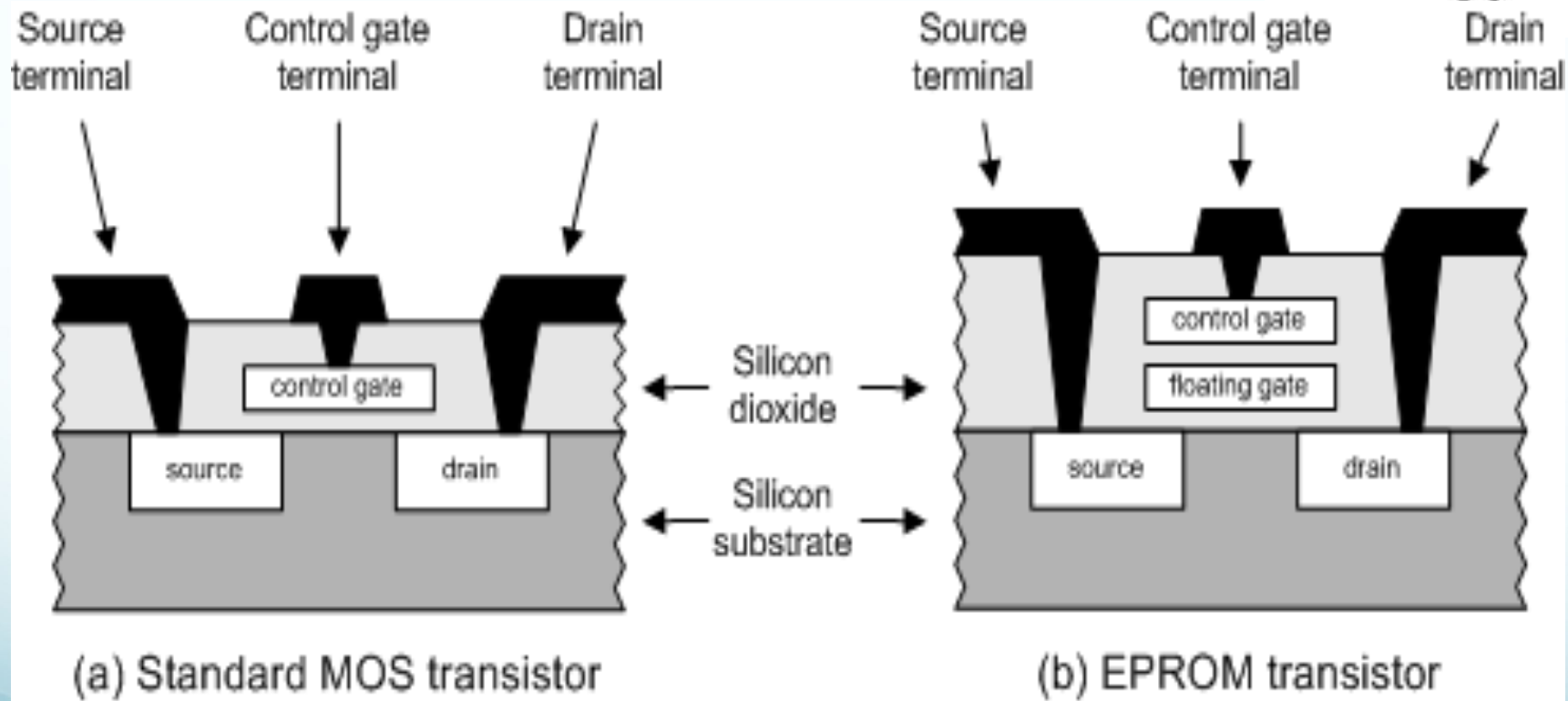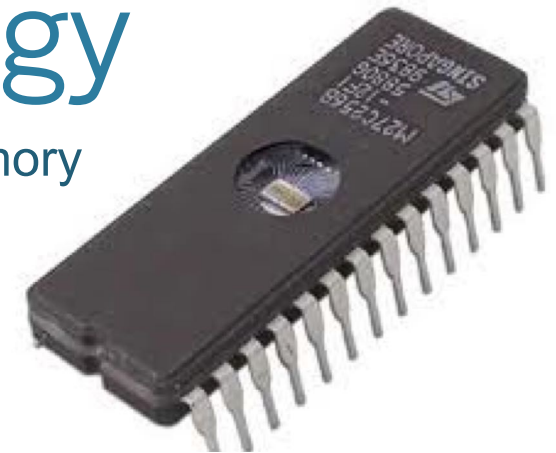
(b) EPROM transistor

Intel, 1971

EEPROM and FLASH Technology

Electrically Erasable Programmable Read Only Memory
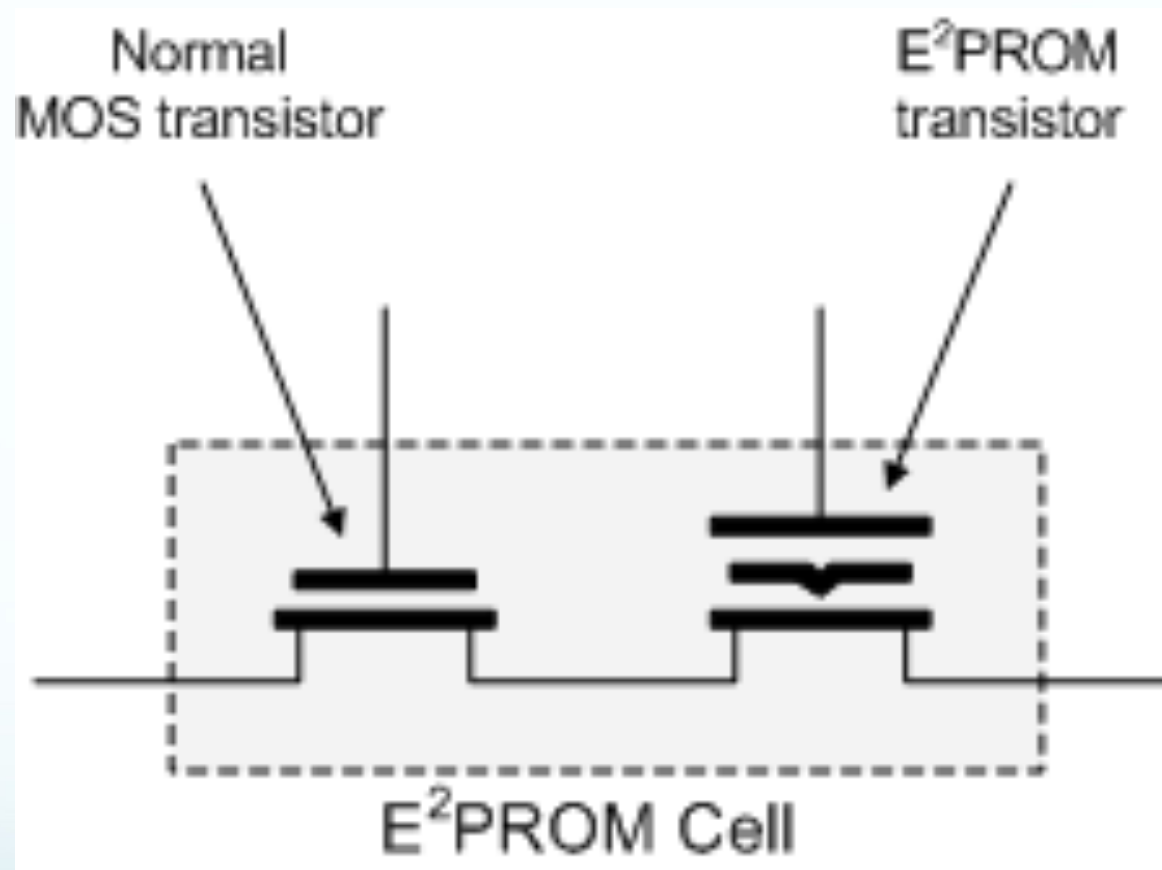
EEPROM: erasable word by word
FLASH: erasable by block or by device

SRAM-Based Devices

Configuration data in

Configuration data out

☐ = I/O pin/pad
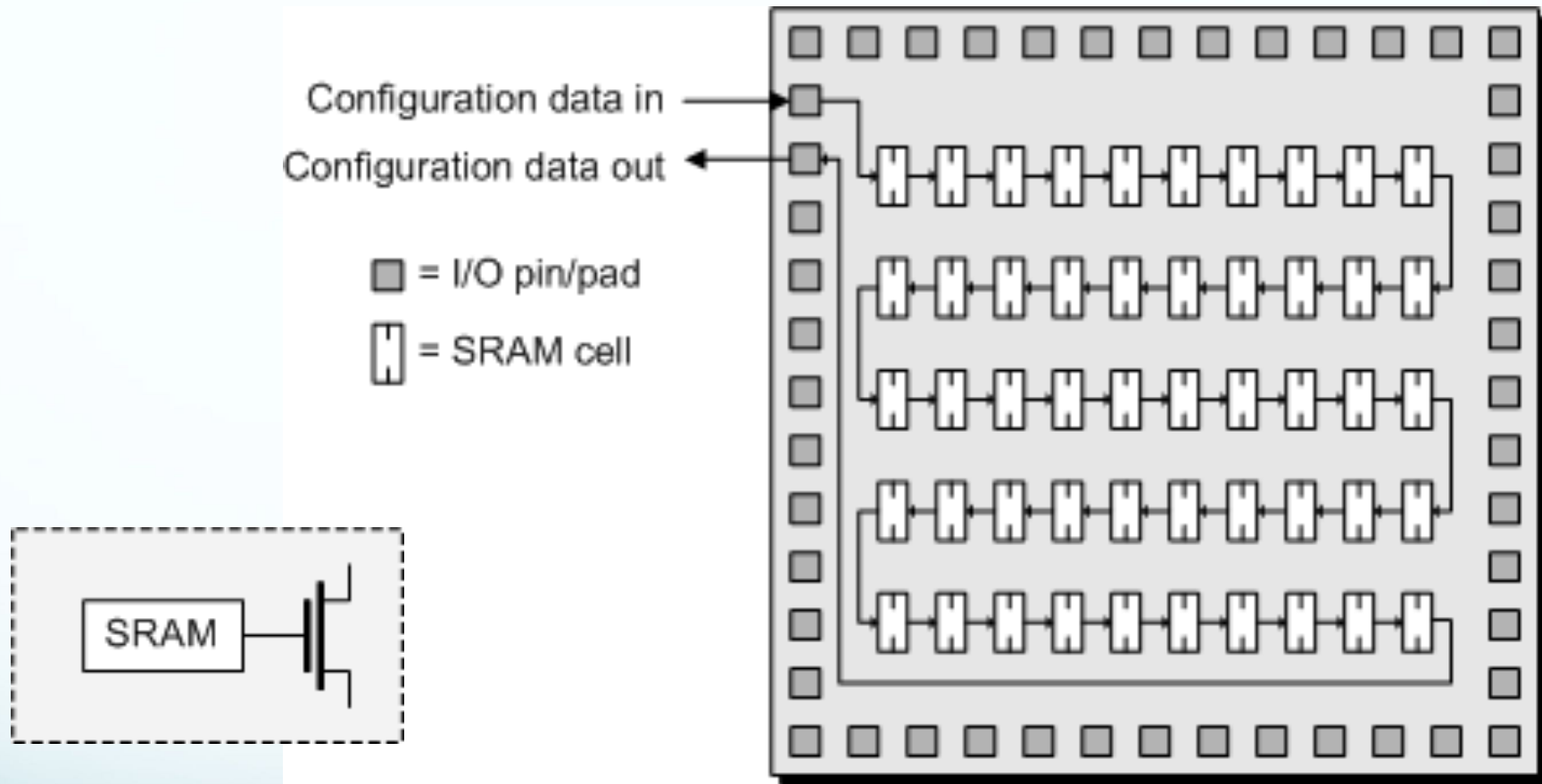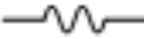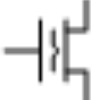
⊔ = SRAM cell

SRAM

Multi-transistor SRAM cell

Programming a 3-bit wide LUT
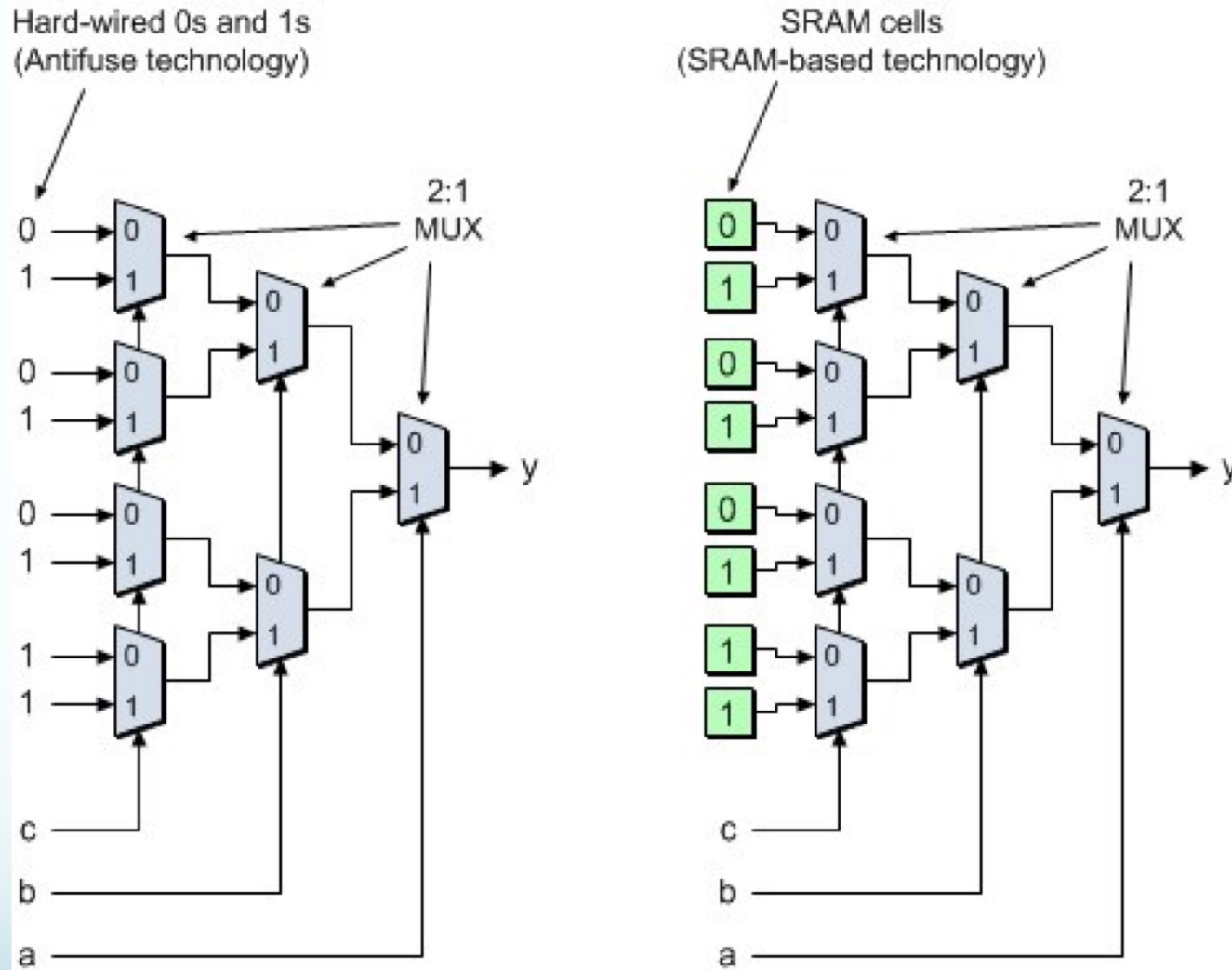
Summary of Technologies

Technology	Symbol	Predominantly associated with ...
Fusible-link	—$\wedge\wedge$—	SPLDs
Antifuse	—$\vdash\square\dashv$—	FPGAs
EPROM		SPLDs and CPLDs
E^2PROM/ FLASH		SPLDs, CPLDs, and FPGAs
SRAM	SRAM	FPGAs (some CPLDs)

Rad-tolerant secure

Rad-tolerant (e.g. Alice)

Used in most FPGAs

Design Considerations (SRAM Config.)
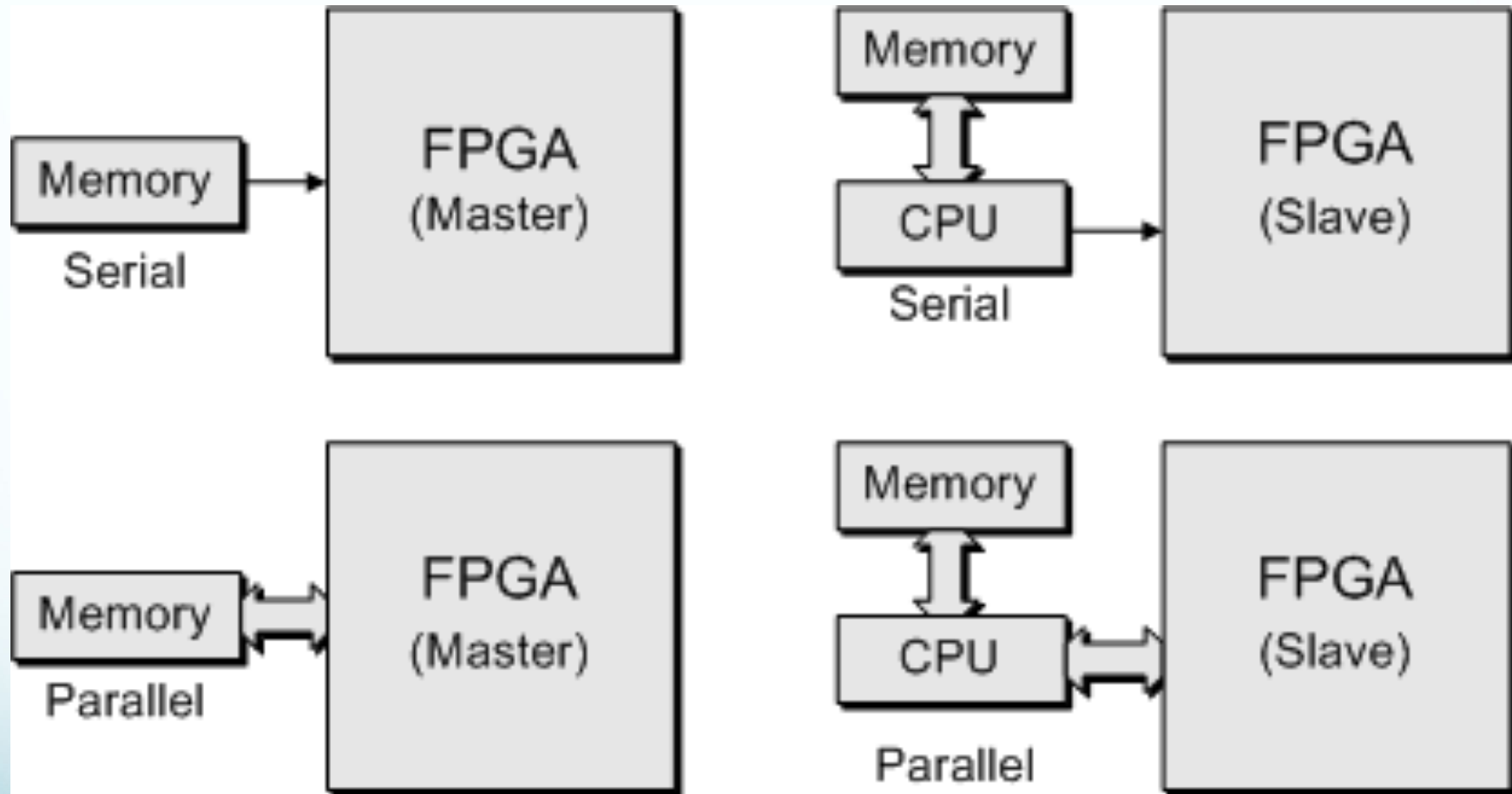
Configuration at power-up

FPGA
(SRAM based)

Flash
PROM

stores
single or
multiple
designs

Serial bit-stream
(may be encrypted)
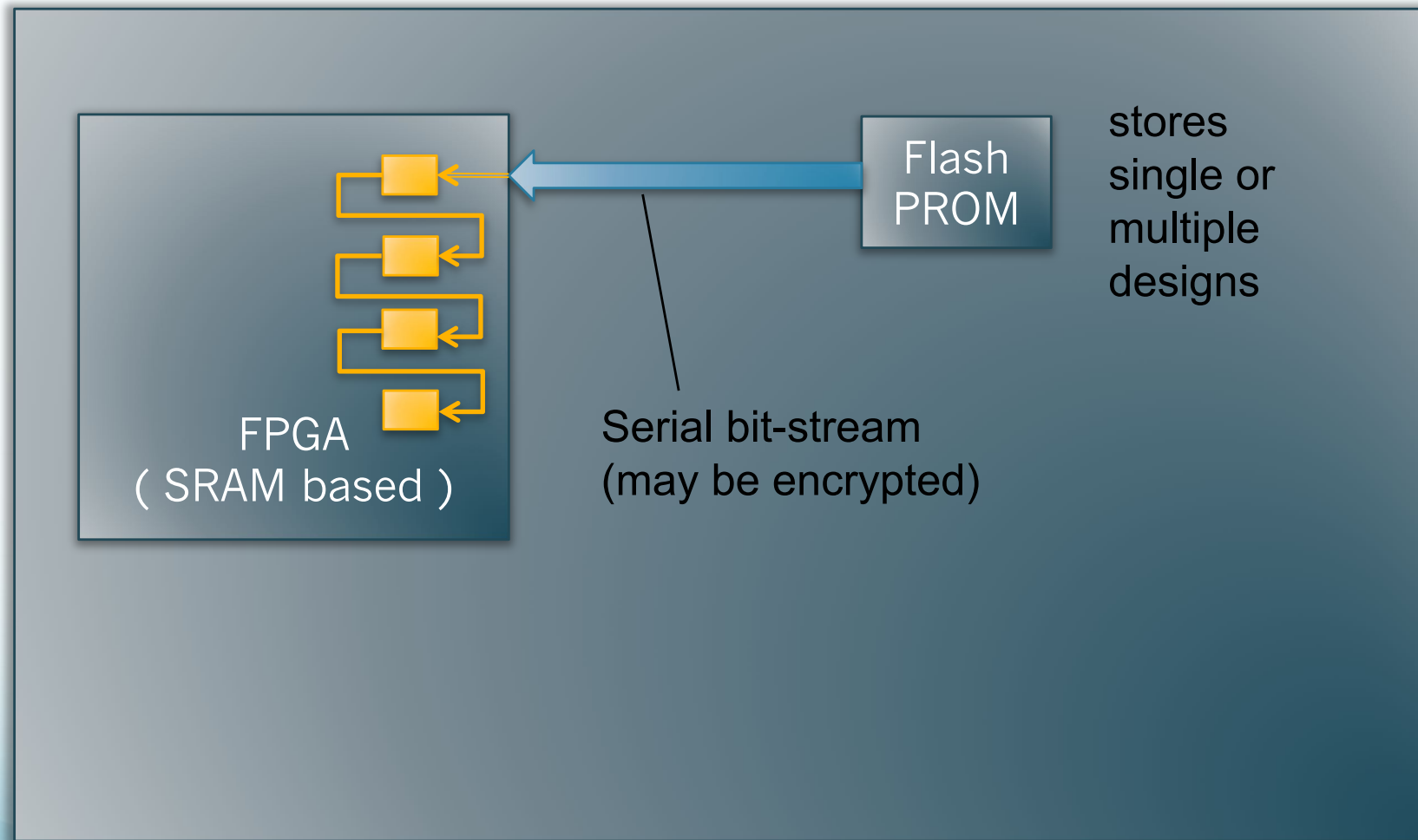
Typical FPGA configuration time: milliseconds

Programming via JTAG

Joint Test Action Group

Flash PROM

JTAG connector

FPGA (SRAM based)

JTAG is a serial bus that can be used to
- Program Flash PROMs
- Program FPGAs
- Read / write the status of all FPGA I/Os
   (= Boundary scan)

Remote programming

Flash PROM

FPGA

PCI, VME
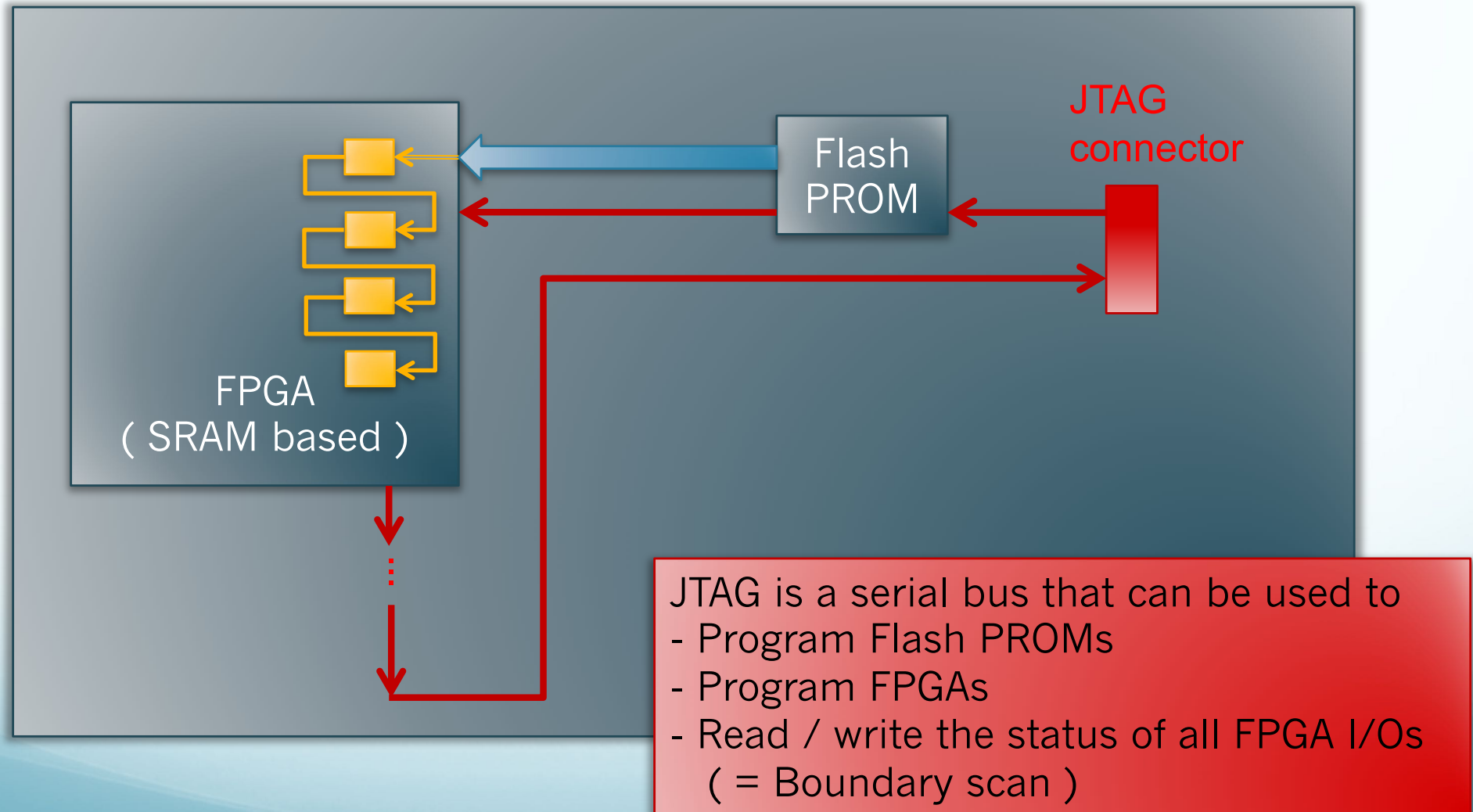
FPGA
(SRAM based)
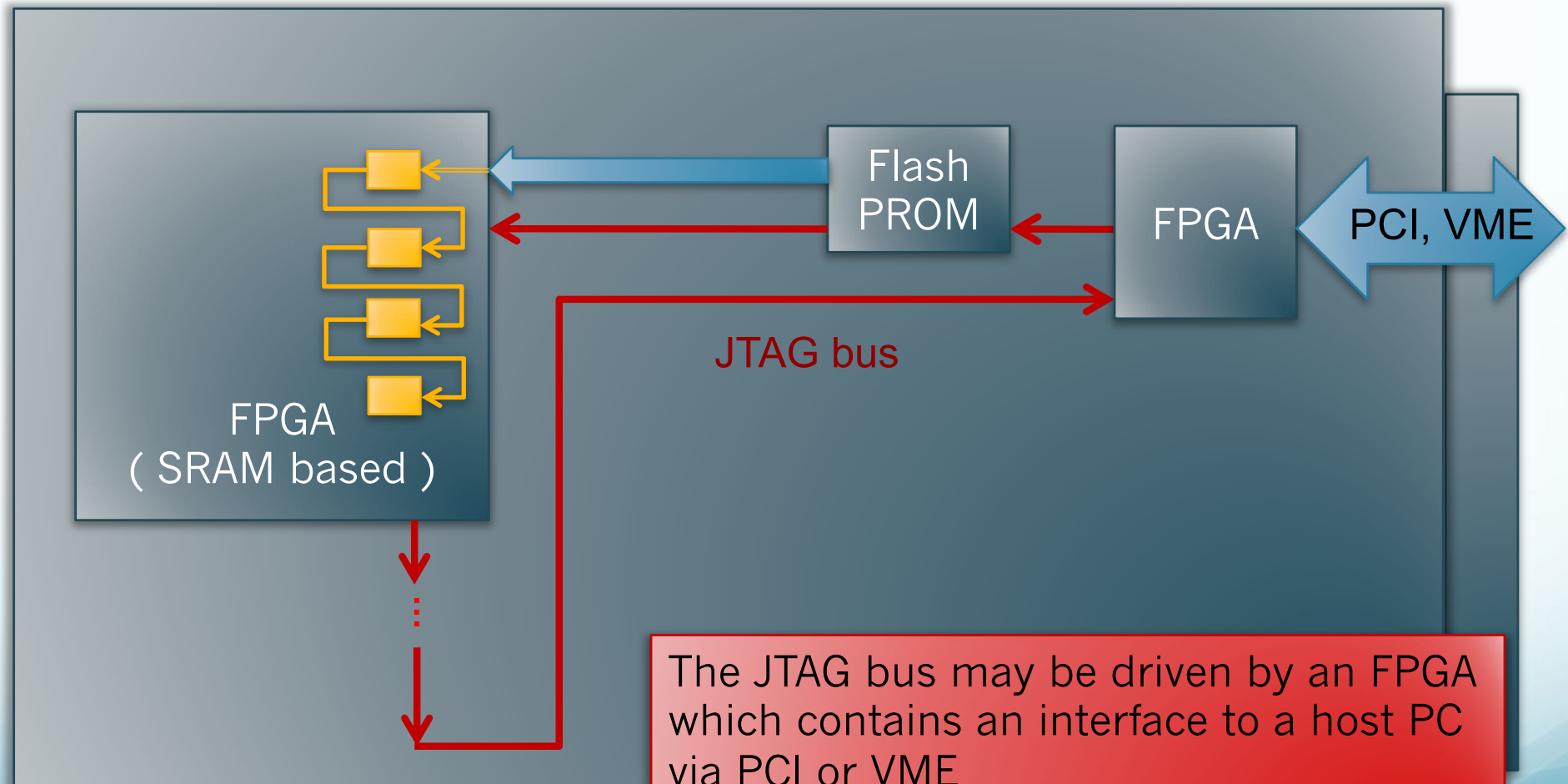
JTAG bus

The JTAG bus may be driven by an FPGA which contains an interface to a host PC via PCI or VME

gateware can then be updated remotely

Major Manufacturers

- Xilinx
  - First company to produce FPGAs in 1985
  - About 55% market share, today
  - SRAM based CMOS devices

- Intel FPGA (formerly Altera)
  - About 35% market share
  - SRAM based CMOS devices

- Microsemi (Actel)
  - Anti-fuse FPGAs
  - Flash based FPGAs
  - Mixed Signal

(Formerly **Actel**®)
POWER MATTERS

- Lattice Semiconductor
  - SRAM based with integrated Flash PROM
  - low power

Trends

Ever-decreasing feature size

10 µm (1971) e.g. Intel 8008

3 µm (1975) e.g. Intel 8088

1.5 µm (1982) e.g. Intel 80286

1 µm (1985) e.g. Intel 80386

800 nm (1989) e.g. P5 Pentium 60 MHz

600 nm (1994) e.g. Motorola PowerPC 601

350 nm (1995) e.g. Pentium II Klamath

250 nm (1998) e.g. AMD K6-2

180 nm (1999) e.g. Coppermine E

130 nm (2000) e.g. PowerPC 7447

90 nm (2002) e.g. VIA C7

65 nm (2006) e.g. Core Duo

45 nm (2008) e.g. Core 2 (Wolfdale)

32 nm (2010) e.g. Core i3 (Clarkdale)

22 nm (2012) e.g. Core i7 (Ivy Bridge)

14 nm (2014) e.g. Core M (Broadwell)

10 nm (2017)

7nm (2019)

Red light (700 nm wavelength)

Violet light (400 nm wavelength)

10 µm

1 µm

100 nm

10 nm

1970   1980   1990   2000   2010

Staphylococcus aureus bacterium

Spermatozoon head

Red blood cell cross-section

- Higher capacity
- Higher speed
- Lower power consumption

130 nm Xilinx Virtex-2
Widely used at LHC startup

28 nm Xilinx Virtex-7 / Altera Stratix V

16 nm Xilinx UltraScale +

14 nm Intel Stratix 10

7 nm Xilinx Versal ACAP(*)

(*) Adaptive Compute Acceleration Platform

Trends

- Speed of logic increasing

- Look-up-tables with more inputs (5 or 6)

- Speed of serial links increasing (multiple Gb/s)

- More integrated memory
  - Integrated High Bandwidth Memory (HBM) in-package
    - 10x faster than DDR4 (Xilinx: up to 8 GB, Intel: up to 16GB)

- Additional Flip Flops in routing resources (Intel hyperflex)

- More and more hard macro cores on the FPGA
  - PCI Express
    - Gen2: 5 Gb/s per lane
    - Gen3: 8 Gb/s per lane   (typically up to 16 lanes)
    - Gen4: 16 Gb/s per lane
  - 10 Gb/s, 40 Gb/s, 100 Gb/s Ethernet, 150 Gb/s Interlaken

- Sophisticated soft macros
  - CPUs
  - Gb/s MACs
  - Memory interfaces (DDR2/3/4)

- Processor-centric architectures – see next slide

System-On-a-Chip (SoC) FPGAs
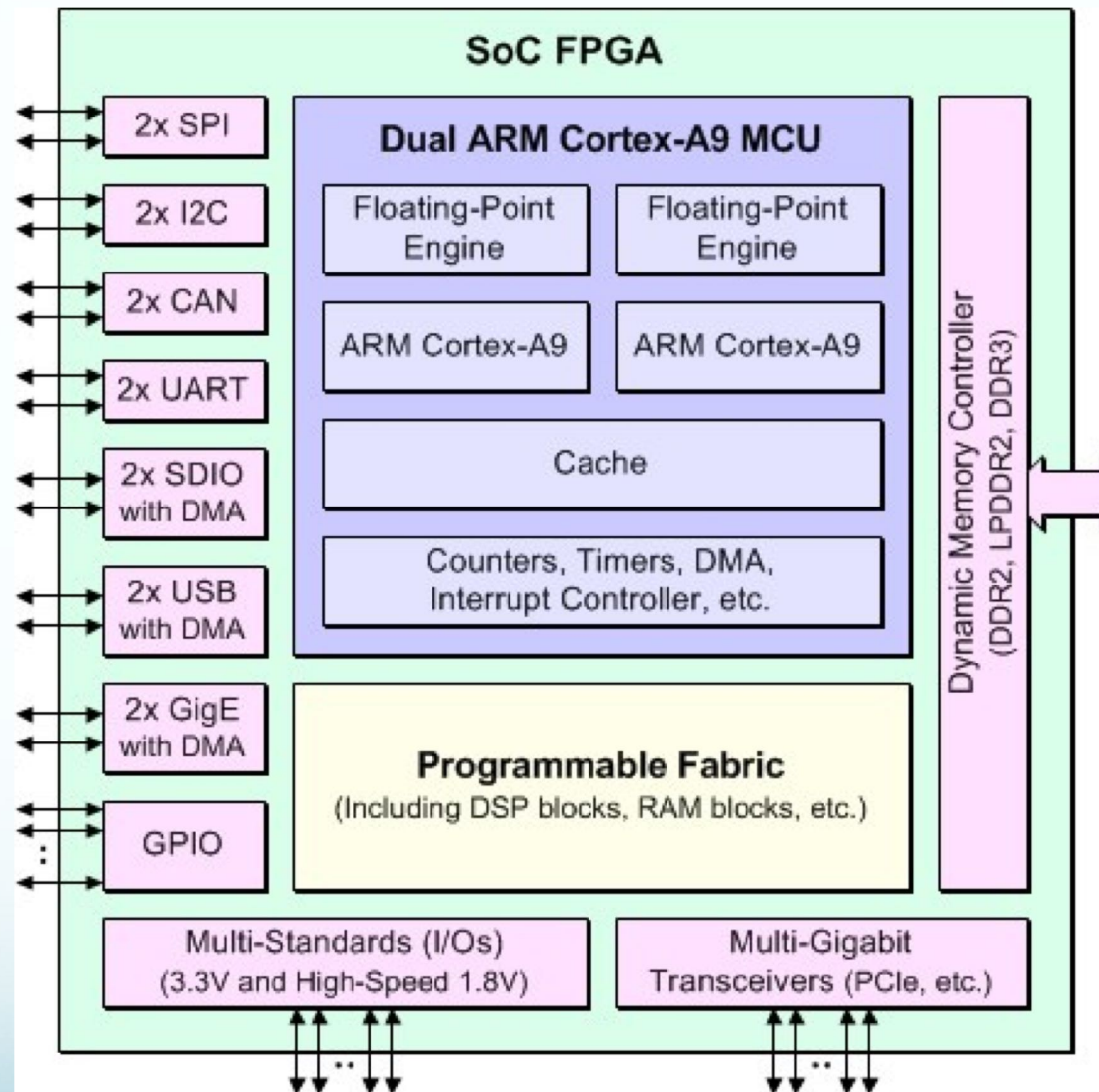
Xlinix Zynq

Intel Stratix 10 SoC

CPU(s) + Peripherals + FPGA in one package

FPGAs in Server Processors and the Cloud

- Released 2018: Intel Xeon Server Processor with FPGA in socket
  - Intel acquired Altera in 2015

  - See: https://www.eejournal.com/article/intel-delivers-xeon-scalable-processor-6138p-with-arria-10-gx-1150-fpga/
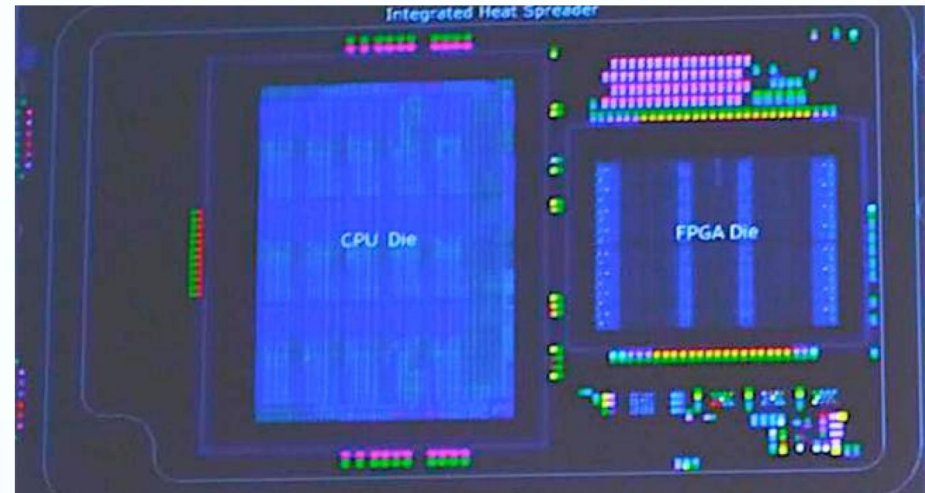
- FPGAs in the cloud
  - Amazon Elastic Cloud F1 instances
    - 8 CPUs / 1 Xilinx UltraScale+ FPGA
    - 64 CPUs / 8 Xilinx UltraScale+ FPGA

FPGA – ASIC comparison

FPGA

- Rapid development cycle (minutes / hours)

- May be reprogrammed in the field (gateware upgrade)
  - New features
  - Bug fixes

- Low development cost
  - You can get started with a development board (< $100) and free software

- High-end FPGAs rather expensive

ASIC

- Higher performance
 - Speed, Area, Power

- Analog designs possible

- Better radiation hardness

- Long development cycle (weeks / months)

- Design cannot be changed once it is produced

- Extremely high development cost
  - ASICs are produced at a semiconductor fabrication facility ("fab") according to your design

- Lower cost per device compared to FPGA, when large quantities are needed

FPGA development

Design entry

Schematics
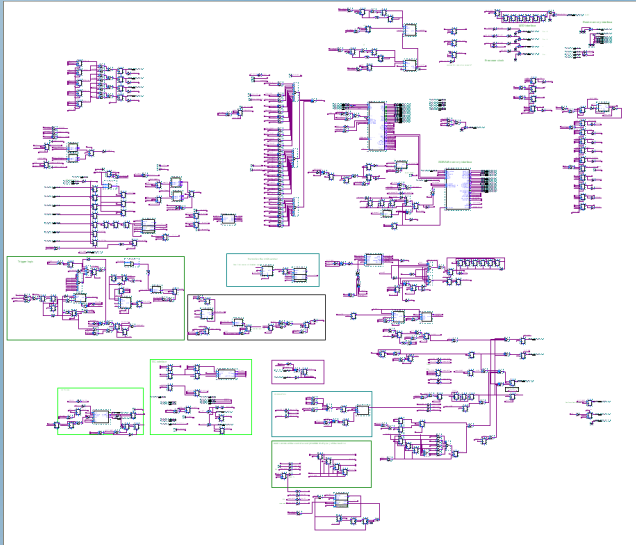
- Graphical overview
- Can draw entire design
- Use pre-defined blocks

Hardware description language
VHDL, Verilog

```
entity DelayLine is

  generic (
    n_halfcycles : integer := 2);

  port (
    x          : in std_logic_vector;
    x_delayed : out std_logic_vector;
    clk        : in std_logic);

end entity DelayLine;
```

- Can generate blocks using loops
- Can synthesize algorithms
- Independent of design tool
- May use tools used in SW development (SVN, git …)

Mostly a personal choice depending on previous experience

Schematics
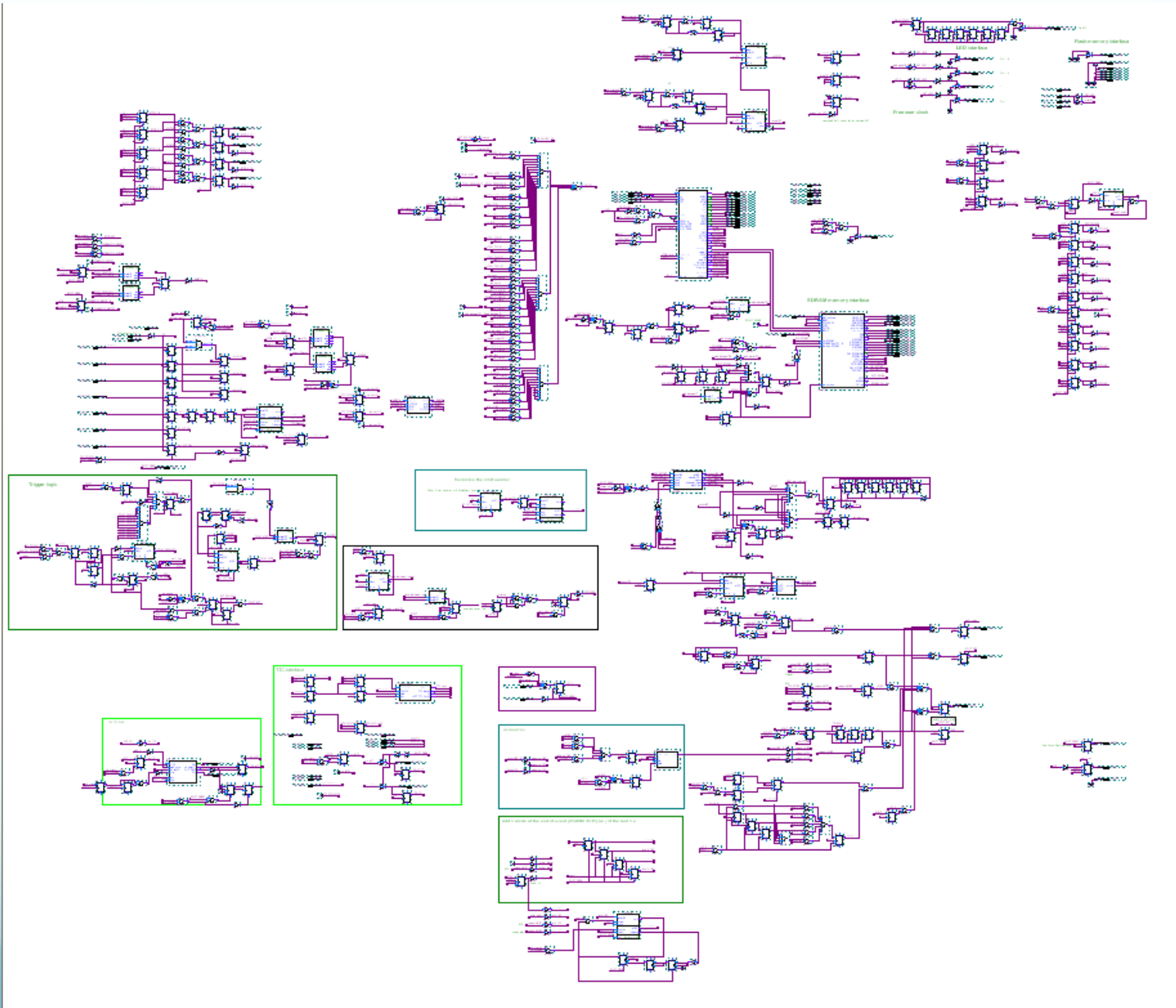
Hardware Description Language

- Looks similar to a programming language
  - BUT be aware of the difference
    - Programming Language => translated into machine instructions that are executed by a CPU
    - HDL => translated into gateware (logic gates & flip-flops)

- Common HDLs
  - VHDL
  - Verilog
  - AHDL (Altera specific)

- Newer trends
  - C-like languages (handle-C, System C)
  - Labview
  - High Level Synthesis (HLS) from C/C++

Example: VHDL

```vhdl
architecture behavioral of VMEReg is

  signal vme_en_i   : std_logic;
  signal Q : std_logic_vector(15 downto 0);

begin  -- behavioral

  vme_addr_decode : process (vme_addr, vme_en) is
    variable my_addr_vec : std_logic_vector(vme_addr'high downto 0);
    variable selected    : boolean;
  begin  -- process vme_addr_decode
    my_addr_vec := std_logic_vector( TO_UNSIGNED ( my_vme_base_address, vme_addr'high+1 ) );
    selected    := my_addr_vec(vme_addr'high downto 1) = vme_addr(vme_addr'high downto 1);
    vme_en_i <= '0' ;
    if selected then
      vme_en_i <= vme_en;
    end if;
  end process vme_addr_decode;

  reg: process (vme_clk, reset) is
  begin  -- process reg
    if reset = '1' then                   -- asynchronous reset
      Q <= init_val;
      vme_en_out <= '0';
    elsif vme_clk'event and vme_clk = '1' then  -- rising clock edge
      vme_en_out <= vme_en_i;
      if vme_en_i = '1' and vme_wr = '1' then
        Q <= vme_data;
      end if;
    end if;
  end process reg;

  data <= Q;
  vme_data_out <= Q;

end behavioral;
```

Asynchronous logic
All signals in sensitivity list

Synchronous logic
Only clock (and reset) in sensitivity list

- Looks like a programming language

- All statements executed in parallel, except inside processes
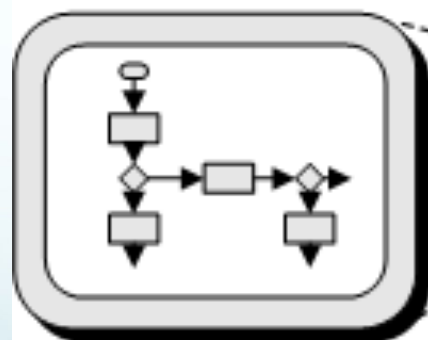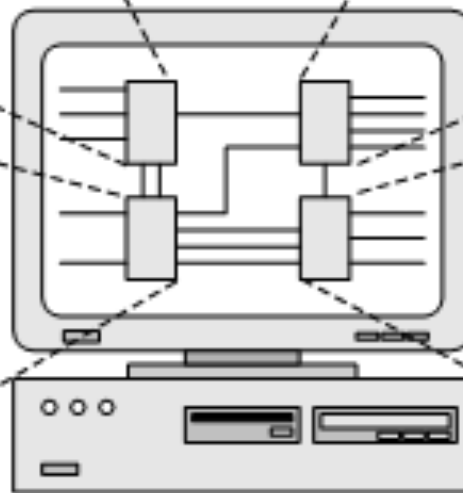
Schematics & HDL combined

Graphical State Diagram
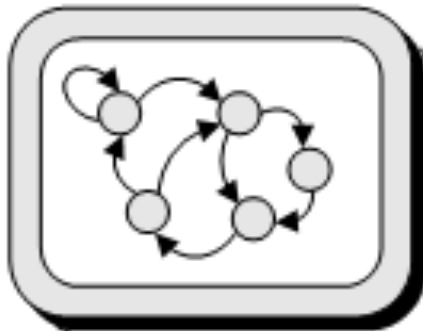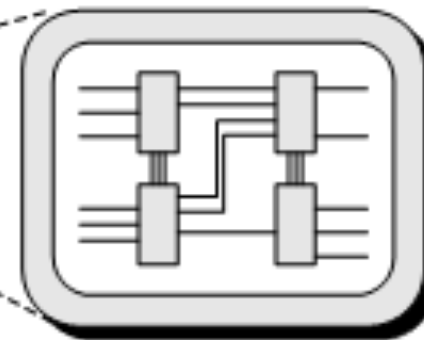
Textual HDL

When clock rises
  If (s == 0)
    then y = (a & b) | c;
    else y = c & !(d ^ e);

Top-level block-level schematic

Graphical Flowchart

Block-level schematic

Design flow

C/C++
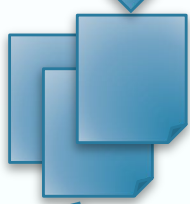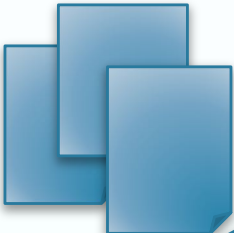
High Level Synthesis
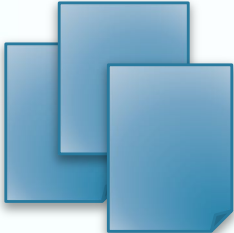
constraints

Schematics

VHDL / Verilog

State Machines

IP Integrator

Commercial Intellectual Property cores

Pins
Timing
Area
...

Counters
FIFOs
...

Processors
Interfaces
Controllers
...

Synthesis

Behavioral Simulation

Register Transfer Level (RTL) model
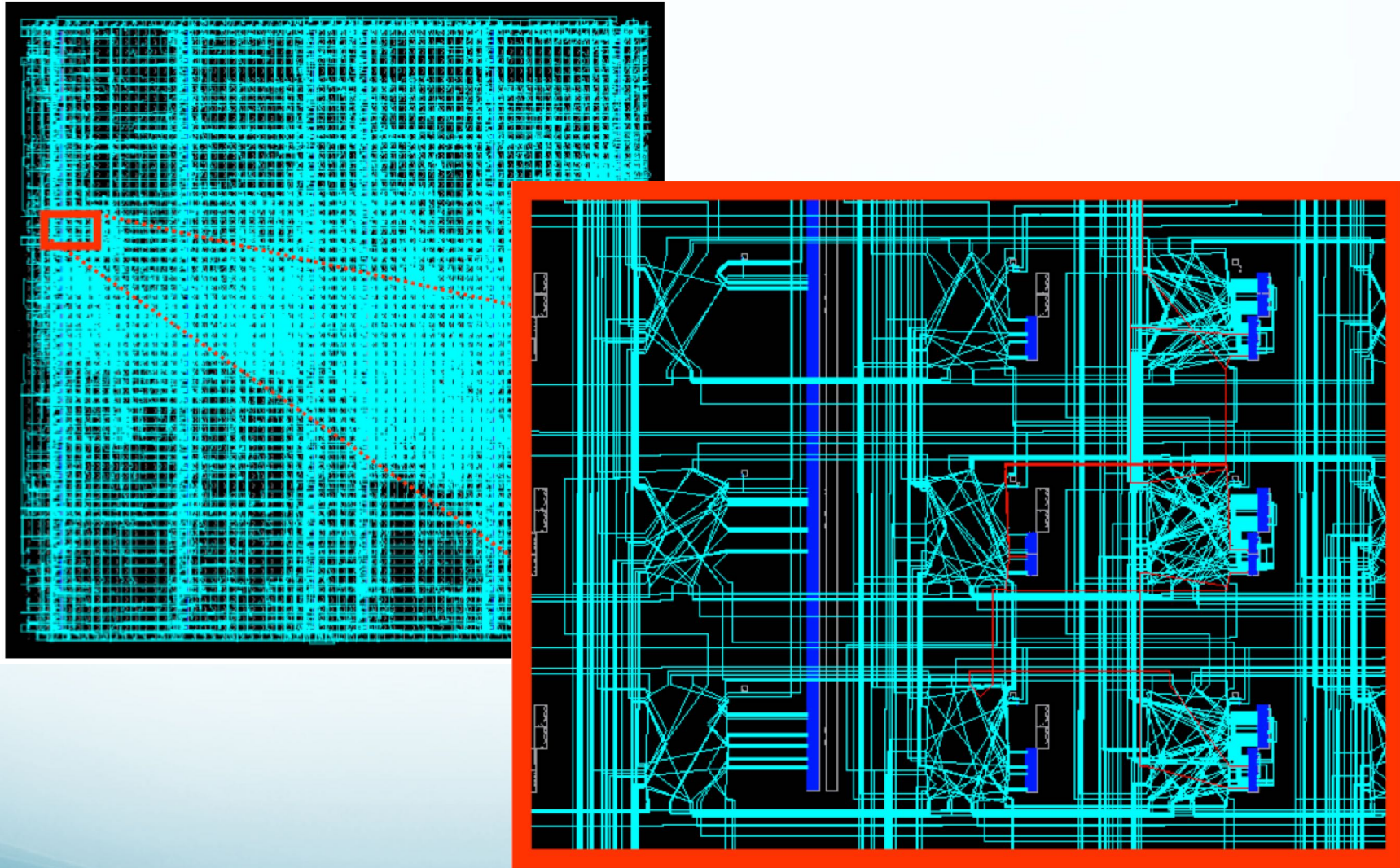
Implementation
Map
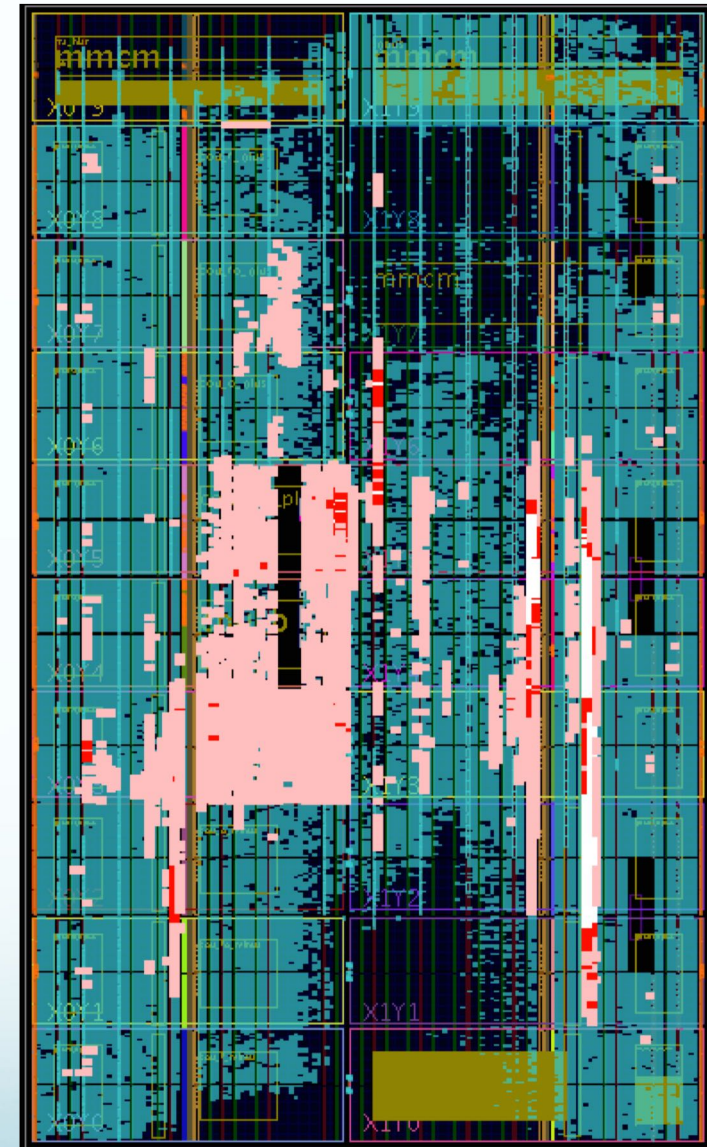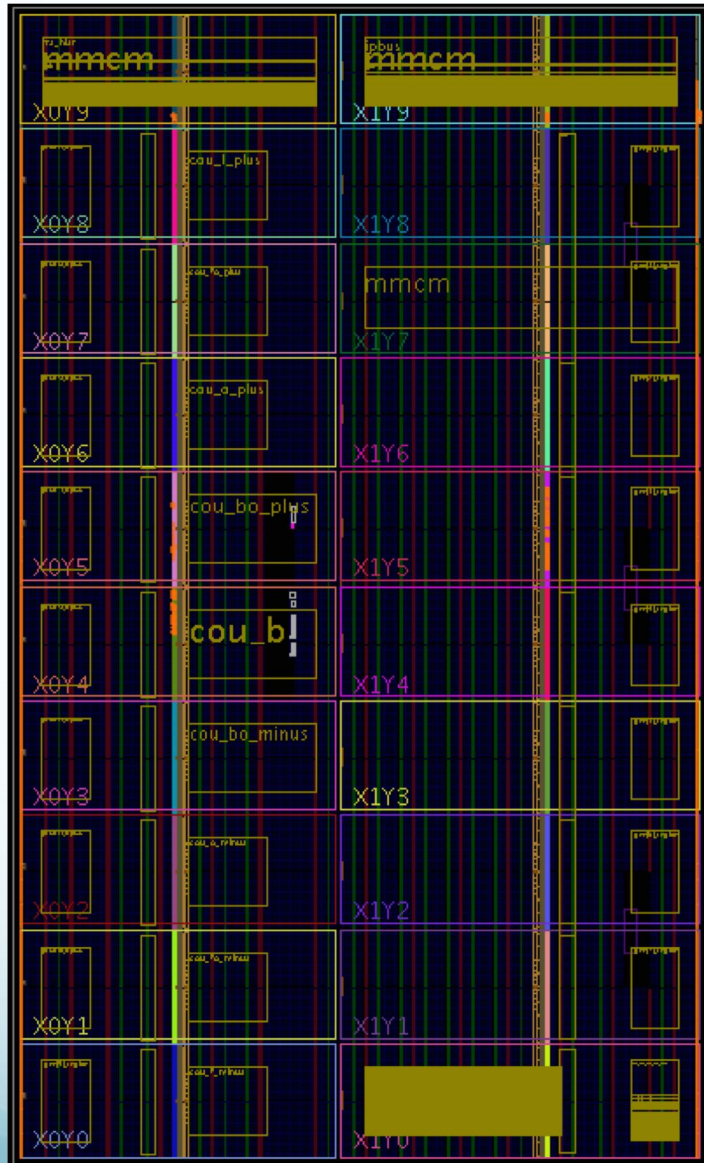Place & Route

Static Timing Analysis

Programming file

Timing Simulation

Floorplan (Xlinx Virtex 2)

Manual Floor planning

For large designs, manual
floor planning may be necessary

Routing congestion
Xilinx Virtex 7 (Vivado)

Simulation
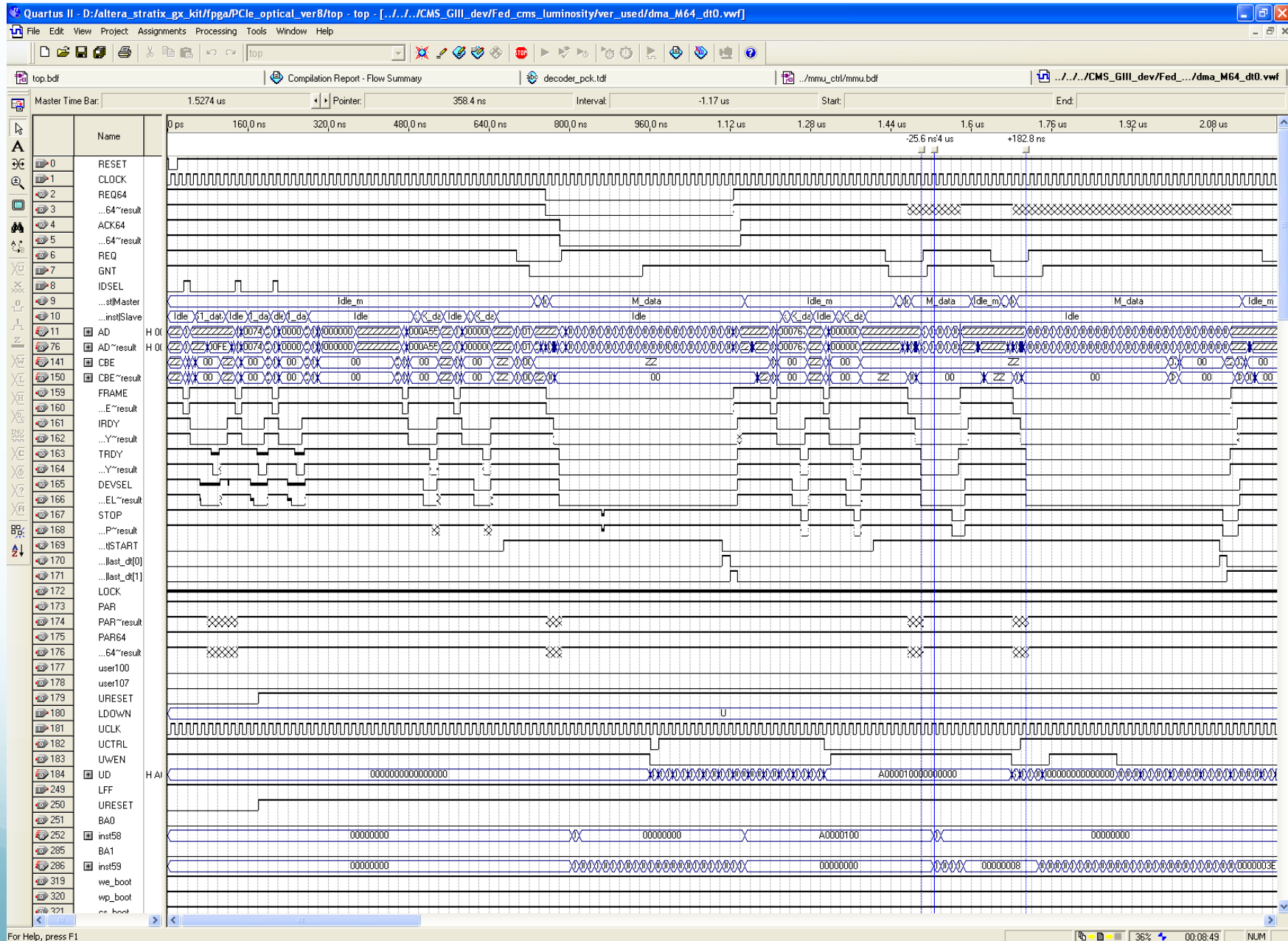
Embedded Logic Analyzers
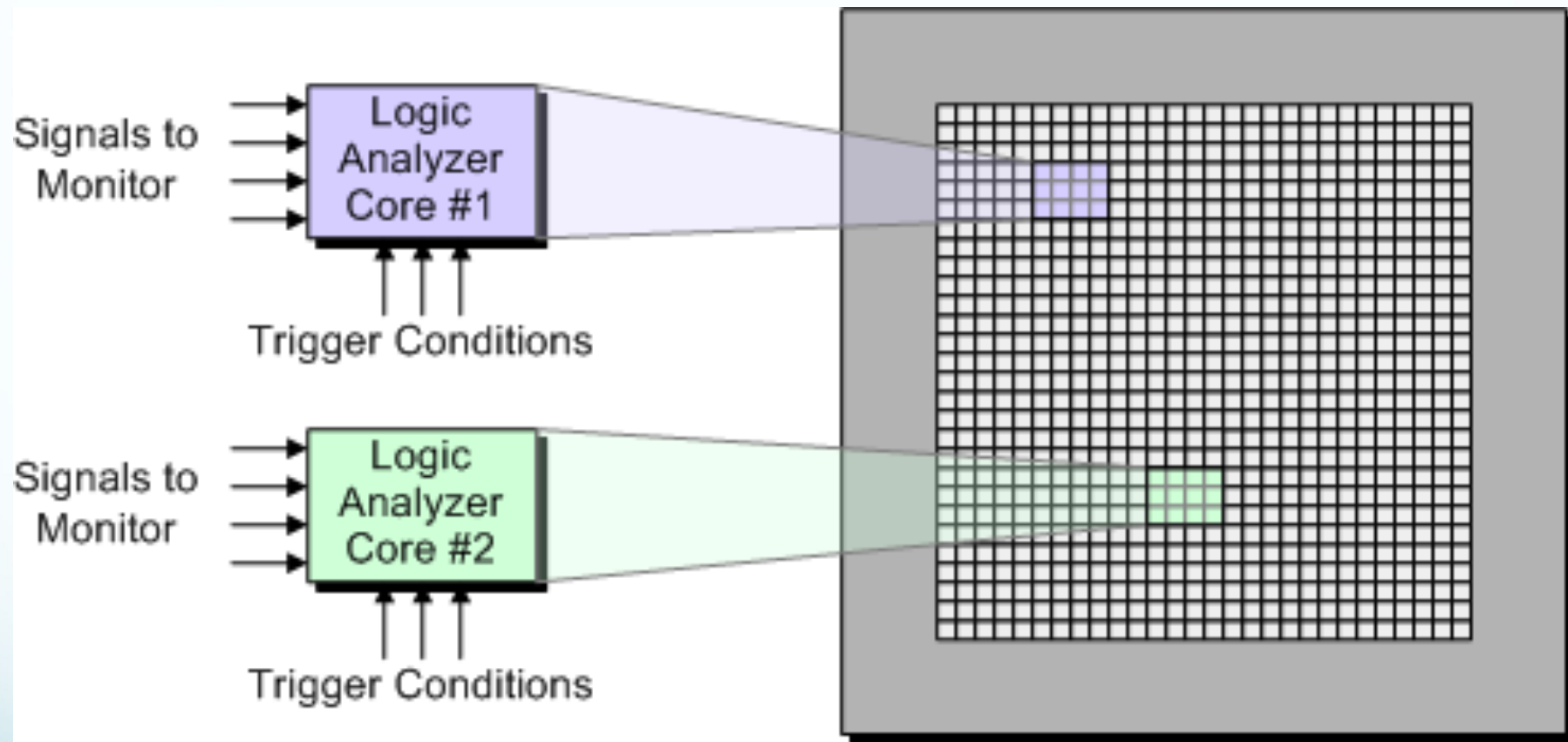
A great tool for debugging your design

FPGA applications
in the Trigger & DAQ domain

First-Level Trigger at Collider
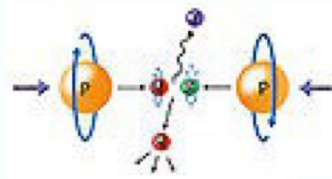
Timing: beam crossings

LHC: 25 ns

detector

Coarse grain data

Full data
(fine grain)

Delay
FIFO

First Level Trigger

Pipelined
Logic
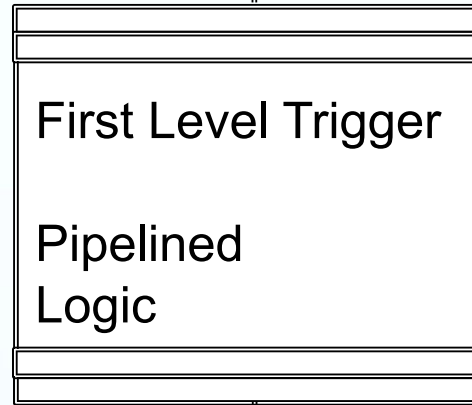
Fixed Latency
(= processing time
of the first
level trigger)

N beam crossings

Trigger decision YES / NO
(for every beam crossing)

De-
randomizer
FIFO

Latency should be short
In order to limit the length
of the delay FIFOS

Pipelined Logic
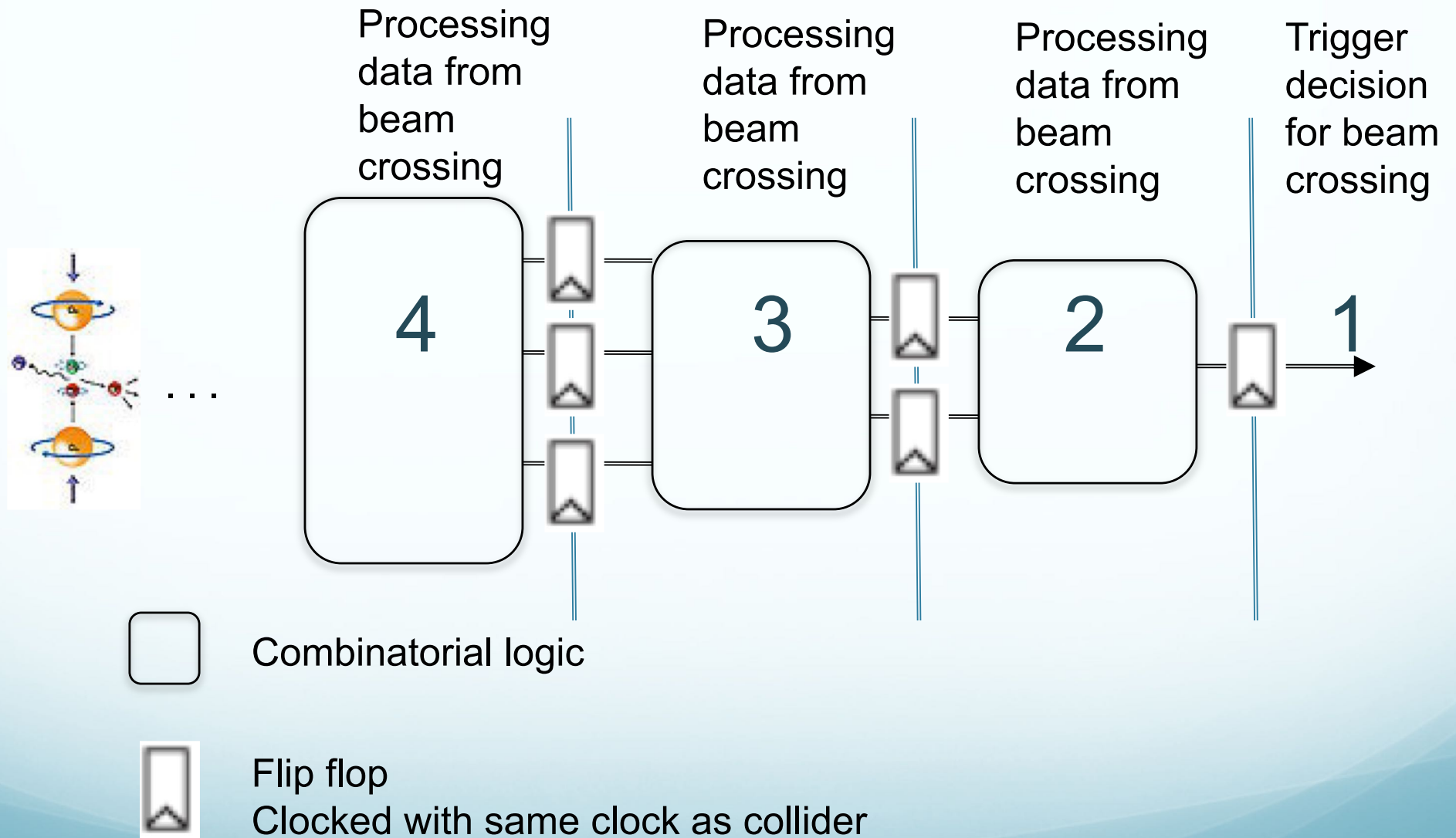
Pipelined Logic – a clock cycle later

Processing data from beam crossing

Processing data from beam crossing

Processing data from beam crossing

Trigger decision for beam crossing

5   4   3   2

☐ Combinatorial logic

☐ Flip flop
Clocked with same clock as collider
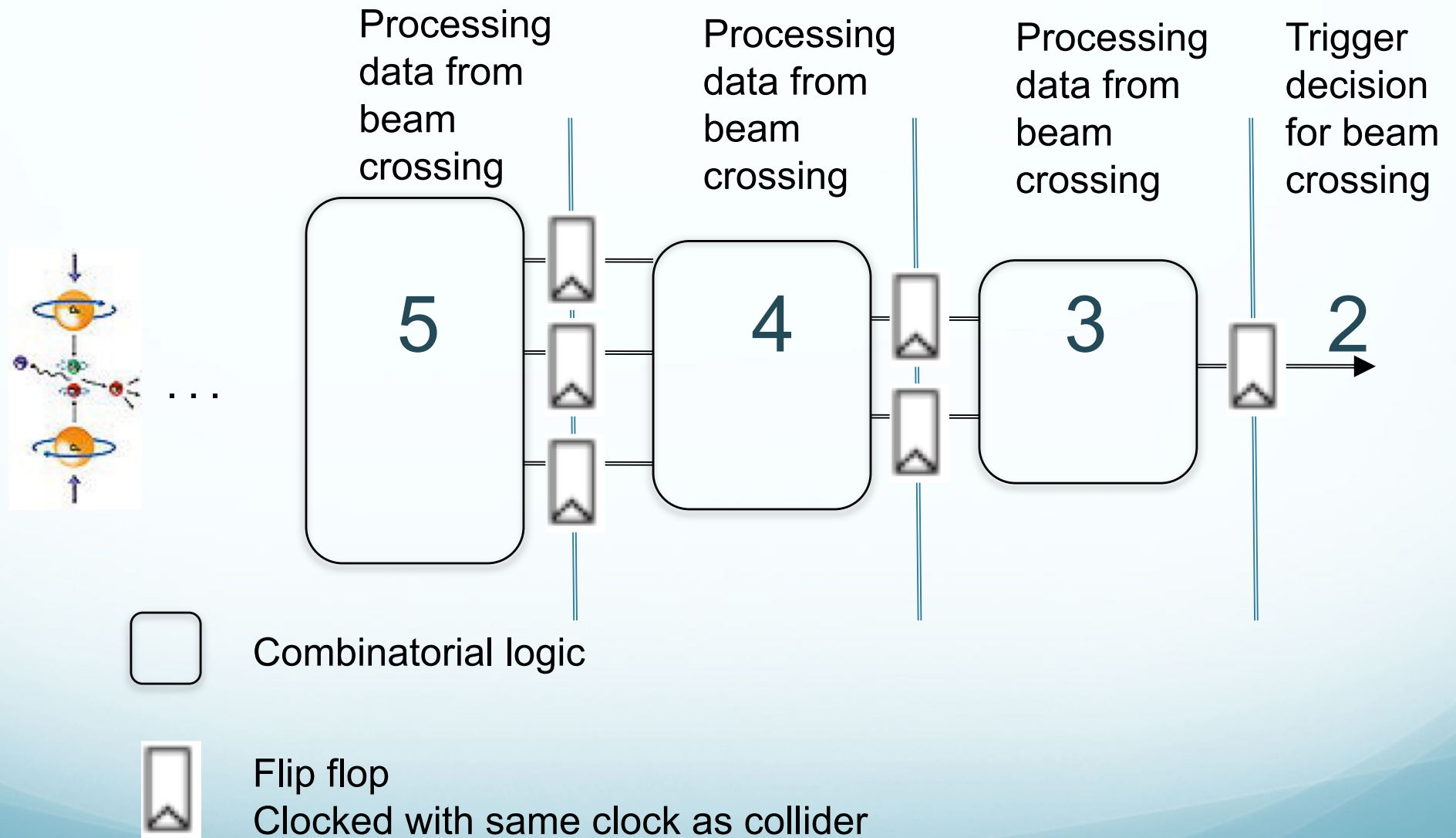
Why are FPGAs ideal for First-Level Triggers ?

- They are fast
  - Much faster than discrete electronics (shorter connections)

- Many inputs
  - Data from many parts of the detector has to be combined

Low latency

- All operations are performed in parallel
  - Can build pipelined logic

- They can be re-programmed
  - Trigger algorithms can be optimized

High performance

Trigger algorithms implemented in FPGAs

- Peak finding

- Pattern Recognition

- Track Finding

- Clustering / Energy summing

- Sorting

- Topological Algorithms (invariant mass)

- Trigger Control system

- Fast signal merging

- New: Inference with Neural Networks

- Many more …

CMS Global Muon Trigger

- Input: ~1000 bits
  @ 40 and 80 MHz

- Output: ~50 bits @ 80MHz

- Processing time: 250 ns

- Pipelined logic
  one new result every 25 ns

- 10 Xilinx Virtex-II FPGAs

- up to 500 user I/Os per chip

- Up to 25000 LUTs per chip used

- Up to 96 x 18kbit RAM used

- In use in the CMS trigger 2008-2015
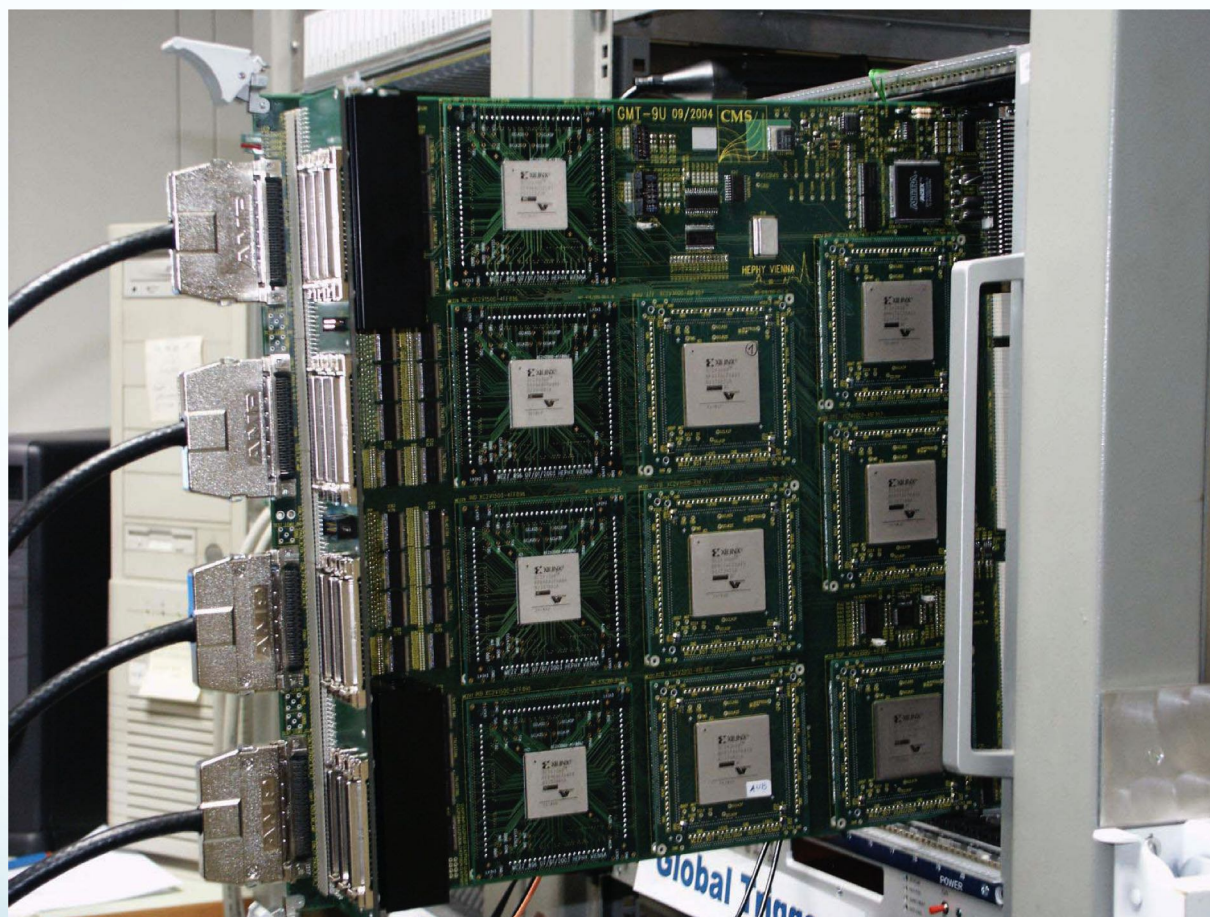
- The CMS Global Muon trigger received 16 muon candidates from the three muon systems of CMS

- It merged different measurements for the same muon and found the best 4 over-all muon candidates

CMS Global Muon Trigger main FPGA
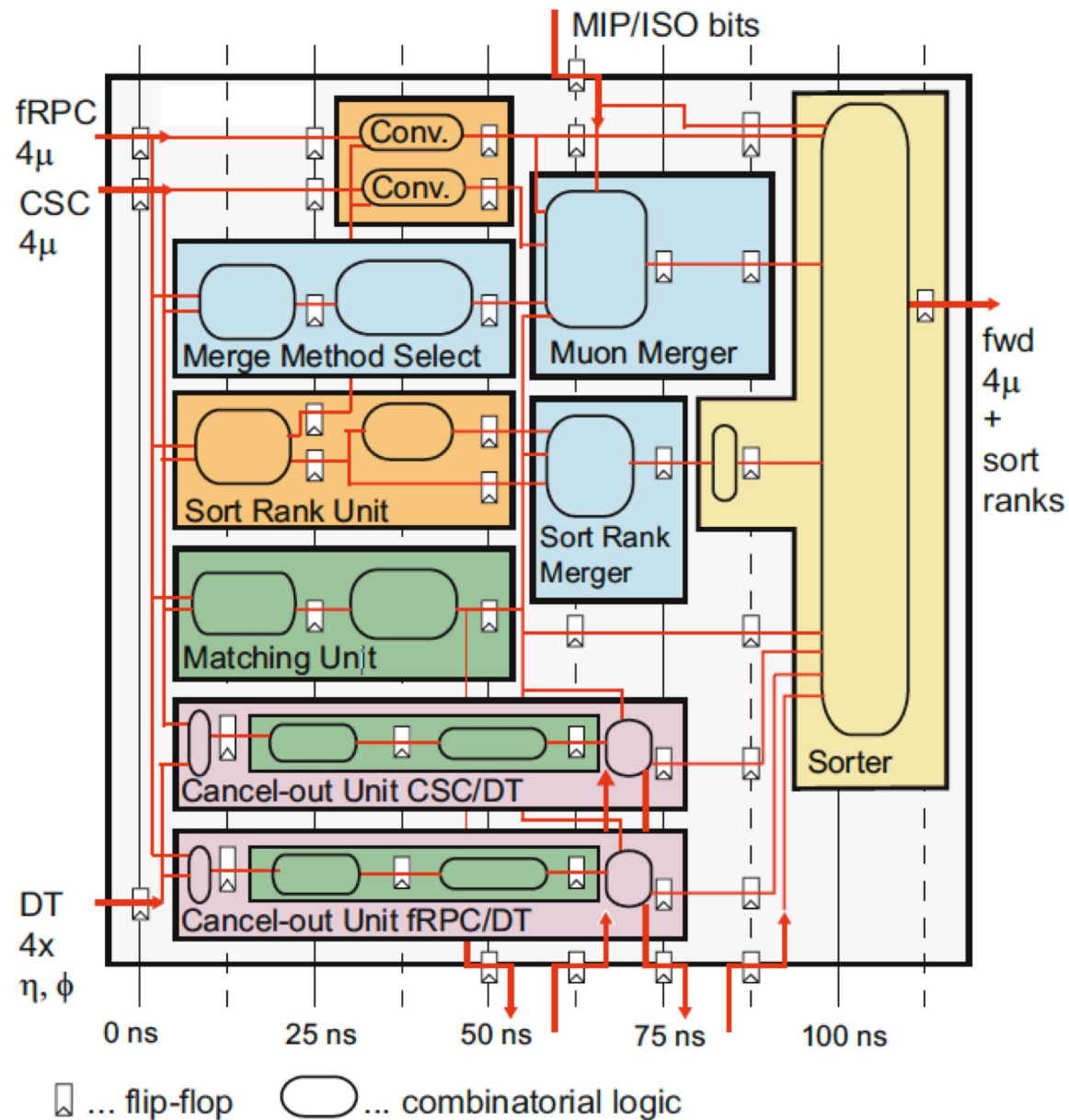
µTCA board for Run 2&3 CMS trigger based on Virtex 7

360 Gb/s
36 x
10 Gb/s

Rx
Tx
Rx
Tx
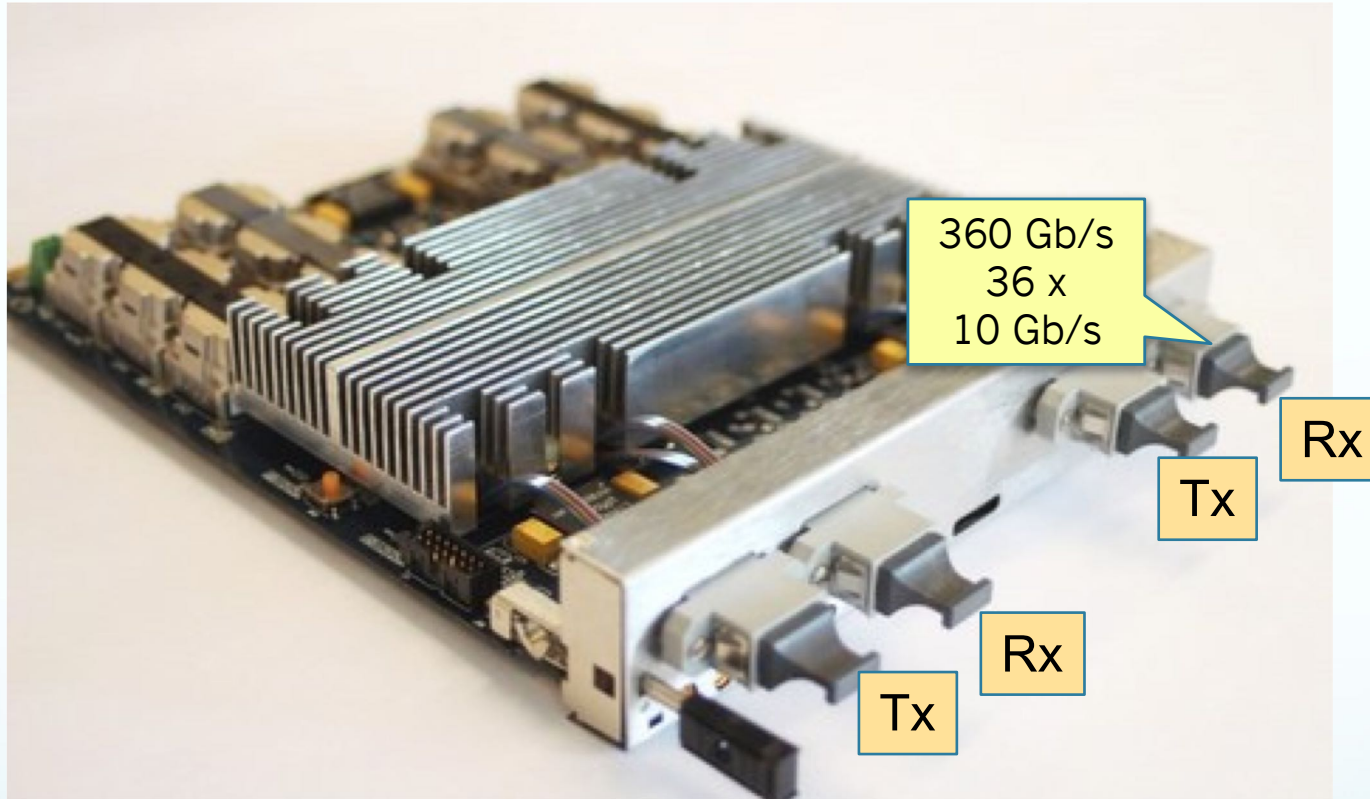
MP7, Imperial College

Virtex 7 with 690k logic cells
    80 x 10 Gb/s transceivers bi-directional
    72 of them as optical links on front panel
    0.75 + 0.75 Tb/s
Being used in the CMS trigger since 2015

Input/output:
up to 14k bits per 40 MHz clock

Same board used for different functions
(different gateware)
Separation of framework + algorithm fw

CMS ATCA Trigger boards for HL-LHC

Serenity, UK

APX, US
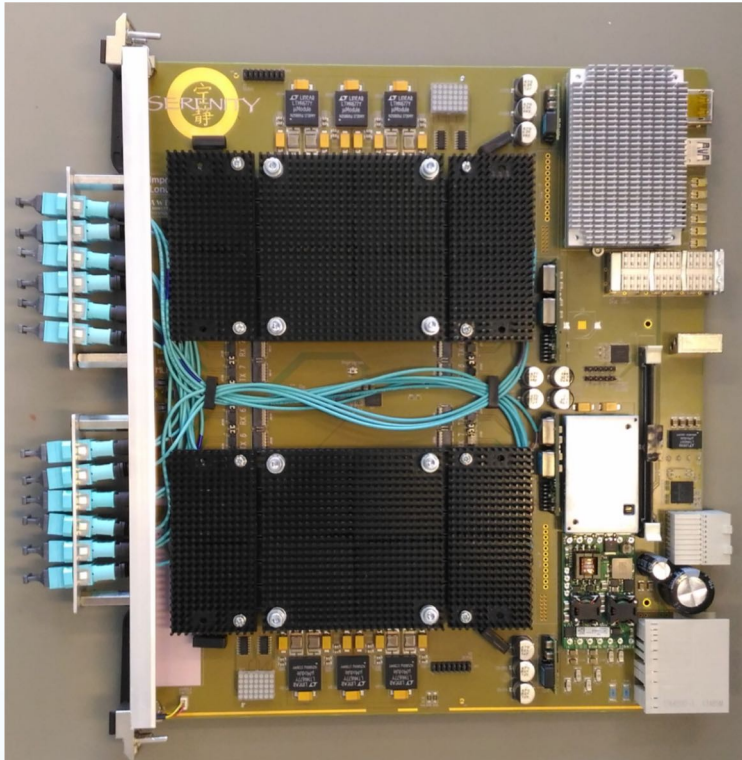
- Few types of generic boards, ATCA standard
- Xilinx Virtex/Kintex Ultrascale+ FPGAs
- 25-28 Gb/s optical links
- SoC FPGAs used for board control (on some boards)
- Advanced firmware algorithms
  - Vertex finding
  - Particle flow
  - Neural network classifiers

Neural Networks in Trigger

One or many hidden layers
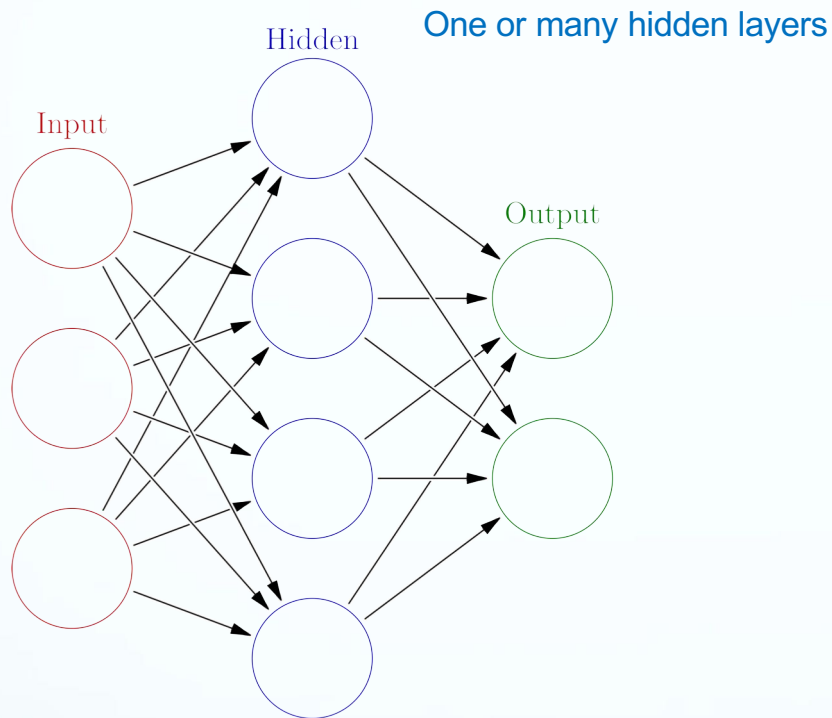
By Glosser.ca - Own work, Derivative of File:Artificial neural network.svg, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=24913461
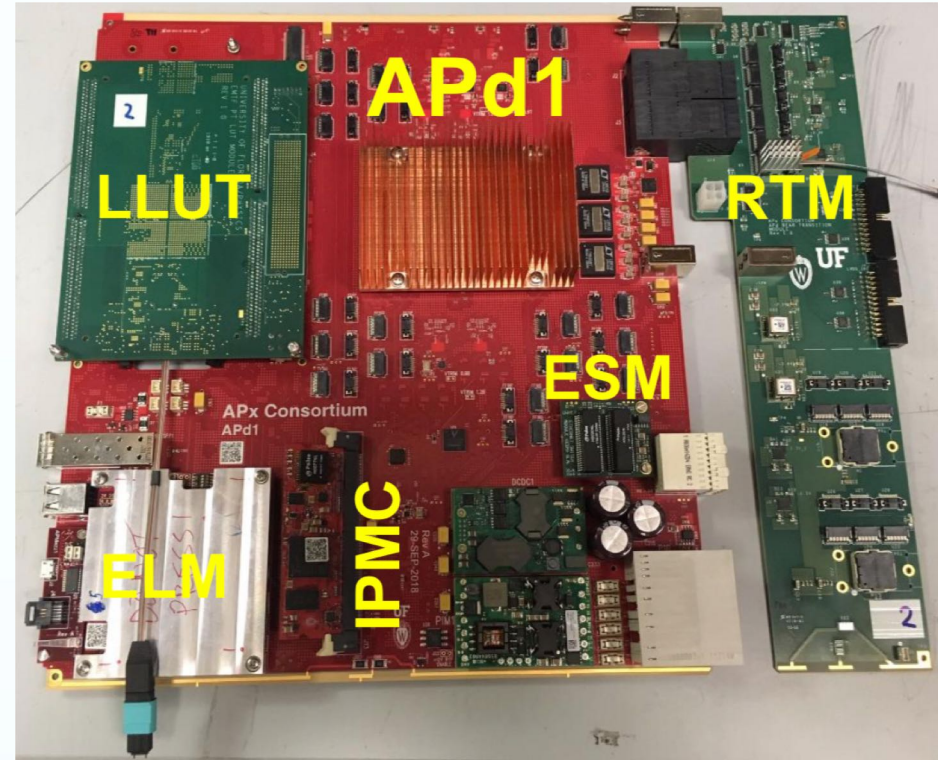
- Principle
  - Node is assigned a value based on the weighted sum of nodes in the previous layer
  - Maps well to DSP resources in FPGA (multiplier + adder)

- Applications:
  - Jet classification
  - Assignment of transverse momentum based on many measurements
  - ...

- Tools
  - Many commercial tools
  - hls4ml (optimized for latency)
    - Firmware generation from high-level model using Vivado HLS

See lecture by Sioni on Friday

hls 4 ml

FPGAs in Data Acquisition
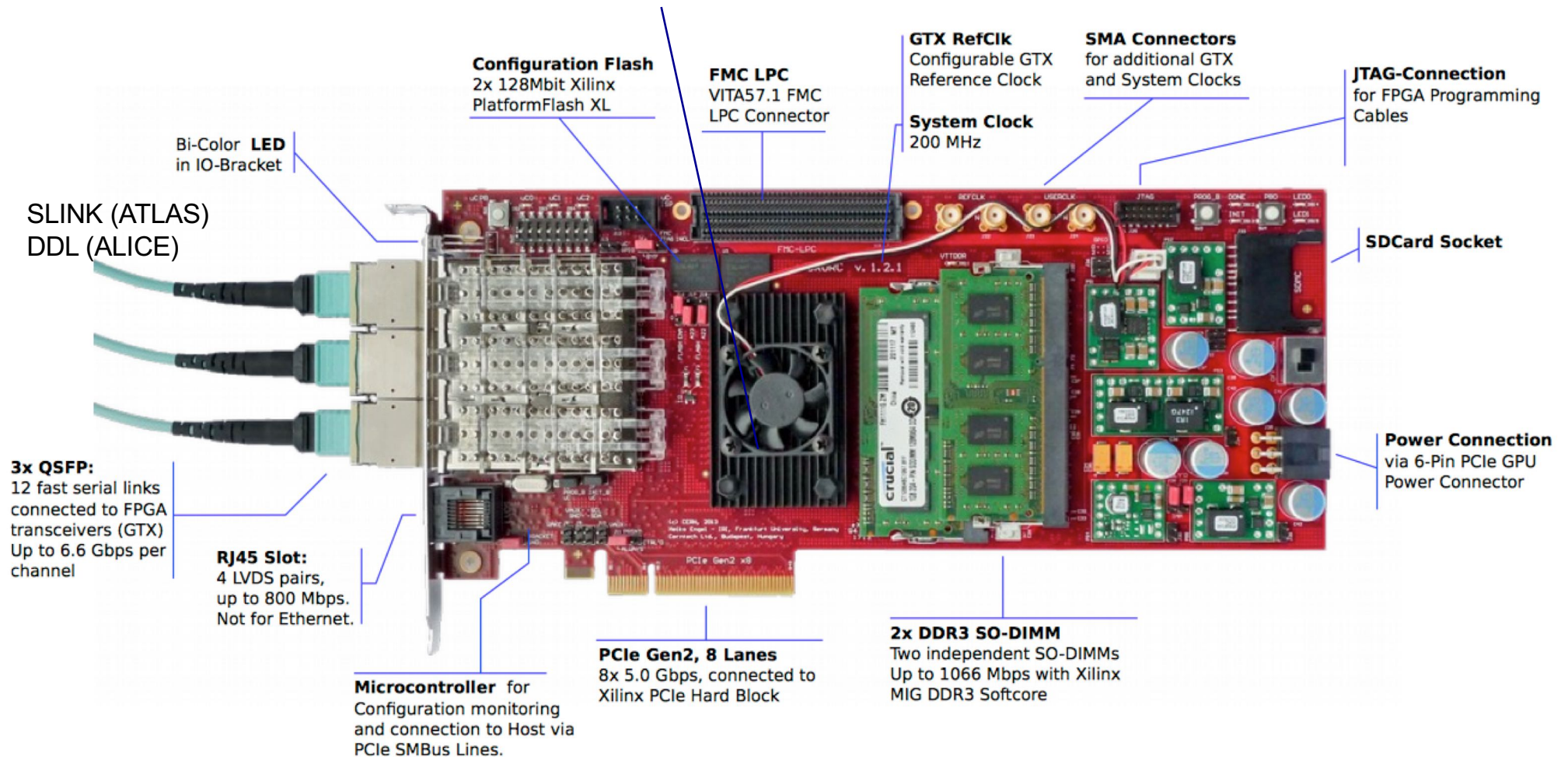
- Frontend Electronics
  - Pedestal subtraction
  - Zero suppression
  - Compression
  - ...

- Custom data links
  - E.g. SLINK-64 over copper
    - Several serial LVDS links in parallel
    - Up to 400 MB/s
  - SLINK/SLINK-express over optical

- Interface from custom hardware to commercial electronics
  - PCI/PCIe, VME bus, Myrinet, 10/40/100 Gb/s Ethernet etc.

C-RORC (Alice) / Robin NP (ATLAS) for Run-2

Xilinx Virtex-6 FPGA

Configuration Flash
2x 128Mbit Xilinx
PlatformFlash XL

FMC LPC
VITA57.1 FMC
LPC Connector

GTX RefClk
Configurable GTX
Reference Clock

SMA Connectors
for additional GTX
and System Clocks

JTAG-Connection
for FPGA Programming
Cables

System Clock
200 MHz

Bi-Color **LED**
in IO-Bracket

SLINK (ATLAS)
DDL (ALICE)

SDCard Socket

3x QSFP:
12 fast serial links
connected to FPGA
transceivers (GTX)
Up to 6.6 Gbps per
channel

Power Connection
via 6-Pin PCIe GPU
Power Connector

RJ45 Slot:
4 LVDS pairs,
up to 800 Mbps.
Not for Ethernet.

Microcontroller for
Configuration monitoring
and connection to Host via
PCIe SMBus Lines.

PCIe Gen2, 8 Lanes
8x 5.0 Gbps, connected to
Xilinx PCIe Hard Block

2x DDR3 SO-DIMM
Two independent SO-DIMMs
Up to 1066 Mbps with Xilinx
MIG DDR3 Softcore

CMS Front-end Readout Link (Run-1)

- SLINK Sender Mezzanine Card: 400 MB / s
  - 1 FPGA (Altera)
  - CRC check
  - Automatic link test

Commercial Myrinet Network Interface Card on internal PCI bus

- Front-end Readout Link Card
  - 1 main FPGA (Altera)
  - 1 FPGA as PCI interface
  - Custom Compact PCI card
  - Receives 1 or 2 SLINK64
  - 2nd CRC check
  - Monitoring, Histogramming
  - Event spy
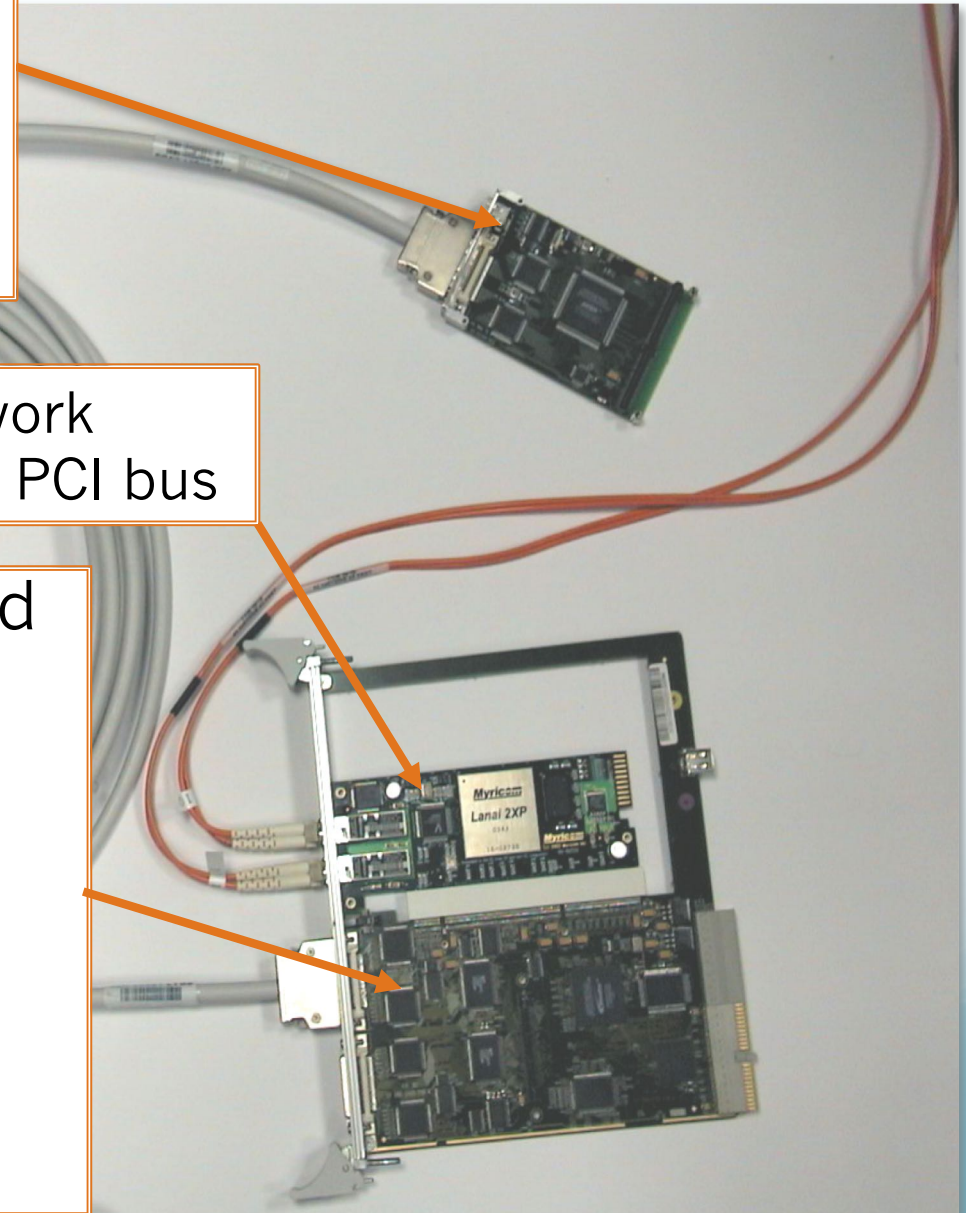
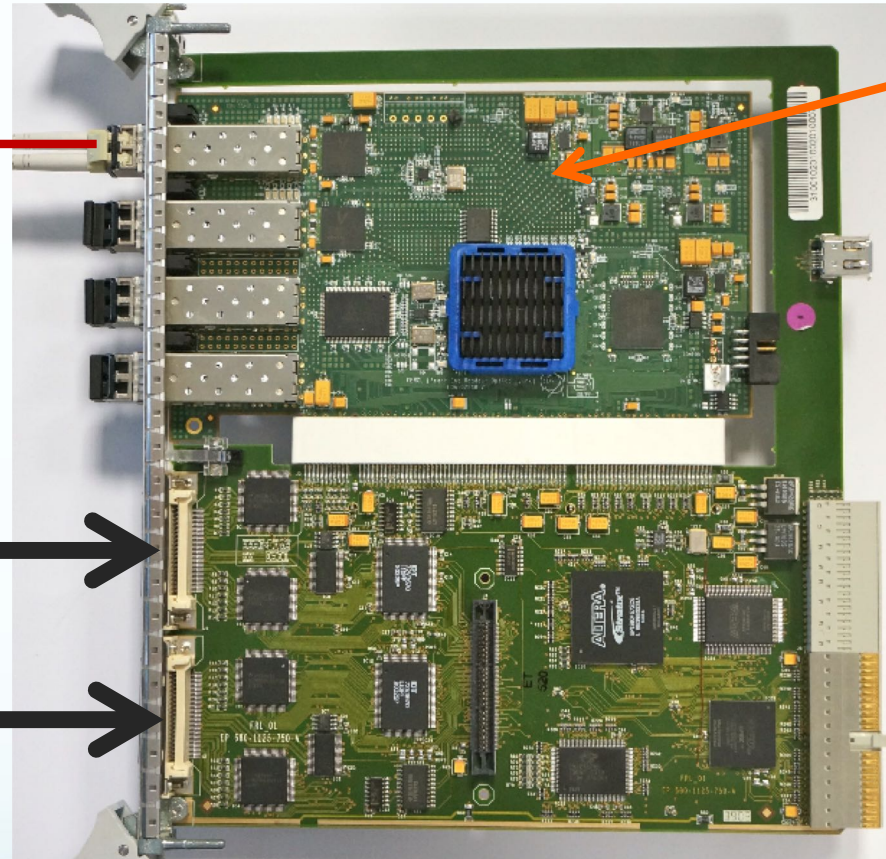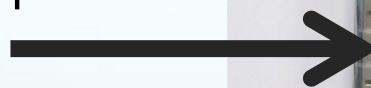# CMS Readout Link for Run-2 in use since 2015

10 Gb/s TCP/IP

Myrinet NIC replaced by custom-built card ("FEROL")

SLINK-64 input
LVDS / copper

Cost effective solution
(need many boards)
Rather inexpensive FPGA
+ commercial chip to combine
3 Gb/s links to 10 Gb/s

FEROL (Front End Readout Optical Link)
Input: 1x or 2x SLINK (copper)
           1x or 2x 5Gb/s optical
           1x 10Gb/s optical

Output: 10 Gb/s Ethernet optical
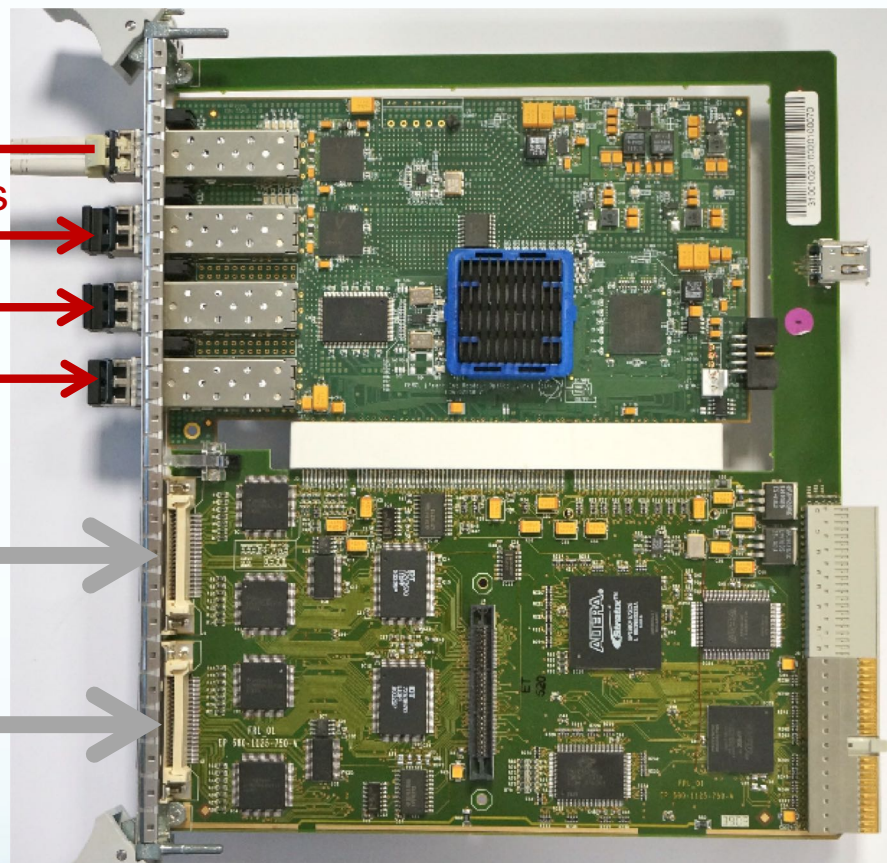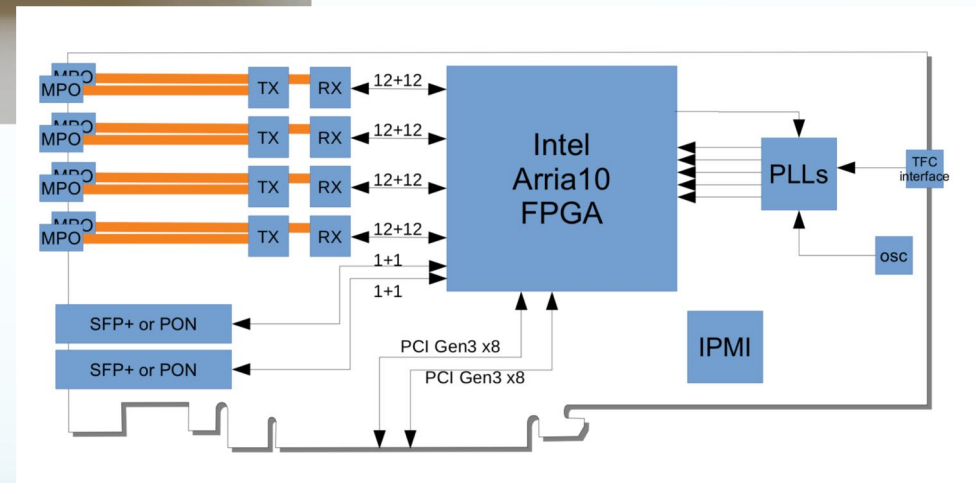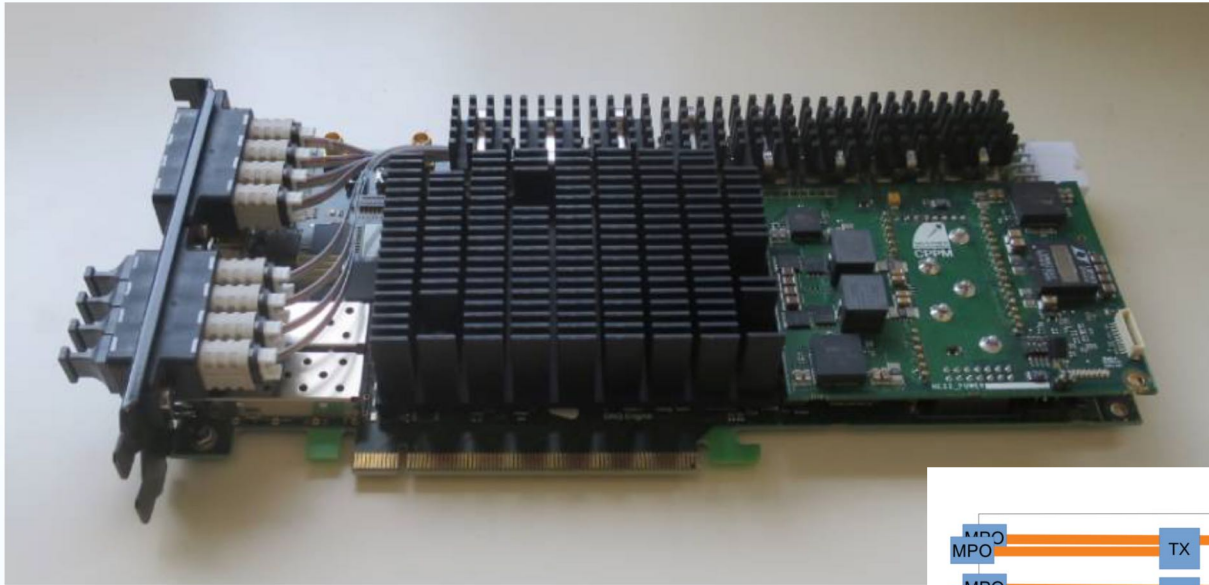   TCP/IP sender in FPGA

CMS Readout Link for Run-2

10 Gb/s TCP/IP

10 Gb/s SLINK Express

5 Gb/s SLINK Express

5 Gb/s SLINK Express

SLINK-64 input
LVDS / copper

FEROL (Front End Readout Optical Link)
Input: 1x or 2x SLINK (copper)   Output: 10 Gb/s Ethernet optical
          1x or 2x 5Gb/s optical   TCP/IP sender in FPGA
          1x 10Gb/s optical

PCIe40 – LHCb and ALICE Run-3

- 48 bidirectional links running at up to 10 Gbits/s each (minipods)
- 2 bidirectional links running at up to 10 Gbits/s devoted to time distribution (can use SFP+ or 10G PON devices)
- Sustained 112 Gbits/s interface with CPU through PCIe

J.P. Cachemiche, ACES 2018

CMS DTH (DAQ and Timing Hub) for HL-LHC

DTH prototype 1

- ATCA board using Xilinx Virtex Ultrascale + FPGAs

- Several DAQ units per board
  - Each unit receiving optical inputs at 16 Gb/s and 25 Gb/s
  - 4x 100 Gb/s Ethernet to commercial network (4 links of 25Gb/s)
  - TCP/IP in FPGA

- Board distributes timing and control

- Board contains switch for control network

FPGAs in other domains

- Medical imaging

- Advanced Driver Assistance Systems (Image Processing)

- Speech recognition

- Cryptography

- Bioinformatics

- Aerospace / Defense

- (Bitcoin mining)

- 5G Wireless

- ASIC Prototyping
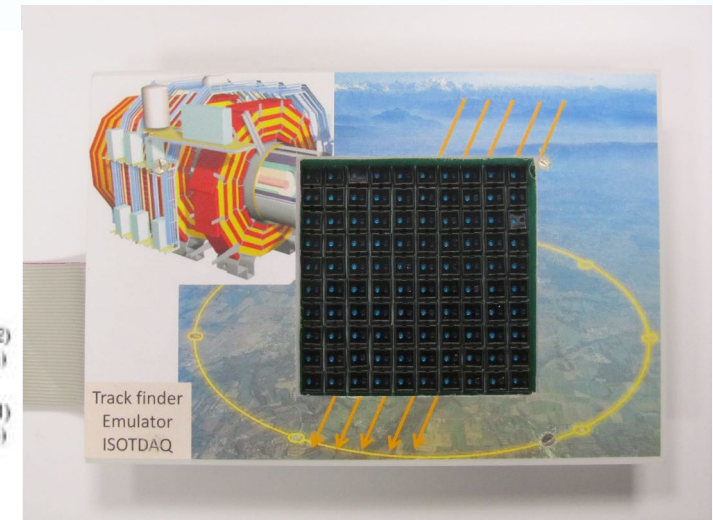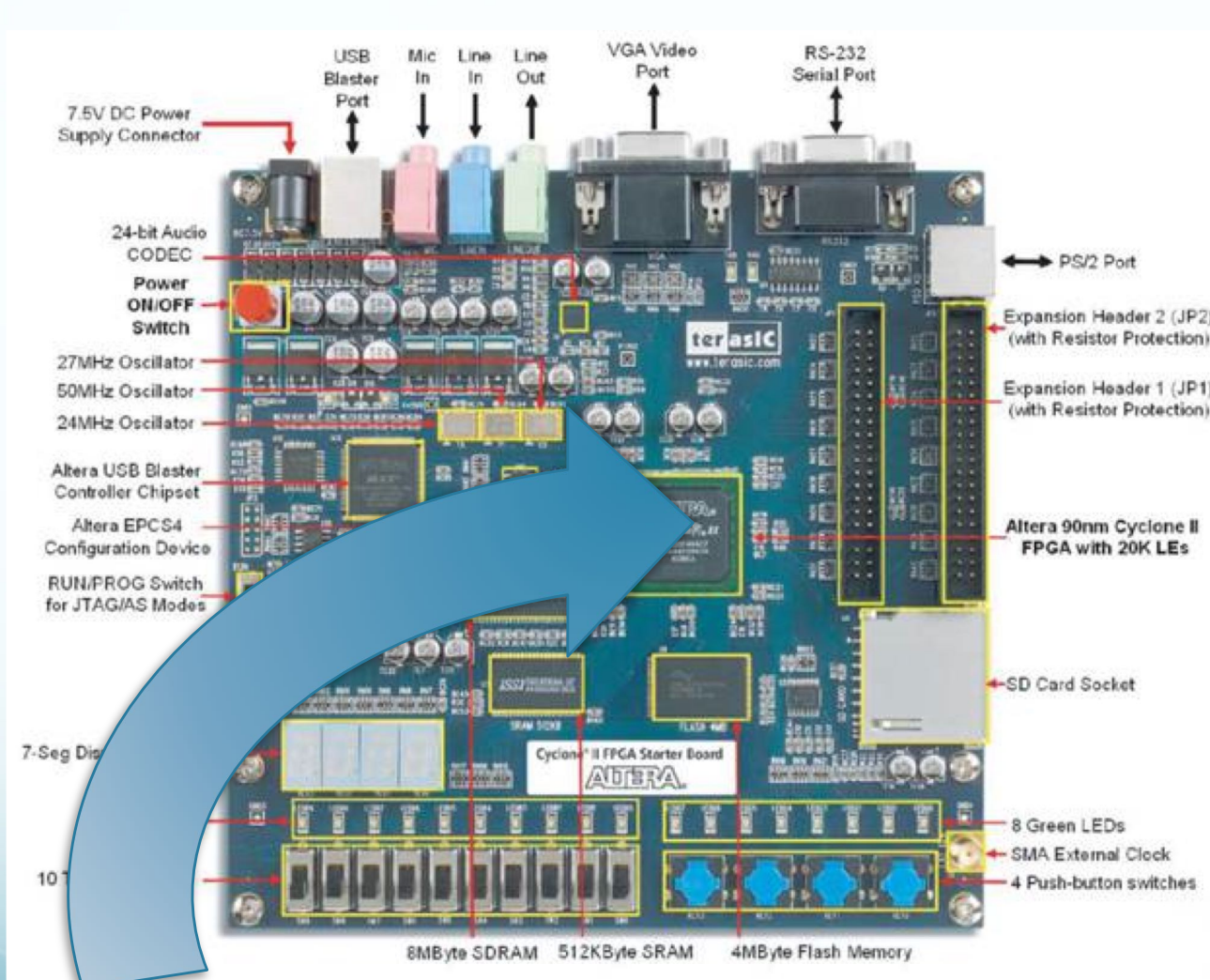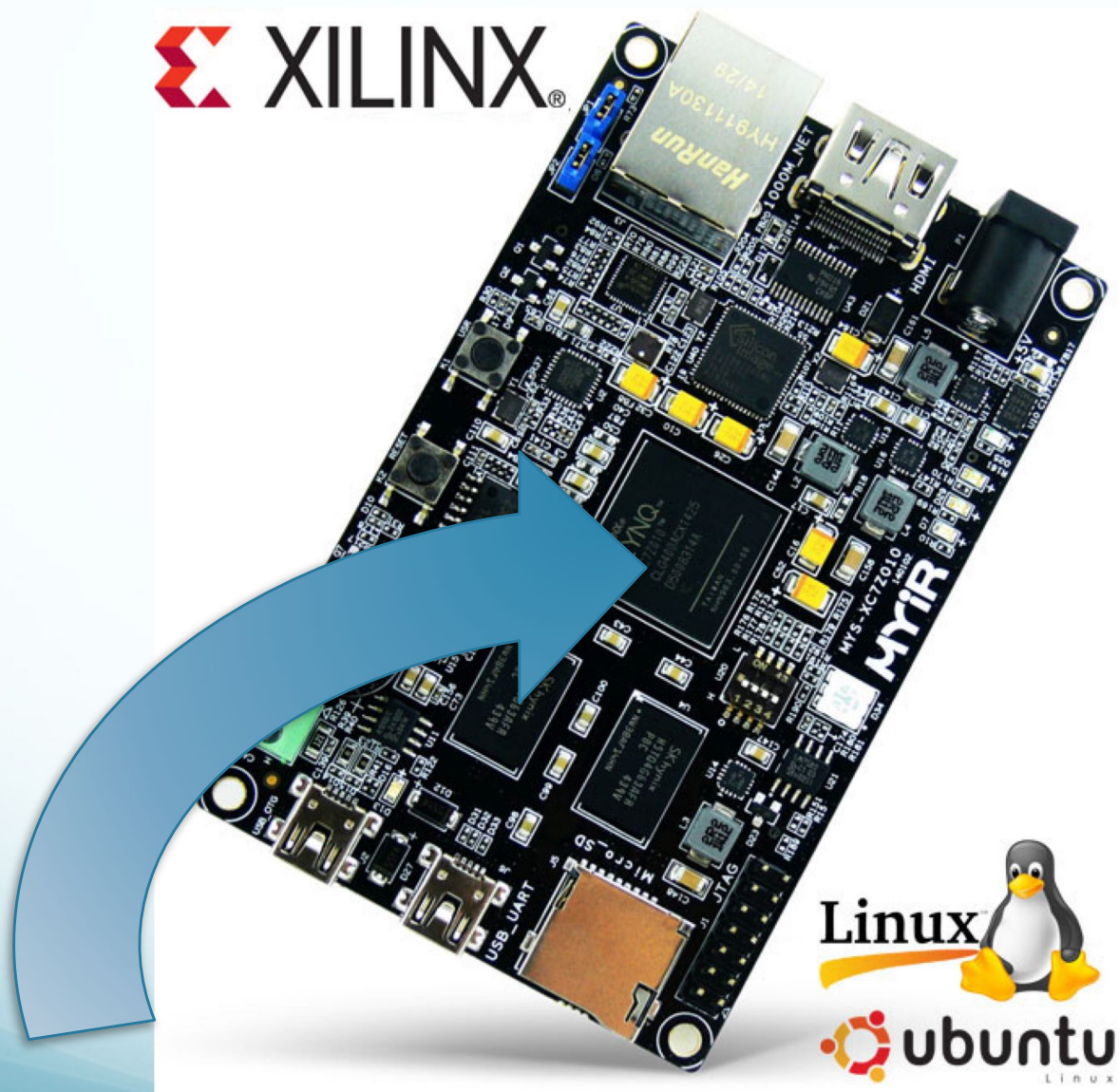
- Compute accelerators
  - Accelerator cards

3 TFlop

  - Server processors w. FPGA
  - Financial
  - Inferencing
  - Video transcoding
  - …

Lab Session 5: Programming an FPGA

You are going to design the digital electronics inside this FPGA !

Lab Session 13: System-on-a-chip FPGA

Z-turn board
Zynq w. dual-core ARM

Design the digital electronics and software in this SoC FPGA!