



Introduction to Field Programmable Gate Arrays

Hannes Sakulin
CERN / EP-CMD

ESIPAP school,
Archamps, France, 27th February, 2018



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!

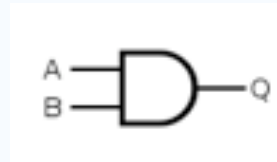
Re-use

- 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

AND gate



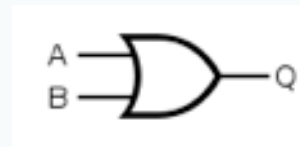
Truth table

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

C equivalent

`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

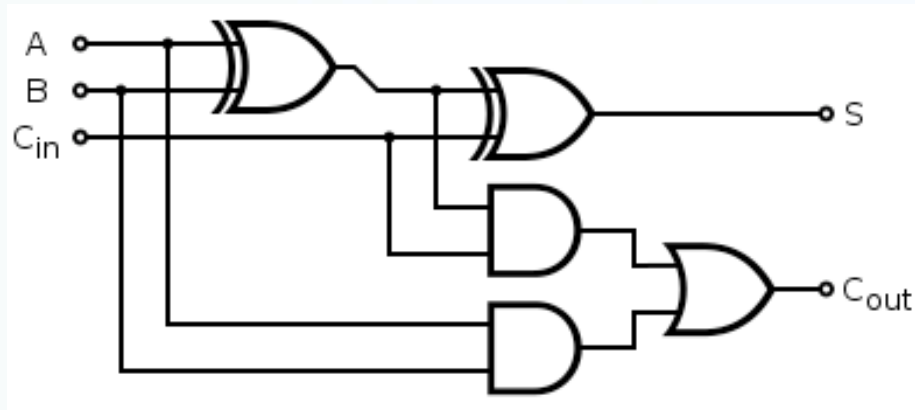


INPUT		OUTPUT
A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

`q = a != b;`

⋮

Combinatorial logic (asynchronous)



Outputs are determined by Inputs, only

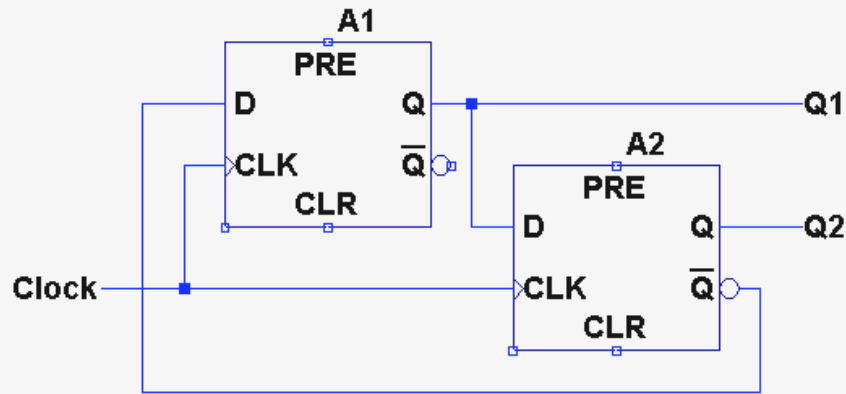
Example: Full adder with carry-in, carry-out

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)

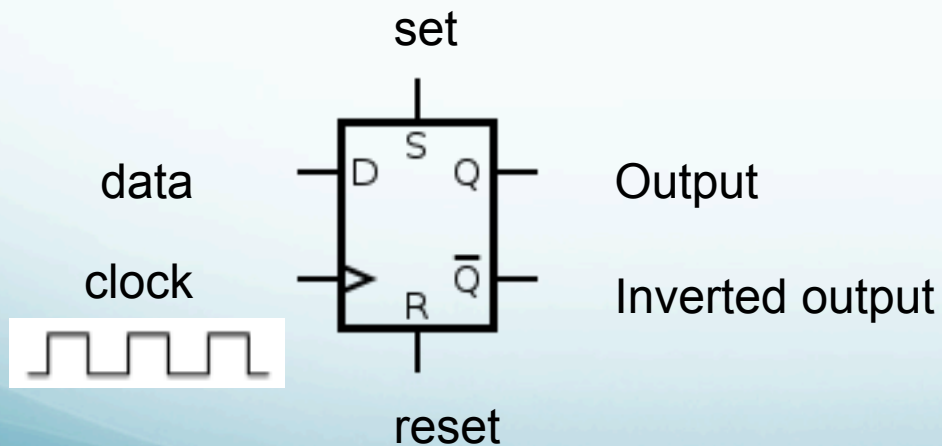
LUT = small memory

(Synchronous) sequential logic



2-bit binary counter

Outputs are determined by Inputs and their History (Sequence)
The logic has an internal state



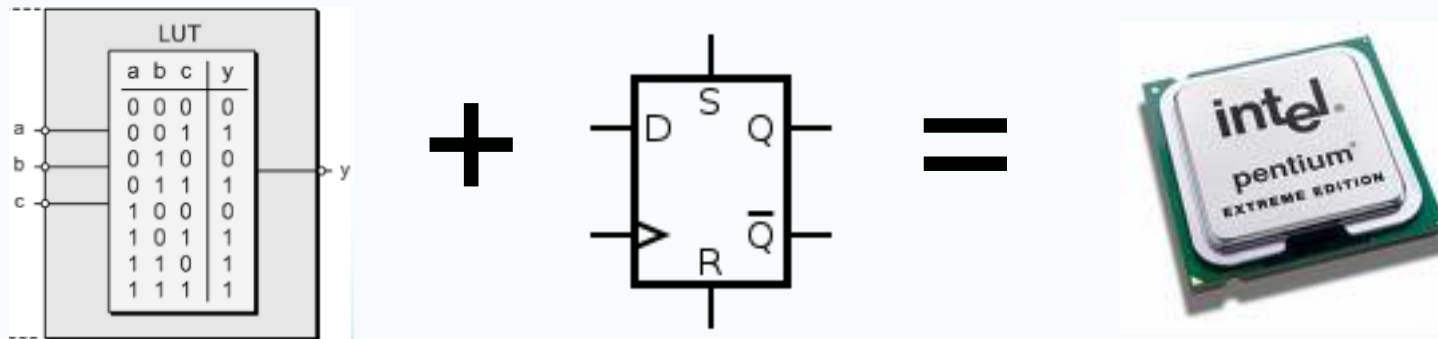
D Flip-flop:

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

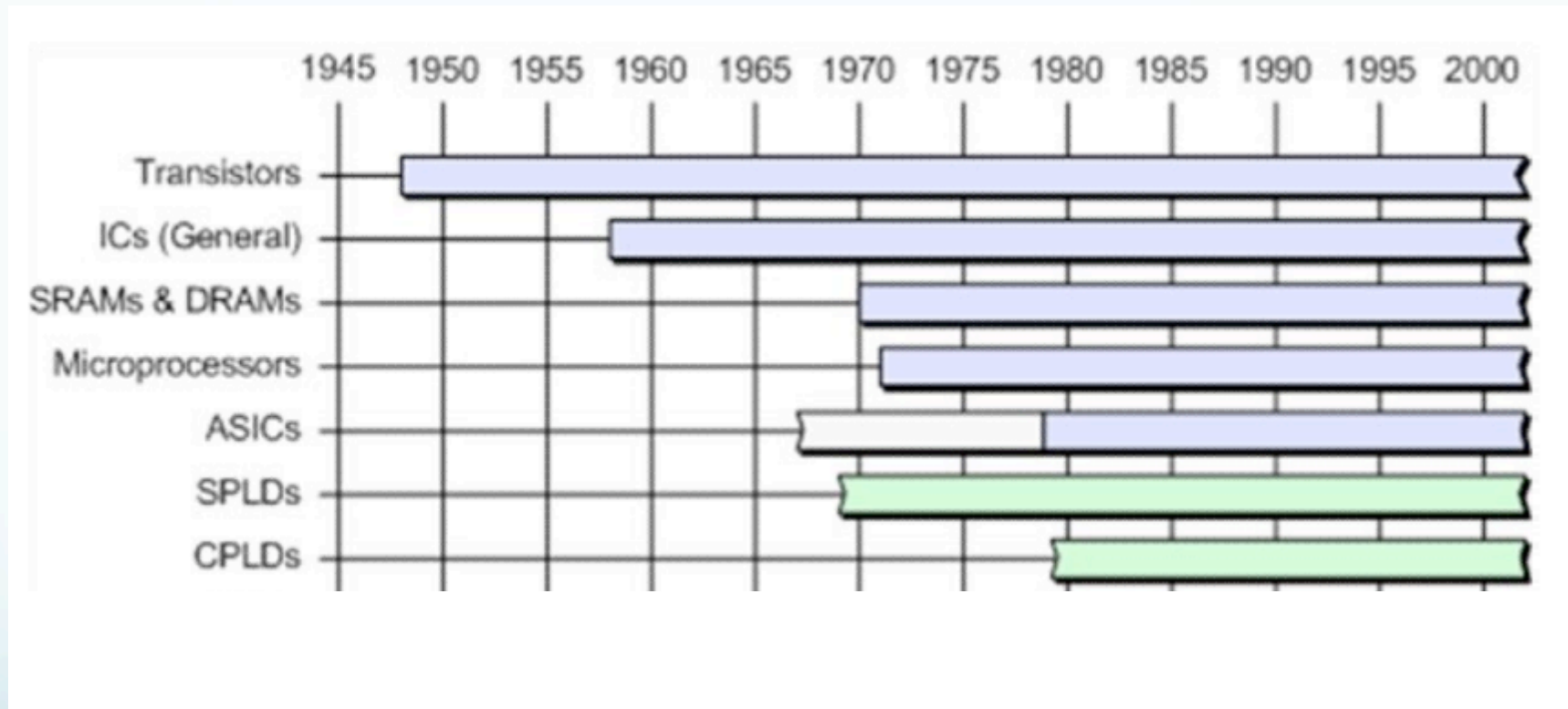


Using Look-Up-Tables and Flip-Flops
any kind of digital electronics may be implemented

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

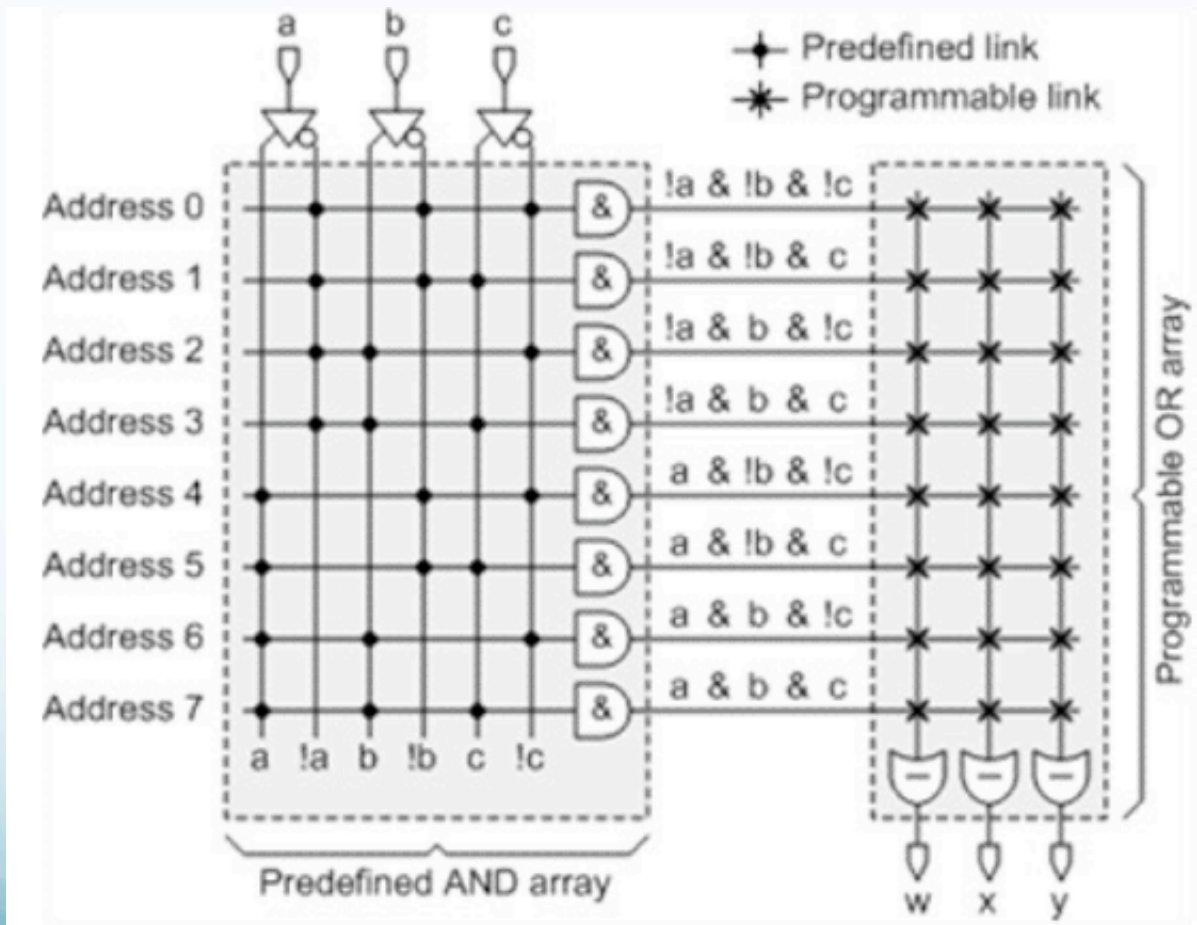
Programmable digital electronics

Long long time ago ...



Simple Programmable Logic Devices (sPLDs)

a) Programmable Read Only Memory (PROMs)

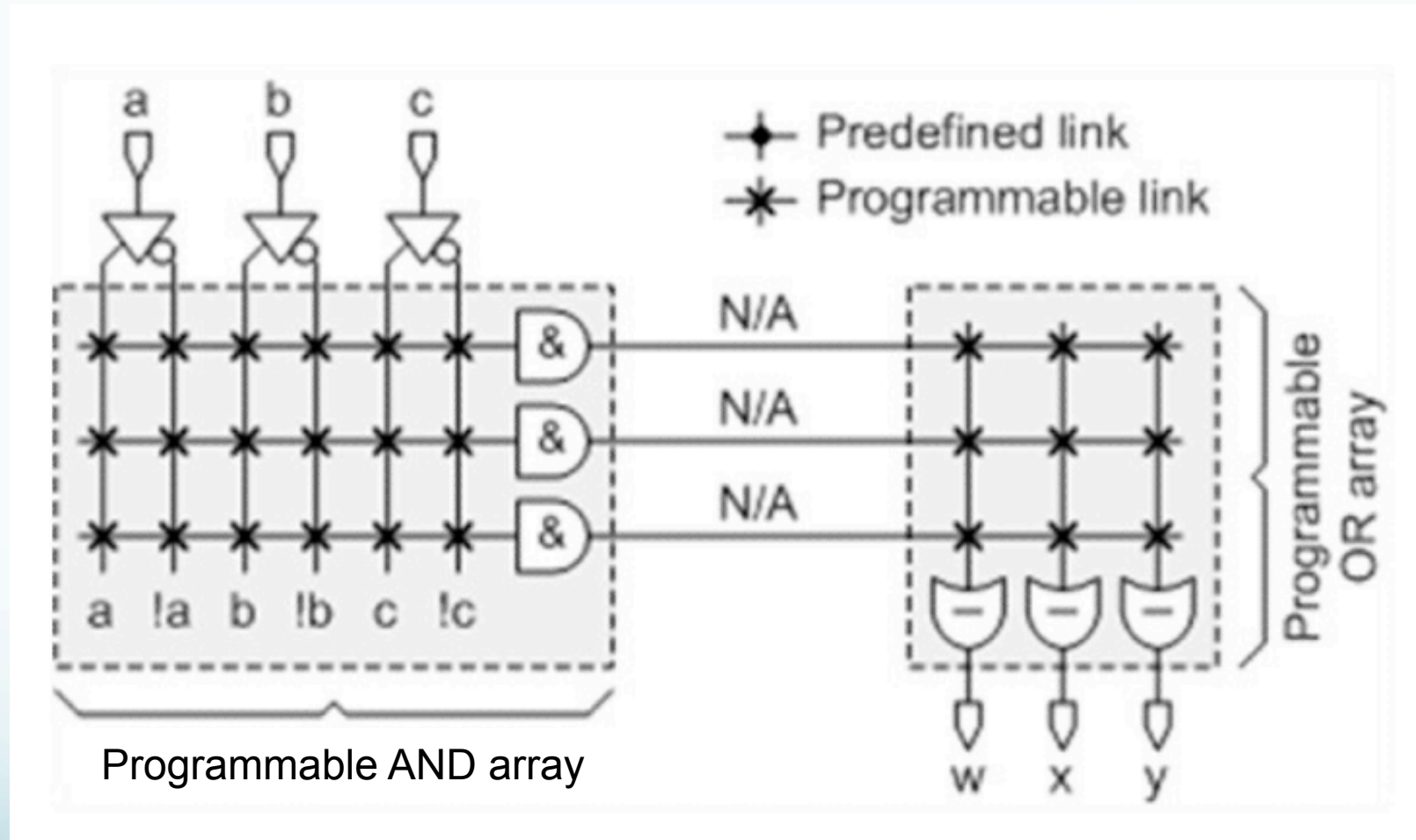


Late 60's

Unprogrammed PROM (Fixed AND Array, Programmable OR Array)

Simple Programmable Logic Devices (sPLDs)

b) Programmable Logic Arrays (PLAs)



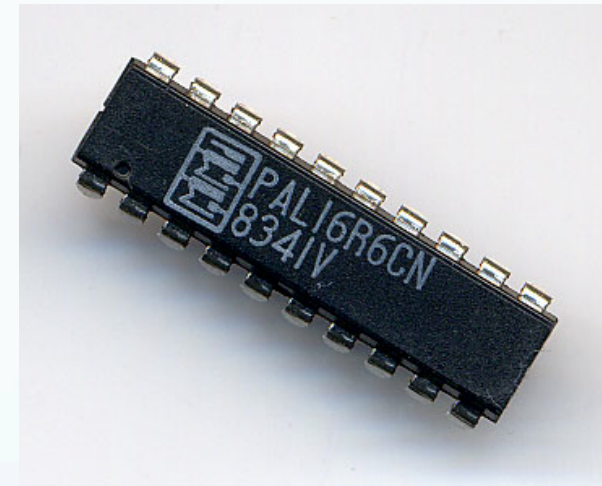
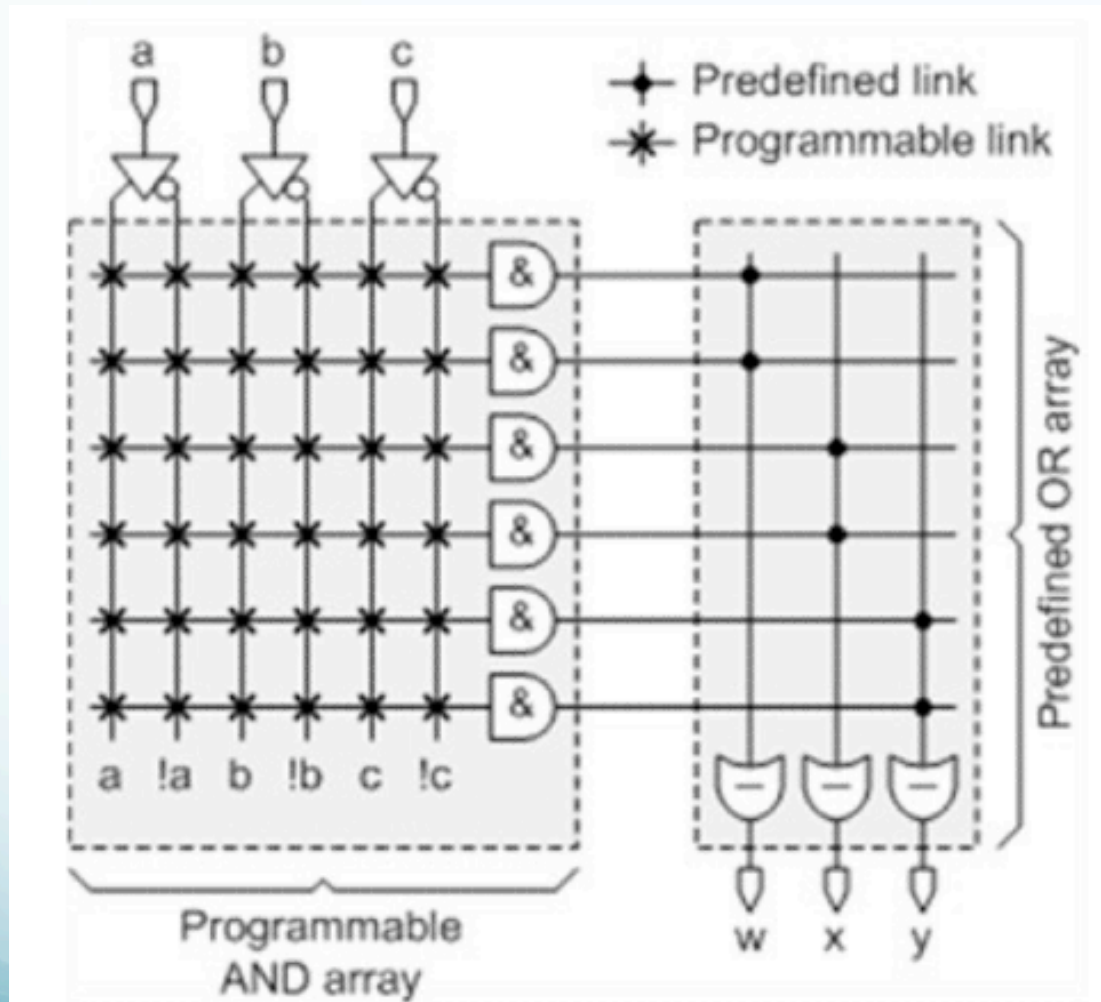
Unprogrammed PLA (Programmable AND and OR Arrays)

Most flexible
but slower

1975

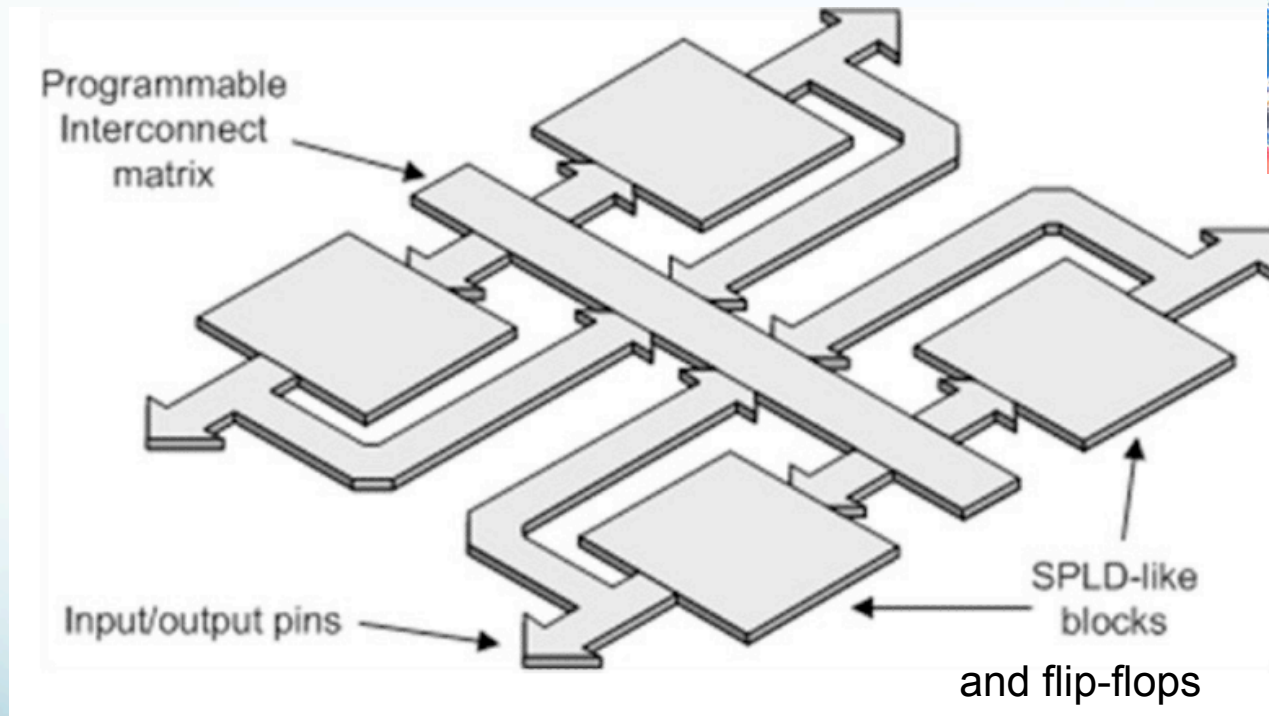
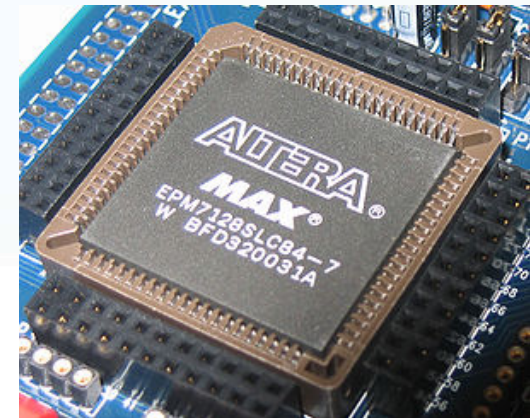
Simple Programmable Logic Devices (sPLDs)

c) Programmable Array Logic (PAL)



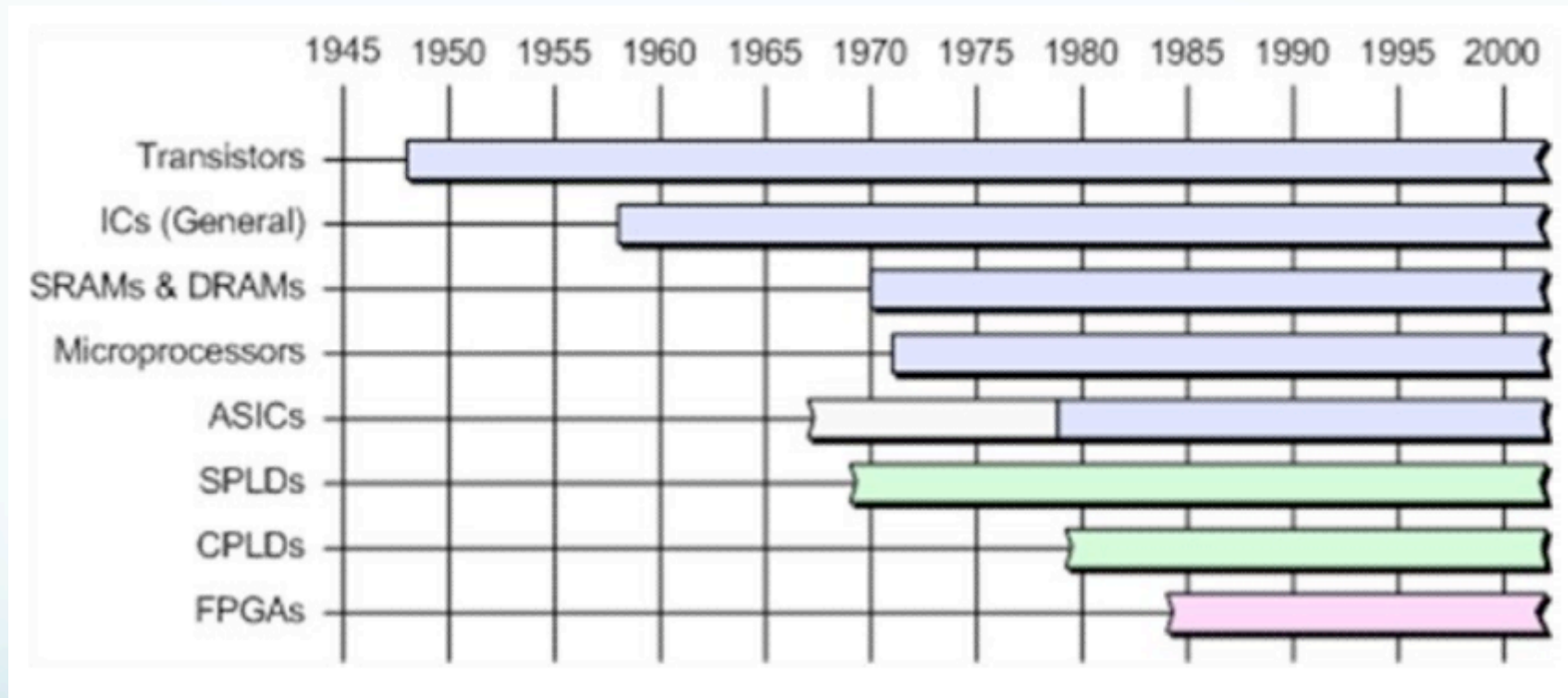
Unprogrammed PAL (Programmable AND Array, Fixed OR Array)

Complex PLDs (CPLDs)

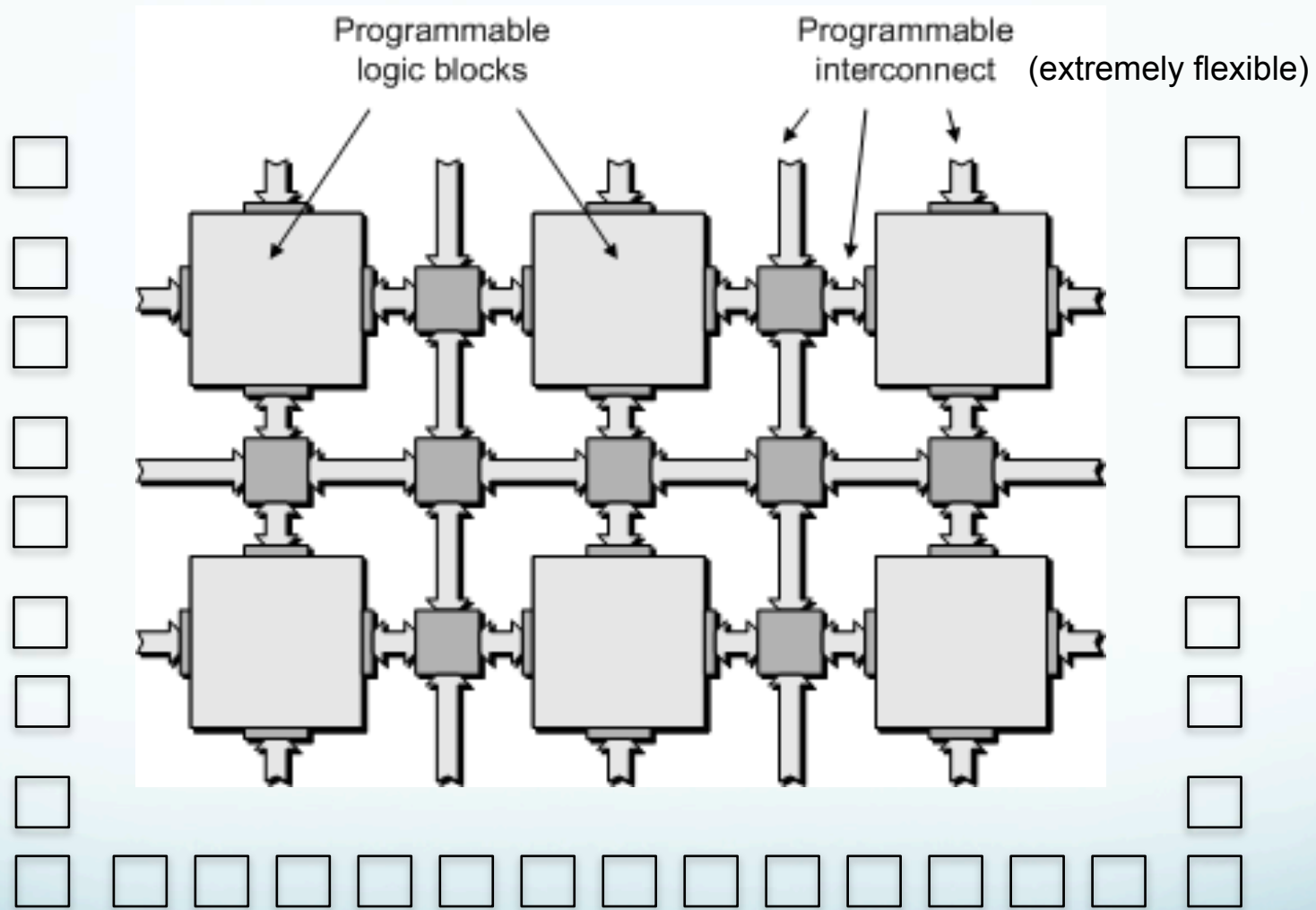


Coarse grained
100's of blocks, restrictive structure
(EE)PROM based

FPGAs ...



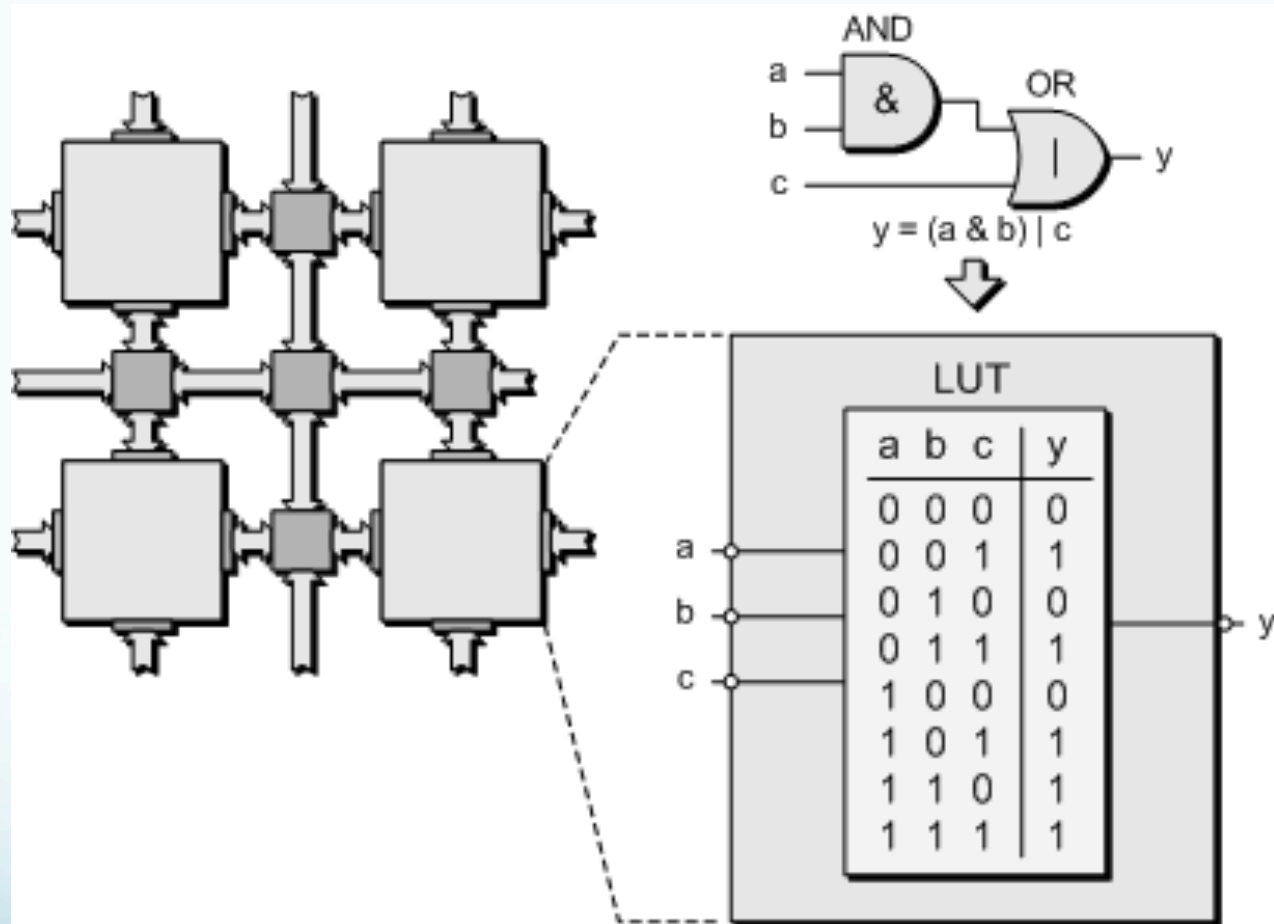
FPGAs



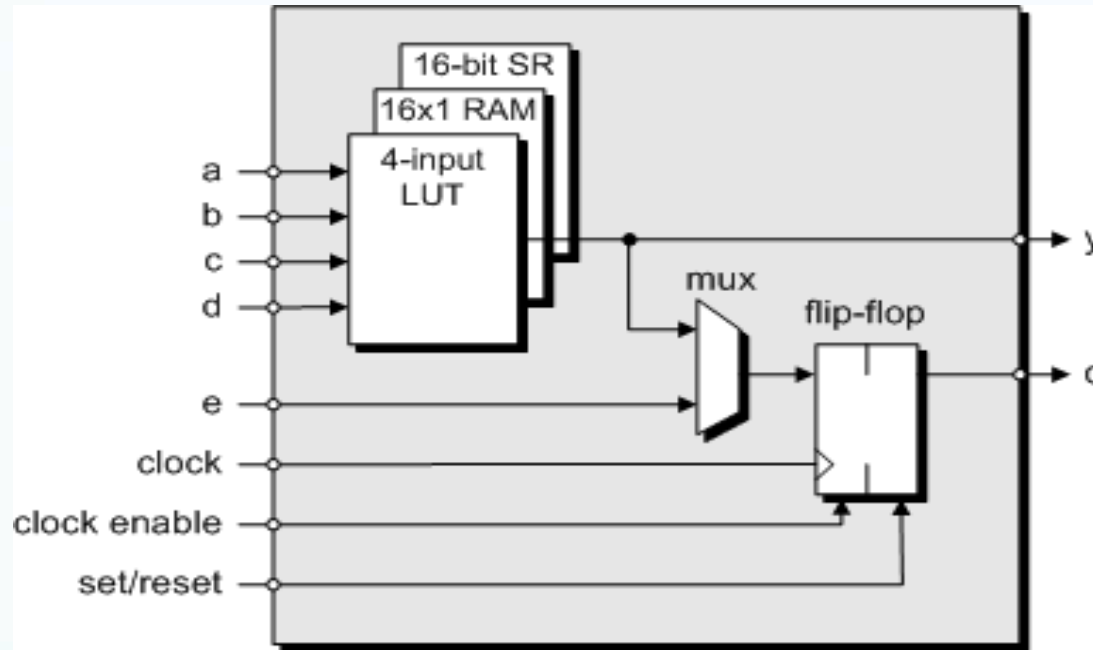
Fine-grained: 100.000's of blocks
today: up to 5 million logic blocks

Programmable Input / Output pins

LUT-based Fabrics



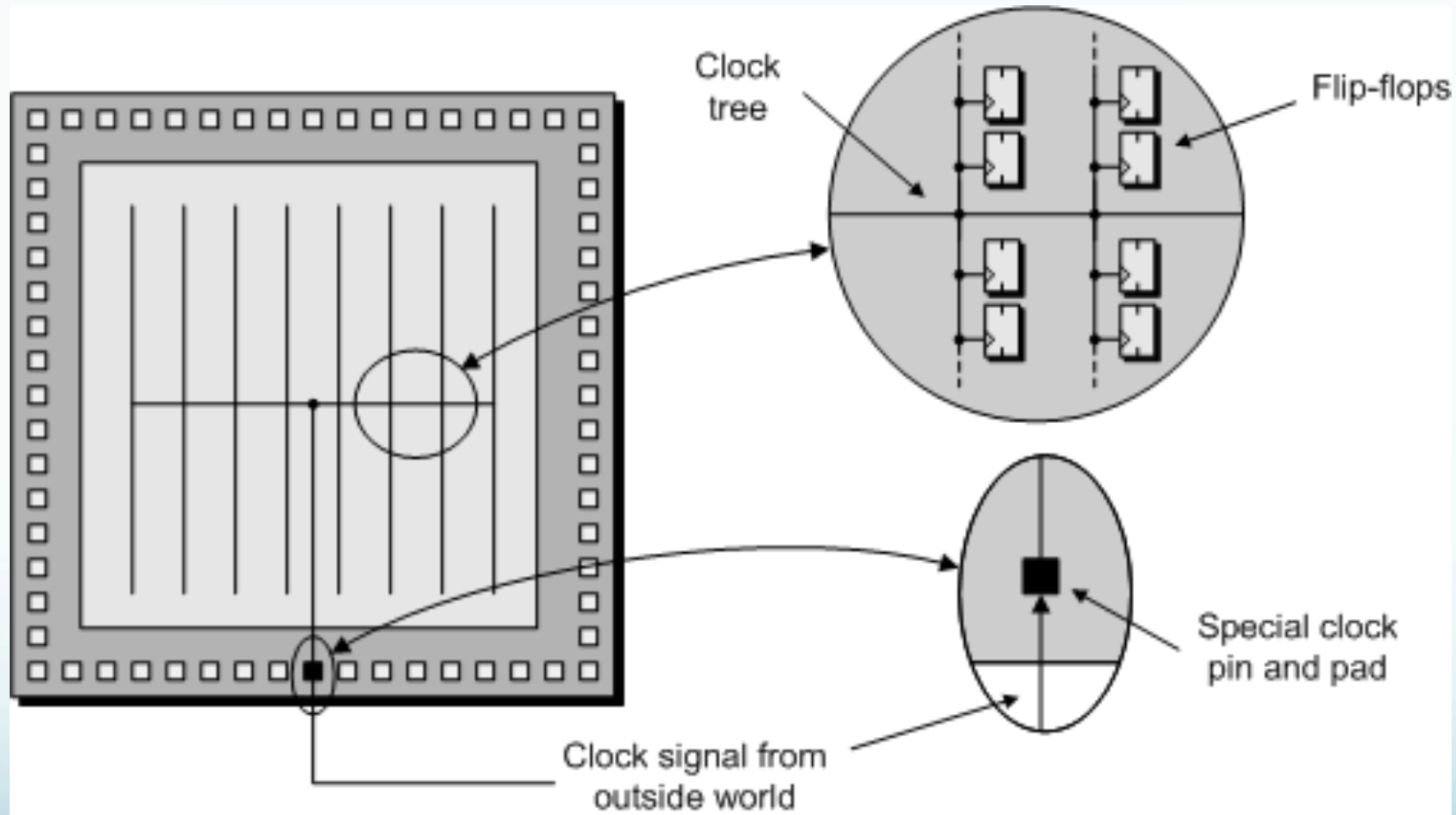
Typical LUT-based Logic Cell



Xilinx: logic cell,
Altera: logic element

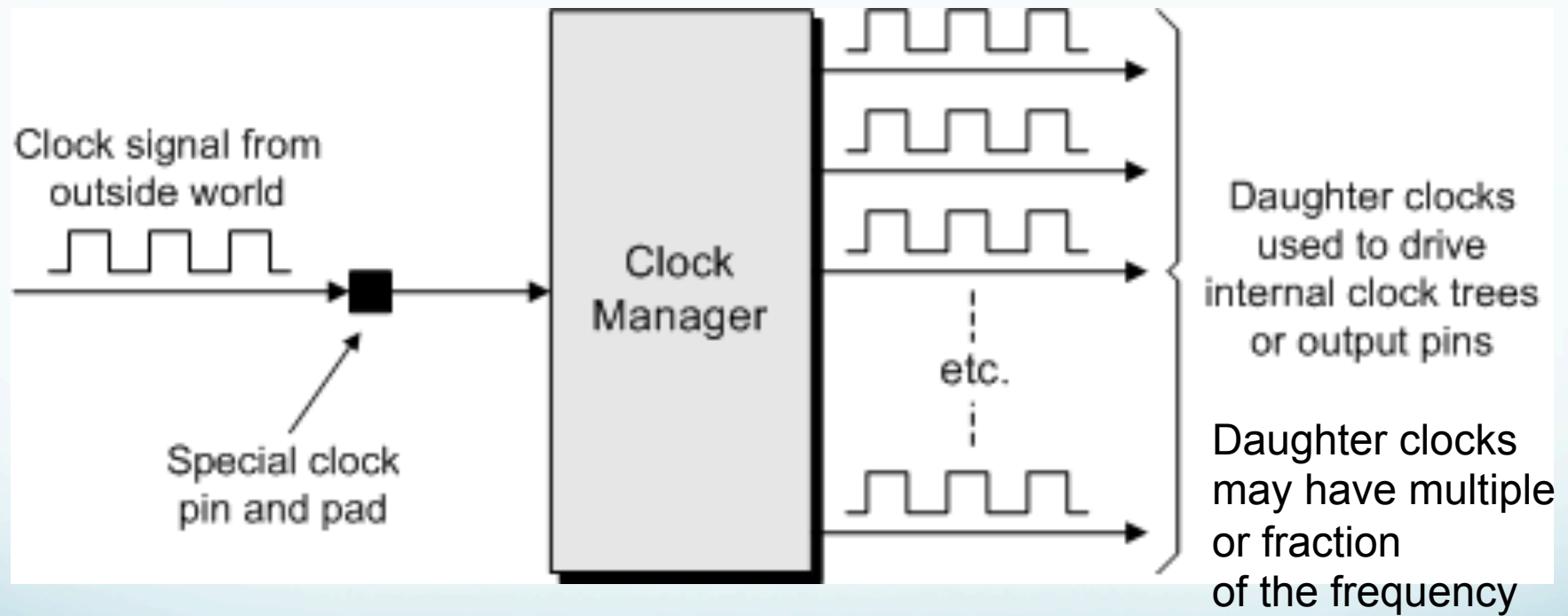
- LUT may implement any function of the inputs
- Flip-Flop registers the LUT output
- May use only the LUT or only the Flip-flop
- LUT may alternatively be configured a shift register
- Additional elements (not shown): fast carry logic

Clock Trees

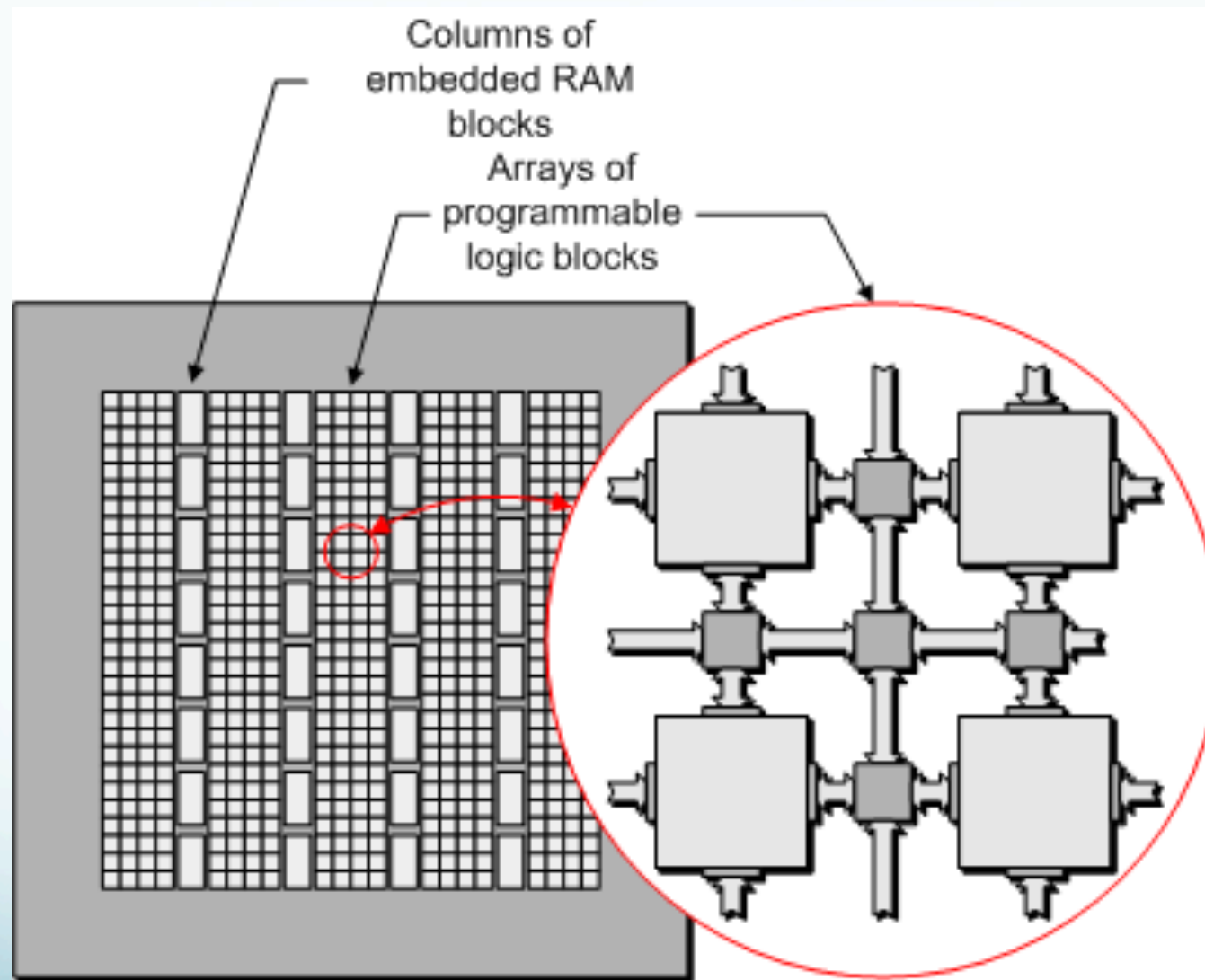


Clock trees guarantee that the clock arrives at the same time at all flip-flops

Clock Managers

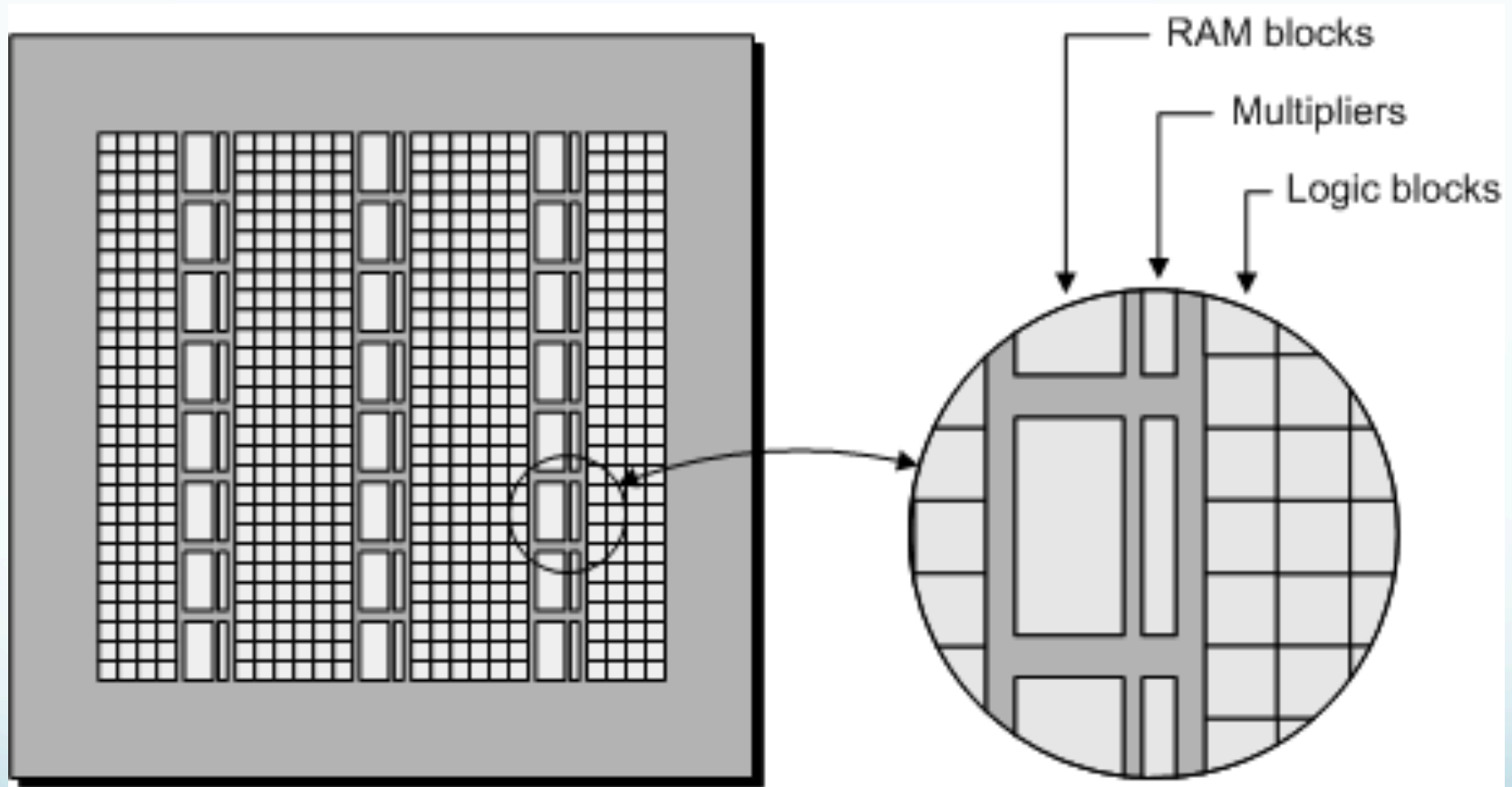


Embedded RAM blocks

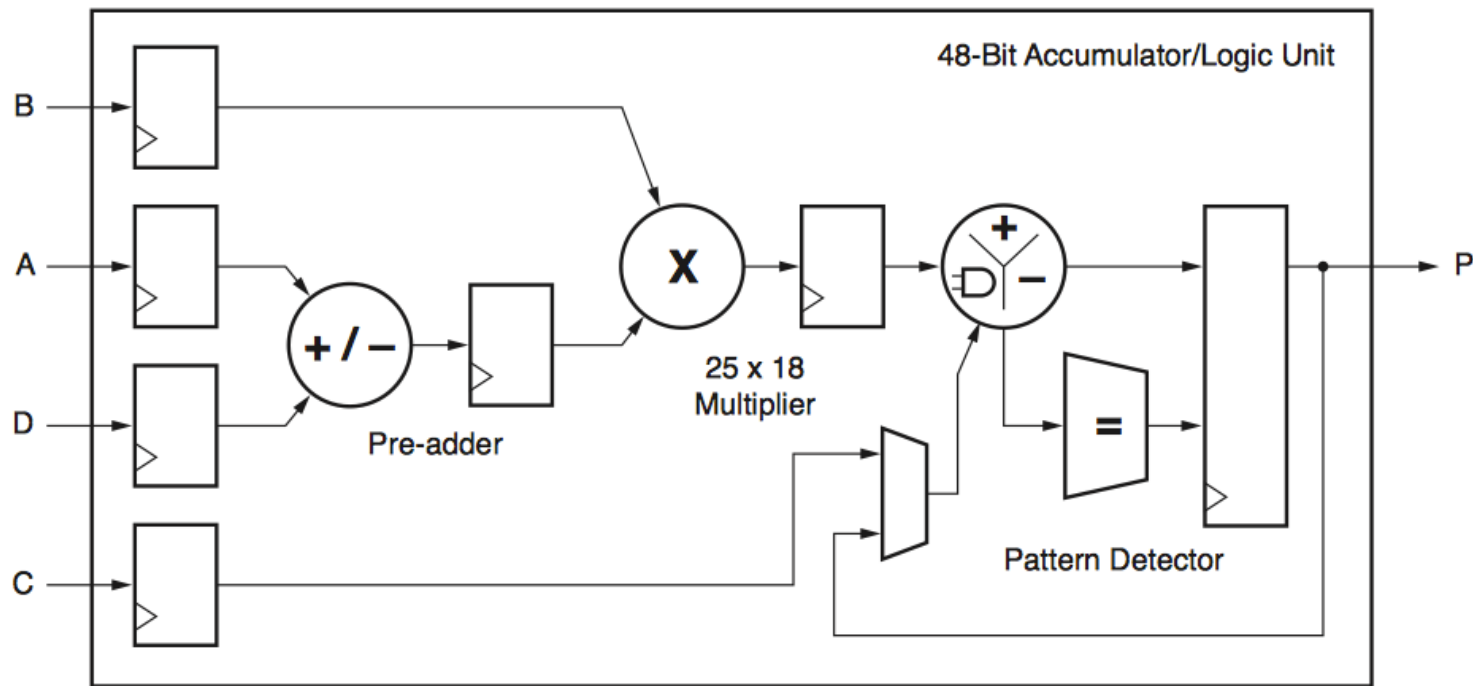


Today: Up to ~500 Mbit of RAM

Embedded Multipliers & DSPs



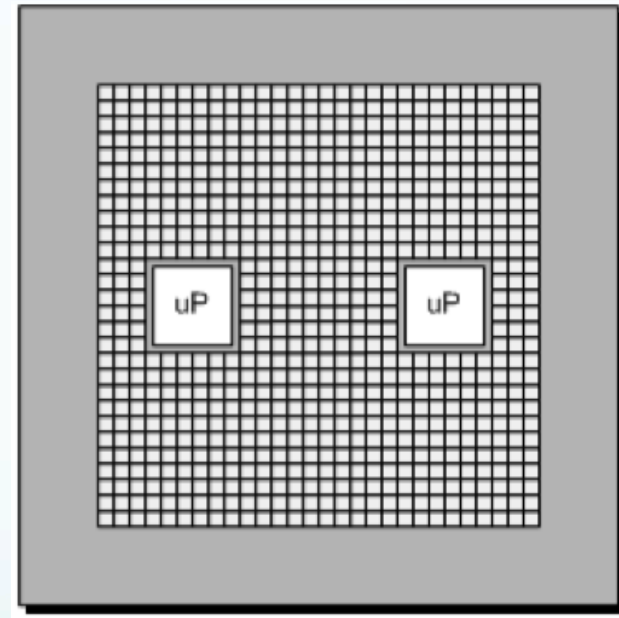
Digital Signal Processor (DSP)



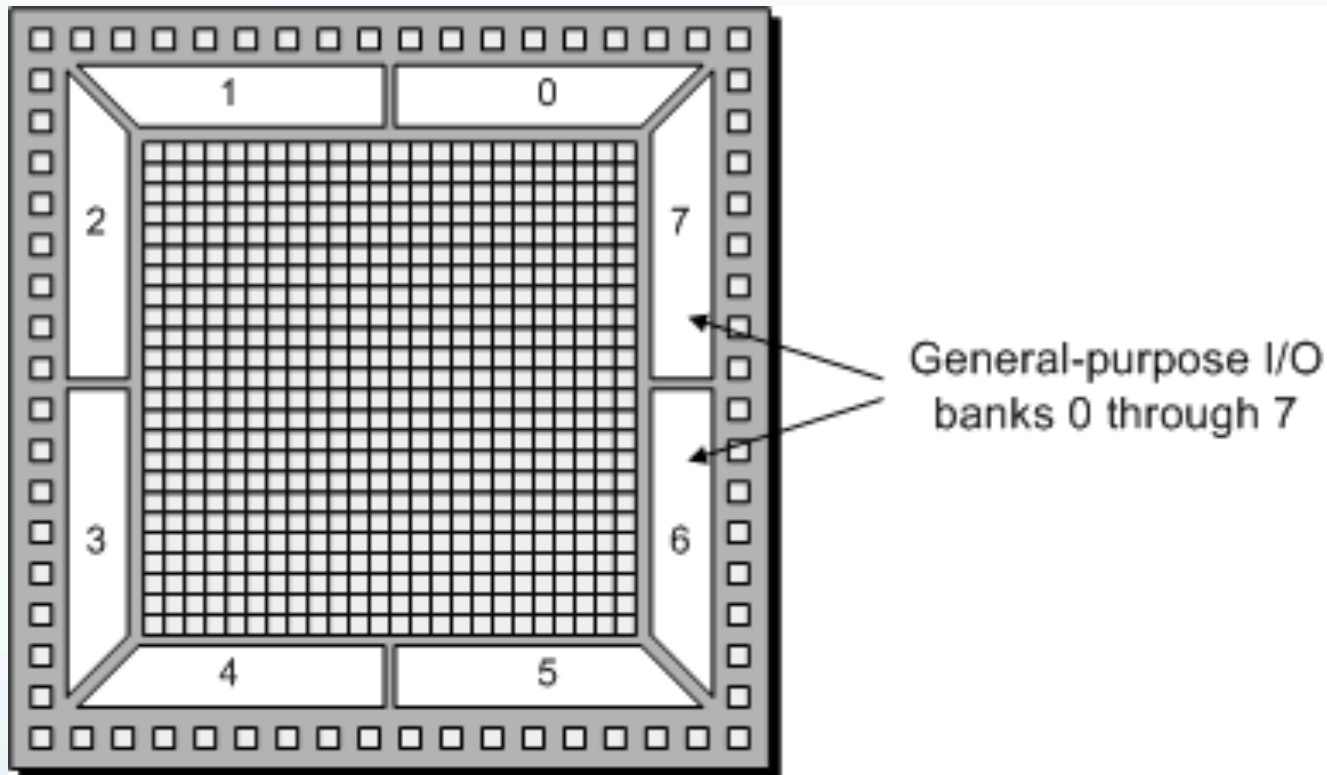
DSP block (Xilinx 7-series)
Up to several 1000 per chip

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



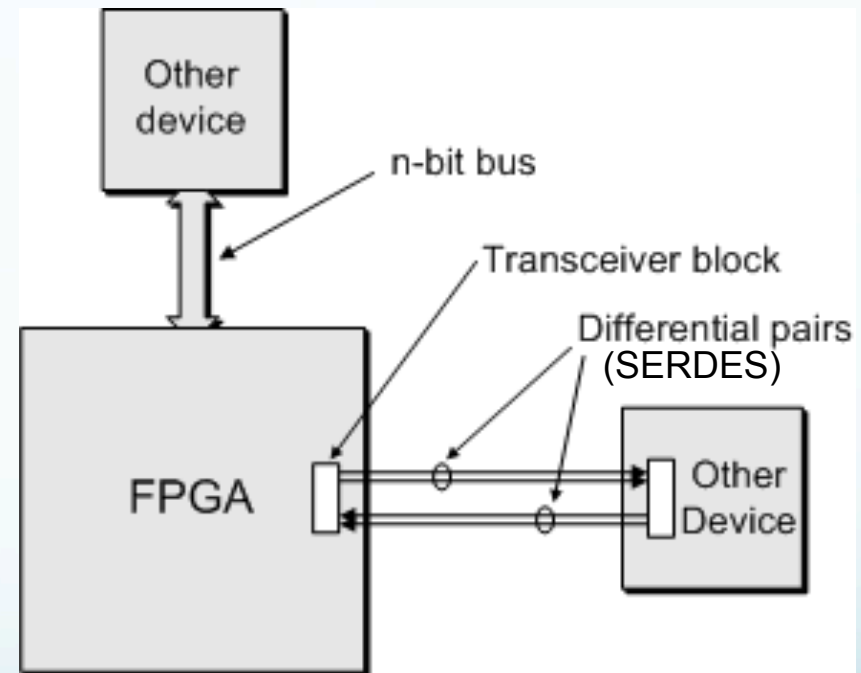
General-Purpose Input/Output (GPIO)



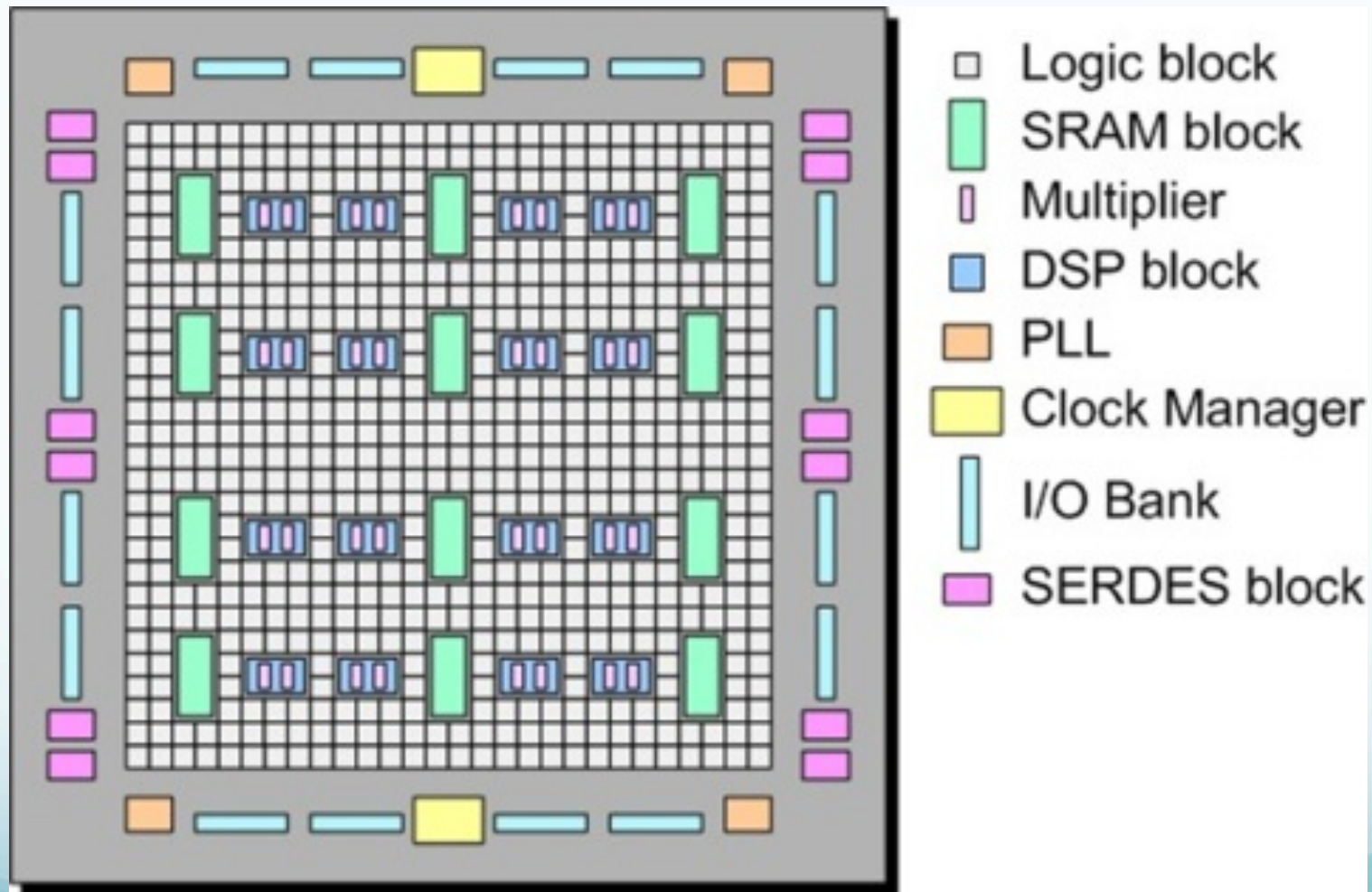
Today: Up to 1200 user I/O pins
Input and / or output
Voltages from (1.0), 1.2 .. 3.3 V
Many IO standards
Single-ended: LVTTL, LVCMOS, ...
Differential pairs: LVDS, ...

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

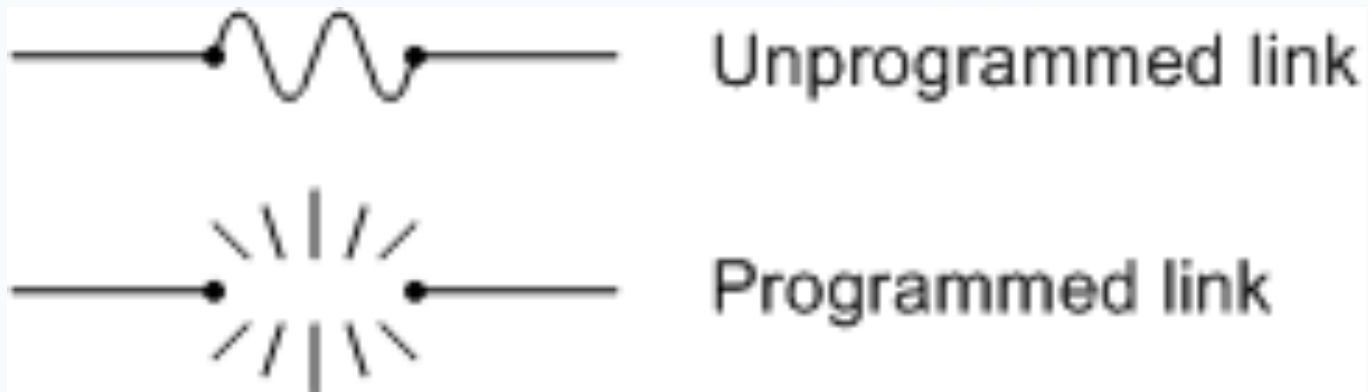


Components in a modern FPGA

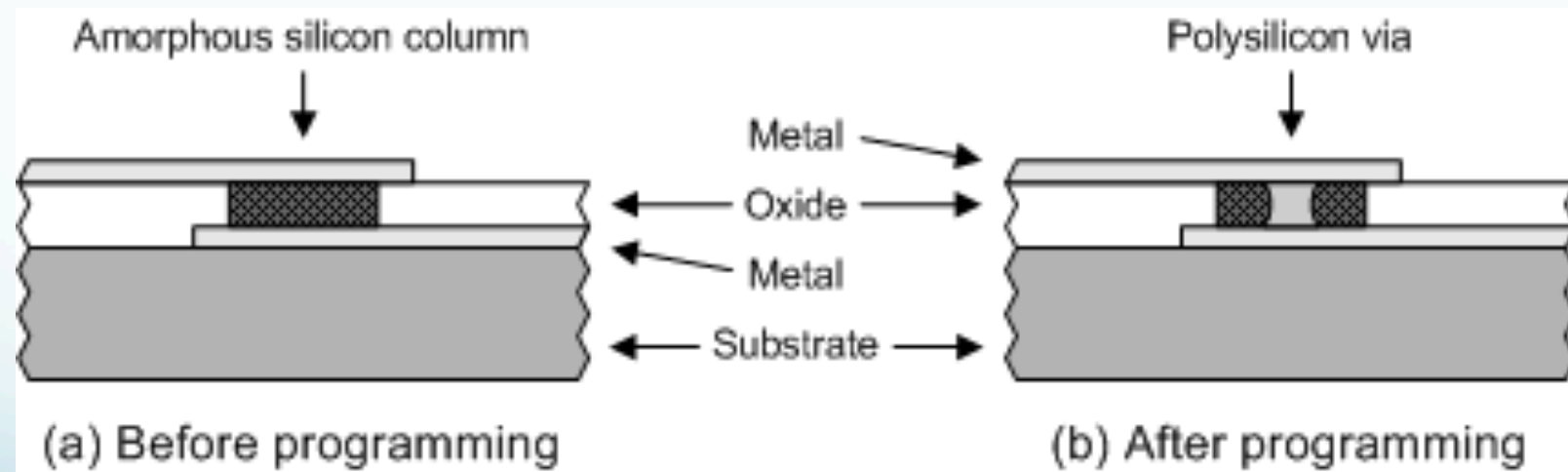
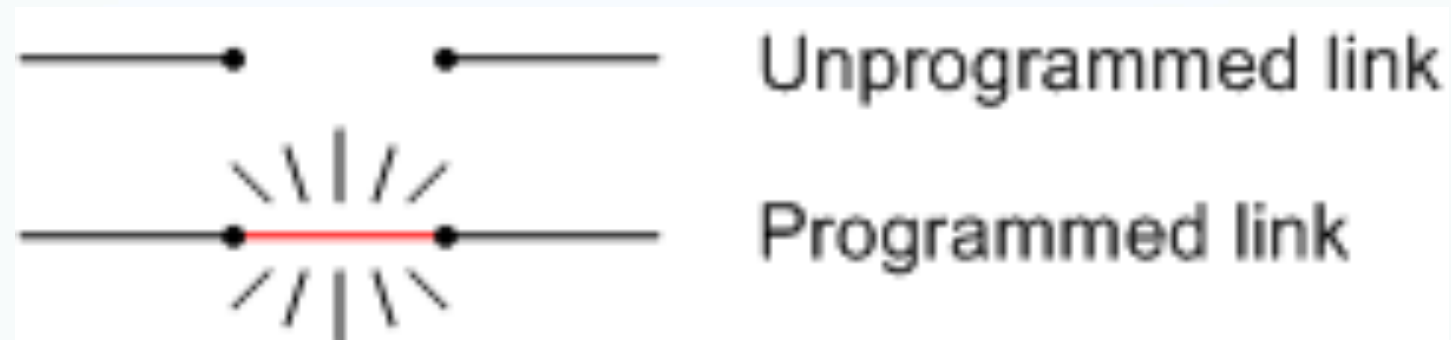


Programming techniques

Fusible Links (not used in FPGAs)

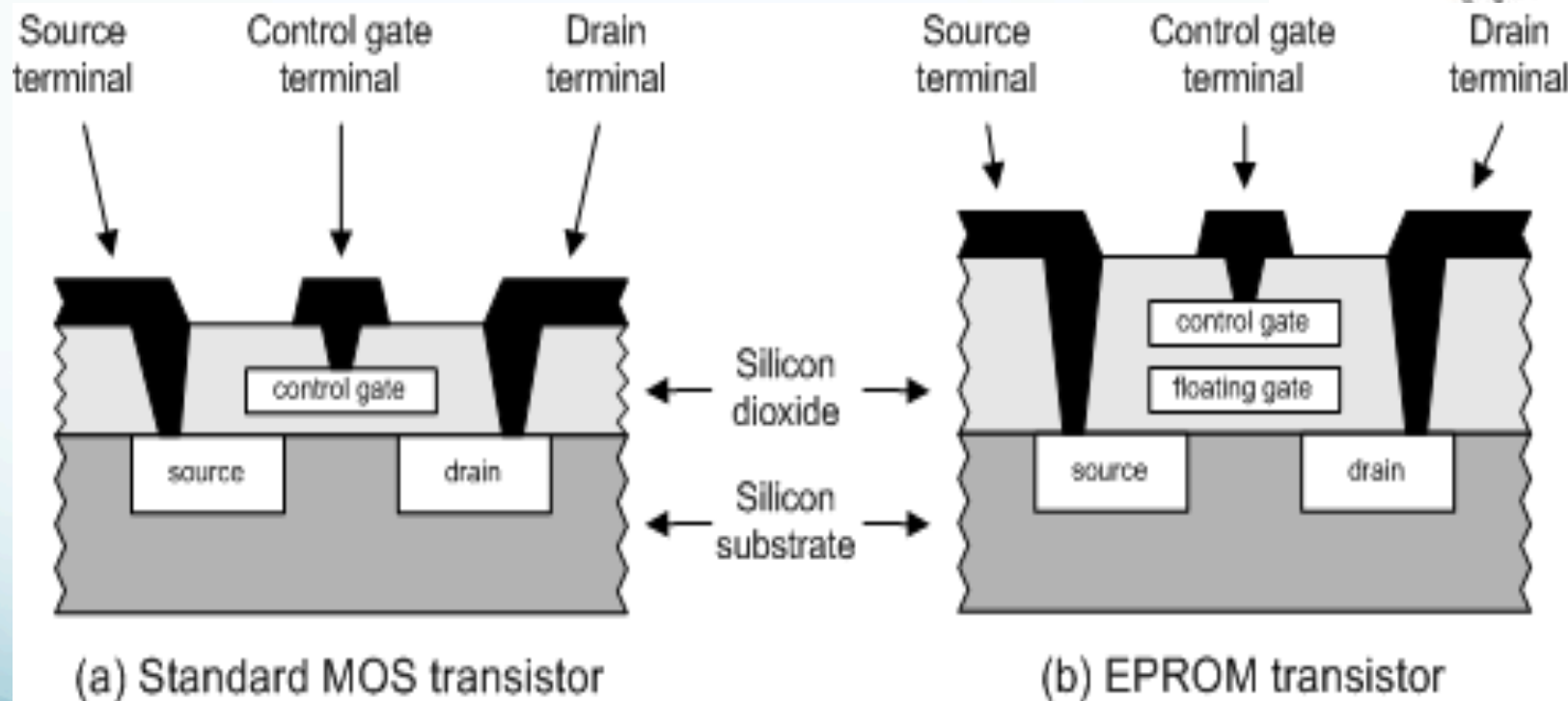


Antifuse Technology



EPROM Technology

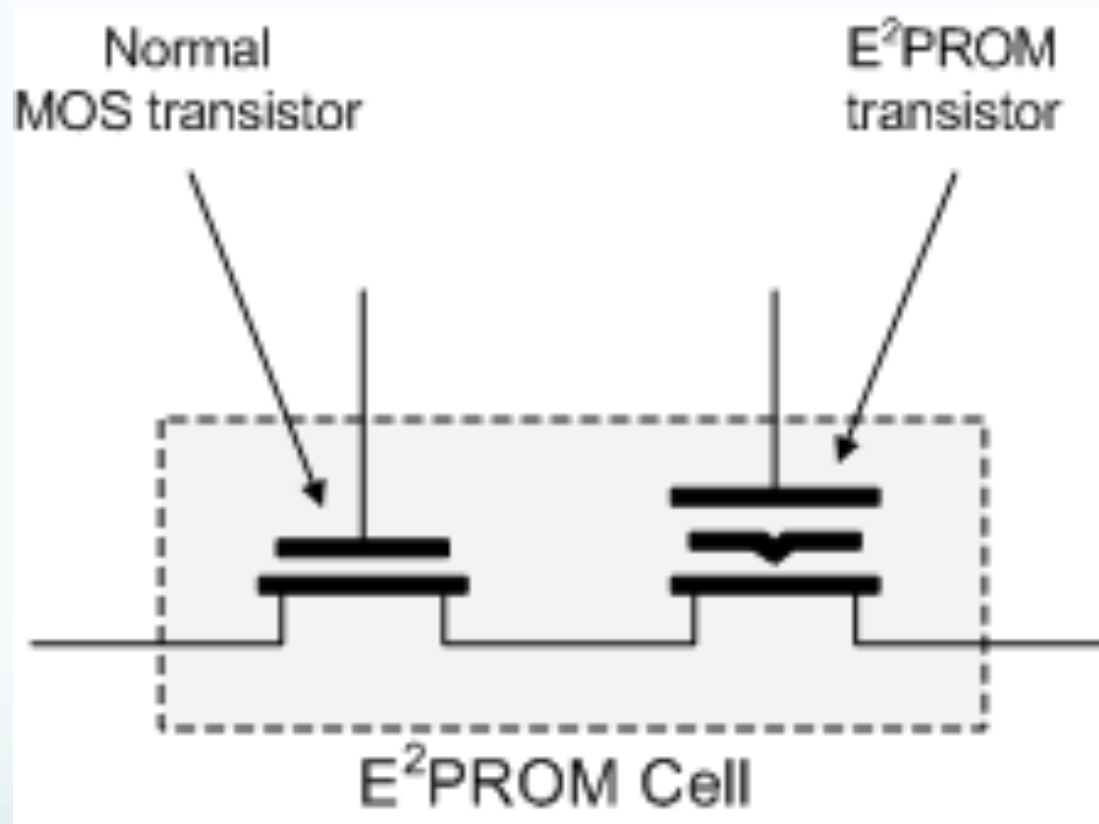
Erasable Programmable Read Only Memory



Intel, 1971

EEPROM and FLASH Technology

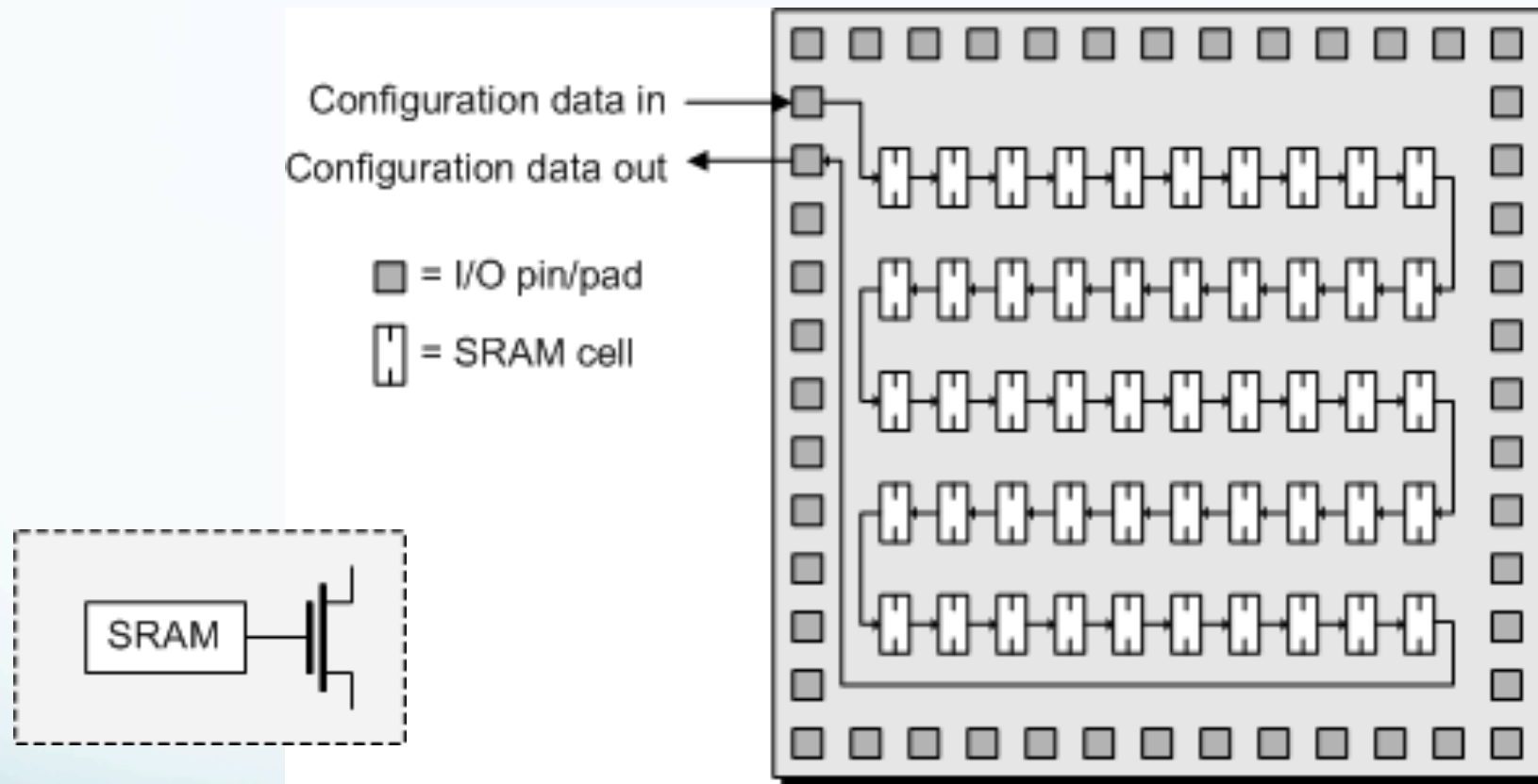
Electrically Erasable Programmable Read Only Memory



EEPROM: erasable word by word

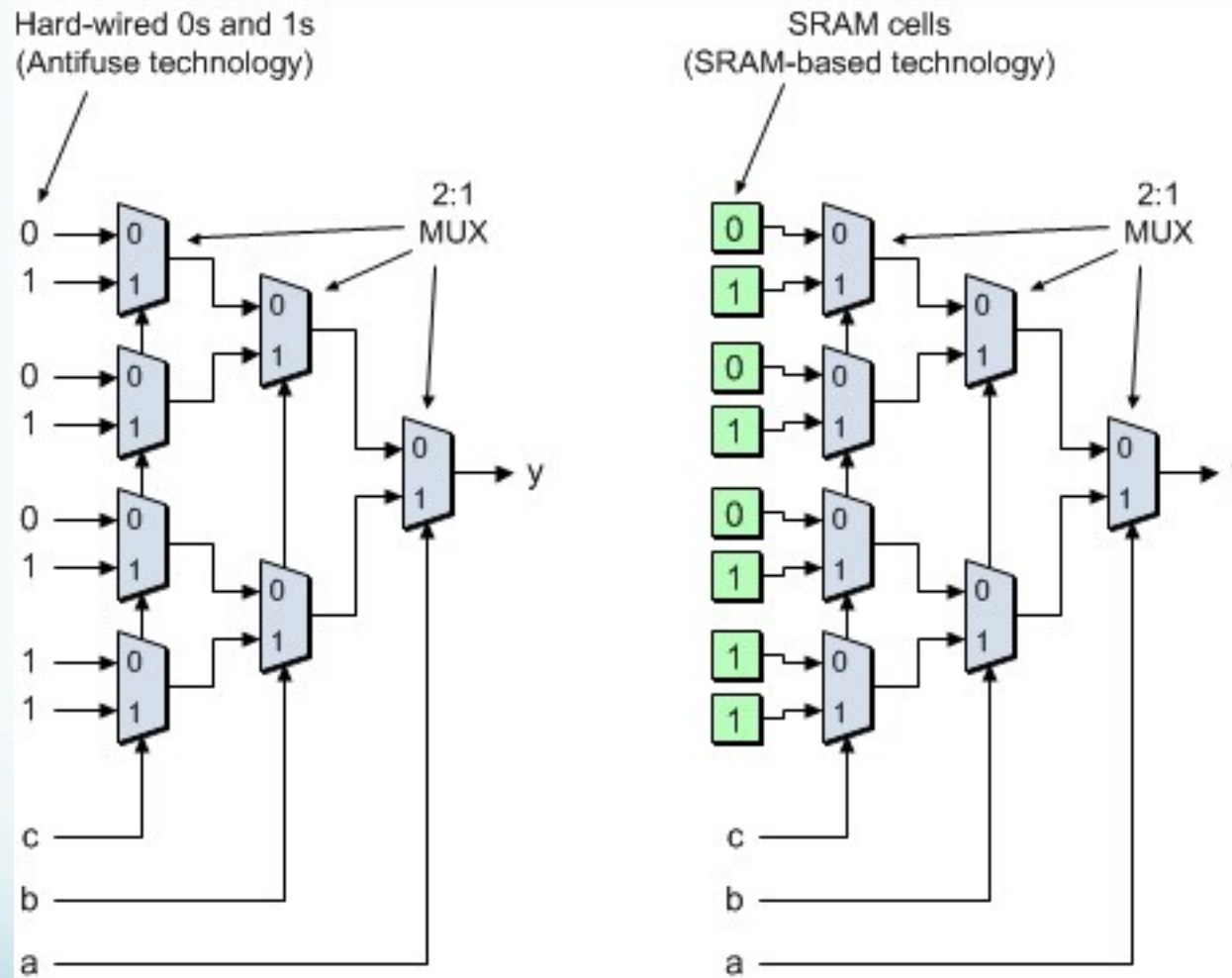
FLASH: erasable by block or by device

SRAM-Based Devices








Multi-transistor SRAM cell

Programming a 3-bit wide LUT



Summary of Technologies

Technology	Symbol	Predominantly associated with ...
Fusible-link		SPLDs
Antifuse		FPGAs
EPROM		SPLDs and CPLDs
E ² PROM/ FLASH		SPLDs, CPLDs, and FPGAs
SRAM		FPGAs (some CPLDs)



Rad-tolerant
secure

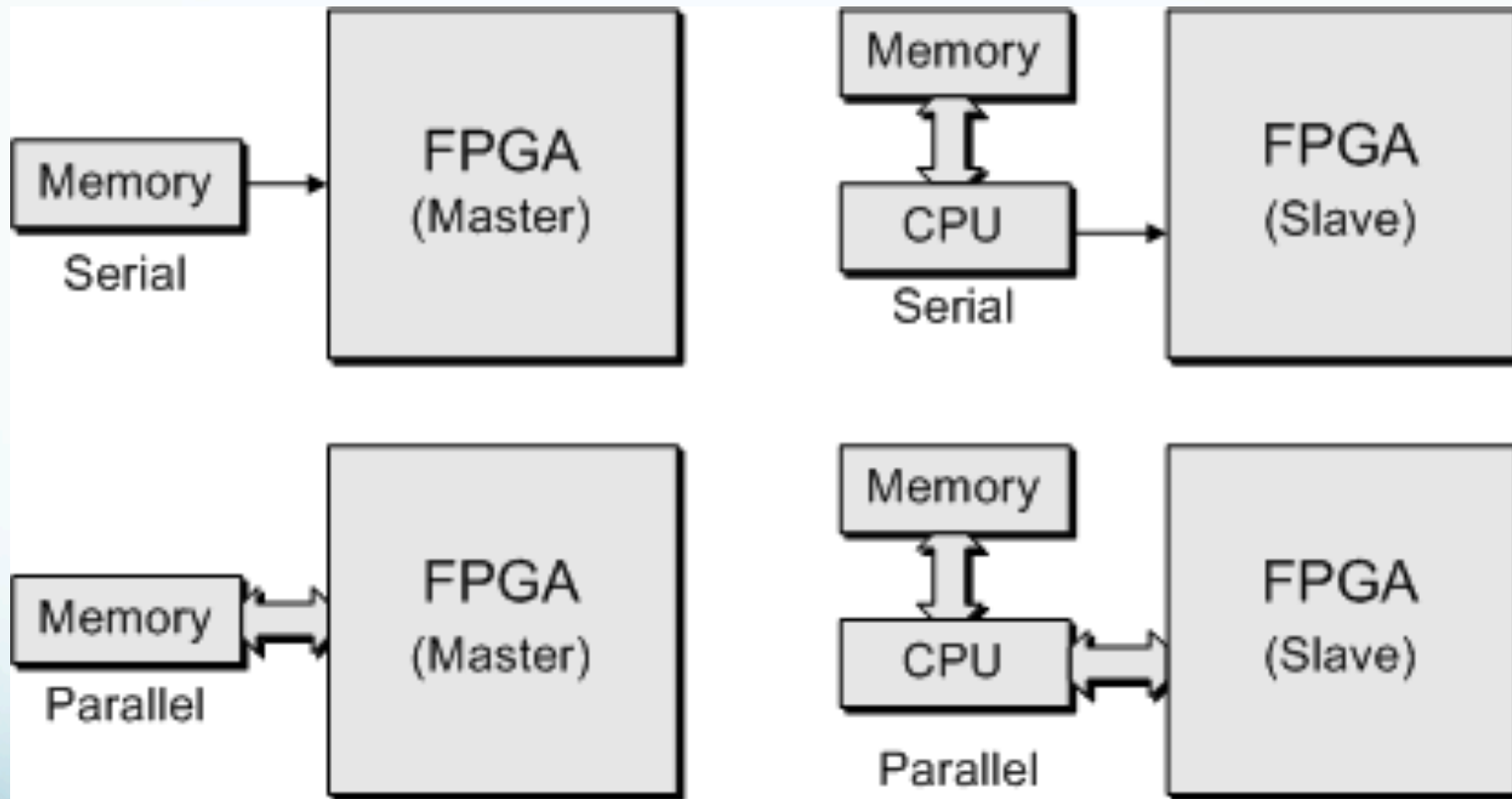


Rad-tolerant
(e.g. Alice)

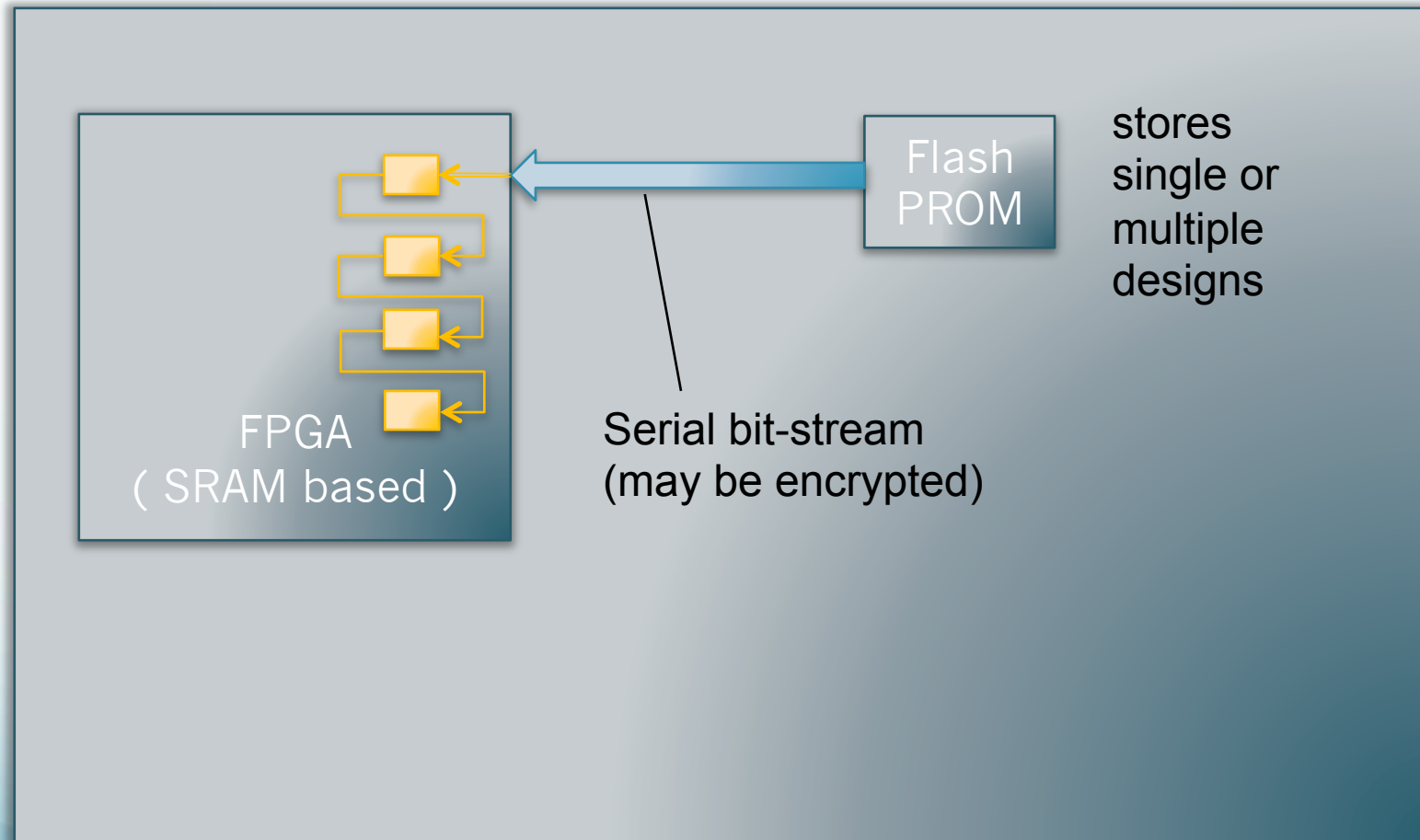


Used in most
FPGAs

Design Considerations (SRAM Config.)



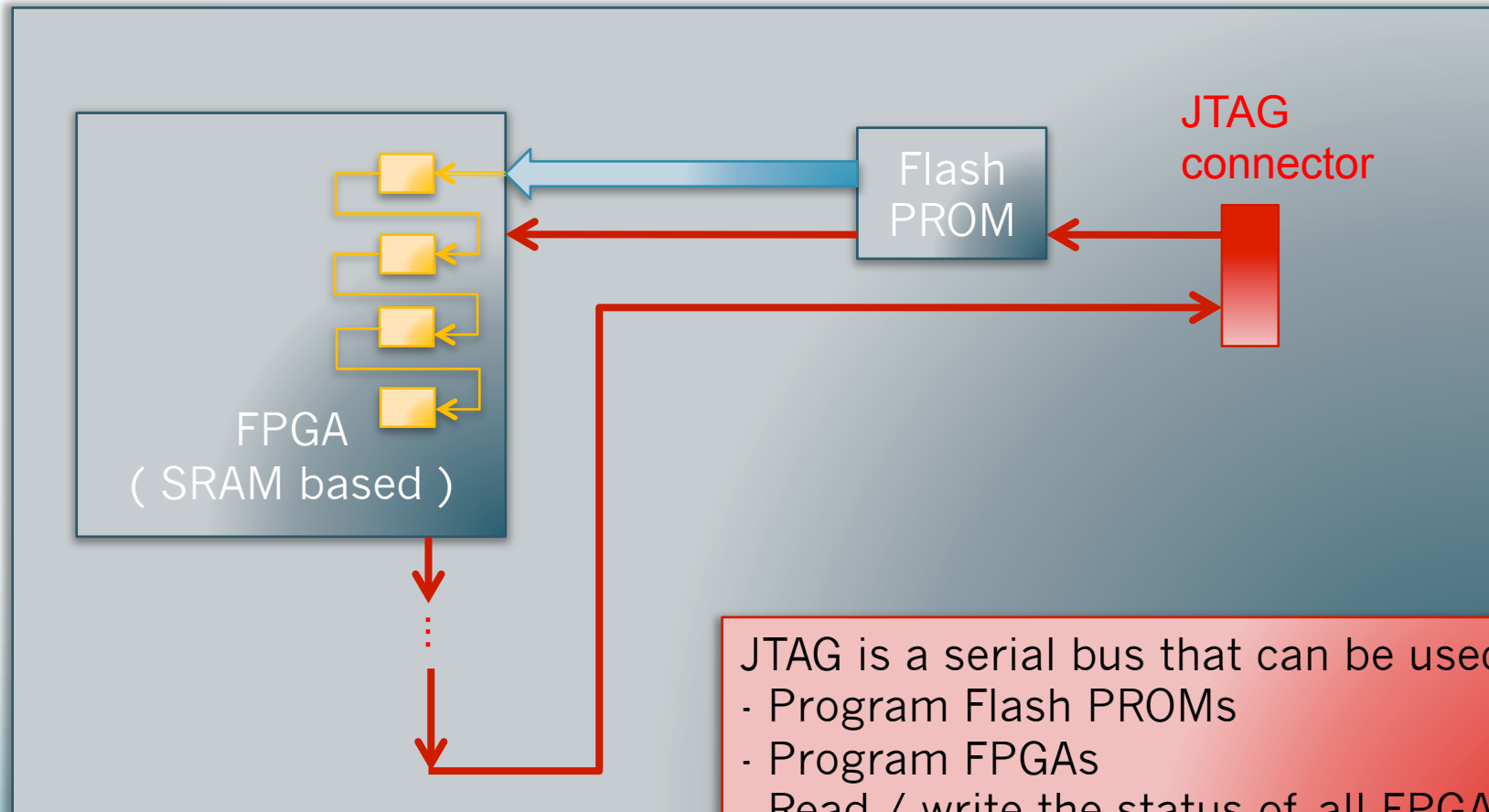
Configuration at power-up



Typical FPGA configuration time: milliseconds

Programming via JTAG

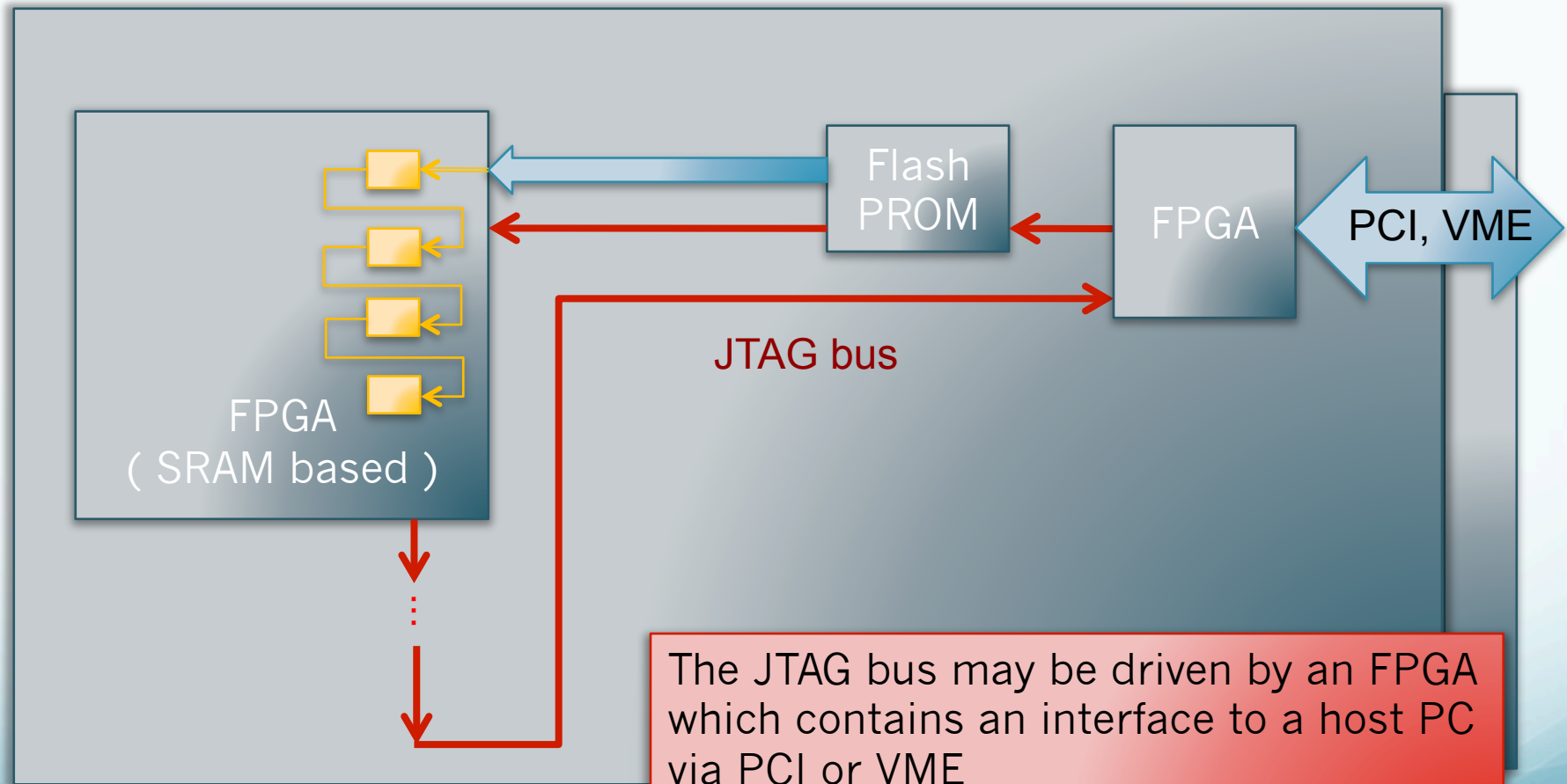
Joint Test Action Group



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



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

gateway can then be updated remotely

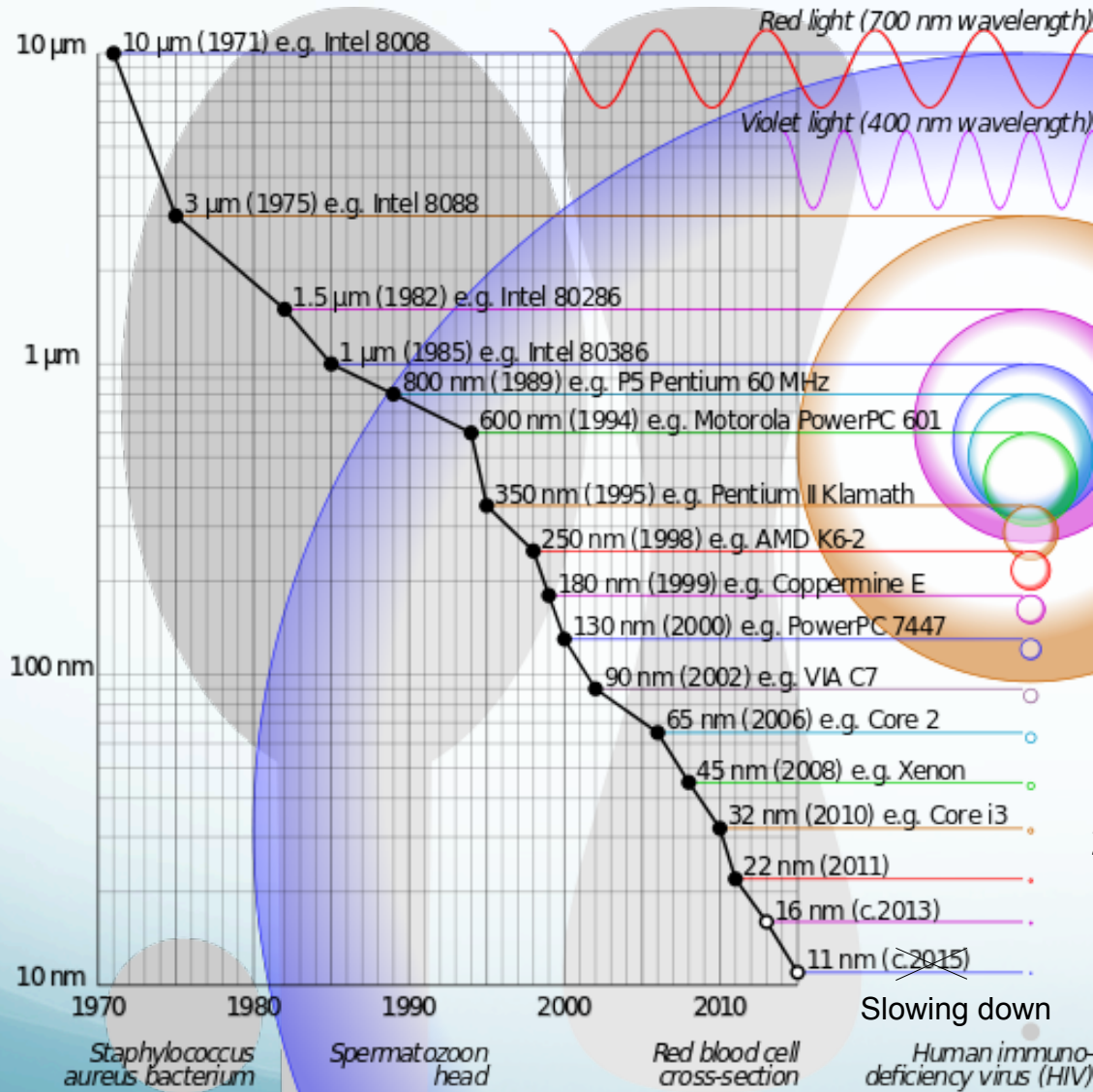
Major Manufacturers

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



Trends

Ever-decreasing feature size



- Higher capacity
- Higher speed
- Lower power consumption

130 nm
Xilinx Virtex-2

28 nm Xilinx Virtex-7 / Altera Stratix V

16 nm Xilinx UltraScale +

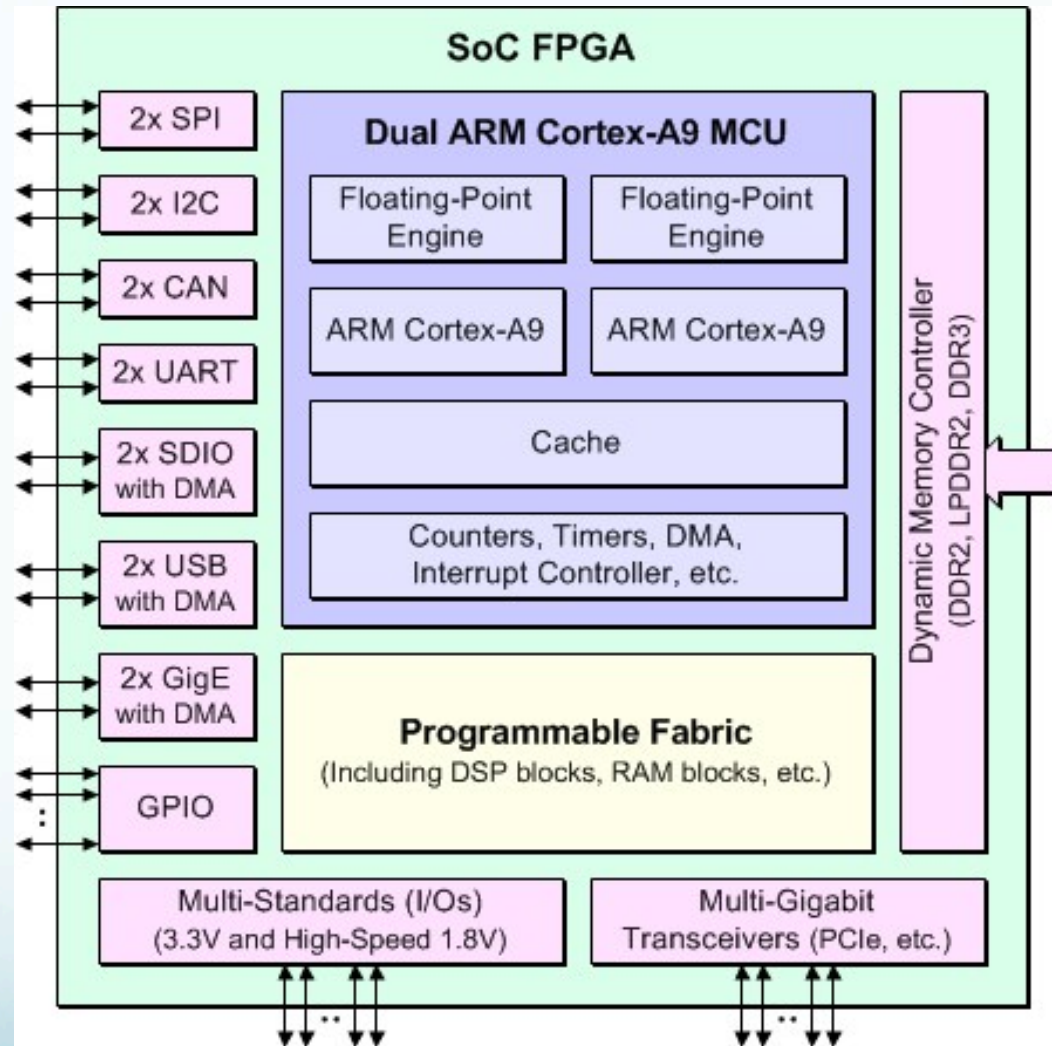
14 nm Intel Stratix 10

5.5 million logic cells
4 million logic cells

Trends

- Speed of logic increasing
- Look-up-tables with more inputs (5 or 6)
- Speed of serial links increasing (multiple Gb/s)
- Integrated High Bandwidth Memory (HBM) in-package
- Additional Flip Flops in routing resources
- More and more hard macro cores on the FPGA
 - PCI Express
 - Gen2: 5 Gb/s per lane
 - Gen3: 8 Gb/s per lane
up to 8 lanes / FPGA
 - Gen4: 16 Gb/s per lane
 - 10 Gb/s, 40 Gb/s, 100 Gb/s Ethernet
- Sophisticated soft macros
 - CPUs
 - Gb/s MACs
 - Memory interfaces (DDR2/3/4)
- Processor-centric architectures – see next slides

System-On-a-Chip (SoC) FPGAs



Xilinx Zynq

Altera Stratix 10

CPU(s) + Peripherals + FPGA in one package

FPGAs in Server Processors and the Cloud

- Since 2016: Intel Xeon Server Processor with FPGA in socket
 - Intel acquired Altera in 2015



- FPGAs in the cloud
 - Amazon Elastic Cloud F1 instances
 - 8 CPUs / 1 Xilinx UltraScale+ FPGA
 - 64 CPUs / 8 Xilinx UltraScale+ FPGA

FPGA – ASIC comparison

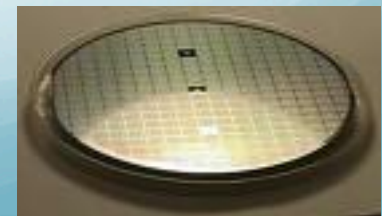
FPGA

- Rapid development cycle (minutes / hours)
- May be reprogrammed in the field (gateway upgrade)
 - New features
 - Bug fixes
- Low development cost
 - You can get started with a development board (< \$100) and free software



ASIC

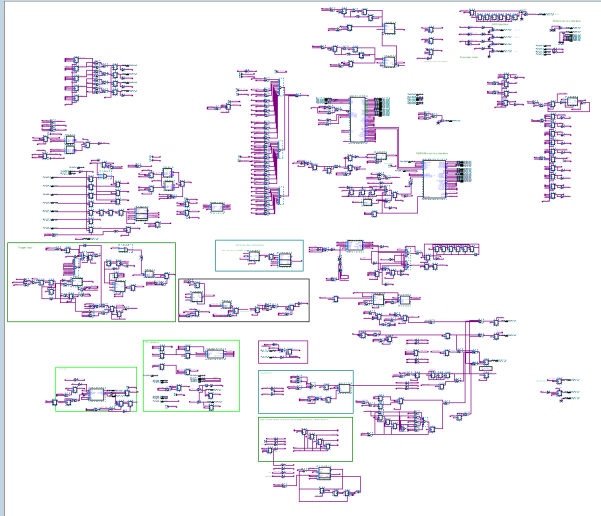
- Higher performance
- Analog designs possible
- Better radiation hardness
- Long development cycle (weeks / months)
- Design cannot be changed once it is produced
- Extremely high development cost
 - ASICs are produced at a semiconductor fabrication facility ("fab") according to your design
- Lower 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

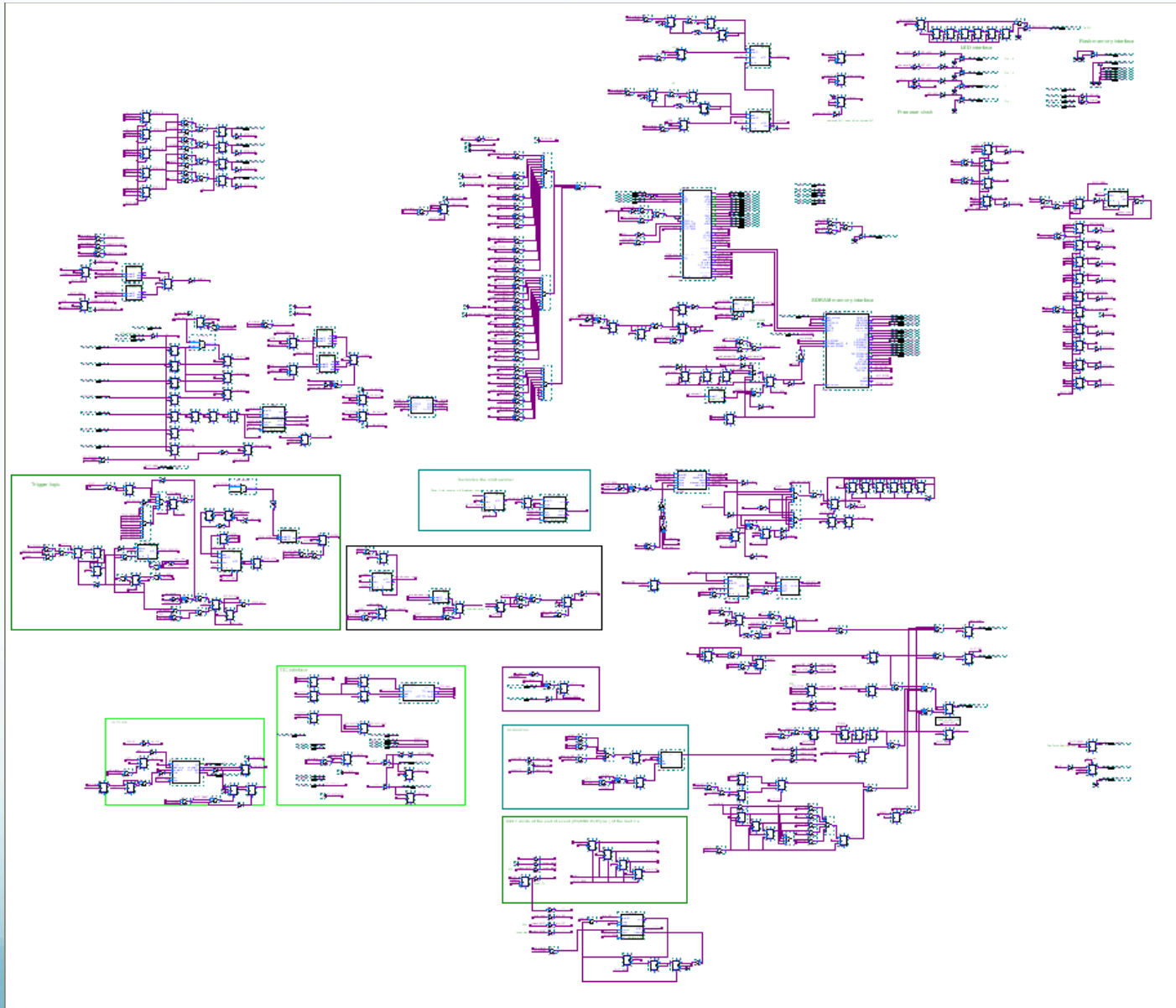
Hardware description language VHDL, Verilog

```
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

Example: 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;
```

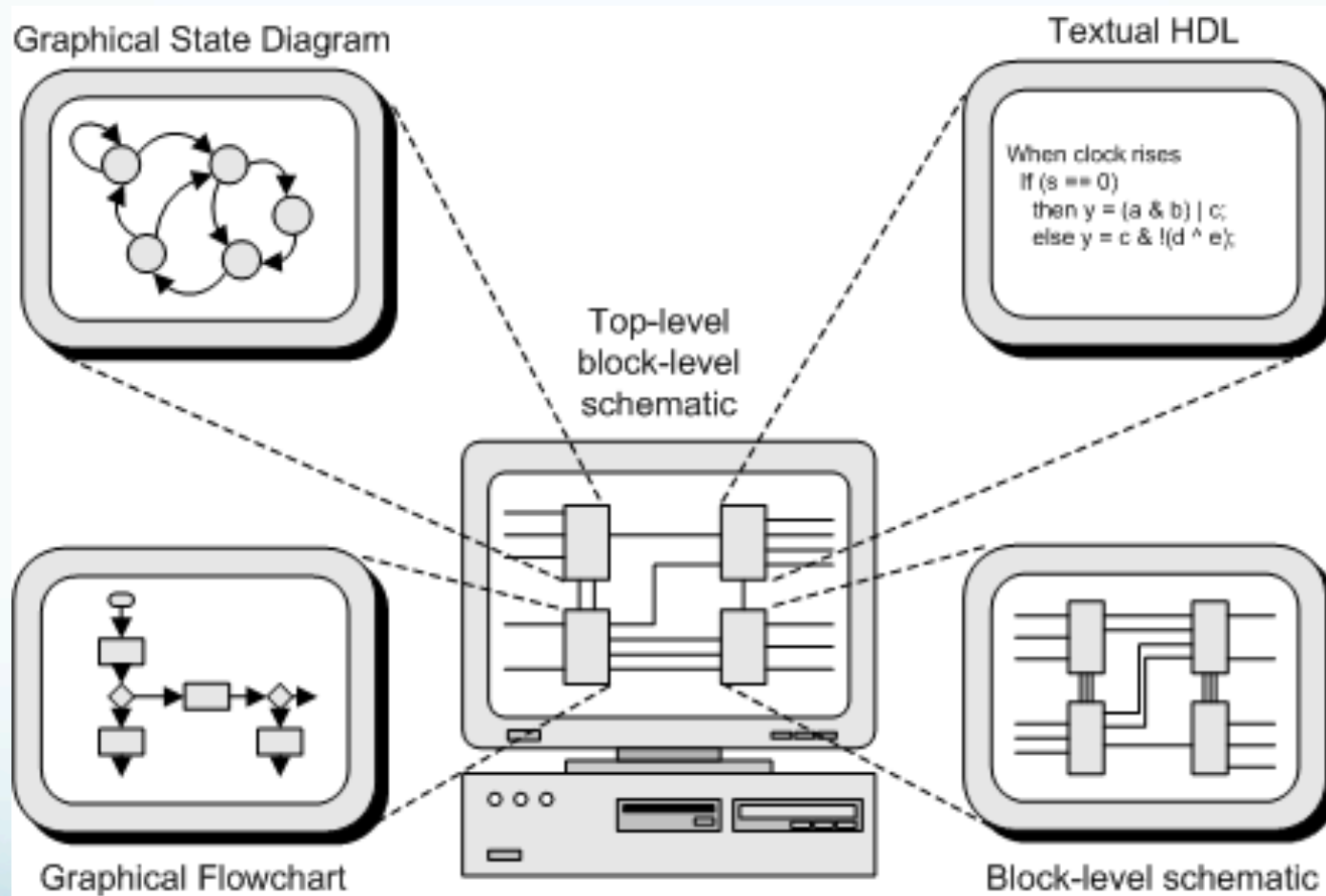
```
    reg: process (vme_clk, reset) is  
    begin -- process reg  
        if reset = '1' then                -- asynchronous reset  
            Q <= init_val;  
            vme_en_out <= '0';  
        elsif vme_clk'event and vme_clk = '1' then -- rising clock edge  
            vme_en_out <= vme_en_i;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

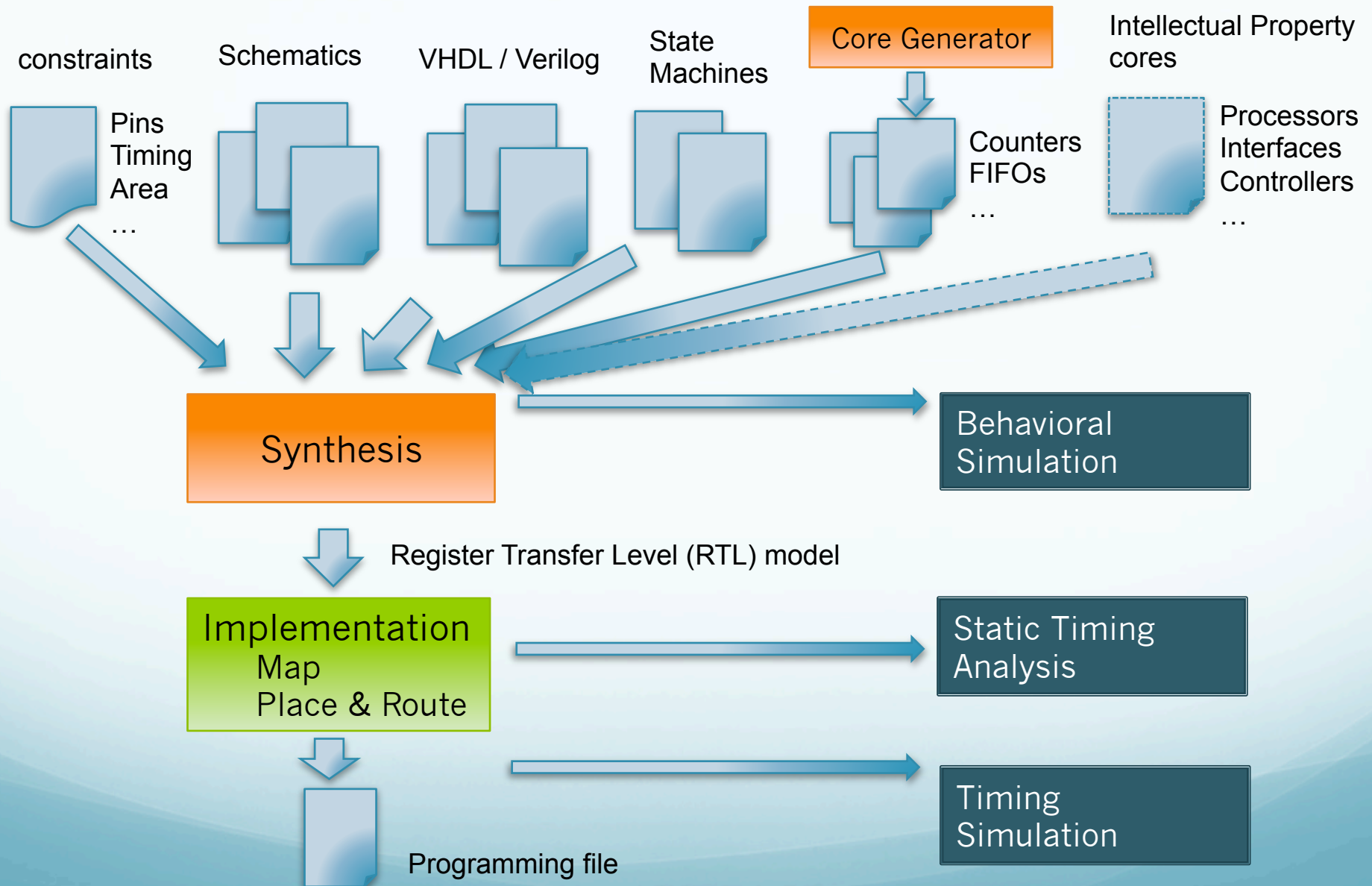
```
end behavioral;
```

- Looks like a programming language
- All statements executed in parallel, except inside processes

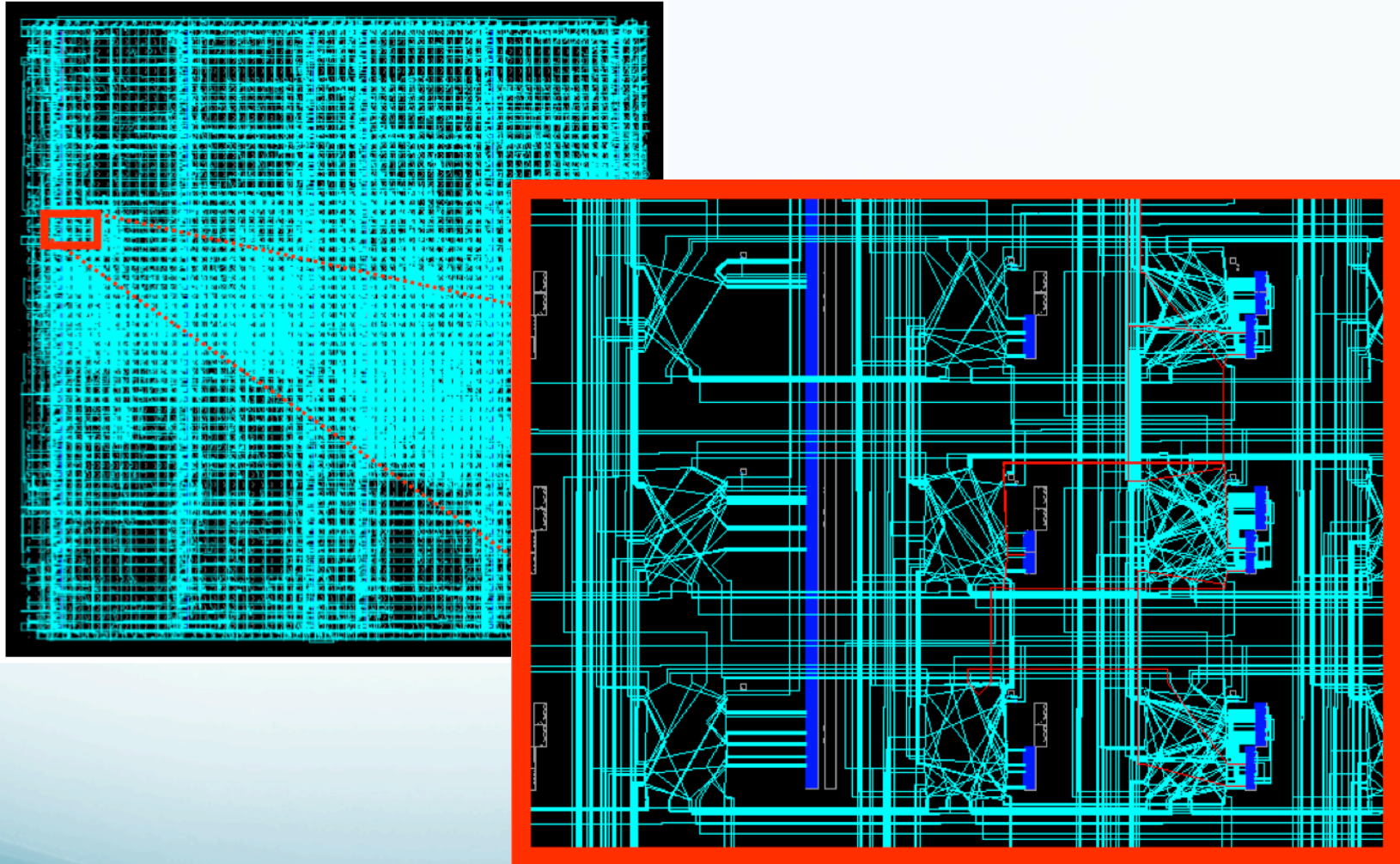
Schematics & HDL combined



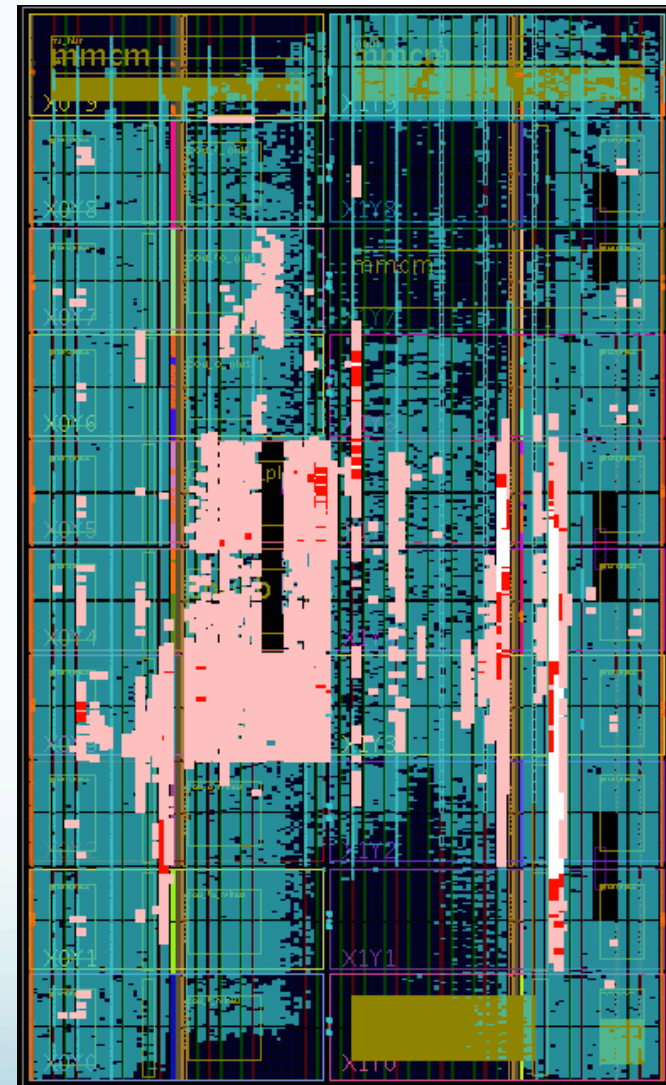
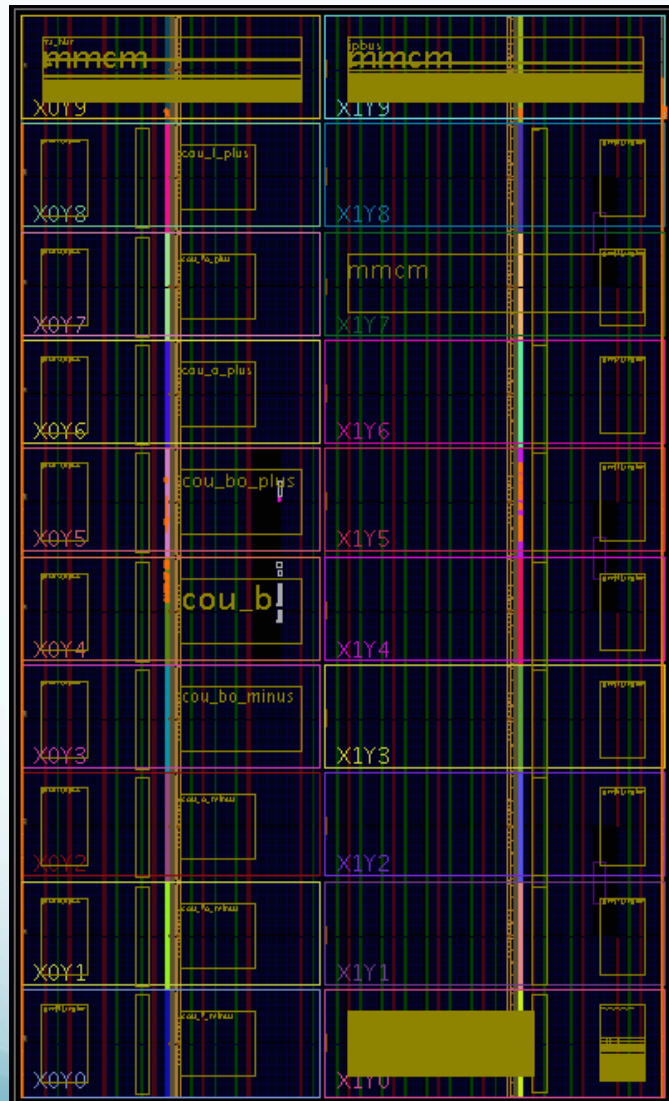
Design flow



Floorplan (Xilinx Virtex 2)



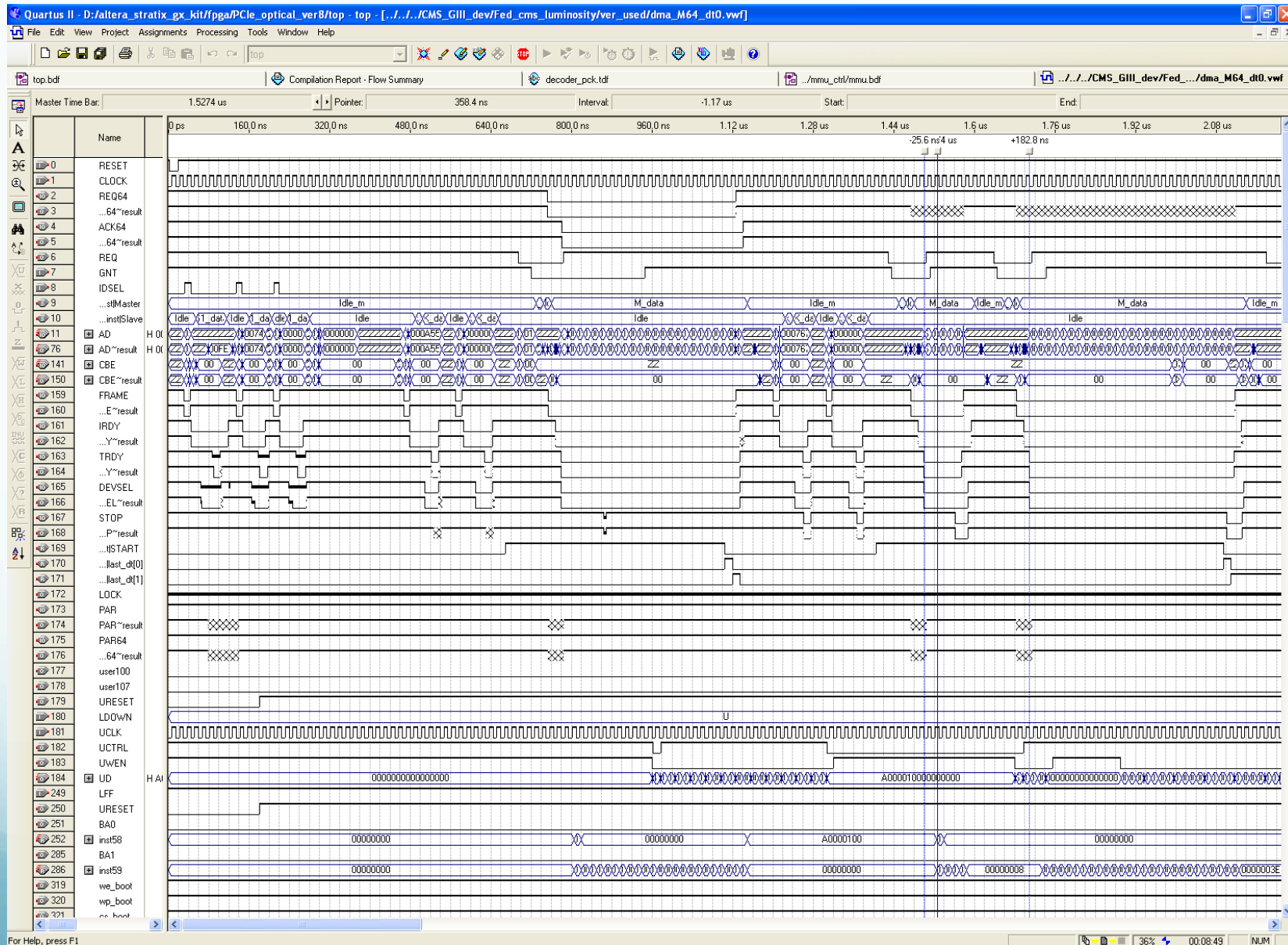
Manual Floor planning



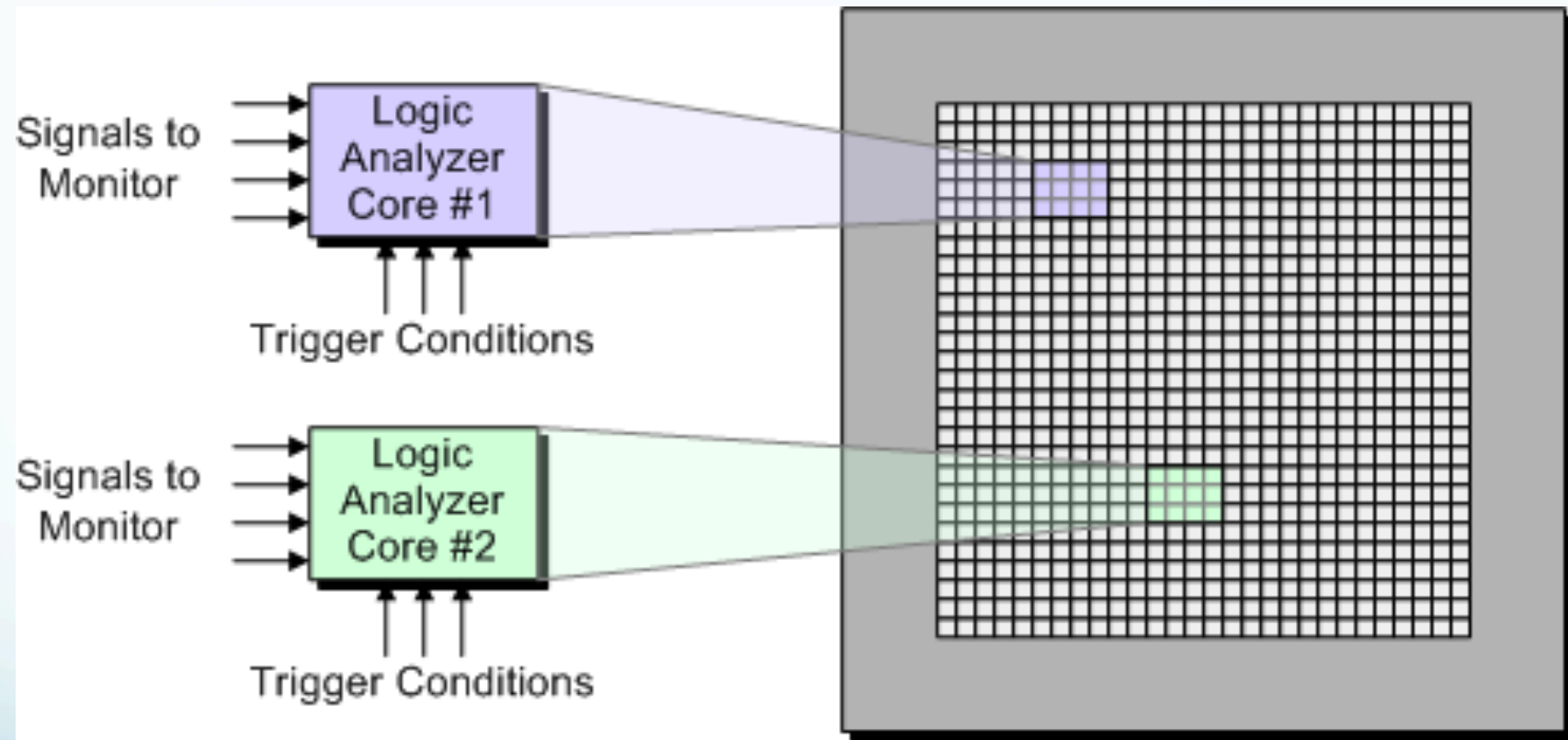
- For large designs, manual floor planning may be necessary

Routing congestion
Xilinx Virtex 7 (Vivado)

Simulation



Embedded Logic Analyzers



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)

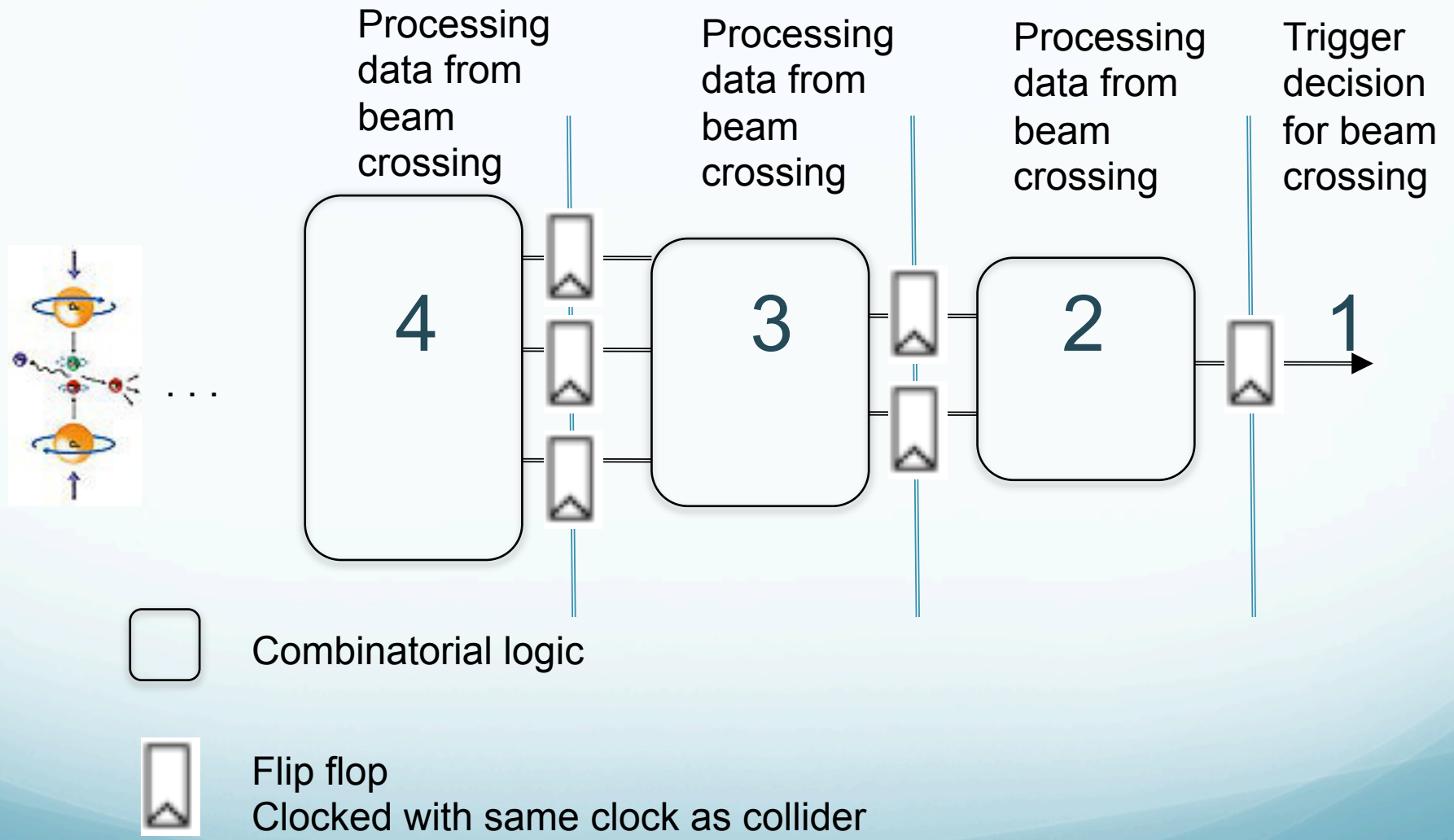
N beam crossings

Trigger decision YES / NO
(for every beam crossing)

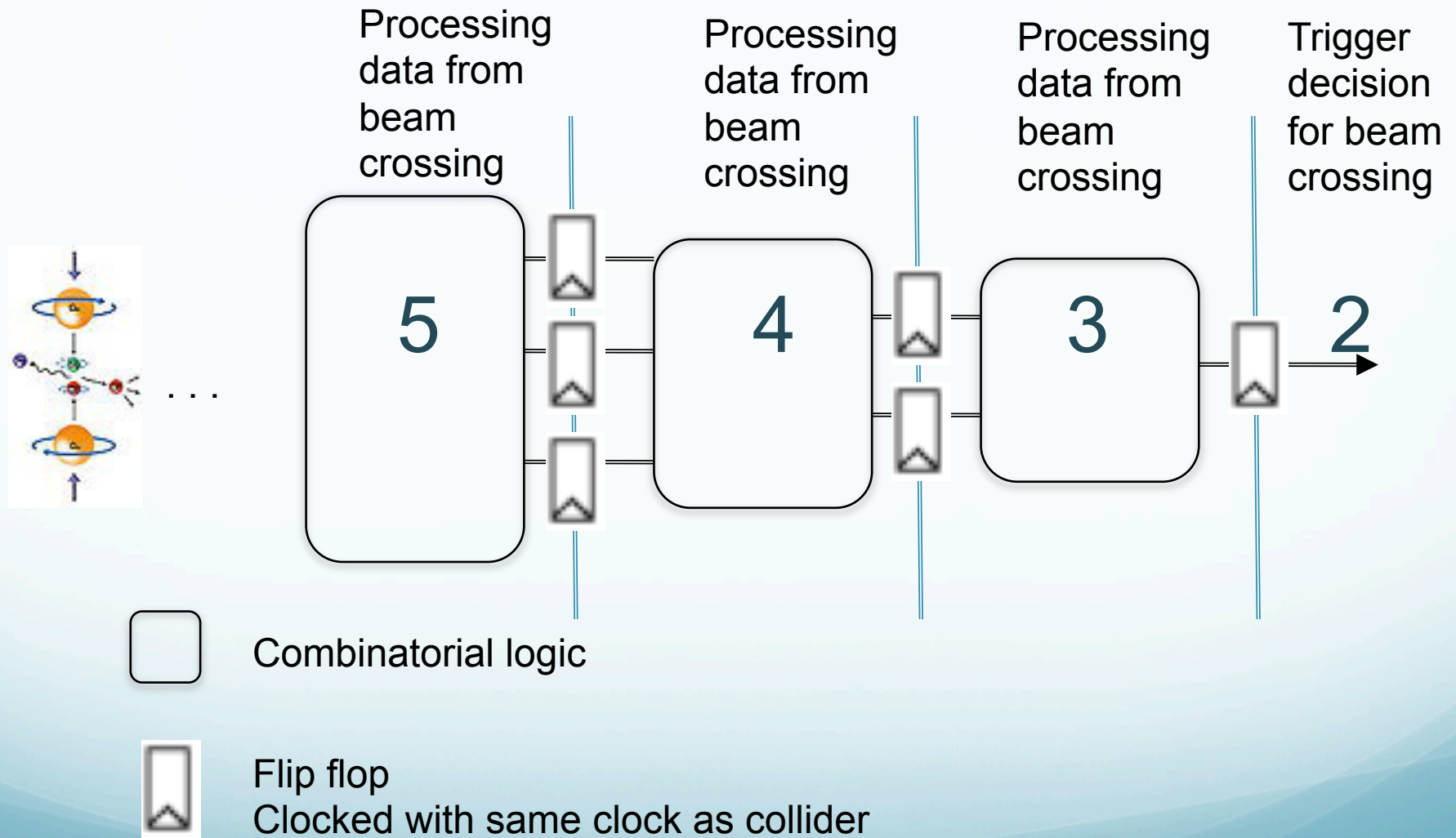
De-
randomizer
FIFO

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

Pipelined Logic



Pipelined Logic – a clock cycle later



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
- All operations are performed in parallel
 - Can build pipelined logic
- They can be re-programmed
 - Trigger algorithms can be optimized



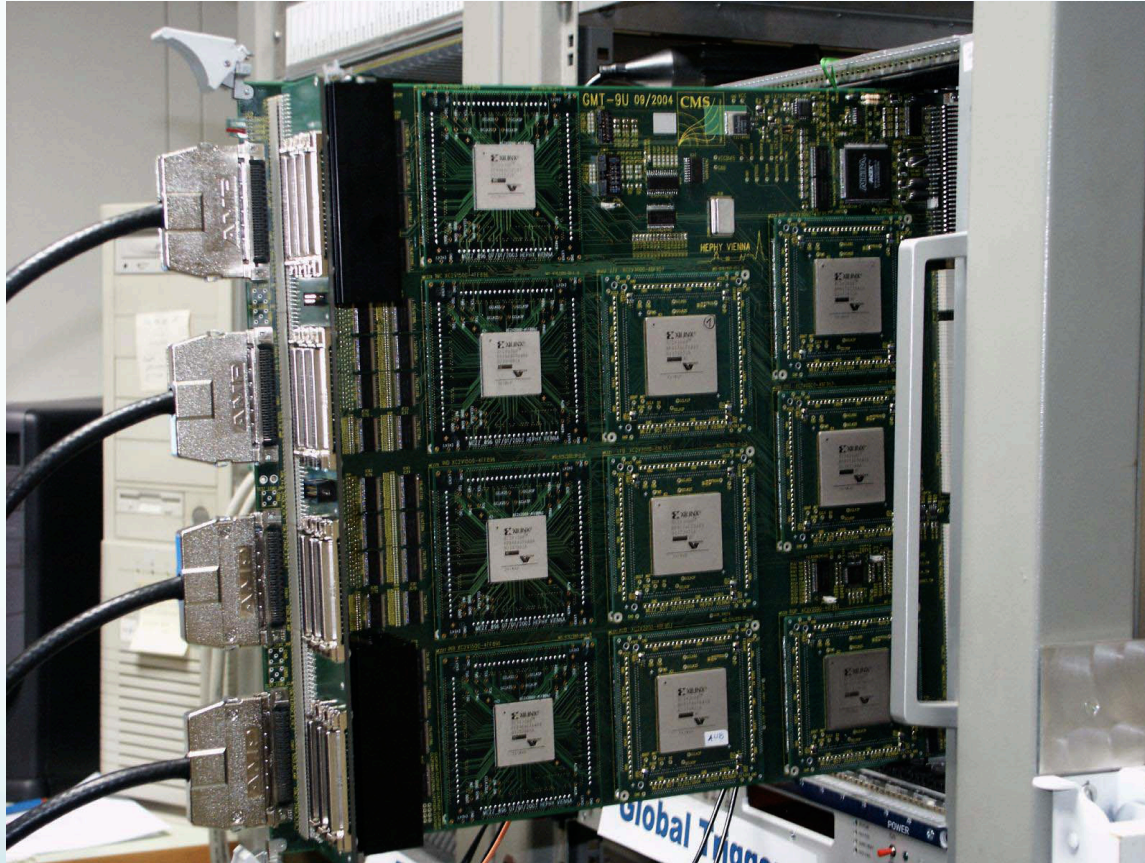
Low latency

High performance

Trigger algorithms implemented in FPGAs

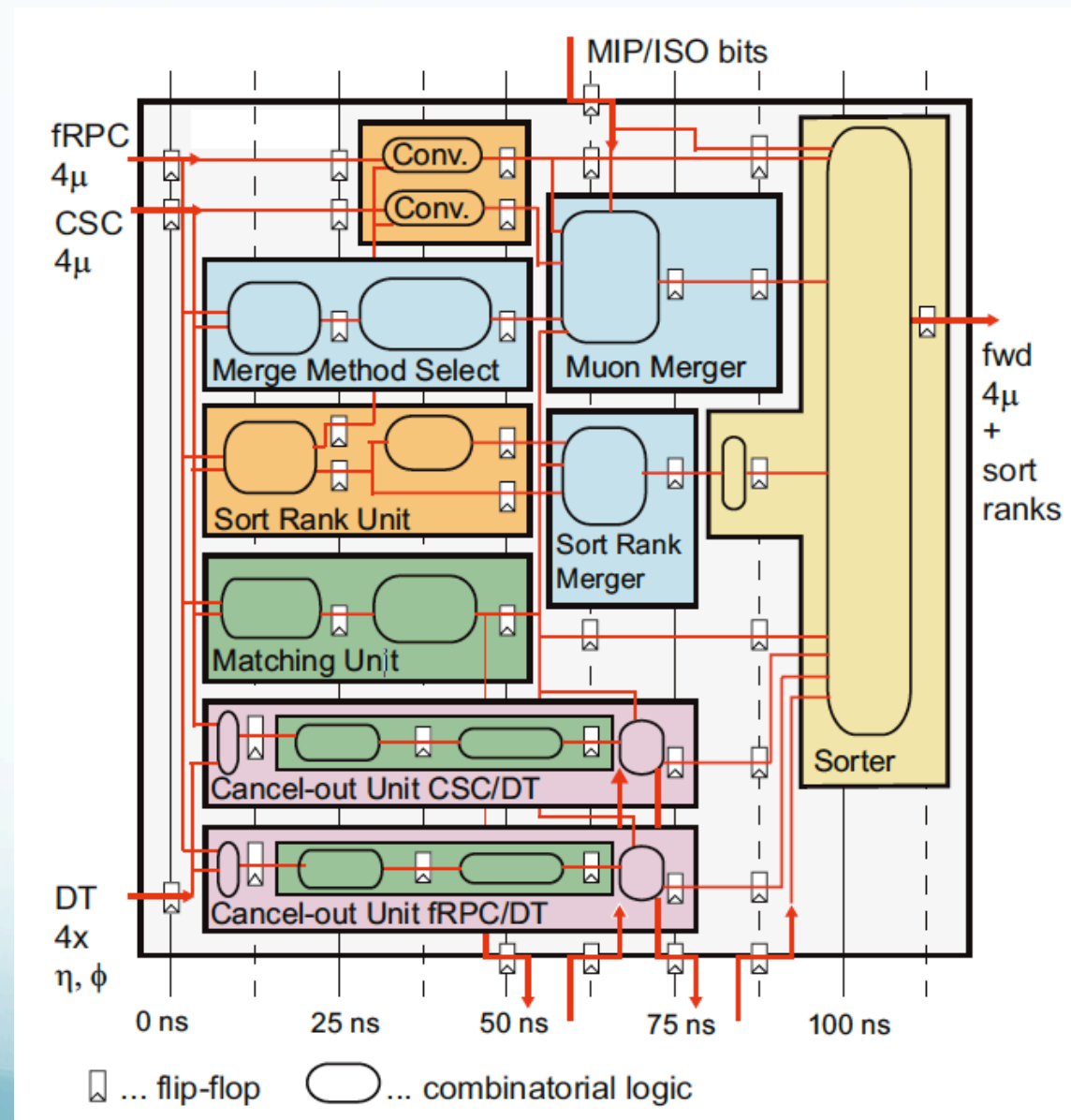
- Peak finding
- Pattern Recognition
- Track Finding
- Energy summing
- Sorting
- Topological Algorithms (invariant mass)
- Trigger Control system
- Fast signal merging
- Many more ...

Example 1: CMS Global Muon Trigger

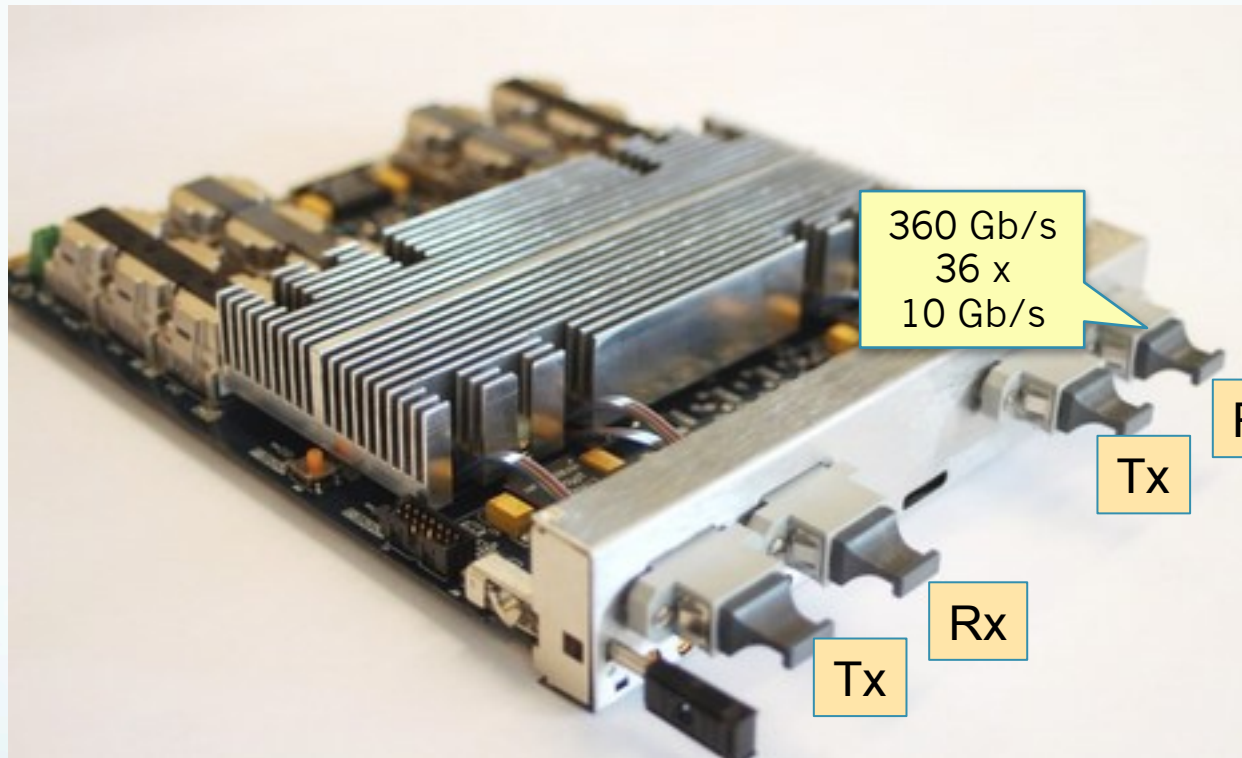


- Input: ~1000 bits @ 40 and 80 MHz
 - Output: ~50 bits @ 80MHz
 - Processing time: 250 ns
 - Pipelined logic one new result every 25 ns
 - 10 Xilinx Virtex-II FPGAs
 - up to 500 user I/Os per chip
 - Up to 25000 LUTs per chip used
 - Up to 96 x 18kbit RAM used
 - In use in the CMS trigger 2008-2015
- 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



Example 2: New μ TCA board for CMS trigger upgrade based on Virtex 7



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

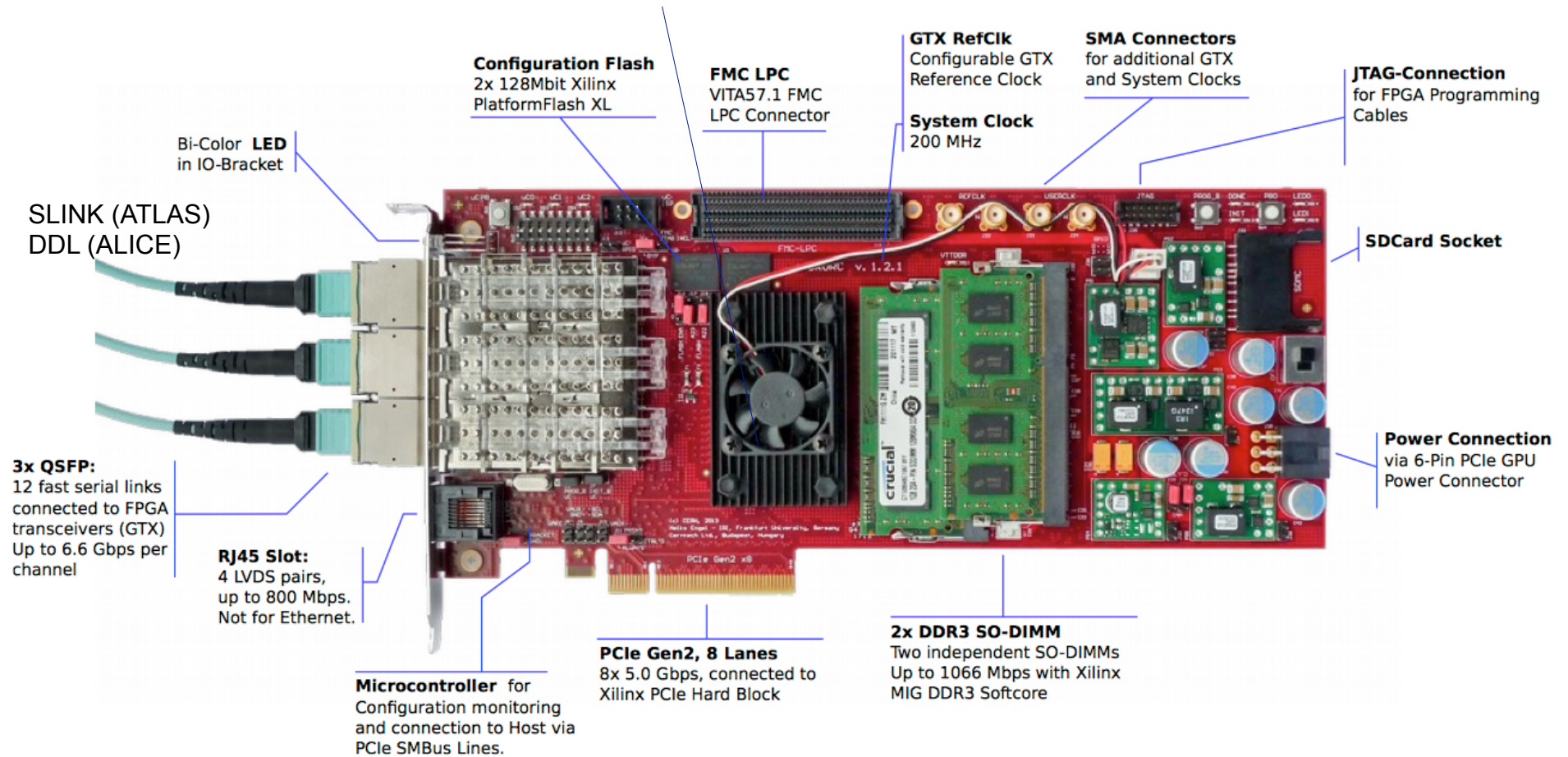
Same board used for different functions
(different gateway)
Separation of framework + algorithm fw

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

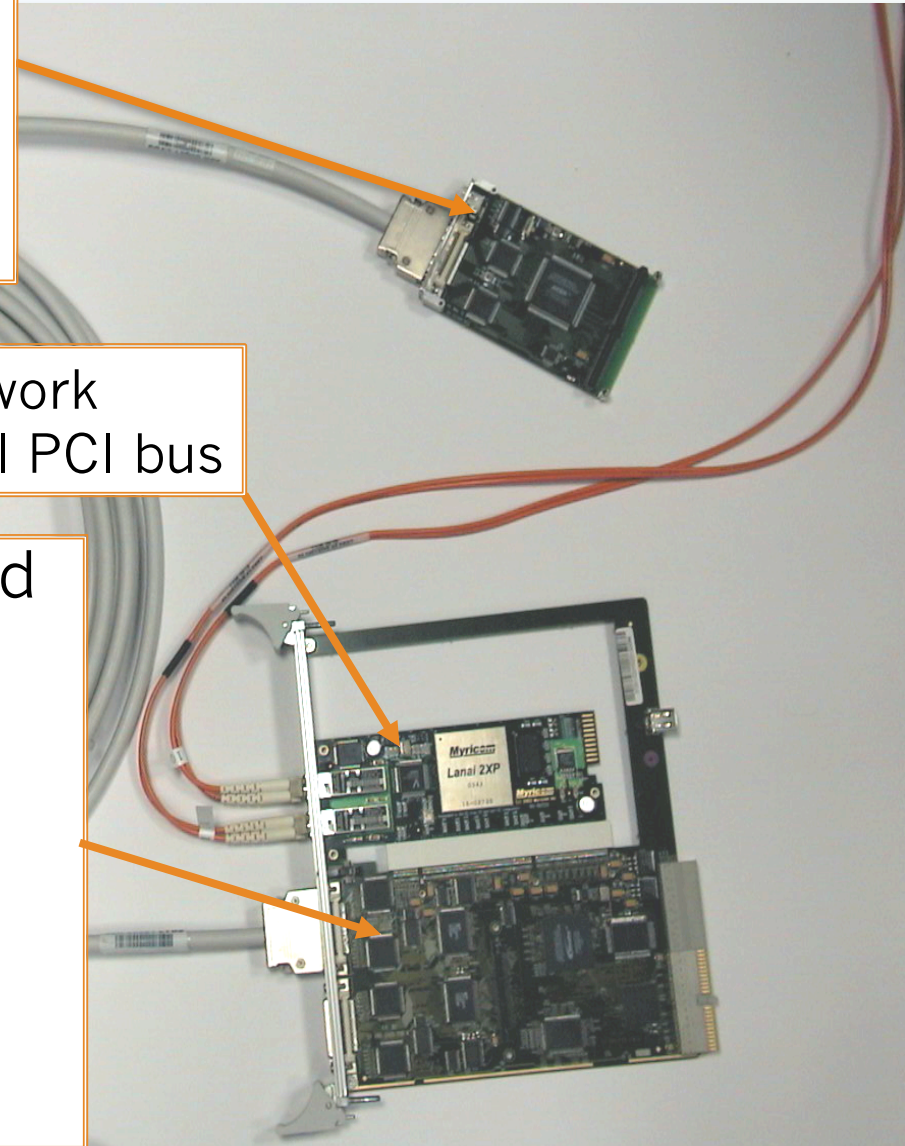


Example 3: CMS Front-end Readout Link (Run-1)

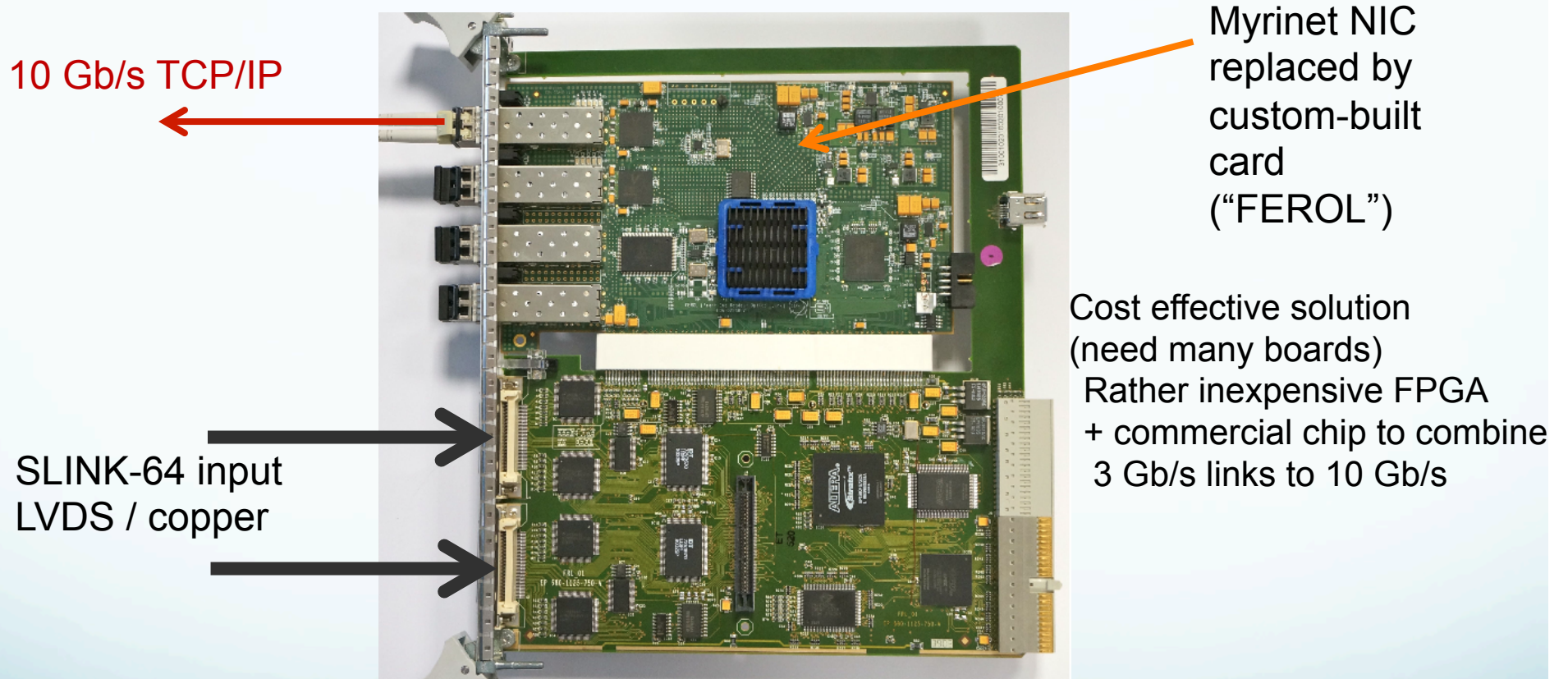
- SLINK Sender Mezzanine Card: 400 MB / s
 - 1 FPGA (Altera)
 - CRC check
 - Automatic link test

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



Example 4: CMS Readout Link for Run-2 in use since 2015

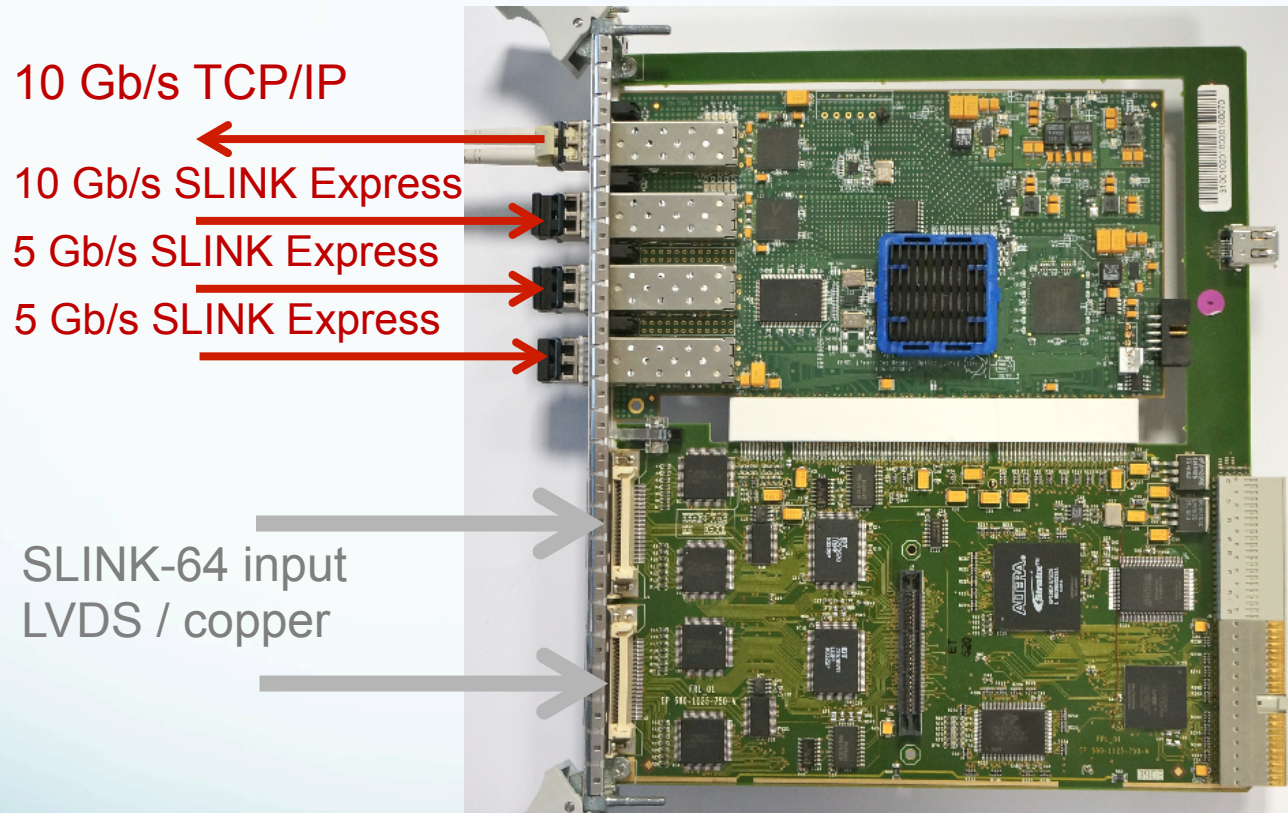


FEROL (Front End Readout Optical Link)

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

Example 4: CMS Readout Link for Run-2



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

FPGAs in other domains

- Medical imaging
- Advanced Driver Assistance Systems (Image Processing)
- Speech recognition
- Cryptography
- Bioinformatics
- Aerospace / Defense
- Bitcoin mining

- ASIC Prototyping
- High performance computing
 - Accelerator cards



- Server processors w. FPGA