Field Programmable Gate Arrays

Hannes Sakulin
CERN / PH-CMD

Third International School of Trigger and Data Acquisition
Krakow, Poland, Feb 2nd, 2012

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
  - Your electronic circuit is ready to use

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

Outline

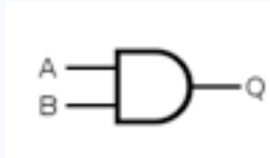
- Quick recap of digital electronics
- Short history of programmable logic devices
- FPGAs and their features
- Trends
- Programming techniques
- Design flow
- Example Applications in the Trigger and DAQ domain

Quick recap of digital electronics

The building blocks: logic gates

	Truth table	C equivalent

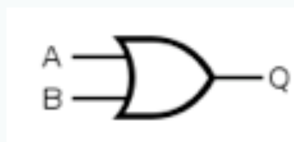
AND gate



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

q = a && b;

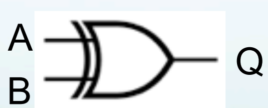
OR gate



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

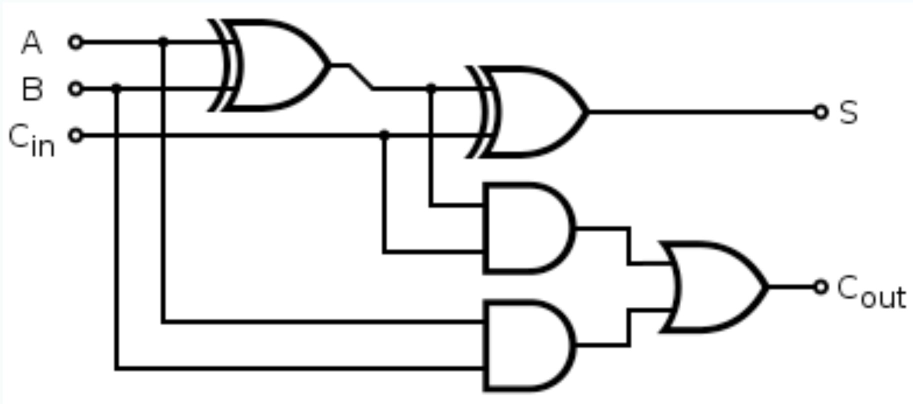


INPUT		OUTPUT
A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

q = a != b;

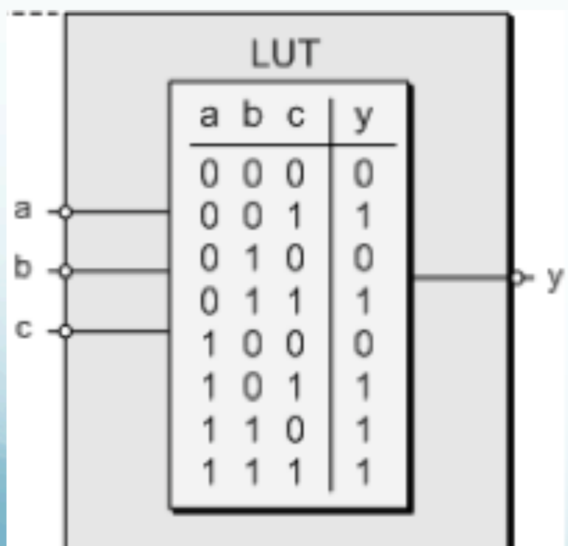
⋮

Combinatorial logic (asynchronous)



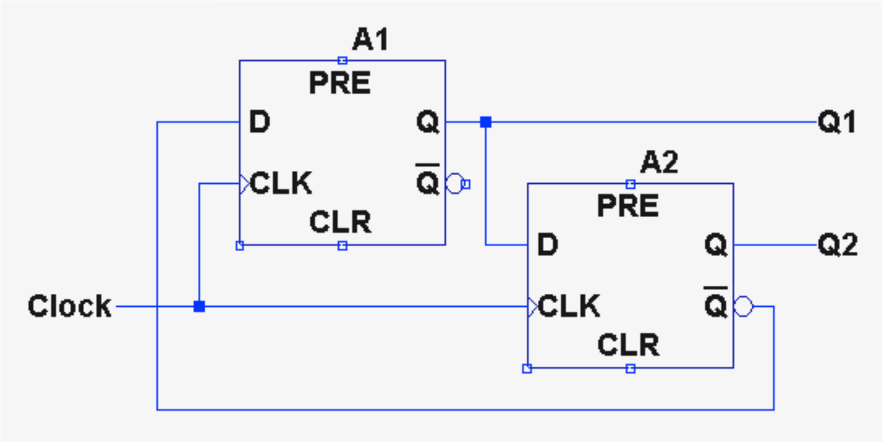
Example: Full adder with carry-in, carry-out

Outputs are determined by Inputs, only



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

(Synchronous) sequential logic



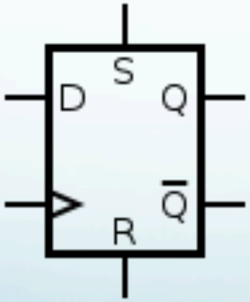
2-bit binary counter

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

set

data

clock



reset

Output

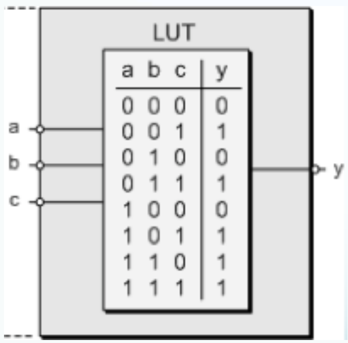
Inverted output

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

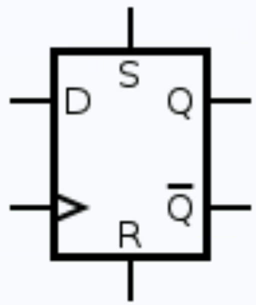
The output will be equal to
the last sampled input until the
next rising (or falling) clock edge

D Flip-flop

Synchronous sequential logic



+



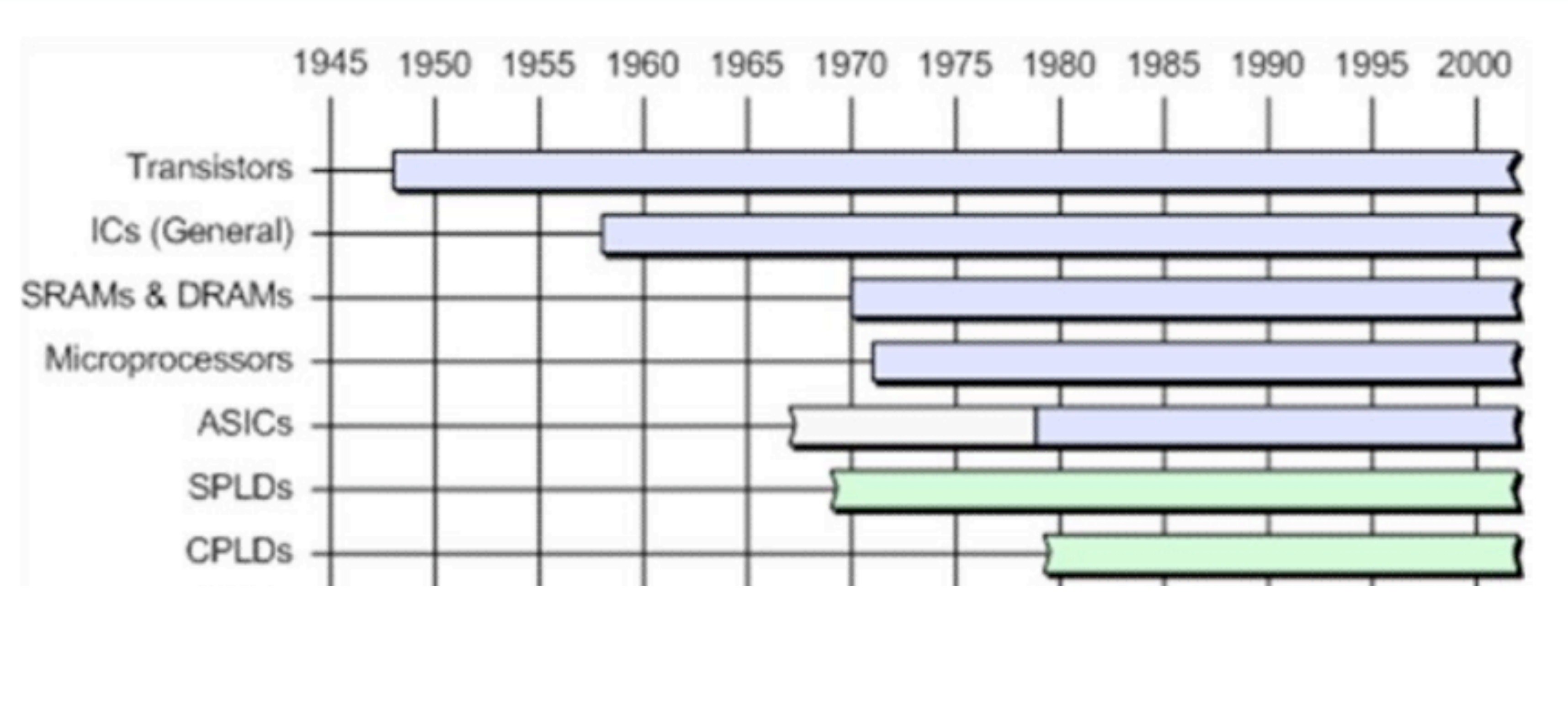
=

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

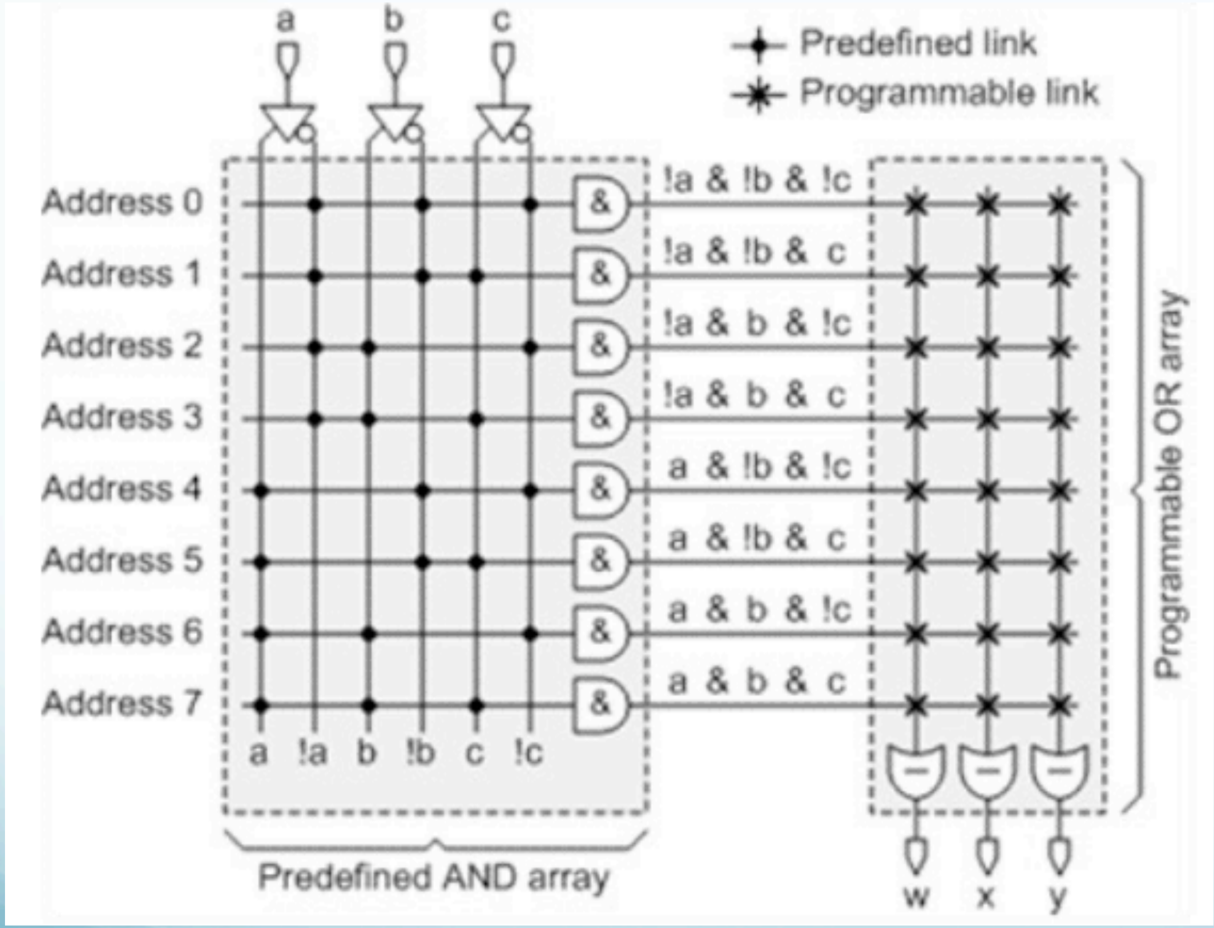
Of course there are some details
to be learnt about electronics design …

Programmable digital electronics

Long long time ago …



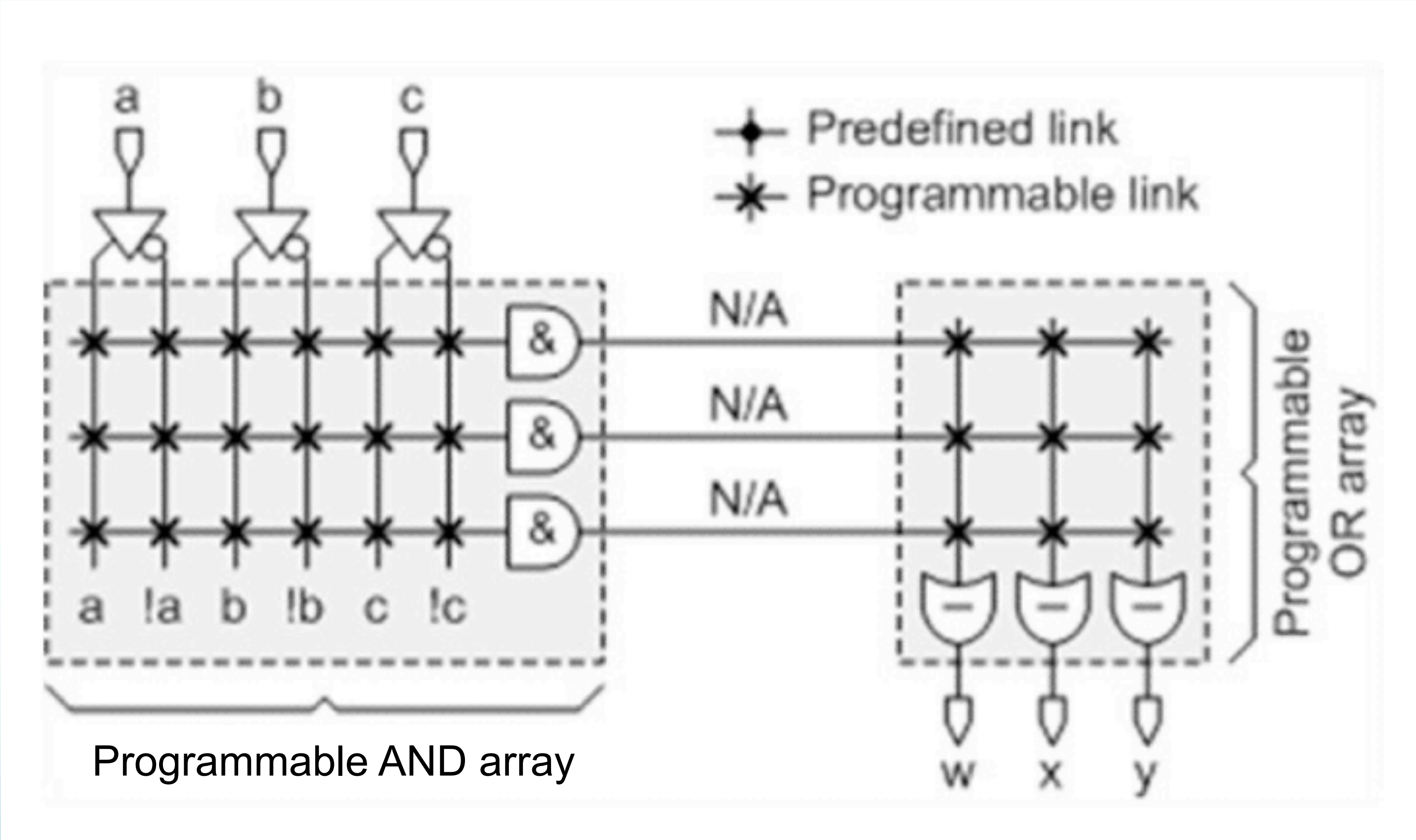
Simple Programmable Logic Devices (sPLDs)
a) Programmable Read Only Memory (PROMs)



Late 60's

Unprogrammed PROM (Fixed AND Array, Programmable OR Array)

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

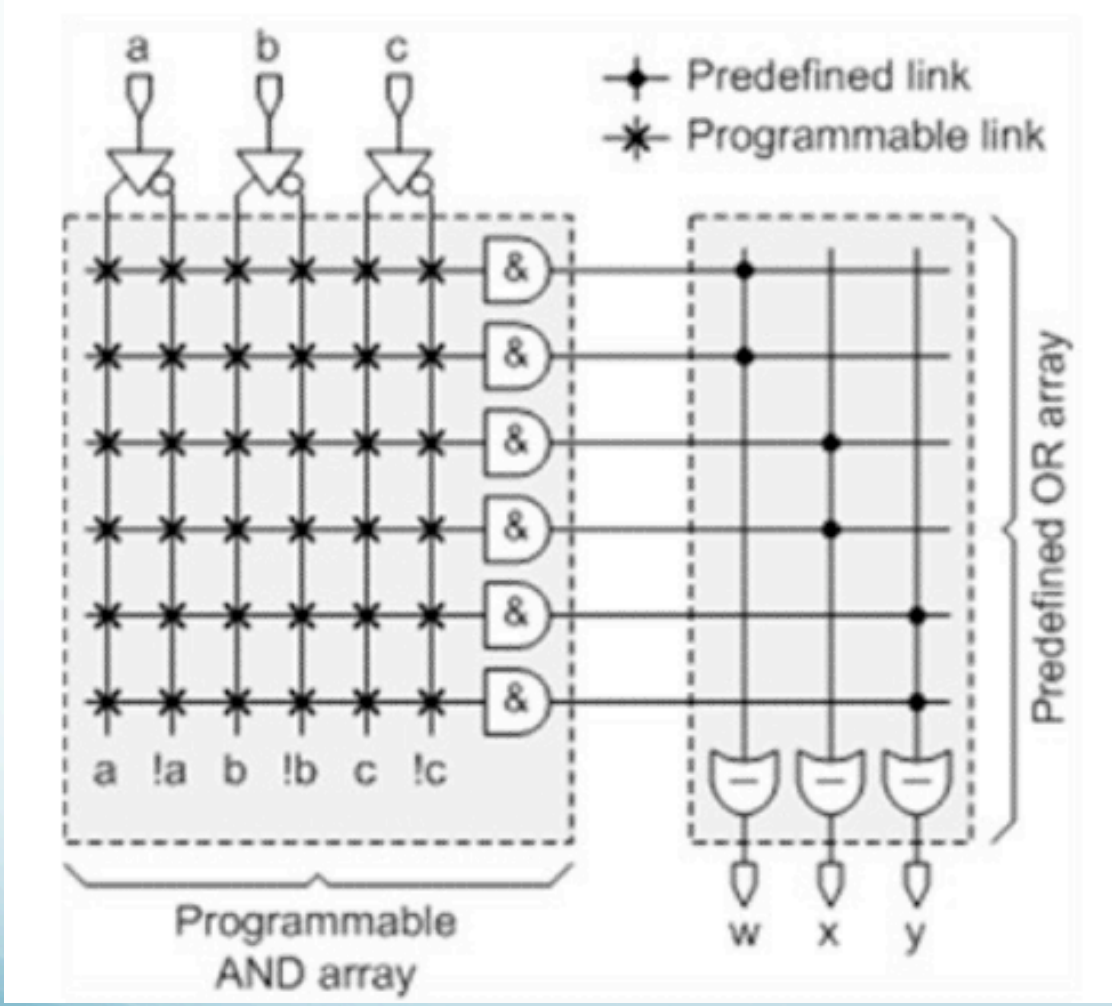


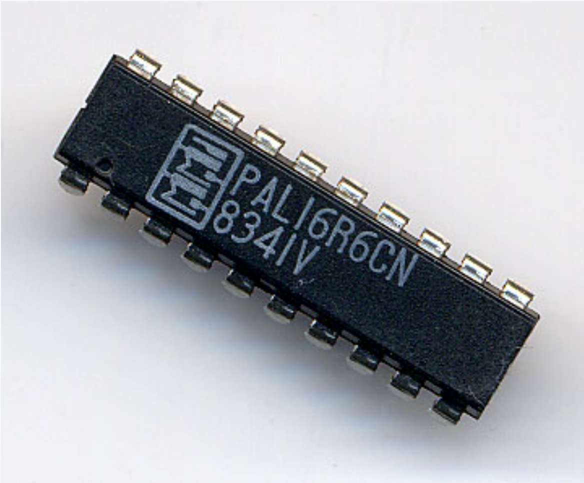
Unprogrammed PLA (Programmable AND and OR Arrays)

Most flexible
but slower

1975

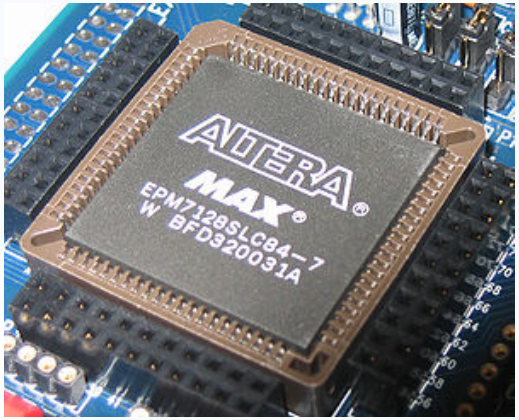
Simple Programmable Logic Devices (sPLDs)
c) Programmable Array Logic (PAL)

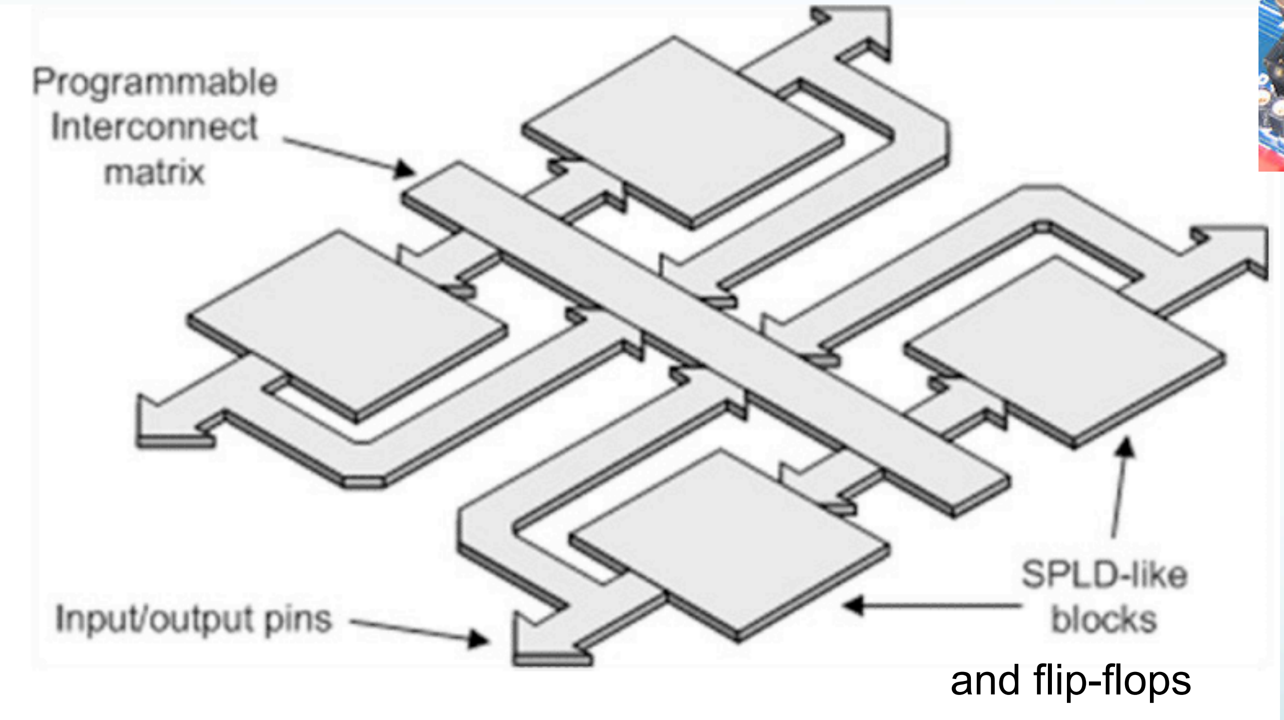




Unprogrammed PAL (Programmable AND Array, Fixed OR Array)

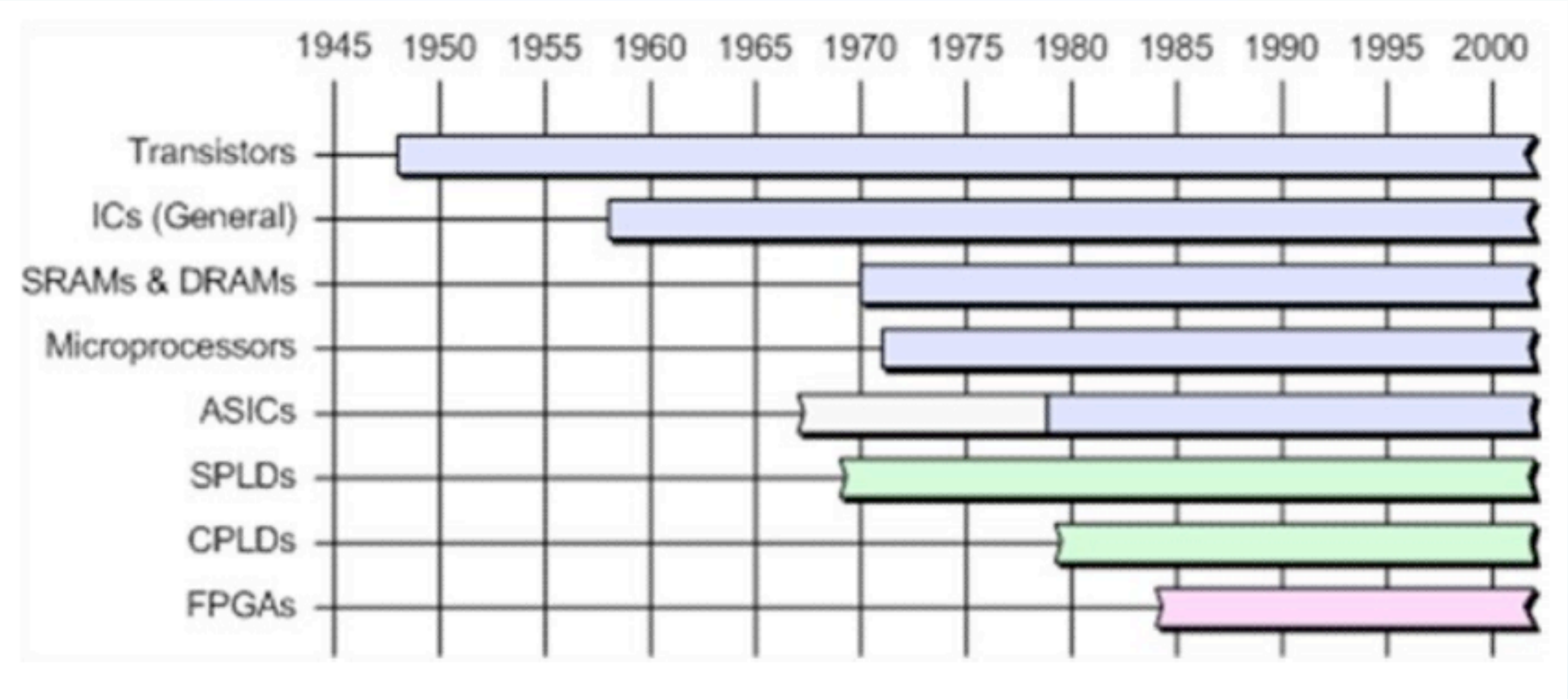
Complex PLDs (CPLDs)





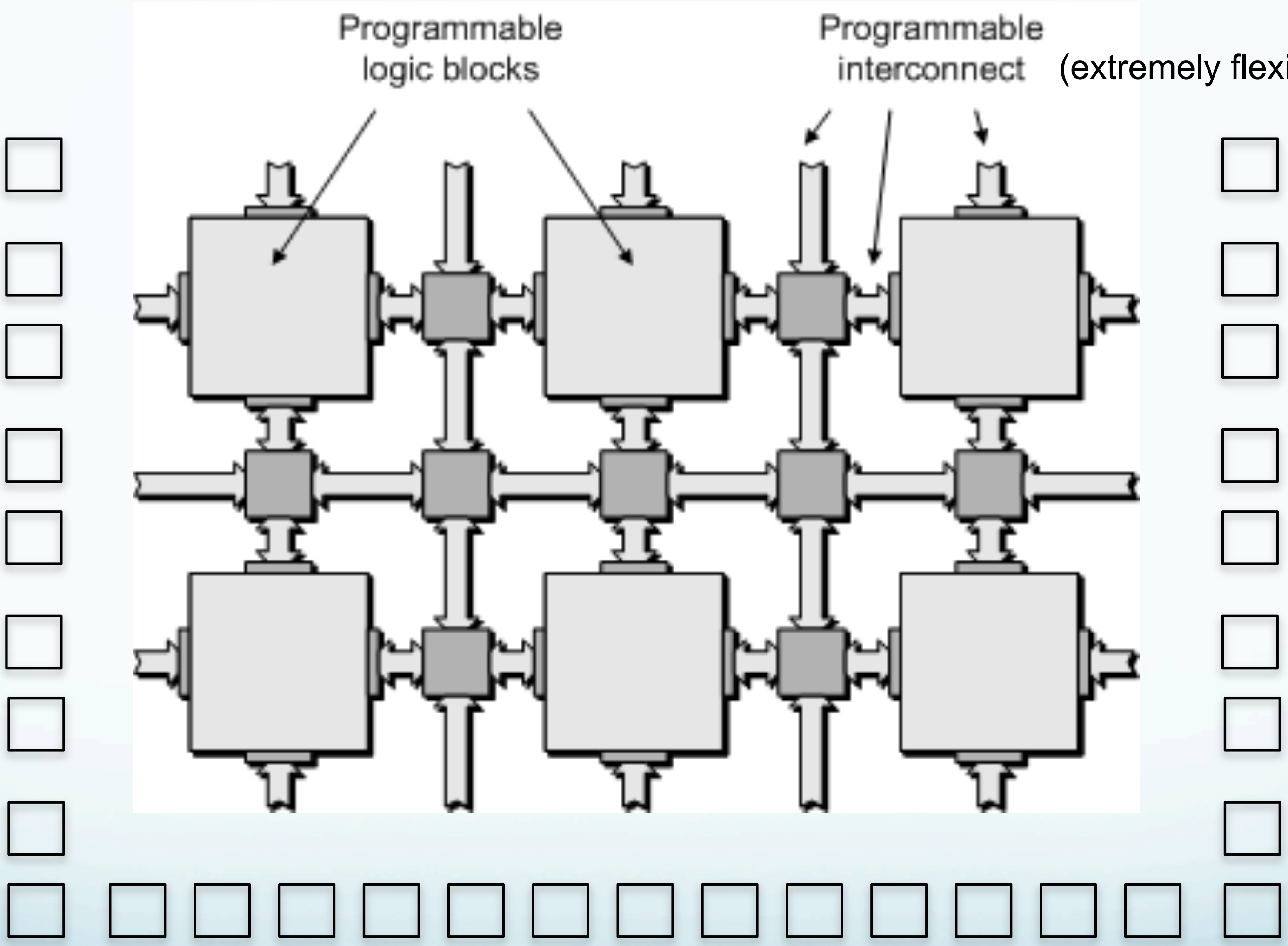
Coarse grained
100's of blocks, restrictive structure
(EE)PROM based

FPGAs ...



1945
1950
1955
1960
1965
1970
1975
1980
1985
1990
1995
2000
Transistors
ICs (General)
SRAMs & DRAMs
Microprocessors
ASICs
SPLDs
CPLDs
FPGAs

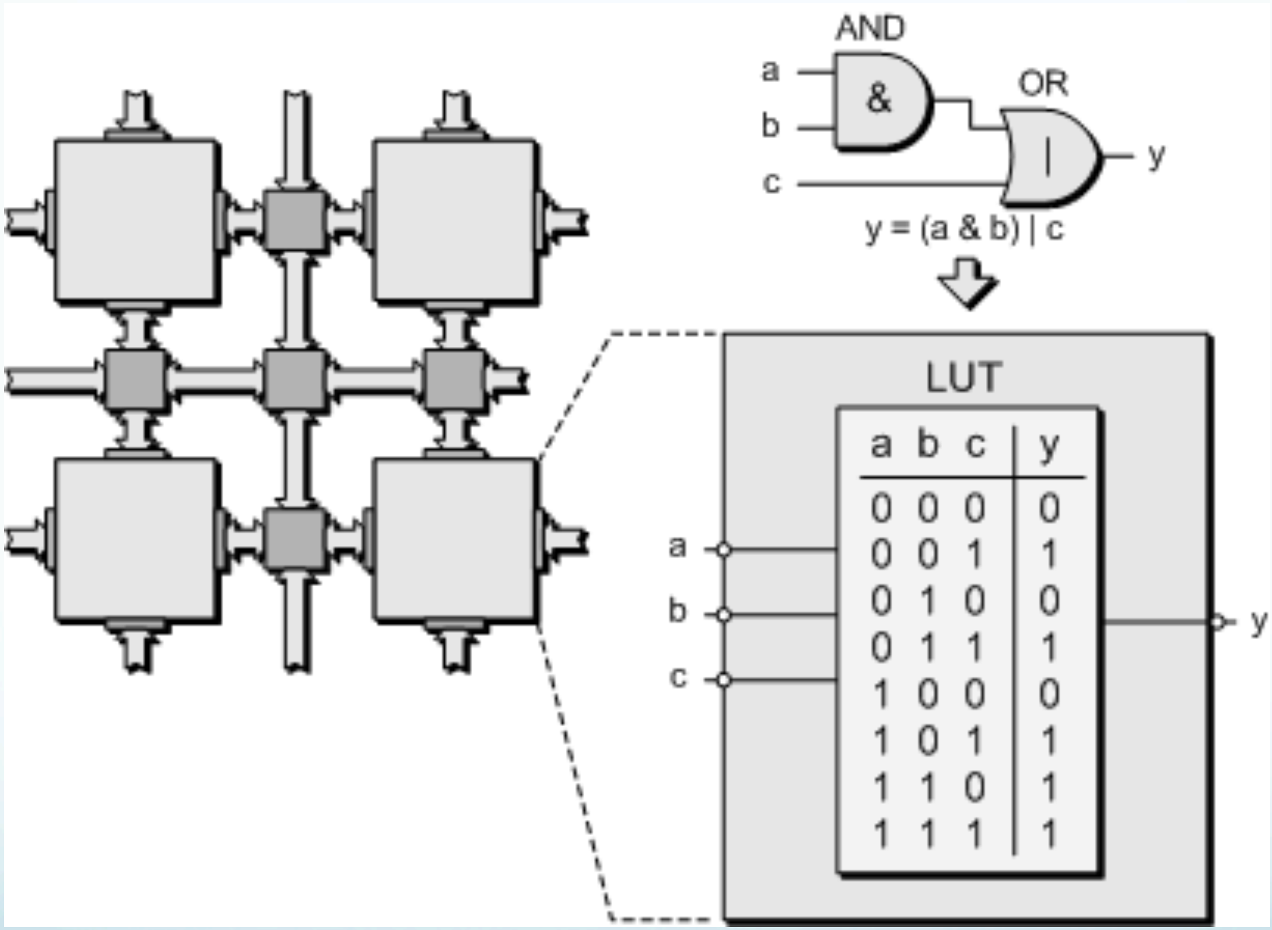
FPGAs



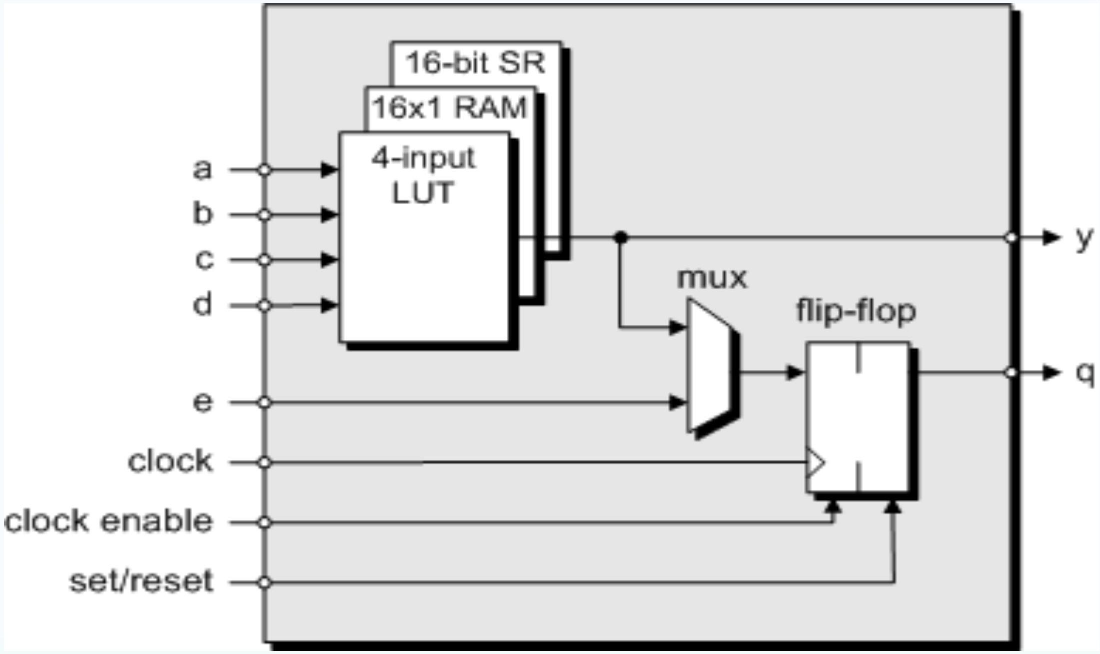
Fine-grained: 100.000's of blocks
today: up to 2 million logic blocks

Programmable Input / Output pins

LUT-based Fabrics



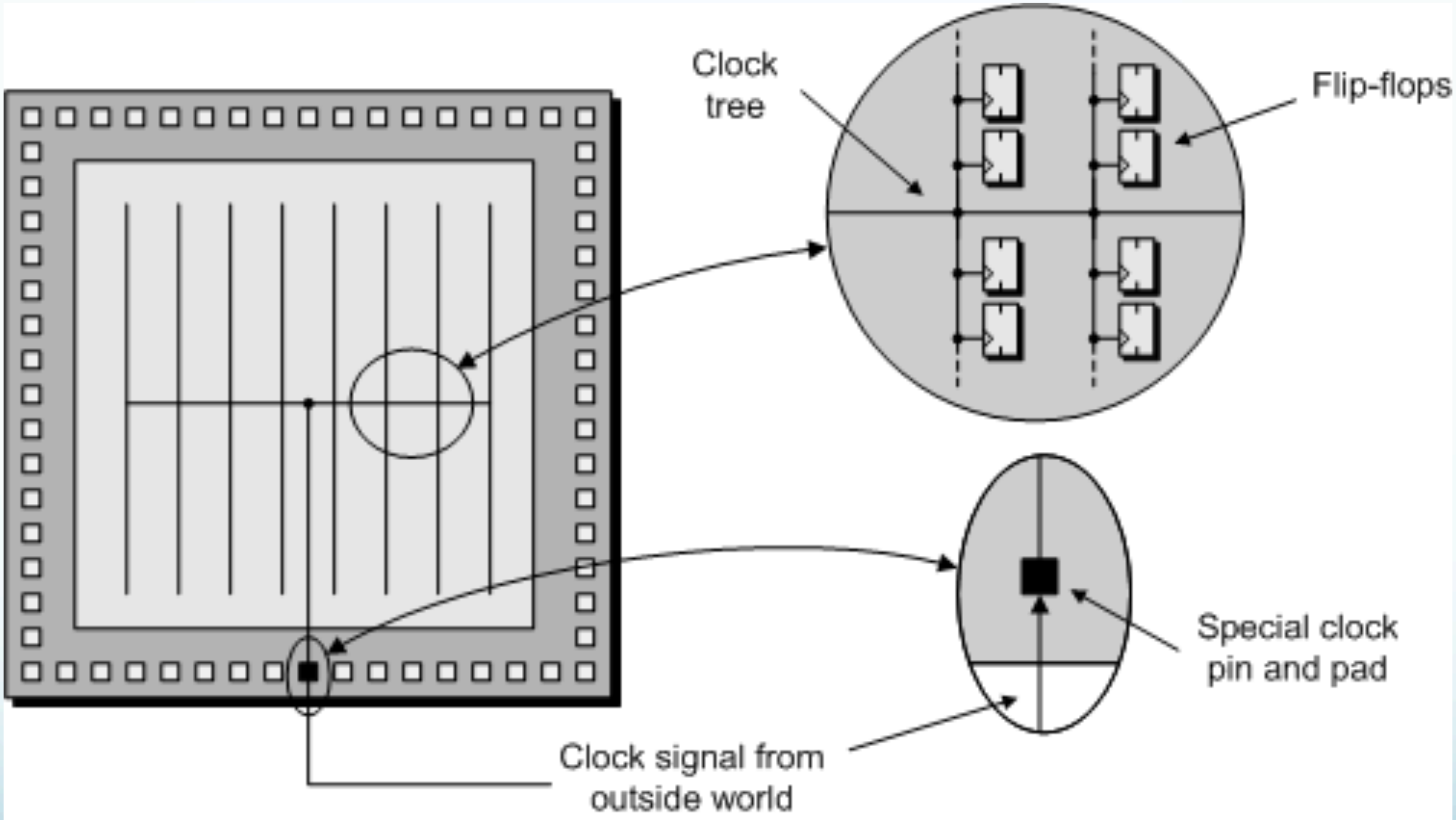
Typical LUT-based Logic Cell



Xilinx: logic cell,
Altera: logic element

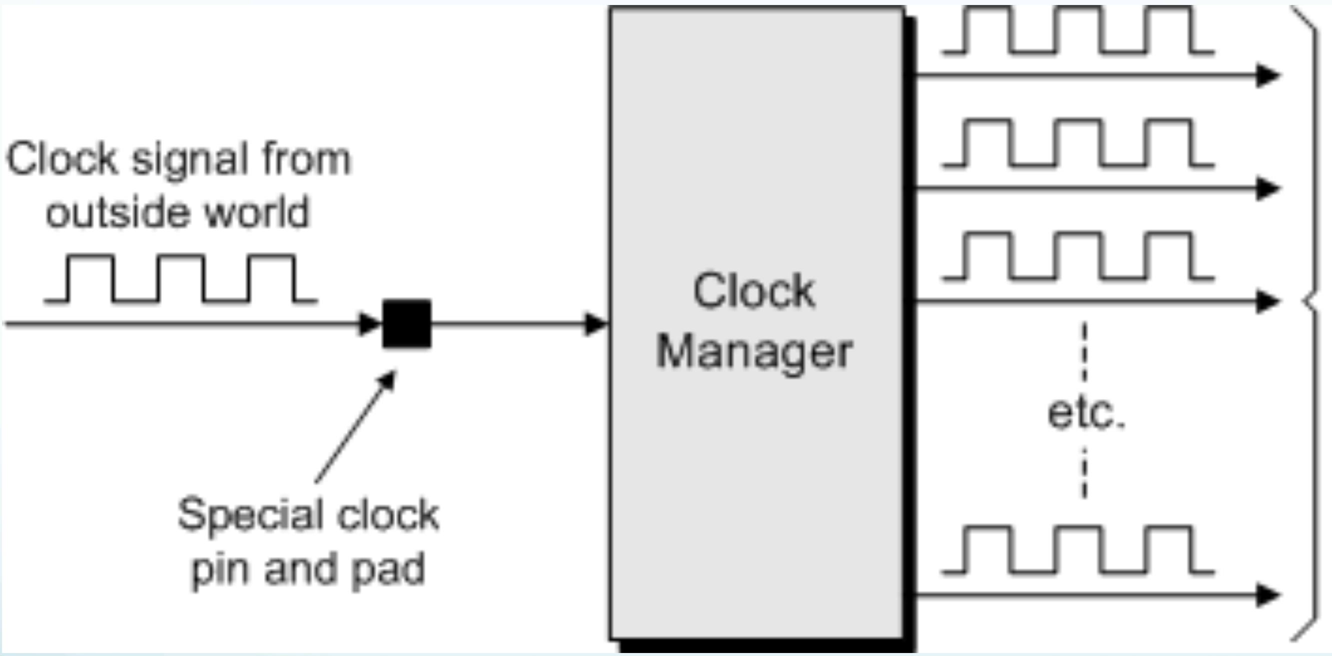
- LUT may implement any function of the inputs
- Flip-Flop registers the LUT output
- May use only the LUT or only the Flip-flop
- LUT may alternatively be configured a shift register
- Additional elements (not shown): fast carry logic

Clock Trees



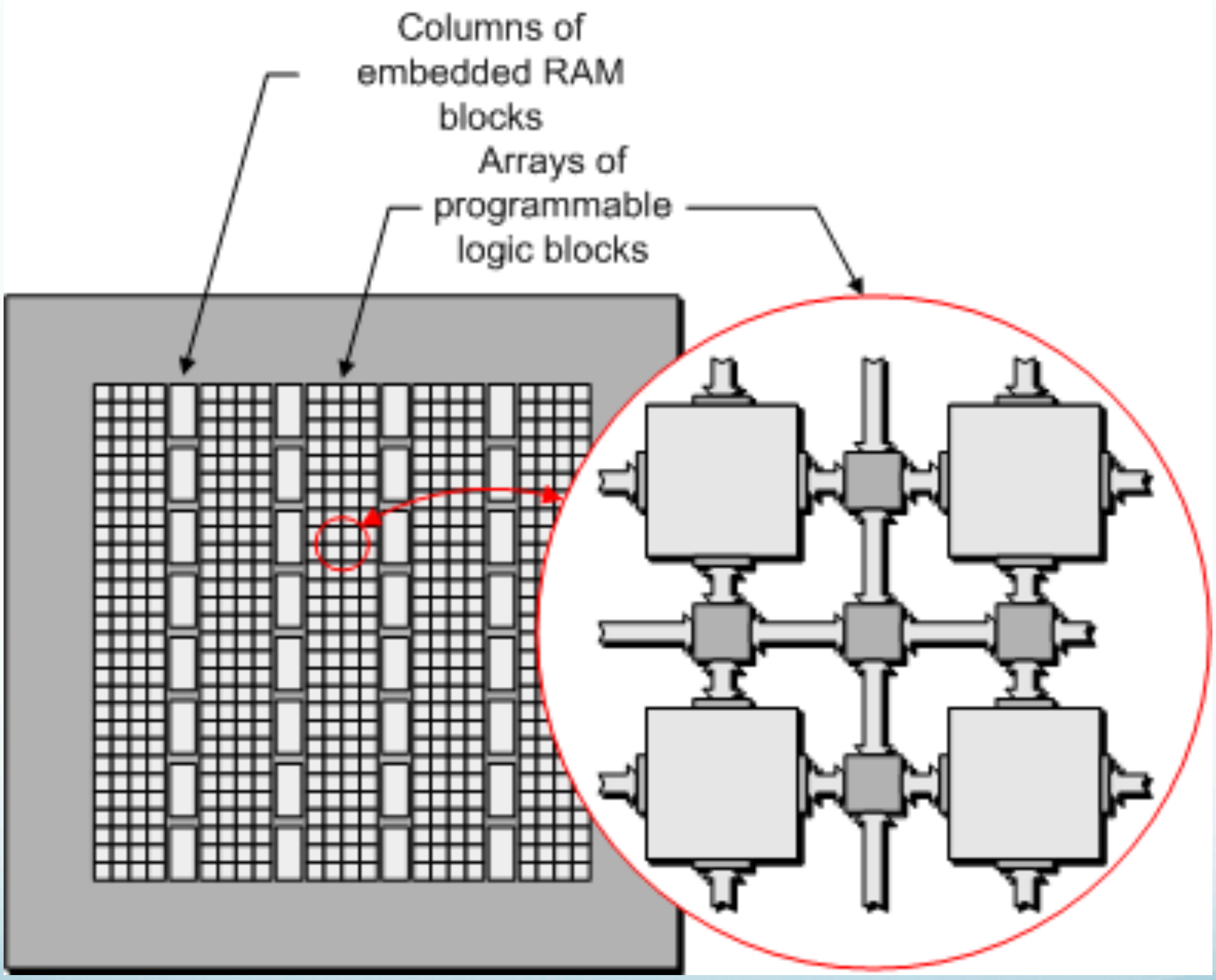
Clock trees guarantee that the clock arrives at the same time at all flip-flops

Clock Managers



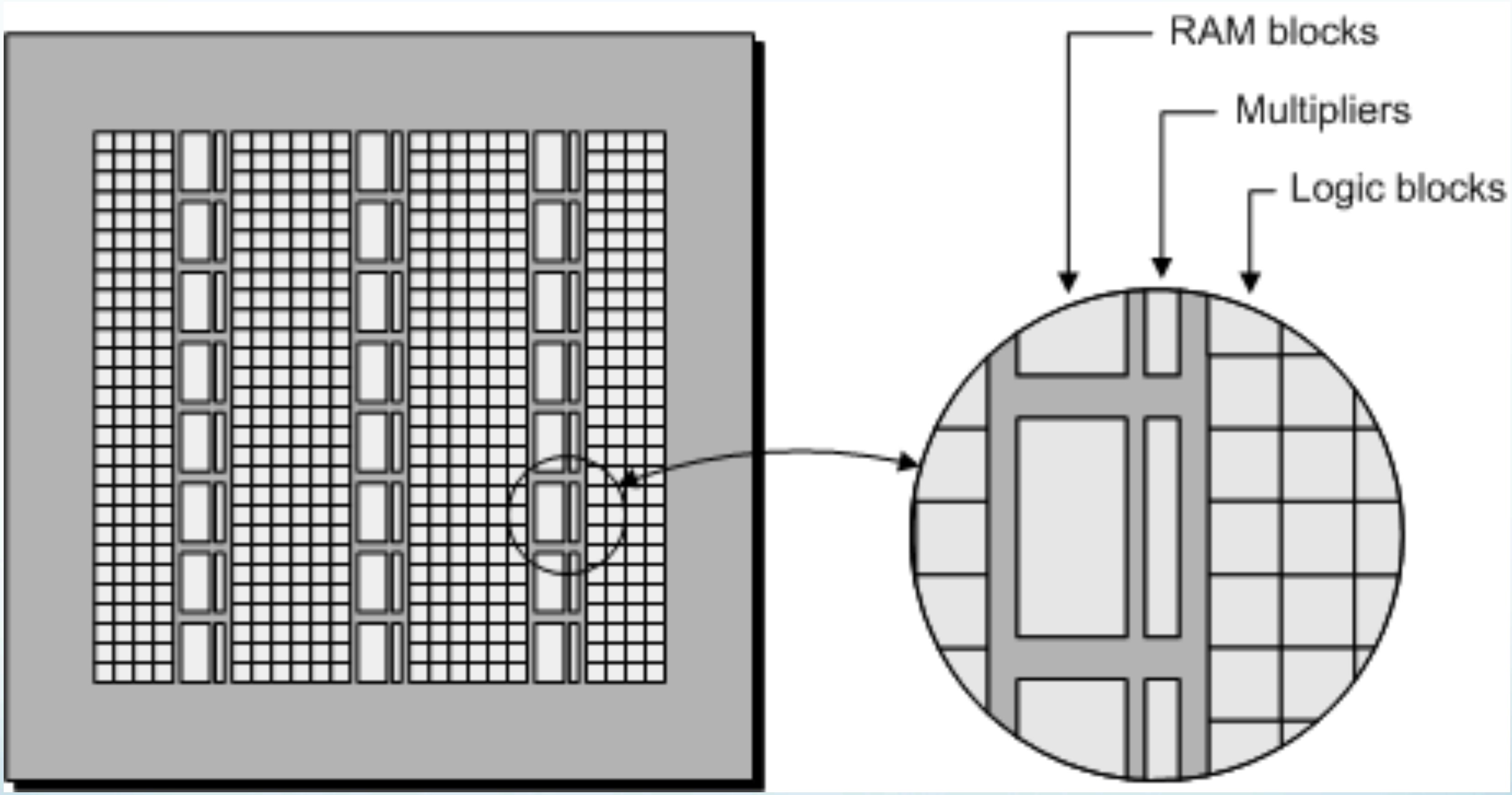
Daughter clocks may have multiple or fraction of the frequency

Embedded RAM blocks



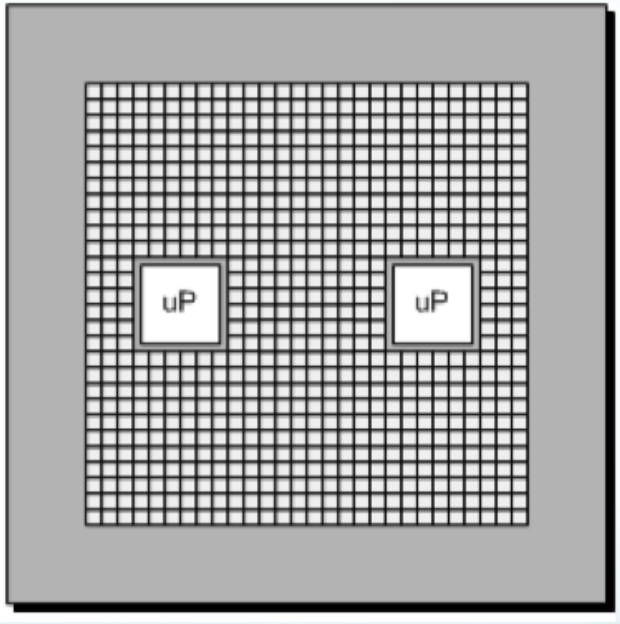
Today: Up to ~100 Mbit of RAM

Embedded Multipliers & DSPs

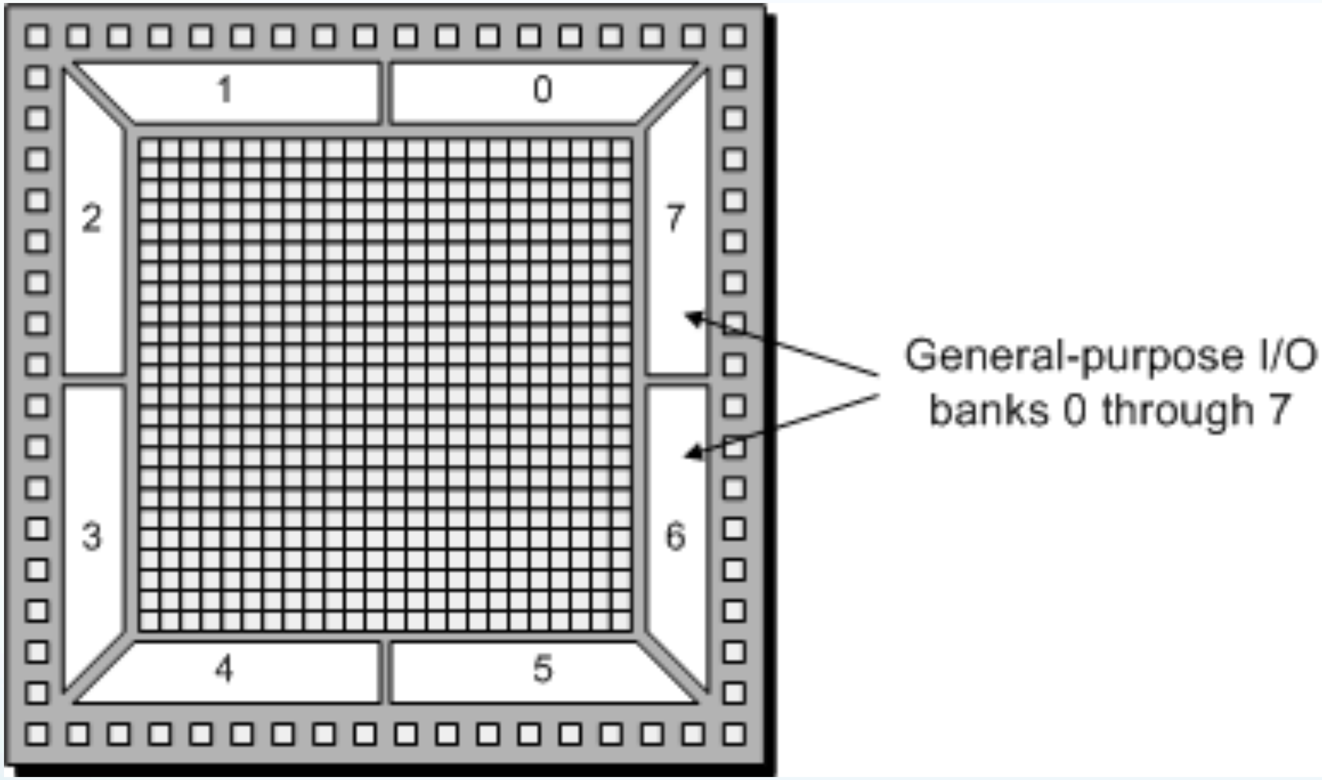


Soft and Hard Processor Cores

- 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)

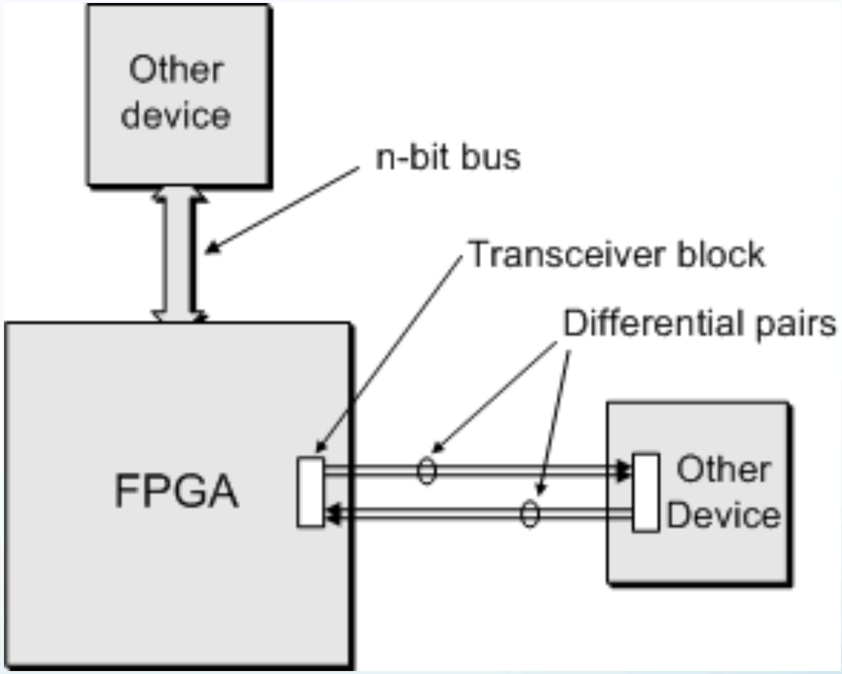


Today: Up to 1200 user I/O pins

- Input and / or output
- IO standard (such as LVTTL, LVDS) programmable
- Single signals or differential pairs

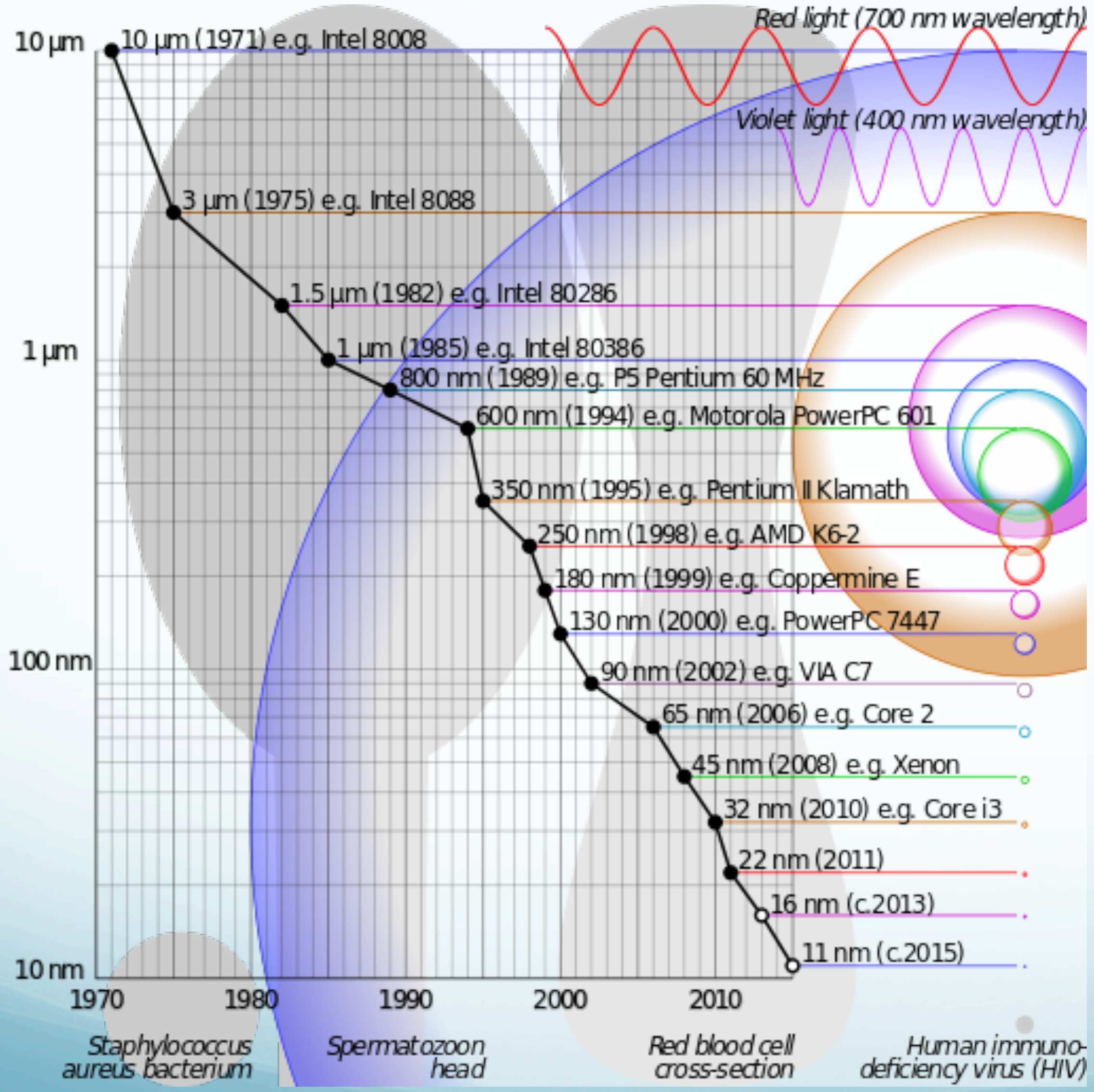
High-Speed Serial Interconnect

- Using differential pairs
- Latest serial transceivers:
  up to 28 Gb/s
  88 per FPGA = (2.8 Tb/s)
- PCI Express
  - Gen2: 5 Gb/s per lane
  - Gen3: 8 Gb/s per lane
    up to 8 lanes / FPGA
  - Hard and soft macros
- 10 Gb/s, 40 Gb/s, 100 Gb/s Ethernet



Trends

Ever-decreasing feature size



- Higher capacity
- Higher speed

130 nm
Xilinx Virtex-2

28 nm
Xilinx Virtex-7 / Altera Stratix V

2 million logic cells

Trends

- Look-up-tables with more inputs (5 or 6)
- Speed of serial links increasing (multiple Gb/s)
- More hard macro cores (Ethernet MAC, Memory interfaces, PCI express …)
- Sophisticated soft macros
  - CPUs
  - Gb/s MACs
  - Memory interfaces (DDR2/3)
  - CPUs
- Processor-centric architectures
  - Multi-core processor + FPGA logic (Xilinx ZYNQ)
- Domain-specific devices
- Ultra-low-power FPGAs
- Mixed-signal FPGAs

FPGA – ASIC comparison

FPGA

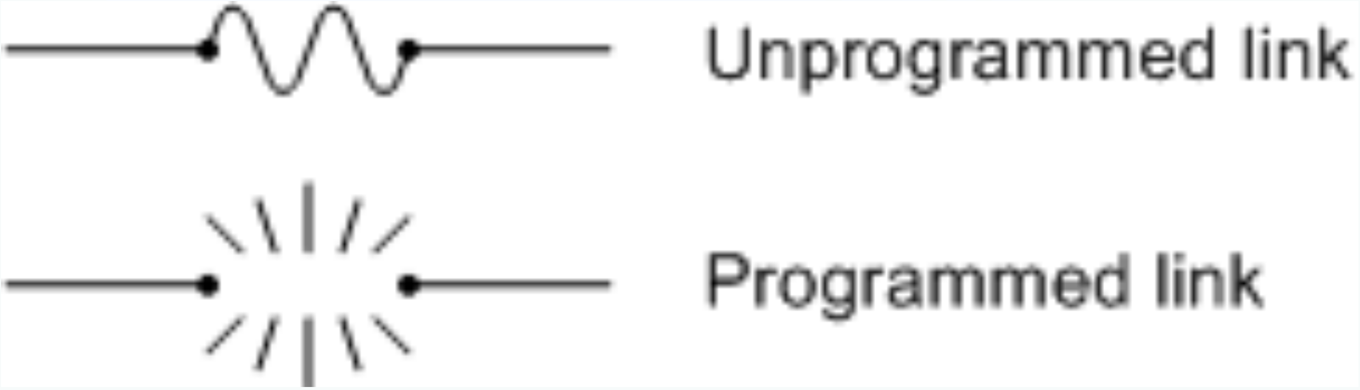
- Rapid development cycle (minutes / hours)
- May be reprogrammed in the field (firmware upgrade)
  - New features
  - Bug fixes
- Low development cost
  - You can get started with a $200 development board and free software

ASIC

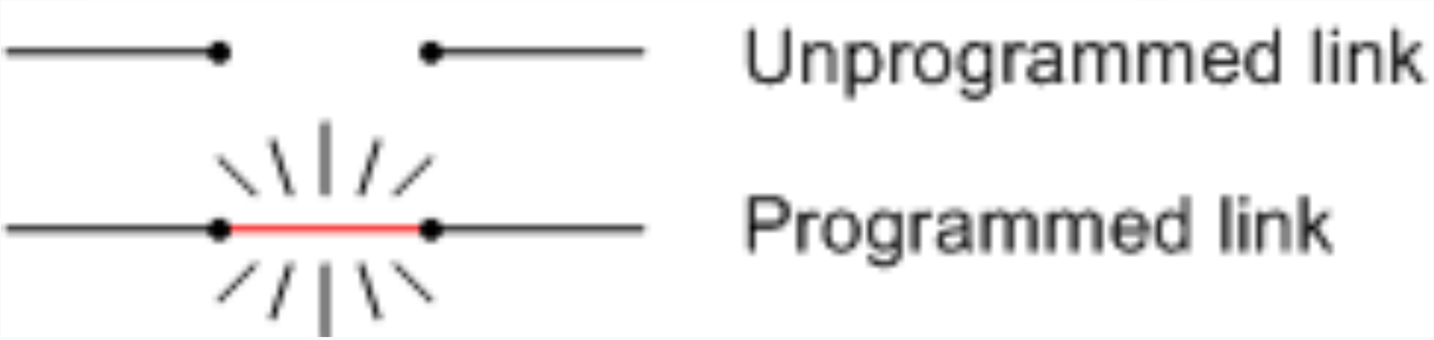
- Higher performance
- Analog designs possible
- Better radiation hardness
- Long development cycle (weeks / months)
- Design cannot be changed once it is produced
- Extremely high development cost
  - ASICs are produced at a semiconductor fabrication facility (“fab”) according to your design
- Lower cost per device compared to FPGA, when large quantities are needed

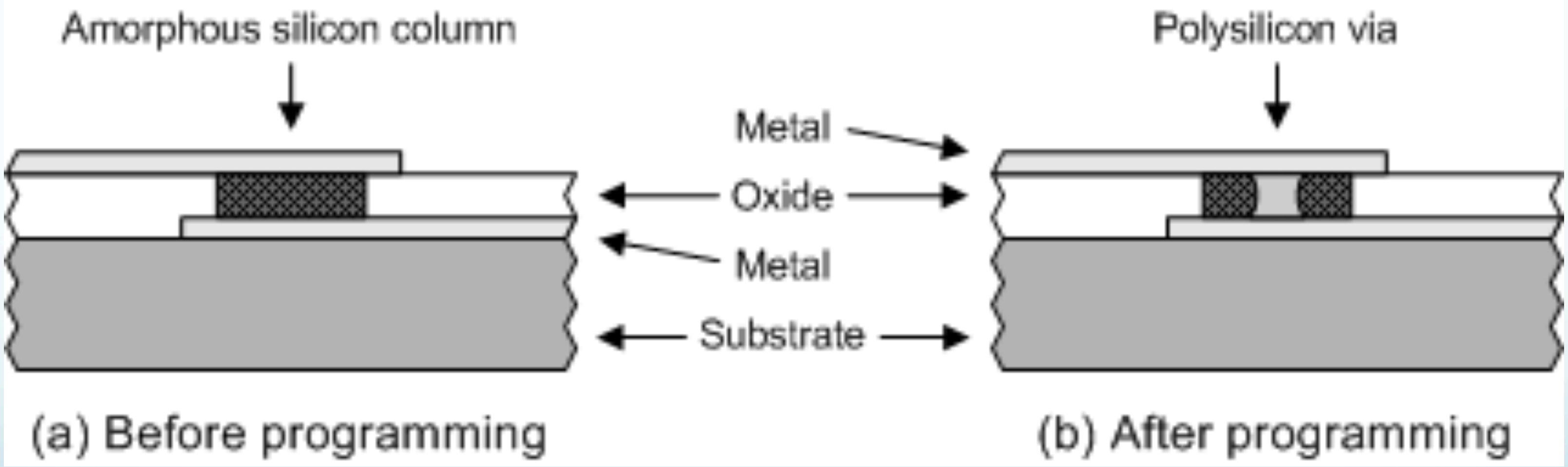
Programming techniques

Fusible Links (not used in FPGAs)



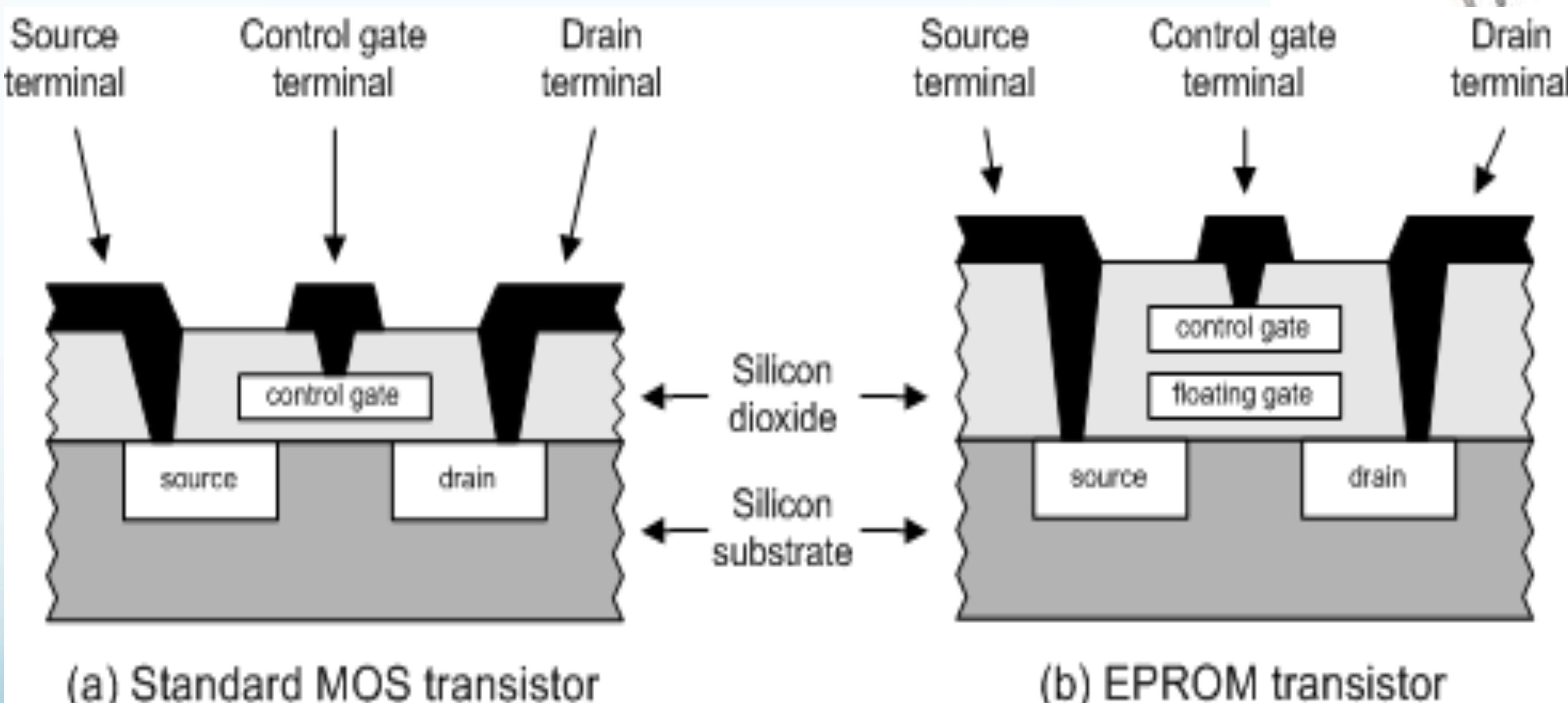
Antifuse Technology





EPROM Technology

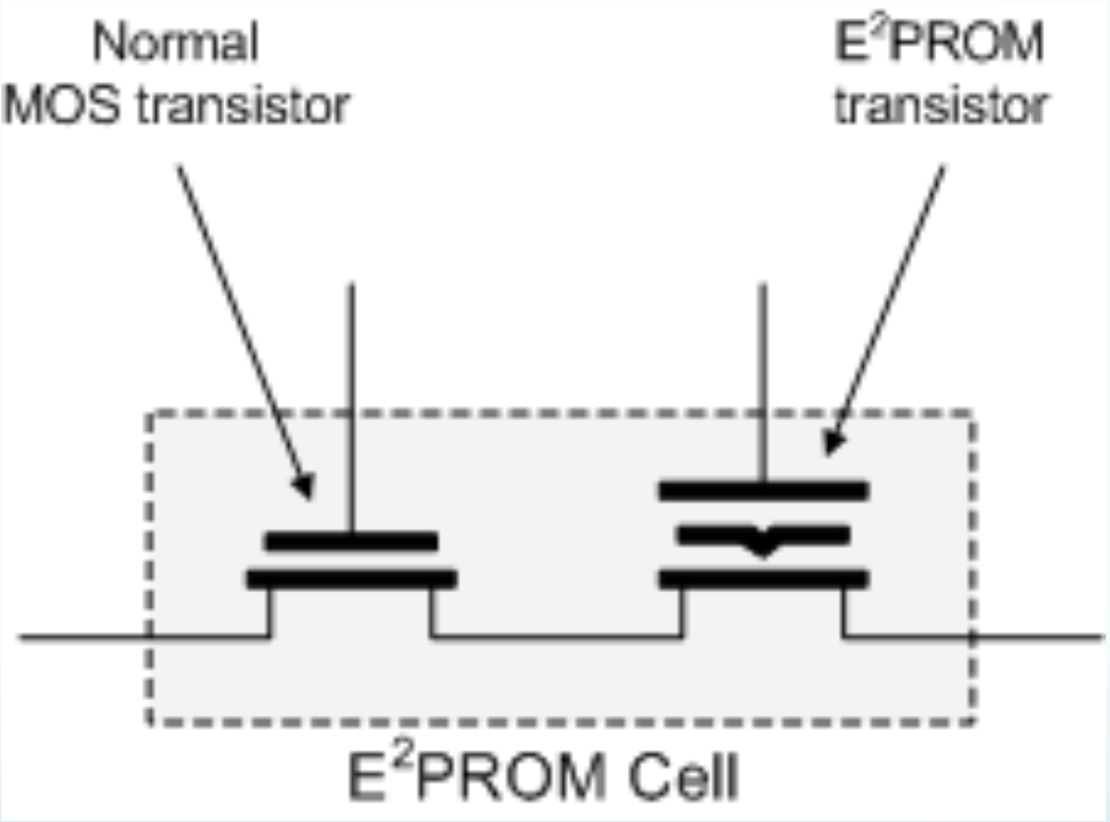
Erasable Programmable Read Only Memory



Intel, 1971

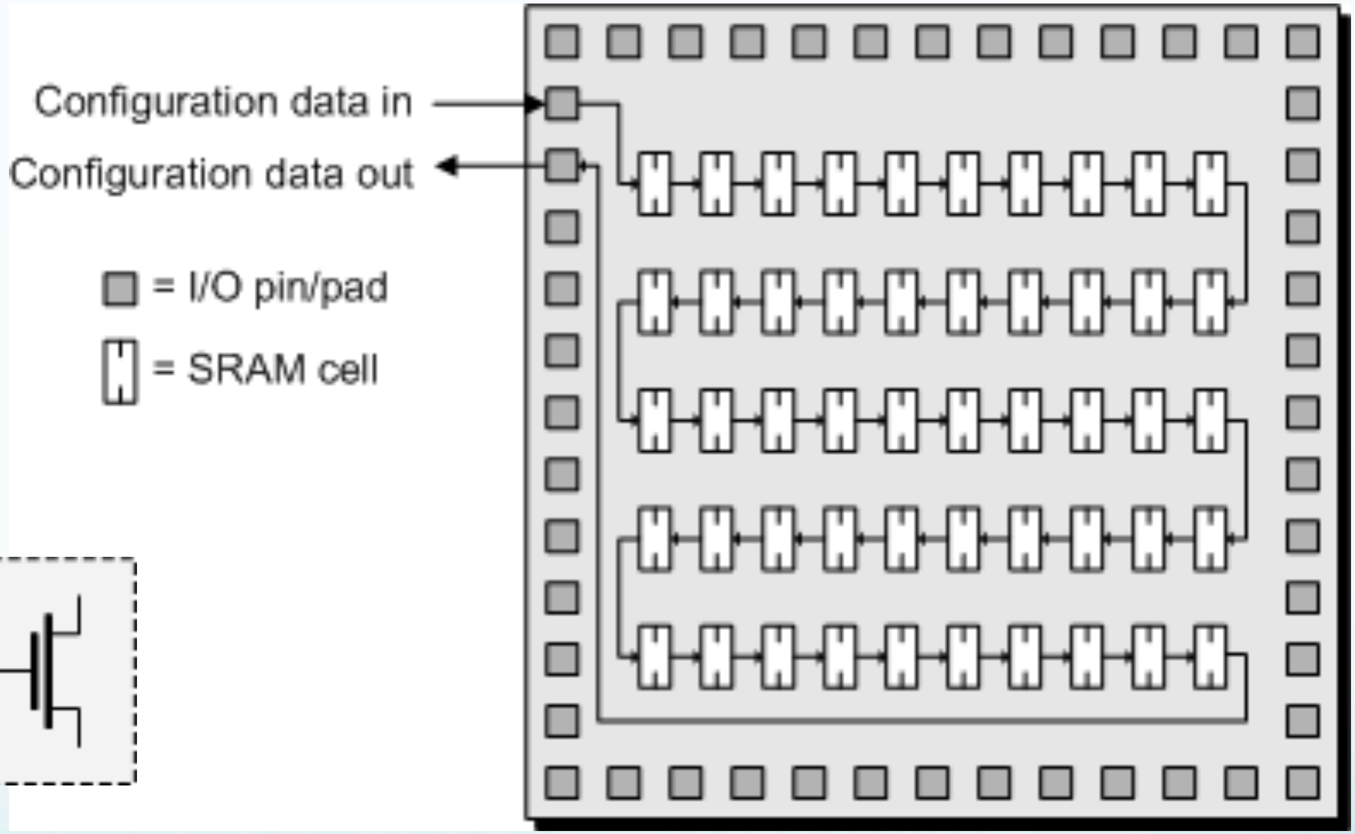
EEPROM and FLASH Technology

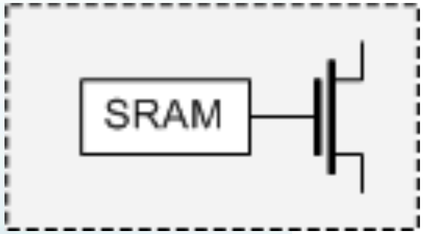
Electrically Erasable Programmable Read Only Memory



EEPROM: erasable word by word
FLASH: erasable by block or by device

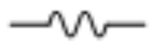
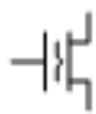
SRAM-Based Devices





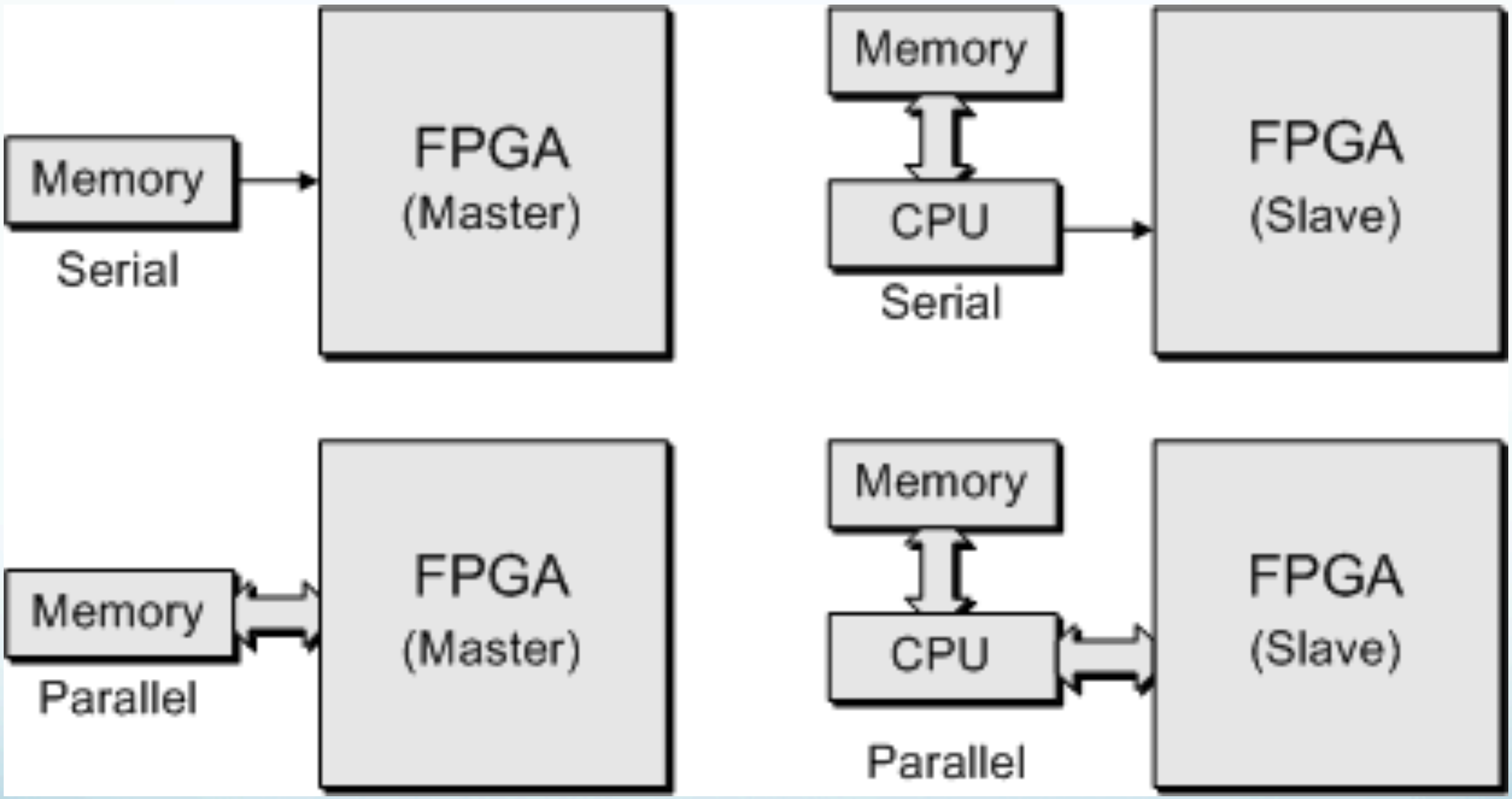
Multi-transistor SRAM cell

Summary of Technologies

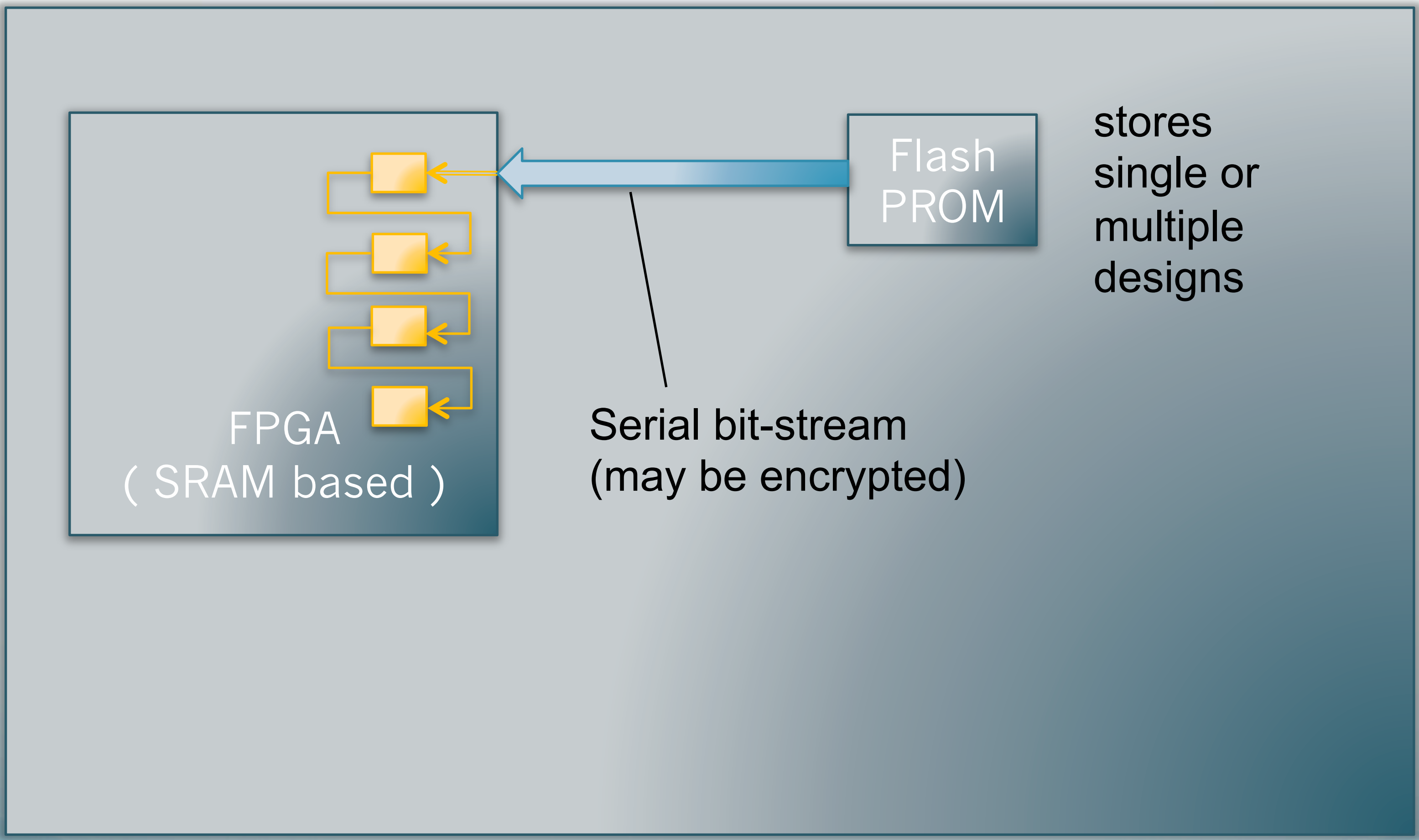
Technology	Symbol	Predominantly associated with ...
Fusible-link		SPLDs
Antifuse		FPGAs
EPROM		SPLDs and CPLDs
E^2PROM/ FLASH		SPLDs, CPLDs, and FPGAs
SRAM		FPGAs (some CPLDs)

Used in most FPGAs

Design Considerations (SRAM Config.)



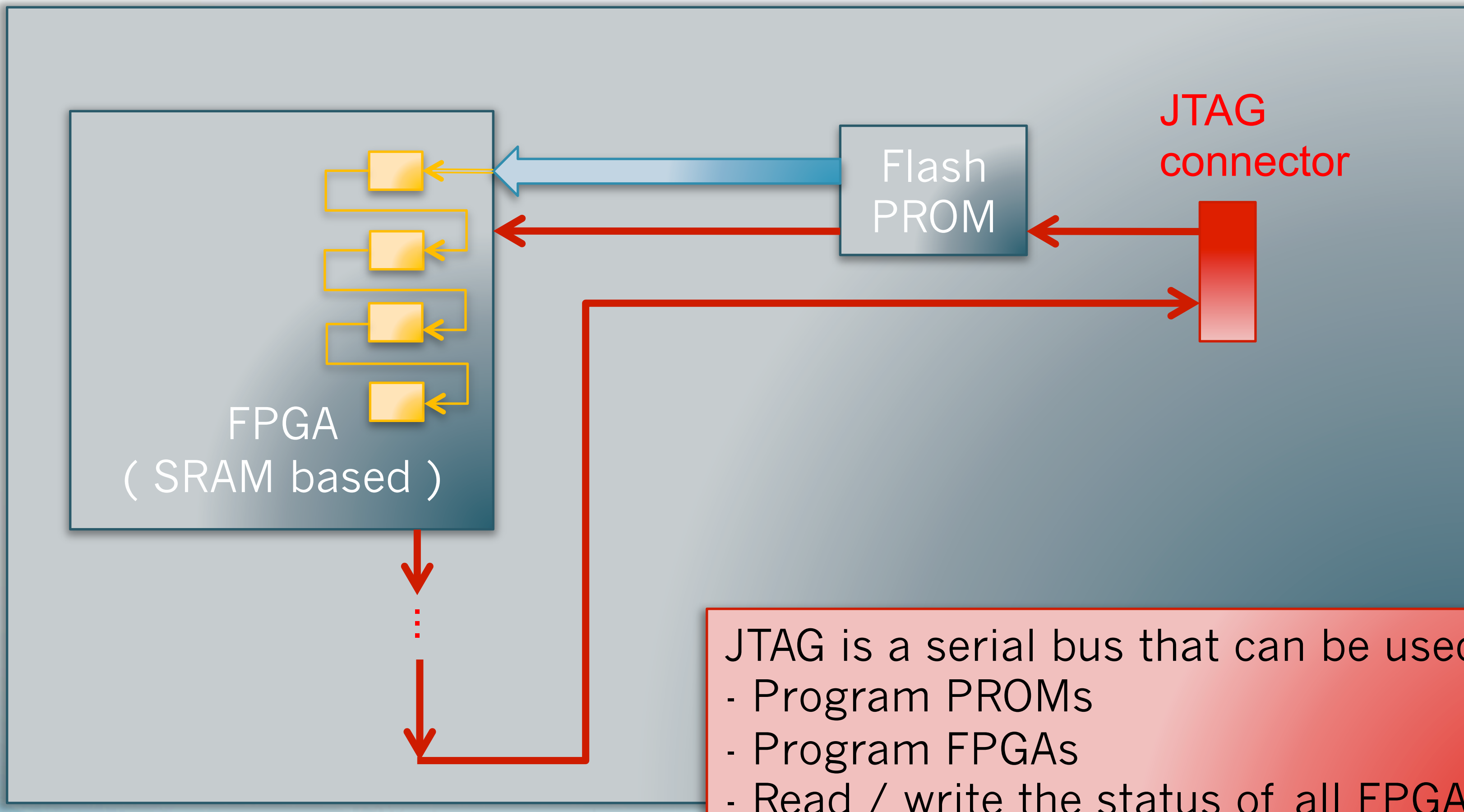
Configuration at power-up



Typical FPGA configuration time: milliseconds

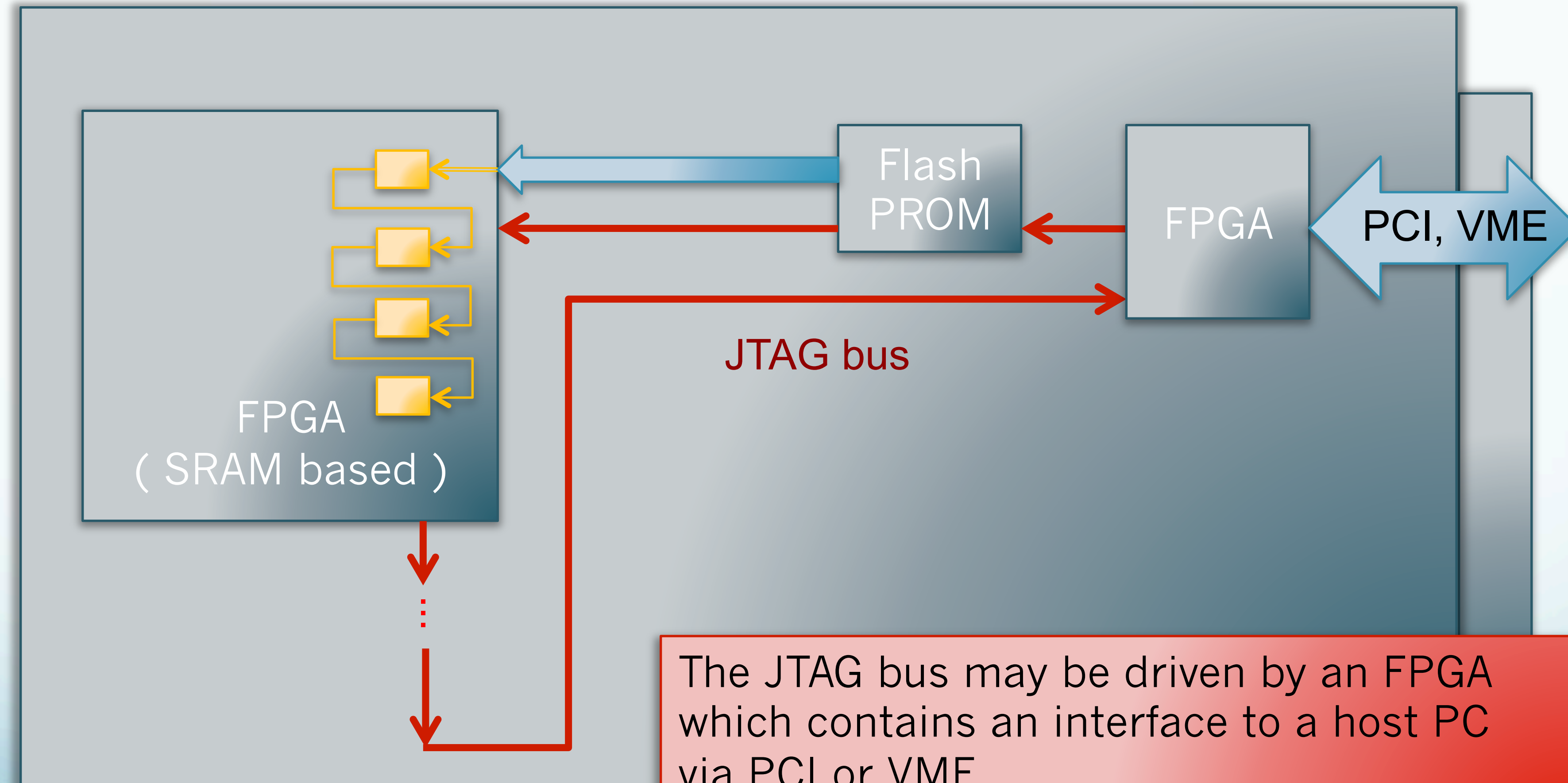
Programming via JTAG

Joint Test Action Group



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

Programming from a host PC



The JTAG bus may be driven by an FPGA which contains an interface to a host PC via PCI or VME
Firmware can then be updated from the PC

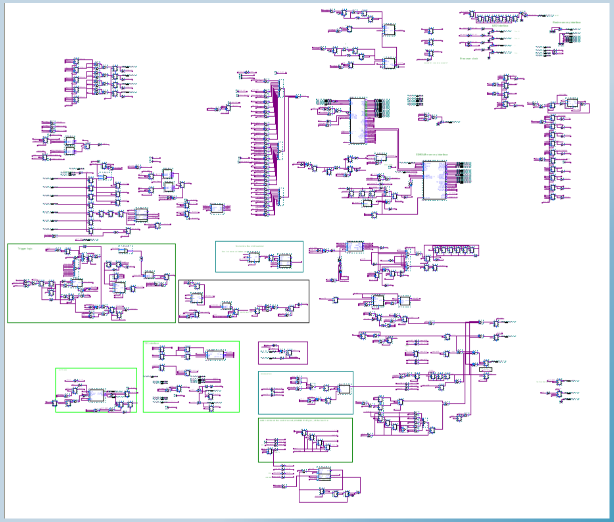
Major Manufacturers

- Xilinx
  - First company to produce FPGAs in 1985
  - About 50% market share, today
  - SRAM based CMOS devices
- Altera
  - About 40% market share
  - SRAM based CMOS devices
- Actel
  - Anti-fuse FPGAs
  - Flash based FPGAs
  - Mixed Signal
- Lattice Semiconductor
  - SRAM based with integrated Flash PROM

FPGA development

Design entry

Schematics



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

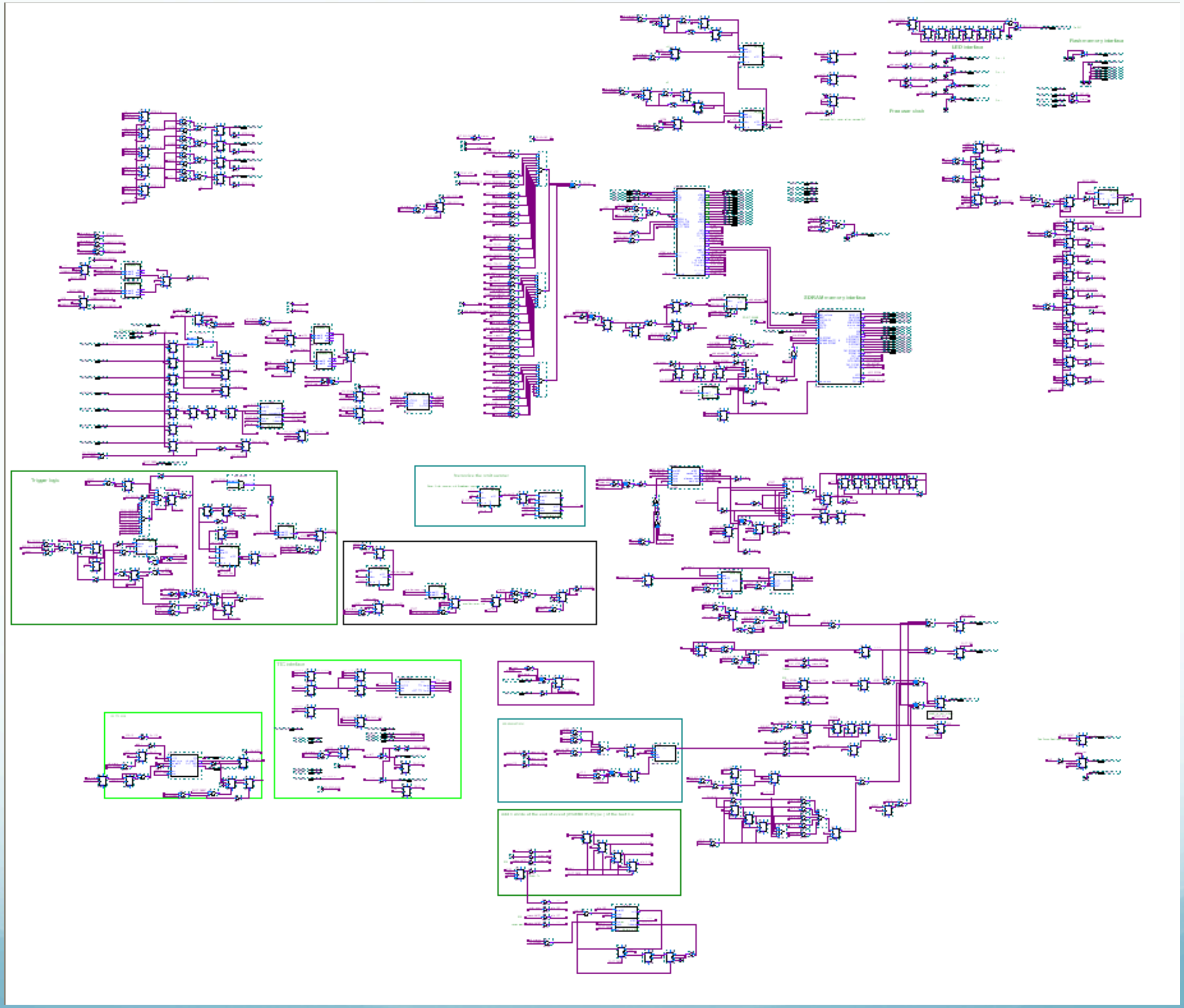
Hardware description language
VHDL, Verilog

```
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 (CVS, SVN ...)

Mostly a personal choice depending on previous experience

Schematics



Hardware Description Language

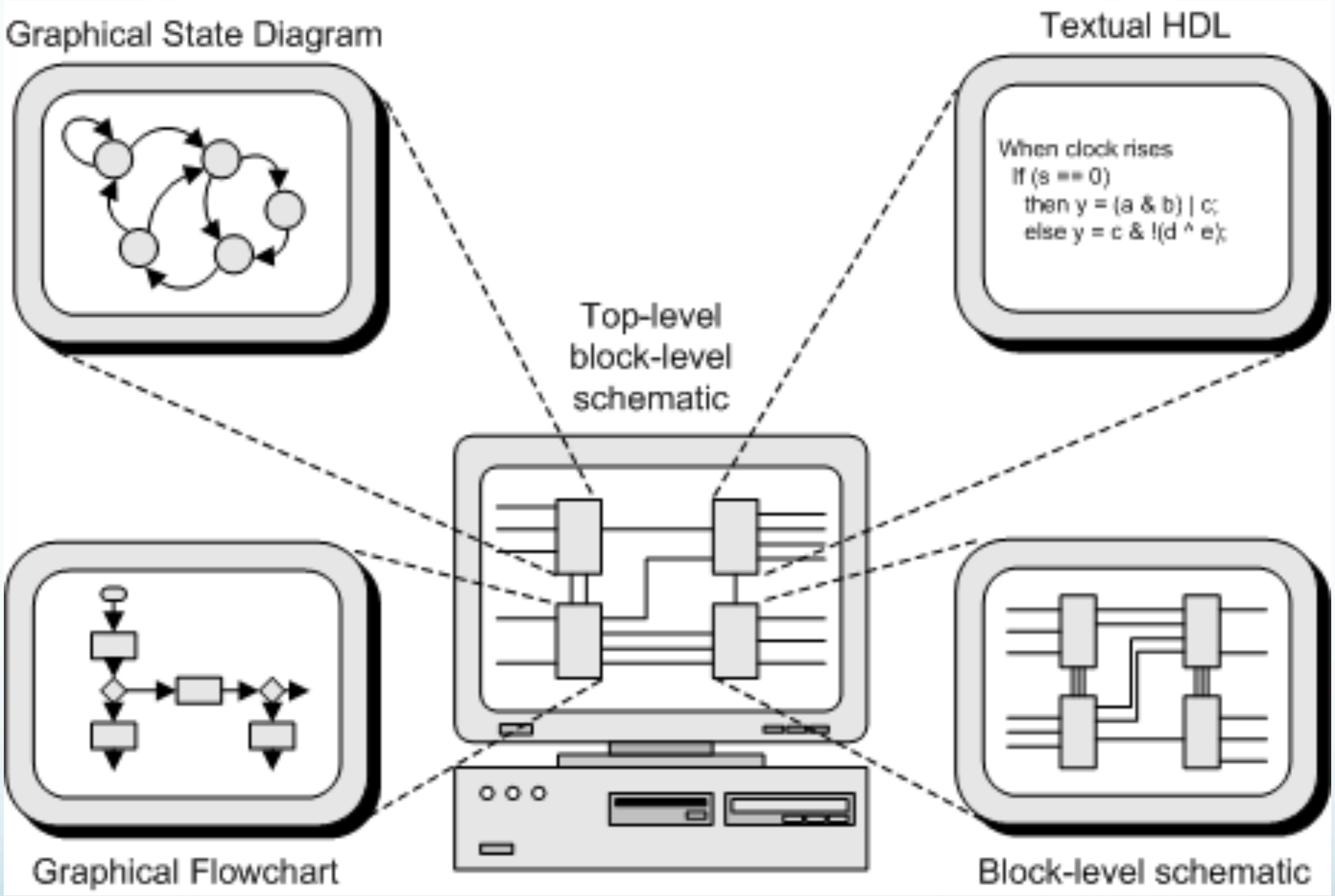
- Similar to a programming language
- Common HDLs
  - VHDL
  - Verilog
  - AHDL (Altera specific)
- Newer trends
  - C-like languages (handle-C)
  - Labview

Example: VHDL

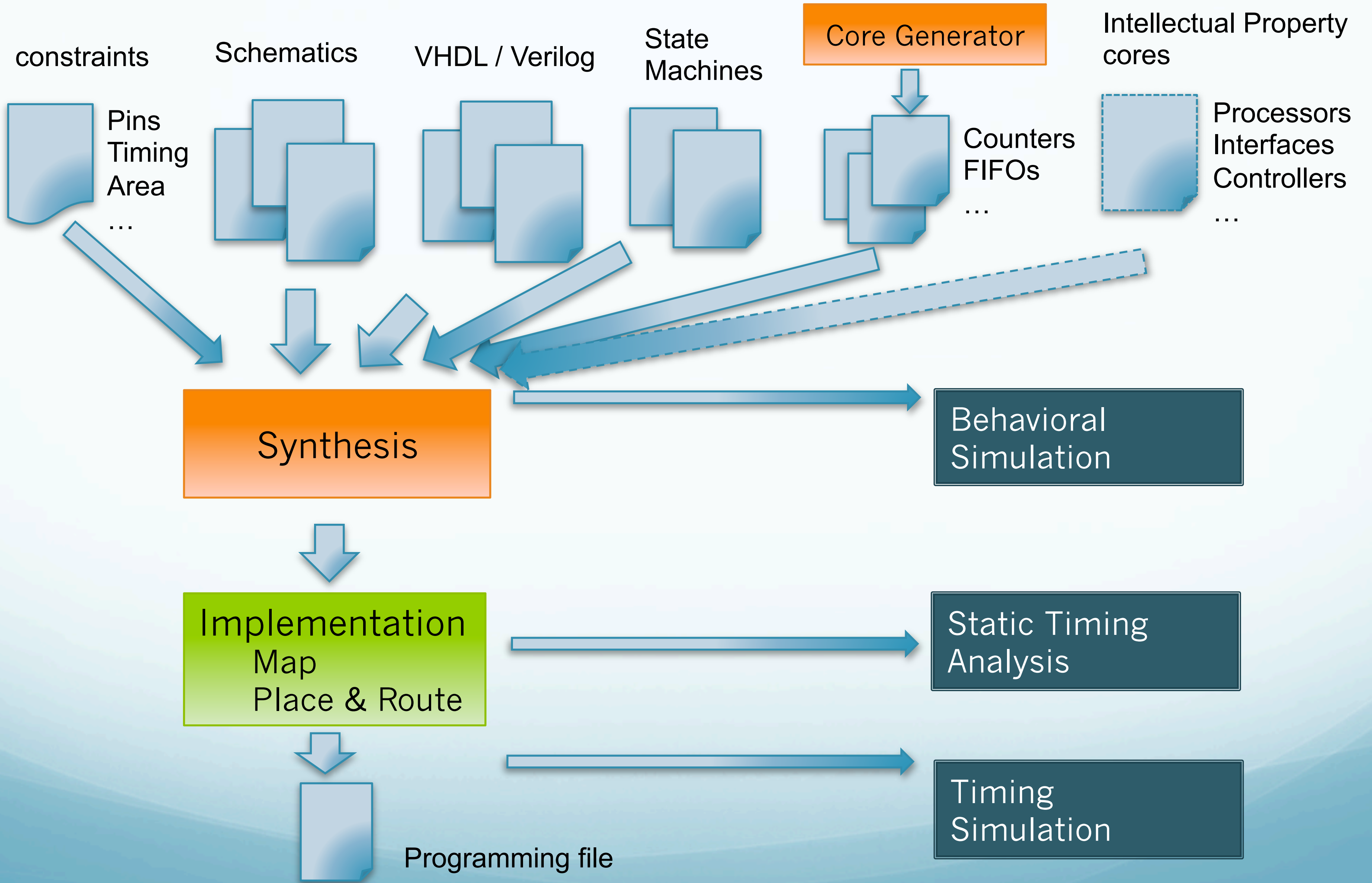
```
architecture behavioral of VMEReg is

  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

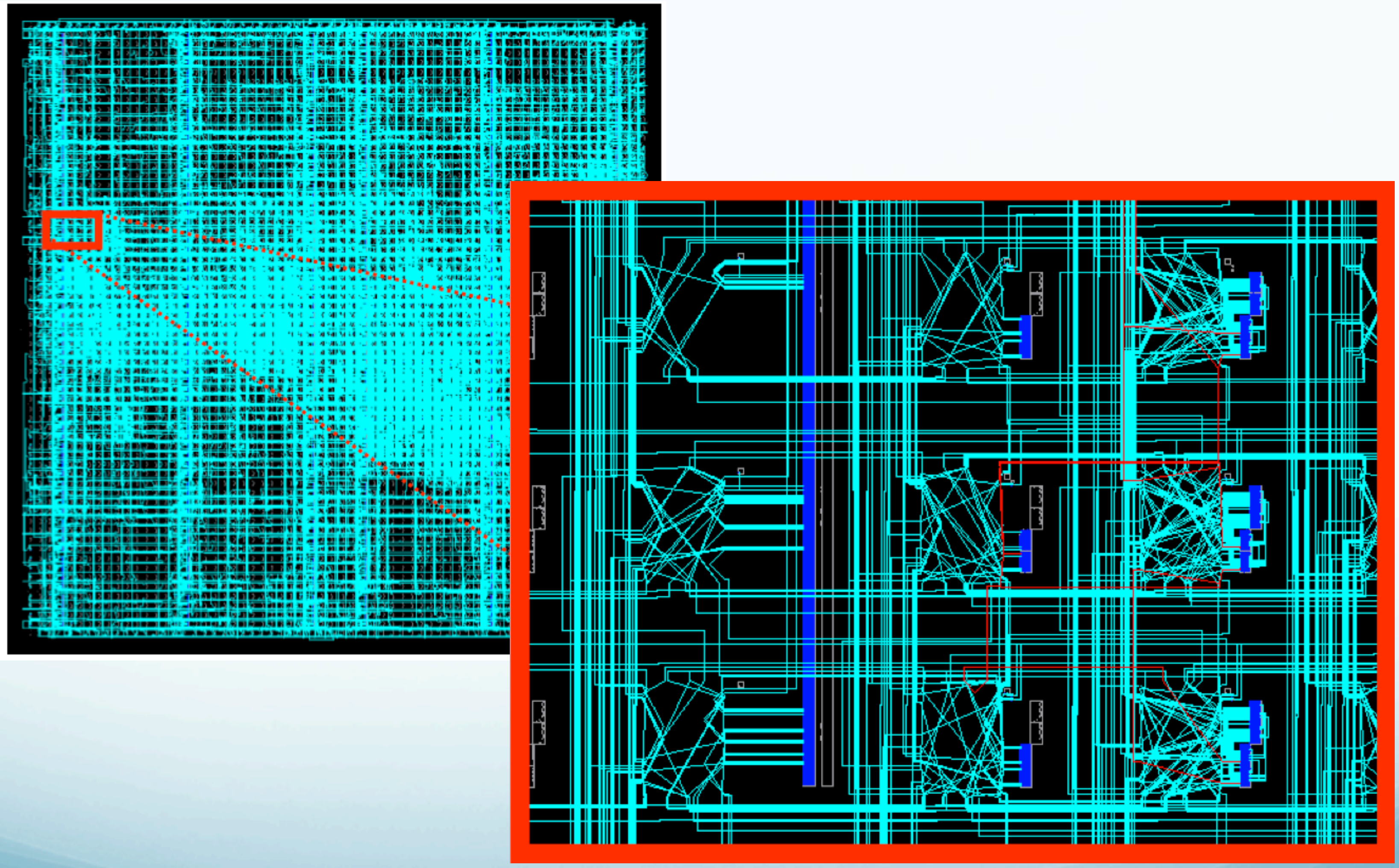
Schematics & HDL combined



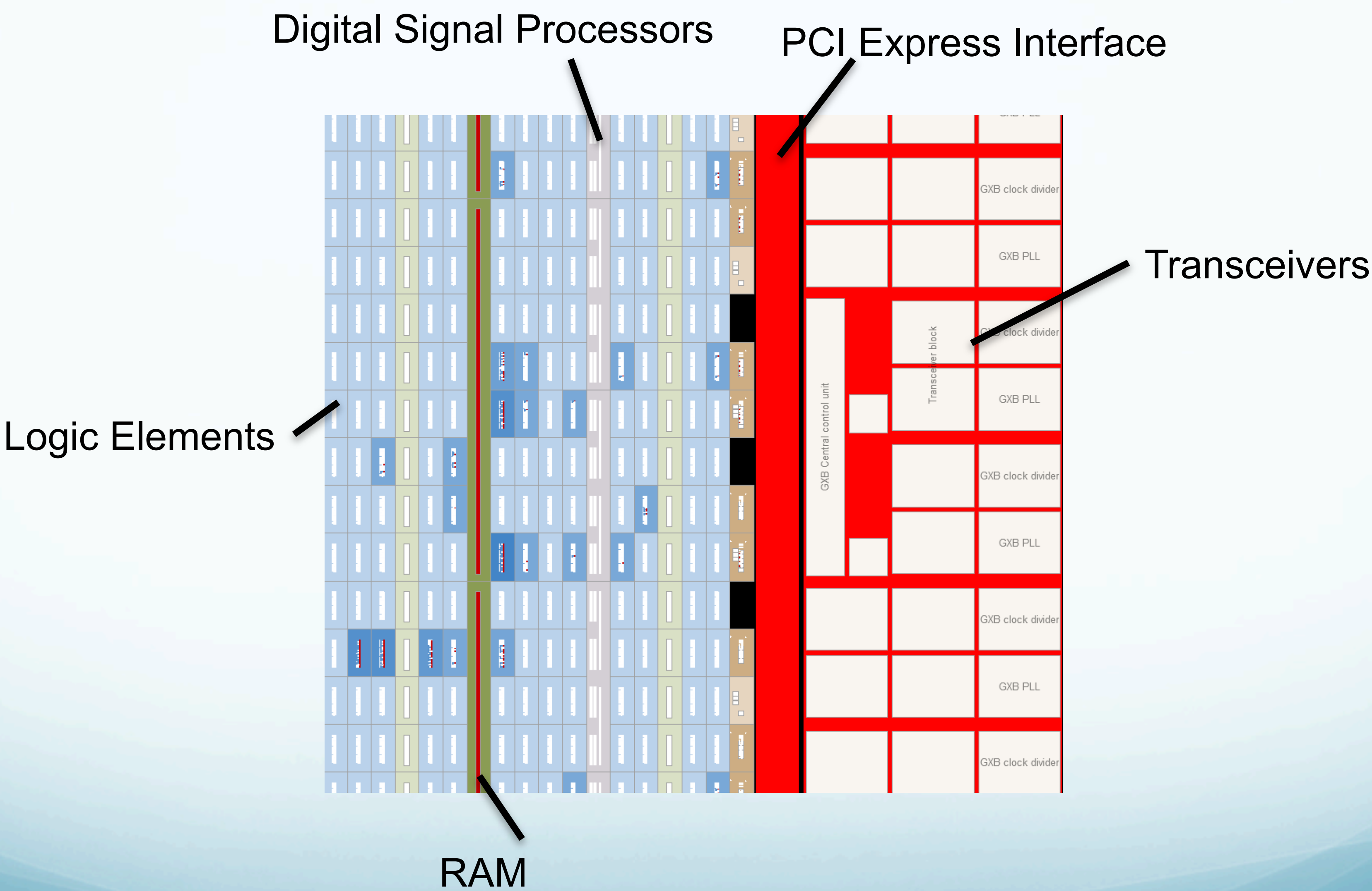
Design flow


constraints
Schematics
VHDL / Verilog
State Machines
Core Generator
Intellectual Property cores
Pins
Timing
Area
…
Counters
FIFOs
…
Processors
Interfaces
Controllers
…
Synthesis
Behavioral Simulation
Implementation
Map
Place & Route
Static Timing Analysis
Timing Simulation
Programming file

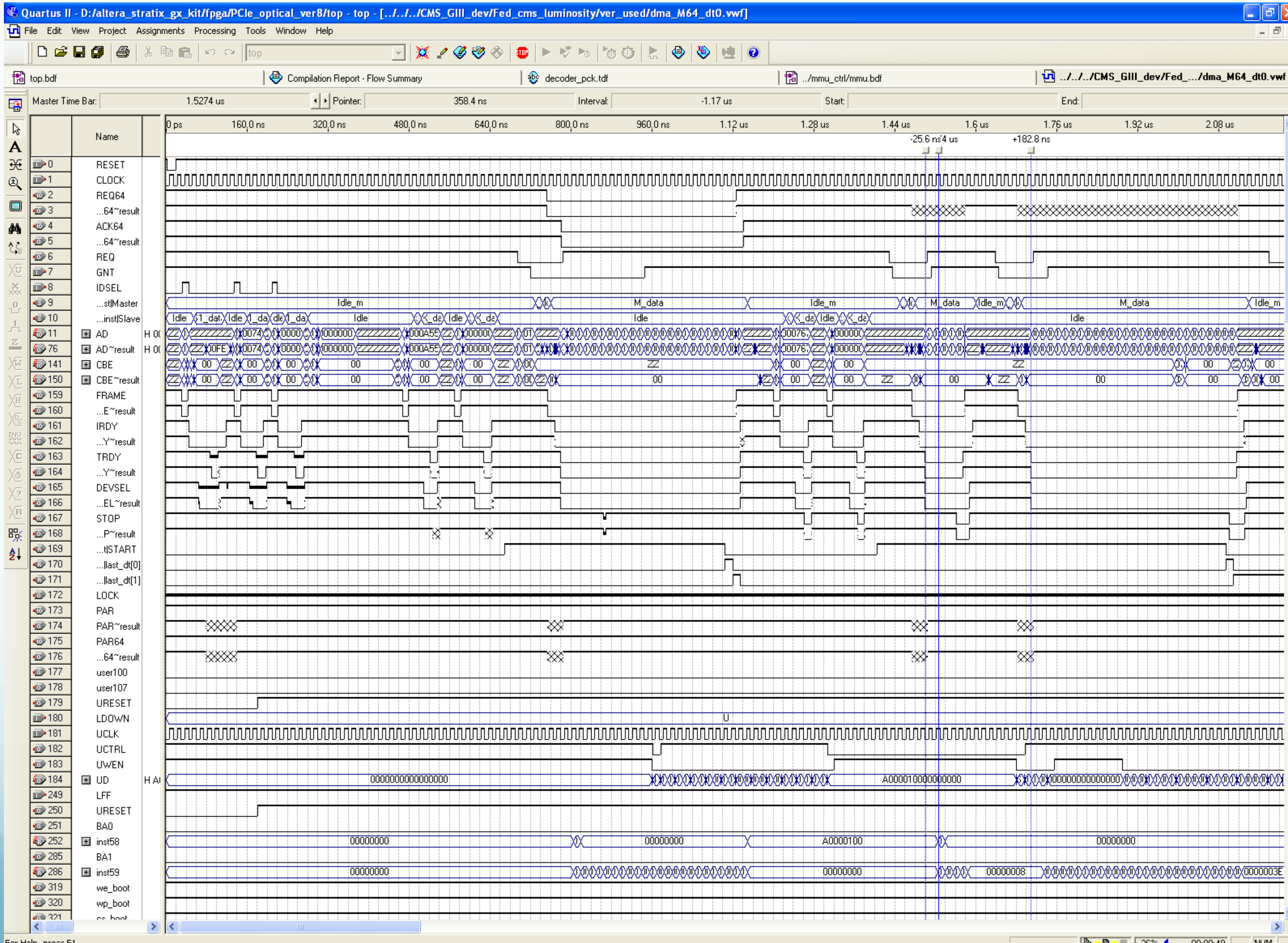
Floorplan (Xlinx Virtex 2)



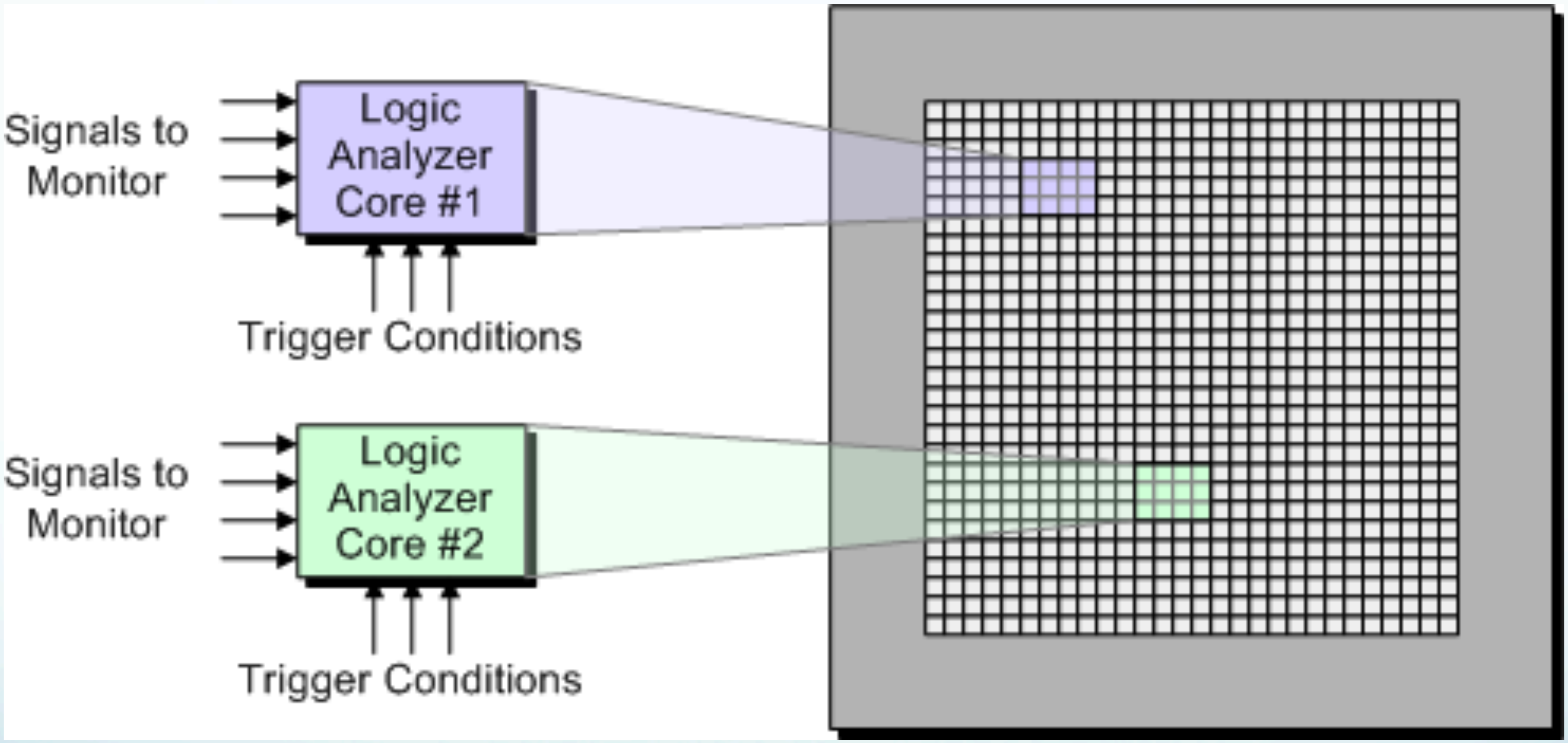
Floorplan (Altera Stratix 4)



Simulation

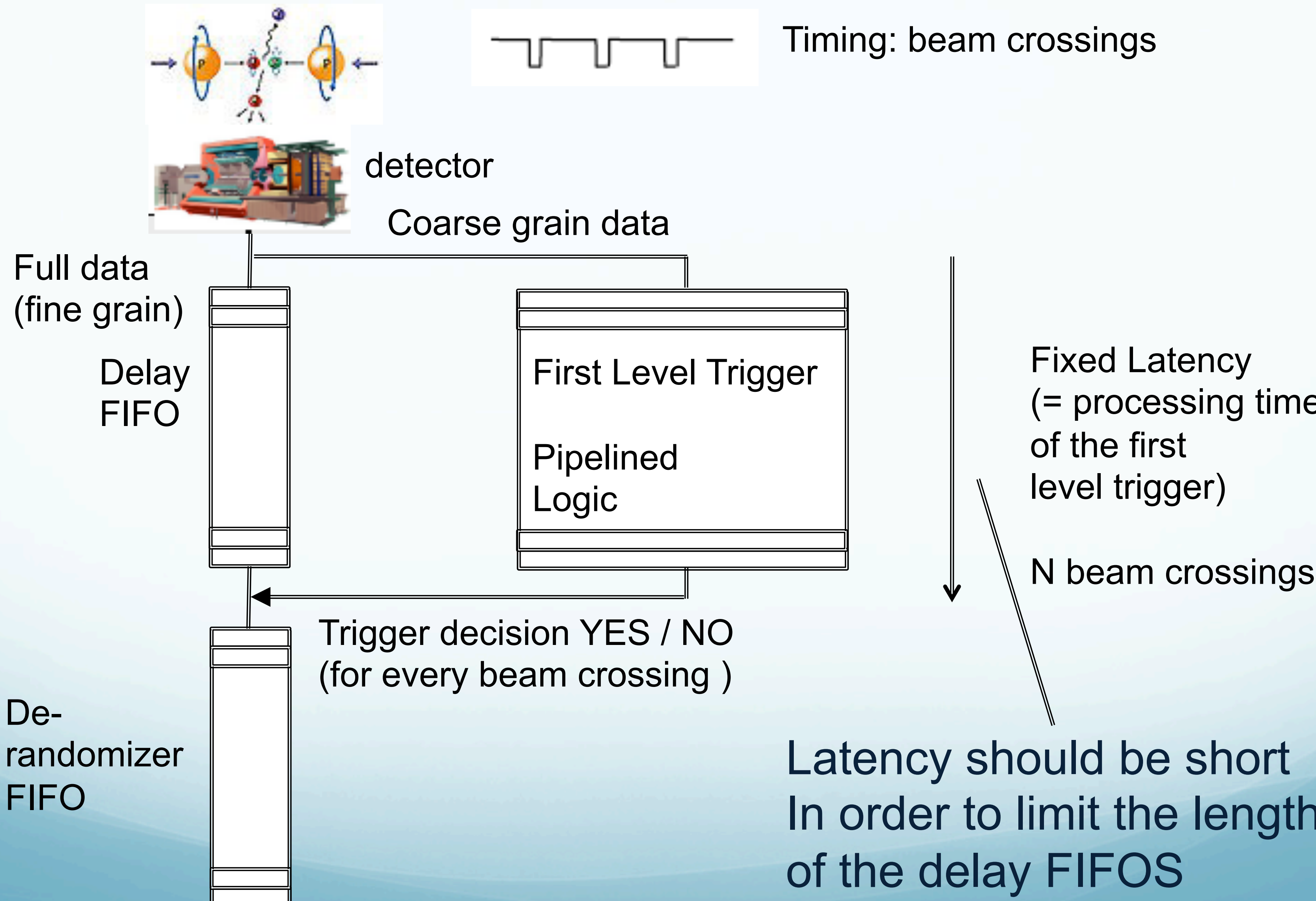

Quartus II - D:/altera_stratix_gx_kit/fpga/PCIe_optical_ver8/top - top - [../../../CMS_GIII_dev/Fed_cms_luminosity/ver_used/dma_M64_dt0.vwf]
File Edit View Project Assignments Processing Tools Window Help
top.bdf
Compilation Report - Flow Summary
decoder_pck.tdf
../mmu_ctrl/mmu.bdf
../../../CMS_GIII_dev/Fed_.../dma_M64_dt0.vwf
Master Time Bar: 1.5274 us
Pointer: 358.4 ns
Interval: -1.17 us
Start:
End:
Name
0 ps
160,0 ns
320,0 ns
480,0 ns
640,0 ns
800,0 ns
960,0 ns
1.12 us
1.28 us
1.44 us
1.6 us
1.76 us
1.92 us
2.08 us
-25.6 ns
4 us
+182.8 ns
RESET
CLOCK
REQ64
...64~result
ACK64
...64~result
REQ
GNT
IDSEL
...st|Master
...inst|Slave
AD
AD~result
CBE
CBE~result
FRAME
...E~result
IRDY
...Y~result
TRDY
...Y~result
DEVSEL
...EL~result
STOP
...P~result
...t|START
...|last_dt[0]
...|last_dt[1]
LOCK
PAR
PAR~result
PAR64
...64~result
user100
user107
URESET
LDOWN
UCLK
UCTRL
UWEN
UD
LFF
URESET
BA0
inst58
BA1
inst59
we_boot
wp_boot
Idle_m
M_data
Idle
00000000
A0000100
0000003E
For Help, press F1
36%
00:08:49
NUM

Embedded Logic Analyzers



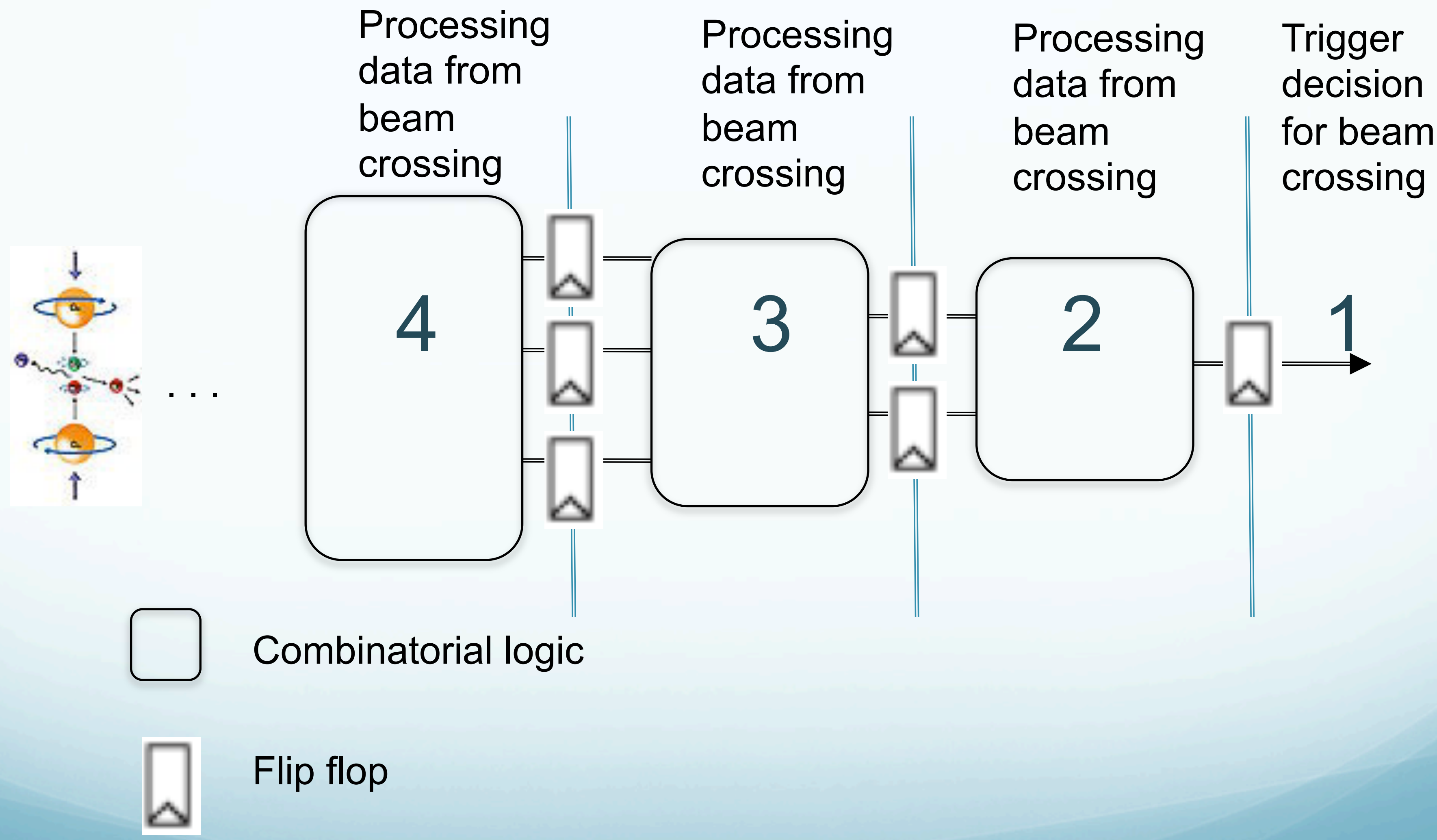
FPGA applications in the Trigger & DAQ domain

First-Level Trigger at Collider

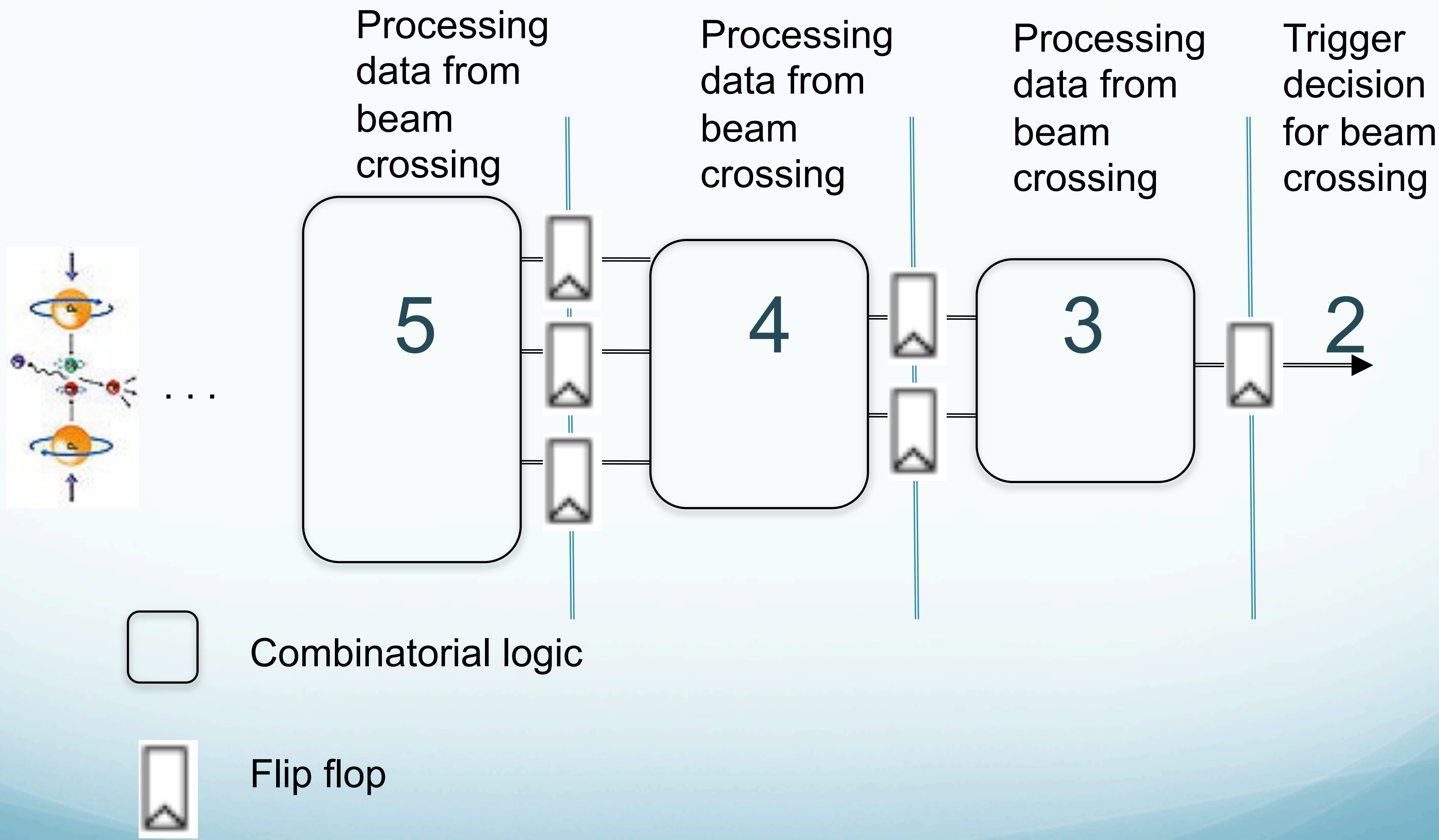


Latency should be short
In order to limit the length
of the delay FIFOS

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 parallel inputs
  - Data from many parts of the detector has to be combined
  - Can send parallel data rather than serial

Low latency

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

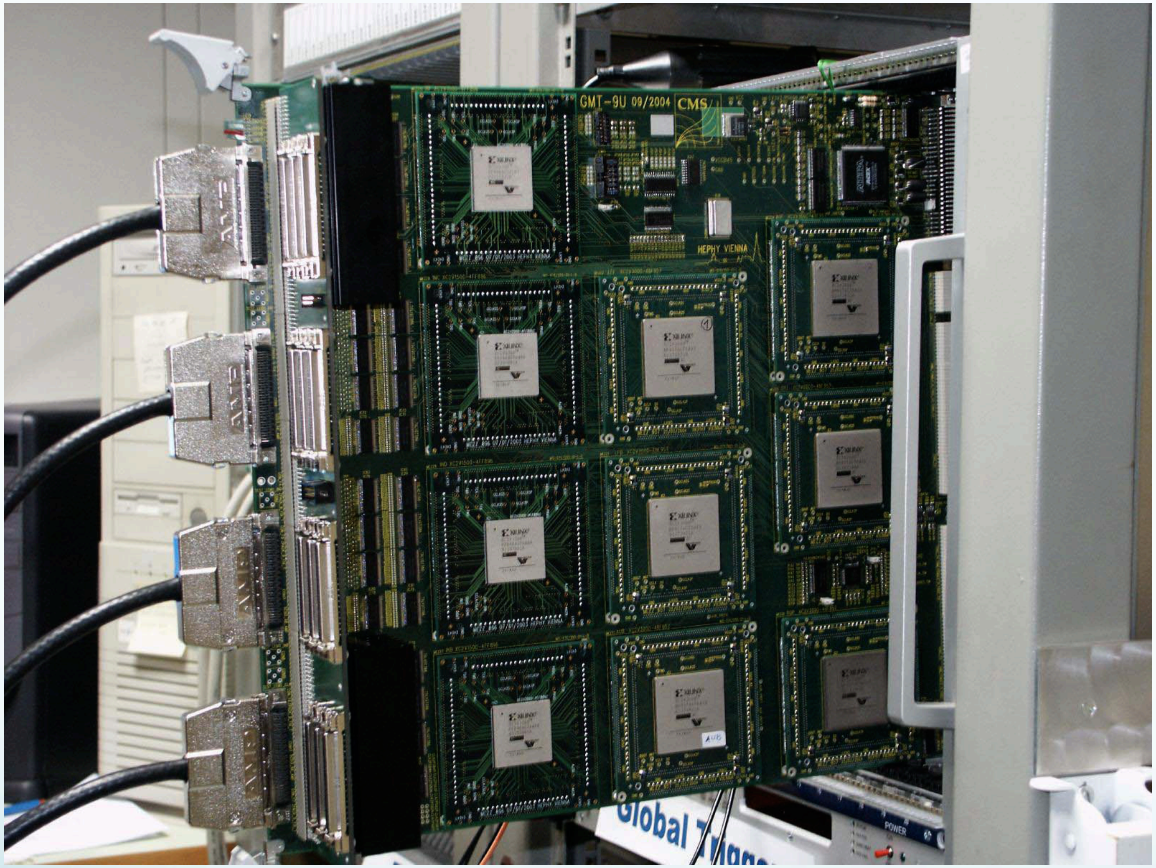
High performance

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

Trigger algorithms implemented in FPGAs

- Peak finding
- Pattern Recognition
- Track Finding
- Energy summing
- Sorting
- Topological Algorithms
- Trigger Control system
- Fast signal merging
- Many more …

Example: CMS Global Muon Trigger

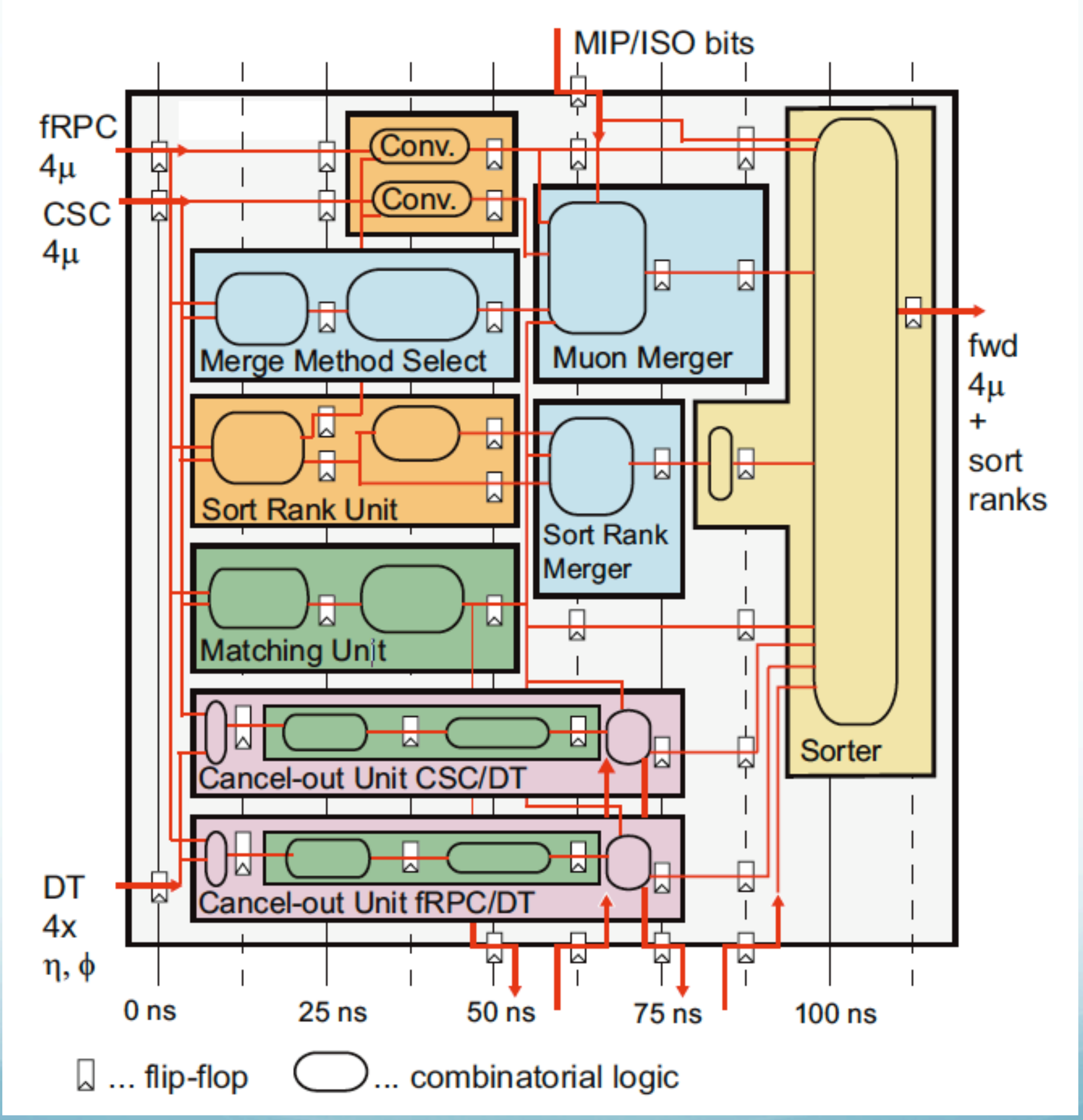


- Input: ~1000 bits
  @ 40 and 80 MHz
- Output: ~50 bits @ 80MHz
- Processing time: 250 ns
- Pipelined logic
  one new result every 25 ns

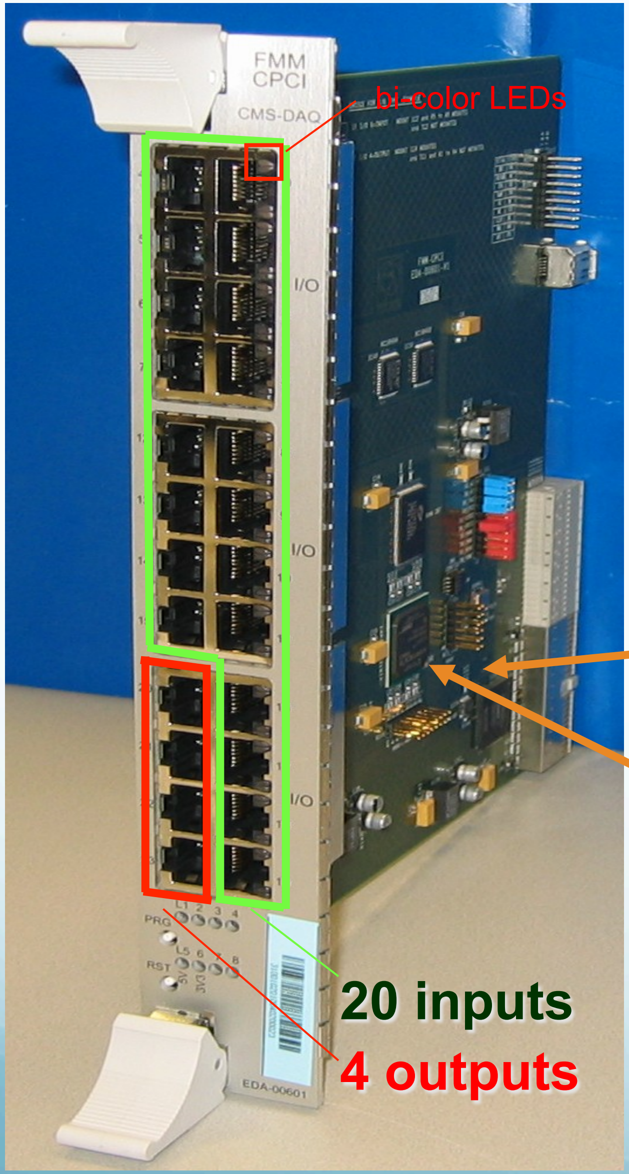
- 10 Xilinx Virtex-II FPGAs
- up to 500 user I/Os per chip
- Up to 25000 LUTs per chip used
- Up to 96 x 18kbit RAM used

- The CMS Global Muon trigger receives 16 muon candidates from the three muon systems of CMS
- It merges different measurements for the same muon and finds the best 4 over-all muon candidates

CMS Global Muon Trigger main FPGA



Example 2: CMS Fast Merging Modules



These modules merge the status of all detector front-ends in CMS in order to throttle the trigger when buffers fill up.

Additionally these modules monitor all status changes of detector frontends.

1 FPGA (Altera) : PCI interface

1 FPGA (Xilinx) :

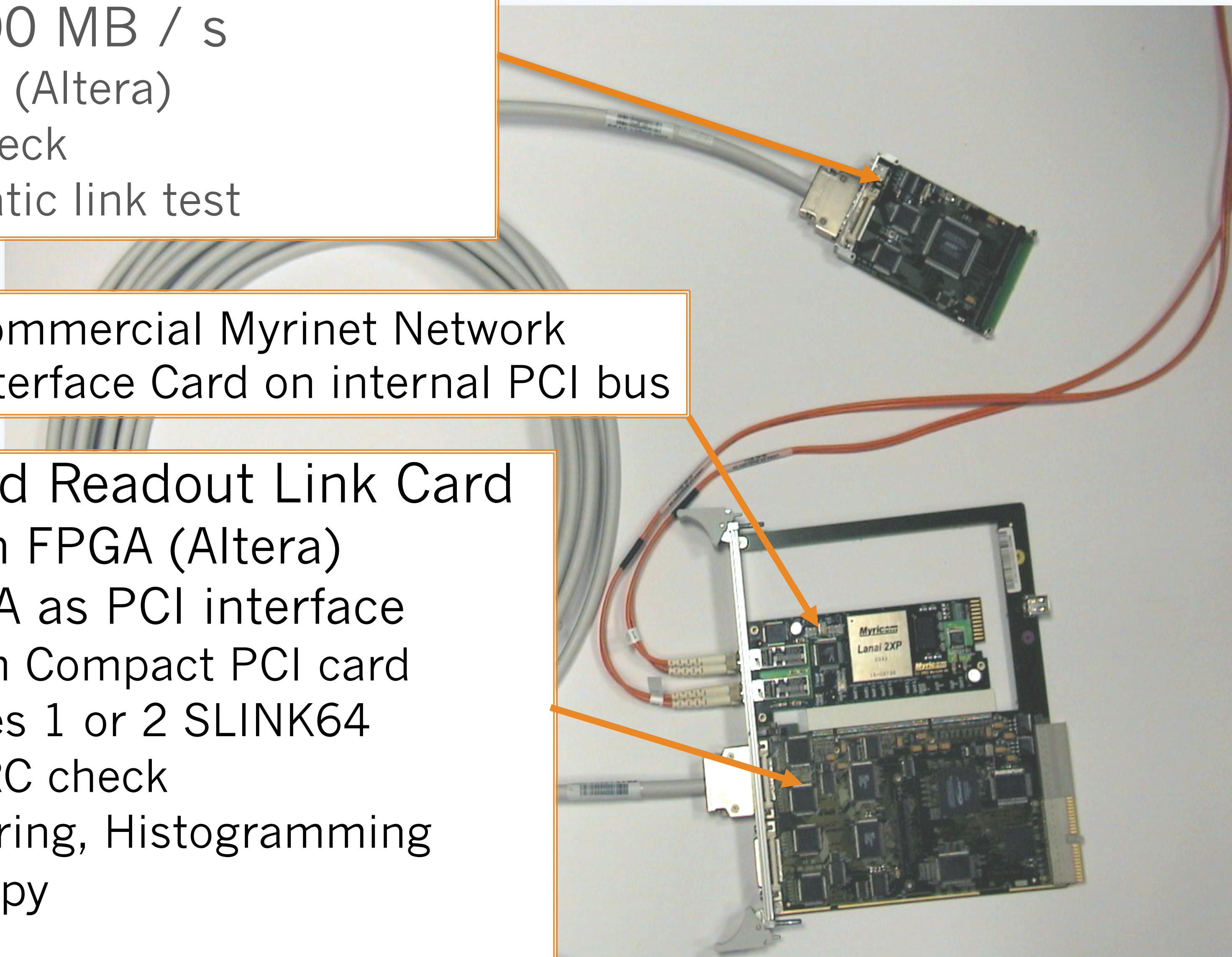
- Merging logic (1 μs latency)
- Monitoring logic
- Interface to SRAM

FPGAs in Data Acquisition

- Frontend Electronics
  - Pedestal subtraction
  - Zero suppression
  - Compression
  - …
- Custom data links
  - E.g. SLINK-64
    - Several serial LVDS links in parallel
    - Up to 400 MB/s
- Interface from custom hardware to commercial electronics
  - PCI bus, VME bus, Myrinet, etc.

Example 3: CMS Front-end Readout Link

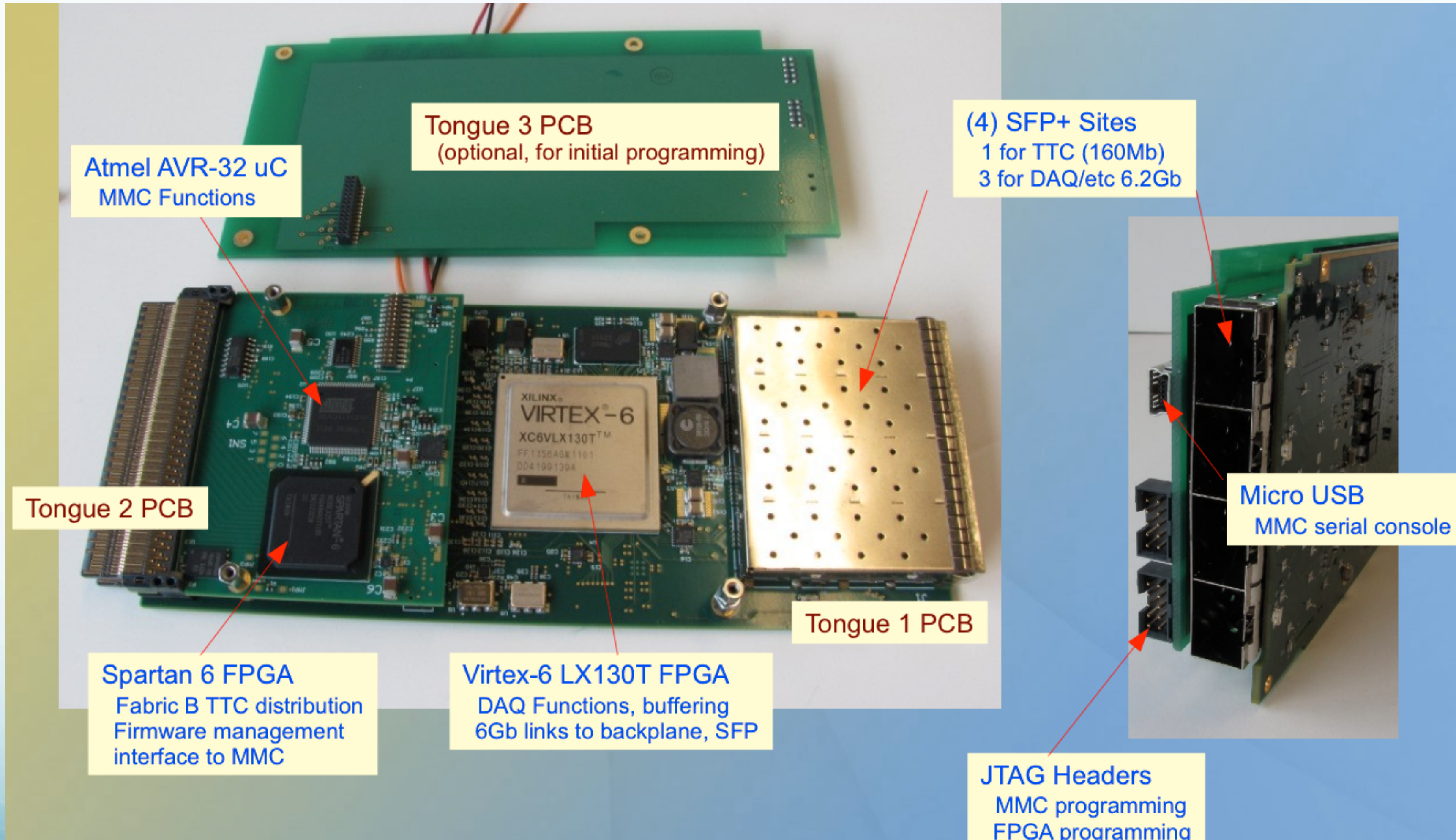
- SLINK Sender Mezzanine Card: 400 MB / s
  - 1 FPGA (Altera)
  - CRC check
  - Automatic link test



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: Prototype of future CMS DAQ card for μTCA crate



High speed serial links replacing the parallel links …

DAQ card in μTCA crate

Very compact
Backplane supports high-speed serial links
Possibility to develop generic FPGA cards

FPGAs in other domains

- Set-top boxes
- Medical imaging
- Computer vision
- Speech recognition
- Cryptography
- Bioinformatics
- Software-Defined Radio
- Aerospace
- Defense
- Digital Signal Processing
- ASIC Prototyping
- High performance computing
  - Computations performed by FPGA: FFT, Convolution
- Reconfigurable computing

Acknowledgement

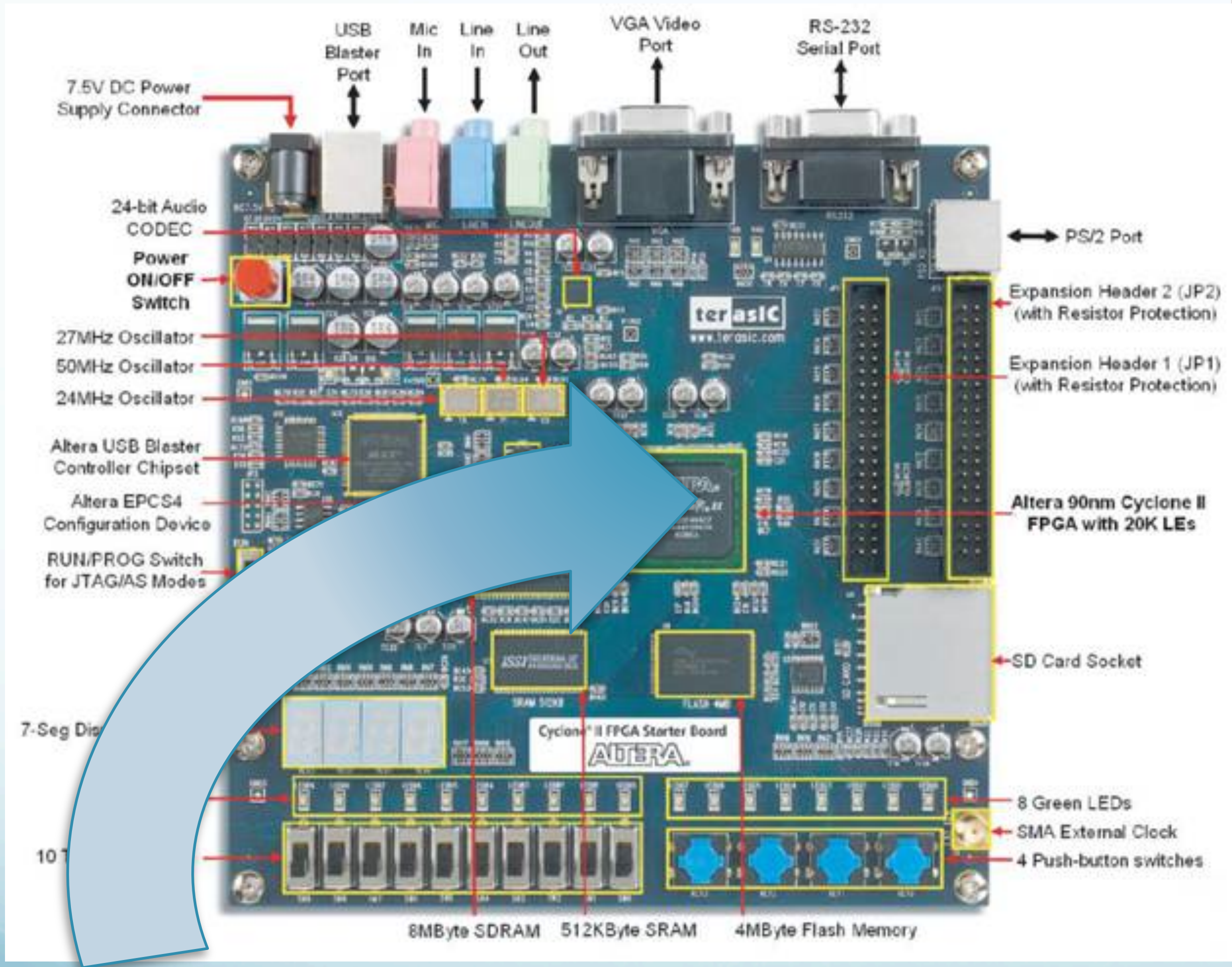
- Parts of this lecture are based on material by Clive “Max” Maxwell, author of several books on FPGAs. Many thanks for his kind permission to use his material!

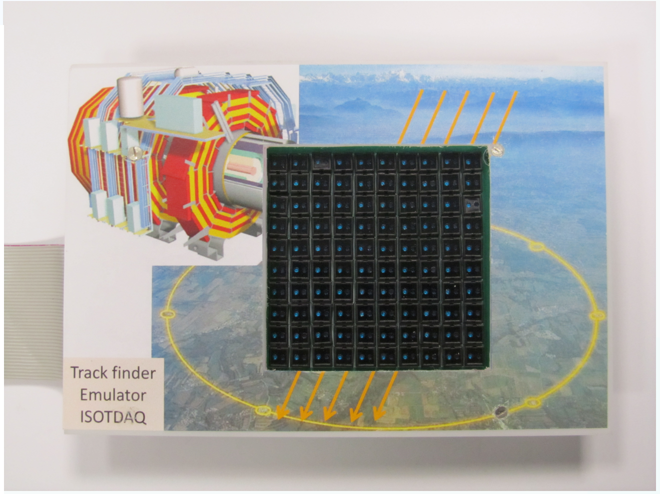
  Web lecture: courses.techonline.com
  Max@techbites.com , www.TechBites.com

- Many thanks also to Dominique Gigi, who provided material for the lecture and who will be the main tutor for the FPGA exercise

FPGA lab exercise

Lab 6: Programming an FPGA





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