Field Programmable Gate Arrays

Dr. Hannes Sakulin
CERN / PH-CMD

First ISO TDAQ school
Ankara, 1st Feb, 2010

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

- Introduction to digital electronics
- Short history of programmable logic devices
- Programming techniques
- The internals of an FPGA
- Where are FPGAs used / Examples Applications
- How are FPGAs designed?

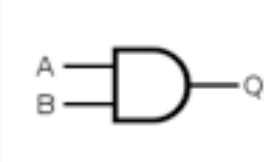
Introduction to digital electronics

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

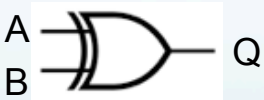
```
q = a && b;
```

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

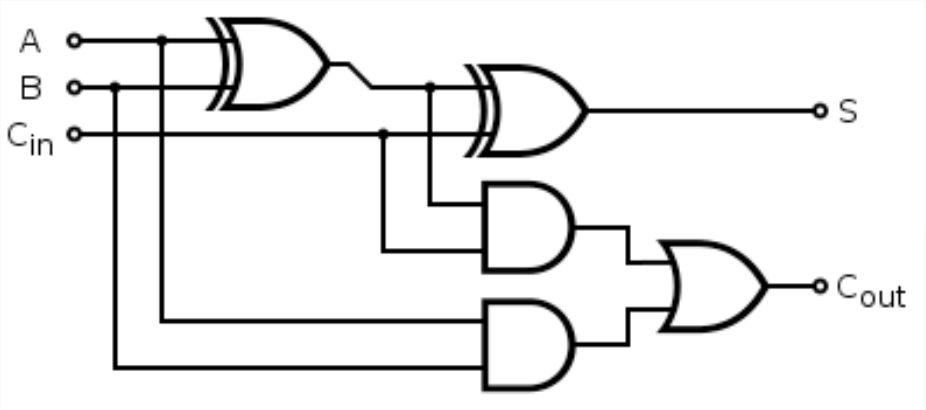


INPUT		OUTPUT
A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

```
q = a != b;
```

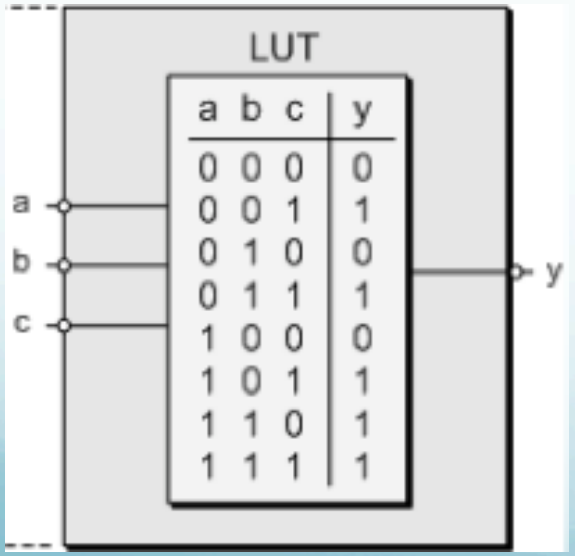
⋮

Combinatorial logic



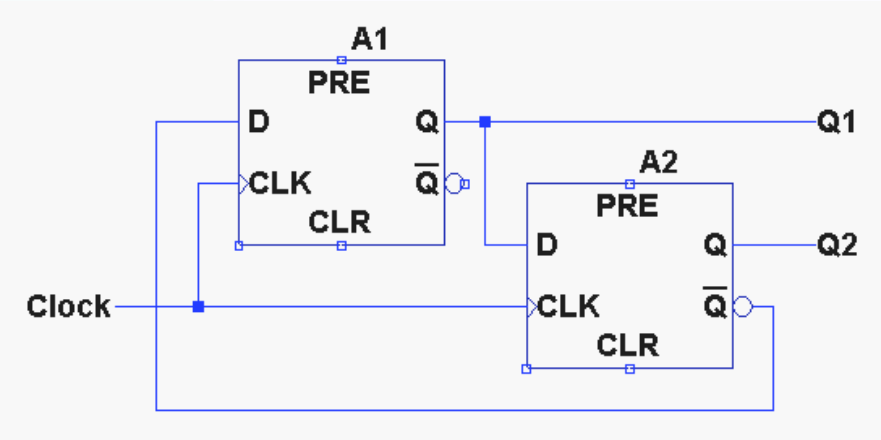
Example: Full adder with carry-in, carry-out

Outputs are determined by Inputs, only



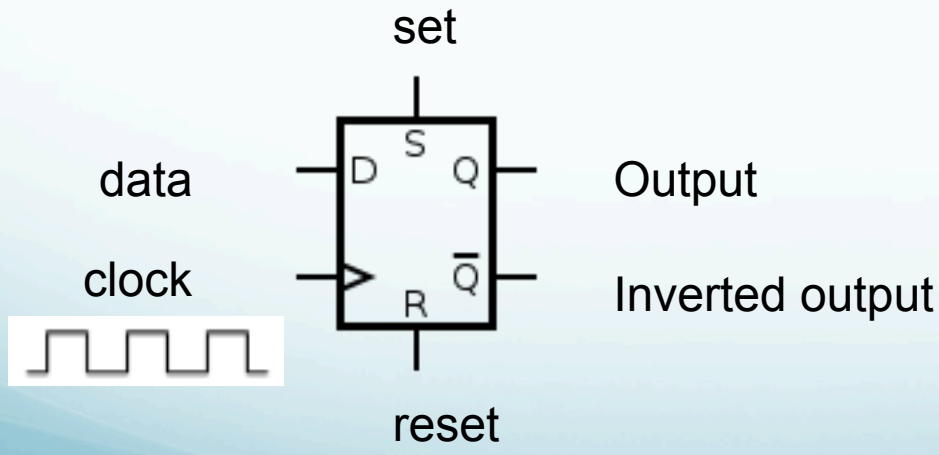
Combinatorial logic may be implemented using Look-Up Tables (LUTs)

(Synchronous) sequential logic



2-bit binary counter

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

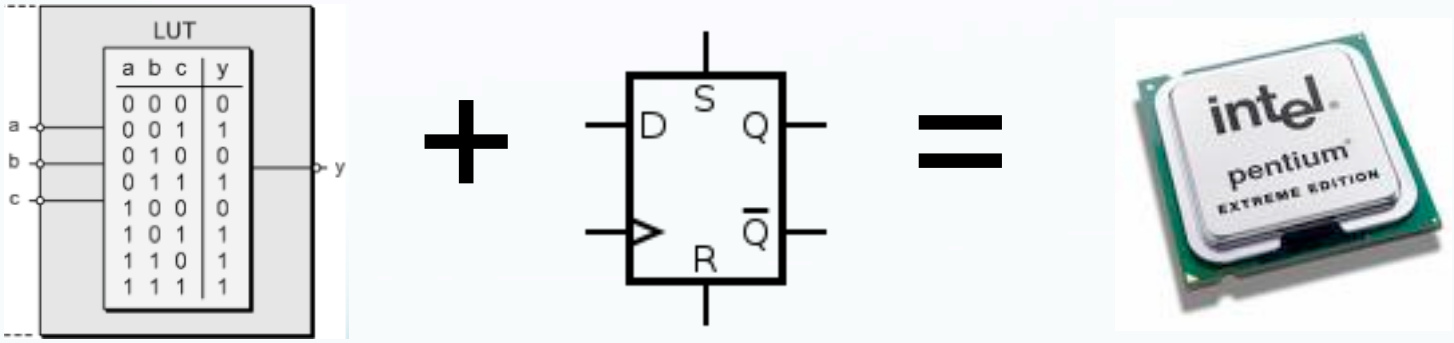


D Flip-flop:
samples the data at the rising (or falling) edge of the clock

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

D Flip-flop

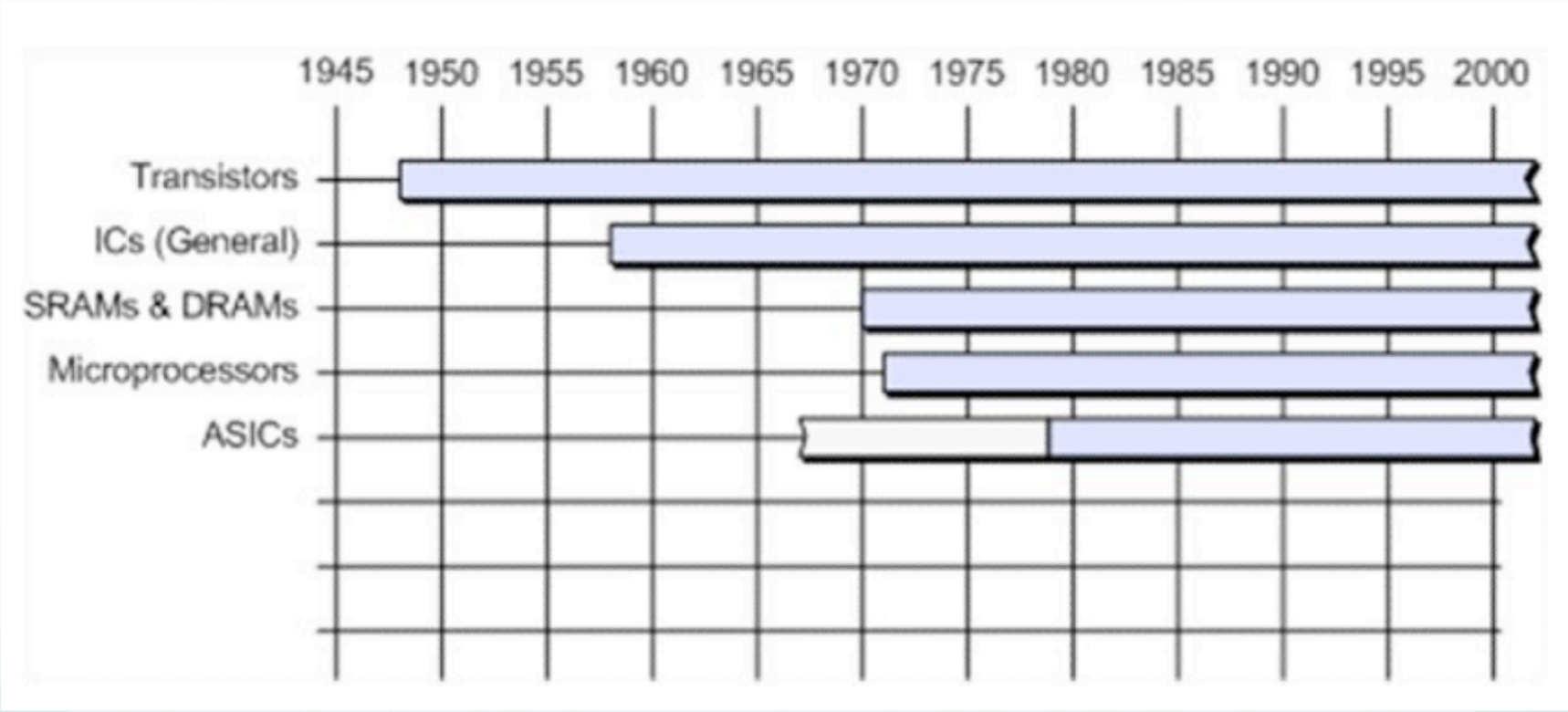
Synchronous sequential logic



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

Programmable digital electronics

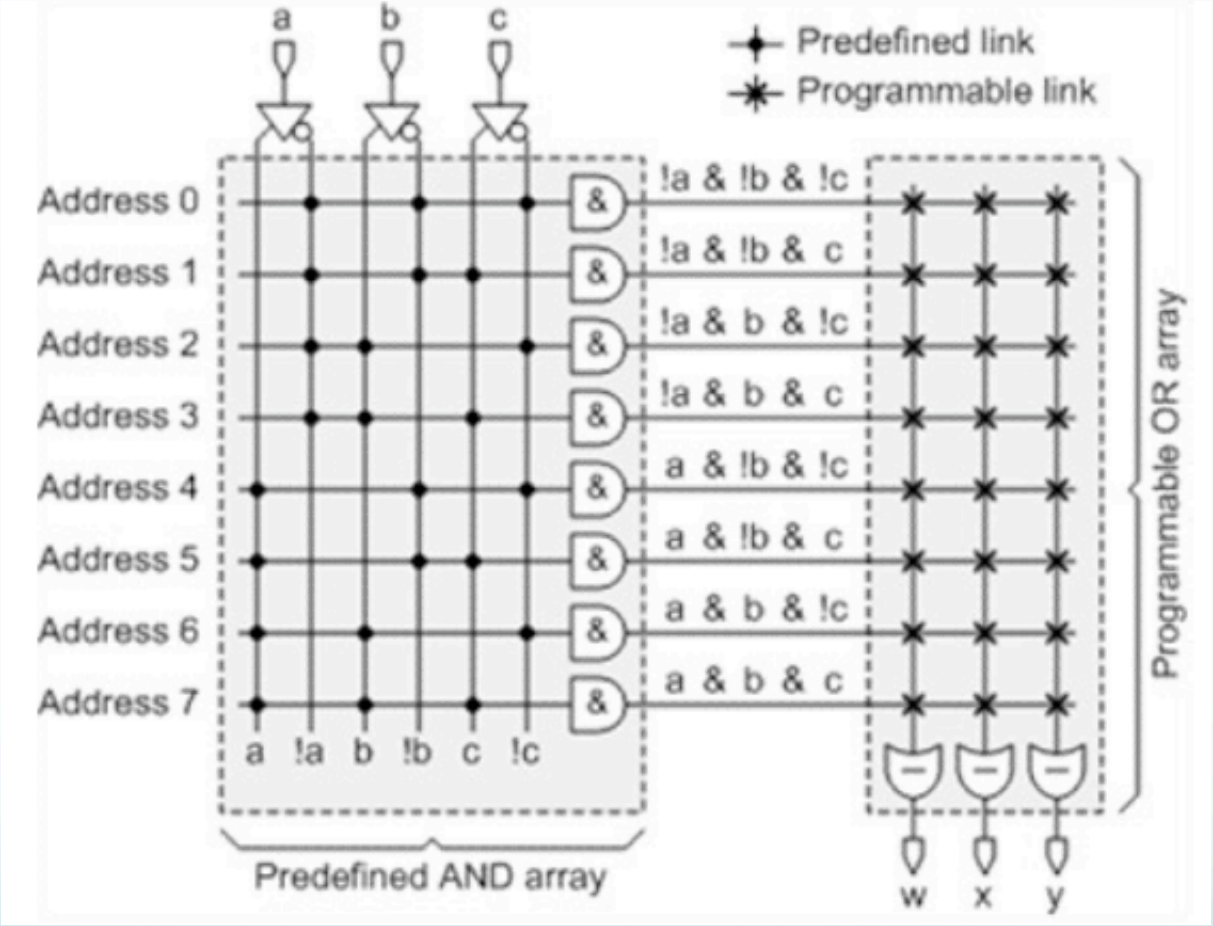
Long long time ago ...



Simple PLDs (sPLDs): PROMs

Programmable Logic Devices

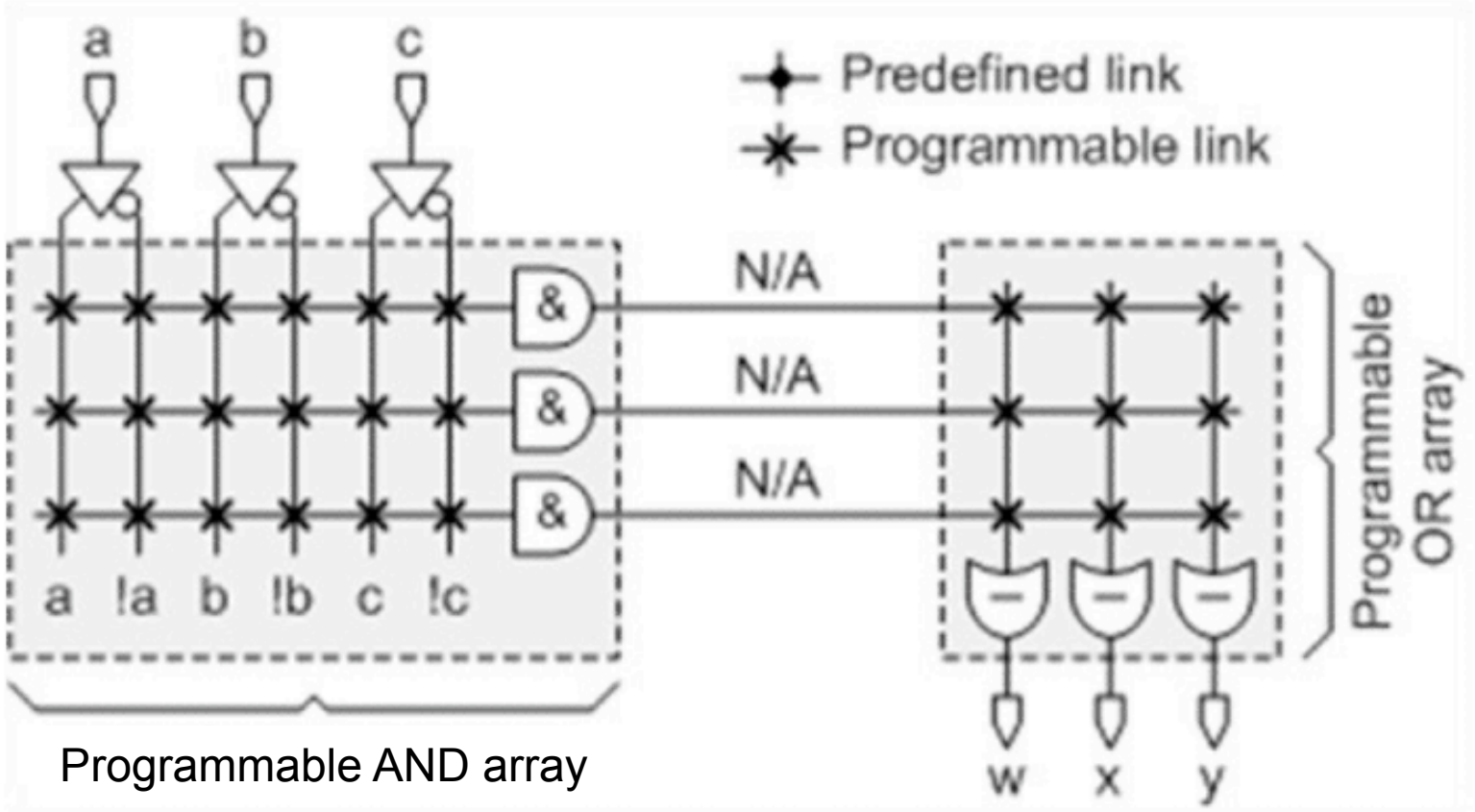
Programmable Read-Only Memory



Unprogrammed PROM (Fixed AND Array, Programmable OR Array)

Simple PLDs (sPLDs): PLAs

Programmable Logic Arrays



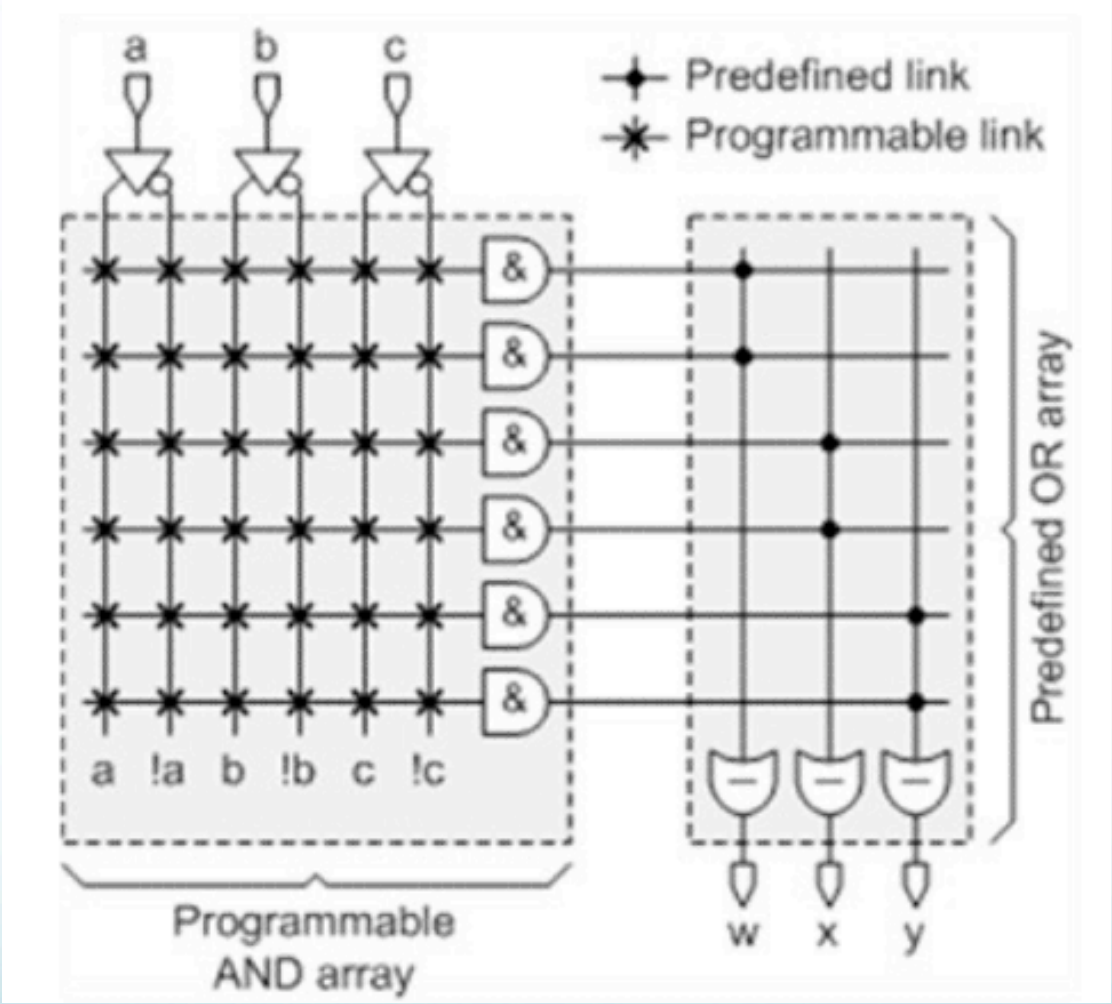
Unprogrammed PLA (Programmable AND and OR Arrays)

Most flexible
but slower

1975

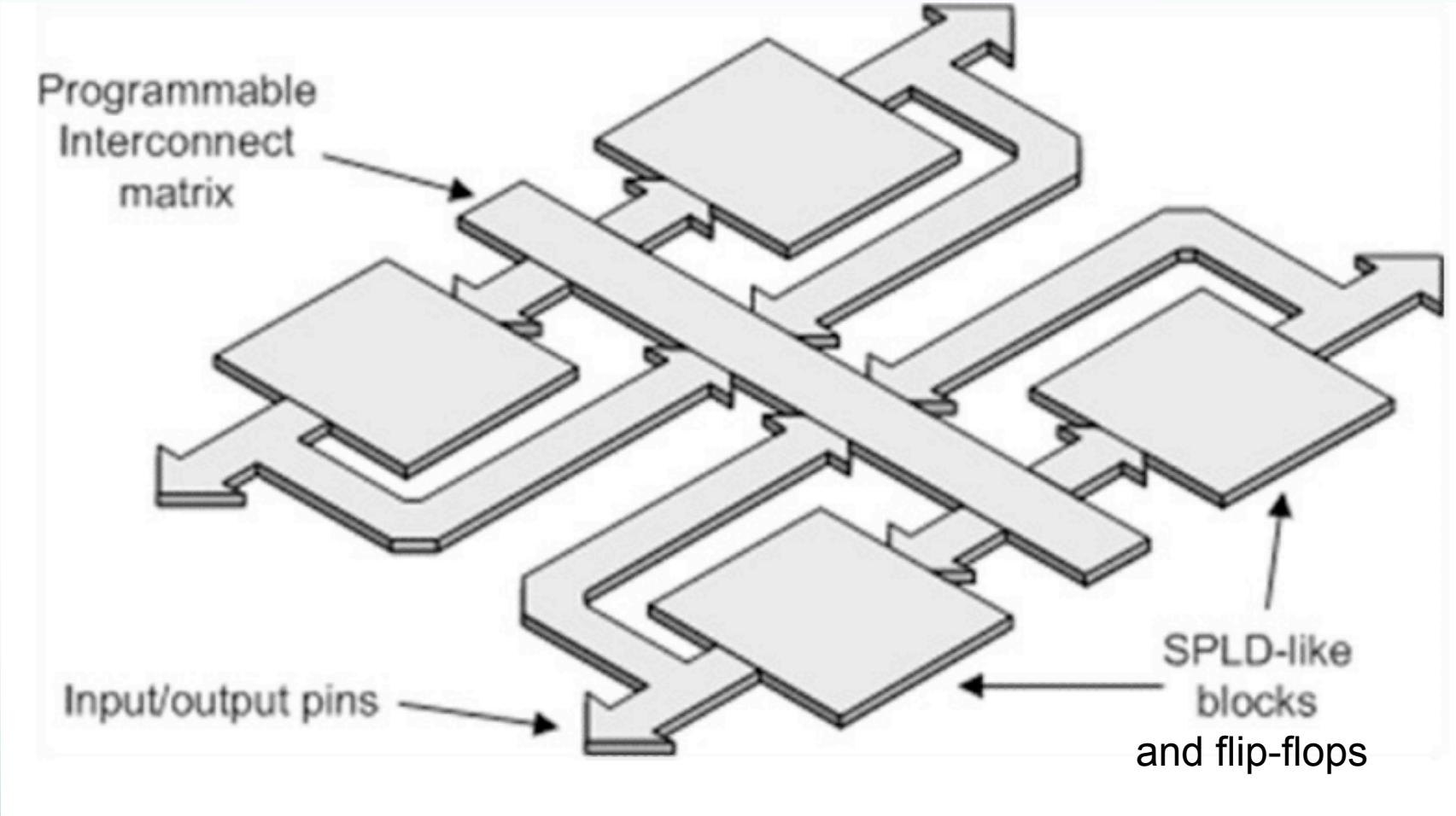
Simple PLDs (sPLDs): PALs

Programmable Array Logic



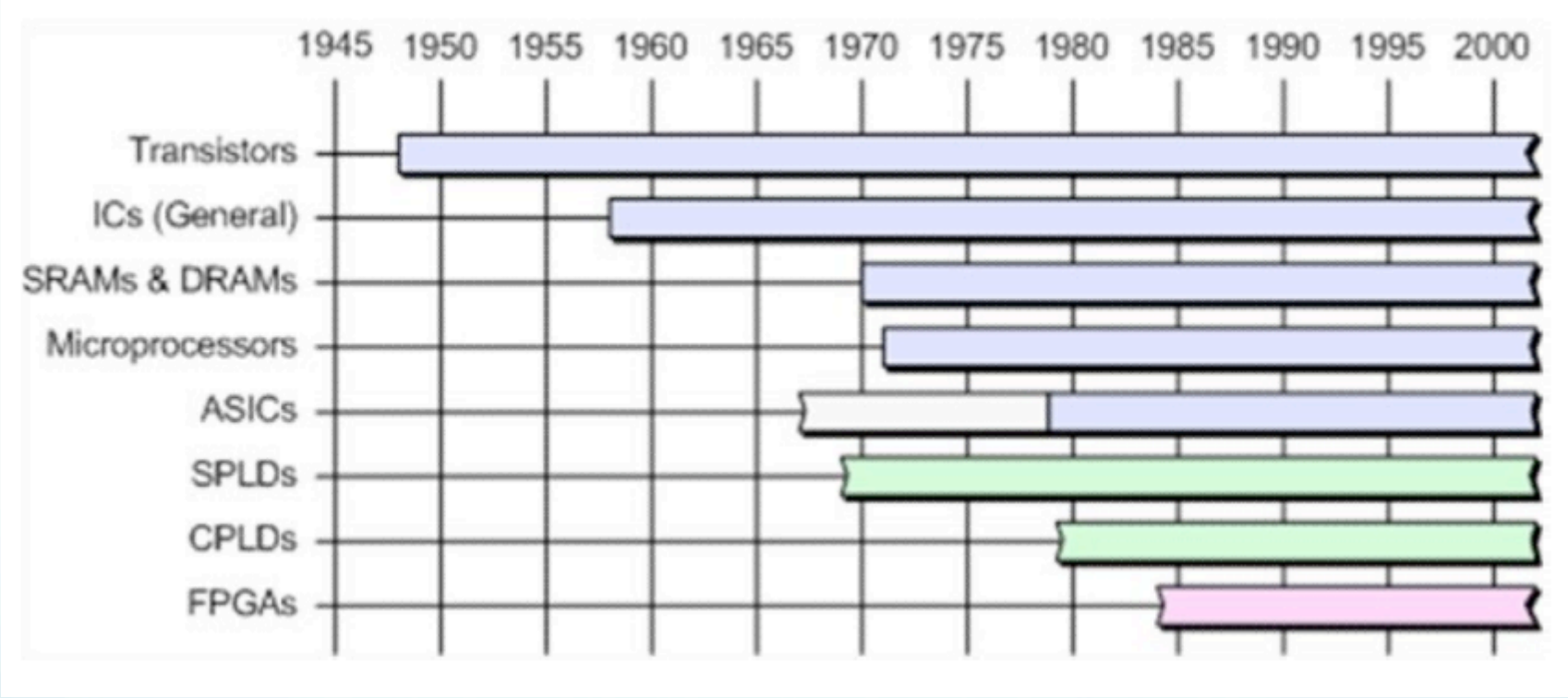
Unprogrammed PAL (Programmable AND Array, Fixed OR Array)

Complex PLDs (CPLDs)

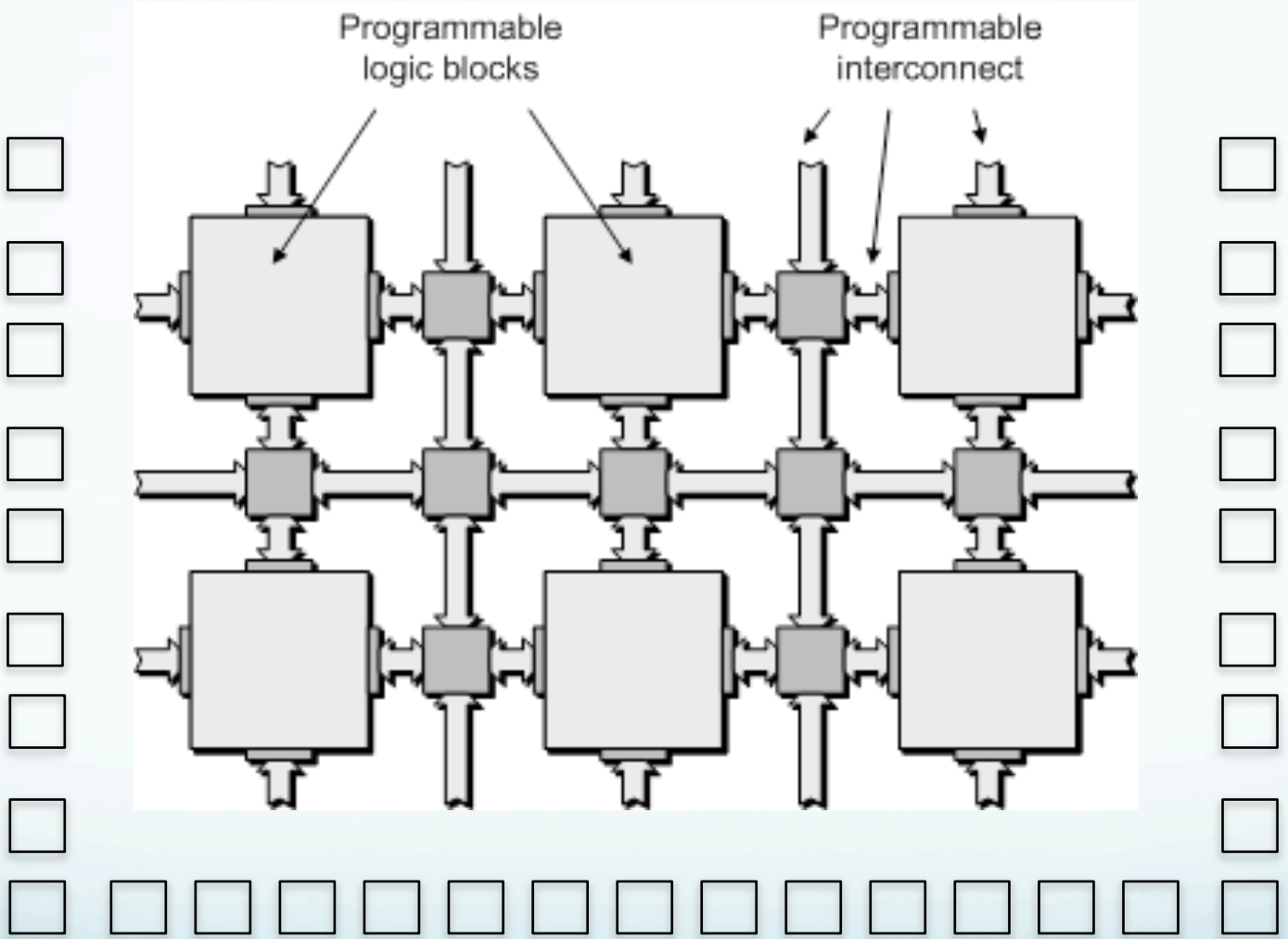


Coarse grained
100's of blocks
(EE)PROM based

FPGAs ...


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

FPGAs

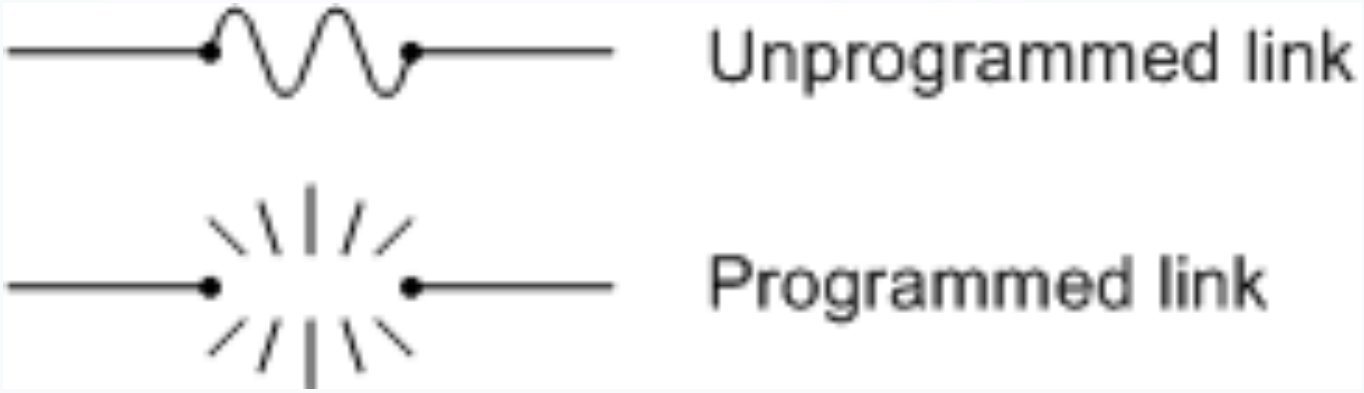


Fine-grained
today: 100.000's logic blocks

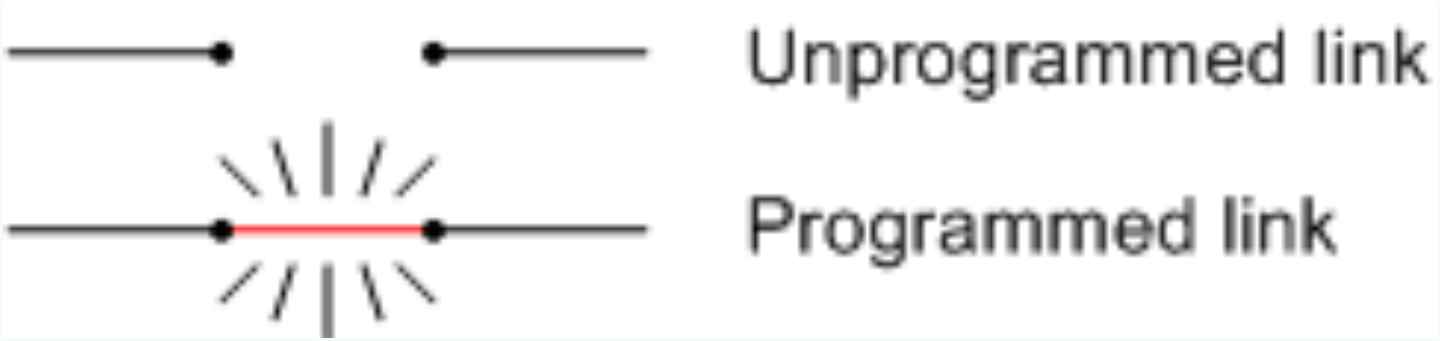
Programmable Input / Output pins

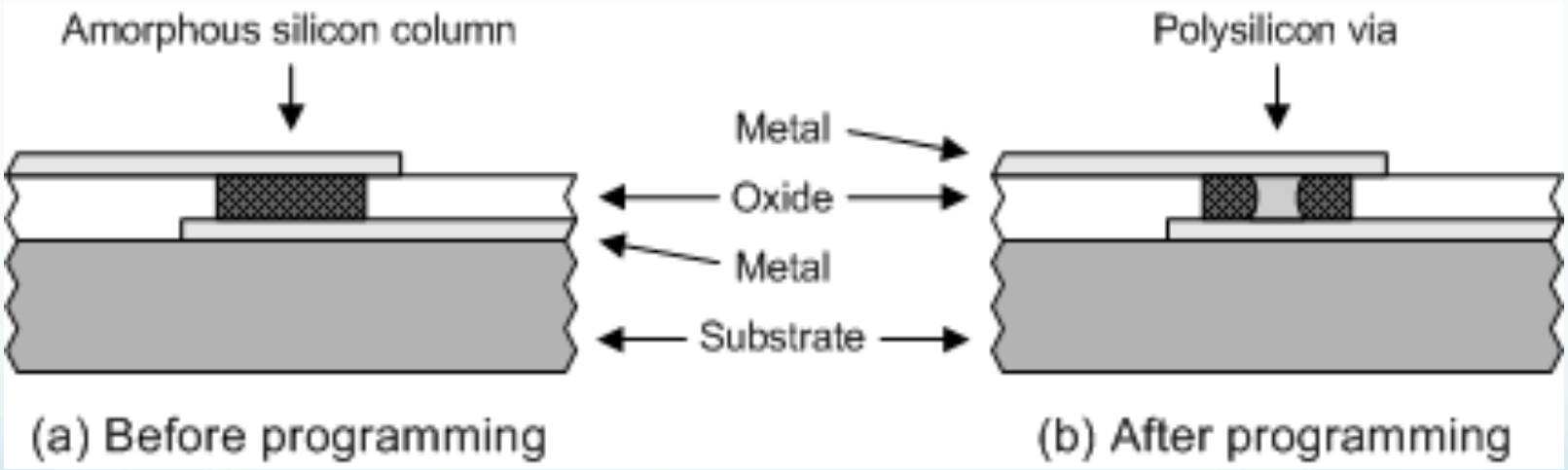
Programming techniques

Fusible Links (not used in FPGAs)

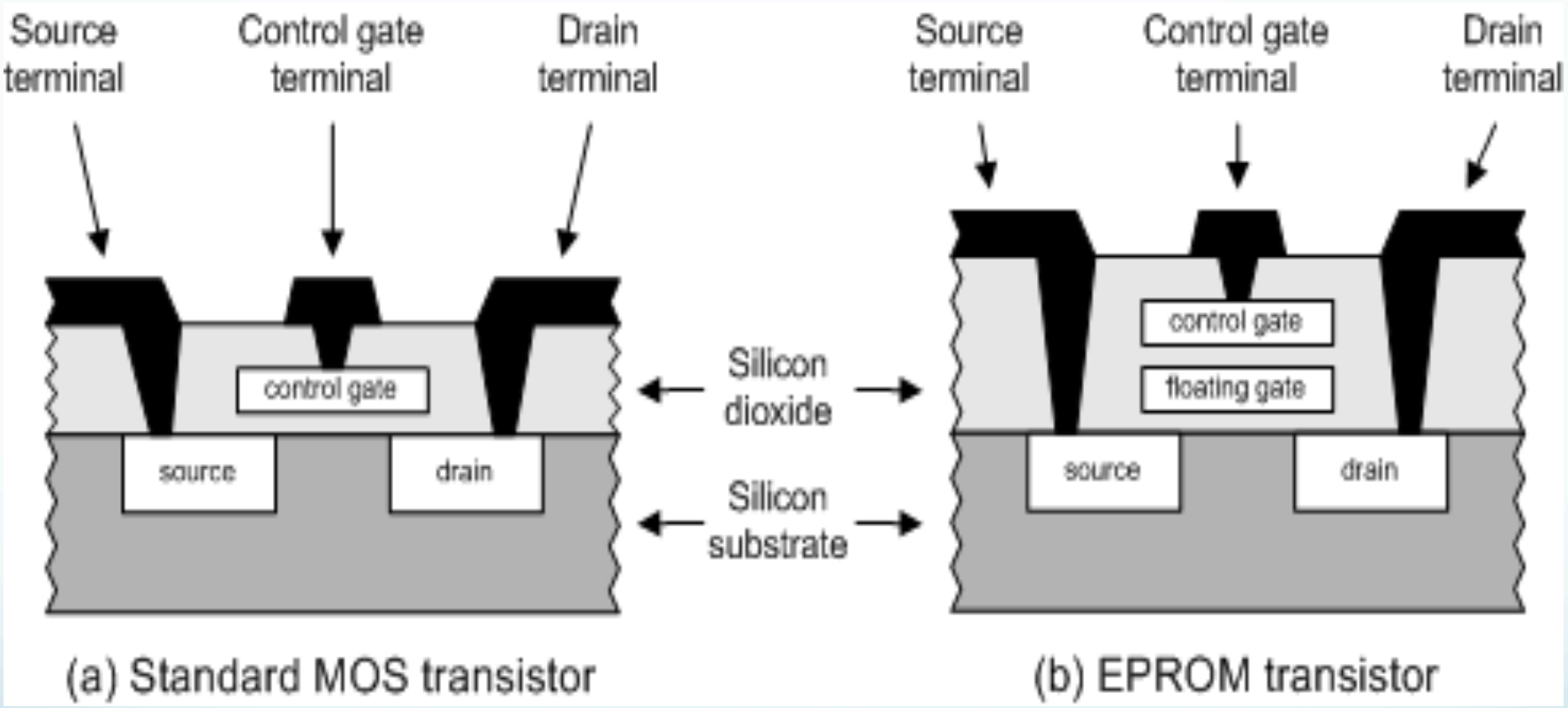


Antifuse Technology



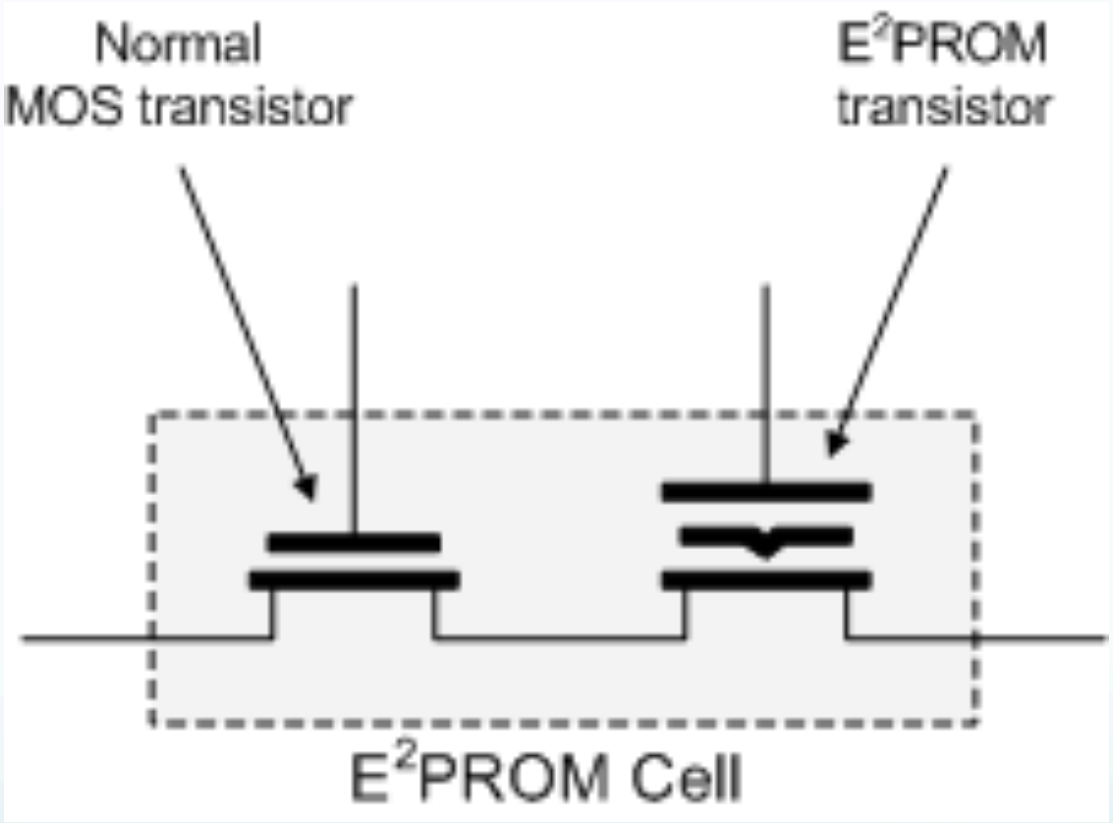


EPROM Technology

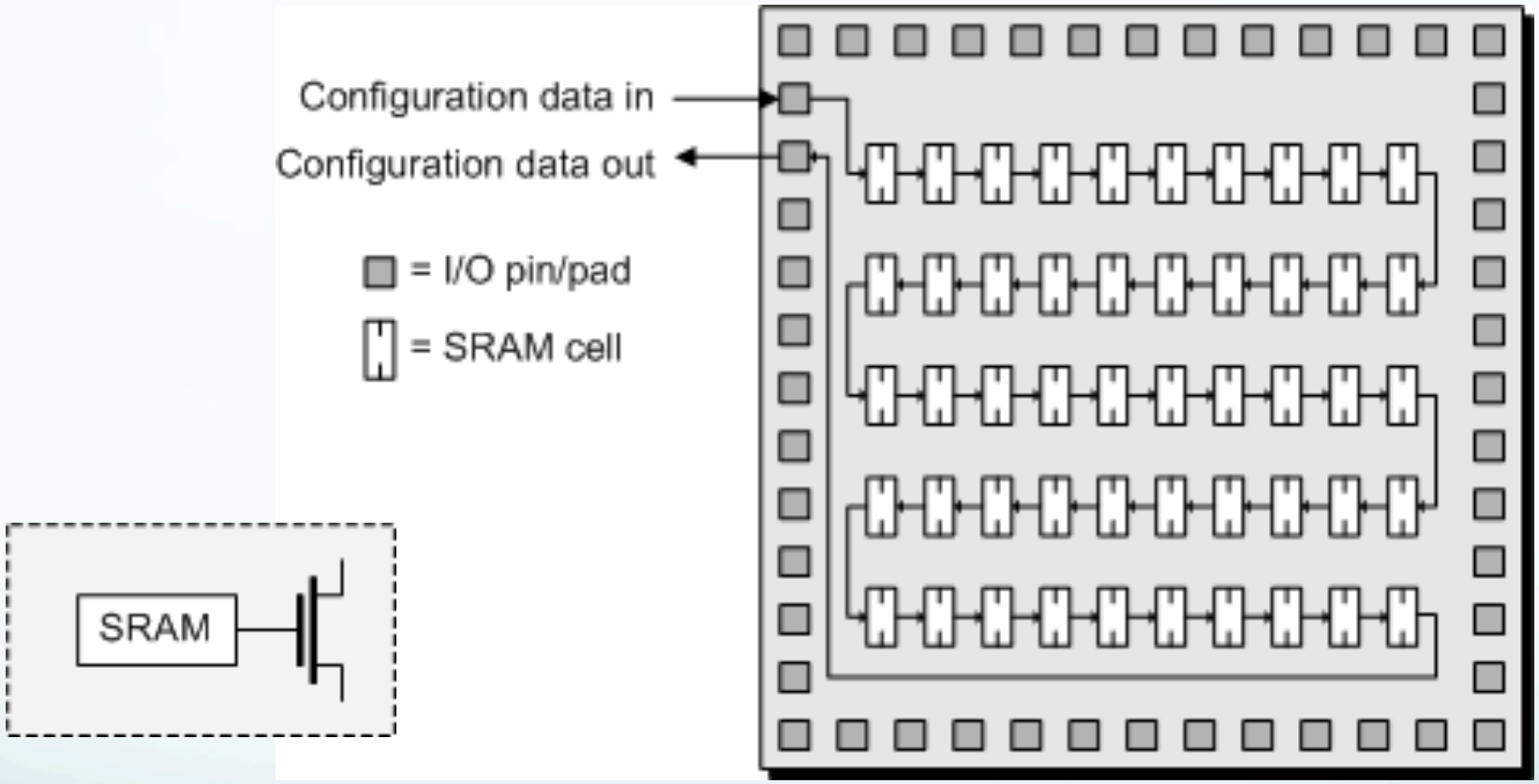


Intel, 1971

EEPROM and FLASH Technology

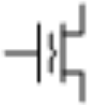


SRAM-Based Devices



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)

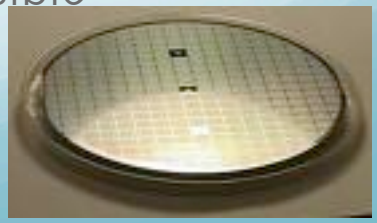
FPGA – ASIC comparison

FPGA

- Lower cost per device when small quantities are needed
  - Prices getting lower all the time
- Low development cost
  - You can get started with a $200 development board and free software
- Rapid development cycle (minutes / hours)
- May be reprogrammed in the field (firmware upgrade)
  - New features
  - Bug fixes

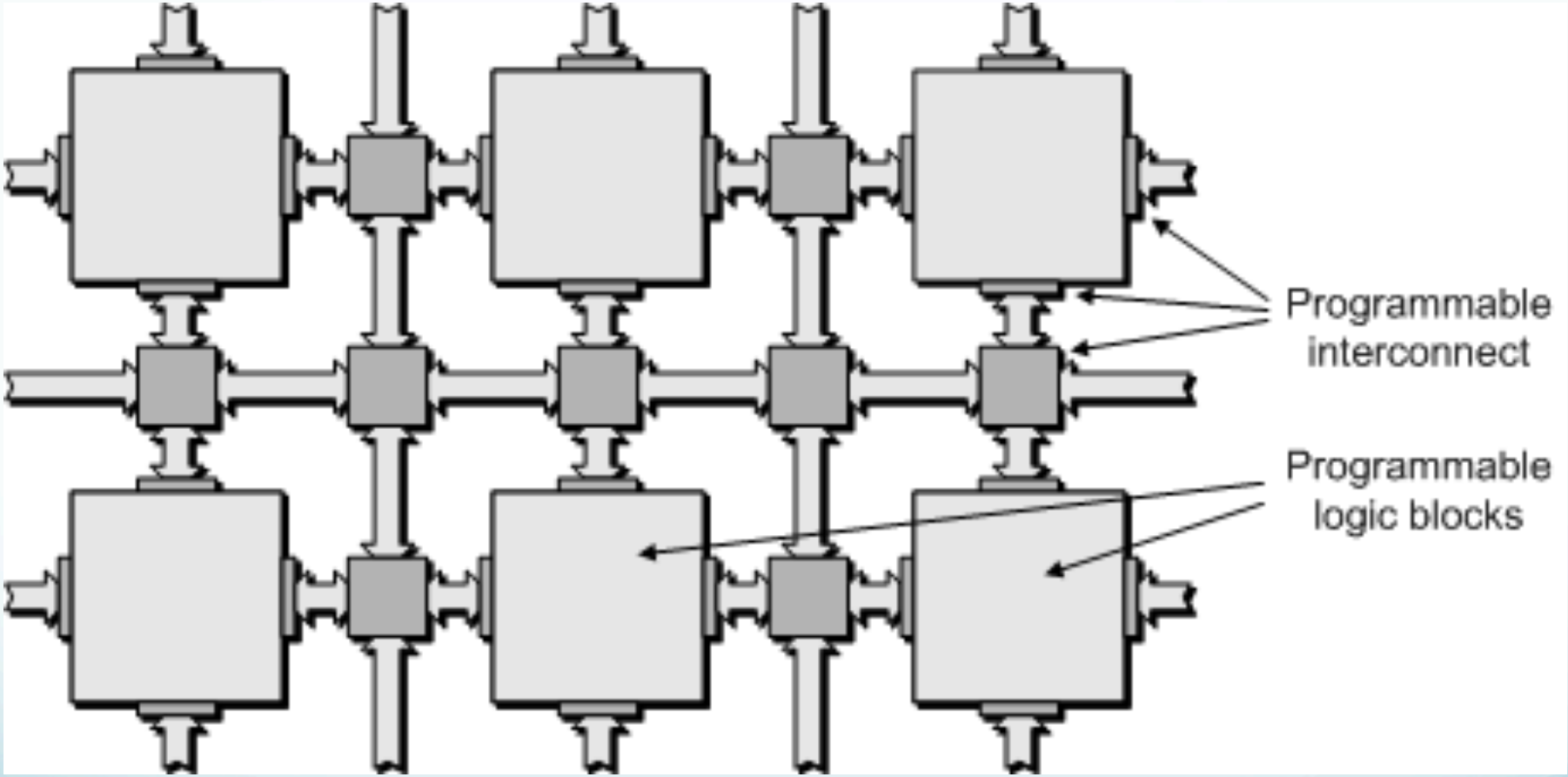
ASIC

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



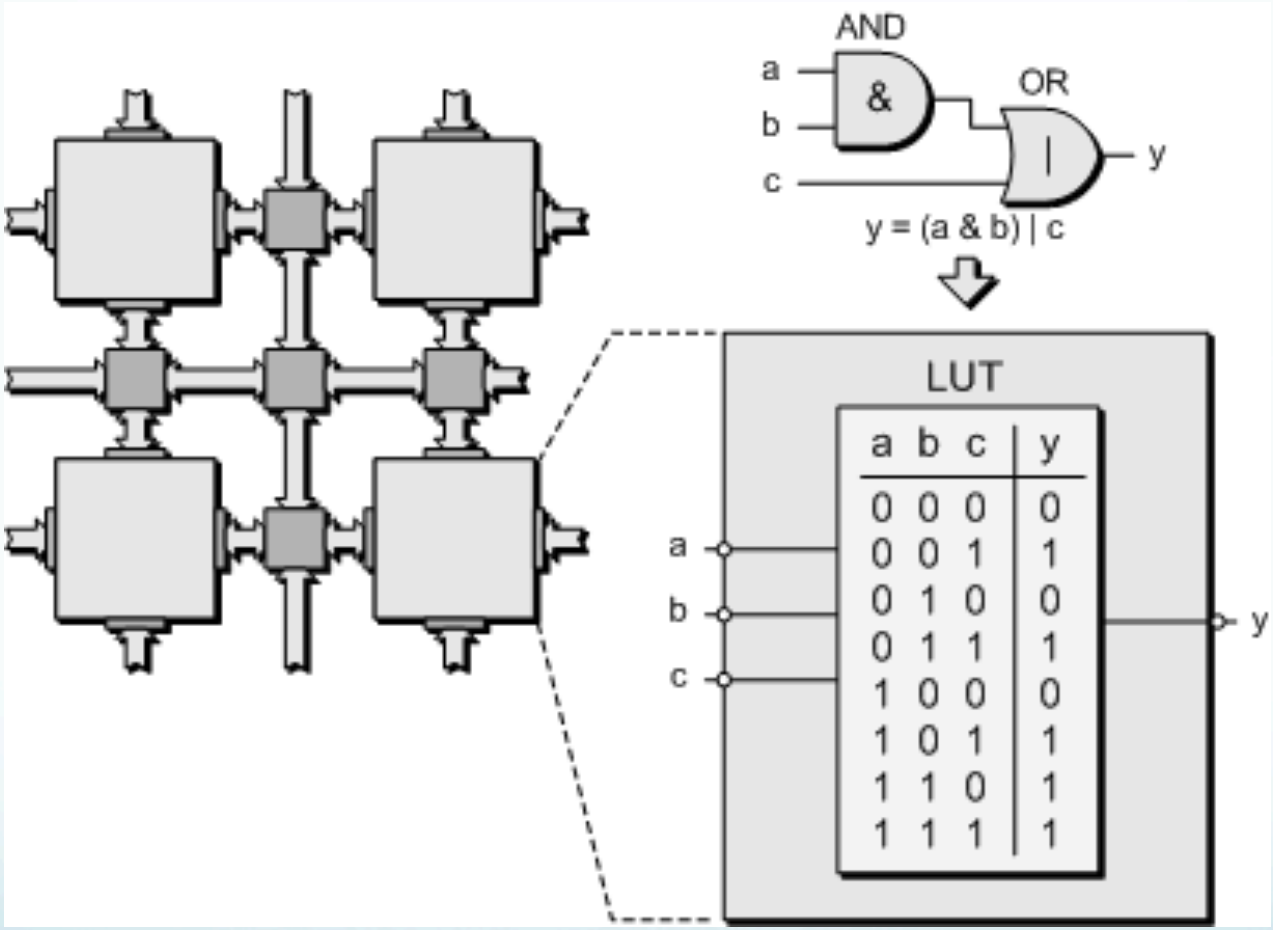
FPGA internals

Architectures and Fabrics

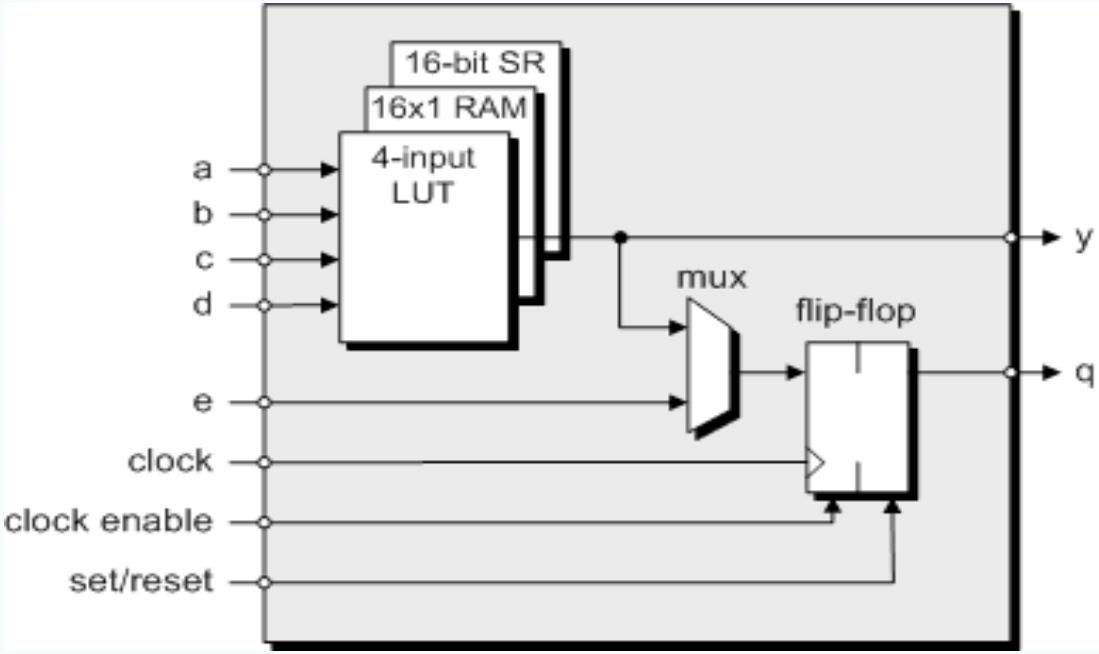


A generic FPGA architecture

LUT-based Fabrics



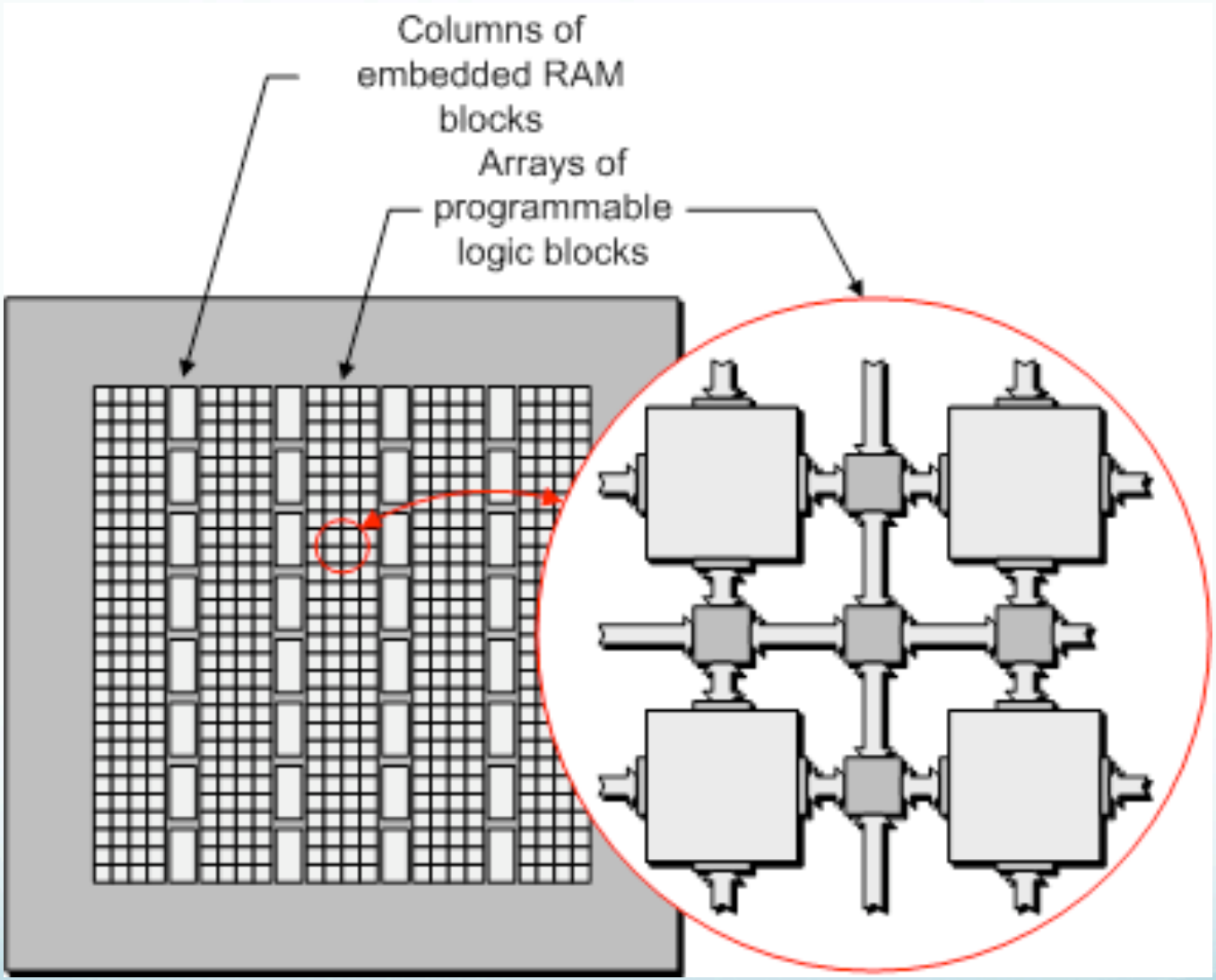
Typical LUT-based Logic Cell



Xilinx: logic cell,
Altera: logic element

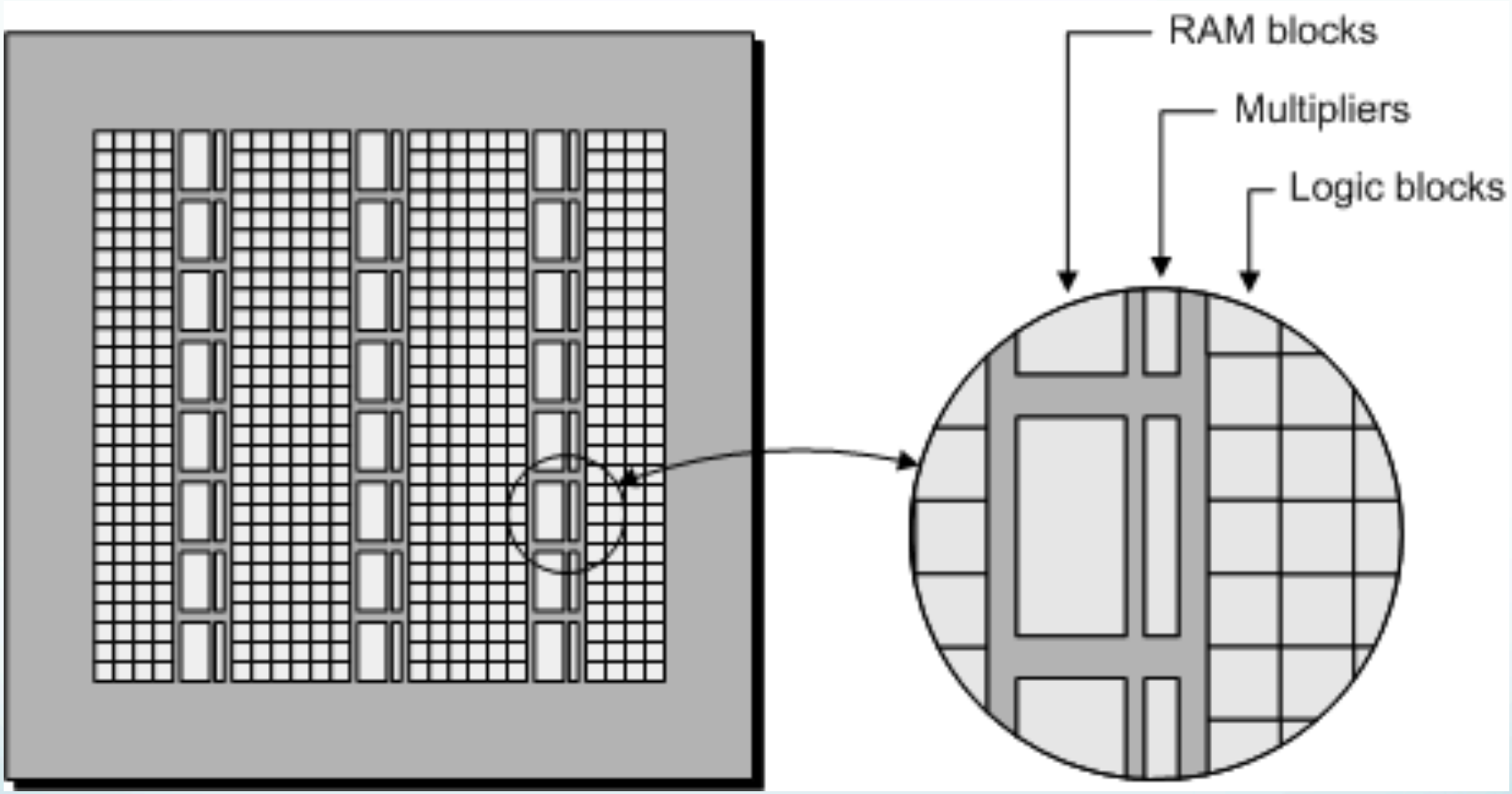
- LUT may implement any function of the inputs
- Flip-Flop registers the LUT output
- May use only the LUT or only the Flip-flop
- LUT may alternatively be configured a shift register
- Additional elements (not shown): fast carry logic

Embedded RAM blocks

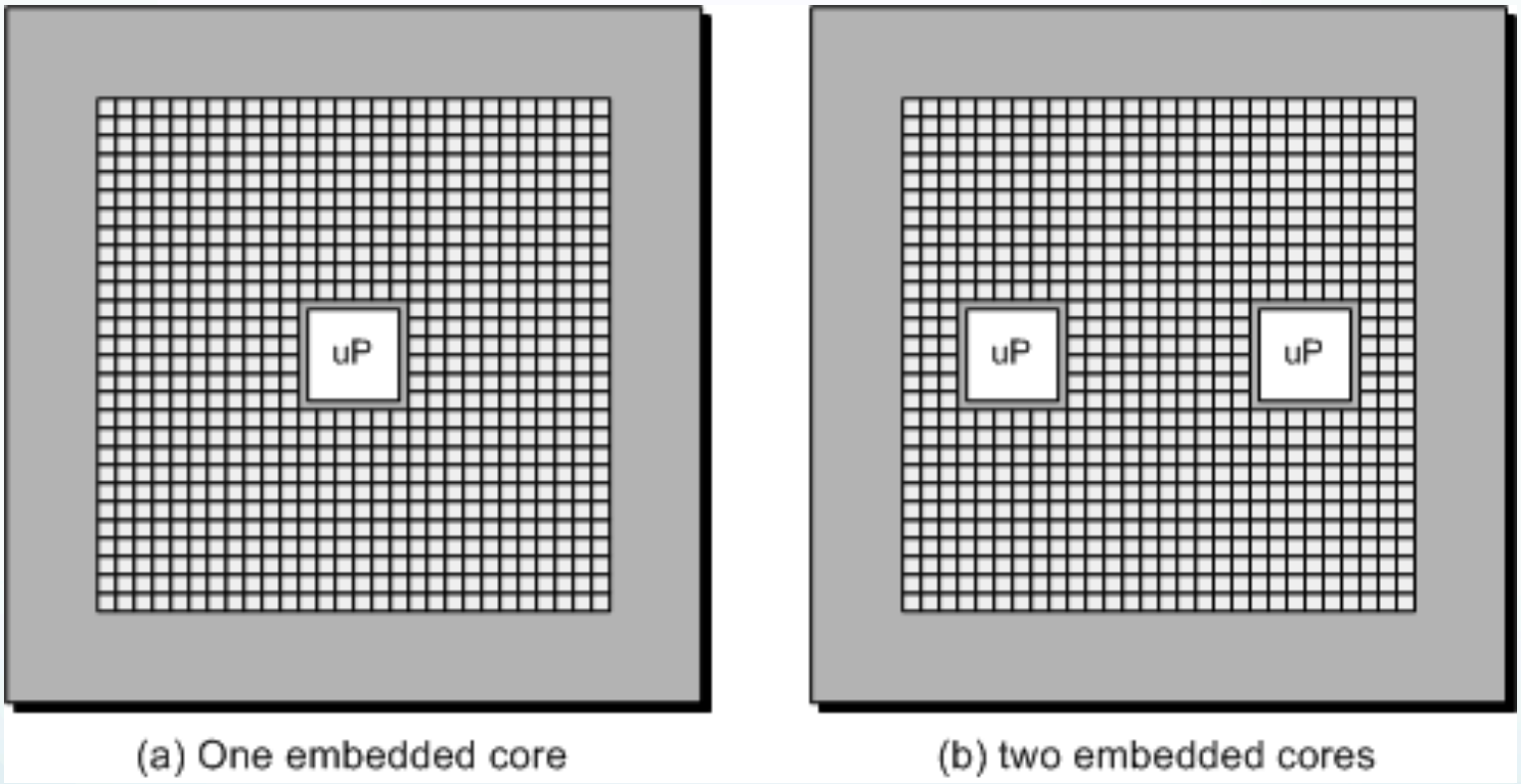


Today: Up to several MB of RAM

Embedded Multipliers



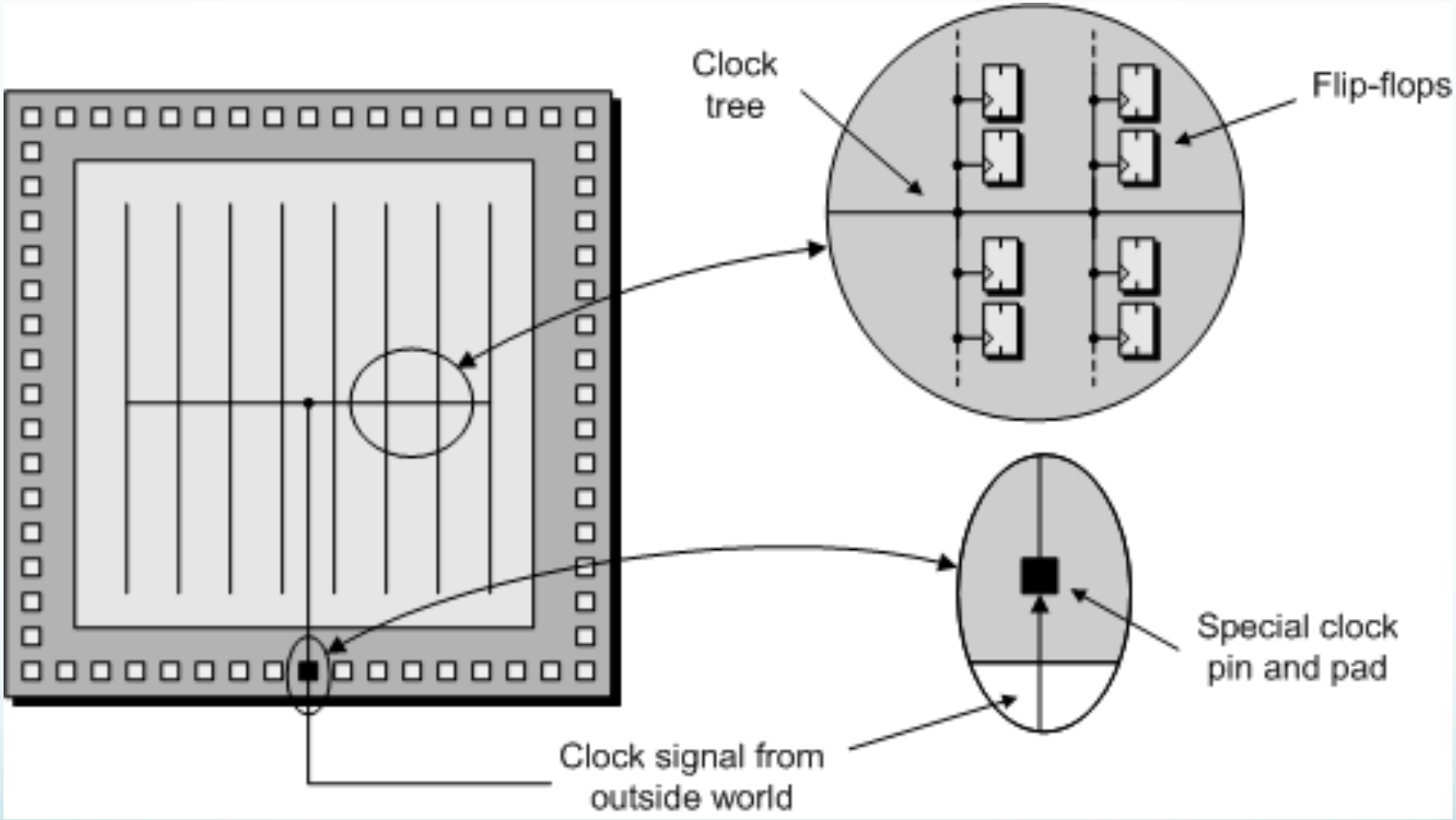
Soft and Hard Processor Cores



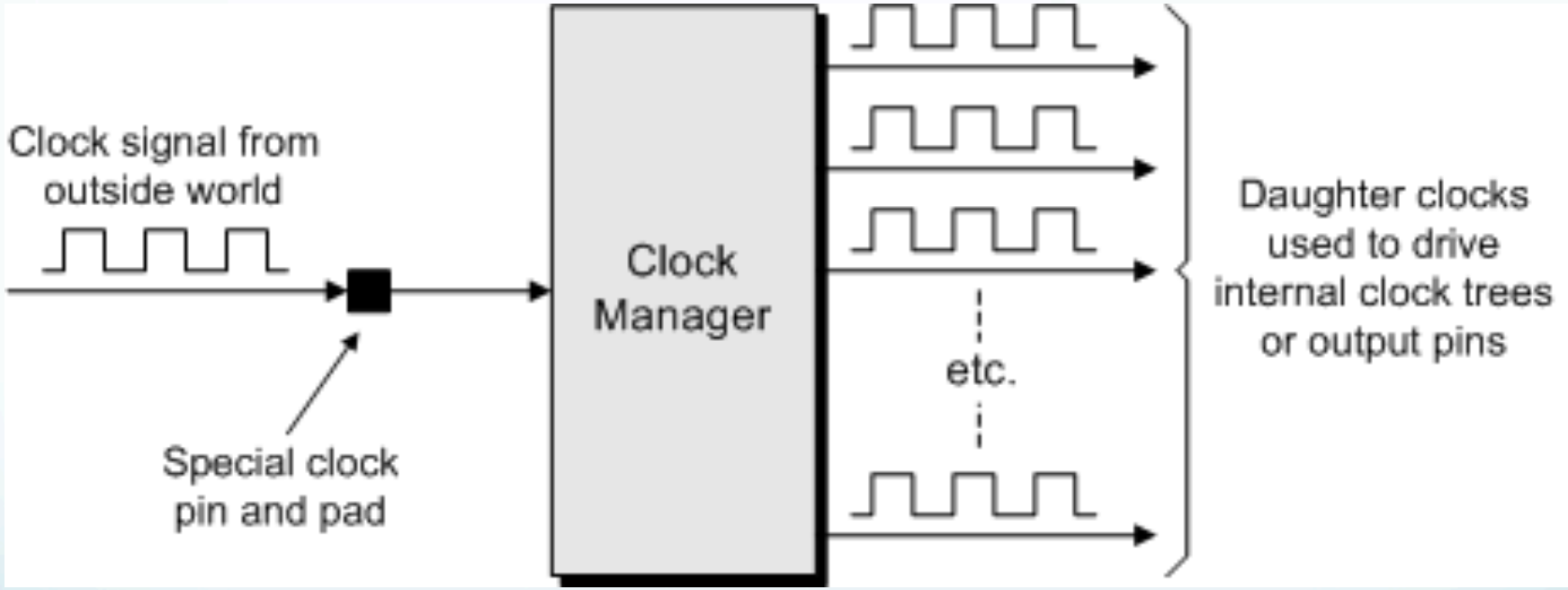
(a) One embedded core

(b) two embedded cores

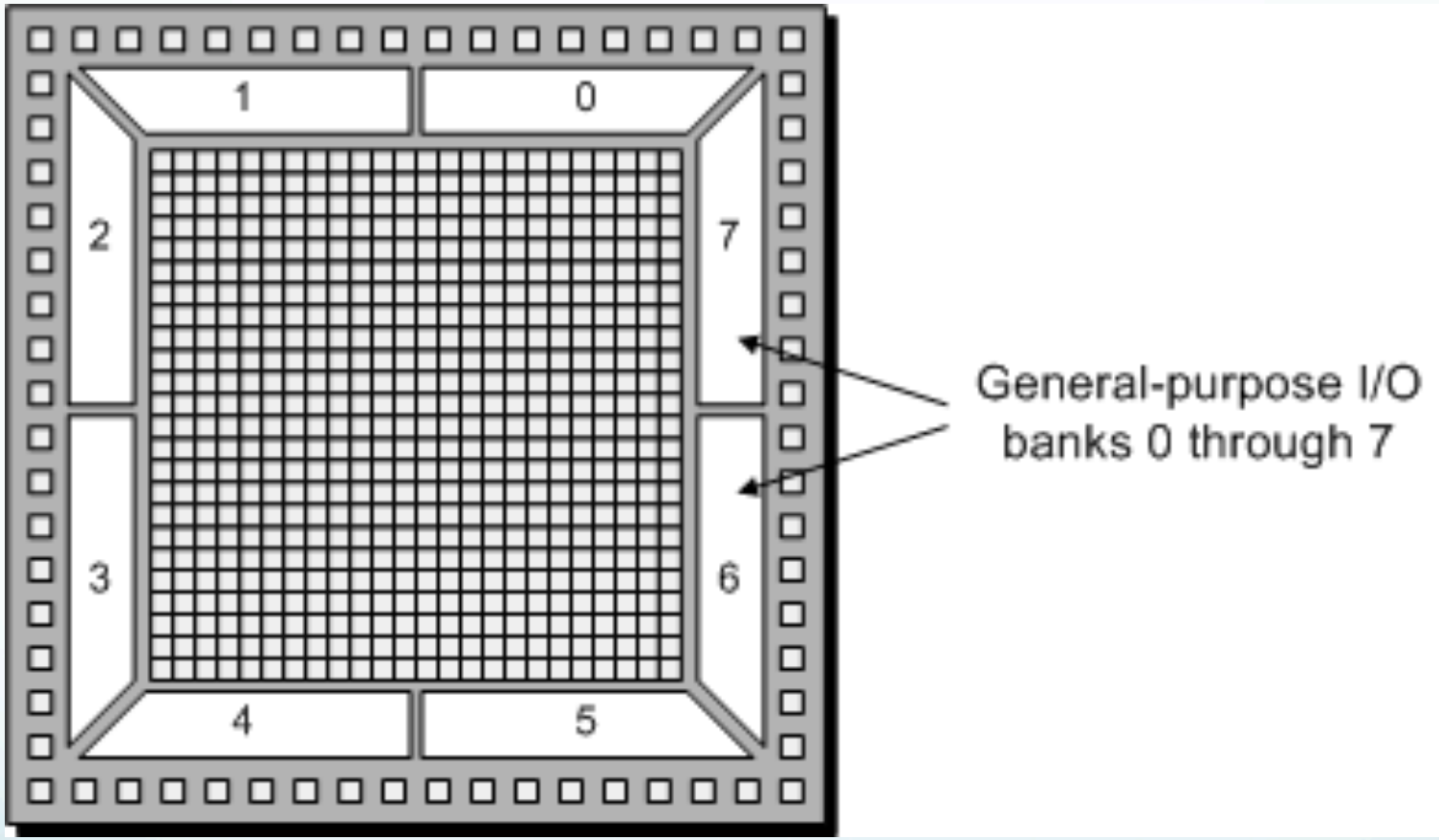
Clock Trees


Clock
tree
Flip-flops
Special clock
pin and pad
Clock signal from
outside world

Clock Managers



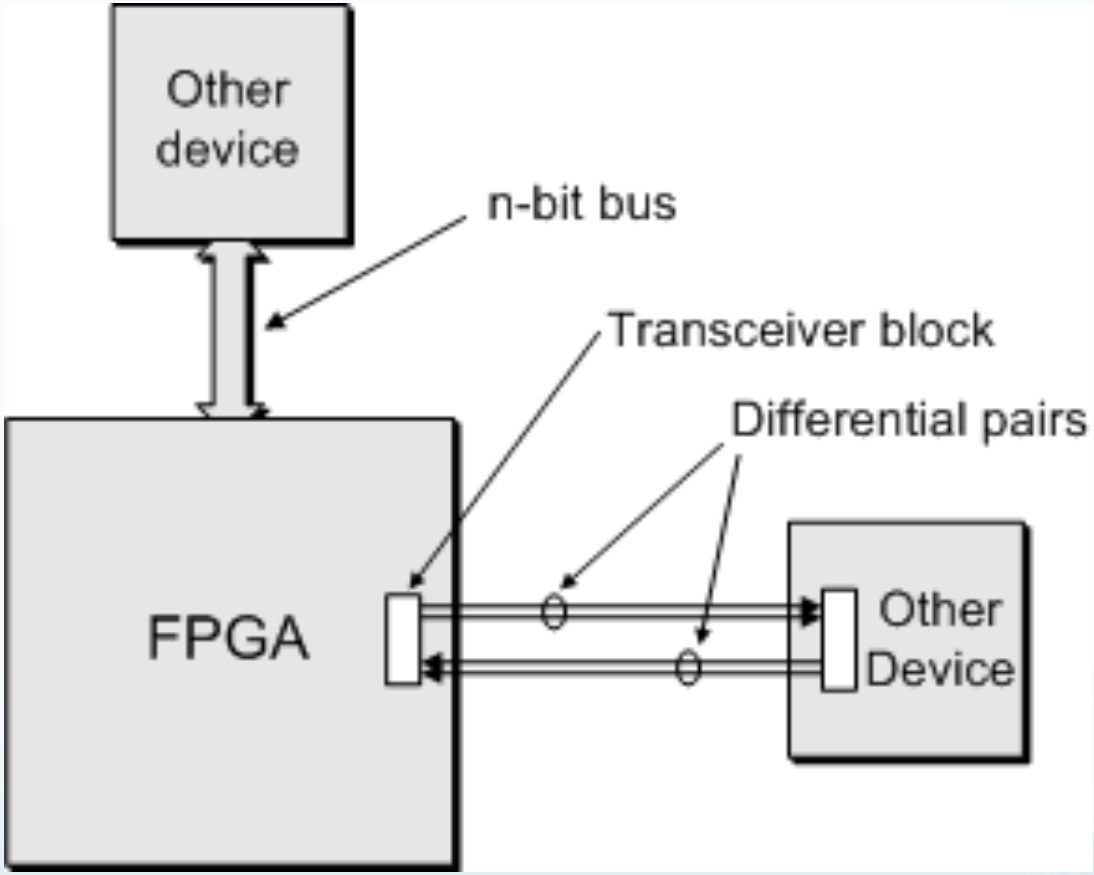
General-Purpose Input/Output (GPIO)



Today: Up to 1200 user I/O pins

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

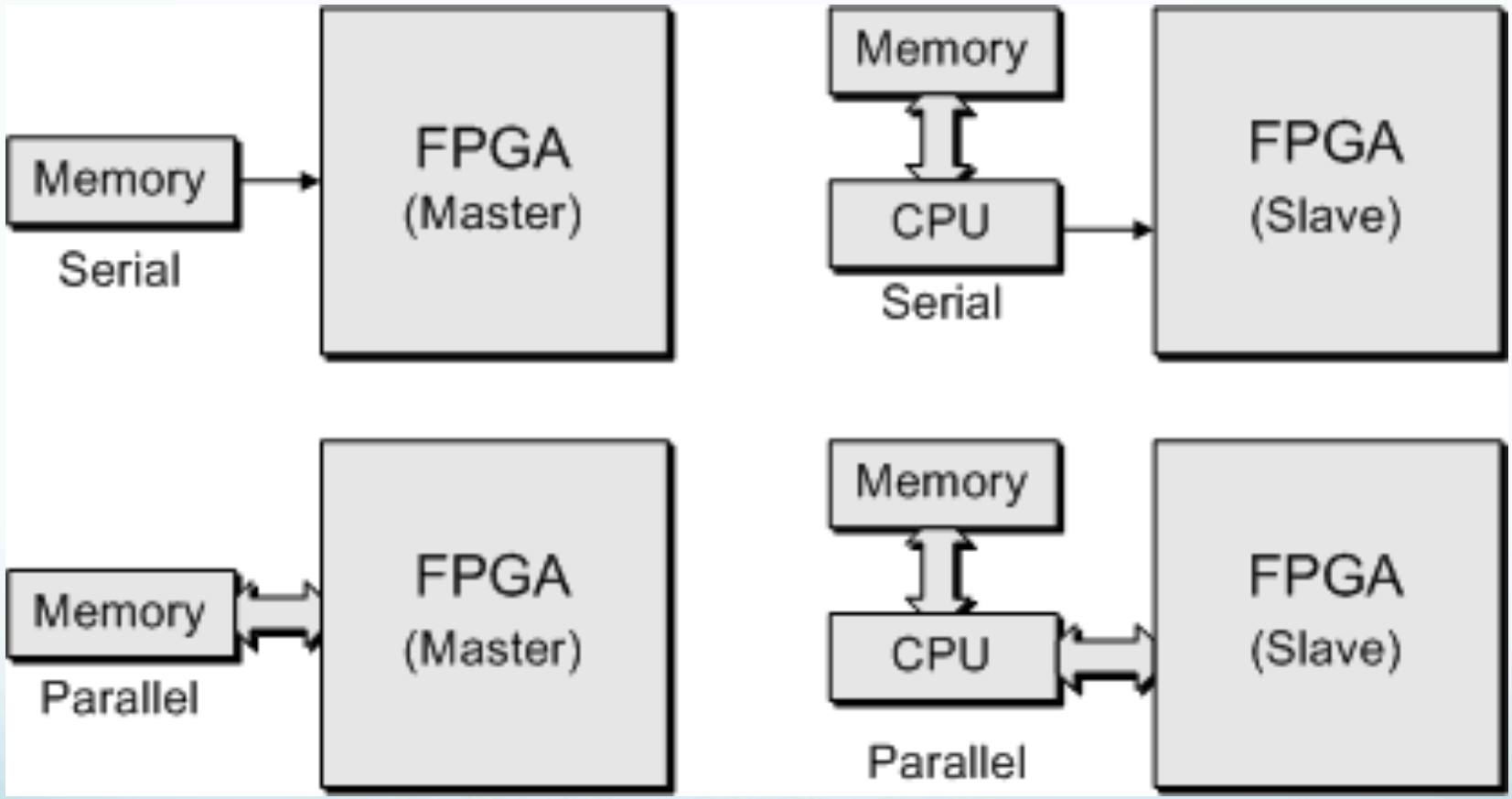
High-Speed Serial Interconnect



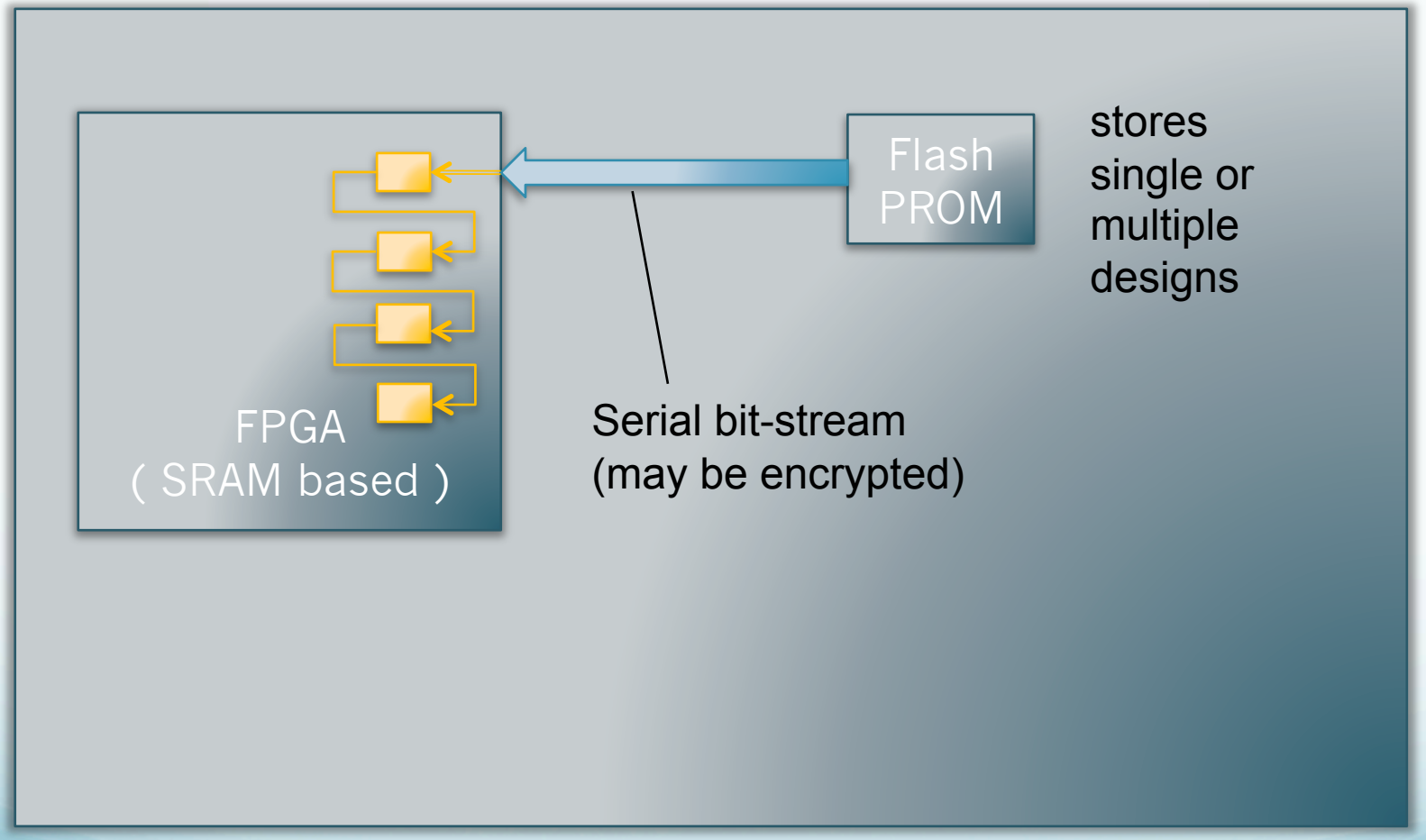
Trends

- Increased capacity and speed
- Hard macro cores
- Embedded CPUs
- Sophisticated interfaces
- Domain-specific devices
- Ultra-high-capacity/performance FPGAs
- Ultra-low-power FPGAs
- Mixed-signal FPGAs

Design Considerations (SRAM Config.)



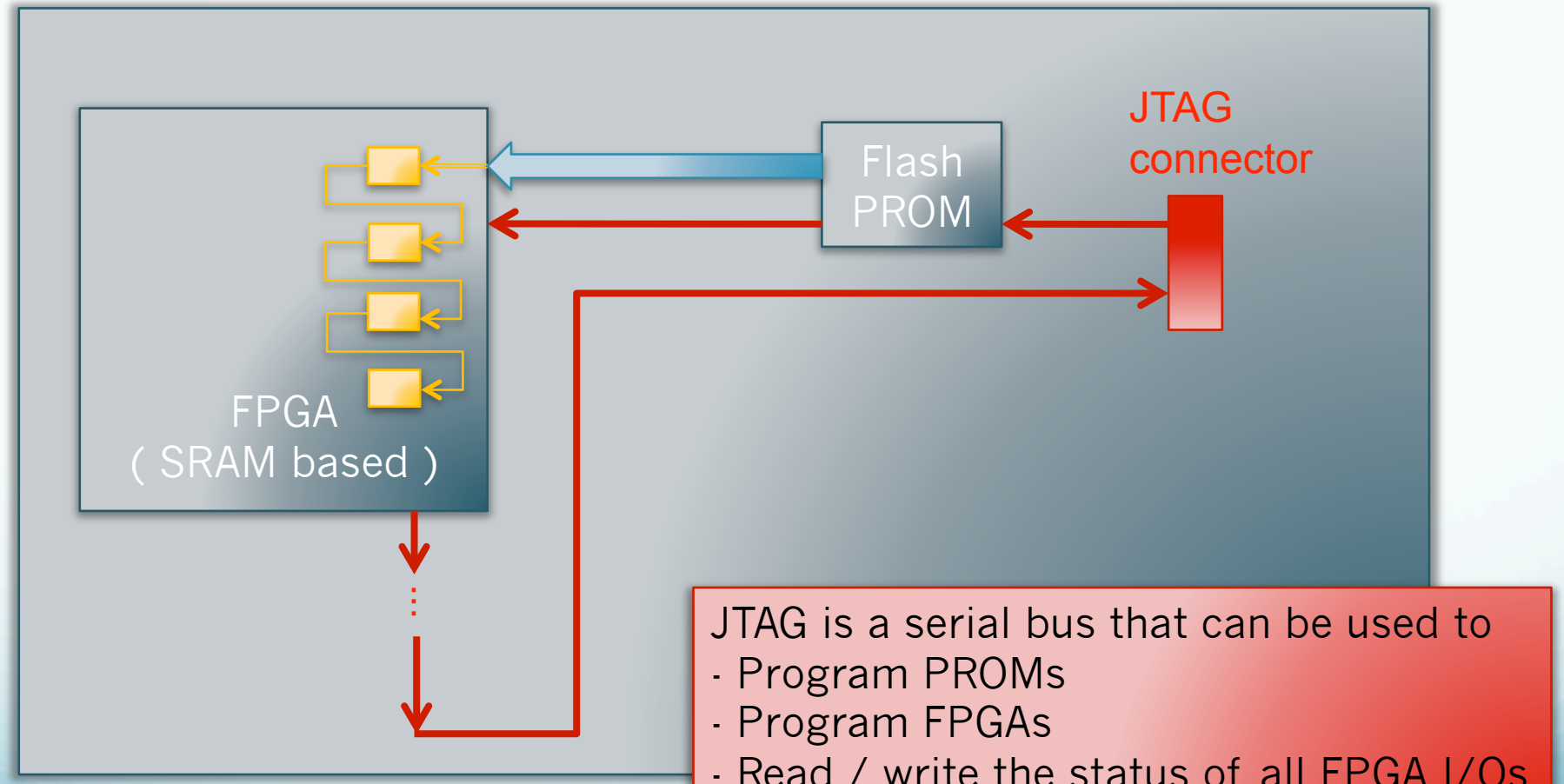
Configuration at power-up



Typical FPGA configuration time: milliseconds

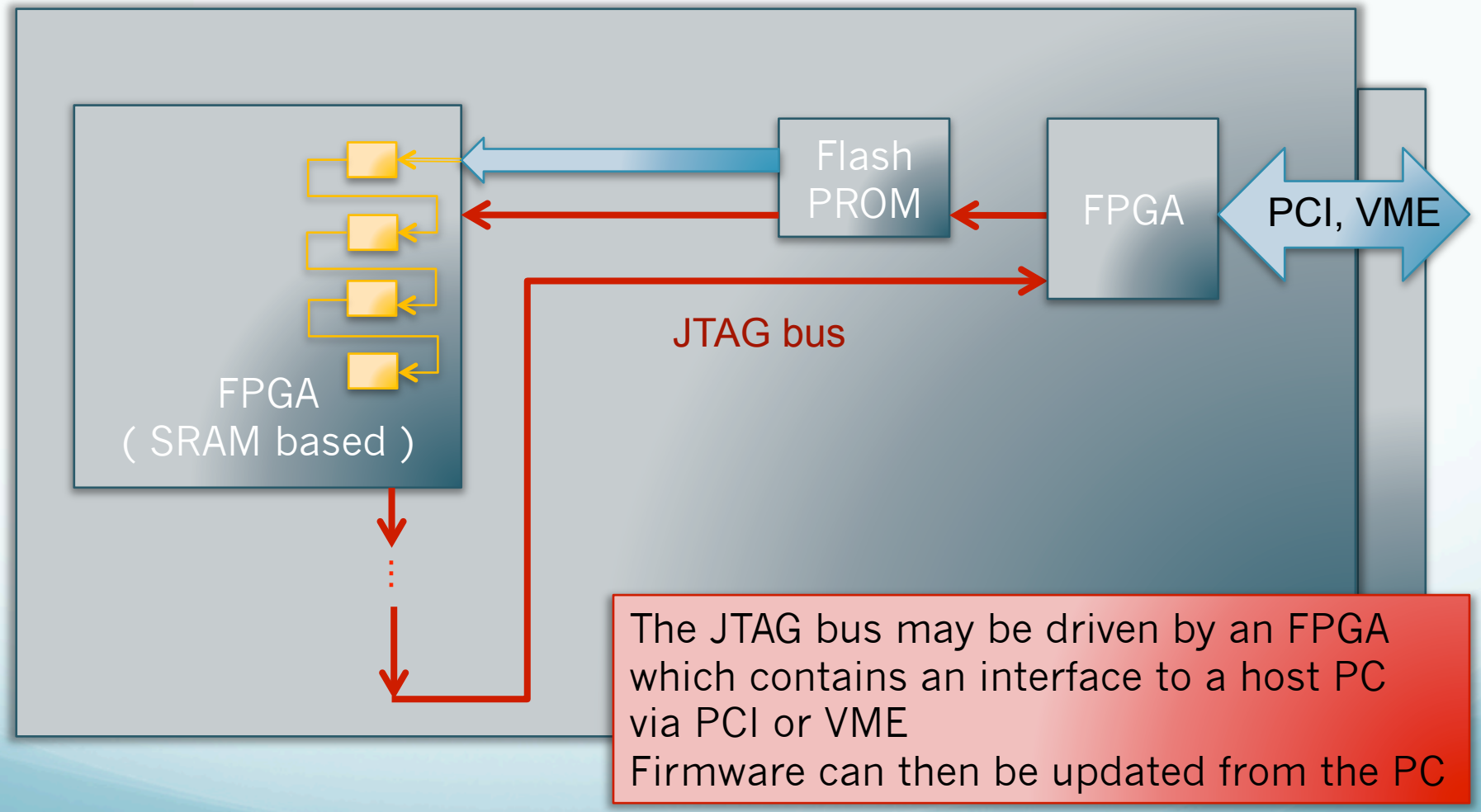
Programming via JTAG

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

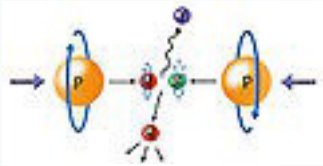


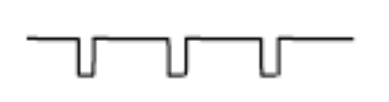
Major Manufacturers

- Xilinx
  - First company to produce FPGAs in 1985
  - About 50% market share, today
  - SRAM based CMOS devices
- Altera
  - About 30% market share
  - SRAM based CMOS devices
- Actel
  - Anti-fuse FPGAs
  - Flash based FPGAs
  - Mixed Signal
- Lattice Semiconductor
  - SRAM based with integrated Flash PROM

FPGA applications

First-Level Trigger at Collider





Timing: beam crossings

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

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

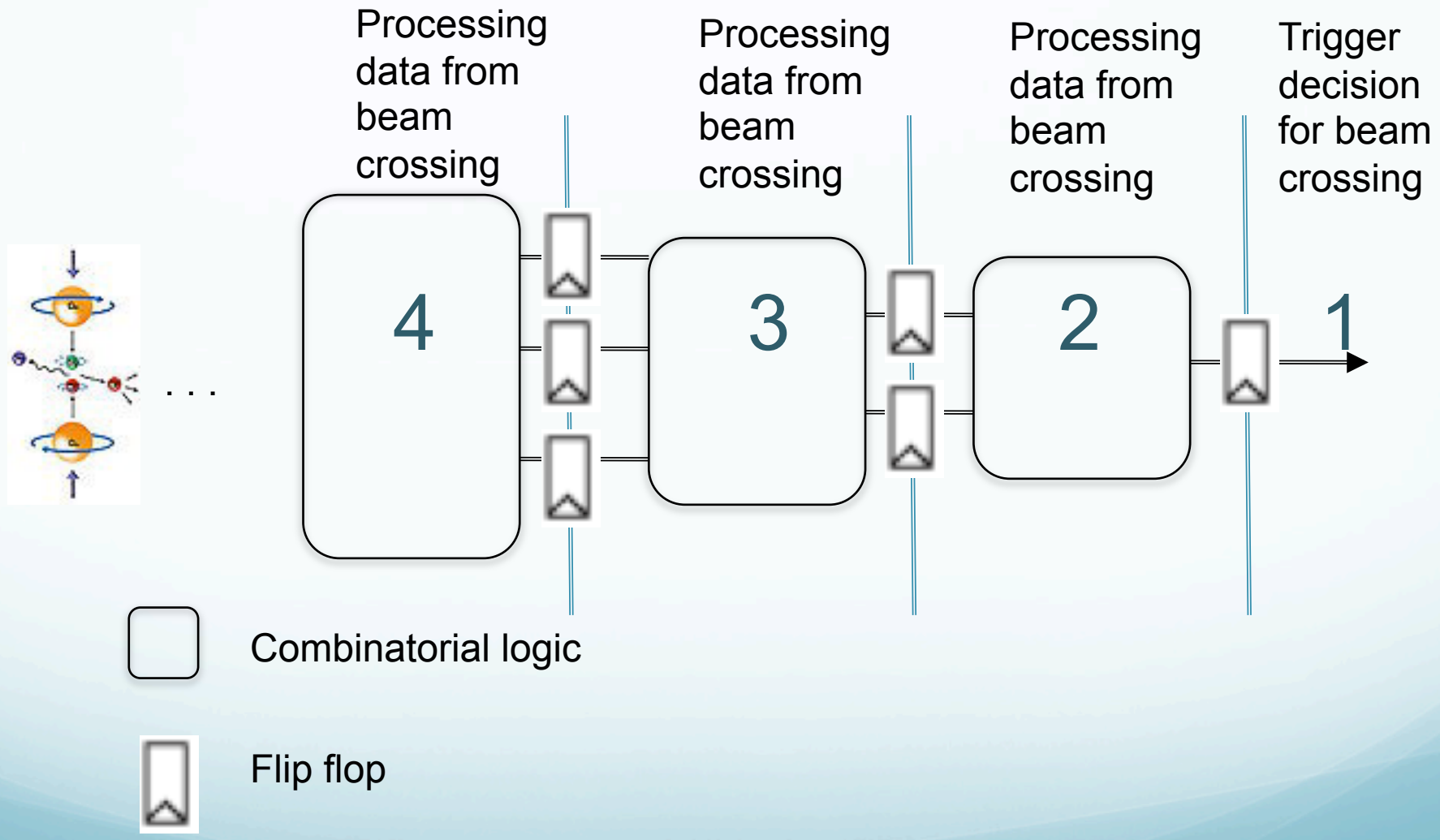
Low latency

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

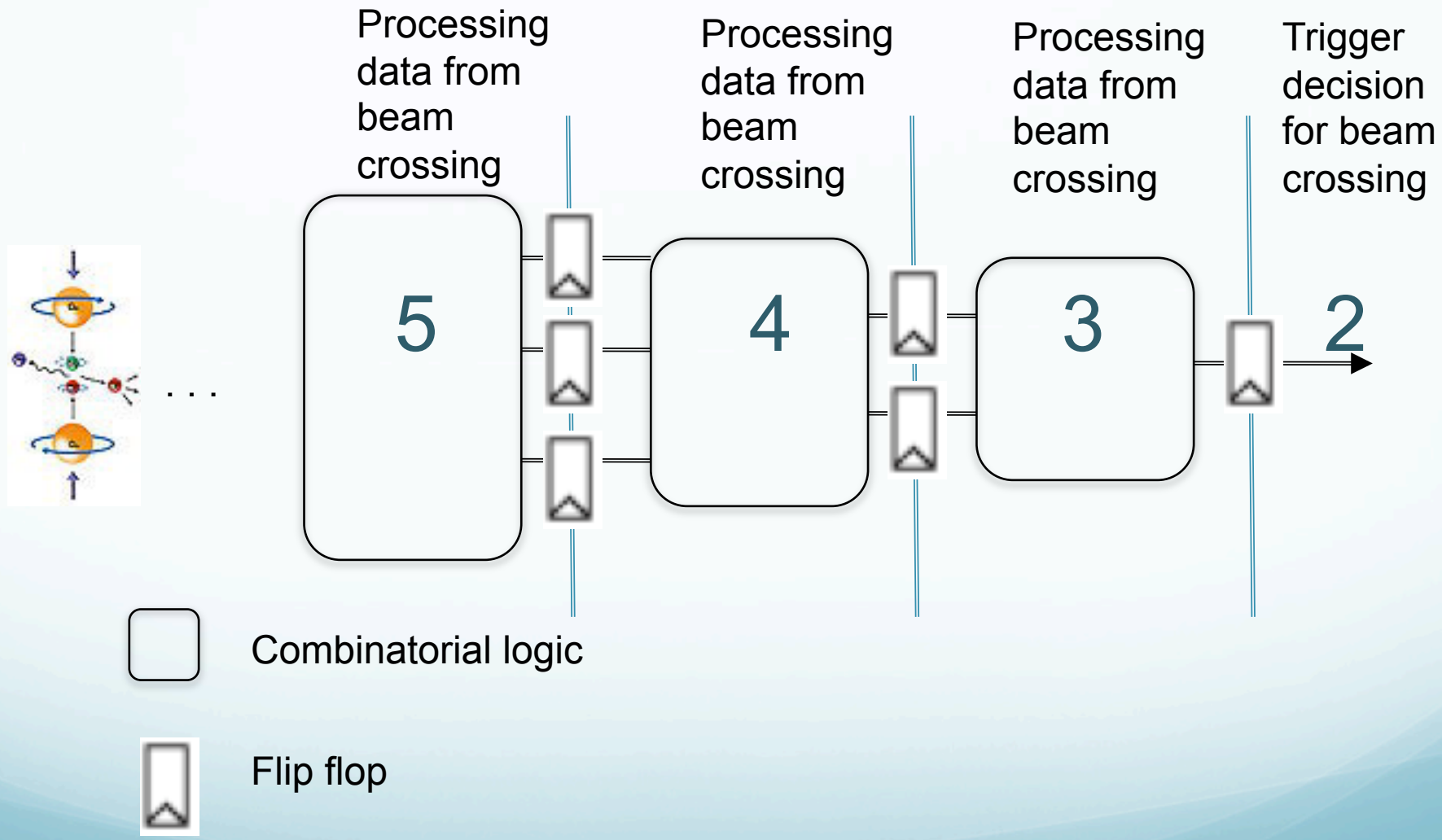
High performance

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

Pipelined Logic



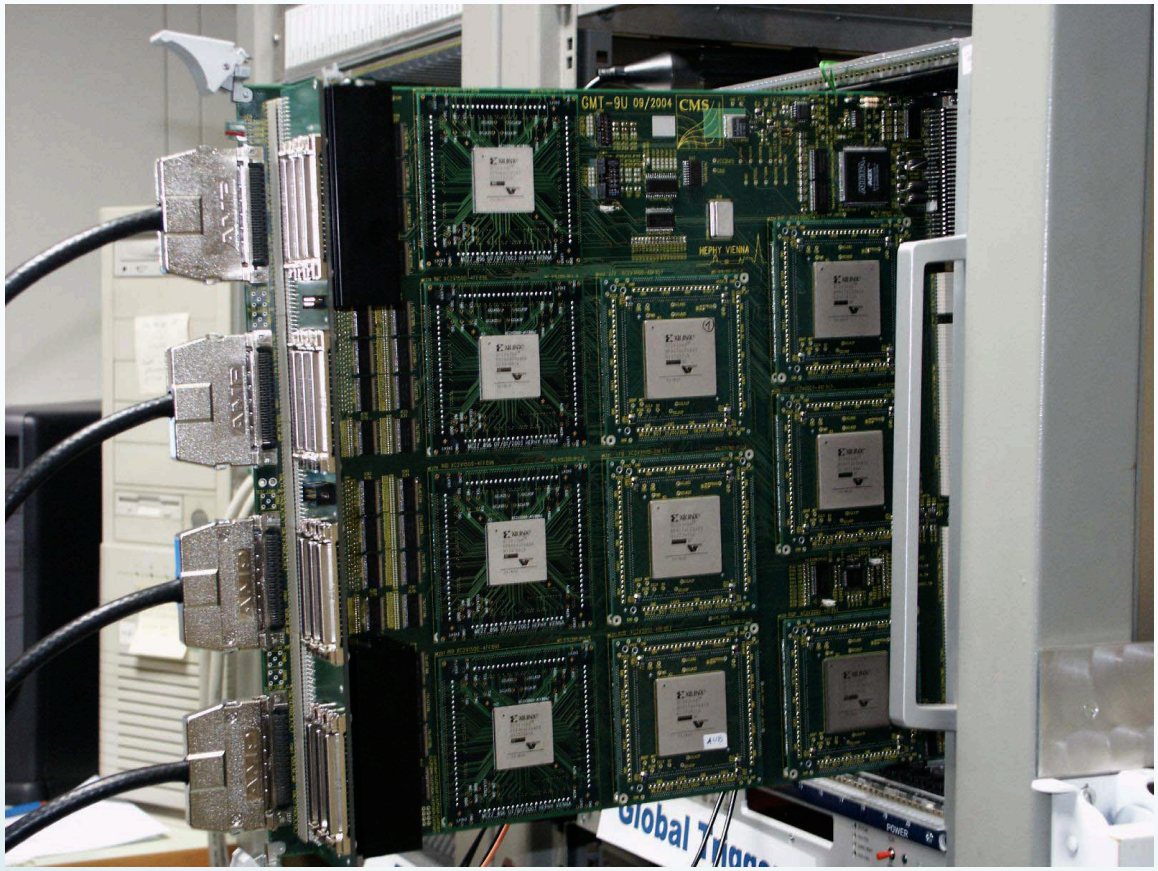
Pipelined Logic – a clock cycle later



Trigger algorithms implemented in FPGAs

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

Example: CMS Global Muon Trigger

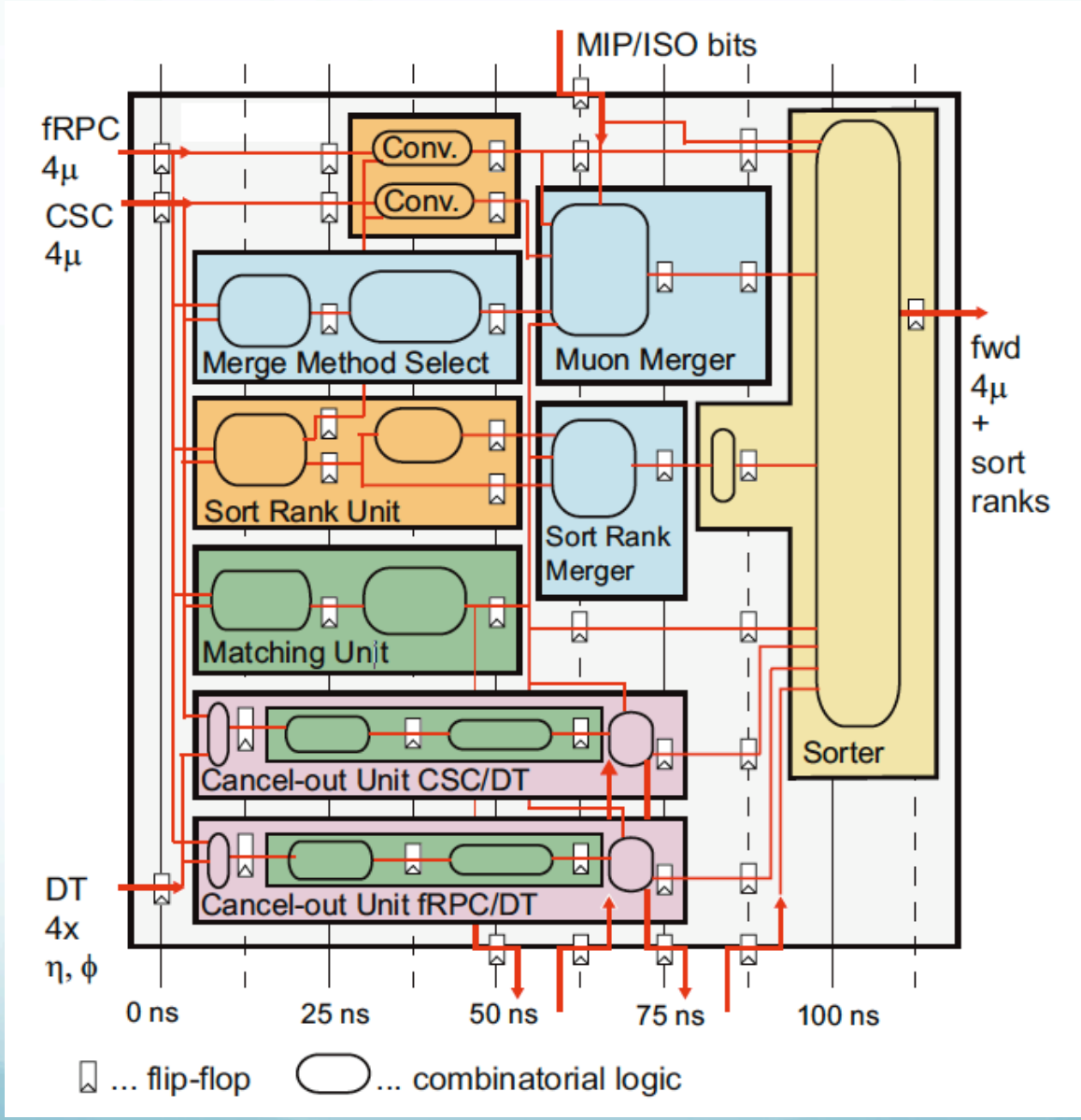


- Input: ~1000 bits @ 40 and 80 MHz
- Output: ~50 bits @ 80MHz
- Processing time: 250 ns
- Pipelined logic one new result every 25 ns

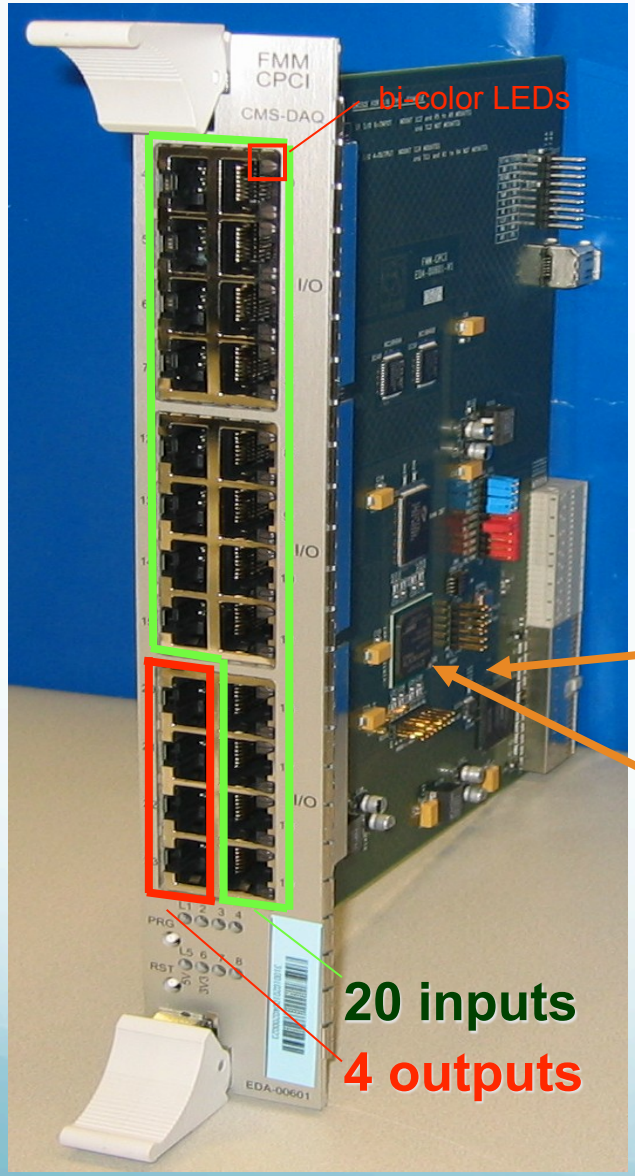
- 10 Xilinx Virtex-II FPGAs
- up to 500 user I/Os per chip
- Up to 25000 LUTs per chip used
- Up to 96 x 18kbit RAM used

- 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

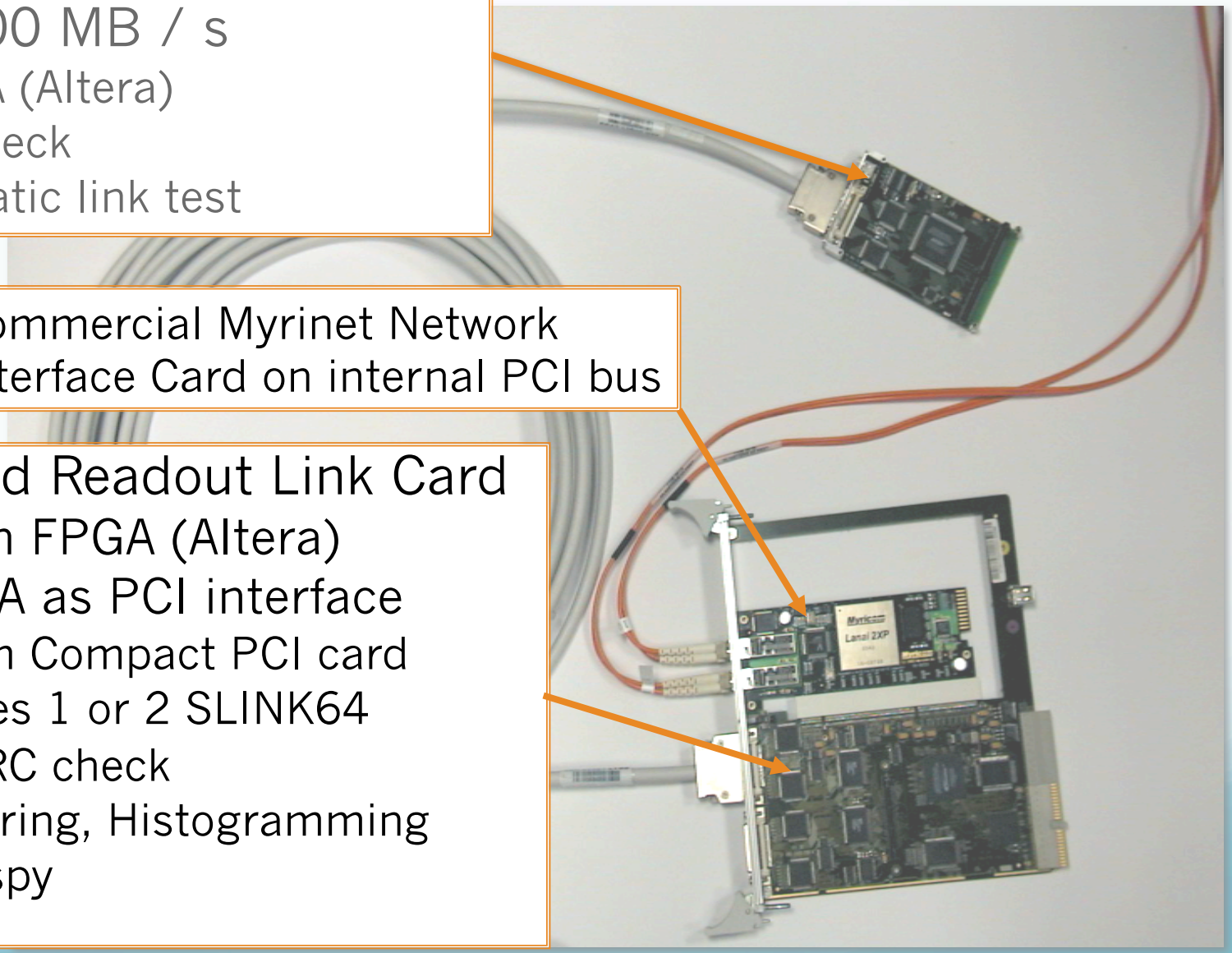
- Frontend Electronics
  - Pedestal subtraction
  - Zero suppression
  - Compression
  - ...
- Custom data links
  - E.g. SLINK-64
    - Several serial LVDS links in parallel
    - Up to 400 MB/s
- Interface from custom hardware to commercial electronics
  - PCI bus, VME bus, Myrinet, etc.

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

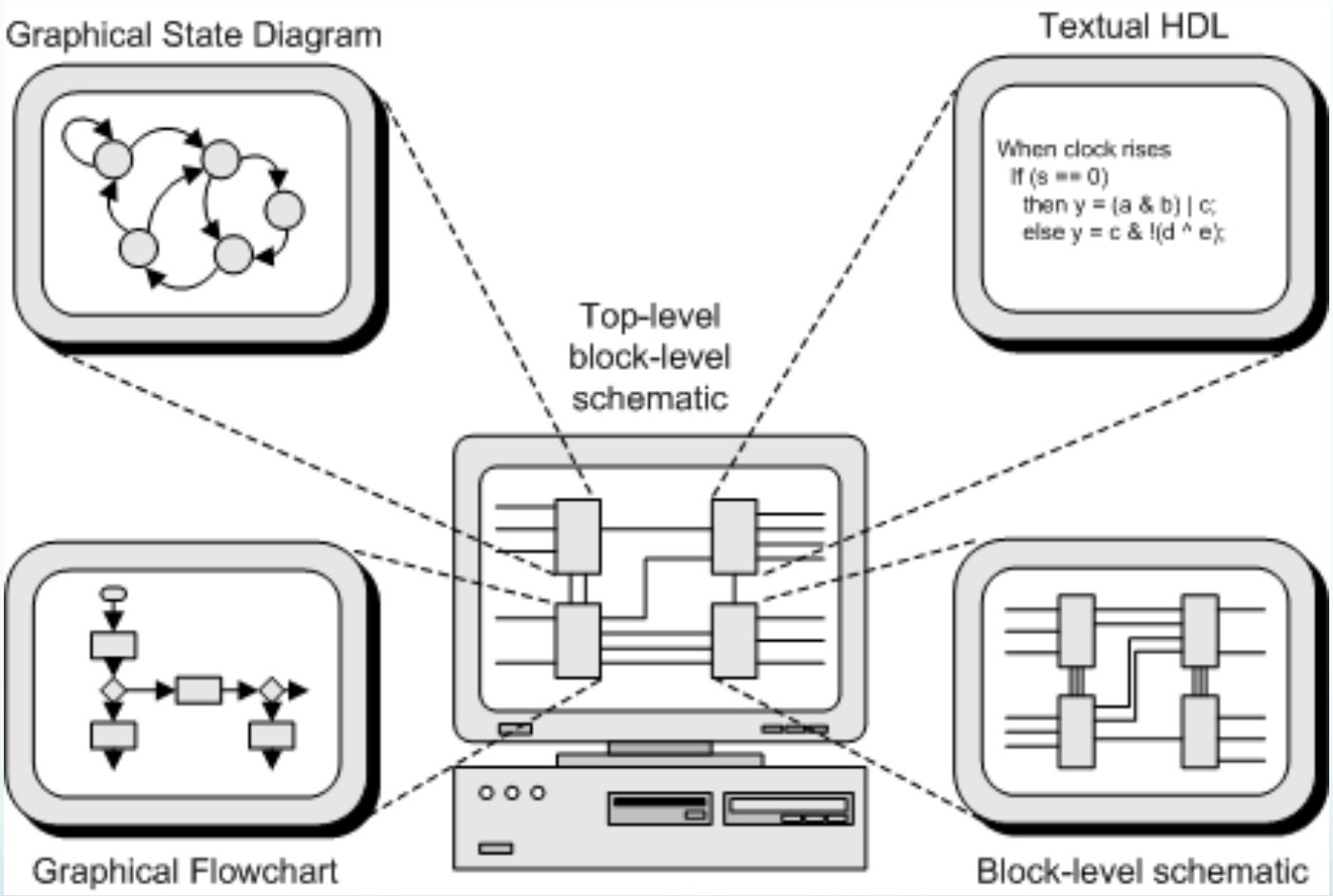


FPGAs in other domains

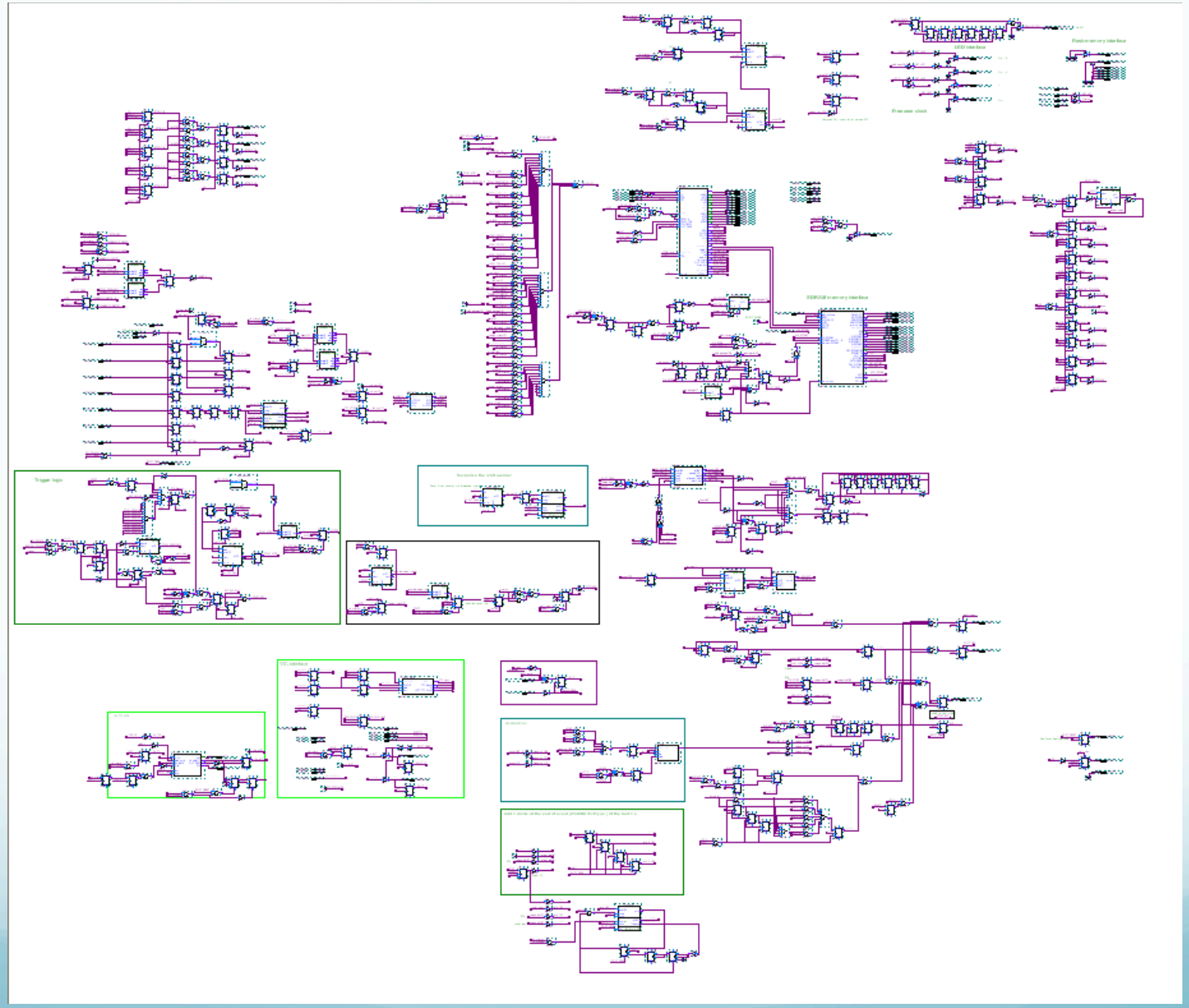
- Set-top boxes
- Medical imaging
- Computer vision
- Speech recognition
- Cryptography
- Bioinformatics
- Software-Defined Radio
- Aerospace
- Defense
- Digital Signal Processing
- ASIC Prototyping
- High performance computing
  - Computations performed by FPGA: FFT, Convolution
- Reconfigurable computing

FPGA development

Development Tools



Schematics



Hardware Description Language

- Similar to a programming language
- Common HDLs
  - VHDL
    (VHSIC hardware description language;
    VHSIC: very-high-speed integrated circuit)
  - Verilog
  - AHDL (Altera specific)
- Newer trends
  - C-like languages (handle-C)
  - Labview

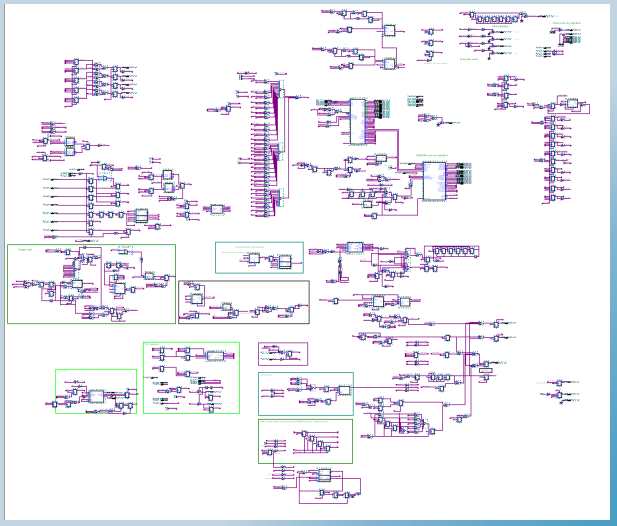
Example: VHDL

```
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

Design entry considerations

Schematics



- Graphical overview
- Can draw entire design
- Can also include blocks in hardware description language.

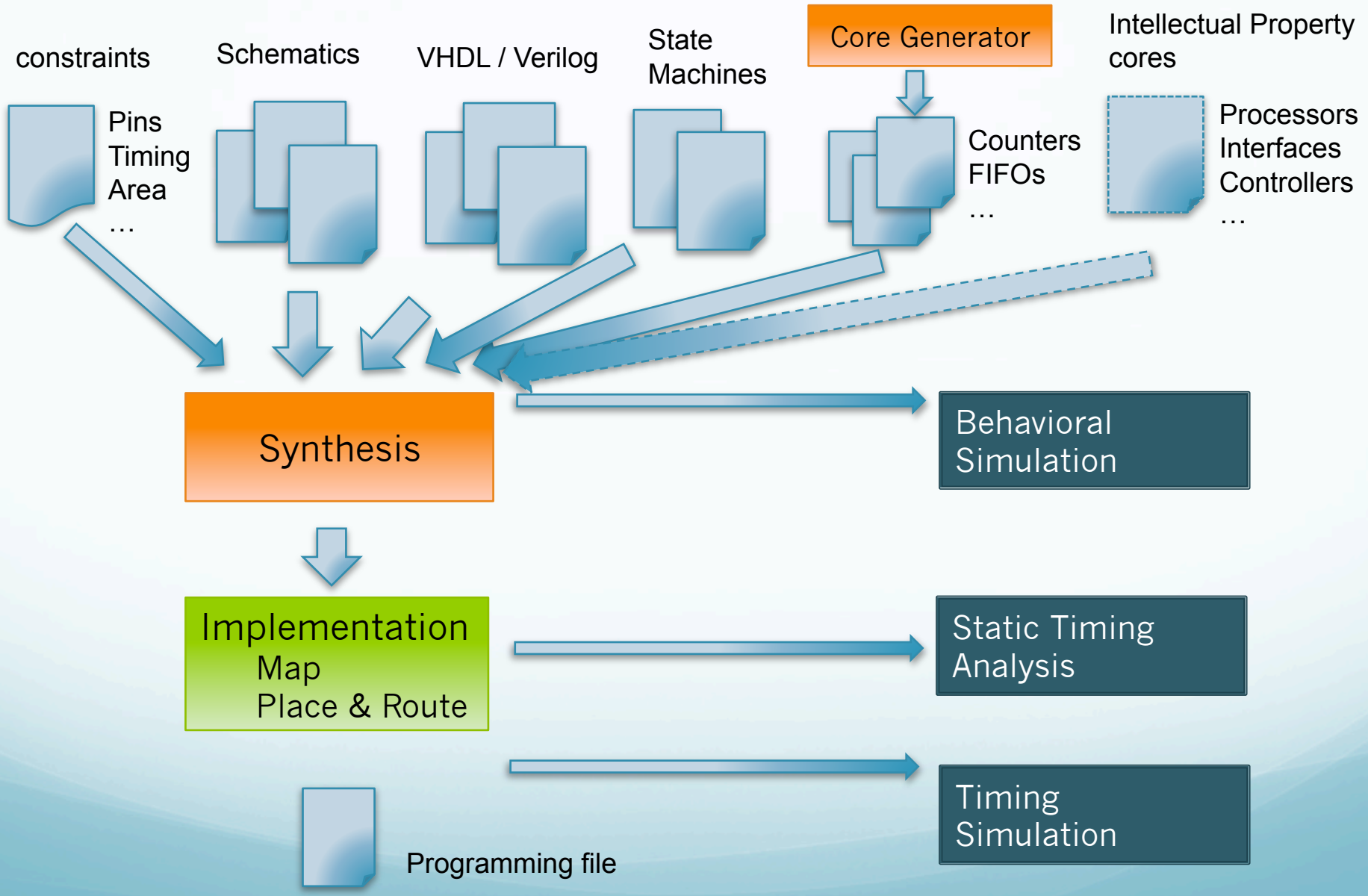
Hardware description language
VHDL, Verilog

```
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;
```

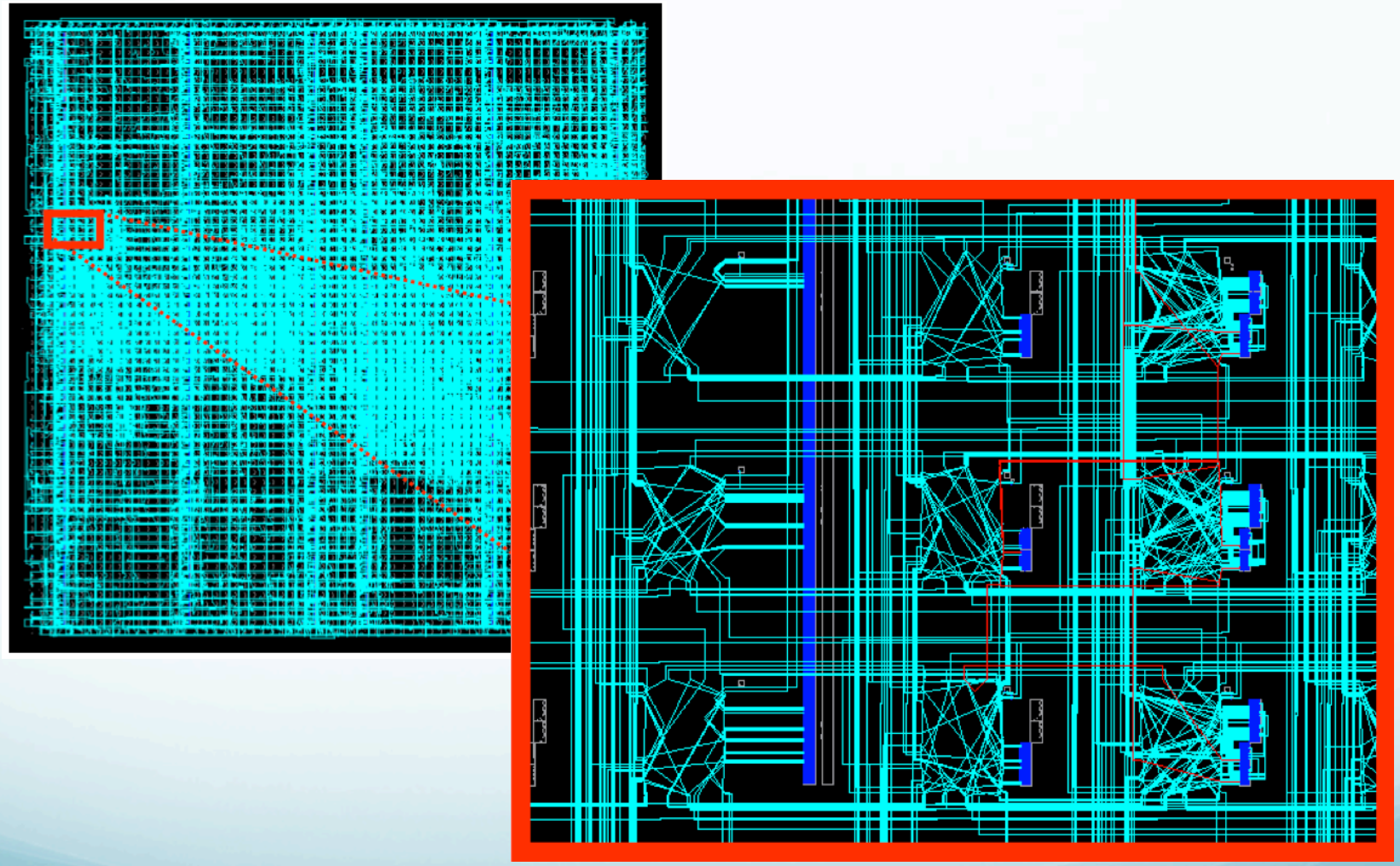
- Can generate blocks using loops
- Can synthesize algorithms
- Independent of editor
- May use tools used in SW development (CVS, ...)

Mostly a personal choice depending on previous experience

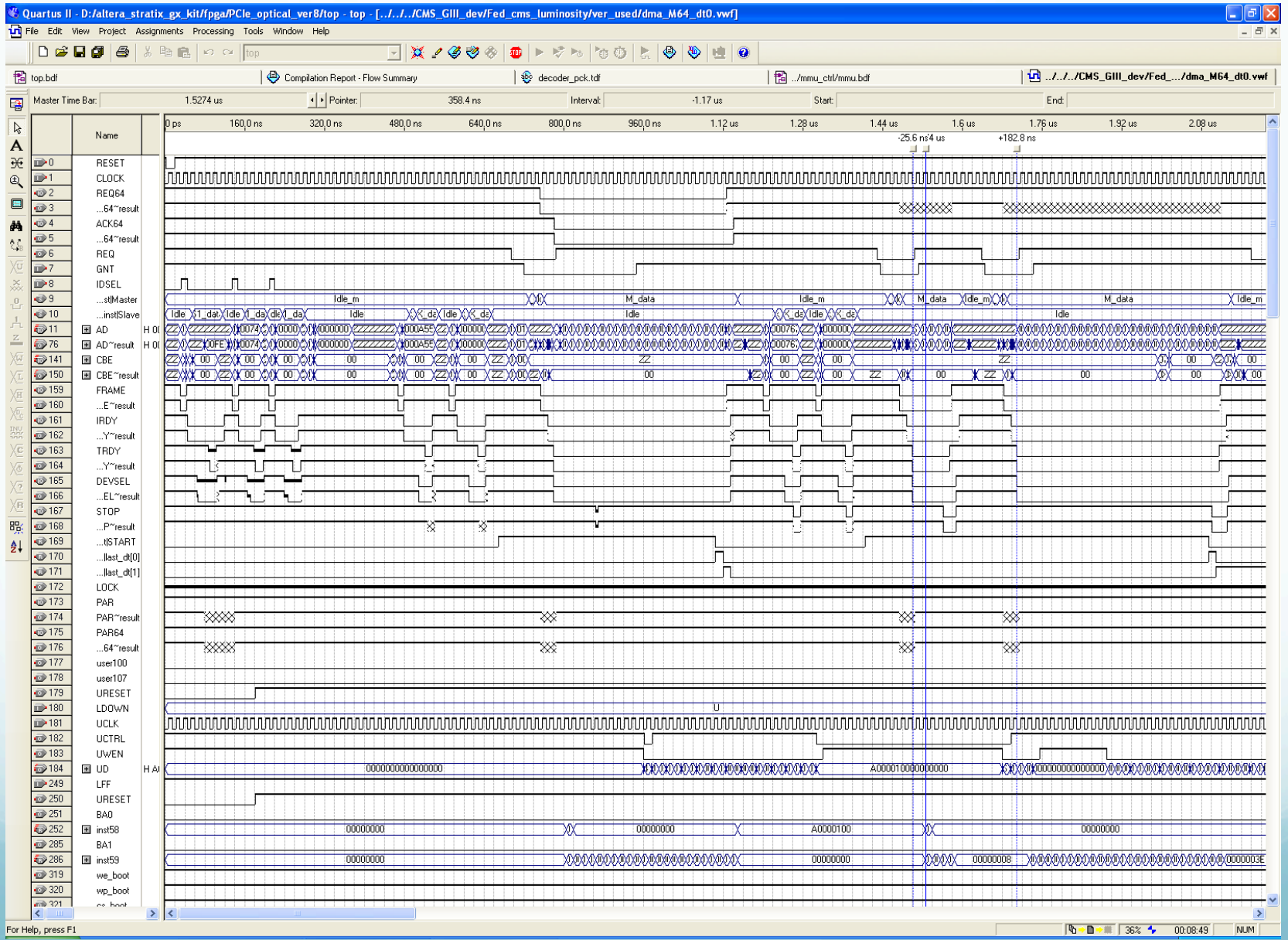
Design flow


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

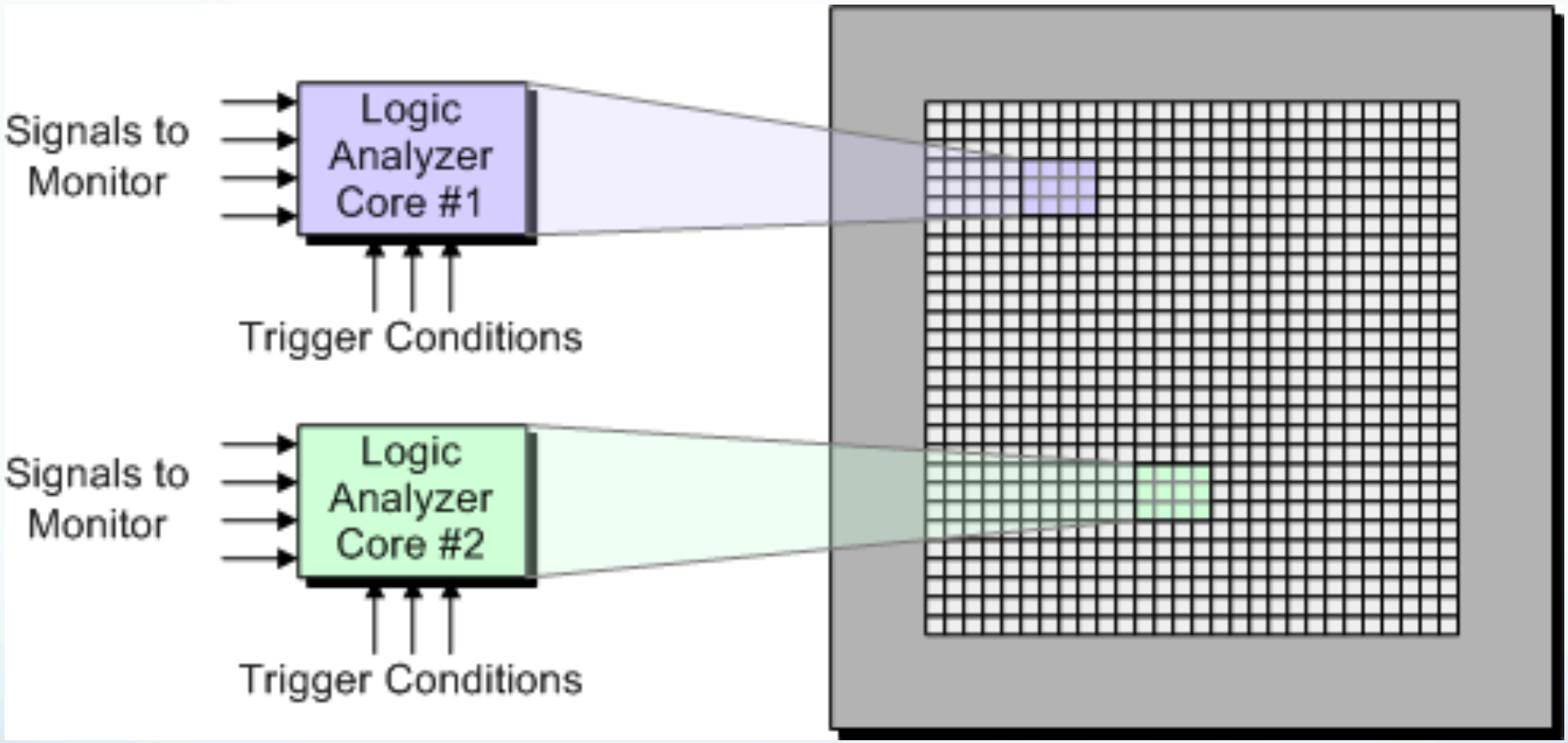
Floorplan



Simulation

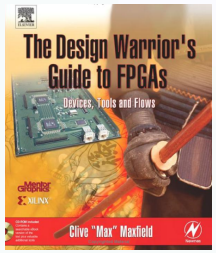

Quartus II - D:/altera_stratix_gx_kit/fpga/PCIe_optical_ver8/top - top - [../../../CMS_GIII_dev/Fed_cms_luminosity/ver_used/dma_M64_dt0.vwf]
File Edit View Project Assignments Processing Tools Window Help
top.bdf
Compilation Report - Flow Summary
decoder_pck.tdf
../mmu_ctrl/mmu.bdf
../../../CMS_GIII_dev/Fed_.../dma_M64_dt0.vwf
Master Time Bar: 1.5274 us
Pointer: 358.4 ns
Interval: -1.17 us
Start:
End:
Name
RESET
CLOCK
REQ64
ACK64
REQ
GNT
IDSEL
AD
CBE
FRAME
IRDY
TRDY
DEVSEL
STOP
LOCK
PAR
PAR64
user100
user107
URESET
LDOWN
UCLK
UCTRL
UWEN
UD
LFF
URESET
BA0
BA1
we_boot
wp_boot
Idle
M_data
For Help, press F1
NUM

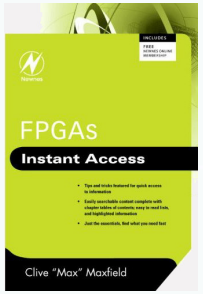
Embedded Logic Analyzers

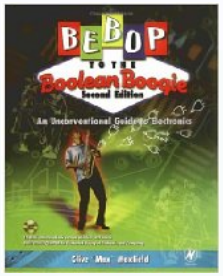


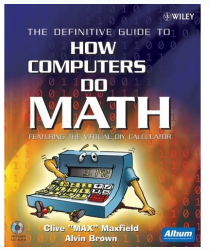
Thanks

- 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!







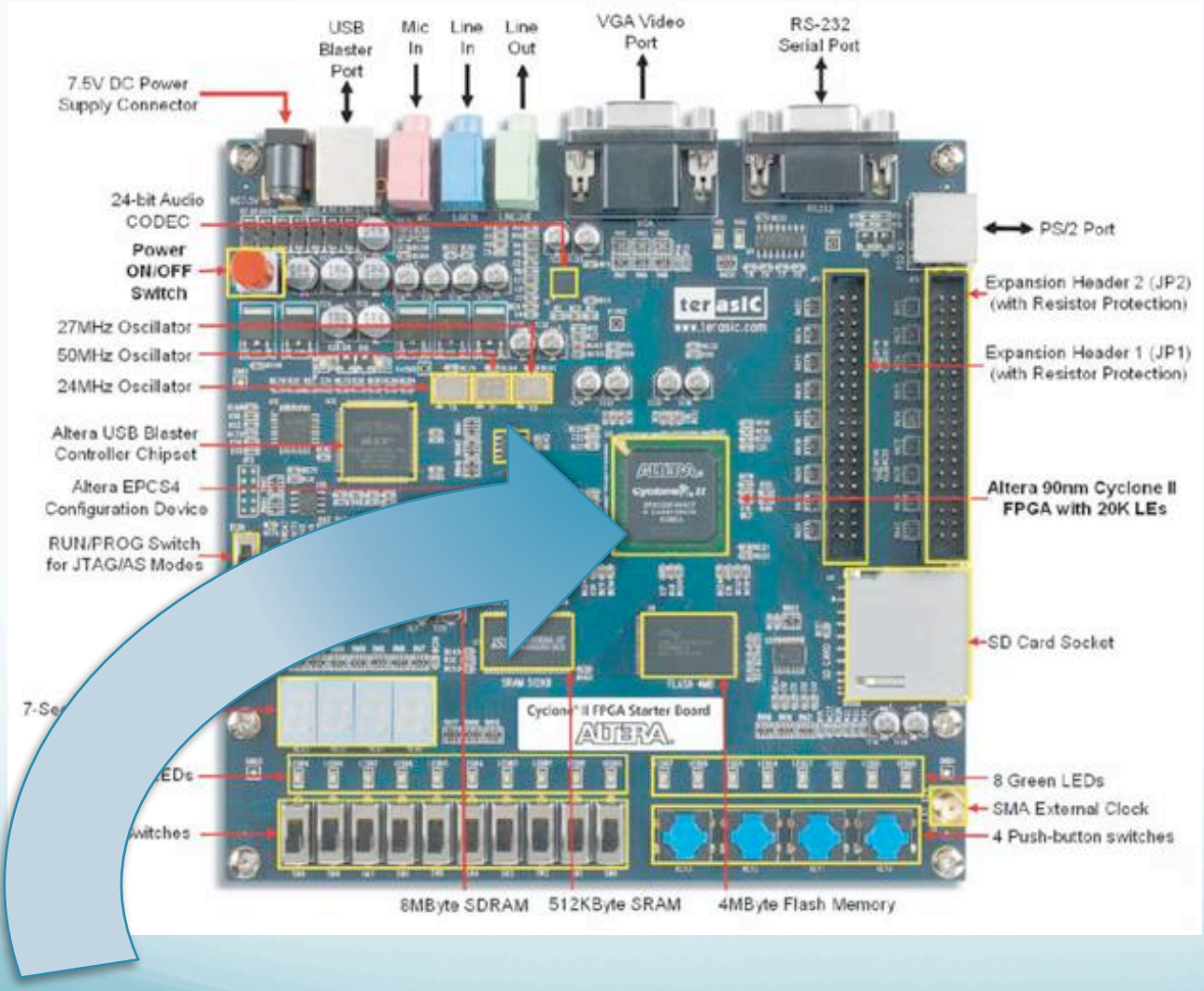


Web lecture: courses.techonline.com
Max@techbites.com , www.TechBites.com

- 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 !