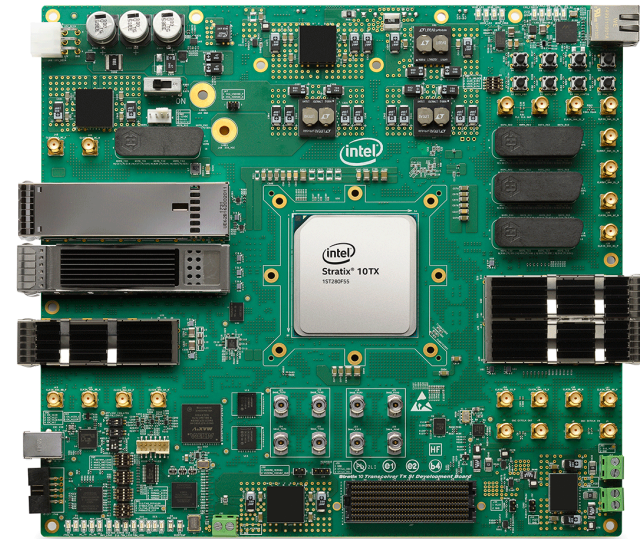




Introduction to Field Programmable Gate

Arrays (FPGAs)

MicroLab, NTUA



What is a **Field Programmable Gate Array** ?

.. 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 (gateware or firmware)
 - ◆ 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 bread board or soldering iron
... without plugging together NIM modules
... without having a chip produced at a factory



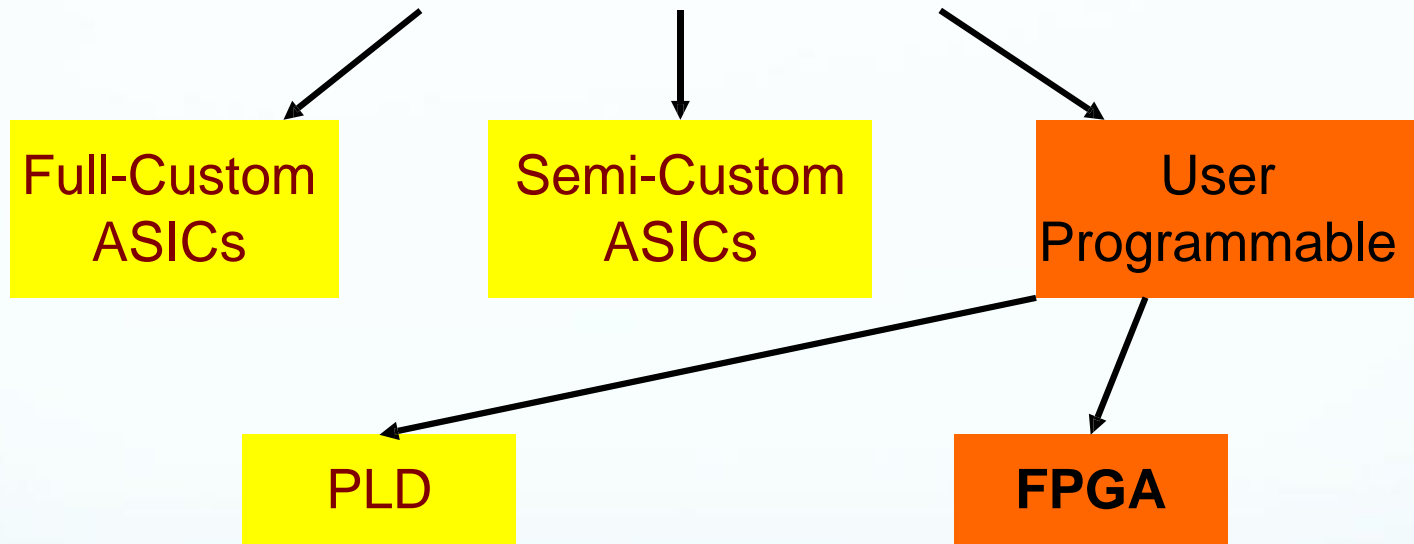
Outline

- ◆ Quick look at digital electronics
- ◆ Short history of programmable logic devices
- ◆ FPGAs and their features
- ◆ Programming techniques
- ◆ Design flow

Acknowledgement

- ◆ Parts of this lecture are based on material by of several books on FPGAs and running/completed projects.

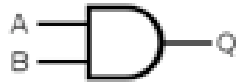
World of Integrated Circuits



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

$q = a \ \&\& \ b;$

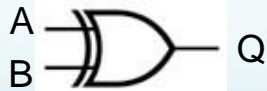
OR gate



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

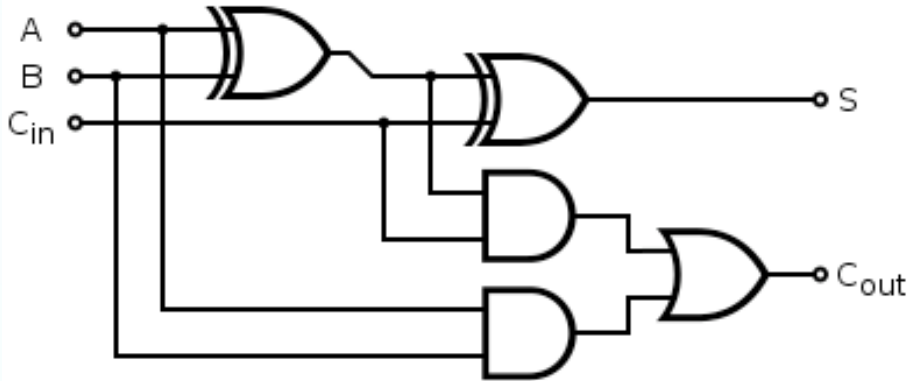


INPUT		OUTPUT
A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

$q = a \ != \ b;$

⋮

Combinatorial logic (asynchronous)



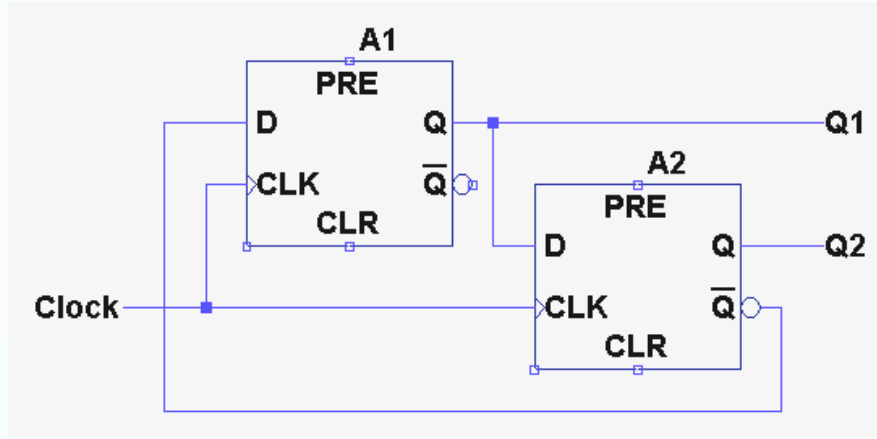
Outputs are determined by Inputs, only

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

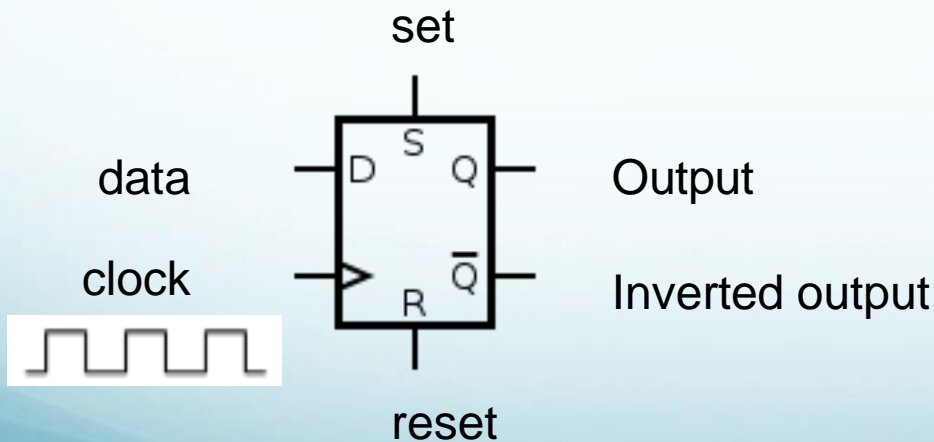
Combinatorial logic may be implemented using Look-Up Tables (LUTs)

(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

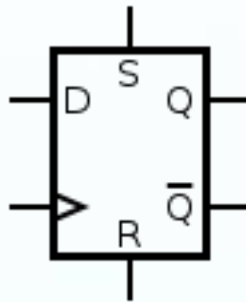
The output will be equal to the last sampled input until the next rising (or falling) clock edge

D Flip-flop (D=data, delay)

Synchronous sequential logic

LUT			
a	b	c	y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

+



=

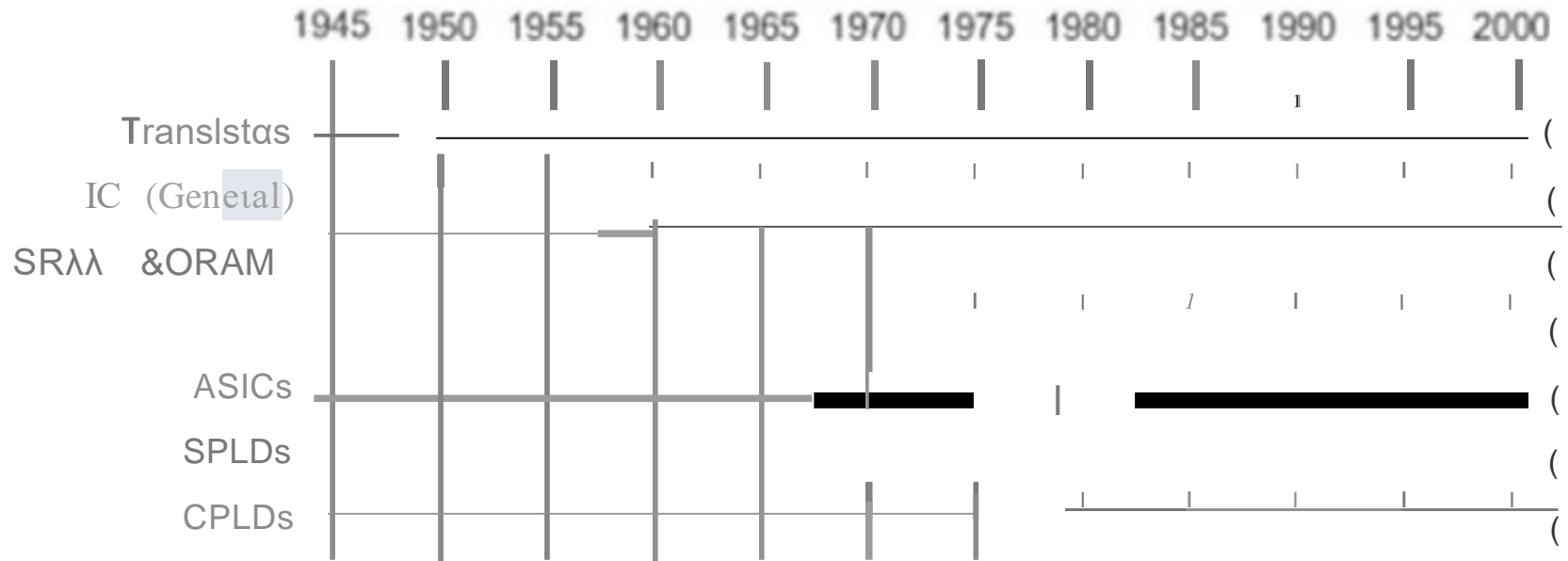


Using Look-Up-Tables and Flip-Flops
any kind of digital electronics may be implemented

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

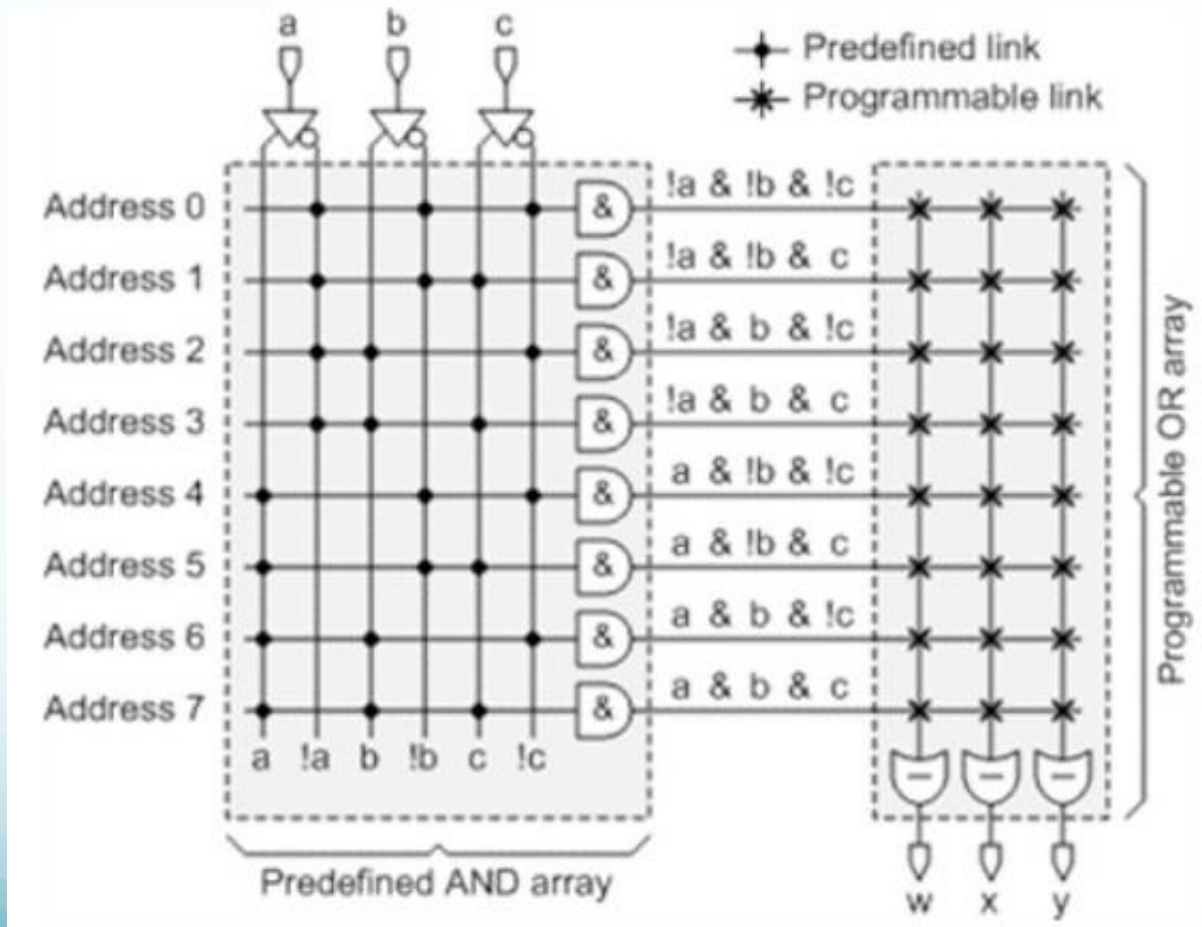
Programmable digital electronics

Long long time ago ...



Simple Programmable Logic Devices (sPLDs)

a) Programmable Read Only Memory (PROMs)

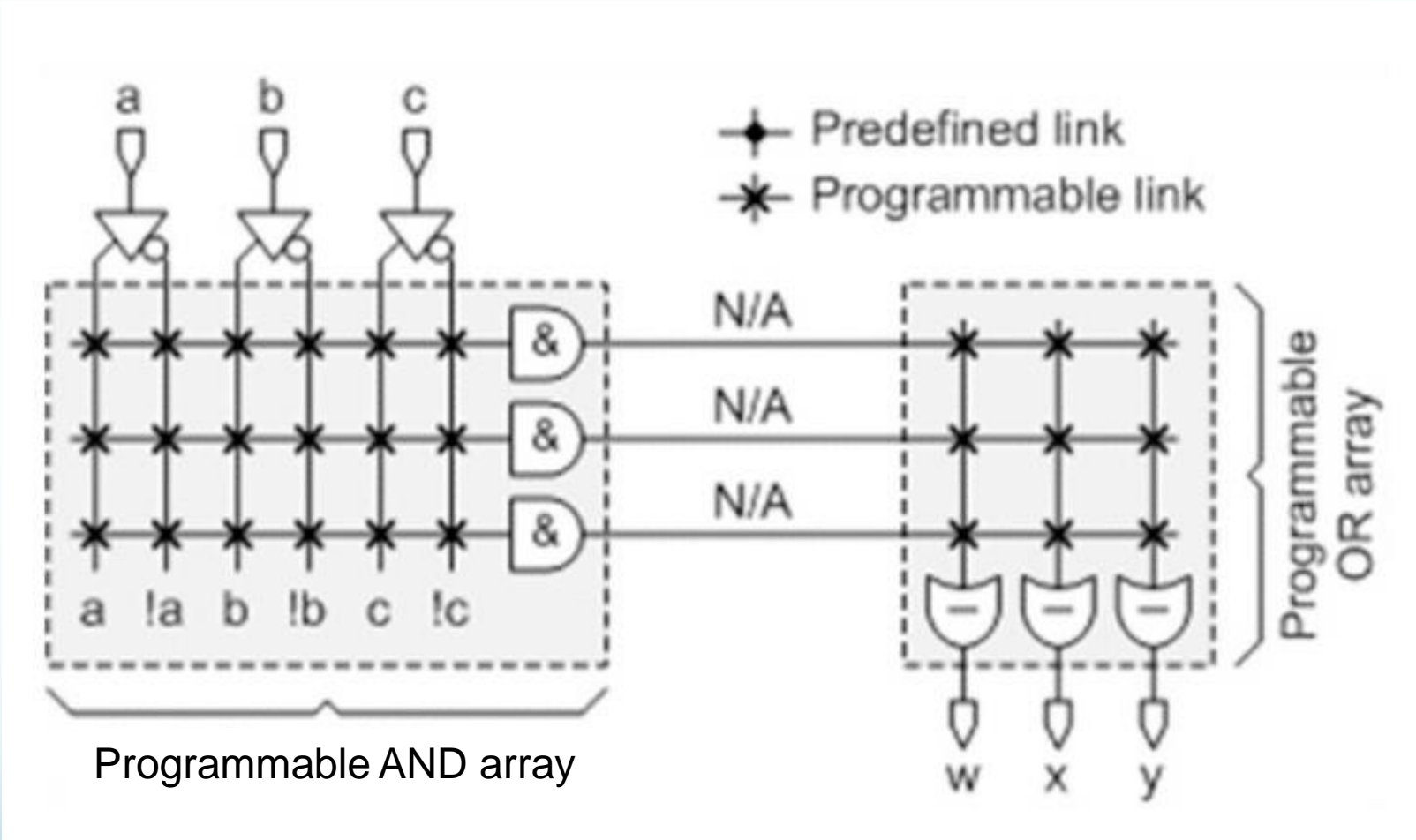


Late 60's

Unprogrammed PROM (Fixed AND Array, Programmable OR Array)

Simple Programmable Logic Devices (sPLDs)

b) Programmable Logic Arrays (PLAs)

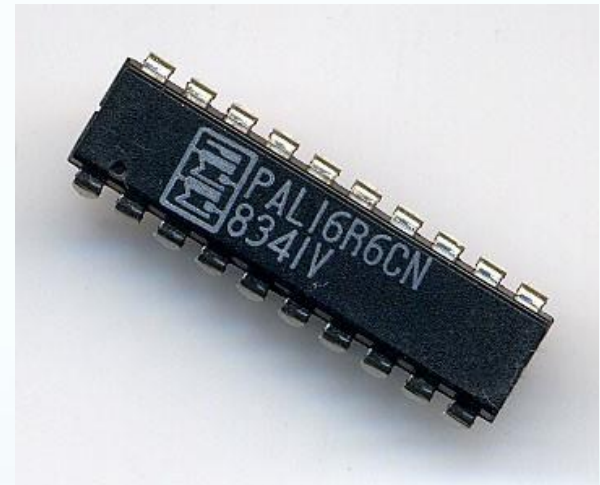
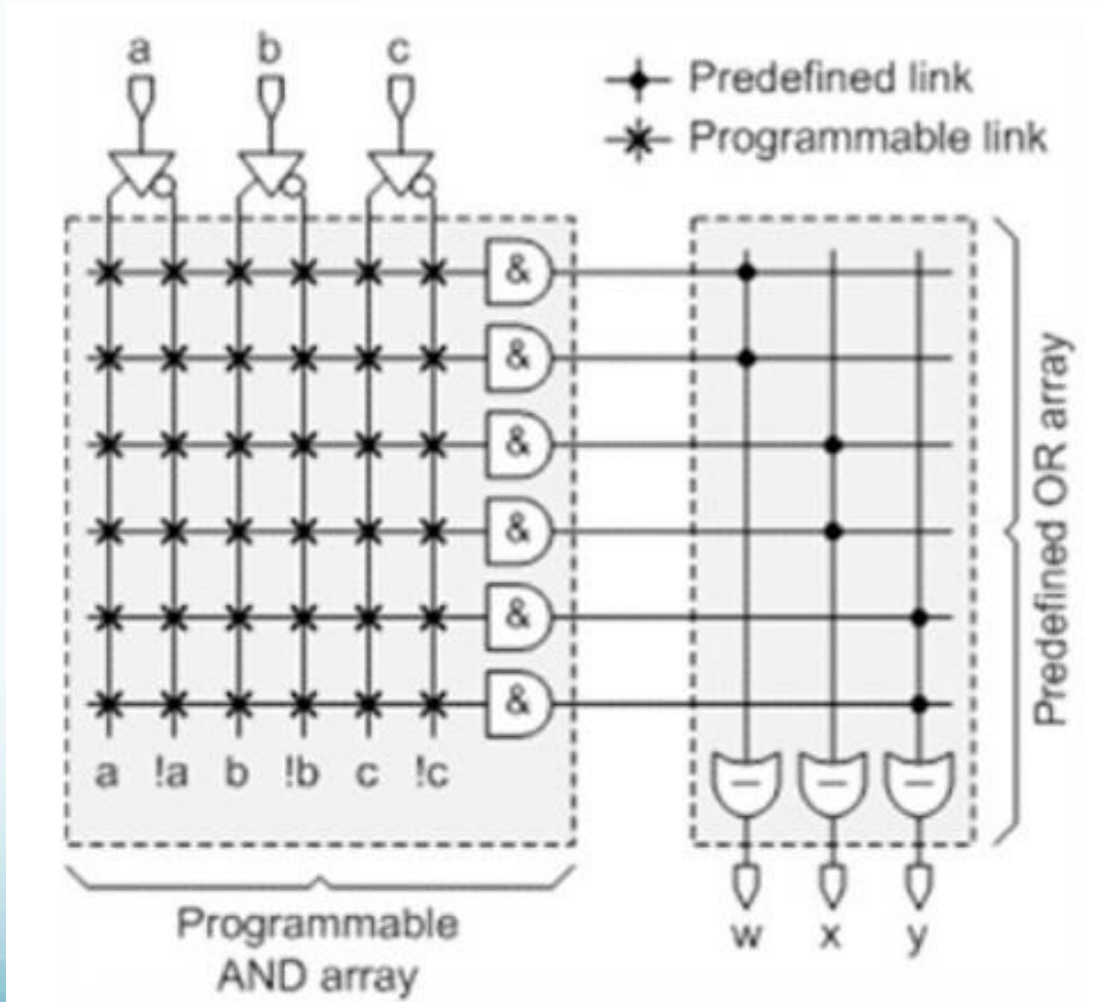


Unprogrammed PLA (Programmable AND and OR Arrays)

MicroLab, NTUA Most flexible but slower

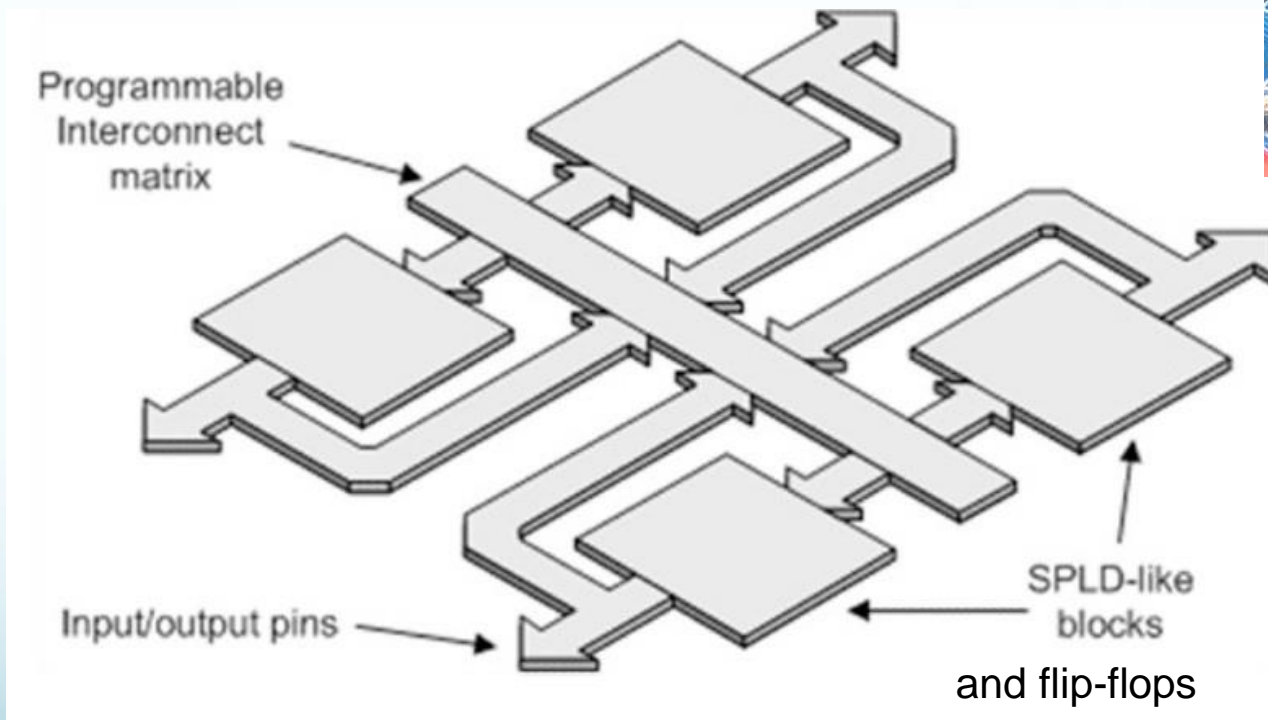
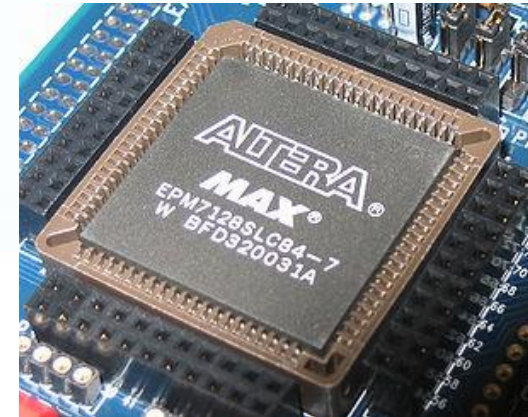
Simple Programmable Logic Devices (sPLDs)

c) Programmable Array Logic (PAL)



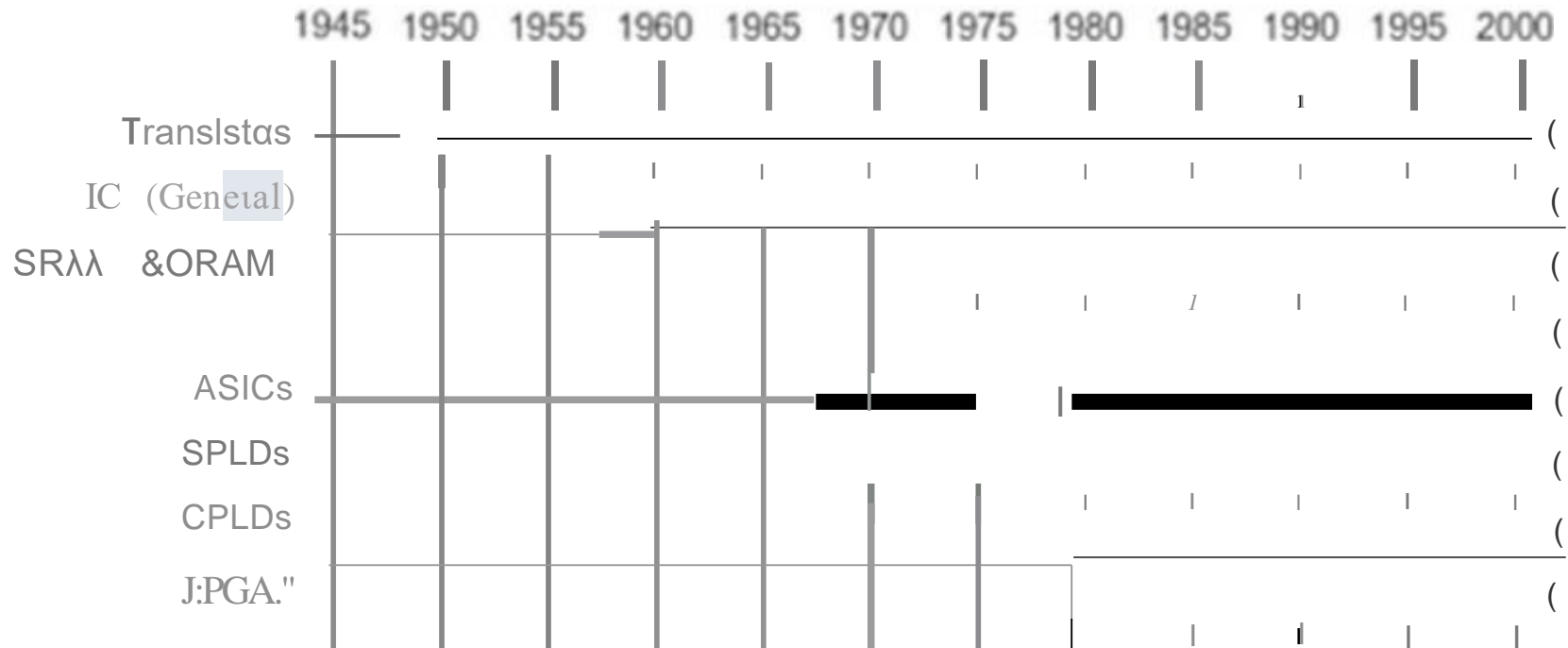
Unprogrammed PAL (Programmable AND Array, Fixed OR Array)

Complex PLDs (CPLDs)

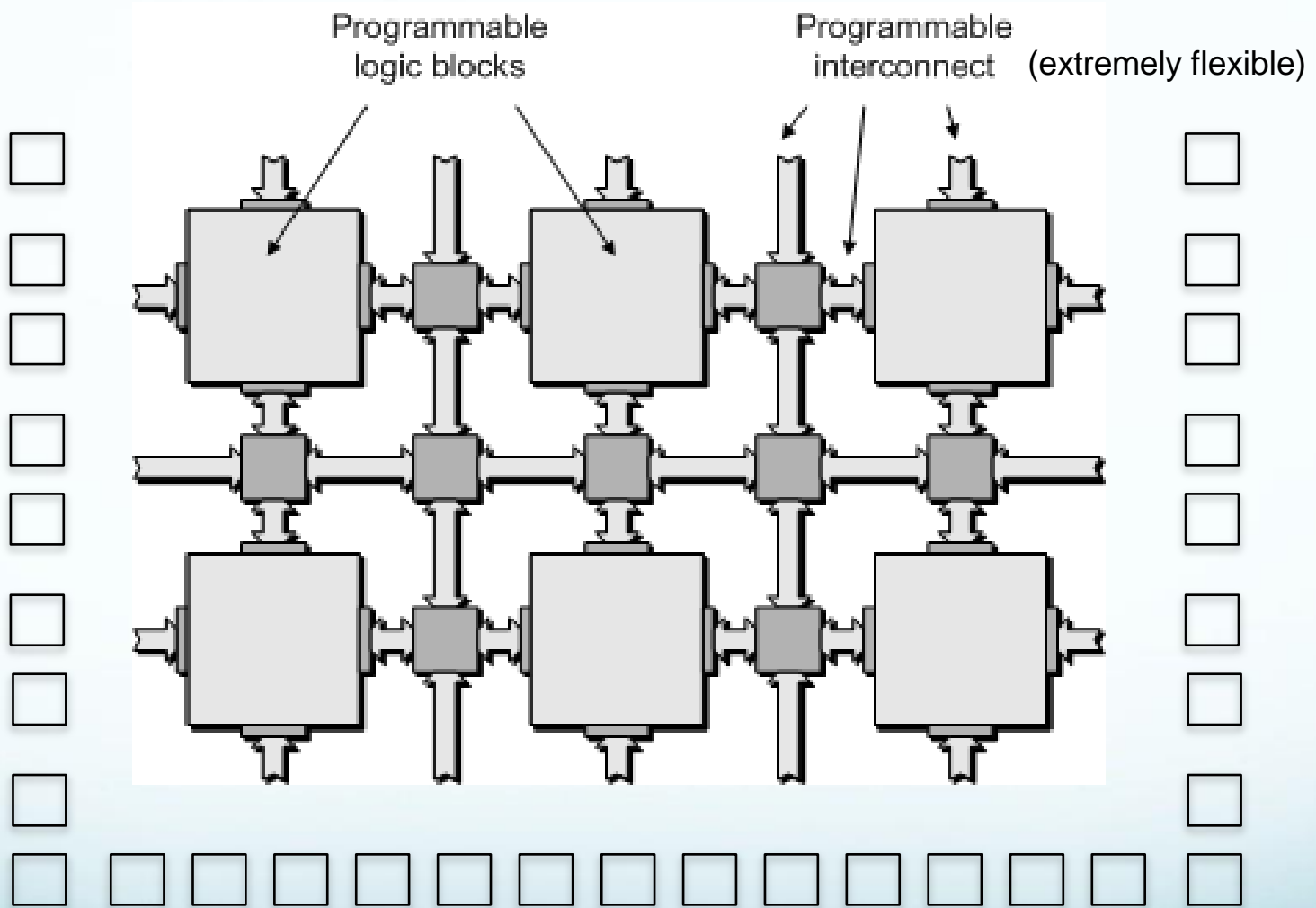


Coarse grained
100's of blocks, restrictive structure
(EE)PROM based

FPGAs ...



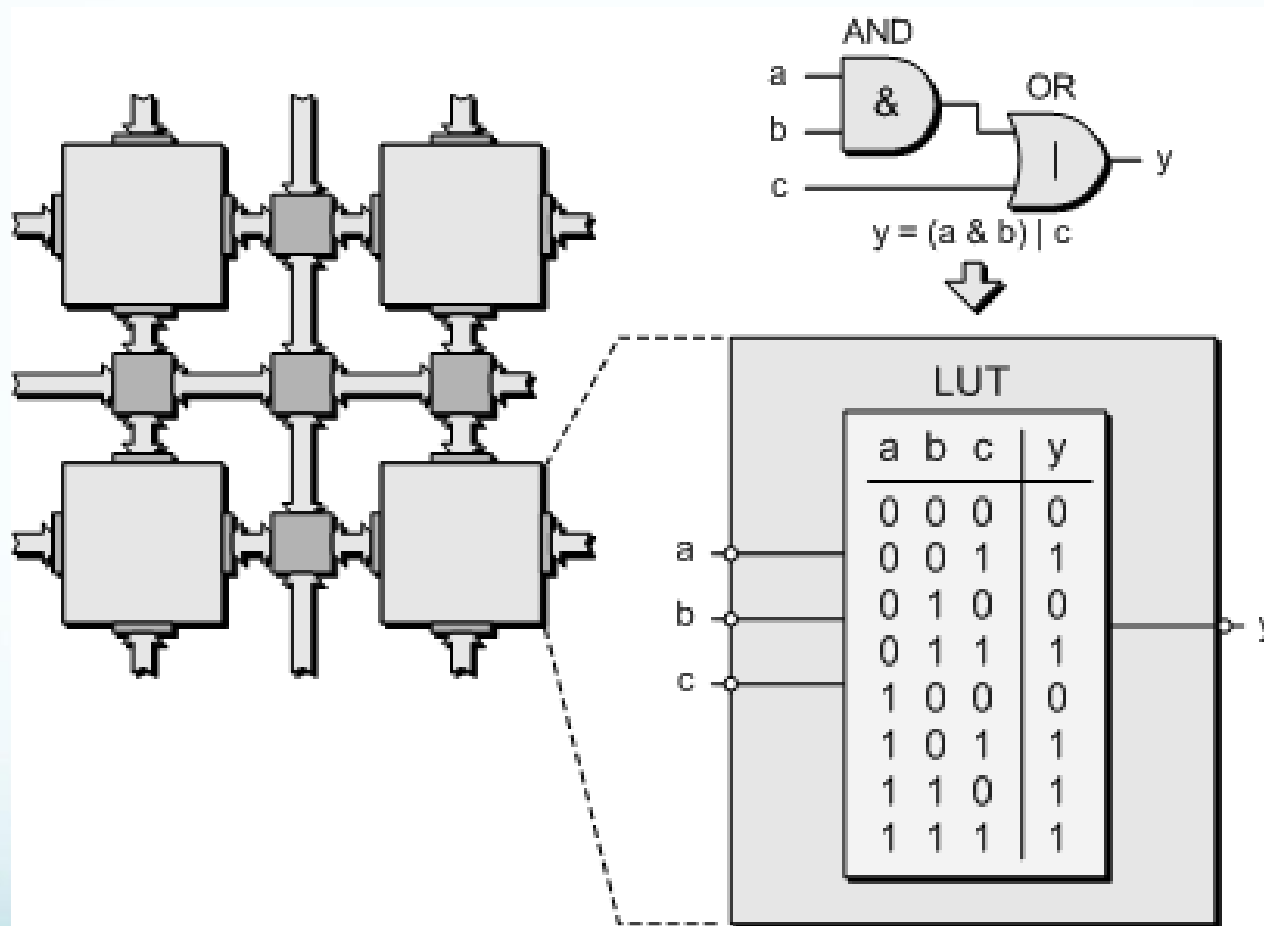
FPGAs



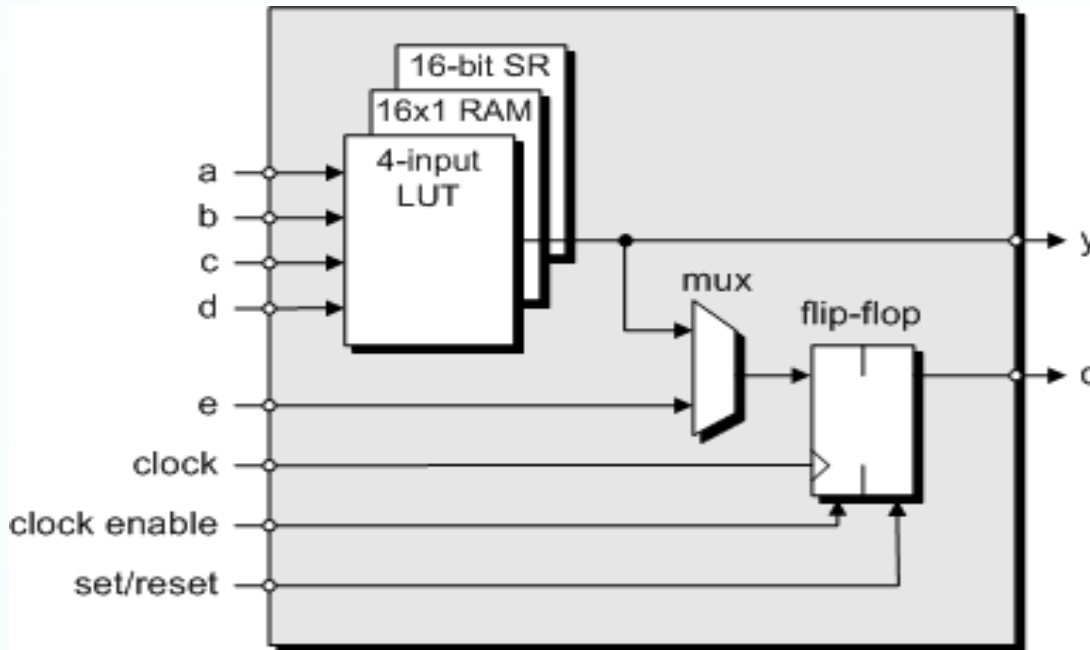
Fine-grained: 100.000's of blocks
today: up to 4 million logic blocks

Programmable Input / Output pins
MicroLab, NTUA

LUT-based Fabrics



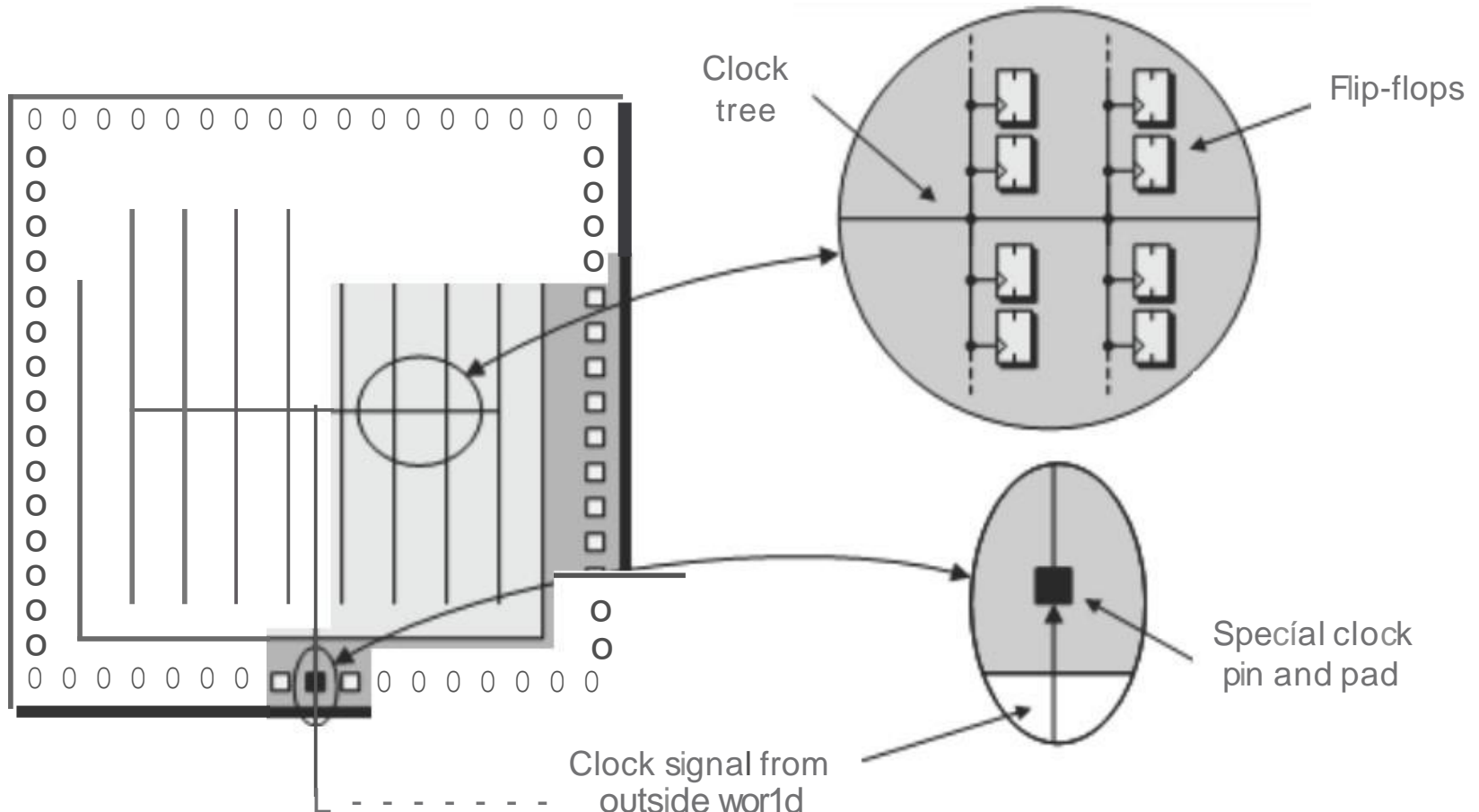
Typical LUT-based Logic Cell



Xilinx: logic cell,
Altera: logic element

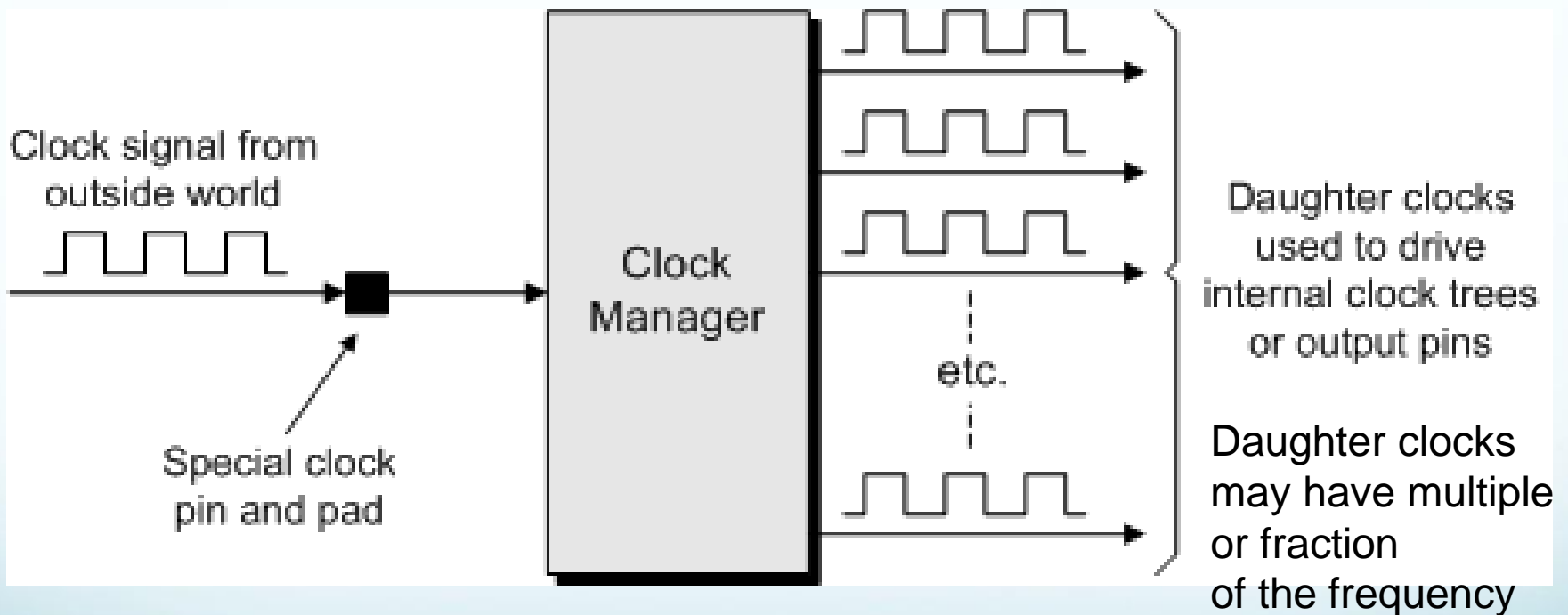
- ◆ LUT may implement any function of the inputs
- ◆ Flip-Flop registers the LUT output
- ◆ May use only the LUT or only the Flip-flop
- ◆ LUT may alternatively be configured a shift register
- ◆ Additional elements (not shown): fast carry logic

Clock Trees

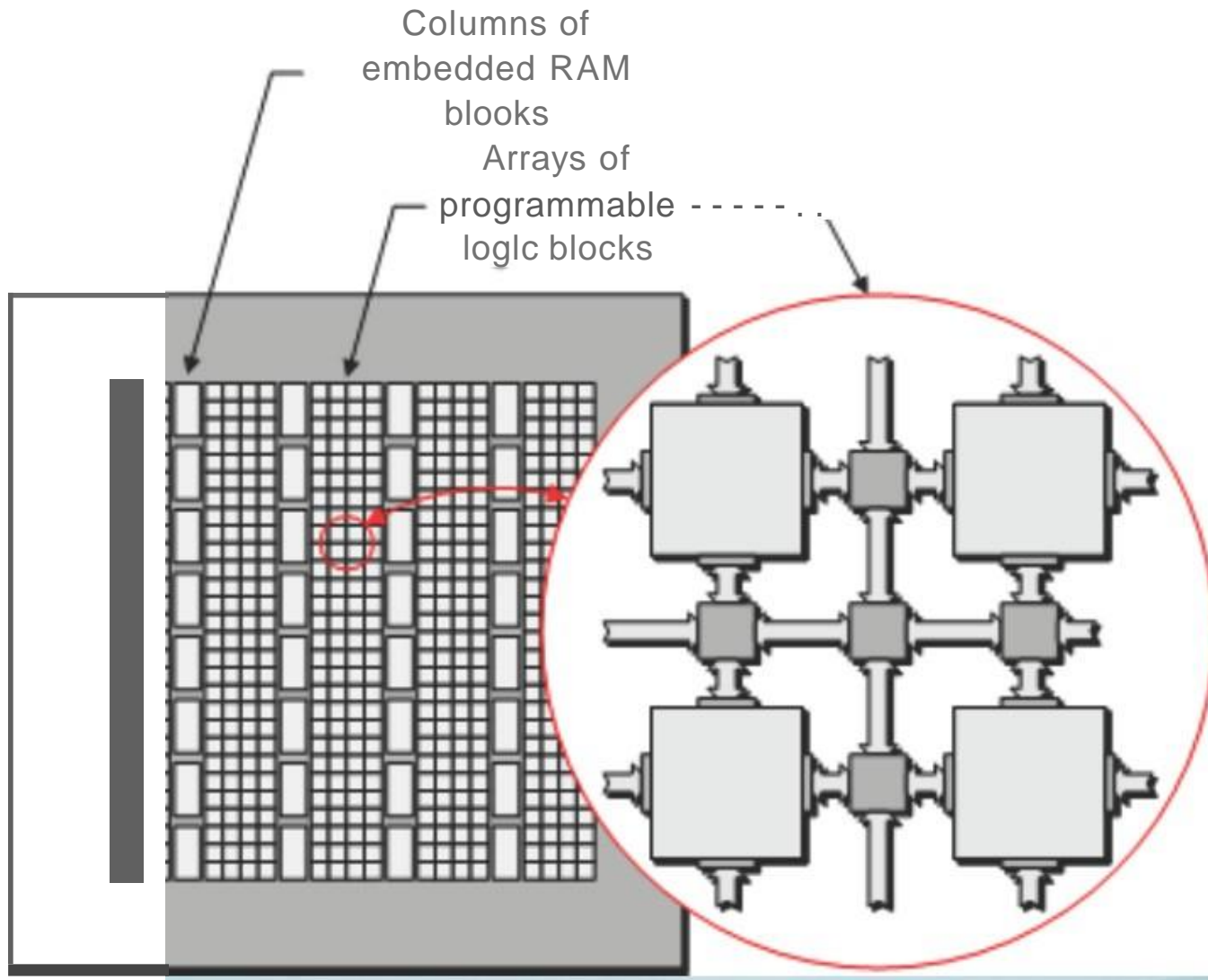


Clock trees guarantee that the clock arrives at the same time at all flip-flops

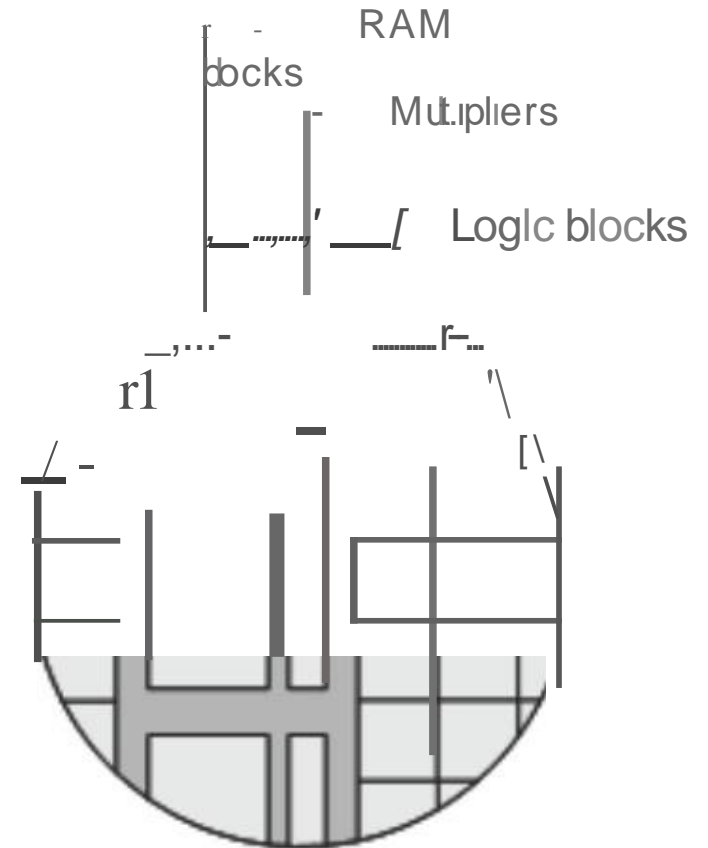
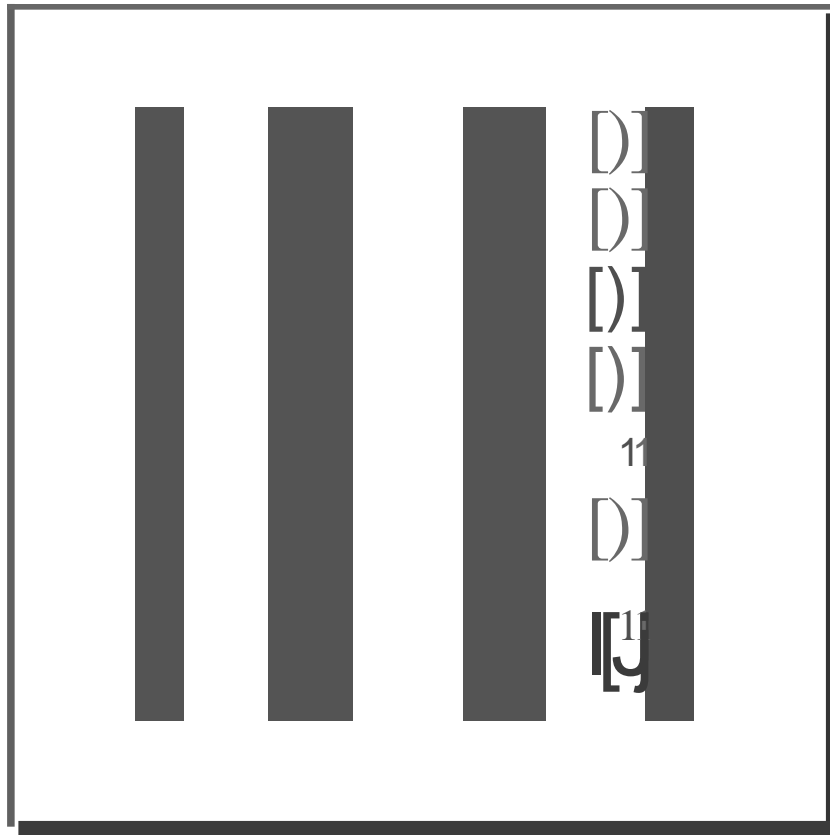
Clock Managers



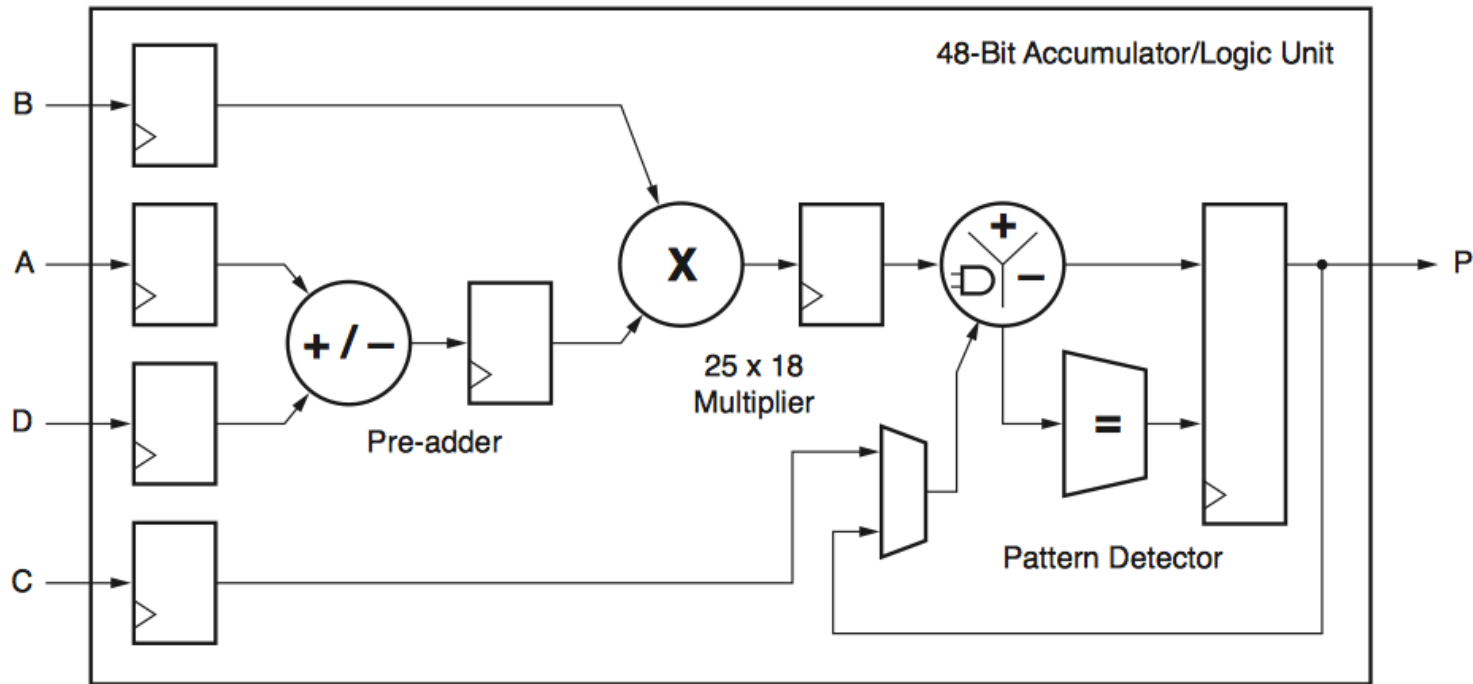
Embedded RAM blocks



Embedded Multipliers & DSPs



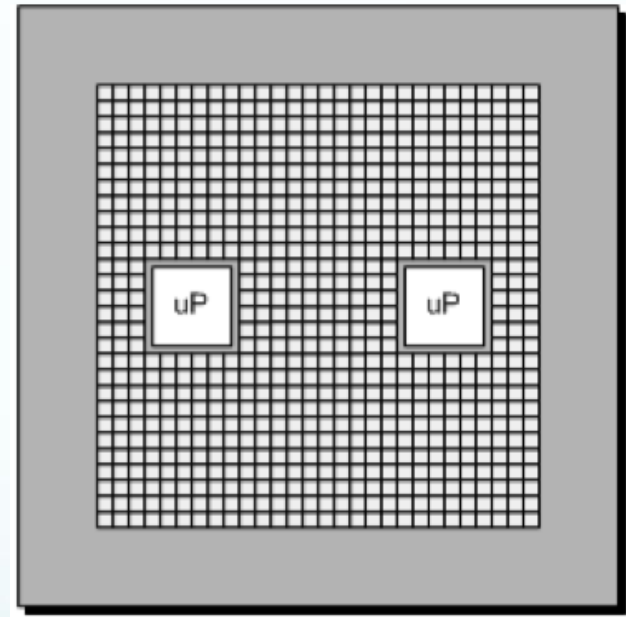
Digital Signal Processor (DSP)



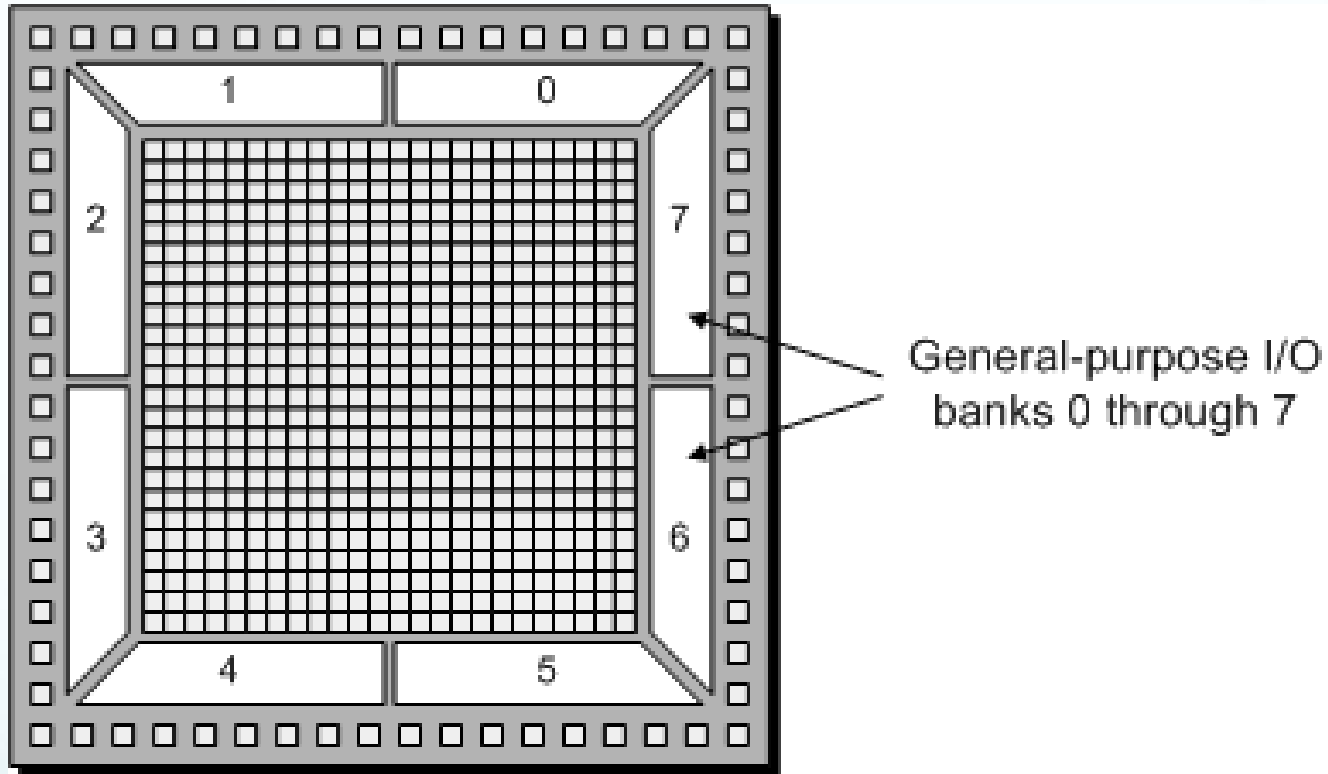
DSP block (Xilinx 7-series)
Up to several 1000 per chip

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

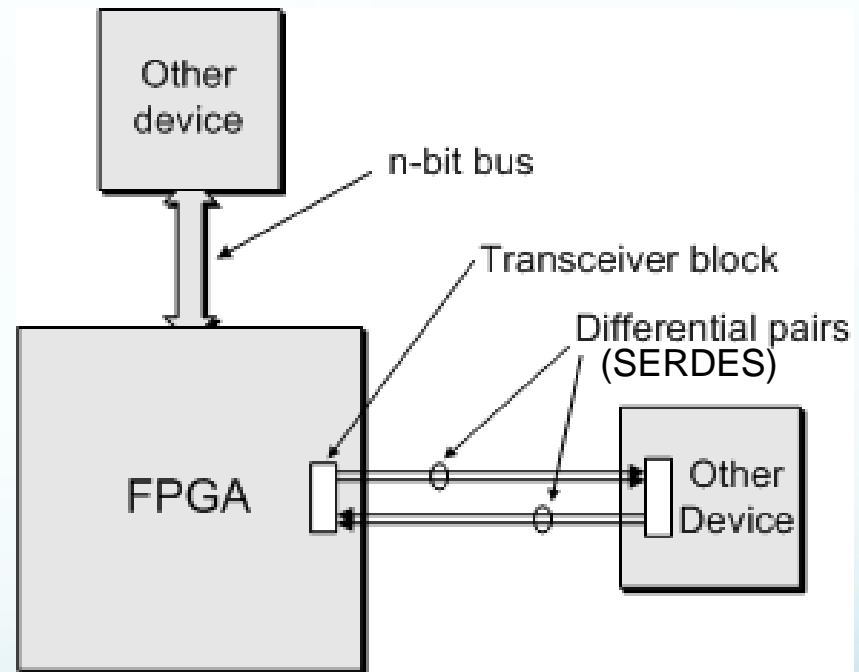
Many IO standards

Single-ended: LVTTTL, LVCMOS, ...

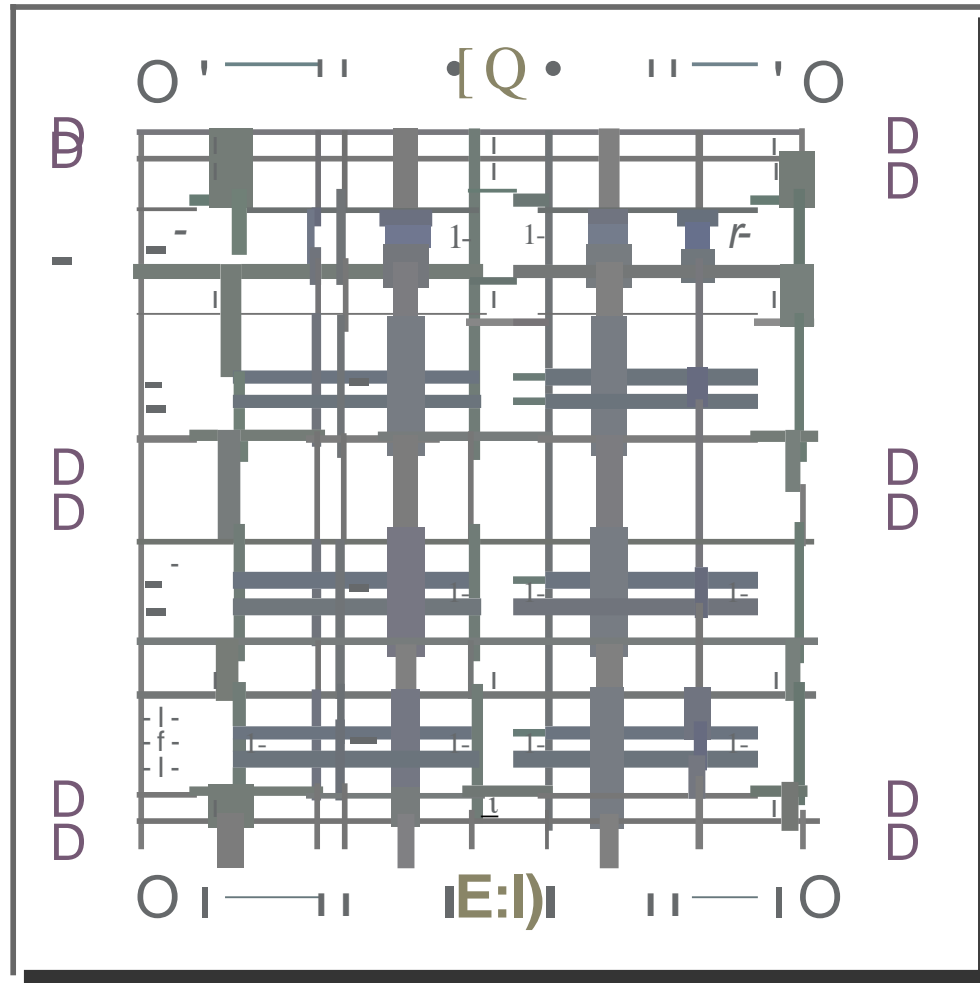
Differential pairs: LVDS, ...

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



- o Logic block
- SRAM block
- o Multiplier
- DSP block
- D PLL
- D Clock Manager
- 10 Bank
- D SERDES block

Programming techniques

Fusible Links (not used in FPGAs)

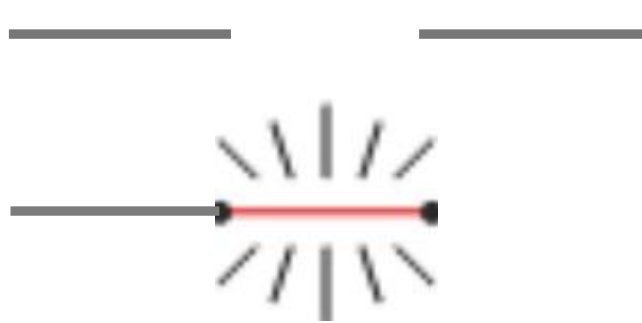


Unprogrammed link



Programmed link

Antifuse Technology

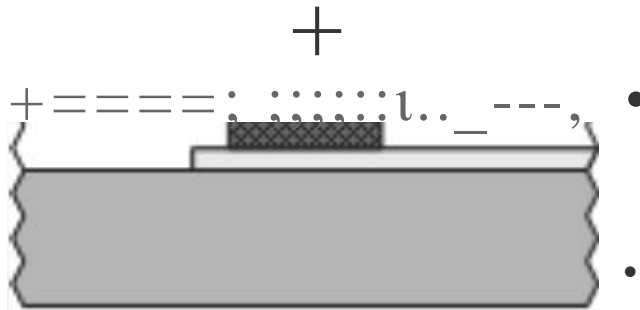


Unprogrammed link

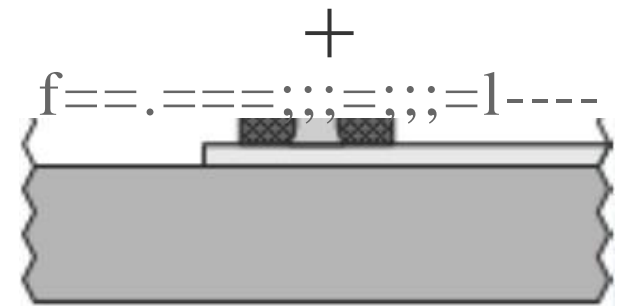
Programmed link

Amorphous silicon column

Polysilicon via



- Metal-Oxide
- Metal
- Substrate

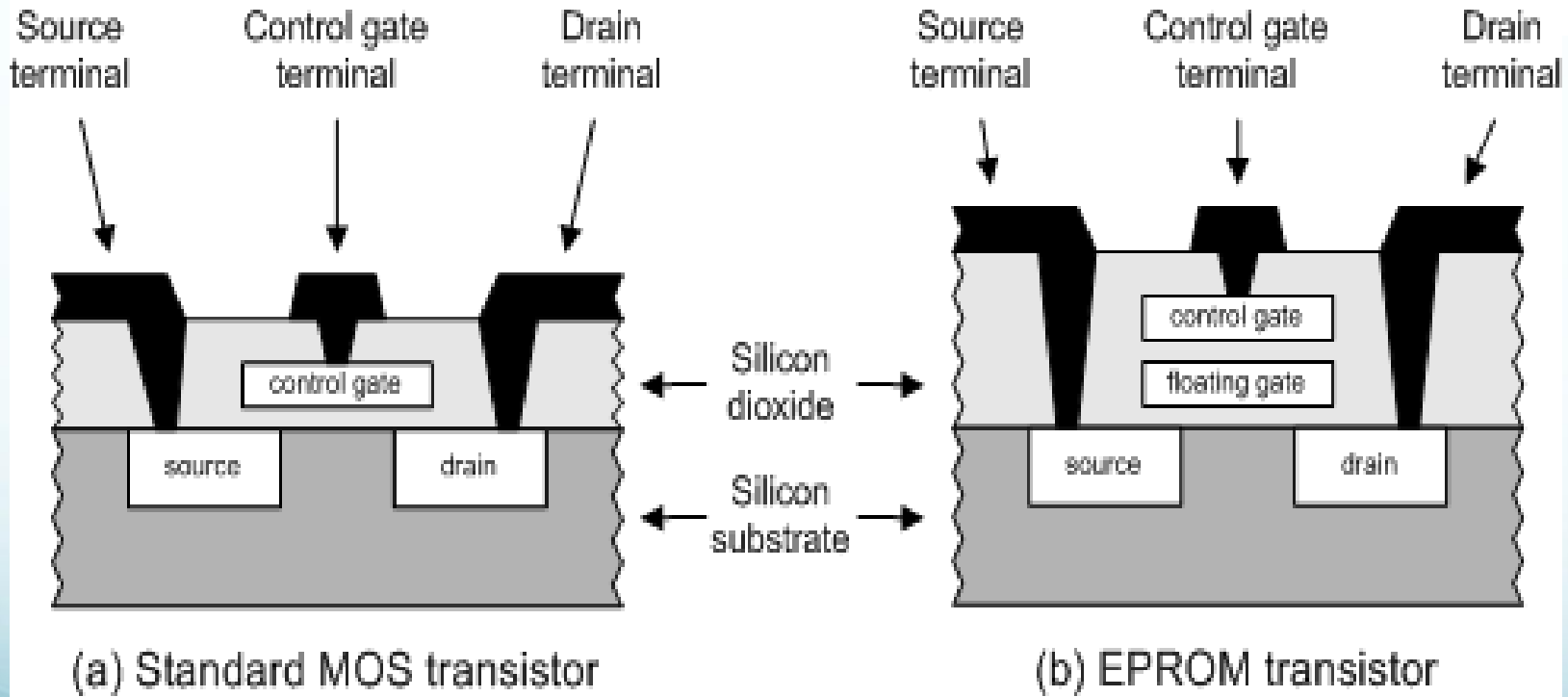
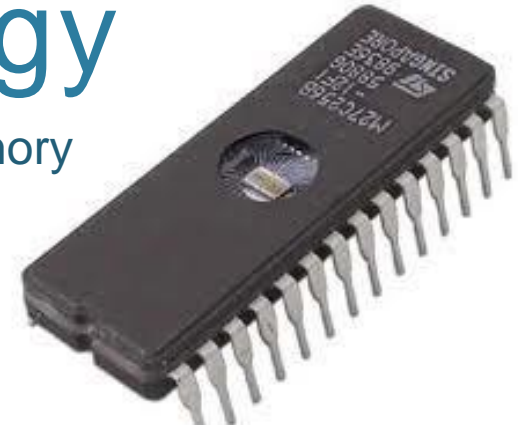


(a) Before programming

(b) After programming

EPROM Technology

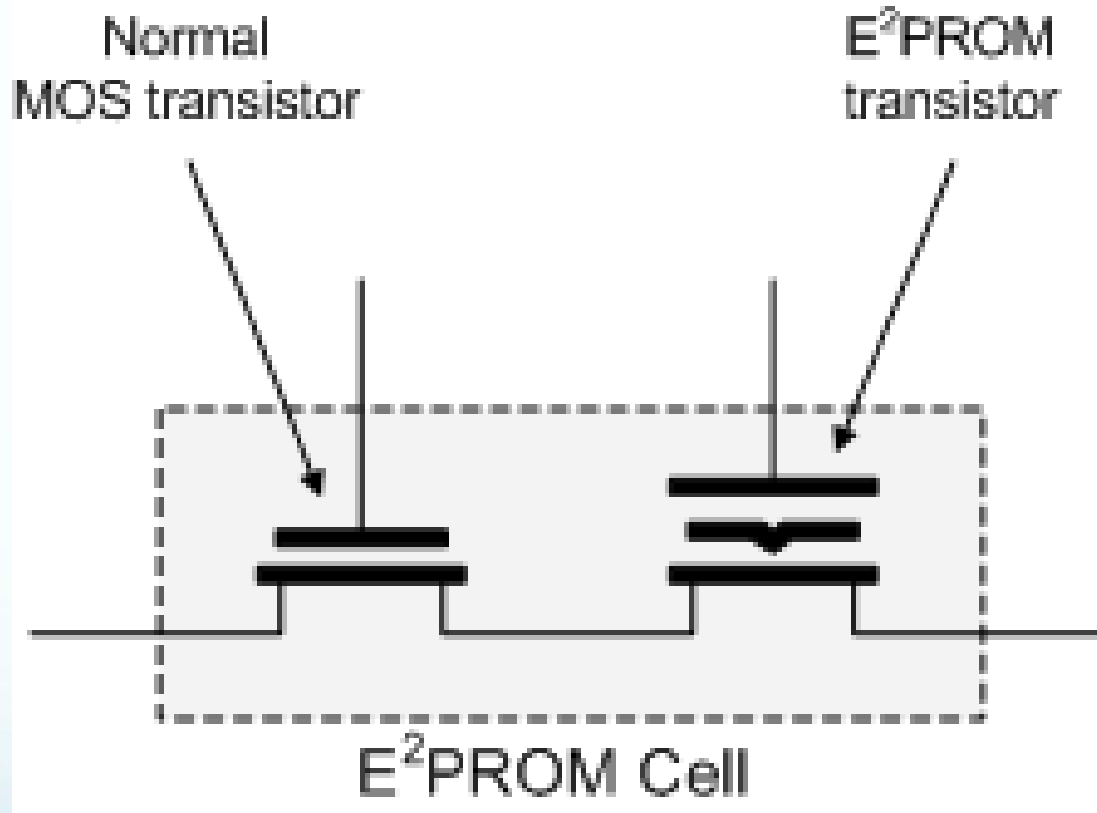
Erasable Programmable Read Only Memory



Intel, 1971

EEPROM and FLASH Technology

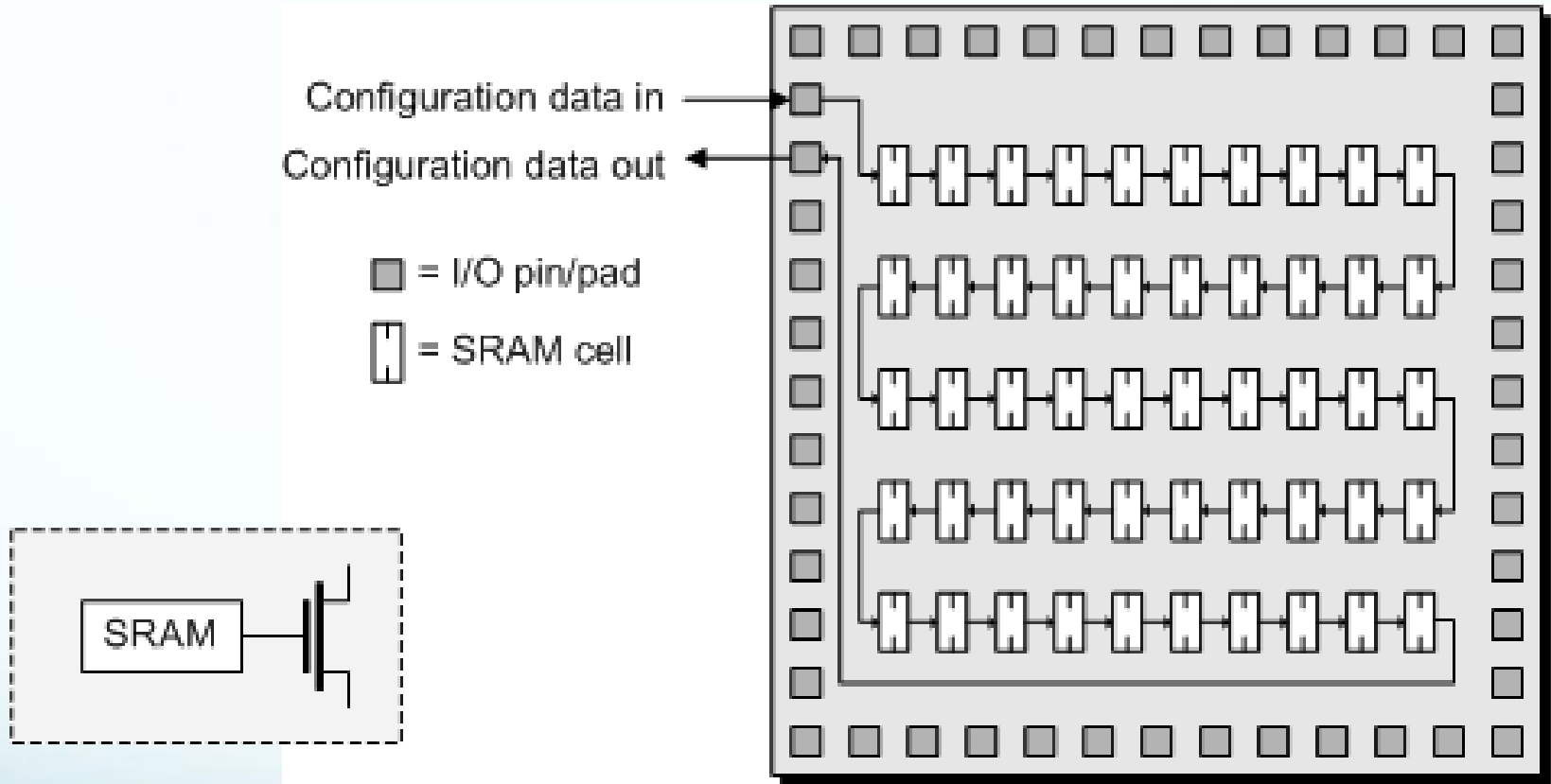
Electrically Erasable Programmable Read Only Memory



EEPROM: erasable word by word

FLASH: erasable by block or by device

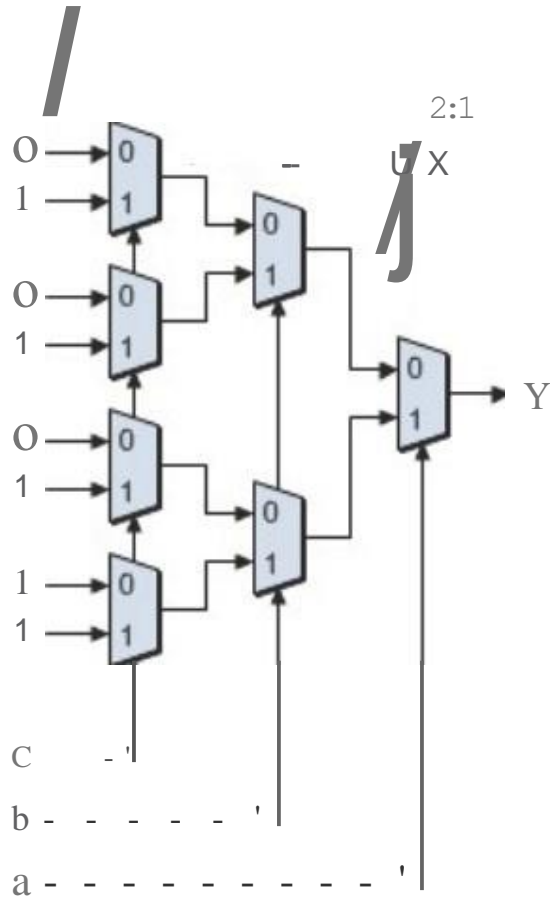
SRAM-Based Devices



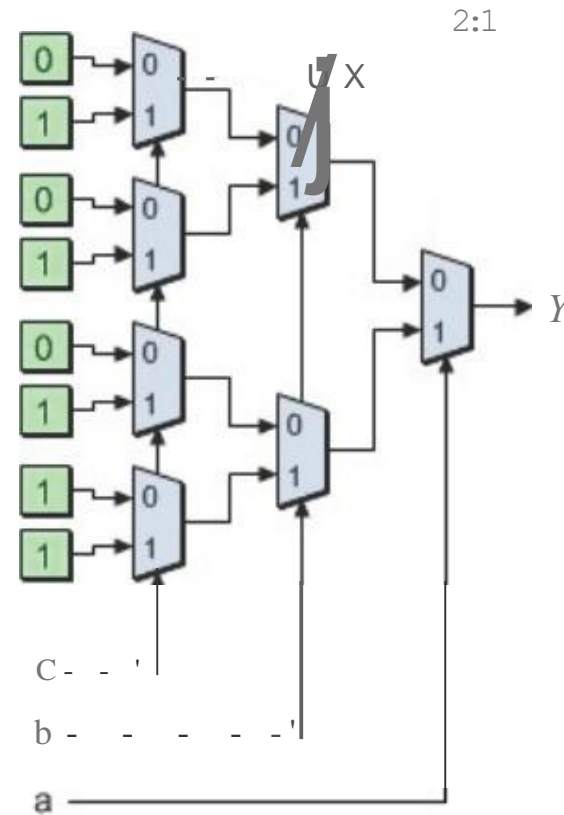
Multi-transistor SRAM cell

Programming a 3-bit wide LUT






Hard-wired 0s and 1s
(Antifuse technology)



SRAM cells
(S; · b a · & d looh, ogy)



Summary of Technologies

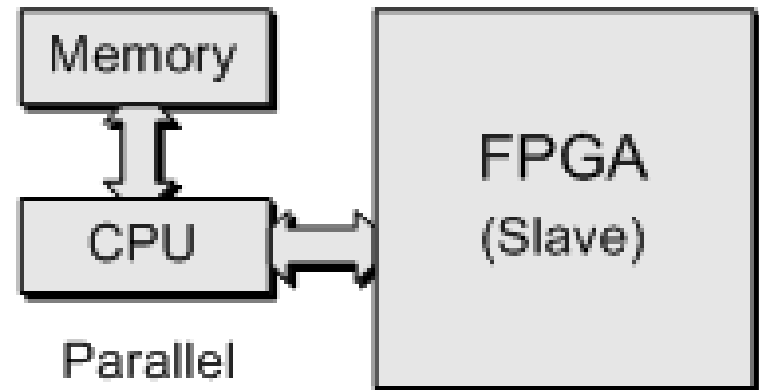
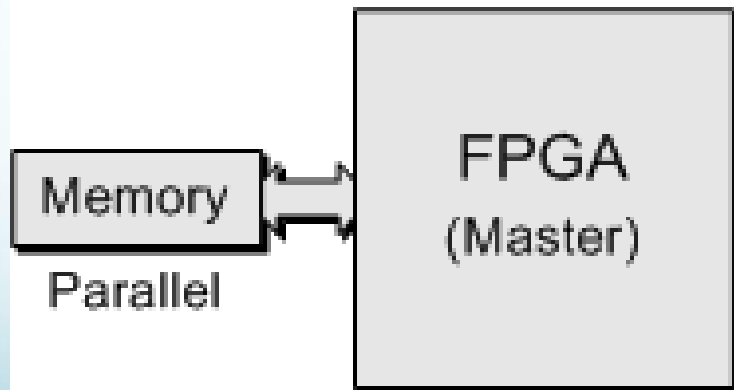
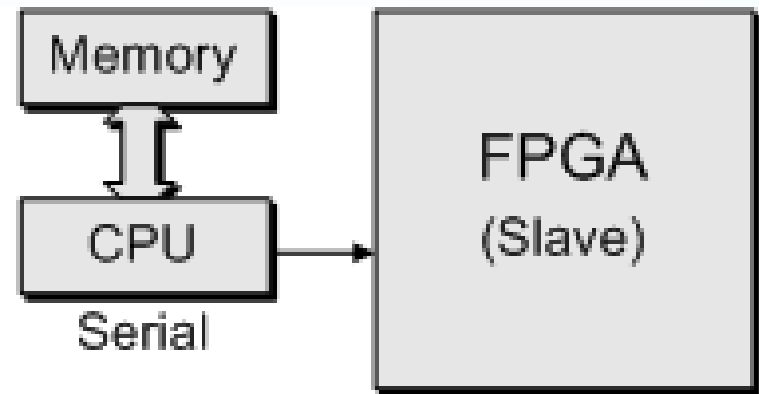
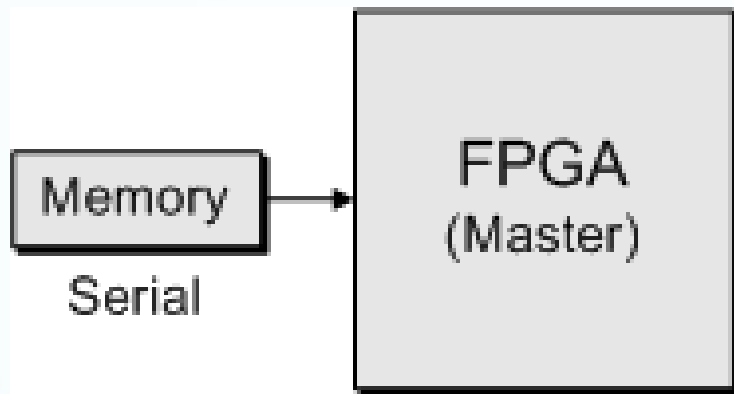
Technology	Symbol	Predominantly associated with ...
Fusible-link		SPLDs
Antifuse		FPGAs
EPROM		SPLDs and CPLDs
E ² PROM/ FLASH		SPLDs, CPLDs, and FPGAs
SRAM		FPGAs (some CPLDs)

← Rad-tolerant secure

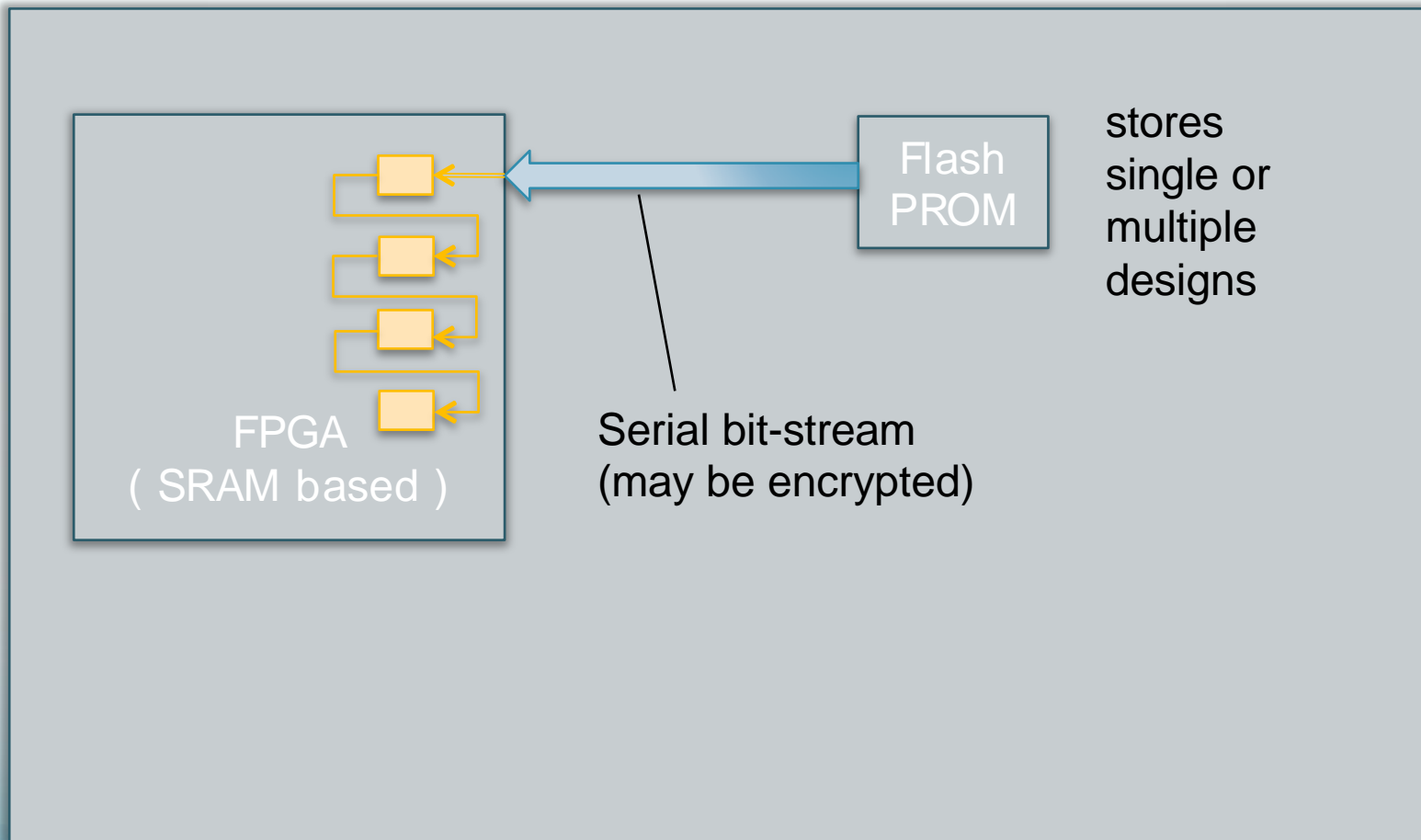
← Rad-tolerant (e.g. Alice)

← Used in most FPGAs

Design Considerations (SRAM Config.)



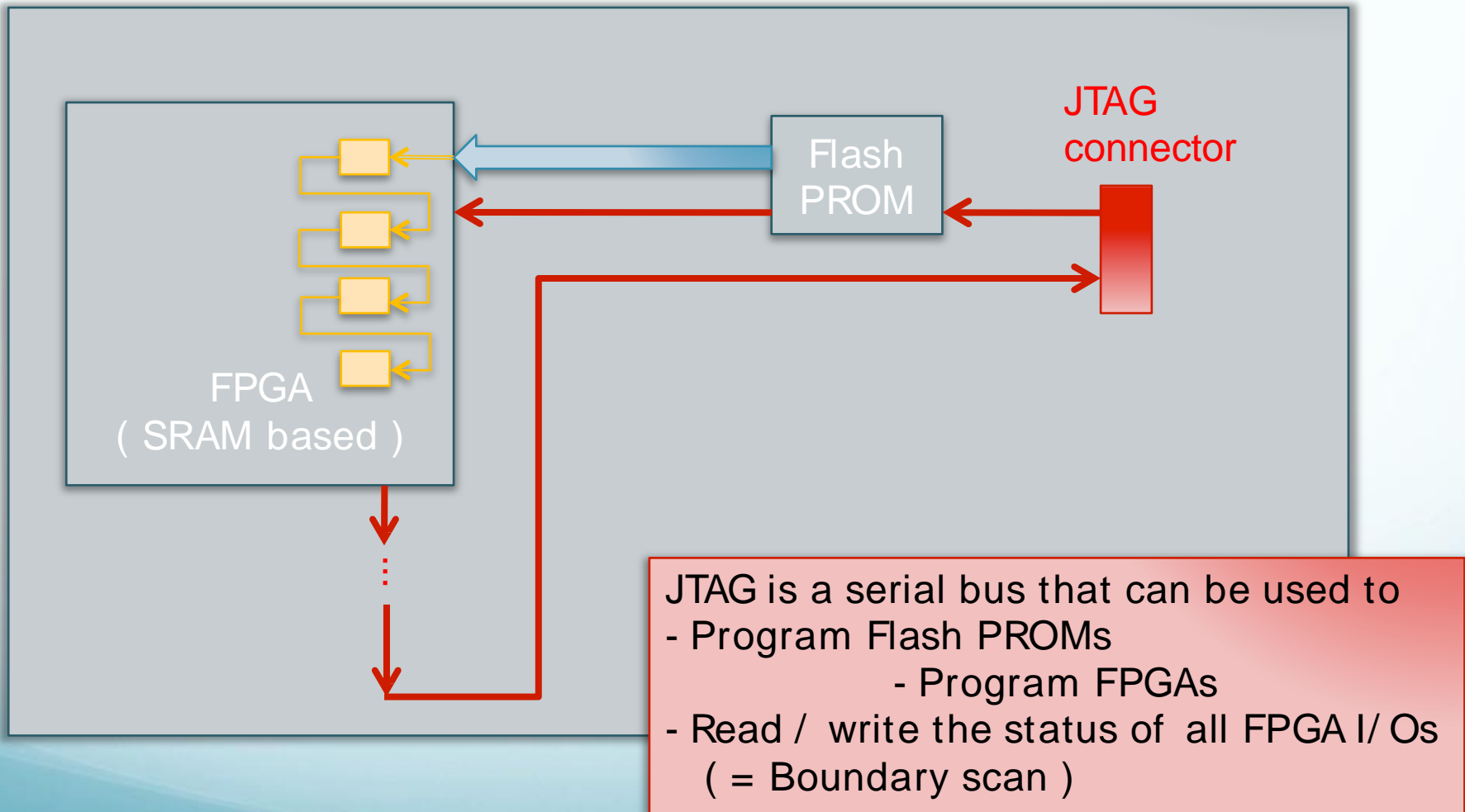
Configuration at power-up



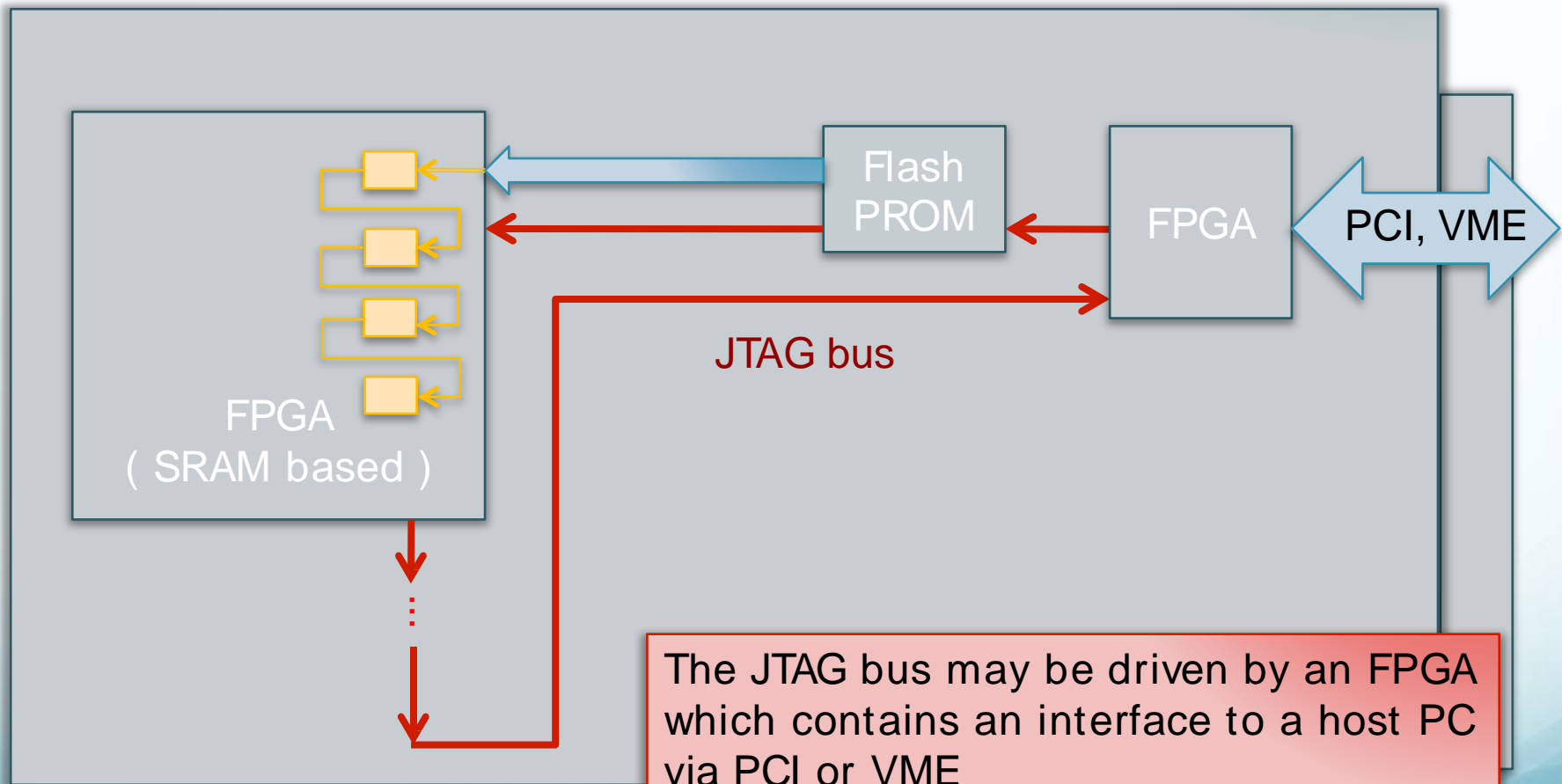
Typical FPGA configuration time: milliseconds

Programming via JTAG

Joint Test Action Group



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 45-50% market share, today
 - ◆ SRAM based CMOS devices



- ◆ Intel FPGA (formerly Altera)
 - ◆ About 40-45% market share
 - ◆ SRAM based CMOS devices



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

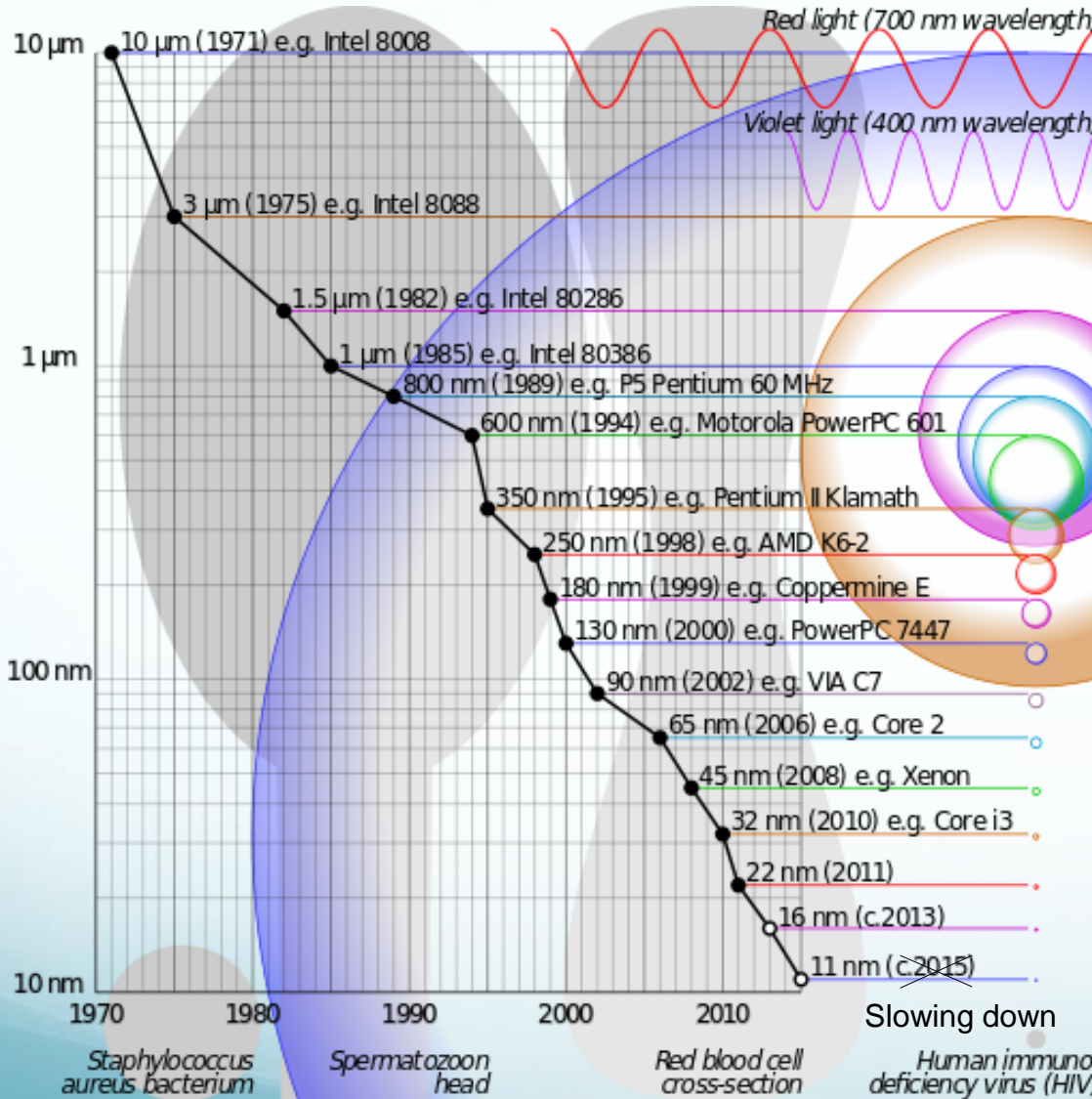


- ◆ Lattice Semiconductor
 - ◆ SRAM based with integrated Flash PROM
 - ◆ low power



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 Altera Stratix 10

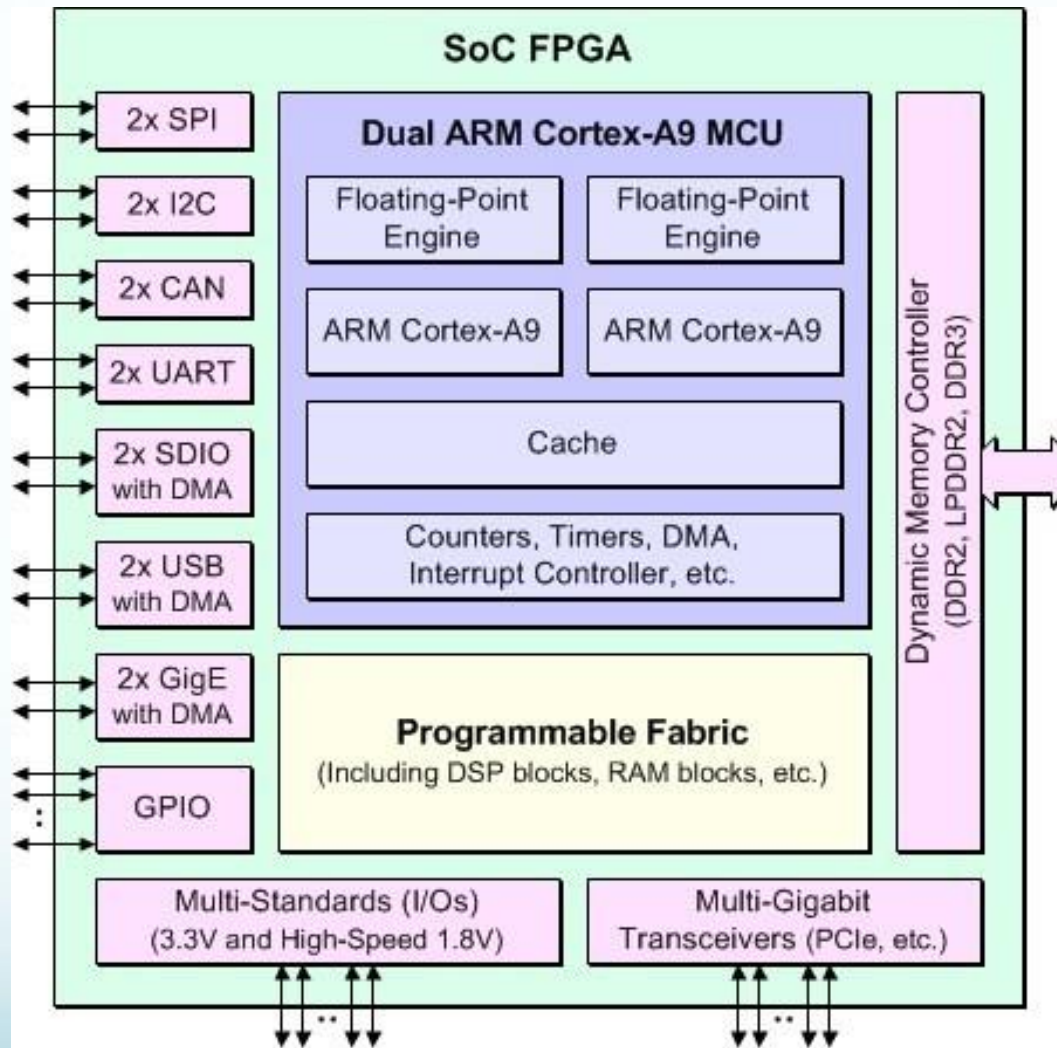
5.5 million logic cells

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)
- 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



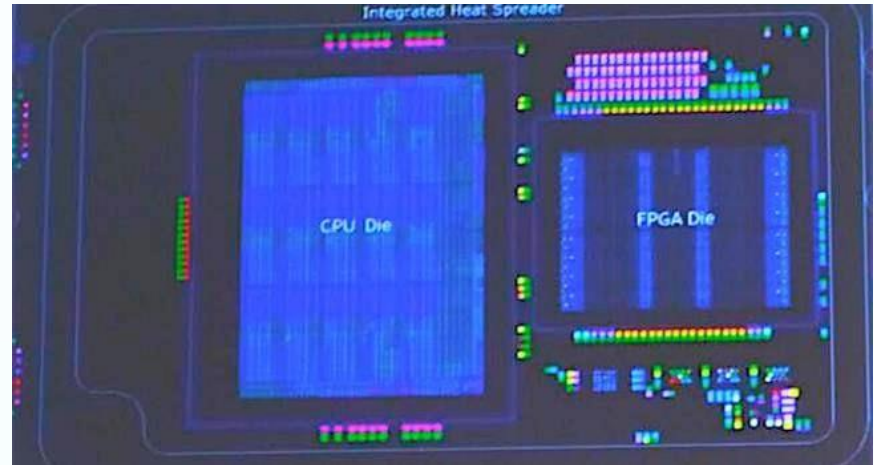
Xilinx Zynq

Altera Stratix 10

CPU(s) + Peripherals + FPGA in one package

FPGAs in Server Processors and the Cloud

- ◆ New in 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

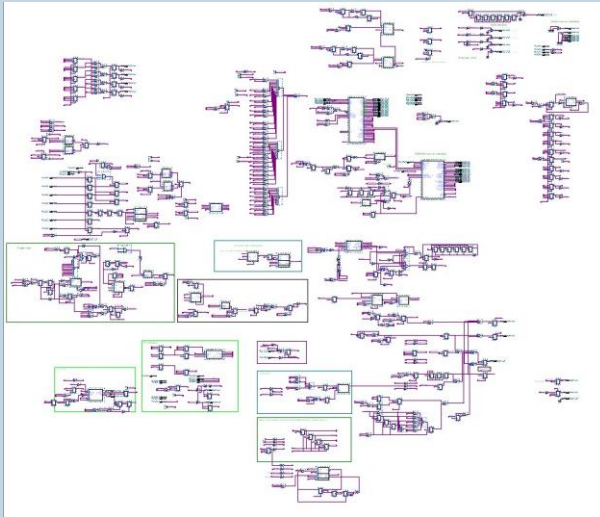
- ◆ Higher performance
- ◆ Analog designs possible
- ◆ Better radiation hardness
- ◆ Long development cycle (weeks / months)
- ◆ Design cannot be changed once it is produced
- ◆ Extremely high development cost
 - ◆ ASICs are produced at a semiconductor fabrication facility (“fab”) according to your design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



FPGA development

Design entry

Schematics



Hardware description language VHDL, Verilog

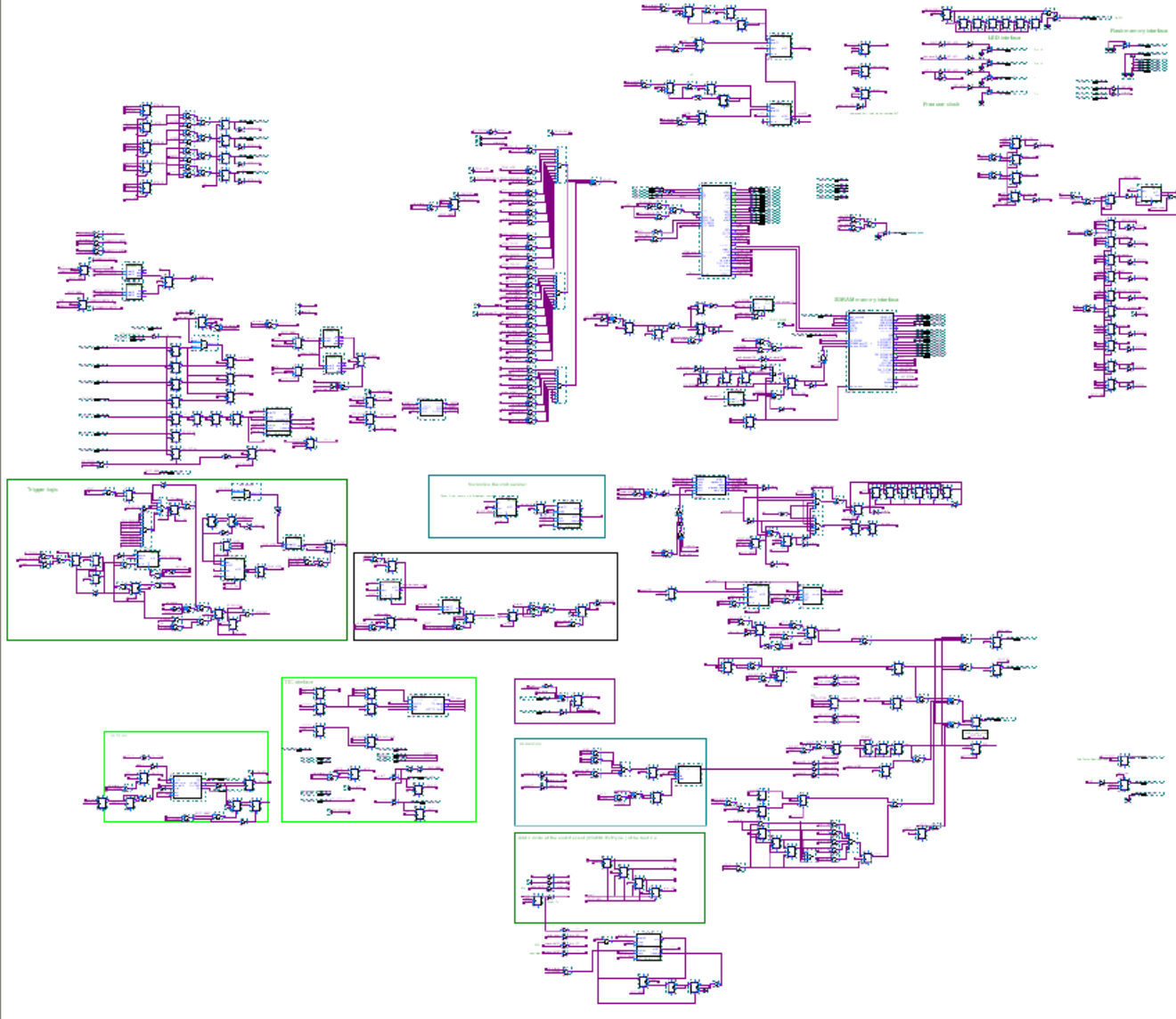
```
entity DelayLine is
    generic (
        n_halfcycles : integer := 2);
    port (
        x          : in std_logic_vector;
        x_delayed  : out std_logic_vector;
        clk        : in std_logic);
end entity DelayLine;
```

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

- ◆ 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;
```

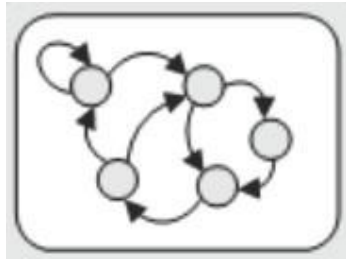
```
data <= Q;
vme_data_out <= Q;
```

end behavioral;

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

Schematics & HDL combined

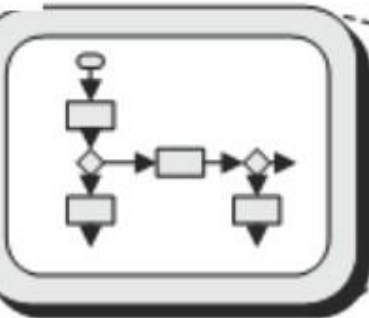
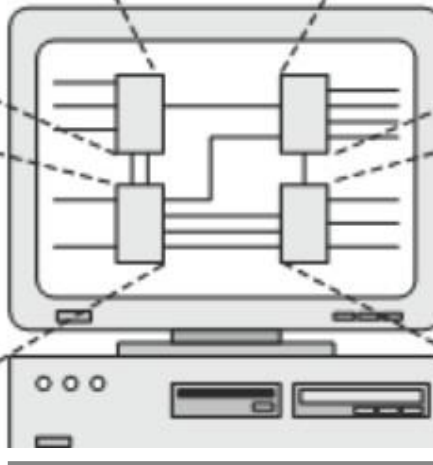
Graphical State Diagram



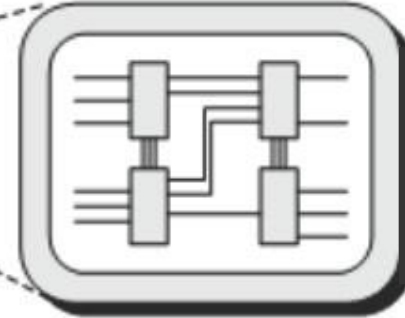
Textual HDL

```
IMicroCircuit nAcs  
η(s •• O)  
th0ny" ta & bl It;  
elsey. c & l(d."
```

Top-level
block-level
schematic

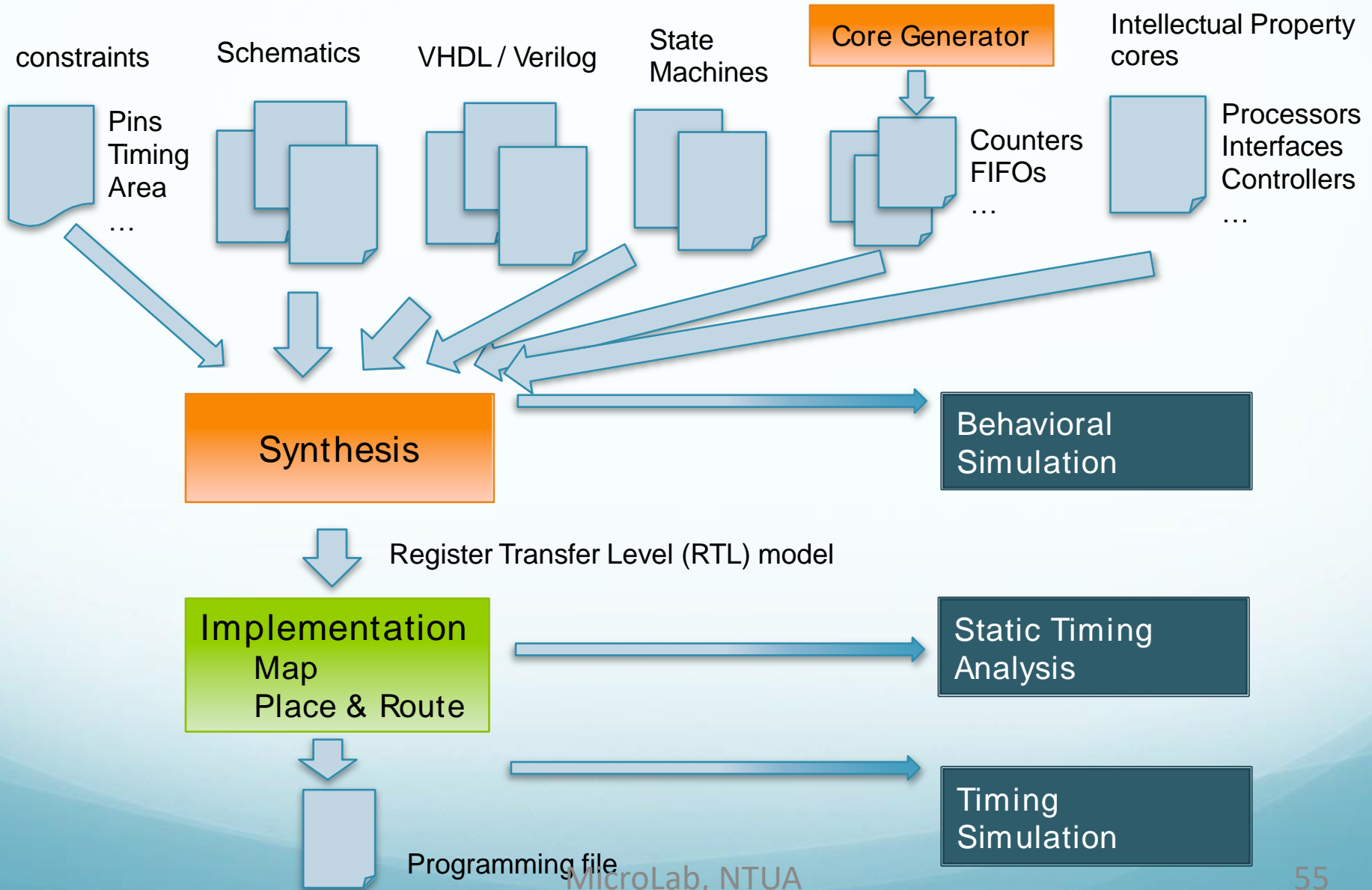


Graphical Flowchart

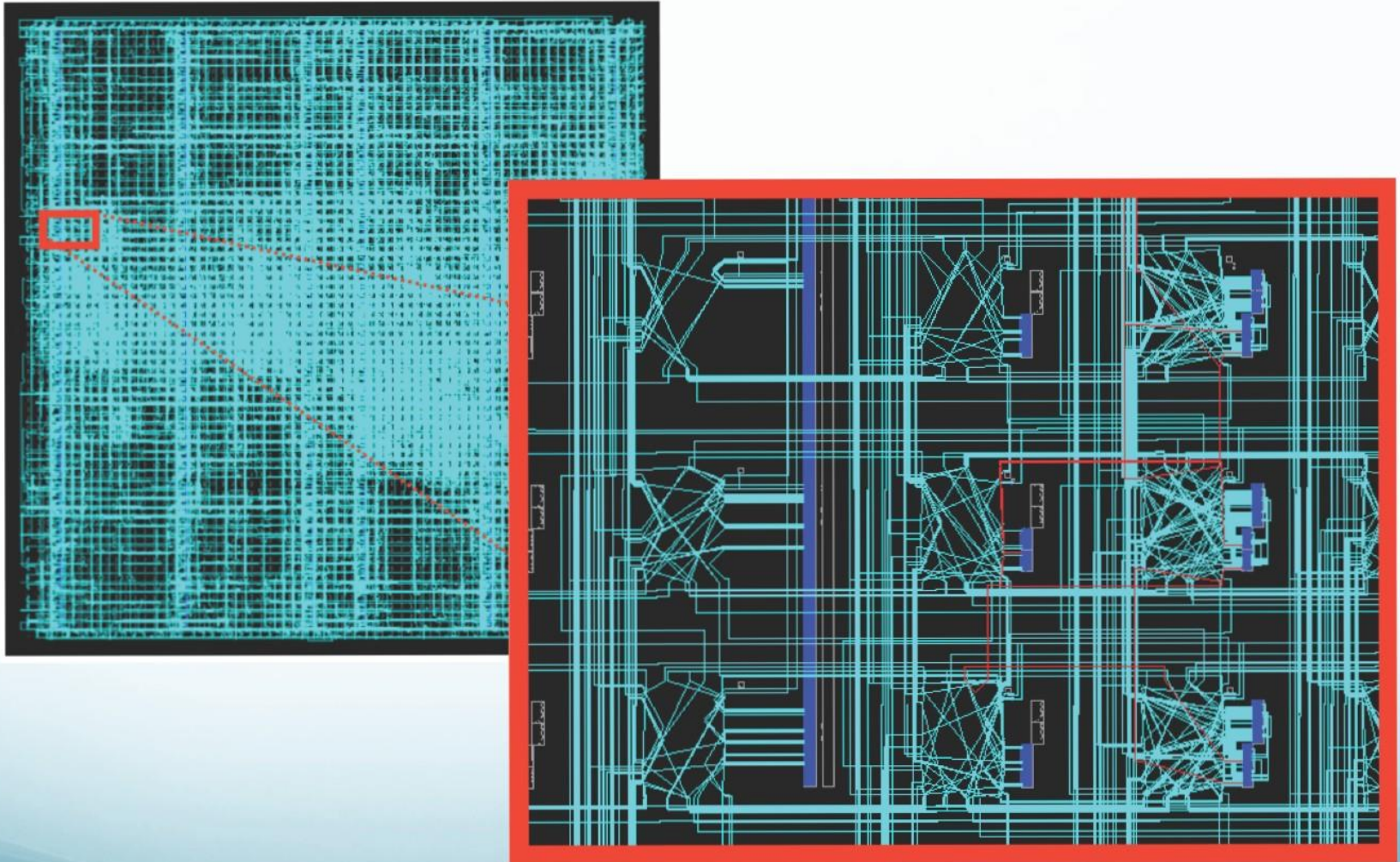


Block-level schematic

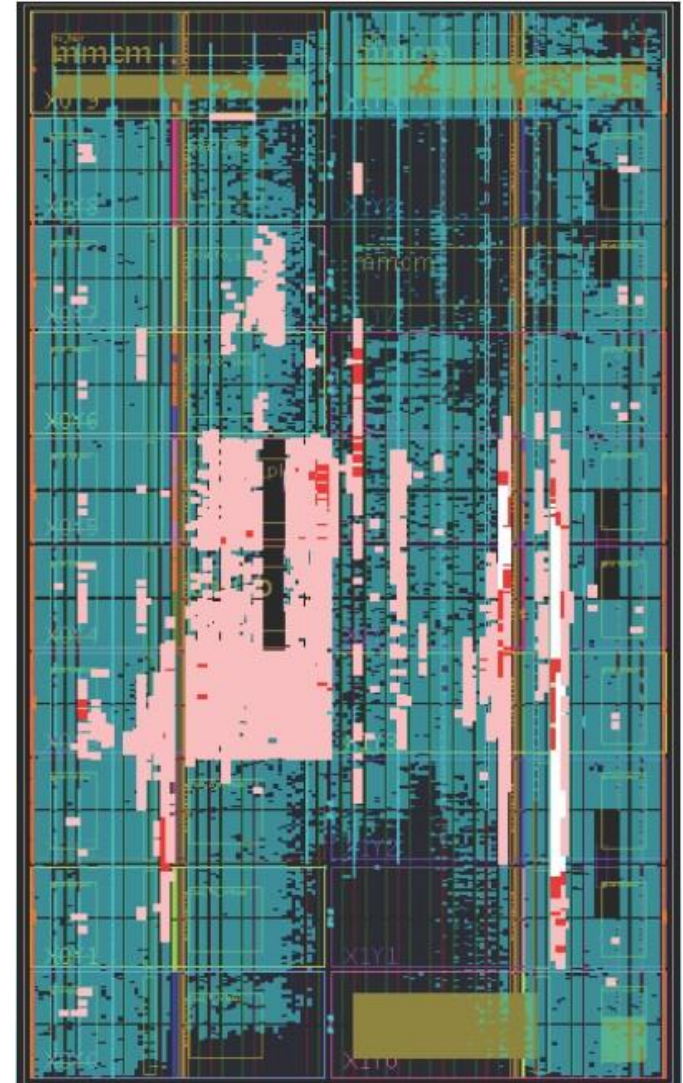
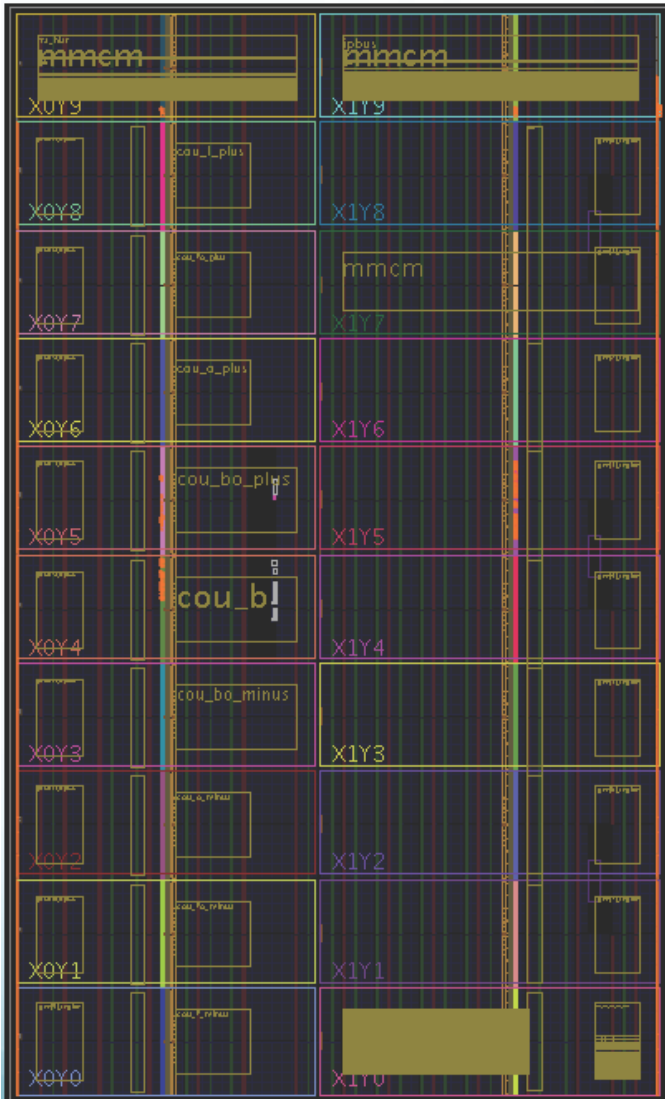
Design flow



Floorplan (Xilinx Virtex 2)



Manual Floor planning



For large designs, manual floor planning may be necessary

MicroLab, NTUA

Routing congestion
Xilinx Virtex 7 (Vivado)

Embedded Logic Analyzers

