Programmierbare Integrierte Schaltungen I

Application Specific Integrated Circuits (ASIC)

Me

Klaus-Peter Kirchner

University of Rostock
Faculty of Electrical Engineering and Information Technology

Contact:
Dr. Klaus-Peter Kirchner, Südstadt H III, Room 30
Phone 0381 498 7210
klaus-peter.kirchner@uni-rostock.de

Skript/Materials (several parts):

Stud-IP -> Course 24175 -> Documents

Campus Südstadt H III, Room S08
Drive N:\PIS_Part1\...

www.igs.uni-rostock.de
-> Intranet (ITMZ-account) -> Skripts

Roadmap

Overview

Simple PLD/CPLD: Structure, "Programming", Configuration
Indroduction into a description language (old: ABEL, new: Verilog)
2 practical exercises (GAL, CPLD)
FPGA-structure, simple and complex structures (from logic-cell to ARM-core)
FPGA-development-tools (Design-entry, simulation, synthesis, placement/routing)
Special design-related chapters
FPGA-design example using schematic entry (practical exercise))

VHDL as a hardware description language in detail
Comparison VHDL, Verilog
FPGA-design using VHDL and special FPGA-structures
Simulation (Behavioural and Timing) , Synthesis ….
Practical exercise: VGA/HDMI-control using VHDL (different levels of difficulty)

Books

Christian Siemers**: Logikbausteine**, ISBN 3-8023-1873-0,
Vogel Buchverlag Würzburg

Frank Kesel, Ruben Bartholomä: **Entwurf von digitalen Schaltungen mit HDLs und FPGAs,** ISBN 978-3-486-58976-4,
Oldenburg Verlag, München

Peter. J. Ashenden: Digital Design – an Embedded Systems Approach Using VERILOG, ISBN: 978-0-12-369527-7,
Norgan Kaufman Pblishers, 2008

Seiichi Aritome: **Nand Flash Memory Technologies,** ISBN: 9781119132608 (Print)
Wiley Online Library: http://onlinelibrary.wiley.com/book/10.1002/9781119132639

Joe E. Brewer, Manzur Gill: **Nonvolatile Memory Technologies With Emphasis On Flash**, ISBN 978-0471-77002-2 (Print)
http://http://ieeexplore.ieee.org/xpl/bkabstractplus.jsp?reload=true&bkn=5201541

Eduardo Bezerra, Djones Lettnin: **Synthesizable VHDL-Design for FPGAs**, Springer, ISBN: 978-3-319-02546-9 (Print)
http://link.springer.com/book/10.1007%2F978-3-319-02547-6

Andreas Heppner: **Das isp-Buch**, ISBN 3-89567-119-2
Elektor-Verlag Aachen

Peter. J. Ashenden:, **The Designer's Guide to VHDL (Third Edition**), ISBN: 978-0-12-088785-9, Elsevier, 2008
http://www.sciencedirect.com/science/book/9780120887859

To understand structures and usage of FPGAs look for documentations, whitebooks etc. at www.xilinx.com, www.altera.com, www.microsemi.com, www.latticesemi.com

Exam: A 20 minutes oral exam including the defense of the projects elaborated during the course

Questions to prepare for the Oral Exam PIS 2017
(It's always one of the first questions asked by student: What will be asked in exam??)

- Explain and compare structure and complexity of programmable logic ICs and ASICs!
- Explain how the structure of PLD maps the structure of a state machine.
- Explain the design flow for CPLD and FPGA!
- Compare FPGA, Gate arrays, Standard-Cell-ICs and Full-Custom-ICs relating to structure, complexity and personalization/programming!
- Compare the cost factors for the development of ICs (from PLD to Custom-IC)!
- Illustrate the structure of a GAL (sketch)!
- Illustrate the structure of FPGA (Keywords: Functional blocks, LUT, wiring, connections, switch matrix, I/O, fabric, distributed and block-RAM, special blocks for signal processing …)! You should be able to create and discuss a sketch.
- What are the steps processed for an FPGA design from design entry to running design (Keywords: Design entry, HDL, schematic, simulation, mapping, placement, routing, bitfile, download)?
- Name the different delay times in a synchronous digital system! How do they influence the maximum frequency of a synchronous design?
- What is a clock skew? Why may it be a problem? Explain methods to minimize its impact on synchronous designs in FPGA designs!
- What is the difference between asynchronous and synchronous reset?
- Explain the problems occuring when signals are sampled by a non-synchronized clock!
- Explain the use of Hardware Description Languages (HDL)! What are differences between HDLs and common computer programming languages?
- What is Verilog? Explain what a Verilog description includes (global description, not details like individual instructions etc.)!
- How can we implement a state machine (driven by a given clock) in verilog?
- How can we simulate a digital design (Keywords: logical/behavioural simulation, timing simulation, testbench, testfixture, event driven simulation, vendor libraries)?
- What is a testbench? Explain it using a sketch (Keywords: UUT, DUT, instantiation, stimulus signals)!

Exam: A 20 minutes oral exam including the defense of the projects elaborated during the course

VHDL-related questions

- How can a "finite state machine" (FSM) be specified in VHDL?
- What is a process in VHDL? Explain their representation in hardware!
- How can you perform a simulation of a circuits VHDL-model?
- Explain the different delay models in VHDL!
- What is an entity in VHDL? Write an example to explain it!
- What are the elements of a port declaration in a VHDL entity?
- Compare the entity declaration with the symbol in a schematic entry tool!
- What are generics? Give an example of their usage!
- What is an enumeration type in VHDL? Give examples!
- Explain the difference between integers and bit-vectors! Where in your VHDL code do you prefer the different types?
- Where do we need conversion functions in VHDL? Give an example!
- How are digital signals represented in VHDL? (Explain types in VHDL!)
- How can delays be included in VHDL-models?
- What are the differences between signals and variables in VHDL?
- How can an existing VHDL-model of a partial design can be used in a top design? (Hierarchical designs)
- How can we implement special blocks like drivers, memories, clock generators from a hardware library in VHDL?
- Explain the difference between behavioural and structural descriptions in VHDL!
- How do you perform a post layout simulation in a VHDL design environment?
- Which information do we need in addition to the VHDL-description fort design synthesis?
- How can synchronous and asynchronous resets be implemented in a clocked process?
- How can signal bundles (busses) be implemented in VHDL?
- Explain the idea of resolved signals and their use!
- What should you observe when you use the data type real?
- Explain the effects of the use of operators like '+', '-' et cetera for the VHDL synthesis and the synthesized design!
- Explain the terms top-down and bottom-up by means of a VHDL based design!
- What are packages in VHDL?
- Explain the use of functions and procedures in VHDL!
- What are attributes in VHDL? Give examples fort the usage of attributes for signals, arrays and types!
- Why should a case construct in a VHDL model for synthesis cover all possible choices of the selector?
- Give examples for VHDL code that is usable in simulation but not for synthesis!

Abschluss: 20 min mdl. Prüfung, einschließlich Diskussion der im Kurs erarbeiteten Projekte

Vorbereitungsfragen für die mündliche Prüfung PIS 2017

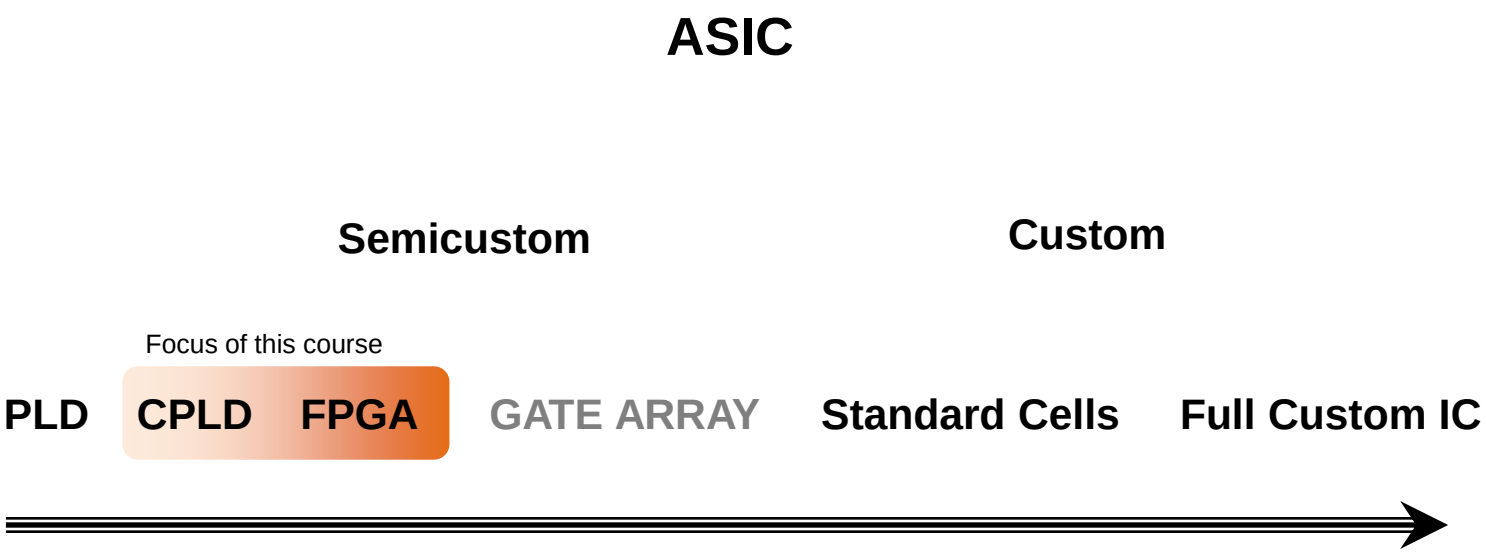
- Erläutern Sie Struktur, Komplexität Programmierbare Logikbausteine und ASICs!
- Erläutern Sie Möglichkeiten des rechnergestützten Entwurfs digitaler Logik mit PLD und FPGA!
- Wie wird eine Zustandsmaschine auf ein CPLD abgebildet?
- Stellen Sie FPGA, Gate Arrays, Standardzellen-IC, und Full Custom-IC bzgl. Struktur, Komplexität und der „Personalisierung“/Programmierung gegenüber!
- Wie verhalten sich verschiedene Kostenfaktoren bei der Realisierung von Entwürfen mit unterschiedlichen ASIC-Kategorien zueinander
- Skizieren Sie den Aufbau eines GAL-Schaltkreises.
- Vergleichen Sie typische CPLD und FPGA.
- Stellen Sie den Entwurfsablauf für eine Logikrealisierung mit FPGA dar.
  - Ausgehend von einem Schaltplanentwurf
  - Ausgehend von Automatengraphen
- Welche Schritte werden bei der Schaltungsimplementierung in FPGA durchlaufen? Was passiert in den einzelnen Teilschritten? (Stichworte: Mapping, Placement …)
- Wie können FPGA "programmiert" werden?
- Stellen Sie den Entwurfsablauf für eine Logikrealisierung mit PLD dar.
- …….
- Skizzieren Sie den Aufbau eines FPGA und benennen Sie die Komponenten.
- Wie können FPGA und PLD programmiert werden (Aufbau und Programmierung der Zellen)?
- Wie kann eine Zustandsmaschine in Verilog beschrieben werden?
- Wie unterscheiden sich asynchrones und synchrones Reset?
- Was ist zu beachten, wenn mit einem getakteten digitalen Schaltkreis Signale abgetastet werden, die nicht mit dem Takt synchronisiert sind?
- Welche Möglichkeiten der Takterzeugung und Synchronisierung bieten FPGA?
- Welche Formen der Beschreibung einer digitalen Schaltung bietet die Hardwarebeschreibungssprache Verilog?
- Was ist eine Testbench? Erläutern Sie ihre Funktion anhand einer Skizze (Stichworte: UUT, DUT, Instanz, Stimuli, Ausgänge).

Abschluss: 20 min mdl. Prüfung, einschließlich Diskussion der im Kurs erarbeiteten Projekte

VHDL-bezogene Fragen

- Wie kann man in VHDL eine „Finite State Machine“ beschreiben?
- Was sind Prozesse in VHDL?
- Wie führt man eine Simulation eines VHDL-Modells einer Schaltung aus?
- Erläutern Sie die unterschiedlichen Verzögerungsmodelle in VHDL.
- Was ist eine „Entity“ in VHDL? Wie wird sie beschrieben?
- Was wird in der Port-Deklaration einer Entity beschrieben?
- Vergleichen Sie die Entity-Deklaration mit dem Symbol eines grafischen Eingabetools.
- Was sind Generics? Geben Sie Beispiele für deren Verwendung.
- Was ist ein „Enumeration Type“? Geben Sie Beispiele an.
- Wie kann man digitale Signale in VHDL darstellen (Typen in VHDL)?
- Wie können Verzögerungszeiten in ein VHDL-Modell eingearbeitet werden?
- Worin besteht der Unterschied zwischen Signalen und Variablen in VHDL?
- Wie kann ein bereits vorhandenes VHDL-Modell einer Teilschaltung in eine VHDL-Beschreibung eingebunden werden?
- Erläutern Sie den Unterschied zwischen Verhaltens- undStrukturbeschreibungen in VHDL.
- Wie kann in einer VHDL-Entwurfsumgebung die Simulation nach der Synthese und nach dem Layout durchgeführt werden?
- Welche Daten werden für die Synthese einer Schaltung zusätzlich zur VHDL-Beschreibung benötigt?
- Wie kann in einem Prozess das synchrone und asynchrone Reset dargestellt werden?
- Wie werden Signalbündel (Busse) in VHDL dargestellt?
- Erläutern Sie das Konzept der „Resolved Signals“.
- Was ist bei der Verwendung des Datentyps „Real“ in VHDL zu beachten?
- Erläutern Sie die Verwendung der Operatoren `+`, `-` usw. in VHDL-Beschreibungen für die Synthese.
- Erläutern Sie „Top-Down-„ und „Bottom-Up-„ Entwurf anhand des Schaltungsentwurfs mit VHDL.
- Was sind Packages in VHDL?
- Erläutern Sie den Unterschied zwischen „Function“ und „Procedure“ in VHDL.
- Was sind “Attributes“ in VHDL?
- Geben Sie Beispiele für Attribute von Typen, Signalen und Arrays!
- Warum soll eine CASE-Anweisung in einer synthetisierbaren VHDL-Beschreibung immer alle möglichen Werte des Selektors umfassen?
- Geben Sie Beispiele für Anweisungen an, die nur in der Simulation, aber nicht in der Synthese wirksam werden.

ASIC (Application Specific Integrated Circuits)



Increasing number of production steps at vendor (FAB/Foundry)
Decreasing configuration capabilities at user site

PLD (Programmable Logic Devices)

Shipment of **identical** integrated circuits to all users (costumers).

No data flow from the user (customer) to the vendor required.

It's NOT a programme but modification of structure

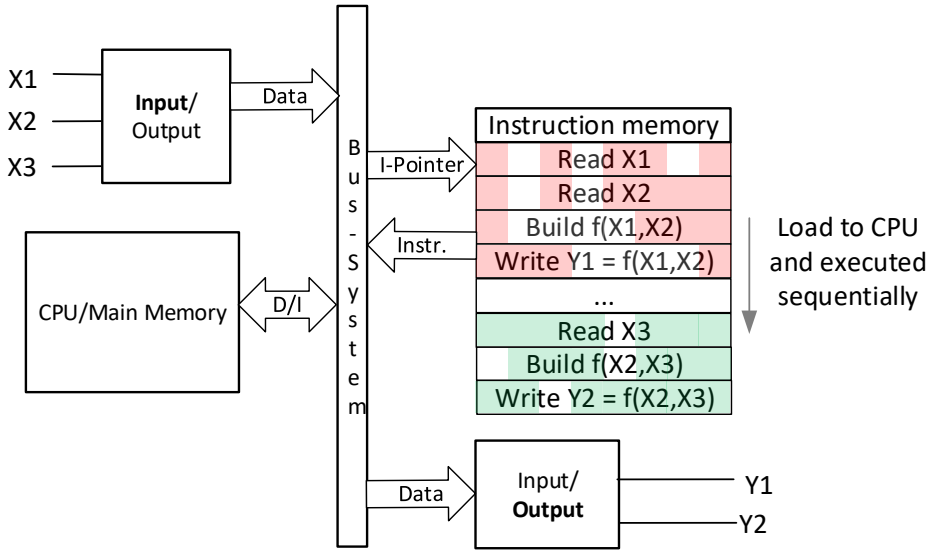
Implementation of the user (customer) design by PROGRAMMING
or better: "Personalisation"

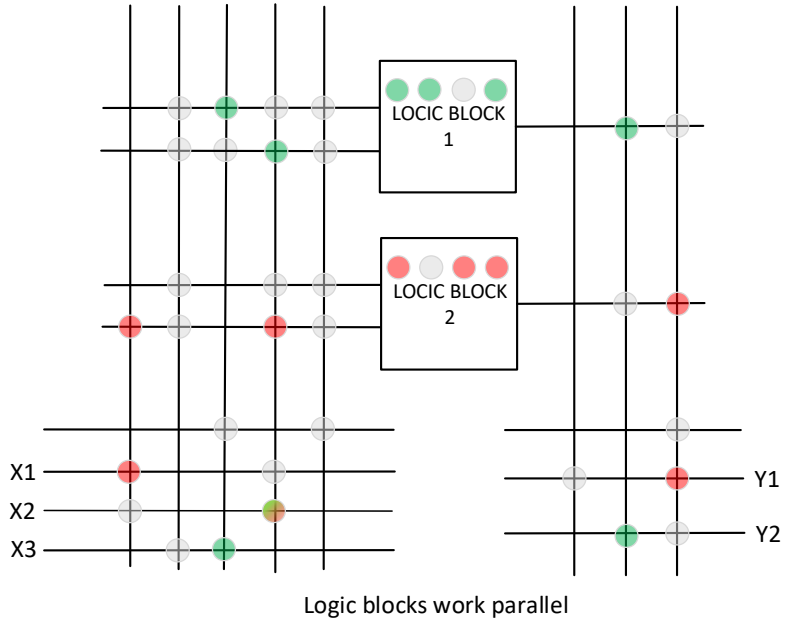
Simple PLD: "Programming" is standard in development systems, often there is simple software delivered with programmer hardware.
Examples: PAL, GAL, MACH etc.
Hardware and software in the range of 100 …. 10.000 €

Complex PLD: Some Hardware and Software from the vendor necessary.
Examples: CPLD from XILINX, Lattice, Altera, Actel.
Hardware and software in the range of 0 … 20.000 €

Programming (Cells/Hardware):
Fuses ("Blow Out" of connections)
EPROM and EEPROM-Cells / Flash
Antifuses ("Break Through " of isolators)
RAM-Cells with MOS-switches

Programming vs. Personalization/Configuration





Logic blocks work parallel

It is possible to parallelise some instructions
by use of input/output vectors, ALUs of n bit width
and multiple processors.
But in principle the flow is sequential.

Connection and Configuration Switches

Program (Instruction) Data Bits

Gate Array

Offer of **prefabricated** chips/dies with given layout and placement of core logic cells and I/O pads (Master) by the vendor.

Implementation of the user design by generation of WIRING PATTERNS.
Wires are manufactured at the vendor.
Only the wire layers (2 .. 4) are made user (customer) specific.

Placement of the cells (all layers of transistors) is identical for all customers using the same master type.

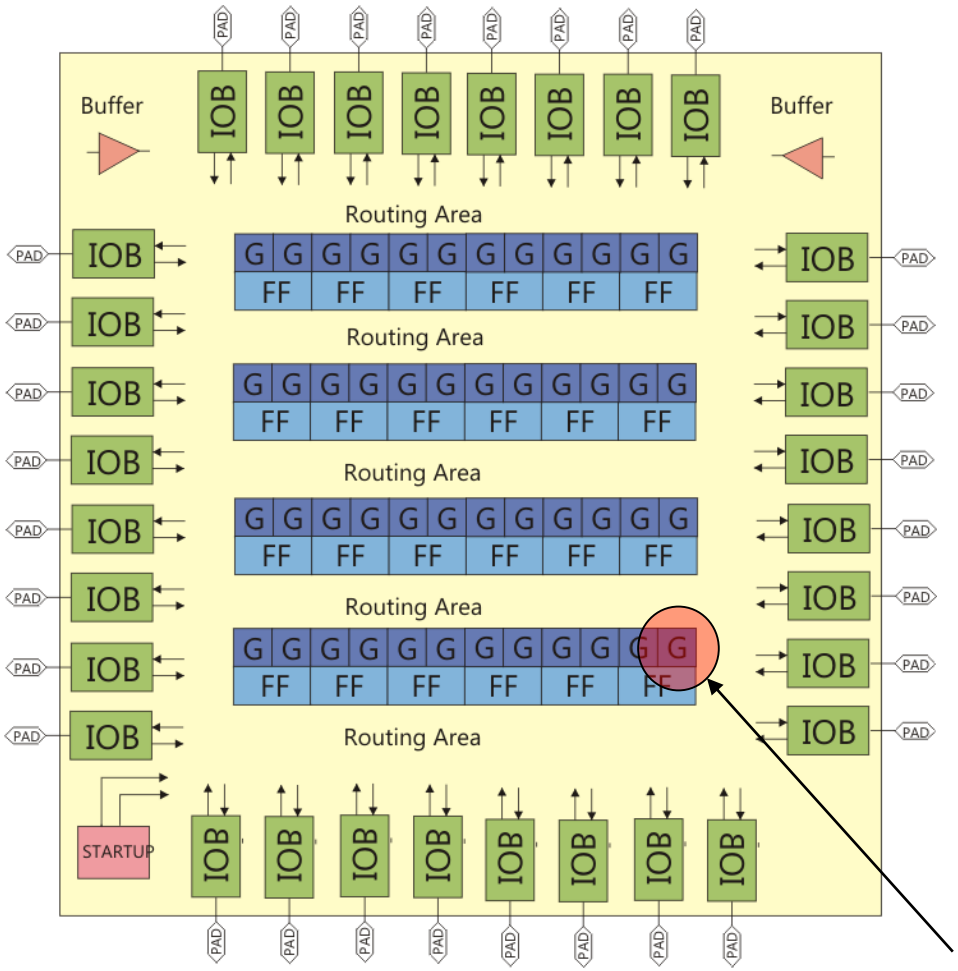
The function of the cells can be established by the wiring layer. Different wiring of transistors produce different logic cells.

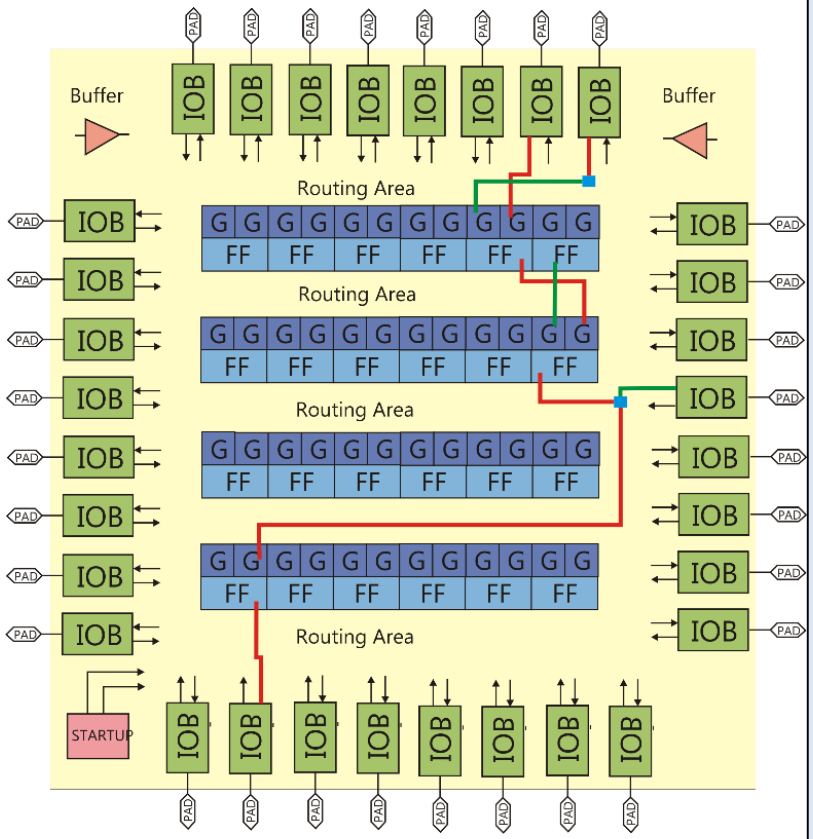
Example: 4 MOS transistors can be combined as a NAND- or NOR-gate by different wiring.

Vendor offers CELL- and MACRO LIBRARIES

Necessary: Development software including simulation

Gate Array





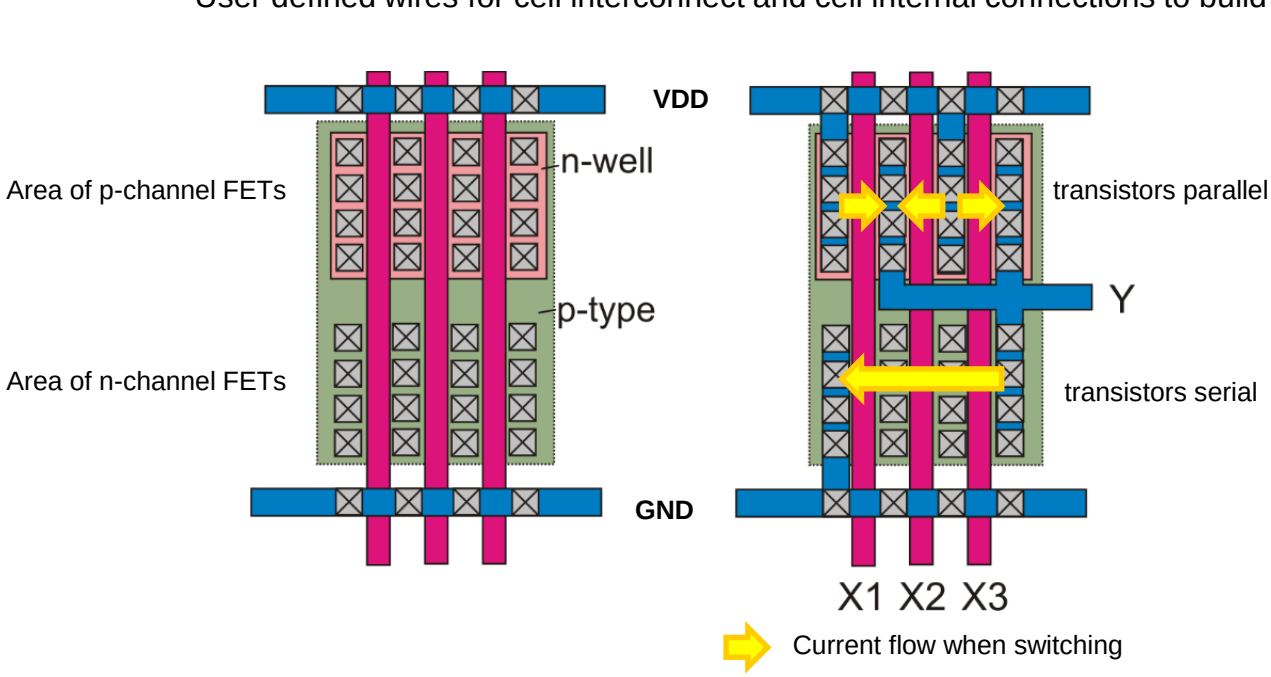
See next slide!

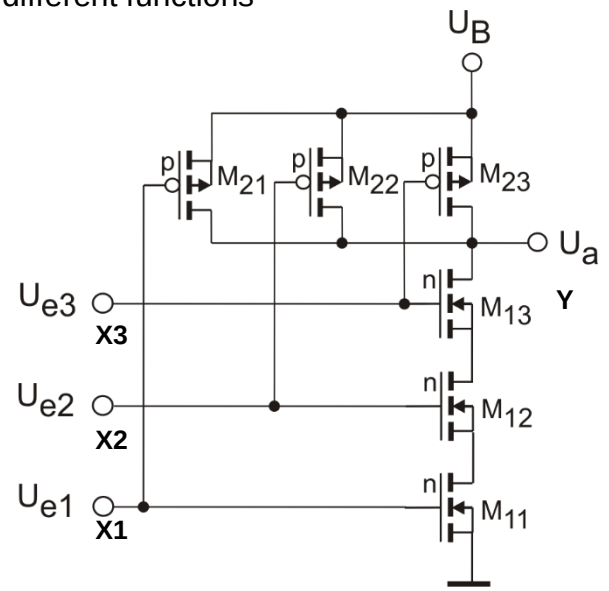
Prefabricated die : MASTER

Final masks with user routing (wires)

Gate Array: Cell Configuration

User defined wires for cell interconnect and cell internal connections to build different functions





3 n-channel and 3 p-channel FETs
unwired SOURCE and DRAIN areas

3 n-channel and 3 p-channel FETs
wired to build a 3-input NAND
n-Kanal-FETs serial
p-Kanal-FETs parallel

Larger structures to build complex gates and flip-flops

FPGA (Field Programmable Gate Arrays)

Offer a prefabricated MASTER similar to GATE ARRAY.
Cells are stucked (number , placement).
Additional: Prefabricated connection structures (wire segments
and programmable interconnection points)

Shipment of **identical** integrated circuits to the customer. In difference to CPLD more granular structure.

CPLD: Mostly AND-OR-field structure with flip-flops at the output.
Simple connection structure (crossbar).

coarse -> fine grained

FPGA: Distributed pattern (lattice/grid) of gates and flip-flops
Connection structure with high degree of freedom but
less variable than in gate arrays because of premanufactured
wire segments instead of free routable wire layer.

At the user site: Programming of the cells
Programming of connections

Development and simulation software necessary

FPGA vs. Microcontroller and Custom-IC

Custom IC	FPGA	Microcontroller
Fixed, expensive development, low cost per chip parallel processing, very fast	**Flexible, expensive, parallel processing, fast but hardware overhead for configuration**	**Flexible, cheap, sequential processing, slow**
	Good for algorithms with parallel data at high data rates	**Good for algorithms with data dependent branches, forks**

SoC: FPGA + Microcontroller(s)

Examples for FPGA Application Fields

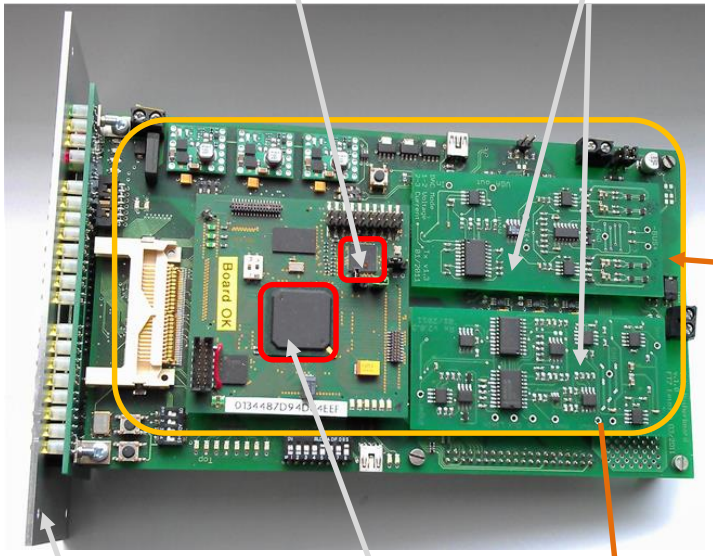
ASIC-Prototyping
Aerospace -> Navigation, Position Stabilization, Motion Control, Communication)
Communication -> Data Encryption/Decryption
Military Applications
Medical Applications -> Image Processing (Endoscope, Ultrasound, X-ray)
Measurement Systems (Oscilloscopes, Spectrum Analyser, Mass Spectrometer …)

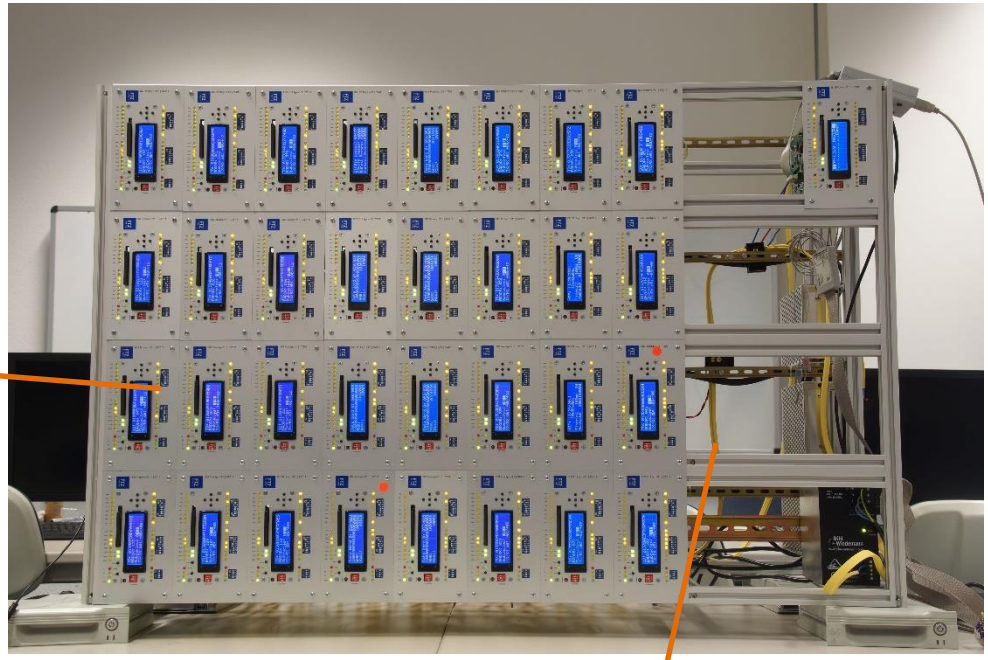
FPGA as a Prototype for Industrial Communication Circuit

OFDM (Orthogonal Frequency-Division Multiplexing

Controller for higher levels (optional)

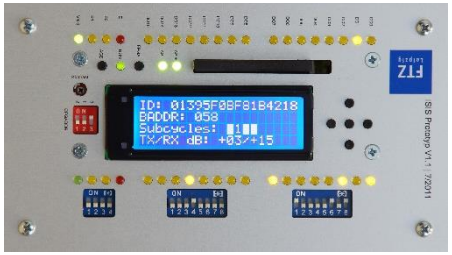
Analog Frontend (Cable driver, receiver)

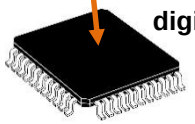




Display and switches replace local application

Prototype FPGA Modulation/ Demodulation MAC-Layer



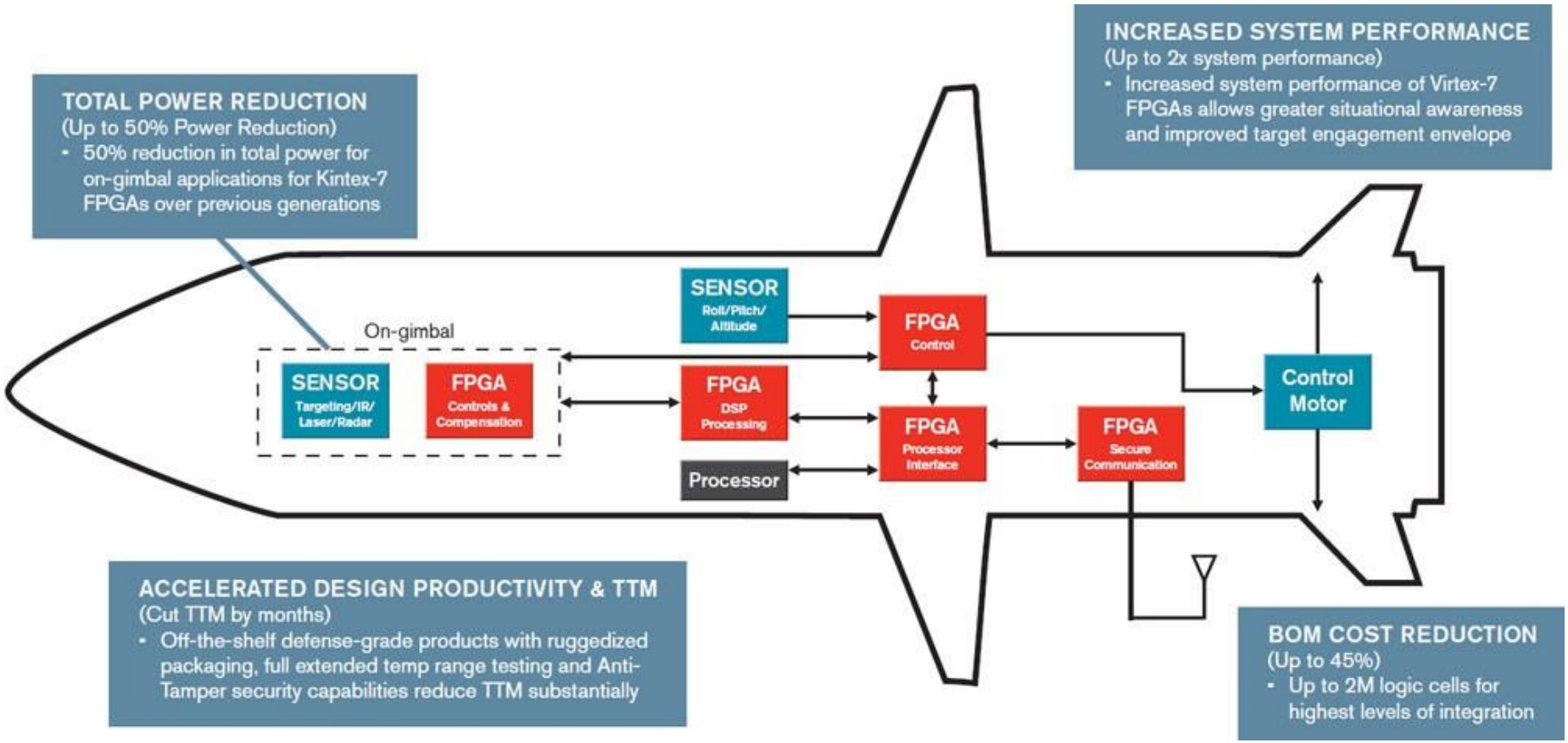


That will be the final system containing analog and digital parts in a single IC

Communication System with Maste-r and 32 Slave-nodes

Communication on Cable

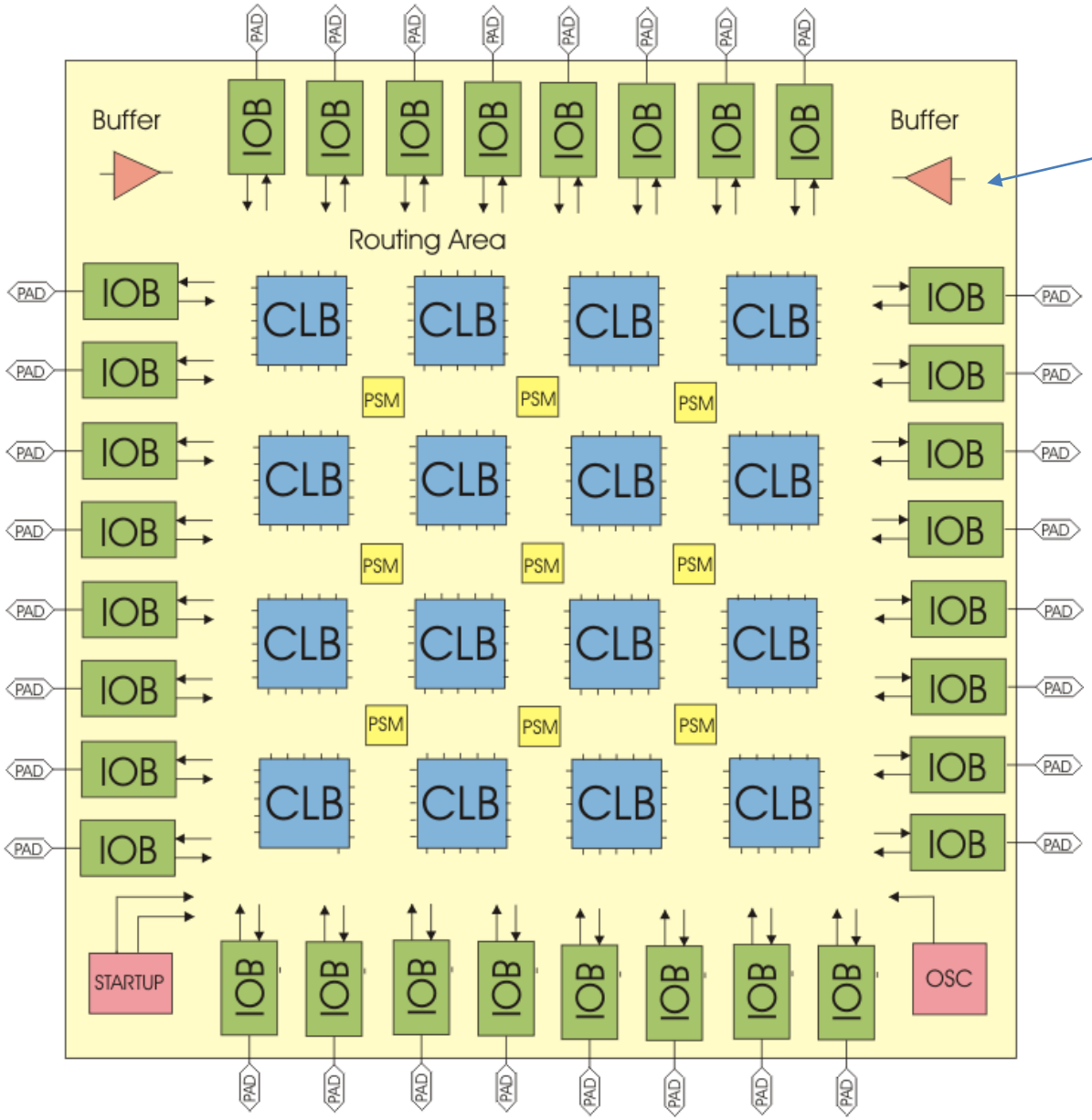
FPGA Applications



FPGA-Vendor XILINX shows use of FPGA in weapons (missiles, munitions)

Source: https://www.xilinx.com/applications/aerospace-and-defense/missiles-and-munitions.html

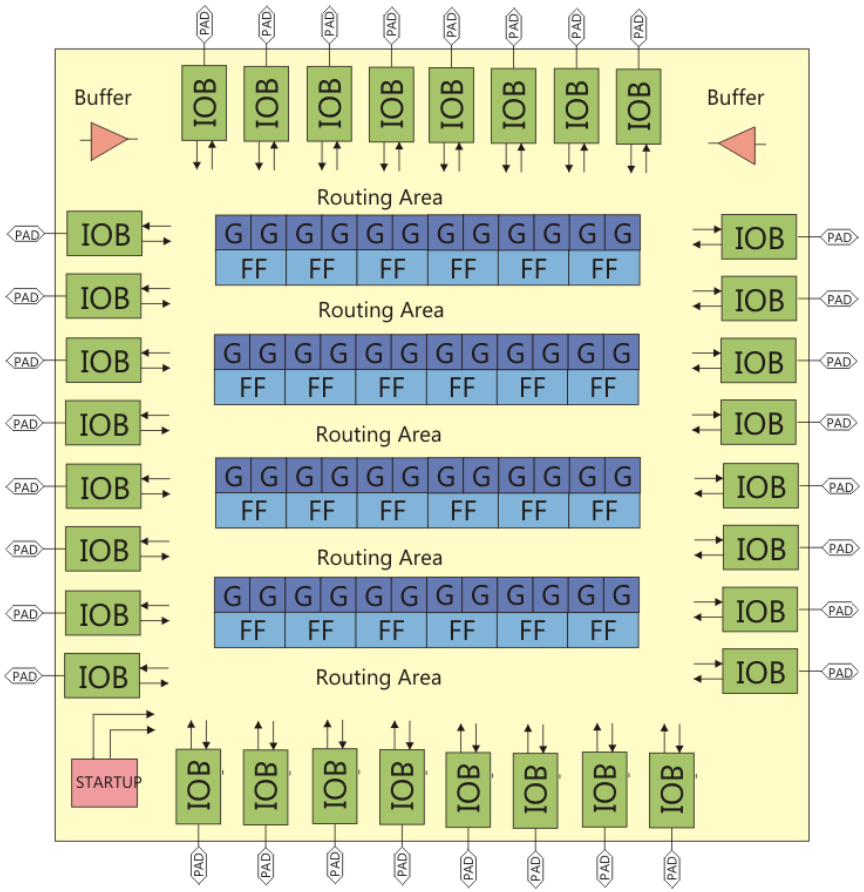
FPGA (Field Programmable Gate Arrays)

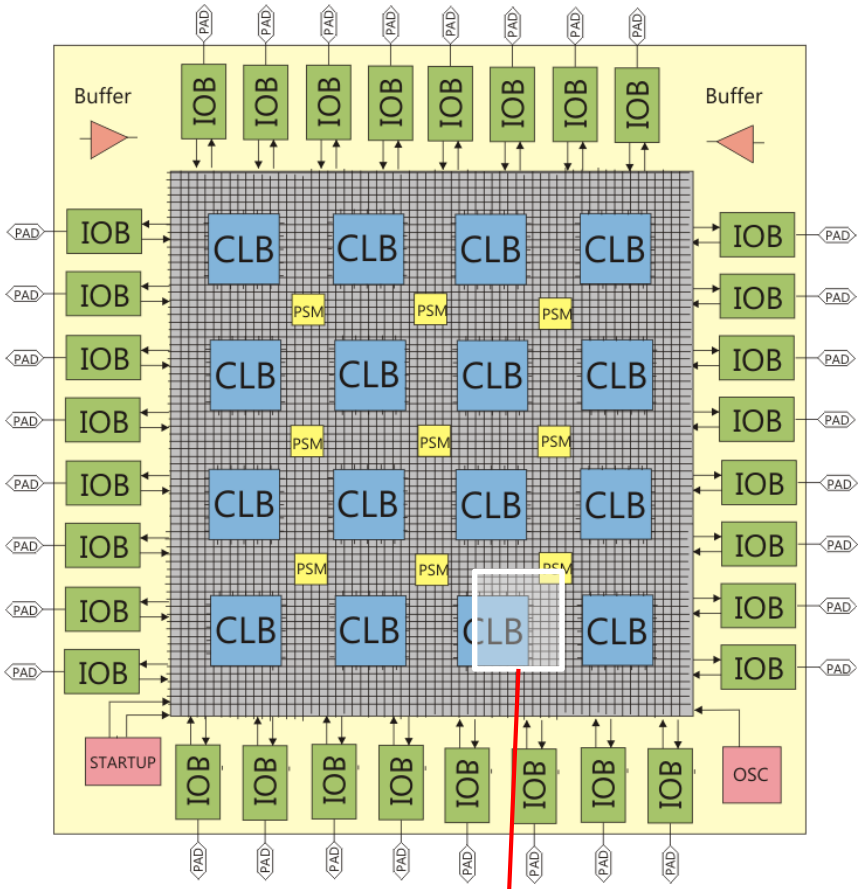


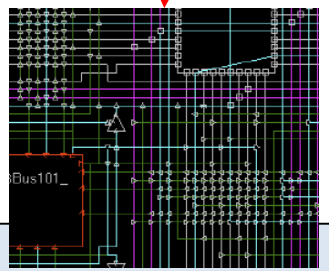
The Buffers and other special cells are spread across the die area even though they are shown in the corners here.

First Xilinx FPGA in 1984 was the XC2064 with **64** Logic Cells with1 FF each, 38 I/O in an 48 Pin DIP.

GATE ARRAY vs. FPGA







Standard Cell IC

Subject of other lessons (VLSI, Integrated Circuit Design)

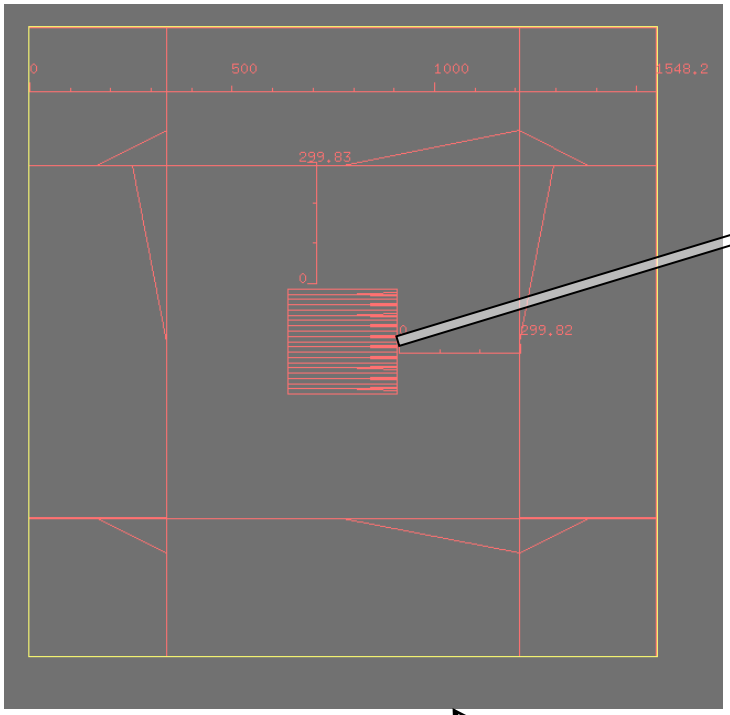
Offer of cell libraries (core logic, I/O), sometimes analog cells
Macrocells (memories)

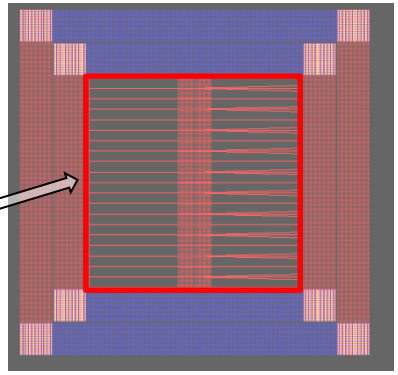
Implementation by the user: Selection of cells (type, number)
Placement
Routing (wires)
According to the rules: Layout constraints
max. power consumption

Placement of transistors depends on user design.
All production masks are manufactured specifically for the customer.

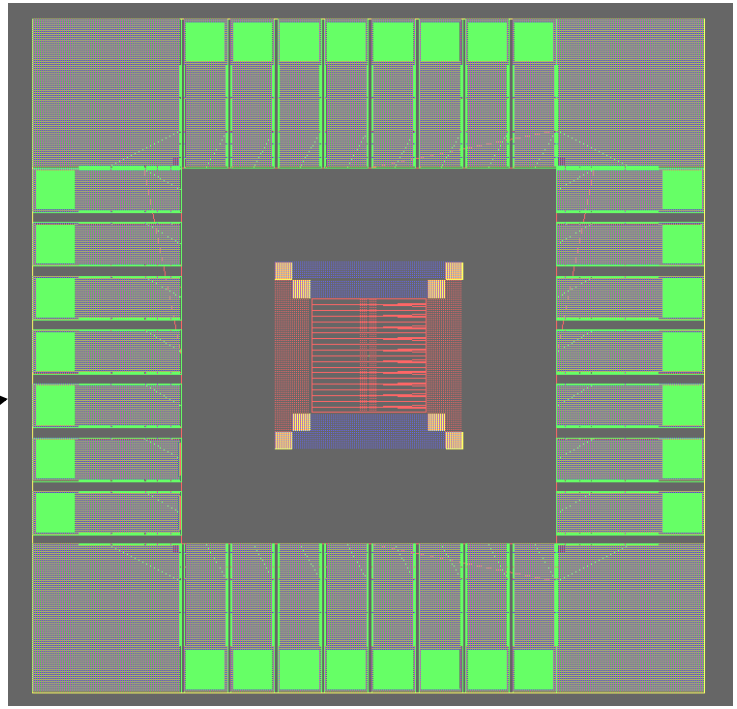
High preparation costs.
Necessary: Development- and simulation software

Standard Cell Design

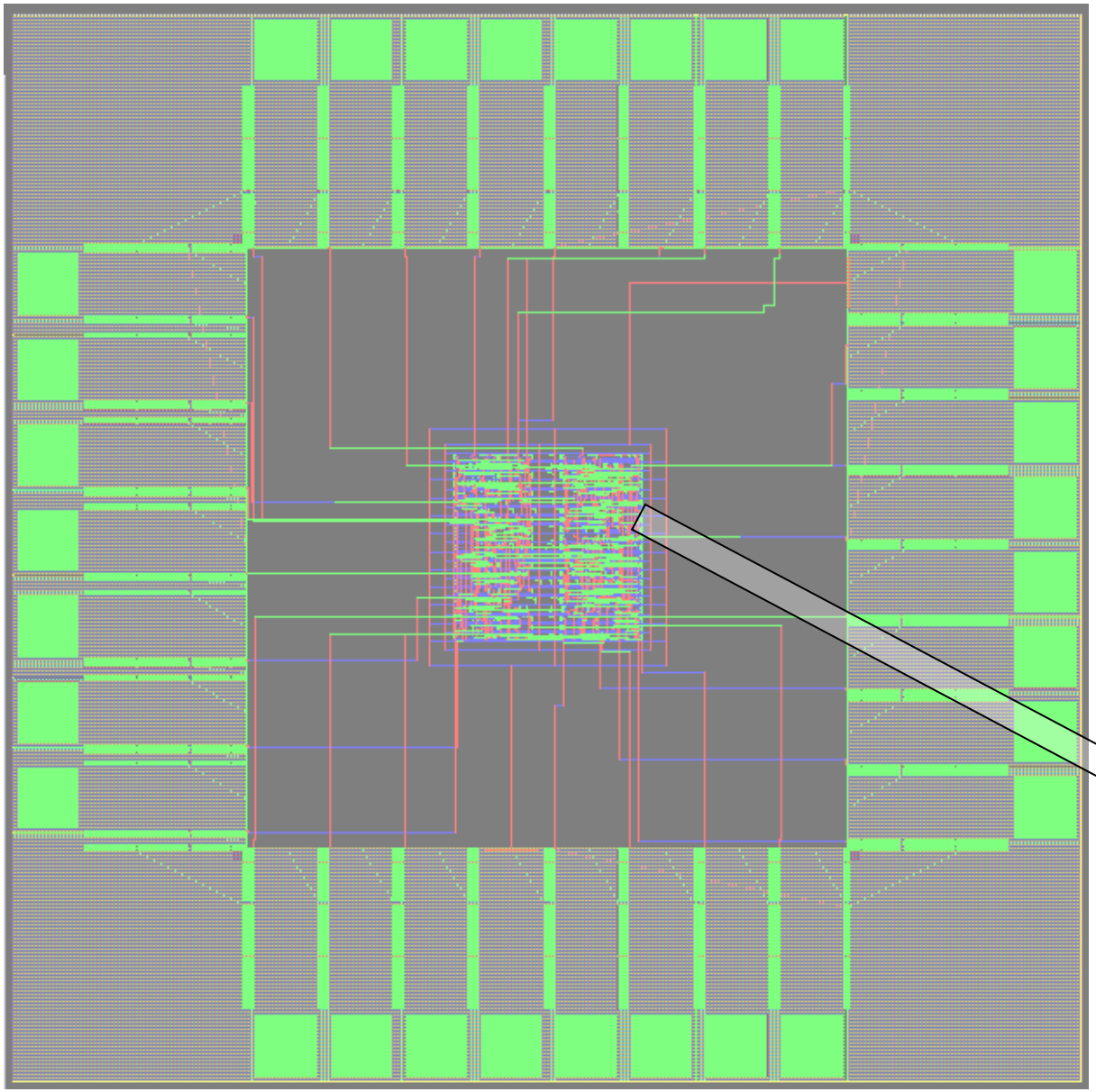




Floorplan
Defining raws for logic cells,
Assign powere lines and rings

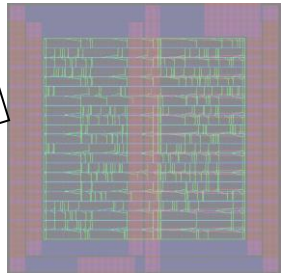


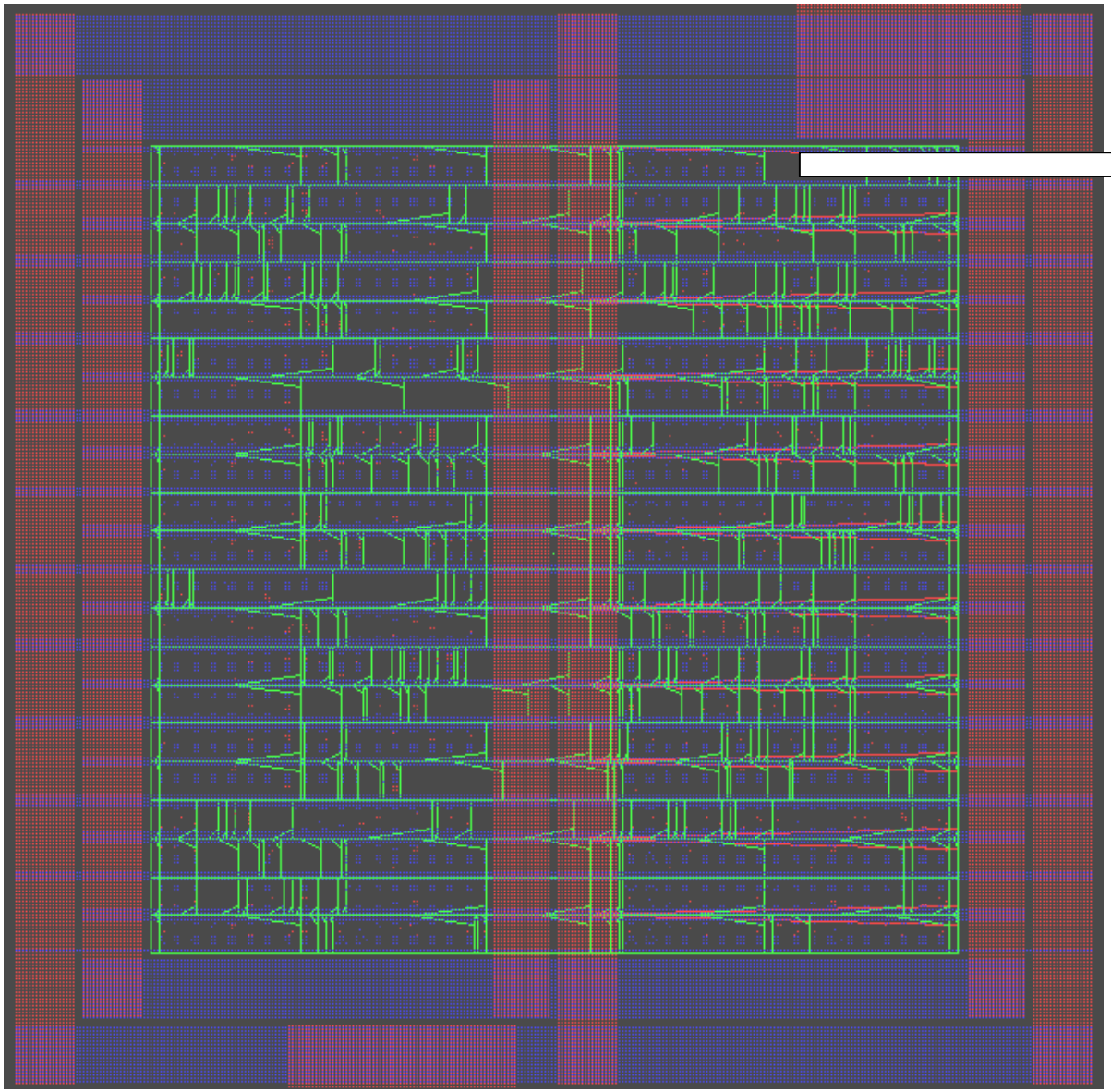
Floorplan
Size, Aspect ratio, core size
Pad cells (core/pad limited)

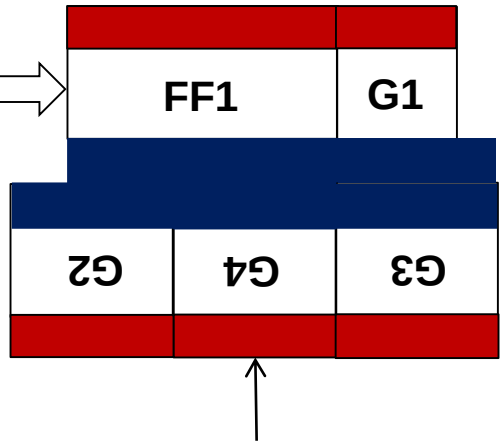


Standard Cell IC
0.25µm AMS-CMOS
Appr. 2.3 mm^2

Core







Cells have identical height .
Cells are placed in raws.
Supply pads at top and bottom.
Every second raw flipped.
Horizontal power bars.

Well defined access points for inputs and outputs.
Abstracts of cells used for routing.

Full Custom IC

Subject of other course (Integrated Circuit Design)

The user gets technology data i. e. information how to construct transistors, capacitors, resistors etc. with well-defined properties

Additional information is given for possible dimensions of transistors, connections, distances etc. -> DESIGN RULES

The user is free in placement and in dimensioning the components

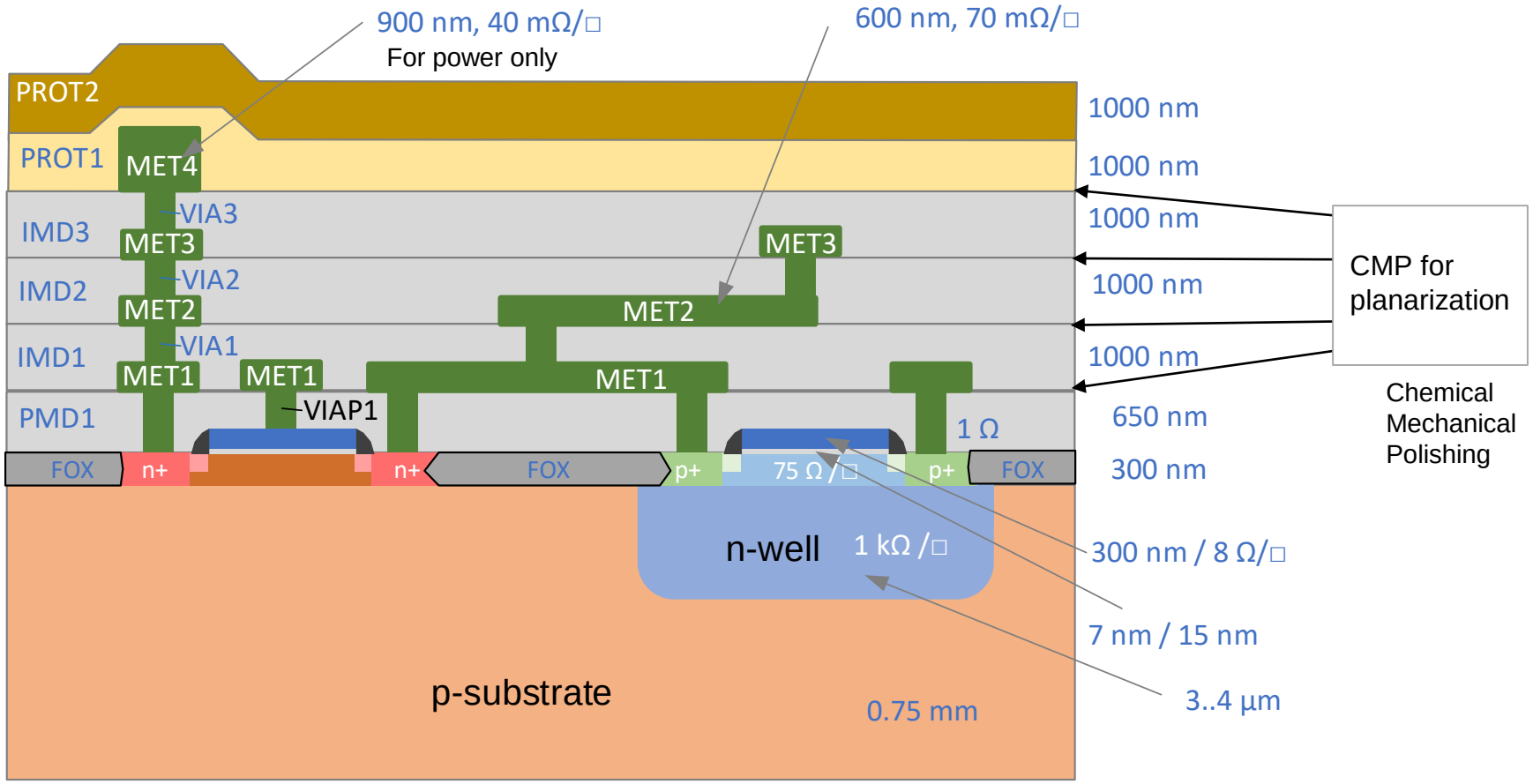
User must have contact to the vendor.

All production masks are manufactured specially for the customer.

Very high preparation and prototype costs.

It is necessary to simulate at transistor level.

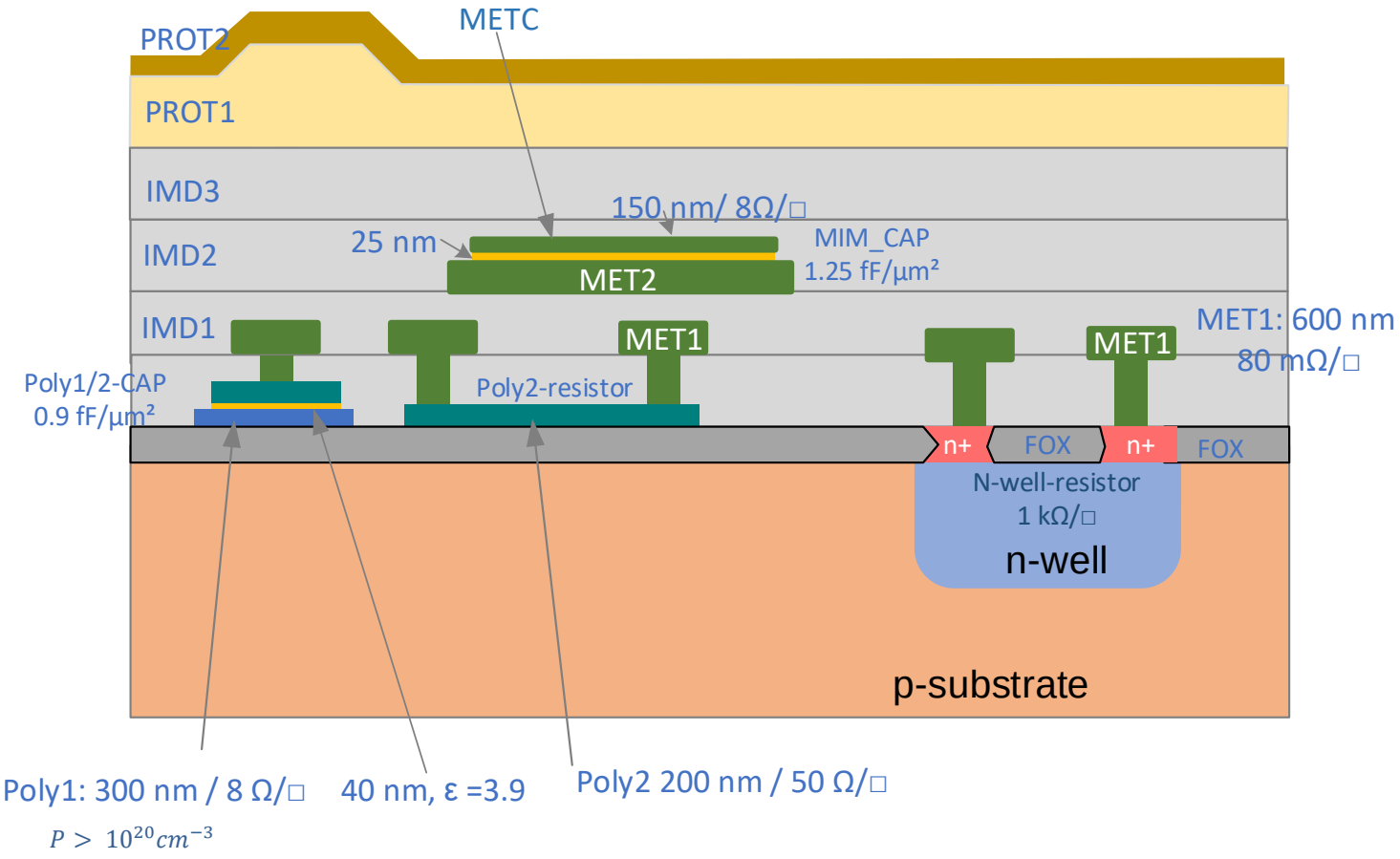
Custom IC Example: CMOS-Process – 4 Layers of wire metal



All dimensions similar to an existing 0.35µm CMOS –process with 4 metal layers

Transistor: $$I_D = \frac{W}{L} \cdot KP_n \cdot \frac{(V_{GS} - V_{th})^2}{2}$$

CMOS-Process – Implementation of resistors and capacitors

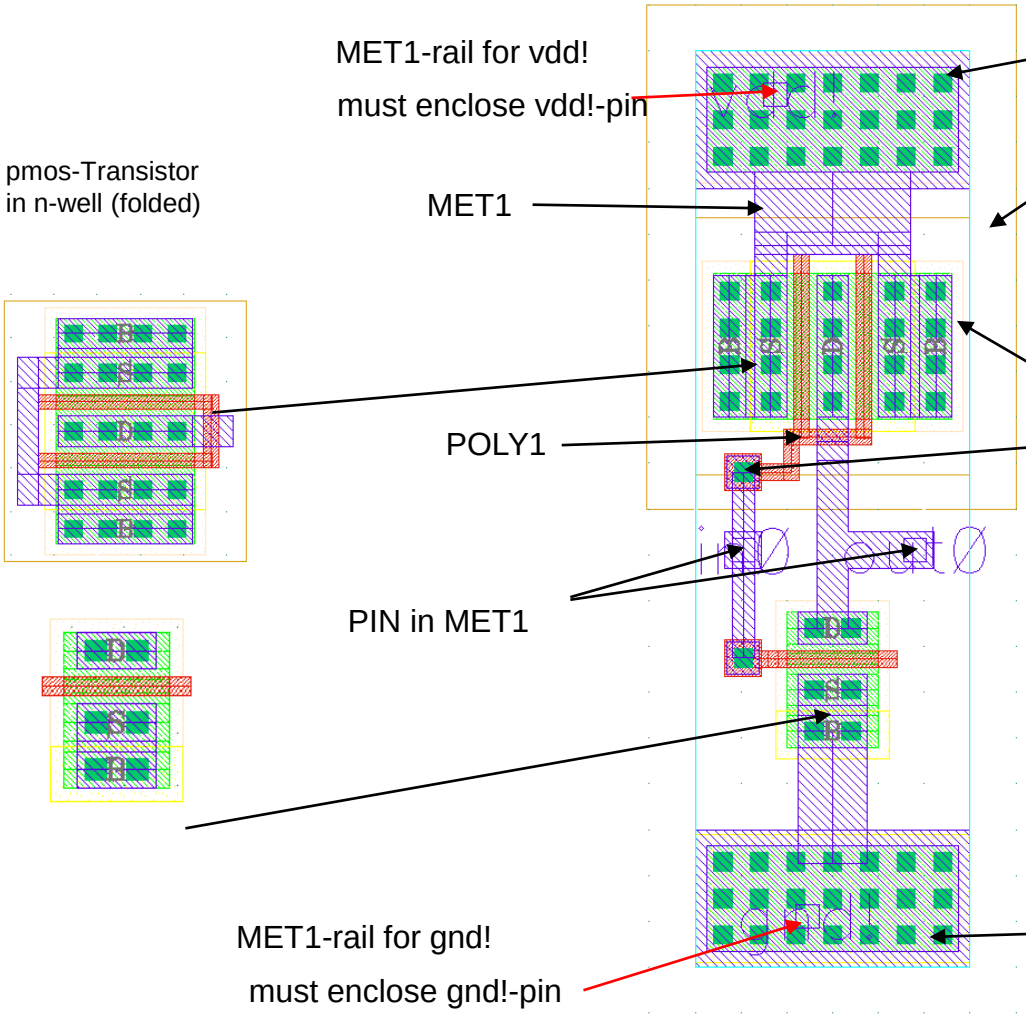


$$R = \frac{\rho}{d} \cdot \frac{L}{W} = R_{\square} \cdot \frac{L}{W}$$

NDIF-Resistor	n+ 80 Ω/□
PDIF-Resistor	p+ 150 Ω/□

Used in I/O cells

Cell Layout in Full Custom IC



Programmable Logic Devices

Objective: Implementation of combinationial and sequential designs
Reduction of size, wires, power consumption
Improvement of reliability
Implementation of design security

Every combinational circuit can be represented as a (Canonical or Standard) Sum Of Products (SSOP) form: (German: Kanonische Disjunktive NormalForm (KDNF)

POS
KANF

Alternative

With n inputs and m outputs we can write:

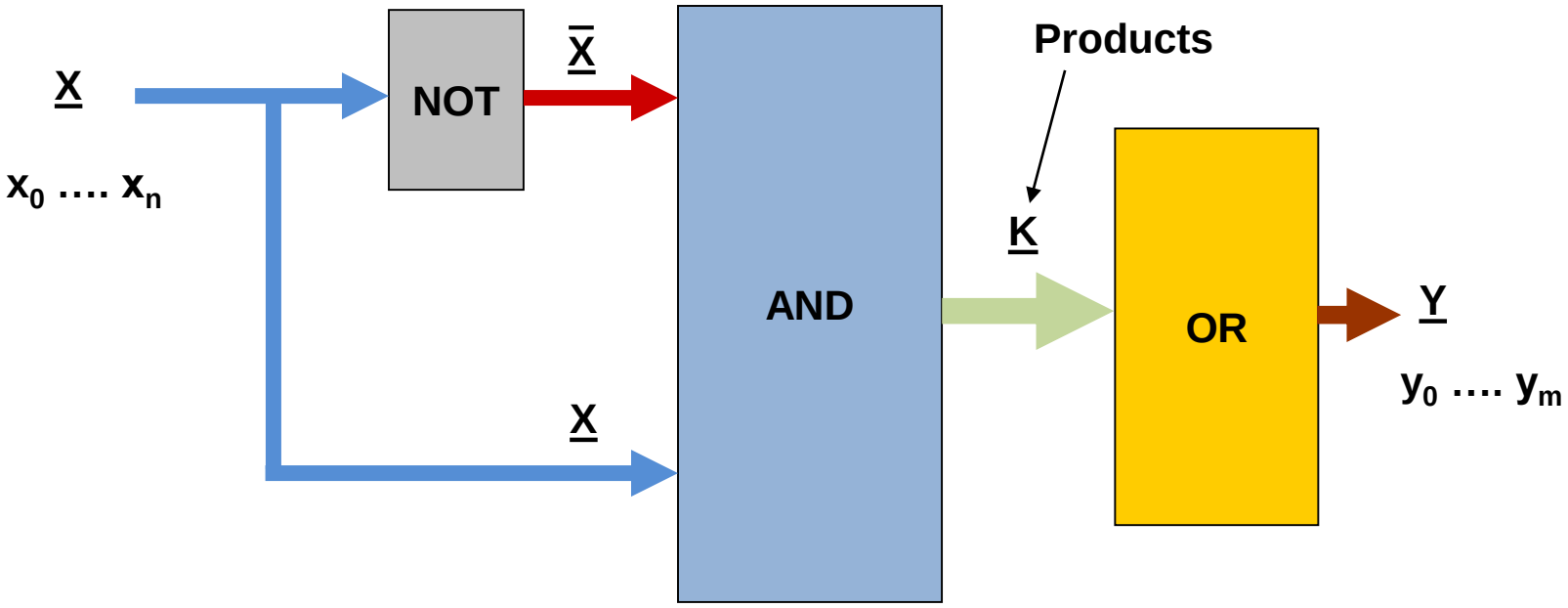
$$Y_k = \bigvee_j P_j \qquad j \in \{0 \ldots 2^{n-1}\} \qquad k \in \{0 \ldots m\}$$

$$P_j = \bigwedge_{i,l} x_i \overline{x_l} \qquad i,l \in \{0 \ldots (n-1)\}$$

Minterm (SOP/DNF)
Maxterm (POS/ANF)
Product
Sum
Prime Implicant (reduced)
Essential Prime Implicant

$$Y_2 = x_2 \overline{x_1} \vee x_3 x_1 \overline{x_0}$$

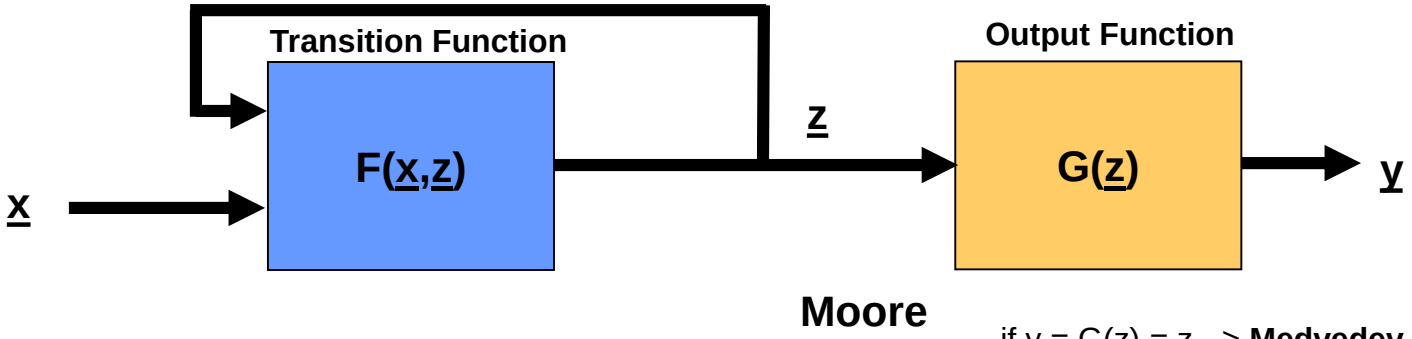
Combinatorial circuit in (NOT)-AND-OR structure or SOP (Sum Of Products)



PROM-Implementation: Canonical form -> All minterms are implemented (one address = one minterm) -> Lookup-table
Number of inputs/outputs limited
Large PROMS are slow (>5…100 ns)

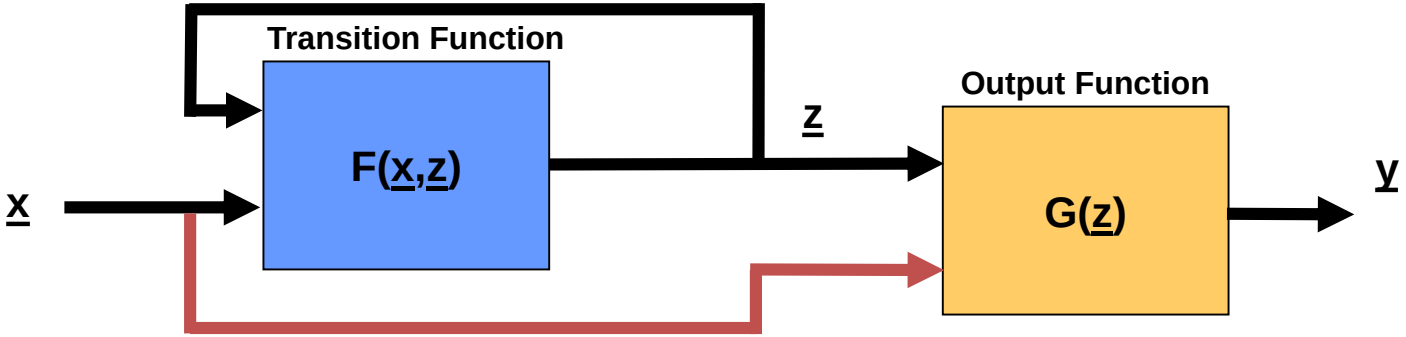
$$Capacity: m \cdot 2^n$$

Finite State Machines (FSM)



Moore

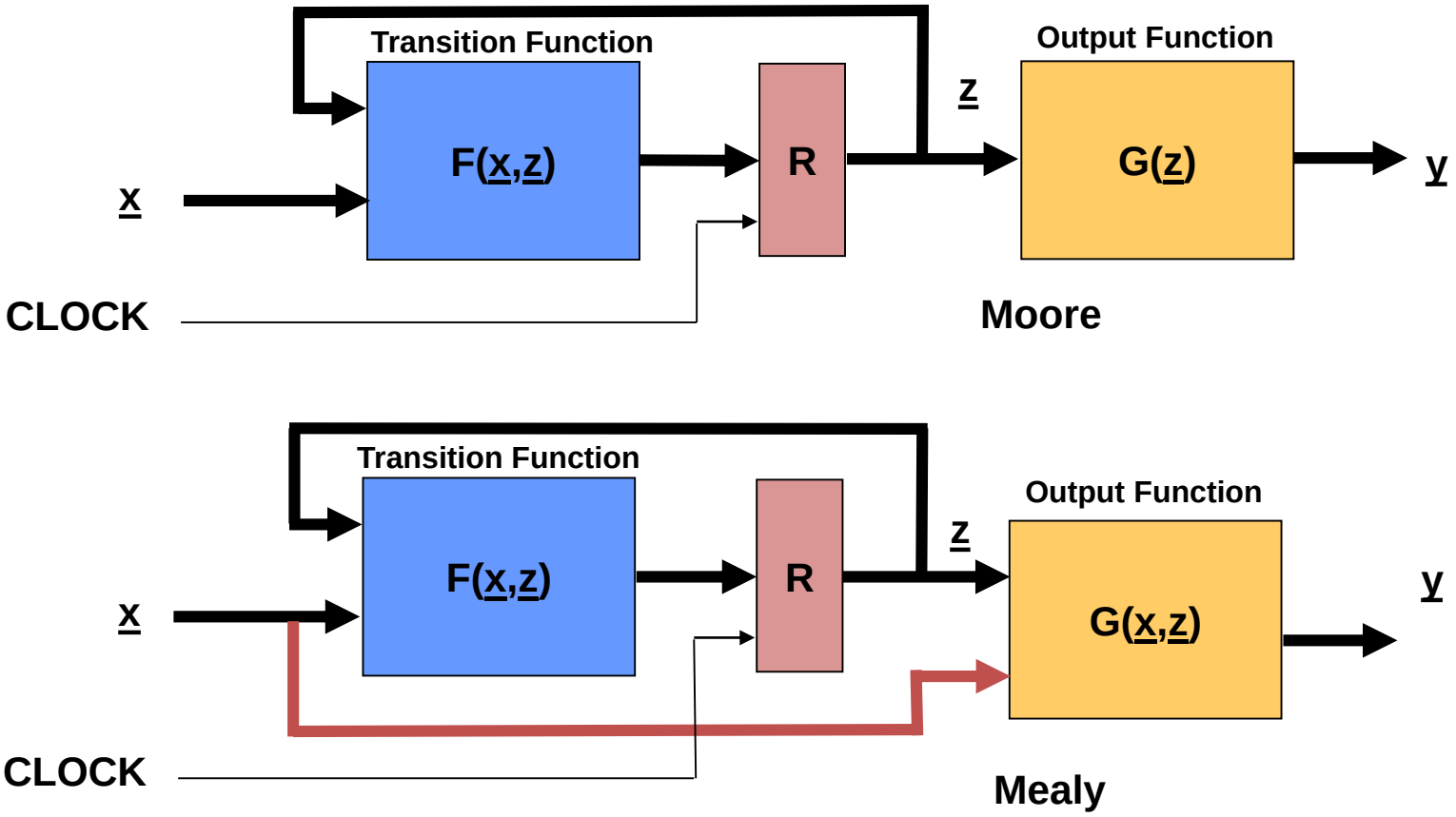
if y = G(z) = z -> **Medvedev**-Machine



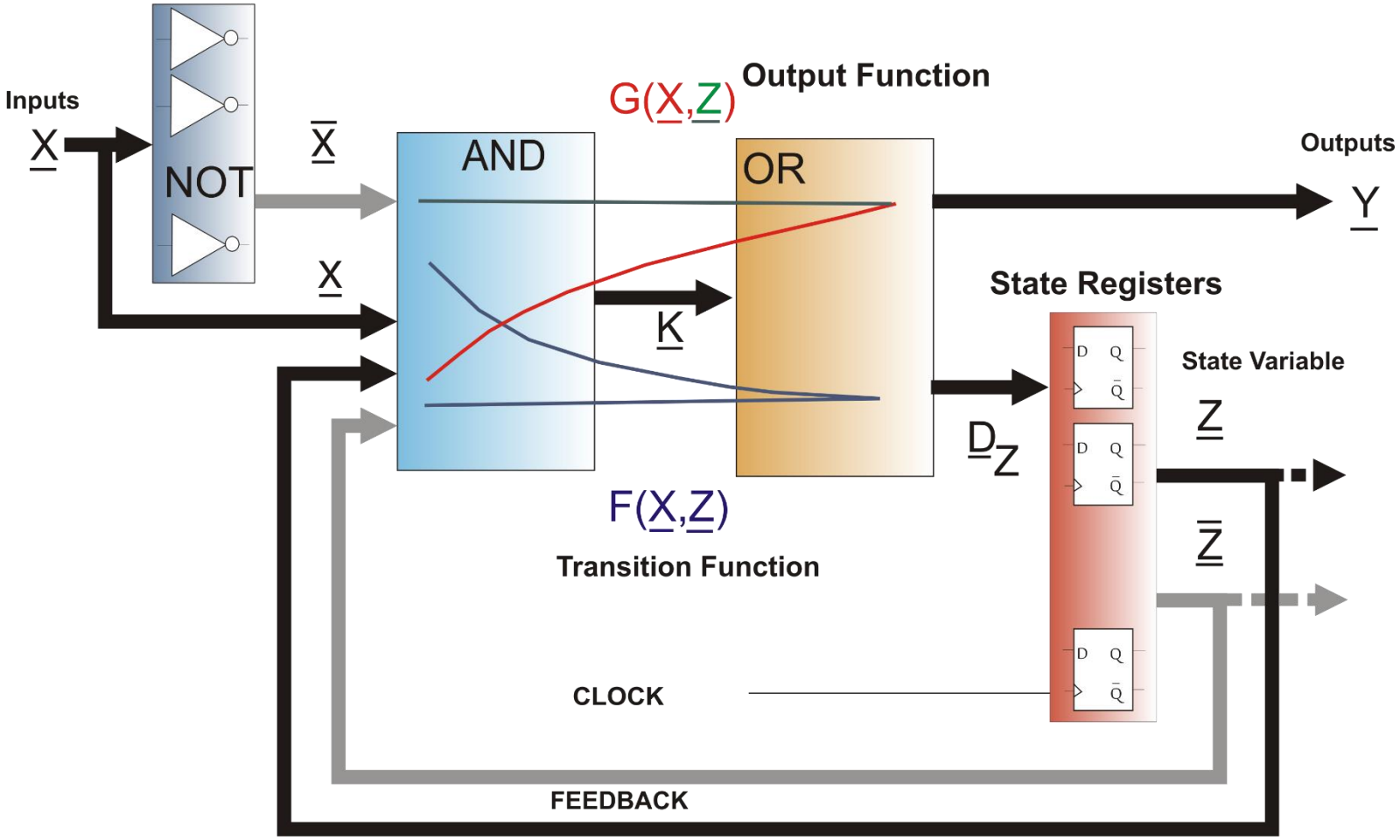
Mealy

Difficult to implement if z has more than one component (runtimes may determinate sequence of states)

Finite State Machines (FSM) with Registers



Implementation of State Machines with PLD



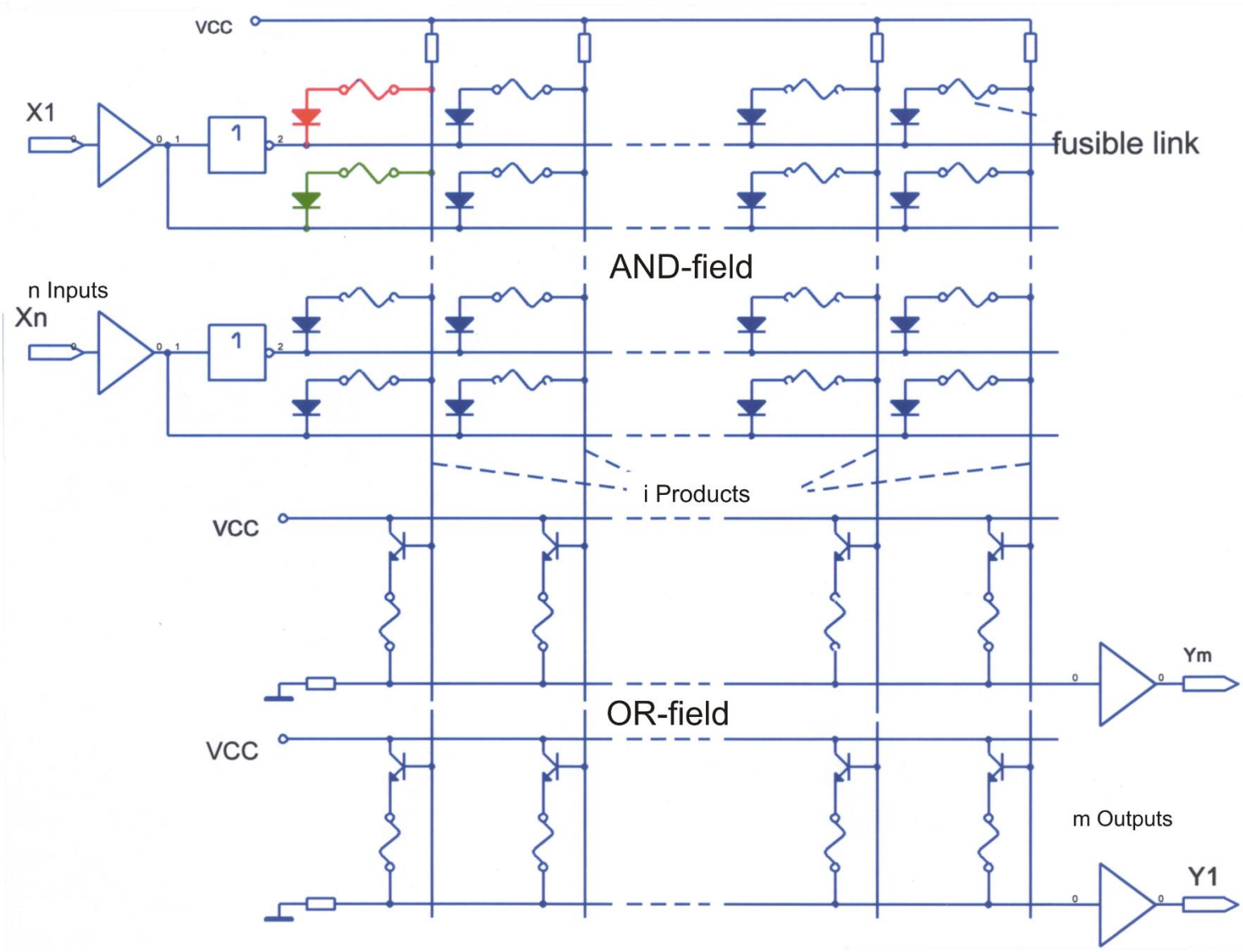
Simple PLD Types

FPLA Field Programmable Logic Aray
Programmable AND and OR Fields
(Productterm Sharing possible)

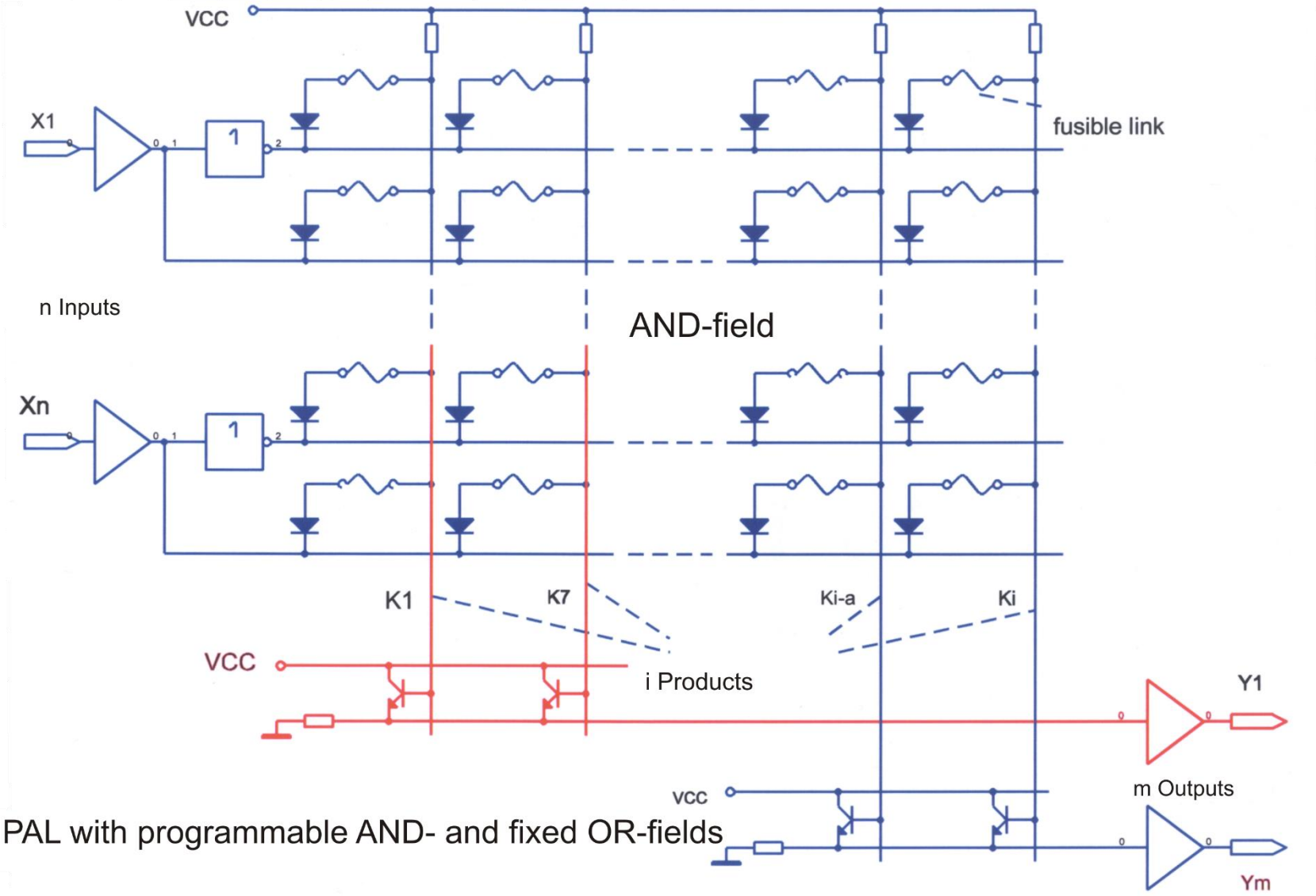
$$y_1 = x_1\overline{x_2} \vee x_3\overline{x_4} \vee x_5$$
$$y_2 = x_1\overline{x_2} \vee x_7\overline{x_4} \vee x_4x_7x_8$$

PAL Programmable Logic Array (Trademark of AMD)
Programmable AND-Field, Fixed OR-Field
(Fixed Number of Products connected to OR gates)

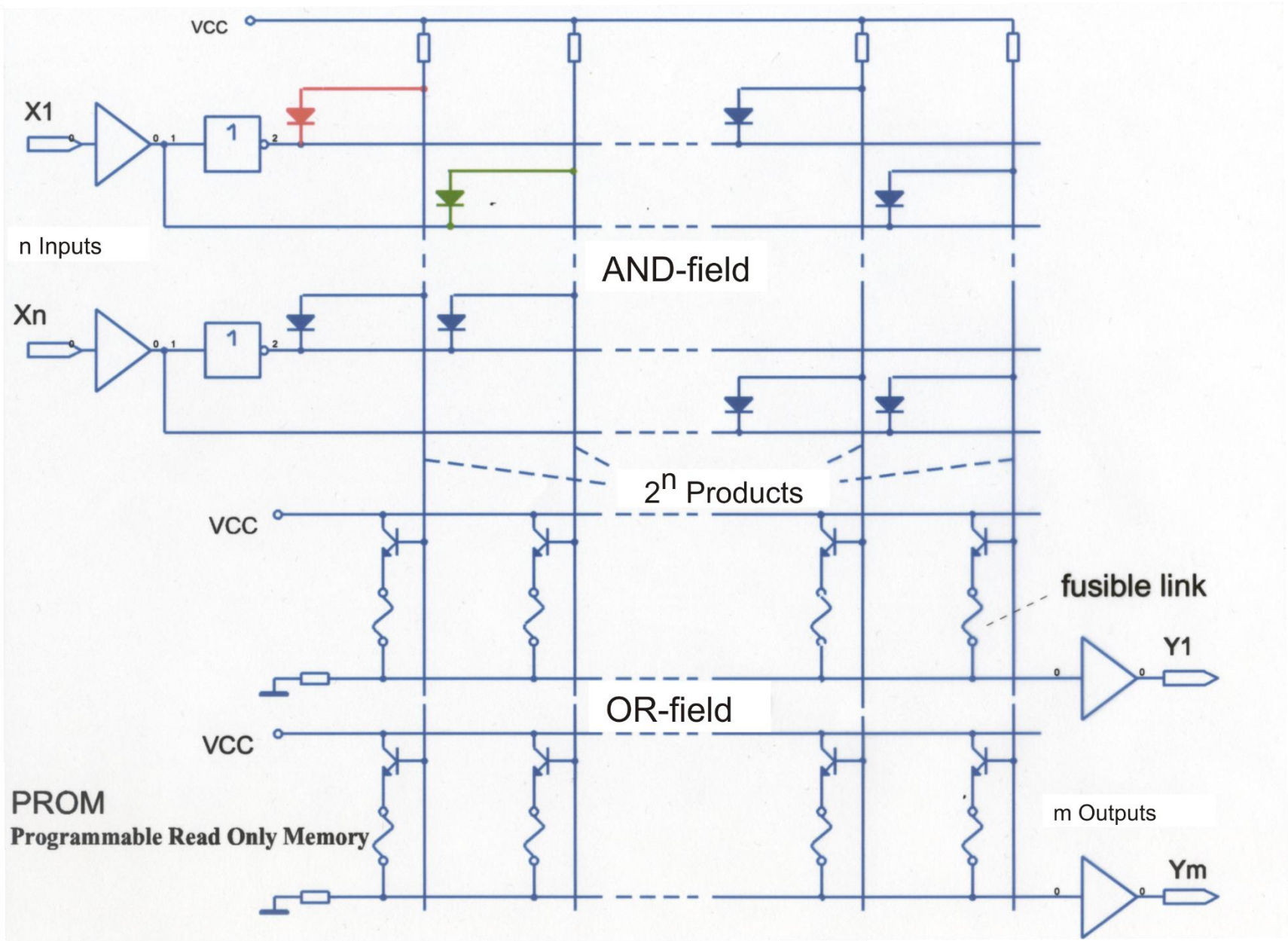
PROM Programmable Read Only Memory
Fixed AND-Field, Programmable OR-Field
(All Minterms are implemented)



FPLA with both programmable AND- and OR-fields



PAL with programmable AND- and fixed OR-fields


VCC
X1
n Inputs
AND-field
Xn
2^n Products
VCC
fusible link
Y1
OR-field
VCC
PROM
Programmable Read Only Memory
m Outputs
Ym

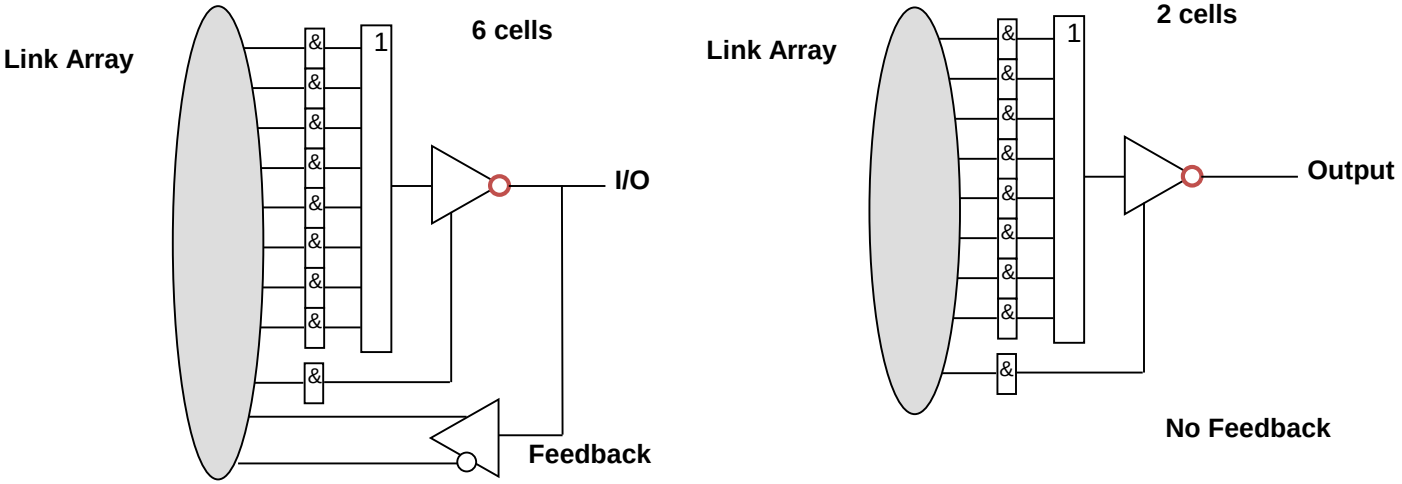
PAL

- Example: PAL 16 L 8 …

Max. number of outputs

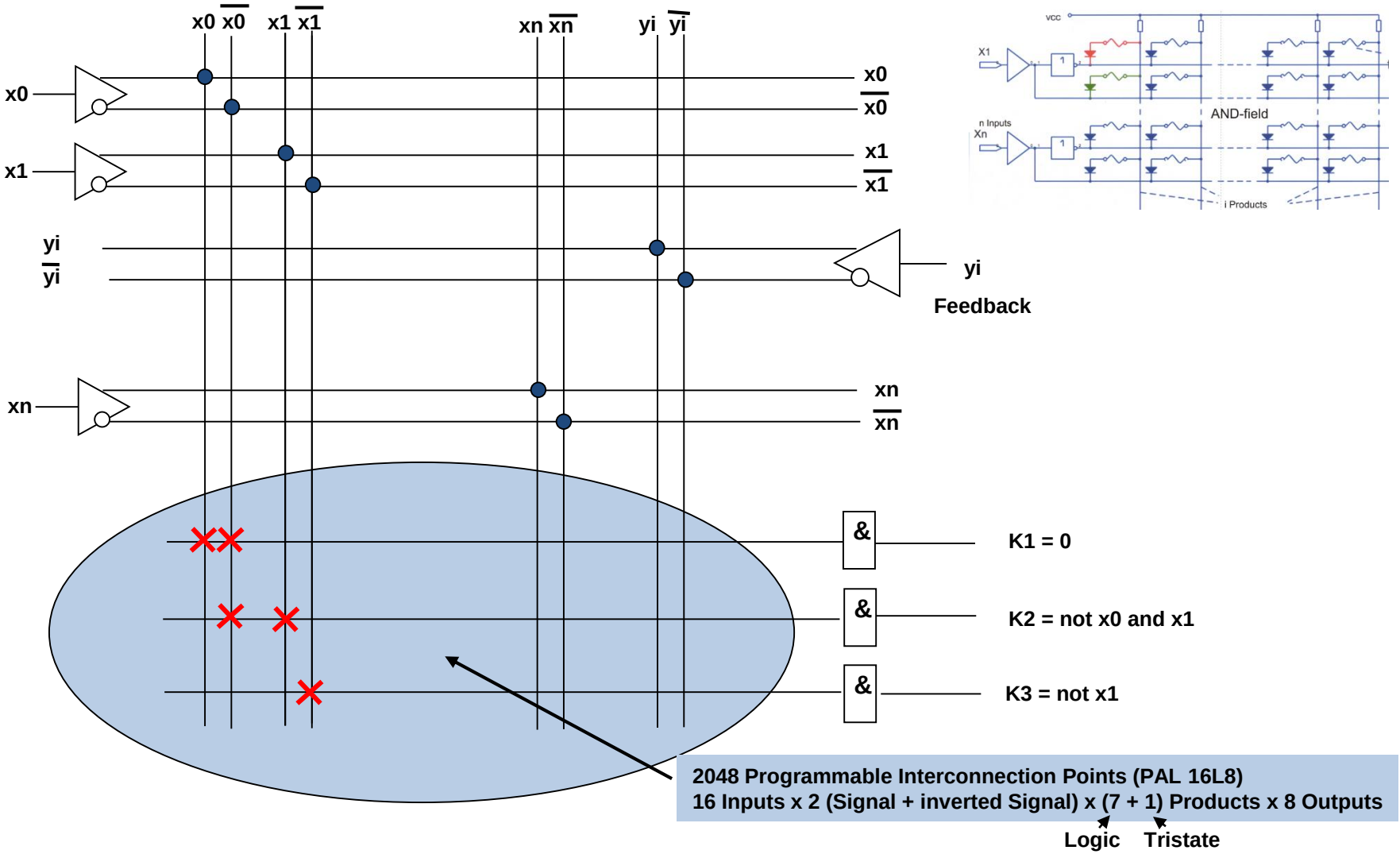
Output cell type (H, L, V, Z …)

Max. number of inputs



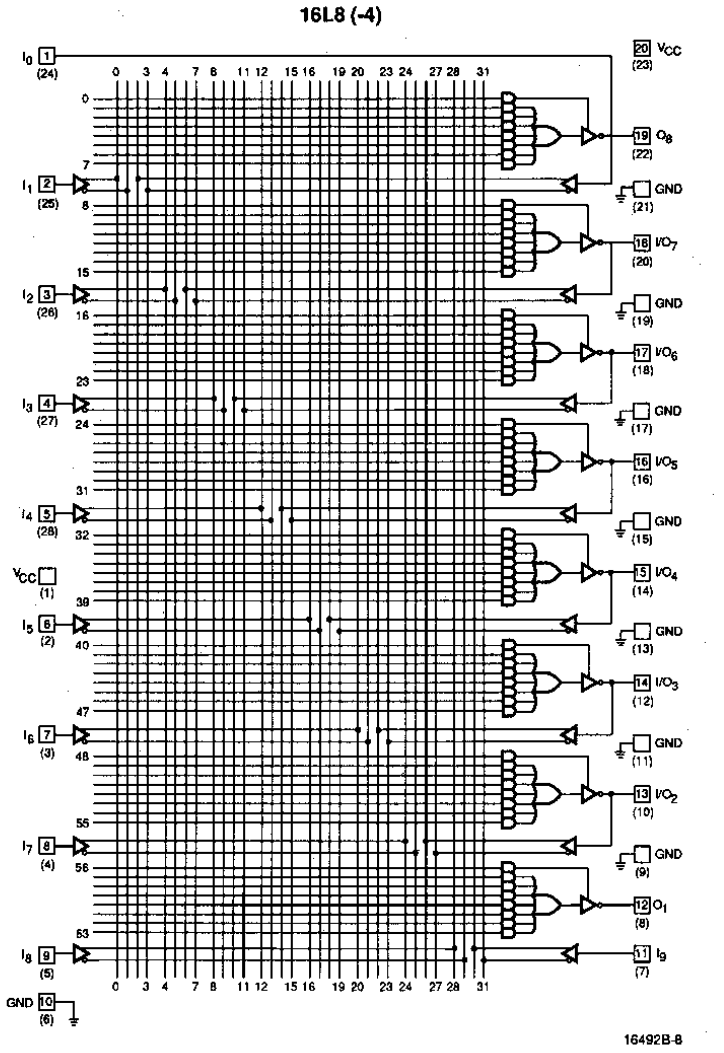
10 inputs fixed
2 outputs fixed → **6 pins variable** → **max. 16 inputs**, **max, 8 outputs**

AND-Field in PLD

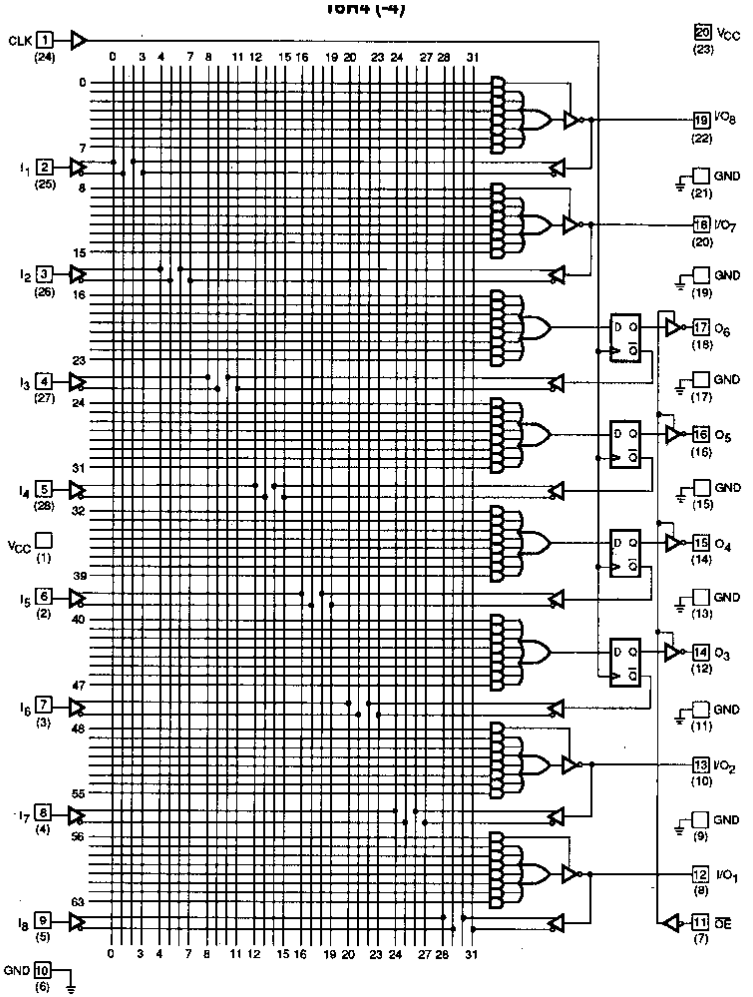


PAL 16L8 16R4 (Databook Schematics)

LOGIC DIAGRAM
DIP and 20-Pin PLCC (28-Pin PLCC) Pinouts



LOGIC DIAGRAM
DIP and 20-Pin PLCC (28-Pin PLCC) Pinouts

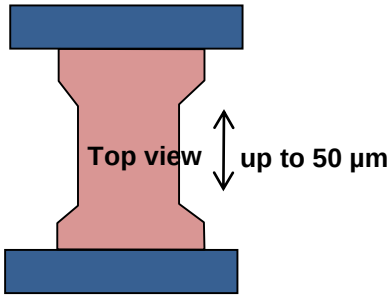


Physical Programming – Programmable Connections

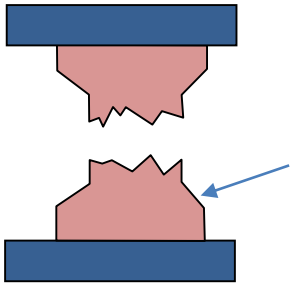
OTP (One Time Programmable)
- **Fuses**
- **Antifuses**

Reprogrammable
- **EEPROM**
- **Flash**
- **RAM**

Fuses



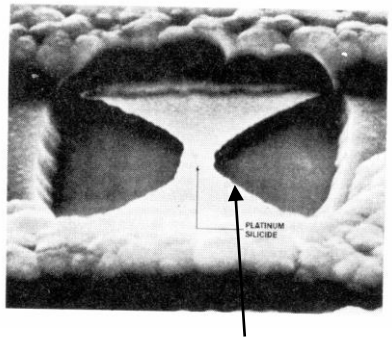
Fuse intact



Both sides must be disconnected forever, even at different temperatures, mechanical stress

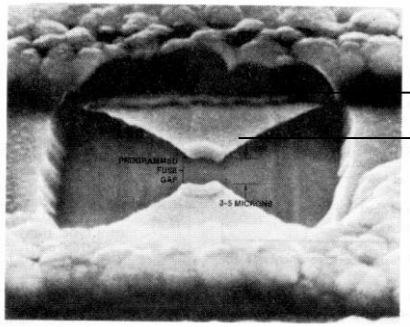
Programmed (Fuse blown)

Inefficient for large structures where
up to 98% of the fuses must become blown.



Unprogrammed

Platinum Silicide Pt-Si



Programmed (blown)

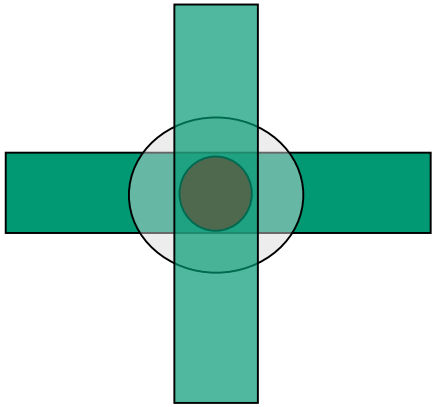
3 .. 5 µm

Scanning electron microscope photos from:
AMD (Advanced Micro Devices) PAL-Databook

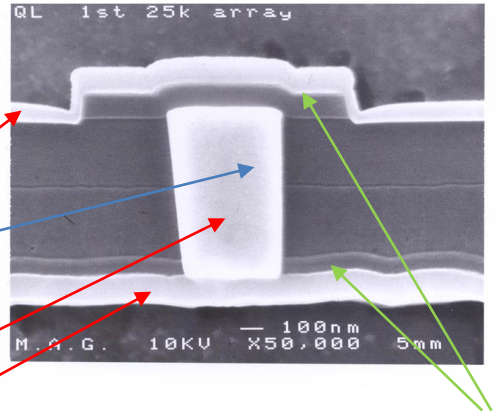
Antifuses

ViaLink antifuse (metal-metal) -> QuickLogic antifuse

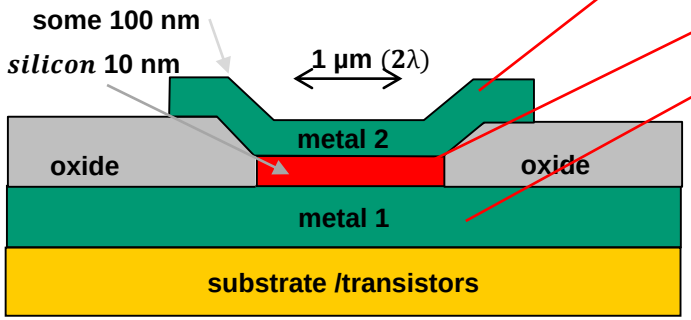
Top view

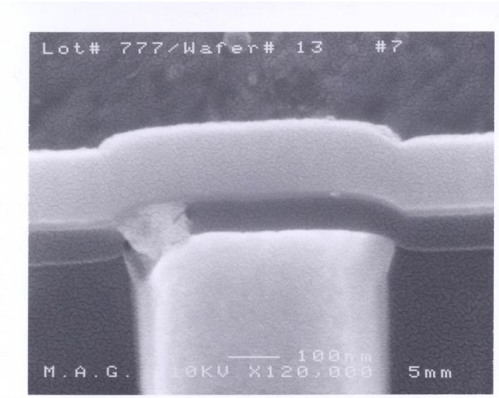


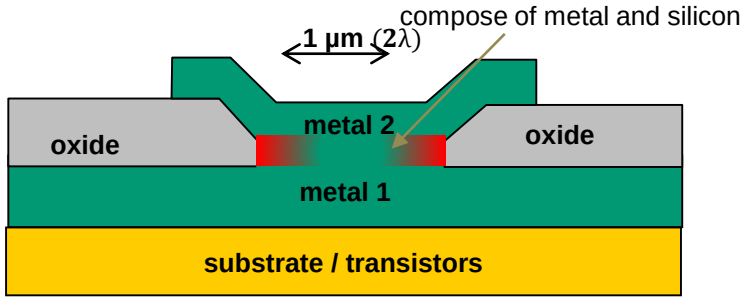
Modification:
Tungsten plug



Thin TiN (Titanium Nitride Ti_xN_y)-Layers are inserted to improve reliabilty





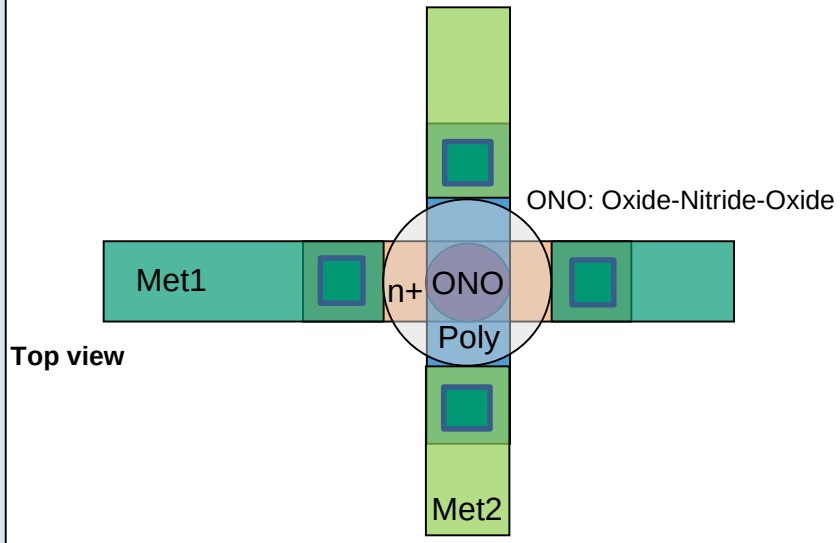


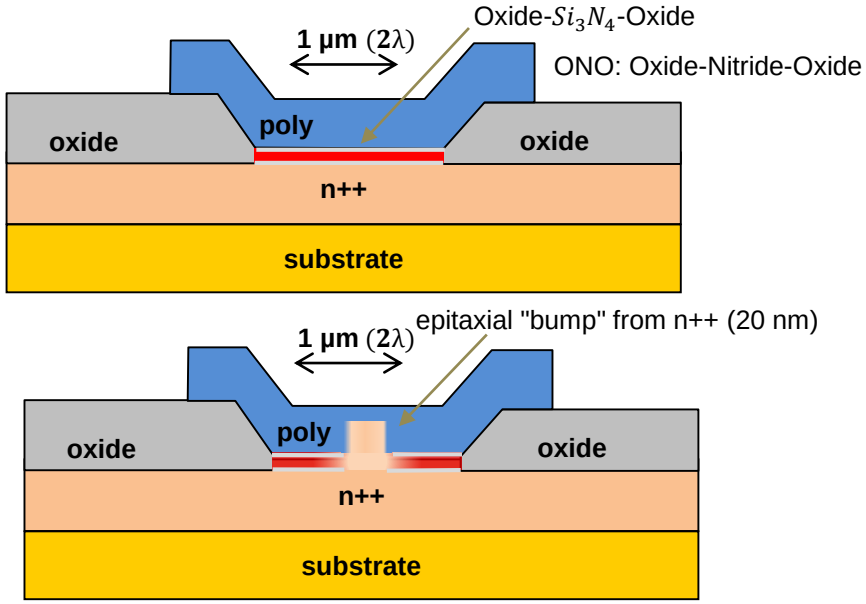
15 mA, resistance when blown about 20 .. 100 Ω

Pictures from: QuickLogic White Paper:
Reliabilty of Amorphous Silicon Antifuse, 2008

Antifuses

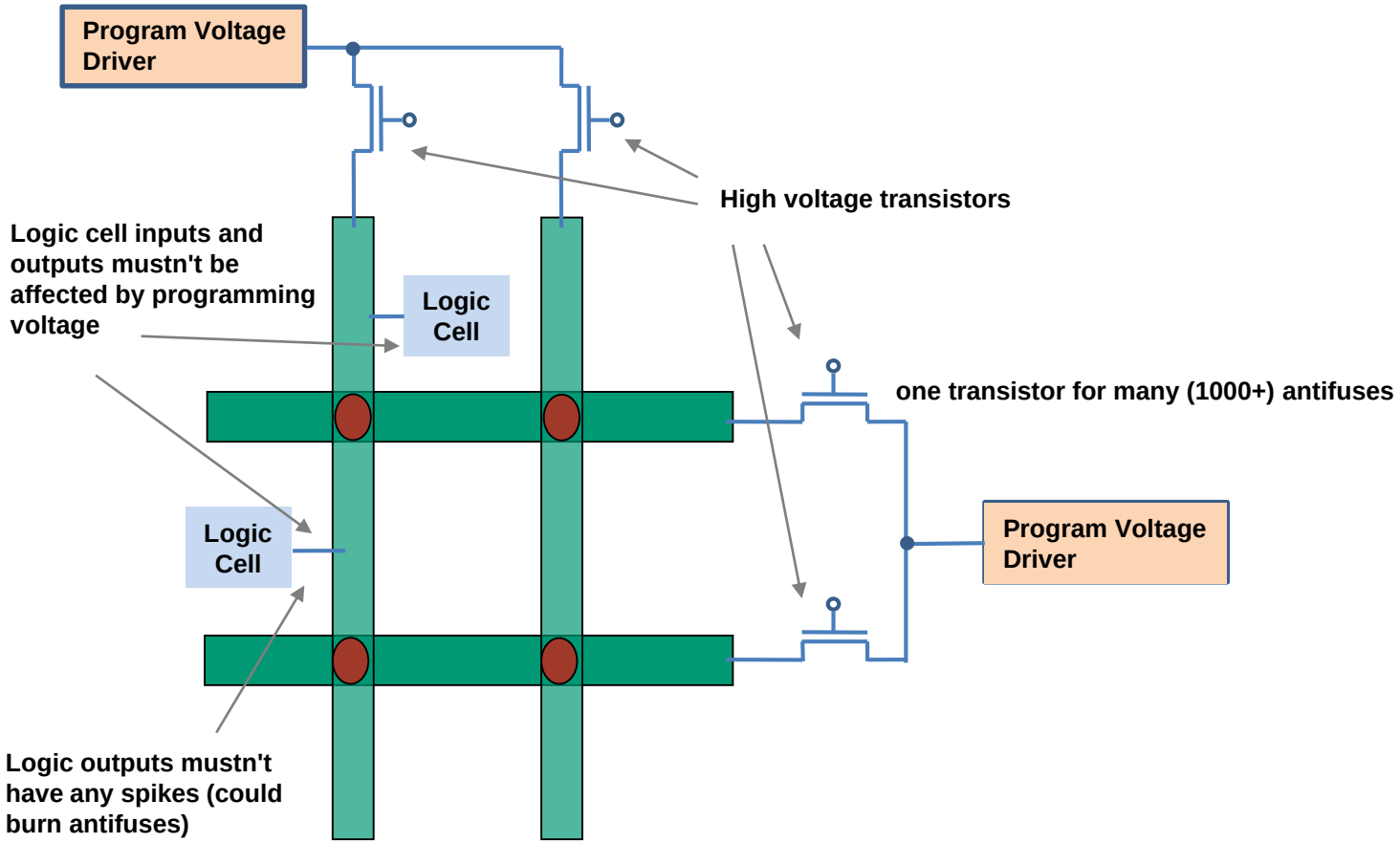
PLICE (programmable low-impedance circuit element) -> ACTEL antifuse



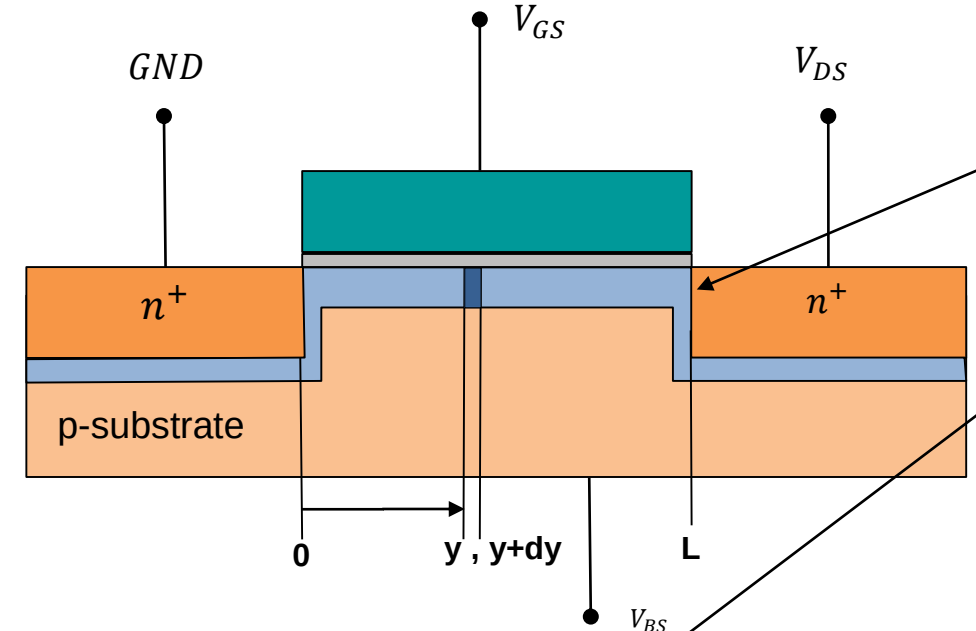


18 V, 5 mA, resistance blown 300 .. 1000 Ω
Up to some million antifuses, only some % blown)

Antifuse Programming



MOSFET: Control Equation



$$Q'(y) = C'_{OX}(V_{GS} - V(y) - V_{GS0})$$

Charge of gate capacitor per area $\Delta y \cdot W$

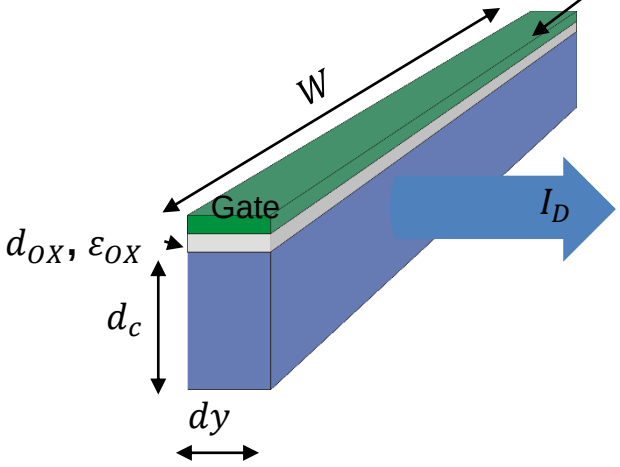
$$\rho = \frac{1}{e \cdot n \cdot \mu} = \frac{1}{\frac{Q'(y)}{d_c} \cdot \mu}$$

Charge in volume $\Delta y \cdot W \cdot d_c$

$$r_\square = \frac{\rho}{d_c} = \frac{1}{\mu \cdot Q'(y)}$$

$$dR = \frac{1}{\mu_n Q'(y)} \cdot \frac{dy}{W}$$

$r_\square$



$$dV(y) = I_D \cdot dR = \frac{I_D}{W \cdot \mu_n Q'(y)} \cdot dy$$

$$I_D \cdot dy = W \cdot \mu_n \cdot C'_{OX}(V_{GS} - V(y) - V_{th})dV(y)$$

$$\mu_n \cdot C'_{OX} = \mu_n \cdot \frac{\varepsilon_{OX}}{d_{OX}} = KP_n$$

Effective mobility in inverted channel less than mobility in table for n-Si : 450 vs. 1650 $\frac{cm^2}{Vs}$

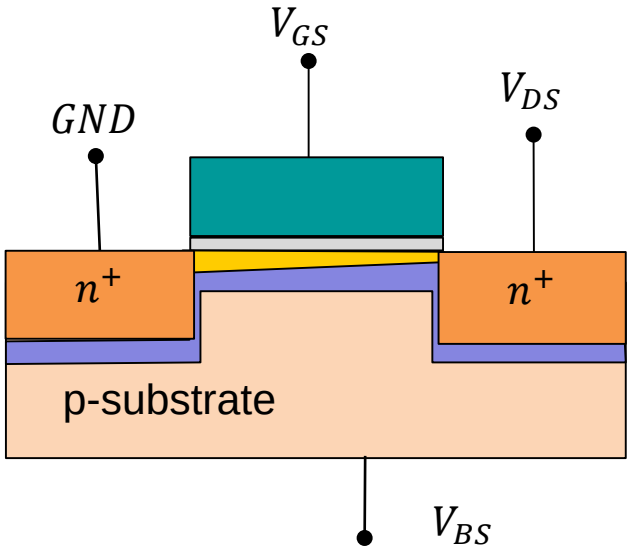
$$I_D \cdot dy = W \cdot KP_n \cdot (V_{GS} - V(y) - V_{th})dV(y)$$

MOSFET: Control Equation, non pinched off

$$I_D \cdot dy = W \cdot KP_n \cdot (V_{GS} - V(y) - V_{th}) \cdot dV(y)$$

$$I_D \int_0^L dy = W \cdot KP_n \int_0^{V_{DS}} (V_{GS} - V_{th} - V(y)) \cdot dV(y)$$

$$I_D = \frac{W}{L} \cdot KP_n \cdot \left[(V_{GS} - V_{th}) V_{DS} - \frac{V_{DS}^2}{2} \right] = \beta \cdot \left[(V_{GS} - V_{th}) V_{DS} - \frac{V_{DS}^2}{2} \right]$$

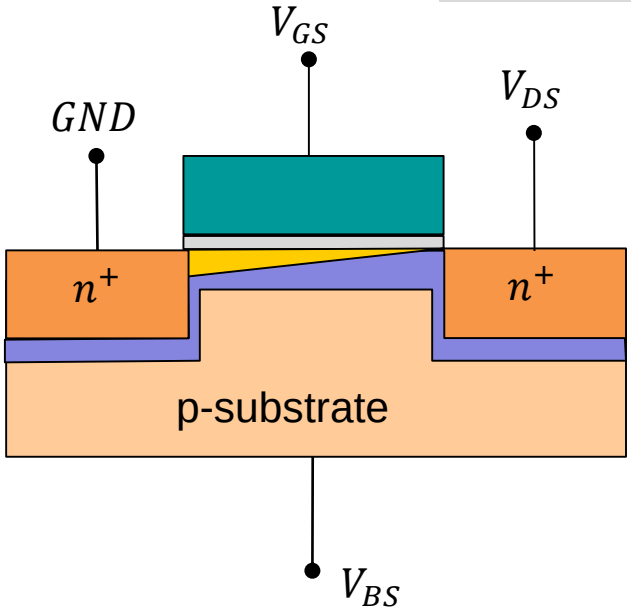


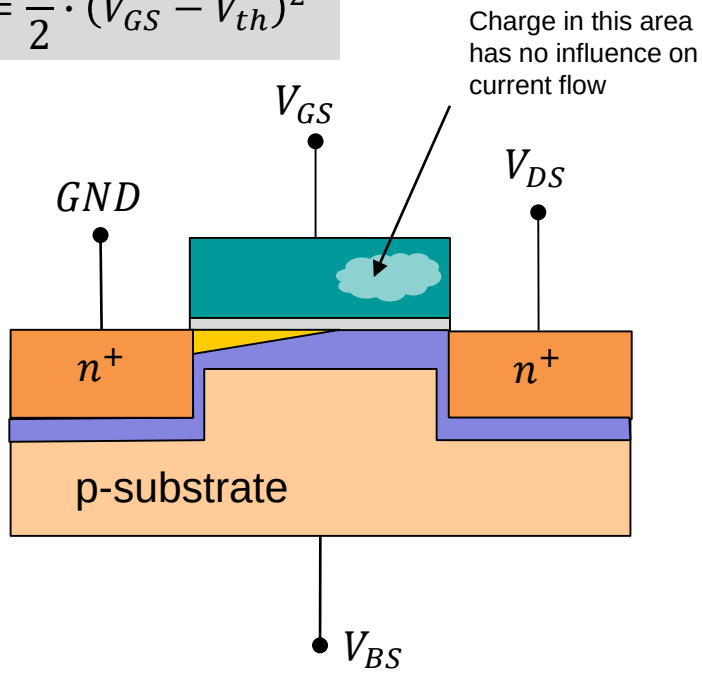
MOSFET: Control Equation, pinched off

$$I_D = \frac{W}{L} \cdot KP_n \cdot \left[(V_{GS} - V_{th}) U V_{DS} - \frac{V_{DS}^2}{2} \right]$$

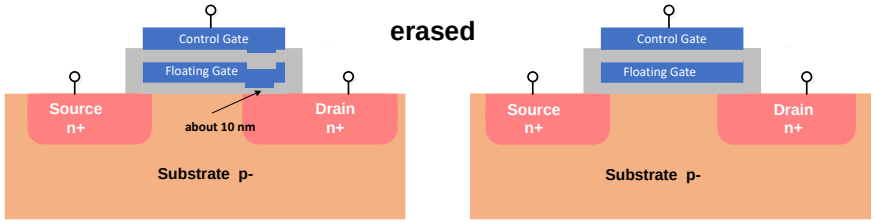
Pinch off when: $V_{GS} - V_{th} \leq V_{DS}$

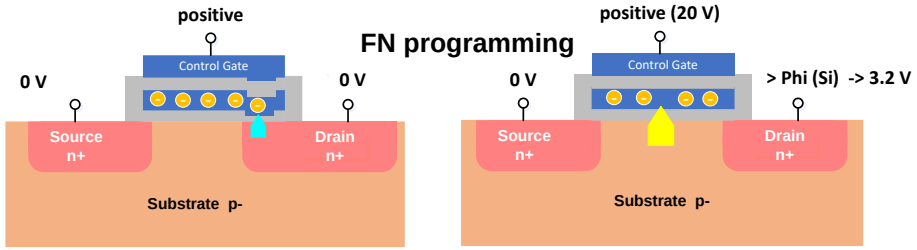
$$I_D = \frac{W}{L} \cdot KP_n \cdot \frac{(V_{GS} - V_{th})^2}{2} = \frac{\beta}{2} \cdot (V_{GS} - V_{th})^2$$

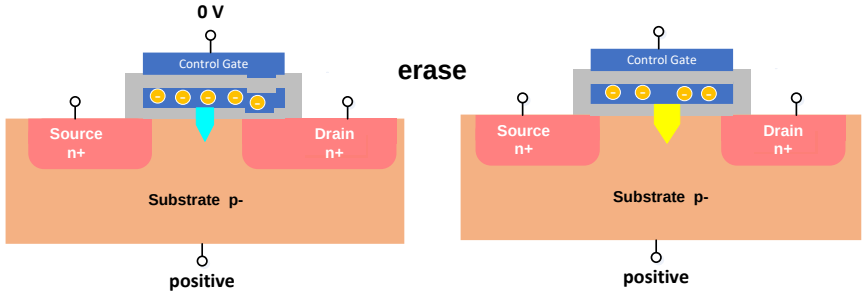


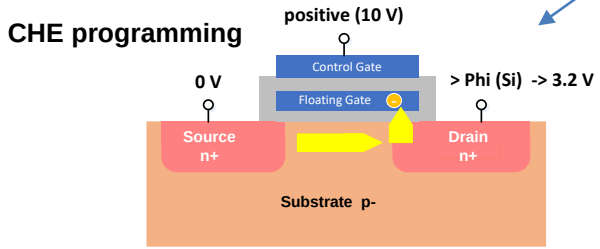


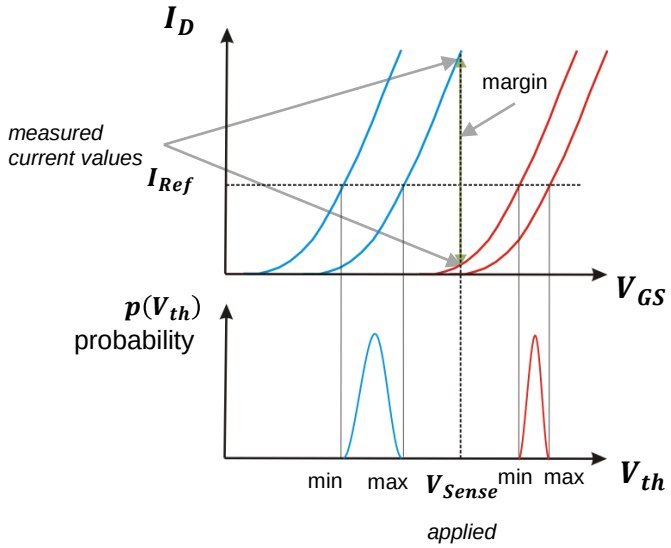
EEPROM and FLASH-cells









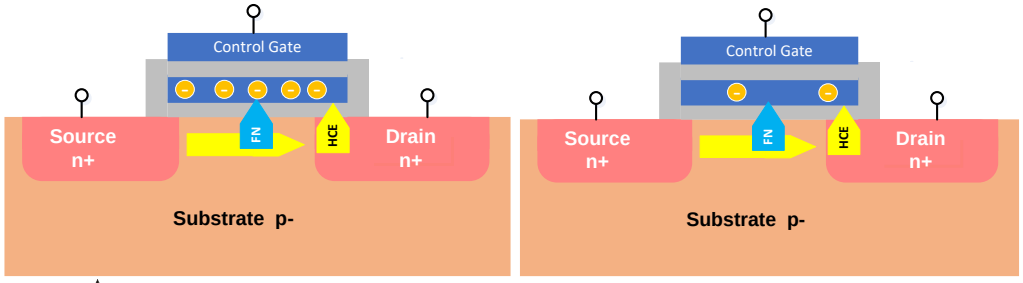


- Thin oxide, floating gate is conductor (Poly-Si)
- Electrons on the floating gate increase the threshold voltage
- FLOTOX cell (**FLO**ating gate **T**hin **OX**ide)
- **F**owler-**N**ordheim tunnelling: FN, programmming voltage about 20 V, slow (ms)
- Alternative Programming by hot electrons: **C**hannel **H**ot **E**lectrons : CHE,
- Fast (µs), needs a current through channel to generate hot electrons at the end

Challenge: Single Event Upset (SEU) -
"Softerrors" provoked by high energy particles

FLASH-cells with 2 and 3 bits

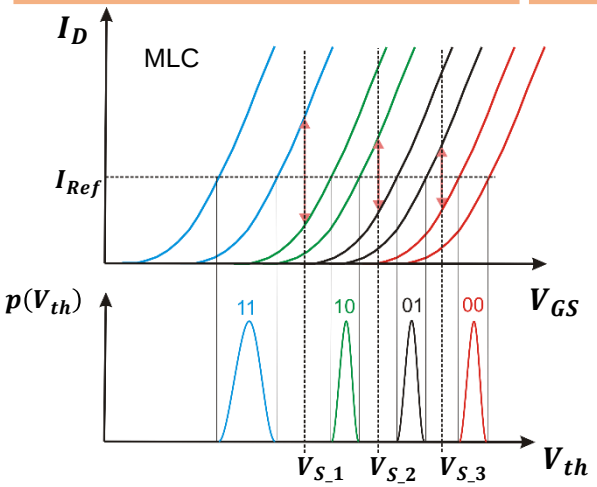
V_{S_2}

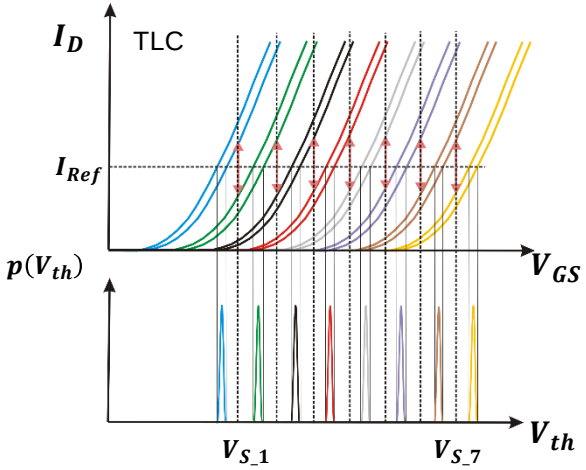


Different amount of charge results in different threshold voltages.

Common naming scheme:

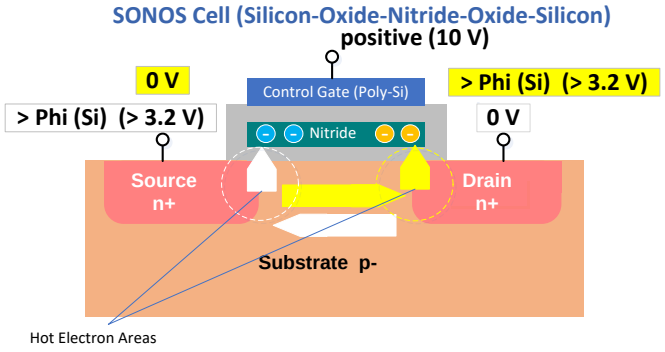
MLC- 2 bits (4 voltage levels)
TLC- 33 Bits (8 voltage levels)

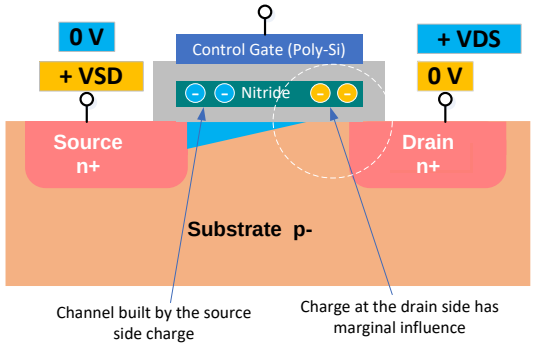




Confusion:
TLC: "Three Level" means 3 bits but uses 8 voltage levels

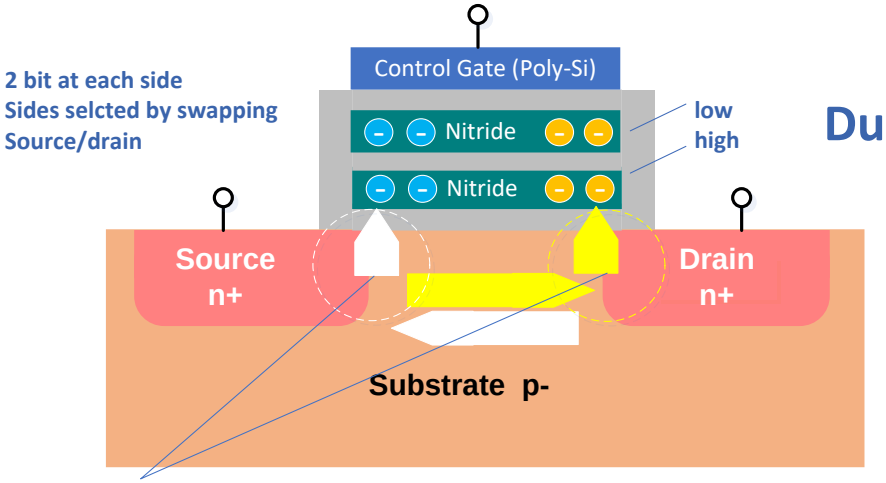
MLC (**Multi**-..) used for 2 bit cells today





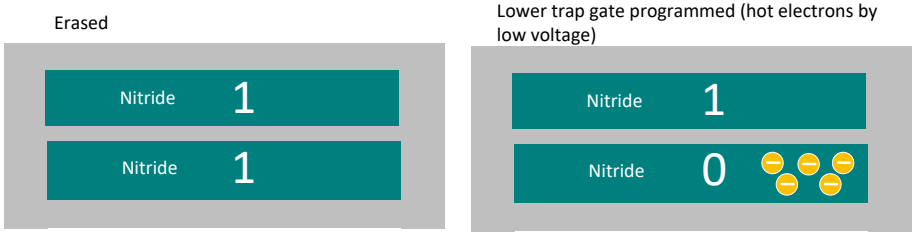
- **Gate is insolator with local charge traps**
- **Material Si_3N_4 can hold electrons/holes in crystal defects**
- **charges do not move on the gate**
- **Source side has main influence for channel building**
- **Alternate use in forward and reverse mode**

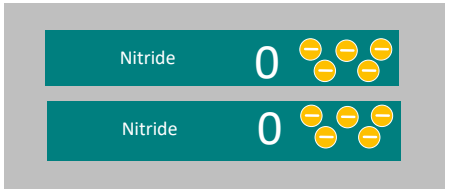
FLASH-cells with 2 floating gates

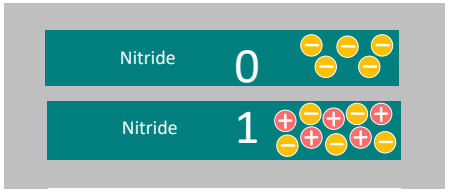


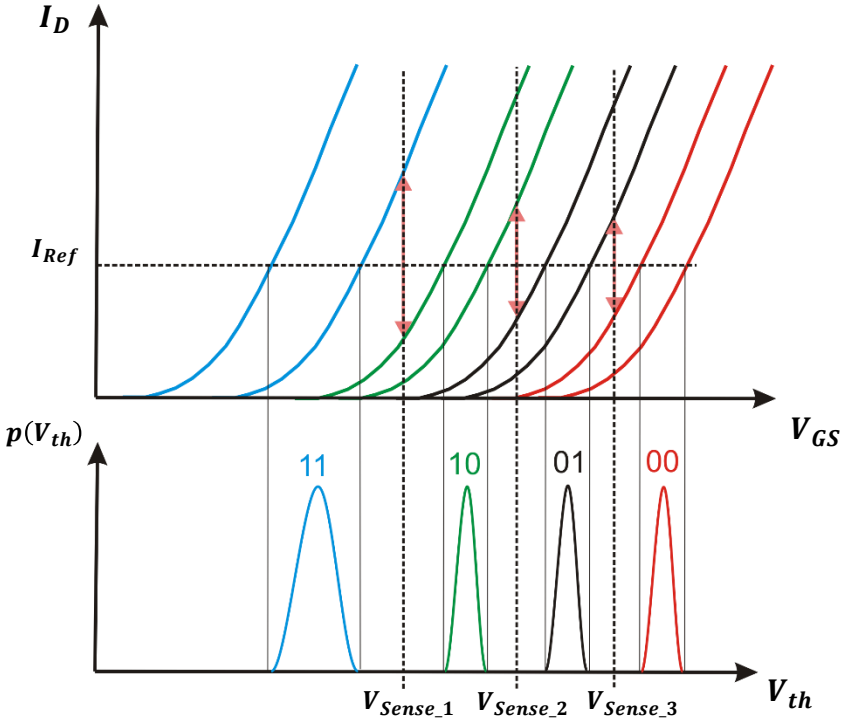
Dual SONOS Cell (Silicon-Oxide-Nitride-Oxide-Silicon)

- **2 gates offer 4 different threshold voltages per side (2 bits)**
- **Using forward and reverse operation offers 4 bits.**

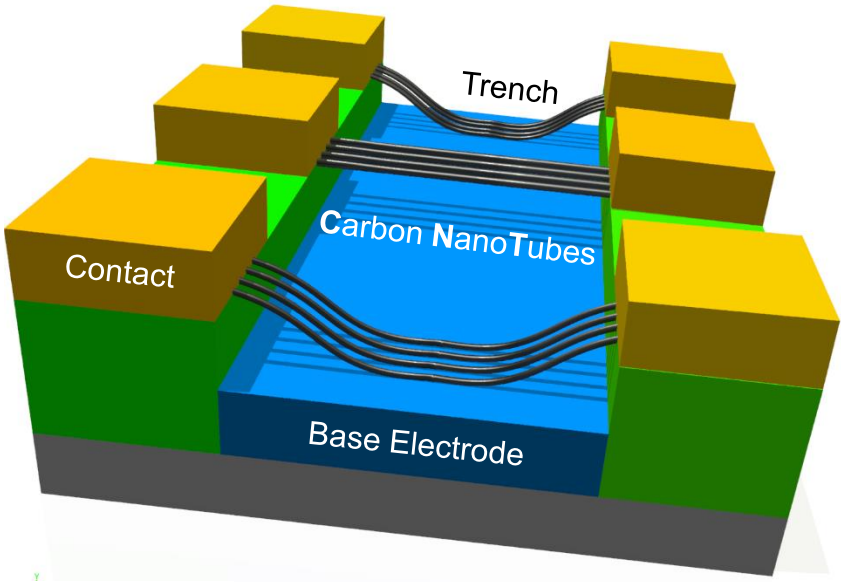








Nanotube RAM

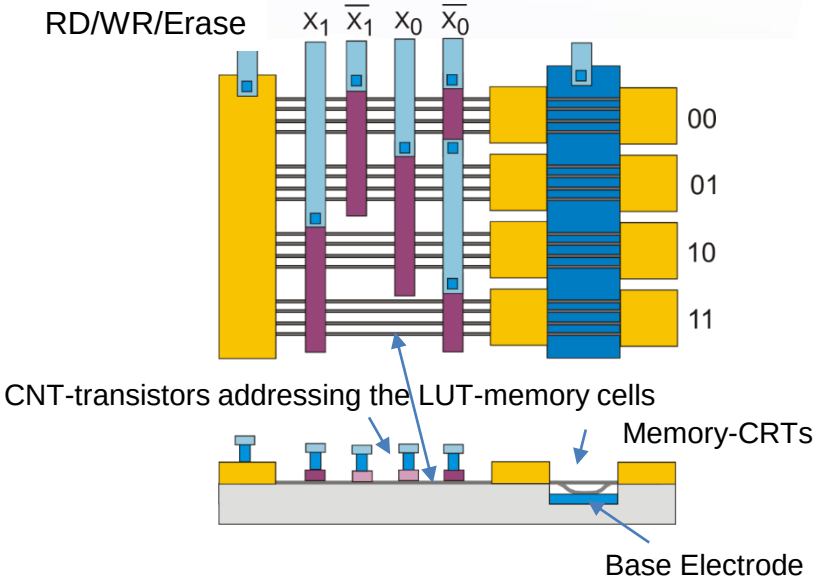


Carbon Nanotubes: Diameter <1 nm to some 10 nm

Nanomechanical memory

2 stable states:
1. Elastic stability holds the nanotubes over the trench/gap
   – no connection to base electrode
2. Van der Waals force holds the nanotubes at the base electrode
   – stable connection to base electrode

Switching by electrostatic forces involved
by base electrode voltage

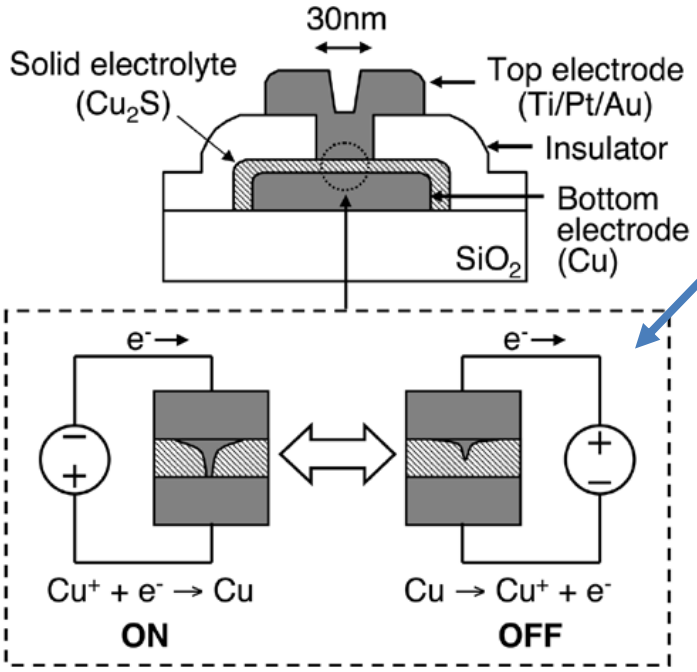


Lookup Table (LUT) representing 2 bit truth table
using the CNT-RAM and CNT-trransistors

Problems:
Nanotube production not integrated in Si-process
Must be "placed" – printing, electrostatic manipulator ?
reliability problems

Solid Electrolyte Nanoswitches

Presented about 2005*



Source: dto. *

Cu-ions in copper sulfide are reduced to copper
forming a conducting bridge by applying a voltage
Reverse voltage can dissolve the copper (with current over R_{on})

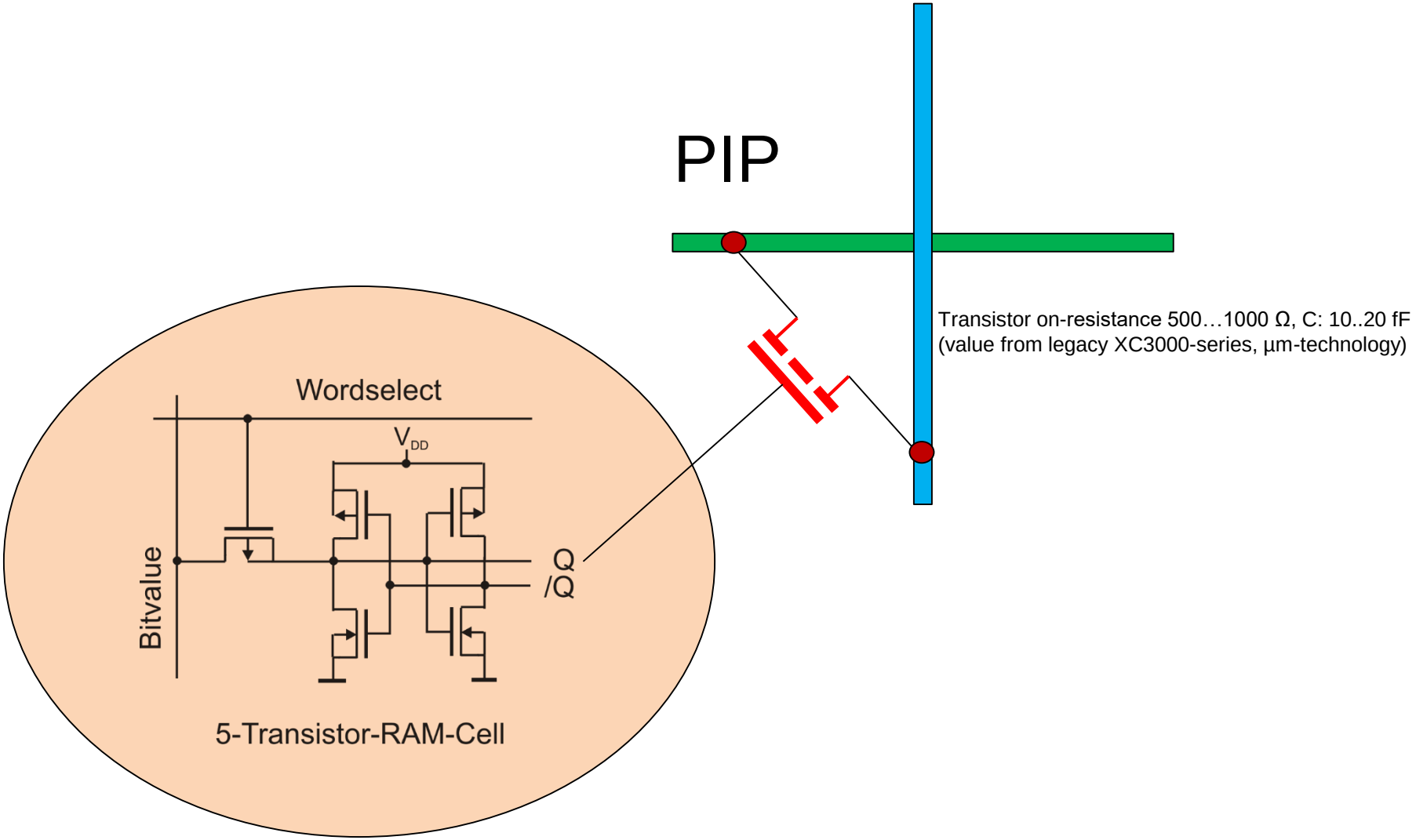
$R_{on}: downto\ 50\Omega$
Programming time 0.1 .. 10 ms
Small (2λ x 2λ) where λ is half of minimal structure

Problems: Small voltage of 0.2 V for both directions of configuration
- > must not be reached in normal operation
Short retention times with bias voltage

Later articles describe more reliable implementations using other materials in the sandwich
Usage in real FPGAs ?

* Shunichi Kaeriyama, *et al.* A Nonvolatile Programmable Solid-Electrolyte Nanometer Switch, IEEE JOURNAL OF SOLID-STATE CIRCUITS, VOL. 40, NO. 1, JANUARY 2005, pp168 ff.

RAM Cell with Transistor Switch as Programmable Interconnect Point



GAL

Generic Array Logic

Name given by Lattice Semiconductor Corporation

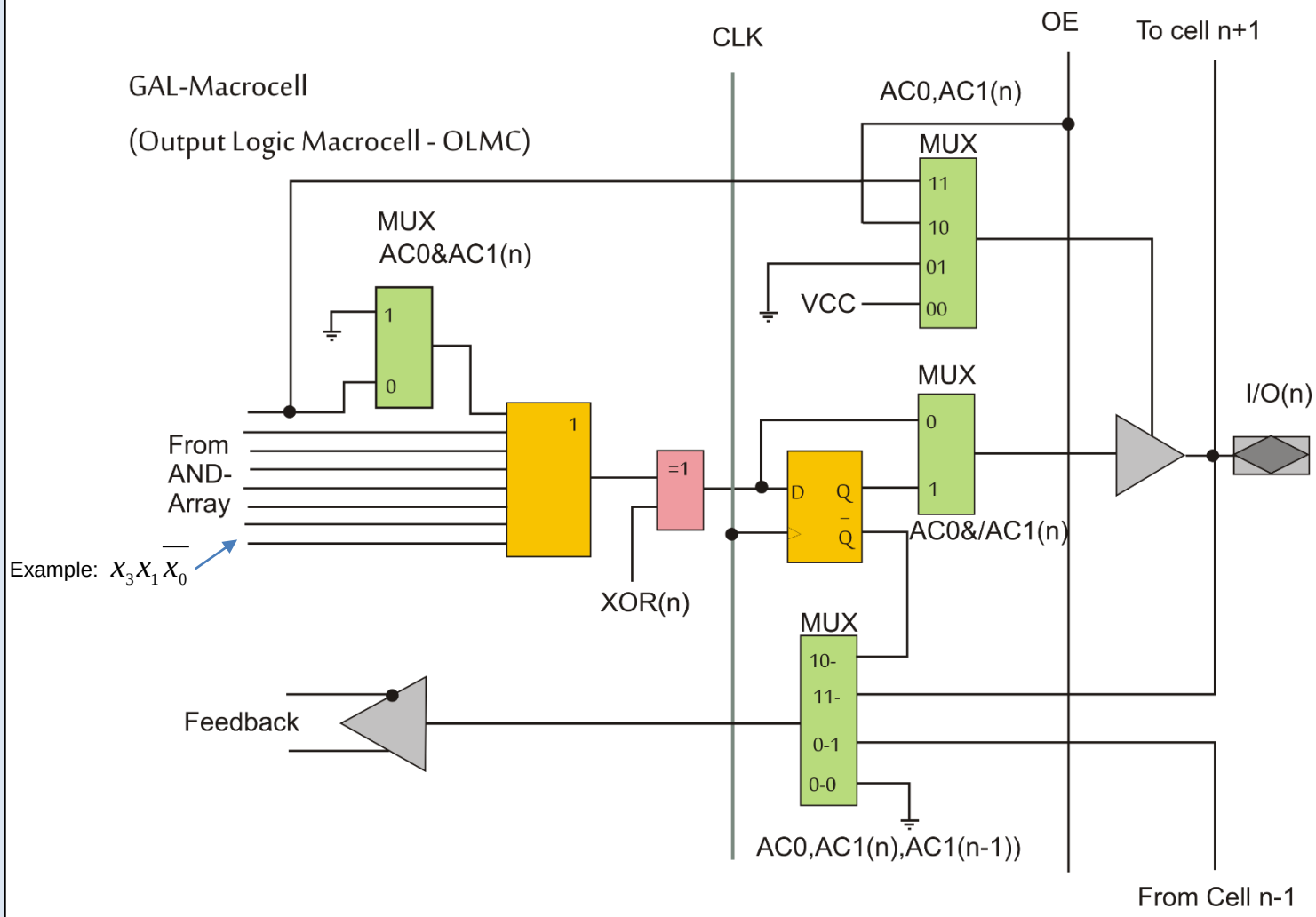
Logic Array with PAL-Structure **(fixed OR-Field)**
Variable Output Cells
- Combinatorial <-> Sequential
- Low active <-> High active
- Programmable Output Enable

Some Devices with
- **Product term sharing**
- **Buried logic cells**
- **Input registers**

One Clock
EEPROM Programming

Because of their limitations (size, power) GALs do not play any role in practical design today, but they are stiil a good example to introduce programmable logic

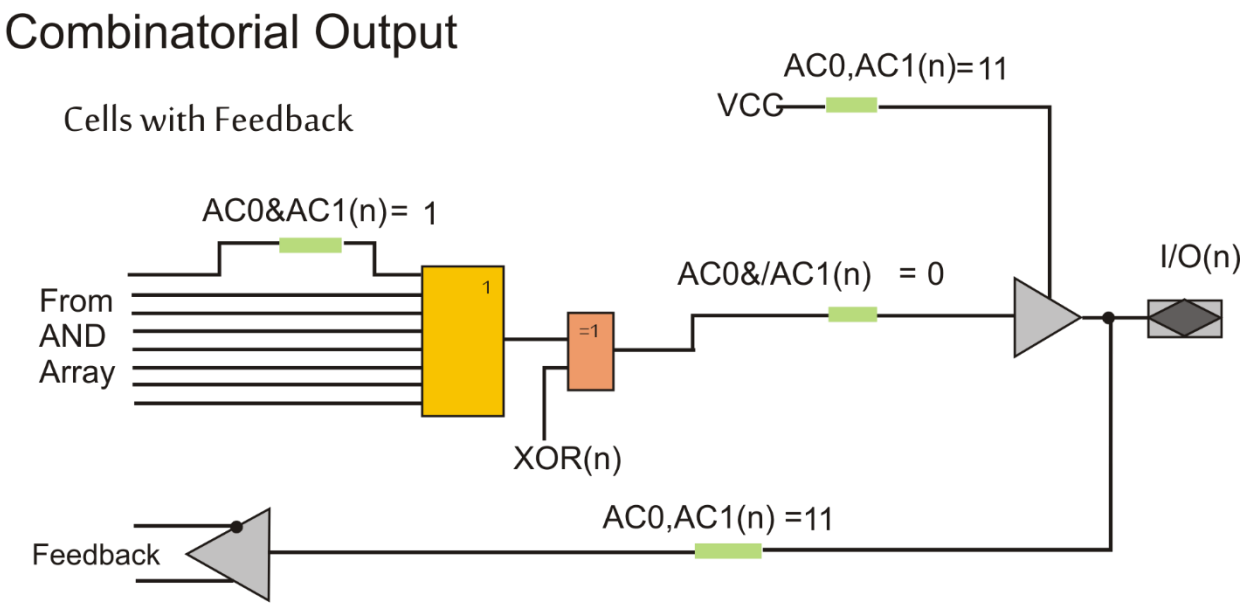
GAL Macrocell

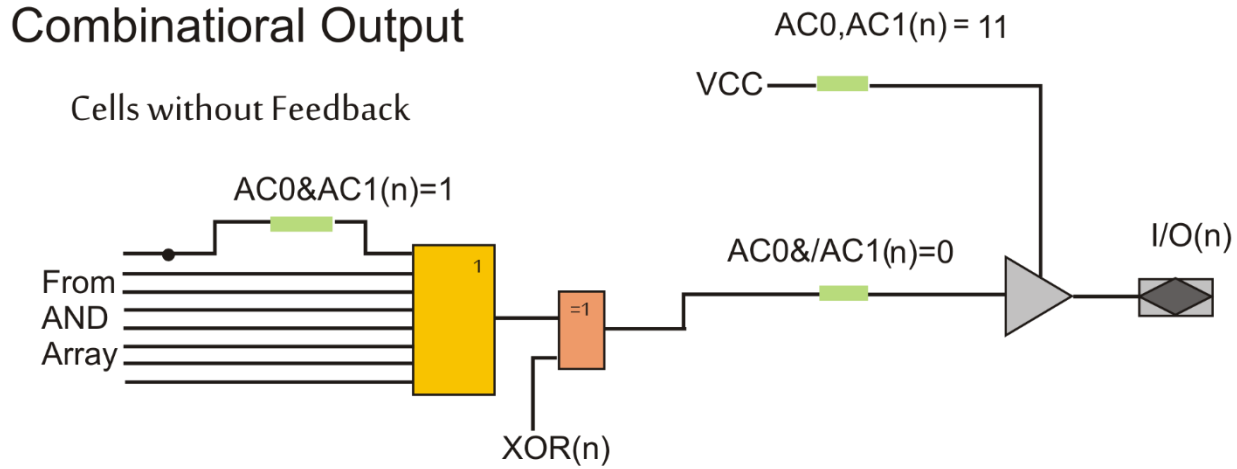


XOR as a programmable Inverter

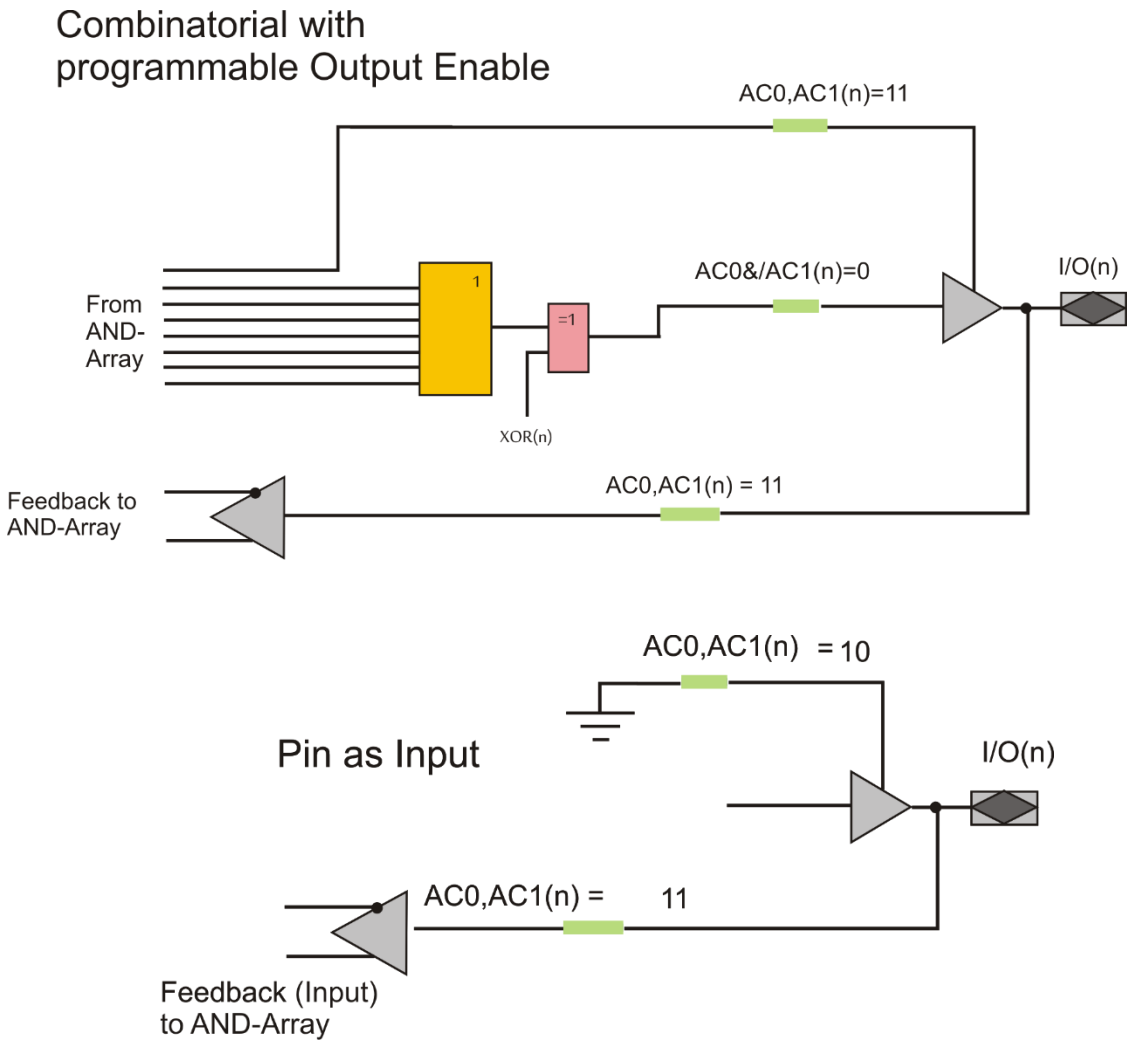
XOR(n)	In	Out
0	0	0
0	1	1
1	0	1
1	1	0

GAL OLMC Combinatorial

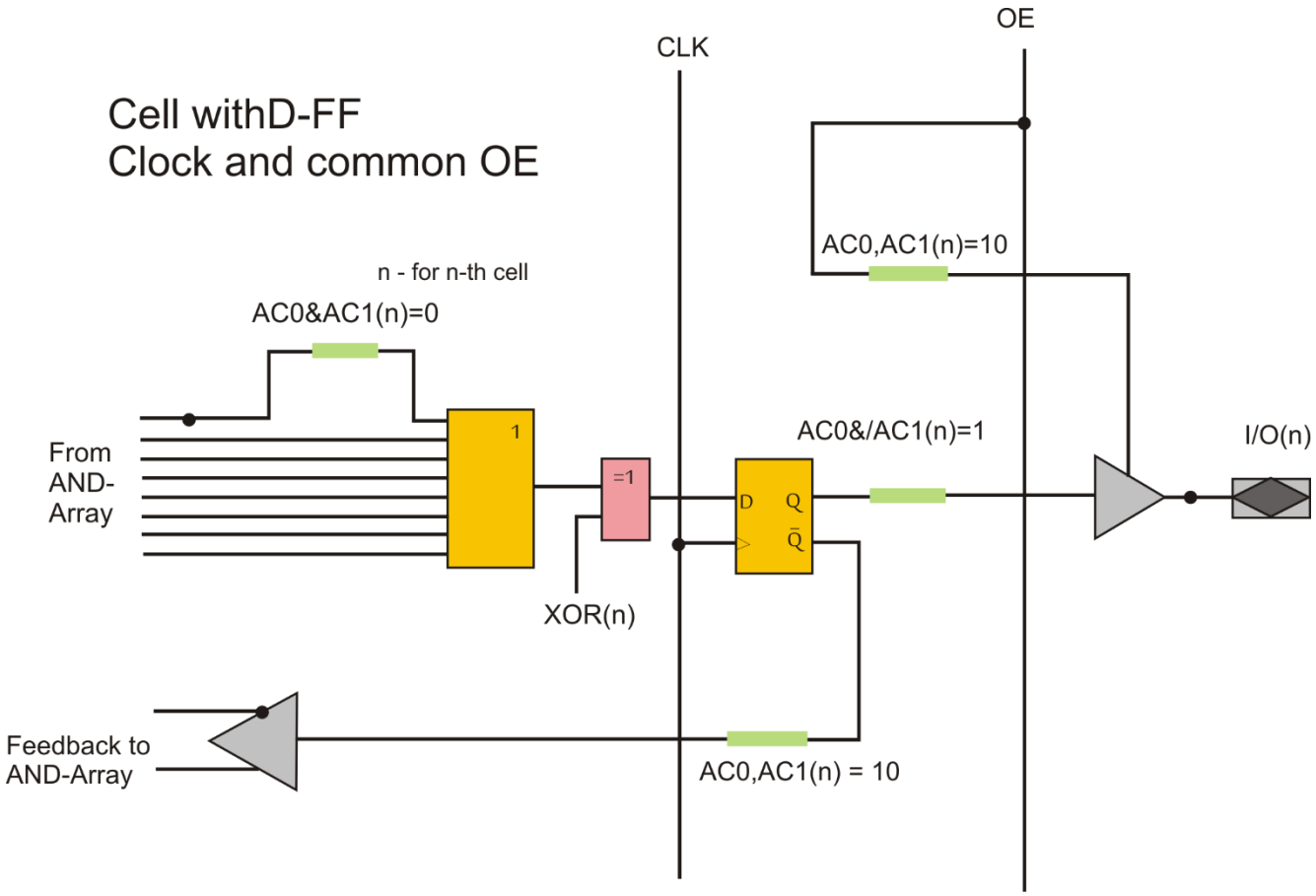




GAL OLMC Combinatorial with OE



GAL, OLMC with **Flip-Flop** and OE



GAL Programming 1 (classical)

Advanced Devices: isp (in system programmable) -> separate Pins for JTAG programming

EEPROM-cells are addressed as raws
Example GAL 16V8

1 – connection
0 – no connection

Raw 0 …31	**64 Bits Logic each (AND-Field)**

16 Inputs x 2 (inverted/not inverted) x 8 Products x 8 Outputs

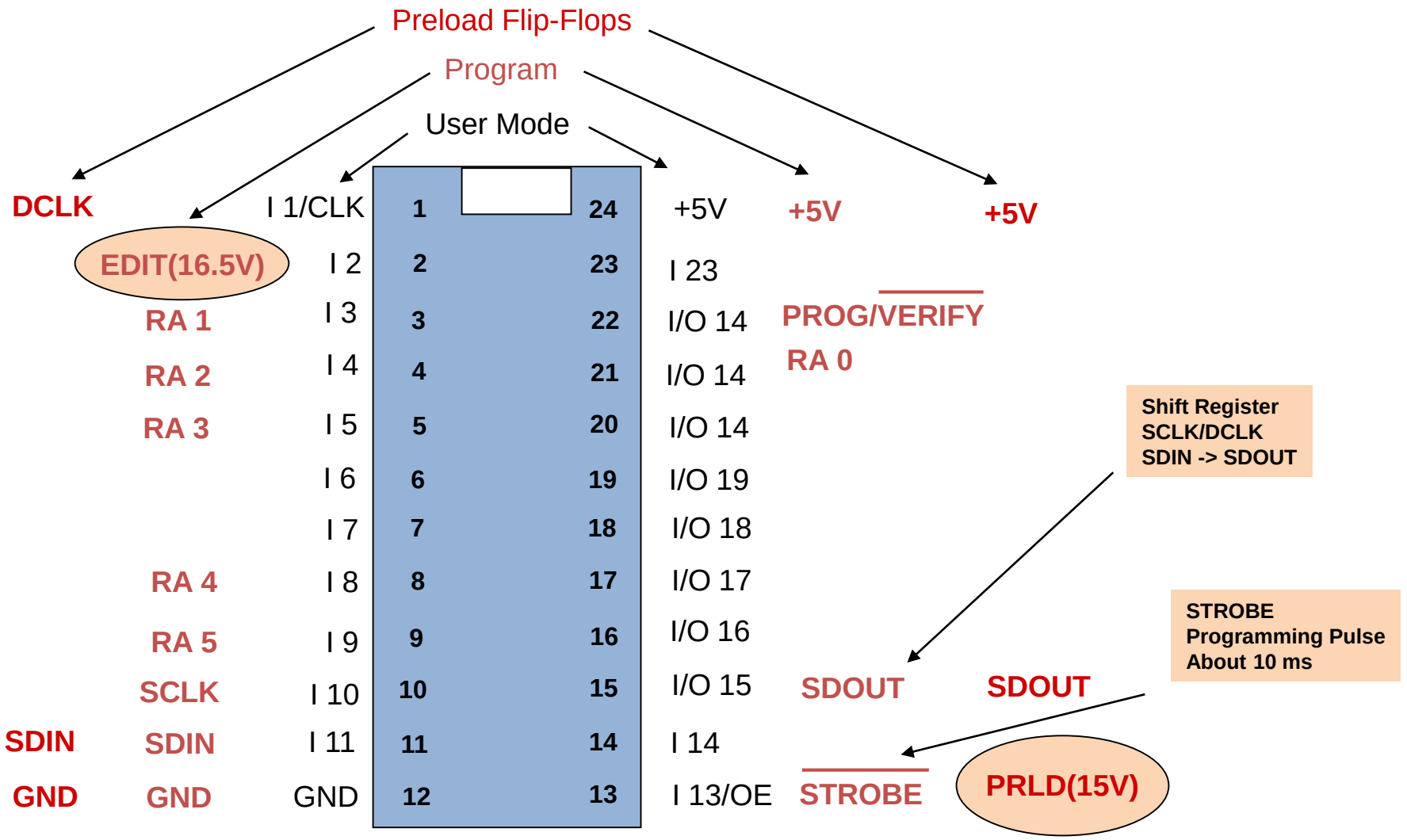
Raw 32	**64 Bits signature (User information like version etc.)**
Raws 33 … 59	**not used**
Raw 60	**Architecture Control Word (ACW)** **64 Bits product term enable (Reduction of power consumtion)** **8 Bits XOR(n) -> programmble output inverter** **8 Bits AC1(n)** **1 Bit SYN** **1 Bit AC0** **-> 82 Bits !**
Raw 61	**Safety bit (if set no read of programmed structure possible, but reprogramming is still possible)** **-> Design security**
Raw 62	**not used**
Raw 63	**Bulk Erase (Erase all information)**

GAL: Programming and Cell-Preload

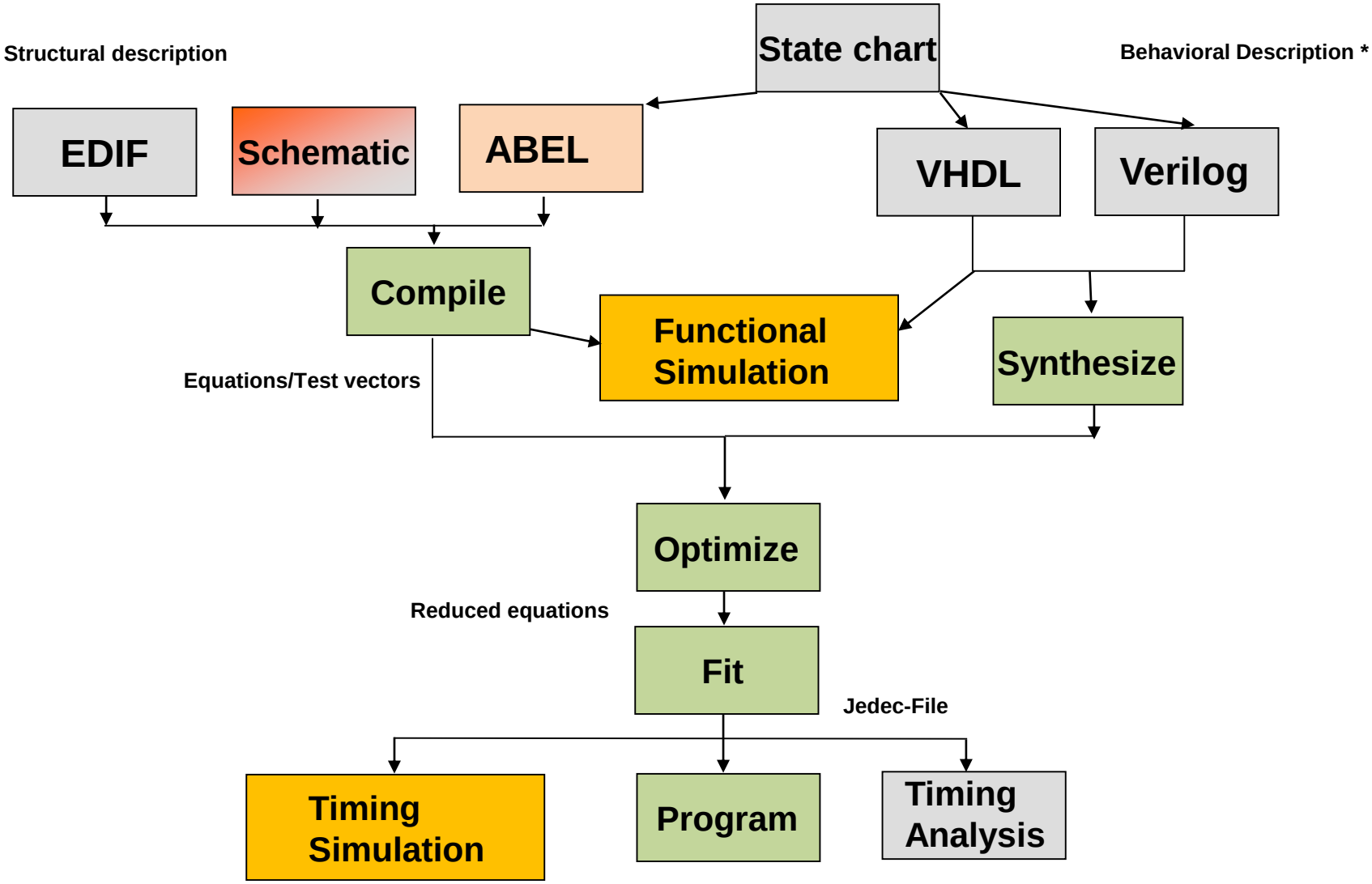
3 modes of operation:

- **normal/user mode**
- **programming mode**
- **test pattern read/write mode**

Conventional method, newer devices: JTAG



Design Flow for PLD, CPLD

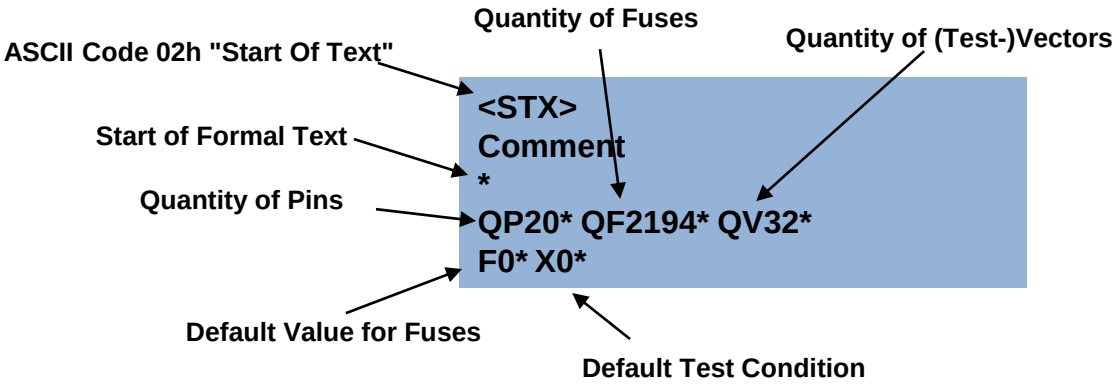


Jedec File 1

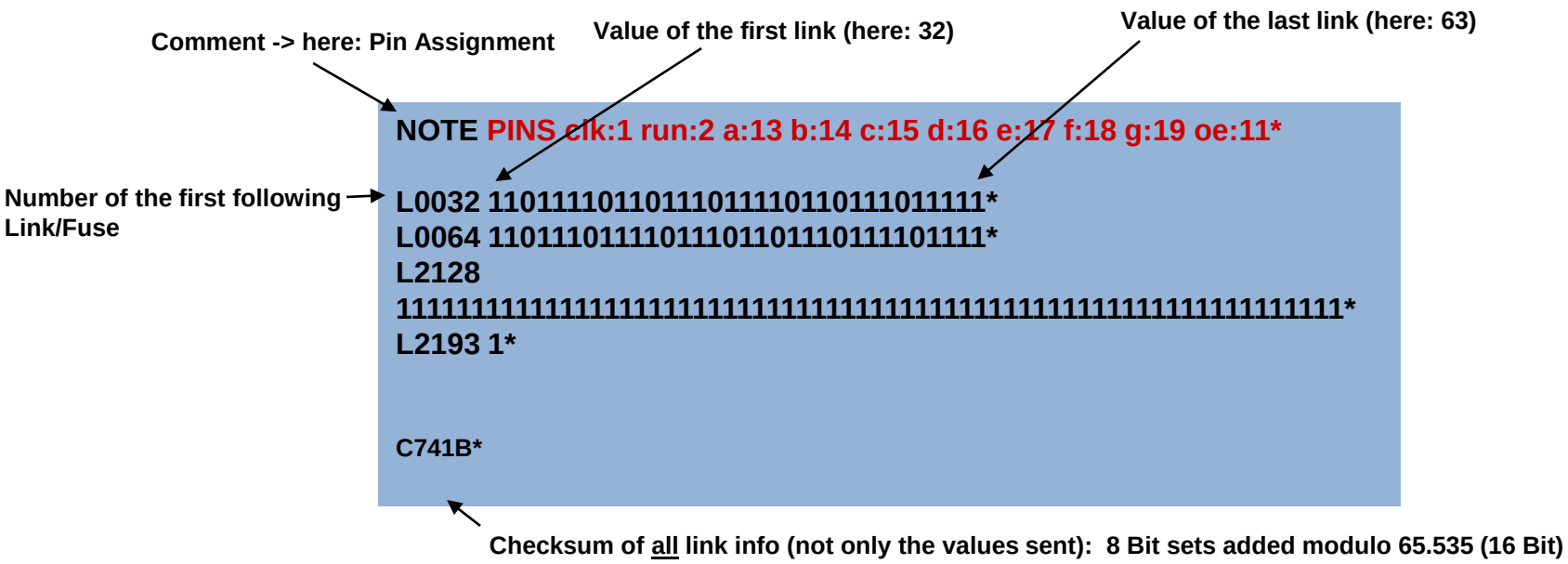
JEDEC (Joint Electron Devices Engeneering Council) since 1958 ->
Standardization Body of EIA (Electronic Industries Alliance)

Standardization started 1924 with Radio Manufacturers Association (RMA)
Electronic vacuum tubes

JESD3-C "Standard Data Transfer Format Between Data Preparation System and Programmable Logic Device Programmer" 1994



Jedec File 2



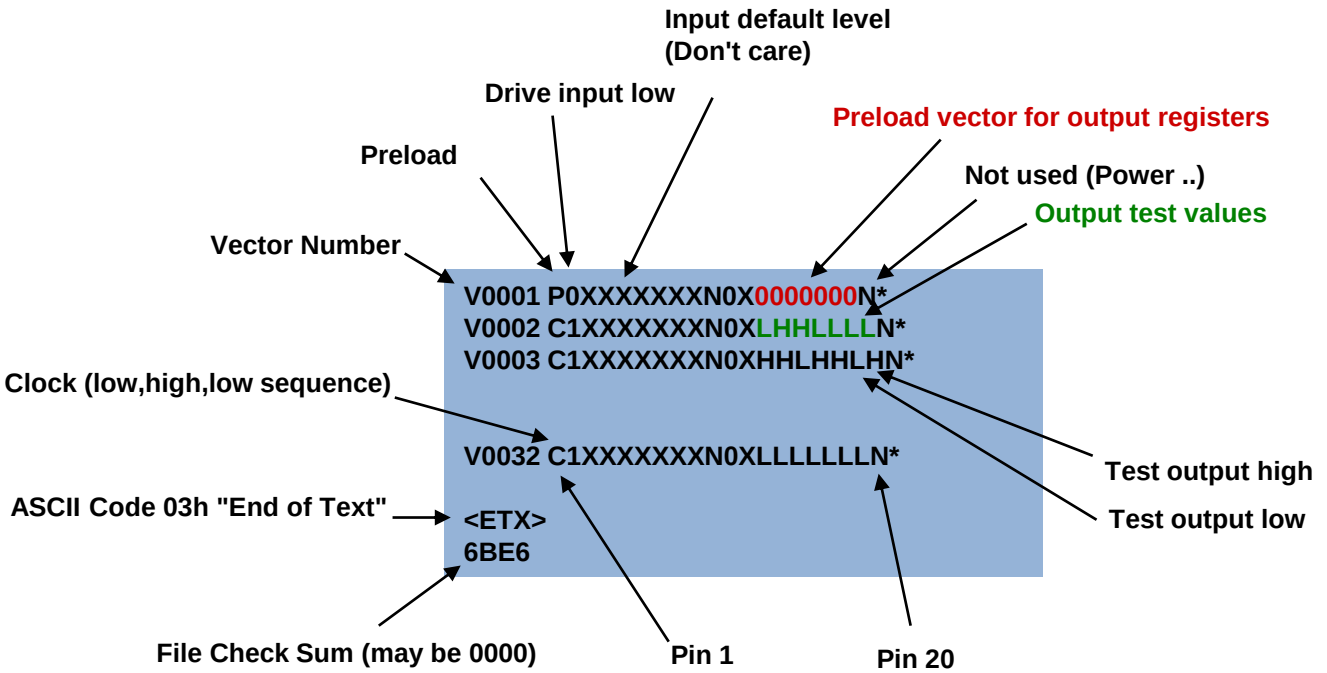
Number of luses/links after "L" can vary from 1 to all fuses of the device.
A sigle link list is closed by *
If a link is not mentioned it gets the value from the F-field

G Security Fuse (1- enabled)
A Access time for test vector s in ns

S Starting vector for signature analysis
R Remainder for signature analysis
T Test cycles

Jedec File 3

Test Vectors (Number according to QV-field)



B Buried Register Preload -> see standard
F Float input or output
K Clock (high,low,high sequence)
T Observer Registers -> see standard
Z Input or test high impedance

2..9 Drive input to super voltage 0..9
(before : QE3 10500* super voltage 3 is 10.5 volts)

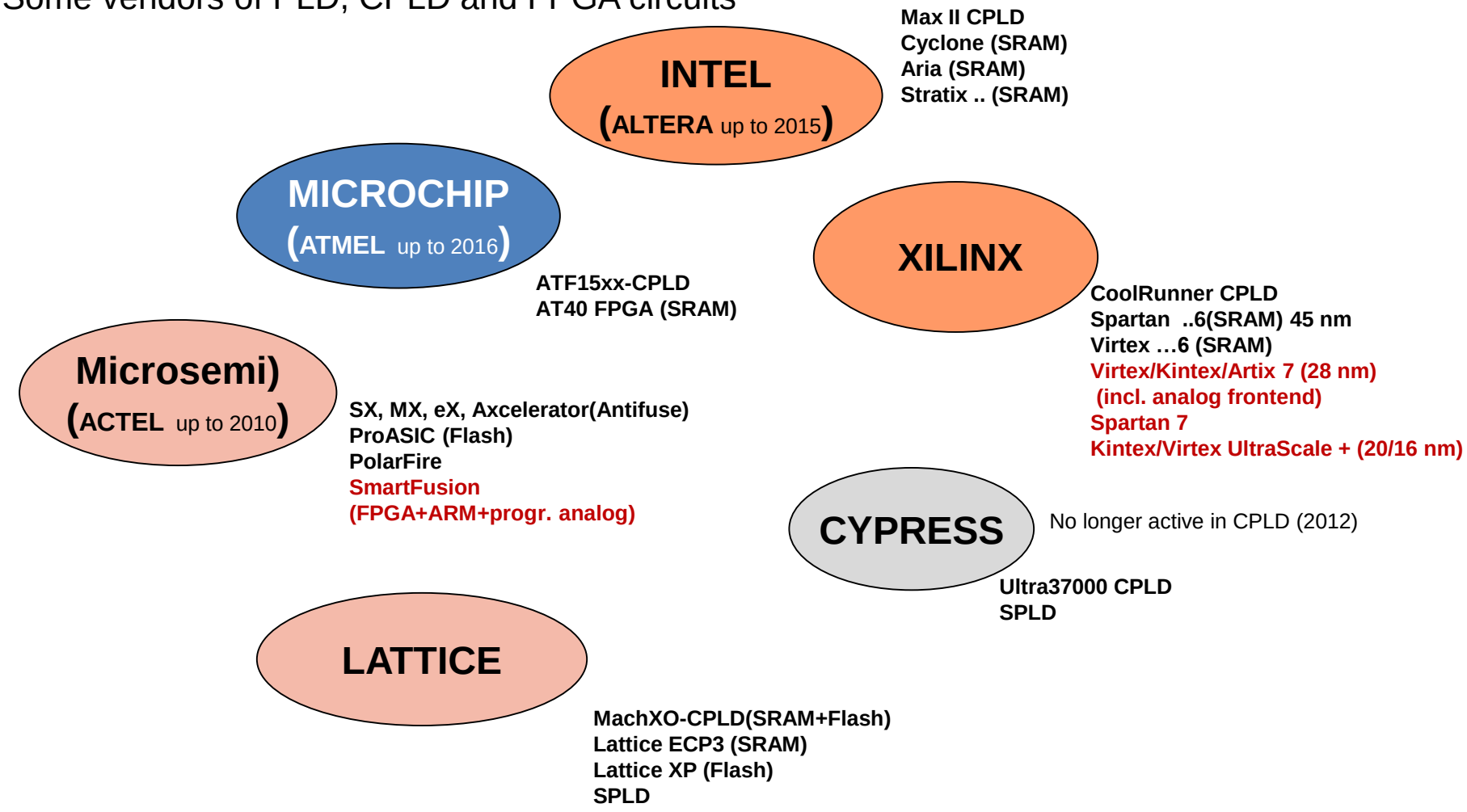
Jedec Example

<STX>
*
QP20* QF2194* QV32* F0*
X0*
NOTE Die implemented in GAL20V8*
NOTE PINS clk:1 run:2 a:13 b:14 c:15 d:16 e:17 f:18 g:19 oe:11*
L0000 01101110111011101101110111101111*
L0032 11011110110111011110110111011111*
L0064 11011101111011101101110111101111*
L0096 10011101111111011101111011011111*
L0128 11011101111011011101111011011111*
L0160 11011110111011011101110111011111*
…..
L1600 10011101111111011101111011011111*
L1632 11011101111011011101111011011111*
L1664 01011101111011101101110111101111*
L1696 10011101111011011101110111011111*
L2048 11111110*
L2128 11111111111111111111111111111111111111111111111111111111111111111111*
L2193 1*
V0001 P0XXXXXXXN0X0000000N*
V0002 C1XXXXXXXN0XLHHLLLLN*
V0003 C1XXXXXXXN0XHHLHHLHN*
V0004 C0XXXXXXXN0XHHLHHLHN*
……..
V0030 C1XXXXXXXN0XLLLLLLLN*
V0031 P0XXXXXXXN0X0100000N*
V0032 C1XXXXXXXN0XLLLLLLLN*
C741B*
<ETX>
6BE6

GAL16V8 Programming File

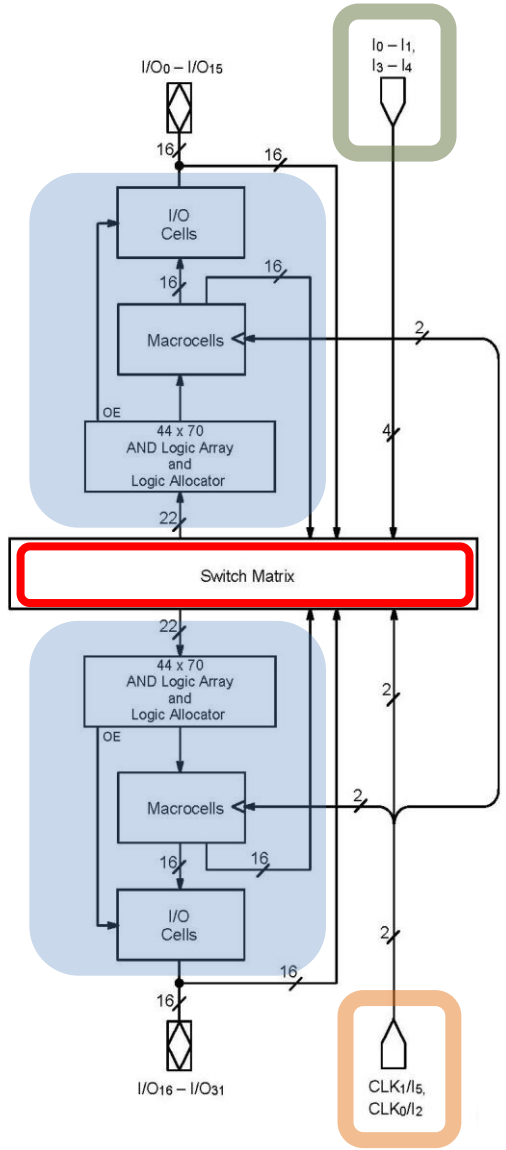
CPLD – Complex Programmable Logic Devices

Some vendors of PLD, CPLD and FPGA circuits



CPLD-Example: Lattice MACH 110

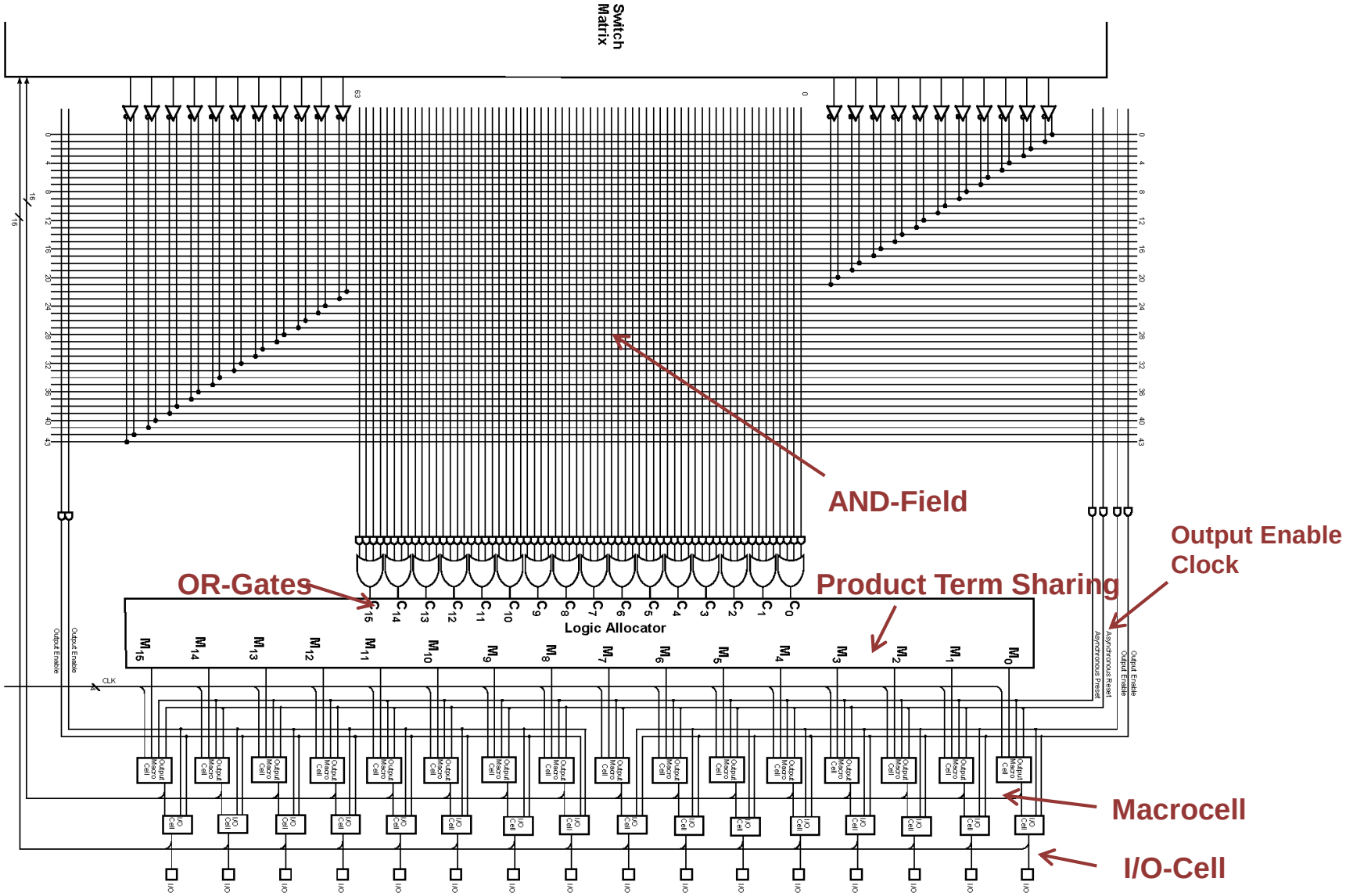
GAL-like blocks connected by a switch matrix



I/O and clocks can be assigned to cells using the connection matrix

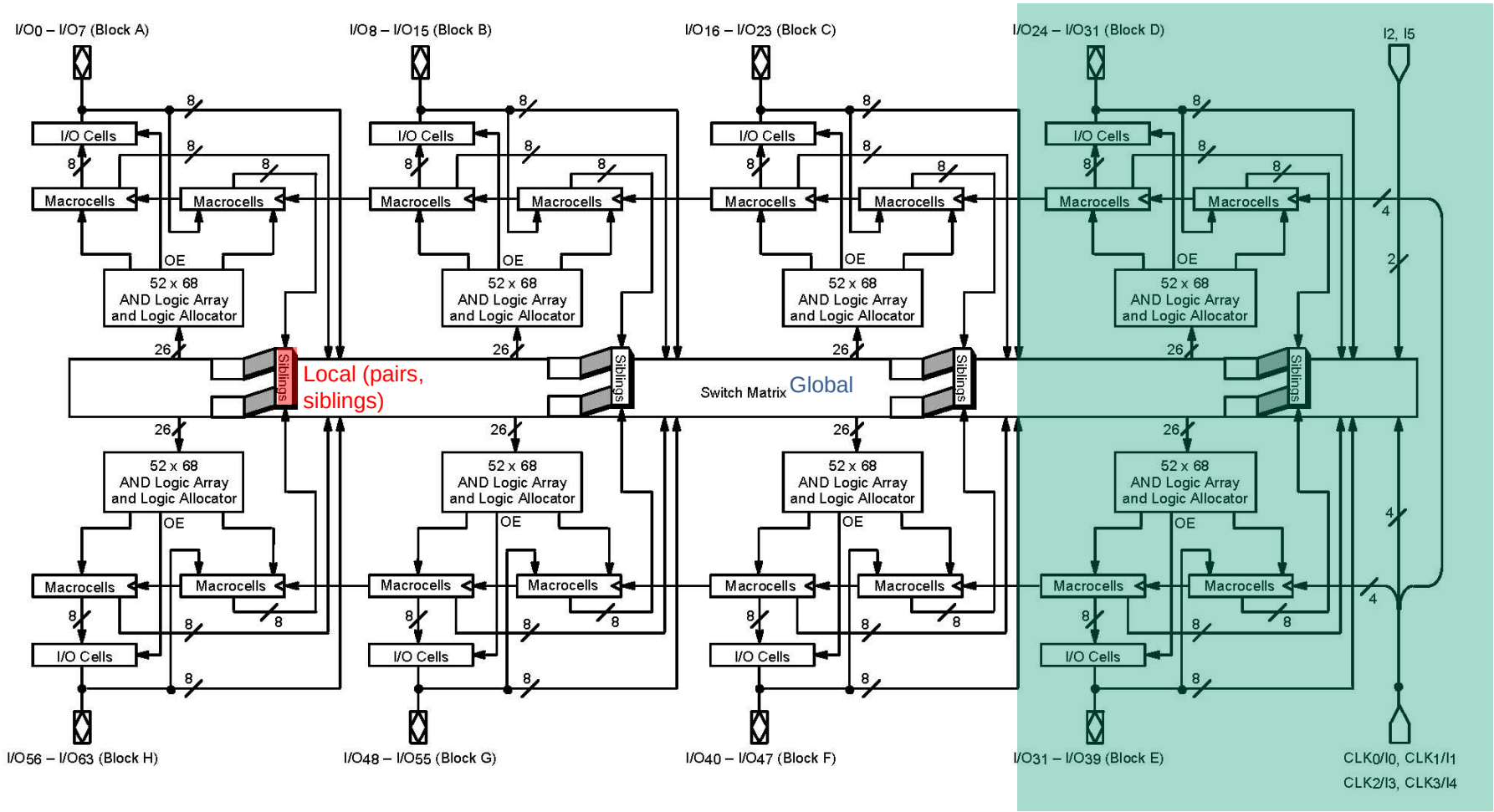
Source of Schematic: Lattice Semiconductor MACH 110-Data Sheet

CPLD-Example: Lattice MACH 110



Source of Schematic: Lattice Semiconductor MACH 110-Data Sheet

CPLD-Example: Lattice MACH 230



Source of Schematic: Lattice Semiconductor MACH 230-Data Sheet

See preceeding slide with simple MACH 110 !

Example: XILINX Coolrunner II-Family

1.8V core, 1.5V … 3.3 V IO, Dual edge clocking reduces clock frequency

	XC2C32A	XC2C64A	XC2C128	XC2C256	XC2C384	XC2C512
I/O Standards	LVTTL,LVCMOS 15,18,25,33	LVTTL,LVCMOS 15,18,25,33	LVTTL,LVCMOS 15,18,25,33 SSTL2-1, SSTL3-1 HSTL-1	LVTTL,LVCMOS 15,18,25,33 SSTL2-1, SSTL3-1 HSTL-1	LVTTL,LVCMOS 15,18,25,33 SSTL2-1, SSTL3-1 HSTL-1	LVTTL,LVCMOS 15,18,25,33 SSTL2-1, SSTL3-1 HSTL-1
Max I/O	33	64	100	184	240	270
T_{PD}(ns)	3.8	4.6	5.7	5.7	7.1	7.1
I/O Banks	2	2	2	2	4	4
DualEDGE Registers	Yes	Yes	Yes	Yes	Yes	Yes
Input Hysteresis	Yes	Yes	Yes	Yes	Yes	Yes
DataGATE & Clock divide	—	—	Yes	Yes	Yes	Yes
Packages	QFG32 VQ44 PC44 CP56	VQ44 PC44 QFG48 CP56 VQ100	VQ100 CP132 TQ144	VQ100 CP132 TQ144 PQ208 FT256	TQ144 PQ208 FT256 FG324	PQ208 FT256 FG324

Source: XILINX Coolrunner II, Product Brief

Coolrunner II Architecture

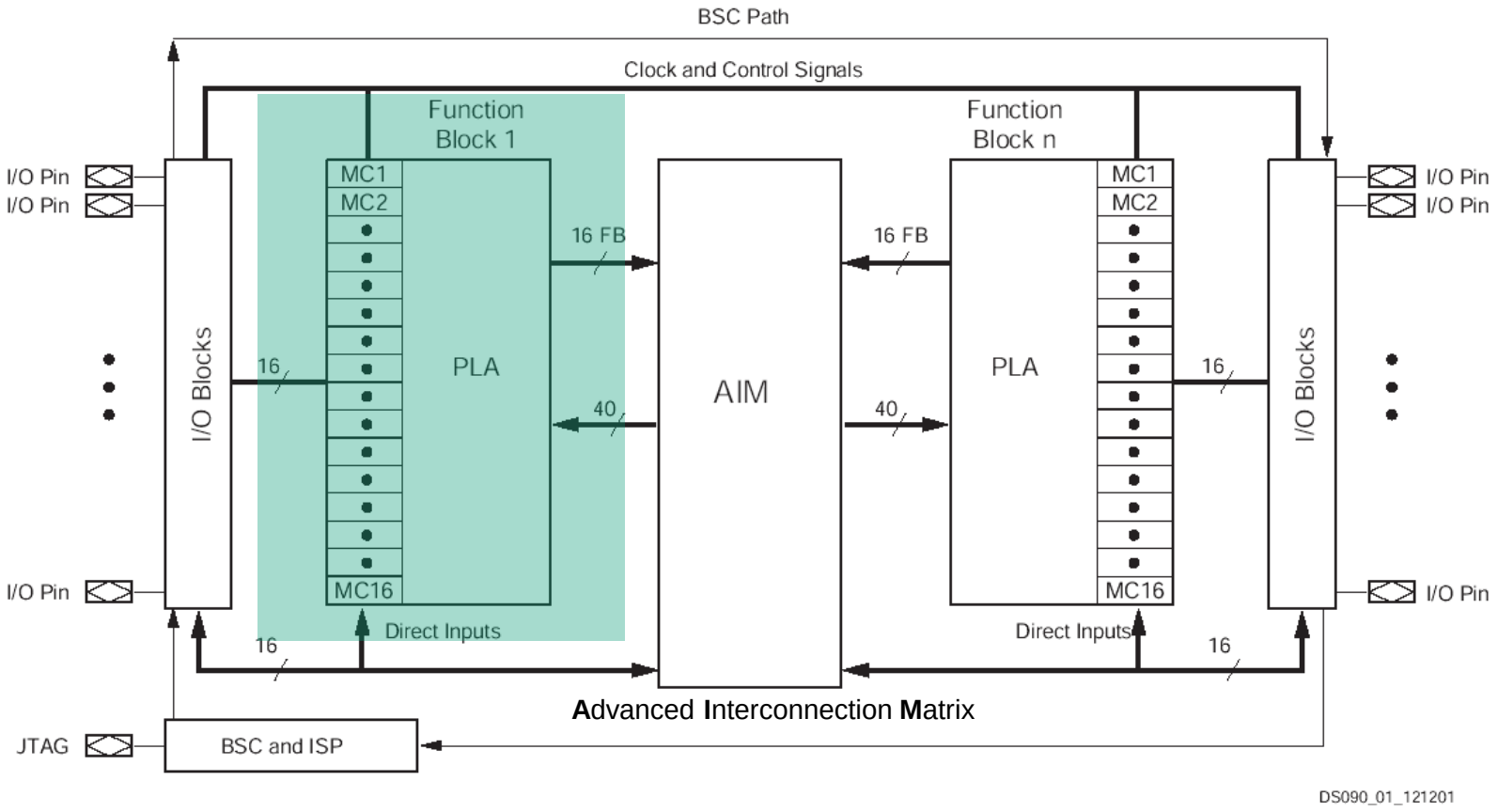
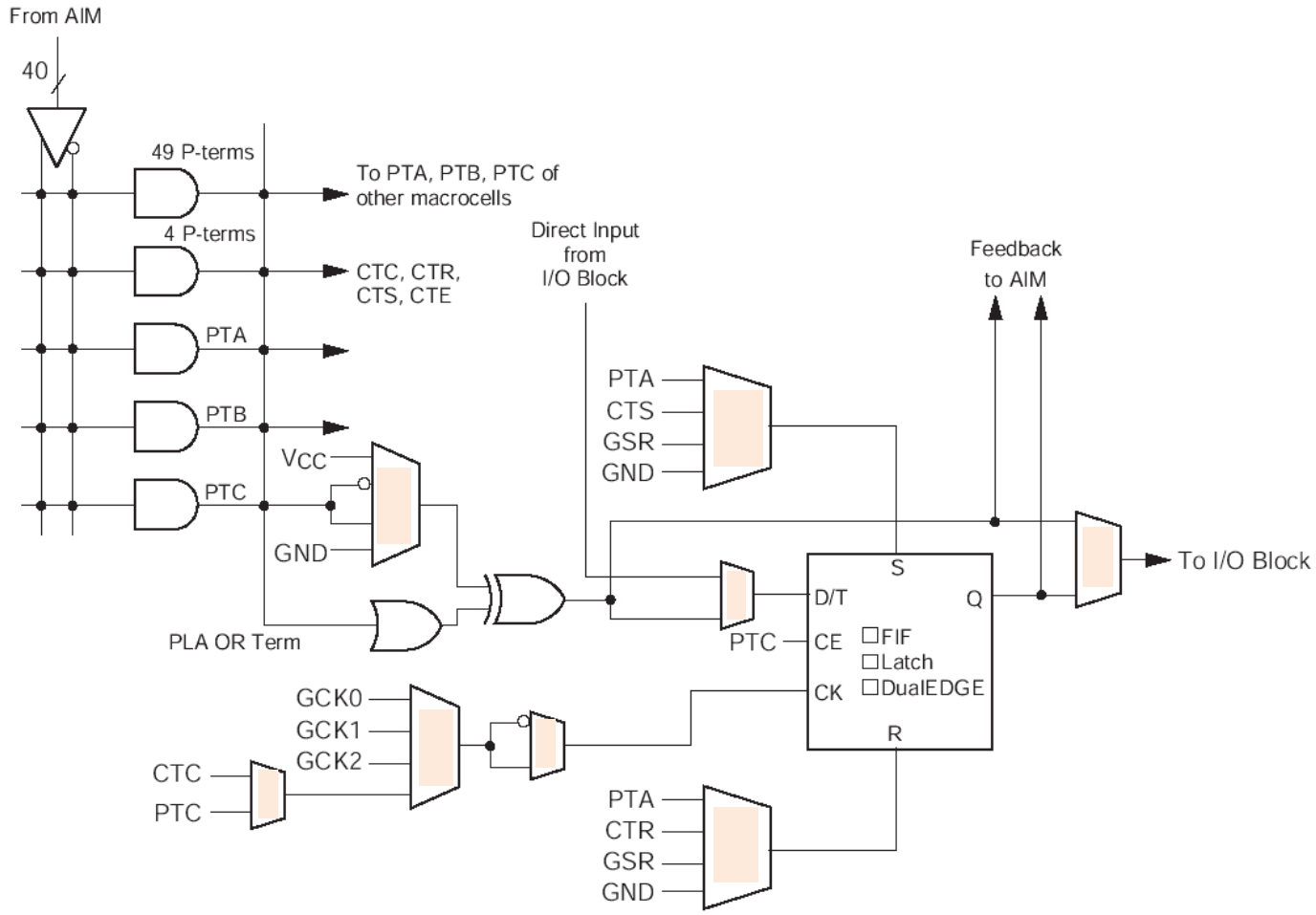


Figure 1: **CoolRunner-II CPLD Architecture**

Source: XILINX, Coolrunner-II CPLD Family, Data Sheet, Sept 2008

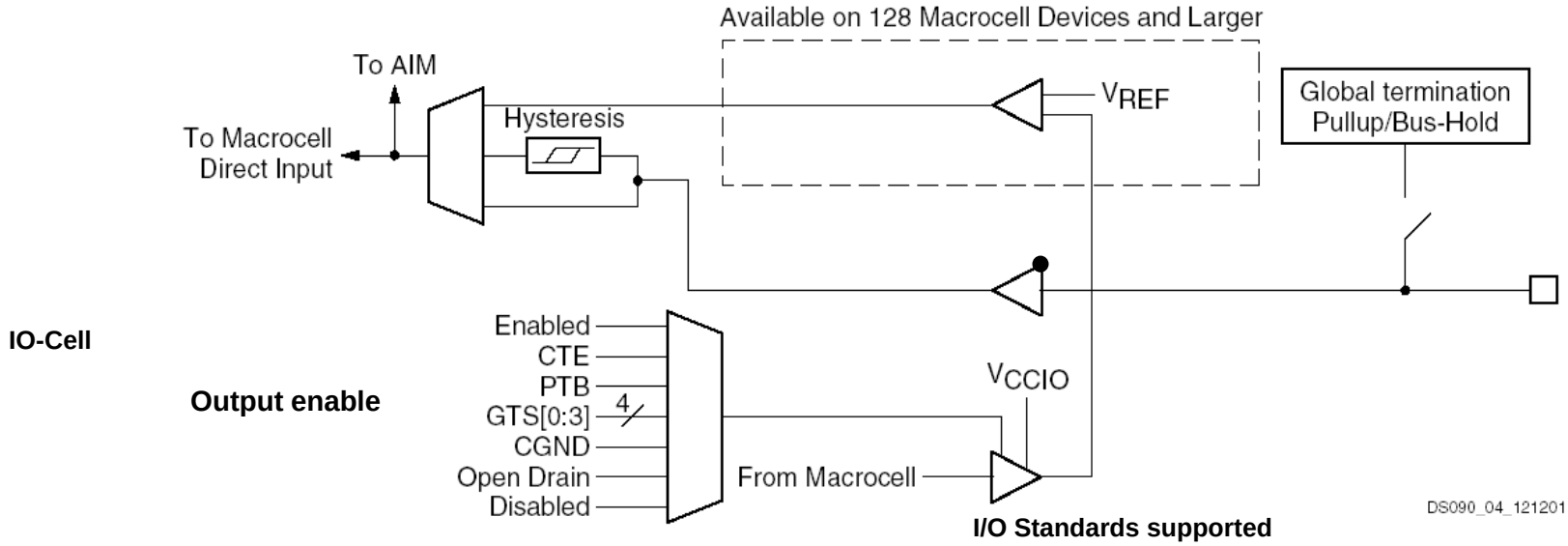
Coolrunner II Logic



CoolRunner-II CPLD Macrocell

Source: XILINX, Coolrunner-II CPLD Family, Data Sheet, Sept 2008

Coolrunner II Input/Output



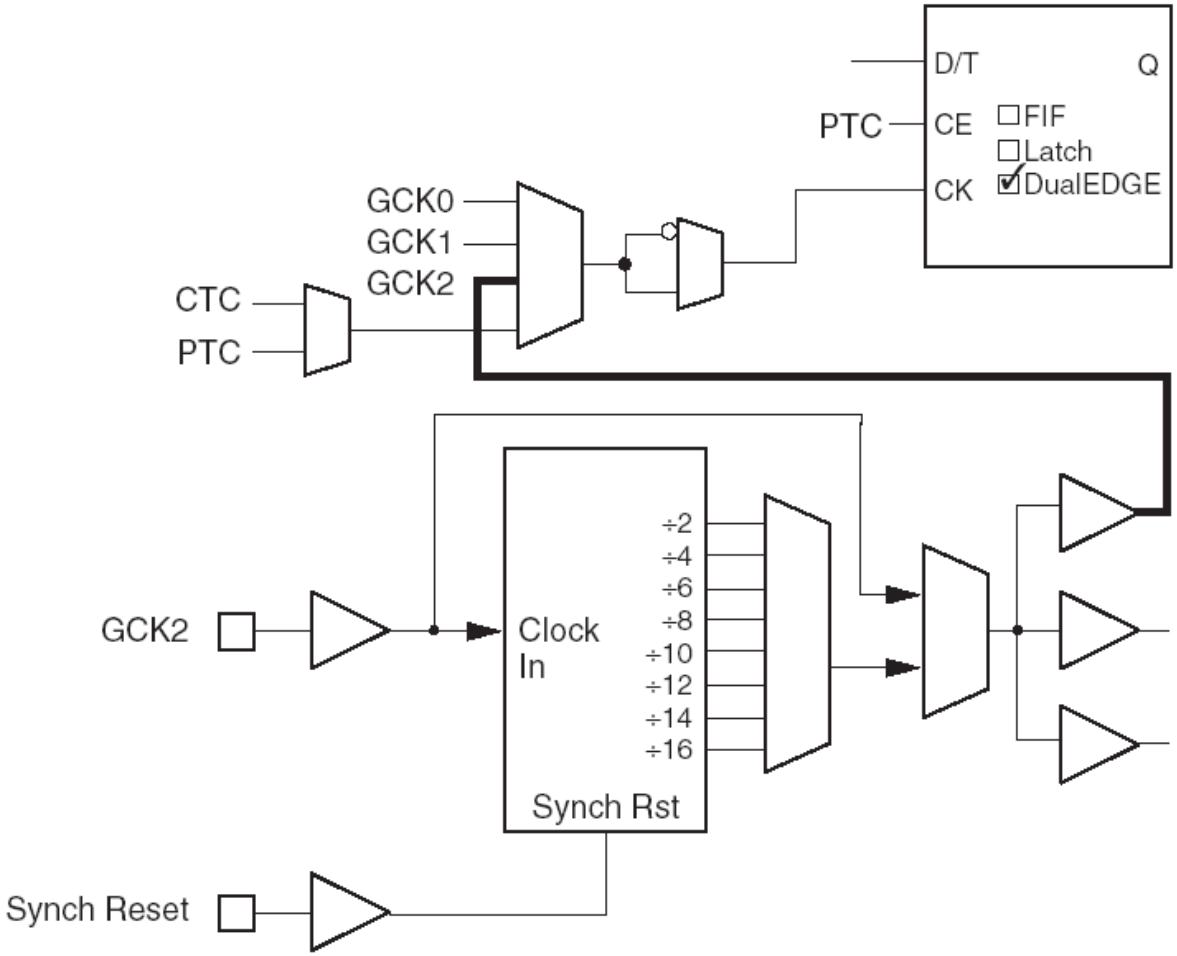
I/O Standards supported

IOSTANDARD Attribute	V_{CCIO}	Input V_{REF}	Board Termination Voltage (V_{TT})	Schmitt-trigger Support
LVTTL	3.3	N/A	N/A	Optional
LVCMOS33	3.3	N/A	N/A	Optional
LVCMOS25	2.5	N/A	N/A	Optional
LVCMOS18	1.8	N/A	N/A	Optional
LVCMOS15	1.5	N/A	N/A	Not optional
HSTL_1	1.5	0.75	0.75	Not optional
SSTL2_1	2.5	1.25	1.25	Not optional
SSTL3_1	3.3	1.5	1.5	Not optional

Low voltage TTL / CMOS
High speed transceiver logic
Stub series terminated logic (DRAM, DDR,PCI Express)

Source: XILINX, Coolrunner-II CPLD Family, Data Sheet, Sept 2008

Coolrunner II Clock Management



Clock Divider and Dual-Edge-FlipFlop

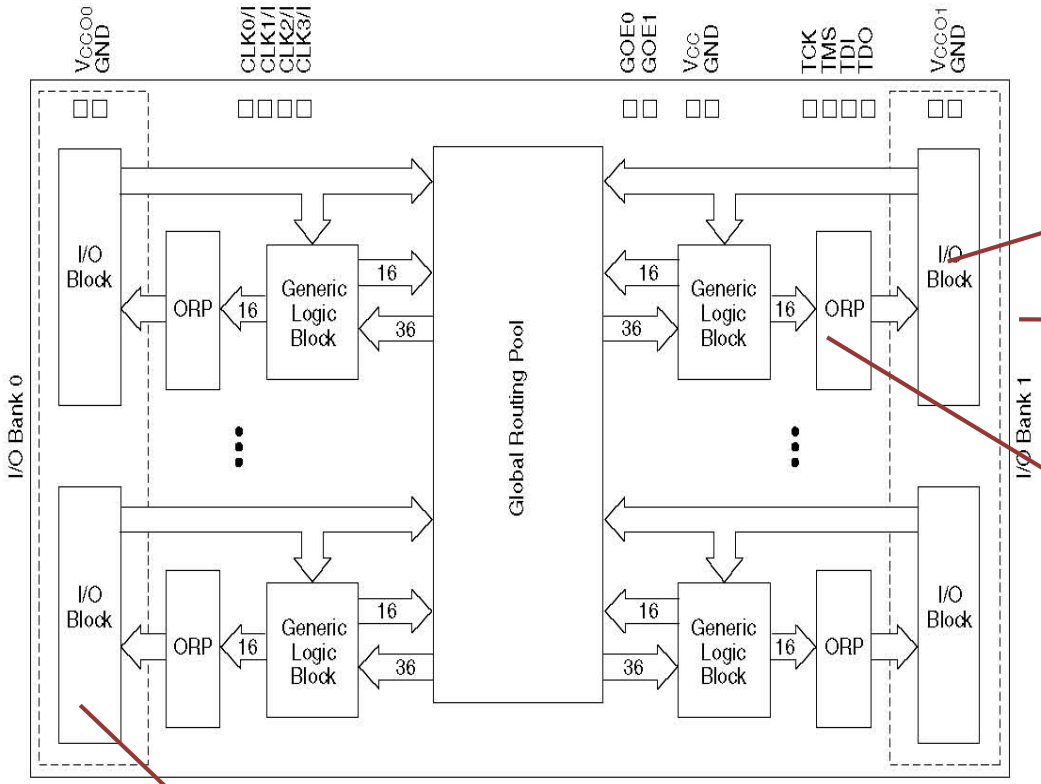
Source: XILINX, Coolrunner-II CPLD Family, Data Sheet, Sept 2008

Device for 2nd lab: Lattice ispMACH 4256ZE

EEPROM-Technologie
Supply Voltage 1,8V
Propagation Delay < 5,8 ns
$F_{max} \approx 200\ MHz$
User I/O: max. 108, depends on package
32 Bit signature
Security bit
Output slew rate control
Boundary scan testable (IEEE 1149.1)
JTAG programming (ISP) (IEEE1532)

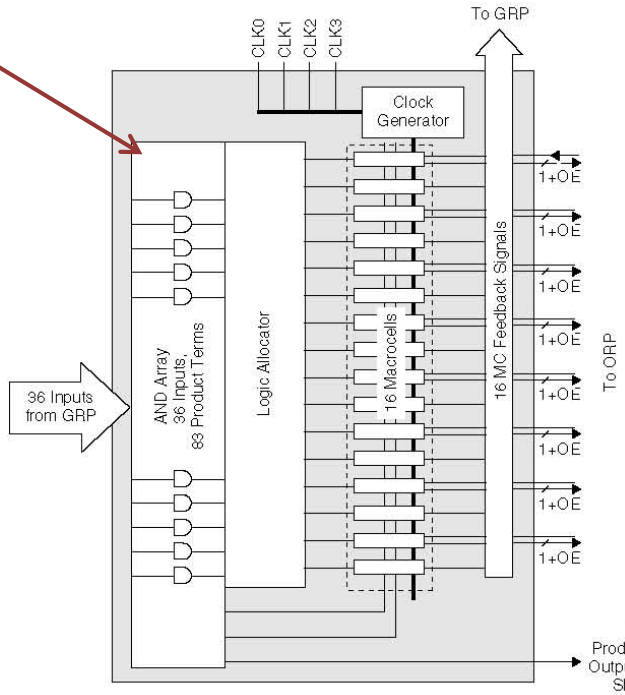
On-chip-oscillator : 5 MHz ($\pm 30\%$) , divider $2^7, 2^{10}$ *or* 2^{20} $(39\ KHz, 5\ KHz, 5\ Hz)$

CPLD-Example: Lattice MACH 4000ZE

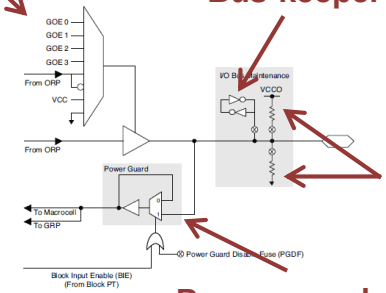


Output Routing Pool
(Outputs, OE signals)

I/O banks may use different I/O standards
LVTTL
LVCMOS 1.5,1.8, 2.5, 3.3
3.3 V PCI
(Inputs 5V-tolerant)



Bus-keeper latch

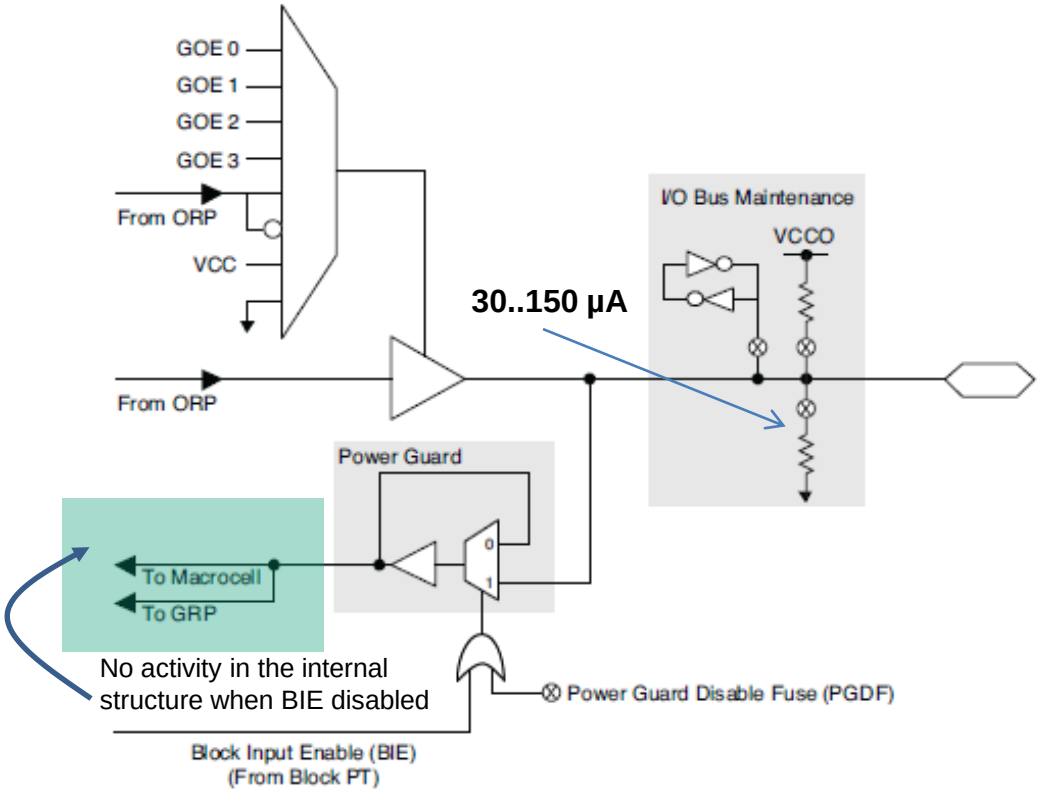


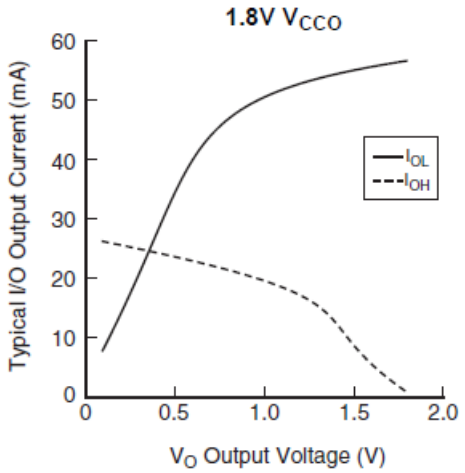
Pull-up and pull-down resistors

Power guard (Input disable)

Source of Schematics: Lattice Semiconductor MACH 4000ZE-Data Sheet

Lattice Mach Z4000ZE Input/Output Cell



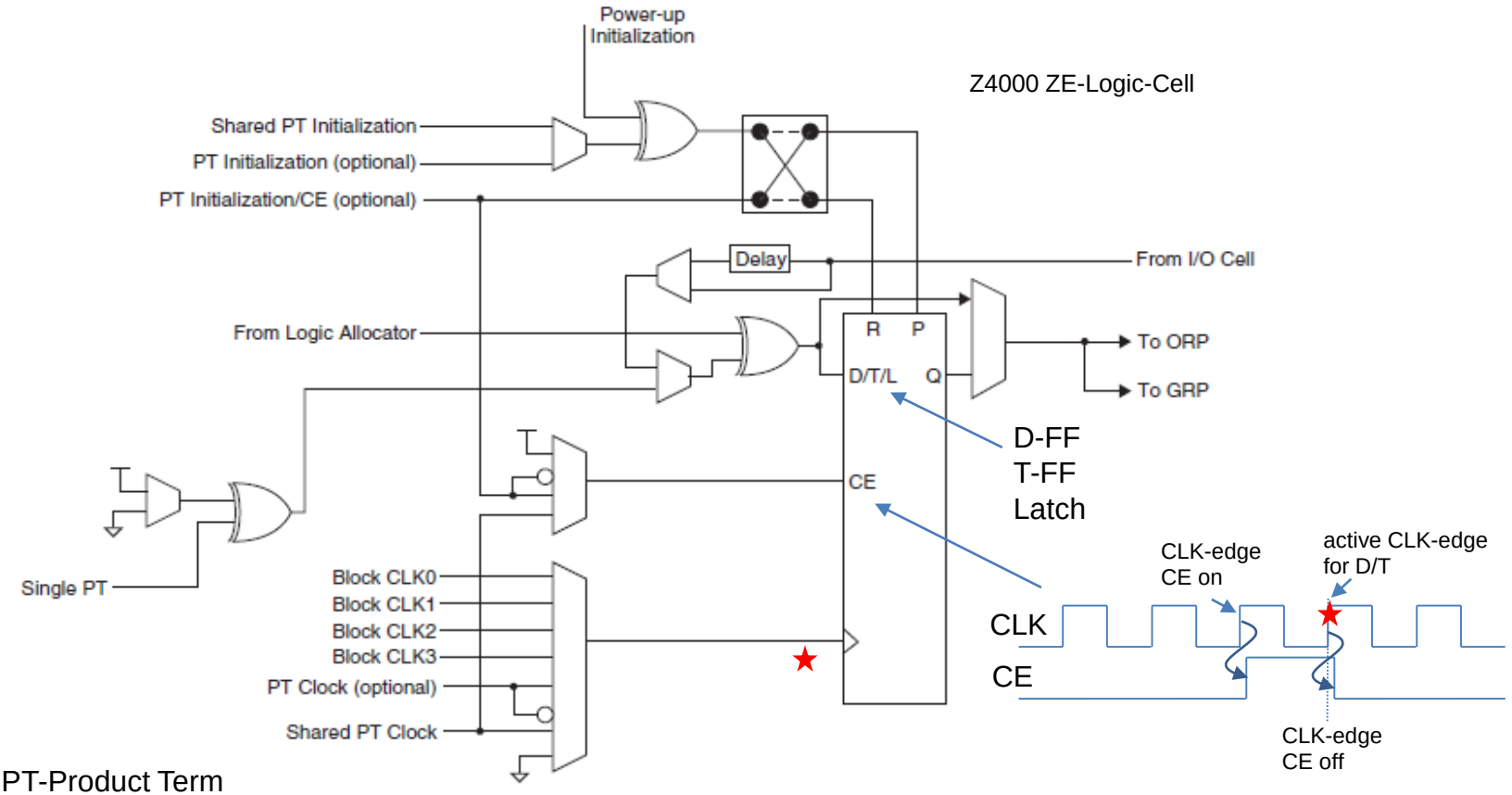


Better:
V(I)-diagram

Power Guard:
Disable change of internal logic levels when feedback not used

Source of Schematics and IO-characteristic: Lattice Semiconductor MACH 4000ZE-Data Sheet

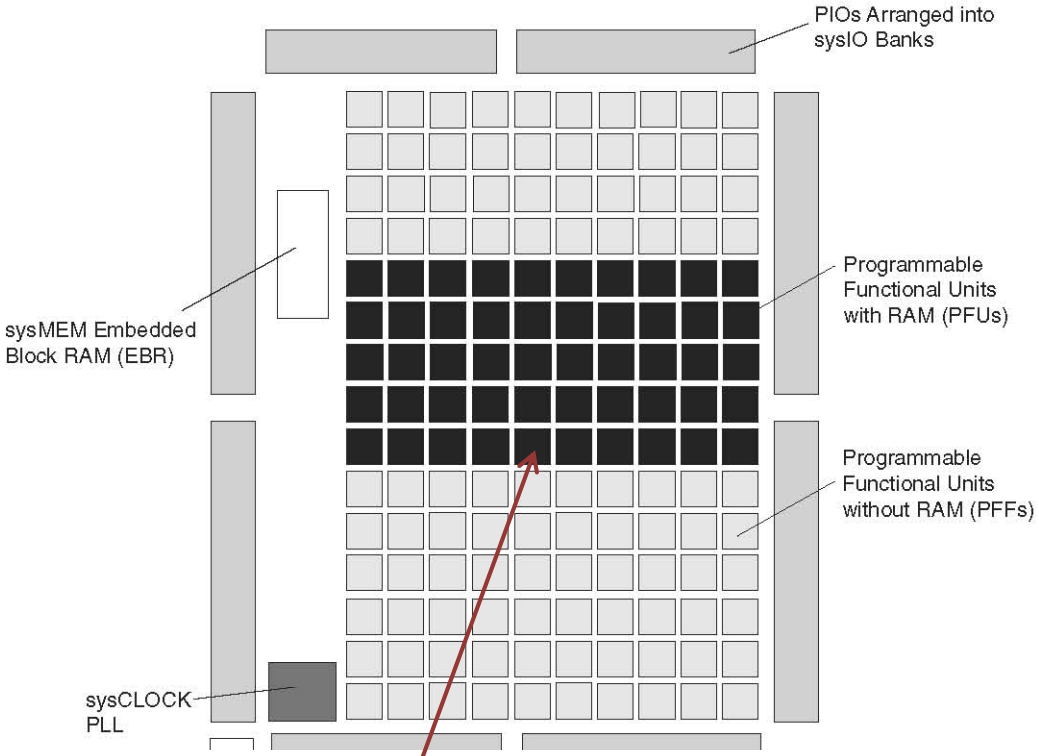
Lattice Mach 4000ZE Logic Cell



PT-Product Term
GRP-Global Routing Pool
ORP-Output Routing Pool

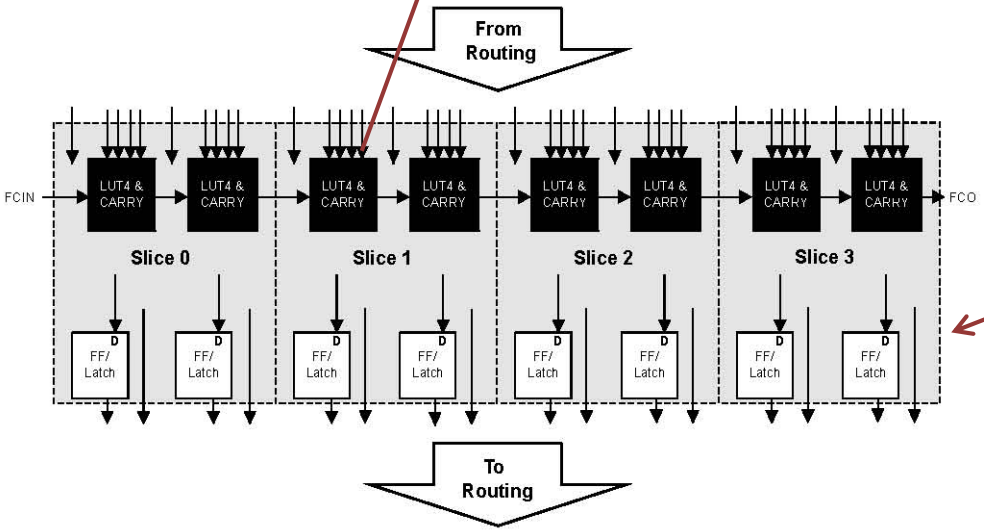
Source of Schematics: Lattice Semiconductor MACH 4000ZE-Data Sheet

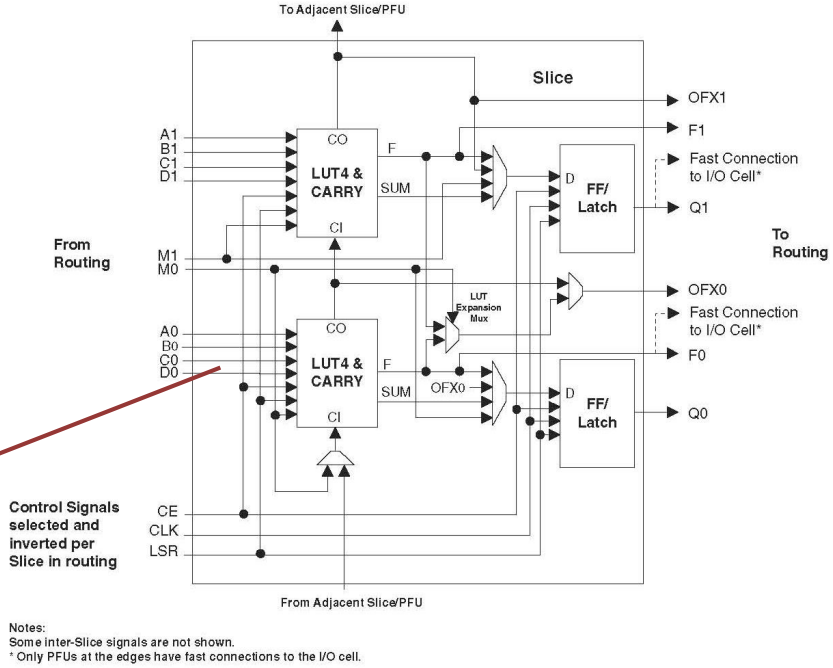
Lattice MachXO



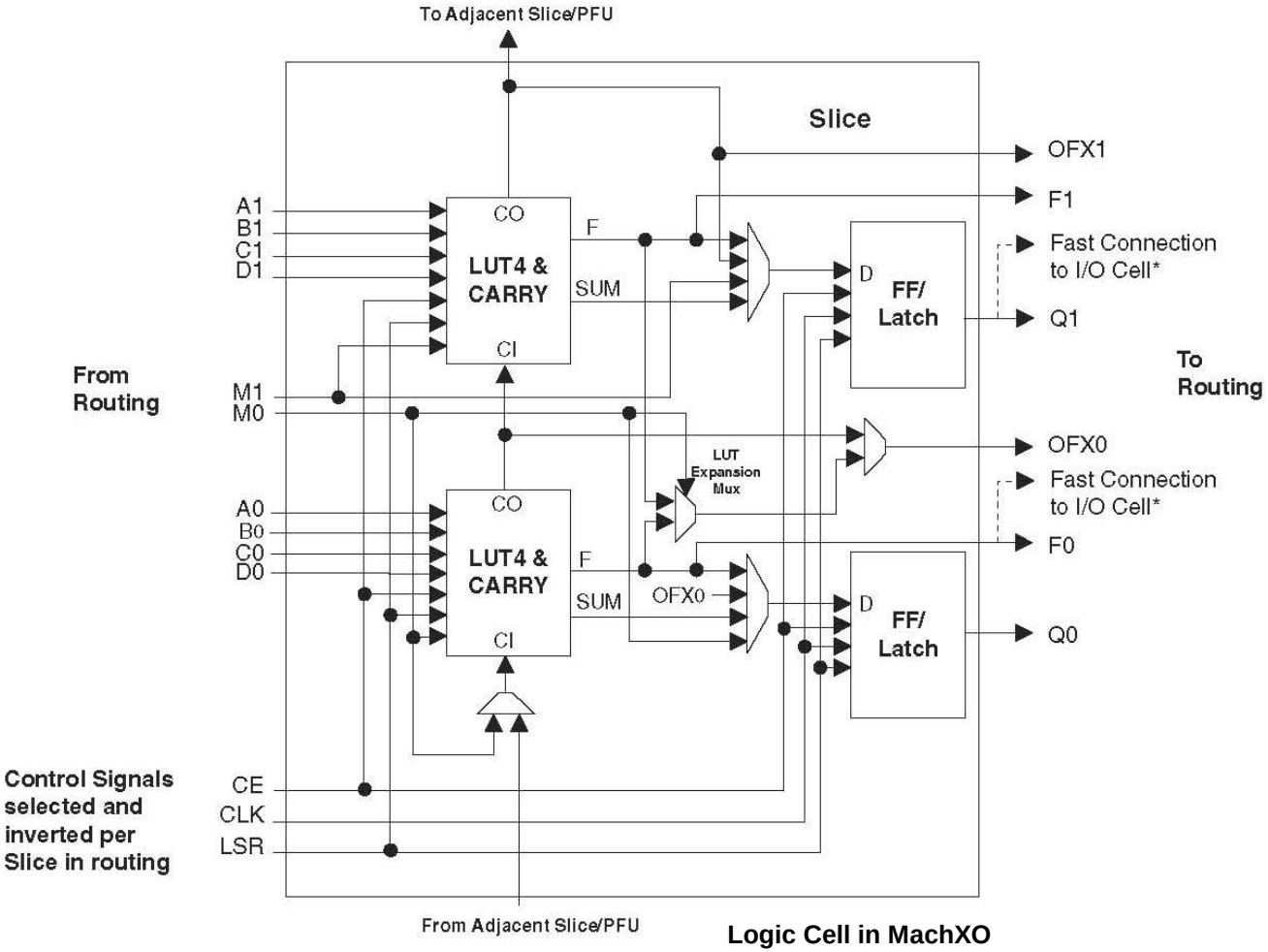
More a FPGA structure than typical CLPD

Configuration held in RAM-cells, additional non-volatile memory for configuation boot on-chip





Lattice Mach XO Logic Cell



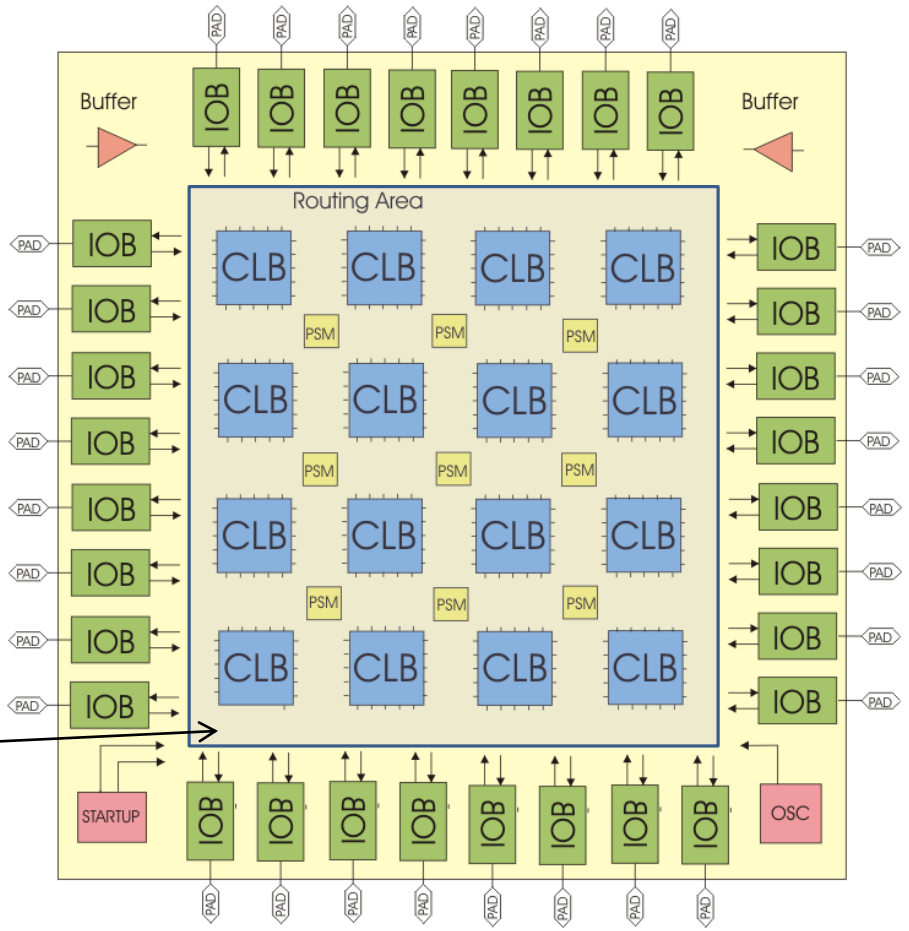
Logic Cell in MachXO

Notes:
Some inter-Slice signals are not shown.
* Only PFUs at the edges have fast connections to the I/O cell.

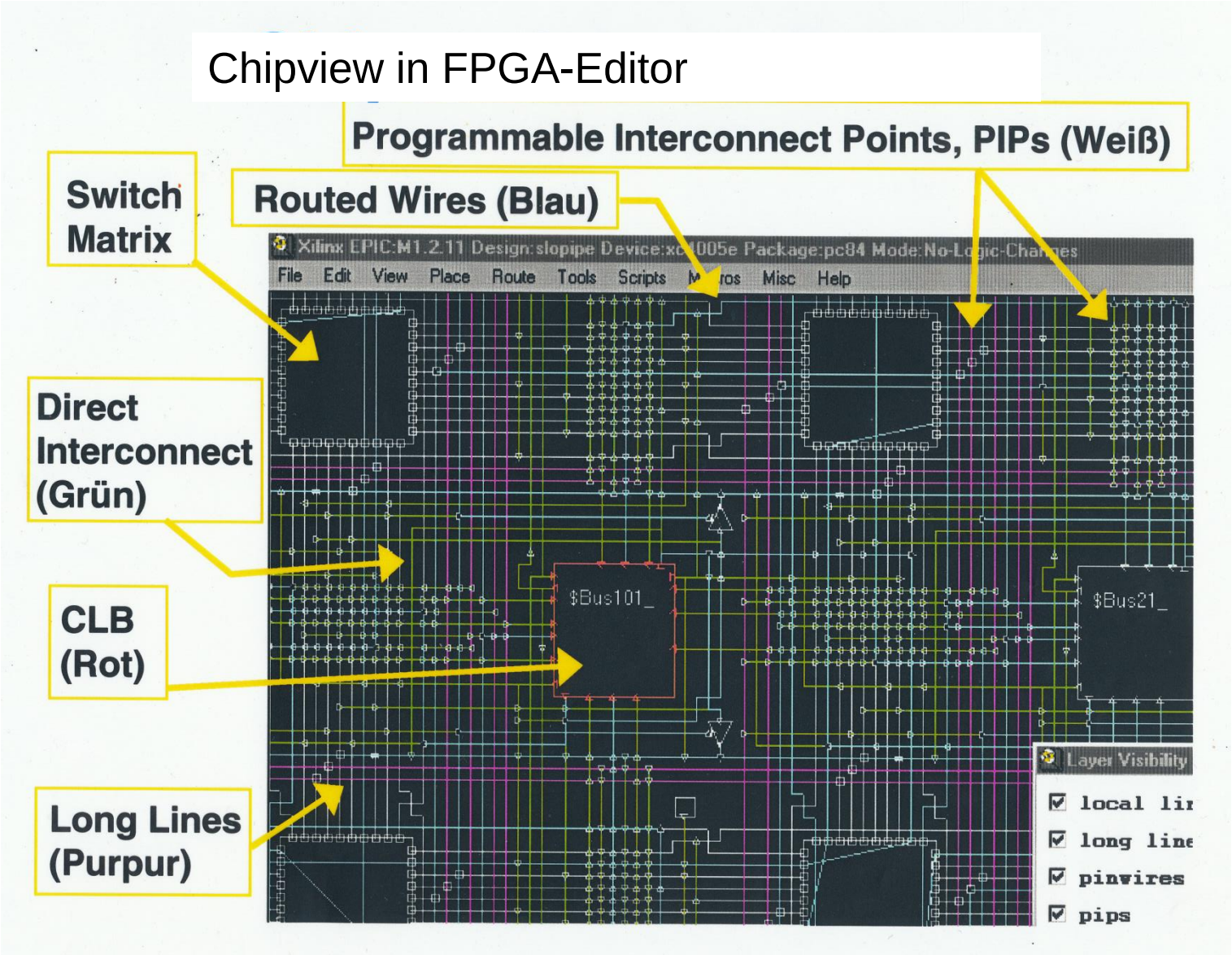
Source of Schematic: Lattice Semiconductor MachXO-Data Sheet

Field Programmable Logic Array - FPGA

CLB – Configurable Logic Block
IOB – Input/Output Block
PSM – Programmable Switch Matrix



Chipview in FPGA-Editor

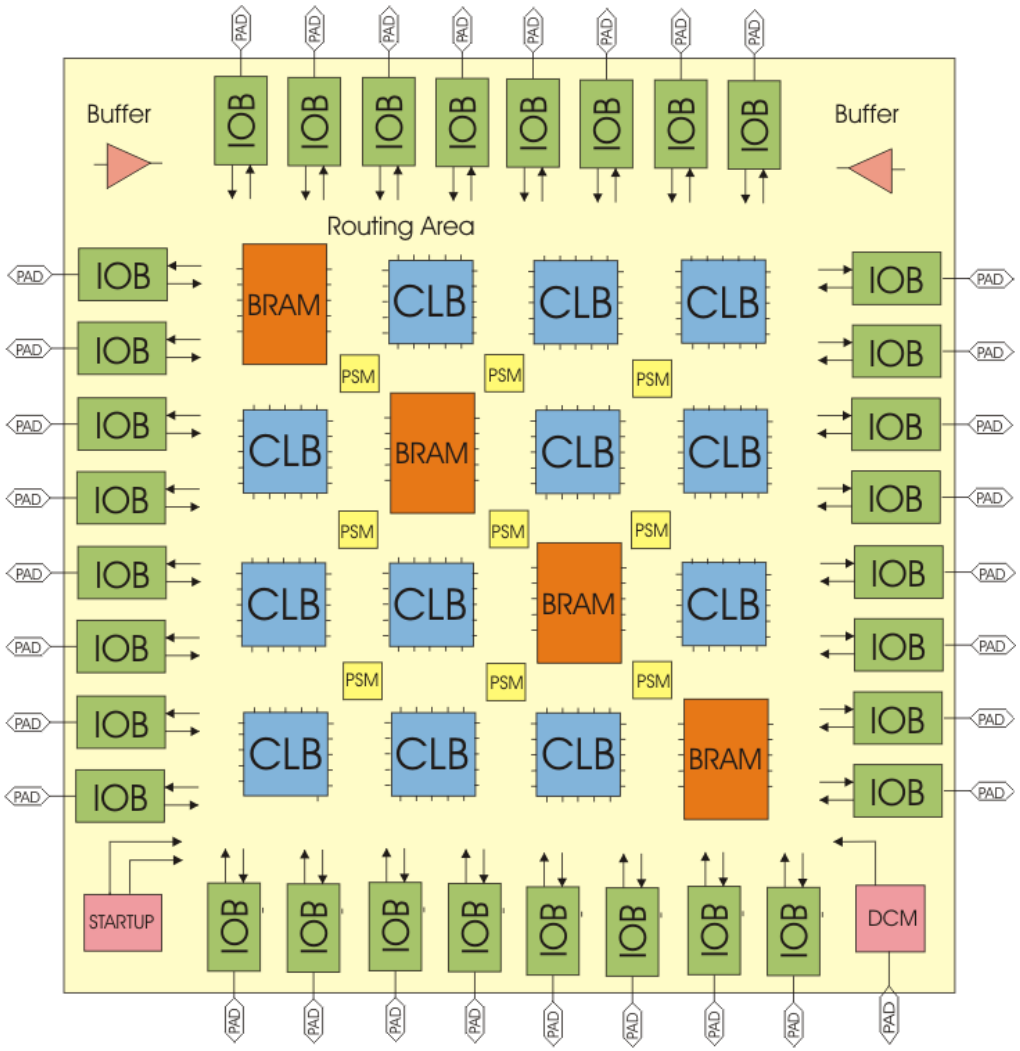


View of (an older) XILINX FPGA in FPGA-Editor, showing the routing area with available interconnections

Field Programmable Logic Array - FPGA

CLB – Configurable Logic Block
IOB – Input/Output Block
PSM – Programmable Switch Matrix

BRAM- Block RAM
DCM – Digital Clock Manager (PLL)



Field Programmable Logic Array - FPGA

CLB - Configurable Logic Block
IOB - Input/Output Block
PSM - Programmable Switch Matrix

BRAM - Block RAM
DCM - Digital Clock Manager (PLL)

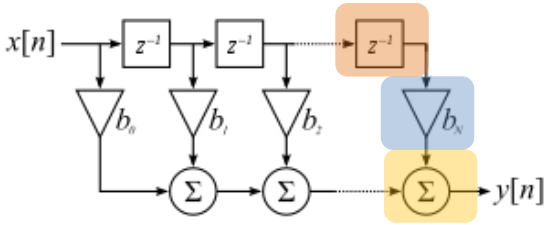
DSP - Digital Signal Processing Blocks

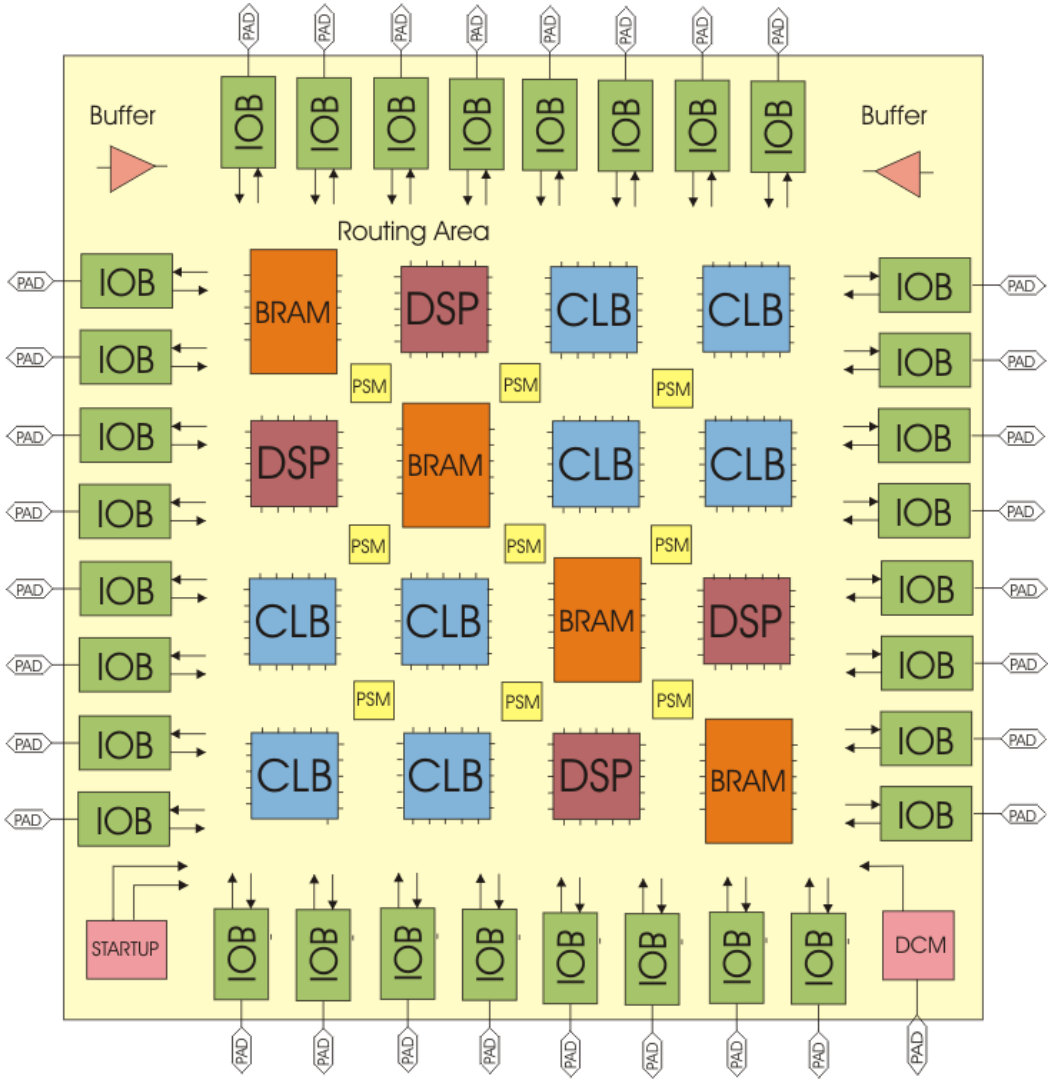
Delay: Memory

Coefficients: Multiplier

Sum: Adder

Example: Finite Impulse Response (FIR) - Filter





Microcontroller using Softcore:
MicroBlaze made from CLBs (700..2000) in fabric

Field Programmable Logic Array - FPGA

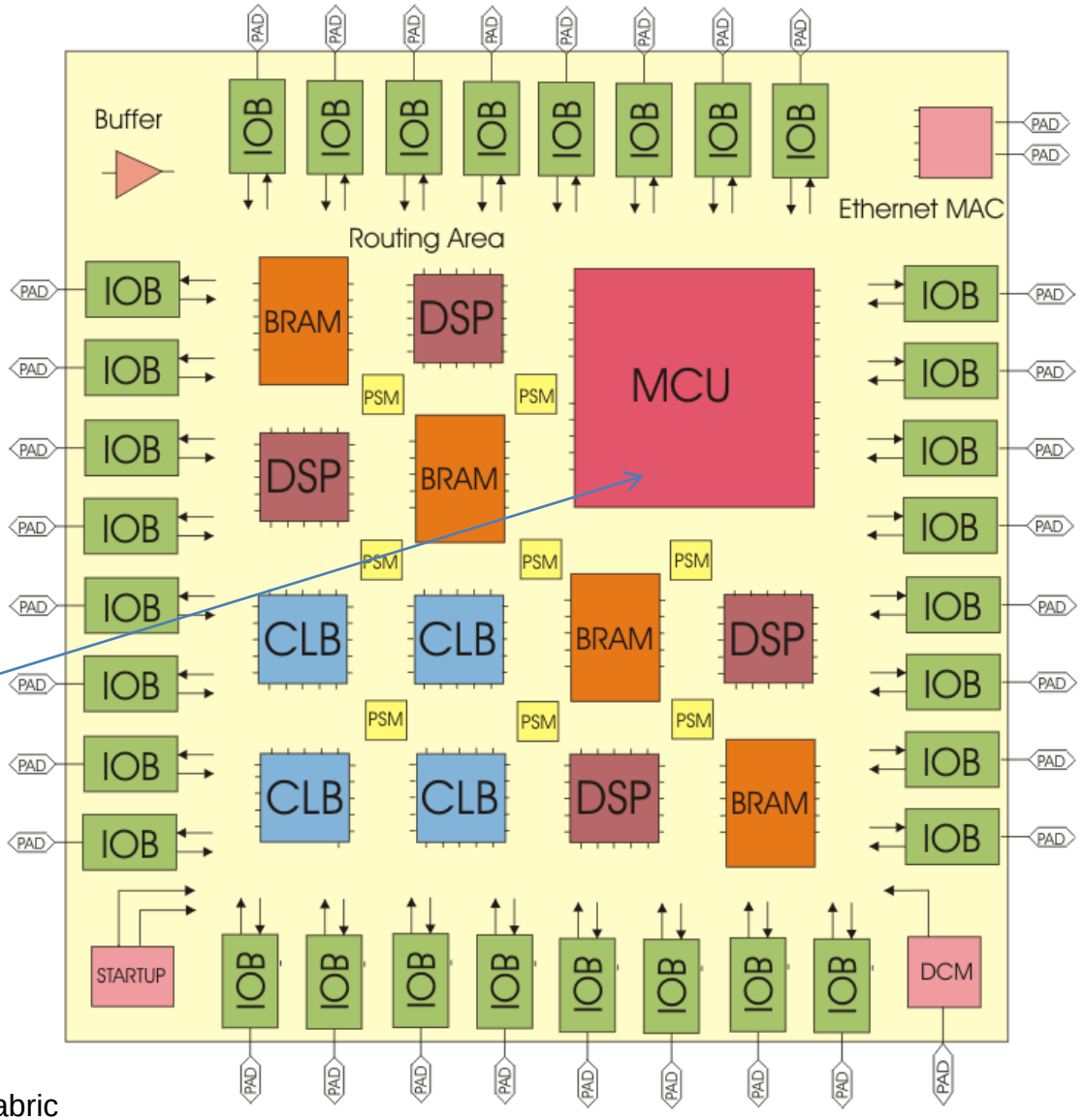
-> SoC System on Chip

CLB – Configurable Logic Block
IOB – Input/Output Block
PSM – Programmable Switch Matrix

BRAM- Block RAM
DCM – Digital Clock Manager (PLL)
DSP-Digital Signal Processing Blocks
MCU-MicroController Unit (or PC-core)

Examples:
XILINX Virtex 5 FXT with 32 Bit
RISC-core (PowerPC 440)

Zynq-7000 with 2 Cortex-A9 Cores
+ NEON SIMD (Fixed and Floating Point)



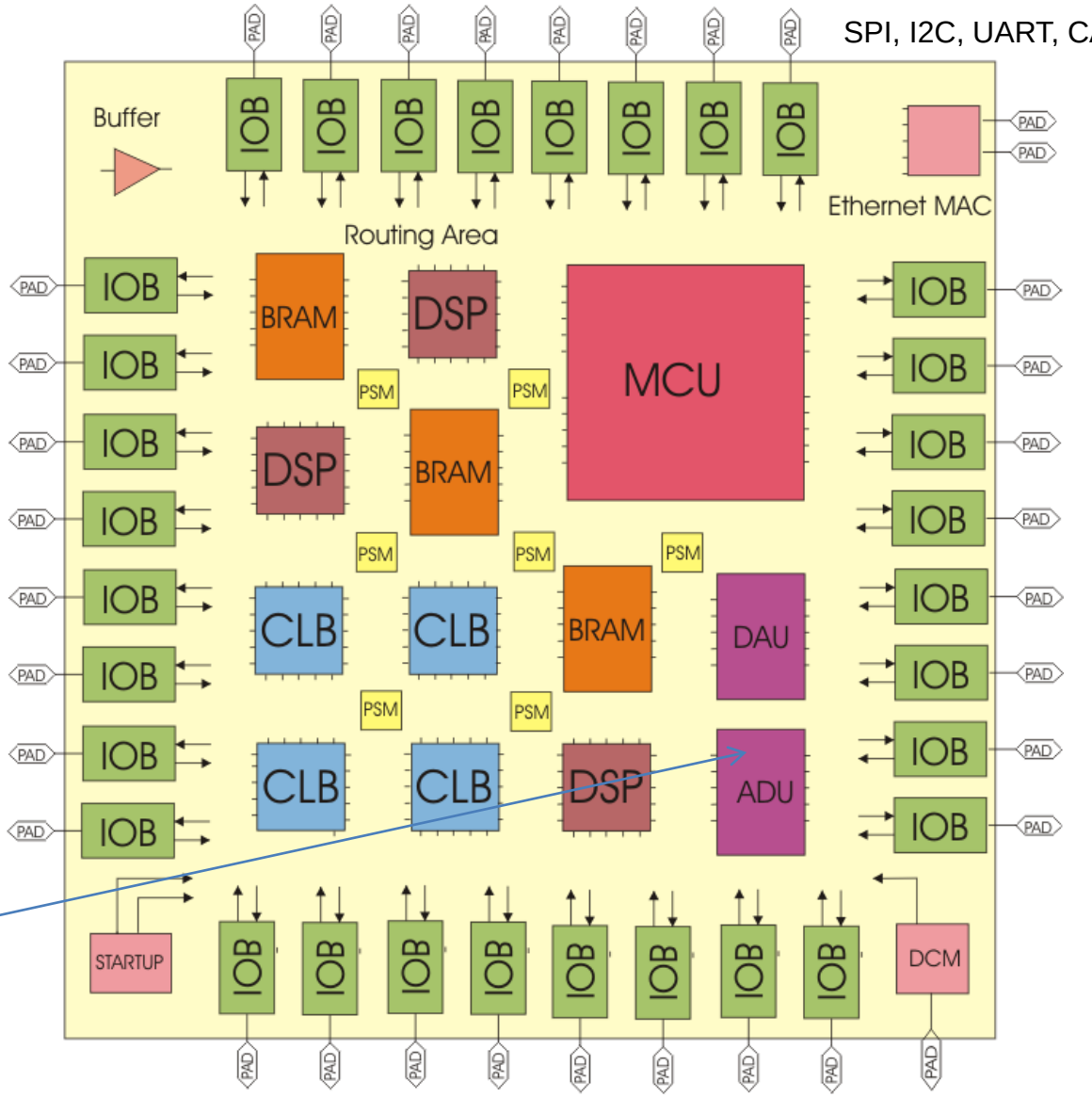
Other solution: **Softcore**
MicroBlaze made from CLBs (700..2000) in fabric

Field Programmable Logic Array - FPGA

CLB – Configurable Logic Block
IOB – Input/Output Block
PSM – Programmable Switch Matrix

BRAM- Block RAM
DCM – Digital Clock Manager (PLL)
DSP-Digital Signal Processing Blocks
MCU-MicroController Unit (or PC-core)

Example:
Atmel SmartFusion
Fabric+ARM+Analog Frontend



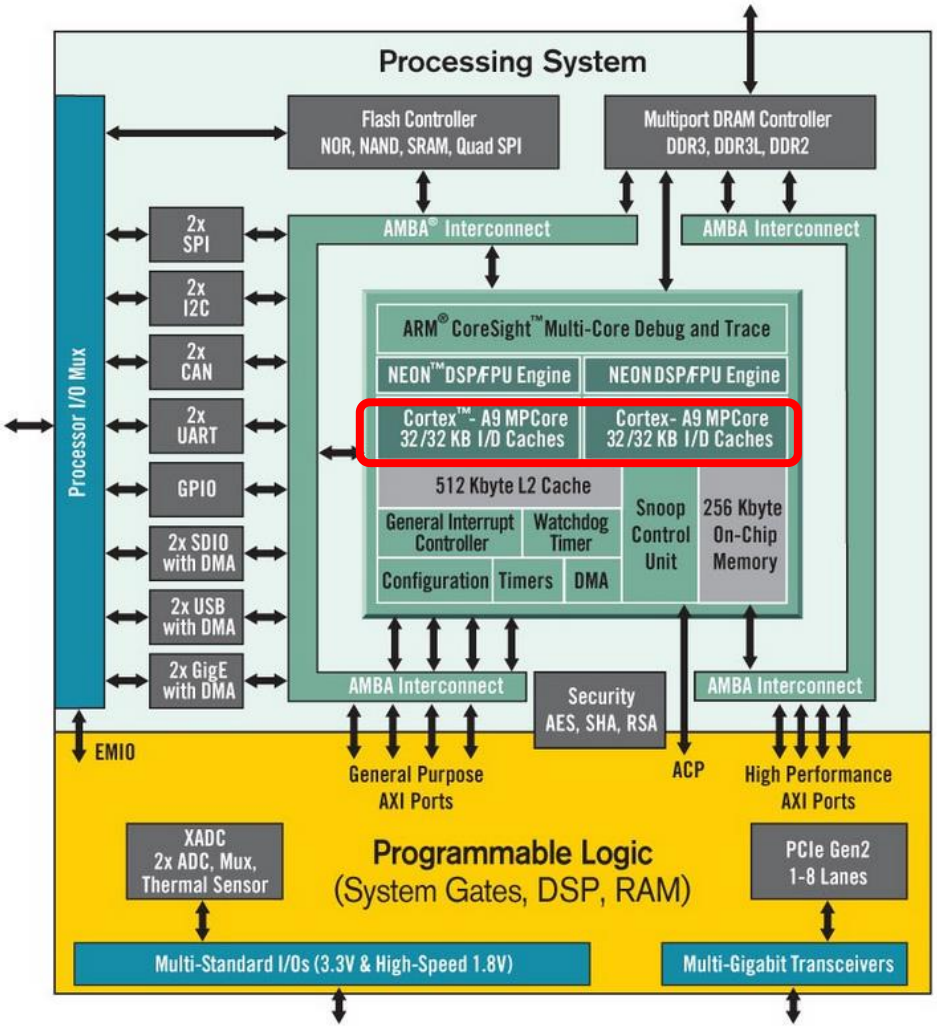
FPGA Examples for Size, Featues …

Device	Logic Cells	Configurable Logic Blocks (CLBs)		DSP48E1 Slices[2]	Block RAM Blocks			MMCMs[4]	Interface Blocks for PCI Express	Ethernet MACs[5]	Maximum Transceivers		Total I/O Banks[6]	Max User I/O[7]
		Slices[1]	Max Distributed RAM (Kb)		18 Kb[3]	36 Kb	Max (Kb)				GTX	GTH		
XC6VLX75T	74,496	11,640	1,045	288	312	156	5,616	6	1	4	12	0	9	360
XC6VLX130T	128,000	20,000	1,740	480	528	264	9,504	10	2	4	20	0	15	600
XC6VLX195T	199,680	31,200	3,040	640	688	344	12,384	10	2	4	20	0	15	600
XC6VLX240T	241,152	37,680	3,650	768	832	416	14,976	12	2	4	24	0	18	720
XC6VLX365T	364,032	56,880	4,130	576	832	416	14,976	12	2	4	24	0	18	720
XC6VLX550T	549,888	85,920	6,200	864	1,264	632	22,752	18	2	4	36	0	30	1200
XC6VLX760	758,784	118,560	8,280	864	1,440	720	25,920	18	0	0	0	0	30	1200
XC6VSX315T	314,880	49,200	5,090	1,344	1,408	704	25,344	12	2	4	24	0	18	720
XC6VSX475T	476,160	74,400	7,640	2,016	2,128	1,064	38,304	18	2	4	36	0	21	840
XC6VHX250T	251,904	39,360	3,040	576	1,008	504	18,144	12	4	4	48	0	8	320
XC6VHX255T	253,440	39,600	3,050	576	1,032	516	18,576	12	2	2	24	24	12	480
XC6VHX380T	382,464	59,760	4,570	864	1,536	768	27,648	18	4	4	48	24	18	720
XC6VHX565T	566,784	88,560	6,370	864	1,824	912	32,832	18	4	4	48	24	18	720

XILINX Virtex 6 features
One slice contains 4 LUTs and 8 FF

Source table: Xilinx Document DS150

System on Chip - Device



Zynq-7000

80 … 450 I/O
Processor I/O + Logic I/O

28k .. 444k Logic Cells

17600 LUT
35200 FF

277400 LUT
554800 FF

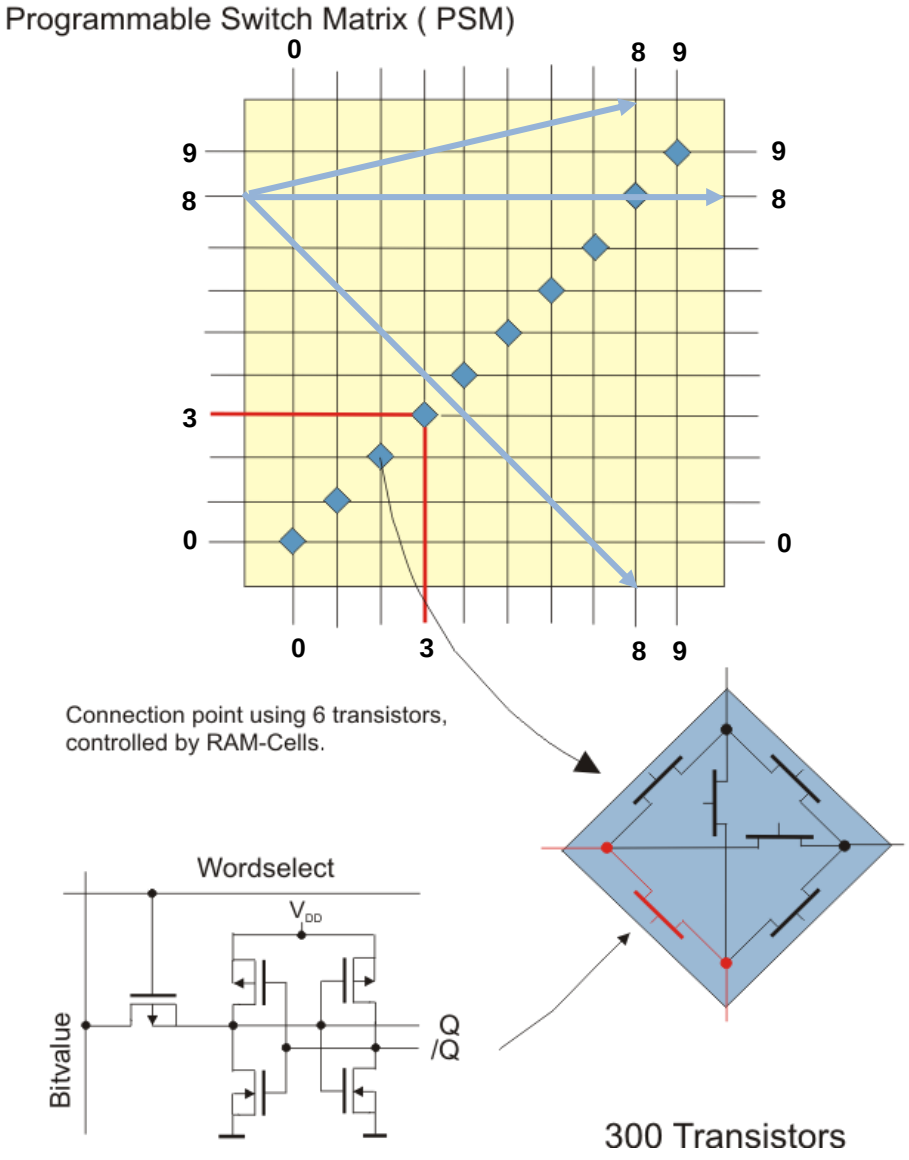
240kB .. 3 MB Block RAM

80 .. 2020 DSP Blocks

12 Bit – ADC with Multiplexers

Source: http://www.xilinx.com/products/silicon-devices/soc/zynq-7000/index.htm

Crossconnection Matrix with 10 Connection Points

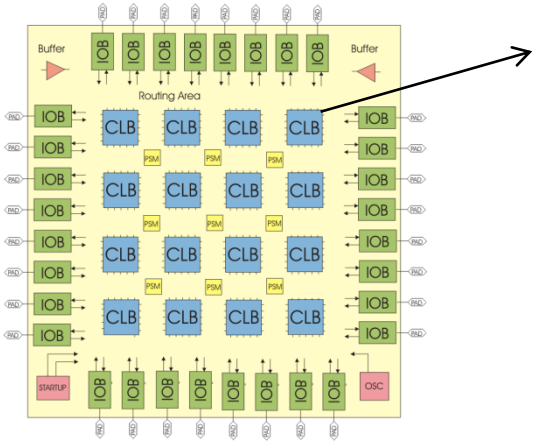


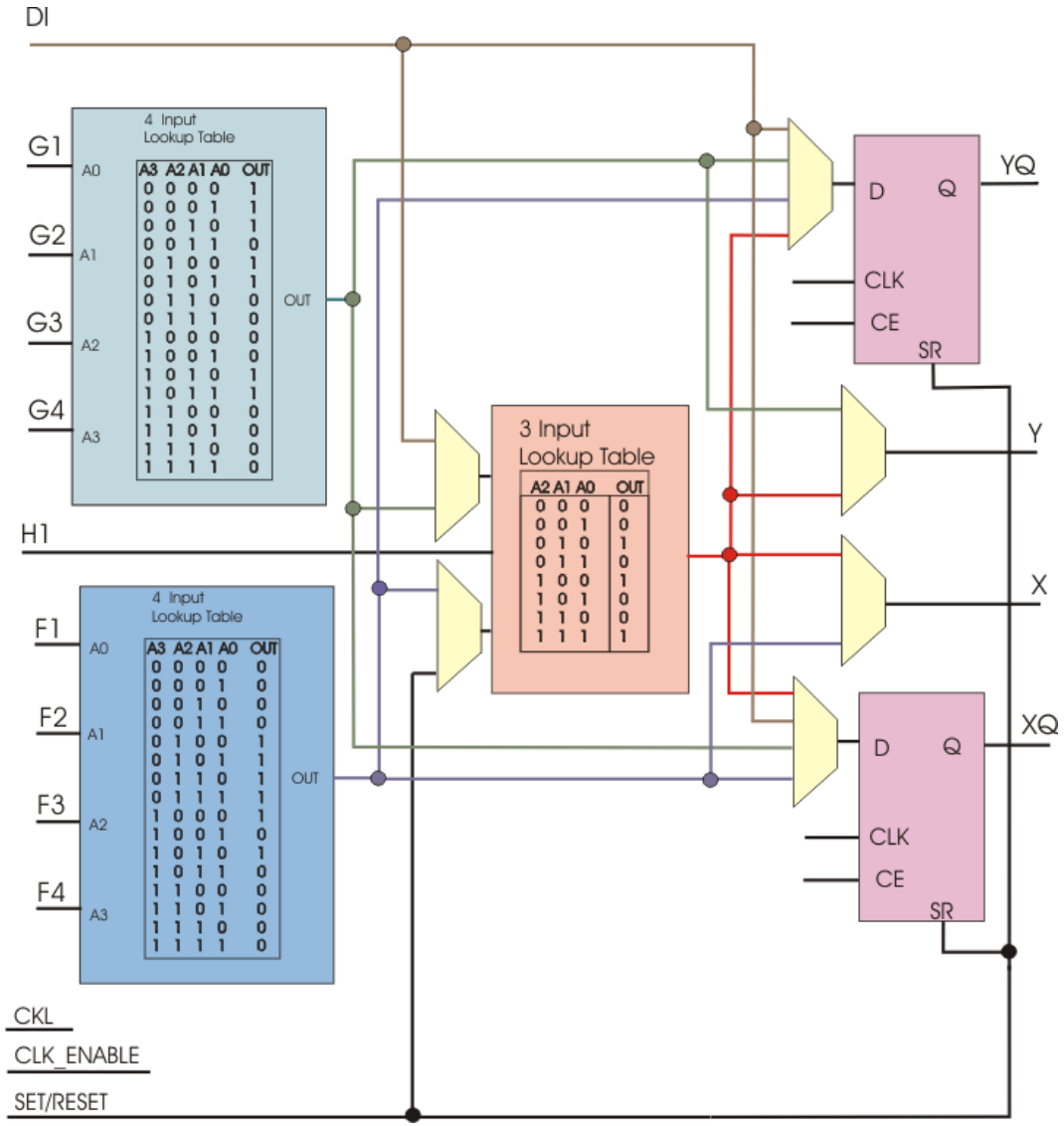
Configurable Logic Block

Lookup Table:
Programming: writable RAM to store content

Logic mode: ROM address is input vector
data is output

RAM mode: Used as normal RAM
"distributed" (in the CLBs) RAM)

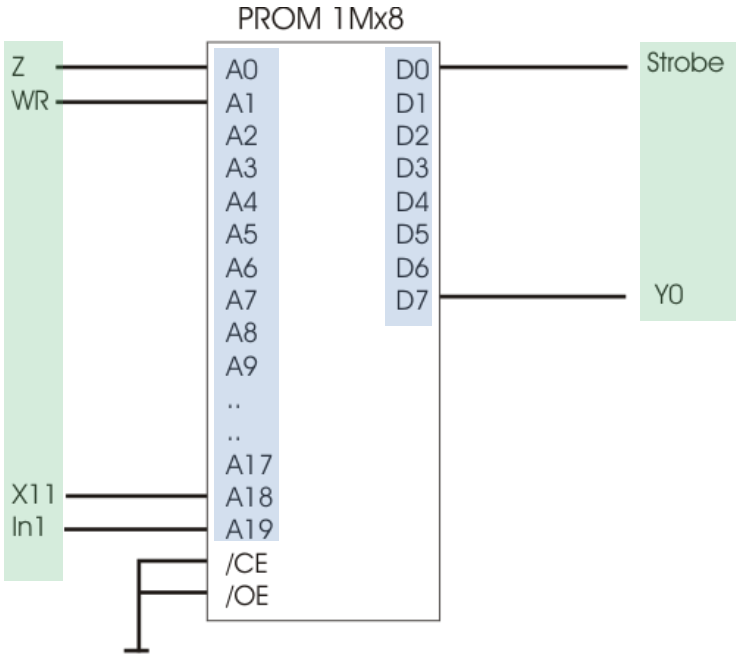




LOOKUP TABLE

Implementation of Logic as Truth Table in Memory (ROM)

	Address Inputs						Data Outputs		
	A19	A18	.....	A2	A1	A0	D7	.....	D0
	In1	X11	.....	En	WR	Z	y0	.....	Strobe
0	0	0	.....	0	0	0	0	.....	1
1	0	0		0	0	1	0	.....	1
2	0	0	.....	0	1	0	0	.....	1
3	0	0	.....	0	1	1	1	.....	1
.....									
.....									
1048575	1	1	.....	1	1	0	0	.....	0
1048576	1	1	.....	1	1	1	1	.....	1



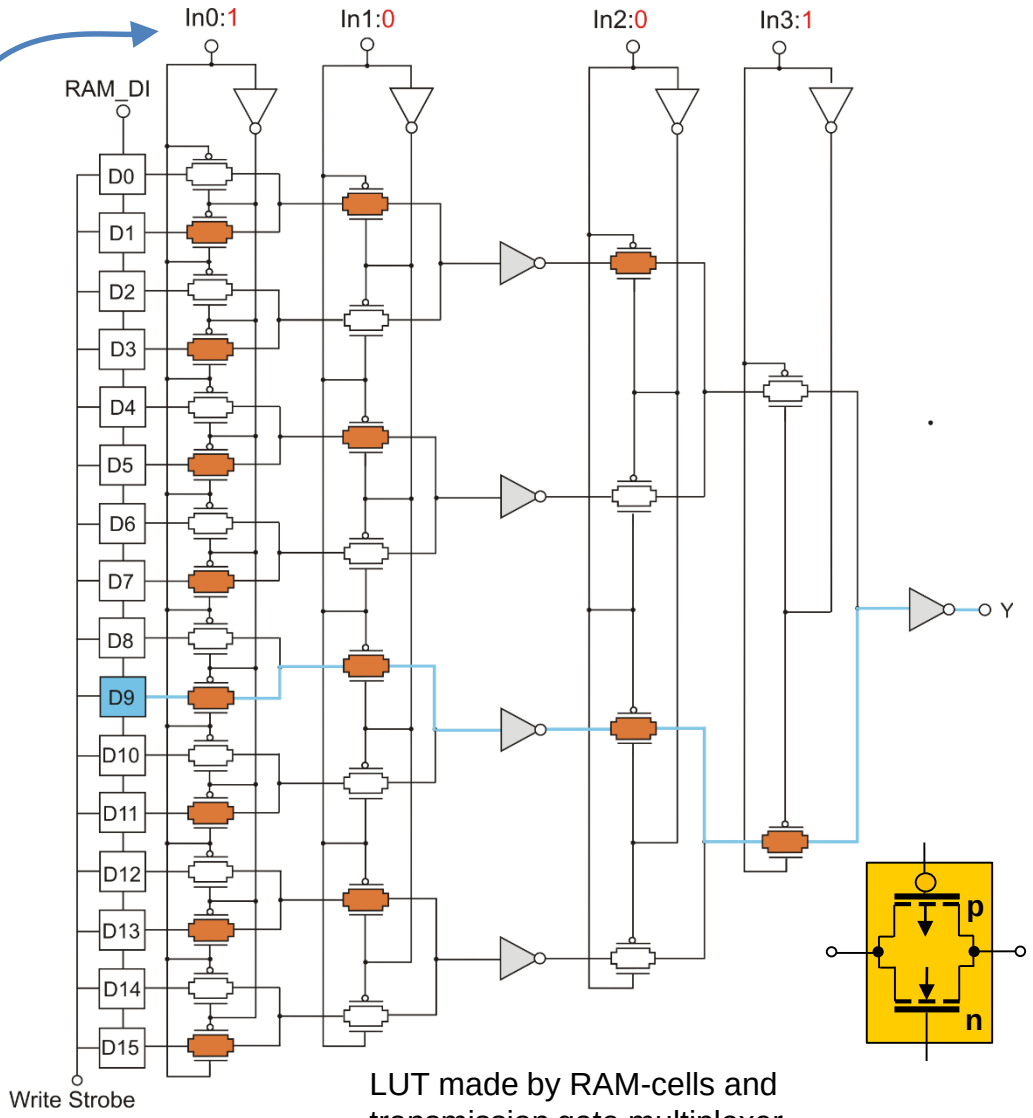
PROM for implementation of boolean function as lookup table

-- For greater number of inputs large PROMs are necessary. No benefit by minimization.
-- Bipolar PROMS consume much power (fast)
-- EPROMS/EEPROMS slower than "glue logic"
-- In FPGA: Static RAM (written during configuration only -> used as PROM)

LOOKUP-Table with Transmission Gates

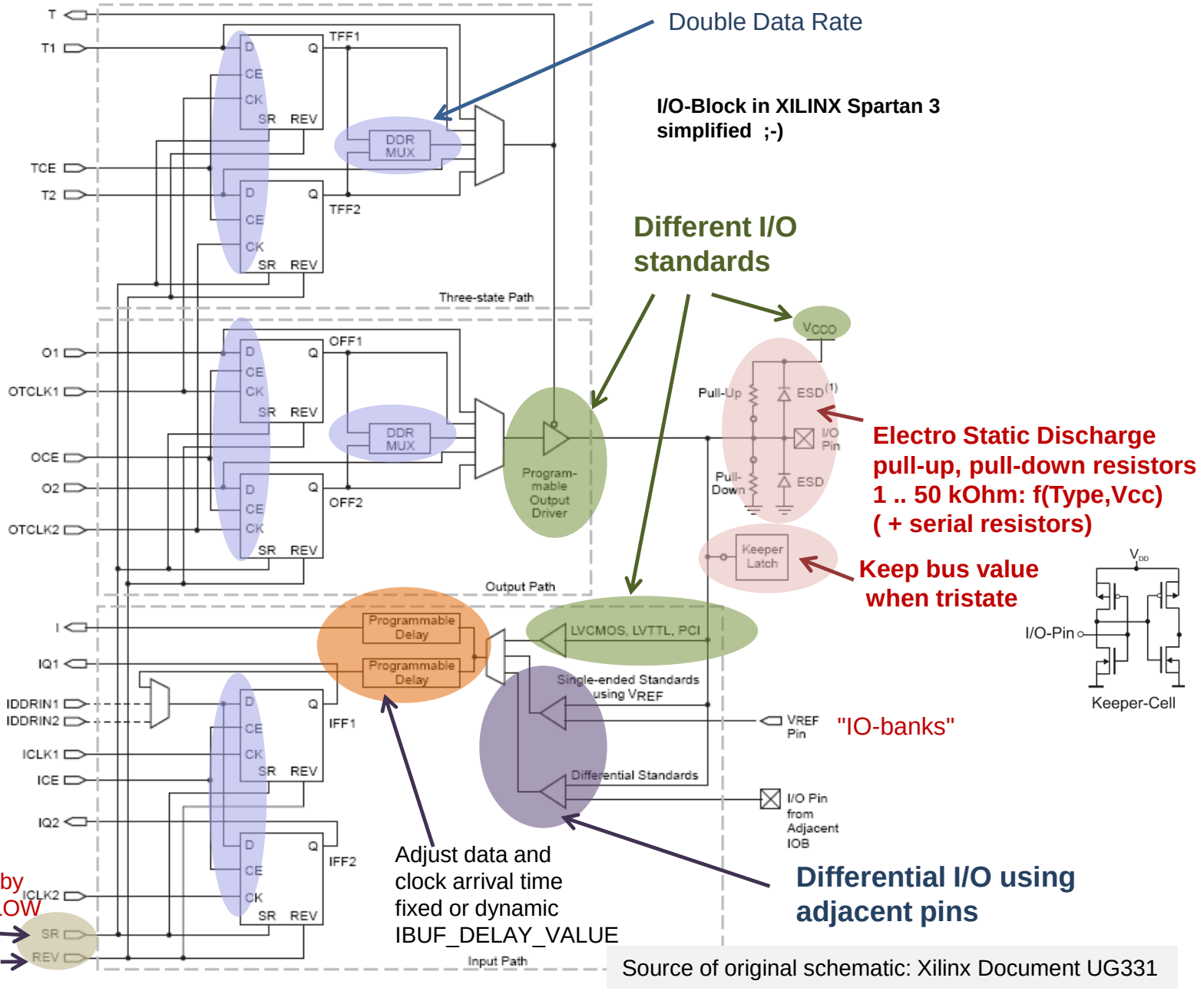
truth table

In3	In2	In1	In0	Y
0	0	0	0	D0
0	0	0	1	D1
0	0	1	0	D2
0	0	1	1	D3
0	1	0	0	D4
0	1	0	1	D5
0	1	1	0	D6
0	1	1	1	D7
1	0	0	0	D8
1	0	0	1	**D9**
1	0	1	0	D10
1	0	1	0	D11
1	1	0	0	D12
1	1	0	1	D13
1	1	1	0	D14
1	1	1	1	D15



LUT made by RAM-cells and transmission gate multiplexer

FPGA I/O-Cell



Source of original schematic: Xilinx Document UG331

Other (simple) Resources in FPGA

Tristate Buffer	Implementation of Busses
Clock Drivers	Implementation of Clocks with minimal Skew
Global Reset Logic	
Boundary Scan Cells and Access Logic	
Oscillators	
Wide Decoders	

Other (complex) Hardware Resources in FPGA

DCM: Clock Synchronization, Clock Divider/Multiplier
Phase Shifter

Block RAM : Configurable as DPM, FIFO

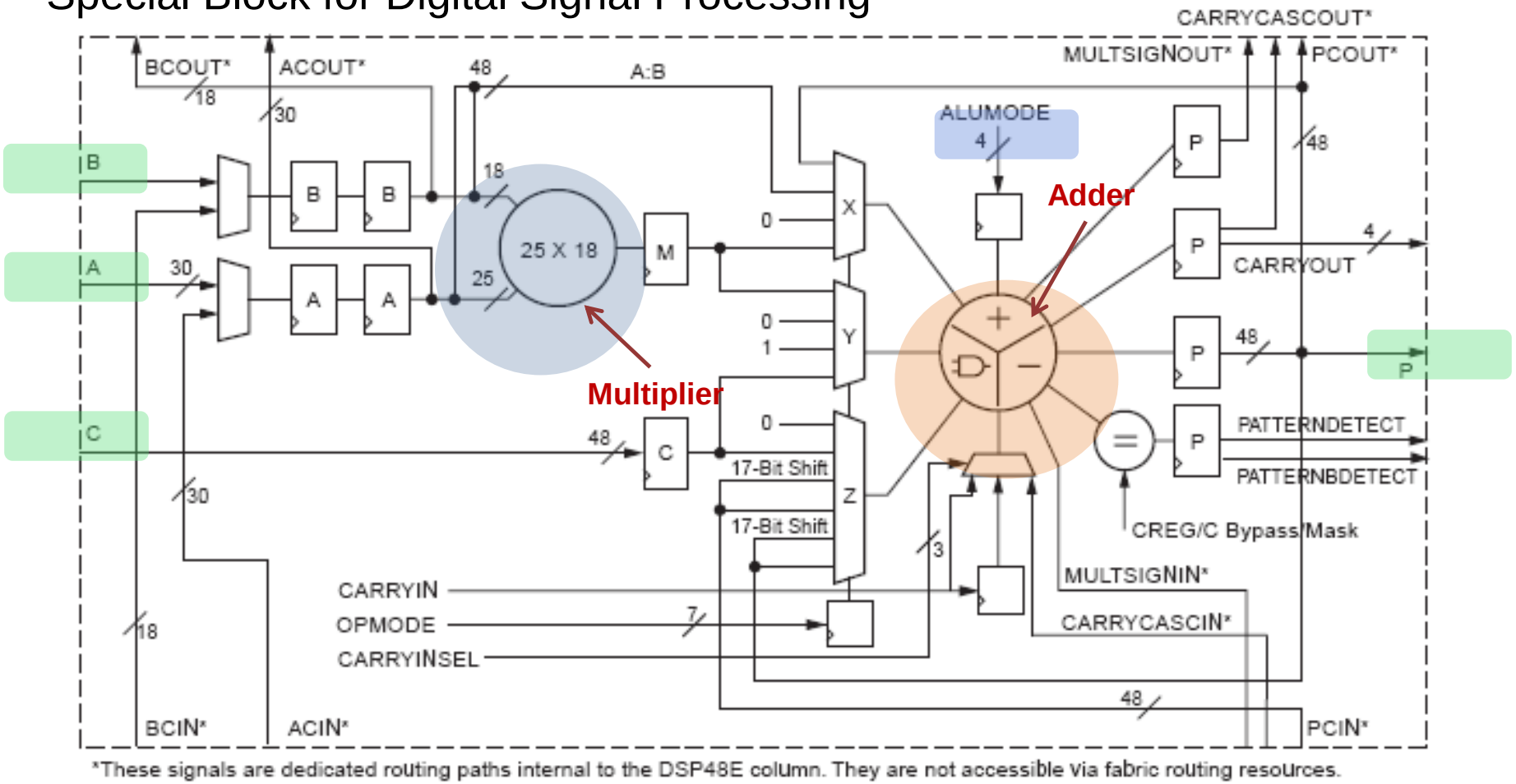
I/O : Multiple standards including fast serial IO up to some Gb/s

Multiplier / DSP48 : 18x18 Multiplier and 48 Bit ADD/SUB for DSP

Processor Cores

Ethernet MACs

Special Block for Digital Signal Processing

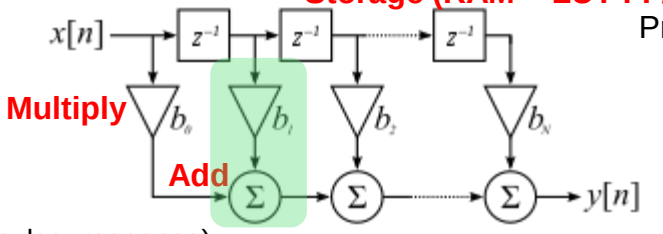


Xilinx Virtex 5 DSP 48

Source: Xilinx Document UG193

Storage (RAM – LUT-FF/Distributed RAM/BRAM ?)

Problem of parallel access



FIR-Filter structure (from: http://en.wikipedia.org/wiki/Finite_impulse_response)

Designmodules (Cores)

Special Hardware <-> Implementation in Fabric

Precompiled <-> HDL-description

Free Cores <-> Licensed IP

Special architecture dependend blocks (ROM, RAM, DPRAM, FIFO, DCM, DSP48..)

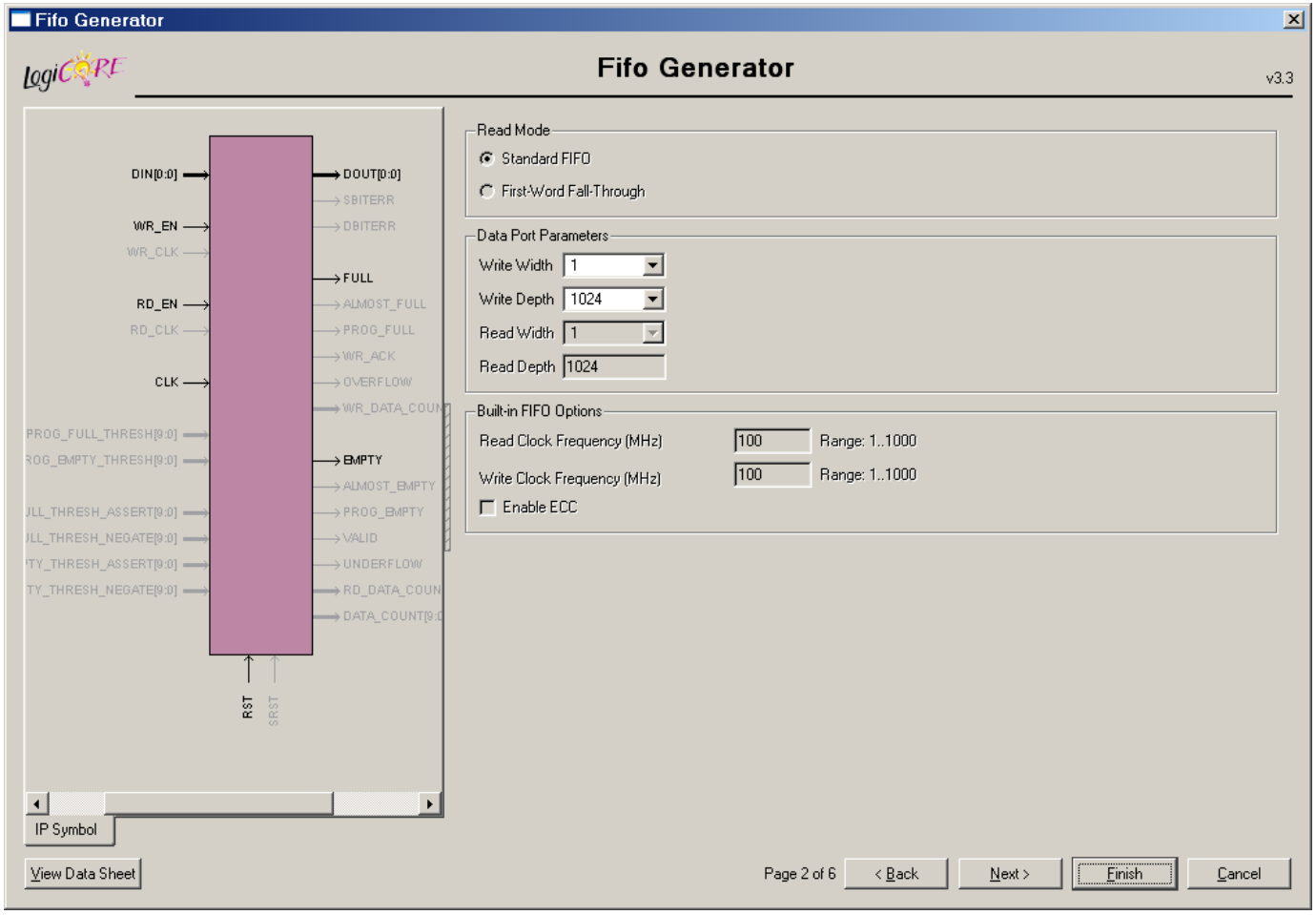
Comparators, Shiftregisters,Counters
Simple Arithmetic
Busstructures

Arithmetic (Floating, Complex, SQRT ..)
Transformation (Fourier, Wavelet …)
Filters

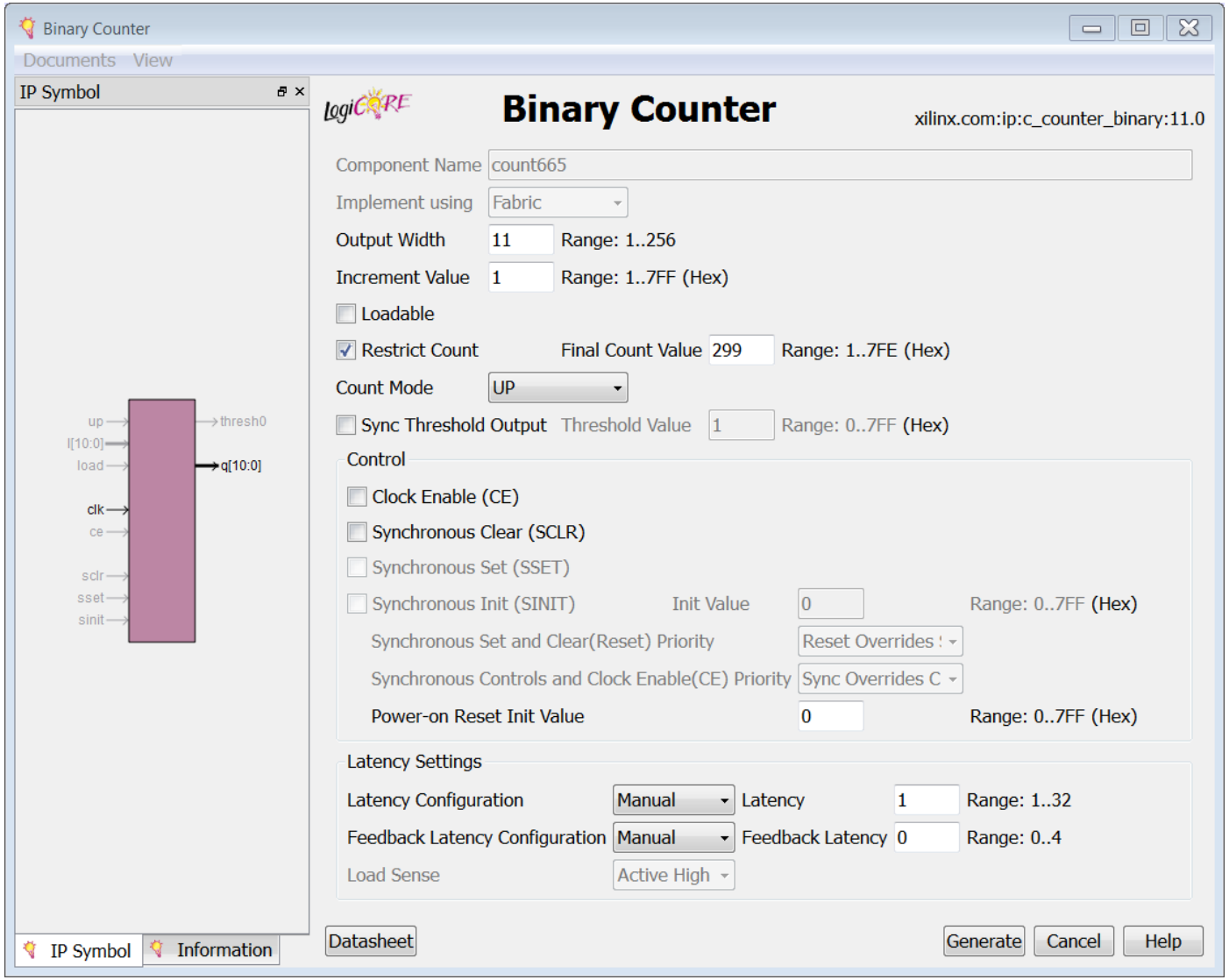
Prozessors (Microblaze …)
Bus-Interface (PCI ..)
Peripherals (UART, CAN, Ethernet, USB …)

Cores available as HDL-blocks, Schematic-symbols, DSP-blocks (Matlab)

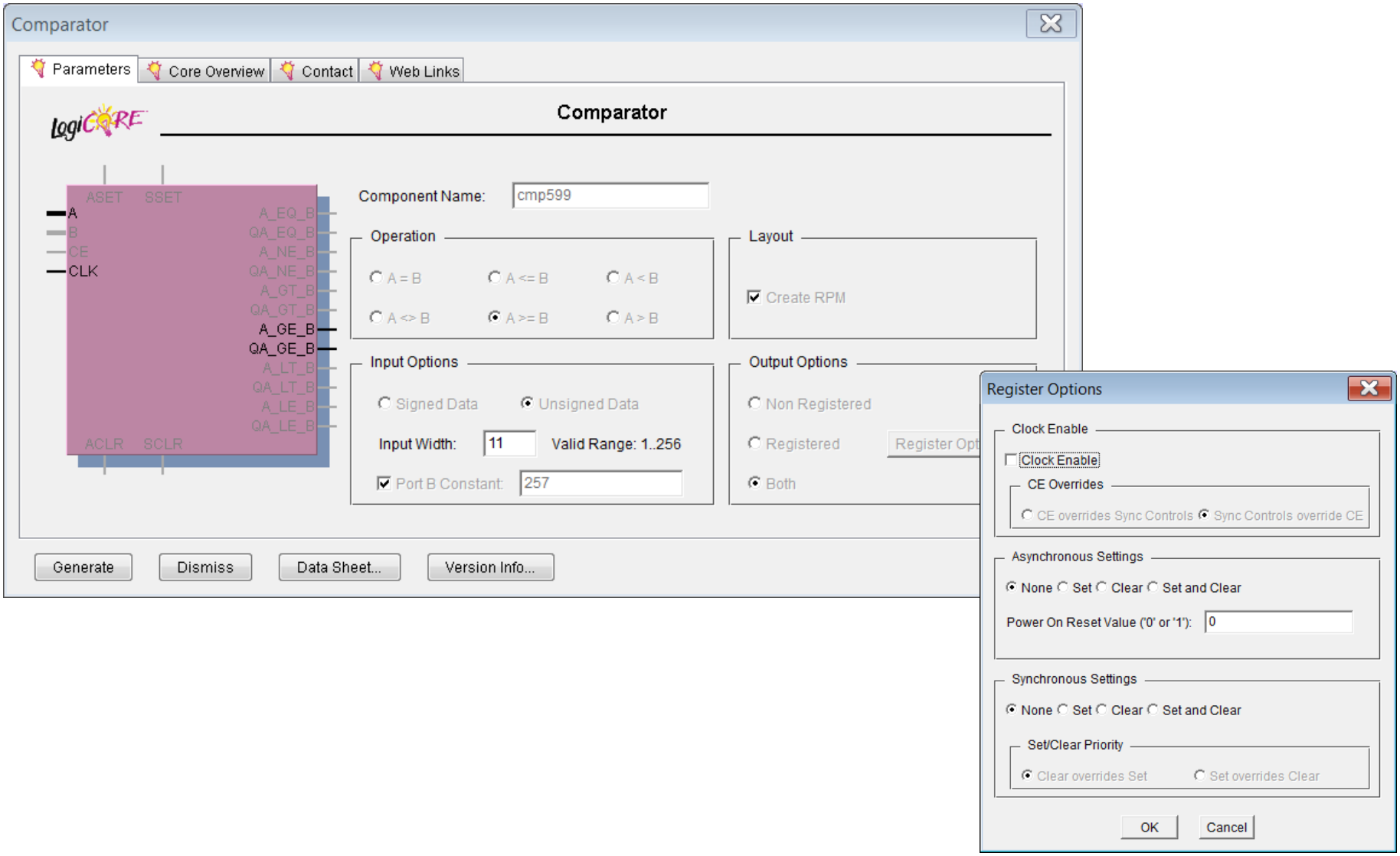
Core Generator (FIFO)



Core Generator (Counter)

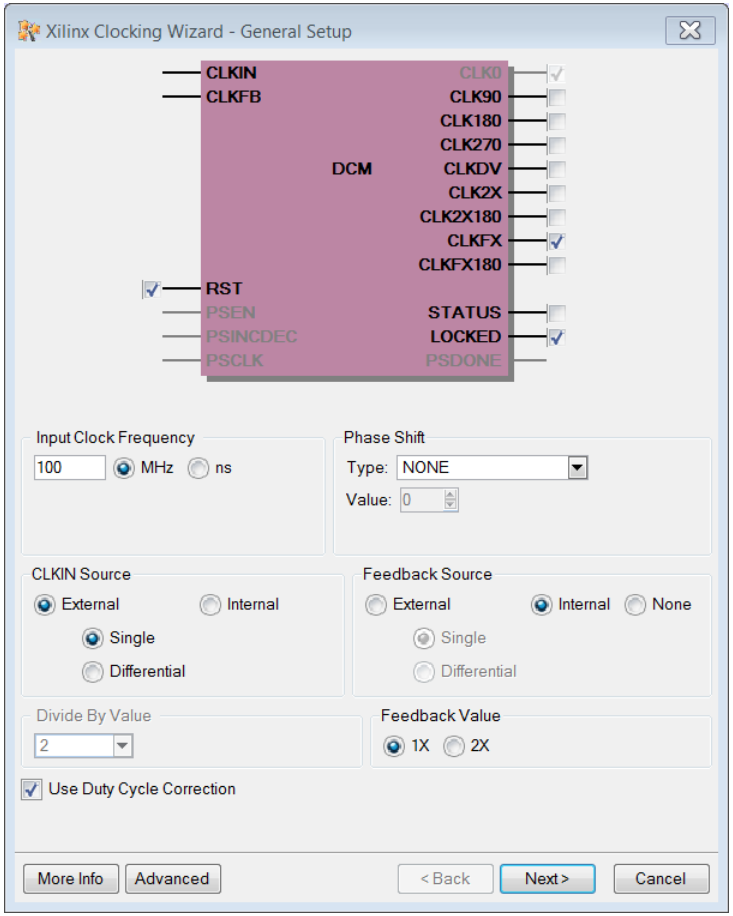


Core Generator (Comparator)

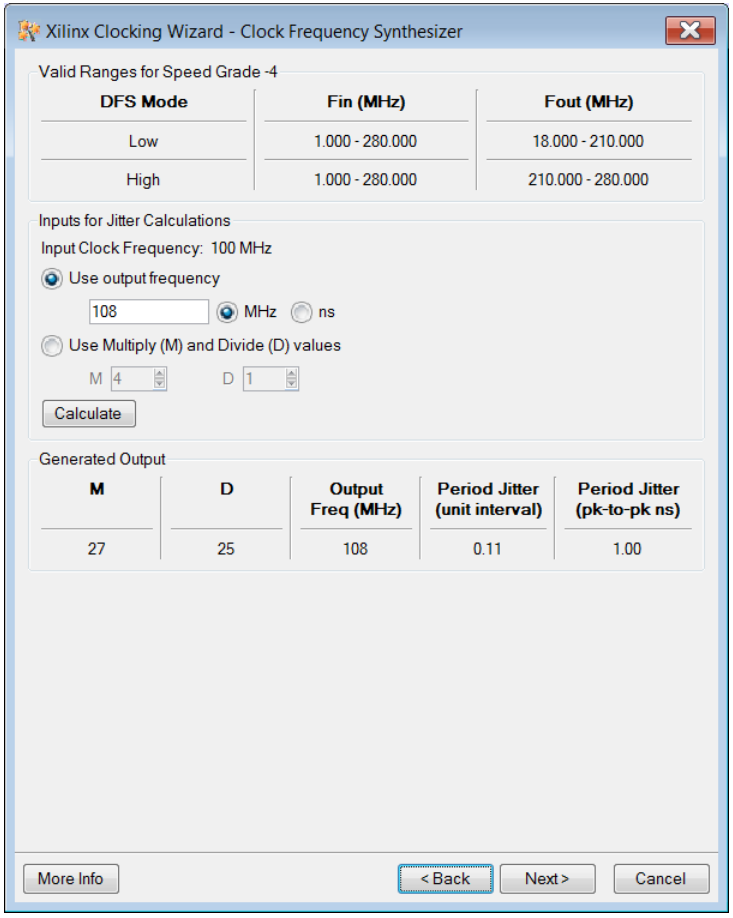


Core Generator (Clocking Wizard)

DCM - Digital Clock Module

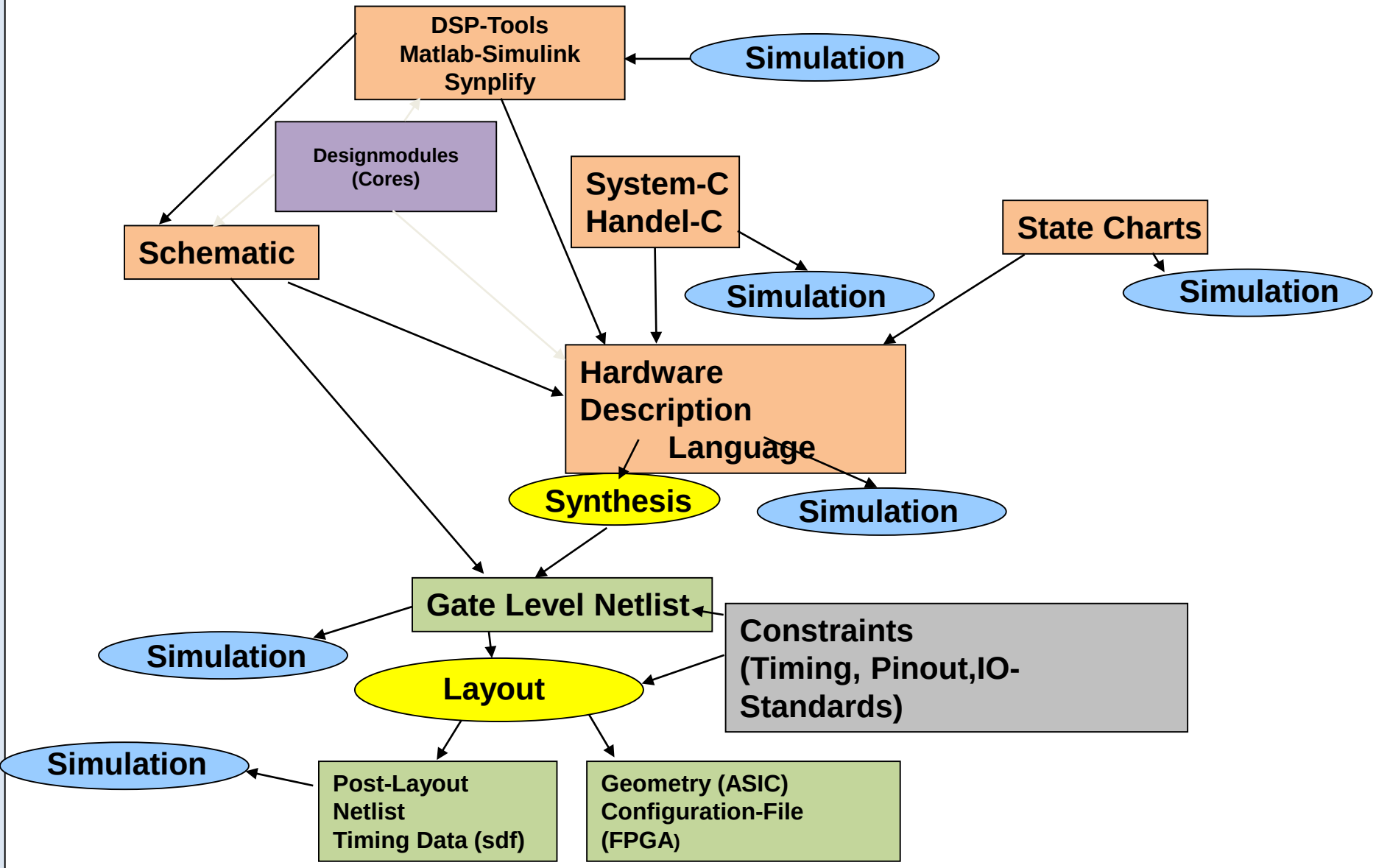


DFS - Digital Frequency Synthesizer



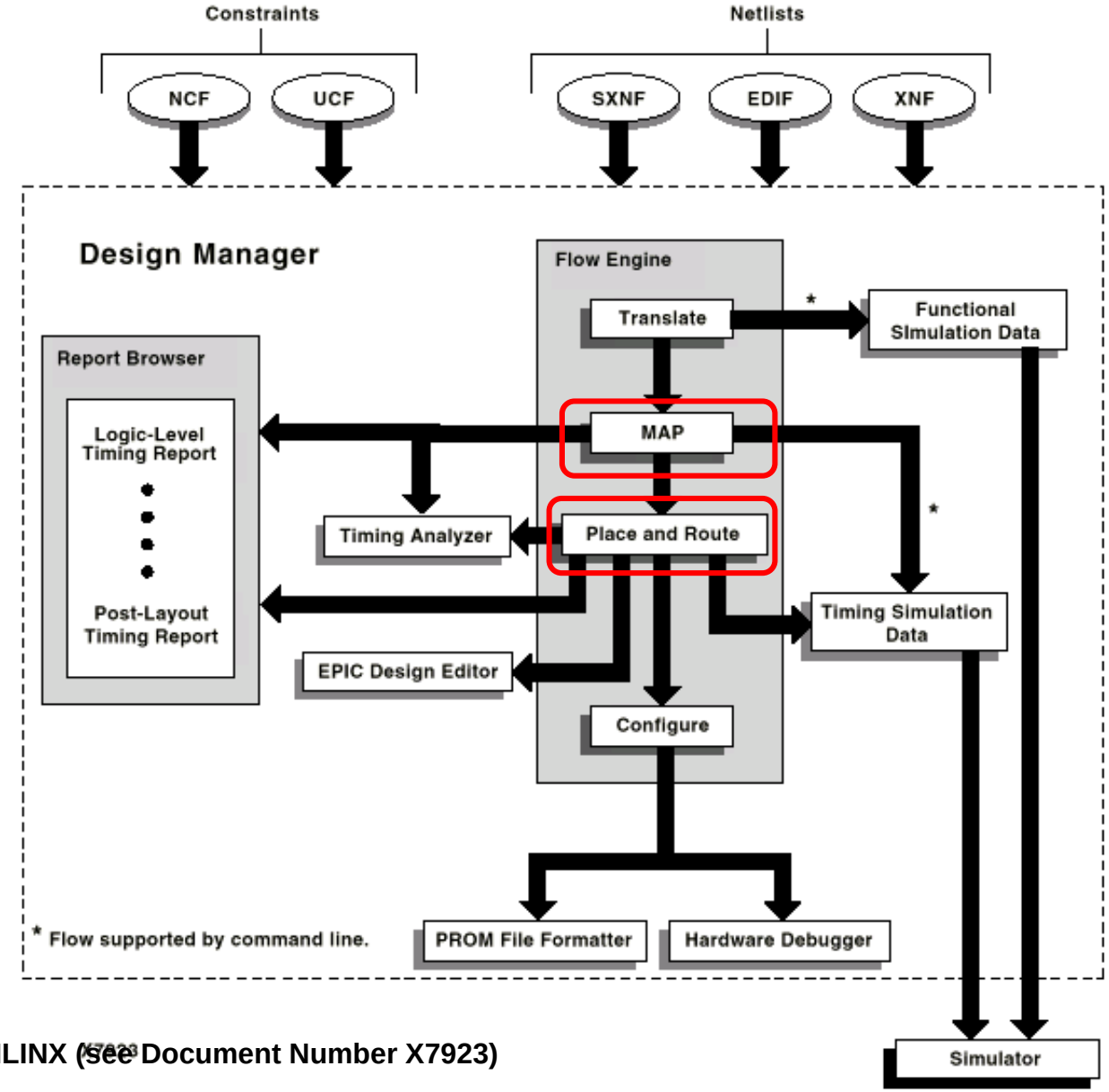
There is another setup page available to select buffers

Design Entry



Design Manager Flow

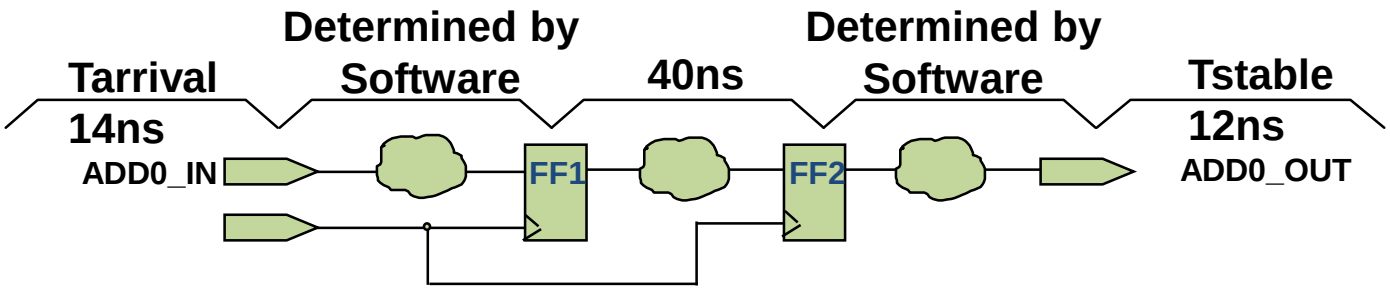
FPGA
Implementation



Source : XILINX (See Document Number X7923)

Synthesis: Synchronous Constraint Example

- OFFSET defines the delay of a signal external to the chip, relative to a clock. Internal clock delays are determined by Software



NET “CLK” PERIOD = 40;
NET “ADD0_IN” OFFSET = IN 14 AFTER CLK;
NET “ADD0_OUT” OFFSET = OUT 12 BEFORE CLK;

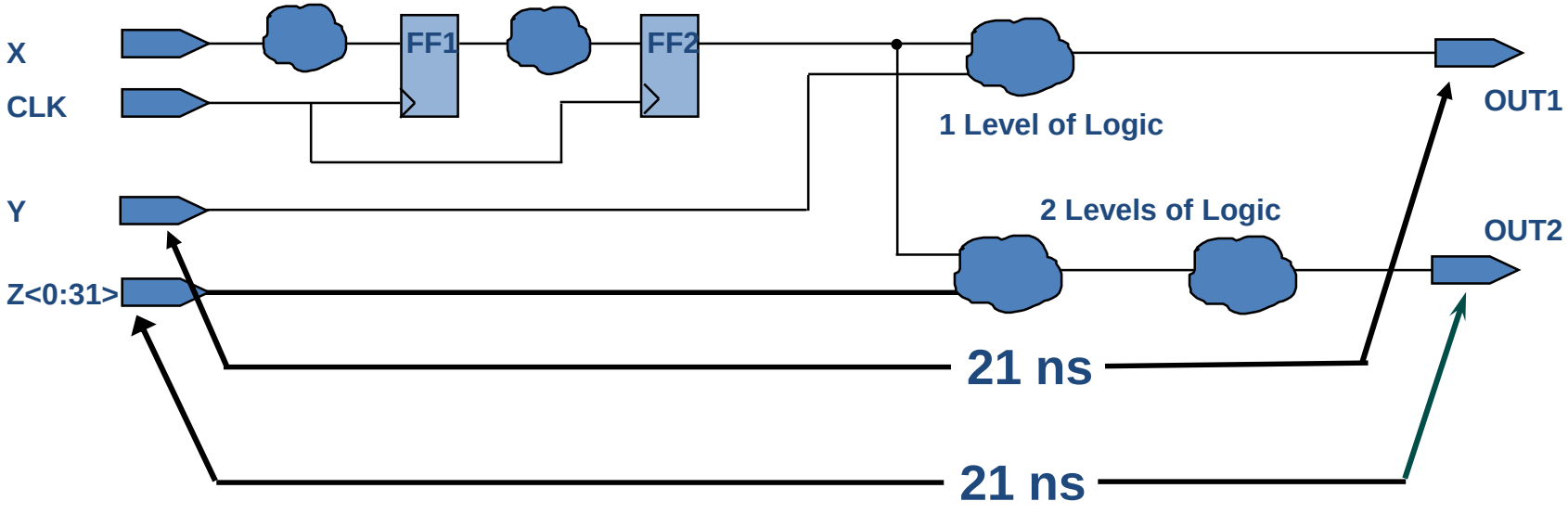
Source : XILINX University Workshops

“FROM-TO” Constraint Example

- Consider the example shown below with TIMESPEC:

```
TIMESPEC TS01 = FROM PADS TO PADS 21;
```

- TS01 is applied to both Y - OUT1 and Z - OUT2.



Source : XILINX University Workshops

Mapping in FPGA

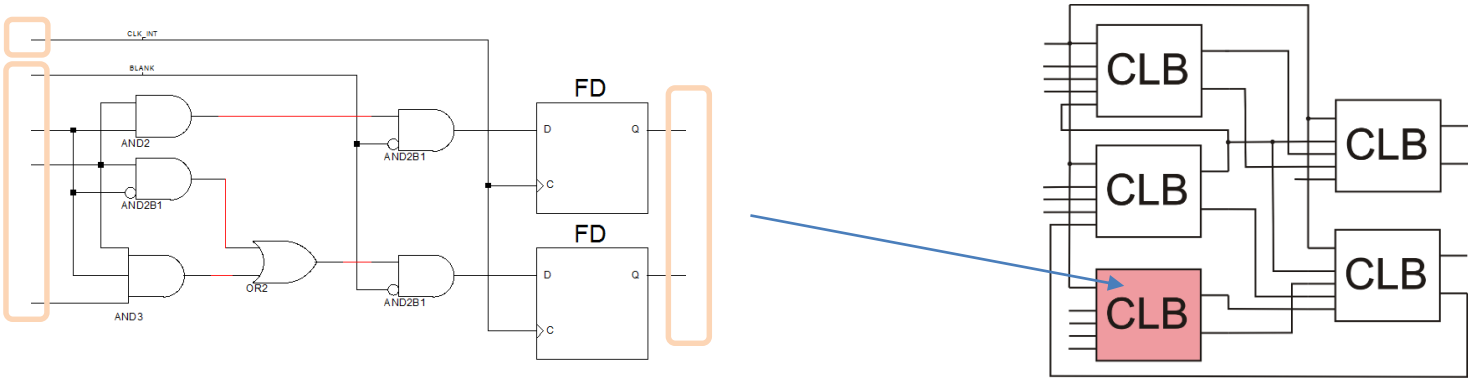
Original netlist is made of **gates, flip-flops**, multiplexers etc.
FPGA comprises **CLB**s with look-ahead tables and flip-flops

Mapping converts the netlist of gates and flip-flops to a new netlist of CLBs

Groups of gates are merged into lookup tables.

Mapping tries to find appropriate resources for special items in the input netlist. Examples: special I/O-cells, buffers etc.
May use Block-RAMs for Logic, may duplicate registers, respects clock regions

Typical error messages: "No sufficient resources.", "No adequate resources available."



The hardware represented by the schematic above
fits in one typical CLB (4-input-lookup-table + 2 FFs)

Observe that some nets (connections between gates) may disappear.
You will not find them anymore in postlayout simulation.
(red marked in the schematic)

Control of FPGA Implementation

Any process may be influenced by setting its properties (Pull-down menu, command line options)

Setting options for Mapping using pull down

Switch Name	Property Name	
-timing	Perform Timing-Driven Packing and Placement	☑
-ol	Map Effort Level	High
-xe	Extra Effort	None
-t	Starting Placer Cost Table (1-100)	1
-logic_opt	Combinatorial Logic Optimization	
-register_duplication	Register Duplication	Off
-x	Ignore User Timing Constraints	☐
-ntd	Timing Mode	Non Timing Driven
-u	Trim Unconnected Signals	☑
-ignore_keep_hierarchy	Allow Logic Optimization Across Hierarchy	☐
-cm	Optimization Strategy (Cover Mode)	Area
-detail	Generate Detailed MAP Report	☑
-ir	Use RLOC Constraints	Yes
-pr	Pack I/O Registers/Latches into IOBs	Off
-c	CLB Pack Factor Percentage	100
-bp	Map Slice Logic into Unused Block RAMs	☐
-power	Power Reduction	
-activityfile	Power Activity File	
	Other Map Command Line Options	

→ will activate the greyed options

or "Speed" or "Power"

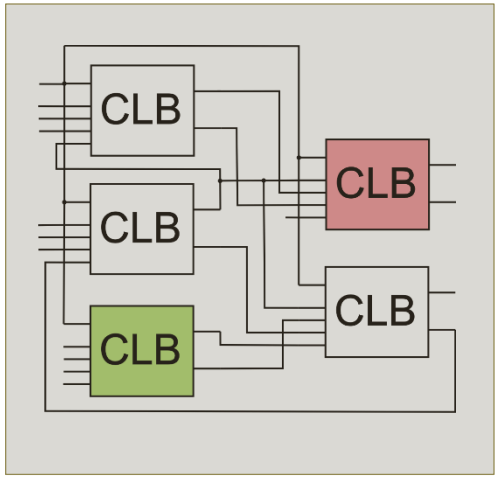
switching activity data from simulator

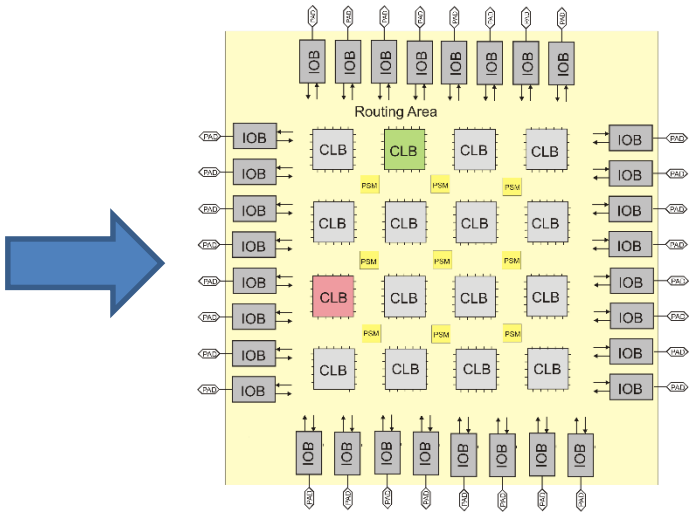
Command line:

map -intstyle ise -p xc6slx9-ftg256-3 -w –timing on -logic_opt off –ol high -t 1 -xt 0 -register_duplication off -r 4 -global_opt off -mt off -ir off -pr off -lc off -power off -o vga_map.ncd vga.ngd vga.pcf

Placement and Routing in FPGA

No totally free placement in FPGA: CLBs are placed in a fixed grid
Placement is more a mapping of the CLBs in the netlist to the available CLBs in the grid

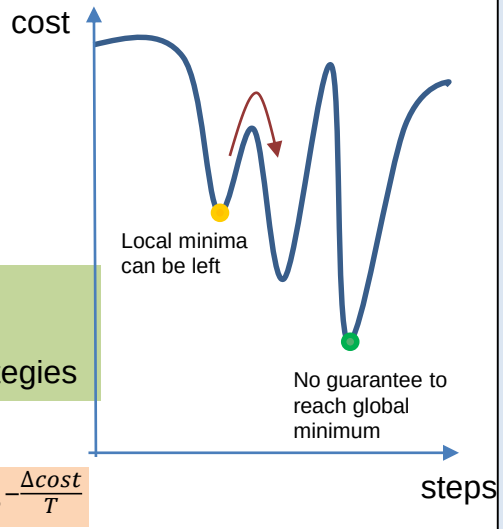




Find the best location/placement for the CLBs.
Different methods: Force Direct, Neural Networks, **Simulated Annealing**

1. Define cost function: Example: $\sum w_i l_i^m \rightarrow min$
2. Initial placement, Set T ("temperature") to a high value
3. Evaluate cost
4. Change placement by swapping
5. Evaluate new cost and compare with old value
6. Use new placement if cost are less or with a probability p even if cost are worse $p = e^{-\frac{\Delta cost}{T}}$
7. Decrease Temperature by a factor (Starting T and decrease factor have great influence on the results)
8. Repeat steps 4 to 7 as long as a minimal temperature or cost equilibrium is reached

w_i - weight for the net i
l_i - length of net-wire I
m – exponent for different strategies



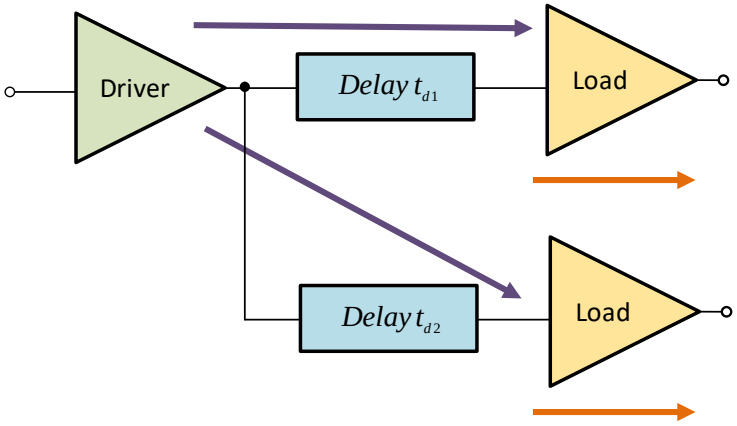
Different runs give different results. Time-consuming algorithms. May take hours. Parallel runs possible.

Placement and Routing in FPGA

Routing in FPGA: Use of predefined wires (short, long), programmable connection points and switch matrices
Difference to Custom-IC with free areas that can be used for wires with any shapes

Different algorithms available.
Two steps: Global and Local routing

After placement and routing the delay times can be extracted.
FPGA-vendor libraries contain data for wire delays and driver delays dependent on load (wires, inputs).



Lumped Model (extracted delay times)

Delay times will be backannotated for timing simulation.

Data Stream for FPGA Configuration

Configuration of the bitfile has many options (encryption, compression, configuration speed, exact timing)

Configuration file sequence (.bit-file)

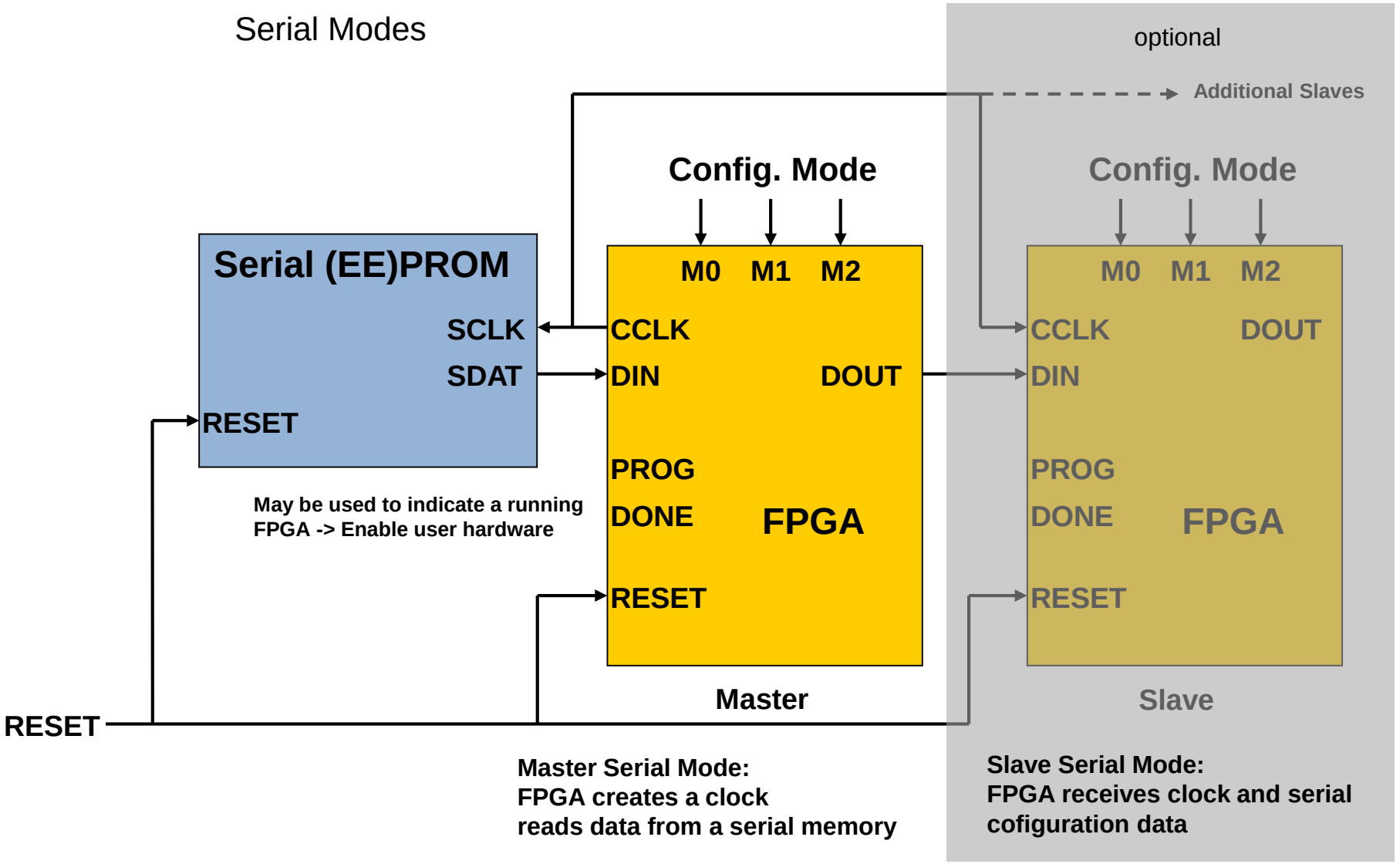
Synchronization
Register packets (CMD, frame length, options, clock rate) - one 32-bit-word
Data packets up to 1 meg 32-bit-words transmitted in frames of frame_length words
CRC-register-packets

BRAM-data transmitted in separate frames (may be omitted if not used -> faster configuration)

Configuration bitstream my be compressed (usefull if FPGA partially unused)

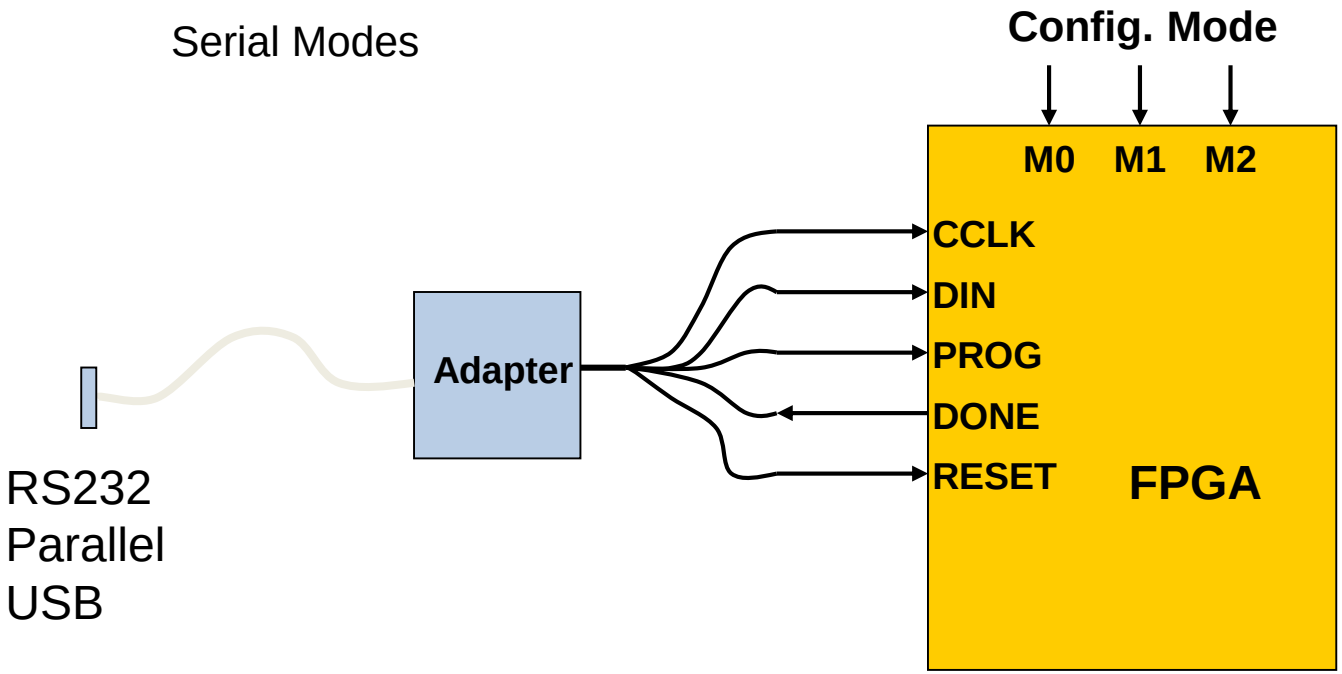
Configuration bitstream may be encrypted.
Special key loading and storing mechanisms available (Battery backup RAM, eFuses)

FPGA-Configuration 1



FPGA-Configuration 2

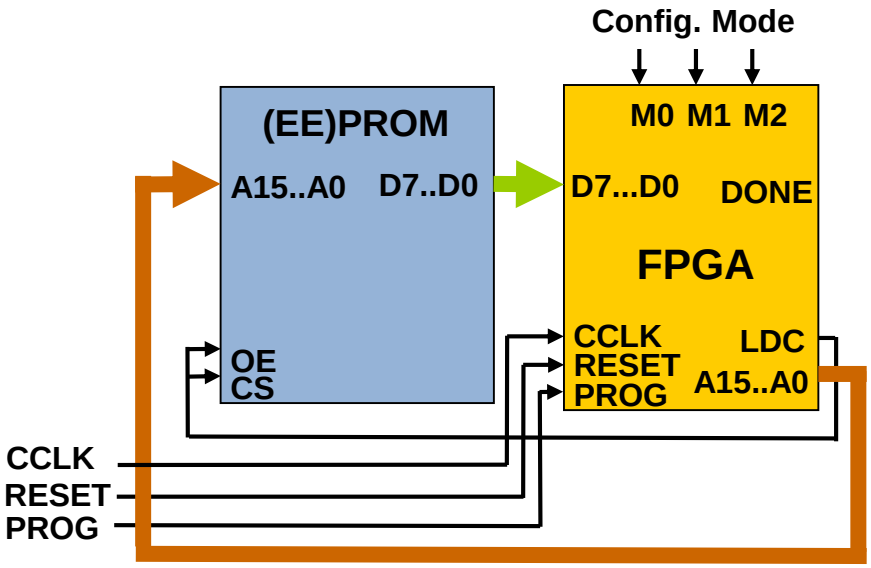
Serial Modes

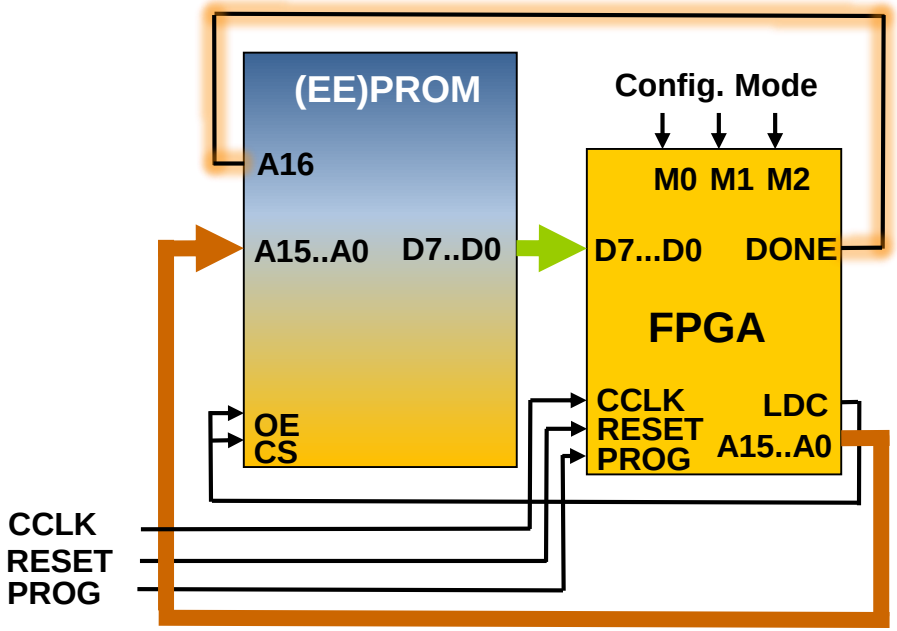


Slave Serial Mode: **FPGA receives clock and serial cofiguration data**

FPGA-Configuration 3

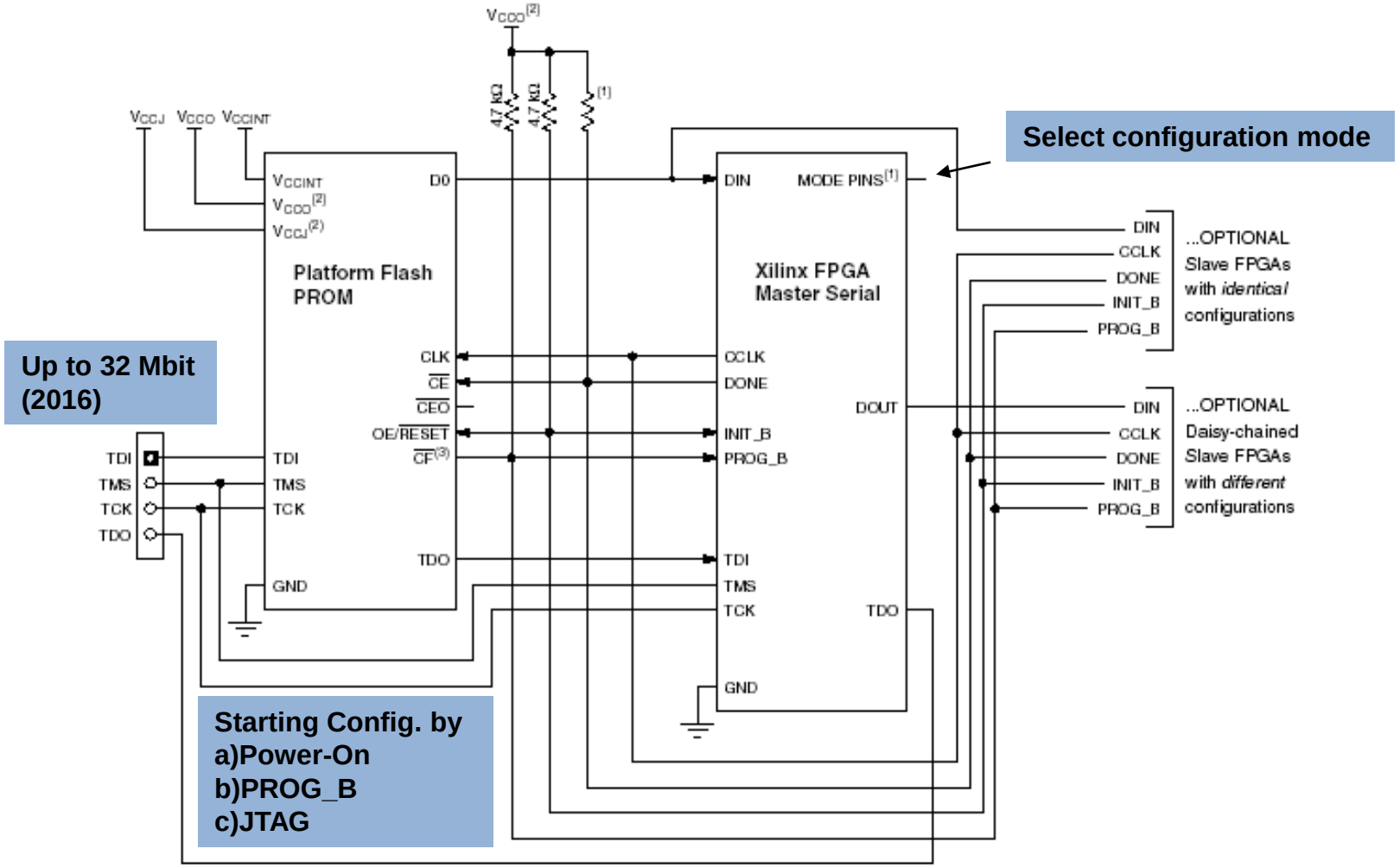
Parallel Mode :
FPGA reads parallel PROM by 1/8 CCLK (internal distribution by CCLK)





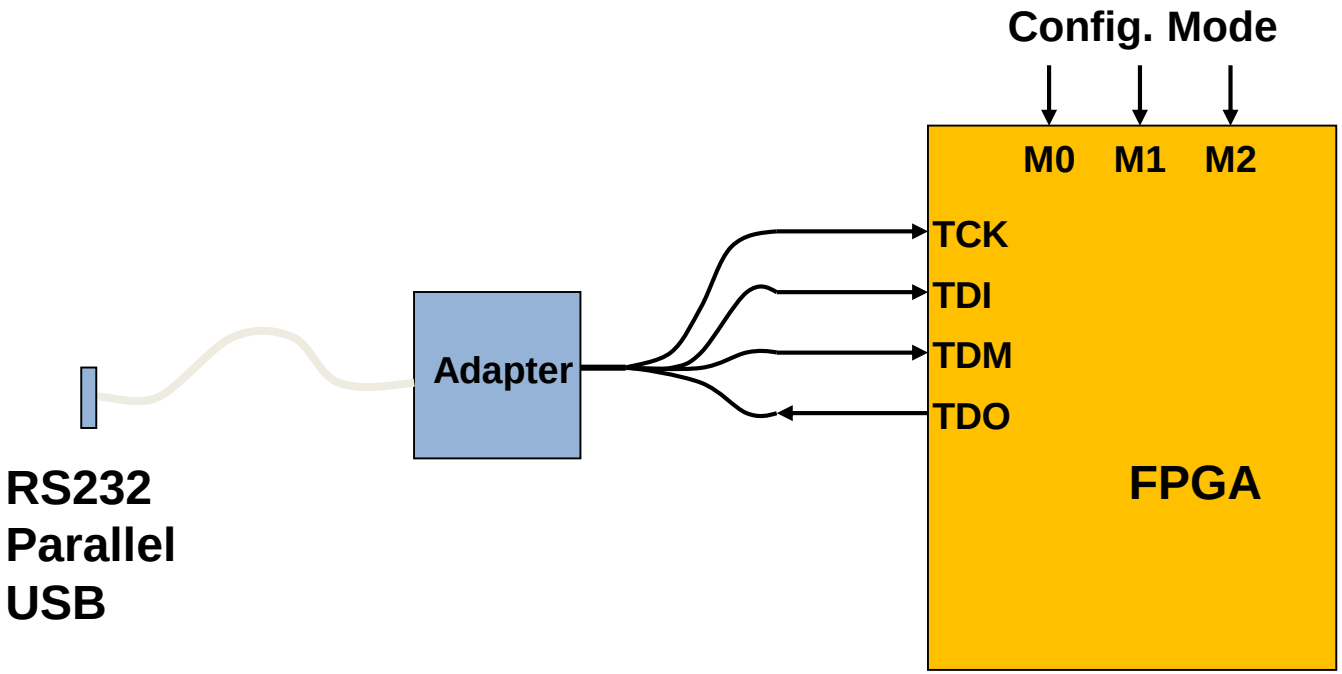
Memory may hold configuration and user design data,
here switched beetween two memory halves
using the DONE-Signal and MSB-address of memory

Serial FPGA-Configuration with "Platform Flash Memory"



Source: XILINX: „Platform Flash In-System Programmable Configuration PROMS“

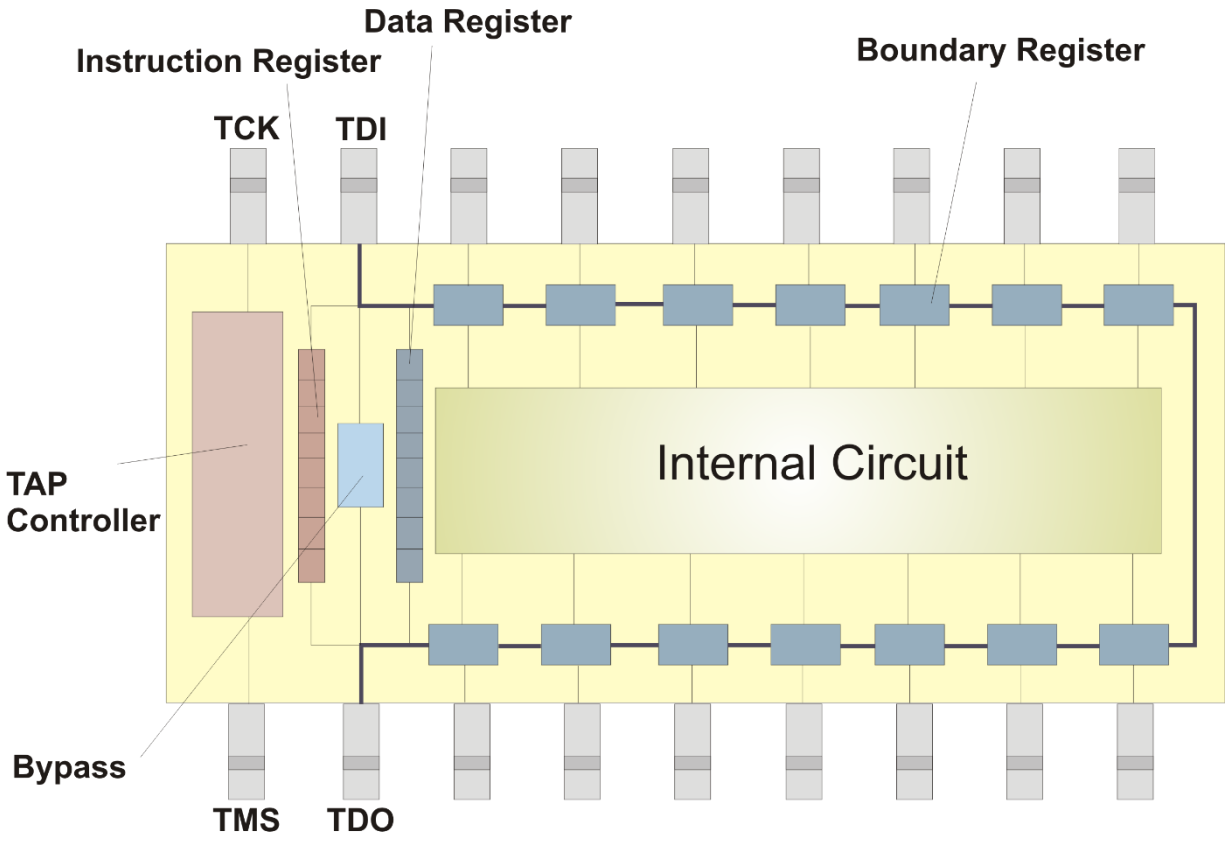
FPGA-Configuration JTAG-Interface



Configuration by JTAG Interface

IC with Boundary Scan (JTAG Interface)

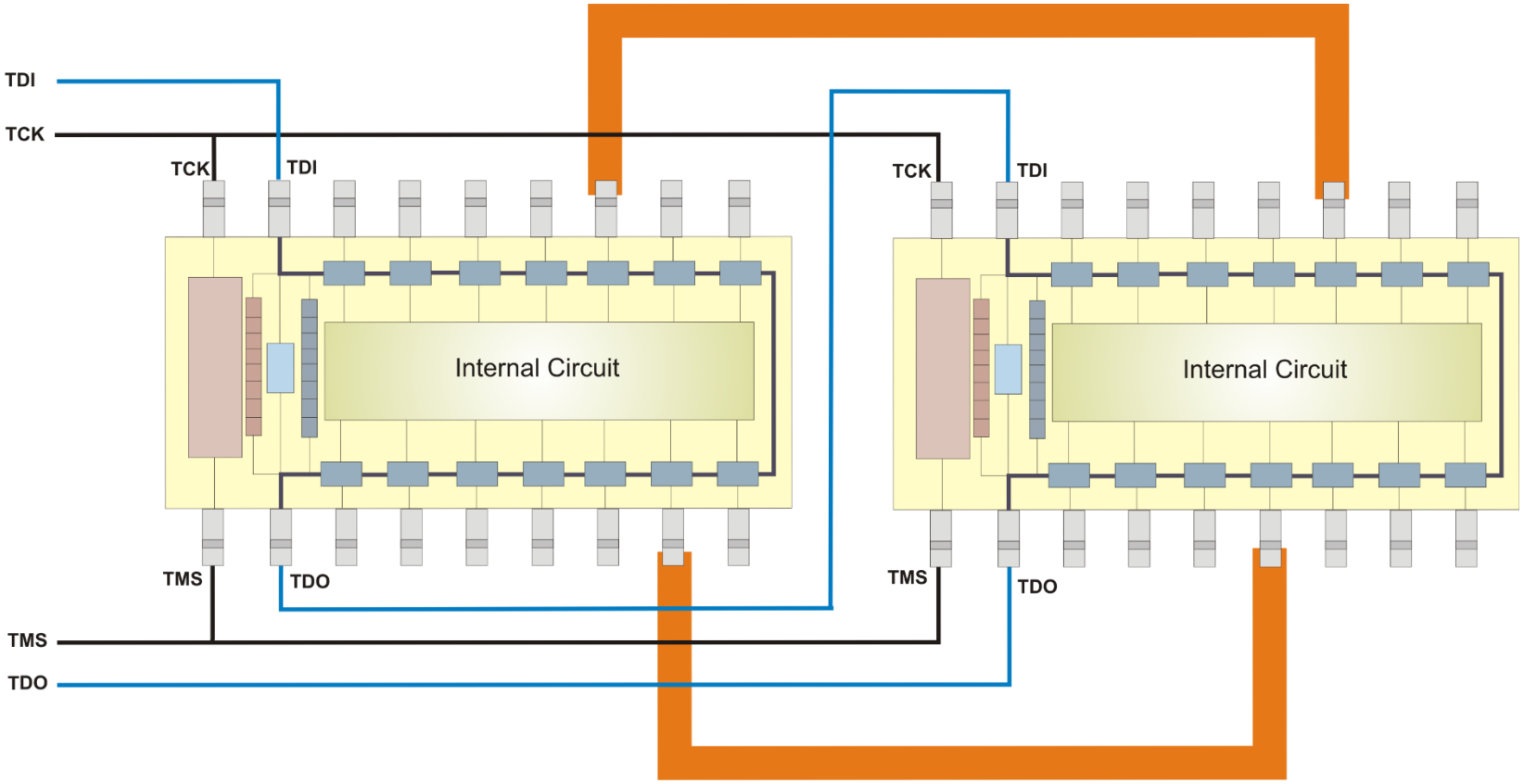
Joint European Test Action Group



IC architecture according to IEEE 1149.1

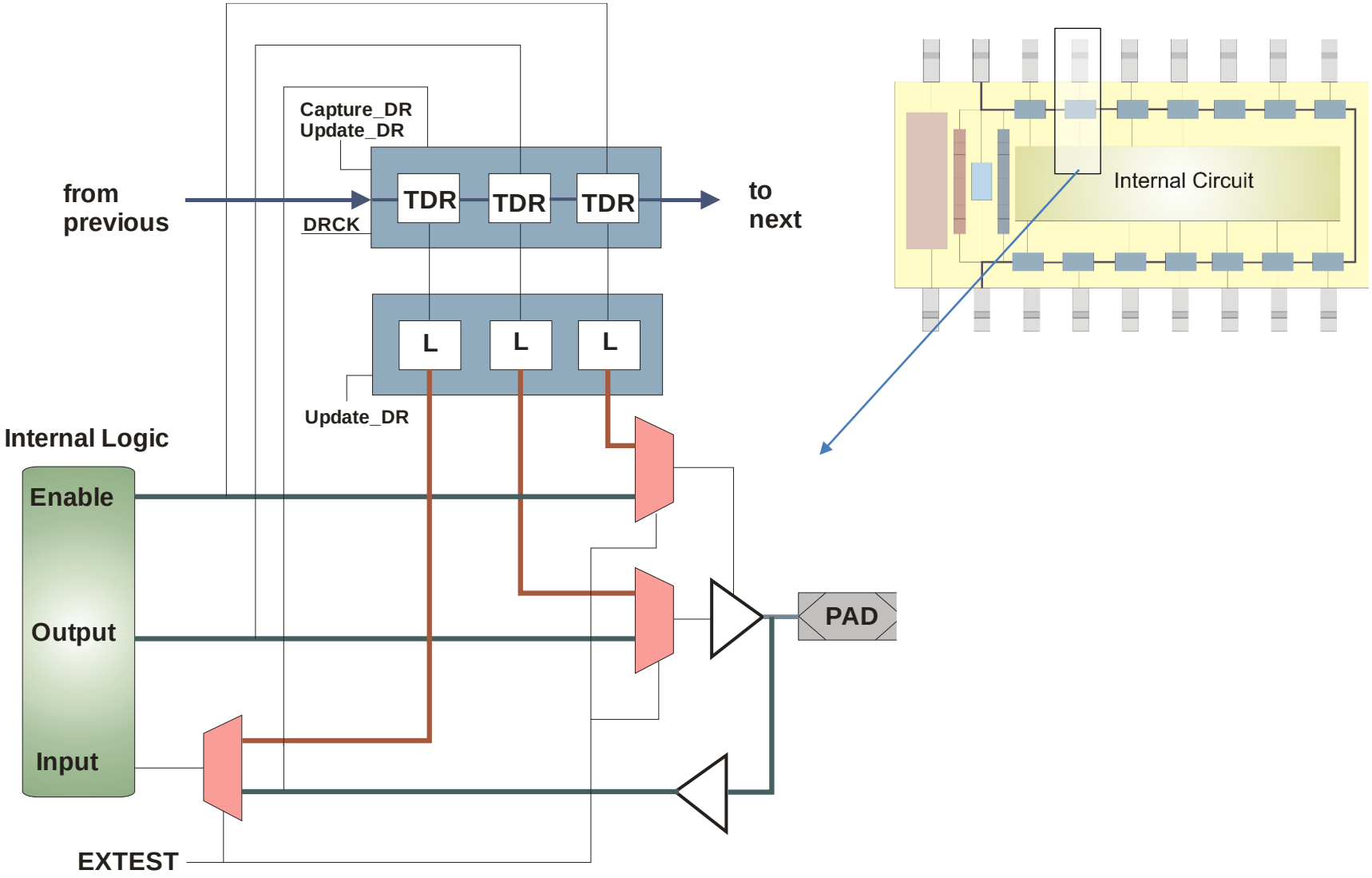
Standard IEEE1149.1 since 1990

Extest

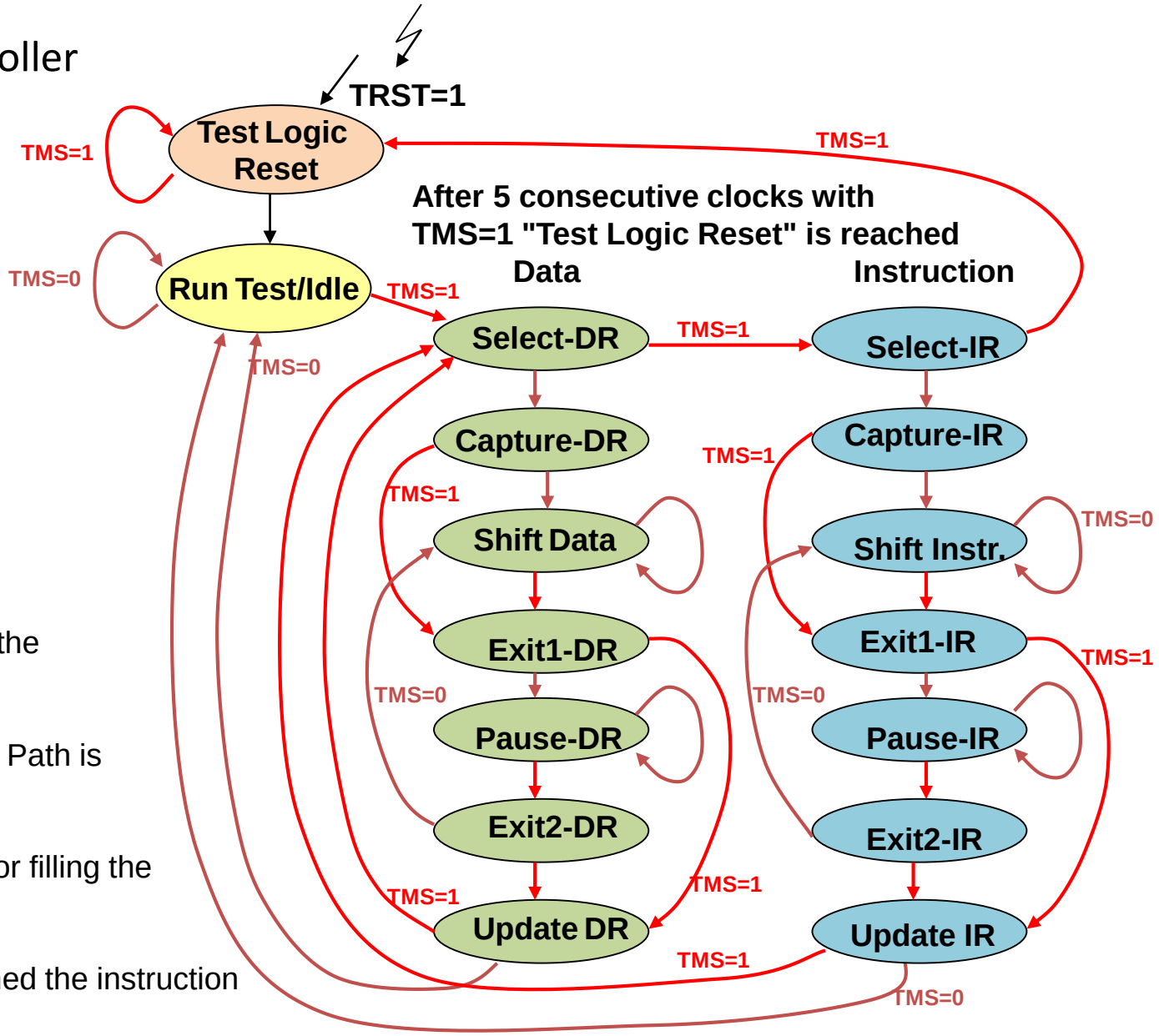


With EXTEST external connections (and/or) logic are tested

Boundary Scan Cell



JTAG TAP-Controller



TCLK and TMS control the State machine

Data Path or Instruction Path is executed

Shift → Exit2 is a loop for filling the Registers

When "Update" is reached the instruction is executed

JTAG- Instructions 1149.1 and IEEE 1532

1149.1
BYPASS
SAMPLE
PRELOAD
EXTEST
INTEST
RUNBIST
CLAMP
HIGHZ
IDCODE
USERCODE

ADDITIONAL INSTRUCTIONS (1532)
for In System Configuration
ISC_ENABLE
ISC_DISABLE
ISC_PROGRAM
ISC_NOOP

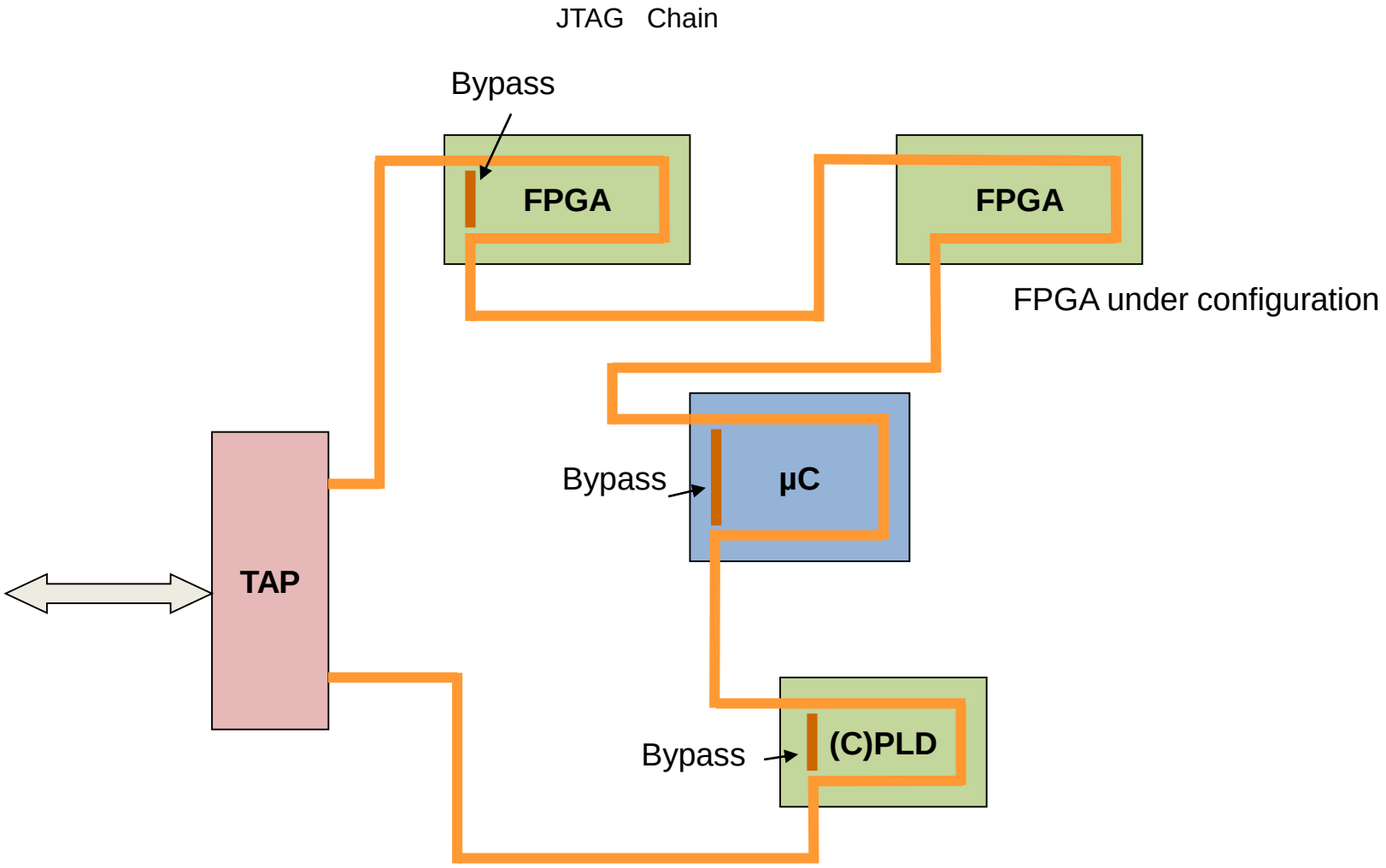
ISC_ERASE
ISC_READ
ISC_PROGRAM_USERCODE
ISC_PROGRAM_SECURITY
ISC_READ_INFO
ISC_DISCHARGE

ISC_ERASE_DONE
ISC_PROGRAM_DONE
ISC_SETUP
ISC_ADDRESS_SHIFT
ISC_DATA_SHIFT
ISC_INCREMENT

ADDITIONAL REGISTERS
ISC_ADDRESS
ISC_DATA
etc.

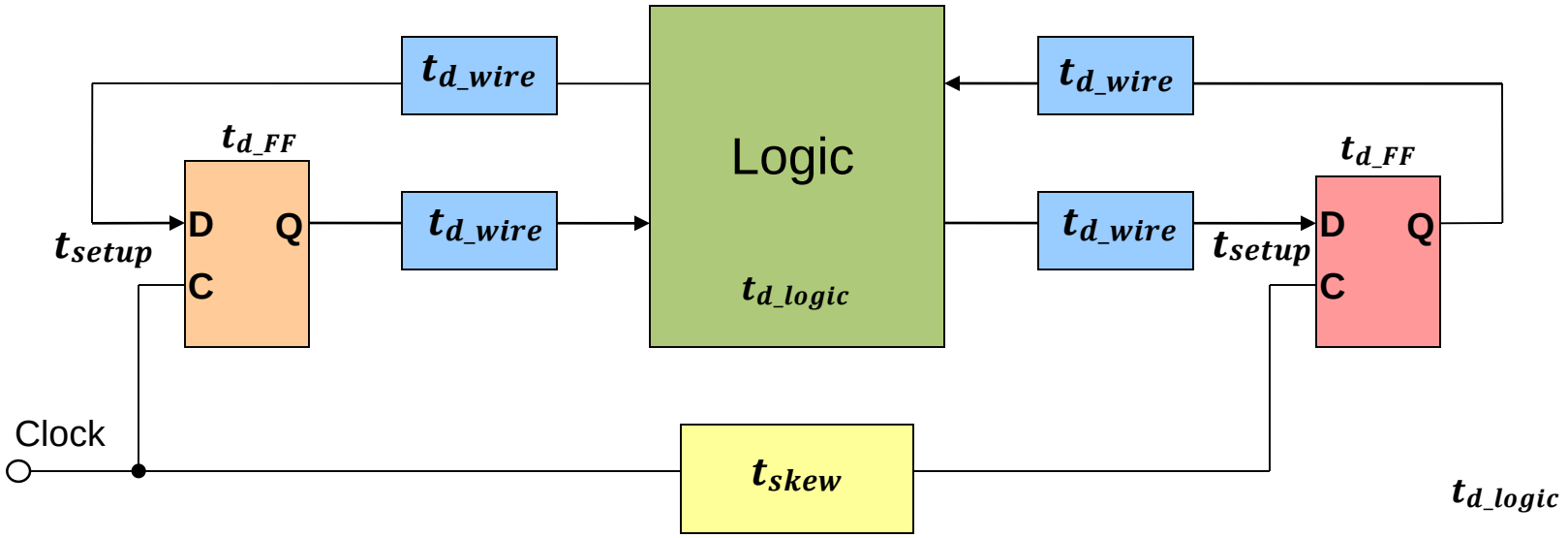
Boundary Scan Description Language

Detailed Information for Devices in BSDL Files (one per device)
special VHDL-Files

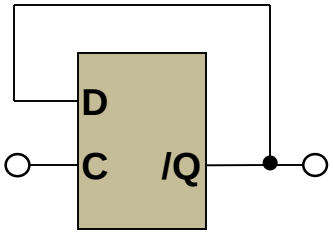


Configuration/Programming of multiple Devices in a JTAG Chain

Estimating the maximum Clock Frequency

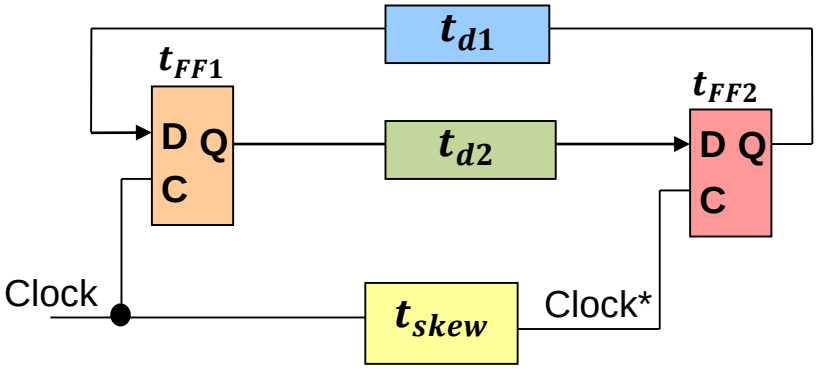


$$f_{max} = \frac{1}{\max\left[t_{setup} + t_{d\_FF} + \sum t_{d\_wire} + t_{d\_logic} + t_{skew}\right]}$$



Toggle-Frequency:
max. frequency of a FF
with simple feedback

Clock Tree Problems

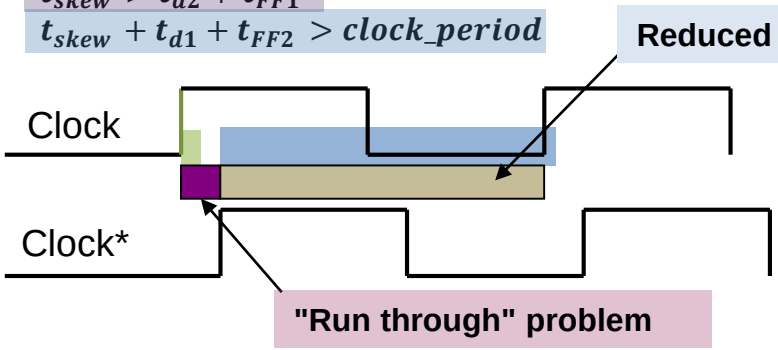


Construct clock nets for minimal clock skew

2 problematic situations:

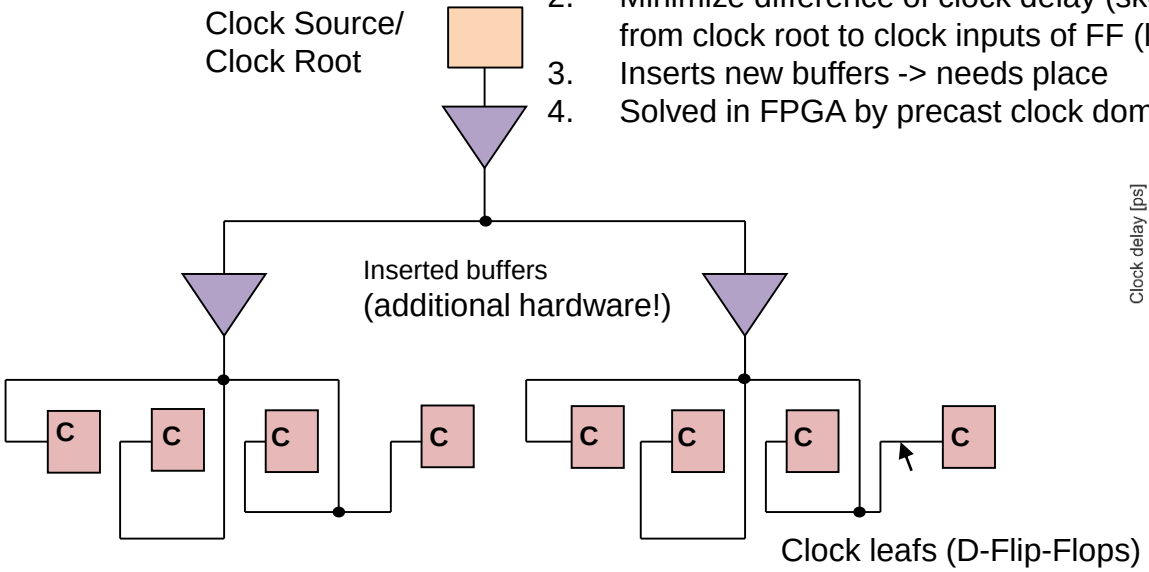
$t_{skew} > t_{d2} + t_{FF1}$

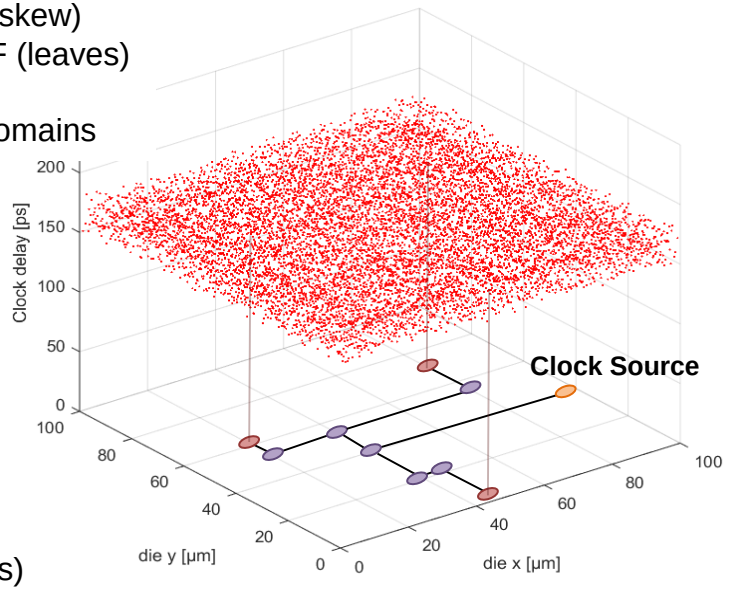
$t_{skew} + t_{d1} + t_{FF2} > clock\_period$



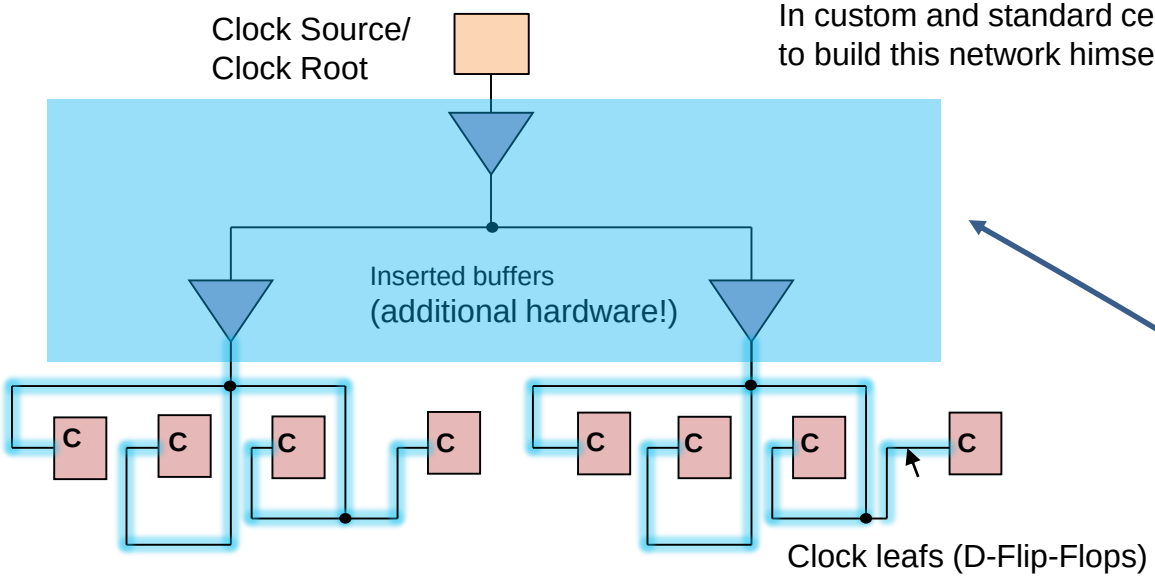
Synchronous Design:
At a certain clock edge new date values are calculated and will be valid at the inputs at the **next** clock edge

1. Limit number of clock inputs per net
2. Minimize difference of clock delay (skew) from clock root to clock inputs of FF (leaves)
3. Inserts new buffers -> needs place
4. Solved in FPGA by precast clock domains





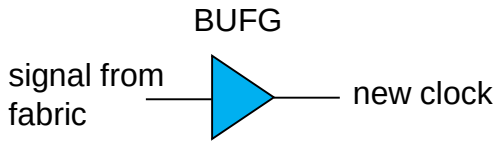
Clock Tree Problems

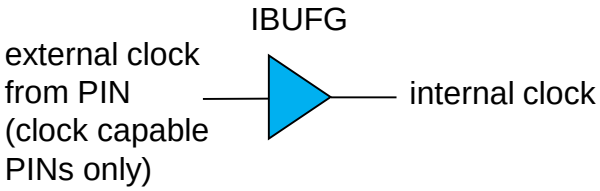


In custom and standard cell IC developer has
to build this network himself (by help of supporting software)

In FPGA groups of Flip-Flops are furnished with such networks.

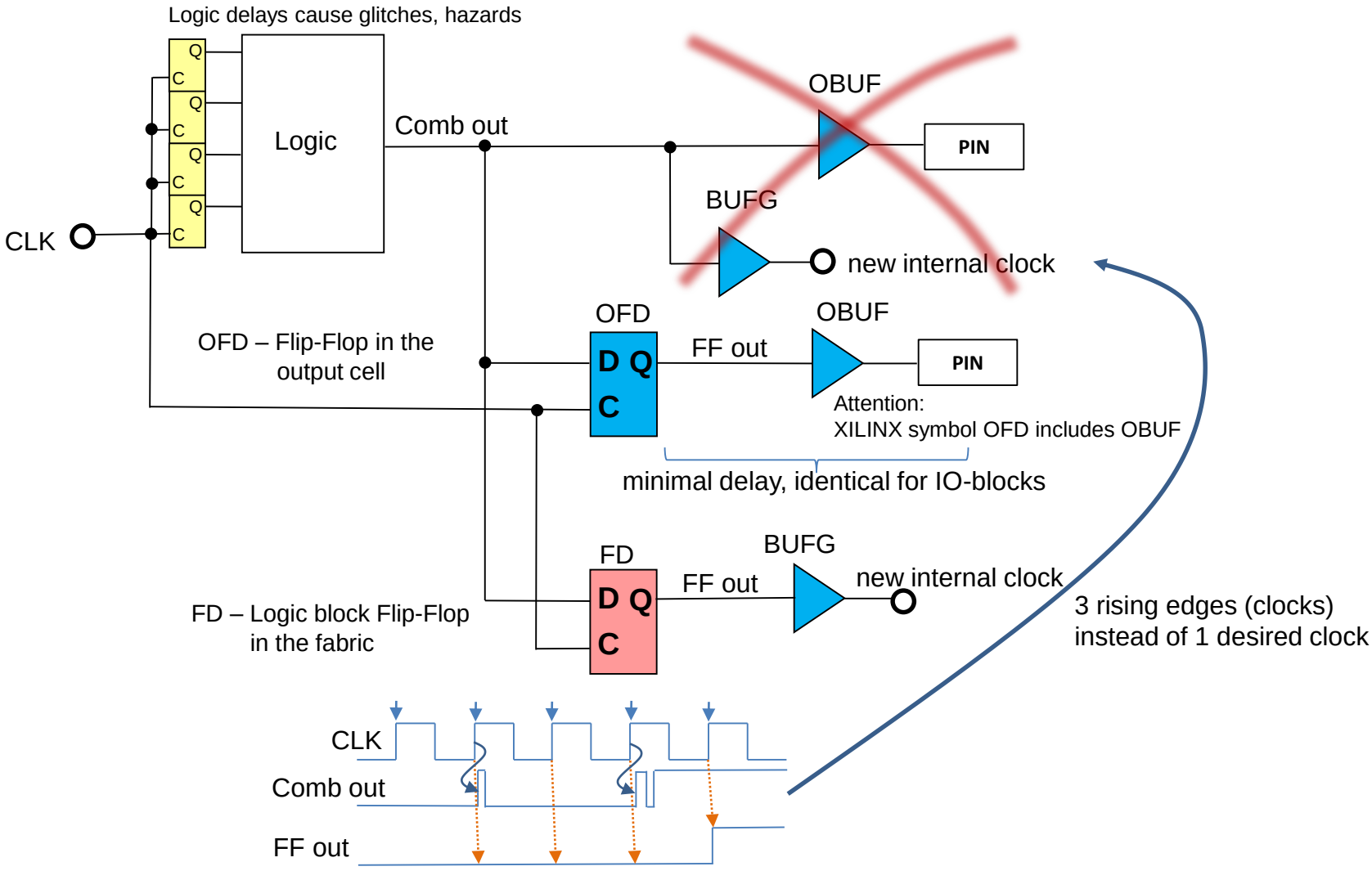
If the developer uses special clock buffers there is inserted
not only a single buffer but a complete clock tree. The group builds
a clock region. There may be only one or up to 20 or more of such regions.
All the Flip-Flops using one clock will be mapped and placed so that they
are connected to a usable clock tree (into one clock region).
Combining of regions possible-> Multi region clocking.



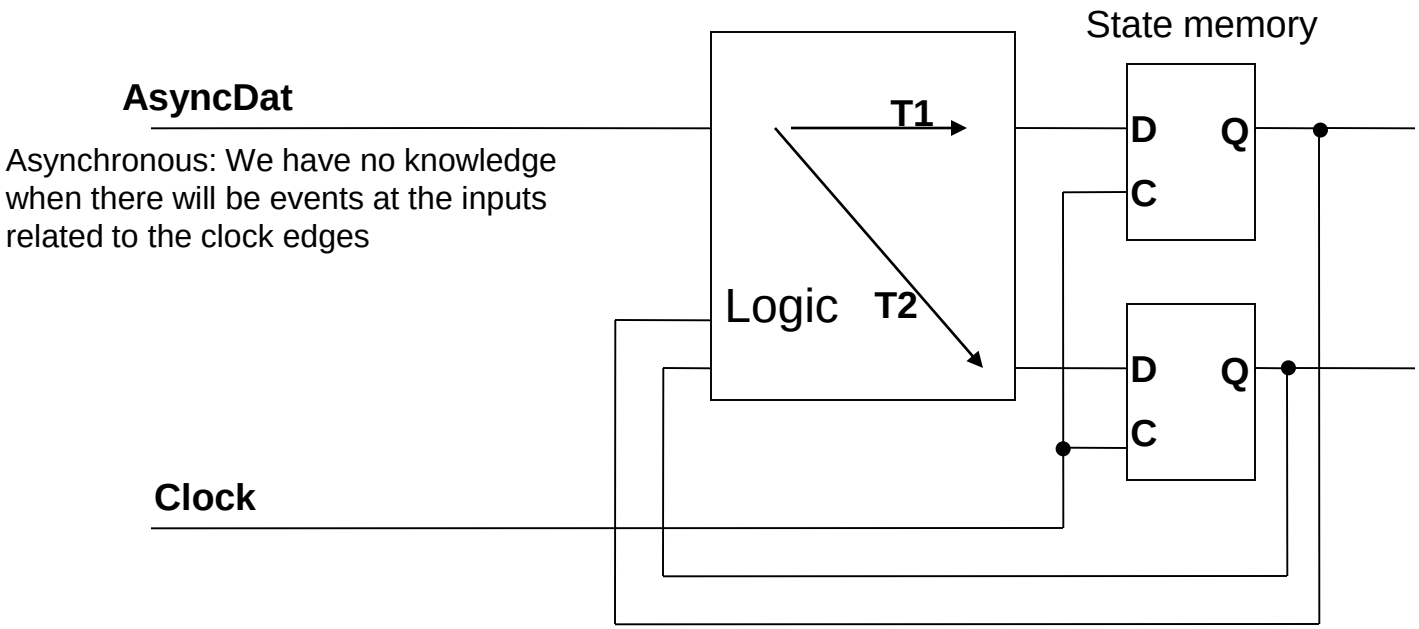


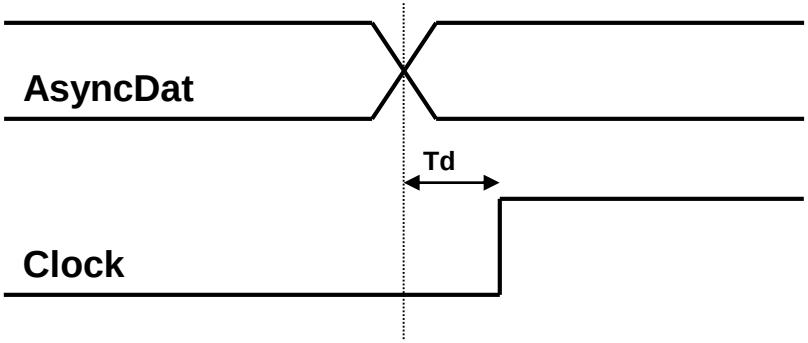
There are more special clock drivers BUFH, BUFR, BUFMR, BUFIO …

Avoiding Glitches



Injput Synchronisation (Sampling) 1

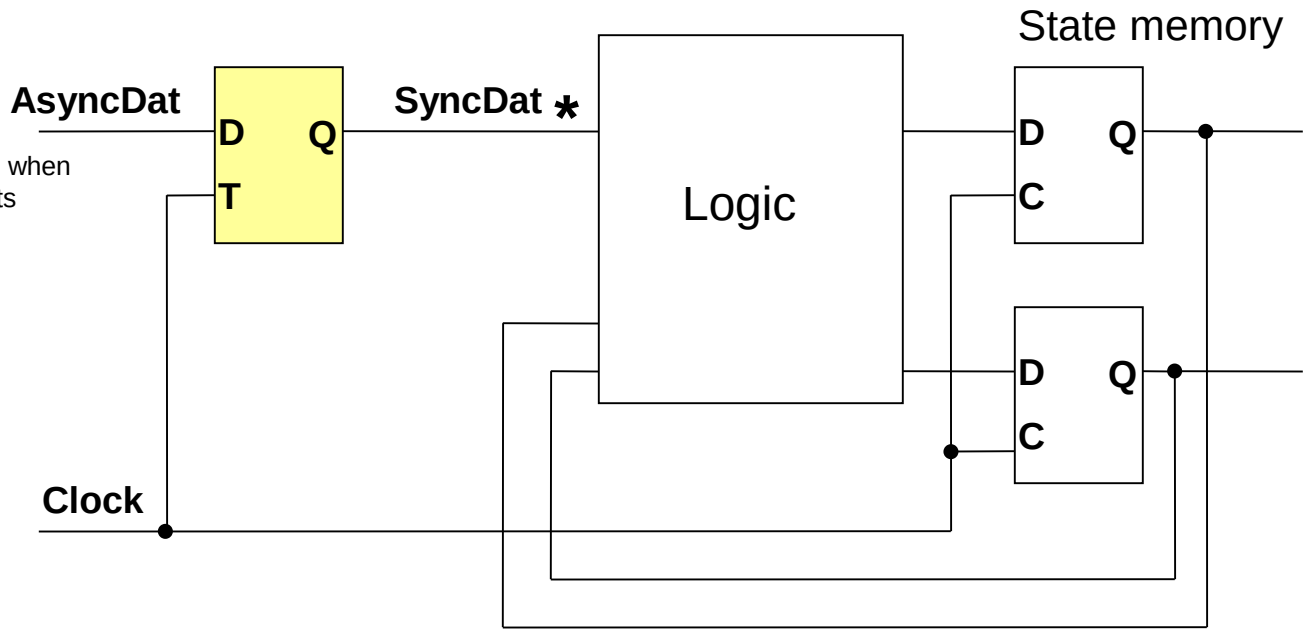




Problem, if there is a difference beetween T1 and T2 one FF may sample the old value other FF the new one: FSM not consistent

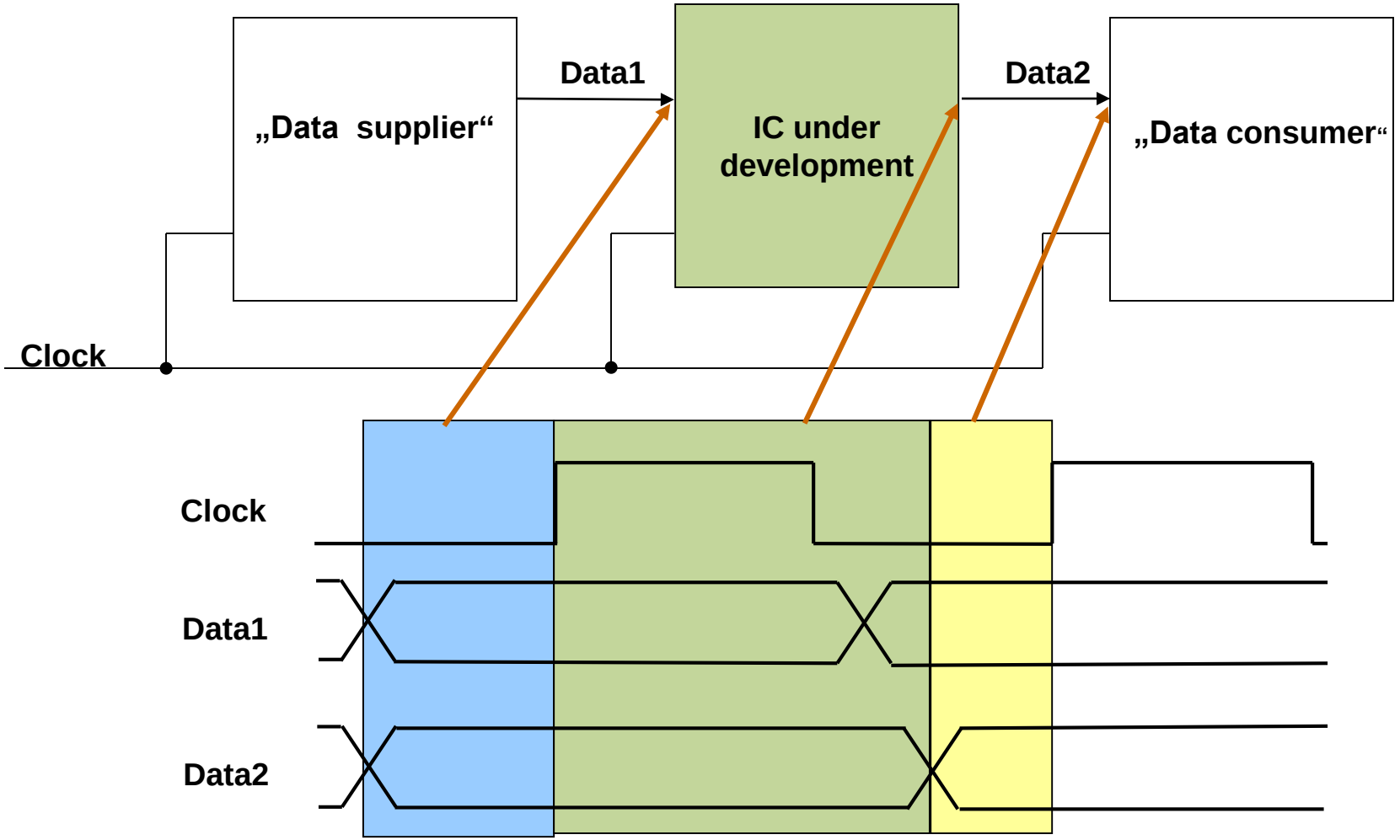
Input Synchronisation (Sampling) 2

Async: We have no knowledge when there will be events at the inputs related to the clock edges



Use a FF (without any logic) for sampled inputs.
Distribute the sampled signals.
Full clock period available for the synchronized inputs

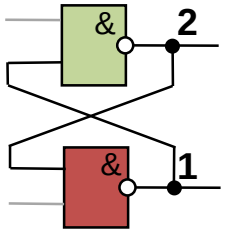
Synchronisation of different IC's

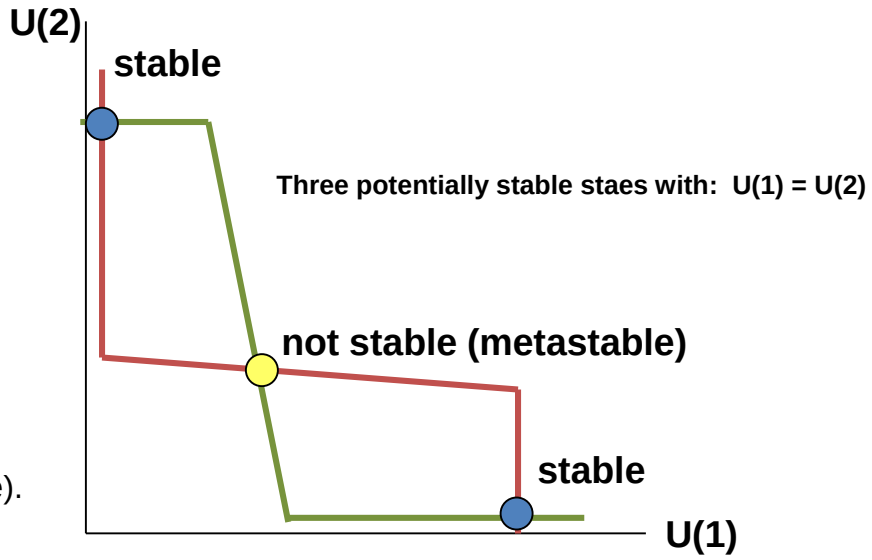


Metastability 1

A specter is haunting digital design – the specter of metastability

from: XILINX Databook





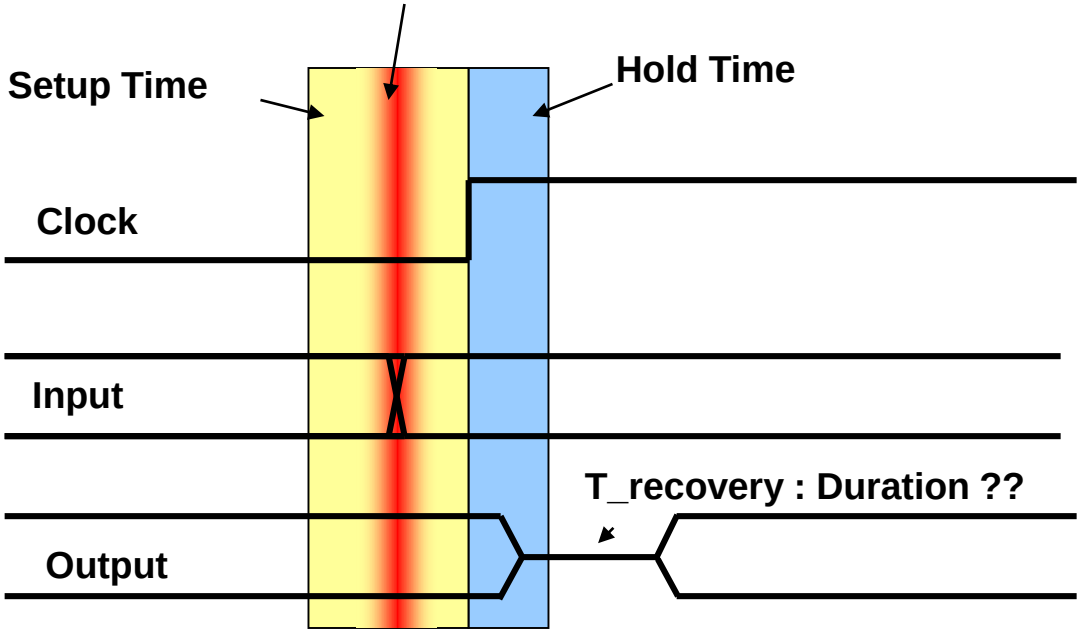
From metastable state the machine changes to one of the stable states by minimal changes of voltage (noise).

Time for the change to stable state Is dependent of the circuit properties (loop gain).

Metastability 2

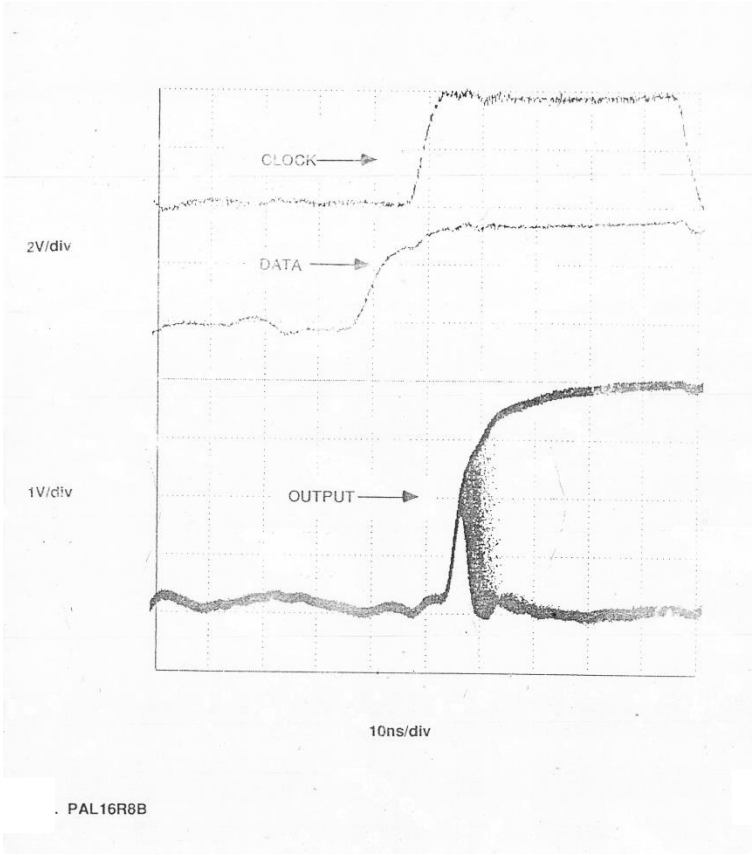
In circuits metastable states can be reached when the setup times are violated (Change of clock and data level at the same time).
There is a time window dependend on the circuit properties (internal delays, gain).

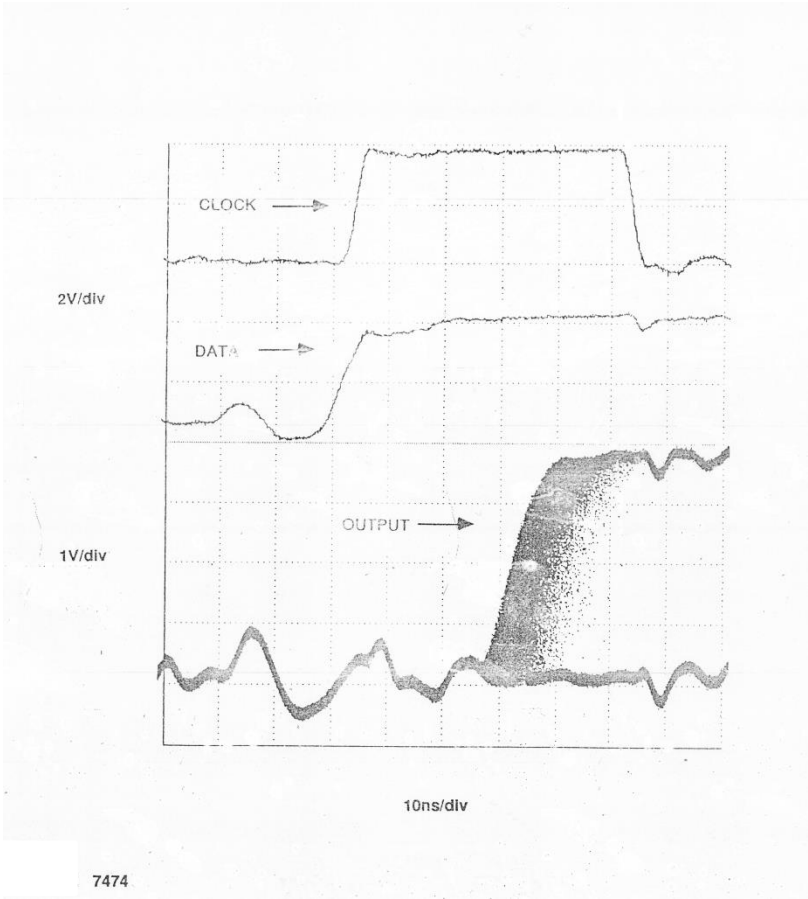
Time window for Metastability (Capture window, T_w)



Impacts: „Illegal“ Value for a certain time
Delay of output signal
Short pulses of inverted value

Metastability 3





Plots of Metastability Effects at PAL 16R8 and 7474

Metastability 4

Mean time between two occurences of metastable events of a certain duration is a function of:

Frequency of the signal to be sampled
Frequency of the sample clock
Circuit properties (Capture window, gain)

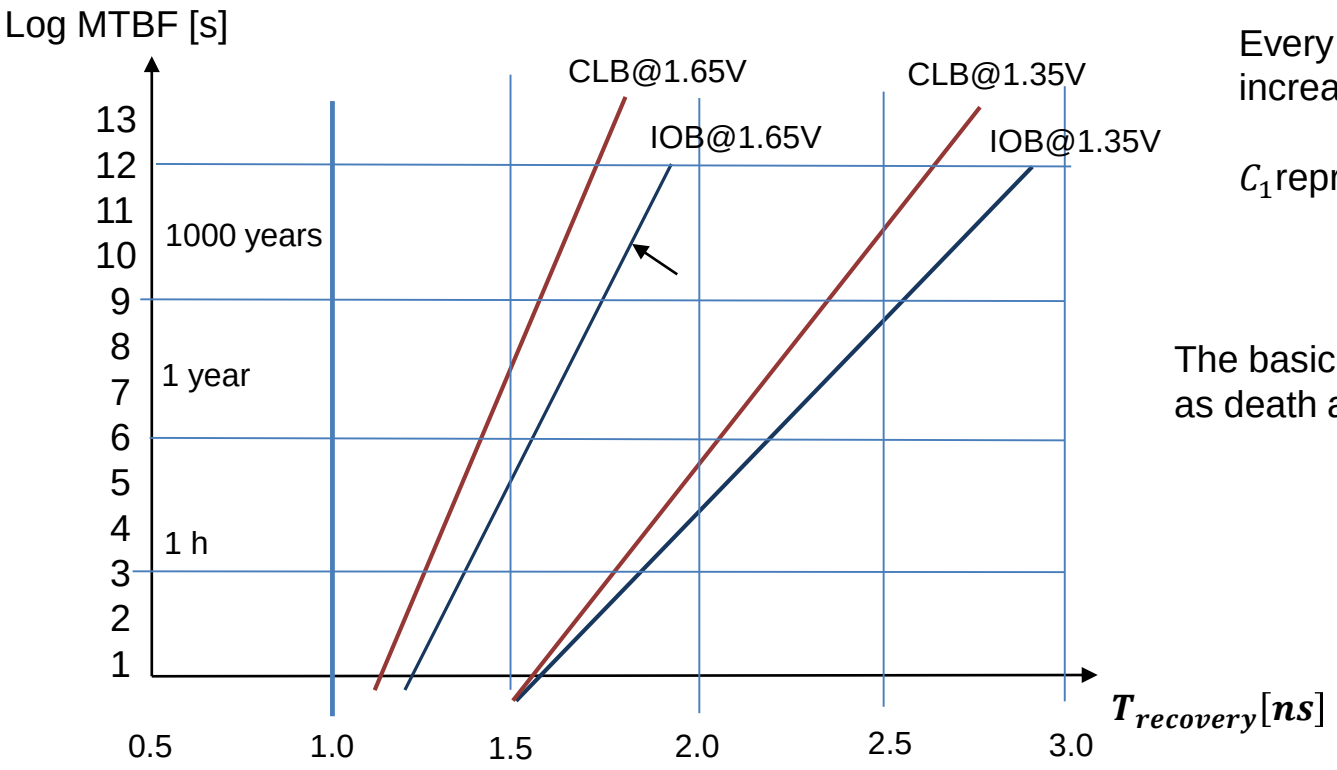
$$MTBF(T_{rec}) = \frac{1}{f_{clk} \cdot f_{dat} \cdot C_1} \cdot e^{C_2 \cdot t_{recovery}}$$

XILINX Virtex II : f_clk=300 MHz, f_dat = 50 MHz

Xilinx Virtex II (1.5V): C2 = 25

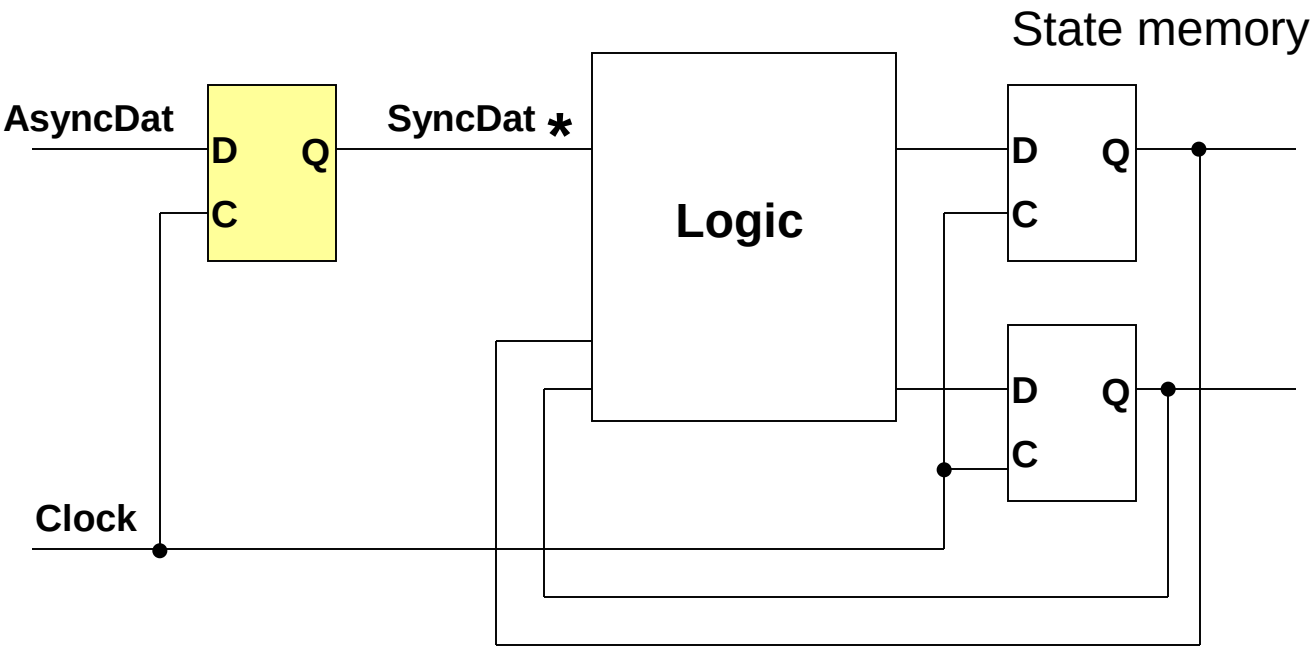
Every 0.1 ns additional recovery time increases MTBF by 12 times

C_1 represents the capture window width



The basic phenomenon is unavoidable as death and taxes

Metastability 5



With metastability the SyncDat is not save, and neither is the function state machine