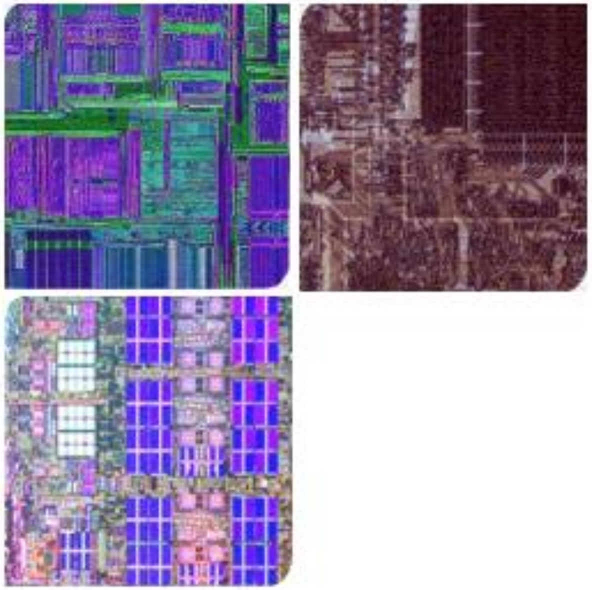
PE713 FPGA Based System Design



Why VLSI?

- Integration improves the design:
  - higher speed;
  - lower power;
  - physically smaller.
- Integration reduces manufacturing cost (almost) - no manual assembly.

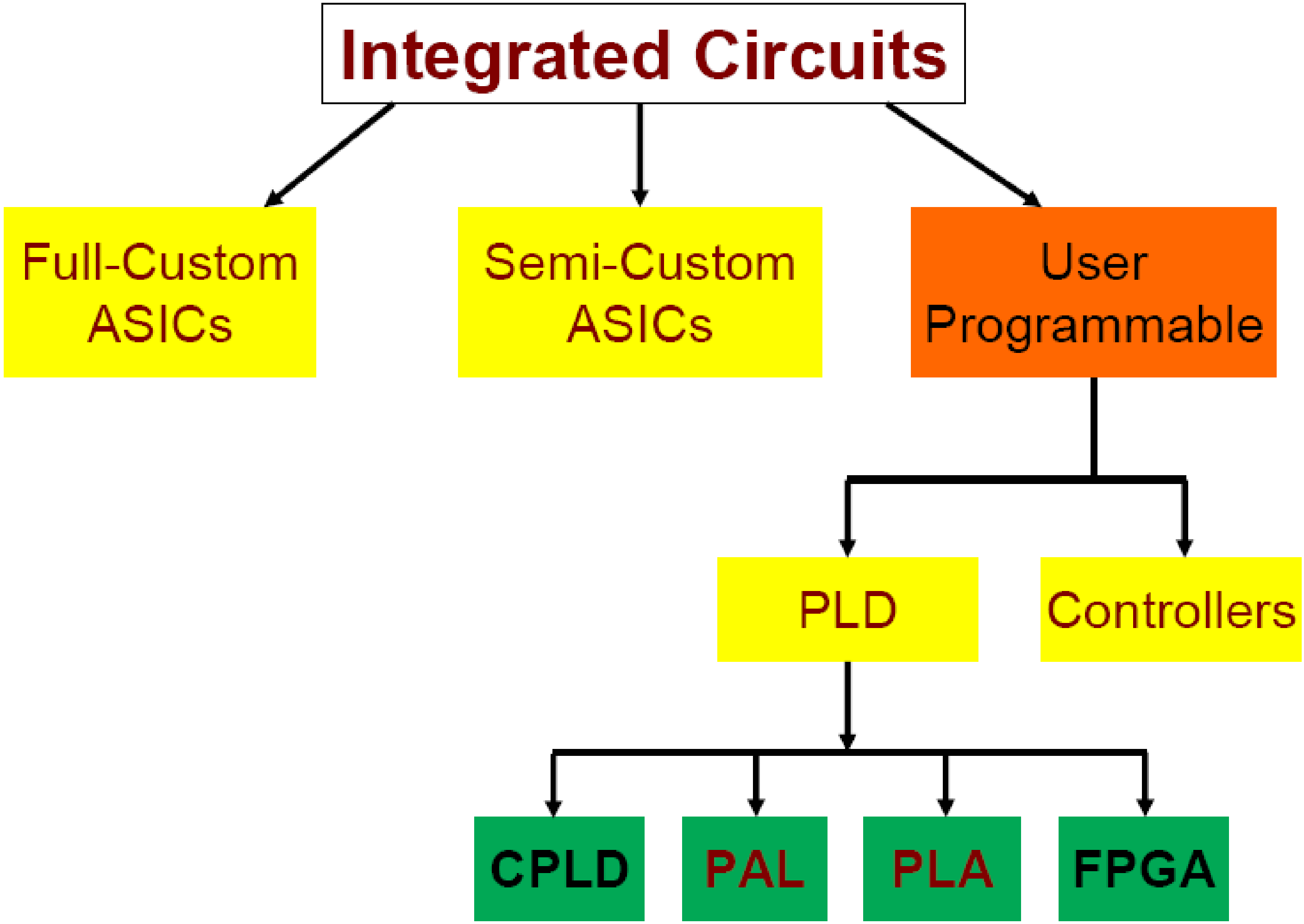
Why ICs?

- Miniaturization
- Increased equipment density
- Cost reduction due to batch processing
- Increased system reliability
- Improved functional performance
- Increased operating speeds
- Reduction in power consumption

IC Classification

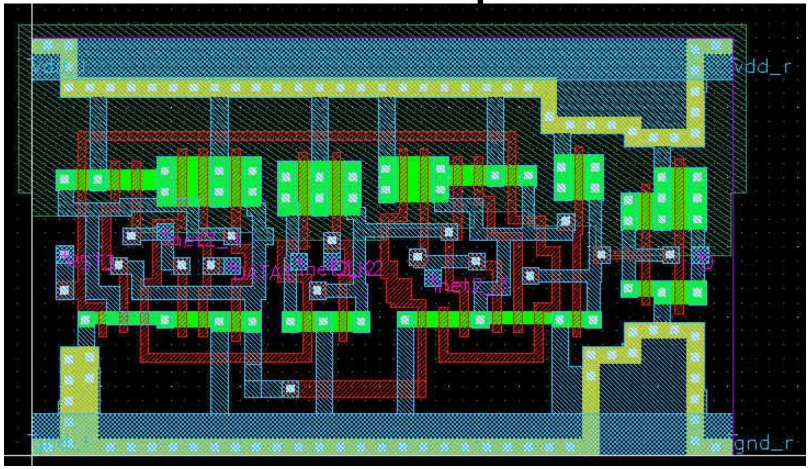
- ANALOG (OR LINEAR) ICs
  - produce, amplify, or respond to variable voltages
  - include many kinds of amplifiers, timers, oscillators, and voltage regulators.
- DIGITAL (OR LOGIC) ICs
  - respond to or produce signals having only two voltage levels, high and low
  - Digital ICs include microprocessors, memories, microcomputers

IC's


Integrated Circuits
Full-Custom ASICs
Semi-Custom ASICs
User Programmable
PLD
Controllers
CPLD
PAL
PLA
FPGA

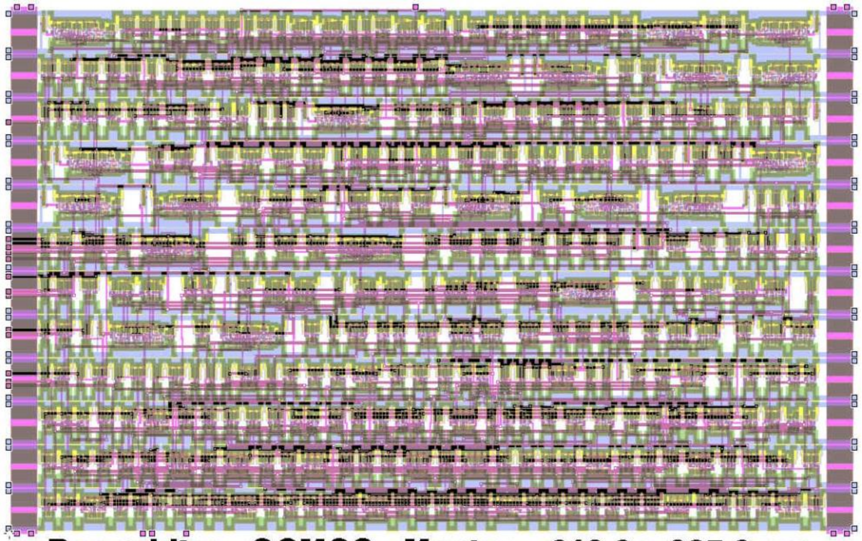
Full Custom ICs

- Can achieve very high transistor density (transistors per square micron)
- design time can be very long (multiple months).
- Involves the creation of a completely new chip, which consists of masks (for the photolithographic manufacturing process)
- Benefits - Excellent performance, small size, low power



Standard Cell

- Designer uses a library of standard cells
- an automatic place and route tool does the layout
- *Transistor density* and performance degradation depends on type of design being done.
- Design time can be much faster than full custom because layout is automatically generated.



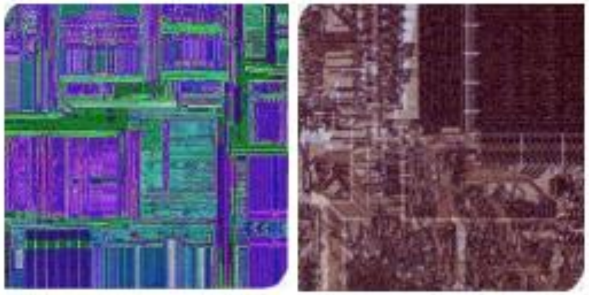
Bus arbiter - GCMOS - Mentor 619.2 x 385.2 um

Gate Array

- Designer uses a library of standard cells.
- The design is mapped onto an array of transistors which is already created on a wafer
- wafers with transistor arrays can be created ahead of time
- A *routing tool* creates the masks for the routing layers and "customizes" the pre-created gate array for the user's design
- Transistor density can be almost as good as standard cell.
- Design time advantages are the same as for standard cell.

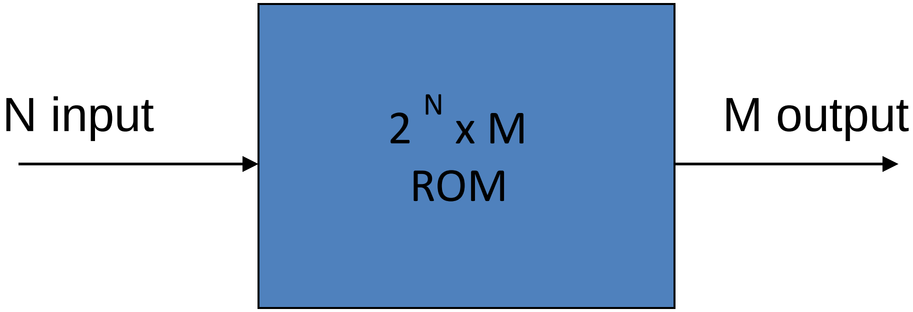
Semi-custom ICs

- Flexible as portion of the IC is customized by the user
- Suitable for specific applications
- Gate array + standard cell
- Paves way for application specific ICs (ASIC)



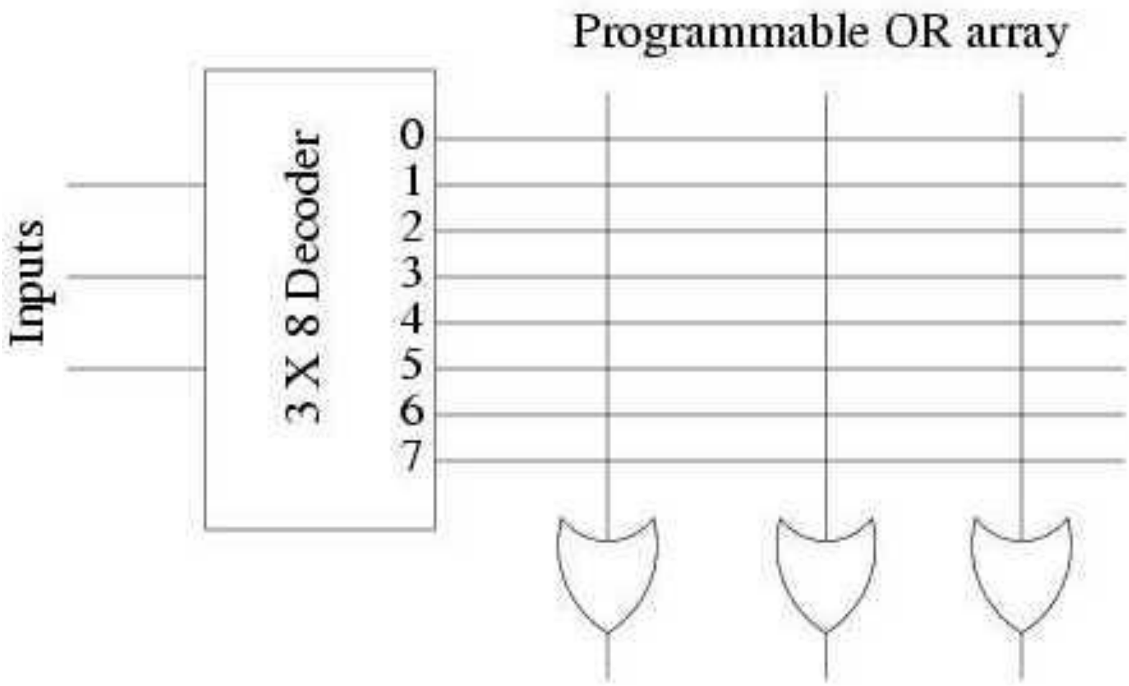
Programmable Logic Devices

Programmable ROM (PROM)



- Address: N bits; Output word: M bits
- ROM contains 2^N words of M bits each
- The input bits decide the particular word that becomes available on output lines

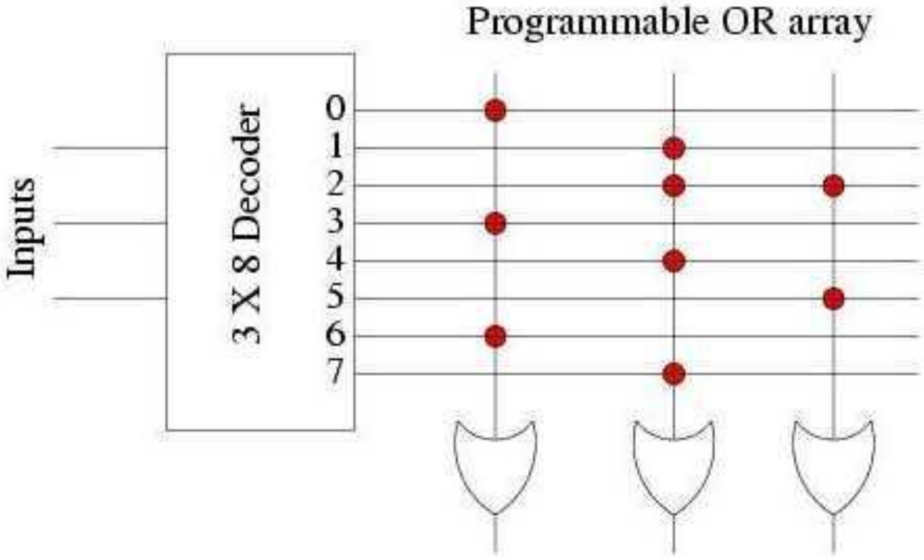
Logic Diagram of 8x3 PROM



Sum of minterms

Combinational Circuit Implementation using PROM

I0	I1	I2	F0	F1	F2
0	0	0	1	0	0
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	0	1	0
1	0	1	0	0	1
1	1	0	1	0	0
1	1	1	0	1	0



F0 F1 F2

PROM Types

- Programmable PROM
  - –Break links through current pulses
  - –Write once, Read multiple times
- Erasable PROM (EPROM)
  - –Program with ultraviolet light
  - –Write multiple times, Read multiple times
- Electrically Erasable PROM (EEPROM)/ Flash Memory
  - –Program with electrical signal
  - –Write multiple times, Read multiple times

PROM: Advantages and Disadvantages

- Widely used to implement functions with large number of inputs and outputs

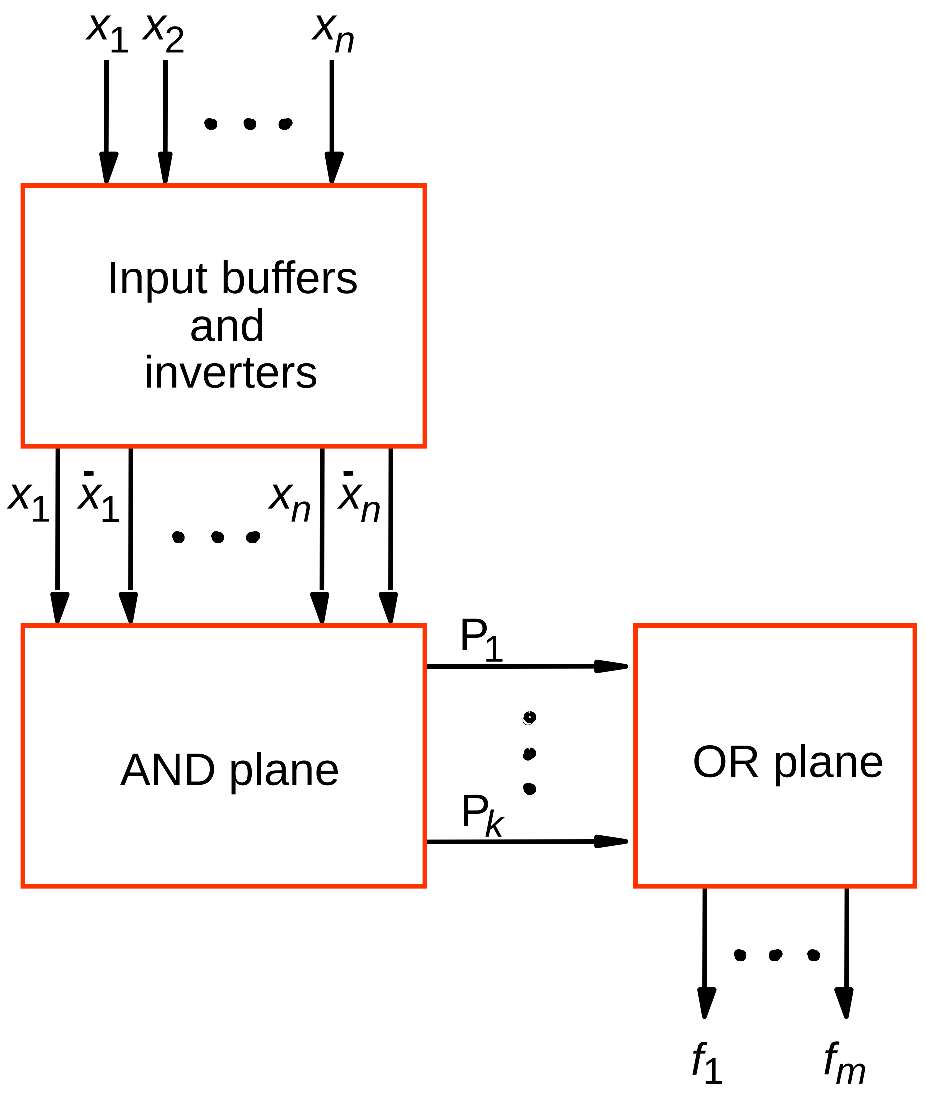
- For combinational circuits with lots of don’t care terms, PROM is a wastage of logic resources

Programmable Logic Array (PLA)

- Use to implement circuits in SOP form

- The connections in the AND plane are programmable

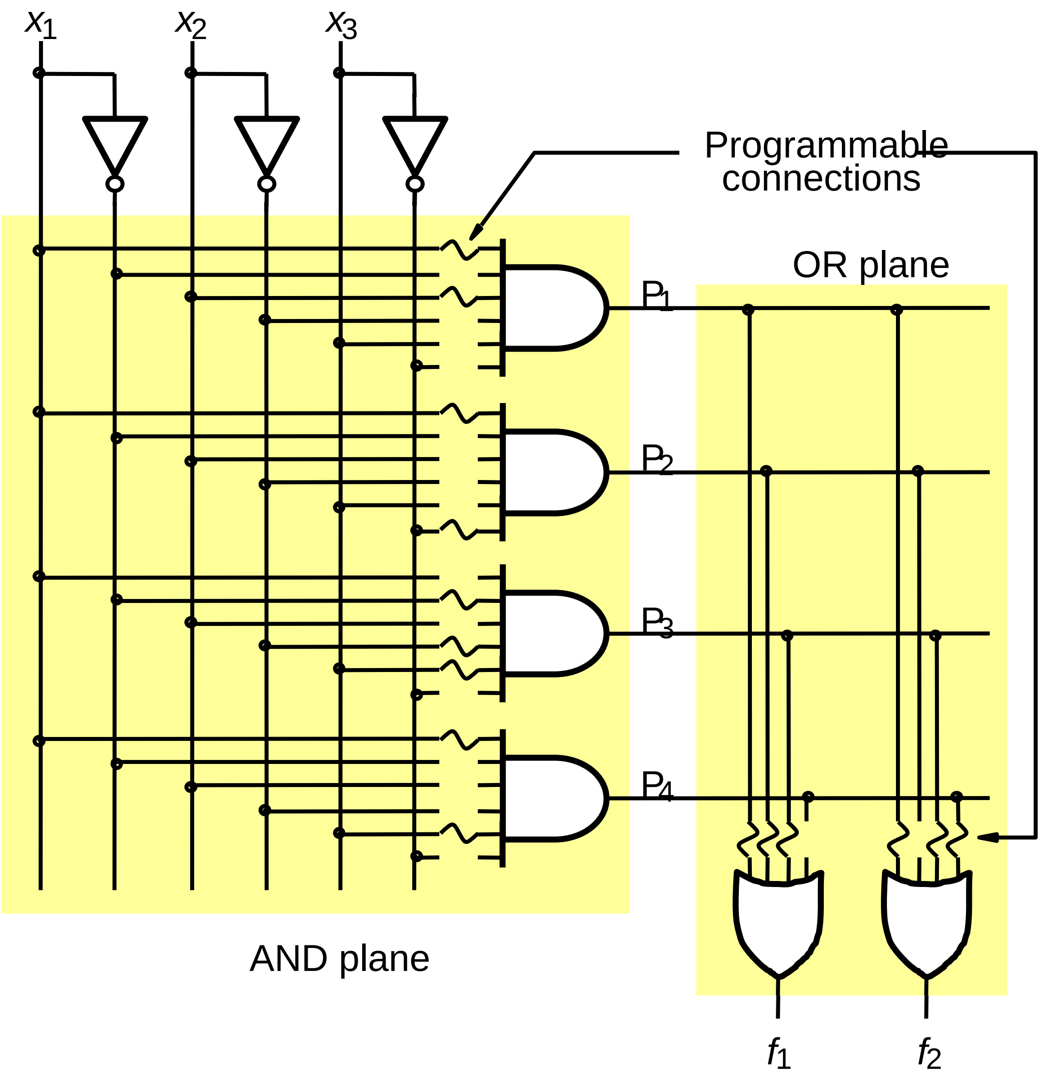
- The connections in the OR plane are programmable


x_1 x_2 x_n
Input buffers and inverters
x_1 $\bar{x}_1$ x_n $\bar{x}_n$
AND plane
P_1
P_k
OR plane
f_1 f_m

Gate Level Version of PLA

$f_1 = x_1x_2+x_1x_3'+x_1'x_2'x_3$

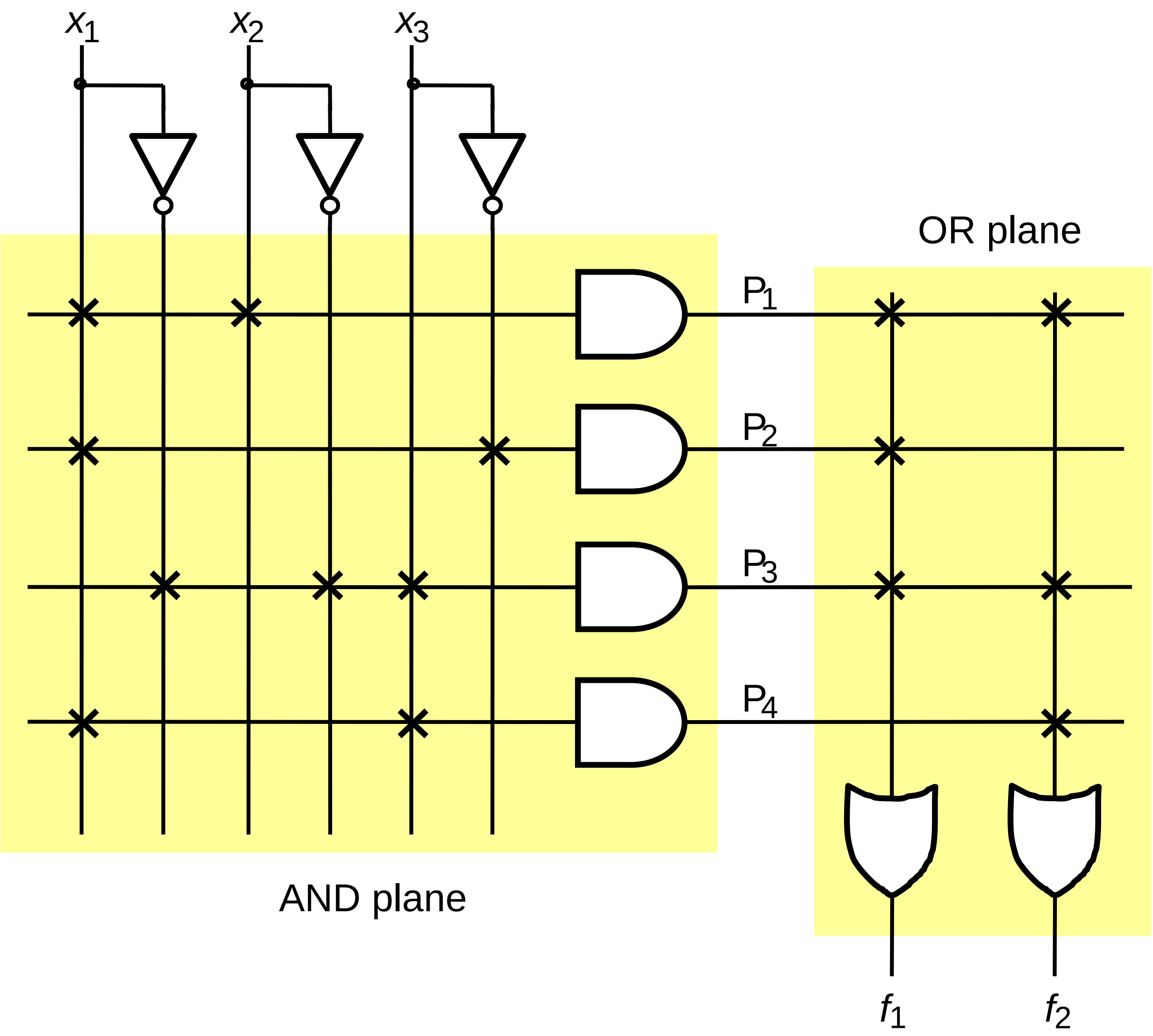
$f_2 = x_1x_2+x_1'x_2'x_3+x_1x_3$



Customary Schematic of a PLA

$f_1 = x_1x_2+x_1x_3'+x_1'x_2'x_3$

$f_2 = x_1x_2+x_1'x_2'x_3+x_1x_3$



x marks the connections left in place after programming

Limitations of PLAs

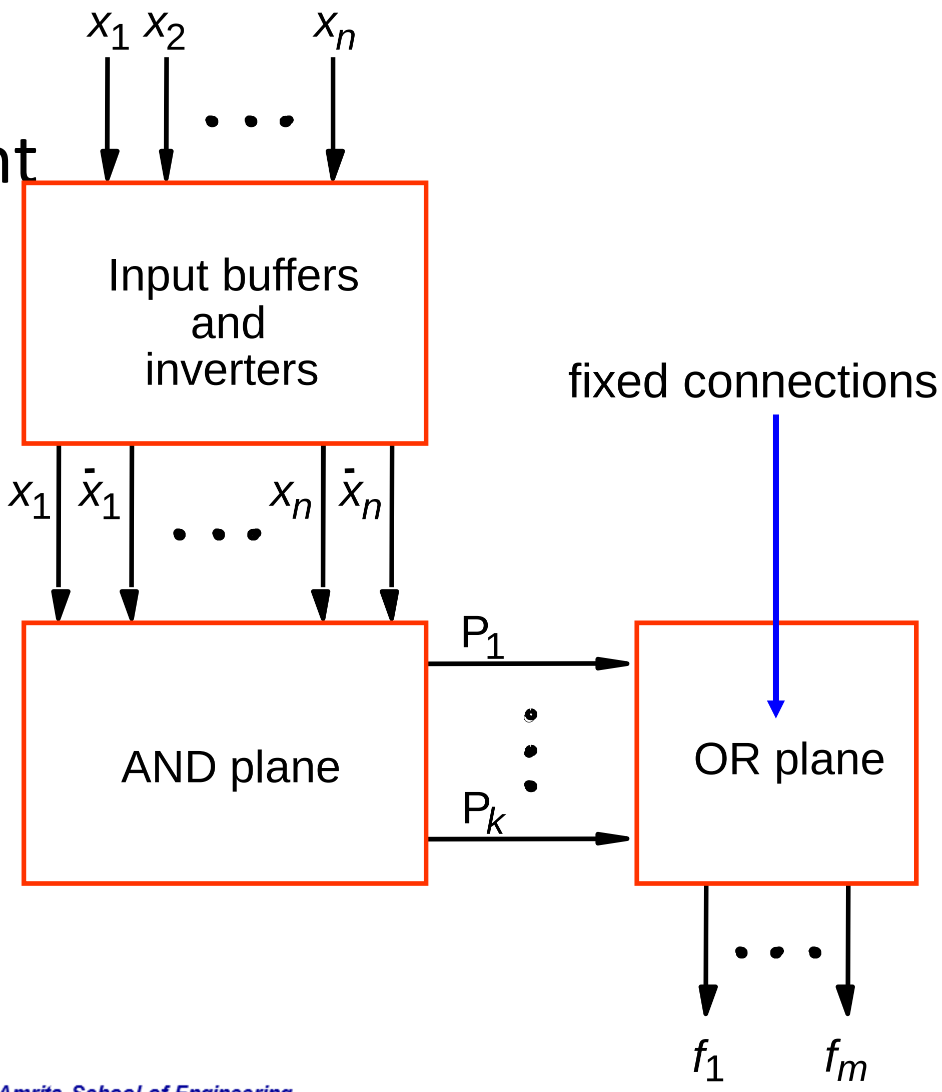
- Typical size is 16 inputs, 32 product terms, 8 outputs
  - –Each AND gate has large fan-in - this limits the number of inputs that can be provided in a PLA
  - –16 inputs → 2^{16} = possible input combinations; only 32 permitted (since 32 AND gates) in a typical PLA
  - –32 AND terms permitted → large fan-in for OR gates as well
- This makes PLAs slower and slightly more expensive than some alternatives to be discussed shortly

Programmable Array Logic (PAL)

- Also used to implement circuits in SOP form

- The connections in the AND plane are programmable

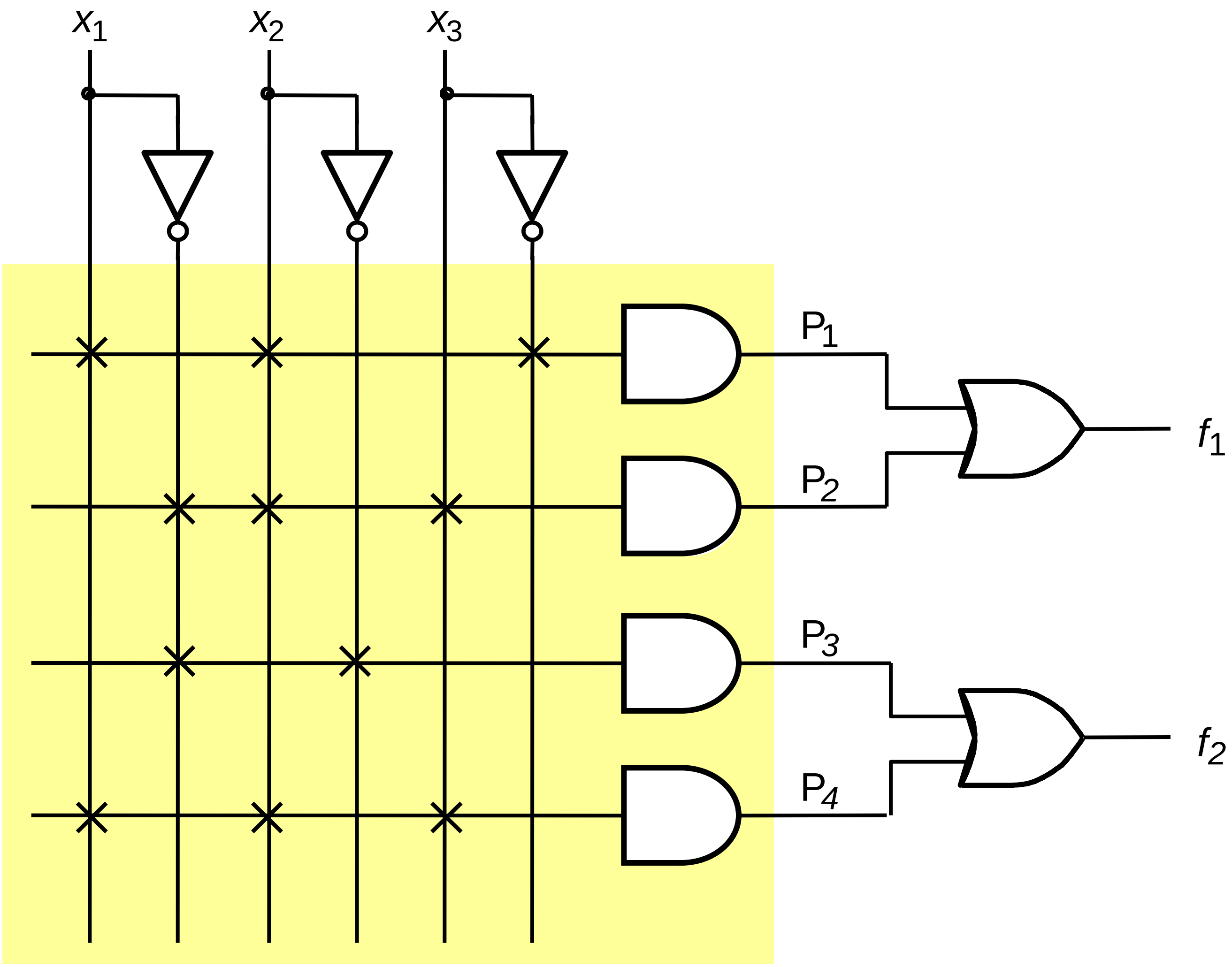
- The connections in the OR plane are NOT programmable



Example Schematic of a PAL

$f_1 = x_1x_2x_3'+x_1'x_2x_3$

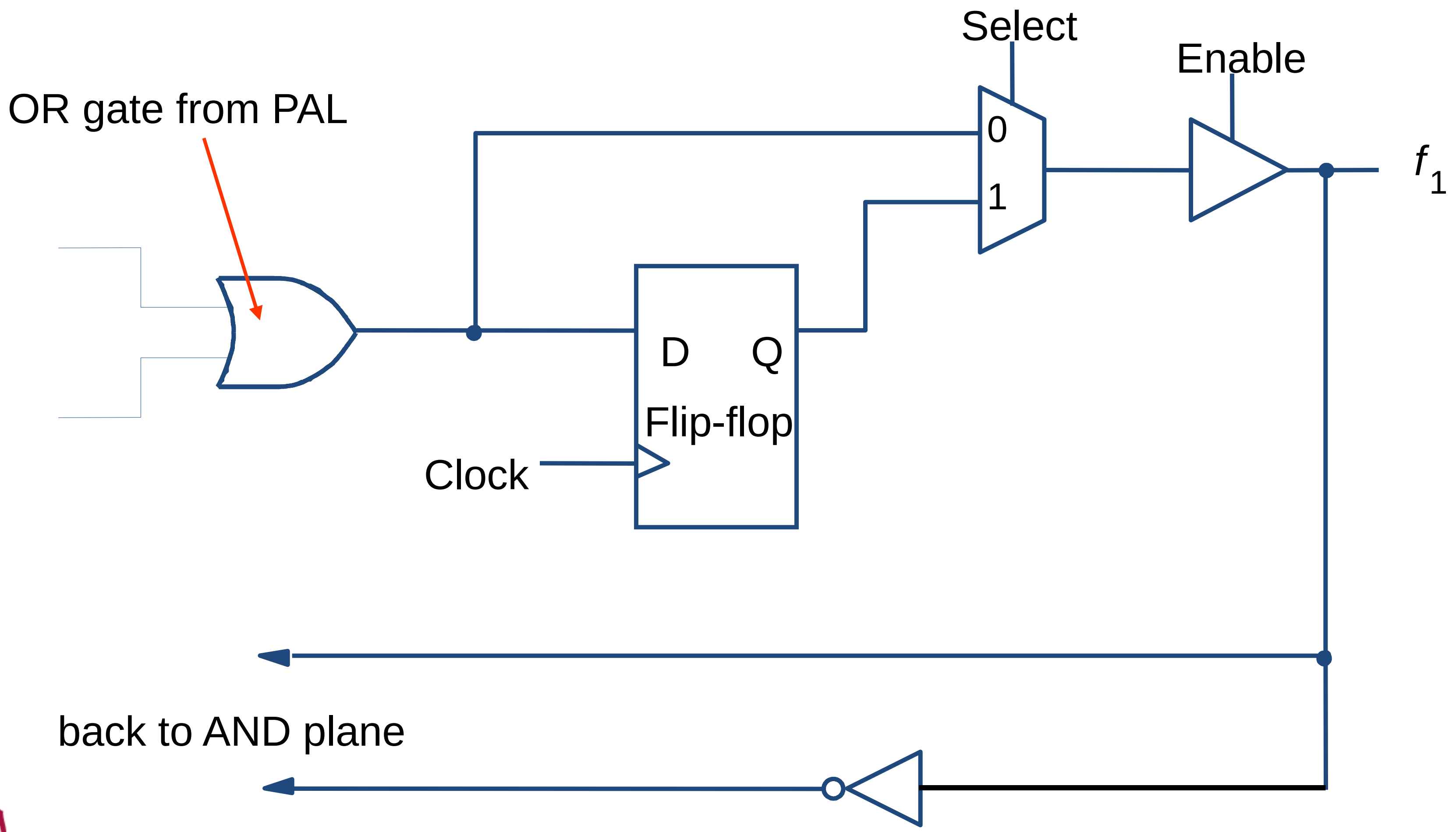
$f_2 = x_1'x_2'+x_1x_2x_3$

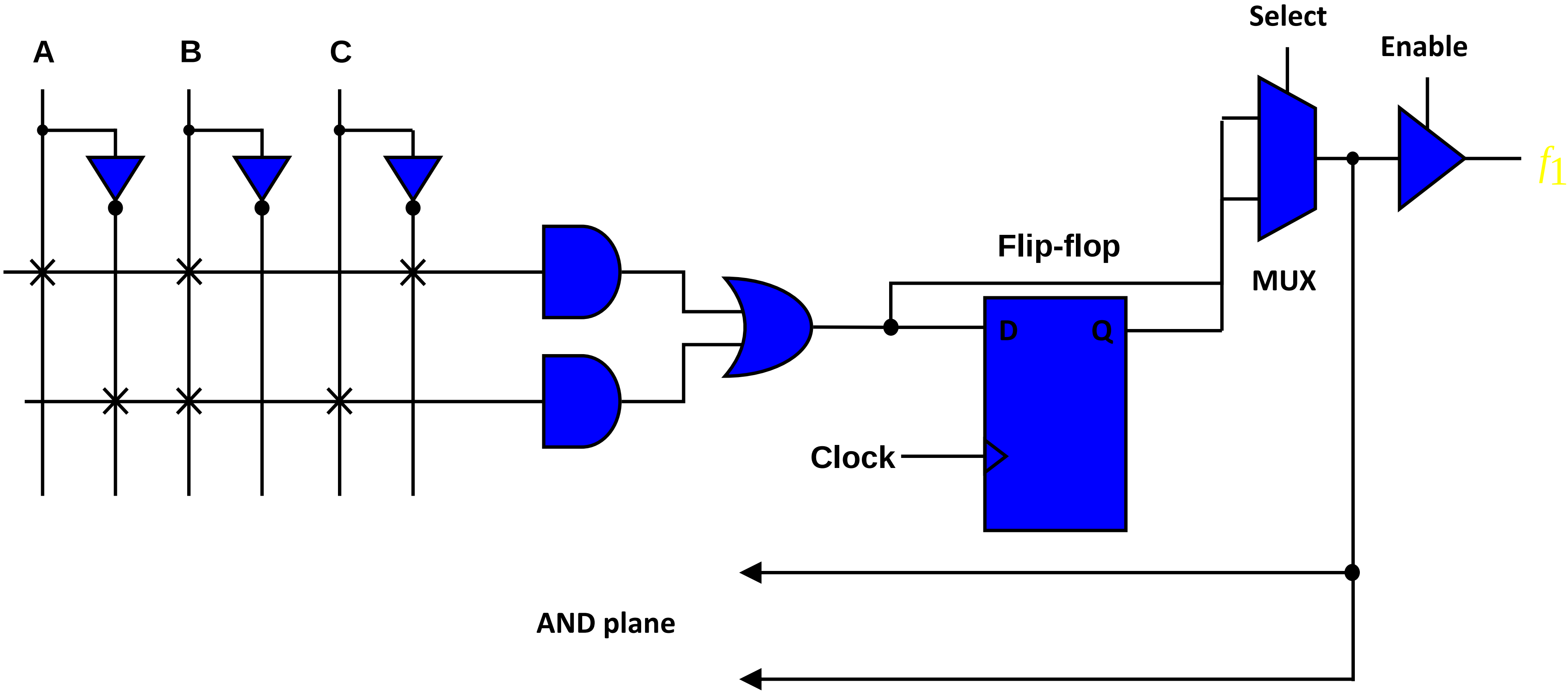


AND plane

- Comparing PALs and PLAs
  - PALs have the same limitations as PLAs (small number of allowed AND terms) plus they have a fixed OR plane → less flexibility than PLAs
  - PALs are simpler to manufacture, cheaper, and faster (better performance)
  - PALs also often have extra circuitry connected to the output of each OR gate
    - The OR gate plus this circuitry is called a *macrocell*

- Macrocell


Select
Enable
OR gate from PAL
0
1
f_1
D
Q
Flip-flop
Clock
back to AND plane

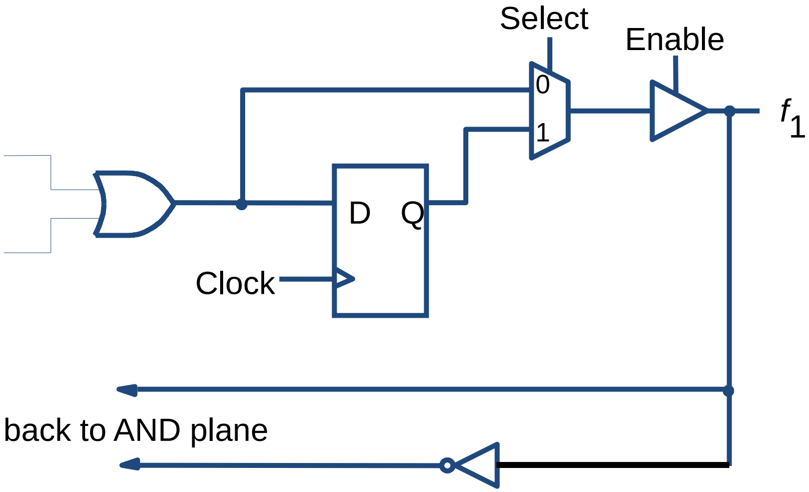

A
B
C
Flip-flop
Select
Enable
MUX
D
Q
Clock
f_1
AND plane

- Macrocell Functions

  - Enable = 0 can be used to allow the output pin for f_1 to be used as an additional <u>input</u> pin to the PAL

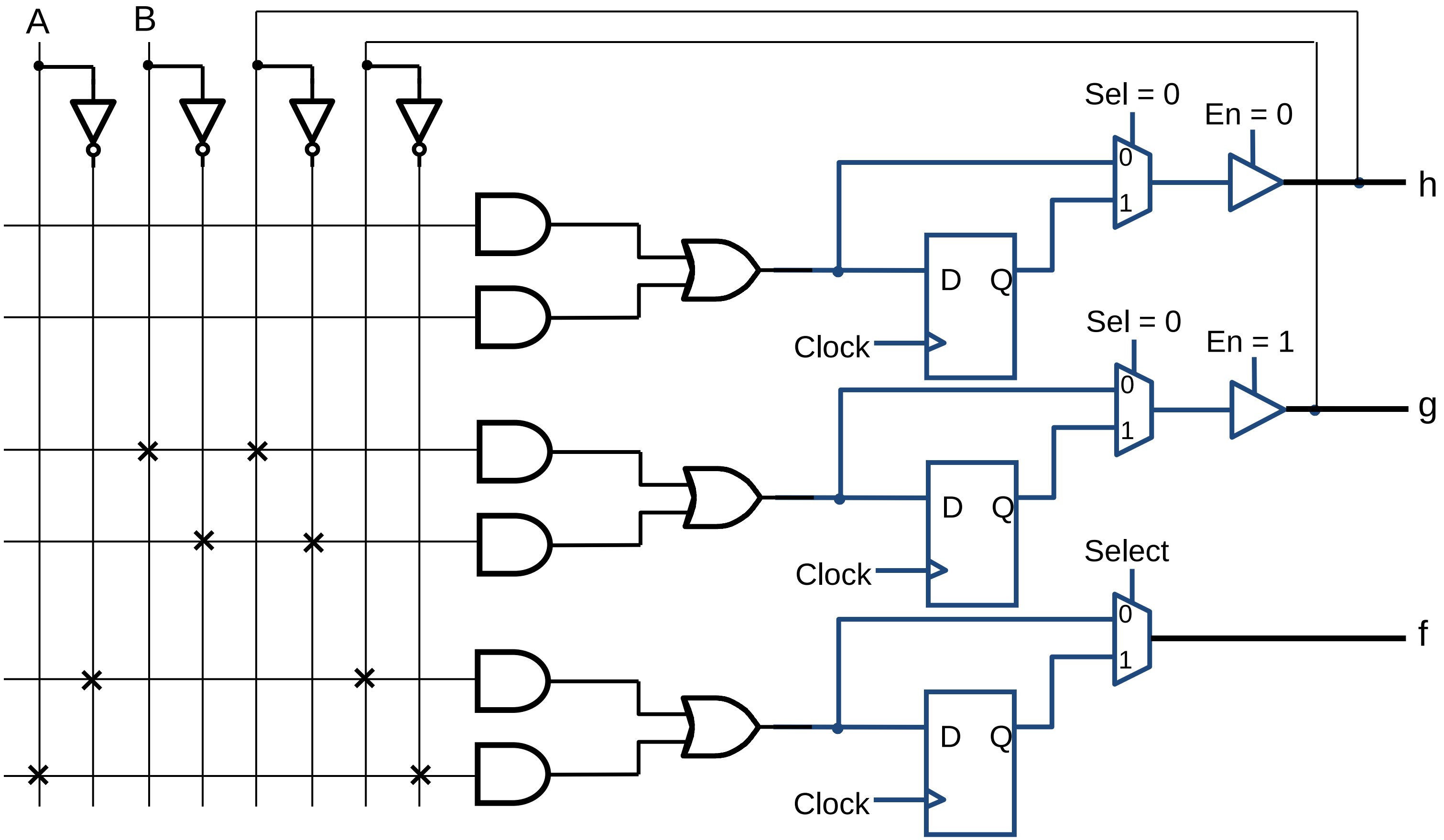
  - Enable = 1, Select = 0 is normal for typical PAL operation

  - Enable = Select = 1 allows the PAL to synchronize the output changes with a clock pulse

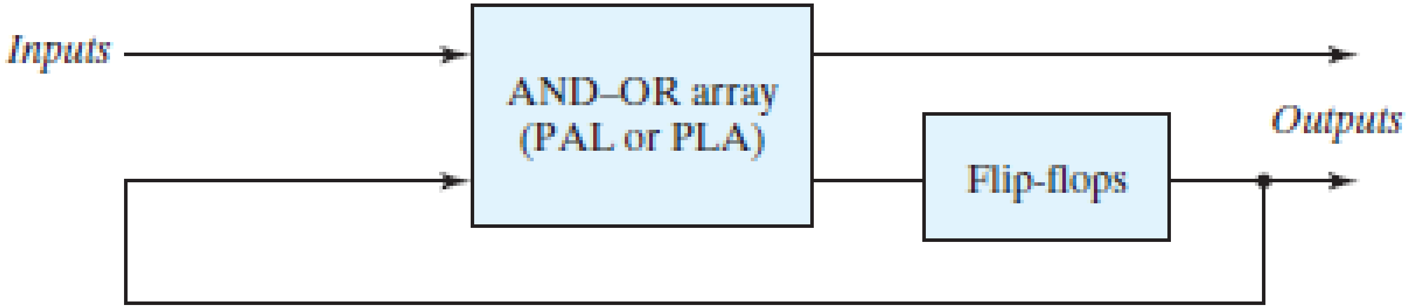


  - The feedback to the AND plane provides for multi-level design

- Multi-Level Design with PALs
  - f = A'BC + A'B'C' + ABC' + AB'C = A'g + Ag'
    - where g = BC + B'C' and C = h below

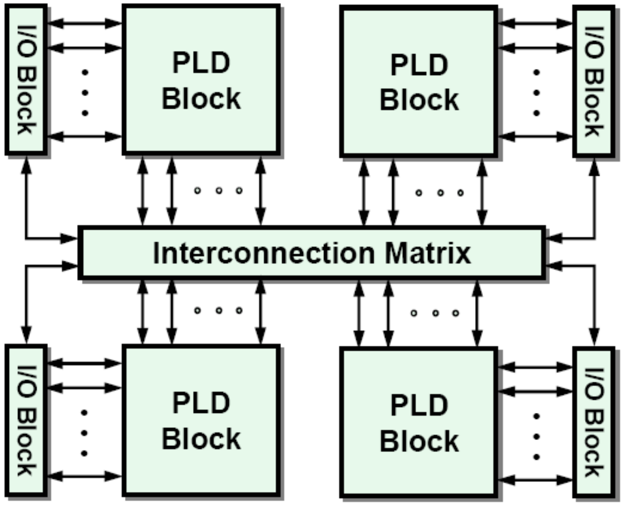


SPLD

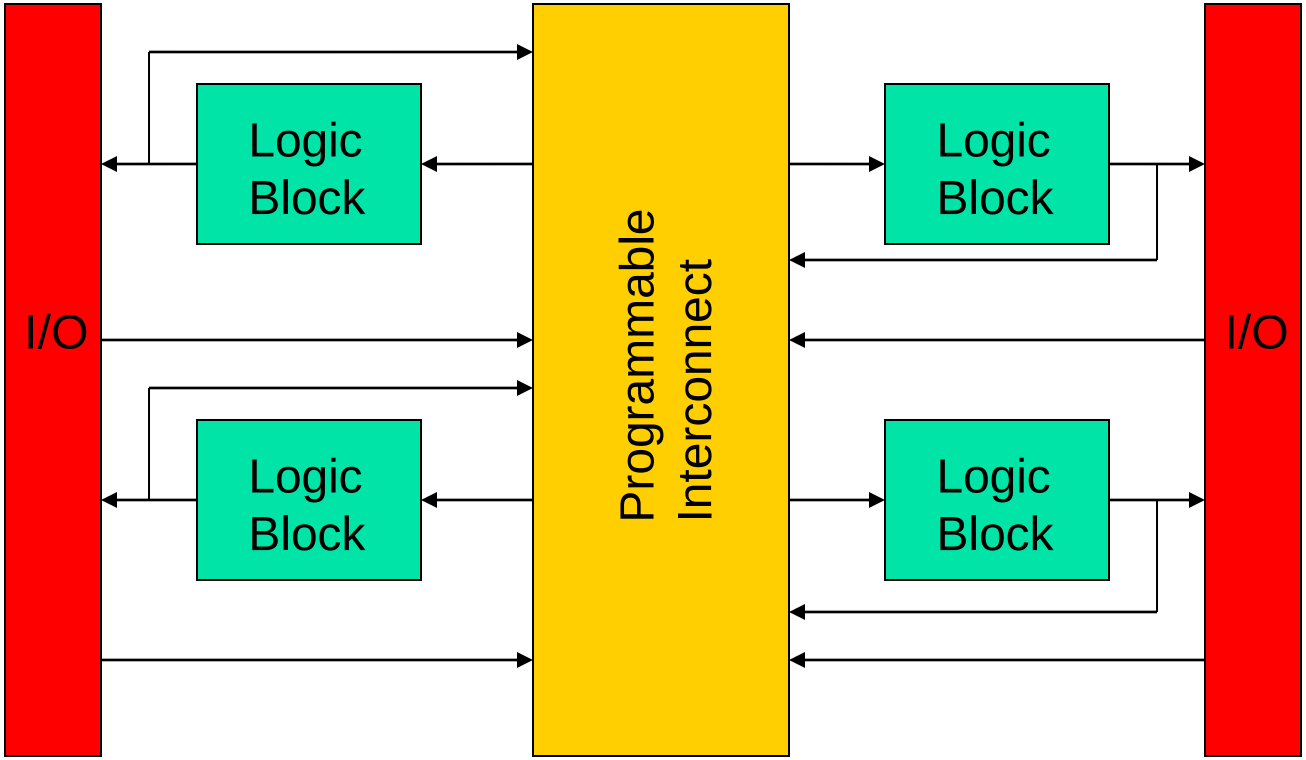


CPLD

- Interconnection of several PLD blocks with Programmable interconnect on a single chip
- Logic blocks executes sum-of-product expressions and stores the results in micro-cell registers
- Programmable interconnects route signals to and from logic blocks

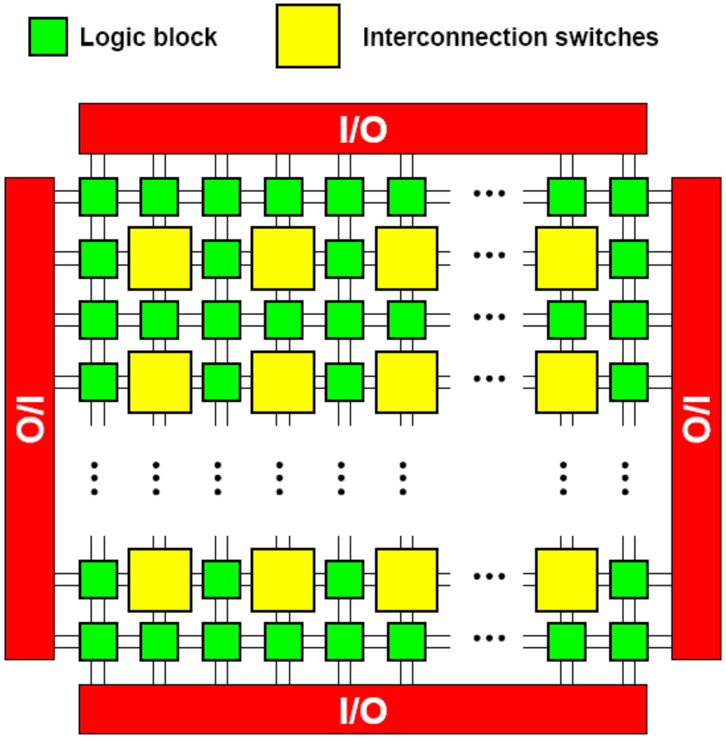


CPLD

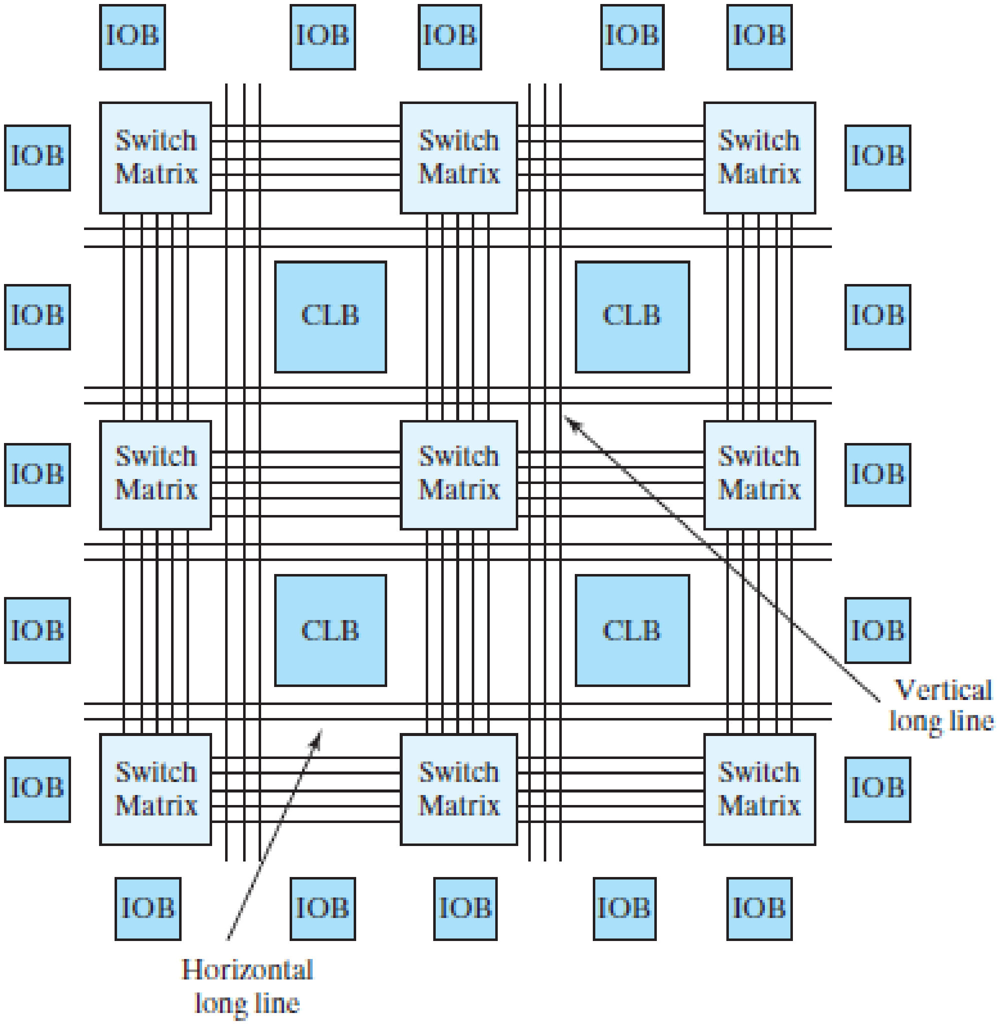

I/O
Logic Block
Logic Block
Programmable Interconnect
Logic Block
Logic Block
I/O

FPGA

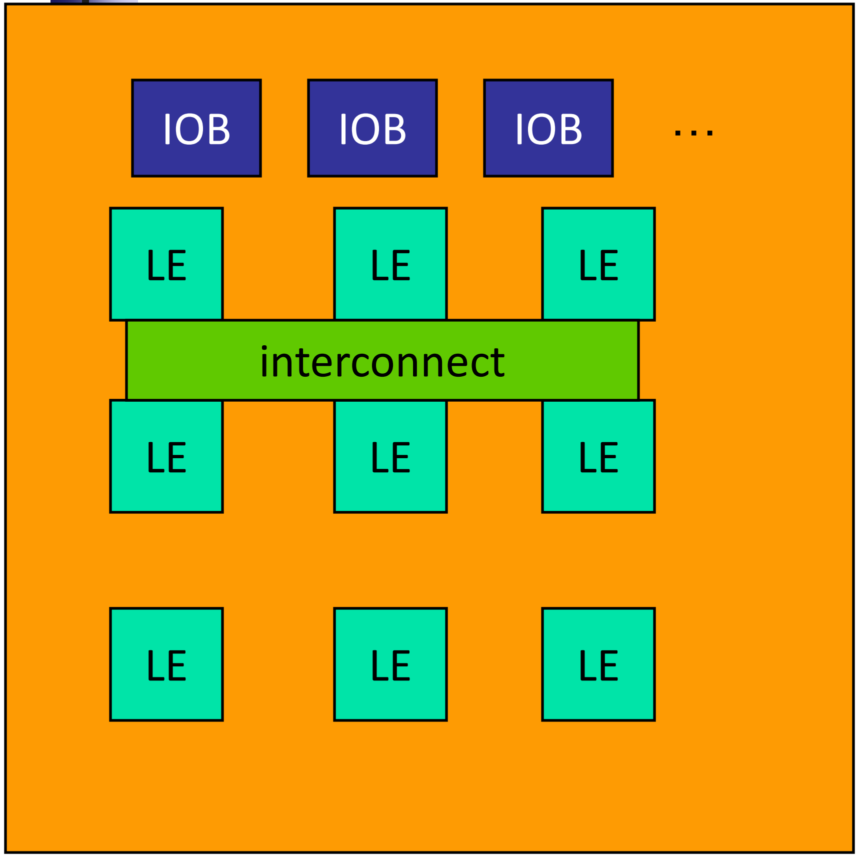
- Programmable Logic Blocks
  - Implement combinational & sequential logic
- Programmable Interconnect
  - Wires to connect inputs and outputs to logic blocks
- Programmable I/O blocks
  - Logic blocks at the periphery for external connections



Structure of FPGA



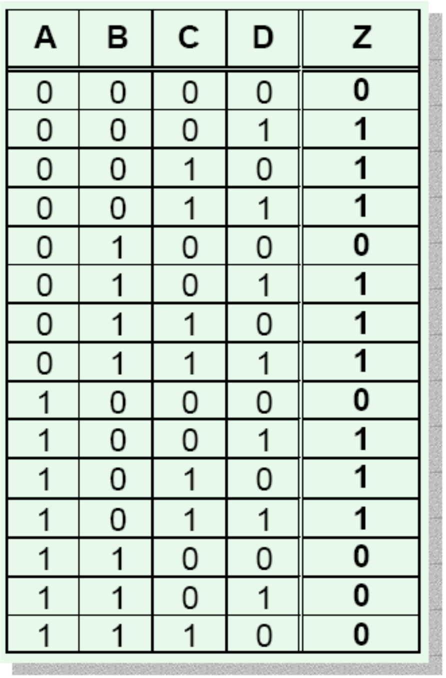
FPGA Fabric



- CLB: combinational logic block = logic element (LE).
- LUT: Lookup table = SRAM used for truth table.
- I/O block (IOB): I/O pin + associated logic and electronics.

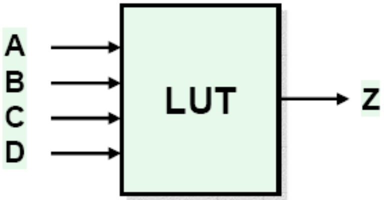
FPGA Fabric

- Look-up table with N-inputs can be used to implement any combinational function of N-inputs
- LUT is programmed with truth table

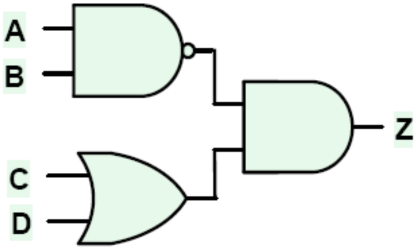


A	B	C	D	Z
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0

Truth-table

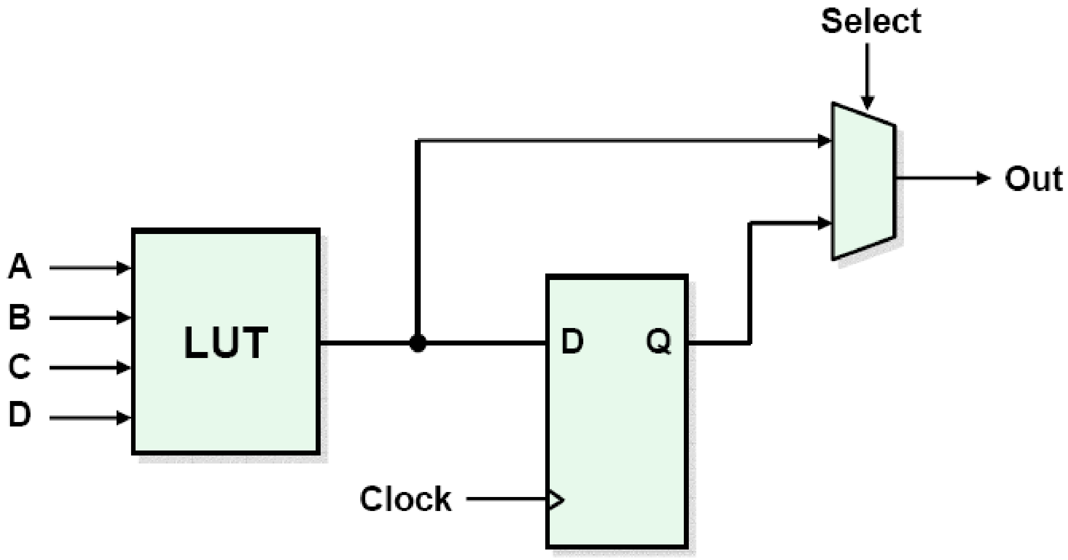


LUT implementation



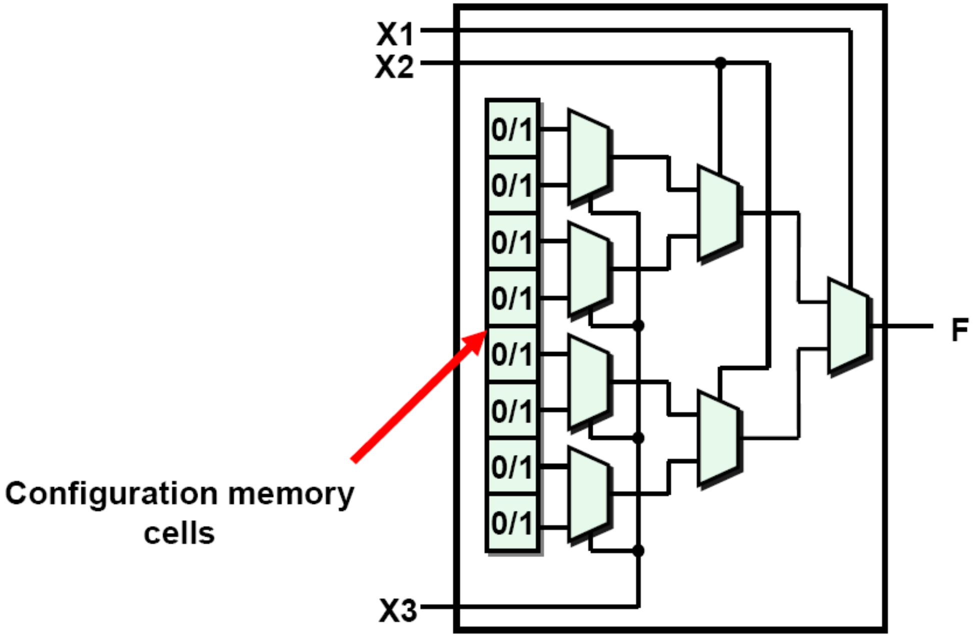
Gate implementation

FPGA Fabric

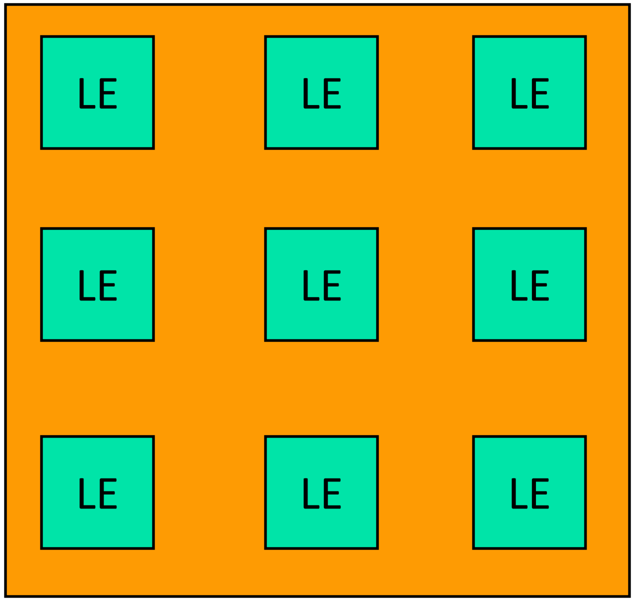


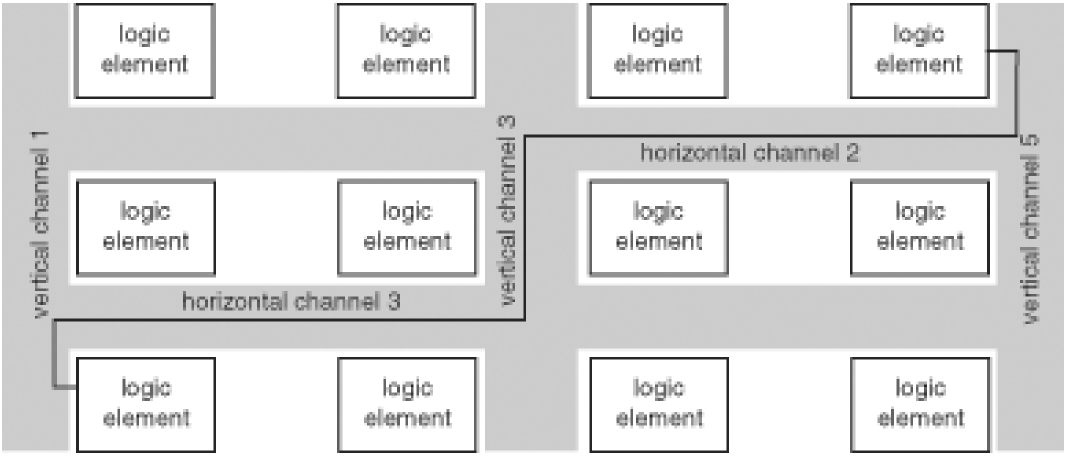
LUT

- 3-input LUT
- Based on Multiplexers
- LUT entries stored in configuration memory cells



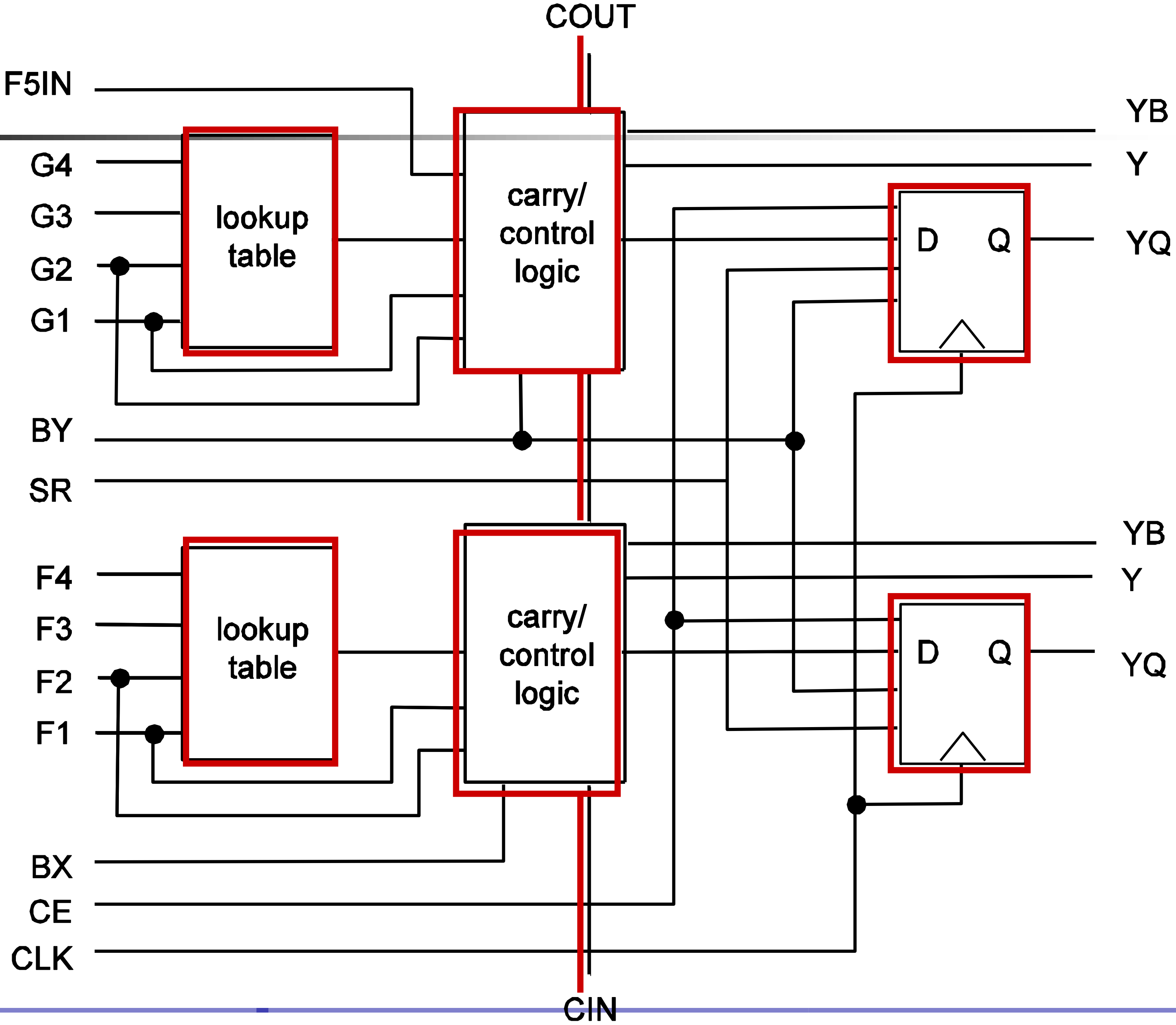
FPGA Fabric (contd)





Xilinx Spartan-II CLB

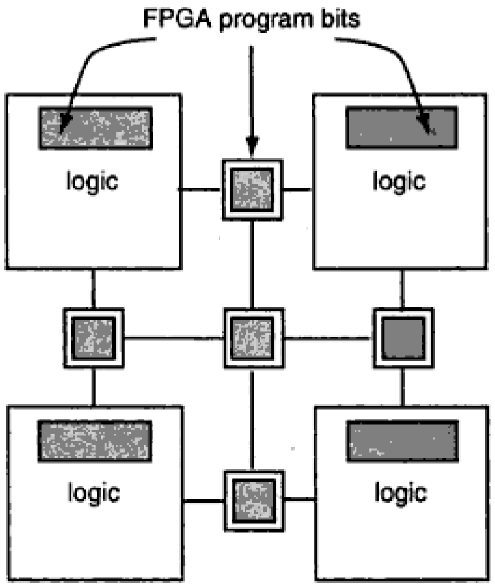
- Each CLB has two identical slices.
- Slice has two logic cells:
  - LUT.
  - Carry/control logic.
  - Registers.


COUT
F5IN
G4
G3
G2
G1
lookup table
carry/ control logic
YB
Y
D
Q
YQ
BY
SR
F4
F3
F2
F1
lookup table
carry/ control logic
YB
Y
D
Q
YQ
BX
CE
CLK
CIN

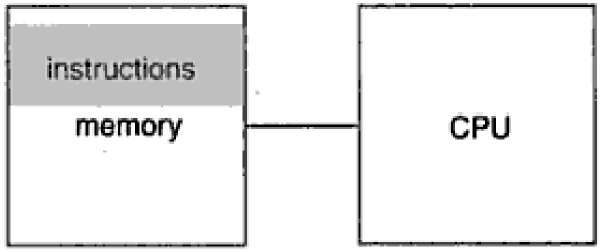
FPGA Programming

- FPGAs implement multi-level logic
- Need both programmable logic blocks and programmable interconnect
- Combination of logic and interconnect is fabric

- Microprocessor is a stored-program computer



FPGA



microprocessor

Role of FPGA

- Microprocessors used in variety of environments
  - Rely on software to implement functions
  - Generally slower and more power-hungry than custom chips
- When FPGAs?
  - Design economics
    - Shortest time to market
    - Lowest NRE cost
    - Highest unit cost
  - Make quick grab for market share
  - Same FPGA reused in several designs

FPGAs and VLSI

- FPGAs are standard parts:
  - Pre-manufactured.
  - Don’t worry (much) about physical design.
- Custom silicon:
  - Tailored to your application.
  - Generally lower power consumption.

Standard parts vs. custom

- Do you build your system with an FPGA or with custom silicon?
  - FPGAs have shorter design cycle.
  - FPGAs have no manufacturing delay.
  - FPGAs reduce inventory.
  - FPGAs are slower, larger, more power-hungry.

FPGA Advantages

- Faster time-to-market.
- Simpler design cycle.
- More predictable project cycle.
- Field Reprogramability.
- Reusability.
- Good for prototyping.
- Applicable for lower speed, lower complexity and lower volume designs.

FPGA Disadvantages

- FPGA consumes more power.
- Limits Design Capability.
- Not suitable for volumes of production.

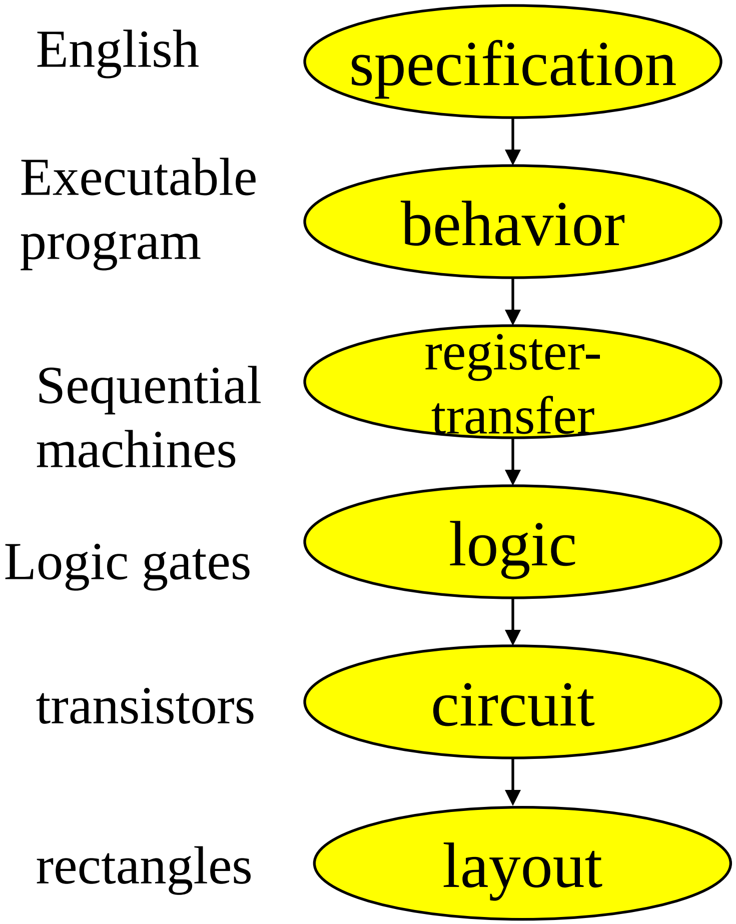
Goals and Techniques

- Performance
  - Logic rate
- Power/energy
- Design time
- Design cost
  - FPGA tools less expensive than custom VLSI tools
- Manufacturing cost

Design Challenges

- Multiple levels of abstraction
- Power consumption
- Short design time

FPGA Abstractions



English

specification

Executable program

behavior

Sequential machines

register-transfer

Logic gates

logic

transistors

circuit

rectangles

layout

Methodology

- Hardware Description logic (HDL)
    - ABEL
    - CUPL
    - PALASM
    - VHDL
    - VerilogHDL

Design process (1)

Design and implement a simple unit permitting to speed up encryption with RC5-similar cipher with fixed key set on 8031 microcontroller. Unlike in the experiment 5, this time your unit has to be able to perform an encryption algorithm by itself, executing 32 rounds…..

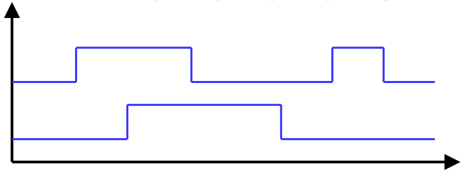
Specification (Lab Experiments)

VHDL description (Your Source Files)

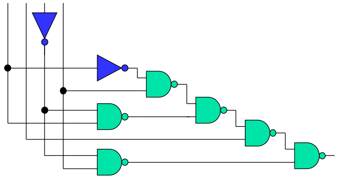
```
Library IEEE;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity RC5_core is
        port(
            clock, reset, encr_decr: in std_logic;
            data_input: in std_logic_vector(31 downto 0);
            data_output: out std_logic_vector(31 downto 0);
            out_full: in std_logic;
            key_input: in std_logic_vector(31 downto 0);
            key_read: out std_logic;
        );
end AES_core;
```

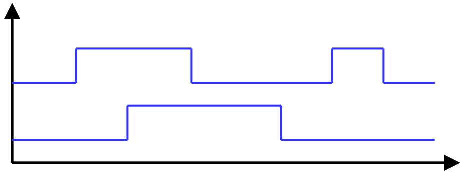
Functional simulation



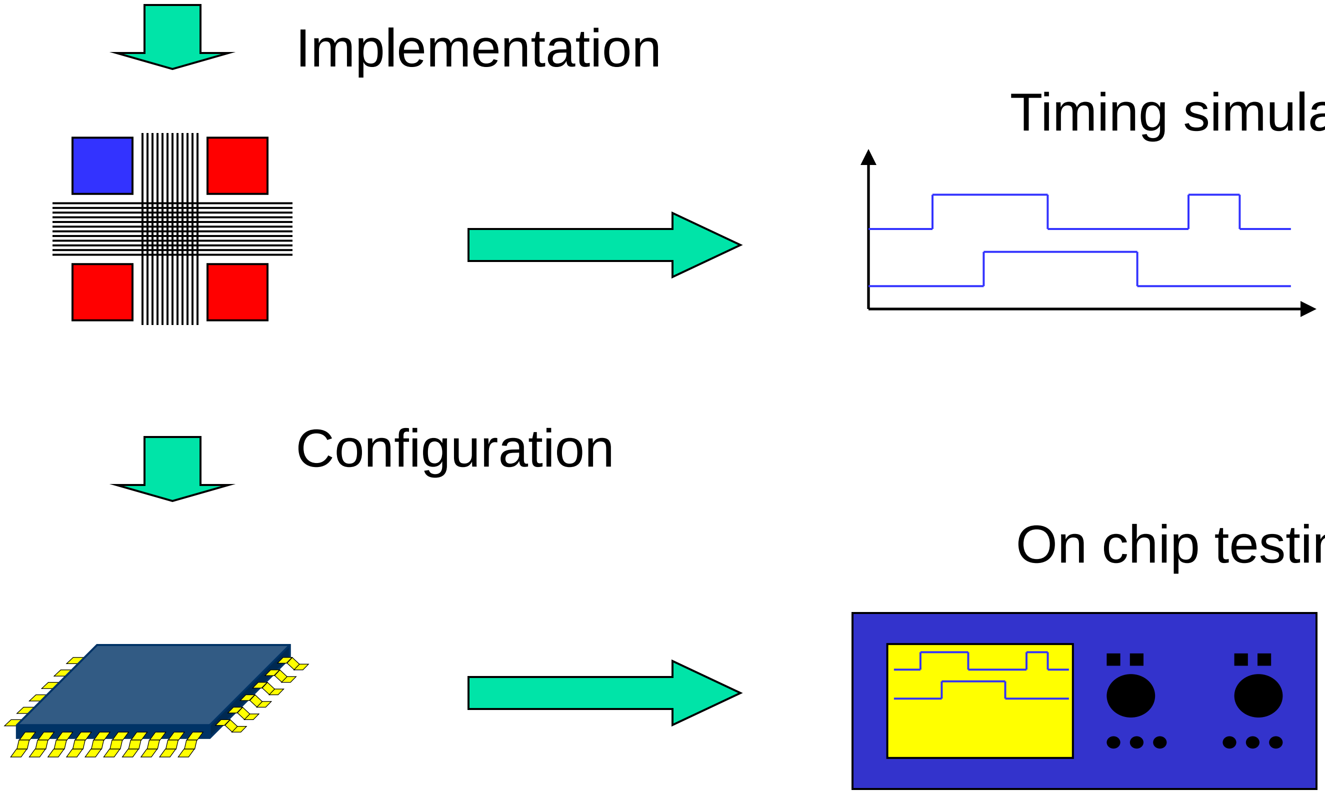
Synthesis



Post-synthesis simulation



Design process (2)



Implementation

Timing simulation

Configuration

On chip testing

Major FPGA Vendors

- SRAM-based FPGAs
  - **Xilinx, Inc**
  - **Altera Corp.**
  - Atmel
  - Lattice Semiconductor

Share 80% of the market

- Flash & Antifuse FPGAs
  - Actel Corp.
  - Quick logic Corp.

FPGA Vendors and Device families

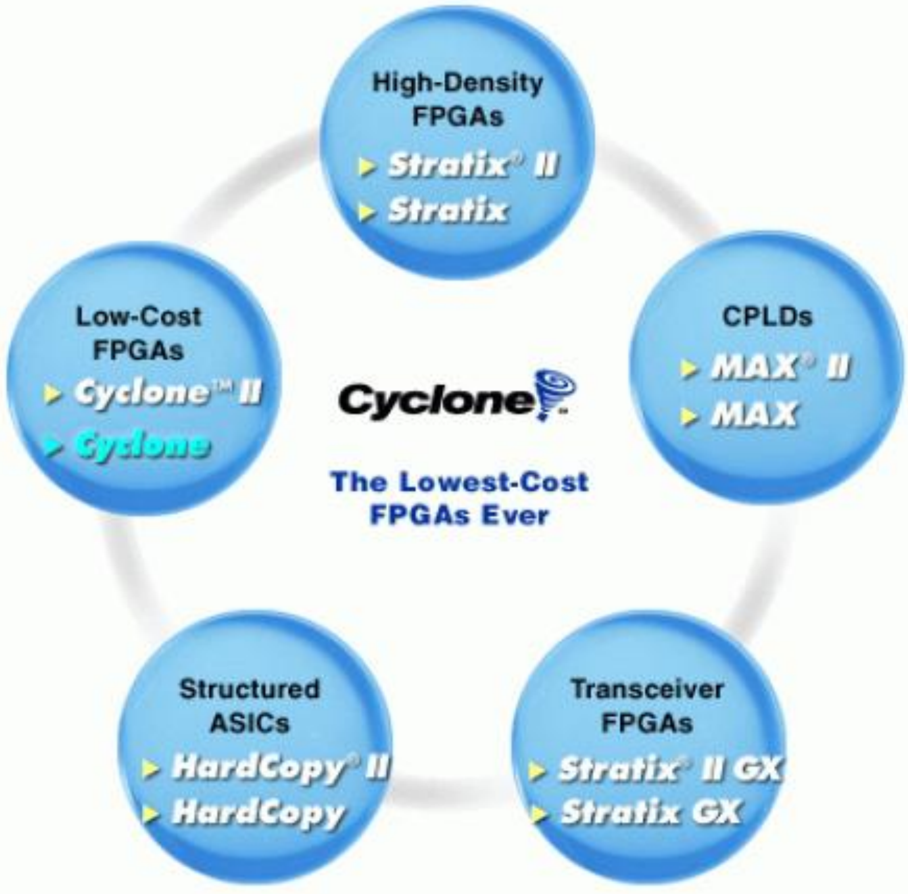
- Xilinx
  - Spartan
  - Virtex
  - Kintex
  - Artix
- Altera
  - Stratix
  - Cyclone
  - MAX 3000/7000 CPLD
  - MAX-II

Xilinx Families

- Old families
  - XC3000, XC4000, XC5200
  - Old 0.5µm, 0.35µm and 0.25µm technology. Not recommended for modern designs.
- High-performance families
  - Virtex (0.22µm)
  - Virtex-E, Virtex-EM (0.18µm)
  - Virtex-II, Virtex-II PRO (0.13µm)
  - Virtex-4 (0.09µm)
- Low Cost Family
  - Spartan/XL – derived from XC4000
  - Spartan-II – derived from Virtex
  - Spartan-IIE – derived from Virtex-E
  - Spartan-3 – derived from Virtex-II

Altera Families

- **Old Families**
  - FLEX 10K, FLEX 6000, FLEX 8000
- **High-performance Families**
  - Mercury
  - Stratix, Stratix GX, Stratix II
  - APEX 20K , APEX II
  - Excalibur
- **Low Cost Family**
  - Cyclone, Cyclone II



Device Family Overview

Simulation and Synthesis Tools

Active-HDL™
Complete FPGA Verification Environment

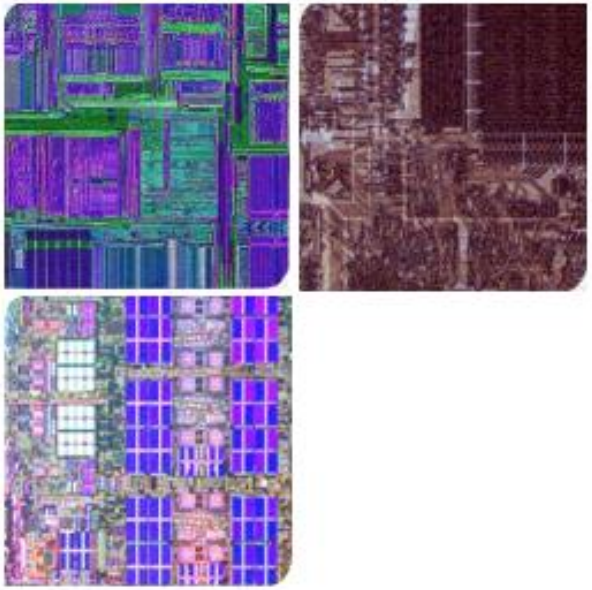
ModelSim®

Synplicity®

XILINX®

SYNOPSYS®

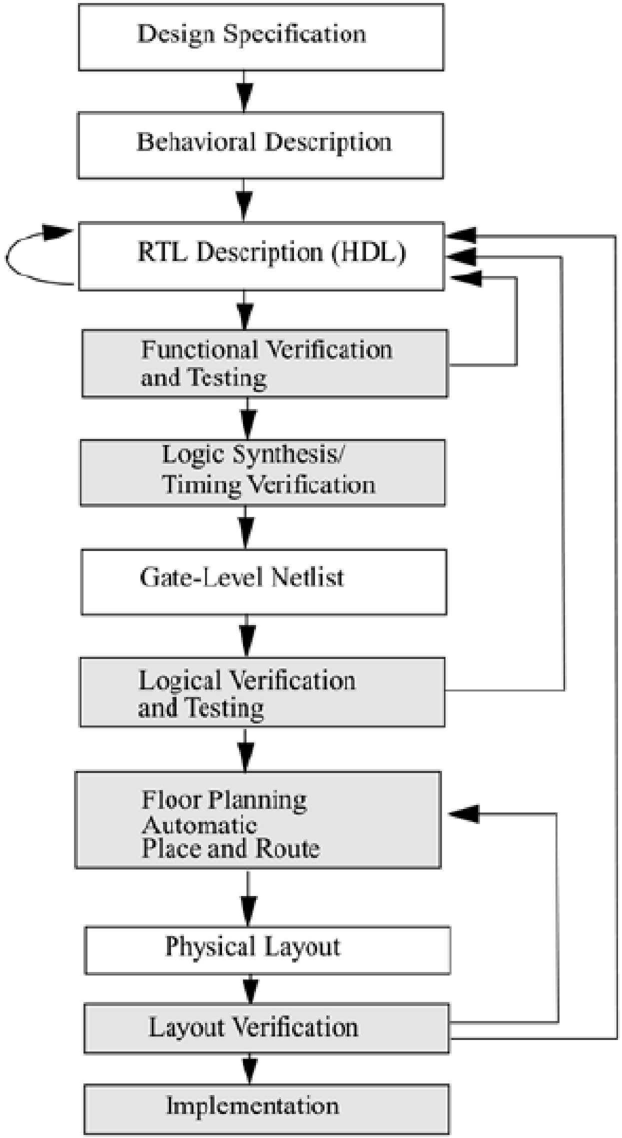
Verilog HDL



Verilog

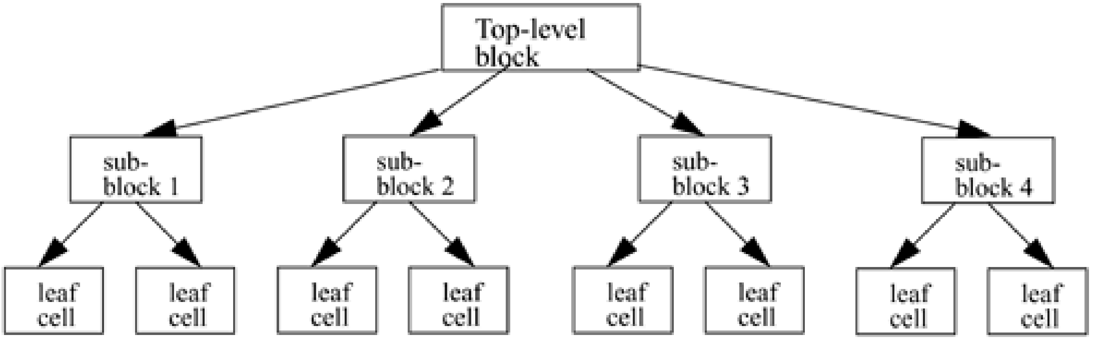
- Automated Integrated Design Systems (Gateway Design Automation) in 1986
- Initially a simulation language - more complete and easier to use than its predecessors
- Simulation, Documentation and Synthesis
- Synopsys introduced synthesis from Verilog in 1987
- Event-driven simulation
- Loosely typed language
- Hardware concurrency

Design Flow

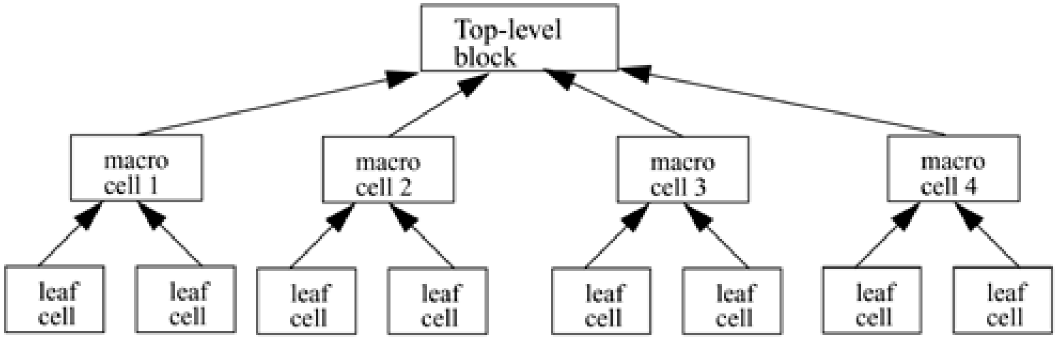

Design Specification
Behavioral Description
RTL Description (HDL)
Functional Verification and Testing
Logic Synthesis/ Timing Verification
Gate-Level Netlist
Logical Verification and Testing
Floor Planning Automatic Place and Route
Physical Layout
Layout Verification
Implementation

Design Methodology

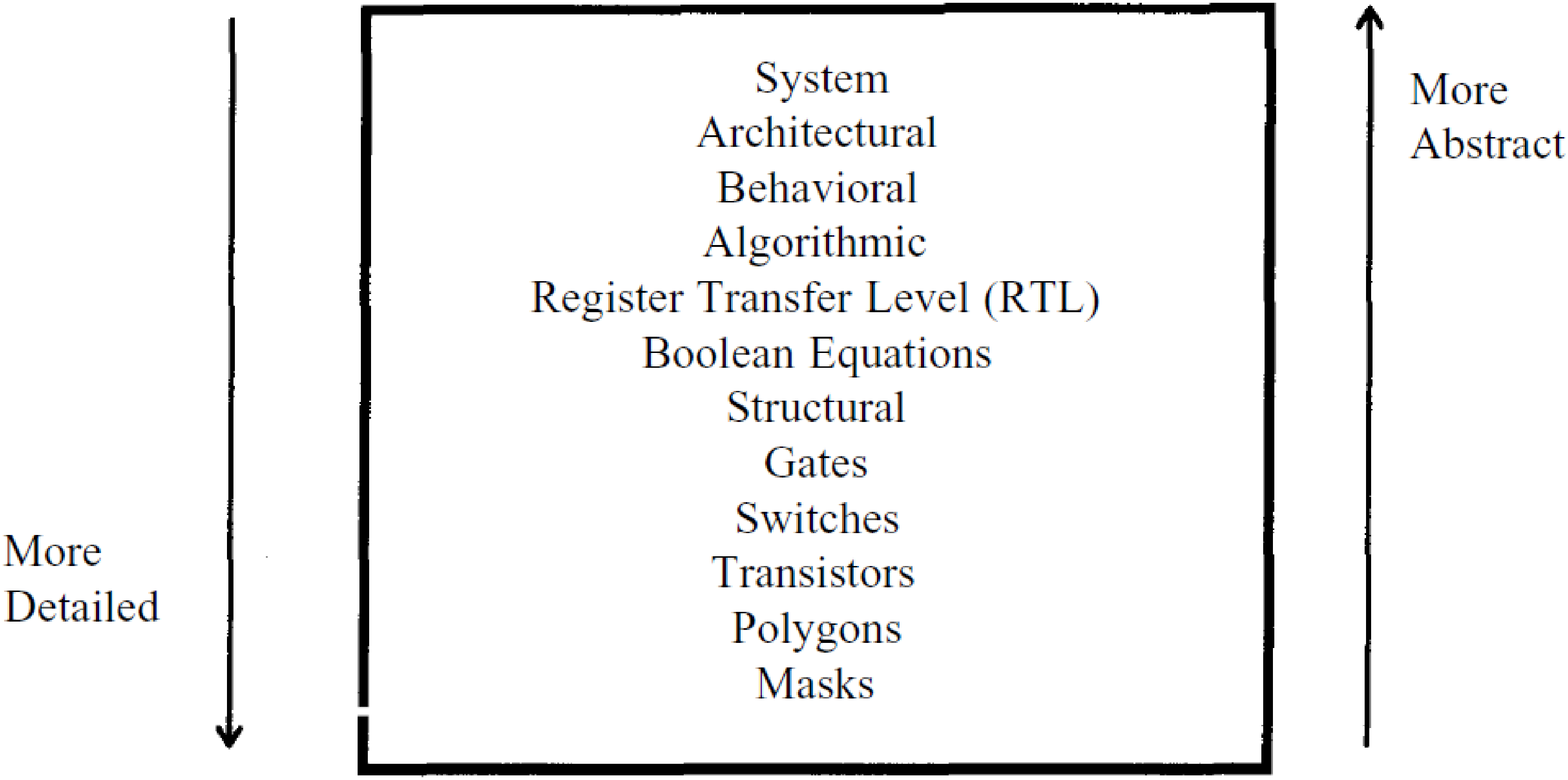
- Top-down Design Methodology



- Bottom-up Design Methodology



Design Abstraction Hierarchy



More
Abstract

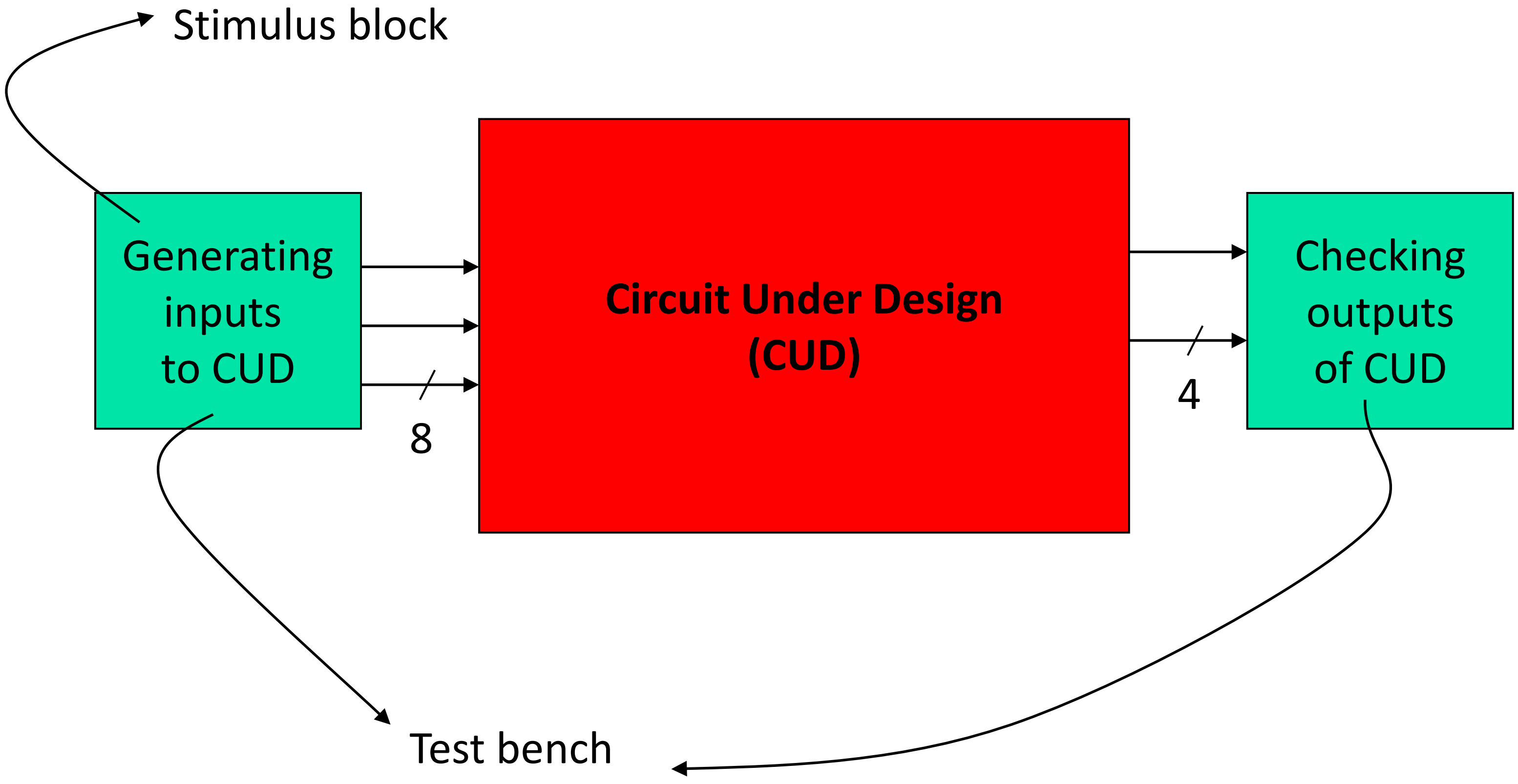
System
Architectural
Behavioral
Algorithmic
Register Transfer Level (RTL)
Boolean Equations
Structural
Gates
Switches
Transistors
Polygons
Masks

More
Detailed

Design Abstraction Levels

- Switch level
- Gate level
- Dataflow level
- Behavioral or algorithmic level

Components of Simulation


Stimulus block
Generating inputs to CUD
8
Circuit Under Design (CUD)
4
Checking outputs of CUD
Test bench