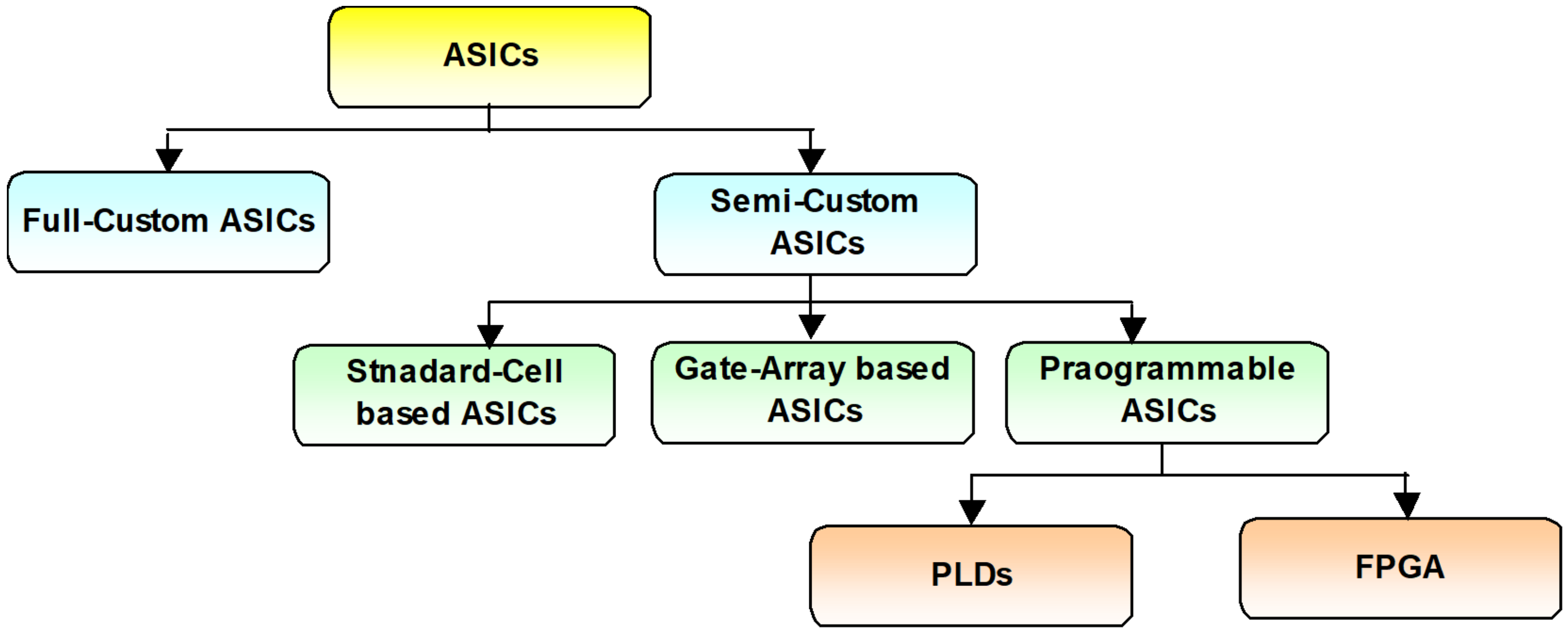
# INTRODUCTION TO DIGITAL SYSTEM DESIGN

Handong Global University

# Tyeps of ASICs



Full-Custom ASICs: Possibly all logic cells and all mask layers customized

Semi-Custom ASICs: all logic cells are pre-designed and some (possibly all) mask layers customized

# Tyeps of ASICs

- Standard cell
  - Designer uses a library of standard cells
  - Design time can be much faster than full custom.
  - Manufacturing cost is the same as that of full custom.
- 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.
  - A routing tool creates the masks and customizes the pre-created gate array for the user's design.
  - Fabrication costs are cheaper than standard cell, and fabrication time can be very short.

# Characteristics of ASICs

- In 2010
  - Die area 2.5x2.5 cm
  - Voltage: 0.6V
  - Technology: 0.07μm

| | Density (Mgates/cm$^2$) | Max. Ave. power (W/cm$^2$) | Clock Rate (3 GHz) |
|---|---|---|---|
| Custom | 25 | 54 | 3 |
| Std. Cell | 10 | 27 | 1.5 |
| Gate Array | 5 | 18 | 1 |
| FPGA | 0.4 | 4.5 | 0.25 |

## Programmable Logic Devices

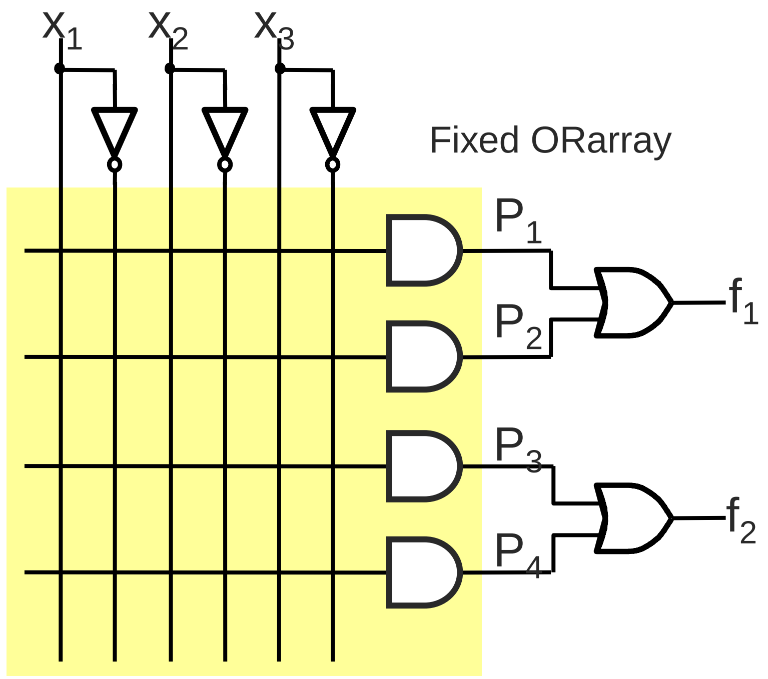
# PLD (Programmable Logic Device)

- Many types of PLDs
  - PAL: Programmable Array Logic
  - PLA: Programmable Logic Array
  - ROM: Read Only Memory
  - CPLD: Complex PLD
  - FPGA: Field Programmable Gate Array

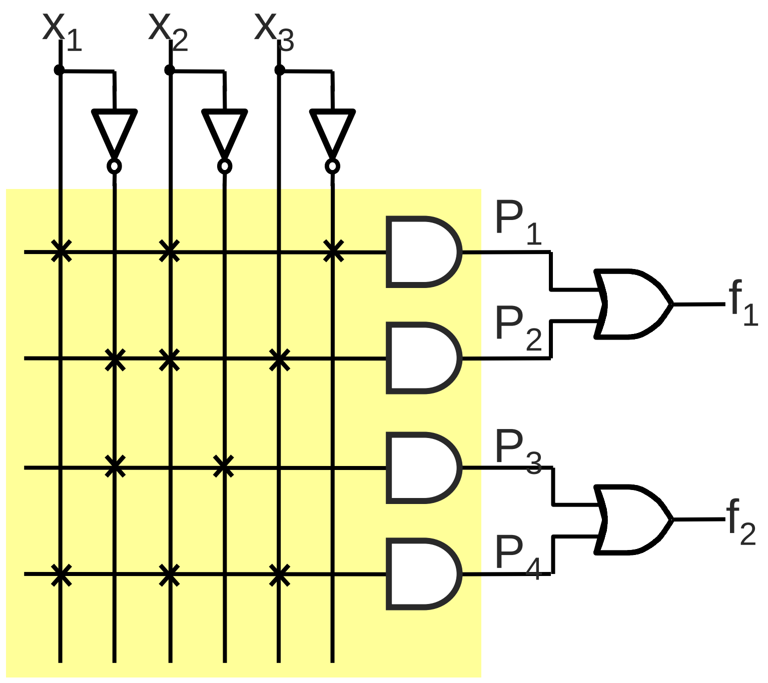
# PAL

$f_1 = x_1x_2x_3'+x_1'x_2x_3$

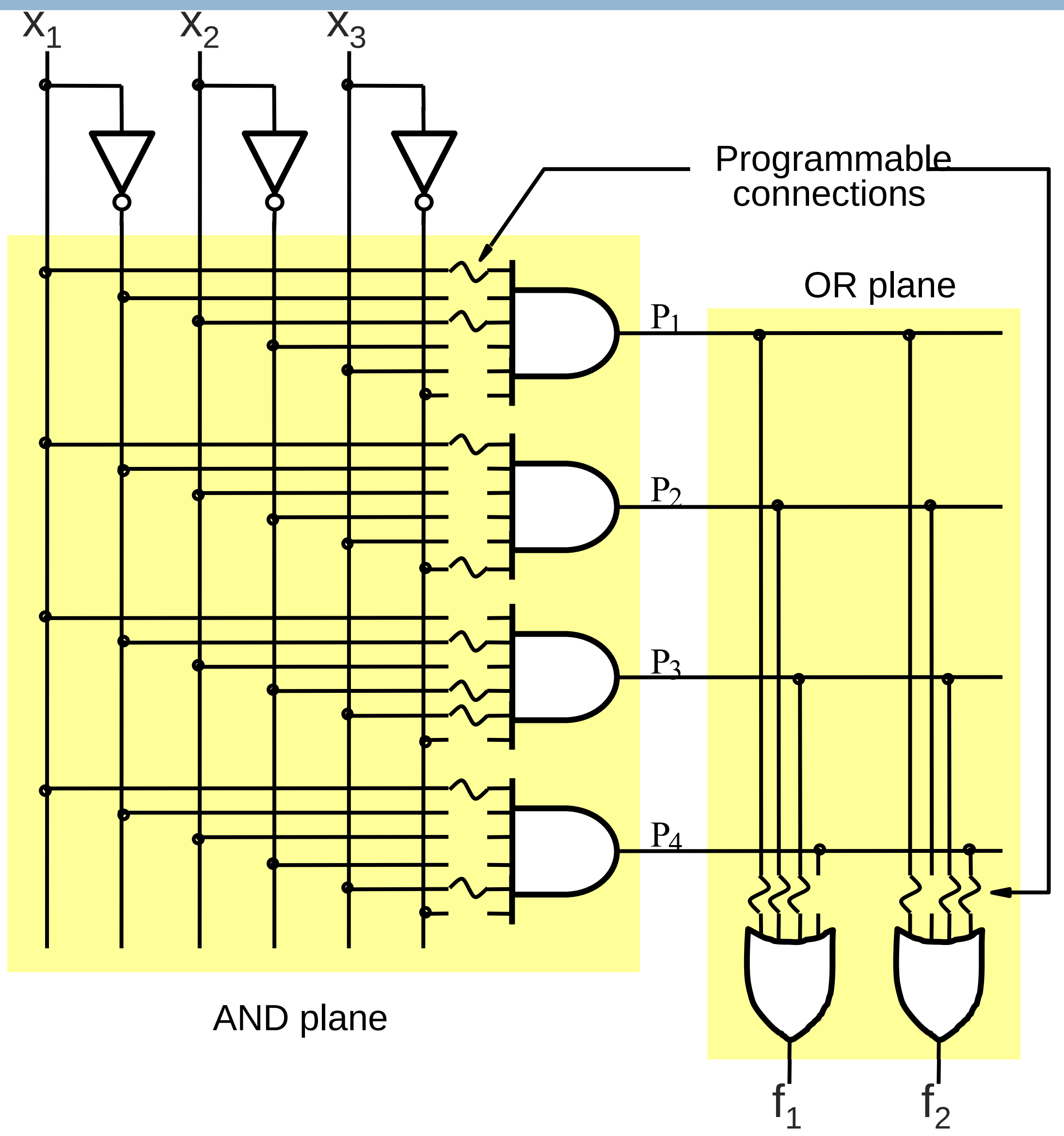
$f_2 = x_1'x_2'+x_1x_2x_3$



Programmable AND array

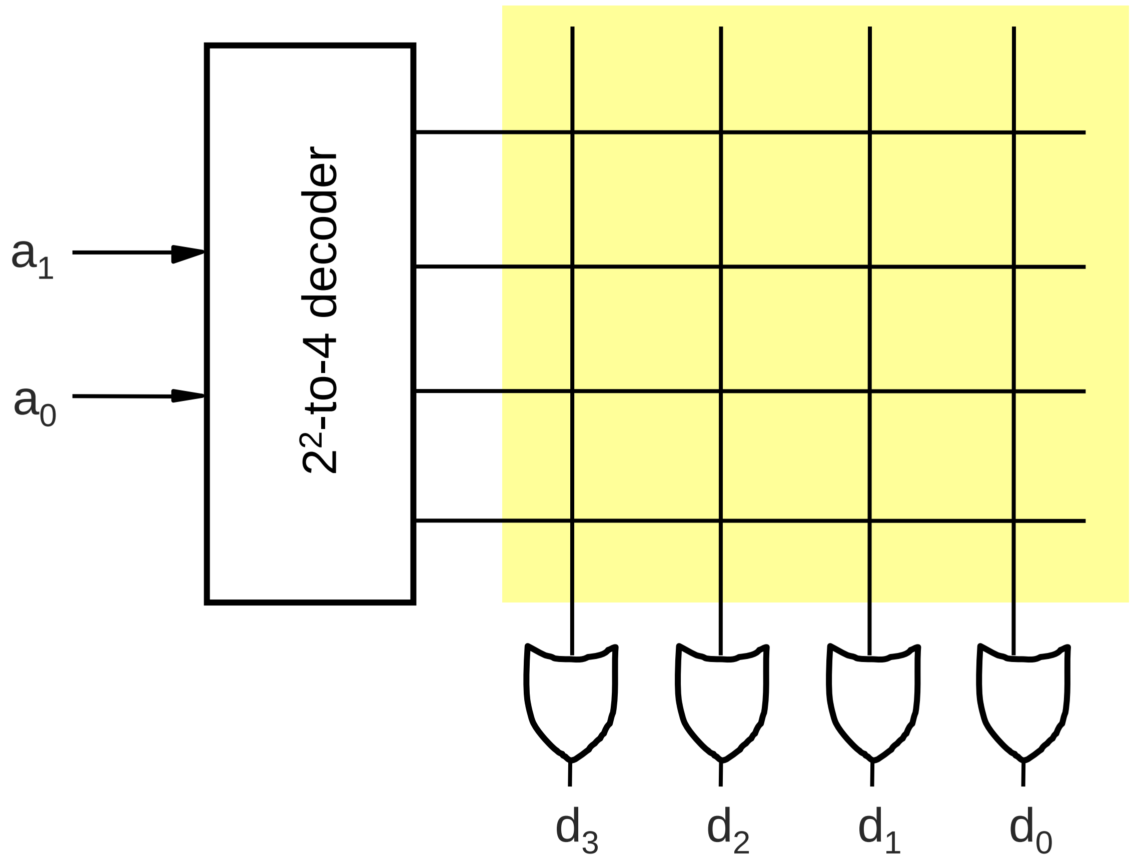


# PLA

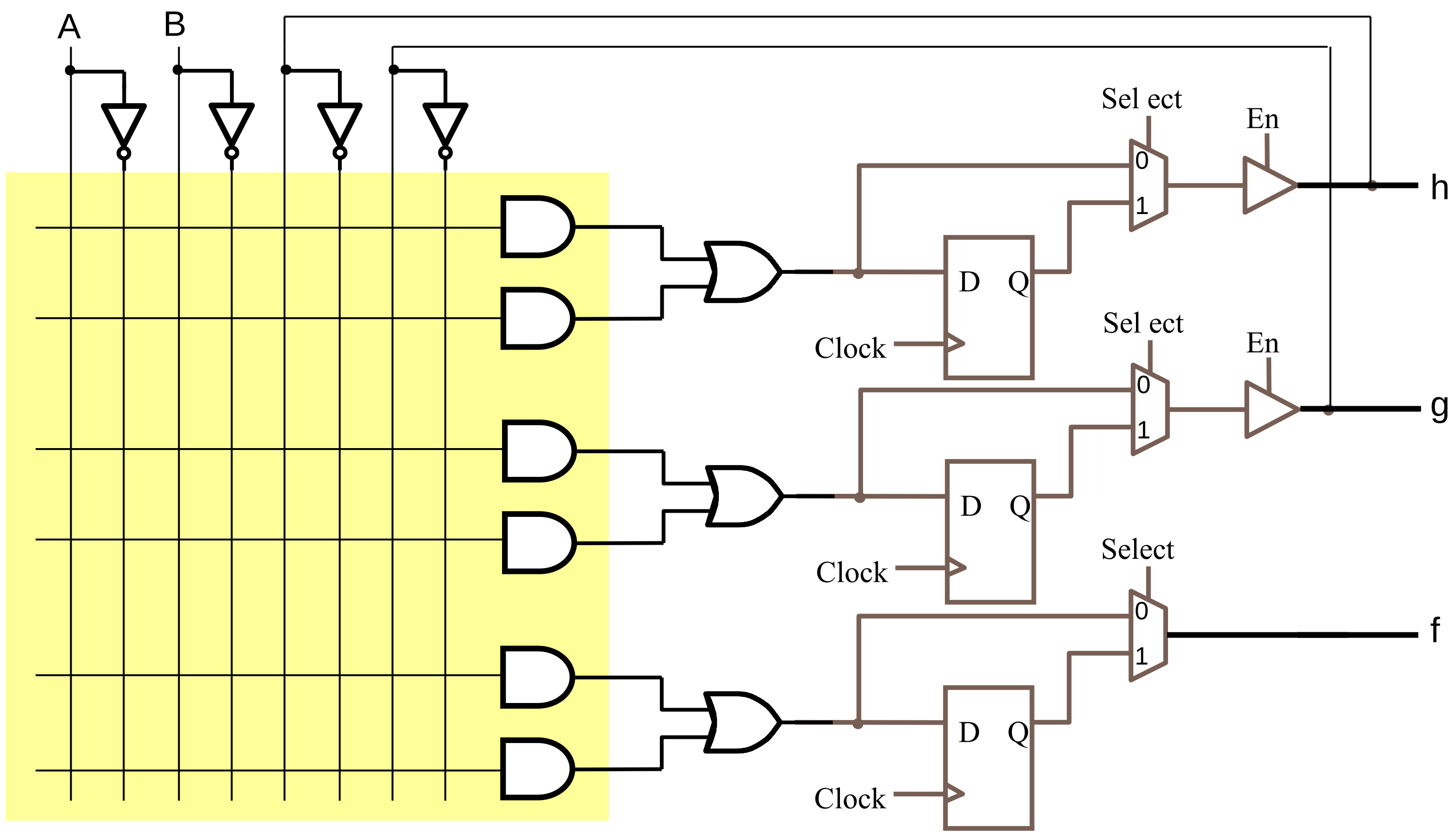

$x_1$
$x_2$
$x_3$
Programmable connections
OR plane
$P_1$
$P_2$
$P_3$
$P_4$
AND plane
$f_1$
$f_2$

# ROM

- A ROM has a fixed AND plane and a programmable OR plane.



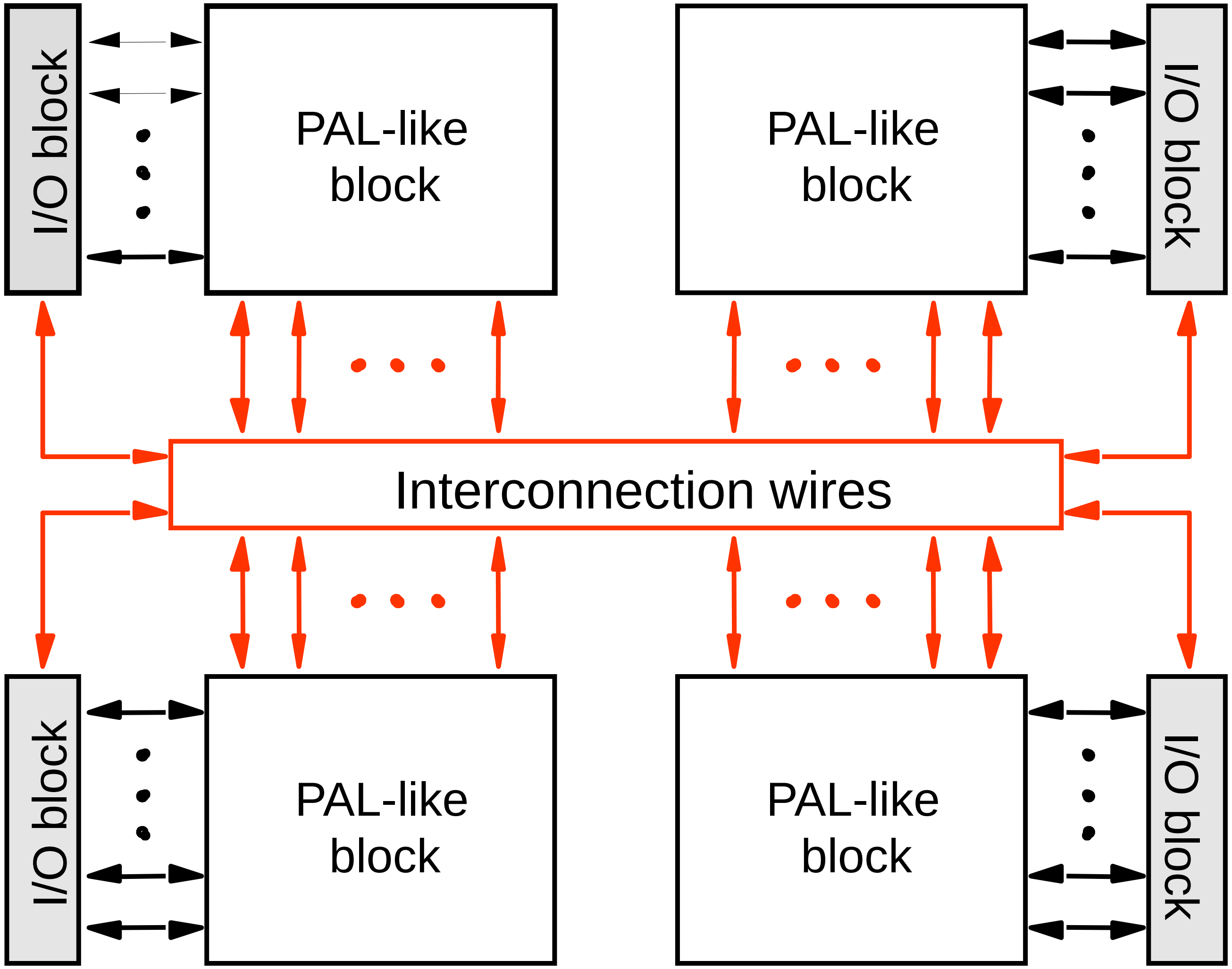
# PAL with macrocell function



# CPLD

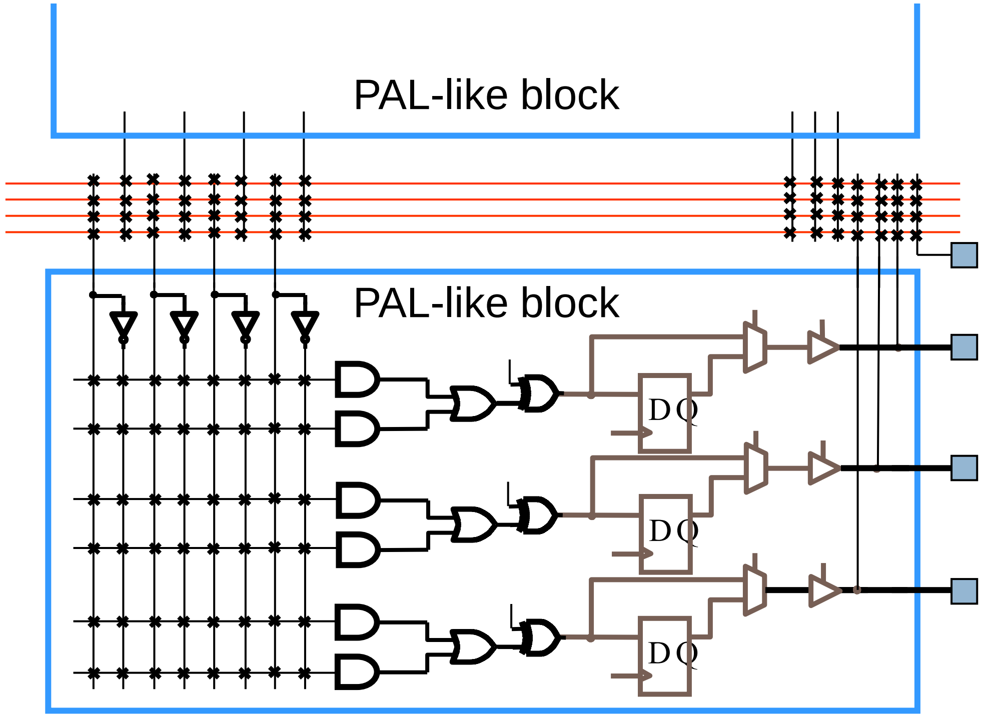
- SPLDs (PLA, PAL) are limited in size due to the small number of input and output pins and the limited number of product terms
- CPLDs contain multiple circuit blocks on a single chip
  - Each block is like a PAL: PAL-like block
  - Connections are provided between PAL-like blocks via an interconnection network that is programmable
  - Each block is connected to an I/O block as well

# Structure of CPLD



# Internal structure of CPLD

- Includes macrocells
- Fixed OR planes
- XOR gates provide negation ability
  - XOR has a control input

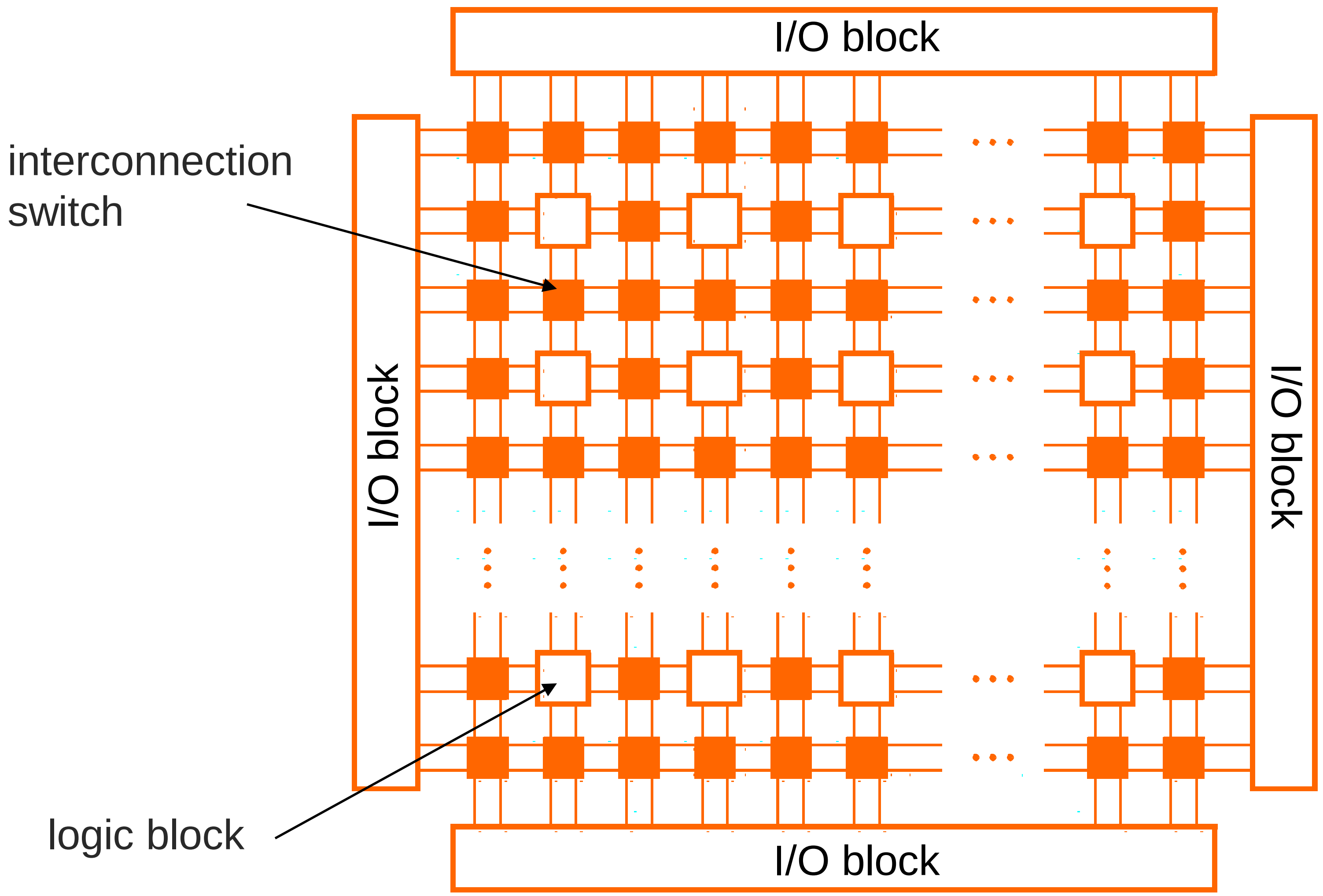


## FPGA (Field Programmable Gate Array)

- SPLDs and CPLDs are relatively small and useful for simple logic devices
  - Up to about 20000 gates

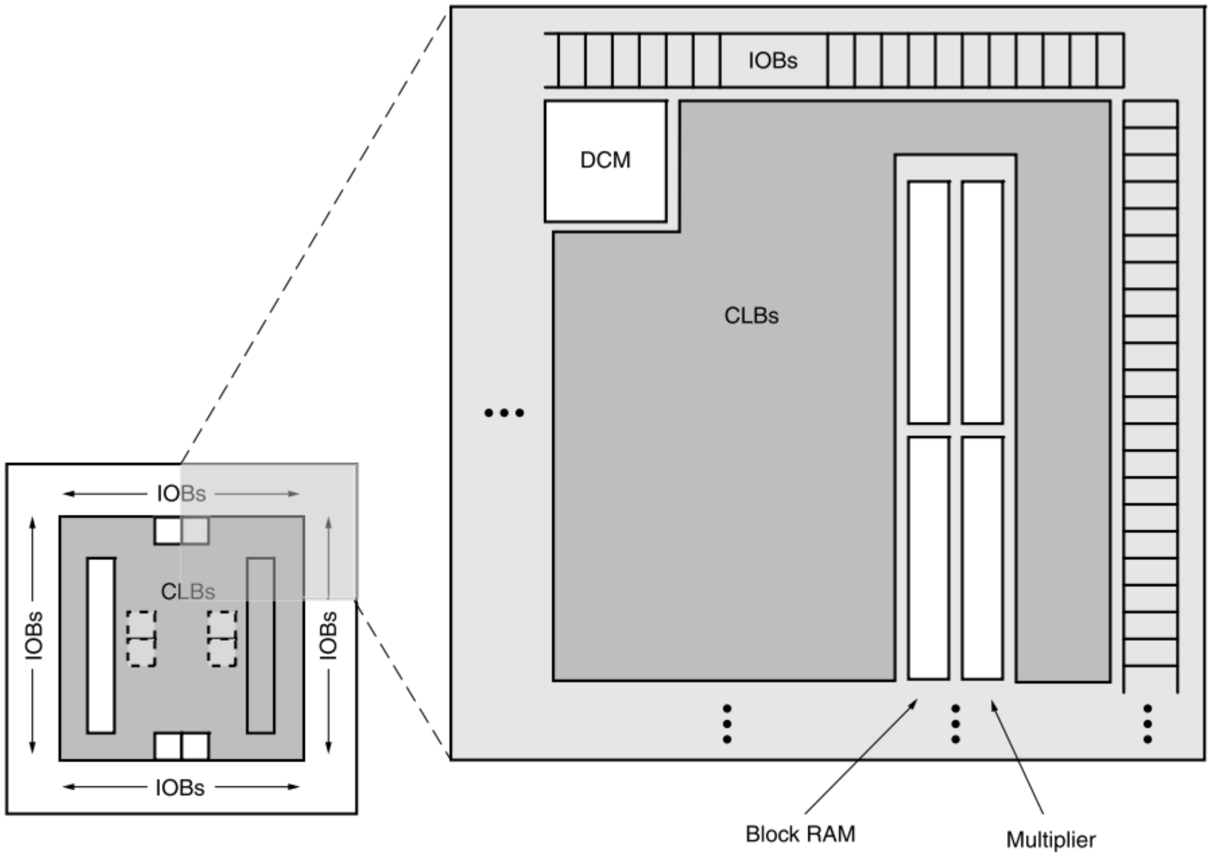
- FPGAs can handle larger circuits
  - No AND/OR planes
  - Provide logic blocks, I/O blocks, and interconnection wires and switches
  - Logic blocks provide functionality
  - Interconnection switches allow logic blocks to be connected to each other and to the I/O pins

# Structure of FPGA



# Xilinx FPGA: Spartan 3E

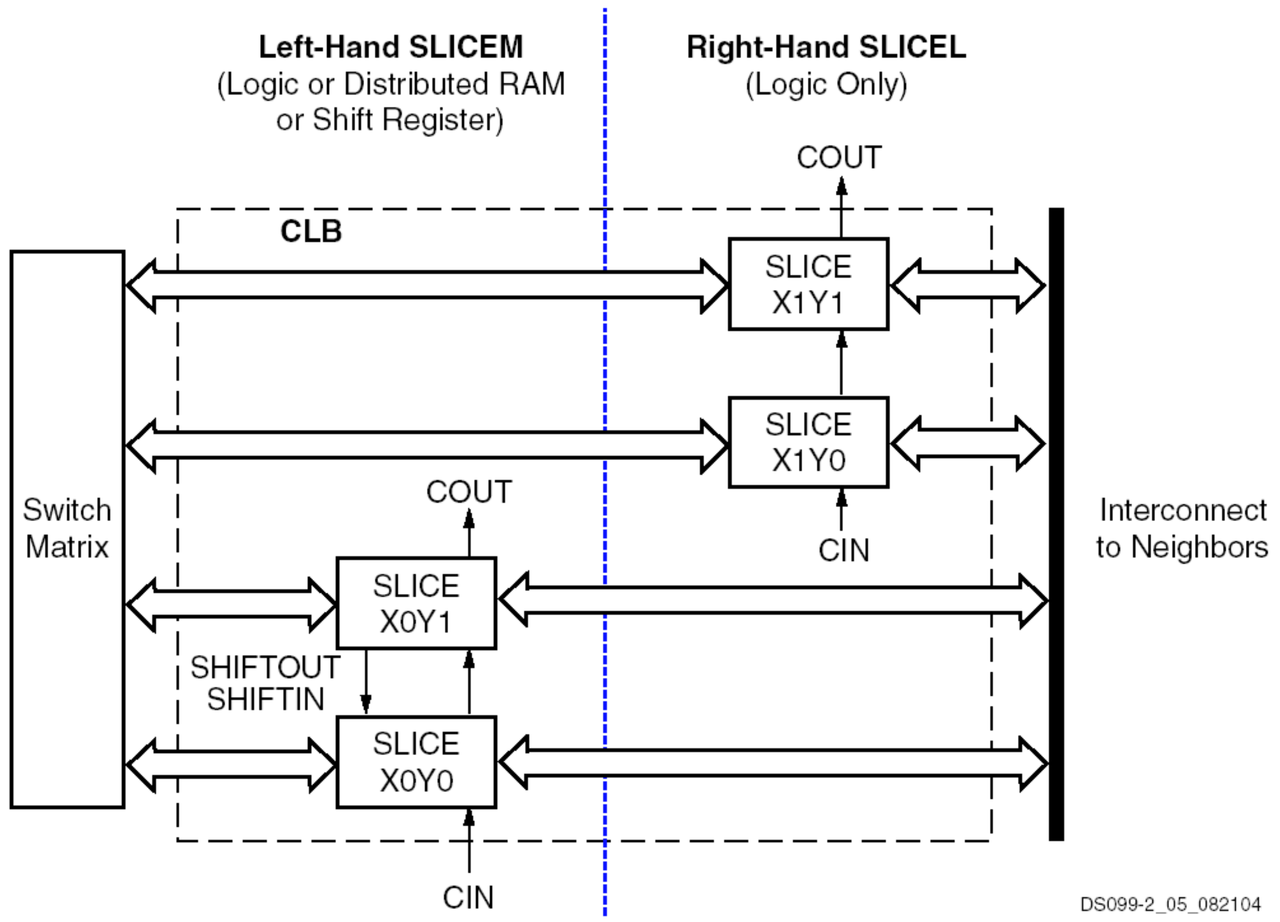
- Structure



IOB: IO Block, CLB: Configurable Logic Block, DCM: Digital Clock Manager

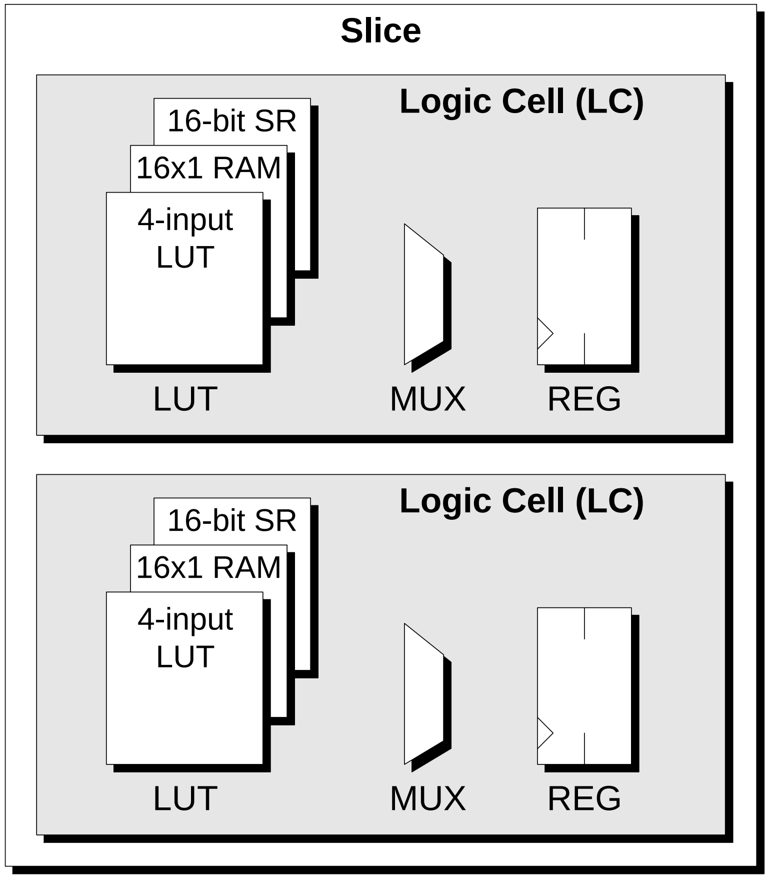
# Spartan 3E: CLB structure

- 4 slices/CLB

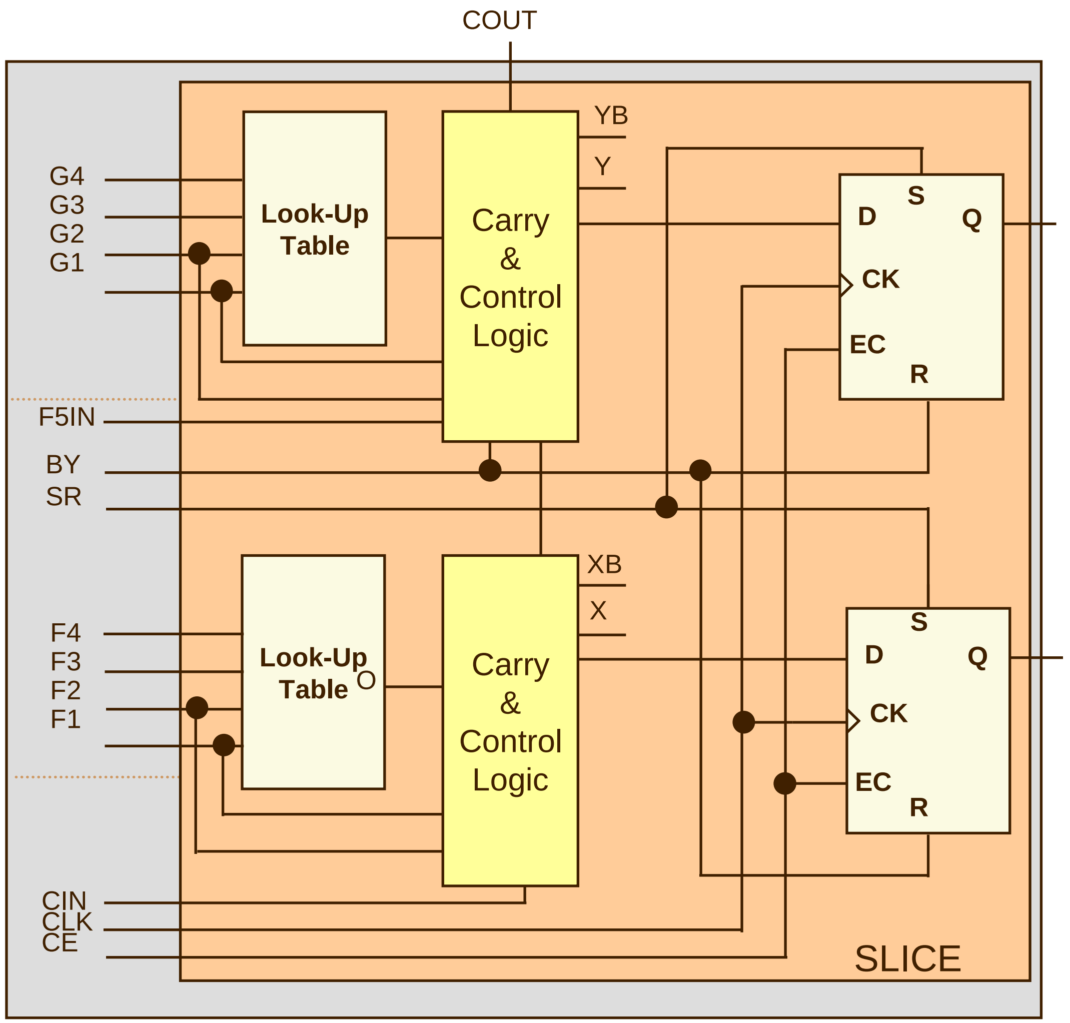


# Spartan 3E: Slice structure

- Each slice contains two sets of the following:
  - Four-input LUT(look-up table)
    - Any 4-input logic function,
    - or 16-bit x 1 sync RAM (SLICEM only)
    - or 16-bit shift register (SLICEM only)
  - Carry & Control
    - Fast arithmetic logic
    - Multiplier logic
    - Multiplexer logic
  - Storage element
    - Latch or flip-flop
    - Set and reset
    - True or inverted inputs
    - Sync. or async. control



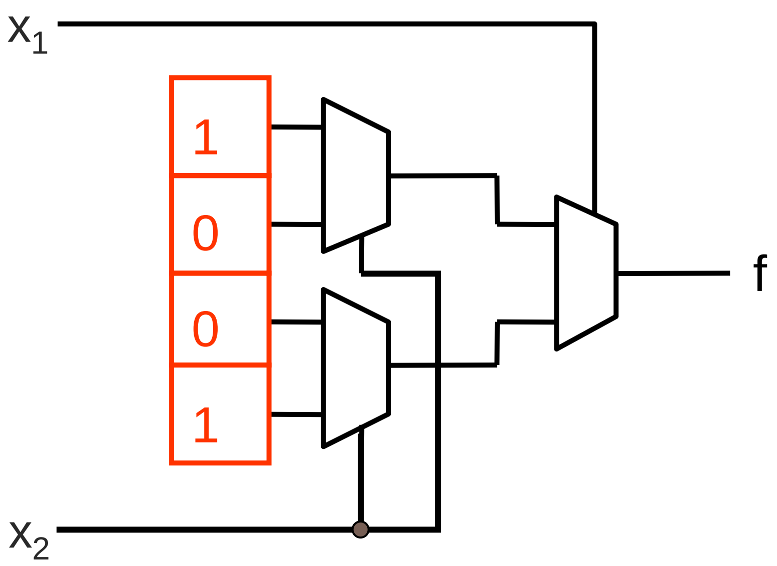
# Spartan 3E: Slice structure



# LUT

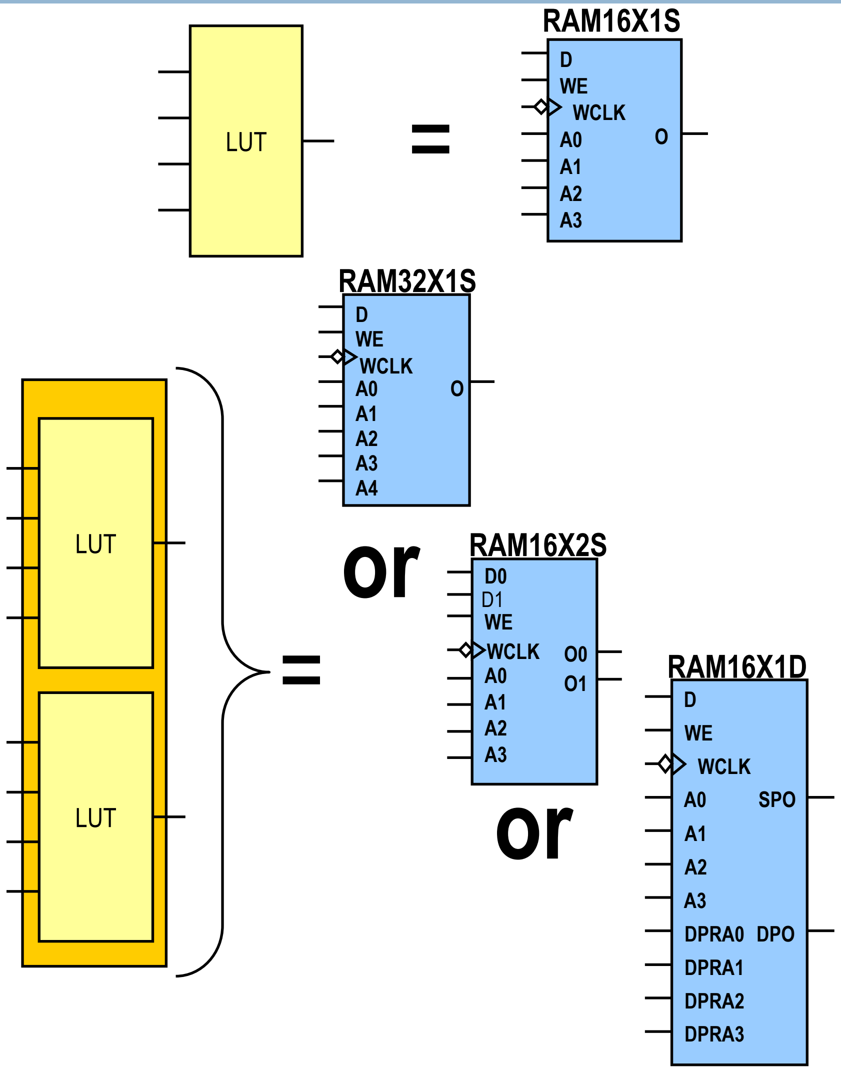
- Ex.: 2-input LUT

| $x_1$ | $x_2$ | f |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |



# Spartan 3E: Distributed RAM

- CLB LUT configurable as Distributed RAM
  - A single LUT equals 16x1 RAM
  - Two LUTs Implement Single and Dual-Port RAMs
  - Cascade LUTs to increase RAM size



## VLSI Design

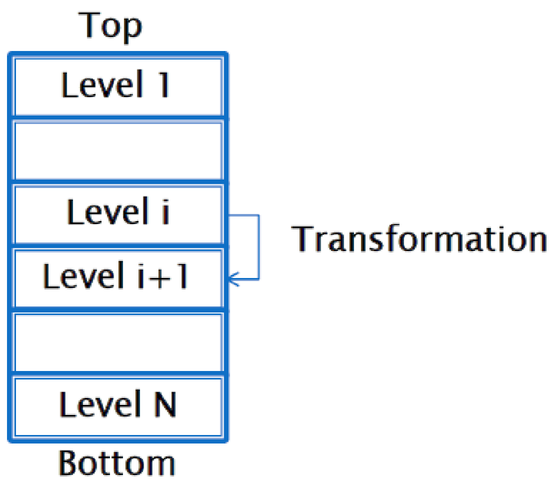
- VLSI Design process:
  - An iterative process that refines an *idea* to a *device* (through the design abstraction).

$$\text{The idea} \xrightarrow[\text{a set of requirements}]{\text{refined into}} \textit{Specification:}$$

- What does the device do?
- How fast does it need to operate? (speed constraint)
- How much power will it consume? (power constraint)
- How big will it be? (area constraint)

# VLSI Design

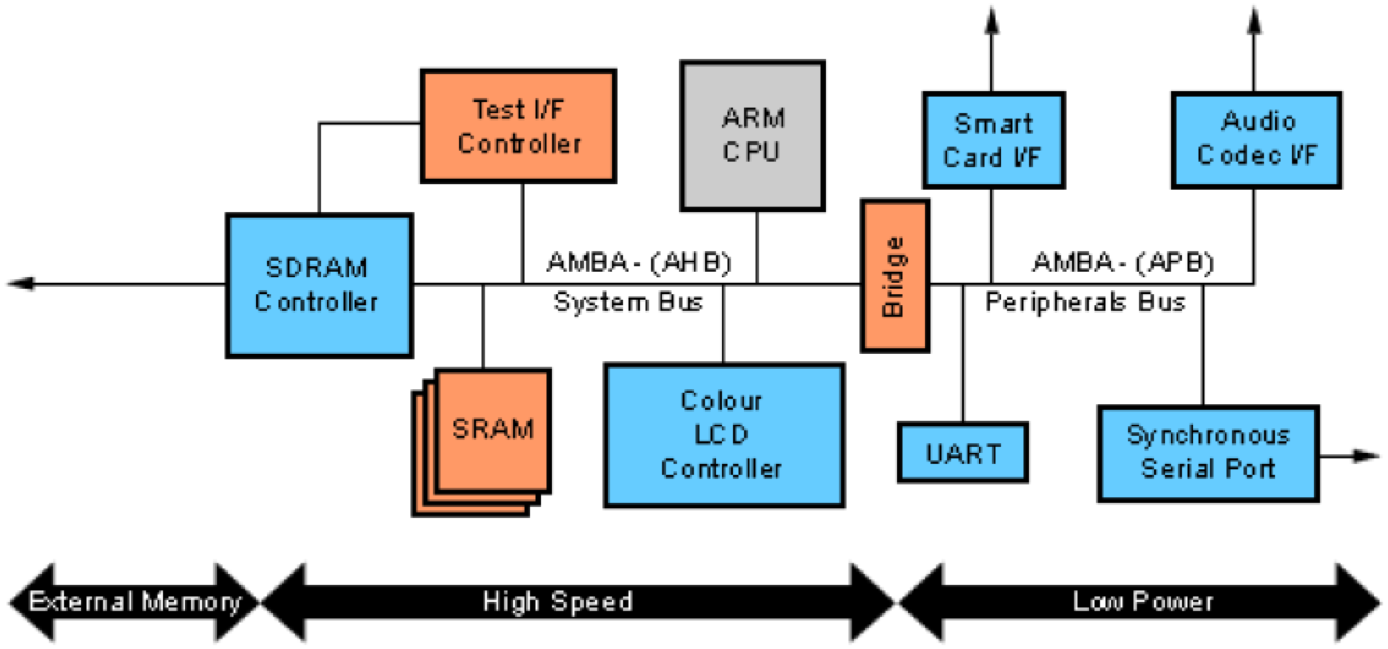
- Abstraction
  - A very effective means of dealing with design complexity.
  - Creating a model at a higher level of abstraction involves replacing detail at the lower level with simplifications.
- Abstraction hierarchy
  - A set of interrelated representation levels that allow a system to be represented in varying amount of detail



## Digital Design

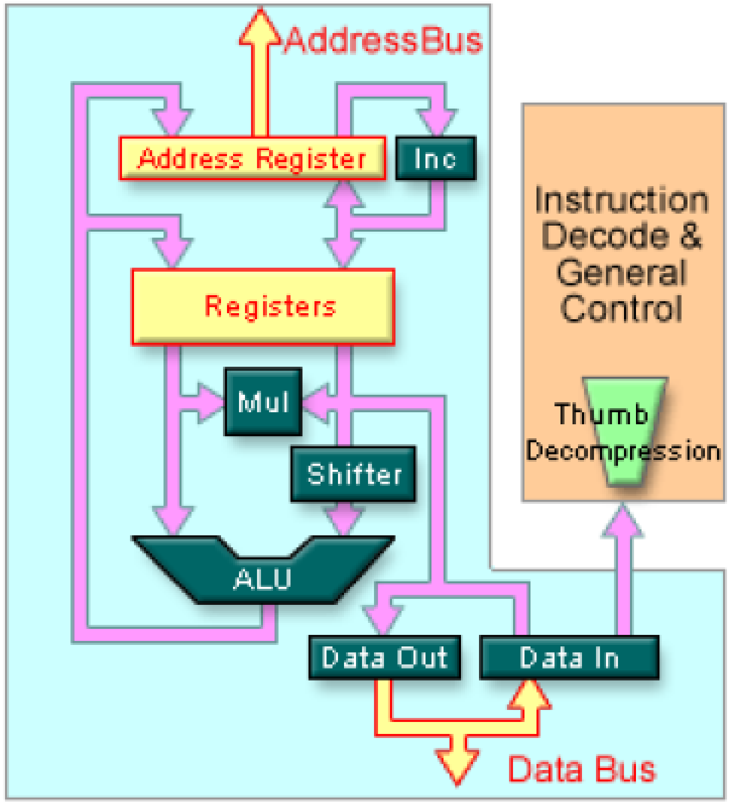
# Examples of Structural Domain

- Example of ARM processor and sub-modules

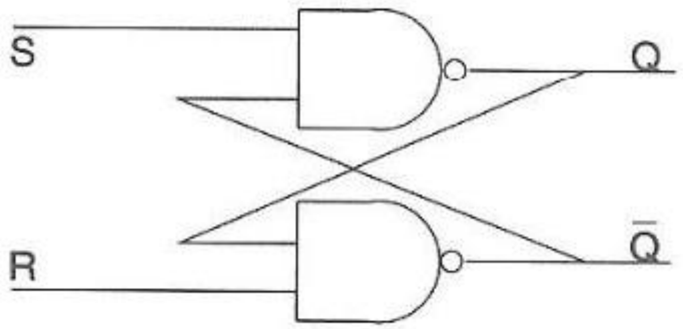


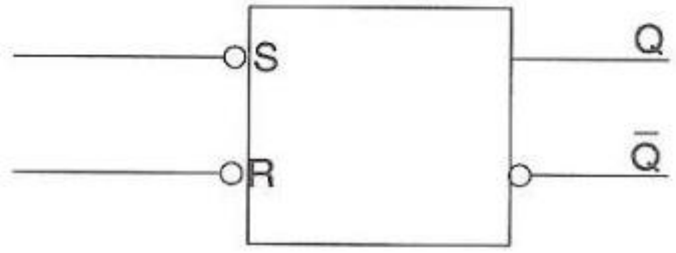
## Examples of Structural Domain

- RTL



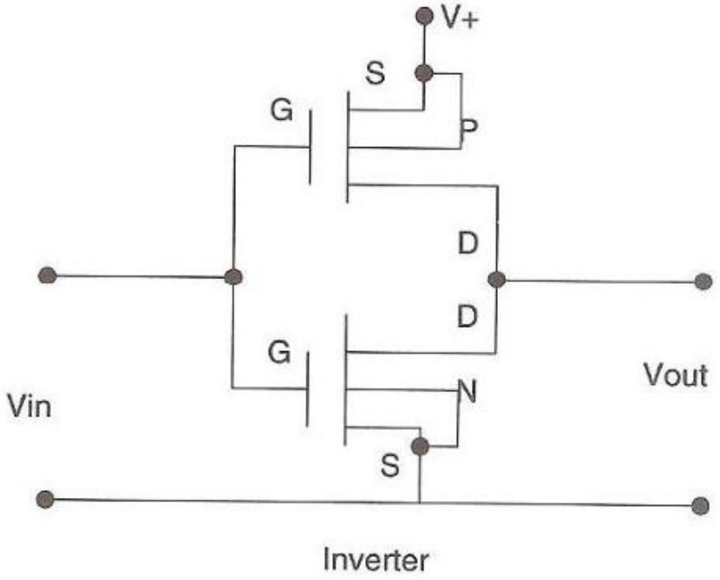
- Gate level

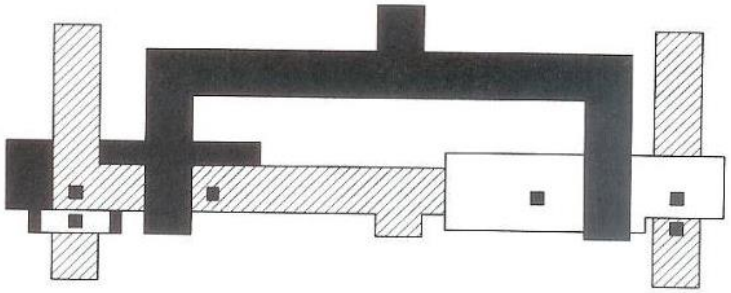




# Examples of Structural Domain

- Transistor Level
- Mask





# Detailed Design

- Choose the design entry method
  - Schematic
    - Intuitive & easy to debug
    - Not portable
    - Poor designer productivity (gates/time)
  - HDL (Hardware Description Language), e.g. Verilog, VHDL, SystemC
    - Requires some experience, harder to debug
    - Descriptive & portable
    - Easy to modify
    - Greater productivity
  - Mixed HDL & schematic

- Functional simulation
  - To verify the functionality of your design only
  - Preparation for simulation
    - Generate simulation patterns
      - Waveform entry
      - HDL testbench
    - Generate simulation netlist
  - Simulation results
    - Waveform display
    - Text output
    - Self-checking testbench
  - Challenge
    - Sufficient & efficient test patterns

- HDL Synthesis
  - Synthesis = Translation + Optimization
    - Translate HDL design files into gate-level netlist
    - Optimize according to your design constraints
      - Area constraints
      - Timing constraints
      - Power constraints
  - Main challenges
    - Learn synthesizable coding style
    - Use proper design partitioning for synthesis
    - Specify reasonable design constraints
    - Use HDL synthesis tools efficiently

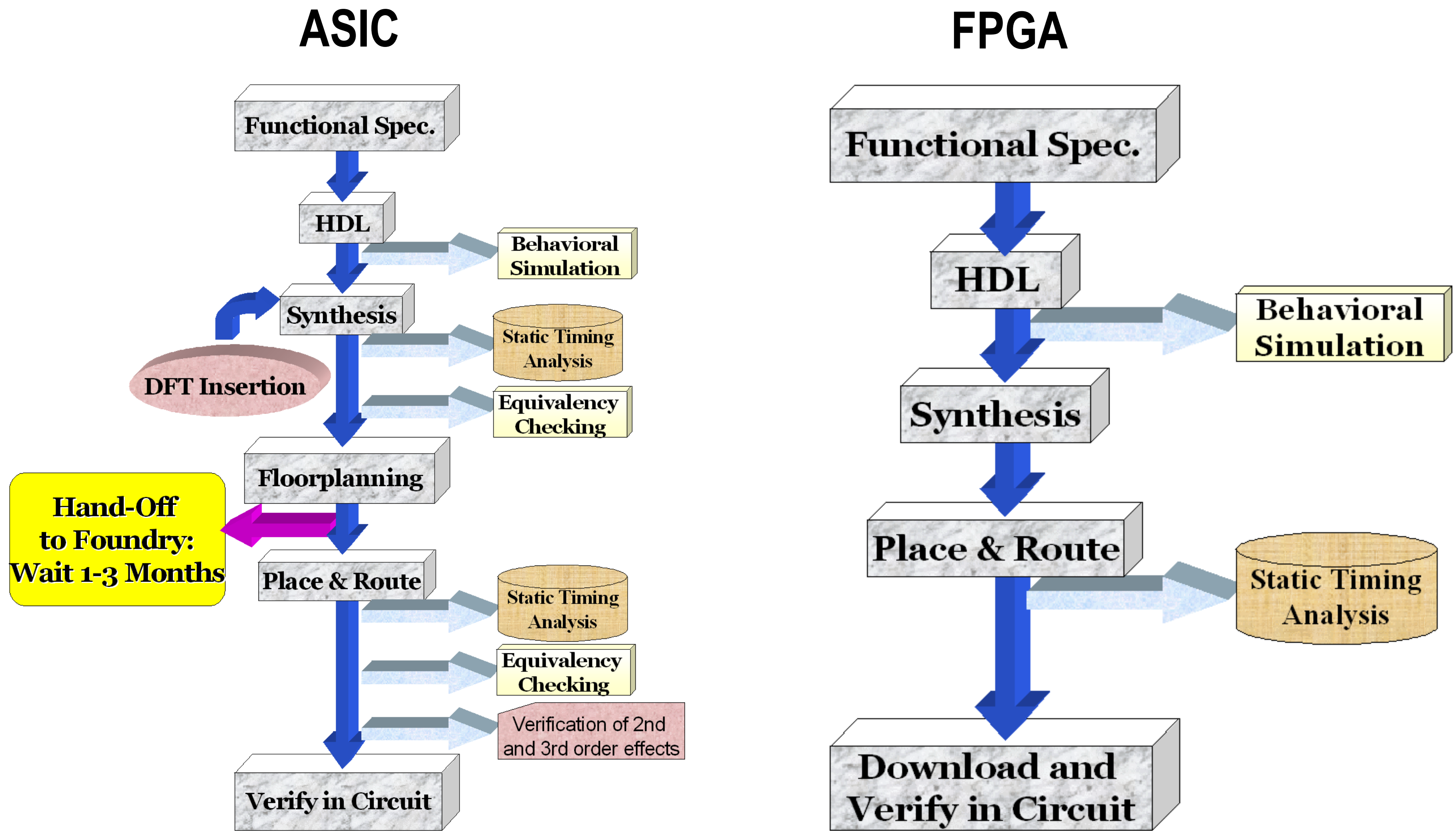
- Design implementation
  - Implementation flow
    - Netlist merging, flattening, data base building
    - Design rule checking
    - Logic optimization
    - Block mapping & placement
    - Net routing
    - Configuration bitstream generation (FPGA only)
    - Scan flip-flop insertion (ASIC only)
  - Implementation results
    - Design error or warnings
    - Device utilization (FPGA)
    - Die size (ASIC)
    - Timing reports
  - Challenge
    - How to reach high performance & high utilization implementation?

## FPGA Design Flow

## Design Flow in ASIC vs FPGA

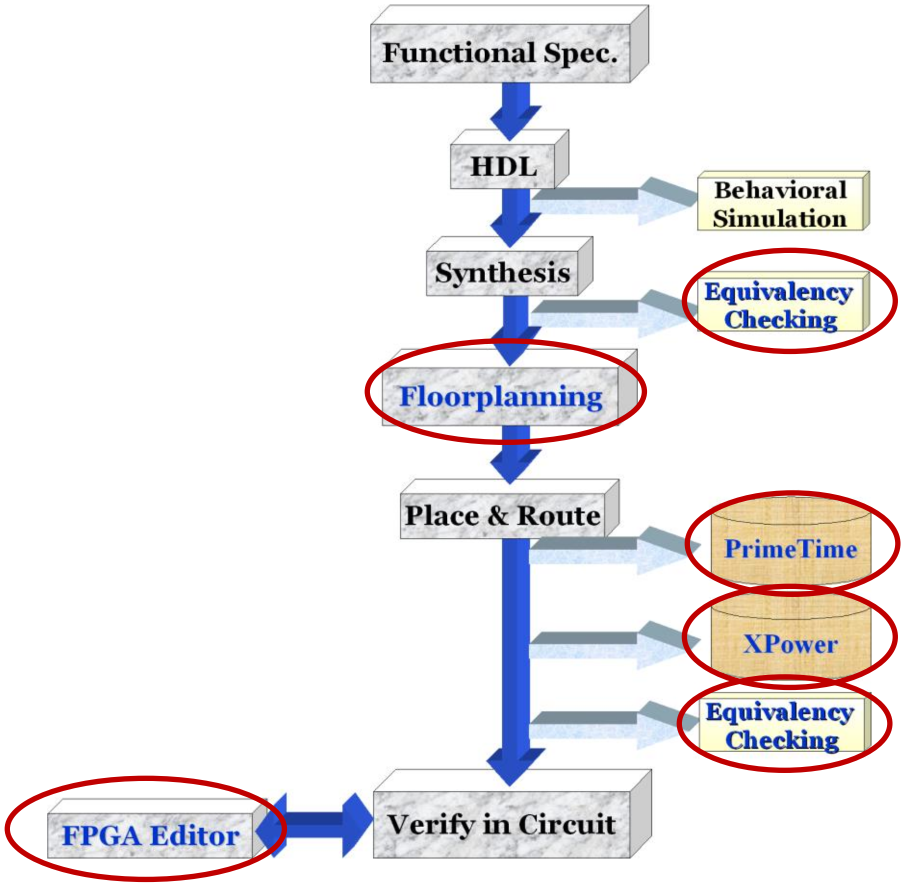
- ASIC and FPGA design and implementation methodologies differ moderately
  - Xilinx FPGAs provide for reduced design time and later bug fixes
    - No design for test logic is required
    - Deep sub-micron verification is done
    - No waiting for prototypes

# Design Flow Comparison



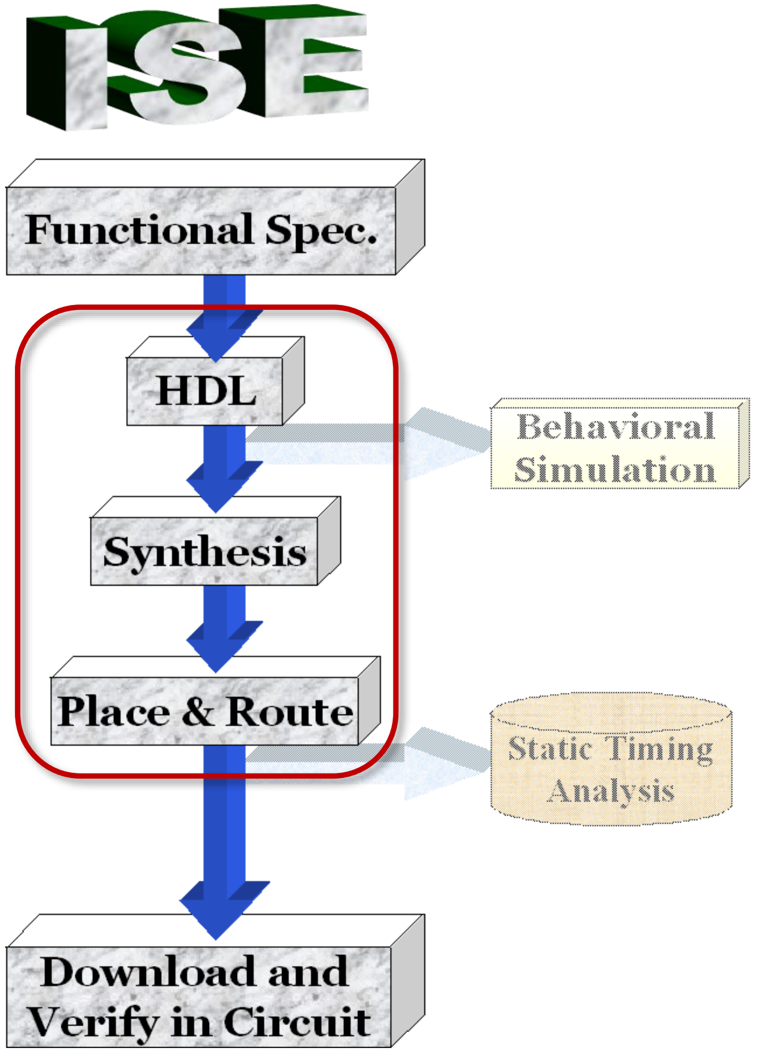
# Advanced FPGA Tool Flow

- Equivalency checking
  - Synopsys Formality
- Floorplanning and layout
  - Synopsys Amplify (physical synthesis)
  - Xilinx Floorplanner or PlanAhead$^{TM}$ software
- Static timing analysis
  - Synopsys PrimeTime
- Calculating power use
  - Xilinx XPower
- Edit routing and placement
  - Xilinx FPGA Editor



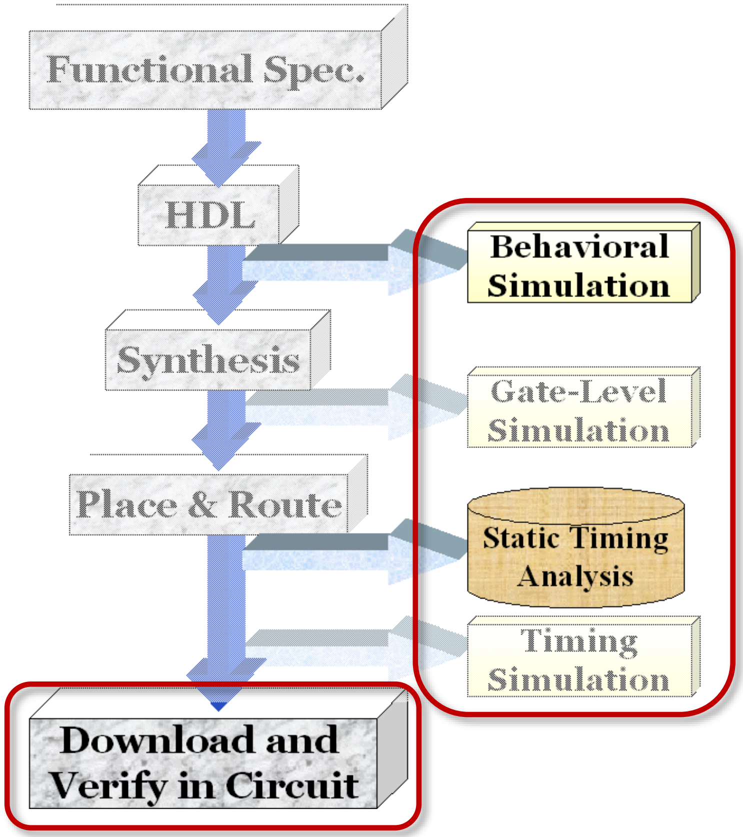
# FPGA Implementation

- Create HDL
  - Optimized for Xilinx FPGAs and performance
- Synthesis
  - XST, Synopsys, Mentor
  - Pushbutton flow with scripting capabilities
- Place & route
  - Completed by the user
  - Xilinx implementation tools – ISE® software
  - Pushbutton flow, scripting capabilities



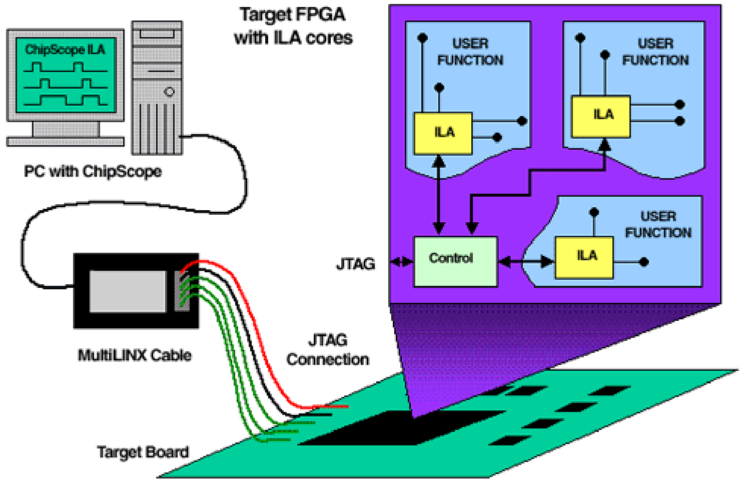
# FPGA Verification

- Three key verification points for FPGA implementation
  - Behavioral simulation
  - Post-place & route static timing analysis
  - Download and verify in circuit
- Post-synthesis gate-level simulation and post-place & route timing simulations can be done for production sign off
  - Post-place & route timing simulations are also often done to verify board- and system-level timing



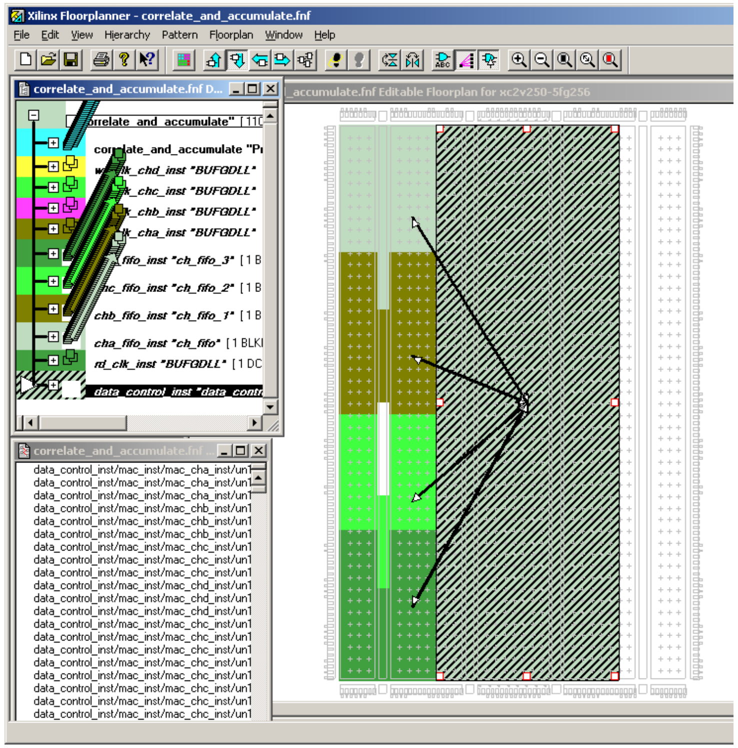
# In-Circuit Verification Tools

- ChipScope$^{TM}$ Pro software Integrated Logic Analysis (ILA) provides in-circuit logic verification through the dedicated JTAG pins
  - No need for extra headers
  - The ChipScope Pro software is a standalone tool for logic analysis
  - Data channels from 1 to 256; sample sizes from 256 to 4096



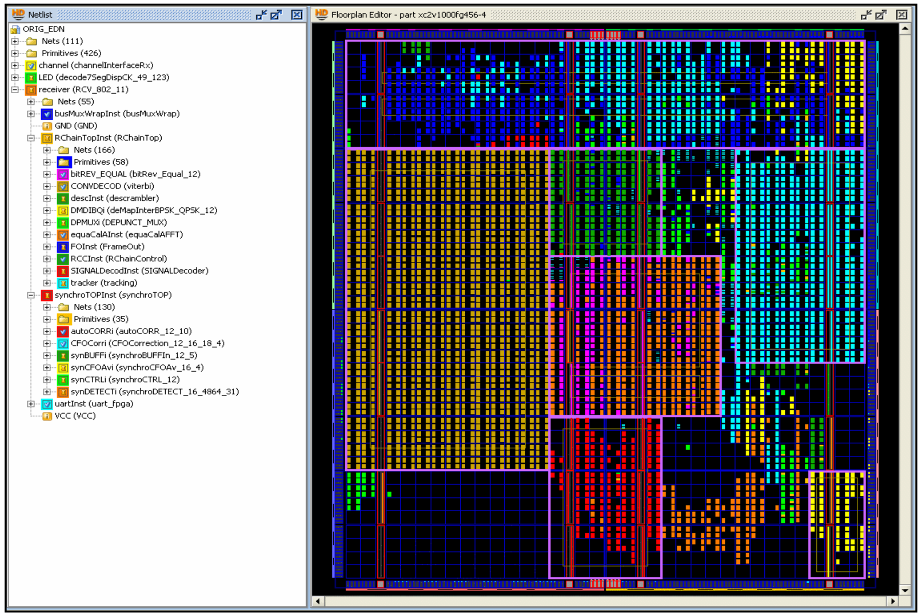
# Floorplanning and Layout

- The Floorplanner utility and PlanAhead software are used for design layout


Xilinx Floorplanner - correlate_and_accumulate.fnf
File Edit View Hierarchy Pattern Floorplan Window Help
correlate_and_accumulate.fnf D...
accumulate.fnf Editable Floorplan for xc2v250-5fg256
correlate_and_accumulate.fnf ...

correlate_and_accumulate" [11
correlate_and_accumulate "P
wr_clk_chd_inst "BUFGDLL"
clk_chc_inst "BUFGDLL"
clk_chb_inst "BUFGDLL"
clk_cha_inst "BUFGDLL"
fifo_inst "ch_fifo_3" [1 B
chc_fifo_inst "ch_fifo_2" [1 B
chb_fifo_inst "ch_fifo_1" [1 B
cha_fifo_inst "ch_fifo" [1 BLK
rd_clk_inst "BUFGDLL" [1 DC
data_control_inst "data_contr

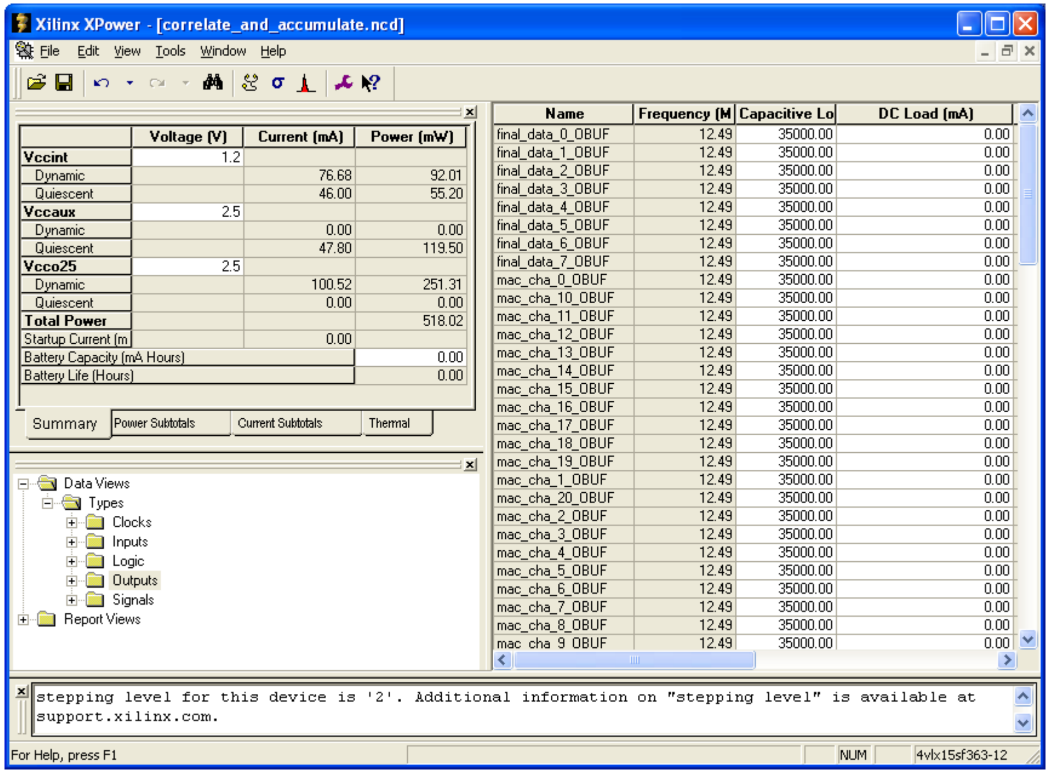
data_control_inst/mac_inst/mac_cha_inst/un1
data_control_inst/mac_inst/mac_cha_inst/un1
data_control_inst/mac_inst/mac_cha_inst/un1
data_control_inst/mac_inst/mac_chb_inst/un1
data_control_inst/mac_inst/mac_chb_inst/un1
data_control_inst/mac_inst/mac_chb_inst/un1
data_control_inst/mac_inst/mac_chc_inst/un1
data_control_inst/mac_inst/mac_chc_inst/un1
data_control_inst/mac_inst/mac_chc_inst/un1
data_control_inst/mac_inst/mac_chd_inst/un1
data_control_inst/mac_inst/mac_chd_inst/un1
data_control_inst/mac_inst/mac_chd_inst/un1
data_control_inst/mac_inst/mac_chc_inst/un1
data_control_inst/mac_inst/mac_chc_inst/un1
data_control_inst/mac_inst/mac_chc_inst/un1
data_control_inst/mac_inst/mac_chc_inst/un1
data_control_inst/mac_inst/mac_chc_inst/un1
data_control_inst/mac_inst/mac_chc_inst/un1
data_control_inst/mac_inst/mac_chc_inst/un1
data_control_inst/mac_inst/mac_chc_inst/un1



ORIG_EDN
Nets (111)
Primitives (426)
channel (channelInterfaceRx)
LED (decode7SegDispCK_49_123)
receiver (RCV_802_11)
Nets (55)
busMuxWrapInst (busMuxWrap)
GND (GND)
RChainTopInst (RChainTop)
Nets (166)
Primitives (58)
bitREV_EQUAL (bitRev_Equal_12)
CONVDECOD (viterbi)
descInst (descrambler)
DMDIBQi (deMapInterBPSK_QPSK_12)
DPMUXi (DEPUNCT_MUX)
equaCalAInst (equaCalAFFT)
FOInst (FrameOut)
RCCInst (RChainControl)
SIGNALDecodInst (SIGNALDecoder)
tracker (tracking)
synchroTOPInst (synchroTOP)
Nets (130)
Primitives (35)
autoCORRi (autoCORR_12_10)
CFOCorri (CFOCorrection_12_16_18_4)
synBUFFi (synchroBUFFIn_12_5)
synCFOAvi (synchroCFOAv_16_4)
synCTRLi (synchroCTRL_12)
synDETECTi (synchroDETECT_16_4864_31)
uartInst (uart_fpga)
VCC (VCC)

# Xilinx XPower

- XPower is used to estimate the power consumption and junction temperature of your FPGA
  - Reads an implemented design (NCD file) and timing constraint data
  - You supply activity rates, clock frequencies, capacitive loading on output pins, power supply data, and ambient temperature
    - You can also supply design activity data from simulation (VCD file)



| | Voltage (V) | Current (mA) | Power (mW) |
|---|---|---|---|
| **Vccint** | 1.2 | | |
| Dynamic | | 76.68 | 92.01 |
| Quiescent | | 46.00 | 55.20 |
| **Vccaux** | 2.5 | | |
| Dynamic | | 0.00 | 0.00 |
| Quiescent | | 47.80 | 119.50 |
| **Vcco25** | 2.5 | | |
| Dynamic | | 100.52 | 251.31 |
| Quiescent | | 0.00 | 0.00 |
| **Total Power** | | | 518.02 |
| Startup Current (m | | 0.00 | |
| Battery Capacity (mA Hours) | | | 0.00 |
| Battery Life (Hours) | | | 0.00 |

| Name | Frequency (M | Capacitive Lo | DC Load (mA) |
|---|---|---|---|
| final_data_0_OBUF | 12.49 | 35000.00 | 0.00 |
| final_data_1_OBUF | 12.49 | 35000.00 | 0.00 |
| final_data_2_OBUF | 12.49 | 35000.00 | 0.00 |
| final_data_3_OBUF | 12.49 | 35000.00 | 0.00 |
| final_data_4_OBUF | 12.49 | 35000.00 | 0.00 |
| final_data_5_OBUF | 12.49 | 35000.00 | 0.00 |
| final_data_6_OBUF | 12.49 | 35000.00 | 0.00 |
| final_data_7_OBUF | 12.49 | 35000.00 | 0.00 |
| mac_cha_0_OBUF | 12.49 | 35000.00 | 0.00 |
| mac_cha_10_OBUF | 12.49 | 35000.00 | 0.00 |
| mac_cha_11_OBUF | 12.49 | 35000.00 | 0.00 |
| mac_cha_12_OBUF | 12.49 | 35000.00 | 0.00 |
| mac_cha_13_OBUF | 12.49 | 35000.00 | 0.00 |
| mac_cha_14_OBUF | 12.49 | 35000.00 | 0.00 |
| mac_cha_15_OBUF | 12.49 | 35000.00 | 0.00 |
| mac_cha_16_OBUF | 12.49 | 35000.00 | 0.00 |
| mac_cha_17_OBUF | 12.49 | 35000.00 | 0.00 |
| mac_cha_18_OBUF | 12.49 | 35000.00 | 0.00 |
| mac_cha_19_OBUF | 12.49 | 35000.00 | 0.00 |
| mac_cha_1_OBUF | 12.49 | 35000.00 | 0.00 |
| mac_cha_20_OBUF | 12.49 | 35000.00 | 0.00 |
| mac_cha_2_OBUF | 12.49 | 35000.00 | 0.00 |
| mac_cha_3_OBUF | 12.49 | 35000.00 | 0.00 |
| mac_cha_4_OBUF | 12.49 | 35000.00 | 0.00 |
| mac_cha_5_OBUF | 12.49 | 35000.00 | 0.00 |
| mac_cha_6_OBUF | 12.49 | 35000.00 | 0.00 |
| mac_cha_7_OBUF | 12.49 | 35000.00 | 0.00 |
| mac_cha_8_OBUF | 12.49 | 35000.00 | 0.00 |
| mac_cha_9_OBUF | 12.49 | 35000.00 | 0.00 |

```
stepping level for this device is '2'. Additional information on "stepping level" is available at
support.xilinx.com.
```

For Help, press F1 NUM 4vlx15sf363-12

# Summary

- FPGAs provide for reduced design time and later bug fixes
  - No design for test logic is required
  - Deep sub-micron verification has been completed
  - No waiting for prototypes
    - Firmware development starts sooner in the design cycle for the FPGA design flow
    - Faster production ramp up
- Xilinx provides powerful tools for advanced control over implementation
- The Xilinx ChipScope Pro software tool provides powerful in-circuit verification