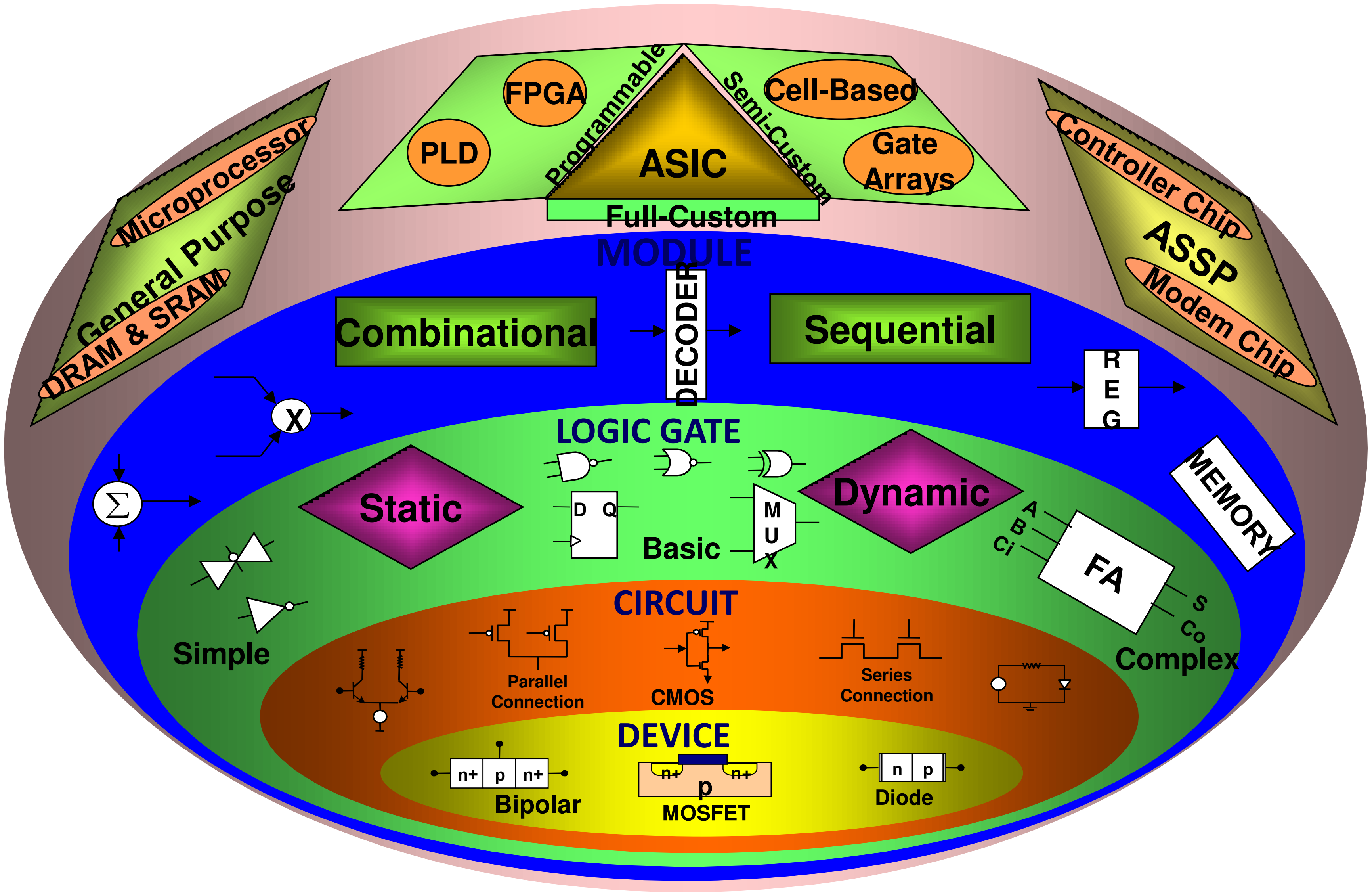
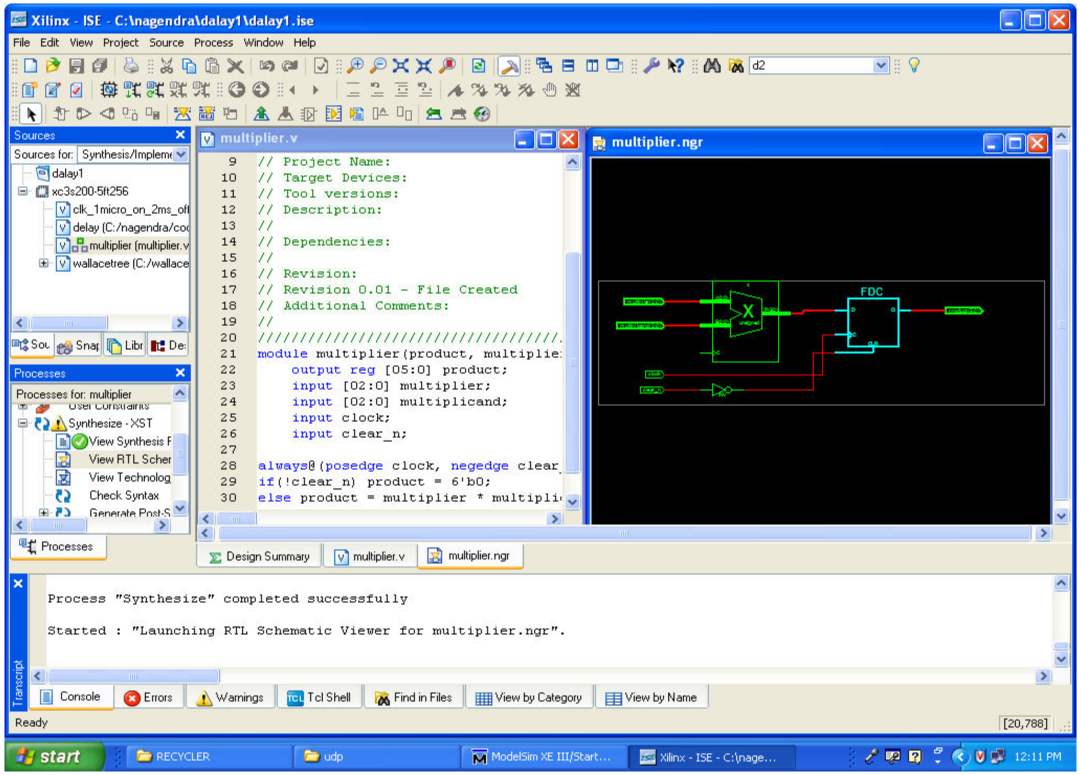
VLSI Design Flow

Digital design



FPGA Design

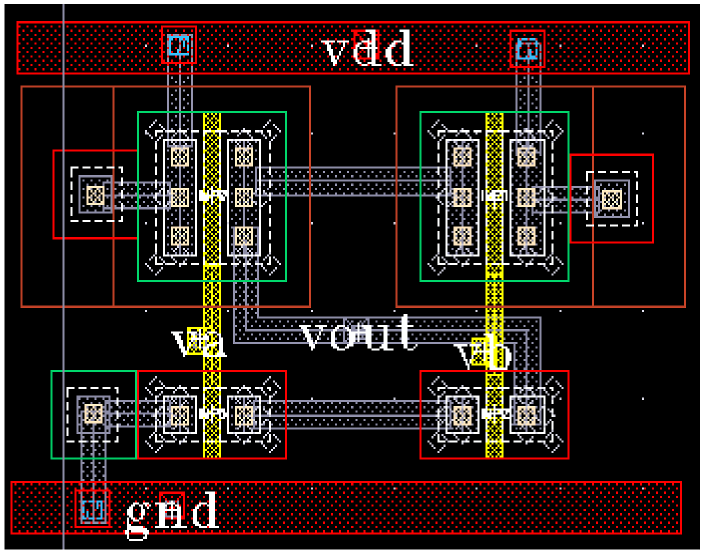


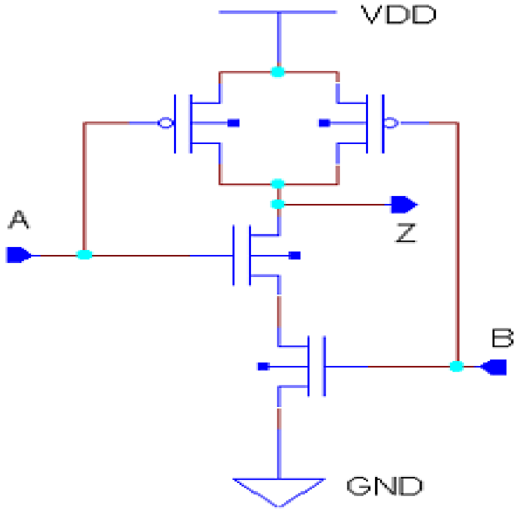
```
// Project Name:
// Target Devices:
// Tool versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
/////////////////////////////////////.
module multiplier(product, multiplie
     output reg [05:0] product;
     input [02:0] multiplier;
     input [02:0] multiplicand;
     input clock;
     input clear_n;

always@(posedge clock, negedge clear_
if(!clear_n) product = 6'b0;
else product = multiplier * multipli
```

```
Process "Synthesize" completed successfully

Started : "Launching RTL Schematic Viewer for multiplier.ngr".
```

- Fastest turn around time
- Cheapest at lower volumes
- Limited by resources
- Power, Timing Performance is compromised

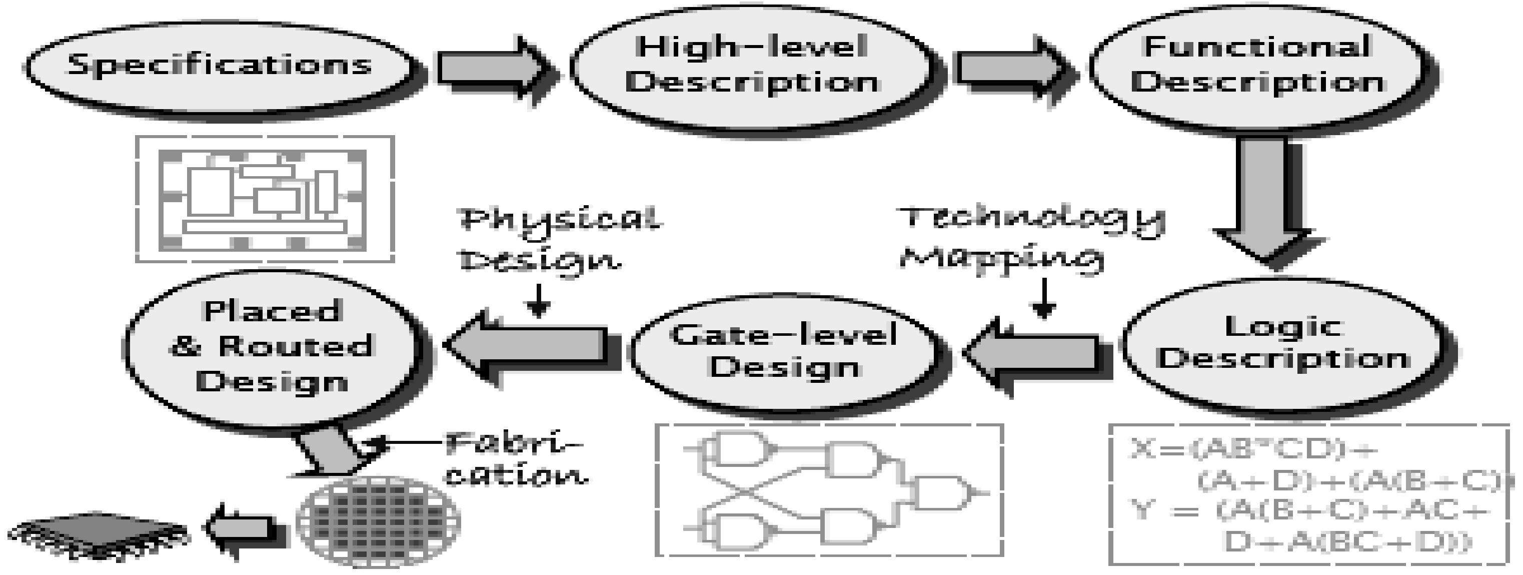
CMOS Design Flow





- Huge Time To Market
- Labour Intensive
- The most efficient of all flows

ASIC Design Flow

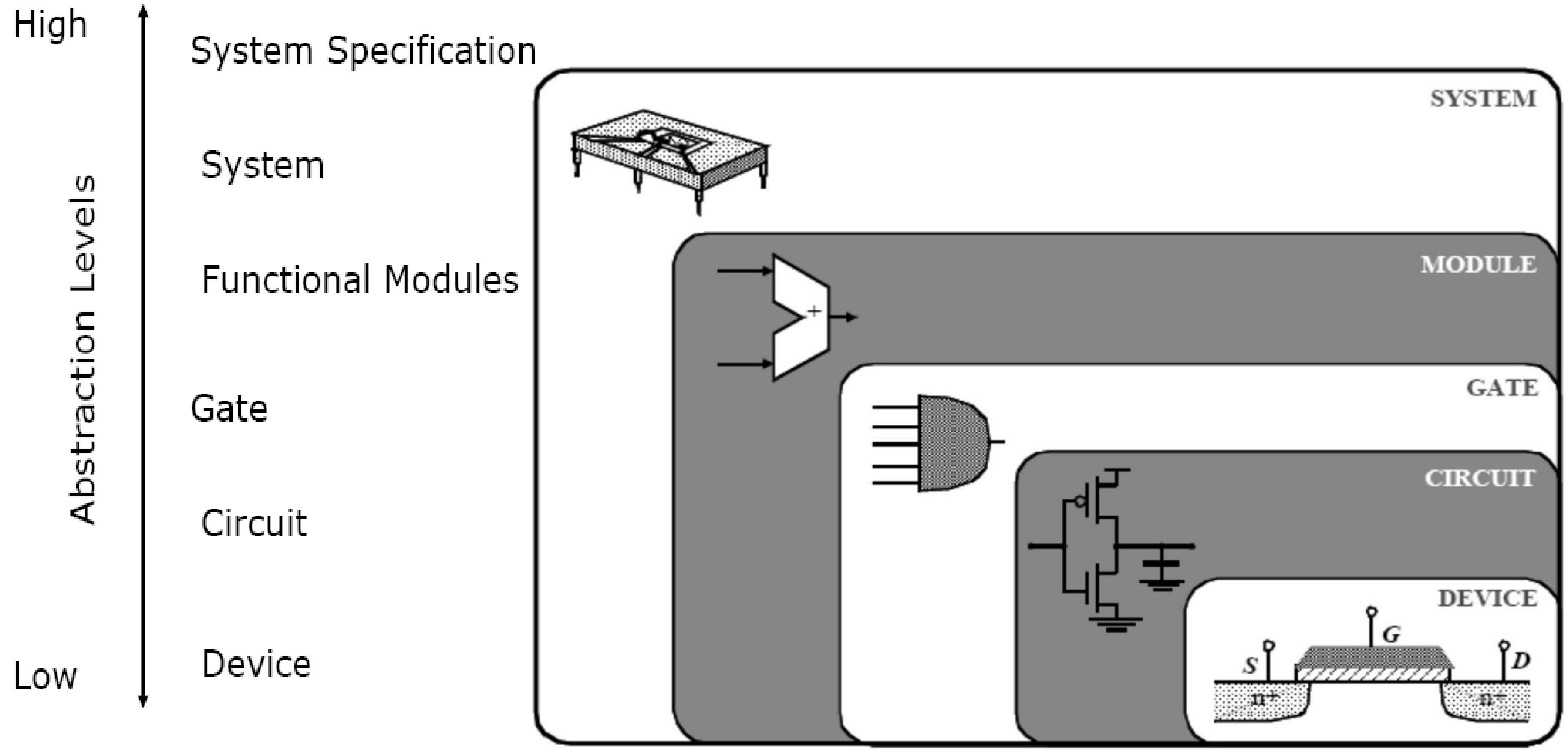


- Faster Time To Market
- Fairly optimized for Area, Power and Timing

Advantages of Semi-Custom

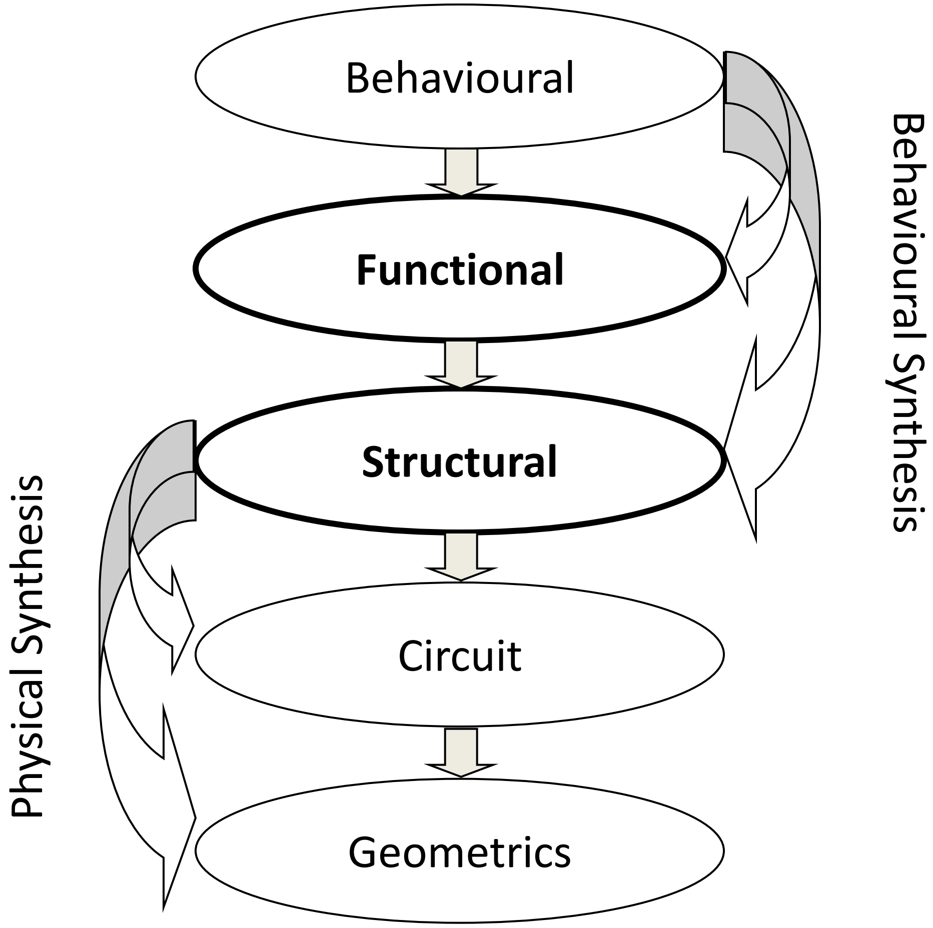
- Faster time to market than full-custom Design
- Better performance than FPGAs
- Mature tools available for all stages
- Cheaper in the long run
- Better debug and re-use

Abstraction Levels

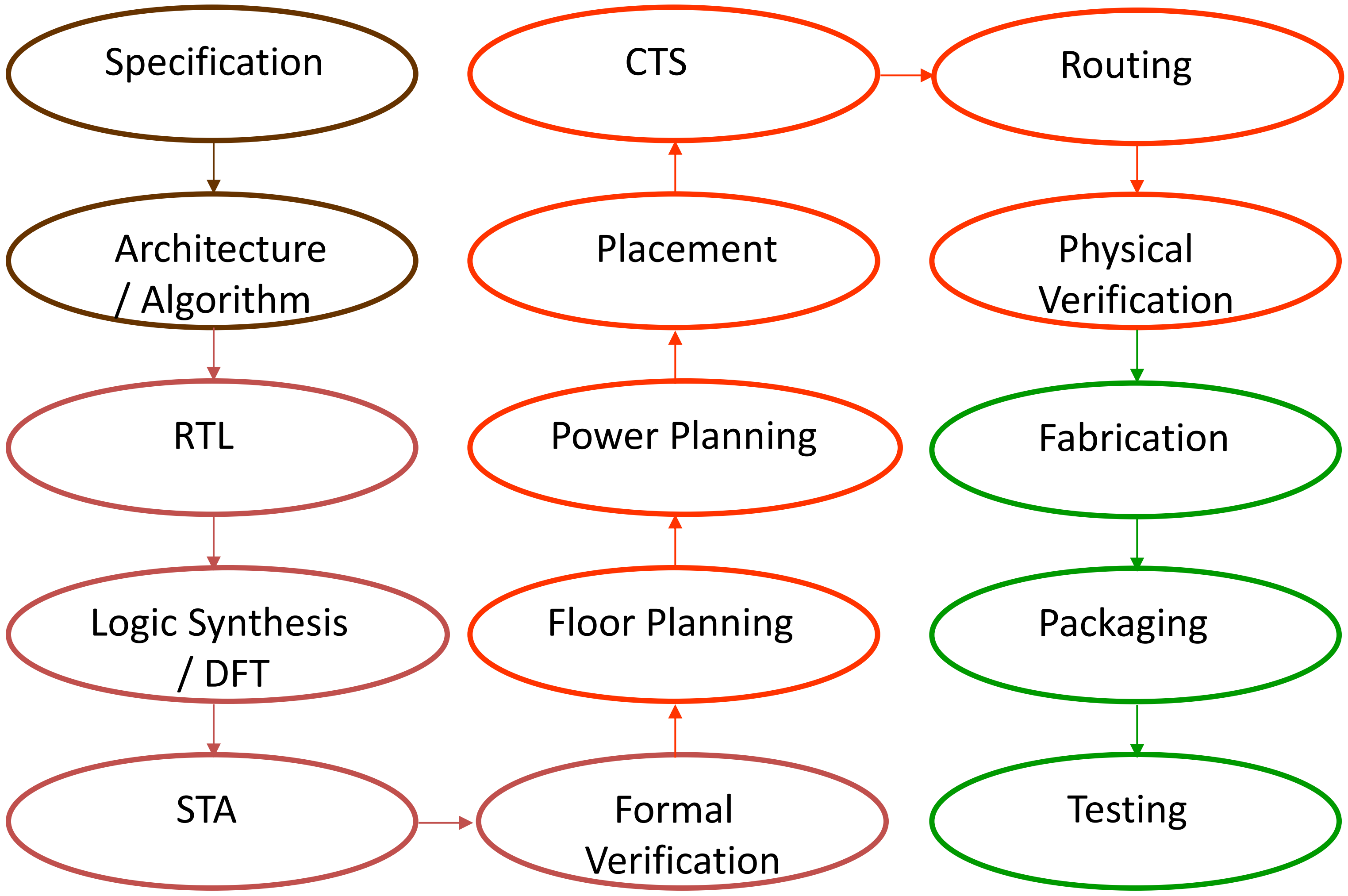


Synthesis

- Transformation of design from higher level of abstraction to lower level of abstraction



Semi-Custom ASIC Design Flow



Types of ASICs

❑**Full-Custom ASICs**

❖ Include some (possibly all) customized logic cells

❖ Have all their mask layers customized

❖ Full-custom ASIC design makes sense only

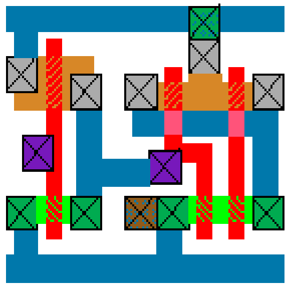
✓ *When no suitable existing libraries exist or*

✓ *Existing library cells are not fast enough or*

✓ *The available pre-designed/pre-tested cells consume too much power that a design can allow or*

✓ *The available logic cells are not compact enough to fit or*

✓ *ASIC technology is new or/and so special that no cell library exits.*



❖ Offer highest performance and lowest cost (smallest die size) but at the expense of increased design time, complexity, higher design cost and higher risk.

❖ **Some Examples:** High-Voltage Automobile Control Chips, Analog-Digi Communication Chips, Sensors and Actuators

ASIC Design Process

S-1 Design Entry: Schematic entry or HDL description

S-2: Logic Synthesis: Using Verilog HDL or VHDL and Synthesis tool, produce ***a netlist***-logic cells and their interconnect detail

S-3 System Partitioning: Divide a large system into ASIC sized pieces

S-4 Pre-Layout Simulation: Check design functionality

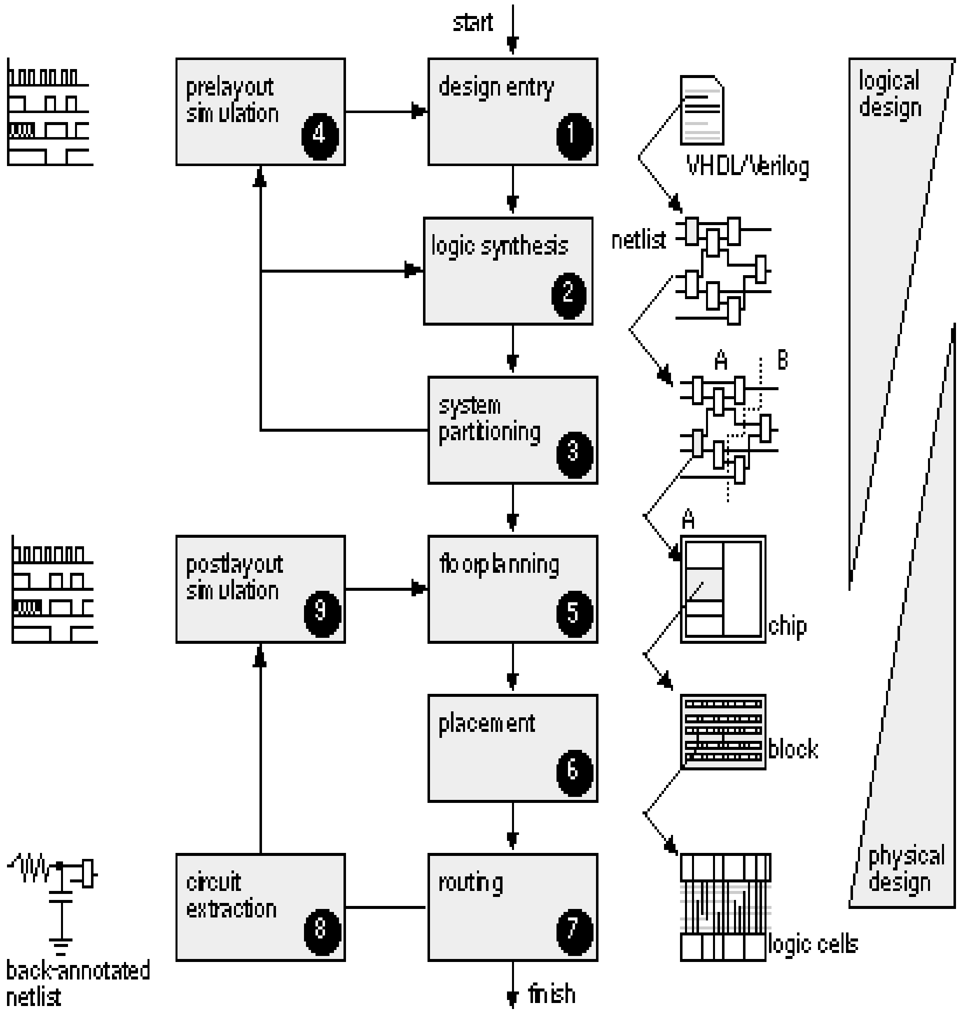
S-5 Floorplanning: Arrange netlist blocks on the chip

S-6 Placement: Fix cell locations in a block

S-7 Routing: Make the cell and block interconnections

S-8 Extraction: Measure the interconnect R/C cost

S-9 Post-Layout Simulation



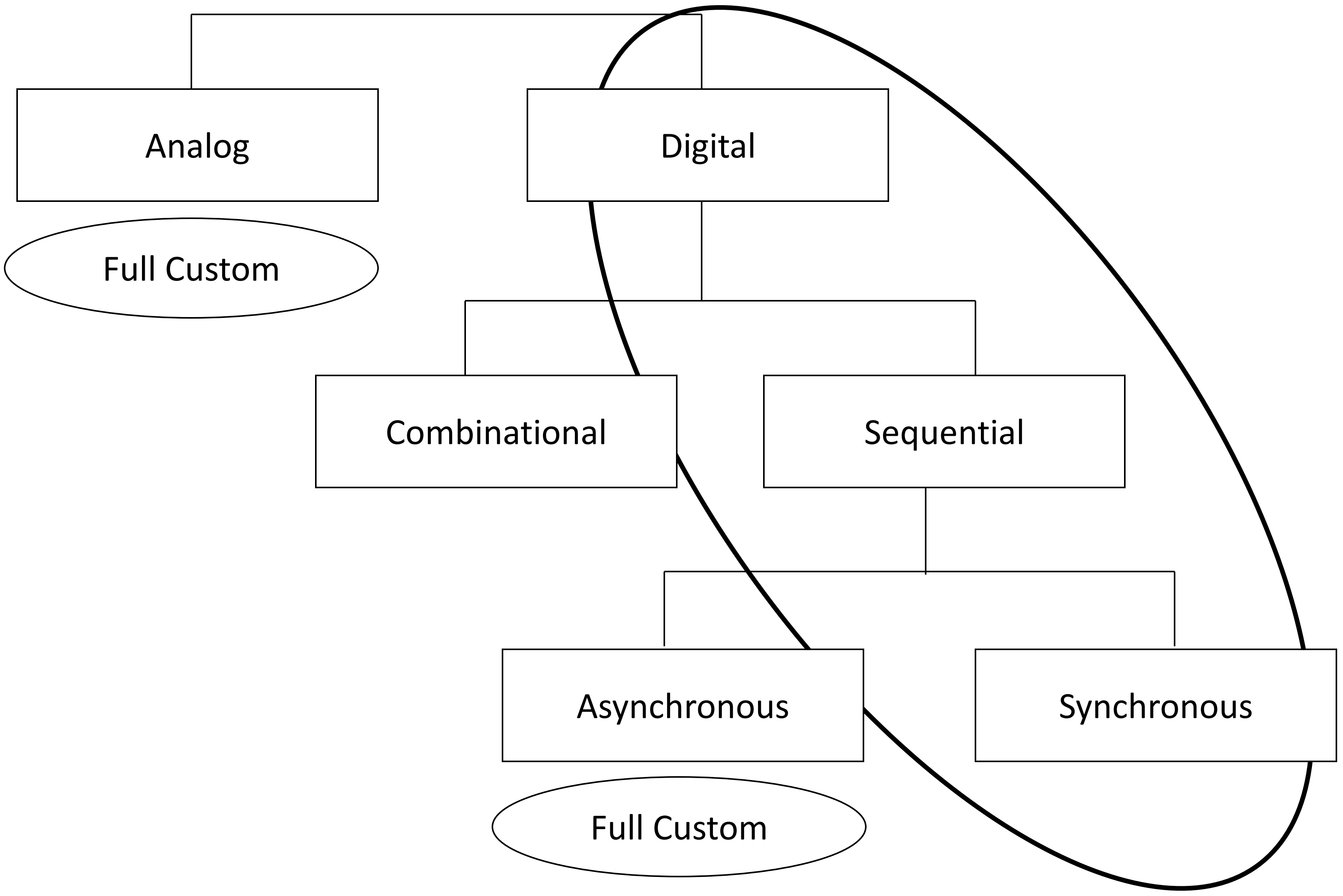
Applications

- FPGA :
  - Research
  - Academics
  - Defense
  - Prototyping
  - Start Ups
  - IP Core developers

- Full Custom :
  - Memory
  - I/O
  - Analog Blocks
  - Processors

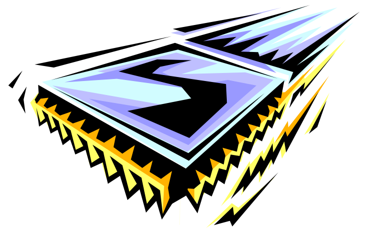
- Semi Custom :
  - Wireless Applications
  - Modulators
  - Decoders
  - DSP Block Sets

Designs


Analog
Digital
Full Custom
Combinational
Sequential
Asynchronous
Synchronous
Full Custom

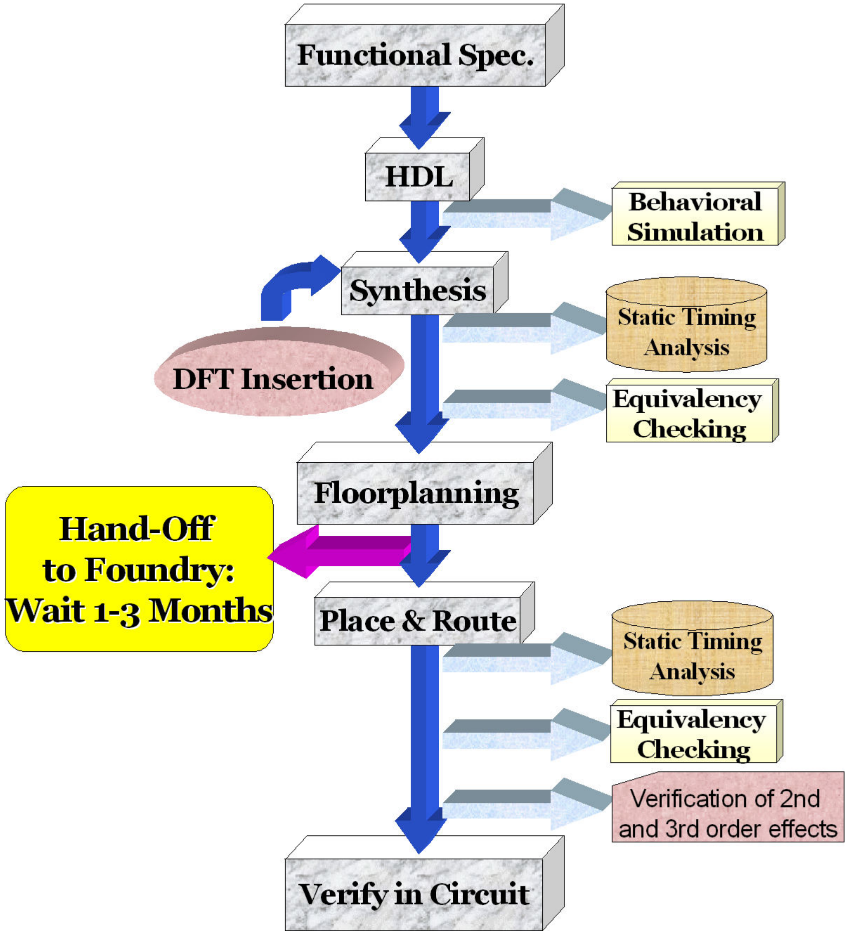
Design Flow

- ASIC and FPGA design and implementation methodologies differ moderately
  - Xilinx FPGAs provide for reduced design time and later bug fixes
    - No design for test is required, deep sub-micron verification has been completed, and no waiting for prototypes
- Coding style
  - For high-performance designs, FPGAs require more pipelining
  - When re-targeting code from an ASIC to an FPGA, the code usually requires optimization



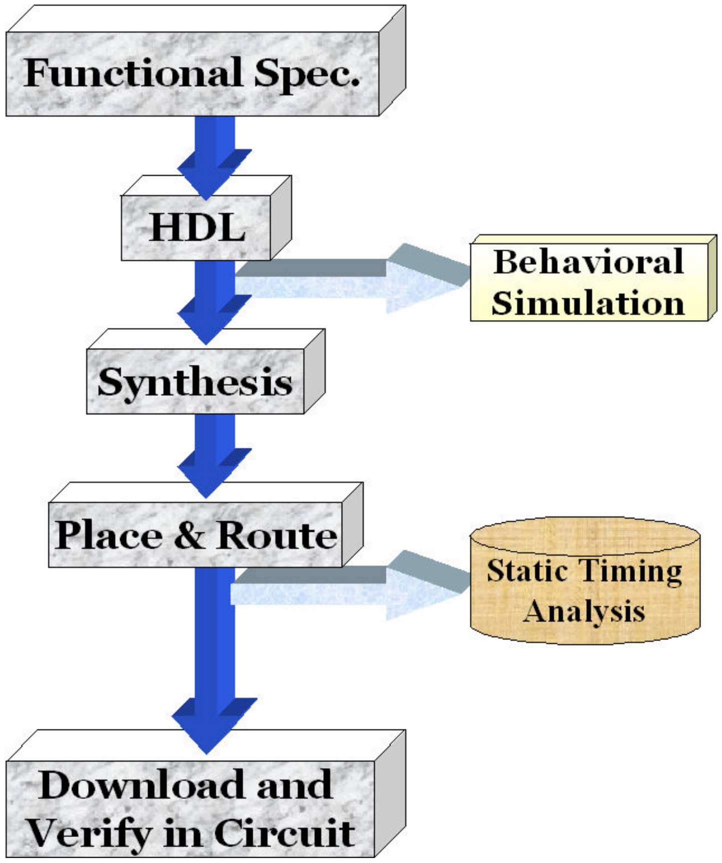
ASIC Design Flow

- ASIC tools are generally driven by scripts
- Post-synthesis static timing analysis and equivalency checking are musts for sign-off to foundry
- Verification of deep sub-micron effects (second- and third-order effects) is required
  - Internal, deep sub-micron effects are already verified for Xilinx FPGAs

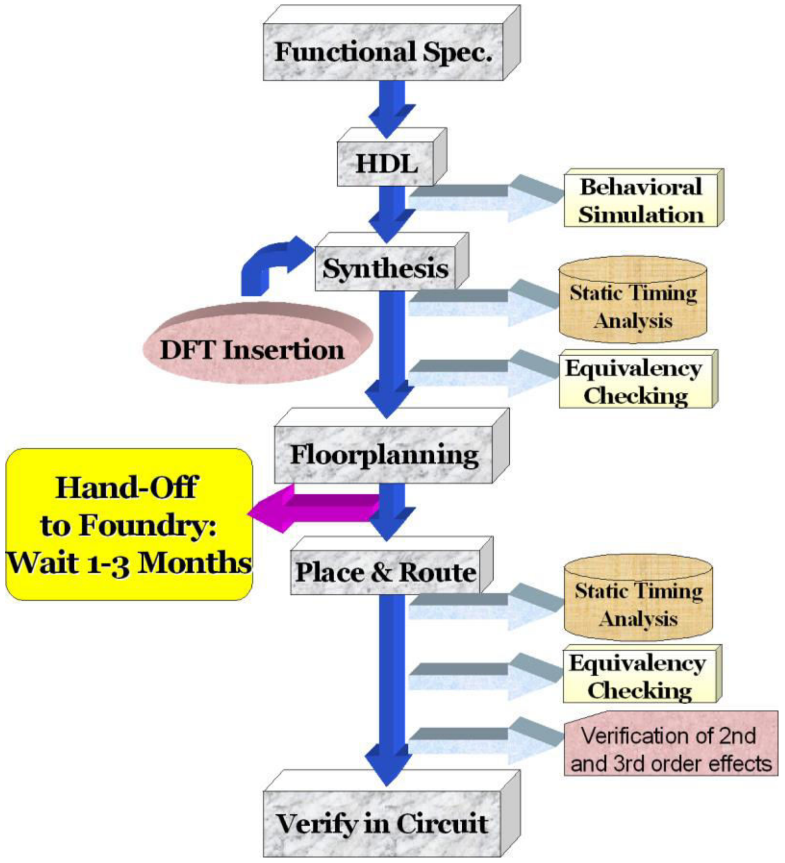


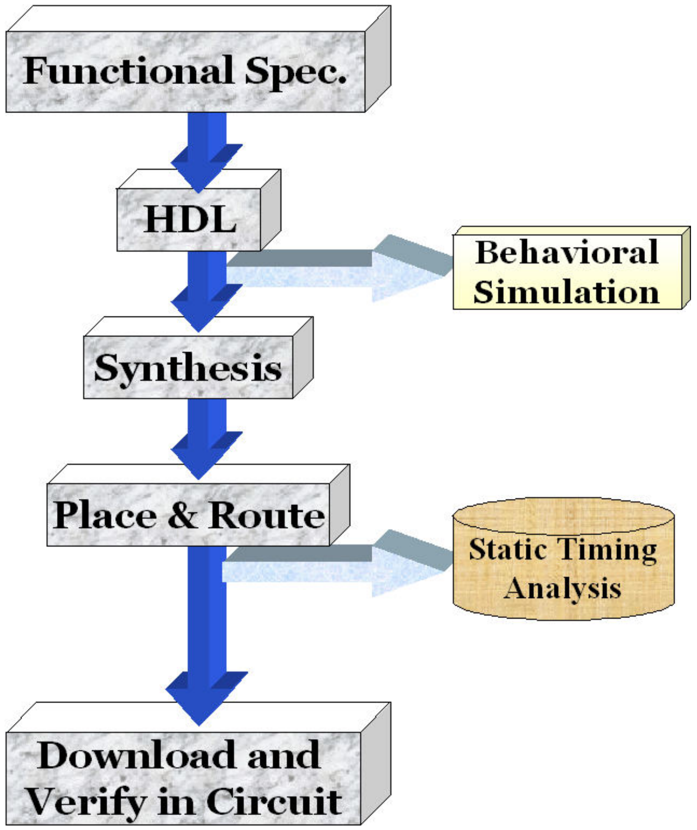
FPGA Design Flow

- FPGA tools are generally GUI-driven, push-button flows
  - FPGA tools also have scripting capabilities
- After the design passes behavioral simulation and static timing analysis, verification is completed most efficiently by verifying in circuit
  - Fast turn-around times
  - Static timing analysis is used to verify timing of the design



ASIC Versus FPGA Design Flow





Primary Design Flow Differences

HDL	FPGA code may need optimization for high performance
Post-Synthesis Static Timing Analysis	ASIC only; not required for FPGAs
Design for Test Insertion	ASIC only; not required for FPGAs
In-Circuit Verification and Waiting for Prototypes	No waiting time for FPGAs; ASICs wait 1–3 months for first prototype
Deep Sub-Micron Verification	ASIC only; already completed for FPGAs

Two competing implementation

ASIC
Application **S**pecific
Integrated **C**ircuit

- Designed all the way from behavioral description to physical layout
- Designs must be sent for expensive and time consuming fabrication in semiconductor foundry

FPGA
Field **P**rogrammable
Gate **A**rray

- No physical layout design; design ends with a bitstream used to configure a device
- Bought off the shelf and reconfigured by designers themselves

Which Way to Go?

ASICs

- High performance
- Low power
- Low cost in high volumes

FPGAs

- Off-the-shelf
- Low development cost
- Short time to market
- Reconfigurability

IC Products

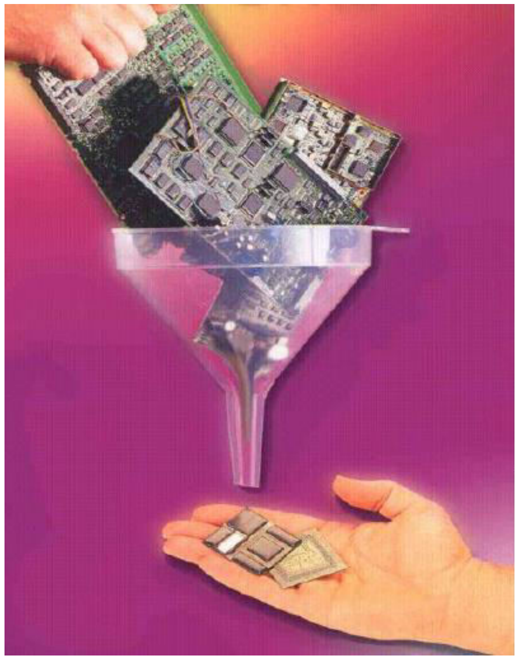
- Processors
  - CPU, DSP, Controllers
- Memory chips
  - RAM, ROM, EEPROM
- Analog
  - Mobile communication, audio/video processing
- Programmable
  - PLA, FPGA
- Embedded systems
  - Used in cars, factories
  - Network cards
- System-on-chip (SoC)

System On chip

- System
  - A collection of all kinds of **components** and/or **subsystems** that are appropriately **interconnected** to perform the specified functions for end users.
- An SoC design is a "**product creation process**" which
  - Starts at identifying the **end-user needs**
  - Ends at delivering a product with enough **functional satisfaction** to overcome the payment from the end-user

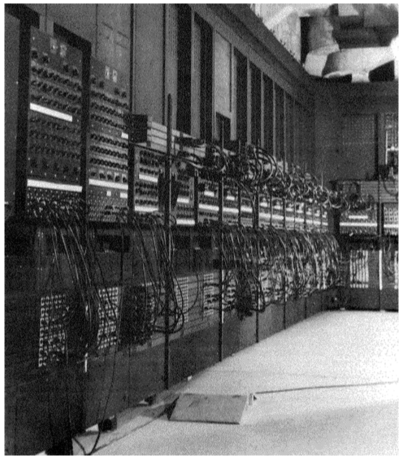
System-on-Chip Contains..

- An SoC contains:
  - Portable / reusable IP
  - Embedded CPU
  - Embedded Memory
  - Real World Interfaces (USB, PCI, Ethernet)
  - Software (both on-chip and off-chip)
  - Mixed-signal Blocks
  - Programmable HW (FPGAs)
  - > 500K gates
- Not an ASIC !



System Evolution

System complexity and density

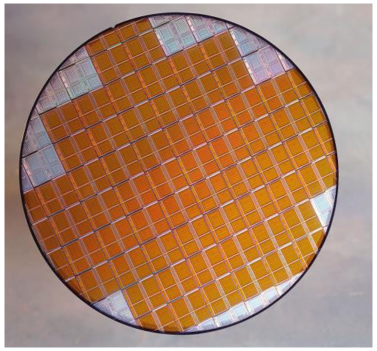


Components of the system are large.

“In-room” interconnect

All components reside in relatively small box.

On-board interconnect



Components of the system reside on single chip.

On-chip interconnect

We are here

Time

Old systems

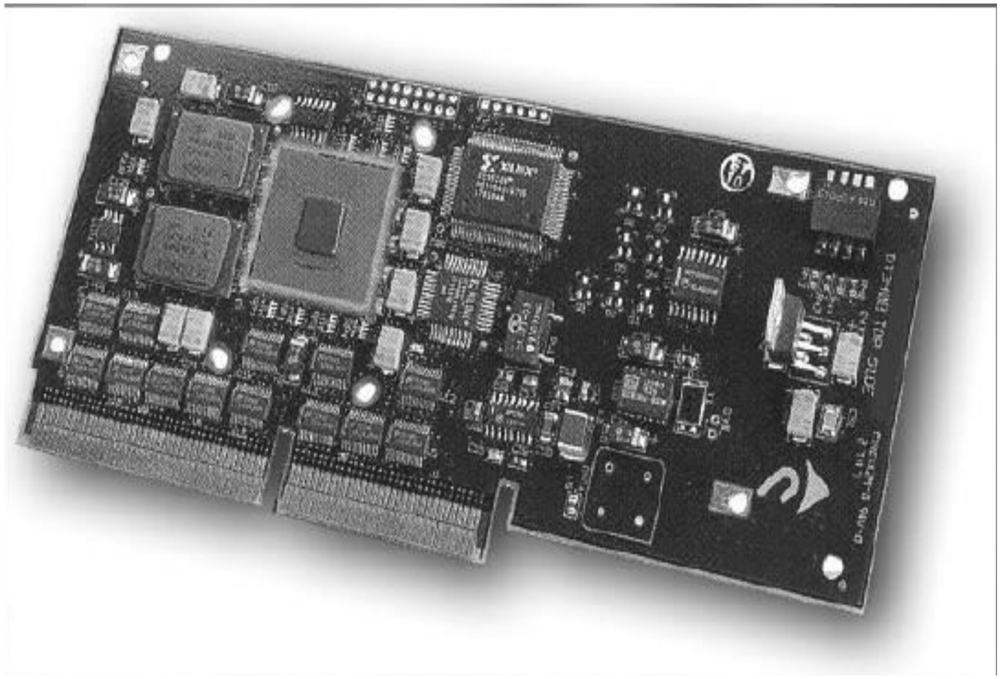
Recent systems

Modern systems

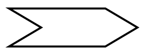
System on a Programmable Chip …

- On the other hand, FPGA-type solutions are also evolving
  - On-chip processor cores
  - Multi-million gate capacity
  - FPGA-based SoC -type platforms thus have a growing niche
- System-on-a-Programmable Chip (SOPC) term coined by Synopsys
- Platform FPGA, Programmable System-on-Chip (PSOC) or System-on-Reconfigurable-Chip (SORC) are alternative names for these solutions

Paradigm Shift in SoC Design

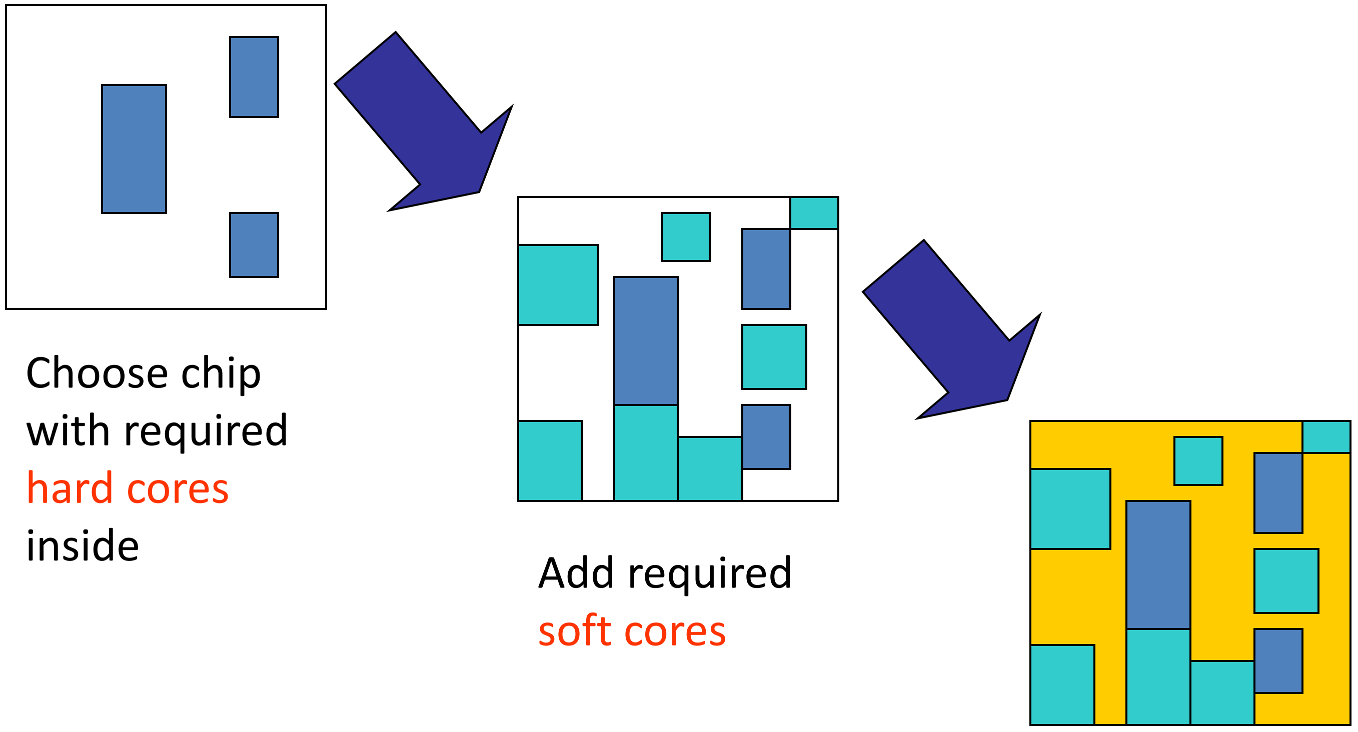


System on a board



System on a chip

SoC design process

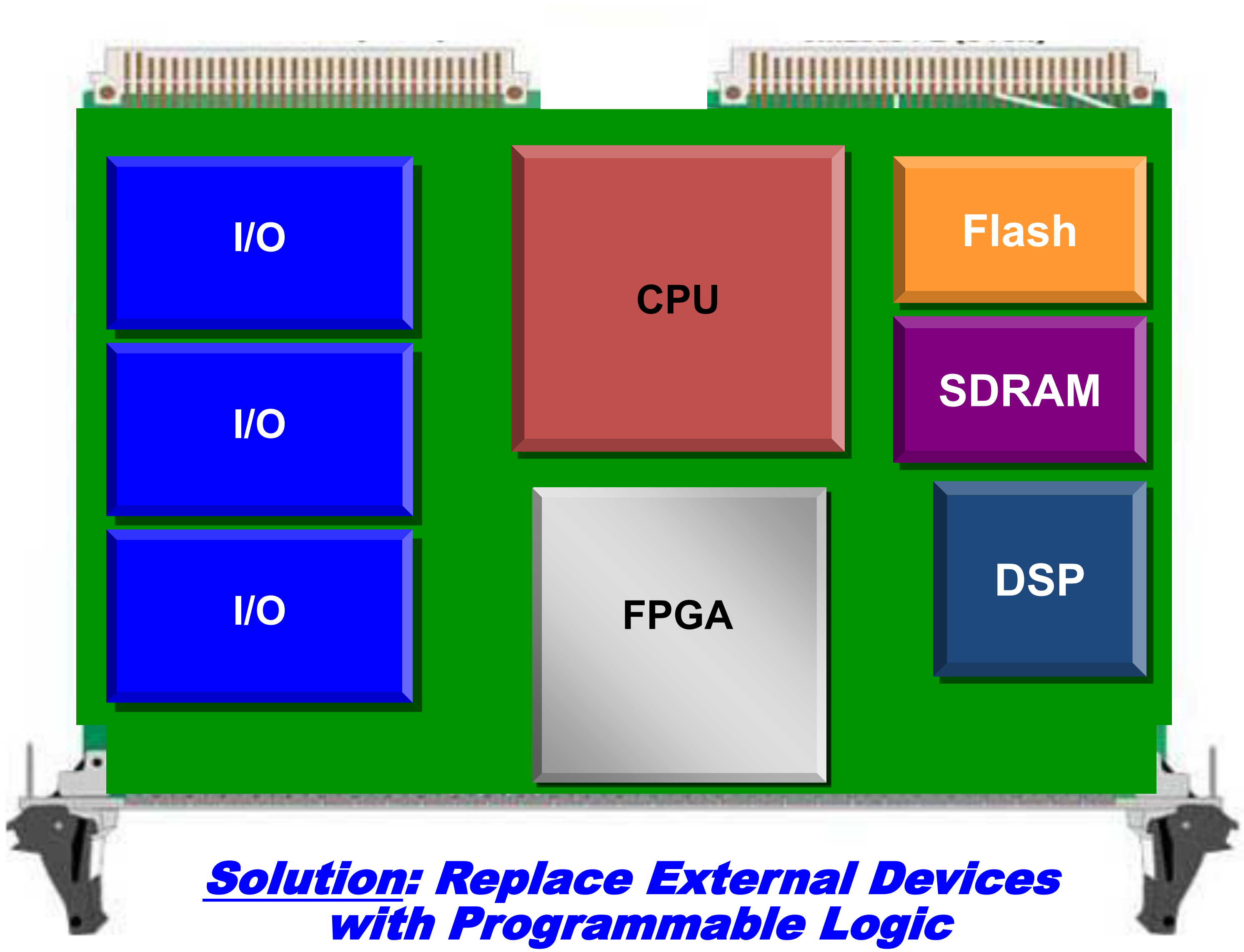


Choose chip with required hard cores inside

Add required soft cores

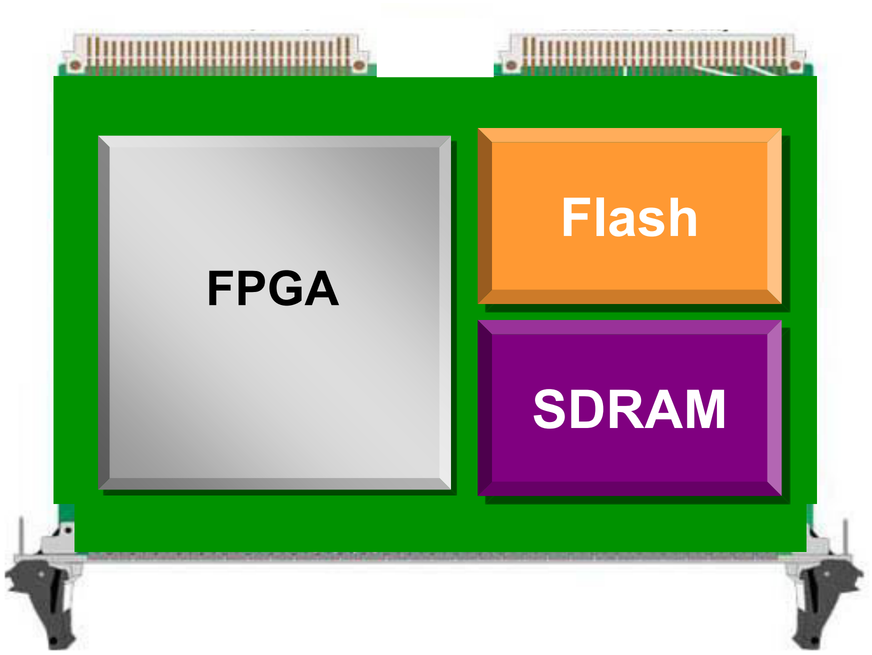
Add user logic

Problem: Reduce Cost, Complexity & Power



Solution: Replace External Devices with Programmable Logic

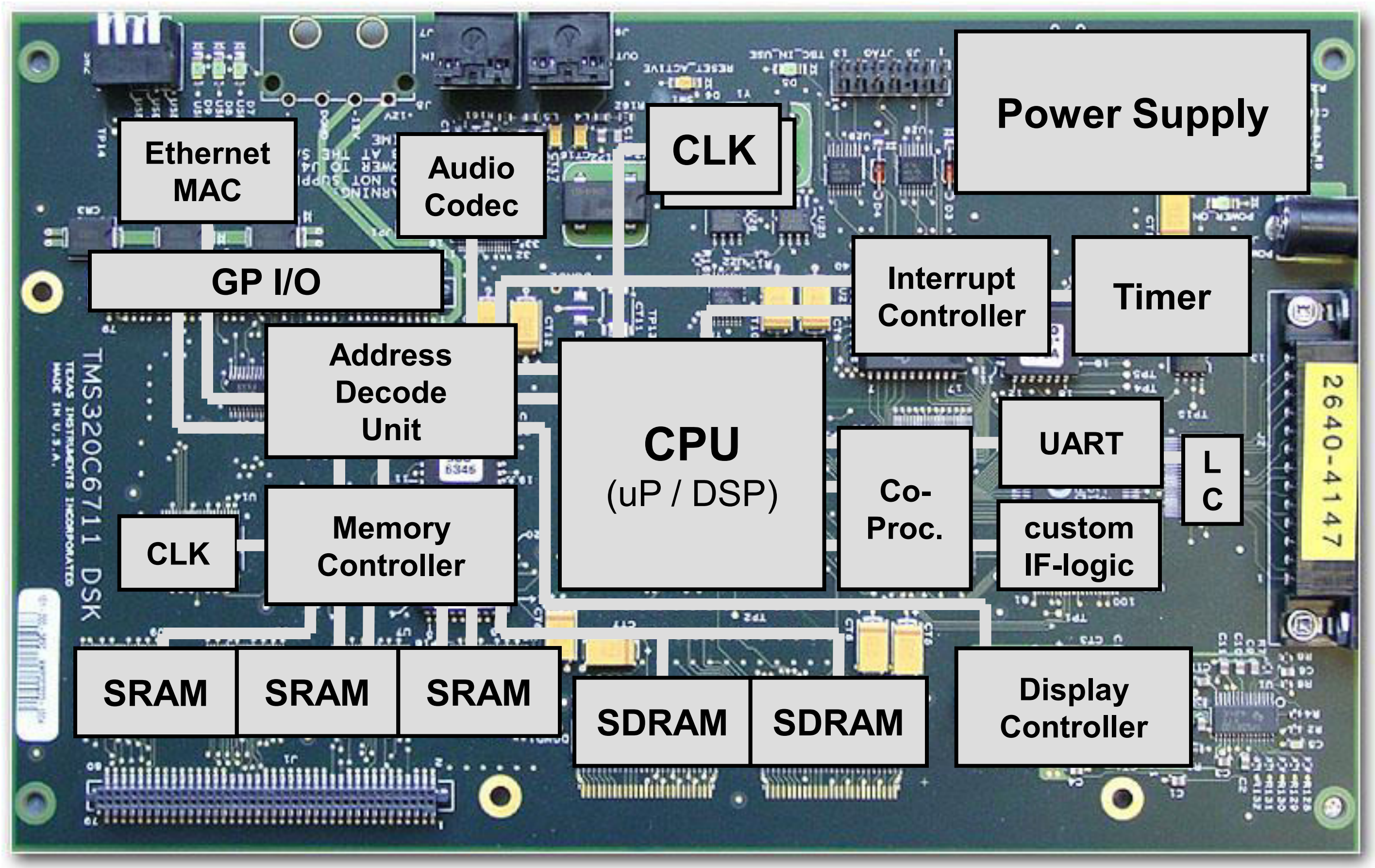
System On A Programmable Chip (SOPC)



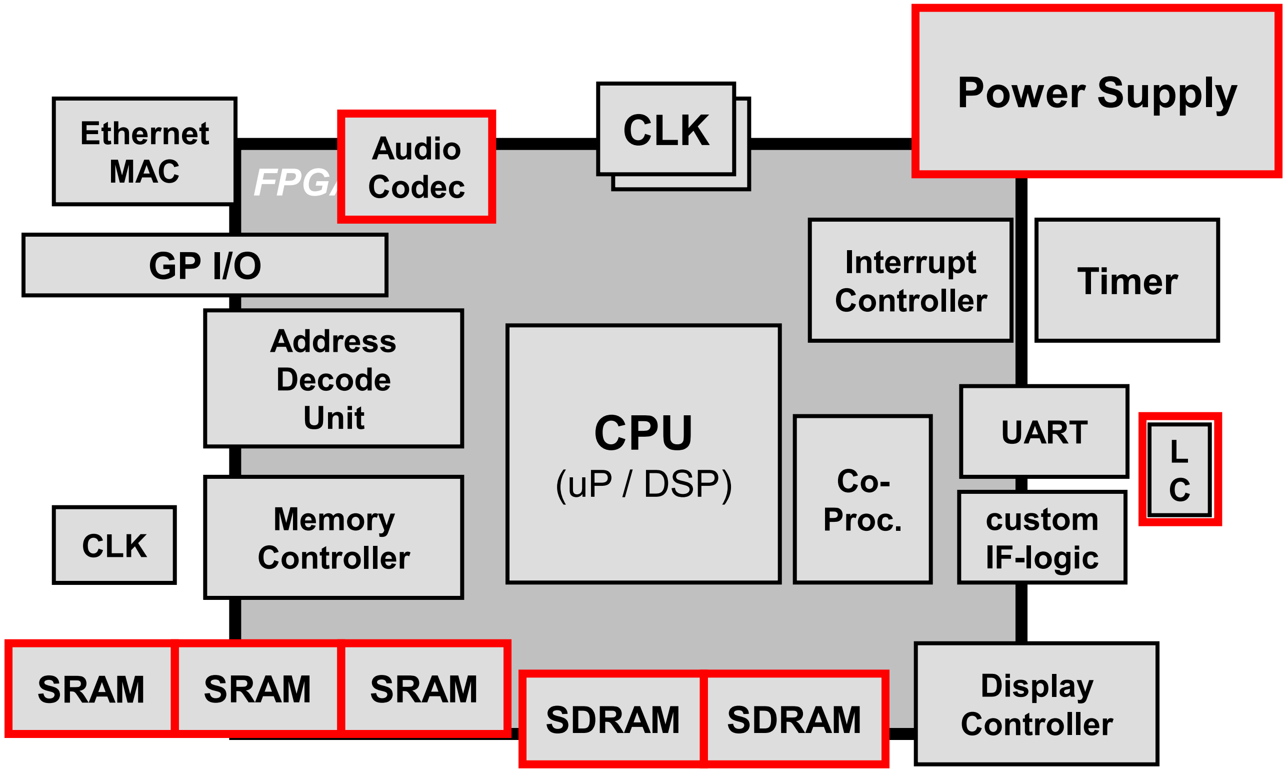
CPU is a Critical Control Function Required for System-Level Integration

Traditional Embedded System

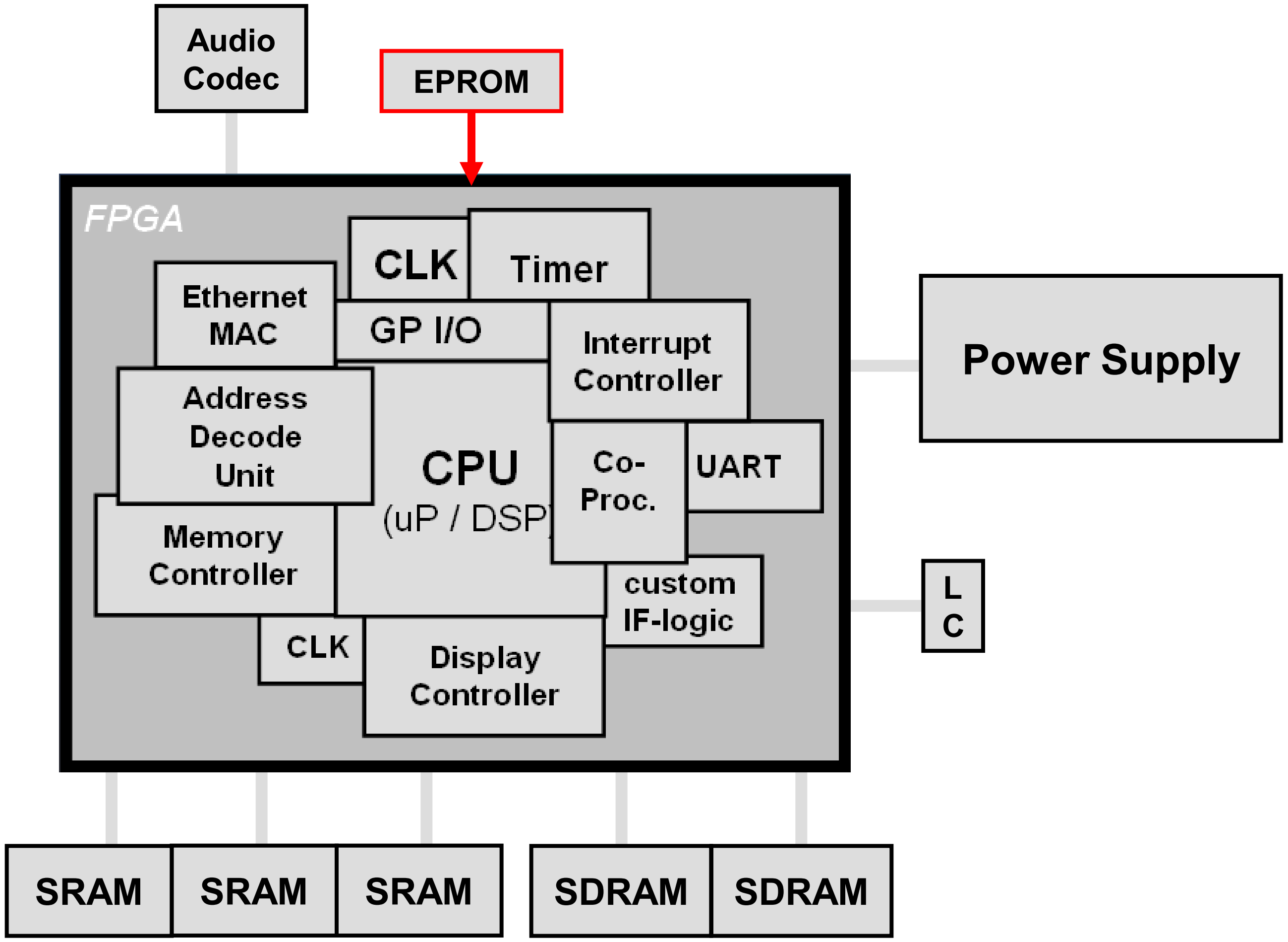
Outline



Next Step...


Ethernet MAC
FPGA
Audio Codec
CLK
Power Supply
GP I/O
Interrupt Controller
Timer
Address Decode Unit
CPU (uP / DSP)
Co-Proc.
UART
L C
CLK
Memory Controller
custom IF-logic
SRAM
SRAM
SRAM
SDRAM
SDRAM
Display Controller

Configurable System on a Chip (CSoC)



Reconfigurable System on a Chip (SoC)

How to build such a system ?

. Overview:

→ *Xilinx EDK / MicroBlaze Soft CPU core*

→ *Design- / Tool-Flow*

2. Demonstration:

→ *Create a simple system*

→*Implement the system on a Xilinx Virtex 7/Kintex 7 FPGAs*