香港中文大學
The Chinese University of Hong Kong

CENG3430 Rapid Prototyping of Digital Systems

Lecture 02: Introduction to ZedBoard

Ming-Chang YANG

mcyang@cse.cuhk.edu.hk

Outline

- Digital System Design Basics
  - Integrated Circuit Technology
  - Design Flow of Digital Systems
  - System-on-Chip (SoC)
  - Application Specific Integrated Circuit (ASIC)

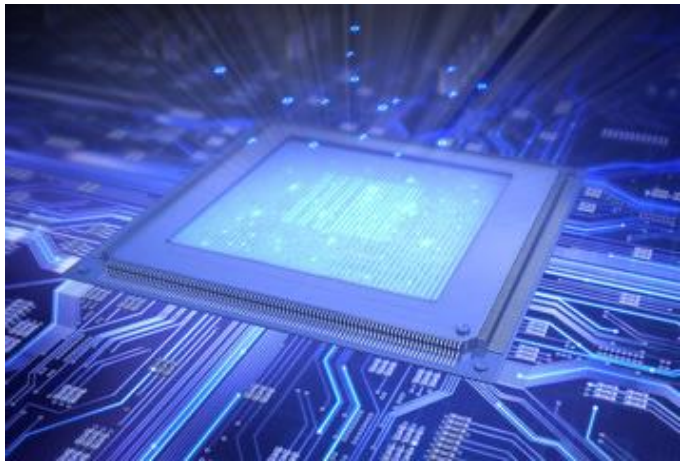
- Zynq: All-Programmable SoC (APSoC)
  - Our Board: ZedBoard
    - ZedBoard Layout and Interfaces
    - Specifying ZedBoard in Vivado
    - Hardware Setup for ZedBoard
    - Programming the ZedBoard
    - Xilinx Design Constraints (XDC) File

Integrated Circuit Technology

- Integrated circuit (IC) technology has improved to allow more and more components on a chip.
  - Small Scale Integration (SSI): 1 to 20 gates
  - Medium Scale Integration (MSI): 20 to 200 gates
  - Large Scale Integration (LSI): 200 to few thousands gates
  - Very Large Scale Integration (VLSI): More than 10,000 gates
  - Ultra Large Scale Integration (ULSI): 100 million transistors

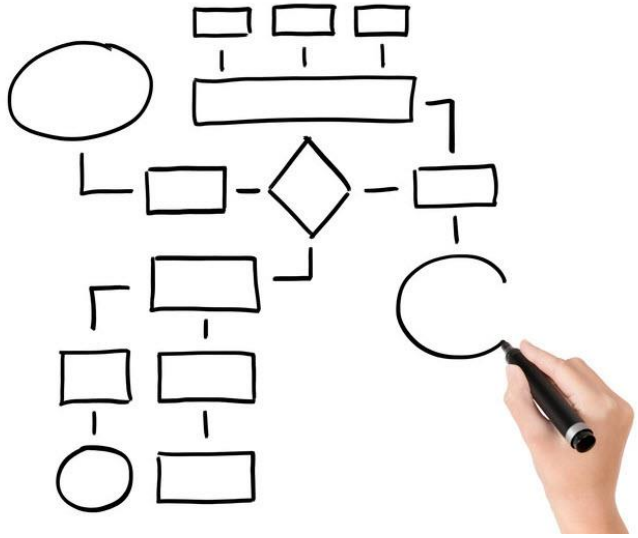
Gates

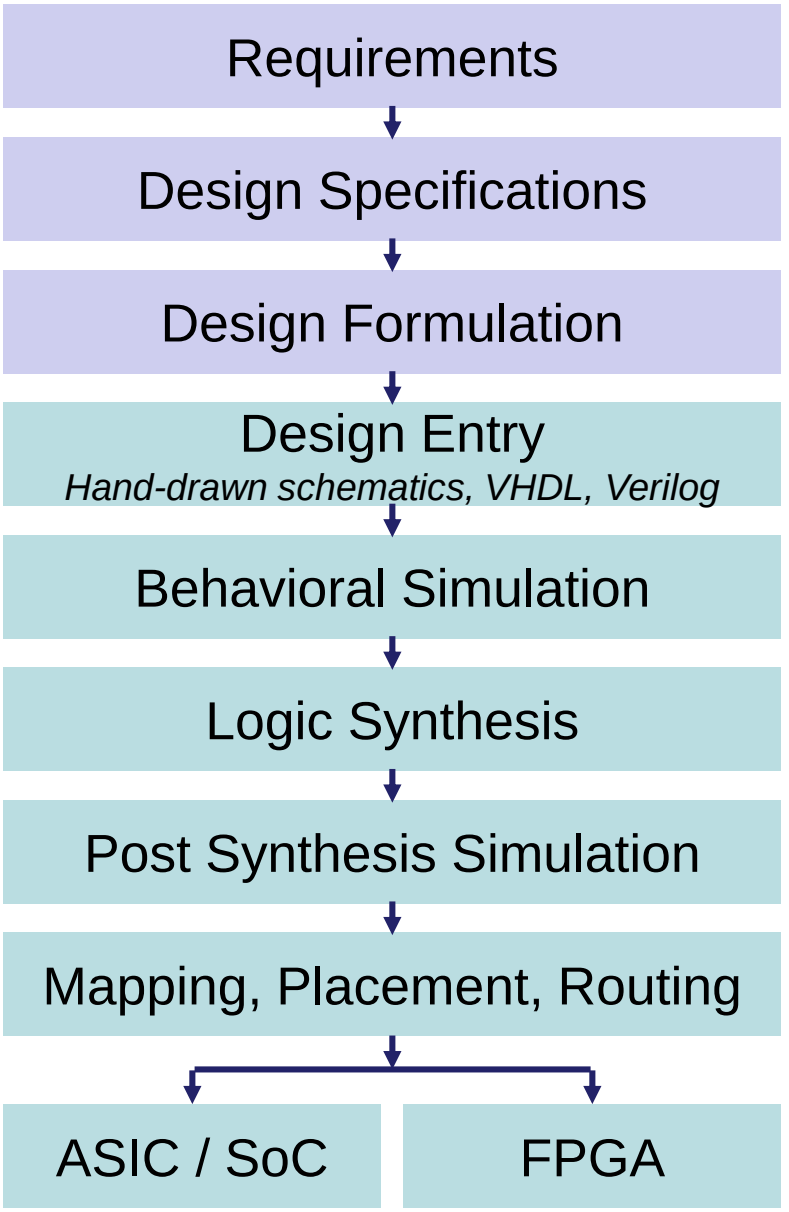
- Digital system design have become more complex.



Design Flow of Digital Systems (1/4)

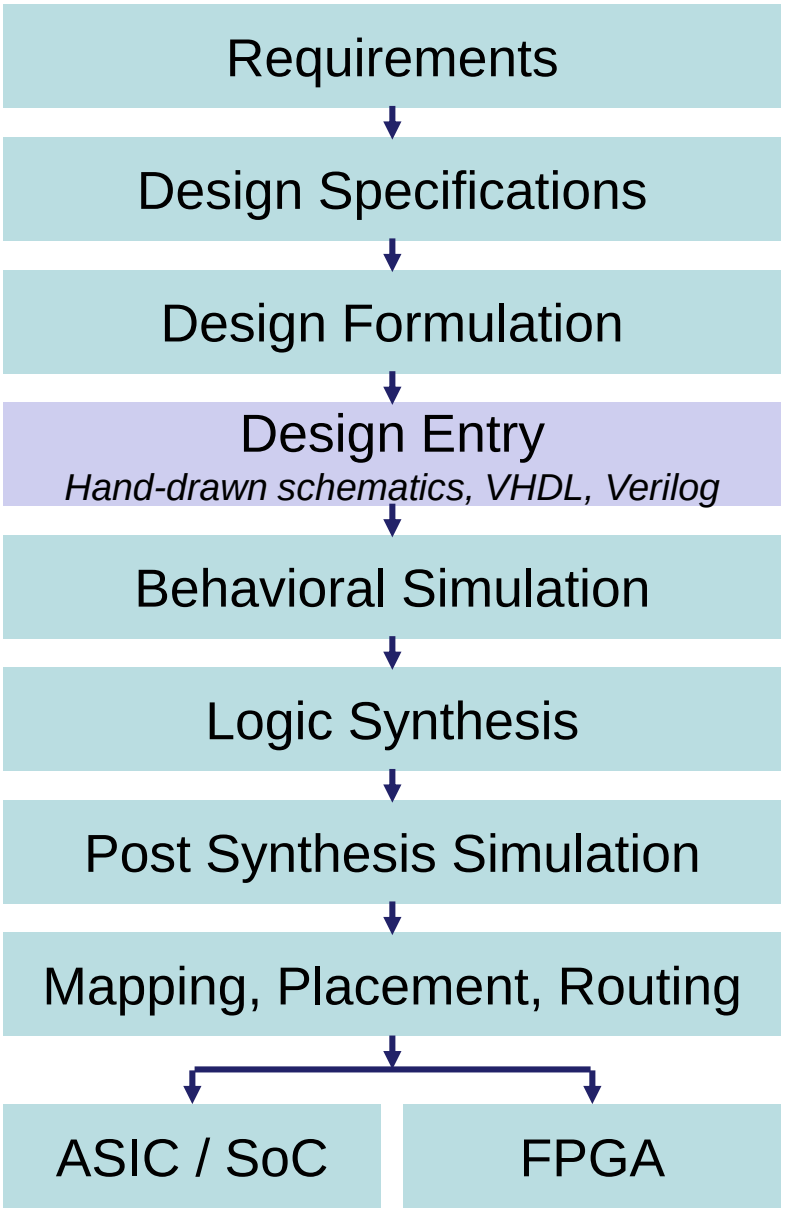
- All the designs start with design requirements and design specifications.
- The next step is to formulate the design conceptually.
  - Either at a block diagram level or at an algorithmic level.





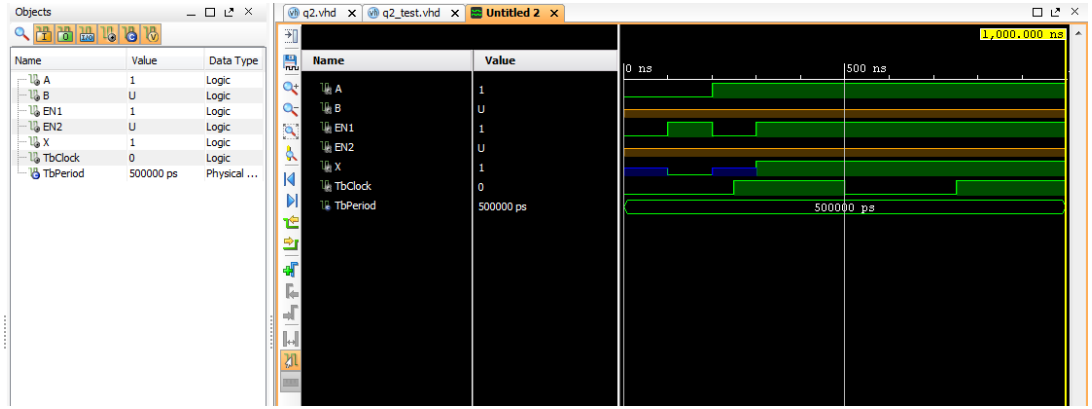
Design Flow of Digital Systems (2/4)

- Design Entry:
  - *Olden days*: Hand-drawn schematic or blueprint
  - *Now*: Computer-aided design (CAD) tools
    - **Schematic Capture**: Design with gates, flip-flops, and standard building blocks.
      - E.g., ORCAD
    - **Hardware Descriptions Languages (HDLs)**: Design and debug at higher level
      - E.g., VHDL and Verilog



Design Flow of Digital Systems

- The entered design then should be simulated.
  - To ensure it function correctly at high-level behavioral model
  - To unveil problems in the design
- Recall: In Lab01, we run
  - Flow Navigator → Run Simulation → Run **Behavioral Simulation**



Requirements

↓

Design Specifications

↓

Design Formulation

↓

Design Entry

Hand-drawn schematics, VHDL, Verilog

↓

Behavioral Simulation

↓

Logic Synthesis

↓

Post Synthesis Simulation

↓

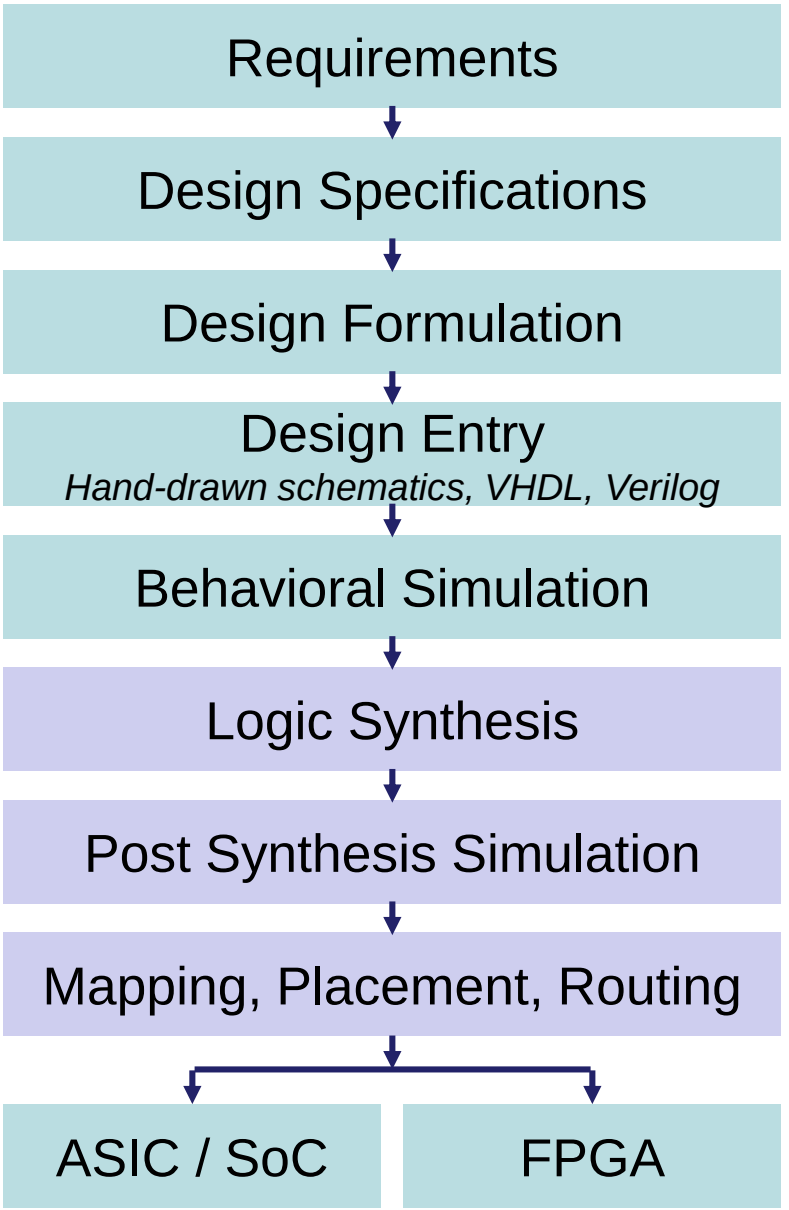
Mapping, Placement, Routing

↓

ASIC / SoC | FPGA

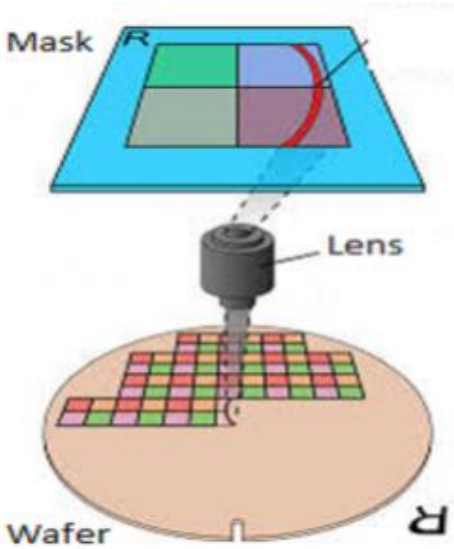
Design Flow of Digital Systems

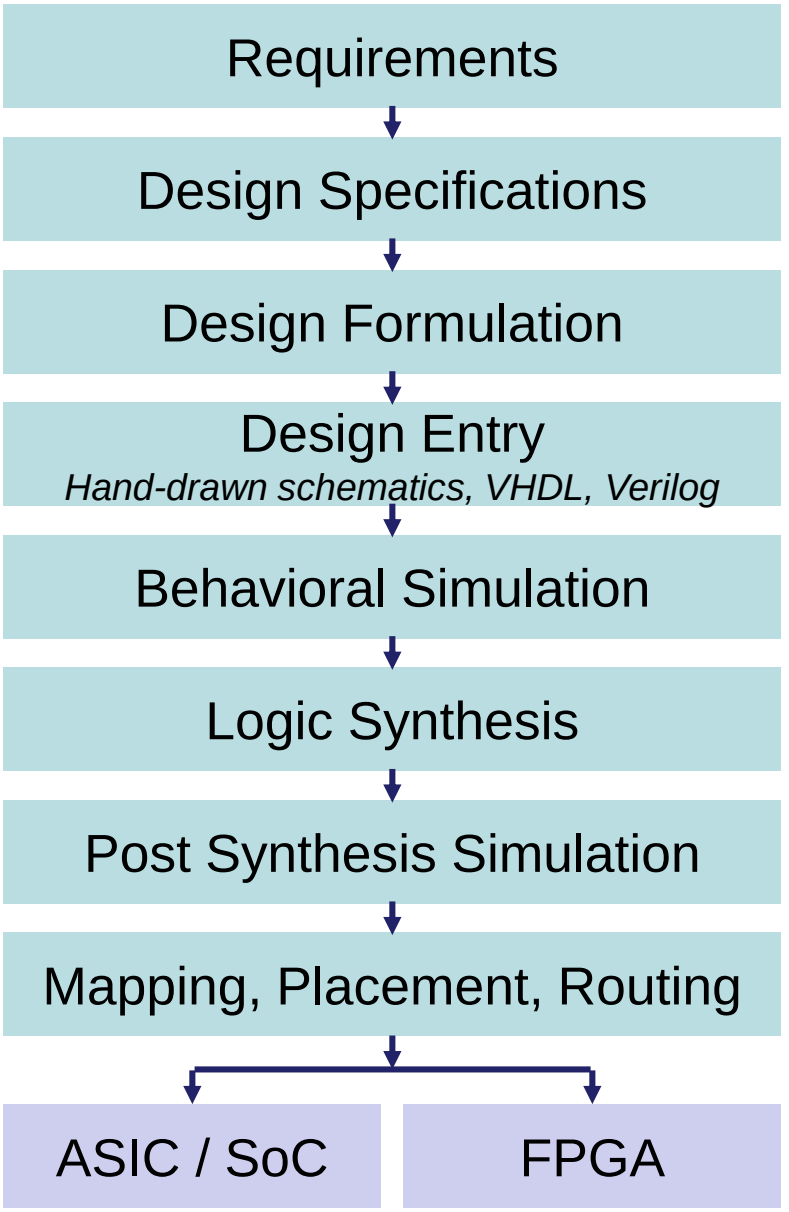
- Logic Synthesis
  - Convert the high-level abstract descriptions to components at the gate and flip-flop levels
    - Netlist: Outputs of synthesis tool
      - Specify internal connections
- Post-Synthesis Simulation
  - Test specific implementations of the hardware components
- Mapping, Placement, Routing
  - Mapped into the specific target
  - Placed into specific parts
  - Route the paths of connections



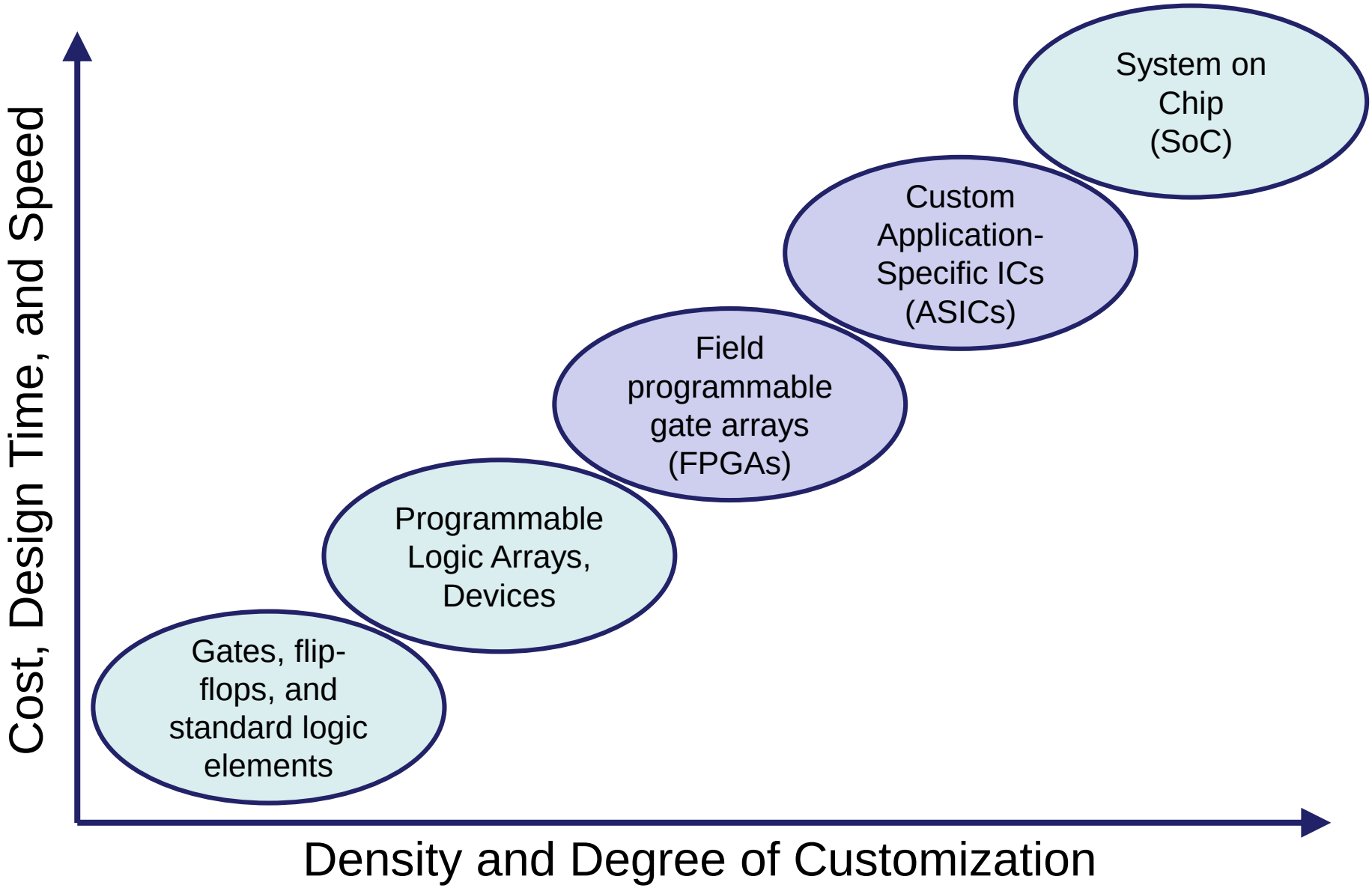
Design Flow of Digital Systems

- Two most common targets:
  - Field Programmable Gate Array (FPGA)
    - Programming simply involves writing a sequence of 0's and 1's into the programmable cells of FPGA.
  - AISC or SoC
    - The routed design is used to generate a photomask for producing integrated circuits (ICs).





Spectrum of Design Technologies



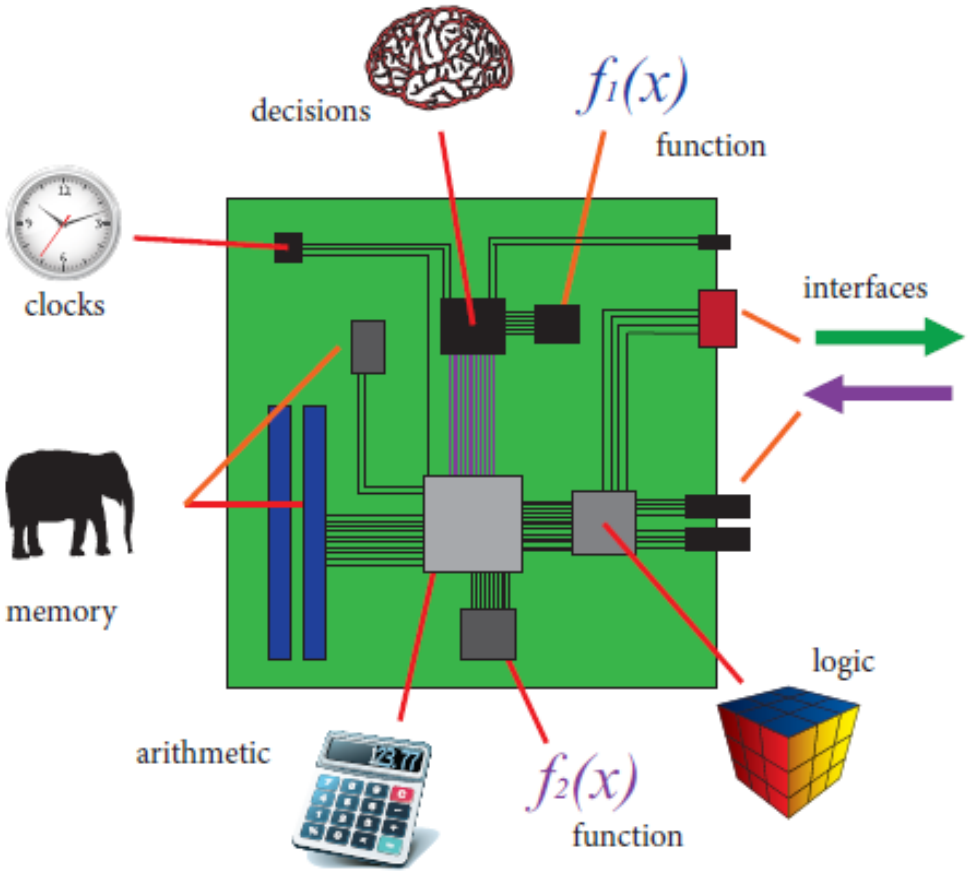
What is System-on-Chip (SoC)?

- **System-on-Chip**
  - The implication is a single silicon chip can be used to implement the functionality of an entire system.
    - Rather than several different physical chips being required.
  - An SoC can combine all aspects of a digital system.
    - E.g., processing, high-speed logic, interfacing, memory, and etc.

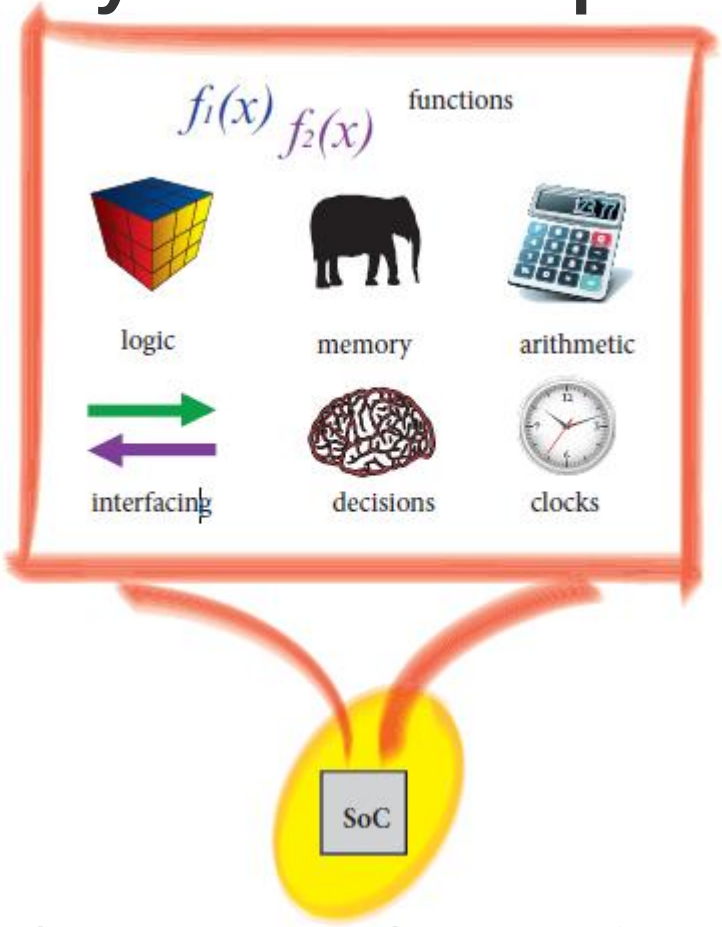
- Alternative: **System-on-a-Board**
  - Otherwise, all of these functions might be realized using physically separate devices.
    - By combining them together into a system at the Printed Circuit Board (PCB) level.

System-on-a-Board vs. System-on-Chip

System-on-a-Board



System-on-Chip



- The SoC solution has lower cost, faster and more secure data transfers between the various system elements, higher overall system speed, lower power consumption, smaller physical size, and better reliability.

Application Specific Integrated Circuit

- In the past, the term SoC has usually referred to an Application Specific Integrated Circuit (ASIC).
  - ASIC can include digital, analogue and radio frequency components, together with mixed signal blocks for implementing analogue-to-digital and digital-to-analogue converters (ADCs and DACs).
- ASIC-based SoCs
  - Representative examples are the integrated processors found in PCs, tablets, and smartphones.
    - These typically comprise multiple processor cores, memory, graphics, interfacing, and other functions, and are manufactured in high volume for products with a limited lifetime.
  - The major disadvantages are development time and cost, and lack of flexibility.

System-on-Programmable-Chip

- There is a clear need for a more flexible solution: the System-on-Programmable-Chip.
  - A specific flavor of SoC implemented on a programmable, reconfigurable device.

- The natural solution has long been the FPGA.
  - FPGAs are inherently flexible devices that can be configured to implement any arbitrary system, including embedded processors if needed.
  - FPGAs can also be reconfigured as often as desired, thus offering a more fundamentally flexible platform than ASICs for implementing SoCs.

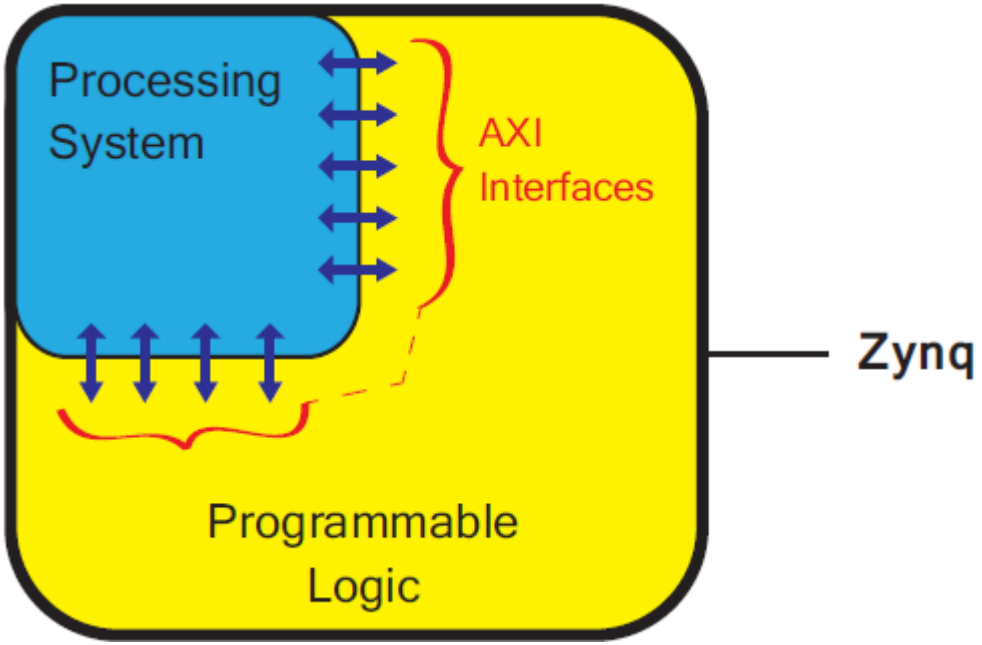
Outline

- Digital System Design Basics
  - Integrated Circuit Technology
  - Design Flow of Digital Systems
  - System-on-Chip (SoC)
  - Application Specific Integrated Circuit (ASIC)

- Zynq: All-Programmable SoC (APSoC)
  - Our Board: ZedBoard
    - ZedBoard Layout and Interfaces
    - Specifying ZedBoard in Vivado
    - Hardware Setup for ZedBoard
    - Programming the ZedBoard
    - Xilinx Design Constraints (XDC) File

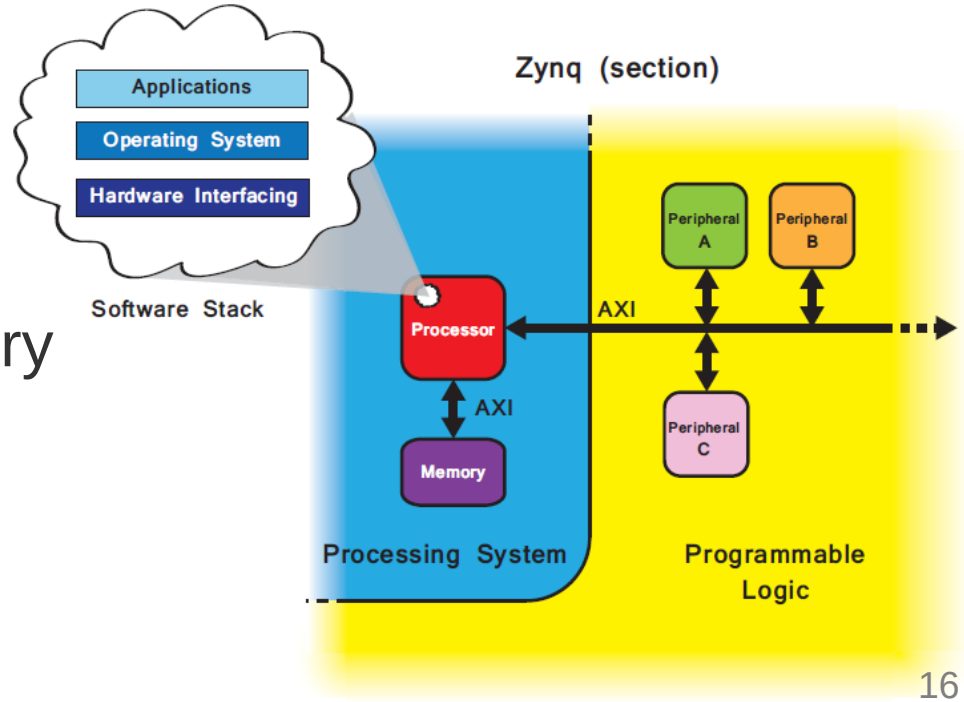
Zynq: All-Programmable SoC (1/2)

- All-Programmable SoC (APSoC): Zynq provides an more ideal platform for implementing flexible SoCs.
- Zynq comprises two main parts:
  - Programmable Logic (PL): equivalent to that of an FPGA
  - Processing System (PS): formed around a dual-core ARM Cortex-A9 processor



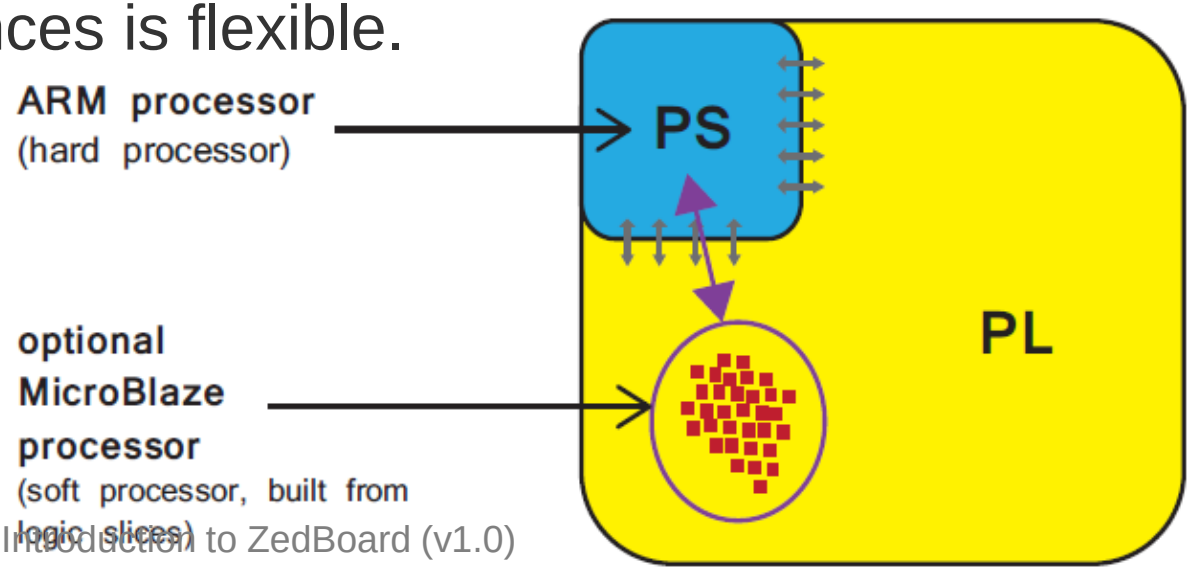
Zynq: All-Programmable SoC (2/2)

- **Programmable Logic (PL)**: Implements high-speed logic, arithmetic and data flow subsystems.
- **Processing System (PS)**: Supports software routines and/or operating systems.
  - The overall functionality of any designed system can be appropriately partitioned between hardware and software.
- **Advanced eXtensible Interface (AXI)**:
  - Links between the PL and PS are made using industry standard Advanced eXtensible Interface (AXI) connections.



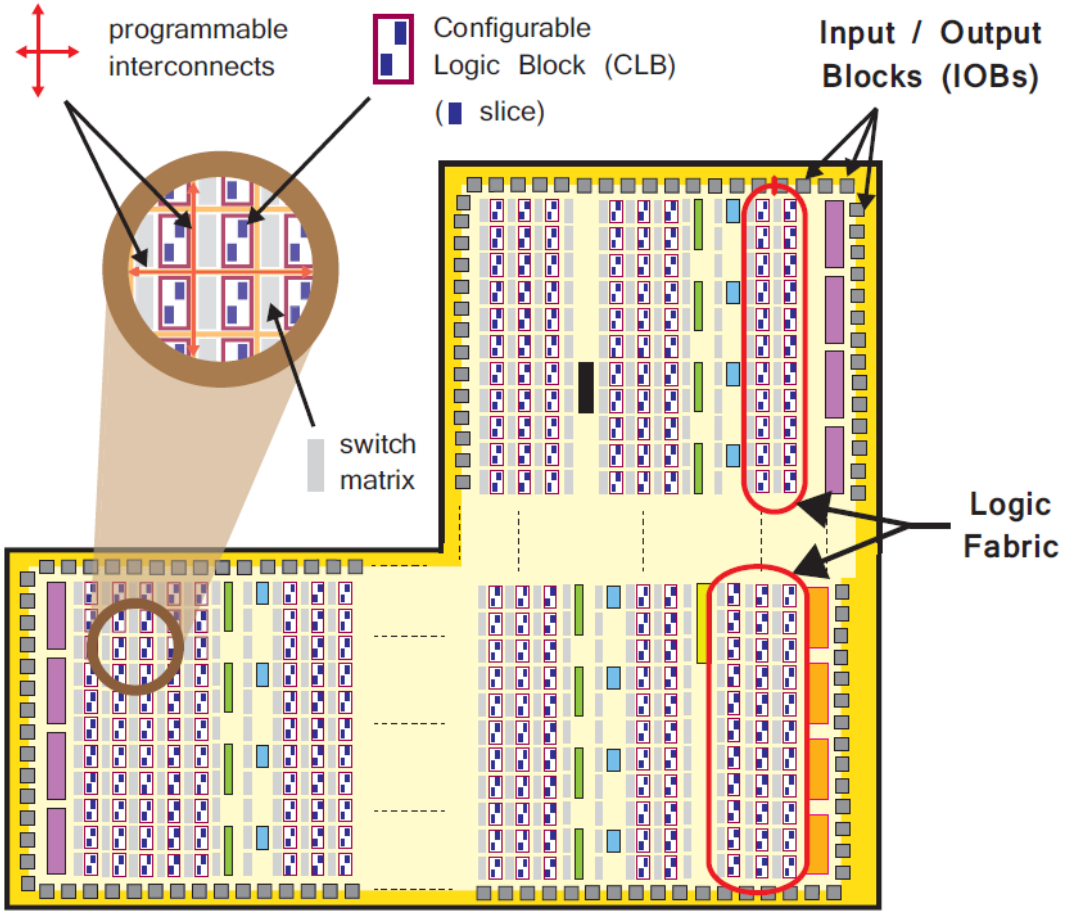
Processing System (PS)

- PS supports software routines and operating systems.
  - The overall functionality of any system can be partitioned.
- PS has a "hard" dual-core ARM Cortex-A9 processor.
  - Hard processors can achieve higher performance.
- By contrast, "soft" processor (e.g., Xilinx MicroBlaze) can be made by the programmable logic elements.
  - The number and precise implementation of soft processor instances is flexible.



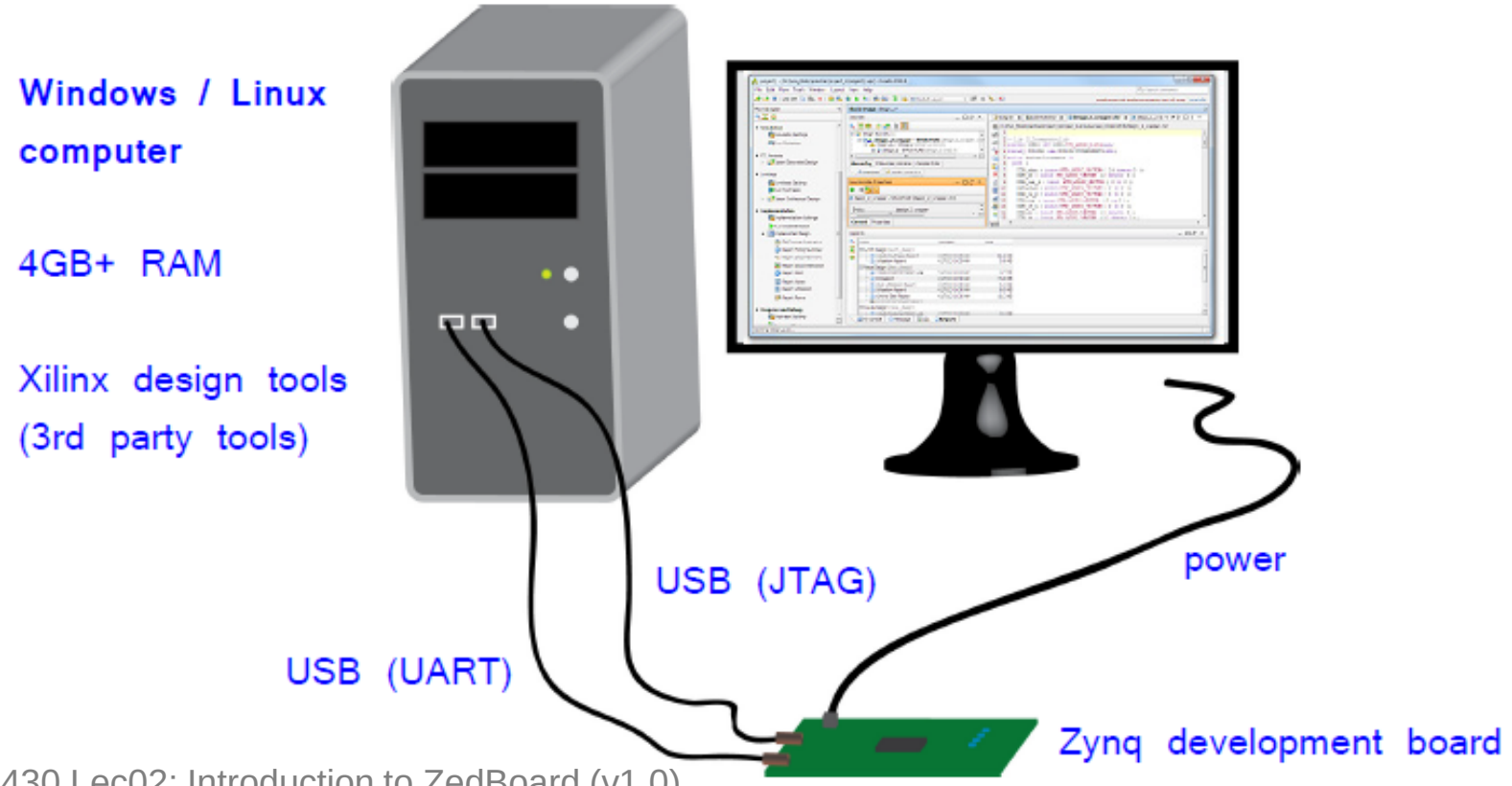
Programmable Logic (PL)

- PL section is ideal for implementing high-speed logic, arithmetic and data flow subsystems.
- PL is composed of general purpose FPGA logic fabric.



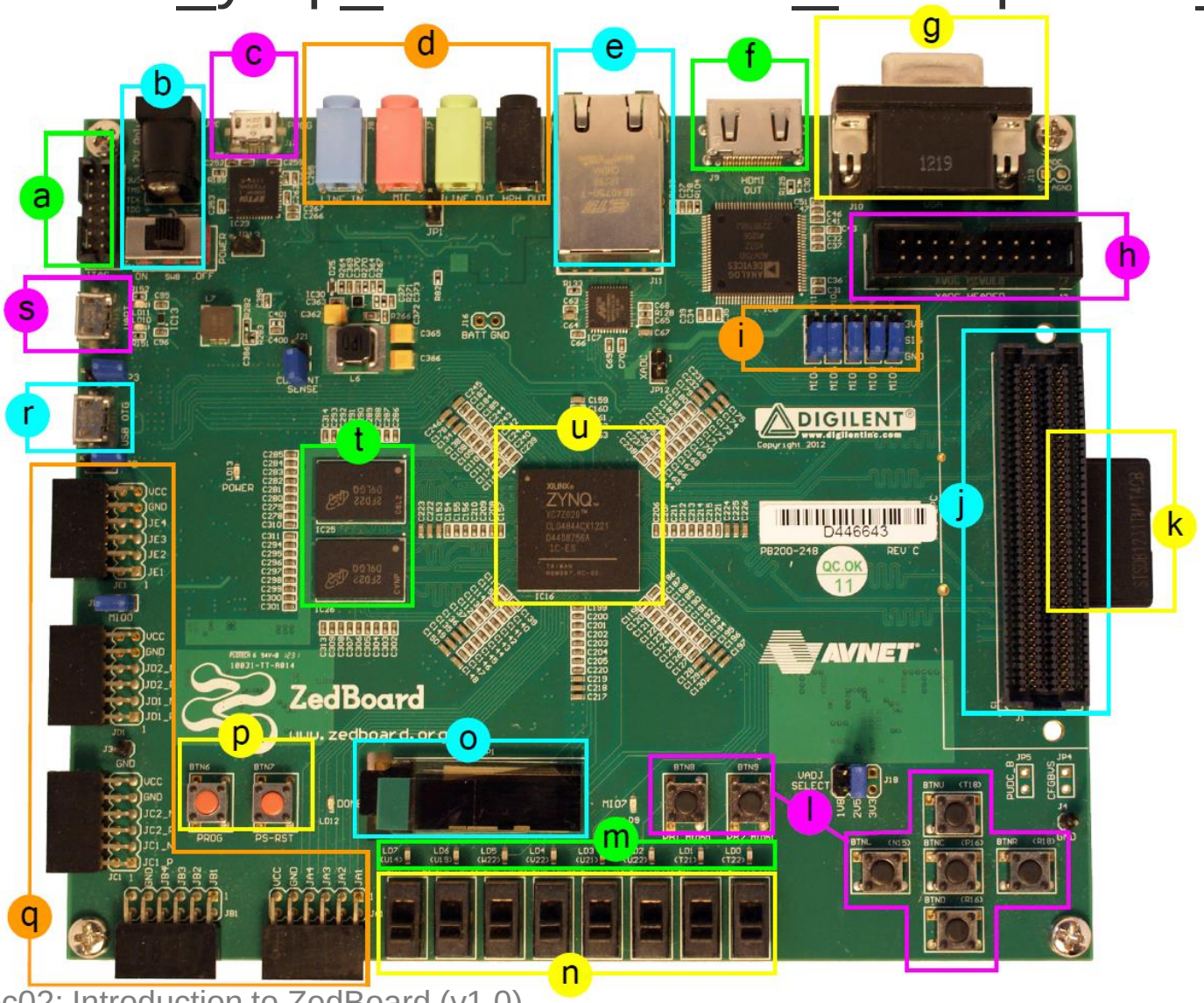
Zynq Development Setup

- Joint Test Action Group (JTAG): Downloading designs onto the development board over JTAG
- Universal Asynchronous Receiver/Transmitter (UART) and Terminal Applications: Interfacing and debugging



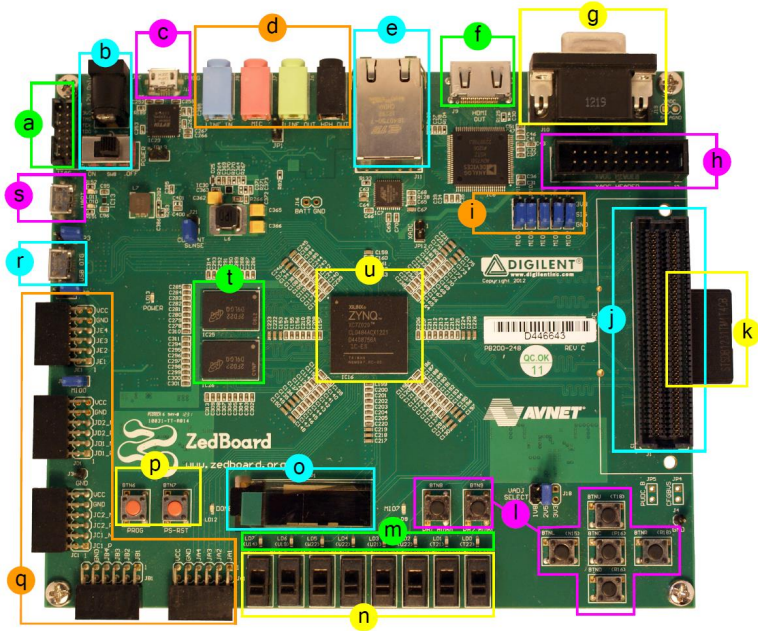
Our Board: Zynq ZedBoard

- ZedBoard: **Z**ynq **E**valuation and **D**evelopment **Board**



ZedBoard Layout and Interfaces

- ZedBoard features a ZC7Z020 Zynq device.
  - Artix-7 logic fabric, with a capacity of 13,300 logic slices, 220 DSP48E1s, and 140 BlockRAMs
  - DDR3 Memory, and Flash
  - Several peripheral interfaces



a Xilinx JTAG connector

b Power input and switch

c USB-JTAG (programming)

d Audio ports

e Ethernet port

f HDMI port (output)

g VGA port

h XADC header port

i Configuration jumpers

j FMC connector

k SD card (underside)

l User push buttons

m LEDs

n Switches

o OLED display

p Prog & reset push buttons

q 5 x Pmod connector ports

r USB-OTG peripheral port

s USB-UART port

t DDR3 memory

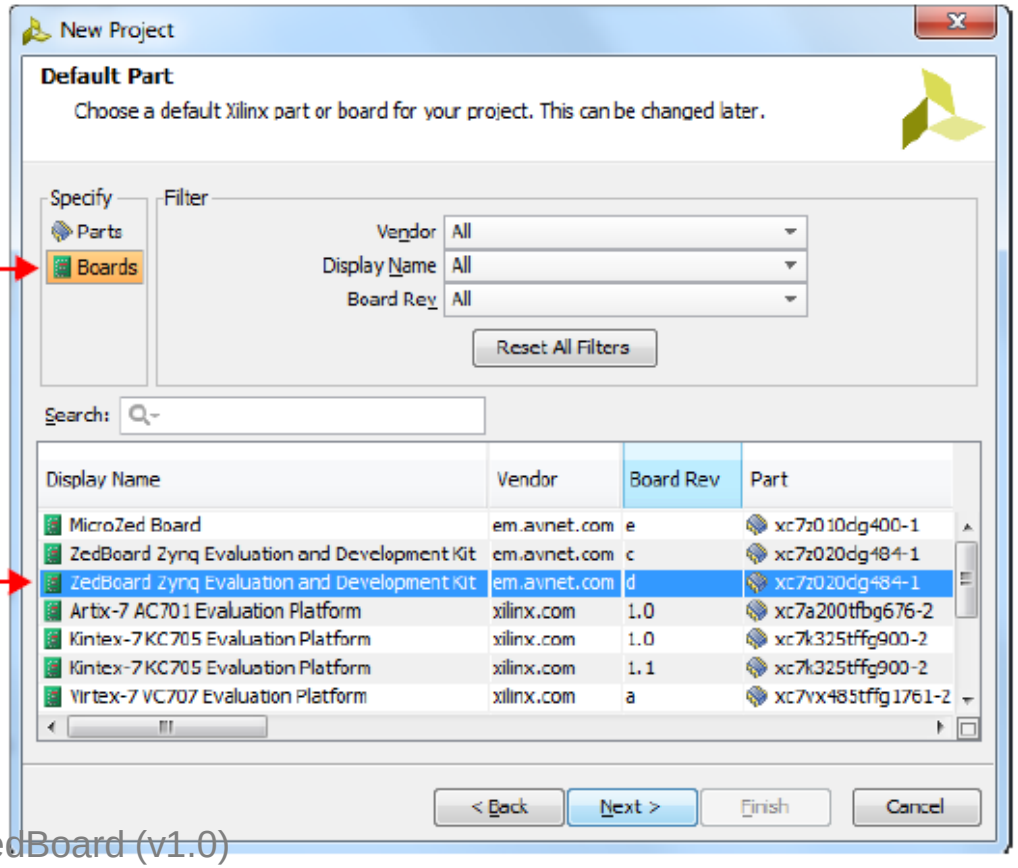
u Zynq device (+ heatsink)

Specifying ZedBoard in Vivado

- **ZedBoard Zynq Evaluation and Development Kit**:
  - The design tools have knowledge of the specific facilities and peripheral connections of ZedBoard.
- Target part: **xc7z020clg484-1**, Rev: **d**

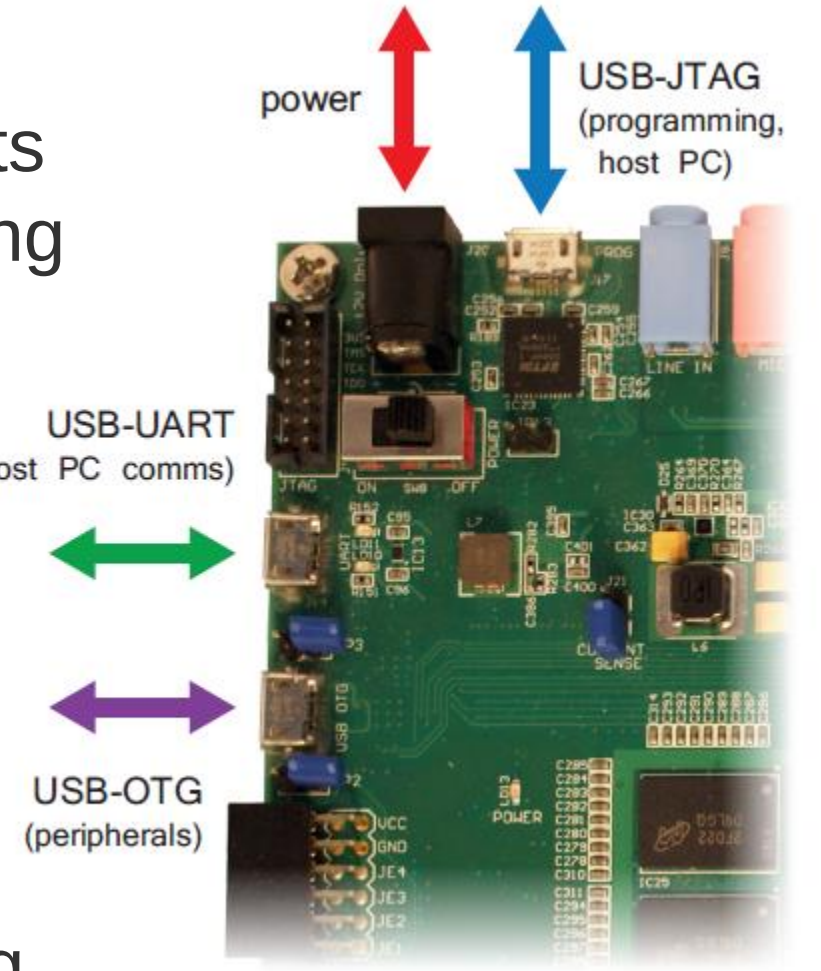
choose ***Boards*** here to see the selection...

select the ***ZedBoard*** (choose the 'Board Rev' for you board)



Hardware Setup for ZedBoard

- The ZedBoard must be connected to a power supply.
- By default, ZedBoard connects to the host PC for programming over USB-JTAG.
- An additional connection can be made over USB-UART.
  - If intending to facilitate simple board-PC communication using the Terminal application.
- Note that there is also a third micro-USB port for connecting USB peripherals (USB-OTG)



Programming the ZedBoard (1/2)

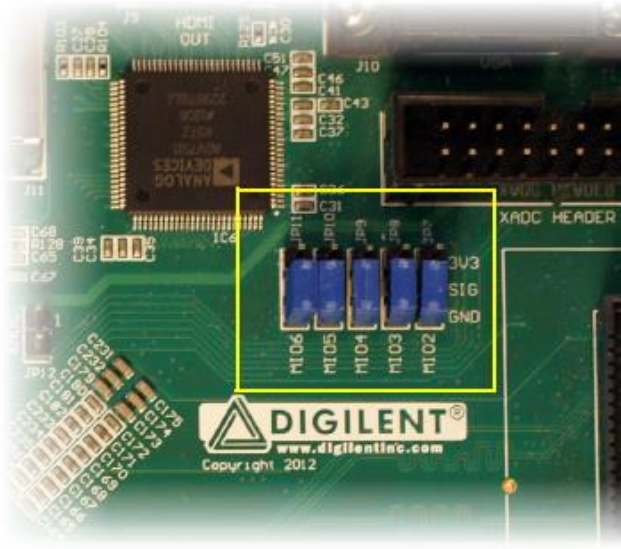
- ZedBoard can be programmed in four different ways:
  - **USB-JTAG:** This is the default and most straightforward method of programming the ZedBoard, given that it can be done directly over the USB-micro-USB cable supplied in the ZedBoard kit.
  - **Traditional JTAG:** A Xilinx JTAG connector is available on the board and may be used in place of the USB-JTAG connection, if desired. This will require a different type of cable or a *Digilent USB-JTAG programming cable*.
  - **Quad-SPI flash memory:** The flash memory on the board is non-volatile and therefore can be used to store configuration data which persists when the board is powered off. Using this method removes the requirement for a wired connection to program the Zynq device.
  - **SD card:** There is an SD slot on the underside of the ZedBoard. This facility can be used to program the Zynq with files stored on the SD card, thus requiring no wired connections.

Programming the ZedBoard (2/2)

- The ZedBoard user specifies the method of booting / programming via a set of jumper pins.
  - The middle three are for specifying programming source.

	MIO[6]	MIO[5]	MIO[4]	MIO[3]	MIO[2]
In Xilinx Technical Reference Manual...	Boot_ Mode[4]	Boot_ Mode[0]	Boot_ Mode[2]	Boot_ Mode[1]	Boot_ Mode[3]
JTAG Mode					
Cascaded JTAG[a]	-	-	-	-	0
Independent JTAG	-	-	-	-	1
Boot Device					
JTAG	-	0	0	0	-
Quad-SPI (flash)	-	1	0	0	-
SD Card[a]	-	1	1	0	-
PLL Mode					
PLL Used[a]	0	-	-	-	-
PLL Bypassed	1	-	-	-	-

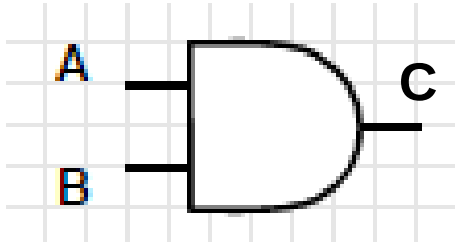
Cascaded: A single JTAG connection is used to interface to the debug access ports in both the PS and PL.



The PLL mode determines whether the process of configuring the device includes a phase of waiting for the PLL to lock

Xilinx Design Constraints (XDC) File

- To map the external interfaces in the design to physical pins on the ZedBoard, a Xilinx Design Constraints (XDC) file must be created and included.
  - External interfaces examples: switches, LEDs



- For example: **`C <= A and B`**
  - set_property PACKAGE_PIN T22 [get_ports {**C**}]; **# "LD0“**
  - set_property PACKAGE_PIN F22 [get_ports {**A**}]; **# "SW0"**
  - set_property PACKAGE_PIN G22 [get_ports {**B**}]; **# "SW1“**
- How to create your own XDC file? See Lab02.

Summary

- Digital System Design Basics
  - Integrated Circuit Technology
  - Design Flow of Digital Systems
  - System-on-Chip (SoC)
  - Application Specific Integrated Circuit (ASIC)

- Zynq: All-Programmable SoC (APSoC)
  - Our Board: ZedBoard
    - ZedBoard Layout and Interfaces
    - Specifying ZedBoard in Vivado
    - Hardware Setup for ZedBoard
    - Programming the ZedBoard
    - Xilinx Design Constraints (XDC) File