VietNam National University
University of Engineering and Technology

THIẾT KẾ HỆ THỐNG NHÚNG

Instructor: Dr. Nguyễn Kiêm Hùng
Email: kiemhung@vnu.edu.vn

Lecture 1: FUNDAMENTAL OF EMBEDDED COMPUTING SYSTEMS

Objectives

In this lecture you will be introduced to:

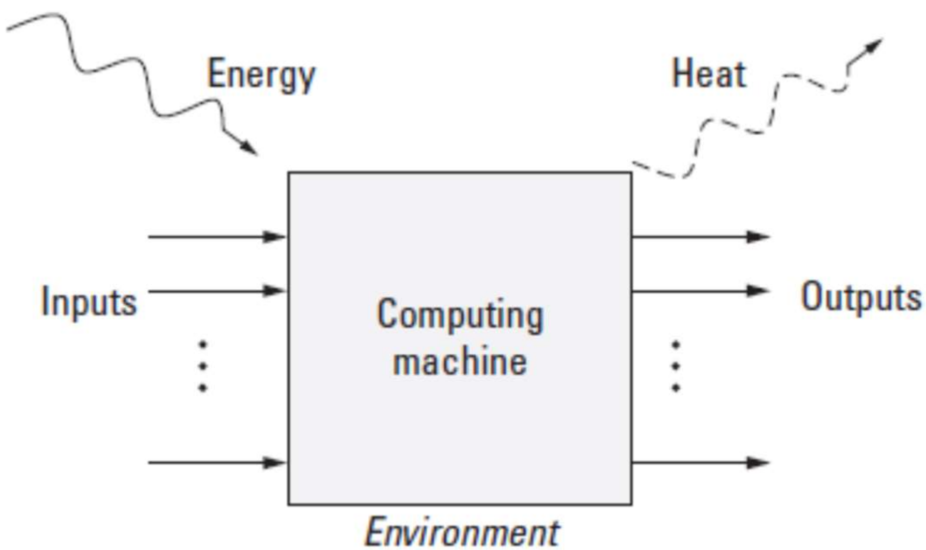
- Basic concepts of Embedded Computing Systems,
- What is **difference** between embedding computing system and general-purpose computing system,
- The embedded system design process.

Outline

- **Embedded Computing System**
- Embedded System Design using FPGA
- Characteristics of Embedded Computing Applications
- Challenges in Embedded Computing System Design
- Embedded System Design Process
- Summary

Review of Computing System

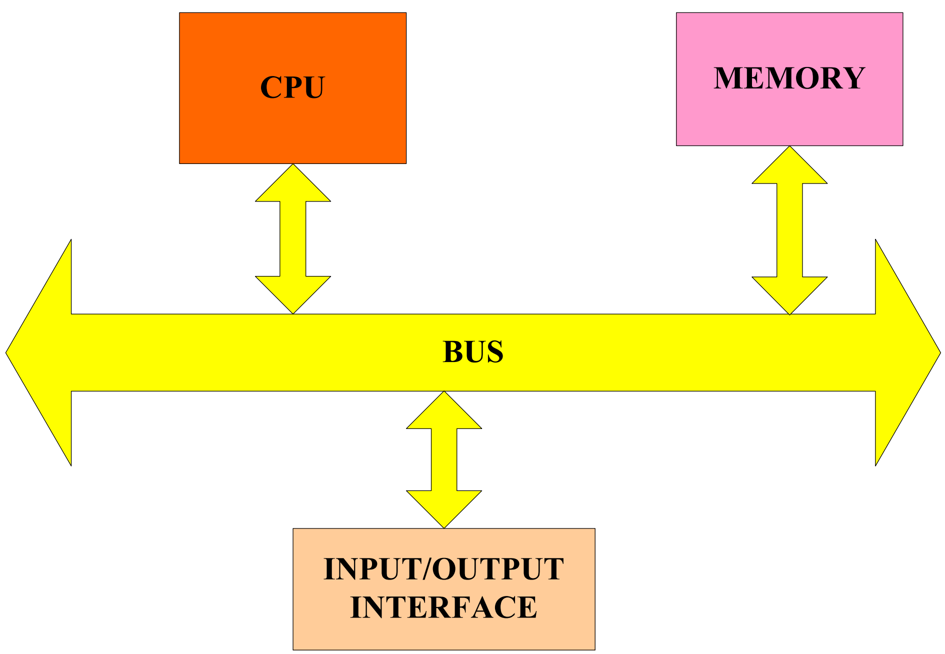
- A computing system (or just computer) is frequently modeled as a system that includes:
  - **inputs**: physical signals from the environment ,
  - **outputs**: response of system to the environment ,
  - and a **processing unit**: fetches and executes instructions from a memory



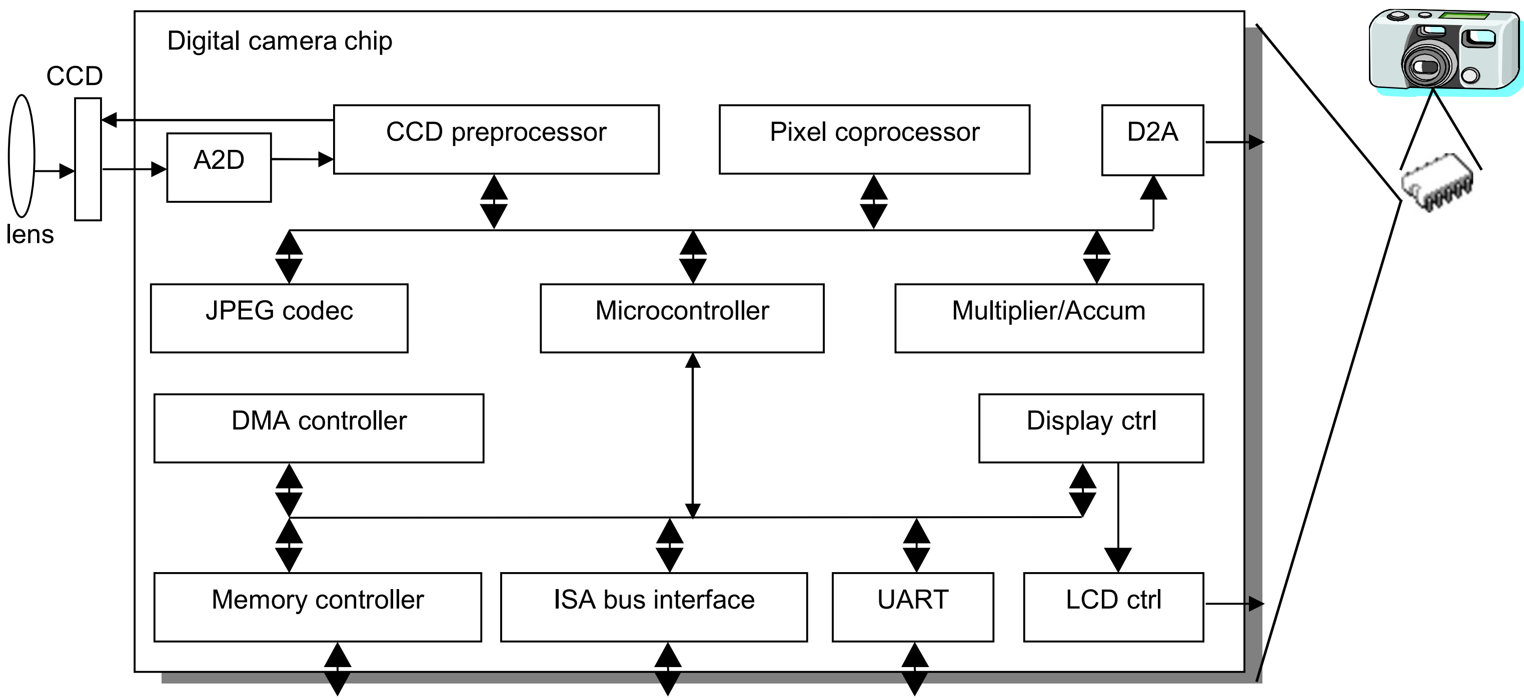
An abstract view of a computing system

Review of Computing System

Which component does a minimal computing system consists of ?



An embedded system example -- a digital camera



- Single-functioned -- always a digital camera
- Tightly-constrained -- Low cost, low power, small, fast
- Reactive and real-time -- only to a small extent

Embedded Computing System

- An embedded system is **a specialized computing machine** (other than a general-purpose computer) with the following characteristics:
  - Single dedicated function
    - Typically designed to perform a predefined function
  - Tightly constraints
    - Low cost
    - Single-to-fewer component
    - High performance
    - Low power consumption
  - Real-time response
    - Must continually monitor the desired environment and react to changes in real-time
  - Hardware and software coexistence

Embedded Computer vs. General-purpose Computer

How the computing machine is used?

Embedded Computing System	General-purpose computer
- A component of some larger product; its purpose is narrowly focused on supporting that product	- A complete product itself
- End user of the product typically does not directly interact with the embedded system, or interacts with only a limited interface	- End user directly interacts with it
- User don’t consider the product (e.g. DVD players, MP3 players, game consoles) they are buying is a computer	- User explicitly knows the product they are buying is a computer

Embedded Computer vs. General-purpose Computer

How the computing machine is used?

Embedded System	General-purpose computer
- Having some of these standard peripherals as well, but also include much more specialized ones: + special-purpose sensors: accelerometers, temperature probes, magnetometers, push button contact switches, etc. + special-purpose outputs: lamps and LEDs, actuators, TTL electrical signals, LCD displays, etc.	- Having relatively few, standardized inputs and outputs: keyboards, mice, network connections, video monitors, and printers

Embedded system Definition

"Embedded systems are computing systems with **tightly coupled hardware and software integration**, that are designed to perform a **dedicated function**. The word '**embedded**' reflects the fact that these systems are usually an integral part of a larger (mechanical or electrical) system/product, known as the **embedding** system. Multiple embedded systems can coexist in an embedding system." – (*Qing Li and Carolyn Yao*)

Embedded Computer vs. General-purpose Computer

Is a tablet computer an embedded system?

- Characteristics of general-purpose computer:
  - what is its enclosing product?
  - It is not used to control anything
  - the computing system is exposed to the user: the user can download general-purpose applications.
- Characteristics of embedded systems:
  - very sensitive to size, weight, and power constraints.
  - a limited user interface
  - Etc.

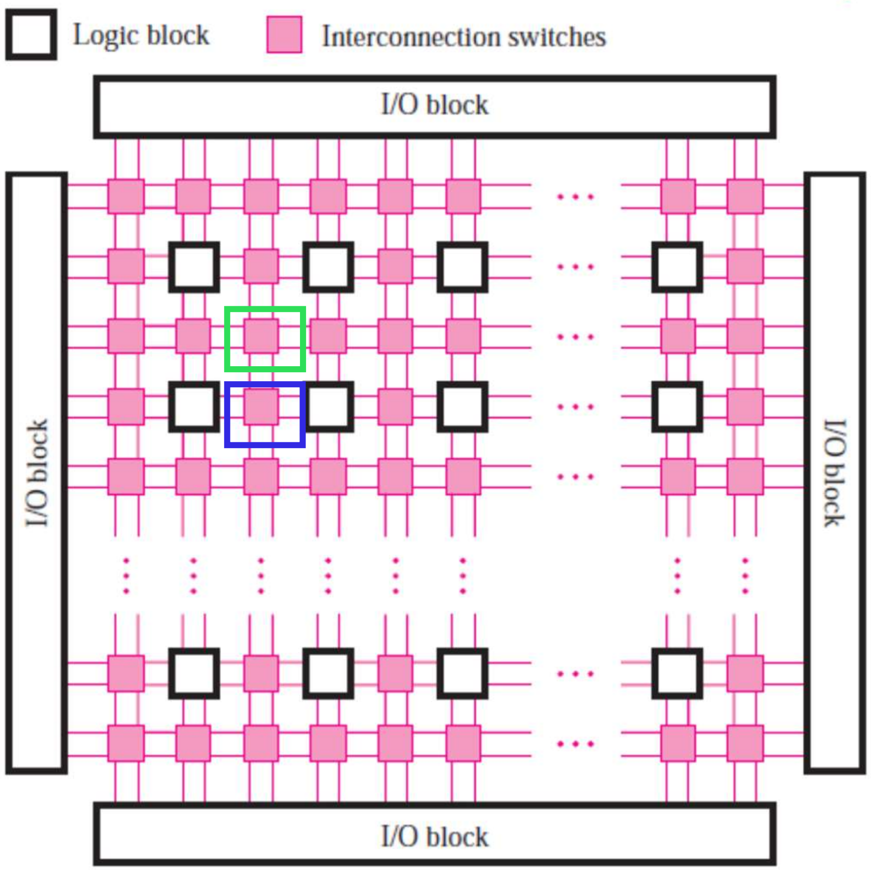
Embedded Computer vs. General-purpose Computer

The exact boundary between general-purpose and embedded is not black and white but rather a spectrum.

Outline

- Embedded Computing System
- **Embedded System Design using FPGA**
- Characteristics of Embedded Computing Applications
- Challenges in Embedded Computing System Design
- Embedded System Design Process
- Summary

What is FPGA?

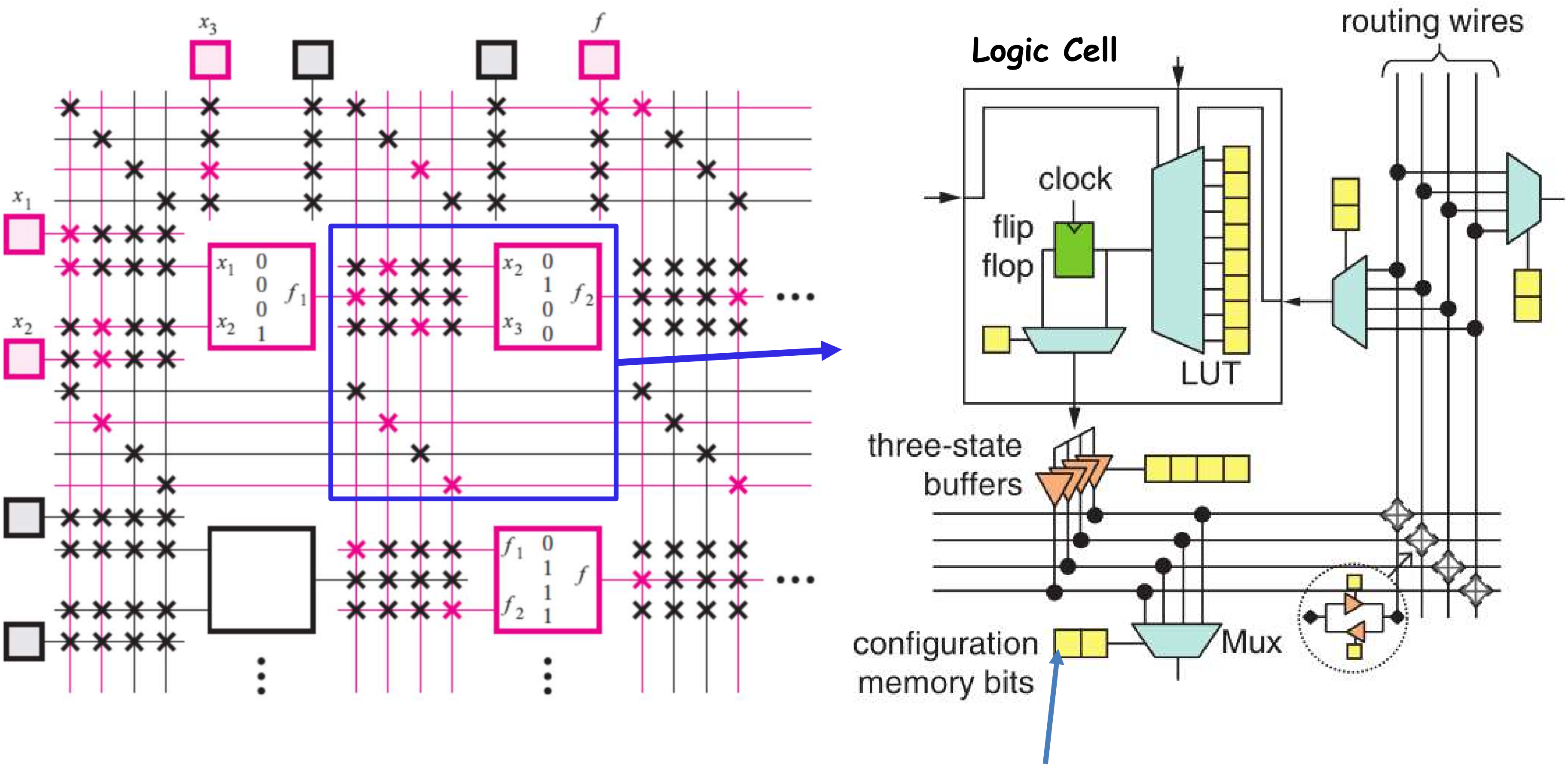


Overview of the FPGA architecture

Field Programmable Gate Array:

- **Pre-fabricated digital** (IC) devices
- Electrically **programmed to become almost any kind of digital circuit or system**
- **Programming takes place "in the field".**
- Comprises of
  - Configurable logic blocks (CLB),
  - Programmable routing resources: wires and switches
  - I/O blocks.
- Adopts the programming technologies:
  - SRAM-based technology
  - Flash/EEPROM technology
  - Anti-fuse technology

Example of simple FPGA

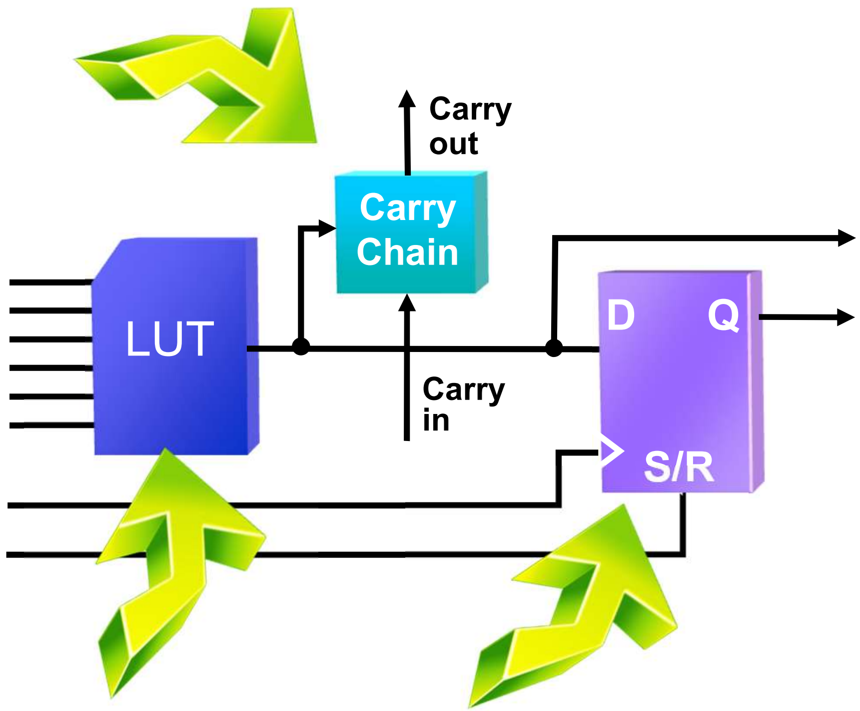


$$f = f_1 + f_2 = x_1 x_2 + \bar{x}_2 x_3$$

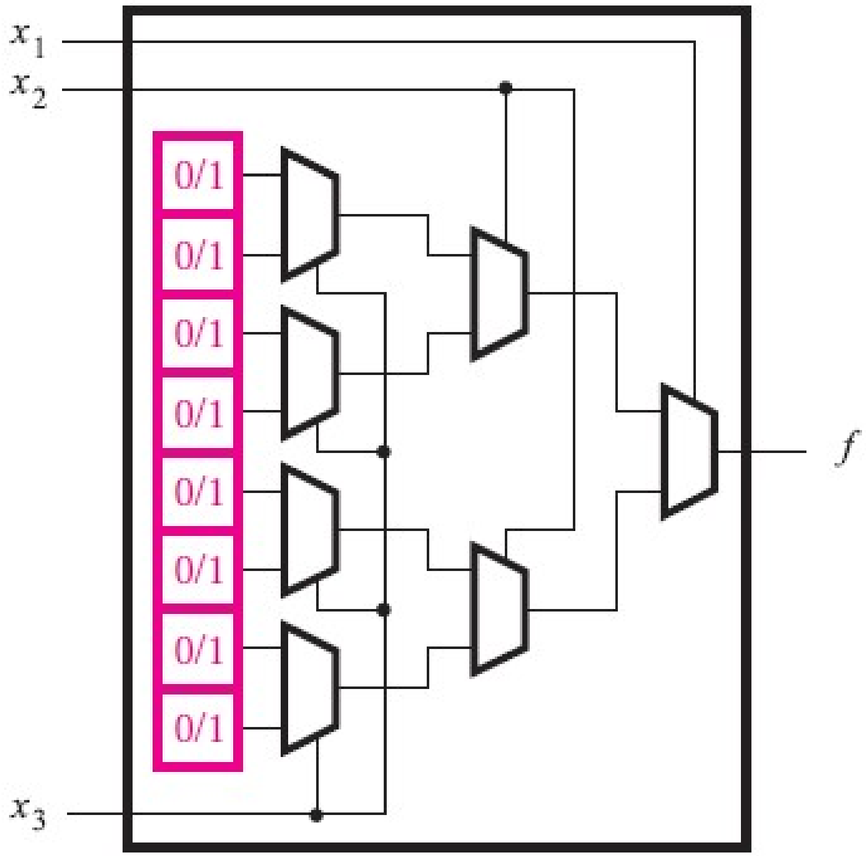
A memory element for storing configuration information

Logic Cell

- ***Logic cells include***
  - Combinatorial logic, arithmetic logic, and a register
- ***Combinatorial logic*** is implemented using Look-Up Tables (LUTs)
- **Register** can function as latches, JK, SR, D, and T-type flip-flops
- ***Arithmetic logic*** is a dedicated carry chain for implementing fast arithmetic operations



LUT: Lookup Table

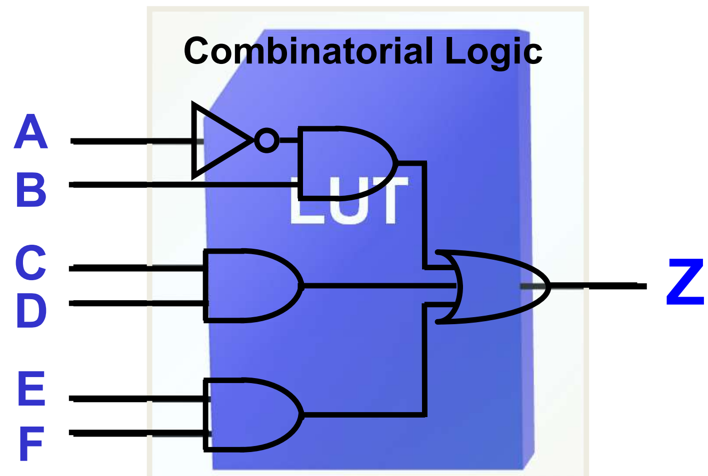


- **Used to implement a small logic function**
- **Composed of:**
  - *storage cells* store values that produce the output of the logic function *f*
  - Multiplexers select the content of one of the storage cells as the output of the LUT
- **LUT's *size* is defined by the number of inputs**

Combinatorial Logic

A	B	C	D	E	F	Z
0	0	0	0	0	0	0
0	0	0	0	0	1	0
0	0	0	0	1	0	0
0	0	0	0	1	1	1
0	0	0	1	0	0	1
0	0	0	1	0	1	
	1		.	.	.	
0	0	1	1	0	0	0
0	0	1	1	0	1	0
0	0	1	1	1	0	0
0	0	1	1	1	1	1

- LUTs function as a Memory or can **perform any combinatorial function**

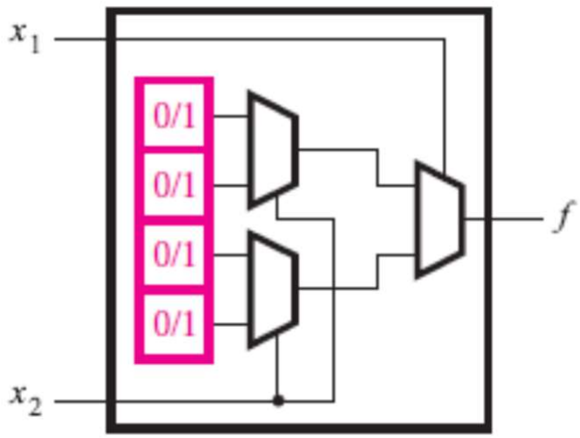


They generate the output value…

for a given set of inputs

- Constant delay through a LUT
- Limited by the number of inputs and outputs, not by complexity

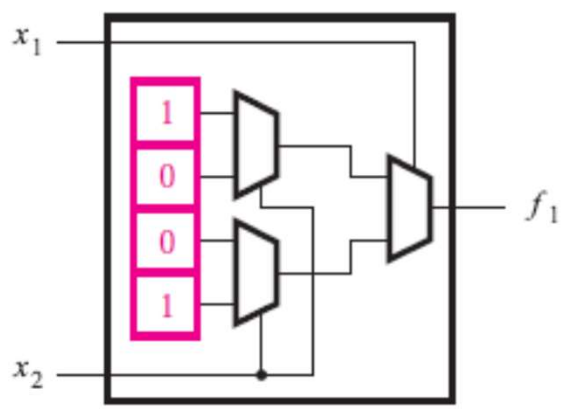
LUT: A Simple Example



(a) Circuit for a two-input LUT

x_1	x_2	f_1
0	0	1
0	1	0
1	0	0
1	1	1

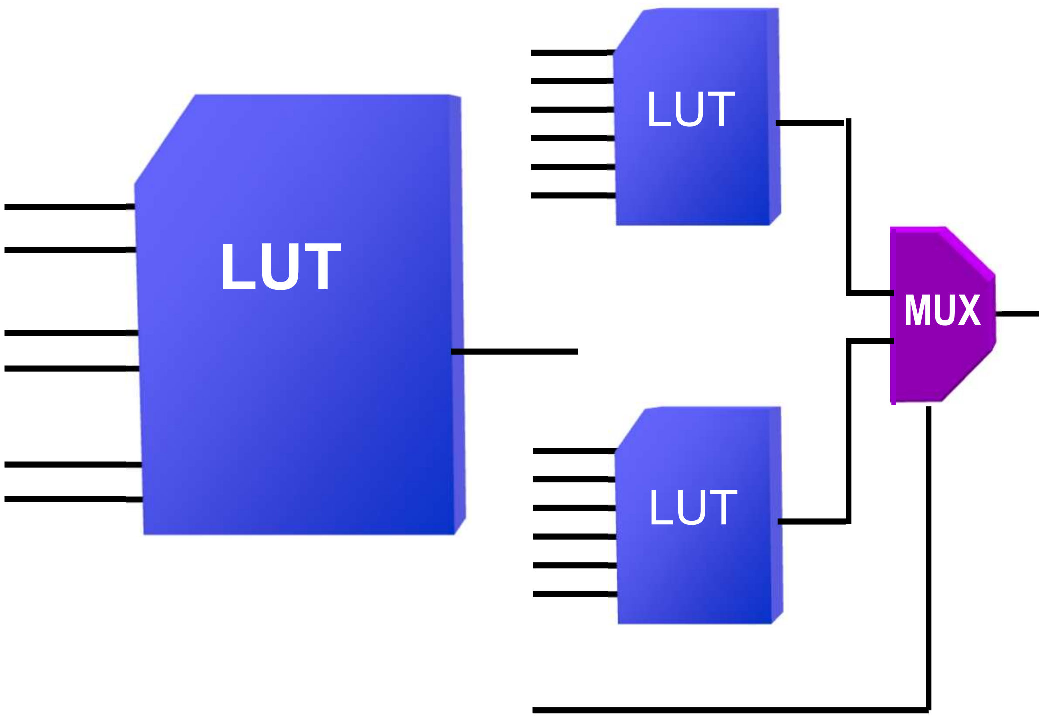
(b) $f_1 = \bar{x}_1\bar{x}_2 + x_1x_2$



(c) Storage cell contents in the LUT

Wide Input Functions

- For wider input functions, LUTs can be combined using a multiplexer
- These muxes are dedicated, so they are fast

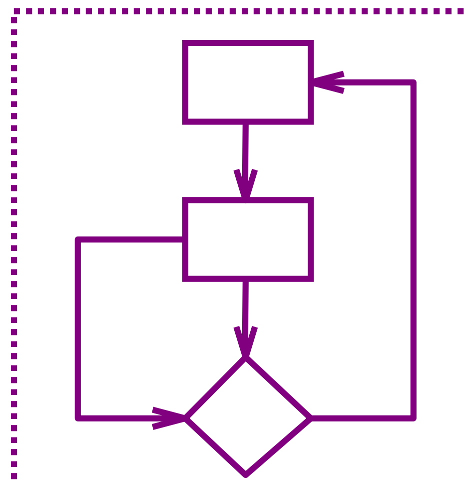


FPGA Design Flow

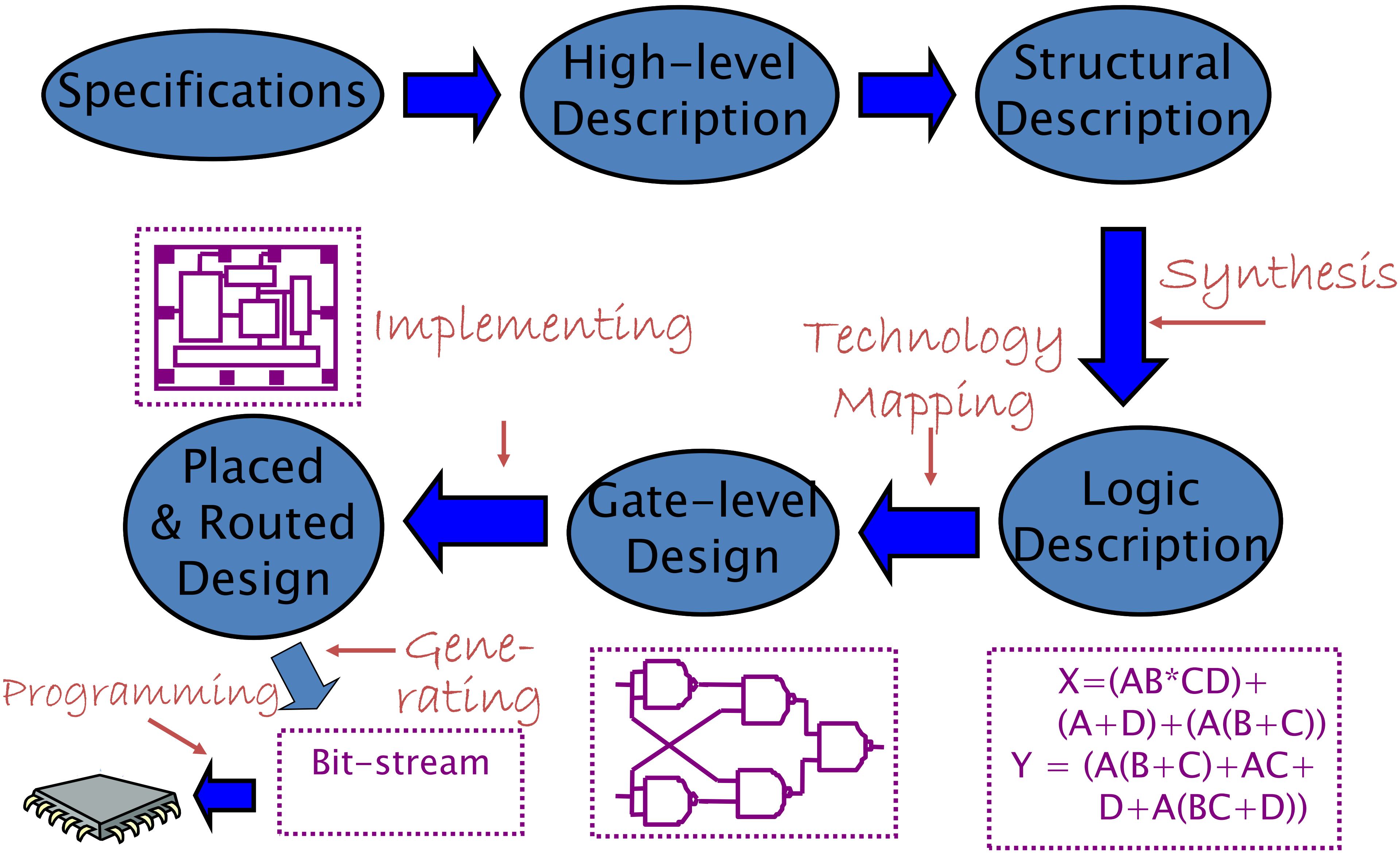
Specifications → High-level Description → Structural Description

Behavioral VHDL, C

Structural VHDL



FPGA Design Flow



FPGA Applications

- **Implementing the prototype for ASIC designs**
- **Providing a hardware platform to verify the physical implementation of new algorithms in:**
  - Digital signal processing (DSP),
  - Baseband processing in communication,
  - Software-defined radios,
  - Radar,
  - Video, image processing,
  - Physical layer communication interfaces, etc
- **On-Chip embedded processing systems**
- **Functioning reconfigurable hardware in Reconfigurable Computing**

Embedded Design using FPGA

- Embedded systems are often composed of special-purpose hardware and software:
  - Hardware:
    - physical implementation of a computing machine
    - a collection of electronic circuits and perhaps some mechanical components
    - all of the hardware components are active concurrently
  - Software:
    - Is information and does not manifest itself in the physical world
    - Compiled from *Programs*:
      - ✓ a specification that describes the behavior of the machine
      - ✓ written in a programming language (such as C, MATLAB, or Java)
- FPGAs can blur the distinction between hardware and software
  - Hardware components are written like software that specifies how the FPGA device is to be configured.

Embedded Design using FPGA

What is difference between Hardware and Software in context of FPGA?

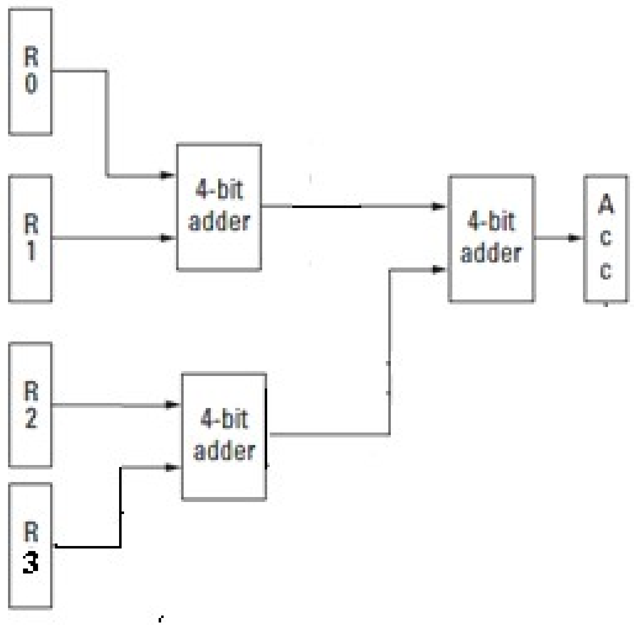
- **Software**
  - is characterized by a **sequential execution model**
  - refer to specifications intended to be **executed** by a processor (***where a processor is hardware that implements the sequential execution model***).
- **Hardware**
  - is characterized by a **parallel execution model**.
  - refer to specifications intended to **configure** the fabrics of an FPGA (***where a FPGA is the physical device that does not use the sequential execution model***).

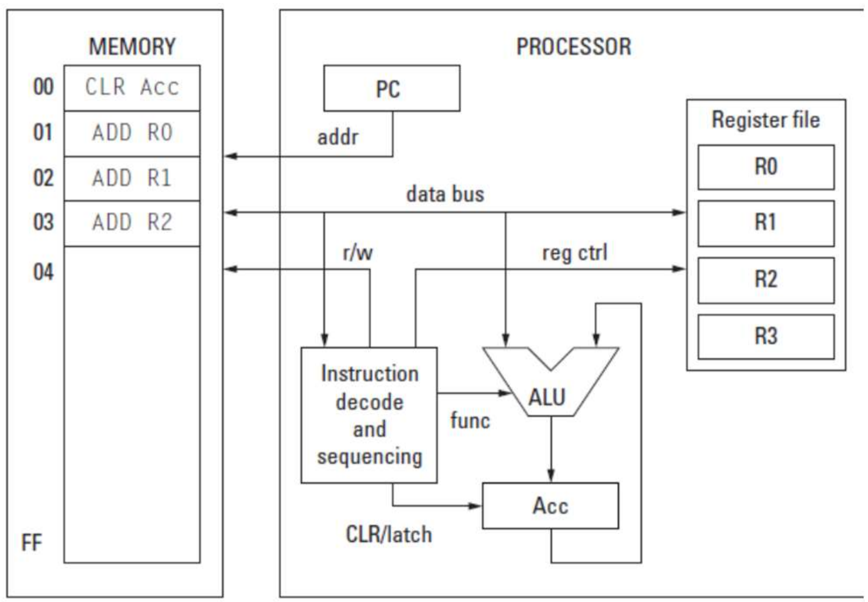
Embedded Design using FPGA

Difference between Hardware and Software from the execution model

Hardware	Software
Parallel/data-flow Spatial	Sequential Temporal

R0+R1+R2+R3→ Acc



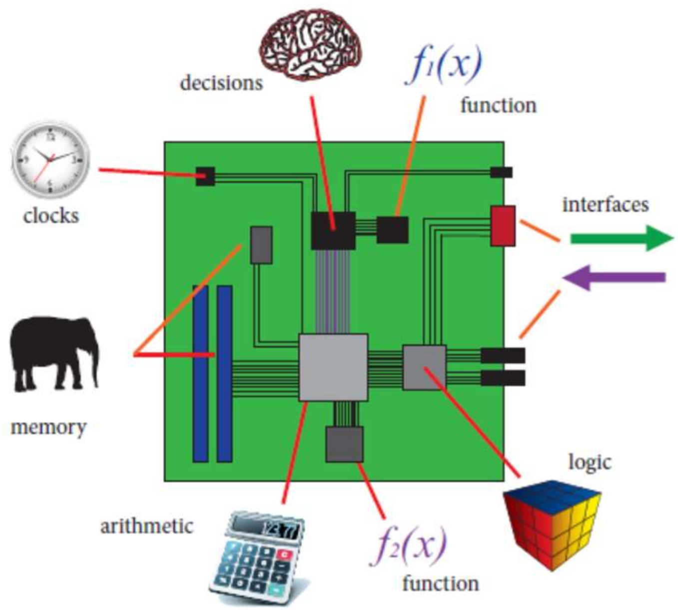


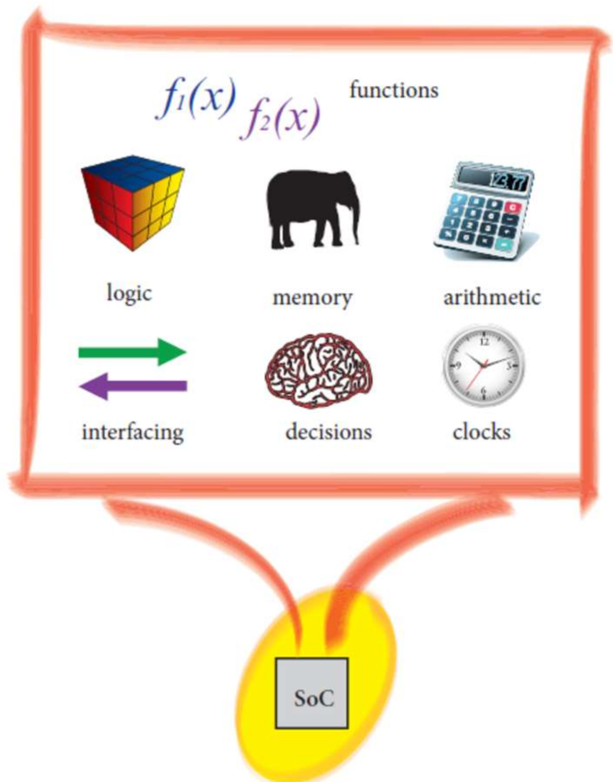
Embedded Design using FPGA

- Embedded design in an FPGA consists of the following:
  - Developing a computing system based on FPGA
    - Processor core:
      - Soft core: MicroBlaze processor
      - Hardcore: PowerPC
    - Peripherals: Timers, Interrupt controller, UART, GPIO, etc.
    - Bus interface
      - PLBv46 (XPS)
      - AXI interconnect
    - Reset, clocking, debug ports
  - Developing Standalone application or Use Operating System (OS) or Real Time Operating System (RTOS) (optional)
    - Xilinx Kernel: XilKernel
    - Linux Kernel
  - Generate drivers and libraries
  - Writing the software application
    - Software routines
    - Interrupt service routines (optional)

System-on-Chip Architecture

System-on-a-Board Vs. System-on-Chip





System-on-Chip:

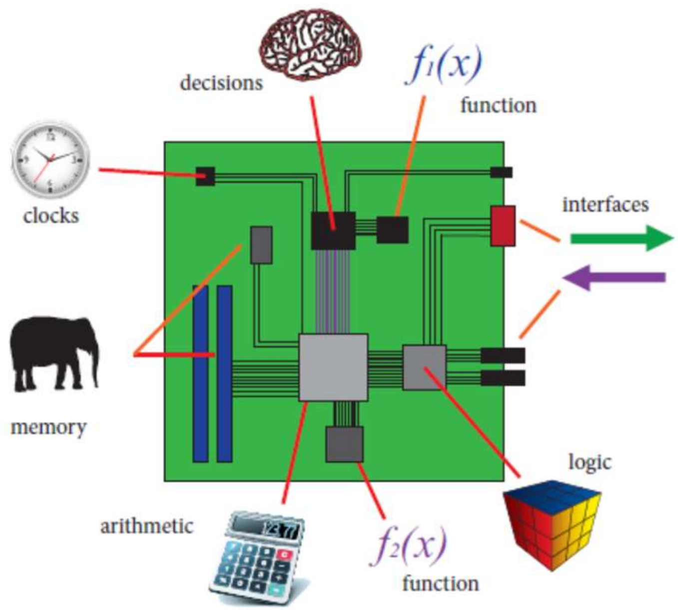
•The functionality of an entire system is implemented on a single silicon chip (area ~1 inch^2)

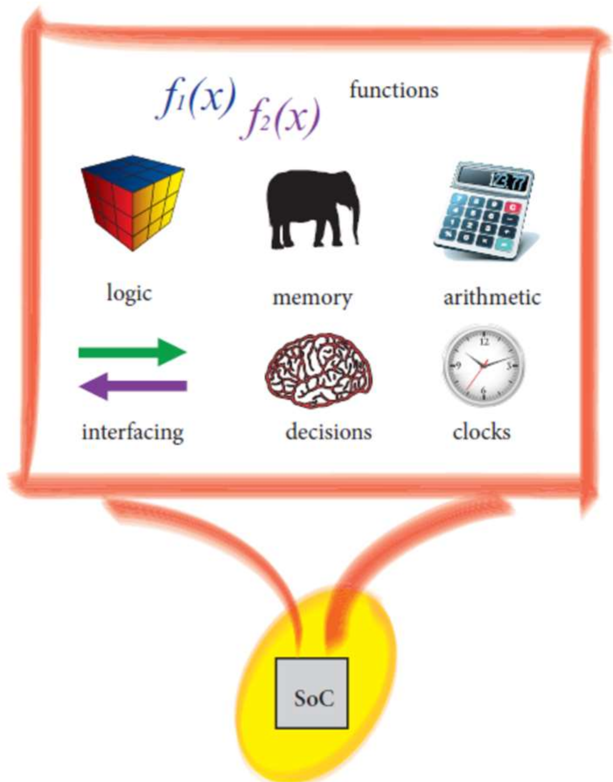
•SoC solution is lower cost, higher overall system speed, lower power consumption, smaller physical size, and better reliability.

https://dientuthongminh.blogspot.com/2018/03/he-thong-tich-hop-don-chip-soc.html

System-on-Chip Architecture

System-on-a-Board Vs. System-on-Chip





System-on-Chip:

•The functionality of an entire system is implemented on a single silicon chip (area ~1 inch^2)

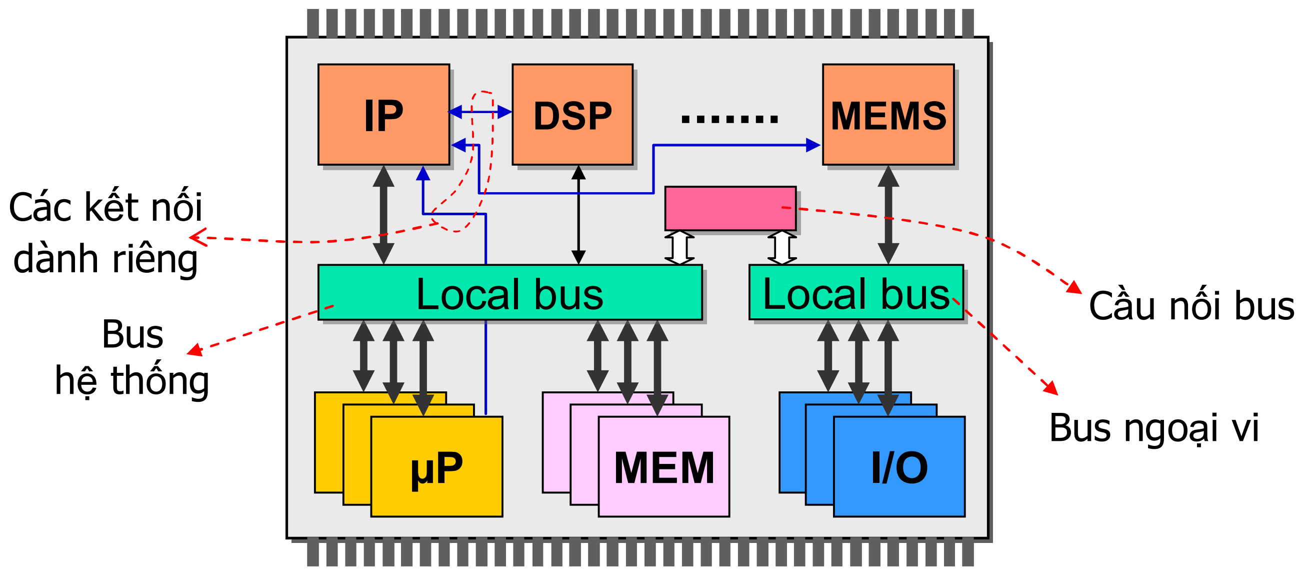
•SoC solution is lower cost, higher overall system speed, lower power consumption, smaller physical size, and better reliability.

https://dientuthongminh.blogspot.com/2018/03/he-thong-tich-hop-don-chip-soc.html

System-on-Chip Architecture

System-on-Chip can include:

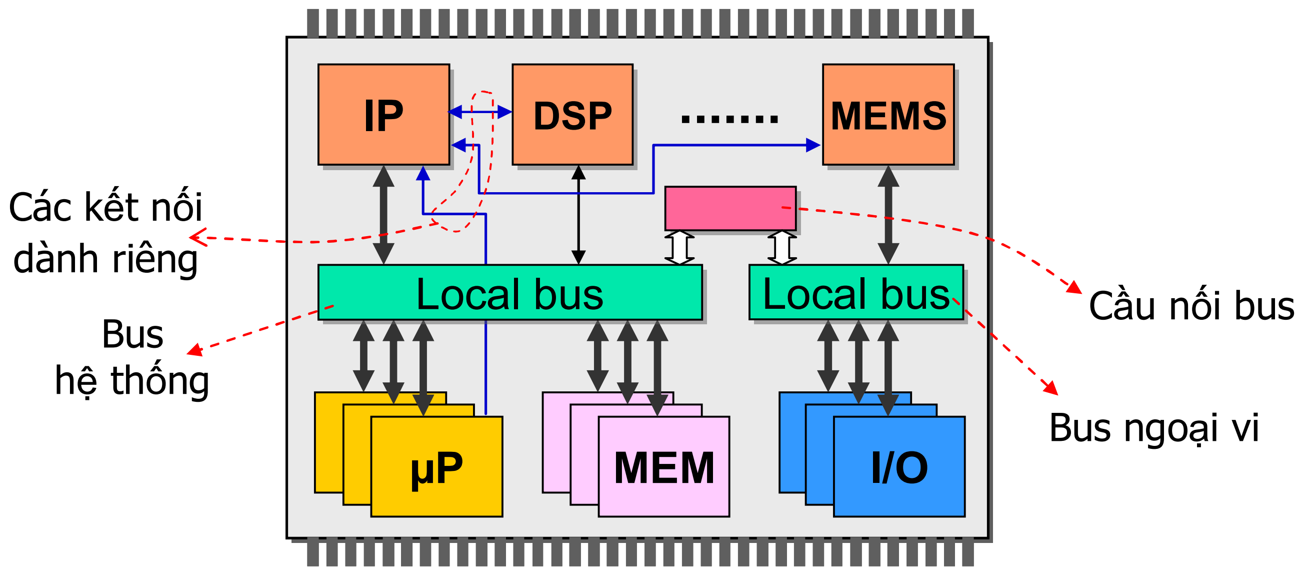
- Digital blocks: CPU (Central Processing Unit), memories (ROM, RAM), DSP (Digital Signal Processor), v.v...;
- I/O interfaces: Enthernet, Bluetooth, USB, USRT, GPIO;
- Analog Blocks and radio frequency components ;
- Mixed-signal blocks: ADC, DAC;
- Microelectromechanical systems (MEMS);
- v.v...



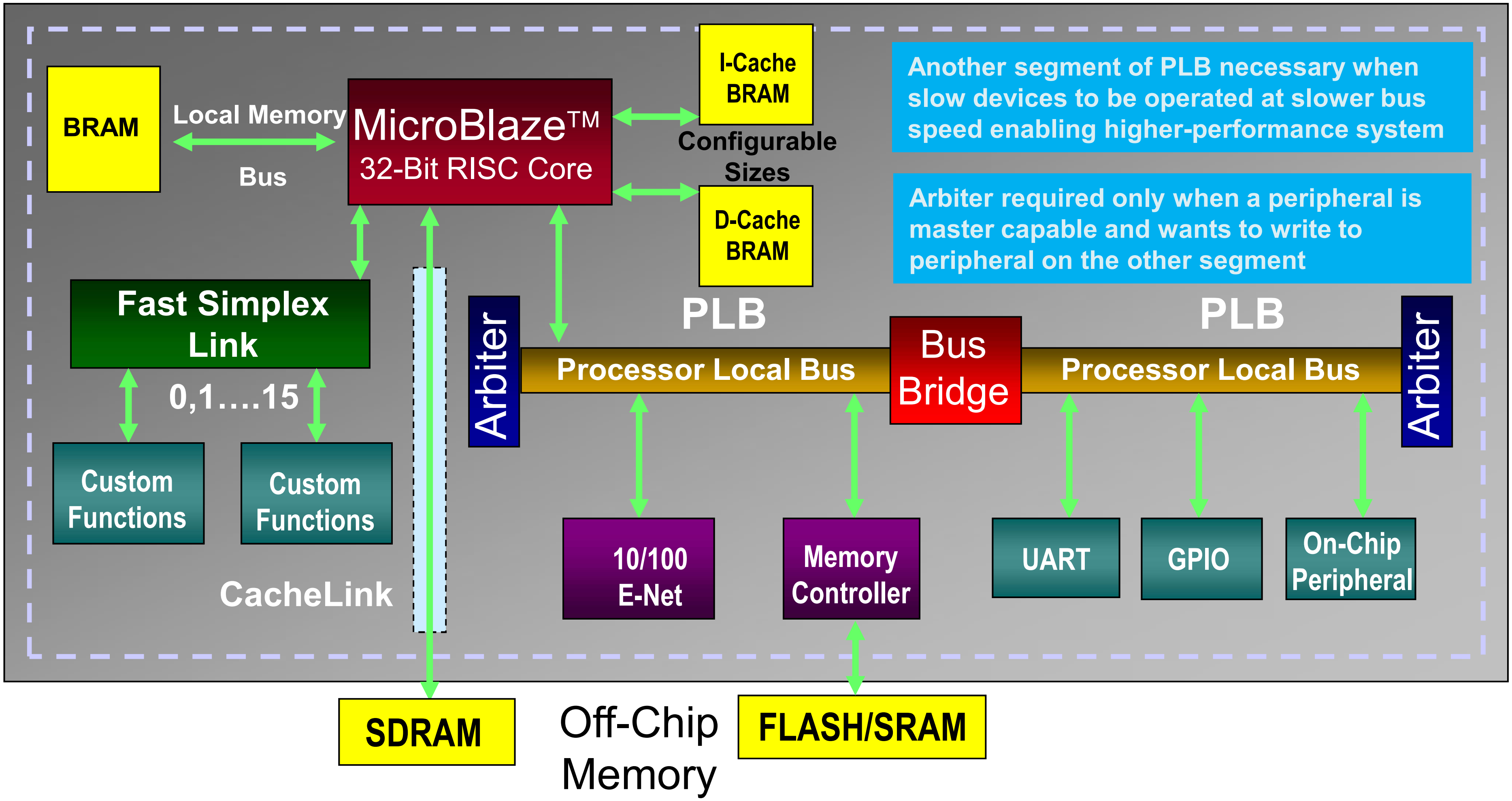
System-on-Chip Architecture

System-on-Chip can include:

- Digital blocks: CPU (Central Processing Unit), memories (ROM, RAM), DSP (Digital Signal Processor), v.v…;
- I/O interfaces: Enthernet, Bluetooth, USB, USRT, GPIO;
- Analog Blocks and radio frequency components ;
- Mixed-signal blocks: ADC, DAC;
- Microelectromechanical systems (MEMS);
- v.v…



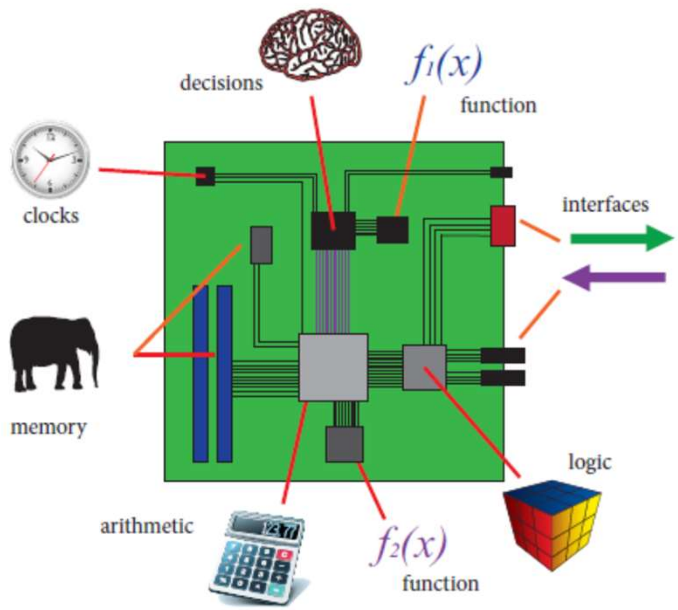
Embedded Design using FPGA

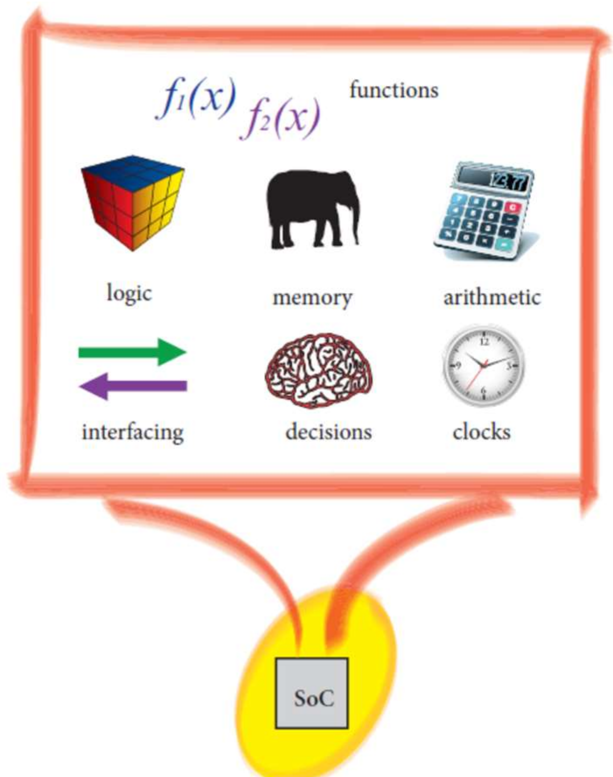


This is a v8.x architecture. Versions 7.x supported OPB bus off the core if PLB interface was disabled
Versions 6.0 or earlier did not support PLB bus off the processor. Instead they had OPB bus

System-on-Chip Architecture

System-on-a-Board Vs. System-on-Chip





System-on-Chip:

•The functionality of an entire system is implemented on a single silicon chip (area ~1 inch^2)

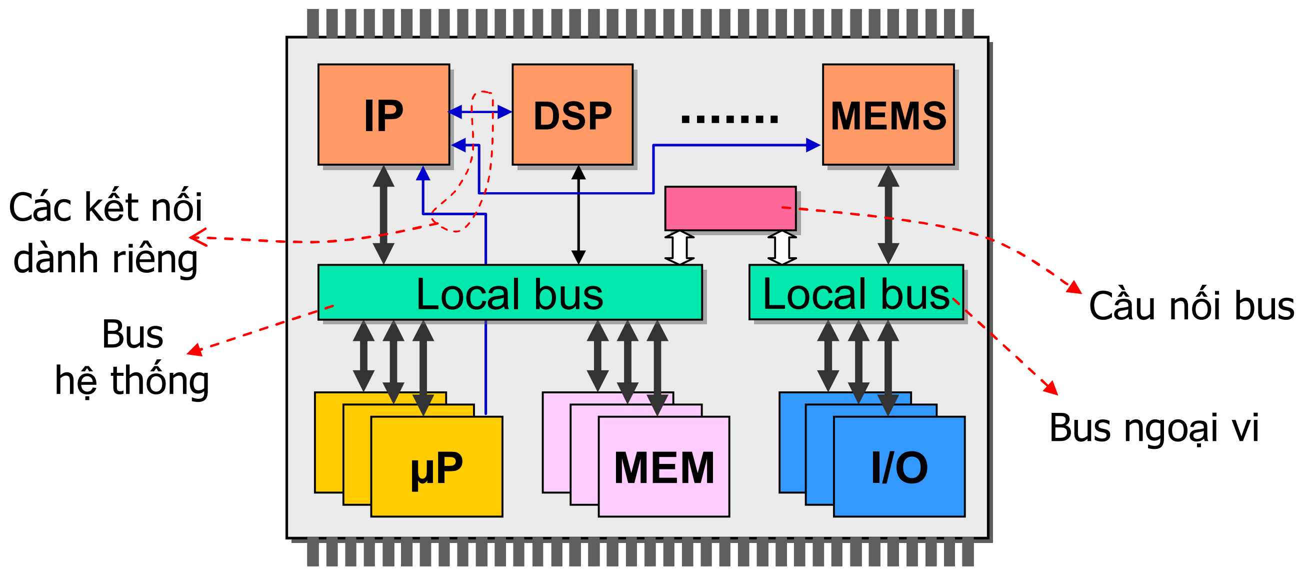
•SoC solution is lower cost, higher overall system speed, lower power consumption, smaller physical size, and better reliability.

https://dientuthongminh.blogspot.com/2018/03/he-thong-tich-hop-don-chip-soc.html

System-on-Chip Architecture

System-on-Chip can include:

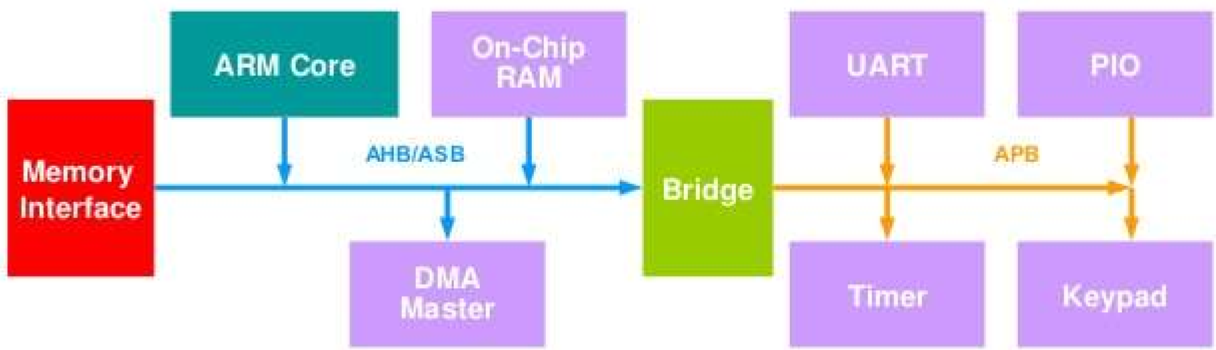
- Digital blocks: CPU (Central Processing Unit), memories (ROM, RAM), DSP (Digital Signal Processor), v.v…;
- I/O interfaces: Enthernet, Bluetooth, USB, USRT, GPIO;
- Analog Blocks and radio frequency components ;
- Mixed-signal blocks: ADC, DAC;
- Microelectromechanical systems (MEMS);
- v.v…



Embedded Design using FPGA

ARM On-Chip Bus

0xlab



A typical AMBA system

AHB: Advanced High-performance Bus

ASB: Advanced System Bus

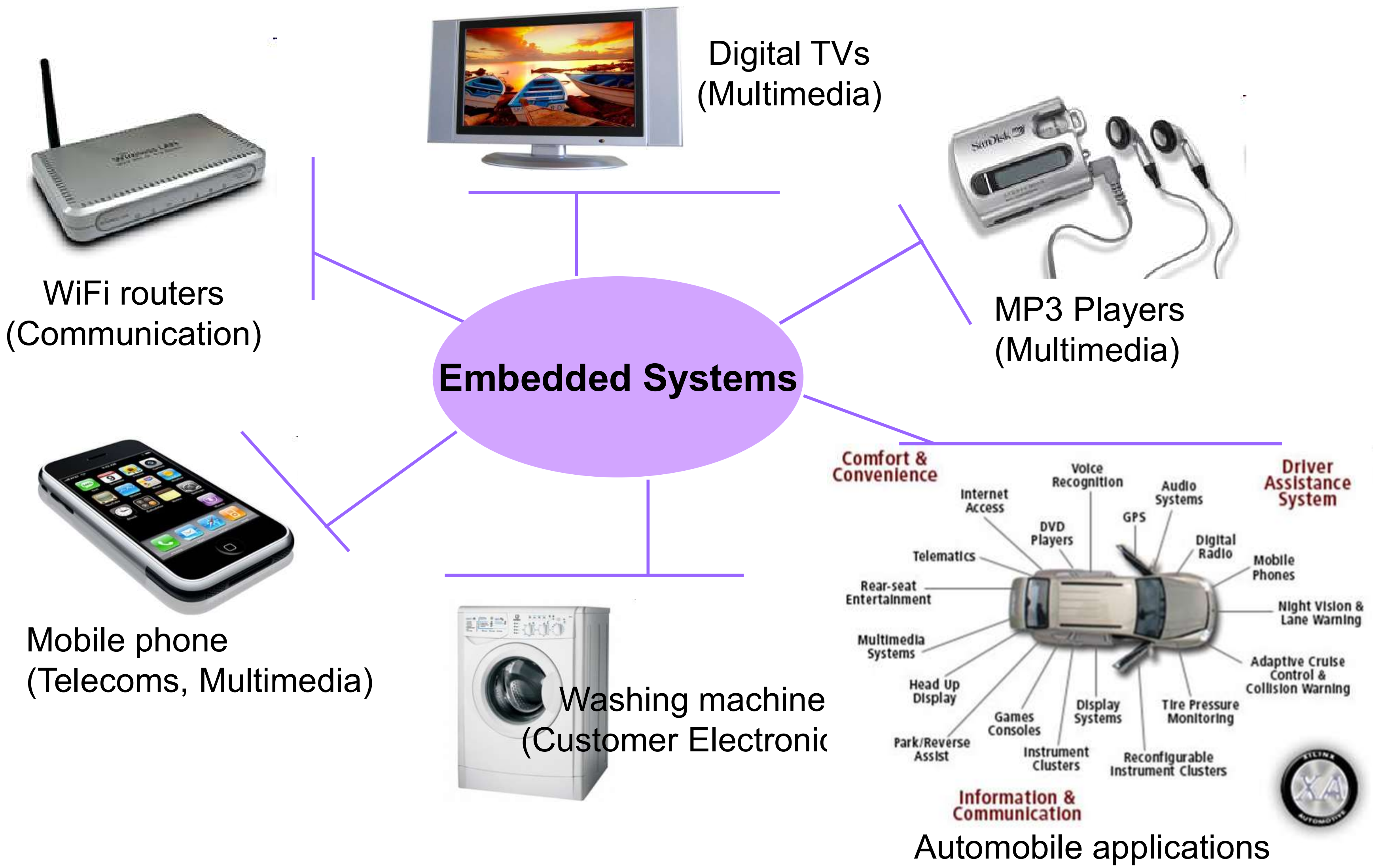
APB: Advanced Peripheral Bus

*** AMBA: Advanced Microcontroller Bus Architecture*

Outline

- Embedded Computing System
- Embedded System Design using FPGA
- ☞ **Characteristics of Embedded Computing Applications**
- Challenges in Embedded Computing System Design
- Embedded System Design Process
- Summary

Embedded Applications



Embedded Applications

Characteristics of Embedded Applications

- Sophisticated functionality:
  - *Complex algorithms*
  - *User interface*
- Performance Constrains:
  - *Real time*
  - *Multirate*
- Cost Requirements:
  - *Manufacturing cost*
  - *Power and energy*

Functionality is important,
But embedded applications must meet many other constraints as well.

Why we embed microprocessor in system?

- **Microprocessors are a very efficient way to implement digital systems:**
  - high-performance processors can execute several instructions per cycle
  - CPUs are highly optimized for speed by utilizing the latest manufacturing technology
  - A microprocessor, on the other hand, can be used for many different algorithms simply by changing the program it executes.
  - Design-time of your application is faster than designing your own custom logic
- **Microprocessors are flexible thank to their programmability**:
  - it is easier to design families of products
  - it is easier to provide new features to keep up with rapidly changing markets
  - it is possible to reuse software for next-generation products, thereby reducing development time and cost.

Outline

- Embedded Computing System
- Embedded System Design using FPGA
- Characteristics of Embedded Computing Applications
- **Challenges in Embedded Computing System Design**
- Embedded System Design Process
- Summary

Challenges in Embedded Computing System Design

- **Some important problems that must be taken into account in embedded system design:**
  - *How much hardware do we need?*
    - the choice of hardware must meet both ***performance deadlines*** and ***manufacturing cost constraints***
  - *How do we meet time constrains?*
    - to speed up the hardware so that the program runs faster
  - *How do we minimize power consumption?*
    - slowing down the system
  - *How do we design for upgradeability?*
    - The design must be enough flexible for several product generations, or for several different versions of a product in the same generation
  - *How does it really work?*
    - Reliability is always important when selling products

Challenges in Embedded Computing System Design

- **The factors makes embedded system designs more difficult**
  - *Complex testing*:
    - have to run a real machine in order to generate the proper data
  - *Limited observability and controllability:*
    - do not come with keyboards and screens
    - in real-time applications it is not easy to stop the system
  - *Restricted development environments:*
    - the tools used to develop software and hardware) are often much more limited

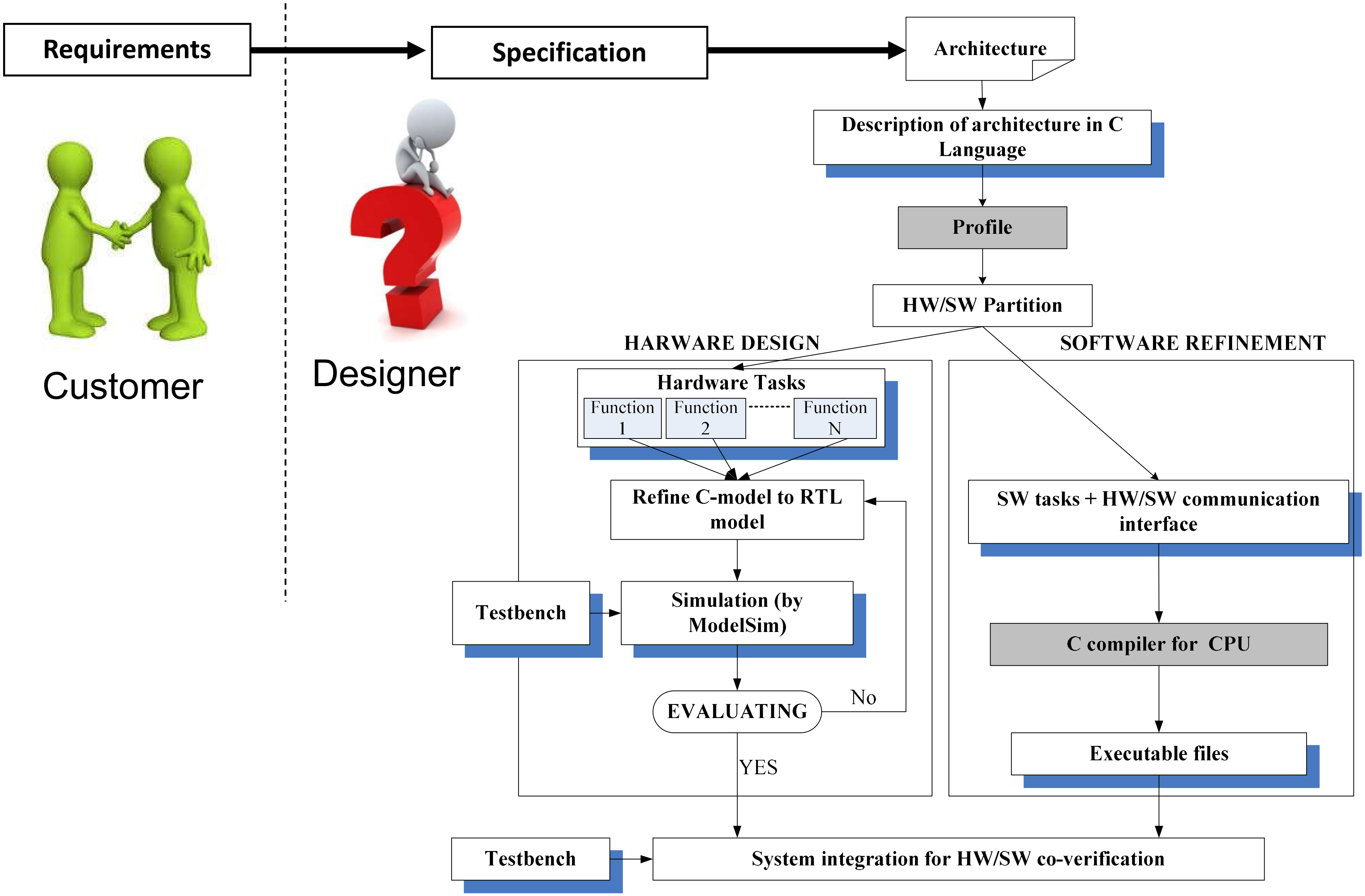
Outline

- Embedded Computing System
- Embedded System Design using FPGA
- Characteristics of Embedded Computing Applications
- Challenges in Embedded Computing System Design
- **Embedded System Design Process**
- Summary

Design Methodology

- **What?**
  - **A procedure for designing a system by breaking the process into manageable steps.**
- **Why?**
  - **Understanding your methodology helps you ensure you didn’t skip anything.**
  - **Compilers, software engineering tools, computer-aided design (CAD) tools, etc., can be used to:**
    - help automate methodology steps;
    - keep track of the methodology itself.
  - **A design methodology makes it much easier for members of a design team to communicate**

Embedded System Design Process



Design goals

- Functionality and user interface.
- Performance.
  - Overall speed, deadlines.
- Manufacturing cost.
- Power consumption.
- Other requirements (physical size, etc.)

Levels of abstraction

Requirements
(Description of what we want)

Specification
(How the system behaves)

Architecture
(System structure in terms of function blocks)

Hardware/Software Design
(Software modules and specialized hardware)

System Integration
(Building a complete system)

Hardware/Software Co-verification
(Debugging, Performance and Power evaluation)

Requirements

- *An informal description from the customers:*
  - Idea about what the user wants and expects to get
- *Functional requirements:*
  - output as a function of input.
- *Nonfunctional requirements*:
  - **Performance:** The speed of the system to meet a certain time constrain.
  - **Cost:** includes manufacturing cost and nonrecurring engineering (NRE) costs
  - **Physical size and weight:** are tight requirements for a handheld device
  - **Power consumption:** is important in battery-powered systems

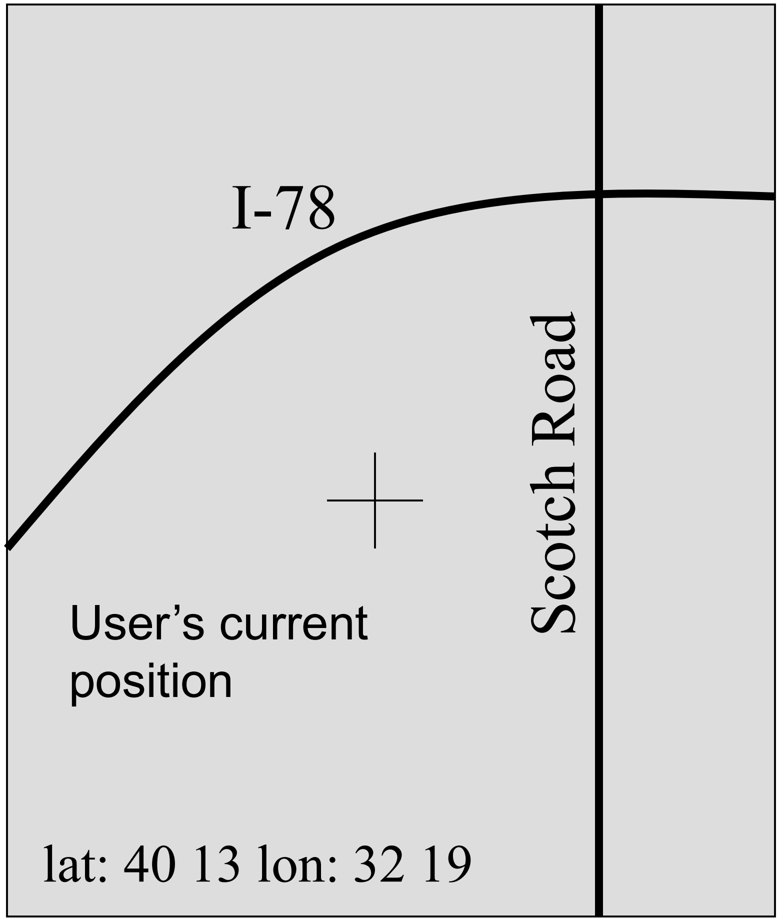
Requirements

Building a requirements form

Name	Name of the project
Purpose	A brief one- or two-line description of what the system is supposed to do
Inputs/Outputs	- Types of data: Analog electronic signals? Digital data? - Data characteristics: Periodically arriving data? How many bits per data element? - Types of I/O devices: Buttons? Analog/digital converters? Video displays?
Functions	A more detailed description of what the system does
Performance	The speed of the system to meet a certain time constrain
Manufacturing cost	cost of the hardware components and assembly
Power	How much power the system can consume? how much power the system can consume?
Physical size and weight	The physical size of the system

Example: GPS moving map requirements

- a handheld device that displays for the user a map of the terrain around the user's current position based on the information from GPS;
- the map display changes as the user and the map device change position.



Example: GPS moving map requirements

- ***Functionality:*** Designed for automotive use. Show major roads and other landmarks available.
- ***User interface*:** At least 400 x 600 pixel screen. Three buttons max. Pop-up menu.
- ***Performance:*** Map should scroll smoothly. No more than 1 sec power-up. Lock onto GPS within 15 seconds.
- ***Cost:*** The selling cost of the unit $\leq$ \$120.
- ***Physical size and weight:*** should fit comfortably in the hand.
- ***Power consumption:*** should run for at least eight hours on four AA batteries.

Example: GPS moving map requirements

Requirements form for a GPS moving map system

Name	GPS moving map
Purpose	Consumer-grade moving map for driving use
Inputs	Power button, two control buttons
Outputs	Back-lit LCD display 400 $\times$ 600
Functions	Uses 5-receiver GPS system; three user-selectable resolutions; always displays current latitude and longitude
Performance	Updates screen within 0.25 seconds upon movement
Manufacturing cost	$40
Power	100 mW
Physical size and weight	No more than 2″ $\times$ 6″, 12 ounces

Specification

- **A more precise description of the system**
  - refined from the customer's requirements
  - states only how the system behaves,
  - should not imply a particular architecture;
  - provides input to the architecture design process.
- **should be understandable enough so that someone can verify that it meets system requirements of the customer**
- **should also be unambiguous enough that designers know what they need to build.**
- **May be executable or may be in mathematical form for proofs.**

Specification

Example: **A specification of the GPS Moving Map would include several details:**

- What is data received from the GPS;
- map data;
- user interface;
- operations that must be performed to satisfy customer requests;
- Background actions required to keep the system running, such as operating the GPS receiver.

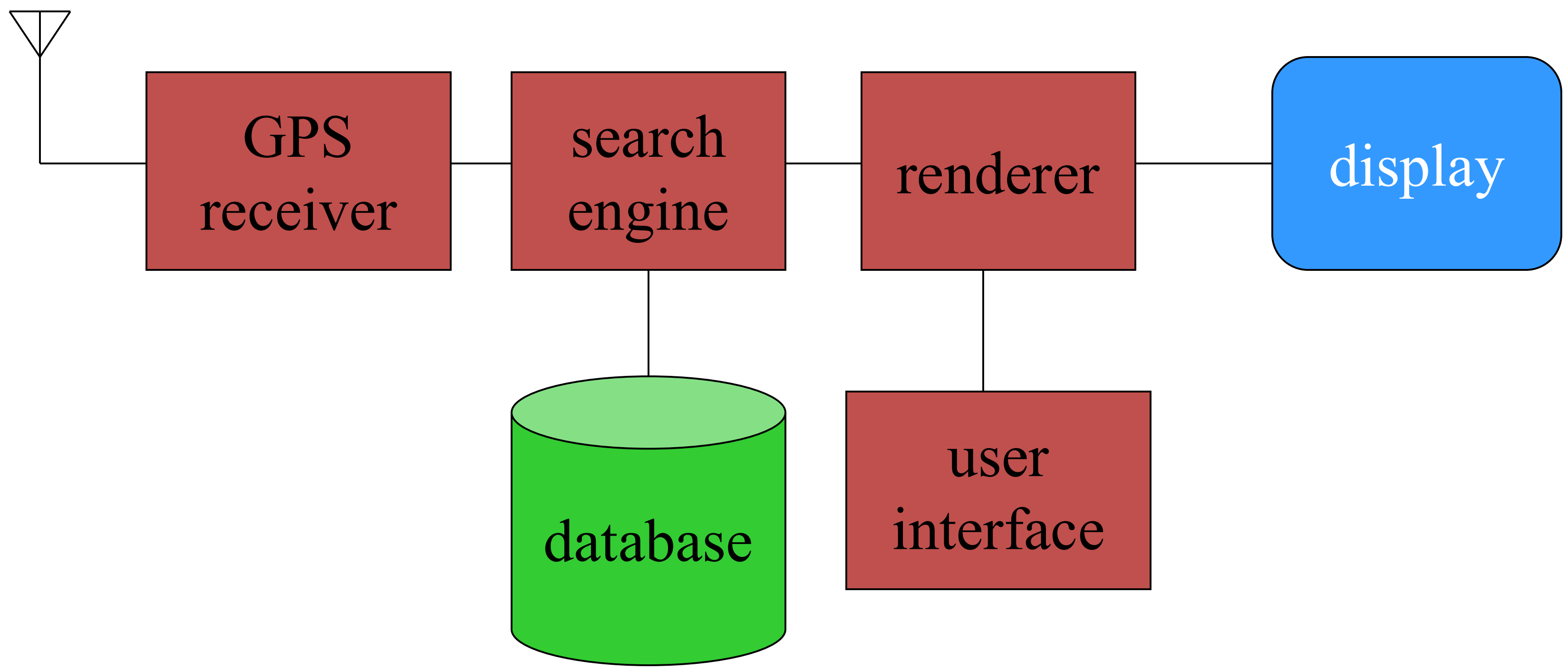
Architecture Design & HW/SW Partition

- **Architecture**
  - describes how the system is implemented to satisfy the specification?
  - represented in the form of a block diagram that **shows major operations and data flows among them**
- **HW/SW Partition** specifies which operations will be performed by software running on a CPU, what will be done by special-purpose hardware
  - *Hardware components*: such as CPUs, peripherals, ICs, Hardwired IPs, etc.
  - *Software components*: major programs and their operations.
- Must take into account functional and non-functional specifications.
  - The functional specifications $\rightarrow$ the system block diagram
  - The non-functional specifications $\rightarrow$ HW/SW Partition

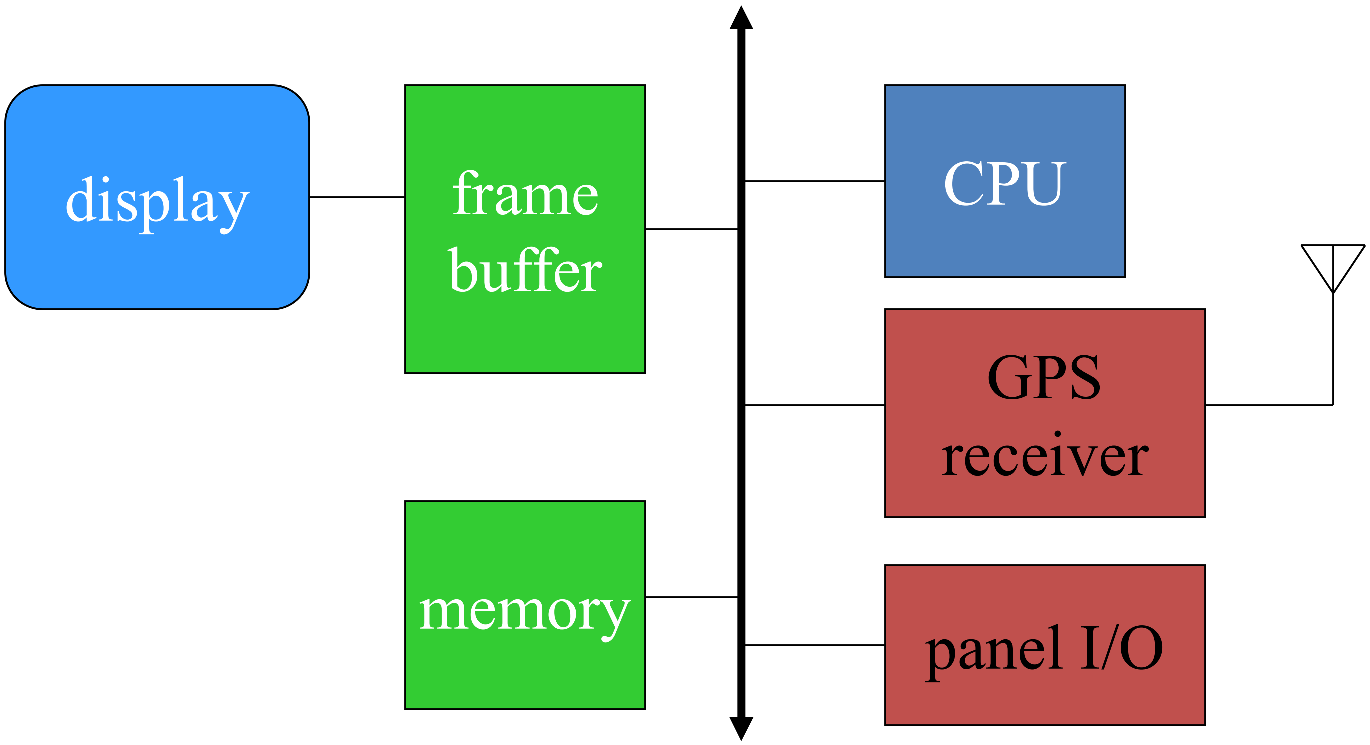
Architecture Design & HW/SW Partition

- One of the main partitioning criteria is how fast you wish the various functions to perform their tasks:
  - *Picosecond and nanosecond logic: is* implemented in hardware
  - *Microsecond logic*: can be implemented either in hardware or software
  - *Millisecond logic:* is implemented by the microprocessor code

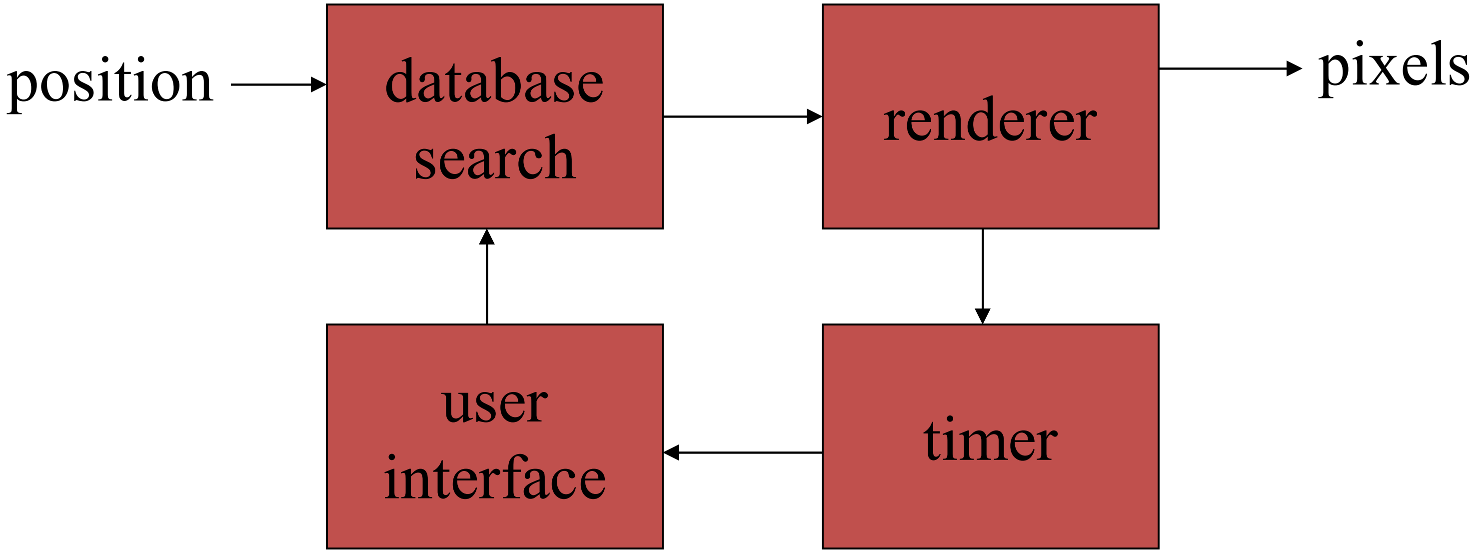
GPS moving map block diagram



GPS moving map hardware architecture



GPS moving map software architecture



Designing Hardware and Software Components

- Hardware and Software components are built according to the architecture and specifications
  - **Hardware components**:
    - ready-made standard components: CPU, memory chips, etc.
    - Special-purpose component (e.g. GPS receiver): designed using FPGAs, ASIC
    - Printed circuit board (PCB):
  - **Software components**
    - *Standard software modules*: standard routines or API to access the topographic database.
    - *Special-purpose software modules*: data decompression routines
    - *Peripheral Drivers*: driver for GPS receiver
    - *Scheduler*: where units in the software block diagram will be executed in the hardware block diagram and when operations will be performed in time.

System Integration and HW/SW co-verification

- Putting them together and testing whether the system works well or not.
- Debugging:
  - How to find bugs during system integration and how they can be fixed?
- Performance and power evaluation

Outline

- Embedded Computing System
- Embedded System Design using FPGA
- Characteristics of Embedded Computing Applications
- Challenges in Embedded Computing System Design
- Embedded System Design Process
- **Summary**

Summary

- **What we Learned**
  - Basic concepts of Embedded Computing Systems
  - Distinguish between embedded Computing Systems and general-purpose computing Systems
  - Hardware, software concept in Embedded Computing Systems design using FPGA
  - The embedded system design process and different abstract levels for describing Embedded Computing Systems

Review Questions

Q1: What is an embedded computer system? What is different and unique about embedding computing and general-purpose computer?

Q2: Name three consumer electronics products that have embedded systems. Name three consumer electronics products that do not contain embedded computer systems.

Q3: Briefly describe the distinction between requirements and specification.

Q4: Briefly describe the distinction between specification and architecture.

Q5: Briefly describe the distinction between specification and architecture.

Q6: At what stage of the design methodology would we determine what type of CPU to use (8-bit vs. 16-bit vs. 32-bit, which model of a particular type of CPU, etc.)?

Review Questions

Q7: At what stage of the design methodology would we choose a programming language?

Q8: At what stage of the design methodology would we test our design for functional correctness?

Q9: Explain the difference between hardware and software in terms of a traditional processor-based system.

Q10: How does the “hardware” of a Platform FPGA-based system differ from the hardware of a traditional processor-based system?

Q11: What is the difference between a soft IP core and a hard IP core in a Platform FPGA system?