Embedded Software development Process and Tools:

Lesson-6

ISSUES IN HARDWARE SOFTWARE DESIGN AND CO-DESIGN

1. Embedded system design

Two approaches for the embedded system design device programmer

- (1) When the software development cycle ends then the cycle begins for the process of integrating the software into the hardware at the time when a system is designed.
- (2) Both cycles concurrently proceed when co-designing a time critical sophisticated system

2. Hardware Software Trade-off

Software Hardware Tradeoff

- It is possible that certain subsystems in hardware (microcontroller), IO memory accesses, real-time clock, system clock, pulse width modulation, timer and serial communication are also implemented by the software.

Software Hardware Tradeoff

- A serial communication real-time clock and timers featuring microcontroller may cost more than the microprocessor with external memory and a software implementation.
- Hardware implementations provide advantage of processing speed

Hardware implementation advantages

- (*i*) Reduced memory for the program.
- (*ii*) Reduced number of chips but at an increased cost.
- (*iii*) Simple coding for the device drivers.

Hardware implementation advantages…

- (*iv*) Internally embedded codes, which are more secure than at the external ROM
- (v) Energy dissipation can be controlled by controlling the clock rate and voltage

Software implementation advantages

- (*i*) Easier to change when new hardware versions become available.
- (*ii*) Programmability for complex operations.
- (*iii*) Faster development time.
- (*iv*) Modularity and portability.

Software implementation advantages…

- (*v*) Use of standard software engineering, modeling and RTOS tools.
- *(vi)* Faster speed of operation of complex functions with high-speed microprocessors.
- (vii) Less cost for simple systems.

2. Choosing the right platform

Units to be choosen

- Processor ASIP or ASSP
- Multiple processors
- System-on-Chip
- Memory
- Other Hardware Units of System
- Buses

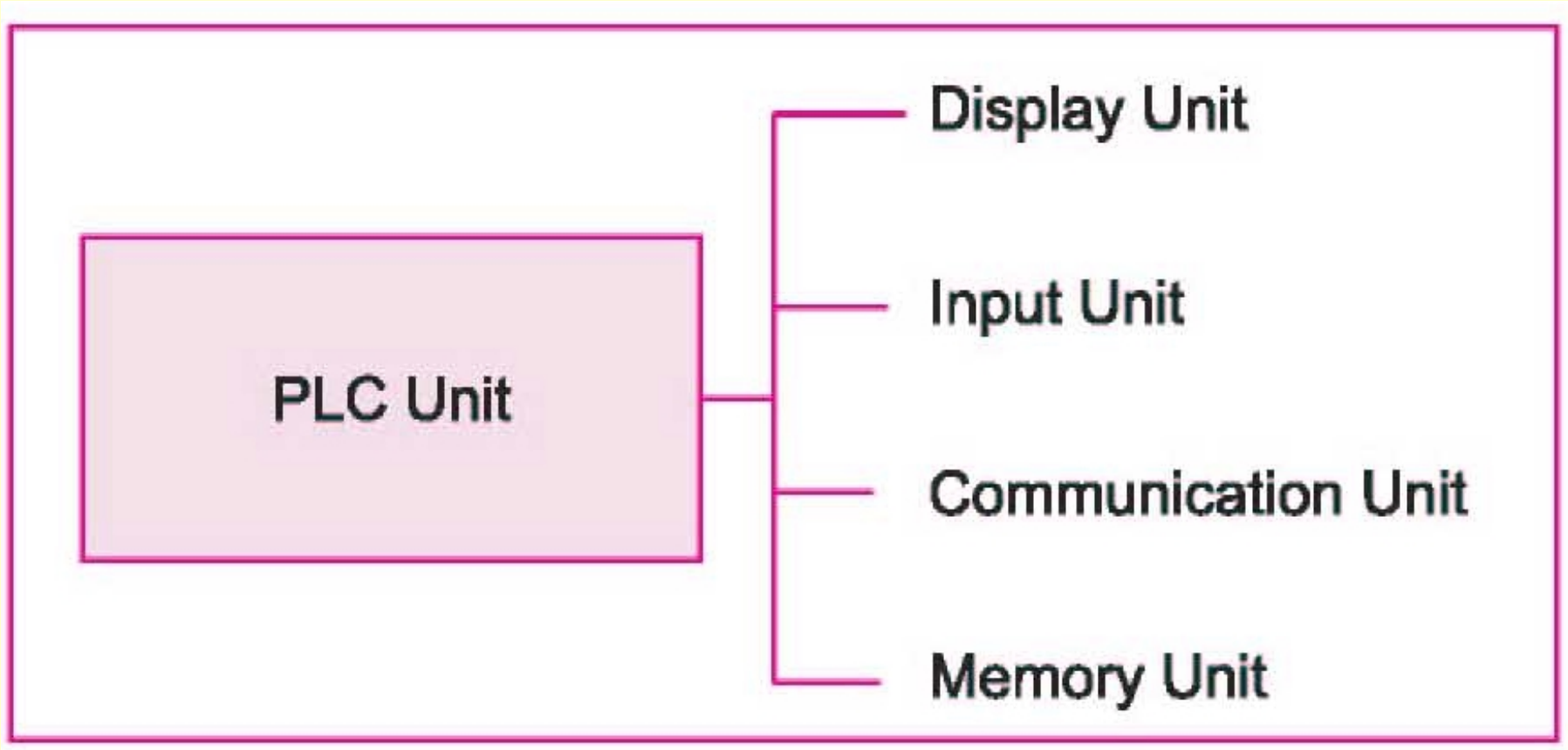
Units to be chosen

- Software Language
- RTOS (real-time programming OS)
- Code generation tools
- Tools for finally embedding the software into binary image

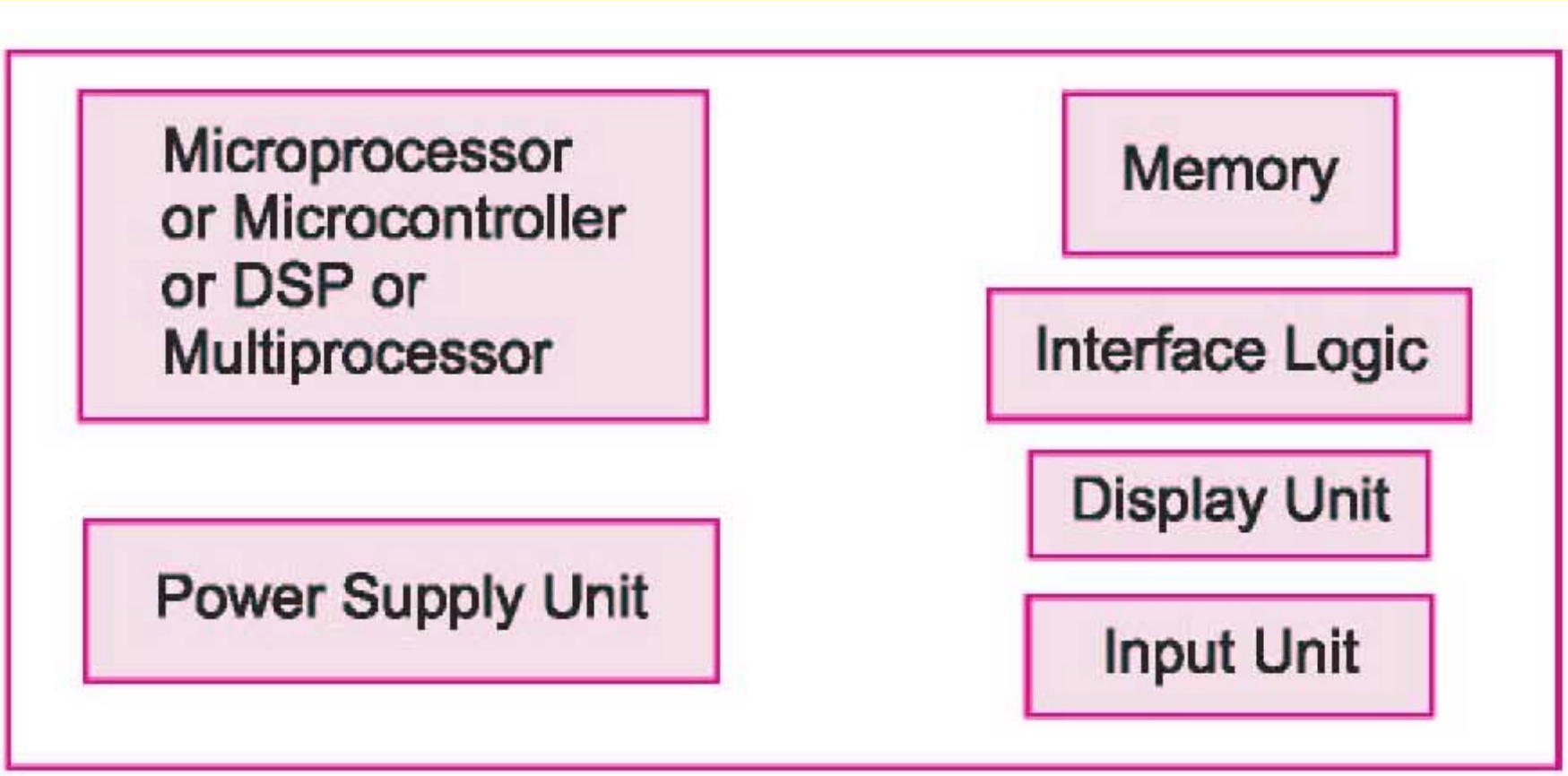
Embedded System Processors Choice

- Processor Less System
- System with Microprocessor or Microcontroller or DSP
- System with Single purpose processor or ASSP in VLSI or FPGA

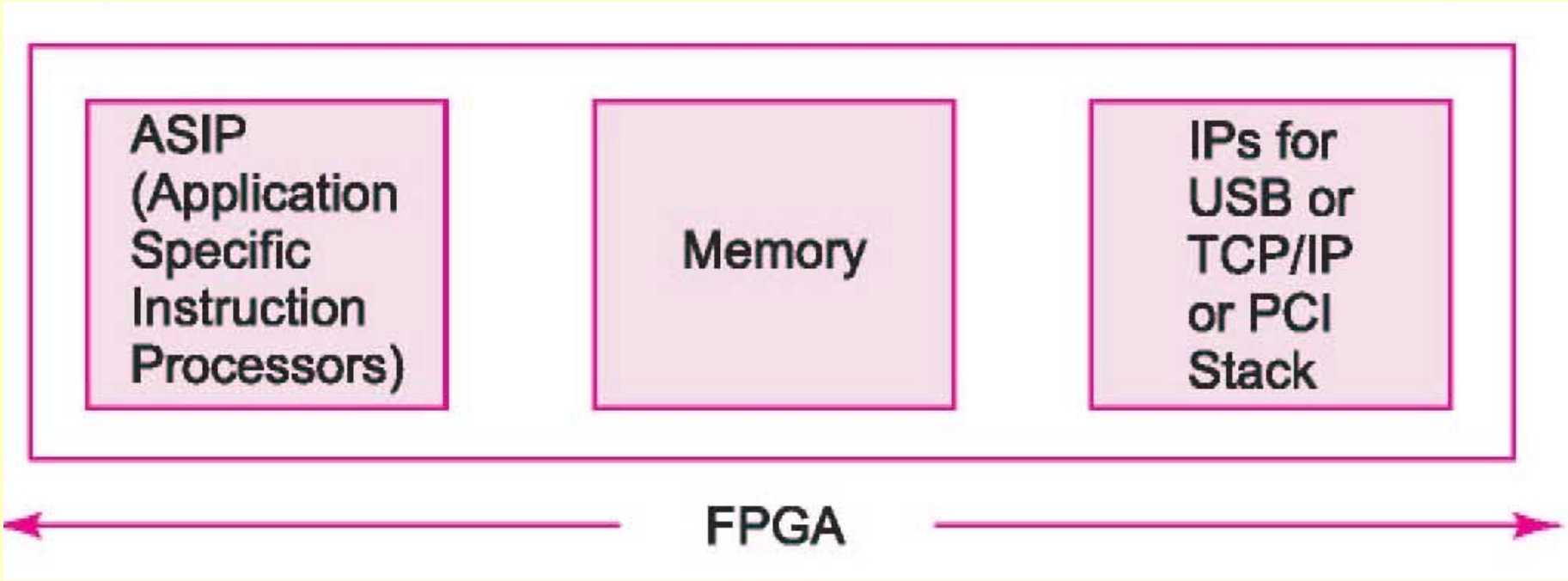
Processor Less System



Microprocessor or Microcontroller or DSP



Processing of functions by using IP embedded into the FPGA instead of processing by the ALU



Factors and Needed Features Taken into Consideration

- When the 32-bit system, 16kB+ on chip memory and need of cache, memory management unit or SIMD or MIMD or DSP instructions arise, we use a microprocessor or DSP.
- Video game, voice recognition and image-filtering systems– need a DSP.

Factors and Needed Features Taken into Consideration…

- Microcontroller provides the advantage of on-chip memories and subsystems like the timers.

Factors for On-Chip Feature

- 8 bit or 16 bit or 32 bit ALU
- Cache,
- Memory Management Unit or
- DSP calculationsI
- Intensive computations at fast rate
- Total external and internal Memory up to or more than 64 kB

Factors for On-Chip Feature…

- Internal RAM
- Internal ROM/EPROM/EEPROM
- Flash
- Timer 1, 2 or 3
- Watchdog Timer
- Serial Peripheral Interface Full duplex
- Serial Synchronous Communication Interface (SI) Half Duplex

Factors for On-Chip Feature…

- Serial UART
- Input Captures and Out-compares
- PWM
- Single or multi-channel ADC with or without programmable Voltage reference (single or dual reference)
- DMA Controller
- Power Dissipation

3. Hardware Sensitive Programming

Processor Sensitive

- Can have memory mapped IOs or IO mapped IOs.
- IO instructions are processor sensitive.
- Fixed point ALU only or Floating-point operations preetn
- Provision for execution of SIMD (single instruction multiple data) and VLIW (very large instruction word) instructions.

Processor Sensitive

- Programming of the modules needing SIMD and VLIW instructions is handled differently in different processors.
- Assembly language– sometimes facilitate an optimal use of the processor's special features and instructions.
- Advanced processors– usually provide the compiler or optimizing compiler sub-unit to obviate need for programming in assembly.

Memory Sensitive

- Real-time programming model and algorithm used by a programmer depend on memory available and processor performance.
- Memory address of IO device registers, buffers, control-registers and vector addresses for the interrupt sources or source groups are prefixed in a microcontroller.

Memory Sensitive

- Programming for these takes into account the addresses.
- Same addresses must be allotted for these by the RTOS.
- Memory-sensitive programs need to be optimized for the memory use by skillful programming for example, ARM Thumb® instruction set use

ALLOCATION OF ADDRESSES TO MEMORY

- Program segments and device addresses
- Different sets and different structures of data at the memory
- Device, Internal Devices and I/O devices Addresses and Device Drivers

4. Porting issues in an Embedded Platform

Porting Issues

- Porting issues of OS
- I/O instructions
- Interrupt Servicing Routines
- Data types
- Interface specific data types

Porting Issues

- Byte order
- Data Alignment
- Linked Lists
- Memory Page Size
- Time Intervals

5. Performance Metrics

Performance Metrics

- Performance Modeling for metric
- System Performance Index as Performance Metric
- Multiprocessor system performance as Performance Metric
- MFLOPs and DMIPS (Dhrystone/s) as Performance Indices as Performance Metrics
- Buffer Requirement, IO performance and Bandwidth Requirement as Performance Metrics

Performance Metrics

- Real time program performance as Performance Metric

6. Performance Accelerators

Performance Accelerators

- Conversion of CDFGs into DFGs for example by using loop flattening (loops are converted to straight program flows) and using look-up tables instead of control condition tests to decide a program flow path

Performance Accelerators

- Reusing the used arrays in memory, appropriate variable selection, appropriate memory allocation and de-allocation strategy
- Using stacks as data structure when feasible in-place of queue and using queue in place of list, whenever feasible.

Performance Accelerators

- Computing slowest cycle first and examining the possibilities of its speed-up.
- Code such that more words are fetched from ROM as a byte than the multi-byte words

Summary

We learnt

- Selection of right hardware during hardware design critical especially for a sophisticated embedded system development
- Hardware-Software trade-off
- Processor and memory sensitive instructions
- Choosing right processor, memory, IOs, devices

We learnt

- Ways of measuring system performance.
- System performance as per required and agreed specifications, power dissipation, throughputs, IO throughputs, response time of tasks, deadline misses, response to sporadic tasks, memory buffers, bandwidth requirements and memory optimization

We learnt

- Performance acceleration

End of Lesson-6 of chapter 13 on
ISSUES IN HARDWARE SOFTWARE DESIGN AND CO-DESIGN