Faculty of Computers & Information
Fourth Year. CS department
Embedded Systems
Instructor: **Dr. Anas Youssef**

Midterm Exam
Time: **1 hour**
Number of Pages: **2**
Total marks: **20**

Student Name	
Student Section No.	

Question 1: **(8 Marks)**

(a) What does **ASIC** stand for in an embedded system architecture?

(b) Briefly describe the difference between a **critical hard-realtime** system and a **non-critical hard-realtime** system. State **one** example for **each** of them.

(c) Draw a simple diagram that shows the behavioral specification of **an embedded system that controls an ATM machine** with at **least four different states.**

(d) Write a sample **pseudocode** that implements a **power-saving super loop** for a sequence of tasks that are performed in an embedded system.

Question 2: **(12 Marks)**

(a) Briefly describe the purpose of having a **Realtime Kernel** in building an embedded system.

(b) Draw a diagram that shows a **hardware/software co-design methodology** for an embedded system.

(c) **Briefly** describe difference between the **priority inversion** problem and the **unbounded priority inversion** problem that may occur in processor scheduling of tasks in **real time embedded systems**. Which of the two problems has more critical effect on missing task deadlines? **Justify with reasons.**

(d) Consider the following set of periodic real-time tasks with the following execution profiles: **Task T1** execution time is **50 ms** and execution rate is **10 hz, Task T2** execution time is **100 ms** and execution rate is **8 hz** and **Task T3** execution time is **150 ms** and execution rate is **5 hz.**

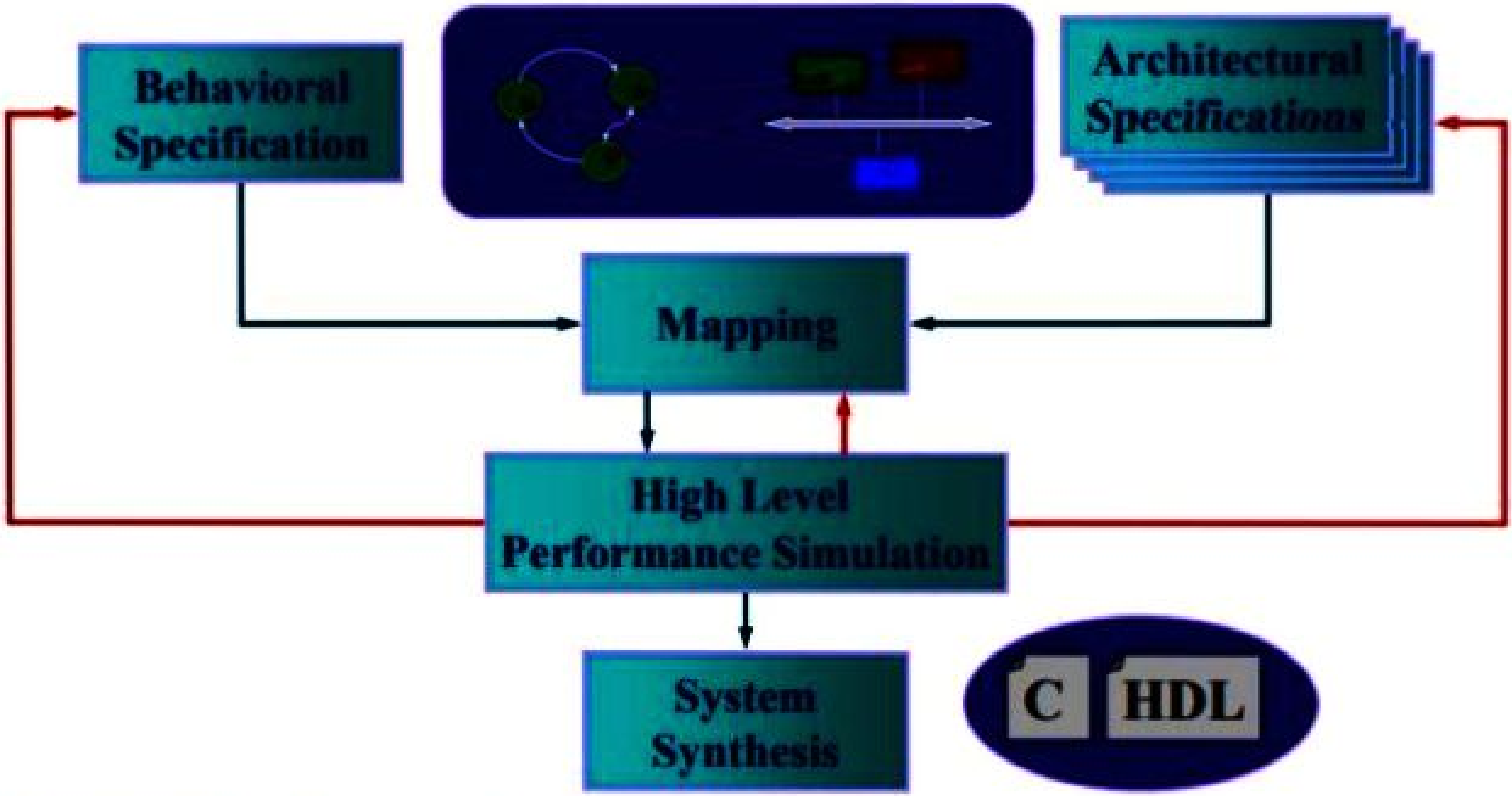
(i) Can the three tasks be successfully scheduled using **perfect scheduling**? **Why**?

(ii) Suppose that the first instance of each of the **three** tasks arrives at time $t = 0$. Assume that the **deadline** for each task **is less than its corresponding execution period by 10%.** Draw a timing diagram that uses **rate monotonic scheduling** to show the steps of task scheduling over time. Will **all** the tasks' deadlines be met or not? **Why**?

– **ASICs** application-specific integrated circuit

1. **Critical Hard Real-Time:**
   - These are tasks or processes within an embedded system that have strict, non-negotiable timing requirements.
   - Failure to meet the specified deadlines can lead to catastrophic consequences, safety hazards, or system failure.
   - Examples include control systems in automotive applications (like ABS braking), medical devices (like pacemakers), or avionics in aircraft (like flight control systems).
   - These tasks are time-critical and must be completed within specified time limits for the system to operate safely and correctly.
2. **Non-Critical Hard Real-Time:**
   - These are tasks within an embedded system that have timing requirements, but missing a deadline doesn't result in catastrophic failure or safety hazards.
   - While timing is important, missing a deadline might lead to degraded performance or reduced efficiency rather than a system failure.
   - Examples might include user interface updates, background maintenance tasks, or some communication processes in the system.
   - Though these tasks have timing constraints, they're not as critical as those in the critical hard real-time category.

Embedded System Design Behavior/Architecture Co-Design Methodology



Behavioral Specification and Architectural Specification -> Mapping -> High Level Performance Simulation with the ability of returning back modifying specification -> System Synthesis (بناء السيستم كهاردوير)

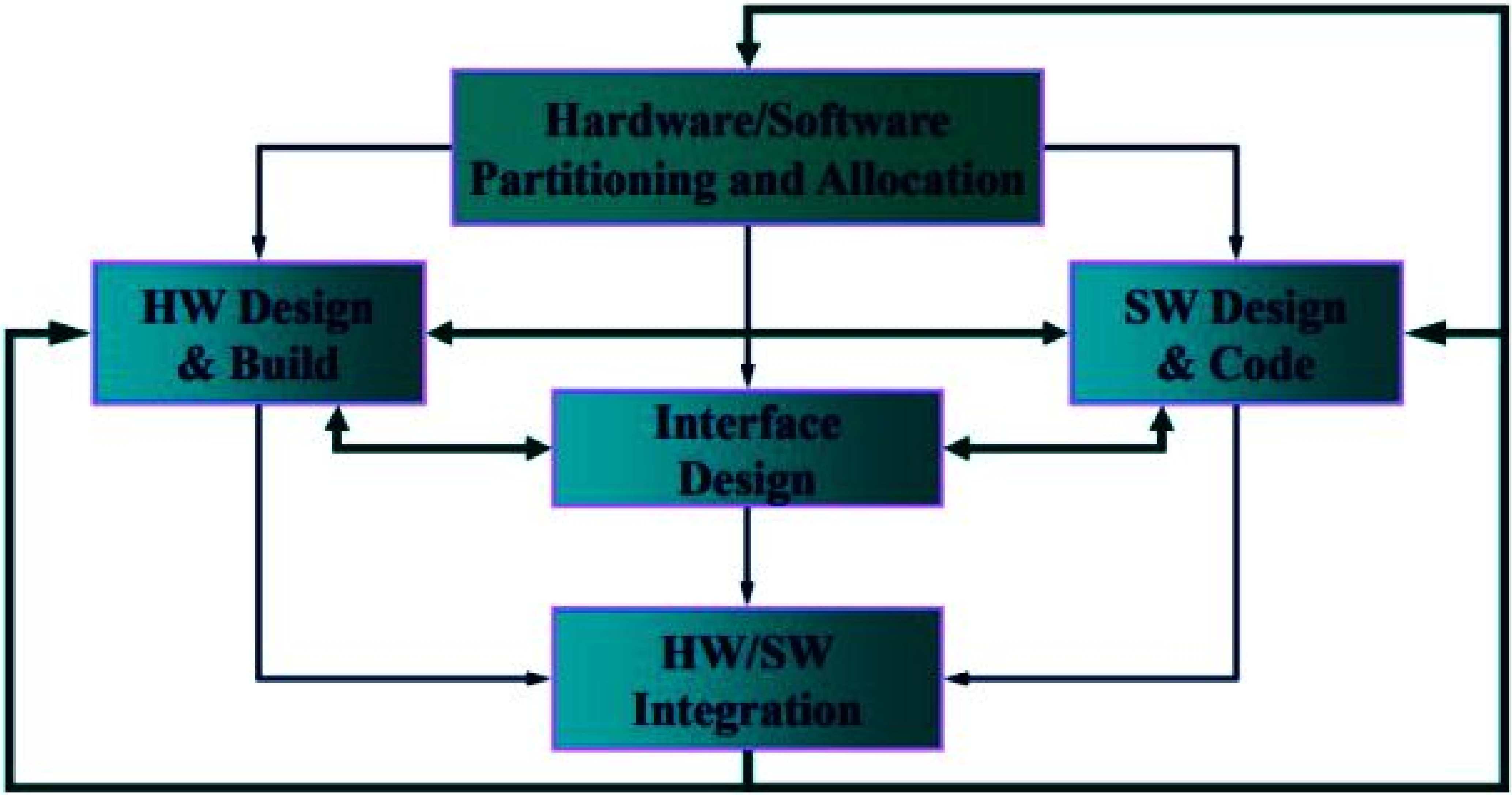
HDL: Hardware Description Language

Power-Saving Super Loop Architecture

```
Function Main_Function()
{
    Initialization();
    Do_Forever
    {
            Check_Status_of_Task();
            Perform_Calculations();
            Output_Result();
            Delay_Before_Starting_Next_Loop();
    }
}
```

HOL Code -> Compiler -> Assembler with Assembly code -> object files with real time kernel and reentrant libraries (like runtime libraries in desktop process) -> Linker -> Executable image file -> Locator -> ROM image file -> ROM Burner -> ROM (Flash) -> program initialize Read Write memory (RAM)

Embedded System Design HW/SW Co-Design Methodology



Hardware/Software partitioning and allocation -> HW Design & Build with SW Design and Code with Interface Design, with the ability of returning back modifying any Design-> HW/SW Integration and going back if any modification.

Priority Inversion

Best-known instance involved the Mars Pathfinder mission in 1997, occurs when circumstances within the system force a higher priority task to wait for a lower priority task.

If a lower-priority task has locked a resource and a higher-priority task attempts to lock that resource, the higher-priority task will be put in a blocked state until the resource is available.

If the lower-priority task soon finishes with the resource and releases it, the higher-priority task may quickly resume and it is possible that no real time constraints are violated.

Unbounded Priority Inversion

The duration of a priority inversion depends not only on the time required to handle a shared resource, but also on the unpredictable actions of other unrelated tasks