UNIVERSITY OF TEHRAN

Electrical and Computer Engineering Department

Core-Based Embedded System Design
ECE 160 - Spring 1403-1404
Computer Assignment 2 – Expanding to a Real Embedded System
Due Date: Ordibehesht 19

Description:

In this assignment, you will expand the tiny embedded system from CA1 into a real embedded system by adding:

- Separate instruction/data memories,
- Interrupt handling,
- GPIO (General-Purpose Input/Output), and
- A hardware accelerator.

You will build the system step-by-step across four parts, simulate each part in ModelSim or SystemC (your choice), and analyze the results.

Part A: Data/Instruction Memory

A minimal embedded system comprises a processor and memory. In this step, you will separate the instruction and data memories.

- **Memory Mapping:** Specify address ranges for instruction and data memories. For example:
    - Instruction Memory: 0x00000000 - 0x00001FFF (4KB, read-only)
    - Data Memory: 0x00010000 - 0x00010FFF (4KB)
- **Memory Design:** Design separate instruction (read-only) and data memory modules.
- **Building System Hardware:** Integrate the processor core with both memory modules. Add the address decoder logic to support the specified memory-mapped address ranges.
- **Building System Software:** Write a C/C++ program to compute the factorial of n=10. Compile the code using the provided *Makefile* and the link script (*link.common.ld* in the *boot* folder).
- **Evaluation:** Simulate the system in ModelSim/SystemC and verify correct memory access and factorial computation.

Part B: Interrupt Handling

Modify the system to trigger factorial calculation via an external interrupt.

- **Building System Hardware:** Add a process (in the testbench or top module) to issue an external machine interrupt after a delay (e.g., 1000ns).
- **Building System Software:** Rewrite *main()* as an infinite loop (*while(1) {}*). Move the factorial code to an external interrupt handler. Enable interrupts and configure *mtvec* (CSR instructions) in *main()*.
- **Evaluation:** Simulate and verify interrupt triggering, handler execution, and result storage.

Part C: GPIO Integration

Add a GPIO module to interface with real-world inputs/outputs. The GPIO module features:

- Input port: n (data input)
- Output port: factorial(n) (result output)
- Bus interface: data input/output, address, and control signals
- Control interface: interrupt signal output (connected to processor)
- Data registers (for input/output storage)
- Configuration registers (command/status) with mandatory bits:
  - Interrupt enable (*int_en*)
  - Interrupt status (*int_sts*)

- **Bit Length:** Determine the input/output port bit-length.
- **Memory Mapping:** Specify address ranges for memory-mapped registers.
- **GPIO:** Design the GPIO module.
- **Building System Hardware:** Add the GPIO module to the system, connecting its interrupt output and address/data interface with the processor.
- **Building System Software:** Modify the factorial code to read input (*n*) from GPIO input port and write the result to the GPIO output port (instead of memory).
- **Evaluation:** Simulate and verify GPIO functionality and interrupt-driven operation.

Part D: Hardware Accelerator

Offload factorial computation to a dedicated hardware accelerator. The factorial accelerator features:

- Bus interface: data input/output, address, and control signals
- Control interface: interrupt signal output
- Register set:
  - Data registers
  - Configuration registers (minimum required bits: *start, done, int_en*)

- **Memory Mapping:** Specify address ranges for memory-mapped registers.
- **Accelerator Design:** Design the accelerator hardware for factorial calculation.
- **Building System Hardware:** Integrate the factorial accelerator into the system.
- **Building System Software:** The processor can now offload calculations to the accelerator and act as a controller. Modify the interrupt handler to:
    - Send the input value (n) to the accelerator
    - Wait for the accelerator's completion interrupt
    - Read the result from the accelerator

  The processor performs these operations through read/write accesses to:
    - Data registers (for input/output values)
    - Command registers (to initiate operations)
    - Status registers (to check completion)
- **Evaluation:** Evaluate the entire system by simulation. Compare execution times with/without the accelerator (Part D and Part C).

Deliverables

- System Schematics:
    - Diagrams for each part (e.g., Visio or hand-drawn).
- Hardware Design Codes:
    - All HDL/SystemC codes.
- Software Design Codes:
    - All C/C++ codes.
- Simulation Results:
    - Screenshots verifying correct functionality.
- Execution Time Report:
    - Comparison of execution times (with/without accelerator).
- Put all codes (HDL/SystemC, C/C++) and the report file (including only screenshots and the execution time report) in a folder and zip it. Name your zipped file using the following format:

  ‘YourName_YourFamily_Last5digitOfYourStudentID_CA#2’