UNIVERSITY OF TEHRAN

Electrical and Computer Engineering Department

Core-Based Embedded System Design
ECE 160 - Spring 1403-1404
Computer Assignment 1 – Part B: A Tiny System
Due Date: Farvardin 24

Description:

In this assignment, you will simulate a tiny embedded system at two different abstraction levels using VHDL and SystemC. The system is based on the AFTAB processor core, which is a RISC-V processor implementing the RV32IM ISA. AFTAB is an in-order, single-issue core with sequential stages, supporting the base integer instruction set (RV32I) and the multiplication extension (RV32M). In Part A, you simulated this embedded system at the RTL. This part guides you through simulating this processor at the system level using SystemC. The code files are attached.

Part B (Due Date: Farvardin 24):

1. **Startup SystemC Environment:**
   - **a.** Write a 32-bit counter in SystemC.
   - **b.** Write a testbench for the counter.
   - **c.** Simulate the counter.
   - **d.** Extract the VCD file and take a screenshot of it opened in a VCD viewer.
2. **Simulate the Factorial Program on the AFTAB Core, Implemented in the SystemC Environment (ISS with SystemC):**
   - **a. Convert C++ to Machine Code:** Since the machine code in Part A was obtained using a compiler and assembler, these steps are skipped here, and the resulting machine code file is used directly. However, due to the memory model in the SystemC implementation of the AFTAB core, an instruction mapper is required to convert the previous memory file (“dram_dump.txt”) into the new format (“output.txt”).
   - **b. Convert Machine Code into New Format:** Open the 'Instruction Mapper' project. Copy the machine code from the factorial program (i.e., the contents of the 'dram_dump.txt' file) and paste it into the 'main.txt' file. This file serves as the

mapper's input. Run the program to convert the memory file into the new format. The output will be saved as 'output.txt'.

c. **Create a New SystemC Project:** You can reuse the previous counter project.
d. **Assign New Memory to the System:** Copy the new machine code memory file and paste it into the new SystemC project folder.
e. **Simulate SystemC Codes:** Compile and run SystemC codes of our tiny system.
f. **Verify the Result:** Verify the result is correctly written to the expected memory address. The post-simulation memory contents are recorded in 'dump.txt'.
g. **Use GCC Compiler Instead:** In Part A of Question 2, use the GCC compiler/assembler to convert your C++ code into RISC-V machine code. First, install it (a readme file is attached), then use it to generate RISC-V machine code and save the output in a new text file. Use the instruction mapper to convert the memory file into "output.txt". Continue with Parts B to F to verify the results.

Deliverables for Part B - Due Date: Farvardin 24

- Assembly and Machine Code:
  - Generated assembly and machine codes for the Factorial program.
  - Compare the assembly and machine codes of part A and part G.
- Simulation Results:
  - Screenshots or logs of the simulation results, including memory or register values.
- Execution Time Report:
  - A table comparing the execution times for online and GCC compilers.
- Put all codes (C++, Assembly, Machine, and System level) 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#1-B'
