University of Tehran

Electrical and Computer Engineering Department

ECE (8101) 342

Object Oriented Modeling of Electronic Circuits – Spring 1401-02

Homework 5: SystemC High-level Modeling and Communication Modeling

Due Date: Khordaad 05

The system shown in Figure 1 is being considered here. In this system, the processor passes vector-based data through a FIFO channel to the multiplier engine. The multiplication results will be available in a scalar-based format for the processor to pick up. The four following paragraphs discuss individual components of the system. The system functionality is explained in detail in the last paragraph.

The processor is a 32-bit CPU with the same port signals as the SAYAC processor. The port signals consist of the data bus, address bus, read memory, write memory, read IO, write IO, and memory ready. The processor can be modeled as a dataflow-accurate level by bracketing it in a SystemC module. In such an implementation, the processor is nothing more than a software program that provides vector-based data for the multiplier and collects the results.

The intermediate component and the multiplier engine are register-accurate models that communicate with the processor in a non-blocking fashion through the bus interface, and are connected together through a blocking FIFO channel. A register-accurate view of them is shown in Figure 2. As shown, each of these has a control register for handshaking with the processor and a data register for sending or receiving data to/from the processor.

The bus interface only plays the role of a decoding logic to connect the processor from the master side to the elements of the slave side. It can be modeled as a data flow-accurate BFM as discussed in the lectures.

The queue channel is a blocking FIFO channel with a size of 32. It is to be modeled as an abstract channel with *put*-and-*get* interfaces. Use template classes to configure the type of stored data and the channel size.

A 32-bit data memory retains 128 multiplication operands in address locations 0000_0008 to 0000_0087. In each calculation round, the processor reads 32 vector-based data from the data memory and puts that in the queue channel through the intermediate component. The processor starts sending data to the intermediate component if the queue channel is empty and the *empty* flag is active. When this is completed, the processor issues the *start* signal of the multiplier by writing a 1 in the *start* flag of the control register of the multiplier. After that, the multiplier starts the multiplication process for the data stored in the channel. When a multiplication result is ready, the processor collects the result by reading the output register of the multiplier (*dataOutReg* in Figure 1). The way the processor collects the result data is referred to as scalar-based format here. A pseudo-code of the program that is run on the processor is shown in Figure 3.

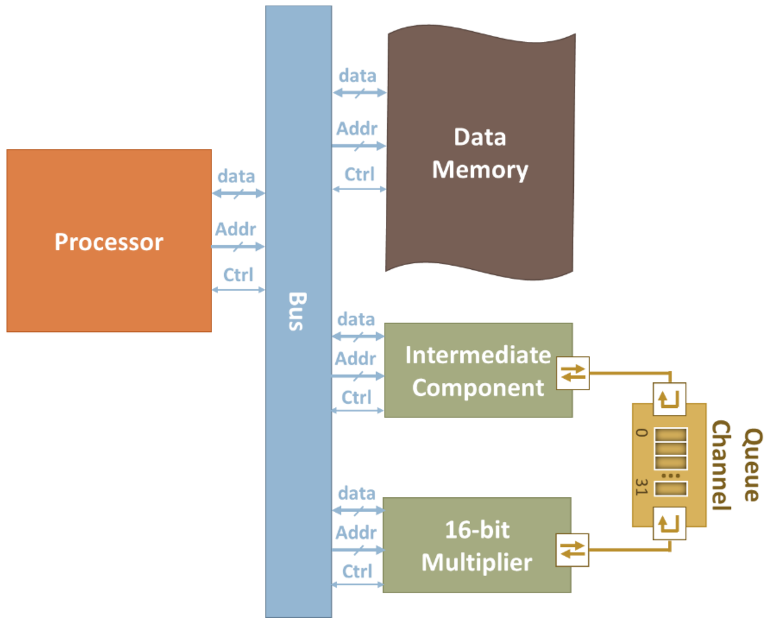

Figure 1 – An overall view of the system

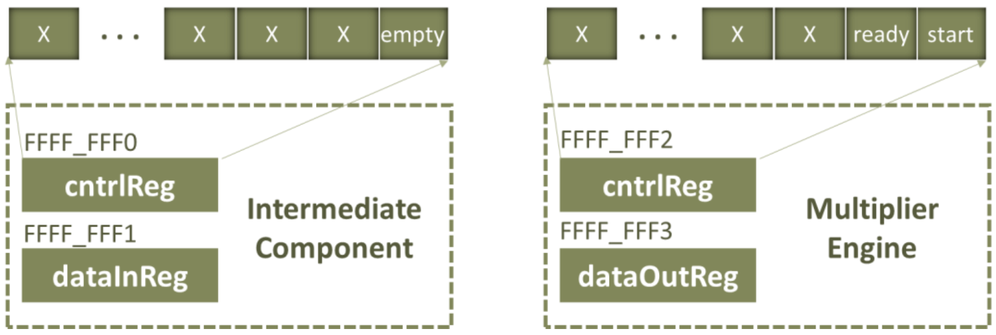

Figure 2 – A register-accurate view of the intermediate component and the multiplier

```
READ 128 32-bit data from memory
LOOP (4)  // apply all data through 4 rounds
    IF (channel is empty)
        WRITE a 32 32-bit data in channel
        ISSUE start of multiplier
        doSomethingElse()
        LOOP (32) // collect all 32 results
            IF (multiplication result is ready)
                READ multiplication result
            ELSE
                doSomethingElse()
        end LOOP
    ELSE
        doSomethingElse()
end LOOP
```

Figure 3 – A pseudo-code of the processor program

In this homework, you are to

A) Model all components of the system shown in Figure 1 in SystemC. You can use the codes provided in the lectures.

B) Write a testbench and test each element using at least five test cases.

C) Put together the elements modeled in Part A to form the system.

D) Write a testbench and test the system that performs functions discussed above.

Deliverables:

1. All SystemC codes with proper naming
2. A complete report containing
    - Schematic diagrams drawing in Visio or other visualization tools,
    - Enough design illustration and description,
    - Simulation results, input data, and output justification.