System Level Languages
IL2452

Lab 2

Design and Exploration Using SystemC

Name:_____

Personal Number:_____

Approval

Date:_____

Sign:_____

1.General information

Each Lab event has been thought of as consisting of 3 sections:

- The first section where students have to accomplish a set of *preparation tasks*. Their main purpose is to put the students in the condition of being able to solve the lab tasks.
- The second section where students will solve the actual lab tasks.
- The third section will be a discussion with the lab assistant about both the preparation and the lab tasks. To get a Pass, students will have to justify their answers.

2.Introduction to lab2

Prerequisites: before starting working on Lab 2, students should have gone through the material shown during Lectures. It will be also useful for you to consult the other materials placed on the course web-page.

Goal: the main goal of Lab 2 is to allow students to improve their skills in design, simulate and explore systems using SystemC.

Skills acquired: at the end of Lab 2, we expect that students will be able to use SystemC to design simple real life systems and to explore their functionality.

3.Preparation tasks

Task 3.1

The Random Access Memory (RAM) shown in Figure 1 has the following specifications:

- Single bidirectional data port (read/write)
- Byte addressable
- Data port width is 8 bits
- Address port
- Address port width is 32
- Chip enable signal, memory is read or written if CE is 1
- Write enable signal, write is performed if WE is 1
- Read/write operations

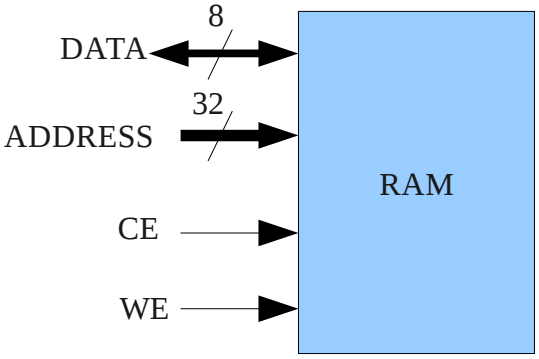

Figure 1

Write a systemC code to model this RAM while maintaining the following properties:

- Memory size should be generic, so that the model can be instantiated with different sizes.
- During memory construction, initial data have to be loaded from the file "T1.txt" which is structured as:

Address ***Data***

- The Model should be able to detect out of range addresses and to report an error when this case happen. Hint! use ***SC_REPORT_ERROR(.....)***
- Write and read operations are controlled by WE input. If WE is 1 then the operation is write otherwise, operation is read. If the requested operation (write/read) occurs at the presence of opposite WE value, memory should report an error.
- If write operation is requested and CE is 0, operation should be discarded.
- If read operation is requested and CE is 0, memory should give the value **0xFF**.
- Suitable data types should be selected to represent memory inputs and outputs.

Task 3.2

Write a systemc test bench for simulating the RAM model shown in Task 3.1. This test bench is to be used for:

- Setting and resetting memory signals
- Verifying that Memory is initialized correctly (case I). To do so, Memory content have to be read and compared against the file "T1.txt". In case of mismatch between memory content and "T1.txt", all mismatches have to be reported in an error file structured as:

  Address_of_mismatch File_data Memory_data

- Verifying memory write and read operations (case II). To do so, the test bench has to write to memory all the values given in the file "T2.txt" which has the same structure as "T1.txt". In case of mismatch between memory and "T2.txt", mismatch has to be displayed on the console as:

  Address_of_mismatch File_data Memory_data

Note: You can use a separate test bench for each of the two tasks above or you can keep one for both.

Task 3.3

Make two separate instances of the RAM model, memory A and memory B. each of them is 1KB while considering each byte is composed of 7 bits data and 1 bit as odd parity check (MSB). In this task, initialization values of both memories have to be loaded from the file "T3.txt".

The main assumption of this task is that the content of both memories A and B are affected be external factors (like noise) and thus can be distorted. Listing 1 and Listing 2 are pseudo codes for emulating two possible distortions for memory A and memory B respectively.

- Write the two code segments that implement Listing 1 and Listing 2 and insert them inside the constructor of Memory A and Memory B respectively. Make sure that data distortion is done after memory initialization.
- The test bench used in Task 3.2 can be modified and used here as well. The only modification is to calculate the odd parity value of the data in each memory read operation and to report it together with the mismatches. In other words, In case of mismatch between memory and "T3.txt", mismatches and odd parity values have to be reported in an error file which is structured as:

  Address_of_mismatch File_data Memory_data odd_parity_value

Task 3.4

Implement a single bit odd parity correction to enhance each of memory A and B so that the data in each memory read operation is checked and corrected if it is erroneous. The correction is done by flipping a single bit so that, the (absolute) difference between current memory value and the previous memory value (last read value) is minimized. In this task, Memory initialization values are read from the file "T3.txt" and the test bench is the same as Task 3.3.

4.Lab tasks

The mark (*) means that the task/step need to be saved as ready to run version since it is needed during lab presentation.

Task 4.1

- Use the memory model implemented in Task 3.1 to instantiate a 2KB memory.
- Bind the test bench implemented in Task 3.2 to this memory instance, set the correct memory input values and verify memory for both cases mention in Task 3.2 (*).
- Verify the memory for case II in the following scenarios:
  - WE is set to 1 and remains 1 (*).
  - CE is set to 0 and remains 0 (*).
  - memory size is 1KB (*).

Task 4.2

Run both memory instances together with the test benches as explained in Task 3.3, analyze error reports (*)

- For both memories, calculate total number and percentage of mismatches .
- For both memories, calculate total number and percentage of incorrect odd parity value.

Task 4.3

Run both memory instances together with the test benches as explained in Task 3.4, analyze error reports (*)

- For both memories, calculate total number and percentage of mismatches .
- For both memories, calculate total number and percentage of incorrect odd parity value.

5. Conclusion

Consider the results collected from the Tasks 4.2 and 4.3 while having the following assumptions:

- Memory cost: $A_{enhanced} > B_{enhanced} > A > B$
- Memory Latency: $B_{enhanced} > A_{enhanced} > B > A \geq Minimum\ Requirements$

Which of these memories can be the best choice for each of the applications listed in Table 1? Justify why.

Application	Memory
Real time audio streaming	
Storage of program instructions	
Image storage	
Video streaming	

Table 1: Memory selection

Appendix

Listing 1

```
for (index from 0 to end of memory)
begin
data = read memory[index];
data = data Bitwise OR 0x04;
memory[index] = data;
end
```

Listing 2

```
for (index from 0 to end of memory)
begin
data = read memory[index];
if(data == 0x70)
data = data Bitwise AND 0x73;
else if (data == 0x67)
data = data Bitwise AND 0x63;
memory[index] = data;
end
```