Electronic System Level Design

Assignment 1, 2017-3-7

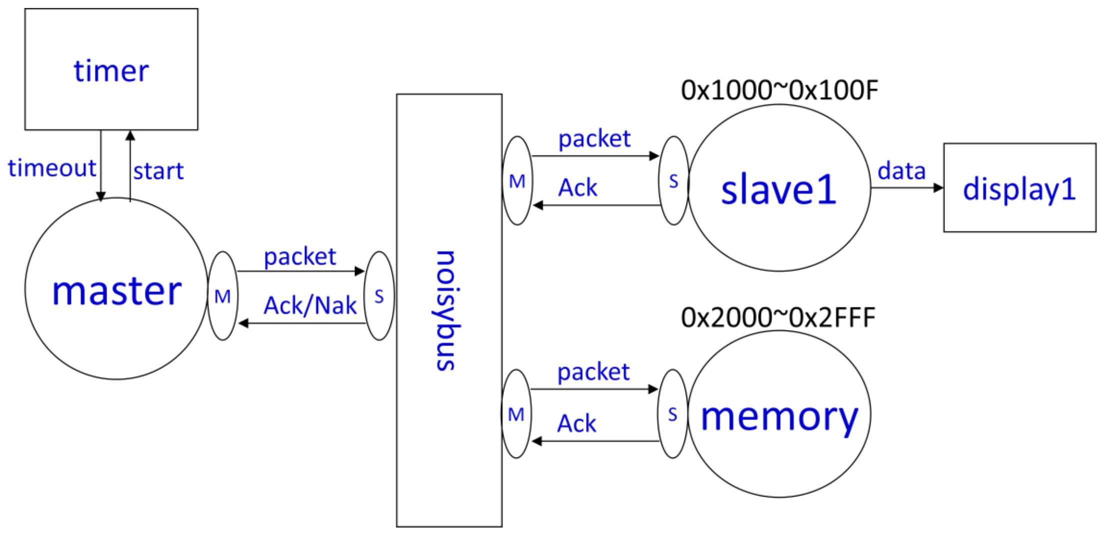
Abstract

Develop a system based on the "A Simple Example" on page 39, Section 2.2 lecture notes. This system is comprised of 1 master component, 1 bus, and 2 addressable salve components.

System Spec

Block Diagram

Please construct the system exactly as depicted in this diagram. Ten (10) points will be taken for each mismatch, except for adding extra slaves described in the 40% bonus.



Description

Please read carefully. All outputs required are described in the text. Ten (10) points will be taken for each missing required output and behavior.

1. First, to practice more on SystemC coding, please **hand copy** the "Simple Example" to your workstation. Create a Makefile and test the original code to see if it works. Fix any SystemC errors if presented. This is a code from SystemC 2.0 so no guarantee it works in SystemC 2.3. To ease my grading effort, it is welcomed not to change too much on the original code.
2. **packet :** Modify the 'packet' class as follows:

a. Make data members as

```
sc_uint<32> addr;  // The address field
sc_uint<8>  data;  //  The data field
bool        rw;    // 1: Write; 0: Read
bool        ack;   // 1: Ack; 0: Nak
```

b. Modify all member functions to support new data members
c. Modify packet.cpp to support the trace of the new data members

2. **master :** Rename 'transmit" SC_MODULE as 'master', and add/modify fields and member functions as you see fit so the **master**:
   a. Reads "MasterData.txt" as provided. Each line in the MasterData.txt is one instruction to write/read a data to/from the addressed slave component. The format of the MasterData.txt is
   ```
   instr addr data
   ```
   where `instr` is `1` (write) or `0` (read), `addr` is a hexadecimal number to represent the target address, and `data` is an 8-bit integer. For example:
   ```
   1 0x1003 -63
   0 0x2458
   ```
   b. Prepares a **packet** as the instruction retrieved, and writes out to the salve through the bus, like in the "Simple Example." The **master** then waits for an Ack/Nak from the slave/bus
   c. On receiving a returned message, log it to a MasterLog.txt, for example:
   ```
   Read 0x3588 0, Nak
   Write 0x2348 56, Ack
   Read 0x1002 34, Ack
   ```
   d. No matter it is an Ack or Nak received, the **master** proceeds and reads in the next instruction from the MasterData.txt then repeats above procedure
   e. If no Ack nor Nak received before a timeout alarm is triggered by the **timer**, sends the original **packet** again, and logs to the MasterLog.txt, for example:
   ```
   Timeout on "Read 0x2100"
   Timeout on "Write 0x1007 55"
   ```
3. **noisybus :** This is a bus that
   a. Needs to resolve the `addr` received from the **master**. If the `addr` is not within 0x1000~0x100F and 0x2000~0x2FFF, sends immediately a 'Nak' **packet** back to the **master**, by copying the `addr`, `data` and `rw` fields it receives but set the `ack` field to 0. Only on this condition to log an error to a file named NoisybusLog.txt. The error message should read like:

```
Invalid address 0x1010, read failed
Invalid address 0x3700, write failed
```

   b. If the address is within 0x1000~0x100F or 0x2000~0x2FFF, then the **noisybus** passes the **packet** to **slave1** or **memory** accordingly
   c. Using the same random logic to generate a noise. However, on a noisy condition the **noisybus** simply drops the **packet** without passing the packet down to the slave, then logs a message to the NoisybusLog.txt file, for example:

```
Read 0x1004 dropped
Write 0x2334 dropped
```

2. On receiving an Ack **packet** (notice that salves always send Ack back) from a **slave**, simply passes the packet to the master. Again, using the same random logic to mimic a noise. The noise is to simply drop the Ack **packet**. Do not alter the **packet** in any way and return to the **master**
4. **<u>slave1</u>** : Rename 'receiver' SC_MODULE as 'slave1' , and add/modify fields and member functions as you see fit so **slave1**:
   a. Creates an array to implement a register file addressed from 0x1000 to 0x100F, which contains sixteen (16) `sc_uint<8>` registers
   b. On receiving a write command from bus, **slave1** writes the data received to the addressed register, then logs to a file named Slave1Log.txt, for example

```
Address 0x1003, write 106 successfully
```

   c. On receiving a read command, **slave1** reads the data stored in the addressed register, then writes to Slave1Log.txt, for example

```
Address 0x1002, read -79 successfully
```

   d. Sends the data read or written to the **display** component
   e. Returns an Ack packet to the bus, with `addr` and `data` read (from its own register) or written (received from **master**,) and sets the `ack` field to 1
5. **<u>memory</u>** : Copy '**slave1**' and name it '**memory**', and create a 1KB array as the memory. Each cell of the array is a `sc_uint<8>` and addressed from 0x2000 to 0x2FFF. The behavior of the **memory** is almost identical to the **slave1**, except there is no sending data to the **display** component, and the log file is named MemoryLog.txt
6. **<u>display</u> & <u>timer</u>** : Use original components and no need to modify
7. **<u>Please</u>** turn in the complete system including the Makefile you created

Due date

11:59PM, March 14, 2017

Score weight (towards the final grade)

10%

40% Bonus

Make the **noisybus** support arbitrary number of slaves. To prove it works, please add two more slaves to the system.