COMP3211/9211 2003 Assignment 1

Simple PCI

Contribution to final assessment: 10%

Due: 4.30 PM, Wednesday, 17 September

Overview

The aim of this assignment is to describe, simulate, and analyse a simplified PCI protocol for a variety of transaction types. A device (disk, network interface, graphics adapter, bridge to processor/memory, etc.) is connected to a Simple PCI bus via a generic bus driver that can initiate or target the transactions between devices. Such a driver is to be designed, with interfaces both to the bus and any one of the possible devices that are to be connected to the bus.

In order to test the driver and bus protocol, a bus system is to be constructed by interconnecting a set of identical drivers. The system is to be tested by simulating the activities of devices interacting with the drivers. In turn, these drivers will communicate across the bus by initiating and targeting the required transactions. The operation of the system can thus be observed from the bus signal waveforms and the signals produced at the interface between the drivers and their associated devices.

Note that you are not required to describe the devices themselves, just the drivers and bus system. Each device will be simulated by providing external inputs to its driver and observing the outputs from the driver.

Detailed Requirements

You are to implement the simplified PCI protocol given in Stallings Fig 3.23, reproduced below as Figure 1, with the following modification to the definition of C/BE#: the use of C/BE# will be restricted to simply encoding the transaction type (BUS CMD) for the first clock cycle of a transaction when FRAME# is asserted.

The following transaction types are to be supported:

Transaction Type	4-bit C/BE# Encoding
Read single word	0100
Read double word	0101
Write single word	1000
Write quad (4) word(s)	1010

A word will be 8 bits wide, and the 8-bit address/data lines are to be shared as for PCI; the remaining signals are to be as in Figure 1. Note that all bus signals are to be tri-stated (capable of bi-directional signalling to a driver).

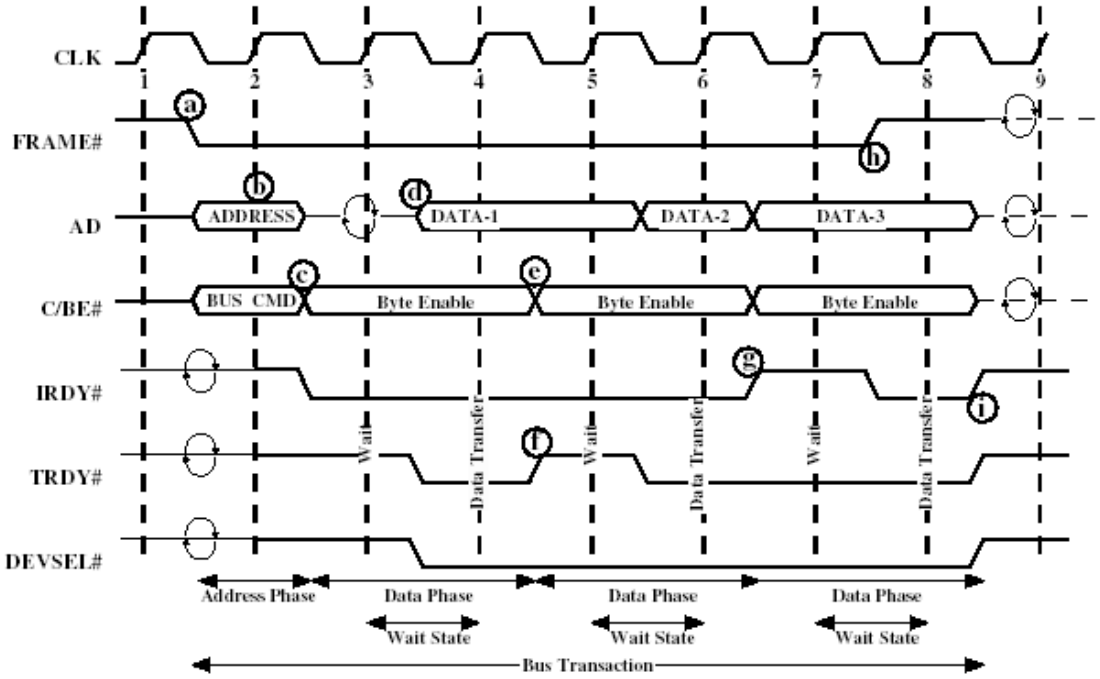

Figure 1 – PCI Read Transaction

Design bus drivers that are capable of assuming the initiator and target role. Each driver is to be provided with the following external data, address, and control ports in order to interface with a device:

Port name	Mode	Size	Purpose
DEV_DATA	I/O	8	write/read data to/from bus
DEV_ADDR	I/O	8	address to be written to/read from
DEV_RTYP	I/O	4	request type (transaction type)
DEV_RDY	I	1	data available for write/device ready to read
DRV_RDY	O	1	data available for read/driver ready for write
DEV_GRNT	I	1	signal indicating device granted bus access (assumed to be provided by bus arbiter)

These ports are used as follows. Let us say a device attached to a particular bus driver, DRV00 say, wishes to read two words from address 01010110. This would be simulated by driving address 01010110 onto the DEV_ADDR input of DRV00 and request type 0101 onto the DEV_RTYP input. The grant line DEV_GRNT would then be asserted to simulate the arbiter granting this device access to the bus, and the driver would initiate a Simple PCI transaction by driving the FRAME#, AD, and C/BE# outputs of the driver. These signals would be conveyed via the bus to the remaining devices, one of which is targeted by the address and commences responding once IRDY# is asserted. In this case, let us say the device attached to driver DRV01 is targeted. DRV01 outputs the address received to DEV_ADDR and commences responding. The simulation then needs to provide two words as DEV_DATA input to DRV01. These are conveyed across the bus back to DRV00 from which they are output on DEV_DATA. Availability of the required words is simulated at the target driver by asserting DEV_RDY, and at the initiator by asserting DRV_RDY. Unavailability of data would be conveyed to the target by de-asserting DEV_RDY, and by the initiator de-asserting DRV_RDY. In particular, note that unavailability of data is to be supported in the Simple PCI protocol as depicted in Figure 1. Each

driver should be supplied with a clock input, CLK, and a RESET to reinitialise all state machines.

Driver ports to the bus must be labelled as in Figure 1. Do not use the '#' symbol in a port label, as it will not be accepted. Since '/' is also unacceptable, use C_BE in lieu of C/BE#.

It can be assumed that arbitration is managed externally to the bus system by providing appropriate stimuli to the DEV_GRNT control signal for each driver - your test data should assert the grant line of drivers that are required to initiate a transaction. It can be assumed that access to the bus is granted until a transaction is complete. It can also be assumed that the arbiter is informed by the device that a transaction is complete, and that granting of bus access can thus be revoked. Furthermore, it should be noted that no two drivers should be granted access to the bus at the same time.

It can be assumed that the devices connected to the drivers behave correctly, i.e., they accept the correct multiple of words when being written to as initiator or target, and supply the required number of words when initiating or responding to a multi-word write request. However, your design must be able to cope with delays in delivering data and must therefore handle DEV_RDY, TRDY, IRDY, and DRV_RDY signals correctly.

You are required to instantiate 4 bus drivers in your bus system. These will be labelled DRV00 through DRV11. DRV00 responds to transactions targeting addresses in the range 00000000-00111111; DRV01 will cover addresses in the range 01000000-01111111, and so on. It can be assumed that no device would attempt to target an address in its own address range via a bus transaction.

The arrangement of the bus system is depicted in Figure 2. A skeleton for the top-level entity and architecture is provided in Figure 3.

Deliverables

1) Sketch waveforms similar to Figure 1 for the bus transaction types that are to be supported - it is recommended that you have these checked by your tutor [10 marks]

2) Draw state diagrams for a bus transaction initiator and target [15 marks]

3) Draw the state diagrams for the device-driver interactions when the driver is to initiate and target either a read or write transaction. [15 marks]

4) Produce entity/architecture descriptions for a bus driver. Use the port labels as specified. [15 marks]

5) Produce entity/architecture descriptions for a bus system consisting of 4 driver components suitably interconnected. [5 marks]

6) Produce 8 simulation traces that demonstrate correct operation of the bus system for each of the required transaction types including and without delays in delivering data from a device to a driver. These traces should include as a minimum all relevant input and output signals to the bus system as well as the waveforms for the bus signals. [40 marks]

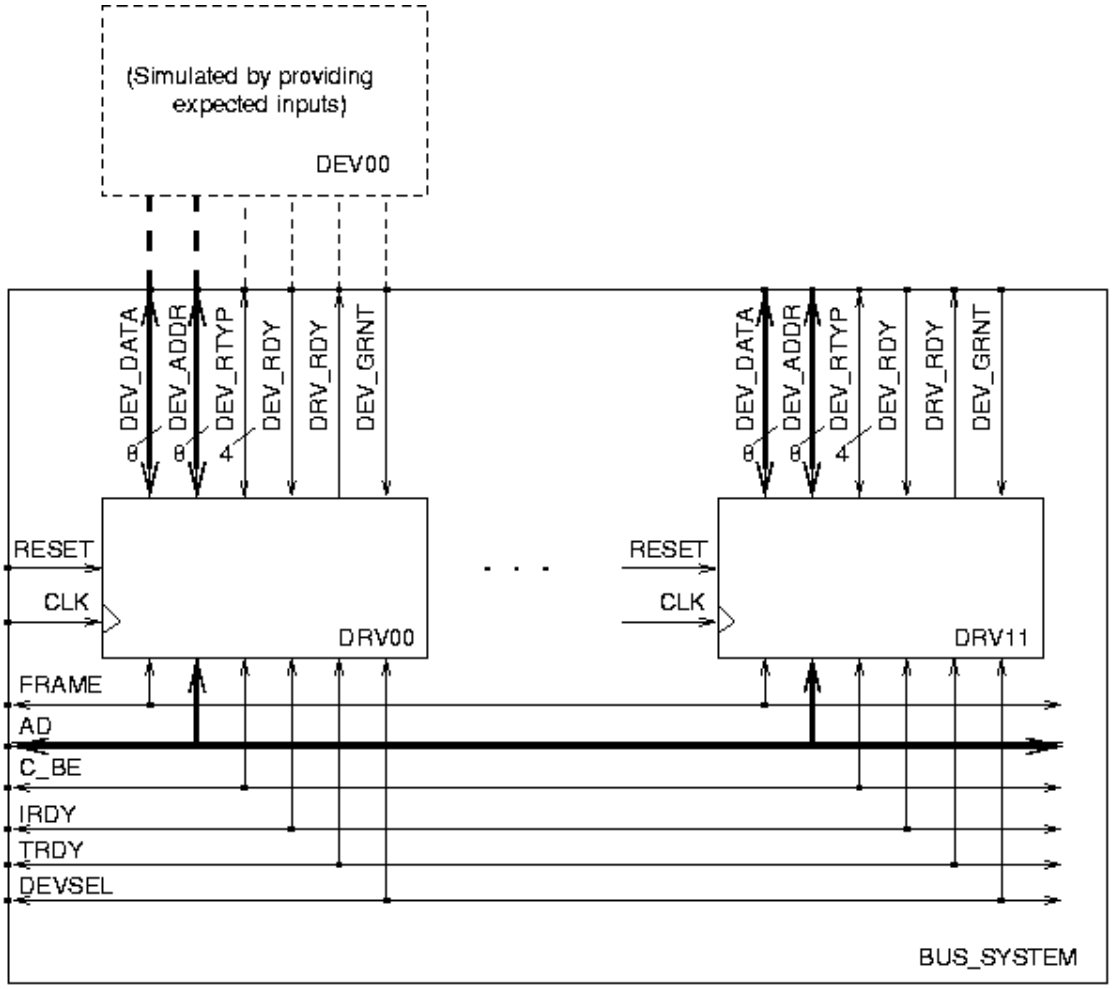

Figure 2 – Arrangement of bus system

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity bus_system is
    Port ( DRV00_DEV_DATA : inout std_logic_vector(7 downto 0);
           DRV00_DEV_ADDR : inout std_logic_vector(7 downto 0);
           ...
           DRV11_DEV_GRNT : in std_logic;
           CLK : in std_logic;
           RESET : in std_logic);
end bus_system;

architecture structural of bus_system is

  component driver
    port (DEV_DATA: ...
          ...
          CLK:...
          RESET:...);
  end component;
```

```
  -- signal labels to be used for bus

  signal FRAME: std_logic;
  signal AD: std_logic_vector(7 downto 0);
  signal C_BE: std_logic_vector(3 downto 0);
  signal IRDY: std_logic;
  signal TRDY: std_logic;
  signal DEVSEL: std_logic;

begin

  DRV00:...
  ...
  DRV11:...

end structural;
```

Figure 3 – Skeleton of bus system description

A printed copy of the above deliverables including a title page identifying

(i) the assignment title: COMP3211/9211 2003 Assignment 1,
(ii) your Tutorial day & time,
(iii) your Tutor's name, and
(iv) the names and student numbers of the members of your group (for this assignment, you are expected to pair with another person from your tutorial group)

should be submitted to and marked RECEIVED by the student office by 4.30 PM, Wednesday, 17 September, 2003.

Your project should be archived into a project ZIP named **`assign1.zip`**. You should also merge all VHDL source files you wrote into a single file that should not be zipped called **`check.vhd`**. These two files are then to be turned in using the give system using the command **`give a1-pci assign1.zip check.vhd`** by 5.30 PM, Wednesday, 17 September, 2003.

Please refer to the course outline for an explanation of the penalties that apply for late or copied work.