Computer Structure Lab - Fall 1998
Dr. Guy Even

Handout #3: monitoring a non-trivial application (v 1.0)

The purpose of this handout is to design a monitoring device that can monitor all the signals and registers values in a bus master. The tasks in this assignment are:

1. Extend the design of the Monitor Slave to support write transactions.
2. Extend the design of the Monitor Slave with a small RAM that can store cycle by cycle values of internal signals.
3. Test the extended Monitor Slave using a simple read machine and a simple write machine.

1 The Logic Analyzer - Storing monitored values in a RAM

As mentioned in the 2nd handout, there is a conflict when the Monitor Slave monitors the bus activity of the application. This conflict is due to the fact that the Monitor Slave reports the monitored signals via the bus as well, thus creating a conflict over the shared resource (the RESA bus).

The natural way to solve this problem is to store the monitored values in a RAM and to send the sampled values later. This method also combines well with our future desire to be able to monitor the DLX instruction by instruction. Recall that an execution of a DLX instruction takes several clock cycles. If we want to monitor control signals and register values during the execution of a single instruction, we should be able to do the following:

1. Store the monitored signals cycle by cycle during the execution of an instruction.
2. After the instruction's execution is completed, be prepared to answer bus read transactions in which the PC monitor program asks about the sampled values.

The structure of the Logic Analyzer and its connections with the Monitor Slave and the application is depicted in Figure 1. The Logic Analyzer consists mainly of two parts:

1. The Logic Analyzer's RAM. This is a 32×32-bit RAM that stores the sampled values from the application. In each clock cycle, up to 32 signals can be stored.
2. The counter. The 5-bit counter generates the address into which sampled values are stored. At the beginning of an execution of an application's instruction, the counter is reset by the application. The counter's output value equals the number of cycles that have elapsed since the beginning of the last instruction. This way, the Logic Analyzer's RAM is filled "row by row" by the sampled values until the execution of the current instruction ends. The Monitor Slave can read the value of the counter and send it to the PC monitor program, so that the PC monitor program knows how many queries to make.

The Monitor Slave has two registers that are related to the Logic Analyzer.

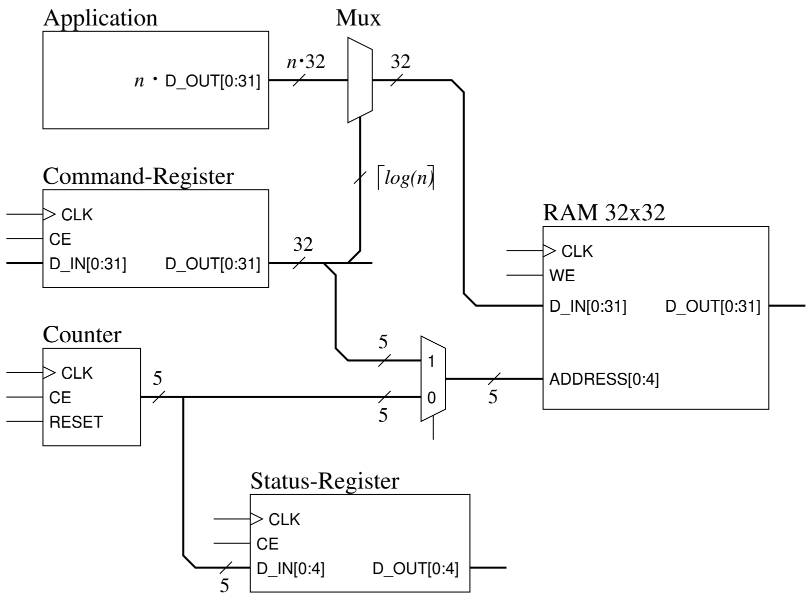

Figure 1: The structure of the Logic Analyzer and its connections with the Monitor Slave and the application.

1. The Command Register. This 32-bit register is divided into two fields.

   (a) The bottom 5 bits are used to address the Logic Analyzer's RAM when the PC monitor program reads the sampled values (i.e. the address corresponds to the number of cycles that have elapsed since the beginning of the execution of the last instruction).

   (b) The top bits are used to select which 32-bit data set is routed from the application to the data-in input of the Logic Analyzer's RAM.

2. The Status Register latches the value of the Logic Analyzer's counter so that the Monitor Slave can report the number of rows that contain relevant data in the Logic Analyzer's RAM. Note that the output of Logic Analyzer's counter is registered. This means that one does not really need a new register to store the counter's output since it suffices to deactivate the counter enable input of the counter.

2 Extending the Monitor Slave

The Command Register, which is part of the Monitor Slave, is fed values from the PC monitor program. This means that the Monitor Slave must support write transactions as well.

An interesting direction is to use the write transactions to the Monitor Slave to set certain values in the application. However, we will not pursue this direction.

3 A Read Machine

The Read Machine is an application that reads from a fixed memory address (0x800000). It is connected as a bus master to the I/O Control Logic. It stores the read value in a register.

The state diagram of the Read Machine is depicted in Figure 2. The bus signals are activated by the Read Machine during fetch, kept active during wait4ack, and deactivated during store. During wait the state machine is waiting for the STEP_EN signal.

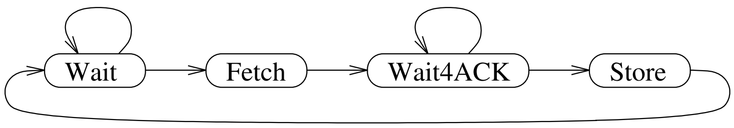

Figure 2: State diagram of the Read Machine

4 A Write Machine

The Write Machine is an application that writes to a fixed memory address (0x800000) your favorite value. It is connected as a bus master to the I/O Control Logic.

The state diagram of the Write Machine is depicted in Figure 3.

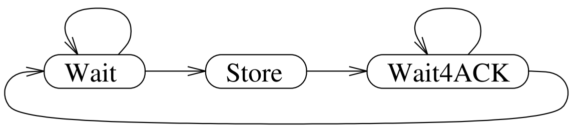

Figure 3: State diagram of the Write Machine

5 Homework

1. (20 pts) Design a Read Machine and a Write Machine.
2. (5 pts) Which control signals should the application transmit to the Monitor Slave and the Logic Analyzer? Differentiate between signals that are monitored by the Logic Analyzer, and signals that aid the Monitor Slave and the Logic Analyzer in functioning properly.
3. (20 pts) Write the equations for the following signals:
   (a) Write enable (`WE`) signal of Logic Analyzer's RAM.
   (b) Count enable (`CE`) signal of the Logic Analyzer's counter.
   (c) The reset signal of the Logic Analyzer's counter.

(d) The Select signal of the mux that selects the address input of the Logic Analyzer's RAM.

Remark: The Logic Analyzer should sample signals from the Read Machine during the fetch, wait4ack, and store states. Make sure that the reset signal enables the Logic Analyzer to store the sampled signals from the application in address 0 when the Read Machine enters the fetch state.

4. (15 pts) Extend the Monitor Slabe from Exercise #2 to support write transactions and the Logic Analyzer.

5. (20 pts) Design the Logic Analyzer. (Bonus (10 pts): a 32×32-bit RAM is available in the web site. If you design it yourself using 32×8-bit RAMs, you get the bonus).

6. (20 pts) Prepare two designs:

   (a) A design with the I/O control logic, the Monitor Slave, the Logic Analyzer, and the Read Machine.

   (b) A design with the I/O control logic, the Monitor Slave, the Logic Analyzer, and the Write Machine.

   Implement your designs on the RESA. Write a configuration file for the PC monitor program. Monitor the bus activity of the applications (the Read and Write Machines) and the accessed memory address (In the Read Machine, this means that the register of the Read Machine is monitored as well as the main memory address. In the Write Machine, only the main memory address is monitored).

Warnings and reminders:

1. **This exercise requires using the RESA. Do not touch it! The RESA is connected to a 486-PC and all the operations should be done via the PC control program.**

2. **Do not use drivers in your design!!**

3. **All registers are clocked by the same clock. The regisers differ in the clock enable signals.**