

14:332:437 Digital Systems Design
Fall 2009
Problem Set 6
Electrical and Computer Engineering Department
College of Engineering
Rutgers University

Prof. Michael L. Bushnell

Assigned: November 9, 2009

Due: November 16, 2009

STUDENTS ARE EXPECTED TO WORK INDEPENDENTLY ON BOTH HOMEWORK AND EXAMINATIONS. NO COLLABORATION IS PERMITTED, UNLESS YOU ARE OTHERWISE INSTRUCTED. YOUR WORK MUST BE YOUR OWN.

1. *Automated Teller Machine Design.* Automated Teller Machines (ATMs) are now widely used around the world, and each such machine has a built-in microprocessor to control its display. However, the ATM still requires custom hardware to enable the microprocessor to control it. Design a finite state machine to control the user interface of an ATM and function as an intermediary between the microprocessor and the user interface as well as a check on the dispensing of cash. You do not need to control the display or the printer unit – assume that standard microprocessor RS232C output ports are used for that.

Design an ATM controller as follows:

- (a) The machine should accept signals from the microprocessor, *INITIAL-CASH*, indicating how much cash has been loaded into the machine. Store this signal in a register named *CASH*.
- (b) Whenever any money is dispensed to the user, you must deduct that amount of money from *CASH*. If *CASH* is about to become negative, abort the transaction and shut down the ATM. If *CASH* becomes 0, shut down the ATM (no money available).
- (c) When the microprocessor asserts a signal *GIVE-THEM-20*, your controller should assert the *DUMP-20* signal. This causes the ATM to move a \$ 20 bill into the intermediate counting station. If, instead, the microprocessor asserts the signal *GIVE-THEM-10*, your controller should assert the *DUMP-10* signal. This causes the ATM to move a \$ 10 bill into the intermediate counting station.
- (d) While the bill moves through the counting station, additional mechanical hardware checks for a single bill in the counting station. If, instead, there is no bill or more than one bill, the mechanical hardware asserts *MISCOUNT*. This means that the ATM miscounted cash at the intermediate counting station. Your controller should

abort the transaction by asserting *ABORT* to the microprocessor. It should return the customer's card by asserting *EJECT-CARD*. It should also shut down the ATM (by asserting *LOCK-KEYBOARD* to "1", *LOCK-OUT-CARD* to "1", and *ACCEPT-ENVELOPE* to "0").

- (e) Whenever anything goes wrong with a transaction, assert *LOCK-KEYBOARD* to prevent anyone from typing at the keyboard.
- (f) When you wish to shut down the ATM, asserting *LOCK-OUT-CARD* to "1" prevents anyone from inserting their ATM card into the card reader.
- (g) When someone successfully sticks their ATM card into the ATM and it is successfully positioned for reading, the mechanical card reader hardware will assert *CARD-REGISTERED* to "1". This means that the card was successfully positioned in the read, and now the microprocessor can read it, so your machine should assert *CARD-OK* to the microprocessor.
- (h) When a card is stuck in the ATM card reader, the mechanical hardware asserts *CARD-JAMMED*. Your state machine should abort the transaction (by asserting *ABORT*) and shut down the ATM.
- (i) When the microprocessor indicates that it is ready to accept a deposit envelope from the ATM, it will assert *GRAB-ENVELOPE*. When this happens, your controller should assert *ACCEPT-ENVELOPE* to "1". This will allow the deposit envelope reader to accept an envelope from the user.

Table 1 describes this machine's signals. Design and turn in the following for your ATM

Table 1: Signals for ATM Machine.

Signal	Type
<i>INIT</i>	input
<i>ACCEPT-ENVELOPE</i>	output
<i>GRAB-ENVELOPE</i>	input
<i>CARD-OK</i>	output
<i>CARD-JAMMED</i>	input
<i>LOCK-OUT-CARD</i>	output
<i>CARD-REGISTERED</i>	input
<i>ABORT</i>	output
<i>EJECT-CARD</i>	output
<i>CASH</i>	state variables
<i>INITIAL-CASH</i>	input
<i>DUMP-10</i>	output
<i>DUMP-20</i>	output
<i>GIVE-THEM-10</i>	input
<i>GIVE-THEM-20</i>	input
<i>MISCOUNT</i>	input

system controller using the Synopsys system:

- (a) Mnemonic documented state transition diagram.
- (b) System block diagram (from the Synopsys system).

- (c) VHDL code for your state machine.
- (d) Synthesized and optimized logic circuit schematic (from the Synopsys system).
- (e) A COMPLETE simulation of all state transitions for your controller, using the supplied test bench. Grading is by each correct state transition.