Project #1 – I2C Master Controller:
Simulation, Submission, and Demo Information

This document outlines the required testing method, submission details, and demo information for the I2C master controller of Project #1. *Please be sure to read it carefully.*

SUBMISSION DEADLINE: Your completed problem is **due by 4:00 p.m. on Friday, November 18.** You will have both a hardcopy and electronic submission for this problem. In addition you will have a demo with Yichun Deng or Song Wang on Monday 11/21, Tuesday 11/22, Wednesday 11/23. *See below for details.*

Required Simulations: After modeling the two versions of the I2C master controller in VHDL using the Quartus II CAD tool, you are to simulate your VHDL specification on a set of input sequences using the guidelines provided below. During the demo, you will be asked to show the output of these input sequences as well as a few other input sequences supplied by the CA. Thoroughly test your design in advance, so that all input patterns applied by the CA work correctly.

Clock Periods and Initialization Requirements. Note the clock period of clk_hi for your simulation should be 10ns.

IMPORTANT NOTE: Although you should follow the above guidelines for the required simulations below, you *still* must be able to correctly handle cases where the master *does not immediately get activated*, and other activity happens first on the bus.

Initialization Requirements. At time t=0, assume the master is *inactive* and in the appropriate initial state, with all signals initialized to *correct initial values*. See Handout #27 and elsewhere for details on initial signal values. Also, at t=0, assume that clk_hi has a rising clock edge, starting the first local clock cycle.

Next, assume that the master is activated (i.e. wins arbitration) immediately afterwards. In particular, in the middle of the first local clock period (at time=2ns), the local counter input "arb_win" is asserted to 1 (for 1 local clock cycle), which begins the activation of the master controller. Follow Handout #27 and elsewhere, for details on the steps and timing of the resulting initialization.

Finally, the master should broadcast a START symbol as soon as possible, immediately followed by a slave's address, for each of the simulations below.

The *SCL clock period* is determined as follows: assume that the local counter always counts 5 local clock cycles between a transition on its input "cnt_enable" to a transition on its "SCL_toggle" output.

IMPORTANT NOTE: Although you should follow the above guidelines for the required simulations below, you *still* must be able to correctly handle cases where the master *does not immediately get activated*, and other activity happens first on the bus.

***You are asked to create the following input sequences for the following cases for* <u>master as transmitter in error-free mode</u>** *(i.e. write mode):*

The following simulations should be performed on your version #1 design (but CA's may also test it on your version #2 design):

(i) After observing the START symbol, the master sends address "1101001", which is the address of a slave, and the 8th bit (0) indicates write, and the slave then responds with an ACK symbol. The master sends a ***single byte of data***, which is immediately followed by an appropriate acknowledgment. The entire transaction is then terminated with a STOP symbol.

(ii) After observing the START symbol, the master sends address "1000010", which is the address of a slave, and the 8th bit (0) indicates write, and the slave then responds with an ACK symbol. The master then sends ***two or more bytes of data***. Each byte of data is appropriately acknowledged, and the entire transaction is terminated with a STOP symbol.

***You are asked to create the following input sequences for the following cases for* <u>master as receiver in error-free mode</u>** *(i.e. read mode):*

The following simulations should be performed on your version #1 design (but CA's may also test it on your version #2 design):

(iii) After sending the START symbol, the master sends "1010111", which is the address of a slave, and the 8th bit (1) indicates read, and the slave responds by sending an ACK symbol. Then the slave transmits ***one byte of data***. After the slave receives an appropriate acknowledgment to this data byte, the entire transaction is terminated with a STOP symbol.

(iv) After sending the START symbol, the master sends "0111111", which is the address of a slave, and the 8th bit (1) indicates read, and the slave responds by sending an ACK symbol. Then the slave sends ***two bytes of data***. The slave receives an appropriate acknowledgment for each of the data bytes, and the entire transaction is terminated with a STOP symbol.

***You are asked to create the following input sequences for the following cases for* <u>master as transmitter with error handling mode</u>** *(i.e. write mode)*:

The following simulations should be performed on your version #2 design:

(v) After sending the START symbol, the master sends address "0101011", which is the address of a slave, and the 8th bit (0) indicates write. The slave responds by sending an ACK symbol. The master sends a ***single byte of data with correct parity***, which is immediately followed by an appropriate acknowledgment, and the entire transaction is terminated with STOP symbol.

(vi) After sending the START symbol, the master sends address "0101011", which is the address of a slave, and the 8th bit (0) indicates write. The slave responds by sending an ACK symbol. The master then sends ***one data byte with correct parity***. The data byte is appropriately acknowledged. After that, the master sends ***the second data byte with wrong parity***. The slave then responds with an appropriate acknowledgment. This data byte is then correctly retransmitted. Finally, the master sends ***the third data byte with correct parity***, which is immediately followed by an appropriate acknowledgment, and then the entire transaction is then terminated with a STOP symbol.

(vii) After sending the START symbol, the master sends address "0101011", which is the address of a slave, and the 8th bit (0) indicates write. The slave responds by sending an ACK symbol. The master sends a ***single byte of data with correct parity***, which is immediately followed by an appropriate acknowledgment. After that, the master sends ***one data byte with wrong parity***. The slave then responds with an appropriate acknowledgment. This last data byte is then ***incorrectly retransmitted***. The entire transaction is then terminated.

You are asked to create the following input sequences for the following cases for **master as receiver with error handling** *(i.e. read mode):*

The following simulations should be performed on your version #2 design:

(viii) After sending the START symbol, the master sends address "0101011", which is the address of a slave, and the 8th bit (0) indicates write. The slave responds by sending an ACK symbol. The slave transmits a ***single data byte with wrong parity***. The master then responds with an appropriate acknowledgment. This data byte is then correctly retransmitted, and is immediately followed by an appropriate acknowledgment, and the entire transaction is terminated with STOP symbol.

(ix) After sending the START symbol, the master sends address "0101011", which is the address of the slave, and the 8th bit (0) indicates write. The slave responds by sending an ACK symbol. The slave then sends ***one data byte with wrong parity***. The master then responds with an appropriate acknowledgment. This data byte is then correctly retransmitted. Finally, the slave sends ***a second data byte with correct parity***, which is immediately followed by an appropriate acknowledgment, and then the entire transaction is then terminated with a STOP symbol.

(x) Design your own input sequence which test something unique in your design not previously test in (i)-(ix), for either master as transmitter or receiver. Do not just vary the address or the data bytes being sent but instead, come up with a creative input case to test something meaningful and the correctness of your design.

Remember, you are responsible for ALL assumptions, clarifications, and information posted on Handout #27e (FAQ), as well as Piazza discussion and other guidelines presented in class, as well as Handouts #27, 27a, 27b, 27c and 27d, and other required web handouts on I2C, so be sure to read all postings carefully.

What to turn in?: Turn in the following written and electronic parts for Project #1:

(a) The symbolic state diagrams of the Moore FSMs for both an error-free master controller (v. #1) and an error-tolerant master controller (v. #2), neatly labeled, either handwritten or using a graphical editor;
IMPORTANT NOTE: If your drawing is messy, hard to read, or poorly labeled, it will be difficult for the CA's to follow, and points will be deducted. The state diagram should be drawn as clearly as possible (i.e. as if it were to be presented for management at a design review). Also, you should ***label small groups of states (with a dotted or colored box around each small group) to highlight which portion of your state diagram is detecting a 0 data input, a 1 data input, an address, an ACK/NACK symbol, a START symbol, and a STOP symbol. In addition, clearly label and circle larger regions of the FSM, indicating 'read mode', 'write mode', and error-handling scenarios.*** That is, annotate small portions of your FSM diagram to clarify what part of the protocol is being handled by which states.

(b) Printout of VHDL code for your Moore FSMs for the two versions of the master controller (with comments inserted to clearly document the different portions of the state diagrams, see 'important note' of (a) above);

(c) Printout of waveforms that result from your simulations for your ten test sequences (i)-(x) above, as well as text details on each simulation (which data bytes used, etc.);

(d) A discussion explaining your design experiences, testing methods, assumptions and challenges (0.5-1.0 pages).

(e) **Electronic copies of your VHDL code and waveforms in a compressed .tar or .zip file named "CSEE4823_proj1_last_name_last_name" (fill in last_name with the last name of each group member). You need to upload the compressed file on *Courseworks*.** ***Details of electronic submission will be announced shortly***.

Where to turn in the hardcopy parts of the assignment? (parts (a)-(d)). Hand these in either at start of class to Prof. Nowick on Thursday November 17, or to one of the CAs by 4pm on Friday, November 18.

Where to send the electronic part of the assignment? (part (e)) Must by submitted on courseworks by 4pm on Friday, November 18 *(details announced shortly)*.

Demo sign-up procedure: You must reserve a 30-minute demo time with a CA. An online 'doodle' URL signup will be made available shortly. *All group members must be present at the demo.* The demo will take place in the Embedded Systems Lab (1235 Mudd) on Monday 11/21, Tuesday 11/22, Wednesday 11/23. The exact time slots will be disclosed shortly.

What to bring to the demo?: Bring a copy of your written parts (parts (a)-(d)) above. Of course, these written parts must already be handed in by 4pm on Friday November 18.