EE 209 Lab 4 – Mastermind Remix

1 Introduction

In this lab you will redesign your 4-bit incrementer to no longer use a sum of minterm (decoder) approach, but rather a minimal 2-level implementation per output. In addition, you will produce logic to compute the next state of the controlling state machine.

2 What you will learn

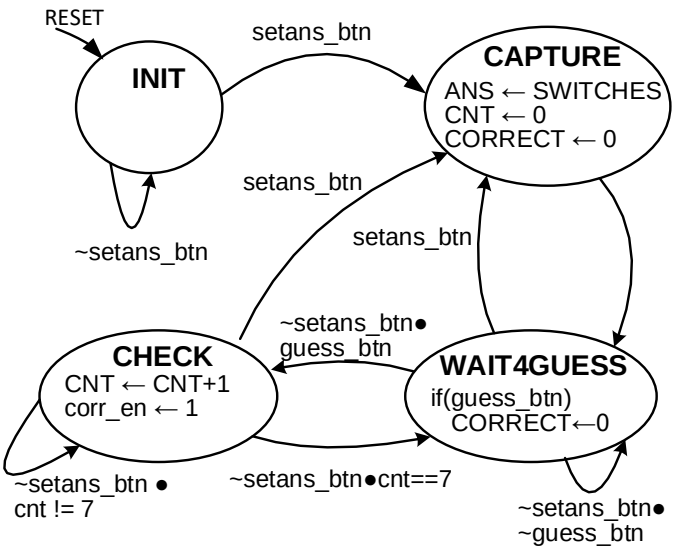
This lab is intended to have your practice finding minimal 2-level implementations by using Karnaugh Maps.

3 Background Information and Notes

The Mastermind Game: Review your previous lab for the gameplay and operation of the mastermind game.

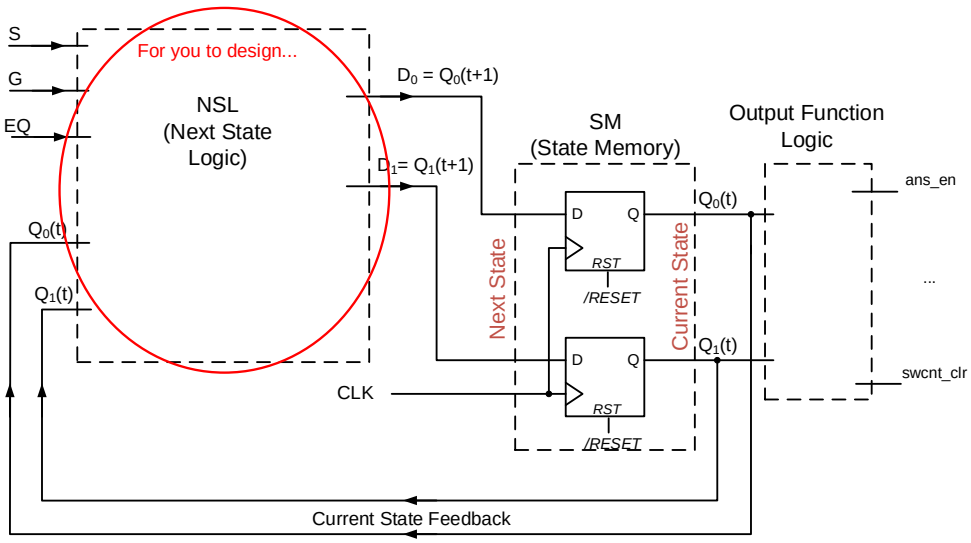
Hints/Reminders:

- In a K-Map you need to cover all the 1's (or 0's) with as few groups as possible but making those groups as big as possible.
- Grouping 1's yields AND-OR expressions (aka Sum of Products / SOP) while grouping 0's yields OR-AND expressions (aka Product of Sums / POS).



State Machine Implementation:
Examine the state diagram for the controlling state machine.

To implement a state machine we use flip-flops (i.e. a register) to store/remember what state we are in. Then surrounding that register is combinational logic to examine the current state plus the inputs and determine the next state (i.e. the state to transition too). We call this the Next State Logic (NSL). Finally, other combinational logic can examine the current state to produce the outputs (aka Output Function Logic).



General Block diagram of a State Machine

To represent the abstract state, we just select binary code/numbers. Since we have 4 states we will need a 2-bit code (and thus 2-bit register / 2 flip-flops). We will refer to these 2 bits as Q1 and Q0. For this lab we will choose this encoding:

Code (Q1, Q0)	State
0,0	INIT
0,1	CAPTURE
1,1	WAIT4GUESS
1,0	CHECK

Your job is to complete the Next-State Logic. To do this we create a truth table showing all combinations of current state and inputs, and determine the desired next state by matching it to the state diagram. This table has been completed for you at the end of this lab. Be sure to read it over and understand how we arrived at the Next state values (D1,D0).

4 Prelab

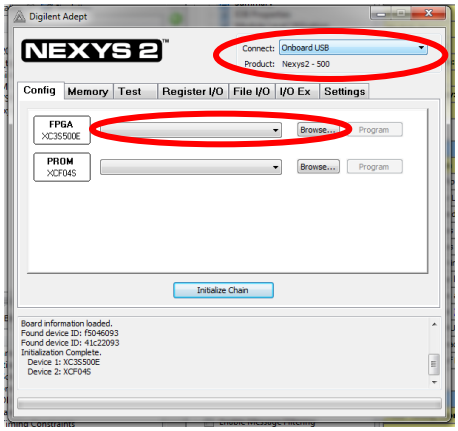
If you would like to finish in lab it would save a lot of time if you wrote the truth table for the 4-bit incrementer as well as the next state function and solved the Karnaugh Maps in advance of attending. Truth tables and K-Maps have been provided on the worksheet at the end of this document.

5 Procedure

1. Download the new mmind_fsm.v file from our website and replace the old version of the file in your project folder with this new version.
2. Complete the truth table (or reference your completed truth table) for the 4-bit incrementer outputs [a blank truth table is provided below].

3. Draw a K-Map for each output:
   a. Output F[0] can be either SOP or POS (or something simpler?).
   b. F[1] and F[2] should be implemented as SOP (AND-OR)
   c. F[3] should be implemented as POS (OR-AND)
4. Now return to your Xilinx design for the master mind game by opening the Xilinx project
      a. Open your inc4.v
      b. Delete your implementation and replace it with the logic you found for each output F[3:0].
5. Now use the truth table at the end of this handout that describes the Next State Logic producing the D1 and D0 flip-flop input logic for the state machine.
      a. Complete the 5-variable K-Map for each output D1 and D0 to arrive at an AND-OR expression for each. Remember these are 5-variable K-maps and you should imagine them as two 4x4 K-Maps stacked on top of each other with adjacencies into the z-axis.
      b. In mmind_fsm.v implement the logic you found for D1 and D0 by instantiating AND, OR, and NOT.
6. Simulate your design and ensure it is still working. We STRONGLY encourage you to find the state bits (Q1 and Q0 in the mmind_fsm component and add them to your waveform. Verify that the state represented by the two bits of Q1 and Q0 really do go through the state transitions as shown in state diagram earlier in this handout.
7. Once you are satisfied that your logic changes are equivalent to your previous working description, re-synthesize, implement and generate the programming file.
   a. To do this, in the Design tab, select the top-level file **mmind_top.v.**
   b. **Important**: In the Processes pane, right-click “Generate Programming File”, click “Properties” and under “Startup Options” ensure that the “FPGA Start-Up Clock” is set to “JTAG Clock”. This should have been done in the 1st part of this lab, but let's be sure to double check.
   c. Now double click the Generate Programming File. It will take some time to synthesize the design and implement it but when it is done double check that there are no errors (look at the errors tab in the bottom console area) or at the color of the icon next to the Synthesize and Implement processes. Warnings (yellow triangles) are fine however.
8. Get a Nexys-3 board from your TA and connect it via USB to your laptop. If you are running on the Remote Desktop (VDI) you'll need have the virtual machine "take control" of the Nexys board by selecting "Connect USB Device" and then choosing "Digilent Onboard USB".

9. Start the Digilent Adept software which is used to download the HW configuration. Make sure the Connect field in the upper right says Onboard USB and the Product is recognized as Nexys3. Finally in the FPGA row, select the Browse button and find the **mmind_top.bit** file in your project folder. Then click Initialize Chain which should configure your hardware and start the program running.



10. Try playing the game on the board and ensure the gameplay still works.
11. Demonstrate your working game to a TA and get their signatures/initials. Sumbit your updated files.

6 Review

7 Lab Report

Name: _____ Score: _____

Due: _____

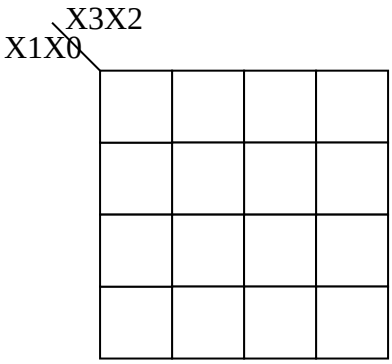
(Detach and turn this sheet along with any other requested work or printouts)

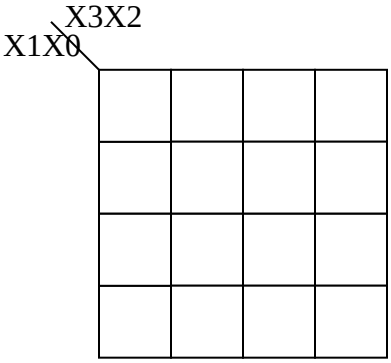
Turn in the following items:

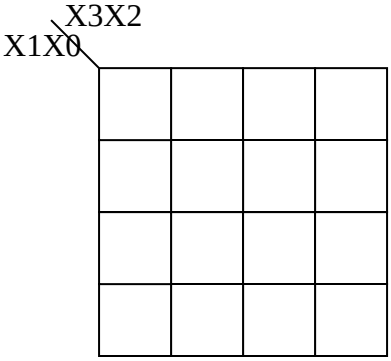
1. TA Signature: _____
2. Completed truth table below.

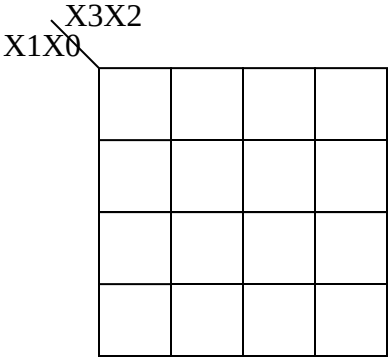
X[3]	X[2]	X[1]	X[0]	F[3]	F[2]	F[1]	F[0]
0	0	0	0				
0	0	0	1				
0	0	1	0				
0	0	1	1				
0	1	0	0				
0	1	0	1				
0	1	1	0				
0	1	1	1				
1	0	0	0				
1	0	0	1				
1	0	1	0				
1	0	1	1				
1	1	0	0				
1	1	0	1				
1	1	1	0				
1	1	1	1				

Write equations on the reverse side:









F[3] =
F[2] =
F[1] =
F[0] =

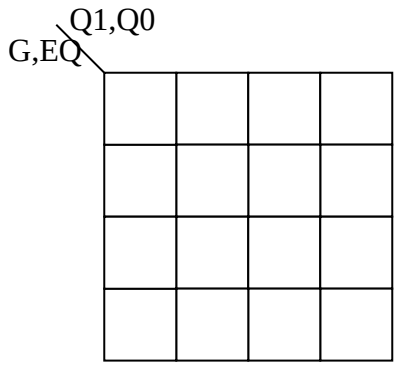
Completed truth table for the D-flip flop inputs of the controlling state machine.

	Current State		Inputs			Desired Next State	
	Q1	Q0	S	G	EQ	D1	D0
INIT	0	0	0	0	0	**0**	**0**
			0	0	1	**0**	**0**
			0	1	0	**0**	**0**
			0	1	1	**0**	**0**
			1	0	0	**0**	**1**
			1	0	1	**0**	**1**
			1	1	0	**0**	**1**
			1	1	1	**0**	**1**
CAPTURE	0	1	0	0	0	**1**	**1**
			0	0	1	**1**	**1**
			0	1	0	**1**	**1**
			0	1	1	**1**	**1**
			1	0	0	**1**	**1**
			1	0	1	**1**	**1**
			1	1	0	**1**	**1**
			1	1	1	**1**	**1**
WAIT4GUESS	1	1	0	0	0	**1**	**1**
			0	0	1	**1**	**1**
			0	1	0	**1**	**0**
			0	1	1	**1**	**0**
			1	0	0	**0**	**1**
			1	0	1	**0**	**1**
			1	1	0	**0**	**1**
			1	1	1	**0**	**1**
CHECK	1	0	0	0	0	**1**	**0**
			0	0	1	**1**	**1**
			0	1	0	**1**	**0**
			0	1	1	**1**	**1**
			1	0	0	**0**	**1**
			1	0	1	**0**	**1**
			1	1	0	**0**	**1**
			1	1	1	**0**	**1**

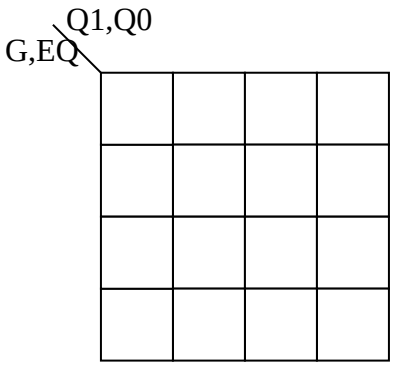
Complete the K-Maps for D1 and D0 on the next page.

5-variable K-Map for D1

S=0



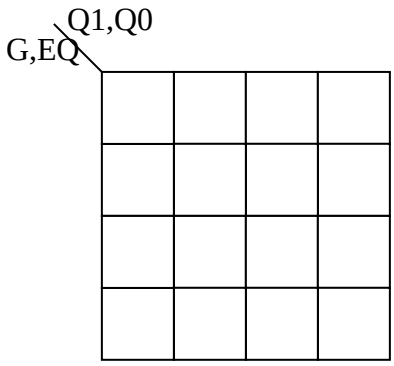
S=1



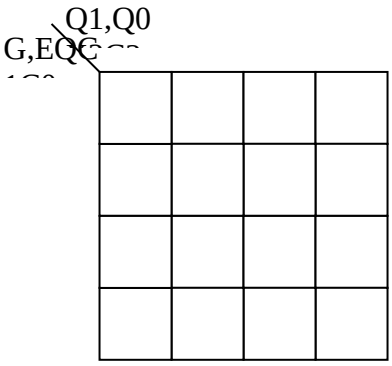
D1 =

5-variable K-Map for D0

S=0



S=1



D0 =

8 EE 209 Lab 4 Grading Rubric

Student Name: _____

Item	Outcome	Score	Max.
Design • Equations F[3:0]	# Correct		4
• Equations for D1 and D0	2 pts. each		4
• Demoed and working	Yes / No		2
SubTotal			10
Late Deductions (-1 pts. per day)			
Total			10
Open Ended Comments:			