San Diego State University
CS 370 Computer Architecture

Lab Assignment 3: The “Gottcha Anti-Theft” Machine
(100 points, weight 15%, due 11:59pm on April 20)

Goal:
The purpose of this exercise is to create Finite State Machine (FSM) which can detect a certain eight digit binary sequence in a continuous serial binary input stream. When the correct sequence is detected, the single output signal should be a logical true value, in all other cases it should be a logical false. This exercise is designed to allow a gradual transition from strictly combinational circuits to a simple sequential circuit. Note that this lab assignment is a group assignment with each group having 2 students. If you need help to find out a partner, please send an email to the instructor.

Problem Statement: *The "Gottcha Anti-Theft" Machine*
There are many anti-theft devices on the market that attempt to foil a would-be robber from starting your car and driving off with it. One popular item has a keypad like a touch-tone telephone. In order to start your car, you must key in a secret four digit decimal code, such as '3719'. It only "remembers" the most recent four digits you have keyed in. Thus the sequence '3723719' will let you start your car.
For simplicity here we use a code based on a sequence of eight bits, and use two push buttons to enter a sequence serially. Each press of a button enters the corresponding digit.

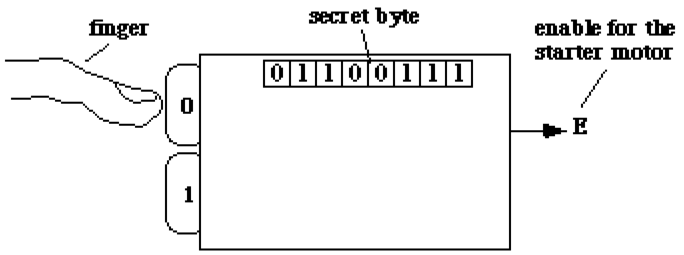

Figure 1. The Gottcha machine in action.

As you key in the bits in sequence, the device outputs $E = 0$ until the most recent eight bits agree with a built-in secret code byte. Then E switches to 1.

The shut-off gottcha: If 16 bits are toggled in without the secret code being observed, the system shuts off and won't accept any more bits. Add a reset button to the system that presumably only the owner could control: when RESET is pressed the system is again enabled and can accept bits.

Design and test (on LogicWorks) the miracle Gottcha machine. Do this in two different ways (thus designing and simulating two different circuits):

1) The **Factory Preset Model:** The specific secret byte preset at the factory MUST be one of your team members’ secret keys. For groups with single member, please use your assigned secret key. A table of secret key corresponding to each student’s RedID is attached

to this document. You can find your secret key from the table. All groups, please explicitly explain which key is used for Factory Preset Model in your Readme file. Your group will receive zero point if you don't follow this key use policy specified above. Do not use registers: solve for the finite state machine with the fewest states and flip-flops possible.

2) The **User-Programmable Model**: This machine allows the car owner to enter a secret byte into a register. The user selects a byte using two hex keyboards, and presses an ENTERCODE pushbutton to store the code word. Now when the machine is used "in the field", the car owner enters a sequence using the same pushbutton arrangement as for the previous machine. When the sequence so entered matches the code word stored in the register, E goes HIGH. As with the Factory Preset Model, if more than 16 bits are entered without the proper sequence being observed, the system shuts off (until RESET is pressed).

Some Hints:

1) These systems have no actual clock - the *release* of either pushbutton produces a transition that is used to trigger the flip flops involved. The main flip flops in the circuit are triggered by this transition.

2) You might put the outputs of the two pushbuttons into asynchronous inputs of a flip flop, which therefore instantly stores the value (i.e. informs "which" pushbutton was pressed) of the newest input bit. The output of this flip flop is then used as the actual "input" value.

3) Use a shift register in part 2 to store the most recently received bits. Compare the shift register output to the programmed code word. (So part 2 is *very* simple.)

Submission Instructions:

a) A one-student group only needs to complete the **Factory Preset Model**. A two-student group needs to accomplish both the **Factory Preset Model** and the **User-Programmable Model.** For one-student groups: please find out your assigned 8-bit binary secrete key associated to your SDSU RedID in the table attached to this document. For two-student groups: please use an 8-bit binary secrete key assigned to one of your members in the table attached; Please tell us which member's secrete key is used.

b) Write a Readme text file that includes (1) your group member names and the contribution of each member (who did what); (2) list each file enclosed in your submission package and briefly explain its function and usage.

c) A brief description of the design process you used to obtain the circuits. Be sure to point out why you chose a particular design, along with the steps you took to minimize the amount of hardware required in each case.

d) Your LogicWorks files including your circuit schematic file (.cct) and your library file (.clf). In your .cct file, please document each section using the Text tool (Ctrl_E) so that the grader knows how it works. **Note that documentation is essential for full credit.**

e) Zip all your LogicWorks files and the Readme text file into one document and name it using the first initial and the last name of one of your group members. (e.g., assume that Nathan Adams is in your group, your zip file should be NAdams.zip). Note that only one member should submit this lab assignment on behalf of the entire group.

f) Submit your zip file from within the Canvas.

Fixes for frequent issues from students:

1. When a correct sequence is detected, the lock counter should reset back to 0.

2. You should enter your input with 2 push buttons, one representing the input x=0, and one representing x=1.

3. If the flip flops don't update after you press an input button, try adding a 'buffer' piece before the clock ports.

4. It is recommended that you use the flip flops with just the D, clock, Q, and reset ports. Keep in mind that these flip flops will reset when the reset input signal is 0.

5. In some circuit implementations, the output will only stay 1 for as long as the user holds down the final input. This is perfectly fine!

Rubric:

1-Person Groups:

	20 points	90%	98%	100%
Lab Functionality 90% (90 points)		The 16-bit lock does not work correctly.	The 16-bit lock does not reset after the correct input is typed in.	Lab circuit works perfectly with no problems.
Documentation 10% (10 points)	If the state table, state diagram, and optimized equations are provided in a document and they are all correct, you will receive a score of 20%.			Readme file contains students name, redid, and secret code. This should be on the testing lab circuit as well as descriptions on the part.

2-Person Groups:

	20 points	90%	98%	100%
Lab Functionality (Factory Preset		The 16-bit lock does not	The 16-bit lock does not reset after the	Lab circuit works

Model) 60% (60 points)		work correctly.	correct input is typed in.	perfectly with no problems.
Lab Functionality (User-Programmable Model) 30% (30 points)				Lab circuit works perfectly with no problems.
Documentation 10% -10 points)	If the state table, state diagram, and optimized equations are provided in a document and they are all correct, you will receive a score of 20%.			Readme file contains students name, redid, and secret code. This should be on the testing lab circuit as well as descriptions on the part.

Secret keys:

RedID	Lab #3 key
824500931	01110000
823389093	01101111
817330469	01010100
822440678	00110101
825573964	00011011
821124649	01001101
823971415	01000110
824709737	00011001
824554530	00111110
825991524	00110001
825530141	00111001
824089247	01100100
820619521	00001110
823479079	01001010
826985322	00010100
822744267	01000100
822366812	00000010
822789091	00011101

823476115	01001001
826173017	01001010
826465413	01101010
823486177	00111001
826093613	00011000
826333346	00000011
825631827	01001001
826370149	00010000
822506718	00010001
823638459	01111110
825575927	01110011
824131718	00001000
818749029	01110000
821634795	01111010
821272563	00100110
826404183	01110011
822763806	00011110
824263304	01111011
826403260	00000000
826620932	00110111
826622622	00100111
822801038	01011100
821907561	01110010
822531171	00100101
817070664	00101110
822857354	01110101
824445603	00000011
819264921	00010111
821596445	00111110
822500101	00110001
823663471	00101110
824208379	00000110
826629499	00001100
823560225	00111010
822202934	01000100
826472030	00000111
823909470	01101101
826003692	01101001
824304553	01001000
823559237	01001000
823701808	00110011

824385309	01111000
826563732	01111110
824155391	00001011
825895805	00001000
825145653	00001100
823109099	01111110
826482963	00010010
825331124	01010101
826008892	00110101
826003653	01011000
824193403	01110110
824996218	01011101