San Diego State University
CS 370 Computer Architecture

Lab Assignment 3: The "Gottcha Anti-Theft" Machine
(100 points, weight 13%, due 11:59pm on April 9)

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.

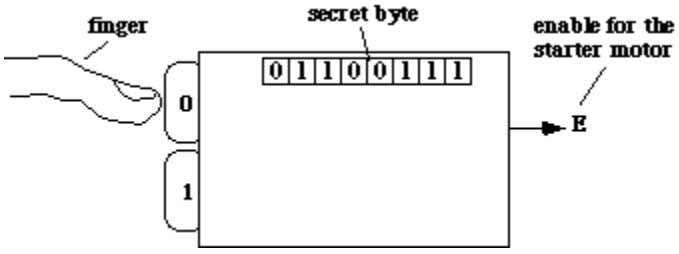

Figure 1. The Gottcha machine in action.

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 Tony La is in your group, your zip file should be TLa.zip)

f) Submit your zip file from within the Canvas.

Secret keys:

RedID	Lab #3 Key
823121696	10001000
825140518	11000000
818292365	11000100
824812385	11101000
824223654	10010000
813119210	11111111
823621182	10001000
825571325	11100001
818449496	10100010
820591948	10100010
825619438	11111000
821268507	10010000
823979085	10011100
823622118	10100101
823170758	11101011
822789091	11001001
825609688	10010110
821465405	11110110
822918155	11010100
821253947	11011000
823584782	10101000
823941905	10100110
821775962	10100100
822508746	11001000
820087964	11111100
822602112	10011000
822727692	10100101
820715175	11001111
822254518	11100110
824994424	10100000
824333283	11100000
823622872	11001001
821009794	11110100
822628762	10111100
822439196	11011000

823853596	10111010
813819364	10111111
821685391	11110010
821652137	11110110
822489428	10111100
825538318	10100110
821385624	11011000
822560590	10011000
824453377	10111000
821331115	10111000
821404162	11000111
822656621	11001100
823631439	11011011
825101687	1000111
819434948	10001100
823998533	10100000
822607845	10100110
824282700	10101110
821770463	10100001
817290273	11010110
824060322	11000001
818278182	11110101
825966447	11001000
824357515	10000000
824267724	11101010
820476118	11010000
824031774	10110001
817314271	11010110
825192206	10100000
825223653	11100110
822380592	10001010
822298094	11110000
822591933	11001100
823660429	11010100
823181886	11011000
822942790	11100100
822112545	10100100