Building a custom car alarm for better security

Overview

As hardware engineers, 6.111 students see unique possibilities for quality-of-life improvements. After becoming a car-jacking victim, Professor Gim Hom recognized one such opportunity. He decided that a factory-standard car alarm just would not suffice. After all, thieves know the ins and outs of these security systems. He realized that he could achieve security through obscurity, by building a custom system that only he understood. The design needed to be robust, but not complicated. He soon thereafter developed the car alarm implemented in this lab.

In addition to the standard siren-sounding functionality of all car alarm systems, Professor Hom designed a secret method of controlling power to the fuel pump. Furthermore, he wanted to build the system to be a pleasure to use, by including allowances various common actions. For example, he wanted to be able to reopen a door "soon enough" after exiting his vehicle. He also wanted to be able to open the door for his wife, enter the car himself, and start the ignition before the alarm went off.

In this lab, we build a system to Professor Hom's specifications on an FPGA. We will control most of the system logic using a finite state machine. In addition, we will configure a block to generate a sound signal to be used as a siren. Furthermore, we will build a block to enable precise timing control. Lastly, we will build control logic for the fuel pump (the "secret sauce" that makes the system so secure). In this case, we simulate all human actions (e.g. opening a door) by using various switches and buttons on our FPGA.

DESIGN DESCRIPTION

Conceptually, we'll consider our system as an interconnected set of four main modules. The modules, and the connections between them, can be observed in Figure 1. First we will discuss the anti-theft FSM, followed by the audio/visual generators, followed by the timing control, and finally the fuel pump control.

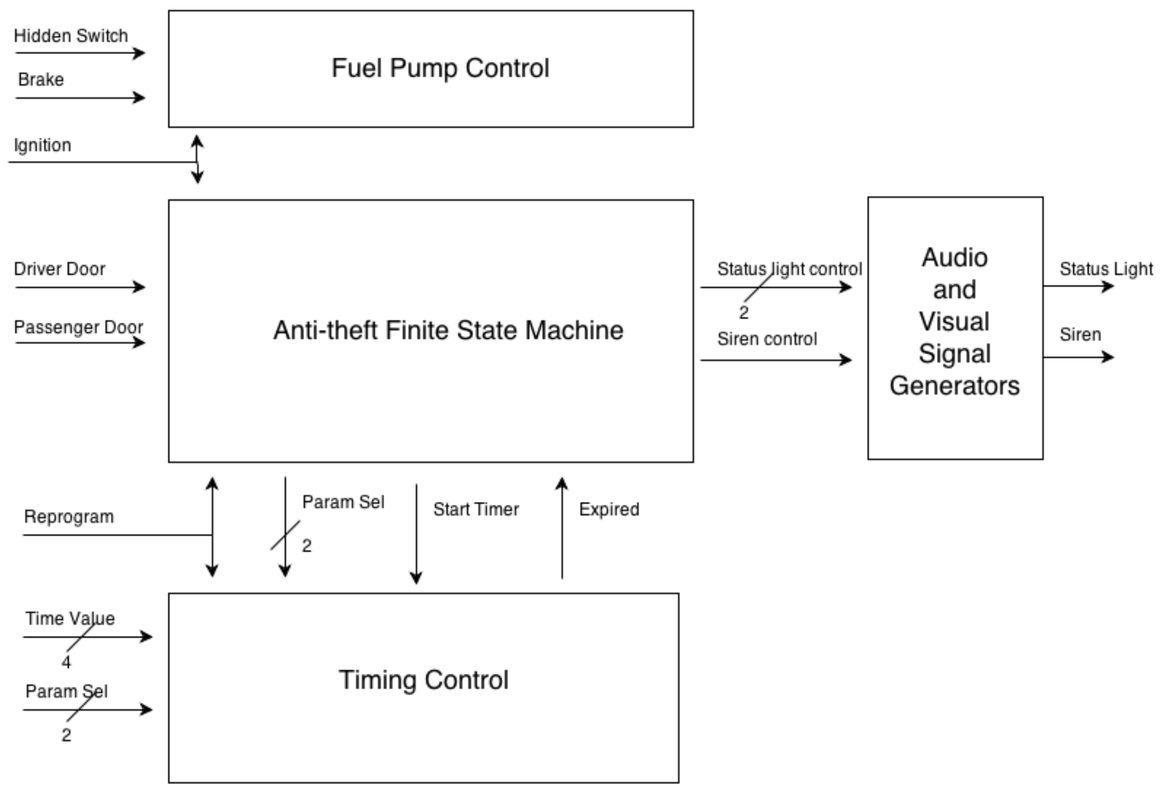

Figure 1.

The anti-theft finite state machine has four operating modes, which are described below.

1. Armed. This state represents the state of the car when it is parked and the alarm is armed. It is the default state of the system upon start-up. The status light should be blinking with a two second period. The siren should be off. Upon turning on the ignition, we move to “Disarmed mode”. Upon opening a door (either driver or passenger), we move to “Triggered mode”.
2. Triggered. This mode represents the time in which the alarm waits (to give the legitimate passengers time to enter the car) before sounding. The length of this wait is dependent on which door-opening triggered this mode. The driver door and passenger cause delays T_DRIVER_DELAY and T_PASSENGER_DELAY, respectively. The status light is constantly on and the siren if off. If the ignition is turned on, we move to “Disarmed mode”. Upon the expiration of the timer, we move to “Sound Alarm mode.
3. Sound Alarm. In this mode, the alarm is sounding. The status light is constantly on and the siren is sounding with alternating pitch (400Hz and 700Hz). It alternates pitch every second. When both doors are closed, we begin counting. After a certain amount of time (T_ALARM_ON) for which both doors have been constantly closed, we move to “Armed mode”. If at any point the ignition is turned on, we move to “Disarmed mode”.
4. Disarmed. This is the mode in which the driver is inside the vehicle. The status light and siren should both be off. After the ignition has been turned

off, the driver's door has opened, and the driver's door has closed (all in that order), a timer will start with time (T_ARM_DELAY). After the timer expires, we move to "Armed mode". If a door is opened before the time expires, we restart the time and continue again from T_ARM_DELAY once all doors are once again closed.

Note that some of these modes require multiple states in the FSM to implement. The Sound Alarm mode requires two states. The first of these states represents the time when a door is opened. When all doors are closed, we move to the second Sound Alarm state and start counting. If a door is opened when in the second Sound Alarm state, we return to the first one. The Disarmed mode requires three states. We transition from the first to the second upon the ignition being turned off. We transition from the second to third upon the driver's door closing. In the third state, the countdown is running. If a door opens again, we return to the second disarmed state. If the ignition is turned on we return to the first disarmed state. In the Verilog code, the state machine is quite simple and can easily be described by a "case" statement.

The audio/visual signal generating logic is considerably simpler. The finite state machine sends a simple command to the generator module, which them implements the desired signal. In the case of the status light, the finite state machine specifies one of three possible commands ("on, "off", "blinking") and the generator creates the correct signal. For the siren, the finite state machine simply specifies "on" or "off". The siren generator creates a square wave, the frequency of which alternates between 400Hz and 700Hz every second. In order to do all the timing necessary for the audio/visual generators (alternating blinking, pitch, or the sound signal itself), the generators use clock divider modules that will be explained as part of our description of the timing control module. These modules allow us to create constantly alternating signals at a much lower frequency than our clock.

The timer control module is comprised of three sub-modules: a time module, a parameters module, and a divider module. The parameters module has four reprogrammable parameters, as described in the FSM section. I simply store the parameters in a register and select the correct register based on which parameter is being asked for by the control lines. The logic to "reprogramming" parameters is straightforward. The FSM selects a parameter, whose value is constantly constantly being fed to the time module. The time module, upon receiving a "start_timer" pulse from the FSM, begins counting down from the value specified by the parameters module. When time expires, it tell the FSM with a pulse. Lastly, the diver module creates a pulse every N clock cycles, where we can specify N. This allows the timer to get a 1 Hz input pulse so that it can count accurately. The timer also synchronizes the divider to make sure that the first second is a "full" second.

Lastly, the fuel control logic is extremely simple. Upon three conditions being simultaneously met (ignition on, brake pressed, hidden switch pressed) the fuel pump will power on. When the ignition turns off, the pump powers off. The "hidden switch" is Professor Hom's key innovation in deterring would-be car thieves. The thieves are unlikely to know about the existence or functionality of the hidden switch. In hardware, this control logic can be modeled by a two-state FSM (states are "on" and "off").

CONCLUSION

In this lab, we explored some key 6.111 concepts. Most importantly, we used a finite state machine to drive the critical logic of the system. We also learned how to implement other important structures. We can now, using a divider module, create signals with any frequency that is an integral divisor of the clock frequency. We also, thanks to the divider, can create signals that represent sound! I feel that this design is very flexible and robust. I think, in the future, it would be interesting to make it somewhat more complicated, especially the siren. Any number of cool or useful sirens can be implemented.