Team: FPGA
Project: No One is Safe!

Members:
Alex Dunmire
David Kim
Michelle Dean
Rohan Nagarkar

Overview:

"No One is Safe" is an RC laser tag game, and an attempt to replace the conventional remote-control functionality of an RC car with an FPGA via Bluetooth control through an Android device. The digital signals governing the motion of the RC Motor Controller and Servo were characterized by an oscilloscope and then replicated via clock-division by the FPGA. A custom Android application was developed to enable throttle control through a slider, servo-control through the android device's accelerometers, and the ability to fire an affixed laser diode with the push of a button.

Code:

Verilog

Top.v

The top module organizes submodules into single cycle triggered modules. In addition, the top module stores the eight bit packets received by bluetooth and stores in the temp register. This register is parsed into the controlling input bits for the state machines used to control steering, throttle and laser fire.

bluetooth.v

This module connects the FPGA with our Android app. The bluetooth hardware module is a uart device which we set to operate at a BAUD rate of 9600. The verilog module shifts the eight serial bits received by the module into eight individual registers. These eight packets are then parsed by the top module and distributed state machines for driving control and laser fire.

clock.v

With the pmodbt2 set to operate at Baud rate of 9600, the FPGA receiver needs to ready to reveive at the same rate. Clock.v generates the 9600hz clock (BAUD RATE) rate for the bluetooth module.

steering_pwm.v

To emulate the RC car controller and receiver, we implemented three state machines to control the steering, throttle and laser fire. Using an oscilloscope we measured signals on a period of 20ms and the controller inputs changed the duty cycle. For example, pulling the trigger of the throttle controller changed the duty cycle from 1.4ms to 2ms.

To replicate the 20ms period we divided the clock, 100Mhz, by 1953 to achieve a 52kHz clock then again by the 50kHz clock by the 9 bit value PulseCount. Dividing the clock twice achieves a 20ms pulse and the ability to control the duty cycle.

laser.v

Laser control is taken care of with a two state state machine. The input is the fifth most significant bit of the 8bit packet sent by the app.

Android

(The application is broken down into the following subdirectories):

accelerometer

Steering control is implemented using Android tablet's accelerometer. To establish a noiseless steering control, a Kalman filter was used by interpreting the change of accelerometer values as opposed to the raw inputs received by the sensor. The filtering algorithm was found online (reference documented in source code file AccelerometerHighPassFilter.java), and through experimentation, the optimal constant k-filter was float value 0.65f. This was tested across multiple devices to make sure it was consistent.

bluetooth

Establishing a connection with the vehicle was handled using Android SDK's built-in Bluetooth support library. During initial trials of trying to establish communication, there was some ambiguity as to whether the Bluetooth pmod module behaved as a server or client (different protocols in establishing socket). It our case, it behaved as a server, and the Android device was the client. Once connected, two separate threads are used to send and receive bytestream. The following instruction syntax is used, and outputted to the connected bluetooth socket:

Steering	Bits[1:0]	00 Neutral 01 Right 10 Left
Throttle	Bits[3:2]	00 Stop 01 Forward 10 Backwards
Lasers	Bits[4]	0 No Fire 1 Fire

When trying to utilize the same output that was used for button controls, we came across a problem in such that the vehicle wasn't receiving clear signals. The default configuration that was used

to transmit signals was to output the state of throttle, steering, and fire button in 20ms intervals. Experimentation in manipulating data at faster or slower rates have not yielded better performance. The exact reason of this interrupted signal is unclear whether it is the bluetooth peripheral module or the data being outputted from controller to device. One speculation is that the current byte per instruction is not producing a consistent stream of data due to the propagation delay that causes interruptions between each signal, or noise interruption. While the output is sent at every 20ms, the bluetooth baud rate is at 9600. Possible solution to resolving this issue would be to increase the instruction to three bytes per instruction.

slider and button

A modified Seekbar View widget was utilized to allow lateral directional control to correspond to the vehicle's throttle state. When throttle is released, the thumb is set to return to stop position gradually.

application

The structure of the Android application is modularized to handle controller-related functions in separate source files. These are then implemented at the Main Activity level, declared as static to allow configurations to persist across views.

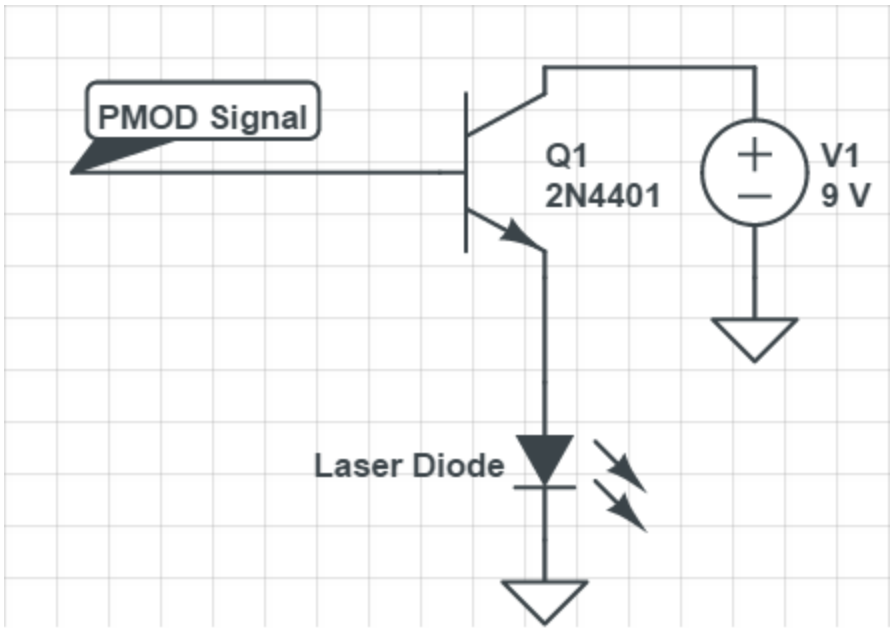
Special Instructions:

Battery-Powering and Pre-Programming the FPGA

First, generate .mcs file in iMPACT. The Nexys3 has an NP5Q128 SPI device, which is not an option in iMPACT for devices, so generate the .mcs for N25Q128 and load the .mcs file to the FPGA SPI PROM through Adept. Once Adept has finished programming the SPI PROM, the FPGA can be shut off and hooked up to a battery.. For our battery, we used the Anker 2nd gen Astro3 hooked up through the USB Prog port on the Nexys3. Once hooked up to the battery, make sure the J8 mode selection is jumped across M0 to enable booting from SPI. Once this is done, the FPGA can be turned on and will configure according to the project loaded in SPI PROM.

Attaching Laser Diode to PMOD Ports

The PMOD signal pins on the Nexys 3 board provide an output voltage of around 2VDC, which lies below the suitable operation range for most laser diodes. The 5mW, 650nm laser diode we used came with an integrated driver which operated at a minimum input voltage of 2.8VDC. Driving the laser diode with less than the required input voltage resulted in a very faint beam. To solve this issue, we implemented a level-shifter circuit with an 2N4401 NPN BJT connected in-between a 9V battery and a laser diode (see schematic below). Logic high signals from the PMOD are used to switch the transistor on to allow current to flow from the battery into the diode, thus turning the laser on; logic low from the PMOD electrically isolates the battery and diode, preventing current flow, and ensuring the laser is off.


PMOD Signal
Q1
2N4401
V1
9 V
Laser Diode