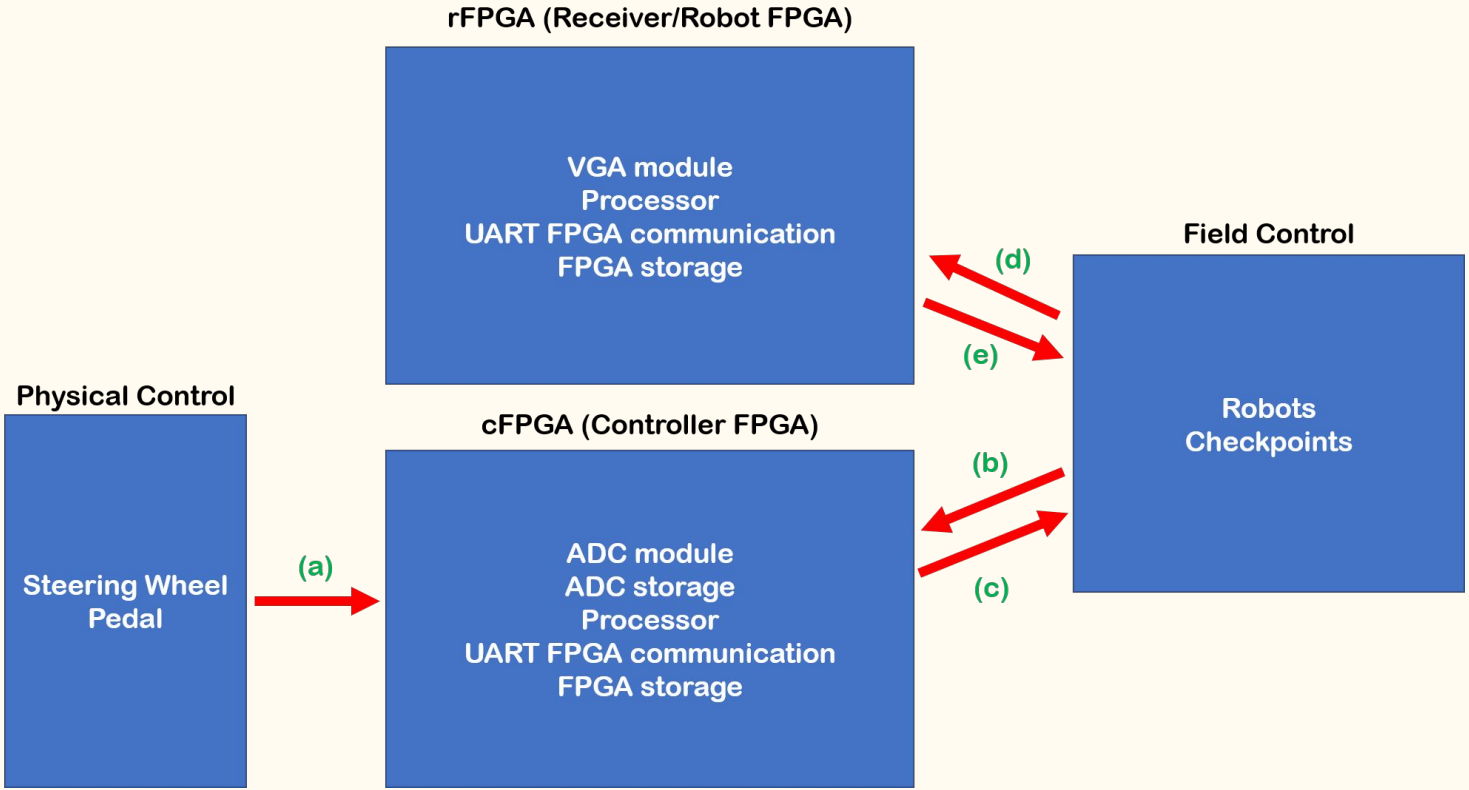
Robot Racers

Gerry Chen, Faith Rodriguez, and Aditya Sridhar

Project Overview

In this project, two remote controlled robots race around a game space trying to run into checkpoints in order to earn points and to trigger power-ups.

Project Schematic



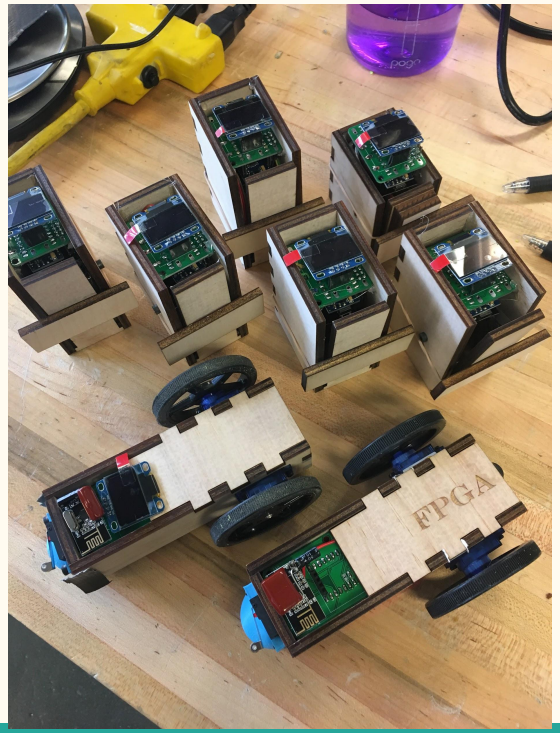
Key Features

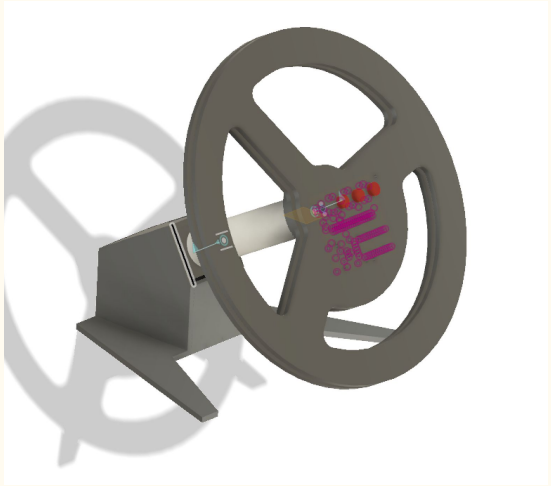
- Adopts a hardware-intensive approach
- Creates modularization of Verilog code into well-defined storage units to facilitate the flow of robot control
- Implements the five-stage pipelined processor with full bypassing from project checkpoint 4
- Incorporates Universal Asynchronous Receiver/Transmitter (UART) communication for multi-FPGA gaming
- Uses Arduino to control the robots and transmits and receives input using UART communication

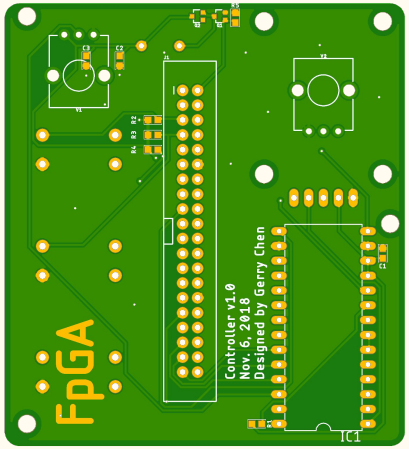
High Level Design

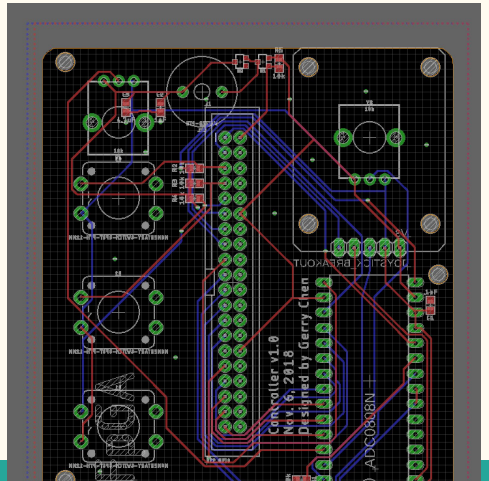
- Physical Control
  - Steering wheel and pedal control the steering and throttle for each robot, respectively
  - Values are passed into an Analog-to-Digital Converter (ADC) to translate motion into a computable digital value
- Controller FPGA
  - Converts digital values received from the robots via the ADC and stores the values in an ADC storage (registers)
  - Uses UART communication to send and receive information about each robot's state (speed/throttle and steering)
  - Handles speed/steering power-up duration logic for robots and stores motor effects
- Receiver/Robot FPGA
  - Determines and stores point values after robots touch a checkpoint
  - Outputs the robots' points to the VGA
  - Uses UART communication to receive information about each robot's state (points)
- Field
  - Contains robots and checkpoints that are controlled by the cFPGA
  - Robots and checkpoints interact with each other using Arduino

Building/Designing









byte 0	byte 1	byte 2	byte 3	byte 4	byte 5	byte 6
time[27-21]	time[20-14]	time[13-7]	time[6-0]	command	payload[13:7]	payload[6:0]

sender	receiver	command		payload	
		6 5 4 3 2 1 0		13 12 11 10 9 8 7	6 5 4 3 2 1 0
cFPGA	robot	000	robot ID	left motor vel	right motor vel
robot	FPGAs	001	robot ID	touch duration	
checkpoint	FPGAs	010	checkpoint ID	touch duration	
rFPGA	cFPGA	100	game state	auxilliary info	
cFPGA	rFPGA	101	powerup state	auxilliary info	

Processor Logic

- Performs point-calculations for each robot
- Converts ADC digital values for speed/throttle and steering into left- and right-motor velocities
- Implements additional custom functions for reading from and writing to storage units:
  - Times and touch durations for robot-checkpoint touches found in a communication storage
  - Power-up time effect timeouts for robot-checkpoint touches found in a timeout storage
  - ADC digital values found in an ADC storage
  - Game time monitored during the game; used primarily in power-up effect calculation
- Instructions designed using MIPS Assembly

VGA for Robot Game

- A VGA display was created to keep track of the scores of both players
  - Images were converted to MIF files that were strategically chosen to be displayed
  - A decimal to binary converter interprets each value to choose which digit's MIF file should be displayed
- Since point calculation was not a feature that we were able to complete, player 1's score is simply a counter that aligns with game time in order to show that the VGA can in fact update in response to inputs

```
# Cycle through robot/checkpoint combinations
robotlpl:
imova $6, $1, 0     # Robot 1 current speed from ADC; "imova" is a custom function that extracts data from ADC storage's RF $1
imova $7, $2, 0     # Robot 1 steering from ADC
addi $7, $7, -64
imova $8, $3, 0     # Robot 2 current speed from ADC; "imova" is a custom function that extracts data from ADC storage's RF $3
imova $9, $4, 0     # Robot 2 steering from ADC
addi $7, $7, -64
imovp $10, $1, 0    # Robot 1 time; "imovp" is a custom function that extracts data from FPGA storage's RF1 $1 and stores in processor's $10
imovp $11, $2, 0    # Robot 1 touch duration
imovt $12, $1, 0    # Robot 1 powerup effect cycle comes from the FPGA storage's RF2 $1
imovp $13, $5, 0    # Checkpoint 1 time
imovp $14, $6, 0    # Checkpoint 1 touch duration
time $15            # "time" is a custom function to get the current time/cycle in the game
lw $16, 0($0)
lw $17, 1($0)
lw $18, 2($0)
lw $19, 3($0)
lw $20, 4($0)
lw $21, 5($0)
lw $22, 6($0)
lw $23, 7($0)
lw $24, 8($0)
bne $12, $0, robotlplcheckpart1
j robotlp2
robotlplcheckpart1:
blt $15, $12, applyExistingOplpl
blt $10, $13, robotlplcheckpart2
sub $5, $10, $13
addi $5, $5, -500
blt $0, $5, robotlp2
j robotlplcheckpart3
robotlplcheckpart2:
sub $5, $13, $10
addi $5, $5, -500
blt $0, $5, robotlp2
robotlplcheckpart3:
blt $11, $14, robotlplcheckpart4
sub $5, $11, $14
```

Snippet of cFPGA logic