Field Programmable Gate Array Robot

University of New Mexico

Electrical and Computer Engineering

Justin Bitchtits Johnston, Ryan Fenn

*Abstract*—This paper discusses the construction, testing and results of a FPGA robot. Due to the previous high cost of FPGA’s little work has been done on small-scale robotics that utilizes FPGA’s. This project also demonstrates the ability to interface other control devices such as the open source microcontroller Arduino.

**I. INTRODUCTION**

There are very few references to FPGA robots in existence, so the concept and construction is heavily based upon robots previously produced with Arduino components. The main purpose of this project was to build a robot that utilized the concepts of ECE 238 and provide a model to demonstrate the versatility of the programming language VHDL. This paper describes the construction and testing process that enabled the authors to produce a fully functional FPGA robot equipped with bumper and infrared sensors to interact with its environment.

**II. CONCEPT**

The digital components for this project can be broken down to 3 primary parts and the hardware can be broken into 4 parts. A Pulse Width Modulator (PWM), State Machine, and low-level gate logic to provide enable signals. The PWM creates a signal output that the user can very closely control the duty cycle. This allows the user to change the average voltage going to the external H bridge circuits, ultimately controlling DC motors speed. The state machine is the component that determines which drive signal is sent to the low-level gate logic component. Depending on the current sensor reading, the low-level gate logic circuit matches a drive signal with a PWM signal to send 11V to the respective H bridge and then to the servo.

**III. LOGIC SYNTHESIS**

*A. The Pulse Width Modulator*

A brief overview of the operation is that the PWM creates a variable duty cycle

square wave to decrease or increase the average power to the DC motors. This allows a wide range of speeds that can be obtained from the motors. I wanted a way to change the duty cycle without having to manually go into the code and change the preset threshold. To resolve this issue we used the switches that are on the Nexys 3 board to change the pre-set value. This allows for quick adjustment of the duty cycle without excessive down time due to recompiling the code. Next, the state machine takes into account all of the

sensor inputs from the IR sensors and/or bumper switches to determine which state the robot should be in. The current state is then displayed on the 7 segment display which allows for easy trouble shooting and it also sends out the various enable signals to allow the PWM signal to be switched to the appropriate motor and with the needed polarity to provide forward or reverse motion of the robot.

*B. The State Machine*

Extra states were needed in the state machine because the Pmods H-bridges

cannot be switched from forward operation to reverse immediately. The transistors in the Pmods have a slight amount of delay time for both switching on and off that if the

polarity is immediately switched there is a potential that a short across the bridges will

occur and the Pmod will be destroyed. To combat this issue I created 2 extra states that

contain nothing but a counter. These extra states will count to a preset variable and then

switch into or out of the reverse state. This delay time was calculated from the data sheets of the Pmods to ensure that enough safety margin existed to prevent any damage.

*C. Low-Level Gate Logic*

The low-level gate logic is extremely simple and consists of simply one AND gate. The function of the simple AND gate was to match the drive signal from the state machine to the voltage signal from the PWM. When both signals were true, voltage was then sent to the H-bridge, through a voltage regulator and finally to the servos.

**IV. Hardware Design**

*A. FPGA Board*

We decided to use a Nexys 3 board because of its external I/O ports and wide availability. The Nexys 3 is also compatible with a wide array of available and cost effective peripheral boards (P-Mods). This board was also used to keep continuity with the ECE 238L.

*B. P-Mods*

The external peripheral boards were an ideal solution for our robotic application. The P-Mods we implemented in our design were the Digilent PmodHB3 2A H-Bridge Module. The H-bridge circuit allowed us to apply a voltage across a load in either direction. This let us run the DC motors to either forwards or backwards. Using an integrated circuit was especially useful rather than using discrete components because of its simplicity and portability.

*C. External Sensors*

Our original design employed two forward facing bumper switches. These switches were effective at detecting low lying objects and walls. When the switch was depressed, a 5V signal is sent to the state machine for processing. After the initial robot was built, phase II consisted of implementing infrared sensors. The Nexys 3 does not have any on-board analog to digital conversion capabilities. We decided to use an Aurduino board to make the conversions for us rather than a third external P-mod because of its affordability and availability. We also decided that addition of a second board could help future students increase their exposure to the Aurduino language. The analog 5V signal from either of the 3 IR sensors is converted to a digital signal. That signal is passed to the state machine for processing.

**V. Testing**

*A. Testing the PWM*

After the initial PWD VHDL code was written, the only way we could test our design was with an oscilloscope. With the oscilloscope, we could view the pulse width of the square wave and adjust the duty cycle with the external switches on the Nexys 3 board. We were able to fine-tune our code to achieve nearly maximum voltage.

*B. Testing the State Machine*

The state machine, combined with the AND gate, was initially verified by using an oscilloscope. We triggered the bumper switches and verified that the duty cycle signal was not allowed to be sent to the servos. We also designed a decoder that displays the state on the Nexys 3’s 4 digit LED’s. This was particularly useful when trying to troubleshoot the IR sensors.

**VI. Conclusions**

This project took considerably more time than the 2 weeks given to build the

“Counter” final project. Most of that extra time was due to there being very little hobby

level FPGA robot data available. Since FPGA hardware has been considerably more

expensive then microcontroller hardware, such as the Arduino, very little has been done with it in terms of small-scale robotics. It took a considerable amount of time and

research to determine which motors and sensors would be compatible with this

combination of hardware. Overall this project has been a success. The project

demonstrated the application of decoders, state machines and multiplexers. It also

demonstrated the ability of the Nexys 3 to intercommunicate with different platforms that utilize different programming languages. Lastly, It showed that through programming, it is possible to take very different styles of hardware and allow a digital system to interact with the physical world and provide a desired response.

**REFERENCES**

**We should probably reference both of our text books so we don’t look like we just googled some shit and ended up with a robot.**