

Open Loop Robot Arm Controller

Charles Jarvis

July 31, 2008

1 Introduction

The purpose of this project is to achieve open loop control of a five-motor robot arm using digital logic (via VHDL) as both an interface layer/driver as well as a means of controlling the system independently. This simply means that this controller was created, designed, and layed out in such a manner that it can easily be used and programmed using push-buttons and switches on an Altera DE2-70, but at the same time can be connected to a PCI bus and be controlled just as easily.

2 Background

The robot arm used in this project is an OWI-535. It uses 5 motors to control its movement: 1 motor on the base to allow the arm to rotate, 3 motors at the joints, and 1 motor that allows the gripper to open and close. Each motor module on the arm is geared so that the small motor can provide enough torque to move the arm under load.

The process for designing the system was very modular and systematic. The first milestone in the project was getting the motor to work in a purely analog fashion. This involved two steps: Constructing the motor kit and using the joysticks that came with it to control the motor, and then replacing that with a motor driver IC. From there, a single motor control using digital logic on an FPGA was created. After control of a single motor was attained, the program was expanded to control 5 motors.¹

3 The Final Design

The first part of the design centered around the electronic components (integrated circuits, wiring) needed to translate the signals from the FPGA into the motion of the arm. This required finding the appropriate drivers to handle motor direction switching and source the current each motor would need to work. This also meant that a circuit need to be made that would allow a 'plug-and-play' interface between the FPGA and the motors. The second part of the design focused on making a user interface that would allow independent control of motor speed and direction for five motors, while either operating one motor at a time or all five simultaneously.

¹It should be noted that the scope of this report covers only the final product and not the agonizing hours of staring at a computer screen wondering what went wrong or fixing failed attempts.

3.1 Analog Design

Testing showed that each motor, loaded only by the weight of the arm, drew anywhere from 300mA to 400mA at 5V, and spiked anywhere from 800mA to 1A when starting up or switching direction. The SN754410² made by Texas Instruments was chosen as the motor driver. The SN754410 (hereafter, driver) is a moderately powerful quadruple half-H driver, which (obviously) means that it has 4 halves of an H-bridge on a single chip. For this project, it is used as 2 full H-bridges, thus requiring 3 chips to run 5 motors.

The driver separates the motor logic from the H-bridge itself, thus requiring two separate power sources for each chip. In the case of this project, one of the Extech Regulated DC Power Supplies on the 3rd floor of MEB can provide enough power and current to satisfy the needs of all six power inputs. The top and bottom inputs 1A and 2A control the top and bottom halves of one of the H-bridges (On the right side of the chip, 4A controls the top and 3A controls the bottom). Figure 1 shows the two main operational modes of the H-bridge.³

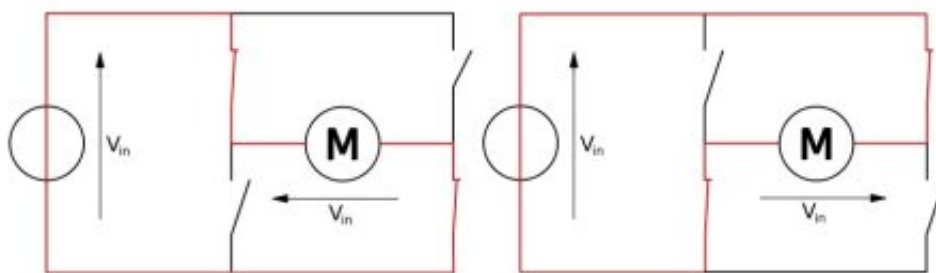


Figure 1: (a) 1A = '1', 2A = '0', (b) 1A = '0', 2A = '1'

If both inputs are low, both outputs will be low thus allowing the motor to 'freewheel' and slow down due to the gear train.

3.2 Digital Design

The program was developed for an Altera Cyclone II FPGA on the Altera DE2-70 Development Board. It was designed using Quartus, the Altera CPLD/FPGA Development Kit, so the project can be ported to any CPLD or FPGA that has a large enough capacity to fit the design. That said, the design was comprised of five different modules, four of which had to be created from scratch: a Pulse Width Modulation (PWM) generator, a motor logic module, a memory controller, and an 8-bit register. The fifth module is a clock divider used to divide the DE2-70's 50MHz on-board clock into decades starting at 1MHz and going down to 1Hz. *Rapid Prototyping of Digital Systems* by Hamblen and Furman includes a clock divider for Altera's UP2 (which has a 25MHz on-board clock) that can be easily modified to divide the DE2-70's clock.

3.2.1 PWM Generator

It is known (by magic hand waving) that the easiest way to digitally control the speed of a DC motor is through pulse width modulation. This controller was designed with a 20ms cycle, which

²<http://www.ti.com/lit/gpn/sn754410>

³<http://en.wikipedia.org/wiki/H-bridge>

means the pulse width can be as small as 0ms and as large as 20ms. However, the motors on the robot arm generally don't move until about 7ms or so. The module has three inputs: a clock, an enable pin, and a 'high time' setting.

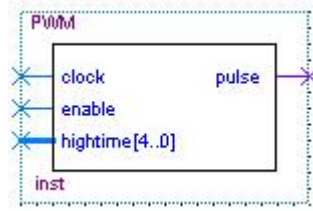


Figure 2: The PWM Module

To get a 20ms cycle, the clock needs to be a 1KHz clock. In the final design, the PWM enable pin is connected to a push-button ANDed with an enable bit stored in memory (See §3.2.3). Finally, the high time is a 5 bit number, limited internally to 20, set by the toggle switches on the DE2-70.

The PWM controller works by having a counter that counts up from zero to nineteen, and then resets itself. Twenty is subtracted from the high time ($hitime = 20 - hightime$), and then compared to the count. If the count is greater than the high time ($hitime$), then the output is set high, and vice versa.

3.2.2 Motor Logic

Since the motor driver's input combinations (as mentioned in §3.1) can be rather confusing to a user, it is easier to create a module with two inputs, braking and direction, and let it compute the proper inputs for the motor driver.

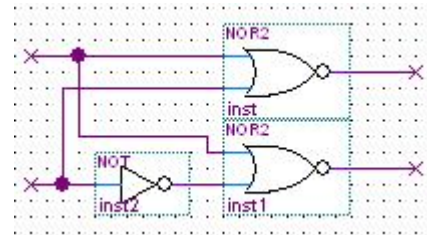
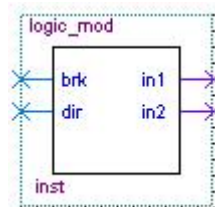


Figure 3: Logic Module Block Schematic File Figure 4: The innards of the module

This logic ensures that whenever 'BRK' is high, both 'IN1' and 'IN2' will be low so that the motor will freewheel. Table 1 shows the truth table which governs the logic module's operation.

Table 1: Truth table for logic module

BRK	DIR	IN1	IN2
0	0	1	0
0	1	0	1
1	0	0	0
1	1	0	0

3.2.3 Memory

In order to configure and run five motors at once with only 20 toggle switches, it was decided that some form of memory would be needed in order to store the configurations for each motor. This would require two things: registers to store the data in, and an address controller to tell the data which register to go into.

The registers in reality are just a bank of D flip-flops. However, in the design, each register appears as a single D flip-flop with an 8-bit bus for its 'D' input and 'Q' output. Figure 5 shows a single instance of one of the registers.

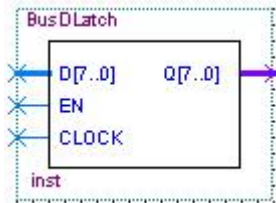


Figure 5: 8-bit D flip-flop register

As it can be seen in Figure 5, the register has an enable input and a clock input. Any clock speed will do as long as the memory controller has the same or slower clock speed. Otherwise, data coming from the controller may change before it is written to the register. The enable signal is controlled by the memory controller, shown in Figure 6.

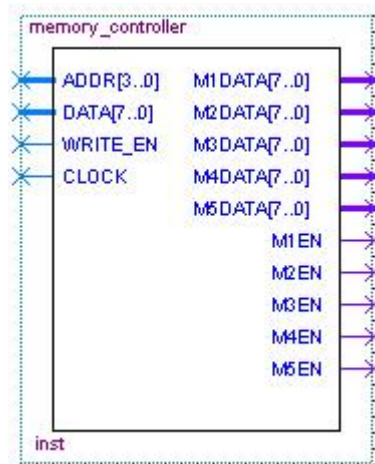


Figure 6: Memory controller

The memory controller has independent outputs and enables for each register. This is to avoid data being sent to the wrong registers, which happened in a previous design using the Altera Megafunction `altsyncram`. The 8-bit data string contains the following bits of information for each motor: motor enable, direction, brake, and speed. The MSB determines if the motor is enabled or not, bit 6 determines the direction, bit 5 determines if the brake is on or not, and bits 4 through 0 determine the 'high time' being sent to the PWM generator. The 4-bit address tells

the controller which register to direct the 8-bit data line to.⁴ The addresses start at 0x00 and go up to 0x04, which correspond to motors 1 through 5 respectively (Motor 1 being the base motor and Motor 5 being the gripper motor). For any addresses above 0x04, all of the data lines are set to 0x00000000 and the enable lines are all set low. When set high, the write enable pin will set one of the motor enables high (M1EN to M5EN) depending on the address being used, the appropriate data will be written to the appropriate register.

4 Remarks

At the time of writing, a closed loop control version of the robot arm control was being developed. However, it was not tested due to the fact that none of the sensors had been mounted on the robot arm. This is suggested for future work, and should be relatively easy to finish once the sensors are mounted on the arm.

5 Attachments

1. Schematic for interface board
2. Block diagram for final design

⁴The address is 4 bits instead of 3 for debugging purposes. The seven segment decoder used in debugging only accepts a 4-bit input.