

Chapter 3

Combinational Logic (Custom Remote Control)

3.1 Section Overview

One of the most challenging concepts in robotics and control designs is the system used in remote operated vehicles (ROVs). Such systems need to be able to work in remote locations, (sometimes under extreme conditions) and perform for long periods of time. In order to implement this capability in this system, the top priority is to control the device effectively, so that it does not waste valuable energy and does not damage itself.

There are two main types of controls used in modern robotics: local control and remote control. Local control refers to the usage of artificial intelligence, so that the robot itself can think without external control. This intelligence is usually very basic, and can be compared to an insect's brain (with only simple functions such as obstacle avoidance). Remote control allows the ROV to be directed from a different location, essentially moving the brains away from the body. Section 3 focuses on designing a custom remote controller for the TekBot.

3.2 Objectives

In this section, the following items will be covered:

1. Defining the problem — what is the controller really doing?
2. Defining the inputs to the combinational logic
3. Defining the outputs from the combinational logic

3.3 Materials

1. Xilinx ISE 12 software (Currently installed on the lab computers)
2. Digital Logic Board (d.logic.2 board)
3. Small breadboard, pushbuttons, and resistors OR another Digital Logic Board (d.logic.2 board)
4. A TekBot with a working motor controller and batteries

3.4 Procedure

There are 5 steps to digital logic design:

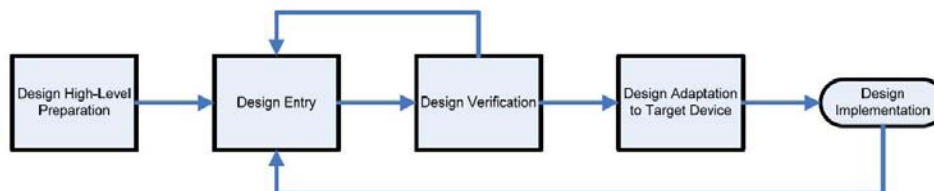


Figure 3.1: Use this process for designing the custom remote control.

1. *High Level Preparation*: Figure 3.2 shows all of the different actions that the remote control should accept.

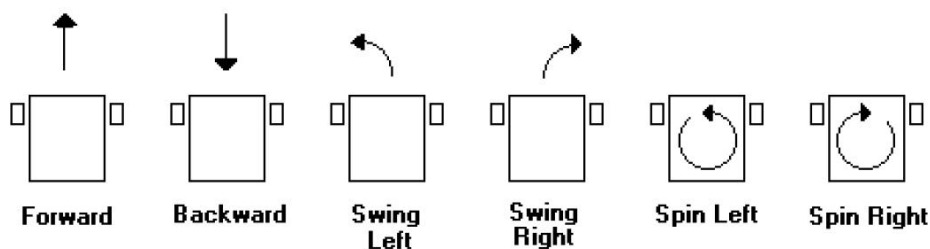


Figure 3.2: The remote control should use all of these actions

Make a block diagram

Add the power source used for each of the three blocks.

Label the Port and Port pin on the block diagram if the secondary digital logic board is used for the button inputs. Label the location on the breadboard if the button inputs are coming from a breadboard circuit.

Label which inputs on the motor controller are connected to Le, Ld, Re, and Rd. Le stands for left motor enable, Ld stands for left motor direction, Re stands for right motor enable, and Rd stands for right motor direction.

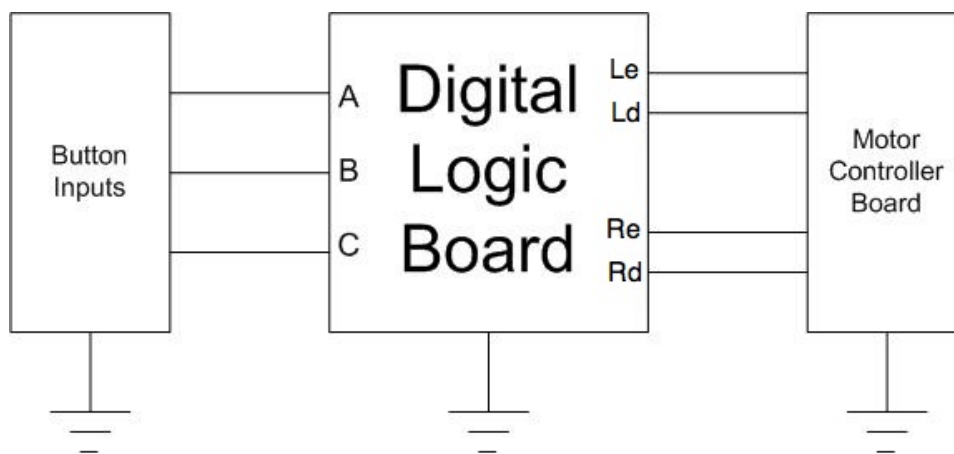


Figure 3.3: This is the block diagram for the remote control

Make a functional truth table

Add the binary value of the inputs for each row. Remember if the buttons are active high or active low.

Add the binary value for the outputs for each row. Use page 53 in Section 3 of ECE 112 for a reference

on how the motor controller operates.

TekBotActions	Inputs, ABC	Left Motor Direction	Right Motor Direction	Outputs: Le,Ld,Re,Rd
Forward				
Backward				
SwingLeft				
SwingRight		Forward	Stopped	
SpinLeft				
SpingRight				
DoNothing				
DoNothing				

Minimize the logic

Write out the canonical form for $Le(A,B,C)$, $Ld(A,B,C)$, $Re(A,B,C)$, and $Rd(A,B,C)$. Use Σ or Π .

Use four K-maps to make minimized logic for Le , Ld , Re , and Rd . Write out the minimized Boolean Equations for each output.

2. *Design Entry*: Enter the design using the same process as in Section 2.
3. *Design Verification*: See the instructions for this step below.
4. *Design Adaptation to Target Device*: Follow the same process for using PACE and iMPACT as in Section 2.
5. *Design Implementation*: Program the Xilinx CPLD with the Universal Programmer software.

3.5 Design Verification

In this section, the design is tested by entering in all possible input combinations and comparing the simulated output to the desired output. If they match, we move on to the next step in the design process. If there are discrepancies, debug and try again until the desired output is achieved.

3.5.1 Creating the Testbench

1. Click on Project → New Source
2. Select "Verilog Test Fixture" and name the file "Section3Test"
3. Select your design schematic as the source to associate it with and click Next, then Finish.
4. Xilinx creates a Verilog Testbench file that will be used to test the circuit.

3.5.2 Changing the Verilog Template

1. Remove the line "ifdef auto_init".
2. Change the line "endif" to only say "end".
3. Change the text between "initial begin" and "end" to match the image below.



The #10 at the beginning of each line denotes how long the simulation is to use that combination of inputs before moving on to the next line. It is measured in nanoseconds.

```

initial begin
    A = 0; B = 0; C = 0;
#10 A = 0; B = 0; C = 1;
#10 A = 0; B = 1; C = 0;
#10 A = 0; B = 1; C = 1;
#10 A = 1; B = 0; C = 0;
#10 A = 1; B = 0; C = 1;
#10 A = 1; B = 1; C = 0;
#10 A = 1; B = 1; C = 1;
end

```

Figure 3.4: The lines of code between initial begin and end should match this.

3.5.3 Running the Simulation

1. In the Design pane in section A, change the selection on the "View:" radio button to "Simulation".
2. To select the testbench as the process, select Test1.v in the heirarchy pane (you may have to resize this pane to see the file names).
3. Expand the "ISim Simulator" node in the Process pane.
4. Right click on Simulate Behavioral Model in the Processes pane in section B and select Process Properties.
5. Change the simulation time to 100ns, then press OK.



Depending on how many lines of input combinations you have, the simulation time may need to be increased. Add up the #10s to see how much time is needed.

6. Double click on Simulate Behavioral Model.



If the simulated outputs are not correct and you change your testbench or design, ISim must be closed before you can simulate again.



TA Signature: _____
(TekBot functions properly & work is displayed)

3.6 Study Questions

1. Make a modified block diagram using Figure 3.3, that includes a block of logic for reverse lights. Include the required logic to turn on the backup lights when the TekBot is moving backwards. Use A, B, and C as inputs for this combinational logic.
2. Make a modified block diagram using Figure 3.3, that includes a block of logic for reverse lights. Include the required logic to turn on the backup lights when the TekBot is moving backwards. Use Le, Ld, Re, and Rd as inputs for this combinational logic. Include the required logic to turn on the reverse lights.
3. Recreate a block diagram similar to Figure 3.3 and a functional truth table like the one used in this lab to use the digital logic board to control a L293 motor controller chip. The datasheet for this chip is on the lab website. Note that this chip uses tri-state buffers which are described in page 80 of the ECE 271 textbook. Do not use

the block diagram or the functional truth table from the datasheet; those are not examples of what we are looking for.

(Hint: This is the same motor controller IC that was used in ECE 111.)

3.7 Challenge - Extra Credit

The Section 3 project currently uses 4 or 5 wires (A, B, C, ground, and maybe Power). Redesign this project to only use 2 wires (Analog signal and ground). Use a R2R ladder as an analog encoder and use a discrete analog digital converter as the analog decoder. Draw the exact schematic for the analog encoder, analog decoder, and how everything connects together.