

CMPE-240 Laboratory Exercise 04

Logic

By submitting this report, I attest that its contents are wholly my individual writing about this exercise and that they reflect the submitted code. I further acknowledge that permitted collaboration for this exercise consists only of discussions of concepts with course staff and fellow students; however, other than code provided by the instructor for this exercise, all code was developed by me.

Daniel Santoro

Performed: 10/19/16

Submitted : 10/31/16

Lecture Section: 01

Professor: Alessandro Sarra

TA: Kevin Millar, Adrian Cruzat, Humza Syed

Abstract

The goal of this lab is to combine the experience of the previous labs (timer, UART, and GPIO) to produce a program which analyzes boolean sum of product (SOP) equations and produce a truth table. The result of the lab will utilize UART to present an interactive menu for a user to select a predefined couple of Boolean SOP equations. Then the Raspberry Pi will compute and print the corresponding truth table to UART; additionally the Raspberry Pi will utilize LEDs to represent the output of the truth table by representing 0s with off and 1s with on for one second and using a two second delay between each cycle for each value of the truth table.

Design Methodology

The UART connection to the GPIO pins is set up (as shown in figure 1.1).



Figure 1.1

In the config.txt file of the Raspberry Pi image, UART needed to be enabled and Bluetooth needed to be disabled (as shown in figure 1.2).

```
# Enable UART
enable_uart=1

# Disable bluetooth (UART and bluetooth share the same resources,
# so both can't be active at the same time)
dtoverlay=pi3-disable-bt
```

Figure 1.2

Putty was used to connect over serial port via UART to the Pi (as shown in figure 1.3). To find the COM number to connect to, device manager was used (on Windows 10) and the COM number was listed under Ports.

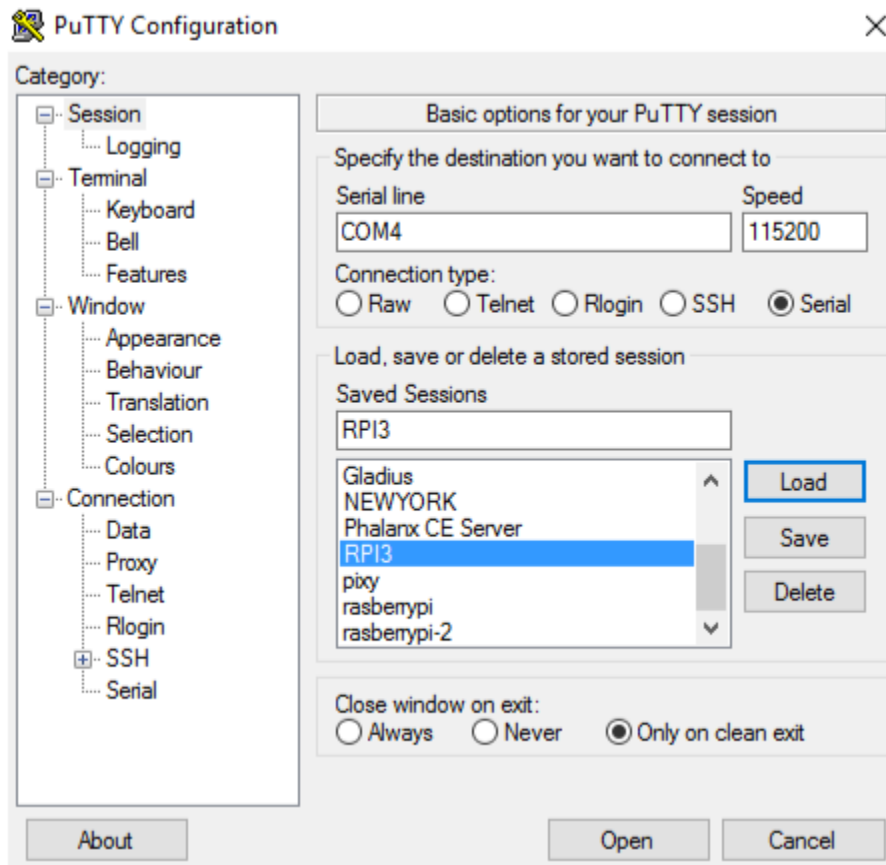


Figure 1.3

In the main code function, a while loop looking for user input via UART was created. If the input is 1 or 2, the code computes one of two predefined set of two Boolean equations through the truth table; an input of 3 will toggle whether to use LEDs in the output function.

The code utilizes a compute function which takes parameters such as headers, titles, and the Boolean equations strings to compute (as shown in figure 1.4). The algorithm accepts SOP equations using either abcd, or wxyz sets.

The code parses each character in the string and then applies the current value of the truth table for that index to the corresponding characters. The final return value starts at 0. The code uses a “temporary” value (which starts with 1) to hold the results of “ands”. Then when it reaches an “or”, applies the temporary value “or” with the return value. If it sees a single quote, it “xors” the value of the index (which results in flipping its value).

```
logic("R1", "b'd'+bcd'",  
      "R2", "b'd'+a'bc",  
      "Don't Care");
```

Figure 1.4

For the LED portion of the lab, a circuit to operate the LED was constructed using a breadboard similar to the one shown in figure 1.4 except it used two LEDs & resisters and four wires. The 16th and 20th pins were used as an output controlling pins for each LED. Do not use the same rows on the breadboard for the separate LED connections.

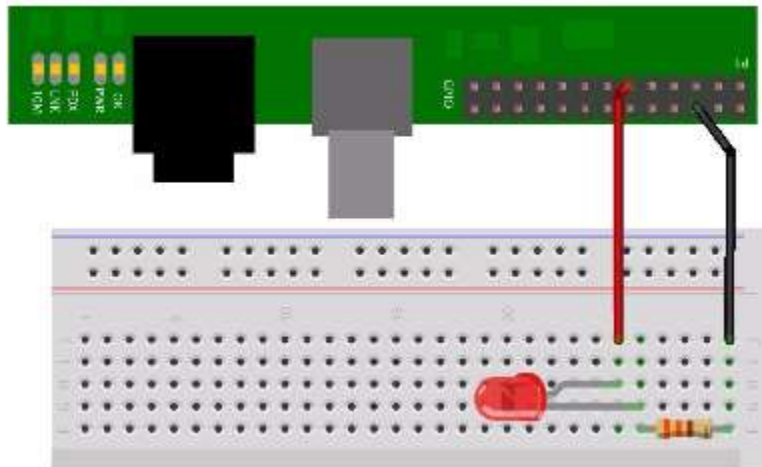


Figure 1.5

A sleep function was written using the Raspberry Pi's system timer channel; the delay in nano seconds would be inserted into the C0 timer register and the code would spin until the CS register was written to (which signified that the timer expired). Two additional functions which scaled up the timer's function input into milliseconds and seconds respectively were also written.

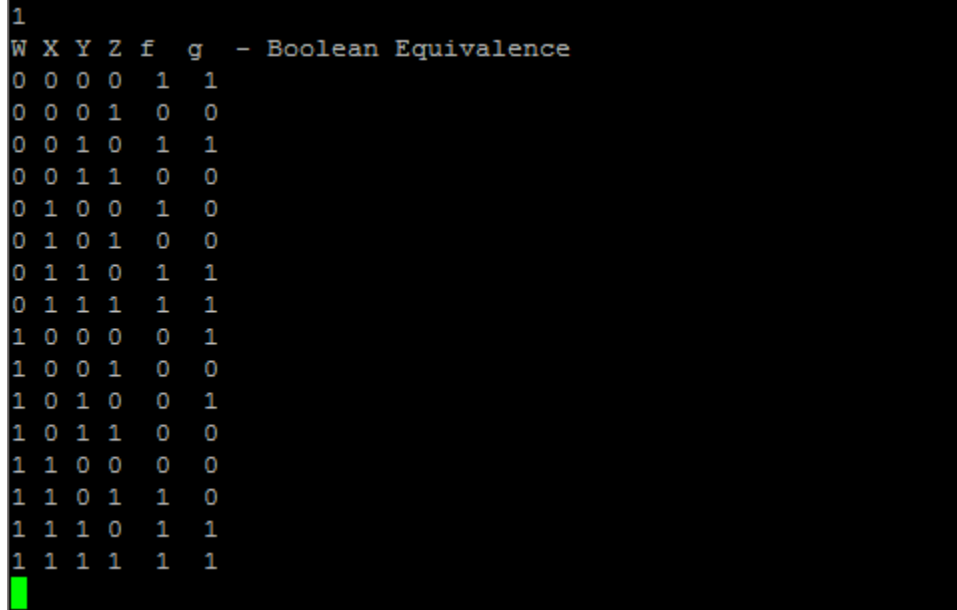
In the main, a blink_once function was written which would take two “states of pins” (whether 0 or 1) set the registry to turn on a specified port (in this instance pin 16 or 20) if the value was

1, wait one second, and they turn off the port. When printing the truth table, the print function will call `blink_once` with the current states of the results for that row in the truth table.

Results and Analysis

When printing the truth table, if the user had selected to use LED as output in the UART menu, if the user toggled the blink functionality, the Raspberry Pi would blink the corresponding LEDs. The below figures (2.1 and 2.2) show the outputs of the two sets of Boolean equivalence equations.

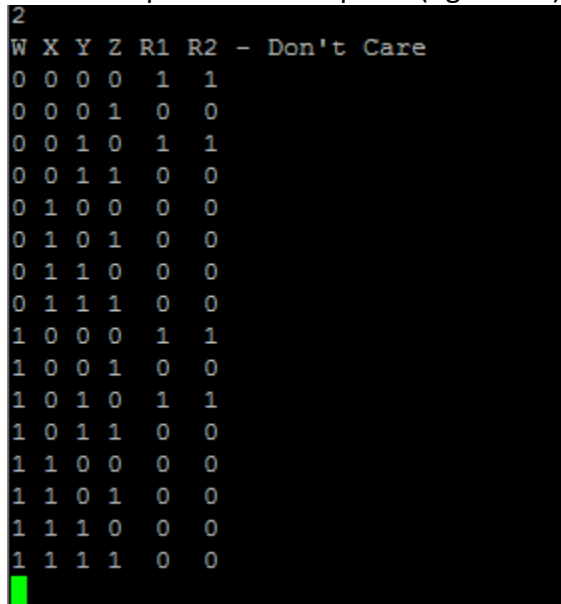
Boolean Equivalence 1 Capture (figure 2.1)

A terminal window showing a truth table for Boolean Equivalence 1. The title is "1" and the header is "W X Y Z f g - Boolean Equivalence". The table contains 16 rows of binary data. A green cursor is visible at the end of the last row.

W	X	Y	Z	f	g	- Boolean Equivalence
0	0	0	0	1	1	
0	0	0	1	0	0	
0	0	1	0	1	1	
0	0	1	1	0	0	
0	1	0	0	1	0	
0	1	0	1	0	0	
0	1	1	0	1	1	
0	1	1	1	1	1	
1	0	0	0	0	1	
1	0	0	1	0	0	
1	0	1	0	0	1	
1	0	1	1	0	0	
1	1	0	0	0	0	
1	1	0	1	1	0	
1	1	1	0	1	1	
1	1	1	1	1	1	

Figure 2.1

Boolean Equivalence 2 Capture (figure 2.2)

A terminal window showing a truth table for Boolean Equivalence 2. The title is "2" and the header is "W X Y Z R1 R2 - Don't Care". The table contains 16 rows of binary data. A green cursor is visible at the end of the last row.

W	X	Y	Z	R1	R2	- Don't Care
0	0	0	0	1	1	
0	0	0	1	0	0	
0	0	1	0	1	1	
0	0	1	1	0	0	
0	1	0	0	0	0	
0	1	0	1	0	0	
0	1	1	0	0	0	
0	1	1	1	0	0	
1	0	0	0	1	1	
1	0	0	1	0	0	
1	0	1	0	1	1	
1	0	1	1	0	0	
1	1	0	0	0	0	
1	1	0	1	0	0	
1	1	1	0	0	0	
1	1	1	1	0	0	

Figure 2.2

Conclusions

The lab didn't call for parsing of strings for the equations; this implementation of the lab was due to a misunderstanding of the UART menu portion (initially read as users may supply input). It caused an increase in the time to code and debug of the lab, but allowed for an increase in understanding of Boolean logic implementation in C.

The lab was successful: the UART menu prompts a user for input, accepts either 1 or 2 for the predefined couple of sum of equations, computes the truth table and outputs the results. Additionally, LEDs could be used as an alternate form of output.

A major problem encountered in the lab was the truth table not producing correct results (either all 0s or all 1s). Strenuous debugging using print statements was used to narrow down the problems: incorrect calculation of a char array's size caused the program to only compute the first four values of an inputted string; additionally, the order of checking the not operator (initially after checking for the "or" operator) caused the algorithm to miscalculate the result.

Questions

In the case of the first boolean equivalence, the results of the un-simplified equation did not match the results of the k-map simplified equation.