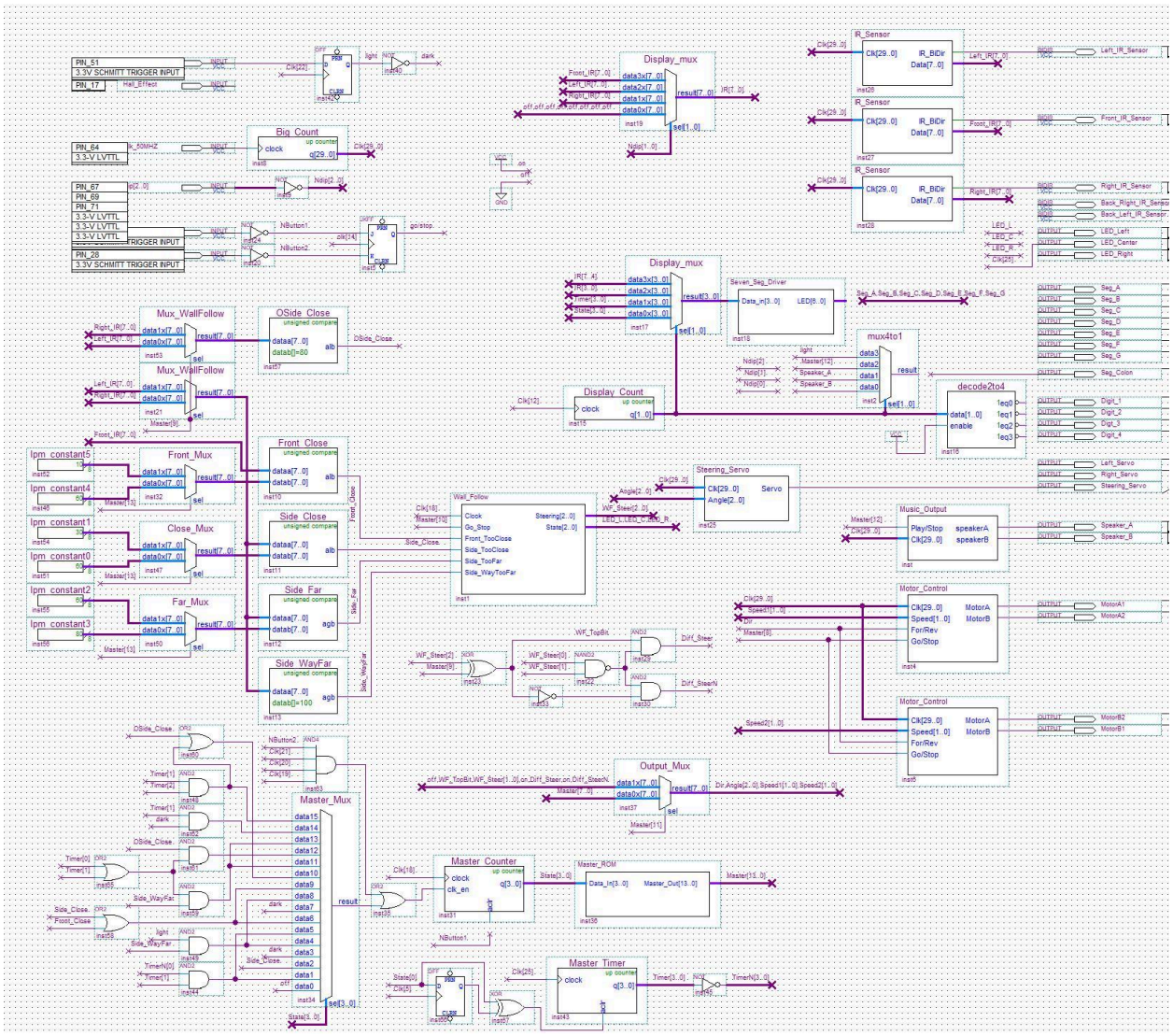
Kenneth Lee III

Advanced Digital Logic Design

Robot Project Report

April 27, 2024

Top Level Design File

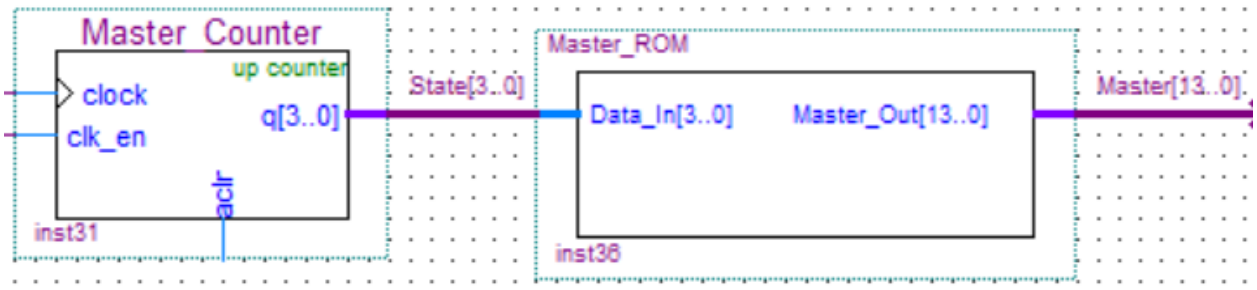


Logic Cell Usage Details

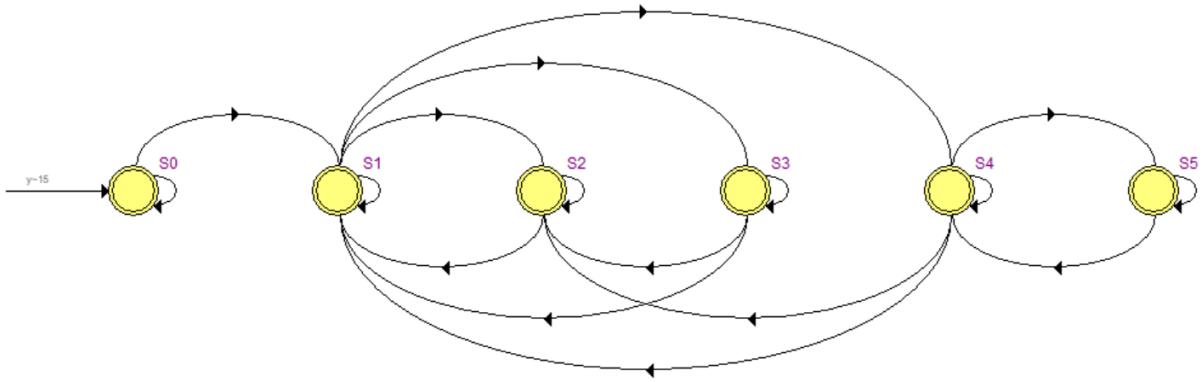
Entity	Logic Cells
MAX II: EPM240T100C5	
Main	185 (10)
Music_Output:inst	34 (4)
Wall_Follow:inst1	13 (13)
mux4to1:inst2	0
Motor_Control:inst4	7 (2)
Motor_Control:inst6	5 (2)
Big_Count:inst8	26 (0)
Front_Close:inst10	0 (0)
Side_Close:inst11	2 (0)
Side_Far:inst12	3 (0)
Side_WayFar:inst13	2 (0)
Display_Count:inst15	0
decode2to4:inst16	0
Display_mux:inst17	0
Seven_Seg_Driver:ins...	0
Display_mux:inst19	0
Mux_WallFollow:inst21	0 (0)
Steering_Servo:inst25	13 (1)

Entity	Logic Cells
IR_Sensor:inst26	12 (0)
IR_Sensor:inst27	11 (0)
IR_Sensor:inst28	13 (1)
Master_Counter:inst31	4 (0)
Front_Mux:inst32	0
Master_Mux:inst34	12 (0)
Master_ROM:inst36	11 (11)
Output_Mux:inst37	3 (0)
Master_Timer:inst43	3 (0)
lpm_constant4:inst46	0
Close_Mux:inst47	0
Far_Mux:inst50	0
lpm_constant0:inst51	0
lpm_constant5:inst52	0
Mux_WallFollow:inst53	0
lpm_constant1:inst54	0
lpm_constant2:inst55	0
lpm_constant3:inst56	0
OSide_Close:inst57	1 (0)

Master Finite State Machine

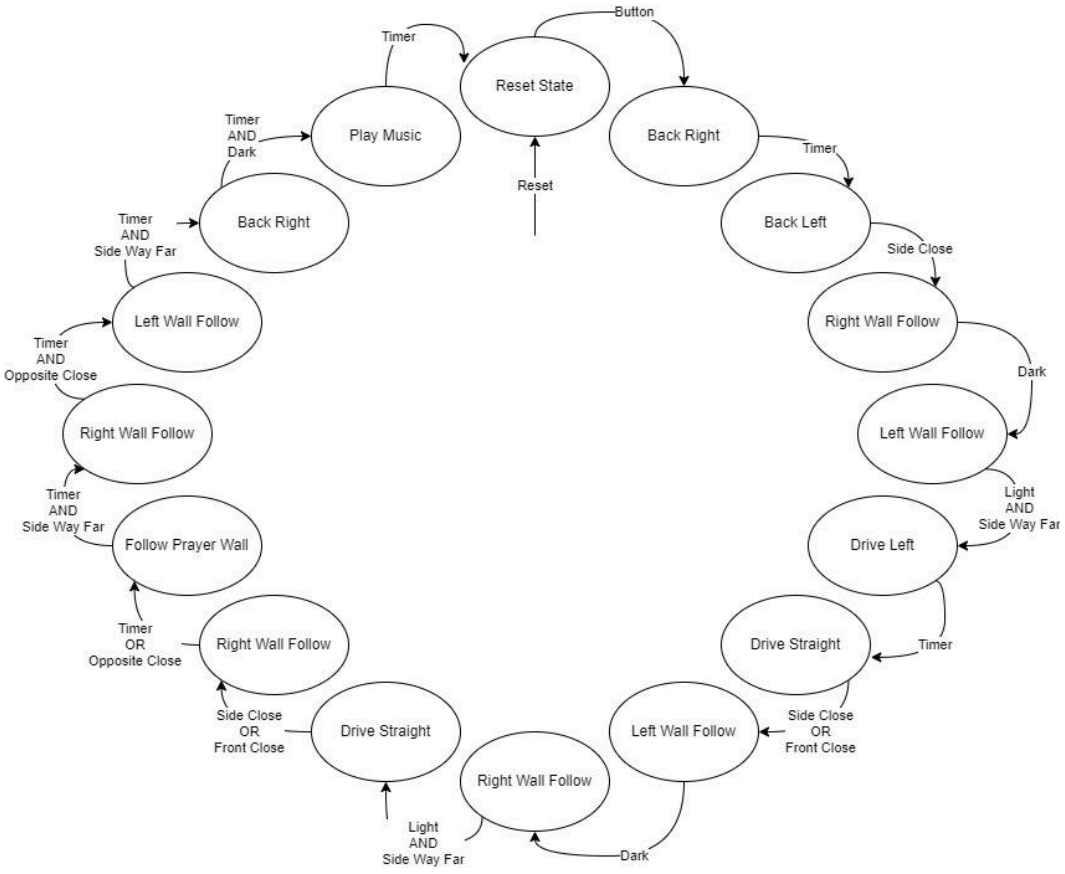


Wall Following Finite State Machine

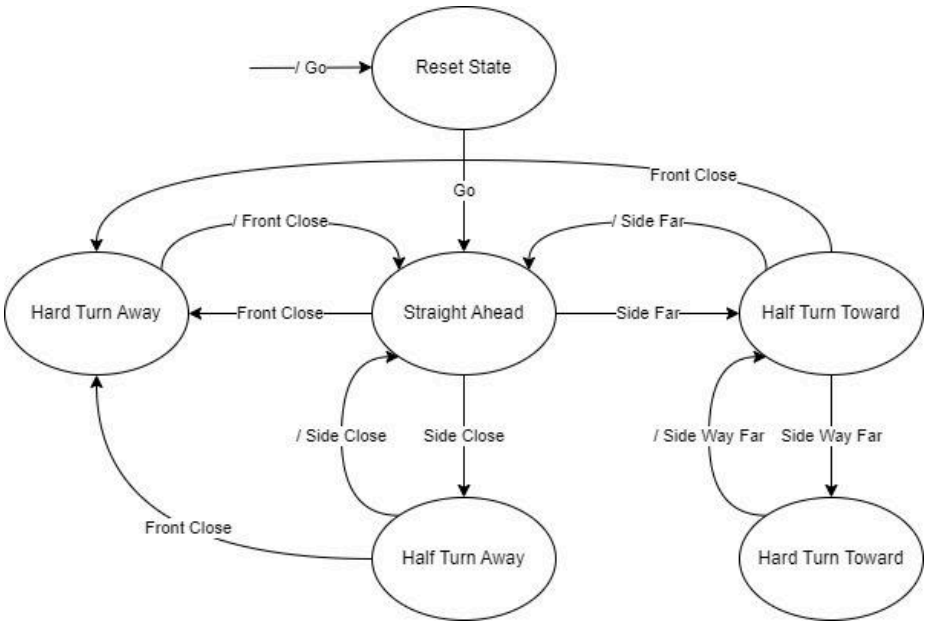


	Source State	Destination State	Condition
1	S0	S0	(!Go_Stop)
2	S0	S1	(Go_Stop)
3	S1	S1	(!Side_TooFar).(!Front_TooClose).(!Side_TooClose)
4	S1	S2	(Front_TooClose)
5	S1	S3	(Side_TooClose).(!Front_TooClose)
6	S1	S4	(Side_TooFar).(!Front_TooClose).(!Side_TooClose)
7	S2	S1	(!Front_TooClose)
8	S2	S2	(Front_TooClose)
9	S3	S1	(!Front_TooClose).(!Side_TooClose)
10	S3	S2	(Front_TooClose)
11	S3	S3	(Side_TooClose).(!Front_TooClose)
12	S4	S1	(!Side_TooFar).(!Side_WayTooFar).(!Front_TooClose)
13	S4	S2	(Front_TooClose)
14	S4	S4	(!Side_WayTooFar).(Side_TooFar).(!Front_TooClose)
15	S4	S5	(Side_WayTooFar).(!Front_TooClose)
16	S5	S4	(!Side_WayTooFar)
17	S5	S5	(Side_WayTooFar)

Master Flow Chart



Wall Following Flow Chart



Original Design Details

For the majority of the main design file, I followed the instructions and tips that we learned in the labs, but there are several areas in which I used my own ideas or designs to make the robot work.

First, when I implemented the wall following state machine, I did it the conventional way except I did not include any functionality inside the state machine for back wheel differential steering. I did need differential steering for the wall following to work well so I did it in the following way: take the top bit of the steering output and XOR it with the side selector bit; then AND it with the bottom two steering output bits NANDed. The result of this operation produces a bit which can be used as the top bit of the Speed input of the motor control blocks. This concept was a little difficult to explain to others how it worked but ultimately it efficiently implemented differential steering.

Second, I took the recommendation to implement the master finite state machine using a counter, a multiplexer, and a ROM. The counter increments either when the multiplexer evaluates to true or when the short button is pressed ANDed with Clk[19], Clk[20], and Clk[21] so that the state doesn't increment more than once. The counter is a four-bit counter which allowed me to have sixteen states, which turned out to be just enough. The output of the counter controls the multiplexer and the ROM. The ROM has fourteen output bits, half of which are selector bits for various controls, while the other half are drive speed and steering bits for overriding the wall following bits when necessary. There is another counter which serves as a timer for when a timed event is needed. This timer is reset at the change of each state and always counts up whether it is needed as a condition or not.

Finally, I implemented a second type of wall following by setting all of the comparators to be changed based on a multiplexer, allowing for me to have a version of wall following that follows at a decent distance and a second version where I can hug the wall much more tightly. This was important for wall following inside the T-tunnel and right outside the church.

Design Challenges

The biggest and most frustrating design challenge I faced was when my robot would be completely thrown off course by “ghost walls”. These were triggered by the IR sensor even when there was no wall close at all. Dr. Kohl explained what was happening to the IR sensor input reading that caused this problem, and correctly told us that ANDing Clk[15], Clk[16], Clk[17], and Clk[18] as a trigger for the IR flip-flop would solve this problem.

The second big issue I faced was that my original design had the output of the master multiplexer clocking the master counter instead of setting the clock enable on the master counter. Making this change fixed many problems I was having with the robot jumping through states, but also caused other small problems which I had to debug individually.

The third largest issue I encountered was that for a long time I didn’t know why my wall following wasn’t working when it was supposed to be controlled by the master ROM. After significant debugging, I realized that my wall following block had a low-active reset instead of a high-active reset.

Project Conclusions

Overall, I found the robot project to be extremely helpful in that I learned a lot and enjoyed it immensely. I had experience with 3D printing and soldering before, but through this project I learned laser cutting and gained more experience with soldering than before. Even more so, programming my robot to complete the obstacle course gave me much more experience with how to work with VHDL and Quartus, including how to debug and diagnose mine and others’ problems. I enjoyed the project a ton.