Controlling a Robotic Arm with a DE1-SoC Board and Quartus Prime Schematic

Aidan Kaneshiro

EECE2160

Northeastern University

Boston, United States

kaneshiro.ai@northeastern.edu

***Abstract*— This paper introduces Pulse Width Modulation and controlling Servo Motors using a DE1-SoC and Quartus Prime Schematic. The design for controlling a Servo Motor by generating Pulse Width Modulation was extended to provide additional functionality to controlling five Servo Motors within a Robotic Arm – including controlling and rotating each motor through 180 degrees using switches and pushbuttons on the DE1-SoC Board, saving set positions, and automatically moving between positions. This program was completed with idea of controlling a robotic arm to repeatedly move bottles between Position 1 and Position 2 to meet specific timing requirements.**

I. Introduction

Pulse-width Modulation is a technique that involves generating pulses of different widths to transfer an analog signal. The duty cycle of a pulse wave is the proportion of time the analog signal is "ON" or "OFF" over a set period time of a cycle. In this project, Pulse-width Modulation was used to control five servo motors on a robotic arm. The greater the duty cycle, or greater percentage the signal was "ON," the larger the angle the motor would rotate to was. Using Quartus Prime Schematic, a DE1-SoC Board, and a Robotic Arm consisting of five servo motors, a design was created to control the robotic arm, save two set positions, and cycle through the two set positions within a set period of time. The schematic was designed with the intention of controlling the arm using switches and push buttons, choosing a set position for position one and position two, and continuously move water bottles at position one to position two. In the final product, the robot must assume an initial upwards position with all sliders OFF. Using pushbuttons and/or switches, the robot arm motors must be controlled to move the arm to a desired position. This position must be saved with SW7. Pushbuttons and/or switches must be used to move the robot to another position and this position must be saved using SW8. Finally, SW9 is toggled and the robotic arm must pick up an empty water bottle and move it from position 1 to position 2 every 4 seconds. Once the bottle is at position 2, the bottle must be let go immediately, pause for four seconds, and move back to position 1 to move another bottle. Once SW9 is toggled OFF, the will stop moving. Using SW7 and SW8, new position can be set for position 1 and position 2, respectively.

II. Algorithm Implmentation

A. Pulse Width Modulation in Quartus Prime Schematic

A necessary component in controlling the motors on the Robotic Arm is generating Pulse Width Modulation. Controlling the arm using Quartus Prime provided an advantage since LPM blocks were used to easily generate pulse-width modulation signals. This was achieved using a counter and varying values for a comparator. Using Quartus Prime's IP Catalog, a counter block with a modulus of 1,000,000 was connected to the DE1-SoC's 50MHz Clock. This made the counter repeatedly count up to 1,000,000. Next, Quartus Prime's IP Catalog was used to create a comparator block that would output a positive signal when the counted value was less than a set value. The corresponding comparator value for any given angle can be found using the following formulas.

Duty cycle (in milliseconds) = (angle x 10) + 600 (1)

Comparator value = duty cycle (in seconds) * 50*10^6 (2)

The PWM Signal from the clock on the DE1-SoC board has a frequency of 50 Hz and a period of 20ms. Since the duty cycle of the servo motors can range from 3% to 12% for 0 degrees to 180 degrees, the duty cycle should be between 0.6 ms to 2.4ms. This also means that every 0.1 ms represents 10 degrees the arm will rotate. From this knowledge, Equation 1 was derived. Once the duty cycle is ascertained, the comparator value can be discovered. The duty cycle is multiplied by the frequency of the clock to uncover the corresponding value of the comparator.

To move the motors to 90 degrees, the comparator was created to output a "ON" signal when the counter was less than 75,000 and an "OFF" signal when the counter was greater than or equal to 75,000. By changing the value of the compared value, the percentage of time an "ON" and "OFF" signal is sent the output changes, changing the duty cycle of the pulse. This simple schematic was modified to achieve the desired functionality to control the position of the robotic arm using the DE1-SoC Board, save two set positions of the Robotic Arm, and oscillate between set position. The schematic to move each arm to a set position can be seen below in Figure 1.

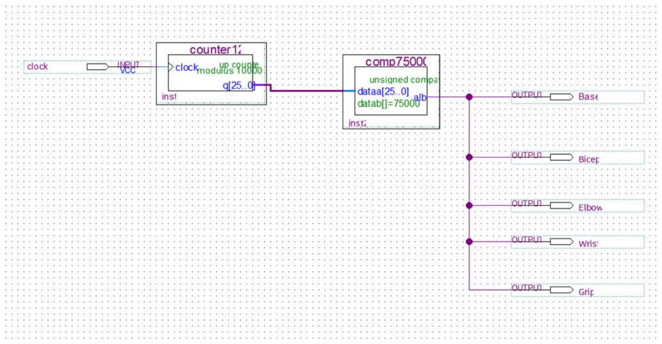


Figure 1: Basic Schematic for PWM Generation

B. *Controlling the Motors Position*

By building on the schematic for PWM Generation shown in Figure 1, the motors one the robotic arms were able to be controlled using switches and push buttons. In this design, a push button was used to change the angle of a motor and a switch was used to change the direction the angle changed. SW0 and KEY0, SW1 and KEY1, SW2 and KEY2, SW3 and KEY3 were used to control the motors for the base, bicep, elbow, and wrist, respectively. SW4 was used to control the Grip because it only needed to be an opened or closed position. SW5 was used to enable movement of the bicep, elbow, and wrist. If SW5 was "OFF," these three joints would be locked in the initial upwards position. A schematic for controlling the bicep, elbow, and wrist joints can be seen below in Figure 2.

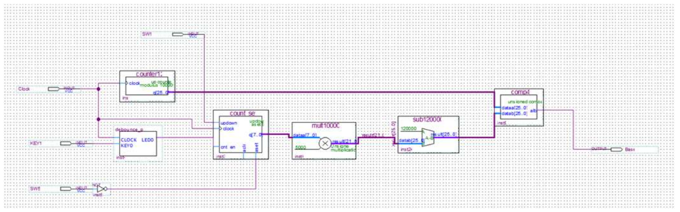


Figure 2: Controlling a Servo Motor with a Switch and Push Button

Figure 2, above, shows a schematic for controlling the bicep, elbow, and wrist joints. This schematic was built of the original design by implementing an updown counter and multiplier. The asynchronous set input of the counter is set to the opposite of SW5, which is initially OFF. Thus, the counter initially sets itself to a decimal value of 9 at the beginning of the program. The output of the counter is multiplied by 5,000 and subtracted from 120,000. Thus, when the program begins. The value comparator compares the output of the 1,000,000 modulus counter to 75,000. As demonstrated above in Figure 1, this initially sets the motors to 90 degrees, allowing the arm to assume an upright position.

When the SW5 is turned ON, push button is now able to change the angle of a motor. The switch input, in this case SW1, changes the direction the counter counts. When OFF, the counter counts down and when ON, the counter counts up. The counter is only able to count when a push button is pressed. To solve the problem of signal bouncing, a debounce was implemented between the push button input and the clock input. Using Equation 1 and Equation 2, the servo motors are can rotate through 0 degrees and 180 degrees when the comparator values are 30,000 and 120,000. By multiplying an 8-bit number by 5,000 and subtracting it from 120,000, the values of 30,000 to 120,000 can be reached in increments of 5,000. This means the motor can reach angles of 0 degrees to 180 degrees in 10 degree increments.

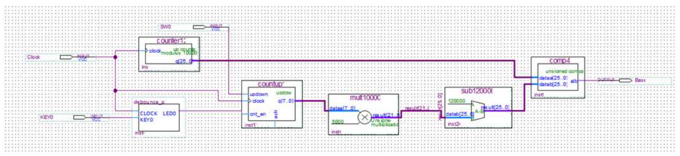


Figure 3: Schematic for Controlling Base

The Schematic for controlling the base of the arm, shown above in Figure 3, is nearly identical to the schematic for controlling the main joints of the arm (bicep, elbow, and wrist), seen in Figure 2. The difference between the two is that the counter for the Base can be freely controlled and does not load a set value using SW5. This limits unnecessary movement for the arm if SW5 is switched to reset the arm to the initial starting point.

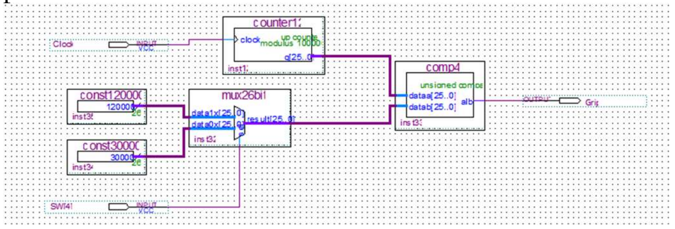


Figure 4: Schematic for Controlling Grip

The design for controlling the grip, shown above in Figure 4, is much different than the schematics for controlling the other motors. This is because the grip only needs two positions, open and close. When the duty cycle is 0.6ms and 2.4 ms, the grip will be fully closed and fully opened respectively. Thus, it will be close and open when the comparator value is 30,000 and 120,000 respectively. These comparator values were created as constant blocks and were inputs of a 2-to-1 multiplexor controlled by SW4. When SW4 was OFF, the grip will be fully closed, and when SW4 is ON, the grip will be fully open.

C. *Saving Set Positions*

The next task was to implement the ability to save set positions using SW7 and SW8 by storing PWM values. In other words, the individual comparator values for each motor would need to be saved. The objective of saving set positions is to move bottles from position one to position two repeatedly. Since this was achieved on a flat table, the two positions for the base were saved while the positions for the bicep, wrist, and grip were saved for only one of the positions. Additionally, the position of the grip was not saved at all because it will open and close on different timing than the rest of the arm. This will be explained in Part D.

Identify applicable funding agency here. If none, delete this text box.

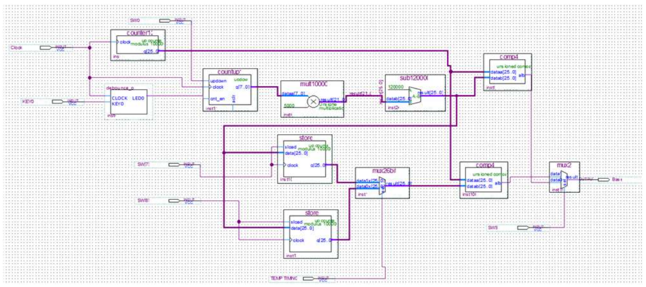


Figure 4: Saving Positions for the Base

The design for controlling the position for the base motor was extended to save the position of the base using SW8 and SW9. This was achieved by using a counter with a synchronous load. When SW7 or SW8 are ON, the counter saves the current values for the datab of the comparator block. The outputs of these counters are mapped to data0 and data1 of a 26bit mux controlled by a timing input (this is implemented later). The output of the 26bit mux is compared to the original 1,000,000 counter and the less-than output of this comparator is mapped to data1 of a 2-to-1 one-bit mux. Data0 of the one-bit mux was mapped to the original comparator controlled by the switches. This mux was controlled by SW9. This design can be seen above in Figure 4.

When SW7 and SW8 are saved, the robot does not initially move to SW7 or SW8 because of the one-bit mux. Because of the one-bit mux, the robot arm is controlled by the buttons and switches until SW9 is ON. When SW9 is ON, the either the value saved by SW7 or the value saved by SW8 will be output. This depends on a timing circuit that will be described later in part D.

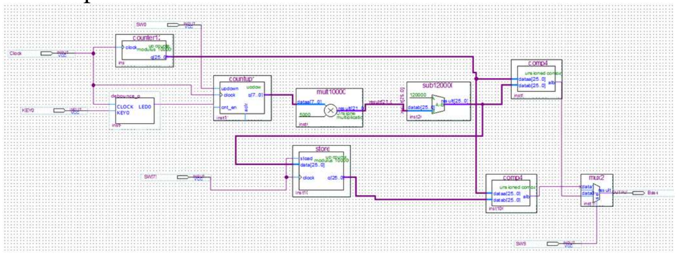


Figure 5: Saving Position of Bicep and Wrist

As previously described, only one position for the Bicep and Wrist needed to be saved. The schematic for this can be seen above in Figure 5. This Schematic is nearly identical to the schematic for saving two positions, except one of the store blocks was removed, as well as the 26-bit multiplexor.

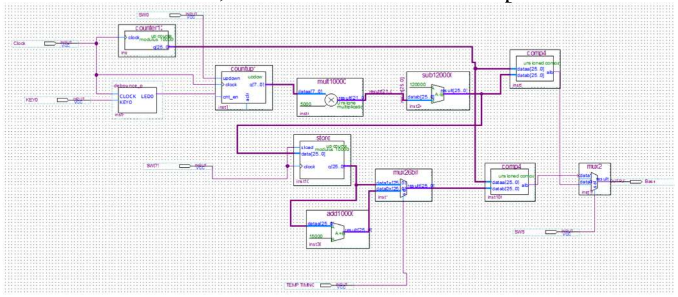


Figure 6: Saving Position for Elbow

Saving the Position for the Elbow was presented a unique. While only one position for the bicep needed to be saved, if the arm did not move up before moving back to position one from position two, the arm would knock the bottle over. This problem was solved by moving the arm up 30 degrees before moving back to position one. The schematic for saving the position of the bicep is nearly identical to the schematic for saving the position of the base. The difference is that instead of storing a value using SW8 and mapping it to an input on the 26-bit multiplexor, the value of 30 degrees larger than the position saved by SW7 was saved and mapped to the input of the 26-bit multiplexor. This was achieved by saving the Comparator value for SW7 and adding 15,000 to it. This design can be seen above in Figure 6.

D. Timing

The final part of designing the circuit was to implement proper timing to allow the circuit to oscillate between Position 1 and Position 2. The constraints of the design were

1. Moving from Position 1 to Position 2 must not take more than 10 seconds
2. The bottle must be released immediately after reaching position 2.
3. After the bottle has been released, the arm will wait exactly four seconds before restarting at Position 2.
4. The bottle must also not be knocked over at any point of this cycle.

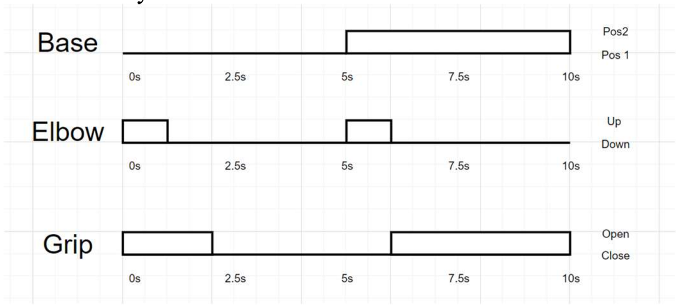


Figure 7: Timing Diagram

Figure 7, above, shows the timing diagram used to accomplish the design with the given constraints. Every 5 seconds, the base will rotate between Position 1 and Position 2. For the first second every five seconds, the Elbow will move up – this is to avoid the problem of knocking the bottle over and dragging the bottle on the table, resulting in the bottle "tripping" over a small object. Finally, the grip will be open for two seconds, close for four seconds, then open for four more seconds. This is so the grip will open and release the bottle when base is not moving at position 2. Notably, four seconds after the grip has opened the base will begin to rotate to back to the Position 1 and at the same time, the elbow will move up to avoid knocking the bottle over. The schematic to implement this functionality can be seen below in Figure 8.

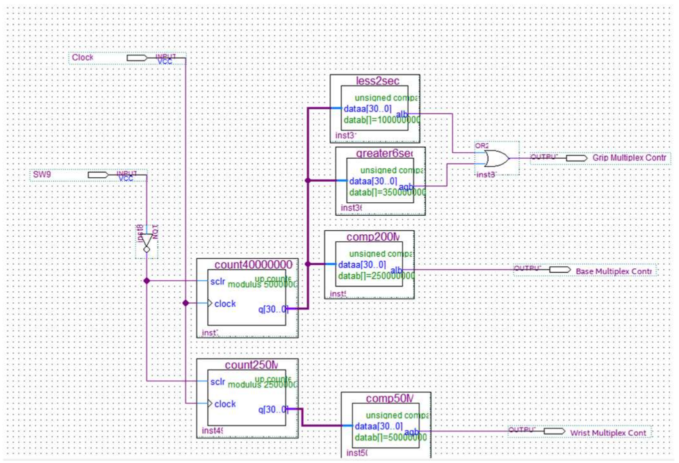


Figure 8: Timing Circuit for Robotic Arm

Figure 8, above, shows the timing of the Robotic Arm. The output of the comparator values are not mapped to an output on the board but to the control of the multiplexors controlling the positions saved by SW7 and SW8. The Grip and Base were connected to a counter that counts to 10 seconds, the Wrist was connected to a counter that counts to 5 seconds. However, these counters are continuously counting, even when the position of the arm is still being controlled by switches. This means the robotic arm might not start at the position for time 0s indicated by the timing circuit in Figure 9 when SW9 is toggled ON. To solve this problem, a synchronous clear output was mapped to the inverse of SW9, to reset the counter to 0 when SW9 is toggled.

III. Results

The final result of this project was a schematic that allowed the user to independently control each of the motors, save two positions of the robotic arm, and automatically move the arm to move water bottles from Position 1 to Position 2 while meeting specific timing requirements. This functionality was achieved using Quartus Prime Schematic, a DE1-SoC board, and a robotic arm consisting of five servo-motors.

The base, bicep, elbow, and wrist motors were controlled with a SW0 and KEY0, SW1 and KEY 1, SW2 and KEY2, and SW3 and KEY3. The keys changed the angle of the motor by 10 degrees, while the switch changed the direction the motor moved. The wrist grip was controlled by SW4 and had two set positions – open and closed. SW5 was used to enable and disable the control for the bicep, elbow, and wrist motors. Toggling SW5 off was set to automatically reset the motor to its base position, upwards. SW7 and SW8 were used to save Position 1 and Position 2. SW9 was used to enable the arm to move between Position 1 and Position 2.

The functionality of the Robotic Arm was demonstrated to the TA before the scheduled interview and performed as expected. Therefore, the schematic can be deemed as accurate as it has met all the technical requirements. Additionally, the functionality of the robot and a test of the arm can be seen and described in the attached presentation and video. The final schematic for this project can be seen in the Appendix.

A. *Limitations*

One limitation of the algorithm is that it was only capable that the bottle can only be moved along different positions on a flat surface perpendicular to the arm. Since this was assumed in the prompt for the project, the ability to move the bottle between positions of different heights was not taken into consideration when designing the schematic. However, this potential problem can be easily solved by saving the induvial position of each of the motors.

Another limitation of the current design is that the motor only rotates every 10 degrees. While this was sufficient for satisfying the given prompt, the design could be improved to rotate each of the base motors through 5 degrees instead of 10 degrees. To easily solve this problem, the value in the multiplicators would need to be changed from 5000 to 2500. This would allow the robotic arm to reach a greater amount of positions.

These two limitations of the algorithm do not affect the design's ability to complete the given prompt. However, factoring these two considerations when completing the design provides additional functionality and control of the robotic arm.

B. *Improvements*

In the initial design for this project, the Elbow was not programmed to move up and out of the way of the bottle. This meant bottle had to be a certain distance away from the robot to avoid the arm from knocking over the bottle. The bottle could only be grabbed when the grip was in range of the bottle when closed and not in range of the bottle when open. When this occurred, the elbow did not need to move up because the difference in the fully extended grip and the fully open grip was greater than that of a bottle cap. Therefore, the grip was able to grab the bottle when fully extended and avoid knocking the bottle over when fully open. This severely limited the program's ability to sufficiently complete the prompt. This was improved by implementing the ability to move the robotic arm up and out of the way before the robotic arm moves back to position one. Therefore, the arm is fully above the bottle and does not knock the bottle over when moving back to Position 1.

IV. Conclusion

In this project, Pulse Width Modulation Techniques were used to control the servo motors of a robotic arm. Using Quartus Prime Schematic, PWM signals were generated, allowing the positions for the robotic arm to be controlled using switches on the DE1-SoC. A schematic was designed to use switches and pushbuttons to control the angle of each of the motors. In addition, the ability to save two positions of the motors was implemented. Once SW9 was toggled, the arm oscillated between two positions to repeatedly move water bottles from Position 1 to Position 2.

Acknowledgments

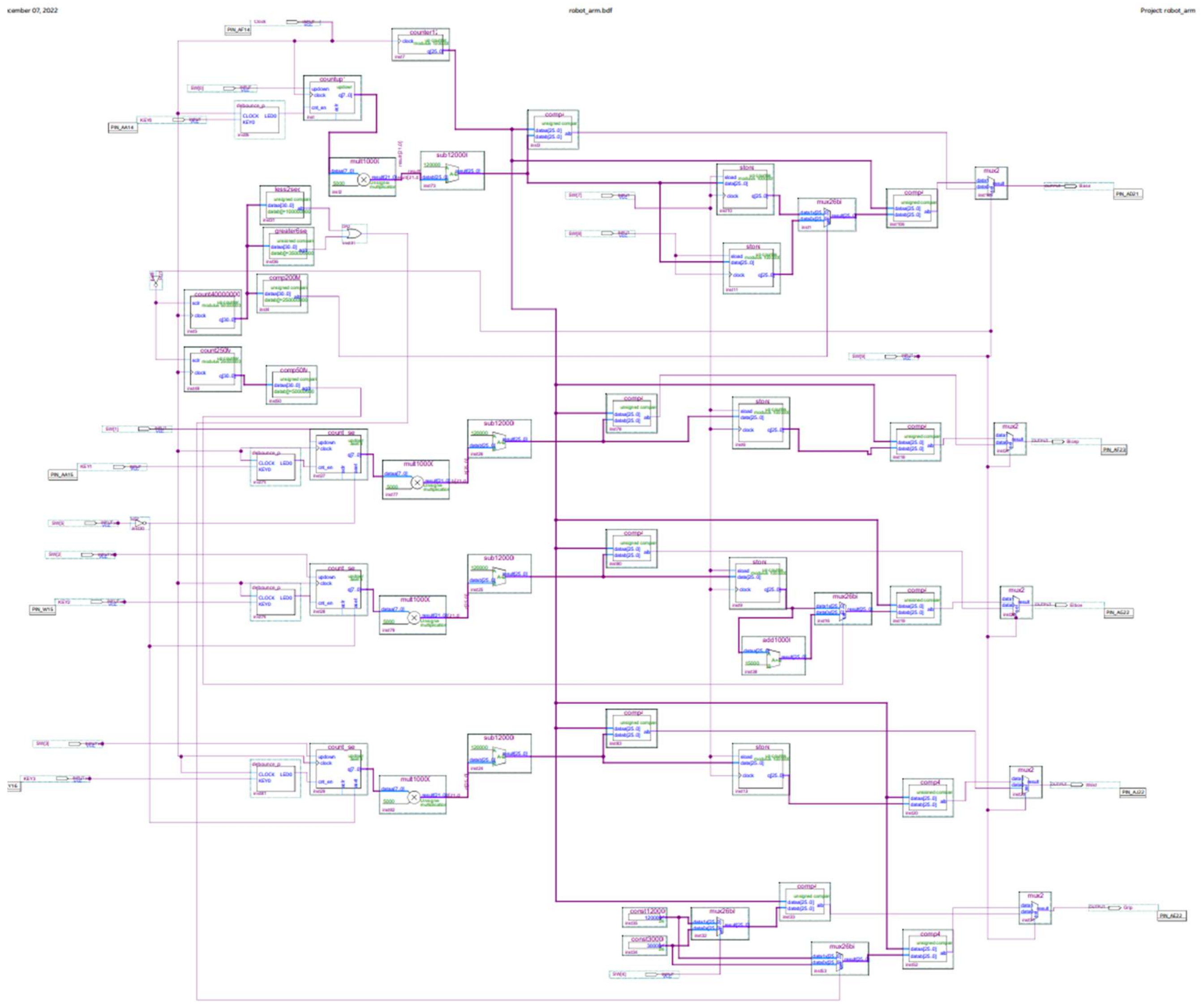
I would like to thank Professor Julius Marpaung for providing the hardware, the DE1SoC and Robotic Arm, for this project. I would also like to thank him for providing the necessary knowledge of the Quartus Prime Schematic, Pulse Width Modulation Techniques, and other knowledge that is necessary for completing the project. This would not be possible without him.

References

[1] DE1-SoC User Manual. Terasic, 28 Jan. 2019.

[2] "Manuscript Templates for Conference Proceedings," *@IEEEorg*, 2020. https://www.ieee.org/conferences/publishing/templates.html

Appendix


cember 07, 2022
robot_arm.bdf
Project: robot_arm