Temperature-Controlled Fan

Final Report

List of Authors (Gjozef Ivanaj, Samuel Urban, Luke Nuculaj, Andrew Czarnecki)
Electrical and Computer Engineering Department
School of Engineering and Computer Science
Oakland University, Rochester, MI
E-mails: gjozefivanaj@oakland.edu, surban@oakland.edu, lukenuculaj@oakland.edu, asczarnecki@oakland.edu

***Abstract*—The purpose of this project was to design and build a temperature-controlled fan. When the temperature sensor on the FPGA senses a temperature that is higher than the specified temperature desired, the fan will be activated. This will be done through the use of a DC motor that receives a pulse-width modulation signal from the FPGA. Simultaneously, the seven-segment display will read the temperature and display the value on a Nexys 4 DDR board.**

When designing and executing this project, it seemed that it would be rather simple and for the most part it was because it went to plan. However, there were some things that were unanticipated and therefore adjustments were made to accommodate. For instance, to get the temperature displayed on the board in a format that every person can understand became somewhat tricky. The 2's complement output from the temperature sensor was not ideal to display. Also, what if the temperature was negative? This too created new problems for the group. That is why, when beginning a project of this caliber it is important to use time management skills and plan for unexpected problems.

Overall, creating a detailed version of the design is sought after before trying to create it with VHDL code. Furthermore, it is crucial to research the hardware and software that is necessary to avoid confusion and time wasted. This will help create a project that is well-constructed and performs as expected.

I. Introduction

This report will cover the process that it took to reach a final implemented design of a temperature-controlled fan with a display of the temperature. Temperature controlled fans are used in homes to make daily life more comfortable and have many other real-world applications. This is especially the case with the exponential growth in assistive human technology. In recent times, almost all technology is being created so that it is hands free. This is why it was important for our design to activate without any assistance as well as be practical for daily life when being used on a larger scale.

The second motivation behind this design, other than the desire for human assistive technology, came from the idea that many electronics, furniture, artwork, etc., require a climate-controlled environment. The goal we wanted to achieve was to eliminate the worry of constantly checking that the temperature is at a certain spot. Therefore, the goal is to initiate the fan when the temperature exceeds a certain programmed interval. The design of this includes the use of several topics learned in class ECE 2700. To start, the VHDL code used to output bits from the temperature sensor (ADT7420) entails the use of several registers, parallel shift registers, finite state machines, pulse generators, and D-flip flops. This will generate a 16-bit output that is the temperature. The first 9 integer bits are significant bits and the other 7 are fractional bits that are less important for this design [2].

Pulse-width modulation through the use of the Nexys 4 DDR was a part of the code that was not taught in class. However, through research along with trial and error, this part of the project was also completed. From a general standpoint, PWM effectively chops a signal into discrete parts. The advantage of this being that power loss in swapping accessories is very little. The methodology of this report will further cover the PWM design in detail. The rest of this report will cover the steps taken to reach a final temperature-controlled fan with a temperature display. As well as the conclusions drawn from the from the design and creation process.

II. Methodology

The methodology of the project ultimately revolves around a goal that must be accomplished and how each group member can play a role in completing the task at hand. When looking at the overall design of the temperature-controlled fan, there are various sections of code and hardware that ultimately create the end product that is foreseen. The following divide up the project into certain tasks that were later combined to make the project whole.

A. WR Register

The WR register is specifically designed for the ADT7420 temperature sensor. It has several internal components such as two FSMs, parallel shift registers, a D flip-flop and a pulse generator. Figure 1 shows in detail each step that the data takes to reach the final output desired. The FSMs specify the signals. One of them adjusts for high pulse width in cycles of the clock frequency and the other is an asynchronous signal that also specifies default values.

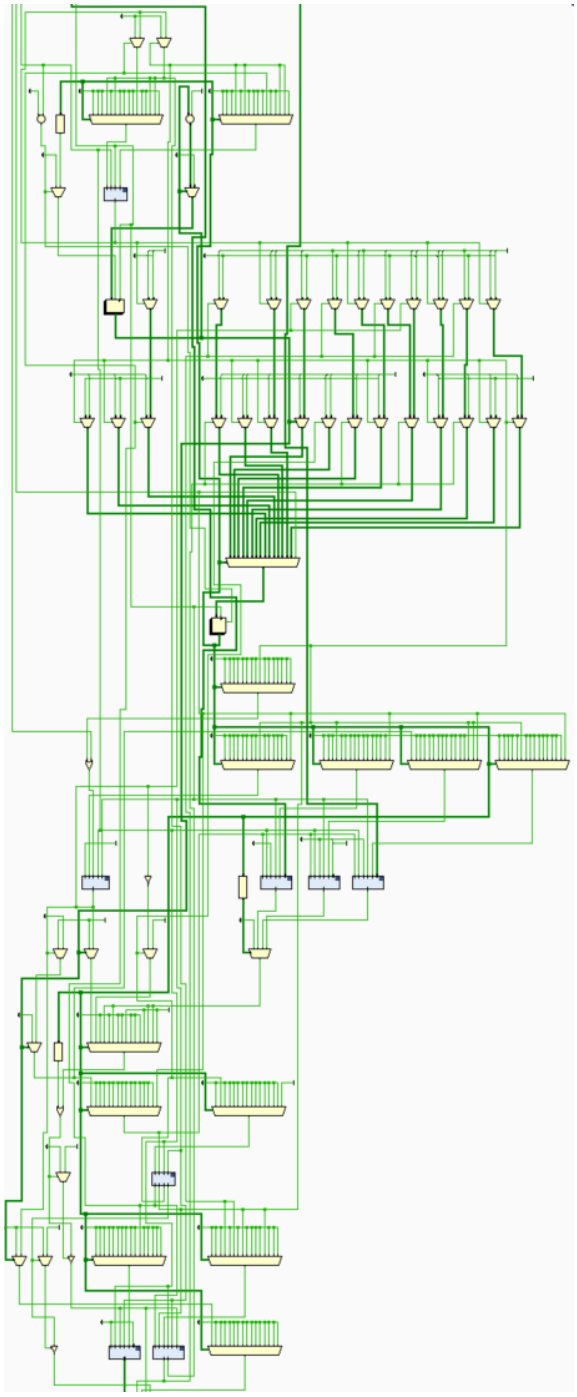


Figure 1. WR Register

The parallel shift register has the ability to alter directions. It can shift the left from LSB to MSB or to the right from MSB to LSB. In this case, when s_l is '0' there is a shift operation and when s_l is '1' data is loaded. The D flip-flop tracks the input and makes transitions that match the input data. The flip flop then stores the value on the data line. In this case there is an enable and when the enable is low, the data remains unchanged. The pulse generator deals with the cycles. It generates rectangular pulses instead of a constant signal. It is important to note that the output of WR register is a 16-bit temperature in two's compliment. 9 of these bits are integer bits and considered significant bits. The other 7 are fractional bits that are less significant.

B. Serializer

In general, serializers use the process of converting an object into a stream of bytes. This stores the object or transmits to save the state of the object. If the state is saved the serializer can recreate it when it is needed. Figure 2 shows the serializer and its components in depth.

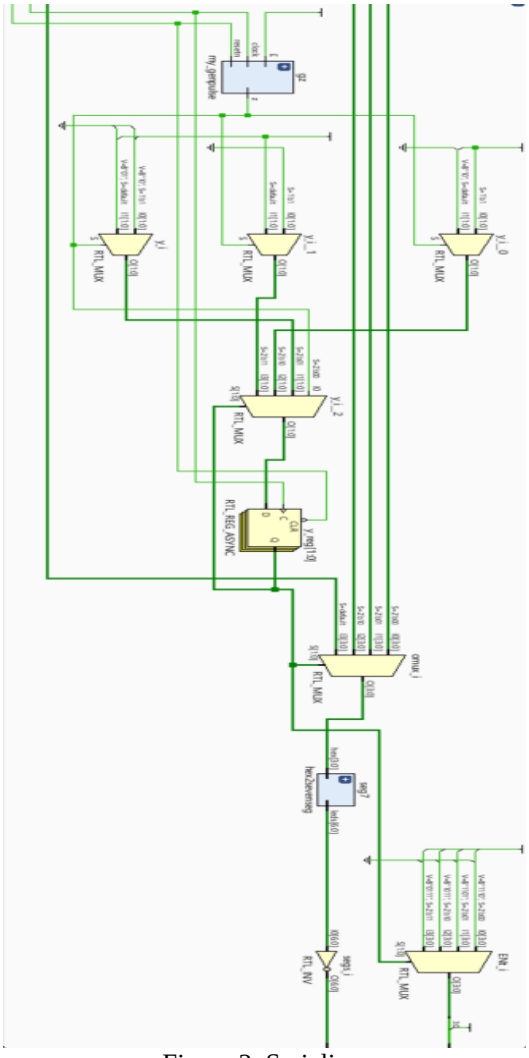


Figure 2. Serializer

For the project the serializer was used to get the temperature on to the seven segment displays. So that they read in degrees Celsius instead of 16 bits in two's compliment. The serializer is made up of a pulse generator and a hex to seven segment decoder. It takes the 9 significant bits from the WR register and outputs them as a temperature in degrees on the seven segment displays of the Nexys 4 DDR. A binary to BCD converter was also added to get the display to show in proper decimal form. The converter was

placed so that the output from the temperature sensor is converted and then connected to the input of the serializer.

C. *Temperature I^2C*

The temperature I^2C is the top file of this entire temperature sensor to display operation. It encases each component and delivers data through a string of code to give the final requested display. As previously mentioned, the WR register and serializer are two major components. However, there are also two registers that store data at the end of the configuration. There was a signal added to get rid of the fractional bits that are not important for this project. The two bytes that make up the 16 bits of data. This data is mapped to the signal. The first 9 bits were used and the other 7 were not. These 9 bits were then port mapped to the inputs of the binary to BCD converter. After being converted into a binary number the serializer decodes and displays the bits.

D. *Pulse-Width Modulation via Linear Interpolation*

Once the 9-bit temperature data is obtained from the serializer, it is to be interpreted as signed. That given, if the MSB of the bit string is '1', then "0000" is written to the signal "duty" as this indicates a negative ambient temperature. If no switches are activated, then the duty cycle is dictated by the input temperature and linearly interpolated between 21 degrees Celsius (the minimum temperature) and 32 degrees Celsius (the maximum temperature). Any lower than 21 degrees Celsius and the duty cycle is 0% (0000), but any temperature 32 degrees and higher produces a duty cycle of 100% (1111). The resolution of the duty signal, in our case, is four bits because higher resolutions are not needed for the purposes of this project. Also, it will be easier to differentiate between different duty cycles from how quickly the motor rotates. If there are switches activated, then the temperature data is ignored and the state of the switches is written to the duty cycle instead.
All the while, a clock input is controlling a 4-bit counter. The counter increases by '1' every time the rising edge of the clock is encountered. So long as the accumulated value of the counter is less than the value of "duty", the program outputs a logic high for the PWM value. But, when the value of the counter exceeds the value of "duty", the program outputs a logic low for PWM value. This entire process is repeated when the value on the counter is equal to that of the period (the period, in this case, is "1111" just so the counter can achieve its maximum value before it resets).

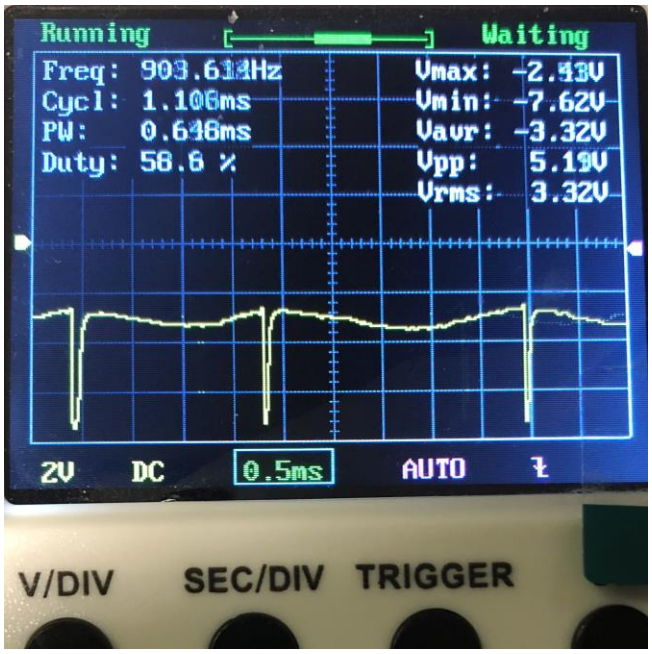
III. Experimental Setup

The setup of the project was completed by creating separate parts of the code that each group member worked on and combining them. They were then tested to ensure that they would run individually and correspond with the hardware as needed. The software was eventually implemented to work into the hardware and tested to make sure important aspects like the values of temperature were displayed properly on the board. Once certain temperatures were able to be calculated through the hardware we were able to focus on the fan. The next step was to get the fan controlled by a DC motor to initiate after it exceeded a certain interval of temperature. The debugging process was crucial because sections of the code were worked on separately and then pieced together to get the final desired code. This created difficulty when exchanging the code and making a final format that was sound with the creation that was anticipated.

Once overcoming the challenges of creating an overall system, it was important to perfect the code that was existing to make the best version possible. We expected to have the results that would give the temperature within the display and ultimately activate the fan when necessary. This created an editing process to clean up some of the code such as getting the proper number of displays to turn on instead of more or less. Some of the seven segment displays were unneeded and more so unnecessary when it comes to displaying the temperature. The fractional bits that output from the ADT7420 are pointless in our situation therefore these bits were dismissed. There are many more code adjustments that were made similar to this to adjust the output so that it was clean and precise.

IV. Results

Ultimately, the design was successful in achieving our goal of creating a temperature-controlled fan. When analyzing the end product, the fan is able to detect the surrounding temperature which is then directly displayed within the 7 Segment display on the Nexys Board. Once this is acquired, the fan activates and ideally serves the purpose of lowering the temperature in a specific area as desired by the user. Through the use of FSM's, parallel shift registers, and a variety of in class topics already showcased, the end result is a configuration that consists of diverse material that was acquired throughout the semester. The results were as expected therefore the project works effectively and is able to collect the data that is necessary for it to function all while activating on its own. Overall, the results were as predicted. The time spent adjusting code and certain components made for a perfect explanation of what was anticipated. We also used an oscilloscope to view the wave form of varying duty cycles. We ran into the issue of Vivado's hardware manager shutting down at high duty cylces. Seemingly we were drawing too much current from the I/O pins so adding a larger resistor on the output of the Nexys board helped
for most instances.



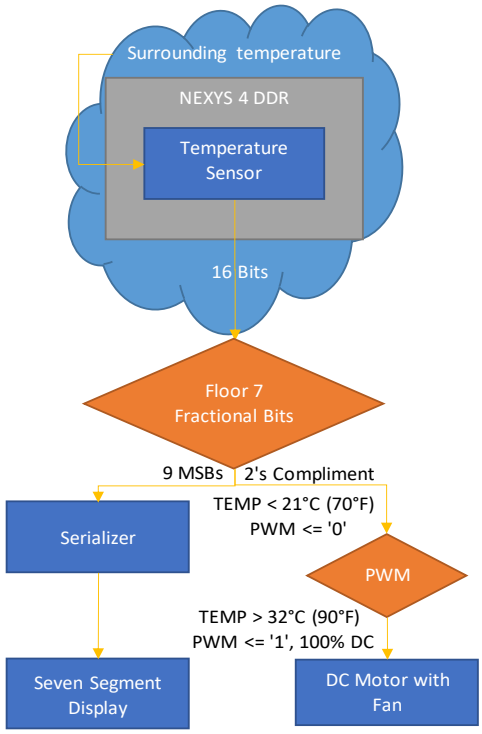


Figure 3. Block Diagram of Temperature Controlled Fan

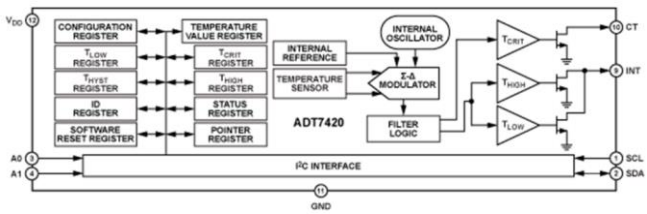


Figure 4. Block Diagram of Temperature Sensor

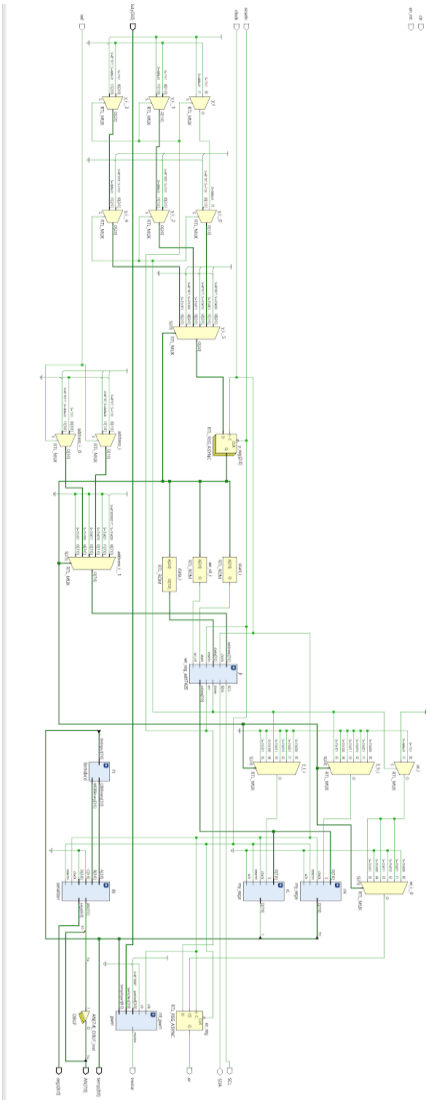


Figure 5. Block Diagram from Vivado (Left)

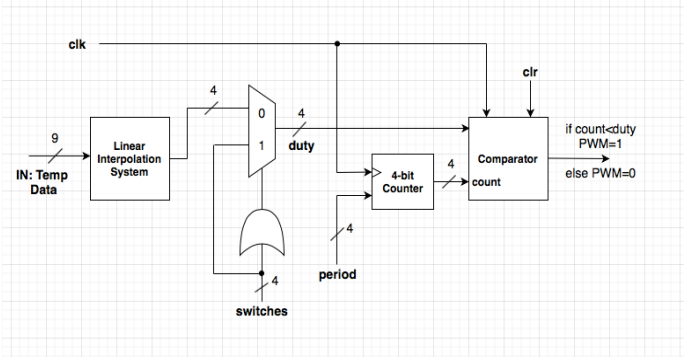


Figure 6: Pulse-Width Modulation System

CONCLUSIONS

Overall, the project displays the real-world applications of the topics covered throughout the course. Through the implementation of the various software and hardware applications, the project is able to grasp the complexity of how the two function together to ultimately achieve a goal that is desired. The debugging process as well as experimenting with ways to implement unique a design, allowed groups to work together and achieve something that could not be done alone. The fan ended up working as desired and our group was satisfied with how the end result came out. If there is one thing that could be changed to better the outcome of this design it would be altering the code, so that if there was a negative temperature the display would show a negative sign. Nonetheless, this is a fan that is meant to cool things down in hot temperatures. Therefore, for this case the negative temperature is not included. Given more time this would be the first thing added so that the display is entirely correct in all temperatures.

REFERENCES

[1] Digilent.(n.d.). *NexysA7referencemanual[Reference.Digilentinc]*. Digilent Documentation [Reference.Digilentinc]. https://reference.digilentinc.com/reference/programmable-logic/nexys-a7/reference-manual

[2] Llamocca, D. (n.d.). *VHDL coding for FPGAs*. https://www.secs.oakland.edu/~llamocca/VHDLforFPGAs.html