PWM Fan Controlled by Temperature Data

Final Report

David Pattison, Jonathan Nguyen, Maxwell Hammond, Ryan Paye
Electrical and Computer Engineering Department
School of Engineering and Computer Science
Oakland University, Rochester, MI
e-mails: dapattis@oakland.edu, jtnguyen2@oakland.edu, maxwellhammond@oakland.edu, rpaye@oakland.edu

***Abstract*— The purpose of this design and experiment is to create a functional temperature cooling system that can be used for a variety of purposes using the Artix 7 Nexus FPGA board. This project will handle the challenge of monitoring CSP temperature while defining temperature limits for optimum operation. The components used in this design are the onboard digital temperature sensor ADT7420, the Diligent KYPD Pmod, Digilent 12V Motor/Gearbox, Diligent HB5 H-bridge driver for interfacing with the motor, and onboard seven-segment displays.**

I. Introduction

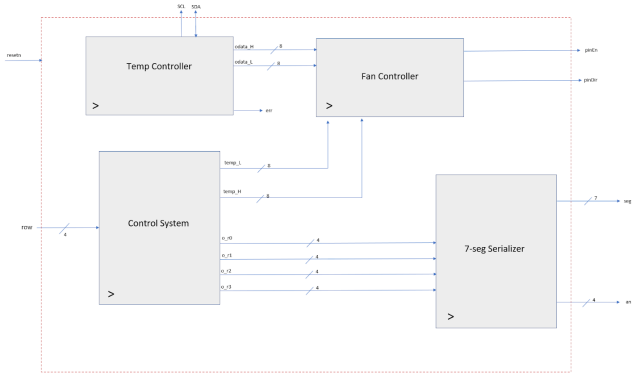
Many devices within a digital systems architecture are required to be maintained within specific operating temperatures. For example, CPU's are often located beneath a heatsink in order to properly function under heavy load. Desktop computers often have several fans within the case to dispel heat, often with limits in place by the developer for fans to turn on, off, or operate at various speeds. Even many home appliances such as air conditioners must maintain proper cooling by interpreting internal temperature data.

The proposed design for this project seeks to emulate these principals within the scope of knowledge and skills developed from this class. The system shall receive high and low temperature limits designated by the user input from the keypad, interpret current temperature data from the ADT7420, and drive the motor controlling a fan accordingly.

The design for this experiment will apply many of the techniques and skills obtained in class and from laboratory experiments such as creating design schematics for cohesive circuit designs, designing ASM charts for logical state transitions and outputs for finite state machines, communication protocols of an I2C device, using LUTs to facilitate necessary binary to BCD conversions, fixed point representation of binary values, and utilizing output devices like the motor and seven-segment displays to perform functions based on input data.

II. Methodology

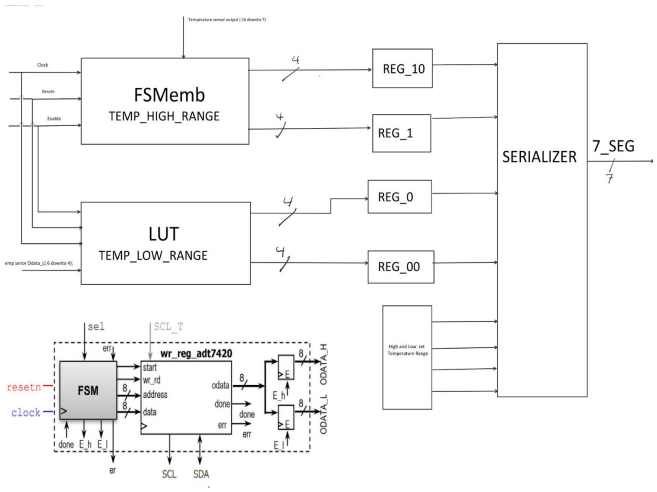
The design of this project utilizes several control systems, each containing various components within. This includes the basic controller for the ADT7420 temperature sensor, the controller for the keypad, the control system which interprets data generated by the keypad, the system for controlling the seven-segment displays, and the fan control system which interprets the temperature data from the ADT7420 sensor as well as the threshold values defined by the user to drive the fan motor when necessary.



(Fig.1 Project Block Diagram)

A. ADT7420 Temp. Sensor

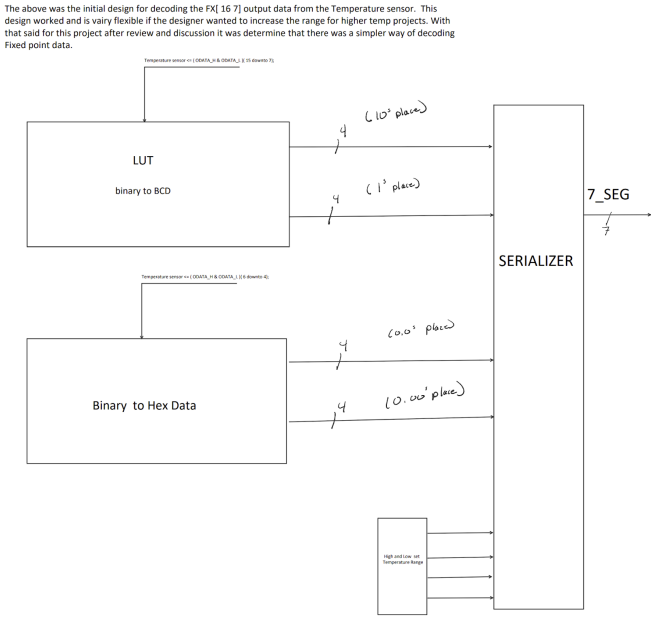
The temperature sensor controller has been provided by Professor Llamocca and will be interfacing with the I2C communication protocol, reading register address data from 0x01 and 0x00 correlating with TEMP_H and TEMP_L. The output result is a 16-bit resolution fixed number of [16 7] FX format. The component that controls I2C communication based on data and address is the wr_reg_adt7420 component. An FSM is also included to issue commands determining reading/writing, and which registers are selected.



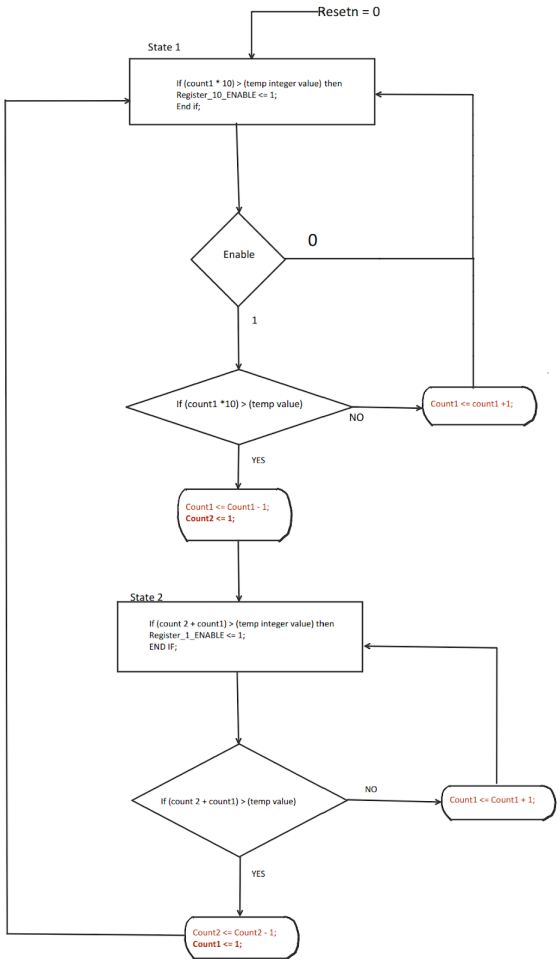
(Fig.2 Temperature Sensor Decoder Circuit)

Because we are not able to drop the temperature below zero this project considers the FX number to be unsigned. The temperature select (sel) will be held at '1' so the sensor will constantly read the current temperature that will be output on the ODATA_H and ODATA_L signals. After at least 2 cycles to fully output the 16-bits will go to the FSM (High_Temp_Range) which will take the [15 down to 7] which are the integer values of the temp sensor. These 9 bits will be computed into two 4-bit vectors that will be displayed on the 7-segment display (refer to Fig.1). Further bits [6 down to 4] will be computed [Low_Temp_Range] into at most 2 decimal places (each place holder will be of 4-bits) for relative temperature accuracy to be displayed. In addition to translating FX[16 7] to the 7-seg display. This design offered great flexibility for much higher temperatures but for this project it was found that a simpler design using LUT would give the same results. Because there were only 7 factional bits a zero was added to LSB so accuracy of the temperature is only to the integer plus 4 factional bits.

The output data will also be read by the fan controller for comparison to the defined upper and lower set temperature limits.



(Fig. 3 Temperature Sensor Decoder LUT)



(Fig.4 Temperature Sensor State Machine Internal Logic)

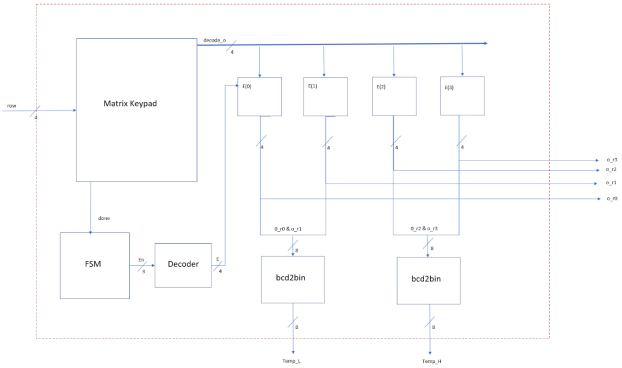
The signals ODATA_H and ODATA_L are appended together to form a signal output_temp in the top file for use on the seven segment controller. To accomplish this, output_temp is fed through a binary to BCD LUT, where input is nine-bits and the output of the LUT is only 8-bits so the most significant bit is not used.

B. Matrix Keypad Controller

The matrix keypad component has been provided by Professor Llamocca and is used to interpret data provided by inputs to the Digilent KYPD Pmod. This is achieved by using a pulse generator and ring counter to constantly scan the value of each keypad column to pair with the selected row. This allows the component to identify which output to select based on the row and column data. The component also debounces the signal, and outputs a "done" signal when all operations are complete, and the 4-bit decoded output is valid. The done signal also will be used in the main control circuit to transition from each state. It should be noted that for the purposes of this project, the only buttons that should be pressed by the keypad are zero through nine. The hex buttons A through F as well as any other characters should not be pressed by the user as it may cause issues.

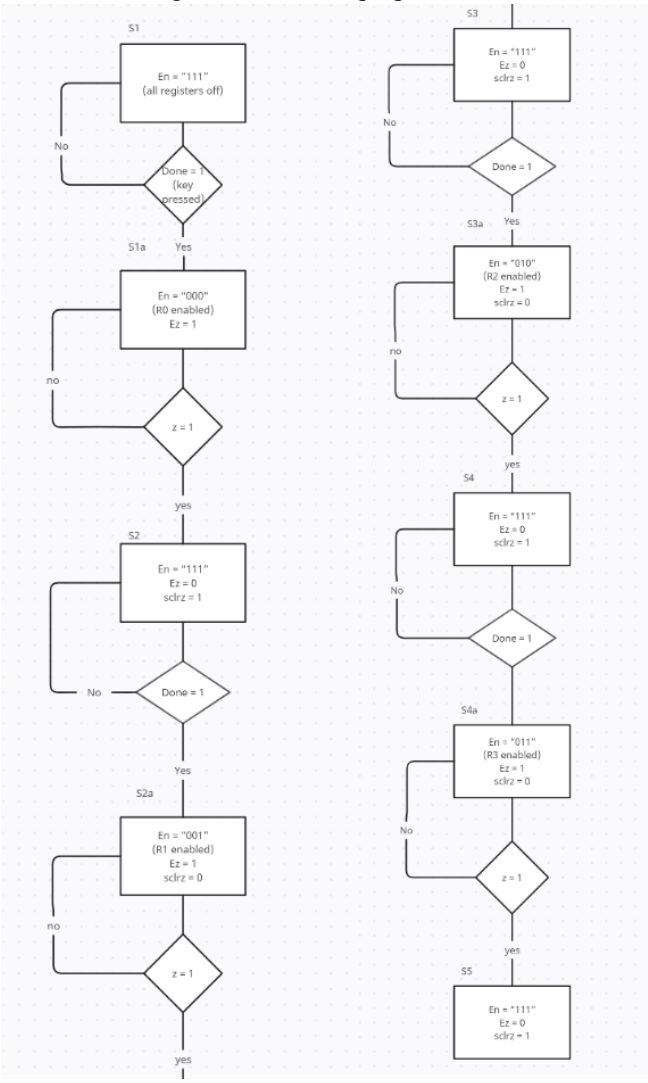
C. Control Circuit

The control circuit receives the decoded output from the keypad as well as the done signal to control its outputs. This component consists of a finite state machine, decoder, four registers, and two BCD to binary LUT components. Since the output of the keypad controller is of four-bits and is used to designate temperature (which will often be higher than the maximum of fifteen degrees from a four-bit value), the proposed design must interpret two button presses. Each digit will be stored as a BCD value into one of four 4-bit registers and used for the seven-segment output. Furthermore, the first and second digit selected will be combined into an 8-bit signal, and the third and fourth digit will also be combined. These 8-bit BCD signals will be converted into binary via two LUTs and used for the low and high temperature thresholds respectively within the control circuit.



(Fig. 5 Control Circuit Diagram)

The enables of each register are controlled by the finite state machine and the enable decoder. This is done to ensure that each sequential button pressed properly stores its respective value. When the done signal is asserted by the matrix keypad component the proper register is enabled on the next clock cycle. In order to ensure data is processed as expected, a counter was implemented and set to a half-second delay before turning all registers off. Implementing the counter in this way ensures that data is not being written into two registers using a single button press. Once the counter has finished, the circuit waits for "done" to be asserted once again to write into the next register, until all registers obtain the proper values.

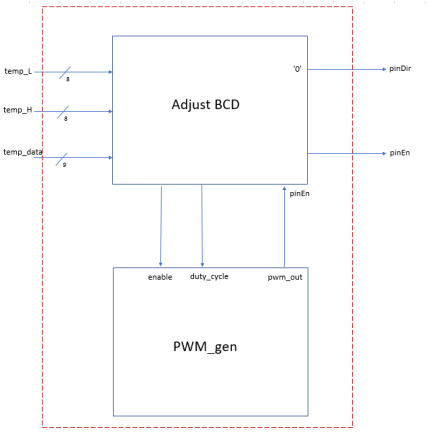


(Fig.6 Control System FSM Chart)

D. Fan Controller

This component is designated to the output of the system, where it will generate an output vector for the enable pin of the HB5 H-Bridge based on the previously obtained

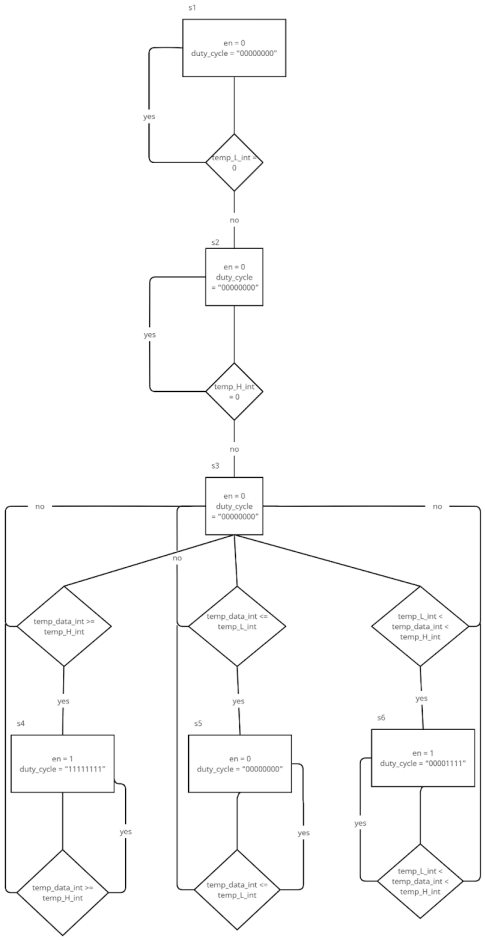
temperature data. The component uses a PWM generator (modified component sourced via Jonas Julian Jensen of vhdlwhiz.com [1]) with enable and a finite state machine to consider the current temperature, and the temperature threshold values the user input via the keypad to control the motor.



(*Fig. 7 Block Diagram of Fan Controller)*

This is achieved by converting the threshold data inputted by the user via the control circuit into integer values, and additionally converting the current temperature data given by the ADT7420 temperature sensor component into an integer. The finite state machine checks whether the current temperature is within the two threshold values, above the high threshold temperature, or below the low threshold temperature. The PWM generator component is capable of writing onto the enable pin of the HB5 H-Bridge several times within a set time frame to drive the motor at various speeds, and shutting off when an enable signal is driven low.

When the current temperature lies below the low threshold, the controller sets the enable for PWM low thus causing the motor to deactivate. If the current temperature lies in between the low and high threshold (the temperature that is determined to be maintained), the duty cycle for PWM will be established at approximately 50% and enable driven high. Lastly, when the current temperature is above the high threshold (meaning the temperature must be dropped quickly), the PWM duty cycle will be set to 100% with the PWM enable driven high.



(Fig. 8 Fan Controller Logical ASM Diagram)

E. Seven-Segment Display Controller

The final control component will be used to display various useful information to the user throughout the program's runtime. Initially the left four displays will denote zeros until the user inputs both digits of the low temperature threshold. The right four seven-segment displays denote the current temperature data read by the ADT7420 temperature sensor in [16 7] FX format. Of these four displays (from left to right), the first two take the upper nine bits read from the sensor converted from binary to BCD but neglects the most significant bit. The last two displays output the precision bits, which are preserved in their hex format. The precision bits were chosen to remain in this format due to time constraints and difficulty during implementation.

III. Experimental Setup

The project was experimented via simulation in Vivado using a testbench. Various signals such as the inputs and outputs of the registers in the control system, states of the finite state machines, and keypad signals have been

monitored via timing diagram and ensured to be functioning as intended. One of the hurdles when performing experimental testbenches arises from the implementation of the keypad component which must account for the bouncing signal during key presses. Furthermore, temperature data being read from the ADT7420 sensor could/would not be implemented in a testbench setting. The group had to perform many trials of debugging live on the FPGA board rather than relying on data from the testbench. Debugging on the FPGA board consisted of testing keypad inputs and ensuring the motor was enabled, disabled, and duty cycle adjusted properly depending on temperature data, and utilizing an ice pack directly onto the temperature sensor in order to drop current readings.

IV. Results

The results of the behavioral simulation concluded that multiple keypad button presses resulted in the proper values being written into registers 0-3 and proper state transitions within the control circuit. The input and output signals of the binary to BCD converter LUT were monitored in binary format and determined to reach their proper values for use by the seven-segment serializer. The signals for temp_L and temp_H denoting the threshold values were found to be working as expected. The components within the fan controller could not be determined to be functional due to parameters that could not be adjusted within the testbench, and were thus tested live on the FPGA board.

Firstly, the current temperature reading would be determined followed by setting the low threshold two degrees below the current, and the high threshold around the current temperature. Once it was determined that the motor was driven with 100% duty cycle as the temperature surpassed the high threshold, an ice pack was applied to the sensor. Eventually when the current temperature landed just below the high threshold, the motor was observed to have slowed down to the 50% duty cycle setting. The ice pack was continually applied until the current temperature dropped below the low threshold and the motor shuts off entirely. This routine was performed several times until the conclusion was reached that the implementation was working as intended.

Conclusions

The group has achieved their initial goal in developing a system using the Artix A-7 FPGA board to receive temperature data using the onboard ADT7420 temperature sensor, manually denote temperature thresholds using the Digilent KYPD PMOD, display relevant information using onboard seven-segment displays, and drive a motor using the Digilent HB5 H-Bridge with PWM according to the relevant input variables.

Though this project functions as a fan capable of maintaining temperature of a component in theory, for the purposes of this assignment this project should be used more so as a proof of concept that it could be feasible to scale up into a project with real world functionality.

To accomplish this, a temperature sensor that does not exist onboard the FPGA would likely be used and mounted to or within a component desired to be monitored. A more powerful motor would also likely be incorporated into the project and mounted to a heatsink to deliver the cooling and ventilation necessary for a component that undergoes heavy loads (such as a computer's CPU). Other ways the project could be adjusted for real world applications could be including functionality to convert degrees Celsius into Fahrenheit, implementing further offset temperatures to control the duty cycle of the motor, and incorporating variables to shut off the system entirely when it is determined that ideal temperatures cannot be maintained.

References

[1] Jensen, Jonas J. “How to Create a PWM Controller in VHDL.” *VHDL Whiz*, 19 May 2020, https://vhdlwhiz.com/pwm-controller/. Accessed 18 Apr. 2023.