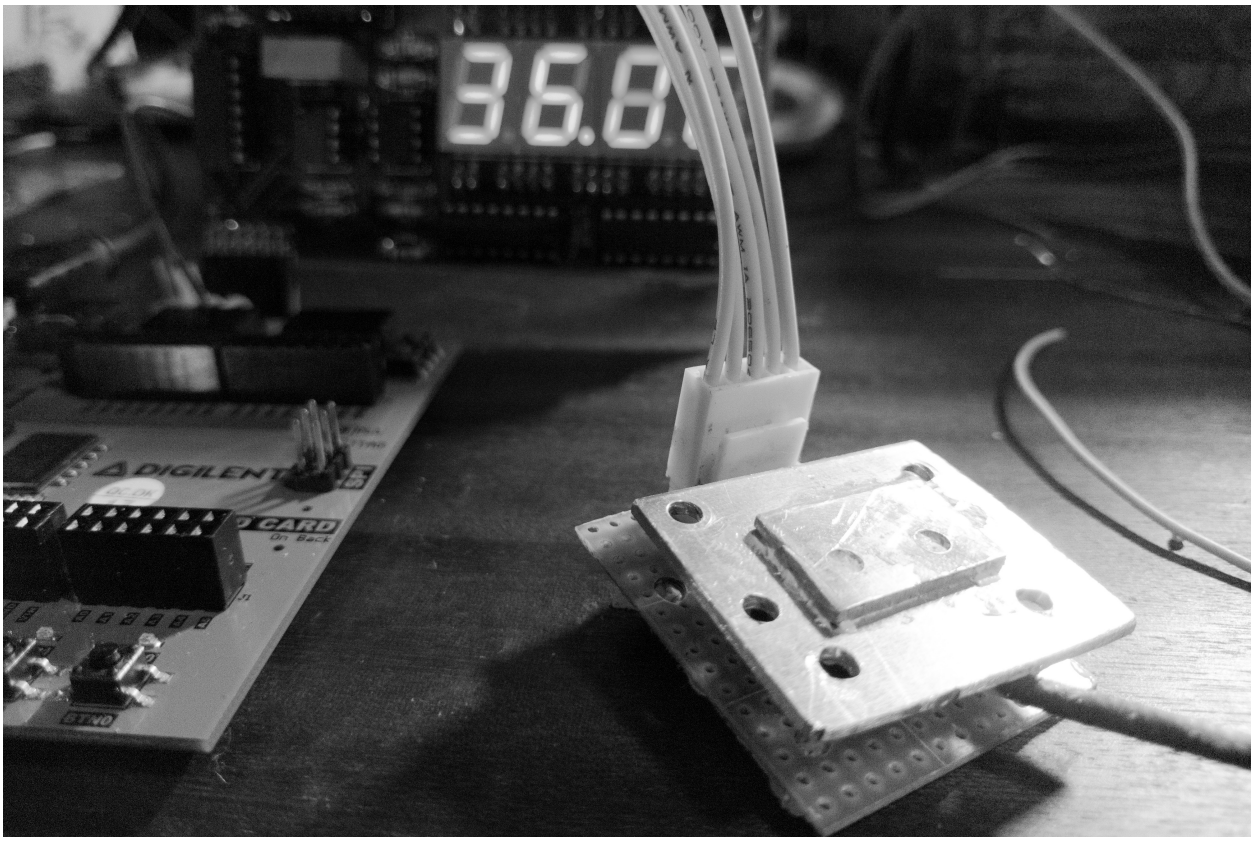
On-chip Temperature Sensor

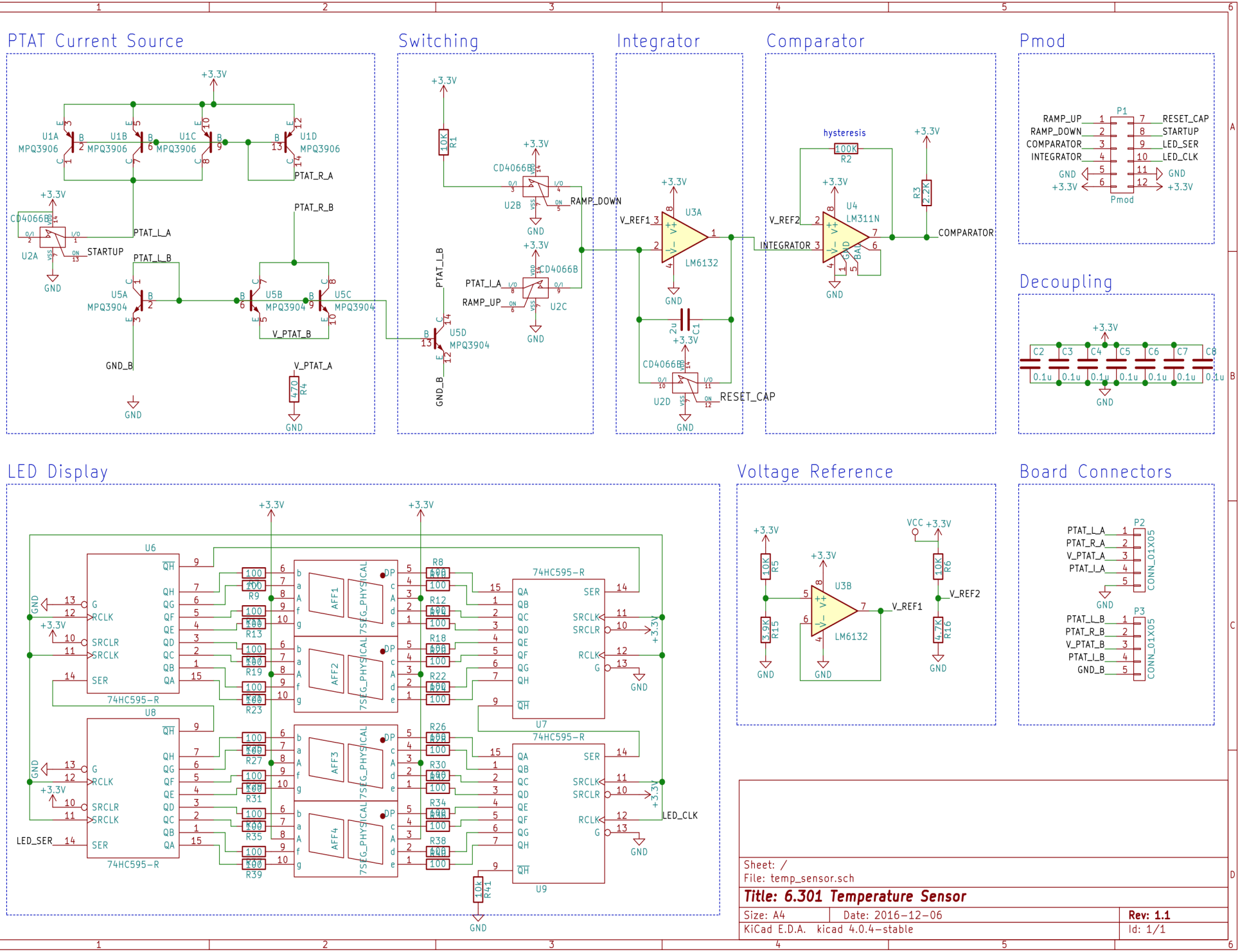
6.301 Final Project

Mitchell Gu
December 13, 2016



Summary

In this report, I present my design for a temperature sensor implemented using a proportional-to-absolute-temperature (PTAT) cell, a dual-slope ADC, and digital controlling logic programmed onto an FPGA. The sensor was found to have excellent linear temperature response consistent with theoretical and simulation results. After calibration, the sensor was accurate to within 3°C in a 20°C - 80°C range. The sensor was assembled on a custom PCB with 7-segment readout and a Pmod connector for maximum robustness and usability.



Sheet: /
File: temp_sensor.sch

Title: 6.301 Temperature Sensor

Size: A4	Date: 2016-12-06	**Rev: 1.1**
KiCad E.D.A. kicad 4.0.4-stable		Id: 1/1

I. The PTAT Cell

Design

A simple way to generate a current or voltage that is proportional to absolute temperature is to take advantage of the $V_{th} = \frac{kT}{q}$ term in the diode equation for BJTs.

$$I_c = I_s \left(\exp\left(\frac{qV_{BE}}{kT} \right) - 1 \right)$$

Given a certain collector current in a BJT, if the temperature increases, the base-emitter voltage must increase proportionally to maintain the same current. By varying a combination of the I_s value and collector current ratio of two BJT's with their bases tied together, one can extract a PTAT voltage as the difference in their base-emitter voltages. My implementation of this type of PTAT cell is shown in Figure 1.

Assuming all NPN and PNP transistors are matched and ignoring base currents, one can observe that there will be three times the collector current through the diode-connected NPN BJT than the parallel-connected NPN pair. Thus each NPN BJT in the pair will have one about sixth the collector current of the diode-connected NPN BJT. This results in a difference in base-emitter voltages, a voltage across the resistor, and a output I_{PTAT} that are all linearly proportional to temperature.

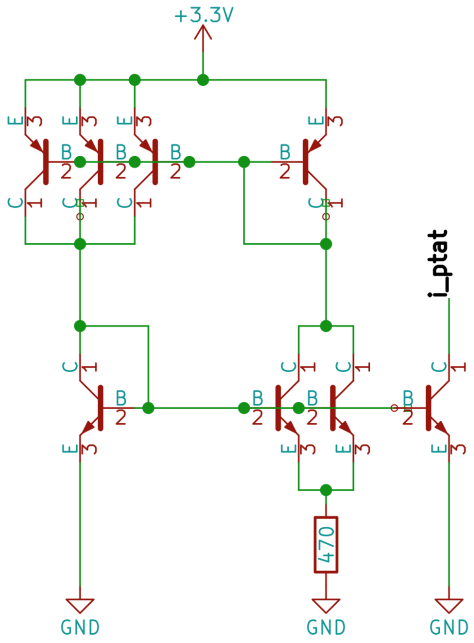


Figure 1: The PTAT cell design, featuring a PNP current mirror with a 3:1 current ratio and a doubled up NPN BJT for increased effective I_s.

Note on BJT counts and matching

This proportionality can be made steeper by increasing the number of parallel connected resistors, but in practice this was limited by the number of transistors in each trasistor array IC used (4 for both the PNP and NPN transistor array ICs). Designing for more than 4 of each type would require more than one transistor array IC, in which case there are no guarantees on transistor matching from batch to batch. While the MPQ transistors used in each IC weren't ideal in that they weren't fabricated on a monolithic die, they proved to be sufficiently matched for the purposes of the project.

Detailed analysis

From the prior analysis and the diode equation, the difference in base emitter voltages should be $\Delta V_{BE} = V_{th} \ln 6$. This yields a current across the resistor of $\frac{kT}{q \cdot 470} \ln 6$. Since the output transistor's base-emitter voltage is the same as the diode-connected transistor's, the output current will be roughly three times the resistor current, and $I_{PTAT} = \frac{kT}{q \cdot 470} \ln 6 \cdot 3$.

To take account for base currents, one can observe that 4 base currents are being added to the collector current of the parallel NPN pair, and $2.\overline{3}$ base currents are being subtracted from the collector current of the diode-connected NPN on the left (the parallel BJTs draw roughly one third the base current). The overall ratio of the collector current of the NPN pair to the collector current of the diode-connected NPN is then

$$I_{\text{gain}} = \frac{3}{(1 + \frac{4}{\beta})(1 + \frac{2.\overline{3}}{\beta})}$$

and the PTAT current is

$$I_{PTAT} = \frac{kT}{q \cdot 470} \ln\left(2 I_{\text{gain}}\right) \cdot \frac{\beta}{\beta + 1} \cdot I_{\text{gain}} = 8.847 \times 10^{-7} \cdot T$$

when $\beta = 100$. Simulating the PTAT cell in LTSpice with identical transistor parameters yielded output currents that agreed with my calculations exactly. (Figure 2)

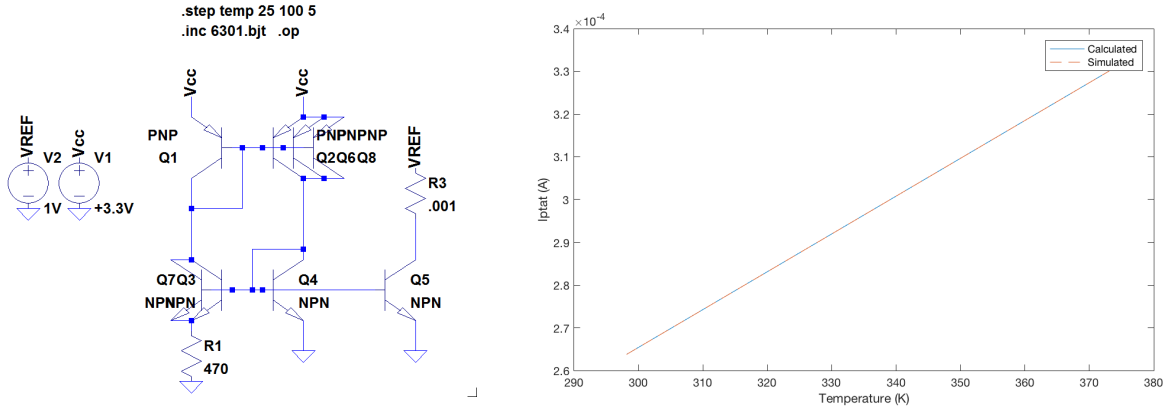


Figure 2: LTSpice PTAT Cell Schematic (left) and comparison of simulated vs calculated output current (right). The lines are exactly coincident.

The 470Ω resistance was chosen in order to yield currents large enough to charge the $2\mu F$ integrator capacitor during the ramp up period to a significant voltage voltage without clipping into the positive voltage rail (3.3V). $2\mu F$ was chosen as a relatively large capacitance, flexible to change while still being well in the range of ceramic or film capacitors (not electrolytic).

II. The Dual Slope ADC

Design

The theory behind the dual slope ADC is to integrate the PTAT current over a known time (the ramp up time), then discharge the integrator using a known current and measure how long it takes to return to the initial voltage. (the ramp down time). If implemented correctly, the ramp down time should be directly proportional to the PTAT current. A LTSpice schematic of the ADC, used for simulation, follows in Figure 3)

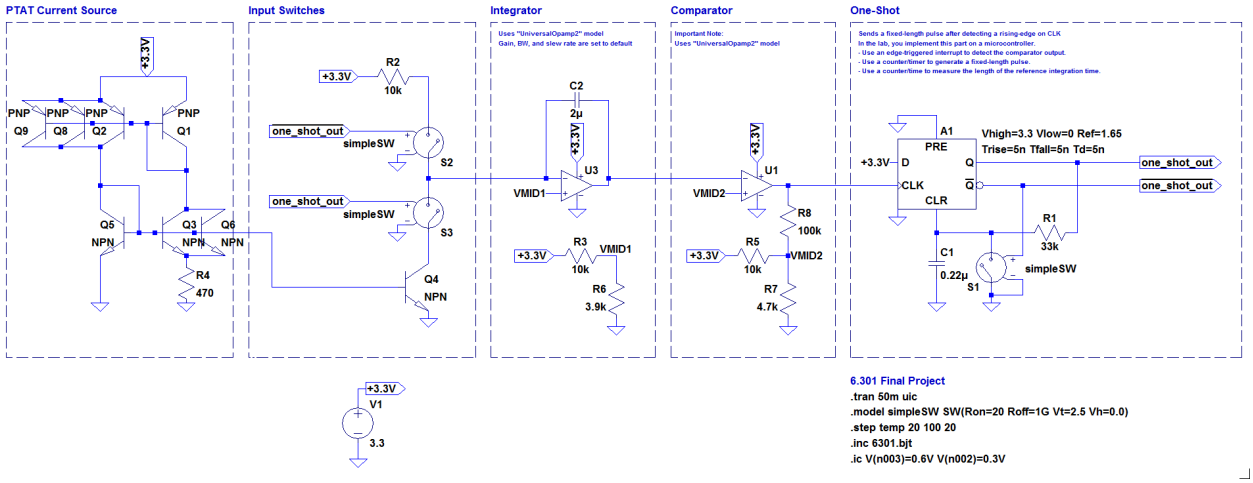


Figure 3: LTSpice schematic of the ADC

During the ramp up time, the integrator input is connected to the PTAT current source by an analog switch so that current will be drawn from through the capacitor, charging it. During the ramp down time, the integrator input is connected instead to a 10k resistor tied high so that current will flow into the capacitor's input side, discharging it.

The integrator output is connected to the inverting input of a comparator and the other input is connected to a constant reference voltage, 1V in my design. When the comparator transitions high, the integrator has discharged to lower than the reference voltage, marking the end of the ramp down period. Digital logic is then responsible for resetting the integrator (by shorting the cap), then switching to the ramp up state and recharging the integrator for the ramp up time.

The reference voltage for the comparator is intentionally set slighly higher than the reference voltage for the integrator so that after the integrator's capacitor is reset to the integrator's reference voltage, the capacitor will charge up, but the ramp up time will only start once the capacitor has charged past the comparator's reference voltage, sending the comparator output back low. Since the capacitor doesn't reset to the same potential as the comparator's reference voltage, this mitigates the effects of start voltage inaccuracy and comparator offset voltage.

A small amount of hysteresis (~10mV) is added to the comparator to prevent false transitions about the reference voltage that would interrupt and pollute the sampling process.

Ramp down times

To calculate the ramp down time for a particular PTAT current, first the peak integrator voltage (relative to reference) is calculated: $V_{pk} = \frac{I_{PTAT} \cdot \text{RAMP\_UP\_TIME}}{2\mu F} = 2500 I_{PTAT}$ with a where the ramp up time is fixed at 5ms. In my circuit, the integrator reference voltage was 0.926V, so with a 10k resistor during the ramp down time, the discharging current is $237.4\ \mu\text{A}$. To discharge the peak voltage with this current takes $\frac{2500 I_{PTAT} \cdot 2\ \mu\text{F}}{237.4\ \mu\text{A}} = 21.06 I_{PTAT}$. For the range of I_{PTAT} currents of $241.7\ \mu\text{A}$ to $330.1\ \mu\text{A}$ across 0°C to 100°C, the down ramp times should be between 5.09 ms to 6.95ms. The $10\ \text{k}\Omega$ discharge resistor was chosen to yield a ramp down time similar to the ramp up time so the measured interval wouldn't be too steep or drawn out.

To confirm these calculations, a LTSpice simulation was run for temperatures from 0°C to 100°C (Figure 4). The simulated ramp times agree exactly with the calculated range.

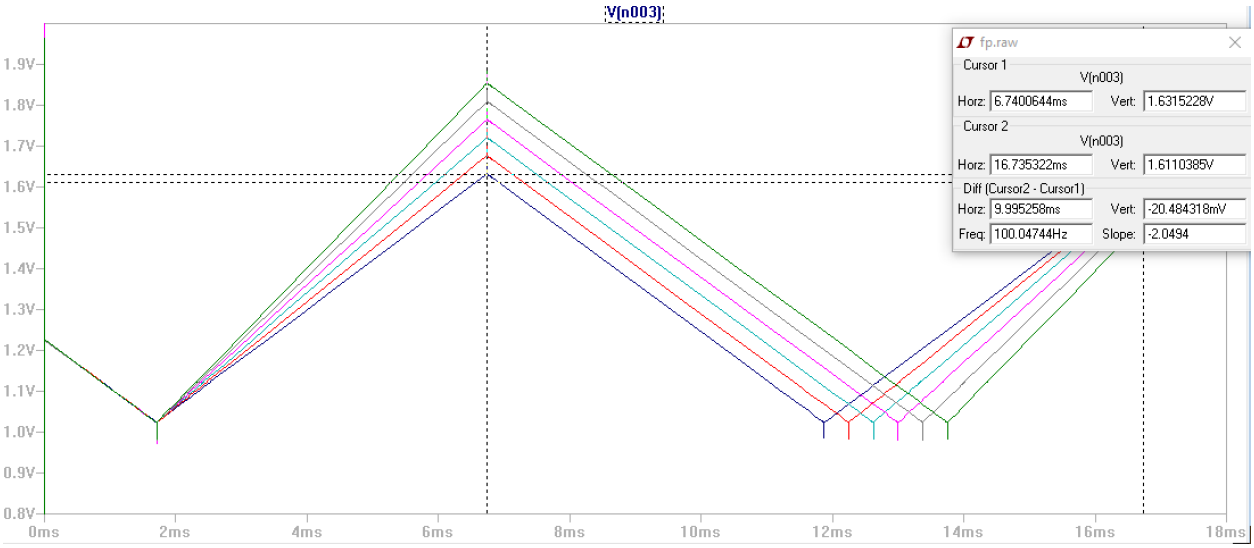


Figure 4: The simulated comparator output, with ramp down times changing linearly from 0°C to 100°C.

III. The Digital Logic

Instead of an Atmel microcontroller, I opted to use an FPGA development board to switch the ADC and measure the down ramp time. The FPGA, running at a 65MHz clock, could guarantee very low latencies for all control signals and also provide high resolution when measuring the down ramp time. In contrast, I felt that programming a 16MHz microcontroller in C with slow I/O libraries would be slower and prone to more bugs.

ADC Controller Module

The digital logic for the ADC control is in the form of a four state state machine:

- **RESET:** This state asserts a digital output to reset the integrator output to the integrator reference for 100 μs, after which it switches to the WAIT state.

- **WAIT:** This state asserts the ramp up switch and waits for the comparator's output to return to low, indicating that the integrator has past the comparator's voltage reference and the RAMP_UP state should start.
- **RAMP_UP:** This state continues to assert the ramp up switch and simply counts until exactly 5ms has passed, after which it switches to the RAMP_DOWN state.
- **RAMP_DOWN:** This state asserts the ramp down switch and maintains a clock cycle counter until the comparator switches back high, indicating that the integrator output has returned to the comparator's voltage reference. The value of the counter when this transition happens is saved for conversion into a temperature.

Temperature Conversion Module

This module takes two statically defined parameters: the expected ramp down clock cycles for 0°C and the expected ramp down clock cycles for 100°C. From an input bus containing the cycles it took to ramp down, it calculates a temperature between 0°C and 100°C in the form of that temperature multiplied by 10 in a 10 bit bus to provide 1 decimal digit. This conversion requires a divider module to effectively multiply the ramp down cycles by the slope of the line determined by the two parameters.

Display modules

Once a 10 bit representation of the temperature is calculated, the latest temperature is output to both a 4-digit 7 segment display soldered to my custom PCB and over HDMI video to an external monitor in the form of a time domain graph.

The 7 segment display is updated via 4 daisy chained 74HC595 8-bit shift registers so that only two pins are required: serial data and clock. The 10-bit temperature goes through a conversion to binary-coded decimal, then is shifted out serially by way of another state machine.

The HDMI display is run at a fixed XVGA (1024x768) resolution, and at the end of each frame, the latest temperature value is shifted into a 1024 element circular buffer to store the last 1024 temperatures. The temperature graph can then be rendered easily by addressing the circular buffer according to the current horizontal pixel position in the video frame.

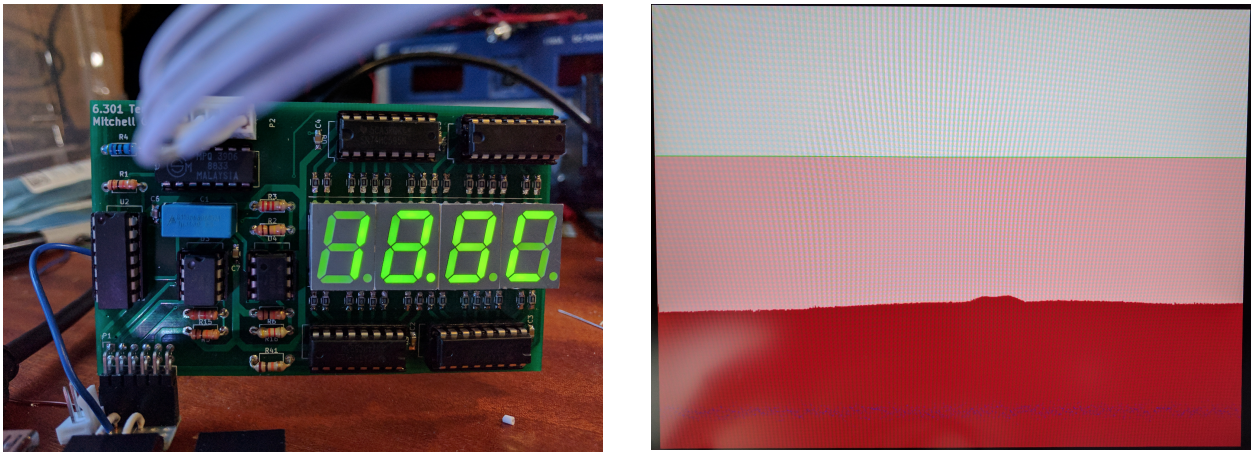


Figure 5: The 7 segment temperature display mid-update (left) and the HDMI temperature graph (right)

IV. Calibration, Testing, & Measurements

Thermal configuration

Based on my experience breadboarding the PTAT cell, I knew that heating the NPN transistor array uniformly and accurately would be critical to getting good measurements. In my breadboard configuration, simply heating one side of the array instead of the other could swing the temperature past bounds in either direction easily. For the PCB version, I first made sure to isolate the NPN array on a separate board. To ensure even heating, I affixed a 1/8" aluminum heat spreading plate to the top of the transistor array with Arctic Alumina thermal adhesive (Figure 5). Since aluminum conducts heat well, I figured a relatively thick sheet between the heating element and the IC would keep the heat distribution on the IC side fairly uniform. This configuration also allowed me to place the thermocouple on the IC side of the heat spreader to ensure that I was measuring the temperature at the IC rather than at the heating element.

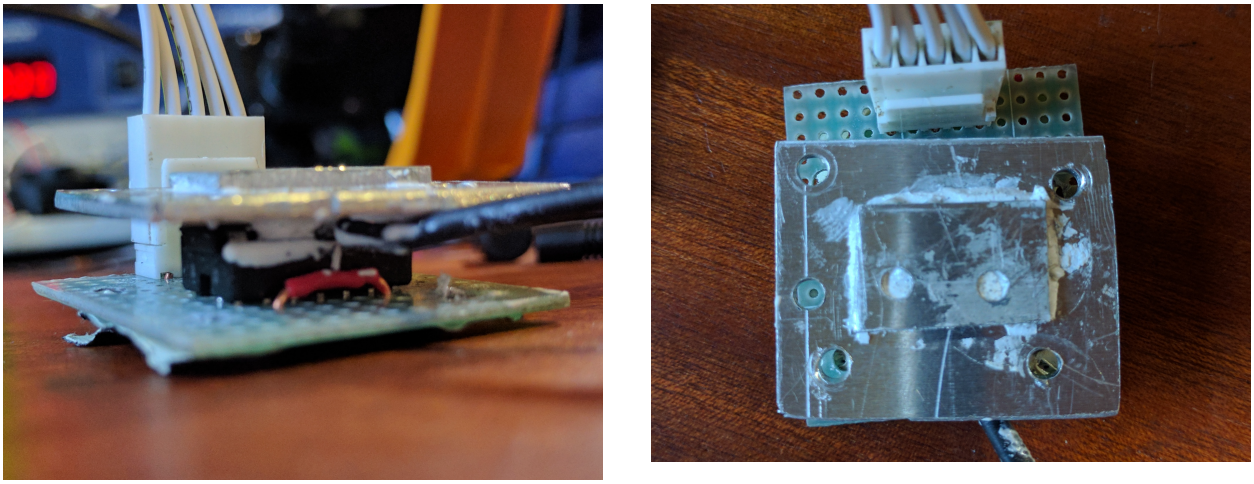


Figure 6: A side view of the aluminum heat spreader and thermocouple mounting position (left) and a top view of the heat spreader (right)

I used peltier elements to heat the upper face of the heat spreader by securing the heated side directly onto the heat spreader with tape. To accelerate the heat transfer, I mounted a heatsink and fan to the cool side.

Temperature calibration

Once I assembled the PCB, I noticed that the PTAT current was offset to a higher current than my calculations predicted (about 310 μA instead of 240 μA), likely due to imperfect transistor matching. As a result, my initial values for the 0°C and 100°C down ramp cycles parameters didn't map to 0°C and 100°C in real life. In order to calibrate the scale, I took two measurements as far apart as possible (usually room temperature and 70°C), calculated the new values of the two parameters, and reprogrammed the FPGA. Eventually, after some practice, I could get the calibration dialed in pretty tight.

Performance & Linearity

The temperature response of the sensor was overall very linear and only deviated from the thermocouple temperature by 5°C. Figure 7 shows a plot of a ramp of the sensor temperature from room temperature to 72°C. The plot's linear regression, as well as the correlation coefficient of 0.9929 are indicative of a strong linear response.

Something interesting that shows up in the residual plot is a swing from 33°C to 49°C from 2 degrees higher than actual to 2 degrees lower than actual. This is representative of a pattern I saw fairly often where the sensor drifts a few degrees off, then after a minute or two drifting back to center. I've yet to be able to explain the phenomenon, as the surrounding environment was always constant, but the pattern isn't too severe. The remaining residuals appear fairly random and more in line with expectations.

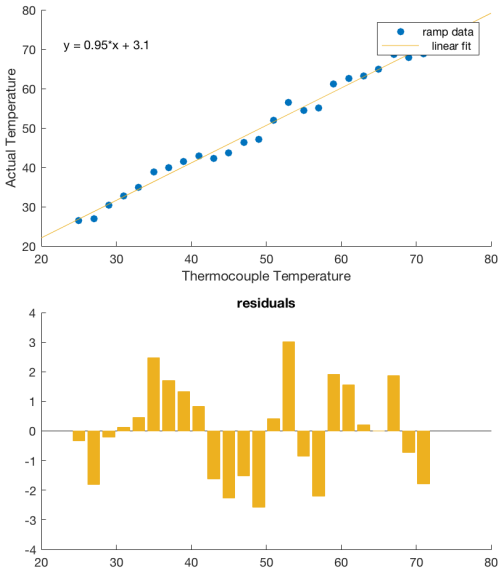
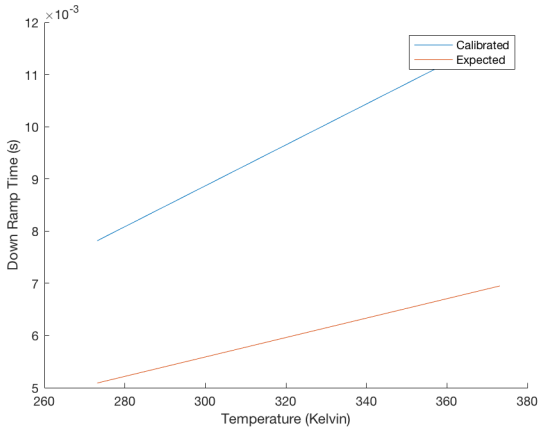


Figure 7: A plot of the sensor output during a temperature ramp and the residuals with a linear regression.

Actual Down Ramp Times

To look at the discrepancy between the actual circuit and the theoretical model, I compared the calibrated 0°C and 100°C ramp down times to the theoretical ones (Figure 8).

The figure shows that the calibrated down ramp times have significantly increased slope than the theoretical ramp times. This is consistent with my observation of higher PTAT currents in my PTAT cell than expected, which I think is due to slightly unmatched transistors. For example, if the PNP current mirror has a 4:1 ratio instead of the expected 3:1, the output PTAT current will be scaled by a factor of over $\frac{4}{3}$, as derived in part I.



Irregularities

While the vast majority of time, the design performed as expected, there remain two quirks about my circuit I had trouble diagnosing.

- The first is an equilibrium that the PTAT cell often settles to when the main PCB is reconnected or the FPGA is reprogrammed. As opposed to the equilibrium where there no current flows through the PTAT cell, this equilibrium forces the base-emitter voltages to around 0.9V and conducts unhealthy amounts of current through analog switch. Since the analog switch is the main affected component I could imagine that the ramp up switch is stuck on and perhaps the reset_cap switch is on at the same time so that the integrator output can source current through the PTAT cell's output BJT, but it shouldn't be possible for both switches to be on with the FPGA logic, and therest of the PTAT cell should drive the system away from that operating point.
- The second symptom was infrequent but consistent glitching of the dual slope ADC where the integrator seemingly fails to ramp up, and the ramp down period thus ends prematurely. I suspect this results from some part of my ADC control timing, but haven't been able to track it down yet. In the meantime, I implemented a 3-element median filter that filters out any outliers outliers before they make it to the display modules.

V. PCB Design

Coming into this project, I had only designed PCBs in Eagle, but I had been meaning to learn KiCAD for a while, and there wasn't much to go wrong with this board, so I decided to try KiCAD. This was also be my first time ordering directly from a Chinese vendor instead of OSHPark, so I was fairly conservative on all the clearances and trace widths when I sent out the board.

Since this wasn't high frequency project, I didn't think too much about parasitics when I routed the board. I mainly placed a ground plane on the bottom layer and tried to route as much of the traces on the top as possible. I also made sure to route the shift register serial and clock lines together and placed $0.1\ \mu\text{F}$ bypass capacitors next to each powered IC.

After I sent out the boards, I realized that I had reversed the power and ground rails on the Pmod connector to the FPGA, so I had to solder an adapter board to compensate. I also realized I could have routed things a lot neater and faster if I had used KiCAD's "interactive" push and shove router and convinced myself that 6 mil traces is plenty wide for most signals. In the end, I rerouted the whole board just to get more practice and learned a few more KiCAD features on the way. The following figures show the most updated gerbers.

VI. Appendix

The KiCAD files and Verilog sources for this project are available at http://github.com/mitchgu/6301-temp-sensor (http://github.com/mitchgu/6301-temp-sensor)

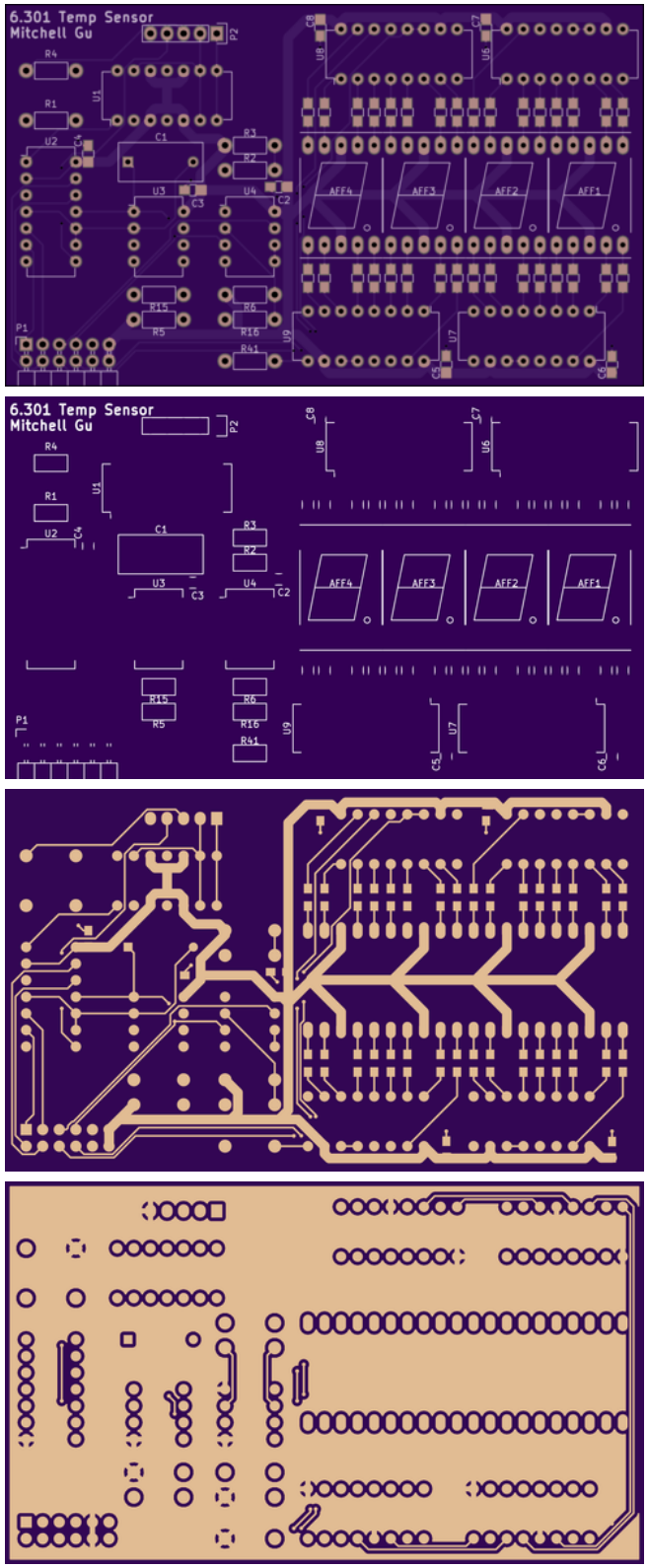


Figure 8: Overall view of board top (first), top silkscreen (second), top copper (third), bottom copper (fourth)