FPGA Accelerated Tactile Sensing Textiles Final Report

1st Austin White
Department of Electrical Engineering and Computer Science
Massachusetts Institute of Technology
Cambridge, MA, USA
akwb@mit.edu

2nd Tiffany Louie
Department of Electrical Engineering and Computer Science
Massachusetts Institute of Technology
Cambridge, MA, USA
tklouie@mit.edu

***Abstract*—Research into flexible sensors has become increasingly important in developing human computer interactions and fine motor control. The Computational Design & Fabrication Group (CDFG) lab in CSAIL is currently developing a piezoresistive textile sensor where wires are embroidered in an array across a piezoresistive sheet, and pressure on the material reduces the resistance. This sampling on normal microprocessors is limited by speed, only reading a 32 x 32 array up to 10Hz max. We utilized FPGA design to speed up this process and provide more data through a greater sampling rate, data analysis, VGA and UART visualization at up to 80 times the original array scanning rate.**

***Index Terms*—Digital systems, Field programmable gate arrays, Tactile Sensors**

I. Design Overview

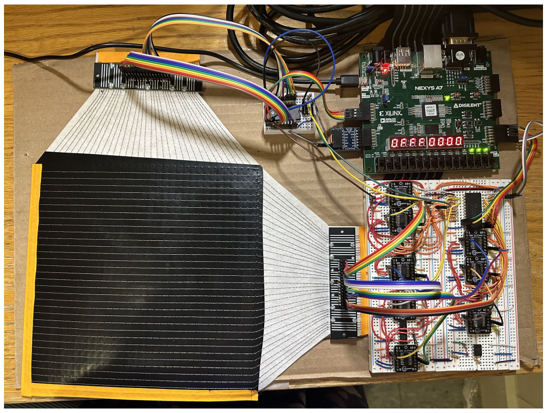

Fig. 1. Image of System

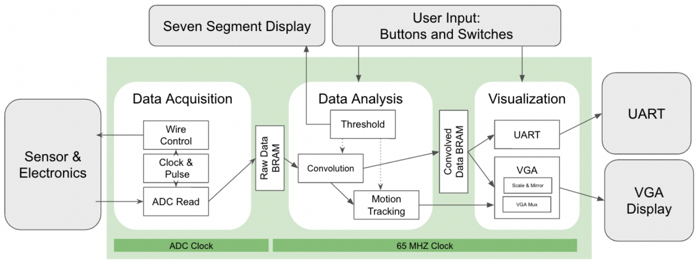

Fig. 2. FPGA Block Diagram.

The system consists of the embroidered sensor, the sensor reading electronics, an FPGA, and visualization outputs including the VGA and UART. The FPGA design is split into three main parts: data acquisition that controls the sensor electronics, data analysis that refines and extracts tracking information from the data, and a visualization of the sensor data. Users can control settings in the data analysis and visualization using switches and buttons on the FPGA.

II. Physical Construction

The sensor and the readout electronics are adapted from the sensors in the IntelligentCarpet system and tactile gloves [2]. These readout electronics are also based on recommendations listed in a paper exploring piezoresistive array sensor readout options [3].

The system operates off two voltages, 3.3V which is provided from the FPGA and powers the electronics. The V_{ref} refrence voltage is 1.66V, which is provided by a voltage divider from the 3.3V and buffer.

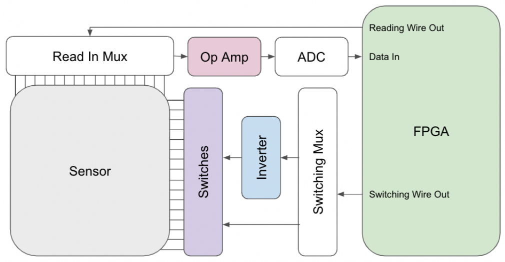

Fig. 3. Electronics Block Diagram

A. Sensor

The sensor is an embroidered *Velostat* sheet with conductive silver thread. The threads are arranged in a grid, with lines on either side of the material. The array in which the points cross are the readout points for the sensor. The sensor reads out a voltage proportional to the resistance (and consequently the pressure) applied on the point of the crossing.

B. Switching Side

All wires are set to V_{ref} except for the selected wire, which is driven to ground. The selected wire is controlled by a *CD74HC4067* multiplexer, each output signal is also connected to a *SN74HC14* inverter. The inverted signal is connected to a *74HC4316* SPST switch held at V_{ref} and the original signal is connected to another SPST switch driven to ground.

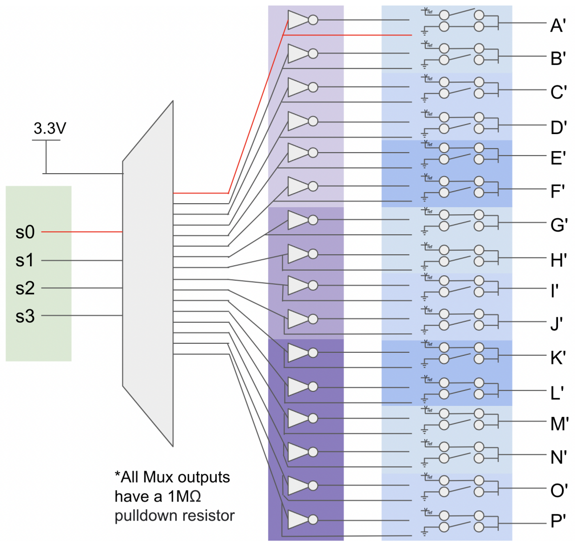

Fig. 4. Switching Electronics where wire A' is selected

C. Reading Side

The array is scanned through each of the read-in wires. The wire is selected using a *CD74HC4067* multiplexer. The signal is then passed into the negative terminal of a *LT1632* op amp with a 100 Ohm feedback resistor. The output of the op amp is wired into the *AD7476A* ADC.

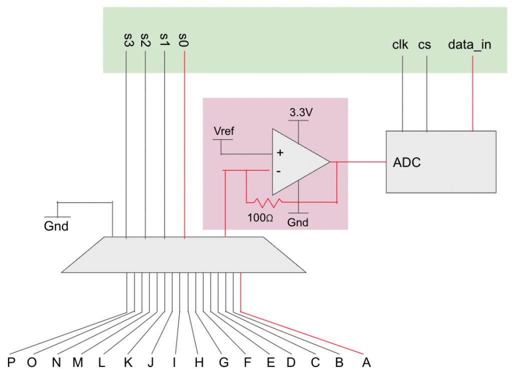

Fig. 5. Reading Electronics where wire A is selected

III. Data Acquisition

A. Clocks

Our reference clock for the design comes from a 65 MHz VGA clock, which is generated using the 100 MHz FPGA base clock and a Xilinx clock wizard generator block.

The ADC clock comes from dividing the VGA clock by an integer parameter `ADC_CLK_DIVIDE`. This is so we can have a separate control from the scanning frequency of the sensor to the display and computations. We designed our system to work on this ADC clock by calculating the array scanning rate at:

$$\frac{65,000,000 \text{ Mhz}}{\text{ADC\_CLK\_DIVIDE} * (16 + \text{ TQUIET}) * \text{RD\_WIRE\_CNT} * \text{SW\_WIRE\_CNT}} \quad (1)$$

B. Wire Control

To ensure that the wires are only switching when the ADC has finished reading out the value, the pulse generation module counts how many ADC clock cycles it needs for the ADC to read and output the value indicates that the reading wire should be changed by creating a pulse. For every `RD_WIRE_CNT` number of wires, the signal to change the switching wire is also pulsed out.

A counting module controls the switching and reading wires by counting from 0 to the max number of wires (stored in `SW_WIRE_CNT` and `RD_WIRE_CNT`). The counts are incremented once for every corresponding pulse the module receives. The values are wired from PMOD headers to the mux that controls the corresponding wire.

C. ADC Read

The FPGA communicates with the ADC following a SPI protocol and operates using the ADC clock. When chip select is driven low, the ADC outputs four leading zeros, then 12 bits of value. The module drives the chip select signal low, verifies these leading zeros, then saves the value to a buffer. When all the values are in, the FPGA drives the chip select signal high for `ADC_TQUIET` cycles. The module outputs the saved value and a valid out signal. Every time the reading wire is changed, the voltage is read in through the ADC.

D. Data Storage

Because the sensor input and data analysis run on different clocks, we use a dual port dual clock Xilinx BRAM to manage the sensor data. The data is written to the BRAM at the rate of the ADC clock and data is read from the BRAM at the rate of the 65mhz clock. This first BRAM contains only the raw data.

IV. Computation Data Analysis

A. Threshold Input

The threshold input for the data is controlled by a seven segment display, switches, and the buttons. It is stored by multiple counting modules that increment on the rising edge of each button push. Since the data is 12 bits, the thresholds are represented by 3 hex figures in the seven segment display. The top four nibbles in the display represent the upper threshold, and the bottom four nibbles represent the lower threshold.

A switch determines whether the user controls the upper or lower threshold. The left and right buttons toggle between which nibble ("significant figure") the user controls. The user then uses the top and bottom button to change the nibble value.

The threshold values are used in noise filtering, convolution, and center of mass calculation.

B. Convolution

This module reads in the threshold module's filtered data and stores it into a three line internal buffer. This data is used to perform convolutions, or combining our array signals to highlight patterns in data. Due to the input requirements of our convolution algorithm, we are unable to perform arithmetic on the input of our outer wires. Our 3x3 convolution kernels include:

1) Identity
2) Gaussian Blur
3) Sharpen
4) Ridge Detection
5) Sobel X Edge Detection
6) Sobel Y Edge Detection

This data is then written to two duplicate BRAMs for visualization. The BRAMs are a dual port dual clock Xilinx BRAM that operates at the rate of the VGA clock on both ports. One BRAM is connected to the VGA visualization modules, while the other is connected to the UART output.

C. Center of Mass

This module takes in any pixel that passes the thresholding value after convolution. It adds the pixel to the sum and divides out the total number of pixels on the monitor.

D. Motion Tracking

This module takes in the coordinates from the Center of Mass module and tracks its movement. This module sends a signal to the VGA Display if the Center of Mass coordinates move across 3 pixel regions, correlating to switching wire and reading wire combinations, within a specified timeout. This signal highlights blue all of the pixel regions that the Center of Mass coordinates moves through. This signal stops when the Center of Mass coordinates do not move for the specified timeout amount of time.

V. Visualization

A. VGA Display

The scaling and mirror modules control the size that the sensor appears on the monitor. The scale module divides whether the `hcount` and `vcount` are within the range, then outputs the address to the BRAM for which corresponding data point value it should display.

The visualization module scales the 12 bit data in the BRAM into a RGB color scale. This data is then scaled to a combined RGB value ranging from 0-767. This combined RGB value is used to make a set of RGB values, (R, G, B) where each ranges from 0-255. This conversion is done by having the first third of the RGB value correspond to the R value, the second third correspond to the G value, and the final third correspond to the B value.

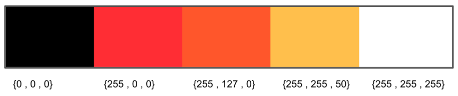

Fig. 6. Visualization Heat Map Color Gradient

A final vga_mux module controls additional displays that can be layered on top of the visualization, including:

1) Crosshair for Center Of Mass
2) Blue squares indicate motion tracking
3) Thresholding all non-relevant values to black

B. UART

The data for the sensor can also be sent through UART for storage purposes. It runs on a parameterized baud width (the default is the fastest baud width of 115.2kHz) by dividing out the reference 65 MHz clock. The UART module has an internal counter for the switching and reading wires, then pulls the relevant data from the convolution BRAM. Since UART is 8 bits maximum serial communication protocol, we found that it was best to scale the 12 bits down and store it in a shift buffer that outputs to the UART line.

The UART will output a user-set `NEWFRAME_VALUE` character indicator (default is 0) every time the internal counters have finished the maximum `SW_WIRE_CNT` and `RD_WIRE_CNT` wires. A python script on the computer can parse out this new value, and draw the array.

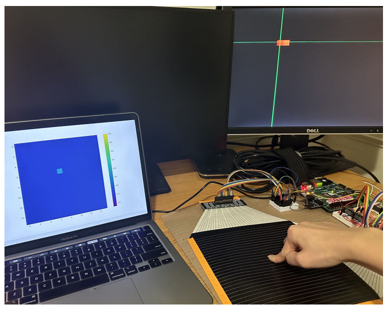

Fig. 7. Sensor Visualisation

VI. Design Evaluation

A. External Electronic Timing

By tracking the timing constraints of our electronics, we found that the limiting factor is the input into the ADC which operates at a reading input frequency of 1 MSPS.

Since the ADC needs 16 clock cycles (4 leading zeros, 12 bits of data) and a we set our `ADC_TQUIET` of to 4 clock cycles, we can operate our ADC clock at 20 MHz. This means the minimum reliable clock divider `ADC_CLK_DIVIDE` out of our reference 65 MHz clock would be 3.25. Given the maximum possible timing constraint, our entire sensor scans at:

$$\frac{65,000,000}{3.25 * (16 + 4) * 16 * 16} = 3906.25 \text{ Hz} \tag{2}$$

For some flexibility in our system and additional electronics, we decided on a default clock divider `ADC_CLK_DIVIDE` of 4. In this case, we are marginally exchanging signal integrity (as seen in figure 9 below) for speed of scanning.

On the switching side, each wire is held for about 100 µs. The time between switching to a wire and the responding switch driving it to ground was around 265 ns. The time between switching out of a wire and driving it high again was 9 µs.

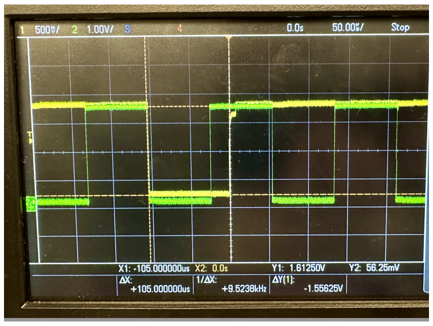

Fig. 8. Switching signal with a ADC_CLK_DIVIDE of 4. Green is FPGA signal, yellow is switch output.

On the reading side, each reading wire is held for about 4.9 µs. We measured about a 1.8 µs delay between the reading wire input into the reading mux and a stable op amp output.

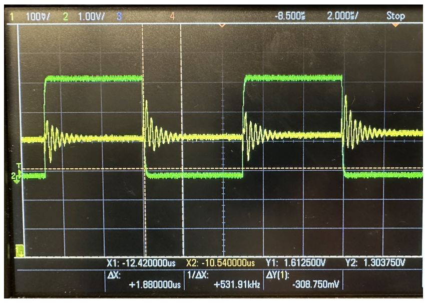

Fig. 9. Reading signal with a ADC_CLK_DIVIDE of 4. Green is FPGA signal, yellow is op amp output.

B. FPGA Timing

The majority of the latency is acquired from the limitations on the ADC reading clock. The maximum rate that we can reliably sample the entire sensor is 3906.25 Hz. However, this data is then fed into the first BRAM , which operates at a rate of 78.125 kHz. Then this data moves through the computation modules, which consist of 5 pipeline stages and feeds into the second BRAM. From the second BRAM the data is output over VGA and UART. This amount of latency is undetectable by humans and can thus be considered non-existent for the use-cases of the project.

Other than the ADC Read module, every module in the design has full and constant data throughput. The ADC output consists of quiet cycles and leading 0s, limiting the data throughput and operating rate of the entire system.

The biggest overhead in the design is the number of clocks that are all derived from the 65 MHz VGA clock. These clocks are required for driving the numerous signals sent to external electronics in the project design.

On the digital side, the path of worst delay is found within our convolution module. According to Vivado's timing report, it has a 5.609 ns slack delay with a requirement of 15.385 ns, representing the 65 MHz VGA clock it operates on. This is the fastest clock within the design and the convolution module is the most mathematically intense logic within the design.

C. FPGA Usage

With 16 switching wires and 16 reading wires, the design uses 1.5 of the available 135, 1.11 utilization percentage, block RAM tiles. There isn't a clear way to decrease the block RAM usage, as it allows for the scalability of the design to any number of wires.

With 16 switching wires and 16 reading wires, there are 10 DSPs of the available 240, 4.17 utilization percentage, in use by the design. This number could be reduced by further pipelining the convolution module, where all 10 DSPs are located, at the expense of undetectable latency.

With 16 switching wires and 16 reading wires, the design uses 1670 of the available 63400, 2.63 utilization percentage, slice LUTs. With the same wire configuration, the design uses 1675 of the total 126800, 1.32 utilization percentage, slice registers.

D. Design Scalability

Since the sensor is essentially a series of powered wires over an array in space, it is limited by physical constraints that we include in our measurements. Performing a bode plot analysis on the sensor shows that for a 3 dB dropoff point, the sensor will begin to drop off at 146.8 kHZ.

The limiting factor of the sensor resolution in this case was the hand mounted electronics. While the sensor can be manufactured up to an arbitrary amount of wires, we chose to read out of a 16x16 array. In the future, we would like to consolidate our electronics into a PCB which would allow us to expand into further exciting applications.

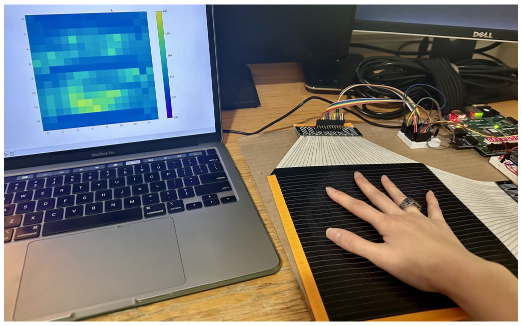

Fig. 10. Hand pressure as read by sensor on UART output

Assuming that the usage of DSPs can be minimized through pipelining at the cost of unobservable latency, the utilization percentages of the FPGA resources scale linearly with the number of wire intersections (`SW_WIRE_CNT * RD_WIRE_CNT`), and the slice LUTs are the limiting factor of the design, the maximum number of wire intersections that the FPGA could support with this design is 9,733.

$$(16 * 16) * \frac{100}{2.63} = 9733 \text{ wire intersections} \tag{3}$$

9,733 is a prime number, but, for visualization purposes, 9,734 wire intersections corresponds to a sensor consisting of 61 by 157 wires. This would require a total of 14 IO ports to switch through the wires through external MUXs. Using the same ADC and number of ADC quiet cycles, this gives a maximum sensor sample rate of 10.4 Hz.

$$\frac{65,000,000}{3.25 * (16 + 4) * 61 * 157} = 10.4 \text{ Hz} \tag{4}$$

E. Project Specifications

We were able to achieve all of our goals for the project. The design allows for the user to visualize filtered data from a touch sensor on a VGA monitor. It also allows for the user to fine tune the filtering process, track the motion on the sensor, and output the filtered and processed sensor data over a UART signal.

In the future, there are a variety of methods the design could expanded to detect extremely fast events on the sensor. The digital design was written to be fully scalable to any number of wires, which allows larger and/or more precise sensors at the cost of IO ports, sampling speed, and electronics cost. This is, however, assuming we simply linearly increase the number of electronics. It is also possible to have multiple configurations of electronic grids running on the same sensor and extrapolate pressure points via superposition. The electronics have been redesigned to be more accessible for future development.

VII. Retrospective

1) Electronics are hard and, even though we read up about the system, there are a lot of intricate details that come along with analog measurements. Despite the documentation we received from previous work on a similar project, the progress was slow and the documented design required many improvements to properly function. Even so, we did what we could about the system and are excited to turn it into a PCB layout in the future.
2) Managing various clock signals and timing rates is extremely difficult and tedious. In the future, it would be wise to minimize the number of clock signals in the design.
3) Constant communication, realistic goal setting, and proper time management is what allowed us to finish our project in a timely manner. We ran into many pitfalls, but we were able to adeptly maneuver tasks and priorities between the two of us in order to successfully complete the project.

A. Code Repository

http://github.mit.edu/akwb/tactile-sensing

B. Team Member Contribution

We worked in a very integrated manner, so nearly each module is split evenly between our work. Austin built the convolution and motion tracking, and wrote the FPGA evaluations. Tiffany worked on the electronics, ADC reader, and UART, and wrote the electronic descriptions. Everything else is made up of both our contributions.

We would like to thank our project supervisor, Joe Steinmeyer. We would additionally like to thank the CDGF Lab in CSAIL and mentor Mike Foshley for providing the sensor technology and references that form the base of this project.

References

[1] Y. Luo et al., "Intelligent Carpet: Inferring 3D Human Pose from Tactile Signals," 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2021, pp. 11250-11260, doi: 10.1109/CVPR46437.2021.01110.

[2] Sundaram, S., Kellnhofer, P., Li, Y. et al. "Learning the signatures of the human grasp using a scalable tactile glove" Nature 569, 698–702 (2019). https://doi.org/10.1038/s41586-019-1234-z

[3] Tommaso D'Alessio "Measurement errors in the scanning of piezoresistive sensors arrays" Sensors and Actuators A: Physical, Volume 72, Issue 1, 1999, Pages 71-76, ISSN 0924-4247, https://doi.org/10.1016/S0924-4247(98)00204-0.