SENIOR DESIGN PROJECT I

Secure and Reliable Hardware

Instructor: Dr. A. Abul-Fadl

Faculty Advisor: Dr. John C. Kelly

Spring 2022

Submitted by: 5/06/2022

Group #: 3
Jade Wilds
Keanna Wright
Claudia Morings
Joseph Miller

Table of Contents

List of Figures

List of Tables

Abstract

A significant problem that the cybersecurity industry faces is the vulnerability of internal memory chips. This vulnerability allows hackers to gain control of a device by exploiting this flaw in the DRAM. To develop a high-performing and reliable DRAM-based main memory, we must analyze the current aspects of DRAM memory chips. This project focuses on pinpointing errors, exercising the DRAM, and capturing performance data using an FPGA-based testing platform. This FPGA-based testing platform will consist of a computer, a heater, and a DRAM module. This design will consist of seven proposed project deliverables described in a timeline. In September and October, the main task will be to construct a Literature Review to gain insight into the manufacture of DRAM. By the end of November, we expect to have finished the development of the system's architecture and our preliminary prototype's design. We should have acquired all parts and equipment for design construction by the end of December. We should complete the hardware Setup by January. This aspect leads into the experiment's run and Implementation phase, which is expected to be completed by March. The data will then be analyzed and recorded from March to April. Lastly, a final report will be compiled to document the results by the end of May. The total estimated cost for this project is $500.00.

Introduction

Entering our Senior year, we have accumulated a plethora of skills over our academic career. We consist of three computer engineers and one electrical engineer. Choosing a project that played to our strengths was the goal and exposing ourselves to new ideas and challenges. This began with choosing an advisor: Dr. John C. Kelly. He presented a senior project that intrigued us all. The project was to round out our three years of engineering education to have these skills working together. The main topic of our project is DRAM, and it is used to build the main memory systems of modern computers. In the past few years, this technology has been on the rise in the past years, but with more advancements, its reliability and performance have been critically impacted. We are implementing research on SoftMC Practical DRAM characterization using an FPGA-Based Infrastructure to try and combat this issue. Dr. Kelly broke down the main parts that will be completed throughout these senior projects classes. The sections are the System Configuration and Control Computer, Rowhammer DRAM Controller (FPGA), Digitally Controlled SDRAM, and Heater and Temperature Controller. The timeline of our project has been adjusted to complete parts of this project in a guided layout.

Literary Review

In today's society fraud is everywhere, people are always trying to "get ahead". Criminals do this through spam calls, lottery and sweepstakes scams, banking fraud, and many other ways, says USA.gov. One of the main ways people choose to "get ahead" is by hacking electronic devices. When it comes to most electronic devices, FPGA Boards specifically, the hacker attacks the memory components of the device and has access to all your records, account numbers, search history, tax documents, and a variety of other things. According to Madelyn Bacon, an editor of the website TechTarget, and researchers from Carnegie Mellon University and Intel Labs, the process by which hackers use to access your memory is a process called rowhammering. Rowhammering is the repeated hammering of the rows within a memory unit to flip the bits which reverse their binary state, turning ones into zeros and zeros into ones. The sole purpose of our research and testing is to find out why and how this is happening in FPGA Boards specifically, allowing you to rest easy knowing your memory is safe.

Body

Organization and Management In this project the Project Manager & Leader are responsible for guiding the team through the process of decision making and problem-solving. The recorder is responsible for maintaining a record of the team's work, documenting important results, and recording responsibilities for different tasks. The spokesperson articulates the team's results. The Optimist & Pessimist examines why ideas will work while also making sure that opposing ideas are represented and discussed. The method used to schedule control for this project was dividing and conquering. The Project Manager & Leader is in charge of knowledge of FPGAs and which will be best to use for the project. As well as understanding the temperature controller/heater and working on implementation. Our Recorder is in charge of statistics on the project and serial communication between the PC and the FPGA. The Spokesperson must know the Rowhammer DRAM vulnerability that this project is based on. The Optimist & Pessimist of our team is to know about all aspects of the DRAM and which fits best into our design, while also leading the implementation process of incorporating it into the system.

Table.1.0 Personnel

Project Manager & Leader:	Jade Wilds
Recorder:	Claudia Morings
Spokesperson:	Joseph Miller
Optimist & Pessimist:	KeannaWright

Description of Work*:*
Up to this point, we have researched the FPGA, DRAM, Rohammer Phenomena, PC to FPGA interface, and Temperature Sensor/Heater. Due to our research on

interfacing between a PC and FPGA, we know that it is written in a high-level language, C, C#, or C++, and the different ways to interface. One way is by ethernet, which can move large amounts of data with a high transfer rate but requires a significant amount of time to set up. Another way is by USB UART, which is easier to set up and less complicated but is slower and has a lower data transfer rate. We were originally using a serial interface which is the communication between two digital systems that transmits data as a series of voltage pulses down a wire. These two digital systems include a PC and the FTDI Basic Breakout board. The goal was to produce a local echo using serial communication. After the design modification stage we decided to settle on implementing a reference design that includes a JTAG_UART which is used to communicate with an external terminal. Choosing an appropriate FPGA for our project begins with the DE10-Lite Board. With the existing capabilities on the board, we are utilizing the Analog-Digital Converter to add the Temperature Sensor and Heater to our system. We have been progressing with the research portion connected to this section so that the specifications that we want to implement will work with the overall setup. We are trying to set up an individual test for communication between the sensor and the ADC feature on the board. When it comes to the DRAM portion of this project, we decided on using the built-in SDRAM located on the Terasic DE10-Lite board. This SDRAM is a single 64MB(32Mx16) chip. It consists of a 16-bit data line, control line, and address line connected to the FPGA. We implemented a NIOS-based SOC design to use as the foundation for our Rowhammer project. This design incorporates integrated peripherals connected together through an Avalon interconnect and uses a NIOS II CPU with some reference C code that is used to access the SDRAM and format the User Interface. When learning how the Rowhammer Phenomenon works, we wanted to put it into practice on a device in order to help with the recreation of the process. In our research, we discovered a code that tests the memory of a device; the code then reports on how the device performed under the test. We chose to use a PC for this test, testing its memory to understand the Rowhammer Phenomenon better. The code must run off of a Linux-based environment, and our problem was that our PC uses a Windows environment. We had to use an external application called Cygwin that we downloaded onto the PC to run the test. Cygwin allows Windows and Apple OS users to try the test on their computers and see if the memory of their device can withstand the Rowhammer Phenomenon.

The schedule is provided with an accompanying description of key dates that supports scheduling decisions.

Table.2.0 Schedule

Design Activity	*Date complete*
Literature review	October 29
Experimental Design Process	December 01
Obtain Parts & Equipment	December 10

Hardware Setup	January 15
Implementation & Experimentation	March 11
Testing and Design Modification cycle	March 28
Data Analyzation	April 15
Final Report	April 23

Table.3.0 Budget

Budget This budget for this project was $500.00. With this 500.00 dollars, we were to buy the materials needed for the project.

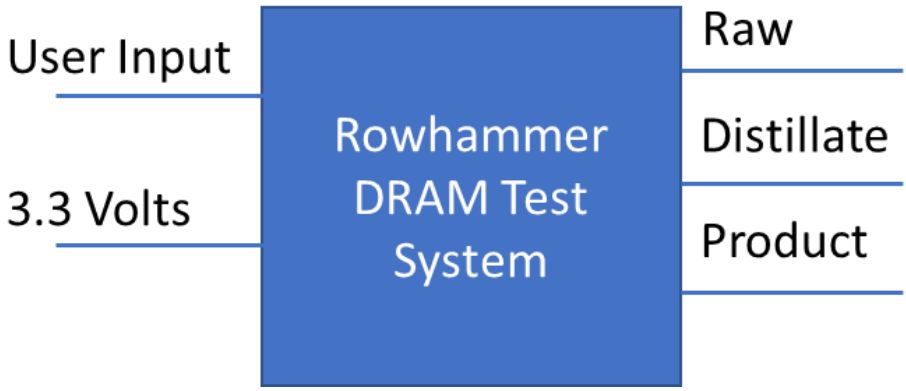
Materials	$500
FPGA (Quantity)	-$65
Heater	-$80
Relay	-$20
FTDI Basic board	-$15
Parts for Power Supply	
Beardboard	-$25
CircuitBoard Kit	-$15
Transistors	-$10
Equipment Rental	$0
Total	$300

Attachments

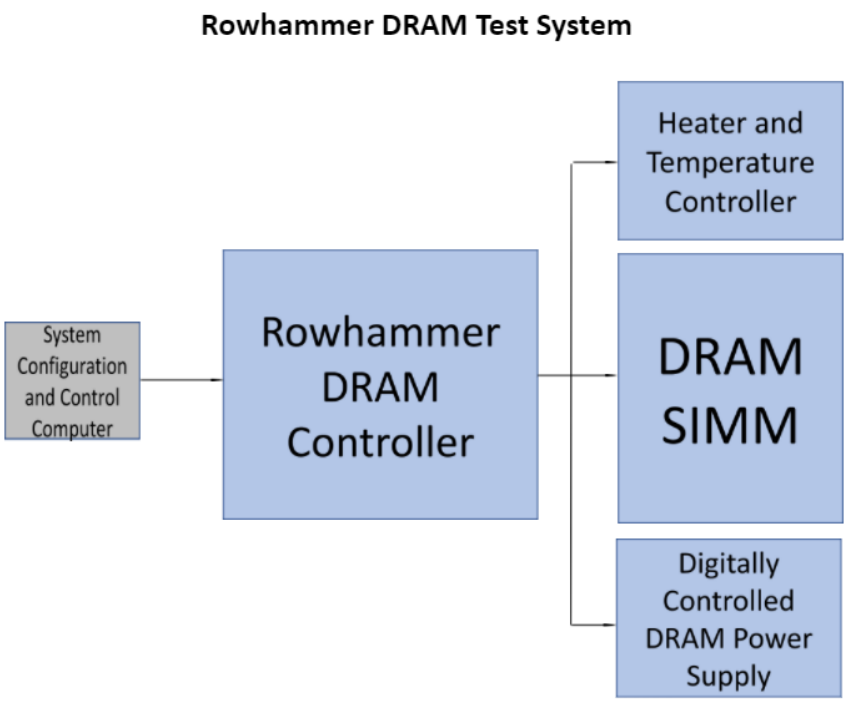
Attachment 1 - DDR2 SO-DIMM Module

http://www.farnell.com/datasheets/1766165.pdf#page=1&zoom=auto,-178,245

Analytical

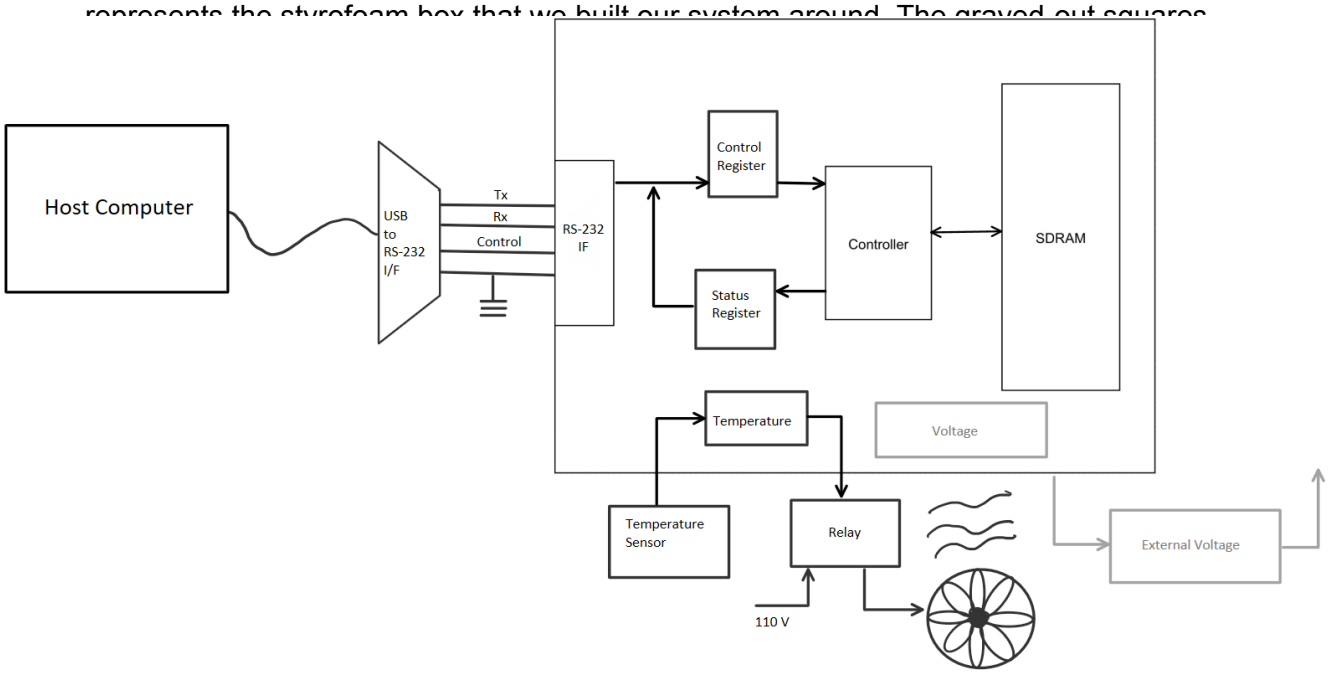


Level 0: Block Diagram Fig 1.0

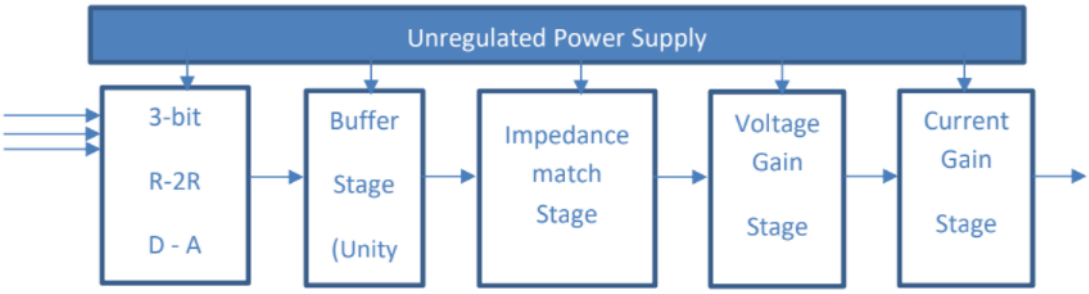


Level 1: Overall Block Diagram of DRAM test system Fig.2.0

The overall Block Diagram highlights the different sections of our system. The large box represents the styrofoam box that we built our system around. The graved out squares



Level 2: Overall Block Diagram of DRAM test system Fig 3.0



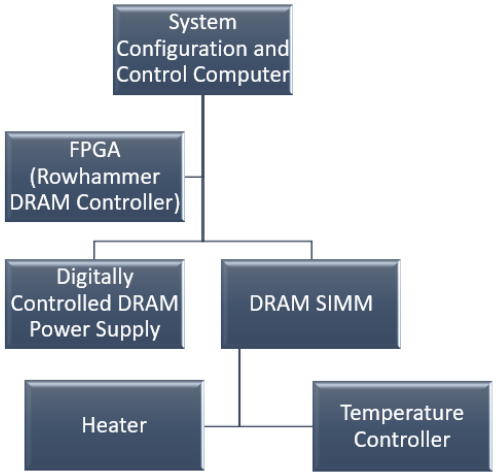
Level 2: Block Diagram of First Alternate solution of the Digitally Controlled DRAM Power Supply Fig.4.0

Level 2: Example of PC-to-FPGA diagram Fig. 5.0

Level 0: Visual of Rowhammer Test System Fig. 6.0

This is the current structure in which our project will be housed; the DE10-Lite and heater will all be in this foam box.

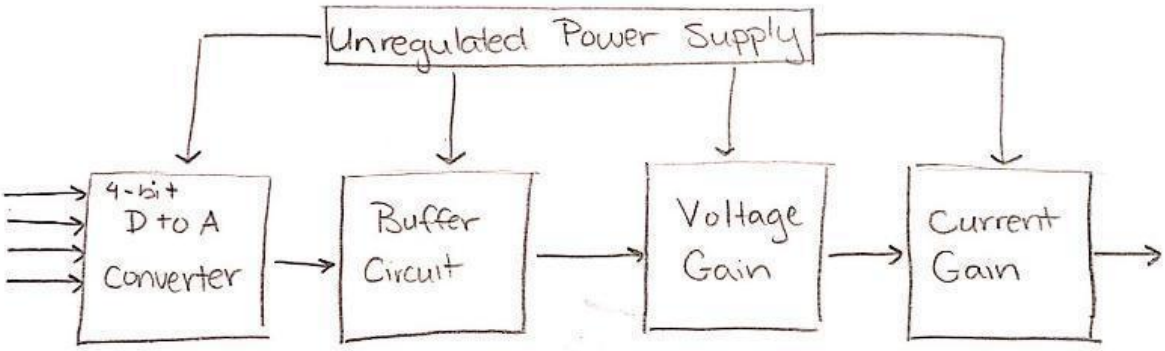
Body Flow Chart



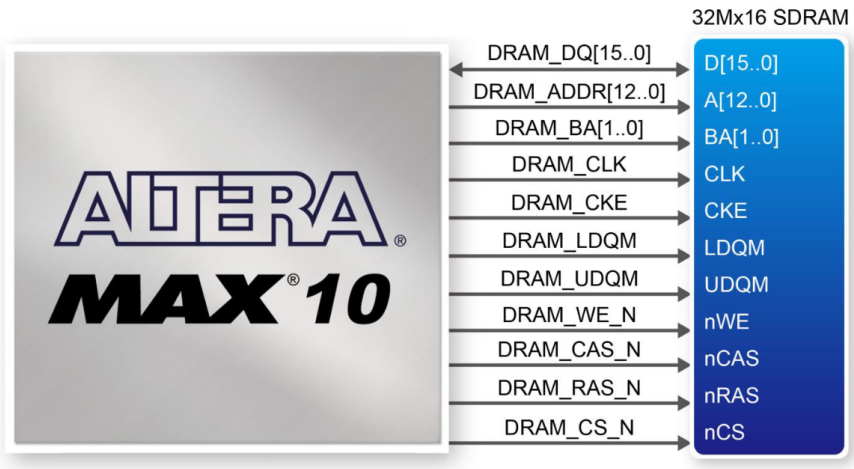
Level 2: Flowchart that describes the hierarchy of our design Fig.7.0

Block Diagram

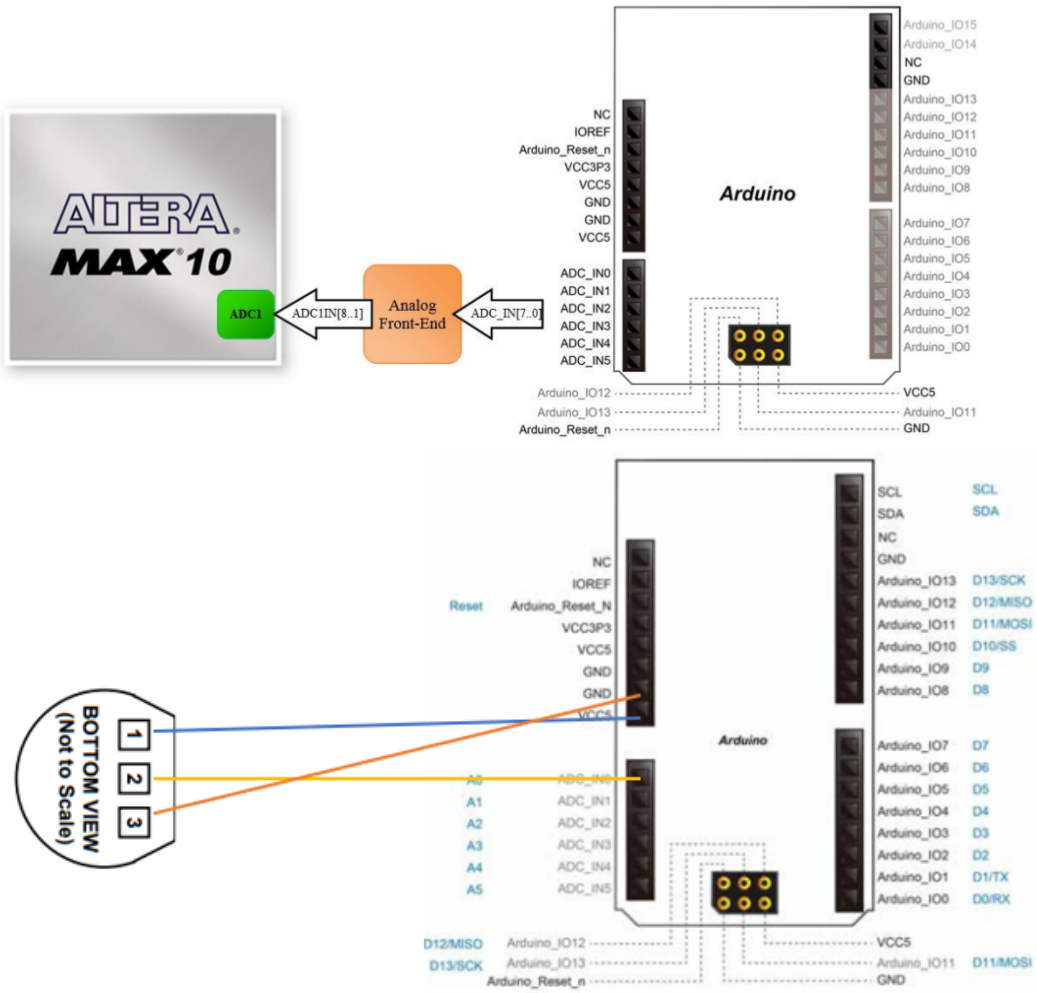
Level 2: Block Diagram of Alternate solution 2 for power supply that we ultimately built



as the first prototype of power supply Fig.8.0



Level 2: Block diagram of the SDRAM system Fig 9.0.



Level 2: Block Diagram of ADC (Temperature Sensor) Fig 10.

This project consists of five main parts. These parts are what the system configuration is responsible for controlling the system, the FPGA accountable for

controlling the SDRAM module and the temperature, the external voltage, the SDRAM module, and the heater. So far, we have begun testing the relay interface and testing the SDRAM controller. After extensive research, we have finalized our FPGA of choice for this project to be the Terasic DE10 Lite. For the serial interface, we chose the FTDI Basic Breakout board.

Design Criteria

For our criteria, we chose cost, time, and difficulty to put in the decision matrix. We picked cost because it is a factor that is always taken into account. Time was added to our list of criteria due to the time-sensitivity of our project. We learned that row hammering is detrimental to anyone with a computer or digital device, and the faster we can fix the issue, the more minor damage is caused. Lastly, we added difficulty to our criteria due to the demanding features of this project. We have five equally essential segments for the end goal. Each segment pulls from all of the knowledge that we have learned, are learning, and still have to learn, so the objective is not to make it too complex or straightforward. We do not want the system to be too simple because simplicity blocks creativity. Still, on the other hand, complexity brings confusion.

Decision Matrix

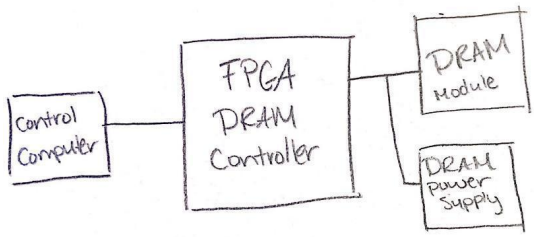
Criteria	Cost	Time	Difficulty	Geometric Mean	Weight
Cost	1	1/5	3	0.8	0.18
Time	5	1	7	3.2	0.72
Difficulty	1/3	1/7	1	0.4	0.09

Alternate Solutions

1.) Rowhammer DRAM Test System 2: This FPGA-based test system would include a control computer, an FPGA to control access to the DRAM, a DRAM module, and a power supply for the DRAM module. This would differ from the chosen model by not including a heater in the design. By excluding this, we

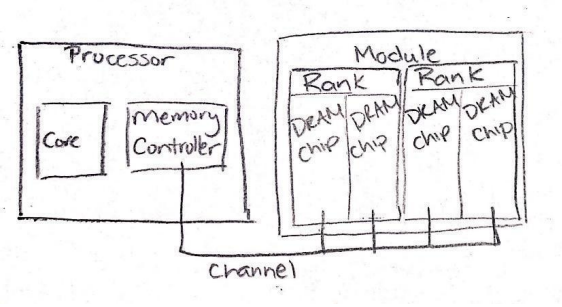
would have one less aspect for the FPGA to control; however, we would not be able to distinguish if temperature affects the results.

Fig.7.1 Block diagram of Alternate Solution 1



2.) Rowhammer DRAM Test System 3: This Rowhammer Test System would test in a cloud-based environment. It would include a core processor that controls the DRAM memory module's channels. This would differ from the chosen model because the test rowhammer attacks would be mounted in a para-virtualized setup.

Fig.7.2 Block diagram of Alternate Solution 2



3.) Rowhammer DRAM Test System 4: This DRAM Rowhammer Test System would include the Host PC, an FPGA to control the DRAM using wishbone and LiteDRAM, and a DRAM module. This alternate solution would only include the bare minimum to perform this experiment, and the FPGA would have to power the DRAM module.

Fig.7.3 Block diagram of Alternate Solution 3

Testing

Testing Overview:
So far, we have tested the power supply prototype that we built. We built this power supply on a breadboard using resistors, transistors, capacitors, and various other things. When in the lab, we tested to see if it was sufficient to supply the power we needed. After extensive testing of this power supply, we discovered that we could use an alternate external voltage source that will work more fluidly with our design. Testing has also been done for the heater; It began with creating a simple state transition to simulate the heater for the system. The state machine works in two states, one for opening the relay and the other for closing. We also developed a test for the ADC feature on the board to understand how the values for the temperature will be passed through. We were also to test our SDRAM controller. This implementation of the SDRAM controller was tested to write data to memory and read data from memory. We also did final tests on the SDRAM controller when hammering the desired address with write commands and reading the values from the neighboring address.

Testing Procedures:

UART

Receiving Operation

1) Counter values are hardcoded to receive 9600 bits/sec
2) Each bit is stored in an array
3) LED is mapped to the receive to show that it is happening

Transceiver Operation

1) Separate counter is used to send characters stored in an array at a rate of 9600 bits/sec
2) Updates counter
3) When connected to the DE10-Lite and PC a local echo is produced using the FTDI Basic Board and PuTTY

SDRAM Controller

First Implementation of SDRAM Controller:
Read Operation

1) Write desired address to the *addr* bus.
2) Request a read operation by de-asserting the *we* signal and asserting the *req* signal.
3) Wait for the *ack* LED to turn on - This shows that the read request was acknowledged and has begun.
4) Wait for the *valid* LED to turn on - This shows that the read request has been completed and the data is available on the bus.
5) Get the value off the *q* data bus.
6) Turn off the *req* signal.

Write Operation

1) Write desired address to the *addr* bus.
2) Write a value to the data bus.
3) Request a write operation by asserting the *we* and *req* signals.
4) Wait for the *ack* LED to turn on - This shows that the write request was acknowledged and has begun.
5) Turn off the *we* and *req* signals.

Final Implementation of SDRAM Controller:

1) Build C code

2) Press button 1 to run controller
3) View results for values before Hammer attack and after Hammer attack.

ADC

First Implementation of Finite State Machine: Temperature Controller

1) Compile the provided code (First Phase).
2) There is a button that will represent the temperature threshold. The button indicates that the temperature has dropped below the threshold.
3) The next step is to observe an LED that turns on after a few seconds. The LED represents the relay opening and closing.

Second Implementation of Finite State Machine: Temperature Controller

1) Compile the provided code (Second Phase).
2) This Finite State Machine operates in three phases to activate the first one. Turn all switches to 0 to begin.
3) Flipping switch 0 indicates that the temperature has fallen below the lower set temperature.
4) Observe the led turn on to show the relay opening to the heater.
5) Flip switch 0 off; this says the temperature falls between the two set points.
6) Flip switch 1; this will indicate that the temperature is warmed to the higher setpoint, and the relay can be closed.
7) Observe the led turning off.
8) Repeat to verify the state machine is flowing properly.

Observing the ADC on a Oscilloscope

1) Once the appropriate connections are established on the breadboard and FPGA.
2) Then start the ADC Toolkit to observe the values that are coming from the TMP 36.
3) Understanding the values and the math that go with the translation.
4) Try getting an ice cube in a bag to test the flux in voltage values.
5) Also, use two fingers to rub the tmp 36 to warm it up and observe those values,
6) Understand the conversions.

Conclusion

In conclusion, this team has accomplished its goal on the Senior Design project. We have successfully created the environment for Rowhammer testing. Although we were not able to obtain a bit flip, we now know how the process works, and we can implement other attacks to obtain a bit flip. Through our research and testing, we learned about the vulnerability of the DRAM Chip, and how heat can be a huge factor in relation to bit flipping, adding heat increases the probability of bit flipping. In the future, we can bridge our modules together into one cohesive system, as well as performing other tests, such as the double-sided attack on our device. We have conducted the research, in hope that we can pass the torch off to others and they take what we have learned, and run with it.

Recommendations

During this process, we have gotten the opportunity to collaborate with Intel Engineers Gary Wallichs and Larry Landis. They have been a great help in testing and pinpointing flaws in our designs. For the next steps of our project, we

would pass off our project to the next group of engineers. We set the groundwork for the system. They can take what we built and improve.

Appendix I

References

1. Hassan, Hasan, et al. “SoftMC: A Flexible and Practical Open-Source Infrastructure for Enabling Experimental Dram Studies.” *2017 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2017, https://doi.org/10.1109/hpca.2017.62.

2. Kim, Yoongu, et al. “Flipping Bits in Memory without Accessing Them: An Experimental Study of DRAM Disturbance Errors.” *2014 ACM/IEEE 41st International Symposium on Computer Architecture (ISCA)*, 2014, https://doi.org/10.1109/isca.2014.6853210.

3. “FPGA Programming - What Is It, How It Works and Where It Can Be Used.” *Codilime*, 30 Apr. 2021, https://codilime.com/blog/FPGA-programming-how-it-works-and-where-it-can-be-used/.

4. *Efficient PC-FPGA Communication over Gigabit Ethernet*. https://cme.h-its.org/exelixis/pubs/Exelixis-RRDR-2010-4.pdf.

5. Koinonia. “How to Interface FPGA Board to PC.” *Forum for Electronics*, Forum for Electronics, 4 Apr. 2011, https://www.edaboard.com/threads/how-to-interface-fpga-board-to-pc.101176/.

6. Venkataraman, Adithya, "Row hammer exploit in cloud environment" (2018). Creative Components. 113. https://lib.dr.iastate.edu/creativecomponents/113

Reference Code:

Temperature Controller Simulator Code

```
library ieee;
use ieee.std_logic_1164.all;

entity Heater_sim is
port( clock, reset: in std_logic;
     Cold_indic : in std_logic;
          Relay : out std_logic);
end;
```

```
architecture behave of Heater_sim is
    type stateMachine is (S0, S1);
        signal current_state, next_state : stateMachine;

            signal count: integer:=1;
            signal tmp : std_logic := '0';
            signal newclock : std_logic;

begin

            process(clock, reset)
            begin
            if (reset = '1') then
            count <= 1;
            tmp <= '0';
            elsif(clock'event and clock='1') then
            count <= count+1;
            if (count = 50000000) then
            tmp <= NOT tmp;
            count <= 1;
            end if;
            end if;
            newclock <= tmp;
            end process;


            process(newclock, reset)
            begin
                if (reset = '1') then
                        current_state <= S0;
                    elsif (newclock'event and newclock = '1') then
                        current_state <= next_state;
                    else
                        null;
                    end if;
        end process;


            process(current_state, Cold_indic)
begin

            next_state <= current_state;

            Relay <= '0';
            case current_state is
```

```
                when S0 =>
                        if Cold_indic = '0' then
                            next_state <= S1;
                        else
                            next_state <= S0;
                        end if;
                    when S1 =>
                       Relay <= '1';
                        if Cold_indic = '1' then
                            next_state <= S0;
                        else
                            next_state <= S1;
                        end if;
                    end case;
            end process;
end behave;
```

UART Code

```
-- The following program receives a UART (8N1) character on the rx pin and then
stores it in an array. It then outputs the same character
-- back on the tx pin.

library IEEE;
use IEEE.STD_Logic_1164.ALL;

-- Define Clock as input, rx as input, tx as output and LED_rx as Output
entity UART is
    Port(   clk:        in STD_LOGIC;
            rx:         in STD_LOGIC;
            tx:         out STD_LOGIC;
            LED_rx:     out STD_LOGIC);
    end UART;

architecture Behavioral of UART is

begin

-- Instructions are executed sequentially with the process
    process(clk)

    variable counter:               integer range 0 to 50000 :=0;
                                    -- Counter used for reading rx pin
    variable counter_2:         integer range 0 to 50000 :=0;
                                    -- Counter used for reading tx pin
```

```
        variable tx_flag:                       std_LOGIC :='0';
                                                        -- Flag to start transmitting
data on tx pin
        variable UART_buffer:       STD_LOGIC_VECTOR (9 DOWNTO 0)
:="0000000000";             -- Array to store incoming data

        variable data_receiving: STD_LOGIC :='0';
                                                    -- Flag to indicate incoming serial
data
        begin
                        if clk' event and clk='1' then
                                                        -- The IF condition is
executed during the rising edge of the clock


                                                if (rx='0' and counter=0) then
                                                        -- Detects Start bit of the
message
                                                        data_receiving:='1';
-- Receiving UART message on rx
-- Counter values are hard coded to receive 9600 bits/sec. Each bit is stored in
an array.

                                        elsif (counter=2604 and data_receiving='1') then
                                                        UART_buffer(0):=rx;

                                        elsif (counter=7812 and data_receiving='1') then
                                                        UART_buffer(1):=rx;

                                        elsif (counter=13020 and data_receiving='1') then
                                                        UART_buffer(2):=rx;

                                        elsif (counter=18228 and data_receiving='1') then
                                                        UART_buffer(3):=rx;

                                        elsif (counter=23436 and data_receiving='1') then
                                                        UART_buffer(4):=rx;

                                        elsif (counter=28644 and data_receiving='1') then
                                                        UART_buffer(5):=rx;

                                        elsif (counter=33852 and data_receiving='1') then
                                                        UART_buffer(6):=rx;

                                        elsif (counter=39060 and data_receiving='1') then
                                                        UART_buffer(7):=rx;
```

```
                elsif (counter=44268 and data_receiving='1') then
                                UART_buffer(8):=rx;

            elsif (rx='1' and data_receiving='1' and counter=49476) then
                                counter:=0;
                                data_receiving:='0';
                                tx_flag:='1';
                                UART_buffer(9):=rx;

                        end if;

                        if(data_receiving='1') then
                                counter:=counter+1;
                        end if;
-- Sending UART message on tx
-- A separate counter is used to send the character stored in an array at a rate of
9600 bits/sec.

                        if(tx_flag='1' and counter_2=0) then
                                tx<=UART_buffer(0);

                        elsif (tx_flag='1' and counter_2=5208)
then
                                tx<=UART_buffer(1);

                        elsif (tx_flag='1' and counter_2=10416)
then
                                tx<=UART_buffer(2);

                        elsif (tx_flag='1' and counter_2=15624)
then
                                tx<=UART_buffer(3);

                        elsif (tx_flag='1' and counter_2=20832)
then
                                tx<=UART_buffer(4);

                        elsif (tx_flag='1' and counter_2=26040)
then
                                tx<=UART_buffer(5);

                        elsif (tx_flag='1' and counter_2=31248)
then
                                tx<=UART_buffer(6);
```

```
                                        elsif (tx_flag='1' and counter_2=36456)
then
                                                tx<=UART_buffer(7);

                                        elsif (tx_flag='1' and counter_2=41664)
then
                                                tx<=UART_buffer(8);

                                        elsif (tx_flag='1' and counter_2=46872)
then
                                                tx<=UART_buffer(9);
                                                tx_flag:='0';           --
Initializes counter and flag
                                                counter_2:=0;

                                        end if;
                                        if (tx_flag='1') then
                                        counter_2:=counter_2+1;   -- Updates
counter_2
                                        end if;
                        end if;
        end process;
LED_rx<=rx;                     --Maps Rx to LED
end Behavioral;
```

SDRAM Controller Code

Implementation 1:

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
entity sdram2 is
port(
--SDRAM--
DRAM_ADDR: inout STD_LOGIC_VECTOR(12 downto 0);
DRAM_BA: in STD_LOGIC_VECTOR(1 downto 0);
DRAM_CAS_N: in STD_LOGIC;
DRAM_CKE: in STD_LOGIC;
DRAM_CLK: in STD_LOGIC;
DRAM_CS_N: in STD_LOGIC;
DRAM_DQ: inout STD_LOGIC_VECTOR(15 downto 0);
DRAM_RAS_N: in STD_LOGIC;
DRAM_WE_N: inout STD_LOGIC;
DRAM_LDQM, DRAM_UDQM: inout STD_LOGIC
);
end sdram2;
```

```
architecture behav of sdram2 is

component RWcomp is
	port
	(
		clk	: in std_logic;
		addr	: in natural range 0 to 2**ADDR_WIDTH - 1;
		data	: in std_logic_vector((DATA_WIDTH-1) downto 0);
		we		: in std_logic := '1';
		RAS	: in std_logic := '1';
		CAS	: in std_logic := '1';
		q		: out std_logic_vector((DATA_WIDTH -1) downto 0)
	);
	end component;

SIGNAL CLK143, CLK143_2: STD_LOGIC;

begin

uut1: RWcomp port map(clk => DRAM_CLK, addr => DRAM_ADDR, data
=>DRAM_DQ, we => DRAM_WE_N, RAS => DRAM_RAS_N, CAS =>
DRAM_CAS_N, q => DRAM_DQ);

end;
--------------------------------------------------------------------------------
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity RWcomp is

	generic
	(
		DATA_WIDTH : natural := 15;
		ADDR_WIDTH : natural := 13
	);

	port
	(
		clk	: in std_logic;
		addr	: in natural range 0 to 2**ADDR_WIDTH - 1;
		data	: in std_logic_vector((DATA_WIDTH-1) downto 0);
		we		: in std_logic := '1';
		RAS	: in std_logic := '1';
		CAS	: in std_logic := '1';
```

```
		q		: out std_logic_vector((DATA_WIDTH -1) downto 0)
	);

end RWcomp;

architecture behav of RWcomp is

	-- Build a 2-D array type for the DRAM
	subtype word_t is std_logic_vector((DATA_WIDTH-1) downto 0);
	type memory_t is array(2**ADDR_WIDTH-1 downto 0) of word_t;

	function init_ram
		return memory_t is
		variable tmp : memory_t := (others => (others => '0'));
	begin
		for addr_pos in 0 to 2**ADDR_WIDTH - 1 loop
			-- Initialize each address with the address itself
			tmp(addr_pos) := std_logic_vector(to_unsigned(addr_pos,
DATA_WIDTH));
		end loop;
		return tmp;
	end init_ram;

	-- Declare the DRAM signal and specify a default value.
	-- default value.
	signal ram : memory_t := init_ram;

	-- Register to hold the address
	signal addr_reg : natural range 0 to 2**ADDR_WIDTH-1;

begin

	process(clk)
	begin
	if(rising_edge(clk)) then
		if(we = '1') then
			ram(addr) <= data;
		end if;

		-- Register the address for reading
		addr_reg <= addr;
	end if;
	end process;

	q <= ram(addr_reg);
```

end behav;

Advisor Meetings:

04/20/22 5:01 - 6:10 PM EST

In this meeting, Jade told us she got the ADC to work with the PLL; Keanna had results to share with the group about the SDRAM, and Joe talked with Yohannes about meeting with Keanna. The demo was also discussed, and what times we would be available to meet.

04/06/22 4:55 - 6:20 PM EST

In this meeting, the group presented to Larry and Boateng the spring progress report. Jade needed help with the PLLs; Claudia needed support with making the host interface, and Joe spoke to Larry about the SDRAM and DE10-Lite board. Larry gave the group documents and links to refer to when they got stuck, and Boateng aided Jade with the PLLs.

03/23/22 4:54 - 6:04 PM EST

We started this meeting by talking about what we needed to get done for the project. For Jade, Dr. Kelly wanted her to be able to read the voltages in a week and get the ADC working by next Wednesday. For Keanna, it was to work on the interface with Claudia. For Joe, it was to reach out to Yohannes and meet to figure out Cygwin. For Claudia, it was to modify her FSM to correlate with what Keanna and Jade already have.

03/02/22 9:01 - 9:59 AM EST

We talked about any updates we had about last week's meeting in this meeting. Keanna revisited documentation on SDRAM and got into the datasheet, which helped explain how the SDRAM is running and how the pins work. Joe went through the document Dr. Kelly sent out about bit flipping and memory. This highlighted some vital information that Joe needed to know. Jade's still figuring out the specifics on the temperature sensor, but she did get the FSM to work. She also needed to reach out to Gary to talk about the ADC.

02/23/22 9:05 - 9:51 AM EST

In this meeting we went over updates on the code people have been working on. We also talked about making a one-page report for the meetings going forward. Jade talked about creating the FSM for the heater. Keanna talked about working on the SDRAM test running on the FPGA board. She will need to have the code written and read to memory. Joe talked about guidance for what a

row hammer attack has to do with memory. Lastly, Claudia talked about talking to registers and how to control when our machine reads/writes to memory.

02/16/22 9:04 - 9:52 AM EST

In this meeting; we went over downloading CD-ROM for the DE10-Lite, QYS tool in Quartus, and seeing if a local echo can be produced. Jade's next step was to search for DE10-Lite & ADC and find a simpler way for the ADC. Claudia's task was to schedule a time with Dr. Kelly and see if Claudia could produce the local echo. Joe needed to schedule a meeting with Yohan so he could walk through what needed to happen to replicate the phenomena.

02/09/22 8:53 - 9:30 AM EST

We went over our ongoing tasks and any issues we ran into in this meeting. Claudia got the part for serial communication with Dr. Kelly's help. She was tasked with downloading a simple program to run on the FPGA for understanding. Joe downloaded an app called sight one which helps people that do not have Linux. Jade was confused about the ADC on the board. We were all tasked with continuing to reach our two-week goals. Dr. Kelly went over the schematic for our project and drew it out as a visual aid.

02/02/22 9:04 - 9:58 AM EST

In this meeting, we went over our updates on what we worked on. Keanna created a base code for access to SDRAM. She found a VMware alternative for Quartus Primes. Jade made code for the heater and a two-state FSM. Joe could not find how to run rowhammer on his computer but will try to reach out to Yohan for help. Claudia went over information about serial communication. Towards the end, two-week goals were set. Jade's was to have the hardware, implement storage, and know what parts drive the Fahrenheit. Keanna's was to include a timing portion to code using loops. Joe's was to run rowhammer on his computer. Claudia's goal was to create a UART code on the FPGA and implement it to turn it into an echo machine.

12/07/21 2:02 - 2:49 PM EST

We discussed the parts list form in this meeting. We went through our block diagram and picked out the needed components: a temperature sensor, relay, simple transistor, and ceramic heater. Still required to build, test, and integrate functional blocks for power supply. During this time, we were to continue investigating how to communicate from an FPGA to the outside world and on ADCs. We created a task list for our members. Jade was tasked with the FPGA and power supply. This also included the temperature controller. Keanna was charged with the DRAM module and reading and writing to that memory. Claudia was tasked with stats and the FPGA interface, specifically programs that can talk over USB. Joe was tasked with running row hammer on his PC to see how it works.

11/30/21 2:02 - 2:21 PM EST

In this meeting, we went over the oral presentation and what we could improve as a group. We did a recap of what our senior design project was and how to explain it in a minute or less to someone. We went over what due dates were coming up and how we planned to accomplish them. Dr. Kelly talked about software programs that can exercise rowhammer phenomena on our PCs which run on a website. Questions were asked to put our thoughts into perspective such as: “Do you think all of this will fit inside our FPGA?”. In continuation, we looked into the weaknesses of our written report and went over what needs to be fixed. In the end, we talked about our next meeting and some fixes that needed to be implemented into our report.

11/09/21 2:00 - 2:52 PM EST

In this meeting we discussed our extensive research on our individual tasks. Joe dove deeper into his research on rowhammer; Keanna talked about DRAM; Jade discussed her FPGA research and I, Claudia, talked about the statistics for the project. As for the row hammer, we found that it becomes more difficult to happen the smaller the DRAM chip is. As for the DRAM, we talked about ECC, which is error-correcting code. On the FPGA side of things, we went into detail about what the De-10 lite board does and what kind of features it has. Statistics for the project include the probability of a bit flip occurring and confidence level. For future planning, we looked into starting to implement designs, writing code that implements the FSM, and hammering the SDRAM on the De-10 lite.

11/02/21 2:00 - 3:09 PM EST

In this meeting, we discussed the oral presentation and PowerPoint that were coming up as well as wrapping up any final touches on the project overall. We went over what needed to be highlighted and the “coolest” parts of our design. We then went into what the presentation would look like and contain. After we went into the architecture of our project, which contained the thermostat, row hammering, inputs/outputs, what power is needed, and what our system will include. Finally, we planned for the following week.

10/26/21 2:03 - 2:13 PM EST

In this meeting, we revamped a couple of factors for the mini-project and were in progress on the report. We needed to redo some of the calculations to get better values for the capacitors and resistors for the current amplification stage. The due date for our mini-project was coming up, so we sent a “progress report”/draft to Dr. Kelly to get any feedback he might have had. We then planned for the following week which was to design what was inside our block diagram and look at architecture and implementation.

10/19/21 2:00 - 2:33 PM EST

In this meeting, we talked about the three stages of the mini-project; output and input impedance and power dissipation. We spoke about simulating

the resistors with a simulated input for the power supply in PSpice as well as what we need to model. The four things we needed to model were: resistors w/o an active buffer, a buffer, an impedance matching circuit in the middle, and a sense resistor in the output. Historicise is the difference in two-turn off turn points, and it works with fold back. We also talked about the next steps in our project which were: meeting up to work on the stages we talked about in PSpice and getting the simulations running to test it in the software. Lastly, set a goal to finish the PSpice models by that following Wednesday.

10/05/21 2:01 - 2:55 PM EST

In this meeting, we talked about DDR2 SO-DIMM, EEPROM, DRAM, D/A converter, and what FPGA board we will be using. As for the board we chose the DE10 Lite board due to the familiarity that we have with it. We discussed the list of pin identifications in the DDR2 SO-DIMM as well as what the EEPROM stores and what the serial EEPROM does for the DDR2 SO-DIMM.

09/28/21 2:02 - 2:58 PM EST

In this meeting, we talked about what FPGA boards do and how they operate, the power supply, DRAM, and interfacing from a PC to an FPGA. As we know, FPGAs have a logic module and come in different sizes based on what is needed. Microcontrollers can't negate the time, but an FPGA can, which is why we are using them. They are universal hardware, and they can be programmed with two different languages, VHDL and Verilog. For the power supply, we talked about the high-level view, bandwidth, voltage gain, and the block diagram. For the DRAM, we talked about storing bits into rows and columns, SIM (single in-line memory module), and how external memory can be connected directly to an FPGA.

09/21/21 2:00 - 2:56 PM EST

In this meeting, we talked about how our senior design project will flow and the process altogether. Our PC will talk to the FPGA and access the rows. Our hardware will have to generate all of the signals that exercise the DRAM module. We wanted to have it up and running by January which should contain the interface between the PC and FPGA board; a heater that will be set to a point; a control power supply; row hammering, and an analog component and digital component. We tried to select the programming language we would be using for our project and put together a schematic/circuit to show how we would be putting our design together.

Group Meetings:

03/30/22 8:50 - 11:00 PM EST

In this meeting, our group met to update our progress report and PowerPoint slides for the class. We spent some time making changes to the conclusion, testing, and other areas that needed to be updated.

03/19/22 1:00 - 4:30 PM EST

In this group meeting, Jade and Claudia met with Dr. Kelly to put together the house for their device. This meeting was very hands-on and included taking measurements of the foam pieces that were provided and cutting them to the desired size. After that, we taped the pieces together to see how the final product would look.

02/15/22 7:40 - 11:40 PM EST

In this group meeting, we went over what needs to be fixed for the first progress report of this semester. Our team members went over their parts and made the necessary changes.

11/18/21 2:30 - 6:10 PM EST

In this meeting, we went over how to improve the progress report. In enhancing our report, alternative options were discussed, and our block diagram needed more information. After we fixed those items, a decision matrix was created and added to the report.

11/04/21 2:30 - 5:17 PM EST

In this meeting we were working on our progress report.

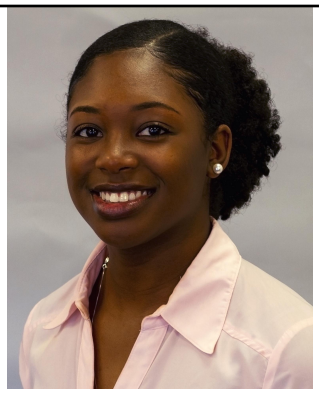
09/30/21 2:36 - 2:57 PM EST

In this meeting, we added to the abstract, revised the mini project, and gave ourselves a group name and project title. We discussed lab access so we could start doing the electrical work that was needed and our leader sent the updated abstract/proposal.

09/23/21 2:33 - 3:31 PM EST

In this group meeting, we made changes to the abstract to fit what Dr. Abul-Fadl wanted. We emailed Dr. Kelly and asked him what budget we would need for our project. We also put a calendar together to see how we would be moving through our process.

Headshot	Name	Role/Responsibility
	Jade Wilds	Project Manager & Leader

		
	Claudia Morings	Recorder
	Joseph Miller	Spokesperson
	Keanna Wright	Optimist & Pessimist