EE/CprE/SE 492 WEEKLY REPORT 06

10/27/2020 – 11/09/2020

Group number: 08

Project title: High Resolution Digitally Trimmable Resistor

Client &/Advisor: Prof. Randy Geiger

Team Members/Role: Clark Reimers - Test Engineer, Pierce Nablo - Design Engineer, Alek Benson - Information Manager, Oluwatosin Oyenekan - Meeting Lead

❖ Weekly Summary

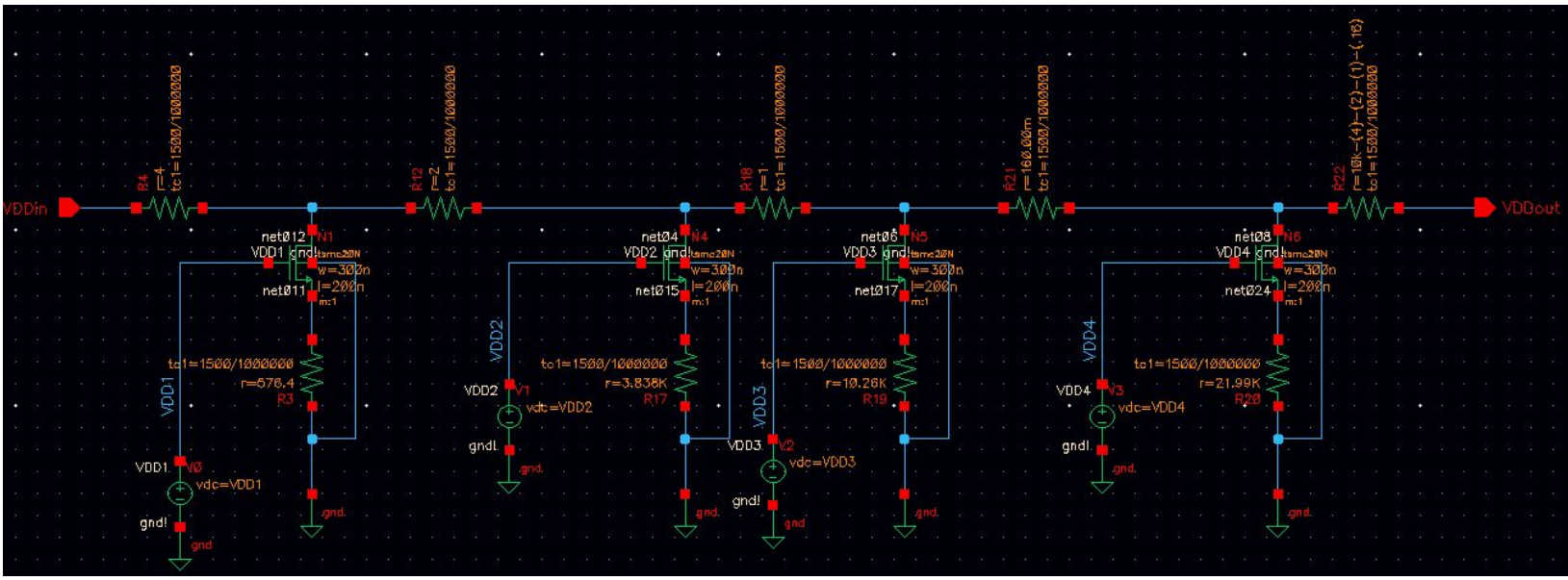
During the past 2 weeks our group has been working mainly on getting a binary weighted trim format for our resistor structure designs. From our advisor's advice we went ahead and made MATLAB code to simulate the 2-bit and 4-bit ladder structure. We have found that testing the circuit in MATLAB is a lot quicker than what we have been doing before which was to simulate the circuit in virtuoso then save the results and export it to excel. In addition we fine tooned the performance of the voltage divider structure to ensure the step sizing between bit combinations were evenly distributed.

❖ Past week accomplishments

Clark Reimers:

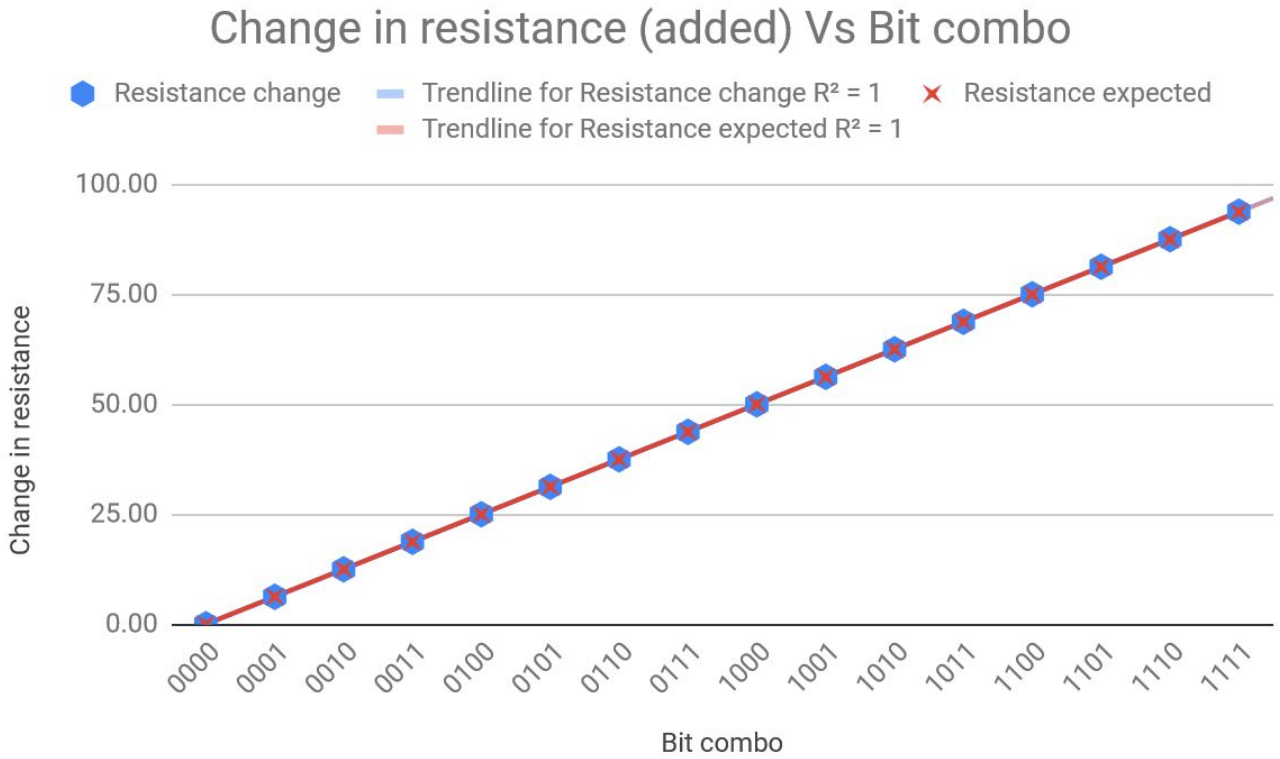
- Expanded voltage divider to 4 bits
    - Achieved binary weighting
    - Achieved low TCV
    - Achieved uniform voltage output adjustments with bit combinations
    - Achieved low area design
- Made comparisons between the ladder structure and the voltage divider structure
    - Voltage divider circuit is a good alternative to the ladder structure based on our tests.
- Worked on our poster
- Worked on final report

Voltage Divider (4 bit):



Testing

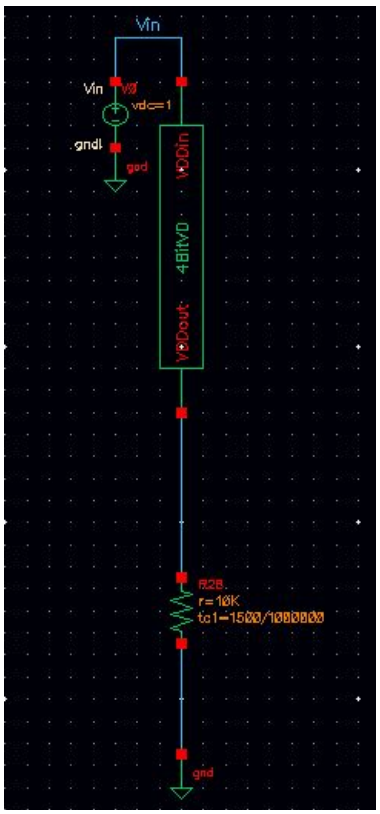
This graph shows that the voltage divider structure is correctly binary weighted.



TCV values for each bit combo:

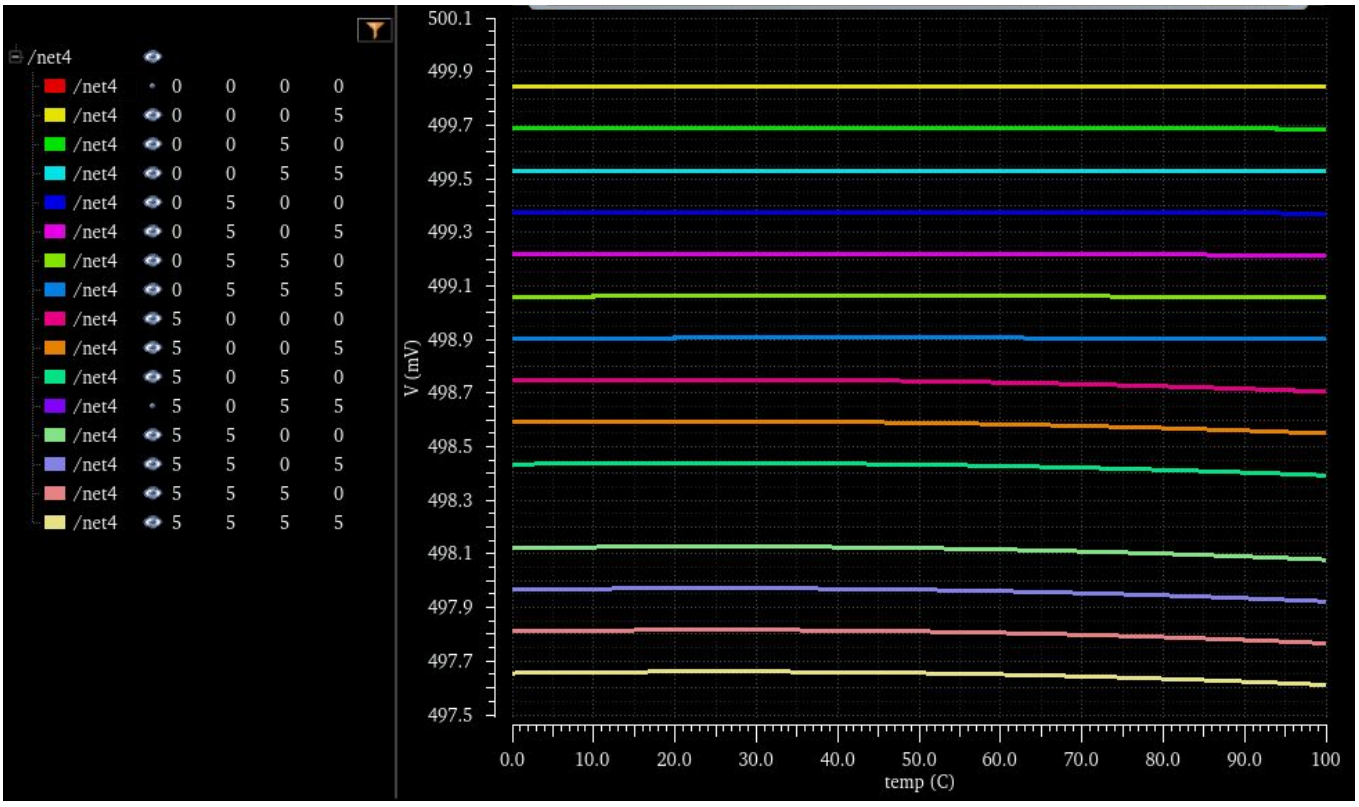
Bit combos	TCV
0000	-0.000002776
0001	0.005190602
0010	0.019135387
0011	0.024775990
0100	0.062854667
0101	0.073474643
0110	0.081841379
0111	0.092922495
1000	-0.162639859
1001	-0.157448289
1010	-0.143553029
1011	-0.137916192
1100	-0.100015642
1101	-0.089324100
1110	-0.081079560
1111	-0.069921784

Voltage output testbench:



Voltage output of the testbench (measured between VDDout and R28):

Uniform spacing as expected is observed with only minor drops in voltage of a 100 C range. This test shows that the voltage divider circuit is a good alternative to the ladder structure.



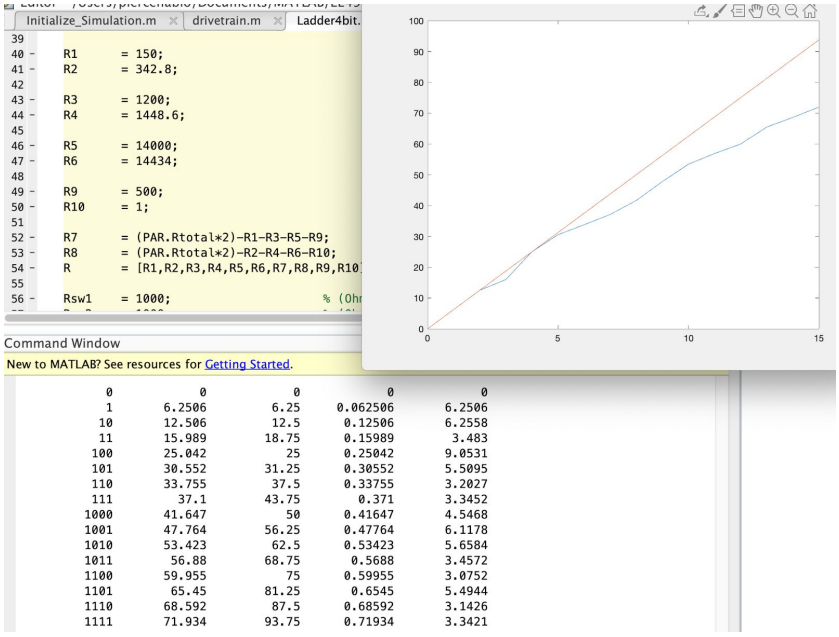
Pierce Nablo:

- I completed a MATLAB script for the 4 bit ladder structure simulation. During testing we found that it is very close to what virtuoso will calculate. Some sample code is included below with an image of a test result from the given resistor input.
- Worked on Poster
- Worked on Final design document

```
%% Initialize
%-------------------------------------------------------------------
%-------------------------------------------------------------------

PAR.Vdd    = 1;      % (volt) Input voltage
PAR.Rtotal  = 10000;   % (Ohms) Base Value of the trimmable resistor
PAR.Rload   = 10000;   % (Ohms) Resistor for testing in voltage divider


R1    = 150;                    % (Ohms) Resistor to Calibrate
R2    = 342.8;                  % (Ohms) Resistor to Calibrate

R3    = 1200;                   % (Ohms) Resistor to Calibrate
R4    = 1448.6;                 % (Ohms) Resistor to Calibrate

R5    = 14000;                  % (Ohms) Resistor to Calibrate
R6    = 14434;

R9    = 500;                    % (Ohms) Resistor to Calibrate but done
automaticaly
R10    = 10;                    % (Ohms) Resistor to Calibrate

R7    = (PAR.Rtotal*2)-R1-R3-R5-R9;              % (Ohms) Resistor to
Calibrate
R8    = (PAR.Rtotal*2)-R2-R4-R6-R10;             % (Ohms) Resistor to
Calibrate
R     = [R1,R2,R3,R4,R5,R6,R7,R8,R9,R10];    % (Ohms) Array to input into
LadderDyn function

Rsw1   = 900;          % (Ohms) Rung Resistor and/or Switch Resistance
Rsw2   = 800;          % (Ohms) Rung Resistor and/or Switch Resistance
Rsw3   = 700;          % (Ohms) Rung Resistor and/or Switch Resistance
Rsw4   = 600;          % (Ohms) Rung Resistor and/or Switch Resistance
Rsw    = [Rsw1,Rsw2,Rsw3,Rsw4];   % (Ohms) Array to input into LadderDyn
function

%% Run
%-------------------------------------------------------------------
%-------------------------------------------------------------------

% Calculate Vout, Iout given difference bit combinations
LadderData = LadderDyn(R,Rsw,PAR);

% Calculate Trim given Resistance of ladder structure
TrimPerf = LadderPerformance(LadderData,PAR);

%Display TrimPerf as double
sprintf('_____Bit_________Trim_________Ideal______Trim perc_____Trim_chg_')
format shortG
double(TrimPerf)

x = 0:1:15;
plot(x,TrimPerf(:,2))
hold on
plot(x,TrimPerf(:,3))
hold off
xlabel('Bit Combination');
ylabel('Ohms Trim');



%% Ladder Function
%-------------------------------------------------------------------
%-------------------------------------------------------------------
function [Y] = LadderDyn(R,Rsw,PAR)
```

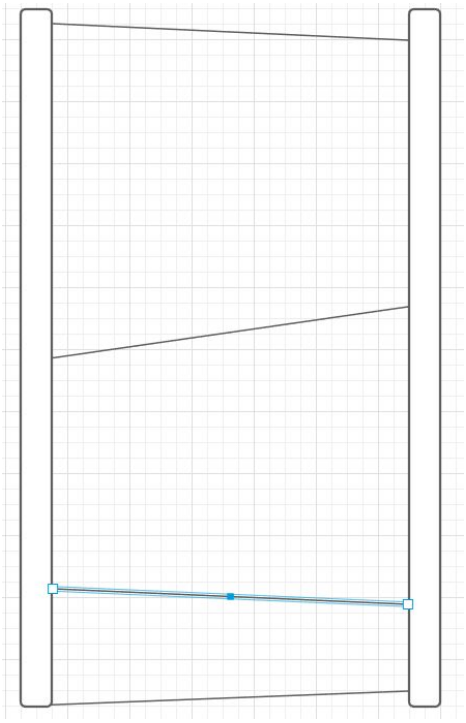
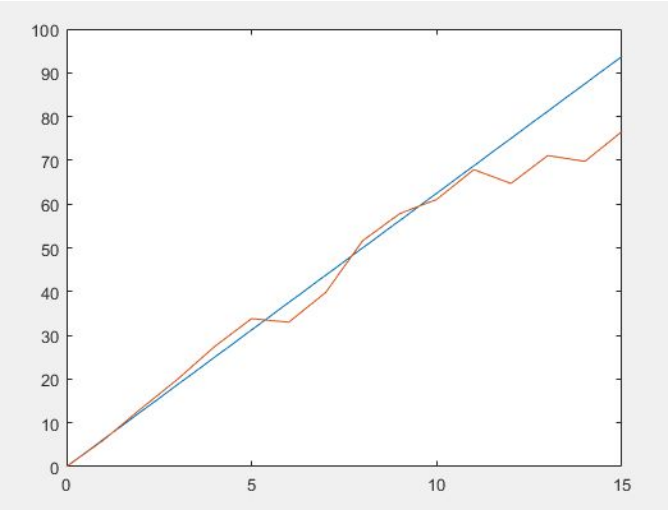
```
%% SW1=OFF;SW2=OFF;SW3=OFF;SW4=OFF;        Vout_0000 ; Iout_0000
%-------------------------------------------------------------------
%-------------------------------------------------------------------
Rx_0000 = R(1) + R(3) + R(5) + R(7) + R(9);    % Left leg
Ry_0000 = R(2) + R(4) + R(6) + R(8) + R(10);   % Right leg

% Left & Right leg in parallel
Req_0000 = (1/Rx_0000 + 1/Ry_0000)^(-1);

% Outputs for Y
VVout_0000 = (PAR.Vdd*PAR.Rload)/(PAR.Rload+Req_0000);
IIout_0000 = PAR.Vdd/(PAR.Rload+Req_0000);
Req_0000   = (PAR.Vdd - VVout_0000)/IIout_0000;
%-------------------------------------------------------------------
%-------------------------------------------------------------------
%% SW1=ON;SW2=OFF;SW3=OFF;SW4=OFF;         Vout_0001 ; Iout_0001
%-------------------------------------------------------------------
%-------------------------------------------------------------------
% Combine series resistors where able
Rx_0001 = R(3) + R(5) + R(7) + R(9);    % Left leg
Ry_0001 = R(4) + R(6) + R(8) + R(10);   % Right leg

% Node A
% Ir1 == Isw1 + Irx;
% Node B
% Ir2 + Isw1 == Iry;
% Node Vout
% Irx + Iry == Irl;

% System of equations
syms Va_0001 Vb_0001 Vout_0001
eqn1_0001 = (PAR.Vdd-Va_0001)/(R(1))    == (Va_0001-Vb_0001)/(Rsw(1))    +
(Va_0001-Vout_0001)/(Rx_0001);
eqn2_0001 = (PAR.Vdd-Vb_0001)/(R(2))    + (Va_0001-Vb_0001)/(Rsw(1))    ==
(Vb_0001-Vout_0001)/(Ry_0001);
eqn3_0001 = (Va_0001-Vout_0001)/(Rx_0001) +
(Vb_0001-Vout_0001)/(Ry_0001) == (Vout_0001)/(PAR.Rload);

% Solve system of equations
sol_0001 = solve([eqn1_0001,eqn2_0001,eqn3_0001],
[Va_0001,Vb_0001,Vout_0001]);

% Outputs for Y
VVout_0001 = sol_0001.Vout_0001;
IIout_0001 = VVout_0001/PAR.Rload;
Req_0001   = (PAR.Vdd - VVout_0001)/IIout_0001;
%-------------------------------------------------------------------
%-------------------------------------------------------------------
%% SW1=OFF;SW2=OFF;SW3=ON;SW4=OFF;         Vout_1111 ; Iout_1111
%-------------------------------------------------------------------
%-------------------------------------------------------------------

% Node A
% Ir1 == Isw1 + Ir3;
% Node B
% Ir2 + Isw1 == Ir4;
% Node C
% Ir3 == Isw2 + Ir5;
% Node D
% Ir4 + Isw2 == Ir6;
% Node E
% Ir5 == Isw3 + Ir7;
% Node F
% Ir6 + Isw3 == Ir8;
```

```
% Node G
% Ir7 == Isw4 + Ir9;
% Node H
% Ir8 + Isw4 == Ir10;
% Node Vout
% Ir9 + Ir10 == Irl;

% System of equations
syms Va_1111 Vb_1111 Vc_1111 Vd_1111 Ve_1111 Vf_1111 Vg_1111 Vh_1111
Vout_1111
eqn1_1111 = (PAR.Vdd-Va_1111)/(R(1)) == (Va_1111-Vb_1111)/(Rsw(1)) +
(Va_1111-Vc_1111)/(R(3));
eqn2_1111 = (PAR.Vdd-Vb_1111)/(R(2)) + (Va_1111-Vb_1111)/(Rsw(1)) ==
(Vb_1111-Vd_1111)/(R(4));
eqn3_1111 = (Va_1111-Vc_1111)/(R(3)) == (Vc_1111-Vd_1111)/(Rsw(2)) +
(Vc_1111-Ve_1111)/(R(5));
eqn4_1111 = (Vb_1111-Vd_1111)/(R(4)) + (Vc_1111-Vd_1111)/(Rsw(2)) ==
(Vd_1111-Vf_1111)/(R(6));
eqn5_1111 = (Vc_1111-Ve_1111)/(R(5)) == (Ve_1111-Vf_1111)/(Rsw(3)) +
(Ve_1111-Vg_1111)/(R(7));
eqn6_1111 = (Vd_1111-Vf_1111)/(R(6)) + (Ve_1111-Vf_1111)/(Rsw(3)) ==
(Vf_1111-Vh_1111)/(R(8));
eqn7_1111 = (Ve_1111-Vg_1111)/(R(7)) == (Vg_1111-Vh_1111)/(Rsw(4)) +
(Vg_1111-Vout_1111)/(R(9));
eqn8_1111 = (Vf_1111-Vh_1111)/(R(8)) + (Vg_1111-Vh_1111)/(Rsw(4)) ==
(Vh_1111-Vout_1111)/(R(10));
eqn9_1111 = (Vg_1111-Vout_1111)/(R(9)) + (Vh_1111-Vout_1111)/(R(10)) ==
(Vout_1111)/(PAR.Rload);

% Solve system of equations
sol_1111 =
solve([eqn1_1111,eqn2_1111,eqn3_1111,eqn4_1111,eqn5_1111,eqn6_1111,eqn
7_1111,eqn8_1111,eqn9_1111],
[Va_1111,Vb_1111,Vc_1111,Vd_1111,Ve_1111,Vf_1111,Vg_1111,Vh_1111,Vout_
1111]);

% Outputs for Y
VVout_1111 = sol_1111.Vout_1111;
IIout_1111 = VVout_1111/PAR.Rload;
Req_1111  = (PAR.Vdd - VVout_1111)/IIout_1111;
%----------------------------------------------------------------
%----------------------------------------------------------------
%% Output Data

%Y=  Vout  ,  Iout ,  Req
Y = [VVout_0000,IIout_0000,Req_0000;
   VVout_0001,IIout_0001,Req_0001;
   VVout_0010,IIout_0010,Req_0010;
   VVout_0011,IIout_0011,Req_0011;
   VVout_0100,IIout_0100,Req_0100;
   VVout_0101,IIout_0101,Req_0101;
   VVout_0110,IIout_0110,Req_0110;
   VVout_0111,IIout_0111,Req_0111;
   VVout_1000,IIout_1000,Req_1000;
   VVout_1001,IIout_1001,Req_1001;
   VVout_1010,IIout_1010,Req_1010;
   VVout_1011,IIout_1011,Req_1011;
   VVout_1100,IIout_1100,Req_1100;
   VVout_1101,IIout_1101,Req_1101;
   VVout_1110,IIout_1110,Req_1110;
   VVout_1111,IIout_1111,Req_1111];

end
```



0	0	0	0	0
1	6.2506	6.25	0.062506	6.2506
10	12.506	12.5	0.12506	6.2558
11	15.989	18.75	0.15989	3.483
100	25.042	25	0.25042	9.0531
101	30.552	31.25	0.30552	5.5095
110	33.755	37.5	0.33755	3.2027
111	37.1	43.75	0.371	3.3452
1000	41.647	50	0.41647	4.5468
1001	47.764	56.25	0.47764	6.1178
1010	53.423	62.5	0.53423	5.6584
1011	56.88	68.75	0.5688	3.4572
1100	59.955	75	0.59955	3.0752
1101	65.45	81.25	0.6545	5.4944
1110	68.592	87.5	0.68592	3.1426
1111	71.934	93.75	0.71934	3.3421

Alek Benson:

- Worked on the Poster for the project and got it almost all the way done.
- Spent time with Pierce getting the 2-bit MATLAB code working
- Did some code proofreading and fixed some of the 4-bit MATLAB code
- Spent time automating the 2 and 4 bit ladder structures to randomly test different configurations with error values, and exported csv data
- Spent time testing different configurations for the ladder structure to understand the trends in the trim levels for different bit combinations
- Tried to optimize the trim levels for the 4-bit ladder structure
- 


Oluwatosin Oyenekan:

- Assisted Pierce on getting the matlab code for the 4-bit ladder structure
- Worked on the poster
- Worked on the final design document

❖ **<u>Pending issues</u>**

Clark Reimers:

- No pending issues.

Pierce Nablo:

- No issues at the moment

Alek Benson:

- No Issues
-

Oluwatosin Oyenekan:

- No Issues

❖ **Individual contributions**

Name	Hours 10/27 - 11/02	Hours 11/03 - 11/09	Hours cumulative
Clark Reimers	9	8	87
Pierce Nablo	25	5	95
Alek Benson	8	8	85
Oluwatosin Oyenekan	7	6	72

❖ **Plans for the upcoming week**

Clark Reimers: Continue work on documentation and prepping for final presentation. Work on finishing final assignments and finalizing our research.

Alek Benson: The plan is to continue to try optimizing the binary weighting for the ladder structure. We are planning on spending most of the rest of the semester working on getting the data documentation ready for presentation, and working on the final paper and presentation.

Oluwatosin Oyenekan: My Plan for this coming week is to finish up working on the final design document and round up our research

Pierce Nablo: I will be shifting my focus away from the researching side and I will start to work on the design document assignment.

❖ **Summary of weekly advisor meeting**

During our advisor meetings for the past 2 weeks, we showed Dr. Geiger the performance of the voltage divider structure which has better performance than the ladder structure. In addition we showed Dr. Geiger MATLAB code to assist in getting a calibrated trim for its respective structure. When discussing the future tasks for the project we determined that, we have completed the project with the voltage divider structure, and we can begin to finalize all the administrative paperwork assignments.