بسم الله الرحمن الرحیم

دانشکده مهندسی برق

Digital Communications Laboratory

Supervisor: Dr. Shahriar Shirvani Moghaddam

Student: Mohammad Reza Farhadi Nia

Final Project

4-Binary 3-Ternary Encoding & Decoding (Line Code)

Fall 2020 – Hormozgan Province

Shahid Rajaei Teacher Training University, Tehran

Problem Description:

A. Produce random 100-bit string with LFSR, which the feedback-polynomial is used is:

$$f(x) = 1 + x^5 + x^9$$

B. Show line encoding for above string by 4-Binary 3-Ternary in Matlab & Proteus Soft Wares.

C. Then, Also show Decoding in M&P SW.

Introduction

- 4B3T[1]

4B3T, which stands for 4 (four) Binary 3 (three) Ternary, is a line encoding scheme used for ISDN PRI interface. 4B3T represents four binary bits using three pulses.

It uses three states:

- *+ (positive pulse),*
- *0 (no pulse),*
- *− (negative pulse).*

This means we have 24 = 16 input combinations to represent, using 33 = 27 output combinations. 000 is not used to avoid long periods without a transition. 4B3T uses a ***paired disparity code*** to achieve an overall zero DC bias: six triplets are used which have no DC component (0+−, 0−+, +0−, −0+, +−0, −+0), and the remaining 20 are grouped into 10 pairs with differing disparity (e.g. ++− and −−+). When transmitting, the DC bias is tracked and a combination chosen that has a DC component of the opposite sign to the running total.

We can see ***no disparity, negative disparity and positive disparity in these tables:***[2]

3T	4B
0 0 0	Leitung ungenutzt
+ 0 −	1011
+ − 0	0010
0 + −	1110
0 − +	0001
− 0 +	0111
− + 0	0100

3

3T (pos)	3T (neg)	4B
+ + 0	0 0 −	1111
+ 0 +	0 − 0	0000
+ 0 0	0 − −	1000
0 + +	− 0 0	0101
0 + 0	− 0 −	1101
0 0 +	− − 0	0011
+ − +	− − −	1001
− + +	− − +	0110
+ + +	− + −	1100
+ + −	+ − −	1010

4

[1] https://en.wikipedia.org/wiki/4B3T
[2] My footpath :)
[3] https://de.wikipedia.org/wiki/MMS43-Code
[4] https://de.wikipedia.org/wiki/MMS43-Code

This mapping from 4 bits to three ternary states is given in a table known as Modified Monitoring State 43 (*MMS43*). A competing encoding technique, used for the ISDN basic rate interface where 4B3T is not used, is 2B1Q.

The sync sequence used is the 11-symbol Barker code, +++−−−+−−+− or its reverse, −+−−+−−−+++

Each 4-bit input group is encoded as a 3-symbol group (transmitted left to right) from the following table. Encoding requires keeping track of the accumulated DC offset, the number of + pulses minus the number of − pulses in all preceding groups. The starting value is arbitrary; here we use the values 1 through 4, although −1.5, −0.5, +0.5 and +1.5 is another possibility.

• Paired disparity code[5]

In telecommunication, a paired disparity code is a line code in which at least one of the data characters is represented by two codewords of opposite disparity that are used in sequence so as to minimize the total disparity of a longer sequence of digits.

A particular codeword of any line code can either have no disparity (the average weight of the codeword is zero), negative disparity (the average weight of the codeword is negative), or positive disparity (the average weight of the codeword is positive).

In a paired disparity code, every codeword that averages to a negative level (negative disparity) is paired with some other codeword that averages to a positive level (positive disparity).

In a system that uses a paired disparity code, the transmitter must keep track of the running DC buildup – the running disparity – and always pick the codeword that pushes the DC level back towards zero. The receiver is designed so that either codeword of the pair decodes to the same data bits.

The digits may be represented by disparate physical quantities, such as two different frequencies, phases, voltage levels, magnetic polarities, or electrical polarities, each one of the pair representing a 0 or a 1.

Most line codes use either a paired disparity code or a constant-weight code.

The simplest paired disparity code is alternate mark inversion signal. Other paired disparity codes include 8B10B, 8B12B, the modified AMI codes, coded mark inversion, and 4B3T.

[5] https://en.wikipedia.org/wiki/Paired_disparity_code

- MMS43-Code[6]

Each 4-bit input group is encoded as a 3-symbol group (transmitted left to right) from the following table. Encoding requires keeping track of the accumulated DC offset, the number of + pulses minus the number of − pulses in all preceding groups. The starting value is arbitrary; here we use the values 1 through 4, although −1.5, −0.5, +0.5 and +1.5 is another possibility.

Encoding table

MMS 43 coding table[1]

Input		Accumulated DC offset			
Hex	Binary	1	2	3	4
0	0000	+ 0 + (+2)	0 −0 (−1)		
1	0001	0 − + (+0)			
2	0010	+ − 0 (+0)			
3	0011	0 0 + (+1)			− − 0 (−2)
4	0100	− + 0 (+0)			
5	0101	0 + + (+2)	− 0 0 (−1)		
6	0110	− + + (+1)		− − + (−1)	
7	0111	− 0 + (+0)			
8	1000	+ 0 0 (+1)			0 − − (−2)
9	1001	+ − + (+1)			− − − (−3)
A	1010	+ + − (+1)		+ − − (−1)	
B	1011	+ 0 − (+0)			
C	1100	+ + + (+3)	− + − (−1)		
D	1101	0 + 0 (+1)			− 0 − (−2)
E	1110	0 + − (+0)			
F	1111	+ + 0 (+2)	0 0 − (−1)		

7

[6] https://en.wikipedia.org/wiki/4B3T

[7] *"Wired Communications T-SMINTO 4B3T Second Gen. Modular ISDN NT (Ordinary)" (PDF) (Data sheet). Version 1.1. Infineon. November 2001. PEF 80902.*

This code forces a transition after at most five consecutive identical non-zero symbols, or four consecutive zero symbols.

Decoding table

Ternary	Binary	Hex	Ternary	Binary	Hex	Ternary	Binary	Hex
0 0 0	N/A	N/A	− 0 0	0101	5	+ − −	1010	A
+ 0 +	0000	0	− + +	0110	6	+ 0 −	1011	B
0 − 0	0000	0	− − +	0110	6	+ + +	1100	C
0 − +	0001	1	− 0 +	0111	7	− + −	1100	C
+ − 0	0010	2	+ 0 0	1000	8	0 + 0	1101	D
0 0 +	0011	3	0 − −	1000	8	− 0 −	1101	D
− − 0	0011	3	+ − +	1001	9	0 + −	1110	E
− + 0	0100	4	− − −	1001	9	+ + 0	1111	F
0 + +	0101	5	+ + −	1010	A	0 0 −	1111	F

Decoding is simpler, as the decoder does not need to keep track of the encoder state, although doing so allows greater error detection. The 000 triplet is not a legal encoded sequence, but is typically decoded as binary 0000.

Contents


```
%-------------------------------------------------------------------------%
%%---- Final Project -- Simple Version -- Digital Communication -----%%
 %--------------- Supervisor: Dr.Shirvani Moghaddam ---------------%
  %------ Source by Mohammad Reza Farhadi Nia ---- Date:6 Dec 2020 --%
%-------------------------------------------------------------------------%
```

Encoding

```
pnSequence1 = comm.PNSequence('Polynomial','x^9+x^5+1',...
    'SamplesPerFrame',1022,'InitialConditions',[0 0 0 0 0 0 0 0 1]);
Binary_Random_Input = pnSequence1();
[Binary_Random_Input(1:511) Binary_Random_Input(512:1022)]; %validity test

pnSequence2 = comm.PNSequence('Polynomial','x^9+x^5+1',...
    'BitPackedOutput',true,'NumPackedBits',4,...
    'SamplesPerFrame',127,'InitialConditions',[0 0 0 0 0 0 0 0 1]);
Hex_Random_Input = pnSequence2();
FourBinary = dec2bin(Hex_Random_Input(1:25));

DC_bias_tracked = 0*(1:10);
Encoding = [];

    for i = 1:25

        switch Hex_Random_Input(i)

            case 0
                DC_bias_tracked(1) = ~DC_bias_tracked(1);
                if DC_bias_tracked(1) == 1
                    Encoding = [Encoding 1 0 0];
                else
                    Encoding = [Encoding -1 0 0];
                end

            case 1
                DC_bias_tracked(2) = ~DC_bias_tracked(2);
                if DC_bias_tracked(2) == 1
                    Encoding = [Encoding 0 -1 0];
                else
                    Encoding = [Encoding 0 1 0];
                end

            case 2
                DC_bias_tracked(3) = ~DC_bias_tracked(3);
                if DC_bias_tracked(3) == 1
                    Encoding = [Encoding 0 0 1];
                else
```

```
                Encoding = [Encoding 0 0 -1];
            end

        case 3
            DC_bias_tracked(4) = ~DC_bias_tracked(4);
            if DC_bias_tracked(4) == 1
                Encoding = [Encoding 1 1 0];
            else
                Encoding = [Encoding -1 -1 0];
            end

        case 4
            DC_bias_tracked(5) = ~DC_bias_tracked(5);
            if DC_bias_tracked(5) == 1
                Encoding = [Encoding 0 1 1];
            else
                Encoding = [Encoding 0 -1 -1];
            end

        case 5
            DC_bias_tracked(6) = ~DC_bias_tracked(6);
            if DC_bias_tracked(6) == 1
                Encoding = [Encoding 1 0 1];
            else
                Encoding = [Encoding -1 0 -1];
            end

        case 6
            DC_bias_tracked(7) = ~DC_bias_tracked(7);
            if DC_bias_tracked(7) == 1
                Encoding = [Encoding -1 1 -1];
            else
                Encoding = [Encoding 1 -1 1];
            end

        case 7
            DC_bias_tracked(8) = ~DC_bias_tracked(8);
            if DC_bias_tracked(8) == 1
                Encoding = [Encoding -1 -1 1];
            else
                Encoding = [Encoding 1 1 -1];
            end

        case 8
            DC_bias_tracked(9) = ~DC_bias_tracked(9);
            if DC_bias_tracked(9) == 1
                Encoding = [Encoding 1 -1 -1];
            else
                Encoding = [Encoding -1 1 1];
            end

        case 9
            DC_bias_tracked(10) = ~DC_bias_tracked(10);
            if DC_bias_tracked(10) == 1
                Encoding = [Encoding 1 1 1];
            else
                Encoding = [Encoding -1 -1 -1];
```

```
            end

        case 10
            Encoding = [Encoding -1 0 1];

        case 11
            Encoding = [Encoding 1 0 -1];


        case 12
            Encoding = [Encoding 0 -1 1];

        case 13
            Encoding = [Encoding 0 1 -1];


        case 14
            Encoding = [Encoding -1 1 0];


        case 15
            Encoding = [Encoding 1 -1 0];

    end
end
```

Decoding

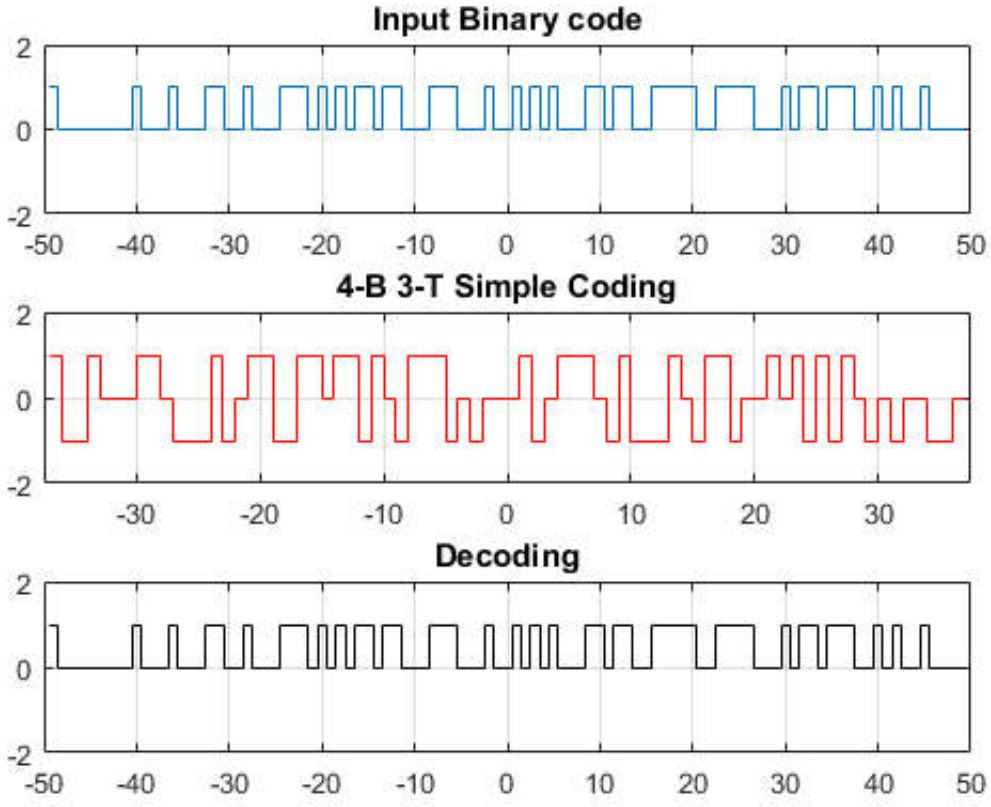
Most Important Code with Trick

```
ThreeTernary = join(string(reshape(Encoding,3,[]))'); % & HEAVY DEBUGING :)))
Decoding = [];
    for i = 1:25

        switch ThreeTernary(i)

           case {'1 0 0' , '-1 0 0'}
               Decoding = [Decoding 0 0 0 0];

           case {'0 -1 0' , '0 1 0'}
               Decoding = [Decoding 0 0 0 1];

           case {'0 0 1' , '0 0 -1'}
               Decoding = [Decoding 0 0 1 0];

           case {'1 1 0' , '-1 -1 0'}
               Decoding = [Decoding 0 0 1 1];

           case {'0 1 1' , '0 -1 -1'}
               Decoding = [Decoding 0 1 0 0];

           case {'1 0 1' , '-1 0 -1'}
               Decoding = [Decoding 0 1 0 1];

           case {'-1 1 -1' , '1 -1 1'}
```

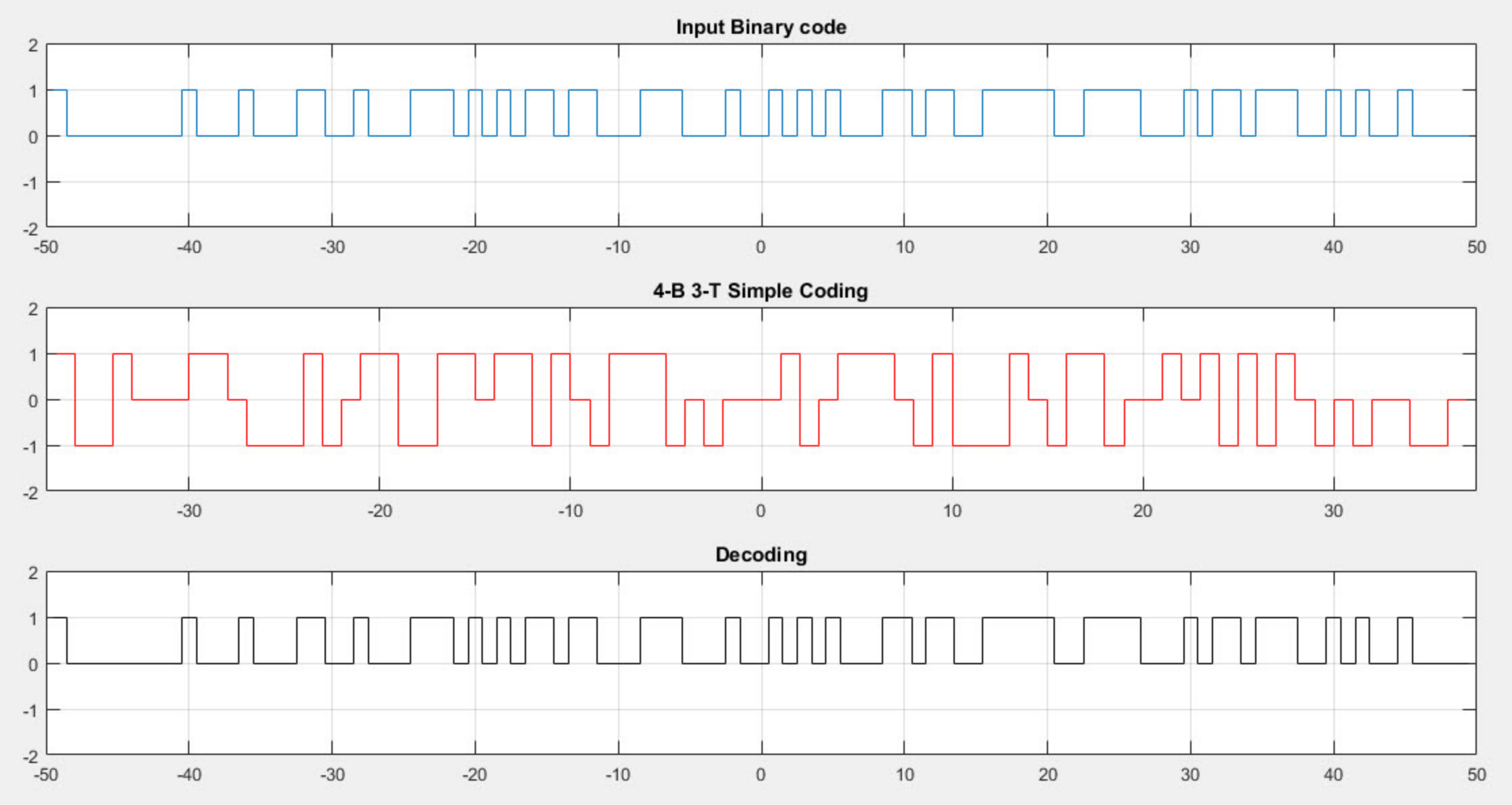
```
                Decoding = [Decoding 0 1 1 0];

            case {'-1 -1 1' , '1 1 -1'}
                Decoding = [Decoding 0 1 1 1];

            case {'1 -1 -1' , '-1 1 1'}
                Decoding = [Decoding 1 0 0 0];

            case {'-1 -1 -1' , '1 1 1'}
                Decoding = [Decoding 1 0 0 1];

            case '-1 0 1'
                Decoding = [Decoding 1 0 1 0];

            case '1 0 -1'
                Decoding = [Decoding 1 0 1 1];

            case '0 -1 1'
                Decoding = [Decoding 1 1 0 0];

            case '0 1 -1'
                Decoding = [Decoding 1 1 0 1];

            case '-1 1 0'
                Decoding = [Decoding 1 1 1 0];

            case '1 -1 0'
                Decoding = [Decoding 1 1 1 1];

        end
    end
```

Plot

```
figure

subplot(3,1,1);stairs([-length(Binary_Random_Input(1:100))/2+1/2:length(Binary_Random_Input(1
:100))/2-1/2],Binary_Random_Input(1:100));
axis([-length(Binary_Random_Input(1:100))/2 length(Binary_Random_Input(1:100))/2 -2 2]);title
('Input Binary code');grid on;

subplot(3,1,2);stairs([-length(Encoding)/2+1/2:length(Encoding)/2-1/2],Encoding, 'r');
axis([-length(Encoding)/2 length(Encoding)/2 -2 2]);title('4-B 3-T Simple Coding');grid on;

subplot(3,1,3);stairs([-length(Decoding)/2+1/2:length(Decoding)/2-1/2],Decoding,'black');
axis([-length(Decoding)/2 length(Decoding)/2 -2 2]);title('Decoding');grid on;
```



Published with MATLAB® R2016b


Input Binary code
2
1
0
-1
-2
-50
-40
-30
-20
-10
0
10
20
30
40
50
4-B 3-T Simple Coding
2
1
0
-1
-2
-30
-20
-10
0
10
20
30
Decoding
2
1
0
-1
-2
-50
-40
-30
-20
-10
0
10
20
30
40
50

Contents


```
%-------------------------------------------------------------------------%
%%-- Final Project -- Alternative Version -- Digital Communication --%%
  %--------------- Supervisor: Dr.Shirvani Moghaddam ----------------%
  %------ Source by Mohammad Reza Farhadi Nia ---- Date:6 Dec 2020 --%
%-------------------------------------------------------------------------%
```

Encoding

```
pnSequence1 = comm.PNSequence('Polynomial','x^9+x^5+1',...
    'SamplesPerFrame',1022,'InitialConditions',[0 0 0 0 0 0 0 0 1]);
Binary_Random_Input = pnSequence1();
[Binary_Random_Input(1:511) Binary_Random_Input(512:1022)]; %validity test

pnSequence2 = comm.PNSequence('Polynomial','x^9+x^5+1',...
    'BitPackedOutput',true,'NumPackedBits',4,...
    'SamplesPerFrame',127,'InitialConditions',[0 0 0 0 0 0 0 0 1]);
Hex_Random_Input = pnSequence2();
FourBinary = dec2bin(Hex_Random_Input(1:25));

DC_bias_tracked = 0*(1:10);
Encoding = [];

    for i = 1:25

        switch Hex_Random_Input(i)

            case 0
                DC_bias_tracked(1) = ~DC_bias_tracked(1);
                if DC_bias_tracked(1) == 1
                    Encoding = [Encoding 1 0 1];
                else
                    Encoding = [Encoding 0 -1 0];
                end

            case 1
                Encoding = [Encoding 0 -1 1];

            case 2
                Encoding = [Encoding 1 -1 0];

            case 3
                DC_bias_tracked(2) = ~DC_bias_tracked(2);
                if DC_bias_tracked(2) == 1
                    Encoding = [Encoding 0 0 1];
                else
                    Encoding = [Encoding -1 -1 0];
                end
```

```
        case 4
            Encoding = [Encoding -1 1 0];

        case 5
            DC_bias_tracked(3) = ~DC_bias_tracked(3);
            if DC_bias_tracked(3) == 1
                Encoding = [Encoding 0 1 1];
            else
                Encoding = [Encoding -1 0 0];
            end

        case 6
            DC_bias_tracked(4) = ~DC_bias_tracked(4);
            if DC_bias_tracked(4) == 1
                Encoding = [Encoding -1 1 1];
            else
                Encoding = [Encoding -1 -1 1];
            end

        case 7
            Encoding = [Encoding -1 0 +1];

        case 8
            DC_bias_tracked(5) = ~DC_bias_tracked(5);
            if DC_bias_tracked(5) == 1
                Encoding = [Encoding 1 0 0];
            else
                Encoding = [Encoding 0 -1 -1];
            end

        case 9
            DC_bias_tracked(6) = ~DC_bias_tracked(6);
            if DC_bias_tracked(6) == 1
                Encoding = [Encoding 1 -1 1];
            else
                Encoding = [Encoding -1 -1 -1];
            end

        case 10
            DC_bias_tracked(7) = ~DC_bias_tracked(7);
            if DC_bias_tracked(7) == 1
                Encoding = [Encoding 1 1 -1];
            else
                Encoding = [Encoding 1 -1 -1];
            end

        case 11
            Encoding = [Encoding 1 0 -1];

        case 12
            DC_bias_tracked(8) = ~DC_bias_tracked(8);
            if DC_bias_tracked(8) == 1
                Encoding = [Encoding 1 1 1];
            else
                Encoding = [Encoding -1 1 -1];
            end
```

```
            case 13
                DC_bias_tracked(9) = ~DC_bias_tracked(9);
                if DC_bias_tracked(9) == 1
                    Encoding = [Encoding 0 1 0];
                else
                    Encoding = [Encoding -1 0 -1];
                end

            case 14
                Encoding = [Encoding 0 1 -1];


            case 15
                DC_bias_tracked(10) = ~DC_bias_tracked(10);
                if DC_bias_tracked(10) == 1
                    Encoding = [Encoding 1 1 0];
                else
                    Encoding = [Encoding 0 0 -1];
                end
        end
    end
```

Decoding

Most Important Code with Trick

```
ThreeTernary = join(string(reshape(Encoding,3,[]))'); % & HEAVY DEBUGING :)))
Decoding = [];
```

Ternary	Binary	Hex	Ternary	Binary	Hex	Ternary	Binary	Hex
0 0 0	N/A	N/A	− 0 0	0101	5	+ − −	1010	A
+ 0 +	0000	0	− + +	0110	6	+ 0 −	1011	B
0 − 0	0000	0	− − +	0110	6	+ + +	1100	C
0 − +	0001	1	− 0 +	0111	7	− + −	1100	C
+ − 0	0010	2	+ 0 0	1000	8	0 + 0	1101	D
0 0 +	0011	3	0 − −	1000	8	− 0 −	1101	D
− − 0	0011	3	+ − +	1001	9	0 + −	1110	E
− + 0	0100	4	− − −	1001	9	+ + 0	1111	F
0 + +	0101	5	+ + −	1010	A	0 0 −	1111	F

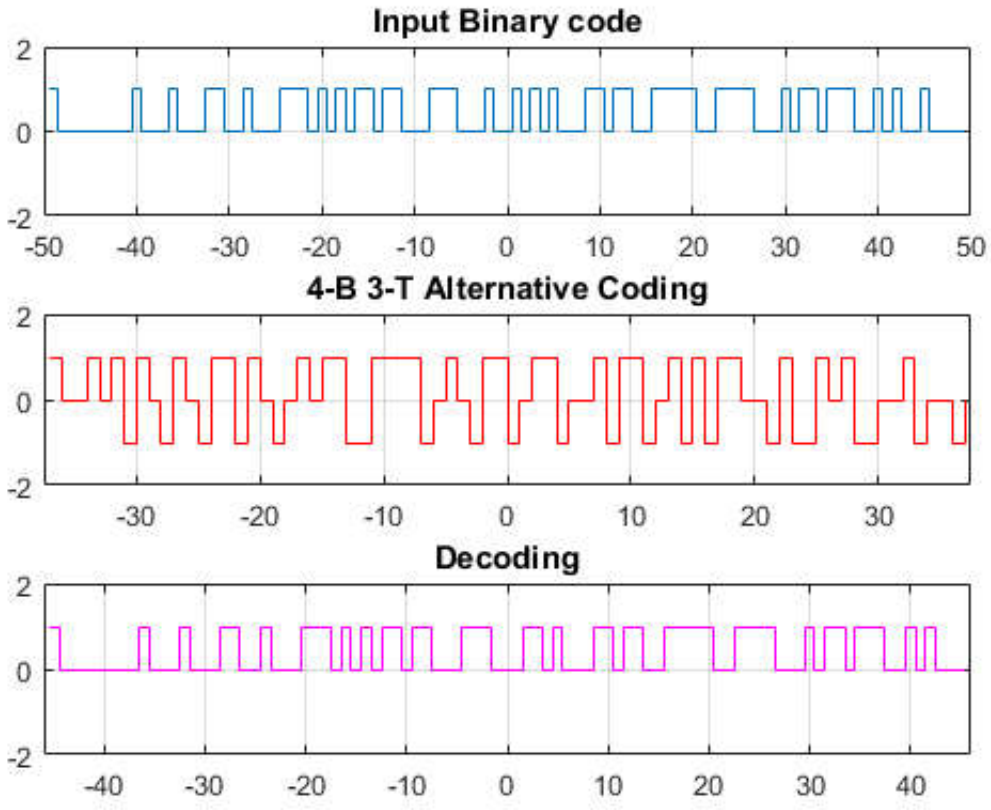
```
    for i = 1:25

        switch ThreeTernary(i)
```

```
            case {'1 0 1' , '0 -1 0'}
                Decoding = [Decoding 0 0 0 0];

            case {'0 -1 1'}
                Decoding = [Decoding 0 0 0 1];

            case {'1 - 0'}
                Decoding = [Decoding 0 0 1 0];

            case {'0 0 1' , '-1 -1 0'}
                Decoding = [Decoding 0 0 1 1];

            case {'-1 1 0'}
                Decoding = [Decoding 0 1 0 0];

            case {'0 1 1' , '-1 0 0'}
                Decoding = [Decoding 0 1 0 1];

            case {'-1 1 1' , '-1 -1 1'}
                Decoding = [Decoding 0 1 1 0];

            case {'-1 0 1'}
                Decoding = [Decoding 0 1 1 1];

            case {'1 0 0' , '0 -1 -1'}
                Decoding = [Decoding 1 0 0 0];

            case {'1 -1 1' , '-1 -1 -1'}
                Decoding = [Decoding 1 0 0 1];

            case {'1 1 -1' , '1 -1 -1'}
                Decoding = [Decoding 1 0 1 0];

            case {'1 0 -1'}
                Decoding = [Decoding 1 0 1 1];

            case {'1 1 1' , '-1 1 -1'}
                Decoding = [Decoding 1 1 0 0];

            case {'0 1 0' , '-1 0 -1'}
                Decoding = [Decoding 1 1 0 1];

            case {'0 1 -1'}
                Decoding = [Decoding 1 1 1 0];

            case {'1 1 0' , '0 0 -1'}
                Decoding = [Decoding 1 1 1 1];

        end
    end
```

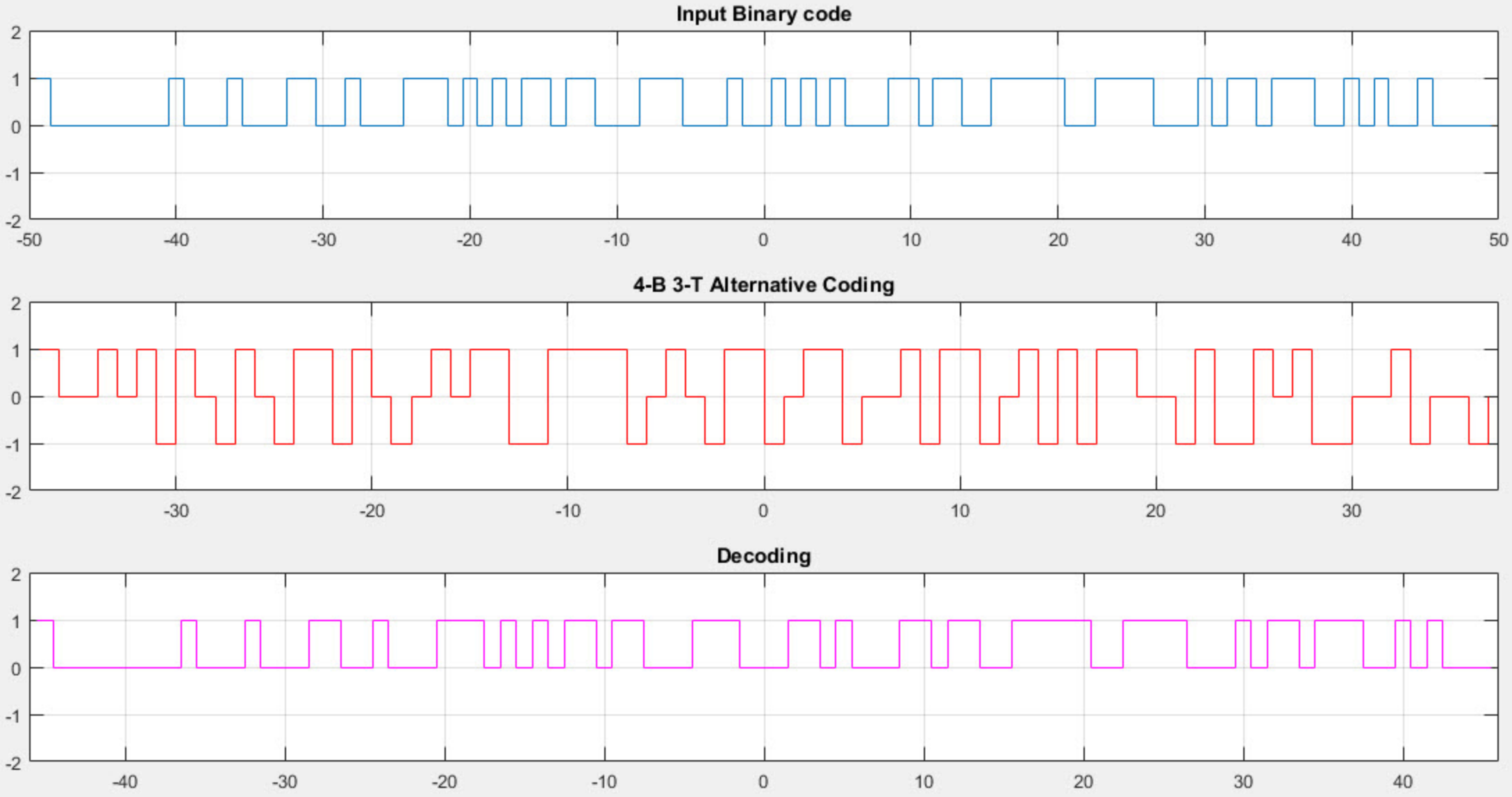
Plot

```
figure
```

```
subplot(3,1,1);stairs([-length(Binary_Random_Input(1:100))/2+1/2:length(Binary_Random_Input(1
:100))/2-1/2],Binary_Random_Input(1:100));
axis([-length(Binary_Random_Input(1:100))/2 length(Binary_Random_Input(1:100))/2 -2 2]);title
('Input Binary code');grid on;

subplot(3,1,2);stairs([-length(Encoding)/2+1/2:length(Encoding)/2-1/2],Encoding, 'r');
axis([-length(Encoding)/2 length(Encoding)/2 -2 2]);title('4-B 3-T Alternative Coding');grid
on;

subplot(3,1,3);stairs([-length(Decoding)/2+1/2:length(Decoding)/2-1/2],Decoding,'magenta');
axis([-length(Decoding)/2 length(Decoding)/2 -2 2]);title('Decoding');grid on;
```



Published with MATLAB® R2016b


Input Binary code
2
1
0
-1
-2
-50
-40
-30
-20
-10
0
10
20
30
40
50
4-B 3-T Alternative Coding
2
1
0
-1
-2
-30
-20
-10
0
10
20
30
Decoding
2
1
0
-1
-2
-40
-30
-20
-10
0
10
20
30
40

Contents


```
%-------------------------------------------------------------------------%
%%---- Final Project -- Optimum Version -- Digital Communication ----%%
 %--------------- Supervisor: Dr.Shirvani Moghaddam ----------------%
  %----- Source by Mohammad Reza Farhadi Nia -- Date:6 Dec 2020 ----%
%-------------------------------------------------------------------------%
```

Encoding

```
pnSequence1 = comm.PNSequence('Polynomial','x^9+x^5+1',...
    'SamplesPerFrame',1022,'InitialConditions',[0 0 0 0 0 0 0 0 1]);
Binary_Random_Input = pnSequence1();
[Binary_Random_Input(1:511) Binary_Random_Input(512:1022)]; %validity test

pnSequence2 = comm.PNSequence('Polynomial','x^9+x^5+1',...
    'BitPackedOutput',true,'NumPackedBits',4,...
    'SamplesPerFrame',127,'InitialConditions',[0 0 0 0 0 0 0 0 1]);
Hex_Random_Input = pnSequence2();
FourBinary = dec2bin(Hex_Random_Input(1:25));

DC_bias_tracked = zeros(3,10); %Row 3 is for checking
Encoding = [];
```

MMS 43 coding table[1]

Input		Accumulated DC offset			
Hex	Binary	1	2	3	4
0	0000	+ 0 + (+2)	0 − 0 (−1)		
1	0001	0 − + (+0)			
2	0010	+ − 0 (+0)			
3	0011	0 0 + (+1)			− − 0 (−2)
4	0100	− + 0 (+0)			
5	0101	0 + + (+2)	− 0 0 (−1)		
6	0110	− + + (+1)		− − + (−1)	
7	0111	− 0 + (+0)			
8	1000	+ 0 0 (+1)			0 − − (−2)
9	1001	+ − + (+1)			− − − (−3)
A	1010	+ + − (+1)		+ − − (−1)	
B	1011	+ 0 − (+0)			
C	1100	+ + + (+3)	− + − (−1)		
D	1101	0 + 0 (+1)			− 0 − (−2)
E	1110	0 + − (+0)			
F	1111	+ + 0 (+2)	0 0 − (−1)		

```
for i = 1:25

    switch Hex_Random_Input(i)

        case 0
            DC_bias_tracked(3,1) = DC_bias_tracked(3,1)+1;
            if DC_bias_tracked(1,1) == 0 && DC_bias_tracked(2,1) == 3
                Encoding = [Encoding 1 0 1];
                DC_bias_tracked(1,1) = ~DC_bias_tracked(1,1);
                DC_bias_tracked(2,1) = 0;
            else
                Encoding = [Encoding 0 -1 0];
                DC_bias_tracked(2,1) = DC_bias_tracked(2,1) + 1;
                DC_bias_tracked(1,1) = 0;
            end

        case 1
            Encoding = [Encoding 0 -1 1];

        case 2
```

```
            Encoding = [Encoding 1 -1 0];

        case 3
            DC_bias_tracked(3,2) = DC_bias_tracked(3,2)+1;
            if DC_bias_tracked(1,2) < 3
                Encoding = [Encoding 0 0 1];
                DC_bias_tracked(1,2) = DC_bias_tracked(1,2) + 1;
            else
                Encoding = [Encoding -1 -1 0];
                DC_bias_tracked(1,2) = 0;
            end

        case 4
            Encoding = [Encoding -1 1 0];

        case 5
            DC_bias_tracked(3,3) = DC_bias_tracked(3,3)+1;
            if DC_bias_tracked(1,3) == 0 && DC_bias_tracked(2,3) == 3
                Encoding = [Encoding 0 1 1];
                DC_bias_tracked(1,3) = ~DC_bias_tracked(1,3);
                DC_bias_tracked(2,3) = 0;
            else
                Encoding = [Encoding -1 0 0];
                DC_bias_tracked(2,3) = DC_bias_tracked(2,3) + 1;
                DC_bias_tracked(1,3) = 0;
            end

        case 6
            DC_bias_tracked(3,4) = DC_bias_tracked(3,4)+1;
            if DC_bias_tracked(1,4) < 2
                Encoding = [Encoding -1 1 1];
                DC_bias_tracked(1,4) = DC_bias_tracked(1,4) + 1;
            elseif DC_bias_tracked(2,4) < 2 && DC_bias_tracked(1,4) == 2
                Encoding = [Encoding -1 -1 1];
                DC_bias_tracked(2,4) = DC_bias_tracked(2,4) + 1;
            else
                DC_bias_tracked(1,4) = 0;
                DC_bias_tracked(2,4) = 0;
            end

        case 7
            Encoding = [Encoding -1 0 +1];

        case 8
            DC_bias_tracked(3,5) = DC_bias_tracked(3,5)+1;
            DC_bias_tracked(1,5) < 3;
            if DC_bias_tracked(1,5) == 1
                Encoding = [Encoding 1 0 0];
                DC_bias_tracked(1,5) = DC_bias_tracked(1,5) +1;
            else
                Encoding = [Encoding 0 -1 -1];
                DC_bias_tracked(1,5) = 0;
            end

        case 9
            DC_bias_tracked(3,6) = DC_bias_tracked(3,6)+1;
            if DC_bias_tracked(1,6) < 3
```

```
            Encoding = [Encoding 1 -1 1];
            DC_bias_tracked(1,6) = DC_bias_tracked(1,6) +1;
        else
            Encoding = [Encoding -1 -1 -1];
            DC_bias_tracked(1,6) = 0;
        end

    case 10
        DC_bias_tracked(3,7) = DC_bias_tracked(3,7)+1;
        if DC_bias_tracked(1,7) < 2
            Encoding = [Encoding 1 1 -1];
        elseif DC_bias_tracked(2,10) < 2 && DC_bias_tracked(1,10) == 2
            Encoding = [Encoding 1 -1 -1];
            DC_bias_tracked(2,10) = DC_bias_tracked(2,10) + 1;
        else
            DC_bias_tracked(1,10) = 0;
            DC_bias_tracked(2,10) = 0;
        end

    case 11
        Encoding = [Encoding 1 0 -1];

    case 12
        DC_bias_tracked(3,8) = DC_bias_tracked(3,8)+1;
        if DC_bias_tracked(1,8) == 0 && DC_bias_tracked(2,8) == 3
            Encoding = [Encoding 1 1 1];
            DC_bias_tracked(1,8) = ~DC_bias_tracked(1,8);
            DC_bias_tracked(2,8) = 0;
        else
            Encoding = [Encoding -1 1 -1];
            DC_bias_tracked(2,8) = DC_bias_tracked(2,8) + 1;
            DC_bias_tracked(1,8) = 0;
        end

    case 13
        DC_bias_tracked(3,9) = DC_bias_tracked(3,9)+1;
        if DC_bias_tracked(1,9) < 3
            Encoding = [Encoding 0 1 0];
            DC_bias_tracked(1,9) = DC_bias_tracked(1,9) + 1;
        else
            Encoding = [Encoding -1 0 -1];
            DC_bias_tracked(1,9) = 0;
        end

    case 14
        Encoding = [Encoding 0 1 -1];


    case 15
        DC_bias_tracked(3,10) = DC_bias_tracked(3,10)+1;
        if DC_bias_tracked(1,10) == 0 && DC_bias_tracked(2,10) == 3
            Encoding = [Encoding 1 1 0];
            DC_bias_tracked(1,10) = ~DC_bias_tracked(1,10);
            DC_bias_tracked(2,10) = 0;
        else
            Encoding = [Encoding 0 0 -1];
            DC_bias_tracked(2,10) = DC_bias_tracked(2,10) + 1;
```

```
                DC_bias_tracked(1,10) = 0;
            end
        end
    end
```

Decoding

Most Important Code with Trick

```
ThreeTernary = join(string(reshape(Encoding,3,[]))'); % & HEAVY DEBUGING :)))
Decoding = [];
```

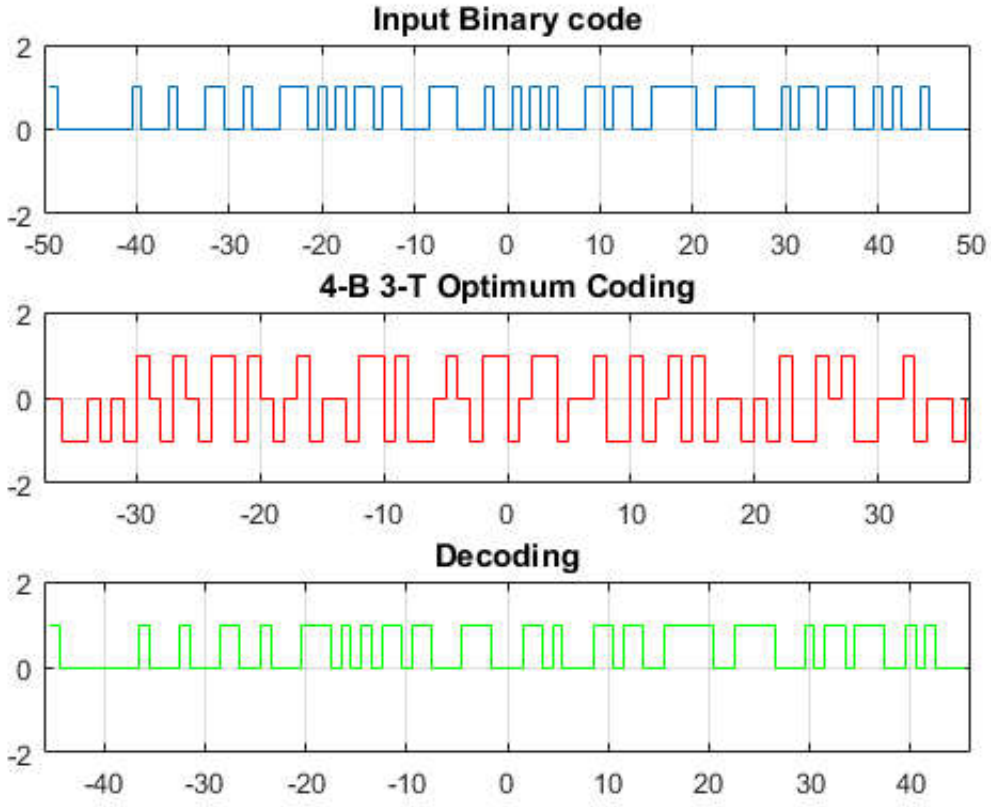
Ternary	Binary	Hex	Ternary	Binary	Hex	Ternary	Binary	Hex
0 0 0	N/A	N/A	− 0 0	0101	5	+ − −	1010	A
+ 0 +	0000	0	− + +	0110	6	+ 0 −	1011	B
0 − 0	0000	0	− − +	0110	6	+ + +	1100	C
0 − +	0001	1	− 0 +	0111	7	− + −	1100	C
+ − 0	0010	2	+ 0 0	1000	8	0 + 0	1101	D
0 0 +	0011	3	0 − −	1000	8	− 0 −	1101	D
− − 0	0011	3	+ − +	1001	9	0 + −	1110	E
− + 0	0100	4	− − −	1001	9	+ + 0	1111	F
0 + +	0101	5	+ + −	1010	A	0 0 −	1111	F

```
    for i = 1:25

        switch ThreeTernary(i)

            case {'1 0 1' , '0 -1 0'}
                Decoding = [Decoding 0 0 0 0];

            case {'0 -1 1'}
                Decoding = [Decoding 0 0 0 1];

            case {'1 - 0'}
                Decoding = [Decoding 0 0 1 0];

            case {'0 0 1' , '-1 -1 0'}
                Decoding = [Decoding 0 0 1 1];

            case {'-1 1 0'}
                Decoding = [Decoding 0 1 0 0];

            case {'0 1 1' , '-1 0 0'}
                Decoding = [Decoding 0 1 0 1];
```

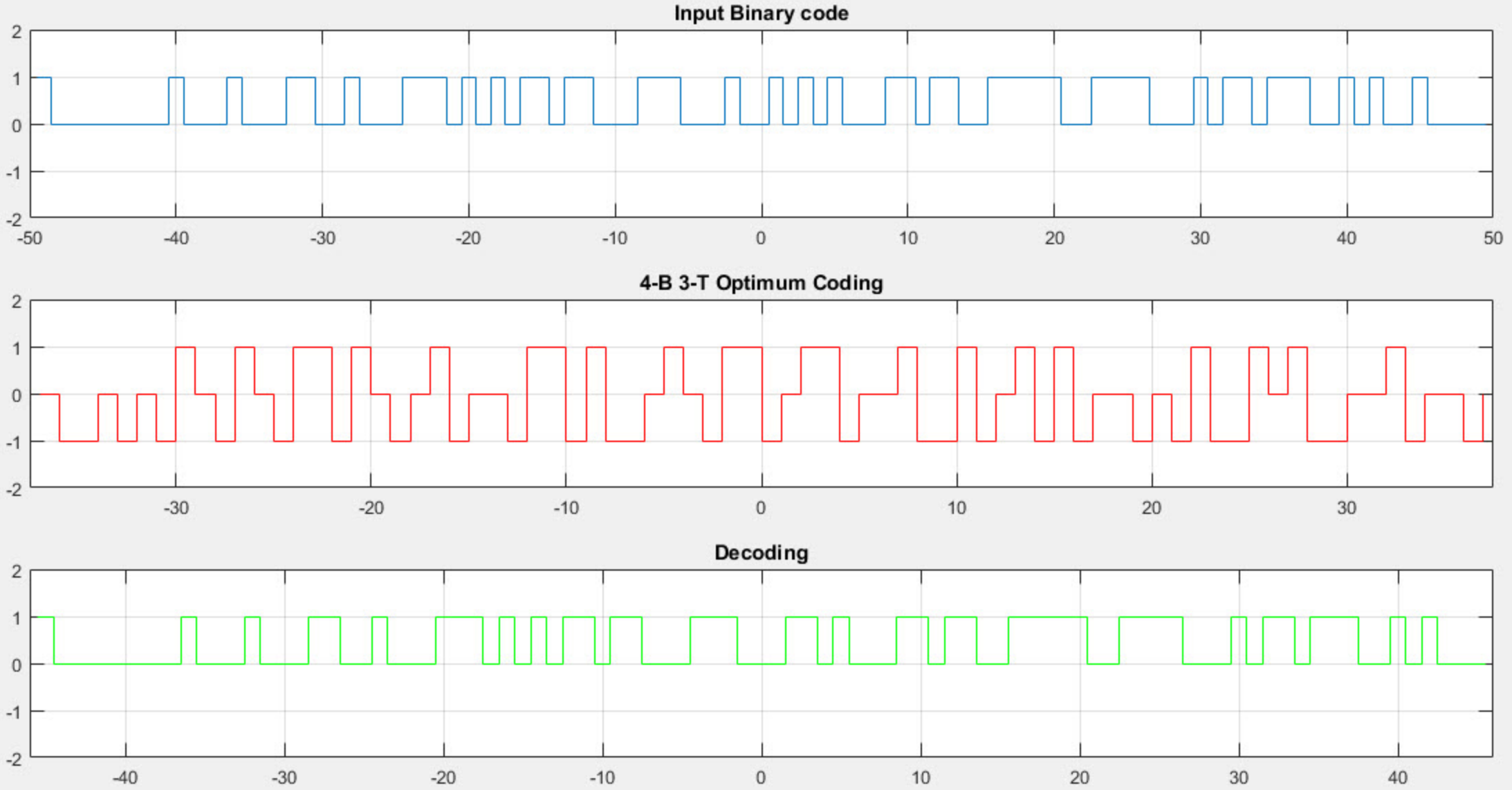
```
            case {'-1 1 1' , '-1 -1 1'}
                Decoding = [Decoding 0 1 1 0];

            case {'-1 0 1'}
                Decoding = [Decoding 0 1 1 1];

            case {'1 0 0' , '0 -1 -1'}
                Decoding = [Decoding 1 0 0 0];

            case {'1 -1 1' , '-1 -1 -1'}
                Decoding = [Decoding 1 0 0 1];

            case {'1 1 -1' , '1 -1 -1'}
                Decoding = [Decoding 1 0 1 0];

            case {'1 0 -1'}
                Decoding = [Decoding 1 0 1 1];

            case {'1 1 1' , '-1 1 -1'}
                Decoding = [Decoding 1 1 0 0];

            case {'0 1 0' , '-1 0 -1'}
                Decoding = [Decoding 1 1 0 1];

            case {'0 1 -1'}
                Decoding = [Decoding 1 1 1 0];

            case {'1 1 0' , '0 0 -1'}
                Decoding = [Decoding 1 1 1 1];

        end
    end
```

Plot

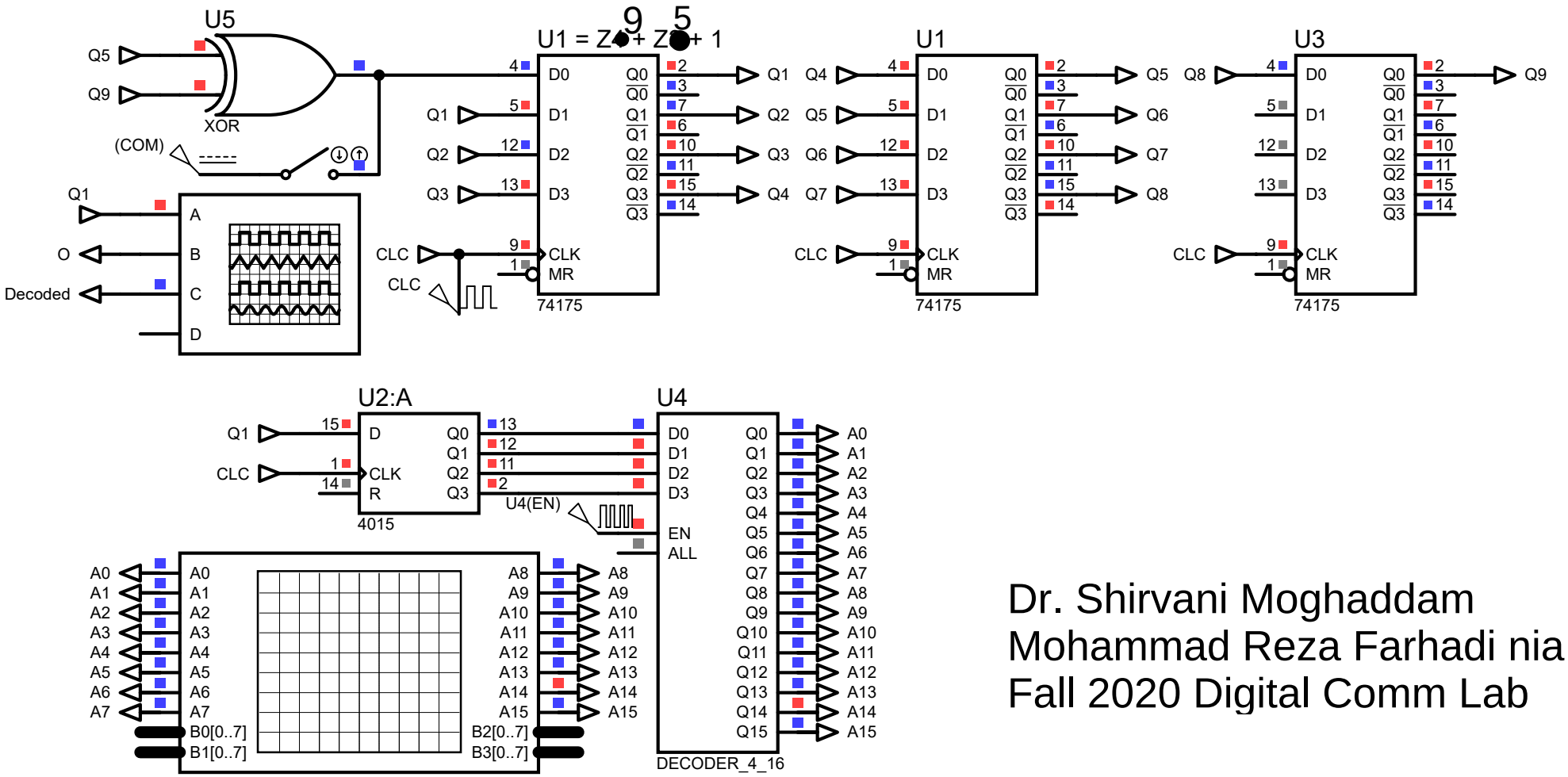
```
figure

subplot(3,1,1);stairs([-length(Binary_Random_Input(1:100))/2+1/2:length(Binary_Random_Input(1
:100))/2-1/2],Binary_Random_Input(1:100));
axis([-length(Binary_Random_Input(1:100))/2 length(Binary_Random_Input(1:100))/2 -2 2]);title
('Input Binary code');grid on;

subplot(3,1,2);stairs([-length(Encoding)/2+1/2:length(Encoding)/2-1/2],Encoding, 'r');
axis([-length(Encoding)/2 length(Encoding)/2 -2 2]);title('4-B 3-T Optimum Coding');grid on;

subplot(3,1,3);stairs([-length(Decoding)/2+1/2:length(Decoding)/2-1/2],Decoding,'green');
axis([-length(Decoding)/2 length(Decoding)/2 -2 2]);title('Decoding');grid on;
```



Published with MATLAB® R2016b

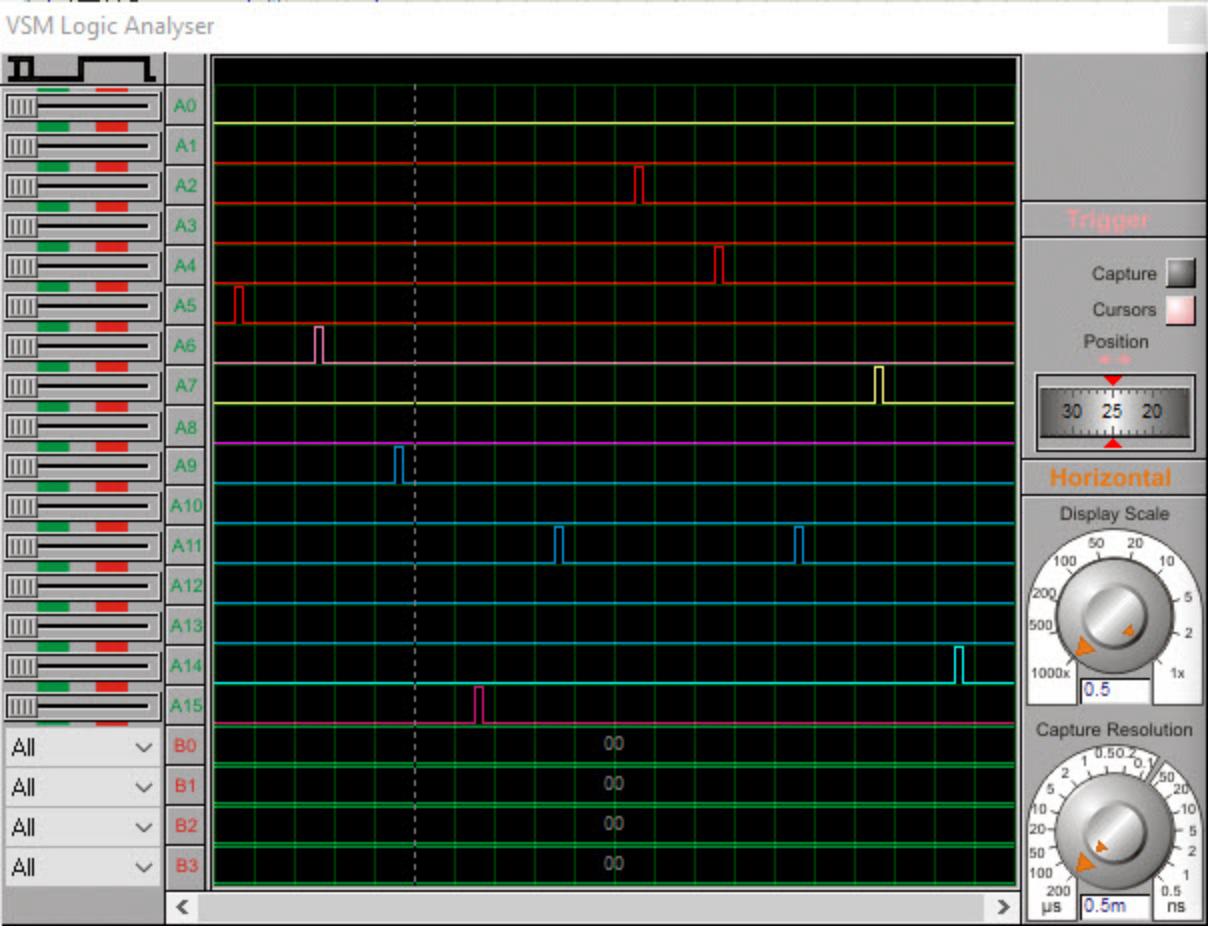

Input Binary code
2
1
0
-1
-2
-50
-40
-30
-20
-10
0
10
20
30
40
50
4-B 3-T Optimum Coding
2
1
0
-1
-2
-30
-20
-10
0
10
20
30
Decoding
2
1
0
-1
-2
-40
-30
-20
-10
0
10
20
30
40

Section 1 & 2
LFSR - 4B3T Modulation

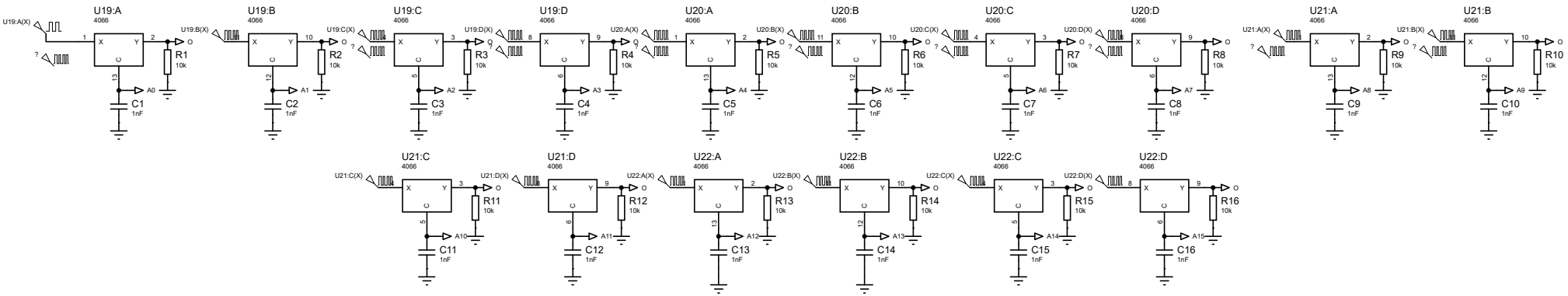


Dr. Shirvani Moghaddam
Mohammad Reza Farhadi nia

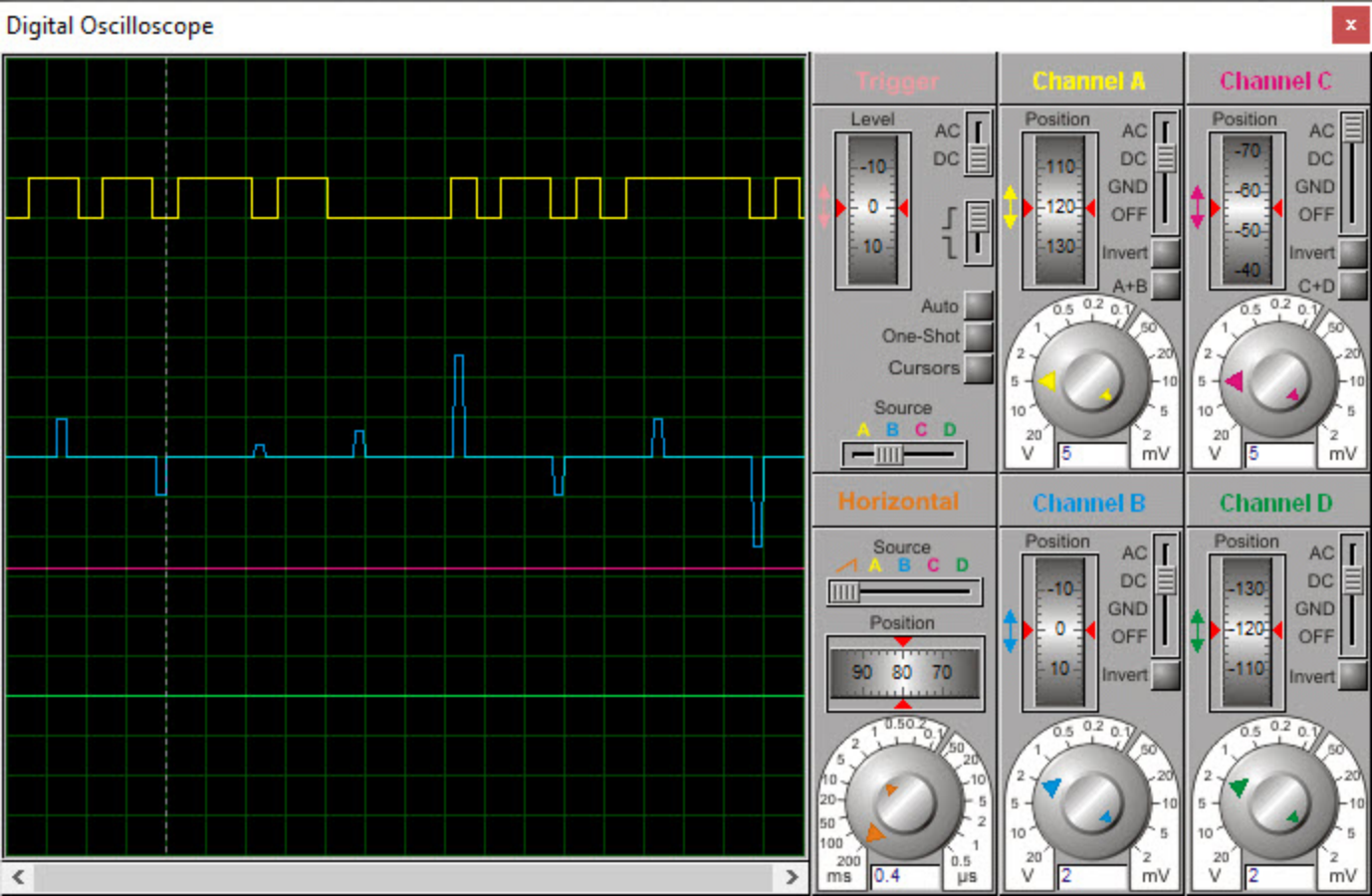
Fall 2020 Digital Comm Lab


VSM Logic Analyser
A0
A1
A2
A3
A4
A5
A6
A7
A8
A9
A10
A11
A12
A13
A14
A15
All
All
All
All
B0
B1
B2
B3
00
00
00
00
Trigger
Capture
Cursors
Position
30 25 20
Horizontal
Display Scale
50 20 10 100 5 200 2 500 1000x 1x
0.5
Capture Resolution
0.50.2 1 0.1 2 50 20 5 10 10 20 5 50 2 100 1 200 0.5 µs ns
0.5m

Section 2
4B3T Modulation



I could not exactly simulation in Proteus, becuase I could not make pattern pulse with different frequency so I have used different DC voltage in project files. and I think for simulation in Proteus we need some Ternary Gate or Microcontroller, finally maybe Binary Gate become cumbersome


Digital Oscilloscope
Trigger
Level
AC
DC
Auto
One-Shot
Cursors
Source
A B C D
Channel A
Position
AC
DC
GND
OFF
Invert
A+B
V
mV
Channel C
C+D
Horizontal
ms
µs
0.4
Channel B
Channel D
2
5

Section 3
Demodulation

Microcontroller or Ternary Gate and Latch