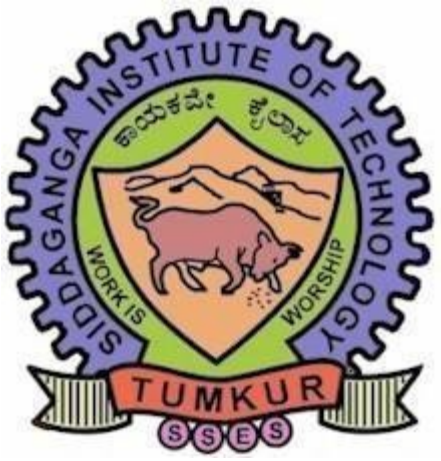
A
Group Activity Report
on

8b6t

Submitted by
Kaisar Shabir 1SI17CS045



Department of Computer Science and Engineering

Siddaganga Institute of Technology, Tumakuru – 572103

(An Autonomous Institution, Affiliated to VTU, Belagavi & Recognized by AICTE, New Delhi)

2019 -2020

SIDDAGANGA INSTITUTE OF TECHNOLOGY, TUMAKURU-3

(An Autonomous Institution, Affiliated to VTU, Belagavi & Recognized by AICTE, New Delhi)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Name of student	USN	Marks
Kaisar Shabir	1SI17CS045	

Signature of In charge Faculty

Department of Computer Science & Engineering

Siddaganga Institute of Technology, Tumakuru-3

(An Autonomous Institute affiliated to VTU,Accredited by NBA)**2019-20**

TABLE OF CONTENTS

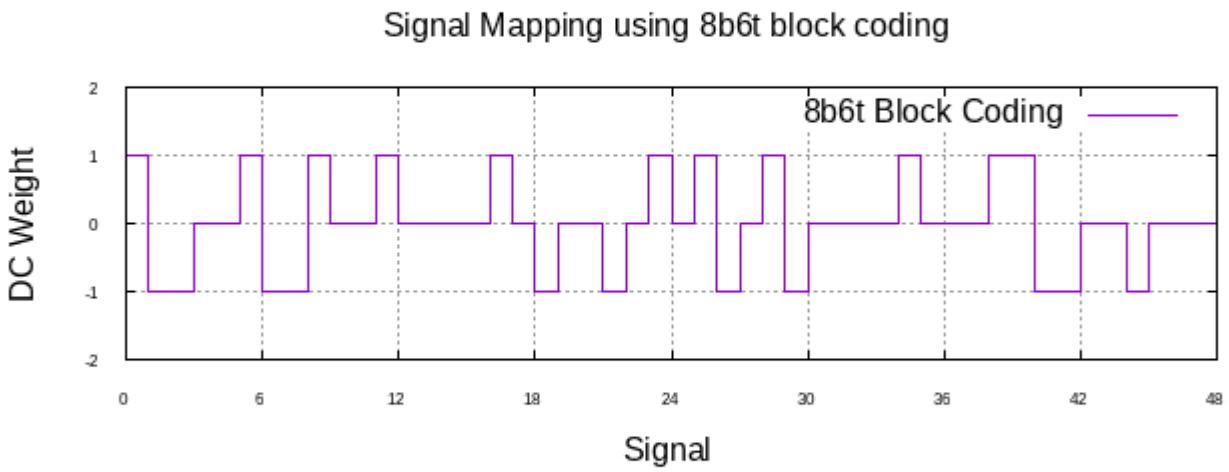
ABSTRACT

In 8B6T we encode a pattern of 8 bits as a pattern of 6 signals, where the signal has three signal levels rather than 2 in the original data.

Since the original data unit is of length 8 bits and has 2 signal levels, we can have 2 8 different data patterns where as each signal unit has length of 6 and 3 levels, we can have 3 6 signal patterns.

There are 729 − 256 = 473 redundant signal elements that provide synchronization and error detection. Redundancy is also used to provide DC balance.

Each signal pattern has a weight of 0 or +1 DC values. This means that there is no pattern with the weight −1. To make the whole stream DC-balanced, the sender keeps track of the weight. If two groups of weight 1 are encountered one after another, the first one is sent as is, while the next one is totally inverted to give a weight of −1.



IMPLEMENTATION

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define VAL_INDEX 6
#define W_INDEX 7
#define TOTAL_COMBINATION 256

//ASSING A NUMBER AS IF IT WAS BASE 3, SO THAT PATTERN COULD BE UNIQUELY IDENTIFIED
int power(int x, int y){
	int i, res = 1;

	for(y; y > 0; y--)
		res *= x;

	return res;
}

//GENERATE 6T CODE
void fnGen6t(int* ia6t){
	int i;
	int sum = 0, weight = 0;

	for(i = 0; i < 6; i++){
		ia6t[i] = rand() % 3 - 1;
		weight = weight + ia6t[i];
		sum = sum + (power(3, 6 - i)) * (ia6t[i] + 1);
	}

	ia6t[VAL_INDEX] = sum;
	ia6t[W_INDEX] = weight;
```

```
	if(weight != 0 && weight != 1)
		fnGen6t(ia6t);
}

//GENERTAE THE MAPPING
void fnGen8b6t(int ia8b6tMapping[][8], int iBytes){
	int i, j;
	int ia6t[8];

	for(i = 0; i < iBytes; i++){
		startAgain:
			fnGen6t(ia6t);

		if(i != 0)
			for(j = 0; j < i; j++)
				if(ia8b6tMapping[j][VAL_INDEX] == ia6t[VAL_INDEX])
					goto startAgain;

		for(j = 0; j < 8; j++)
			ia8b6tMapping[i][j] = ia6t[j];
	}
}

//DISPLAY THE SINGLE UNIT
void fnDispSignalUnit(int signalUnit){
	if(signalUnit == -1)
		printf("-");
	else if(signalUnit == 0)
		printf("0");
	else
		printf("+");
```

```
}

//CHECK IF THE SIGNAL NEEDS TO BE INVERTED
char* fnCheckInver(int check){
	if(!check)
		return "NO";
	else if(check == 1)
		return "YES";
	return "NA";
}

//DISPLAY THE SIGNAL MAPPING
void fnShow8b6tMapping(int map[][8], int iTotalBytes, int iInverted){
	int i, j;
	printf("-----------------------------------------------------------------\
n");
	printf("|   Signal   |\t Value\t |   Data    |    Weight  |  Inverted   |\
n");
	printf("-----------------------------------------------------------------\
n");

	for(i = 0; i < iTotalBytes; i++){
		printf("|   ");
		for(j = 0; j < 6; j++)
			fnDispSignalUnit(map[i][j]);
		printf("   |\t  %d\t |    %-4d   |   %4d     |       %s     |", map[i]
[VAL_INDEX], i, map[i][W_INDEX], fnCheckInver(iInverted));
		printf("\n");
	}
	printf("-----------------------------------------------------------------\
n");
	printf("\n");
```

```
}

//DISPLAY THE MAPPING FOR ALL PATTERNS
void fnShowSignalMapping(int iaSignal[][8], int iBytes){

	int i, j;
	int weight = 0;
	int iInversion = 0;

	system("clear");

	printf("---------------------------------\n");
	printf("|Signal\t| Data\t|Weight\t|Inverted|\n");
	printf("---------------------------------\n");

	for(i = 0; i < iBytes; i++){
		printf("|");
		if(weight == 1 && iaSignal[i][W_INDEX] == 1){
			for(j = 0; j < 6; j++)
				fnDispSignalUnit(iaSignal[i][j] * -1);
			weight = 0;
			iInversion = 1;
		}
		else{
			for(j = 0; j < 6; j++)
				fnDispSignalUnit(iaSignal[i][j]);
			weight += iaSignal[i][W_INDEX];
		}

		printf("\t|   %d\t|%4d\t|%6s\t |", iaSignal[i][VAL_INDEX], iaSignal[i]
[W_INDEX], fnCheckInver(iInversion));
```

```
            iInversion = 0;

            printf("\n");
      }

      printf("---------------------------------\n");
}

//CONVERT DATA TO SIGNAL
void fnDataToSignal(int ia8b61Mapping[][8], int iaSignal[][8], int iBytes){
      int i, j;

      for(i = 0; i < iBytes; i++){
            label3:
            printf("Enter the data byte %d in decimal:\t", i + 1);
            scanf("%d", &iaSignal[i][VAL_INDEX]);

            if(iaSignal[i][VAL_INDEX] < 0 || iaSignal[i][VAL_INDEX] > 255){
                  printf("Invalid data. Try again");
                  goto label3;
            }

            for(j = 0; j < 6; j++){
                  iaSignal[i][j] = ia8b61Mapping[iaSignal[i][VAL_INDEX]][j];
            }

            iaSignal[i][W_INDEX] = ia8b61Mapping[iaSignal[i][VAL_INDEX]][W_INDEX];
      }
}

//PLOTTING EACH POINT ON THE GRAPH
void fnPlotSignalUnit(int y, int x){
```

```
	FILE *fp;
	fp = fopen("8b6tPlot.dat", "a");
	fprintf(fp, "%d\t%d\n", x, y);
	fprintf(fp, "%d\t%d\n", x + 1, y);
	fclose(fp);
}

//PLOT THE SIGNAL ON TGHE GRAPH
void fnPlotSignal(int iaSignal[][8], int iBytes){
	int i, j;
	int weight = 0;
	int iInversion = 0;
	FILE *fp;

	fp = fopen("8b6tPlot.dat", "w");
	fprintf(fp, "%d\t%d\n", 0, 2);
	fprintf(fp, "%d\t%d\n", 0, -2);
	fclose(fp);

	for(i = 0; i < iBytes; i++){
		if(weight == 1 && iaSignal[i][W_INDEX] == 1){
			for(j = 0; j < 6; j++)
				fnPlotSignalUnit(iaSignal[i][j] * -1, i * 6 + j);
			weight = 0;
			iInversion = 1;
		}
		else{
			for(j = 0; j < 6; j++)
				fnPlotSignalUnit(iaSignal[i][j], i * 6 + j);
			weight += iaSignal[i][W_INDEX];
		}
```

```
            iInversion = 0;
      }
}

int main(void){

      int i, j;
      int ia8b6tMapping[TOTAL_COMBINATION][8];
      int maxBytes, iBytes;

      fnGen8b6t(ia8b6tMapping, TOTAL_COMBINATION);

      printf("Enter the maximum bytes of data:\t");
      scanf("%d", &maxBytes);

      system("clear");

      int iaSignal[maxBytes][8];

      int iChoice;
      int iAssigned = 0;

      while(1){
            printf("1. Genrate New 8b6t Mapping\n2. Show 8b6t Mapping\n3. Enter New
Input data\n");
            printf("4. Convert Data to Signal\n5. Plot cordinates.\n6. Take input
from file.\n");
            printf("7. Exit\n8. Edit code and execute\n");
            printf("Enter your choice:\t");
            scanf("%d", &iChoice);

            system("clear");
```

```
            switch(iChoice){
                  case 1:
                        fnGen8b6t(ia8b6tMapping, TOTAL_COMBINATION);

                  case 2:
                        fnShow8b6tMapping(ia8b6tMapping, TOTAL_COMBINATION, 2);
                        break;

                  case 3:
                        tryAgain:
                        iAssigned = 1;
                        printf("Enter the amount of data in bytes:\t");
                        scanf("%d", &iBytes);
                        if(iBytes > maxBytes){
                              printf("Maximum  data  can  be  %d  bytes.  PLease  try
again.\n", maxBytes);
                              goto tryAgain;
                        }
                        fnDataToSignal(ia8b6tMapping, iaSignal, iBytes);

                  case 4:
                        if(!iAssigned){
                              system("clear");
                              printf("No  previous  data  found.  Enter  data  first.\
n");
                              goto tryAgain;
                        }
                        fnShowSignalMapping(iaSignal, iBytes);

                        break;
                  case 5:
```

```
                        if(!iAssigned){
                              system("clear");
                              printf("No previous data found. Enter data first.\
n");
                              goto tryAgain;
                        }

                        fnPlotSignal(iaSignal, iBytes);
                        printf("Coordinates has been plotted in 8b6tPlot.dat file\
n");
                        system("gnuplot 8b6tPlot.gpl");
                        system("eog 8b6tPlot.png");
                        break;
                  case 6:
                        {
                              FILE *fp;
                              iBytes = 0;
                              int val;
                              char inFile[30];
                              printf("Enter the file name:\t");
                              scanf("%s", inFile);

                              fp = fopen(inFile, "r");
                              if(fp == NULL){
                                    printf("File %s does not exist.\n", inFile);
                                    break;
                              }
                              printf("Data in file is:\n");
                              while(fscanf(fp, "%d ", &val) != EOF && iBytes <=
maxBytes){
                                    printf("%d\t", val);
                                    if(val < 0 || val > 255){
```

```
                                    continue;
                                }

                                iaSignal[iBytes][VAL_INDEX] = val;

                                for(j = 0; j < 6; j++){
                                    iaSignal[iBytes][j] =
ia8b6tMapping[iaSignal[iBytes][VAL_INDEX]][j];
                                }
                                iaSignal[iBytes][W_INDEX] =
ia8b6tMapping[iaSignal[iBytes][VAL_INDEX]][W_INDEX];
                                iBytes++;
                            }
                            printf("\n");
                            iAssigned = 1;
                        }
                        break;
                case 7:
                        exit(0);
                case 8:
                        system("gedit 8b6t-final.c");
                        system("gcc 8b6t-final.c");
                        system("./a.out");
                        exit(0);
                default:
                        printf("Invalid option. Try again.\n");
            }
        }

        return 0;
}
```

RESULTS

```
(base) kaiser@kaiser-Swift-SF514-52T:~/Documents/projects/DC$ gcc 8b6t-final.c
(base) kaiser@kaiser-Swift-SF514-52T:~/Documents/projects/DC$ ./a.out
Enter the maximum bytes of data:        456
```

figure 1: Enterint the maximumdata to be transferred

Signal	Value	Data	Weight	Inverted
-0+-++	429	0	1	NA
+0-+-0	1758	1	0	NA
+00+0-	1845	2	1	NA
0++-0-	1386	3	0	NA
+---++	1482	4	0	NA
+-+0+-	1665	5	1	NA
++--0+	1959	6	1	NA
--+++-	234	7	0	NA
-+0+-0	624	8	0	NA
-00+00	390	9	0	NA
00-++-	1044	10	0	NA
+-0-+0	1560	11	0	NA
+-+--+	1626	12	0	NA
00000+	1095	13	1	NA
0-+000	930	14	0	NA
0+--+0	1236	15	0	NA
+-+00-	1656	16	0	NA
0-+0-+	924	17	0	NA
+--++0	1533	18	1	NA
++-00-	1980	19	0	NA
+-+0-+	1653	20	1	NA
-+-00+	528	21	0	NA
000-++	1077	22	1	NA
0-++00	957	23	1	NA
+00-0+	1797	24	1	NA
0-0+0+	879	25	1	NA
+0-0-+	1734	26	0	NA
++0-+-	2043	27	1	NA
-0++-+	465	28	1	NA

figure 2: 8b6t mapping

```
Enter the amount of data in bytes:      20
Enter the data byte 1 in decimal:       1
Enter the data byte 2 in decimal:       2
Enter the data byte 3 in decimal:       56
Enter the data byte 4 in decimal:       225
Enter the data byte 5 in decimal:       256
Invalid data. Try againEnter the data byte 5 in decimal:        -1
Invalid data. Try againEnter the data byte 5 in decimal:        90
Enter the data byte 6 in decimal:       78
Enter the data byte 7 in decimal:       12
Enter the data byte 8 in decimal:       123
Enter the data byte 9 in decimal:       213
Enter the data byte 10 in decimal:      12
Enter the data byte 11 in decimal:      12
Enter the data byte 12 in decimal:      23
Enter the data byte 13 in decimal:      12
Enter the data byte 14 in decimal:      43
Enter the data byte 15 in decimal:      12
Enter the data byte 16 in decimal:      45
Enter the data byte 17 in decimal:      5
Enter the data byte 18 in decimal:      67
Enter the data byte 19 in decimal:      67
Enter the data byte 20 in decimal:      78
```

figure 3: Entering the data

Signal	Data	Weight	Inverted
+0-+-0	1	0	NO
+00+0-	2	1	NO
0000-0	56	1	YES
++0+--	225	1	NO
0-+--+	90	1	YES
-0+00+	78	1	NO
+-+--+	12	0	NO
0+0-+-	123	0	NO
+-0-0+	213	0	NO
+-+--+	12	0	NO
+-+--+	12	0	NO
0+--00	23	1	YES
+-+--+	12	0	NO
--++-+	43	0	NO
+-+--+	12	0	NO
-++000	45	1	NO
-+-0-+	5	1	YES
0+0+-0	67	1	NO
0-0-+0	67	1	YES
-0+00+	78	1	NO

figure 4: signal mapping of the data to be transferred

figure 5: Signal to be transferred

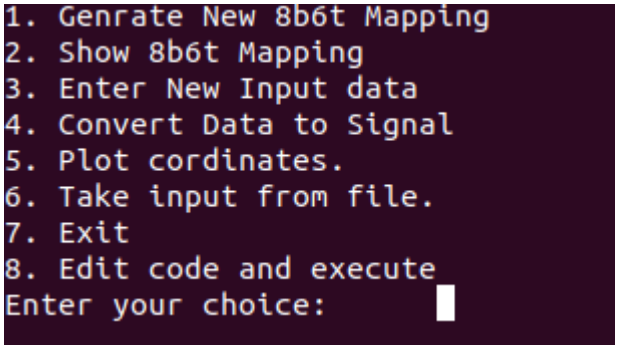


figure 6: All the options available

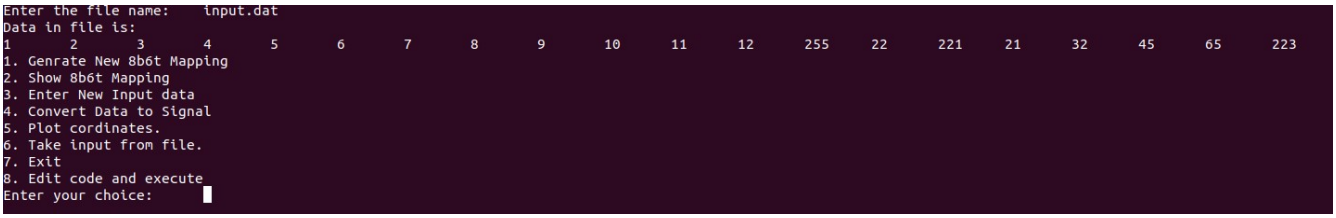


figure 7: Data read from the file