Experiment 1: Error Detection using Parity

Aim: To apply Parity check rules for error detection

Objective: After carrying out this experiment, students will be able to:

- Apply 1D and 2D parity rules for error detection
- Analyze the difference between 1D and 2D parity and their limitations

Problem statement: You are required to write separate programs to demonstrate the use of 1D and 2D parity. Take the input bit streams (max five) of 7 bit each from the user. Your programs should calculate the parity and display the input and output bit streams.

Analysis: While analyzing your program, you are required to address the following points:

- Why can this method not be used to correct errors?
- How are 1D and 2D parity different?
- What are the limitations of this method of error detection?

MARKS DISTRIBUTION

Component	Maximum Marks	Marks Obtained
Preparation of Document	7	
Results	7	
Viva	6	
Total	**20**	

Submitted by: **Deepak R**

Register No: **18ETCS002041**

1. Algorithm/Flowchart

1D-Parity(data): Algorithm

1. bit_count = 0
2. for_each(bit : data)
3. if (bit == 1) bit_count++
4. if (bit_count % 2 == 0)
5. parity_bit = 0
6. else
7. parity_bit = 1

- **2D Parity(Data) :Steps**
- Start
- Print the string “Enter 7 2D binary values:”
- Scan for 7 2D binary values
- Print the string “Enter 1 for even parity and 2 for odd parity.”
- Scan for user input
- Based on the user’s input the parity is calculated and printed.
- Stop

Algorithm 2D Parity

```
declarep[50][50], t, ch, i, j,n;
display the Number of bits
display the n bit 2D binary numbers
for(i=0;i<n;i++)
  {
for(j=0;j<n;j++)
display p[i][j]
  }
Display the choice, 1 for even parity and 2 for odd parity
switch(ch)
  {
case 1:
for(i=0;i<n;i++)
      {
         t=0;
for(j=0;j<n;j++)
      {
if(p[i][j]==1)
t++;
      }
if(t%2==0)
p[i][n]=0;
else
p[i][n]=1;
      }
for(j=0;j<n;j++)
      {
         t=0;
for(i=0;i<n;i++);
      {
if(p[i][j]==1)
t++;
      }
if(t%2==0)
p[n][j]=0;
else
p[n][j]=1;
      }
break;
case 2:
for(i=0;i<n;i++)
      {
         t=0;
for(j=0;j<n;j++)
if(p[i][j]==1)
t++;
      }
if(t%2!=0)
p[i][n]=0;
else
p[i][n]=1;
      }
for(j=0;j<n;j++)
      {
         t=0;
for(i=0;i<n;i++);
      {
if(p[i][j]==1)
t++;
      }
if(t%2!=0)
p[n][j]=0;
else
p[n][j]=1;
      }
break;
default:
display wrong choice
  }
for(i=0;i<n+1;i++)
  {
for(j=0;j<n+1;j++)
display p[i][j]
  }
```

2. Program 1D Parity

```c
#include <stdio.h>
#include <stdlib.h>
int n, array[100],received[100],count,choice;
int inputs(int array[100],int n)
{
    count=0;
     for(int i=0 ; i<n;i++)
    { scanf("%d",&array[i]);}
    for(int i=0;i<n;i++)
    {   if (array[i]==1)
           count++;
    }
}
int parity(int array[100], int choice){
     switch (choice) {
        case 1:
            array[n]= count%2;
            break;
        case 2:
            if(count%2 == 0)
                array[n]=1;
            else array[n]=0;
            break;
        default : printf("invalid choice");
    }
}
int main(int argc, char** argv) {
    printf("Enter the Number of bits:");
    scanf("%d",&n);
    printf("Enter bit: ");
    inputs(array,n);
    printf("1. Even Parity\n2. Odd parity\n");
    printf("select the choice: ");
    scanf("%d",&choice);
    parity(array,choice);
    printf("\nEnter the received data: \n");
    inputs(received,n);
    parity(received,choice);
    if(array[n]==received[n])
        printf("\nSystem has No Error\n");
    else printf("\nSystem has an Error\n");
    return (EXIT_SUCCESS);
}
```

2D-Parity

```c
//Program done by Deepak R 18ETCS002041
#include <stdio.h>
#include <stdlib.h>
int main (int argc, char** argv){
    int p[50][50];int t;int ch; int i;int j;int n=7;
    printf("Enter the 7 bit 2D binary numbers = \n");
    for(i=0;i<n;i++)
            {
        for(j=0;j<n;j++)
            scanf("%d",&p[i][j]);}
    printf(" the choice, 1 for even parity and 2 for odd parity = ");
    scanf("%d",&ch);
    switch(ch)
    {
        case 1:
            for(i=0;i<n;i++)
            {
                t=0;
                for(j=0;j<n;j++)
                {
                    if(p[i][j]==1)
                        t++;
                }
                if(t%2==0)
                    p[i][n]=0;
                else

                    p[i][n]=1;
            }
            for(j=0;j<n;j++)
            {
                t=0;
                for(i=0;i<n;i++);
                {
                    if(p[i][j]==1)
                        t++;
                }
                if(t%2==0)
                    p[n][j]=0;
                else
                    p[n][j]=1;
           }
            break;
        case 2:
            for(i=0;i<n;i++)
            {
                t=0;
                for(j=0;j<n;j++)
```

```
                {
                    if(p[i][j]==1)
                        t++;
                }
                if(t%2!=0)
                    p[i][n]=0;
                else
                    p[i][n]=1;
            }
            for(j=0;j<n;j++)
            {
                t=0;
                for(i=0;i<n;i++);
                {
                    if(p[i][j]==1)
                        t++;
                }
                if(t%2!=0)
                    p[n][j]=0;
                else
                    p[n][j]=1;
            }
            break;
        default:
            printf("Invalid option");
    }
    for(i=0;i<n+1;i++)
    {printf("\n");
        for(j=0;j<n+1;j++)
            printf("%d",p[i][j]);
    }
    printf("\n");
}
```

‘

3. Results

```
Enter the Number of bits:7
Enter bit: 1 0 0 1 0 0 1
1. Even Parity
2. Odd parity
select the choice: 1

Enter the received data: 1 0 0 1 0 0 1 1

System has No Error
```

```
Enter the Number of bits:7
Enter bit: 1 0 0 1 0 0 1
1. Even Parity
2. Odd parity
select the choice: 1

Enter the received data: 1 1 0 1 0 0 1 1

System has an Error
```

Figure 0-1 1D-Parity

Here the data taken is of 7 bits, the corresponding parity bit is calculated, the receiver then gets the data and checks with the parity bit, here we've taken two examples, in example 1 the data is intact, hence the output is CORRECT, while in the next example the data is mutated and the message outputs INVALID.

```
Enter the 7 bit 2D binary numbers =
1 1 1 0 0 1 0
0 0 1 1 0 1 1
1 1 0 0 1 0 1
1 0 0 1 1 1 0
1 1 1 1 1 1 1
0 0 0 1 0 1 0
1 1 0 0 1 1 1
 the choice, 1 for even parity and 2 for odd parity = 1
Received after addition of Parity bits

11100100
00110110
11001010
10011100
11111111
00010100
11001111
00000000
```

Figure 0-2 2D-Parity

In 2D parity, The parity check is done by adding an extra bit, called parity bit to the data to make a number of 1s either even in case of even parity or odd in case of odd parity. While creating a frame, the sender counts the number of 1s in it and adds the parity bit in the following way • In case of even parity: If a number of 1s is even then parity bit value is 0. If the number of 1s is odd then parity bit value is 1. • In case of odd parity: If a number of 1s is odd then parity bit value is 0. If a number of 1s is even then parity bit value is 1. • On receiving a frame, the receiver counts the number of 1s in it. In case of even parity check, if the count of 1s is even, the frame is accepted, otherwise, it is rejected. A similar rule is adopted for odd parity check.

4. Analysis and Discussions

- **Why can this method not be used to correct errors?**

Parity Bit for error detection is prone to data mutations, such as in 1D-Parity is even number of bits are simultaneously mutated then the parity bits remain unchanged, the receiver will not be able to detect the error in the data. Same goes to 2D-Parity where if the bits in the data grid are mutated in a square region the receiver will not be able to detect corrupted data. Another problem with this method is that the parity bit itself can mutate, which will represent the data to be corrupted even though the data transmitted might be correct.

- **How are 1D and 2D parity different?**

1D Parity has only one parity bit, while 2D parity has multiple bits based on how the data is divided, 2D parity can detect burst errors, while 1D can only detect single bit errors or odd-number of bit errors.

- **What are the limitations of this method of error detection?**

Parity bits cannot detect all forms of bit error

For 2D Parity, rows + column number of parity bits are added that are to be transmitted along with data, more data, means that there's more chances of corruption.

5. Conclusions

Parity bits are a very simple way to check if the transmitted data is corrupted on the way, although it cannot detect all forms of errors, it is very simple to implement.

6. Comments

a. Limitations of the experiment

The experiment assumes the data to be binary values, although in the real world the data is taken as bit streams, the experiment is limited to proof of concept.

b. Limitations of the results obtained

The results obtained can be checked for bitstrings, and does not work on generic data.

c. Learning

The concept of Parity bits (1D and 2D) was learnt in this lab.

d. Recommendations

Instead of using an array/vector to represent the data, the data should be taken as byte streams and encoded accordingly, such as a string, or a binary file, this was the program will be more generic.