

**UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL**

**CURSO DE CIÊNCIA DA COMPUTAÇÃO**

**ANDREI POCHMANN KOENICH**

**MATRÍCULA 00308680**

**SIMULADOR DE RIPPLE CARRY ADDER**

**Projeto de Iniciação Científica**

**Orientador: Paulo Francisco Butzen**

**Porto Alegre, junho de 2022.**

## 1 – CONCEITOS INICIAIS

### 1.1 – Numeração dos bits de um operando binário

Os bits de um operando binário são numerados em ordem crescente a partir do bit menos significativo (ou seja, da direita para a esquerda), iniciando em zero, conforme ilustrado no exemplo abaixo, com um operando de 8 bits.

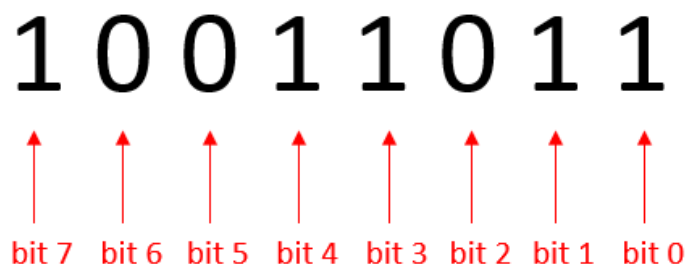


Figura 1 - Numeração dos bits de um número binário

### 1.2 – Corte de carry in

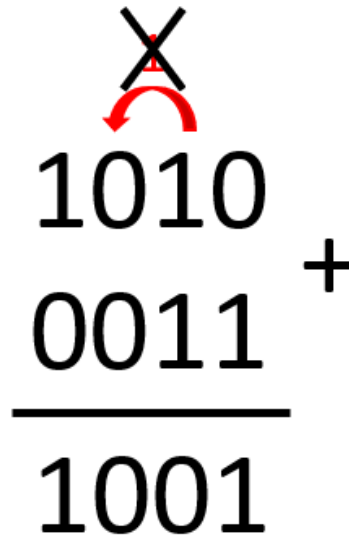
Um corte de carry in é entendido como o desprezo de um determinado carry in em uma soma com dois operandos binários. Para fins de exemplo, considera-se a soma simples abaixo, entre dois operandos binários de 4 bits. A ocorrência de carry in está destacada em vermelho. Nota-se que a ocorrência de um carry out na soma entre os bits da posição 1 dos operandos faz com que ocorra um carry in que deverá ser considerado na soma entre os bits da posição 2, a fim de manter o resultado final correto.

$$\begin{array}{r} 1010 \\ + 0011 \\ \hline 1101 \end{array}$$

Um '1' vermelho com uma seta curva vermelha indica o carry in para a posição 2 da soma.

Figura 2 - Soma binária comum

No próximo exemplo, visualizamos a mesma soma, porém com um corte de carry in no bit 2 dos operandos. Com isso, obtém-se um resultado final diferente.



The diagram shows a binary addition of 1010 and 0011. A red 'X' is placed over the carry-in at bit 2, with a red arrow pointing to the right, indicating that the carry-in is ignored. The result shown is 1001.

$$\begin{array}{r} 1010 \\ + 0011 \\ \hline 1001 \end{array}$$

Figura 3 – Soma binária com corte de carry in

Os resultados binários obtidos nos exemplos das figuras 2 e 3 possuem valor decimal igual a 13 e 9, respectivamente (considerando a representação em inteiros positivos). Portanto, o erro absoluto obtido com o corte de carry in considerado possui valor igual a 4.

### 1.3 – Indução de carry in

Uma indução de carry in é entendida como a inserção de um determinado carry in em uma soma com dois operandos binários. Para fins de exemplo, considera-se a soma simples abaixo, entre dois operandos binários de 4 bits. Conforme já visto, existe uma ocorrência de carry out no bit 1 dos operandos. Entretanto, será inserida uma ocorrência de carry in no bit 1 dos operandos, o que altera o resultado que foi demonstrado na figura 2. A ocorrência de carry out será destacada em vermelho, enquanto que a ocorrência do carry in inserido será destacada em azul.

$$\begin{array}{r} 1010 \\ + 0011 \\ \hline 1111 \end{array}$$

Figura 4 - Soma binária com indução de carry in

Nesse exemplo, o resultado binário obtido com a indução de carry in possui valor decimal igual a 15 (considerando a representação em inteiros positivos). Portanto, tem-se um erro absoluto (em relação ao valor decimal 13, obtido com a operação mostrada na figura 1) igual a 2.

### 1.4 – Erro absoluto obtido com corte ou indução de carry in

O erro absoluto obtido com corte ou indução de carry in é calculado por meio do módulo da diferença entre os valores decimais dos resultados obtidos com corte ou indução de carry in e sem qualquer interferência de carry in.

Comparando os resultados obtidos nas figuras 2 e 3, o erro absoluto é calculado da forma abaixo, com o seguinte resultado decimal:

$$Erro\ Absoluto = |1101_2 - 1001_2| = |13 - 9| = 4$$

Comparando os resultados obtidos nas figuras 2 e 4, o erro absoluto é calculado da forma abaixo, com o seguinte resultado decimal:

$$Erro\ Absoluto = |1101_2 - 1111_2| = |13 - 15| = 2$$

## 2 – MÓDULOS DO PROGRAMA

No início da execução do programa, visualiza-se que ele está dividido em seis módulos, sendo cada um deles descrito a seguir.

```
SIMULADOR DE RIPPLE CARRY ADDER - Andrei Pochmann Koenich
Pressione a tecla para escolher a analise desejada:
1 - Analise com dois operandos especificos.
2 - Cortes de CARRY IN - Analise estatistica com operandos variaveis e um fixo, em uma faixa de valores.
3 - Cortes de CARRY IN - Analise estatistica com operandos aleatorios.
4 - Inducoes de CARRY IN - Analise estatistica com operandos variaveis e um fixo, em uma faixa de valores.
5 - Inducoes de CARRY IN - Analise estatistica com operandos aleatorios.
6 - Mudancas de Portas Logicas - Analise estatistica com operandos aleatorios.
7 - Analise estatistica com as propriedades metricas da IEEE.
ESC - Encerrar programa.
```

Figura 5 - Menu inicial do programa

### 2.1 – Análise com dois operandos específicos

Uma vez que esse módulo é escolhido pelo usuário, o programa irá solicitar o número de bits dos dois operandos (impondo um limite de até 16 bits), além dos valores binários dos operandos em si.

```
SIMULADOR DE RIPPLE CARRY ADDER - Andrei Pochmann Koenich
QUANTIDADE DE BITS
Digite a quantidade de bits dos dois numeros binarios a serem somados (maximo 16 bits):
8_
```

Figura 6 - Solicitação do número de bits ao usuário

```
SIMULADOR DE RIPPLE CARRY ADDER - Andrei Pochmann Koenich
OPERANDOS BINARIOS
Digite o primeiro numero binario de 8 bits:
01010011
Digite o segundo numero binario de 8 bits:
1101011
```

Figura 7 - Solicitação dos dois operandos binários

Depois de inseridas as informações mostradas nas figuras 6 e 7, o programa irá exibir o valor da soma (sem qualquer corte ou indução de carry in) entre os dois operandos (em binário e em decimal), além da quantidade de ocorrências de carry out. O programa também destaca a ocorrência de carry out no bit mais significativo, conforme demonstrado na figura 8.

```
SIMULADOR DE RIPPLE CARRY ADDER - Andrei Pochmann Koenich

RESULTADO DA SOMA

BINARIO:

  01010011 +
  11010110
  -----
  00101001

DECIMAL:

  83 + 214 = 41

Quantos CARRY OUT ocorreram durante a soma?
5

Ocorreu CARRY OUT no bit mais significativo?
SIM

Pressione a tecla correspondente ao resultado desejado:

1 - Resultado com cortes cumulativos de CARRY IN.
2 - Resultado com cortes em todos os CARRY IN.
3 - Resultado com corte somente no CARRY IN mais significativo.
4 - Resultado com cortes de CARRY IN especificos.
5 - Resultado com inducoes de CARRY IN especificas.
6 - Resultado trocando todas as portas logicas XOR por OR no RCA
7 - Resultado trocando todas as portas logicas XOR por AND no RCA
8 - Geracao de arquivo texto com todos os cortes de CARRY IN possiveis.
9 - Realizar nova operacao.
ESC - Encerrar programa.
```

Figura 8 - Resultados, com exibição de novas opções (submódulos)

Uma vez que os resultados comuns das somas entre os dois operandos são exibidos, é possível escolher uma nova operação a ser realizada com os dois operandos, conforme mostra o novo menu exibido na parte inferior. Cada operação será explicada a seguir (também é possível retornar para o menu inicial do simulador mostrado na figura 5, pressionando a tecla 7). Na figura 9, está demonstrada a soma comum realizada com os dois operandos de 8 bits da figura 8, com cada ocorrência de carry in sendo destacada em vermelho.

$$\begin{array}{r}
 \overset{1}{\curvearrowright} \overset{1}{\curvearrowright} \quad \overset{1}{\curvearrowright} \quad \overset{1}{\curvearrowright} \overset{1}{\curvearrowright} \\
 01010011 \\
 + 11010110 \\
 \hline
 00101001
 \end{array}$$

Figura 9 - Soma comum com os dois operandos escolhidos para testes

Para todas as operações (submódulos) explicadas a seguir, os seguintes resultados serão exibidos, para cada cálculo em específico:

- 1) Valor binário do resultado obtido;
- 2) Valor decimal do resultado obtido;
- 3) Erro absoluto do resultado obtido, em comparação com o resultado gerado pela soma comum entre os dois operandos binários escolhidos pelo usuário, sem qualquer interferência de carry in;
- 4) Erro relativo do resultado obtido, em comparação com o resultado gerado pela soma comum entre os dois operandos binários escolhidos pelo usuário, sem qualquer interferência de carry in.

#### 2.1.1 – Resultados com cortes cumulativos de carry in

Nesse submódulo do programa, são demonstrados ao usuário os resultados obtidos ao serem realizadas quantidades cada vez maiores de cortes de carry in nos bits menos significativos dos operandos considerados. Considerando os operandos da figura 9, nas figuras 10, 11 e 12 são demonstradas as operações de soma considerando um número cada vez maior de cortes de carry in (até 3 cortes), até que não seja possível realizar mais nenhum corte que ocasione uma mudança no resultado final da operação.

Conforme esperado, é possível perceber que o corte de um determinado carry in faz com que outras ocorrências de carry in se tornem inexistentes (na figura 10, por exemplo, é possível perceber que o corte do carry in no bit 2 anulou a ocorrência de carry in no bit 3, que era existente na operação demonstrada na figura 9).

$$\begin{array}{r}
 \overset{1}{\curvearrowright} \overset{1}{\curvearrowright} \overset{1}{\curvearrowright} \overset{\times}{\curvearrowright} \\
 01010011 \\
 + 11010110 \\
 \hline
 00100101
 \end{array}$$

Figura 10 – Realização de um corte no carry in menos significativo

$$\begin{array}{r}
 \overset{1}{\curvearrowright} \overset{1}{\curvearrowright} \overset{\times}{\curvearrowright} \overset{\times}{\curvearrowright} \\
 01010011 \\
 + 11010110 \\
 \hline
 00000101
 \end{array}$$

Figura 11 - Realização de dois cortes nos carry in menos significativos

$$\begin{array}{r}
 \overset{\times}{\curvearrowright} \overset{\times}{\curvearrowright} \overset{\times}{\curvearrowright} \\
 01010011 \\
 + 11010110 \\
 \hline
 10000101
 \end{array}$$

Figura 12 - Realização de três cortes, atingindo o limite de cortes possíveis no exemplo considerado

Na figura 13, visualizamos os resultados obtidos com o simulador.

```

RESULTADOS COM CORTES CUMULATIVOS DE CARRY IN:

1 CORTES:
Valor binario: 00100101
Valor decimal: 37
Erro absoluto: 4
Erro relativo: 9.7561

2 CORTES:
Valor binario: 00000101
Valor decimal: 5
Erro absoluto: 36
Erro relativo: 87.8049

3 CORTES:
Valor binario: 10000101
Valor decimal: 133
Erro absoluto: 92
Erro relativo: 224.3902

Pressione qualquer tecla para voltar ao menu anterior.

```

Figura 13 - Resultados obtidos com o primeiro submódulo do programa



### 2.1.2 – Resultados com cortes em todos os carry in

Nesse submódulo, todas as ocorrências de carry in são ignoradas diretamente, conforme mostrado na figura 14.

$$\begin{array}{r} \text{X X} \quad \text{X} \quad \text{X X} \\ \text{01010011} \\ + \\ \text{11010110} \\ \hline \text{10000101} \end{array}$$

Figura 14 - Resultado obtido cortando-se diretamente todos os carry in

```
RESULTADOS DESPREZANDO TODOS OS CARRY IN:  
  
Valor binario: 10000101  
Valor decimal: 133  
Erro absoluto: 100  
Erro relativo: 303.0303  
  
Pressione qualquer tecla para voltar ao menu anterior.
```

Figura 15 - Resultados obtidos com o simulador, ignorando todas as ocorrências de carry in

### 2.1.3 – Resultado com corte somente no carry in mais significativo

Nesse submódulo, é realizado somente um corte no carry in mais significativo, conforme mostrado na figura 16. Nota-se que, no exemplo considerado, o corte no carry in mais significativo anulou a ocorrência de carry out no bit 7 (ou seja, no bit mais significativo).

$$\begin{array}{r} \text{X} \quad 1 \quad 1 \quad 1 \\ \text{01010011} \\ + \\ \text{11010110} \\ \hline \text{10101001} \end{array}$$

Figura 16 - Operação com corte somente no carry in mais significativo

```

RESULTADO DESPREZANDO SOMENTE O CARRY IN MAIS SIGNIFICATIVO:

Valor binario:  10101001
Valor decimal:  169
Erro absoluto:  136
Erro relativo:  412.1212

Pressione qualquer tecla para voltar ao menu anterior.

```

Figura 17 - Resultados obtidos com o simulador, ignorando somente o carry in mais significativo

#### 2.1.4 – Resultado com cortes de carry in específicos

Nesse submódulo, o usuário pode escolher em quais bits em específico deverão ocorrer os cortes de carry in. Para tanto, o programa informa ao usuário em quais bits há ocorrência de carry in para a operação escolhida. Em seguida, o usuário deve digitar os bits em que ocorrerão os cortes, separando-os por espaços ou vírgulas, conforme demonstrado na figura 18. Para o exemplo considerado, sabe-se que ocorreu carry in nos bits 2, 3, 5 e 7, conforme já demonstrado na figura 9, com a operação de soma comum. No teste aqui demonstrado, foram realizados cortes nos bits 5 e 2 (conforme demonstrado na figura 19), obtendo-se o resultado mostrado na figura 20.

```

Ocorreram CARRY IN nos seguintes bits:
7 5 3 2

Digite os bits em que ocorrerão os cortes de CARRY IN, separados por espaços ou vírgulas:
5 2_

```

Figura 18 - Escolha dos cortes de carry in pelo usuário no simulador

$$\begin{array}{r}
 \begin{array}{ccccccc}
 \overset{1}{\curvearrowright} & \overset{1}{\curvearrowright} & \overset{\times}{\curvearrowright} & & \overset{\times}{\curvearrowright} & & \\
 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \\
 + & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\
 \hline
 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1
 \end{array}
 \end{array}$$

Figura 19 - Operação com cortes de carry in nos bits 5 e 2



```
Ocorreram CARRY IN nos seguintes bits:
7 5 3 2

Digite os bits em que ocorrerão as induções, separados por espaços ou vírgulas:
4 0

Valor binário: 00111010
Valor decimal: 58
Erro absoluto: 17
Erro relativo: 41.4634

Pressione qualquer tecla para voltar ao menu anterior.
```

Figura 23 – Resultados obtidos com induções de carry in específicas com o simulador

### 2.1.6 – Resultado trocando todas as portas lógicas XOR por OR no RCA

Nesse submódulo do programa, o simulador executa a operação de soma entre os dois operandos binários escolhidos pelo usuário considerando que o circuito original do Ripple Carry Adder foi modificado, de tal forma que todas as portas lógicas XOR da sua composição foram substituídas por portas lógicas OR.

```
RESULTADO TROCANDO AS PORTAS LOGICAS <XOR> POR PORTAS LOGICAS <OR> NO RCA:

Valor binario: 11110111
Valor decimal: 247
Erro absoluto: 206
Erro relativo: 502.4390

Pressione qualquer tecla para voltar ao menu anterior.
```

Figura 24 – Resultados obtidos com troca de portas lógicas no RCA

### 2.1.7 – Resultado trocando todas as portas lógicas XOR por AND no RCA

Nesse submódulo do programa, o simulador executa a operação de soma entre os dois operandos binários escolhidos pelo usuário considerando que o circuito original do Ripple Carry Adder foi modificado, de tal forma que todas as portas lógicas XOR da sua composição foram substituídas por portas lógicas AND.

```
RESULTADO TROCANDO AS PORTAS LOGICAS <XOR> POR PORTAS LOGICAS <AND> NO RCA:

Valor binario: 00000000
Valor decimal: 0
Erro absoluto: 41
Erro relativo: 100.0000

Pressione qualquer tecla para voltar ao menu anterior.
```

Figura 25 - Resultados obtidos com troca de portas lógicas no RCA

### 2.1.8 – Geração de arquivo texto com todas os cortes de carry in possíveis

Nesse módulo, o simulador possibilita ao usuário a geração (ou atualização) de um arquivo texto de saída denominado “cortes\_somas.txt”, o qual irá conter o resultado da operação de soma sem qualquer corte de carry in e, juntamente, os resultados de somas considerando todas as combinações possíveis de cortes de carry in.

Exemplificando, caso uma determinada operação entre dois operandos binários possua ocorrência de carry in nos bits 1, 3 e 5, serão calculadas oito somas, cada uma considerando um conjunto diferente de ocorrências de carry in, até que todos os conjuntos possíveis tenham sido testados:

**Operação 1:** sem cortes de carry in

**Operação 2:** corte no bit 1

**Operação 3:** corte no bit 3

**Operação 4:** cortes nos bits 3 e 1

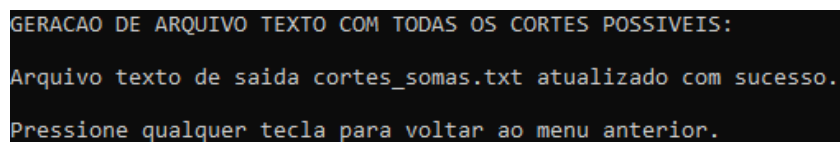
**Operação 5:** corte no bit 5

**Operação 6:** cortes nos bits 5 e 1

**Operação 7:** cortes nos bits 5 e 3

**Operação 8:** cortes nos bits 5, 3 e 1

Destaca-se também que a manipulação do arquivo de saída ocorre no formato *append*, ou seja, caso o arquivo já exista e já contenha informações de uma operação passada, as novas informações serão inseridas no fim do arquivo, conservando o conteúdo já existente. Caso o arquivo não exista, ele será criado.



```
GERACAO DE ARQUIVO TEXTO COM TODAS OS CORTES POSSIVEIS:
Arquivo texto de saida cortes_somas.txt atualizado com sucesso.
Pressione qualquer tecla para voltar ao menu anterior.
```

Figura 26 – Mensagem exibida pelo simulador quando da criação ou atualização do arquivo texto de saída “cortes\_somas.txt”

Nas figuras 27 e 28, visualizamos trechos do arquivo cortes\_somas.txt gerado pelo simulador para os operandos binários dos exemplos já vistos nos itens anteriores.

```

1 -----
2 OPERACAO ANALISADA:
3
4 01010011 +
5 11010110
6 -----
7 00101001
8
9 DECIMAL:
10
11      83 + 214 = 41
12
13
14 BITS COM OCORRENCIA DE CARRY IN:
15 7 5 3 2
16
17 -----
18
19 OPERACAO 1: sem cortes de CARRY IN
20
21 Valor binario: 00101001
22 Valor decimal: 41
23
24 Erro absoluto: 0
25
26 Erro relativo: 0.000000
27
28 -----
29 -----
30
31 OPERACAO 2: cortes nos bits 2
32
33 Valor binario: 00100101
34 Valor decimal: 37
35
36 Erro absoluto: 4
37
38 Erro relativo: 9.756098
39
40 -----
41 -----

```

Figura 27 – Visualização do trecho inicial do arquivo “cortes\_somas.txt”, gerado para os operandos binários dos exemplos anteriores

```

103 OPERACAO 8: cortes nos bits 5 3 2
104
105 Valor binario: 00000101
106 Valor decimal: 5
107
108 Erro absoluto: 36
109
110 Erro relativo: 87.804878
111
112 -----
113 -----
114
115 OPERACAO 9: cortes nos bits 7
116
117 Valor binario: 10101001
118 Valor decimal: 169
119
120 Erro absoluto: 128
121
122 Erro relativo: 312.195122
123
124 -----
125 -----
126
127 OPERACAO 10: cortes nos bits 7 2
128
129 Valor binario: 10100101
130 Valor decimal: 165
131
132 Erro absoluto: 124
133
134 Erro relativo: 302.439024
135
136 -----
137 -----
138
139 OPERACAO 11: cortes nos bits 7 3
140
141 Valor binario: 10100001
142 Valor decimal: 161
143
144 Erro absoluto: 120
145
146 Erro relativo: 292.682927
147
148 -----

```

Figura 28 – Visualização de outro trecho do arquivo “cortes\_somas.txt”, gerado para os operandos binários dos exemplos anteriores

## 2.2 – Cortes de carry in – Análise estatística com operandos variáveis e um fixo, em uma faixa de valores

Esse módulo tem como objetivo realizar um conjunto de somas entre um operando binário variável que possui um valor mínimo e incrementa seu valor em uma unidade (até que seja atingido um valor máximo) e um operando binário fixo. Em cada soma, será calculado o erro absoluto obtido por meio de cortes fixos de carry in que serão realizados em cada operação. Os bits nos quais deverão ocorrer esses cortes de carry in serão escolhidos pelo usuário.

Exemplificando, caso o usuário escolha os valores binários 000 e 111 como valores mínimo e máximo para o operando variável (respectivamente), e atribua o valor 010 para o operando fixo, serão realizadas as operações descritas abaixo (cada operação individual está agrupada em parênteses, e as operações estão separadas por vírgulas. O operando variável está destacado em negrito, em cada operação):

(**000** + 010), (**001** + 010), (**010** + 010), (**011** + 010), (**100** + 010), (**101** + 010), (**110** + 010), (**111** + 010)

Para cada soma, será realizada a operação sem e com cortes de carry in (de acordo com os bits escolhidos pelo usuário para realizar os cortes), armazenando o erro absoluto individual obtido, dessa forma tornando possível calcular a média aritmética e o desvio padrão dos erros absolutos gerados em cada operação.

Uma vez que esse módulo é escolhido pelo usuário, o programa irá solicitar o número de bits dos dois operandos (impondo um limite de até 16 bits), além dos valores binários mínimo e máximo do operando variável e do valor do operando binário fixo e os bits nos quais ocorrerão os cortes de carry in.

```
SIMULADOR DE RIPPLE CARRY ADDER - Andrei Pochmann Koenich
QUANTIDADE DE BITS
Digite a quantidade de bits dos dois numeros binarios a serem somados (maximo 16 bits):
8_
```

Figura 29 – Escolha do número de bits dos operandos do conjunto de somas a serem realizadas

```
SIMULADOR DE RIPPLE CARRY ADDER - Andrei Pochmann Koenich
OPERANDOS BINARIOS
Digite o numero binario minimo de 8 bits:
00101011
Digite o segundo numero binario maximo de 8 bits:
1101011_
```

Figura 30 – Escolha dos valores binários mínimo e máximo do operando variável

```

SIMULADOR DE RIPPLE CARRY ADDER - Andrei Pochmann Koenich

PONTOS DE INTERFERENCIA

Digite os bits em que ocorrerão os cortes de CARRY IN, separados por espaços ou vírgulas:
2 5_

```

Figura 31 – Solicitação dos bits em que ocorrerão os cortes de carry in, para cada soma parcial

```

SIMULADOR DE RIPPLE CARRY ADDER - Andrei Pochmann Koenich

OPERANDO BINARIO FIXO

Digite o operando binario fixo de 8 bits:
0101011

```

Figura 32 – Solicitação do operando binário que irá ser mantido fixo em todas as operações

Ao fim das operações, programa irá indicar as seguintes informações, conforme indicado na figura 33:

- 1) Valor decimal do intervalo numérico escolhido para o operando variável, bem como o valor decimal do operando binário fixo escolhido;
- 2) Bits nos quais ocorreram cortes de carry in fixos durante todas as somas;
- 3) Quantidade de somas realizadas;
- 4) Quantidade de vezes em que o resultado de uma soma com cortes de carry in foi diferente do resultado exato;
- 5) Média aritmética dos erros absolutos durante as somas;
- 6) Desvio padrão dos erros absolutos durante as somas.

```

SIMULADOR DE RIPPLE CARRY ADDER - Andrei Pochmann Koenich

RESULTADOS ESTATISTICOS DAS SOMAS COM CORTES DE CARRY IN:

Somas com 86 ocorridas no intervalo de inteiros:
[ 43 , 215 ]

Durante as somas, ocorreram cortes de CARRY IN fixos nos bits abaixo:
2 5

Quantidade de somas realizadas:
173

Quantidade de vezes em que o resultado de uma soma com interferencia foi diferente do resultado exato:
143

Media aritmetica dos erros absolutos durante as somas:
47.3295

Desvio padrao dos erros absolutos durante as somas:
69.5675

Pressione a tecla para escolher a proxima acao:

X - Anexar os resultados em um arquivo texto de saida.
ESC - Encerrar programa.

Pressione qualquer outra tecla para realizar uma nova operacao.

```

Figura 33 – Exemplo de resultados obtidos com o segundo módulo do simulador



Adicionalmente, é possível anexar os resultados em um arquivo texto de saída denominado “cortes\_faixavalores.txt” ao pressionar a tecla “X”, novamente em formato “*append*”, conforme mostra a figura 34 e 35.

```
Arquivo texto de saida cortes_faixavalores.txt atualizado com sucesso.  
Pressione qualquer tecla para retornar para a tela anterior.
```

Figura 34 – Mensagem exibida pelo simulador quando da criação ou atualização do arquivo texto de saída “cortes\_faixavalores.txt”

```
1 -----  
2  
3 Somas com 86 ocorridas no intervalo de inteiros:  
4 [ 43 , 215 ]  
5  
6 Durante as somas, ocorreram cortes de CARRY IN fixos nos bits abaixo:  
7 2 5  
8  
9 Quantidade de somas realizadas:  
10 173  
11  
12 Quantidade de vezes em que o resultado de uma soma com interferencia foi diferente do resultado exato:  
13 143  
14  
15 Media aritmetica dos erros absolutos durante as somas:  
16 47.3295  
17  
18 Desvio padrao dos erros absolutos durante as somas:  
19 69.5675  
20  
21 -----
```

Figura 35 – Exemplo de trecho de um arquivo “cortes\_faixavalores.txt”, armazenando as informações obtidas, mostradas na figura 31

## 2.3 – Cortes de carry in – Análise estatística com operandos aleatórios

Nesse módulo do programa, o usuário deve escolher uma quantidade de bits (menor ou igual a 16 bits), além de pontos fixos de cortes de carry in e um número total de operações a serem realizadas (que deve ser menor ou igual a dez mil operações). Com isso, o programa irá gerar pares operandos binários aleatórios (com uma quantidade de bits igual à escolhida) e, para cada par de operandos, irá realizar a soma com e sem os cortes de carry in nos bits escolhidos pelo usuário (a quantidade de somas realizadas corresponde ao número total de operações escolhido pelo usuário), armazenando o erro absoluto obtido em cada operação, fazendo com que seja possível mostrar, ao final, a média aritmética e o desvio padrão dos erros absolutos gerados em cada operação.

Exemplificando, caso o usuário escolha operandos de 4 bits, o programa irá gerar pares de operandos binários com valores maiores ou iguais a 0000 (menor operando de 4 bits possível) e menores ou iguais a 1111 (maior operando de 4 bits possível).

```
SIMULADOR DE RIPPLE CARRY ADDER - Andrei Pochmann Koenich  
  
QUANTIDADE DE BITS  
  
Digite a quantidade de bits dos operandos das somas (maximo 16 bits):  
8_
```

Figura 36 – Escolha da quantidade de bits dos pares de operandos aleatórios a serem gerados

```

SIMULADOR DE RIPPLE CARRY ADDER - Andrei Pochmann Koenich

PONTOS DE INTERFERENCIA

Digite os bits em que ocorrerão os cortes de CARRY IN, separados por espaços ou vírgulas:
1 6

```

Figura 37 – Escolha dos pontos em que ocorrerão os cortes fixos de carry in, nas operações com operandos aleatórios

```

SIMULADOR DE RIPPLE CARRY ADDER - Andrei Pochmann Koenich

DEFINICAO DO NUMERO DE OPERACOES:

Intervalo de operacoes:
[0 , 256)

Digite o numero total de operacoes a serem realizadas (maximo 10000 operacoes):
7777

```

Figura 38 – Escolha do número total de operações que serão realizadas com operandos binários aleatórios

Ao fim das operações, programa irá indicar as seguintes informações, conforme indicado na figura 39:

- 1) Valor decimal do intervalo numérico para a quantidade de bits escolhida;
- 2) Bits nos quais ocorreram cortes de carry in fixos durante todas as somas;
- 3) Quantidade de somas realizadas;
- 4) Quantidade de vezes em que o resultado de uma soma com corte de carry in foi diferente do resultado exato;
- 5) Média aritmética dos erros absolutos durante as somas;
- 6) Desvio padrão dos erros absolutos durante as somas.

```

SIMULADOR DE RIPPLE CARRY ADDER - Andrei Pochmann Koenich

RESULTADOS ESTATISTICOS DAS SOMAS ALEATORIAS COM CORTES DE CARRY IN:

Intervalo de operacoes:
[0 , 256)

Durante as somas, ocorreram cortes de CARRY IN fixos nos bits abaixo:
1 6

Quantidade de somas realizadas:
7777

Quantidade de vezes em que o resultado de uma soma com interferencia foi diferente do resultado exato:
4781

Media aritmetica dos erros absolutos durante as somas:
47.8534

Desvio padrao dos erros absolutos durante as somas:
62.0782

Pressione a tecla para escolher a proxima acao:

X - Anexar os resultados em um arquivo texto de saida.

ESC - Encerrar programa.

Pressione qualquer outra tecla para realizar uma nova operacao.

```

Figura 39 – Exemplo de resultados obtidos com o terceiro módulo do simulador

Adicionalmente, é possível anexar os resultados em um arquivo texto de saída denominado “somascortes\_aleatorias.txt” ao pressionar a tecla “X”, novamente em formato “*append*”, conforme mostram as figuras 40 e 41.

```
Arquivo texto de saida somascortes_aleatorias.txt atualizado com sucesso.  
Pressione qualquer tecla para retornar para a tela anterior.
```

Figura 40 – Mensagem exibida pelo simulador quando da criação ou atualização do arquivo texto de saída “somascortes\_aleatorias.txt”

```
1 -----  
2 OPERACAO ANALISADA:  
3  
4 Foram realizadas 7777 somas com operandos aleatorios de 8 bits, no intervalo [0 , 256)  
5  
6 Durante as somas, ocorreram cortes de CARRY IN fixos nos bits abaixo:  
7 1 6  
8  
9 Quantidade de somas realizadas:  
10 7777  
11  
12 Quantidade de vezes em que o resultado de uma soma com interferencia foi diferente do resultado exato:  
13 4754  
14  
15 Media aritmetica dos erros absolutos durante as somas:  
16 47.7125  
17  
18 Desvio padrao dos erros absolutos durante as somas:  
19 62.1810  
20  
21 -----
```

Figura 41 – Exemplo de trecho de um arquivo “somascortes\_aleatorias.txt”, armazenando as informações obtidas, mostradas na figura 31

## 2.4 – Induções de carry in - Análise estatística com operandos variáveis e um fixo, em uma faixa de valores.

Esse módulo tem como objetivo realizar um conjunto de somas entre um operando binário variável que possui um valor mínimo e incrementa seu valor em uma unidade (até que seja atingido um valor máximo) e um operando binário fixo. Para cada soma, será calculado o erro absoluto obtido por meio de induções fixas de carry in que serão realizadas em cada operação. Os bits nos quais deverão ocorrer essas induções de carry in serão escolhidos pelo usuário.

Para cada soma, será realizada a operação sem e com induções de carry in (de acordo com os bits escolhidos pelo usuário para realizar as induções), armazenando o erro absoluto individual obtido, dessa forma tornando possível calcular a média aritmética e o desvio padrão dos erros absolutos gerados em cada operação.

Uma vez que esse módulo é escolhido pelo usuário, o programa irá solicitar o número de bits dos dois operandos (impondo um limite de até 16 bits), além dos valores binários mínimo e máximo do operando variável e do valor do operando binário fixo e os bits nos quais ocorrerão os cortes de carry in.

```

SIMULADOR DE RIPPLE CARRY ADDER - Andrei Pochmann Koenich

QUANTIDADE DE BITS

Digite a quantidade de bits dos dois numeros binarios a serem somados (maximo 16 bits):
8_

```

Figura 42 – Escolha do número de bits dos operandos do conjunto de somas a serem realizadas

```

SIMULADOR DE RIPPLE CARRY ADDER - Andrei Pochmann Koenich

OPERANDOS BINARIOS

Digite o numero binario minimo de 8 bits:
00101011

Digite o segundo numero binario maximo de 8 bits:
1101011_

```

Figura 43 – Escolha dos valores binários mínimo e máximo do operando variável

```

SIMULADOR DE RIPPLE CARRY ADDER - Andrei Pochmann Koenich

PONTOS DE INDUCAO:

Digite os bits em que ocorrerao as inducoes, separados por espacos ou virgulas:
1 6_

```

Figura 44 – Solicitação dos bits em que ocorrerão as induções de carry in, para cada soma parcial

```

SIMULADOR DE RIPPLE CARRY ADDER - Andrei Pochmann Koenich

OPERANDO BINARIO FIXO

Digite o operando binario fixo de 8 bits:
0101011

```

Figura 45 – Solicitação do operando binário que irá ser mantido fixo em todas as operações

Ao fim das operações, programa irá indicar as seguintes informações, conforme indicado na figura 46:

- 1) Valor decimal do intervalo numérico escolhido para o operando variável, bem como o valor decimal do operando binário fixo escolhido;
- 2) Bits nos quais ocorreram induções de carry in fixas durante todas as somas;
- 3) Quantidade de somas realizadas;
- 4) Quantidade de vezes em que o resultado de uma soma com induções de carry in foi diferente do resultado exato;
- 5) Média aritmética dos erros absolutos durante as somas;
- 6) Desvio padrão dos erros absolutos durante as somas.

```

SIMULADOR DE RIPPLE CARRY ADDER - Andrei Pochmann Koenich
RESULTADOS ESTATISTICOS DAS SOMAS COM INDUCOES DE CARRY IN:

Somos com 86 ocorridas no intervalo de inteiros:
[ 43 , 215 ]

Durante as somas, ocorreram inducoes de CARRY IN fixas nos bits abaixo:
1 6

Quantidade de somas realizadas:
173

Quantidade de vezes em que o resultado de uma soma com interferencia foi diferente do resultado exato:
173

Media aritmetica dos erros absolutos durante as somas:
72.0578

Desvio padrao dos erros absolutos durante as somas:
74.4675

Pressione a tecla para escolher a proxima acao:

X - Anexar os resultados em um arquivo texto de saida.

ESC - Encerrar programa.

Pressione qualquer outra tecla para realizar uma nova operacao.

```

Figura 46 – Exemplo de resultados obtidos com o quarto módulo do simulador

Adicionalmente, é possível anexar os resultados em um arquivo texto de saída denominado “inducoes\_faixavalores.txt” ao pressionar a tecla “X”, novamente em formato “*append*”, conforme mostram as figuras 47 e 48.

```

Arquivo texto de saida inducoes_faixavalores.txt atualizado com sucesso.
Pressione qualquer tecla para retornar para a tela anterior.

```

Figura 47 – Mensagem exibida pelo simulador quando da criação ou atualização do arquivo texto de saída “inducoes\_faixavalores.txt”

```

1 -----
2
3 Somas com 86 ocorridas no intervalo de inteiros:
4 [ 43 , 215 ]
5
6 Durante as somas, ocorreram inducoes de CARRY IN fixas nos bits abaixo:
7 1 6
8
9 Quantidade de somas realizadas:
10 173
11
12 Quantidade de vezes em que o resultado de uma soma com interferencia foi diferente do resultado exato:
13 173
14
15 Media aritmetica dos erros absolutos durante as somas:
16 72.0578
17
18 Desvio padrao dos erros absolutos durante as somas:
19 74.4675
20
21 -----

```

Figura 48 – Exemplo de trecho de um arquivo “inducoes\_faixavalores.txt”, armazenando as informações obtidas, mostradas na figura 44

## 2.5 – Induções de carry in - Análise estatística com operandos aleatórios

Nesse módulo do programa, o usuário deve escolher uma quantidade de bits (menor ou igual a 16 bits), além de pontos fixos de induções de carry in e um número total de operações a serem realizadas (que deve ser menor ou igual a dez mil operações). Com isso, o programa irá gerar pares de operandos binários aleatórios (com uma quantidade de bits igual à escolhida) e, para cada par de operandos, irá realizar a soma com e sem as induções de carry in nos bits escolhidos pelo usuário (a quantidade de somas realizadas corresponde ao número total de operações escolhido pelo usuário), armazenando o erro absoluto obtido em cada operação, fazendo com que seja possível mostrar, ao final, a média aritmética e o desvio padrão dos erros absolutos gerados em cada operação.

```
SIMULADOR DE RIPPLE CARRY ADDER - Andrei Pochmann Koenich
QUANTIDADE DE BITS
Digite a quantidade de bits dos operandos das somas (maximo 16 bits):
10_
```

Figura 49 – Escolha da quantidade de bits dos pares de operandos aleatórios a serem gerados

```
SIMULADOR DE RIPPLE CARRY ADDER - Andrei Pochmann Koenich
PONTOS DE INDUCAO:
Digite os bits em que ocorrerao as inducoes, separados por espacos ou virgulas:
3 8_
```

Figura 50 – Solicitação dos bits em que ocorrerão as induções de carry in, para cada soma parcial

```
SIMULADOR DE RIPPLE CARRY ADDER - Andrei Pochmann Koenich
DEFINICAO DO NUMERO DE OPERACOES:
Intervalo de operacoes:
[0 , 1024)
Digite o numero total de operacoes a serem realizadas (maximo 10000 operacoes):
1337_
```

Figura 51 – Escolha do número total de operações que serão realizadas com operandos binários aleatórios

Ao fim das operações, programa irá indicar as seguintes informações, conforme indicado na figura 52:

- 1) Valor decimal do intervalo numérico para a quantidade de bits escolhida;
- 2) Bits nos quais ocorreram induções de carry in fixas durante todas as somas;
- 3) Quantidade de somas realizadas;
- 4) Quantidade de vezes em que o resultado de uma soma com corte de carry in foi diferente do resultado exato;
- 5) Média aritmética dos erros absolutos durante as somas;
- 6) Desvio padrão dos erros absolutos durante as somas.

```
SIMULADOR DE RIPPLE CARRY ADDER - Andrei Pochmann Koenich
RESULTADOS ESTATISTICOS DAS SOMAS ALEATORIAS COM INDUCOES DE CARRY IN:
Intervalo de operacoes:
[0 , 1024)
Durante as somas, ocorreram inducoes de CARRY IN fixas nos bits abaixo:
3 8
Quantidade de somas realizadas:
1337
Quantidade de vezes em que o resultado de uma soma com interferencia foi diferente do resultado exato:
1336
Media aritmetica dos erros absolutos durante as somas:
191.1025
Desvio padrao dos erros absolutos durante as somas:
253.2539
Pressione a tecla para escolher a proxima acao:
X - Anexar os resultados em um arquivo texto de saida.
ESC - Encerrar programa.
Pressione qualquer outra tecla para realizar uma nova operacao.
```

Figura 52 – Exemplo de resultados obtidos com o quinto módulo do simulador

Adicionalmente, é possível anexar os resultados em um arquivo texto de saída denominado “somasinducoes\_aleatorias.txt” ao pressionar a tecla “X”, novamente em formato “*append*”, conforme mostram as figuras 53 e 54.

```
Arquivo texto de saida somasinducoes_aleatorias.txt atualizado com sucesso.
Pressione qualquer tecla para retornar para a tela anterior.
```

Figura 53 – Mensagem exibida pelo simulador quando da criação ou atualização do arquivo texto de saída “somasinducoes\_aleatorias.txt”

```

1 -----
2 OPERACAO ANALISADA:
3
4 Foram realizadas 1337 somas com operandos aleatorios de 10 bits, no intervalo [0 , 1024)
5
6 Durante as somas, ocorreram inducoes de CARRY IN fixas nos bits abaixo:
7 3 8
8
9 Quantidade de somas realizadas:
10 1337
11
12 Quantidade de vezes em que o resultado de uma soma com interferencia foi diferente do resultado exato:
13 1336
14
15 Media aritmetica dos erros absolutos durante as somas:
16 191.1025
17
18 Desvio padrao dos erros absolutos durante as somas:
19 253.2539
20
21 -----

```

Figura 54 – Exemplo de trecho de um arquivo “somasinducoes\_aleatorias.txt”, armazenando as informações obtidas, mostradas na figura 52

## 2.6 – Mudanças de Portas Lógicas – Análise Estatística com Operandos Aleatórios

Nesse módulo do programa, o usuário deve escolher uma quantidade de bits (menor ou igual a 16 bits), uma porta lógica (OR ou AND) que irá substituir todas as portas lógicas XOR do Ripple Carry Adder e, por fim, um número total de operações a serem realizadas (que deve ser menor ou igual a dez mil operações). Com isso, o programa irá gerar pares operandos binários aleatórios (com uma quantidade de bits igual à escolhida) e, para cada par de operandos, irá realizar a soma com e sem as mudanças de portas lógicas (a quantidade de somas realizadas corresponde ao número total de operações escolhido pelo usuário), armazenando o erro absoluto obtido em cada operação, fazendo com que seja possível mostrar, ao final, a média aritmética e o desvio padrão dos erros absolutos gerados em cada operação.

```

SIMULADOR DE RIPPLE CARRY ADDER - Andrei Pochmann Koenich
QUANTIDADE DE BITS
Digite a quantidade de bits dos dois numeros binarios a serem somados (maximo 16 bits):
8_

```

Figura 55 – Escolha do número de bits dos operandos do conjunto de somas a serem realizadas

```

SIMULADOR DE RIPPLE CARRY ADDER - Andrei Pochmann Koenich
ESCOLHA DE PORTA LOGICA
Escolha qual porta logica substituir as portas XOR no Ripple Carry Adder:
1 - Porta logica OR
2 - Porta logica AND

```

Figura 56 – Escolha das portas lógicas que substituirão todas as ocorrências da porta XOR no RCA



```

SIMULADOR DE RIPPLE CARRY ADDER - Andrei Pochmann Koenich
DEFINICAO DO NUMERO DE OPERACOES:
Intervalo de valores dos operandos:
[0 , 256)
Digite o numero total de operacoes a serem realizadas (maximo 10000 operacoes):
1234

```

Figura 57 – Escolha do número total de operações que serão realizadas com operandos binários aleatórios

```

SIMULADOR DE RIPPLE CARRY ADDER - Andrei Pochmann Koenich
RESULTADOS ESTATISTICOS DAS SOMAS ALEATORIAS COM MUDANCA DE PORTA LOGICA
Intervalo de valores dos operandos:
[0 , 256)
As portas logicas XOR do Ripple Carry Adder foram substituidas por portas OR.

Quantidade de somas realizadas:
1234

Quantidade de vezes em que o resultado de uma soma com interferencia foi diferente do resultado exato:
1226

Media aritmetica dos erros absolutos durante as somas:
91.3055

Desvio padrao dos erros absolutos durante as somas:
80.2213

Pressione a tecla para escolher a proxima acao:
X - Anexar os resultados em um arquivo texto de saida.
ESC - Encerrar programa.
Pressione qualquer outra tecla para realizar uma nova operacao.

```

Figura 58 – Exemplo de resultados obtidos alterando as portas XOR por OR no RCA

```

SIMULADOR DE RIPPLE CARRY ADDER - Andrei Pochmann Koenich
RESULTADOS ESTATISTICOS DAS SOMAS ALEATORIAS COM MUDANCA DE PORTA LOGICA
Intervalo de valores dos operandos:
[0 , 256)
As portas logicas XOR do Ripple Carry Adder foram substituidas por portas AND.

Quantidade de somas realizadas:
1234

Quantidade de vezes em que o resultado de uma soma com interferencia foi diferente do resultado exato:
1230

Media aritmetica dos erros absolutos durante as somas:
97.2042

Desvio padrao dos erros absolutos durante as somas:
80.8769

Pressione a tecla para escolher a proxima acao:
X - Anexar os resultados em um arquivo texto de saida.
ESC - Encerrar programa.
Pressione qualquer outra tecla para realizar uma nova operacao.

```

Figura 59 – Exemplo de resultados obtidos alterando as portas XOR por OR no RCA

Adicionalmente, é possível anexar os resultados em um arquivo texto de saída denominado “somasaleatorias\_mudaportas.txt” ao pressionar a tecla “X”, novamente em formato “*append*”, conforme mostram as figuras 60 e 61.

```
Arquivo texto de saida somasaleatorias_mudaportas.txt atualizado com sucesso.  
Pressione qualquer tecla para retornar para a tela anterior.
```

Figura 60 – Mensagem exibida pelo simulador quando da criação ou atualização do arquivo texto de saída “somasaleatorias\_mudaportas.txt”

```
1 -----  
2 OPERACAO ANALISADA:  
3  
4 Foram realizadas 1234 somas com operandos aleatorios de 8 bits, no intervalo [0 , 256)  
5  
6 As portas logicas XOR do Ripple Carry Adder foram substituidas por portas AND.  
7  
8  
9 Quantidade de somas realizadas:  
10 1234  
11  
12 Quantidade de vezes em que o resultado de uma soma com interferencia foi diferente do resultado exato:  
13 1230  
14  
15 Media aritmetica dos erros absolutos durante as somas:  
16 97.2042  
17  
18 Desvio padrao dos erros absolutos durante as somas:  
19 80.8769  
20  
21 -----
```

Figura 61 – Exemplo de trecho de um arquivo “somasaleatorias\_mudaportas.txt”, armazenando as informações obtidas, mostradas na figura 59

## 2.7 – Análise estatística com as propriedades métricas da IEEE

As propriedades métricas em questão estão listadas no artigo da IEEE denominado “Approximate Arithmetic Circuits: A Survey, Characterization, and Recent Applications”. As formas como são realizados os cálculos de cada uma dessas propriedades pelo simulador serão explicadas a seguir.

Para que sejam realizados os cálculos, o usuário deve inserir a quantidade de bits dos operandos (que deve ser menor ou igual a 16 bits), um operando binário fixo e os bits nos quais ocorrerão cortes de carry in fixos, durante as operações. Conforme mostram as figuras 62, 63 e 64.

```
SIMULADOR DE RIPPLE CARRY ADDER - Andrei Pochmann Koenich
QUANTIDADE DE BITS
Digite a quantidade de bits dos operandos das somas (maximo 16 bits):
12_
```

Figura 62 – Escolha do número de bits dos operandos binários a serem considerados

```
SIMULADOR DE RIPPLE CARRY ADDER - Andrei Pochmann Koenich
OPERANDO BINARIO FIXO
Digite o operando binario fixo de 12 bits:
01101010101_
```

Figura 63 – Escolha do operando binário fixo a ser considerado nas operações

```
SIMULADOR DE RIPPLE CARRY ADDER - Andrei Pochmann Koenich
PONTOS DE INTERFERENCIA
Digite os bits em que ocorrerao os cortes de CARRY IN, separados por espacos ou virgulas:
10 5 7_
```

Figura 64 – Solicitação dos bits em que ocorrerão os cortes de carry in, durante as operações

Para as quatro operações explicadas a seguir, novamente ocorre a consideração de um conjunto de erros absolutos, além do erro relativo, para cada par de operandos. No contexto do simulador, é realizado um conjunto de somas envolvendo todos os operandos binários possíveis (que possuem uma quantidade de bits igual à inserida pelo usuário) e o operando fixo inserido pelo usuário, obtendo-se um conjunto de erros absolutos, de forma similar às operações dos módulos já apresentados.

Exemplificando, caso o usuário escolha operandos de 4 bits e escolha o número 1010 como operando fixo, o programa irá considerar o seguinte conjunto de operandos, para calcular cada um dos quatro parâmetros do artigo (o parâmetro variável está destacado em negrito):

(**0000** + 1010) , (**0001** + 1010), (**0010** + 1010), (**0011** + 1010), ... ,(**1111** + 1010)

### 2.7.1 - MED - Mean Error Distance

$$MED = \sum_{i=1}^N ED_i \cdot P(ED_i)$$

Figura 65 – Apresentação da fórmula de cálculo do parâmetro MED no artigo da IEEE

Nessa operação, considera-se ED como o erro absoluto obtido da forma já mencionada, para cada par de operandos diferente. O parâmetro  $P(ED_i)$ , por sua vez, representa a probabilidade de ocorrência de um determinado operando binário. Assumindo uma distribuição uniforme, o simulador considera essa probabilidade como o inverso da quantidade de números binários representáveis com a quantidade de bits escolhida pelo usuário. Exemplificando, caso o usuário escolha 4 bits, o valor dessa probabilidade será 1/16, tendo em vista que existem 16 números binários distintos, compostos por 4 bits

### 2.7.2 - MRED - Mean Relative Error Distance

$$MRED = \sum_{i=1}^N RED_i \cdot P(RED_i)$$

Figura 66 – Apresentação da fórmula de cálculo do parâmetro MRED no artigo da IEEE

Nessa operação, considera-se RED como o erro relativo obtido considerando-se o resultado de uma soma entre dois operandos binários, realizada com e sem cortes de carry in (ou seja, divide-se o erro absoluto pelo resultado da soma sem qualquer corte de carry in, gerando um conjunto de erros relativos). O parâmetro  $P(RED_i)$  representa, novamente, a probabilidade de ocorrência de um determinado operando binário, assumindo-se uma distribuição uniforme. Com isso, soma-se o resultado do produto de cada erro relativo com a probabilidade ocorrência, obtendo-se o valor final do parâmetro MRED.

Cabe ressaltar que, em alguns casos, a soma entre dois operandos binários pode fazer com que o resultado obtido sem qualquer interferência de carry in possa ser igual a zero, em razão da ausência da quantidade de bits necessários para a representação do resultado exato (*overflow*). Exemplificando, caso os operandos envolvidos na soma sejam 111 e 001, o resultado obtido (considerando-se somente 3 bits de representação) inevitavelmente seria igual a 000 (zero), o que

causaria uma divisão por zero, quando do cálculo do erro absoluto. A fim de contornar esses casos e evitar divisões por zero, o simulador considera, para esses casos, que a soma resultou em um, ao invés de zero.

### 2.7.3 - MSE - Mean Squared Error

$$\text{MSE} = \sum_{i=1}^N \text{ED}_i^2 \cdot P(\text{ED}_i)$$

Figura 67 – Apresentação da fórmula de cálculo do parâmetro MSE no artigo da IEEE

Nessa operação, realiza-se o produto entre o quadrado do erro absoluto obtido (para cada par de operandos) e a probabilidade de ocorrência de uma determinada entrada, obtendo-se um conjunto de valores que são somados, gerando o valor final do parâmetro MSE.

### 2.7.4 - RMSE - Root Mean Square Error

$$\text{RMSE} = \sqrt{\text{MSE}}.$$

Figura 68 – Apresentação da fórmula de cálculo do parâmetro RMSE no artigo da IEEE

Nessa última operação, o valor do parâmetro RMSE é obtido por meio da raiz quadrada do parâmetro MSE, conforme indicado na figura 68.

### 3 – OBSERVAÇÕES FINAIS

- O simulador foi criado com a utilização da linguagem de programação C;

- Para todas as operações realizadas pelo simulador, o programa apresenta robustez, isto é, caso o usuário insira alguma informação inválida (por exemplo, inserir operandos não-binários para as operações de soma, ou escolher uma quantidade de operações ou de bits de operandos acima do permitido), tal informação não será aceita pelo programa, e o usuário será informado sobre essa invalidez;

- Os trechos de arquivos de texto demonstrados ao longo desse relatório foram visualizados com a utilização de Notepad++;

- O código-fonte do programa, assim como o artigo “Approximate Arithmetic Circuits: A Survey, Characterization, and Recent Applications” de autoria da IEEE e referenciado no item 2.7 podem ser encontrados no repositório disponível no link abaixo:

<https://github.com/AndreiKoenich/Simulador-Ripple-Carry-Adder>