UNIVERSIDADE CATÓLICA DE PELOTAS
Mestrado em Engenharia Eletrônica e Computação - PGEEC

SOMADORES COMPLETOS DE 1 BIT BINÁRIO E GRAY

(TRABALHO 1)

PEDRO A. TAVARES

DISCIPLINA: Introdução ao Projeto de Circuitos Integrados
PROFESSOR: Eduardo César da Costa

Pelotas, julho de 2019

Índice

1. TRABALHO PROPOSTO

Implementar na ferramenta Spice Opus circuitos somadores completos de 1 bit nas codificações Binária e Gray.

Realizar os circuitos com o menor número de portas lógicas possível. Comparar os dois circuitos em termos de números de transistores.

2. MÉTODO e FERRAMENTAS UTILIZADAS

Para a realização do trabalho foram utilizados dois softwares: o Autodesk Eagle para a construção do esquemático dos circuitos somadores; e o Spice Opus para fazer a simulação destes circuitos.

2.1 Autodesk EAGLE Premium v9.5.1

O *Autodesk EAGLE* é um software que visa simplificar o projeto e confecção de circuitos e PCB's. Ele já possui uma série de componentes eletrônicos pré-instalados e permite a instalação de novas bibliotecas. Para este trabalho foi utilizada a biblioteca de portas lógicas *ngspice-digital*, em conjunto com o *plugin* do *Spice Opus*, que pode ser instalado através deste link.

2.2 Spice Opus

O *Spice Opus* é um software livre desenvolvido pela Faculdade de Engenharia Elétrica da Universidade de Ljubljana da Slovenia e tem como objetivo facilitar a simulação de circuitos através da interpretação de arquivos codificados na linguagem *spice*.

2.3 Metodologia

1. Para cada circuito, foi elaborada sua tabela verdade e, em seguida, foi gerada a equação de saída através de simplificações aplicando álgebra *booleana,* obtendo portanto o circuito com o menor número de portas lógicas.

2. Através das equações de saída, foi elaborado o esquemático do circuito no software *Autodesk EAGLE*, utilizando as portas lógicas da biblioteca *ngspice-digital*.

3. Com o circuito montado, a *netlist* (arquivo .cir) era exportada. Esse arquivo contém toda a pinagem, nós e interligação de elementos do circuito.

4. Através do console do *EAGLE*, executou-se o comando do *plugin* do *Spice Opus* para exportar um novo *netlist* para ser incluído no arquivo de simulação *spice*.

A execução do *plugin* foi feita através do seguinte comando:

run spice.ulp C:\dev\ arquivo_defs.cir arquivo_exp.cir

Onde:

- "C:\dev\": pasta onde estão os dois arquivos que serão lidos pelo *plugin*;
- "arquivo_defs.cir": arquivo do *netlist* padrão do *EAGLE*, descrito anteriormente no item 3;
- "arquivo_exp.cir": arquivo do *netlist* atualizado que será utilizado para a simulação no *Spice Opus*. (Obs: este arquivo deve ser criado anteriormente pelo usuário antes da execução do comando, caso contrário, ocorrerá um erro de exportação no *plugin*);

5. Geração do código *spice* e inclusão do circuito gerado no item 4 como um sub-circuito no *script* de simulação;

6. Simulação do circuito no *Spice Opus* plotando a saída em um gráfico no tempo e comprovando se o mesmo obteve o comportamento esperado.

3. SOMADOR COMPLETO DE 1 BIT (BINÁRIO)

O somador completo de 1-bit binário possui a seguinte tabela verdade:

Variáveis de entrada			**Saídas do somador**	
CIN	**B**	**A**	**COUT**	**S**
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Para chegarmos às equações das saídas "S" e *carry out* (COUT) do somador, foram aplicadas simplificações de álgebra *booleana* conforme a seguir (Obs: a variável C representa CIN):

Equação para S:

$$S = A\bar{B}\bar{C} \oplus \bar{A}B\bar{C} \oplus \bar{A}\bar{B}C \oplus ABC$$

$$S = \bar{C}(A \oplus B) + C(A \odot B)$$

$$S = \overline{\overline{\bar{C}(A \oplus B) + C(A \odot B)}}$$

$$S = \overline{C + \overline{(A \oplus B)}.(\bar{C} + (A \oplus B))}$$

$$S = \overline{C.(A \oplus B) + \bar{C}.\overline{(A \oplus B)}}$$

$$S = (\bar{C} + \overline{A \oplus B}).(C + A \oplus B)$$

$$S = \bar{C}.A \oplus B + C.\overline{A \oplus B}$$

$$\boxed{S = C \oplus A \oplus B}$$

Equação para COUT:

$$COUT = AB\bar{C} + A\bar{B}C + \bar{A}BC + ABC$$

$$COUT = AB\bar{C} + C(A \oplus B) + ABC$$

$$\boxed{COUT = AB + C(A \oplus B)}$$

Através das equações de S e COUT, elaborou-se o circuito lógico no *EAGLE* mostrado abaixo:

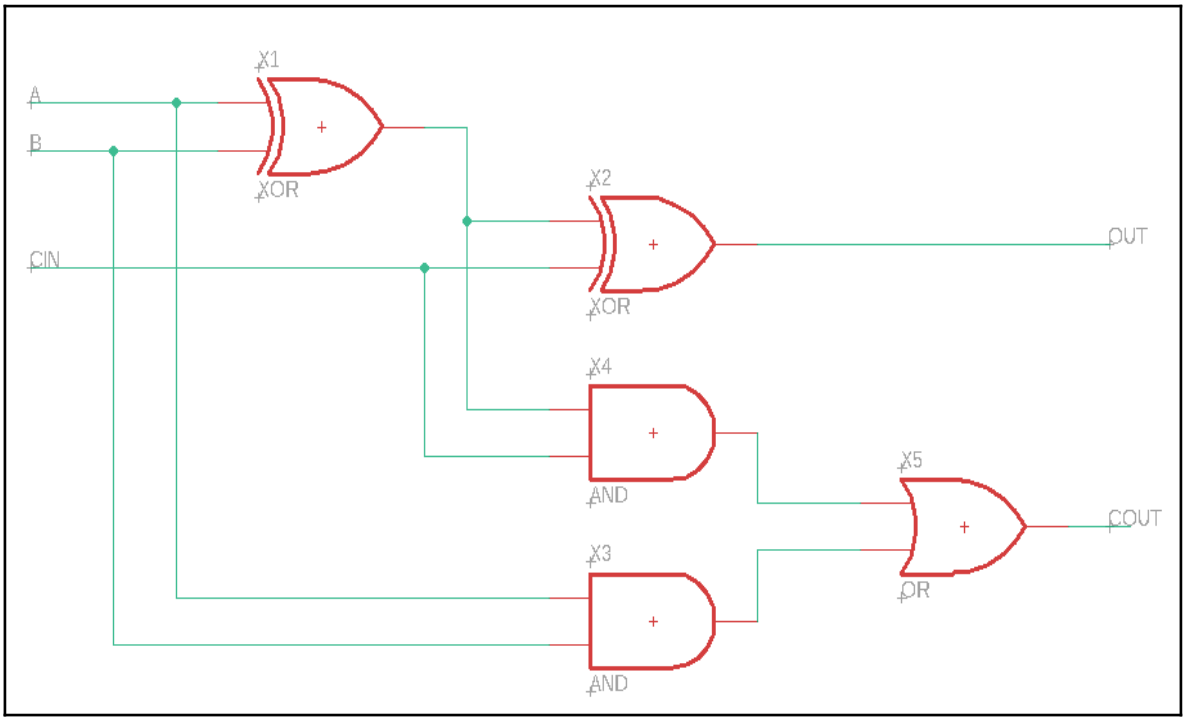

Figura 1: Circuito lógico do somador completo de 1 bit binário

Obs: Cada componente foi conectado nas referências “0” e VDD através do comando *Invoke* (suprimidos da figura para melhor visualização).

Ao exportar o *netlist* com as definições do circuito para o arquivo *somador_bin_defs.cir* obteve-se o seguinte arquivo (aqui simplificado):

```
* SpiceNetList
*
* Exported from somador_bin.sch
*
* EAGLE Version 9.5.1 Copyright (c) 1988-2019 Autodesk, Inc.
*
.TEMP=25.000000
* --------- .OPTIONS ---------
* --------- .PARAMS ---------

* --------- devices ---------
X_X4 N_3 CIN N_1 VDD 0 AND2
X_X5 N_1 N_2 COUT VDD 0 OR2
X_X1 A B N_3 VDD 0 XOR2
X_X2 N_3 CIN OUT VDD 0 XOR2
X_X3 A B N_2 VDD 0 AND2
```

Executando o comando do *plugin* do S*pice Opus*, foi gerado um novo *netlist* atualizado conforme a seguir:

run spice.ulp C:\dev\ somador_bin_defs.cir somador_bin_exp.cir

```
*Autogenerated from schematic somador_bin.sch
*.INCLUDE this file from somador_bin_defs.cir
*Do not edit anything except lines starting with '+'

X1 A B N3 VDD 0 XOR2
X2 N3 CIN OUT VDD 0 XOR2
X3 A B N5 VDD 0 AND2
X4 N3 CIN N6 VDD 0 AND2
X5 N6 N5 COUT VDD 0 OR2
```

Abaixo, segue o código que foi executado para a simulação no *Spice Opus*. A linha “.include somador_bin_exp.cir”, adiciona o arquivo anterior no *script*.

```
* Somador completo 1-bit binário

VDD VDD 0 dc=5
VCLK1 A 0 pulse=(5 0 0 1p 1p 1u 2u)
VCLK2 B 0 pulse=(5 0 0 1p 1p 2u 4u)
VCLK3 CIN 0 pulse=(5 0 0 1p 1p 4u 8u)

.include amis_c5n.txt
.include cell_lib.spice
.include somador_bin_exp.cir

.control
```

```
   tran 1n 16u
   plot A B+6 CIN+12 OUT+18 COUT+24
.endc

.end
```

Na figura 2 pode-se observar o gráfico gerado pelo *Spice Opus* onde, comparando-o com a tabela verdade, observou-se que o circuito simplificado está correto.

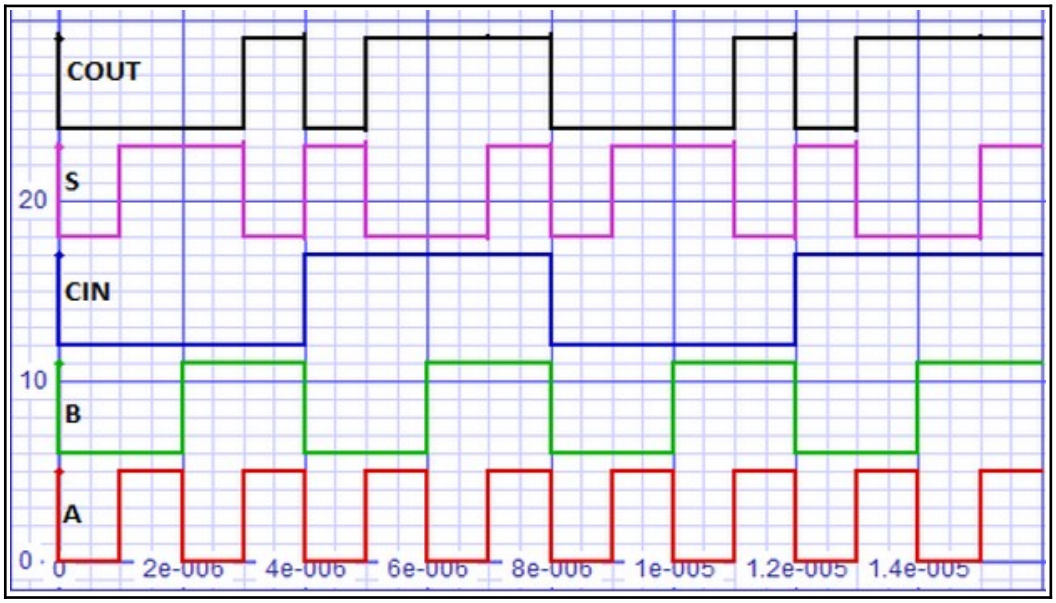

Figura 2 – Resultado da simulação do somador completo binário no *Spice Opus*

4. SOMADOR COMPLETO DE 1 BIT (GRAY)

O somador completo de 1-bit *Gray* possui a seguinte tabela verdade:

Variáveis de entrada			**Saídas do somador**	
CIN	**B**	**A**	**COUT**	**S**
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	1
1	0	0	0	1
1	0	1	1	1
1	1	0	1	1
1	1	1	1	0

Tal qual para o somador completo binário, foram aplicadas simplificações de álgebra *booleana* (Obs: a variável C representa CIN):

Equação para S:

$$S = B\bar{A} + \bar{C}A + C\bar{B}$$

$$S = \overline{\overline{B\bar{A} + \bar{C}A + C\bar{B}}}$$

$$S = \overline{\overline{B\bar{A}}.\overline{\bar{C}A}.\overline{C\bar{B}}}$$

$$S = \overline{(\bar{B}C) + \bar{B}\bar{A} + CA.(\bar{C} + B)}$$

$$S = \overline{(\bar{B}C + \bar{B}\bar{A} + CA).(\bar{C} + B)}$$

$$S = \overline{\bar{C}.\bar{A}.\bar{B} + CAB}$$

$$S = (B + \overline{\bar{C}\bar{A}}).(\bar{B} + \overline{AC})$$

$$S = B\overline{CA} + \bar{B}\overline{\bar{C}\bar{A}} + \overline{\bar{C}\bar{A}} + \overline{CA}$$

$$S = B\overline{CA} + \bar{B}\overline{\bar{C}\bar{A}}$$

$$S = B(\bar{C} + \bar{A}) + \bar{B}(A + C)$$

$$S = B\bar{A} + \bar{B}A + \bar{C}B + C\bar{B}$$

$$\boxed{S = B \oplus A + B \oplus C}$$

Como o COUT do somador Gray apresentou os mesmos valores que o do somador binário, a sua equação é a mesma:

$$\boxed{COUT = AB + C(A \oplus B)}$$

Através das equações de S e COUT, elaborou-se o circuito lógico no *EAGLE* mostrado abaixo:

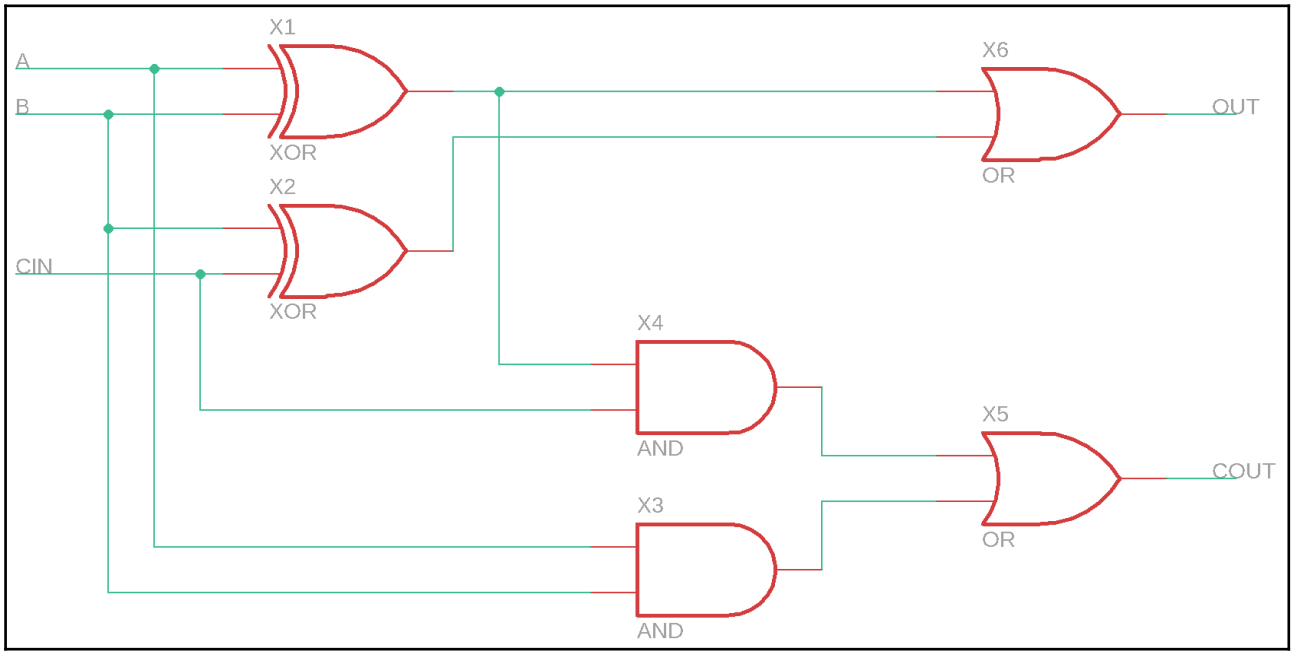

Figura 3: Circuito lógico do somador completo de 1 bit *Gray*

Obs: Cada componente foi conectado nas referências “0” e VDD através do comando *Invoke* (suprimidos da figura para melhor visualização).

Ao exportar o *netlist* para o arquivo *somador_gray_defs.cir* obteve-se o seguinte arquivo (aqui simplificado):

```
* SpiceNetList
*
* Exported from somador_gray.sch
*
* EAGLE Version 9.5.1 Copyright (c) 1988-2019 Autodesk, Inc.
*
.TEMP=25.000000
* --------- .OPTIONS ---------
* --------- .PARAMS ---------

* --------- devices ---------
X_X4 N_3 CIN N_1 VDD 0 AND2
X_X5 N_1 N_2 COUT VDD 0 OR2
X_X6 N_3 N_4 OUT VDD 0 OR2
X_X2 B CIN N_4 VDD 0 XOR2
X_X3 A B N_2 VDD 0 AND2
X_X1 A B N_3 VDD 0 XOR2
```

Executando o comando do *plugin* do S*pice Opus*, foi gerado um novo *netlist* atualizado conforme a seguir:

run spice.ulp C:\dev\ somador_gray_defs.cir somador_gray_exp.cir

```
*Autogenerated from schematic somador_gray.sch
*.INCLUDE this file from somador_gray_defs.cir
*Do not edit anything except lines starting with '+'

X1  A B N3 VDD 0 XOR
X2  B CIN N4 VDD 0 XOR
X3  A B N2 VDD 0 AND
X4  N3 CIN N1 VDD 0 AND
X5  N1 N2 COUT VDD 0 OR
X6  N3 N4 OUT VDD 0 OR
```

Abaixo, segue o código que foi executado para a simulação no *Spice Opus*. A linha “.include somador_gray_exp.cir”, adiciona o arquivo anterior no *script*.

```
* Somador completo 1-bit gray

VDD VDD 0 dc=5
VCLK1 A 0 pulse=(5 0 0 1p 1p 1u 2u)
VCLK2 B 0 pulse=(5 0 0 1p 1p 2u 4u)
VCLK3 CIN 0 pulse=(5 0 0 1p 1p 4u 8u)

.include amis_c5n.txt
.include cell_lib.spice
.include somador_gray_exp.cir

.control
   tran 1n 16u
   plot A B+6 CIN+12 OUT+18 COUT+24
```

```
.endc

.end
```

Na figura 4, tal qual foi feito com o somador binário, comparou-se o gráfico gerado pela simulação com a tabela verdade, comprovando-se que a simplificação do circuito correspondeu com sua tabela verdade:

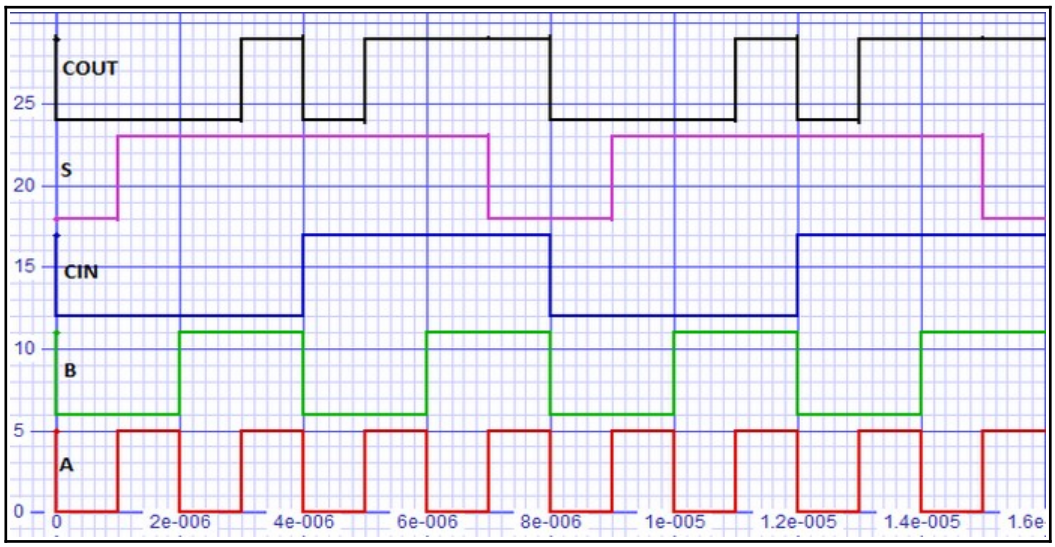

Figura 4 – Resultado da simulação do somador completo *Gray* no *Spice Opus*

5. COMPARAÇÃO DOS SOMADORES

Podemos fazer a comparação entre os dois somadores quanto ao número de transistores utilizados em cada circuito. A tabela a seguir mostra o número de transistores utilizado para cada tipo de porta lógica baseado na tecnologia CMOS[1]. Utilizou-se, também como premissa a não-degradação do sinal. Isso porque há maneiras de se construir portas XOR utilizando um número menor de transistores, ignorando-se possíveis variações no sinal. Mas como os dois somadores utilizam o mesmo número de portas XOR (duas cada) o resultado final, que é a comparação entre os dois não ia sofrer alteração.

Porta Lógica	Inversor	AND 2-input	OR 2-input	XOR 2-input
Nº de Transistores	2	6	6	6

Considerando o número de portas lógicas utilizadas em cada somador, conclui-se que o número de transistores utilizados para cada somador é:

Somador	**Binário**	**Gray**
Nº de Transistores	30	36

6. CONCLUSÃO

Neste trabalho foram abordados aspectos relacionados à simulação de circuitos digitais, através das ferramentas *EAGLE* e *Spice Opus*. Foram construídos circuitos somadores de 1 bit e, através da álgebra *booleana*, foi possível a simplificação dos mesmos. Essa redução gera como um dos benefícios, a simplificação do hardware sendo possível reduzir a utilização de transistores para a construção destes circuitos. Ao final do trabalho uma comparação entre os dois somadores testados mostrou que, com relação ao número de transistores, o somador binário leva vantagem em relação ao somador Gray, pois utiliza seis transistores a menos em sua construção.

7. REFERÊNCIAS

[1] Jan M. Rabaey, ***Digital Integrated Circuits***, Fall 2001:*Course Notes, Chapter 6: Designing Combinatorial Logic Gates in CMOS*.

[2] ***Website oficial da ferrementa Spice Opus.*** Disponível em: <http://fides.fe.uni-lj.si/spice/index.html> - Último acesso em 05 de julho 2020.

[3] COSTA, Eduardo Antonio César da. ***Portas Lógicas – Slides Adaptados dos Cursos dos Professores Ricardo Reis e Rabaey e do curso Circuitos Integrados Digitais do Prof. José Luis Guntzel***. Pelotas: UCPel, 2020.

[4] KRAUSE, Guilherme K. ***Software Spice Opus Tutorial – Descrevendo e Simulando Uma Porta AND2 No Simulador Spice Opus***. Pelotas: UCPel, 2010.