UNIVERSIDADE CATÓLICA DE PELOTAS

MESTRADO EM ENGENHARIA ELETRÔNICA E COMPUTAÇÃO

DOUGLAS ADALBERTO SCHEUNEMANN

DISCIPLINA DE INTRODUÇÃO AO PROJETO DE CIRCUITOS VLSI (PCVLSI)

TRABALHO I

Pelotas, maio de 2015

SUMÁRIO

1 Trabalho proposto

1 – Implementar na ferramenta Spice Opus um circuito Contador de 4 bits utilizando Flip-Flops JK Mestre-Escravo.

2 – Implementar na ferramenta Spice Opus somadores completos (Full Adders) nas codificações Binária e Gray. Realizar os circuitos com o menor número de portas lógicas possível. Comparar os dois circuitos em termos de número de Transistores.

2 Materiais e métodos

A seguir serão descritos os softwares e as técnicas utilizadas na execução deste trabalho.

2.1 Software Spice Opus

O software Spice Opus foi utilizado para a simulação dos circuitos propostos no trabalho. O mesmo foi desenvolvido pelo grupo de EDA da Faculdade de Engenharia Elétrica da Universidade de Ljubljana da Slovenia. Trata-se de um software *free*, dedicado para a simulação de circuitos utilizando a linguagem *Spice*.

A especificação da simulação através deste software pode ser feita em nível de transistores, especificando características de tecnologia como largura do canal, capacitâncias parasitas, dentre outras. Através da organização em subcircuitos é possível modelar arquiteturas a partir de uma biblioteca de células, elevando assim o nível de abstração do modelo e facilitando a sua análise e entendimento.

O Spice Opus oferece ainda suporte para a utilização do software EAGLE como editor gráfico de circuito. Isto é feito através da inclusão de um *plugin* no EAGLE.

2.2 EAGLE PCB Design Software

O software EAGLE foi utilizado para a edição gráfica dos circuitos e subcircuitos criados no trabalho. O mesmo foi desenvolvido pela empresa Cadsoft e possui uma versão *free* para uso não comercial, disponível para download a partir deste <Link>, acesso em 02 de maio de 2015.

Mesmo os circuitos analisados neste trabalho sendo de baixa complexidade e de fácil implementação em linguagem Spice, foi explorada a utilização do software EALGE

como editor gráfico, servindo como referência teórica para criação de circuitos mais complexos em trabalhos futuros.

Um tutorial sobre a utilização do EAGLE em conjunto com o Spice Opus pode ser encontrado no <link>, acesso em 02 de maio de 2015. Ao logo do trabalho serão apresentados os comandos e arquivos gerados para exportar os dados do EAGLE para o Spice Opus.

3 Contador de quatro bits utilizando flip-flop JK Mestre Escravo

Um contador de quatro bits pode ser construído utilizando-se quatro flip-flops do tipo JK Mestre-Escravo. Este flip-flop apresenta a reposta mostrada na Tabela 1, o seu símbolo é apresentada na Figura 1.

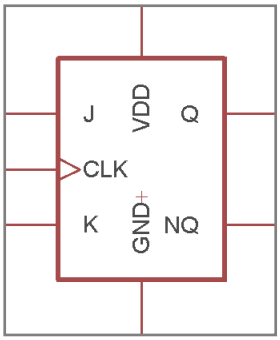

Figura 1: símbolo flip-flop JK.

Tabela 1: tabela verdade do flip-flop JK.

J	***K***	***Qf***
0	*0*	Qa
0	*1*	*0*
1	*0*	*1*
1	*1*	$\overline{Qa}$

Um flip-flop JK pode ser configurado para inverter o seu valor de saída para uma borda do sinal de clock ao qual é sensível. Isto é feito aplicando nível lógico “1” nas entradas “J” e “K”, como pode ser visto na Tabela 1. Desta forma, a saída do flip-flop apresentará uma frequência que corresponde à metade da frequência do sinal de clock.

Interligando a saída Q de um flip-flop com a entrada de clock de outro é possível obter sucessivas divisões por dois do sinal de clock da entrada. Considerando que a entrada de clock do primeiro flip-flop será conectada ao sinal que se deseja contar o número de transições, a saída Q de cada flip-flop corresponderá a um bit do contador binário.

Na figura 2, pode ser visto o circuito que corresponde ao contador binário de quatro bits em questão. Cabe ressaltar que esta topologia corresponde ao contador conhecido na literatura como assíncrono. Uma desvantagem deste circuito é que o caminho crítico

resultante para a propagação da contagem corresponde a soma do atraso de cada flip-flop. Pode destacar-se com principal vantagem deste circuito a sua simplicidade de construção.

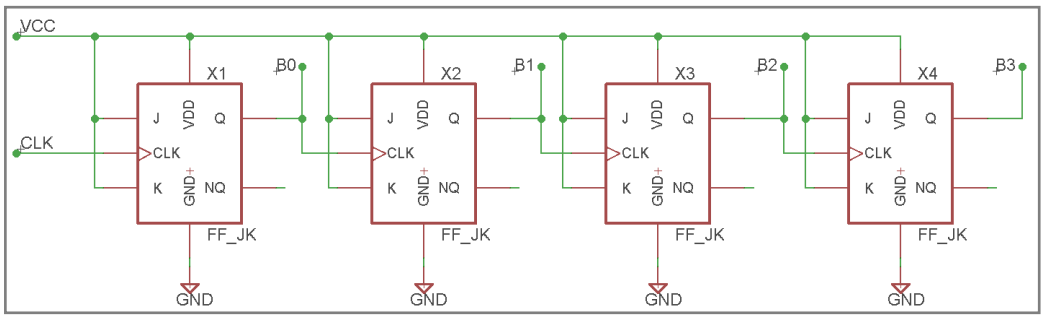

Figura 2: circuito para o contador binário de 4 bits utilizando flop-flop JK Mestre-Escravo.

O circuito mostrado na Figura 2 foi criado no software EAGLE, e a partir do mesmo, foi feita a exportação das interligações dos blocos para a simulação no Spice Opus. As principais etapas para a realização deste processo serão descritas a seguir.

3.1 Etapas realizadas no software EAGLE

Inicialmente foi criada uma biblioteca no EAGLE contendo um componente para o flip-flop JK. O nome do componente no EAGLE deve ser igual ao do componente da biblioteca do Spice Opus.

Utilizando o editor de esquemáticos do EAGLE, foi desenhado o circuito mostrado na Figura 2. Logo, foi executado o *plugin* para exportação da net list do circuito para o Spice Opus. A execução do plugin é feita através do comando mostrado no código abaixo, executado no prompt de comandos do EAGLE.

```
run spice.ulp C:\Documents\Contador_4b\ Cell_Defs_Eagle.cir  Contador_4b_Exp_Eagle.cir
```

Código 1: comando para execução do plugin de exportação para Spice Opus a partir do EALGE.

O comando “run spice.ulp” corresponde a chamada para o plugin. Os demais campos separados por espaços correspondem aos parâmetros para o plugin, sendo eles:

- “C:\Documents\Contador_4b\”: pasta no computador onde se encontram os arquivos que serão lidos e escritos pelo plugin. Deve ser utilizado “\” como separado e não “/” como é o padrão do sistema operacional Windows;
- “Cell_Defs_Eagle.cir”: arquivo que contem a definição de nomes de pinos e ordem de interligação do subcircuitos utilizados no esquemático;

- “Contador_4b_Exp_Eagle.cir”: arquivo onde será gerado o código para a utilização no Spice Opus, contendo os comandos Spice para inclusão e interconexão dos blocos do circuito. Este arquivo deve ser criado manualmente pelo usuário antes da execução do plugin.

Para o circuito em questão, foi utilizado o código abaixo no arquivo “Cell_Defs_Eagle.cir”. Para cada subcircuito deve ser inserida uma linha com o prefixo “.SUBCKT”, seguida do nome do componente e dos pinos de entrada e saída, com os nomes utilizados no EAGLE e na ordem com que são declarados na biblioteca do Spice Opus.

```
* Definição de subcircuitos
.SUBCKT FF_JK J K CLK Q NQ VDD GND
```

Código 2: definição de subcircuitos utilizados no contador binário.

Após executar o comando de exportação no EALGE, foi gerado no arquivo “Contador_4b_Exp_Eagle.cir” com o código mostrado abaixo.

```
X1  VCC VCC CLK B0 N4 VCC 0 FF_JK
X2  VCC VCC B0 B1 N3 VCC 0 FF_JK
X3  VCC VCC B1 B2 N2 VCC 0 FF_JK
X4  VCC VCC B2 B3 N1 VCC 0 FF_JK
```

Código 3: código gerado para o circuito do contador binário de 4 bits.

3.2 Simulação no Spice Opus

Após a geração da *net list* do circuito no formato Spice apartir do EAGLE, foi criado o arquivo “Contador_4b_Simulacao.cir”, com o conteúdo mostrado abaixo. Cabe ressaltar que, a partir da diretiva “.include Contador_4b_Exp_Eagle.cir” são adicionados ao arquivo de simulação os parâmetros exportados do EALGE.

```
*Definições para simulação do contador de 4 bits com FF JK

VDD VCC 0 dc=5
VCLK CLK 0 pulse(5 0 0 100p 100p 1u 2u)

.include amis_c5n.txt
.include cell_lib.spice
.include Contador_4b_Exp_Eagle.cir

.control
    tran 1n 40u
    plot CLK B0+6 B1+12 B2+18 B3+24
.endc

.end
```

Código 4: comandos para simulação do contador definido no arquivo “Contador_4b_Exp_Eagle.cir”.

Executando o código de simulação no Spice Opus foi obtido o resultado que pode ser visto na Figura 3. No gráfico podem ser observados dois ciclos de contagem de zero a oito, mostrando que o circuito apresentou o funcionamento que era esperado.

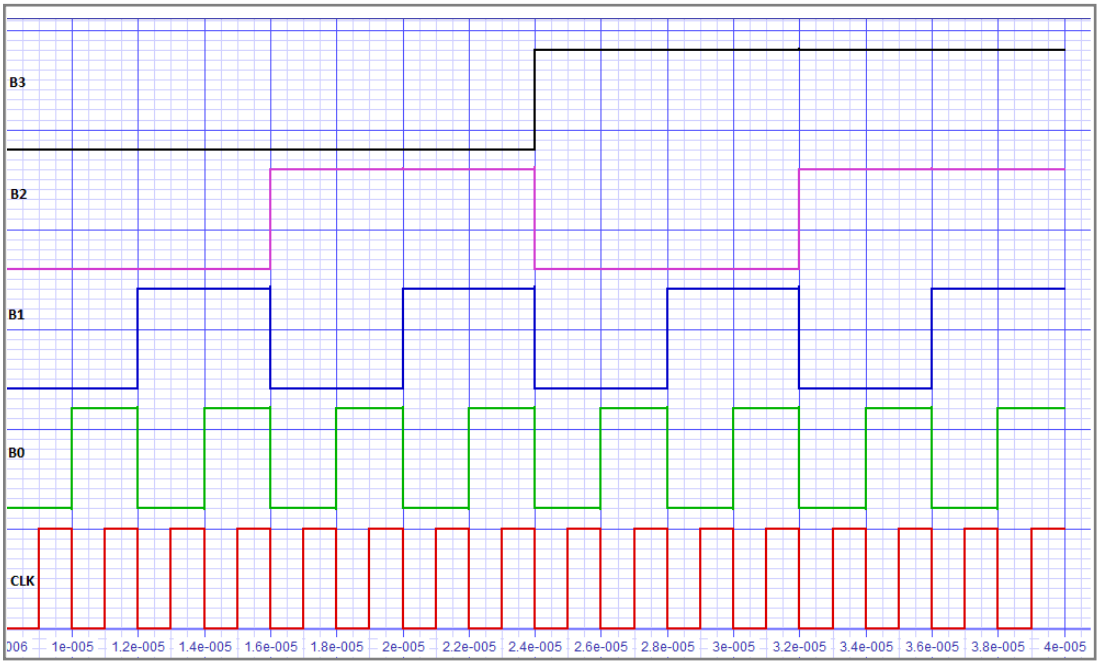

Figura 3: resultado obtido para a simulação do contador de 4 bits no Spice Opus.

4 Somadores Full Adder Binário e Gray

Para criação dos somadores, assim como foi feito apara o contador descrito na seção 3, utilizou-se o software EALGE para edição gráfica do circuito e exportação da *net list* no formato Spice. Nas subseções a seguir serão descritas as etapas de desenvolvimento de cada somador e da porta XOR compartilhada entre ambos.

4.1 Porta XOR2

A porta XOR2 é necessária para a construção de ambos os somadores abordados neste trabalho. Foram exploradas três formas de construção da mesma, descritas a seguir.

4.1.1 Porta XOR2 a partir de portas lógicas AND2, OR2 e Inversor

Utilizando-se as portas lógicas básicas AND2, OR2 e inversor, é possível criar a porta XOR2 partindo da equação $S = A\bar{B} + \bar{A}B = A \oplus B$, cujo circuito pode ser visto na Figura 4 e simulação na Figura 5.

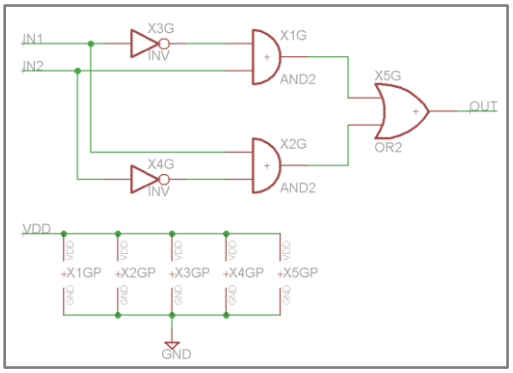

Figura 4: circuito para porta XOR2.

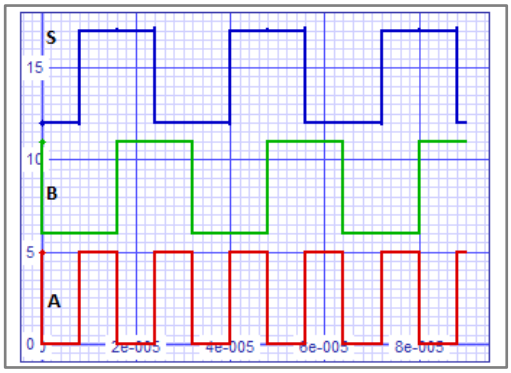

Figura 5: simulação porta XOR2.

```
X1   N1 IN2 N2 VDD 0 AND2
X2   IN1 N4 N3 VDD 0 AND2
X3   IN1 N1 VDD 0 INV
X4   IN2 N4 VDD 0 INV
X5   N2 N3 OUT VDD 0 OR2
```

Código 5: *net list* para porta XOR2 exportado a partir do EAGLE, arquivo "Porta_Xor_Exp_Eagle.cir".

```
* Definição de subcricuito para porta XOR2
.SUBCKT XOR2 IN1 IN2 OUT VDD GND
.INCLUDE Porta_Xor_Exp_Eagle.cir
.ENDS XOR2
```

Código 6: definição da porta XOR2.

```
*Simulação Porta XOR2

VDD VCC 0 dc=5
VCLK2  IN1 0 pulse=(5 0 0 1p 1p 8u 16u)
VCLK1  IN2 0 pulse=(5 0 0 1p 1p 16u 32u)

.include amis_c5n.txt
.include cell_lib.spice
.include Porta_Xor.cir

X1 IN1 IN2 OUT VCC 0 XOR2

.control
    tran 1n 90u
    plot IN1 IN2+6 OUT+12
.endc

.end
```

Código 7: comandos para simulação no Spice Opus da porta XOR2.

4.1.2 Porta XOR2 com transistores de passagem

A porta XOR2 pode ser construída de maneira simplificada utilizando-se transistores de passagem, conforme Figura 6. Durante o texto, esta porta será chamada de XOR2_TP.

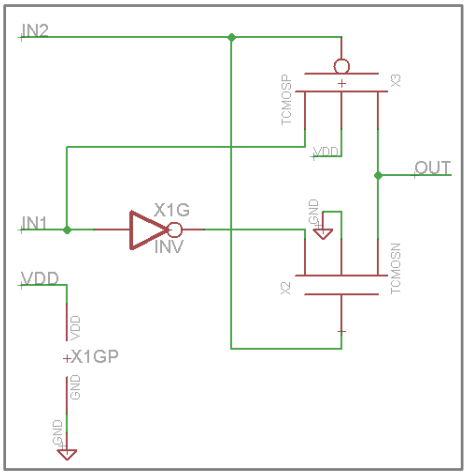

Figura 6: porta XOR2 com transistores de passagem.

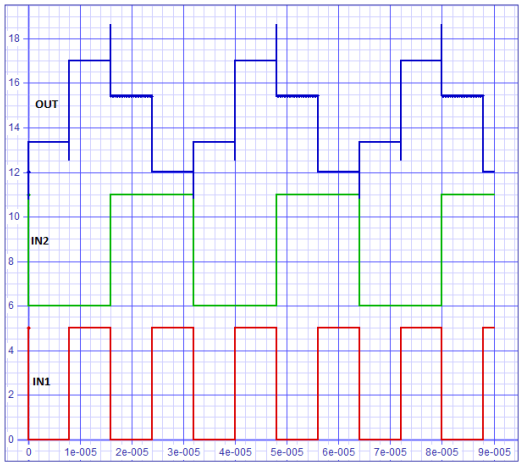

Figura 7: resultado para simulação da porta XOR2 com transistores de passagem.

A partir da simulação mostrada na Figura 7, nota-se que a porta XOR2 construída com transistores de passagem, apesar de possuir um número muito menor de transistores, quando comparada com a versão utilizando portas básicas, apresenta degradação do sinal de saída para as combinações mostradas na Tabela 2. Para ambos os casos mostrados na tabela, a degradação do sinal foi de cerda de **1,4V**.

Cabe ressaltar que, para fins de simulação, foram adicionados resistores de 10 MΩ da saída para VCC e da saída para GND, com a finalidade simular a impedância de entrada do circuito em que a porta é aplicada.

Tabela 2: combinações em que a porta apresenta sinal fraco na saída.

IN1	IN2	S
0	*0*	*0*
1	*0*	*1*

Os códigos utilizados na simulação podem ser vistos a seguir.

```
X1   IN1 N1 VDD 0 INV
X2   IN2 N1 OUT 0 TCMOSN
X3   IN2 OUT IN1 VDD TCMOSP
```

Código 8: *net list* para porta XOR2 utilizando transistores de passagem exportado a partir do EAGLE, arquivo "Porta_Xor_TP_Exp_Eagle.cir".

```
* Definição de subcricuito para para porta XOR2_TP com transistores de
passagem.
.SUBCKT XOR2_TP IN1 IN2 OUT VDD GND
.INCLUDE Porta_Xor_TP_Exp_Eagle.cir
.ENDS XOR2_TP
```

Código 9: definição da porta XOR2 com transistores de passagem.

```
*Simulação Porta_XOR2_TP

VDD VCC 0 dc=5
VCLK2  IN1 0 pulse=(5 0 0 1p 1p 8u 16u)
VCLK1  IN2 0 pulse=(5 0 0 1p 1p 16u 32u)

.include amis_c5n.txt
.include cell_lib.spice
.include Porta_Xor_TP.cir

X1 IN1 IN2 OUT VCC 0 XOR2_TP
R1  0 OUT 10Meg
R2  OUT VCC 10Meg

.control
    tran 1n 90u
    plot IN1 IN2+6 OUT+12
.endc

.end
```

Código 10: comandos para simulação no Spice Opus da porta XOR2_TP.

4.1.3 Porta XOR2 Trannmission Gate

Como tentativa de minimizar a degradação do sinal apresentado pela porta XOR com transistores de passagem, existe na literatura referência para a implementação com tecnologia *Transmission Gate*. O Circuito montado a partir desta técnica pode ser visto na Figura 8. Ao longo do texto esta porta será chamada de XOR2_TG.

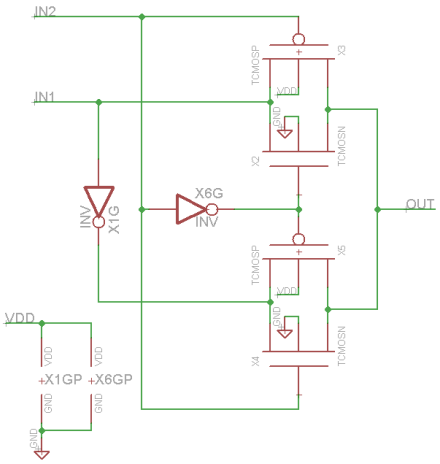

Figura 8: porta XOR2 com *Transmission Gate*.

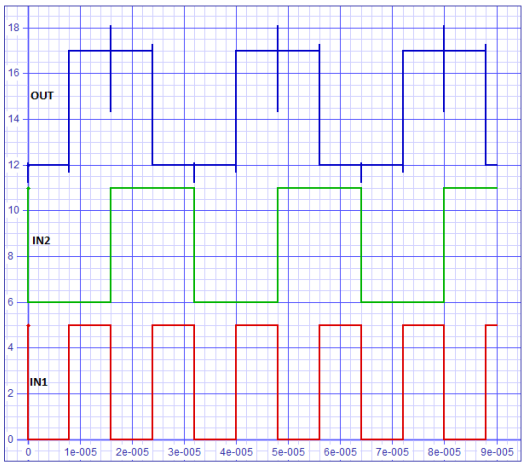

Figura 9: resultado para simulação da porta XOR2 com *Transmission Gate*.

Nota-se na simulação mostrada na Figura 9 que a implementação utilizando *Transmission Gate* é mais eficaz no que diz respeito à degradação do sinal, quando

comparada com a solução utilizando transistores de passagem, uma vez que são notadas distorções de nível somente quando ocorrem transições simultâneas dos sinais de entrada. Na prática, estas oscilações são automaticamente rejeitadas por circuitos síncronos.

Os códigos utilizados na simulação podem ser vistos a seguir.

```
X1   IN1 N3 VDD 0 INV
X2   N5 IN1 OUT 0 TCMOSN
X3   IN2 OUT IN1 VDD TCMOSP
X4   IN2 N3 OUT 0 TCMOSN
X5   N5 OUT N3 VDD TCMOSP
X6   IN2 N5 VDD 0 INV
```

Código 11: *net list* para porta XOR2 utilizando *Transmission Gate* exportado a partir do EAGLE, arquivo "Porta_Xor_TG_Exp_Eagle.cir".

```
* Definição de subcircuito para porta XOR utilizando Transmission Gate
.SUBCKT XOR2_TG IN1 IN2 OUT VDD GND
.INCLUDE Porta_Xor_TG_Exp_Eagle.cir
.ENDS XOR2_TG
```

Código 12: definição da porta XOR2 com *transmission Gate*.

```
*Simulação Porta_XOR2_TG

VDD VCC 0 dc=5
VCLK2  IN1 0 pulse=(5 0 0 1p 1p 8u 16u)
VCLK1  IN2 0 pulse=(5 0 0 1p 1p 16u 32u)

.include amis_c5n.txt
.include cell_lib.spice
.include Porta_Xor_TG.cir

X1 IN1 IN2 OUT VCC 0 XOR2_TG
R1  0 OUT 10Meg
R2  OUT VCC 10Meg

.control
    tran 1n 90u
    plot IN1 IN2+6 OUT+12
.endc

.end
```

Código 13: comandos para simulação no Spice Opus da porta XOR2_TG.

4.1.4 Comparação entre os tipos de porta XOR2

Foram descritas nesta seção três topologias de construção da porta XOR2. A seguir, será apresentada uma tabela comparativa entre elas.

Tabela 3: comparação entre as topologias de porta XOR2 simuladas.

	XOR2	XOR2_TP	XOR2_TG
Número de Transistores	24	4	8
Degradação de nível do sinal	Nenhuma	1,4 V em duas combinações de entrada	Somente na transição simultânea das entradas

Com base nos dados da Tabela 3, pode concluir-se que a porta XOR2 construída a partir da técnica *Transmission Gate*, apresenta uma relação cuto-benefício bastante interessante, uma vez que emprega a metade do número de transistores do que a porta utilizando portas lógicas básicas AND2, OR2 e inversor, e ainda possui baixo nível de degradação do nível do sinal de saída.

4.2 Somador Binário Full Adder

A primeira etapa para construção do somador Binário foi a elaboração da tabela verdade do mesmo, que pode ser vista na Figura 10.

Variáveis Entrada			Somador Binário	
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

Figura 10: tabela verdade para o somador Binário.

Partindo da tabela verdade e aplicando simplificação por álgebra de Boole, foram obtidas as equações booleanas para os sinais de saída (S) e *carry out* (COUT) do somador.

Equação de saída, termo C corresponde a Cin:

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

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

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

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

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

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

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

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

Equação para COUT, termo C corresponde a Cin:

$$COUT = \bar{C}BA + C\bar{B}A + CB\bar{A} + CBA$$

$$COUT = \bar{C}BA + C(B \oplus A) + CBA$$

$$\boldsymbol{COUT = BA + C(B \oplus A)}$$

Utilizando as equações de S e COUT descritas acima, foi criado o circuito mostrado na Figura 11.

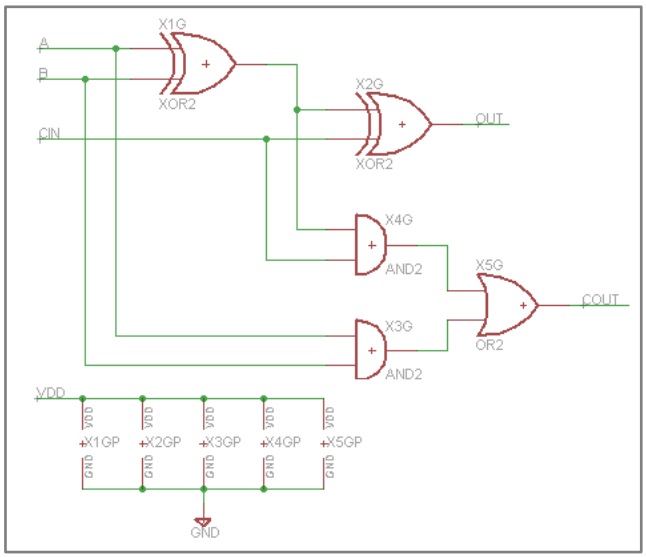

Figura 11: circuito para somador Full Adder Binário.

A seguir são apresentados os códigos utilizados para definição e simulação do circuito mostrado na Figura 11.

```
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
```

Código 14: *net list* Spice para o somador Binário exportado a partir do EAGLE, arquivo "Somador_Bin_Exp_Eagle.cir"

```
* Definições para simulação do somador 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 Porta_Xor.cir
.include Somador_Bin_Exp_Eagle.cir

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

.end
```

Código 15: comandos para simulação no Spice Opus do somador *Full Adder* Binário.

Na Figura 12 pode ser visto o resultado para a simulação feita no Spice Opus para o somador em questão. Pode observar-se que, o resultado da simulação está condizente com a tabela verdade apresenta na Figura 10, logo, pode concluir-se que, o circuito obteve o funcionamento esperado.

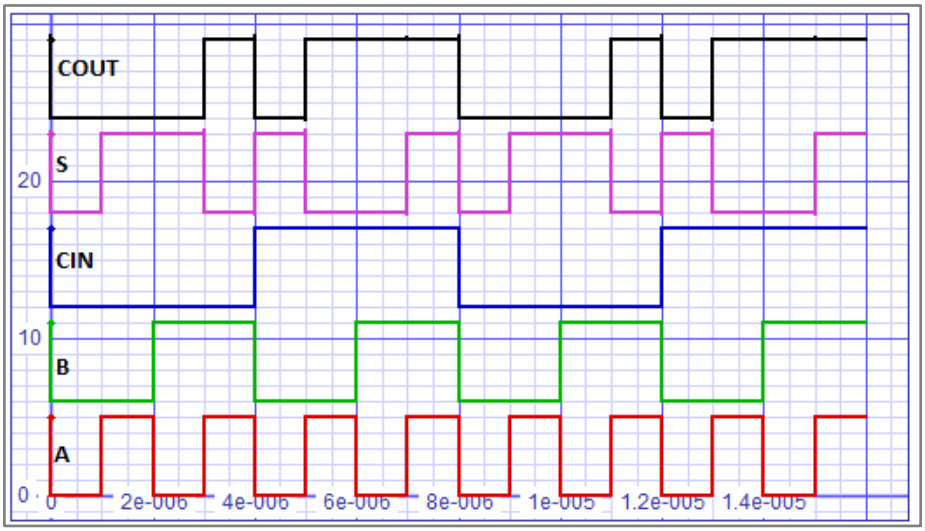

Figura 12: resultado para a simulação do somador Binário.

Como forma de comparação, a simulação do somador foi feita também utilizando as postas XOR2_TG e XOR2_TP, descritas na seção 4.1. Pode ser verificado na Figura 13 e na Figura 14 que para ambos os casos o circuito funcionou corretamente. Para a porta COR2_TP o sinal de saída apresentou maior distorção, como já era esperado em função das simulações individuais das portas. Ainda para o circuito com a porta XOR2_TP, observa-se que o sinal COUT não apresentou distorção de nível. Isso ocorre porque este sinal é regenerado pela porta AND2, a qual recebe o sinal proveniente da porta XOR2_TP.

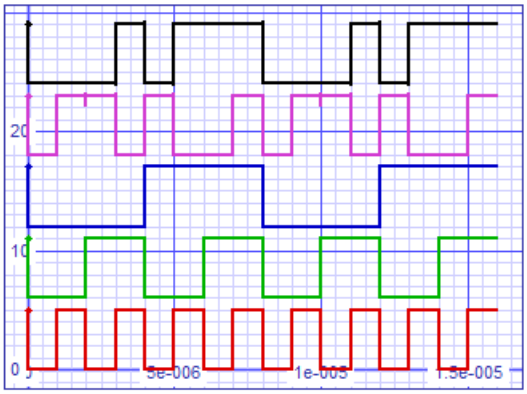

Figura 13: simulação do somador Binário com porta XOR2_TG

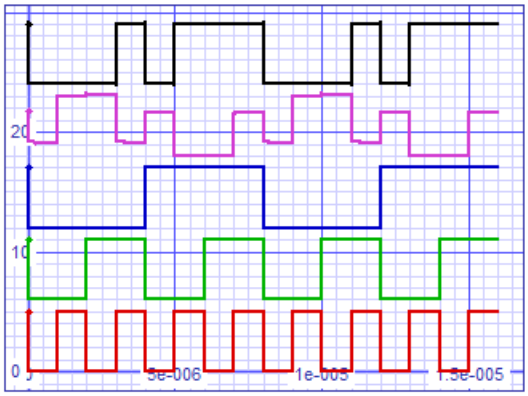

Figura 14: simulação do somador Binário com porta XOR2_TP.

4.3 Somador Gray Full Adder

Assim como foi feito para somador Binário, primeiramente foi obtida a equação booleana para o somador Gray Full Adder, partindo de sua tabela verdade, a qual é mostrada na Figura 15.

Variáveis Entrada			Somador Gray	
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

Figura 15: tabela verdade para o somador Gray Full Adder.

A partir da tabela verdade mostrada na figura Figura 15, pode ser obtido para o sinal de saída (S) a representação em mapa de Karnaugh mostrada na Figura 16. O sinal de saída COUT tem os mesmos valores obtidos para o somado Binário, logo, a sua equação lógica apresentada na seção 4.2 será reutilizada.

CIN \ BA	00	01	11	10
0	0	1	1	1
1	1	1	0	1

Figura 16: mapa de Karnaugh para saída (S).

Equação de saída, termo C corresponde a Cin:

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

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

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

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

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

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

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

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

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

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

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

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

Equação para o COUT, termo C corresponde a Cin:

$$\boldsymbol{COUT = BA + C(B \oplus A)}$$

Utilizando as equações de S e COUT descritas acima, foi criado o circuito mostrado na Figura 17.

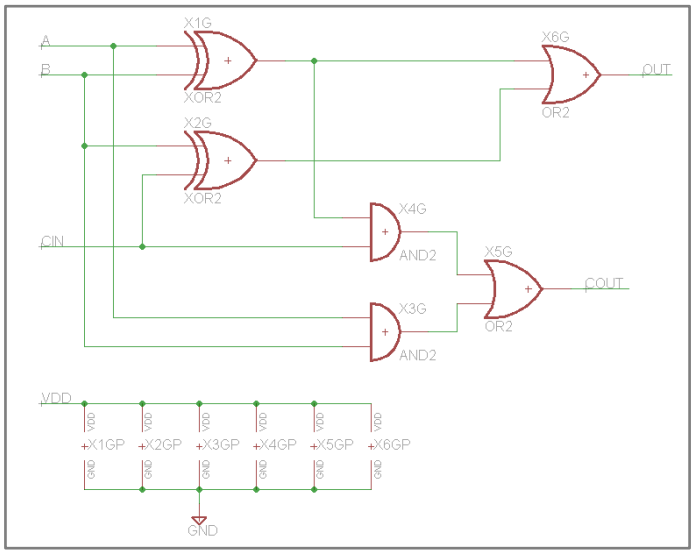

Figura 17: circuito para somador Full Adder Gray.

A seguir são apresentados os códigos utilizados para definição e simulação do circuito mostrado na Figura 17.

```
X1   A B N3 VDD 0 XOR2
X2   B CIN N1 VDD 0 XOR2
```

```
X3   A B N5 VDD 0 AND2
X4   N3 CIN N6 VDD 0 AND2
X5   N6 N5 COUT VDD 0 OR2
X6   N3 N1 OUT VDD 0 OR2
```

Código 16: *net list* Spice para o somador Gray exportado a partir do EAGLE, arquivo "Somador_Gray_Exp_Eagle.cir"

```
*  Definições para simulação do somador 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 Porta_Xor.cir
.include Somador_Gray_Exp_Eagle.cir

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

.end
```

Código 17: comandos para simulação no Spice Opus do somador *Full Adder* Gray.

Na Figura 18 pode ser visto o resultado para a simulação feita no Spice Opus para o somador em questão. Observa-se que o resultado da simulação está condizente com a tabela verdade, apresenta na Figura 15. Logo, pode-se concluir que o circuito obteve o funcionamento esperado.

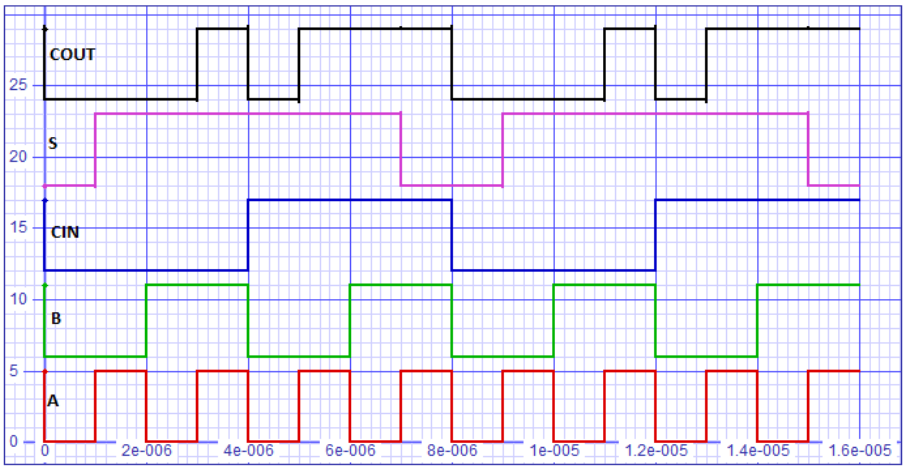

Figura 18: resultado para a simulação do somador Gray.

Para o somador Gray, foi feita também a simulação com as portas XOR2_TP e XOR2_TG. O circuito apresentou funcionamento correto para ambas as portas. Um aspecto importante que foi observado é que, para o somador Gray, as portas XOR2_TP não

estão conectadas diretamente à saída S ou COUT. Desta forma, o sinal fraco gerado por elas é regenerado pelas portas do estágio de saída, tornando a porta XOR2_TP uma solução interessante para o circuito, visto que a mesma apresenta o menor número de transistores dentre as arquiteturas de portas XOR2 abordadas neste trabalho.

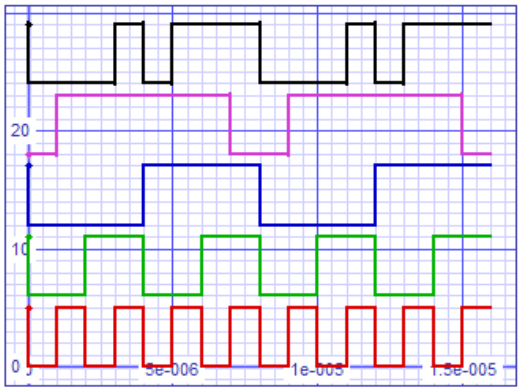

Figura 19: simulação do somador Gray com porta XOR2_TG.

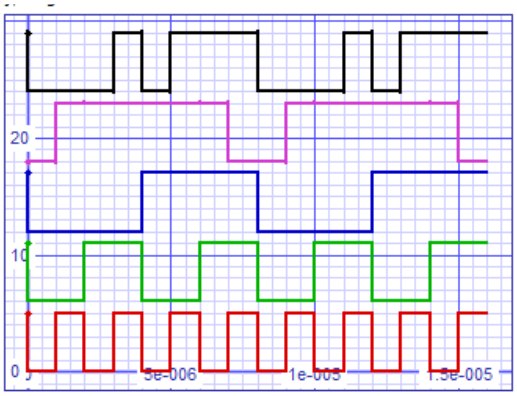

Figura 20: simulação do somador Gray com porta XOR2_TP.

4.4 Comparação das arquiteturas de somador Binário x Gray

Foram simuladas três arquiteturas para cada somador, empregando as portas: XOR2, XOR2_TG e XOR2_TP. Na Tabela 4 tem-se um resumo do número de transistores utilizados em cada caso, e também se houve degradação do sinal de saída em função do tipo de porta XOR aplicada.

Tabela 4: comparação entre as arquiteturas de somadores Binário e Gray construídos neste trabalho.

	Somador Binário			Somador Gray		
	XOR2	XOR2_TG	XOR2_TP	XOR2	XOR2_TG	XOR2_TP
Transistores	66	34	26	72	40	32
Degradação	Nenhuma	Nenhuma	Nível saída	Nenhuma	Nenhuma	Nenhuma

Nota-se na Tabela 4 que o número de transistores empregados no somador Gray foi maior em seis. Isso se dá em função da utilização de uma porta OR2 a mais que somador Binário. Porém, o somador Gray apresenta uma menor probabilidade de transição do sinal de saída, o que na prática gera um menor consumo de energia.

Conforme foi demonstrado na simulação da Figura 20, para o somador Gray, construído com portas XOR2_TP, ocorre a regeneração do sinal de saída, visto que as portas XOR_TP não estão ligadas diretamente à saída. Desta forma, o somador Gray pode

ser construído sem prejuízo à qualidade do sinal, utilizando-se trinta e dois transistores, ou seja, dois transistores a menos que o somador Binário, que possui o menor número de transistores sem distorção de nível na saída.

Conclui-se, portanto, que o somador Gray, construído com portas XOR baseadas em transistores de passagem, apresentou a melhor relação desempenho x número transistores.

5 Conclusão

Através deste trabalho foram abordados aspectos relacionados à simulação de circuitos digitais, partindo da especificação em nível de transistores. Foi possível verificar como os softwares EAGLE e Spice Opus podem ser utilizados em conjunto para realizar este tipo de simulação.

Pode analisar-se ainda como as otimizações realizadas nos níveis arquiteturais e de transistores, podem reduzir a complexidade dos circuitos e consequentemente o seu custo. Este tipo de análise é importante, uma vez que, em sistemas reais busca-se encontrar o ponto ótimo entre as métricas desempenho, custo e consumo de energia.

6 Bibliografia

COSTA, Eduardo Antonio César da. **Operadores Aritméticos De Baixo Consumo Para Arquiteturas De Circuitos DSP**. Porto Alegre: PPGC da UFRGS, 2002.

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, 2015.

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

Spice Opus - Getting Started. Disponível em:
< http://www.spiceopus.si/schematic.html> Acesso em: 02 de maio de 2015.

Using EAGLE as a SPICE OPUS Schematic Editor. Disponível em:
< http://www.spiceopus.si/schematic.html> Acesso em: 02 de maio 2015.