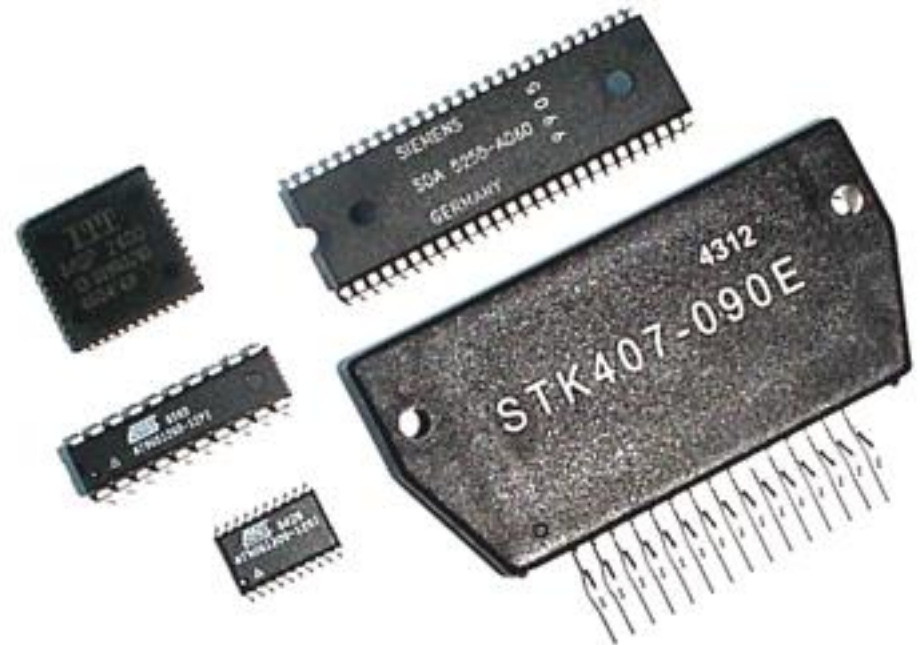
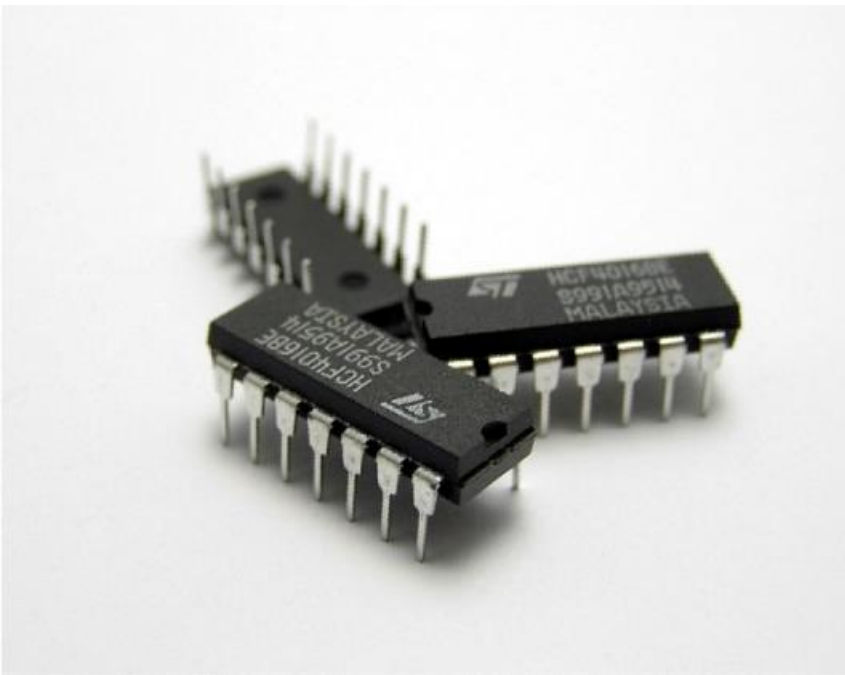




CIRCUITO SOMADOR

Objetivos:

- Adquirir conhecimentos em dispositivos de lógica programável;
- Entendimento do sistema numérico;
- Estudo do circuito somador;
- Estudo do circuito somador real usado em ULA (unidade lógica aritmética).



Representação dos números:

Existem várias formas de se representar números, entre elas no formato binário:

- Não sinalizada (Somente números positivos);
 - Simples ou direta.
- Sinalizada (suporta número negativos);
 - Complemento de 2.
 - sinal (positivo=0, negativo=1) e magnitude (valor absoluto).

Para os nossos estudos neste momento usaremos a simples também conhecida como direta.

Simples ou direta:

7	6	5	4	3	2	1	0
---	---	---	---	---	---	---	---

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
-------	-------	-------	-------	-------	-------	-------	-------

128	64	32	16	8	4	2	1
-----	----	----	----	---	---	---	---

Ex:

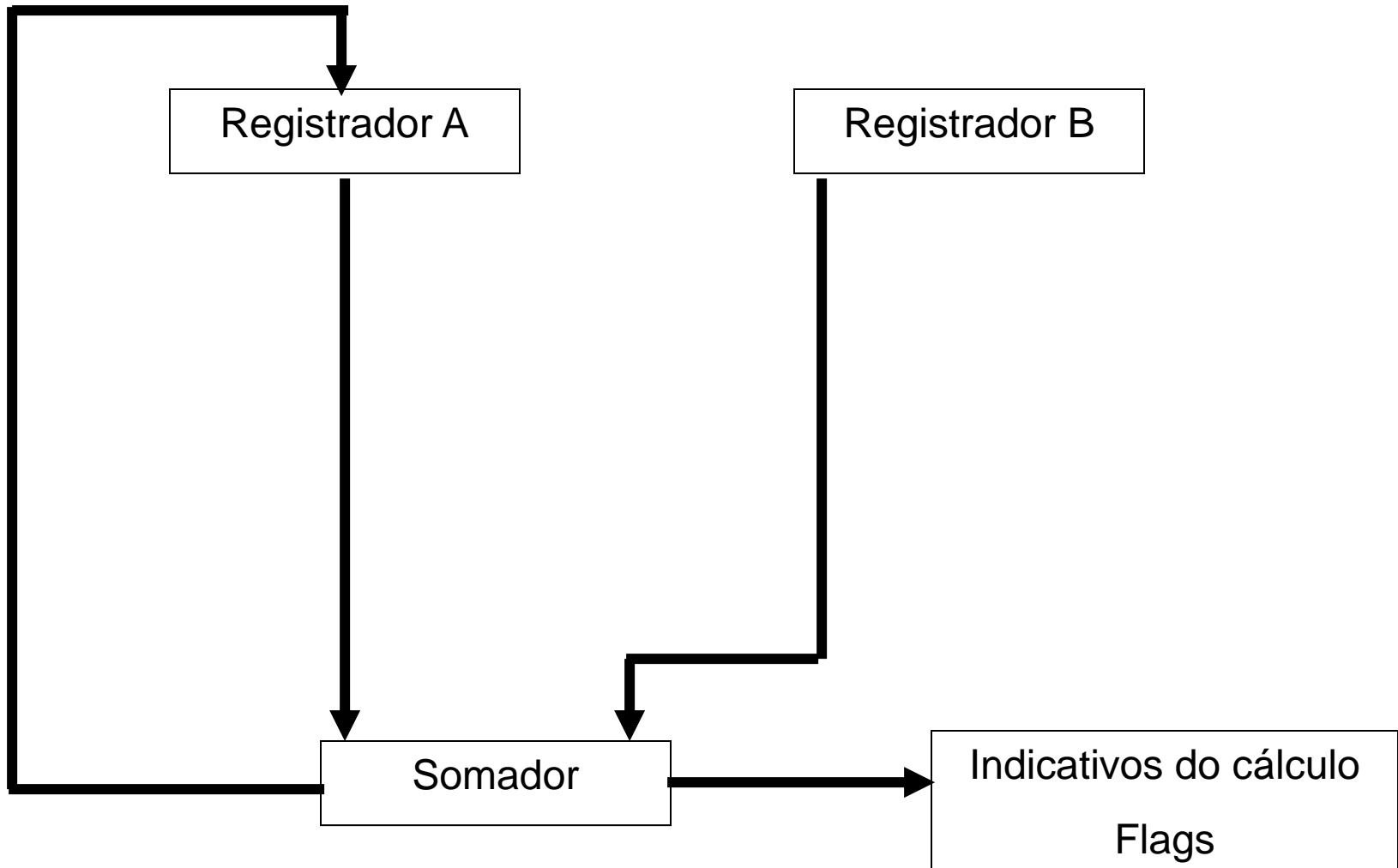
1	0	0	0	0	0	0	1
---	---	---	---	---	---	---	---

 = 129

Forma de adição

$$\begin{array}{rcll} \text{Carry-out} & \text{ou vai-um} & & 111 \\ & \text{Número A} & = & 0011 \quad 0111 \\ + & \text{Número B} & = & 0000 \quad 0011 \\ & & & \hline & & & 0011 \quad 1010 \end{array}$$

Diagrama de blocos de um Hardware para Adição



O meio-somador

Unidade de soma é baseada na coluna:

Carry-out ou vai-um

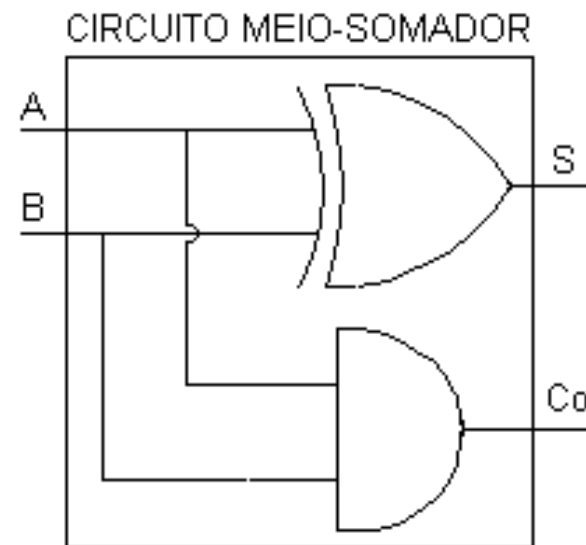
$$\begin{array}{r}
 \text{Número A} = 0011 \quad 0111 \\
 + \text{Número B} = 0000 \quad 0011 \\
 \hline
 0011 \quad 1010
 \end{array}$$

Onde o termo S é a resposta da soma e Co é o carry-out da soma.

Abaixo temos a expressão lógica para um **meio-somador**:

A	B	S	Co
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

$$\begin{aligned}
 S &= \bar{A}.B + A.\bar{B} \\
 Co &= A.B
 \end{aligned}$$



O somador completo:**Construído a tabela verdade e resolvendo a solução fica:**

A	B	Ci	S	Co
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Função S (soma) - Soma dos minitermos

$$S = \bar{A}.\bar{B}.Ci + \bar{A}.B.\bar{Ci} + A.\bar{B}.\bar{Ci} + A.B.Ci$$

Simplificação

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

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

$$S = Ci \oplus (A \oplus B)$$

Função Co ("vai um" ou "carry out") - Soma dos minitermos

$$Co = (\bar{A}.B.Ci) + (A.\bar{B}.Ci) + (A.B.\bar{Ci}) + (A.B.Ci)$$

Simplificação

$$Co = Ci . (\bar{A}.B + A.\bar{B}) + A.B . (\bar{Ci} + Ci)$$

$$Co = Ci . (A \oplus B) + A.B \underbrace{(\bar{Ci} + Ci)}_1$$

Propriedades úteis:

Soma dos minitermos:

$$A \oplus B = \bar{A}.B + A.\bar{B}$$

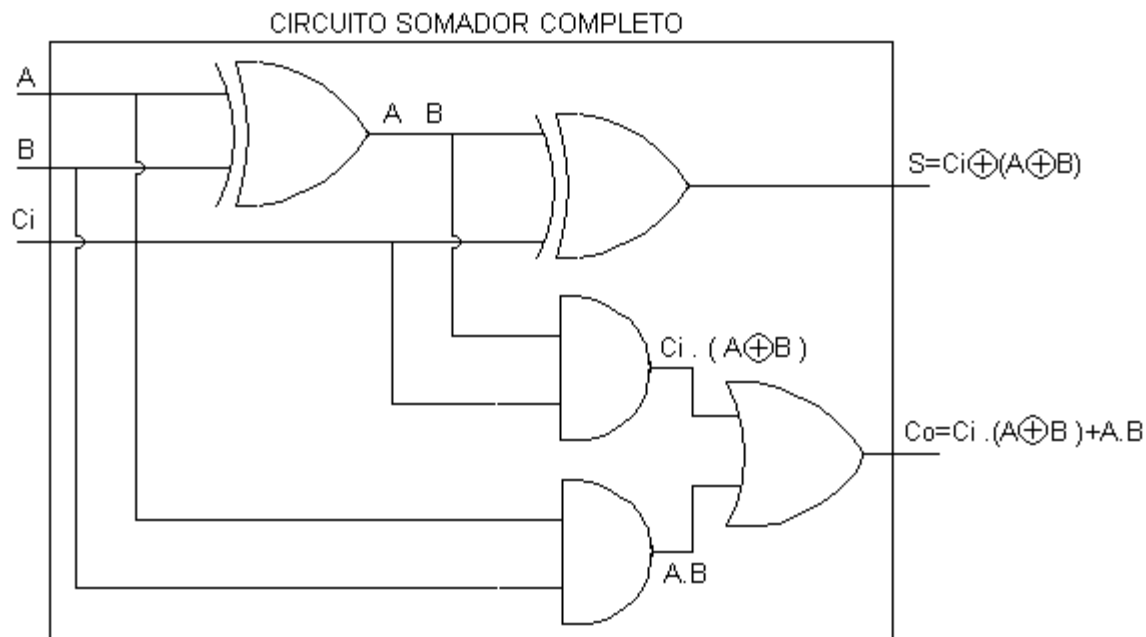
$$\overline{A \oplus B} = A.B + \bar{A}.\bar{B}$$

Outras propriedades:

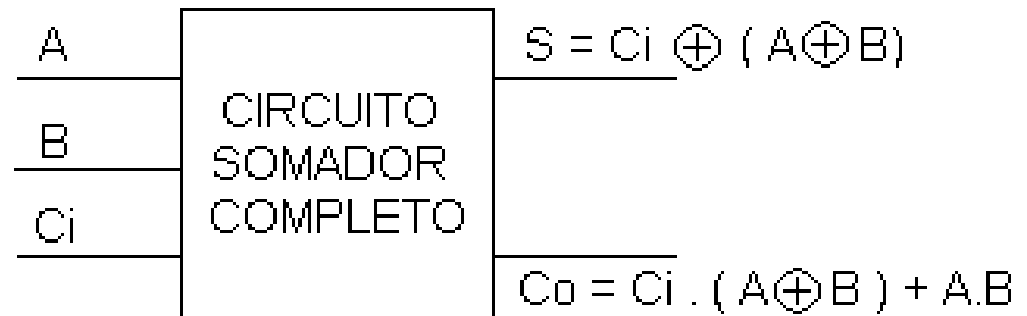
$$A \oplus A = 0$$

$$A \oplus \bar{A} = 1$$

O somador completo: o circuito

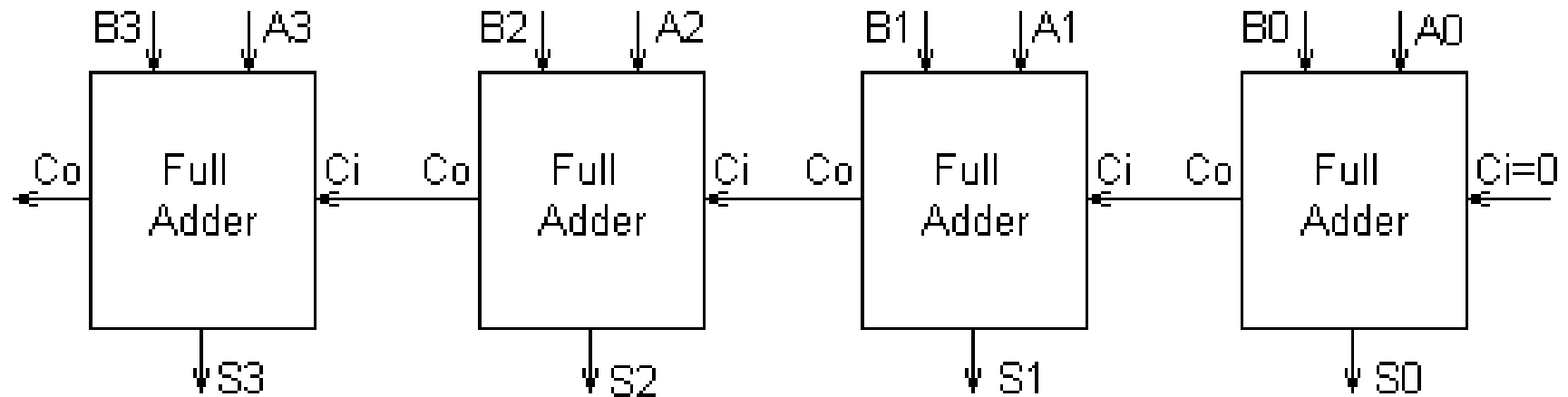


Resumindo para um bloco



Circuito somador de 4 bits utilizado com base em uma ULA:

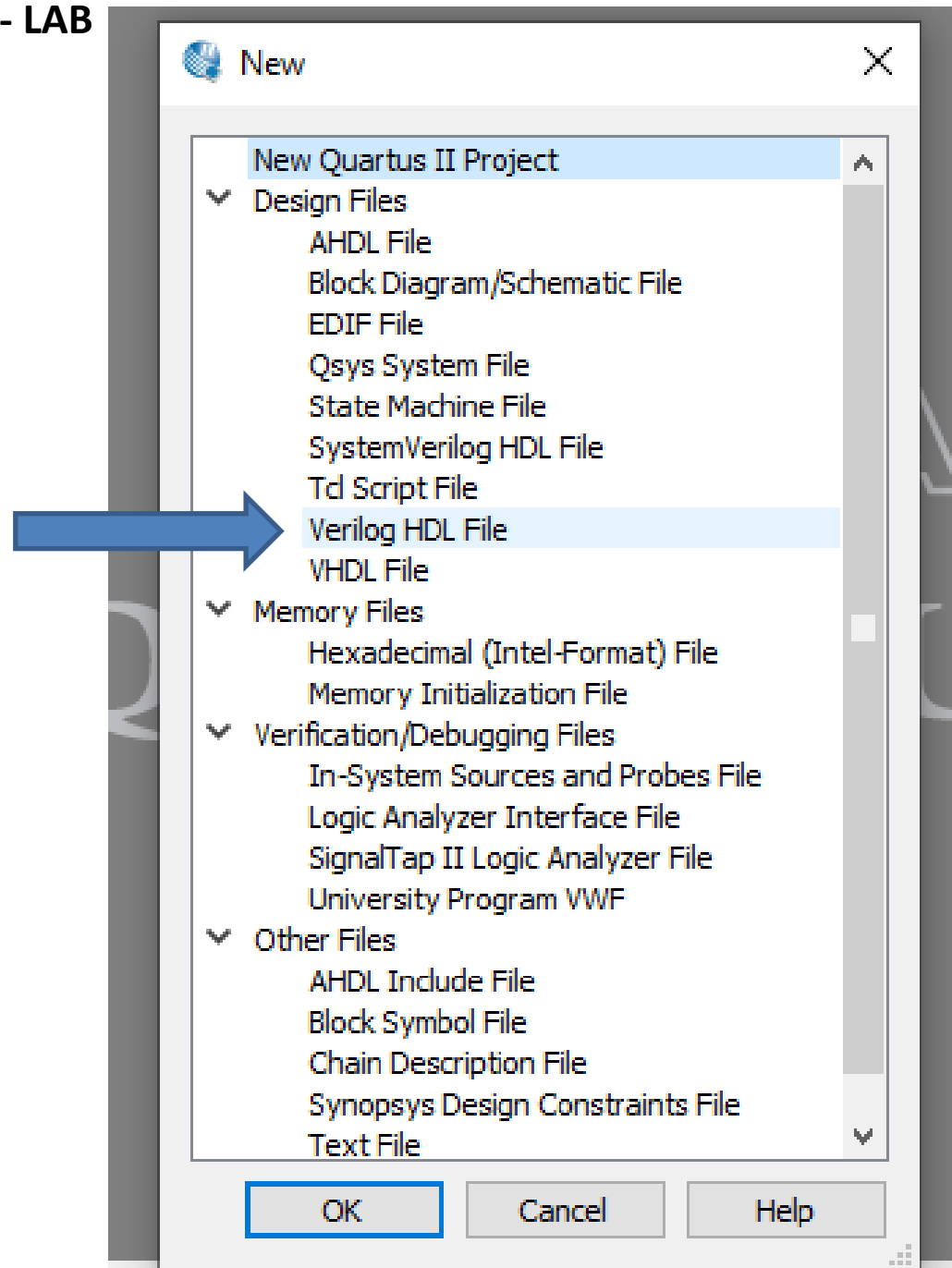
SOMADOR DE 4 BITS



Passos para o experimento:

- Crie um projeto novo, chame o mesmo de **somador**;
- Crie um arquivo do tipo **Verilog HDL**, chame o mesmo de **verilog1**;
 - Com esse arquivo podemos digitar a expressão lógica do somador completo.
 - Utilizar o mesmo como se fosse um bloco.
 - É uma linguagem padrão para definição de hardware.
 - É otimizado para o hardware.

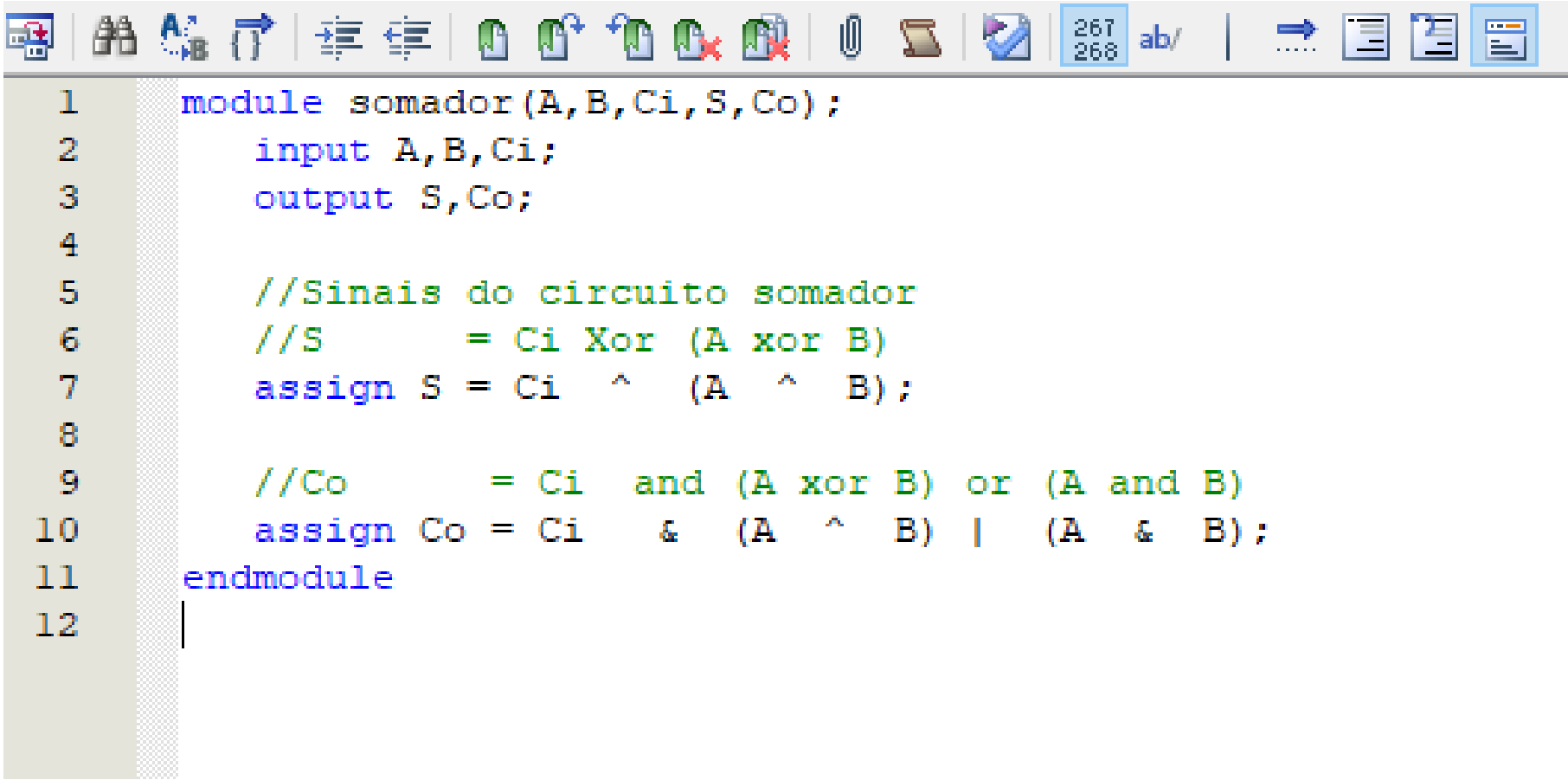
Criando o arquivo Verilog HDL:



Digite o script:

```
module somador(A,B,Ci,S,Co) ;  
    input A,B,Ci;  
    output S,Co;  
  
    //Sinais do circuito somador  
    //S      = Ci Xor (A xor B)  
    assign S = Ci ^ (A ^ B);  
  
    //Co      = Ci and (A xor B) or (A and B)  
    assign Co = Ci & (A ^ B) | (A & B);  
endmodule
```

O script **verilog1.v**



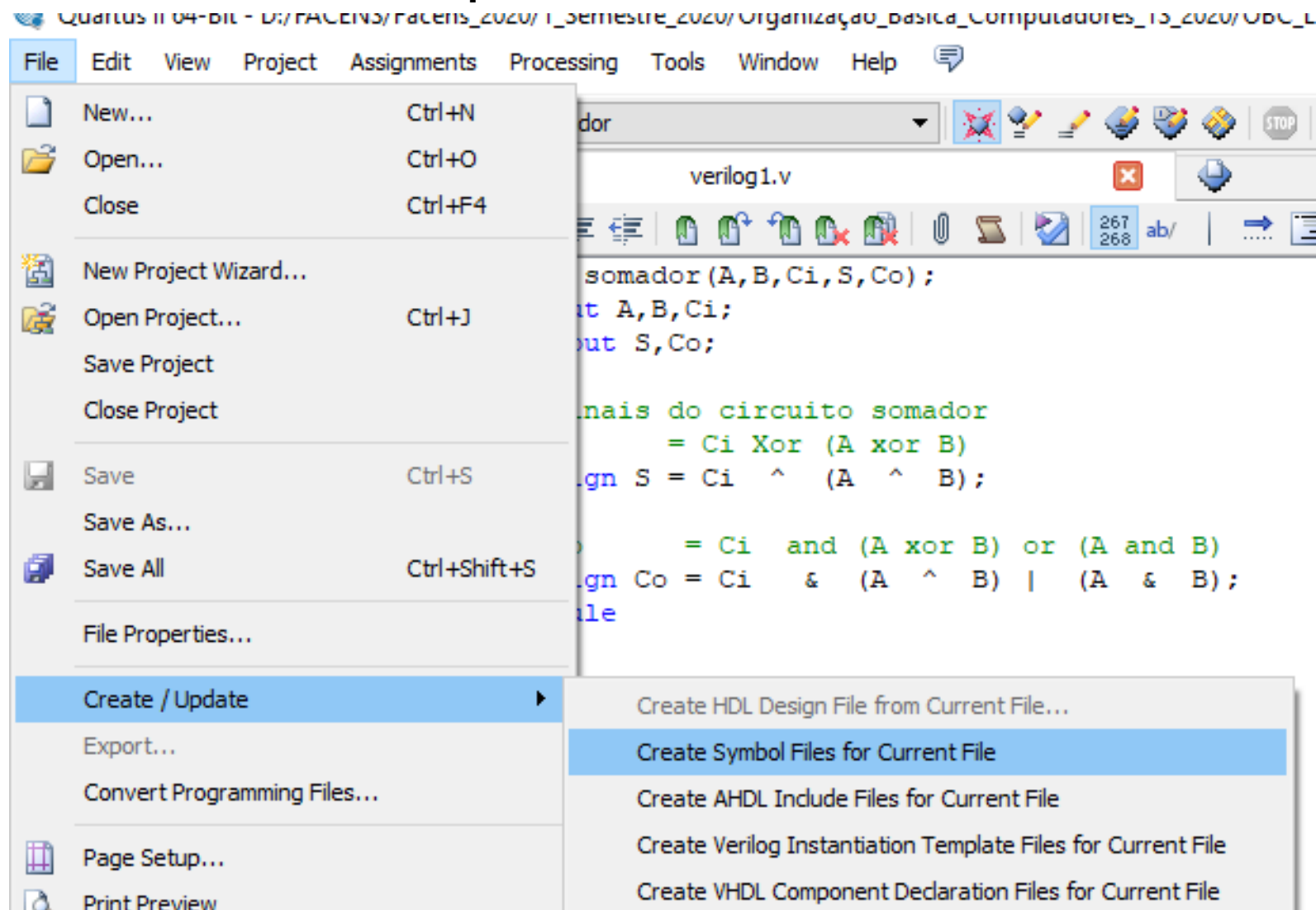
The image shows a screenshot of a Verilog code editor. The editor has a toolbar at the top with various icons for file operations, editing, and simulation. The code is written in a monospaced font and is color-coded: keywords are blue, identifiers are black, and comments are green. The code defines a module named 'somador' with inputs A, B, and Ci, and outputs S and Co. It implements a 4-bit adder using XOR and AND gates. The code is as follows:

```
1  module somador(A,B,Ci,S,Co) ;
2      input A,B,Ci;
3      output S,Co;
4
5      //Sinais do circuito somador
6      //S      = Ci Xor (A xor B)
7      assign S = Ci ^ (A ^ B);
8
9      //Co      = Ci and (A xor B) or (A and B)
10     assign Co = Ci & (A ^ B) | (A & B);
11 endmodule
12
```

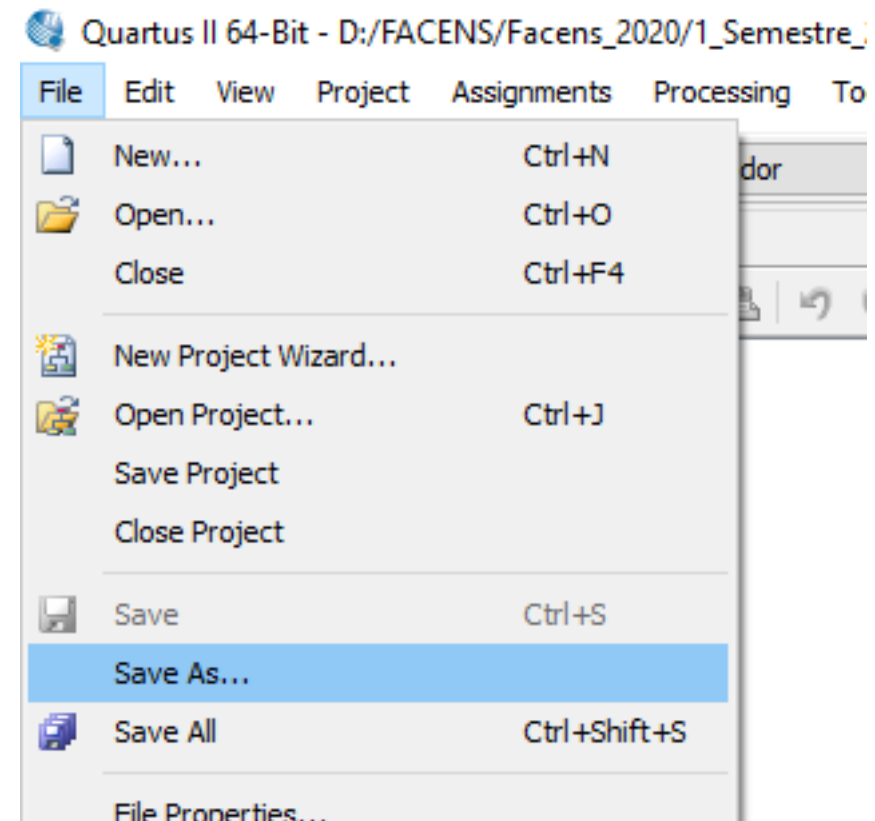
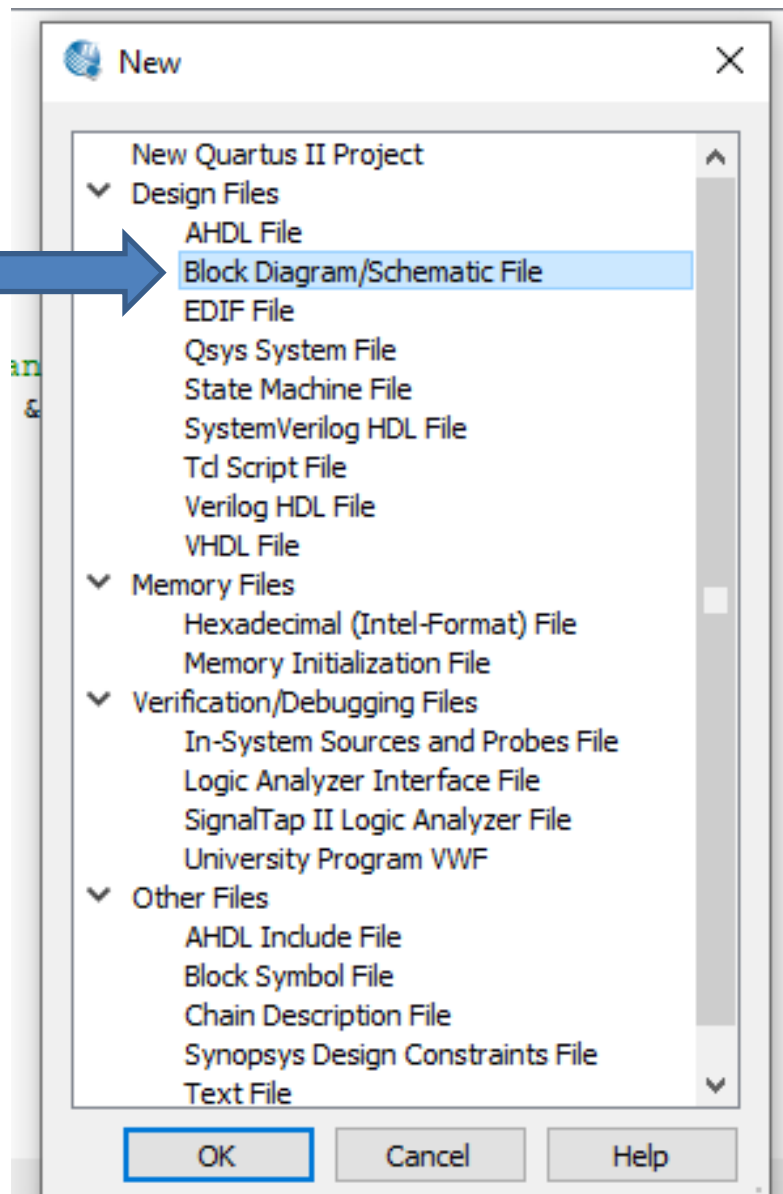
Após digitado o script do componente devemos salvar e compilar;
Para o script virar um Symbol file utilize a sequência de menus abaixo:

File > Create/Update > Create Symbol Files for Current File

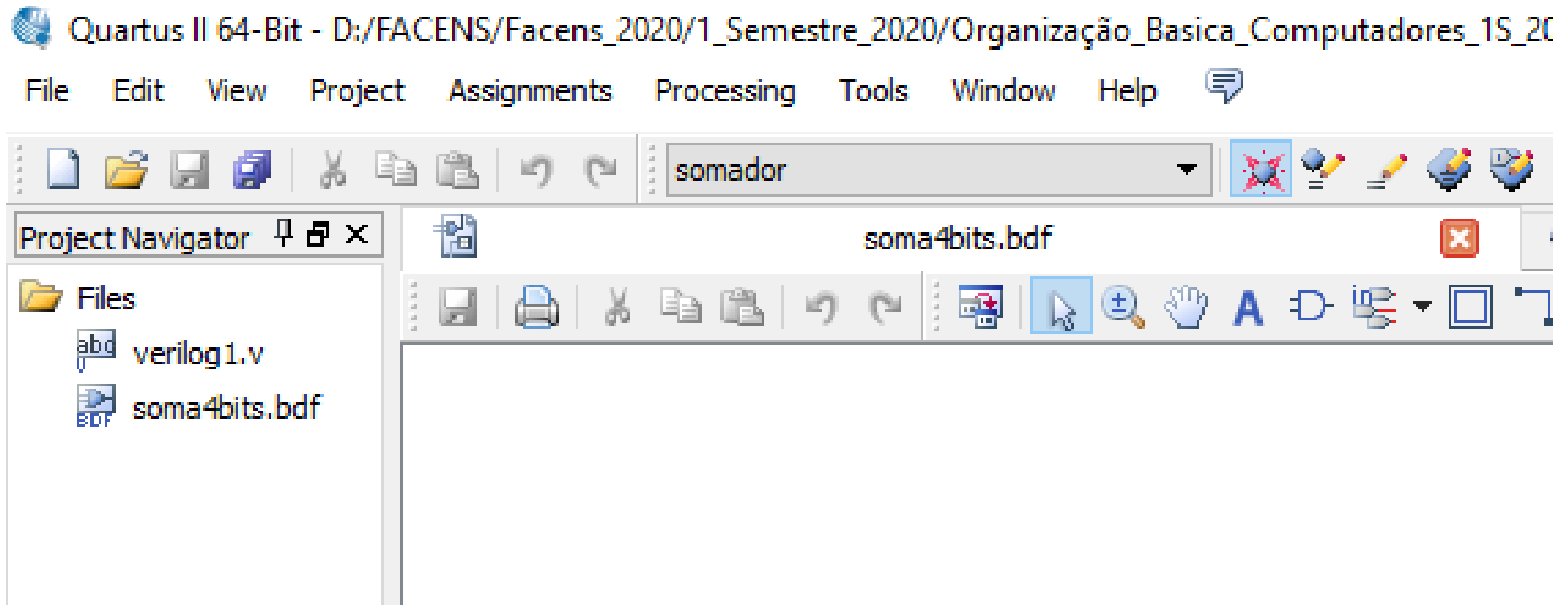
OBS: Mantenha a tela do script aberta.



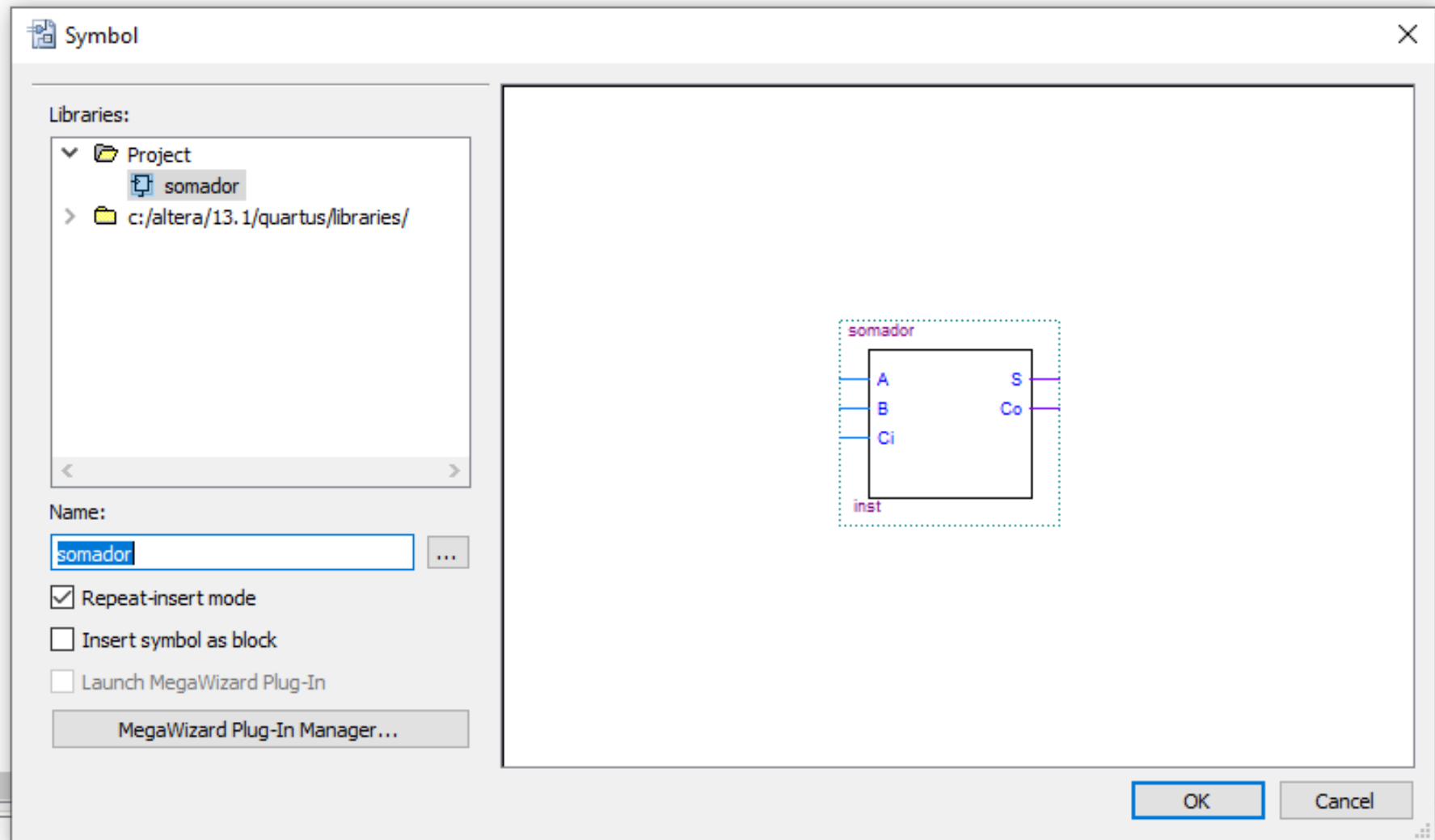
- Crie um arquivo do tipo **Block Diagram/Schematic File** com o nome: **soma4bits**



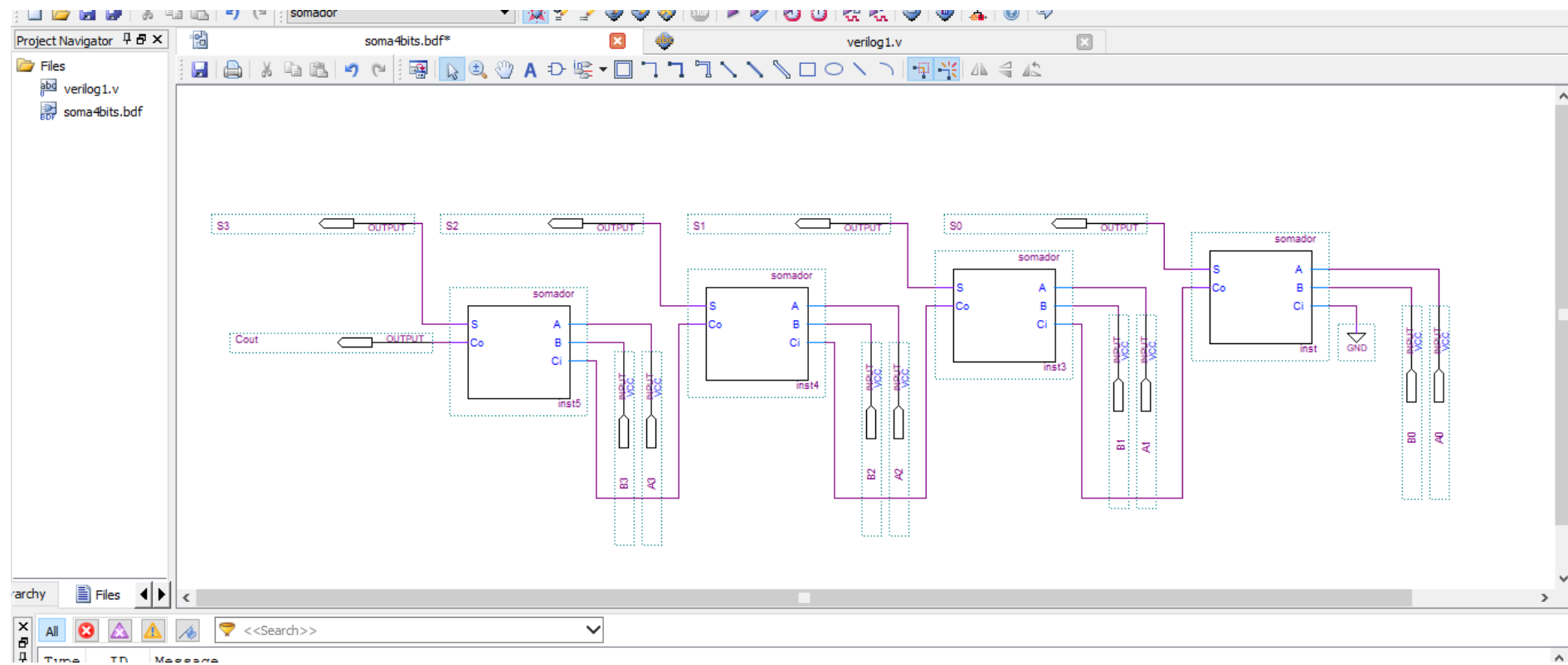
O projeto terá uma estrutura de arquivos conforme indicada abaixo:



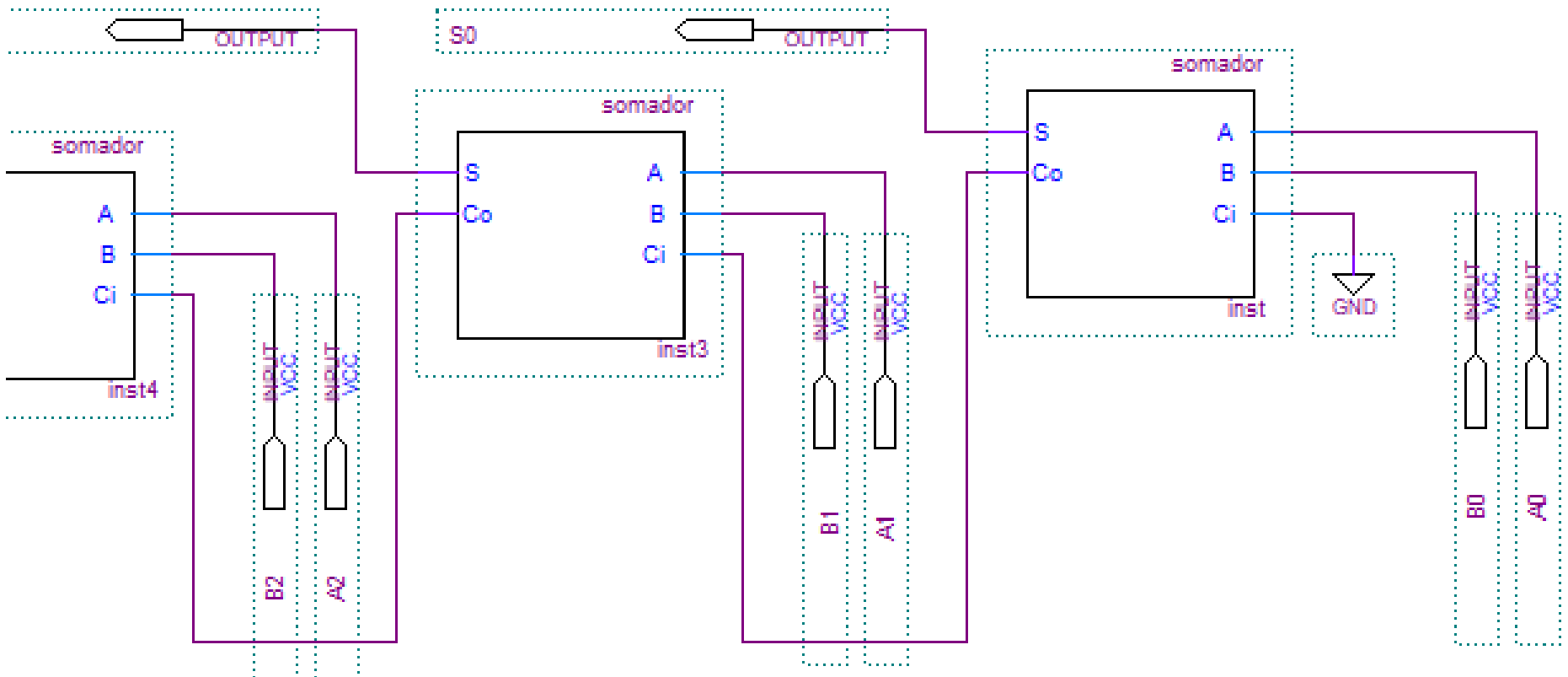
Ao inserir novos componentes, no projeto de desenho de circuitos, o bloco **somador** já se encontra disponível:



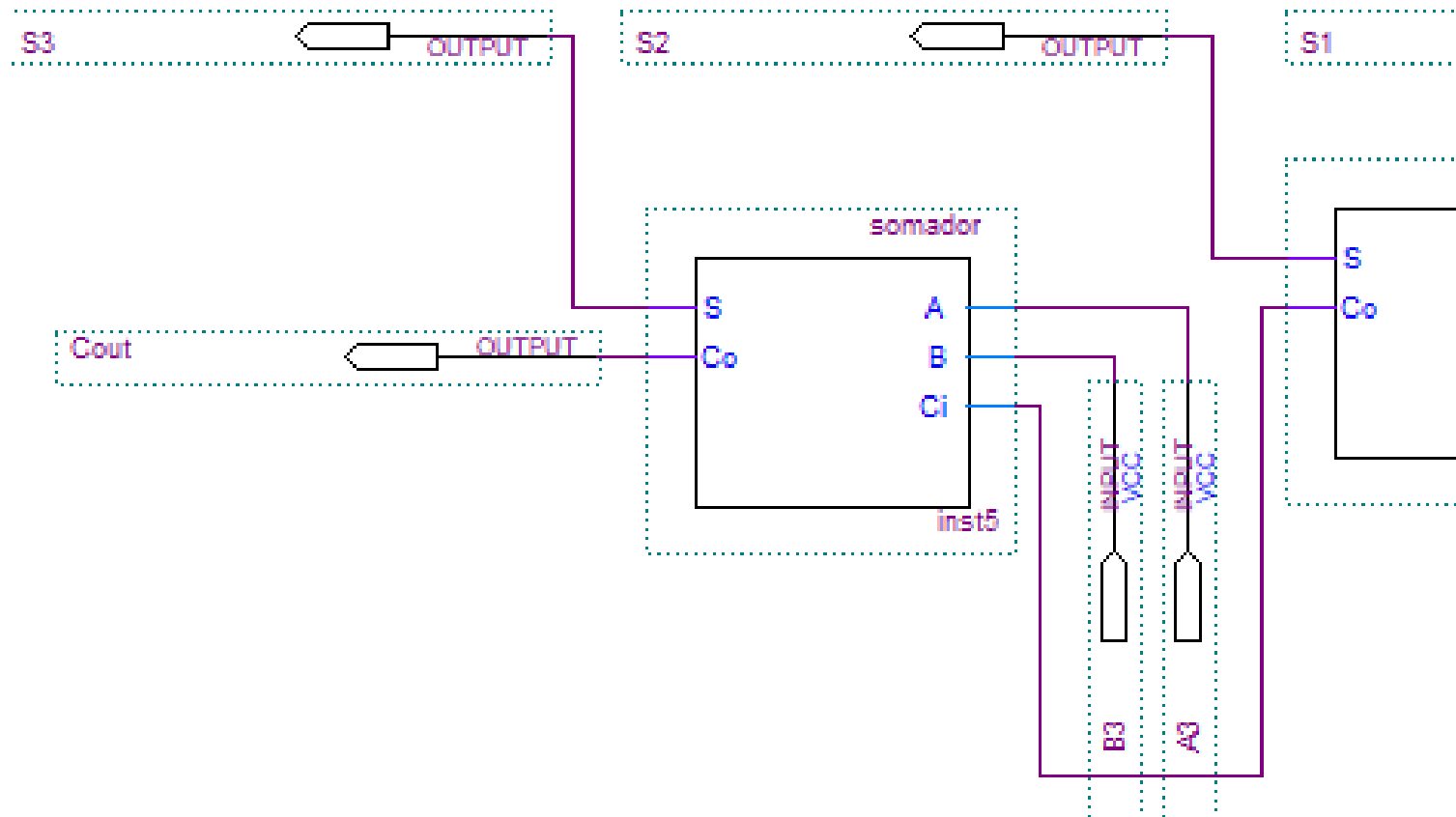
Crie o circuito abaixo no Quartus II:



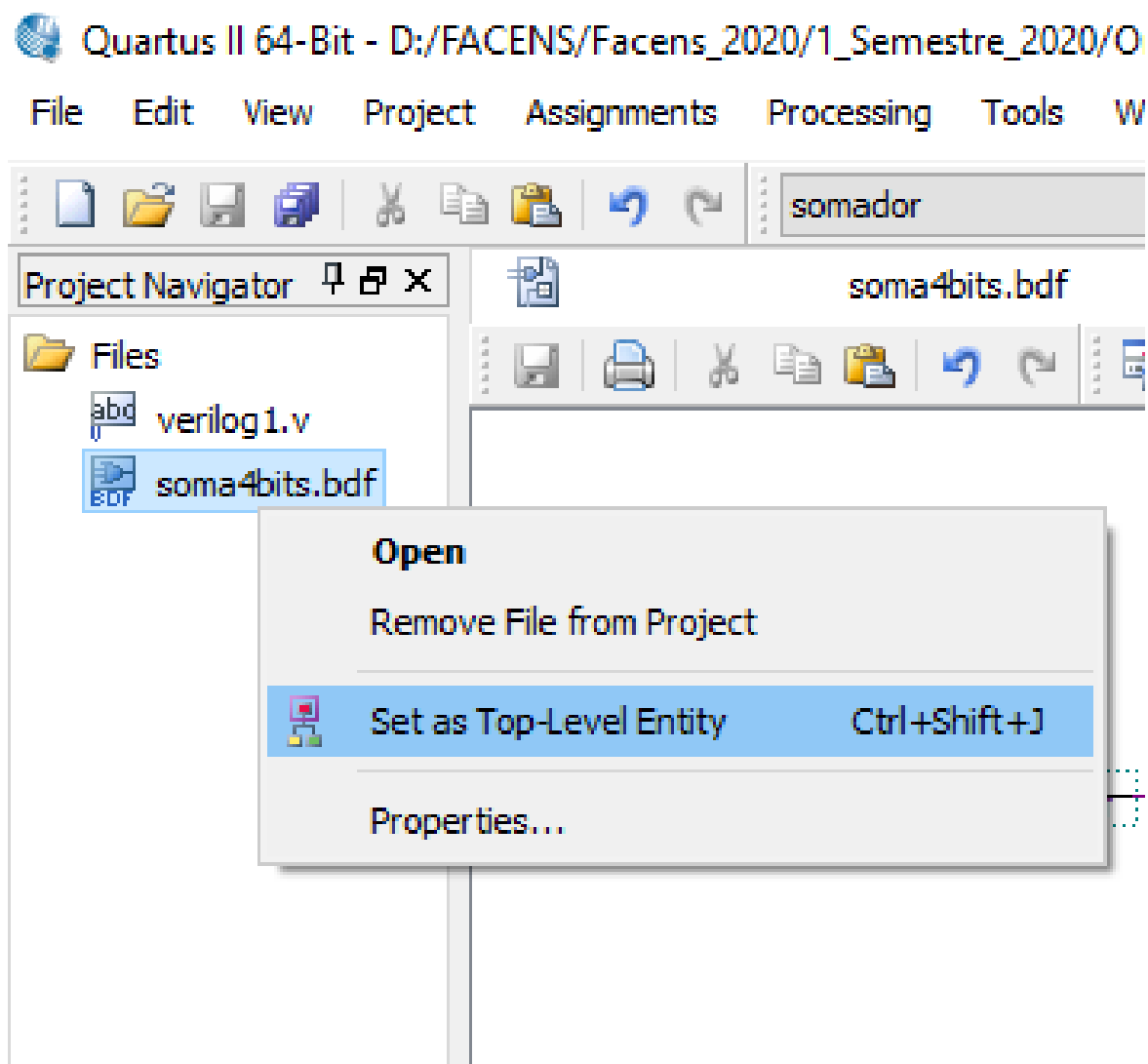
Crie o circuito abaixo no Quartus II: (parte direta)



Crie o circuito abaixo no Quartus II: (parte esquerda)



Criado desenho, salvar o projeto e definir o diagrama como nível superior e por fim compilar!



Definição de pinagem através do **Assignment Editor** (caso não aparecer os pinos criados definir utilizando o **Pin Planner**):

Tabela dos pinos de entrada (chaves on/off)

<i>Signal Name</i>	<i>FPGA Pin No.</i>	<i>Description</i>	<i>I/O Standard</i>
SW[0]	PIN_V28	Slide Switch[0]	2.5V
SW[1]	PIN_U30	Slide Switch[1]	2.5V
SW[2]	PIN_V21	Slide Switch[2]	2.5V
SW[3]	PIN_C2	Slide Switch[3]	2.5V
SW[4]	PIN_AB30	Slide Switch[4]	2.5V
SW[5]	PIN_U21	Slide Switch[5]	2.5V
SW[6]	PIN_T28	Slide Switch[6]	2.5V
SW[7]	PIN_R30	Slide Switch[7]	2.5V
SW[8]	PIN_P30	Slide Switch[8]	2.5V
SW[9]	PIN_R29	Slide Switch[9]	2.5V
SW[10]	PIN_R26	Slide Switch[10]	2.5V
SW[11]	PIN_N26	Slide Switch[11]	2.5V
SW[12]	PIN_M26	Slide Switch[12]	2.5V
SW[13]	PIN_N25	Slide Switch[13]	2.5V
SW[14]	PIN_J26	Slide Switch[14]	2.5V
SW[15]	PIN_K25	Slide Switch[15]	2.5V
SW[16]	PIN_C30	Slide Switch[16]	2.5V
SW[17]	PIN_H25	Slide Switch[17]	2.5V

Definição de pinagem através do **Assignment Editor** (caso não aparecer os pinos criados definir utilizando o **Pin Planner**):

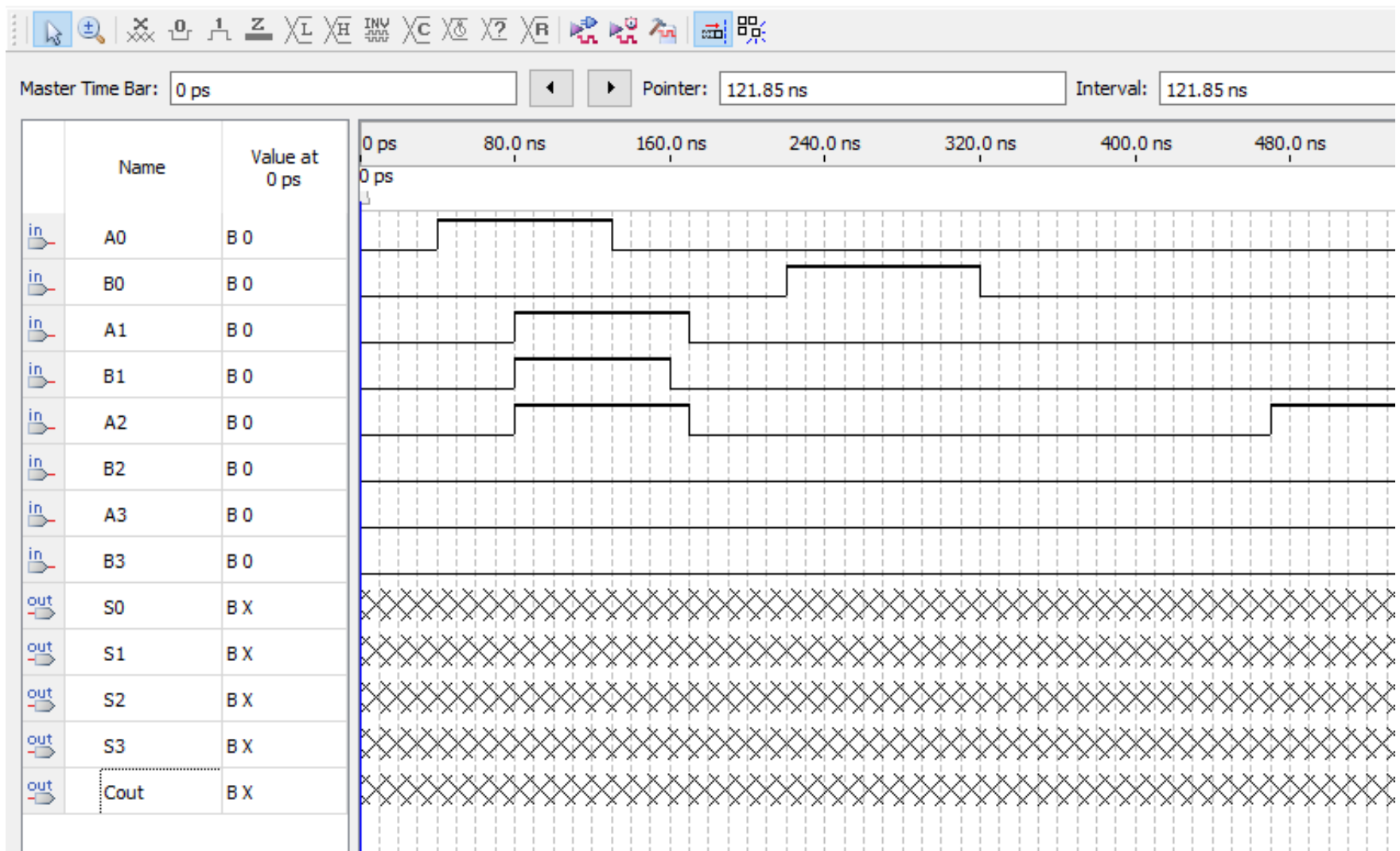
Tabela dos pinos de saída (Leds)

<i>Signal Name</i>	<i>FPGA Pin No.</i>	<i>Description</i>	<i>I/O Standard</i>
LEDR[0]	PIN_T23	LED Red[0]	2.5V
LEDR[1]	PIN_T24	LED Red[1]	2.5V
LEDR[2]	PIN_V27	LED Red[2]	2.5V
LEDR[3]	PIN_W25	LED Red[3]	2.5V
LEDR[4]	PIN_T21	LED Red[4]	2.5V
LEDR[5]	PIN_T26	LED Red[5]	2.5V
LEDR[6]	PIN_R25	LED Red[6]	2.5V
LEDR[7]	PIN_T27	LED Red[7]	2.5V
LEDR[8]	PIN_P25	LED Red[8]	2.5V
LEDR[9]	PIN_R24	LED Red[9]	2.5V

Crie as formas de onda, conforme EXEMPLO abaixo (reorganizar as posições das entradas e saídas):

Simulation Waveform Editor - D:/FACENS/Facens_2020/1_Semestre_2020/Organização_Basica_Computadores_1S_2020/OBC_LABOTARÓRIO_1S_2020/E

File Edit View Simulation Help



- Após a geração do gráfico conectar o KIT da INTEL ao computador e carregar o programa (circuito elaborado) para dentro do FPGA.

Fazer as devidas configurações dos pinos das entradas e saídas do kit e fazer o teste prático.

Demonstrar ao professor o funcionamento.

Relatório 6

- Introdução
- Construção do circuito somador para ULA no programa Quartus II
 - Definição do circuito no software
- Procedimento experimental executado
- Demonstração com forma de onda na execução do circuito
 - Para modelo de simulação *funcional*
 - Para modelo de simulação *timing*
 - Análise as formas de onda nos dois casos acima descritos
- Conclusão