

Criando e Simulando Circuitos Digitais no Quartus

Prof. Felipe W. D. Pfrimer

25 de março de 2024

Sumário

1	Introdução	3
2	Objetivo do Tutorial	3
3	Conceitos importantes	3
3.1	O que é Simulação Funcional?	3
3.2	O Quartus Prime	3
4	Somadores <i>carry ripple</i>	4
5	Instalação	4
5.1	Download do Software	5
5.2	Instalação do Software	5
5.3	Inicialização do Programa	5
6	Criando um novo projeto	6
7	Primeiro esquemático	8
8	Criando um bloco lógico	11
9	Reaproveitando o bloco criado	13
10	Análise e Síntese	17
11	Simulação funcional	18

1 Introdução

O IntelFPGA Quartus Prime representa uma plataforma abrangente de desenvolvimento para Dispositivos Lógicos Programáveis (PLDs), oferecida pela IntelFPGA. Este Ambiente de Desenvolvimento Integrado (IDE) capacita os desenvolvedores na criação, análise, e síntese de sistemas digitais, empregando Linguagens de Descrição de Hardware (HDLs) ou diagramas esquemáticos. A versão Lite do Quartus Prime, especificamente a 18.1, está disponível gratuitamente e pode ser obtida diretamente através do site da Intel, usando a macro LaTeX para URLs como segue: <https://www.intel.com.br/content/www/br/pt/products/details/fpga/development-tools/quartus-prime/resource.html>.

Este tutorial é direcionado a fornecer ao leitor uma introdução prática ao desenvolvimento e simulação de sistemas digitais utilizando diagramas esquemáticos no Quartus Prime Lite 18.1. Embora nosso foco esteja na versão 18.1, é importante notar que os princípios e procedimentos aqui descritos podem ser aplicáveis a outras versões do software, podendo haver pequenas alterações dependendo da versão utilizada.

2 Objetivo do Tutorial

O objetivo principal deste tutorial é guiar o leitor através do processo de simulação funcional de um somador *carry ripple* de 4 bits, o qual será projetado utilizando diagramas de blocos. Através deste guia, você aprenderá como configurar, implementar e simular um circuito digital básico dentro do ambiente de desenvolvimento do Quartus Prime Lite 18.1, focando especificamente na versão funcional de simulação.

3 Conceitos importantes

Essa seção trata de alguns conceitos importantes para o entendimento deste tutorial.

3.1 O que é Simulação Funcional?

A simulação funcional é uma técnica fundamental no design de circuitos digitais, permitindo aos designers verificar e analisar o comportamento de um circuito sem a necessidade de construí-lo fisicamente. Este método de simulação envolve a execução de um modelo de software do circuito para testar suas funções sob várias condições de entrada, assegurando que ele atenda às especificações desejadas antes de prosseguir para a etapa de implementação física.

Na simulação funcional, o foco está em validar a lógica e as operações do circuito, sem levar em conta as características físicas dos componentes, como atrasos de tempo ou consumo de energia. Isso permite uma verificação rápida e eficiente do design em nível lógico, facilitando a detecção e correção de erros em estágios iniciais do desenvolvimento do projeto.

Ao concluir este tutorial, você terá adquirido conhecimentos práticos sobre como realizar simulações funcionais de circuitos digitais, usando o exemplo prático de um somador de 4 bits para ilustrar os conceitos e procedimentos envolvidos. Este conhecimento será valioso para futuros projetos de design de circuitos, permitindo uma abordagem mais eficaz e eficiente na verificação de designs antes da implementação.

3.2 O Quartus Prime

O Quartus Prime [1] é uma IDE (*Integrated Development Environment*) desenvolvida pela Intel, anteriormente pela Altera, para o design, simulação, programação e depuração de sistemas

digitais integrados, incluindo FPGAs (*Field-Programmable Gate Arrays*), CPLDs (*Complex Programmable Logic Devices*) e SoCs (*Systems on Chips*). Como uma IDE, o Quartus Prime integra ferramentas e funcionalidades ao desenvolvimento de hardware, auxiliando engenheiros e projetistas na criação de sistemas eletrônicos.

Uma IDE é um software que oferece ferramentas para programadores e desenvolvedores, como edição de código, compilação, depuração e, em alguns casos, simulação. A principal função de uma IDE é proporcionar um ambiente que otimize a produtividade, reduzindo a necessidade de alternar entre diferentes aplicações durante o desenvolvimento de software ou hardware. Uma IDE serve como uma plataforma onde desenvolvimento, testes e depuração ocorrem de forma integrada.

No contexto do Quartus Prime, a IDE se especializa no design e implementação de componentes eletrônicos programáveis. Permite aos usuários elaborar esquemáticos, escrever e testar códigos em linguagens de descrição de hardware como VHDL e Verilog, e simular o comportamento de circuitos digitais antes da implementação física. Adicionalmente, o Quartus Prime simplifica a programação dos dispositivos com os designs desenvolvidos, sendo uma ferramenta importante na engenharia eletrônica para o desenvolvimento de sistemas digitais.

4 Somadores *carry ripple*

Um somador digital é um circuito que executa a soma de dois números binários e considera um *carry-in* (um bit adicional de uma soma anterior). Este circuito é utilizado na construção de sistemas de computação e processamento digital, habilitando operações aritméticas básicas[2, 3].

Um somador *carry ripple* é construído pela concatenação de somadores completos de 1 bit, onde cada somador processa dois bits e um *carry-in*, resultando em um bit de soma e um *carry-out*. A principal limitação dessa abordagem reside na velocidade de operação. Especificamente, o tempo de cálculo de uma soma aumenta proporcionalmente ao número de bits envolvidos, devido à necessidade de esperar a propagação do *carry-out* de um somador para o *carry-in* do somador subsequente [4].

Esta propagação sequencial do *carry* implica que o bit de *carry* precisa “viajar” por todos os somadores na cadeia, um após o outro, antes que a soma total possa ser completada. Assim, para somas de números com muitos bits, o atraso total introduzido pela propagação do *carry* pode se tornar significativo, impactando negativamente o desempenho da operação de soma em aplicações que demandam alta velocidade e baixa latência. No entanto, devido a sua simplicidade, esse tipo de abordagem é ideal para os propósitos deste guia.

Ao conectar quatro somadores completos em sequência, pode-se somar dois números de 4 bits. O primeiro somador recebe os bits menos significativos dos números a serem somados, junto com o *carry-in* inicial (geralmente 0). O *carry-out* deste somador é passado como *carry-in* para o somador subsequente, até o último somador, cujo *carry-out* representa o *carry-out* final da operação de soma.

Esta metodologia não apenas simplifica o design, mas também permite escalabilidade, facilitando a expansão para somadores de maiores dimensões através do aumento do número de somadores completos concatenados.

5 Instalação

Essa seção do tutorial destina-se a auxiliar na instalação do Quartus Prime Lite 18.1 em sistemas operacionais Windows 10 ou 11. Caso já possua o Quartus instalado, vá para a seção 6.

5.1 Download do Software

1. Acesse a página oficial de downloads da Intel para o Quartus Prime Lite 18.1 por meio do link: <https://www.intel.com.br/content/www/br/pt/products/details/fpga/development-tools/quartus-prime/resource.html>;
2. Na página, procure pela seção de downloads do **Quartus Prime Lite Edition**;



Certifique-se de escolher a versão Lite 18.1.

3. Selecione a versão apropriada para Windows e clique no grande botão azul botão de download, na aba de *Multiple Download*, para baixar todos os arquivos necessários;
4. Aguarde baixar o arquivo de extensão .tar;
5. Pode ser necessário criar uma conta Intel ou fazer login com uma conta existente para prosseguir com o download.

5.2 Instalação do Software

1. Uma vez concluído o download, localize o arquivo baixado e execute-o com um duplo clique;
2. Se for exibida uma janela pedindo permissão para que o aplicativo faça alterações no seu dispositivo, clique em “Sim”;
3. Siga as instruções apresentadas pelo assistente de instalação. Aceite os termos de licença e selecione o diretório de instalação conforme desejado (de preferência, deixe o diretório proposto);
4. Durante a instalação, você pode selecionar os componentes específicos do Quartus que deseja instalar (dispositivos). Para uma instalação padrão, é recomendável deixar as opções pré-selecionadas;



Tenha certeza de manter a família MAX10 e o modelsim IntelFPGA Edition ou Altera Edition. O modelsim padrão é um aplicativo que exige a compra de uma licença.

5. Após configurar suas preferências, prossiga com a instalação e aguarde até que o processo seja concluído.

5.3 Inicialização do Programa

1. Após a conclusão da instalação, você pode iniciar o Quartus Prime Lite 18.1 através do menu Iniciar do Windows, procurando por "Quartus Prime Lite" ou através do atalho criado na área de trabalho, se disponível.
2. Na primeira execução, pode ser necessário configurar algumas preferências iniciais ou realizar o registro do software, dependendo das exigências do programa.
3. Com o Quartus Prime Lite aberto, você está agora pronto para começar a criar e simular seus projetos de sistemas digitais.

Este guia deve ajudar você a instalar e iniciar o Quartus Prime Lite 18.1 em sua máquina Windows com facilidade. Para qualquer suporte adicional ou questões técnicas, referencie a documentação oficial do Quartus ou os fóruns de suporte da Intel.

6 Criando um novo projeto

A Figura 1 ilustra a tela inicial do Quartus Prime Lite 18.1.



É crucial seguir os próximos passos exatamente como descritos e prestar atenção especial aos nomes dos arquivos mencionados. A precisão é essencial para o sucesso da configuração.

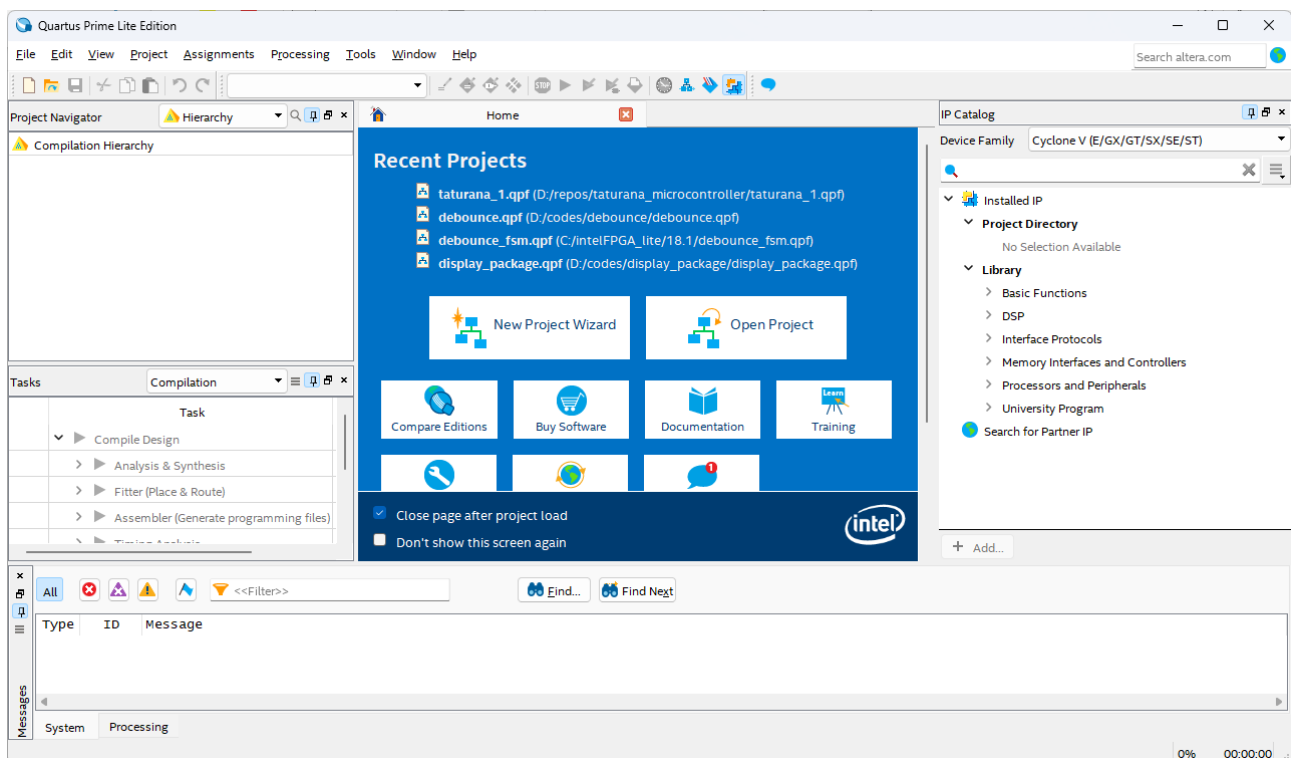


Figura 1: Tela inicial do Quartus Prime Lite 18.1.

Um novo projeto pode ser criado clicando em **New Project Wizard** ou acessando o menu **File → New Project Wizard**. O *Project Wizard* será aberto em uma nova janela para configurar o projeto, como ilustrado na Figura 2. Nesta primeira tela, uma breve introdução sobre a criação de projetos será apresentada. Assim, basta clicar em **Next** e uma nova tela se abrirá, conforme mostrado na Figura 3, onde será possível escolher o diretório do projeto.

Ao configurar um novo projeto no Quartus Prime Lite 18.1, é essencial prestar atenção aos seguintes campos na tela de configuração do projeto:

1. No primeiro campo, selecione o diretório que armazenará os arquivos do projeto. É necessário escolher um caminho de diretório (o caminho completo) livre de espaços e caracteres especiais da língua portuguesa, como cedilha (ç) e vogais acentuadas. Para nosso exemplo, o diretório será nomeado como `somador_de_4_bits`, já que o objetivo é criar um circuito somador.

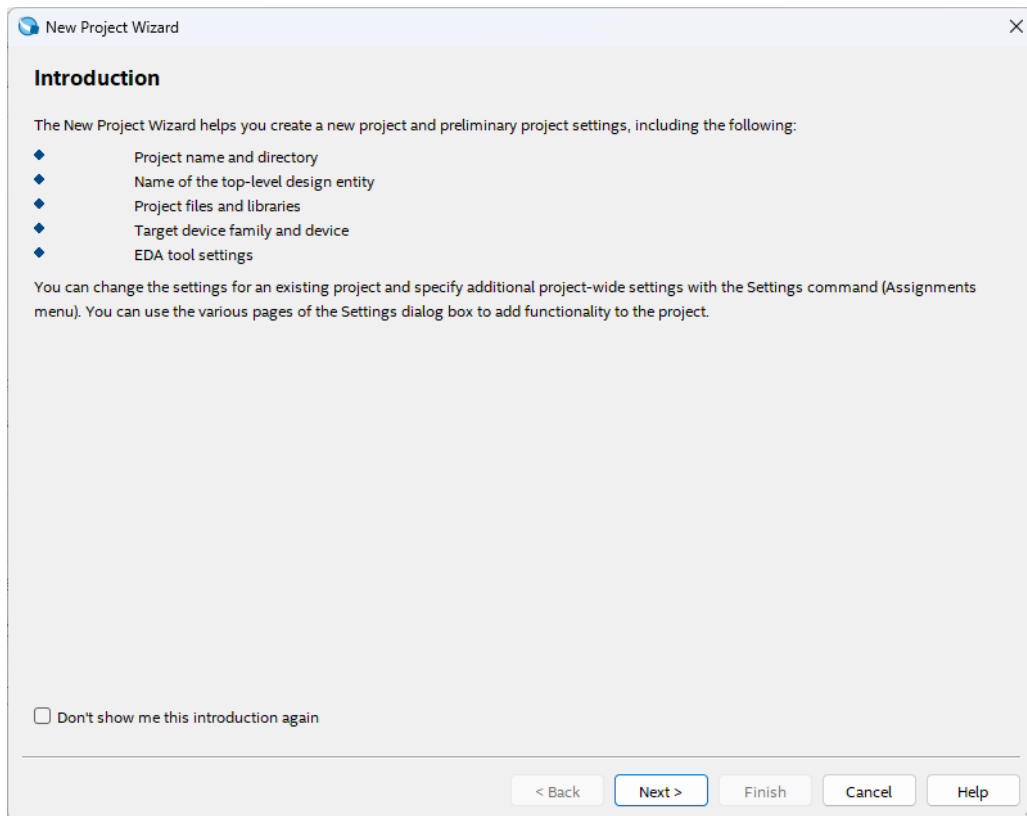
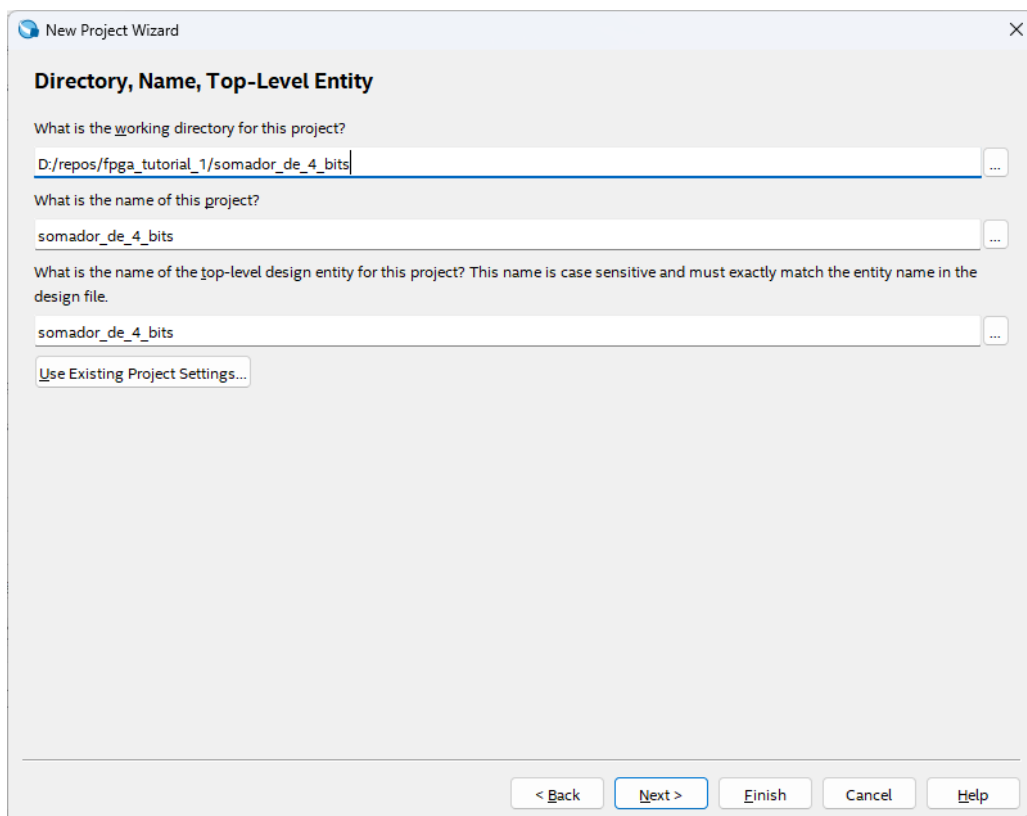


Figura 2: Tela inicial do New Project Wizard.

Figura 3: Tela para escolha do diretório, nome do projeto e *Top Level*.

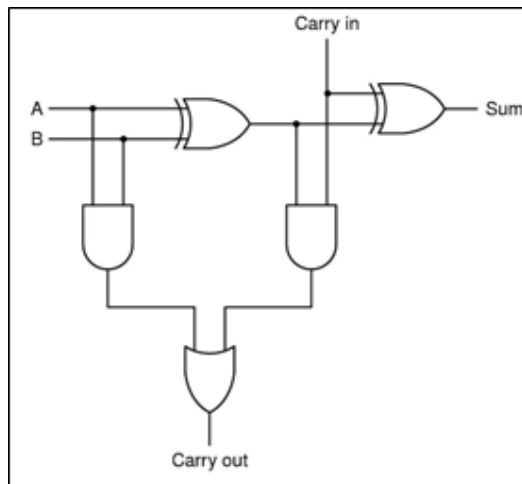
2. Na interface apresentada na Figura 3, insira o nome do projeto no segundo campo: `somador_de_4_bits`. Observe que o terceiro campo é preenchido automaticamente com o mesmo nome do projeto. Este campo refere-se ao nome do arquivo principal do projeto (*top-level design*). Embora seja possível modificar o nome no terceiro campo, evite iniciar o nome do arquivo principal com números, como em `4_bit_adder`.

Após ajustar esses detalhes, clique em **Finish**. Assim, um projeto vazio será criado. Dado que este exemplo foca em uma simulação funcional, a seleção de um dispositivo específico para programação não é exigida, algo que seria feito em telas de configuração posteriores; o programa automaticamente escolherá um dispositivo padrão. O próximo passo envolve a criação do primeiro arquivo esquemático do projeto.

7 Primeiro esquemático

Para este tutorial, vamos criar um somador *carry ripple* de 4 bits. Contudo, a elaboração de tal somador requer a construção de um circuito **somador completo**, que será replicado quatro vezes no circuito final. Para mais informações sobre somadores digitais, consulte a referência [3].

O circuito de um somador completo, bem como sua tabela verdade, estão ilustrados na Figura 4 e Tabela 1, respectivamente. Observe que o circuito possui três entradas (*A*, *B* e *Carry in*) e duas saídas (*Sum* e *Carry out*). Além disso, é composto por cinco portas lógicas: duas XOR, duas AND e uma OR. Neste ponto, é interessante que o leitor tente deduzir o circuito lógico a partir da tabela verdade, utilizando métodos de simplificação de expressões booleanas, embora isso seja opcional.



A	B	Carry_in	Sum	Carry_out
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

Tabela 1: Tabela verdade de um somador completo

Figura 4: Circuito do somador completo.

Para a criação do esquemático do somador completo, clique em **File** → **New** e uma janela se abrirá com várias opções, conforme ilustrado na Figura 5. Escolha a opção **Block Diagram/Schematic File** e clique em **OK**. Um ambiente para edição de circuitos será aberto.

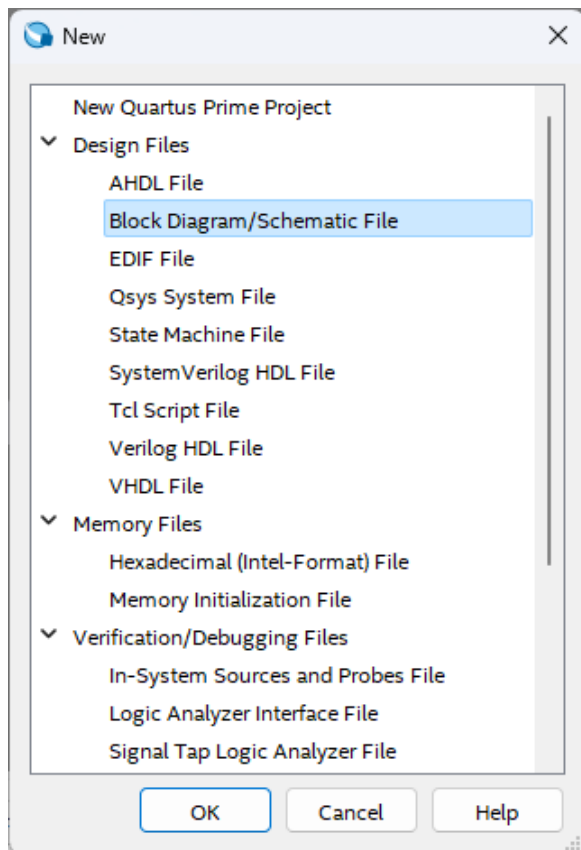


Figura 5: Opções de novos arquivos.

Ícone	Descrição
	Selection tool: utilizado para selecionar objetos.
	Zoom tool: utilizado para aproximar o afastar o desenho. Clique para aproximar e segure shift e clique para afastar.
	Hand tool: utilizado para arrastar o desenho pela tela.
	Text tool: utilizado para inserir comentários na figura.
	Symbol tool: utilizado para inserir componentes no esquemático.
	Pin tool: utilizado para inserir entradas e saídas.
	Orthogonal node and bus tools: utilizados para ligar nós de circuitos e representar barramentos.

Tabela 2: Principais ferramentas do editor de diagrama de blocos e suas funções

Neste novo ambiente será possível criar um circuito digital do somador completo. Repare na barra de ferramentas na parte superior do ambiente de edição. As principais ferramentas desse ambiente estão listadas na Tabela 2.

Primeiramente, acrescente e nomeie as entradas e saídas do somador completo. Para isso, clique no **Pin tool** da barra de ferramentas (). Surgirão três opções:

- In → terminal de apenas entrada;
- Out → terminal de apenas saída;
- Bidir → terminal bidirecional.

Acrescente três terminais de entrada e dois de saída. O esquemático deverá ficar de acordo com a Figura 6. Dê um clique duplo no nome do terminal para editá-lo. Alternativamente, clique com o botão direito do mouse sobre o símbolo do terminal e, em seguida, clique em **properties** e edite o nome no campo **pin name(s)**.

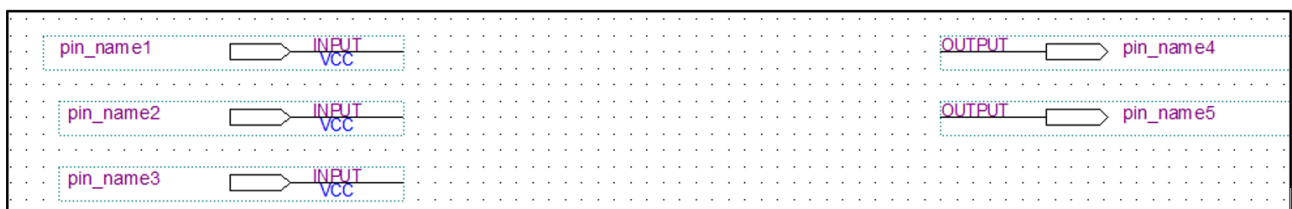


Figura 6: Pinos de entrada e saída do somador completo antes da nomeação dos terminais.

Para editar os nomes dos terminais de acordo com o projeto, utilize o sobrescrito para representar espaços quando houver. Por exemplo, um terminal nomeado como **Carry in** deve

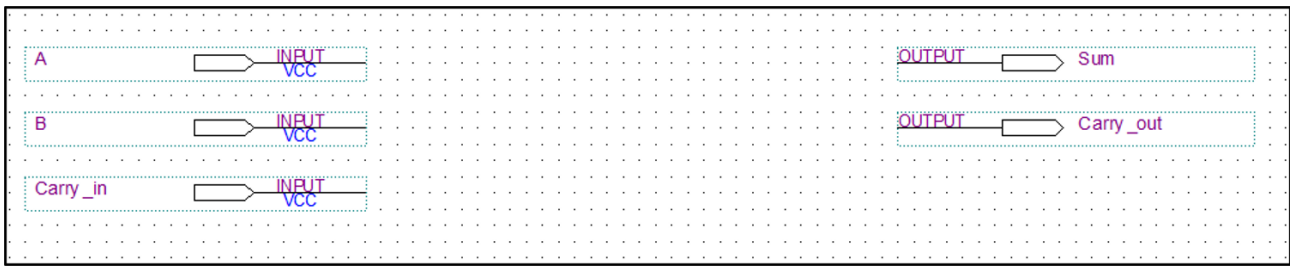


Figura 7: Pinos de entrada e saída do somador completo antes da nomeação dos terminais.

ser editado para `Carry_in`, pois o programa não aceita espaços para o nome dos terminais. Seu esquemático deverá ficar parecido com o ilustrado na Figura 7.

Para acrescentar as portas lógicas ou outros dispositivos básicos, clique em **Symbol tool** (ícone de chave de fenda). Uma janela deverá ser exibida, conforme ilustrado na Figura 8. No campo **Libraries**, navegue, seguindo o indicado na Figura 8, na pasta de bibliotecas do Quartus até chegar em **logic** (subpasta de **primitives**). Esta pasta contém exclusivamente portas lógicas.

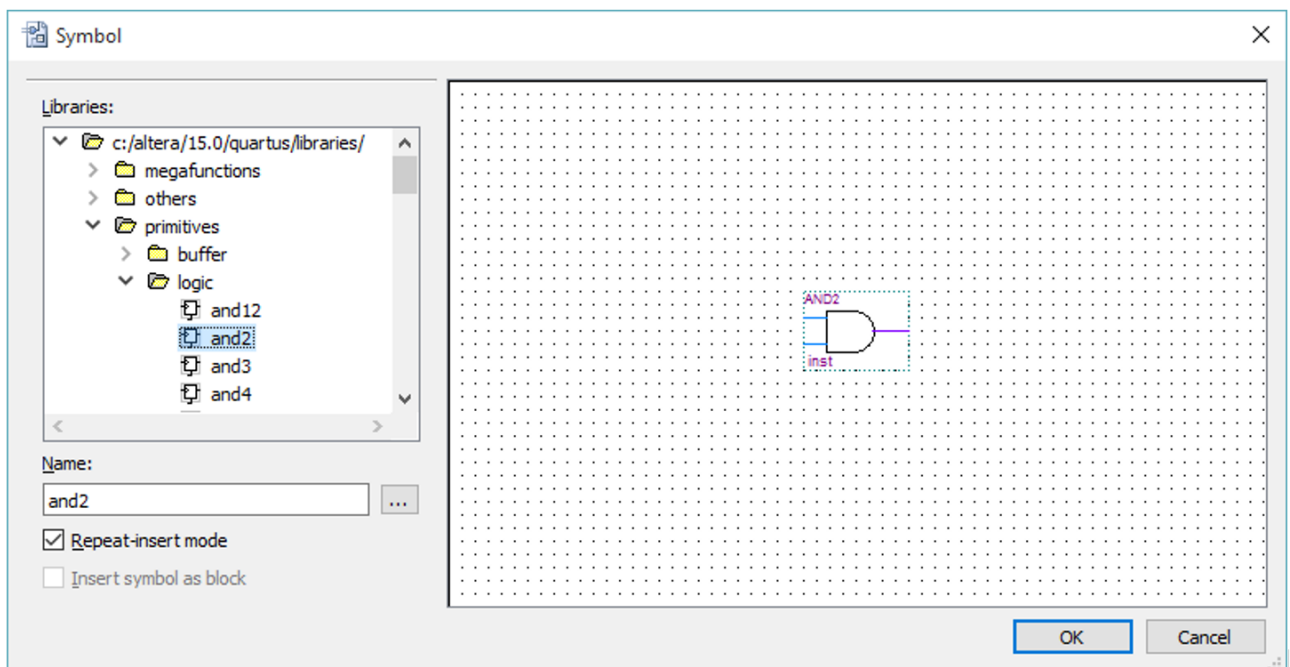


Figura 8: Janela para inserção de símbolos digitais (**Symbol tool**).

Alternativamente, é possível especificar o nome do dispositivo lógico diretamente no campo **Name** da Figura 8. O dispositivo será identificado apenas se o nome completo for inserido. Por exemplo, ao digitarmos `and2` no campo **Name**, seu símbolo aparecerá no lado direito da janela, conforme ilustrado na Figura 8.

Para adicionar uma porta **XOR** ao seu projeto, selecione-a e clique em **OK**. O cursor do mouse será substituído pelo símbolo de uma porta **XOR**, indicando que ela pode ser posicionada em qualquer local da área de edição de circuitos. Para inserir a porta **XOR** no esquemático, clique na região desejada. Observe que o cursor permanece como o símbolo de uma porta **XOR**, permitindo a adição de uma segunda porta com outro clique no esquemático. O desenho resultante deverá assemelhar-se à Figura 9.

Acrescente as outras portas necessárias para o circuito somador da mesma maneira que as portas **XOR** foram inseridas. Seu esquemático deverá assemelhar-se à Figura 10. Para girar um símbolo no esquemático, clique com o botão direito do mouse sobre o símbolo desejado e

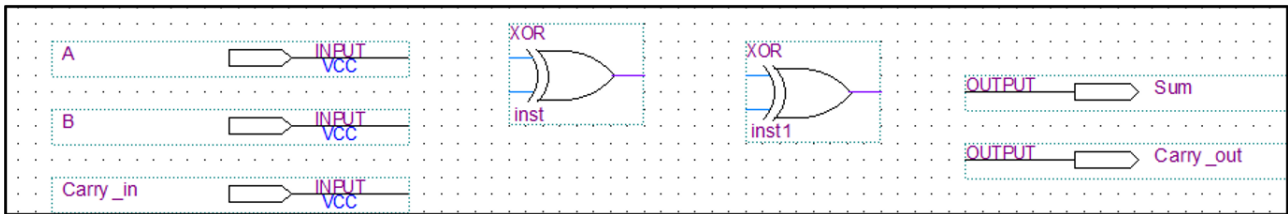


Figura 9: Esquemático com a adição das portas XOR.

selecione a opção **Rotate by Degrees**. A porta AND de duas entradas pode ser encontrada no **Symbol Tool** como and2.

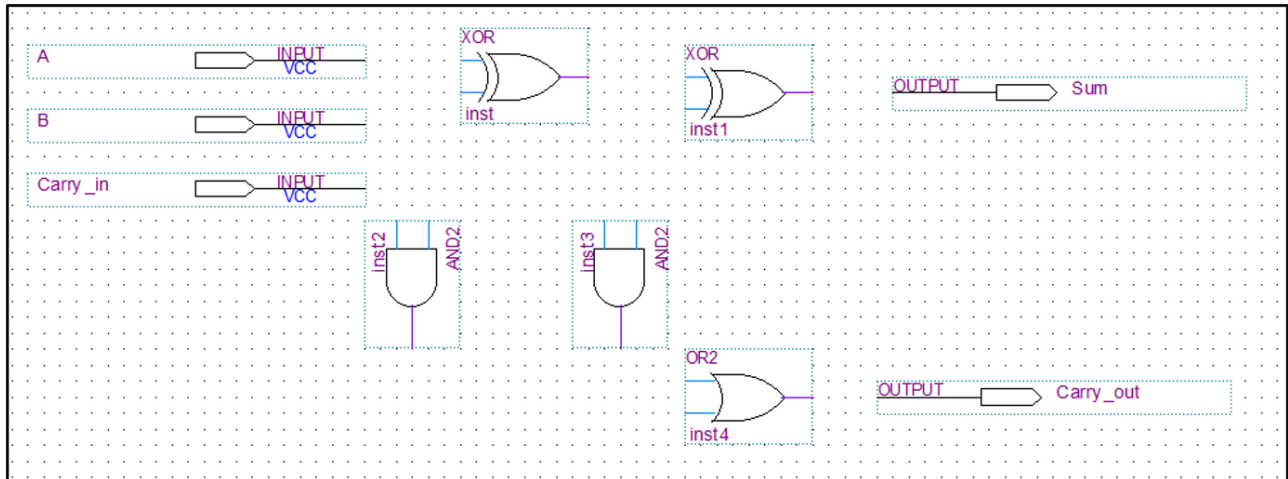


Figura 10: Esquemático do somador completo com todas as portas inseridas, mas sem as ligações.

Para finalizar o circuito somador completo, resta fazer as ligações entre os terminais conforme ilustrado na Figura 4. Para isso, aproxime o cursor do mouse à extremidade de um dos terminais e observe que o símbolo do mouse será substituído por uma “mira”. Quando isso acontecer, clique, segure e arraste o mouse até o outro ponto de conexão. Alternativamente, clique **Orthogonal node toll** (↵) para inserir linhas de conexão. Após realizar as ligações, seu circuito será semelhante à Figura 11. Salve o esquemático com o nome `somador_completo.bdf`, onde `bdf` é a extensão do arquivo (adicionada automaticamente). Não feche este esquemático.



Certifique-se de salvar o arquivo do somador completo com o nome `somador_completo.bdf`. Isso é importante para os próximos passos.

8 Criando um bloco lógico

Para reaproveitar um circuito em diferentes esquemáticos, é necessário criar um novo bloco lógico. Este processo inclui a criação do símbolo lógico para o circuito que deseja-se reutilizar. Como exemplo, na construção de um somador *carry ripple* de 4 bits, o somador completo precisa ser replicado quatro vezes e ligado em série. A Figura 12 mostra essa configuração, com as entradas X3, X2, X1, e X0 representando uma parcela da soma, e Y3, Y2, Y1, e Y0 a outra parcela. As saídas S3, S2, S1, e S0 apresentam o resultado da soma, enquanto Cout é o carry de saída, que indica se o total excede o valor representável por quatro bits (“1111”). O carry de entrada é mantido em nível lógico baixo.

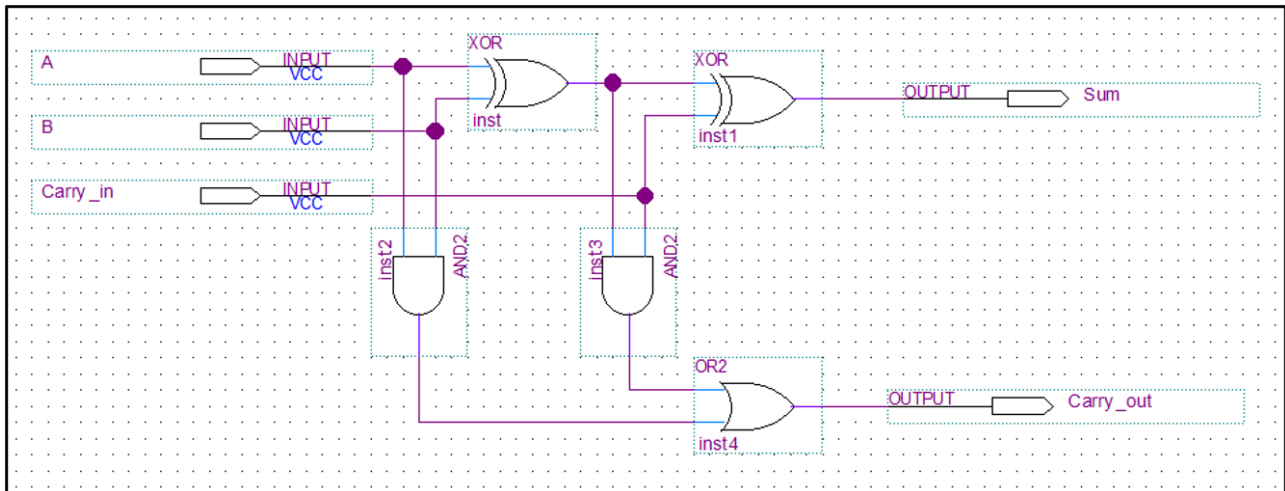


Figura 11: Esquemático do somador completo com todas as portas inseridas, e conexões completas.

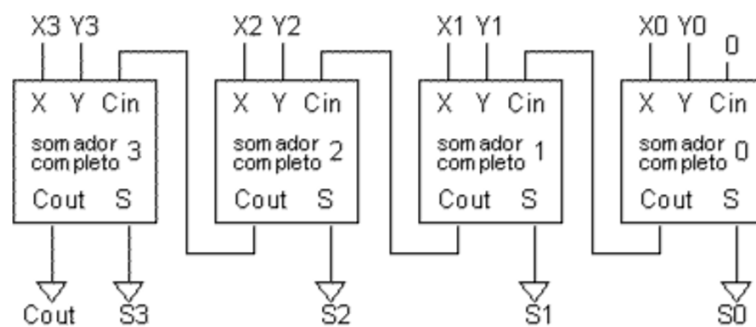


Figura 12: Ligação em série de somadores completos para formar um somador *Carry ripple* de quatro bits.

Embora seja possível desenhar o circuito inteiro utilizando apenas portas lógicas, essa abordagem tornaria a visualização e compreensão do design muito complexas. Por isso, o Quartus oferece a opção de criar blocos lógicos, que simplificam o esquemático e podem ser reutilizados em diferentes projetos, facilitando tanto o design quanto a análise do circuito.

Para a criação de um **bloco somador completo**, verifique se o esquemático do circuito está aberto em primeiro plano no Quartus. Em seguida, siga os passos abaixo:

1. Clique em File → Create/Update → Create Symbol Files for Current File.
2. Salve o arquivo na pasta do projeto com o nome "somador_completo.bsf" (sem as aspas).

A extensão .bsf indica que o arquivo é um *bloco lógico* (*block symbol file*) que pode ser utilizado em outros esquemáticos.



Tenha certeza de salvar o esquemático do somador completo com o nome "somador_completo.bsf" (sem as aspas). Isso é importante para a sequência desse tutorial.

9 Reaproveitando o bloco criado


O **top level design** é o arquivo principal do projeto e deve conter o circuito *somador de 4 bits* ilustrado na Figura 15. Para criar um novo arquivo esquemático, siga os passos abaixo:

1. Clique em File → New e escolha Block Diagram/Schematic File.

A seguir, vamos acrescentar as entradas e saídas ao esquemático. É interessante notar que algumas entradas podem ser representadas como um *array de bits*, a exemplo das parcelas e do resultado da soma. Proceda da seguinte forma para adicionar as entradas e saídas:

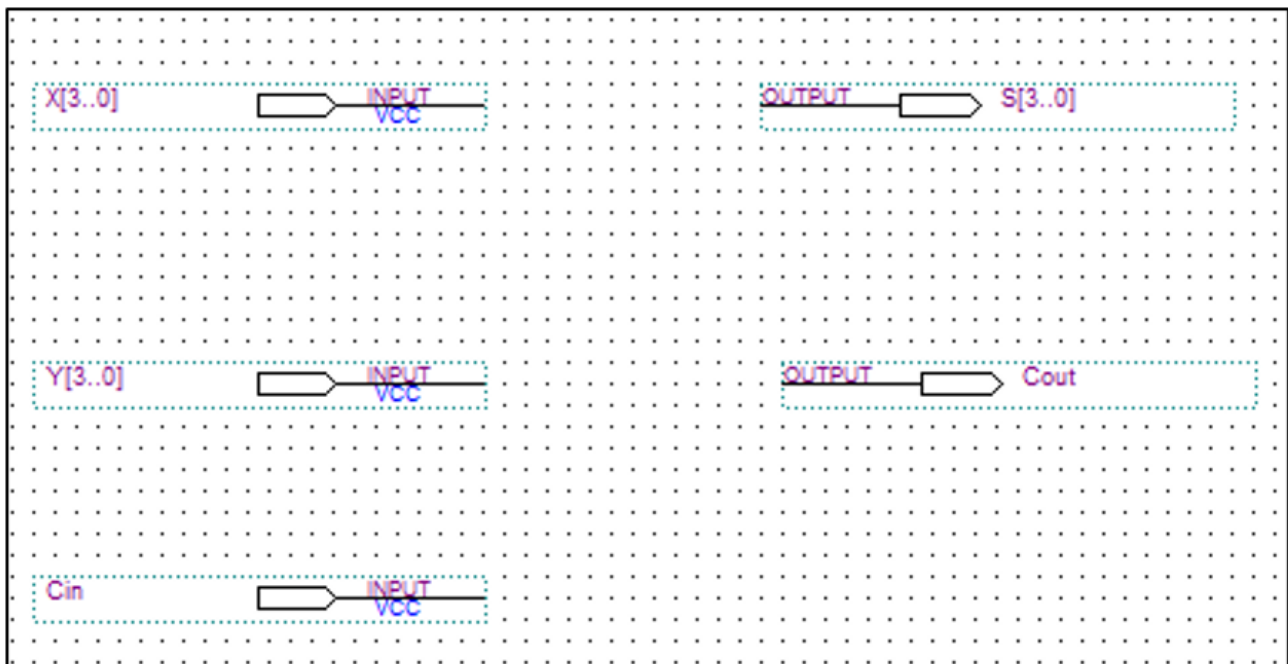
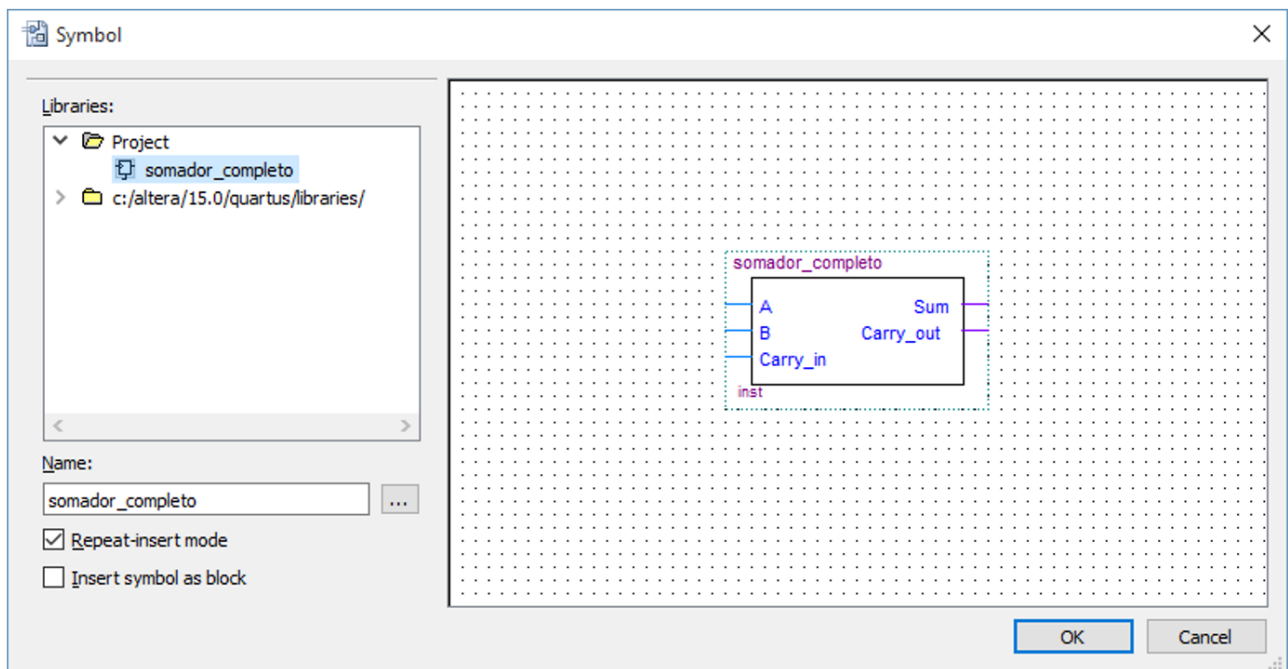
- Acrescente três entradas e duas saídas ao esquemático.
- Nomeie a entrada **X** como **X[3..0]**. Ao fazer isso, estamos indicando ao programa que esse terminal é formado por quatro pinos, onde o bit mais significativo (MSB) é **X[3]** e o bit menos significativo (LSB) é **X[0]**. Ou seja, o índice ou intervalo de índices do terminal deve ser informado dentro dos colchetes que seguem o nome do terminal.
- Repita o processo para a entrada **Y** e para a saída **S**.
- Nomeie a terceira entrada como **Cin** e a última saída como **Cout**.

O esquemático resultante deverá ser semelhante à Figura 13.

Para inserir os **blocos dos somadores completos**, selecione a ferramenta **Symbol tool** () , procedendo da mesma forma empregada na elaboração do esquemático do somador completo. A ação subsequente desencadeará a exibição da janela de seleção de dispositivos, conforme ilustrado na Figura 14.

Nesta etapa, observa-se a disponibilização de uma nova pasta no campo **Libraries**, denominada "Project", que incorpora todos os blocos desenvolvidos ao longo do projeto. Nota-se que o único componente disponível é o **somador_completo**. Proceda com a inserção de quatro instancias do somador completo no esquemático, conforme demonstrado na Figura 15.

Neste exemplo, optamos por designar que o bloco mais à direita somará os LSBs de **X** e **Y**, gerando o LSB da soma. As entradas **X** e **Y** são conjuntos de bits (4 bits cada um), e ao tentar desenhar uma linha partindo de uma dessas entradas, notamos a formação de uma linha

Figura 13: Entradas e saídas do somador *Carry Ripple*.Figura 14: **Symbol tool** com o somador completo disponível.

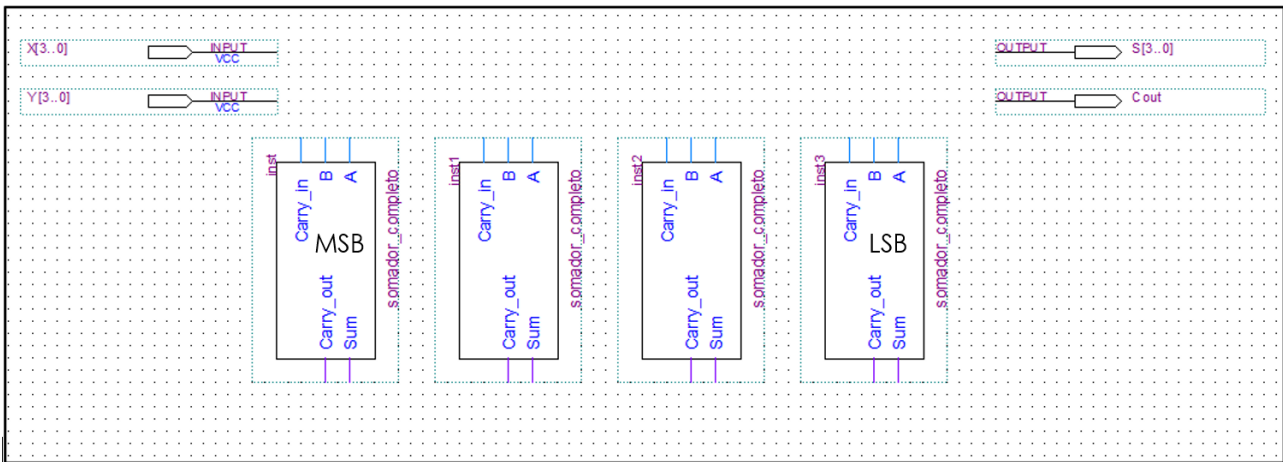


Figura 15: Posicionamento de quatro instancias do `somador_completo`.

mais espessa, indicando que se trata de um barramento de bits (para o caso em análise, um barramento de 4 bits). Portanto, é necessário derivar os bits correspondentes para conectá-los às entradas apropriadas nos somadores completos. Para isso, trace um barramento saindo do terminal de X até que este barramento abranja todos os somadores, conforme ilustrado na Figura 16.

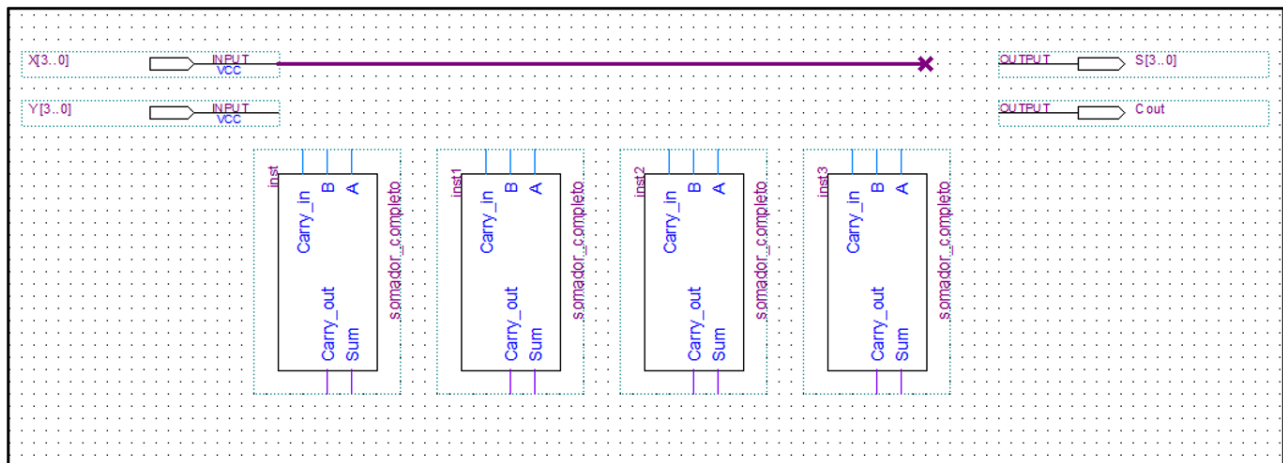


Figura 16: Traçando um barramento a partir da entrada X que possui 4 bits.

Em seguida, conecte as entradas A de cada `somador_completo` ao barramento. Para isso, posicione o mouse no terminal A e, ao visualizar o símbolo do alvo, clique, segure e arraste-o até o barramento. Após realizar a conexão nas quatro instancias, clique com o botão direito do mouse sobre o fio do LSB e selecione “properties”. No campo name da aba `general`, digite: X[0], e clique em OK. Esta ação designa que a conexão corresponde ao bit menos significativo de X. Repita o procedimento para as outras três conexões, e o esquemático apresentará semelhanças com a Figura 17. Execute o mesmo processo para a entrada Y e para a saída S, conforme ilustrado na Figura 21.

Complete o esquemático conforme indicado na Figura 19. Observe que uma nova entrada denominada Cin foi vinculada ao carry de entrada do somador que processa os LSBs. Salve o diagrama com o nome `somador_de_4_bits.bdf`, que corresponde ao nome do design de nível superior (*top-level design*) estabelecido no início do projeto. Se o esquemático for salvo com um nome diferente, ocorrerão erros durante a análise do projeto.

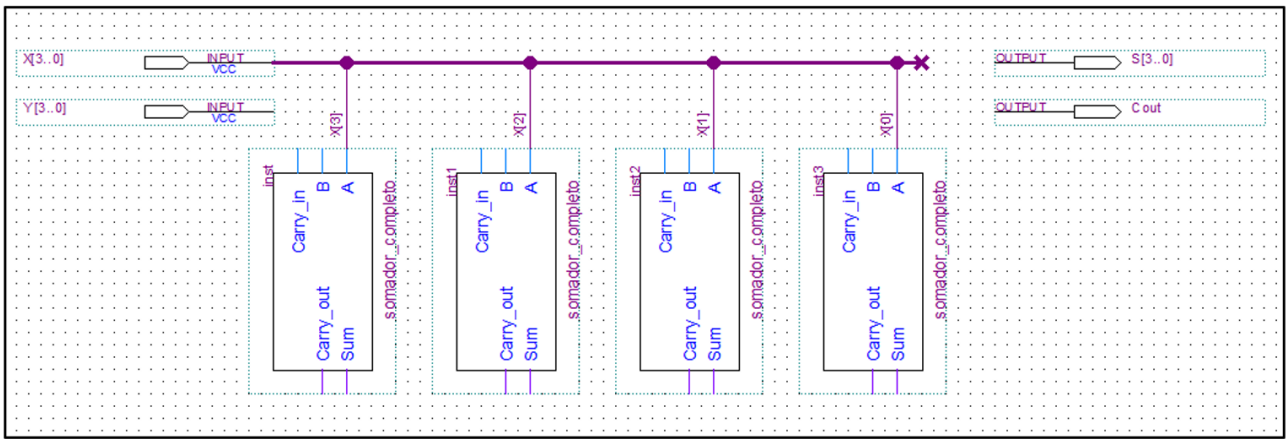


Figura 17: Ligando o barramento X às respectivas entradas das instancias do `somador_completo`.

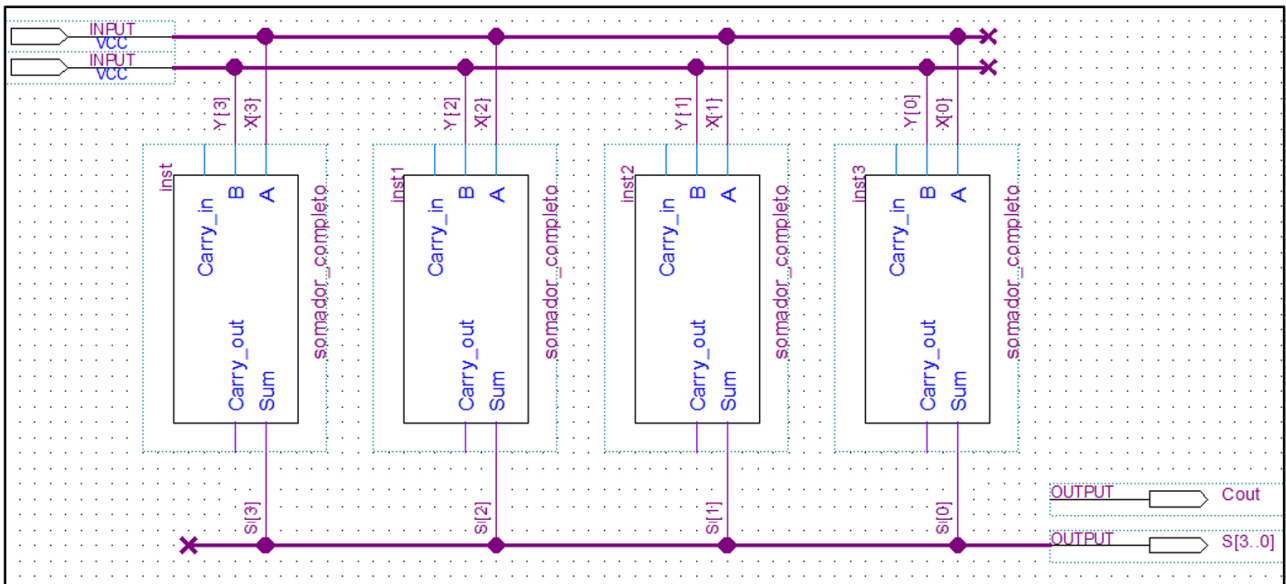


Figura 18: Ligando os barramentos X, Y e S às respectivas entradas e saídas das instancias do `somador_completo`.

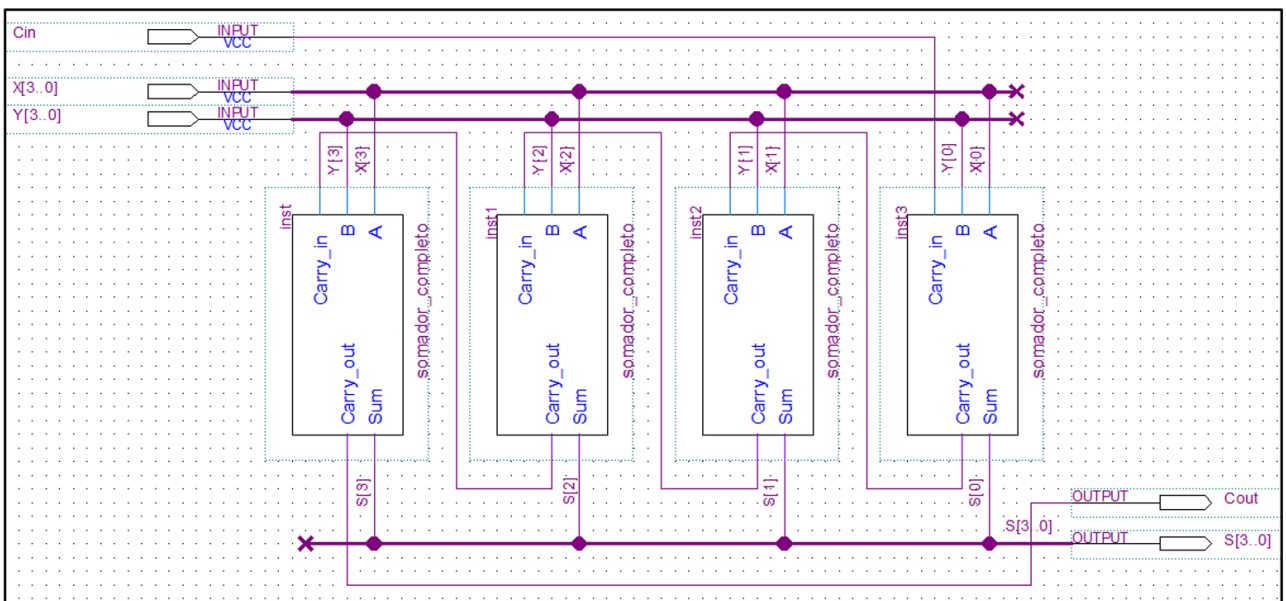


Figura 19: Esquemático do `somador_de_4_bits` finalizado.



Salve o diagrama com o nome `somador_de_4_bits.bdf`, que corresponde ao nome do design de nível superior (*top-level design*) estabelecido no início do projeto. Se o esquemático for salvo com um nome diferente, ocorrerão erros durante a análise do projeto.

10 Análise e Síntese

A compilação de um projeto no Quartus ou em qualquer outra IDE para PLDs envolve vários passos. No caso do Quartus a primeira etapa é chamada de *Analysis and Synthesis* (Análise e Síntese). As diversas etapas podem ser observadas no quadro **Tasks** que geralmente fica no canto esquerdo da tela, conforme a Figura 20. Para o intuito deste tutorial exploraremos apenas essa primeira etapa.

	Task	Time
✓	Compile Design	00:00:20
✓	> Analysis & Synthesis	00:00:11
✓	> Fitter (Place & Route)	00:00:04
✓	> Assembler (Generate programming files)	00:00:02
✓	> Timing Analysis	00:00:02
✓	> EDA Netlist Writer	00:00:01
	Edit Settings	
	Program Device (Open Programmer)	

Figura 20: Quadro **Tasks** do quartus mostrando as etapas de compilação.


A análise e síntese no Quartus, representam etapas fundamentais no processo de desenvolvimento de circuitos digitais, especialmente para FPGAs (*Field Programmable Gate Arrays*) e CPLDs (*Complex Programmable Logic Devices*).

Durante a etapa de **análise**, o Quartus examina o código ou o esquemático submetido para o projeto, que pode estar em VHDL, Verilog, ou como um esquemático gráfico. Esta fase inclui a verificação da sintaxe e da semântica do código ou do esquemático, garantindo que eles estejam isentos de erros e sejam interpretáveis pela ferramenta. A análise assegura a correção do design em relação às regras da linguagem de descrição de hardware utilizada e às especificações do projeto.

Após a análise, ocorre a **síntese**, que é o processo de conversão do código ou esquemático de alto nível em uma representação de baixo nível que mapeia diretamente para os recursos físicos do dispositivo alvo, como lógica programável, células lógicas, blocos de memória, entre outros. Durante a síntese, a ferramenta realiza otimizações no design para melhorar o desempenho e a utilização dos recursos, respeitando as restrições definidas pelo usuário (tais como frequência de operação, consumo de energia, tamanho, etc.).

O resultado da síntese é um *netlist*, que pode ser traduzido como “lista de interconexões”, descrevendo como os blocos lógicos são interconectados dentro do FPGA ou CPLD. Esse *netlist* é então utilizado nas etapas subsequentes do processo de design, como a colocação e o roteamento (*place and route* ou *fitter*), onde a ferramenta determina a localização física dos componentes no dispositivo e como eles serão interconectados fisicamente. O *netlist* também pode ser utilizado para a realização de simulações.

Com o esquemático finalizado, a próxima etapa é proceder com a análise e síntese do circuito, permitindo que o programa execute simulações funcionais. O processo de análise tem por objetivo verificar a presença de erros ou incoerências no projeto. Para iniciar a análise e a

síntese, pressione **Ctrl + K**, ou clique no botão correspondente na parte superior da janela do Quartus () ou dê um duplo clique sobre **Analysis & Synthesis** no quadro **Tasks**. Aguarde alguns minutos para que o processo seja concluído. Caso sejam identificados erros, efetue as correções necessárias e repita o procedimento.

11 Simulação funcional

A **simulação funcional** é um tipo de simulação utilizada no design de circuitos eletrônicos, especialmente em circuitos digitais e sistemas embarcados, para verificar o comportamento lógico do design. Ela é realizada antes ou durante as fases iniciais do processo de desenvolvimento para garantir que o circuito ou sistema realiza as funções desejadas conforme especificado, sem considerar os aspectos físicos como tempos de atraso, capacidade de carga, ou outras características elétricas do hardware.

No Quartus, clique em **File** → **New** e escolha a opção **University Program VWF**. Uma nova janela se abrirá, conforme a Figura 21. Observe que o arquivo está vazio e sem possibilidade de configuração; para adicionar os terminais que serão simulados (entradas e saídas do circuito somador de 4 bits), clique em **Edit** → **Insert** → **Insert Node or Bus...** e uma nova janela se abrirá, conforme ilustrado na Figura 22. Nesta nova janela, clique no botão **Node Finder**, que abrirá a janela da Figura 23

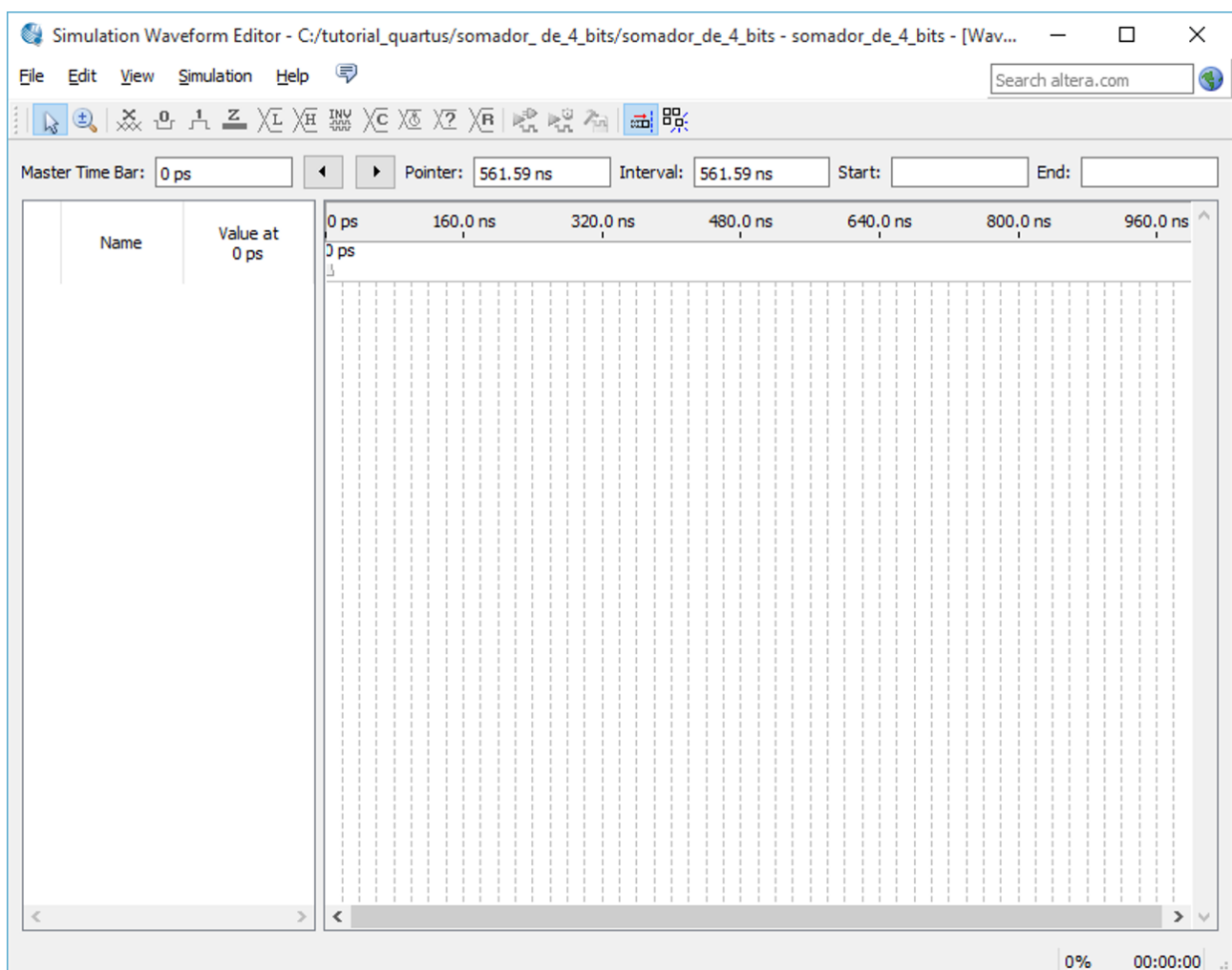


Figura 21: Ambiente de simulação do University Program VWF.

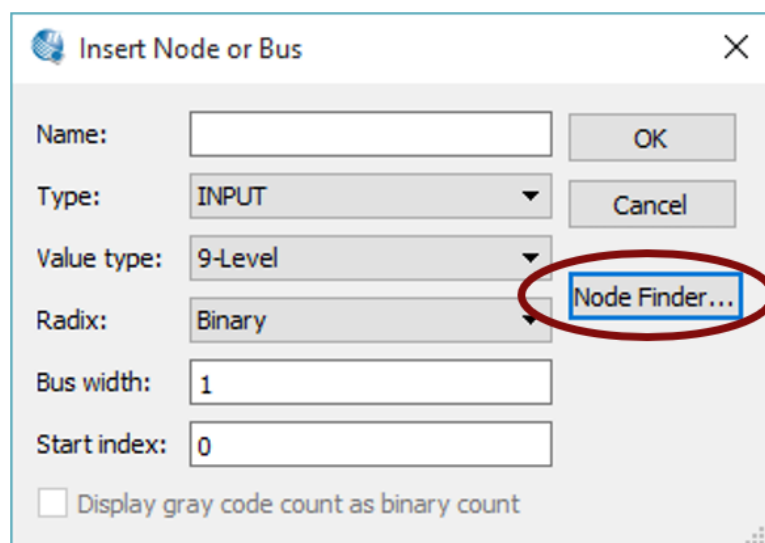


Figura 22: Janela Insert node or bus.

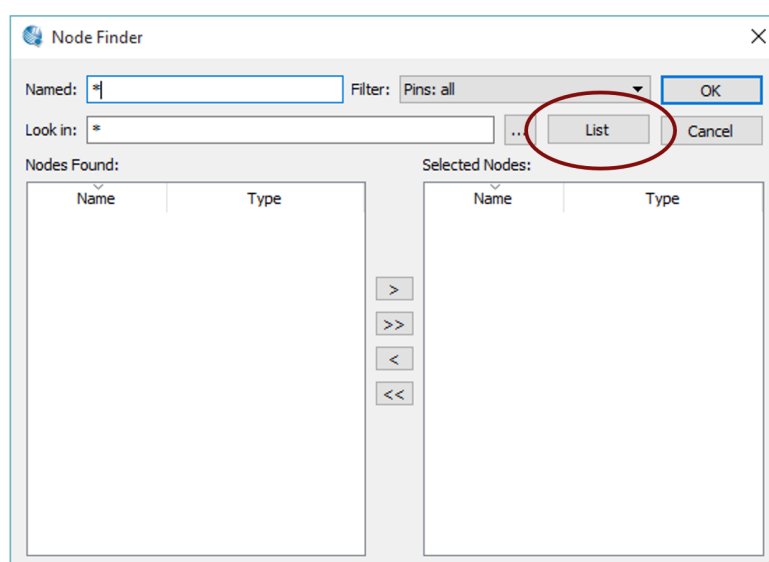


Figura 23: Janela Node finder.

Clique em **List** e os terminais de entrada e saída do projeto serão exibidos no campo **Nodes Found**. Clique no botão com o símbolo “»” para selecionar todos os terminais criados no projeto. Clique em **OK** e, novamente, em **OK**. O ambiente de simulação ficará similar ao apresentado na Figura 24.

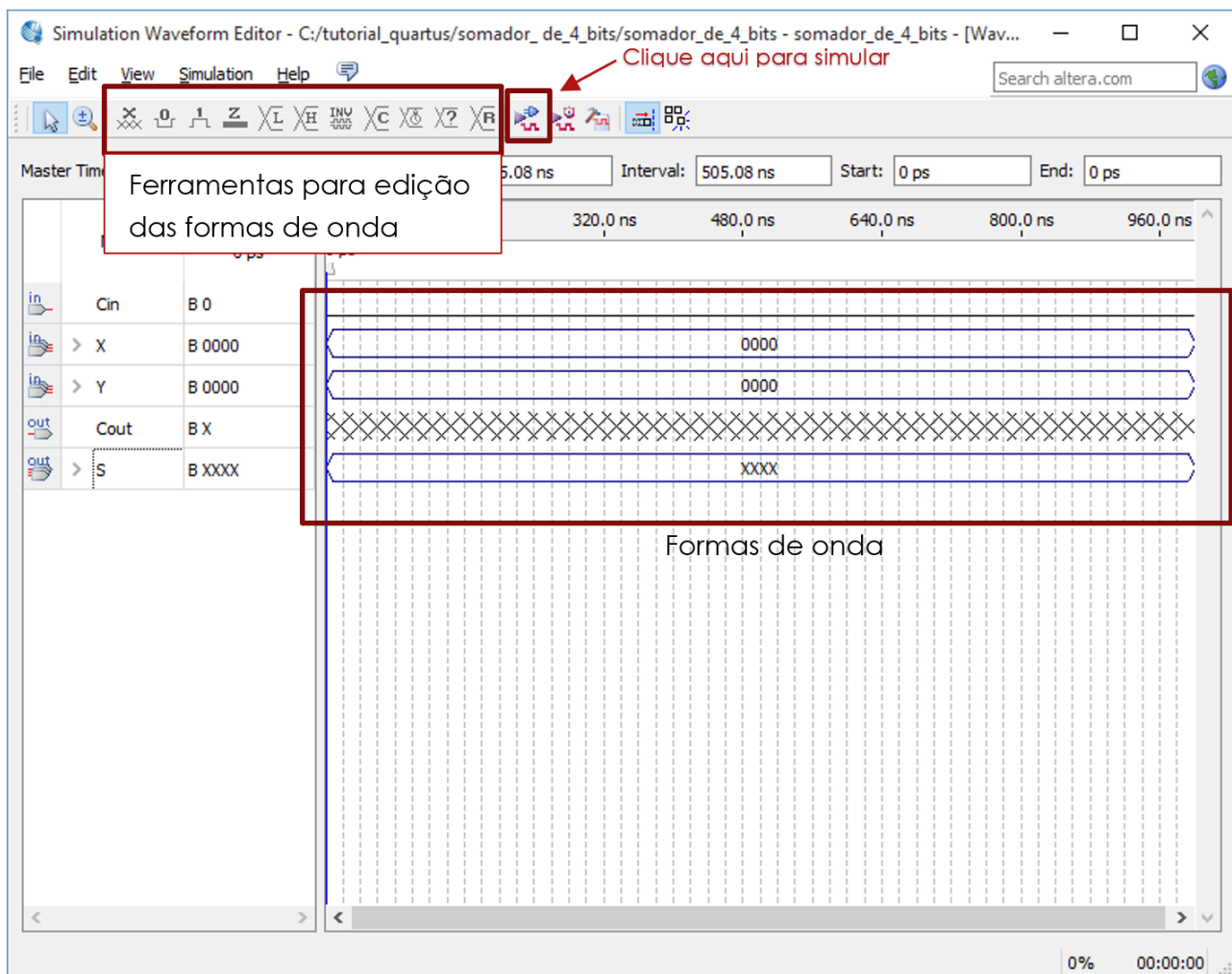


Figura 24: Ambiente de simulação do University Program VWF com as entradas e saídas inseridas e prontas para edição.

No canto direito do ambiente de simulação, conforme visto na Figura 24, encontram-se as formas de onda que podem ser editadas. É importante enfatizar que somente as formas de onda de terminais classificados como de entrada e bidirecionais estão sujeitas à edição. As ferramentas para edição das formas de onda digitais localizam-se no canto superior esquerdo.

A manipulação das formas de onda de entrada é intuitiva. Para editar, selecione com o mouse (clcando e arrastando) a faixa de tempo desejada em um dos terminais de entrada e empregue uma das ferramentas disponíveis para modificar os valores lógicos dos sinais. Por exemplo, caso seja necessário inserir o valor binário “1100” em X, no intervalo de tempo de 0 a 300 ns, deve-se selecionar o intervalo de tempo correspondente para a entrada X. Após a seleção, clica-se no botão com o símbolo de interrogação para forçar a entrada de um valor arbitrário. Consequentemente, uma janela será aberta permitindo a inserção do valor desejado.

Proceda com a edição das formas de onda de X, Y, e Cin de maneira a testar exaustivamente o circuito. Confira se a operação de soma está sendo executada corretamente. Este momento também é oportuno para explorar as funcionalidades de cada ferramenta de edição. Para iniciar a simulação, clique no botão indicado na Figura 24 (Run Functional Simulation), é possível que seja solicitado que o arquivo de simulação seja salvo.

A Figura 28 ilustra o resultado de uma simulação para o circuito somador de 4 bits. Importante ressaltar que a simulação funcional considera apenas o uso de componentes ideais, desconsiderando, portanto, quaisquer tempos de atraso. Uma simulação que incorpora a análise de tempos de atraso é denominada simulação temporal e será abordada em outro tutorial.

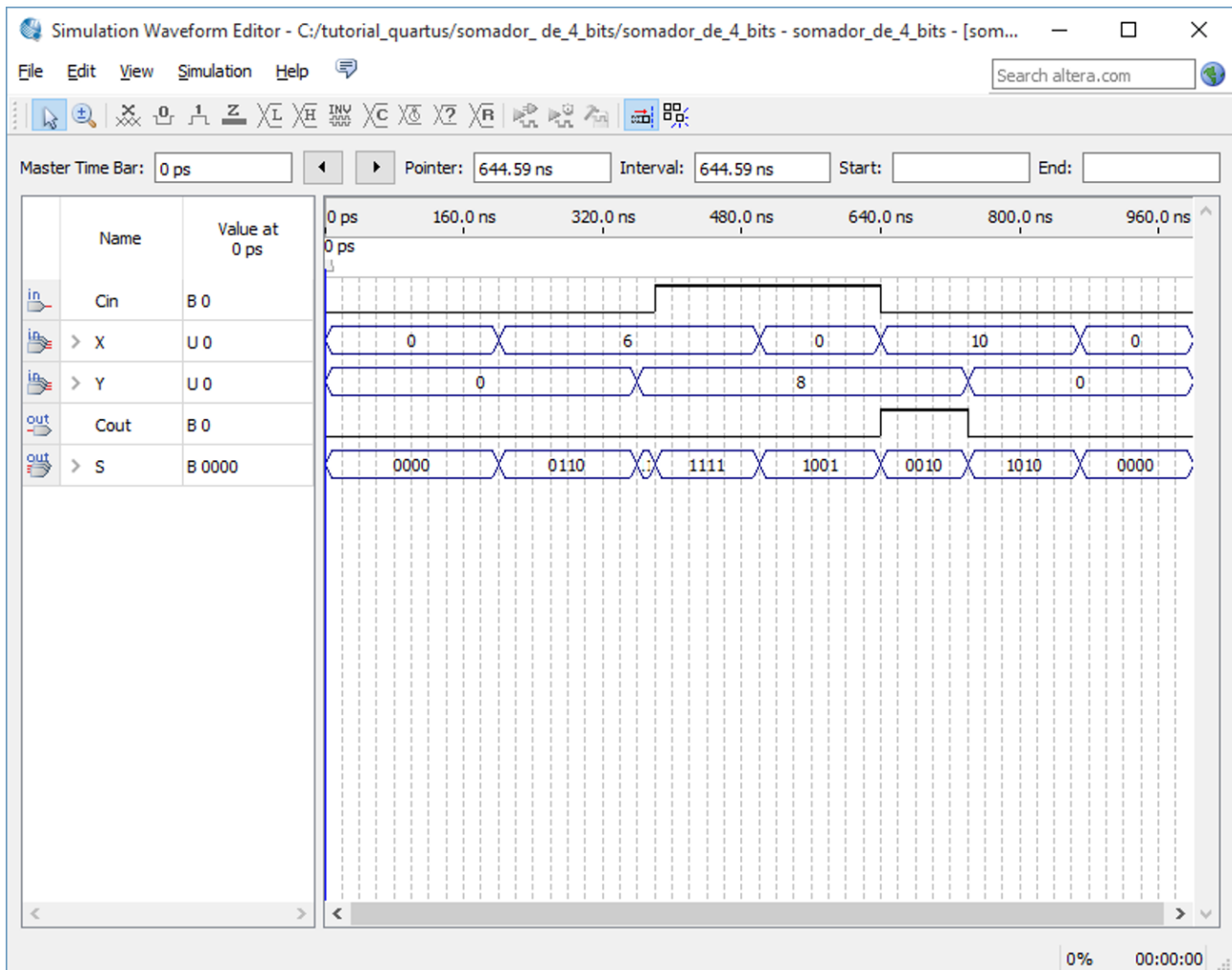


Figura 25: Exemplo de resultado de simulação para o somador de 4 bits.

Referências

- [1] Intel Corporation, *Intel® Quartus® Prime Software User Guides*, 2023, acessado em 15 de março de 2024. [Online]. Available: <https://www.intel.com/content/www/us/en/support/programmable/support-resources/design-software/user-guides.html>
- [2] Luis Meira, “Aula 5 Parte 10 Circuito Somador 1 4 e 8 bits,” <https://www.youtube.com/watch?v=eiFXNVBd8LQ>, 2020, acessado em 20 de março de 2024.
- [3] R. J. Tocci, N. S. Widmer, and G. L. Moss, *Sistemas Digitais: Princípios e Aplicações*, 11th ed. São Paulo: Editora Pearson, 2007.
- [4] DesCOMPLICA, Oliba!, “Circuito Somador Completo *FullAdder* em Detalhes - Soma de Inteiros Positivos com Vários Bits,” https://www.youtube.com/watch?v=486Rhht8_8U, 2021, acessado em 20 de março de 2024.