

µProcessador 1 Introdução ao Quartus II

O Quartus II é uma ferramenta para síntese de circuitos programáveis: um circuito digital qualquer pode ser projetado e testado, para posteriormente ser gravado em uma FPGA (*Field Programmable Gate Array*: é um CI programável em que podemos gravar qualquer circuito digital que desejarmos). Os módulos são especificados usualmente nas linguagens VHDL ou Verilog, mas sugiro usar o desenho esquemático com portas e flip-flops.



Os arquivos fonte exigem compilação e análise para adaptar e otimizar o circuito às estruturas presentes na FPGA a ser utilizada e para assegurar a consistência do esquemático (que pode queimar o circuito com curtos, por exemplo). A simulação é *não-interativa*, especificando entradas e observando a resposta do circuito nas saídas.

A versão utilizada em sala é a 13.0 Web Edition SP1¹ para Windows. Quem quiser usar ou aprender VHDL, esteja à vontade².



Faça um circuito somador completo de 1 bit a partir de portas lógicas. Crie o arquivo de teste e verifique se o seu circuito funciona. Utilize as informações a seguir.

Criação de Projeto/Arquivo

- Para criar um projeto, utilize o Wizard. *Não repita o nome do projeto* nos arquivos e componentes.
- *Atenção*: deve-se criar uma pasta *exclusiva* para o projeto (se colocar dois projetos na mesma pasta, há conflitos): basta colocar uma pasta inexistente no Wizard, ele cria. Coloque todos os arquivos fonte lá dentro.
- Para criar um novo arquivo (um esquemático): *File -> New: Block Diagram/Schematic File*
- Para inserir um componente/elemento, basta dar duplo-clique no esquemático ou clicar no botão  das ferramentas e clicar no esquemático. Um diálogo de escolha surge:
 - digite o nome INPUT ou OUTPUT ou escolha em “altera/primitives/pin” para especificar e colocar as entradas e saídas do seu circuito;
 - *nomeie* adequadamente os pinos (acredite em mim);
 - digite AND2, OR3, XOR, etc. (ou vá em “altera/primitives/logic”) para portas;
 - digite VCC ou GND (“altera/primitives/other”) para colocar níveis fixos em 0 ou 1.
- Para conectar tudo, use o botão  (condutores, são riscos “finos”).
- Salve com um nome diferente do projeto (assegure-se de estar gravando na pasta certa).

Note que *inputs* e *outputs* são os sinais usados para simulação e para construção de blocos. O somador completo deverá ter pinos de entrada para bits A, B e carry_in, e saídas de S e carry_out.

Compilação

1. Selecione a aba **Files** do *Project Navigator*. É a janela que deve estar no canto superior esquerdo; se estiver fechada, abra com *View -> Utility Windows -> Project Navigator*
2. Clique o botão direito no arquivo a simular -> *Set As Top-Level Entity (IMPORTANT!)*
3. Botão do Play (*Processing -> Start Compilation*, compila a entidade Top-Level); demora
4. *Conserte os erros* (errors, em vermelho na janela de mensagens; sempre há diversos warnings por não usarmos uma FPGA de verdade)



1 Dá pra baixar free de <https://www.altera.com/download/software/quartus-ii-we> com o registro de usuário e senha mas tem 4,7GB (demora); o professor tem cópias em DVD. Instalado ele ocupa 11GB. Linux? Tem lá, mas fica por sua conta e risco.

Como alternativa mais leve, a versão 9.1 SP2 tem 2GB (<https://www.altera.com/download/software/quartus-ii-we/9.1sp2>), fica com uns 4GB instalado. Os arquivos das versões não são totalmente compatíveis, entretanto.

2 VHDL é o assunto da disciplina de Lógica Reconfigurável. É uma linguagem de “programação de hardware.”


Simulação com Formas de Onda no Quartus II 13.0

Devemos criar um arquivo de simulação .VWF (Vector Waveform File), especificando pinos com valores de entrada e os sinais que desejamos observar.

- Crie o arquivo de simulação: *File -> New: (Verification/Debugging Files) University Program VWF*
- Uma janela de simulação irá se abrir
- Salve o arquivo com qualquer nome
- Para escolher os sinais de interesse, tanto de entrada quanto de saída:
 - Clicar com o botão direito no painel da esquerda da janela, e *Insert -> Node or Bus...*
 - Clicar *Node Finder...* na caixa de diálogo
 - Clicar em *List* (o sinal não está lá? *Recompile!*)
 - Escolher os sinais e clicar ">", e Ok, e Ok de novo
- ESSENCIAL: ajustar os tempos em *Edit -> Grid Size* e *Edit -> End Time* (os defaults são ruins)
- Há dois tipos de simulação:
 - *Functional* (botão  ao lado do play na janela principal, que não compila tudo)
 - *Timing analysis* (botão 


Simulação com Formas de Onda no Quartus II 9.1

Devemos criar um arquivo de simulação .VWF (Vector Waveform File), especificando pinos com valores de entrada e os sinais que desejamos observar.

- Crie o arquivo de simulação: *File -> New: (Verification/Debugging Files) Vector Waveform File*
- Uma nova aba, para simulação, irá se abrir
- Salve o arquivo; é **obrigatório** nomeá-lo com o NOME DO PROJETO.vwf
- Para escolher os sinais de interesse, tanto de entrada quanto de saída:
 - Clicar com o botão direito no painel da esquerda da aba, e *Insert -> Node or Bus...*
 - Clicar *Node Finder...* na caixa de diálogo
 - Mudar *Filter* para "Pins: all" e clicar em *List* (o sinal não está lá? *Recompile!*)
 - Escolher os sinais e clicar ">", e Ok, e Ok de novo
- ESSENCIAL: ajustar os tempos em *Edit -> Grid Size* e *Edit -> End Time* (os defaults são muito pequenos e causam "erros" devido ao tempo de propagação das portas)
- Há dois tipos de simulação:
 - *Functional*: apenas teórica, não considera o atraso das portas; mais rápida
 - *Timing analysis*: considera o mapeamento do circuito e o atraso das portas
- Para simular com *Timing analysis*, clique em  (ou *Processing -> Start Simulation*)
- Para fazer a *Functional analysis* é mais complicado:
 - Abrir *Processing=>Simulator tool*
 - Nesta janela, mudar "Simulation mode" para *Functional*
 - Aqui pode-se usar arquivos com nome diferente do projeto
 - Pressionar "Generate Functional Simulation Network"; *fazer isso de novo a qualquer alteração no circuito*
 - (Se os pinos de I/O forem alterados, use CTRL-K para recompilar)
 - Pressionar o botão de "Start"

Comparação das Simulações

Faça uma simulação funcional como a abaixo:

- | | |
|---|--|
| ► | Ajuste o Grid Size para 20ns e o End Time para 200ns. Defina a entrada A como um clock (clique nela e depois no botão  e escolha o período em 20ns), o <i>carry_in</i> como um clock com 40ns de período e escolha os níveis de B manualmente (basta arrastar o mouse clicado sobre a forma de onda e depois clicar os botões "0" ou "1"). Salve, simule e verifique o funcionamento. |
|---|--|

Atalho: CTRL-ALT-W é zoom para todo o tempo de simulação definido (só CTRL-W no 9.1)

Agora faça a simulação temporal com atrasos.

	Utilizando as mesmas formas de onda de entrada, realize a simulação de análise temporal. Verifique os sinais espúrios nas transições de níveis, usando zoom: são devidos aos atrasos das portas.
►	Agora ajuste o Grid Size para 1µs e o End Time para 20µs. Defina A como um clock de período 3µs, o <i>carry_in</i> como um clock com metade da frequência de A e escolha os níveis de B manualmente. <i>Salve</i> e simule. Perceba com zoom que os efeitos do atraso estão lá, mas muito menos perceptíveis.

Criação e Utilização de Blocos

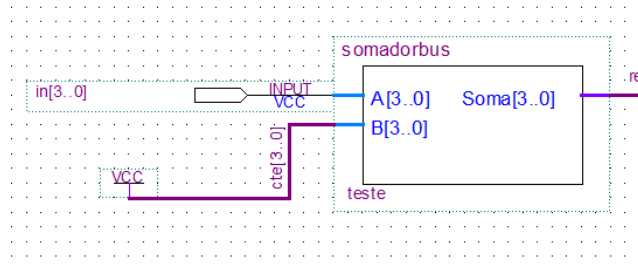
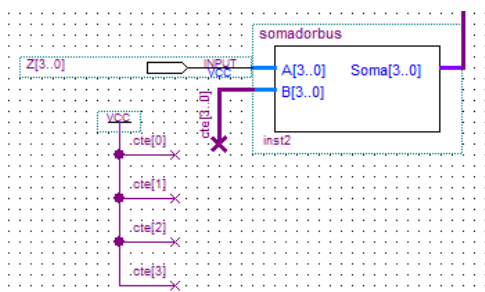
- Para criar uma entidade (um bloco): clique na aba do circuito para vê-lo e execute *File -> Create/Update -> Create Symbol Files for Current File*. Isso deve ser repetido a cada alteração interna do bloco. Assegure-se de que o arquivo foi compilado.
- Para usar, crie um novo esquemático e insira o símbolo normalmente, como uma porta lógica qualquer; ele está em uma pasta “project” no início da árvore de componentes.
- Se a interface do bloco (pinos de entrada e saída) forem alterados, ele precisa ser reinserido (ou seja: delete o bloco antigo onde estava sendo usado e coloque-o de novo).

	Crie um bloco para o somador completo (compile, clique na aba, <i>create symbol files</i>). Crie um novo esquemático e usando quatro somadores, faça um somador de 4 bits. Grave, sete ele para Top-Level, compile.
►	Crie uma cópia do arquivo de simulação original (por bom senso) e crie um novo arquivo de simulação com o nome do projeto (o nome é fixo, como já dito). Selecione os quatro bits do primeiro número (digamos, A3 a A0) e os agrupe com o botão direito. Defina uma contagem com XC ; para o outro número (B), use randômico com XR .

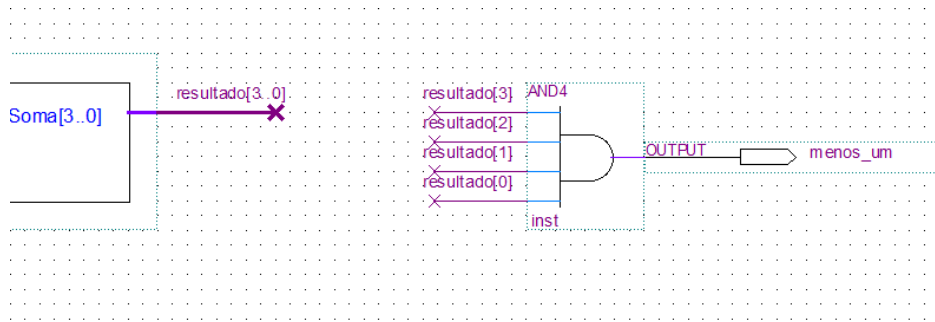
Atalhos: CTRL-ALT-SPACE: *toggle fullscreen*; CTRL-SPACE: *zoom in*; SHIFT-CTRL-SPACE: *zoom out*

Barramentos

- Para criar um barramento, basta inserir um pino de *input* ou *output* e dar um nome como um vetor (ex.: o nome *op[3..0]* cria um *bus* de 4 bits chamado *op*). Ligações a eles também são *buses* (fios mais espessos no esquemático). Na figura a seguir à esquerda: *bus* *cte[3..0]* ligado ao *bus* B, sinais *cte[3]*, *cte[2]*, *cte[1]* e *cte[0]* do *bus* *cte[3..0]* ligados a VCC.
- Podemos curto-circuitar um barramento inteiro a VCC ou GND, se desejarmos, como pode se ver abaixo à direita.

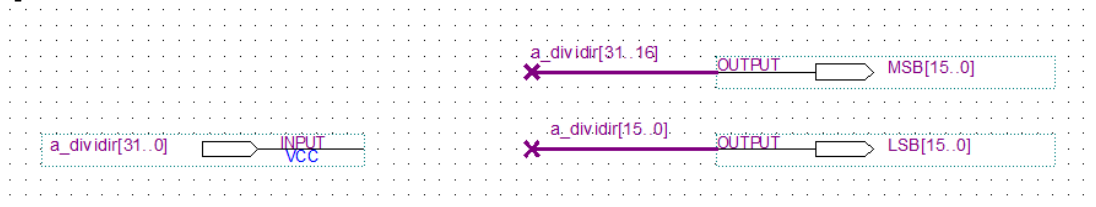


- Para ligar um sinal individual no barramento:
 - Criar o condutor individual (um fio) e o ligar ao ponto do sinal individual
 - Alterar o nome dos condutores para especificar o sinal individual (digamos: *a[2]* ou *data[0]*), acessando as propriedades do fio com o botão direito do mouse
 - Não ligar “fisicamente” o fio ao barramento (pode gerar curto-circuitos)
 - Note a seguir a instanciação (“renomeação”) de *Soma* para o *bus resultado*.



- Faça um circuito somador de 4 bits usando blocos de somadores completos, com barramentos nos pinos e simulação de *todas as possibilidades de entrada*. Aterre *carry_in* (o nome do símbolo é “GND”) e ignore *carry_out*.
Agrupe os conjuntos de bits de entrada selecionando os bits individuais e usando o botão direito do mouse. Mostre os dados em decimal sinalizado.

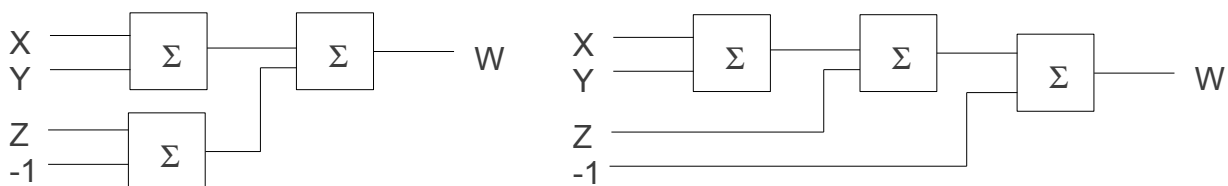
Em práticas posteriores, será necessário dividir ou agrupar barramentos e “sub-barramentos”. Isto é feito como na figura a seguir, onde se divide $a_dividir[31..0]$ nos barramentos $MSB[15..0]$ e $LSB[15..0]$.



Circuito a apresentar

- **Apresentar até a próxima aula vem** (não aceito este circuito por e-mail): circuito que utiliza blocos somadores de 4 bits como acima para calcular $W=X+Y+Z-1$. A constante -1 deve estar implícita no circuito (sinais VCC e GND), não sendo alterada pela simulação. As variáveis são em 4 bits complemento de dois.
Crie um projeto novo em uma pasta nova e **copie para lá** os arquivos que você vai reutilizar.

Circuito (escolher uma das duas versões possíveis):




Testes a Apresentar

- Entradas: apenas os três grupos X, Y e Z (ou seja, X é agrupamento de x3x2x1x0)
- Saída: apenas o grupo W de 4 bits
- Visualização em *decimal sinalizado (!)* – favor conferir
- Os blocos somadores de 4 bits não devem usar *carry in* nem *carry out*³
- Ordem de visualização: X, Y, Z, W, de cima para baixo (cuidado com LSBs e MSBs)
- Para testes, caso o circuito esteja suspeito, melhor usar binário; pode colocar outros sinais

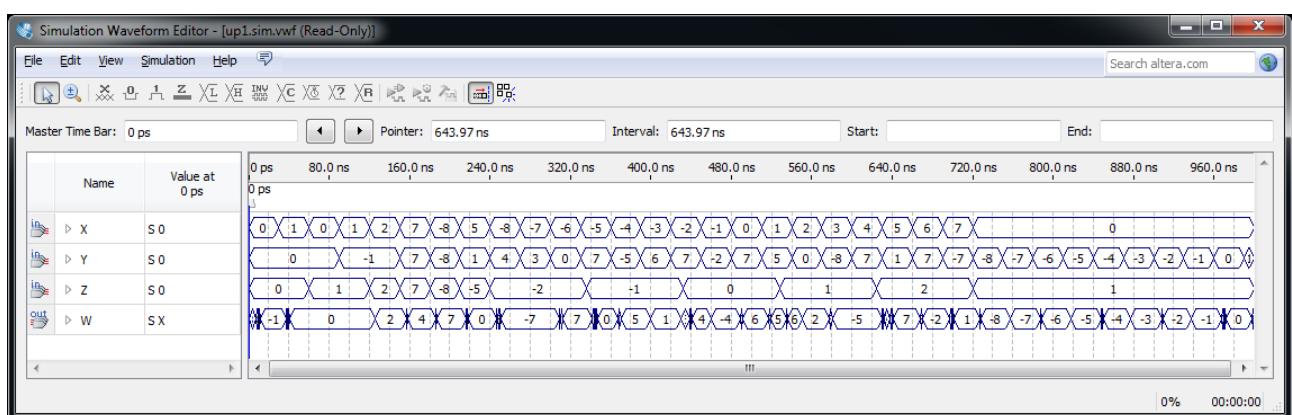
³ Perceba que se o circuito propagar o valor de *carry* (que deveria servir para positivos exclusivamente), isso faz com que ele não funcione direito para negativos! Tipicamente, -1-1 resulta em +1 ao invés de -2... Faça um somador que entrem dois número de 4 bits e sai apenas um número de 4 bits.

de debug, mas retire-os para apresentar

- De novo: a constante -1 não pode ser colocada como um sinal simulado, tem que ser colocada com Vcc e Gnd no circuito (os nomes dos símbolos são “VCC” e “GND”)
- Para inserir os valores constantes da tabela, usar o botão 
- Valores a testar obrigatoriamente seguindo a tabela abaixo:

	tempo => ...											
X	0	1	0	1	2	7	-8	5	contagem -8 a +7	0	16x randômico	-2
Y	0	0	0	-1	-1	7	-8	1	16x randômico	contagem -8 a +7	2	16x randômico
Z	0	0	1	1	2	7	-8	-5	contagem -2 a +2	1	16x randômico	16x randômico
W

- Portanto eu quero ver uma janela como a abaixo (repare no trecho que escolhi):



Pegadinhas (“Deu um pau esquisito, professor...”)

Repetindo alguns itens espalhados pelo documento:

- Para compilar um módulo, ele deve ser a TOP-LEVEL ENTITY (ou ser incluído no *top level*): verifique isso
- O arquivo de simulação tem que ter o *mesmo nome do projeto*
- “O circuito demora pra responder à entrada”: você está usando tempos muito pequenos, próximos de ns; aumente isso pra dezenas de ns
- Se na criação da simulação o sinal não aparece na lista de pinos, basta recompilar
- Se um bloco está dando erro, recompile; se mudou os pinos dele, delete os lugares onde ele é usado e insira novamente
- Não pode ter dois projetos na mesma pasta
- Procure colocar todos os arquivos de esquemático e simulação em uma única pasta
- Se a janelinha com os arquivos e módulos sumiu: View -> Utility Windows -> Project Navigator
- “Tem warnings de pinos *unassigned*”: isso não importa
- **Teclas para zoom:** Ctrl-Alt-W, Ctrl-Espaço, Shift- Ctrl-Espaço, Ctrl-Alt-Espaço

Se quiser adiantar o próximo passo: fazer uma ULA de 4 bits ou mais com 4 operações ou mais.