Universidade Federal de Minas Gerais
Departamento de Ciência da Computação

Organização de Computadores I
Lab 1: Introdução.
Quartus II e Simulação com o ModelSim

Professor: Antonio Otavio Fernandes.
Monitor: Omar Vidal Pino.

21 de Setembro de 2016

1 Quartus II

O Quartus II é um programa da ALTERA orientado ao projeto e implementação de circuitos lógicos nas suas famílias de dispositivos lógicos programáveis FPGA (baseado em memória RAM) e CPLD (baseado em memória FLASH). Ele é uma ferramenta de desenvolvimento com ambiente integrado que incorpora utilitários para entrada de projetos lógicos, compilação do projeto e sua simulação.
Além disto, através do cabo apropriado, é possível gravar um FPGA ou CPLD com o projeto lógico desenvolvido. Uma versão WEB está disponível para *download* no site (procurar por *Quartus II Web Edition*)`http://www.altera.com/`. Para uso no laboratório, a edição 13.1 é a sugerida, porque é a última com suporte aos dispositivos da família Cyclone II, utilizado no módulo DE2 disponível no laboratório. Após sua instalação, é necessária a obtenção de uma licença, também do mesmo site.
A entrada de projeto possui dois modos de operação básicos: editor gráfico (coberto nesta aula prática) e editor em modo texto (assunto das próximas aulas), este último voltado para a edição de aplicações utilizando *linguagens de descrição de hardware*, tais como AHDL, VHDL ou Verilog HDL. A compilação e a simulação são etapas subsequentes através do qual é possível verificar se há erros de lógica e corrigí-los.

1.1 Objetivos

O editor gráfico é usualmente empregado para circuitos de baixa complexidade ou para conexão entre projetos de forma rápida. Atualmente, a ênfase é no uso de linguagens de descrição de *hardware*, por apresentar uma série de vantagens: inclusão de comentários, projeto da lógica

desenvolvido pelo *software* e não pelo usuário, verificação de erros lógicos, otimização do projeto e outras. Nesta primeira prática, a ênfase é apresentar o ciclo de desenvolvimento com o Quartus II: entrada do projeto lógico, compilação, simulação e gravação.

1.1.1 Entrada do projeto lógico

O Quartus II trabalha com o conceito de projeto, que engloba os arquivos de entrada de projeto lógico, os arquivos de simulação e vários outros que ele cria quando compila o projeto. Assim, a etapa inicial, antes da entrada do projeto propriamente dito, é a entrada ou definição do arquivo que contém a lógica a ser sintetizada. Para criar um novo projeto, selecione o menu *File* $\longrightarrow$*New Project Wizard*. Clique em *Next* e, na próxima janela,selecione o diretório onde o projeto será armazenado. Por conveniência, crie um novo diretório com o nome da aula ("Aula1", por exemplo) e dê o mesmo nome ao projeto.
Nas opções seguintes, selecione *Next*, o codigo do dispositivo (ex: **EP2C35F672C6N**) da *família Cyclone II* e *Finish*, para terminar a criação do projeto. Note que o arquivo lógico ainda não foi especificado.
Para criar um projeto lógico baseado em diagrama lógico, selecione *File*$\longrightarrow$ *New e Block Diagram/Schematic File* na janela que for aberta. Para adicionar um componente lógico ao editor gráfico, dê um clique duplo sobre a área de edição e, na caixa de diálogo que abrirá em seguida, selecione o símbolo navegando nas pastas em *Libraries*, ou digite seu nome no campo *Name* (por exemplo, *or2* para uma porta lógica or com duas entradas, *input* para uma entrada, *output* para saída, etc.).
Qualquer circuito lógico criado pelo usuário através do editor gráfico pode ser definido como um componente que, por sua vez, poderá ser empregado no projeto de outro circuito. Para tal, após ter salvado o arquivo correspondente ao circuito, selecionar *File*$\longrightarrow$*Create / Update Create Symbol Files for Current File*.

1.1.2 Compilação do projeto lógico

Uma vez que o projeto tenha sido criado e um arquivo de entrada de projeto adicionado, a próxima etapa consiste em compilar o projeto, ou seja, traduzí-lo para um formato que possa ser carregado num **FPGA** ou CPLD.
O compilador realiza várias operações nesta tradução:

- análise e síntese do projeto, levando em consideração os requisitos de temporização;
- adaptação do projeto ao dispositivo especificado;
- criação do arquivo binário para gravação do dispositivo;
- criação de arquivos para suporte à simulação do projeto;

A análise dos erros gerados pelo relatório de compilação permitirá identificar a sua causa e a sua correção. Para quaisquer arquivos de entrada de projeto, clicando-se duas vezes numa mensagem de erro será aberta uma janela com o arquivo com problemas e o cursor será posicionado próximo à provável causa, permitindo a sua rápida correção.

1.1.3 Simulação do projeto lógico

A etapa seguinte a uma compilação sem erros do projeto consiste da sua simulação. Para isto, é utilizado um editor de formas de onda, através do qual os sinais de entrada são definidos e os sinais de saída são gerados pelo simulador, que usa para isto o projeto lógico para deduzi-las. Através da análise dos sinais lógicos, o projetista determina se o comportamento do projeto é o esperado. Se não for, ele deve voltar à etapa de entrada de projeto para corrigir o erro ou alterá-lo, compilá-lo novamente e simulá-lo. Este ciclo se repete até que o projetista *tenha assegurado* que o comportamento do projeto atende aos seus requisitos originais.

1.1.4 Gravação do código no dispositivo

Se o resultado da simulação está correto, ou seja, está de acordo com os seus requisitos lógicos, a próxima etapa (que nem sempre é realizada) consiste em configurar o dispositivo lógico escolhido (**FPGA** ou CPLD) com o projeto lógico, de tal forma que o comportamento nele seja o esperado, tal como aquele verificado no simulador. Como este processo pode envolver a gravação de um arquivo binário em memória FLASH, diz-se também que o dispositivo é gravado ao invés de configurado.
Em *projetos hierárquicos*, tipicamente a simulação é limitada aos módulos menores e a gravação aos módulos de maior complexidade ou mesmo limitado ao módulo final. *Daí a importância na simulação dos módulos intermediários, já que nem sempre seu comportamento será verificado no dispositivo.*

1.2 O módulo DE2 da Altera

O módulo experimental que será utilizado em laboratório nas práticas de VHDL é o kit DE2 da Altera, cuja fotografia é apresentada abaixo (Ver Figura 1.1). Ele contém, além de um **FPGA EP2C35F672C6N**, dispositivos periféricos tais como LEDs, chaves, interfaces de áudio e vídeo e interface Ethernet, além de memórias RAM e FLASH. Todos estes dispositivos estão disponíveis para uso, sendo descritos nos manuais de usuário e referência, disponibilizados no site da disciplina.
Um aspecto muito importante da gravação, e que é definida na etapa de entrada do projeto, é a atribuição de pinos, ou seja, em quais pinos do dispositivo lógico uma determinada entrada ou saída vai estar disponível. Para todos os sinais do projeto principal definidos como entrada ou saída (ou bidirecionais, se for o caso), deve ser feita a *atribuição a um pino do dispositivo* que seja compatível. Nos dispositivos da Altera, alguns dos pinos têm atribuição preferencial (exemplos: sinais de relógio, sinais de habilitação de portas tri-estado, sinais de inicialização) ou mesmo obrigatória (todos os sinais de alimentação, sinais de configuração do dispositivo) e não estão disponíveis para atribuição a funções lógicas do usuário.
Para a placa DE2, seu manual de usuário apresenta a atribuição que é feita para todos os periféricos e componentes disponíveis, desde os passivos, tais como as chaves deslizantes (liga/desliga), chaves de pressão, LEDs vermelhos e verdes, visores de 7-segmentos e visor de cristal líquido, até os ativos, tais como memórias RAM e FLASH, codec de áudio, codificador e decodificador de vídeo, interface Ethernet e outros. Para facilitar seu uso, foi disponibilizado um arquivo de atribuição padrão do módulo, que identifica todos estes dispositivos, bastando fazer uso de seu nome, como definido nele, para ter acesso ao dispositivo. O arquivo tem o nome *DE2_pin_assignments.csv* e se encontra disponível no *site* da disciplina.
Assim, se o objetivo é acender o LED vermelho identificado no módulo DE2 como LEDR0, basta atribuir a um pino de saída o nome LEDR[0] (note a presença do subscrito "[0]"). Se se

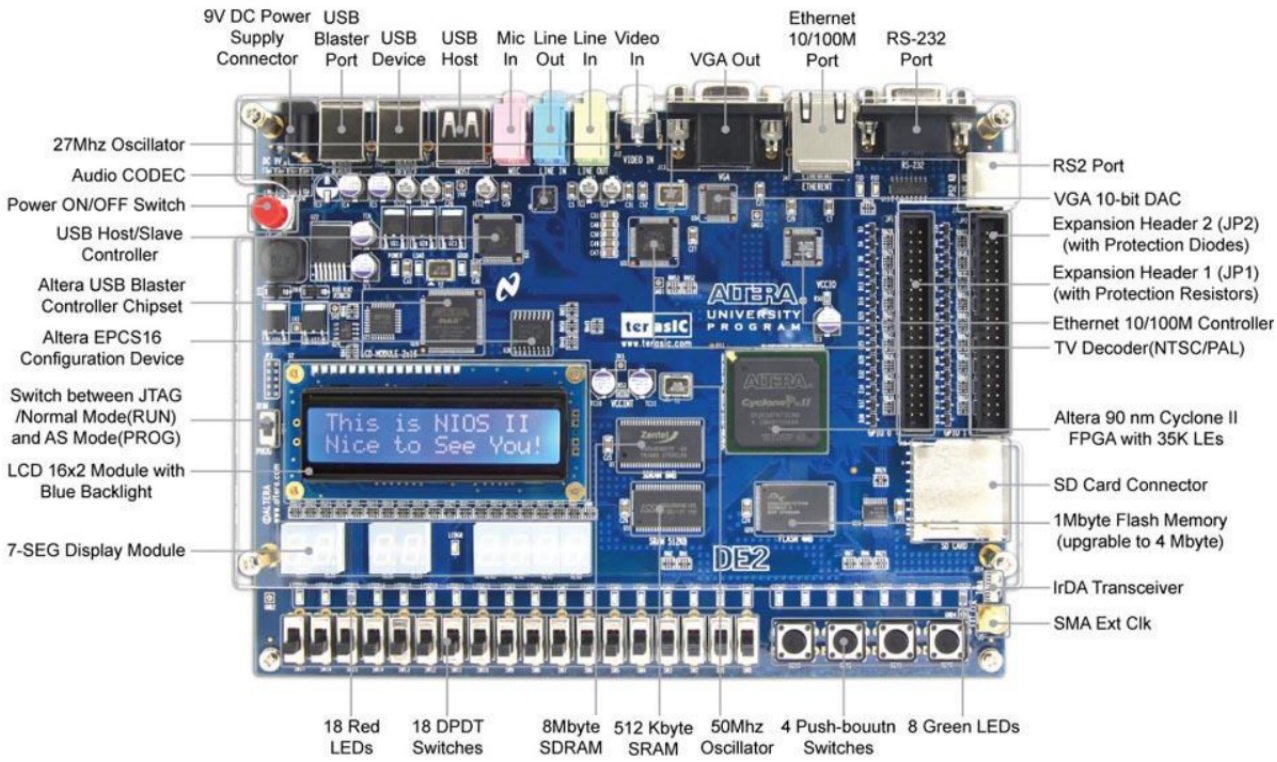


Figura 1.1: O módulo DE2 da Altera.

deseja acender todos os 18 LEDs vermelhos, a atribuição seria a um vetor de nome LEDR[17..0]. Da mesma forma, a atribuição aos outros dispositivos deve seguir a nomenclatura definida no arquivo de atribuição de pinos.

1.3 **Parte Experimental**

A parte experimental consiste em realizar um projeto de luzes rítmicas: os LEDs verdes serão acionados em uma ordem cíclica, um de cada vez, começando do LED à direita (LEDG0) e acendendo sucessivamente os LEDs à esquerda até o LEDG7, quando a sequência é invertida e os LEDs são acesos se deslocando à direita. Esta sequência se repete sem parar. Como o objetivo nesta primeira aula prática é o aprendizado do Quartus II e do módulo DE2, será utilizada a entrada de projeto através do editor gráfico mas, a partir da próxima aula, será utilizado apenas a linguagem Verilog.

O diagrama de blocos do projeto de ativação dos LEDs verdes é mostrado na figura abaixo (arquivo "Luz_ritmica.bdf"). Salve este arquivo em seu diretório de trabalho antes de iniciar as atividades.

Notem que, ao contrário do que será feito com o uso da linguagem Verilog, a lógica do funcionamento do circuito foi desenvolvida por alguém, que teve o trabalho de derivar as equações lógicas e identificar os blocos necessários. Em Verilog, o objetivo é fazer com que o compilador tenha esta função, deixando o projetista livre para tarefas mais conceituais.

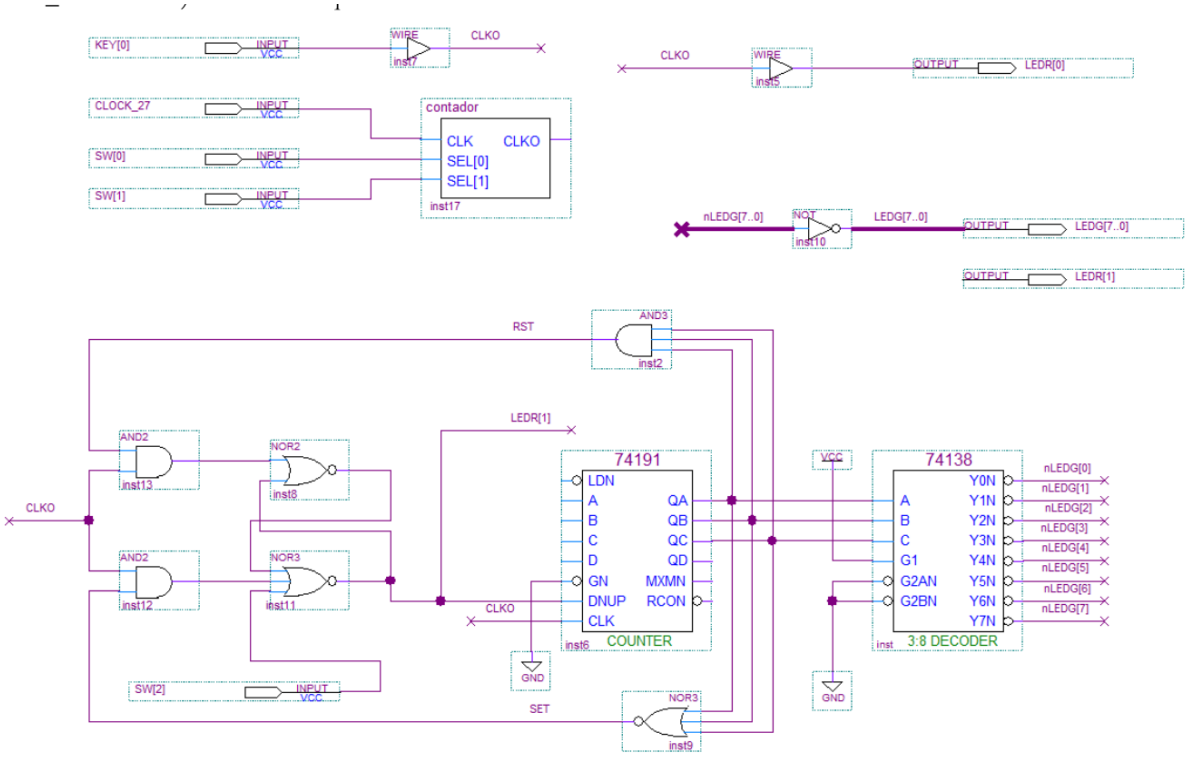


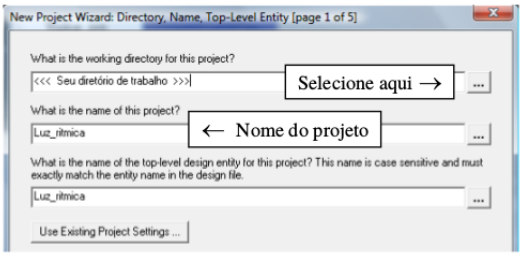
Figura 1.2: Projeto.

1.3.1 Atividade 1

Nesta atividade, você deverá verificar o funcionamento do projeto das luzes rítmicas no módulo DE2. Para isto, proceda como descrito a seguir, onde os comandos ou seleções a serem feitos no Quartus II estão identificados:

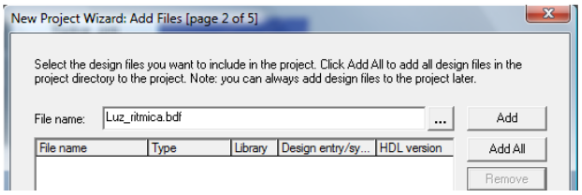
1. Criar um novo projeto: *File* $\longrightarrow$ *New Project Wizard*. Uma janela se abre informando sobre os próximos passos. Clique em *Next*.

2. Na janela de definição do diretório de trabalho e do nome do projeto, selecione o seu diretório de trabalho na primeira janela de edição, utilizando o botão de seleção de diretório à direita, e entre com o nome "Luz_ritmica" (traço sublinhado, sem acento), como mostrado a seguir. O Quartus II utiliza o mesmo nome para o projeto do módulo principal.

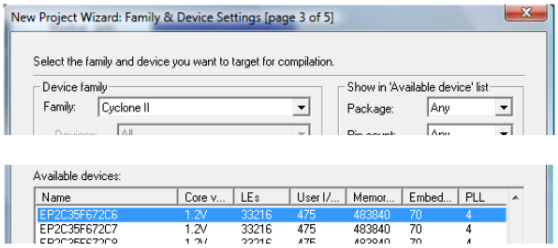


3. Na janela de identificação do arquivo que contém o projeto lógico, selecione o arquivo com o nome "Luz_ritmica", clique em *Add* (o nome do arquivo deve aparecer na janela de

arquivos do projeto) e em *Next*.



4. Na janela de definição do dispositivo lógico, selecione a família Cyclone II e o dispositivo do módulo DE2 (EP2C35F672C6N), como mostrado abaixo.



5. Selecione *Next* e *Finish* na próxima janela. O projeto foi criado e, para ver o diagrama lógico, dê um duplo clique no nome do arquivo na janela de navegação do projeto, no canto superior esquerdo.Para ter uma visão geral do diagrama lógico, selecione *View* $\longrightarrow$ *Full Screen* (para sair da visão geral faça o mesmo).

6. Note que foram utilizados os nomes dos dispositivos de entrada e saída como definidos para o módulo DE2, tais como LEDG[7..0]. Falta, agora, indicar para o Quartus II o nome do arquivo que contém tais atribuições. Para isto, selecione *Assignments* $\longrightarrow$ *Import Assignments* e selecione o arquivo "DE2 _ pin_assignments.csv".

7. O projeto está pronto para ser compilado. Para isto, selecione na barra de ferramentas a seta roxa à direita ou, no menu, a opção *Processing* $\longrightarrow$ *Start Compilation*.

8. Após a compilação do projeto, o Quartus II gera um relatório que contém uma série de mensagens de alertas e de erros que devem ser analisadas para verificar o grau dos problemas a serem resolvidos. No caso de erros, é obrigatório corrigí-los antes de continuar. No caso de alertas, é aconselhável verificar se elas não implicam em algum erro de lógica que leve a um funcionamento indesejado do projeto.

   No caso do projeto das luzes rítmicas, a grande maioria das mensagens de erro está relacionada à definição de atribuição de pinos que não estão sendo utilizados e podem, por isso, ser ignoradas.

9. Antes de verificar o funcionamento do projeto no kit, é aconselhável verificar seu funcionamento através de sua simulação. Para isso, crie um arquivo de formas de onda selecionando *File* $\longrightarrow$ *New* $\longrightarrow$*Verification/Debugging Files*$\longrightarrow$ *University Program VWF*. Na janela que se abrir, clique com o botão direito na linha logo abaixo da coluna identificada como *Name* e selecione as opções *Insert* e *Insert Node or Bus* no menu suspenso. Na próxima janela selecione *Node Finder* e, na janela que se abrirá, clique no botão *List* para listá-los.

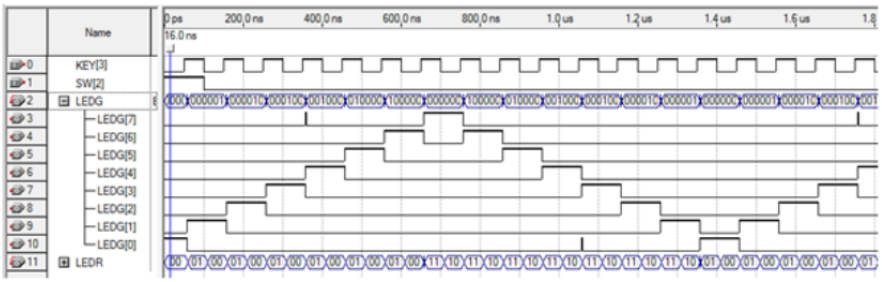
Selecione os sinais KEY(3), SW(2), LEDG(7) a LEDG(0) e LEDR(1) e LEDR(0), utilizando a tecla de controle (CTRL) no teclado e passe-os para a janela da direita clicando em '¿' na barra central. Clique em *OK* para aceitar esta seleção, selecione hexadecimal como a base dos sinais (*Radix*) e clique em OK para os sinais serem adicionados ao arquivo de simulação.

10. Na simulação, há dois parâmetros a serem definidos e que podem variar em função do projeto: o tempo de duração da simulação e a grade a ser utilizada para os sinais de relógio e seleção de sinais. Para o primeiro, selecione *Edit* $\longrightarrow$ *End Time* e selecione 2 us como tempo de duração da simulação. Para o segundo, selecione *Edit* $\longrightarrow$ *Grid Size* e selecione 100 ns como o tamanho da grade. Para ver a janela de simulação completa, com o botão direito posicionado na área de visualização dos sinais selecione a opção *Zoom* e, em seguida, *Fit in Window*, ou tecle CTRL-W. Toda a janela de 2us será visualizada.

11. Para definir a forma de onda adequada para o pulso de relógio, clique no sinal KEY(3) e toda a forma de onda será selecionada. Na barra de ferramentas vertical à esquerda clique no símbolo (*Overwrite clock*) e aceite a sugestão de período de 100ns, como definido pelo tamanho da grade. O sinal agora será substituído por um sinal de onda quadrada com este período. Ative o sinal SW(2), que serve de Reset para o circuito, durante um ciclo de relógio, como mostrado abaixo.

12. Salve o arquivo de simulação e aceite a sugestão do nome, que deverá ser igual ao do projeto lógico.

13. Para simular o projeto, selecione *Simulation*$\longrightarrow$ *Run Functional Simulation*. O projeto será simulado e os sinais de saída serão definidos no relatório de simulação produzido, como mostrado abaixo. Note que a cada instante um dos LEDs verdes é aceso e a sequência se inverte sempre que o LED[0] ou o LED[7] é atingido.



14. Uma vez verificado o funcionamento correto através da simulação, o código gerado pelo Quartus II pode ser descarregado no módulo DE2, através da ferramenta de programação. Esta ferramenta é ativada através do símbolo da barra de ferramentas ou através do comando "ToolsProgrammer" (antes disso, energize o módulo DE2 e conecte o cabo USB ao PC). Na janela que se abre, verifique se a interface USB Blaster está selecionada como o *hardware* de gravação, clique no botão "Add File" e selecione o arquivo "Luz_ritmica.sof". Em seguida, clique em "Start" e o arquivo será enviado ao módulo, para configuração do FPGA, e o funcionamento do projeto nela poderá ser verificado. Observação importante: a chave RUN/PROG deve estar na posição RUN.

Todos estes passos deverão ser tomados para cada aula de laboratório, alterando-se apenas a forma de entrada do projeto que, a partir da próxima aula, será baseada na linguagem Verilog.

2 Simulação com o ModelSim

A simulação de um projeto lógico com a ferramenta ModelSim é a técnica sugerida para versões do Quartus II atuais, não sendo mais recomendável a simulação baseada em forma de onda. Além disso, a compilação também pode ser realizada ealizada no ModelSim, tornando-o uma a ferramenta apropriada para projetar circuitos lógicos com dispositivos Altera e simulá-los. O Quartus é necessário para as funções de atribuição de pinos e gravação dos dispositivos dispositivos.

2.1 Passos para a simulação

1. O passo inicial é a criação do projeto que incluirá dois arquivos: o do projeto do SOMADOR e o de testes. Para isso, após iniciar o ModelSim, clique em *Start* e, na janela seguinte, selecione a opção "*Create New Design*". Dê um nome apropriado para o Projeto (SOMADOR, por exemplo) e navegue para o diretório onde foram colocados os dois arquivos do SOMADOR: o do projeto lógico (SOMADOR.v) e o de testes (SOMADOR_tb.v).

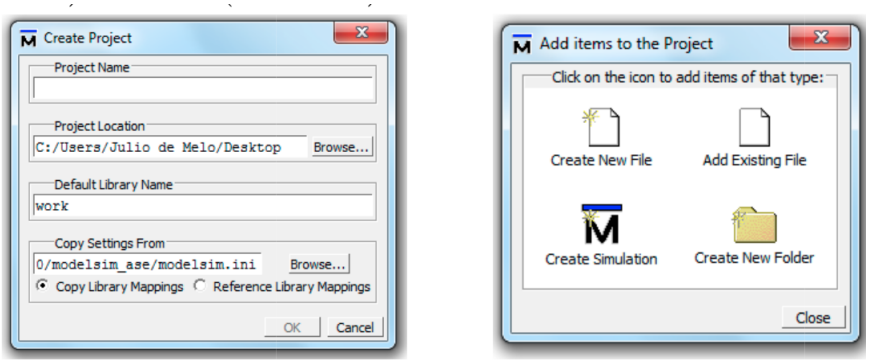


Figura 2.1: New Project com MedolSim.

2. Ao ser perguntado sobre os arquivos de projeto, selecione a opção *Add Existing File*, adicionando os dois arquivos mencionados.

3. Selecione a opção do menu *Compile*⟶ *Compile Order...* e na janela que se abrir, selecione a opção *Auto Generate*. Em seguida, compile o projeto com a opção *Compile* ⟶ *All* (ou clique no ícone correspondente na barra de ferramentasferramentas), até é que nenhuma mensagem de erro seja produzida.

4. Para simular o projeto, selecione a opção Simulate → Start Simulation e, na janela que se abrir, clique no sinal de '+' da entrada "*work*" e selecione o arquivo de testes ("SOMADOR_tb.v"). Isso cria uma nova aba na janela de projeto ("sim") e uma janela para visualização de formas de onda.

5. Para adicionar os sinais a esta janela, selecione os sinais na janela "Objects" e arraste-os para a janela de forma de onda. Defina as bases dos sinais para de decimal, selecionando todos eles e,com o botão direito do mouse, selecione a opção "Radix" e, em seguida, "Decimal".

6. Rode a simulação selecionando *Simulate* $\longrightarrow$ *Run* ou clicando no ícone correspondente. O tempo de simulação pode ser definido na lista suspensa disponível na barra de ferramentas. O "wave"das sinais pode tambem ser visualizado.

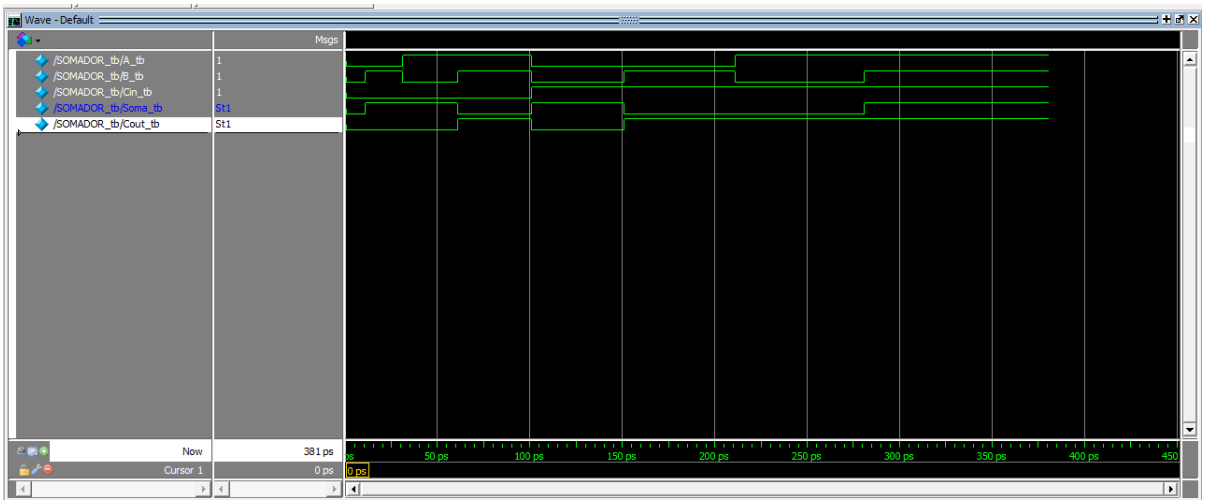


Figura 2.2: Wave da Simulação.

Objetivo

Esta prática de laboratório tem como principal objetivo possibilitar ao aluno uma experiência inicial com programação em Verilog.

Pré-requisitos

As documentações relacionadas a programação em Verilog e utilização da placa Altera estão disponíveis no moodle. Além disto estão disponíveis os códigos para utilizar os displays e leitura das chaves. Para simulação de Verilog, os alunos podem usar o Icarus Verilog (iverilog), ou o programa ModelSim de Altera que tem nas máquinas da graduação.
Para este laboratório foi necessário adicionar um *Material Complementar*. Uma primeira pasta (*Quartus Prime*) contém alguns programas de exemplo utilizados no laboratório. A segunda pasta (*ModelSim* contém vários exemplos de como usar o simulador. Em cada exemplo há um arquivo *README.txt* onde se encontra o link do tutorial para fazer cada um deles.