PROJETO DE CIRCUITOS COM QUARTUS(R) II 9.1
Versão 1.0

Laboratório Digital

Departamento de Engenharia de Computação e Sistemas Digitais

Escola Politécnica – USP – Campus Butantã – São Paulo

Marlim Pereira Menezes,
Profa. Dra. Liria Matsumoto Sato
e Prof. Dr. Edson T. Midorikawa / 2011.

Adaptado de:

PROJETO DE CIRCUITOS COM MAX+PLUS II
Edith Ranzini,
Edson Lemos Horta
e Edson T. Midorikawa / 2002.

2011

RESUMO

Este material apresenta, de uma maneira breve, o uso do Quartus(R) II 9.1 no desenvolvimento de um circuito digital.

Através de um projeto-exemplo, são apresentados os principais módulos do programa e os comandos essenciais.

1. INTRODUÇÃO

Um sistema digital (SD) é um sistema com entradas e saídas, como qualquer outro sistema real, vide figura 1.

Figura 1 – Sistema digital geral.

O fato que difere um SD de outros diz respeito ao tipo de entrada e saída que são manipulados: os dados são digitados, ou seja, são representados por um conjunto finito de sinais binários e discretos.

1. METODOLOGIA DE PROJETO UTILIZANDO O QUARTUS(R) II

O projeto de circuitos digitais pode ser automatizado através do uso de ferramentas de EDA (Electronic Design Automation). Uma destas ferramentas é a Quartus(R) II, da Altera, utilizada na descrição, compilação, simulação e programação de sistemas digitais implementados através de EPLDs.

Como mostrado na figura 1.1, o projeto de um Sistema Digital (SD) pode ser dividido nas seguintes etapas: entrada de dados do circuito, compilação, simulação e programação de um dispositivo lógico programável (EPLD).

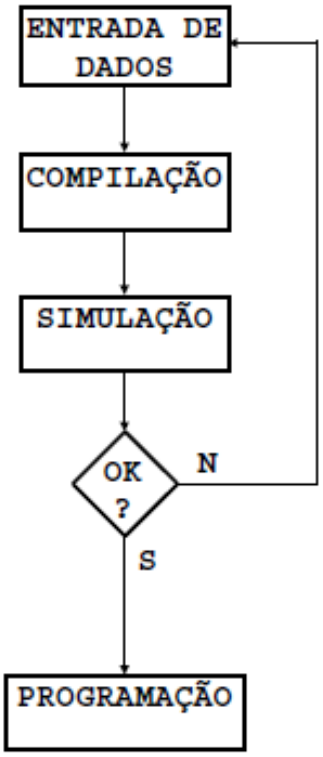


Figura 1.1 – Metodologia de Projeto de SD com o Quartus(R) II.

A **ENTRADA DE DADOS** pode ser realizada de três maneiras distintas:

- ***Diagrama Lógico*** (Captura Esquemática) – neste modo, o projetista tem duas opções para descrever o projeto. Na primeira opção, o QUARTUS II permite a importação de arquivos construídos com outras ferramentas. Na segunda opção, que é a mais utilizada, o Quartus(R) II disponibiliza um aplicativo denominado Editor Gráfico (**Block Diagram/Schematic File**). Os símbolos utilizados no diagrama lógico podem ser obtidos de uma biblioteca padrão, ou podem ser gerados pelo próprio projetista, a partir de outros projetos já implementados, permitindo a descrição de forma hierárquica de um sistema digital. A biblioteca padrão do Editor Gráfico possui símbolos que representam todos os circuitos integrados da família 74xx, permitindo que um projetista familiarizado com estes componentes descreva o sistema digital de maneira mais rápida e eficiente. É importante salientar que estes símbolos apenas implementam as funções lógicas de um 74xx em uma EPLD, não possuindo nenhuma característica física dos CIs comerciais (pinagem, tempos de propagação, consumo, etc), pois estas últimas serão determinadas pelo tipo de EPLD na qual o sistema será implementado. Além destes símbolos que representam a família 74xx, o Quartus(R) II também possui bibliotecas com funções lógicas básicas (flip-flops, portas lógicas, etc) e avançadas (contadores especiais, microprocessadores, etc), sendo que estas últimas bibliotecas devem ser adquiridas de terceiros.
- ***Arquivo Texto*** – este modo permite a descrição de um sistema digital através de linguagens de descrição de hardware (HDL). O Quartus(R) II aceita três tipos de HDLs: AHDL, que é uma linguagem proprietária da ALTERA; Verilog e VHDL, que são linguagens padronizadas pelo IEEE, utilizadas mundialmente. O Quartus(R) II possui editores scripts para diversos formatos (Texto, VHDL, AHDL, TCL, Verilog e outros) para apoiar essa forma de descrição do SD.

- ***Formas de Onda*** – este modo permite que o projetista descreva o comportamento de um sistema digital através do desenho das formas de onda na entrada e na saída do mesmo. Este recurso é utilizado apenas quando o SD é simples e "bem comportado" (por exemplo, um contador síncrono). A descrição é feita através de um Editor de Formas de Ondas (**Vector Waveform File**).

Na **COMPILAÇÃO,** o projetista determina qual será a EPLD que deverá implementar o sistema digital e o Quartus(R) II procura seguir esta diretriz. Caso não consiga, é fornecida uma mensagem de erro, e projetista pode optar por escolher outra EPLD para implementar o seu sistema, ou deixar que a ferramenta implemente o projeto em mais de uma EPLD do tipo escolhido. A compilação também é responsável pela geração de todos os arquivos necessários à simulação e programação da EPLD.

Na **SIMULAÇÃO**, é possível descrever cada uma das formas de onda de entrada do SD e observar as formas de onda de saída, geradas pela ferramenta. Com isto o projetista consegue verificar o funcionamento do SD antes de implementá-lo fisicamente, corrigindo eventuais erros que possam ter ocorrido no projeto. Para a simulação também é utilizado o Editor de Formas de Ondas.

Na **PROGRAMAÇÃO,** os arquivos gerados pela compilação são transferidos para a EPLD, programando a mesma para funcionar de acordo com o projeto descrito anteriormente. Esta transferência pode ser feita através de programadores de componentes ou cabos especiais, conectados ao PC.

Para o projeto do SD através da ferramenta Quartus(R) II, será adotada a seguinte nomenclatura para os arquivos de projeto:

- Fluxo de Dados : ..... nome_do_projeto_FD;
- Unidade de Controle : ..... nome_do_projeto_UC;
- Sistema Digital Completo : ..... nome_do_projeto_SD.

Por exemplo, para um SD denominado MULTI, há os seguintes “projetos”:

- MULTI_FD (Fluxo de Dados);
- MULTI_UC (Unidade de Controle);
- MULTI_SD (Sistema Digital Completo).

2. PROJETO-EXEMPLO

É apresentado nesta seção o desenvolvimento do projeto de um sistema digital (muito simples), utilizando a metodologia descrita com o Quartus(R) II.

2.1. Especificação do Projeto

Projetar um Sistema Digital (SD) que seja responsável pelo armazenamento de dois números quaisquer (4 bits, cada um), que serão introduzidos no mesmo através de uma única via de dados.

Após esta programação, o SD deve disponibilizar os números armazenados, um de cada vez, em uma única via de dados de saída. Tanto o armazenamento quanto a saída dos dados devem ocorrer sincronamente com o CLOCK do SD.

Para realizar esta tarefa, o SD deve possuir os seguintes sinais:

- IN[1..4] – entrada de dados de quatro bits;
- N1 – armazena primeiro valor de quatro bits presente na entrada IN;
- N2 – armazena segundo valor de quatro bits presente na entrada IN;
- M1 – apresenta na saída de dados OUT o primeiro valor armazenado;
- M2 – apresenta na saída de dados OUT o segundo valor armazenado;
- OUT[1..4] – saída de dados de quatro bits.

A figura 2.1 abaixo mostra o SD com os sinais descritos anteriormente:

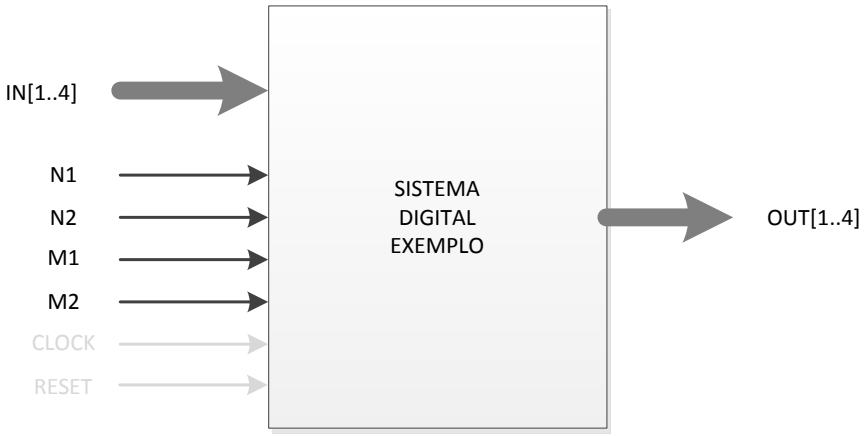


Figura 2.1 – Esquema do projeto exemplo.

Para que o primeiro número seja armazenado no sistema, ele deve ser colocado na entrada de dados IN e o sinal N1 deve ser ativado. De maneira análoga, o segundo número é armazenado quando o mesmo é colocado na entrada IN e o sinal N2 é ativado. Para mostrar cada um destes números na saída OUT, os sinais M1 ou M2 devem ser acionados. Caso seja acionado M1, a saída OUT deverá mostrar o primeiro número. Caso seja acionado M2, a saída OUT deverá mostrar o segundo número.

Convém ressaltar que todos os sinais são definidos como **ativo em alto**, ou seja, para acioná-lo, deve-se colocar um valor lógico ALTO.

2.2. Definição do Sistema Digital do Projeto

O SD especificado no item 2.1 pode ser projetado de diversas maneiras. Uma delas é baseada no particionamento visto no item 3. Portanto, é necessário dividir o SD em Fluxo de Dados (FD) e Unidade de Controle (UC).

O FD deve conter os elementos de transformação e/ou armazenamento do SD. No caso do SD estar sendo projetado, não serão necessárias transformações de dados. Como serão armazenados dois números quaisquer e apenas, um deles será apresentado na via de saída, uma das possíveis soluções é utilizar três registradores: **R1** para armazenar o primeiro número, **R2** para armazenar o segundo número e **R3** para armazenar o número que será mostrado na saída. Para que o registrador **R3** possa receber o conteúdo de **R1** ou **R2**, é necessária a colocação de um elemento que permita a escolha de uma, dentre duas entradas disponíveis. Esta função é executada por um multiplexador 2x1.

Para que cada um dos elementos do FD execute as suas funções de acordo com a especificação do projeto, são necessários alguns sinais de controle, a saber, EN ("enable"-habilitação) de cada um dos registradores e SEL (seleção) do multiplexador. Estes sinais devem são gerados pela Unidade de Controle (UC). A ordem em que os mesmos serão gerados irá depender dos sinais de controle externos ao SD, a saber, N1, N2, M1 e M2.

Finalmente, de acordo a definição acima, obtém-se a partição do SD que está sendo projetado, que é mostrada na figura 2.2.

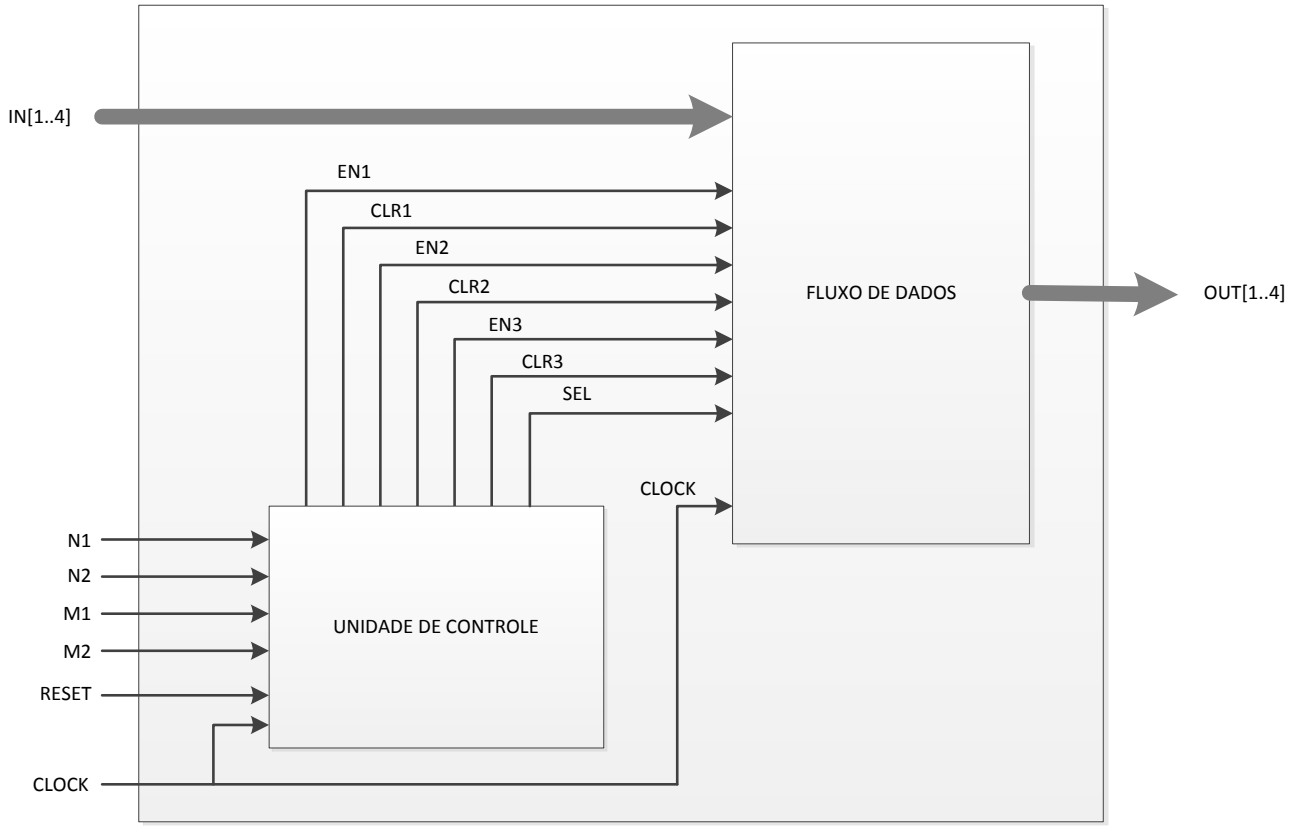


Figura 2.2 – Sistema Digital do projeto.

Nos próximos itens serão detalhados todos os blocos acima e como projetar os mesmos, utilizando os recursos do Quartus(R) II.

2.3. Projeto do Fluxo de Dados

Um sistema é dito estar no nível de transferência de registradores (RTL – *Register Transfer Level*) se as informações fluírem pelo circuito através de componentes de memória ou registro de dados (registradores). À medida que os dados fluem pelo circuito, estes podem ser manipulador por blocos combinatórios implementando uma determinada lógica, figura 2.3.

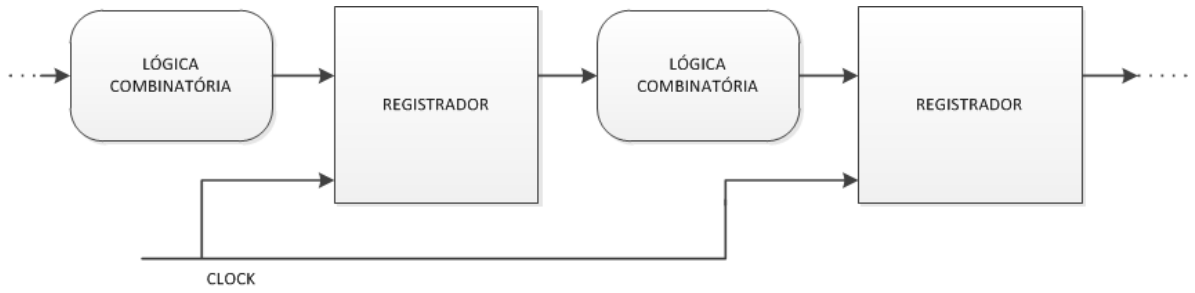


Figura 2.3 – Circuito no nível RTL.

O projeto do fluxo de dados pode ser divido em duas etapas principais:

- **descrição e compilação**: o fluxo de dados é descrito através de um diagrama lógico e, em seguida, sintetizado para uma EPLD específica;
- **simulação**: as formas de onda dos sinais de entrada do fluxo de dados são descritas e o Quartus(R) II fornece as formas de onda dos sinais de saída, seguindo os parâmetros gerados na etapa anterior.

2.3.1. Descrição e compilação do Fluxo de Dados (FD)

O diagrama lógico que representa o FD do SD é gerado a partir de um Diagrama de Blocos, como o da figura 2.5. As etapas seguintes consistem da escolha dos componentes que realizam as funções dos blocos, da descrição do diagrama lógico e da compilação na EPLD escolhida.

Para o projeto, foram selecionados os componentes da biblioteca da família 74xx:

R1 = R2 = R3 = 74173
MUX2x1 = 74157

que realizam funções compatíveis com as dos blocos do diagrama. Como adotaram-se, no ASM, sinais de enable “active-high”, e como esses sinais, nos componentes selecionados, são “active-low”, será necessária a utilização de inversores (símbolo NOT).

O diagrama lógico será desenhado com o auxílio do Editor Gráfico e o projeto será “compilado” num componente de família (Cyclone II), que é o EP2C35F672C6.

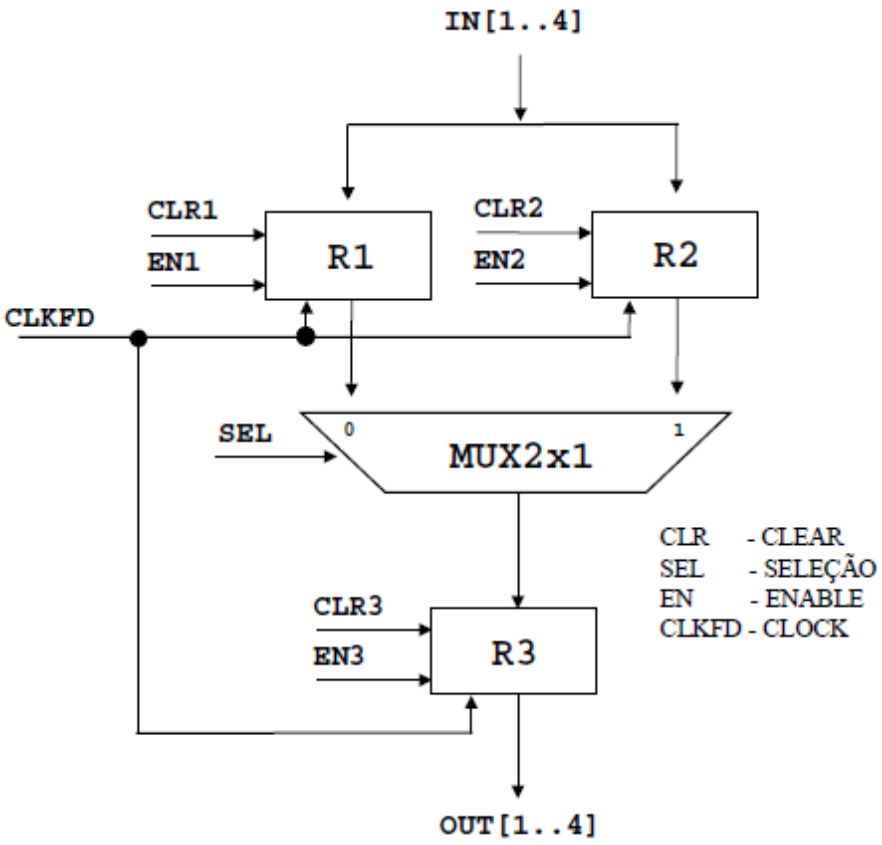


Figura 2.5 – Diagrama em blocos do FD do projeto.

Utilização do Quartus(R) II

Para a descrição e compilação do projeto do FD devem ser adotados os procedimentos a seguir:

a) Iniciando o Quartus(R) II 9.1.

Localize o ícone de partida do Quartus(R) II na área de trabalho do Windows, dê duplo clique ou selecione-o com um clique do mouse em seguida pressione a tecla **ENTER**, ou um clique se o mesmo estiver na barra de ferramentas. Veja as figuras 2.5 e 2.6:

Figura 2.5 - Ícone de partida do Altera Quartus(R) II 9.1 Web Edition na área de trabalho do Windows.

Figura 2.6 - Ícone de partida do Altera Quartus(R) II 9.1 Web Edition na barra de ferramenta do Windows.

No caso do ícone não estar na área de trabalho, siga um dos roteiros a seguir:

Em Português:

- **Iniciar -> Todos os Programas -> Altera -> Quartus II 9.1 Web Edition -> Quartus II 9.1 Web Edition** (**ENTER** ou clique simples).

Em Inglês:

- **Start -> All Programs -> Altera -> Quartus II 9.1 Web Edition -> Quartus II 9.1 Web Edition** (**ENTER** ou clique simples).

Se aparecer uma janela intitulada **Getting Started With Quartus(R) II Software**, então clique com o botão esquerdo do mouse no "**X**" localizado no canto superior direito, para fechá-la. Aguarde a janela de Splash sumir e o Quartus(R) II estará pronto para uso.

b) ***Criando o projeto digital do FD***

Lembrete: Se não se deseja criar um projeto novo, e sim prosseguir o desenvolvimento de um projeto já existente, ou utilizar um arquivo esquemático de um circuito já editado, os respectivos procedimentos estão apresentados na seção **Outras Informações** da apostila *Tutorial para Criar e Simular Circuitos Digitais no Altera Quartus(R) II – versão 9.1 – Versão 1.1***.**

- **File -> New Project Wizard...** (na janela New Project Wizard: Introduction) **-> Next >**;

Aparecerá uma sequência de cinco páginas numeradas, mas não será necessário preencher todas neste tutorial. Para cancelar a criação do projeto, em qualquer momento, basta clicar no botão **Cancel** da página corrente.

Preencha os campos necessários ao projeto nas páginas conforme indicado nas linhas que se seguem:

- Tela 1 de 5: **New Project Wizard: Directory, Name, Top-Level Entity**
  - Na primeira caixa de texto, clique no botão () para escolher uma pasta onde será criado o seu projeto, na segunda caixa de texto digite **tutorial_fd** para o nome do projeto e, a terceira será automaticamente preenchida com o texto digitado na segunda caixa de texto, mas poderá ser alterado. Em nosso caso, aceitaremos o mesmo nome dado ao projeto (**tutorial_fd**). Veja a imagem na figura 2.7:

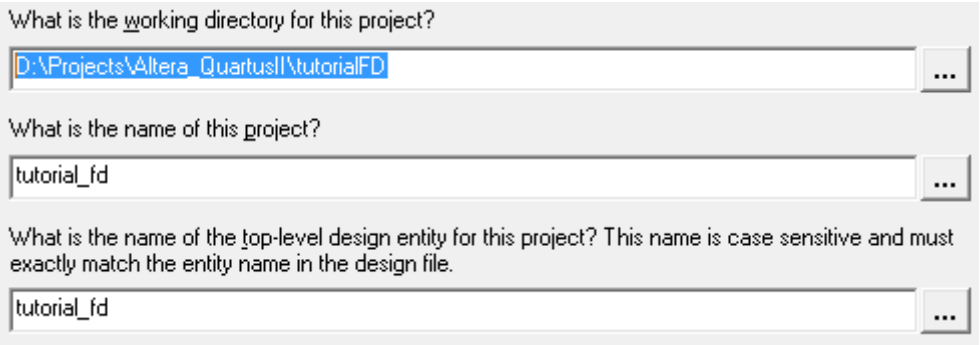


Figura 2.7: Tela 1 da criação de projeto.

- Pressione no botão **Next >** para avançar;

➢ Tela 2 de 5: **New Project Wizard: Add Files**
- Não será inserido arquivo externo, então pressione em **Next >** para avançar;

➢ Página 3 de 5: **New Project Wizard: Family & Device Settings**
- Aqui é possível selecionar o FPGA a ser utilizado no projeto durante a sua criação. Esta opção será mostrada mais adiante no item **d)** desta apostila;
- Pressione em **Next >** para avançar;

➢ Página 4 de 5: **New Project Wizard: EDA Tool Settings**
- Ignore esta página e pressione em **Next >** para avançar;

➢ Página 5 de 5: **New Project Wizard: Summary**
- Esta página contém um resumo das configurações feitas anteriormente para o projeto sendo criado;
- Pressione em **Finish** para concluir a criação do projeto.

c) *Criando o arquivo esquemático do circuito digital do Fluxo de Dados*

Nesta seção prepararemos o ambiente de desenvolvimento para receber o circuito digital que será editado e simulado. Para isto execute os comandos a seguir:

➢ **File -> New** (Aparecerá uma janela intitulada New);
➢ **Design Files -> Block Diagram/Schematic File -> OK**, note que a área à direita da janela principal do Quartus(R) II aparecerá com uma grade de pontos (esses pontos são utilizados como coordenadas da tela para a disposição dos diversos componentes que constituem o circuito digital).

Na margem esquerda da área de desenho há uma régua vertical com alguns botões, que disponibilizam as opções de projeto tais como a biblioteca dos componentes, caixa de edição de texto, lupa para zoom etc. Vide figura 2.8.

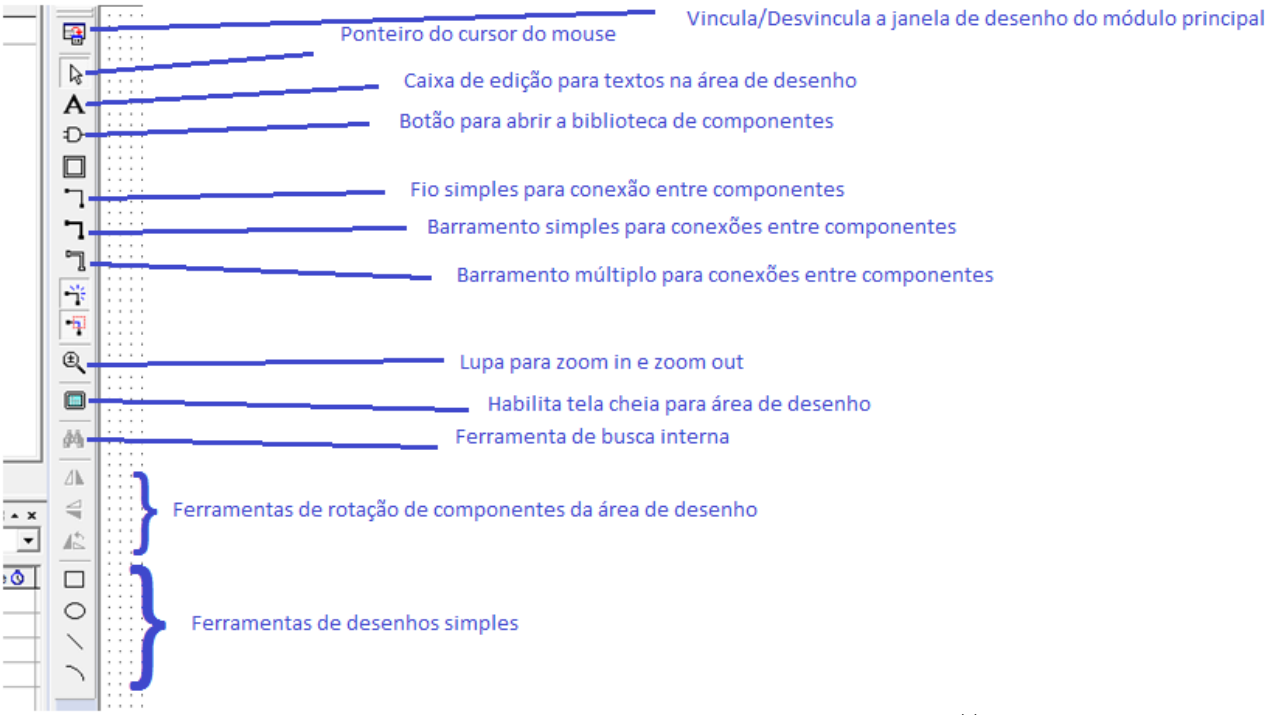


Figura 2.8 - Régua de ferramentas de desenvolvimento do Quartus(R) II 9.1.

d) *Escolhendo o FPGA a ser usado no projeto*

Para especificar o FPGA que será usado no projeto execute as seguintes instruções:

- **Assignments -> Device...** (Aparecerá a janela Settings); então selecione:
- **(**Category**: Device) -> (**Family**: Cyclone II) -> (**Available devices: Name**: EP2C35F672C6) -> OK**
  (Este chip FPGA foi selecionado porque é o que tem no Laboratório Digital).

Para verificar qual é o FPGA sendo usado no projeto, basta observar a região **Project Navigator: Entity**, normalmente localizado à esquerda do ambiente do Quartus(R) II 9.1. Veja a figura 2.9.

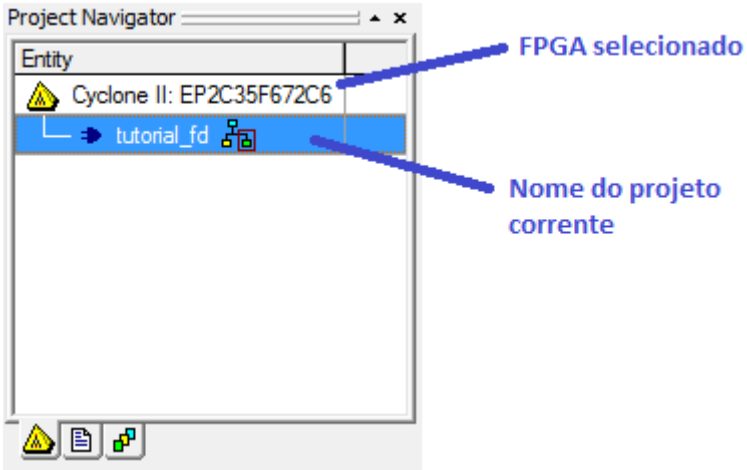


Figura 2.9 – **Project Navigator: Entity** com o FPGA selecionado.

e) *Colocando os componentes na área de desenho*

A figura 2.10, a seguir, apresenta o diagrama completo do esquema elétrico do fluxo de dados do projeto.

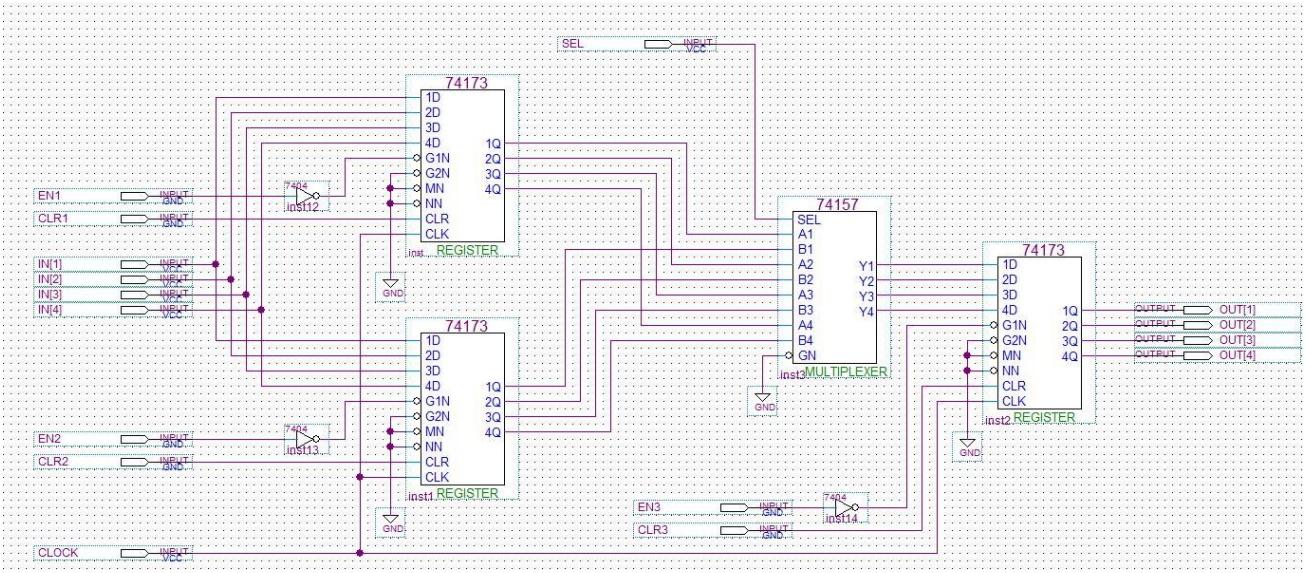


Figura 2.10 – Diagrama do fluxo de dados do projeto exemplo.

Execute uma das sequências de comandos abaixo para abrir a janela **Symbol** contendo a biblioteca dos diversos componentes disponibilizados pelo Quartus(R) II 9.1.

- **[Botão direito do mouse**, na área de desenho**] -> Insert -> Symbol...** ou;
- **[Botão direito do mouse**, na área de desenho**] -> Insert -> Symbol as Block...** ou;
- Clique no botão com o desenho de uma porta lógica ( ), localizado na régua de ferramentas da figura 2.8.

Na janela **Symbol**, execute a sequência de comandos:

Se você desejar mais de uma cópia do componente selecionado, então ative o recurso (Repeat-insert mode) caso contrário mantenho-o desativado (Repeat-insert mode). Este recurso estando ativado não nos impede de colocar apenas uma cópia do componente desejado na área de desenho, bastando para isto pressionar a tecla **[ESC]** ou executando a sequência de comandos com o mouse: **[Botão direito do mouse] -> Cancel**.

Vamos colocar alguns dos componentes necessários ao nosso projeto na área de desenho. Para isto, execute os comandos a seguir:

- **( ):** Execute os comandos conforme mostrados na figura 2.11. Note que o nome do chip da família 74xx foi digitado diretamente na caixa de edição **Name**. Lembre-se de clicar em (OK) ou pressionar a tecla ENTER para fechar esta janela e disponibilizar o componente para inclusão na área de desenho. Estando com o componente selecionado,

suspenso no ponteiro do mouse, na área de desenho, basta pressionar o botão esquerdo do mouse para liberá-lo na posição desejada.

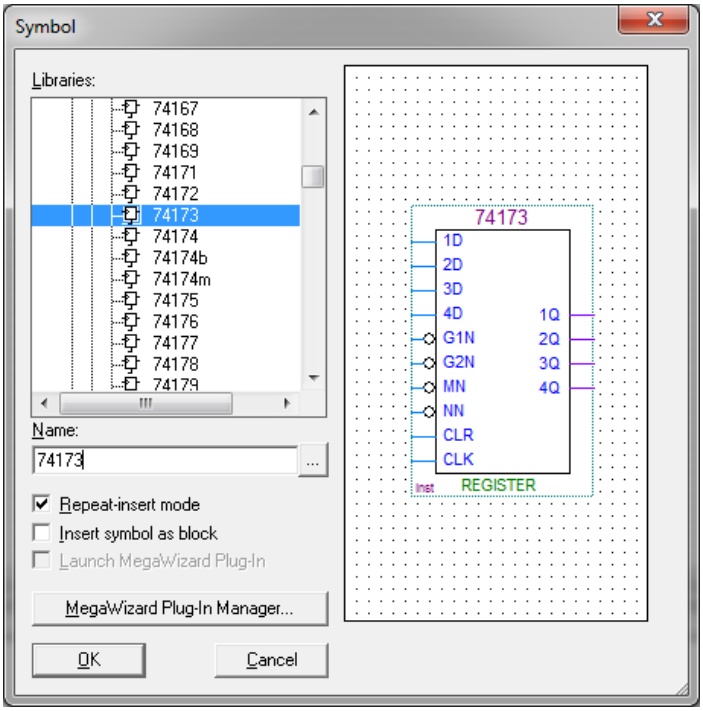


Figura 2.11 – Tela de inclusão de objetos (componentes e portas lógicas) em um esquema digital, através da digitação do seu nome na caixa de edição **Name**.

- **(**  **):** Para a inclusão dos demais componentes, siga o procedimento anterior ou o descrito na figura 2.12 onde podemos selecioná-los navegando nas opções da biblioteca mostrada no formato de estrutura de diretórios:

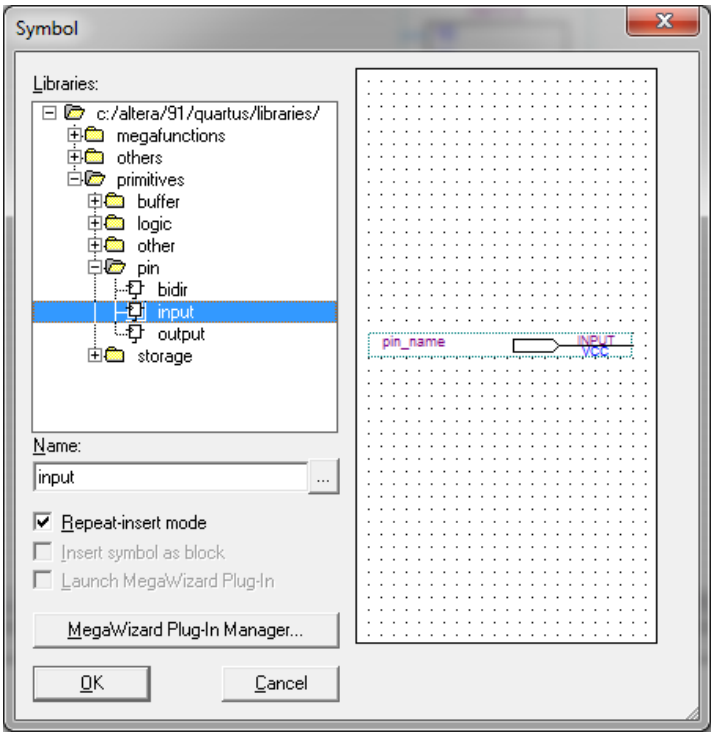


Figura 2.12 – Tela de inclusão de objetos (componentes e portas lógicas) em um esquema digital, através da seleção do mesmo na biblioteca em forma de estrutura de diretórios.

Se houver necessidade de girar algum objeto (ou componente) do circuito na área de desenho, selecione um de cada vez, ou quantos se desejar ao mesmo tempo, (clicando com o mouse sobre ele ou o conjunto selecionado) e execute os seguintes passos:

- **[Botão direito do mouse] -> Rotate by Degrees -> 270**

Por fim, reorganize todos os componentes na área de desenho de modo a ficarem dispostos como na figura 2.10. Pronto, então salve o projeto em disco, conforme descrito no item **f)**.

f) ***Salvando o projeto em disco***

Para salvar o projeto em disco execute os comandos a seguir:

- **File -> Save -> (**Save as type**: Block Diagram/Schematic File (*.bdf)) -> (**File Name**: tutorial_fd) -> (**Save in**: tutorialFD) -> Save**.

g) ***Dando nomes aos componentes***

Para dar nome a um componente basta dar duplo clique sobre o componente desejado e seguir os passos (use a figura 2.10 como referência, se necessário) abaixo:

Pinos de entrada: EN1, CLR1, EN2, CLR2, ...

- **Pin name(s): EN1 -> OK** (Na janela "**Pin Properties**").
- **Pin name(s): CLR1 -> OK** (Na janela "**Pin Properties**").
- **Pin name(s): EN2 -> OK** (Na janela "**Pin Properties**").
- **Pin name(s): CLR2 -> OK** (Na janela "**Pin Properties**").
- **…**

Pinos de saída: OUT[1], OUT[2], OUT[3], OUT[4].

- **Pin name(s): OUT[1] -> OK** (Na janela "**Pin Properties**").
- **Pin name(s): OUT[2] -> OK** (Na janela "**Pin Properties**").
- **Pin name(s): OUT[3] -> OK** (Na janela "**Pin Properties**").
- **Pin name(s): OUT[4] -> OK** (Na janela "**Pin Properties**").

Siga os procedimentos acima para nomear (ou renomear) todos os componentes existentes no esquema elétrico do circuito digital, conforme mostrado na figura 2.10.

h) ***Conectando os diversos componentes***

Conectar os componentes do circuito (veja a figura 2.10) é muito simples, bastando para isto posicionarmos o ponteiro do mouse sobre o terminal desejado do componente de origem,

pressionar o botão esquerdo, mantê-lo pressionado e arrastar até o ponto ou terminal desejado do componente destino, soltando-o em seguida. O aspecto do ponteiro do mouse será uma cruz com o desenho ( ) **Fio simples**, vide figura 2.8.

Lembrete: É boa prática de desenvolvimento, usando computador, **SEMPRE** salvar o arquivo em disco, para prevenir perda de dados em caso de eventual queda de energia ou pane na máquina.

i) *Compilando o projeto*

Uma vez terminado o desenho do circuito, devemos compilá-lo, para em seguida fazermos a simulação. A compilação é feita seguindo-se os passos abaixo:

- **Processing -> Start Compilation** ou
- **[Ctrl-L]** mantendo a área de desenho com o circuito digital em foco, ou
- **Processing -> Compiler Tool -> Start** ou
- um clique no botão indicado na figura 2.13, da barra de ferramentas do Quartus(R) II 9.1.

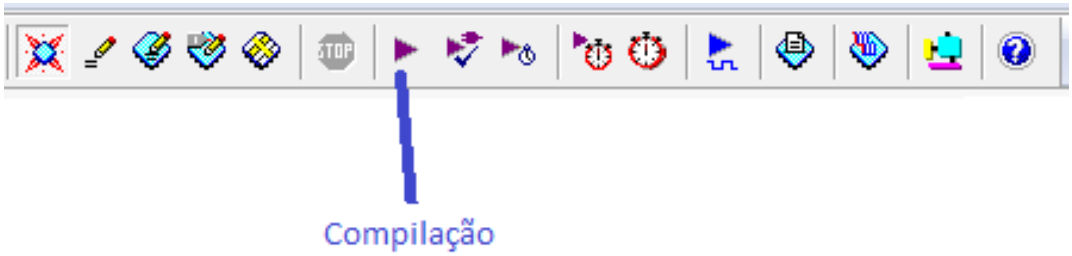


Figura 2.13 - Barra de ferramentas do Quartus(R) II 9.1, destacando o botão de compilação.

Durante a compilação diversos relatórios são gerados e mantidos à disposição do usuário.

2.3.2. Simulação do Fluxo de Dados

Para verificar se o projeto está funcionando de acordo com sua especificação inicial, é necessária a simulação do mesmo. Através do Quartus(R) II esta simulação é bastante facilitada, possibilitando a visualização das formas de onda de entrada e saída do circuito, além de prever com exatidão todos os tempos de propagação entre os sinais do projeto.

Para simular um circuito digital é necessário adotar alguns procedimentos, os quais são apresentados nos itens que se seguem:

a) *Criando uma simulação*

Execute os comandos a seguir, para criar uma simulação:

- **File -> New -> [+] Verification/Debugging Files -> Vector Waveform File -> OK**

Neste momento um arquivo que conterá os dados da simulação será criado e uma janela própria da ferramenta de simulação do Quartus(R) II 9.1 será aberta e ficará pronta para uso. Essa janela é dividida em quatro partes:

- Régua de ferramentas para as configurações da simulação;
- Régua com as bases de tempo;
- Coluna para os nomes dos pontos de teste com a base de tempo e;
- Área das cartas de tempo (timing chart) do circuito.

b) *Salvando a simulação*

Salve o arquivo de simulação com o mesmo nome do projeto, ou seja, **tutorial_fd**. Para isto execute a sequência de comandos a seguir:

- **File -> Save -> (**Save as type: Vector Waveform File (*.vwf)**) -> (**Save in: tutorialFD**) -> (**File name: **tutorial_fd) -> Save**

c) *Incluindo os pontos de teste para a Simulação*

Clique o botão direito do mouse sobre a região dos pontos de teste do circuito simulado (colunas: Name e Value at). Veja a figura 2.14:

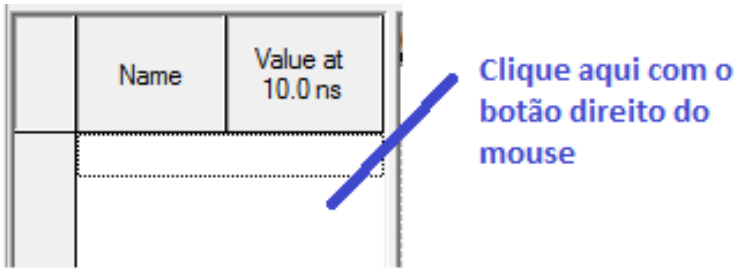


Figura 2.14 – Área destinada à declaração dos pontos de teste do circuito.

Podemos entrar com um ponto de teste de cada vez ou um conjunto, dentre os disponíveis, em uma operação só. Veremos os dois jeitos:

Entrando um ponto de teste por vez:

- [**Botão Diteiro do Mouse**] **-> Insert -> Insert Node or Bus... -> (**Name**: A) -> OK**

Entrando um conjunto dentre os pontos de teste disponíveis ou todos de uma só vez:

- [**Botão Diteiro do Mouse**] **-> Insert -> Insert Node or Bus... -> Node Finder... -> (**Named**: *) -> (**Filter**: Pins: All) -> List -> [>>] -> OK -> OK**

d) *Descrever as formas de onda*

As formas de onda da figura 2.15 mostram uma possível simulação, com a entrada do valor 6 em R1 e do valor 7 em R2 e, posteriormente, a saída destes valores na saída. Ao final, realiza-se uma reinicialização do circuito. A simulação é conduzida com o acionamento correto dos sinais de controle.

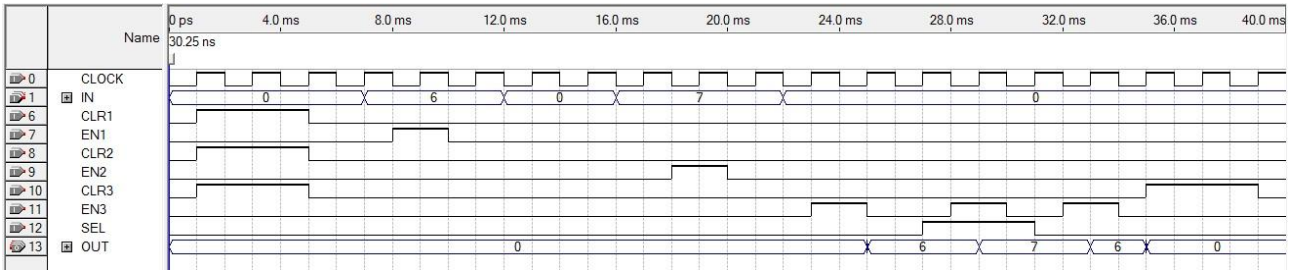


Figura 2.15 – Formas de onda da simulação do fluxo de dados.

Siga os passos abaixo: Utilize a região dos pontos de teste, vide figura 2.15:

- **CLOCK (linha 0): [Clique no desenho**  **] ->**  **-> (**Radix**: Binary) ->Timing -> (**Multiplied by**: 1) -> OK**

Siga este procedimento para definir a forma de onda de todos os sinais.

e) *Configurando a simulação*

O Altera Quartus(R) II 9.1 tem três modos de simulação: Functional; Timing e; Timing using Fast Timing Model. O default é o modo Timing que considera os atrasos internos do tempo no FPGA e será aqui adotado.

Vamos configurar o instante final (**End Time**) e a largura da grade de tempos (**Grid Size**) da simulação. Clique na aba **tutorial_fd.vwf**, para trazer a sua janela ao primeiro plano do ambiente de desenvolvimento.

- **Edit -> End Time... -> (**Time**: 2.0 µs) -> OK** (Para podermos analisar melhor o comportamento do circuito)
- **Edit -> Grid Size… -> (**Period**: 20.0 ns) -> OK**

f) *Executando a simulação*

A simulação pode ser iniciada por três caminhos diferentes:

1. Clique na aba **[Quartus II] -> Simulator Tool -> Start -> OK** (Janela de Diálogo informando o resultado da compilação) **-> Report**;

2. **Processing -> Start Simulation -> OK** (Janela de Diálogo informando o resultado da compilação) e;
3. Através da régua de ferramentas, conforme indicado na figura 2.16.

Caso a simulação tenha sido iniciada a partir do menu (caminho 1) ou pressionando-se o botão em destaque na figura 2.16 (caminho 3), então poderemos visualizar os relatórios (inclusive a carta de tempos) executando a sequência de comandos: **Processing -> Simulation Report**, ou pelo atalho [**Ctrl+Shift+R**].

A figura 10 mostra o botão da barra de ferramentas usado para executar a simulação.

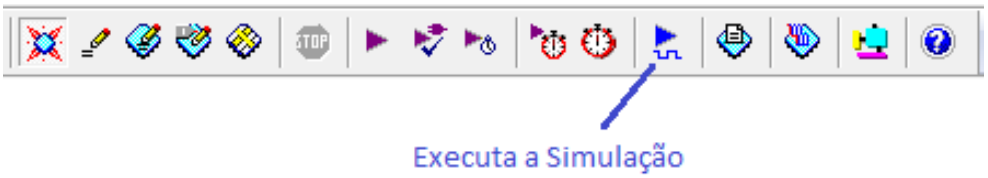


Figura 2.16 - Botão para iniciar a simulação no Simulador do Quartus(R) II 9.1.

O resultado da simulação é mostrado na figura 2.17:

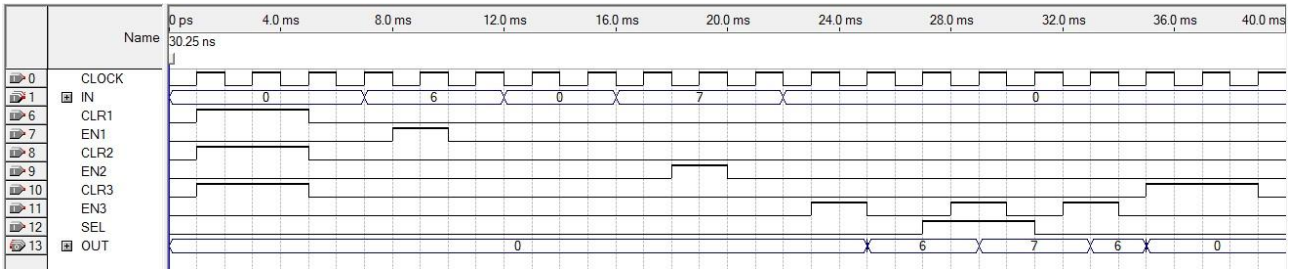


Figura 2.17 – Carta de tempos (Timing Chart) da simulação no modo Timing.

Note que o FPGA escolhido (Cyclone II – EP2C35F672C6) apresenta atraso significativo nos pinos de saídas OUT[1..4] em relação aos pinos de entrada do circuito do fluxo de dados.

g) *Geração do componente do FD*

A transformação de circuitos digitais inteiros em "simples" componente é importante para a diminuição da complexidade do sistema digital sendo projetado, pois isso acarreta na diminuição de eventuais erros de projeto e, também um melhor entendimento da funcionalidade do mesmo de forma mais abstrata.

Depois de implementado, compilado e simulado um circuito digital pode ser facilmente transformado em um componente pelo Quartus(R) II de forma extremamente simples, bastando para isto executar apenas uns poucos comandos como mostrado a seguir:

NOTA: Antes de executar os comandos abaixo certifique-se de estar com a área de desenho de circuitos (**Block Diagram/Schematic File**) ativa com o circuito digital a ser transformado em componente, carregado.

Dessa forma, vamos transformar o circuito digital do fluxo de dados, vide figura 2.24, em componente o qual será utilizado no projeto do SD completo.

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

A figura 2.18 mostra a sequência de comandos acima aplicada no Quartus(R) II.

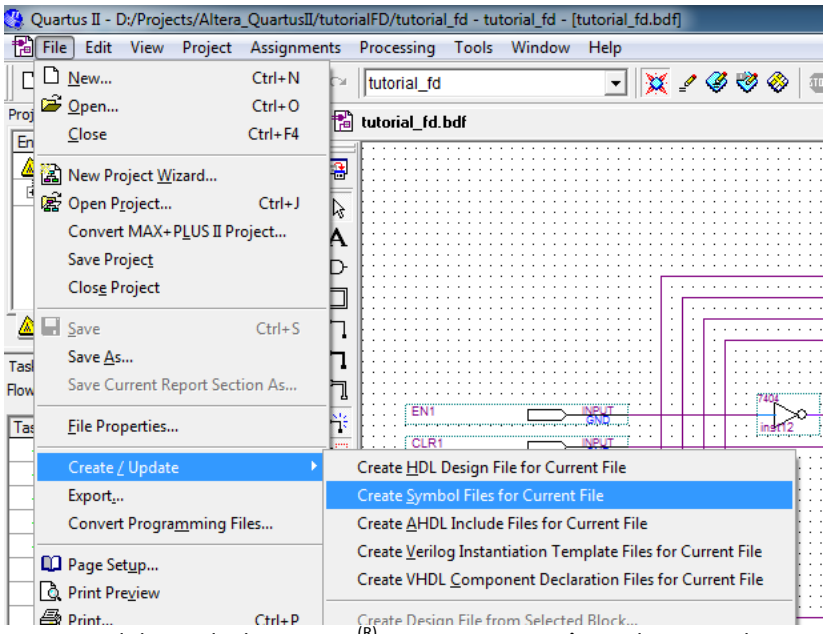


Figura 2.18 – Imagem parcial da janela do Quartus(R) II 9.1 com a sequência de comandos para a transformação de um circuito digital em um componente.

A figura 2.19 mostra a caixa de diálogo para criação de diretório, nomeação do arquivo contendo o componente do circuito digital e gravação do mesmo em disco. O arquivo contendo o componente pode ter qualquer nome e ser gravado no diretório que o projetista desejar.

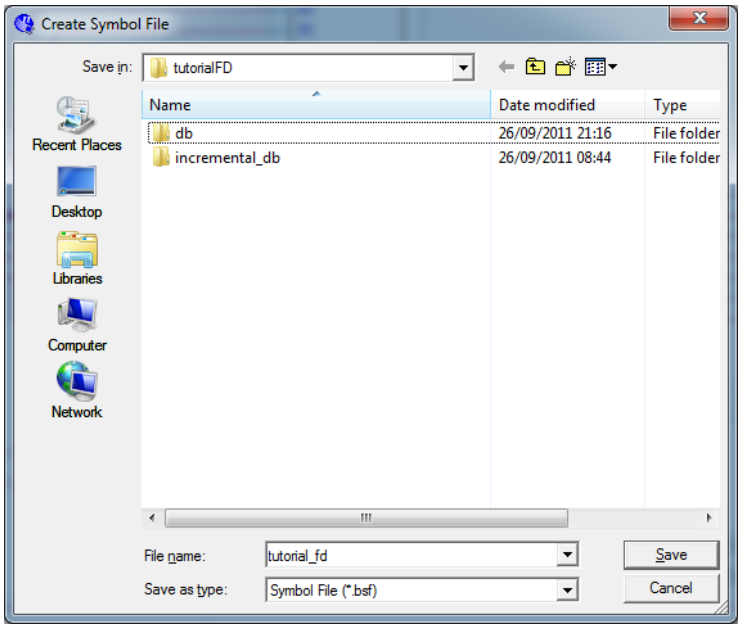


Figura 2.19 – Note que o componente será gravado no próprio diretório do projeto e com o mesmo nome, mas pode ter qualquer nome e ser salvo onde o projetista desejar.

A figura 2.20 mostra a caixa de diálogo de sucesso na geração do componente.

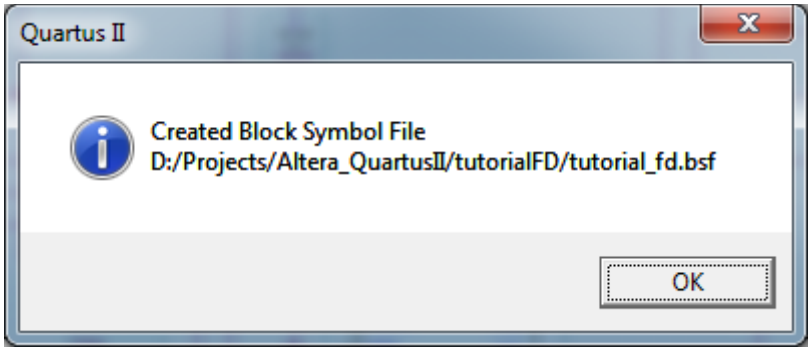


Figura 2.20 – Caixa de diálogo de sucesso na geração do componente.

A partir deste ponto o componente gerado está pronto para ser utilizado em outros projetos digitais, conforme a necessidade do projetista. Entretanto, para isto precisamos executar alguns procedimentos, de modo a fazer com que o mesmo seja reconhecido pelo Quartus(R) II.

2.4. Projeto da Unidade de Controle

A unidade de controle pode ser projetada de vários modos, sendo que o adotado nesta experiência será o Diagrama ASM (ver Anexo 1).

Analogamente ao projeto do fluxo de dados, o projeto da unidade de controle possui duas etapas principais:

- **descrição e compilação**: consiste em se transformar o diagrama ASM em um arquivo texto, na linguagem AHDL;
- **simulação**: as formas de onda dos sinais de entrada da unidade de controle são descritas e o Quartus(R) II fornece as formas de onda dos sinais de saída, seguindo os parâmetros gerados na etapa anterior.

Para o SD proposto, uma das soluções para a unidade de controle pode ser vista na figura 2.21. O diagrama ASM desta figura descreve o algoritmo de geração dos sinais de controle do fluxo de dados, a partir dos sinais de controle externos do sistema digital.

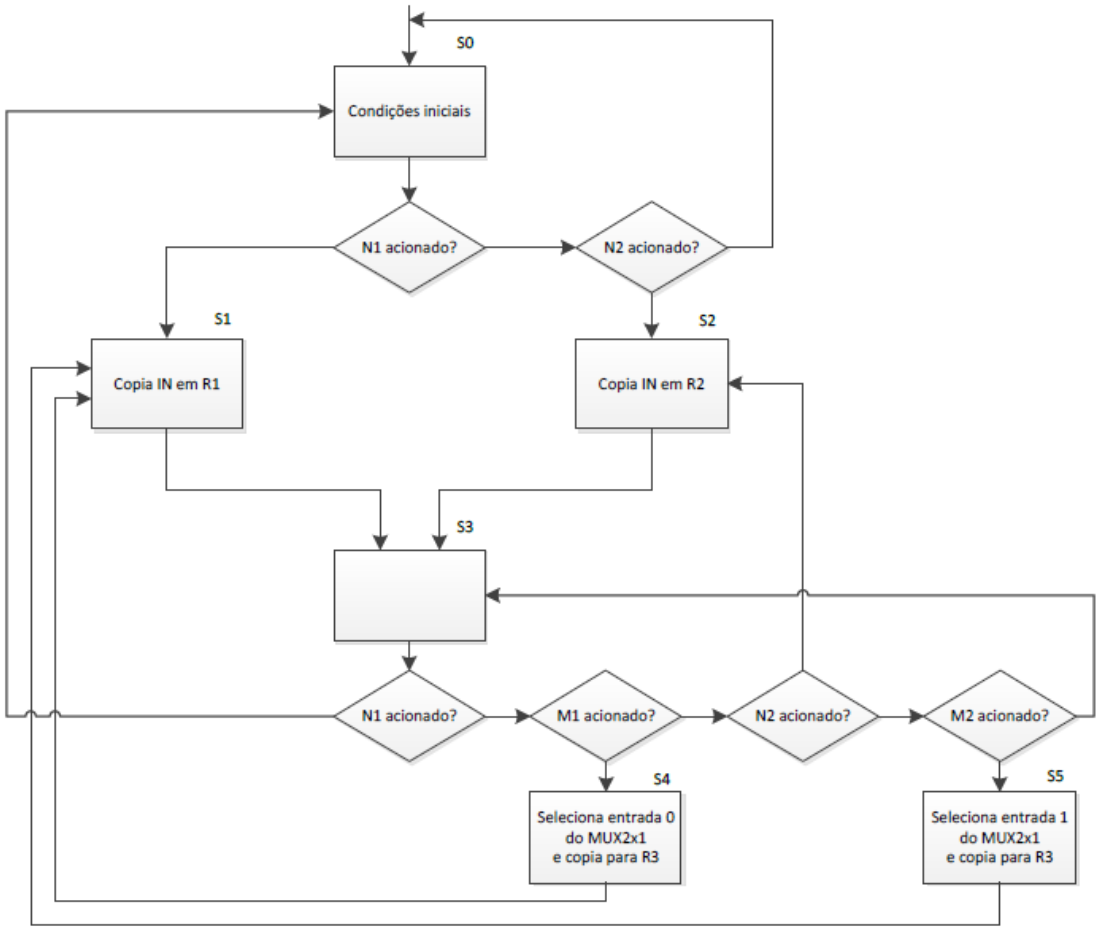


Figura 2.21 – Diagrama ASM de UC do projeto.

a) *Criando o projeto da UC - Unidade de Controle*

Siga as instruções do subitem **b)** do item **2.3.1** para criar o projeto da unidade de controle. Em seguida execute os comandos apresentados neste item.

O projeto da UC será feito na linguagem de descrição AHDL, conforme mostrado nos passos a seguir:

- Criar uma nova área de edição para o arquivo AHDL:
  File -> New... (aparecerá uma caixa de diálogo para a criação de novos recursos para o projeto), vide figura 2.22 abaixo:

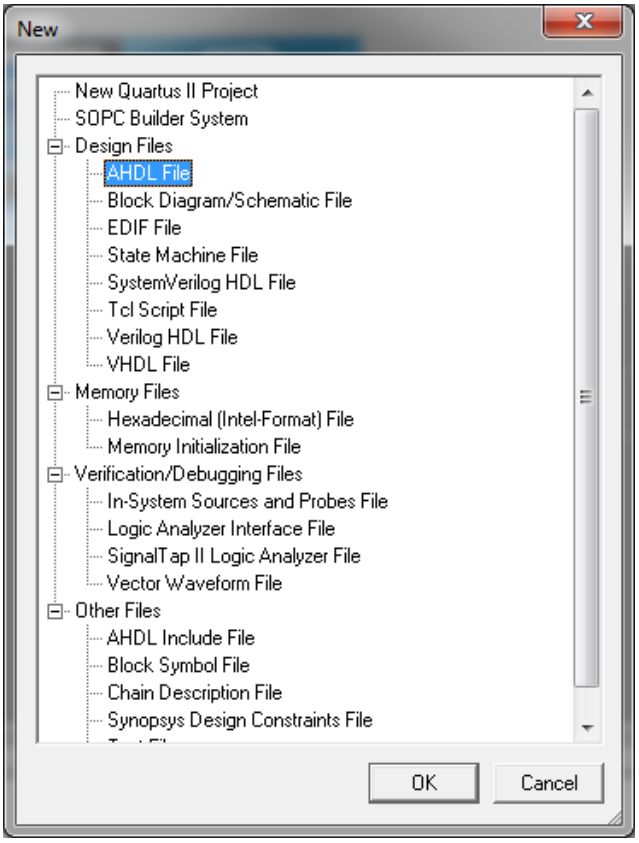


Figura 2.22 – Caixa de diálogo inclusão criação de novos recursos ao projeto.

Nesta janela, selecione a opção hachurada **AHDL File** e pressione (OK) ou simplesmente dê um duplo clique na mesma. Um editor de texto específico será aberto em branco na área de trabalho do Quartus(R) II. Salve-o com o nome **tutorial_uc.tdf**. Para isto utilize a sequência de comandos:

- **File -> Save As... ->** (File name: **tutorial_uc**) e (Save as type: **AHDL file (*.tdf)**) **-> Save**.

2.4.1. Descrição e Compilação da Unidade de Controle

O primeiro passo, a seguir, para se descrever um diagrama ASM para o Quartus(R) II, é a descrição (designação) de cada um dos estados presentes no diagrama.

O Quartus(R) II, através da linguagem AHDL, permite várias alternativas para designar os estados. Nesta experiência adotar-se-á a designação de cada estado pelo valor das saídas da UC, que devem ser geradas em cada estado. Por exemplo, o estado **S0** será caracterizado como sendo aquele em que: CLR1=1, CLR2=1, CLR3=1, EN1=0, EN2=0, EN3=0 e SEL=0.

Uma vez caracterizados os estados, é necessário descrever as transições entre os mesmos, ou seja, descrever o diagrama ASM através de uma tabela de transições. Também há alternativas, em função da estratégia adotada na construção do ASM (MEALY ou MOORE). Nessa experiência, adotou-se a solução MOORE.

Tanto os estados, quanto a tabela de transições, são descritos através da linguagem AHDL, usando o editor de script de código AHDL próprio do Quartus(R) II já descrito no item anterior.

As construções desta linguagem formam as seções do arquivo texto, a saber:

a) denominação do projeto

Nesta seção, iniciada pela palavra **SUBDESIGN,** é especificado o nome do projeto, que deverá ser o mesmo declarado no início da descrição da UC;

b) declaração de entradas e saídas

Nesta seção, delimitada por parênteses, são declarados todos os sinais de entrada/saída do diagrama. As declarações de entradas são finalizadas por **:INPUT;** e as de saída por **:OUTPUT;**;

c) denominação dos estados

Nesta seção, iniciada pela palavra **VARIABLE,** os estados do diagrama ASM são definidos em função das saídas da UC. Primeiramente deve ser descrita a ordem dos bits que definirão cada estado. Isto é feito através da seguinte sentença:

UC: MACHINE OF BITS (.....) WITH STATES

Entre parênteses devem ser colocados os nomes das saídas do diagrama. Por exemplo, para o diagrama da figura 2.23:

(CLR1,CLR2,CLR3,EN1, EN2, EN3,SEL)

Em seguida, cada um dos estados deve ser descrito em função das saídas do diagrama, de acordo com a ordem estabelecida na construção anterior. Isto é feito da seguinte maneira:

(*nome_do_estado* = B'*valores_das_saídas*',);

Para o diagrama da figura 2.23, o estado **S0** seria descrito como **S0=B'1110000'**.

d) declaração da máquina de estados

Nesta seção, delimitada pelas palavras **BEGIN** e **END;** são descritos os dois sinais de controle da máquina de estados que representa o diagrama ASM (CLK e RES) e a tabela de transições entre os estados. Esta tabela é feita baseada nos caminhos existentes entre cada um dos estados do mesmo, em função das entradas a eles associadas (caso existam), e deve ser delimitada pelas palavras **TABLE** e **END TABLE;**. Por exemplo, na figura 2.23, para que a UC passe do estado S0 para o estado S1, é necessário que N1=1 e N2=M1=M2=X(0 ou 1). Portanto, a linha que representa este caminho é:

S0, 1,X,X,X => S1;.

Na figura 2.20 apresenta-se a descrição da UC, em AHDL. Os comandos para a compilação deste código são os mesmos usados na compilação do projeto tutorial_fd. Reveja esses procedimentos, se necessário.

```
SUBDESIGN tutorial_uc
(
      CLKUC: INPUT;
      RES: INPUT;
      N1,N2,M1,M2: INPUT;
      CLR1,CLR2,CLR3,EN1,EN2,EN3,SEL: OUTPUT;
      E1,E2,E3: OUTPUT;
)

VARIABLE
UC: MACHINE OF BITS (CLR1,CLR2,CLR3,EN1,EN2,EN3,SEL,E3,E2,E1)
      WITH STATES
      % Estado    Saidas            %
      % atual                       %
      (
      s0=         B"1110000000",
      s1=         B"0001000001",
      s2=         B"0000100010",
      s3=         B"0000000011",
      s4=         B"0000010100",
      s5=         B"0000001101",
      s6=         B"0000011110"
      );

BEGIN
      UC.CLK = CLKUC;
      UC.RESET = RES;

      TABLE
      %     Estado      Entradas          =>          Proximo %
      %     atual                                     estado  %
            UC,         N1,N2,M1,M2       =>          UC;
            s0,         0, 0, X, X        =>          s0;
            s0,         1, X, X, X        =>          s1;
            s0,         0, 1, X, X        =>          s2;
            s1,         X, X, X, X        =>          s3;
            s2,         X, X, X, X        =>          s3;
            s3,         0, 0, 0, 0        =>          s3;
            s3,         1, X, X, X        =>          s1;
            s3,         0, 1, X, X        =>          s2;
            s3,         0, 0, 1, X        =>          s4;
            s3,         0, 0, 0, 1        =>          s5;
            s4,         X, X, X, X        =>          s3;
            s5,         X, X, X, X        =>          s6;
            s6,         X, X, X, X        =>          s3;
      END TABLE;
END;
```

Figura 2.23 – Código da AHDL da UC.

e) compilação da UC

Apesar da descrição da UC ser diferente daquela apresentada no FD o processo de sua compilação segue os mesmos passos da compilação do FD. Portanto, basta seguir aqueles procedimentos.

A figura 2.24 mostra o símbolo do componente gerado após a compilação do código AHDL mostrado na figura 2.23 e a execução dos comandos correspondentes a esta tarefa, conforme descrito no projeto do FD.

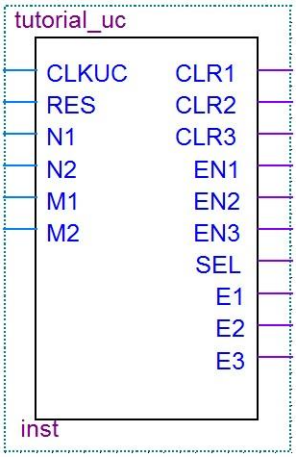


Figura 2.24 – Símbolo gerado para a unidade de controle.

2.4.2. Simulação da Unidade de Controle

A simulação da unidade de controle é feita de maneira análoga à simulação do fluxo de dados. A diferença principal consiste na existência de dois sinais especiais, que são responsáveis pela inicialização e sincronização do diagrama.

Estes sinais são denominados RES e CLKUC. O sinal RES faz com que o diagrama inicie no estado **S0** e o sinal CLKUC faz com que o diagrama passe de um estado a outro, de maneira sincronizada.

Em seguida será mostrado como descrever as formas de onda e os sinais especiais da unidade de controle. A figura 2.25 mostra a carta de tempos da simulação da UC.

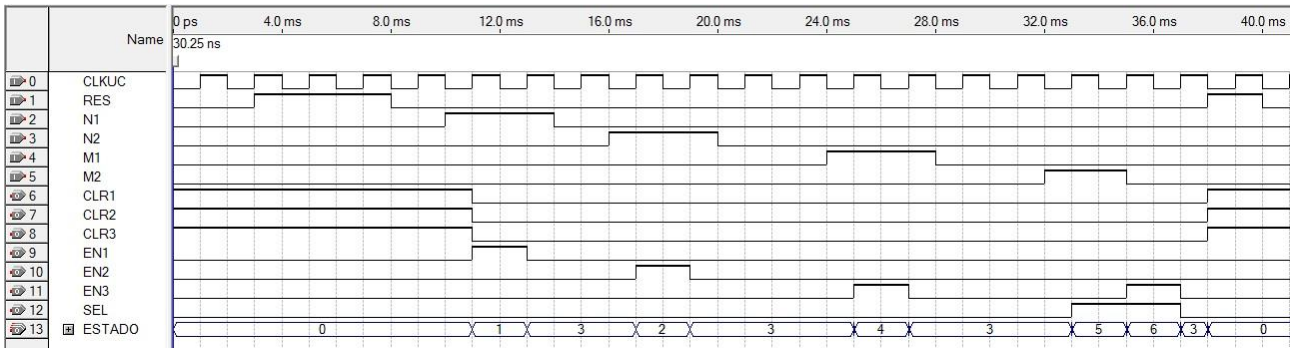


Figura 2.25 – Simulação da unidade de controle do projeto exemplo.

As configurações dos tempos para a simulação da UC são as mesmas aplicadas ao projeto do FD.

2.5. Projeto do Sistema Digital Completo

O projeto do sistema digital completo também é composto pelas etapas já vistas nos projetos do fluxo de dados e da unidade de controle: descrição e síntese; simulação.

Na descrição e síntese do SD completo, basta apenas usar os componentes gerados para cada uma das partes projetadas anteriormente e conectá-los em um diagrama lógico, conforme mostrado nos procedimentos a seguir.

Dessa forma, criaremos um novo projeto **tutorial_sd** para demonstrar o uso dos componentes gerados nos passos anteriores. Para isto seguiremos os passos já apresentados ao longo deste tutorial, entretanto mostraremos somente os passos adicionais relacionados à inclusão dos novos componentes. O primeiro passo desta etapa é incluir os arquivos de descrição de ambos projetos FD e UC, conforme mostrado na figura 2.26.

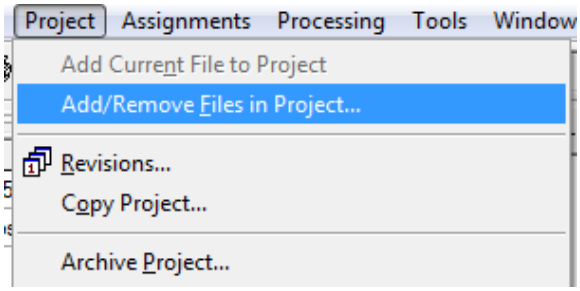


Figura 2.26 – Opção no menu para inclusão dos diagramas lógicos correspondentes aos componentes gerados no Quartus(R) II que serão utilizados no projeto corrente.

A figura 2.27 mostra a caixa de diálogo para a inclusão/remoção de arquivos esquemáticos (circuitos digitais correspondentes aos componentes gerados no Quartus(R) II, usados no projeto).

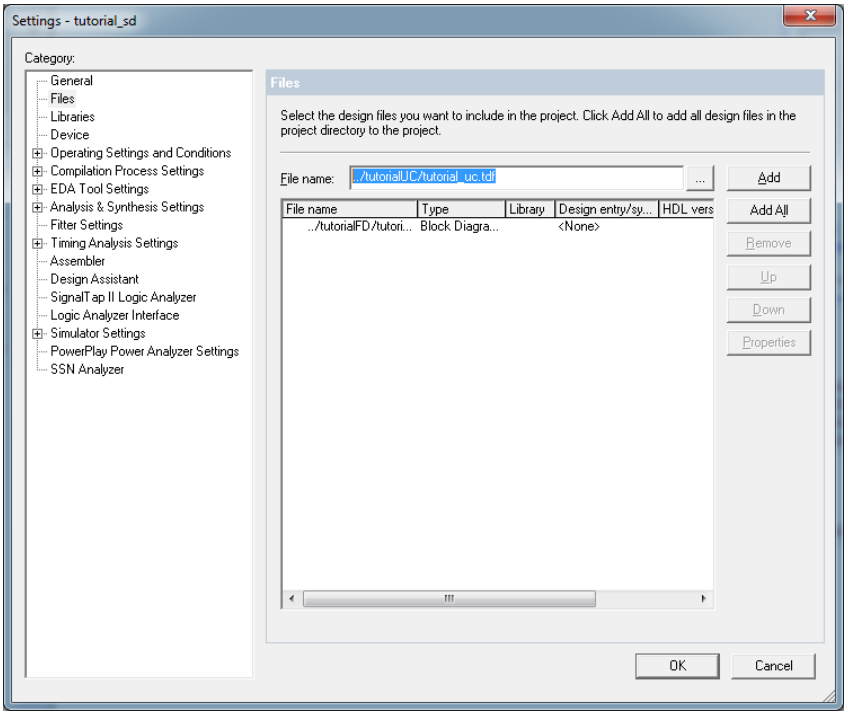


Figura 2.27 – Janela de inclusão/remoção de arquivos esquemáticos representativos dos componentes gerados no Quartus(R) II, a serem usados no novo projeto.

Nesta tela o usuário deve clicar em ( ) e, a partir da caixa de diálogo de seleção de arquivos, selecionar o arquivo de descrição correspondente ao componente gerado no Quartus(R) II desejado e pressionar o botão (Add) para inclui-lo na lista, um de cada vez. Então, pressionar no botão (OK) para finalizar esta etapa do processo. Usaremos dois novos componentes gerados no Quartus(R) II no nosso sistema digital.

A figura 2.28 mostra como selecionar e carregar o componente **tutorial_fd** na folha de desenho do Quartus(R) II para o projeto **tutorial_sd**. Mas, antes siga estes passos: na régua de ferramentas de desenvolvimento pressione em ( ) e, na janela **Symbol** pressione em ( ) para abrir a caixa de diálogo de seleção de arquivos, na qual o projetista deverá abrir o diretório onde se encontram os componentes criados (geralmente é um diretório com nome sugestivo como myLibrary ou minhaBiblioteca ou meusComponentes...) ou o próprio diretório onde o componente foi criado originalmente e, selecionar o arquivo com o nome do componente desejado, então pressionar (Open), em seguida pressionar o botão (OK) na caixa de diálogo da figura 2.27.

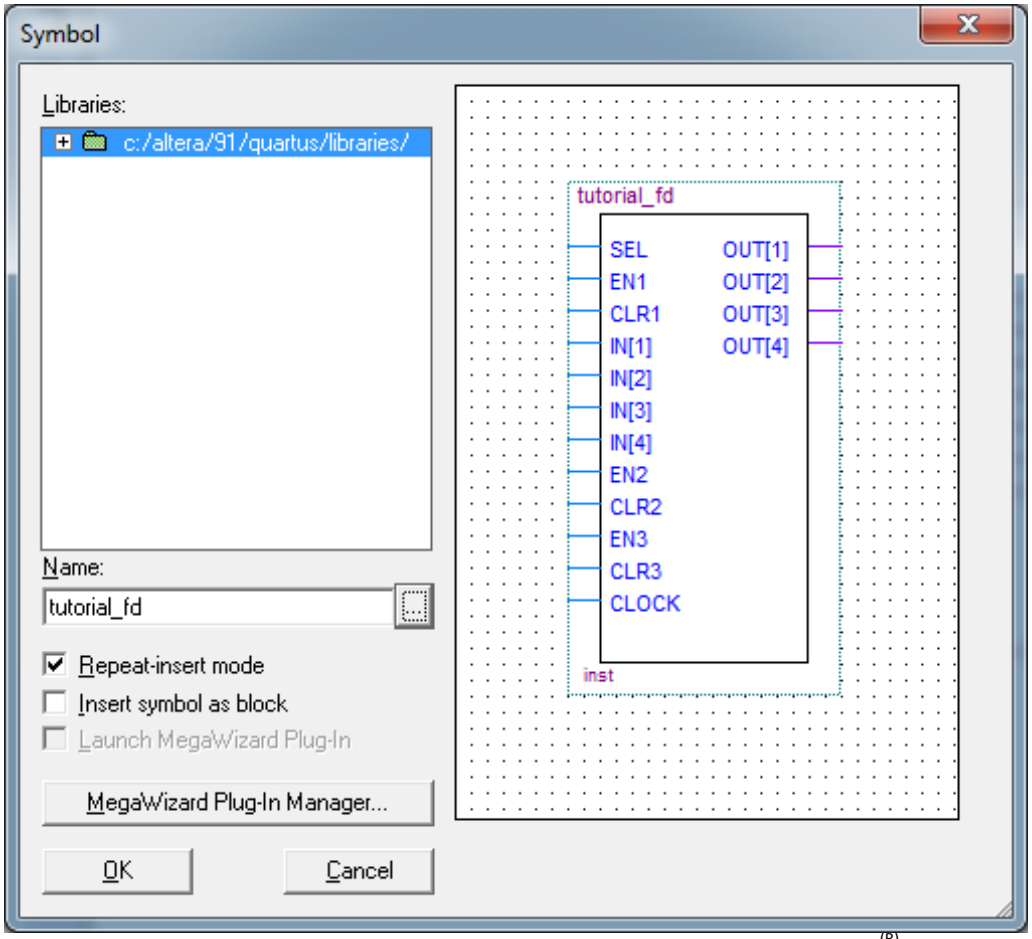


Figura 2.28 – Carregando o componente **tutorial_fd** gerado no Quartus(R) II.

A figura 2.29 mostra a seleção e inclusão do componente **tutorial_uc** no projeto **tutorial_sd**.

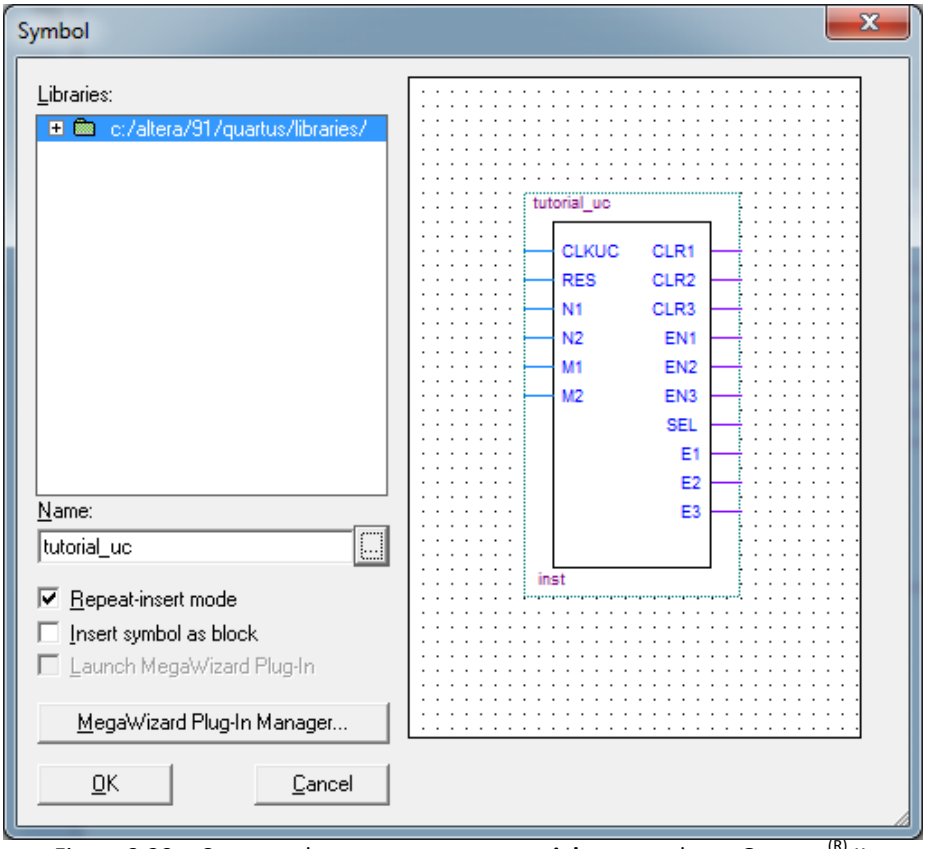


Figura 2.29 – Carregando o componente **tutorial_uc** gerado no Quartus(R) II.

Daqui em diante é só seguir os passos que faltam, conforme apresentado anteriormente neste tutorial e concluir o projeto do nosso sistema digital completo, conforme mostrado na figura 2.30.

Para esta etapa do projeto, basta seguir o mesmo procedimento do item 2.3.1. A única diferença será o nome do projeto (**tutorial_sd**) e os símbolos que serão colocados no diagrama elétrico, que são apenas os pinos de entrada e saída, mais os símbolos gerados nos itens anteriores.

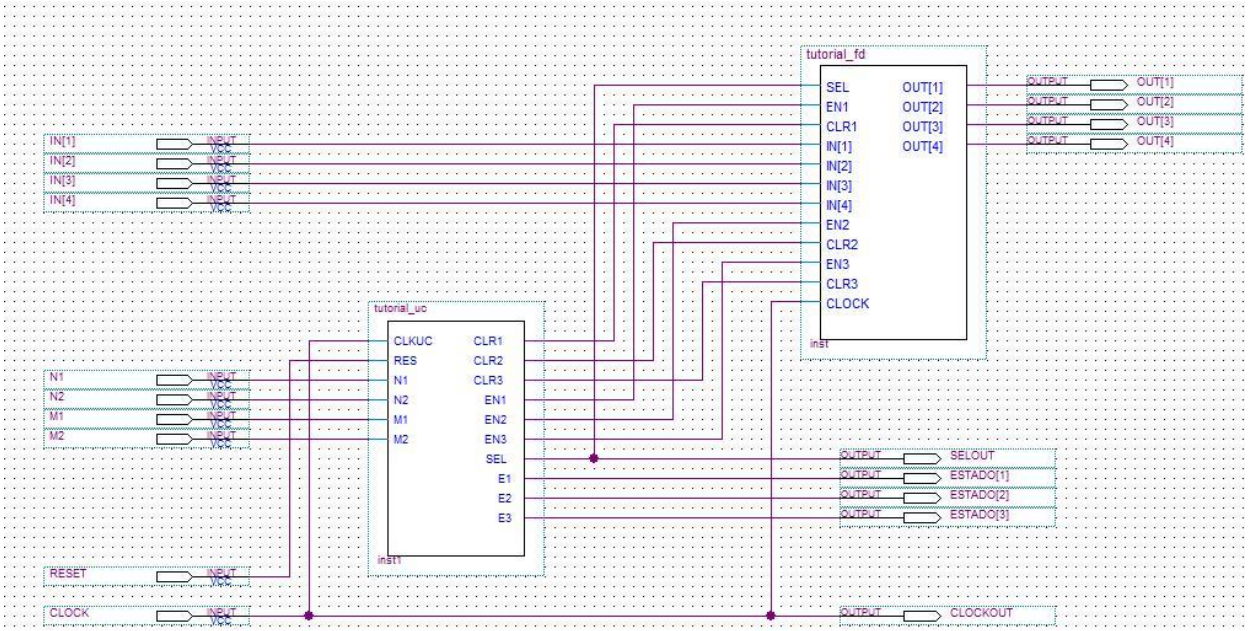


Figura 2.30 – Diagrama do SD completo.

A compilação deste projeto segue os procedimentos já apresentados em etapas anteriores.

A simulação do SD completo deverá ser feita de acordo com os procedimentos aplicados na simulação do projeto do FD, com a ressalva de que os sinais que estarão disponíveis agora serão os sinais externos do SD, mais o RES e o CLKSD.

2.6. Gravação do Sistema Digital Completo em uma EPLD

Para a gravação do SD na EPLD que se encontra na placa Altera DE2, é necessário conectar uma das extremidades do cabo especial de gravação ao conector USB do PC, e a outra extremidade ao conector USB-Blaster da placa. Feito isto, basta seguir o procedimento abaixo.

NOTA: Antes da designação da pinagem o projeto deve ser compilado.

a) designação da pinagem do FPGA da placa DE2 para o SD Completo

Antes da programação da placa DE2, precisamos designar a pinagem do seu FPGA com os sinais do projeto SD. Para isto, execute os comandos que se seguem e acompanhe pela figura 2.31:

- **Assignments -> Pins**

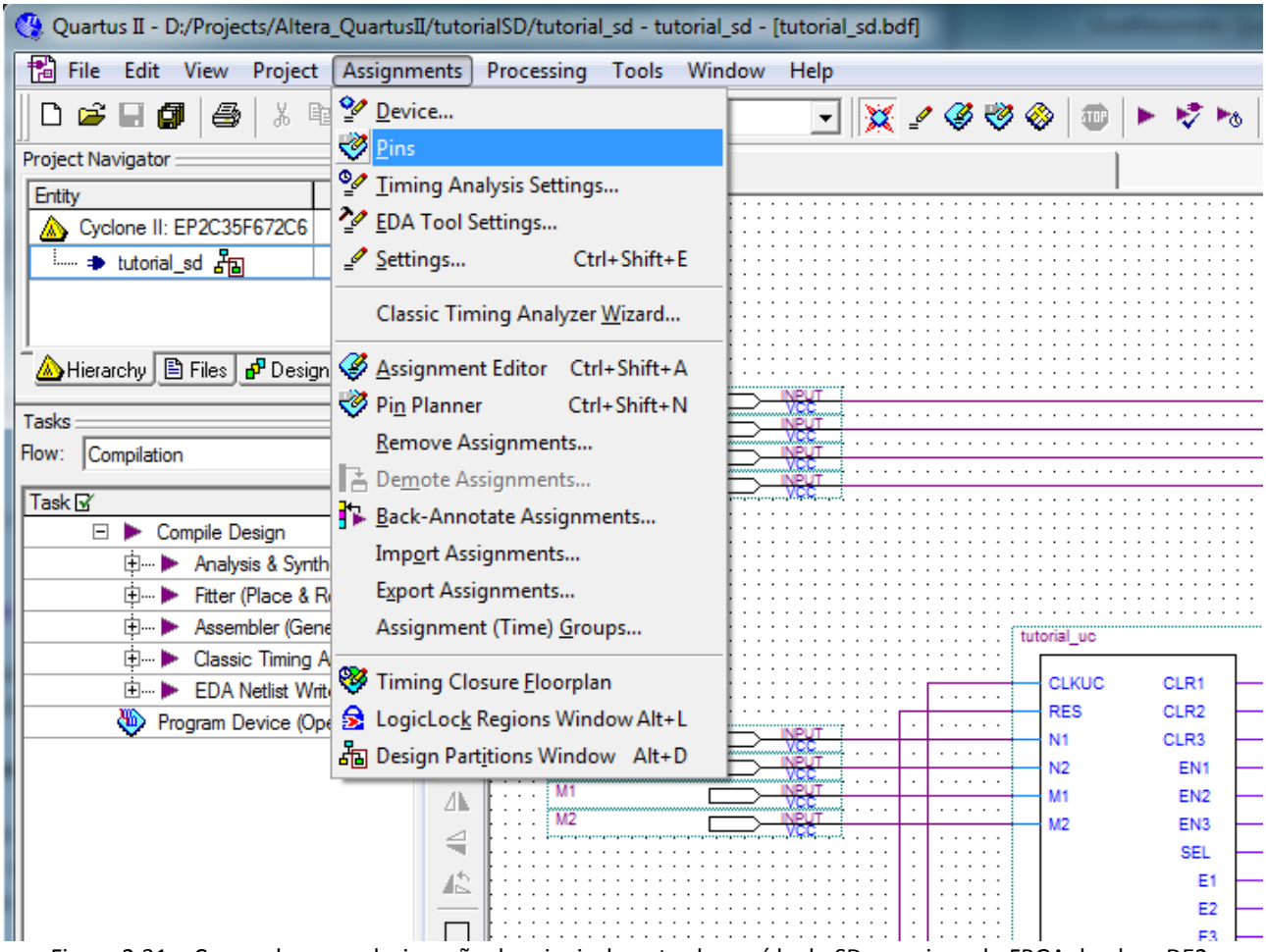


Figura 2.31 – Comandos para designação dos sinais de entrada e saída do SD nos pinos do FPGA da placa DE2.

Após a execução destes comandos aparecerá uma janela de edição dos pinos no FPGA, conforme mostrado na figura 2.32, a seguir.

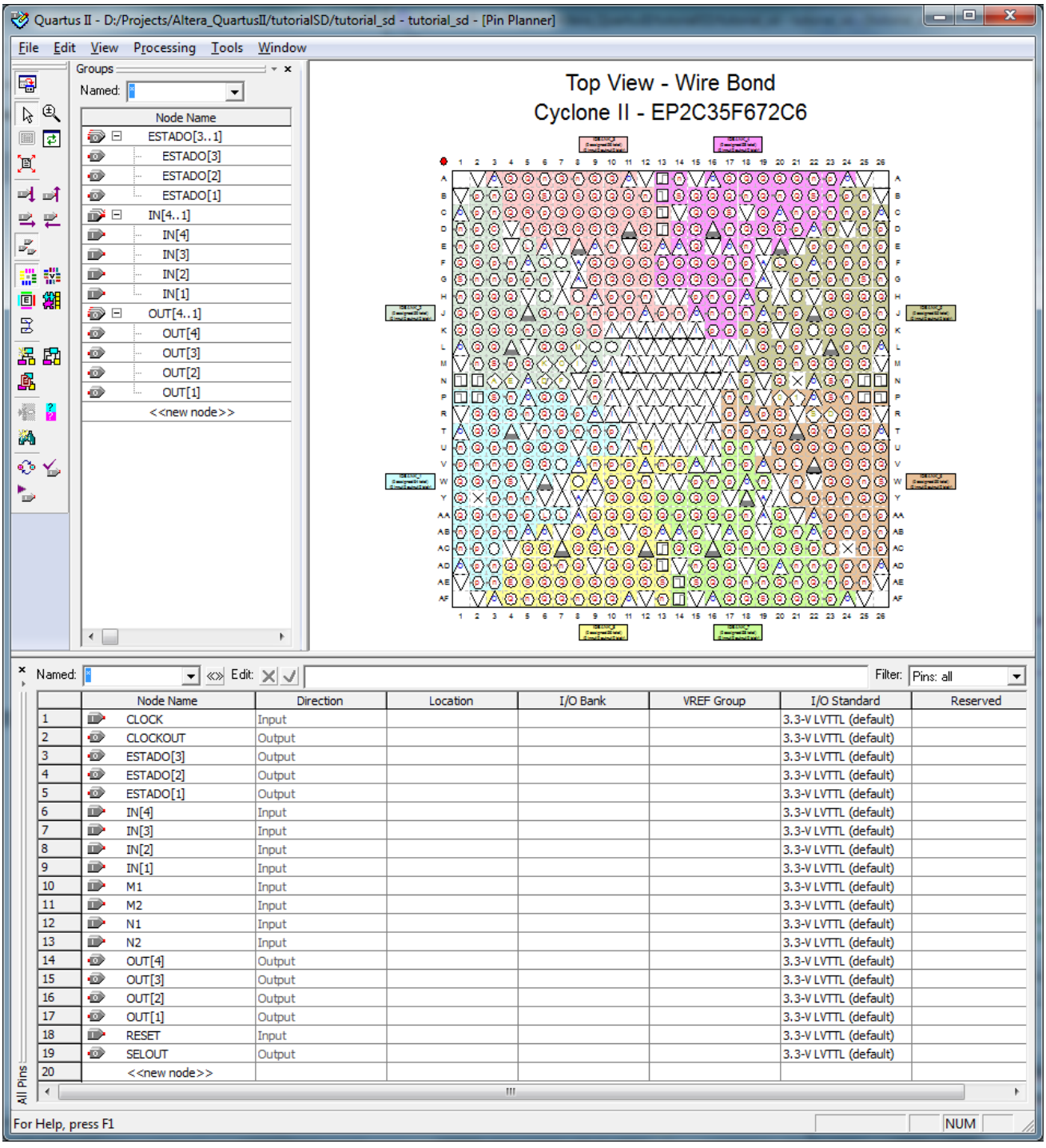


Figura 2.32 – Janela de designação da pinagem do SD no FPGA.

Vamos designar alguns componentes (chaves, botões e LEDs...) da DE2 aos pinos correspondentes no mapa de pinagem (figura 2.32) do FPGA, vide tabela 1 abaixo:

NOTA: A relação entre os pinos do FPGA e os componentes (chaves, botões e LEDs...) de acesso da placa DE2 encontra-se na tabela **Altera DE2 Board Pin Table** no arquivo DE2_Pin_Table.pdf.

Tabela 1 – Designação dos sinais do SD com chaves, botões e LEDs da placa DE2.

Nome do Sinal	Sentido do Sinal	Placa DE2
CLOCK	Entrada	CLOCK_50 (On Board 50 MHz)
CLOCKOUT	Saída	LED Red[16]*
ESTADO[3]	Saída	LED Red[2]
ESTADO[2]	Saída	LED Red[1]
ESTADO[1]	Saída	LED Red[0]
IN[4]	Entrada	SW[3]
IN[3]	Entrada	SW[2]
IN[2]	Entrada	SW[1]
IN[1]	Entrada	SW[0]
M2	Entrada	KEY[3]
M1	Entrada	KEY[2]
N2	Entrada	KEY[1]
N1	Entrada	KEY[0]
OUT[4]	Saída	LED Green[3]
OUT[3]	Saída	LED Green[2]
OUT[2]	Saída	LED Green[1]
OUT[1]	Saída	LED Green[0]
RESET	Entrada	SW[17]
SELOUT	Saída	LED Red[17]*

* - Estes sinais foram ligados nos últimos LEDs vermelhos apenas por conveniência.

Após a designação da pinagem teremos a imagem da figura 2.32 com o aspecto da figura 2.33.

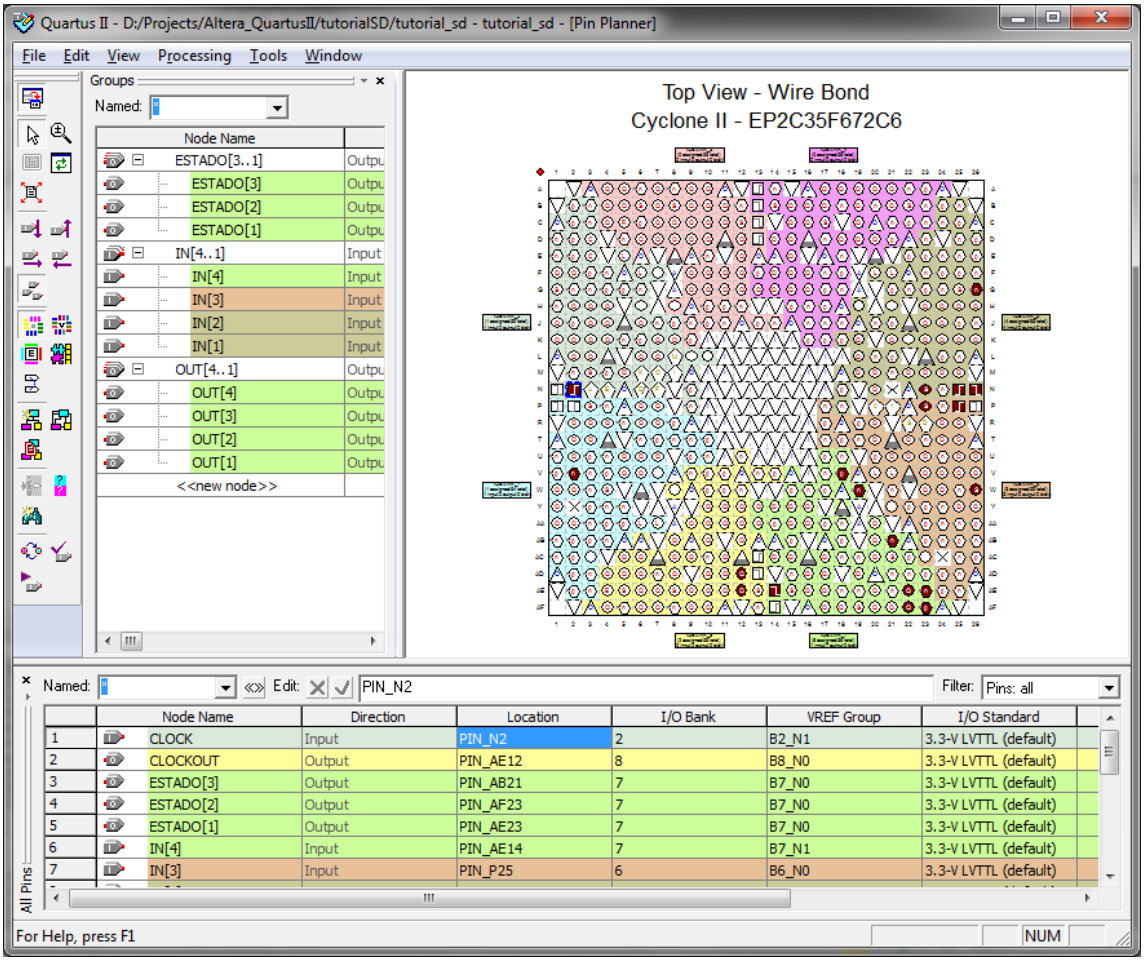


Figura 2.33 – Mapa da designação dos pinos do FPGA com os sinais do SD e os componentes da placa DE2.

Volte à janela principal do Quartus(R) II e observe que os nomes dos pinos designados foram acrescentados ao esquema elétrico do sistema digital do projeto, conforme mostrado na figura 2.34.

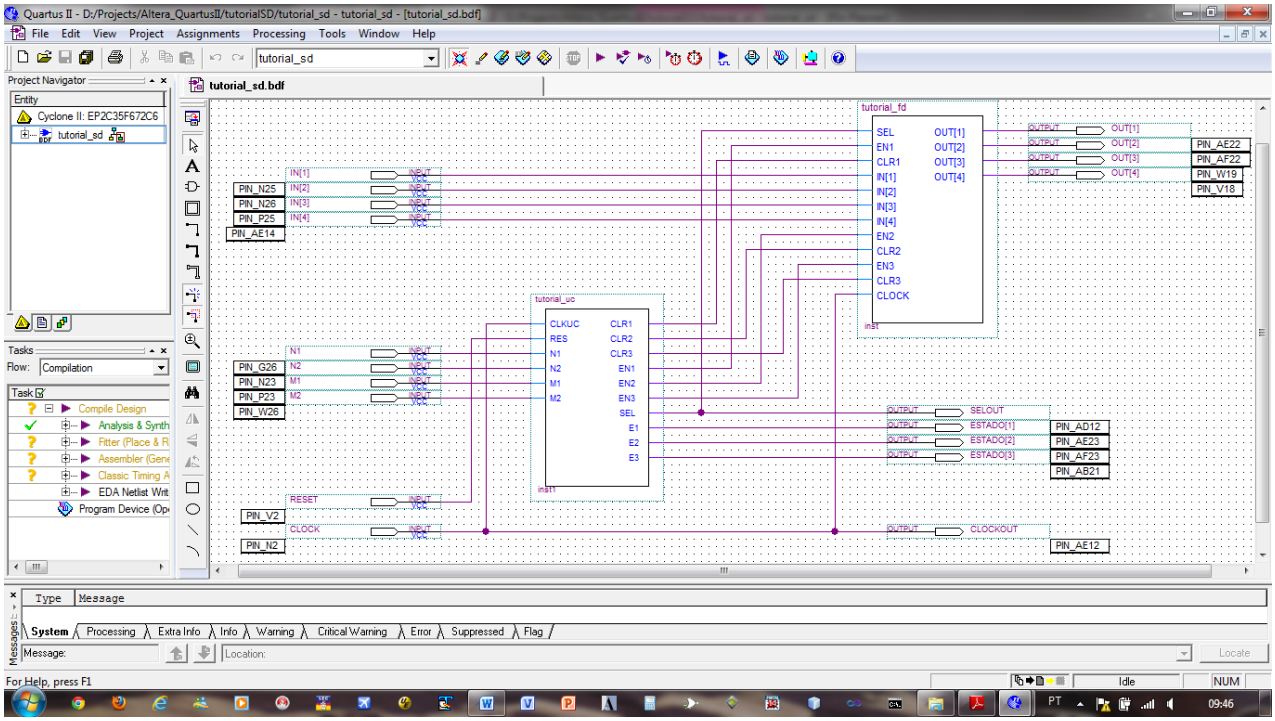


Figura 2.34 – Esquema elétrico do SD depois da designação dos pinos.

Então, recompile o projeto com os pinos designados para que estas configurações façam efeito na etapa de programação da placa de desenvolvimento Altera DE2.

a) programando a placa DE2 com o código objeto do SD gerado com o Quartus(R) II

Esta etapa consiste na gravação do nosso projeto no FPGA da placa DE2, para testes em tempo real. Para isto basta executar os comandos a seguir:

- **Tools -> Programmer**

A figura 2.35 mostra a sequência de comandos anterior:

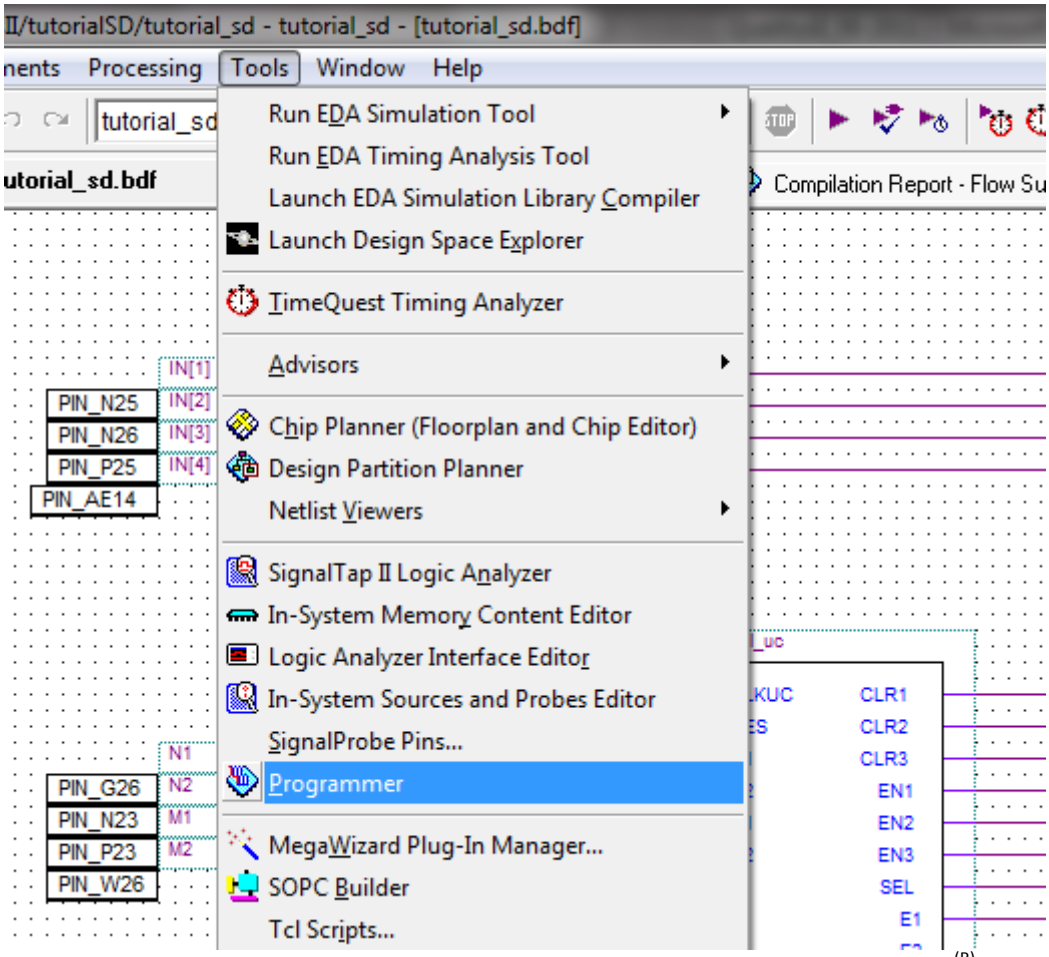


Figura 2.35 – Sequência de comandos para a programação da placa DE2 com o Quartus(R) II 9.1.

Então aparecerá a tela mostrada na figura 2.36:

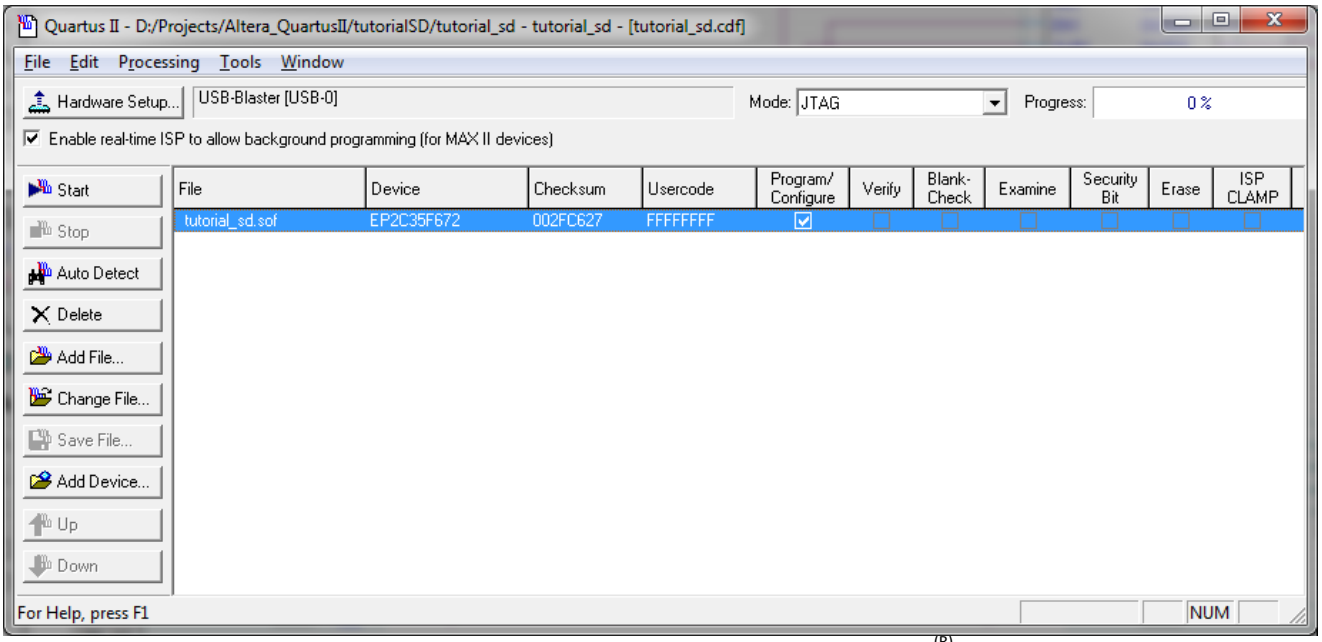


Figura 2.36 – Tela de programação da placa DE2 do Quartus(R) II 9.1.

Na figura 2.36, não há necessidade de selecionar o projeto a ser gravado na placa de desenvolvimento, pois existe somente um. Note que o nome do hardware configurado (USB-Blaster [USB-0]) aparece à direita do botão (

), isto porque a placa de DE2 já se encontra conectada ao computador.

Ligue a placa DE2, antes de iniciar a sua programação, pressionando a chave de pressão vermelha, localizada próximo ao conector de alimentação da mesma.

Para efetuar a programação da placa de desenvolvimento basta pressionar o botão (

) e o código objeto do SD **tutorial_sd.sof** será transferido para o FPGA da mesma.

Após o envio do código objeto para a placa DE2, o resultado de sucesso aparece como na figura 2.36, com a barra de progresso da programação **Progress** dessa forma (

), vide figura 2.37.

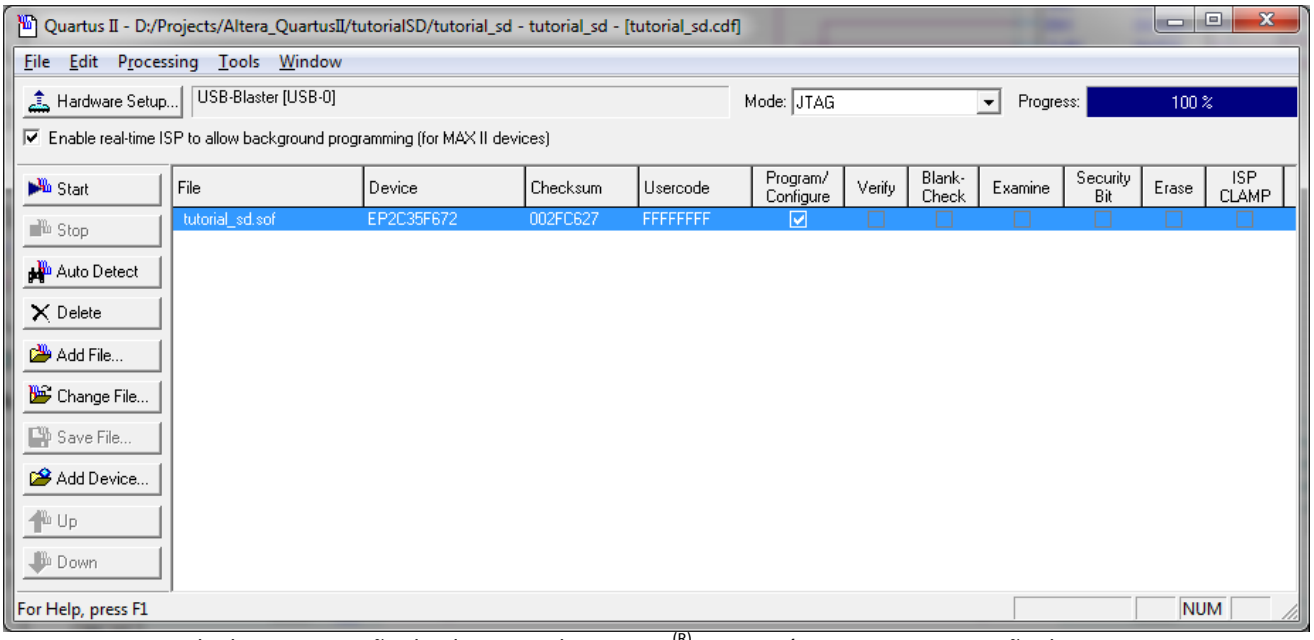


Figura 2.37 – Tela de programação da placa DE2 do Quartus(R) II 9.1 após sucesso na gravação do projeto na mesma.

2.7. Testes do SD na placa Altera DE2

Após a programação da placa DE2 basta acionarmos os botões, ligarmos as chaves e observarmos os resultados nos LEDs e, com isto comparar os resultados obtidos com aqueles esperados para o projeto SD.

3. BIBLIOGRAFIA

[Chu, 2006] CHU, P.P. **RTL Hardware Design Using VHDL: coding for efficiency, portability, and scalability**. Wiley, 2006.
[de Micheli, 1994] DE MICHELI, G. **Synthesis and Optimization of Digital Circuits**. McGraw-Hill, 1994.
[Flynn & Luk, 2011] FLYNN, M.J. & LUK, W. **Computer System Design: System-on-Chip**, Wiley, 2011.
[Fregni & Saraiva, 1995] FREGNI, E. & SARAIVA, A.M. **Engenharia do Projeto Lógico Digital: conceitos e prática**. Edgard Blücher, 1995.
[Givone, 2003] GIVONE, D.D. **Digital Principles and Design**. McGraw-Hill, 2003. (Capítulo 8)
[Harel, 1987] HAREL, D. *Statecharts: a Visual Formalism for Complex Systems*. **Science of Computer Programming**, vol. 8, pp. 231-274, 1987.
[Harris & Harris, 2007] HARRIS, D.M. & HARRIS, S.L. **Digital Design and Computer Architecture**. Morgan Kaufmann, 2007.
[Lala, 2007] LALA, P.K. **Principles of Modern Digital Design**. Wiley, 2007.
[Morris & Miller, 1978] MORRIS, R.L. & MILLER, J.R. **Projeto com Circuitos Integrados TTL**. Editora Guanabara Dois, 1978.
[Patterson & Hennessy, 2009] PATTERSON, D.A. & HENNESSY, J.L. **Computer Organization and Design: the hardware/software interface**. 4th edition, Morgan Kaufmann, 2009.
[Wakerly, 2006] WAKERLY, J.F. **Digital Design: Principles and Practices**. 4th edition, Prentice-Hall, 2006.

ANEXO 1

RESUMO - DIAGRAMA ASM

(VERSÃO PRELIMINAR)

ASM – Algorithmic State Machine

Introdução

ASM é um fluxograma através do qual se representa a sequência de ações que a unidade de controle de um sistema digital deve realizar, para se obter o comportamento especificado.

As ações realizadas dependem das entradas externas do sistema digital e também de condições que traduzem a situação em que se encontram a própria unidade de controle e o fluxo de dados. Na verdade, ASM é uma representação gráfica do algoritmo que descreve o comportamento do sistema digital, OU seja, é uma ferramenta para descrever a "máquina de estados" de forma mais completa do que os diagramas de estados apresentados na disciplina PCS-214 (naquela disciplina, os circuitos sequenciais, que são a realização física das máquinas de estado, eram muito simples e, geralmente, possuíam apenas uma entrada).

Um fluxograma ASM parece semelhante aos fluxogramas convencionais, mas deve ser interpretado de outra maneira.

Nos fluxogramas convencionais há apenas a descrição dos passos a serem seguidos e as decisões a serem tomadas, sem nenhuma relação com a variável tempo. Já nos diagramas ASM, além da descrição da sequência dos eventos há as relações temporais entre os estados da unidade de controle e as ações que ocorrem, em cada estado, em resposta às bordas do CLOCK.

Uma metodologia alternativa para o Diagrama ASM, para representar "máquinas de estados", são as Redes de Petri, assunto que será apresentado na disciplina "Organização de Sistemas Digitais".

O Diagrama ASM

O diagrama ASM contém três elementos básicos:

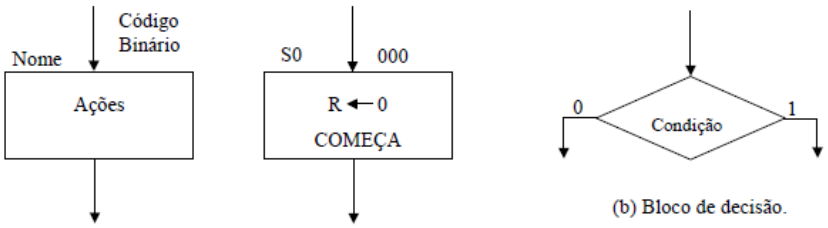
- o bloco de estado;
- o bloco de decisão;
- o bloco de saída condicional.

Na figura A.1 estes três blocos são mostrados.

Descrição dos blocos:

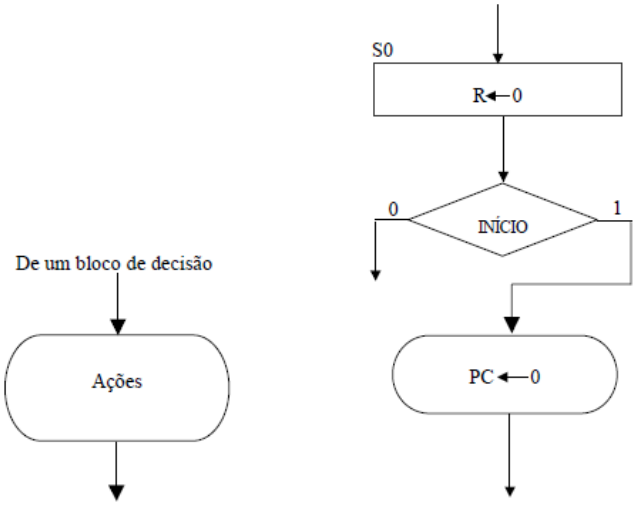
Bloco do estado: o nome do estado é colocado externamente ao bloco e, dentro do mesmo, aparecem as ações a serem tomadas. No exemplo da figura A.1 (a), no estado S0 o registrador R deve ser limpo sincronamente com qualquer clock que ocorra enquanto em S0, e a variável COMEÇA deve assumir o valor 1, enquanto em S0. Observe que a variável COMEÇA deve assumir o valor Zero em todos os estados onde ela não aparece dentro do bloco.

Bloco de decisão: representa o efeito das entradas, na sequência de controle. A condição mostrada na figura A.1 (b) tanto pode ser expressa por uma única variável como por uma expressão booleana. Os dois caminhos referem-se aos 2 possíveis valores que a condição pode assumir.



(a) Bloco de estado com exemplo.

(b) Bloco de decisão.



(c) Bloco de saída condicional com exemplo.

Figura A.1 – Elementos do ASM.

Bloco de Saída Condicional: este bloco é característico dos fluxogramas ASM e não existe um equivalente nos fluxogramas convencionais. A entrada de um bloco de saída condicional sempre deve se originar numa das saídas de um bloco de decisão. Na figura A.1 (c) apresenta-se um exemplo. Se o ASM está no estado S0, a cada pulso de Clock o registrador R é limpo. Já o registrador PC também é limpo no estado S0, mas apenas se o sinal INÍCIO for igual a 1.

Bloco ASM: é o conjunto construído com um bloco de estado e todos os blocos de decisão e de saídas condicionais que ficam entre a saída do bloco de estado e a entrada do mesmo bloco ou de um outro bloco de estado. Ver figura A.2.

A cada borda de subida, as condições dos blocos de decisão são examinadas e, dependendo do seu valor (0 ou 1), seguisse para o estado seguinte indicado. No exemplo da figura A2, enquanto o sinal INICIO = 0, o diagrama fica no estado S0.

Se, num instante qualquer, o sinal INICIO = 1, na primeira borda de subida do CLOCK o ASM mudará de estado. Nessa mesma borda também é examinado o sinal Q0, através do qual decide-se se o estado seguinte é S1 ou S2 .

Independentemente de ser S1 ou S2 , ao mudar de estado, o sinal AVAIL vai para ZERO.

Observar que a saída condicional só ocorre se INICIO = 1.

Pode-se comparar os diagramas ASM, com os diagramas de estado tipo MEALY e tipo MOORE. Quando se usam os blocos de saída condicional, tudo se passa como se a solução adotada fosse do tipo MEALY. Se a opção for MOORE, os blocos de saída condicional são desnecessários.

O projeto do circuito lógico descrito por um diagrama ASM é facilitado quando todos os módulos sequenciais (flipflops, contadores, registradores, etc) forem do tipo “com enable”.

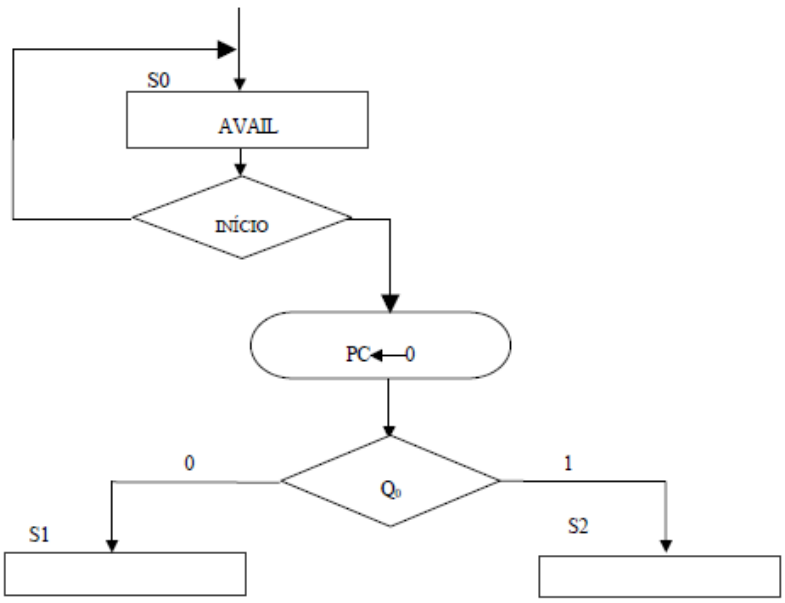


Figura A.2 – Bloco ASM.

ANEXO 2

GLOSSÁRIO

AHDL	**A**ltera **H**ardware **D**escription **L**anguage.
ASIC	**A**pplication **S**pecific **I**ntegrated **C**ircuit.
CPLD	**C**omplex **P**rogrammable **L**ogic **D**evice.
EDIF	**E**letronic **D**esign **I**nterchange **F**ormat (Formato padrão industrial para transferência de arquivos de projetos de circuitos).
EDA	**E**lectronic **D**esign **A**utomation.
EPLD	**E**rasable **P**rogrammable **L**ogic **D**evice.
FPGA	**F**ield **P**rogrammable **G**ate **A**rray.
FPLD	**F**ield **P**rogrammable **L**ogic **D**evice.
GAL	**G**eneric **A**rray **L**ogic.
HDL	**H**ardware **D**escription **L**anguage.
JEDEC	**J**oint **E**lectron **D**evice **E**ngineering **C**ouncil (Formato padrão para transferência de informações entre o sistema onde o projeto foi desenvolvido e o programador do dispositivo.)
MPGA	**M**ask **P**rogrammable **G**ate **A**rray.
MPLD	**M**ask **P**rogrammable **L**ogic **D**evice.
PAL	**P**rogrammable **A**rray **L**ogic.
PLA	**P**rogrammable **L**ogic **A**rray.
PLD	**P**rogrammable **L**ogic **D**evice.
PLS	**P**rogrammable **L**ogic **S**equencer.
PSA	**P**rogrammable **S**equencial **A**rray.
RTL	**R**egister **T**ransfer **L**evel.
VERILOG	Linguagem HDL da empresa Verilog.
VHDL	**V**ery High Speed Integrated Circuits **H**ardware **D**escription **L**anguage.
VHSIC	**V**ery **H**igh **S**peed **I**ntegrated **C**ircuits.