

1 Introdução à Modelagem e Simulação

1.1 Modelagem

VHDL é uma linguagem de descrição de hardware na qual as atribuições do hardware são executadas na sequência em que estão declaradas (Figura 1.1).

Há dois tipos básicos de declarações:

Seqüencial - são declarações executadas uma após a outra, como na programação em linguagens formais (C, Pascal, etc.) e as declarações anteriores são ignoradas após sua execução;

Concorrente - são declarações continuamente ativas. Portanto, a sua ordem não é relevante. Declarações concorrentes são especialmente adaptadas ao modelo de hardware paralelo.

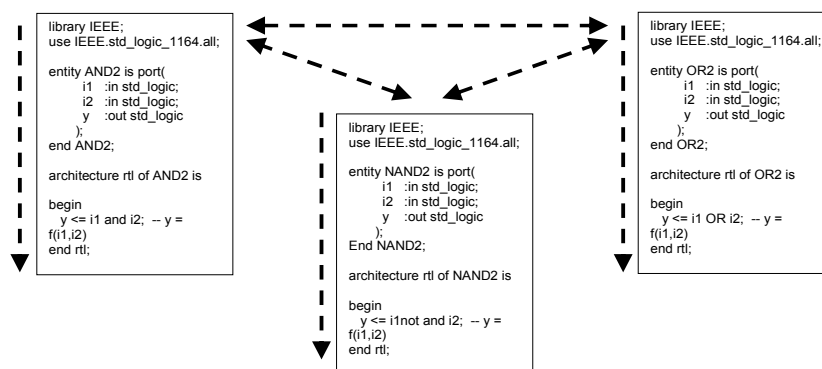


Figura 1.1 - Execução de atribuições em VHDL.

VHDL também utiliza uma metodologia baseada em três importantes características técnicas de modelagem:

Abstração - permite a descrição de diferentes partes de um sistema com diferentes níveis de detalhes. Em módulos utilizados para simulação não há necessidade de serem descritos com o mesmo detalhamento dos módulos que serão sintetizados¹;

Modularidade - permite ao projetista dividir o projeto em blocos funcionais e após descrevê-los como um bloco único contendo vários blocos funcionais interconectados (Figura 1.2);

Hierarquia - permite ao projetista construir módulos individuais e cada um destes pode ser composto de vários sub-módulos. Cada nível da hierarquia pode conter módulos de diferentes níveis abstração. Um sub-módulo em um determinado nível da hierarquia maior pode estar presente em um módulo de nível hierárquico menor.

¹ Síntese é o processo de “tradução” ou compilação de um código VHDL para uma descrição abstrata, em uma linguagem mais próxima da implementação. A síntese lógica é ainda um processo independente da tecnologia. O resultado obtido é uma representação do dispositivo em nível de transferência entre registradores, na qual se definem os registradores, suas entradas e saídas e a lógica combinacional entre eles.

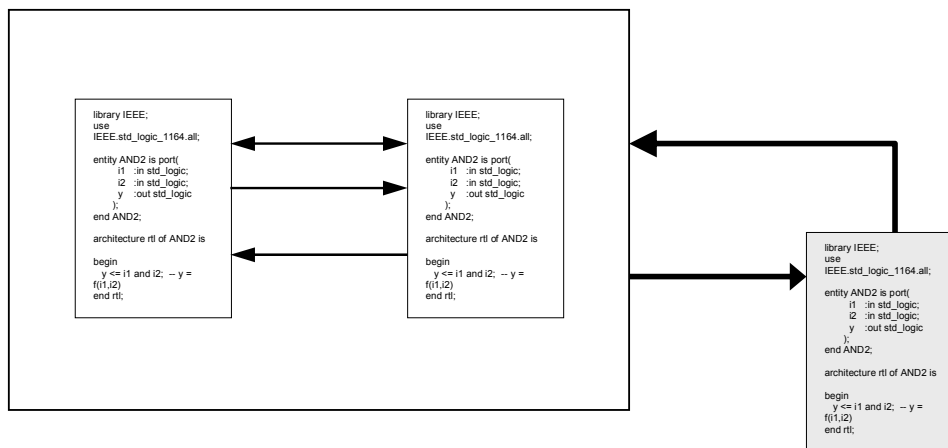


Figura 1.2 - Interconexão de módulos individuais.

Uma descrição em VHDL pode conter diferentes níveis de abstração: comportamental (ou algorítmico); transferência entre registradores; funcional em nível de portas lógicas com atraso unitário ou funcional em nível de portas lógicas com atrasos detalhados.

O nível mais alto de abstração é o comportamental (*behavioral*), que permite descrever o comportamento do circuito através de laços (*loops*) e processos. Na descrição comportamental, faz-se uso de texto ou equações para descrever como o dispositivo eletrônico deve se comportar. Neste nível de abstração o circuito é definido na forma de um algoritmo, utilizando construções similares àquelas de linguagens de programação formais.

Já o nível intermediário de abstração possibilita descrever o funcionamento do circuito em termos de lógica combinacional (booleana), englobando a representação do dispositivo em nível de transferência entre registradores (*Register Transfer Level* - RTL), que consiste na utilização de funções lógicas combinacionais e registradores.

No nível mais baixo de abstração (estrutural) faz-se uma representação do circuito semelhante a uma lista, descrevendo a rede de portas lógicas e suas interconexões. Neste nível de abstração o circuito é descrito mais próximo da implementação real, podendo ser definidas portas lógicas com atrasos unitários ou com atrasos detalhados.

1.2 Fluxo de projeto em VHDL

Seguir um fluxo de projeto é básico para um desenvolvimento em VHDL. É o que permite chegar à síntese de um circuito, ou seja, a geração de uma lista otimizada de portas lógicas e registradores (RTL) configurando um sistema eletrônico sobre um dispositivo programável (PLD² ou FPGA³) e/ou posteriormente sua implementação como ASIC⁴.

Na metodologia Top-Down o projetista inicia a descrição do hardware no nível de abstração mais elevado e simula o sistema. Posteriormente, ele pode descrevê-lo com maiores detalhes (descendo níveis) e voltar a simulá-lo. O fluxo diagrama da Figura 1.3 mostra os passos básicos para modelagem de um projeto em HDL, iniciando por criar o projeto e seus módulos, simular e depurar os resultados obtidos pela simulação dos seus módulos bem como o projeto como um todo.

² PLD - *Programmable Logic Device* - dispositivo lógico programável.

³ FPGA - *Field Programmable Gate Array* - dispositivo lógico configurado pelo usuário (*fabless*).

⁴ ASIC - *Application Specific Integrated Circuit* - circuito integrado projetado para executar uma tarefa específica.

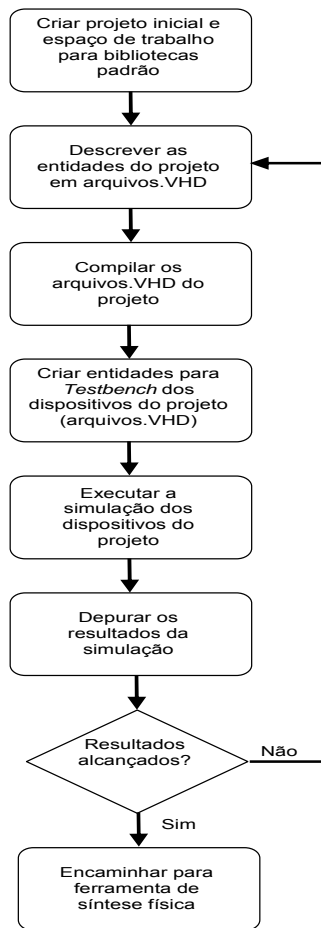


Figura 1.3 - Fluxo básico para modelagem de um projeto em HDL.

1.3 Ferramentas para projetos em HDL

Para se trabalhar com linguagens de descrição de hardware utilizam-se ambientes de projeto denominados Electronic Design Automation (EDA) que permitem modelar, descrever, verificar, simular e compilar a descrição do hardware sob projeto. Adotou-se como EDA para este livro o ModelSim⁵, que permite projetos em VHDL, Verilog, System Verilog e linguagens mistas de descrição de hardware.

No ModelSim todos os projetos são compilados em uma biblioteca. Normalmente se inicia uma modelagem através da criação de uma biblioteca de trabalho, denominada "Work". A biblioteca "Work" é utilizada pelo compilador do ModelSim como diretório padrão, destino das unidades compiladas do projeto.

Para iniciar um novo projeto siga o Fluxo Básico da Figura 1.3. Carregue o ModelSim, previamente instalado no computador. O Anexo A contém informações detalhadas de como obter e instalar este EDA. Após carregar o programa inicie um novo projeto conforme ilustrado na tela capturada do ModelSim (Figura 1.4).

⁵ ModelSim é um produto da Mentor Graphics.

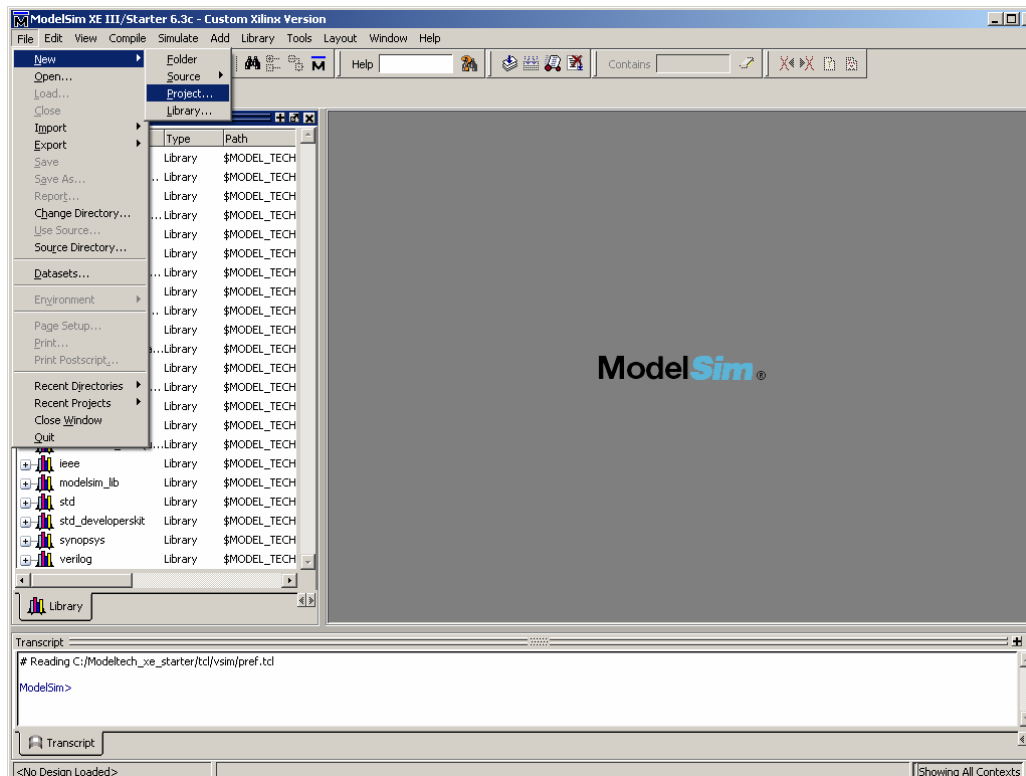


Figura 1.4 - Tela inicial do ModelSim.

Aponte o mouse e clique na barra do menu inicial na sequência <File→New→Project> chegando à tela "Create Project" (Figura 1.5). Determine um nome para o projeto, por exemplo, "Proj_Cap1", e após escolha sua localização (pasta) que pode ser identificada com o mesmo nome do projeto para facilitar a identificação do diretório no computador.

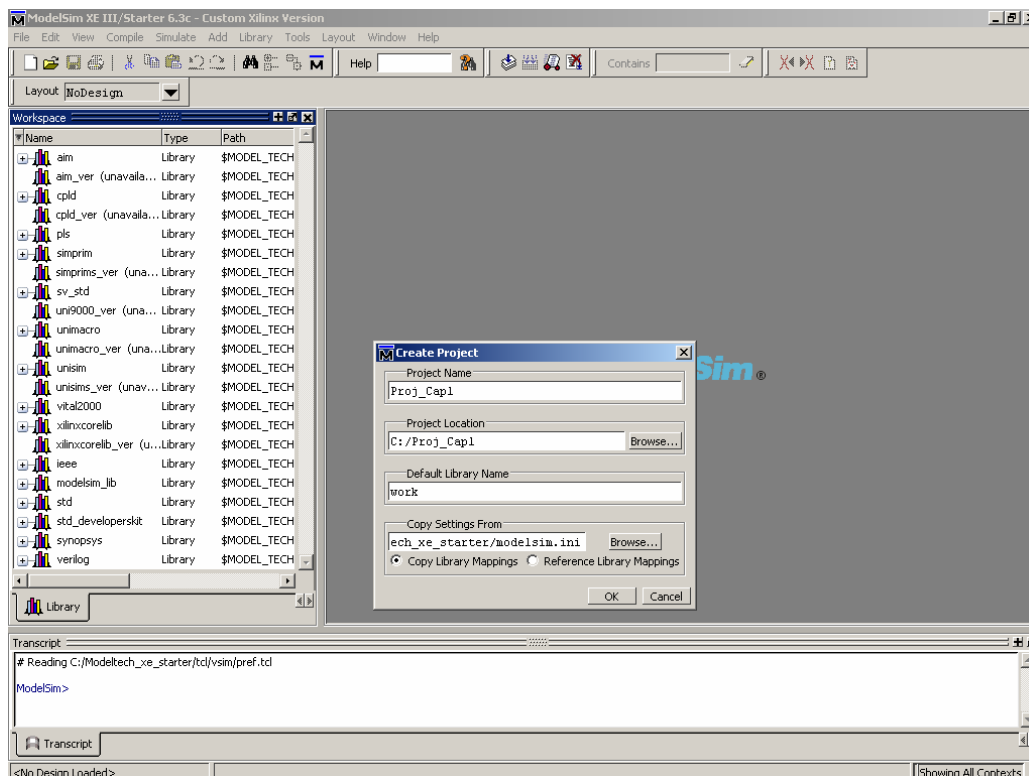


Figura 1.5 - Janela para a criação do "Proj_Cap1".

Uma vez digitados os nomes do projeto e do respectivo diretório, os demais campos serão assumidos por definição padrão (*default*) do ModelSim e desta forma encerra-se a criação do projeto, confirmando com "OK".

A próxima tela, "Add items to the Project" permite a criação de um novo arquivo fonte que será adicionado ao projeto. Esta tela também é utilizada para a inserção de novos itens ao projeto, conforme ilustrado na Figura 1.6.

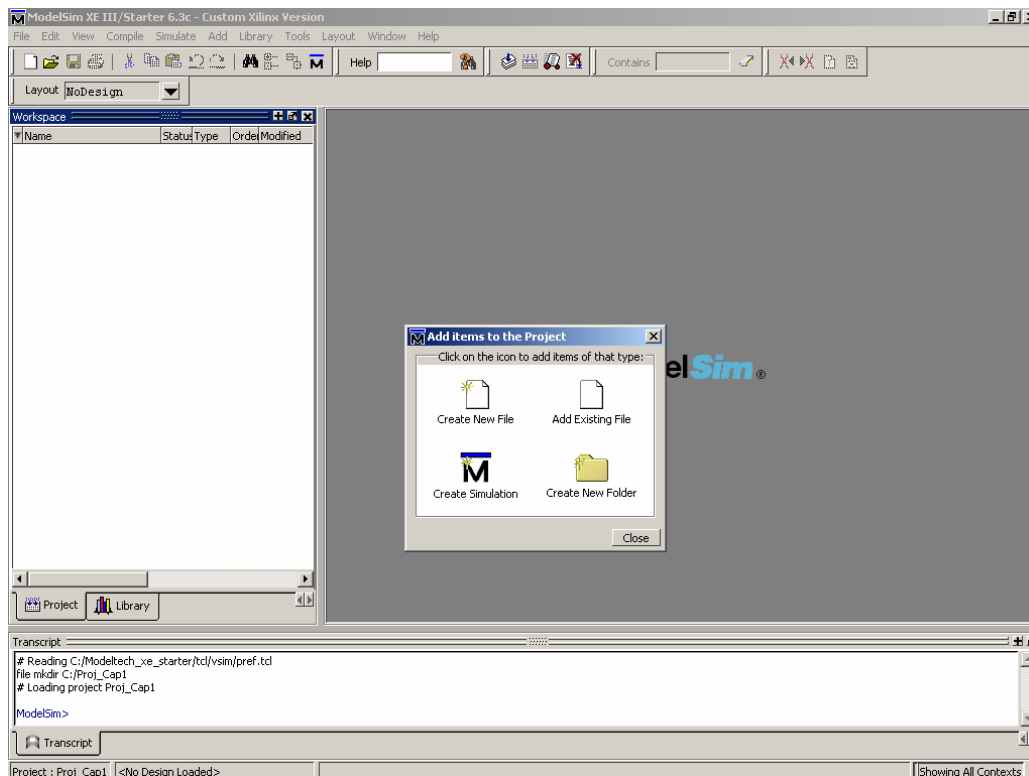


Figura 1.6 - Janela para a inserção de novos itens ao projeto.

A escolha do item "Create New File" permite introduzir o nome de um novo arquivo que será anexado ao projeto e determina uma extensão de acordo com o tipo da linguagem escolhida para desenvolver o projeto. Dado que a linguagem de descrição de hardware VHDL é o objeto de estudo adotado neste livro, todas as descrições de circuitos para a prática estão em VHDL.

Conforme ilustra a Figura 1.7, a tela "Create Project File" na sequência do item "Create New File", determina o nome de um arquivo fonte, que poderá conter desde a descrição uma simples porta lógica até um sistema complexo, por exemplo, um núcleo processador, tal como o PicoBlaze⁶. Para este tutorial sugere-se "and2" para o nome do arquivo que irá conter a descrição da entidade, e respectiva arquitetura, de uma porta lógica AND de duas entradas. Na tela "Create Project File" o campo "ADD File as type" determina a extensão que é agregada ao arquivo de trabalho (and2.vhd) e o campo "Folder" determina o "Top Level", a mais alta hierarquia de compilação⁷ dos arquivos do projeto. Esta hierarquia poderá ser reordenada de acordo com a adição de novos arquivos ao projeto. A ordenação poderá ser determinada de forma manual ou automática de acordo com o desenvolvimento das partes do projeto.

⁶ Microprocessador desenvolvido em VHDL para as famílias de FPGAs Spartan™-3, Virtex™-4, Virtex-II, e Virtex-II Pro da Xilinx.

⁷ O código fonte VHDL é compilado, gerando uma biblioteca de componentes que descrevem algebricamente a lógica dos componentes e uma listagem de conexões em nível de portas (Netlist).

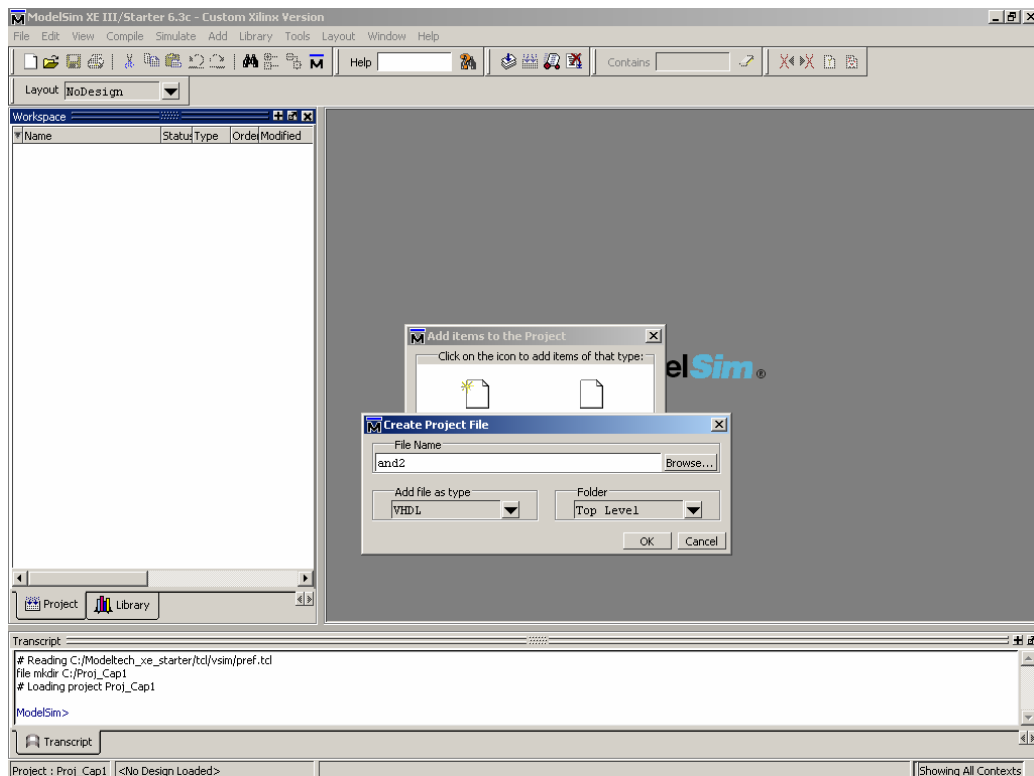


Figura 1.7 - Janela para criar um novo arquivo fonte do projeto.

Uma vez digitado nome do arquivo que irá conter a descrição da porta lógica AND de duas entradas, os demais campos serão assumidos por *default* e desta forma encerra-se a criação do arquivo fonte, confirmando com "OK".

Na próxima tela, ilustrada na Figura 1.8, observa-se que o novo arquivo fonte é adicionado ao projeto com a extensão "vhd". A coluna estado (*status*) apresenta o caractere "?", em função do arquivo estar vazio, não contendo uma descrição em VHDL.

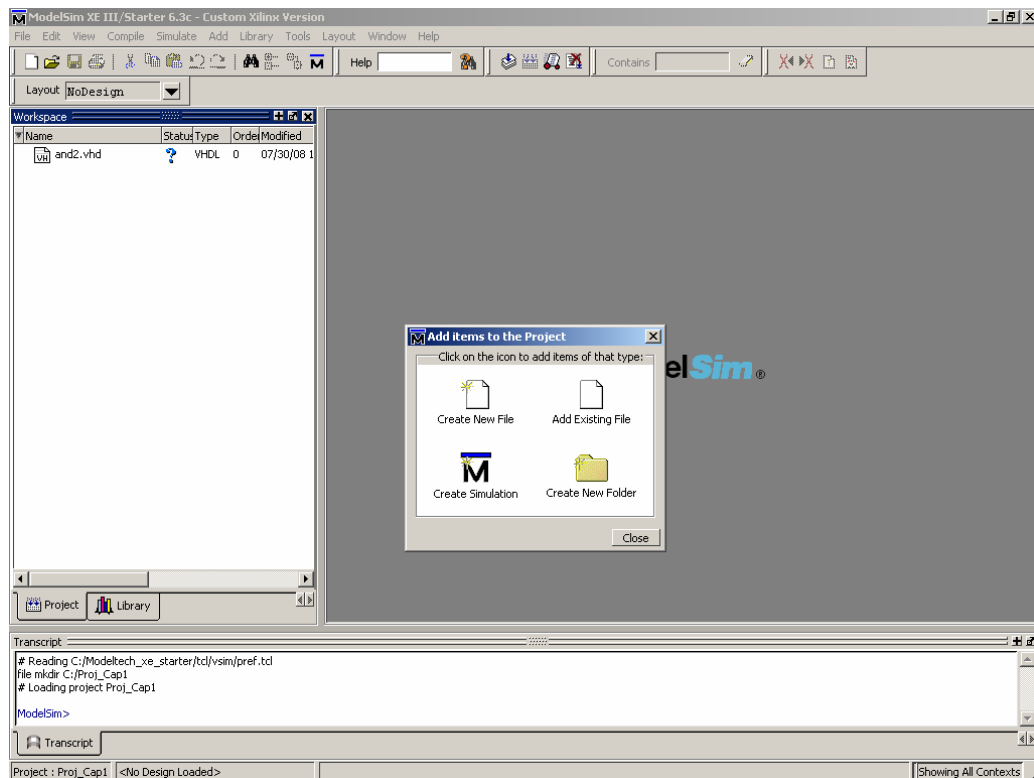


Figura 1.8 - Tela apresentada após criar um novo arquivo fonte.

Após o fechamento da tela "Add itens to the project", confirmando-se com "Close", é necessário abrir o arquivo de trabalho and2.vhd por meio de um duplo clique, como apresenta a tela da Figura 1.9.

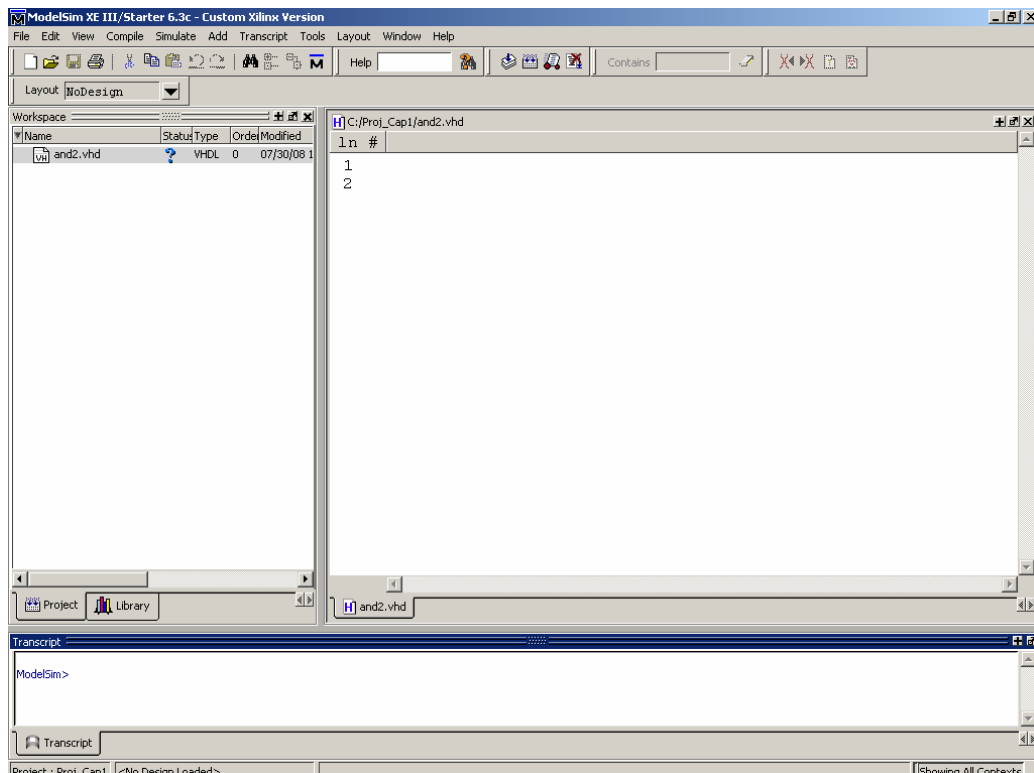



Figura 1.9 - Tela apresentada após abrir o arquivo de trabalho and2.vhd.

O novo arquivo de trabalho and2.vhd é aberto com duas linhas em branco pronto para ser digitada a descrição da porta lógica. Contudo, se for solicitada uma compilação deste

arquivo vazio, com um clique sobre ícone  localizado na barra de ferramentas para compilação, certamente ocorrerá um erro, conforme ilustrado na janela da Figura 1.10.

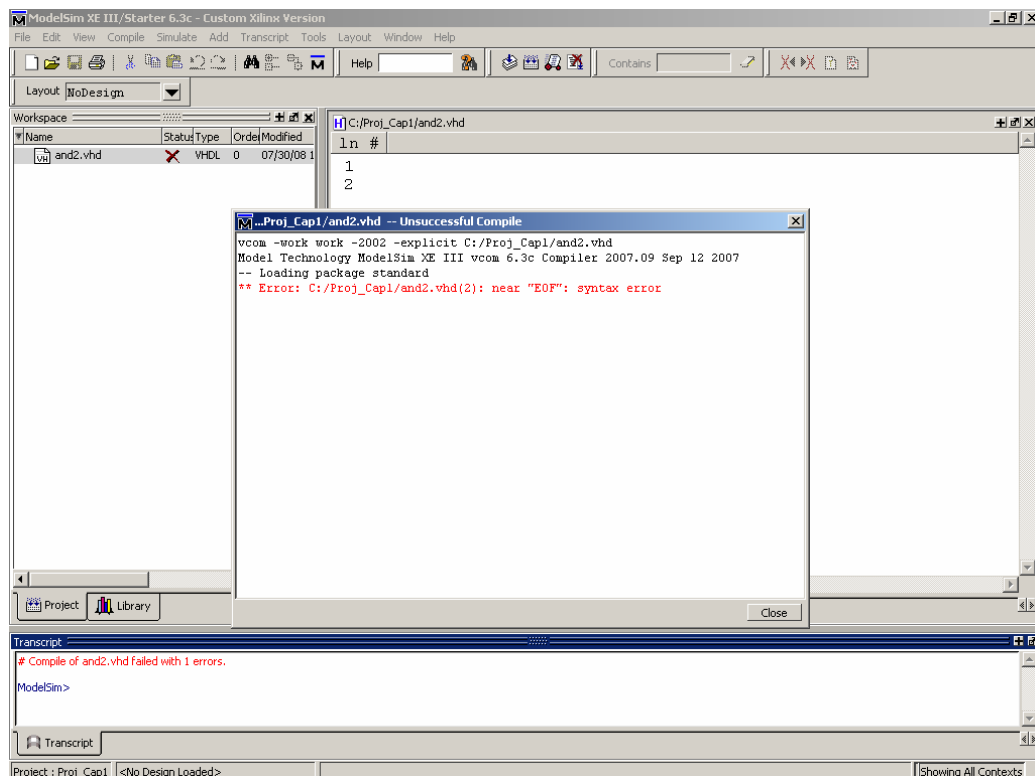


Figura 1.10 - Janela apresentada após a compilação do arquivo and2.vhd.

Sempre que a compilação resultar com erro, informado na janela "Transcript" localizada na parte inferior da tela principal, é necessário apontar sobre o resultado:

```
"# Compile of and2.vhd failed with 1 errors."
```

E com duplo clique obtém-se uma nova janela com mais detalhes sobre o erro ocorrido.

Para obter-se uma compilação com sucesso do hardware objeto do projeto, no caso do nosso exemplo uma porta lógica AND de duas entradas, é necessário digitar uma descrição completa e adequadamente formatada definida pelo padrão IEEE⁸ para linguagem VHDL.

Inicia-se uma descrição de hardware pela declaração do pacote (Package), contendo as constantes e bibliotecas que serão utilizadas na arquitetura da entidade objeto do projeto. A descrição deve seguir rigorosamente a formatação ilustrada pela Tabela 1, determinado pelo padrão IEEE 1164⁹.

⁸ IEEE - Institute of Electrical and Electronics Engineers - Instituto Americano de normalização para sistemas aeroespaciais, computadores e telecomunicações, engenharia biomédica, eletricidade e eletroeletrônicos.

⁹ Ao padrão IEEE 1076 de 1987, primeiro padrão industrial, foi acrescentada a norma IEEE 1164 de 1993, adicionando novos tipos de dados, tais como: std_logic e std_logic_vector.

Tabela 1 - Padrão IEEE 1164 para formatação da descrição VHDL.

<pre> LIBRARY IEEE; USE IEEE.STD_LOGIC_1164.all; USE IEEE.STD_LOGIC_UNSIGNED.all; </pre>	<p>PACKAGE (Bibliotecas)</p>
<pre> ENTITY exemplo IS PORT (<descrição dos pinos de entrada e saída>); END exemplo; </pre>	<p>ENTITY (Pinos de Entrada/Saída)</p>
<pre> ARCHITECTURE teste OF exemplo IS BEGIN PROCESS(<pinos de entrada e sinal >) BEGIN < descrição do dispositivo > END PROCESS; END teste; </pre>	<p>ARCHITECTURE (Arquitetura)</p>

Uma entidade (ENTITY) é uma abstração que descreve um sistema, uma placa, um chip, uma função ou uma simples porta lógica. Define a interface do sistema digital descrito com o mundo externo, identificando os pinos de entrada e saída.

A função de uma entidade é determinada por sua arquitetura, e a organização de uma arquitetura é dada por:

Declarações - que podem ser sinais, constantes, componentes, subprogramas;

Comandos - sempre iniciam com "begin" contendo blocos, atribuições a sinais, chamadas a subprogramas, instanciação de componentes e processos, terminando sempre por "end".

Assim a arquitetura (Architecture) descreve o comportamento ou a estrutura de um sistema digital, definindo como sua função é realizada.

A Figura 1.11 apresenta a entidade para o projeto proposto: uma porta lógica AND de duas entradas. Sua arquitetura é descrita pela função $y = f(i1, i2)$ e definida pela equação booleana $y = i1.i2$ resultante de sua tabela verdade.

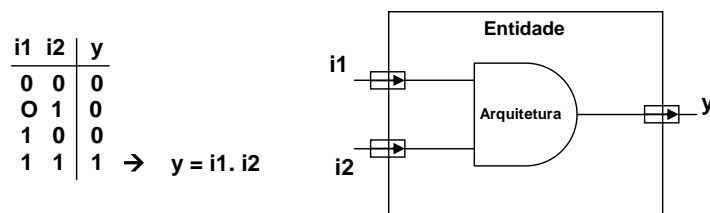


Figura 1.11 - Tabela verdade, equação booleana e diagrama da entidade.

Definida a entidade inicia-se a sua descrição. A partir de um cabeçalho sob forma de comentário identifica-se o objetivo do circuito descrito em VHDL. Na sequência seguem as declarações do pacote de bibliotecas e constantes, a entidade e respectiva arquitetura.

Na Figura 1.12 observa-se que no arquivo de trabalho foi digitado, da linha 1 até a 6, um cabeçalho sob forma de comentário. Comentários em VHDL devem ser precedidos por dois hífens (--). As linhas 7 e 8 contêm a indicação do pacote a ser utilizado e determinado pelo padrão IEEE 1164.

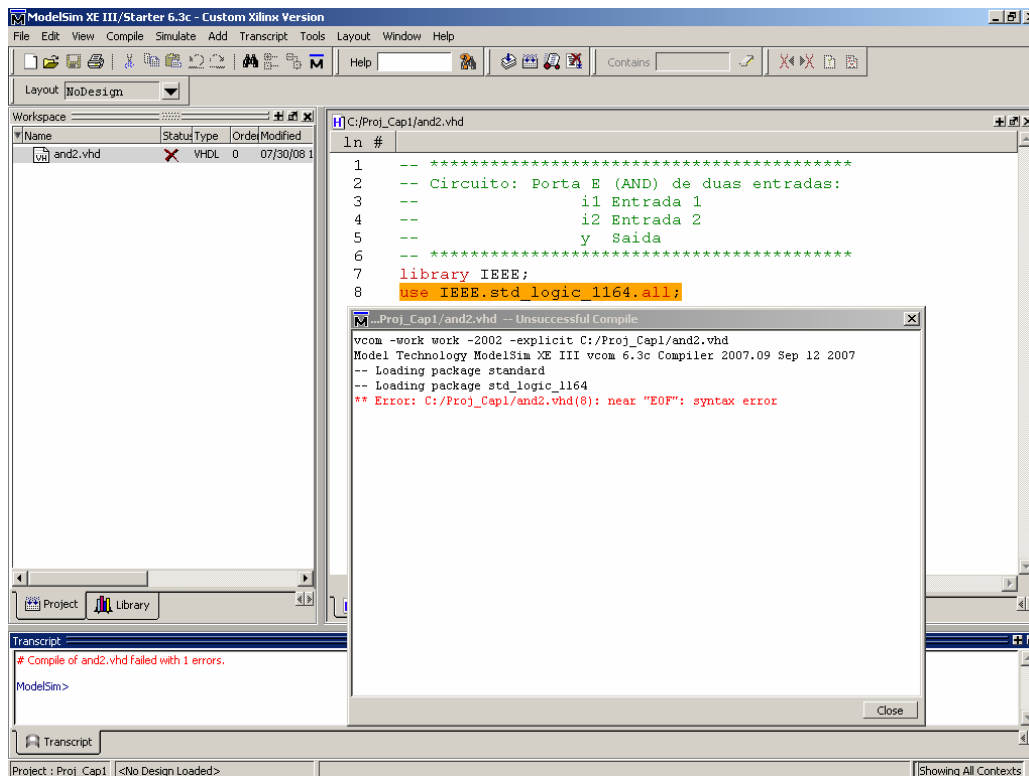


Figura 1.12 - Tela apresentada após a compilação.

Mesmo estando incompleta a descrição VHDL da porta lógica AND, foi solicitada a compilação do arquivo and2.vhd, onde verifica-se um erro, que é informado na janela "Transcript" como:

"**Error: E:/Feevale_E/Vhdl_Issue/Pratic_Vhdl/and2.vhd(8): near "EOF":syntax error".

Com um duplo clique sobre a mensagem de erro, obtém-se a indicação da posição no texto (salientado) da última sentença antes do erro.

Este erro ocorre, pois a entidade e sua respectiva arquitetura não foram descritas. Neste caso a compilação foi realizada com o intuito de apresentar a metodologia para a pesquisa de erros na modelagem de um sistema.

O modelo completo para descrição da porta lógica AND de duas entradas é transcrito como segue:

<pre>-- ***** -- Circuito: Porta E (AND) de duas entradas: -- i1 Entrada 1 -- i2 Entrada 2 -- y Saida -- *****</pre>	<p>} Cabeçalho contendo uma breve descrição do dispositivo modelado (Comentário opcional)</p>
<pre>library IEEE; use IEEE.std_logic_1164.all;</pre>	<p>} Package (Pacote) - constantes e bibliotecas;</p>
<pre>entity AND2 is port(i1 :in std_logic; i2 :in std_logic; y :out std_logic); end AND2;</pre>	<p>} Entity (Entidade) - pinos de entrada e saída;</p>
<pre>architecture rtl of AND2 is begin -- a definicao inicia por begin y <= i1 and i2; -- y = f(i1,i2) end rtl; -- a definição termina por end</pre>	<p>} Architecture (Arquitetura) - implementações do projeto;</p>

O modelo completo e sua respectiva compilação com sucesso estão ilustrados na Figura 1.13. Nesta edição da descrição da porta lógica AND é possível determinar claramente os três blocos básicos necessários para modelar qualquer sistema digital.

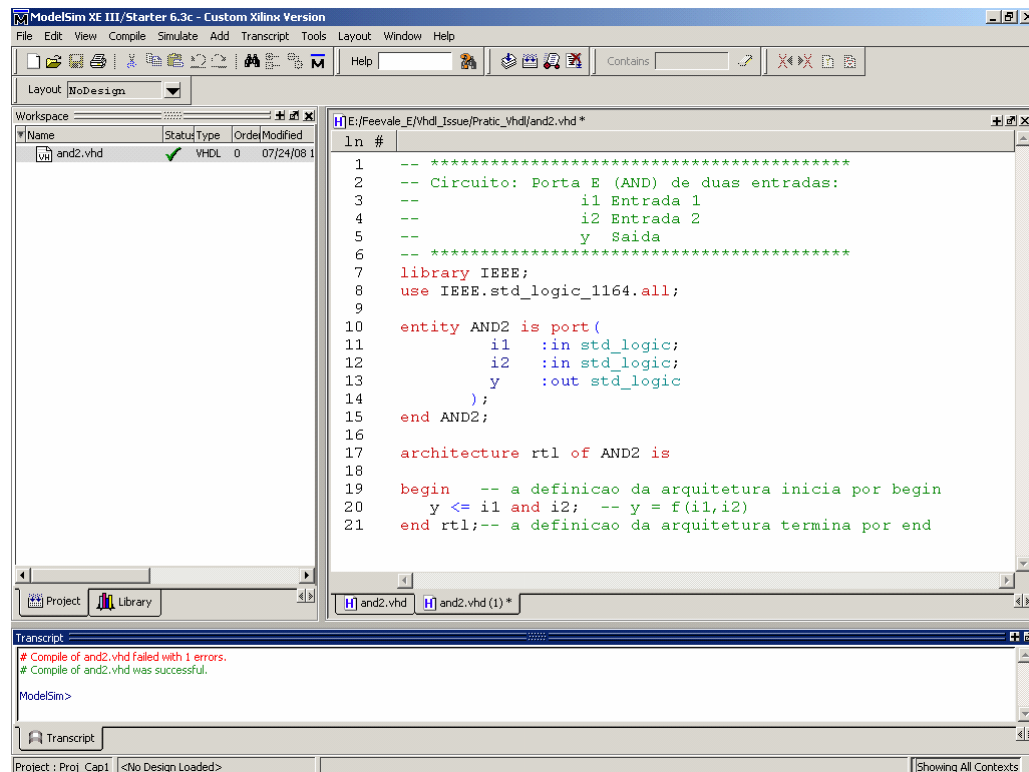


Figura 1.13 - Modelo completo e compilação com sucesso na janela "Transcript".

Quando for necessário utilizar algo não definido nas bibliotecas do VHDL padrão, faz-se uso do Pacote (package). A única restrição é que o pacote deve ser previamente definido, antes do início da entidade (entity).

O uso do Pacote é feito por meio de duas declarações: "library" e "use" (linhas 7 e 8 do código VHDL - Figura 1.13). Dos vários pacotes existentes, o mais conhecido e usado é o "STD_LOGIC_1164 da IEEE" que contém a maioria dos comandos adicionais mais usados em VHDL. O uso deste package é dado por:

```
library IEEE;
use IEEE.std_logic_1164.all;
```

A entidade é a parte principal de qualquer projeto, pois descreve a interface do sistema. Tudo que é descrito na entidade fica automaticamente visível a outras unidades associadas com ela. O nome do sistema é o próprio nome da entidade, assim, deve-se sempre iniciar um projeto em VHDL pela entidade. Como por exemplo, `entity AND2 is`, como descrito na linha 10 do código VHDL e ilustrado na Figura 1.13.

Uma entidade (entity) é composta de duas partes; parâmetros (parameters) e conexões (connections). Os parâmetros referem-se a dimensões, valores e constantes vistos do mundo externo, tais como largura de barramento e frequência de operação, e que são declarados como genéricos (generics). Conexões referem-se a onde e como ocorre a transferência de informações para dentro e fora do sistema, e são declarados por portas (ports).

A entidade de um sistema é tão importante que a própria arquitetura (architecture) é especificada na forma de arquitetura da entidade (architecture rtl of AND2 is, linha 17 do código VHDL - Figura 1.13).

Um sistema pode ser descrito em termos da sua funcionalidade, isto é, o que o sistema faz, ou em termos de sua estrutura, isto é, como o sistema é composto. A descrição funcional especifica as respostas nas saídas em termos das excitações aplicadas nas entradas. Neste caso não há nenhuma informação de como o sistema deverá ser implementado. A descrição estrutural, por sua vez, especifica quais componentes devem ser usados e como devem ser ligados. Esta descrição é mais facilmente sintetizada, porém exige mais experiência do projetista. Desta forma, pode-se ter várias arquiteturas capazes de implementar um mesmo circuito. Uma entidade pode ser formada por mais de uma arquitetura.

1.4 Bancada de Testes Virtual

Uma vez modelado o dispositivo (porta lógica AND) é necessário realizar sua simulação funcional, para comprovar e validar a função lógica implementada pela descrição VHDL.

A simulação funcional é uma estratégia que inicia por gerar sinais e dados necessários para estimular o dispositivo sob teste, de forma que este possa executar as operações modeladas e compiladas com sucesso nas etapas anteriores do projeto.

A estratégia de validação funcional do dispositivo é baseada no uso de *testbenches*, conforme apresentando no diagrama de blocos ilustrada na Figura 1.14.

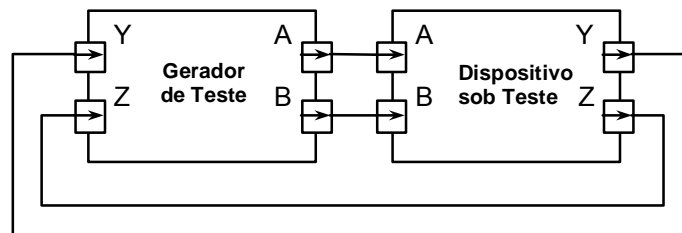


Figura 1.14 - Diagrama de blocos da estratégia para simulação.

Um *testbench* é uma bancada de testes virtual, implementada como uma descrição também em VHDL, que por sua vez contém uma instância VHDL do dispositivo a testar. Esta estrutura é apresentada na Figura 1.15, ilustrando, também, blocos geradores de estímulos, capturadores de saídas e formatares de resultados.

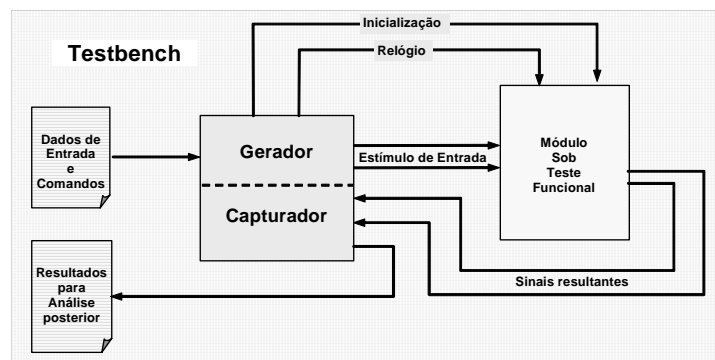


Figura 1.15 - Diagrama de blocos do método *testbench*.

Conforme a Figura 1.15, um *testbench* é um sistema autônomo que descreve o comportamento do ambiente externo instanciando o módulo sob teste e interagindo com este. Os estímulos são produzidos a partir da especificação de uma sucessão de tarefas previamente preparadas para o módulo em teste funcional. Em geral as tarefas são seqüências de dados que serão processados pelo módulo sob teste e devem ser criteriosamente elaboradas para representar as ações que o protótipo no futuro deverá realmente processar conforme especificado no projeto.

Testbenches são projetados para executar testes de um dispositivo de forma automática ou semi-automática. O dispositivo sob teste funcional, em geral, necessita de sinais de relógio (*clock*) e inicialização para o sincronismo e seqüenciamento de operações, fornecidos pelo bloco "Gerador", conforme a Figura 1.15. As partes referentes aos "Dados de Entrada e Comandos" e "Resultados para análise posterior" representam, em geral, informação armazenada em arquivos texto. Esta estrutura permite construir *testbenches* parametrizáveis. Os resultados do processamento, depois de adaptados pelo "Capturador" de sinais, são armazenados em arquivos para análise. Os blocos "Gerador" e "Capturador" para serem executados também necessitam de estímulos de entrada adequados para se

obter as respostas, em conformidade com o projeto, na saída dos módulos em teste, como exemplificado na Figura 1.15.

A Figura 1.16 ilustra o projeto da entidade *testbench1* para simulação funcional da porta lógica AND de duas entradas.

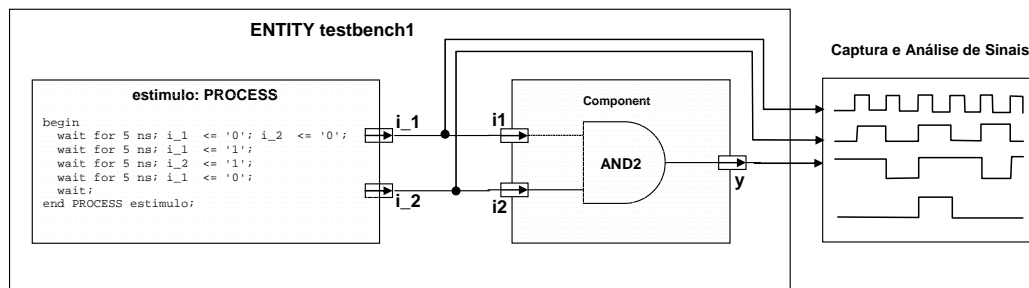


Figura 1.16 - Diagrama de blocos do testbench1.

A entidade *testbench1* é implementada como uma descrição em VHDL, que por sua vez contém uma instância VHDL do dispositivo a testar, no caso o componente AND2.

Para iniciar o desenvolvimento de um *testbench* é necessário criar um arquivo fonte para conter sua descrição, conforme ilustrado pela Figura 1.17. Para tanto clique com o botão direito do mouse, apontando no interior da janela "Workspace" para obter o menu <Add to Project> e desta forma abrir a janela "Create Project File".

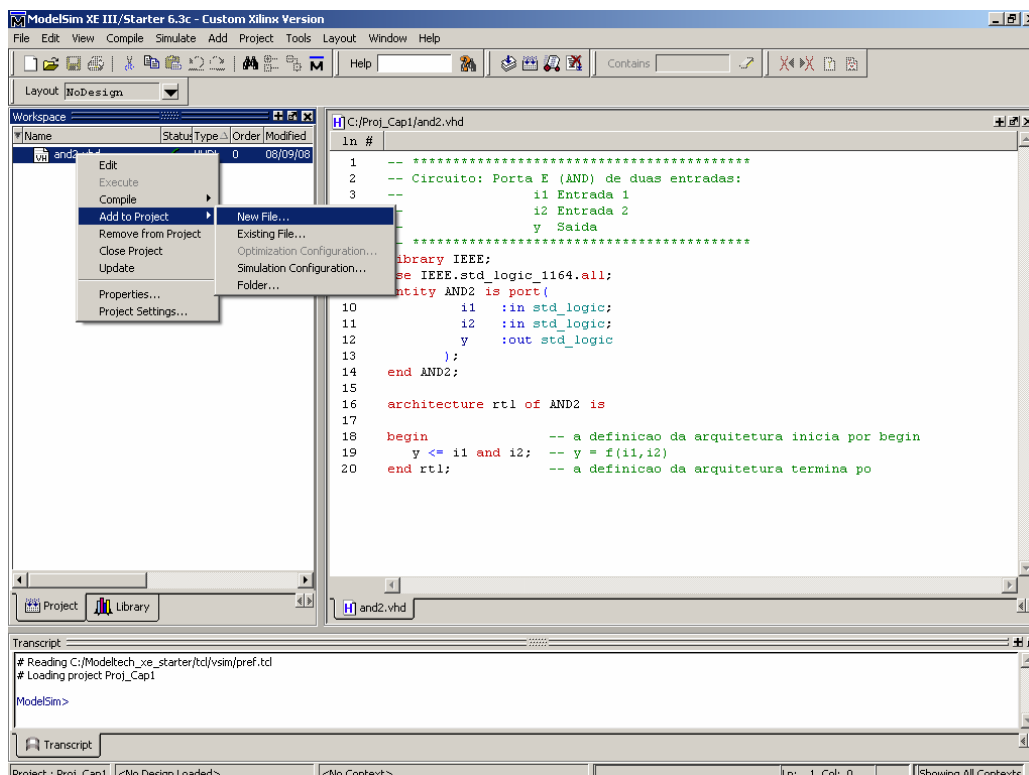


Figura 1.17 - Menu para criação de novo arquivo.

Na janela "Create Project File", apresentada na Figura 1.18, é então digitado o nome do arquivo que irá conter a descrição da entidade *testbench1* e o respectivo componente instanciado - entidade porta lógica AND de duas entradas. Os demais campos serão assumidos por definição inicial do ModelSim e desta forma encerra-se a criação do arquivo fonte para o *testbench* confirmando em "OK".

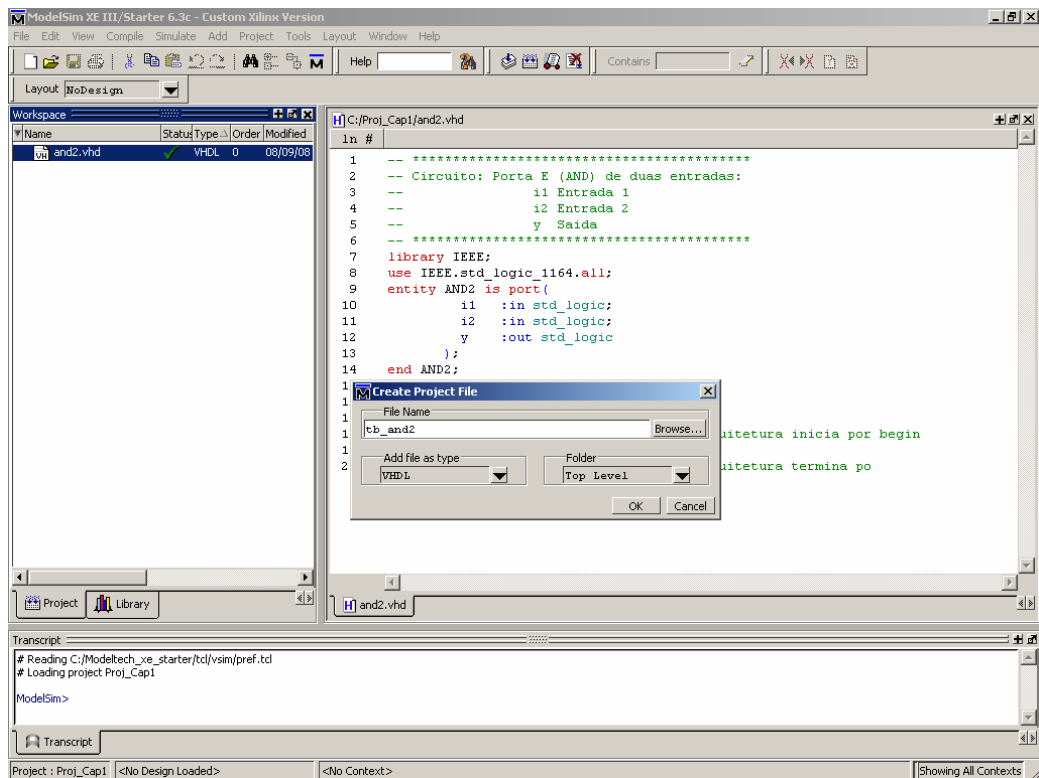


Figura 1.18 - Janela para nomear o arquivo de trabalho do *testbench*.

Na próxima tela, ilustrada na Figura 1.19, observa-se que o novo arquivo fonte *tb_end2* é adicionado ao projeto com a extensão "vhd" e seu estado (Status) apresenta um caractere "?", dado que o mesmo está vazio, não contendo uma descrição em VHDL.

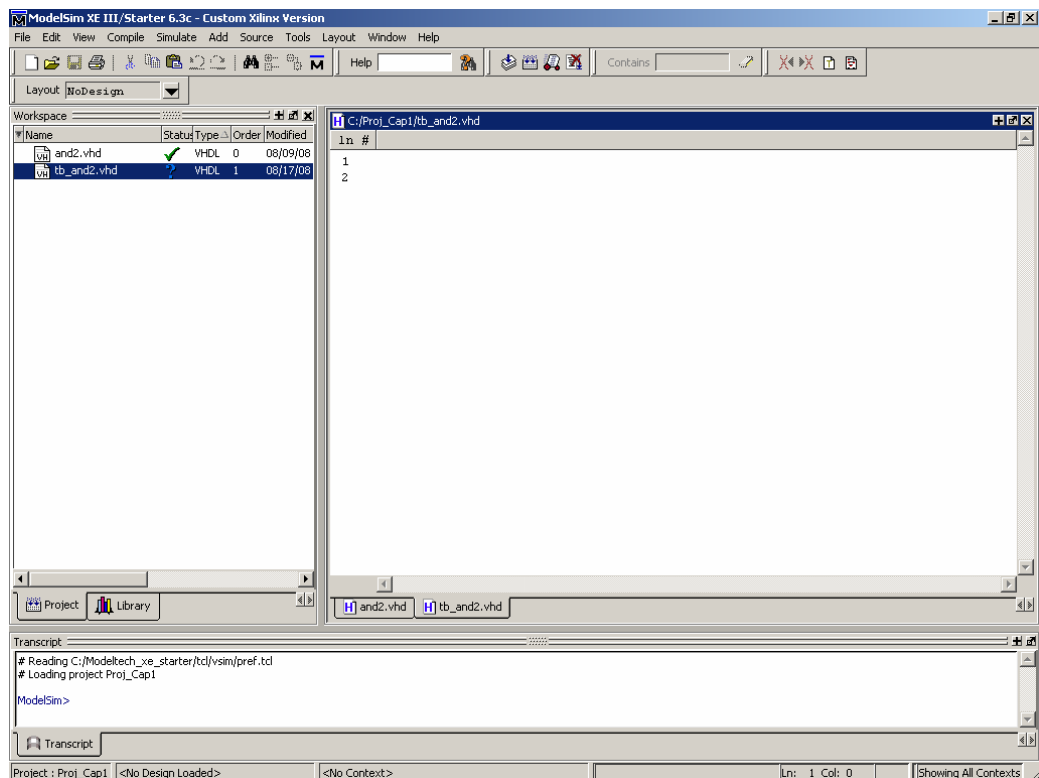


Figura 1.19 - Tela apresentada após criar um novo arquivo fonte.

Definida a arquitetura da entidade testbench1 inicia-se a sua descrição, sempre a partir de um cabeçalho sob forma de comentário para a identificação do objetivo do conteúdo descrito em VHDL, conforme ilustra a tela da janela de projeto da Figura 1.20.

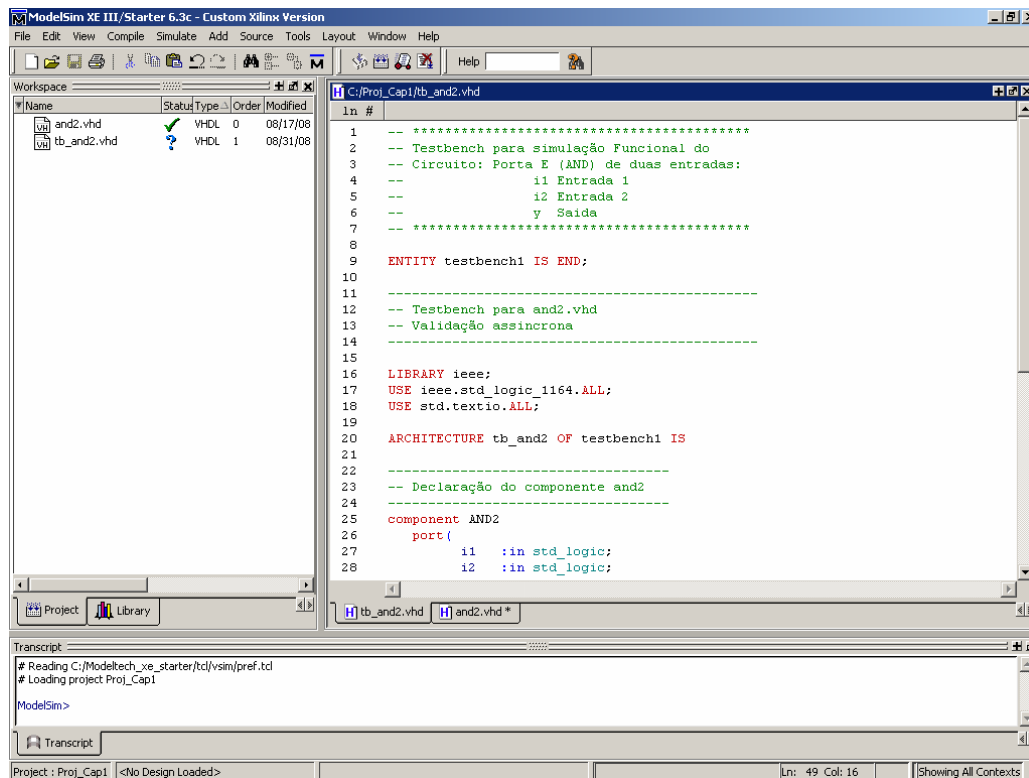


Figura 1.20 - Janela de projeto para o arquivo tb_and2.vhd.

Uma vez modelada a entidade testbench1 é necessário realizar uma compilação com sucesso para obter-se a validação de sua descrição VHDL (Figura 1.21).

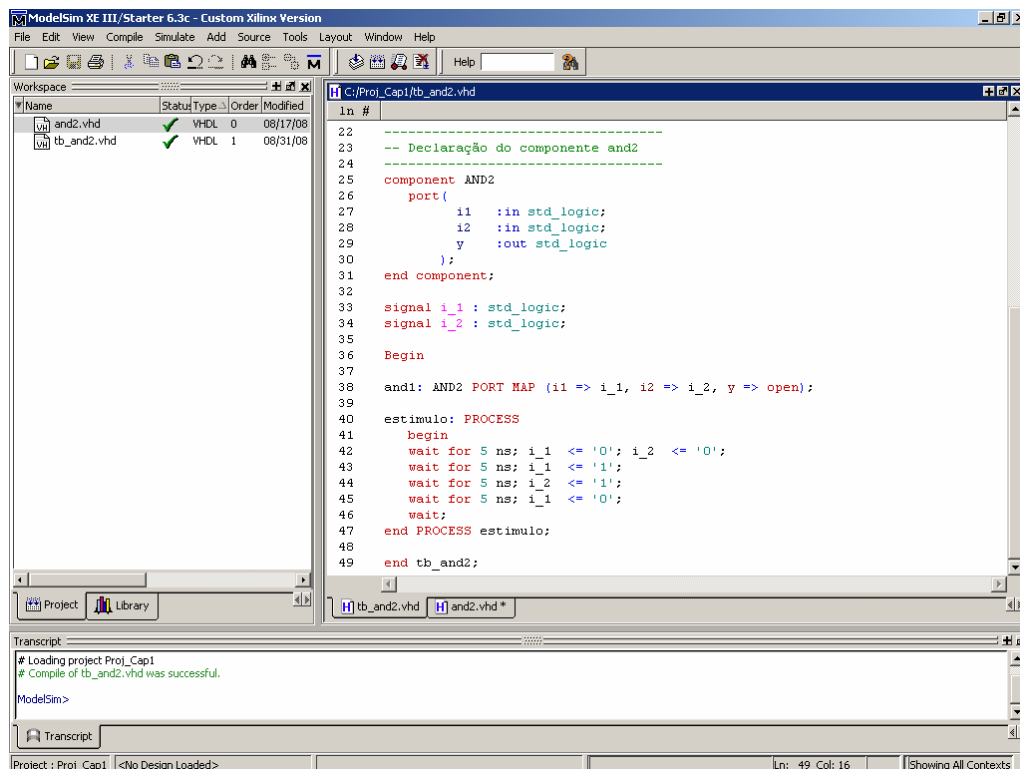


Figura 1.21 - Modelo completo do testbench1 e compilação com sucesso.

A descrição completa da entidade testbench1 é transcrita como segue:

<pre>-- ***** -- Testbench para simulação Funcional do -- Circuito: Porta E (AND) de duas entradas: -- i1 Entrada 1 -- i2 Entrada 2 -- y Saida -- *****</pre>	}	Cabeçalho contendo uma breve descrição do dispositivo modelado (comentário opcional);
<pre>ENTITY testbench1 IS END;</pre>	}	ENTITY (Entidade) - testbench1 é uma entidade sem pinos de entrada e saída;
<pre>-- Testbench para and2.vhd -- Validação assíncrona LIBRARY ieee; USE ieee.std_logic_1164.ALL; USE std.textio.ALL;</pre>	}	Package (Pacote) - constantes e bibliotecas;
<pre>ARCHITECTURE tb_and2 OF testbench1 IS -- Declaração do componente and2 component AND2 port(i1 :in std_logic; i2 :in std_logic; y :out std_logic); end component;</pre>	}	Declaração do componente and2, referente à sua arquitetura descrita no arquivo and2.vhd;
<pre>signal i_1 : std_logic; signal i_2 : std_logic;</pre>	}	Sinais auxiliares para a interconexão ao processo de estímulo;
<pre>Begin and1: AND2 PORT MAP (i1 => i_1, i2 => i_2, y => open);</pre>	}	Instância do componente and2 e interconexão do componente ao processo de estímulo;
<pre>estimulo: PROCESS begin wait for 5 ns; i_1 <= '0'; i_2 <= '0'; wait for 5 ns; i_1 <= '1'; wait for 5 ns; i_2 <= '1'; wait for 5 ns; i_1 <= '0'; wait; end PROCESS estimulo;</pre>	}	Implementação do processo de estímulo;
<pre>end tb_and2;</pre>		

Observe a linha 38:

```
"and1: AND2 PORT MAP (i1 => i_1, i2 => i_2, y => open);"
```

Na janela do projeto (Figura 1.21) está descrita uma única instância do componente AND2 e a respectiva interconexão do componente ao processo de estímulo, que é rotulada por:

```
"and1:"
```

Não há limite para o número de instâncias a serem determinadas, basta identificar cada nova instância por um rótulo diferente. Desta forma é possível identificar outro componente idêntico a AND2, por exemplo:

```
"and2: AND2 PORT MAP (i1 =>...);"
```

E assim por diante, quantos forem necessários para o desenvolvimento do projeto.

1.5 Simulação

O modelo completo e sua respectiva compilação estão ilustrados na Figura 1.22. Nesta edição da descrição somente uma instância do componente AND2, e a interconexão do componente ao processo de estímulo, foi declarada na linha anterior a declaração do bloco processo de estímulo.

Observar que na janela "Workspace" (Figura 1.22) é informada a ordem para compilação. Em um projeto, o ModelSim compila os arquivos de forma padrão, na ordem em que foram adicionados ao projeto. Por exemplo, o arquivo and2.vhd possui ordem zero, pois foi o primeiro a ser adicionado ao projeto. Os demais arquivos são ordenados sequencialmente.

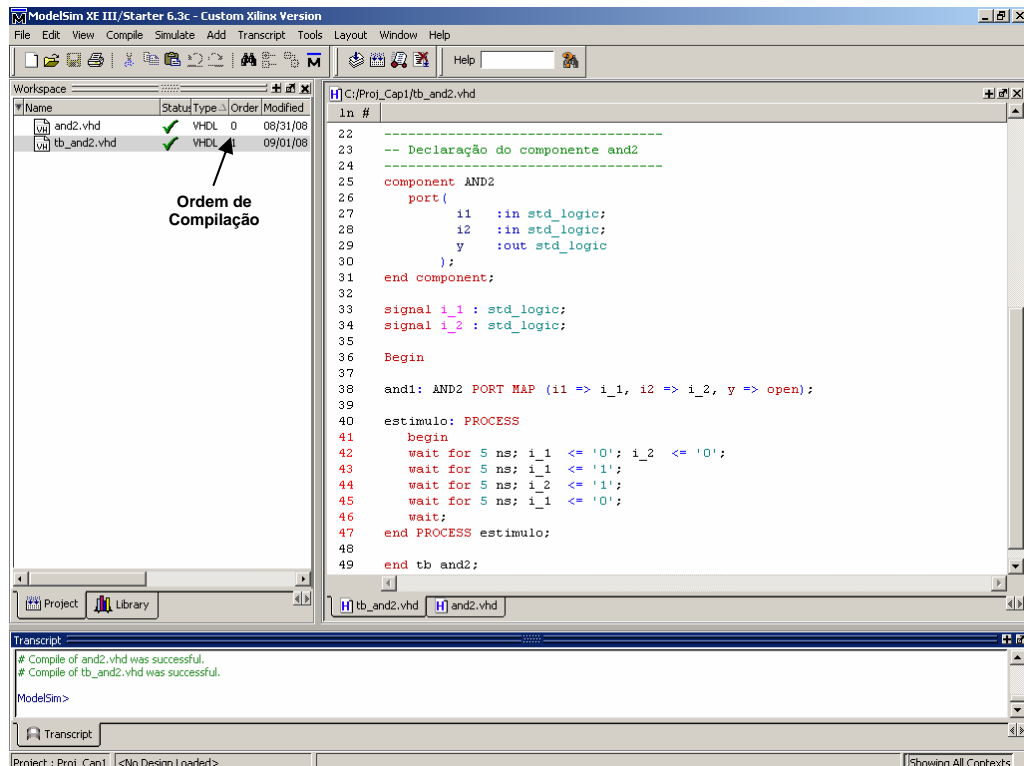


Figura 1.22 - Janela "Workspace" informado a ordem para compilação.

Existem duas alternativas para a mudança na ordem de compilação:

Selecionar e compilar cada arquivo individualmente, ou

Especificar uma ordem para compilação personalizada.

Para especificar uma ordem para compilação personalizada, selecione <Compile → Compile Order...> na barra de menu principal do ModelSim, obtendo a janela "Compiler Order" conforme ilustrado pela Figura 1.23.

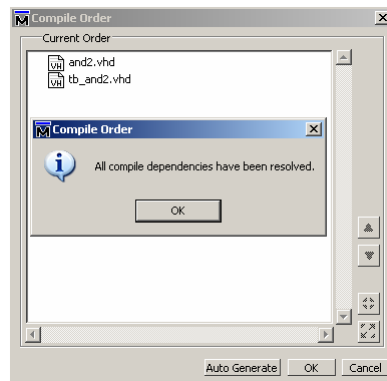


Figura 1.23 - Opção "Auto Generate" ativada.

A opção "Auto Generate" é utilizada somente em projetos em VHDL. Esta opção determina a ordem correta de compilação dos arquivos do projeto, realiza múltiplos passes ao longo dos arquivos e determina a ordem para compilação a partir do topo. Caso a compilação de um arquivo falhar devido a dependências relativas aos outros arquivos, é movido para o baixo e, em seguida, é compilado novamente após a compilação de todos os arquivos de ordem superior. Este processo permanece em execução até que todos os arquivos sejam compilados com sucesso, ou até que um arquivo não possa ser compilado por outros motivos que não suas dependências. A ordem observada na guia projeto não é, necessariamente, a ordem na qual os arquivos serão compilados.

Resolvidas todas as dependências para as compilações das entidades AND2, e respectiva tb_end2, pode-se então iniciar a fase de simulação em <Simulate→Start Simulation...> na barra de menus da janela principal, conforme ilustrado na Figura 1.24.

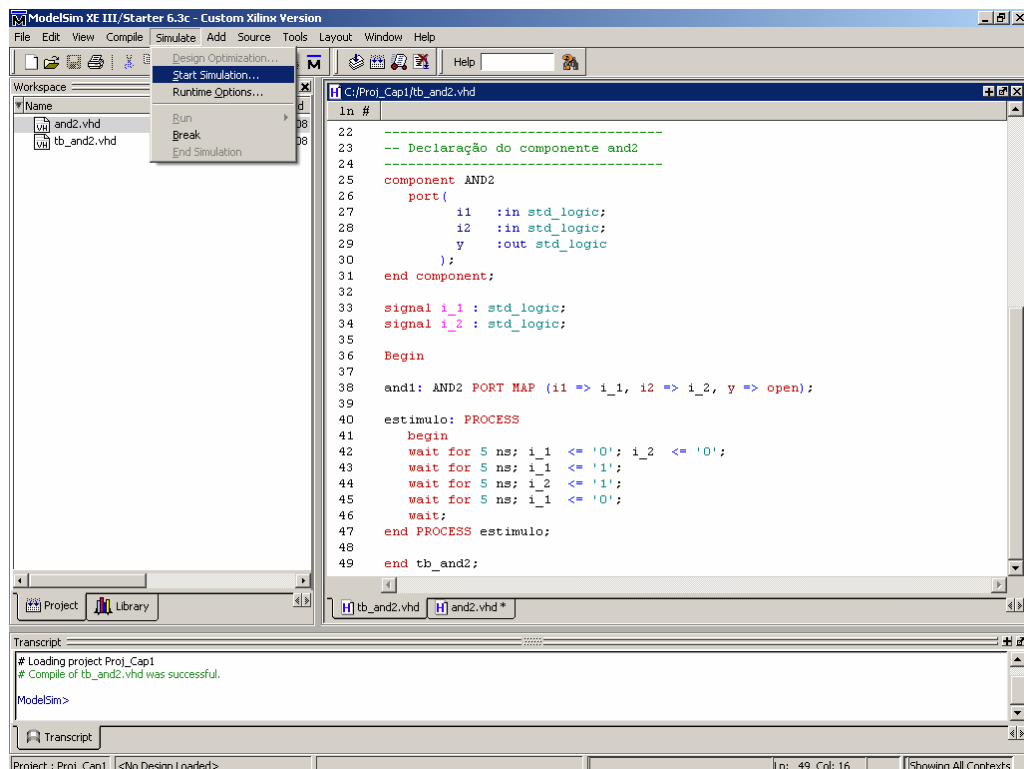


Figura 1.24 - Menu para iniciar a simulação.

A partir deste comando abre uma nova janela denominada "Start Simulation", que na sequência é obtida expandindo-se o diretório <Work→testbench1> (Figura 1.25), permitindo a escolha da entidade testbench1 como alvo da simulação.

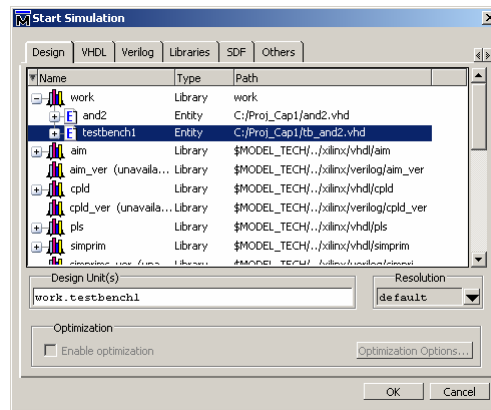


Figura 1.25 - Janela para escolha da entidade alvo da simulação.

Confirmando com "OK" na janela "Start Simulation", o ModelSim determina a entidade testbench1 como o projeto alvo para a simulação e desta forma obtém-se no "Workspace" a janela "sim" para a escolha de qual instância dos componentes que poderá ser analisada.

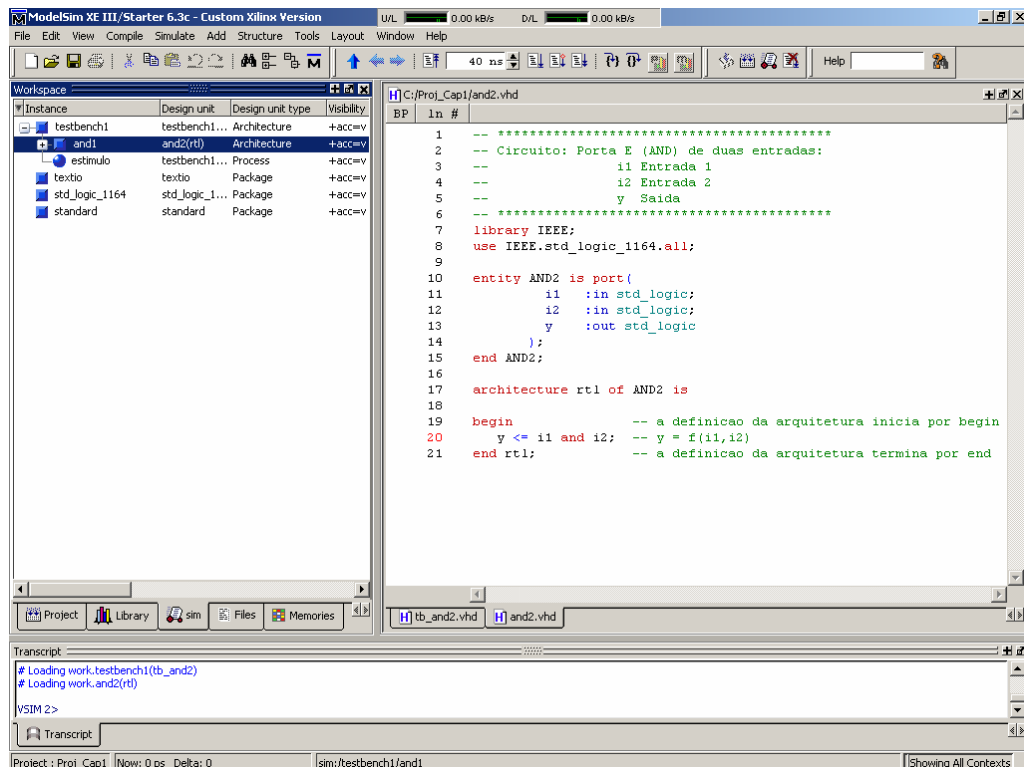


Figura 1.26 - Janela para escolha da entidade alvo da simulação.

A escolha, neste exemplo, será a instância do componente "and1", conforme ilustrado na Figura 1.26. Neste ponto da fase de simulação é importante observar a janela "Transcript", ilustrada na parte inferior da Figura 1.26, onde são descritas as mensagens de carregamento para o simulador das entidades que compõem o projeto e que neste caso ocorreu de forma normal.

As ocorrências de erros nesta fase de carregamento das entidades que compõem o projeto não ocorrem com a mesma frequência que os erros durante a fase de modelagem. Contudo, quando ocorrem, são difíceis de serem localizados, pois não é resultado da lógica funcional dos diversos dispositivos sob simulação.

A próxima etapa da simulação é a da escolha dos sinais para a captura e análise, conforme ilustrado na Figura 1.27. Nesta mesma Figura observa-se que, para se analisar a lógica do

componente AND2, é necessário capturar os sinais "i_1" e "i_2" da entrada do componente AND2 bem como o sinal "y" da saída e analisá-los para validar a operação lógica.

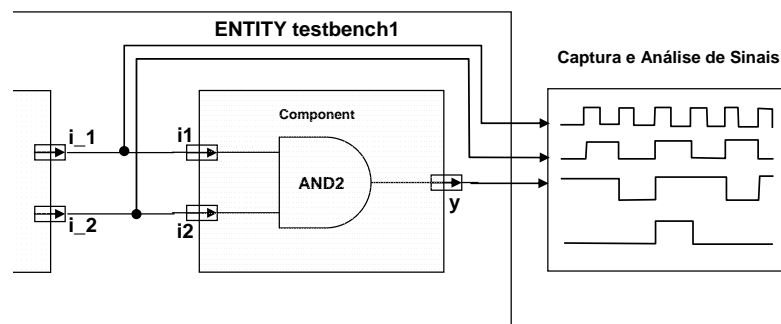


Figura 1.27 - Captura e análise dos sinais.

Para análise dos sinais no ModelSim do componente "and1", seleciona-se sua instância na janela "Workspace" (botão esquerdo do mouse). Após aponte para a tarja azul e clique com o botão direito do mouse para obter o menu <Add→ Add to Wave>. Novamente, com o botão esquerdo do mouse abre-se a janela "Wave - default", conforme ilustrado na Figura 1.28.

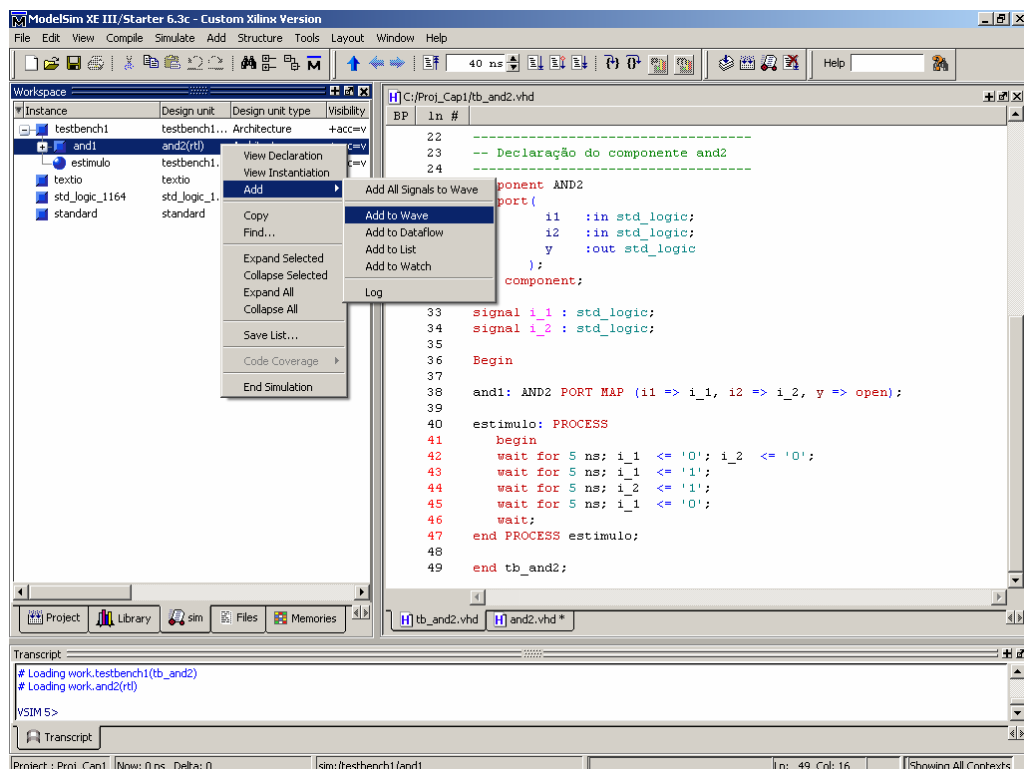




Figura 1.28 - Adicionando os sinais para análise.

A janela "Wave - default" se abre na tela principal do ModelSim como mostra Figura 1.29. Esta janela pode ser desvinculada da janela principal, bastando clicar sobre ícone  ("unlock") na janela "Wave - default" à direita, conforme indicado (Figura 1.29).

Uma vez determinado o tempo de execução da simulação, aponte sobre o ícone  ("run") e execute uma rodada de simulação por um intervalo de tempo previamente ajustado no "Run Length", obtendo as formas de onda dos sinais "i_1" e "i_2" da entrada da AND2 bem como o sinal "y" na saída, conforme ilustrado na Figura 1.31.

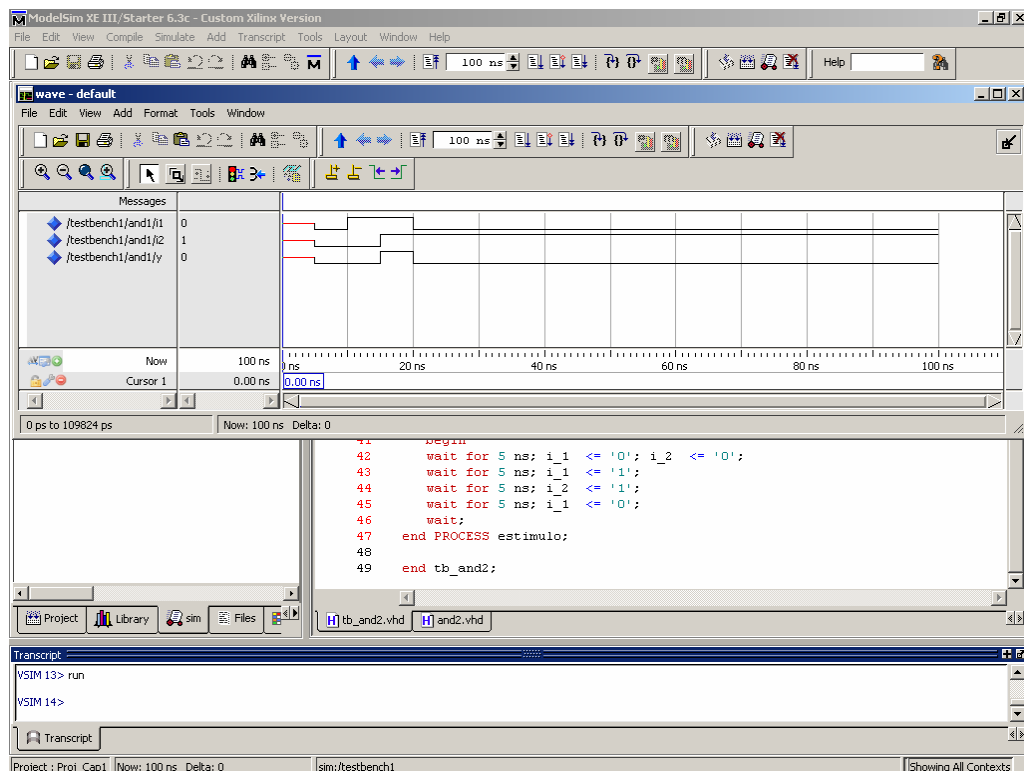


Figura 1.31 - Janela "Wave - default" após a execução de uma rodada de simulação.

Na

Figura 1.32, detalhe da Figura 1.31, há um detalhamento da simulação da porta lógica utilizada como exemplo neste tutorial.

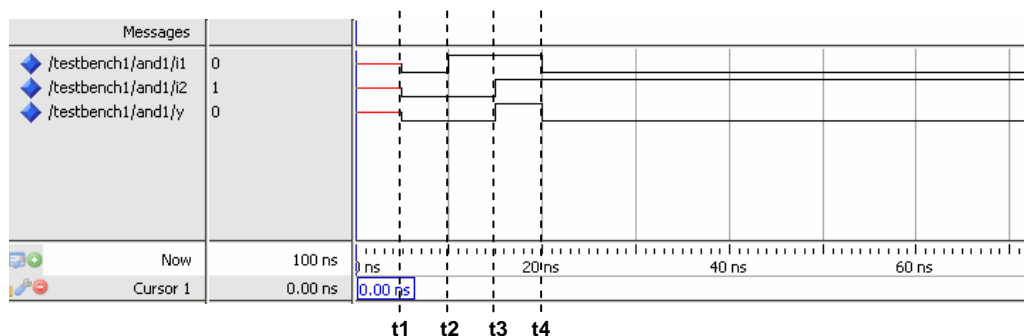


Figura 1.32 - Detalhamento maior da simulação da descrição.

Após 5 ns (linha 42 do código VHDL - Figura 1.31) do início da simulação (t1) as entradas i1 e i2 são colocadas em nível lógico "0" resultando na saída do circuito o nível lógico "0" ($0 \text{ AND } 0 = 0$). Em t2 (linha 43 do código VHDL) a entrada i1 é colocada em nível lógico "1", logo a saída y permanece em "0" ($1 \text{ AND } 0 = 0$). No instante t3 tem-se $i1=1$ e $i2=1$, resultando em $y=1$ ($1 \text{ AND } 1 = 1$) (linha 44 do código VHDL). A partir de t4 a saída y permanece em nível lógico "0" ($0 \text{ AND } 1 = 0$) dado que i1 passa a ter o nível lógico "0" (linha 45 do código VHDL- Figura 1.31).

Após os resultados esperados terem sido validados pela simulação funcional chega-se ao final do fluxo básico de projetos em VHDL apresentado na Figura 1.3 deste capítulo. Caso os resultados obtidos através da simulação não tivessem sido alcançados haveria necessidade de modificação no código fonte e repetição da simulação até atingir os resultados corretos.