Tutorial ModelSim

Autor: Prof. Marcelo Ruaro (SETREM-RS) Julho de 2020

Instalação

1. Acessar o site do ModelSim para download da versão estudante

- https://www.mentor.com/company/higher_ed/modelsim-student-edition
- Clicar em Download Student Edition
- Se registrar
- Aceitar os termos

2. Baixar e Instalar ModelSim

- Ir no email cadastrado e clicar no link para donwload. Deverá baixar o ModelSim (~350 MB)
- Instalar o ModelSim

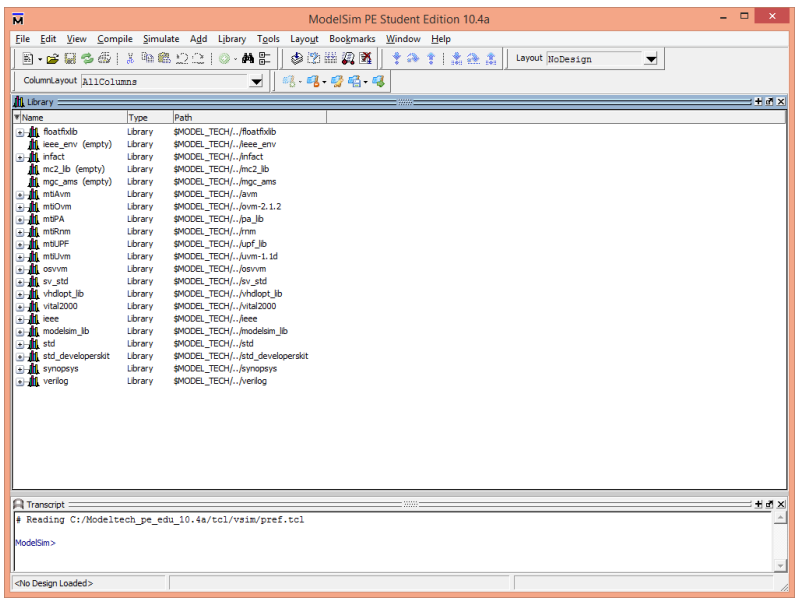
3. Ativar o ModelSim

- Ao final da instalação uma página da web será carregada. Completar os campos obrigatórios (*)
- Clicar em request licence
- Abrir o email e baixar a licença e colar na pasta principal onde o MS foi instalado, no meu caso: C:\Modeltech_pe_edu_10.4
- OBS: a licença é valida por 180 dias (6 meses), depois disso precisará fazer todo o processo novamente, baixar, instalar e ativar.

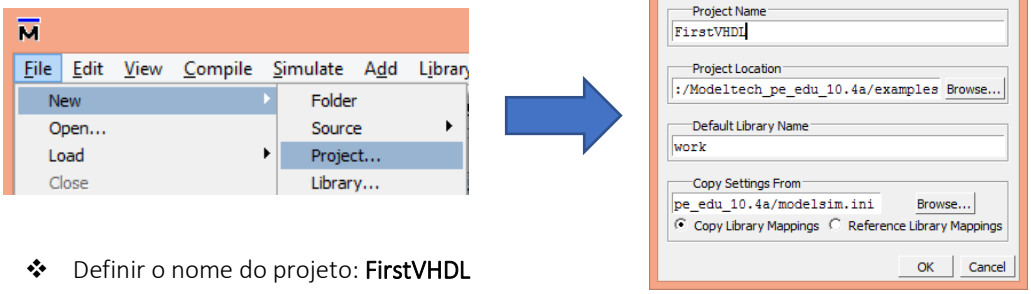
Criação de Projeto, Compilação e Simulação

1. Abrir o ModelSim

- Deverá aparecer uma janela igual a essa:



2. Criar um Novo Projeto



- Definir o nome do projeto: **FirstVHDL**
- Definir a localização
- Criar projeto clicando em OK
- Não selecionar nenhuma das opções "create new file", etc. Apenas clicar em **close**

Note que o projeto está vazio. Primeiro passo é criar um novo arquivo. VHDL ao projeto.

3. Criar um arquivo em VHDL

- Abrir seu editor de texto e escrever algum código VHDL. Pode ser o código abaixo:

```
library ieee;
use ieee.std_logic_1164.all;

entity simple_example is
    port (
        a : in std_logic;
        b : in std_logic;
        o : out std_logic
    );
end simple_example;

architecture simple_example of simple_example is
begin

    o <= a and b;

end simple_example;
```

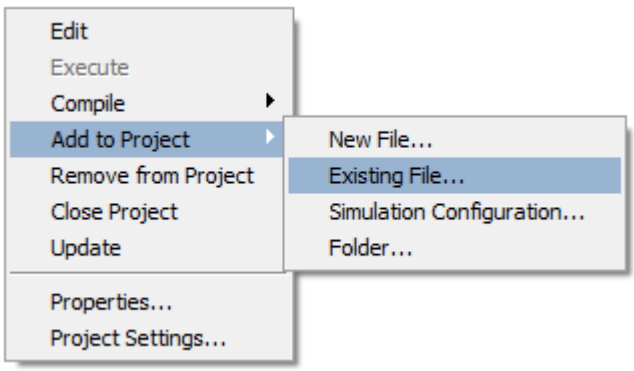
- Salvar o arquivo com o nome **simple_example.vhd** dentro do diretório principal do projeto.

OBS1: Note que **.vhd** é a extensão para qualquer arquivo VHDL

OBS2: Tome cuidado para não haver espaçamentos em branco na URL até o seu arquivo, ex: C:\Documents\Meu projeto aula 1\simple_example.vhd (esta errado, tem espaço na parte vermelha do caminho)

4. Adicionar o arquivo VHDL para o projeto

- Adicionar o arquivo para dentro do projeto. Botão direito do mouse na área em branco do ModelSim, seguir a instrução abaixo

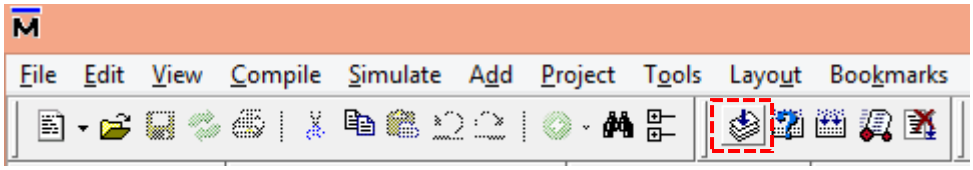


- Adicione o arquivo especificando a localização onde você o salvou.

OBS: Depois disso, faça uma configuração importante para exibir erros detalhados sempre ao compilar. Para isso, vá em *Project Settings* do menu acima, e selecione a opção: ***Display Compiler Output***

5. Compilar o Arquivo

- Clique no ícone de compilar, conforme mostrado abaixo:



- Espere pelo seguinte tipo de mensagem

Em caso de ERRO:

```
# vcom -work work -2002 -explicit -stats=none C:/Users/Marcelo/Desktop/VHDL/simple_example.vhd
# Model Technology ModelSim PE Student Edition vcom 10.4a Compiler 2015.03 Apr  7 2015
# -- Loading package STANDARD
# -- Loading package TEXTIO
# -- Loading package std_logic_1164
# -- Compiling entity simple_example
# ** Error: C:/Users/Marcelo/Desktop/VHDL/simple_example.vhd(11): near "aa": syntax error
#
#

VSIM(paused)>
```

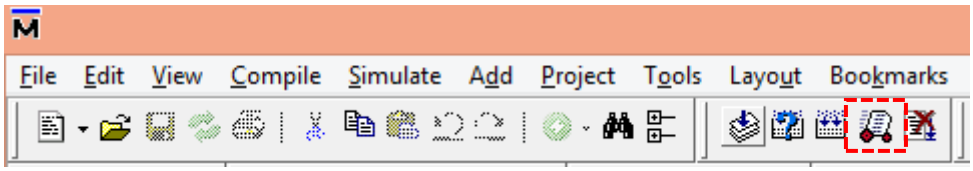
Em caso de SUCESSO (o qual deverá acontecer):

```
# vcom -work work -2002 -explicit -stats=none C:/Users/Marcelo/Desktop/VHDL/simple_example.vhd
# Model Technology ModelSim PE Student Edition vcom 10.4a Compiler 2015.03 Apr  7 2015
# -- Loading package STANDARD
# -- Loading package TEXTIO
# -- Loading package std_logic_1164
# -- Compiling entity simple_example
# -- Compiling architecture simple_example of simple_example
#
#

VSIM(paused)>
```

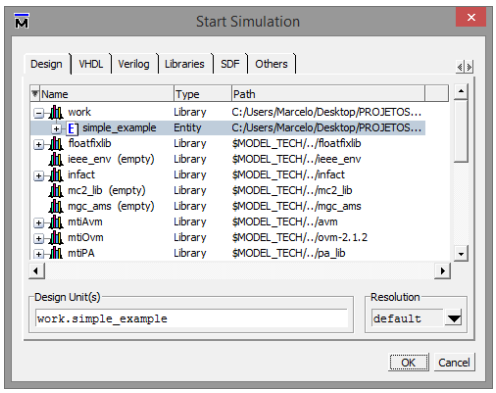
Só para testar se você está compilando o arquivo correto, gere um erro no arquivo fonte escrevendo qualquer coisa nele e tentando compilar. Depois, desfaça o erro e vamos continuar.

6. Simule o projeto

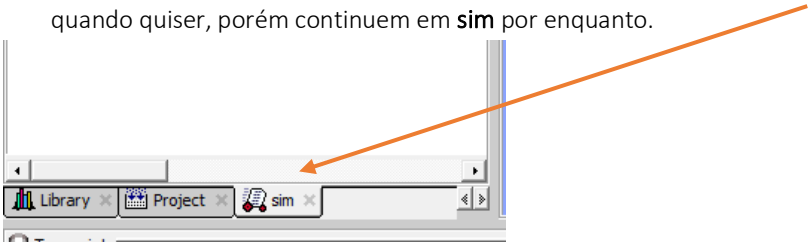
- Inicie a simulação clicando no ícone



- Selecione o arquivo *simple_example* dentro de work



- Note que várias coisas mudaram na janela do ModelSim, essa é a visão de simulação, você pode voltar para a aba **Project** quando quiser, porém continuem em **sim** por enquanto.



- Uma mensagem assim deve aparecer terminal do ModelSim:

```
VSIM(paused)> vsim -novopt -t 1ns +notimingchecks work.simple_example
# vsim
# Start time: 16:12:51 on Jul 23,2020
# Loading std.standard
# Loading std.textio(body)
# Loading ieee.std_logic_1164(body)
# Loading work.simple_example(simple_example)

VSIM(paused)>
```

- PRONTO!

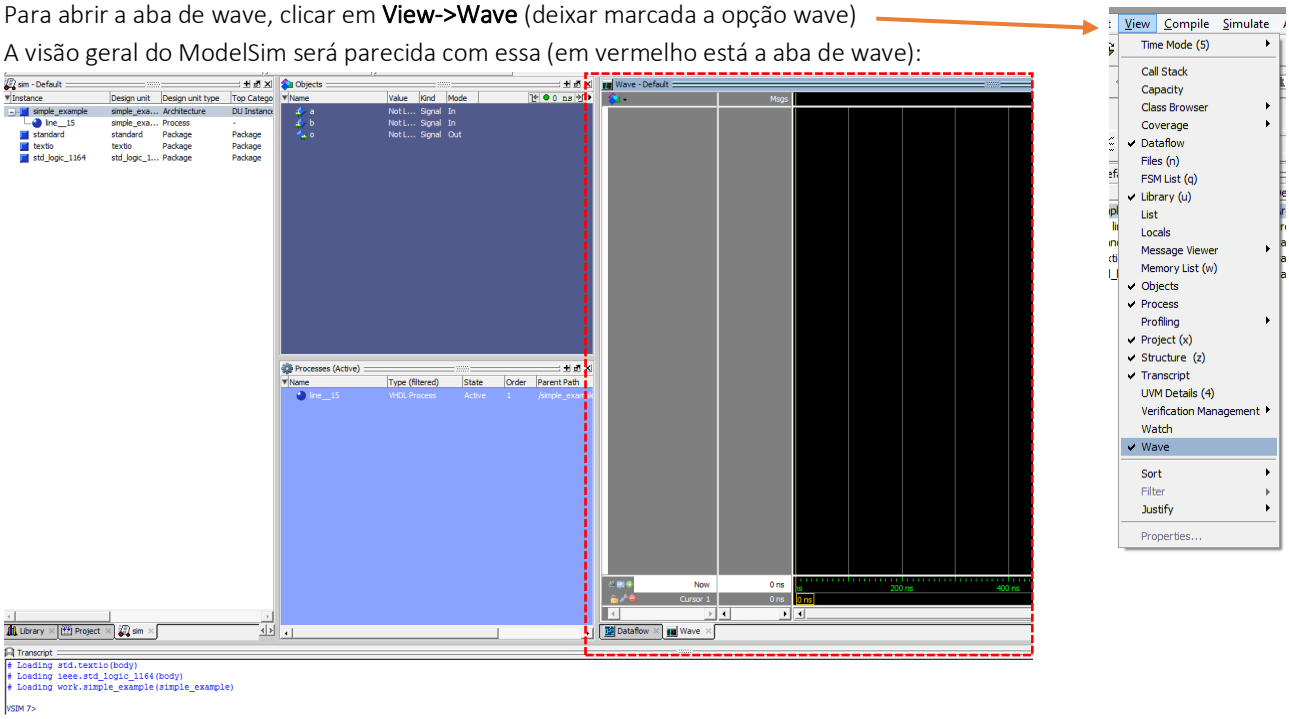
Simulação com Valores de Entrada

A simulação por si só não diz muita coisa pro projetista, a não ser se o projeto compilou ou não. O que realmente interessa é ver se o comportamento do circuito está agindo como esperado, observando o valor dos seus sinais.

Para isso vamos usar uma aba muito importante que mostra o comportamento dos sinais ao longo do tempo, chamada de **waveform** (ou somente de **wave**).

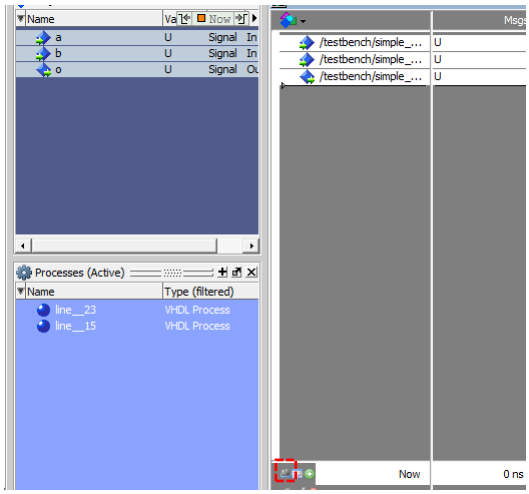
7. Abrir aba Wave

- Para abrir a aba de wave, clicar em **View->Wave** (deixar marcada a opção wave)
- A visão geral do ModelSim será parecida com essa (em vermelho está a aba de wave):



8. Adicionar sinais a wave

O próximo passo consiste em adicionar na wave os sinais que se deseja depurar. Para isso, selecione e arraste os três sinais da parte azul para a parte cinza. Note que ficará todo o caminho do sinal na parte cinza, o que deixa difícil ver seu nome, para suprimir o caminho, clique na parte destacada em vermelho na figura abaixo (dê um zoom para aproximar):

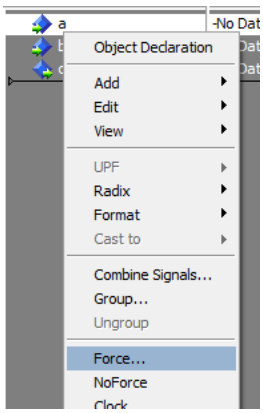


Salve a wave, clicando sob ela e depois através do atalho CTRL+S

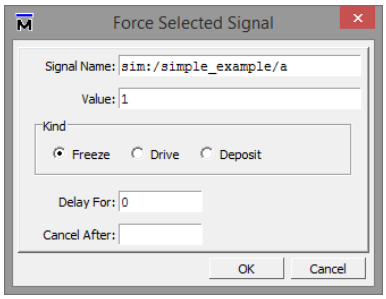
9. Configurando Valores para Sinais de Entrada

O próximo passo consiste em atribuir valores para os sinais de entrada. No nosso projeto temos dois sinais de entrada **a** e **b** e um de saída **o.** Vamos atribuir valores para os sinais **a** e **b** e ver se **o** se comporta como esperado.

- Dentro da aba wave, clique com o *botão direito* do mouse sob o sinal que deseja atribuir e selecione a opção ***Force***



- Selecione um valor para o sinal, neste exemplo, estamos atribuindo o valor (campo **Value**) de 1 para o sinal **a.** Para confirmar, clique em OK.

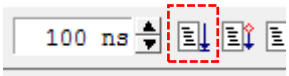


- Atribua o valor de 1 para **a** e 0 para **b**.

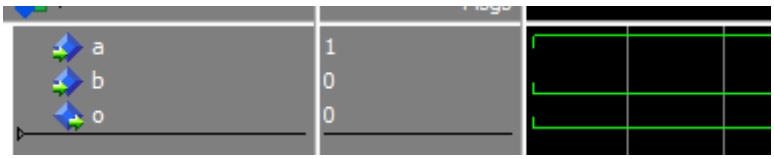
10. Simulando o Projeto

Com os sinais de entrada atribuídos a valores válidos, vamos iniciar a simulação.

- Use a barra superior de simulação. Defina o tempo da simulação no campo da esquerda (pode deixar como está). E clique no ícone de simulação, destacado pelo retângulo vermelho da figura abaixo.



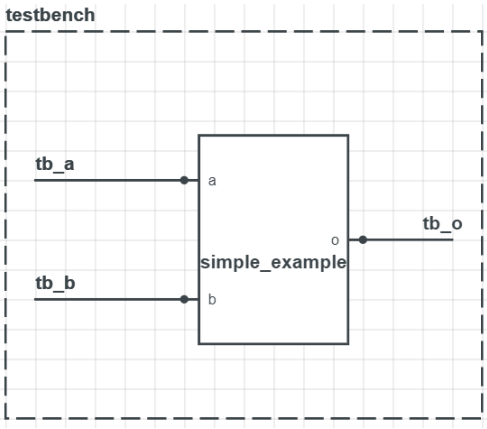
- Observe que a simulação iniciou. As linhas verdes indicam os valores dos sinais. Note que **a** está valendo 1 e **b** valendo 0. Como consequência, **o** sinal o ficou valendo 0 visto que ele representa a porta AND entre **a** e **b.**



- Altere o valor de **b=1** e veja como o comportamento de **o** se altera. Após clicar no ícone de simulação, clique na parte preta da wave e depois clique na tecla F, verá claramente que o valor de **o** se alterou para 1.

TestBench (Gerador de estímulos para os sinais)

Uma alterantiva automatizada para gerar valores para os sinais de entrada é utilizar um arquivo de testbench. Um testbench serve como arquivo de alto-nível (veja figura abaixo), que gera estímulos para o projeto e permite validar se o projeto está se comportando de forma correta. Veja na figura, o testbench é conectado ao projeto através das suas portas. Neste caso, o testbench está ligado com o projeto simple_example. O testbench possui sinais internos chamados *tb_a*, *tb_b* e *tb_o*. Estes sinais são ligados aos sinais externos projetados em simple_example. Desta forma, quando o testbench alterar seus valores (tb_a, tb_b), estes valores são transmitidos para simple_example. Note que tb_o está conectado a um sinal de saída de simple_example, por isso, não faz sentido o testbench alterar este valor, visto que ele é um sinal de saída e não de entrada em simple_example.



Nosso projeto implementa um simples módulo, o simple_example.vhd. Nosso projeto possui 3 portas, 2 de entrada (a e b) e uma de saída (o).

1. Criação de Testbench

- Crie um arquivo .vhd de testbench chamado *testbech.vhd*

Crie este arquivo dentro da pasta do projeto, ele deverá ficar junto com o simple_example.vhd

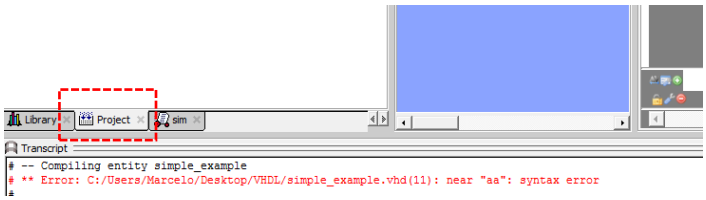
❖ Insira o seguinte código fonte no testbench.vhd

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

entity testbench is
end testbench;

architecture testbench of testbench is

    signal a_tb : std_logic;
    signal b_tb : std_logic;
    signal o_tb : std_logic;

begin

    simple_example : entity work.simple_example
        port map(
            a => a_tb,
            b => b_tb,
            o => o_tb
        );

    process
    begin
        a_tb <= '0';
        b_tb <= '1';
        wait for 10 ns;

        a_tb <= '1';
        b_tb <= '0';
        wait for 10 ns;

        a_tb <= '0';
        wait for 10 ns;

        wait; -- end of test
    end process;

end testbench;
```

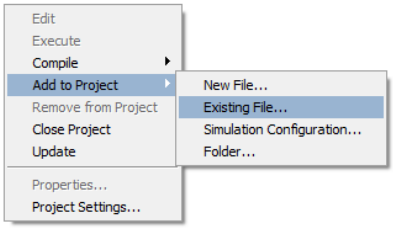
❖ Salve o arquivo (passo importante!)

2. Adicionar o Testbench ao projeto

- Voltando ao projeto no ModelSim clique na aba *Project*



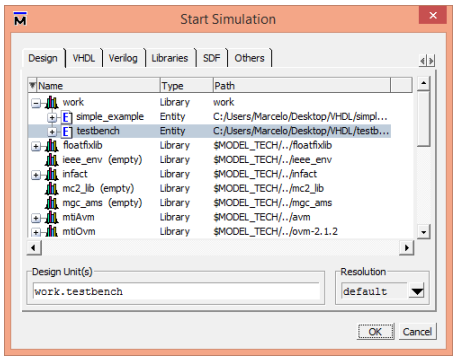
- Botão direito sob a região branca, *Add to Project -> Existing File*



- Selecionar o arquivo testbench.vhd dentro da pasta do projeto
- Compilar. Os dois arquivos deverão estar com o símbolo de ok verde

3. Simulação do Projeto com Testbench

- Clicar em Simulate -> Start Simulate (ou usar o ícone da barra de ícones)
- No momento de selecionar qual item simular, selecionar o arquivo *testbench* e não o arquivo do projeto

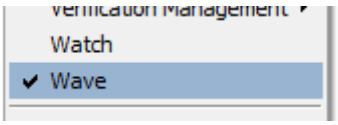


- Ao clicar em Ok a simulação deverá iniciar

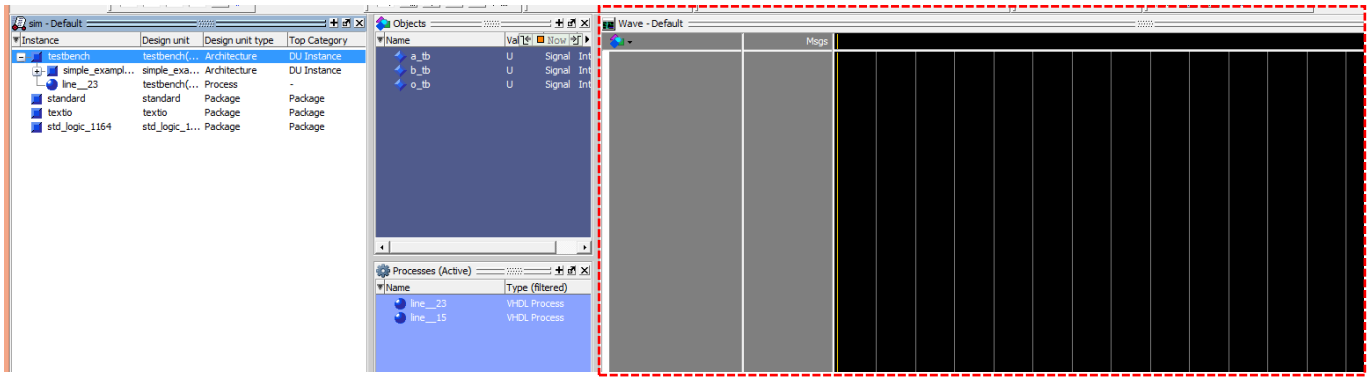
4. Simulação com forma de onda

A forma de onda permite acompanhar o estado dos sinais ao longo do tempo

- Para habilitar a simulação com waveform (chamaremos de wave), clicar em View e certifica-se que a opção *Wave* está marcada com um Ok. Caso naõ estiver, clicar em Wave.

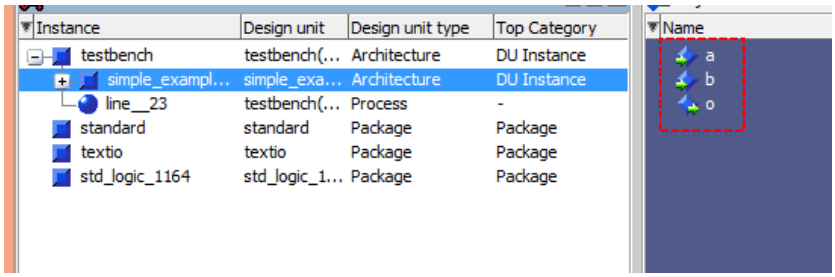


- A janela de wave deve abrir, destacada pelo retângulo vermelho abaixo:

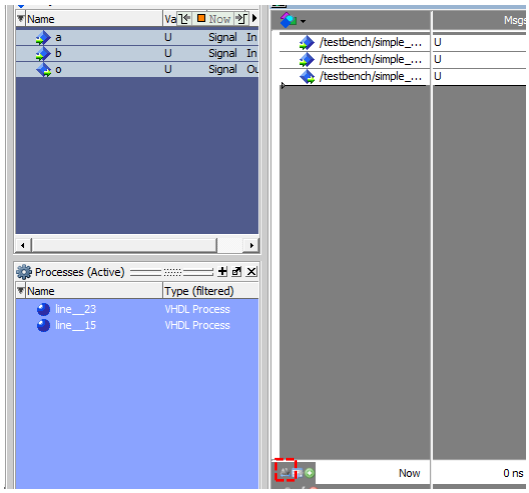


5. Adicionar sinais a wave

- Note que na janela da esquerda o ModelSim exibe a estrutura de módulos que estão dentro de testbench. Como testbench instanciou um módulo simple_example, este módulo está dentro do escopo do testbench (figura abaixo).
- Nosso objetivo será verificar o estado dos sinais *a e b*, que pertencem ao simple_example. Para isso, selecionar o módulo simple_example. Note que ao selecionar, irão aparecer os sinais a sua direita, na janela azul. Nosso objetivo é observar o comportamento destes sinais de acordo com os estímulos fornecidos pelo testbench.



- Selecione e arraste os três sinais da parte azul para a parte cinza. Note que ficará todo o caminho do sinal na parte cinza, o que deixa difícil ver seu nome, para suprimir o caminho, clique na parte destacada em vermelho na figura abaixo (dê um zoom para aproximar):



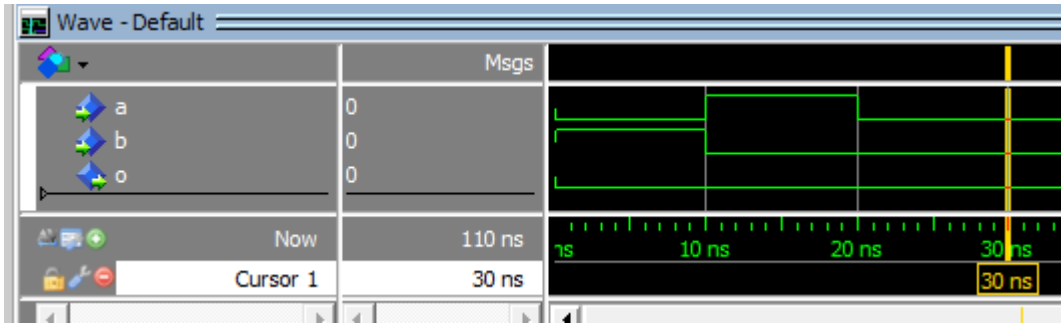
- Salva a wave, clicando sob ela e depois através do atalho CTRL+S

6. Iniciar simulação com a Wave

- O Último passo consiste em iniciar a simulação, para isso digite no terminal do ModelSim o comando

```
run 50ns    pode substituir o ns por: ps (picossegundos), caso queira um tempo menor, ou ms(milisegundos)
```

- Clique na parte preta da wave, e depois clique na tecla ‘f’ para expandir os sinais.
- Observe que os sinais agora estão com seus valores representados pela forma de onda, em verde.



Notar que os valores de **a** e **b** são gerados pelo testbench, uma vez que ele instancia o módulo simple_example. Veja abaixo parte do código do arquivo testbench.vhd. Esta parte corresponde a instanciação do módulo simple_example, e também a conexão das portas daquele módulo com os sinais internos do testbench.

```
simple_example : entity work.simple_example
        port map(
            a => a_tb,
            b => b_tb,
            o => o_tb
        );
```

Para explicar por que os sinais **a** e **b** possuem o comportamento apresentado na wave, basta olhar o código do testbench que altera seus sinais internos:

```
process
    begin
        a_tb <= '0';
        b_tb <= '1';
        wait for 10 ns;

        a_tb <= '1';
        b_tb <= '0';
        wait for 10 ns;

        a_tb <= '0';
        wait for 10 ns;

        wait; -- end of test
    end process;
```

Visto que os sinais do testbench **a_tb** e **b_tb** foram ligados com os sinais **a** e **b** de simple_example, todas as atribuições de valores para os sinais **a_tb** e **b_tb** irão ser enviadas para os sinais **a** e **b** dentro do módulo simple_example. Observando o código acima, notamos que:

- **a_tb** recebe inicialmente 0, **b_tb** recebe 1. Olhe para a figura da wave acima, note que o sinal de **a** está em 0 e **b** está em 1.
- Após 10 ns, **a_tb** recebe 1 e **b_tb** recebe 0. Olhe novamente para a figura da wave e perceba que os valores de **a** e **b** se alteraram de acordo com os de **a_tb** e **b_tb**.
- Após 10 ns, **a_tb** recebe 0, sendo assim o o valor de **a** que estava em 1 vai para 0.
- O último comando *wait* faz com que o processo de gerar estímulos fique esperando para sempre, portanto, os sinais de **a** e **b** não mudam mais.

Com isso, encerramos o tutorial, foi possível ter uma visão básica da ferramenta de simulação de circuitos integrados, Modelsim, além de entender como criar, compilar e simular um arquivo VHDL, juntamente com um testbench que gera estímulos e faz os sinais do projeto sobre teste serem alterados. Estes estímulos são fundamentais para o projetista depurar seu hardware, da mesma forma que em C adicionamos um scanf para poder ler um valor ou um printf e ver como seu algoritmo se comporta.

Infelizmente para projetar hardware o processo não é tão fácil quanto em C, vimos que é necessário criar um outro arquivo VHDL chamado testbench, e gerar nele uma série de estímulos que fazem o teste do nosso projeto.

Uso de script de comandos

O ModelSim permite usar arquivos com a extensão **.do** para fazer algo. Todos os comandos de mouse relativos a compilação e simulação se transformam em comandos de texto que são interpretados pelo terminal do ModelSim.

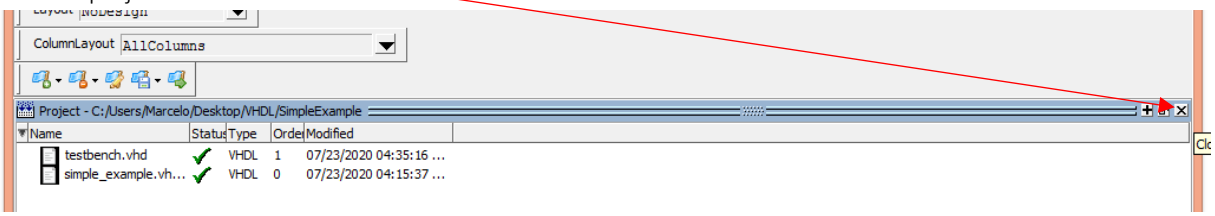
O modo mais prático de compilar um projeto é usando o arquivo de script com extensão **.do**. Geralmente chamamos este arquivo de **sim.do**, mas você pode escolher o nome que quiser para ele.

Para fazer os mesmos comandos acima basta usar o script abaixo:

```
#Cria a biblioteca work
vlib work
vmap work work

#Compila os dois modulos
vcom -work work simple_example.vhd
vcom -work work testbench.vhd

#Simula o arquivo testebench
vsim -novopt -t 1ns +notimingchecks work.testbench

#Carrega os sinais na forma de onda
add wave -noupdate /testbench/simple_example/a
add wave -noupdate /testbench/simple_example/b
add wave -noupdate /testbench/simple_example/o
radix hex

#Simula por 50 ns
run 50ns
```

1. Usando o arquivo .do no ModelSim

- Salve os comandos acima em um arquivo chamado sim.do.
- Crie uma outra pasta, por exemplo C:\VHDL_projects\script_test
- Cole dentro desta pasta o arquivo sim.do, o testbench.vhd e o simple_example.vhd do projeto anterior
- Feche o ModelSim
- Abra o ModelSim
- Feche o projeto atual no ModelSim



- No terminal, vá até a nova pasta que você criou digitando o comando (exemplo do meu caso):

```
cd C:/VHDL_projects/script_test/
```

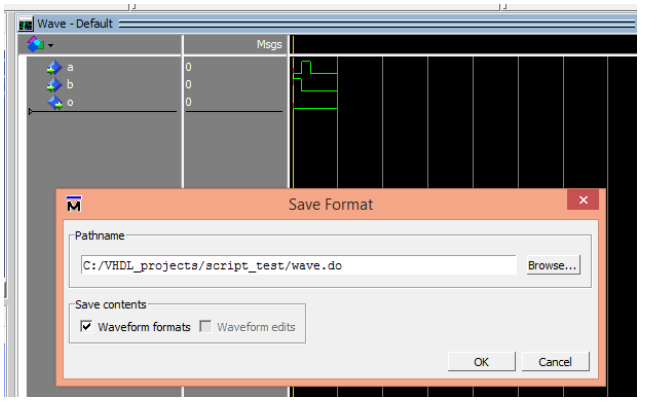
- Certifique-se que os 3 arquivos estão lá dentro

```
dir

resultado:
#
# 24/07/2020  14:03    <DIR>          .
# 24/07/2020  14:03    <DIR>          ..
# 24/07/2020  13:58               313 sim.do
# 23/07/2020  16:15               304 simple_example.vhd
# 23/07/2020  16:35               661 testbench.vhd
```

- Execute o comando:

```
do sim.do
```

- ❖ Confira se a simulação ocorreu conforme esperado, se não houve nenhum erro (mensagem vermelha) no terminal do ModelSim, e se a simulação está exibindo os valores dos sinais na wave da mesma forma que exibiu no modo manual que fizemos anteriormente.
- ❖ **NOTA:**
  - Para adicionar um novo arquivo na compilação do seu projeto, basta adicionar uma nova linha em **sim.do** onde é feita a compilação do projeto.
  - **Sempre faça a compilação primeiro do módulo que estiver no mais baixo nível.** No nosso caso, o módulo de mais baixo nível é o simple_example uma vez que o testbench instancia ele.
- ❖ **DICA DE WAVEFORM**:
  - Você não precisa adicionar todos os sinais da wave descrevendo um por um tal como feito em sim.do.
  - Uma forma mais prática consiste em, na primeira vez que for simular, carregar os sinais através da interface gráfica. Tal como feito nos passos anteriores. Em seguinte, clicar sob a janela de wave do ModelSim e executar o comando **ctrl+s,** o qual irá salvar a wave.
  - Salve a wave.do dentro da pasta onde está o **sim.do**
  - Modifique o **sim.do**, lá na parte de adição de sinais na waveform, removendo os sinais manuais e carregando o arquivo wave.do pelo arquivo sim.do. O arquivo sim.do ficará da seguinte forma:



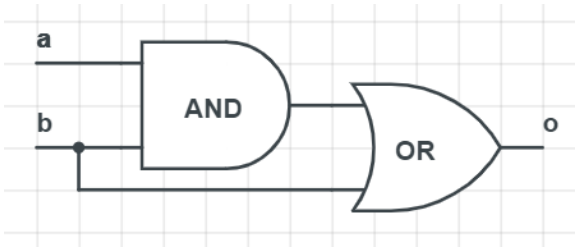

```
... parte anterior se mantém igual...

#Carrega os sinais na forma de onda
do wave.do
radix hex

... parte posterior se mantém igual...
```

Exercícios

1. Altere o testbench para que os valores de **a_tb e b_tb** sejam alterados após um intervalo de 20ns em vez de 10ns (modificando o comando *wait*). Compile, simule e verifique através da forma de onda se os valores mudaram de acordo com o esperado.

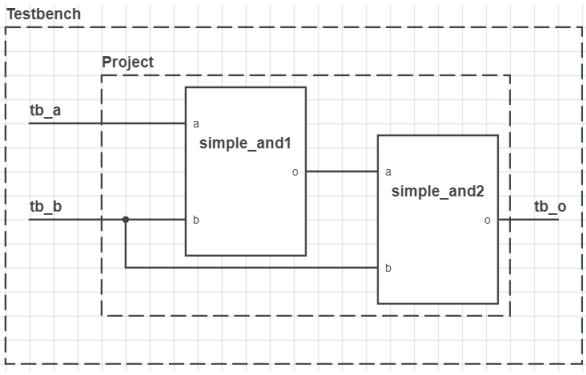
2. Crie um novo módulo, muito similar ao simple_example.vhd. No novo módulo, chamado simple_or.vhd use o mesmo código de simple_example, porém, altere a porta lógica AND do módulo simple_example, do exemplo anterior, por uma porta OR. Note que você precisa trocar todas as menções ao nome simple_example, por simple_or, tanto dentro de testbench.vhd quanto no próprio arquivo simple_and.vhd. Compile, simule e verifique através da forma de onda se o módulo *simple_and* está representando na porta *'o'* o valor esperado.

3. Implemente dentro de *simple_example* o seguinte circuito:



4. Use o módulo *simple_and* anteriormente para fazer o seguinte exercício:
   - Crie duas cópias de *simple_and*, necessariamente, cada cópia deverá ter um nome diferente, poderá ser *simple_and1* e *simple_and2*. Altere também a menção ao nome *simple_and* dentro do código VHDL de cada cópia, por exemplo, trocando tudo que possui o nome *simple_and* por *simple_and1*.

- A ideia é criar uma organização representada pela figura abaixo, onde tb_a e tb_c são conectadas a e b de simple_and1. Já em simple_and2 tem a sua porta a conectada a saída de simple_and1 e sua porta b conectada a tb_b.



- Note que a saída de o de simple_and1 será conectactada a entreda a de simple_and2. Para fazer essa conexão, crie outro sinal em tb chamada tb_conn, ou outro nome que desejar. Este sinal deve conectar a saída de simple_and1 e a entrada a de simple_and2.