µProcessador 0 Instalação dos Softwares

Recomendação: use Linux (se não sabe usar, está mais do que na hora). Dá pra usar Windows, se quiser. Deve dar pra usar Mac...

É preciso instalar um compilador VHDL (*ghdl*) e um visualizador de formas de onda (*gtkwave*), no mínimo. As sugestões dadas são boas, acreditem.

Editor de Código

VHDL é texto puro, então qualquer porcaria serve, mas um bom editor ajuda muito.

Gostei do **SublimeText 3**, mas os acólitos do Bill Gates podem usar o **Notepad++** que é muito bom de forma geral. No Ubuntu, até o **gedit** (configurado adequadamente e com plugins[1]) pode ser agradável.

Outra opção é usar o **Sigasi** na versão *standalone free* (Sigasi Studio Starter Edition), pra qualquer Sistema Operacional. Realmente excelente, específico para VHDL. E pro pessoal *hardcore*, dizem que o **emacs** e o **vim** configurados pra VHDL são os melhores, mas eu quero distância.

Linux

Eu estou utilizando em casa o elementary OS, derivado do Ubuntu 18.04.4 LTS 64 bits, okay?

1. GHDL: Para instalar, baixe o pacote e abra (mais recente, v0.37 em https://github.com/ghdl/ghdl/releases) ou digite "ghdl" no google. A central de programas faz o resto (ou use *sudo dpkg --install*). No meu caso, baixei para Ubuntu com mcode.

Pode ser necessário instalar também *libgnat-4.9*[2] com *sudo apt-get install*[3] (um -f pode ser necessário depois) e talvez também *lib32z1-dev*[4].

2. GtkWave: com alguma sorte, *sudo apt-get install gtkwave* resolve a parada.

Caso contrário, pode ser casca. Vai em http://gtkwave.sourceforge.net/, leia o manual em pdf, baixe e descompacte e leia o arquivo *README.txt*, que tem uma seção sobre instalação no Ubuntu.

3. Teste: veja a seção posterior.

Windows

Testado no Windows 10 há algum tempo com a versão 0.31; versões recentes podem ter mudado um pouco (estou sem Windows funcional para testar neste momento em que escrevo).

1. GHDL: o típico é o seguinte:

1. Baixe e descompacte o arquivo para o mingw em https://github.com/ghdl/ghdl/releases/tag/v0.37 ou vá por "ghdl" no google e ache o caminho.

1 *Snippets, word completion, code comment, smart spaces, quick open* e tamanho 4 para as tabulações, com *snippet* para arquivo vazio.

2 Se acusar falta de *libgnat* durante os testes.

3 A libgnat 4.9 está disponível no Ubuntu 16.04, o Xenial, e não nos mais recentes... Dá pra incluir nos repositórios:

```
sudo nano /etc/apt/sources.list, adicionar as linhas (coloquei ao final):
deb http://us.archive.ubuntu.com/ubuntu xenial main universe
#deb http://us.archive.ubuntu.com/ubuntu xenial main universe
sudo apt update
sudo apt install libgnat-4.9
```

Nota: pode ser necessário **`apt-get install -f libgnat4.9`**, esse -f é "--fix-broken" que resolve alguns problemas de dependências que pode surgir.

4 Caso seja emitido um erro como "*/usr/bin/ld: cannot find -lz*". Caso dê algum problema, confira as dicas de:
http://sourceforge.net/p/umhdl/wiki/Installation%20-%20Linux/#possible-problems-in-linux-64-bits (as bibliotecas a instalar devem ser instaladas em ordem nesse caso, ok?)

2. Abra um terminal de linha de comando (um *shell*), entre na pasta com cd (por ex., se foi instalado na pasta "C:\ghdl", o comando deve ser *cd C:\ghdl\ghdl-0.37...*).
3. Execute os arquivos *batch* como comandos do terminal:
   set_ghdl_path.bat
   reanalyze_libs.bat
4. Teste a instalação com *ghdl -v* e depois *ghdl --dispconfig*

Desta forma, *toda vez que abrir um novo shell precisa executar o set_ghdl_path!* Para evitar isso, edite as variáveis de sistema:

- Coloque o caminho do ghdl.exe no caminho padrão[5] (no exemplo, adicione a linha *c:\ghdl\ghdl-0.37...\bin;* na variável PATH).
- Seguindo o mesmo esquema, crie uma nova variável de ambiente GHDL_PREFIX com o caminho para as libs (*c:\ghdl\ghdl-0.37...\lib*, por ex.)

Se você tiver problemas ou achar difícil, por favor dê um *feedback* ao professor.

2. GtkWave: vai lá no site
http://gtkwave.sourceforge.net/
Baixe a parada e descompacte (às vezes não descompacta...[6]). Aí baixe o *all_libs* e descompacte. Coloque o executável do *gtkwave* dentro da subpasta *bin* da pasta descompactada e, para executar, vá no terminal e use *full path*, como por exemplo:

"C:\Users\Juliano\Arq Comp\GTKWAVE\bin>gtkwave.exe" "..\..\o teste\porta_e_tb.ghw"

Ou então coloque este caminho no *path* padrão, como feito para o *ghdl*, e execute sem o *full path*:

gtkwave porta_e_tb.ghw

Testando a Instalação

Esta é uma versão do 1º laboratório sem os esclarecimentos. Caso você detecte algum problema neste teste, verifique primeiro se o erro não está descrito no site abaixo:

- Linux: http://sourceforge.net/p/umhdl/wiki/Installation%20-%20Linux/
  Também garanta que *libgnat-4.9* e *lib32z1-dev* estejam instalados, se não funcionar.
- Windows: http://sourceforge.net/p/umhdl/wiki/Installation%20-%20Windows/

1. Fonte VHDL: digite o abaixo num editor e grave como *porta_e.vhd*. As explicações estão no roteiro do microprocessador 1.

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

entity porta_e is
   port( in_a,in_b: in std_logic;
         a_e_b: out std_logic
   );
end entity;

architecture aporta_e of porta_e is
begin
   a_e_b <= in_a and in_b;
end architecture;
```

2. Testbench: digite o arquivo para testes abaixo e grave como *porta_e_tb.vhd*.

5 Procure *adding path to windows* no Google.
6 Neste caso, renomeie o gtkwave.exe.gz pra gtkwave.exe (caso ele indique que não é um gzip de verdade...).

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

entity porta_e_tb is
end entity;

architecture aporta_e_tb of porta_e_tb is
    component porta_e is
       port(   in_a,in_b: in std_logic;
             a_e_b: out std_logic
       );
    end component;
    signal a,b,e: std_logic;
begin
   uut: porta_e port map(in_a=>a,in_b=>b,a_e_b=>e);
   process
   begin
       a <= '0';
       b <= '0';
       wait for 50 ns;
       a <= '0';
       b <= '1';
       wait for 50 ns;
       a <= '1';
       b <= '0';
       wait for 50 ns;
       a <= '1';
       b <= '1';
       wait for 50 ns;
       a <= '0';
       b <= '0';
       wait;
   end process;
end architecture;
```

3. Compilação: a sequência de comandos abaixo deve ser executada num terminal, **um a um**, *sem mensagens de erro.* Caso haja mensagens, verifique erros de digitação ou cópia.

ghdl -a porta_e.vhd
ghdl -a porta_e_tb.vhd
ghdl -r porta_e_tb --wave=porta_e_tb.ghw

Note que não se inclui o ".vhd" no 4º comando.

4. Simulação: a visualização das formas de onda finais pode ser feita com o comando abaixo.

gtkwave porta_e_tb.ghw

Note que elas não aparecem a princípio; para vê-las, você deve:

- Expandir a *treeview* que está no painel superior esquerdo (diz SST nele);
- Clicar na entidade *porta_e_tb* na árvore;
- Selecionar todos os sinais que aparecem ali (são *a, b* e *e*) e clicar *Insert;*
- Ir ao menu *Time => Zoom* e escolher *Zoom Full*;
- As formas de onda devem aparecer no painel preto.

Outras Alternativas

Parece uma excelente ideia usar uma IDE que edite, compile e simule, tudo dentro do meio ambiente, não é mesmo?

Não é.

Tanto o **Xilinx ISE** quanto o **Altera Quartus II** são IDEs assim, mas são pesadíssimos e lentos. O Quartus em particular é uma fonte de problemas bastante desagradáveis para o usuário casual, como nós... Os editores integrados são, surpreendentemente, uma droga, e os simuladores nada amigáveis nas versões atuais destes programas.

No entanto, para a gravação de FPGAs destes fabricantes, eles são a única possibilidade. Se você já tem contato ou hábito de usar um destes, o projeto pode ser feito nele sem problemas.

Para interessados em instruções de utilização do Quartus II, há uma versão alternativa do 1º laboratório de microprocessador, basta me pedir que eu envio.