Universidade Católica de Santos
Ciências da Computação

Professor: Walter S. Oliveira
walter.oliveira@unisantos.com
Disciplina: Circuitos Digitais

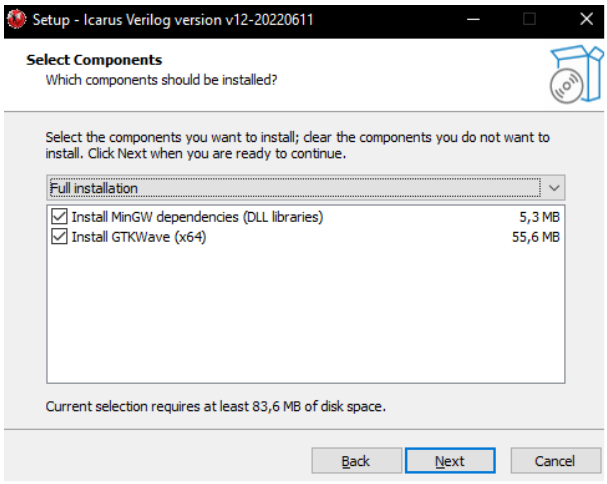
Prática 0: Instalação e configuração do Icarus, Visual Studio Code e GTKWave

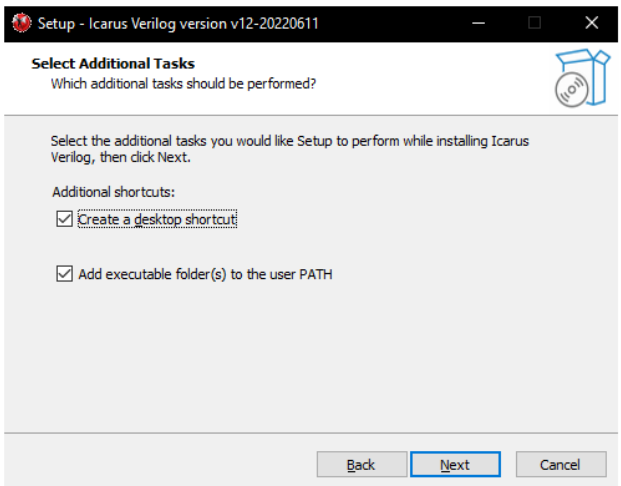
Nesta disciplina, utilizaremos o Icarus Verilog como compilador de Verilog e, com o auxílio do Visual Studio Code e GTKWave, iremos realizar todas as práticas desta disciplina.

① Instalação do Icarus e GTKWave

Primeiramente, vamos instalar o Icarus, compilador de Verliog gratuito. Para isso, procure Icarus Verilog no Google ou acesse https://bleyer.org/icarus/

Baixe e execute o instalador. Nas opções de instalação, selecione **Full Instalation** e adicione o executável no PATH.

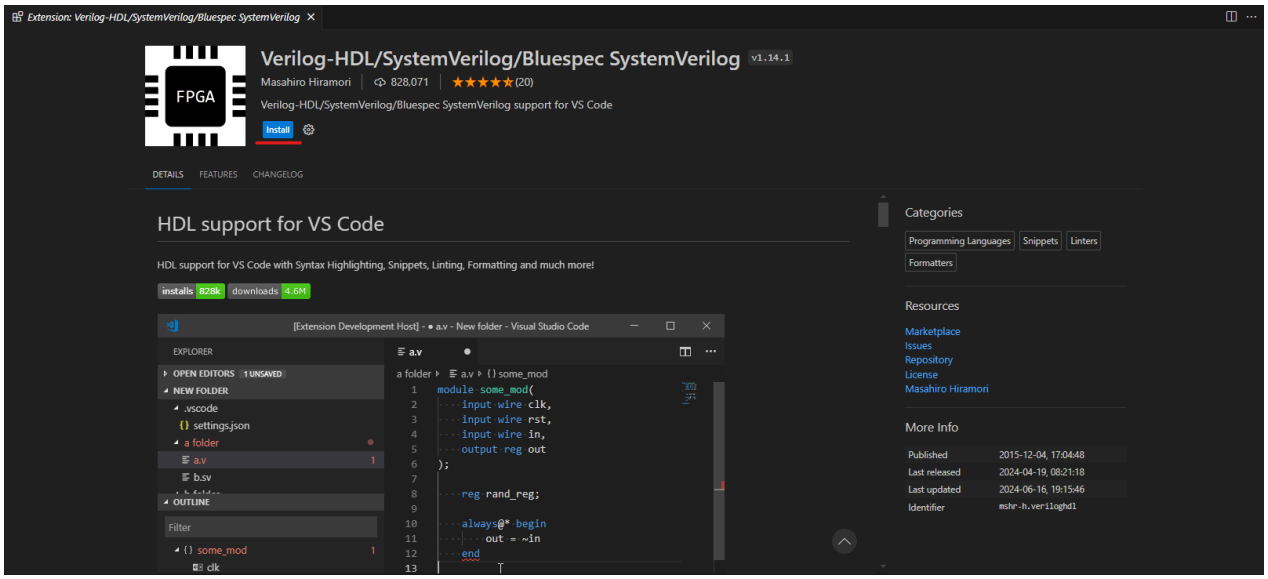




(2) Instalação do Visual Studio Code - VSC (Opcional)

Para o desenvolvimento dos códigos, será utilizado qualquer editor de texto. Devido a possuir extensão de formatação de sintaxe, IntelliSense, PowerShell integrado e outras funções, indica-se a utilização do VSC. Para isso, siga o passo-a-passo de acordo em https://code.visualstudio.com/download.

Com o VSC instalado, vá em Extensões ou digite Ctrl+Shift+X para abrir as extensões. Procure por Verilog e instale a extensão Verilog-HDL/SystemVerilog/Bluespec SystemVerilog.



(3) Programa de teste: Porta AND

Iremos criar um módulo de uma porta E para testar o funcionamento do Icarus e do GTKWave. Para isso, crie uma pasta em seu computador para armazenar os programas. No Visual Studio Code, crie um arquivo chamado `porta_e.v` e digite o seguinte código:

```
module porta_e(A, B, S);  // Criação do módulo da porta e

input A, B;  // Duas Entradas: A e B
output S;    // Uma saída: S

assign S = A & B; // A saída será a operação E entre as entradas A e B

endmodule // Fim do módulo
```

Criamos então um módulo chamado porta_e que possui duas entradas: A e B, e uma saída: S. A saída é a operação E entre as entradas A e B. Agora, precisamos de um arquivo para testar nosso módulo (normalmente chamado de *testbench*). Para isso, iremos criar um outro arquivo, chamado `porta_e_tb.v`. Nele, digite o seguinte código:

```
`timescale 1ns/1ns // Unidade de tempo
`include "porta_e.v" // Import do módulo que desenvolvemos anteriormente

module porta_e_tb; // Criacao do modulo de teste

        reg A, B; // Dois registradores para armazenar os valores
        wire S;  // Fio para conectar o resultado a saída
        porta_e uut(A, B, S); // Instancia da porta E a ser testada

        initial begin  // Inicio do teste
                $dumpfile("porta_e_tb.vcd");  // Criação do arquivo de resultado
                $dumpvars(0, porta_e_tb);    // Variáveis a serem guardadas

                A = 0;  // Seta o valor da variável A para 0
                B = 0;  // Seta o valor da variável B para 0
                #20;    // Aguarda 20 unidades de tempo (20ns)
                A = 0;
                B = 1;
                #20;
                A = 1;
                B = 0;
                 #20;
                A = 1;
                B = 1;
                #20;

                $display("Teste completo"); // Escreve uma mensagem de fim de teste
        end

endmodule
```

Seguimos os seguintes passos na criação do testebench:

1.) Começamos definindo a unidade de tempo do nosso tesbench. Neste caso, definimos a unidade de tempo de 1ns, com precisão de 1ns.

2.) Após isso, importamos o módulo que criamos anteriormente, o `porta_e.v`.

3.) Agora, vamos definir o módulo de teste do nosso código. Para isso, criamos duas variáveis do tipo reg, que armazenam os valores das variáveis A e B, e um wire, que conecta a saída S.

4.) Instanciamos um módulo da porta E com o nome uut (*unit under test*) e passamos como parâmetro as variáveis que criamos

5.) Vamos definir quais os valores para o teste do módulo na estrutura initial. As linhas dumpfile e dumpvars são necessárias para armazenar os resultados da simulação

6.) Após isso, declaramos quais os valores para as entradas e o tempo de execução de delay em função do timescale.

Com os dois arquivos criados, iremos criar o arquivo de resultado. Abra o Powershell na pasta onde os arquivos foram criados (No Visual Studio Code, digite Ctrl+Shift+‘) e digite o seguinte comando:

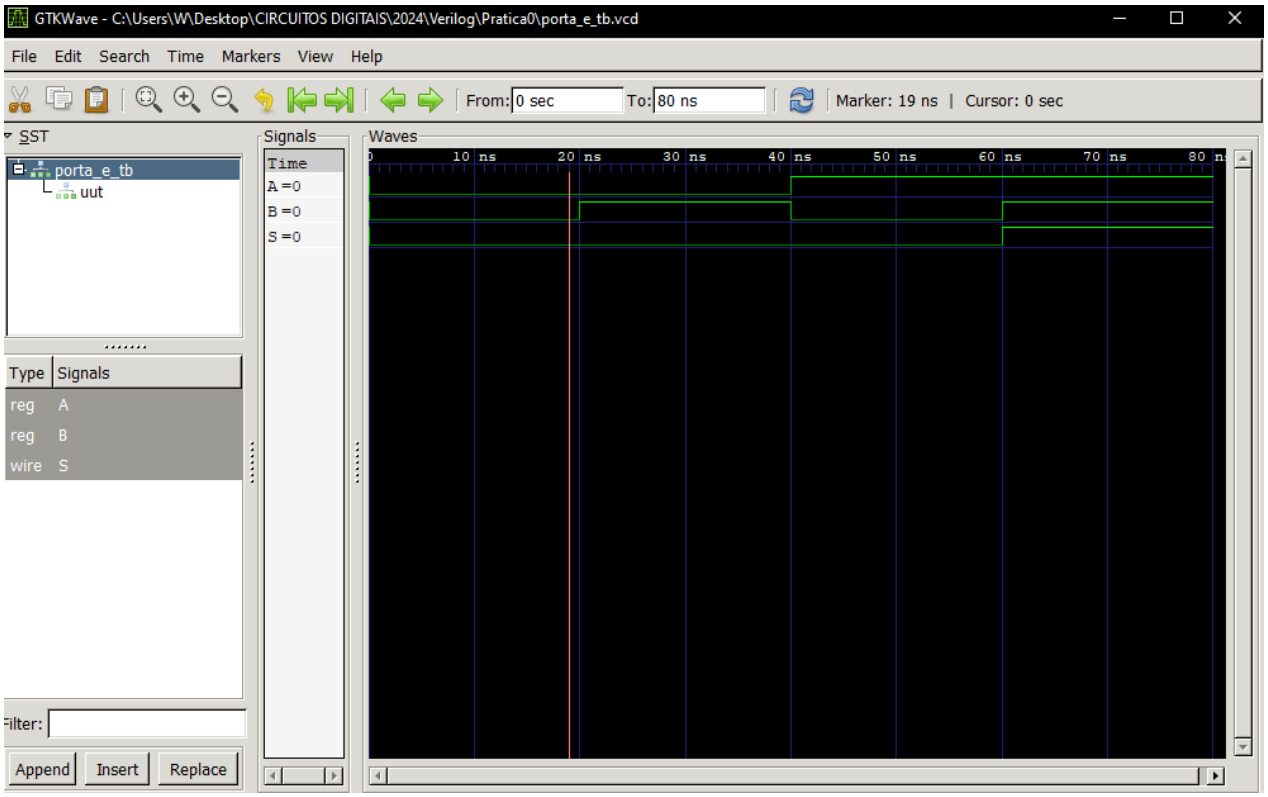
```
iverilog -o porta_e_tb.vvp porta_e_tb.v
```

Este comando `iverilog -o <resultado.vvp> <testebench.v>` gera um arquivo .vvp. Se o comando foi executado sem erros, na mesma pasta irá aparecer o arquivo `porta_e_tb.vvp`. Agora, iremos gerar o arquivo .vcd para ser aberto no GTKwave. Para isso, rode o seguinte comando:

```
vvp porta_e_tb.vvp
```

Este comando realiza a simulação do testbench. É possível visualizar no Terminal a mensagem `Teste Completo`, como escrevemos no final do nosso módulo, assim como a criação do arquivo `porta_e_tb.vcd`. Por fim, vamos abrir o GTKWave, digitando o `gtkwave` no terminal.

Com o GTKwave aberto, vá em File -> Open New Tab e selecione o arquivo `porta_e_tb.vcd`. Clicando no nome do módulo, aparecerá os três sinais do testbench (A, B e S). Selecione os três e clique em append. Na janela, verifique que o módulo está funcionando como uma porta E.



(4) Tarefas

Crie novos módulos para simular outras portas lógicas, como por exemplo a porta OU ou a XOU.

Teste as funcionalidade do GTKWave.