MINISTÉRIO DA EDUCAÇÃO
Universidade Federal de Ouro Preto
Instituto de Ciências Exatas e Aplicadas
Campus João Monlevade

Curso: Engenharia de Computação — 1º semestre de 2024
CSI509 – Organização e Arquitetura de Computadores II
Prof. Eduardo da Silva Ribeiro
Aluno: Vinicius Assis e Silva — **Matrícula: 18.2.6975**

iCarus Verilog

1. Testando Icarus Verilog e GTKWave

   Icarus Verilog (ou iVerilog) é um compilador Verilog e um simulador Verilog. Um compilador Verilog converte seus arquivos de origem Verilog em um arquivo netlist que representa o circuito ou comportamento lógico digital de seu código-fonte. O iVerilog não tem como alvo uma tecnologia digital específica, mas cria uma representação lógica genérica que pode ser simulada para validar a operação do circuito.

   Depois que um projeto for compilado com sucesso, o iVerilog poderá ser usado para simular o projeto. A saída da simulação pode ser um texto ou um arquivo .vcd (despejo de alteração de valor). O arquivo .vcd é um arquivo de texto que pode ser lido por ferramentas gráficas de exibição de forma de onda, como GTKWave, para permitir a criação de saída de simulação gráfica.

   - Criação do arquivo fonte Verilog e arquivo testbench Verilog.
     - Crie um arquivo de texto usando um editor de texto.
     - Digite o seguinte texto abaixo em top.v:

```
module my_and (in1, in2, out1);
input in1, in2;
output out1;
  assign out1 = in1 & in2;
endmodule

```

2. O código acima implementa uma porta AND simples. Estudaremos como construir descrições de circuitos Verilog em classe. Uma explicação rápida é que cada módulo Verilog é encapsulado por module / endmodule. A instrução module contém o nome do seu componente e uma lista de todas as portas de entrada e saída. As instruções de entrada e saída simplesmente identificam a direção do sinal. A instrução de atribuição cria a lógica real, atribuindo o AND lógico das duas entradas à saída.

   Abra a janela de terminal, alterne para o diretório que contém top.v e digite este comando: **iverilog top.v** . Se você vir um erro informando que o iVerilog não foi encontrado, é provável que você precise adicionar o iVerilog à variável PATH do Windows ou adicionar um link simbólico para Linux ou equivalente para IOS.

   Se nenhuma saída adicional aparecer no prompt de comandos, a compilação funcionou. Observe que o arquivo a.out foi gerado, esta é a saída da etapa de compilação do Verilog. Adicionando -o <nome do arquivo de saída desejado> ao comando iVerilog.

   Para simular um projeto em Verilog, você precisa criar um arquivo que contenha construções de linguagem específicas para simulação. Chamamos isso de arquivo testbench.

   Um arquivo testbench não é compilado em lógica, mas é usado para direcionar sinais para as entradas do seu design (in1 e in2 no design my_and) e para capturar a saída criada pelo seu design (out1 no módulo my_and).

Crie um arquivo testbench (novamente, deve ser um arquivo ASCII). Nomeie o arquivo tb.v, com o seguinte texto. Observe que o primeiro caractere é um apóstrofo inclinado, e não uma aspa simples. Na linha final $display, substitua o texto de saída mostrado pelo seu nome.

```
1  `timescale 1 ns / 100 ps
2  module tb();
3  reg a, b; wire c;
4  my_and U1 (.in1(a), .in2(b), .out1(c));
5  initial
6      begin
7          $dumpfile("output.vcd");
8          $dumpvars; $display("Starting simulation");
9          a = 0;
10         b = 0;
11         #10 a = 1;
12         #10 b = 1;
13         #10 a = 0;
14         #10 $display("Simulation ended.");
15             $display( " <Put your name here> " );
16         $finish;
17     end
18 initial
19     $monitor($time, " a = %b, b = %b, c = %b", a, b, c);
20 endmodule
21
```

O arquivo `testbench` tem muito mais coisas acontecendo, mas aqui está uma explicação rápida:

1. A direção da escala de tempo define as unidades de tempo na simulação. O primeiro valor de tempo é a unidade de tempo de simulação, enquanto o segundo valor de tempo define como um valor de tempo não inteiro ou cálculo será arredondado. Os valores de tempo são interpretados como valores de atraso `x 1` nanossegundo, com arredondamento feito para `100 ps` ou precisão de `0,1 ns`.

2. Observe as instruções `module`/`endmodule`. Note também que um `testbench` não possui portas de entrada ou saída (a lista de portas está vazia).

3. O módulo `my_and` é instanciado no `testbench`. Outra maneira de dizer isso é que você está testando uma instância ou uma ocorrência do seu módulo `my_and`. É possível criar quantas instâncias de um módulo desejar em um projeto Verilog. Neste caso, atribuímos um sinal ou variável para conduzir `in1 (a)`, `in2 (b)` e para capturar o valor de `out1 (c)`.

4. Qualquer código dentro de um bloco `initial` será executado em ordem e executado apenas uma vez. Neste primeiro bloco `initial`, definimos um arquivo de saída (`$dumpfile`) para capturar a saída da simulação para posterior exibição no `GTKWave`. Dizemos ao simulador para capturar todos os sinais (`$dumpvars`) e então exibir uma mensagem definida pelo usuário (`$display`). Em seguida, são atribuídos valores que orientam as entradas `my_and`. A sintaxe `#10` significa atrasar 10 unidades de tempo antes de executar a próxima linha. `$finish` instrui a simulação a terminar.

5. A tarefa do sistema `$monitor` (que deve estar em seu próprio bloco `initial`) produzirá a string formatada contendo os valores do sinal sempre que um dos sinais mudar de valor. Esta saída é direcionada para `STDOUT` (normalmente seu terminal ou janela de comando).

Simule o design `my_and`:

a. Na janela de comando, digite estes comandos:

   i. `iverilog tb.v top.v`

   ii. `vvp a.out > out.txt` (a saída da simulação será gravada no arquivo)

   iii. `vvp a.out` (a saída da simulação será gravada no console)

   iv. Se você renomeou o arquivo `a.out` usando a opção `-o` no `iVerilog`, use o novo nome de arquivo no lugar de `a.out`.

b. Como antes, obter um prompt de comando sem saída significa sucesso. Se você encontrar erros, corrija-os até que a simulação seja bem-sucedida.

c. Observe que dois novos arquivos são criados, `out.txt` e `output.vcd`.

d. Abra `out.txt`, você deverá ver isto:

```
Starting simulation

  0 a = 0, b = 0, c = 0

 10 a = 1, b = 0, c = 0

 20 a = 1, b = 1, c = 1

 30 a = 0, b = 1, c = 0

Simulation ended.  <your name here>
```

   O primeiro valor é um marcador de data/hora, seguido pelos valores de entrada e pelo valor de saída.

Exibir saída de simulação em GTKWave:

a. No prompt de comando, digite este comando: `gtkwave output.vcd`.

b. A janela `GTKWave` será aberta. No painel superior esquerdo você verá o nome do módulo `testbench (tb)`. Clicar no sinal de mais mostrará o nome da instância do seu módulo em teste, `U1`.

c. Clique em `U1`, selecione cada um dos sinais `in1`, `in2` e `out1` e clique no botão `Insert`. Isso moverá os sinais para a área do visualizador de forma de onda. Se necessário, clique no botão `Zoom Fit`.

O `GTKWave` é um visualizador de formas de onda, uma ferramenta que pode ler arquivos `.vcd` e converter essas informações em uma visualização gráfica. Se você consultar `tb.v`, verá que `in1` e `in2` receberam o valor `0` em `t=0`. Em `t=10ns`, `in1` foi definido como `1`, em `t=20ns` `in2` foi definido como `1` e em `t=30ns`, ambas as entradas receberam o valor `0`. Observe que o valor de `out1` é o AND lógico das duas entradas. Finalmente, a simulação termina em `t=40ns`.