



UNIVERSIDADE FEDERAL DE SANTA CATARINA

Laboratório: Simulação com Testbenches

EEL5105 – Circuitos e Técnicas Digitais

Objetivos

- Obter uma **visão geral** acerca do uso de **testbenches** para verificação/simulação em **VHDL**.
- Foco em **conceitos básicos e introdutórios**.
- Fazer estudos de caso visando fixar os conceitos estudados.

Introdução

- **Verificação** de funcionalidade por simulação é algo essencial no fluxo de projeto de sistemas digitais.
- **Testbench**: mecanismo para simulação com **VHDL**
 - Ideia: gerar **padrões de entrada** para teste e observar **saídas** para verificar a correta operação de um sistema.
 - Em linhas gerais, **testbench** é um arquivo **VHDL** sem entradas/saídas, onde o componente a ser testado (**DUT** - *device under test*) é declarado/instanciado e onde são descritos os padrões de entrada e teste.
 - Resultados são tipicamente observados a partir de **gráficos de formas de onda** ou mensagens/alertas.

Introdução

- Nesta aula: **simulação lógica básica**.
- Usaremos **GHDL** para simulação (dentro do **Emulador Web**) e **GTKWave** para visualização dos resultados.
 - **Testbench** deve estar em **usertest.vhd**.
- Alternativas:
 - **ModelSim**;
 - **EDAPlayground**.

Introdução

- **Exemplo 1: Testbench** para um **Half Adder**
 - Tabela/Código do **Half Adder**:

A	B	Sum	Cout
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

halfadder.vhd

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

entity halfadder is port (
    A,B: in std_logic;
    Sum,Cout: out std_logic );
end halfadder;

architecture hadder of halfadder is
begin
    Sum <= A xor B;
    Cout <= A and B;
end hadder;
```

Introdução

- Exemplo 1: **Testbench** para um **Half Adder**
 - **Testbench** básico:

usertest.vhd

```
library ieee;
use ieee.std_logic_1164.all;
entity usertest is
end usertest;
architecture tb of usertest is
    signal SA, SB, SSum, SCout : std_logic;
    component halfadder is port (
        A,B: in std_logic;
        Sum,Cout: out std_logic );
    end component;
begin
    DUT : halfadder port map (A => SA, B => SB, Sum => SSum, Cout => SCout);
    SA <= '0', '1' after 20 ns, '0' after 40 ns;
    SB <= '0', '1' after 10 ns, '0' after 20 ns, '1' after 30 ns;
end tb;
```

Introdução

- Exemplo 1: **Testbench** para um **Half Adder**
 - Testbench** básico:

usertest.vhd

```
library ieee;
use ieee.std_logic_1164.all;
entity usertest is
end usertest;
architecture tb of usertest is
    signal SA, SB, SSum, SCout : std_logic;
    component halfadder is port (
        A,B: in std_logic;
        Sum,Cout: out std_logic );
    end component;
begin
    DUT : halfadder port map (A => SA, B => SB, Sum => SSum, Cout => SCout);
    SA <= '0', '1' after 20 ns, '0' after 40 ns;
    SB <= '0', '1' after 10 ns, '0' after 20 ns, '1' after 30 ns;
end tb;
```

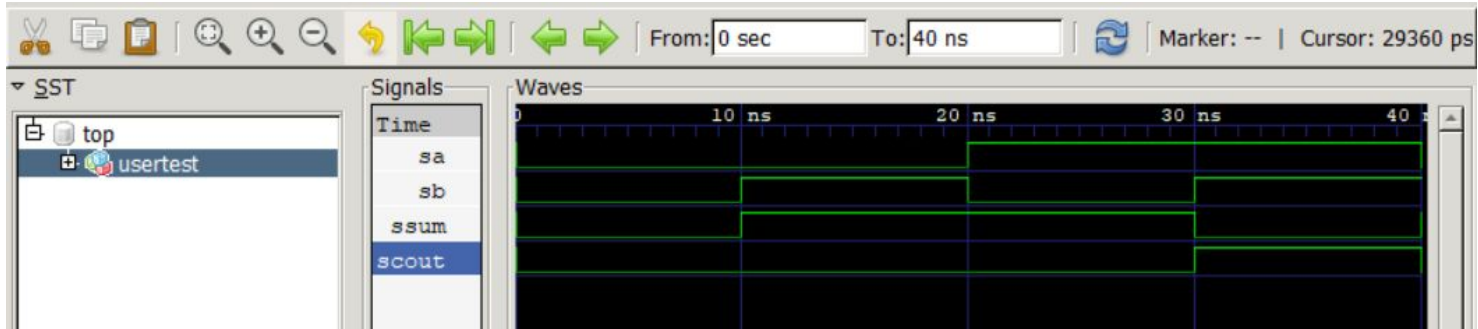
Entity sem ports.

Estímulos para o DUT.

Obs.: **after** não é sintetizável e, portanto, não pode ser usado em projetos.

Introdução

- Exemplo 1: **Testbench** para um **Half Adder**
 - Resultado da simulação:



A	B	Sum	Cout
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Introdução

- Exemplo 2: **Testbench** para um **Half Adder**
 - **Testbench** básico com **process**:

usertest.vhd

```
library ieee;
use ieee.std_logic_1164.all;
entity usertest is
end usertest;
architecture tb of usertest is
    signal A, B, Sum, Cout : std_logic;
    component halfadder is port (
        A,B: in std_logic;
        Sum,Cout: out std_logic );
    end component;
begin
    DUT : halfadder port map (A => A,
                               B => B,
                               Sum => Sum,
                               Cout => Cout);
```

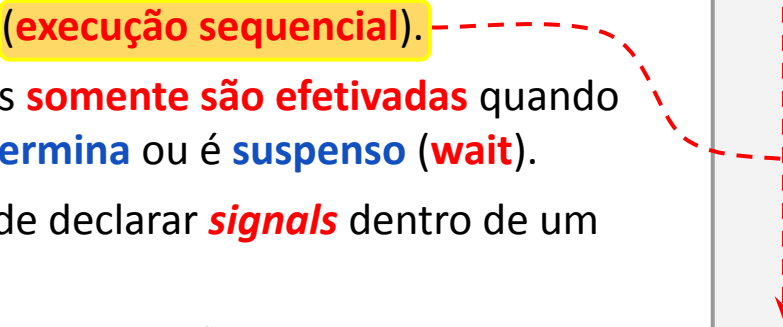
```
process
    constant period: time := 10 ns;
begin
    A <= '0'; B <= '0';
    wait for period;
    A <= '0'; B <= '1';
    wait for period;
    A <= '1'; B <= '0';
    wait for period;
    A <= '1'; B <= '1';
    wait for period;
    wait;
end process;

end tb;
```

Introdução

- Noções iniciais de **processos** em **VHDL**
 - Permitem modelar **lógica sequencial** em **VHDL**.
 - **Atribuições** são executadas na ordem que elas aparecem (**execução sequencial**).
 - Atribuições **somente são efetivadas** quando processo **termina** ou é **suspenso** (**wait**).
 - Não se pode declarar **signals** dentro de um processo.
 - Algumas estruturas só podem ser usadas dentro de processos (e.g., **if...then...else**).
 - Mais informações em outra aula...

```
process
  constant period: time := 10 ns;
begin
  A <= '0'; B <= '0';
  wait for period;
  A <= '0'; B <= '1';
  wait for period;
  A <= '1'; B <= '0';
  wait for period;
  A <= '1'; B <= '1';
  wait for period;
  wait;
end process;
```



Introdução

- Exemplo 2: **Testbench** para um **Half Adder**
 - Testbench** básico com **process**:

```
library ieee;
use ieee.std_logic_1164.all;
entity usertest is
end usertest;
architecture tb of usertest is
    signal A, B, Sum, Cout : std_logic;
    component halfadder is port (
        A,B: in std_logic;
        Sum,Cout: out std_logic );
    end component;
begin
    DUT : halfadder port map (A => A,
                             B => B,
                             Sum => Sum,
                             Cout => Cout);
```

Entity sem ports.

```
process
    constant period: time := 10 ns;
begin
    A <= '0'; B <= '0';
    wait for period;
    A <= '0'; B <= '1';
    wait for period;
    A <= '1'; B <= '0';
    wait for period;
    A <= '1'; B <= '1';
    wait for period;
    wait;
end process;

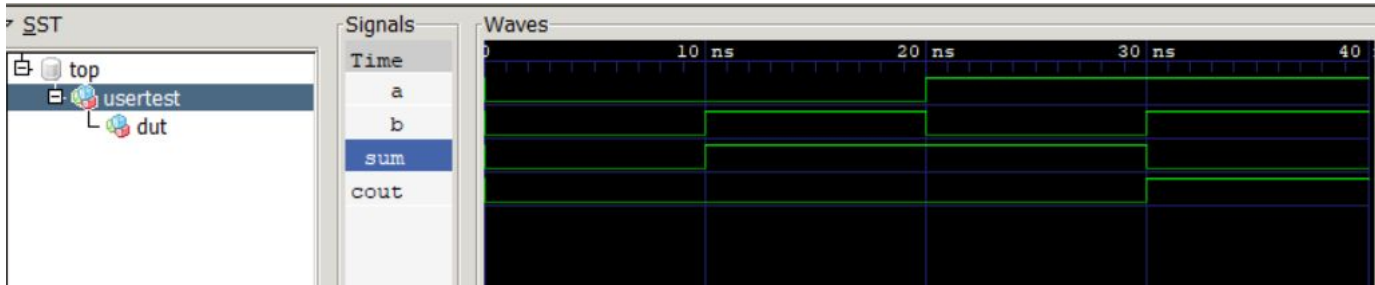
end tb;
```

Constante definida no contexto de um **process**.

Wait for não é sintetizável, portanto não é usado para projeto, mas em **testbenches** sim.

Introdução

- Exemplo 2: **Testbench** para um **Half Adder**
 - Resultado:



A	B	Sum	Cout
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Introdução

- **Exemplo 3: Testbench** para um **Half Adder Incorreto**
 - Tabela/Código do **Half Adder Incorreto**:

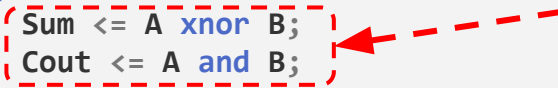
A	B	Sum	Cout
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

halfadder.vhd

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

entity halfadder is port (
    A,B: in std_logic;
    Sum,Cout: out std_logic );
end halfadder;

architecture hadder of halfadder is
begin
    Sum <= A xnor B;
    Cout <= A and B;
end hadder;
```



Introdução

- Exemplo 3: **Testbench** para um **Half Adder Incorreto**
 - **Testbench** com **Assert/Report**:

```
library ieee;
use ieee.std_logic_1164.all;
entity usertest is
end usertest;
architecture tb of usertest is
    signal A, B, Sum, Cout : std_logic;
    component halfadder is port (
        A,B: in std_logic;
        Sum,Cout: out std_logic );
    end component;
begin
    DUT : halfadder port map (A => A,
                               B => B,
                               Sum => Sum,
                               Cout => Cout);

    process
        constant period: time := 10 ns;
        begin
```

```
        A <= '0'; B <= '0';
        wait for period;
        assert ((Sum = '0') and (Cout = '0'))
        report "Failed for 00." severity error;
        A <= '0'; B <= '1';
        wait for period;
        assert ((Sum = '1') and (Cout = '0'))
        report "Failed for 01." severity error;
        A <= '1'; B <= '0';
        wait for period;
        assert ((Sum = '1') and (Cout = '0'))
        report "Failed for 10." severity error;
        A <= '1'; B <= '1';
        wait for period;
        assert ((Sum = '0') and (Cout = '1'))
        report "Failed for 11." severity error;
        wait;
    end process;
end tb;
```

Introdução

- Exemplo 3: **Testbench** para um **Half Adder Incorreto**

- Resultado:

A	B	Sum	Cout
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Simulation Process

Simulating...

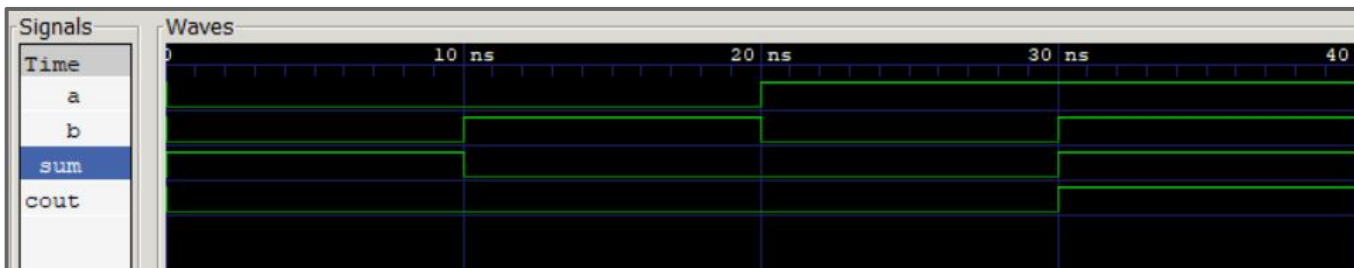
usertest.vhd:23:13:@10ns:(assertion error): Failed for 00.

usertest.vhd:27:13:@20ns:(assertion error): Failed for 01.

usertest.vhd:31:13:@30ns:(assertion error): Failed for 10.

usertest.vhd:35:13:@40ns:(assertion error): Failed for 11.

Simulation finished.



Introdução

- **Exemplo 4:** **Testbench** para **Half Adder** com algumas novidades
 - Ver próximo slide...


```

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity usertest is
end usertest;
architecture tb of usertest is
    signal SA, SB, SSum, SCout : std_logic;
    signal cnt : std_logic_vector(1 downto 0) := "00";
    component halfadder is port (
        A,B: in std_logic;
        Sum,Cout: out std_logic );
    end component;
begin
    DUT : halfadder port map (A => SA, B => SB, Sum => SSum, Cout => SCout);
    SA <= cnt(1);
    SB <= cnt(0);
    process
        constant period: time := 10 ns;
        begin
            for k in 1 to 8 loop
                wait for period;
                cnt <= cnt + '1';
            end loop;
            wait;
        end process;
end tb;

```

Signal que irá armazenar uma contagem de 2 bits (00,01,10,11). O "00" depois de := é a definição do valor inicial de **cnt**.

Essa **soma (+)** faz o incremento de **cnt**. Requer importação de **std_logic_unsigned**.

MSB de **cnt** conectado em **A** e **LSB** em **B** (signals **SA** e **SB** são usados para fazer essa conexão).

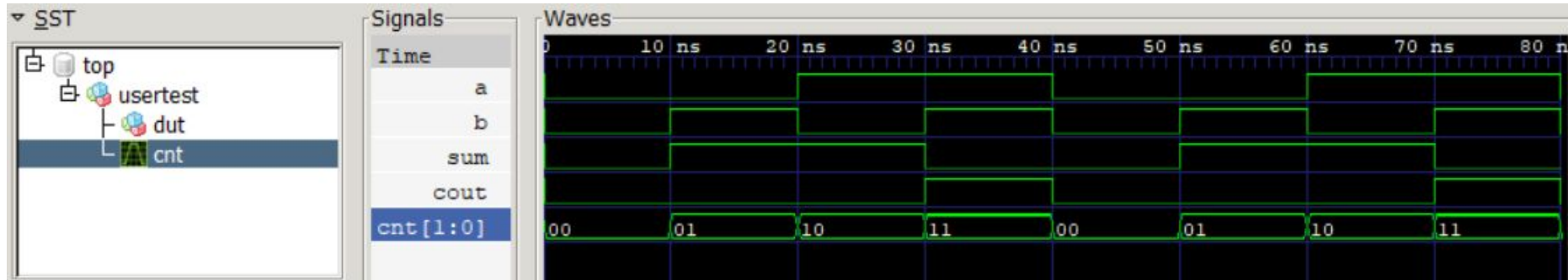
for de 1 a 8. Esse **for** não é o mesmo do **for/generate** e só pode ser usado dentro de **process**.

Esse **wait** termina a simulação após o **for**. Sem ele, **process** iria ser reiniciado pois ele é **cíclico**.

Introdução

.....

- Exemplo 4: **Testbench** para **Half Adder** com algumas novidades
 - Resultado:

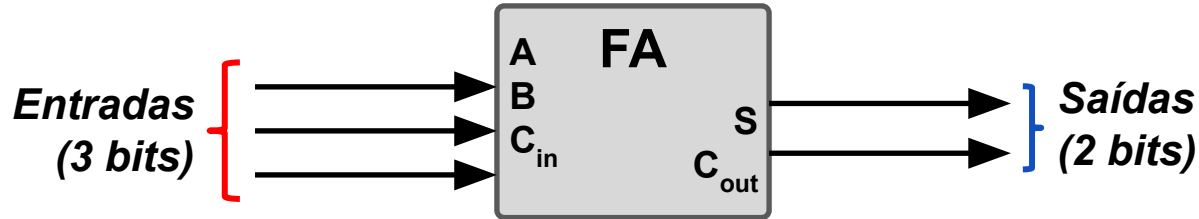


Tarefas

- Ideia:
 - Usar o **Emulador Web** (**GHDL**) para fazer a simulação.
 - Ver os resultados no **GTKWave**.

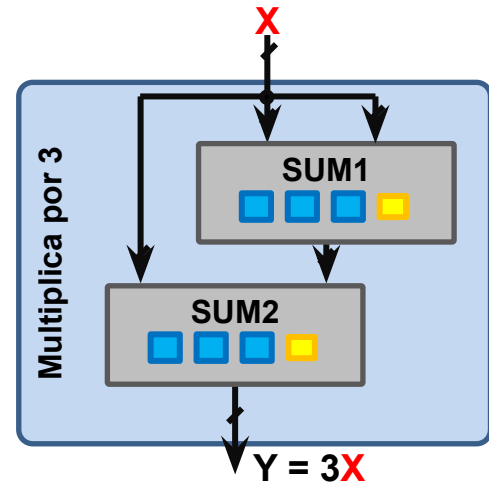
Tarefa 1

- Crie um **testbench** para um **full adder** de tal forma que todas as possíveis condições de entrada sejam testadas. Faça a simulação usando seu **testbench** e veja os resultados no **GTKWave**.



Tarefa 2

- Crie um **testbench** para o **Multiplicador por 3** implementado em aulas anteriores, faça a simulação de tal circuito, e procure entender melhor por que a saída desse circuito não é correta para toda faixa de valores possíveis de entrada.
- **Dica:** observe o sinal que sai de **SUM1** e vai para entrada de **SUM2**.



Tarefa 3

- Faça o **projeto** e depois a **simulação** de um multiplexador 8 x 1 com entrada de **enable**.
 - Não é necessário fazer o projeto em nível de portas lógicas (use **with/select** ou **when/else**).
 - Para **enable = 0**, saída deve ficar sempre igual a 0.

