

Sistemas Digitais

P01 – Introdução VHDL+GHDL+GTKWave
Elaborando nossas primeiras simulações

VHDL

descrevendo um hardware

VHDL

- VHSIC Hardware Description Language

- VHSIC

- Very High Speed Integrated Circuit

- VHDL

Very High Speed Integrated Circuit Hardware
Description Language

- Tradução Livre:

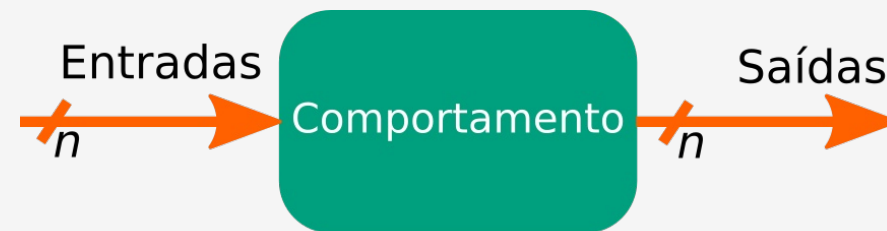
Linguagem de Descrição de Hardware de Circuito
Integrado de Alta Velocidade

VHDL

- Linguagem de Descrição de Entidades Digitais

- Entidade Digital

- É um Circuito Elétrico Digital



Circuito (genérico)

- Interface

- Sinais de Entrada e Saída do Circuito
 - É como o Circuito se comunica com o ambiente externo

- Comportamento

- Sinais, Portas Lógicas e/ou Entidades que processam as Entradas para gerar as Saídas

VHDL

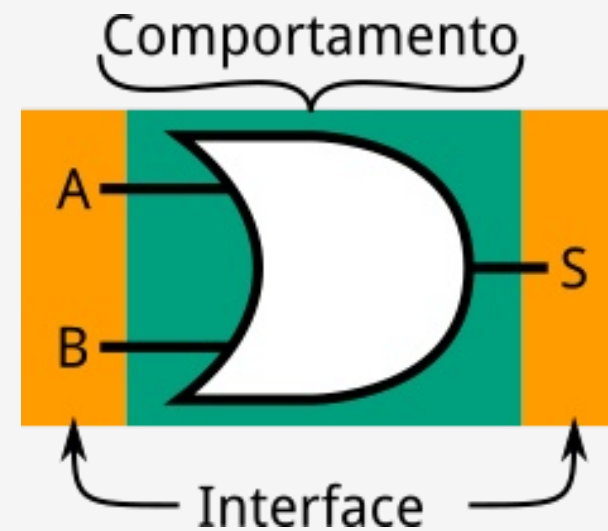
- **Linguagem** de Descrição de Entidades Digitais
 - O usuário **descreve** os Sinais de Entrada e Saída e o Comportamento de uma Entidade Digital

VHDL – Exemplo OR

- Exemplo didático prático:
 - Uma Porta Lógica OR de 2 entradas
 - Interface
 - Entradas: Sinal **A** e Sinal **B**
 - Saída: Sinal **S**
 - Comportamento
 - Função Lógica OR

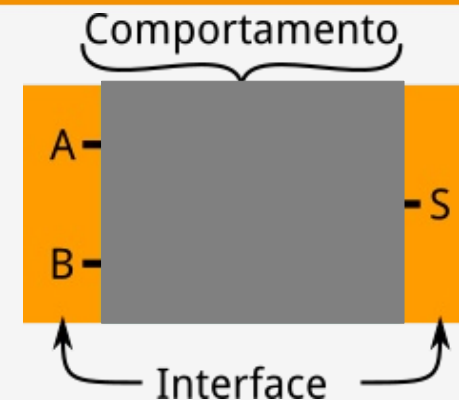
$$S = A + B$$

Operador de atribuição
em Lógica Booleana



VHDL – Exemplo OR

- Interface
 - Entradas: Sinal **A** e Sinal **B**
 - Saída: Sinal **S**



- Em VHDL:

```
4 entity pl_or_2 is
5     port(
6         A : in  bit;
7         B : in  bit;
8         S : out bit;
9     );
10 end pl_or_2;
```

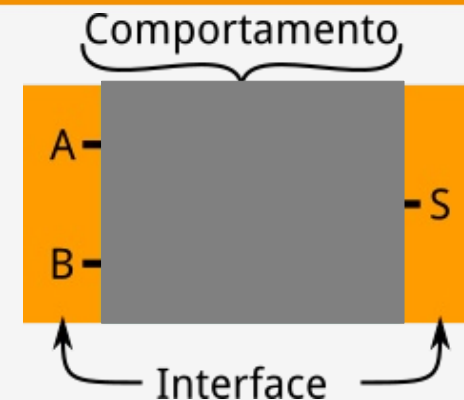
Observação importante:

- Todas as imagens com código contêm número das linhas, porém, não mantém consistência entre imagens do mesmo código. Alterações foram realizadas para melhor ajustar aos slides!

VHDL – Exemplo OR - Interface

- Interface

- Entradas: Sinal **A** e Sinal **B**
- Saída: Sinal **S**



- Em VHDL, uma interface é descrita como:

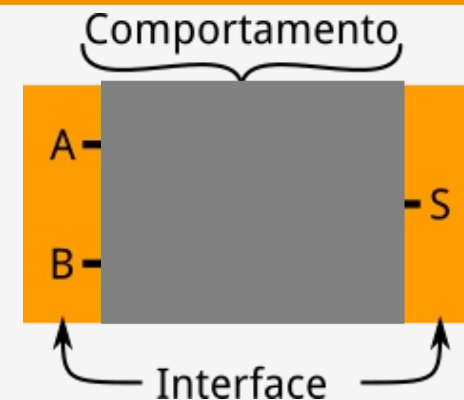
```
1 entity <nomeEntidade> is port(<interface> [<interface>]); end <nomeEntidade>;
```

- <nomeEntidade>: é o nome do Circuito Digital descrito
- <interface>: é o sinal de Entrada/Saída
 - [<interface>]: várias interfaces podem ser descritas (opcional)
- Caracteres e Palavras reservadas:
 - “entity” “is” “port” “end” “;” “(” “)”

VHDL – Exemplo OR - Interface

- Interface

- Entradas: Sinal **A** e Sinal **B**
- Saída: Sinal **S**



- Nome da Entidade OR com 2 entradas:

```
1 entity <nomeEntidade> is
```

```
end <nomeEntidade>;
```

- Será descrito como:

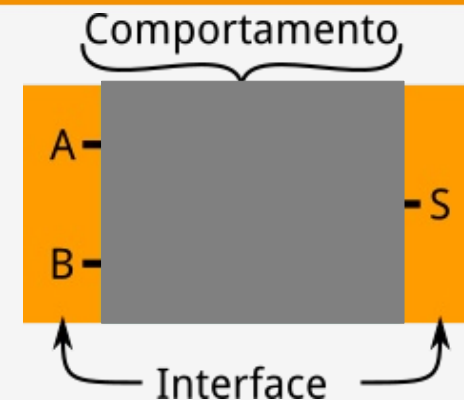
```
6 entity pl_or_2 is ... end pl_or_2;
```

- pl → porta lógica
- or → função/objetivo
- 2 → 2 entradas

VHDL – Exemplo OR - Interface

- Interface

- Entradas: Sinal **A** e Sinal **B**
- Saída: Sinal **S**



- Interfaces de Entidade OR com 2 entradas:

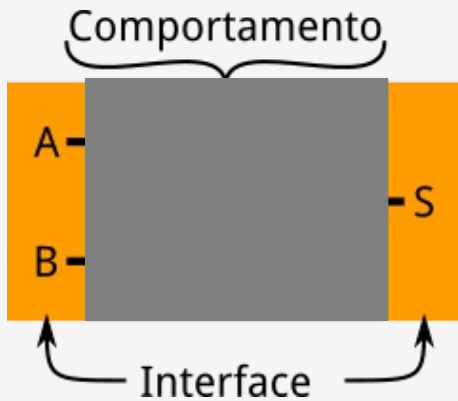
```
port(<interface> [<interface>]);
```

- Entradas A e B e Saída S

- Sinais Digitais → bit

```
8 ... port(A : in bit; B : in bit; S : out bit); ...
```

- Concluindo a Interface:



Nomes dos Sinais
da Interface

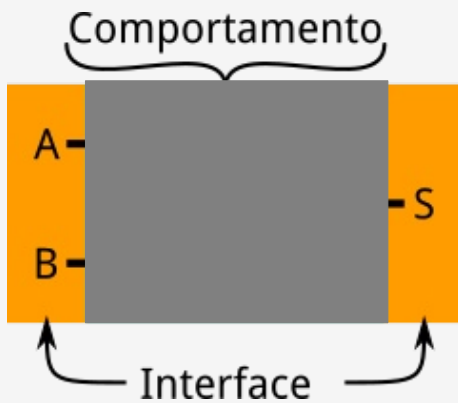
```
4 entity pl_or_2 is
5     port(
6         A : in  bit;
7         B : in  bit;
8         S : out bit;
9     );
10 end pl_or_2;
```

Nome da
Entidade

tipo **bit**

Direção do Sinal
- Entrada: **in**
- Saída: **out**

- Arquivo `pl_or_2.vhdl`
 - Até o momento

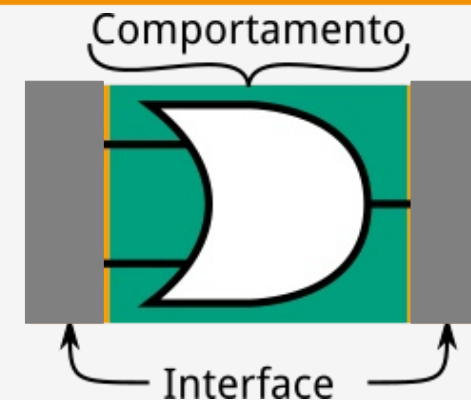


```
pl_or_2.vhdl x
1 entity pl_or_2 is
2     port(
3         A : in  bit;
4         B : in  bit;
5         S : out bit
6     );
7 end pl_or_2;
```

VHDL – Exemplo OR

- Comportamento
 - Função Lógica OR

$$S = A + B$$



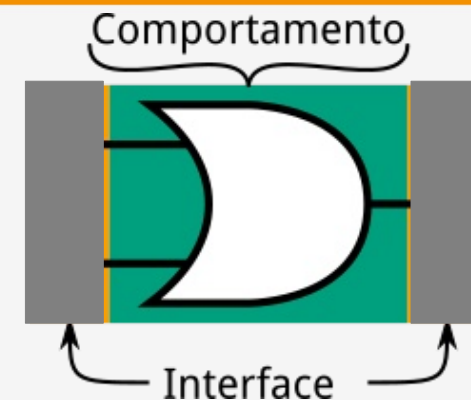
- Em VHDL:

```
9 architecture comportamento of pl_or_2 is
10 begin
11     S <= A or B;
12 end architecture comportamento;
```

VHDL – Exemplo OR

- Comportamento
 - Função Lógica OR

$$S = A + B$$



- Em VHDL, o Comportamento é descrito como:

```
10 architecture <nomeComportamento> of <nomeEntidade> is  
11 <subEntidades>    -- opcional  
12 <Sinais>         -- opcional  
13 begin  
14     <códigoComportamento>  
15 end comportamento;
```

Comentários em Código
são ignorados pelo compilador

VHDL – Exemplo OR

- Em VHDL, o Comportamento é descrito como:

```
10 architecture <nomeComportamento> of <nomeEntidade> is  
11 <subEntidades>    -- opcional  
12 <Sinais>          -- opcional  
13 begin  
14     <códigoComportamento>  
15 end comportamento;
```

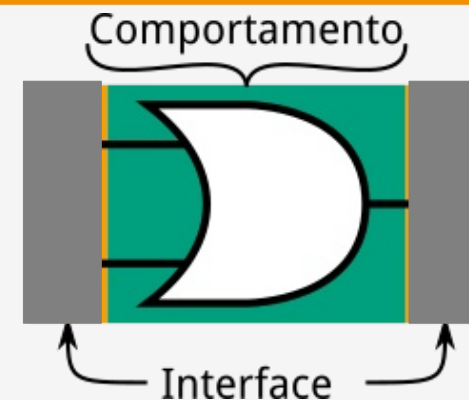
assinatura do
Comportamento

- <nomeComportamento>: é o nome do Comportamento
- <nomeEntidade>: nome da Entidade
- <subEntidades>: Entidades usadas na Entidade principal
- <Sinais>: conexões entre Entradas, Saídas e subEntidades
- <códigoComportamento>: onde a magia acontece!

VHDL – Exemplo OR

- Comportamento
 - Função Lógica OR

$$S = A + B$$



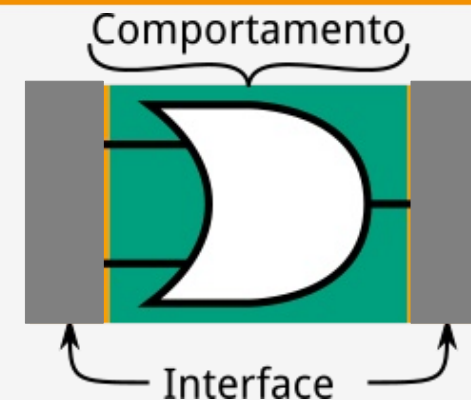
- assinatura do comportamento da Entidade pl_or_2 é:

```
17 architecture comportamento of pl_or_2 is  
18 -- <subEntidades>   -- não utilizado  
19 -- <Sinais>  
20 begin  
21     <códigoComportamento>  
22 end comportamento;
```


VHDL – Exemplo OR

- Comportamento
 - Função Lógica OR

$$S = A + B$$



- O código do comportamento da Entidade `p1_or_2` é:

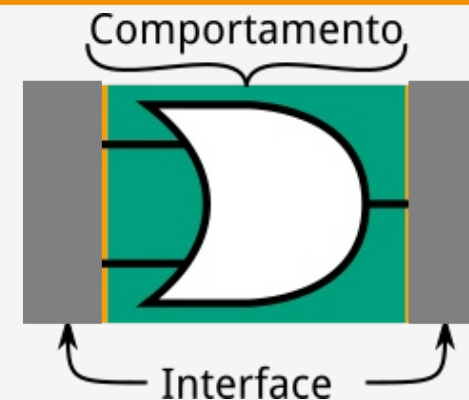
```
10 begin
11     S <= A or B;
12 end comportamento;
```

- VHDL aceita todas as funções lógicas estudadas nas aulas anteriores e resumidas no arquivo

VHDL – Exemplo OR

- Comportamento
 - Função Lógica OR

$$S = A + B$$



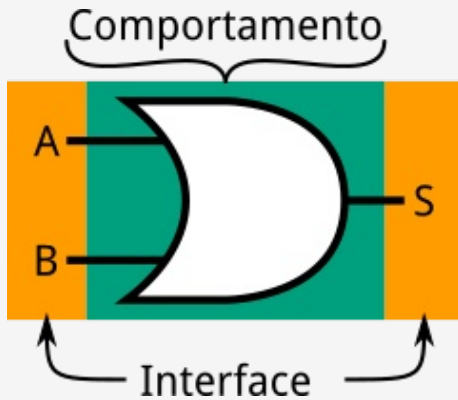
- Notação de atribuição

```
11 S <= A or B;
```

- Usa-se o conjunto de símbolos "<="
- Exemplo:
 - Saída S recebe o resultado de $A + B$

- Arquivo `pl_or_2.vhdl`

- Concluído

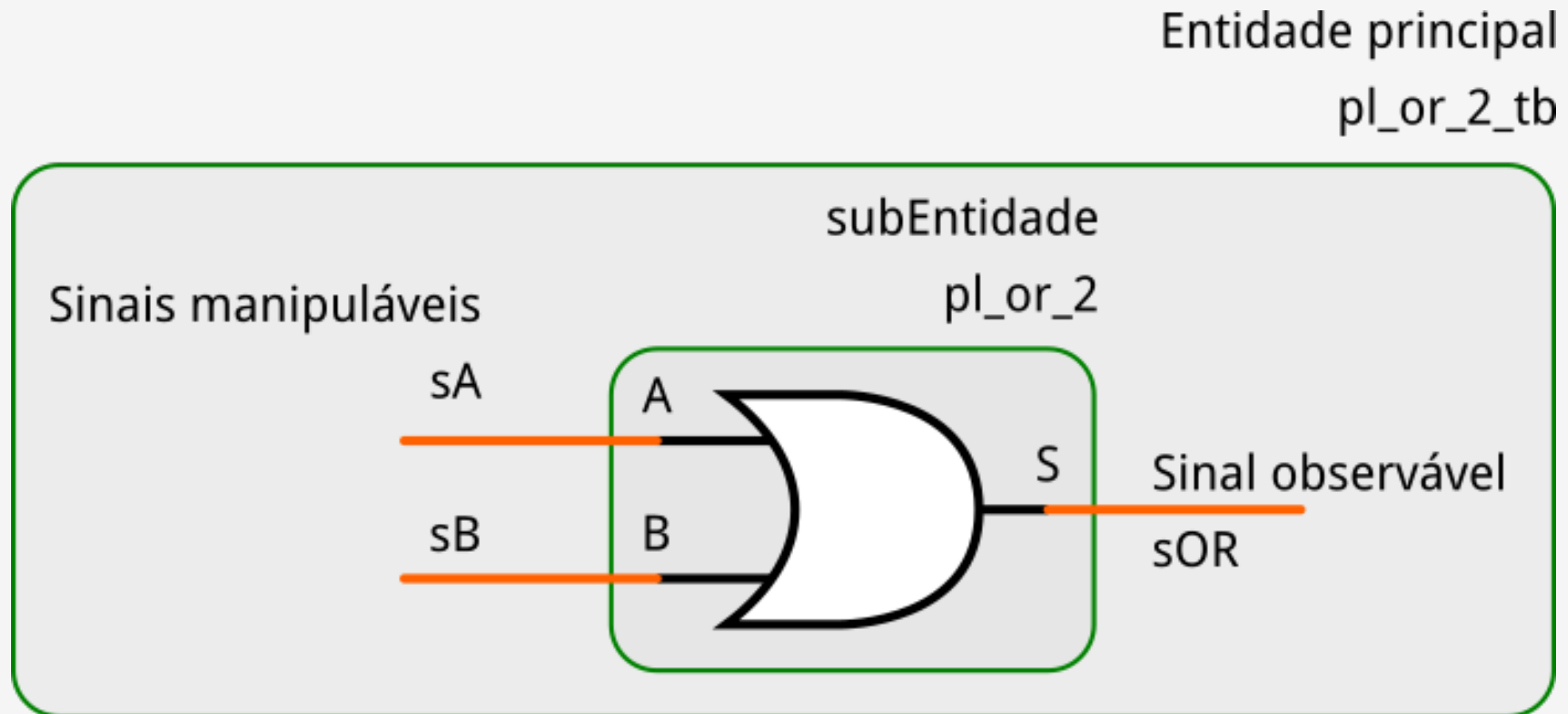


```
pl_or_2.vhdl x
1 entity pl_or_2 is
2     port(
3         A : in  bit;
4         B : in  bit;
5         S : out bit;
6     );
7 end pl_or_2;
8
9 architecture comportamento of pl_or_2 is
10 begin
11     S <= A or B;
12 end comportamento;
```

VHDL – Exemplo OR – Simulação

- E para **simular** esse exemplo?
 - É necessário manipular as entradas **A** e **B** e observar a saída **S**.
 - Devemos criar uma “bancada de testes” (testbench)
 - É uma Entidade que manipula e observa outras Entidades
 - Contém <subEntidades> e <Sinais>
 - Não se comunica com o ambiente externo
 - Interface é vazia

- Diagrama conceitual do Testbench para p1_or_2
 - Entidade p1_or_2tb
 - Contém Sinais **sA**, **sB** e **sOR**



VHDL – Exemplo OR – Simulação

- Entidade **testbench** para **pl_or_2**
 - Sem comunicação com ambiente
 - Nome: **pl_or_2_tb**
 - Interface vazia

```
1 entity pl_or_2_tb is  
2     -- Entidade vazia  
3 end pl_or_2_tb;
```

VHDL – Exemplo OR – Simulação

- Comportamento da `pl_or_2_tb`

```
10 architecture <nomeComportamento> of <nomeEntidade> is  
11 <subEntidades>      -- opcional  
12 <Sinais>            -- opcional  
13 begin  
14     <códigoComportamento>  
15 end comportamento;
```

- <nomeComportamento>: tb
- <nomeEntidade>: pl_or_2_tb
- <subEntidades>: usa Entidade pl_or_2
- <Sinais>: contém sA, sB e sOR
- <códigoComportamento>: ...

VHDL – Exemplo OR – Simulação

- Comportamento da `pl_or_2_tb`

```
10 architecture <nomeComportamento> of <nomeEntidade> is
```

- <nomeComportamento>: tb
- <nomeEntidade>: pl_or_2_tb

```
5 architecture tb of pl_or_2_tb is
```


VHDL – Exemplo OR – Simulação


- Comportamento da `pl_or_2_tb`

```
10 architecture <nomeComportamento> of <nomeEntidade> is  
11 <subEntidades>      -- opcional  
12 <Sinais>            -- opcional
```

- <subEntidades>: usa Entidade `pl_or_2`

```
1 entity pl_or_2 is  
2   port(  
3     A : in  bit;  
4     B : in  bit;  
5     S : out bit;  
6   );  
7 end pl_or_2;
```

```
5 architecture tb of pl_or_2_tb is  
6   -- <subEntidade>  
7   component pl_or_2 is  
8     port(  
9       A : in  bit;  
10      B : in  bit;  
11      S : out bit;  
12    );  
13 end component;
```

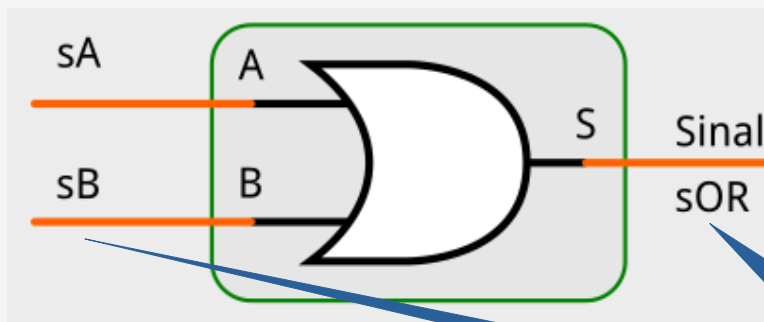


VHDL – Exemplo OR – Simulação

- Comportamento da `pl_or_2_tb`

```
10 architecture <nomeComportamento> of <nomeEntidade> is  
11 <subEntidades>      -- opcional  
12 <Sinais>             -- opcional
```

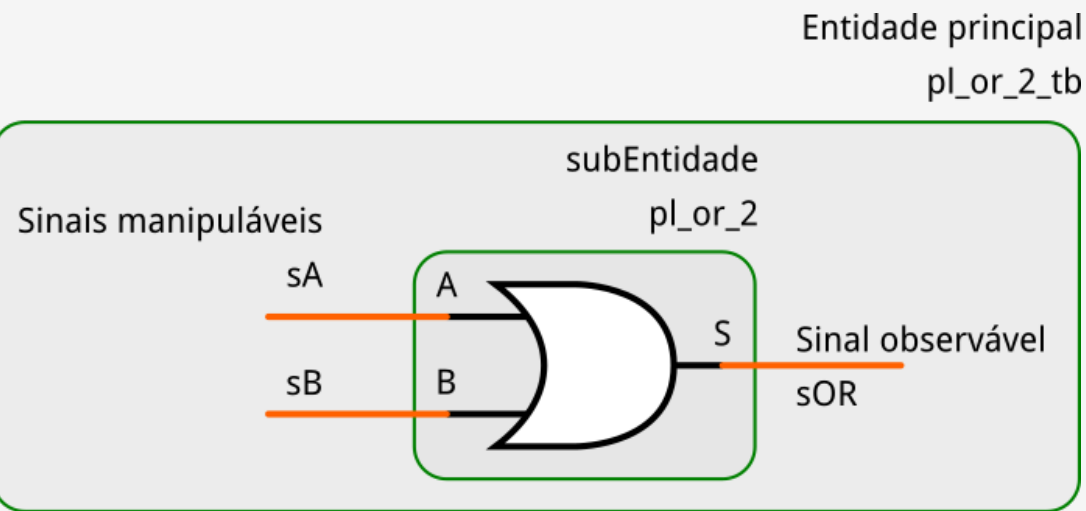
- <Sinais>: contém sA, sB e sOR



```
13      end component;  
14  
15      -- <Sinais>  
16      signal sA, sB : bit;  
17      signal sOR    : bit;
```

mesmo <tipo>

- Arquivo `pl_or_2_tb.vhdl`
 - Até o momento



```
pl_or_2_tb.vhdl x
1 entity pl_or_2_tb is
2     -- Entidade vazia
3 end pl_or_2_tb;
4
5 architecture tb of pl_or_2_tb is
6     -- <subEntidade>
7     component pl_or_2 is
8         port(
9             A : in  bit;
10            B : in  bit;
11            S : out bit
12        );
13     end component;
14
15     -- <Sinais>
16     signal sA, sB : bit;
17     signal sOR   : bit;
```

VHDL – Exemplo OR – Simulação

- Comportamento da `pl_or_2_tb`
 - A arte de conectar fios!

```
13 begin
14     <códigoComportamento>
15 end comportamento;
```

- Importante:
 - Todo o código inserido entre `begin` e `end` do comportamento é executado em `paralelo`.

VHDL – Exemplo OR – Simulação

- Comportamento da `pl_or_2_tb`

```
13 begin
14     <códigoComportamento>
```

- Primeiro passo: **instanciar** as <subEntidades>
 - Instanciar: informar que a <subEntidade> será usada e como esta se conecta com os <Sinais>
 - Duas formas:
 - Instanciação com <mapeamento> explícito
 - Instanciação com <mapeamento> ou Posicional

VHDL – Exemplo OR – Simulação

- Como é feito uma **instanciação**?

```
24 <nomeUnidade>: <subEntidade> port map(<mapeamento>);
```

- <nomeUnidade>: nome para unidade instanciada
 - É possível instanciar diversas vezes a mesma <subEntidade>
 - Todas com <nomeUnidade> diferentes
- <subEntidade>: é a <subEntidade> a ser instanciada
- <mapeamento>: conexão da Interface com os <Sinais>

VHDL – Exemplo OR – Simulação

- **Instanciação** de pl_or_2 em pl_or_2_tb

```
6  -- <subEntidade>
7  component pl_or_2 is
8      port(
9          A : in  bit;
10         B : in  bit;
11         S : out bit;
12     );
13 end component;
14
15 -- <Sinais>
16 signal sA, sB : bit;
17 signal sOR    : bit;
```

Conectar as interfaces de pl_or_2 com os <Sinais> de pl_or_2_tb

VHDL – Exemplo OR – Simulação

- **Instanciação** com <mapeamento> explícito

```
21  -- <mapeamento> explícito
22  u1_pl_or_2 : pl_or_2 port map(A => sA, B => sB, S => sOR);
```

- O conjunto de símbolos “=>” indica **mapeado para**
 - Exemplo:
A => sA
Interface **A** de pl_or_2 **mapeado para** <Sinal> **sA**
 - Na instanciação com <mapeamento> explícito, a ordem não é importante!

VHDL – Exemplo OR – Simulação

- ou¹ **instanciação** com <mapeamento> posicional

```
24    -- <mapeamento> posicional
25    u2_pl_or_2 : pl_or_2 port map(sA, sB, s0R);
```

- Na instanciação com <mapeamento> posicional, a ordem **é** importante!

¹ ou-exclusivo

VHDL – Exemplo OR – Simulação

- <códigoComportamento> do testbench
 - contém uma unidade **process**
 - que fará o controle dos sinais manipuláveis
 - para alterar os sinais observáveis
 - Um **process** é uma sequência de <comandos> VHDL
 - Para nosso exemplo OR usaremos
 - <comando> de atribuição
 <receptor> <= <expressão>
 - <comando> de espera por tempo determinado
 wait for <x>ns
 - A execução do process é “circular”

VHDL – Exemplo OR – Simulação

- **process** é uma unidade instanciada

```
26     <nomeUnidade> : process  
27     begin  
28         <comandos>  
29     end process <nomeUnidade>;
```

- <nomeUnidade>: u3_tb

```
31     u3_tb : process  
32     begin  
33         <comandos>  
34     end process u3_tb;
```

VHDL – Exemplo OR – Simulação

- <comando> de atribuição

```
15  -- <Sinais>  
16  signal sA, sB : bit;  
17  signal sOR    : bit;
```

Situação	sA	sB	sOR
0	0	0	?

sA, sB e sOR são do <tipo> bit

- Bits são escritos em VHDL utilizando aspas simples
 - 0 → '0' e 1 → '1'
- Atribuir aos <Sinais> sA e sB os valores da situação 0
 - E após, aguardar um “tempo”
 - Usaremos 10 ns por observação

VHDL – Exemplo OR – Simulação

- <comando> de atribuição
- <comando> de espera

Situação	sA	sB	sOR
0	0	0	?

```

15  -- <Sinais>
16  signal sA, sB : bit;
17  signal sOR    : bit;

```

– process

```

27  u3_tb : process
28  begin
29      sA <= '0';
30      sB <= '0';
31      wait for 10 ns;

```

atribuição de bit '0' para sA e sB

aguardar 10 ns de observação

valor 10_d é um número decimal,
pois está sem aspas simples!

VHDL – Exemplo OR – Simulação

- <comandos> para as situações restantes

– process

```
33      sA <= '0';  
34      sB <= '1';  
35      wait for 10 ns;  
36  
37      sA <= '1';  
38      sB <= '0';  
39      wait for 10 ns;  
40  
41      sA <= '1';  
42      sB <= '1';  
43      wait for 10 ns;
```

Situação	sA	sB	sOR
1	0	1	?
2	1	0	?
3	1	1	?

VHDL – Exemplo OR – Simulação

- Finalizando o process
 - A execução do process é “circular”
 - Indica que, por padrão, ao chegar no final da lista de <comandos>, retorna ao início e executa novamente a partir do 1º <comando>.

```
26 <nomeUnidade> : process
27 begin
28     <comandos>
29 end process <nomeUnidade>;
```

Loop
ou
Laço de Repetição



- Como prevenir o loop?
 - <comando> wait
 - ao encerrar o process

VHDL – Exemplo OR – Simulação

- Finalizando o **process**

```
45         wait;  
46     end process u3_tb;
```

- Finalizando o comportamento

```
48 end tb;
```

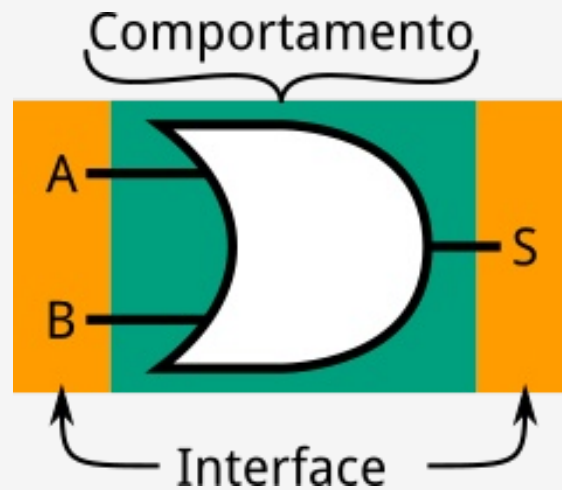

- Arquivo `pl_or_2_tb.vhdl` (completo)

pl_or_2_tb.vhdl x

```
1 entity pl_or_2_tb is
2   -- Entidade vazia
3 end pl_or_2_tb;
4
5 architecture tb of pl_or_2_tb is
6   -- <subEntidade>
7   component pl_or_2 is
8     port(
9       A : in bit;
10      B : in bit;
11      S : out bit
12    );
13   end component;
14
15   -- <Sinais>
16   signal sA, sB : bit;
17   signal sOR   : bit;
18
19 begin
```

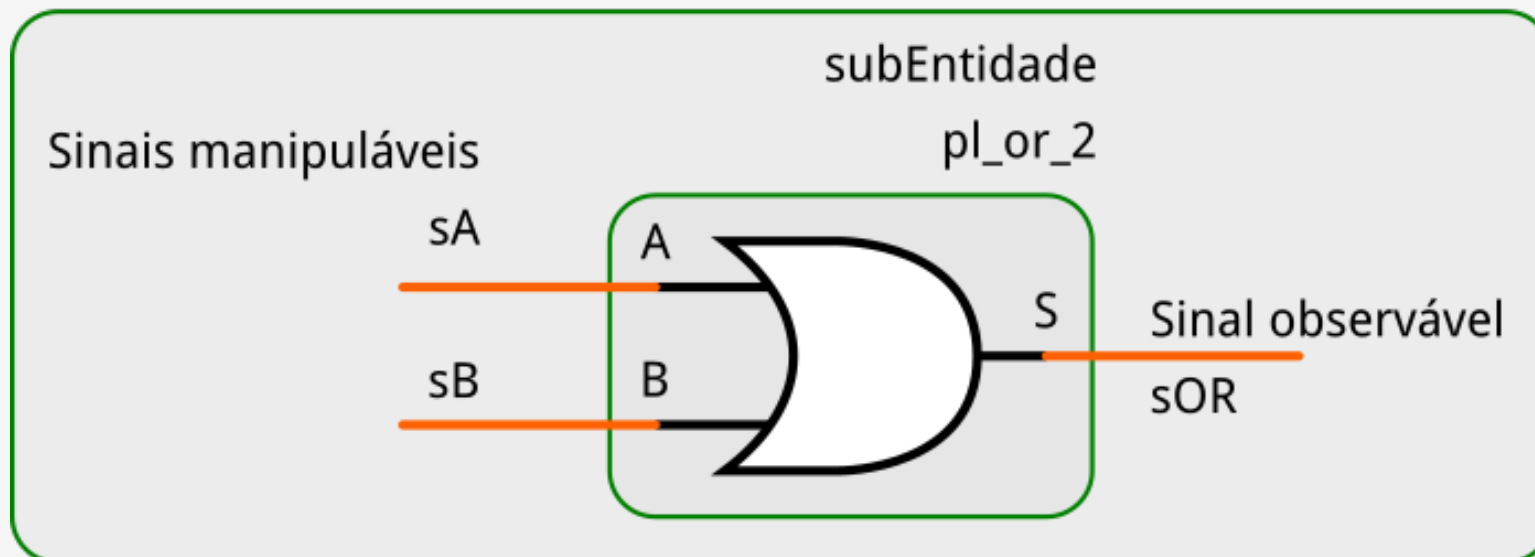
```
19 begin
20   -- <mapeamento> posicional
21   u2_pl_or_2 : pl_or_2 port map(sA, sB, sOR);
22
23   u3_tb : process
24   begin
25     sA <= '0';
26     sB <= '0';
27     wait for 10 ns;
28
29     sA <= '0';
30     sB <= '1';
31     wait for 10 ns;
32
33     sA <= '1';
34     sB <= '0';
35     wait for 10 ns;
36
37     sA <= '1';
38     sB <= '1';
39     wait for 10 ns;
40
41     wait;
42   end process u3_tb;
43
44 end tb;
```

- O que temos até agora:



arquivo `pl_or_2.vhdl`

Entidade principal
`pl_or_2_tb`



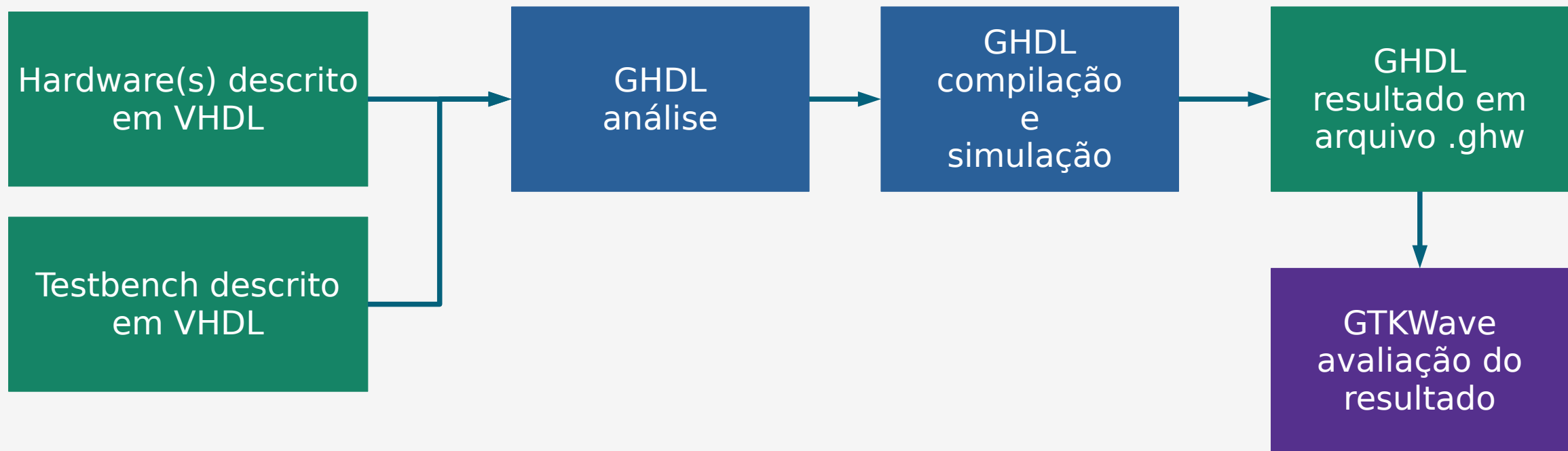
arquivo `pl_or_2_tb.vhdl`

GHLD e GTKWave

simulando e observando
o hardware construído

GHDL

- GHDL é um analisador, compilador e simulador de código escritos em VHDL.



GHDL – Análise

- Avalia a sintaxe e traduz a descrição do hardware para um formato interno utilizado pelo GHDL
 - Qualquer erro de sintaxe será apontado nesta etapa
 - Por exemplo:
 - Nomes incorretos
 - Esquecimento de ponto-e-vírgula
 - Esquecimento de parênteses
 - Sinal de atribuição incorreto
 - ...
 - Se nenhuma informação é mostrada, a etapa de análise foi concluída com sucesso!

- Análise do arquivo `pl_or_2.vhdl`

```
SD_Dir $>: ghdl -a pl_or_2.vhdl
SD_Dir $>: 
```

- Análise do arquivo `pl_or_2_tb.vhdl`

```
SD_Dir $>: ghdl -a pl_or_2_tb.vhdl
SD_Dir $>: 
```

- ou, como somos preguiçosos
 - Podemos analisar TODOS os arquivos `.vhdl`

```
SD_Dir $>: ghdl -a *.vhdl
SD_Dir $>: 
```

- Exemplo de análise (preguiçosa) com erro:

```
SD_Dir $>: ghdl -a *.vhdl  
pl_or_2.vhdl:11:19:error: ';' expected at end of signal assignment  
pl_or_2.vhdl:11:19:error: (found: 'end')  
SD_Dir $>: □
```

- Ao verificar o arquivo pl_or_2.vhdl, na linha 11:

```
9 architecture comportamento of pl_or_2 is  
10 begin  
11     S <= A or B  
12 end comportamento;
```

faltou o “;” no final da linha!
É necessário corrigir e analisar novamente!

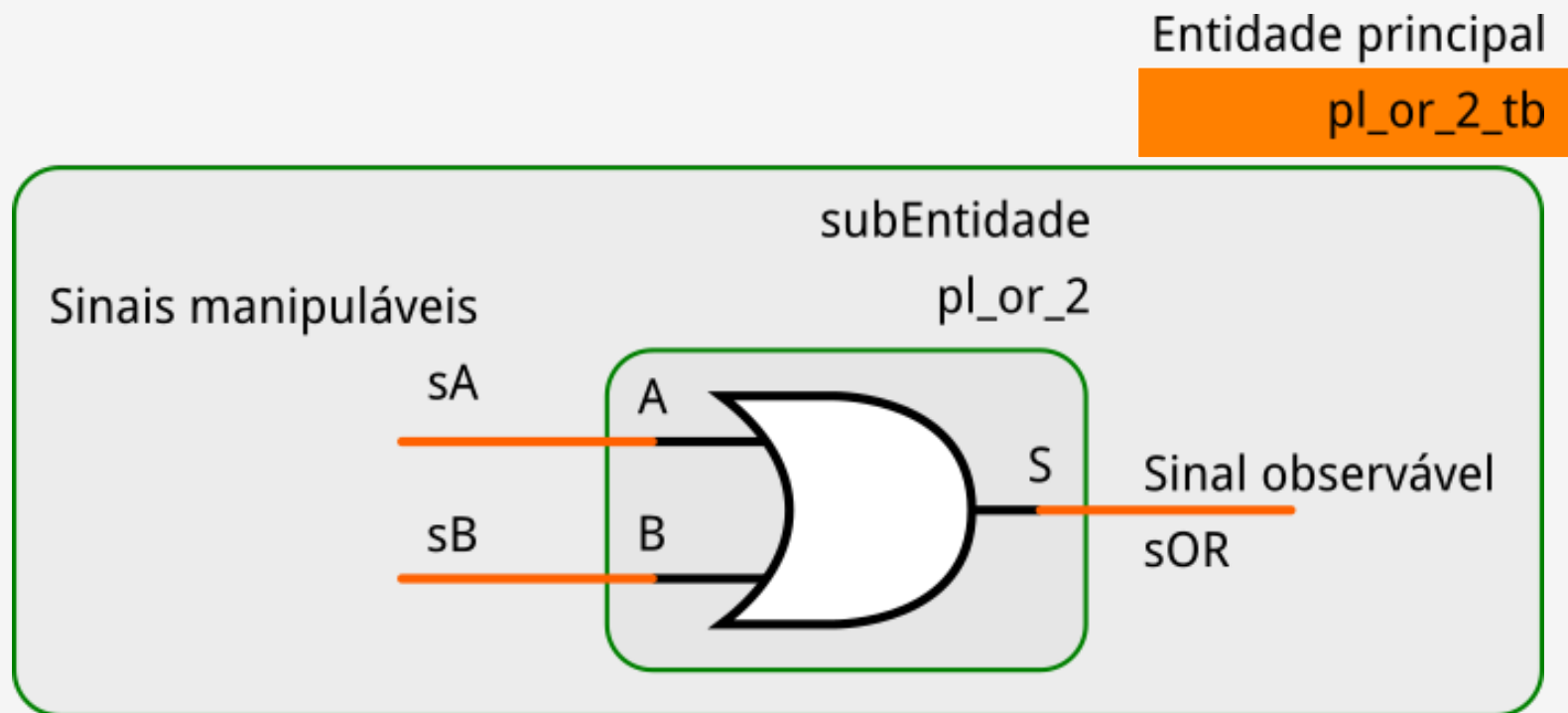
Importante: erros apresentam o **arquivo**, seguido da **linha** onde o erro foi encontrado, e uma **descrição** do mesmo.

GHDL – Compilação

- Após a análise do(s) arquivo(s), será criado um arquivo adicional chamado **work-obj93.cf**
 - Contém informações sobre o hardware que será simulado
 - Para TODA alteração da descrição do hardware em algum arquivo .vhdl, uma nova análise deverá ser executada, SEMPRE!
 - Com todos os arquivos analisados e o arquivo work-obj93.cf existente e atualizado, é hora de **simular!**

GHDL – Simulação


- Para simular um hardware com GHDL é necessário conhecer alguns parâmetros:
 - Qual é o nome da <Entidade> principal?
 - Isto é, a <Entidade> Testbench que manipula e observa os <Sinais>



GHDL – Simulação

- Quanto tempo queremos simular?
 - No nosso caso, é finito e determinado pelo **process**

```
27 u3_tb : process
28 begin
29     sA <= '0';
30     sB <= '0';
31     wait for 10 ns;
32
33     sA <= '0';
34     sB <= '1';
35     wait for 10 ns;
36
37     sA <= '1';
38     sB <= '0';
39     wait for 10 ns;
40
41     sA <= '1';
42     sB <= '1';
43     wait for 10 ns;
44
45     wait;
46 end process u3_tb;
```



4 * wait for 10 ns
TOTAL = 40 ns

GHDL – Simulação

- Onde salvaremos nossa simulação?
 - Normalmente, é em arquivo de onda .ghw com o mesmo nome da <Entidade> simulada
 - `pl_or_2_tb.ghw`

GHDL – Simulação – agora vai!

- Qual é o nome da <Entidade> principal?
 - Parâmetro `-r <nomeEntidade>`
- Quanto tempo queremos simular?
 - Parâmetro `--stop-time=XXns`
- Onde salvaremos nossa simulação?
 - Parâmetro `--wave=<nomeArquivo>.ghw`
- Comando linux:

```
ghdl -r <nomeEntidade> --stop-time=XXns --wave=<nomeArquivo>.ghw
```

GHDL – Simulação – agora vai²!

- Simulação do nosso exemplo:

```
SD_Dir $>: ghdl -r pl_or_2_tb --stop-time=40ns --wave=pl_or_2_tb.ghw
```

<nomeEntidade>

Tempo
em ns

Arquivo de
resultados

GHDL – Simulação – agora foi!

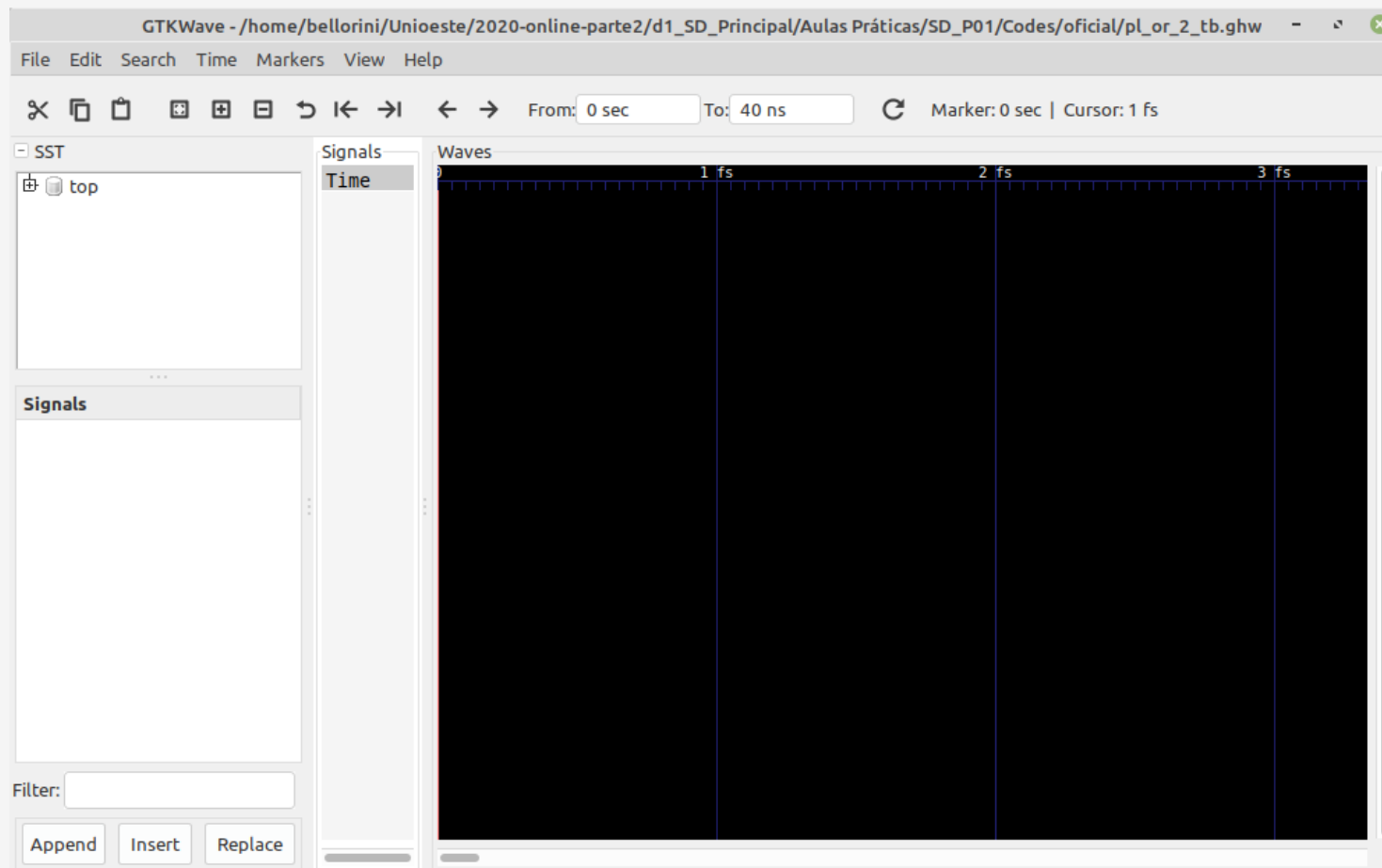
- Simulação do nosso exemplo:

```
SD_Dir $>: ghdl -r pl_or_2_tb --stop-time=40ns --wave=pl_or_2_tb.ghw
```

- Ao executar a linha de comando, o arquivo `pl_or_2_tb.ghw` será criado no mesmo diretório.
 - Este arquivo é aberto com o programa GTKWave

GTKWave – Simulação

- Simulação concluída e arquivo **pl_or_2_tb.ghw** gerado
 - Abra o arquivo com o programa GTKWave

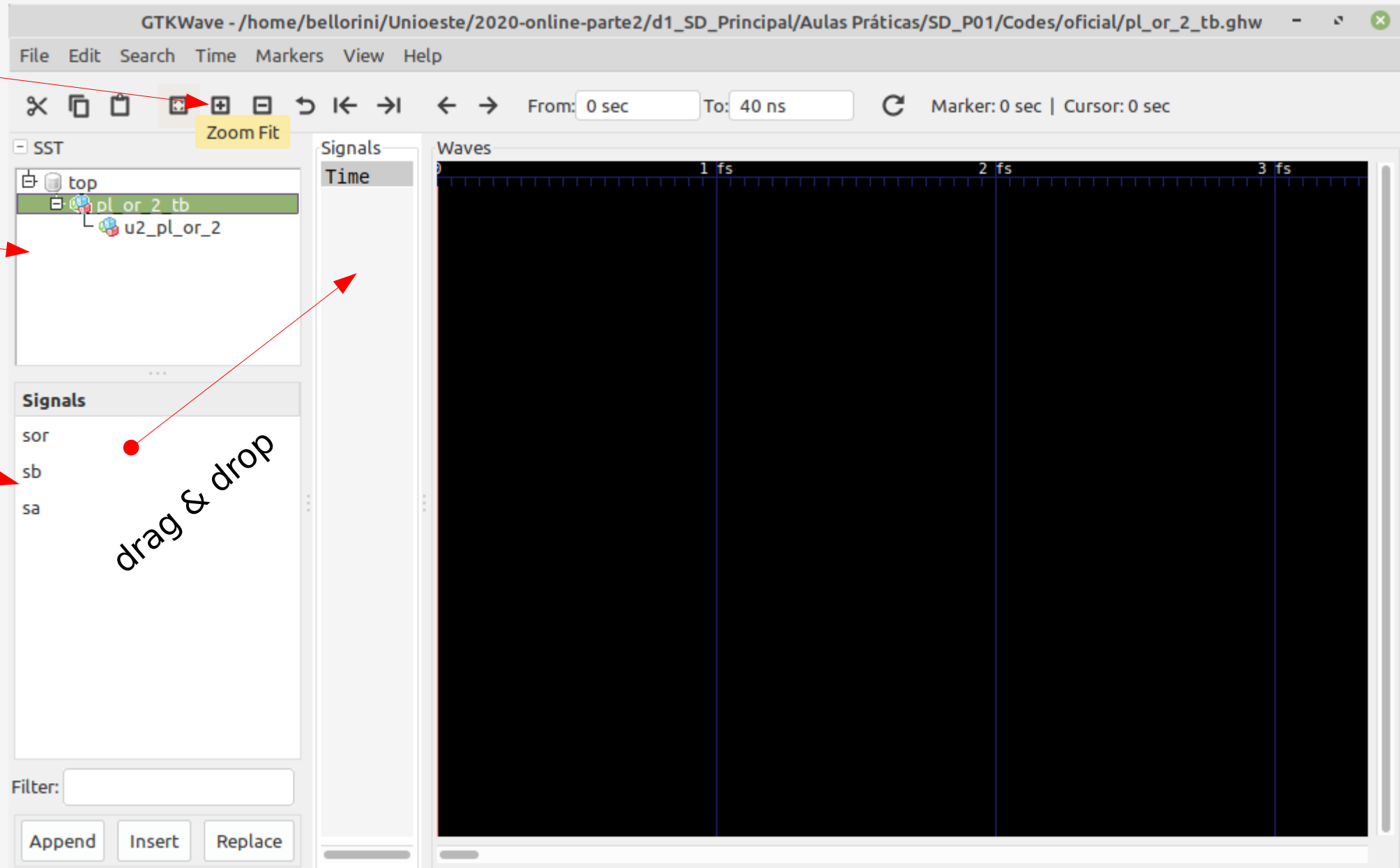


- GTKWave – Simulação (verificando o resultado)

01 - Zoom FIT

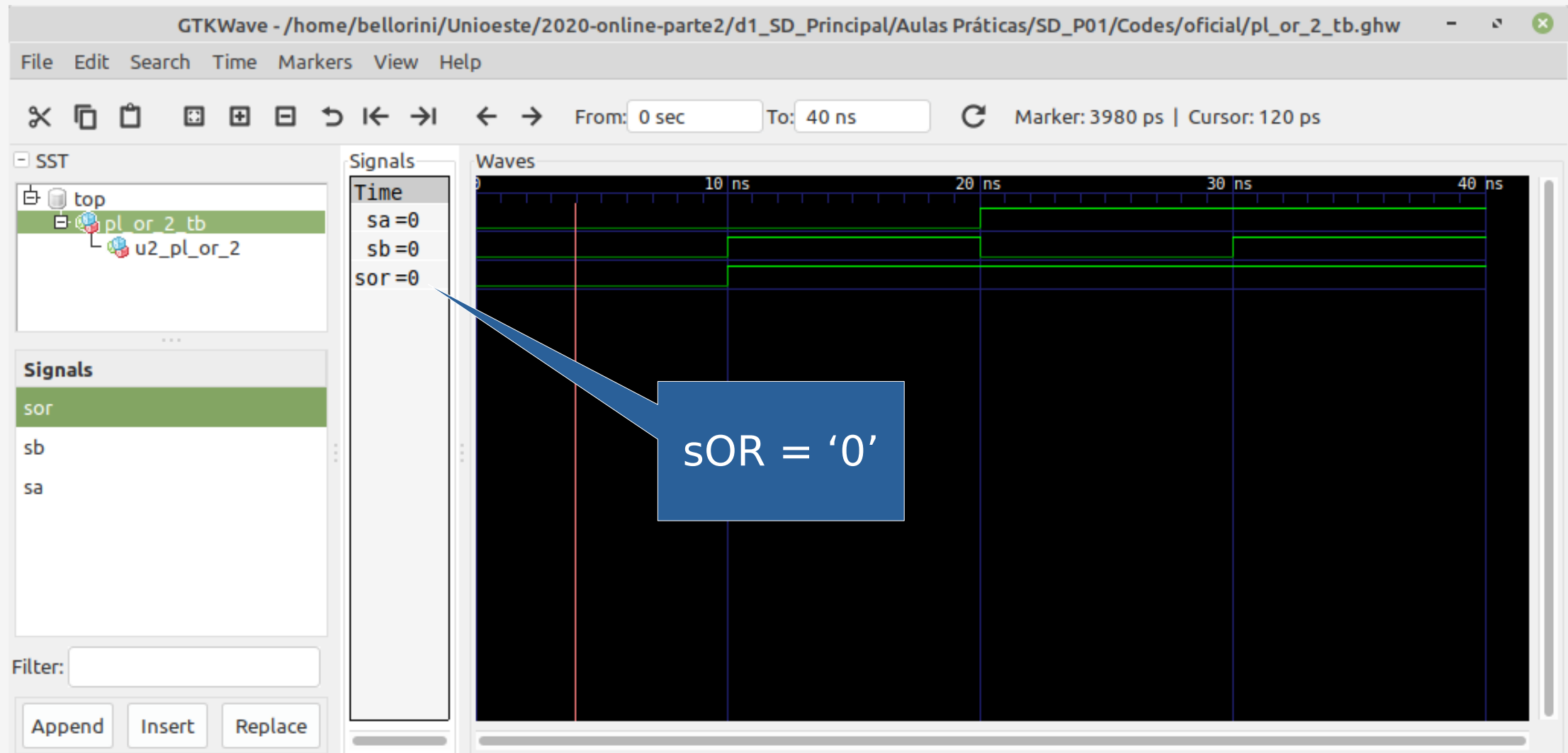
02 - Entidades

03 - Sinais



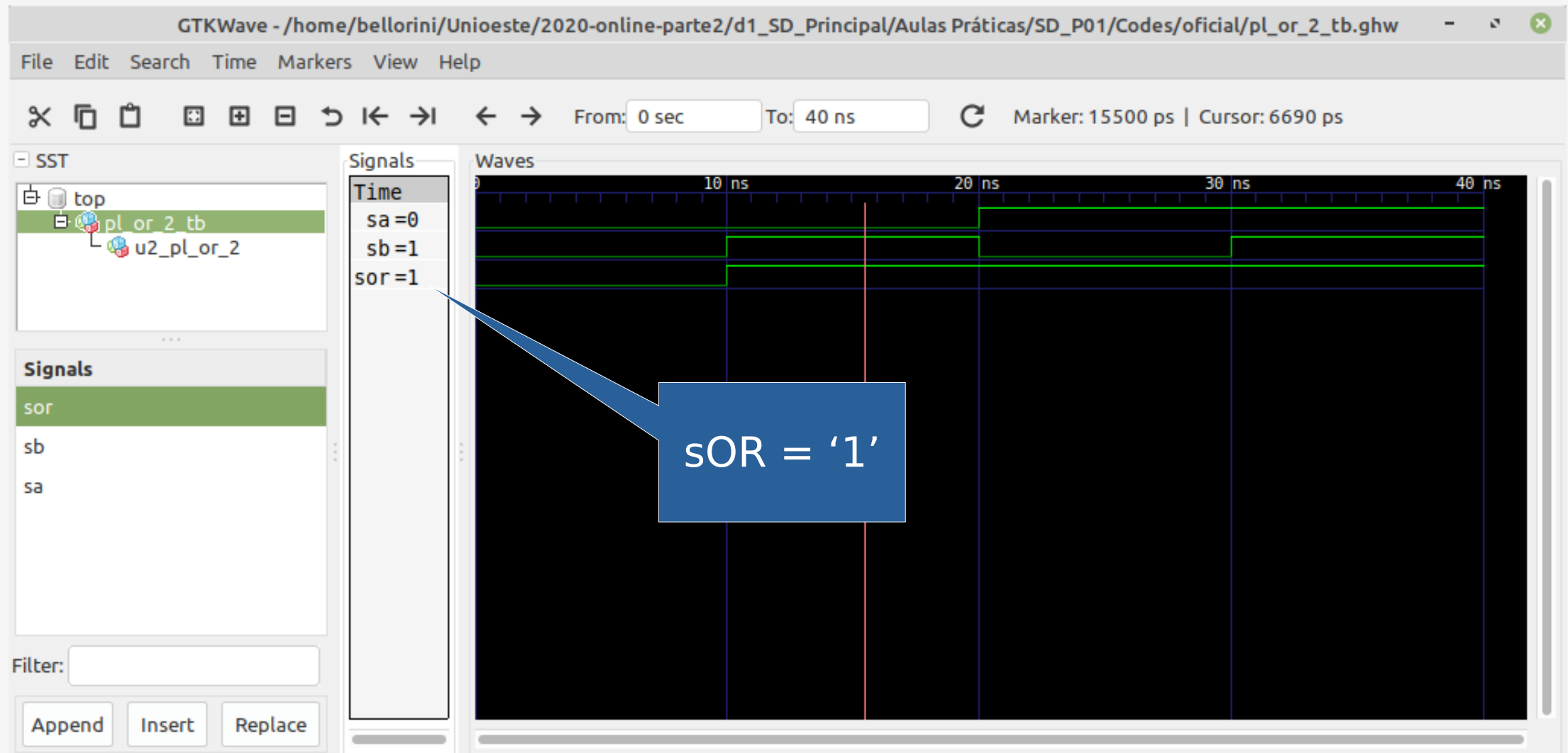
- GTKWave – Simulação (verificando o resultado por Situação)

Situação	sA	sB	sOR
0	0	0	?



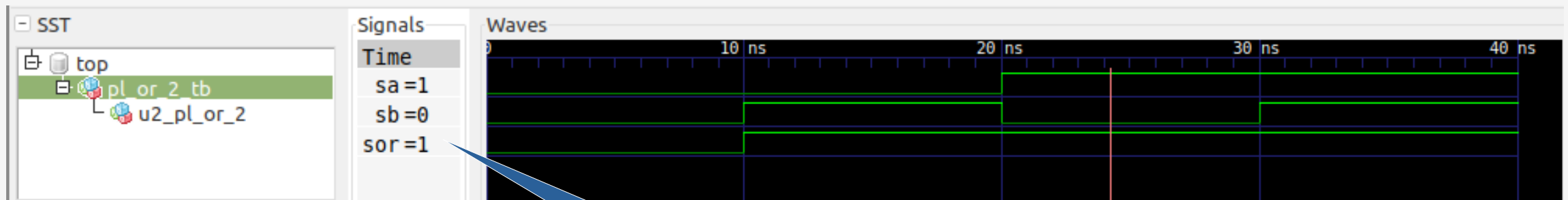
- GTKWave – Simulação (verificando o resultado por Situação)

Situação	sA	sB	sOR
0	0	1	?

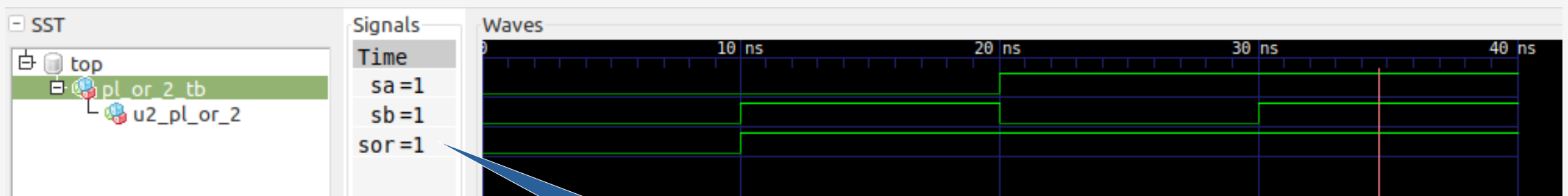


- GTKWave – Simulação (verificando o resultado por Situação)

Situação	sA	sB	sOR
2	1	0	?
3	1	1	?



sOR = '1'



sOR = '1'

VHDL, GHDL e GTKWave

Atividades

Atividades (slide a)

- Utilizando o arquivo:

SD 102 - Introducao a Circuitos Digitais - Companion Document.pdf

- 1ª página contém as palavras reservadas para cada função lógica

- Elabore:

- Entidades:

- AND, NAND, NOR, XOR e XNOR de 2 entradas
 - 5 arquivos .vhd
- AND e OR de 3 e 4 entradas
 - 4 arquivos .vhd

Atividades (slide a)

- Elabore:
 - Testbenchs:
 - AND, NAND, NOR, XOR e XNOR de 2 entradas
 - Único arquivo .vhd
 - Sinais sA e sB compartilhados entre pls
 - Sinais de saídas independentes
 - 5 Sinais de saídas
 - AND e OR de 3 e 4 entradas
 - Único arquivo .vhd para 3 entradas
 - Único arquivo .vhd para 4 entradas
 - Sinais sA, sB, sC [, sD] compartilhados entre pls
 - Sinais de saídas independentes

Dúvidas?