

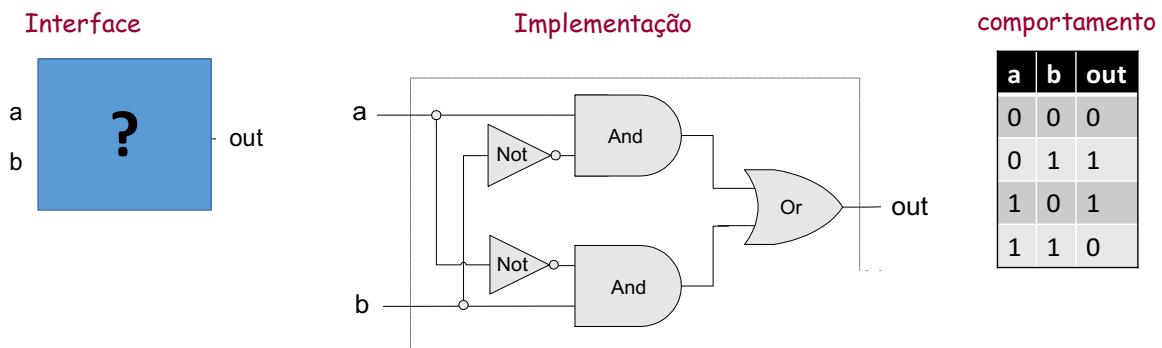
### Laboratório

#### Porta XOR- implementação

#### Introdução

Um simulador de hardware é um ambiente que oferece um conjunto amplo de ferramentas para programadores que facilitam e apoiam o desenvolvimento de hardware, através da simulação. Um simulador de hardware normalmente consiste num editor de código-fonte (com várias ajudas na escrita do programa), ferramentas de automação nos testes (permitem agilizar ações repetitivas), um simulador (ajuda na deteção e correção dos erros).

#### Porta XOR



### Laboratório

#### Porta XOR- implementação

#### Nível 1 - Construir o circuito (ficheiro HDL)

Um circuito é definido através da edição de um ficheiro com a extensão .hdl que descreve a interface e a implementação do circuito. No caso específico do simulador escolhido o nome do ficheiro deve ser o mesmo que o nome do circuito. Na figura 1 é apresentado o ficheiro Xor.hdl que contém a interface e a implementação do chip Xor.

chip interface

```
/** Exclusive-or gate. out = a xor b */
CHIP Xor {
    IN a, b;
    OUT out;

    // Implementation missing.
}
```

- Interface do chip:
  - Nome do chip;
  - Nome dos pins de entradas e de saída;
  - Documentação sobre a operação do chip.
- A interface do chip, tipicamente é fornecida pelo arquiteto, descreve o contrato (i.e., o funcionamento esperado do chip)

Figura 1. Ficheiro Xor.hdl com o chip Xor.

Na figura 2 o mesmo ficheiro já possui a implementação do chip Xor.

chip interface

chip implementation

```
/** Exclusive-or gate. out = a xor b */
CHIP Xor {
    IN a, b;
    OUT out;

    PARTS:
        Not(in=a, out=nota);
        Not(in=b, out=notb);
        And(a=a, b=notb, out=w1);
        And(a=nota, b=b, out=w2);
        Or(a=w1, b=w2, out=out);
}
```

- Qualquer chip pode ser implementado de vários modos diferentes.
- Este exemplo, em particular, possui a seguinte implementação:  

$$\text{Xor}(a,b) = \text{Or}(\text{And}(a,\text{Not}(b)), \text{And}(b,\text{Not}(a)))$$
- Not; And; Or: Chips preexistences. São chips fornecidos pelo simulador ou previamente desenvolvimento que podem ser reutilizados.
- nota, notb, w1, w2, nota, notb: pinos internos, utilizados para ligar os componentes internos.

Figura 2. Implementação do circuito Xor.

### Laboratório

#### Porta XOR- implementação

#### Nível 2- Carregar e Explorar a implementação do chip

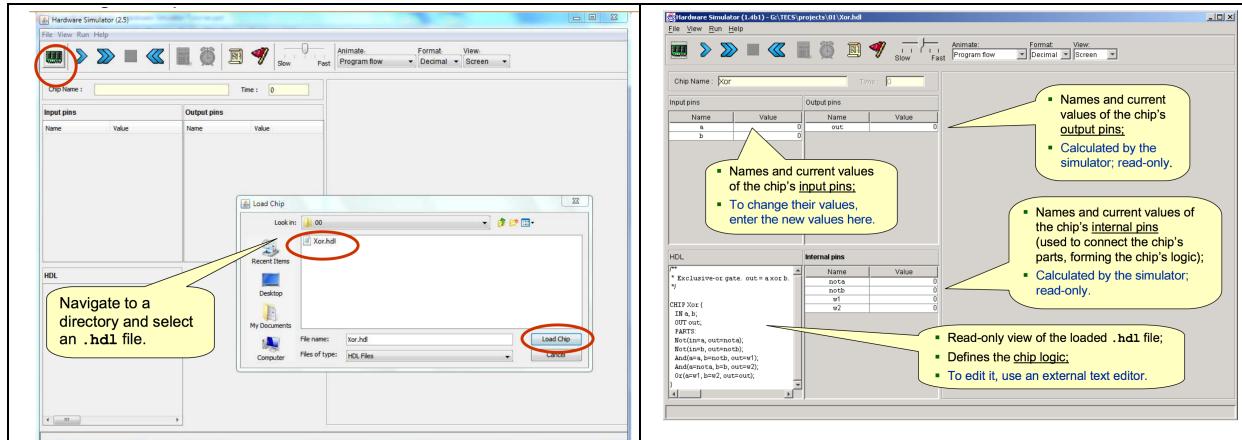


Figura 3. Carregar o Chip.

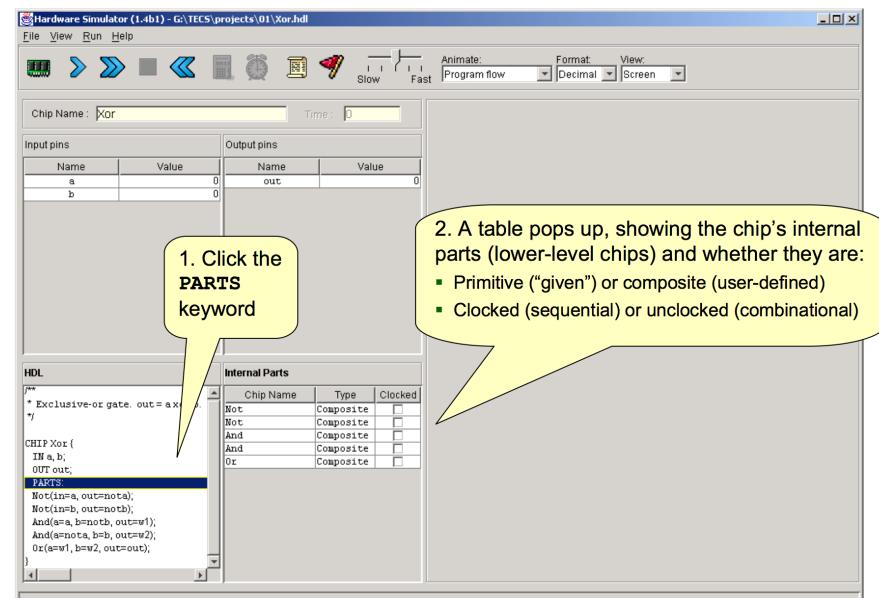


Figura 4. Explorar a implementação do chip.

### Laboratório Porta XOR- implementação

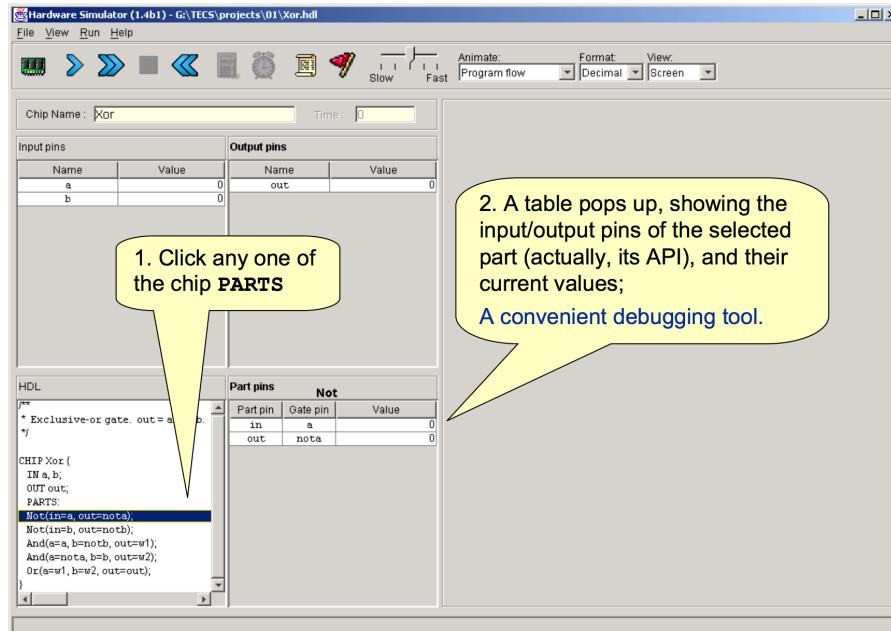


Figura 5. Explorar a implementação do chip

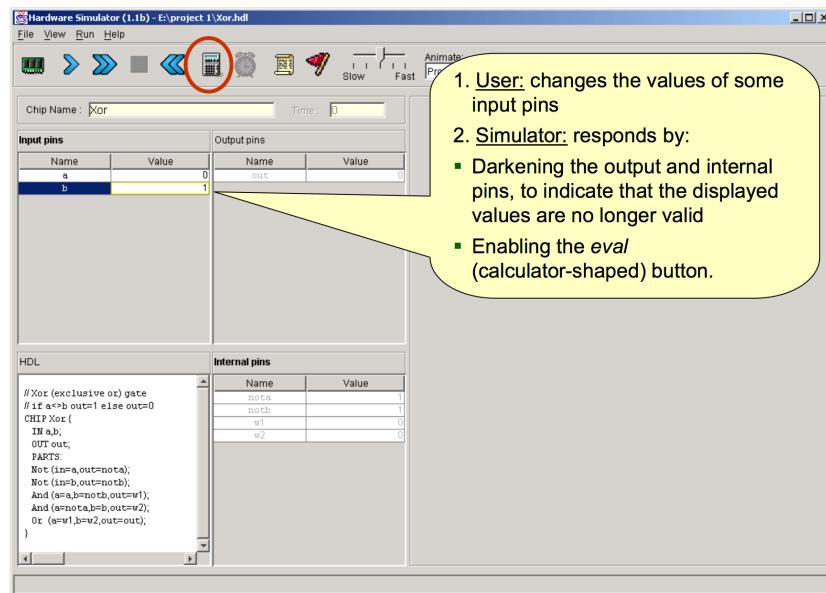


Figura 6. Explorar a implementação do chip

### Laboratório Porta XOR- implementação

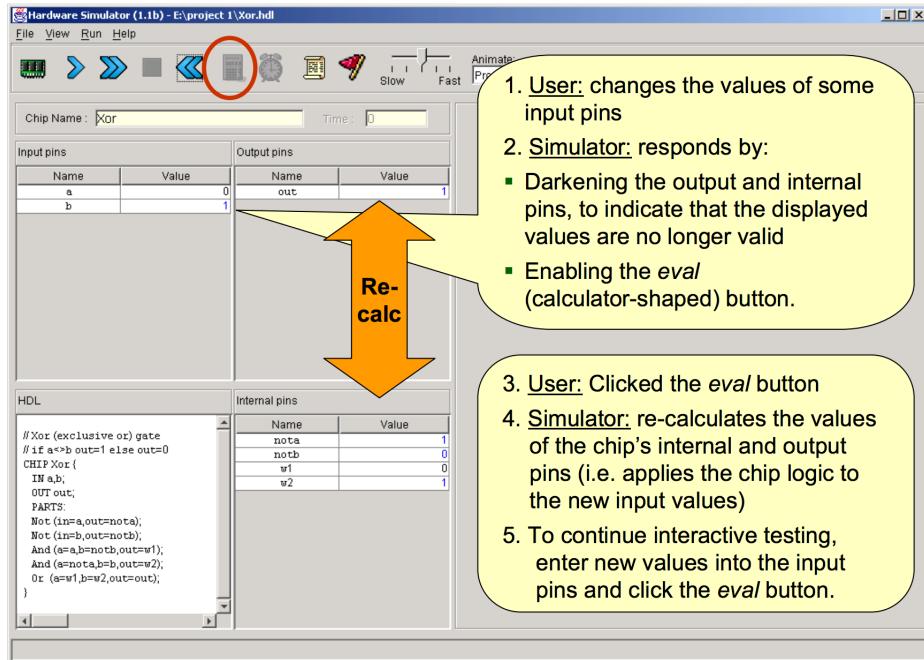


Figura 7. Explorar a implementação do chip.

### Laboratório

#### Porta XOR- implementação

##### Nível 3 Realizar testes automáticos sobre o chip

A automatização dos testes que podemos realizar sobre o chip **Xor.hdl** obriga a produzir o ficheiro **Xor.tst** o qual vai conter o conteúdo da figura apresentada na figura 8.

No ficheiro podemos observar que:

**load Xor.hdl** indica qual o ficheiro que contém o chip que deve ser carregado.

**output-file Xor.out** indica qual o ficheiro que dever ser criado com o resultado dos testes realizados.

**output-list Xor.out a%B3.1.3 b%B3.1.3 out%B3.1.3** indica o formato pelo qual o resultado deve ser escrito no ficheiro **Xor.out**

De seguida podemos fazer os testes no chip. Cada teste é composto pela atribuição de valores à entrada (0 ou 1) com o comando **set**. Uma vez escolhidos os valores de entrada **eval**, produz a saída.

Por fim **output** escreve, no ficheiro de saída, o resultado.

```
load Xor.hdl,
output-file Xor.out,
output-list a%B3.1.3 b%B3.1.3 out%B3.1.3;

set a 0,
set b 0,
eval,
output;

set a 0,
set b 1,
eval,
output;

set a 1,
set b 0,
eval,
output;

set a 1,
set b 1,
eval,
output;
```

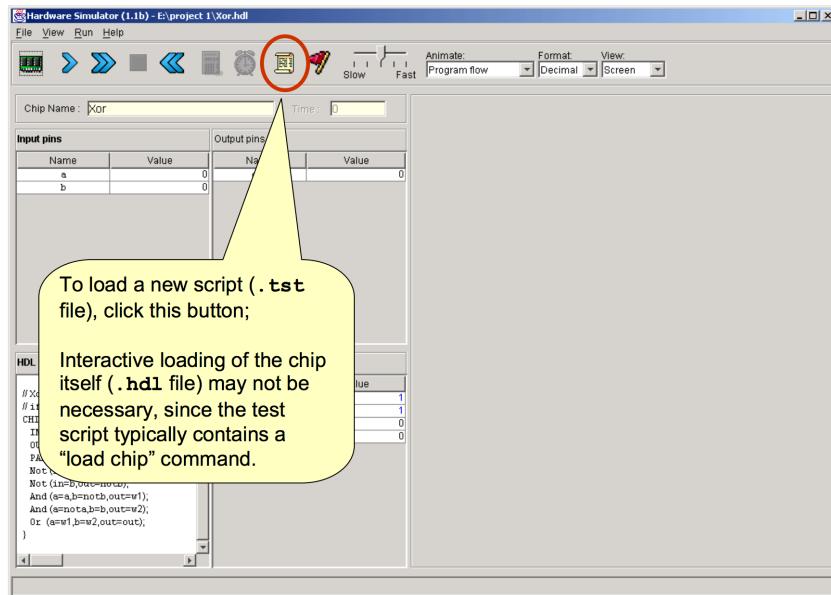
**Figura 8 – ficheiro Xor.tst**

### Laboratório

#### Porta XOR- implementação

#### Nível 4 Carregar os testes e obter os resultados no ficheiro de saída

Para carregar os testes devemos utilizar a opção indicada na figura. Ao fazer irá carregar o script com os testes que devem ser realizados e irá produzir os resultados no ficheiro indicado.



a	b	out
0	0	0
0	1	1
1	0	1

Figura 9 – ficheiro Xor.out

### Laboratório

#### Porta XOR- implementação

##### Nível 5 Realizar testes exaustivos sobre o circuito

O processo de testes tem como objetivo a validação de um chip em relação aos seus requisitos. A execução destes testes envolve tipicamente a interação de um utilizador experiente com o chip em causa, por exemplo, através de ações ou validações sobre a interface da mesma. Os testes podem ser automatizados, através da descrição, uma linguagem específica, da sua execução de um teste que deveria ser feito por um utilizador e a sua posterior reprodução automática.

A forma de criar testes na plataforma é através de tabelas, em que são introduzidos os valores de entrada e os valores esperados e ela automaticamente faz a verificação dos resultados. Este tipo de testes realizados por esta ferramenta não são do que asserções ao chip.

O ficheiro inicial o de são armazenados os testes é em todo semelhante ao apresentado no nível 3, figura 8. Somente é indicado qual é a tabela pela qual será feita a comparação entre o resultado obtido e o resultado esperado. Em seguida é apresentado o ficheiro Xor.tst e o ficheiro Xor.cmp.

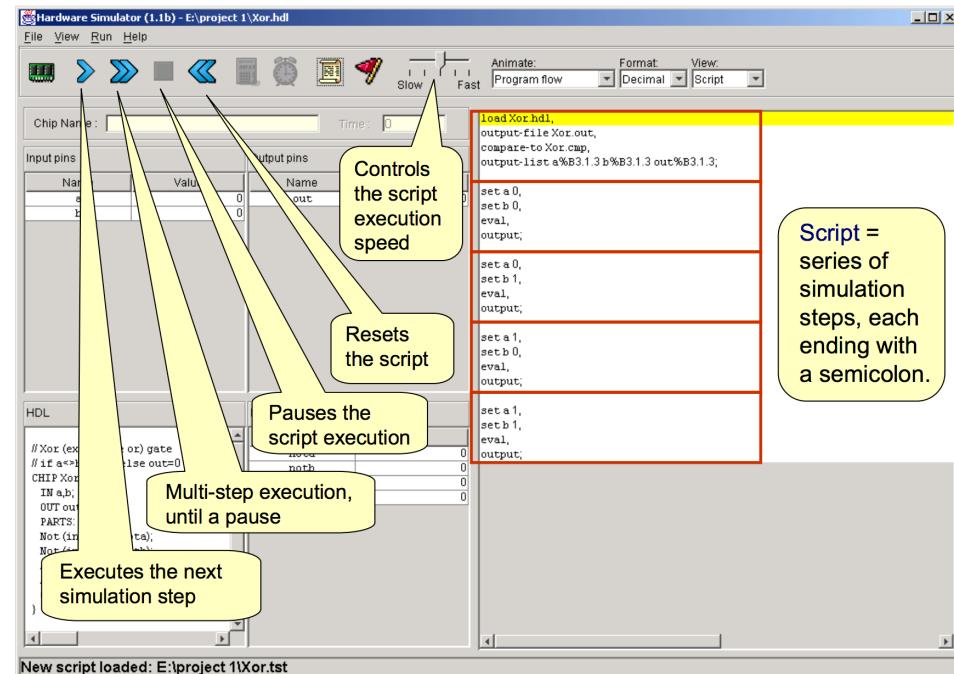
```
load Xor.hdl,
output-file Xor.out,
compare-to Xor.cmp,
output-list a%B3.1.3 b%B3.1.3 out%B3.1.3;
....Indentico....
```

**Figura 10 – tabela Xor.tst**

a	b	out
0	0	0
0	1	1
1	0	1
1	1	0

**Figura 11 – tabela Xor.cmp**

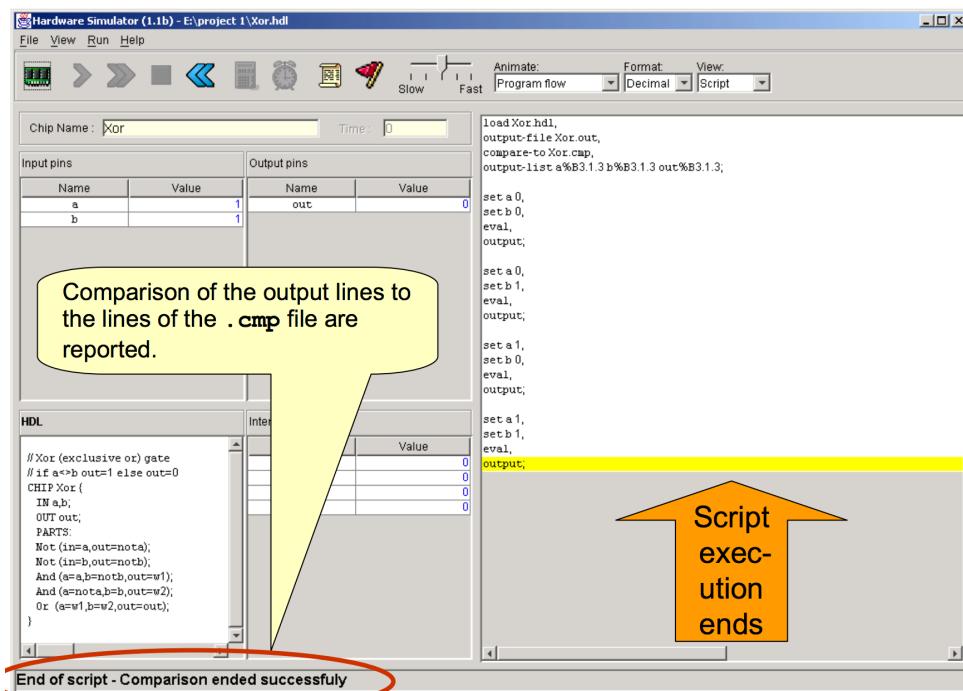
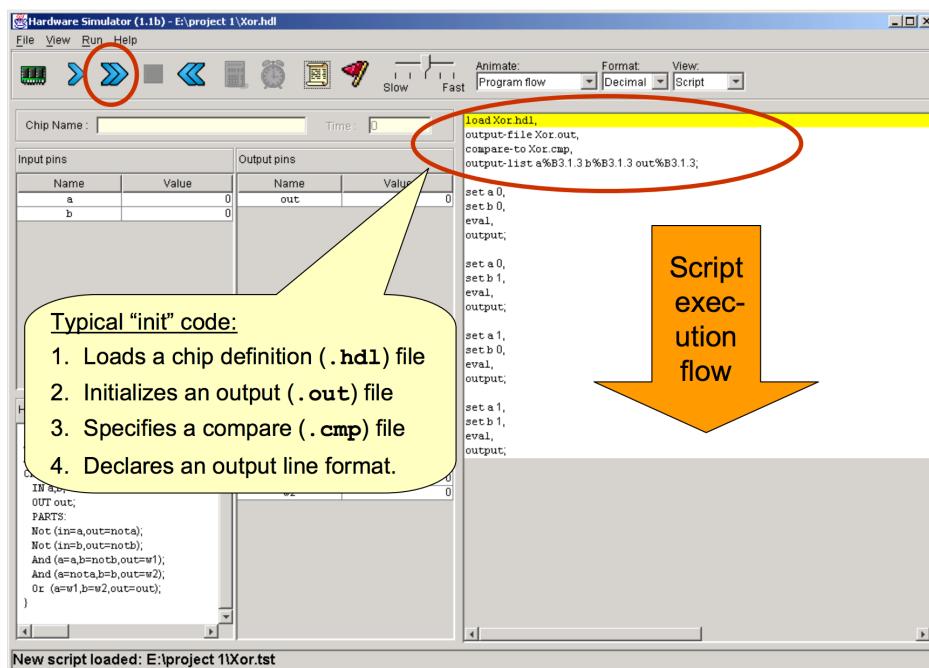
##### Ambiente de testes



### Laboratório

#### Porta XOR- implementação

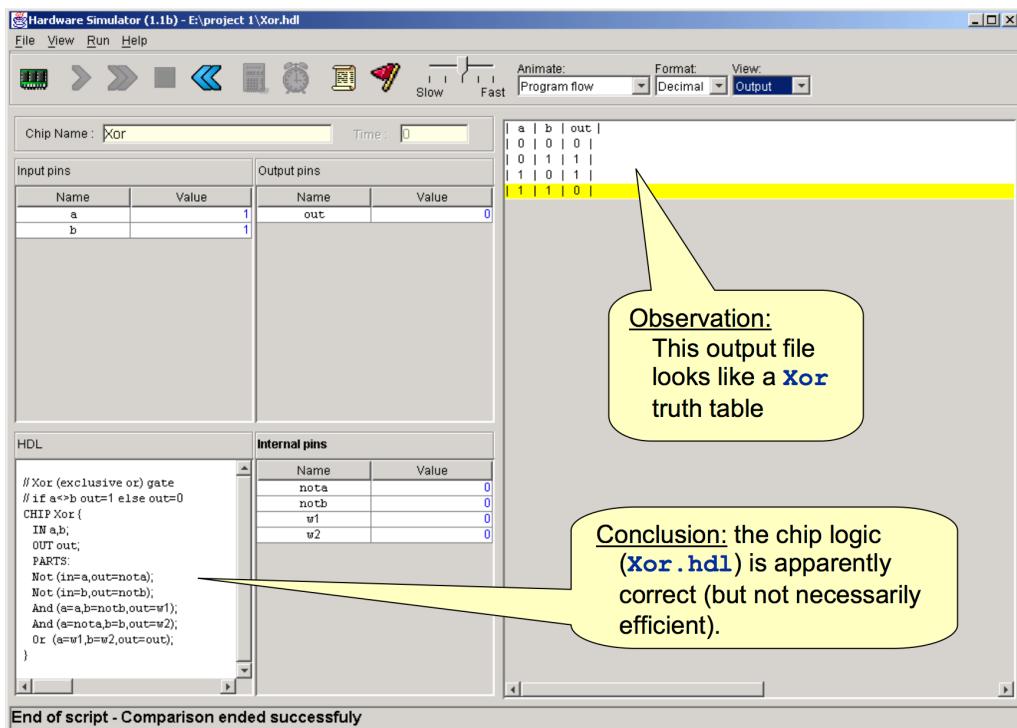
##### Executar os testes



### Laboratório

#### Porta XOR- implementação

**Analizar o ficheiro de saída e o ficheiro esperado**



The screenshot shows a hardware simulation interface with the following details:

- Chip Name:** Xor
- Input pins:** a (Value: 1), b (Value: 1)
- Output pins:** out (Value: 0)
- Internal pins:** nota (Value: 0), notb (Value: 0), w1 (Value: 0), w2 (Value: 0)
- HDL:**

```

//Xor (exclusive or) gate
//if a==b out=1 else out=0
CHIP Xor{
    IN a,b;
    OUT out;
    PARTS:
        Not (in=a,out=nota);
        Not (in=b,out=notb);
        And (a=nota,b=notb,out=w1);
        And (a=nota,b=b,out=w2);
        Or (a=w1,b=w2,out=out);
}

```
- Truth Table:**

a	b	out
0	0	0
0	1	1
1	0	1
1	1	0
- Message:** End of script - Comparison ended successfully

**Observation:**  
This output file looks like a **Xor** truth table

**Conclusion:** the chip logic (**Xor.hdl**) is apparently correct (but not necessarily efficient).