



**Universidade Federal de Santa Catarina
Centro Tecnológico – CTC
Departamento de Engenharia Elétrica**



CTC UFSC

“Circuitos e Técnicas Digitais”

Prof. Héctor Pettenghi Roldán*

Hector@eel.ufsc.br

Florianópolis, março de 2016.

***Baseados nos slides do Professor Eduardo Bezerra EEL5105 2015.2**

Plano de Aula

“Projeto de Sistemas Digitais com VHDL”

- Objetivos:
 - Apresentar uma visão geral de VHDL
 - Exemplo de descrição VHDL
 - Introdução ao Quartus II – ferramentas de desenvolvimento
 - Estudo de caso / exercício

VHDL - Visão Geral

- VHDL - linguagem para descrição de hardware
- VHDL = VHSIC Hardware Description Language
- VHSIC = Very High Speed Integrated Circuits. Programa do governo dos USA do início dos anos 80.
- No final da década de 80, VHDL se tornou um padrão IEEE (Institute of Electrical and Electronic Engineers).
- Existem diversas ferramentas para simular e sintetizar (gerar hardware) circuitos descritos em VHDL.
- Outras linguagens de descrição de hardware: Verilog, SystemC, AHDL, Handel-C, System Verilog, Abel, Ruby, ...

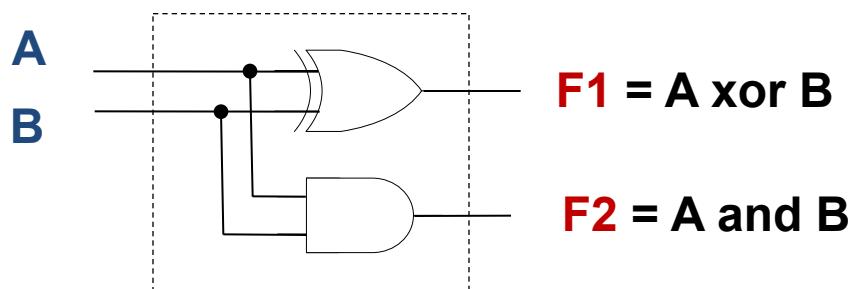
VHDL - Visão Geral

- O projeto de um circuito digital pode ser descrito em VHDL em diversos níveis de abstração (estrutural, comportamental).
- VHDL **NÃO** é uma **linguagem de programação**, e as ferramentas de síntese (não são de “compilação”) não geram códigos executáveis a partir de uma descrição VHDL.
- Descrições em VHDL podem ser simuladas (executadas em um simulador).
- Descrições em VHDL podem ser utilizadas para gerar um hardware (arquivo para configuração de um FPGA, por exemplo).
- A geração de estímulos para simulação VHDL é realizada por intermédio de testbenches.
- Um testbench define os estímulos externos a serem utilizados como entrada para o circuito (definição do comportamento externo ao circuito sob teste).
- O testbench pode ser escrito em VHDL ou em diversas outras linguagens (ex. C, C++, ...).

Descrição de circuito digital em VHDL

ENTITY

A	B	F1	F2
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

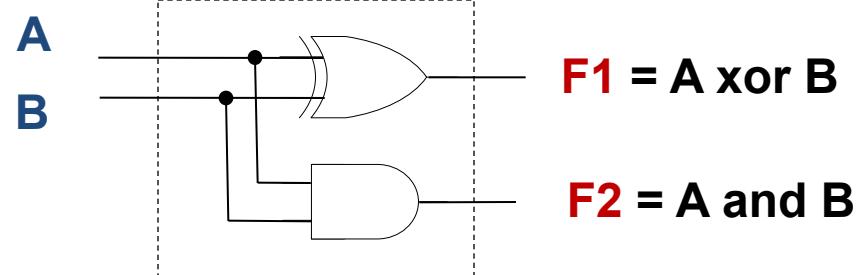


```
entity halfadder is
port (A: in std_logic;
      B: in std_logic;
      F1: out std_logic;
      F2: out std_logic
);
end halfadder;
```

ENTITY – define os “pinos” do circuito digital (sinais), ou seja, a **interface** entre a lógica implementada e o mundo externo.

Descrição de circuito digital em VHDL

ARCHITECTURE



```
architecture circuito_logico of halfadder is
begin
    F1 <= A xor B;
    F2 <= A and B;
end circuito_logico;
```

ARCHITECTURE – define a funcionalidade do circuito digital, utilizando os “pinos” de entrada e saída listados na ENTITY em questão. Uma ENTITY pode possuir diversas implementações diferentes (diversas ARCHITECTURES).

Descrição completa do circuito em VHDL (Entity e Architecture)

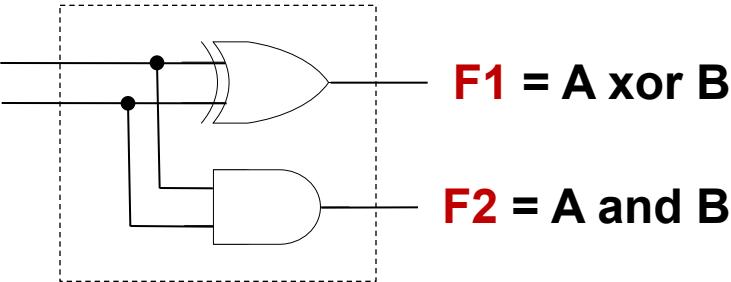
LIBRARIES

```
library IEEE;  
use IEEE.Std_Logic_1164.all;
```

ENTITY

```
entity halfadder is  
port (A: in std_logic;  
      B: in std_logic;  
      F1: out std_logic;  
      F2: out std_logic  
);  
end halfadder;
```

A
B



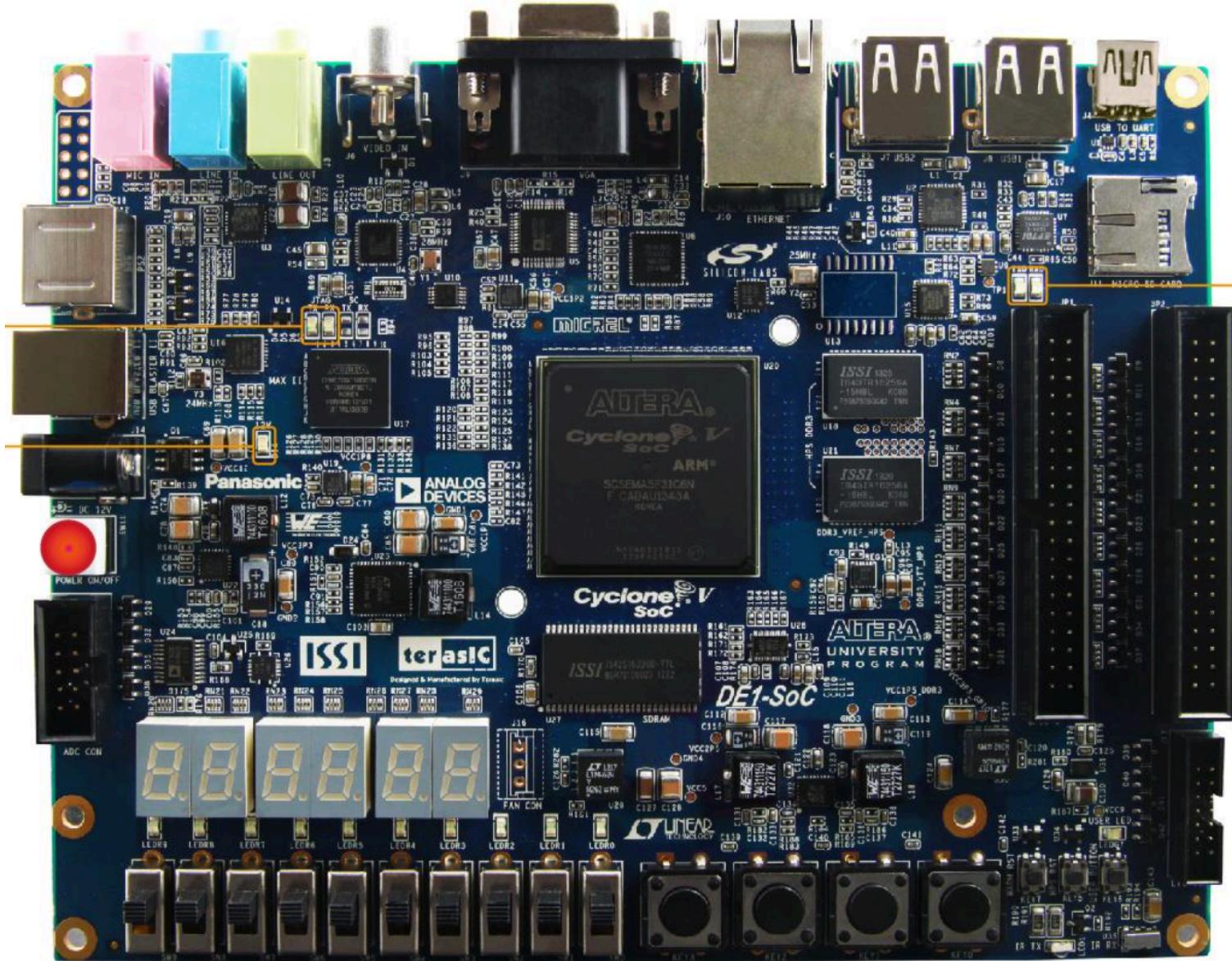
$$\begin{aligned} F1 &= A \text{ xor } B \\ F2 &= A \text{ and } B \end{aligned}$$

ARCHITECTURE

```
architecture circuito_logico of halfadder is  
begin  
  F1 <= A xor B;  
  F2 <= A and B;  
end circuito_logico;
```

Plataforma de prototipação FPGA Altera – DE1-SoC

Kit DE1-SoC da Altera



Interface com o usuário (entrada e saída)

- Placa DE1-SoC possui 10 LEDs vermelhos denominados **LEDR9-0** e 10 chaves denominadas **SW9-0**
- As conexões entre esses componentes e os pinos do FPGA da placa estão definidas no arquivo *pinos.qsf*, disponível no site da disciplina
- São utilizados “vetores” para facilitar o acesso aos LEDs e chaves da placa
- Exemplo: **SW[0]** é o elemento 0 do vetor SW, e está conectado ao pino **PIN_AB12** do FPGA
- No código em VHDL, usar sempre os nomes definidos no arquivo *pinos.qsf*.

Interface com o usuário (entrada e saída)

- Código VHDL para “leitura” das chaves e “escrita” nos LEDs

```
library ieee;  
use ieee.std_logic_1164.all;  
entity part1 is  
    port ( SW      : in std_logic_vector(9 downto 0);  
          LEDR : out std_logic_vector(9 downto 0)  
        );  
end part1;  
architecture behavior of part1 is  
begin  
    LEDR(3 downto 0) <= SW(7 downto 4);  
    LEDR(7 downto 4) <= "0101";  
    LEDR(9) <= (SW(9) AND SW(0)) OR (SW(8) AND '1');  
end behavior;
```

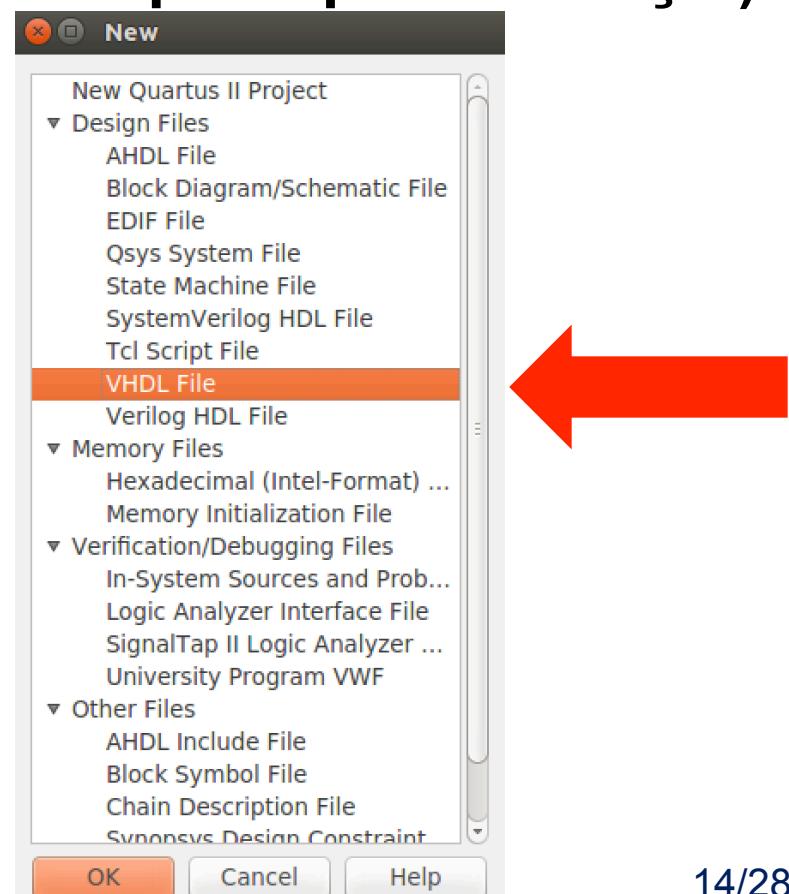
Tarefa a ser realizada na aula prática

Tarefa a ser realizada na aula prática

1. Utilizando a ferramenta Quartus II da Altera, criar um projeto VHDL que implemente o circuito apresentado no **slide 7 (and e xor)**.
2. Realizar a simulação do circuito (VHDL) por intermédio de diagramas de formas de onda, e obter a tabela verdade.
3. Visando fixar o conhecimento do fluxo de ferramentas de projeto, seguir o tutorial descrito no livro texto, e detalhado na última aula.
4. Utilizando as dicas do **slide 11**, alterar o projeto de forma a realizar a entrada dos dados A e B a partir das chaves SW(0) e SW(1), e a apresentação dos resultados F1 e F2 no LEDR(0) e LEDR(1).
5. Testar o circuito no kit DE1-SoC, usando as chaves SW(0) e SW(1) para entrar com os operandos, e observar os resultados nos LEDs.

Resumo do tutorial: *Etapa 1 - Design Entry*

1. [Quartus II] File -> New Project Wizard
2. No “project wizard”, seguir exatamente os passos do tutorial da última aula.
3. File -> New -> VHDL File (Essa é a principal diferença!).
4. Copiar o fonte VHDL do slide 7 para o novo arquivo, e salvar.



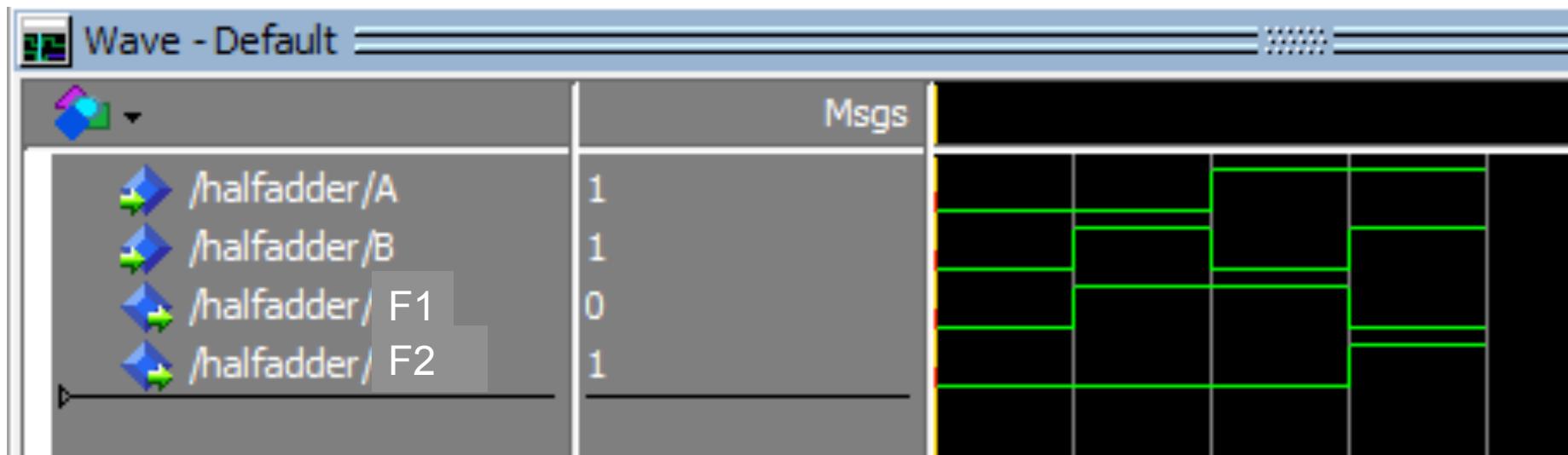
Resumo do tutorial: *Etapa 2 - Simulação*

5. [ModelSim] Simulação Funcional – Teste do circuito

-> não considera informação de temporização.

- Seguir o tutorial de simulação da última aula.

6. Resultado esperado da simulação:



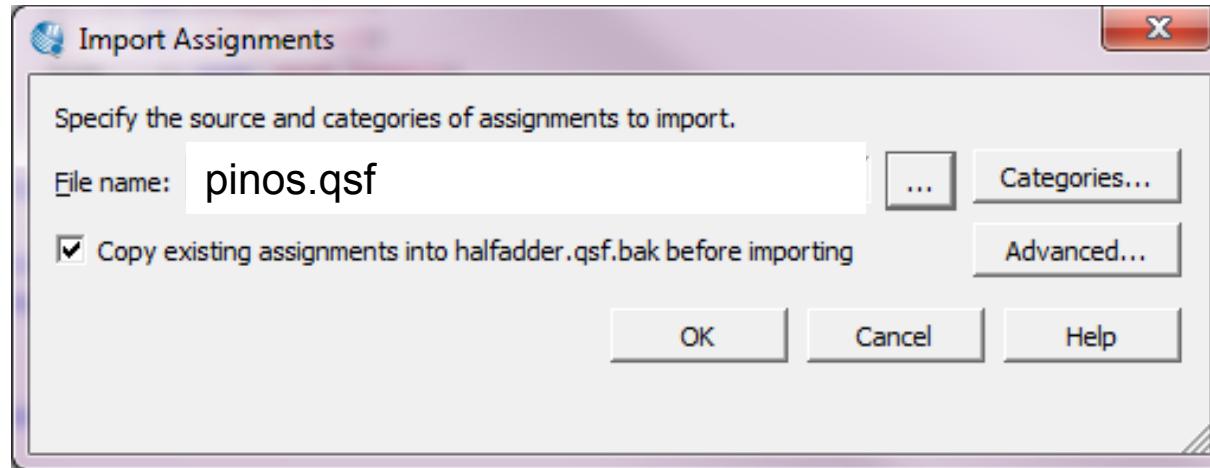
Resumo do tutorial: *Etapa 3 – prototipação FPGA*

7. Adaptar o fonte para os nomes de sinais da DE1-SoC:

```
1  library IEEE;
2  use IEEE.Std_Logic_1164.all;
3
4  entity halfadder is
5    port (
6      SW  : in  std_logic_vector( 9  downto 0);  -- A -> SW(0)
7      LEDR: out std_logic_vector( 9  downto 0);  -- B -> SW(1)
8      );
9    end halfadder;
10
11 architecture ha_stru of halfadder is
12 begin
13   LEDR(0) <= SW(0) xor SW(1);      -- sum    <= A xor B
14   LEDR(1) <= SW(0) and SW(1);    -- carry <= A and B
15 end ha_stru;
```

Resumo do tutorial: *Etapa 3 – prototipação FPGA*

8. Assignments -> Import Assignments (procurar no site e usar o arquivo *pinos.qsf*)



9. Com isso, os pinos do FPGA foram associados aos sinais da entity do VHDL

10. Compilar o VHDL (síntese)

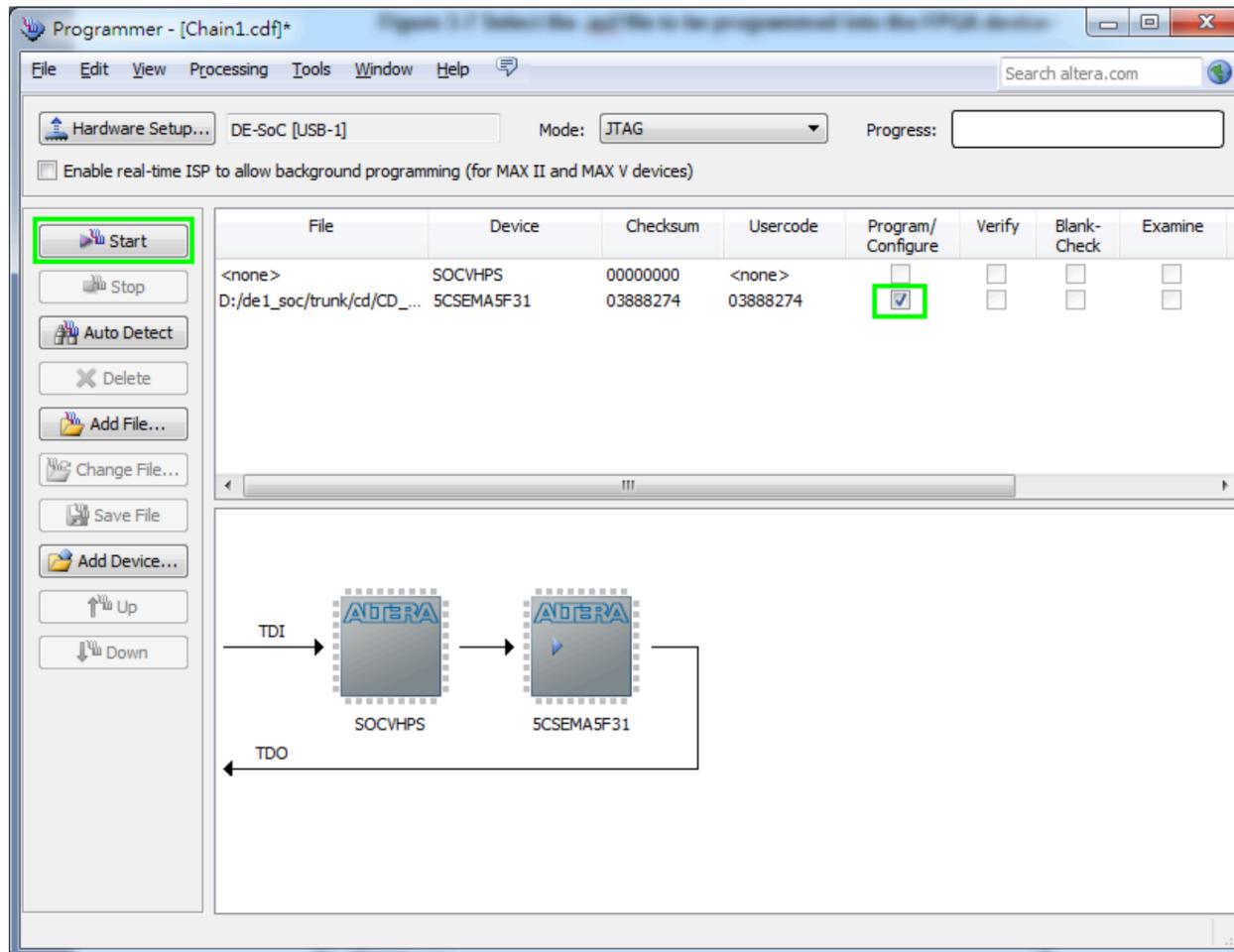
11. **ATENÇÃO!!!** Verificar se o nome da entity é o mesmo nome do projeto, para evitar erros na síntese.

12. A compilação resulta em dezenas de warnings devido aos pinos não conectados do arquivo .qsf

Resumo do tutorial: *Etapa 3 – prototipação FPGA*

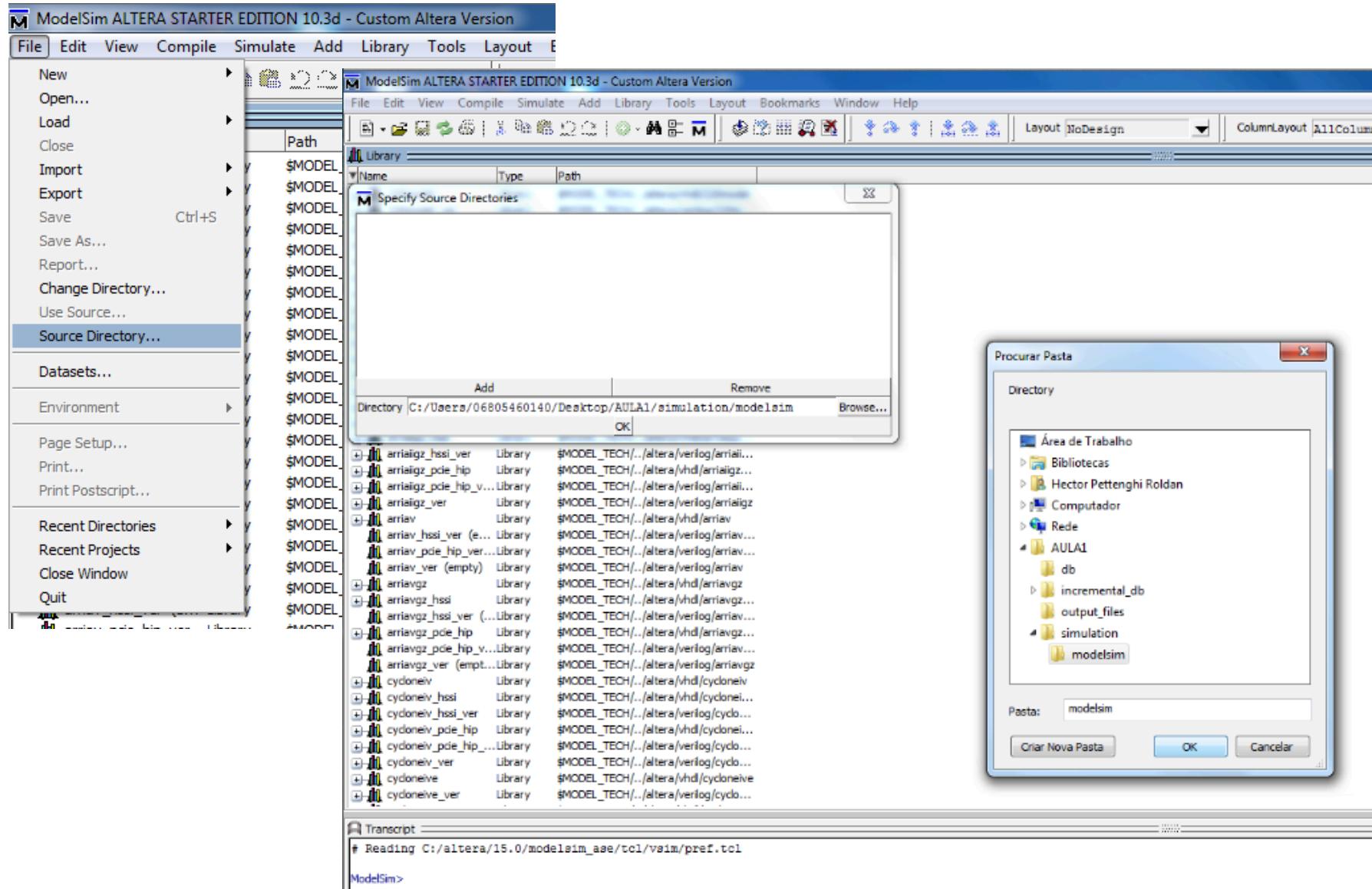
13. Programação – FPGA é carregado com circuito, configurando fisicamente elementos de processamento e roteamento.

Tools – Programmer. Hardware Setup – USB-Blaster. Start!

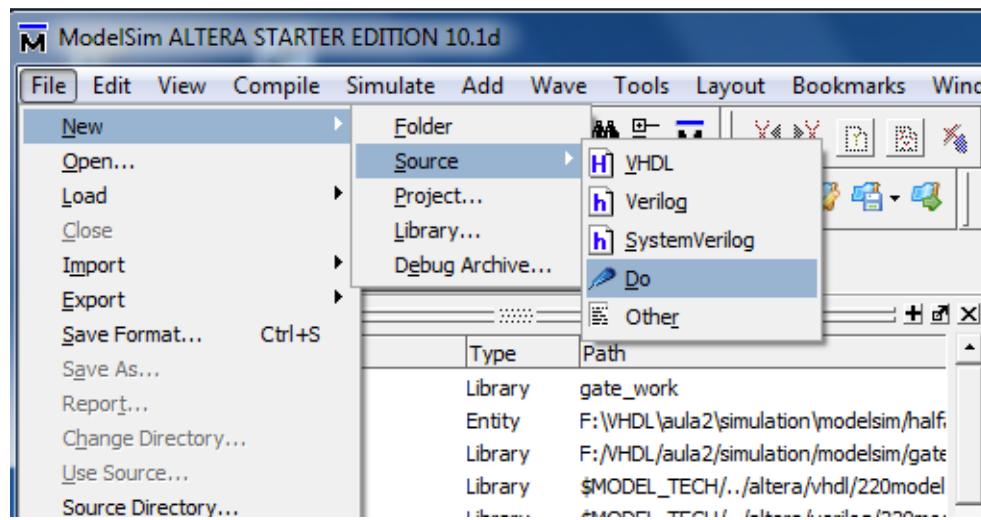


Scripts de simulação MODELSIM

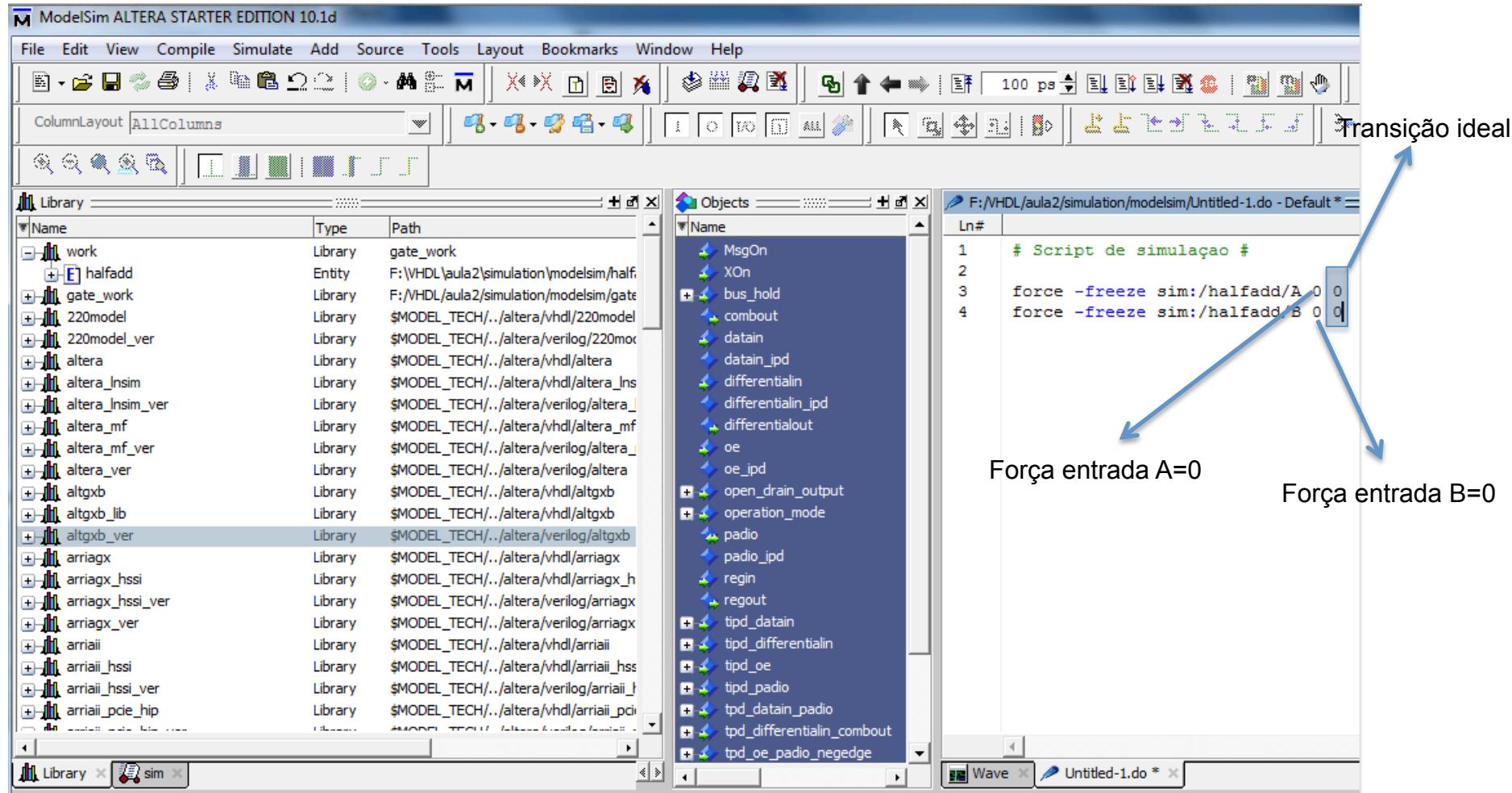
Passo 1: Mudar diretório fonte



Passo 2: Criar um arquivo script .do



Passo 3: Forçar as entradas aos valores desejados



Passo 3: Forçar as entradas aos valores desejados

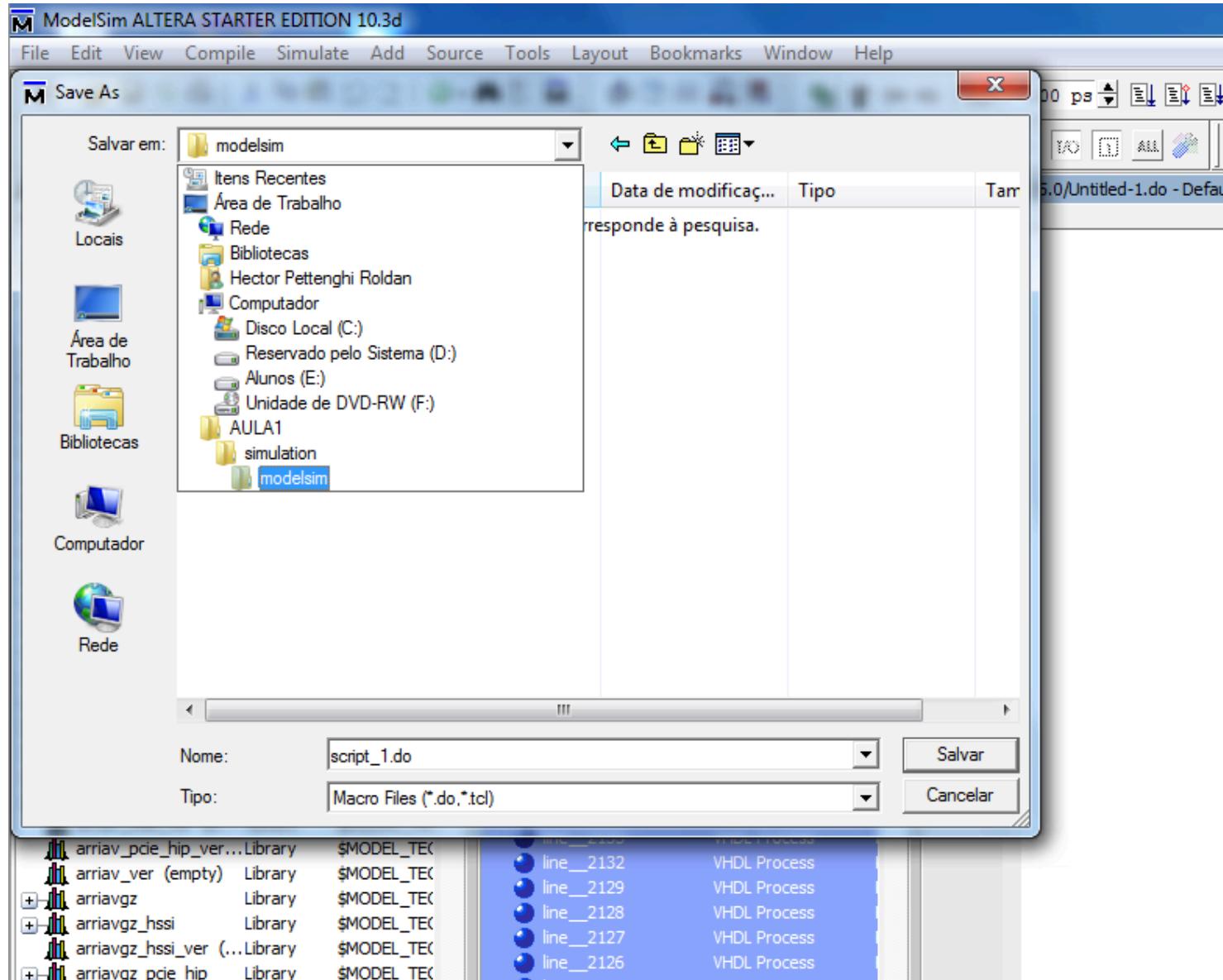
The screenshot shows the ModelSim ALTERA Starter Edition 10.1d interface. The left panel displays the Library browser with various Altera IP cores and models. The center panel shows the Object browser listing components like MsgOn, XOn, bus_hold, and various differential and open-drain output types. The right panel displays the simulation script 'Untitled-1.do' with the following content:

```
# Script de simulação #
force -freeze sim:/halfadd/A 0 0
force -freeze sim:/halfadd/B 0 0
run
force -freeze sim:/halfadd/A 0 0
force -freeze sim:/halfadd/B 1 0
run
force -freeze sim:/halfadd/A 1 0
force -freeze sim:/halfadd/B 0 0
run
force -freeze sim:/halfadd/A 1 0
force -freeze sim:/halfadd/B 1 0
run
```

A blue arrow points from the 'Objects' window to the 'Wave' window, which is currently active. The 'Wave' window shows the waveform for the 'halfadd' component, with two inputs (A and B) and one output (Sum).

Todas as combinações possíveis

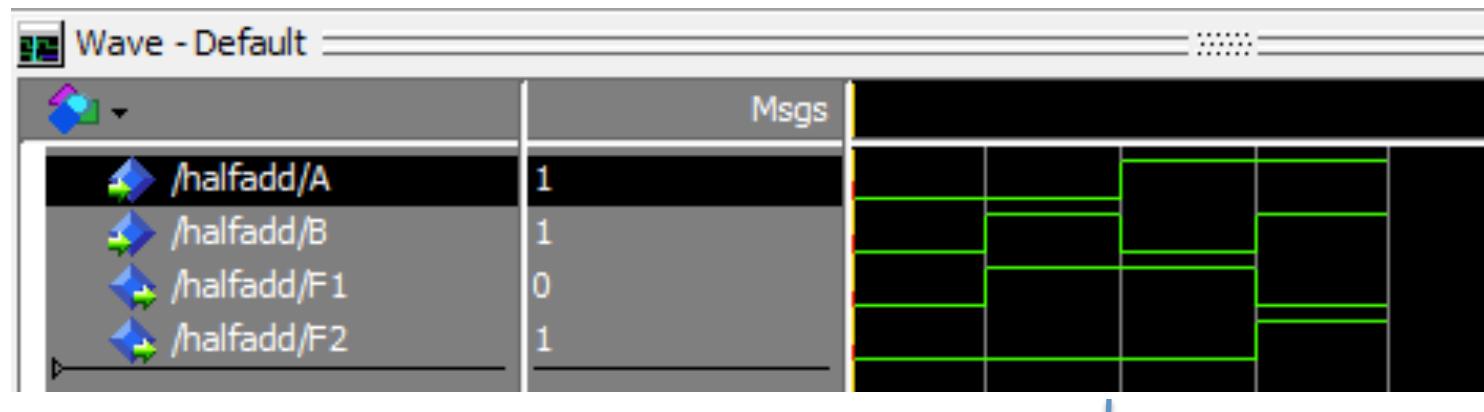
Passo 4: Salvar o script com a extensão .do



Passo 4: Executar o script .do

```
Transcript  
VSIM 9> restart  
VSIM 10> do script_1.do
```

Janela de comando na base do Modelsim



Visualização da simulação com todas as combinações possíveis de entrada