Sistemas Digitais 2

Introdução ao VHDL e Fluxo de projeto em FPGAs

Prof. Daniel M. Muñoz Arboleda

FGA - UnB

Agenda

- Linguagens de descrição de hardware
- Estrutura básica do VHDL
  - Library declaration
  - Entity declaration
  - Architecture
- Fluxo de projeto
  - Descrição do circuito
  - Síntese lógica
  - Mapeamento
  - Place and Route (PAR)
- Testbench e simulação

Linguagens de Descrição de Hardware

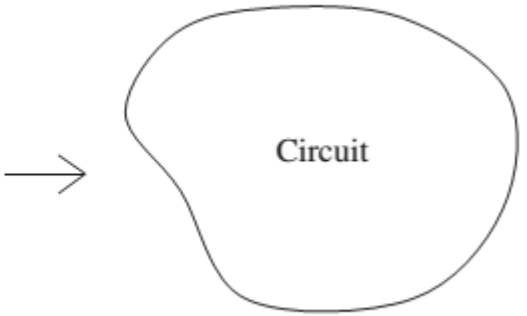
- Permitem elevado nível de abstração em descrições comportamentais.
- Metodologia de projeto top-down, utilizando ferramentas de síntese automática.
- Representação textual, possibilitando portabilidade, edição e documentação.
- Possibilidade de desenvolvimento por meio de esquemáticos e representações gráficas.
- Construção de modelo de simulação e síntese.

Linguagens de Descrição de Hardware

- HDLs são linguagens de simulação que permitem descrever o hardware de sistemas e simular suas operações.

- HDLs permitem também a síntese de circuitos:

```
ENTITY full_adder IS
PORT (a, b, cin: IN BIT;
      s, cout: OUT BIT);
END full_adder;
-------------------------------------
ARCHITECTURE dataflow OF full_adder IS
BEGIN
   s <= a XOR b XOR cin;
   cout <= (a AND b) OR (a AND cin) OR
           (b AND cin);
END dataflow;
```



- Por que HDLs?
  - Antes: diagramas de blocos e esquemáticos
  - Com HDLs é possível projetar sistemas maiores e mais complexos

Alguns HDLs

- VHDL - **V**ery High Speed Integrated Circuits **H**ardware **D**escription **L**anguage
- Verilog (IEEE 1364 ' 95)
- SystemC
- ABEL – **A**dvanced **B**oolean **E**quation **L**anguage

VHDL

O que é VHDL? **V**ery High Speed Integrated Circuits **H**ardware **D**escription **L**anguage

VHDL e uma linguagem de descrição de *hardware.* Descreve o comportamento de um circuito eletrônico ou sistema, a partir do qual o circuito físico pode ser implementado.

História:
• VHSIC (Very High Speed Integrated Circuits) foi uma iniciativa do Departamento de Defesa dos Estados Unidos (DARPA), em meados da década de 1980, para documentar o comportamento de ASICs.
• Isto levou ao aparecimento da linguagem VHDL, desenvolvida para substituir os complexos manuais que descreviam o funcionamento dos ASICs.
• A linguagem VHDL foi posta em domínio público e foi padronizada pela IEEE no ano 1987.
• IEEE 1164 é um pacote VHDL com multi valores lógicos (std_logic_1164)
• Alterações e aprimoramento da linguagem. IEEE lança um padrão atualizado no ano 1993.
• Em 2008 foi aprovada a mais recente versão, IEEE 1076-2008, que considera extensões para sinais analógicas e mistas, VHDL-AMS (analog and mixed signal).

Características do VHDL

- Decomposição hierárquica
- Cada elemento possui interface bem definida e especificação funcional precisa
- Especificações podem ser representadas por algoritmos ou pela estrutura de hardware
- É possível modela concorrência, *timing* e *clocking*
- Operação lógica e comportamento podem ser simulados
- O comportamento do circuito pode ser simulado considerando atrasos e delays (*timing*)

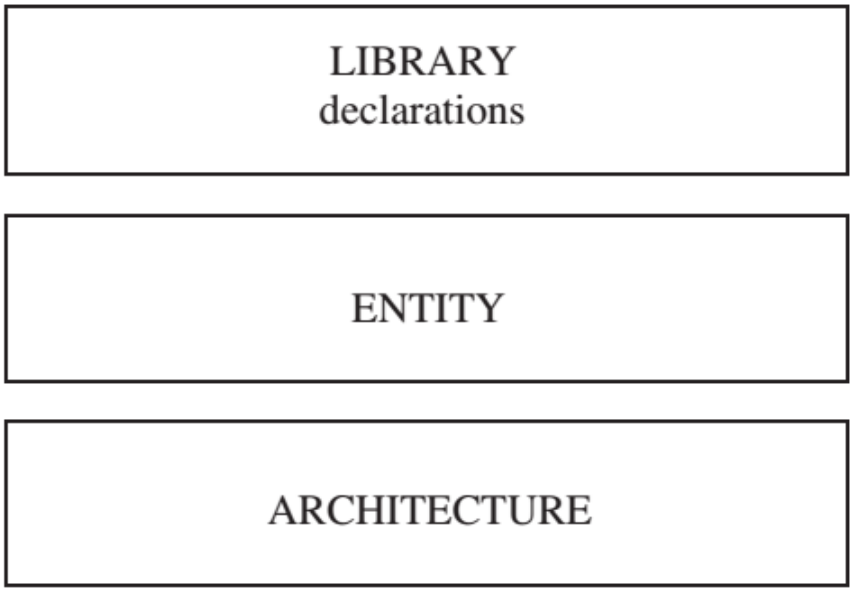
Agenda

- Linguagens de descrição de hardware
- **Estrutura básica do VHDL**
  - Library declaration
  - Entity declaration
  - Architecture
- Fluxo de projeto
  - Descrição do circuito
  - Síntese lógica
  - Mapeamento
  - Place and Route (PAR)
- Testbench e simulação

Estrutura do VHDL

Consultar livro Circuit Design with VHDL, Volnei Pedroni, 2004

- **Library:** contem todas as bibliotecas usadas no projeto, ex: *ieee*, *std*, *work*, etc.
- **Entity:** declaração de portas de entrada e saída (I/O pins).
- **Architecture:** descrição detalhada do comportamento interno ou estrutura do módulo
- A arquitetura de uma entidade pode instanciar outras entidades.



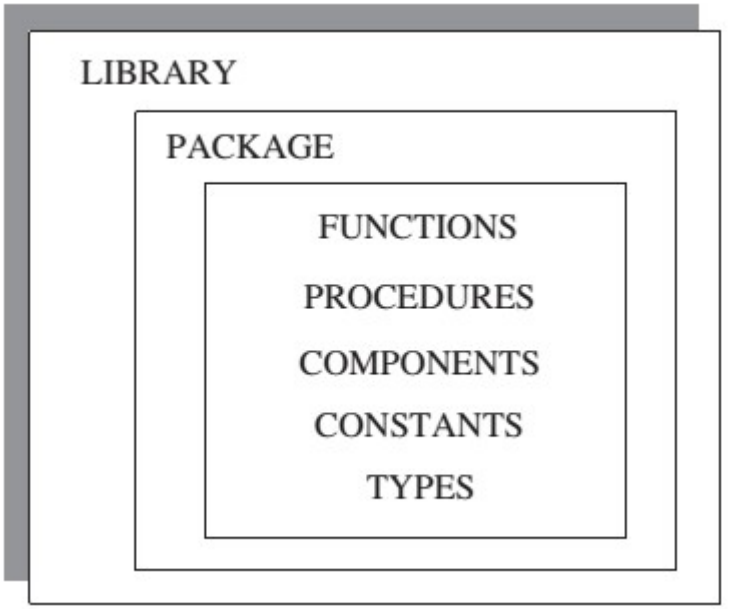
Estrutura do VHDL

Library declaration: Para declarar as bibliotecas são necessárias duas linhas de código, uma contendo o nome da biblioteca e a outra contendo a diretiva *USE* para indicar quais *packages* (pacotes) serão usados.

```
LIBRARY library_name;
USE library_name.package_name.package_parts;
```

Comumente três *packages* são usados os quais contêm tipos de dados, parâmetros, funções, *procedures* e componentes.

- *ieee.std_logic_1164*
- *standard*
- *work*



Library declaration

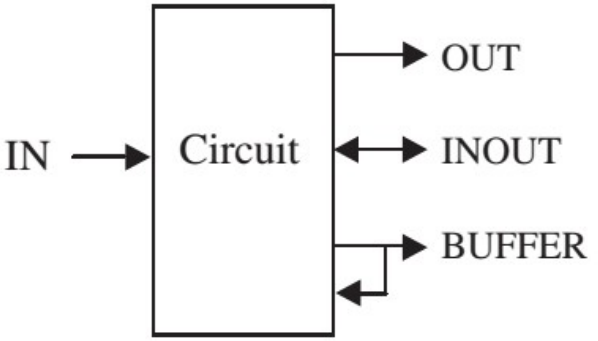
O pacote *std_logic_1164* da biblioteca *ieee* especifica 8 ou 9 níveis lógicos. Esse pacote possui alguns pacotes internos, dentre os quais destacam-se:

- *std_logic_arith*: especifica os tipos de dados SIGNED e UNSIGNED relacionados com operações aritméticas e operações de comparação. Adicionalmente, especifica algumas *funções* de conversão de dados, tai como: *conv_integer*(*p*), *conv_unsigned*(*p*,*b*), *conv_signed*(*p*,*b*), *conv_std_logic_vector*(*p*,*b*).
- *std_logic_signed*: contem *funções* que permitem a operação com dados std_logic_vector do tipo SIGNED.
- *std_logic_unsigned*: contem *funções* que permitem a operação com dados std_logic_vector do tipo UNSIGNED.

Estrutura do VHDL

Entity declaration: Entidade é uma lista com a especificação de todas as portas/pinos de entrada e saída do circuito.

```
ENTITY entity_name IS
   PORT (
      port_name : signal_mode signal_type;
      port_name : signal_mode signal_type;
      ...);
END entity_name;
```



O modo pode ser IN, OUT, INOUT ou BUFFER. As entradas IN e OUT são unidirecionais, a entrada INOUT é bidirecional e BUFFER indica uma saída que pode ser usada (lida) internamente.

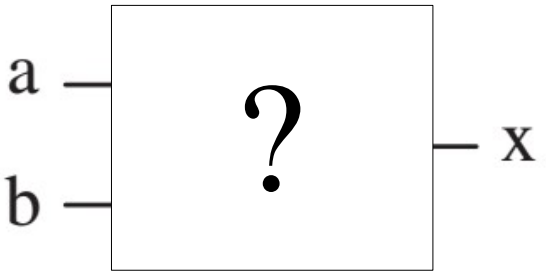
O tipo de dado pode ser *bit, bitvector, std_logic, std_logic_vector, integer*

Estrutura do VHDL

Entity declaration: Entidade é uma lista com a especificação de todas as portas/pinos de entrada e saída do circuito.

Exemplo: a seguir mostra-se a entidade de uma porta nand de duas entradas, cada uma de um bit e uma saída de um bit. O tipo de dado é *std_logic*.

```
1  entity porta_nand2 is port(
2      a, b: in std_logic;
3      x: out std_logic);
4  end porta_nand2;
5
6
```



Estrutura do VHDL

Architecture. A arquitetura é a descrição do comportamento do circuito. A sintaxe é a seguinte.

```
ARCHITECTURE architecture_name OF entity_name IS
    [declarations]
BEGIN
    (code)
END architecture_name;
```

Entre a primeira linha (**architecture**) e o **begin** declaram-se as funções, procedimentos, sinais, constantes e tipos a serem usados na descrição.

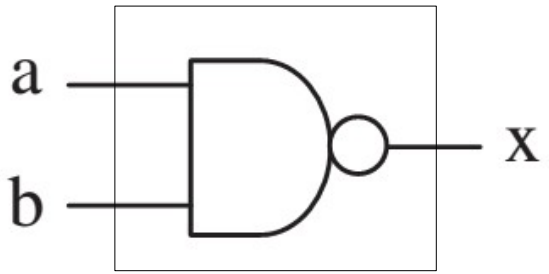
Entre o **begin** e o **end** é feita a descrição do comportamento do circuito, estabelecendo as relações entre as entradas e saídas do circuito. Pode-se usar lógica combinacional e sequencial para descrever o circuito.

Estrutura do VHDL

Architecture.

Exemplo: a seguir mostra-se a descrição de uma porta nand2 (lógica combinacional)

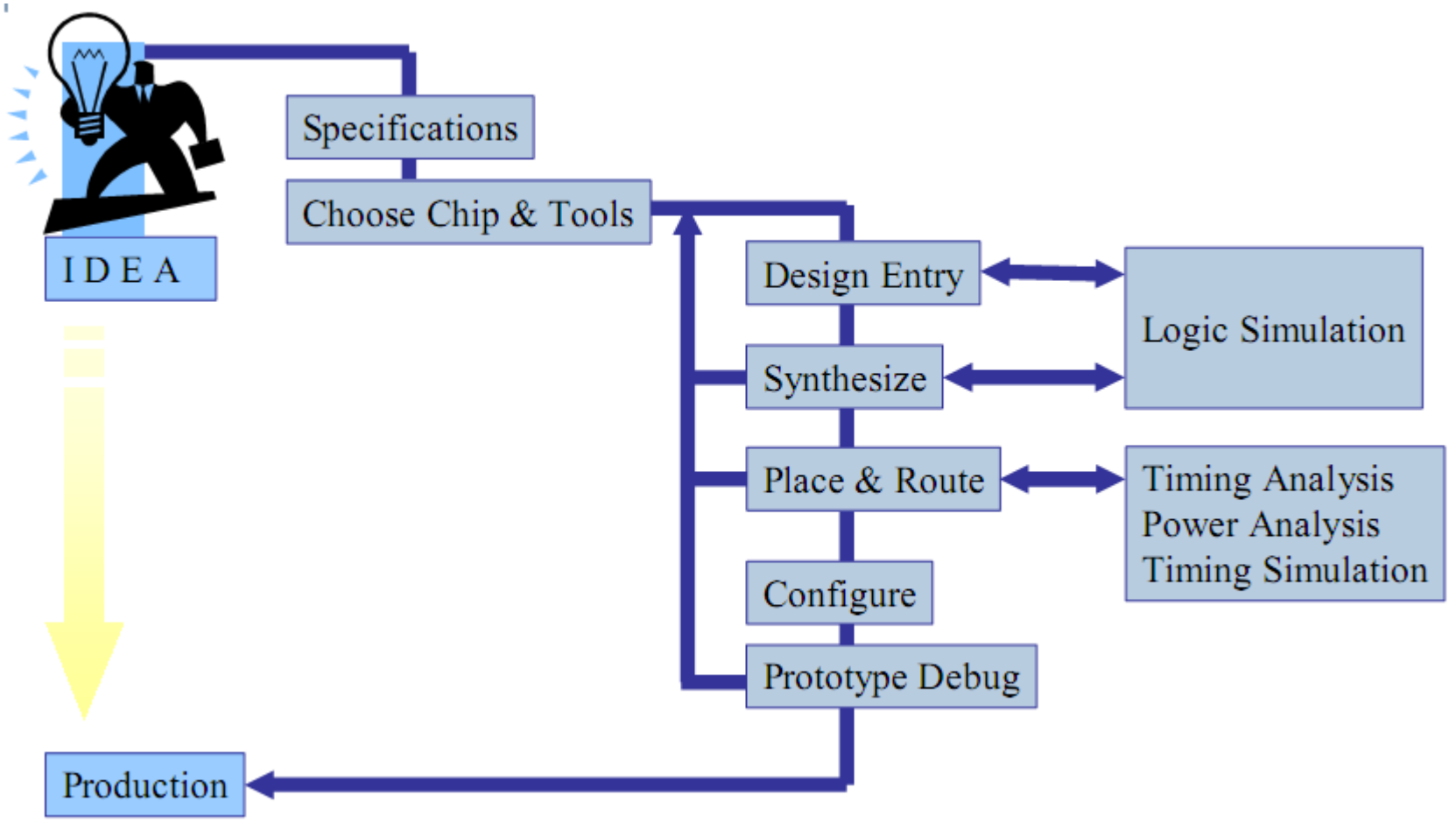
```
6
7  architecture behavioral of porta_nand2 is
8  begin
9      x <= a nand b;
10 end behavioral;
11
```



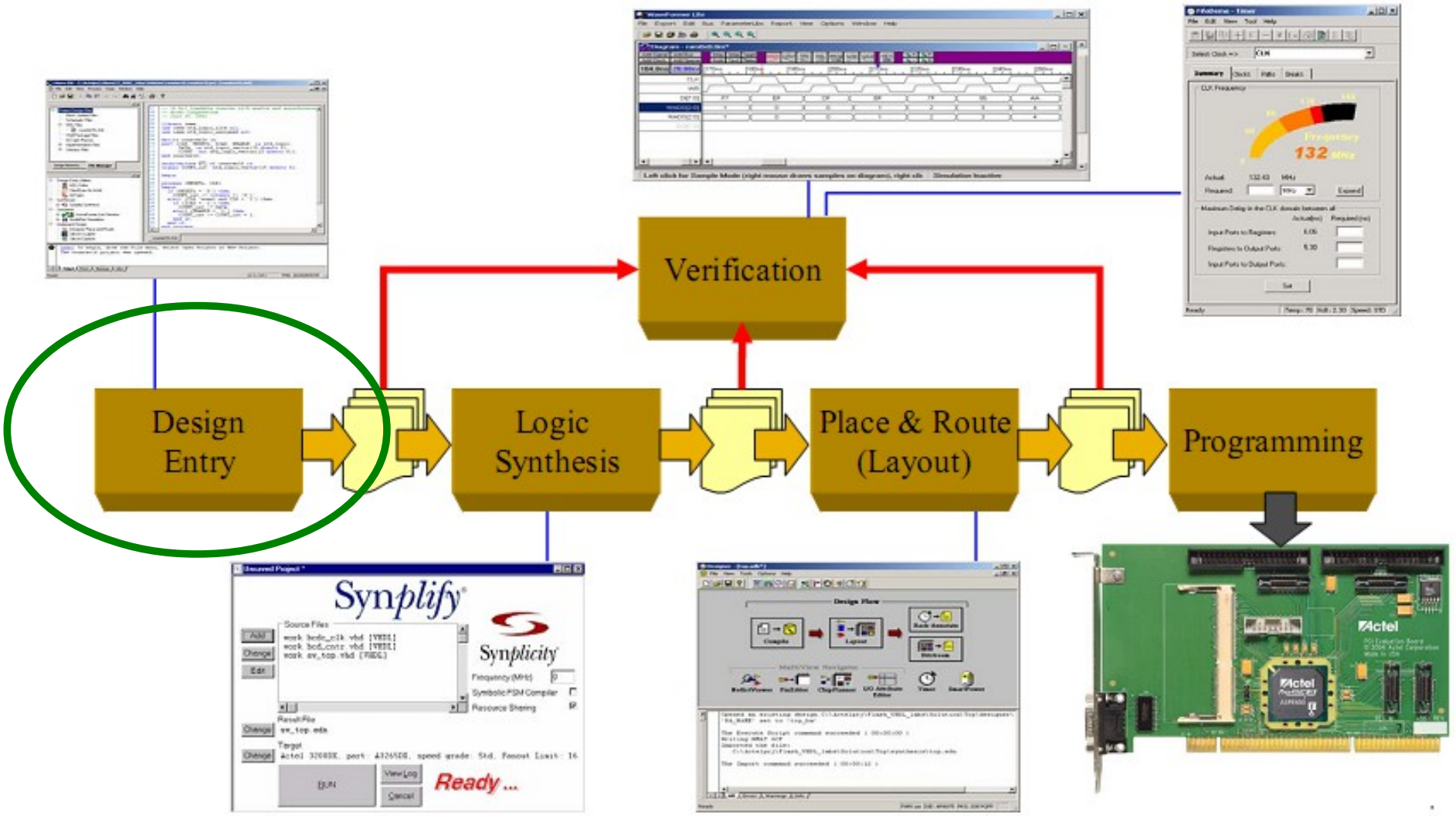
Agenda

- Linguagens de descrição de hardware
- Estrutura básica do VHDL
  - Library declaration
  - Entity declaration
  - Architecture
- **Fluxo de projeto**
  - Descrição do circuito
  - Síntese lógica
  - Mapeamento
  - Place and Route (PAR)
- Testbench e simulação

Fluxo de Projeto com FPGAs



Fluxo de Projeto com FPGAs



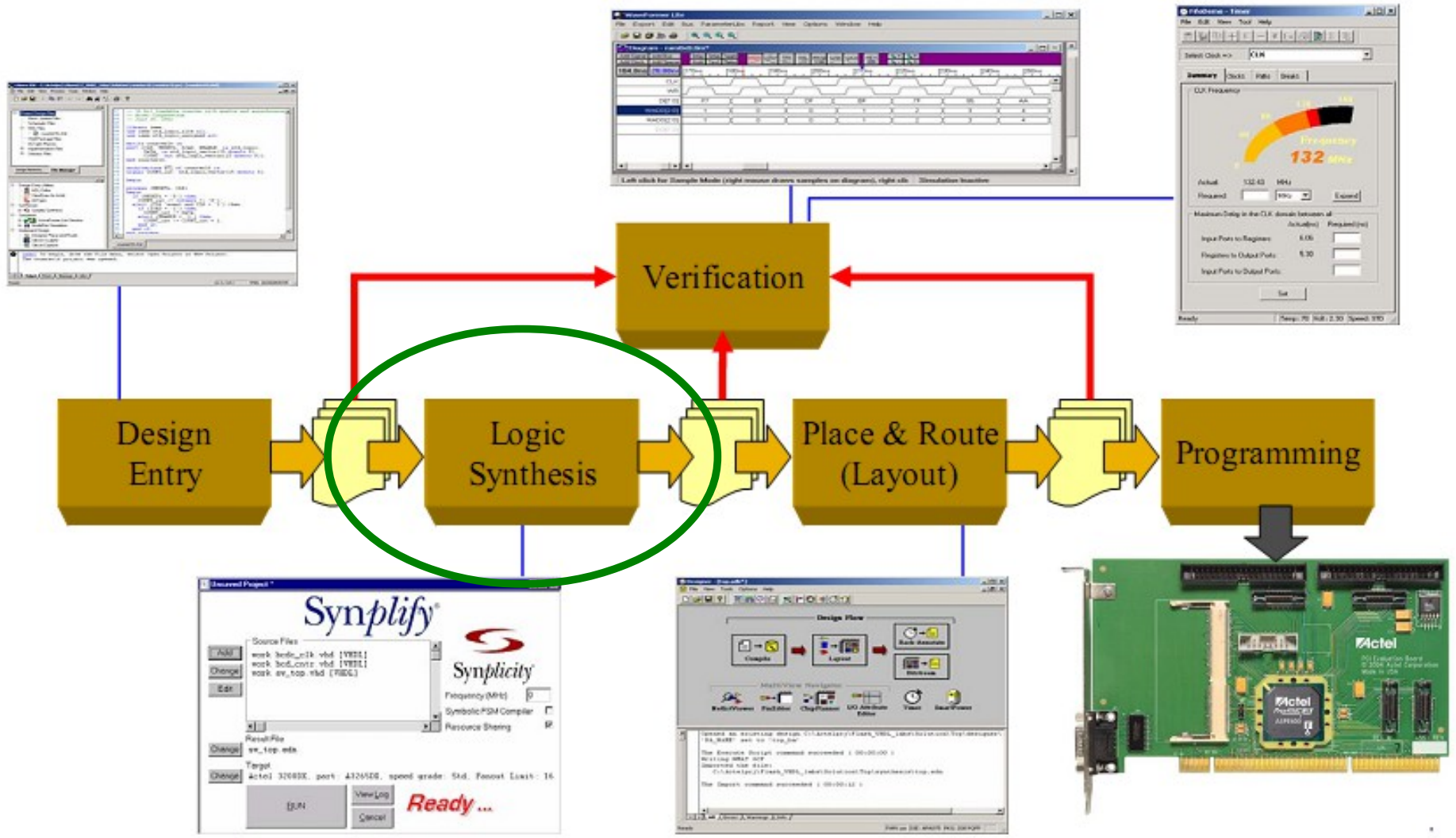
Descrição do circuito (design entry)

- Nesta fase é usada uma linguagem de descrição de hardware para descrever o comportamento do circuito.

- O projetista deve ter uma ideia sólida do comportamento do circuito. Muitas vezes um modelo em alto nível através de processos ou módulos comunicantes é requerido.

- Lógica combinacional e/ou sequencial pode ser usada para estabelecer as relações entre entradas e saídas.

Fluxo de Projeto com FPGAs

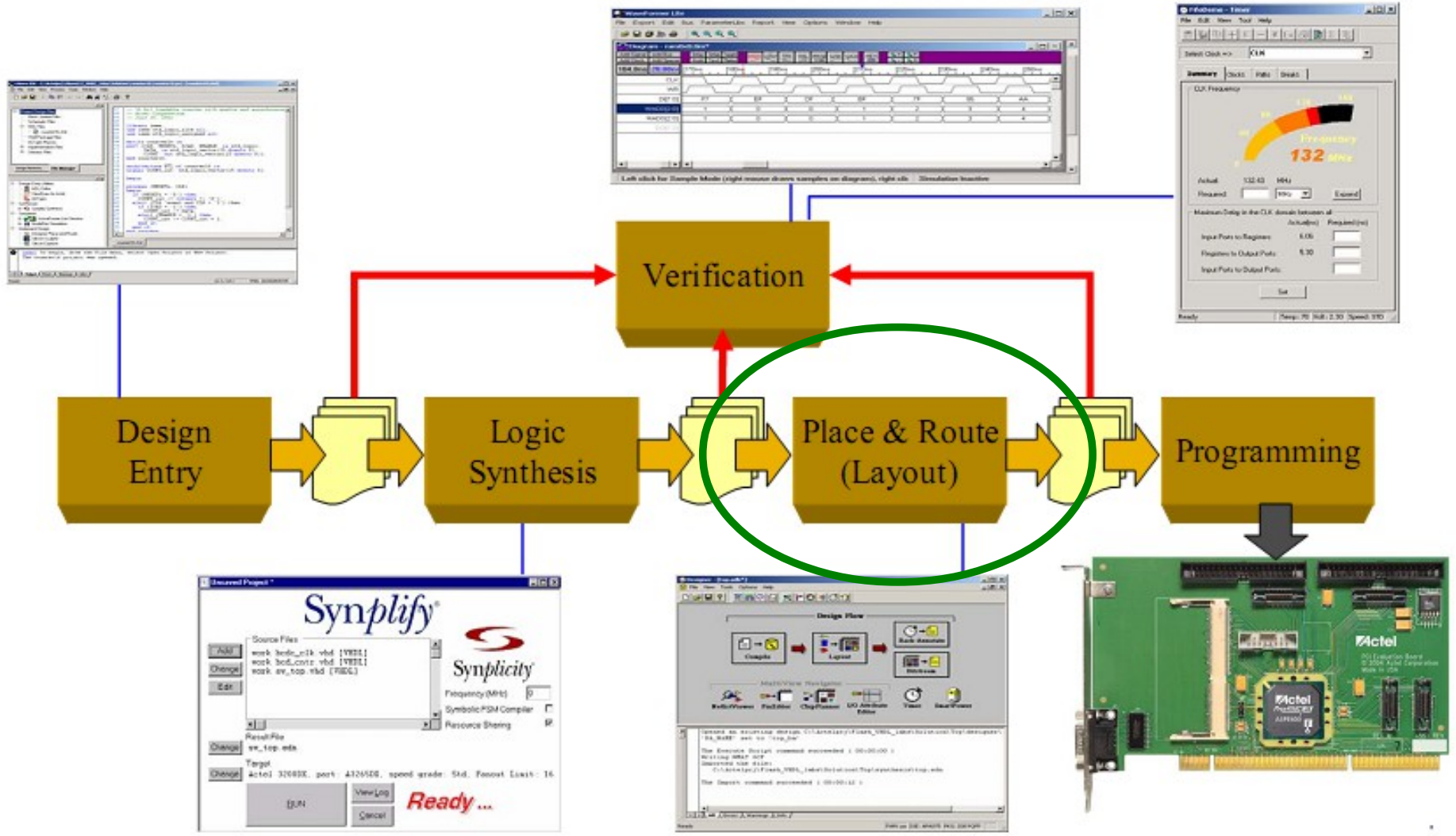


Síntese lógica

• O processo de **síntese lógica** é a transformação do código que descreve o comportamento do circuito em equações booleanas. A saída do processo de síntese é um *Netlist* que contém a lista de portas lógicas e registradores e as conexões entre eles que descrevem o circuito.

• Geralmente, o código HDL é traduzido pela ferramenta de **síntese lógica** em nível de transferência de registradores (do inglês *Register Transfer Level* – RTL). No nível RTL o circuito é descrito em termos de fluxo de sinais ou transferência de dados entre registradores e entre blocos que implementam operações lógicas.

Fluxo de Projeto com FPGAs



Localização e roteamento (Place and Route - PAR)

- O processo de **place and route** (**PAR**) é mapeamento das portas lógicas e registradores usando os elementos disponíveis no FPGA, tais como Slices, LUTs, Flip-flops, blocos DSPs, blocos RAM, etc. Uma vez mapeados os componentes é feito o roteamento (conexão) dos mesmos.

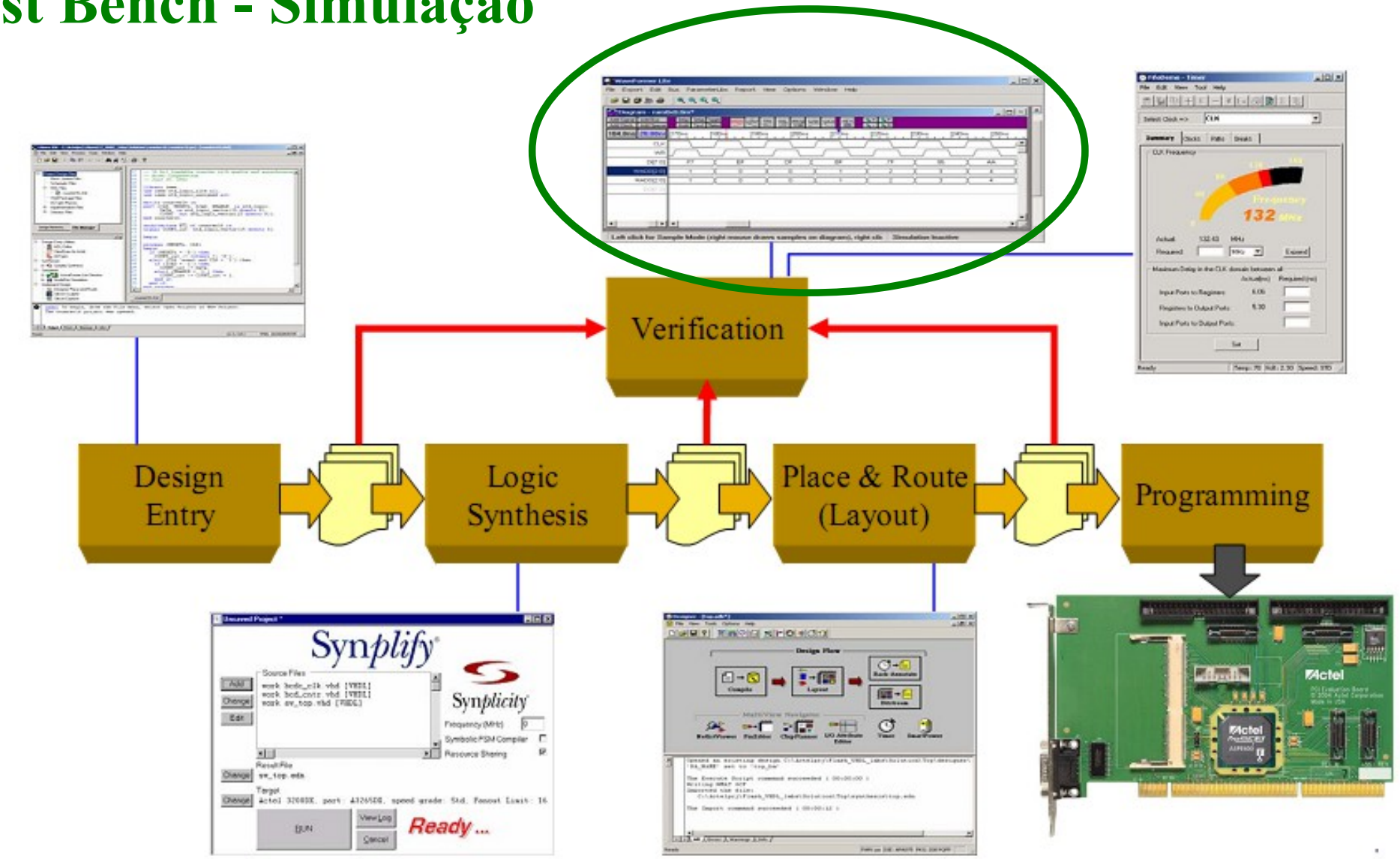
- O processo **PAR** está baseado em algoritmos de otimização que localizam e conectam os componentes de forma a minimizar os caminhos críticos, maximizando a frequência de operação, ou minimizando o consumo de energia.

- O processo **PAR** requere que as portas de entrada e saída do circuito tenham sido mapeadas no dispositivo FPGA. A saída do processo **PAR** é um *Netlist* com maior grau de detalhe, descrevendo em baixo nível o circuito (onde e como se conecta cada porta lógica e cada registrador).

Agenda

- Linguagens de descrição de hardware
- Estrutura básica do VHDL
  - Library declaration
  - Entity declaration
  - Architecture
- Fluxo de projeto
  - Descrição do circuito
  - Síntese lógica
  - Mapeamento
  - Place and Route (PAR)
- **Testbench e simulação**

Test Bench - Simulação



Test Bench - Simulação

Não há declaração de portas

Declara o UUT como componente

Cria sinais para entradas e saídas

Instancia o componente usando *port map*

Gera as combinações dos bits de entrada

```
ENTITY implica_tb IS
END implica_tb;

ARCHITECTURE behavior OF implica_tb IS

    COMPONENT implica
    PORT(x : IN  std_logic;
         y : IN  std_logic;
         z : OUT  std_logic);
    END COMPONENT;

   signal sx, sy, sz : std_logic;

BEGIN
   uut: implica PORT MAP (
          x => sx,
          y => sy,
          z => sz);

   stim_proc: process
   begin
      sx <= '0'; sy <= '0';
      wait for 20 ns;
      sx <= '0'; sy <= '1';
      wait for 20 ns;
      sx <= '1'; sy <= '0';
      wait for 20 ns;
      sx <= '1'; sy <= '1';
      wait;
   end process;
END;
```

Test Bench - Simulação