Pontifícia Universidade Católica do Rio Grande do Sul
Faculdade de Informática - FACIN

LABORG

Prof. Dr. Rafael Garibotti

❖ Baseado no material cedido pelos Profs. **Dr. Fernando Moraes** e **Dr. Ney Calazans**

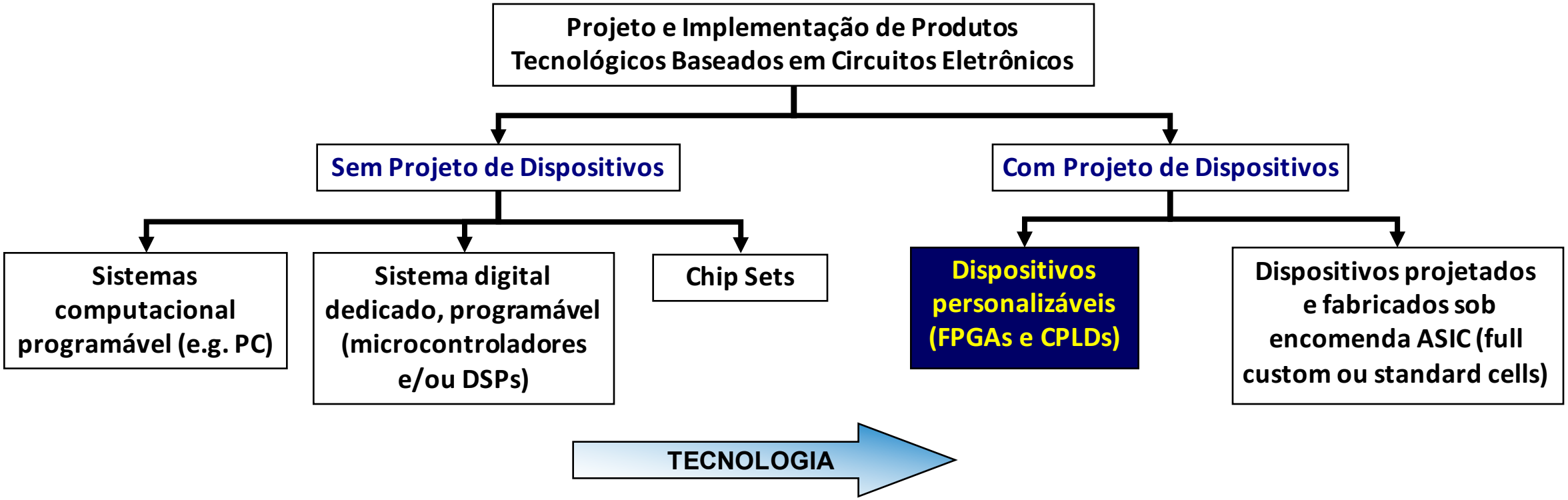
AULA SOBRE:

INTRODUÇÃO A FPGAs E PROTOTIPAÇÃO DE HARDWARE

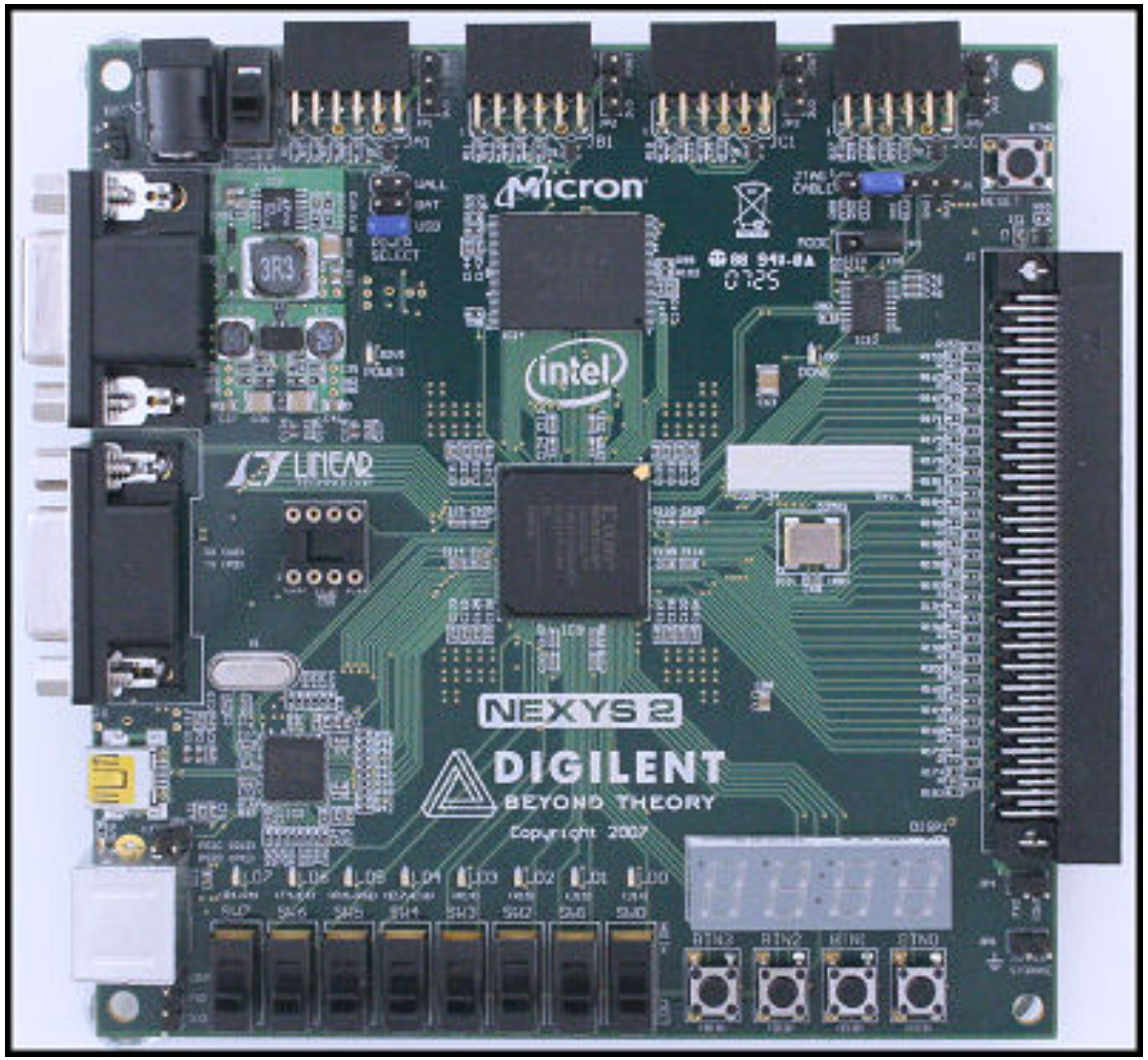
TEORIA – ESTRUTURA DE FPGAs

O QUE SÃO FPGAs?

- FPGAs permitem implementar circuitos digitais diretamente de HDLs, sem os custos de fabricação de chips!



EXEMPLO DE UM FPGA

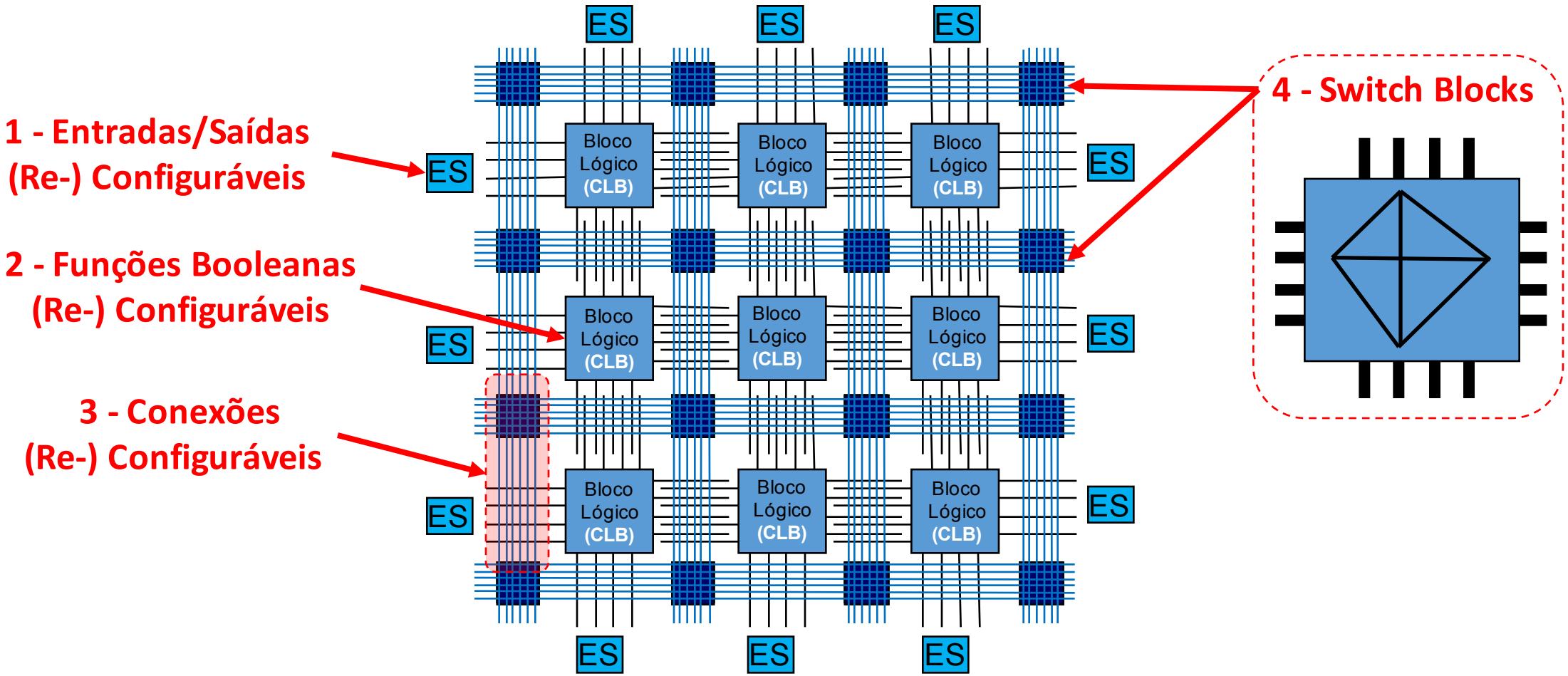


UM POUCO DA HISTÓRIA DE FPGAs

- Primeiro vieram PROMs e PLDs (Programmable logic devices), matrizes de portas (re-) configuráveis.
- Algumas patentes de coisas parecidas com FPGAs (field programmable gate array) surgiram no final dos anos 80 e início dos anos 90 (Casselman, Page, Peterson).
- Os fundadores da Xilinx, Ross Freeman e Bernard Vonderschmitt, inventaram o primeiro FPGA comercial em 1985 – o XC2064.
- O XC2064 tinha 64 blocos lógicos configuráveis e interconexões configuráveis entre os blocos lógicos.
- O XC2064 só tinha blocos lógicos configuráveis (CLBs), cada um com duas LUTs de 3 entradas.

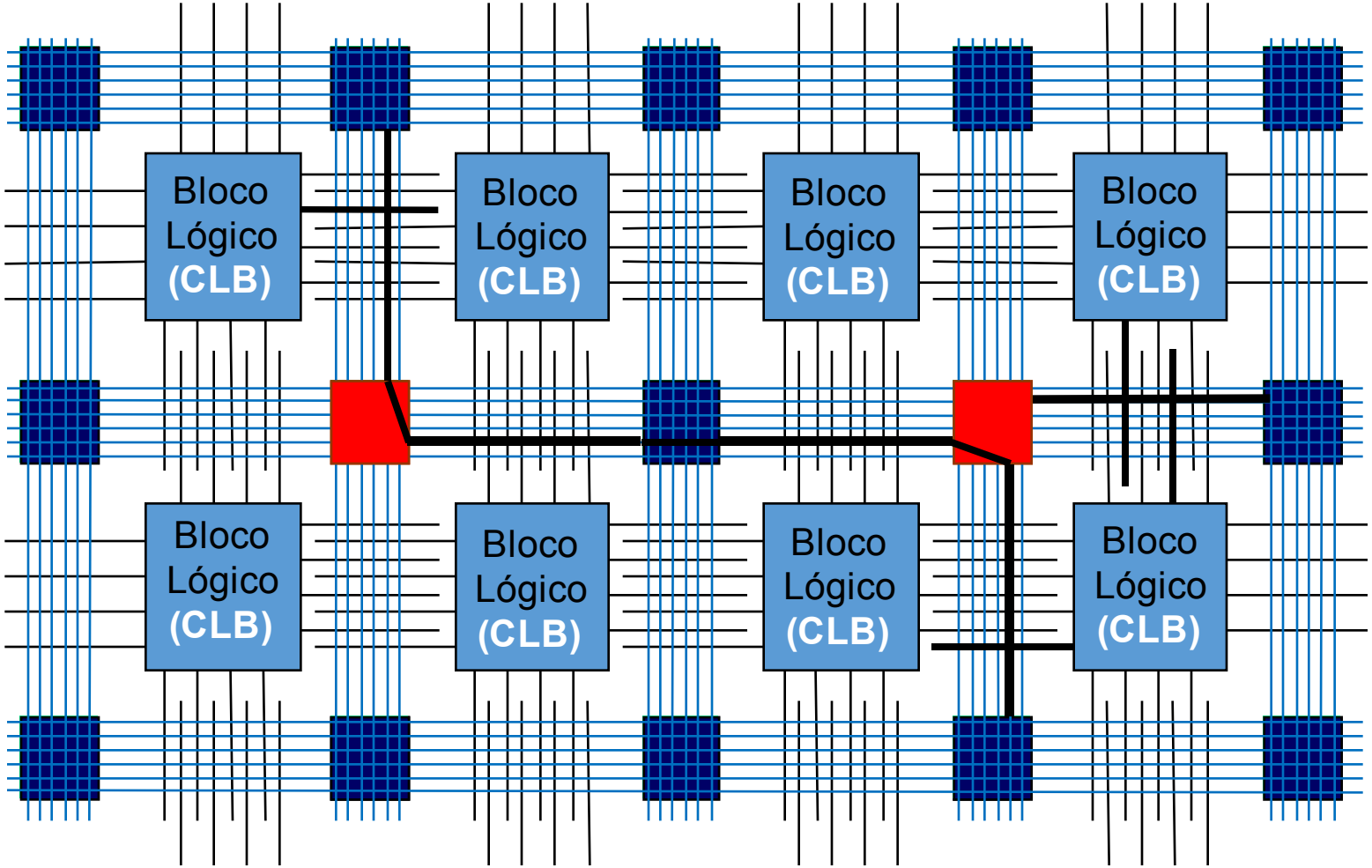
FPGAs – CONCEITOS BÁSICOS

- Matriz de CLBs (*configurable logic blocks*) interconectados por matrizes de chaveamento.



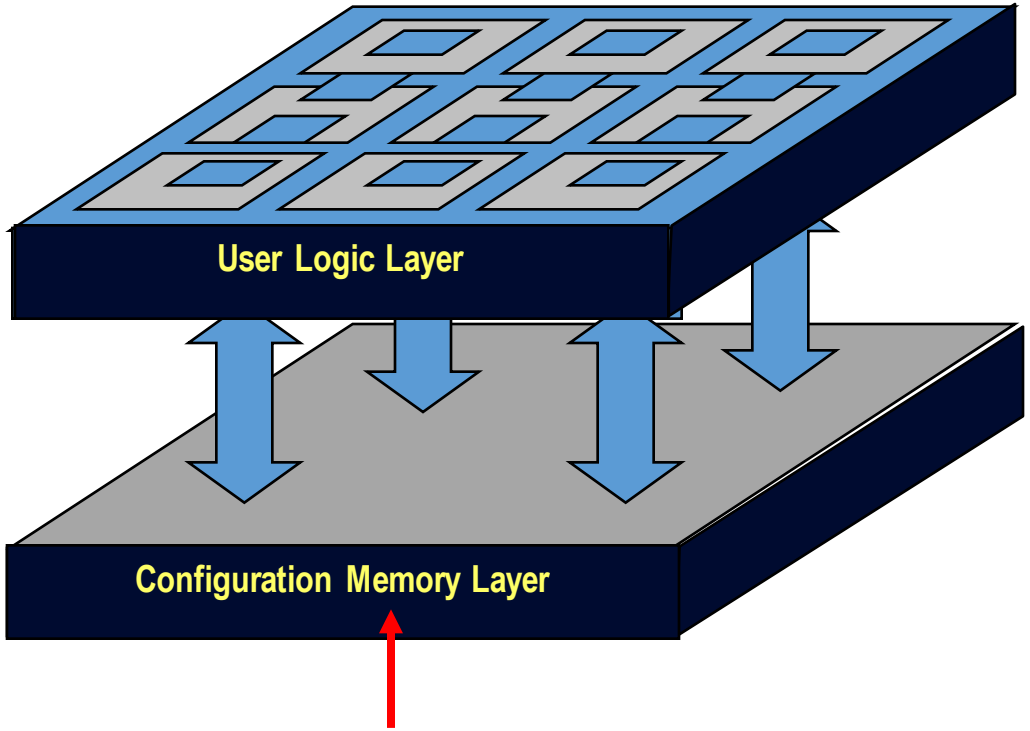
FPGAs – CONCEITOS BÁSICOS

- Exemplo de conexão entre duas redes:



FPGAs – CONFIGURAÇÃO (RAM-BASED)

- FPGA deve ser visto como “duas camadas”
  - ✓ Memória de configuração.
  - ✓ Lógica do usuário.
- Memória de configuração define:
  - ✓ Toda a fiação da lógica do usuário.
  - ✓ Definição das funções lógicas (LUTs).
  - ✓ Interface externas e internas (µproc).
  - ✓ Configuração de memórias.
  - ✓ Conteúdo de memórias.
  - ✓ Configuração dos pinos de E/S.



Por exemplo, Virtex 4: memória de configuração entre 1 MB – 4 MB

TECNOLOGIAS DE CONFIGURAÇÃO

- Algumas das diferentes tecnologias usadas para definir o comportamento de um FPGA:
  - ✓ Antifusível: Configuração uma única vez.
  - ✓ (E)EPROM: Configuração um número limitado de vezes, mantida com o chip desconectado da alimentação.
  - ✓ SRAM: Configuração deve ser realizada cada vez que o FPGA for alimentado.

LUT – O GERADOR UNIVERSAL DE FUNÇÕES

- LUT (*Look-Up Table*) – Um exemplo de Bloco com Função Booleana Reconfigurável
  - ✓ Uma porção de hardware configurável/reconfigurável capaz de implementar **qualquer** tabela verdade de *n* entradas.
  - ✓ Para n=4:

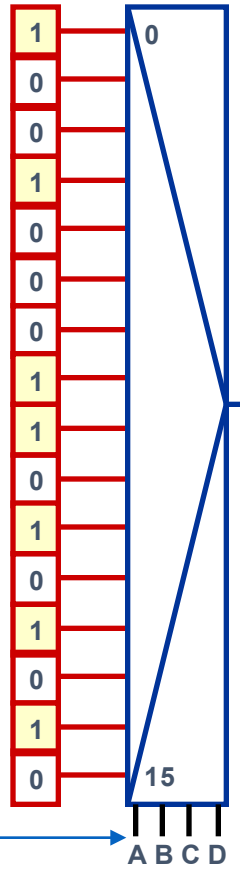
$$2^{(2)^4} = 65.536 \text{ funções implementáveis}$$

  - Altamente flexível
  - Método mais utilizado (Xilinx, Altera e outros)

FPGAs – LUT – O GERADOR UNIVERSAL DE FUNÇÕES

- Implementação física de uma LUT4:

A tabela verdade da função é armazenada em uma memória durante a configuração do FPGA



As entradas (variáveis Booleanas) controlam um multiplexador 2^n:1

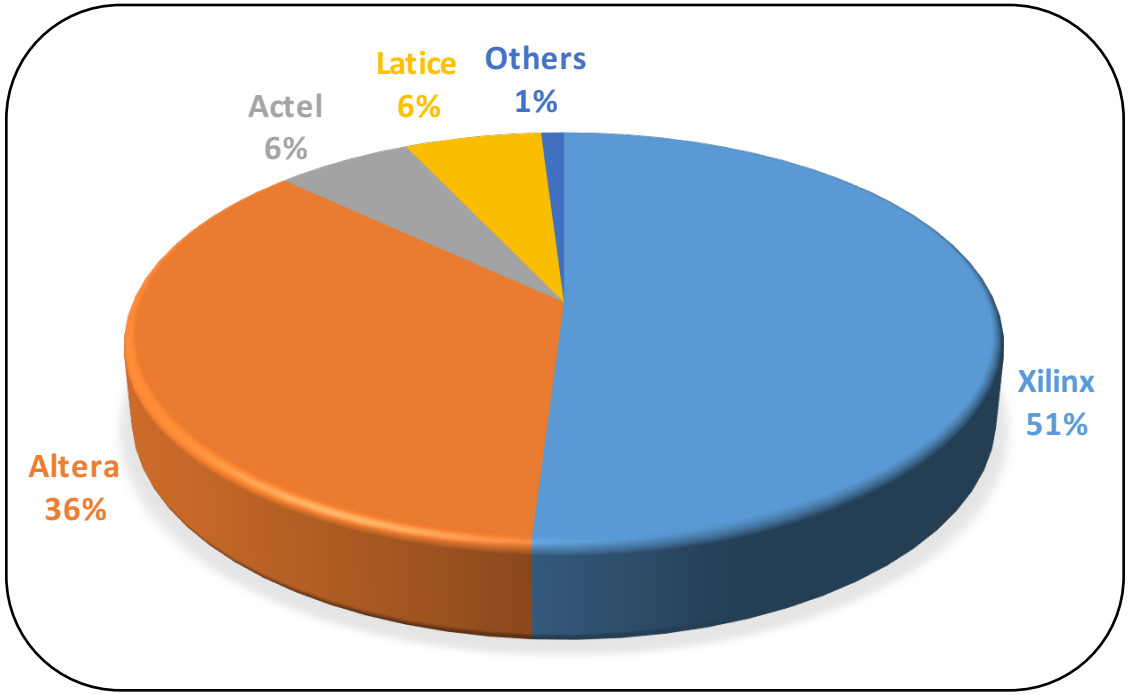
$$F(A,B,C,D) = \bar{A}.\bar{B}.\bar{C}.\bar{D} + \bar{A}.C.D + A.\bar{D}$$

$$F(A,B,C,D) = \sum(0,3,7,8,10,12,14)$$

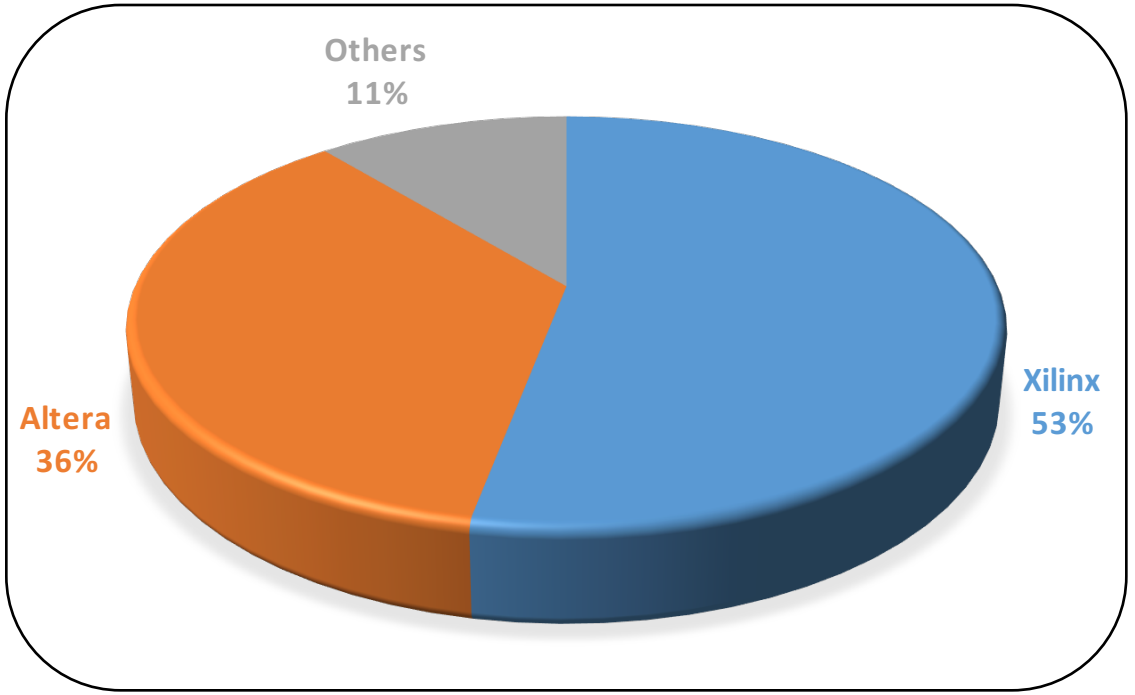
Considerando 150 transistores por LUT4, temos para 50.000 LUTS ➔ 7.500.000 transistores!!!

O MERCADO DE FPGAs

➢ PLD Segment



➢ FPGA Segment



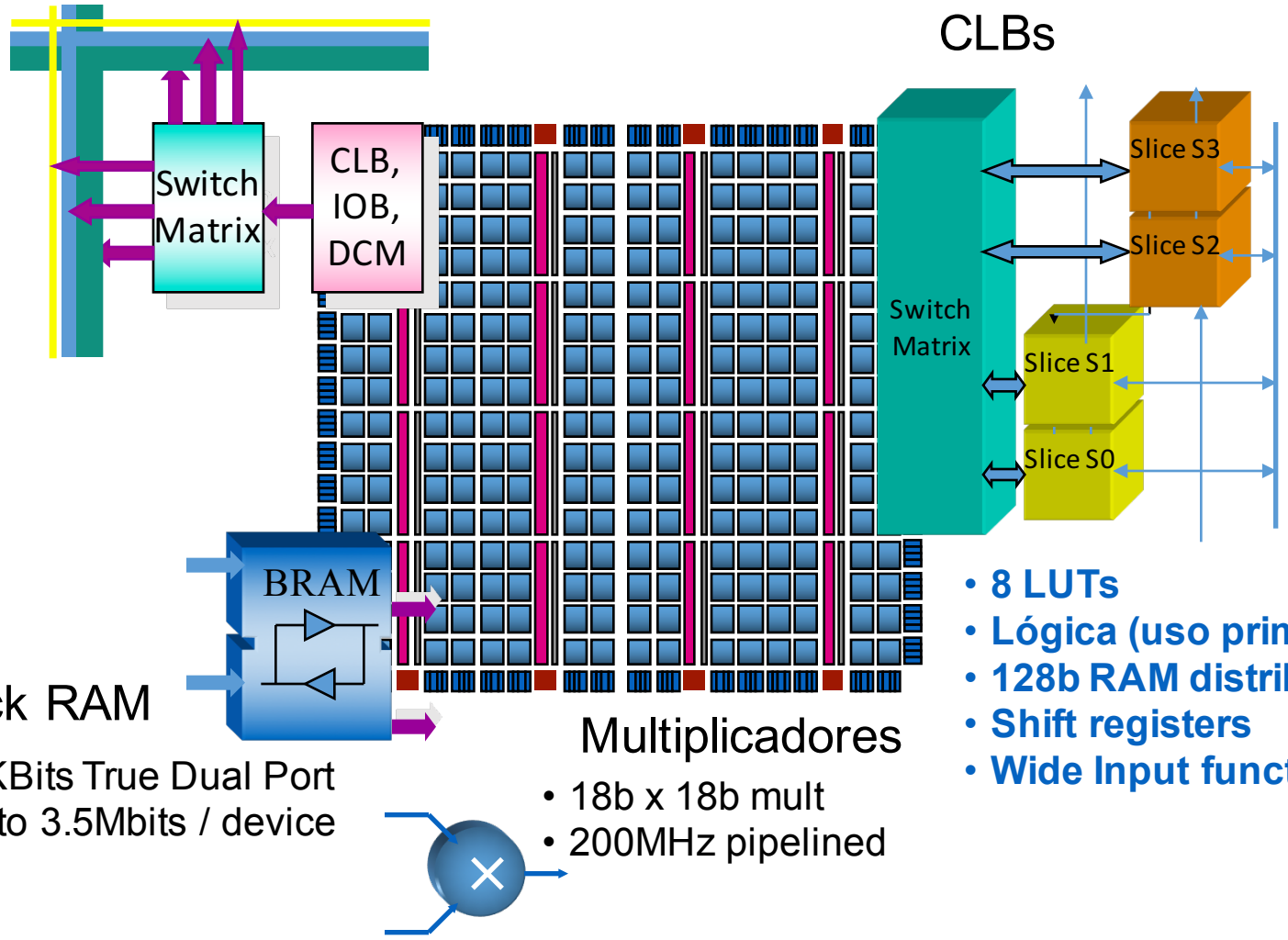
✓ Dados de 2009, onde 2 fornecedores dominam, indicando um mercado maduro.

DISPOSITIVOS XILINX E ALTERA

- **Xilinx**
  - ✓ Baixo Custo
    - Famílias Spartan 2, Spartan 3, Spartan 6
  - ✓ Alto desempenho
    - Virtex 5, Virtex 6,Virtex 7:Artix-Kintex-Virtex e Zynq (usam tec. 28nm)

- **Altera**
  - ✓ Baixo Custo e Mid-Range
    - Famílias Cyclone II, III, IV, V, Arria GX, II, V
  - ✓ Alto desempenho
    - Stratix II, III, IV, V

ALGUNS DETALHES: ARQUITETURA VIRTEX II

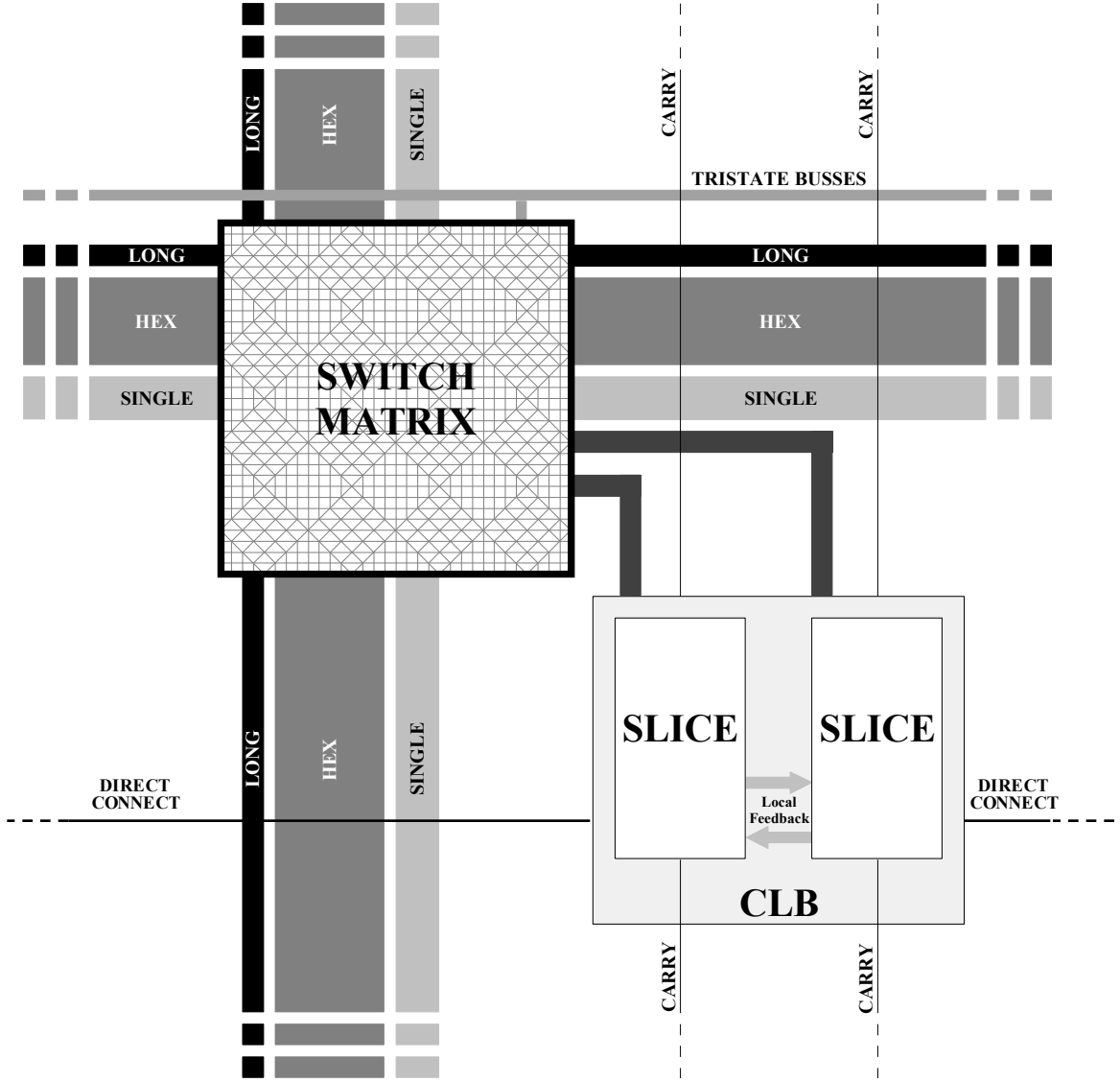


- 1 Slice contém um determinado número de LUTs, flip-flops e multiplexadores.

- **8 LUTs**
- **Lógica (uso primário)**
- **128b RAM distribuída**
- **Shift registers**
- **Wide Input functions (32:1)**

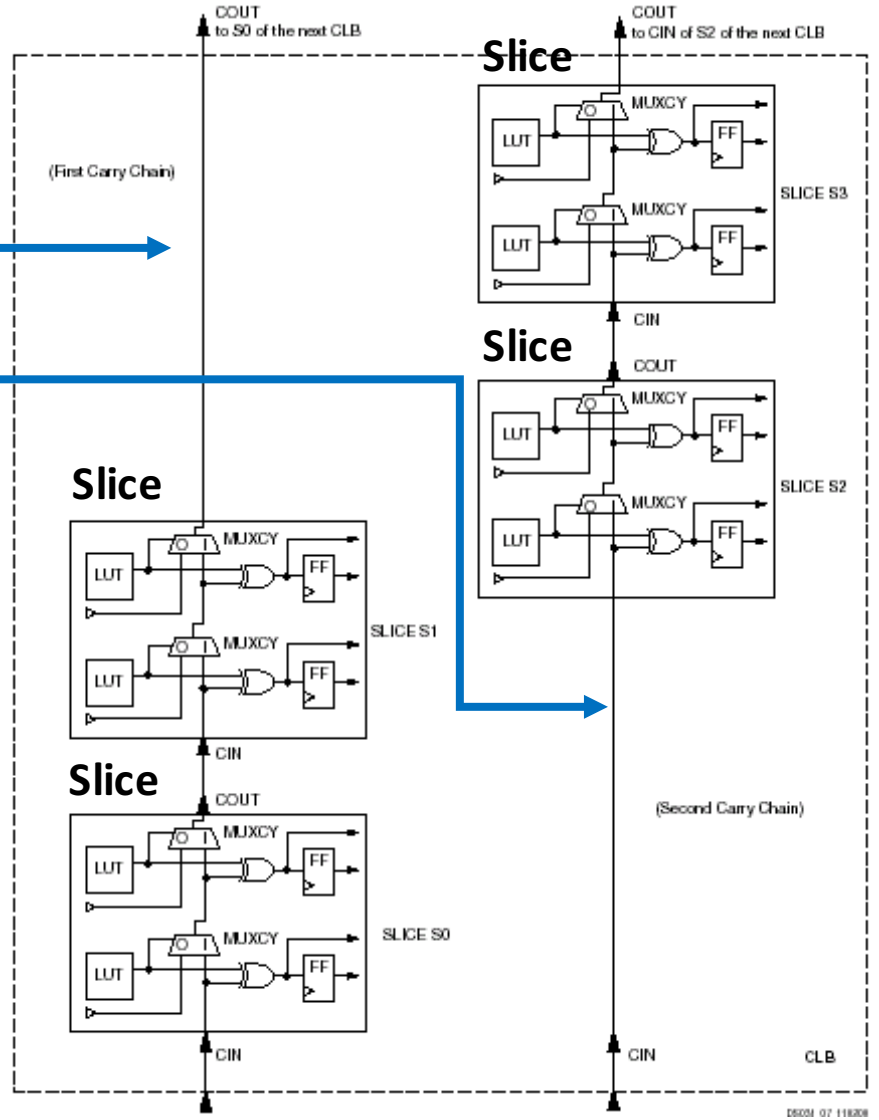
ARQUITETURA VIRTEX II – CLB E INTERCONEXÃO

- Conexões diretas entre CLBs vizinhas
  - ✓ Lógica de vai-um
- Matrix de conexão
  - ✓ CLB às linhas de roteamento
- Linhas de roteamento
  - ✓ Simples
  - ✓ Hexas
  - ✓ Longas
  - ✓ Tri-state



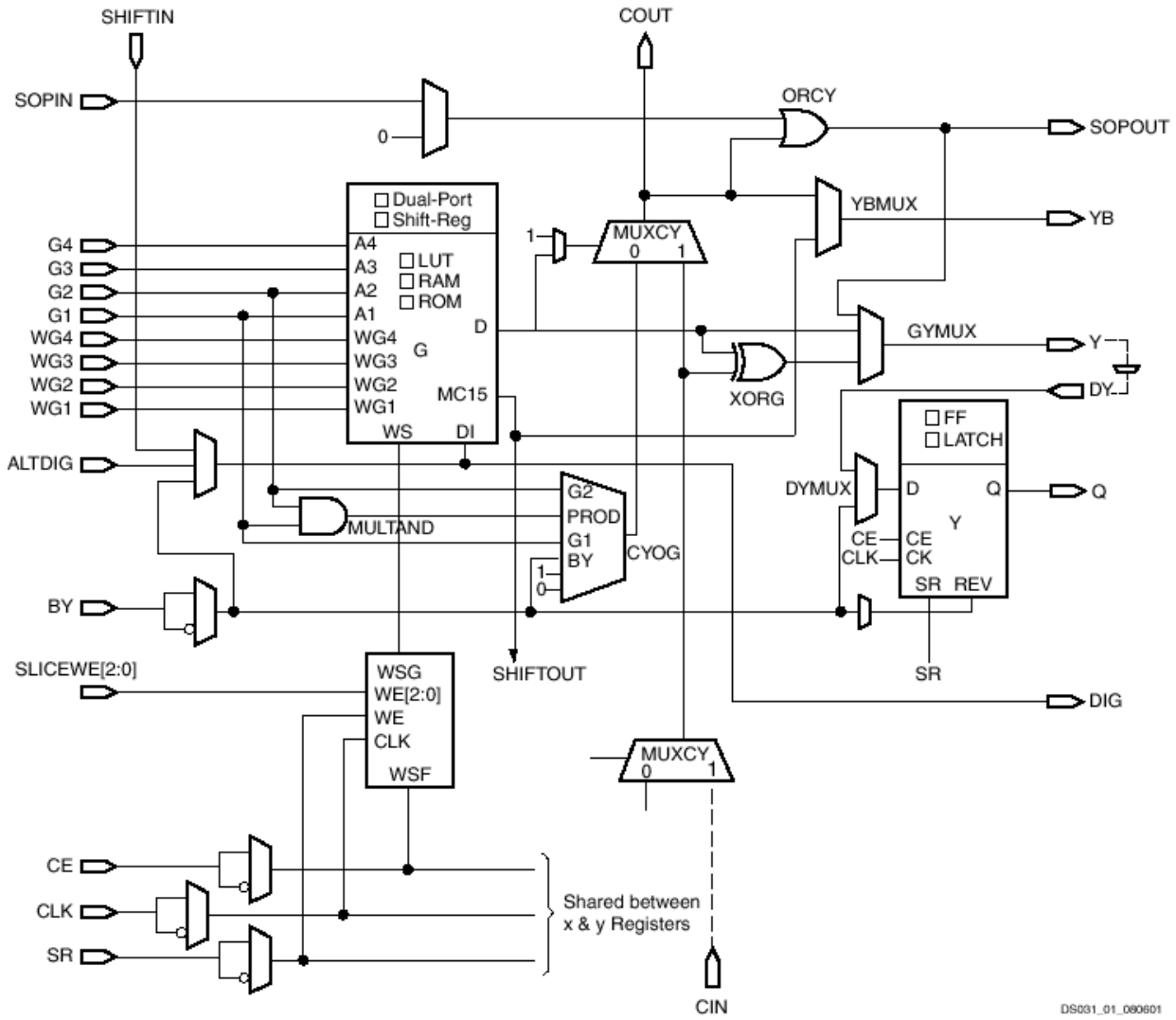
ARQUITETURA DO CLB DO DISPOSITIVO VIRTEX-II

- Fast Carry Logic Path
- Provides fast arithmetic add and sub



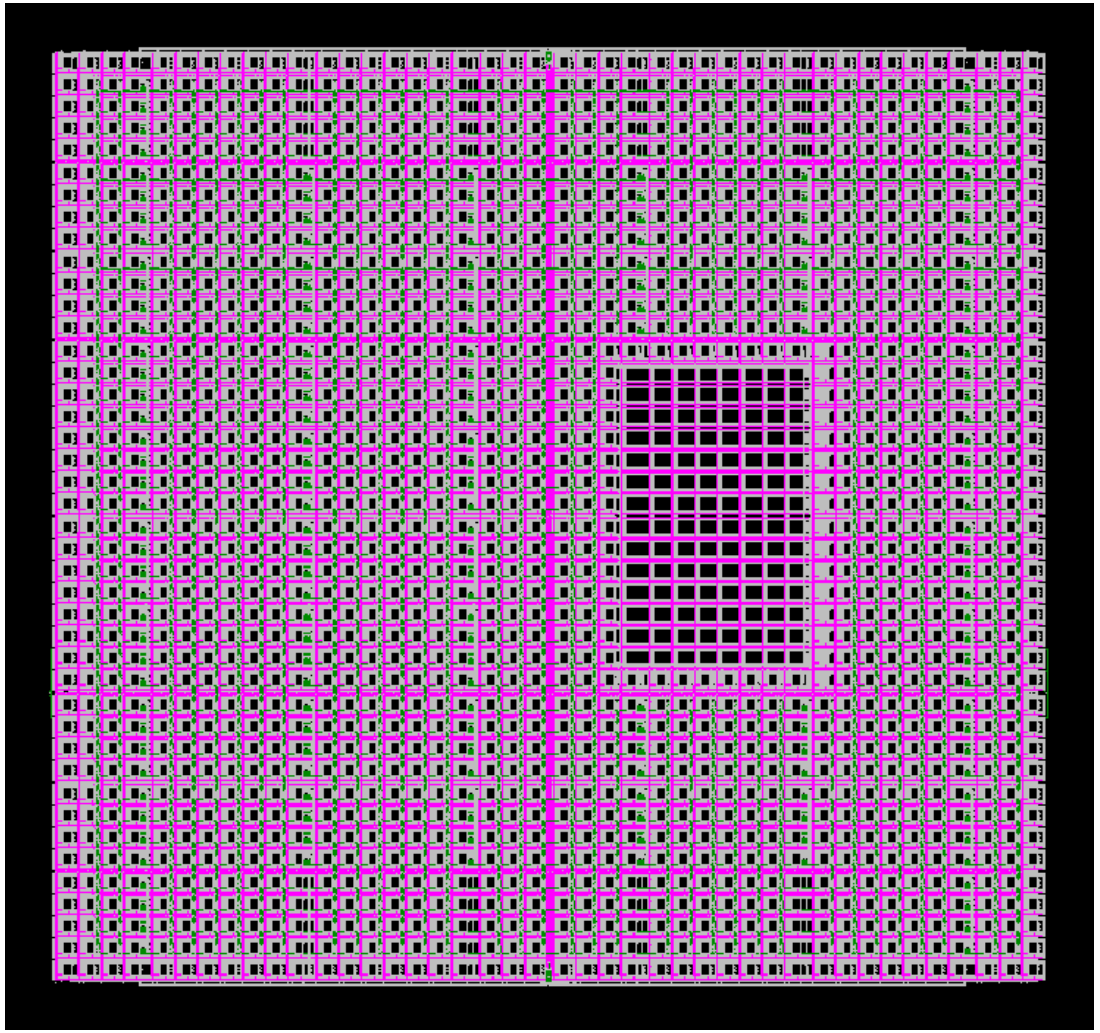
- RESUMINDO O CLB:
  - ✓ 4 Slices
  - ✓ 8 LUTS / 8 Flip-Flops
  - ✓ 2 cadeias de vai-um
  - ✓ 64 bits para memória
  - ✓ 64 bits para shift-register

ARQUITETURA – METADE DE UM SLICE



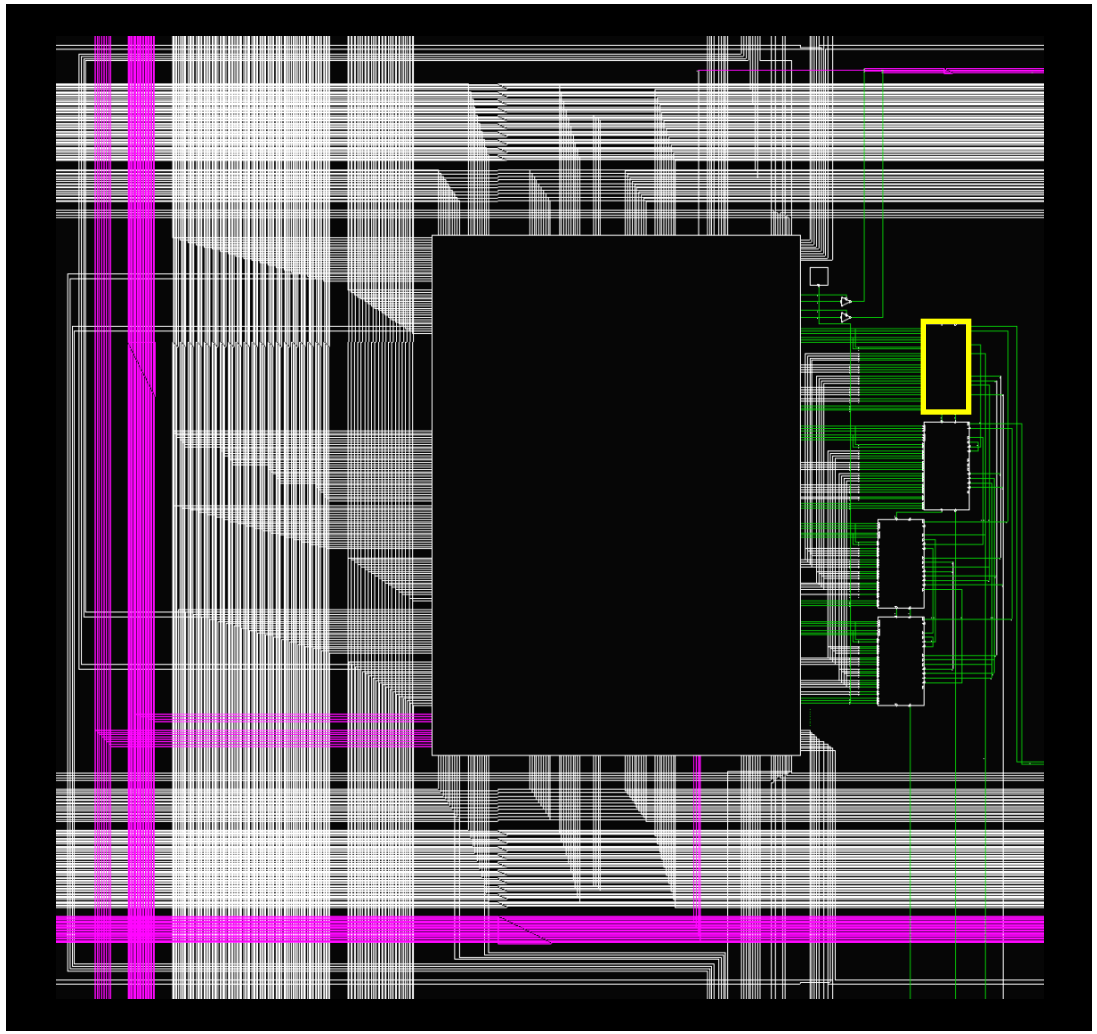
VIRTEX2P XC2VP7

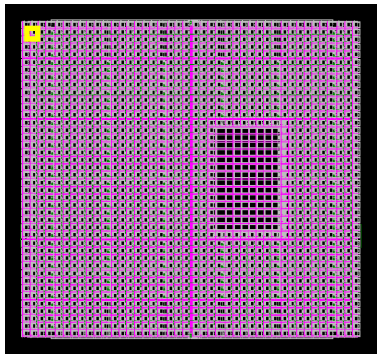
- FPGA Editor View With All Wires
  - ✓ 4,928 slices
  - ✓ 44 BRAMs
  - ✓ 1 PowerPC
  - ✓ 11,627 logic sites
  - ✓ 2,653 tiles
  - ✓ 1,423,681 wires
  - ✓ 544,549 segments



VIRTEX2P XC2VP7

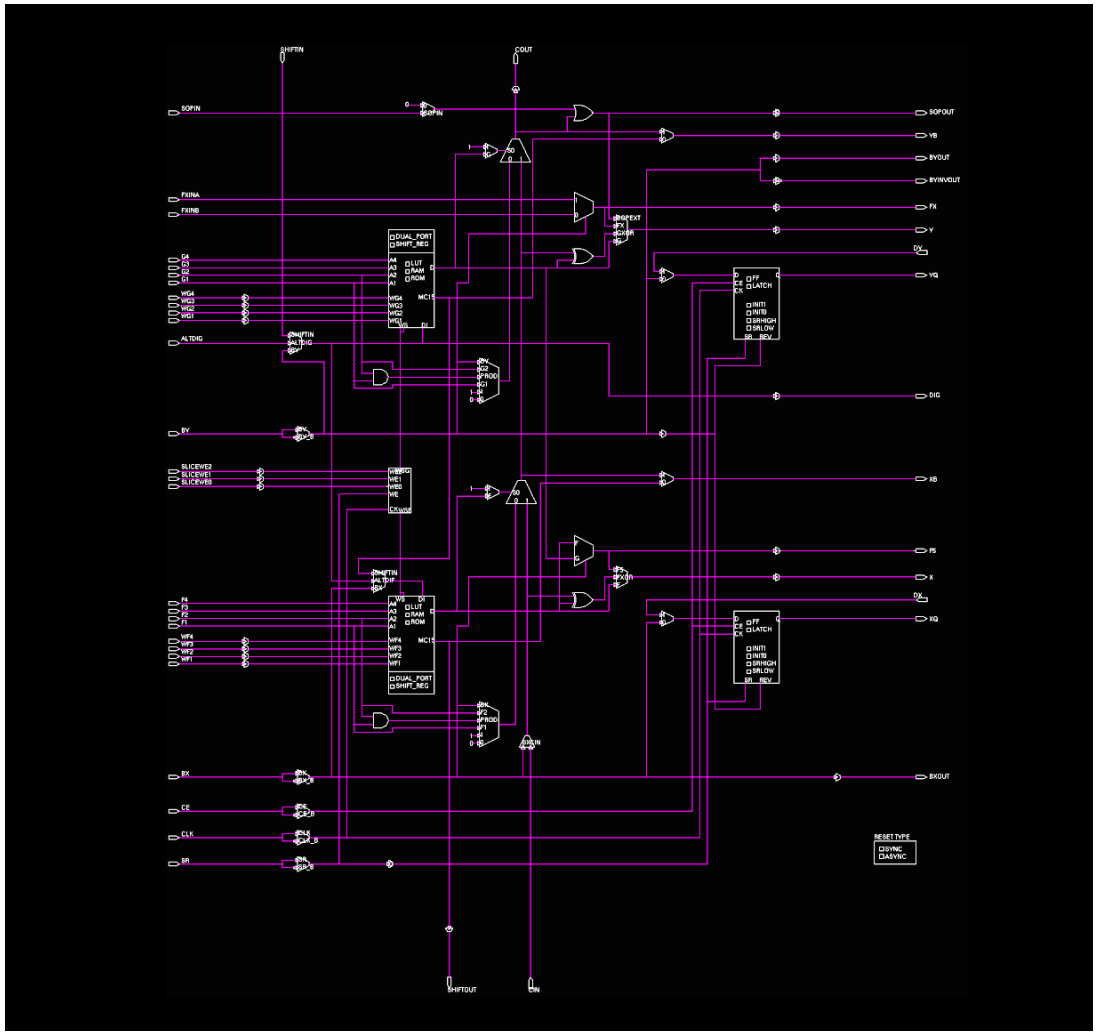
- FPGA Editor View With All Wires
  - ✓ Zoom de um CLB do canto superior esquerdo
  - ✓ Muitos recursos de roteamento
  - ✓ Grande caixa de conexões (switch box)
  - ✓ 4 slices e 2 TBUFs

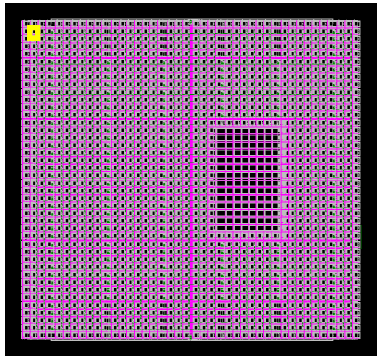


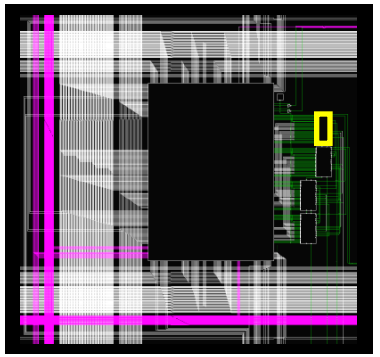


VIRTEX2P XC2VP7

- FPGA Editor View With All Wires
  - ✓ Slice da Família Virtex2Pro
  - ✓ 2 LUTs
  - ✓ 2 flip-flops
  - ✓ Vários muxs
  - ✓ Lógica de vai-um dedicada

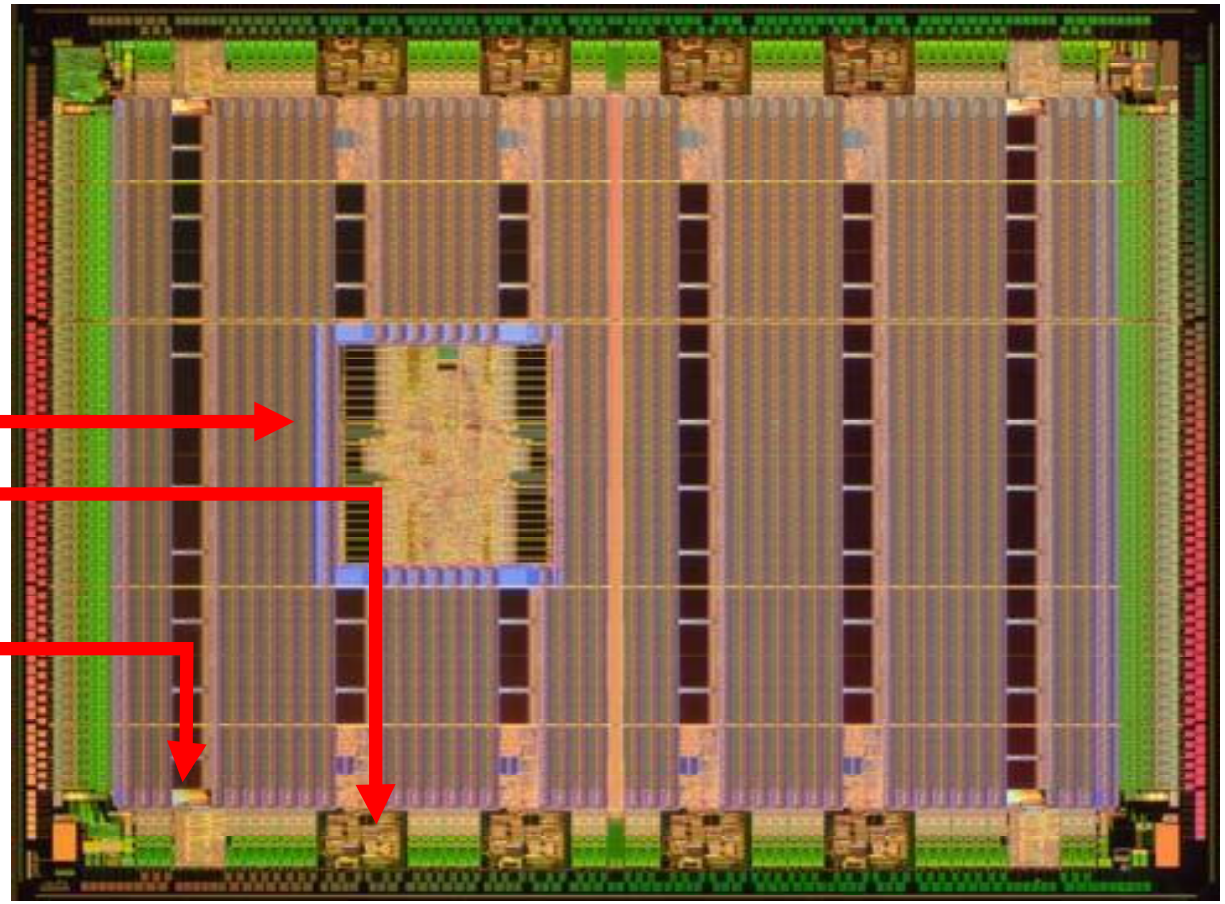






XC2VP7 VIRTEX-II PRO FPGA

- Foto do Layout do XC2VP7



- ✓ Power PC
- ✓ MGTs (gigabit transceiver)
- ✓ DCM (clock manager)

DEMAIS COMPONENTES DE FPGA MODERNO

- Gerenciamento de clock
  - ✓ Reduz escorregamento de relógio.
  - ✓ Permite multiplicar, dividir, mudar a fase da(s) frequências de entrada.
  - ✓ Implementações digitais (DCM – Xilinx, mais baratos) e analógica (PLL – Altera, mais flexíveis).
- Blocos de memória embarcada
  - ✓ Tipicamente blocos de 18kbits ou 36Kbits na Xilinx (Altera tem mais variedade de blocos de memória).
- Blocos DSP
  - ✓ Multiplicadores 18bitsx18bits para funções de imagem, áudio, telecomunicações .

DEMAIS COMPONENTES DE FPGA MODERNO

- Processadores embarcados do tipo *hard macro*
  - ✓ Xilinx disponibiliza o processador PowerPC (clock de 300-500MHz).
  - ✓ Podem executar sistemas operacionais embarcados como Linux.
- Transceptores Gigabit
  - ✓ Blocos serializadores / deserializadores para receber dados em altas taxas de transmissão.
  - ✓ Virtex-4 é capaz de receber e transmitir dados em frequências de 3.2 Gbps usando dois fios.
- Outros
  - ✓ Ethernet MAC.
  - ✓ Criptografia do bitstream.
  - ✓ Controle para reconfiguração interna (de dentro do FPGA - ICAP).

PROTOTIPAÇÃO EM FPGA

UTILIZANDO O FPGA PARA PROTOTIPAÇÃO

1. Abaixo aparece um circuito somador, um daqueles visto no trabalho T1, que deve ser prototipado nesta aula (compare com a implementação daquele trabalho):

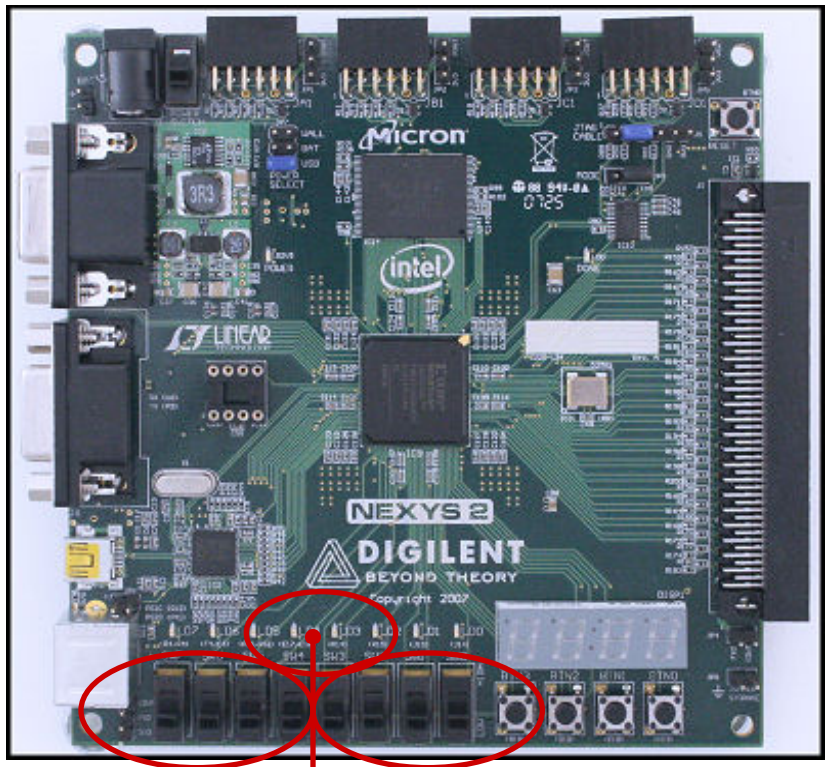
```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all; --Para permitir soma de std_logic

entity somador is
port (A, B: in std_logic_vector(3 downto 0);
      Soma: out std_logic_vector(3 downto 0));
end somador;

architecture somador of somador is
begin
  Soma <= A + B; --Soma de dois vetores de 4 bits
end somador;
```

ONDE AS ENTRADAS E SAÍDAS SE CONECTAM?

- Placas de prototipação têm recursos de entrada e saída:
  - ✓ LEDs, chaves, displays, teclado, serial, USB, Ethernet...



A(3 downto 0); B(3 downto 0);

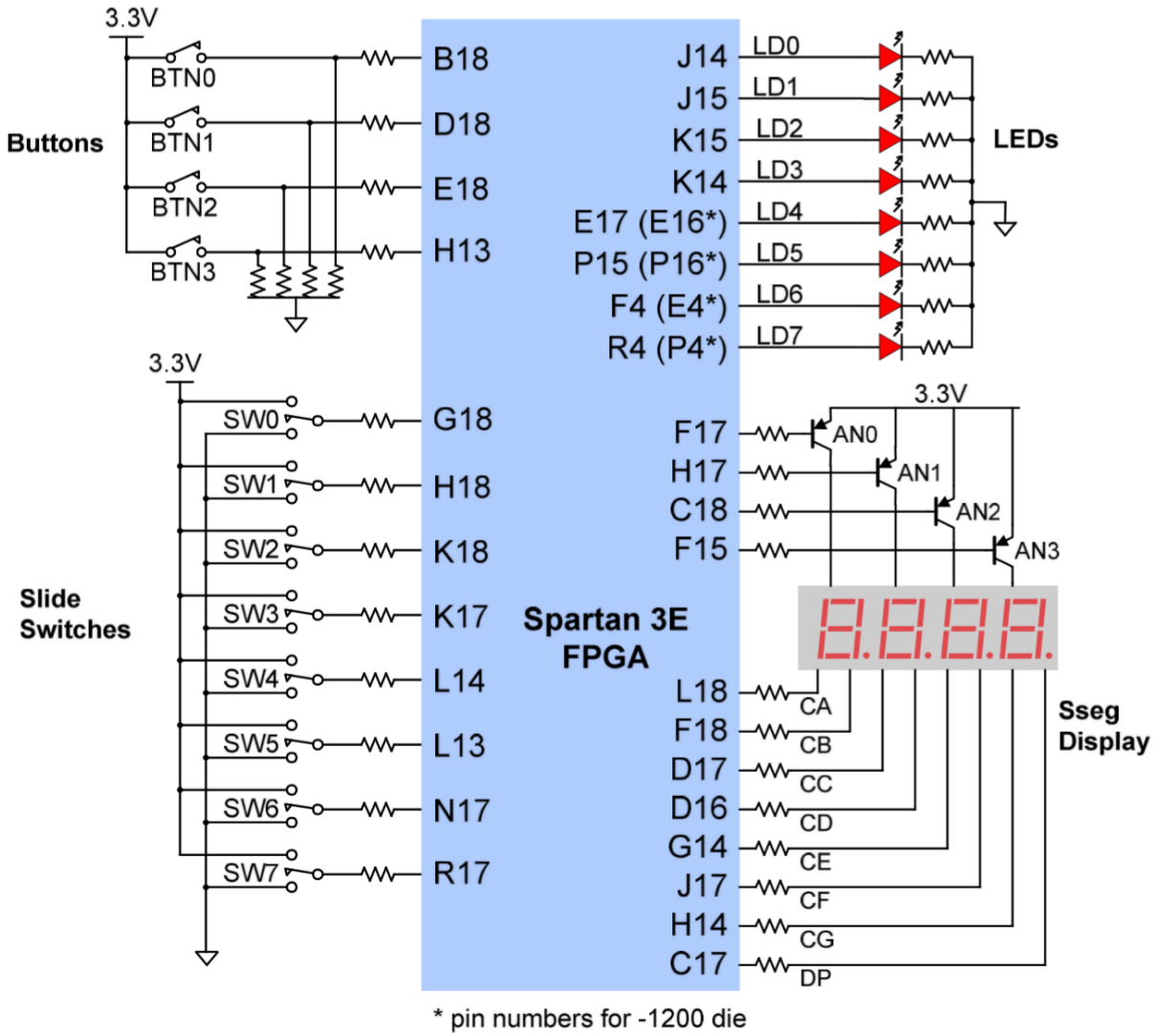
Soma(3 downto 0)

- ✓ Ao lado aparece um exemplo de atribuição de dispositivos de entrada e saída da placa Nexys2 para prototipar o somador deste trabalho.

ONDE AS ENTRADAS E SAÍDAS SE CONECTAM?

2. Abrir o manual da placa que esta disponível no material de apoio do Moodle, ir na página 5 deste manual e achar a Figura ao lado.

   - ✓ Um arquivo de projeto relaciona as entradas e saídas do VHDL com os recursos da placa.
   - ✓ Este arquivo se chama UCF (abreviatura de User Constraint File)



ONDE AS ENTRADAS E SAÍDAS SE CONECTAM?

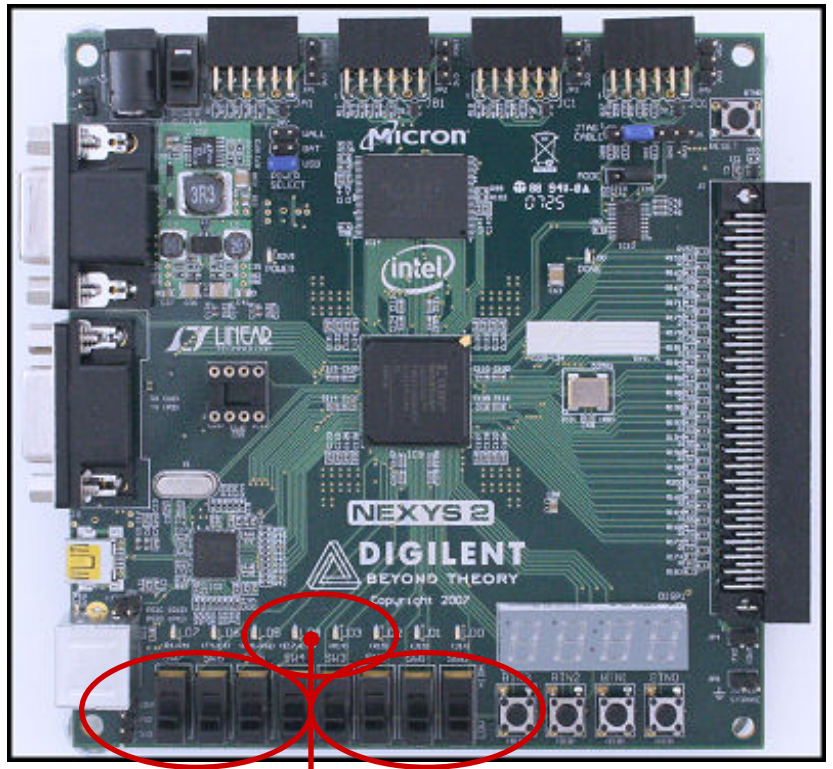
3. Criação do arquivo UCF – Arquivo que define a relação entre cada nome de fio/pino em VHDL e pinos físicos associados do FPGA:

```
### UCF DO PROJETO SOMADOR DE 4 BITS

NET "A<0>"  LOC = "L14"; # Bit 0 do vetor A
NET "A<1>"  LOC = "L13"; # Bit 1 do vetor A
NET "A<2>"  LOC = "N17"; # Bit 2 do vetor A
NET "A<3>"  LOC = "R17"; # Bit 3 do vetor A

NET "B<0>"  LOC = "G18"; # Bit 0 do vetor B
NET "B<1>"  LOC = "H18"; # Bit 1 do vetor B
NET "B<2>"  LOC = "K18"; # Bit 2 do vetor B
NET "B<3>"  LOC = "K17"; # Bit 3 do vetor B

NET "Soma<0>"  LOC = "K15";
NET "Soma<1>"  LOC = "K14";
NET "Soma<2>"  LOC = "E16";
NET "Soma<3>"  LOC = "P16";
```

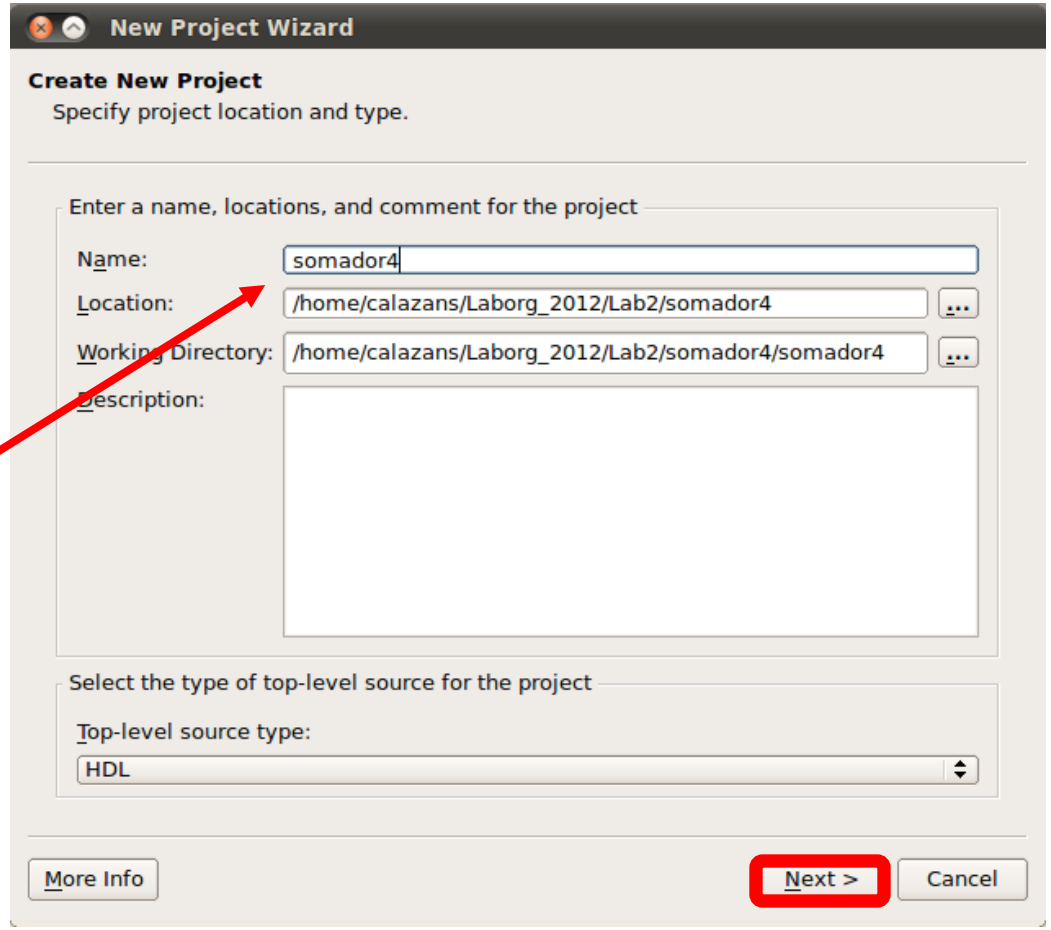


AMBIENTE DE SÍNTESE: ISE

4. Criar um diretório, colocando neste os arquivos VHDL (somador4.vhd) e o arquivo UCF (somador4.ucf).

5. Abrir a ferramenta ISE como descrito nas transparências do trabalho anterior e criar um novo projeto (File → New Project), como ao lado:

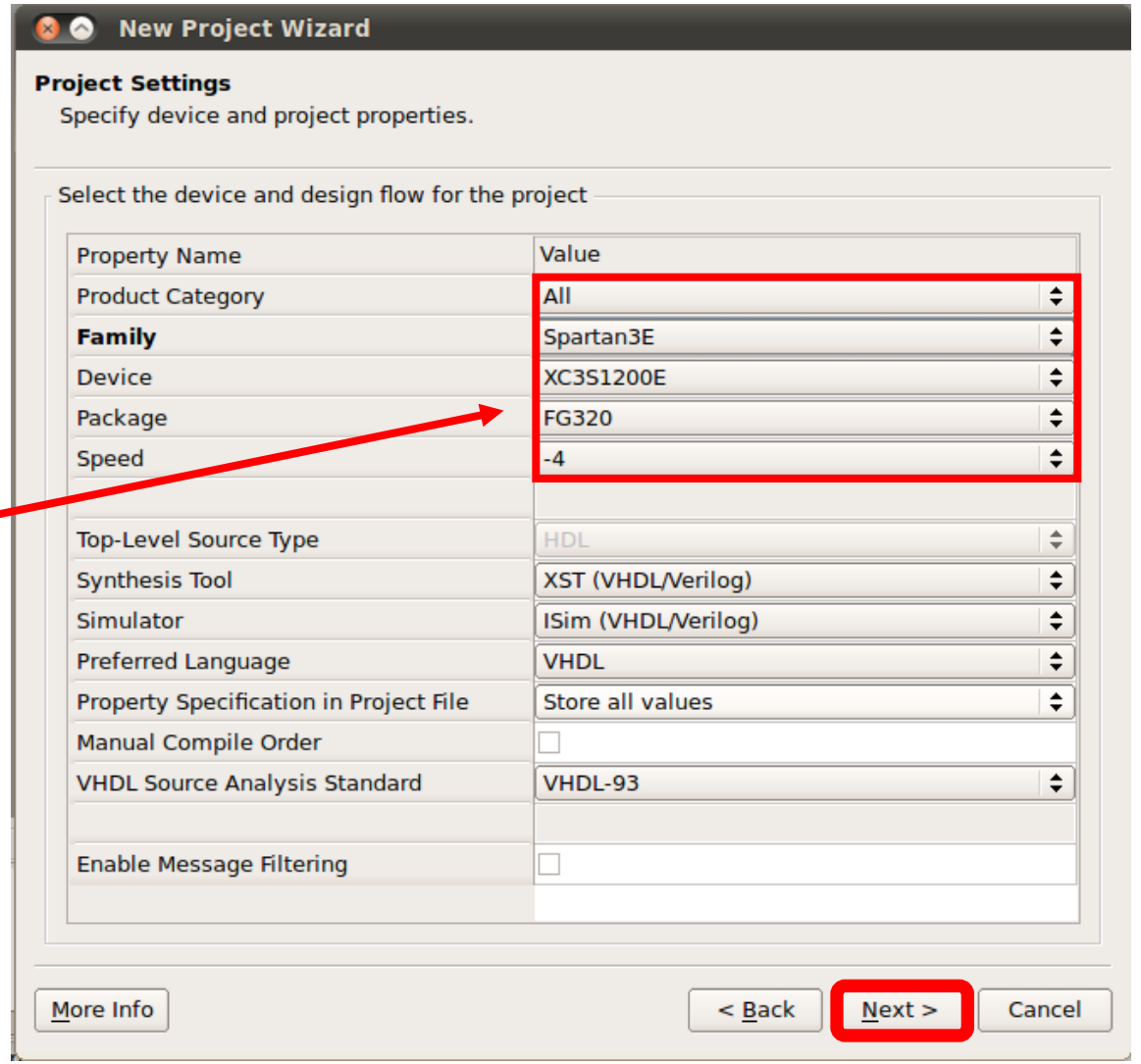
<u>Cuidado:</u> Não podem haver espaços em branco ou caracteres especiais no nome do caminho para o projeto, nem no nome do projeto, só ASCII puro.



DEFINIÇÃO DO FPGA DA PLACA DE PROTOTIPAÇÃO

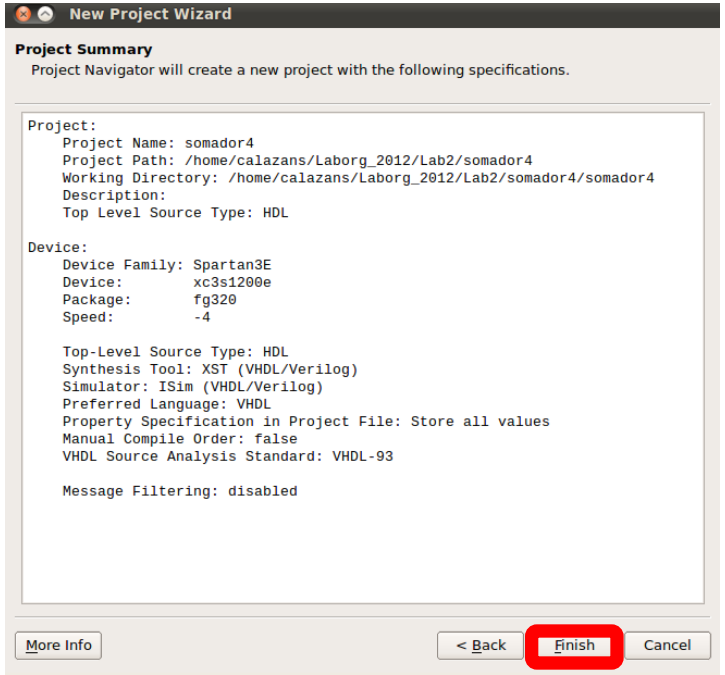
6. Para a placa que estamos trabalhando, o FPGA é um dispositivo da família Spartan3, escolher na janela, como mostrado ao lado:

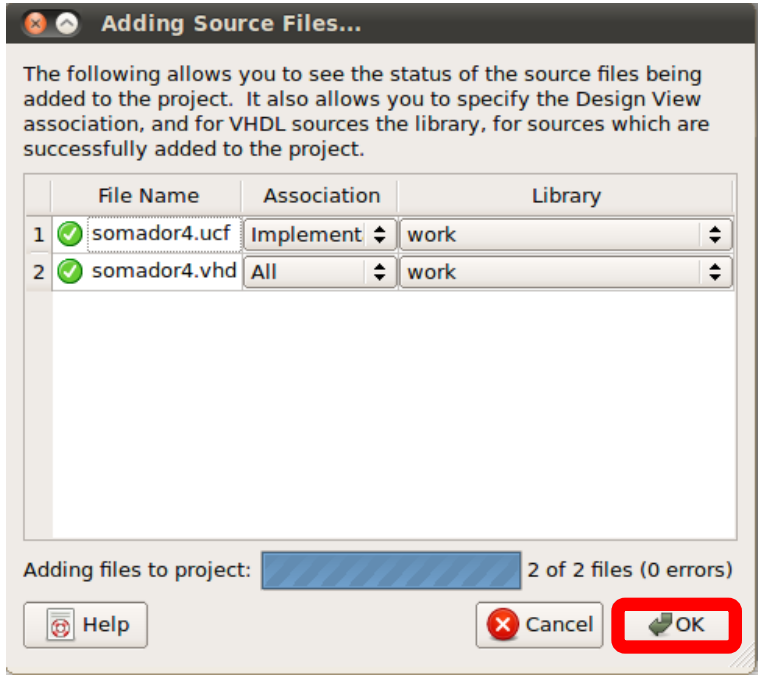
Características do dispositivo FPGA



DEFINIÇÃO DO FPGA DA PLACA DE PROTOTIPAÇÃO

7. A próxima janela é só um resumo do projeto, clicar Finish.
8. A seguir, na janela principal aparece o projeto vazio. Deve-se então acrescentar os arquivos fonte (somador4.vhd e somador4.ucf) gerados anteriormente. Na janela Hierarchy, clicar com o botão direito no ícone do dispositivo (xc3s1200e-4fg320). No menu que surge, escolher Add Copy of Source. Procurar no disco e adicionar os dois arquivos ao projeto.



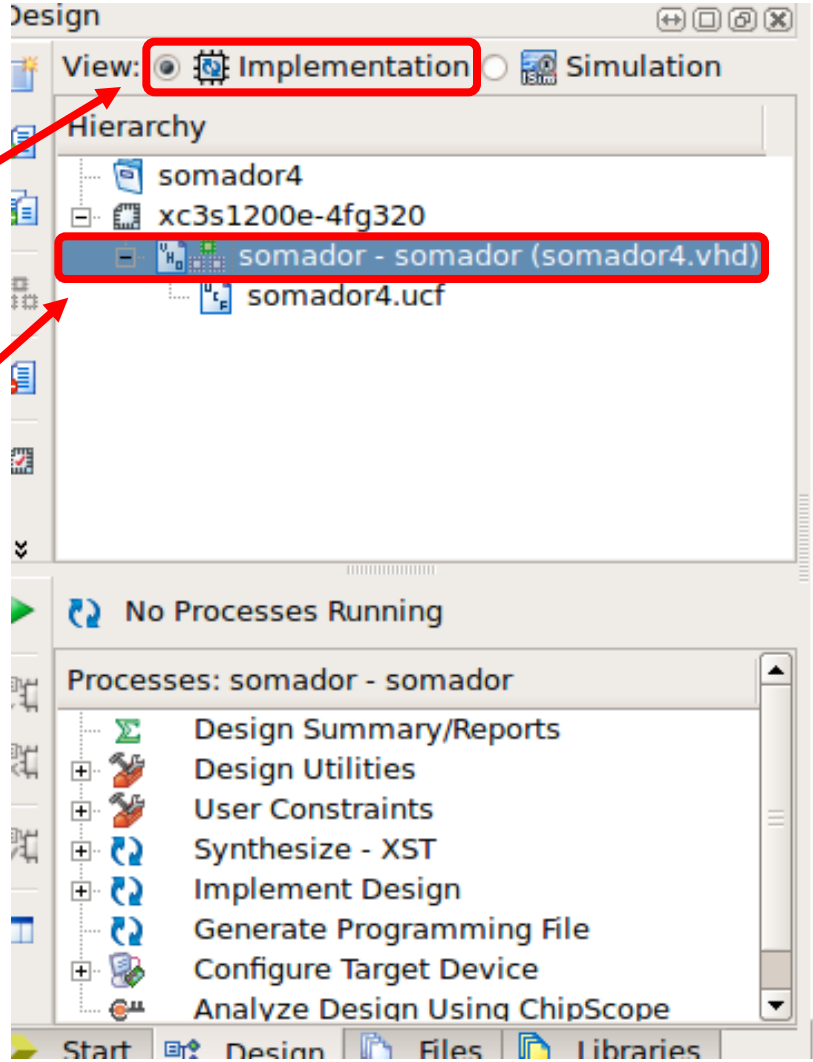


AMBIENTE ISE – BROWSER DO PROJETO

- Se todos os passos de criação foram corretamente seguidos, deve-se ter:

Lembrem-se: aqui trabalha-se com síntese (implementação) e não com simulação.

Arquivo no topo da hierarquia do projeto.



PASSO 1 DA SÍNTESE: SÍNTESE LÓGICA

9. Devemos transformar o VHDL em portas lógicas. Para executar a síntese lógica, você deve dar um duplo click em "Synthesize XST".

   ✓ Ao final tem-se o relatório de ocupação, ver exemplo de resultado abaixo.

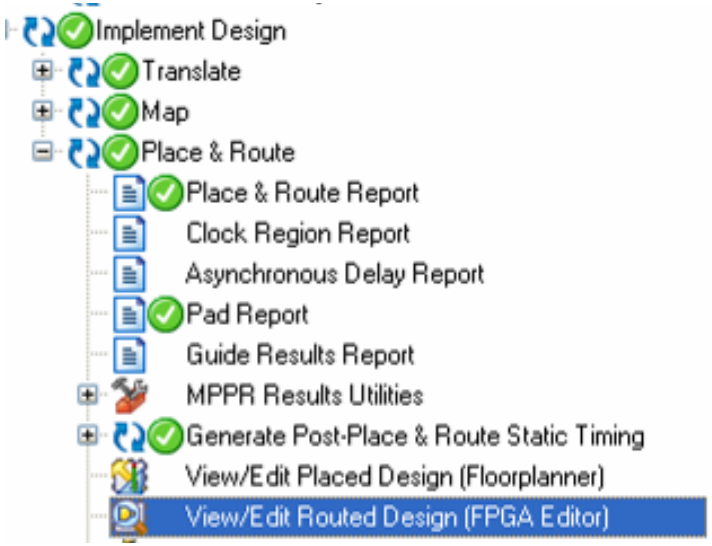
Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	3	8672	0%
Number of 4 input LUTs	6	17344	0%
Number of bonded IOBs	12	250	4%

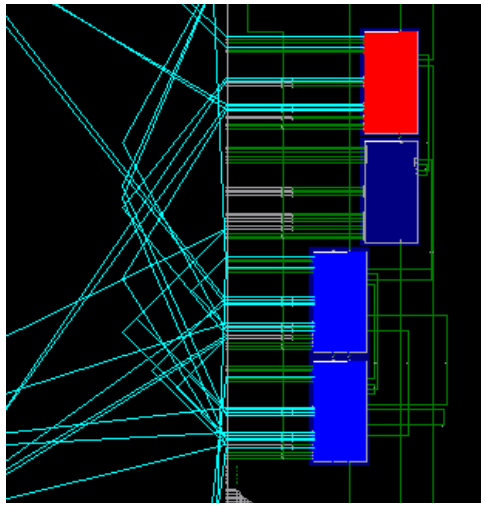
6 LUTs, de 17.344 disponíveis

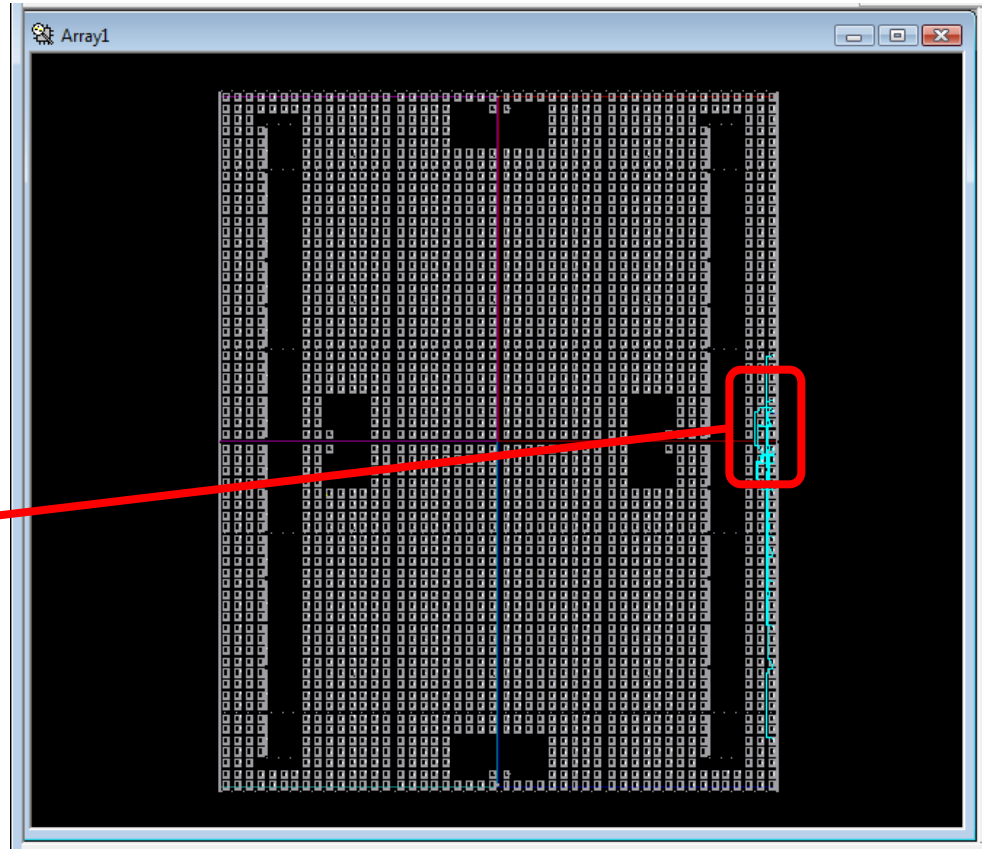
12 pinos, de 250 disponíveis

PASSO 2 DA SÍNTESE: SÍNTESE FíSICA

9. Devemos fazer o posicionamento e o traçado de conexões dentro do FPGA. Para isso, devemos dar um duplo click em “Implement Design”.
10. Em seguida, dar duplo click em “Generate Programming File”.
11. Selecionar FPGA Editor, executar o programa e visualizar o “layout” gerado automaticamente pelo processo de síntese.







CONFIGURAR O FPGA

12. A síntese termina por gerar um arquivo com extensão **.bit** que pode ser usado para configurar o hardware no FPGA. Para tanto, usa-se o comando **djtgcfg** (Digilent JTAG Configuration Utility), instalado nas máquinas do Laboratório e no LAPRO. Executem `djtgcfg ?` para ver as opções do comando.
13. Conectem a placa ao computador via o cabo USB e digitem `djtgcfg enum`. Este comando deve comunicar-se com a placa para identificá-la e obter seus dados, que são impressos.
14. Agora digitem `djtgcfg -d Nexys2 init`. Este comando deve comunicar-se com a placa Nexys2 e listar os dispositivos Xilinx da mesma (o FPGA e a memória XCF04S, que pode guardar uma configuração completa do FPGA).
15. Finalmente, para configurar o FPGA usem o seguinte comando: síntese: `djtgcfg prog -d Nexys2 -i 0 -f somador4.bit`, certificando-se de estar no diretório onde se encontra o arquivo gerado pela síntese.
16. O LED amarelo de configuração carregada deve acender e talvez alguns LEDs de dados. Experimente com o projeto, certificando-se que ela faz somas de forma correta.

TRABALHO

PROJETO 1

- Projeto 1 consiste em acrescentar o cálculo de “vai-um” no circuito do somador apresentado em aula. Segue uma sugestão de modificações no código para resolver o problema:
  - ✓ Modificar a saída “Soma” para ser um vetor de 5 bits.
  - ✓ Declarar dois sinais internos ‘AA’ e ‘BB’, ambos de 5 bits.
  - ✓ Fazer a “Soma <= AA + BB”, e criar ‘AA’ e ‘BB’ através de uma concatenação com ‘0’:

```
AA <= '0'  &  A; -- o símbolo & significa concatenação em VHDL
```

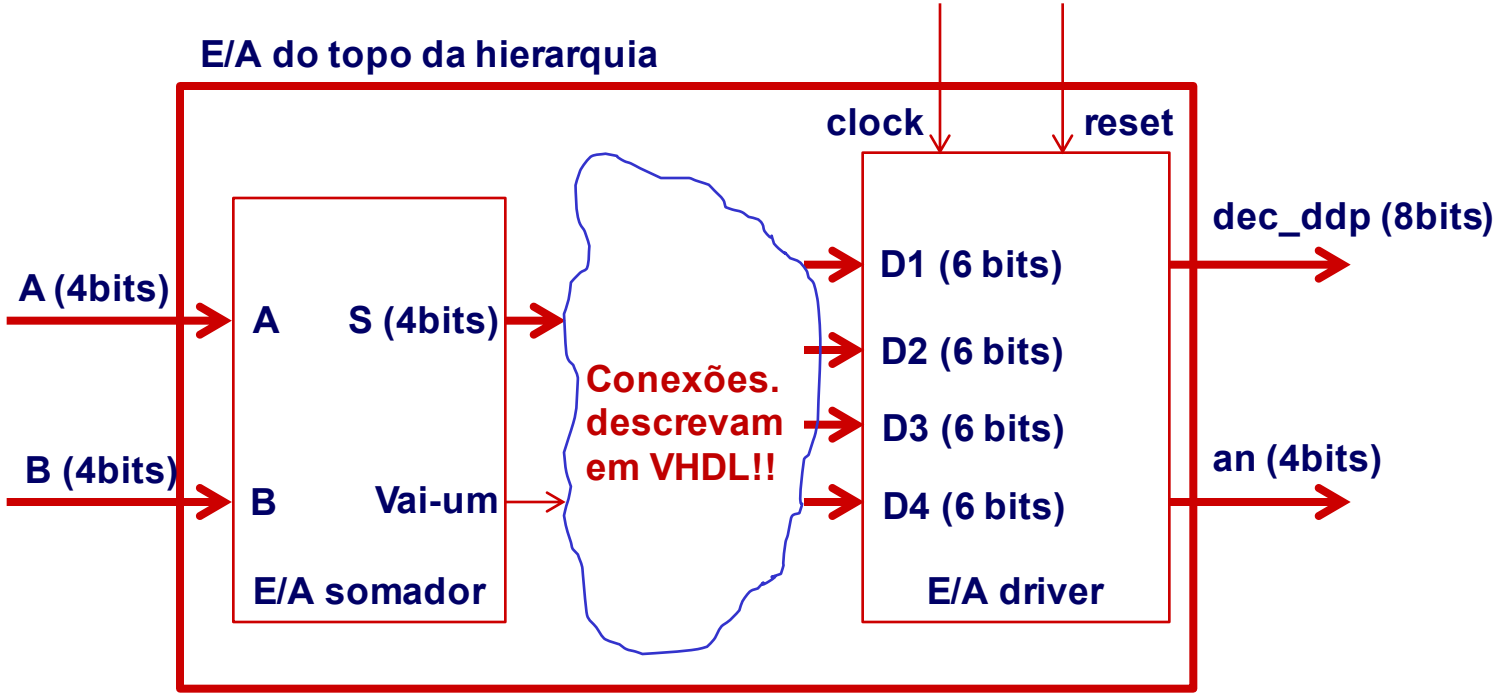
- Modificar o UCF para que um dos LEDs não usados (por exemplo o LD7) seja associado ao “vai-um” gerado (o quinto bit da soma).
- Simular o VHDL com o mesmo testbench do Trabalho 1 para este novo VHDL.
- Aplicar os passos descritos na seção “Prototipação em FPGA” ao novo VHDL gerado, prototipando o novo hardware.

PROJETO 2

- No Projeto 2 o resultado deve aparecer no display de sete segmentos. Segue uma sugestão de modificação do projeto para resolver o problema. Lembre-se de usar como base o Projeto 1 e suponha que o resultado da soma vá ao display mais à direita da placa, e que o "vai-um" seja associado a algum LED (pode ser um LED não usado, como LD4 a LD7 ou um ponto decimal de um dos mostradores de 7 segmentos):
  - ✓ Ler o manual de referência para entender como funcionam os mostradores (fim da página 5 e página 6 do arquivo Documentacao_da_placa_de_prototipacao.pdf).
  - ✓ Abrir e estudar o código VHDL contido no material de apoio (dspl_drv_nexys.vhd). Deve-se acrescentar este arquivo no seu projeto:
    - No ambiente ISE, ir na janela Sources (canto superior esquerdo) clicar com botão direito do mouse no ícone com o nome do dispositivo (xc3s1200E-4FG320) e escolhendo a opção de menu Add Source.
  - ✓ Modificar o UCF para adaptá-lo ao novo formato da saída e prototipar.

PROJETO 2

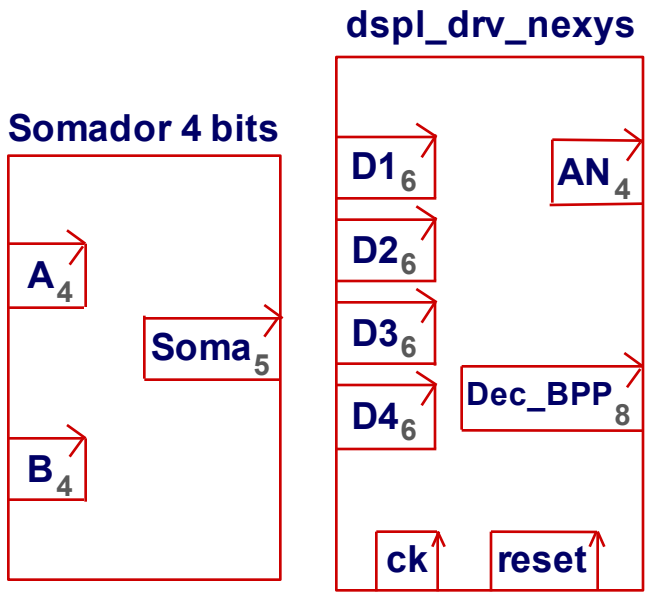
- Para ajudar na compreensão desta parte do trabalho, segue abaixo um diagrama de blocos e conexões parcial do circuito resultante. A expressão E/A corresponde a um par Entidade/Arquitetura que define um módulo de hardware descrito em VHDL.

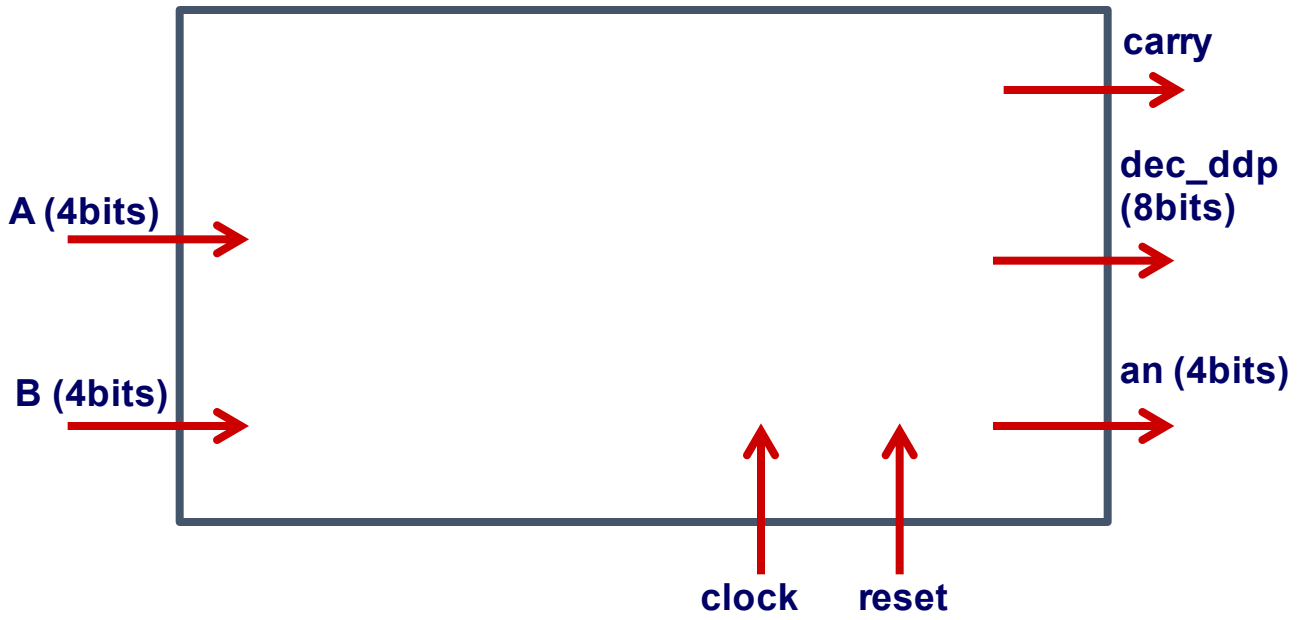


- ✓ Notem que se trata de uma descrição hierárquica: O somador e o driver são instanciados dentro da descrição VHDL da entidade, no topo da hierarquia de projeto.
- ✓ Nem todos os pinos e fios estão nomeados explicitamente no diagrama. Completem-nos nos arquivo(s) VHDL.

PROJETO 2

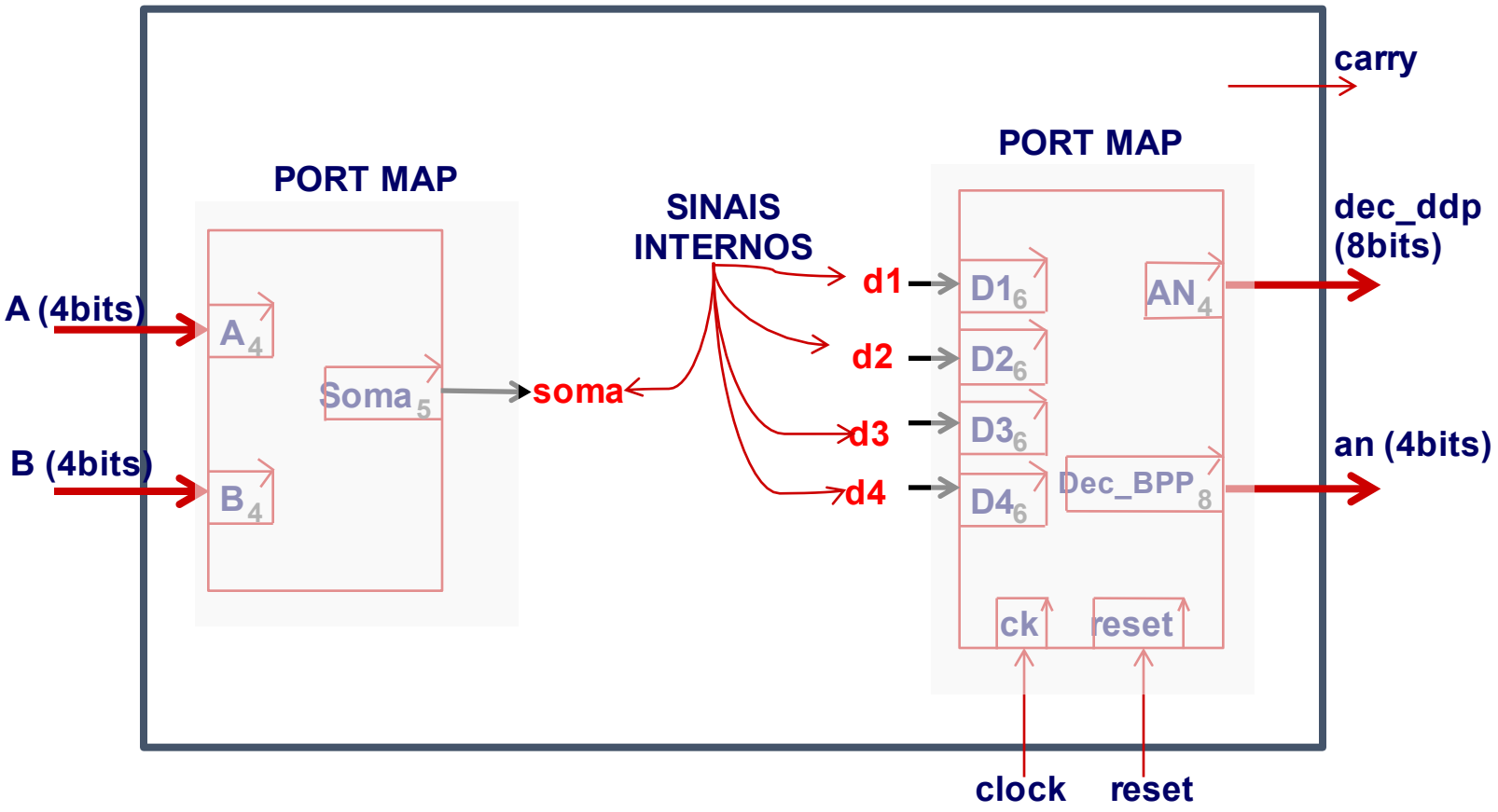
1. Colocar em um diretório src os fontes do somador e o driver de display (disponível no ambiente Moodle).
2. Adicionar no diretório src o arquivo UCF disponível no Moodle.





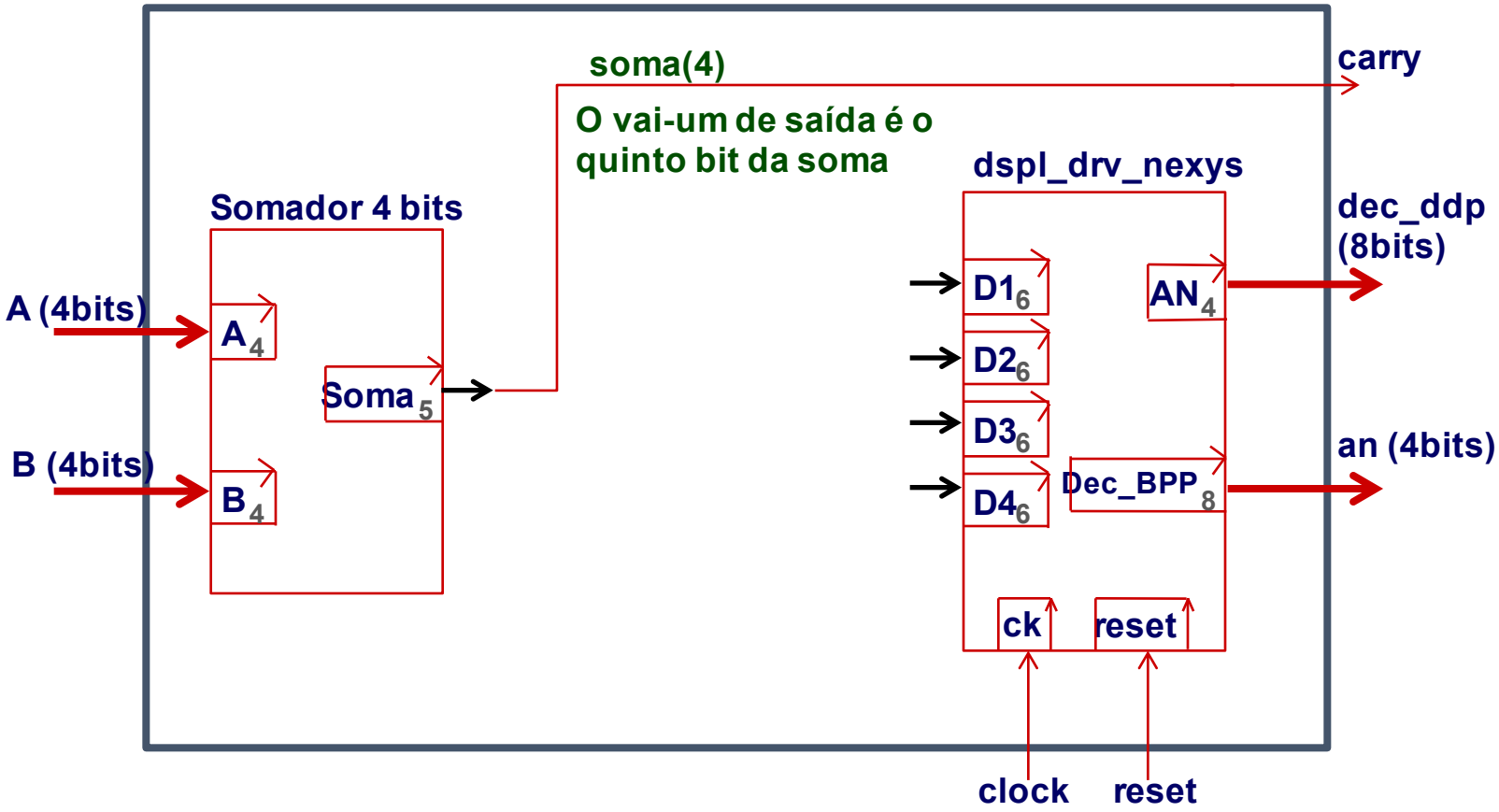
PROJETO 2

3. Devemos criar um VHDL top. Ao fazermos os 2 comandos port_map, alguns sinais internos precisam ser criados.



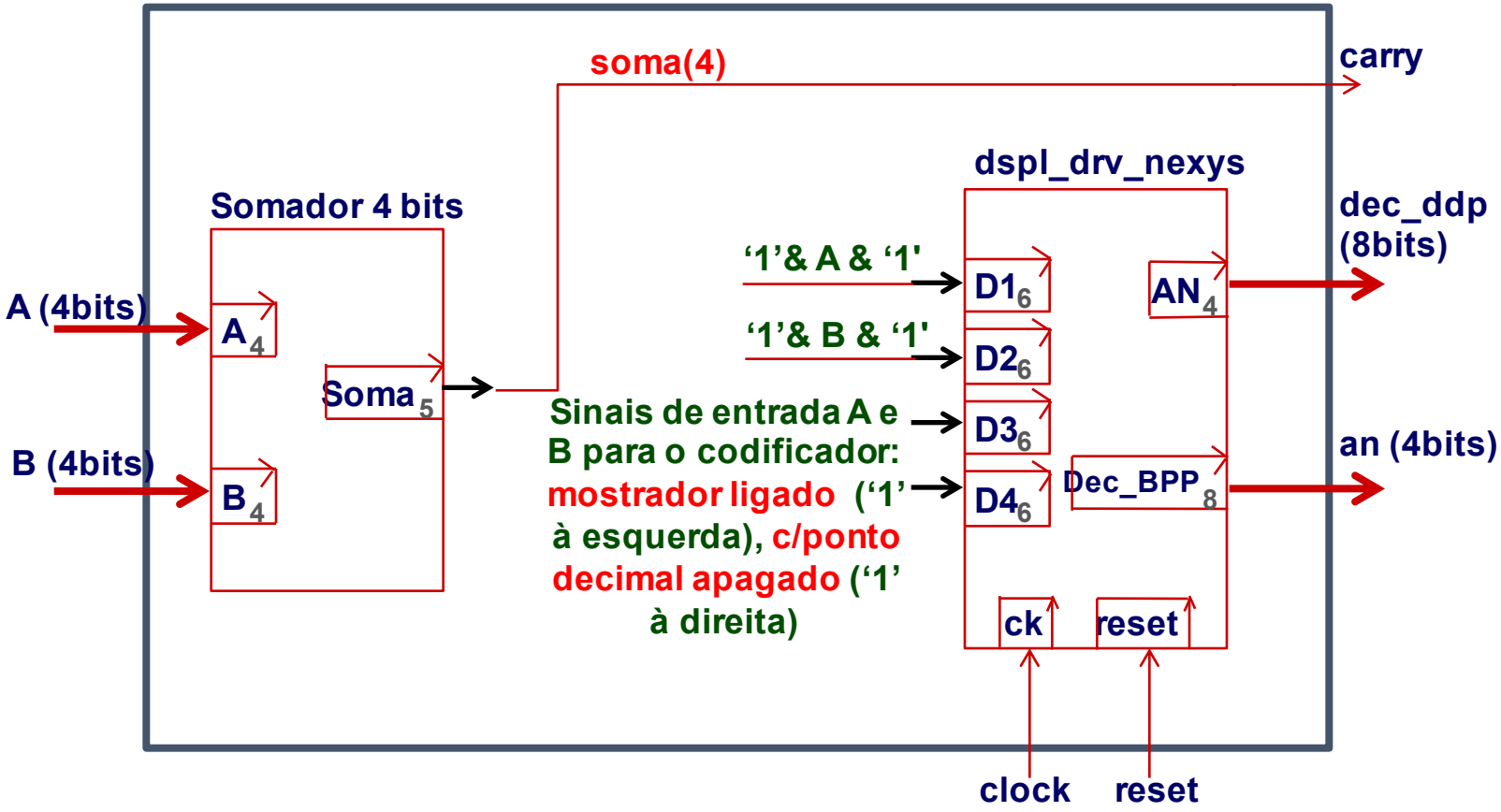
PROJETO 2

4. Ligar os sinais internos.



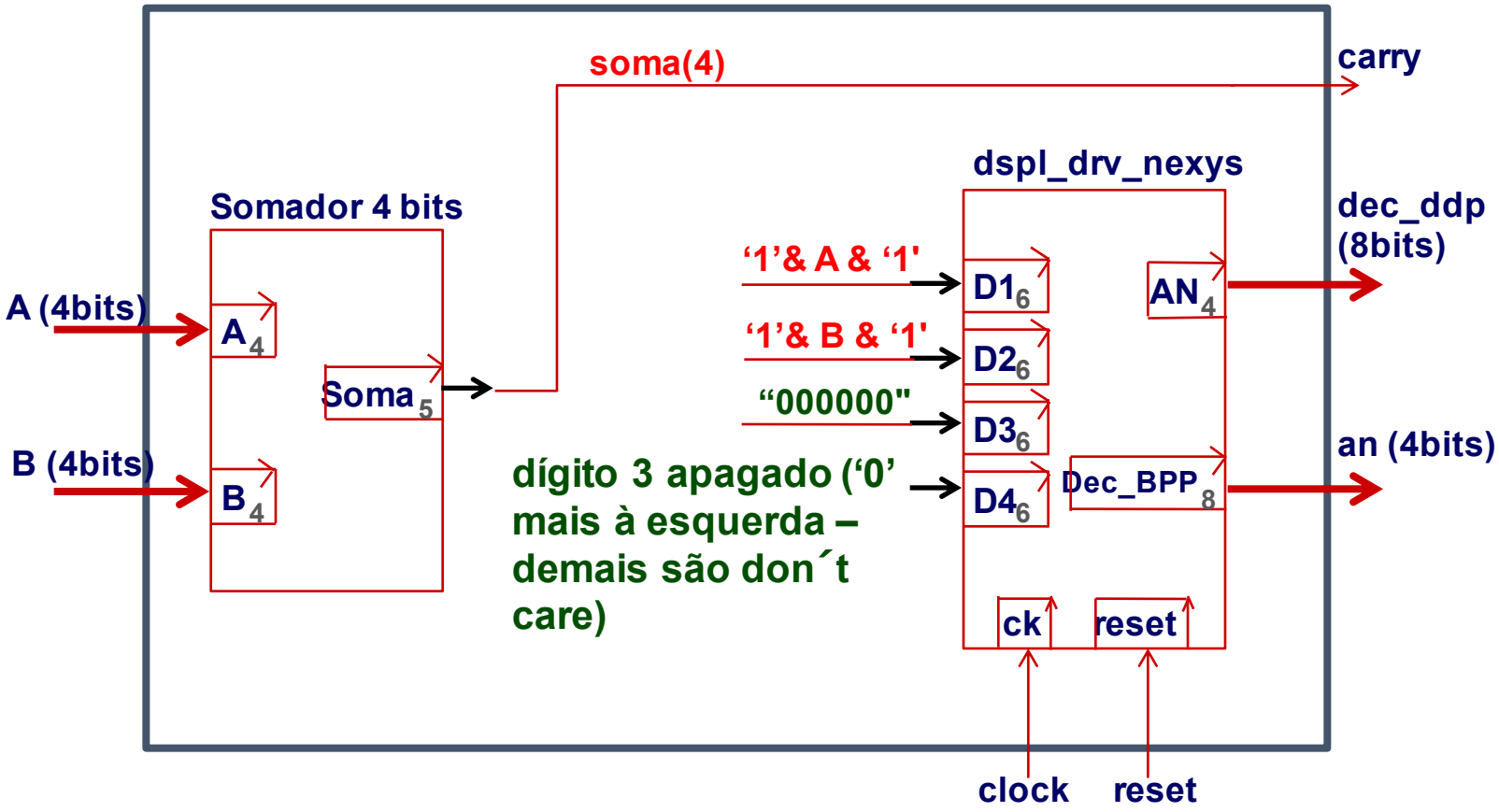
PROJETO 2

4. Ligar os sinais internos.



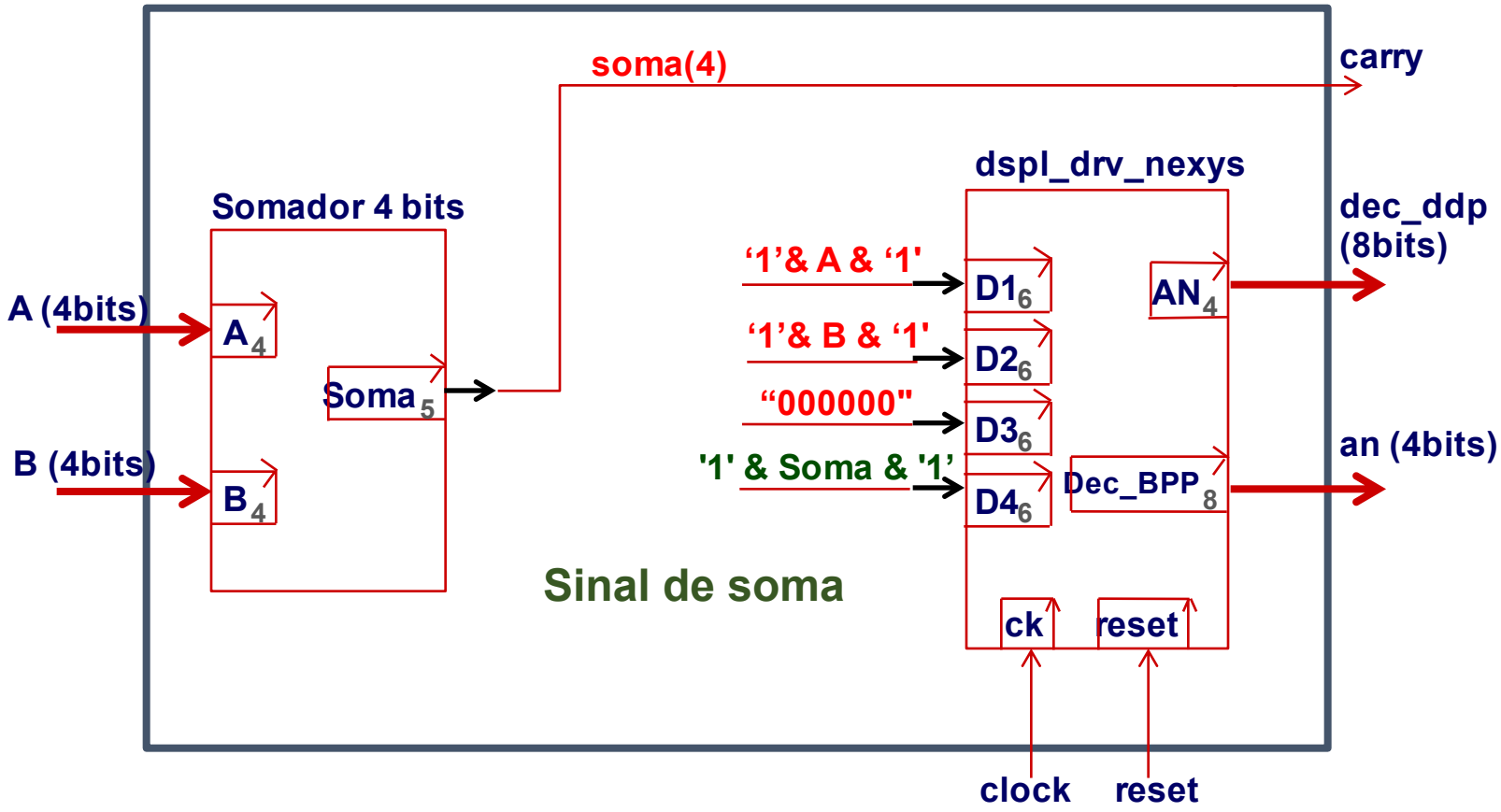
PROJETO 2

4. Ligar os sinais internos.



PROJETO 2

4. Ligar os sinais internos.



RESUMO DO TRABALHO 1B

- O **Trabalho 1B** (T1B) consiste em um arquivo compactado (.zip) contendo:
  - ✓ Um relatório em PDF descrevendo os 2 projetos de acordo com o modelo apresentado.
  - ✓ Um diretório para o Projeto 1, contendo os seguintes arquivos:
    - VHDL do somador com vai um
    - VHDL do test bench
    - arquivo .ucf
    - arquivo .bit
  - ✓ Um diretório para o Projeto 2, contendo os seguintes arquivos
    - VHDL do top com instância do somador / driver
    - arquivo .ucf
    - arquivo .bit