

IPs e Softcores

Dispositivos Lógicos Programáveis

Prof. Renan Augusto Starke

Instituto Federal de Santa Catarina – IFSC
Campus Florianópolis
renan.starke@ifsc.edu.br

22 de maio de 2016



INSTITUTO FEDERAL
SANTA CATARINA

Ministério da Educação
Secretaria de Educação Profissional e Tecnológica
INSTITUTO FEDERAL DE SANTA CATARINA

Tópicos da aula

- 1 Introdução
- 2 IPs
- 3 Arquiteturas básicas
- 4 Softcore – NIOS II
- 5 μ COSII

1 Introdução

2 IPs

3 Arquiteturas básicas

4 Softcore – NIOS II

5 μ COSII

1 Introdução

2 **IPs**

3 Arquiteturas básicas

4 Softcore – NIOS II

5 μ COSII

Intellectual Property


Um núcleo ou bloco IP é uma unidade lógica, célula ou projeto de circuito reutilizável que é parte intelectual de uma das partes: fornecedor do PLD, ferramenta de síntese, ... Este termo deriva do licenciamento de patentes ou copyright de partes do projeto. É frequentemente utilizado nos projetos de ASICs (application-specific Integrated Circuit) e FPGAs.

- ▶ A ideia é não reinventar a roda em todo o projeto
- ▶ Módulos comumente usados estão disponíveis, mas nem sempre são “livres”
- ▶ Blocos IPs implementados pelo fabricante tendem a ser otimizados para o dispositivo alvo
- ▶ IPs podem ser configurados através de uma ferramenta modificando parâmetros e adaptando-se a necessidade
- ▶ Disponíveis em VHDL ou Verilog

Exemplos:

- ▶ Soft IP cores: Nios II (Altera) e MicroBlaze (Xilinx)
- ▶ Hard IP cores: Cortex A9 + Cyclone V
- ▶ ROM, RAM, FIFO
- ▶ DSP - Multiplier
- ▶ Flash memory
- ▶ PCI, PCIe
- ▶ JTAG




RAM: 2-PORT

[About](#)
[Documentation](#)

1 Parameter Settings

2 EDA

3 Summary

General

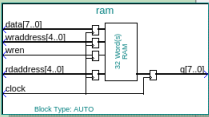
Widths/Blk Type

Clocks/Rd, Byte En

Regs/Clkens/Aclrs

Output1

Mem Init



Currently selected device family: Cyclone V

☒ Match project/default

How will you be using the dual port RAM?

☒ With one read port and one write port
☐ With two read/write ports

How do you want to specify the memory size?

☒ As a number of words
☐ As a number of bits

Resource Usage

1 M10K

Cancel

< Back

Next >

Finish

Após a configuração, a ferramenta gera um arquivo .cmp contendo a declaração do componente.

```
component data_ram
  PORT
  (
    address    : IN STD_LOGIC_VECTOR (15 DOWNT0 0);
    clock      : IN STD_LOGIC  := '1';
    data       : IN STD_LOGIC_VECTOR (15 DOWNT0 0);
    rden       : IN STD_LOGIC  := '1';
    wren       : IN STD_LOGIC ;
    q         : OUT STD_LOGIC_VECTOR (15 DOWNT0 0)
  );
end component;
```

1 Introdução

2 IPs

3 Arquiteturas básicas

4 Softcore – NIOS II

5 μ COSII

- ▶ Um sistema embarcado poder ser modelado, implementado e sintetizado através de uma linguagem de descrição de hardware
- ▶ No nosso caso: VHDL
- ▶ Construir um sistema embarcado com FPGA envolve:
 - análise dos requisitos do sistema
 - projeto de hardware
 - projeto de software

Sistema básico:

- ▶ Processador
- ▶ Memória
- ▶ Entrada e saída
 - Pinos de propósito geral
 - Comunicação
 - ...
- ▶ Temporização
- ▶ Programação e depuração

Sistema com Dispositivo Lógico Programável (PLD)

Sistema básico:

- ▶ Componentes usuário
- ▶ Componentes licenciados (IPs)
- ▶ Máquinas de estado
- ▶ Entrada e saída
 - Pinos de propósito geral
 - Comunicação
 - ...
- ▶ Temporização
- ▶ Síntese e simulação

Sistema básico:

- ▶ Processador (Softcore – IP)
 - Programação e depuração
- ▶ PLD
 - Componentes usuário
 - Componentes licenciados (IPs)
 - Temporização
 - Síntese e simulação
- ▶ Entrada e saída

- 1 Introdução
- 2 IPs
- 3 Arquiteturas básicas
- 4 Softcore – NIOS II**
- 5 μ COSII

Análise de requisitos

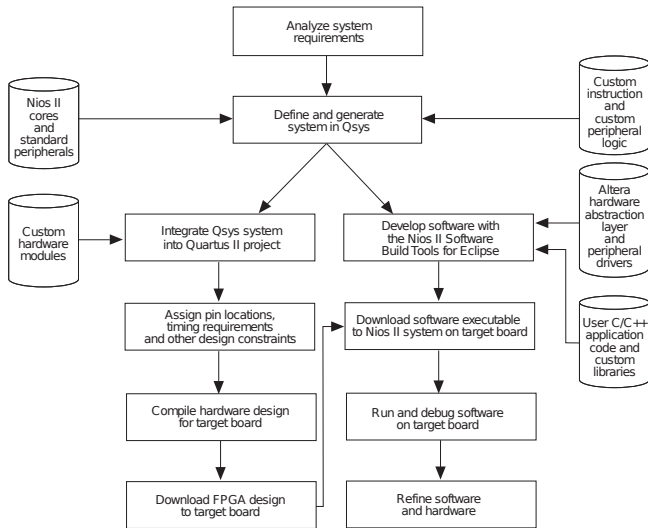
Questões:

- ▶ Qual é o desempenho computacional que a aplicação requer?
- ▶ Quanto de banda ou *throughput* a aplicação requer?
- ▶ Quais são as interfaces necessárias?
- ▶ A aplicação necessita de *software multithreaded*?

Baseando-se nestas questões:

- ▶ Que tipo de processador será necessário?
- ▶ Quais componentes e de que tipo.
- ▶ É necessário o uso de sistema operacional de tempo real?
- ▶ Lógica de aceleração para:
 - DMA?
 - Instruções especiais de DSP?

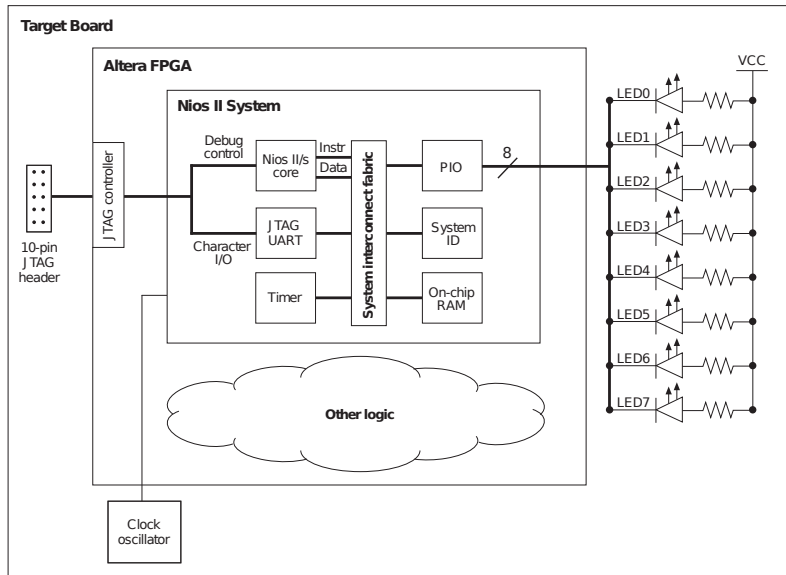
Fluxo de desenvolvimento



Componentes básicos

- ▶ Componentes necessários para o processador:
 - Núcleo do NIOS II
 - Memória interna
 - Timer
 - JTAG Serial (UART)
 - Identificação do sistema
 - Entrada e saída
- ▶ Outros componentes de hardware:
 - Conexão com a entrada e saída: botões, LEDs, ...
 - Hardware de comunicação (UART, Ethernet, ...)
 - Filtros de DSP
 - Multiplicadores
 - Unidades MAC
 - ...

Componentes básicos



Quartus → Tools → Qsys

System Contents ✕ Address Map ✕ Interconnect Requirements ✕

System: nios_2 Path: clk_0

Use	Connections	Name	Description	Export	Q
		debug_reset_request	Reset Output	<i>Double-click to</i>	[clk]
		debug_mem_slave	Avalon Memory Mapped Slave	<i>Double-click to</i>	[clk]
		custom_instruction_ma...	Custom Instruction Master	<i>Double-click to</i>	
<input checked="" type="checkbox"/>		jtag_uart_0	JTAG UART		
		clk	Clock Input	<i>Double-click to</i>	clk
		reset	Reset Input	<i>Double-click to</i>	[clk]
		avalon_jtag_slave	Avalon Memory Mapped Slave	<i>Double-click to</i>	[clk]
		irq	Interrupt Sender	<i>Double-click to</i>	[clk]
<input checked="" type="checkbox"/>		timer_0	Interval Timer		
		clk	Clock Input	<i>Double-click to</i>	clk
		reset	Reset Input	<i>Double-click to</i>	[clk]
		s1	Avalon Memory Mapped Slave	<i>Double-click to</i>	[clk]
		irq	Interrupt Sender	<i>Double-click to</i>	[clk]
<input checked="" type="checkbox"/>		sysid_qsys_0	System ID Peripheral		
		clk	Clock Input	<i>Double-click to</i>	clk
		reset	Reset Input	<i>Double-click to</i>	[clk]
		control_slave	Avalon Memory Mapped Slave	<i>Double-click to</i>	[clk]
<input checked="" type="checkbox"/>		prio_0	PIO (Parallel I/O)		
		clk	Clock Input	<i>Double-click to</i>	clk
		reset	Reset Input	<i>Double-click to</i>	[clk]
		s1	Avalon Memory Mapped Slave	<i>Double-click to</i>	[clk]
		external_connection	Conduit	<i>Double-click to</i>	prio_0_external_con...

Instância e interconexão com o PLD

Após a geração do QSYS, deve-se instanciar o *softcore* no projeto.

```
architecture rtl of DE2_115 is

    component nios_2 is
        port (
            clk_clk          : in  std_logic := 'X'; -- clk
            reset_reset_n    : in  std_logic := 'X'; -- reset_n
            pio_0_external_connection_export : out std_logic_vector(7 downto 0)
        );
    end component nios_2;

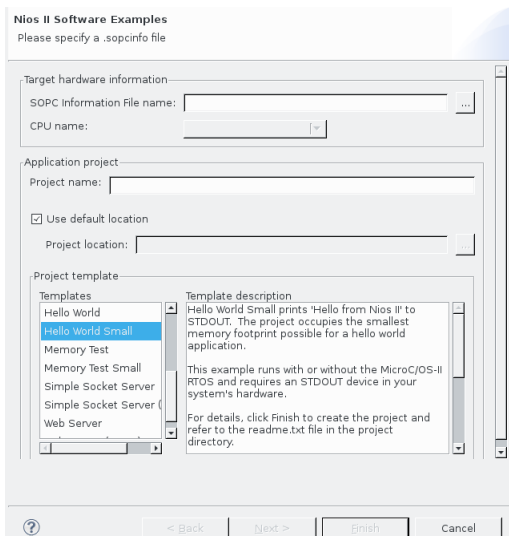
    signal reset_n : std_logic;
    signal pll_locked : std_logic;
    signal phase_done : std_logic;
begin

    reset_n <= '1';

    cpu_0 : component nios_2
        port map ( clk_clk => CLOCK_50, reset_reset_n => reset_n,
            pio_0_external_connection_export => LEDG(7 downto 0)
        );
end;
```

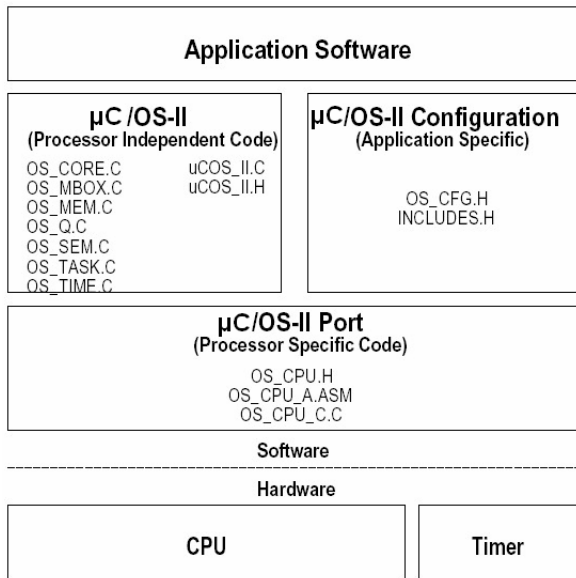
Geração dos componentes de *software*

Quartus → Tools → Nios II Software Build Tools for Eclipse
Eclipse → File → Nios Application and BSP from Template



- 1 Introdução
- 2 IPs
- 3 Arquiteturas básicas
- 4 Softcore – NIOS II
- 5 μ COSII

- ▶ Dependendo da aplicação, é interessante utilizar um Sistema Operacional
- ▶ É possível utilizar o μ COSII em conjunto com o PLDs da Altera
- ▶ μ COSII é um sistema operacional para tempo-real projetado para sistemas embarcados. Outros exemplos:
 - FreeRTOS
 - VxWorks
- ▶ μ COSII possui licença comercial, mas livre para aplicações educacionais



Geração dos componentes de *software*

Quartus → Tools → Nios II Software Build Tools for Eclipse
Eclipse → File → Nios Application and BSP (Board Support Package) from Template

Nios II Software Examples
Please specify a .sopcinfo file

Target hardware information

SOPC Information File name: ...

CPU name: ▼

Application project

Project name:

☒ Use default location

Project location: ...

Project template

Templates	Template description
Float2 Functionality	
Float2 GCC	
Float2 Performance	
Hello Freestanding	
Hello MicroC/OS-II	Hello MicroC/OS-II uses the MicroC/OS-II RTOS. You can use this example as a starting point for developing Nios II MicroC/OS-II applications.
Hello World	For details, click Finish to create the project and refer to the readme.txt file in the project directory.
Hello World Small	The BSP for this template is based on the Micrium MicroC/OS-II operating system.
	For information about how this software example relates to Nios II hardware design examples

► Acesso ao hardware pelo *software*:

```
//-----  
#include "altera_avalon_pio_regs.h"           // Macros de acesso as portas  
      de E/S  
  
IOWR_ALTERA_AVALON_PIO_DATA(base_addr, data); // ← Escrita  
x = IORD_ALTERA_AVALON_PIO_DATA(base_addr);   // ← Leitura  
  
//-----  
#include "system.c"           //Endereço do periférico conforme QSYS  
#define LEDR_BASE (...)   
#define SEG7_BASE (...)   
(...)
```

