Síntese comportamental de componentes de um Sistema Operacional em hardware

João Paulo Pizani Flor
joaopizani@lisha.ufsc.br

Orientador: Prof. Dr. Antônio Augusto Fröhlich
Co-orientador: Tiago Rogério Mück

15 de Junho de 2011

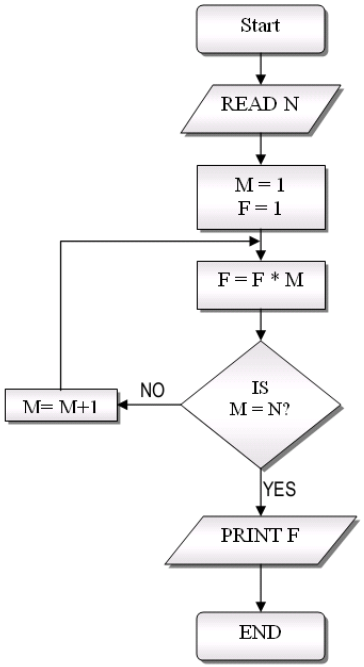
Tópicos

- Motivação / Objetivo geral
- Conceitos fundamentais
- Objetivos específicos
- Desenvolvimento
- Resultados
- Conclusões

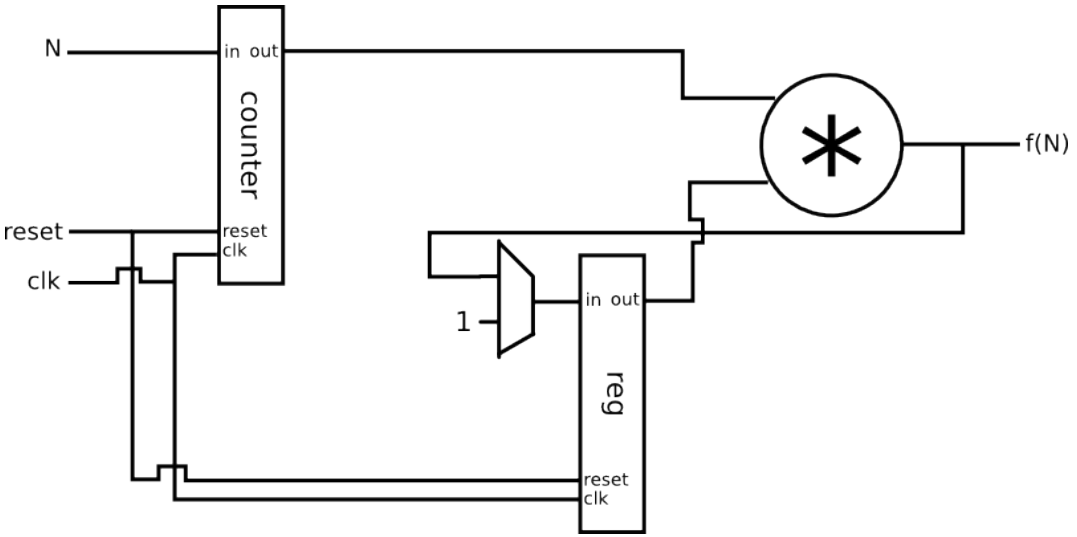
Motivação / Objetivo geral

Motivação / Objetivo geral

Algoritmo



Arquitetura de hardware

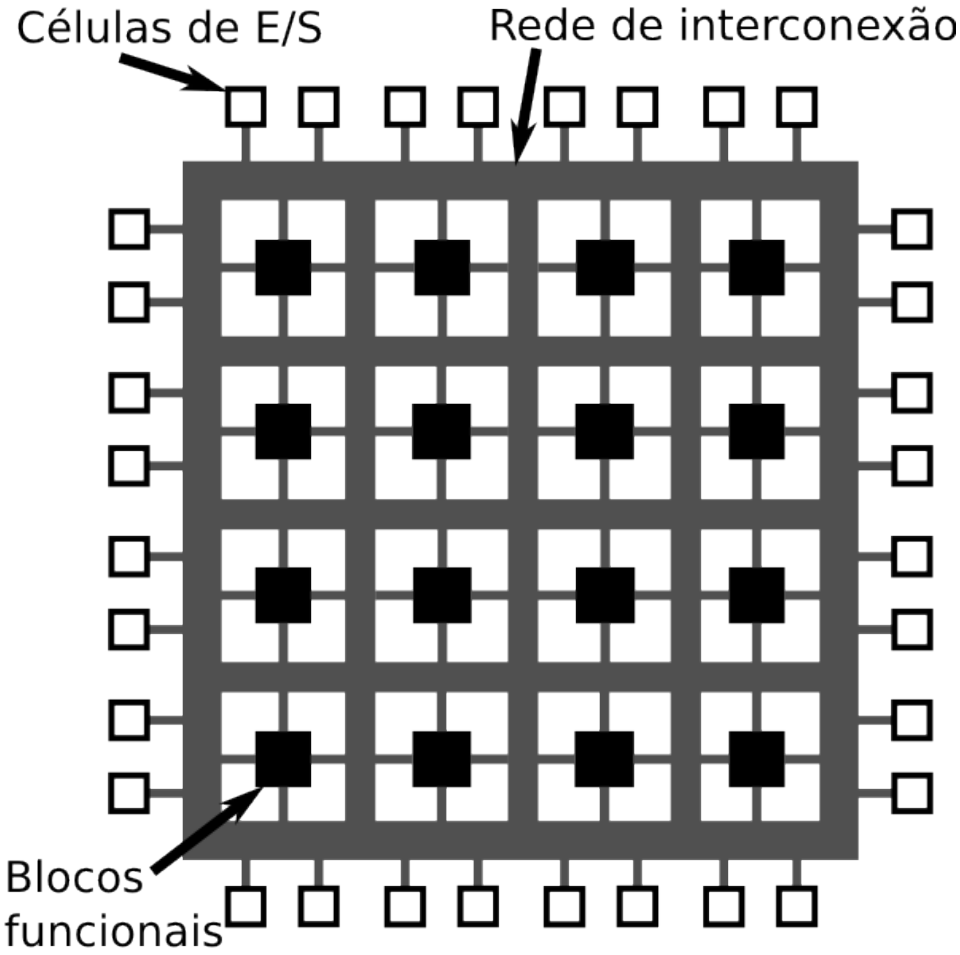


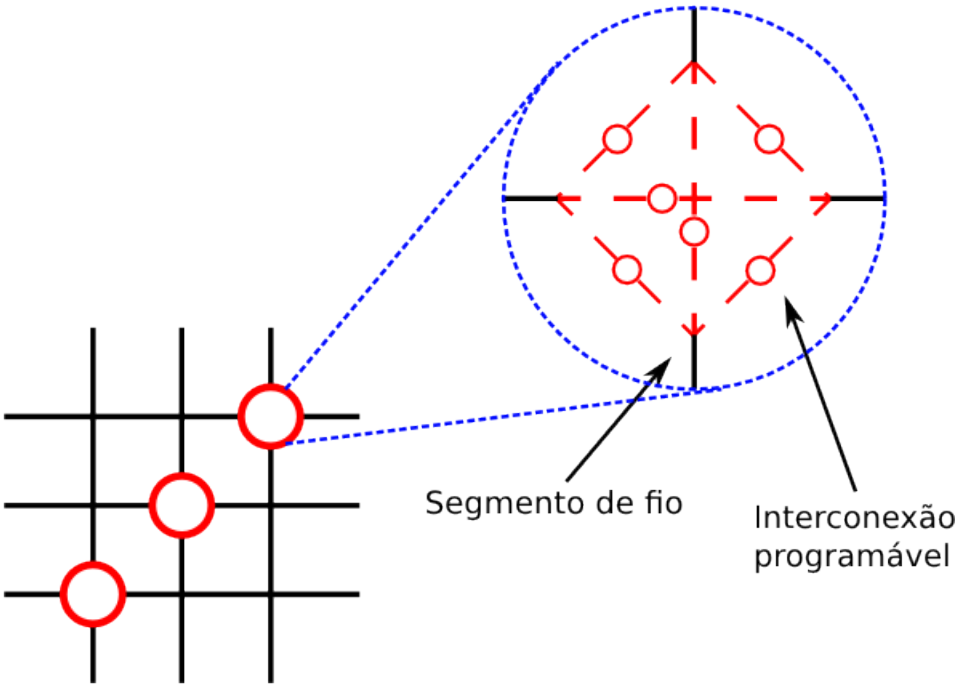
- Estudar a viabilidade de sintetizar para hardware algoritmos descritos em alto nível de abstração
  - Implementar um escalonador para o EPOS, usando síntese de alto nível

Conceitos fundamentais

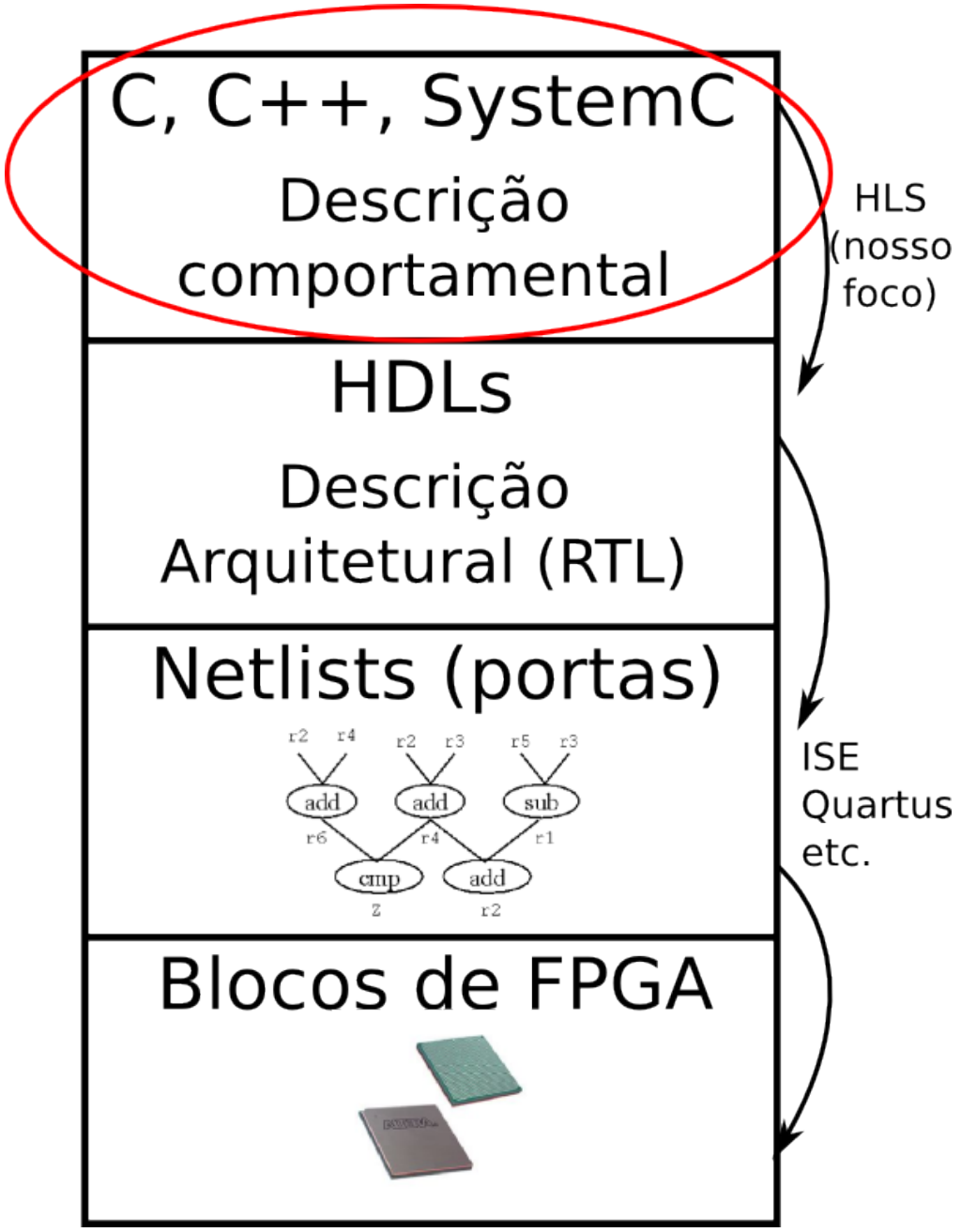
Hardware reconfigurável / FPGA

- "Malha" de blocos lógicos, totalmente flexível





Algoritmos em hardware



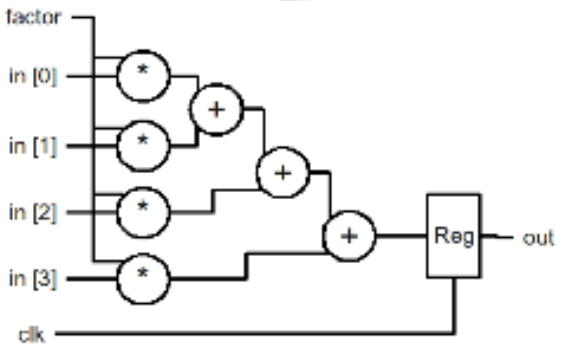
- Comportamento: **o quê** o algoritmo deve fazer
- Arquitetura: **quais** operadores usar e **quando** acioná-los

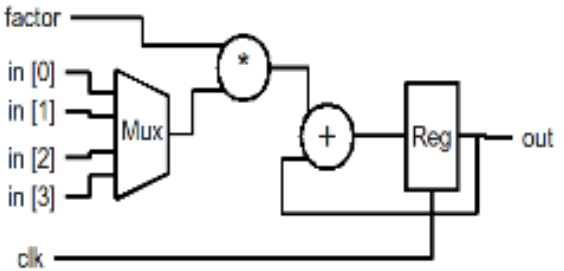
Síntese de alto nível

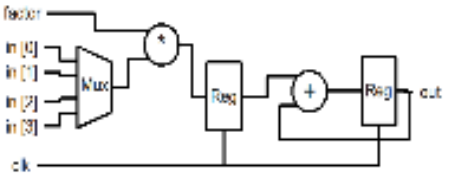
```
int mult_acc_array(int input[SIZE], int factor) {
        int result = 0;

        for (int i = 0; i < SIZE; ++i)
                result += input[ i ] * factor;

        return result;
}
```

System Constraints

HLS







Síntese de alto nível

C/C++

Análise do código-fonte
(Detecção de loops,
parâmetros de funções, etc.)

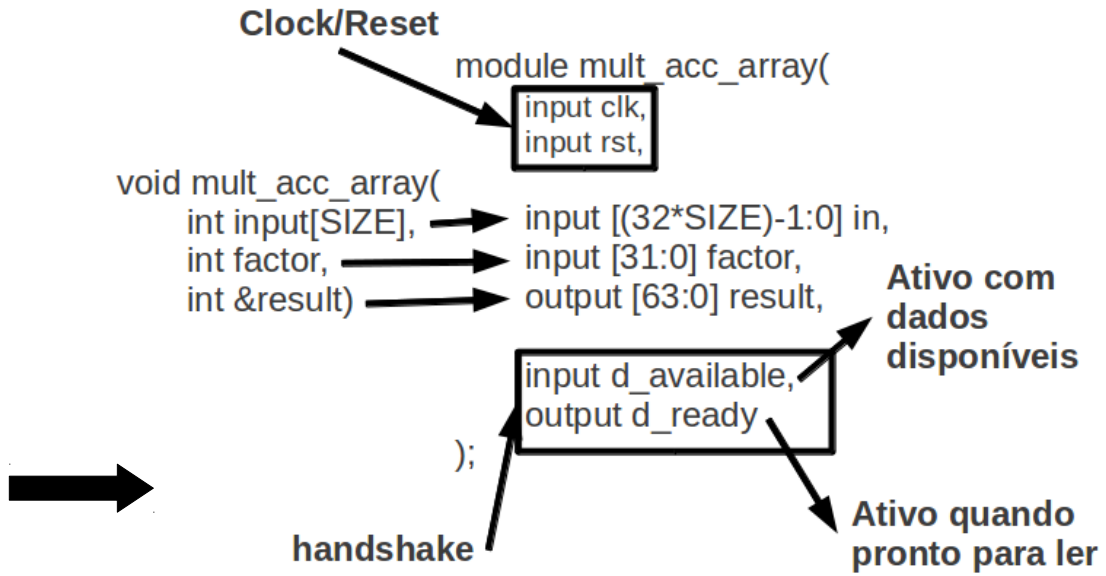
Definição da macroarquitetura
(Frequência de clock, tipo de reset
tecnologia subjacente)

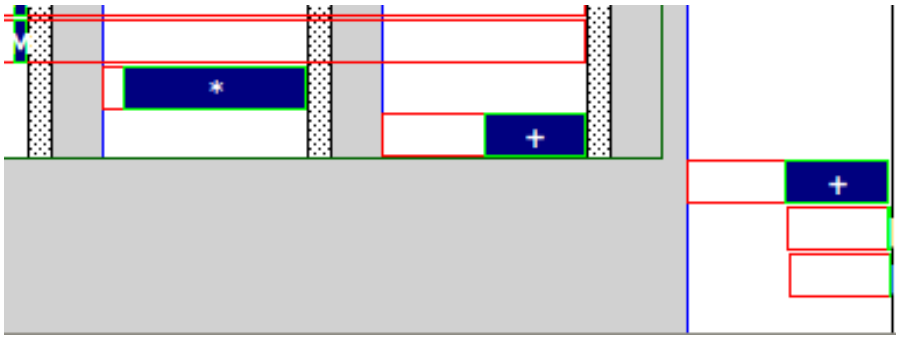
Definição da microarquitetura
(Protocolos de E/S, mapeamento
de arrays para RAM/ROM)

Escalonamento
(Paralelização/fusão de loops,
criação de pipelines)

Geração de RTL
(Geração de código VHDL/Verilog
ou netlists para síntese)

VHDL/Verilog





EPOS

- Embedded Parallel Operating System
  - Sistema Operacional orientado à aplicação
  - Baseado em componentes, utiliza metaprogramação
  - Configurabilidade e **portabilidade** com baixo overhead

- Conceito de **componente híbrido** SW/HW
  - Arquitetura descrita em (MARCONDES, 2009)
  - Idealmente código único

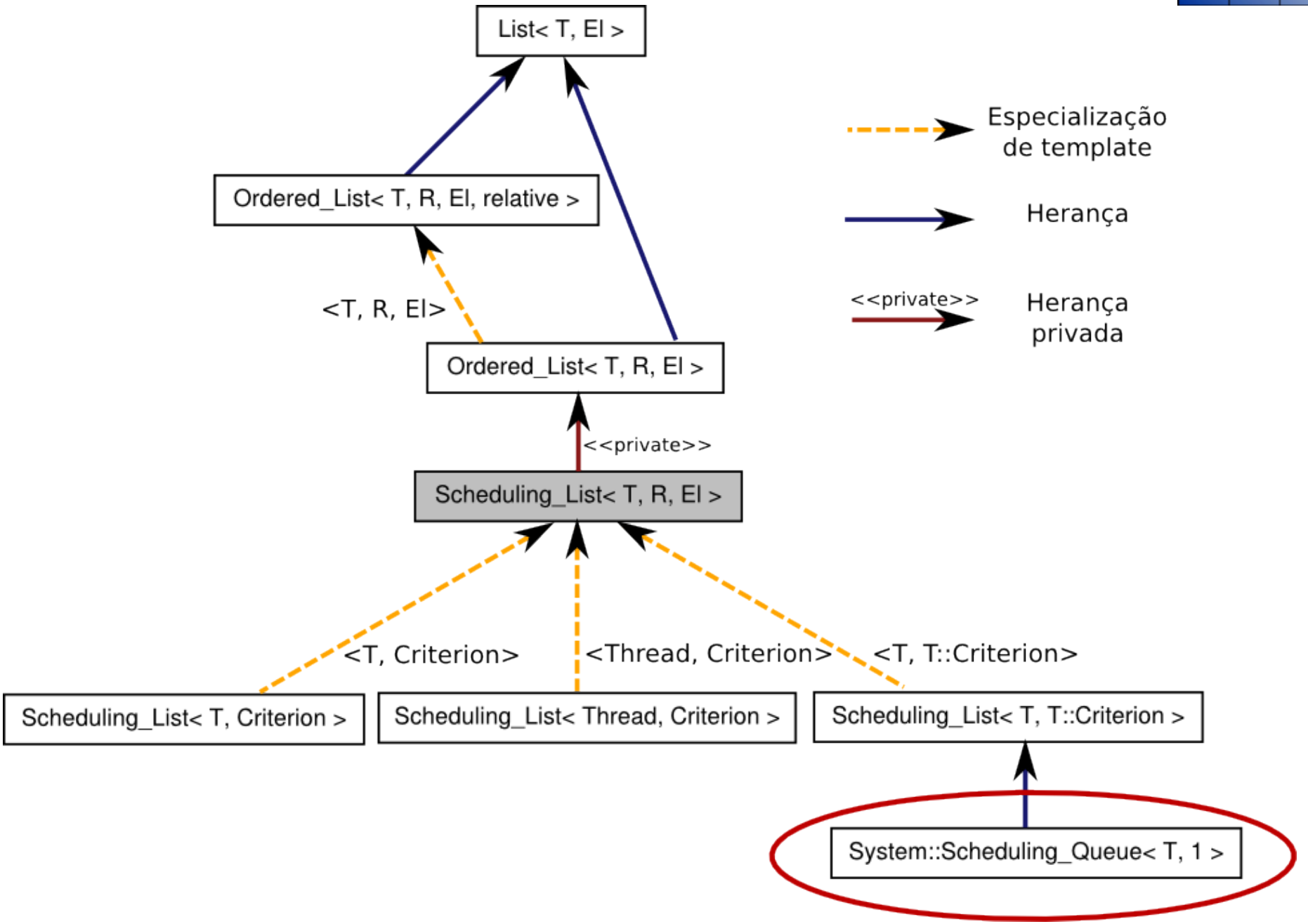
Objetivos específicos

Objetivos específicos

- Descrever um escalonador para o sistema EPOS em C++ padrão
  - Sintetizar esse escalonador para um dispositivo FPGA
  - Verificar o componente gerado com um testbench VHDL
- Coletar dados de área e atraso de algumas das microarquiteturas possíveis
- Comparar com a implementação de referência
  - Escalonador RTL descrito em (MARCONDES, 2009)

Desenvolvimento

Estruturas de dados utilizadas



Limitações impostas

- Utilizamos o Catapult-C®, da Mentor Graphics

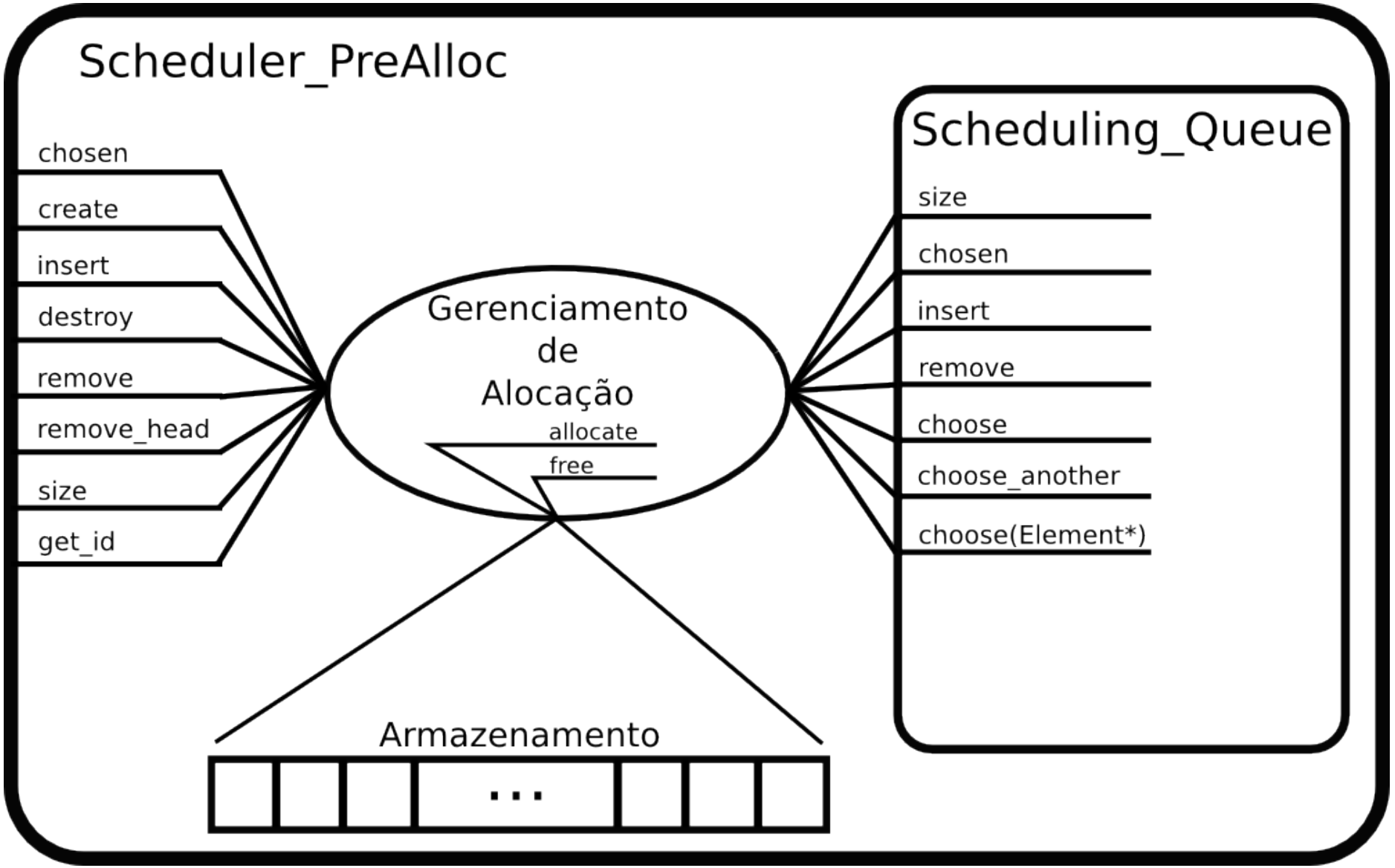
- Suporte bastante completo à linguagem C++, com algumas limitações:
  - Ponteiros sem alvo (*dangling pointers)*
  - Alocação dinâmica de memória
  - Uma função define a interface do hardware

Alterações/Adições - Maybe<T>

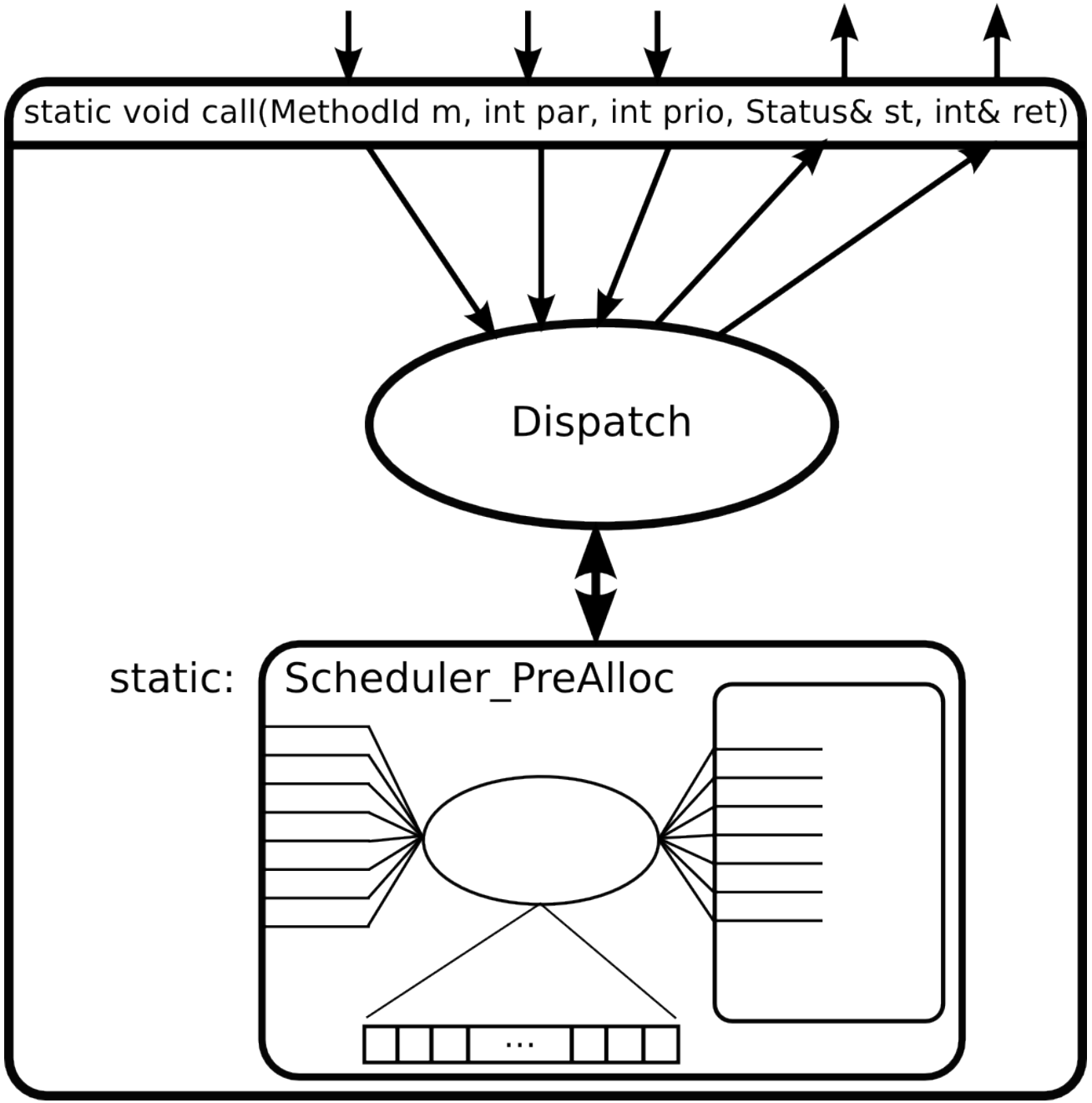
- O uso de ponteiros deve ser **sintetizável**
  - Devem referenciar objetos estaticamente conhecidos
  - Ponteiros inválidos impedem já a 1ª etapa de síntese
  - Estruturas EPOS fazem uso intensivo de ponteiros nulos

- Um tipo Maybe<T>:
  - Representa uma operação que **pode falhar**, mas em caso de sucesso retorna um valor de tipo T
  - Dois construtores possíveis: vazio (falha) e com 1 parâmetro

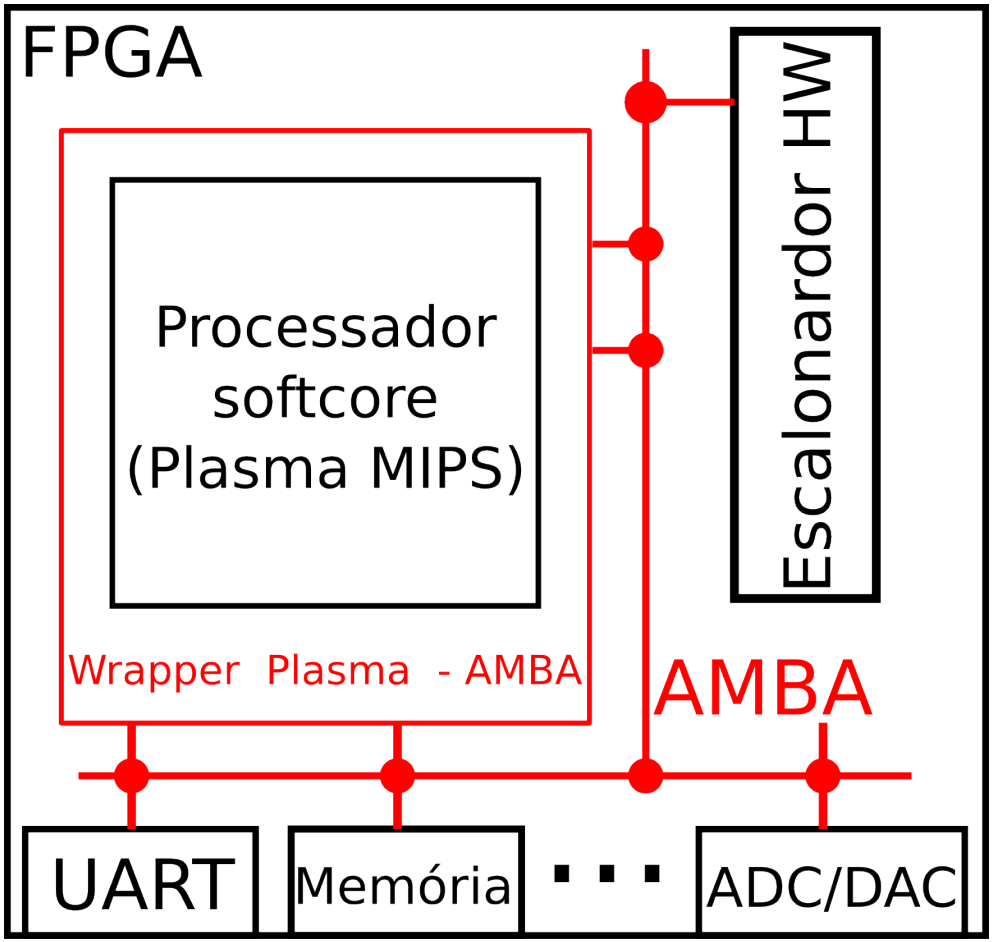
Alterações/Adições - Alocação



Alterações/Adições – Interface raiz



Ambiente de execução



- EPOS em um MIPS32
- Escalonador como *slave* em barramento, mapeado em memória

Resultados

Cenários de síntese

- Dispositivo: Spartan 3E, XC3S200-FT256
- Cada cenário se encontra detalhado no relatório

	Pré-síntese RTL (Catapult-C)		Pós síntese RTL (Xilinx ISE 13)	
	Área (score)	Throughput (ciclos)	Área (LUTs)	Atraso máximo
Serial com memória	4301.698	79	1654 (1.1%)	6.672 ns
Serial diretivas *default*	6496.576	43	3392 (2.25%)	7.341 ns
Paralelizado	9847.783	4	5121 (3.4%)	10.597 ns
RTL de referência	N/A	N/A	1250 (0.83%)	5.598 ns

Resultados


Área x Atraso pós-síntese RTL
Atraso máximo (ns)
12
10
8
6
4
2
0
1000
1500
2000
2500
3000
3500
4000
4500
5000
5500
Área ocupada (LUTs)
RTL
Cenário 1
Cenário 2
Cenário 3

Conclusões e trabalhos futuros

Conclusões

- Confirmada viabilidade de síntese C++ para RTL
  - Melhor que o esperado (poucas alterações nos fontes)
  - Boa modelagem das estruturas utilizadas (EPOS) foi fundamental
- Métodos desenvolvidos de “adaptação” ao hardware podem ser generalizados
  - Gerenciamento de alocação
  - *Dispatch* de chamadas e tratamento de parâmetros

Trabalhos futuros

- Escalonador como primeiro **componente híbrido** do EPOS com descrição única
  - Configurar sistema de compilação, etc.

- Generalização da adaptação de objetos para cenário de execução em hardware
  - Estudar a comunicação entre **objetos em hardware**

Muito Obrigado!

Always look on the bright side of life...

Monty Python – The Life of Brian