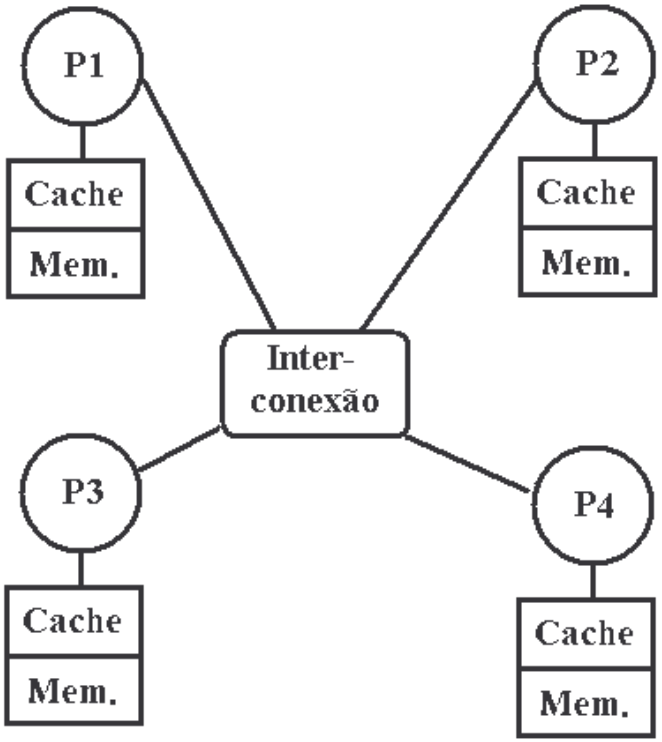
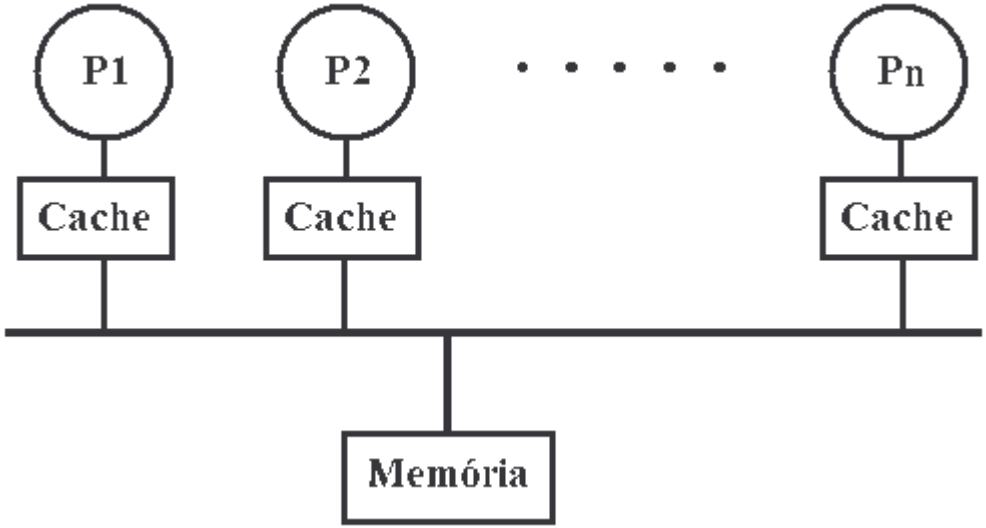
Arquitetura de multiprocessadores (I)

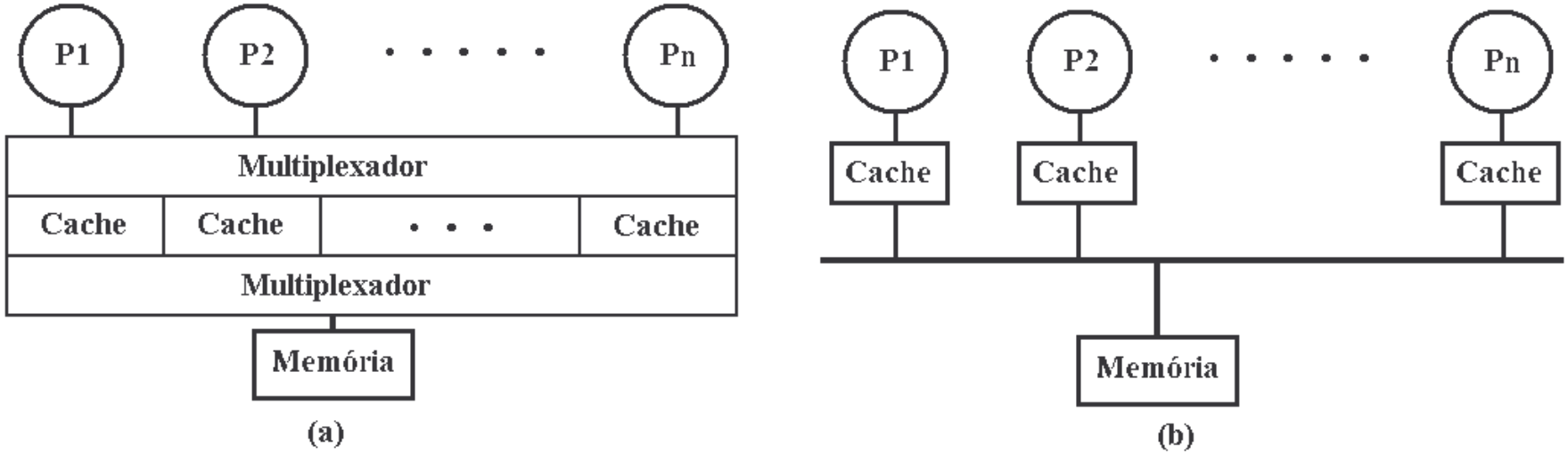
- Origem nos supercomputadores da década de 60 e 70.
- Caches procuram reduzir o tráfego de dados no barramento.
- Duas arquiteturas básicas: NUMA e UMA.
- NUMA: maior complexidade para manter coerência de dados.





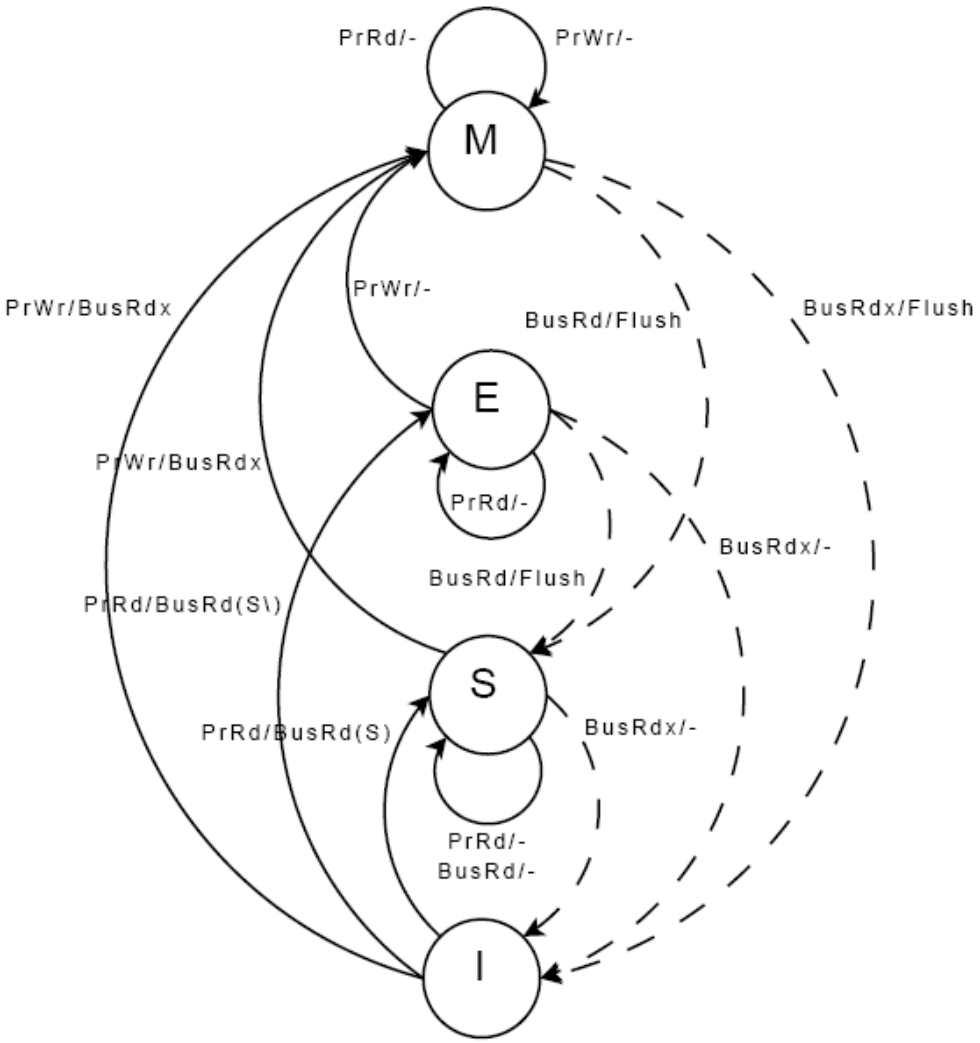
Arquitetura de multiprocessadores (II)

- Arquitetura UMA:
  - Cache L1 compartilhada: aumento do tempo de acerto.
  - Cache L1 individual: pouca diferença de desempenho em relação à compartilhada, mas necessita de protocolo de coerência de caches.



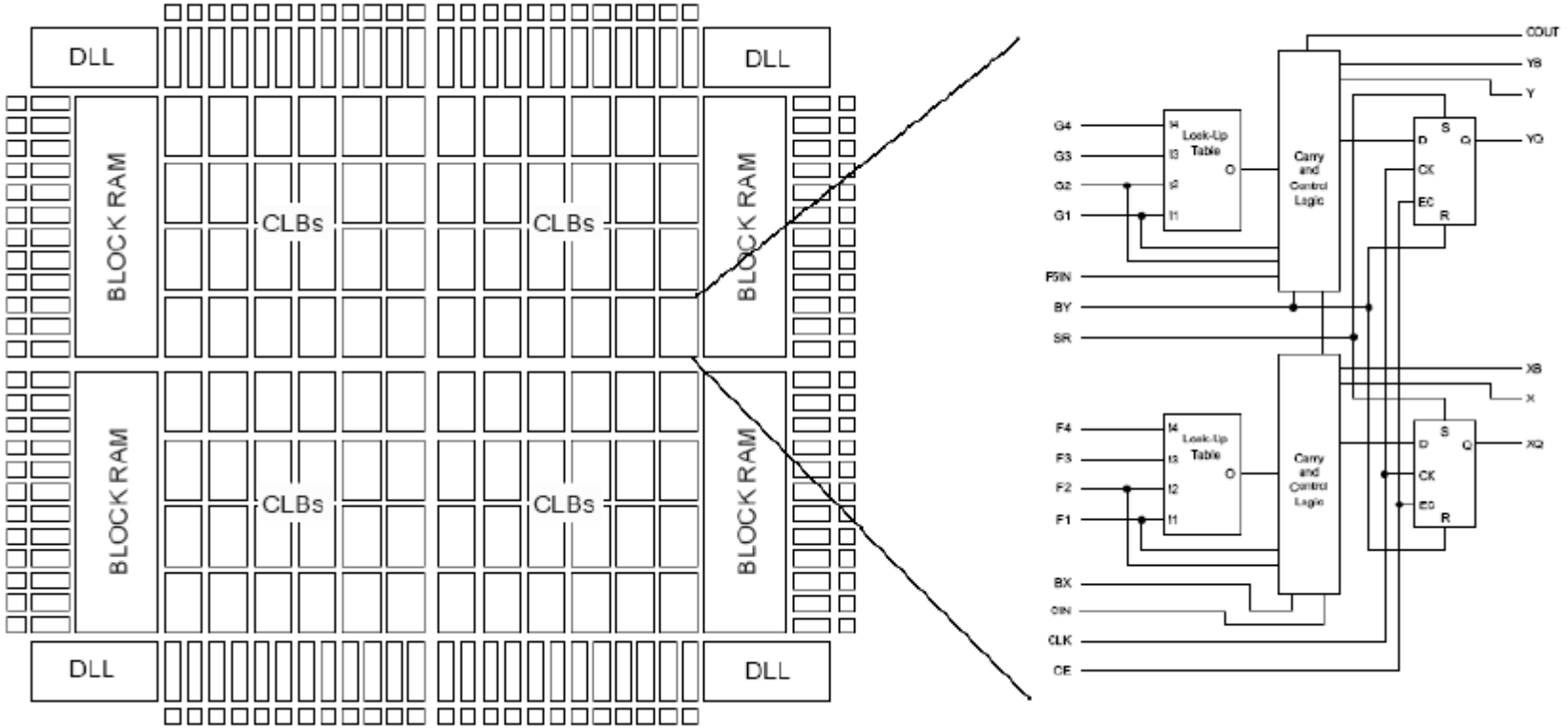
Coerência de Caches - MESI

- Parte da classe de protocolos de espionagem M(O)(E)SI.
- Estado exclusive permite passar diretamente para modified.
- Usado na IA32 (Pentium).
- Deficiência: P1 escrevendo e P2 lendo constantemente: necessita de Flush (M→S) e BusRdx (S→M)!
- Mais eficiente que o MSI e mais simples que o MOESI.



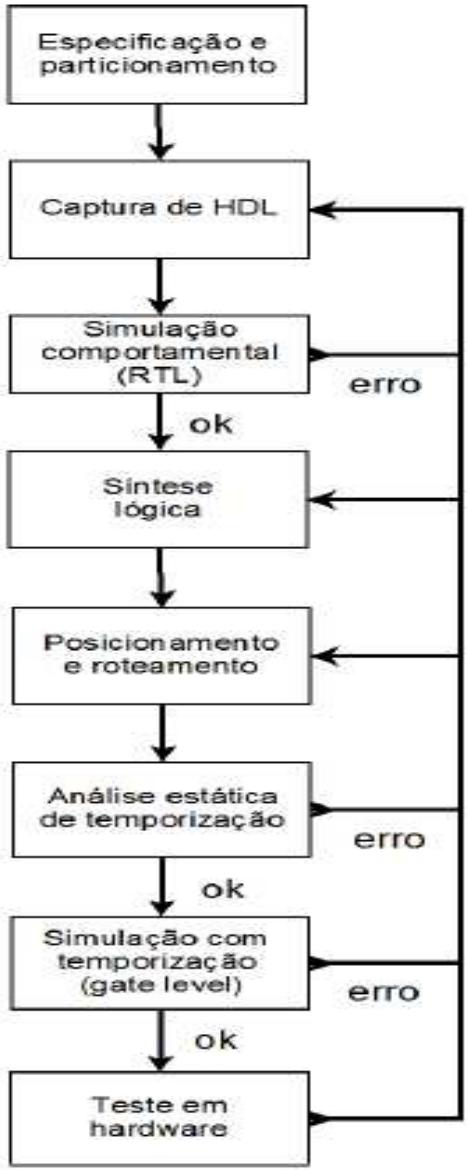
Dispositivos Programáveis (I)

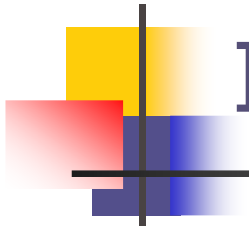
- FPGAs: Desenvolvidas concorrentemente às CPLDs: maiores e baseadas em RAM.
- Matriz de células lógicas + recursos de interconexão.



Dispositivos Programáveis (II)

- Desenvolvimento de Dispositivos programáveis: no início através de esquemáticos lógicos.
- Agora através de linguagens de descrição de HW: VHDL e Verilog.
- Descrições são simuladas e depois sintetizadas.

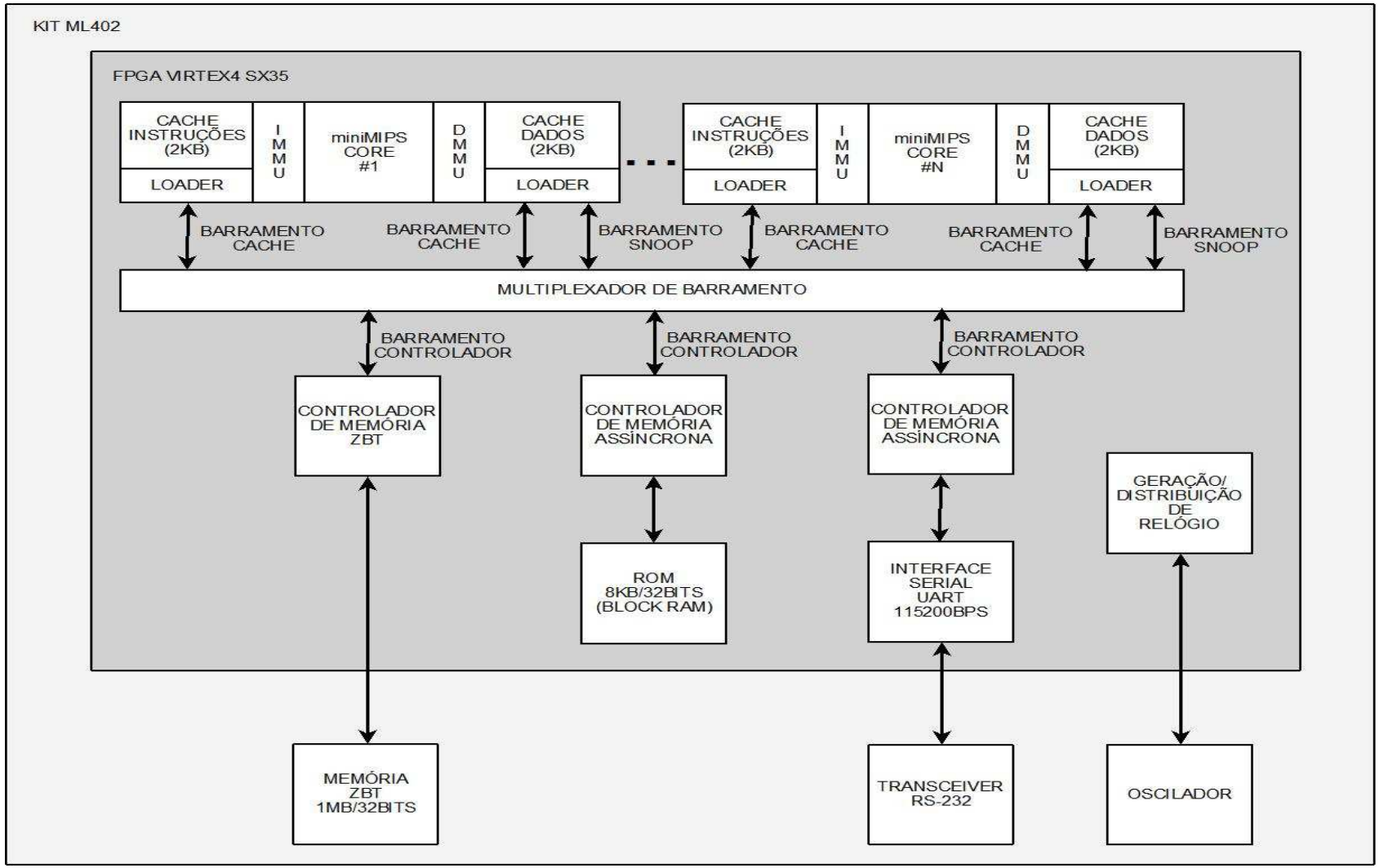




Implementação – miniMIPS

- Implementação em VHDL do conjunto de instruções do MIPS-I disponível no opencores.org.
- Somente núcleo básico, sem periféricos, controladores de memória ou caches.
- Problemas: alguns problemas de implementação de instruções, instruções não implementadas (div, manipulação de bytes).
- MMCC – Multiprocessador Minimalista com Cache Coerentes - implementa caches, suporte à multiprocessamento, controladores de memória, protocolo de coerência de dados e unidade de gerenciamento de memória.

Implementação – MMCC

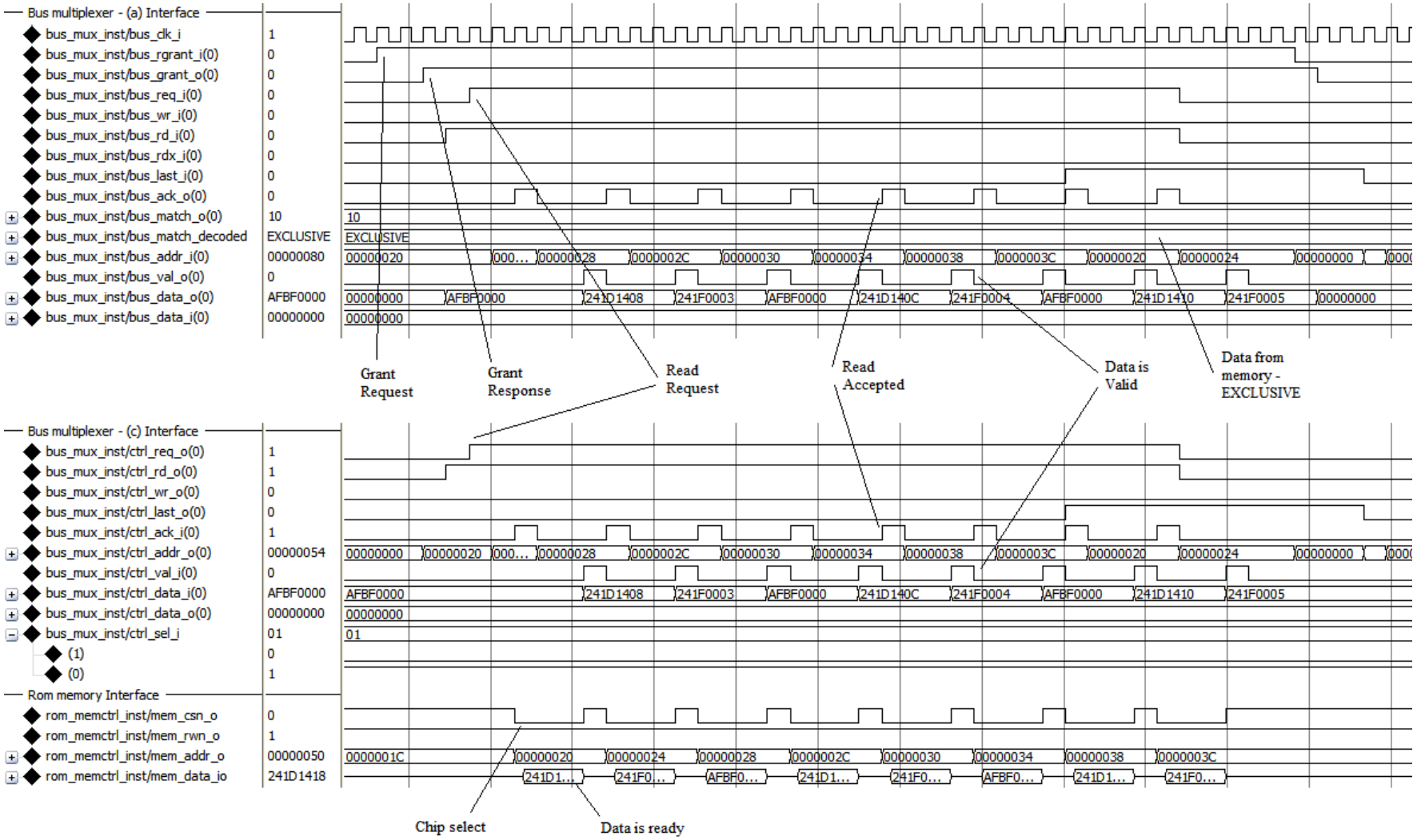


Implementação – Barramento (I)

- Três tipos de interfaces:
  - Controladores: controladores de memória ou periféricos como interfaces seriais.
  - Caches: caches de dados e instruções. São as que iniciam as requisições.
  - Snoop: similar à interface dos controladores, mas com sinais para indicar o estado do bloco da cache.

- Multiplexador de barramento:
  - Arbitragem dos acessos.
  - Espionagem.
  - Controle de acesso exclusivo.

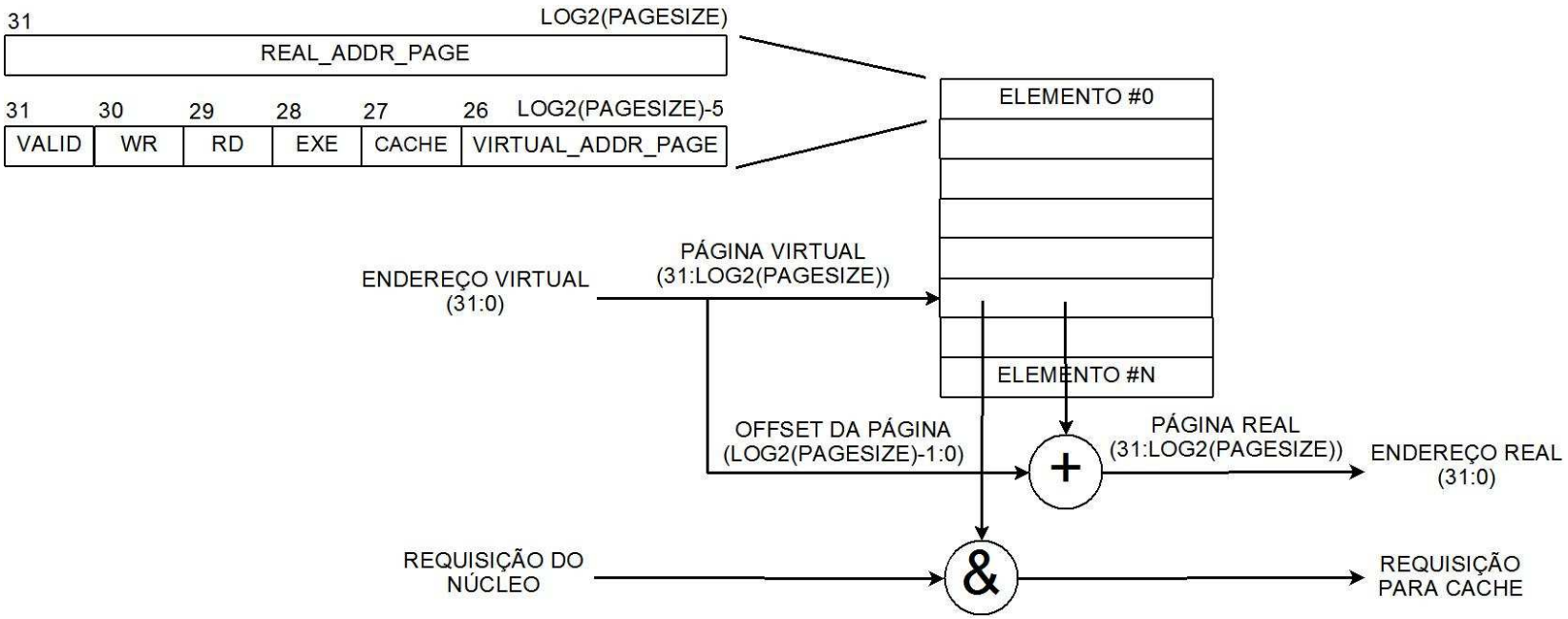
Implementação – Barramento (II)



Implementação – Memória Virtual

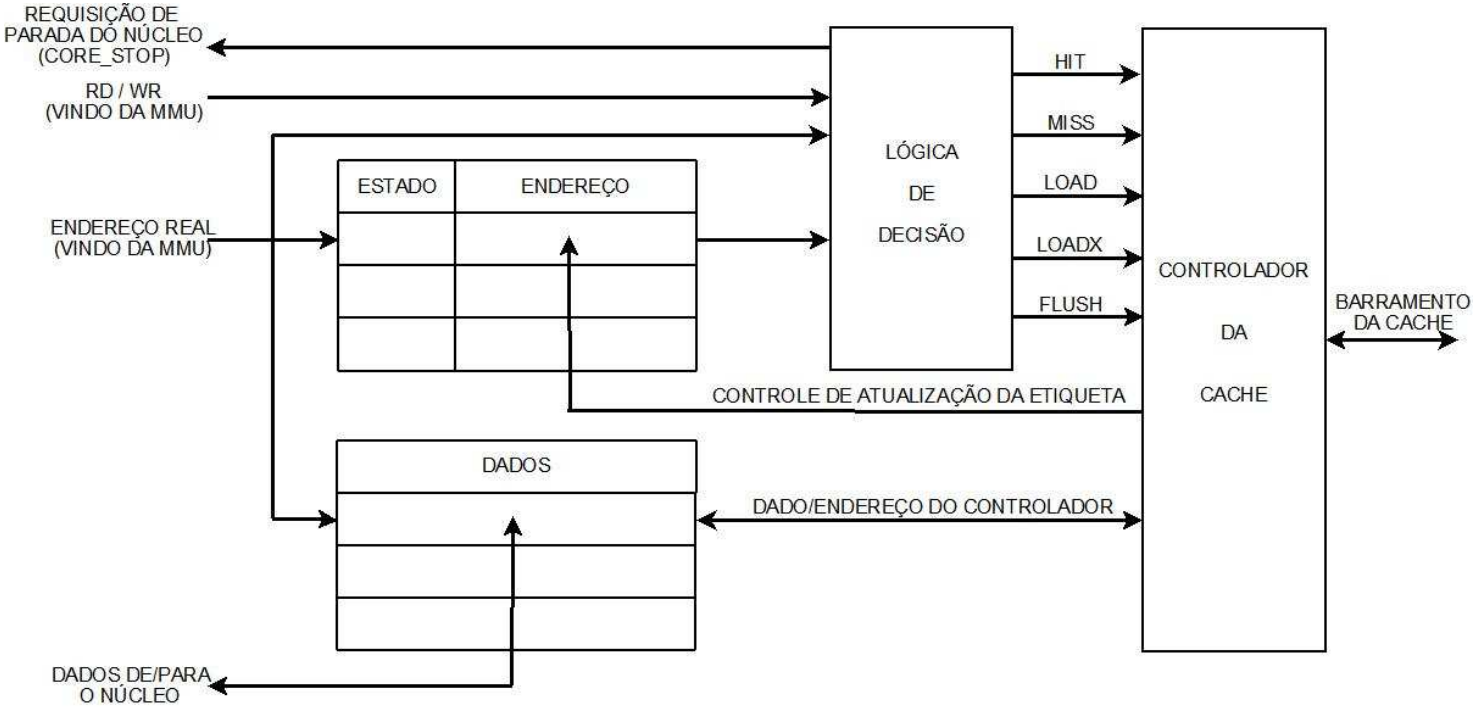
- Memória virtual: compartilhamento de dados/programas e realocação da área de dados de cada processador.

- Proteções básicas de memória: leitura, escrita e habilitação da cache.



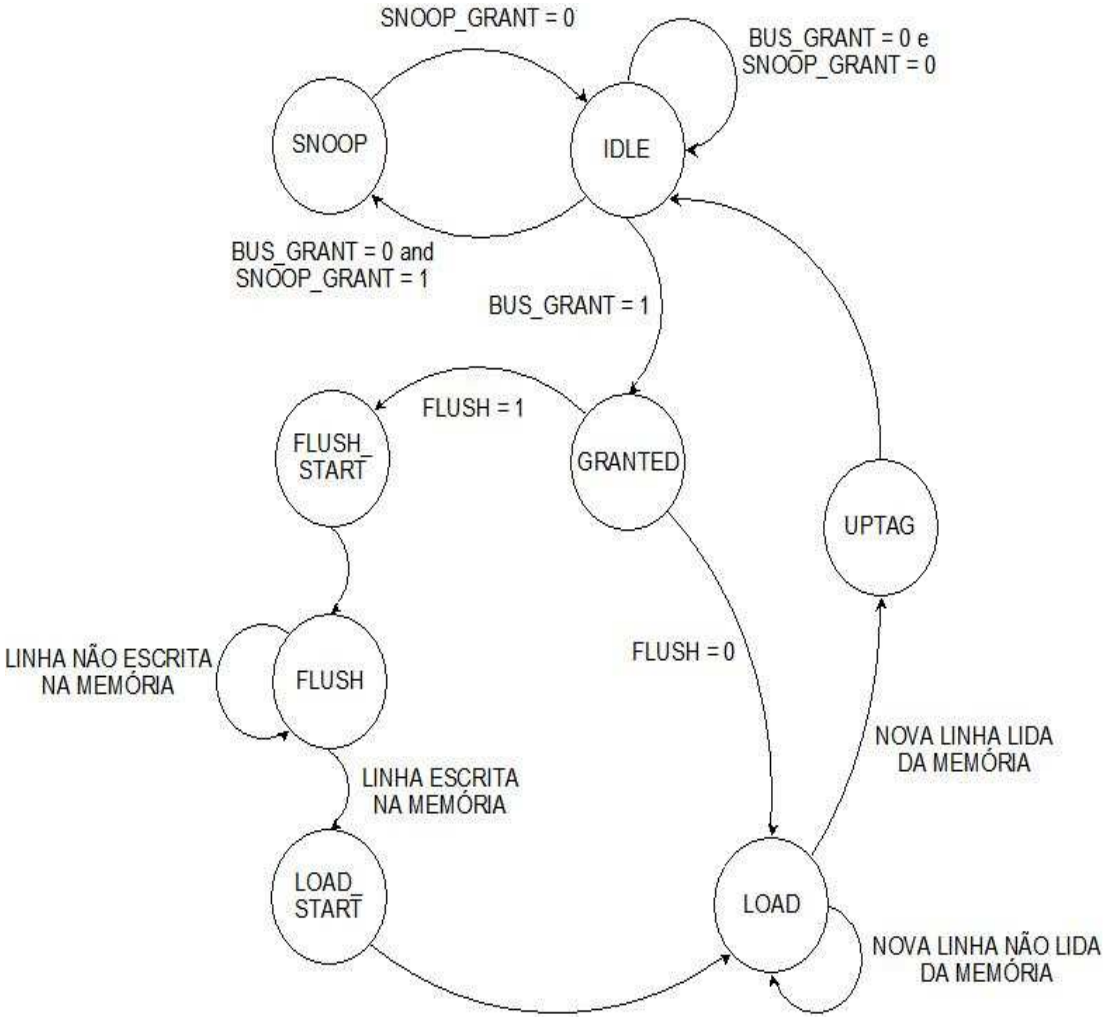
Implementação – Caches

- Diretamente mapeadas.
- Configuráveis em tamanho total e bloco.
- Caches de instruções não-coerentes e dados coerentes: protocolo similar ao MESI.



Implementação – Coerência

- Poucos estados para implementação.
- Diferença para o MESI: blocos modificados não são escritos na memória mas transferidos diretamente entre as caches.
- Evita dois acessos custosos à memória externa e lógica para interromper acessos.

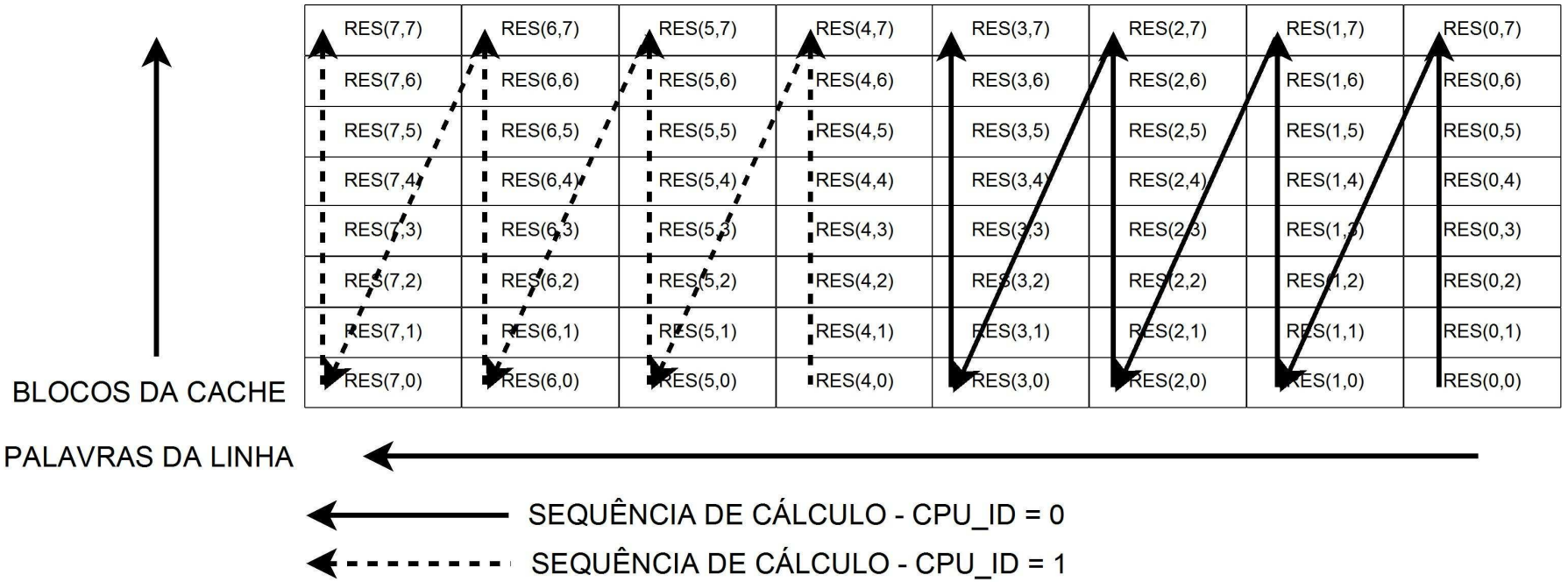


Implementação – Programa de Teste (I)

- Multiplicação de matrizes 8x8 de inteiros: assembly e C, compilado com gcc.
- Programa e matrizes compartilhadas – uso intensivo do barramento.
- Variáveis privadas (controle + pilha).

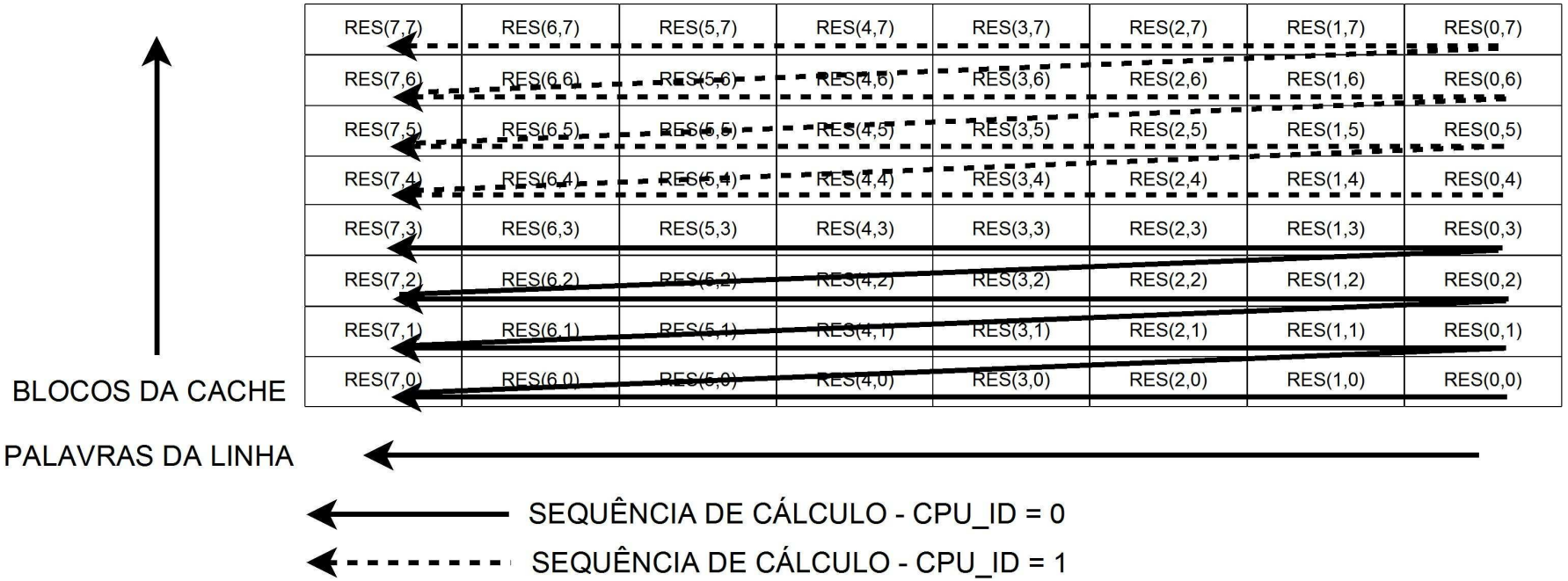
Implementação – Programa de Teste (II)

- Cada processador é responsável por calcular alguns termos de cada linha (APL_COL).
- Blocos da cache são compartilhados no estado modified.



Implementação – Programa de Teste (III)

- Cada processador é responsável por calcular algumas linhas da matriz (APL_LIN).
- Blocos da cache são compartilhados no estado shared.



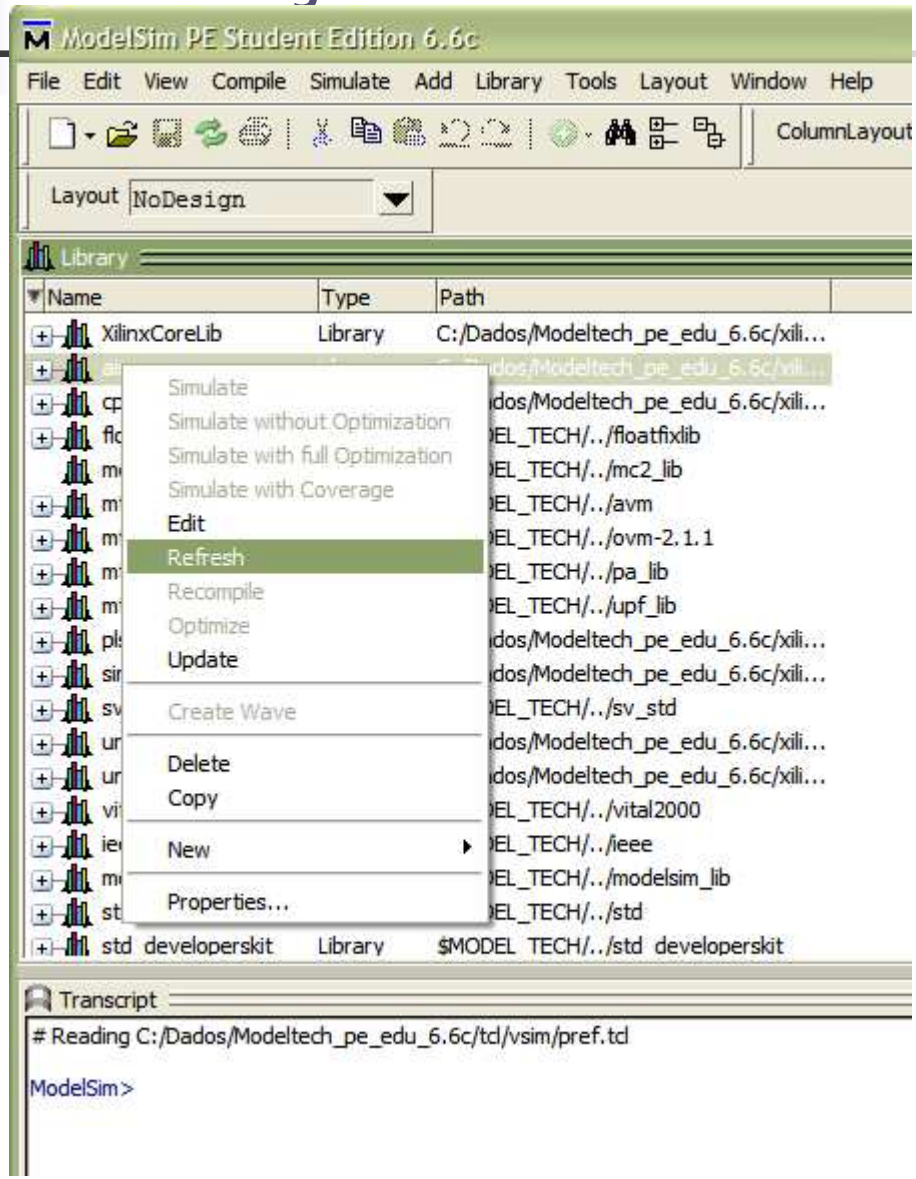
Simulação - Ferramentas (I)

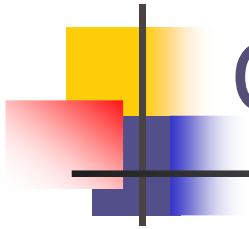
- Simulador VHDL: ModelSim para Windows versão estudantil:
  - Baixar (125MB): http://model.com/content/modelsim-pe-student-edition-hdl-simulation
  - Ao final da instalação será aberto um formulário para pedir licença do SW que será enviada por e-mail. Copiar o arquivo de licença recebido para o diretório de instalação.
  - Copiar as bibliotecas da Xilinx para o diretório de instalação:
    - https://secure.xilinx.com/webreg/register.do?group=dlc&htmlfile=&emailFile=&cancellink=&eFrom=&eSubject=&version=12.3&akdm=1&filename=MXE6.5c_12.3_simulation_libraries.zip

Simulação - Ferramentas (II)

- Adicionar as linhas ao arquivo modelsim.ini:
  - aim = C:/Dados/Modeltech_pe_edu_6.6c/xilinx/vhdl/aim
  - cpld = C:/Dados/Modeltech_pe_edu_6.6c/xilinx/vhdl/cpld
  - pls = C:/Dados/Modeltech_pe_edu_6.6c/xilinx/vhdl/pls
  - simprim = C:/Dados/Modeltech_pe_edu_6.6c/xilinx/vhdl/simprim
  - unimacro = C:/Dados/Modeltech_pe_edu_6.6c/xilinx/vhdl/unimacro
  - unisim = C:/Dados/Modeltech_pe_edu_6.6c/xilinx/vhdl/unisim
  - XilinxCoreLib = C:/Dados/Modeltech_pe_edu_6.6c/xilinx/vhdl/XilinxCoreLib
- Fazer refresh nas bibliotecas novas:
  - Abrir modelsim
  - Selecionar as bibliotecas aim, cpld, pls, simprim, unimacro, unisim e XilinxCoreLib. Com o botão direito

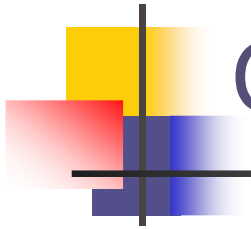
Simulação - Ferramentas (III)





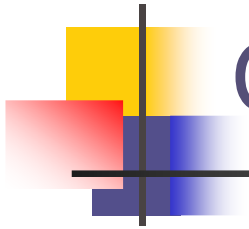
Código - Compilação

- Instalar pacotes no Cygwin (fw/source):
  - Binutils-2.14.tar.bz2
  - Gcc-3.3.1.tar.bz2
  - Newlib-1.11.0.tar.gz
  - Exemplo: FW/source/build.sh
- Compilar o código fonte. Exemplo:
  - Ver FW/matrix/matrix.sh.



Código Assembly/C

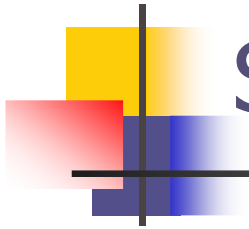
- Composto do start.s:
  - Inicializa pilha, semáforos e memória dos núcleos.
  - Chama função main()
- Código C:
  - Faz os cálculos e coloca resultados na Serial.
  - Verificar serial e preenchimento da memória.



Configurando MMC

- Editar arquivo multicore_cfg.vhd:
  - Número de núcleos
  - Configurações das caches
  - Memória Virtual
  - Tamanho/Endereçamento das memórias

Simulação – Passo a passo

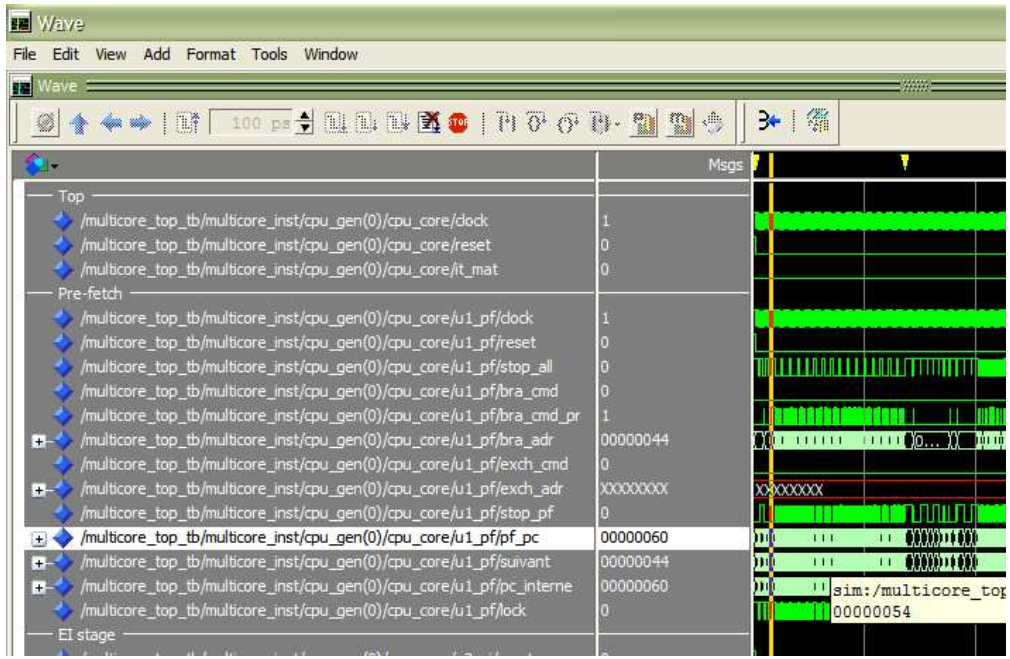


- Instalar ModelSim e compilador (binutils, gcc, etc.)
- Alterar o código C para implementar o que for necessário. Pode ser mantida a estrutura básica: sincronização, prints de debug, etc. Use o arquivo matrix.sh como base.
- Alterar o arquivo multicore_cfg.vhd, alterando CPU_NUM para o número desejado de processadores.
- Alterar o arquivo multicore_top_tb.vhd, linha 116, para que use o arquivo correto do SW gerado.
- Abrir o modelsim e na tela de transcript ir para a pasta sim, usando cd c:/<pasta de arquivos>/sim.

Simulação – Passo a passo

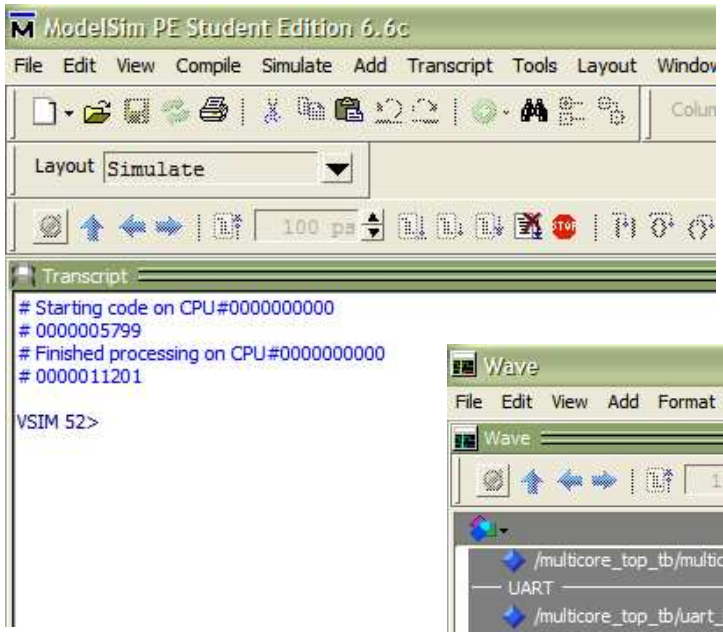
- Na mesma janela de transcript rodar a simulação usando sim_bh.do. O tempo de simulação pode ser ajustado alterando o arquivo sim_bh.do, linha 42.

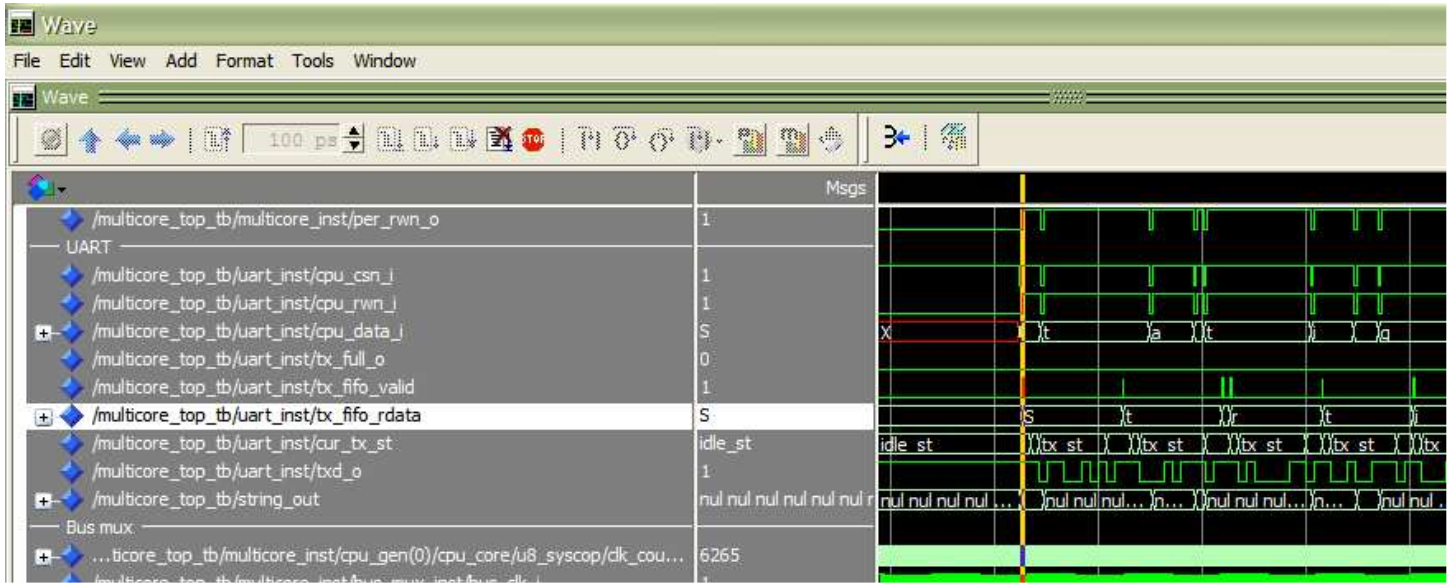
- Verificar na janela wave do modelsim o resultado da simulação:
  - Ver sinal pf_pc para acompanhar a seqüência de instruções executadas.



Simulação – Passo a passo

- Verificar na janela transcript ou wave do modelsim a saída dos prints do código C: get_time() retorna o timestamp em ciclos do processador.





Perguntas ?