Implementação e Simulação do Processador MIPS com a ALU reconfigurável dinamicamente

Iouliia Skliarova, António B. Ferrari

***Resumo*– Este artigo descreve um ambiente de simulação integrado para a arquitectura MIPS que permite ao utilizador escrever programas simples em assembly e visualizar a sua execução em termos da estrutura interna de um processador que implementa a arquitectura em causa.**

Processadores com estruturas diferentes, embora implementando o mesmo subconjunto de instruções MIPS, foram considerados. Foi feita a sua descrição em VHDL e a respectiva simulação utilizando o ambiente V-System, o que permite visualizar o funcionamento interno do processador, isto é, os sinais de controlo gerados ao longo do tempo, os resultados das operações da ALU e o conteúdo dos registos.

A implementação microprogramada do processador permite ainda a extensão do sub-conjunto de instruções suportado através da adição de novas instruções, oferecendo ao utilizador um interface amigável para a escrita de microprogramas que as implementem.

Por fim, e no contexto da computação reconfigurável, descrevem-se duas implementações alternativas de uma ALU para o processador MIPS com base em FPGAs reconfiguráveis dinamicamente da família XC6200.

***Abstract*– This article describes a set of simulation tools for processors implementing subsets of the MIPS architecture. The design environment that has been created allows the user to write simple assembly language programs and to visualize their execution in terms of the processor's internal structure, i.e. of the control signals generated with the flow of time, ALU operation results and contents of registers. Processors with different structures, although implementing the same set of instructions, have been investigated. They allow to contrast the complexity and performance of different realizations.**

A microprogrammed version of the processor allows to add new instructions by writing microprograms that implement those instructions.

Additionally, and as an experiment in the design of reconfigurable function units, two different implementations of an ALU for the MIPS datapath, have been designed using a dynamically reconfigurable FPGA of the XC6200 family.

I. INTRODUCÃO*

Os sistemas digitais podem ser descritos a vários níveis de abstracção. Cada um dos níveis permite efectuar a simulação do sistema e analisar as caraterísticas que são importantes para esse nível. Por exemplo, ao nível arquitectural [1], a simulação verifica o funcionamento do sistema sob o ponto de vista de entrada, fluxo e transformação de informação. O mais importante nesta fase é a análise de fluxos de dados, bem como o número e a sequência de passos necessários para a sua transformação (*sheduling*), as funções dos vários blocos computacionais e das outras unidades que se unem numa noção comum de "recursos" (*resource allocation*), etc. Contudo são quase completamente ignoradas as características físicas de hardware tais como atrasos de sinais, particularidades dos elementos básicos e da tecnologia integrada**.** Mas a outros níveis hierárquicos estas últimas características podem tornar-se determinantes.

Os processadores são os elementos centrais dos sistemas computacionais e de muitos sistemas de controlo. Como, definida a arquitectura, existe uma grande variedade de métodos de construção de processadores, torna-se necessário criar um ambiente que permita analisar as suas características determinantes tais como desempenho e eficácia da implementação do conjunto de instruções dado. Necessária para efectuar esta análise é a existência de modelos construídos com base em linguagens de descrição de hardware, entre as quais as mais utilizadas são VHDL e Verilog. Uma das vantagens destes modelos é a possibilidade de os analisar em qualquer computador no qual existem compiladores e simuladores de uma dessas linguagens. A análise dos diferentes modelos permite comparar as suas características básicas e tirar conclusões sobre a implementação de uma arquitectura já existente ou a criação de uma arquitectura nova.

No trabalho efectuado foi considerada a arquitectura MIPS, paradigmática da filosofia RISC e de entre estas uma das que conseguiu uma mais significativa implantação no mercado, não só de computadores como de sistemas que incorporam microprocessadores ("*embedded systems*"). Foram criados os modelos VHDL de todos os componentes básicos do processador. Foram estudadas e analisadas várias implementações do processador e diferentes variantes dos seus componentes (p.ex., da unidade de controlo). Todos os modelos foram verificados com a ajuda do analisador de VHDL - V-System.

Para uso prático dos modelos é necessário simular através deles a execução de programas em linguagem máquina do MIPS. Para este fim foi desenvolvido um assembler

* Trabalho realizado no âmbito da disciplina de Projecto

simplificado que suporta o subconjunto de instruções MIPS implementado. Este subconjunto inclui as instruções básicas aritméticas e lógicas, as de acesso à memória (*load/store*) e as instruções de salto condicional e incondicional. Por último, foi criado um ambiente que assegura a interface com os modelos elaborados, i.e. permite escolher um determinado modelo, executar neste modelo programas escritos na linguagem assembly do MIPS e analisar diferentes características. O ambiente mencionado foi desenvolvido utilizando o Visual C++ (versão 5.0).

A análise descrita acima permite avaliar várias implementações do processador com base nos seus modelos comportamentais ou estruturais. Nos últimos anos tem-se assistido a progressos muito significativos na tecnologia dos componentes lógicos programáveis, que actualmente atingem complexidades da ordem do milhão de transistores, o que torna actual a avaliação das possibilidades de implementação dos elementos principais do processador com base nessa tecnologia. Esta possibilidade traduz-se no aparecimento de um domínio novo de investigação, o da computação reconfigurável (*Reconfigurable Computing*). A parte central de um processador é, sem dúvida, a ALU. O conjunto de operações que esta é capaz de executar influi nos parâmetros de todo o processador. Torna-se interessante verificar as possibilidades da alteração dinâmica do conjunto de operações durante o funcionamento do processador o que pode ser atingido com base na tecnologia de circuitos reprogramáveis e reconfiguráveis, tais como FPGAs modificáveis dinamicamente. Este trabalho tomou como tecnologia-alvo as FPGAs reconfiguráveis dinamicamente da família XC6200 (da companhia Xilinx).

II. CONSTRUÇÃO DOS MODELOS ALTERNATIVOS DO PROCESSADOR COM ARQUITECTURA MIPS

Foram analisadas e comparadas três implementações diferentes do processador com arquitectura MIPS [2] (fig. 1).

A.1. Implementação single clock cycle

A implementação *single clock cycle* é a mais simples. Neste caso cada instrução demora um ciclo de relógio. O tempo de ciclo é determinado pela sequência de acções necessárias para a execução da instrução mais complexa. Normalmente, esta instrução é *load*. O desempenho desta implementação é baixo porque muitas das instruções poderiam ser executadas com um ciclo de relógio mais curto. O esquema estrutural da implementação *single clock cycle* do processador é apresentado na fig. 2.

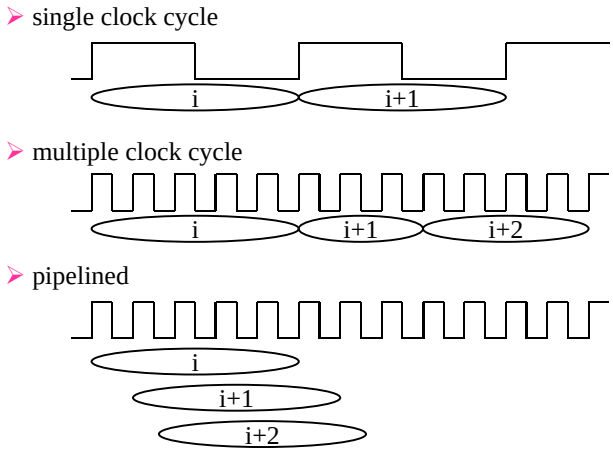

Fig. 1 - Implementações do processador MIPS (ovais correspondem a instruções)

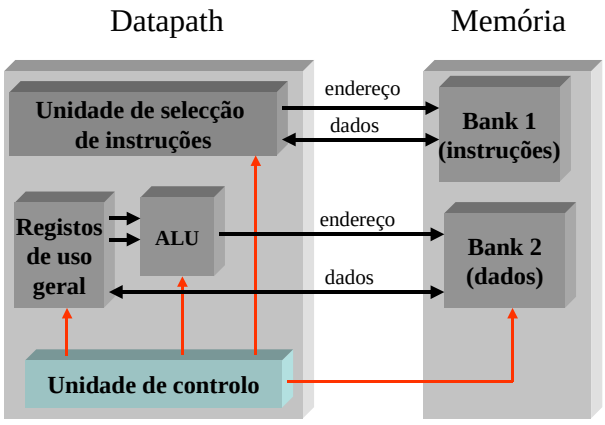

Fig. 2 - Implementação *single clock cycle*

A.2. Implementação multiple clock cycle

Na implementação *multiple clock cycle* cada instrução é dividida em vários passos cada um dos quais demorando um ciclo de relógio. Sendo assim, uma das vantagens desta implementação (em comparação com a implementação *single clock cycle*), é que comandos diferentes podem demorar um número diferente de ciclos de relógio o que significa que o tempo de ciclo pode ser menor e, consequentemente, o tempo médio por instrução diminui. A outra vantagem é que as unidades funcionais podem ser utilizadas mais do que uma vez durante a execução de uma instrução.

A implementação *multiple clock cycle* possui praticamente o mesmo esquema estrutural que a implementação *single clock cycle* (ver fig. 2). As diferenças principais que foram consideradas no modelo VHDL são algumas modificações do esquema lógico. Contudo a unidade de controlo torna-se mais complexa.

A.3. Implementação pipelined

Na implementação *pipelined* (fig. 3) do processador várias instruções podem ser executadas simultaneamente. Logo que uma instrução liberta um recurso do processador e continua a sua execução num outro recurso, o recurso libertado pode ser utilizado por qualquer outra instrução (normalmente a seguinte). Este método não diminui o tempo necessário para a execução de uma instrução individual mas aumenta sim a velocidade de execução de um conjunto de instruções, i.e. o *throughput*.

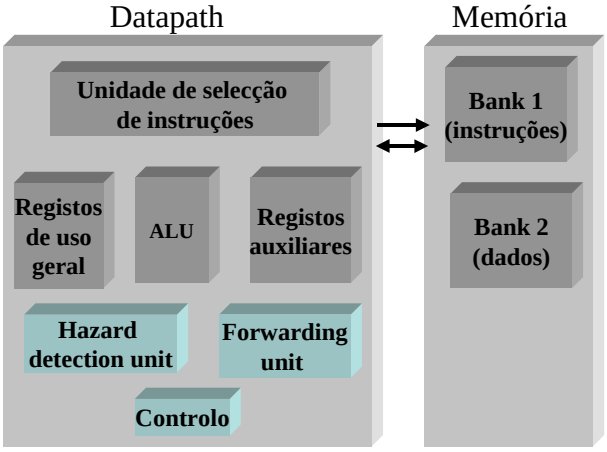

Fig. 3 - Implementação *pipelined*

B. Análise e construção da unidade de controlo

Para cada um dos modelos do processador considerados foi construída a unidade de controlo respectiva.

A unidade de controlo para as implementações *single clock cycle* e *pipelined* baseia-se numa PLA. Esta recebe o valor do campo *opcode* da instrução e gera todos os sinais de controlo necessários. O modelo matemático de PLA é um sistema de funções booleanas que se descreve trivialmente em VHDL. A PLA implementa as funções do circuito combinatório, i.e., este é um exemplo simples de uma máquina de estados finita com um só estado.

Para a implementação *multiple clock cycle* foram analisados e realizados dois métodos diferentes de especificação da unidade de controlo. O primeiro método baseia-se nas máquinas de estado finitas; o segundo usa o conceito da microprogramação.

Uma máquina de estado finita pode ser implementada com a ajuda de um registo temporário que mantém o estado corrente e de um circuito combinatório que determina todos os sinais de controlo e o estado seguinte da máquina.

Para controlar os conjuntos de instruções mais complexos, usa-se normalmente o conceito da microprogramação. Cada microinstrução define um conjunto de sinais de controlo que devem ser activados numa dada fase de execução de um comando. A unidade de controlo é baseada em memória, usualmente ROM, e inclui os respectivos circuitos de endereçagem para extracção da microinstrução adequada. Microinstruções sucessivas residem em células consecutivas da ROM, sendo o seu endereçamento efectuado através do incremento do conteúdo de um contador (*microprogram counter*). A ramificação, correspondente a um salto na execução do microprograma, é realizada através da modificação do valor do *microprogram counter*.

C. Descrição e simulação do processador

O processador MIPS foi descrito em linguagem VHDL com a ajuda do sistema EASE/VHDL da Translogic [3]. Este software permite descrever vários sistemas digitais por via da sua decomposição hierárquica. Cada componente pode ser decomposto em elementos mais detalhados (descrição estrutural) ou ser descrito directamente em linguagem VHDL (descrição comportamental). Por exemplo, o componente que efectua a descodificação das instruções, possui a organização estrutural representada na fig. 4. Contudo, no nível inferior da hierarquia cada componente deve ser obrigatoriamente descrito em linguagem VHDL. A vantagem principal deste método consiste no facto de cada componente poder ser especificado independentemente dos outros.

Obtida a descrição completa do processador em linguagem VHDL deve-se compilar e analisá-la com a ajuda do pacote V-System. Os resultados da simulação são representados pelos diagramas temporais habituais. O processo da simulação pode ser controlado pelo pacote de programas desenvolvidos em linguagem Visual C++.

D. Pacote de programas para a interacção com os modelos do processador MIPS

Como já foi mencionado o pacote foi desenvolvido em linguagem Visual C++ 5.0. Inclui um editor de texto que permite ao utilizador criar programas em linguagem assembly do MIPS. A seguir é efectuada a verificação dos erros sintácticos e semânticos e, caso o programa não tenha erros, é realizada a geração do código máquina e a sua gravação num ficheiro. Para executar um programa em linguagem assembly, o utilizador escolhe o modelo do processador pretendido; com isso é automaticamente efectuado o carregamento do analisador de VHDL V-System e estabelecidos os modos de simulação adequados.

Caso o utilizador escolha o modelo *multiple clock cycle* é oferecida a possibilidade de expandir o conjunto de comandos básico (fig. 5). Isto faz-se através da alteração do conteúdo da memória de microprograma o que permite modificar o microprograma correspondente. As alterações necessárias escrevem-se nos ficheiros especiais que são usados posteriormente pelos programas em linguagem VHDL para inicializar a ROM na qual se guarda o

microprograma. Contudo, como o modelo VHDL

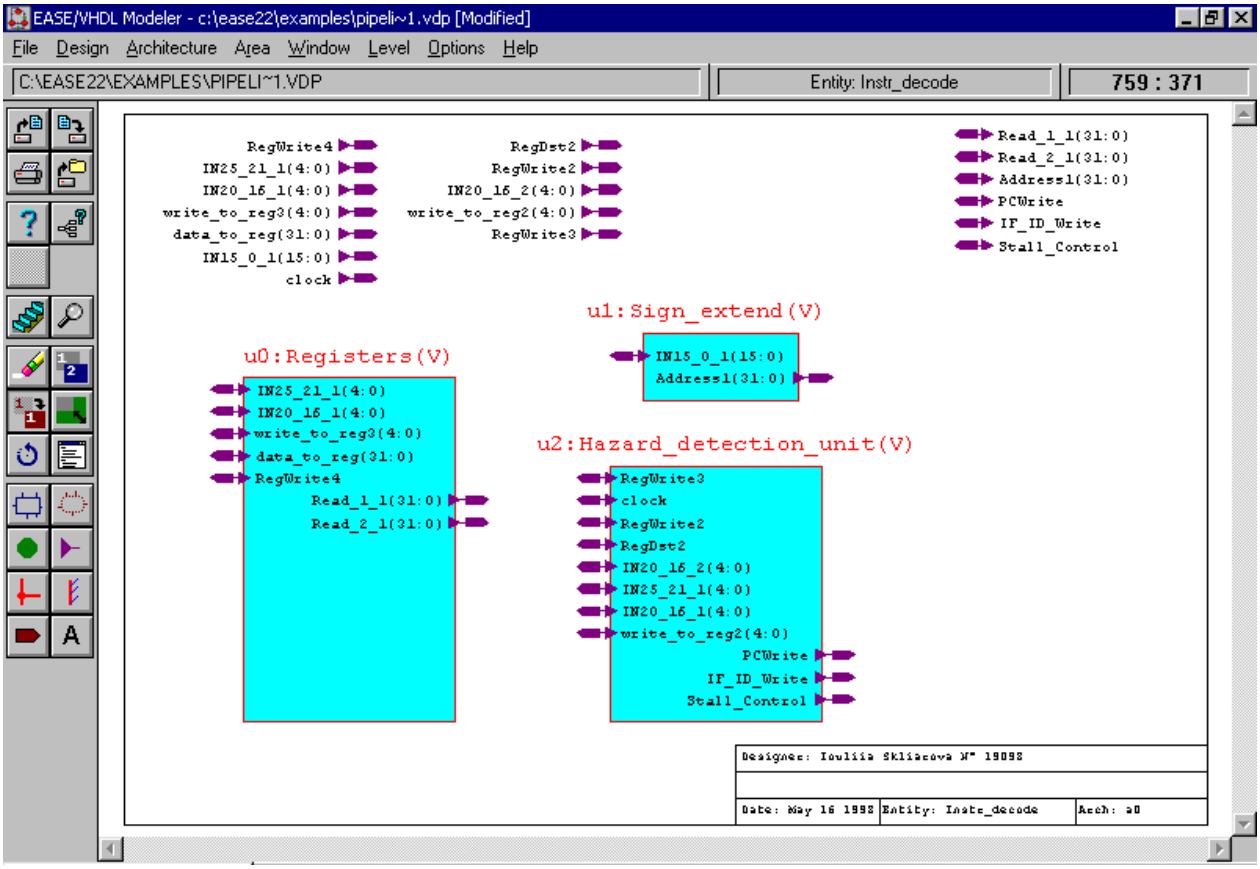

Fig. 4 – Estrutura do componente que descodifica as instruções

descreve um número limitado de microoperações e condições lógicas, as possibilidades da expansão do conjunto de microinstruções também são limitadas. O pacote inclui também o subsistema informativo e do de controlo.

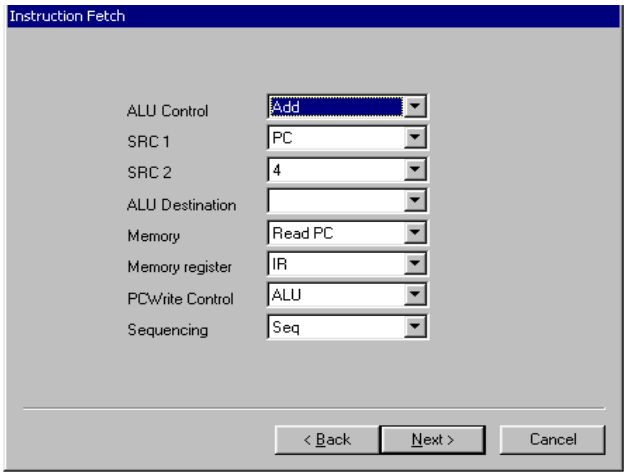

Fig. 5 - Janela de diálogo que permite especificar as instruções novas

III. ALU RECONFIGURÁVEL DINAMICAMENTE

Nesta secção apresentam-se os resultados da construção da ALU com base na FPGA XC6216, i.e. demonstram-se todas as fases a partir da descrição do esquema no editor gráfico até à obtenção do dispositivo físico. Para tal foram usados os seguintes meios de software e hardware: ViewLogic, XACT6000, IDELS e placa FireFly da companhia Annapolis.

A. FPGAs reconfiguráveis dinamicamente

As FPGAs (*Field programmable gate arrays*) da família XC6200 [4] são compostas por uma matriz de células reconfiguráveis. Cada célula contém uma unidade computacional capaz de implementar uma função lógica e uma área de encaminhamento de ligações onde se efectuamas interligações entre células. Cada célula pode ser programada individualmente de maneira a implementar qualquer função lógica de duas entradas, multiplexador 2:1, constantes 0 e 1, qualquer função de uma só variável ou qualquer destas funções mais um D-flipflop.

As FPGAs reconfiguráveis dinâmicamente, p.ex. XC6216, podem ser interligadas num sistema computacional e configuradas em *run-time* de forma a implementar uma função específica. Existe ainda a possibilidade da reprogramação parcial destas FPGAs sem suspensão da operação de outras partes que não necessitam de ser reconfiguradas. Como se pode substituir uma parte das funções anteriormente implementadas por novas funções, torna-se possível a utilização do mesmo hardware para a implementação de várias funcionalidades [5].

Tempo e custo da programação de FPGA são bastante pequenos portanto estas convêm idealmente para a criação de hardware específico que normalmente tem um mercado de dimensão reduzida.

B. Sistema de desenvolvimento de hardware ViewLogic

O *ViewLogic* [6] é um dos ambientes para desenvolvimento de hardware mais conhecidos, e de que existe versão para computadores pessoais. Suporta a utilização de bibliotecas dos principais fabricantes de circuitos integrados, FPGAs e PLDs (Xilinx, Altera, Lattice, Motorola, Quicklogic, etc.). O sistema permite resolver os seguintes problemas básicos:

- Descrição gráfica (esquemáticos) do circuito a vários níveis hierárquicos.
- Descrição mista do circuito. Neste caso uma parte pode ser especificada graficamente e outra descrita com a ajuda das linguagens VHDL ou Verilog.
- Simulação do sistema descrito só graficamente, só em linguagens VHDL, Verilog ou usando ambos os métodos.
- Optimização arquitectural e síntese lógica dos sistemas digitais baseados em FPGAs e PLDs.
- Transformação do circuito para uma determinada tecnologia, i.e. a sua representação por componentes interligados tirados das bibliotecas dos fabricantes de circuitos integrados e das próprias bibliotecas do utilizador.

O ViewLogic suporta também o desenho de circuitos impressos, o desenvolvimento de sistemas analógicos, a análise dos parâmetros temporais de circuitos, etc.

C. XACT6000

O programa XACT6000 [7] efectua o mapeamento, a colocação e a interligação das FPGAs da família XC6200. O mapeamento pressupõe o agrupamento dos elementos do esquemático em blocos que vão ficar numa determinada zona da FPGA. Por ex., um somador de um bit pode ser construído com três células da FPGA dispostas numa linha vertical, numa linha horizontal ou num triângulo. Por sua vez, um somador de mais bits pode-se construir de várias maneiras dos somadores de um só bit, etc. Na fase de roteamento determinam-se as ligações entre várias células do circuito. Estas ligações definem-se com a ajuda de um conjunto de multiplexadores que são controlados por SRAM. O conteúdo desta memória é actualizado através da transferência da memória externa durante o processo da configuração da FPGA.

Os processos de mapeamento e de colocação podem ser controlados com a ajuda dos atributos definidos no esquema (atributos RLOC, REGBIT, e REGNAME). Os processos de mapeamento, colocação e roteamento podem ser executados automaticamente mas nem sempre se pode obter desta maneira um resultado positivo, i.e. em muitos casos algumas ligações não são encaminhadas. A experiência de trabalho com o programa XACT6000 mostra que para qualquer circuito não trivial, é quase sempre necessária uma intervenção externa. Contudo, este programa dispõe de meios que simplificam a detecção e a reparação das ligações suspensas.

D. IDELS

IDELS [8,9] permite verificar em modo interactivo o funcionamento do circuito construído. O *debbuger* deste sistema permite formar dinamicamente todos os recursos de trabalho e modificá-los no processo de análise do circuito. O estado de qualquer uma das células da FPGA pode ser lido e o estado de muitas células pode ser alterado. As células são os componentes estruturais elementares do circuito pretendido e delas é que são construídas outras suas partes compostas do nível superior. IDELS permite resolver muitos problemas adicionais, tais como a especificação de vários modos de sincronização, etc.

E. Placa FireFly

Para as experiências foi utilizada a placa FireFly da campanhia Annapolis [10]. A placa contém uma FPGA XC6216 reconfigurável dinamicamente. Para organizar a interface com o barramento do computador pessoal utiliza uma FPGA XC4013E. A placa contém 4 *chips* de memória estática organizados em dois bancos de memória cada um de 256KB. O acesso à memória pode ser efectuado quer do lado da FPGA quer do lado do barramento do computador pessoal. A placa contém conectores do tipo *Mezzanine* que são ligados a todos os pinos da FPGA e dos outros componentes da placa, tais como gerador de impulsos, *reset*, etc. Isto permite verificar o funcionamento da FPGA em *run-time* com a ajuda de um analisador lógico multicanal ligado a estes conectores.

F. Construção da ALU

Na fig. 6 é apresentado o dispositivo pretendido que se compõe do esquema lógico da ALU, de dois registos de entrada de 32 bits, de um registo de saída e de um registo que define o código da operação. Esta ALU é bastante simples e permite executar as operações lógicas e as operações aritméticas de adição, subtracção e comparação sobre operandos de 32 bits.

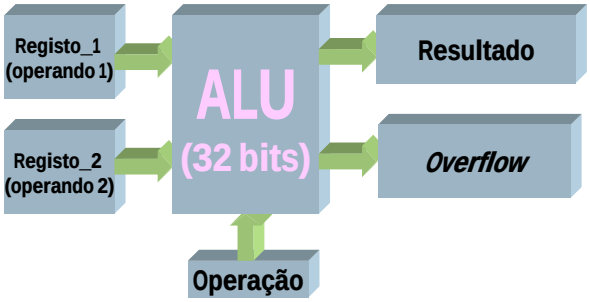

Fig. 6 - ALU de 32 bits

Foram analisadas duas variantes da implementação da ALU, a primeira construída como um elemento estático e a segunda em que é possível modificar o conjunto de operações executáveis durante o funcionamento [9,11].

Primeiramente, o esquema da ALU foi descrito hierarquicamente no editor gráfico ViewDraw que faz parte do sistema ViewLogic [6] (fig. 7). Os blocos básicos são descritos ao nível de *gates* dos elementos da biblioteca XC6000.

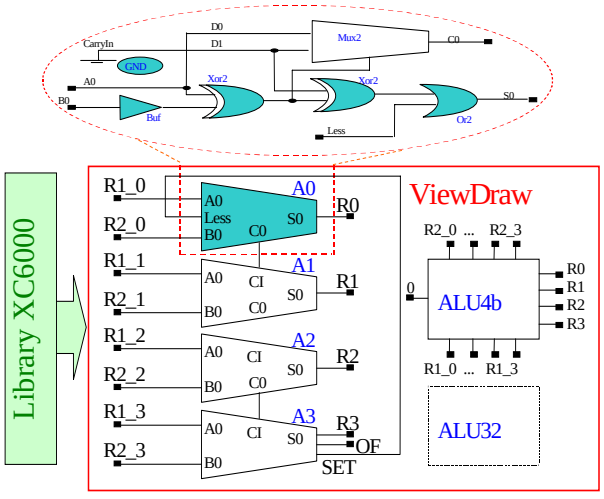

Fig. 7 - Descrição hierárquica da ALU de 32

A ALU pretendida de 32 bits decompõe-se primeiramente em duas sub-ALUs de 16 bits e a seguir em oito sub-ALUs de 4 bits que por sua vez são constituídas por unidades computacionais primárias. A decomposição deste género permite-nos simplificar significativamente os processos de mapeamento, colocação e roteamento executados pelo XACT6000. A ajuda ao XACT6000 é efectuada através de especificação no ambiente ViewLogic de vários atributos topológicos que definem tais parâmetros como configurações das subALUs, suas posições na superfície da FPGA, etc. As unidades computacionais primárias (i.e. ALUs de um bit) foram construídas com base em elementos da biblioteca XC6000. Os *gates* sombreados (ver fig. 7) podem ser reconfigurados em *run-time* o que permite efectuar a modificação dinâmica do conjunto de operações. Na fig. 8 é demonstrado de que maneira o esquema lógico do bit menos significativo da ALU tem que ser reconfigurado para implementar as operações de adição, subtracção e comparação. Note-se que o bit menos significativo, o mais significativo e os outros bits possuem esquemas um pouco diferentes e consequentemente têm que ser configurados de maneira diferente.

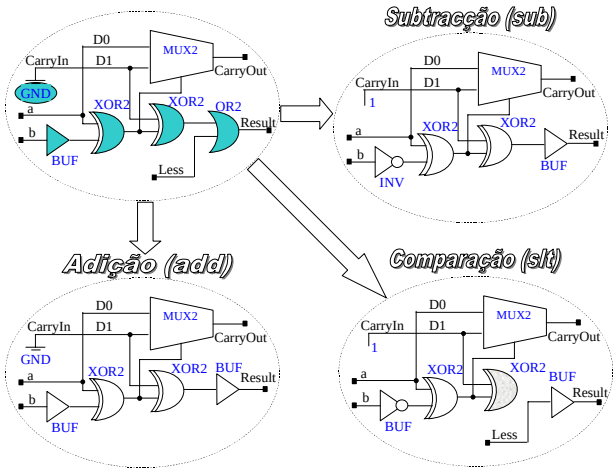

Fig. 8 - Reconfiguração do bit menos significativo da ALU para implementar as operações de adição, subtracção e comparação

A seguir, para o esquema especificado com a ajuda do programa ViewVSM forma-se a lista de ligações (NetList) que é transformada posteriormente para o ficheiro do formato EDIF. Este ficheiro carrega-se no programa XACT6000 [7] que efectua o mapeamento, a colocação e o roteamento da FPGA (fig. 9).

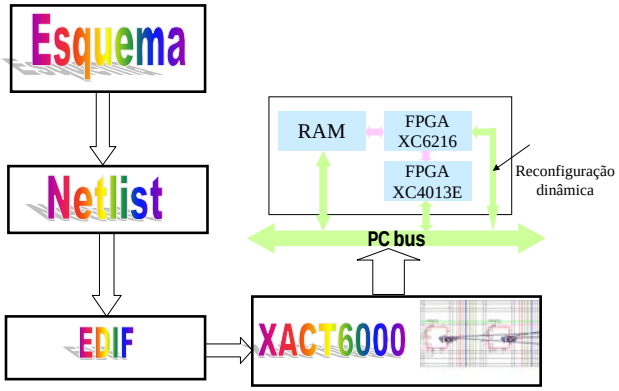

Fig. 9 - Sequência de acções necessárias para a transformação do esquema em hardware

Na fig. 10 são apresentados os resultados produzidos pelo XACT6000 para a FPGA XC6216.

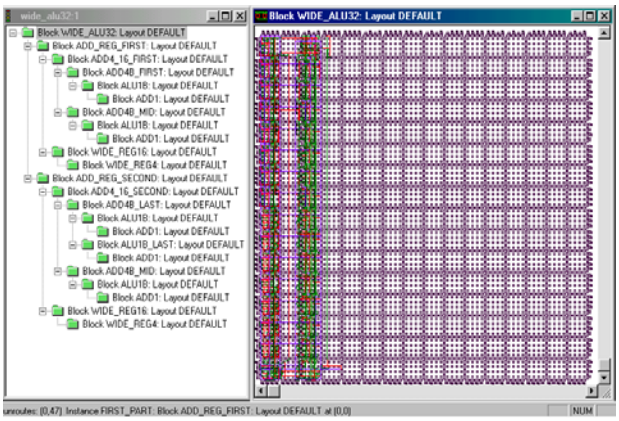

Fig. 10 - Implementação da ALU na FPGA XC6216

A seguir XACT6000 forma os ficheiros que podem ser carregados na FPGA. No caso geral, estes incluem um ficheiro da configuração da SRAM (*.cal), um ficheiro da colocação de vários elementos do esquema na superfície da FPGA (*.sym) e um ficheiro para a reconfiguração dinâmica (*.ral). O último ficheiro não é obrigatório e é gerado só caso seja indicado.

Estes ficheiros são carregados na FPGA com a ajuda do sistema IDELS [8]. A estrutura da ALU é representada no IDELS em forma de árvore (fig. 11). A ALU forma a raiz da árvore. O componente ALU inclui outros componentes, p.ex. – o registo que define o código da operação (*OPERATION*). Este por sua vez consiste nos três flipflops protegidos. Sendo assim, a raiz da árvore representa o dispositivo e as folhas representam os elementos que podem ser implementados directamente em células da FPGA. O programa IDELS visualiza as estruturas apresentadas na fig. 11 e permite reorganizá-las segundo o nível de abstracção necessário.

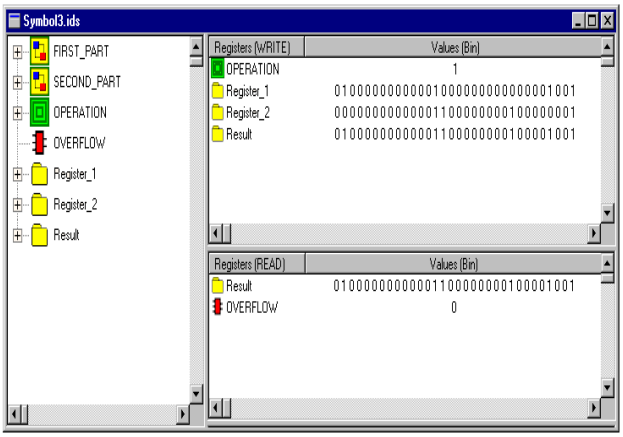

Fig. 11 - Representação hierárquica da estrutura da ALU e execução da operação *ou lógico* na FPGA XC6216

G. Resultados

A FPGA XC6216 contém 4096 células. A ALU de 32 bits implementada com a configuração estática ocupou 385 células e a com a configuração dinâmica - 162 células, i.e. 223 células menos. Portanto as possibilidades da reconfiguração dinâmica permitiram-nos diminuir significativamente a quantidade de hardware necessária. Além disso é possível expandir facilmente o conjunto de operações executáveis pela ALU.

IV. CONCLUSÕES

Neste artigo apresentou-se um sistema que permite examinar e avaliar três implementações alternativas do subconjunto básico da arquitectura MIPS. O sistema inclui os modelos VHDL do processador e um ambiente desenvolvido em Visual C++ que suporta a interacção com os mesmos. O conjunto de facilidades fornecido torna o sistema um poderoso veículo didático para a compreensão do funcionamento interno dos processadores.

Apresenta-se também um exemplo prático da aplicação das possibilidades da reconfiguração dinâmica de hardware. Para tal consideramos duas implementações diferentes da ALU com base nas FPGAs da família XC6200 (Xilinx) reconfiguráveis dinamicamente. A primeira possui estrutura estática enquanto a segunda pode ser modificada em *run-time*. Verificou-se que a implementação dinâmica da ALU requer menos recursos da FPGA em comparação com a implementação estática. Além disso o número de operações executáveis pode ser facilmente expandido.

REFERÊNCIAS

[1] Giovanni De Micheli, Synthesis and Optimization of Digital Circuits, McGraw-Hill, Inc., 1994.

[2] Patterson, D.A., Hennesy, J.L., "Computer Organization and Design. The Hardware/Software Interface", Morgan Kaufmann Publishers, Inc., 1998.

[3] Ease/VHDL Version 2.2. User's Guide, Translogic BV, 1994.

[4] Xilinx, "XC6200 Field Programmable Gate Arrays", Product Description (http://www.xilinx.com/partinfo/6200.pdf), 1997.

[5] Skliarov V., Melo A., Oliveira A., Lau N., Monteiro R., Circuitos Virtuais Baseados em Reprogramação e Reconfiguração Dinâmica, Electrónica e Telecomunicações, 01.1998, pp. 248-260

[6] WorkView Office. User's Guide, Viewlogic Systems, Inc., 1997.

[7] Series 6000 User Guide, Xilinx, 1997.

[8] Skliarov, V., Lau, N., Oliveira A., Melo A., Kondratjuk K., Ferrari A., Monteiro R., Skliarova I., Synthesis Tools and Design Environment for Dynamically Reconfigurable FPGAs, International conference on Circuits and Systems, Brazil, 1998.

[9] Skliarov V., Lau N., Oliveira A., Melo A., Kondratjuk K., Ferrari A., Monteiro R., Skliarova I., "Design Tools for Dynamically Reconfigurable FPGAs", PACT 98, Workshop on Reconfigurable Computing, Paris, França, 1998, pp. 60-65.

[10] Xilinx, XC6200DS, Board Revision: 1, FPGA Revision: Version 1, 25.12.1997.

[11] Skliarova I., Ferrari A., " VHDL Models and Integrated Environment for Analyzing Alternative Implementations of Processors with MIPS Architecture", Proceeding of the 8th BELSIGN Workshop, Madrid, Espanha, 1998.