A PLATAFORMA EDgAR NA PROTOTIPAGEM RÁPIDA DE SISTEMAS DIGITAIS

António J Esteves João M Fernandes Henrique D Santos Alberto J Proença
Dep. Informática, Universidade do Minho, 4710 Braga, Portugal
Email: {esteves,miguel,hsantos,aproenca}@di.uminho.pt

Sumário

A metodologia de co-projecto está vocacionada para o desenvolvimento de sistemas de média e grande complexidade. A prototipagem rápida de sistemas digitais é um complemento da simulação, indispensável em certos sistemas de grande complexidade. Esta comunicação apresenta a plataforma EDgAR que permite obter rapidamente protótipos de sistemas digitais de elevada complexidade, recorrendo a componentes lógicos reprogramáveis (MACHs e FPGAs) e a uma arquitectura paralela baseada em *transputers*. Esta abordagem faz baixar o custo de produção de sistemas, reduzindo o tempo de projecto e de teste dos protótipos. Apresentam-se as ferramentas de suporte do EDgAR, as quais foram concebidas para especificar sistemas com elevado nível de abstracção, numa linguagem normalizada de larga aceitação e tendo em vista permitir o máximo de automatização no processo de síntese. Esta plataforma é validada na emulação dum circuito integrado para processamento de imagem.

Palavras Chave: co-projecto, prototipagem rápida, PLDs, *transputer*.

1 INTRODUÇÃO

Os componentes reprogramáveis (FPLDs[1]) registaram um forte desenvolvimento nos últimos anos, o que se traduziu numa importância crescente da disciplina da prototipagem rápida de sistemas, e posteriormente no surgir duma área de investigação associada: o co-projecto[2] de *software* e *hardware* (de aqui em diante apenas referido como co-projecto).

Os FPLDs tornaram-se atractivos para criar protótipos de um modo rápido e eficiente, pois a sua complexidade atingiu, no caso das FPGAs, o equivalente a dezenas de milhar de portas lógicas, e os fabricantes e empresas associadas disponibilizam sofisticadas ferramentas de CAD electrónico, para apoiar a utilização desses componentes. A redução do tempo de projecto e do custo de produção, bem como o facto de os FPLDs não necessitarem de ser removidos para programação, são outros motivos a favor da sua utilização em plataformas de apoio às áreas de investigação citadas.

O co-projecto está intimamente ligado ao projecto de sistemas com requisitos de desempenho impossíveis de obter com implementações exclusivamente em *software*, e com complexidade superior aos sistemas implementáveis exclusivamente em *hardware* (ASICs) [1, 2].

Um número razoável das plataformas existentes apresenta o inconveniente de ter especificidades que as tornam pouco atractivas para o desenvolvimento de sistemas digitais de elevada complexidade, como é o caso dos sistemas para visão por computador. Além de complexos, quando usados em aplicações de tempo real, estes sistemas possuem requisitos temporais críticos. A plataforma EDgAR[3] surge no contexto da implementação deste tipo de sistemas, não ficando contudo amarrada às suas especificidades.

Esta comunicação apresenta a plataforma EDgAR que permite obter rapidamente protótipos de sistemas digitais de elevada complexidade, recorrendo a uma arquitectura paralela baseada em *transputers*. A plataforma é um compromisso entre uma versão simulada do produto pretendido e um protótipo com tecnologia idêntica ao produto final de custo superior, pois possibilita a integração destes dois modelos a custos reduzidos e com capacidade

[1] *Field Programmable Logic Devices*
[2] Do inglês *codesign*
[3] Emulador Digital Altamente Reprogramável

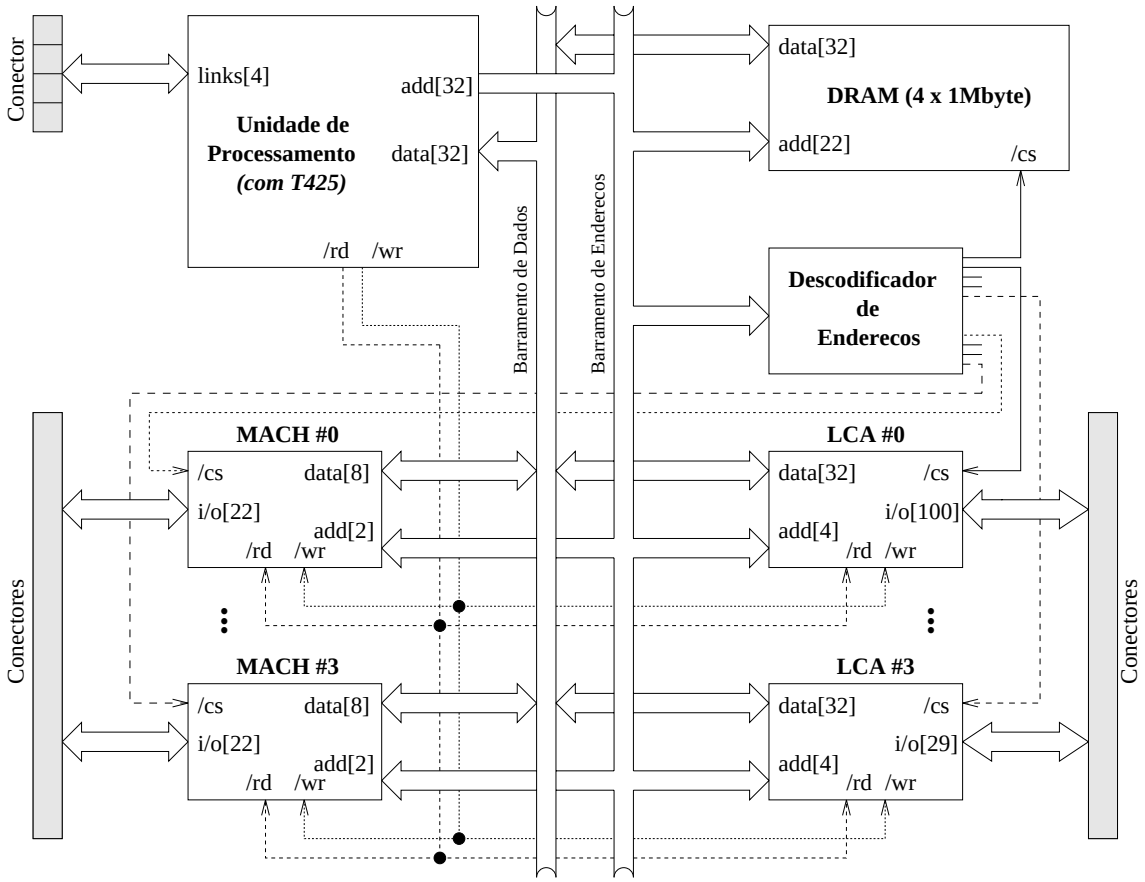

Figura 1: A arquitectura da plataforma EDgAR.

de reconfiguração exclusivamente por programação.

A exposição foi dividida em três secções. Na secção 2, é apresentada a arquitectura da plataforma EDgAR, enquanto a síntese de sistemas digitais com o EDgAR é analisada na secção 3, abordando-se as várias fases do processo: especificação do sistema a prototipificar, partição em componentes de *software* e de *hardware*, alocação dos recursos da plataforma a esses componentes e validação do protótipo obtido. Na secção 4 descreve-se a emulação de um circuito VLSI com o EDgAR.

2 ARQUITECTURA DA PLATAFORMA EDgAR

A estrutura da plataforma EDgAR (figura 1) apoia-se em dois grandes blocos:

i) uma unidade de processamento digital de informação (UPDI), implementando um nó de computação paralela, com capacidades de processamento escalar e de comunicações, e onde a velocidade de processamento dos sinais digitais não é um factor determinante;

ii) uma unidade de lógica programável (ULP), possuidora de elevado número de recursos digitais reconfiguráveis e cuja velocidade de operação se aproxima dos circuitos comerciais com tecnologias rápidas, permitindo desempenhos bem superiores aos simuladores tradicionais.

Na realização da UPDI seleccionou-se um microprocessador com capacidade de processamento escalar e de comunicações, o *transputer*, o qual suporta ainda paralelismo escalável, devido à sua capacidade de interligação a outros microprocessadores idênticos, formando uma rede com uma topologia variável. Este processador é ainda responsável pela interface com o sistema de desenvolvimento do protótipo, e pela configuração e/ou programação inicial dos componentes da ULP [3]. Na fase de depuração, a interface do utilizador com a plataforma foi desenvolvida através duma unidade com várias TRAMs[4] instalada num PC e das ferramentas do compilador ANSI C da Inmos [4]. A ligação entre a unidade de TRAMs e o EDgAR é feita através de um (ou mais) *links*, os quais são assíncronos por natureza. As ferramentas disponíveis para as TRAMs, permitem monitorizar os *transputers* das TRAMs e do EDgAR, compilar os programas e carregar estes nos *transputers*.

[4] *TRAnsputer Module*

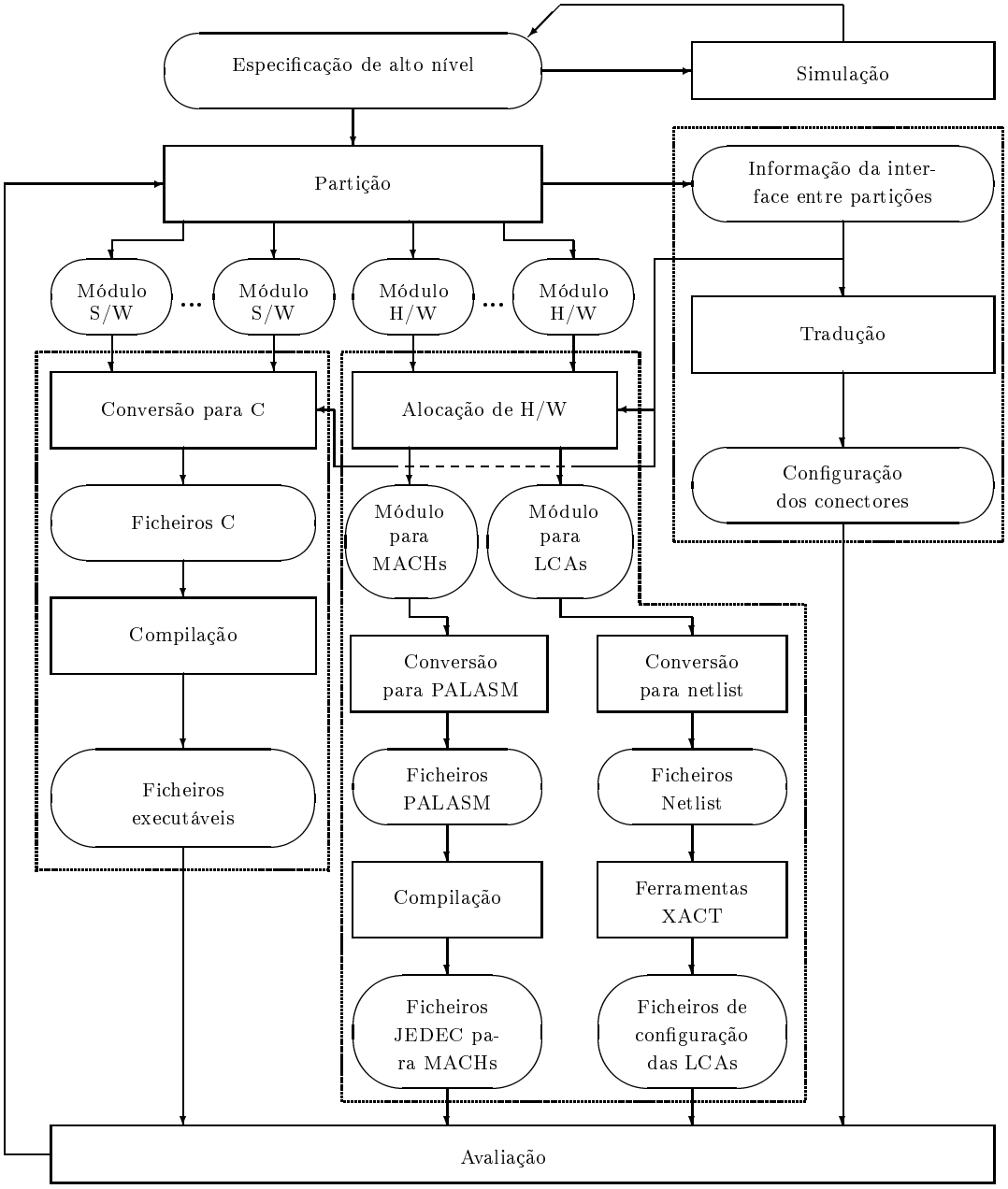

Figura 2: Metodologia de desenvolvimento de sistemas com a plataforma EDgAR.

A ULP disponibiliza um elevado número de recursos, não comprometendo significativamente a velocidade de operação dos sistemas implementados. A estrutura base da ULP assenta em dois tipos de PLDs: uns são vocacionados para a implementação de circuitos com lógica predominantemente a dois níveis (MACHs[5] [5]), os outros possuem uma estrutura matricial que lhe permite implementar circuitos com lógica multi-nível (FPGAs[6] [6]).

O protótipo da plataforma EDgAR é implementado com um *transputer* T425 (ou um T805), com 4Mbytes de DRAM, e ainda com 4 MACHs e 4 FPGAs. As MACHs usadas são da família 2x0 da AMD (com 44 pinos, 64 macrocélulas e 32 células de I/O). As FPGAs usadas são LCAs da família 30xx da Xilinx (duas com 84 pinos e duas com 175, 320 macrocélulas, 64 e 139 células de I/O).

Os vários componentes ligam-se num barramento comum, usando-se espaços de endereçamento separados para as memórias interna e externa do *transputer* e para cada uma das MACHs e LCAs. Para a plataforma poder emular sistemas digitais distintos, e com capacidade de reconfiguração por programação, cada MACH liga-se aos barramentos através de 2 linhas de endereço e 8 linhas de dados, enquanto cada LCA se liga aos barramentos por 4 linhas de endereço e 32 linhas de dados. Os restantes pinos de entrada/saída nas MACHs e nas LCAs são ligados a conectores, permitindo assim a emulação de sistemas com diferente número de sinais de entrada/saída. Para permitir escalar a capacidade de processamento do sistema, disponibilizam-se para o exterior as linhas de comunicação (*links*) do *transputer*. A figura 1 mostra como a utilização dum *transputer* resultou numa plataforma EDgAR com uma arquitectura muito simples, dado a interface com este processador ser simples.

3 A SÍNTESE DE SISTEMAS DIGITAIS COM O EDgAR

A síntese de sistemas digitais com a presente plataforma evolui normalmente por diversas fases, desde a especificação à implementação, passando pela simulação e teste. No que se segue, descreve-se como estes componentes estão a ser incorporados no ambiente em desenvolvimento que servirá de suporte ao EDgAR, podendo acompanhar-se o processo de síntese através da figura 2.

Especificação
Na fase actual de evolução da plataforma, em que o contexto de utilização é o co-projecto, está em estudo a selecção e desenvolvimento de um ambiente de alto nível para especificar os sistemas, que será a base do modelo de especificação a seguir. As hipóteses em análise incluem uma representação baseada em FSMs[7], um sub-conjunto do Occam, uma representação com Redes de Petri ou um sub-conjunto do VHDL. Uma representação formal de alto nível é essencial para provar a correcção da especificação e garantir que ela se mantém nas fases seguintes do projecto.

Em desfavor da modelação de sistemas com FSMs podem ser usados dois argumentos: (i) as FSMs não são uma notacção de nível suficientemente alto, o que contraria o que acaba de ser referido; (ii) as FSMs não são adequadas para representar sistemas de elevada complexidade algorítmica, como interessaria no caso presente [7].

Em favor da linguagem Occam pode mencionar-se a sua simplicidade, o facto de possuir requisitos para tempo-real mínimos, apresentar potencialidades para paralelismo, possuir uma semântica (baseada no CSP [8]) bem definida e ser adequada para representar componentes de sistemas a implementar no *transputer*. Como factores desfavoráveis pode referir-se o facto de não ser uma linguagem de larga aceitação (o que se traduz em que há poucas ferramentas para usar na síntese dos sistemas) e estar muito ligada aos *transputers*, ou seja, não é independente da implementação, como seria desejável [1].

As Redes de Petri (RdP) são um formalismo gráfico para a modelação de sistemas que exibem actividades assíncronas ou concorrentes e a sua representação gráfica permite a sua utilização para animar os sistemas modelados. Os formalismos matemáticos adstritos às RdP possibilitam validar os sistemas em relação a uma série de propriedades (determinismo, inexistência de bloqueios, conflitos, desperdício e código morto) [9].

VHDL é uma linguagem normalizada de descrição de *hardware* e é usada no projecto de sistemas digitais, permitindo a sua concepção com uma notação de especificação clara, bem como a modelação, simulação e síntese de soluções. A funcionalidade dos sistemas assim projectados pode ser representada compacta, hierárquica e adequadamente [10, 11].

A junção de VHDL com Redes de Petri é apresentada como uma solução viável e foi objecto de investigação prévia [12], tendo sido usada com êxito na especificação de unidades de controlo paralelas. Uma análise idêntica será realizada na plataforma EDgAR, para implementar sistemas mais complexos do que os testados até à data.

[5] *Macro Array CMOS High-density*
[6] *Field Programmable Gate Arrays*
[7] *Finite State Machines*

Na concepção do modelo deve ter-se em conta que a implementação de sistemas com a plataforma EdgAR é assíncrona, pois um modelo de especificação completamente síncrono representará mais dificilmente os aspectos associados com a implementação de sistemas que têm partes implementadas em *software* e em *hardware*, os quais têm uma natureza assíncrona. Embora seja desejável uma especificação independente da implementação, isto raramente é conseguido na globalidade.

Partição em Componentes de *Hardware* e *Software*
No contexto de co-projecto, a fase mais complexa da síntese dum sistema é a sua partição em componentes a implementar em *software* e em *hardware*. Esta tarefa dificilmente será feita na globalidade de forma automática, dado que normalmente é necessário fornecer ao algoritmo de partição entradas que ajudem nesse processo. Na plataforma EDgAR, as partições de *hardware* são implementadas através da ULP e as de *software* com a UPDI. De entre as partições de *hardware* há que distinguir entre as que são implementadas nas MACHs das que são implementadas nas FPGAs. O ponto de partida para a partição consiste no isolamento das partes com requisitos apertados em termos temporais, que darão origem a partições de *hardware*, podendo as restantes resultar em partições de *software*. Um ponto fundamental associado com as partições é a definição e implementação de estratégias de comunicação e interface entre partições, do mesmo tipo ou de tipos diferentes. Dado que a ULP não controla a memória DRAM, apenas o *transputer* o faz, o meio físico de comunicação entre partições terá a intervenção do *transputer*.

Alocação e Programação de Componentes
O processo de alocação dos recursos da UPDI às partições a implementar em *software* será feito em duas etapas. Numa primeira fase, converte-se a especificação de alto nível dessas partições para módulos descritos numa linguagem intermédia (C), o que requer a escrita dum conversor da linguagem de especificação dos sistemas para C. Os módulos anteriores são depois compilados para código máquina do *transputer*. Nesta fase, usa-se um compilador de C disponível. A alocação dos recursos da ULP às partições de *hardware* do sistema começa pela atribuição a essas partições de recursos disponíveis nos dois tipos de PLDs (MACHs e LCAs) e é condicionada pela necessidade de mais ou menos elementos de armazenamento e pelos apertados critérios temporais na implementação.

Na programação de MACHs, a compilação e o posterior mapeamento da especificação aos seus recursos é feita de acordo com a alocação de componentes, gerando um ficheiro de configuração em formato JEDEC. Com base na alocação de componentes das LCAs, geram-se ficheiros em formato intermédio (*netlist*) que servirão de entrada às ferramentas da Xilinx: XACT[8]. Estas ferramentas geram ficheiros de configuração binários, que definem a operação das LCAs, passando antes pelas fases de mapeamento, colocação e encaminhamento.

Quando se liga a alimentação do sistema, estabelece-se a configuração interna de cada LCA, por uma acção de escrita efectuada na LCA pelo *transputer*. Dos vários modos de configuração possíveis, optou-se pelo modo de periférico. A reprogramação das LCAs é também possível por programação sem efectuar uma inicialização forçada do sistema.

Verificação de Componentes
Na verificação da parte do sistema implementada com LCAs, o XACT permite dois tipos de simulação: funcional e temporal. A simulação funcional permite detectar possíveis erros lógicos no sistema, enquanto que a simulação temporal possibilita o teste da funcionalidade em situações diferentes, como por exemplo uma temperatura mais elevada, uma tensão de alimentação mais baixa ou um processo mais lento [13].

Para validar o protótipo obtido, a um nível de abstracção superior, usa-se o processo de simulação conjunta (co-simulação). A co-simulação é uma tarefa demorada e exigente do ponto de vista computacional, pelo que tem de se usar um modelo de simulação adaptado a arquitecturas paralelas [14].

4 EMULAÇÃO DUM CIRCUITO VLSI COM O EDgAR

A emulação dum *Associative Processor Array* (APA), projectado para um circuito VLSI a usar em processamento de imagem e constituído por 64 processadores, GLiTCH[9], foi usado para validar a estrutura física da plataforma EDgAR [15]. A estrutura do GLiTCH é organizada em 5 blocos: um *array* de 64 processadores de 1 bit, tendo cada 68 bits de memória associativa, um encaminhador de padrões, um *video shift register* (VSR) de 64x8 bits e um descodificador de instruções [16].

Durante a validação da plataforma EDgAR, utilizaram-se as ferramentas OrCAD e VIEWlogic para especificar os blocos do sistema a implementar com as componentes de *hardware* da plataforma, concretamente as MACHs e FPGAs. Para a especificação dos blocos a implementar em *software*, no *transputer*, utilizou-se uma especificação em lin-

[8] *Xilinx Automatic CAE Tools*
[9] *Goes Like The Clappers, Hopefully*

guagem C. O OrCAD permite especificar PLDs esquematicamente ou numa de entre várias formas textuais [17] [18]. Para especificação de PLDs, o ViewPLD da VIEWlogic aceita ficheiros em formato standard JEDEC, descrições textuais nas linguagens de descrição de *hardware* ABEL ou VHDL e esquemáticos [19] [20].

A forma de partição do GLiTCH pelas unidades do EDgAR foi determinada pelo aspecto seguinte: a parte do APA que está dependente de sinais de relógio externos ao sistema (video) foi implementada com a ULP; as restantes foram implementadas no *transputer*, dado que é possível usar o EDgAR para reconfigurar o relógio externo. Na opção tomada, o VSR e os circuitos tampão dos pinos de entrada/saída foram implementados com a ULP, enquanto que os restantes blocos do GLiTCH foram implementados no *transputer*. Na alocação de recursos da ULP, implementou-se o VSR numa LCA e os circuitos tampão dos pinos de entrada/saída noutra LCA [3]. Apresenta-se aqui, apenas um resumo da forma de implemetação.

O VSR é um registo de deslocamento bidimensional, ou seja, organizado em matriz. Com um barramento de video de 8 bits e com 64 PEs num GLiTCH, o VSR tem a dimensão de 64x8. O acesso, para escrita ou leitura, ao VSR pode ser efectuado por colunas de 8 bits, ou por linhas de 64 bits. Na implementação do VSR efectuada com o EDgAR, o acesso por linhas é feito com meias linhas de 32 bits.

A funcionalidade do VSR traduz-se nas operações que realiza sobre os bits nele guardados. As duas operações possíveis designam-se por *SHIFT* e *SWAP*, e correspondem a fazer o deslocamento de colunas e a rotação de linhas. A operação *SHIFT* efectua-se à frequência de um relógio externo, onde os bits DATA[0-7] do barramento de dados são registados na coluna 0 do VSR, enquanto que todas as colunas são deslocadas uma posição à direita. A coluna 63, a mais à direita, é enviada para as mesmas linhas do barramento de dados. A operação *SWAP* manipula uma linha de 64 bits, mas como no EDgAR a ligação das LCAs ao barramento de dados é de 32 bits (DATA[0-31]), têm que se efectuar duas operações por linha. Para aceder às duas metades de cada linha usam-se dois endereços. Esta operação implica fazer uma operação parcelar de leitura da linha 0 (*parallel read*), seguida duma operação parcelar de escrita na linha 7, acompanhada duma rotação interna de todas as linhas (*parallel write/column rotate*). A operação de *SWAP* permite implementar três instruções do GLiTCH, designadas por *rotate*, *extract* e *restore*.

A característica do VSR mais crítica para a implementação foi o elevado número de elementos de armazenamento necessário (8 * 64 = 512). Como o VSR junta dois factores que contribuem para uma implementação difícil, uma taxa de alocação de elementos de armazenamento elevada e uma taxa de utilização de IOBs[10] também elevada, implementou-se o VSR numa LCA com 175 pinos, surgindo alguns problemas: encaminhamento automático da LCA incompleto, atrasos acumulados elevados e ocorrência de sinais que alimentam muitas portas lógicas. Alguns destes problemas seriam atenuados, ou até eliminados, se o VSR fosse implementado em 2 LCAs; contudo, dai resultaria uma comunicação entre partições com custo superior, e além disso esta opção teve a vantagem de testar a utilização das LCAs numa situação limite.

O aspecto mais crítico para a implementação dos circuitos tampão dos pinos de entrada/saída foi o número de pinos e correspondentemente o número de blocos de entrada/saída a alocar na LCA. Como o número de IOBs necessário (124) é superior aos 64 IOBs existentes numa LCA de 84 pinos, mas é inferior aos 139 disponíveis numa LCA de 175 pinos, optou-se pela implementação com uma única LCA de 175 pinos. Na implementação deste módulo não sugiram problemas relevantes, quer no encaminhamento, quer nos atrasos.

Para a implementação das restantes componentes do GLiTCH (*array* de processadores com memória associativa, encaminhador de padrões e descodificador de instruções) com a UPDI, partiu-se de descrições algorítmicas para cada um dos blocos. A implementação baseou-se em módulos descritos em linguagem C e desenvolvidos para o *transputer*[11].

5 CONCLUSÕES E TRABALHO FUTURO

Com base nos resultados obtidos com a emulação do circuito GLiTCH, conclui-se que o desempenho dos sistemas implementados depende fortemente da alocação dos recursos da ULP. Este desempenho também depende da forma de partição da especificação em componentes de *software* e *hardware*. Não se prevê que o nível da especificação influencie significativamente o desempenho dum sistema a implementar. O exemplo de aplicação serviu ainda para mostrar que a plataforma EDgAR permite implementar sistemas com elevada complexidade, sem precisar de escalar a plataforma, ligando-a a outros módulos. A utilização dum *transputer* na UPDI facilitou bastante o projecto da arquitectura da plataforma, por exigir uma interface simples com o meio envolvente e permitir a depuração da arquitectura em que se

[10] *Input/Output Blocks*

[11] Estes módulos foram aproveitados dum projecto anterior ao EDgAR [21], que visava emular o GLiTCH

engloba.

O exemplo de aplicação utilizado teve o senão de não utilizar MACHs, mas a validade destes componentes foi demonstrada com outros exemplos de menor dimensão.

Dado que os atrasos obtidos com as FPGAs foram superiores aos que se obtiveram com as MACHs, quando se implementaram módulos iguais em ambos os tipos de FPLDs, concluiu-se que a inclusão de dois tipos de FPLDs na ULP permite obter melhores desempenhos, uma vez que cada tipo é mais adequado à implementação de partes dos sistemas com características bem distintas.

Como os resultados obtidos durante o teste da plataforma EDgAR foram encorajadores, o trabalho futuro incidirá na sua integração numa plataforma para prototipagem rápida mais ambiciosa, que incluirá além de cópias do EDgAR, uma unidade microprogramável baseada num sequenciador de 16 bits e uma arquitectura MIMD baseada em *transputers*. Com a linguagem VHDL e as Redes de Petri pretende obter-se uma forma unificadora de especificação, que vai melhorar a comunicação entre as diferentes fases do processo de co-projecto: partição em componentes de *software* e de *hardware*, co-simulação, e síntese destas componentes.

Enquanto que para uma síntese automática, especialmente de *hardware*, existem várias ferramentas disponíveis, para efectuar a partição automática e a co-simulação muito há que investigar. Deste modo, grande parte do esforço futuro destinar-se-á a: (i) obter uma forma de partição que, dada uma especificação, gere representações para as componentes a implementar com os dois tipos de FPLDs do EDgAR, com a unidade microprogramável ou com os diferentes *transputers* da arquitectura; (ii) obter um co-simulador, que corra na arquitectura paralela, para acelerar o processo de co-simulação.

Referências

[1] Mike Spivey and Ian Page. How to Design Hardware with Handel, December 1993. Oxford University.

[2] Rajesh Gupta and Giovanni De Micheli. System-level Synthesis using Reprogrammable Components. In *Proceedings of the European Conference on Design Automation*, pages 2–7, Brussels, Belgium, 1992.

[3] António J. Esteves. Prototipagem Rápida de Sistemas Digitais - Parte I. Technical report, Departamento de Informática Universidade do Minho, July 1994.

[4] Inmos. *ANSI C Toolset User Manual*. August 1990.

[5] Advanced Macro Devices AMD. MACH Family Data Book, October 1991.

[6] Xilinx. *The Programmable Gate Array Data Book*. 1992.

[7] M. Chiodo, P. Giusto, H. Hsieh, A. Jurecska, L. Lavagno, and A. Sangiovanni-Vincentelli. A Formal Specification Model for Harware/Software Codesign. Technical Report ERL-93-48, University of California, Berkeley, June 1993.

[8] C. A. R. Hoare. *Communicating Sequential Processes*. Prentice-Hall International, 1985.

[9] Manuel Silva and Robert Valette. Petri Nets and Flexible Manufacturing. In G. Rozenberg, editor, *Advances in Petri Nets 89*, volume 424 of *Lecture Notes in Computer Science*, pages 376–417. Springer-Verlag, Berlin, Germany, 1990.

[10] Douglas L. Perry. *VHDL*. McGraw-Hill, 1991.

[11] M. Chiodo, P. Giusto, H. Hsieh, Jurecska A, L. Lavagno, and A. Sangiovanni-Vincentelli. A Formal Method for Hardware/Software Co-design of Embedded Systems. *IEEE Micro Magazine*, August 1994.

[12] João Miguel Fernandes. Redes de Petri e VHDL na Especificação de Controladores Paralelos. Master's thesis, Dep. Informática, Universidade do Minho, Braga, Portugal, July 1994.

[13] Xilinx. *User Guide and Tutorials*. 1991.

[14] W. Billowitch. Simulation Models for Support Hardware/Software Integration. *Computer Design*, pages 31 –, 1988.

[15] A. J. Proença, H. D. Santos, J. C. Ramalho, and J. M. Fernandes. An heterogegeous computer vision architecture: implementation issues. *Computing Systems in Engineering (in press)*. Apresentado também no *1o Encontro Intern. sobre Process. Vectorial e Paralelo, Porto, Out,1993*.

[16] A. W. G. Duller, R. H. Storer, A. R. Thomson, E. L. Dagless, M. R. Pout, and A. P. Marriot. Design of an Associative Processor Array, 1989.

[17] OrCAD. OrCAD - Programmable Schematic Design Tools Reference Guide, 1991.

[18] OrCAD. OrCAD - Programmable Logic Design Tools Reference Guide, 1991.

[19] Viewlogic. Workview Utilities Manual - Version B. VIEWlogic Systems Inc., 1990.

[20] Viewlogic. ViewPLD Users Guide. VIEWlogic Systems Inc., 1992.

[21] Manuel J. D. Alves. Emulação dum Processador Associativo Vectorial - GLiTCH, December 1992. Relatório de Estágio da Lic. em Eng. de Sistemas e Informática.