

## Computação Reconfigurável. WebCT, *Tutorials* e Projectos<sup>\*</sup>

Valery Sklyarov, Iouliia Skliarova

**Resumo** – Este artigo apresenta métodos originais e ferramentas inovadoras (nomeadamente, *tutorials* animados, projectos e organização dos dados na WebCT) utilizados no âmbito das disciplinas de computação reconfigurável e de sistemas digitais avançados. São apresentados muitos exemplos que demonstram como aceder a todos os materiais disponíveis e como estes materiais ajudam aos alunos a compreenderem vários aspectos do desenvolvimento de hardware.

**Abstract** – The paper describes original methods and novel tools (namely animated tutorials, projects and organization of data on WebCT) that have been used for teaching disciplines on reconfigurable computing and advanced digital systems. Many examples are provided, which demonstrate how to get access to all the available materials and how these materials help the students to understand many different aspects of hardware design.

### I. INTRODUÇÃO

Com o advento de dispositivos lógicos programáveis (PLDs - *Programmable Logic Devices*) ficou possível projectar e implementar sistemas digitais bem como os seus componentes sem recorrer aos passos tecnológicos que lidam com o silício. Este processo é muito semelhante ao desenvolvimento de software para computadores de uso geral. Inicialmente, os PLDs foram utilizados para implementar circuitos digitais únicos que normalmente possuíam uma complexidade bastante reduzida. A eficácia desta abordagem estimulou uma investigação muito intensiva nesta área conduzindo ao desenvolvimento de novas gerações de PLDs de complexidade elevada que incluem um grande número de primitivas lógicas e possuem capacidades mais poderosas. Hoje em dia, os PLDs representam uma boa alternativa aos ASICs (*Application Specific Integrated Circuits*) pois são utilizados numa ampla gama de aplicações práticas tais como coprocessadores para computadores de uso geral, controladores embutidos, placas de protótipo, etc.

O facto de o domínio de aplicações potenciais aumentar muito rapidamente, requer a formação adequada de engenheiros experientes nas áreas relevantes. Sendo assim, as tendências novas devem ser reflectidas nos conteúdos das disciplinas relevantes que fazem parte da formação inicial e pós-graduada. É de salientar que o domínio do projecto de sistemas reconfiguráveis é muito

dinâmico e vasto. Isto requer uma actualização periódica dos conteúdos das disciplinas respectivas a fim de representar os avanços recentes na tecnologia de PLDs, no desenvolvimento de sistemas digitais e nas ferramentas de projecto assistido por computador (CAD - *Computer Aided Design*). Este artigo dissemina os resultados atingidos neste contexto e descreve as três componentes originais que foram utilizadas com sucesso na leccionação das disciplinas relevantes durante a última meia dúzia de anos.

### II. DIRECÇÕES BÁSICAS NA LECCIONAÇÃO DE SISTEMAS RECONFIGURÁVEIS

A fig. 1 ilustra as direcções principais que serviram de base para as disciplinas relacionadas com o projecto de sistemas digitais reconfiguráveis. Estas direcções são: 1) métodos e ferramentas; 2) interacção dos circuitos baseados em PLDs com os dispositivos externos; e 3) placas de protótipo utilizadas nas aulas práticas. Os rectângulos sombreados denotam algumas aplicações que foram desenvolvidas por alunos. A maioria dos projectos dos alunos estão disponíveis na WebCT [1], sendo muitos destes descritos em artigos publicados na revista do Departamento de Electrónica e Telecomunicações (DET) [2-11].

Actualmente, utilizam-se os dois tipos seguintes de placas de protótipo com os componentes reconfiguráveis instalados (nomeadamente, com as FPGAs - *Field-Programmable Gate Arrays* e CPLDs - *Complex Programmable Logic Devices*):

- As *placas autónomas* ligam-se ao computador hospedeiro através de uma porta externa tal como a porta paralela ou USB. As portas servem para os dois objectivos principais: 1) configurar a FPGA e programar outros componentes da placa (por exemplo, memória *flash*); e 2) suportar a transferência de dados entre a placa e o computador hospedeiro. Exemplos deste tipo de placas são TE-XC2Se da Trenz Electronic [12], RC100 e RC200 da Celoxica [13] e XSA100 da XESS [14], cuja descrição detalhada pode ser encontrada em [15]. Normalmente, as placas autónomas são relativamente baratas e asseguram uma série de interacções compreensivas com os dispositivos externos (à FPGA) localizados na própria placa ou ligados através das portas externas ou dos conectores de expansão [15]. Isto permite que os alunos examinem (com a ajuda de um osciloscópio ou analisador

<sup>\*</sup> Trabalho financiado parcialmente com a bolsa da FCT-PRAXIS XXI/BD/21353/99

lógico) os sinais físicos a fim de perceber melhor a interface entre os dispositivos, fornecer os valores desejados dos sinais de entrada e apreciar os resultados de um modo visual.

- As *placas PCI* são instaladas no barramento PCI do computador hospedeiro ficando assim escondidas dos alunos. A configuração das FPGAs e a interacção com os sistemas desenvolvidos decorrem com a ajuda de um conjunto de funções API (*Application Programming Interface*) assemelhando-se deste modo à programação. Exemplos deste tipo de dispositivos são ADM-XRC e ADM-XPL da Alpha Data [16]. Estas placas são normalmente muito mais dispendiosas que as autónomas. Contudo, as placas PCI [16] contêm FPGAs muito poderosas e portanto são recomendadas para alunos experientes, em particular para alunos de pós-graduação. É de notar também que as placas PCI suportam interacção muito rápida com as aplicações de software permitindo deste modo construir coprocessadores.

Do lado esquerdo da fig. 1 estão indicadas as placas de protótipo que foram utilizadas desde 1997. Estas permitem implementar e testar circuitos digitais com base em todas as famílias principais da Xilinx que estiveram disponíveis no mercado durante o período mencionado. Os nomes das placas de protótipo estão colocados dentro dos rectângulos. A informação sobre muitas placas recentes pode ser encontrada via página da Xilinx [17]. Os tipos de FPGAs estão indicados acima dos rectângulos respectivos. À esquerda de cada rectângulo encontra-se o período em que a placa respectiva foi utilizada.

As ferramentas de projecto foram seleccionadas a maneira de possibilitar os alunos a aprenderem linguagens de especificação a nível de sistema bem como as linguagens de descrição de hardware tradicionais (HDLs – *Hardware Description Languages*). Assim, foram leccionadas duas linguagens de especificação a nível de sistema, nomeadamente SystemC [18] e Handel-C [13]. SystemC é uma biblioteca de classes que permite modelar componentes de hardware usando um compilador C++ *standard*. Dado que o DET não dispõe de ferramentas de software que sintetizem circuitos baseados em FPGAs a partir de uma especificação em SystemC, esta biblioteca só foi considerada a nível teórico. Ao contrário disso, a linguagem Handel-C e o ambiente DK1 da Celoxica [13] foram utilizados para projectar e implementar sistemas digitais baseados em FPGAs.

*Handel-C* permite descrever circuitos digitais de modo parecido com a linguagem C, modelá-los, sintetizá-los e implementá-los em hardware com a ajuda de ferramentas de projecto desenvolvidas e fornecidas pela Celoxica [13]. O fluxo de projecto inclui os passos básicos seguintes:

- especificação em Handel-C;
- verificação e validação da descrição em software;
- síntese do sistema de hardware.

O resultado é um ficheiro EDIF (*Electronic Design Interchange Format*) que pode ser usado para o

mapeamento, colocação, encaminhamento e geração da configuração de FPGA com a ajuda de ferramentas CAD disponíveis comercialmente, tais como ISE 5.2 da Xilinx [17]. Finalmente, o ficheiro de configuração é carregado na FPGA seleccionada.

De modo alternativo, a especificação em Handel-C pode ser convertida no código VHDL sintetizável a fim de o utilizar num ambiente CAD para construir componentes de biblioteca (veja o rectângulo “Combinação” na fig. 1). Actualmente, para o fluxo de projecto baseado em VHDL usa-se o ambiente ISE 5.2 da Xilinx [17] que tem as ferramentas de simulação *ModelSim* [19] incorporadas.

Os métodos de projecto foram seleccionados a maneira de possibilitar os alunos a aprenderem implementar vários modos de reconfiguração que permitem modificar a funcionalidade dos circuitos respectivos estaticamente e dinamicamente. Para tal, primeiro são consideradas as arquitecturas de FPGAs e as suas capacidades funcionais que suportam a reconfiguração. De seguida, são leccionadas duas maneiras novas que permitem modificar a funcionalidade de circuitos com a ajuda dos blocos especiais implementados em FPGA ou das aplicações de software que comunicam com a FPGA através de uma interface *standard* (tal como PCI ou porta paralela). A fim de realizar as modificações são utilizadas as duas técnicas seguintes.

A primeira técnica baseia-se em modelos (HT – *hardware templates*) reutilizáveis. Um HT é um circuito com a estrutura predefinida cujos componentes principais são blocos de memória RAM ou ROM. O HT é implementado em hardware e pode ser reprogramado a fim de estabelecer funcionalidades diferentes. A reprogramação é suportada por blocos adicionais removíveis denominados por *controladores de reconfiguração*. Deste modo torna-se possível avaliar várias funcionalidades alternativas eliminando assim muitos passos de projecto tradicionais que consomem tempo e recursos adicionais. Em [3-6] são apresentados exemplos de projectos baseados nesta técnica e desenvolvidos por alunos. Os projectos permitem alterar as operações dos circuitos implementados reprogramando para tal os blocos RAM de uma máquina de estados finitos (FSM – *Finite State Machine*) que determinam a funcionalidade desejada. A reprogramação é realizada através da porta paralela do computador.

A segunda técnica baseia-se em especificações hierárquicas compostas por módulos e nas implementações relevantes (veja o *tutorial 10* [1]). Neste caso todas as modificações necessárias são efectuadas substituindo qualquer módulo escusado com um módulo novo que assegura a funcionalidade desejada (ou nova).

Ambas as técnicas assumem que os métodos tradicionais de projecto de sistemas digitais são bem conhecidos. Portanto, os métodos tradicionais devem ser periodicamente recordados a fim de esclarecer alguns tópicos e mostrar consistência com os materiais das disciplinas precedentes leccionadas no DET.

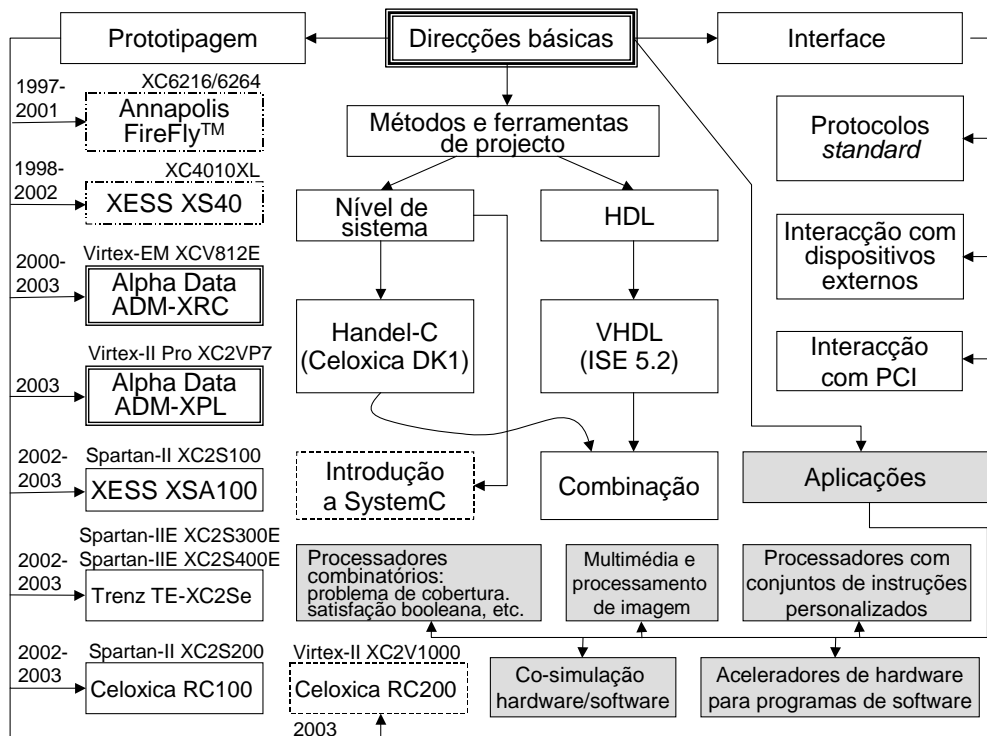


Fig. 1 – Direcções principais das actividades pedagógicas no domínio do projecto de sistemas reconfiguráveis

A maioria das aplicações práticas requerem que os circuitos baseados em FPGA entrem em interacção com os dispositivos externos, tais como memórias, *displays*, microprocessadores, outras FPGAs, etc. Portanto, torna-se necessário implementar as interfaces respectivas. Ahamos estes tópicos muito importantes para aprendizagem e, por esta razão, eles foram incluídos nos conteúdos das disciplinas relevantes. Nomeadamente, lecciona-se como configurar os blocos de I/O (*Input/Output*) da FPGA para suportarem normas de I/O necessárias, como estabelecer os mecanismos de sincronização da FPGA com os dispositivos externos, como construir os *buffers* de entrada/saída, etc.

Nas aulas analisam-se os três tipos seguintes de interfaces (veja a parte direita da fig. 1): protocolos *standard*, tais como RS232, USB, etc.; interface PCI (i.e. troca dos dados através do barramento PCI de computadores); e interacções da FPGA com os dispositivos externos tais como memória estática, LCD (*Liquid Crystal Display*), controladores, microprocessadores, etc. A maioria das interfaces consideradas são suportadas pelos componentes relevantes existentes nas placas de protótipo.

A fig. 2 revela alguns detalhes sobre a metodologia de ensino adoptada que se caracteriza por uma série de propriedades distintas.

A todas as aulas associa-se um conjunto de *tutorials* animados complementados com exemplos de projecto orientado para FPGAs. Os *tutorials*, exemplos e outros materiais suplementares estão disponíveis para as disciplinas relevantes na WebCT. Na parte esquerda da fig. 2 estão indicadas as ferramentas de software/hardware básicas que são utilizadas nas aulas práticas. As partes

central e direita da fig. 2 mostram as características comuns dos *tutorials* desenvolvidos e os atributos típicos dos projectos dos alunos, respectivamente. As secções subsequentes descrevem as propriedades mencionadas acima em mais detalhe. Estas incluem a organização da WebCT, exemplos de *tutorials* e projectos dos alunos.

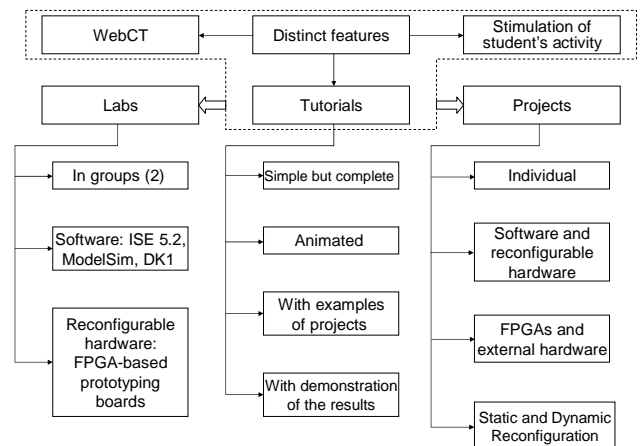


Fig. 2 – Propriedades comuns das aulas teóricas e práticas relacionadas com o projecto de sistemas reconfiguráveis

### III. WEBCT

Para manter a eficácia elevada das aulas, os alunos devem ter acesso a todos os materiais necessários. Estes materiais são fornecidos através da WebCT. A fig. 3 ilustra um exemplo de organização da WebCT para a disciplina “Computação Reconfigurável” [1].

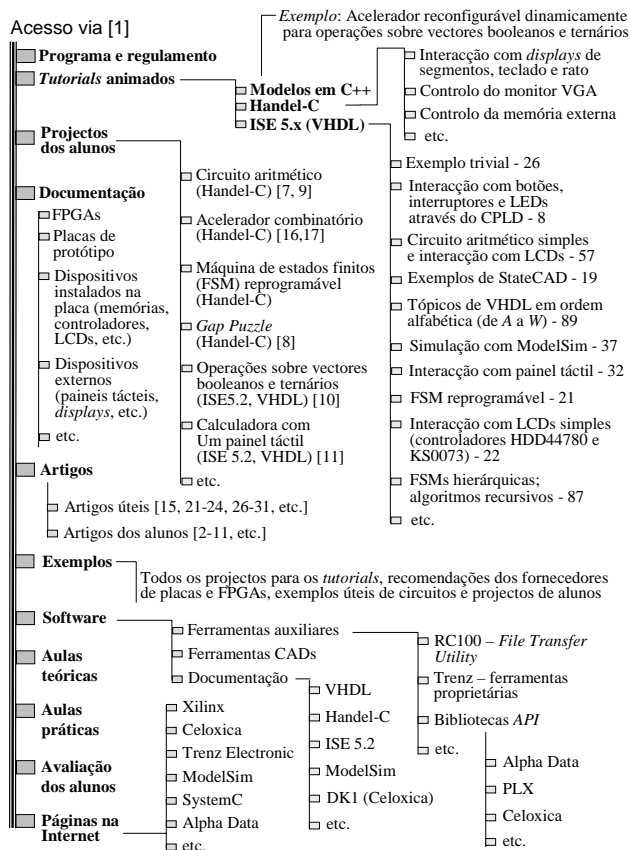


Fig. 3 – Organização da WebCT para a disciplina “Computação Reconfigurável”

Os dados na WebCT estão organizados a maneira de possibilitar os alunos a encontrarem rapidamente e copiarem toda a informação de que precisarem. As secções “Aulas teóricas” e “Aulas práticas” são actualizadas periodicamente durante o semestre e incluem apenas referências aos *tutoriais*, exemplos, projectos, etc. Deste modo pode-se tomar em consideração as capacidades diferentes dos alunos que normalmente variam de ano em ano. Todos os tópicos das aulas teóricas são reflectidos em *tutoriais* (as explicações adicionais são fornecidas em [15]) que são continuamente expandidos. Os números que seguem os nomes dos *tutoriais* denotam a quantidade de *slides* animados respectivos, possibilitando desta maneira estimar o volume dos materiais fornecidos. Os exemplos disponíveis podem ser aproveitados por alunos futuros permitindo por um lado aumentar gradualmente a complexidade dos projectos e por outro lado desenvolver circuitos e sistemas com base em experiência ganha anteriormente. As oportunidades semelhantes adicionais são previstas nos *tutoriais*. Por exemplo, o *tutorial* sobre VHDL descreve todas as construções de VHDL sintetizáveis (enumeradas em ordem alfabética de A a W) e fornece exemplos de código respectivos. Os excertos de código podem ser copiados para os projectos dos alunos (assemelhando-se deste modo com os modelos de linguagem – *language templates* da Xilinx) e modificados de acordo com as necessidades particulares. Esta possibilidade acelera essencialmente o processo de aprendizagem das

linguagens de descrição de hardware e simplifica a concepção de temas novos tais como concorrência, sincronização, etc. Técnicas semelhantes foram aplicadas em outros *tutoriais* disponíveis em [1].

#### IV. TUTORIALS

Todos os *tutoriais* possuem estrutura semelhante e são complementados com exemplos (projectos) relevantes. Os projectos podem ser abertos em ambientes respectivos e sintetizados para gerar o ficheiro de configuração a ser carregado na FPGA e testado. Os exemplos foram preparados para as placas de protótipo utilizadas actualmente (veja a fig. 1) mas são facilmente adaptáveis às FPGAs semelhantes. Embora a complexidade dos *tutoriais* aumente gradualmente estes não são forte e mutuamente dependentes. Normalmente cada *tutorial* pode ser utilizado de uma maneira independente permitindo aprender qualquer tópico particular. Caso sejam necessárias algumas explicações teóricas, estas são fornecidas junto com as referências às publicações relevantes que estão também disponíveis na WebCT. Para os tópicos difíceis (tais como implementação de algoritmos recursivos) são considerados e analisados os modelos de software relevantes (veja, por exemplo, o programa C++ que descreve algoritmos de ordenação recursiva em árvores binárias para o *tutorial* 10). As três subsecções seguintes (A-C) são dedicadas às características básicas dos *tutoriais* que são a *estrutura*, a *animação* e a *aplicabilidade e reutilização*.

A originalidade da *estrutura* consiste na técnica proposta que se baseia em blocos *permanentes* e *testados*. Os blocos *permanentes* são componentes projectados preliminarmente que asseguram interacções com os dispositivos de entrada/saída destinados a fornecer os dados iniciais e visualizar os resultados. Os blocos *testados* permitem verificar os circuitos respectivos que são discutidos no *tutorial* relevante e examinar vários modos do seu funcionamento. A *animação* ilustra cenários diferentes passo a passo reduzindo significativamente o tempo de aprendizagem. A *aplicabilidade* e a *reutilização* dizem que os *tutoriais* incluem muitos fragmentos de código que podem ser ou utilizados directamente ou aproveitados em projectos futuros.

A subsecção D. *Aspectos teóricos e modelação em software* ilustra como abordar aspectos teóricos e utilizar uma linguagem de uso geral para verificar vários algoritmos e arquitecturas antes da sua implementação em hardware. A subsecção E. *Descrição a nível de sistema* é dedicada à relação entre HDLs e as linguagens de especificação a nível de sistema.

##### A. Estrutura

Para simplificar o processo de depuração e verificação de circuitos baseados em FPGAs, as placas de protótipo oferecem muitas oportunidades de interacção que são sustentadas por: 1) componentes existentes na placa (tais

como botões, interruptores, LEDs, *displays* de segmentos, LCDs, etc.); 2) conectores aos dispositivos externos que suportam uma série de interfaces *standard* (tais como RS232, USB, PS2, VGA, etc.); e 3) conectores de expansão que podem ser empregados para interacções com microcontroladores, outras placas, etc. de acordo com o formato definido pelo utilizador.

Os três primeiros *tutorials* disponíveis em [1] apresentam circuitos triviais reutilizáveis que asseguram interacções com componentes típicos instalados nas placas que são botões, interruptores, LEDs e LCDs. A reutilização é suportada pela biblioteca respectiva ou pelo código VHDL que pode ser inserido em projectos novos. A fig. 4 ilustra esta oportunidade para o caso do *tutorial* sobre VHDL.

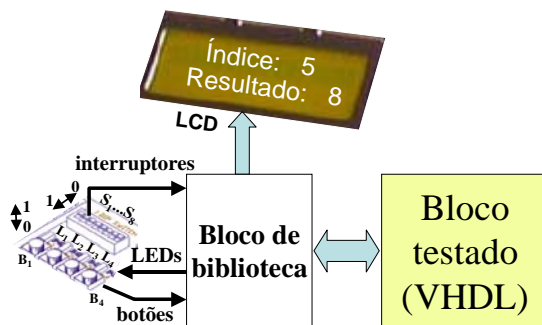


Fig. 4 – Utilização dos blocos de biblioteca para testar várias construções de VHDL

O bloco VHDL testado pode ser modificado copiando extractos de código VHDL de qualquer secção do *tutorial*. Muitas secções são compostas por subsecções que permitem que as construções VHDL relevantes sejam examinadas de modos diferentes. Por exemplo, a secção *Aggregates* explica construções tais como agrupamento de valores, associações de posição e nomeadas, etc. A fig. 5 demonstra como testar uma destas construções.

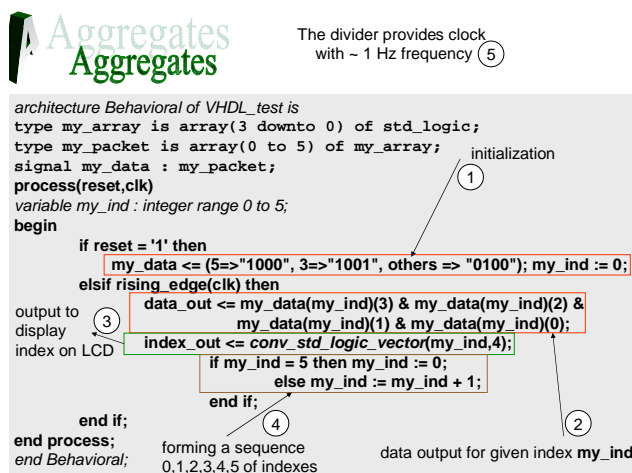


Fig. 5 – Demonstração da técnica utilizada nos *tutorials*

Num dos modos de funcionamento o sinal do relógio (*clk*) provém de um botão existente na placa. Isto permite testar e visualizar todos os seis índices (*my\_ind*) bem

como os valores *data\_out* correspondentes. A fig. 4 ilustra os resultados que aparecem no LCD para *my\_ind*=5 (obviamente, *data\_out*=8). Um outro modo de funcionamento permite gerar automaticamente o sinal de relógio com uma frequência baixa permitindo que o utilizador observe todas as modificações que ocorrem nos sinais.

A técnica considerada (veja fig. 4) é aplicada na maioria dos *tutorials* restantes. Os dois *tutorials* mais avançados (nomeadamente, o sétimo e o nono) estendem estas capacidades básicas e introduzem blocos reutilizáveis [21] para LCDs externos e para o painel tátil (*touch panel*) EA KIT240-7 da Electronic assembly [20].

### B. Animação

A animação permite ilustrar várias sequências de projecto e de execução passo a passo destacando todos os resultados intermédios importantes. Um exemplo trivial está apresentado na fig. 5. É conhecido que qualquer processo em VHDL consiste nos comandos sequenciais. Os números 1-5 e os comentários relevantes descrevem a execução destes comandos que é controlada pelos sinais *reset* e *clk* da lista sensível do processo. Qualquer atribuição de valores aos sinais num processo (que eventualmente provoca alterações no ecrã do LCD) só entra em vigor quando o processo suspender. Este facto também pode ser demonstrado eficientemente com a ajuda da animação. Nas apresentações em *PowerPoint* é possível navegar para frente e para trás o que simplifica essencialmente a compreensão de tópicos diferentes. Isto é importante especialmente para alunos que são familiares com as linguagens de programação de uso geral mas não possuem experiência suficiente no projecto de hardware baseado em HDL. Esta situação é muito comum e portanto conduz a muitos erros.

Uma outra característica importante da animação é a oportunidade de executar um cenário seleccionado e criar um outro cenário semelhante orientado para uma aplicação particular de modo análogo. Esta técnica será ilustrada com o exemplo do *tutorial* que demonstra como interagir com um LCD gráfico e o painel tátil (EA KIT240-7 da Electronic assembly [20]).

A fig. 6 enumera os passos básicos necessários para receber através da porta série RS232 o código da área activa do painel tátil [20]. O painel tátil é ligado à placa TE-XC2Se [12]. O sinal *rs232in* contém os dados seriais da RS232. Os passos 1-4 executados sequencialmente na apresentação animada, permitem perceber como o código VHDL descreve a recepção dos dados da RS232, i.e. como encontra o *start bit*, obtém 8 bits de dados, detecta o *stop bit* e visualiza no LCD um carácter que corresponde ao código ASCII recebido. O sinal *clk* assegura a velocidade de transmissão (*baud rate*) adequada. É possível copiar o código VHDL considerado e utilizá-lo em projectos semelhantes. Para além disso, fragmentos de código semelhantes podem ser construídos de modo análogo (por exemplo, pode-se incorporar a verificação da paridade, etc.).

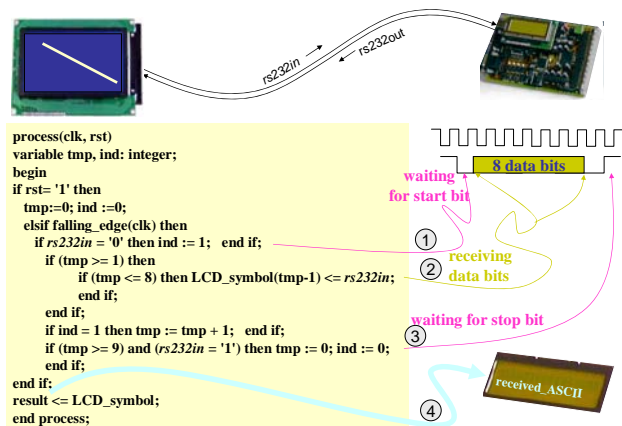


Fig. 6 – Recepção do código da área activa do painel tátil através da interface RS232

A fig. 7 demonstra como verificar o processo de envio dos dados para o painel tátil com a ajuda da ferramenta *ModelSim* [19]. Os passos 1-8 ilustram como o código hexadecimal *1B* com que normalmente começam todas as sequências de controlo [20], é enviado através da RS232. Um *tutorial* separado (o sexto) que pode ser usado como um material suplementar, descreve como trabalhar no ambiente *ModelSim* (veja a fig. 3). É possível organizar qualquer cenário semelhante em *ModelSim* de modo análogo.

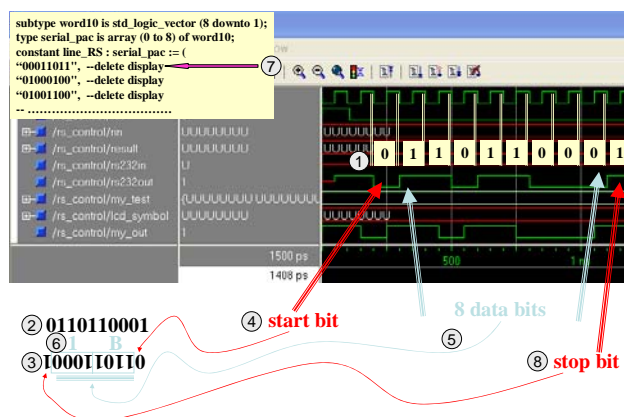


Fig. 7 – Utilização do ambiente ModelSim para verificar o código VHDL que descreve envio dos dados através da RS232

### C. Aplicabilidade e reutilização

A fig. 8 explica como trabalhar com o painel tátil ligado à placa TE-XC2Se depois de programar a FPGA com o ficheiro de configuração preparado para o exemplo anterior. Ilustra-se como construir um menu e como activar as suas opções diferentes através das áreas tácteis A, B, C e D. Os dados provenientes das áreas tácteis são visualizados no ecrã do LCD da placa TE-XC2Se. O menu é composto por três campos que são *ADD*, *ORT* e *TEST*. Assumamos que cada campo é responsável por uma operação associada com o nome respectivo (i.e. *ADD*, *ORT* e *TEST*). Em cada instante apenas um campo

está activo. As áreas tácteis A e C permitem comutar o campo activo movendo-o para baixo ou para cima respectivamente. O *tutorial 9* demonstra um cenário possível que pode ser usado para testar o circuito ilustrado na fig. 8 e, de seguida, explica todo o código VHDL que descreve as operações válidas. Este método faz com que os alunos primeiro compreendam o exemplo considerado e a seguir modifiquem-no a maneira de satisfazer os requisitos de uma aplicação particular tal como desenvolvimento de uma calculadora com base no painel tátil, etc. (veja exemplos de projectos dos alunos em [10, 11]).

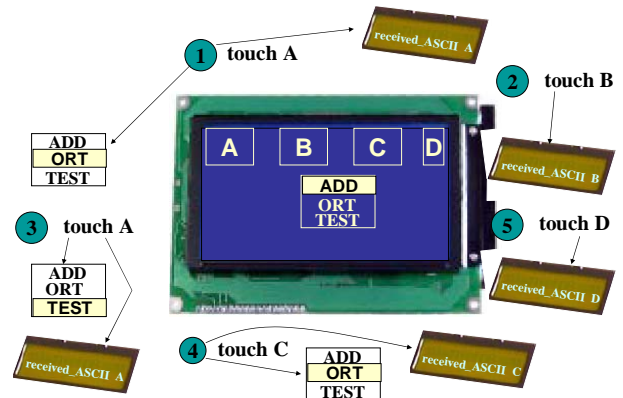


Fig. 8 – Algumas operações permitidas no painel tátil

### D. Aspectos teóricos e modelação em software

Muitos tópicos abordados nos *tutorials* exigem alguns conhecimentos adicionais. Por exemplo, o *tutorial 8* dedicado às FSMs reprogramáveis (RFSMs), baseia-se em modelos específicos [22] e em métodos que suportam a sua reprogramação [23]. O *tutorial 10* demonstra a implementação de algoritmos de ordenação recursivos com a ajuda das estruturas de dados modeladas por árvores binárias. Presume-se que os modelos empregados [24] são bem compreendidos e que o utilizador possui conhecimentos teóricos relevantes [24, 25]. A metodologia adoptada requer que todos os materiais adicionais estejam também disponíveis na WebCT. Normalmente, os *tutorials* respectivos contêm todas as explicações suplementares. Para tal são aplicados os três métodos seguintes:

- Modelação de circuitos de hardware recorrendo à tecnologia orientada por objectos. O programa relevante é construído a maneira de facilitar a compreensão quer da estrutura (arquitectura) do circuito e da interacção dos seus blocos básicos, quer do comportamento do circuito, i.e. dos detalhes de implementação dos algoritmos apropriados. No primeiro caso os objectos do programa modelam blocos de hardware estruturais enquanto a troca de informação entre os blocos é imitada com a ajuda das mensagens circuladas entre objectos. Um exemplo deste modelo considera-se em [26] e permite descrever a estrutura de um acelerador reconfigurável dinamicamente destinado a executar operações sobre vectores booleanos e



ternários. O código C++ deste modelo está disponível na WebCT [1]. O segundo caso, em que o programa modela comportamento do circuito, será considerado mais abaixo.

- Fornecimento de explicações que mostram a ligação entre o modelo de software e o circuito de hardware e explanam como uma descrição estrutural ou comportamental em software pode ser transformada num circuito de hardware. Quando possível, vários modos de depuração disponíveis para os programas de software respectivos, são utilizados para compreender a funcionalidade dos blocos de hardware mais complexos (por exemplo, a execução de algoritmos recursivos no *tutorial 10*).

- Demonstração (em *tutorials* respectivos) da funcionalidade dos circuitos, difíceis para a compreensão, passo a passo realçando todos os resultados intermédios importantes, tais como operações com a pilha para os algoritmos recursivos. Isto é em particular importante para a percepção dos mecanismos de sincronização complexos.

As fig. 9-11 (veja *tutorial 10* em [1]) ilustram as ideias básicas destes métodos. Inicialmente, o *tutorial* apresenta um programa C++ que constrói uma árvore binária especial (com a ajuda da função recursiva *dr*) e ordena os dados extraídos desta árvore binária (com a ajuda da função recursiva *drpr* ilustrada na fig. 9). O programa C++ constitui um modelo comportamental e permite compreender todas as operações necessárias em modos de depuração. As partes subsequentes do *tutorial* explicam como implementar algoritmos recursivos semelhantes em hardware. A fig. 9 ilustra como a função de software *drpr* pode ser transformada num algoritmo de hardware. Todos os passos principais desta conversão são mostrados sequencialmente na apresentação animada respectiva. Materiais adicionais sobre a modelação em software estão disponíveis em [27].

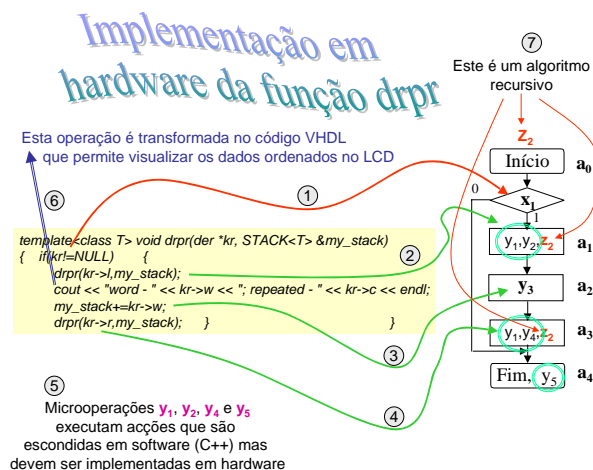
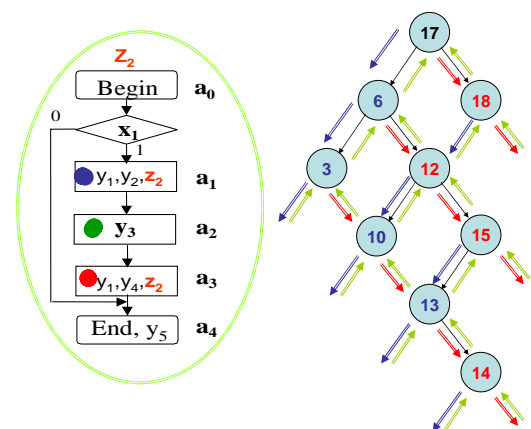


Fig. 9 – Implementação em hardware da função recursiva *drpr* que ordena dados extraídos da árvore binária

A fig. 10 assume que é necessário ordenar os dados de uma árvore binária que já foi construída com a ajuda da função *dr* ou qualquer outra função semelhante. A árvore binária é organizada de tal maneira que para cada nó-pai

particular a sub-árvore esquerda contém apenas valores menores que o valor do nó-pai e a sub-árvore direita guarda respectivamente valores maiores (veja [25] para detalhes). O *slide* animado do *tutorial 10* [1] que foi construído com base na fig. 10, inclui 72 passos sequenciais que demonstram todos os detalhes de execução do algoritmo conduzindo à extração dos dados ordenados (veja a parte inferior da fig. 10).

Assumamos que a árvore binária ilustrada na fig. 10 já foi construída com a ajuda de uma função semelhante à *dr*. O algoritmo (módulo)  $z_2$  (veja fig. 9 e 10) que se pretende implementar em hardware possibilita ordenar inteiros de maneira mostrada na parte inferior da fig. 10. O módulo  $z_2$  interage com os componentes do *datapath* que são explicados em detalhe no *tutorial 10*. Inicialmente, o sinal  $x_1$  (veja a fig. 10) é testado e dependendo do seu valor (0 ou 1) o controlo será transmitido a um dos nós  $a_1$  ou  $a_4$ . O nó  $a_1$  activa dois sinais de saída  $y_1$  e  $y_2$  e chama recursivamente o mesmo módulo  $z_2$ . O nó  $a_4$  é um nó final que activa um sinal de saída  $y_5$ . Caso seja efectuada a chamada recursiva, o nó losangular que contém a condição  $x_1$  será testado novamente. Quando se realiza um retrocesso hierárquico, o controlo deve ser transmitido ao nó que foi a origem da chamada recursiva mais recente. A execução do algoritmo conclui ao atingir o nó final desde que não haja necessidade de realizar retrocessos hierárquicos. Os três nós rectangulares na fig. 10 estão marcados com círculos (no *tutorial* as cores dos círculos são diferentes). A execução do módulo  $z_2$  é modelada destacando sequencialmente os elementos activos da fig. 10. Estes elementos são: 1) círculos coloridos em  $z_2$ ; 2) setas relevantes de cores diferentes na árvore binária; e 3) os inteiros extraídos (quando necessário) que são adicionados à sequência de números em baixo. Normalmente, em qualquer instante está activo mais que um destes elementos, portanto as acções relevantes desenrolam-se no *tutorial* em paralelo. Isto permite compreender como a árvore binária é percorrida e como os inteiros são extraídos e ordenados.



18 17 15 14 13 12 10 6 3

Fig. 10 – Demonstração do algoritmo de hardware que ordena inteiros extraídos duma dada árvore binária

Os *slides* subsequentes explicam como os blocos seguintes foram descritos em VHDL:

- memória do tipo pilha;
- FSM hierárquica (HFSM) [24] que executa algoritmos recursivos compostos por módulos;
- circuito que insere um novo nó na árvore binária;
- circuito que constrói a árvore binária com base numa série de números inteiros (cuja quantidade é previamente desconhecida);
- circuito que ordena os dados extraídos da árvore binária e visualiza os resultados no LCD;
- projecto completo para ordenação de dados de acordo com critérios diferentes.

Existem 7 projectos desenvolvidos no ambiente ISE 5.2 e acessíveis via [1] que estão preparados para serem executados. Estes projectos podem ser analisados para explorar os tópicos relevantes ou aproveitados na implementação de sistemas mais complexos.

Todas as construções VHDL principais são explicadas minuciosamente e a funcionalidade dos blocos de hardware respectivos é demonstrada em apresentações animadas. A fig. 11 ilustra como a pilha da HFSM é empregada nas chamadas recursivas (passo 3), no retrocesso recursivo (passo 4) e nas transições ordinárias entre os nodos rectangulares do algoritmo (passo 5). Todos os materiais adicionais, tais como [24] estão também disponíveis na WebCT [1].

Os *tutorials* considerados simplificam essencialmente o processo de aprendizagem de tópicos diferentes que surgem no domínio da computação reconfigurável e em áreas relacionadas.

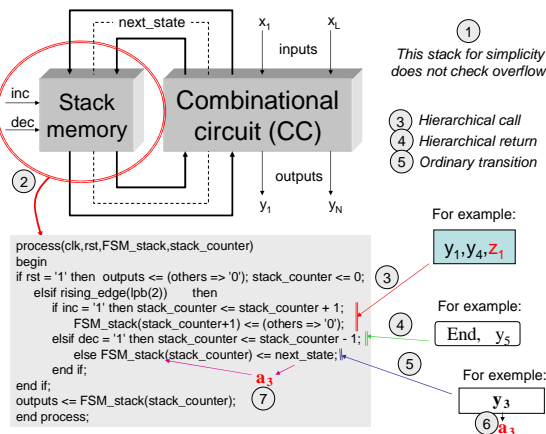


Fig. 11 – Código VHDL da pilha da HFSM

### E. Descrição a nível de sistema

As linguagens de descrição de hardware, tais como VHDL, e as linguagens de especificação a nível de sistema (SLSLs - *System-Level Specification Languages*),

tais como Handel-C, servem para o mesmo fim (no nosso caso, para o desenvolvimento de um circuito baseado em FPGA) mas possuem estilos diferentes de descrição. Normalmente, as HDLs e as ferramentas de síntese relevantes asseguram uma optimização mais profunda enquanto as SLSLs simplificam e aceleram essencialmente o processo de projecto. A nossa experiência mostra que os alunos encontram muitas dificuldades na percepção de algumas construções de SLSLs. Para além disso, as SLSLs escondem muitas particularidades importantes da funcionalidade de hardware. Frequentemente, torna-se impossível estabelecer a sincronização desejada, construir blocos assíncronos, etc. Contudo, o domínio de SLSLs é muito importante e tem a tendência de se expandir no futuro.

O compromisso assumido nas disciplinas consideradas permite ensinar o projecto de sistemas digitais com base em HDLs e SLSLs através das duas fases seguintes.

Na primeira fase, os alunos aprendem todos os aspectos de projecto com base em HDLs (em particular, VHDL). Isto permite conhecer as técnicas fundamentais do projecto de hardware, todos os aspectos importantes da funcionalidade de hardware e uma série de opções e alternativas disponíveis para a síntese.

Na segunda fase são consideradas as SLSLs de modo geral e, de seguida, é leccionada uma linguagem particular (nomeadamente, Handel-C). A experiência preliminar adquirida por parte dos alunos durante a primeira fase simplifica a explicação das construções de SLSLs estabelecendo analogias relevantes com HDLs; permite comparar construções diferentes em HDLs e SLSLs; elimina muitos erros potenciais no projecto; faz os mecanismos de sincronização disponíveis em SLSLs mais compreensíveis, etc. Para além disso, torna-se possível utilizar no mesmo sistema blocos baseados em HDLs e SLSLs.

Os tópicos dos *tutorials* que devem assistir na aprendizagem de SLSLs (em particular, Handel-C) foram escolhidos a maneira de permitir aos alunos a:

- compreender a interacção com os dispositivos periféricos típicos através dos *drivers* relevantes, i.e. com a ajuda do código Handel-C pré-compilado que pode ser ligado ao código de um sistema específico à aplicação. Por exemplo, a Celoxica [13] fornece *drivers* para monitores VGA, rato, teclado, RAM estática, memória *flash*, porta paralela, *displays* de segmentos, etc.

- aprender as capacidades e características de Handel-C por analogias e comparações com os *tutorials* sobre VHDL considerados acima. Portanto, torna-se necessário demonstrar como descrever em Handel-C, validar, modelar, sintetizar e implementar em FPGAs circuitos que já foram desenvolvidos em VHDL (veja o item ISE 5.x (VHDL) na fig. 3).

As duas partes de *tutorials* consideradas acima junto com os materiais complementares disponíveis da Celoxica [13] e os projectos desenvolvidos anteriormente (tais como [7-9] acessíveis através da WebCT [1]) provêm uma ajuda muito significativa para os alunos.



Actualmente, apenas a primeira parte dos *tutorials* está parcialmente concluída. A segunda parte será posta à disposição dos alunos no ano lectivo de 2003/2004.

## V. PROJECTOS

Os projectos são muito importantes para a área considerada. Para estimular este tipo de actividade foram propostos aos alunos dois tipos de avaliação. O primeiro é o exame tradicional. Contudo é permitido substituir o exame pelo segundo tipo de avaliação que consiste num projecto individual que é dado em meados do semestre respectivo. O projecto resume-se ao desenvolvimento, implementação e verificação de um sistema digital baseado em FPGAs disponíveis comercialmente. Normalmente, são propostos vários tipos de projectos, possibilitando escolher um projecto da área que representa o maior interesse para um aluno particular. Os resultados dos projectos devem ser periodicamente discutidos (com a demonstração de funcionamento dos sistemas construídos) e antes do fim do período de exames tem de ser entregue um relatório de execução. Os melhores projectos são recomendados para publicação na revista “Electrónica e Telecomunicações” editada pelo DET. Os artigos [2-11] são exemplos deste tipo de publicações. Alguns projectos estão disponíveis na WebCT [1] para serem consultados por outros alunos. Isto permite aproveitar os circuitos desenvolvidos e os seus componentes em projectos futuros possibilitando os alunos a construírem sistemas de complexidade maior.

Esta metodologia é praticada desde 1997 e possui vantagens bem como desvantagens. A vantagem principal é a oportunidade de os alunos estarem mais próximos ao trabalho prático real. A maior desvantagem surge para os orientadores pois devem ao mesmo tempo discutir e auxiliar a execução de muitos projectos diferentes. Isto faz com que sejam empregados esforços maiores e seja gasto mais tempo do que na avaliação tradicional. A experiência mostra que esta técnica de avaliação pode ser aplicada se o número de alunos (projectos) por cada orientador não exceder 20 e que os melhores resultados podem ser atingidos se este número for igual a 10. Se o número de alunos ultrapassar 20, torna-se razoável praticar tais métodos de avaliação apenas para os estudantes melhores seleccionados com base nos resultados dos testes intermédios e das aulas práticas.

A outra vantagem é a seguinte. A metodologia adoptada dá uma oportunidade muito importante, em particular para os alunos finalistas, pois possibilita uma integração de conhecimentos adquiridos em disciplinas diferentes. Os projectos podem ser seleccionados das várias áreas tais como sistemas embutidos, aplicações em medicina, telecomunicações, dispositivos periféricos, processamento de sinal e imagem, etc. Todas estas direcções são abordadas em disciplinas relevantes leccionadas no DET. Portanto, a integração torna-se natural. As disciplinas da computação reconfigurável sugerem e ensinam métodos e ferramentas, enquanto outras disciplinas fornecem aplicações possíveis. Sendo assim, aparece uma

motivação adicional porque sistemas reconfiguráveis ficam relacionados com algum trabalho prático oferecido por outras disciplinas, em que um dado aluno está particularmente interessado.

É também muito importante que a metodologia considerada é de grande utilidade para os projectos do fim de curso. De facto, sistemas digitais para várias aplicações podem ser desenvolvidos, implementados e testados rapidamente com base em FPGAs e outros dispositivos reconfiguráveis. Se os alunos adquirirem a experiência suficiente em tópicos diferentes do projecto de hardware e forem familiares com as ferramentas CAD relevantes, então poderão aplicar os seus conhecimentos em muitas áreas que são vitais para licenciaturas assim como Licenciatura em Engenharia Electrónica e Telecomunicações e Licenciatura em Engenharia de Computadores e Telemática, para construir protótipos de várias soluções técnicas, comparar a sua eficácia, estimar os recursos de hardware necessários, validar arquitecturas inovadoras, verificar características temporais, etc. É de salientar que só os dispositivos reconfiguráveis permitem realizar esta oportunidade em curto prazo (que varia normalmente de poucos dias para circuitos simples a um mês para sistemas bastante complexos).

Os exemplos de requisitos para os projectos de alunos podem ser encontrados em [28, 29]. Dado que se assume que todos os projectos (ou apenas os melhores) estão disponíveis na WebCT para alunos futuros, deve-se inventar cada ano tarefas diferentes. Na secção de introdução mencionámos que o domínio de sistemas reconfiguráveis é muito dinâmico e vasto. O poder e as capacidades funcionais de dispositivos reconfiguráveis disponíveis no mercado aumentam muito rapidamente. Tomando em consideração estas características não é muito difícil modernizar periodicamente os projectos. Isto não exclui a possibilidade de reutilizar os resultados anteriores. A maioria dos componentes desenvolvidos foram descritos numa HDL ou numa linguagem de especificação a nível de sistema. Portanto, torna-se possível reutilizar os códigos respectivos em projectos novos fazendo apenas modificações triviais. Mas mesmo para efectuar alterações minúsculas é necessário empenhar esforços para compreender muito bem o código. Portanto, este processo constitui uma parte importante da aprendizagem.

Os seis melhores projectos de alunos que foram concluídos no segundo semestre do ano lectivo de 2002/2003 (realizados no âmbito da disciplina “Computação Reconfigurável”) estão disponíveis na WebCT [1] e os seus resultados são publicados em [8-11] (de facto, o artigo [11] sumaria os resultados dos três projectos individuais cada um dos quais está acessível via [1]). Dois projectos [8, 9] foram desenvolvidos em Handel-C (no ambiente DK1) e os quatro restantes [10-11] – em VHDL (no ambiente ISE 5.2).

Na WebCT [1] estão também o código Handel-C e os programas C++ para três sistemas reconfiguráveis desenvolvidos por dois alunos (P. Almeida e M. Almeida) no âmbito do projecto do fim de curso no ano lectivo de

2002/2003. O ficheiro *Final\_Year\_Project.zip* inclui três directórios (nomeadamente, *RC100-Calc*, *RC100-Matrix*, *VIRTEXIIPRO*) que contêm respectivamente:

- um circuito aritmético simples que foi projectado só para ganhar experiência em Handel-C (veja [7] para detalhes);
- dois aceleradores combinatórios (um foi desenvolvido para a placa RC100 da Celoxica com a FPGA XC2S200 da família Spartan-II, e o outro foi implementado na placa ADM-XPL da Alpha Data com a FPGA XC2VP7 da família Virtex-II Pro) que permitem descobrir a cobertura mínima de uma matriz booleana. Ambos os circuitos realizam o algoritmo descrito em [30]. Os resultados deste projecto são discutidos em [21, 31].

## VI. CONCLUSÕES

Neste artigo descrevemos a metodologia pedagógica que combina métodos originais e ferramentas novas e que foi aplicada no processo educativo no âmbito das disciplinas relacionadas com computação reconfigurável e sistemas digitais avançados. Os métodos adoptados incluem: leccionação com a ajuda dos *tutorials* animados; concessão aos alunos de exemplos reutilizáveis que reflectem muitas propriedades distintas do projecto de hardware; combinação dos modelos de software e de hardware para demonstrar a funcionalidade complexa; e avaliação através dos projectos individuais que requerem a implementação e a validação dos circuitos desenvolvidos em hardware reconfigurável. As ferramentas abrangem os *tutorials* animados e um conjunto continuamente expansível de núcleos de propriedade intelectual tais como projectos, bibliotecas de software e de hardware, módulos, modelos (*templates*) reutilizáveis, etc.

## REFERÊNCIAS

- [1] <http://webct.ua.pt>, "2º semestre", disciplina "Computação Reconfigurável".
- [2] V. Silva, F. Santos, V. Sklyarov, "A interligação do Visual C++ com as FPGAs da XILINX", *Electrónica e Telecomunicações* (E&T), Jan., vol. 2, n. 7, 2000, pp. 874-883.
- [3] J. Santos, N. Duarte, "Síntese e Implementação de Circuitos Digitais Reconfiguráveis Dinamicamente (Projecto 1)", *E&T*, vol. 3, n. 8, Jan. 2003, pp. 720-728.
- [4] D. Gomes, N. Carvalho, "Síntese e Implementação de Circuitos Digitais Reconfiguráveis Dinamicamente (Projecto 2)", *E&T*, vol. 3, n. 8, Jan. 2003, pp. 729-732.
- [5] C. Teixeira, J. Girão, "Síntese e Implementação de Circuitos Digitais Reconfiguráveis Dinamicamente (Projecto 3)", *E&T*, vol. 3, n. 8, Jan. 2003, pp. 733-737.
- [6] J.P. Barraca, N.J. Sénica, "Síntese e Implementação de Circuitos Digitais Reconfiguráveis Dinamicamente (Projecto 4)", *E&T*, vol. 3, n. 8, Jan. 2003, pp. 738-742.
- [7] P. Almeida, M. Almeida, "Desenvolvimento de um circuito aritmético a partir da sua especificação em Handel-C", *E&T*, Vol. 3, n. 8, Jan. 2003, pp. 765-769.
- [8] B. Pimentel, "Utilização da Linguagem Handel-C na Criação de um Gap Puzzle", *E&T*, vol. 3, n. 9, Sep. 2003 (nesta revista).
- [9] J. Arrais, "Desenvolvimento de um circuito em Handel-C para experiências com máquinas de estados finitos", *E&T*, vol. 3, n. 9, Sep. 2003 (nesta revista).
- [10] B. Pereira, "Desenvolvimento de um circuito para operações sobre vectores booleanos e ternários", *E&T*, vol. 3, n. 9, Sep. 2003 (nesta revista).
- [11] R. Costa, J. Limas, I. Oliveira, "Desenvolvimento de uma calculadora baseada numa FPGA e num touch panel", *E&T*, vol. 3, n. 9, Sep. 2003 (nesta revista).
- [12] Spartan-II Development Platform, [Online]. Available: [www.trenz-electronic.de](http://www.trenz-electronic.de).
- [13] Handel-C, DK1, RC100, RC200 and other tools of Celoxica. Available: <http://www.celoxica.com/>.
- [14] XSA Board V1.1, V1.2 User Manual, 2002: <http://www.xess.com/>.
- [15] V. Sklyarov, I. Skliarova, "Ferramentas para desenvolvimento de sistemas digitais reconfiguráveis", *E&T*, vol. 3, n. 8, Jan. 2003, pp. 743-764.
- [16] Alpha Data, [Online]. Available: <http://www.alpha-data.com>.
- [17] ISE 5.2, Xilinx series FPGA. Available: <http://www.xilinx.com/>.
- [18] SystemC. Available: <http://www.systemc.org/>.
- [19] ModelSim. Available: <http://www.model.com/support/documentation.asp>.
- [20] Electronic Assembly products, [Online], Available: <http://www.lcd-module.de>.
- [21] V. Sklyarov, I. Skliarova, P. Almeida, M. Almeida, "Design Tools and Reusable Libraries for FPGA-Based Digital Circuits", *Proc. Euromicro Symposium on Digital Systems Design*, Antalya, 2003.
- [22] V. Sklyarov, "Reconfigurable models of finite state machines and their implementation in FPGAs", *Journal of Systems Architecture*, 47, 2002, pp. 1043-1064.
- [23] V. Sklyarov, I. Skliarova, "Design of Digital Circuits on the Basis of Hardware Templates", *Proc. Int. Conference on Embedded Systems and Applications - ESA'03*, Las Vegas, USA, 2003, pp. 56-62.
- [24] V. Sklyarov, "Hierarchical Finite State Machines and Their Use for Digital Control", *IEEE Transactions on VLSI*, Vol. 7, No 2, June, 1999, pp. 222-228.
- [25] B.W. Kernighan, D.M. Ritchie, "The C Programming Language", Englewood Cliffs, NJ: Prentice-Hall, 1988.
- [26] V. Sklyarov, I. Skliarova, A. Oliveira, A. Ferrari, "A Dynamically Reconfigurable Accelerator for Operations over Boolean and Ternary Vectors", *Proc. Euromicro Symposium on Digital Systems Design*, Antalya, 2003.
- [27] V. Sklyarov, "Modelação em C++, Síntese e Implementação de Circuitos Digitais com base em FPGA", *E&T*, vol. 3, n. 5, Jan. 2002, pp. 409-420.
- [28] V. Sklyarov, "Hardware/Software Modeling of FPGA-based Systems", *Parallel Algorithms and Applications*, ISSN 1063-7192, vol. 17, n. 1, pp. 19-39, 2002.
- [29] V. Sklyarov, "Síntese e Implementação de Circuitos Digitais Reconfiguráveis Dinamicamente", *E&T*, vol. 3, n. 8, Jan. 2003, pp. 713-719.
- [30] V. Sklyarov, I. Skliarova, "Architecture of a Reconfigurable Processor for Implementing Search Algorithms over Discrete Matrices", *Proc. Int. Conference on Engineering of Reconfigurable Systems and Algorithms - ERSA'03*, Las Vegas, USA, 2003, pp. 127-133.
- [31] V. Sklyarov, I. Skliarova, P. Almeida, M. Almeida, "High-Level Design Tools for FPGA-Based Combinatorial Accelerators", *Proc. Int. Conference on Field-Programmable Logic and Applications - FPL'03*, Lisbon, 2003.