Síntese e Implementação de Circuitos Digitais Reconfiguráveis Dinamicamente (Projecto 3)

Cláudio Teixeira e João Girão

***Resumo* – Este artigo apresenta os resultados do projecto proposto aos autores (alunos do 4.º ano da LECT) e explica como o problema especifico foi resolvido. A descrição do projecto e os requisitos básicos foram considerados anteriormente nos artigos [1, 2].**

***Abstract* – The paper presents the results of the project proposed to the authors (who are the 4th year students of LECT) and shows how the specified problem has been solved. The description of the project and the basic requirements have been considered in the papers [1,2].**

I. INTRODUÇÃO

No âmbito da cadeira de Computação Reconfigurável aprendemos que os circuitos digitais apresentados em cadeiras anteriores poderiam passar a um novo nível de complexidade e tratamento na integração de circuitos reconfiguráveis até por software. Esta dinâmica permitiu-nos descobrir um novo paradigma na programação de circuitos assim como novos conceitos em linguagens de descrição de hardware.

A facilidade na criação de novos circuitos utilizando FPGA's possibilita o tratamento de certas funções por parte de entidades externas apresentando depois os resultados do cálculo. Tais funções, sendo tratadas externamente, permitem ao processador estar ocupado com outras actividades.

Um maior facilidade no caso anterior seria a possibilidade de reprogramar o circuito que se encontra externamente para tratar os dados de forma diferente consoante o objectivo pretendido. Tal pode ser conseguido através do uso de uma máquina de estados finitos reprogramável e com a ajuda de outros componentes de controlo e de execução.

Uma Máquina de Estados Finitos Reprogramável (MEFR) não é mais que um circuito implementado com base em RAM que nos permite a descrição de uma máquina de estados finitos com as limitações apresentadas pelos níveis e profundidade da RAM.

Juntamente com a MEFR existe, como foi acima descrito, um conjunto de blocos que nos permitem o controlo, reprogramação e introdução de dados para execução na MEFR. Nestes estão incluídas duas partes de extrema importância no processo: a Unidade de Controlo (UC) e a Unidade de Execução (UE).

A UC está ligada à forma como são executadas as operações na FPGA. Garante a conformidade dos dados e permite a reprogramação da MEFR. Além disto, permite a leitura e escrita dos dados na UE que acabam por ser os dados sobre os quais a MEFR vai actuar.

A UE supervisiona e permite um conjunto de operações externas à MEFR. Fornece os vectores sobre os quais esta vai actuar, assim como a forma como os resultados são apresentados.

Especifico do trabalho realizado, foi garantido no projecto um conjunto de possibilidades além das referidas anteriormente.

Entre estas, o facto da MEFR poder ser executada concorrentemente com a reprogramação da mesma (isto, porém, pode trazer resultados inesperados na execução da mesma).

Como dados de entrada na MEFR considerámos dois vectores de tamanho máximo 24 bits que funcionam de forma independente e podem ser acedidos de forma continuada (ao chegar ao fim do vector, um novo incremento no índex do mesmo resulta no retorno à posição de início e a uma flag de final de vector ser accionada). Ambos estes vectores e controlo sobre os mesmos são parte da UE.

Ainda como parte da MEFR e da UC, um conjunto de operações permite-nos ainda confirmar e ler o conteúdo das RAM's de forma a confirmar ou comparar o conteúdo das mesmas. Estas operações facilitam o debugging que pode ser agora apenas a nível de software.

Finalmente, é possível reprogramar a própria UC de forma a maximizar a actuação da mesma sobre todo o circuito. Isto foi conseguido através da utilização de uma MEFR para implementação da UC.

II. ESPECIFICAÇÃO DO PROJECTO

Baseando-se na área anteriormente referida (ver também [3,4]), o projecto tinha como objectivo a implementação de uma MEFR que interagisse com o PC de forma a implementar diferentes algoritmos. Os dados seriam passados do PC para a placa e depois accionado o mecanismo produzindo assim os resultados que seriam depois novamente enviados para o PC.

Diversos blocos foram considerados de forma a montar as peças que iriam depois formar a MEFR. Esta

aproximação permitiu a decomposição do problema em subpartes mais simples que foram. A interacção com as aulas foi feita de forma a que os blocos acima referidos fossem analisados de forma independente e estudada a posição dos mesmos no projecto global.

O projecto consistia de facto em três partes:

Na primeira utilizou-se software proprietário para projectar os blocos principais. Através destas ferramentas o processo de programação da FPGA é facilitado sendo possível desenhar os blocos esquematicamente ou até mesmo utilizando linguagens de descrição de hardware e depois implementá-los na FPGA como o circuito final.

Uma segunda fase passava por desenhar o interface da MEFR com o PC. Visto a comunicação entre o PC e a FPGA ser limitada, é necessária lógica adicional para permitir a multiplexagem dos bits de entrada e saída assim como a utilização de buffers intermediários para guardar os dados.

Finalmente, uma última, cujo produto final era o software do PC que permite que todos estes passos sejam utilizados de forma simples e fácil. É importante que neste ponto o utilizador final não precise de saber nada sobre a implementação da máquina. Isto será transparente utilizando a interface por software.

III. ARQUITECTURA BÁSICA

Na fig. 1 podemos observar os diversos blocos que foram construídos e que constituem todo o projecto. Deles faz parte a interface entre a MEFR e o PC, a MEFR em si, a unidade de execução que alimenta os dados e controla algumas operações da MEFR e a unidade de controlo que permite a reprogramação e controlo da MEFR e UE.

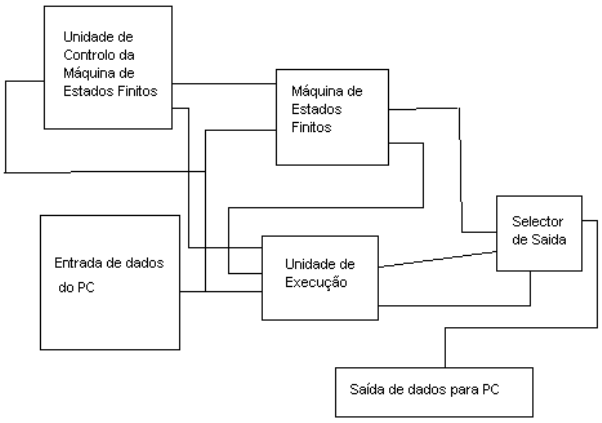

Fig. 1- Estrutura geral do sistema implementado

Alguns dos blocos terminais permitem a visualização de dados contidos em pontos estratégicos do projecto. Isto facilita o debugging e permite um feedback sobre o estado do circuito.

Faz ainda parte deste desenho toda a interface entre a FPGA e o PC.

Embora não seja diferenciável pelo desenho, alguns blocos foram desenhados num editor esquemático e outros utilizando uma linguagem de descrição de hardware chamada VHDL.

Os blocos principais serão descritos de forma mais aprofundada seguidamente.

IV. SÍNTESE, MODELAÇÃO E IMPLEMENTAÇÃO DO SISTEMA

A. Interface

A interface com o PC é dada por 4 bits de escrita na porta paralela para o PC e 8 de leitura na FPGA provenientes do PC pela mesma porta paralela.

Dos 8 bits de leitura apenas os 4 menos significativos são utilizados como dados. O bit mais significativo é utilizado para fazer Reset a todos os blocos e consequentemente a todo o circuito e o bit seguinte é utilizado para informar a FPGA de que existem dados novos. É utilizado um sinal de relógio para este propósito.

Para escrita para o PC existe um registo de 8 bits onde os dados são guardados para posteriormente serem enviados para o PC em blocos de 2 bits. Os outros 2 da comunicação placa – PC são utilizados para manter o sincronismo com o PC através de contador que, no seu primeiro valor, informa o PC de que existem dados novos.

Através desta comunicação é possível reprogramar a MEFR, executar, carregar vectores, etc., utilizando apenas o protocolo definido pela Unidade de Controlo para o efeito pretendido. Além disto é possível receber dados no PC da FPGA tais como o resultado do tratamento de um vector ou a leitura de uma posição de memória para verificação do algoritmo carregado.

B. Unidade de execução

Permite escrever quer o tamanho dos vectores, quer os vectores que podem ser usados para alimentar a MEFR. Tem como função principal alimentar a MEFR. A *Fig. 2* mostra o desenho da UE e a tabela 1 e tabela 2 explicam o significado e propósito de cada uma das entradas e saídas da UE.

Em seguida segue uma descrição do que está representado na Tabela 1.

DI é o barramento que contém 8 bits de informação introduzida via porta paralela. O bloco PC_BUFF4_8BIT contém lógica que permite ao PC escrever nos 4 bits mais ou menos significativos. Uma flag enviada pelo PC, cujo significado é “fim de introdução” permite a este bloco disponibilizar os novos 8 bits de informação.

Y é o barramento que contém os valores calculados pela MEFR. É com base nestes valores que a UE decide qual a próxima instrução a executar. Nos bits de entrada podemos ver o tipo de acções que podemos realizar na nossa UE (o BUS de entrada Y). Estas incluem acções como incrementar a posição de leitura dos vectores, incrementar o vector de resultado, fazer o vector de resultado igual a 0 e até escrever valores directamente neste vector.

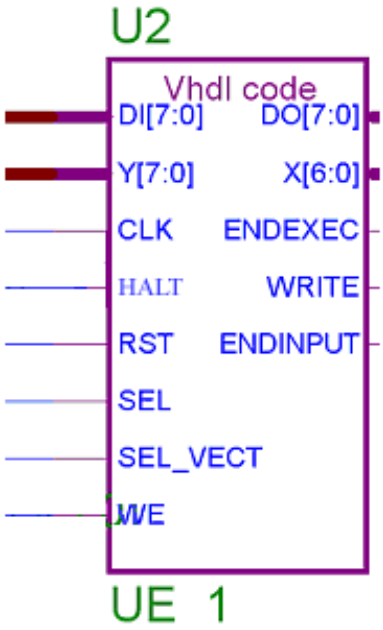

Fig. 2 - Unidade de Execução

HALT é uma flag que permite a paragem momentânea do processamento.

Uma vez que o tamanho do vector de entrada pode ser especificado, SEL faz a agulhagem dos dados para um dos modos.

Como a MEFR possui dois vectores de dados, é necessário especificar a qual deles se referem os novos dados. Esta é a função de SEL_VECT.

Nome	Descrição	Origem
DI	Dados completos de entrada via porta paralela	PC_BUFF4_8BIT
Y	Saídas da MEFR	MEFR
HALT	Permite parar a UE	Porta Paralela
RST	Reset da UE	Porta Paralela
SEL	Selecciona a que se referem os dados introduzidos: tamanho ou vector	Unidade de Controlo
SEL_VECT	Selecciona um de dois vectores possíveis para dados	Unidade de Controlo
WE	Permite a escrita nos registos da UE	Unidade de Controlo

Tabela 1- Entradas da UE

Nome	Descrição	Destino
DO	Resultado das operações da MEFR	OUTPC
X	BUS de entradas na MEFR	MEFR
ENDEXEC	Flag de fim de execução	
WRITE	Selecciona o resultado da UE para entregar a OUTPC	OUTPC
ENDINPUT	Flag de fim de introdução do vector	Unidade de Controlo

Tabela 2 - Saídas da UE

Em seguida segue uma descrição do que está representado na tabela 2.

DO é um barramento cuja finalidade é a de possibilitar o envio do resultado das operações na MEFR para o PC.

X é o bus com os novos valores para a MEFR processar.

WRITE é uma flag que desencadeia o processo de envio para o PC do resultado DO.

C. Unidade de controlo

A UC (ver fig. 3) não é mais que uma MEFR com o conteúdo necessário ao controlo de todas as operações relativas ao circuito.

Como se pode verificar pela fig. 4 distinguimos 4 operações distintas:

Program:

Reprogramação de endereços de memória.

Nesta operação podemos indicar qual o bloco de RAM, o endereço e os dados que pretendemos escrever na mesma de forma a reprogramar aquela célula.

Data:

A funcionalidade desta operação está interligada com a inserção de dados para depois serem utilizados pela MEFR. Primeiro é introduzido o tamanho do vector a passar e só depois é que se introduz a informação relativa ao vector. As interacções utilizam 4 dos 8 bits de comunicação para questões de controlo, implicando então que apenas se podem introduzir 4 bits de informação por ciclo de relógio da FPGA.

Run:

Com este simples comando podemos accionar o funcionamento da MEFR e da UE. O resultado da

execução será enviado para o PC assim que a mesma terminar.

Read:

Finalmente podemos verificar o conteúdo das RAM's da MEFR utilizando este comando. O conteúdo da célula é devolvido ao PC no fim desta operação via bloco OUTPC.

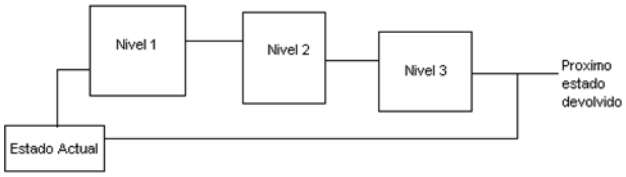

Fig. 3- Unidade de Controlo

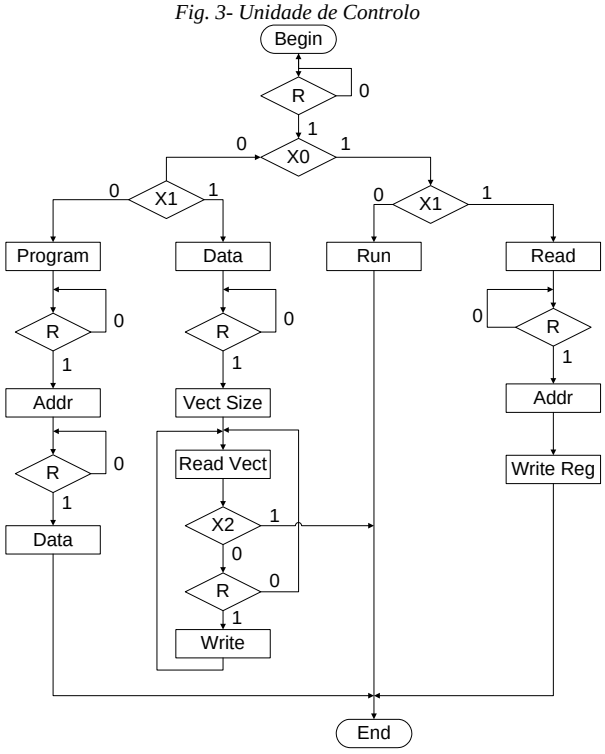

Fig. 4 Diagrama de Estados da UC

Nome	Descrição	Origem
X0	É o bit mais significativo da selecção da instrução que a unidade de controlo deve executar.	Porta Paralela
X1	É o bit menos significativo da selecção da instrução que a unidade de controlo deve executar.	Porta Paralela
X2	Bit de controlo de fim de leitura do vector de dados a introduzir	Unidade de Controlo
R	Bit de controlo que permite à UC continuar a execução pendente de uma leitura	CONTROL_FLAG

Tabela 3- Entradas da UC

Nome	Descrição	Destino
Y0	Escreve conteúdo do Buffer à entrada da placa para o registo de endereços	REG4 * 2
Y1	Selecciona a que se referem os dados introduzidos: tamanho ou vector	Unidade de Execução
Y2	Write enable	Unidade de Execução
Y3	Não Usado	
Y4	Selecciona um de dois vectores possíveis para dados	Unidade de Execução
Y5	Permite a escrita do registo do resultado para o PC	OUTPC
Y6	Run flag	MEFR
Y7	Acknowledge para o PC de que pode introduzir novos dados	OUTPC

Tabela 4- Saídas da UC

D. Reconfiguração

Da forma como a MEFR foi implementada [5], é possível reprogramar byte a byte as memórias que contêm as tabelas e que definem o algoritmo a utilizar [1]. Ainda para mais, visto a Unidade de Controlo ser implementada sobre uma MEFR, é possível mesmo reprogramar esta de forma a actuar de forma diferente sobre os dados do PC.

O PC deve seleccionar primeiro a operação de programação, depois indicar o endereço de escrita que inclui o endereço da memória e a indicação da memória a escrever e finalmente indicar o valor dos dados a escrever. (ver fig. 4)

O processo deve ser repetido até concluir todas as alterações desejadas.

Um conjunto de registos internos garante que os dados que são passados à FPGA de forma sequencial estejam presentes quando for a altura de escrever o valor na memória.

E. A parte de software

O software criado passou pela criação de uma API de interface que consiste dum conjunto de funções para enviar e receber dados da placa.

Este conjunto de funções torna transparente ao programador da aplicação final o protocolo assumido pela placa.

Existem funções para reprogramar um byte de memória da MEFR, carregar vectores na máquina, para ler um byte de uma memória ou o resultado da última operação da máquina.

Através destas funções torna-se simples de escrever ou integrar estas funcionalidades em qualquer aplicação.

O projecto que inclui todas as partes consideradas anteriormente está apresentado no WebCT [3].

V. IMPLEMENTAÇÃO DE OPERAÇÕES SOBRE VECTORES BOOLEANOS

Uma operação de aplicação prática seria a utilização da FPGA para cálculo do operador RCROS sobre dois pontos de uma imagem de forma a calcular as arestas da imagem de 2 em 2 pontos.

Isto é possível ou aumentando o tamanho dos registos de forma a poderem conter imagens, ou alimentando a FPGA com apenas 2 pontos seleccionados do lado do PC.

Visto esta ser uma operação pesada, podemos concluir que o facto de estar a ser calculada externamente ao PC liberta o processador do mesmo para outras operações.

Também parece óbvia uma aplicação para tratamento de imagem em robótica visto a FPGA ser de pequenas dimensões e este um algoritmo específico.

VI. CONCLUSÕES

Nesta cadeira foi possível pela primeira vez criar circuitos complexos e vê-los funcionar de forma rápida e segura. Embora Sistemas Digitais apresente um apoio necessário para a cadeira, falha talvez na sua componente prática. Esta cadeira, como continuação e métodos avançados da cadeira anteriormente descrita, aumenta o contacto com os circuitos digitais e apresenta o novo rumo da tecnologia. Visto que o curso tem uma orientação não tanto para o hardware, é essencial que esta cadeira permita ao aluno ser capaz de projectar pequenos circuitos que sejam implementáveis em FPGA e desenhados a alto nível. Nisto a cadeira mais que satisfez estes requerimentos dando aos alunos ambos os conteúdos teóricos e teoria fundamental assim como a prática com esta tecnologia.

Referências

[1] V.Sklyarov, Síntese e Implementação de Circuitos Digitais Reconfiguráveis Dinamicamente, Electrónica e Telecomunicações, Vol. 3, Nº 8, Jan. 2003 (nesta revista).

[2] Johnny Santos e Nuno Duarte, Síntese e Implementação de Circuitos Digitais Reconfiguráveis Dinamicamente, Electrónica e Telecomunicações, Vol. 3, Nº 8, Jan. 2003 (nesta revista).

[3] Página *http://webct.ua.pt*, "2º Semestre", a disciplina "Computação Reconfigurável".

[4] Página *http://webct.ua.pt*, "2º Semestre", a disciplina "Paradigmas de Programação *I*".

[5] V. Sklyarov, Reconfigurable models of finite state machines and their implementation in FPGAs. Journal of System Architeture, 47, 2002, pp. 1043-1064.