

Estudo de Metodologia para a Implementação de Processadores Usando FPGA – Aplicação para o Processador Acadêmico FIX-1

Mairum Ceoldo Andrade & José Hiroki Saito

UFSCar – Universidade Federal de São Carlos
Centro de Ciências Exatas e de Tecnologia - Departamento de Computação
Rod. Washington Luis, Km 235 - 13565-905 - São Carlos – SP

mairum@comp.ufscar.br, saito@dc.ufscar.br

Abstract. *This work is concerned to the study of a methodology for implementation of reconfigurable circuits, using FPGA devices (Field Programmable Gate Array). For this it was implemented, using the VHDL language, an academic 16-bit processor, called FIX-1. Two methodologies of design had been adopted, being the first one, the full design of all the functional elements of the computer; and the second one, based on the use of parametric megafuntions. For FPGAs devices of reduced size, only the second methodology was viable, which demonstrates the efficiency of the megafuntions.*

Keyword. *Academic Processor, FPGA, VHDL.*

Resumo. *O presente trabalho tem por objetivo o estudo de metodologia para implementação de circuitos reconfiguráveis, usando componentes do tipo FPGA (Field Programmable Gate Array). Para isso implementou-se, utilizando a linguagem VHDL, um processador acadêmico de 16 bits, denominado FIX-1. Foram adotadas duas metodologias de projeto, sendo a primeira, o projeto completo de todos os elementos funcionais do computador; e a segunda, baseada no uso de megafunções parametrizadas. Para componentes FPGAs de tamanho reduzido, somente a segunda metodologia foi viável, o que demonstra a eficiência das megafunções.*

Palavras Chaves. *Processador acadêmico, FPGA, VHDL.*

1. Introdução

Neste trabalho é descrito o projeto de um processador acadêmico FIX-1 [SAI,00], com características semelhantes ao TM-16, proposto por Langdon Jr. [LAN, 85]. O projeto consiste em editar em VHDL [PER, 94] [ASH,90] todos os componentes e interconexões do computador, e realizar a sua simulação, corrigindo os erros, e verificando a viabilidade de reconfiguração de um componente FPGA [TER, 98] para a sua implementação. Foram adotadas duas tentativas, sendo a primeira, a de projeto total do computador usando portas lógicas básicas, e a segunda, a de uso de megafunções parametrizadas pré-definidas no ambiente de desenvolvimento ALTERA, MAX+plus2 [ALT, 97]. O resultado da primeira metodologia permitiu o estudo do comportamento dos elementos funcionais individuais, através da compilação do código desenvolvido, e simulação, porém a integração de todos os componentes para a formação do computador não foi possível, devido ao tamanho máximo dos componentes FPGAs disponíveis no ambiente de desenvolvimento. Numa segunda metodologia, substituindo os elementos funcionais por megafunções pré-definidas

na biblioteca do ambiente MAX+plus2, foi possível a integração e teste de simulação do computador.

2. O FIX –1

Esta máquina denominada FIX –1 trata-se de um computador de 16bits com instruções fixas, com programas e dados armazenados na memória, e como muitas arquiteturas simples de computador, segue a arquitetura de instruções de endereço único. Isto implica na existência de cinco registradores padrão na CPU, quais sejam: *contador de programa* (PC), *registrador de endereço de memória* (MAR), *registrador de dados memória* (MBR), *registrador de instruções* (IR) e *acumulador* (AC); além do ponteiro de pilha (SP), somador (ULA), circuito de controle e uma memória (RAM) com 12 bits de endereçamento (Fig.1).

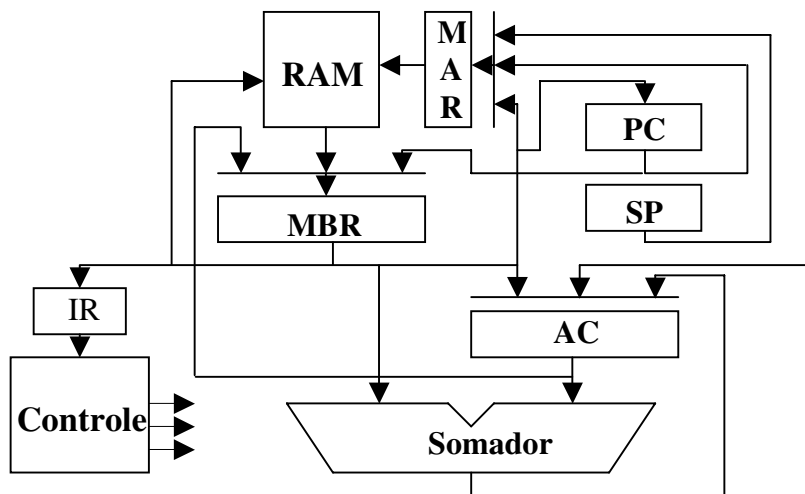


Figura 1. Diagrama de blocos simplificados do FIX –1.

3. Conjunto de Instruções

O processador FIX-1 possui um total de 9 instruções fixas. A Tabela 1 mostra essas instruções em quatro colunas: mnemônico, código binário, descrição e operações.

Tabela 1: Instruções do FIX – 1

Mnemônico	Binário	Descrição	Operações
LODD	0000	Carga direta	AC := M[x]
STOD	0001	Armazena direto	M[x] := AC
ADDD	0010	Soma direta	AC := AC + M[x]
JPOS	0011	Desvia se AC positivo	Se AC >= 0 então PC := x
JUMP	0100	Desvio incondicional	PC := x
CALL	0101	Chama procedimento	SP := SP-1; M[SP] := PC; PC := x
RETN	0110	Retorna	PC := M[SP]; SP := SP + 1
INPT	0111	Entrada	AC := entrada
OUTP	1000	Saída	saída := AC

O formato das instruções é único sendo 4 bits de código de operação, e 12 bits de endereço de operando. A maior instrução é composta por 8 ciclos de clock, sendo que 4 ciclos são de busca de instrução e 4 de execução.

4. Modo de Implementação do Projeto

Na implementação do projeto foi utilizado o software ALTERA MAX+plus II, utilizando-se a linguagem VHDL em modo completamente estrutural, utilizando-se apenas de portas lógicas e megafunções predefinidas.

Inicialmente a implementação utilizou-se somente de portas lógicas, desde os flip-flops, passando pelos registradores, contadores, multiplexadores, somador, circuito de controle e memória RAM. Este tipo de implementação enfrentou muitos problemas devido aos tamanhos de FPGAs disponíveis no software. Nesta fase foram testados todos os componentes separadamente, constatando que todos funcionavam corretamente, mas quando da montagem e junção dos componentes o funcionamento era comprometido por estados indefinidos na saída da memória RAM, o que levava a uma interrupção do processamento.

Após muitos testes e tentativas, passou-se para a utilização de megafunções parametrizadas. Os registradores foram substituídos pela megafunção *lpm_ff* que permite definir o modo de funcionamento e o tamanho do registrador. Os contadores foram substituídos por *lpm_counter* que permite definir o modo de funcionamento, up, down ou up-down, e o tamanho. O somador foi substituído por *lpm_add_sub*, definindo o tamanho desejado (16 bits). A memória RAM foi substituída pela megafunção *lpm_ram_dq* que se trata de uma memória RAM assíncrona com entrada e saídas separadas. Os multiplexadores foram substituídos por *lpm_mux*, que é um multiplexador configurável. Todas essas megafunções são da biblioteca LPM que possui componentes parametrizáveis e adaptáveis às necessidades.

A unidade de controle continuou a ser implementada com a utilização de portas lógicas, mas seu circuito combinatório teve algumas mudanças, pois alguns sinais de controle dos componentes substituídos por megafunções são diferentes dos utilizados anteriormente.

Mesmo com a utilização das megafunções o problema de espaço na FPGA persistiu. Depois de muitos testes o problema foi resolvido diminuindo o tamanho da memória RAM de 4K para 256 palavras, ou seja, de 12 bits de endereçamento para 8 bits. O tamanho dos registradores e contadores foram mantidos como anteriormente, pois existe a possibilidade da utilização de uma memória externa, para a ocupação completa do espaço de endereçamento.

5. Resultados de Simulação

A Fig. 2 mostra um diagrama de tempo obtido através da simulação do circuito. O diagrama mostra os principais sinais do processador, podendo identificar duas fases de execução da simulação: 1) fase de carregamento de dados na memória do processador (CLOCK = 0, e Controle = 1, Contador Anel = 00), em que dois dados são introduzidos (nos endereços 00 e 20), através do acionamento do sinal de controle WE; esses dados correspondem à instrução LODD 20H, e o dado correspondente ao operando no endereço 20H; e 2) fase de interpretação da instrução (CLOCK = ativo, Controle = 0, Contador Anel = 01 .. 80); em que dois ciclos de máquina são visíveis: (a) ciclo de busca de instrução (Contador Anel = 00..08), onde MBR é carregado com 0020H e o valor de PC é incrementado, e (b) ciclo de execução da instrução (Contador Anel = 10..80), onde MBR é carregado com AAAAH e posteriormente o dado é transferido para AC.

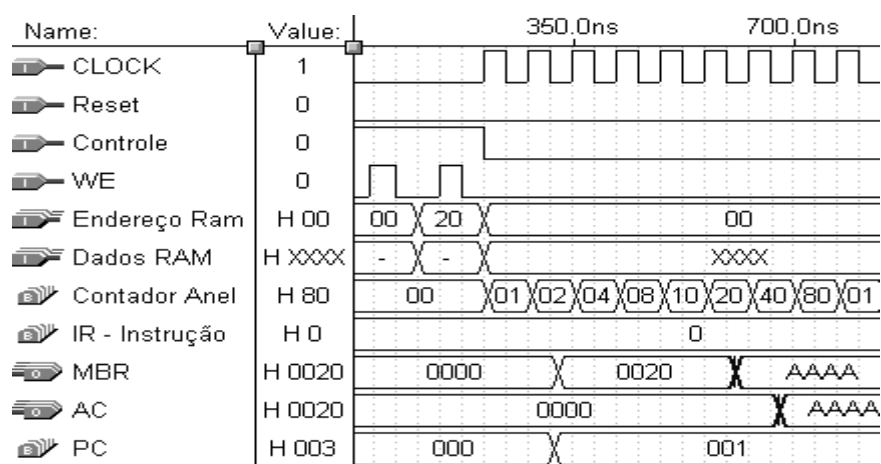


Fig. 2 – Exemplo da execução da instrução LODD 20H.

Todas as instruções foram devidamente implementadas e testadas através do simulador, onde foi possível verificar a eficiência do ambiente de desenvolvimento utilizado, e a validação do projeto desenvolvido.

6. Conclusões

Apesar dos problemas enfrentados todos foram contornados e foi possível chegar ao funcionamento desejado do projeto. Este projeto está sendo de grande utilidade para o aprendizado e estudo dos componentes ALTERA e da linguagem VHDL.

Para o futuro, pretende-se aperfeiçoar o projeto incorporando mais instruções ao processador, para que o mesmo possa ser usado num contexto maior de projeto, como no controle da arquitetura Multiprocessadora Reconfigurável Ortogonal, REOMP (Reconfigurable Orthogonal MultiProcessor), caracterizada pelo uso de diversas unidades de processamento paralelo, e um número quadraticamente maior de módulos de memória [SAI,99].

7. Referências Bibliográficas

- [ALT, 97] Altera Corporation “MAX+PLUS II – Programmable Logic Development System”. San Jose-CA, 1997.
- [LAN, 85] Glen G. Langdon Jr. “Projeto de Computadores”. Cartgraf, Campinas, 1985.
- [SAI, 99] J. H. Saito “A Vector Orthogonal Multiprocessor NEOMP and its Use in Neural Network Mapping”. Proceedings of the SBAC-PAD –11th Symposium on Computer Architecture and High Performance Computing, pp.271-278, Natal, Brasil, October 1999.
- [SAI, 00] J. H. Saito “FIX - 1 – Notas de Aula”. Universidade Federal de São Carlos, São Carlos-SP, 2000.
- [PER, 94] Douglas L. Perry “VHDL”. McGraw-Hill, USA, 1994.
- [TER, 98] Anderson Royes Terroso “Dispositivo Lógico Programável (FPGA) e Linguagem de Descrição de Hardware (VHDL)”. VII Semana da Engenharia PUCRS, Porto Alegre-RS, 1998.
- [ASH, 90] Peter J. Ashenden “The VHDL CookBook”. South Australia, 1990.