Implementação de Microprocessador em FPGA

Docente: Prof.º Dr.º Marcelo C. Tosin
Discentes: Guilherme Pessôa, Guilherme Brandão, Jean Sartor,
João Henrique, Karina Bernardin, Karla Nakamura - Turma 1012
Universidade Estadual de Londrina
Londrina, PR
26 de Janeiro de 2017

***Resumo*— O atual projeto propõe a síntese de microprocessador com arquitetura de 16 bits de instruções polifásicas e micro-programável em FPGA *(Field Programmable Gate Array)* utilizando o kit de desenvolvimento DE1-SoC da Terasic, que apresenta um chip FPGA da Altera. A descrição do *hardware* será feita utilizando VHDL (*Hardware Description Language*).**

I. Introdução

Atualmente os microprocessadores são essenciais à tecnologia, estando presentes nos mais variados tipos de aplicações tais como, *smartphones*, computadores, automóveis entre outras.

A grande utilização dos microprocessadores nos sistemas eletrônicos atuais vem do fato de serem programáveis, ou seja, estes sistemas são compostos por um *hardware* fixo orientado por uma sequência de instruções, proporcionando grande versatilidade e poder de processamento.

Devido à facilidade de desenvolvimento e versatilidade das aplicações, as FPGAs estão sendo cada vez mais difundidas para a realização de projetos. Suas aplicações variam desde implementação de circuitos lógicos simples a *hardwares* mais complexos, como periféricos de processadores e até mesmo microprocessadores.

A. Microprocessador

Concebido como peça fundamental de todo sistema digital de processamento, o microprocessador é construído sobre uma pastilha de silício de alta pureza, integrado e envelopado sob resina. Este dispositivo digital se caracteriza por receber dados digitais binários, processá-los e em seguida devolvê-los, isso tudo de forma controlada e programável.

Em suma grandes matemáticos do século passado traçaram os paradigmas do funcionamento de um microprocesssador, composto basicamente por: unidade lógica e aritmética, registradores e unidade de controle.

1) Unidade lógica e aritmética: Abreviada normalmente pelas suas inicias, ULA, constitui uma máquina de cálculo com números binários, realizando operações matemáticas como soma e subtração e operações logicas como AND e OR.

2) Registradores: Composto por Flip-Flops dispostos em paralelo e uma pequena gama de controles, são responsáveis por reter os dados no interior do microprocessador e, com as mais diversas tarefas atribuídas, dão o passo de qual forma os dados fluem.

3) Unidade de controle: Parte dedicada em decodificar os comandos e controlar as demais partes do microprocessador. Os atuais contam com uma estrutura micro programada em uma diminuta memoria não volátil, o pequeno arquivo nesta memoria se denomina microcódigo. Semelhante a uma lista ou tabela, este código dita os comandos que devem ser fornecidos.

B. FPGA

Field Programmable Gate Arrays (FPGAs), em português Arranjo de Portas Programável em Campo, são dispositivos semicondutores baseados em uma matriz de blocos lógicos (CLBs) conectados via interconexões programáveis.

As FPGAs podem ser reprogramadas para aplicações e funções desejadas, pois são constituídas por milhares de transistores e fazem basicamente o que vários circuitos integrados fariam, com a diferença de que sua matriz é reprogramável, como mencionado anteriormente.

Em geral, as FPGAs trabalham prioritariamente com as linguagens VHDL (VHSIC *Very High Speed Integrated Circuits Hardware Description Language* ou Linguagem de Descrição de Hardware de Circuitos Integrados de Velocidade Muito Alta) e o Verilog HDL. Para trabalhar em tais placas deve ser utilizada uma dessas duas linguagens de descrição de hardware, necessárias na programação e executação dos devidos códigos de configuração de hardware. De posse destas linguagens, é possível descrever sistemas embarcados ou ainda a camada de hardware para sistemas mais complexos.

No plano de pesquisa sobre o tema aparece o trabalho de Montenegro e Girardi (2009), que implementaram em VHDL um microprocessador conceitual, baseado no microcontrolador PIC da *Microchip*, possuindo instruções de 16 bits, onde 5 deles são utilizados como *op code* - com 8 instruções pré programadas - e o restante é direcionado para o endereçamento de variáveis na memória de dados, que apresenta 2kB de capacidade.

No trabalho desenvolvido por Pastor et al (2004) foi projetado um microprocessador em VHDL e implementado em um dispositivo FPGA e, posteriormente, foi realizada uma análise de seu comportamento com alguns programas de teste e uma ferramenta de *MicroDebug*, também por eles desenvolvida. O microprocessador projetado possui arquitetura Harvard com instruções de 16 bits, 4 registradores de uso

geral internos e 9 instruções básicas, como LOAD, STORE, MOVE, ADD, SUB, entre outras.

II. Objetivos

Este trabalho propõe a implementação de um Microprocessador de 16 bits em FPGA, utilizando-se a placa DE1-SoC da Altera e a linguagem de descrição de *hardware* VHDL.

Por meio dessa aplicação, o trabalho busca estudar o funcionamento de uma arquitetura microprogramável de um microprocessador básico e dos recursos da tecnologia FPGA, executando testes através de microoperações programadas e elaboração de documentação técnica.

III. Metodologia

A. Implementação do Microprocessador

A Figura 2 apresenta o diagrama funcional do microprocessador implementado em FPGA. O diagrama RTL é um modelo de abstração de circuitos digitais síncronos focado no fluxo dos sinais entre os registradores e as operações realizadas sobre eles. Portanto, observando o diagrama RTL é possível observar todos os sinais contidos no processador implementado e seu fluxo dentro dos componentes definidos no projeto.

Os blocos verdes da Figura 2 são componentes que desempenham funções específicas no microprocessador. A lista a seguir apresenta a função de cada um deles:

- **CLOCK_DIV:** divide o *clock* base do microprocessador, permitindo ajustar a frequência de execução das operações. O divisor de clock propicia a escolha de frequências suficientemente pequenas para a execução de testes e busca por erros no projeto.
- **CONTROLADOR_PRIN:** este bloco representa os controladores (principal e microprogramado) do microprocessador, executando as fases reset e de 1 a 5 e gerenciando as memórias principal e de microprograma. Seu funcionamento baseia-se na atualização das fases e na identificação da fase atual, realizando as operações entre barramentos, registradores e memória de acordo com os sinais de controle ativos.
- **DISPLAY_SEGS:** o bloco dos *displays* 7 segmentos foi criado como uma interface entre o usuário e os registradores e barramentos do processador. Foram configurados um total de 6 *displays*, que exibem os valores atuais dos barramentos de entrada e saída da ULA, a fase atual do controlador microprogramado e os registradores MPC e PC.

B. Registradores e Barramentos

A primeira etapa do desenvolvimento do microcontrolador foi a definição dos registradores e barramentos internos e externos da máquina. Para a sua criação, foram utilizados sinais e variáveis, que são estruturas do VHDL que armazenam informações. Um exemplo de definição de sinal é a criação do sinal equivalente ao registrador MPC:

```
SIGNAL MPC: STD_LOGIC_VECTOR(9 DOWNTO 0);
```

Na definição é fornecido um nome para o sinal, seu tipo e tamanho, já que o sinal em questão se trata de um vetor de 10 posições. Vale ressaltar que da maneira como foi definido, *MPC(9)* seria o bit mais significativo do vetor, enquanto *MPC(0)* carrega o bit menos significativo.

A declaração de uma variável é semelhante à de um sinal, seguindo a seguinte sintaxe para a definição do registrador *R1*:

```
VARIABLE R1: STD_LOGIC_VECTOR(15 DOWNTO 0);
```

Os sinais e variáveis foram definidos como vetores de bits, do tipo *STD_LOGIC*, com o tamanho segundo o projeto da arquitetura da máquina. Além disto, os registradores que são necessários em diversas partes do código do processador foram definidos como globais. A Tabela I contém a lista completa dos registradores implementados, apresentando seu nome, seu tamanho e sua função no processador.

TABLE I
LISTA DOS REGISTRADORES IMPLEMENTADOS NO PROCESSADOR.

Nome	Tamanho (bits)	Função
MPC	10	Contador de microprograma
SC	24	Registrador dos sinais de controle
IR	16	Registrador de instrução
PC	16	Contador de programa
REM	16	Registrador de endereços da memória principal
RDM	16	Registrador de dados da memória principal
R1	16	Registrador auxiliar
R2	16	Registrador auxiliar

Para a criação dos barramentos também foi escolhida a utilização de sinais. Foram definidos sinais com a mesma dimensão dos barramentos no projeto do processador. A Tabela II apresenta os barramentos implementados, acompanhando seu nome, tamanho e função.

TABLE II
LISTA DOS BARRAMENTOS IMPLEMENTADOS NO PROCESSADOR.

Nome	Tamanho (bits)	Função
BUS_ULA1	16	Barramento 1 de entrada da ULA
BUS_ULA2	16	Barramento 2 de entrada da ULA
BUS_EXT3	16	Barramento de saída da ULA

C. Fases do Controlador Microprogramado

Na arquitetura implementada, o controlador microprogramado possui 5 fases de operação. Cada fase está vinculada a sinais de controle específicos, que correspondem a movimentos e operações básicas nos registradores e barramentos de todo o processador. A Tabela III apresenta quais são as fases de operação do controlador microprogramado, trazendo também a função de cada fase e os sinais de controle vinculados a ela.

TABLE III

LISTA DOS BARRAMENTOS IMPLEMENTADOS NO PROCESSADOR.

Fase	Ações	Sinais de controle
1	Operações da ULA e transferência de barramentos	1 a 9
2	Armazenamento nos registradores PC, ACC, R1, R2, REM e RDM	10 a 15
3	Operações na memória principal e transferência de instrução para o IR	16 a 18
4	Preparação do endereço da próxima microinstrução	19 a 24
5	Atualização do MPC	-

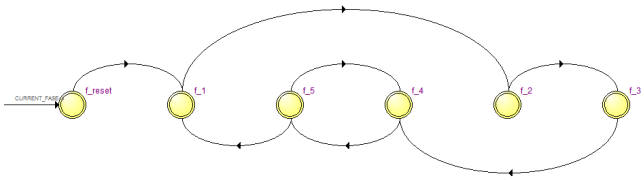
As fases do controlador microprogramado foram implementadas em VHDL com uma máquina de estados. Foram definidos 5 estados diferentes correspondentes às fases, controlados pelos sinais *CURRENT_FASE* e *NEXT_FASE* que indicam, respectivamente, a fase atual e a próxima fase. Projetou-se uma lógica para que a fase seja atualizada após o teste de todos os sinais de controle, executando suas respectivas ações quando eles estiverem ativos.

Para a implementação da máquina de estados, foi utilizada a estrutura de controle de fluxo *CASE*. O uso desta estrutura na definição na máquina de estados pode ser observado no seguinte trecho de código:

```
CASE CURRENT_FASE IS
WHEN F_RESET => NEXT_FASE <= F_1;
WHEN F_1 => NEXT_FASE <= F_2;
WHEN F_2 => NEXT_FASE <= F_3;
WHEN F_3 => NEXT_FASE <= F_4;
WHEN F_4 => NEXT_FASE <= F_5;
WHEN F_5 =>
        IF(SC(24) = '1') THEN
        NEXT_FASE <= F_4;
        ELSE
        NEXT_FASE <= F_1;
        END IF;
END CASE;
```

Para a atualização da fase, é atribuída ao sinal *NEXT_FASE* o valor da próxima fase, definido como F_x, em que x é o número da fase. Uma rotina chamada de *FASE_CHANGE* é responsável por atribuir o valor de *NEXT_FASE* para *CURRENT_FASE*, habilitando a sequência de teste e execução dos sinais de controle da fase atual. O diagrama da Figura 1 apresenta a lógica de mudança das fases do microprocessador.

Fig. 1. Diagrama com a lógica da mudança de fases do microprocessador.



Uma fase extra definida por *F_RESET* é responsável por configurar todos os registradores, barramentos e memórias após o início ou *reset* do microprocessador.

D. Memórias

As memórias de microprograma e principal foram definidas como variáveis bidimensionais, chamadas respectivamente de *MEM_MICRO* e *MEM_PRIN*. Ao vetor *MEM_MICRO* foi atribuído o tamanho de 24 bits para os dados e 18 endereços diferentes, totalizando 108 bytes para a micromemória. Já para *MEM_PRIN* foi atribuído o tamanho de 16 bits para os dados em 50 endereços diferentes, totalizando 200 bytes.

Para os testes inicias, a memória de microprograma foi preenchida com o ciclo de busca, a tabela de *jumps* e os códigos para a instruções *LOAD*, *STORE* e *ADD*. A Tabela IV apresenta os endereços da memória de microprograma e o conteúdo guardado em cada posição da memória.

TABLE IV

MICROMEMÓRIA DO PROCESSADOR.

Endereço	Dado
0x000	0x44C001
0x001	0x620211
0x002	0x80000B
0x003	0x80000D
0x004	0x80000F
0x005	0x000000
0x006	0x000000
0x007	0x000000
0x008	0x000000
0x009	0x000000
0x00A	0x000000
0x00B	0x44C020
0x00C	0x000440
0x00D	0x444020
0x00E	0x012002
0x00F	0x44C020
0x010	0x000442

No caso da memória principal, foram preenchidos apenas os 10 primeiros endereços, contendo uma rotina simples para teste das instruções implementadas. A Tabela V apresenta os endereços da memória principal e o conteúdo guardado nas posições que foram preenchidas.

TABLE V

MEMÓRIA PRINCIPAL DO PROCESSADOR.

Endereço	Dado
0x0000	0x0041
0x0001	0x0093
0x0002	0x0042
0x0003	0x0000
0x0004	0x0008
0x0005	0x0000
0x0006	0x0000
0x0007	0x0000
0x0008	0x0000
0x0009	0x0003

IV. RESULTADOS OBTIDOS

Como resultado deste projeto obteve-se o correto funcionamento do controlador microprogramado proposto, assim

como do microprocessador como um todo. Foram implementadas na micromemória as instruções ADD, LOAD e STORE, conforme proposto. Na memória principal, foi programada o carregamento de um valor, soma com o valor do acumulador e, por fim, o armazenamento deste resultado no mesmo endereço do valor inicial.

Devido à complexidade do microprocessador implementado, assim como o envolvimento de grande quantidade de aspectos referentes à microprogramação, este projeto pode ser utilizado para o ensino de processadores microprogramados, além de proporcionar amplas visão e compreensão do uso da FPGA e da linguagem de descrição de *hardware* utilizada (VHDL).

Referências Bibliográficas

[1] SEDRA A. S. , SMITH K. C. **Microelectronic Circuits, Volume 1**, Oxford University Press, 1998.

[2] DALTRINI, Beatriz M., JINO, Mario, MAGALHÃES, Léo P. **Introdução a Sistemas de Computação Digital**. São Paulo: MAKRON Books, 1999.

[3] Terasic, **DE1-SoC User Manual**, revisão Abr. 2015.

[4] MONTENEGRO, Toni Ferreira; GIRARDI, Alessandro.**Implementação em VHDL do Processador Educacional BIP I**. In: Iberchip XV Workshop, Buenos Aires, 2009. 2009.

[5] PASTOR, Javier Sanchez et al. **A remote laboratory for debugging FPGA-based microprocessor prototypes**. In: Advanced Learning Technologies, 2004. Proceedings. IEEE International Conference on. IEEE, 2004. p. 86-90.

Fig. 2. Diagrama funcional do microprocessador.