DOI: http://dx.doi.org/10.29069/forscience.2017v5n1.e218

DESENVOLVIMENTO DE UM PROCESSADOR BASEADO NA ARQUITETURA MIPS32 UTILIZANDO HARDWARE RECONFIGURÁVEL[1]

João Paulo Fernandes de Cerqueira César[2]
Otávio de Souza Martins Gomes[3]

RESUMO

Esse artigo apresenta o processo de desenvolvimento de um processador MIPS de 32 *bits* com funcionalidades reduzidas. Para isso, utiliza-se de conceitos de computação reconfigurável, por meio da sintetização em *Field Programmable Gate Array (FPGA)* com o uso da linguagem VHDL para descrição do *hardware*. O resultado desse projeto poderá ser utilizado como material didático em disciplinas relacionadas, além de oferecer um IP validado do processador MIPS 32 *bits*, fomentando, assim, o desenvolvimento de variados projetos de pesquisa na área de microeletrônica, arquitetura de computadores e linguagens de descrição de *hardware*.

Palavras-chave: MIPS. FPGA. Hardware Reconfigurável. VHDL.

1 INTRODUÇÃO

Com o crescimento dos investimentos nacionais na área de microeletrônica e na fabricação de circuitos integrados, é importante a realização de projetos de pesquisa e principalmente o seu desenvolvimento na área de prototipação de dispositivos reconfiguráveis, tendo em vista que eles participam do ciclo de projeto de um *Aplication-specific Integrated Circuit (ASIC)*, na fase validação da ideia proposta. As principais arquiteturas disponíveis no mercado para desenvolvimento de *hardwares* que utilizam plataformas reconfiguráveis são os *Programmable Systems on Chip (PSoCs)* e os *Field Programmable Gate Array (FPGAs)*.

O FPGA é um circuito lógico-programável, isto é, uma classe de circuitos integrados com propósito geral. É um dispositivo semicondutor que é largamente utilizado para o processamento de informações digitais. Criado pela Xilinx Inc. e lançado no ano de 1985 como

[1] **Como citar este artigo:**
CÉSAR, J. P. F. C.; GOMES, O. S. M. Desenvolvimento de um processador baseado na arquitetura MIP32 utilizando hardware reconfigurável. **ForScience:** revista científica do IFMG, Formiga, v. 5, n. 1, e00218, jan./jun. 2017.

[2] Mestrando em Ciência da Computação pela Universidade Federal de Lavras (UFLA). Professor Visitante no IFMG - campus Formiga. Bacharel em Ciência da Computação pelo IFMG campus Formiga. Currículo Lattes: <http://lattes.cnpq.br/1903713130721800>. E-mail: joaopaulofcc@gmail.com.

[3] Doutor em Engenharia Elétrica pela Universidade Federal de Itajubá (UNIFEI). Professor Efetivo no IFMG - campus Formiga. Currículo Lattes: <http://lattes.cnpq.br/5092964831326431>. E-mail: otavio.gomes@ifmg.edu.br.

um dispositivo que poderia ser programado de acordo com as aplicações do usuário (programador), tem como foco a flexibilidade de programação, associada a potentes ferramentas de desenvolvimento e modelagem, possibilitando ao usuário o desenvolvimento de projetos de circuitos integrados complexos, sem os altos custos de engenharia associados aos ASICs. Grandes empresas, como a Altera (ALTERA CORPORATION, 2016) e a Xilinx (XILINX INC, 2016), desenvolvem e vendem blocos de *hardware* denominados *Intelectual Property (IP)*, com características específicas, para que sejam integrados a outros sistemas.

Desenvolvida na década de 80, o *Microprocessor without Interlocked Pipe Stages (MIPS)* é uma arquitetura de processador de propósito geral, baseada em um conjunto reduzido de instruções (RISC), com o principal objetivo de oferecer alta performance na execução de códigos compilados (HENNESSY et al., 1982). As instruções da arquitetura MIPS são próximas às instruções de microcódigo executadas pelo processador, além disso, a complexidade individual das instruções são minimizadas, provendo maior desempenho da arquitetura. Processadores MIPS podem ser encontrados em diversos tipos de equipamentos, como por exemplo em videogames Sony Playstation 2, câmeras digitais Canon e roteadores Cisco (RUBIO; COOK, 2004). No ano de 2012, a MIPS Technologies, empresa criada em 1984 por John LeRoy Hennessy e detentora das patentes da arquitetura e processadores MIPS, foi adquirida pela empresa de desenvolvimento de *hardware* Imagination Technologies, por 60 milhões de dólares (HOLLISTER, 2012).

Esse trabalho aborda o desenvolvimento em *hardware* reconfigurável de um IP do processador MIPS de 32 *bits* com funcionalidades reduzidas. O processador desenvolvido será construído por meio da linguagem de descrição de *hardware* VHDL e posteriormente sintetizado em um FPGA. O desenvolvimento deste processador, fomenta a próxima fase a ser desenvolvida futuramente pelos autores, que é a sua integração a um sistema computacional de propósito educacional. Este sistema será desenvolvido utilizado plataformas *open-hardware*, como por exemplo o Arduino, e servirá de instrumento educacional em disciplinas correlacionadas. Esse documento está estruturado da seguinte forma: a seção 2 apresenta os trabalhos relacionados com este, a seção 3 e a seção 4 apresentam uma breve introdução sobre a arquitetura MIPS e sobre FPGA, respectivamente, já a seção 5 apresenta o processo de desenvolvimento do processador, na seção 6 é apresentado o processo de testes realizados para validação do projeto, e, por fim, na seção 7 são apresentadas as considerações finais acerca desse trabalho.

2 TRABALHOS RELACIONADOS

Na literatura estão presentes diversos trabalhos que abordam o desenvolvimento de uma arquitetura de processador MIPS em FPGA. É possível encontrar implementações do MIPS mo-

noprocessado em Reaz, Islam e Sulaiman (2002) ou até mesmo versões com implementação de *pipeline* como em Reis et al. (2000) e Rubio e Cook (2004). Além destes, também se destacam trabalhos com objetivos específicos, como o trabalho de Singh e Parmar (2015) que descreve o desenvolvimento de um processador criptográfico baseado na arquitetura MIPS utilizando o padrão 3-DES *(Triple Data Encryption Standard)*. Já no trabalho de Harris et al. (2017) é apresentada uma infraestrutura educacional utilizando o processador MIPS, em conjunto com estruturas de *hardware* desenvolvidas em FPGA e ferramentas de *software*. Apesar do assunto ser de interesse no meio acadêmico, em nenhum trabalho relatado até o presente momento desenvolveu-se a arquitetura objetivando seu acoplamento a um dispositivo *open-hardware* para uso educacional. Portanto, esse trabalho vem contribuir com a comunidade acadêmica pela experiência de tal desenvolvimento.

3 MIPS

Desenvolvida na década de 80, a arquitetura MIPS é baseada em um conjunto reduzido de instruções (RISC) que se caracteriza por: grande número de registradores de propósito geral ou uso de tecnologias de compilação com o objetivo de otimizar o uso de registradores; conjunto de instruções simples e limitado; enfoque na otimização do pipeline de instruções; uma instrução por ciclo de máquina; operadores de registrador para registrador; modos de endereçamento simples; formato de instruções simples (HENNESSY et al., 1982; STALLINGS, 2006). O termo ciclo de máquina é definido como o tempo gasto para buscar dois operandos do banco de registradores, executar operação na ALU e armazenar o resultado em um registrador (STALLINGS, 2006). Em uma arquitetura sem pipeline, como é o caso da desenvolvida, o ciclo de máquina deverá ser igual ao maior tempo de execução encontrado nas instruções implementadas (PATTERSON; HENNESSY, 2014).

Arquiteturas baseadas em conceitos RISC se opõem àquelas baseadas em conjuntos complexos de instruções (CISC), essas por sua vez fornecem diversos tipos de instruções simples ou complexas, não garantindo a propriedade de uma instrução por ciclo, entre outros aspectos opostos (STALLINGS, 2006). Como característica de projeto, a arquitetura possui 32 registradores de 32 *bits* de largura, dessa forma, o significado de uma *word* nessa arquitetura referencia uma sequência de 32 *bits* (PATTERSON; HENNESSY, 2005). Além disso, a menor parte endereçável na memória pela arquitetura é 1 *byte*, sendo assim, os endereços em *words* são endereçados em múltiplos de 4 *bytes*. Na Figura 1 é possível observar o esquema da arquitetura MIPS de 32 *bits*. É também característica da arquitetura MIPS, que operações aritméticas somente ocorram com operadores armazenados em registradores. Sendo assim, a arquitetura fornece mecanismos de transferência de operadores de 32 *bits* da memória para registradores

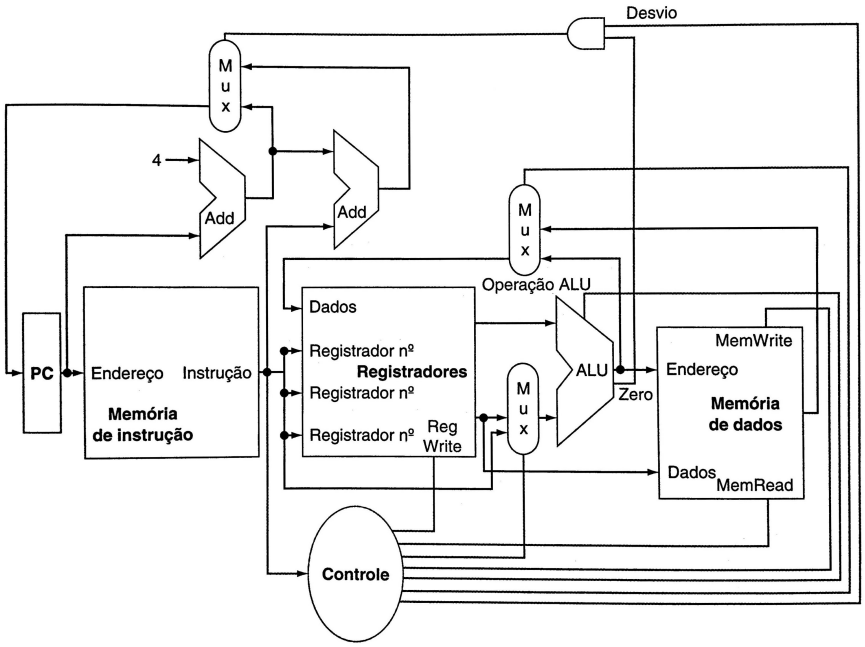


Figura 1 – Esquema da arquitetura do processador MIPS 32 *bits*
Fonte: Patterson e Hennessy (2014).

(load) e vice-versa *(store)*. Além disso, instruções dessa arquitetura possuem tamanho fixo de uma *word*, ou seja 32 *bits*. Seus formatos podem variar entre: **I:** responsáveis por trabalhar com manipulação de memória, ou seja, somente as instruções *Load* e *Store*; **J:** responsáveis por executar instruções de desvio condicionais *"branchs"* ou incondicionais - *jumps*; **R:** responsáveis por executar instruções aritméticas com operadores armazenados em registradores, como por exemplo adições, multiplicações e divisões (PATTERSON; HENNESSY, 2005).

4 FIELD PROGRAMMABLE GATE ARRAYS (FPGA)

A computação reconfigurável é uma solução intermediária na resolução de problemas complexos, possibilitando combinar a velocidade do *hardware* com a flexibilidade do *software*. Uma arquitetura reconfigurável possui várias metas, entre elas o aumento de desempenho. Dentre os vários segmentos relacionados às arquiteturas reconfiguráveis, destacam-se os Processadores Reconfiguráveis. Estes processadores combinam as funções de um microprocessador com uma lógica reconfigurável e podem ser adaptados depois do processo de desenvolvimento (CASILLO, 2006). O FPGA é composto estruturalmente por milhares de unidades lógicas idênticas. Neste aspecto, estas unidades lógicas podem ser vistas como componentes padrões que podem ser configurados independentemente e interconectados a partir de uma matriz de trilhas condutoras e chaves programáveis. A estrutura básica de um FPGA é formada pelo ve-

tor de unidades lógicas, Look-Up Tables (LUTs), blocos para entrada e saída de dados, e pela matriz de interconexão, que podem ser programados pelo usuário (KUON; TESSIER; ROSE, 2007). A configuração do FPGA é realizada por meio de um arquivo que descreve a estrutura e funcionalidade do componente a ser sintetizado. Por meio desse arquivo, é que a matriz de interconexão será fechada. Para gerar o arquivo binário, podem ser utilizadas ferramentas de *software* seguindo um determinado fluxo de projeto.

Recursos adicionais como *flip-flops*, multiplexadores, lógica de transporte, *carry* dedicado e portas lógicas, podem ser utilizados em conjunto com os LUTs para implementar diversas funções booleanas, multiplicadores, somadores, contadores, conversores serial-paralelo e paralelo-serial, e memórias com grande variedade de tamanhos de palavra, fornecendo assim flexibilidade ao desenvolvimento (CASILLO, 2006).

Para que um *hardware* seja desenvolvido em FPGA, podem ser utilizadas várias linguagens de descrição de *hardware* (HDL). Dentre as linguagens mais conhecidas, pode-se destacar o VHDL. Criada pelo Departamento de Defesa Norte Americano nos anos 80, VHDL é uma linguagem para descrição de *hardware* capaz de descrever o comportamento eletrônico de um circuito ou sistema (PEDRONI, 2004).

As aplicações básicas de VHDL são em FPGA's, CPLD's e ASIC's. O VHDL é uma linguagem que trabalha de forma concorrente, ao contrário de muitas linguagens de programação, que são sequenciais (BOTROS, 2006). De modo geral, é uma linguagem adequada para se trabalhar com grandes projetos de alto nível de abstração. Em VHDL, assim como nas outras linguagens, os projetos são organizados em hierarquias, isto é, interpretando componentes básicos como blocos que virão a formar um bloco maior, que neste caso é o projeto em si, tornando o código reutilizável (PEDRONI, 2004). Após o processo de declaração dos códigos, o processo de desenvolvimento de uma aplicação em VHDL passa por um estágio de síntese dos circuitos e entidades declaradas, processo no qual são utilizados algoritmos de otimização para que esses circuitos ocupem o menor espaço possível. É nessa fase também que ocorrem as simulações para validação do comportamento do componente desenvolvido. Caso tudo esteja correto, o componente é então entregue para a camada de controle físico, que será responsável por carregar o componente na placa. Caso o resultado não seja satisfatório nas simulações, o componente é reestruturado pelo desenvolvedor. Além de VHDL, existem outras linguagens tais como Verilog, SystemVerilog e SystemC, cada uma dessas tem seus pontos fortes e fracos, além do que, a utilização de qualquer uma delas no desenvolvimento dependerá do conhecimento por parte dos desenvolvedores e das ferramentas disponíveis. Um sistema pode ser modelado de maneira eficiente em qualquer uma destas linguagens (BOTROS, 2006).

Neste trabalho será utilizada a placa de desenvolvimento Altera DE2, mostrada na Figura 2, equipada com o FPGA Altera Cyclone II EP2C35F672C6. Esse FPGA possui 33.216

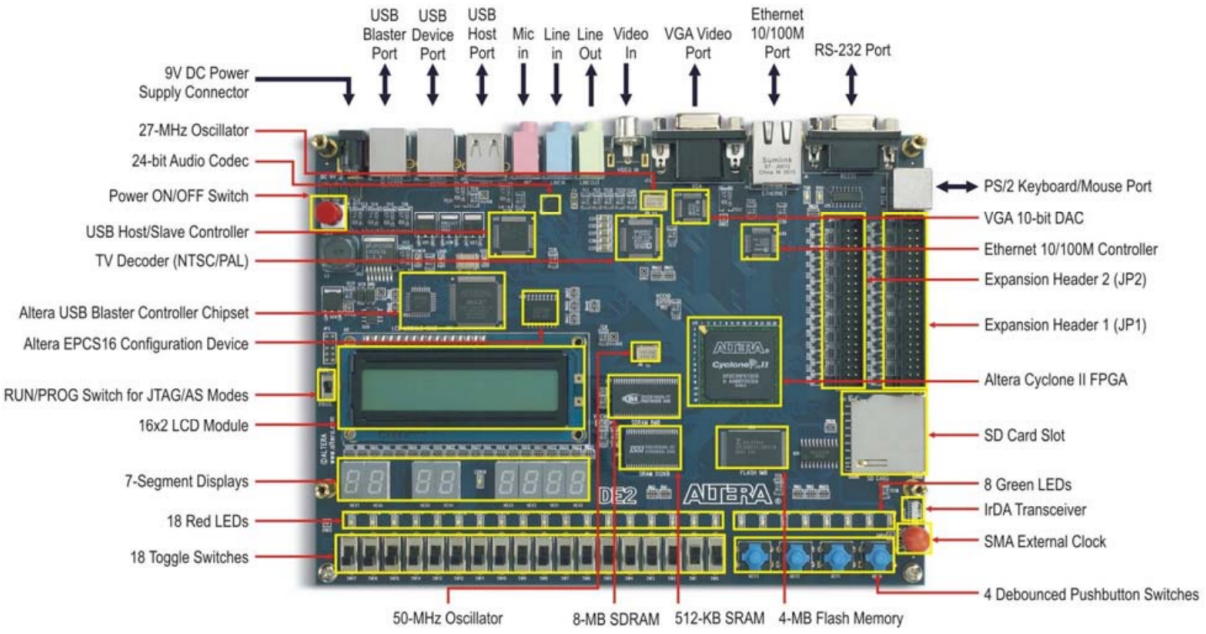


Figura 2 – Placa de desenvolvimento Altera DE2
Fonte: ALTERA CORPORATION (2006).

elementos lógicos e a placa de desenvolvimento conta com diversos elementos como *pushbuttons*, LEDs, *clocks* de 50 MHz e 27 MHZ, decodificador de vídeo e áudio, além de 80 pinos de expansão de uso geral (GPIO).

5 DESENVOLVIMENTO

O desenvolvimento de todos os componentes foi realizado por meio da descrição utilizando VHDL no *software* Altera Quartus II 64-Bit Version 13.0.1 Web Edition. Para desenvolvimento dos componentes do processador em VHDL, algumas práticas foram adotadas, por exemplo, componentes que necessitam de uma camada mais sofisticada de controle e que são naturalmente planejados como uma sequência de tarefas bem definidas, foram desenvolvidos por meio de máquinas de estado, ou *finite state machines (FSM)*.

Para melhor organização do projeto, o desenvolvimento do processador foi separado em módulos, cada um desses responsável por executar uma tarefa específica, e por fim, ocorreu a integração de todos esses módulos por meio de barramentos. Além da vantagem de facilitar o processo de organização, a separação de módulos com responsabilidades restritas favorece o reaproveitamento de código e facilita o processo de alteração do comportamento de cada um desses módulos necessários. Assim, desde que cada bloco mantenha a mesma interface de comunicação, ou seja, barramentos de comunicação, codificação de *opcodes*, e operações

disponíveis, a forma como realiza internamente suas operações pode ser alterada sem o acarretamento de problemas para os demais componentes dependentes. Foram descritos em VHDL os módulos necessários para o correto funcionamento do MIPS, são eles: módulo de *clock*, memórias de dados e instruções, banco de registradores, unidade aritmética e lógica (ALU) e a unidade de controle responsável por gerenciar todo o comportamento dos demais.

Não Implementadas = 4			Implementadas = 60				
ADD	ADDU	AND	DIV	DIVU	JALR	JR	MFHI
MFLO	MOVN	MOVZ	MTHI	MTLO	MULT	MULTU	NOP
NOR	OR	SLL	SLLV	SLT	SLTU	SRA	SRAV
SRL	SRLV	SUB	SUBU	XOR	BAL	BGEZ	BGEZAL
BLTZ	BLTZAL	J	JAL	B	BEQ	BNE	BLEZ
BGTZ	ADDI	ADDIU	SLTI	SLTIU	ANDI	ORI	XORI
LUI	CLO	CLZ	MADD	MADDU	MSUB	MSUBU	MUL
LB	LH	LW	LBU	LHU	SB	SH	SW

Figura 3 – Conjunto de instruções adotado
Fonte: Elaborado pelo autor.

No processo de implementação foi definido previamente um conjunto de instruções a serem reconhecidas pelo processador, esse conjunto foi extraído do conjunto disponibilizado no Volume II do manual da revisão 0.95 do MIPS (MIPS-TECHNOLOGIES, 2001). Cabe ressaltar que não foram implementadas instruções dependentes do coprocessador de ponto flutuante, visto que esse não foi desenvolvido neste trabalho. Também, instruções consideradas ultrapassadas *(deprecated)* segundo o documento consultado, não foram implementadas. A Figura 3 exibe o subconjunto selecionado e as instruções nele implementadas.

Após implementação das memórias e do banco de registradores, iniciou-se o desenvolvimento da ALU, parte fundamental em um processador. Trata-se de um componente acessado pela unidade de controle para realização de diversos tipos de cálculos, tanto lógicos quanto aritméticos. Assim, de acordo com as instruções que foram definidas para integrar o processador (vide Figura 3), aquelas que possuíam perfil de operações lógicas e aritméticas foram implementadas na ALU. A Figura 4 exibe uma esquemático da ALU implementada no processador. O último componente desenvolvido é a unidade de controle, que deve ser capaz de gerenciar toda a operação do processador, ou seja, executar qualquer combinação válida do conjunto de instruções, realizando a correta comunicação entre as memórias, banco de registradores e ALU, um esquemático de sua estrutura é apresentado na Figura 5.

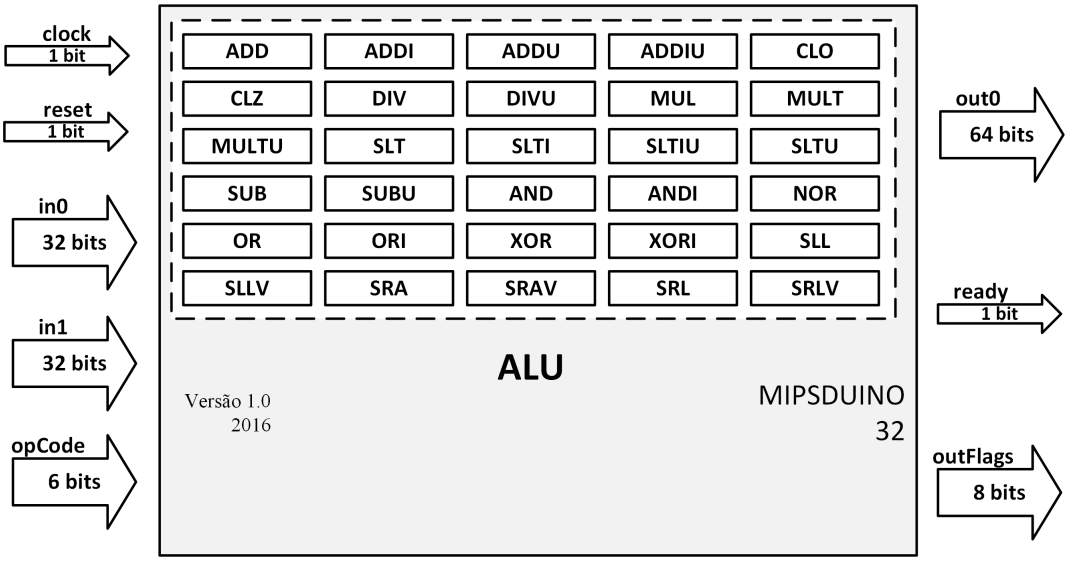


Figura 4 – Esquema da ALU
Fonte: Elaborado pelo autor.

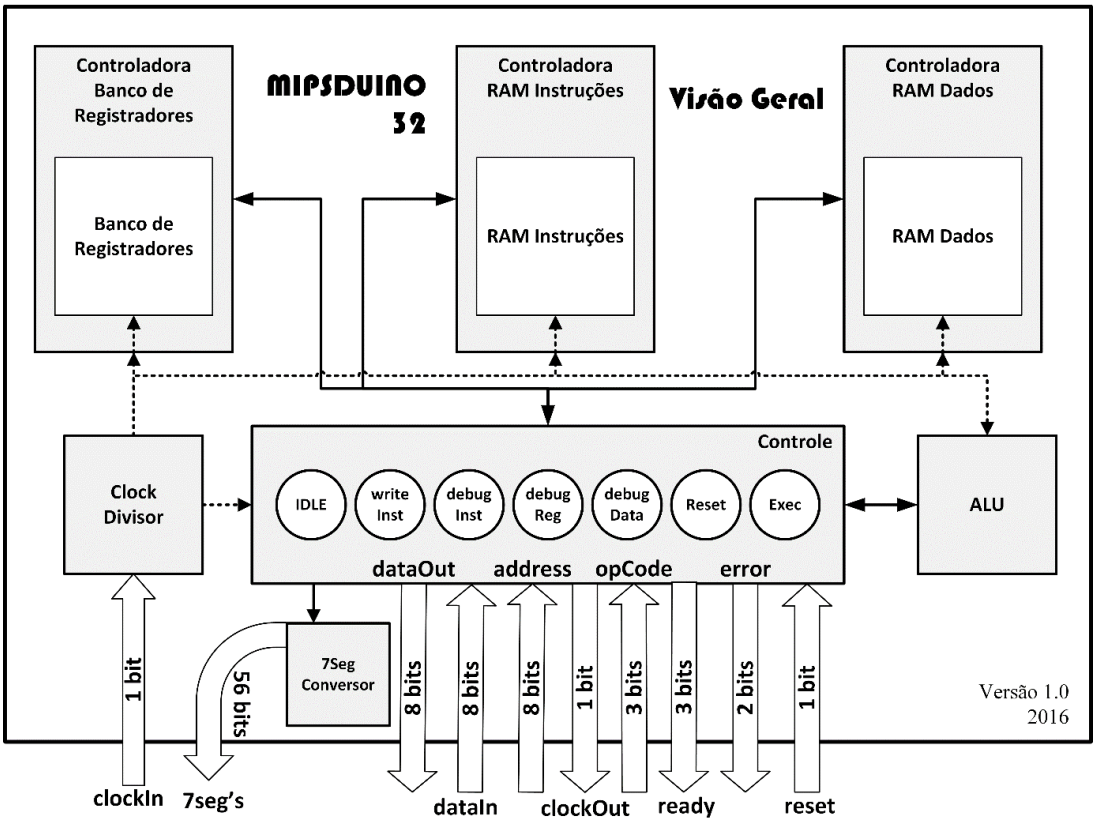
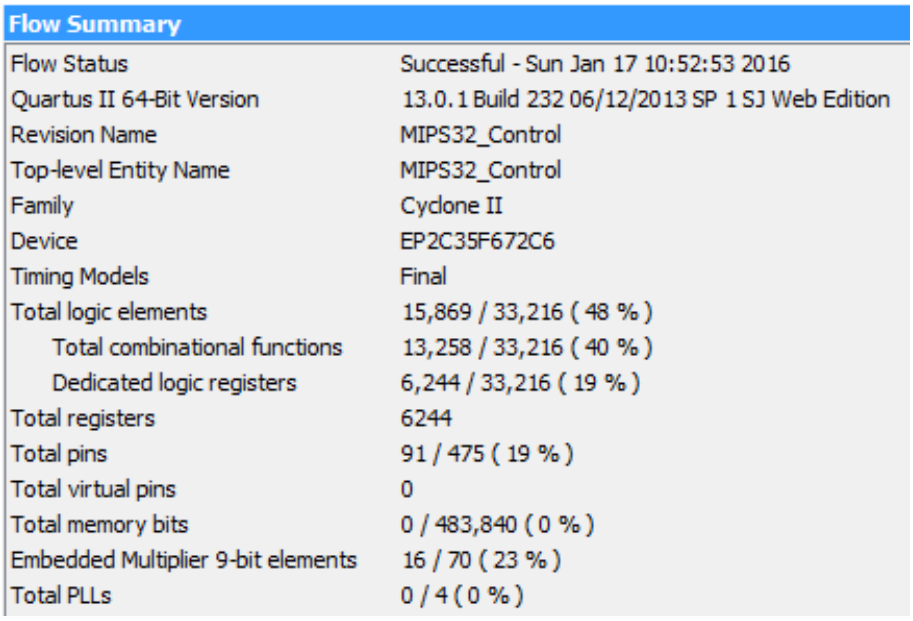


Figura 5 – Esquema da unidade de controle
Fonte: Elaborado pelo autor.

Os componentes da arquitetura, depois de desenvolvidos individualmente, foram integrados à unidade de controle. Esta foi sintetizada e, posteriormente, foram extraídos os dados da utilização dos recursos da Altera DE2 demonstrados na Figura 6. A figura mostra que não foram utilizados *memory bits* disponíveis no FPGA, esse fato deve-se ao modo como foram projetadas as memórias RAM e o banco de registradores. Esses foram descritos com características específicas inerentes ao projeto do processador desenvolvido (múltiplas leituras e escritas em único *clock*, etc.) e diferentes do padrão recomendado pela Altera (ALTERA CORPORATION , 2013. Cap. 13) para que pudessem ser sintetizados utilizando estruturas de memória presentes no FPGA, por isso, estes componentes foram sintetizados utilizando elementos lógicos.



Flow Summary	
Flow Status	Successful - Sun Jan 17 10:52:53 2016
Quartus II 64-Bit Version	13.0.1 Build 232 06/12/2013 SP 1 SJ Web Edition
Revision Name	MIPS32_Control
Top-level Entity Name	MIPS32_Control
Family	Cyclone II
Device	EP2C35F672C6
Timing Models	Final
Total logic elements	15,869 / 33,216 (48 %)
Total combinational functions	13,258 / 33,216 (40 %)
Dedicated logic registers	6,244 / 33,216 (19 %)
Total registers	6244
Total pins	91 / 475 (19 %)
Total virtual pins	0
Total memory bits	0 / 483,840 (0 %)
Embedded Multiplier 9-bit elements	16 / 70 (23 %)
Total PLLs	0 / 4 (0 %)

Figura 6 – Utilização de componentes da Altera DE2 após síntese
Fonte: Elaborado pelo autor.

Na Figura 7 é exibida a máquina de estados que representa a lógica de operação da unidade de controle do processador. Em destaque e numerados estão os blocos de operações, tais blocos são conjuntos de estados da máquina que têm como objetivo a execução de uma determinada operação, seja ela mapeada nos *opcodes* reconhecidos pelo componente ou operações internas da unidade de controle. O Quadro 1 apresenta uma breve descrição de cada bloco.

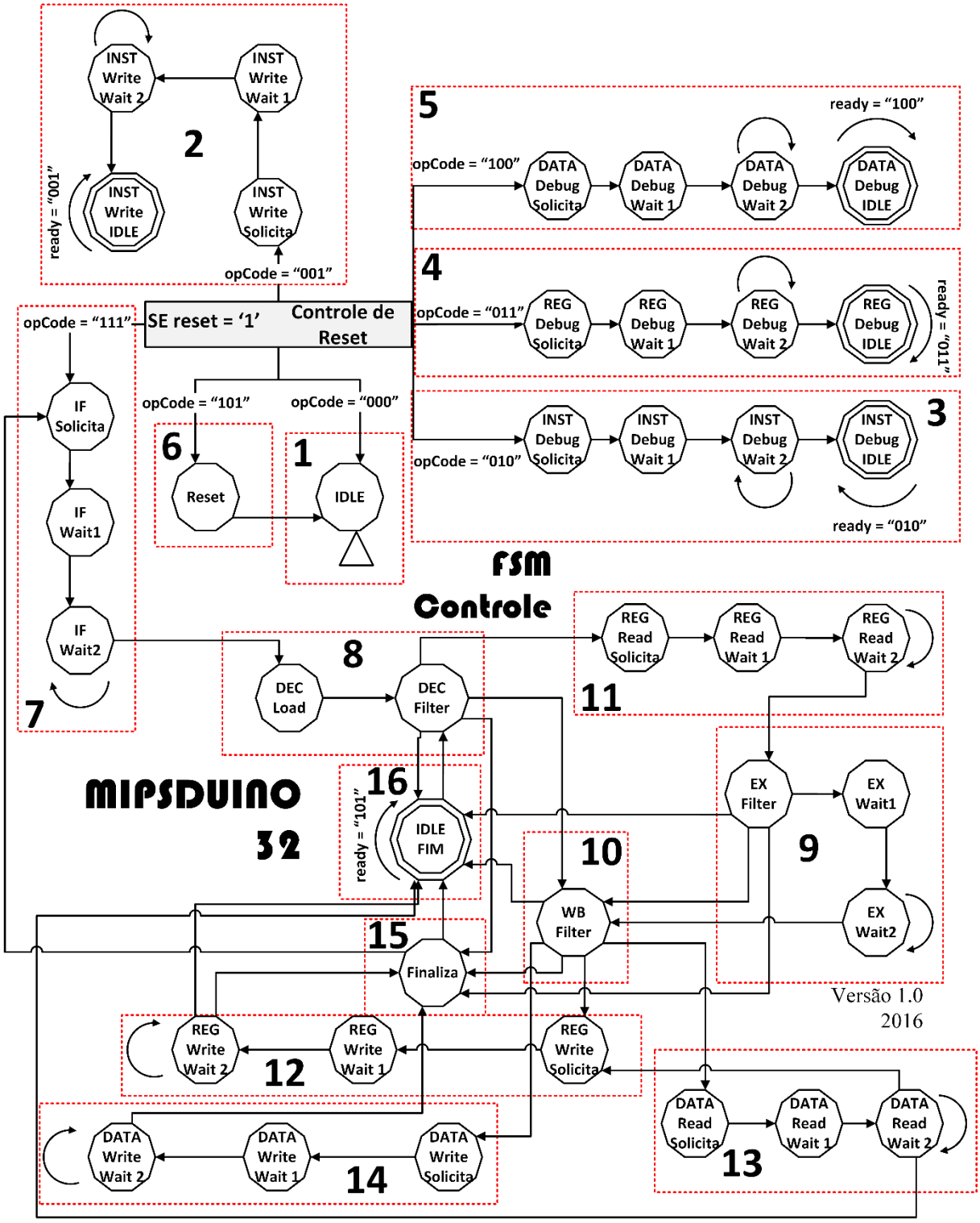


Figura 7 – Comportamento da unidade de controle
Fonte: Elaborado pelo autor.

Bloco 1	execução da operação de solicitação de estado ocioso (IDLE) no circuito.
Bloco 2	execução da operação de escrita de instruções na memória de instrução.
Bloco 3	*dump* da memória de instruções.
Bloco 4	*dump* do banco de registradores.
Bloco 5	*dump* da memória de dados.
Bloco 6	*reset* na unidade de controle.
Bloco 7	início da operação de execução das instruções carregadas na memória.
Bloco 8	decodificação da instrução lida.
Bloco 9	execução da instrução carregada.
Bloco 10	*writeback* da instrução.
Bloco 11	processo de leitura de um registrador do banco de registradores.
Bloco 12	operação de escrita em registrador.
Bloco 13	operação de leitura de dados da memória de dados (tipo "*load*").
Bloco 14	operação de escrita na memória de dados.
Bloco 15	comparação do contador de programa e o contador de instruções carregadas.
Bloco 16	estado de término da execução das instruções, com ou sem erros.

Quadro 1 – Descrição dos blocos para máquina de estados de controle
Fonte: Elaborado pelo autor.

6 TESTES E VALIDAÇÃO

A validação do processador ocorreu por meio de ferramentas de simulação, em que cada componente foi validado individualmente, para que, posteriormente, o processador como um todo fosse analisado. Após o envio das instruções para a memória RAM, foi solicitado à unidade de controle que iniciasse o processo de execução das instruções presentes nessa memória. Todas as instruções foram testadas individualmente tanto por meio de simulação, quanto no circuito já sintetizado no FPGA. Após testes individuais, programas mais complexos foram produzidos e testados. Ambas as fases de validação obtiveram sucesso.

Como exemplo do processo de validação é mostrado na Figura 8 a simulação da execução da instrução "XORI 10 10 5". O primeiro passo nesse processo consiste na leitura do conteúdo do registrador 10, em seguida é solicitada uma operação na ALU para calcular o valor da operação "5 XOR 0", terminado o cálculo o resultado será então salvo no registrador 10. A Figura 8 mostra o resultado dessas operações de forma gráfica, nela é mostrada a instrução que é executada. Nesse exemplo é mostrada a instrução x"394A0005", que corresponde ao binário da instrução "XORI 10 10 5". Além disso, é mostrado também o valor do cálculo de "5 XOR 0 = 5" e o posterior processo de escrita desse resultado no registrador 10. A execução dessa instrução demandou 1200ns de processamento.

Outro exemplo de execução é o mostrado na Figura 9. Nesse, é ilustrado o ciclo de processamento da instrução "MUL 11 11 10". Dessa forma, o processador deverá ler o conteúdo dos registradores 11 e 10, realizar a multiplicação deste e em seguida salvar o resultado no

registrador 11. Neste exemplo os conteúdos dos registradores 11 e 10 são respectivamente os valores 7 e 5, assim, o resultado dessa operação de multiplicação é igual à 35, como observado. Em destaque, estão presentes os processos de solicitação de leitura dos registradores 11 e 10, a solicitação de execução da operação MUL na ALU e também a solicitação de escrita do resultado no registrador 11. A execução dessa instrução demandou 1160ns de processamento.

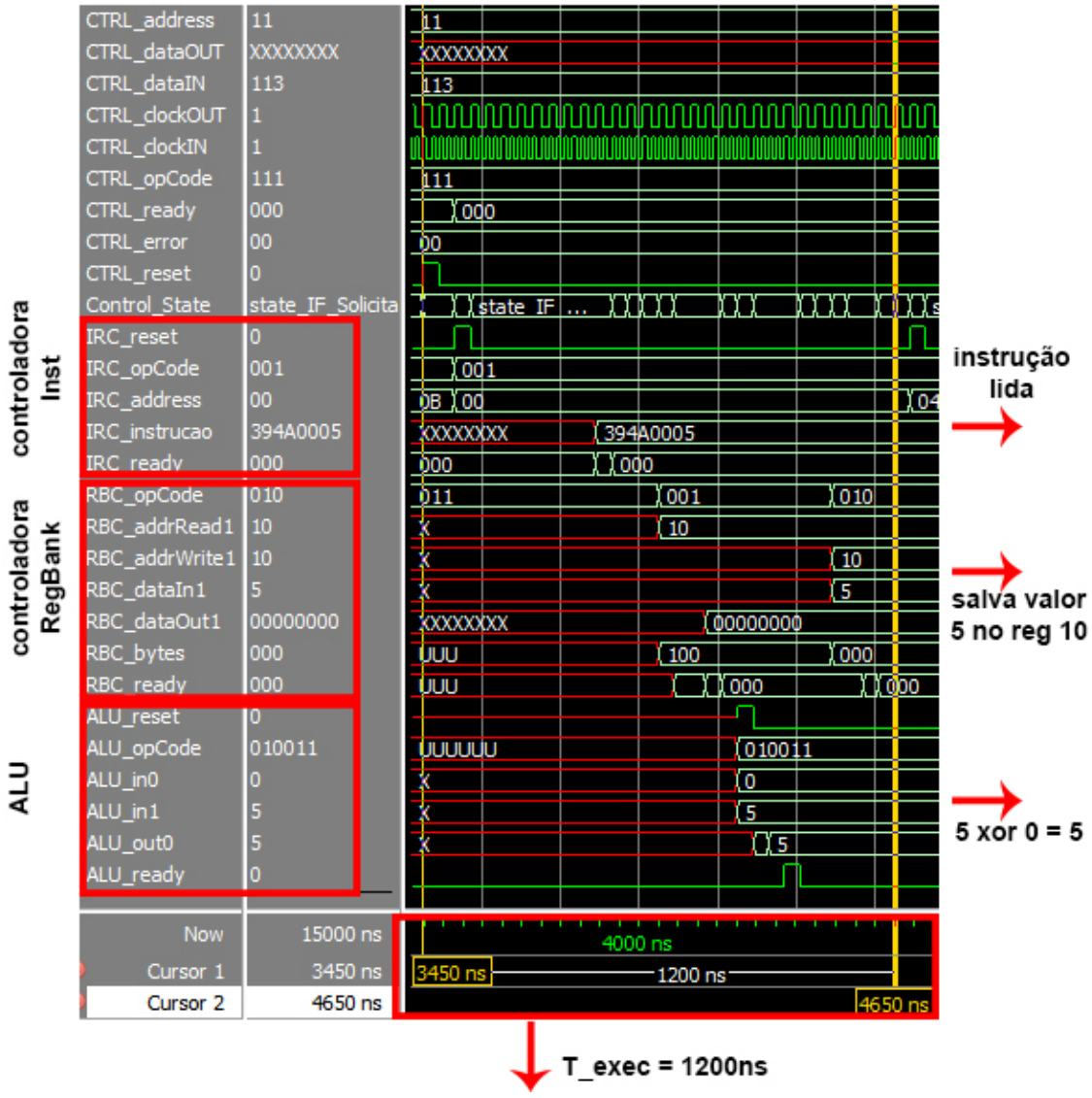


Figura 8 – Execução da instrução XORI 10 10 5
Fonte: Elaborado pelo autor.

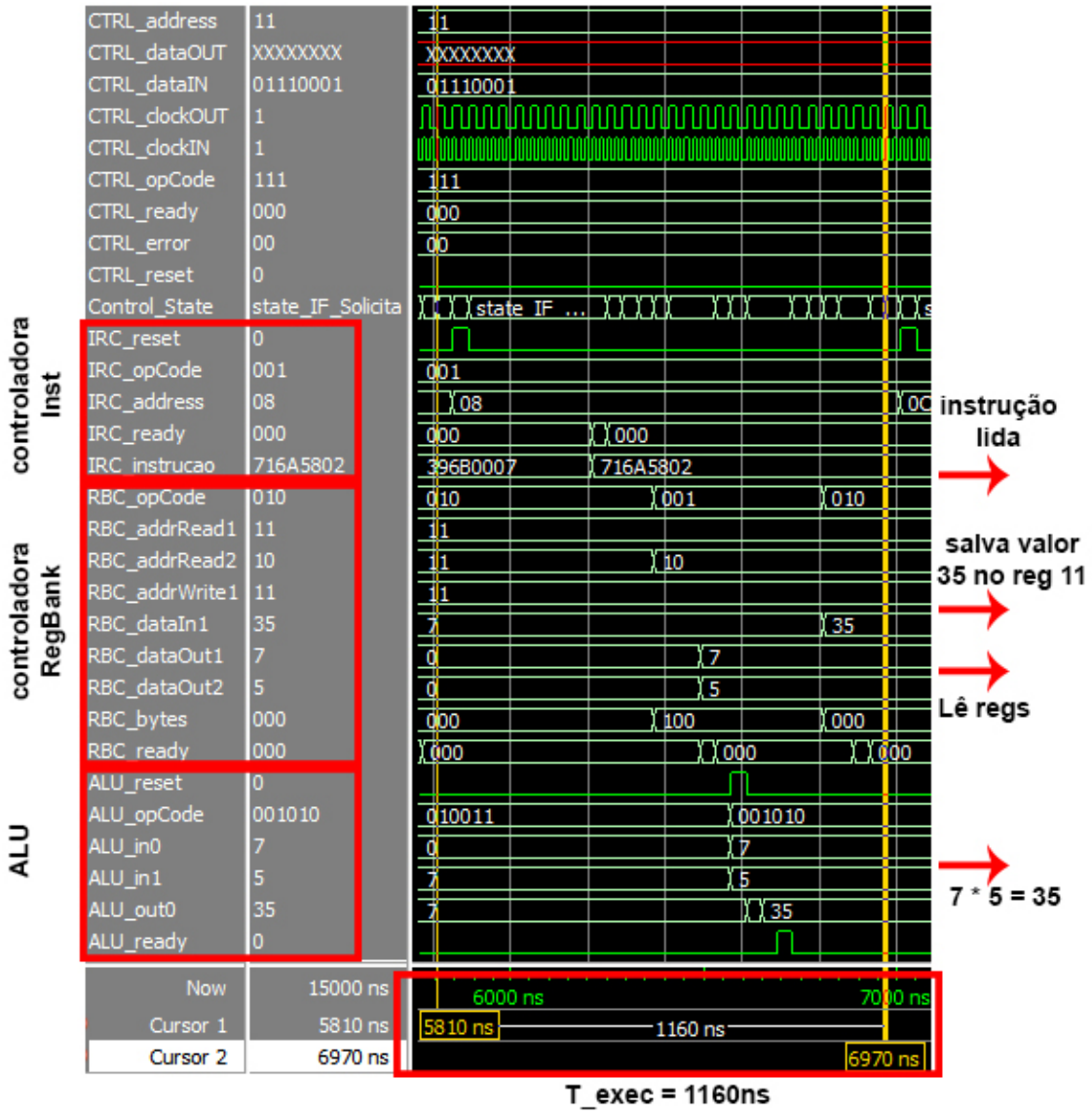


Figura 9 – Execução da instrução MUL 11 11 10
Fonte: Elaborado pelo autor.

7 CONSIDERAÇÕES FINAIS

O trabalho obteve sucesso no desenvolvimento do processador, todos os objetivos foram cumpridos e todas as unidades desenvolvidas foram validadas. Sendo assim, está claro a contribuição do presente trabalho para com a comunidade acadêmica, por meio do desenvolvimento e validação do processador MIPS32. O sucesso da execução dos testes fomenta o próximo passo do trabalho que é o desenvolvimento de uma plataforma educacional e a sua interface com o processador, de forma a facilitar a utilização desse por usuários não familiarizados com ferramentas de simulação e VHDL. Além disso, deve-se considerar uma futura implementação desse

mesmo projeto, porém com processador utilizando *pipeline* e implementação da estrutura de pilha, bem como suas instruções de manipulação. Por último, pode-se planejar a implementação do coprocessador de ponto flutuante. Como melhoria na taxa de utilização dos componentes do FPGA, fica como trabalho futuro a otimização das estruturas de memória para utilização dos *memory bits* disponibilizados, reduzindo assim a taxa de utilização dos elementos lógicos do FPGA.

DEVELOPMENT OF A PROCESSOR BASED IN MIPS32 ARCHITECTURE USING RECONFIGURABLE HARDWARE

ABSTRACT

This paper presents the process of developing a 32-bit MIPS processor with reduced functionality, using concepts of reconfigurable computing through the synthesis on Field Programmable Gate Array (FPGA) using VHDL language to describe the hardware. The result of this project could be used as teaching material in related disciplines, besides offering a validated IP of MIPS 32 bits processor, which can be further used as part of various research projects in microelectronics, computer architecture and hardware description languages.

Keywords: MIPS. FPGA. Reconfigurable Hardware. VHDL.

REFERÊNCIAS

ALTERA CORPORATION . Recommended hdl coding styles. In: **Quartus II Handbook**: Version 13.1: volume 1: design and synthesis. San Jose: Altera Corporation, 2013. Cap. 13. p. 76. Disponível em: <https://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/hb/qts/archives/quartusii_handbook_archive_131.pdf>. Acesso em: 06 set. 2016.

ALTERA CORPORATION. **DE2 Development and Education Board**: user manual. 1.4. ed. San Jose: Altera Corporation, 2006. 72 p. Disponível em: <ftp://ftp.altera.com/up/pub/Webdocs/DE2_UserManual.pdf>. Acesso em: 06 set. 2016.

____. **Intellectual Property**. 2016. Disponível em: <https://www.altera.com/products/intellectual-property/overview.tablet.html>. Acesso em: 06 set. 2016.

BOTROS, N. **HDL Programming Fundamentals**: VHDL and verilog. Boston, Mass: Da Vinci Engineering Press, 2006. 400 p. ISBN 9781584508557.

CASILLO, L. A. **Projeto e implementação em FPGA de um processador com conjunto de instrução reconfigurável utilizando VHDL**. 2006. 110 f. Dissertação (Mestrado em Sistemas e Computação) — Universidade Federal do Rio Grande do Norte, 2006. Disponível em: <http://repositorio.ufrn.br/handle/123456789/18071>. Acesso em: 06 set. 2016.

HARRIS, S. L. et al. Mipsfpga: using a commercial mips soft-core in computer architecture education. **IET Circuits, Devices & Systems**, Institution of Engineering and Technology, 2017. ISSN 1751-858X. Disponível em: <http://digital-library.theiet.org/content/journals/10.1049/iet-cds.2016.0383>.

HENNESSY, J. et al. Mips: a microprocessor architecture. In: ANNUAL WORKSHOP ON MICROPROGRAMMING. 15, 1982, Piscataway, NJ, USA. **Proceedings...** Piscataway: IEEE Press, 1982. (MICRO 15), p. 17–22. Disponível em: <http://dl.acm.org/citation.cfm?id=800036.800930>. Acesso em: 06 set. 2016.

HOLLISTER, S. **Silicon shakeup**: imagination technologies buys mips, hopes to compete with arm processors. 2012. Disponível em: <http://www.theverge.com/2012/11/7/3611972/imagination-buys-mips-arm-gets-patents>. Acesso em: 06 set. 2016.

KUON, I.; TESSIER, R.; ROSE, J. FPGA Architecture: survey and challenges. **Foundations and Trends in Electronic Design Automation**, v. 2, n. 2, p. 135–253, 2007. ISSN 1551-3939. Disponível em: <http://www.nowpublishers.com/article/Details/EDA-005>. Acesso em: 06 set. 2016.

MIPS-TECHNOLOGIES. **MIPS32 Architecture For Programmers Volume II**: the mips32 instruction set. 0.95. ed. Mountain View, CA: [s.n.], 2001. 253 p. Disponível em: <http://www.cs.cornell.edu/courses/cs3410/2015sp/MIPS_Vol2.pdf>. Acesso em: 06 set. 2016.

PATTERSON, D.; HENNESSY, J. **Organização e Projeto de Computadores**: interface hardware / software. Rio de Janeiro: Elsevier, 2005. ISBN 9788535215212.

PATTERSON, D. A.; HENNESSY, J. L. **Organização e Projeto de Computadores**: interface hardware / software. Rio de Janeiro, Brasil: Elsevier, 2014. 736 p. ISBN 8535264108.

PEDRONI, V. A. **Circuit Design with VHDL**. Cambridge, Mass: MIT Press, 2004. 363 p. ISBN 9780262162241.

REAZ, M.; ISLAM, M.; SULAIMAN, M. A single clock cycle mips risc processor design using VHDL. In: INTERNATIONAL CONFERENCE ON NEURAL INFORMATION PROCESSING (ICONIP 02). 9. [S.1.], 2002. **Proceedings...** [s.1.:s.n], Computational Intelligence for the E-Age (IEEE Cat. No.02EX575). IEEE, 2002. p. 17–22. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1217806>. Acesso em: 06 set. 2016.

REIS, A. et al. **Descrição VHDL do MIPS Pipeline**. 2000. 21 p. Disponível em: <http://www.inf.ufrgs.br/~lisane/relatMIPS.pdf>. Acesso em: 06 set. 2016.

RUBIO, V. P.; COOK, J. **A FPGA Implementation of a MIPS RISC Processor for Computer Architecture Education**. 2004. Dissertação (Mestrado) — New Mexico State University, 2004. Disponível em: <http://www.ece.nmsu.edu/~jecook/thesis/Victor_thesis.pdf>. Acesso em: 06 set. 2016.

SINGH, K. P.; PARMAR, S. **Design of High Performance MIPS Cryptography Processor Based on T-DES Algorithm**. **CoRR**, **abs/1503.03166**, 2015. Disponível em: <http://arxiv.org/abs/1503.03166>.

STALLINGS, W. **Arquitetura e organização de computadores**: projeto para o desempenho. São Paulo: Pearson Prentice Hall, 2006. 786 p. ISBN 9788587918536.

XILINX INC. **Intellectual Property**. 2016. Disponível em: <http://www.xilinx.com/products/intellectual-property.html>. Acesso em: 06 set. 2016.

Recebido em: 31/10/2016

Aprovado em: 17/05/2017

Publicado em: 04/09/2017