XIV ENCONTROS UNIVERSITÁRIOS DA UFC EM QUIXADÁ

Processador Didatico de 8 bits em FPGA

Uma Ferramenta para o Ensino Prático em Arquitetura de Computadores

Autor: Pedro Henrique Magalhães Botelho
Orientador: Roberto Cabral Rabêlo da Silva
Coautor: Thiago Werlley Bandeira da Silva

Introdução

- **Problemática**: Computadores estão cada vez mais complexos e poderosos.
  - Profissional de TI deve saber seu funcionamento e como são projetados.
  - Em Arquitetura de Computadores usam-se modelos simplistas.
  - Geralmente teóricos, fornecendo pouco ou nenhum aprendizado prático.
- **Solução**: Modelo didático otimizado para o ensino teórico-prático de computadores.
  - Processador em FPGA implementado em VHDL para simplicidade.
  - Modelo personalizado fornece flexibilidade para a modificação por alunos e professores.

Objetivos

- **Objetivo Principal**:
  - Projeto de um processador de 8 bits para o ensino de arquitetura de computadores e áreas correlatas.
- **Objetivos Específicos**:
  - Fornecer uma descrição de *hardware* completamente funcional em VHDL para FPGA.
  - Permitir o uso do processador por meio da placa didática Zybo.
  - Detalhar decisões de projeto visando a simplicidade e flexibilidade.

Trabalhos Relacionados

- Vários trabalhos propõe processadores didáticos simples com diferentes abordagens.
- **[de Abreu 2014]** implementa o processador Neander em VHDL para FPGA.
  - Implementação real do modelo teórico de 8-bits criado na UFRGS para o ensino.
- **[Santa 2017]** implementa um processador didático de 4-bits em VHDL.
  - Projeto minimalista e modular permite a modificação pelos alunos.
- **[Nascimento 2006]** implementa um processador semelhante ao MIPS usando HDL.
  - Possibilita o ensino prático de *hardware* usando FPGA.

Fundamentação Teórica

- O estudo e projeto de processadores envolve alguns termos chave.
- O **computador** é uma máquina capaz de processar dados usando um *software*.
- O **processador** interpreta e executa **instruções**, operações elementais da CPU.
  - O **datapath** é o caminho que os dados percorrem na CPU.
    - Desde a **ULA** (processamento em si) até os **registradores** (armazenamento interno à CPU).
  - A **unidade de Controle** Gerencia as operações por meio de sinais de controle.
- **Memórias** armazenam dados e instruções (Von Neumann X Harvard).
- A metodologia **RISC** especifica CPUs com instruções simples de um ciclo de clock cada.
- Um *hardware* pode ser descrito em código **VHDL** para um hardware flexível, a **FPGA**.

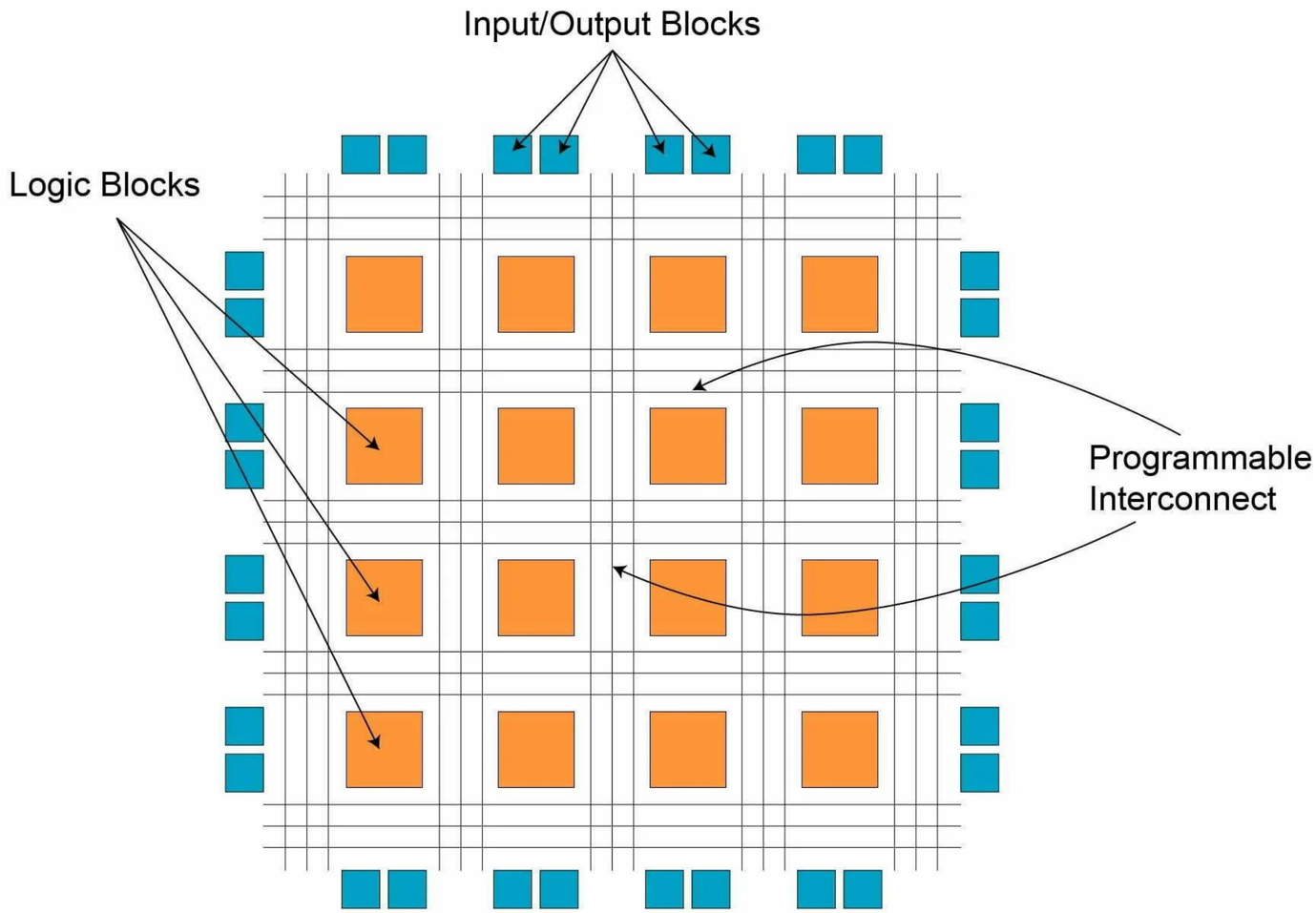

Figura 1. Organização Interna de uma FPGA
Fonte: (Technologies, 2023)

Procedimentos Metodológicos

- **Especificações do Projeto**: Processador CRISC (*Compact* RISC) seguiu os requisitos:
  a. Filosofia RISC: Facilidade de projeto, entendimento e uso.
  b. Arquitetura Load/Store com instruções simples.
  c. Instruções e dados de 8-bits.
  d. Modelo monociclo: Uma instrução por ciclo de *clock*.
  e. Arquitetura de Havard: Memórias separadas para dados (RAM) e instruções (ROM).
  f. Quatro registradores da CPU são acessíveis.
  g. I/O mapeada na memória: Primeiros 2 endereços para LEDs e *switches*.

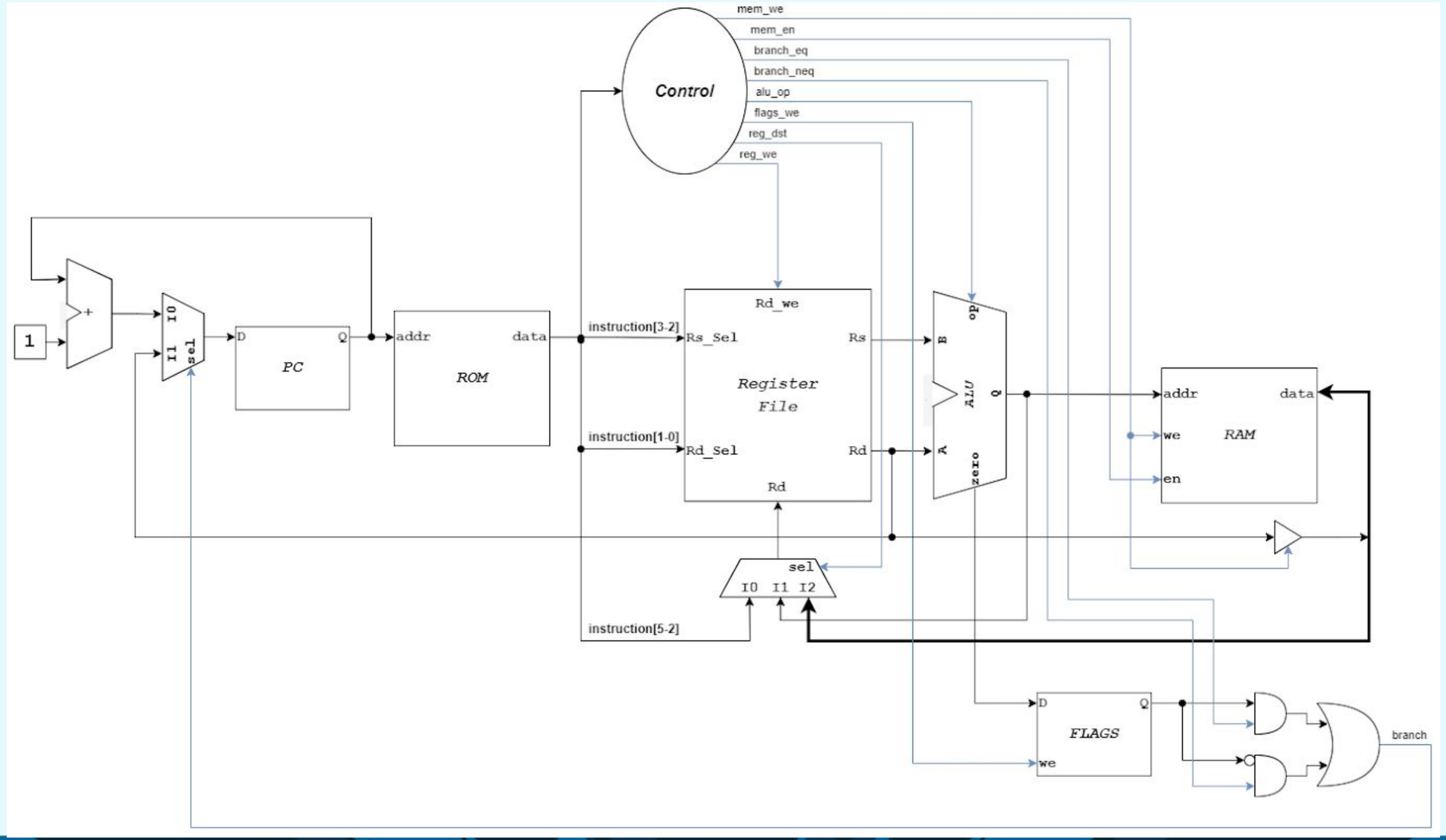

Figura 2. Diagrama do Processador CRISC
Fonte: Elaborado pelo Autor

Procedimentos Metodológicos

- **Instruções do Processador**: 16 instruções simples de até dois operandos.
  - Espaço reduzido limitaram os *opcodes* disponíveis.
  - Incluídas apenas as instruções mais simplistas (e.g. sem instruções para I/O).
- **Implementação em VHDL**: Projeto implementado e simulado usando o Vivado.
  - Código modularizado, permitindo a fácil leitura e modificação do projeto.
- **Gravação na FPGA**: Código projetado para permitir a gravação em FPGA (placa Zybo).
  - Permite o uso do *hardware* pelos alunos, por meio dos LEDs e *switches* da placa.

Instrução	Descrição	7	6	5	4	3	2	1	0
HALT	Para o processador	0	0	0	0	0	0	0	0
MOV Rd, Rs	Rd <= Rs	0	0	0	1	Rs_1	Rs_0	Rd_1	Rd_0
ADD Rd, Rs	Rd <= Rd + Rs	0	0	1	0	Rs_1	Rs_0	Rd_1	Rd_0
SUB Rd, Rs	Rd <= Rd + Rs	0	0	1	1	Rs_1	Rs_0	Rd_1	Rd_0
AND Rd, Rs	Rd <= Rd AND Rs	0	1	0	0	Rs_1	Rs_0	Rd_1	Rd_0
OR Rd, Rs	Rd <= Rd OR Rs	0	1	0	1	Rs_1	Rs_0	Rd_1	Rd_0
SHL Rd, Rs	Rd <= Rd <<Rs	0	1	1	0	Rs_1	Rs_0	Rd_1	Rd_0
SHR Rd, Rs	Rd <= Rd >>Rs	0	1	1	1	Rs_1	Rs_0	Rd_1	Rd_0
NOT Rd	Rd <= ¬Rd	1	0	0	0	0	0	Rd_1	Rd_0
B Rd	PC <= Rd	1	0	0	0	0	1	Rd_1	Rd_0
BEQ Rd	Se Z, PC <= Rd	1	0	0	0	1	0	Rd_1	Rd_0
BNE Rd	Se Z, PC <= Rd	1	0	0	0	1	1	Rd_1	Rd_0
LDR Rd, Rs	Rd <= mem(Rs)	1	0	0	1	Rs_1	Rs_0	Rd_1	Rd_0
STR Rd, Rs	mem(Rs) <= Rd	1	0	1	0	Rs_1	Rs_0	Rd_1	Rd_0
CMP Rd, Rs	Z <= Rd - Rs	1	0	1	1	Rs_1	Rs_0	Rd_1	Rd_0
LI Rd, X	Rd <= X	1	1	X_3	X_2	X_1	X_0	Rd_1	Rd_0

Tabela 1. Instruções do Processador
Fonte: Elaborado pelo Autor

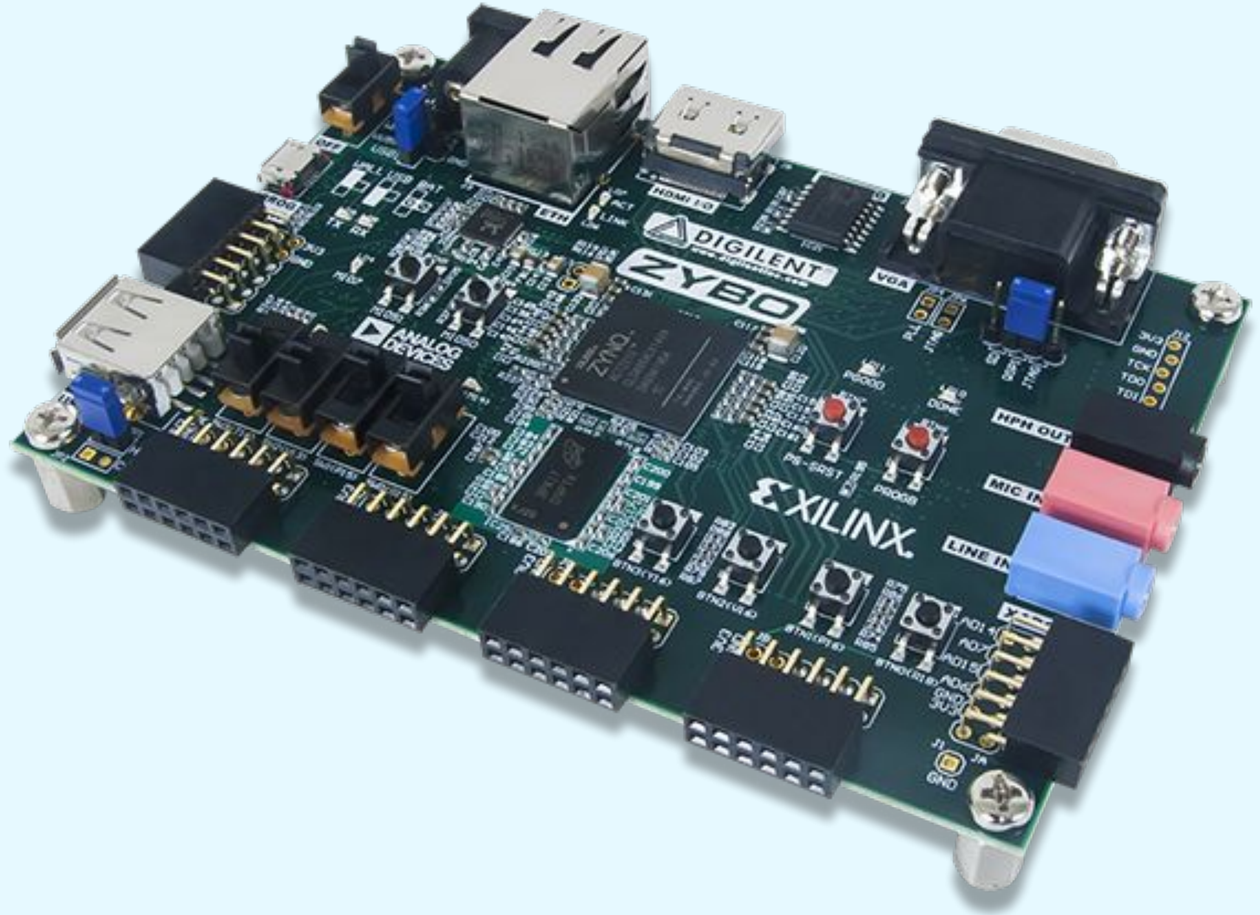

Figura 3. Placa de Desenvolvimento com FPGA Zybo
Fonte: (Digilent, 2014)

Resultados

- Código implementado, simulado e sintetizado para assegurar seu funcionamento.
- **Simulação Virtual**: Assegurar a correta decodificação e execução de cada instrução.
  - Verificou-se resultados de cada instrução, sinais gerados pela UC, fluxo do programa, etc...
- **Execução na FPGA**: Assegurar que o código é completamente sintetizável.
  - Verificou-se leitura dos switches, processamento dos dados e escrita nos LEDs.
- O completo funcionamento do processador foi alcançado.
  - Pode ser usado por alunos e professores em disciplinas relacionadas a processadores.
  - O Assembler CASM por ser usado para facilitar a programação do processador.

```
w_Operation <= f_DecodeInstruction(i_Instruction);

o_Memory_Write_Enable <= '1' WHEN (w_Operation = op_STR) ELSE '0';

o_Memory_Enable <= '1' WHEN (w_Operation = op_LDR OR w_Operation = op_STR) ELSE '0';

o_Branch <= '1' WHEN (w_Operation = op_B) ELSE '0';

o_Branch_Equal <= '1' WHEN (w_Operation = op_BEQ) ELSE '0';

o_Branch_Not_Equal <= '1' WHEN (w_Operation = op_BNE) ELSE '0';

o_Load_Flags <= '1' WHEN (w_Is_Arithmetic_Logic = '1') ELSE '0';

o_Register_Write_Enable <= '1' WHEN (
    (w_Is_Arithmetic_Logic = '1' AND w_Operation /= op_CMP) OR
    w_Operation = op_MOV OR
    w_Operation = op_LDR OR
    w_Operation = op_LI
) ELSE '0';
```

Figura 4. Código VHDL da Geração de Sinais na Unidade de Controle
Fonte: Elaborado pelo Autor

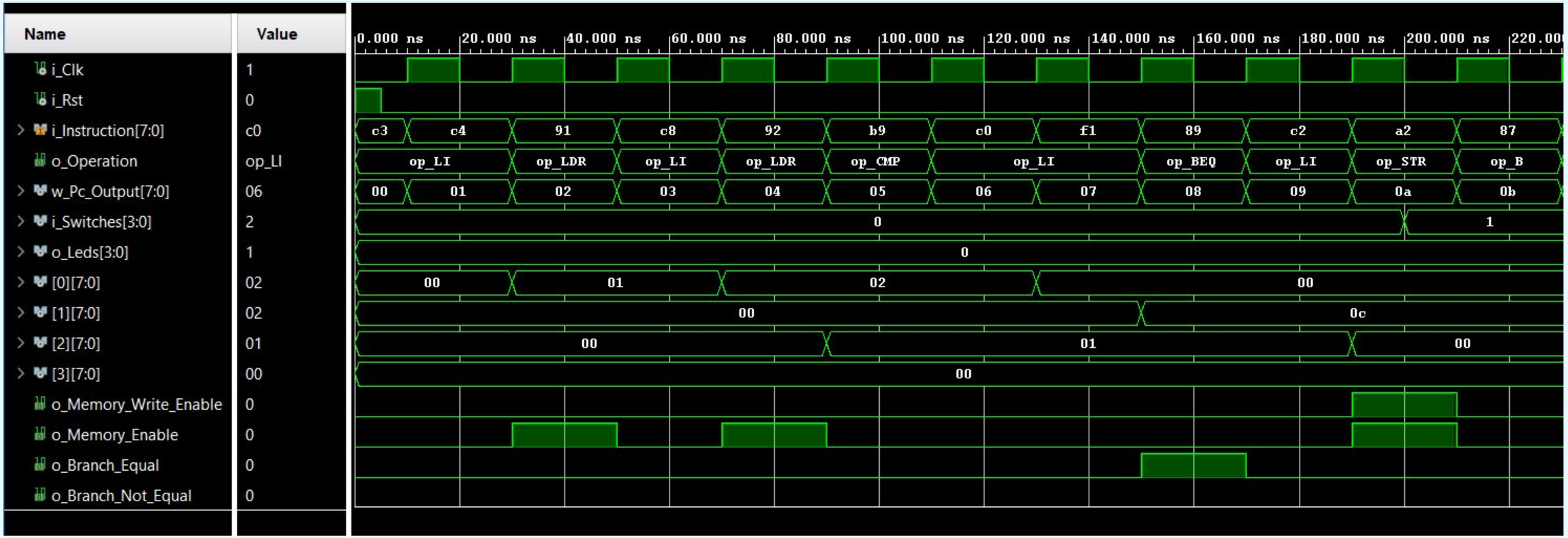

Figura 5. Sinais Gerados na Simulação do Processador
Fonte: Elaborado pelo Autor

Vídeo 1. Processador Executando sobre a FPGA
Fonte: Elaborado pelo Autor

Conclusão

- O estudo da computação cada vez mais essencial: Necessário entender os processadores.
  - Utilização de um modelo simplificado para o ensino é crucial no ensino teórico-prático.
  - Possui ampla aplicabilidade nas diversas disciplinas que estudam processadores.
- Nesse trabalho foi projetado o processador e o montador: Um trabalho em andamento.
  - Devemos avaliar seu uso nas disciplinas e analisar o impacto no desenvolvimento dos alunos.
- No geral, serve como base para projetos mais complexos: Permite um ensino gradual.
  - Alunos e professores também podem modificar sua estrutura conforme necessário.

Referências Bibliográficas

- Botelho, P. (2024a). Repositorio do Assembler. https://github.com/botelhocpp/crisc_assembler.
- Botelho, P. (2024b). Repositorio do Processador. https://github.com/botelhocpp/crisc_vhdl.
- de Abreu, M. A. L. (2014). Implementação de Processadores Didáticos Utilizando Linguagem de Descrição de Hardware.
- Nascimento, V. L. (2006). Projeto e Implementação de um Processador em FPGA para Ensino de Arquitetura de Computadores.
- Santa, F. M. (2017). Minimalist 4-bit Processor Focused on Processors Theory Teaching.
- Techologies, L. F (2023). FPGA Design, Architecture and Applications. https://www.logic-fruit.com/blog/fpga/fpga-design-architecture-and-applications/
- Digilent. (2014). Zybo (Legacy). Disponível em: https://digilent.com/reference/programmable-logic/zybo/start?srsltid=AfmBOoquIsbBzX0tZaEHmmzxIMPZ_2KnQDe8QOE0wzva09mhebFixkGD.

ENCONTROS UNIVERSITÁRIOS 2024

SABERES QUE ILUMINAM

UFC 70

UFC