**UNIVERSIDADE FEDERAL DE PELOTAS**

**CENTRO DE DESENVOLVIMENTO TECNOLÓGICO**

**CURSO DE ENGENHARIA DE COMPUTAÇÃO**

**DISCIPLINA DE SISTEMAS DIGITAIS AVANÇADOS**

****

**RELATÓRIO SOBRE TRABALHO FINAL**

**PROCESSADOR MIPS PIPELINE**

**ANDRÉ NACHTIGALL, HENRIQUE KESSLER E WAGNER LOCH**

**PELOTAS, DEZEMBRO DE 2018**

**André Nachtigall, Henrique Kessler e Wagner Loch**

**Processador MIPS Pipeline**

Relatório realizado como requisito do Trabalho Final da disciplina de Sistemas Digitais Avançados.

**Resumo**

O objetivo do trabalho é aplicar os conhecimentos adquiridos ao longo da disciplina de Sistemas Digitais Avançados em um projeto de maior escala. Foi escolhido pelo grupo implementar o Processador MIPS Pipeline conforme aprendido na disciplina de Arquitetura e Organização de Computadores I.

**Desenvolvimento**

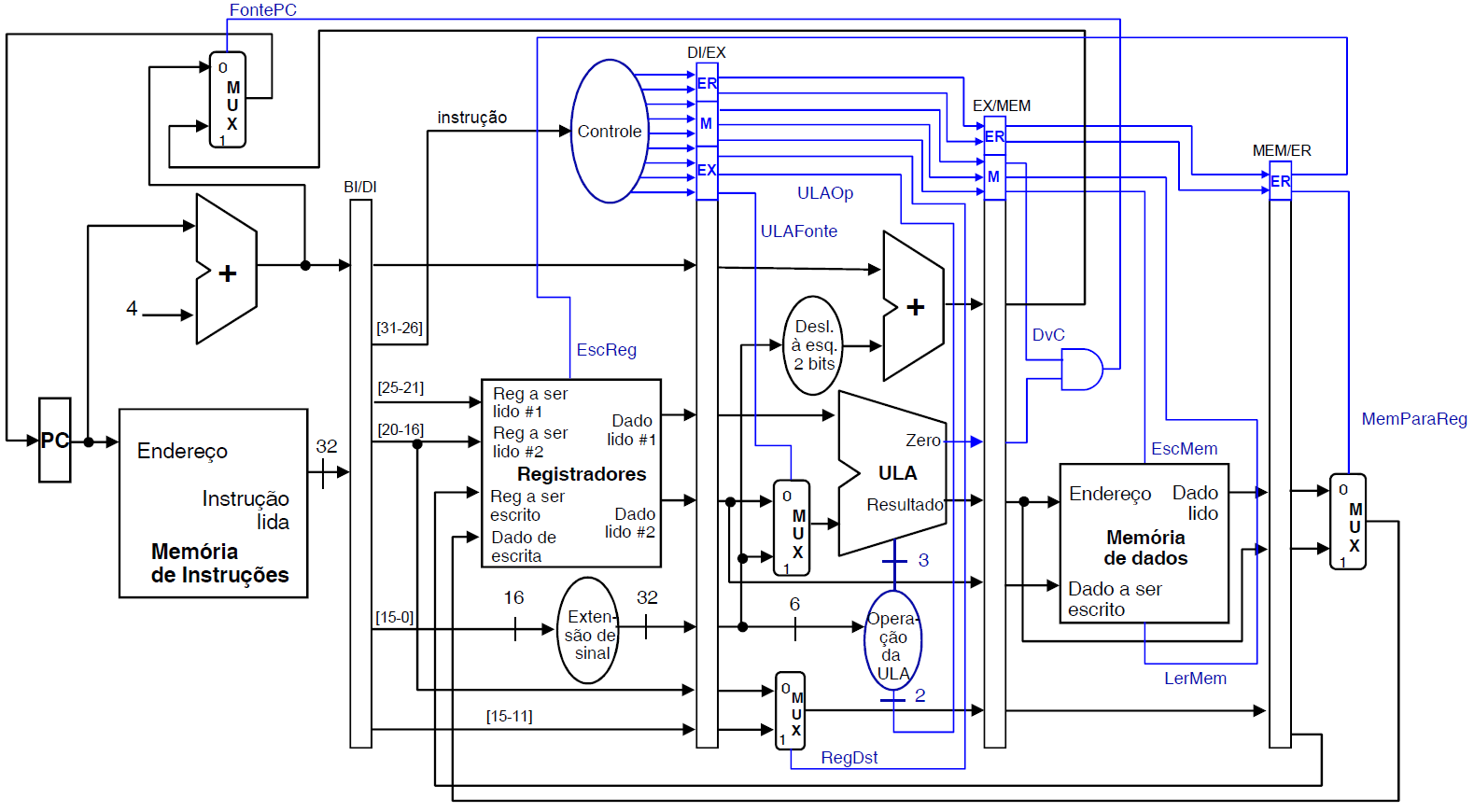
O diagrama de blocos utilizado para desenvolver o Processador MIPS Pipeline segue conforme a Figura 1.

Figura 1. Diagrama de blocos do Processador MIPS Pipeline.

O sistema foi separado em diversas entidades, sendo elas: adder4, adderAB, controle, dec\_5x1, extend\_16\_to\_32, fliflop, matrix32x32, memData, memInst, memInst2, MIPS, mux\_2x1, mux\_32x1, opULA, PC, RegBank, RegN, shift\_left\_2 e ULA. Esta fragmentação facilitou o desenvolvimento do projeto por permitir o reuso de código e por facilitar a validação, tendo em vista que cada entidade foi validada individualmente.

**Resultados**

A tecnologia alvo foi a família Cyclone II da Altera, os resultados obtidos da implementação proposta segue conforme a Tabela 1.

Tabela 1 – Resultados obtidos.

|  |  |
| --- | --- |
| **Categoria** | **Resultado** |
| Funções Combinacionais | 2593 |
| Registradores | 1227 |
| Frequência de Operação | 88,39 MHz |
| Bits de memória | 32768 |
| Vazão\* | 1 |

\*Após o preenchimento do pipeline.

A validação do sistema foi feita através da análise das formas de onda na saída do último estágio de execução conforme a Figura 2. Foi executado o seguinte código:

RESET

ADDI R1 R1 5 (R1 = R1 + 5)

ADDI R1 R1 1 (R1 = R1 + 1)

ADDI R1 R1 2 (R1 = R1 + 2)

ADDI R1 R1 3 (R1 = R1 + 3)

ADDI R1 R1 4 (R1 = R1 + 4)

WAIT

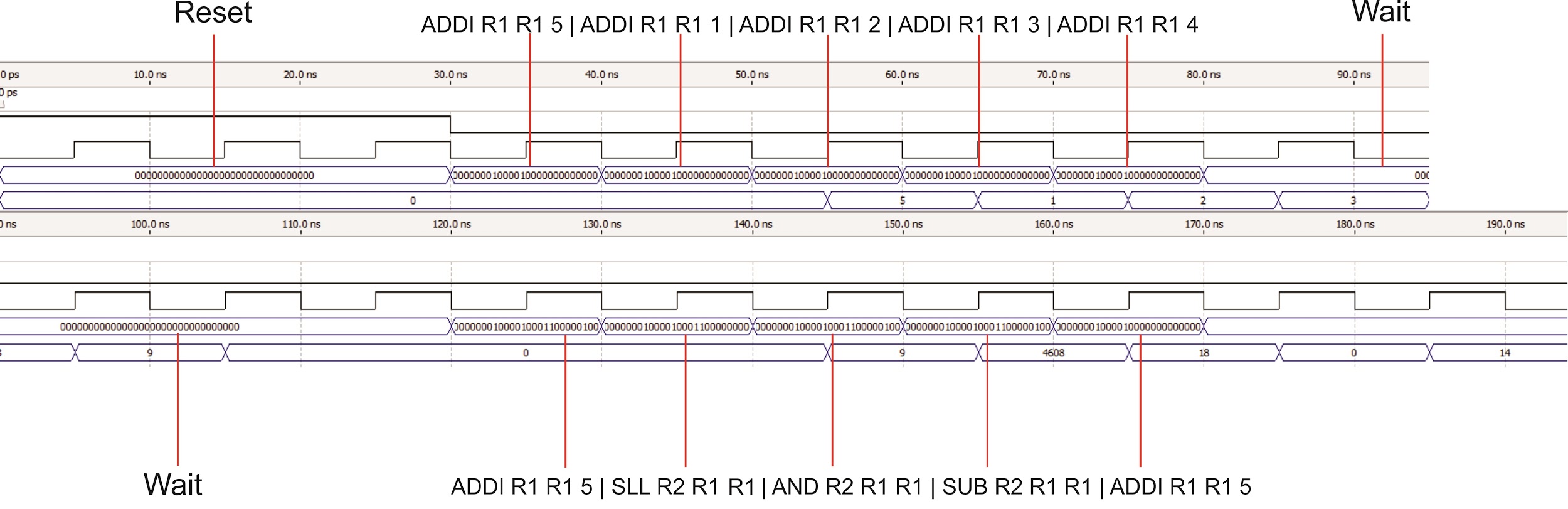
ADDI R1 R1 5 (R1 = R1 + 5)

SLL R2 R1 R1 (R2 = R1 << R1)

AND R2 R1 R1 (R2 = R1 AND R1)

SUB R2 R1 R1 (R2 = R1 - R1 )

ADDI R1 R1 5 (R1 = R1 + 5)

Figura 2 - Formas de onda do código executado.

É possível verificar o funcionamento correto das operações, todavia, o sistema de controle não trata os conflitos entre operações, é possível verificar este evento nas primeiras cinco instruções, onde são feitas as somas com os valores desatualizados.