Especificação e projeto de um microprocessador assíncrono

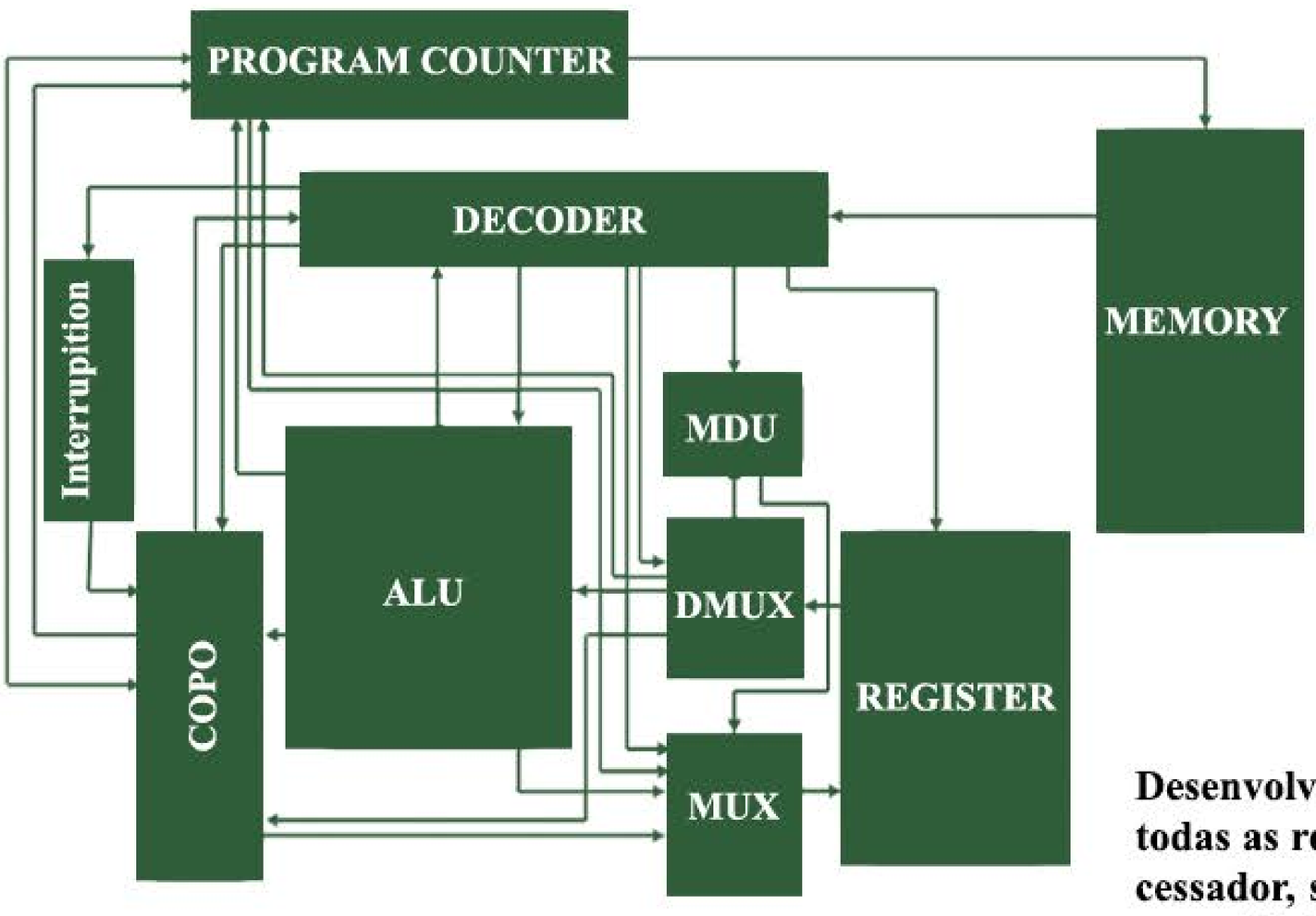
Introdução

Circuitos assíncronos são compostos por comunicação modular que é controlada pela presença dos dados nas entradas e saídas do circuito. Esta anatomia básica de comunicação é a mesma empregada pela linguagem GO para sincronizar os processos concorrentes através de canais de comunicação.

Linguagem GO

Goroutine é uma função que executa concorrentemente com outras goroutines no mesmo espaço de endereço. Os canais combinam trocas de valor com sincronização para garantir que duas goroutines estão em um estado conhecido.

Diagrama



Processador

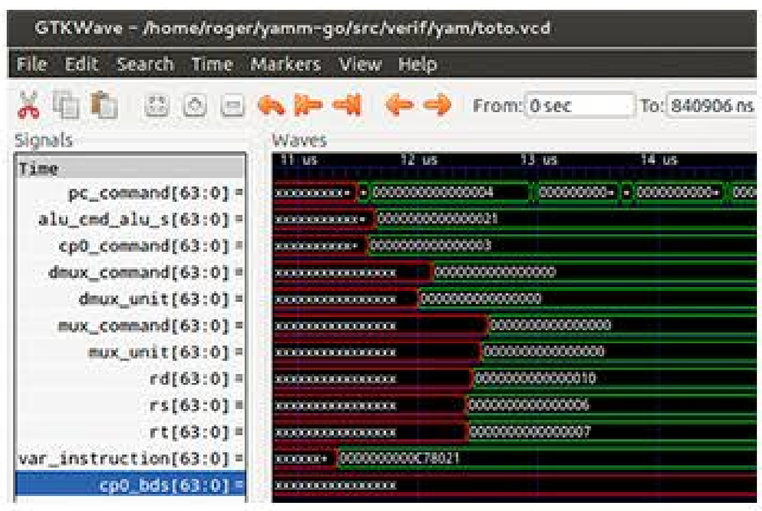
O processador modelado e simulado:
MIPS32 4KsTM
- 32 bits
(compatível com instruções de 16)
- Suporte a interrupções
- Unidade COP0 incluída
- Pipeline de 5 estágios
(executando 3 instruções ao mesmo tempo)

Metodologia

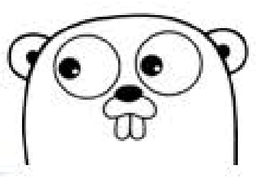
Desenvolvimento de um package em GO que engloba todas as rotinas criadas para executar as funções do processador, sendo que cada uma dessas funções representa um módulo do processador a ser modelado (ver diagrama). A sincronização e a transferência de dados entres as referidas funções é realizada por meio de canais.
Também foram testados aspectos específicos da linguagem, como o não-determinismo, e a capacidade em modelar circuitos.

Código

```
func Yam (
 pcMemory        chan *PC_MEMORY,
 memoryDecode    chan *MEMORY_DECODE,
 aluMemory       chan *ALU_MEMORY,
 memoryAlu       chan *MEMORY_ALU) {

 pcMemory        := make (chan *PC_MEMORY)
 pcMux           := make (chan *PC_MUX)
 pcCopo          := make (chan *PC_COPO)

...
```

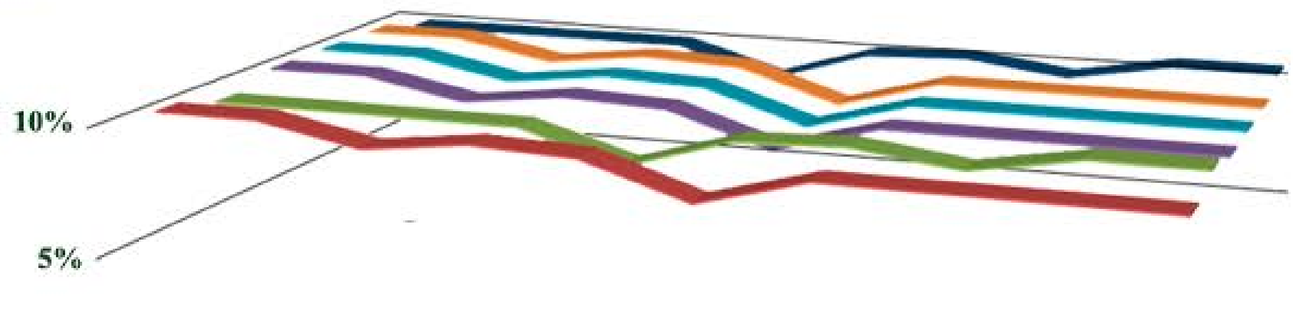



```
 fmt.Println("[YAM] Starting ProgramCounter...");
 go ProgramCounter (decodePc, dmuxPc, COPO_PC, aluPc, memoryDelay, pcMemory, pcMux, pcCopo)

 fmt.Println("[YAM] Starting Decode...");
 go Decode (memoryDecode, copoDecode, aluDecode, COPO, PC, ALU, REGISTER, MUX, DMUX, MDU)

...

 fmt.Println("[YAM] Starting ALU...");
 go Alu ( decodeAlu, dmuxAlu, memoryAlu, aluMemory, aluMux, aluCopo)

 fmt.Println("[YAM] Starting MUX...");
 go MuxRegister( decodeMux, aluMux, mduMux, pcMux, copoMux, muxRegister)

 fmt.Println("[YAM] Starting DMUX...");
 go Dmux (decodeDmux, decodeRegister,dmuxAlu, dmuxMdu, dmuxPc, dmuxCopo)

...

 fmt.Println("[YAM] Going to infinite loop..."); }
```



Resultados preliminares

Teste de Seleção Justa



Conclusões

- **Sucesso em teste realizados em pequenos circuitos.**
- **Validação do não-determinismo na linguagem escolhida.**
- **Possivelmente a linguagem "comercial" mais adequada para modelagem.**
- **Permite a modelagem e simulação de um processador completo.**

uergs
Universidade Estadual do Rio Grande do Sul

ENGENHARIA EM SISTEMAS DIGITAIS
Estrada Santa Maria, 2300 / Guaíba – RS
roger-pereira@uergs.edu.br
robim-araujo@uergs.edu.br
joao-fragoso @uergs.edu.br