Revisão de Implementações Abertas de Controladores de Memória SDRAM

Rodrigo Benfatti Aragão Leite

Julho de 2025

1 Introdução

No contexto do desenvolvimento de um emulador funcional para a arquitetura RISC-V, conforme descrito no Trabalho de Conclusão de Curso (TCC) [4], a pesquisa inicial envolveu a análise de controladores de memória SDRAM de implementação aberta. Esses controladores são componentes críticos em sistemas computacionais, responsáveis por gerenciar a comunicação entre a CPU e a memória SDRAM (*Synchronous Dynamic Random-Access Memory*), garantindo acesso eficiente e confiável aos dados. A escolha por implementações abertas alinha-se à filosofia da arquitetura RISC-V, que prioriza soluções livres de royalties e acessíveis para pesquisa acadêmica e desenvolvimento. Este resumo revisa as principais características, linguagens de implementação e recursos de controladores de memória SDRAM de código aberto pesquisados, destacando sua relevância para o projeto do emulador.

2 Importância dos Controladores de Memória SDRAM

Controladores de memória SDRAM gerenciam operações de leitura e escrita, sincronizando o acesso à memória com o clock do sistema. Eles lidam com aspectos como inicialização da memória, gerenciamento de tempos de acesso (latência CAS, tempo de atualização, entre outros), controle de bancos de memória e tratamento de comandos específicos, como REFRESH e PRECHARGE. Em sistemas baseados em RISC-V, esses controladores são essenciais para garantir a compatibilidade com diferentes tipos de SDRAM (como DDR, DDR2 ou SDR) e otimizar o desempenho em aplicações embarcadas ou de alto desempenho. A pesquisa de implementações abertas foi motivada pela necessidade de integrar um controlador eficiente ao emulador, inicialmente planejado para um *System-on-Chip* (SoC) em FPGA, mas posteriormente adaptado para uma implementação em software [4].

3 Controladores de Memória SDRAM Pesquisados

A seguir, apresentamos uma revisão de dois controladores de memória SDRAM de código aberto amplamente utilizados em projetos acadêmicos e industriais, analisados na fase inicial do TCC devido à sua relevância no ecossistema RISC-V e FPGA.

3.1 OpenCores SDRAM Controller

3.1.1 Características Principais

O controlador de memória SDRAM do OpenCores é um projeto consolidado, amplamente adotado em designs FPGA [3]. Suporta memórias SDR SDRAM (*Single Data Rate*) e é projetado para ser altamente configurável, permitindo ajustes em parâmetros como largura de dados (8, 16 ou 32 bits), número de bancos e tamanho da memória. O controlador gerencia comandos padrão, incluindo inicialização, leitura, escrita e *auto-refresh*, com suporte a *burst reads/writes* para melhorar a eficiência.

3.1.2 Linguagem de Implementação

Implementado em **Verilog**, uma linguagem de descrição de hardware (HDL) amplamente utilizada em projetos FPGA. A escolha do Verilog facilita a integração com ferramentas como Xilinx Vivado e Intel Quartus, comuns em plataformas FPGA.

3.1.3 Recursos

- Suporte a diferentes tamanhos de *burst* (1, 2, 4 ou 8 palavras).
- Configuração flexível de tempos de acesso (latência CAS, tRCD, tRP).
- Interface Wishbone para comunicação com outros componentes do sistema.
- Mecanismo de *auto-refresh* programável para manter a integridade dos dados.
- Documentação detalhada, mas com suporte limitado a memórias DDR mais modernas.

3.1.4 Observações

O OpenCores SDRAM Controller é ideal para projetos educacionais e protótipos devido à sua simplicidade e robustez. Contudo, sua limitação a SDR SDRAM o torna menos adequado para aplicações que exigem memórias de maior largura de banda, como DDR3 ou DDR4.

3.2 MiSTer SDRAM Controller

3.2.1 Características Principais

O controlador de memória SDRAM do projeto MiSTer é otimizado para sistemas retrocomputacionais e emulação de consoles clássicos em FPGA [1, 2]. É projetado para trabalhar com memórias SDR SDRAM de baixo custo, comuns em placas FPGA como a DE10-Nano. Prioriza simplicidade e eficiência, com foco em aplicações que requerem baixa latência e acesso previsível à memória, sendo essencial para núcleos como Neo Geo e Game Boy Advance [1].

3.2.2 Linguagem de Implementação

Implementado em **VHDL**, uma linguagem de descrição de hardware conhecida por sua robustez e verificação formal, sendo uma escolha comum em projetos que exigem alta confiabilidade.

3.2.3 Recursos

- Suporte a memórias SDR SDRAM com largura de dados de 16 ou 32 bits.
- Interface simples para integração com núcleos de emulação.
- Gerenciamento eficiente de comandos de *refresh* e inicialização.
- Suporte a acesso sequencial e aleatório à memória.
- Compatibilidade com placas FPGA de baixo custo, como a DE10-Nano.

3.2.4 Observações

O MiSTer SDRAM Controller é menos flexível que outros controladores, mas sua simplicidade o torna ideal para projetos com requisitos específicos de retrocomputação. Sua implementação em VHDL facilita a integração com ferramentas de síntese FPGA, mas o suporte a memórias DDR mais recentes é limitado [2].

4 Comparação e Relevância para o TCC

Os controladores analisados apresentam características distintas que influenciaram as decisões de design no desenvolvimento do emulador RISC-V. O **OpenCores SDRAM Controller** destacou-se pela simplicidade e adequação a projetos educacionais [3], mas sua limitação a SDR SDRAM restringiu sua aplicabilidade em cenários de maior desempenho. O **MiSTer SDRAM Controller** foi relevante para aplicações específicas de baixa latência, oferecendo uma solução eficiente para sistemas retrocomputacionais [1, 2].

No contexto do TCC [4], a escolha inicial por controladores de memória SDRAM visava suportar a implementação de um SoC em FPGA. Contudo, a transição para um emulador em software (Java) exigiu a virtualização da memória, como descrito na classe `Memory` do emulador. A pesquisa inicial dos controladores contribuiu para a compreensão dos requisitos de gerenciamento de memória, como tratamento de exceções (`MemoryException`) e acesso eficiente via barramento (`Bus`), que foram adaptados para o ambiente simulado.

5 Conclusão

A revisão dos controladores de memória SDRAM de implementação aberta revelou a diversidade de abordagens disponíveis para gerenciar memórias dinâmicas em sistemas baseados em RISC-V. Cada controlador apresentou vantagens específicas, desde a simplicidade do OpenCores [3] até a eficiência do MiSTer para aplicações específicas [1, 2], influenciando o design do emulador desenvolvido no TCC. A escolha por linguagens como Verilog e VHDL reflete as práticas modernas em design de hardware, enquanto os recursos como suporte a *burst*, interfaces padrão (Wishbone) e gerenciamento de temporização garantem eficiência e confiabilidade. Esta análise fundamentou as decisões técnicas do projeto e destacou o potencial das implementações abertas para a inovação em arquitetura de computadores.

Referências

[1] MiSTer Project. Main mister binary and wiki. https://github.com/MiSTer-devel/Main_MiSTer, 2024. Acessado em: 10 de outubro de 2024.

[2] MiSTer Project. Mister sdram controller. https://github.com/MiSTer-devel/Main_MiSTer/tree/master/rtl/sdram, 2024. Acessado em: 10 de outubro de 2024; link não mais ativo.

[3] OpenCores. 8/16/32 bit sdram controller. https://opencores.org/projects/sdram_cntl, 2019. Acessado em: 10 de outubro de 2024.

[4] Gabriel Faustino de Freitas Silva, João Pedro de Melo Roberto, and Rodrigo Benfatti Aragão Leite. Implementação de emulador funcional de sistema computacional baseado em arquitetura risc-v. Trabalho de Conclusão de Curso, Universidade Federal de Mato Grosso do Sul, 2024.