Lógica Digital (1001351)

Experimento NR.8

Memória de Acesso Aleatório (endereçável) [1] [2]

1 Instruções Gerais

- Ler atentamente todo o procedimento desta experiência antes de realizá-la;

2 Objetivos da Prática

- Implementar uma memória endereçável adotando a estratégia estrutural.
- Endender os mecanismos de leitura e escrita em memória.
- Entender o mecanismo de endereçamento.
- Observar os mecanismos síncronos e assíncronos;
- Implementar do projeto no Kit de desenvolvimento Zybo Z7-20 (opcional).

3 Materiais, Equipamentos e Arquivos

- Simulador EDA Playground;
- Kit de desenvolvimento Zybo Z7-20.

4 Fundamentos teóricos

O texto que se segue possui caráter meramente introdutório. Texto adicional pode ser alcançado na referência básica da disciplina [1].

De forma simplificada, uma memória endereçável corresponde a um dispositivo (circuito lógico) que permite armazenamento de vetores de bits (dados), cada qual (vetor) associado a um outro vetor de bits (endereço) utilizado para posterior recuperação do dado armazenado naquele endereço associado. A cada dado armazenado está associado um único endereço. Um outro termo que surge nesse contexto é o de posição de memória. Assim, um dado está armazenado em uma posição de memória definida por um endereço.

[1] Em inglês se diz *random access memory* (RAM).
[2] Revisão 15/10/2021: Prof. Mauricio Figueiredo e Prof. Ricardo Menotti.

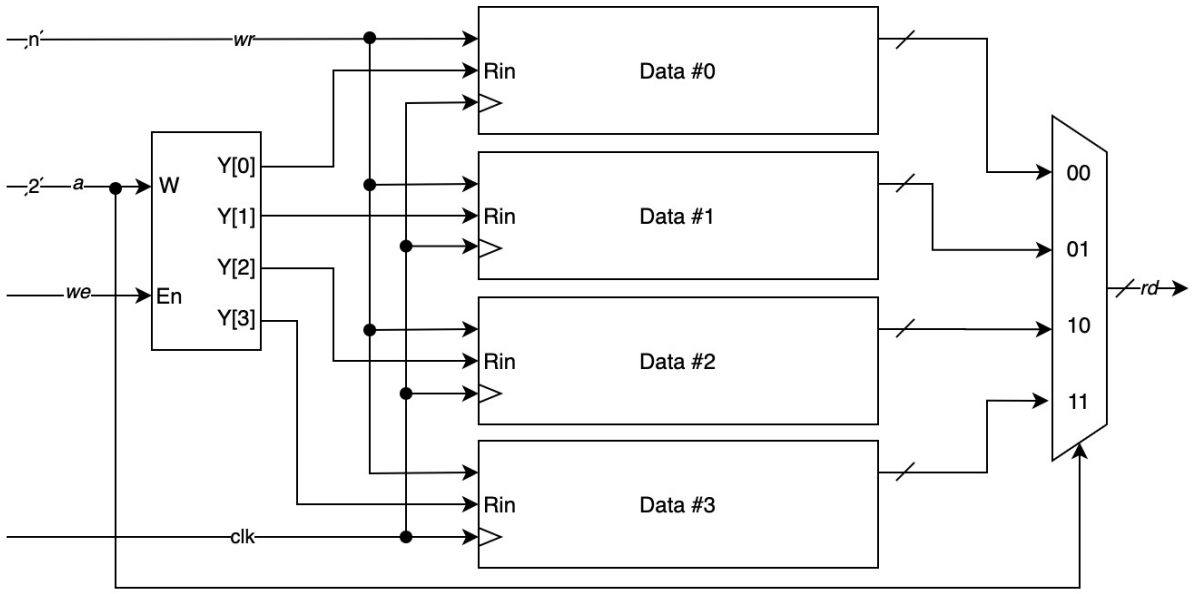

Figura 1: Memória de acesso aleatório (RAM)

Cada endereço corresponde a uma posição da memória, como em um vetor ou arranjo declarado em software.

Em uma operação de escrita (armazenamento), apresentam-se ao circuito: dado, endereço, sinal que indica o tipo de operação, no caso "escrita", e o sinal de relógio, sendo que o dado, o endereço e o sinal de escrita, devem estar presentes para o circuito antes do sinal de relógio. Assim, percebe-se que a operação de escrita é síncrona.

Em uma operação de leitura, basta que seja apresentado o endereço. Imediatamente o dado armazenado associado ao endereço apresentado passa a estar disponível na saída do circuito.

A partir das dinâmicas das operações de escrita e leitura, percebe-se que em uma operação de escrita, em geral, a saída da memória é alterada duas vezes.

As funcionalidades desse circuito, leitura e escrita (armazenamento), podem ser alcançadas a partir de uma composição conveniente de dispositivos conforme a Figura 1.

- REGISTRADORES: Os dados estão armazenados nos registradores: cada dado é armazenado em um registrador (Data #n).
- DECODIFICADOR: Em uma operação de escrita, o dado é armazenado em um registrador específico definido pelo endereço. Cada uma das saídas do decodificador está associado a um registrador específico. Sendo assim, a saída do decodificador tem o papel de identificar o registrador destino do dado (onde dever ser armazenado) a partir do endereço apresentado, juntamente com o sinal de escrita. Em essência, o decodificador seleciona um dos registradores, aquele correspondente ao endereço, para que o dado seja armazenado na subida do relógio. Assim, nota-se que se não houver um sinal de escrita, mesmo que o endereço, o dado e o sinal de relógio estejam eventualmente presentes na entrada do dispositivo de memória, nada é armazenado. A presença do sinal de escrita é imprescindível para garantir que, em um

procedimento de leitura, nenhum registrador tenha seu conteúdo alterado (atentar para o procedimento de leitura e os dispositivos associados, tal como descrito em seguida e também que em geral sempre há um dado na entrada do dispositivo de memória).

- MULTIPLEXADOR: Em uma operação de leitura, o dado armazenado em um dos registradores deve ser apresentado na saída do dispositivo de memória. Para tanto, pode-se adotar um multiplexador cujo número de entradas é igual ao número de registradores (um conjunto de *buffers tristate* também poderia ser usado, mas veremos isso mais adiante em nosso curso). A entrada de seleção corresponde ao endereço referente à posição de memória, ou seja, correspondente ao registrador onde o dado está armazenado.

5 Procedimentos Experimentais

Um modelo de implementação está disponível no EDA Playground para esta prática. Nele há uma implementação de referência para uma RAM, um "top level" incompleto e os três componentes mencionados disponíveis para compor uma RAM alternativa com funcionamento idêntico. Também há um *test bench* que realiza escritas sucessivas em ambas e depois faz leituras, conferindo os valores.

A partir do modelo fornecido, realize os seguintes experimentos:

- Complete o módulo "top level" (`my_ram`), instanciando e ligando os componentes mencionados na Seção 4 de acordo com a Figura 1. Verifique se o seu funcionamento está correto.

- Explique por que e em que momentos a saída da memória é modificada duas vezes durante a operação de escrita.

- A partir do parâmetro `width` do *test bench*, modifique o número de bits dos registradores e, consequentemente, das memórias. Este parâmetro deve ser propagado para os demais módulos internos sem a necessidade de sua alteração. Para tal, sua implementação deve instanciar os módulos internos passando o parâmetro recebido do *test bench*.

- A partir do parâmetro `log2_depth` do *test bench*, modifique a profundidade das memórias. Este parâmetro determina a largura do endereço e, consequentemente, o número de posições de memória (profundidade), tal que $profundidade = 2^{largura}$. Note que não é possível propagar automaticamente este valor na implementação atual.

- **Bonus:** você consegue modificar o código para que também este parâmetro fique transparente/automático para qualquer profundidade?

- Elaborar relatório simplificado consistindo de: página de rosto (com identificação da prática e integrantes entre outras informações, tal como no relatório padrão) e

seção de resultados contendo: diagramas esquemáticos dos circuitos, link para a implementação no EDA Playground, diagrama de formas de onda (*waveform*) da simulação completa, captura de imagem da placa (opcional) e comentários com as explicações solicitadas.

- Envie o relatório em PDF no AVA.

Referências Bibliográficas

[1] S. Brown and Z. Vranesic, *Fundamentals of Digital Logic with Verilog Design.* McGraw Hill, 2000.