**Tarefa 3** – Banco de registradores

| |
|---|
| **Descrição geral do problema**: Implemente um banco com 8 registradores de 8 bits. Esse banco deverá possuir um barramento de escrita síncrono (dados + endereço) e dois barramentos de leitura combinacionais (dados + endereços) |

**Descrição detalhada:**

1. Faça a descrição de hardware de um módulo, denominado RegisterFile, que gere o circuito indicado na Figura 1, usando a linguagem SystemVerilog.

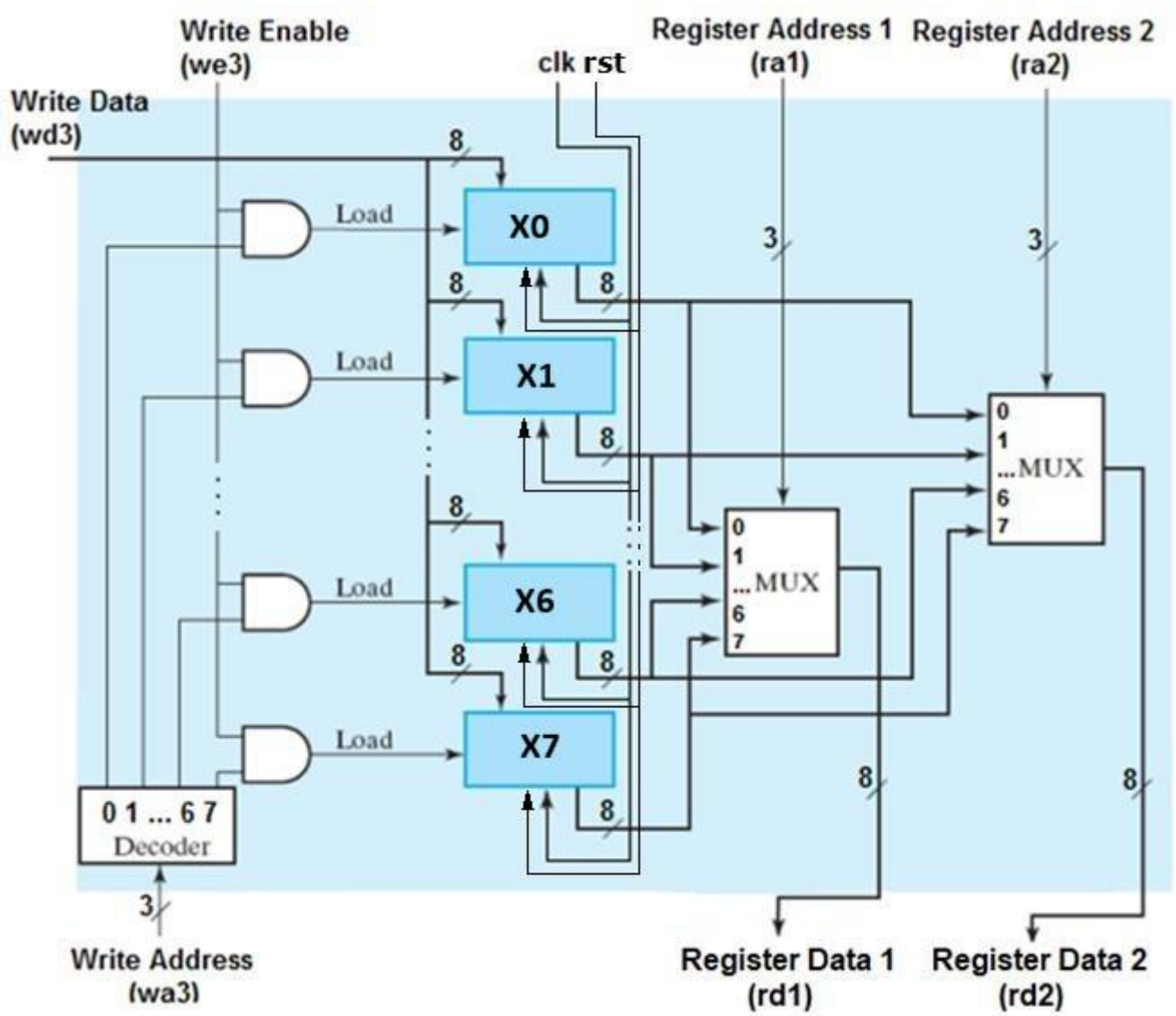


Figura 1 –Diagrama do módulo RegisterFile.

O módulo apresenta as seguintes entradas e saídas:

**Entradas**: *Write Data (**wd3**)* (8 bits) – Entrada de dados;
*Write Address (**wa3**)* (3 bits) – Seleção do registrador que armazenará o dado proveniente de *Write Data (wd3)*;
*Write Enable (**we3**)* (1 bit) – Habilita (1) ou desabilita (0) a gravação de dados nos registradores de X1 a X7; *clk* (1 bit) – Se o sinal *Write Enable (we3)* estiver ativo (1), na borda de subida do clock, o dado é gravado no registrador selecionado.
*Register Address 1 (**ra1**)* (3 bits) – Seleção de qual registrador será disponibilizado na saída rd1;
*Register Address 2 (**ra2**)* (3 bits) – Seleção de qual registrador será disponibilizado na saída rd2;
Reset (**rst**) (1 bit) – Reseta o valor dos registradores, em nível baixo (0);

**Saídas**: *Register Data 1 (**rd1**)* (8 bits) – Barramento de saída de 8 bits;
*Register Data 2 (**rd2**)* (8 bits) – Barramento de saída de 8 bits;

Consideração Importante: O circuito não precisa ficar idêntico à figura, modularizado em decodificadores e portas AND. O funcionamento é que deve ser preservado. Faça a descrição em alto nível, para ganhar produtividade.

Princípio de funcionamento em três partes:

a) **ESCRITA**: A operação de escrita deve ser sincronizada com a borda de subida do clock (clk). O valor de 8 bits de *Write D/ata (wd3)* é armazenado no registrador cujo índice é igual a *Write Address (wa3)* (3 bits) se *Write Enable (we3)* for 1. O valor dos registradores se mantém caso *Write Enable (we3) seja* 0. *Circuito sequencial síncrono => always @(posedge clock) ou always_ff @(posedge clock)*. **OBS: O registrador X0 é somente de LEITURA e seu valor deve ser fixo em ZERO.**
b) **LEITURA**: o valor do registrador de índice *ra1* é disponibilizado de forma contínua na saída *rd1* assim como o registrador de índice *ra2* na saída *rd2*. Variando qualquer entrada, as saídas são atualizadas. *Circuito Combinacional => always @(*), always_comb ou assign.*
c) **RESET**: limpa o valor dos registradores quando a entrada rst for 0. Decida se implementará essa funcionalidade de forma síncrona (no momento da subida do clock) ou assíncrona (na descida do rst).

**Os dois multiplexadores que conectam os registradores às saídas *rd1* e *rd2* são circuitos combinacionais. Os dados gravados nos registradores selecionados, sempre estarão disponíveis nas respectivas saídas.**

2. Durante a instanciação do register_file, no módulo de testbench *"register_file_tb",* instanciando o bloco desenvolvido, que possui a seguinte interface:

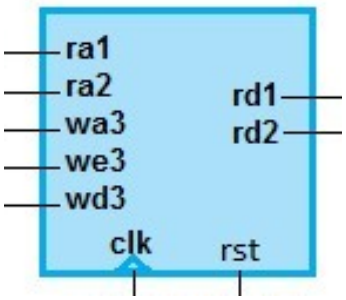

Figura 2 – Interfaces do módulo RegisterFile

3. Para avaliar o funcionamento do circuito, o testbench deve realizar a gravação de um conjunto de dados nos registradores. Após a gravação, o testbench deve verificar se os dados foram de fato gravados, selecionando *ra1* e *ra2* e observando as saídas *rd1* e *rd2.*
4. Após o testbench com básico criado no item anterior, tente desenvolver um testbench auto-verificado para o módulo desenvolvido e simule exaustivamente com as operações de escrita e leitura em cada um dos registradores do banco. Faça testes com verificação automática para todos os 8 registradores.