Trabalho 5 – Organização e Arquitetura de Computadores

Banco de Registradores do RISC-V

Aluno(a): Júlia Yuri Garcia Baba
Matrícula: 19/0057921
Turma C

1. Objetivos

O trabalho tem como objetivo sintetizar e simular um banco de registradores, da mesma for utilizada no RISC-V, através da linguagem VHDL.

2. Documentação do código

- **XREG**

Primeiramente, foi usada a interface deixada nas instruções do trabalho, assim, foram definidos os elementos utilizados posteriormente.

Para a arquitetura criou-se e instanciou-se um tipo de dado (reg), através do comando *type* e *array*, representando um vetor de palavras de 32 bits.

Em seguida, foi criado um processo para o clock, dessa forma, temos algumas possíveis ações:

1. Se temos um flanco de subida e o comando reset é acionado, conteúdo de todos os registradores do banco é zerado
2. Se temos um flanco de subida e o comando wren é acionado, a escrita é habilitada e o registrador endereçado por rd é escrito com o valor presente no barramento data. Note que nessa etapa foi feito uma condição para simular a constante zero em XREGS[0], já que a instrução de escrita só ocorre caso o valor de rd seja diferente de zero,
3. Se nenhum desses casos ocorrer, ro1 e ro2, que são as portas de saída para leitura do registrador endereçado , são escritos com o valor que está dentro do endereço apontado por rs1 e rs2.

- **Testbench**

Já para os testes, feito os procedimentos padrões do testbench, como a definição dos componentes, dos sinais e do port map, foi instanciado o sinal de clock, que altera seu estado a cada 5 ns, o que permite, também, a alteração dos elementos.

Partindo para o processo de estímulo, primeiro foi trabalhado o caso em que XREGS[0]. Sendo assim, escrevemos a lemos, e em seguida testamos, caso ro1 ou ro2 seja diferente de zero o erro é identificado e exibido na tela.

Já para os demais, foi criado um loop, com funcionamento similar ao descrito acima, porém é importante notar, que no começo do processo foi criado uma variável *(random)*, com valor inicial 1, essa é incrementada em cinco unidades e atribuída ao barramento *data*, a cada ciclo do loop. O mesmo acontece para o *rd*, porém esse tem seu valor atribuído pelo contador (*i*). Feito isso, testamos, através do comando *assert*, se os valores recebidos são os mesmos, caso seja identificado algum erro, este é exibido na tela.

Por fim, foi testado a função de reset, usando o mesmo procedimento descrito, contudo não há necessidade de testar ambas portas de saída, por isso apenas ro1 foi utilizada. De modo sucinto, foi criado outro loop, que escreve e lê, assim como, testa caso ro1 seja diferente de zero, identificando o erro.