Trabalho 5 de Arquitetura e Organização de Computadores – Turma C
Aluno: Marcos Paulo Cayres Rosa
Matrícula: 140027131

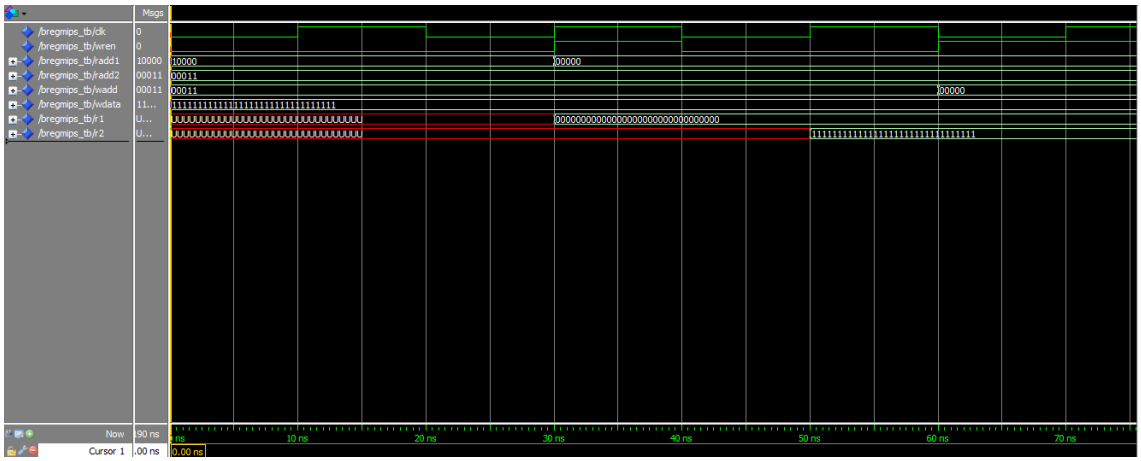
Projeto do Banco de Registradores do MIPS

A partir de uma interface de banco de registradores especificada, elaborar um código VHDL que permita receber com entrada o endereço dos registradores a serem lidos e uma possível habilitação de escrita com o dado a ser escrito correspondente. Ademais, possui como saída a leitura dos registradores determinados e está em sincronia com um relógio.

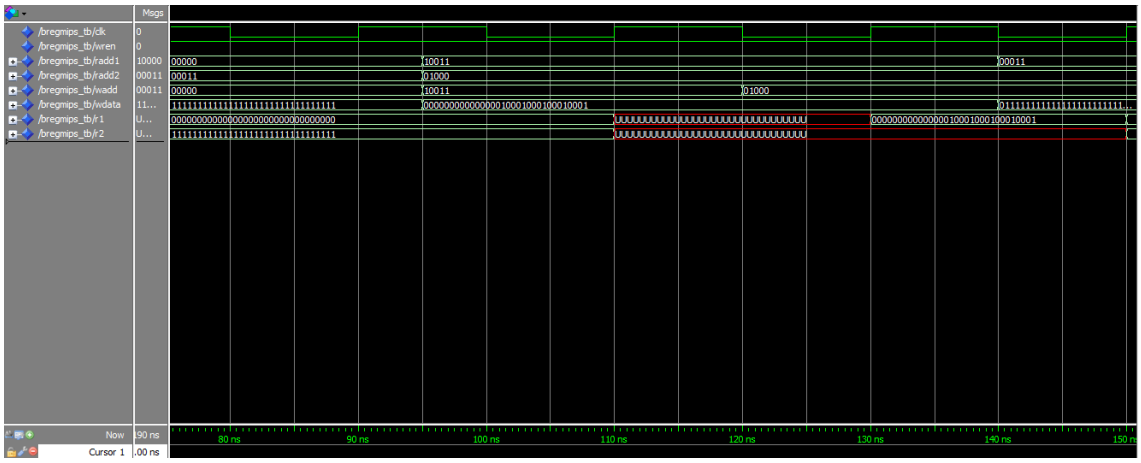
Com as determinações acima, alguns fatores tiveram que ser considerados:

- A leitura e a escrita são feitas na subida do relógio e nessa ordem. No caso, caso ocorra leitura e escrita no mesmo ciclo, inicia este com a leitura e, posteriormente, a escrita. Sendo que nos próximos ciclos, se o registrador com o qual se fez a operação de escrita for lido, possuirá o valor especificado nesse processo.

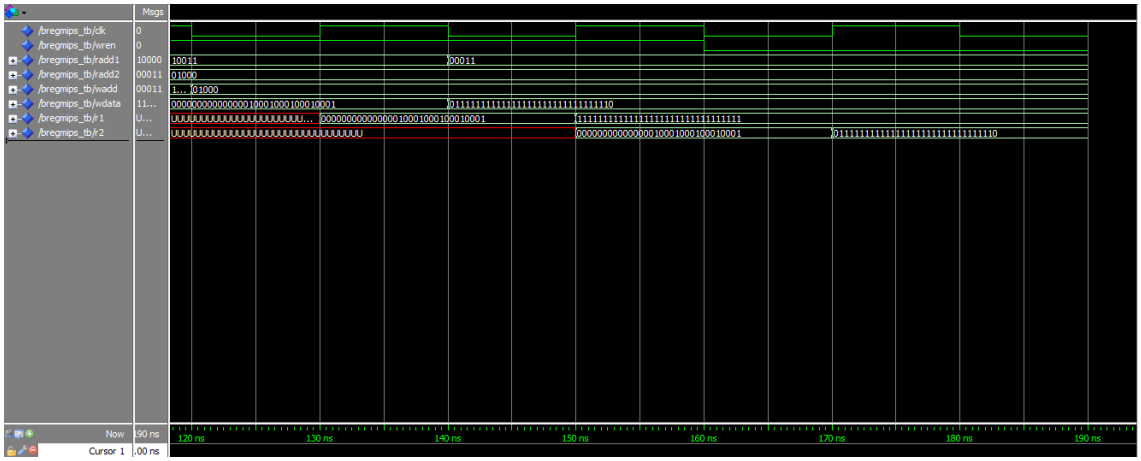
Imagens da execução do test bench para o ModelSim:



- Na primeira batida do relógio foi verificada a leitura dos registradores 16 e 3, recebendo o valor inicial estabelecido para os campos do banco (nulo).
- Em seguida, com a escrita habilitada, verificou que o registrador 0 retorna zero e operou a leitura e, então, a escrita no registrador 3.
- Com o registrador 3 modificado e a escrita desativada, foi lido seu valor, retornando o mesmo que foi especificado para escrita no ciclo anterior.

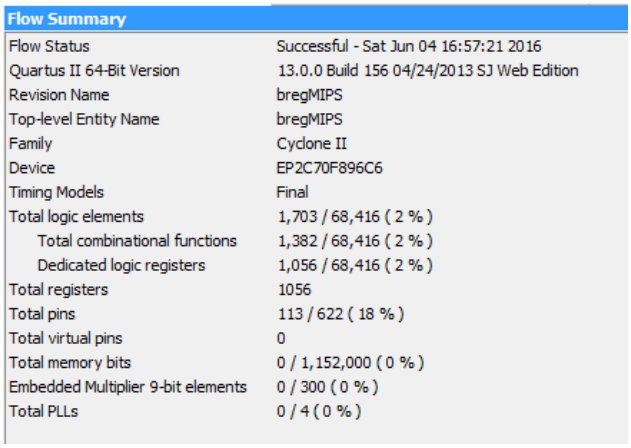


- Houve a tentativa de escrever no registrador 0, porém esta não foi permitida, como ele se mantendo sempre zero.
- Também houve tentativa de determinar novas operações após o relógio ter subido e antes de completar seu ciclo, porém só são consideradas as entradas quando estiver no momento de subida e não após este. Nessa mesma lógica, houve outros testes com as entradas sendo especificadas no momento da subida (funcionando normalmente) ou da descida (esperando a próxima subida para operar).
- Foi feita a leitura dos registradores 19 e 8, retornando inicialmente o valor nulo. E, depois da escrita, os valores indicados para cada.



- Foi efetuada uma segunda operação de escrita do registrador 8, com a leitura deste no ciclo seguinte indicando o novo valor, e uma nova operação de leitura do registrador 3 (primeiro que foi modificado), com este mantendo seu valor.

Imagem da síntese (com o número de células lógicas apresentado)**:**



Flow Summary	
Flow Status	Successful - Sat Jun 04 16:57:21 2016
Quartus II 64-Bit Version	13.0.0 Build 156 04/24/2013 SJ Web Edition
Revision Name	bregMIPS
Top-level Entity Name	bregMIPS
Family	Cyclone II
Device	EP2C70F896C6
Timing Models	Final
Total logic elements	1,703 / 68,416 (2 %)
Total combinational functions	1,382 / 68,416 (2 %)
Dedicated logic registers	1,056 / 68,416 (2 %)
Total registers	1056
Total pins	113 / 622 (18 %)
Total virtual pins	0
Total memory bits	0 / 1,152,000 (0 %)
Embedded Multiplier 9-bit elements	0 / 300 (0 %)
Total PLLs	0 / 4 (0 %)

bregMIPS.vhd:

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity bregMIPS is
	generic (WSIZE : natural := 32);
	port (
			clk, wren : in std_logic;
			radd1, radd2, wadd : in std_logic_vector(4 downto 0);
			wdata : in std_logic_vector(WSIZE-1 downto 0);
			r1, r2 : out std_logic_vector(WSIZE-1 downto 0));
end bregMIPS;

architecture bregArch of bregMIPS is

type regFile is array(0 to WSIZE-1) of std_logic_vector(WSIZE-1 downto 0);
signal regs : regFile;

begin
	process (clk) is

	begin
		if rising_edge(clk) then
			regs(0) <= (others => '0');
			r1 <= regs(to_integer(unsigned(radd1)));
			r2 <= regs(to_integer(unsigned(radd2)));
			if wren = '1' and 0 /= to_integer(unsigned(wadd)) then
				regs(to_integer(unsigned(wadd))) <= wdata;
			end if;
		end if;

	 end process;

end bregArch;
```

bregMIPS_tb.vhd:

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY bregMIPS_tb IS
END bregMIPS_tb;
ARCHITECTURE bregMIPS_tbArch OF bregMIPS_tb IS
	-- constants
	-- signals
	SIGNAL clk : STD_LOGIC;
	SIGNAL wren : STD_LOGIC;
	SIGNAL radd1 : STD_LOGIC_VECTOR(4 DOWNTO 0);
	SIGNAL radd2 : STD_LOGIC_VECTOR(4 DOWNTO 0);
	SIGNAL wadd : STD_LOGIC_VECTOR(4 DOWNTO 0);
	SIGNAL wdata : STD_LOGIC_VECTOR(31 DOWNTO 0);
	SIGNAL r1 : STD_LOGIC_VECTOR(31 DOWNTO 0);
	SIGNAL r2 : STD_LOGIC_VECTOR(31 DOWNTO 0);
	COMPONENT bregMIPS
		PORT (
		clk : IN STD_LOGIC;
		wren : IN STD_LOGIC;
		radd1 : IN STD_LOGIC_VECTOR(4 DOWNTO 0);
		radd2 : IN STD_LOGIC_VECTOR(4 DOWNTO 0);
		wadd : IN STD_LOGIC_VECTOR(4 DOWNTO 0);
		wdata : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
		r1 : OUT STD_LOGIC_VECTOR(31 DOWNTO 0);
		r2 : OUT STD_LOGIC_VECTOR(31 DOWNTO 0)
		);
	END COMPONENT;
	BEGIN
		i1 : bregMIPS
		PORT MAP (
	-- list connections between master ports and signals
		clk => clk,
		wren => wren,
		radd1 => radd1,
		radd2 => radd2,
		wadd => wadd,
		wdata => wdata,
		r1 => r1,
		r2 => r2
		);
	PROCESS
	-- variable declarations
	BEGIN
		clk	<= '0';
		wren	<= '0';
		radd1	<= (4 => '1', others => '0'); -- reg(16)
		radd2	<= (0 | 1 => '1', others => '0'); -- reg(3)
```

```
wadd   <= (0 | 1 => '1', others => '0'); -- reg(3)
wdata  <= (others => '1');
wait for 10 ns;

clk    <= '1';
wait for 10 ns;

clk    <= '0';
wait for 10 ns;

clk    <= '1';
wren   <= '1';
radd1  <= (others => '0'); -- reg(0)
wait for 10 ns;

clk    <= '0';
wren   <= '0';
wait for 10 ns;

clk    <= '1';
wait for 10 ns;

clk    <=     '0';
wren   <= '1';
wadd   <= (others => '0');
wait for 10 ns;

clk    <= '1';
wait for 10 ns;

clk    <= '0';
wait for 10 ns;

clk    <= '1';
wait for 5 ns;

radd1  <= (0 | 1 | 4 => '1', others => '0'); -- reg(19)
radd2  <= (3 => '1', others => '0'); -- reg(8)
wadd   <= (0 | 1 | 4 => '1', others => '0'); -- reg(19)
wdata  <= (0 | 4 | 8 | 12 | 16 => '1', others => '0');
wait for 5 ns;

clk    <= '0';
wait for 10 ns;

clk    <= '1';
wait for 10 ns;

clk    <= '0';
wadd   <= (3 => '1', others => '0'); -- reg(8)
```

```
        wait for 10 ns;

        clk      <= '1';
        wait for 10 ns;

        clk      <= '0';
        radd1  <= (0 | 1 => '1', others => '0'); -- reg(3)
        wdata <= (0 | 31 => '0', others => '1');
        wait for 10 ns;

        clk      <= '1';
        wait for 10 ns;

        clk      <= '0';
        wren   <= '0';
        wait for 10 ns;

        clk      <= '1';
        wait for 10 ns;

        clk      <= '0';
        wait for 10 ns;
        wait;


    END PROCESS;
END bregMIPS_tbArch;
```