Πανεπιστήμιο Δυτικής Αττικής
Τμήμα Μηχανικών Πληροφορικής και Υπολογιστών
Σχεδίαση Ψηφιακών Συστημάτων (ICE – 4005) Project_MIPS _2020
Γκούσαρης Κωνσταντίνος – 711171073 – cs171073@uniwa.gr

5: Register File

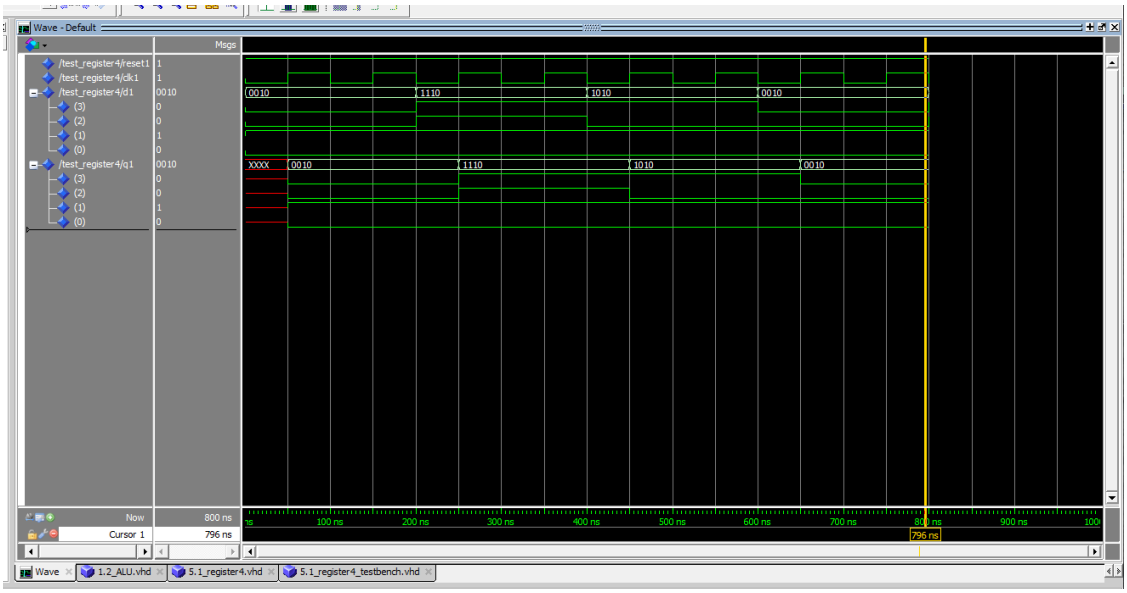
5.1 register4.vhd

```
--MIPS Part_5
--Register 4bit
--27/05/2020, Konstantinos Gkousaris, 711171073, UniWA--

LIBRARY ieee;
USE ieee.std_logic_1164.all;


ENTITY register4 IS PORT (
d : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
resetn, clk : IN STD_LOGIC;
q : OUT STD_LOGIC_VECTOR(3 DOWNTO 0));
END register4;


ARCHITECTURE behavioral OF register4 IS
BEGIN
            PROCESS(resetn, clk)
            BEGIN
                  IF resetn = '0' THEN
                        q <= "0000";
            ELSIF rising_edge(clk) THEN
                        q <= d;
            END IF;
            END PROCESS;
END behavioral;
```

5.1 register4_testbench.vhd

```
--test bench code
--runs for 800ns
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY test_register4 IS
END test_register4;

ARCHITECTURE behavioral OF test_register4 IS
      COMPONENT register4 PORT(
            d : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
            resetn,clk : IN STD_LOGIC;
            q : OUT STD_LOGIC_VECTOR(3 DOWNTO 0));
      END COMPONENT;
```

```
	signal reset1 : STD_LOGIC:='1';
	signal clk1 : STD_LOGIC;
	signal d1 : STD_LOGIC_VECTOR(3 DOWNTO 0);
	signal q1 : STD_LOGIC_VECTOR(3 DOWNTO 0);
	constant clk_period : time := 100 ns;

	BEGIN
	REG4 : register4 PORT MAP (
			d => d1,
			q => q1,
			resetn => reset1,
			clk => clk1);
	--process for clock
	clk_process :PROCESS
	BEGIN
		clk1 <= '0';
		wait for clk_period/2;
		clk1 <= '1';
		wait for clk_period/2;
	END PROCESS;
	--process which controls the 'd' entry
	simulation : PROCESS
	BEGIN
		d1 <= "0010"; wait for 200 ns; --different entries every 200 ns
		d1 <= "1110"; wait for 200 ns;
		d1 <= "1010"; wait for 200 ns;
	END PROCESS;
END;
```



Πίνακας Λειτουργίας

In	Out
0010	0010
1110	1110
1010	1010

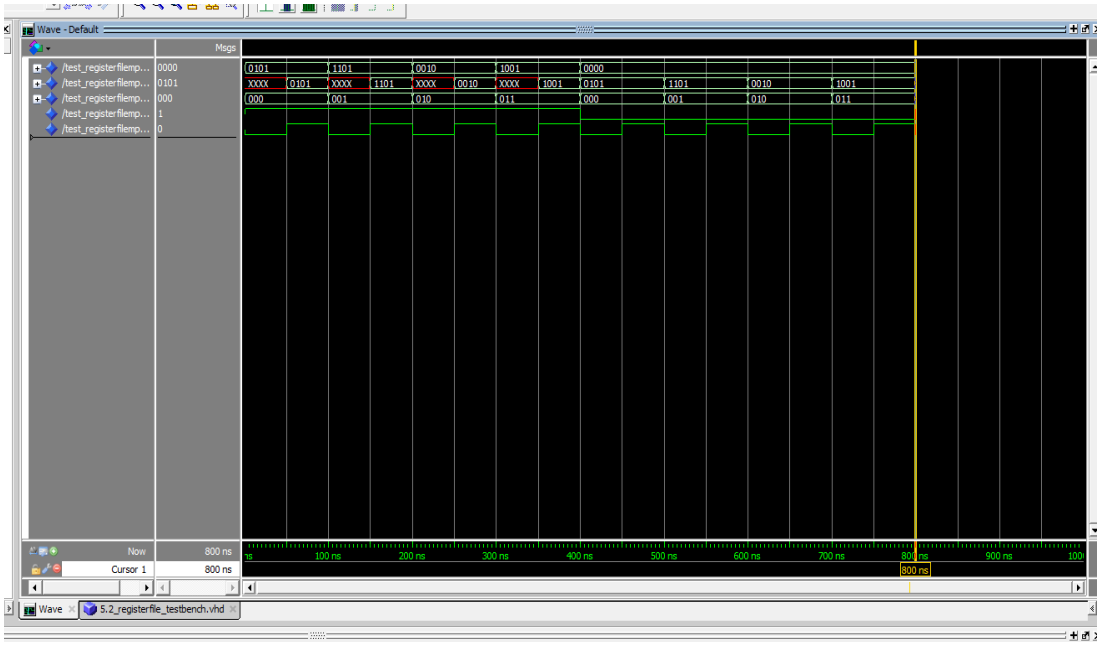
5.2_registerfile.vhd

```
--MIPS Part_5
--RegisterFile
--27/05/2020, Konstantinos Gkousaris, 711171073, UniWA--
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_unsigned.all;
USE ieee.numeric_std.ALL;


ENTITY registerfile IS PORT (
      Datain : IN STD_LOGIC_VECTOR(3 downto 0);
      Addr : IN STD_LOGIC_VECTOR(2 downto 0);
      we : IN STD_LOGIC;
      clk : IN STD_LOGIC;
      Dataout : OUT STD_LOGIC_VECTOR(3 downto 0));
END registerfile;

ARCHITECTURE behavioral OF registerfile IS

TYPE regArray IS ARRAY(0 TO 7) OF STD_LOGIC_VECTOR(3 DOWNTO 0);

      SIGNAL registerfile: regArray;
      BEGIN
      PROCESS(clk)
      BEGIN
            IF (clk'event and clk='0') THEN
                  IF we='1' THEN
                        registerfile(to_integer(unsigned(Addr))) <= Datain;
                  END IF;
            END IF;
                  Dataout <= registerfile(to_integer(unsigned(Addr)));
      END PROCESS;
END;
```

5.2_registerfile_testbench.vhd

```
--test bench code
--this test bench runs for 800 ns
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_unsigned.all;
USE ieee.numeric_std.ALL;

ENTITY test_registerfileMP IS
END test_registerfileMP;

ARCHITECTURE behavioral OF test_registerfileMP IS

      COMPONENT registerfile PORT(
            Datain : IN STD_LOGIC_VECTOR(3 downto 0);
            Addr : IN STD_LOGIC_VECTOR(2 downto 0);
            we : IN STD_LOGIC;
            clk : IN STD_LOGIC;
            Dataout : OUT STD_LOGIC_VECTOR(3 downto 0));
      END COMPONENT;
```

```
      --component signals
      signal Datain1,Dataout1 : std_logic_vector(3 downto 0);
      signal Addr1 : std_logic_vector(2 downto 0);
      signal we1,clk1 : std_logic;

      constant clk_period : time := 100 ns;

BEGIN
      REGISTERFILE1 : registerfile PORT MAP (
            Datain=>Datain1,
            Addr=>Addr1,
            we=>we1,
            clk=>clk1,
            Dataout=>Dataout1);
      --process for clock
      clk_process :PROCESS
      BEGIN
      clk1 <= '0';
      wait for clk_period/2;
      clk1 <= '1';
      wait for clk_period/2;
      END PROCESS;

      write_process : PROCESS
      BEGIN
            we1 <= '1'; wait for 400 ns;  --write for 400 ns
            we1 <= '0'; wait for 400 ns;  --read for 400 ns
      END PROCESS;

      simulation : PROCESS
      BEGIN
            -- write this data in memory
            Datain1 <= "0101"; wait for 100 ns;
            Datain1 <= "1101"; wait for 100 ns;
            Datain1 <= "0010"; wait for 100 ns;
            Datain1 <= "1001"; wait for 100 ns;
            --after specific inputs reset the datain
            Datain1 <= "0000"; wait;
      END PROCESS;

      address_process : PROCESS
      BEGIN
            --write the data on a specific address for 400ns
            addr1 <= "000"; wait for 100 ns;
      addr1 <= "001"; wait for 100 ns;
            addr1 <= "010"; wait for 100 ns;
            addr1 <= "011"; wait for 100 ns;
            --read the data from memory to check if writing process works good
            addr1 <= "000"; wait for 100 ns;
            addr1 <= "001"; wait for 100 ns;
            addr1 <= "010"; wait for 100 ns;
            addr1 <= "011"; wait for 100 ns;
      END PROCESS;
END;
```



Πίνακας Λειτουργίας

Διεύθυνση	Λειτουργία(we)	Περιεχόμενα(DataIN)
000	1 (write)	0101
001	1 (write)	1101
010	1 (write)	0010
011	1 (write)	1001
000	0 (read)	0101
001	0 (read)	1101
010	0 (read)	0010
011	0 (read)	1001

5.3_registerfile_full.vhd

```
--MIPS Part_5
--RegisterFileFull
--27/05/2020, Konstantinos Gkousaris, 711171073, UniWA--

LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_unsigned.all;
USE ieee.numeric_std.ALL;



ENTITY registerfile_full IS
      GENERIC (   dw : natural := 4;
                  size : natural := 4;
                  adrw : natural := 2);
```

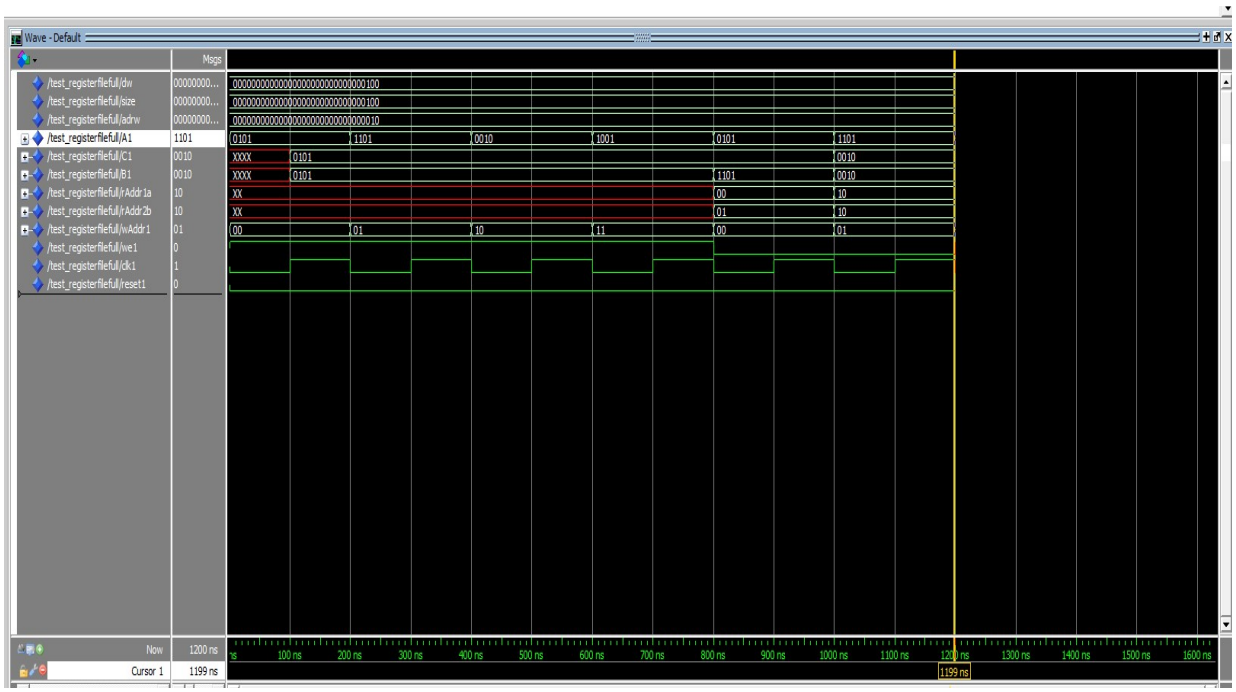
```
      PORT  (     Datain : IN STD_LOGIC_VECTOR(dw-1 downto 0);
                        rAddr1: IN STD_LOGIC_VECTOR(adrw-1 downto 0);
                        rAddr2: IN STD_LOGIC_VECTOR(adrw-1 downto 0);
                        wAddr : IN STD_LOGIC_VECTOR(adrw-1 downto 0);
                        we : IN STD_LOGIC;
                        clk :  IN STD_LOGIC;
                        reset : IN STD_LOGIC;
                        Dataout1 : OUT STD_LOGIC_VECTOR(dw-1 downto 0);
                        Dataout2 : OUT STD_LOGIC_VECTOR(dw-1 downto 0));
end registerfile_full;

ARCHITECTURE behavioral OF registerfile_full IS

TYPE regArray IS ARRAY(0 to size-1) OF std_logic_vector(dw-1 DOWNTO 0);
signal regfileb : regArray;

BEGIN
      PROCESS(clk)
      BEGIN
      IF reset= '1' THEN --reset entire circuit
            Dataout1 <= "0000" ;
            Dataout2 <= "0000" ;
      ELSIF (clk'event AND clk='0') then
            IF we='1' then
                  regfileb(to_integer(unsigned(wAddr))) <= Datain;
            END IF;
      END IF;
      Dataout1 <= regfileb(to_integer(unsigned(rAddr1)));
      Dataout2 <= regfileb(to_integer(unsigned(rAddr2)));
      END PROCESS;
END behavioral;
```

5.3 registerfile_full_testbench.vhd

```
--test bench code
--this test bench runs for 1200ns
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_unsigned.all;
USE ieee.numeric_std.ALL;

ENTITY test_registerfilefull IS
END test_registerfilefull;

ARCHITECTURE behavioral OF test_registerfilefull IS

      --variables for generic to pass into component
      signal dw : natural := 4;
      signal size : natural := 4;
      signal adrw : natural := 2;

      COMPONENT registerfile_full PORT(
            Datain : IN STD_LOGIC_VECTOR(dw-1 downto 0);
              rAddr1: IN STD_LOGIC_VECTOR(adrw-1 downto 0);
                rAddr2: IN STD_LOGIC_VECTOR(adrw-1 downto 0);
                wAddr : IN STD_LOGIC_VECTOR(adrw-1 downto 0);
                we : IN STD_LOGIC;
                clk :  IN STD_LOGIC;
                reset : IN STD_LOGIC;
                Dataout1 : OUT STD_LOGIC_VECTOR(dw-1 downto 0);
              Dataout2 : OUT STD_LOGIC_VECTOR(dw-1 downto 0));
      END COMPONENT;
```

```
      --component signals
      signal A1,C1,B1 : std_logic_vector(4-1 downto 0);--dw
      signal rAddr1a, rAddr2b, wAddr1 : std_logic_vector(2-1 downto 0);--adrw
      signal we1,clk1, reset1 : std_logic;
      constant clk_period : time := 200 ns;
BEGIN --this test bench runs for 1200ns
      REGISTERFILE_FULL1 : registerfile_full PORT MAP (
            Datain => a1,
            rAddr1 => rAddr1a,
            rAddr2 => rAddr2b,
            wAddr => wAddr1,
            we => we1,
            reset => reset1,
            clk => clk1,
            Dataout1 => c1,
            Dataout2 => b1);
      --process for clock

      clk_process :PROCESS
      BEGIN
            clk1 <= '0';
            wait for clk_period/2;
            clk1 <= '1';
            wait for clk_period/2;
      END PROCESS;

      write_process : PROCESS
      BEGIN
            we1 <= '1'; wait for 800 ns;  --write for 800 ns
            we1 <= '0'; wait for 400 ns;  --read for 400 ns
      END PROCESS;

      simulation : PROCESS
      BEGIN
            a1 <= "0101"; wait for 200 ns;  --different entries every 200ns
            a1 <= "1101"; wait for 200 ns;
            a1 <= "0010"; wait for 200 ns;
            a1 <= "1001"; wait for 200 ns;
      END PROCESS;

      write_address_process : PROCESS
      BEGIN
            wAddr1 <= "00"; wait for 200 ns; --change address every 200ns
            wAddr1 <= "01"; wait for 200 ns;
            wAddr1 <= "10"; wait for 200 ns;
            wAddr1 <= "11"; wait for 200 ns;
      END PROCESS;

      read1_address_process : PROCESS
      BEGIN
            wait for 800 ns;
            rAddr1a <= "00"; wait for 200 ns;
            rAddr1a <= "10"; wait for 200 ns;
      END PROCESS;

      read2_address_process : PROCESS
      BEGIN
            wait for 800 ns;
            rAddr2b <= "01"; wait for 200 ns;
            rAddr2b <= "10"; wait for 200 ns;
      END PROCESS;
```

```
	reset_prosess : PROCESS
	BEGIN
		reset1 <= '0'; wait for 1200 ns;
	END PROCESS;

END;
```



Πίνακας Λειτουργίας

Λειτουργία	Διευθυνσή	WR	DR1	DR2
write	00	0101	xxxx	xxxx
write	01	1101	xxxx	xxxx
write	10	0010	xxxx	xxxx
write	11	1001	xxxx	xxxx
read	00 & 10	0101	0101	0010
read	01 & 10	1101	1101	0010