Alexandria University

Faculty of Engineering

CSED 2025

Computer Architecture
Lab 1 Report

Submitted by

Marwan Yasser Sabri 20011870

Makario Michel Azer 20011982

Faculty of engineering Alex.
University

March 2024

Design Code:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.numeric_std.all;

entity register_file is
  port (
    clk : in  std_logic;
    regWrite : in  std_logic;
    dataIn : in  std_logic_vector(31 downto 0);
    readRegA : in  unsigned(4 downto 0);
    readRegB : in  unsigned(4 downto 0);
    writeReg : in  unsigned(4 downto 0);
    dataA : out std_logic_vector(31 downto 0);
    dataB : out std_logic_vector(31 downto 0)
  );
end register_file;

architecture Behavioral of register_file is
  type registerFile is array (0 to 31) of std_logic_vector(31 downto 0);
  signal registers : registerFile;
begin

  process(clk)
  begin
    if rising_edge(clk) then
      -- Write data in the first half cycle
      if regWrite = '1' then
        registers(to_integer(writeReg)) <= dataIn;
      end if;
    elsif falling_edge(clk) then
      -- Read data in the second half cycle
      dataA <= registers(to_integer(readRegA));
      dataB <= registers(to_integer(readRegB));
    end if;
  end process;
end Behavioral;
```

Testbench Code:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.numeric_std.all;

entity register_file_tb is
end register_file_tb;

architecture tb_arch of register_file_tb is
  constant CLK_PERIOD : time := 10 ns; -- Clock period (10 ns) -- clock per cycle

  signal clk_tb       : std_logic := '0'; -- Testbench clock signal
  signal regWrite_tb  : std_logic := '0'; -- Testbench regWrite signal
  signal dataIn_tb    : std_logic_vector(31 downto 0) := (others => '0'); -- Testbench data input signal
  signal readRegA_tb  : unsigned(4 downto 0) := to_unsigned(0, 5); -- Testbench readRegA signal
  signal readRegB_tb  : unsigned(4 downto 0) := to_unsigned(0, 5); -- Testbench readRegB signal
  signal writeReg_tb  : unsigned(4 downto 0) := to_unsigned(0, 5); -- Testbench writeReg signal
  signal dataA_tb     : std_logic_vector(31 downto 0); -- Testbench dataA output signal
  signal dataB_tb     : std_logic_vector(31 downto 0); -- Testbench dataB output signal

  -- Component declaration
  component register_file
    port (
      clk : in  std_logic;
      regWrite : in  std_logic;
      dataIn : in  std_logic_vector(31 downto 0);
      readRegA : in  unsigned(4 downto 0);
      readRegB : in  unsigned(4 downto 0);
      writeReg : in  unsigned(4 downto 0);
      dataA : out std_logic_vector(31 downto 0);
      dataB : out std_logic_vector(31 downto 0)
    );
  end component;
```

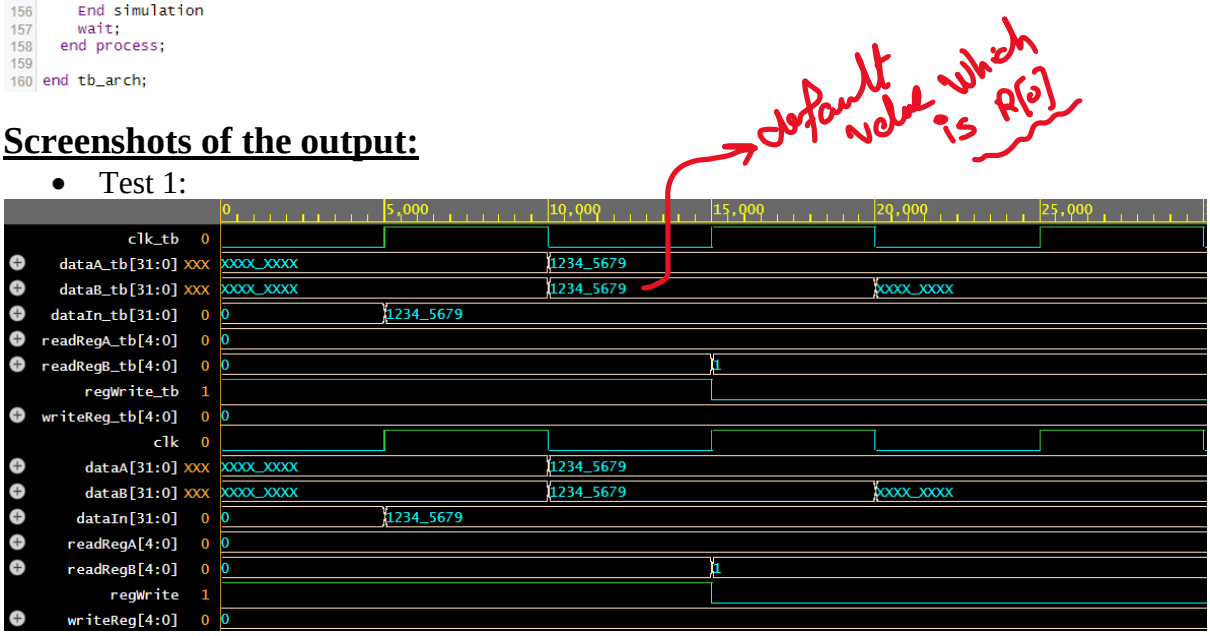
```
begin
    -- connect signals with ports --
  -- Instantiate the register_file entity
  dut : register_file
    port map (
      clk => clk_tb,
      regWrite => regWrite_tb,
      dataIn => dataIn_tb,
      readRegA => readRegA_tb,
      readRegB => readRegB_tb,
      writeReg => writeReg_tb,
      dataA => dataA_tb,
      dataB => dataB_tb
    );

  -- Clock process
  clk_process: process
  begin
    while now < 500 ns loop
      clk_tb <= '0';
      wait for CLK_PERIOD / 2;
      clk_tb <= '1';
      wait for CLK_PERIOD / 2;
    end loop;
    wait;
  end process;
```

```
  -- Stimulus process
  stimulus_process: process
  begin
    wait for CLK_PERIOD/2;
    -- Test case 1: Write to register 0, then read from it
    regWrite_tb <= '1';
    dataIn_tb <= x"12345678";
    writeReg_tb <= to_unsigned(0, 5);
    wait for CLK_PERIOD;
    regWrite_tb <= '0';
    readRegA_tb <= to_unsigned(0, 5);
    readRegB_tb <= to_unsigned(1, 5);
    wait for CLK_PERIOD;
    assert dataA_tb = dataIn_tb report "Test case 1 failed: Incorrect data read from register 0." severity failure;
    assert dataB_tb /= dataIn_tb report "Test case 1 failed: Incorrect data read from register 1." severity failure;

    -- Test case 2: Write to register 10, then read from it
    regWrite_tb <= '1';
    dataIn_tb <= x"ABCDEF01";
    writeReg_tb <= to_unsigned(10, 5);
    wait for CLK_PERIOD;
    regWrite_tb <= '0';
    readRegA_tb <= to_unsigned(10, 5);
    readRegB_tb <= to_unsigned(11, 5);
    wait for CLK_PERIOD;
    assert dataA_tb = dataIn_tb report "Test case 2 failed: Incorrect data read from register 10." severity failure;
    assert dataB_tb = dataIn_tb report "Test case 2 failed: Incorrect data read from register 11." severity failure;


    -- Test case 3: Not Write to register 5, then read from it
    regWrite_tb <= '0';
    dataIn_tb <= x"ABCDEF02";
    writeReg_tb <= to_unsigned(5, 5);       -- won't write
    wait for CLK_PERIOD;
    regWrite_tb <= '0';
    readRegA_tb <= to_unsigned(5, 5);
    readRegB_tb <= to_unsigned(5, 5);
    wait for CLK_PERIOD;
    assert dataA_tb /= dataIn_tb report "Test case 2 failed: Incorrect data read from register 10." severity failure;
    assert dataB_tb /= dataIn_tb report "Test case 2 failed: Incorrect data read from register 11." severity failure;

    -- Test case 4: Writing and reading from the same register in the same clock cycle
    regWrite_tb <= '1';
    dataIn_tb <= x"87654321";
    writeReg_tb <= to_unsigned(2, 5);       -- write first then read
    wait for CLK_PERIOD / 2;
    readRegA_tb <= to_unsigned(2, 5);
    readRegB_tb <= to_unsigned(2, 5);
    wait for CLK_PERIOD / 2;
    regWrite_tb <= '0';
    wait for CLK_PERIOD;
    assert dataA_tb = x"87654321" and dataB_tb = x"87654321" report "Test case 4 failed: Incorrect data read from register 20."
severity failure;


    -- Test case 5: (Not Writing) and reading from the same register in the same clock cycle
    regWrite_tb <= '0';
    dataIn_tb <= x"27654321";
    writeReg_tb <= to_unsigned(20, 5);      -- write first then read
    wait for CLK_PERIOD / 2;
    readRegA_tb <= to_unsigned(20, 5);
    readRegB_tb <= to_unsigned(20, 5);
    wait for CLK_PERIOD / 2;
    regWrite_tb <= '0';
    wait for CLK_PERIOD;
    assert dataA_tb /= x"27654321" and dataB_tb /= x"27654321" report "Test case 5 failed: Incorrect data read from register 20."
severity failure;

    -- Test 6: write data and read other data in the same clock cycle
    regWrite_tb <= '1';
    dataIn_tb <= x"47654321";
    writeReg_tb <= to_unsigned(21, 5);      -- write first then read
    wait for CLK_PERIOD / 2;
    readRegA_tb <= to_unsigned(20, 5);
    readRegB_tb <= to_unsigned(20, 5);
    wait for CLK_PERIOD / 2;
    regWrite_tb <= '0';
    wait for CLK_PERIOD;
    assert dataA_tb /= x"47654321" and dataB_tb /= x"87654321" report "Test case 6 failed: Incorrect data read from register 20."
severity failure;
    assert dataA_tb = dataIn_tb and dataB_tb = dataIn_tb report "Test case 6 failed: Incorrect data read from register 20." severity
failure;
```

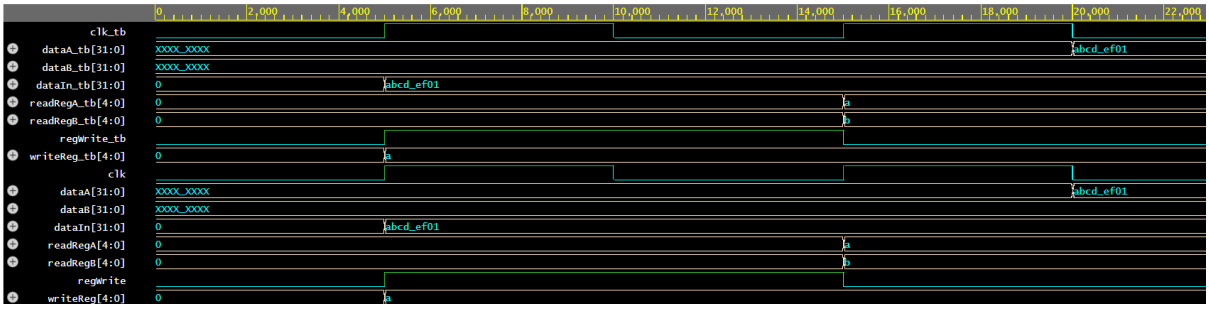
```
141     -- Test 7: many operations
142     readRegA_tb <= to_unsigned(2, 5);
143     readRegB_tb <= to_unsigned(10, 5);
144     wait for CLK_PERIOD;
145     assert dataA_tb = x"87654321" and dataB_tb = x"ABCDEF01" report "Test case 7 failed: Incorrect data read from register 20."
    severity failure;
146     regWrite_tb <= '1';
147     dataIn_tb <= x"11111111";
148     writeReg_tb <= to_unsigned(22, 5);
149     wait for CLK_PERIOD / 2;
150     readRegA_tb <= to_unsigned(22, 5);
151     wait for CLK_PERIOD / 2;
152     regWrite_tb <= '0';
153     wait for CLK_PERIOD;
154     assert dataA_tb = x"11111111" and dataA_tb /= x"87654321" report "Test case 7 failed: Incorrect data read from register 20."
    severity failure;
155
156     End simulation
157     wait;
158   end process;
159
160 end tb_arch;
```

Screenshots of the output:

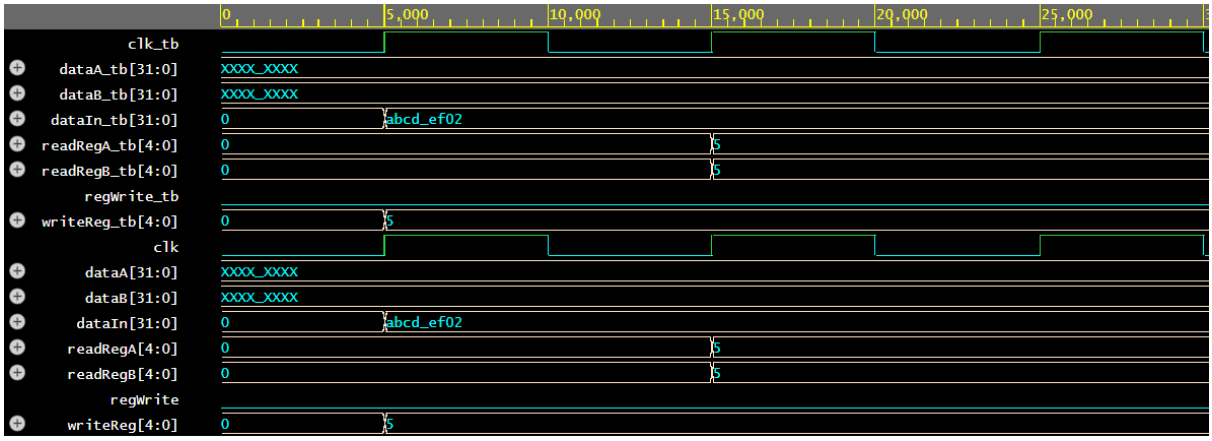
- Test 1:



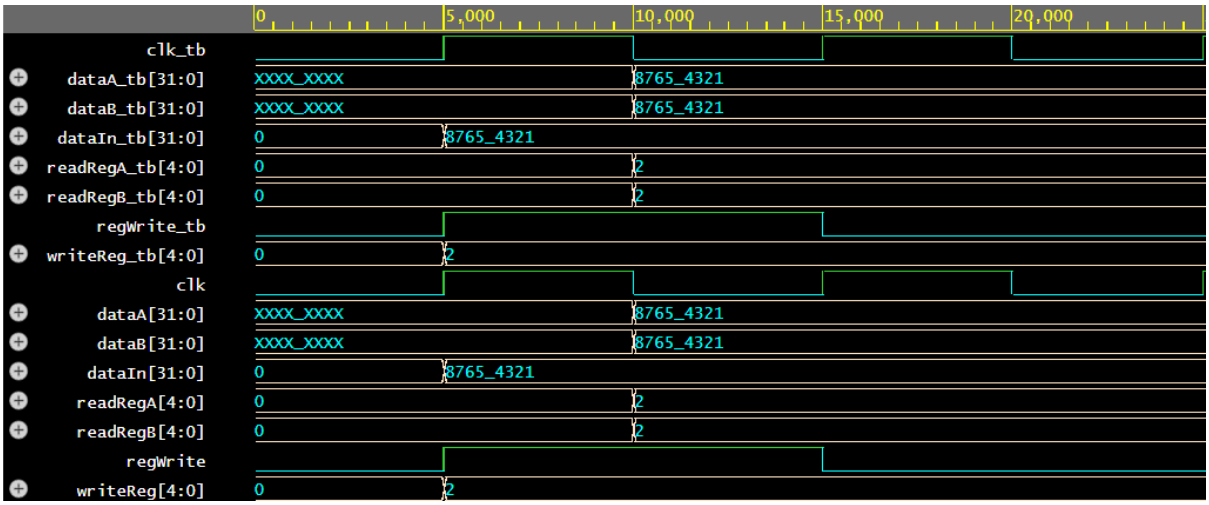
- Test 2:



- Test 3:

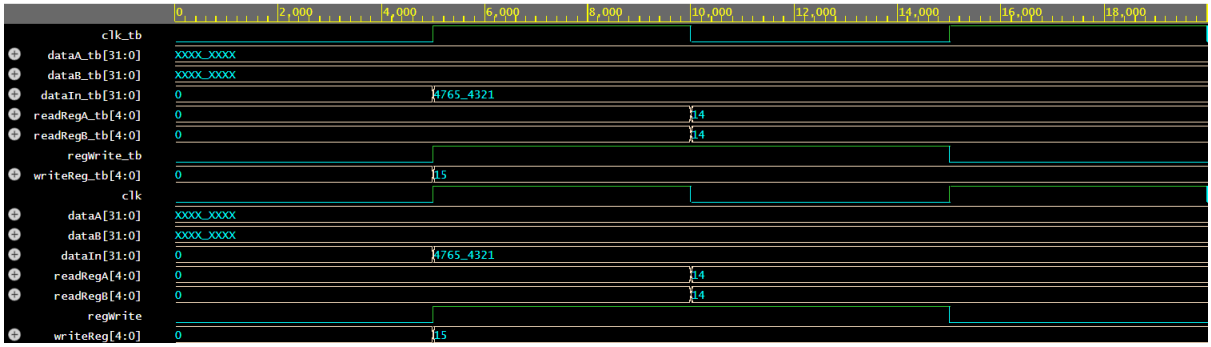


- Test 4:

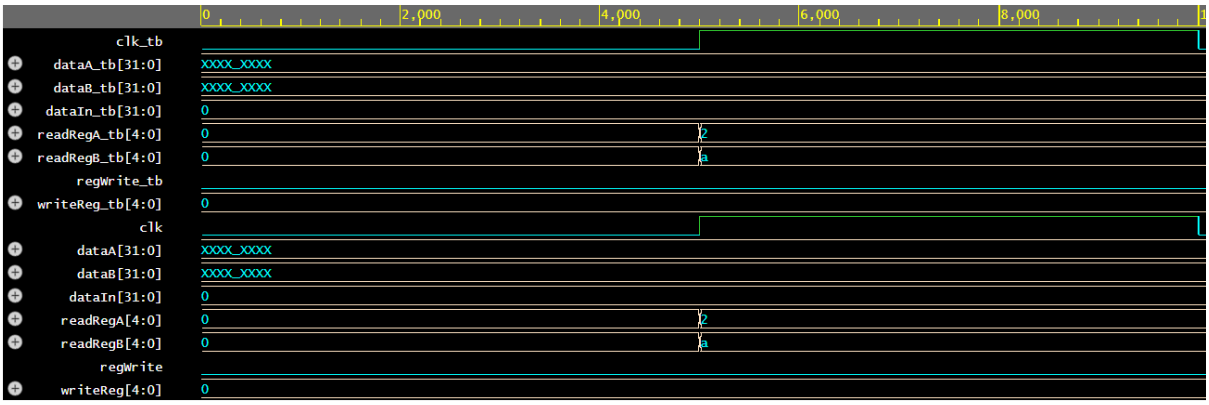


- Test 5:

- Test 6:



- Test 7:



- All tests: