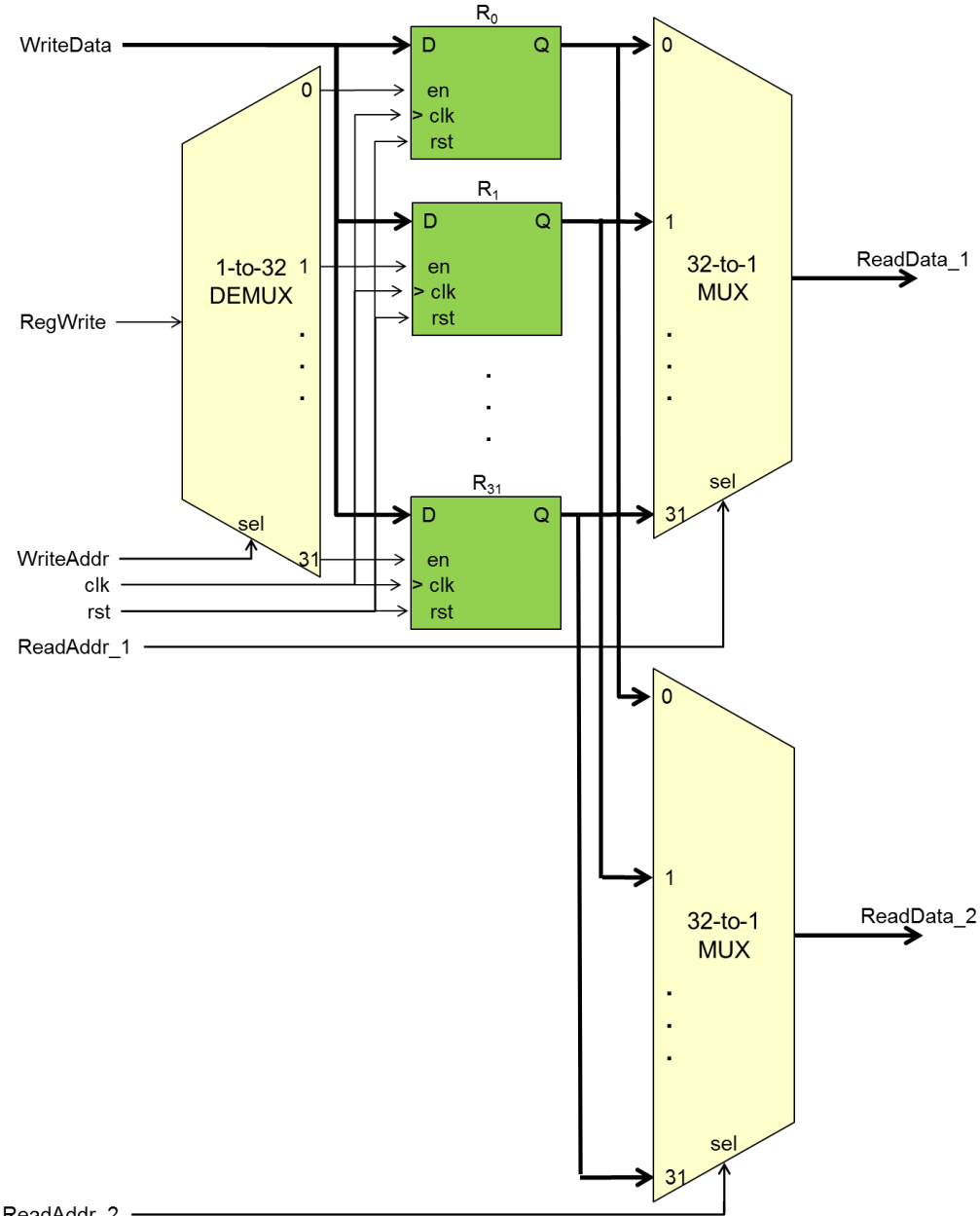
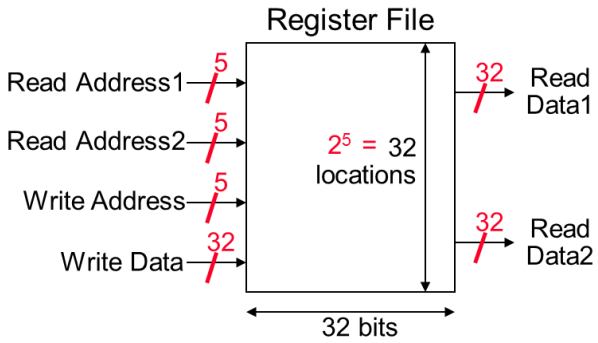
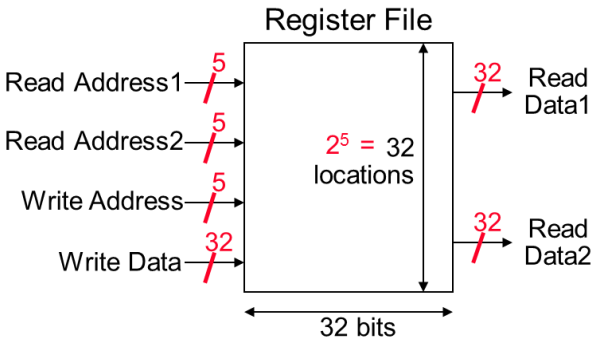
MIPS Register File

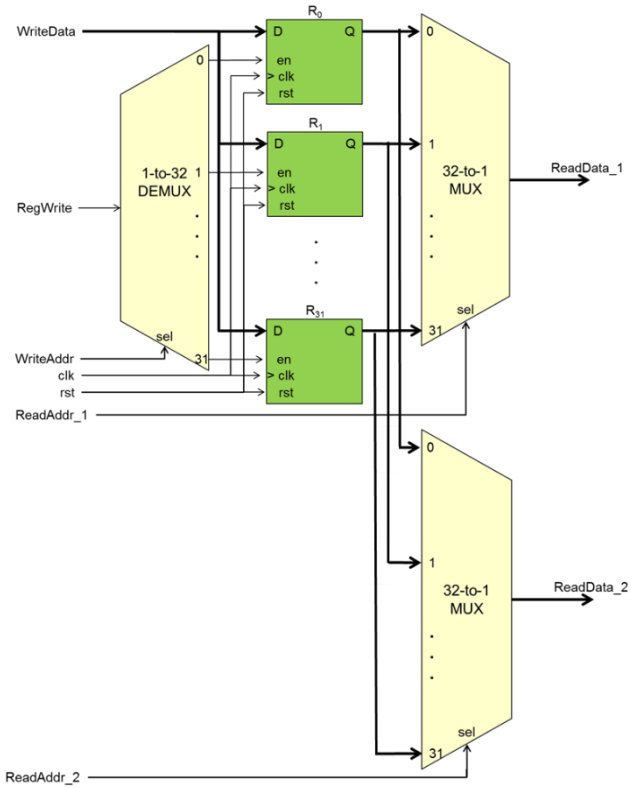




MIPS Register File

```
1  LIBRARY ieee;
2  USE ieee.std_logic_1164.all;
3  USE ieee.std_logic_arith.all;
4
5
6  ENTITY RegFile IS
7     PORT(
8        ReadAddr_1  : IN     std_logic_vector (4 DOWNTO 0);
9        ReadAddr_2  : IN     std_logic_vector (4 DOWNTO 0);
10       RegWrite    : IN     std_logic;
11       WriteAddr   : IN     std_logic_vector (4 DOWNTO 0);
12       WriteData   : IN     std_logic_vector (31 DOWNTO 0);
13       clk         : IN     std_logic;
14       rst         : IN     std_logic;
15       ReadData_1  : OUT    std_logic_vector (31 DOWNTO 0);
16       ReadData_2  : OUT    std_logic_vector (31 DOWNTO 0)
17    );
18 END RegFile ;
19
20
21 ARCHITECTURE struct OF RegFile IS
22
23    -- Architecture declarations
24    type reg_file_mem_type is array(0 to 31) of std_logic_vector(31 downto 0);
25
26    constant zero_register : std_logic_vector(31 downto 0) := (others => '0');
27    constant initial_reg_file : reg_file_mem_type := (others => zero_register);
28
29    -- Internal signal declarations
30    SIGNAL reg_file_mem : reg_file_mem_type;
31
32 BEGIN
33    ---------------------------------------------------------------------------
34    process1 : PROCESS (clk, rst)
35    ---------------------------------------------------------------------------
36    BEGIN
37       -- Asynchronous Reset
38       IF (rst = '1') THEN
39          -- Reset Actions
40          reg_file_mem <= initial_reg_file;
41
42       ELSIF (clk'EVENT AND clk = '1') THEN
43          IF RegWrite = '1' THEN
44             reg_file_mem(CONV_INTEGER(UNSIGNED(WriteAddr))) <= WriteData;
45          END IF;
46       END IF;
47    END PROCESS process1;
48
49    ReadData_1 <= reg_file_mem(CONV_INTEGER(UNSIGNED(ReadAddr_1)));
50    ReadData_2 <= reg_file_mem(CONV_INTEGER(UNSIGNED(ReadAddr_2)));
51
52 END struct;
```





MIPS Register File

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--USE ieee.numeric_std.ALL;

ENTITY regfile_tb IS
END regfile_tb;

ARCHITECTURE behavior OF regfile_tb IS

    -- Component Declaration for the Unit Under Test (UUT)

    COMPONENT RegFile
    PORT(
         ReadAddr_1 : IN  std_logic_vector(4 downto 0);
         ReadAddr_2 : IN  std_logic_vector(4 downto 0);
         RegWrite : IN  std_logic;
         WriteAddr : IN  std_logic_vector(4 downto 0);
         WriteData : IN  std_logic_vector(31 downto 0);
         clk : IN  std_logic;
         rst : IN  std_logic;
         ReadData_1 : OUT  std_logic_vector(31 downto 0);
         ReadData_2 : OUT  std_logic_vector(31 downto 0)
        );
    END COMPONENT;


   --Inputs
   signal ReadAddr_1 : std_logic_vector(4 downto 0) := (others => '0');
   signal ReadAddr_2 : std_logic_vector(4 downto 0) := (others => '0');
   signal RegWrite : std_logic := '0';
   signal WriteAddr : std_logic_vector(4 downto 0) := (others => '0');
   signal WriteData : std_logic_vector(31 downto 0) := (others => '0');
   signal clk : std_logic := '0';
   signal rst : std_logic := '0';

        --Outputs
   signal ReadData_1 : std_logic_vector(31 downto 0);
   signal ReadData_2 : std_logic_vector(31 downto 0);

   -- Clock period definitions
   constant clk_period : time := 10 ns;

BEGIN

        -- Instantiate the Unit Under Test (UUT)
   uut: RegFile PORT MAP (
          ReadAddr_1 => ReadAddr_1,
          ReadAddr_2 => ReadAddr_2,
          RegWrite => RegWrite,
          WriteAddr => WriteAddr,
          WriteData => WriteData,
          clk => clk,
          rst => rst,
          ReadData_1 => ReadData_1,
          ReadData_2 => ReadData_2
        );

```

```
    -- Clock process definitions
    clk_process :process
    begin
                clk <= '0';
                wait for clk_period/2;
                clk <= '1';
                wait for clk_period/2;
    end process;


    -- Stimulus process
    stim_proc: process
    begin
       -- hold reset state for 100 ns.
                rst <= '1';
            ReadAddr_1 <= "00000";
                ReadAddr_2 <= "00000";
                RegWrite <= '0';
            WriteAddr <= "00000";
            WriteData <= "00000000000000000000000000000000";
       wait for clk_period*10;

                -- insert stimulus here
                rst <= '0';
                RegWrite <= '1';
            WriteAddr <= "00101";
            WriteData <= "00000000000000000000000000001111";
       wait for clk_period*3;

                RegWrite <= '1';
            WriteAddr <= "01100";
            WriteData <= "00000000000000000000000000011111";
       wait for clk_period*3;

            ReadAddr_1 <= "00101";
                ReadAddr_2 <= "01100";
                RegWrite <= '0';
       wait for clk_period*3;

                rst <= '1';
       wait for clk_period*3;

       wait;
    end process;

END;
```

MIPS Register File