Lab 4:

-Overview-

The purpose of this lab is to construct a generic register and a generic decoder, then use them to build a register file. Your VHDL code will be verified through simulation first. Synthesis and implementation will be performed as well.

-Design-

The register wasn't too complicated. Actions only happen on every clock cycle (rising edge). Reset has highest priorty, setting all bit to 0 regardless of enable. After that, q is set to d if enable is high, else it keeps its previous state.

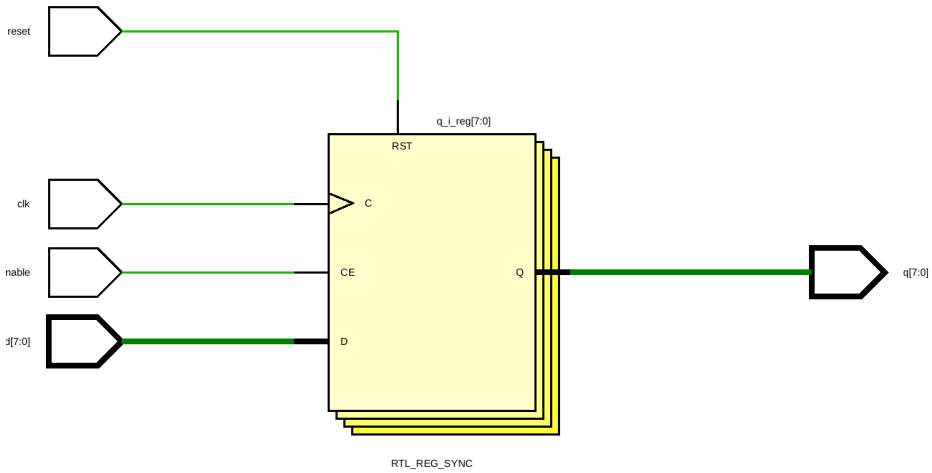


Figure 1: Shematic of Register

For the register file, there is both read and write functionality. With the write, a decoder is used to select the appropriate write enable on from the write address inputted into the decoder. The decoders enable is used a write enable, and doesn't select any registers when write_en is low. For reading, each output A and B are identical. A mux inputs all the registers, and takes outputs only the address given to it, reading the register.

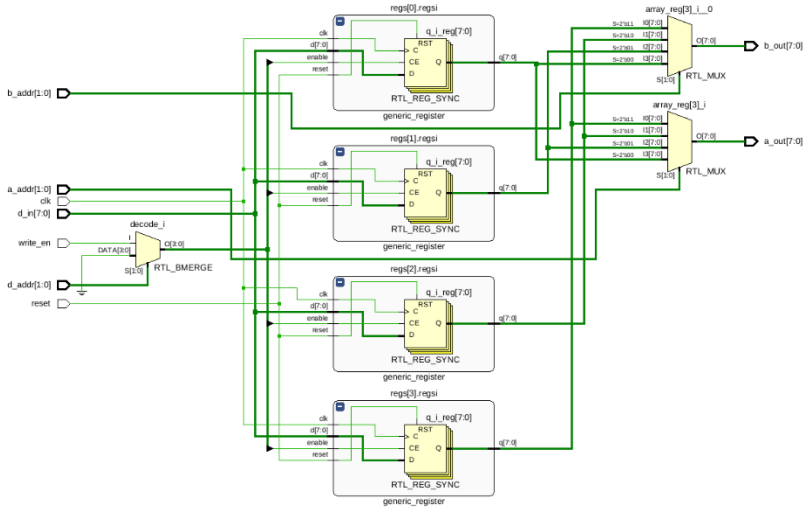


Figure 2: 4 bit register file with 2 outputs

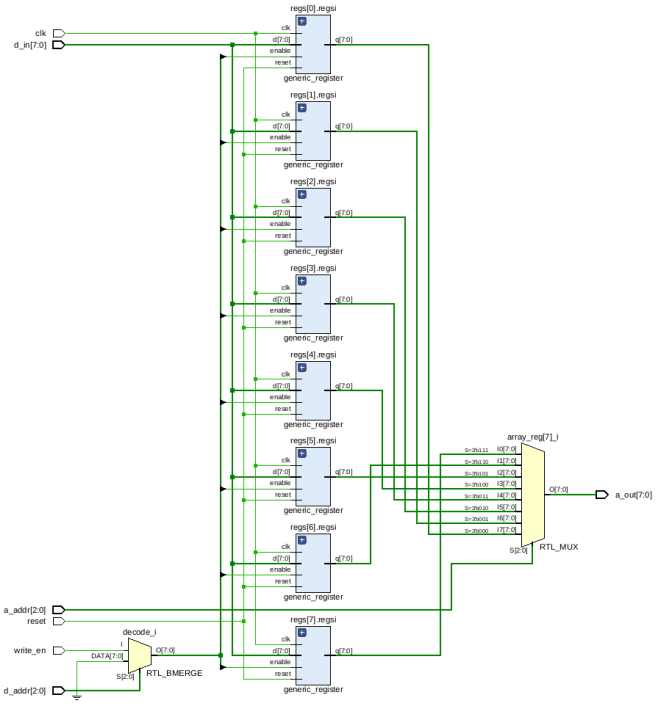
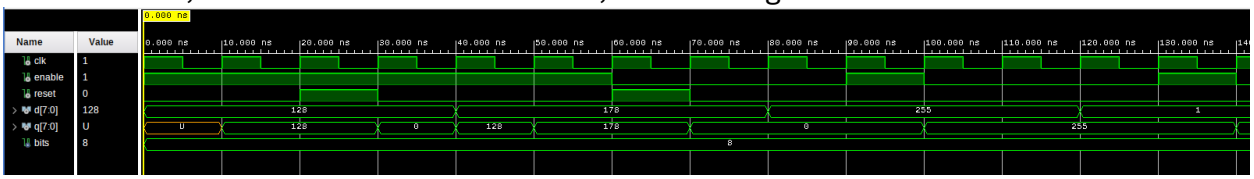


Figure 3: Schematic of 8 bit register file used for implementation

-Simulation –

-Register-

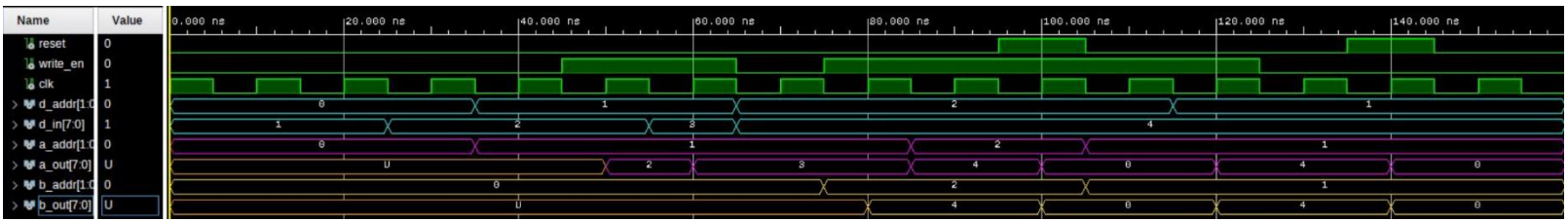
1. At 30 ns, reset is high, so the register clears regardless of enable.
2. At 40 ns, reset is back to low, and enable is high so the register is written to.
3. At 50 ns, a new values is given and enable is high, so that new value is written.
4. At 70 ns, reset is high and enable is low, and register clears.
5. At 90 ns, the value is changed, but doesn't write because enable is low.
6. At 100 ns, enable is high and the register is written to.
7. At 110, enable is low and reset is low, so no change is the value.



-Register File-

(blue is input, magenta is output a, and gold is ouput b)

1. Before 50 ns, any change is d_addr, d_in, or a_addr, doesn't change the registers because write_en is never high.
2. At 50 ns, register 1 is written to and output 'a' outputs that change.
3. At 60 ns, the value of register 1 is changed, and 'a' outputs that change.
4. At 70 ns, nothing changes with the output 'a' or 'b' as only the input parameters changed.
5. At 80 ns, 4 is written to register 2, and output 'b' shows that change.
6. At 85 ns, output 'a' changes address, and the output is updated asynchronously
7. At 90 ns, output 'a' and output 'b' are reading from the same register
8. At 100 ns, reset is high and all the registers are set to 0
9. At 130 ns, a the value of a register stays even without write enable high.
10. At 140 ns, the value is reset to '0'



-Implemented Design Verification-

-Register-

(Switches 3-0 are input, LED 3-0 are ouput, buttons are enable and reset)

Output after enable was pushed

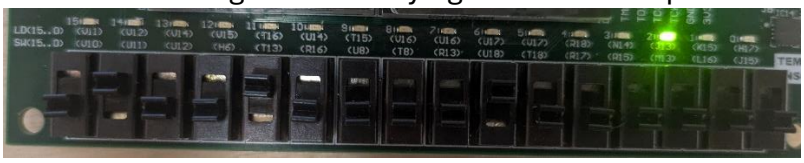
Output input changed but enable not pushed

Output after enable was pushed again

Output after reset was pushed

-Register File-
(switch15-13 is a_addr, 12-10 is d_addr, 0-7 is d_in, LED 0-7 is a_out)
(enable and reset are buttons)

Old Value of register 2 is staying before enable pressed



Register 2 updating after enable is pressed

Register 0 still has old value after chnging register 2

Registers before reset

Registers cleared to 0's after reset is pressed

-Conclusion-

In conclusion, the lab went smoothly and was pretty straightforward overall. The main challenge I faced was with some formatting issues and getting the register to work properly without introducing extra multiplexers. Other than that, the design, simulation, and implementation steps were completed successfully, and everything worked as expected after some minor adjustments.

-Appendix-
-Register-
Main VHDL

```
1   library IEEE;
2   use IEEE.STD_LOGIC_1164.ALL;
3
4   entity generic_register is
5       generic (bits: integer := 8);
6       port(
7           clk, reset,enable: in std_logic;
8           d: in std_logic_vector ( bits-1 downto 0 );
9           q: out std_logic_vector ( bits-1 downto 0 )
10          );
11  end generic_register;
12
13  architecture Behavioral of generic_register is
14      signal q_i, next_q_i : std_logic_vector ( bits-1 downto 0 );
15      signal ctrl : std_logic_vector ( 1 downto 0 );
16  begin
17      q_i <= next_q_i when rising_edge(clk);
18
19      ctrl <= reset & enable;
20      with ctrl select next_q_i <=
21          d when "01",
22          q_i when "00",
23          (others => '0') when others;
24      q <= q_i;
25  end Behavioral;
26
```

Testbench VHDL

```
1
2   library IEEE;
3   use IEEE.STD_LOGIC_1164.ALL;
4
5
6   entity generic_register_tb is
7   end generic_register_tb;
8
9   architecture Behavioral of generic_register_tb is
10      constant bits : natural := 8;
11      signal clk, enable : std_logic := '1';
12      signal reset : std_logic := '0';
13      signal d, q : std_logic_vector ( bits-1 downto 0 ) := "10000000";
14
15  begin
16
17  uut : entity work.generic_register ( Behavioral )
18      generic map ( bits => bits )
19      port map (
20          clk => clk,
21          reset => reset,
22          enable => enable,
23          d => d,
24          q => q);
25
26  clk <= not clk after 5 ns;
27  reset <= '1' after 20 ns, '0' after 30 ns, '1' after 60 ns, '0' after 70 ns;
28  enable <= '1' after 50ns, '0' after 60ns, '1' after 90 ns, '0' after 100 ns, '1' after 130 ns, '0' after 140 ns;
29  d <= "10110010" after 40 ns, "11111111" after 80 ns, "00000001" after 120 ns;
30
31
32  end Behavioral;
33
```

Constraint File (buttons were enable and reset)

```
6   ## Clock signal
7   set_property -dict {PACKAGE_PIN E3 IOSTANDARD LVCMOS33} [get_ports {clk}]
8   #create_clock -period 10.000 -name sys_clk_pin -waveform {0.000 5.000} -add [get_ports {clk}33]
9
10
11  ##Switches
12  set_property -dict {PACKAGE_PIN J15 IOSTANDARD LVCMOS33} [get_ports {d[0]}]
13  set_property -dict {PACKAGE_PIN L16 IOSTANDARD LVCMOS33} [get_ports {d[1]}]
14  set_property -dict {PACKAGE_PIN M13 IOSTANDARD LVCMOS33} [get_ports {d[2]}]
15  set_property -dict {PACKAGE_PIN R15 IOSTANDARD LVCMOS33} [get_ports {d[3]}]
16  #set_property -dict {PACKAGE_PIN R17 IOSTANDARD LVCMOS33} [get_ports {d_in[4]}]
17  #set_property -dict {PACKAGE_PIN T18 IOSTANDARD LVCMOS33} [get_ports {d_in[5]}]
18  #set_property -dict {PACKAGE_PIN U18 IOSTANDARD LVCMOS33} [get_ports {d_in[6]}]
19  #set_property -dict {PACKAGE_PIN R13 IOSTANDARD LVCMOS33} [get_ports {d_in[7]}]
20  #set_property -dict {PACKAGE_PIN T8 IOSTANDARD LVCMOS18} [get_ports {}]
21  #set_property -dict {PACKAGE_PIN U8 IOSTANDARD LVCMOS18} [get_ports {}]
22  #set_property -dict {PACKAGE_PIN R16 IOSTANDARD LVCMOS33} [get_ports {d_addr[0]}]
23  #set_property -dict {PACKAGE_PIN T13 IOSTANDARD LVCMOS33} [get_ports {d_addr[1]}]
24  #set_property -dict {PACKAGE_PIN H6 IOSTANDARD LVCMOS33} [get_ports {d_addr[2]}]
25  #set_property -dict {PACKAGE_PIN U12 IOSTANDARD LVCMOS33} [get_ports {a_addr[0]}]
26  #set_property -dict {PACKAGE_PIN U11 IOSTANDARD LVCMOS33} [get_ports {a_addr[1]}]
27  #set_property -dict {PACKAGE_PIN V10 IOSTANDARD LVCMOS33} [get_ports {a_addr[2]}]
28
29  ## LEDs
30  set_property -dict {PACKAGE_PIN H17 IOSTANDARD LVCMOS33} [get_ports {q[0]}]
31  set_property -dict {PACKAGE_PIN K15 IOSTANDARD LVCMOS33} [get_ports {q[1]}]
32  set_property -dict {PACKAGE_PIN J13 IOSTANDARD LVCMOS33} [get_ports {q[2]}]
33  set_property -dict {PACKAGE_PIN N14 IOSTANDARD LVCMOS33} [get_ports {q[3]}]
```

-Register File-

Main VHDL

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity generic_register is
    generic (bits: integer := 8);
    port(
        clk, reset,enable: in std_logic;
        d: in std_logic_vector ( bits-1 downto 0 );
        q: out std_logic_vector ( bits-1 downto 0 )
        );
end generic_register;

architecture Behavioral of generic_register is
    signal q_i, next_q_i : std_logic_vector ( bits-1 downto 0 );
    signal ctrl : std_logic_vector ( 1 downto 0 );
begin
    q_i <= next_q_i when rising_edge(clk);
    next_q_i <= (others => '0' ) when reset ='1' else d when enable = '1' else q_i;
    q <= q_i;
end Behavioral;
```

Testbench VHDL

```vhdl
    architecture Behavioral of generic_register_file_tb is
        constant word_len : natural := 8;
        constant addr_bits : natural := 2;
        signal reset, write_en: std_logic := '0';
        signal clk : std_logic := '1';
        signal d_addr: std_logic_vector ( addr_bits-1 downto 0 ) := "00";
        signal d_in: std_logic_vector ( word_len-1 downto 0 ) := "00000001";
        signal a_addr: std_logic_vector ( addr_bits-1 downto 0 ) := "00";
        signal a_out : std_logic_vector ( word_len-1 downto 0 ) := "00000000";
        signal b_addr: std_logic_vector ( addr_bits-1 downto 0 ):= "00";
        signal b_out : std_logic_vector ( word_len-1 downto 0 ) := "00000000";

    begin

    uut : entity work.generic_register_file ( Behavioral )
        generic map (
            word_len => word_len,
            addr_bits => addr_bits )
        port map (
            clk => clk,
            reset => reset,
            write_en => write_en,
            d_addr => d_addr,
            d_in => d_in,
            a_addr => a_addr,
            a_out => a_out,
            b_addr => b_addr,
            b_out => b_out
            );

    clk <= not clk after 5 ns;
    d_in <= "00000010" after 25 ns, "00000011" after 55 ns, "00000100" after 65 ns;
    d_addr <= "01" after 35 ns, "10" after 65 ns, "01" after 115 ns;
    a_addr <= "01" after 35 ns, "10" after 85 ns, "01" after 105 ns;
    write_en <= '1' after 45 ns, '0' after 65 ns, '1' after 75 ns, '0' after 125 ns;
    b_addr <= "10" after 75 ns, "01" after 105 ns;
    reset <= '1' after 95 ns, '0' after 105 ns, '1' after 135 ns, '0' after 145 ns;

    end Behavioral;
```

Constraint File (buttons were write enable and reset)

```
## Clock signal
set_property -dict {PACKAGE_PIN E3 IOSTANDARD LVCMOS33} [get_ports {clk}]
#create_clock -period 10.000 -name sys_clk_pin -waveform {0.000 5.000} -add [get_ports {clk}33]


##Switches
set_property -dict {PACKAGE_PIN J15 IOSTANDARD LVCMOS33} [get_ports {d_in[0]}]
set_property -dict {PACKAGE_PIN L16 IOSTANDARD LVCMOS33} [get_ports {d_in[1]}]
set_property -dict {PACKAGE_PIN M13 IOSTANDARD LVCMOS33} [get_ports {d_in[2]}]
set_property -dict {PACKAGE_PIN R15 IOSTANDARD LVCMOS33} [get_ports {d_in[3]}]
set_property -dict {PACKAGE_PIN R17 IOSTANDARD LVCMOS33} [get_ports {d_in[4]}]
set_property -dict {PACKAGE_PIN T18 IOSTANDARD LVCMOS33} [get_ports {d_in[5]}]
set_property -dict {PACKAGE_PIN U18 IOSTANDARD LVCMOS33} [get_ports {d_in[6]}]
set_property -dict {PACKAGE_PIN R13 IOSTANDARD LVCMOS33} [get_ports {d_in[7]}]
#set_property -dict {PACKAGE_PIN T8 IOSTANDARD LVCMOS18} [get_ports {}]
#set_property -dict {PACKAGE_PIN U8 IOSTANDARD LVCMOS18} [get_ports {}]
set_property -dict {PACKAGE_PIN R16 IOSTANDARD LVCMOS33} [get_ports {d_addr[0]}]
set_property -dict {PACKAGE_PIN T13 IOSTANDARD LVCMOS33} [get_ports {d_addr[1]}]
set_property -dict {PACKAGE_PIN H6 IOSTANDARD LVCMOS33} [get_ports {d_addr[2]}]
set_property -dict {PACKAGE_PIN U12 IOSTANDARD LVCMOS33} [get_ports {a_addr[0]}]
set_property -dict {PACKAGE_PIN U11 IOSTANDARD LVCMOS33} [get_ports {a_addr[1]}]
set_property -dict {PACKAGE_PIN V10 IOSTANDARD LVCMOS33} [get_ports {a_addr[2]}]

## LEDs
set_property -dict {PACKAGE_PIN H17 IOSTANDARD LVCMOS33} [get_ports {a_out[0]}]
set_property -dict {PACKAGE_PIN K15 IOSTANDARD LVCMOS33} [get_ports {a_out[1]}]
set_property -dict {PACKAGE_PIN J13 IOSTANDARD LVCMOS33} [get_ports {a_out[2]}]
set_property -dict {PACKAGE_PIN N14 IOSTANDARD LVCMOS33} [get_ports {a_out[3]}]
set_property -dict {PACKAGE_PIN R18 IOSTANDARD LVCMOS33} [get_ports {a_out[4]}]
set_property -dict {PACKAGE_PIN V17 IOSTANDARD LVCMOS33} [get_ports {a_out[5]}]
set_property -dict {PACKAGE_PIN U17 IOSTANDARD LVCMOS33} [get_ports {a_out[6]}]
set_property -dict {PACKAGE_PIN U16 IOSTANDARD LVCMOS33} [get_ports {a_out[7]}]
#set_property -dict {PACKAGE_PIN V16 IOSTANDARD LVCMOS33} [get_ports {leds[8]}]
#set_property -dict {PACKAGE_PIN T15 IOSTANDARD LVCMOS33} [get_ports {leds[9]}]
```