Spring 2019

California State University, Northridge

Department of Electrical and Computer Engineering

Computer Assignment 5: Block memory copy

May 08, 2019

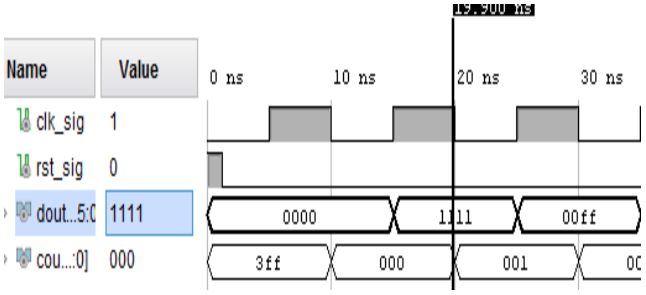
Professor: Shahnam Mirzaei Ph.D

Authors:
Ridge Tejuco

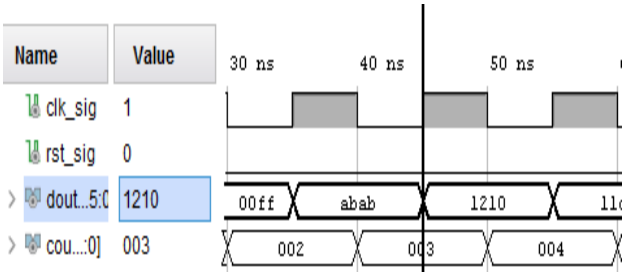
When reading from the RAM, there is a delay that offsets the results. A initial value of "0000" can be read which is not part of the initialization. The counter displayed has been adjusted to read the true values of the address.
After reading M(1) and M(0), they are added together and stored in M(3)

1111
+11FF
1210

Read

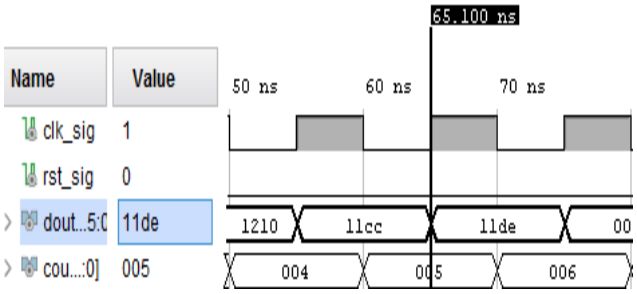


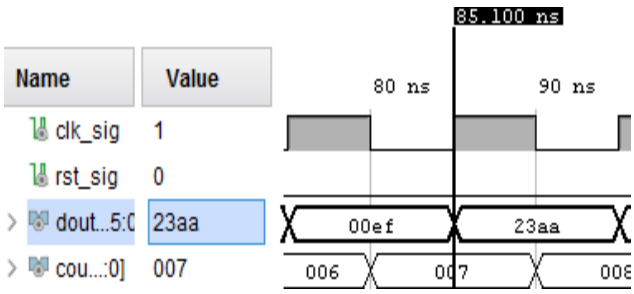
Write



For ease of reading, the counter at the bottom, indicates the current address when reading from or writing to the memory.
Next, after reading M(4) and M(5), the value is store in M(7)

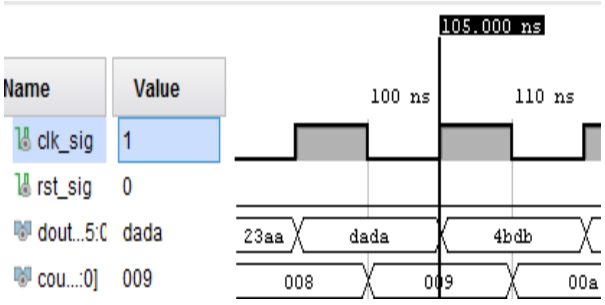
11cc
+ 11de
23aa

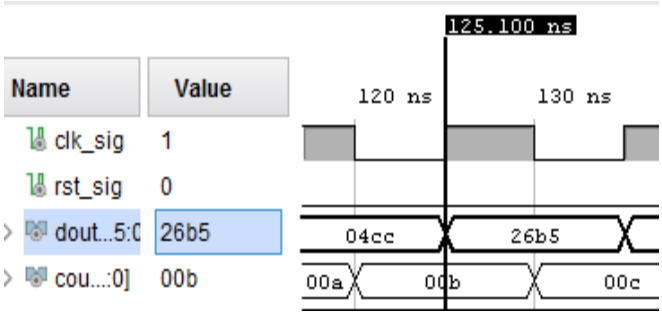




In the last check, M(8) and M(9) were read, and stored in M(11)

dada
+ 4bdb
126b5





The first design with a single port ram correctly copies the added values.

For the dual port Ram, both M(0) and M(1) are read on the same clock cycle.

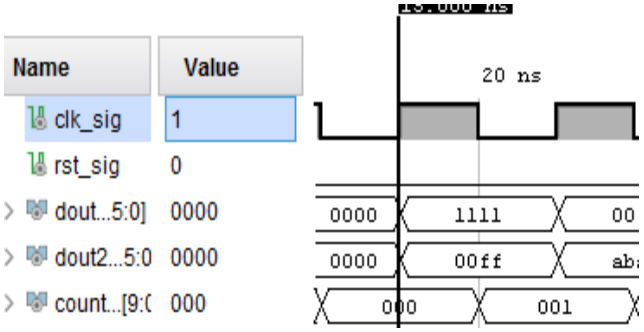
M(3) = M(1) + M(2)

1111

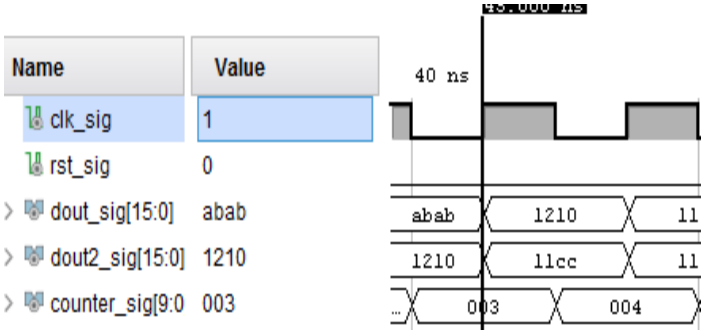
+ 00FF

1210

Read



Write

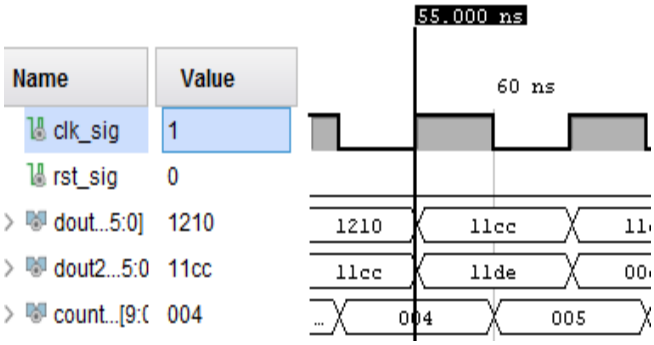


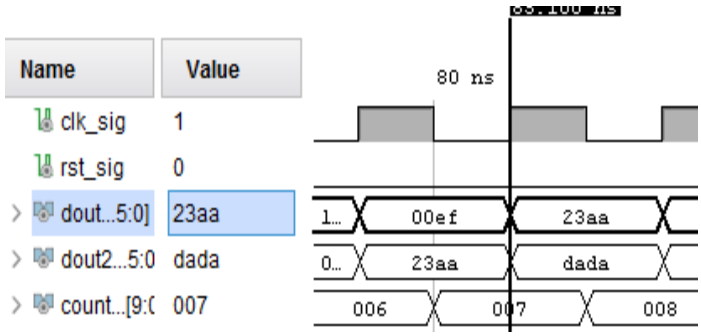
M(7) = M(4) + M(5)

11cc

+ 11de

23aa



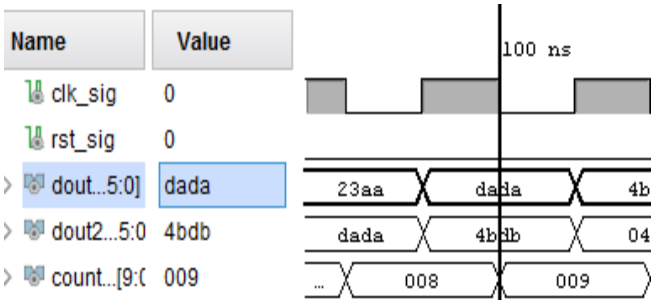


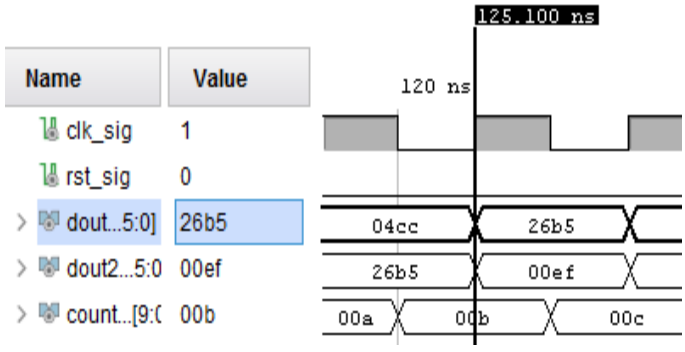
M(11) = M(8) + M(9)

dada

+ 4bdb

126b5





```
--memBlockCopy_top.vhd
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.numeric_std.all;
entity memBlockCopy_top is
      Port(clk,rst: in std_logic;
      counter_out: out std_logic_vector(9 downto 0);
      dout:out std_logic_vector(15 downto 0));
end memBlockCopy_top;
architecture Behavioral of memBlockCopy_top is
      signal high: std_logic_vector(0 downto 0) := "1";
      signal low: std_logic_vector(0 downto 0) := "0";
      signal we: std_logic_vector(0 downto 0);
      signal address,counter_in: std_logic_vector(9 downto 0);
      signal u1: unsigned(9 downto 0) := "0000000001";
      signal data_in,data_out,dff,ADD: std_logic_vector(15 downto 0);
      component blk_mem_gen_0 is
      Port(
            clka : IN STD_LOGIC;
            wea : IN STD_LOGIC_VECTOR(0 DOWNTO 0);
            addra : IN STD_LOGIC_VECTOR(9 DOWNTO 0);
            dina : IN STD_LOGIC_VECTOR(15 DOWNTO 0);
            douta : OUT STD_LOGIC_VECTOR(15 DOWNTO 0));
      end component;
begin
MEMORY: blk_mem_gen_0
      port map(clka => clk, wea => we,
      addra => address, dina => data_in, douta => data_out);
process(clk,rst) -- DFF
begin
      if rising_edge(clk) then
      dff <= data_out;
      end if;
end process;
ADD <= std_logic_vector(unsigned(dff) + unsigned(data_out));
process(clk,rst)
begin
      if(rst = '1') then
      address <= (others => '0');
      elsif falling_edge(clk) then
      address <= counter_in;
      end if;
end process;
counter_in <= std_logic_vector(unsigned(address) + u1);
process(address)
begin
      if(address(1 downto 0) = "11") then
      we <= high;
```

```
      else
      we <= low;
      end if;
end process;
data_in <= ADD;
dout <= data_out;
counter_out <= std_logic_vector(unsigned(address) - u1);
end Behavioral;
```

-- memBlockCopy_tb.vhd

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity memBlockCopy_tb is
--  Port ( );
end memBlockCopy_tb;
architecture Behavioral of memBlockCopy_tb is
signal clk_sig,rst_sig: std_logic := '0';
signal dout_sig: std_logic_vector(15 downto 0);
signal counter_sig: std_logic_vector(9 downto 0);
constant cp: time := 10ns;
component memBlockCopy_top is
      Port(clk,rst: in std_logic;
      counter_out: out std_logic_vector(9 downto 0);
      dout:out std_logic_vector(15 downto 0));
end component;
begin
UUT: memBlockCopy_top
      port map(clk => clk_sig, rst => rst_sig,counter_out => counter_sig, dout =>
dout_sig);
process(clk_sig)
begin
      clk_sig <= not clk_sig after cp/2;
end process;
process
begin
      rst_sig <= '1';
      wait for cp/8;
      rst_sig <= '0';
      wait;
end process;
end Behavioral;
```

```
--dpRamCopy_top.vhd
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.numeric_std.all;
entity dpRamCopy_top is
      Port(clk,rst: in std_logic;
      counter_out: out std_logic_vector(9 downto 0);
      doutA,doutB: out std_logic_vector(15 downto 0));
end dpRamCopy_top;
architecture Behavioral of dpRamCopy_top is
      signal high: std_logic_vector(0 downto 0) := "1";
      signal low: std_logic_vector(0 downto 0) := "0";
      signal low16: std_logic_vector(15 downto 0) := (others => '0');
      signal we: std_logic_vector(0 downto 0);
      signal u1: unsigned(9 downto 0) := "0000000001";
      signal address,addressB: std_logic_vector(9 downto 0);
      signal data_in,data_out,data_out2,ADD,dff,dff2: std_logic_vector(15 downto 0);
      component blk_mem_gen_1 IS
      PORT (
      clka : IN STD_LOGIC;
      wea : IN STD_LOGIC_VECTOR(0 DOWNTO 0);
      addra : IN STD_LOGIC_VECTOR(9 DOWNTO 0);
      dina : IN STD_LOGIC_VECTOR(15 DOWNTO 0);
      douta : OUT STD_LOGIC_VECTOR(15 DOWNTO 0);
      clkb : IN STD_LOGIC;
      web : IN STD_LOGIC_VECTOR(0 DOWNTO 0);
      addrb : IN STD_LOGIC_VECTOR(9 DOWNTO 0);
      dinb : IN STD_LOGIC_VECTOR(15 DOWNTO 0);
      doutb : OUT STD_LOGIC_VECTOR(15 DOWNTO 0)
      );
      END component;
begin
MEMORY: blk_mem_gen_1
      port map(clka => clk,wea => low,dina => low16, addra => address,
            douta => data_out,clkb => clk, web => we,addrb => addressB,
            dinb => data_in, doutb => data_out2);
process(clk,rst)
begin
      if(rst = '1') then
      address <= (others => '0');
      elsif falling_edge(clk) then
      address <= addressB;
      end if;
end process;
process(address)
begin
      if(address(1 downto 0) = "10") then
      we <= high;
```

```
      else
      we <= low;
      end if;
end process;
process(clk)
begin
end process;
addressB <= std_logic_vector(unsigned(address) + u1);
ADD <= std_logic_vector(unsigned(data_out) + unsigned(data_out2));
data_in <= ADD;
doutA <= data_out;
doutB <= data_out2;
counter_out <= std_logic_vector(unsigned(address) - u1);
end Behavioral;
--dpRamCopy_tb.vhd;
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity dpRamCopy_tb is
--  Port ( );
end dpRamCopy_tb;
architecture Behavioral of dpRamCopy_tb is
signal clk_sig,rst_sig: std_logic := '0';
signal dout_sig,dout2_sig: std_logic_vector(15 downto 0);
signal counter_sig: std_logic_vector(9 downto 0);
constant cp: time := 10ns;
component dpRamCopy_top is
      Port(clk,rst: in std_logic;
      counter_out: out std_logic_vector(9 downto 0);
      doutA,doutB: out std_logic_vector(15 downto 0));
end component dpRamCopy_top;
begin
process(clk_sig)
begin
      clk_sig <= not clk_sig after cp/2;
end process;
UUT: dpRamCopy_top
      port map(clk => clk_sig, rst => rst_sig,
            counter_out => counter_sig, doutA => dout_sig,
            doutB => dout2_sig);
process
begin
      rst_sig <= '1';
      wait for cp/8;
      rst_sig <= '0';
      wait;
end process;
end Behavioral;
```