1928

K. N. Toosi University of Technology

1402-1403

Ramin Tavakoli

FPGA

Student Number: 9925063

Dr. Hosseini Nezhad

Homework 3

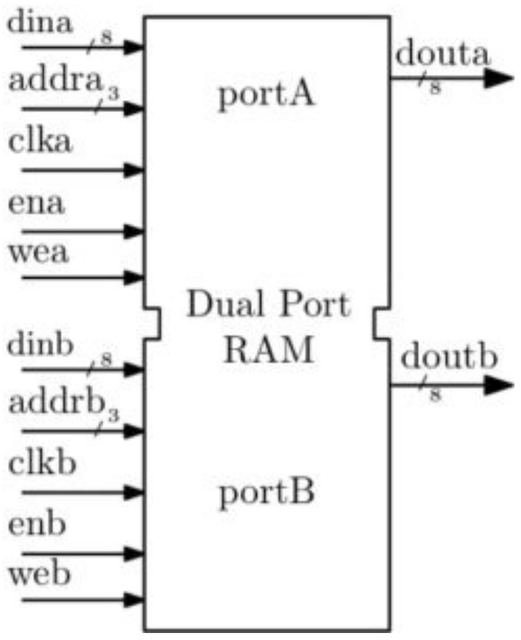
Department of electrical engineering

*K.N Toosi University of Technology
December 12, 2023

سوال ۱) یک حافظه ۲۵۶ بایت Dual Port RAM طراحی نمایید.

Dual port memory elements:

The recent technology has developed dual port memories. Now it is possible to access the same address locations through two ports. Dual port memories have simplified many problems in designing digital systems. Both ROM and RAM can be of dual port. The block diagram of a true dual port RAM is shown below.



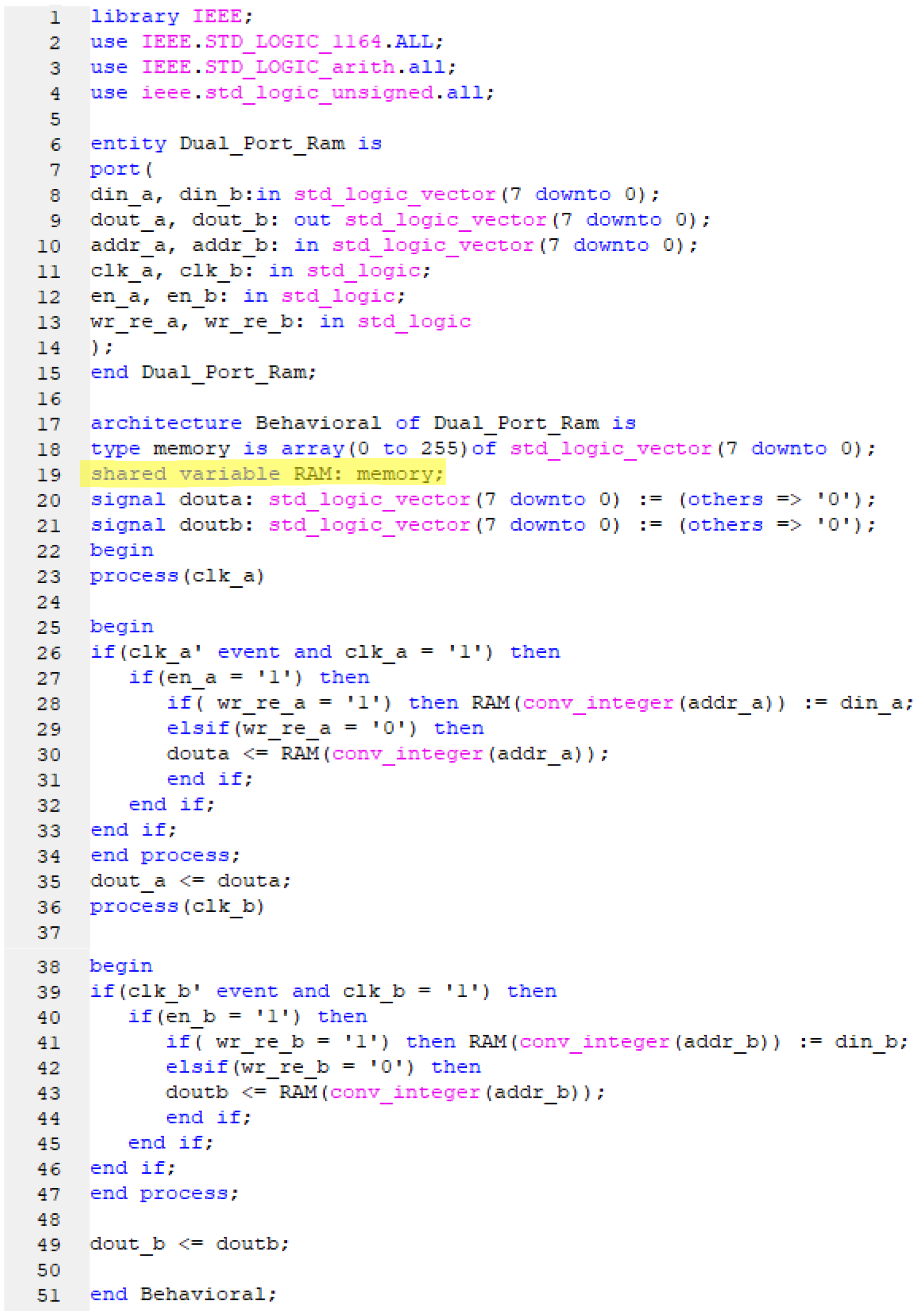
The dual port memories have separate control line for both the ports. The various modes of a typical true dual port RAM are shown below. In the mode 1, writing of data is possible through both the ports but not on the same location. In mode 3, data can be read through both the ports even from the same address location. In mode 2 and 3, one port is busy in writing while another port is reading data.

Table 1: Modes of Dual Port RAM

modes	ena	wea	enb	web	portA	portB
1	1	1	1	1	write	write
2	1	1	1	0	write	read
3	1	0	1	1	read	write
4	1	0	1	0	read	read

Circuit synthesis code:

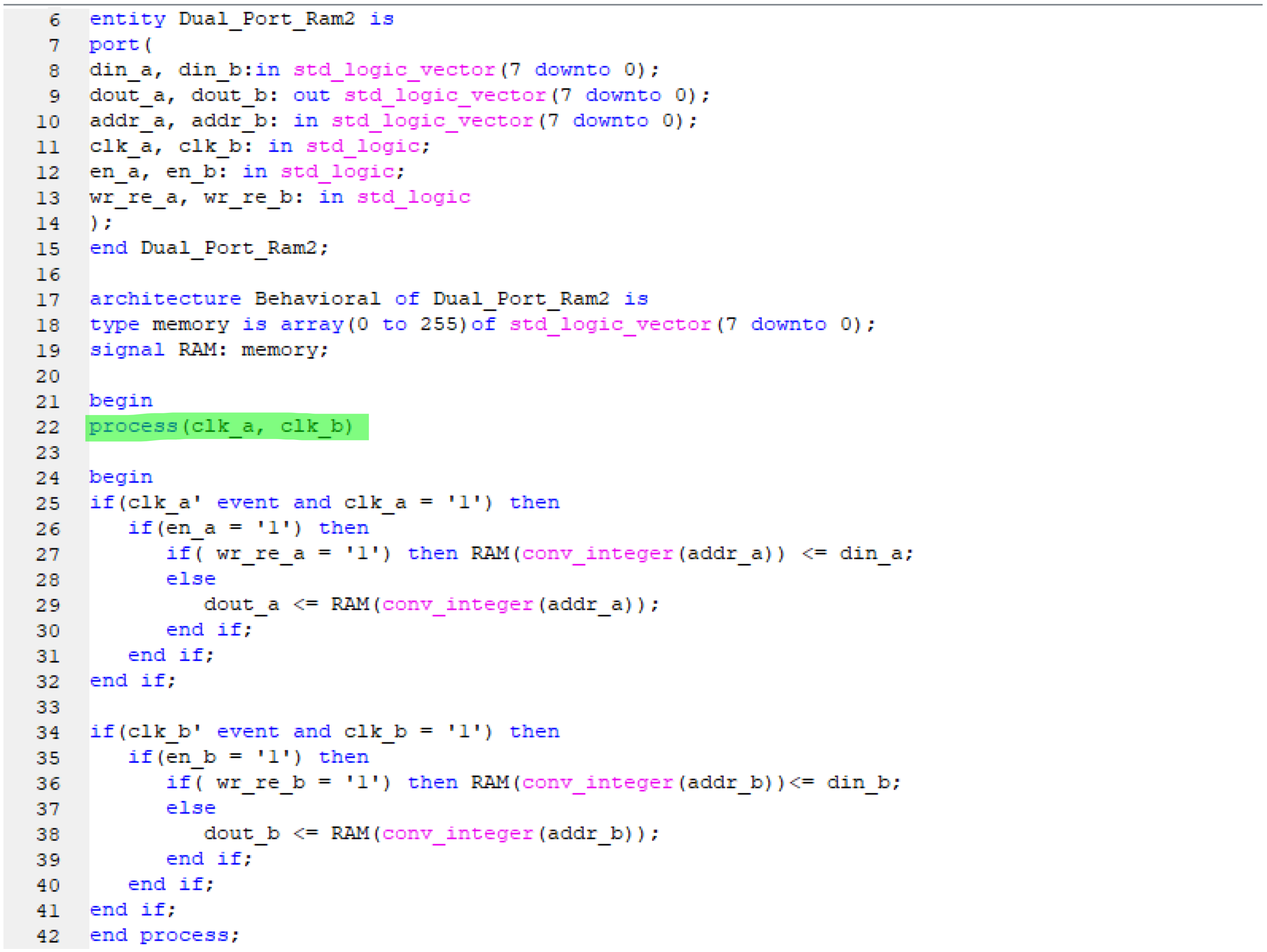
In the first method, we designed the memory by two processes.



```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_arith.all;
use ieee.std_logic_unsigned.all;

entity Dual_Port_Ram is
port(
din_a, din_b:in std_logic_vector(7 downto 0);
dout_a, dout_b: out std_logic_vector(7 downto 0);
addr_a, addr_b: in std_logic_vector(7 downto 0);
clk_a, clk_b: in std_logic;
en_a, en_b: in std_logic;
wr_re_a, wr_re_b: in std_logic
);
end Dual_Port_Ram;

architecture Behavioral of Dual_Port_Ram is
type memory is array(0 to 255)of std_logic_vector(7 downto 0);
shared variable RAM: memory;
signal douta: std_logic_vector(7 downto 0) := (others => '0');
signal doutb: std_logic_vector(7 downto 0) := (others => '0');
begin
process(clk_a)

begin
if(clk_a' event and clk_a = '1') then
   if(en_a = '1') then
      if( wr_re_a = '1') then RAM(conv_integer(addr_a)) := din_a;
      elsif(wr_re_a = '0') then
      douta <= RAM(conv_integer(addr_a));
      end if;
   end if;
end if;
end process;
dout_a <= douta;
process(clk_b)

begin
if(clk_b' event and clk_b = '1') then
   if(en_b = '1') then
      if( wr_re_b = '1') then RAM(conv_integer(addr_b)) := din_b;
      elsif(wr_re_b = '0') then
      doutb <= RAM(conv_integer(addr_b));
      end if;
   end if;
end if;
end process;

dout_b <= doutb;

end Behavioral;
```

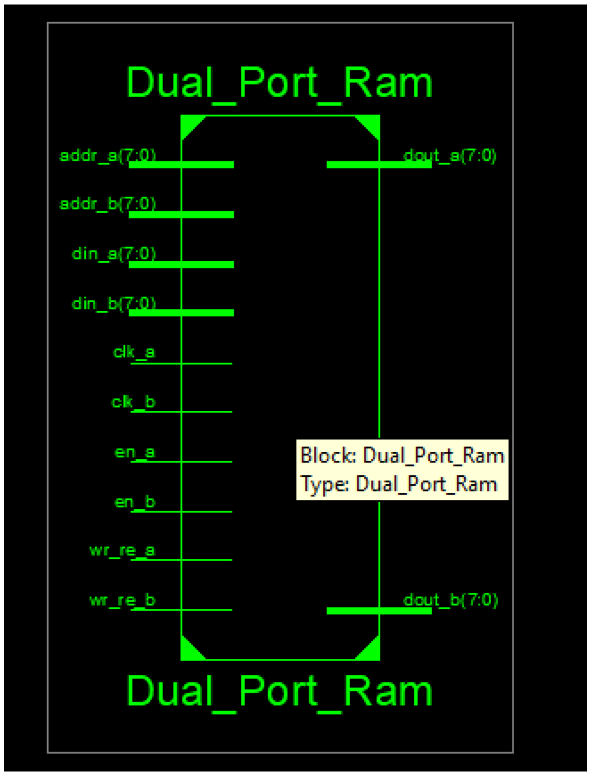
Circuit synthesis code:

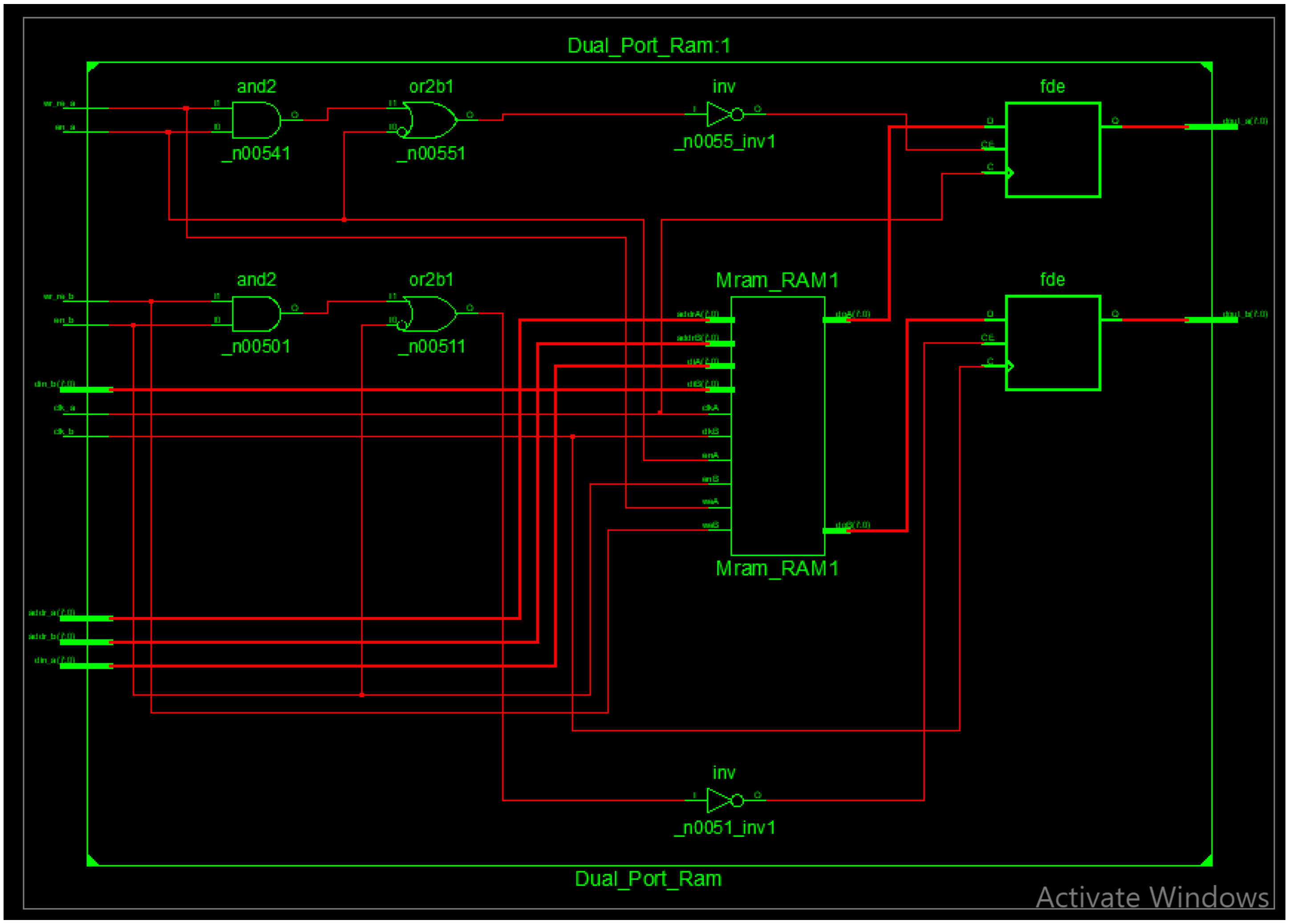
In the second method, we designed the memory by one processes.



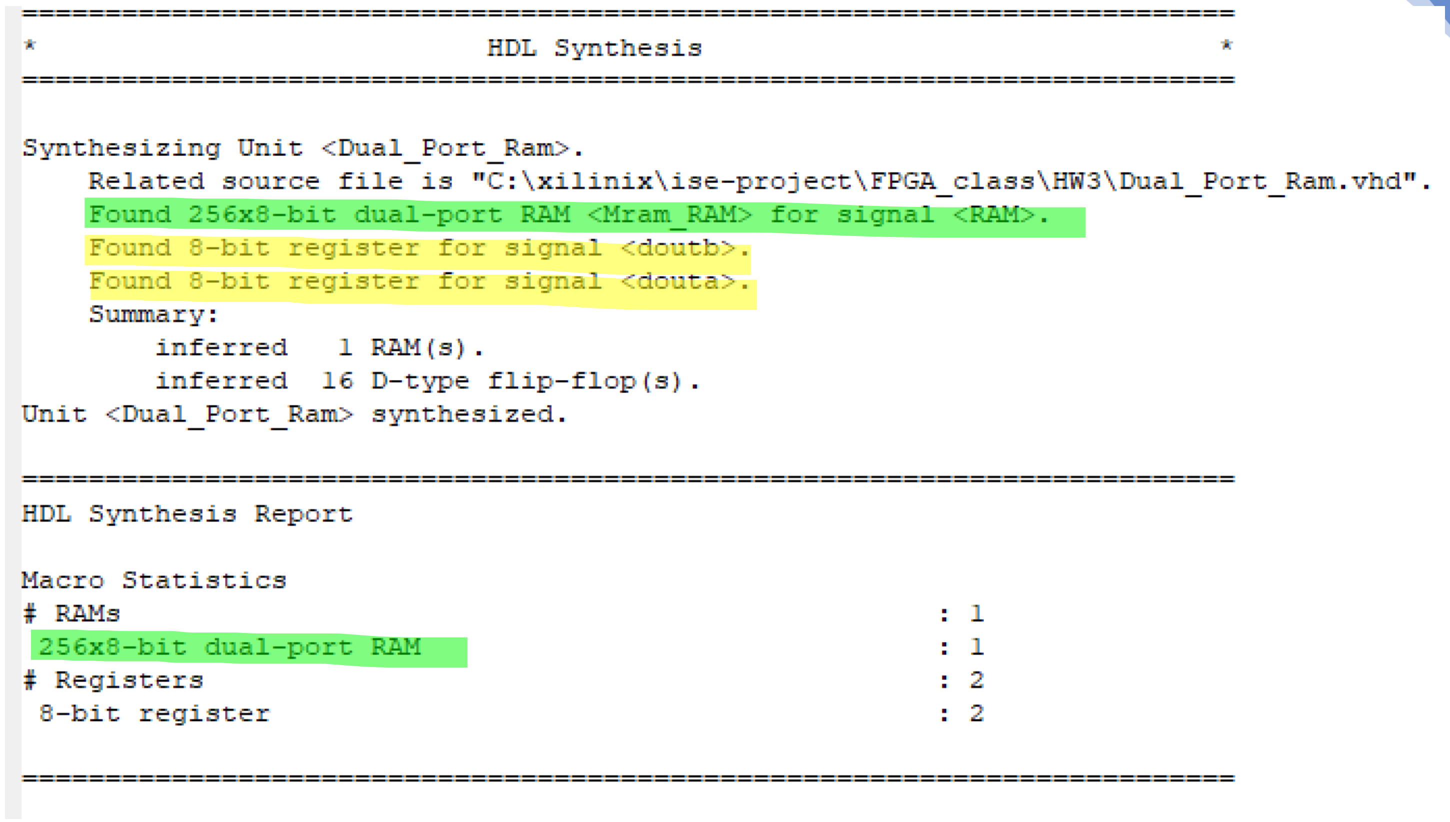
```vhdl
entity Dual_Port_Ram2 is
port(
din_a, din_b:in std_logic_vector(7 downto 0);
dout_a, dout_b: out std_logic_vector(7 downto 0);
addr_a, addr_b: in std_logic_vector(7 downto 0);
clk_a, clk_b: in std_logic;
en_a, en_b: in std_logic;
wr_re_a, wr_re_b: in std_logic
);
end Dual_Port_Ram2;

architecture Behavioral of Dual_Port_Ram2 is
type memory is array(0 to 255)of std_logic_vector(7 downto 0);
signal RAM: memory;

begin
process(clk_a, clk_b)

begin
if(clk_a' event and clk_a = '1') then
   if(en_a = '1') then
      if( wr_re_a = '1') then RAM(conv_integer(addr_a)) <= din_a;
      else
         dout_a <= RAM(conv_integer(addr_a));
      end if;
   end if;
end if;

if(clk_b' event and clk_b = '1') then
   if(en_b = '1') then
      if( wr_re_b = '1') then RAM(conv_integer(addr_b))<= din_b;
      else
         dout_b <= RAM(conv_integer(addr_b));
      end if;
   end if;
end if;
end process;
```

Result of RTL Schematic:

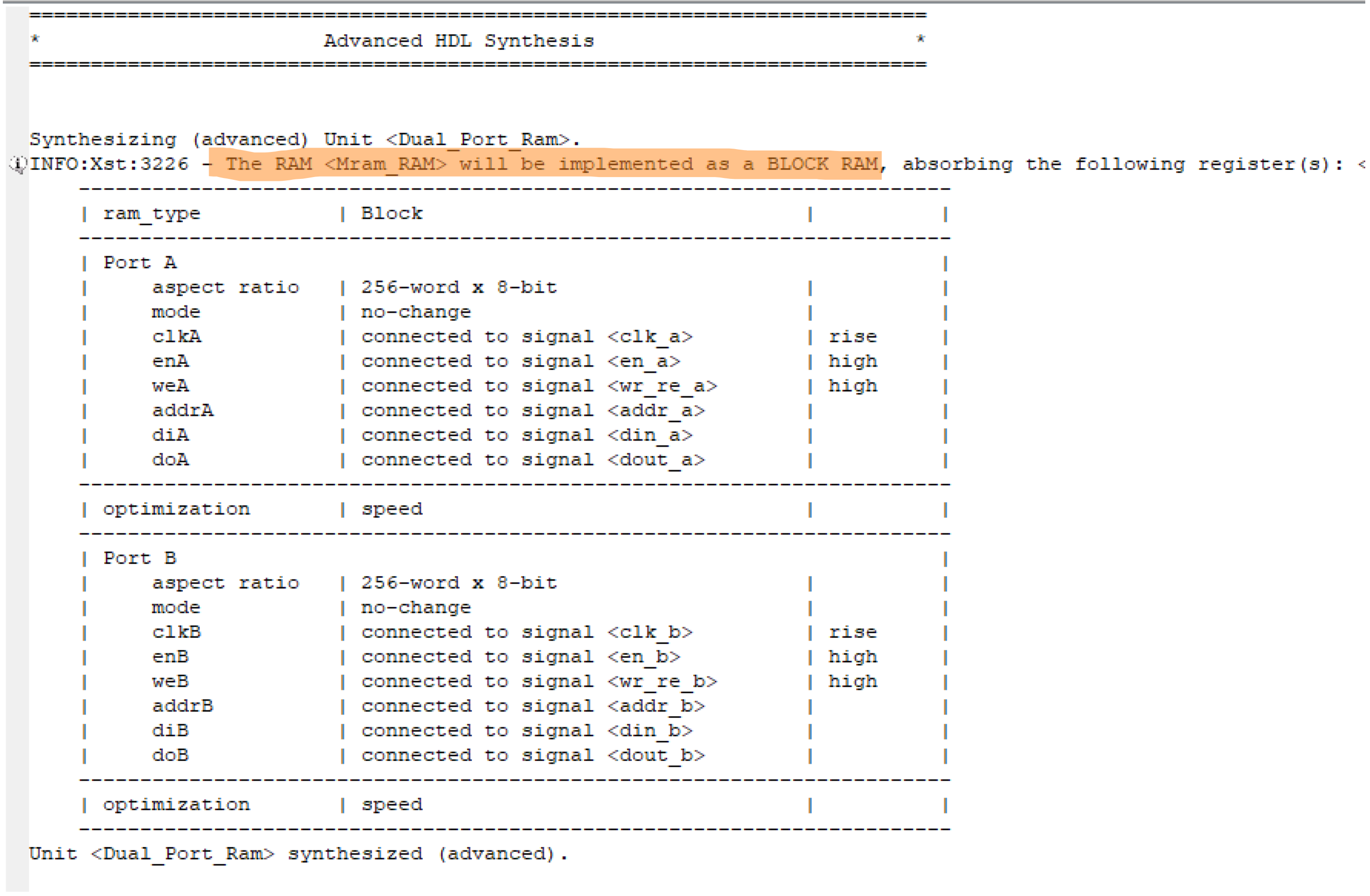




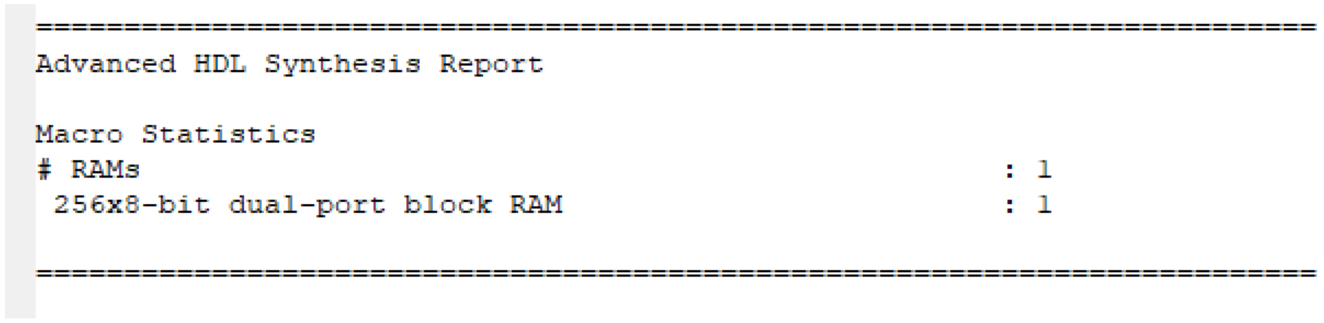
The result and review of the circuit synthesis and extracted elements:



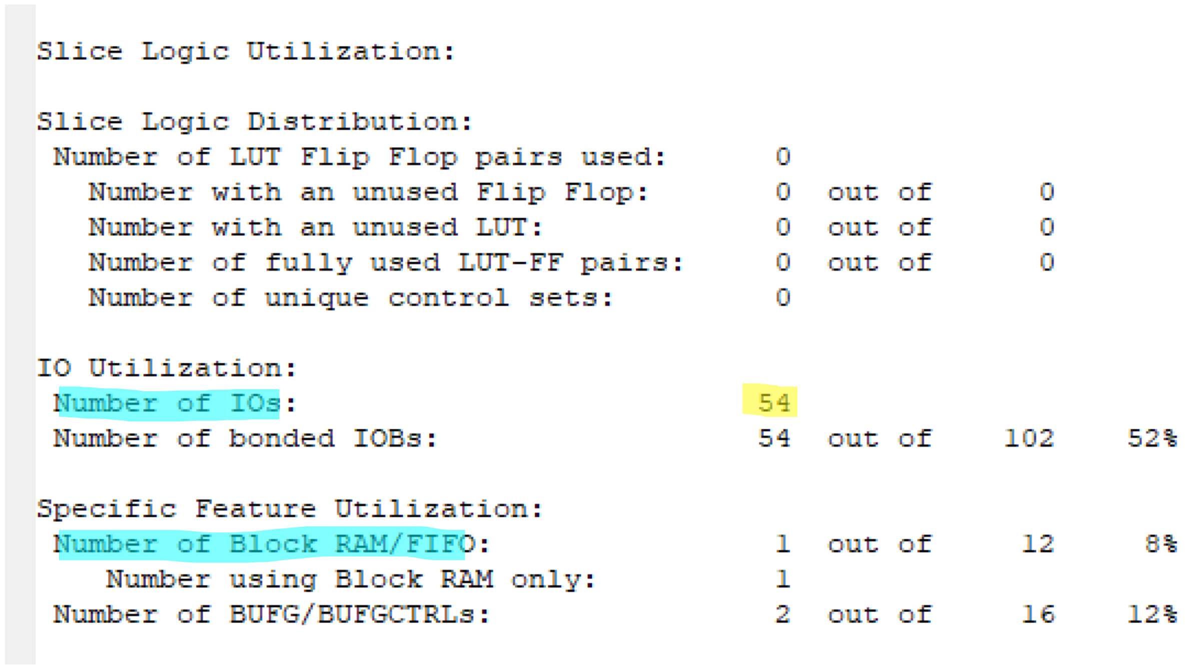
```
=========================================================================
*                          HDL Synthesis                                *
=========================================================================

Synthesizing Unit <Dual_Port_Ram>.
    Related source file is "C:\xilinix\ise-project\FPGA_class\HW3\Dual_Port_Ram.vhd".
    Found 256x8-bit dual-port RAM <Mram_RAM> for signal <RAM>.
    Found 8-bit register for signal <doutb>.
    Found 8-bit register for signal <douta>.
    Summary:
        inferred   1 RAM(s).
        inferred  16 D-type flip-flop(s).
Unit <Dual_Port_Ram> synthesized.

=========================================================================
HDL Synthesis Report

Macro Statistics
# RAMs                                                 : 1
 256x8-bit dual-port RAM                               : 1
# Registers                                            : 2
 8-bit register                                        : 2

=========================================================================
```



```
=========================================================================
*                       Advanced HDL Synthesis                          *
=========================================================================


Synthesizing (advanced) Unit <Dual_Port_Ram>.
INFO:Xst:3226 - The RAM <Mram_RAM> will be implemented as a BLOCK RAM, absorbing the following register(s): <
    -----------------------------------------------------------------------
    | ram_type           | Block                                  |      |
    -----------------------------------------------------------------------
    | Port A                                                             |
    |     aspect ratio   | 256-word x 8-bit                       |      |
    |     mode           | no-change                              |      |
    |     clkA           | connected to signal <clk_a>            | rise |
    |     enA            | connected to signal <en_a>             | high |
    |     weA            | connected to signal <wr_re_a>          | high |
    |     addrA          | connected to signal <addr_a>           |      |
    |     diA            | connected to signal <din_a>            |      |
    |     doA            | connected to signal <dout_a>           |      |
    -----------------------------------------------------------------------
    | optimization       | speed                                  |      |
    -----------------------------------------------------------------------
    | Port B                                                             |
    |     aspect ratio   | 256-word x 8-bit                       |      |
    |     mode           | no-change                              |      |
    |     clkB           | connected to signal <clk_b>            | rise |
    |     enB            | connected to signal <en_b>             | high |
    |     weB            | connected to signal <wr_re_b>          | high |
    |     addrB          | connected to signal <addr_b>           |      |
    |     diB            | connected to signal <din_b>            |      |
    |     doB            | connected to signal <dout_b>           |      |
    -----------------------------------------------------------------------
    | optimization       | speed                                  |      |
    -----------------------------------------------------------------------
Unit <Dual_Port_Ram> synthesized (advanced).
```



```
========================================================================
Advanced HDL Synthesis Report

Macro Statistics
# RAMs                                                     : 1
 256x8-bit dual-port block RAM                             : 1

========================================================================
```



```
Slice Logic Utilization:

Slice Logic Distribution:
 Number of LUT Flip Flop pairs used:          0
   Number with an unused Flip Flop:           0  out of      0
   Number with an unused LUT:                 0  out of      0
   Number of fully used LUT-FF pairs:         0  out of      0
   Number of unique control sets:             0

IO Utilization:
 Number of IOs:                              54
 Number of bonded IOBs:                      54  out of    102    52%

Specific Feature Utilization:
 Number of Block RAM/FIFO:                    1  out of     12     8%
    Number using Block RAM only:              1
 Number of BUFG/BUFGCTRLs:                    2  out of     16    12%
```

Number of I/O = 2*Din + 2*Add + 2*Dout + 2*En + 2*We + 2*Clk

= 16 + 16 + 16 + 2 + 2 +2 = 54

Note: When we define 2 processes, it is not possible to assign a value to the same signal inside those two processes

The synthesis tool gives an error.

more details:

This description of a dual-clock RAM is wrong. You need to use either:

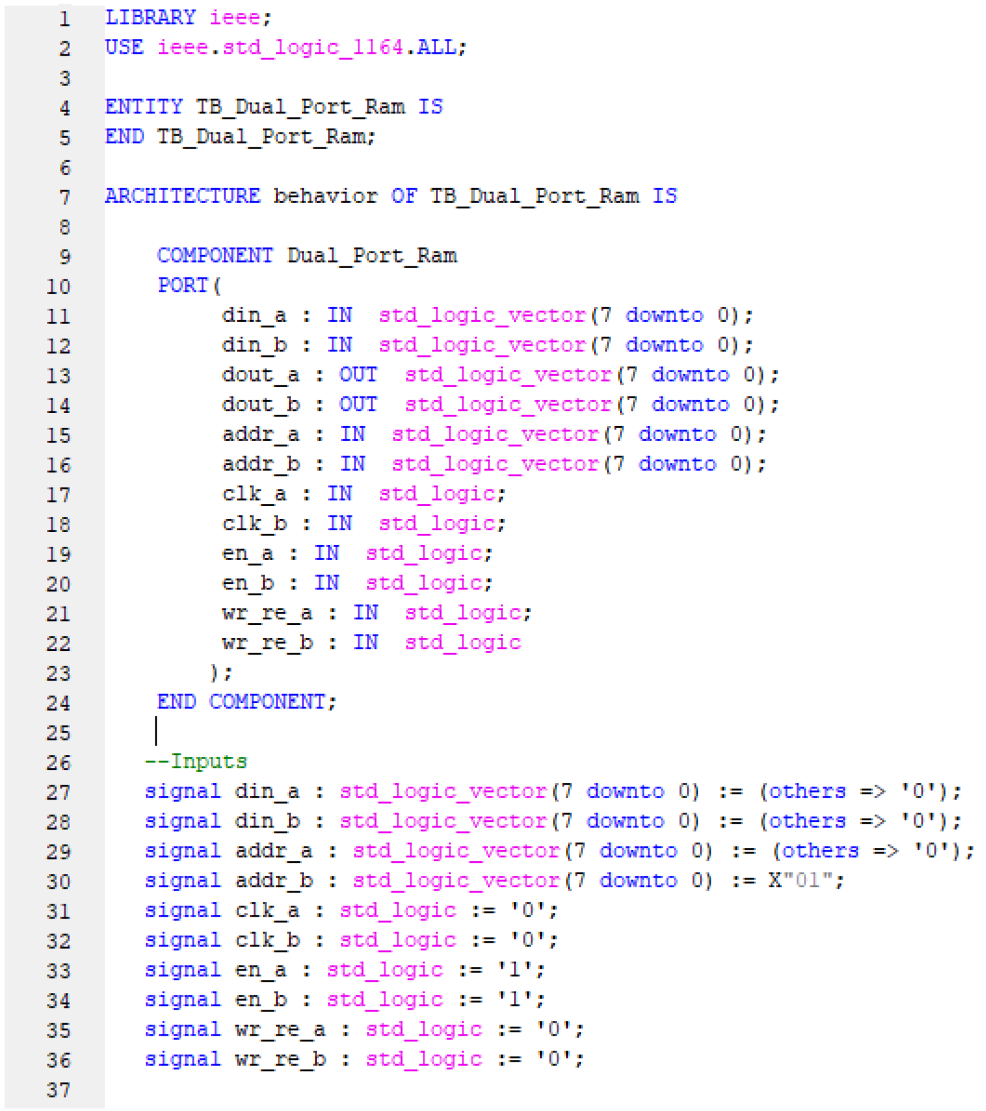
- **a process with two clocks, or**
- **a shared variable.**

Using one signal and two processes is not correct. It creates multiple drives on a signal. This in turn creates a multiple source problem. While your simulation will work, because of the resolved type std_logic_vector in your user defined array type, synthesis will fail.

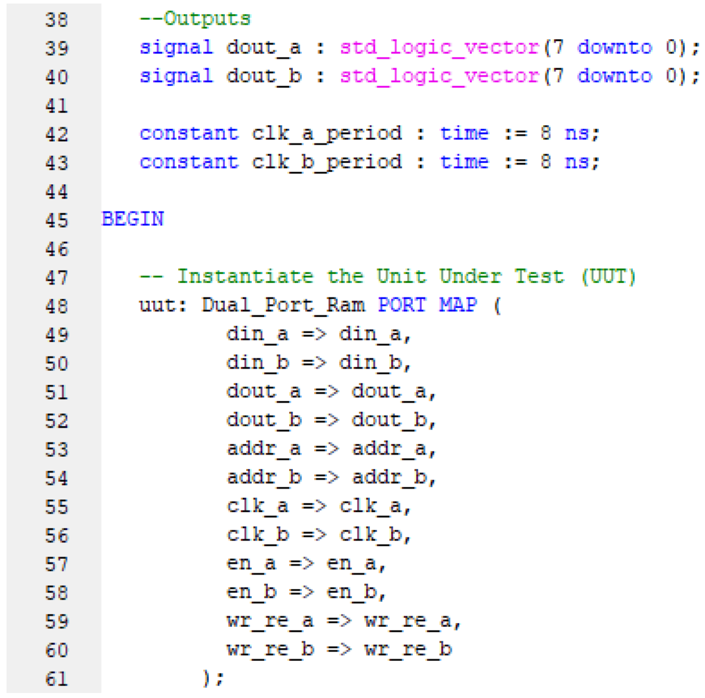
In addition, to allow inference of BlockRAMs, you need to represent the internal structure of BlockRAMs in you VHDL code. This means you need to add pipeline registers on the address path.

To create true (dual-clock) dual-port RAM, I need to create two clocked processes. This requires me to use a shared variable for the memory itself

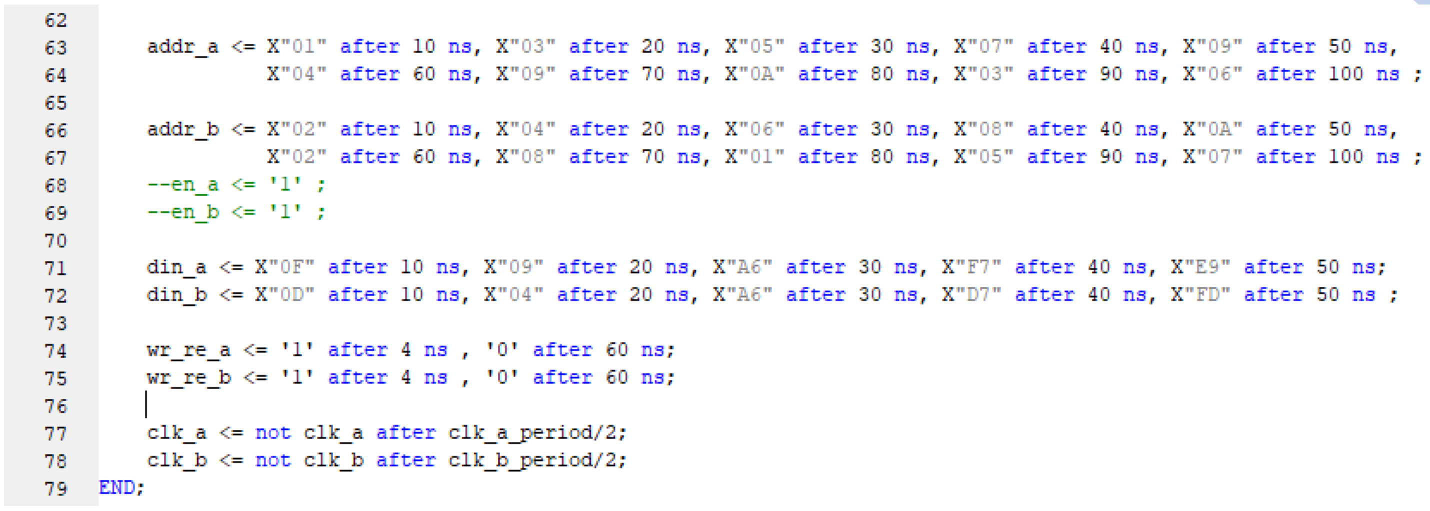
Circuit Test Bench code:



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

ENTITY TB_Dual_Port_Ram IS
END TB_Dual_Port_Ram;

ARCHITECTURE behavior OF TB_Dual_Port_Ram IS

    COMPONENT Dual_Port_Ram
    PORT(
         din_a : IN  std_logic_vector(7 downto 0);
         din_b : IN  std_logic_vector(7 downto 0);
         dout_a : OUT  std_logic_vector(7 downto 0);
         dout_b : OUT  std_logic_vector(7 downto 0);
         addr_a : IN  std_logic_vector(7 downto 0);
         addr_b : IN  std_logic_vector(7 downto 0);
         clk_a : IN  std_logic;
         clk_b : IN  std_logic;
         en_a : IN  std_logic;
         en_b : IN  std_logic;
         wr_re_a : IN  std_logic;
         wr_re_b : IN  std_logic
        );
    END COMPONENT;

   --Inputs
   signal din_a : std_logic_vector(7 downto 0) := (others => '0');
   signal din_b : std_logic_vector(7 downto 0) := (others => '0');
   signal addr_a : std_logic_vector(7 downto 0) := (others => '0');
   signal addr_b : std_logic_vector(7 downto 0) := X"01";
   signal clk_a : std_logic := '0';
   signal clk_b : std_logic := '0';
   signal en_a : std_logic := '1';
   signal en_b : std_logic := '1';
   signal wr_re_a : std_logic := '0';
   signal wr_re_b : std_logic := '0';

```



```vhdl
   --Outputs
   signal dout_a : std_logic_vector(7 downto 0);
   signal dout_b : std_logic_vector(7 downto 0);

   constant clk_a_period : time := 8 ns;
   constant clk_b_period : time := 8 ns;

BEGIN

   -- Instantiate the Unit Under Test (UUT)
   uut: Dual_Port_Ram PORT MAP (
          din_a => din_a,
          din_b => din_b,
          dout_a => dout_a,
          dout_b => dout_b,
          addr_a => addr_a,
          addr_b => addr_b,
          clk_a => clk_a,
          clk_b => clk_b,
          en_a => en_a,
          en_b => en_b,
          wr_re_a => wr_re_a,
          wr_re_b => wr_re_b
        );
```



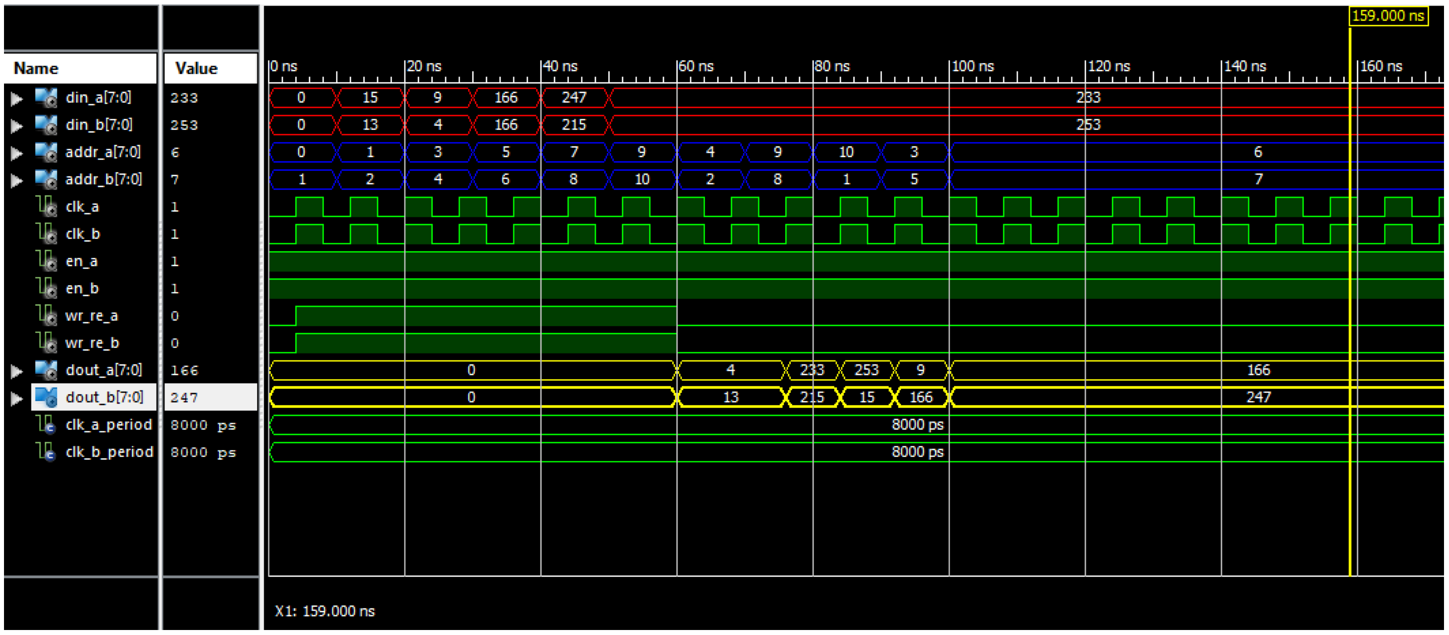
```

    addr_a <= X"01" after 10 ns, X"03" after 20 ns, X"05" after 30 ns, X"07" after 40 ns, X"09" after 50 ns,
              X"04" after 60 ns, X"09" after 70 ns, X"0A" after 80 ns, X"03" after 90 ns, X"06" after 100 ns ;

    addr_b <= X"02" after 10 ns, X"04" after 20 ns, X"06" after 30 ns, X"08" after 40 ns, X"0A" after 50 ns,
              X"02" after 60 ns, X"08" after 70 ns, X"01" after 80 ns, X"05" after 90 ns, X"07" after 100 ns ;
    --en_a <= '1' ;
    --en_b <= '1' ;

    din_a <= X"0F" after 10 ns, X"09" after 20 ns, X"A6" after 30 ns, X"F7" after 40 ns, X"E9" after 50 ns;
    din_b <= X"0D" after 10 ns, X"04" after 20 ns, X"A6" after 30 ns, X"D7" after 40 ns, X"FD" after 50 ns ;

    wr_re_a <= '1' after 4 ns , '0' after 60 ns;
    wr_re_b <= '1' after 4 ns , '0' after 60 ns;

    clk_a <= not clk_a after clk_a_period/2;
    clk_b <= not clk_b after clk_b_period/2;
END;
```

The values we wrote in memory addresses X “00” to X “10”:

Address	Data
X “00”	0
X “01”	15
X “02”	13
X “03”	9
X “04”	4
X “05”	166
X “06”	166
X “07”	247
X “08”	215
X “09”	233
X “0A”	253

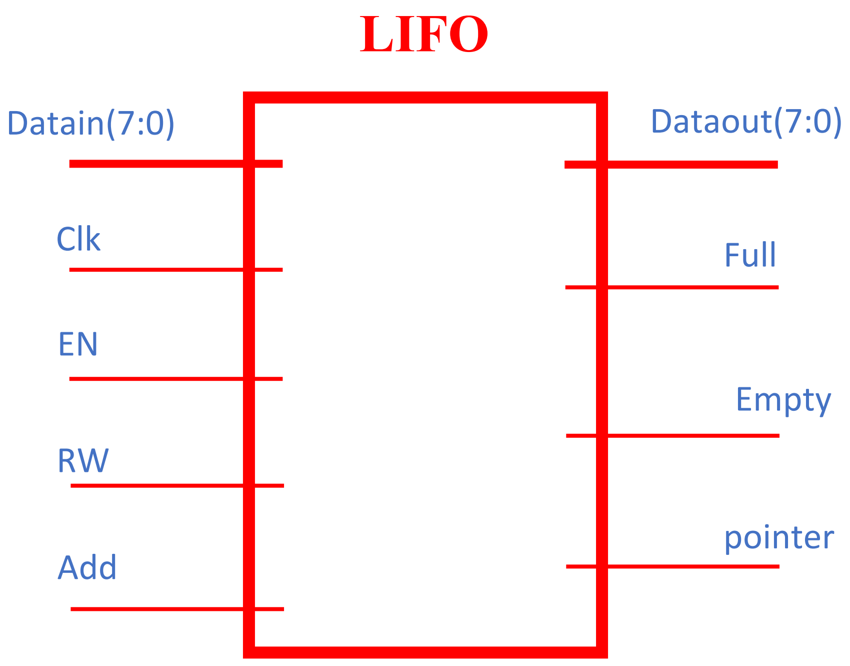
Simulation result:

After writing data in Dual Port memery by using two ports, We read data from those address and we see that our memory works properly through both ports.



۲- یک حافظه با اندازه متغیر به صورت LIFO طراحی کنید .

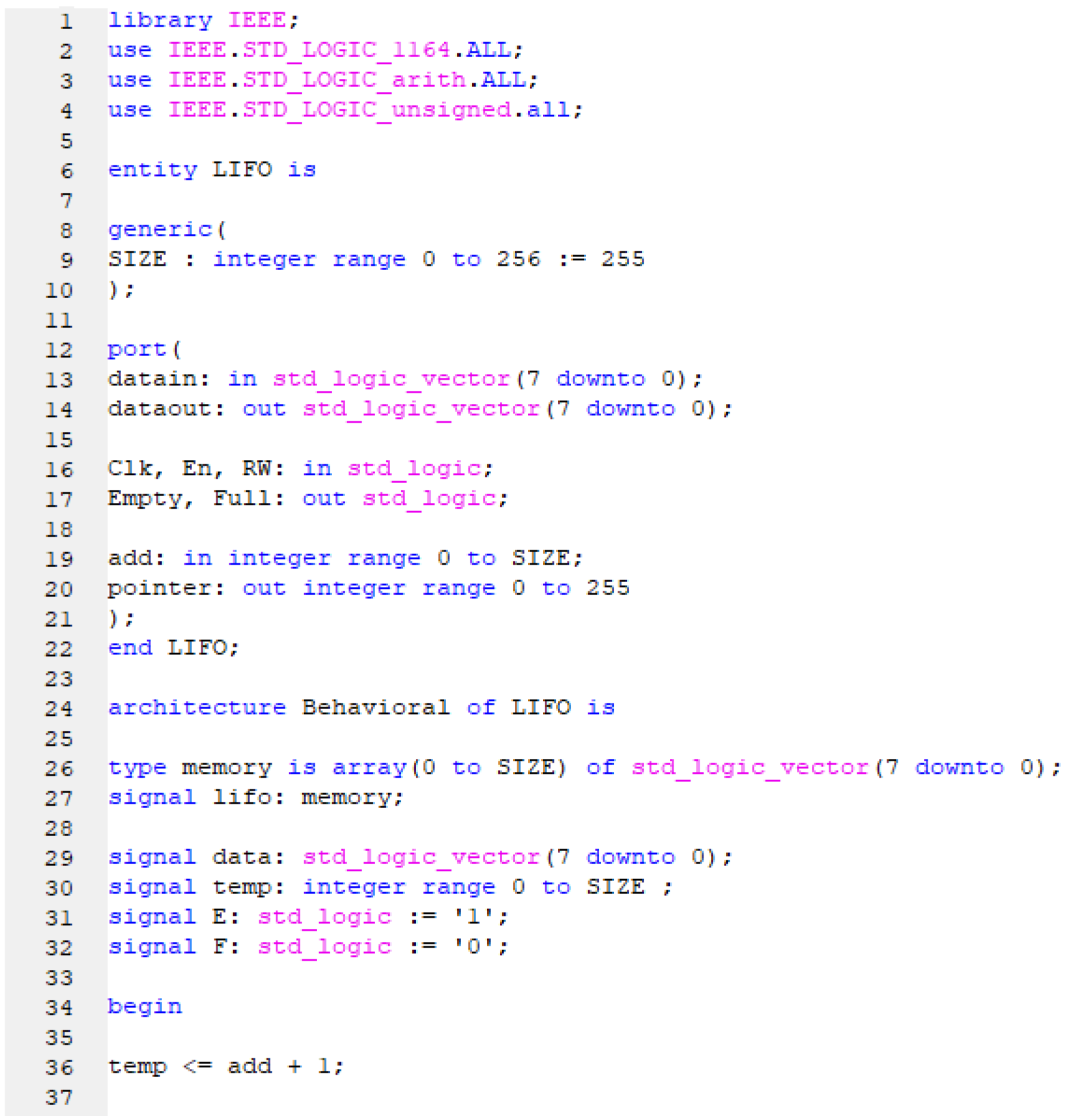
- یک سیگنال constant در نظر بگیرید که تعیین کننده اندازه حافظه باشد.
- این حافظه باید ۸ بیت ورودی برای تعیین آدرس اولیه در stack داشته باشد.
- زمانی که همه خانه های حافظه دیتا داشته باشند و نتوان داده جدیدی در آن نوشت، باید خروجی full فعال باشد.
- زمانی که همه خانه های حافظه خالی باشند و نتوان داده ای از آن خواند، باید خروجی empty فعال باشد.
- یک ورودی wr_rd در نظر بگیرید که خواندن و یا نوشتن از حافظه را تعیین میکند.
- این حافظه یک ورودی ۸ بیتی برای نوشتن دیتا دارد- .این حافظه یک خروجی ۸ بیتی برای خواندن دیتا دارد .



What is the pointer output?

In order to be able to see the stack pointer in the testbench, we added a pointer output to the circuit to ensure that the memory is working correctly.

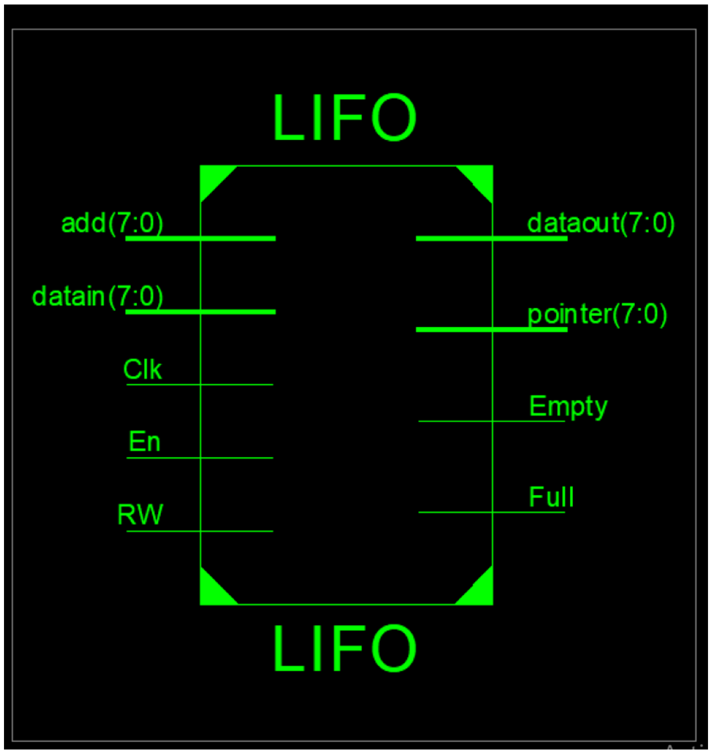
Circuit synthesis code:

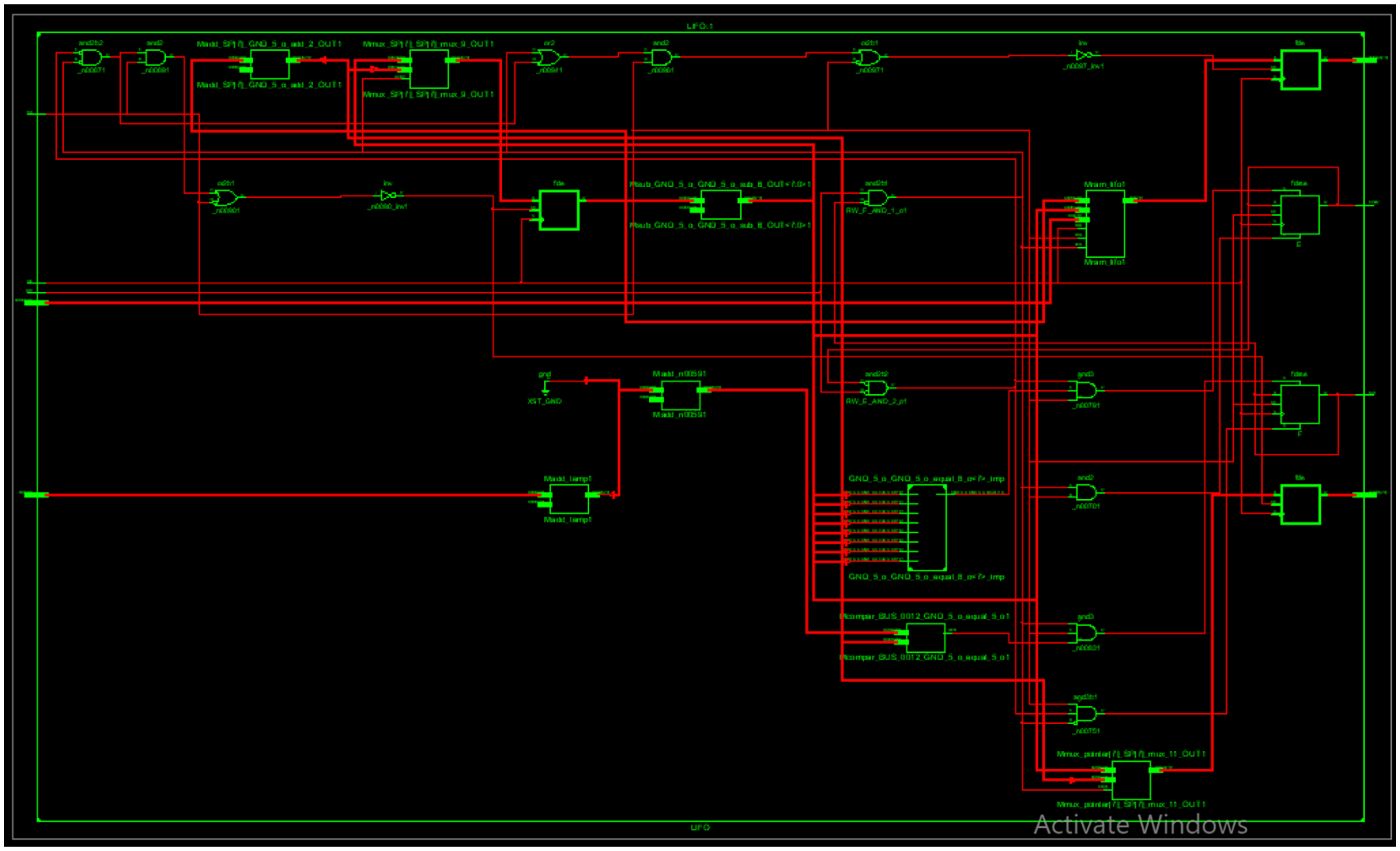


```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_arith.ALL;
use IEEE.STD_LOGIC_unsigned.all;

entity LIFO is

generic(
SIZE : integer range 0 to 256 := 255
);

port(
datain: in std_logic_vector(7 downto 0);
dataout: out std_logic_vector(7 downto 0);

Clk, En, RW: in std_logic;
Empty, Full: out std_logic;

add: in integer range 0 to SIZE;
pointer: out integer range 0 to 255
);
end LIFO;

architecture Behavioral of LIFO is

type memory is array(0 to SIZE) of std_logic_vector(7 downto 0);
signal lifo: memory;

signal data: std_logic_vector(7 downto 0);
signal temp: integer range 0 to SIZE ;
signal E: std_logic := '1';
signal F: std_logic := '0';

begin

temp <= add + 1;

```

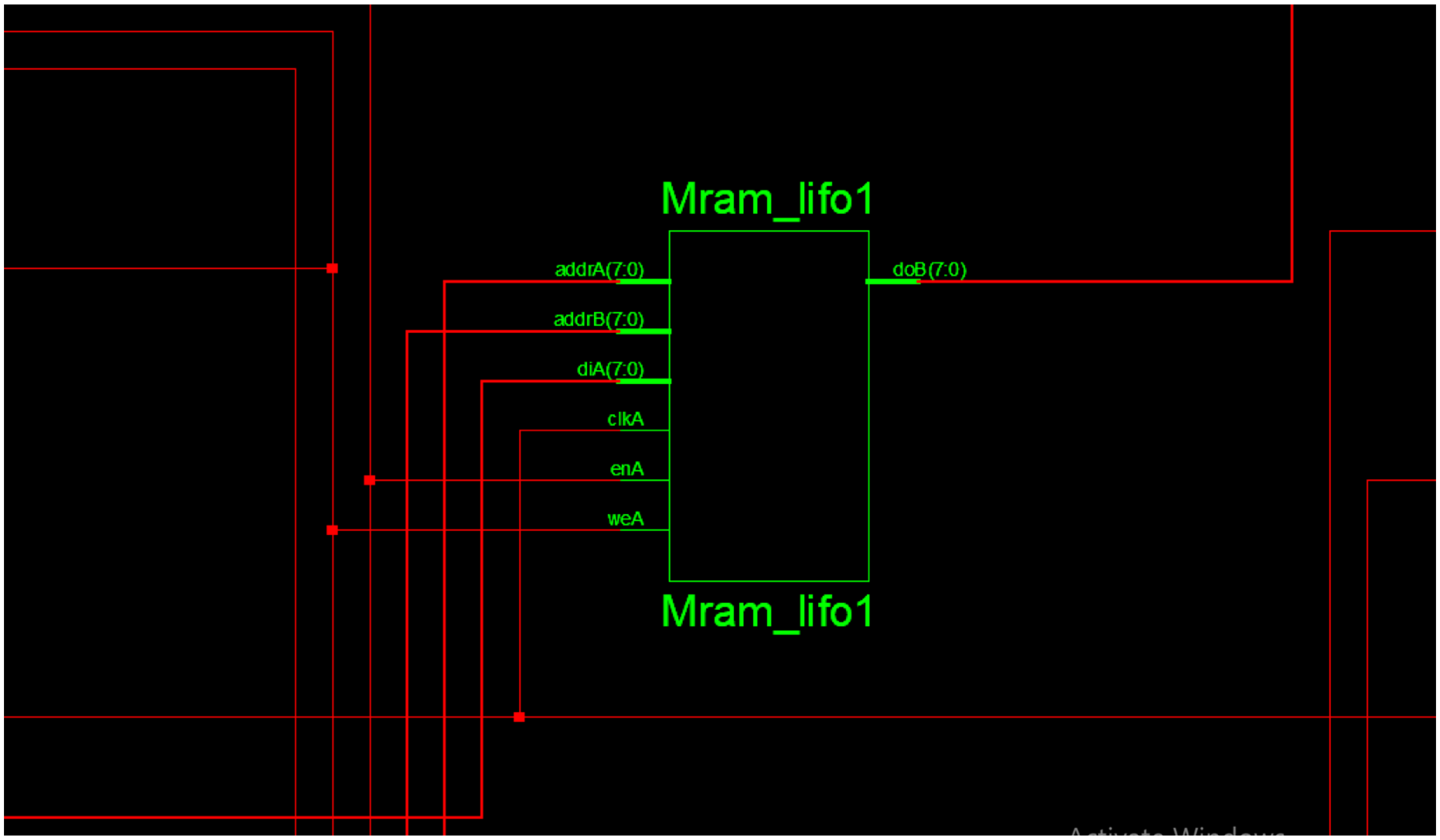
```
process(clk, En)

variable SP: integer range 0 to 255 := 1 ;

begin

if( En ='1') then
   if(clk' event and clk = '1') then
      if( Rw = '1' and F = '0') then
         lifo(SP) <= datain;
         SP := SP + 1;
         pointer <= SP;
         E <= '0';

         if( SP = temp + 1 ) then F <= '1';
         end if;

      elsif (Rw = '0' and E = '0') then
         SP := SP - 1;
         data <= lifo(SP);
         pointer <= SP;
         F <= '0';

         if ( SP = 1 ) then E <= '1';
         end if;
      end if;
   end if;
end if;

end process;
Full <= F;
Empty <= E;
dataout <= data;

end Behavioral;
```

Result of RTL Schematic:

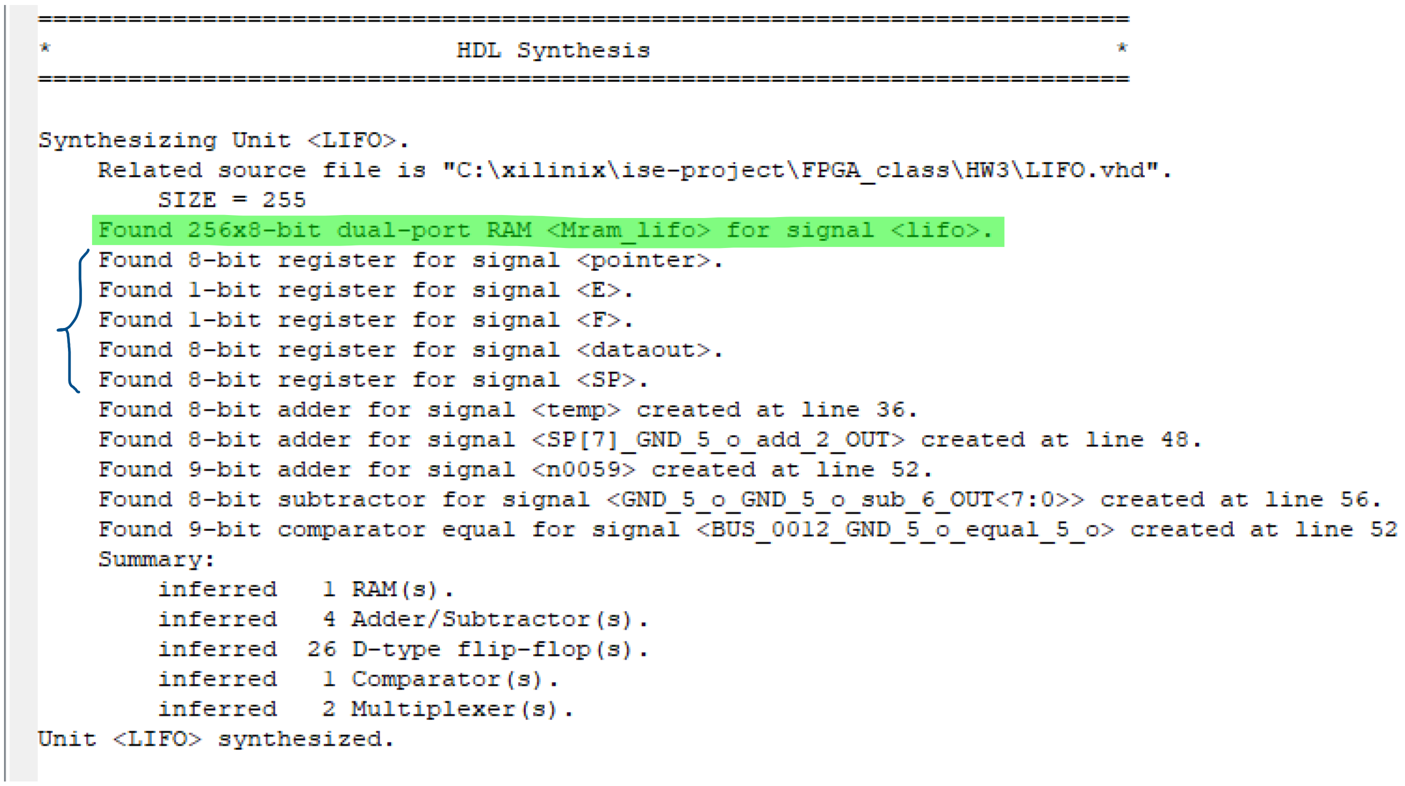




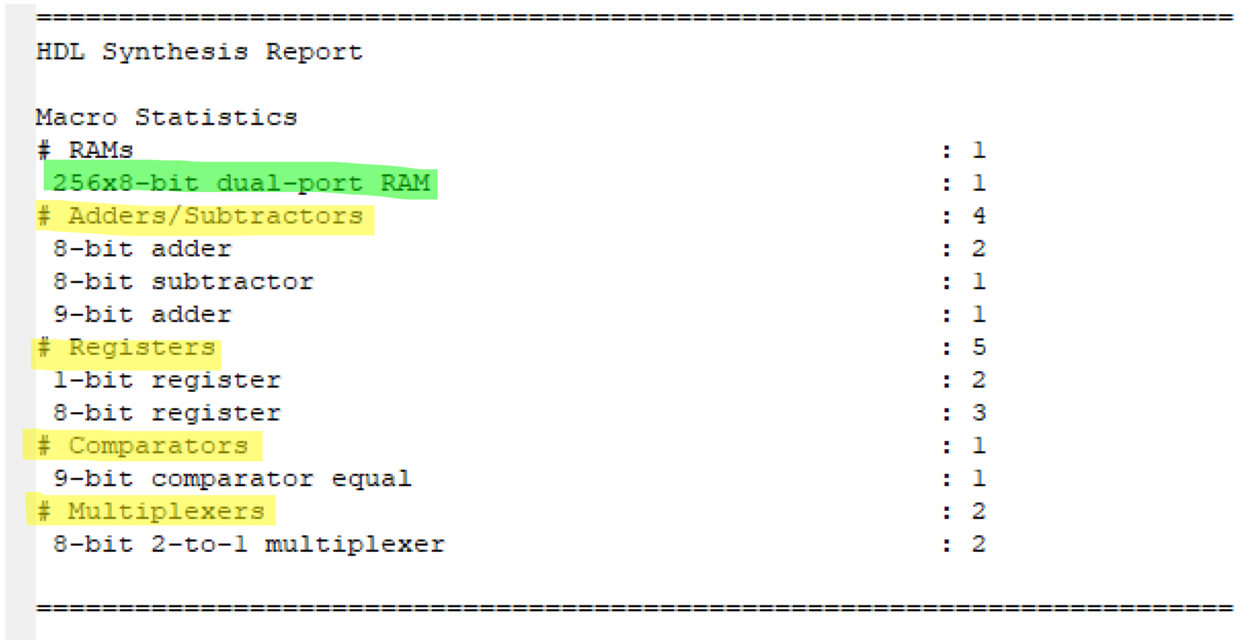
When we zoom in, we can see mram_lifo like this.



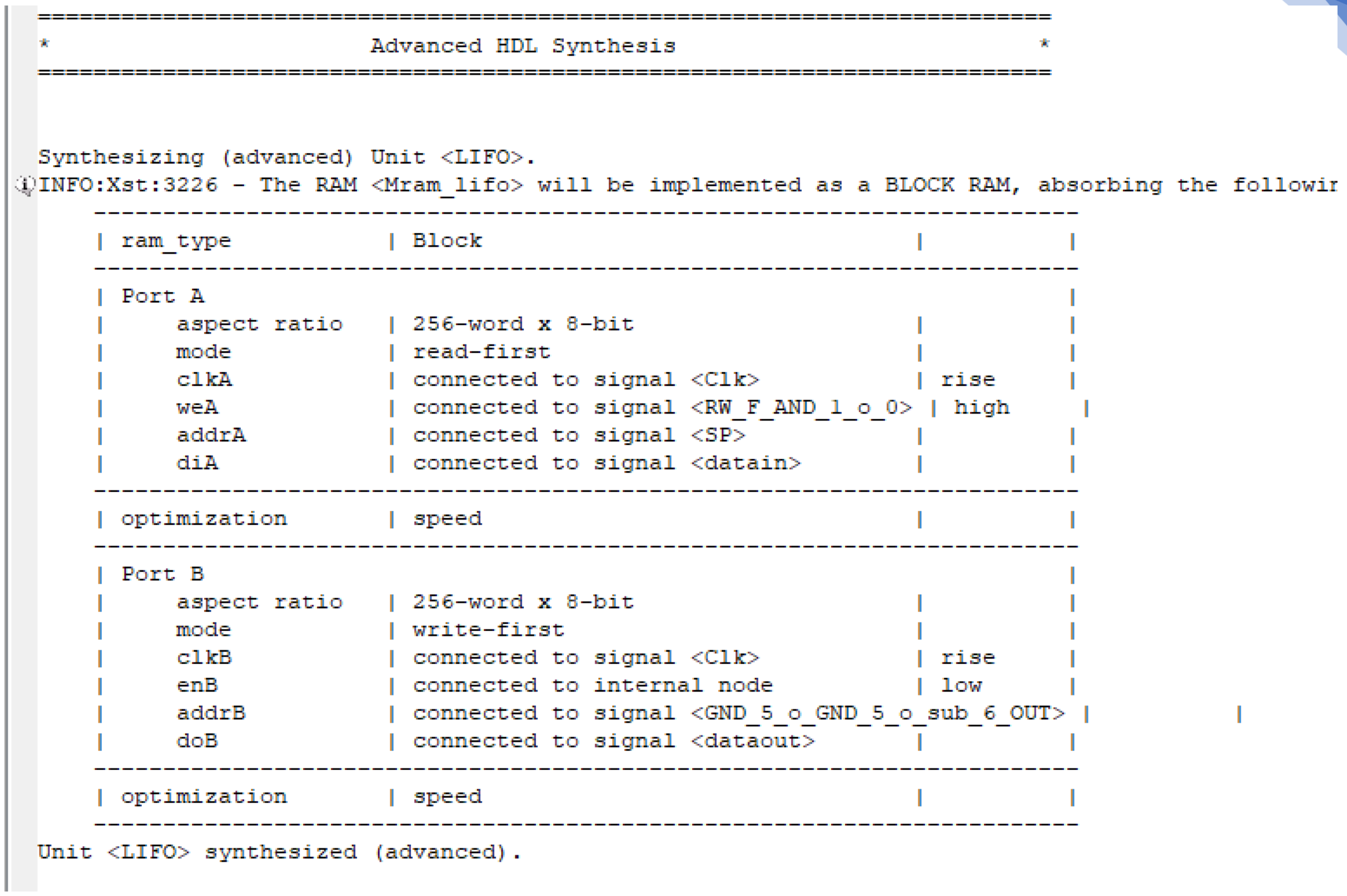
The result and review of the circuit synthesis and extracted elements:



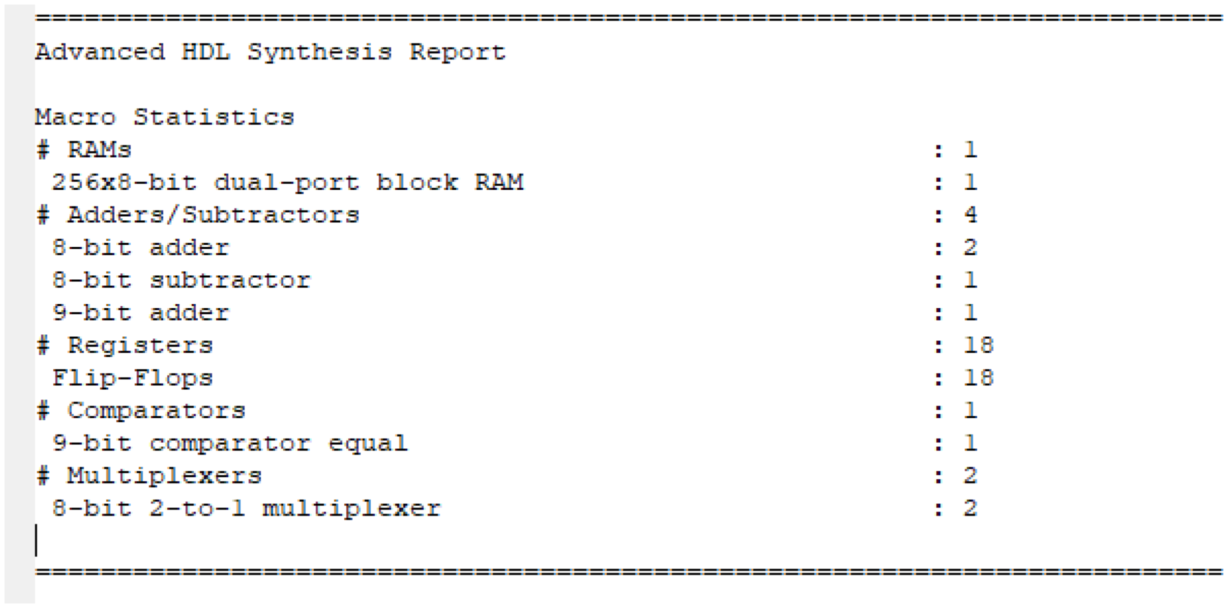
```
=========================================================================
*                           HDL Synthesis                               *
=========================================================================

Synthesizing Unit <LIFO>.
    Related source file is "C:\xilinix\ise-project\FPGA_class\HW3\LIFO.vhd".
        SIZE = 255
    Found 256x8-bit dual-port RAM <Mram_lifo> for signal <lifo>.
    Found 8-bit register for signal <pointer>.
    Found 1-bit register for signal <E>.
    Found 1-bit register for signal <F>.
    Found 8-bit register for signal <dataout>.
    Found 8-bit register for signal <SP>.
    Found 8-bit adder for signal <temp> created at line 36.
    Found 8-bit adder for signal <SP[7]_GND_5_o_add_2_OUT> created at line 48.
    Found 9-bit adder for signal <n0059> created at line 52.
    Found 8-bit subtractor for signal <GND_5_o_GND_5_o_sub_6_OUT<7:0>> created at line 56.
    Found 9-bit comparator equal for signal <BUS_0012_GND_5_o_equal_5_o> created at line 52
    Summary:
        inferred   1 RAM(s).
        inferred   4 Adder/Subtractor(s).
        inferred  26 D-type flip-flop(s).
        inferred   1 Comparator(s).
        inferred   2 Multiplexer(s).
Unit <LIFO> synthesized.
```



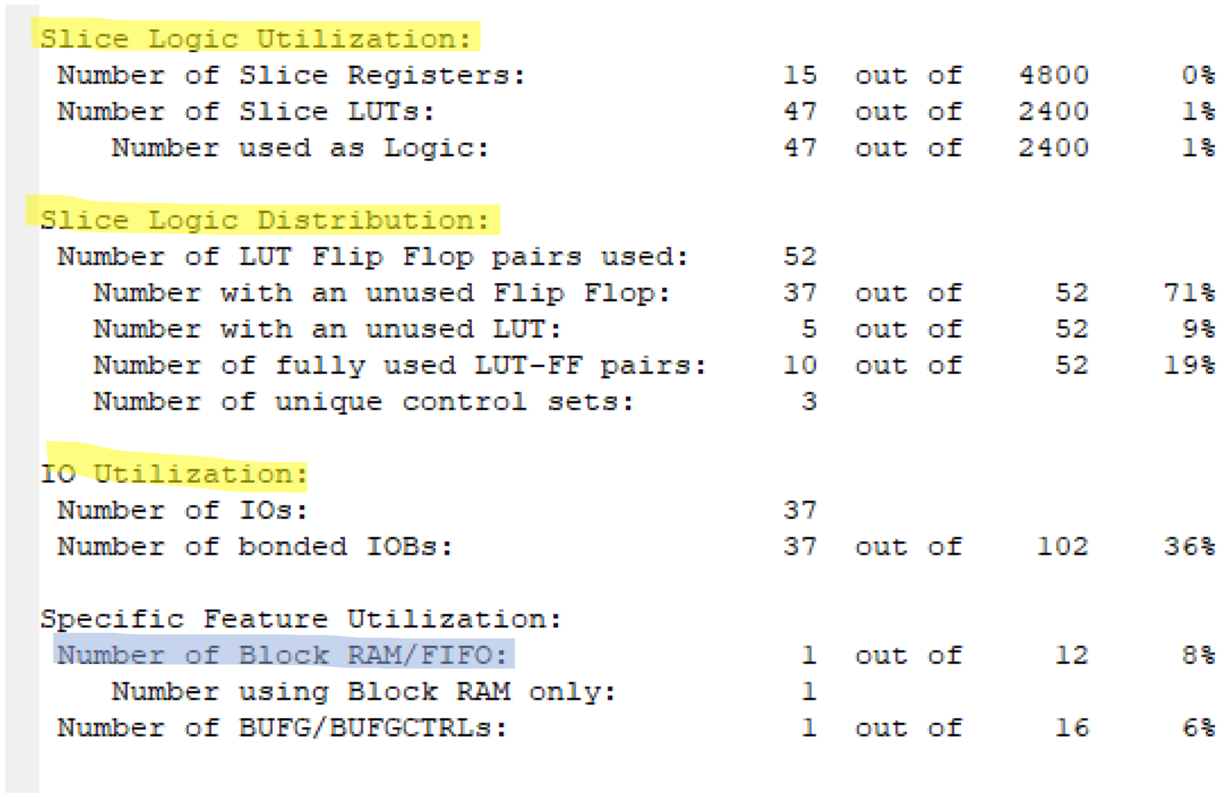
```
=========================================================================
HDL Synthesis Report

Macro Statistics
# RAMs                                                 : 1
 256x8-bit dual-port RAM                               : 1
# Adders/Subtractors                                   : 4
 8-bit adder                                           : 2
 8-bit subtractor                                      : 1
 9-bit adder                                           : 1
# Registers                                            : 5
 1-bit register                                        : 2
 8-bit register                                        : 3
# Comparators                                          : 1
 9-bit comparator equal                                : 1
# Multiplexers                                         : 2
 8-bit 2-to-1 multiplexer                              : 2

=========================================================================
```



```
=========================================================================
*                        Advanced HDL Synthesis                         *
=========================================================================


Synthesizing (advanced) Unit <LIFO>.
INFO:Xst:3226 - The RAM <Mram_lifo> will be implemented as a BLOCK RAM, absorbing the followir
    -----------------------------------------------------------------------
    | ram_type           | Block                                |          |
    -----------------------------------------------------------------------
    | Port A                                                               |
    |     aspect ratio   | 256-word x 8-bit                     |          |
    |     mode           | read-first                           |          |
    |     clkA           | connected to signal <Clk>            | rise     |
    |     weA            | connected to signal <RW_F_AND_1_o_0> | high     |
    |     addrA          | connected to signal <SP>             |          |
    |     diA            | connected to signal <datain>         |          |
    -----------------------------------------------------------------------
    | optimization       | speed                                |          |
    -----------------------------------------------------------------------
    | Port B                                                               |
    |     aspect ratio   | 256-word x 8-bit                     |          |
    |     mode           | write-first                          |          |
    |     clkB           | connected to signal <Clk>            | rise     |
    |     enB            | connected to internal node           | low      |
    |     addrB          | connected to signal <GND_5_o_GND_5_o_sub_6_OUT> |          |
    |     doB            | connected to signal <dataout>        |          |
    -----------------------------------------------------------------------
    | optimization       | speed                                |          |
    -----------------------------------------------------------------------
Unit <LIFO> synthesized (advanced).
```



```
=========================================================================
Advanced HDL Synthesis Report

Macro Statistics
# RAMs                                                 : 1
 256x8-bit dual-port block RAM                         : 1
# Adders/Subtractors                                   : 4
 8-bit adder                                           : 2
 8-bit subtractor                                      : 1
 9-bit adder                                           : 1
# Registers                                            : 18
 Flip-Flops                                            : 18
# Comparators                                          : 1
 9-bit comparator equal                                : 1
# Multiplexers                                         : 2
 8-bit 2-to-1 multiplexer                              : 2

=========================================================================
```

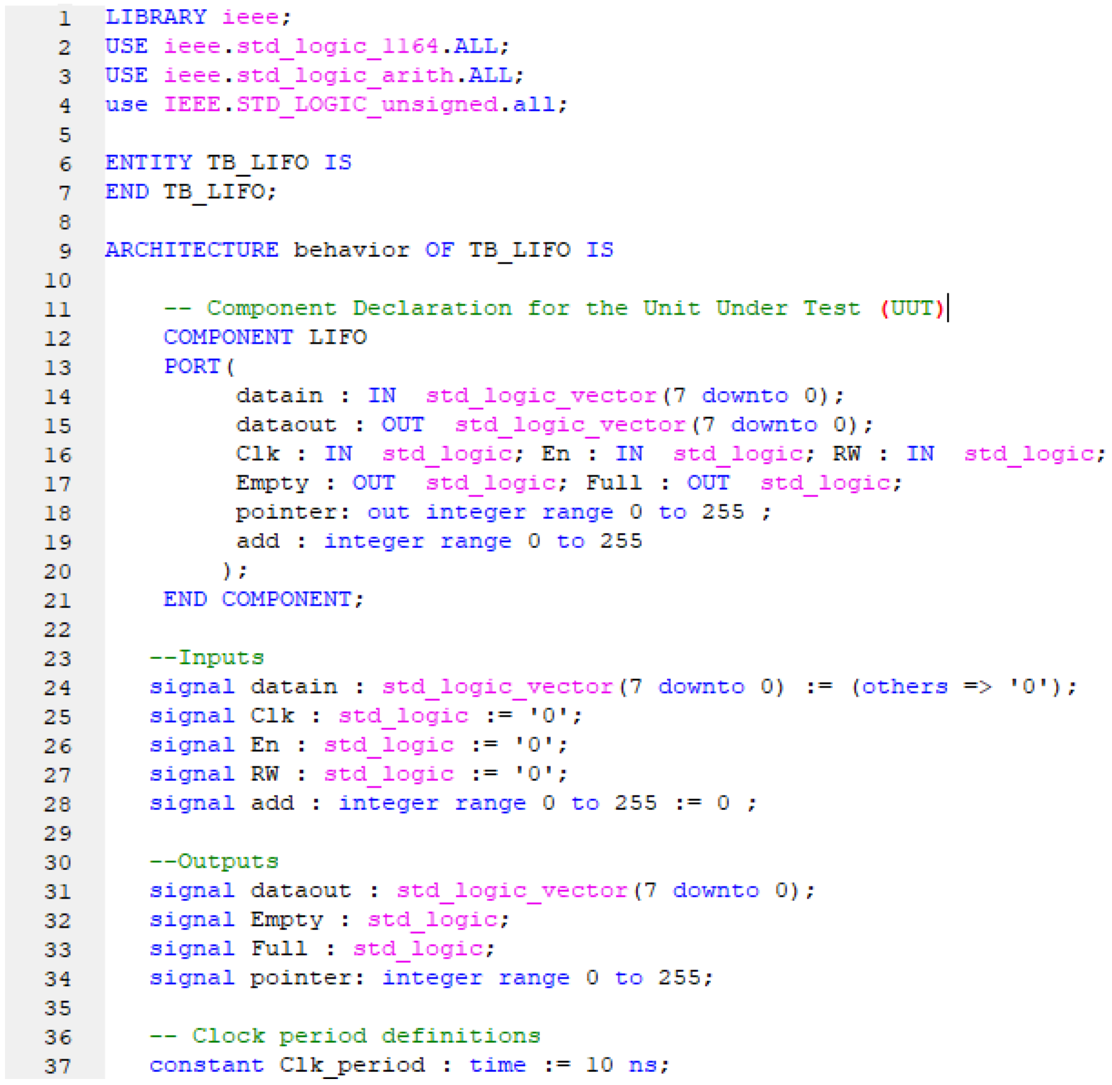


```
Slice Logic Utilization:
 Number of Slice Registers:                 15  out of   4800     0%
 Number of Slice LUTs:                      47  out of   2400     1%
    Number used as Logic:                   47  out of   2400     1%

Slice Logic Distribution:
 Number of LUT Flip Flop pairs used:        52
   Number with an unused Flip Flop:         37  out of     52    71%
   Number with an unused LUT:                5  out of     52     9%
   Number of fully used LUT-FF pairs:       10  out of     52    19%
   Number of unique control sets:            3

IO Utilization:
 Number of IOs:                             37
 Number of bonded IOBs:                     37  out of    102    36%

Specific Feature Utilization:
 Number of Block RAM/FIFO:                   1  out of     12     8%
    Number using Block RAM only:             1
 Number of BUFG/BUFGCTRLs:                   1  out of     16     6%
```

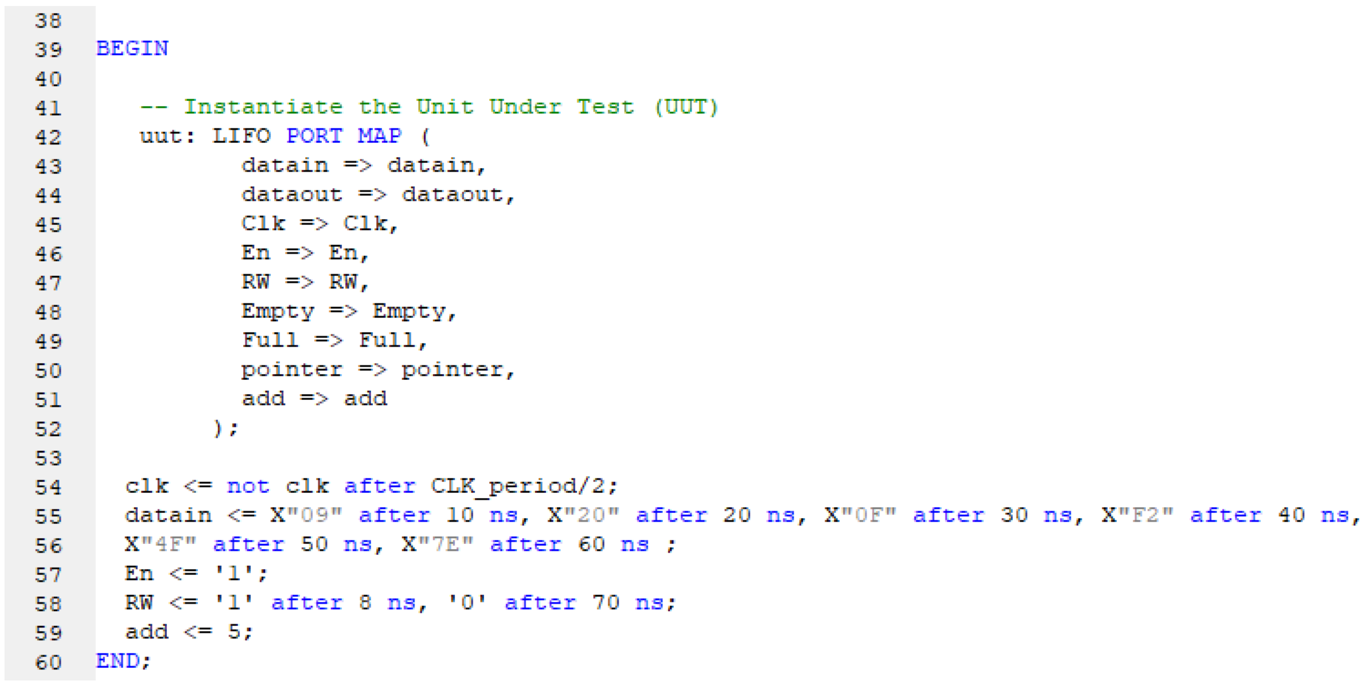
Why did synthesis tool extract Dual Port RAM?

Because we put two 8-bit ports for input, one is address and one is data, and also two 8-bit ports for output, one is data and the other is pointer.

Circuit Test Bench code:

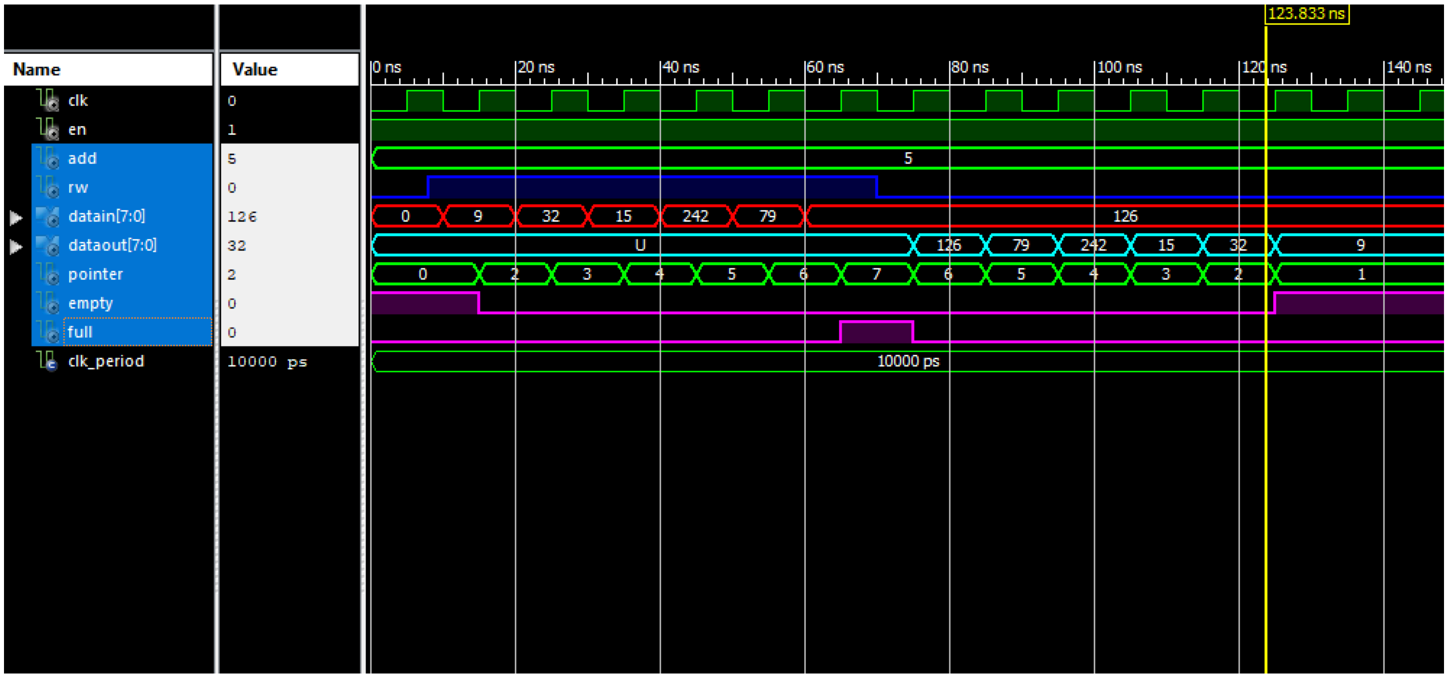


```vhdl
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.std_logic_arith.ALL;
use IEEE.STD_LOGIC_unsigned.all;

ENTITY TB_LIFO IS
END TB_LIFO;

ARCHITECTURE behavior OF TB_LIFO IS

    -- Component Declaration for the Unit Under Test (UUT)
    COMPONENT LIFO
    PORT(
         datain : IN  std_logic_vector(7 downto 0);
         dataout : OUT  std_logic_vector(7 downto 0);
         Clk : IN  std_logic; En : IN  std_logic; RW : IN  std_logic;
         Empty : OUT  std_logic; Full : OUT  std_logic;
         pointer: out integer range 0 to 255 ;
         add : integer range 0 to 255
        );
    END COMPONENT;

   --Inputs
   signal datain : std_logic_vector(7 downto 0) := (others => '0');
   signal Clk : std_logic := '0';
   signal En : std_logic := '0';
   signal RW : std_logic := '0';
   signal add : integer range 0 to 255 := 0 ;

   --Outputs
   signal dataout : std_logic_vector(7 downto 0);
   signal Empty : std_logic;
   signal Full : std_logic;
   signal pointer: integer range 0 to 255;

   -- Clock period definitions
   constant Clk_period : time := 10 ns;
```



```vhdl

BEGIN

   -- Instantiate the Unit Under Test (UUT)
   uut: LIFO PORT MAP (
          datain => datain,
          dataout => dataout,
          Clk => Clk,
          En => En,
          RW => RW,
          Empty => Empty,
          Full => Full,
          pointer => pointer,
          add => add
        );

  clk <= not clk after CLK_period/2;
  datain <= X"09" after 10 ns, X"20" after 20 ns, X"0F" after 30 ns, X"F2" after 40 ns,
  X"4F" after 50 ns, X"7E" after 60 ns ;
  En <= '1';
  RW <= '1' after 8 ns, '0' after 70 ns;
  add <= 5;
END;
```

Simulation result:



Stack Pointer	Address	Data
6	X “00”	126
5	X “01”	79
4	X “02”	242
3	X “03”	15
2	X “04”	32
1	X “05”	9