## 1. Problem/Objective

State the problem statement and/or objective of the lab. This must be a complete paragraph (i.e., at least 5 sentences).

The objective of this lab was to use the PicoBlaze assembly to implement a square program that uses an I/O interface. We were also given a portion of code to finish and implement on the Digilent S3 board.

## 2. Methodology

You are to search on the web (i.e., CSUF library, Google Scholar, etc.) to find a scholarly paper (i.e., 1 paper) on an application of a System-on-Chip (SoC). Briefly describe (about 2 complete paragraphs) what the paper is about.

The paper chosen was "Three-Dimensional Integrated Circuits and the Future of System-on-Chip Designs" by R.S. Patti. He talks about 3D integrated circuits and how they offer significant improvements over two-dimensional circuits. The paper discusses how the evolution of the integrated circuit (IC) has begun to slow. In the past, technical difficulties presented real but surmountable barriers and that now we are approaching a domain where physics forbids smaller gate technologies.

He goes into detail about the different construction methods for 3D ICs. The methods include chip stacking, transistor stacking, die-on-wafer stacking and wafer-level stacking. Chip stacking is when fully processed and tested standalone components are stacked to produce a system-in-package. Transistor stacking is the creation of multiple levels of transistors on a single substrate and then so on with the other two methods.

## 3. Question(s)

a) What is a SoC?

A SoC is a system on chip which is an integrated circuit that integrates all components of a computer.

b) What are the benefits of using a SoC versus a traditional microcontroller?

SoC are basically microcontrollers with small FPGA on the same chip, rather than having built in peripherals, you can implement whatever is needed within the available resources of the FPGA.

## 4. Program Code

Copy your code here. Please provide comments in your code. This will help me analyze your code and remove any ambiguity. **Provide your code as text, not as a screenshot/image**.

**Top Level:**

```
library ieee;
use ieee.std_logic_1164.all;

entity top_level is
    port (
            clk, reset : in std_logic;
```

```
            sw : in std_logic_vector (7 downto 0);
            btn : in std_logic_vector(1 downto 0);
            an : out std_logic_vector(3 downto 0);
            sseg : out std_logic_vector (7 downto 0)
    );
end top_level;

architecture behavioral of top_level is
    ----------------------------------------------------------------------
    -- TODO: i/o modules
    ----------------------------------------------------------------------
    -- debounce and disp_mux instantiaion
    component disp_mux
            port (
            clk, reset : in std_logic;
            in3, in2, in1, in0 : in std_logic_vector(7 downto 0);
            an : out std_logic_vector(3 downto 0);
            sseg : out std_logic_vector(7 downto 0));
    end component;

    component debounce
            port (
            clk, reset : in std_logic;
            sw : in std_logic;
            db_level, db_tick : out std_logic);
    end component;

    ----------------------------------------------------------------------
    -- kcpsm and rom instantiation
    ----------------------------------------------------------------------
    component kcpsm3
            port (
                    address : out std_logic_vector(9 downto 0);
                    instruction : in std_logic_vector(17 downto 0);
                    port_id : out std_logic_vector(7 downto 0);
                    write_strobe : out std_logic;
                    out_port : out std_logic_vector(7 downto 0);
                    read_strobe : out std_logic;
                    in_port : in std_logic_vector(7 downto 0);
                    interrupt : in std_logic;
                    interrupt_ack : out std_logic;
                    reset : in std_logic;
                    clk : in std_logic
            );
    end component;

    component rom
            port (
```

```
                address : in std_logic_vector(9 downto 0);
                instruction : out std_logic_vector(17 downto 0);
                clk : in std_logic
        );
    end component;

    ----------------------------------------------------------------------
    -- kcpsm3/rom signals
    ----------------------------------------------------------------------
    signal address : std_logic_vector(9 downto 0);
    signal instruction : std_logic_vector(17 downto 0);
    signal port_id : std_logic_vector(7 downto 0);
    signal in_port, out_port : std_logic_vector(7 downto 0);
    signal write_strobe, read_strobe : std_logic;
    signal interrupt, interrupt_ack : std_logic;
    signal kcpsm_reset : std_logic;

    ----------------------------------------------------------------------
    -- TODO: i/o port signals
    ----------------------------------------------------------------------
    --output enable
    signal en_d: std_logic_vector(3 downto 0);
    --four digit seven-segment display
    signal ds3_reg, ds2_reg: std_logic_vector(7 downto 0);
    signal ds1_reg, ds0_reg: std_logic_vector(7 downto 0);
    --push buttons
    signal btnc_flag_reg, btnc_flag_next: std_logic;
    signal btns_flag_reg, btns_flag_next: std_logic;
    signal set_btnc_flag, set_btns_flag: std_logic;
    signal clr_btn_flag: std_logic;

    ----------------------------------------------------------------------
    -- start of circuit description
    ----------------------------------------------------------------------
begin
    ----------------------------------------------------------------------
    -- TODO: I/O modules
    ----------------------------------------------------------------------
    disp_unit: disp_mux
        port map(
                clk=>clk,
                reset=>'0',
                in3=>ds3_reg,
                in2=>ds2_reg,
                in1=>ds1_reg,
                in0=>ds0_reg,
                an=>an,
                sseg=>sseg);
```

```
btnc_db_unit: debounce
        port map(
                clk=>clk,
                reset=>reset,
                sw=>btn(0),
                db_level=>open,
                db_tick=>set_btnc_flag);

btns_db_unit: debounce
        port map(
                clk=>clk,
                reset=>reset,
                sw=>btn(1),
                db_level=>open,
                db_tick=>set_btns_flag);

----------------------------------------------------------------------
-- KCPSM and ROM instantiation
----------------------------------------------------------------------
processor : kcpsm3
port map(
        address => address,
        instruction => instruction,
        port_id => port_id,
        write_strobe => write_strobe,
        out_port => out_port,
        read_strobe => read_strobe,
        in_port => in_port,
        interrupt => interrupt,
        interrupt_ack => interrupt_ack,
        reset => reset,
        clk => clk
);

program : rom
port map(
        address => address,
        instruction => instruction,
        clk => clk
);

-- unused inputs on processor
kcpsm_reset <= '0';
interrupt <= '0';

----------------------------------------------------------------------
-- TODO: output interface
```

```
----------------------------------------------------------------------
-- outport port id:
-- 0x00: ds0
-- 0x01: ds1
-- 0x02: ds2
-- 0x03: ds3
----------------------------------------------------------------------
output : process (clk)
begin
	if(rising_edge(clk)) then
		if en_d(0)='1' then
			ds0_reg <= out_port;
		end if;
		if en_d(1)='1' then
			ds1_reg <= out_port;
		end if;
		if en_d(2)='1' then
			ds2_reg <= out_port;
		end if;
		if en_d(3)='1' then
			ds3_reg <= out_port;
		end if;
	end if;
end process;

----------------------------------------------------------------------
-- TODO: decoding circuit for enable signals
----------------------------------------------------------------------
decoding : process (port_id, write_strobe)
begin
	if write_strobe='0' then
		en_d <= "0000";
	else
		case port_id(1 downto 0) is
			when "00" =>
				en_d <= "0001";
			when "01" =>
				en_d <= "0010";
			when "10" =>
				en_d <= "0100";
			when others =>
				en_d <= "1000";
		end case;
	end if;
end process;

----------------------------------------------------------------------
-- TODO: input interface
```

```
	-----------------------------------------------------------------------
	-- input port id:
	-- 0x00: flag
	-- 0x01: switch
	-----------------------------------------------------------------------
	input : process (clk)
	begin
		if (rising_edge(clk)) then
			btnc_flag_reg <= btnc_flag_next;
			btns_flag_reg <= btns_flag_next;
		end if;
	end process;

	btnc_flag_next <= '1' when set_btnc_flag='1' else
					'0' when clr_btn_flag='1' else
					btnc_flag_reg;
	btns_flag_next <= '1' when set_btns_flag='1' else
					'0' when clr_btn_flag='1' else
					btns_flag_reg;

	-- decoding circuit for clear signal
	clr_btn_flag <= '1' when read_strobe='1' and port_id(0)='0' else '0';
	-----------------------------------------------------------------------
	-- TODO: input multiplexing
	-----------------------------------------------------------------------
	in_mux : process (port_id, btns_flag_reg, btnc_flag_reg, sw)
	begin
		case port_id(0) is
			when '0' =>
				in_port <= "000000" & btns_flag_reg & btnc_flag_reg;
			when others =>
				in_port <= sw;
		end case;
	end process;

end behavioral;
```

**Top Level UCF:**

```
# Switches
NET "sw<0>" LOC = "F12";
NET "sw<1>" LOC = "G12";
NET "sw<2>" LOC = "H14";
NET "sw<3>" LOC = "H13";
NET "sw<4>" LOC = "J14";
NET "sw<5>" LOC = "J13";
NET "sw<6>" LOC = "K14";
NET "sw<7>" LOC = "K13";
```

```
# Buttons
NET "btn<0>" LOC = "M13";
NET "btn<1>" LOC = "M14";

# 7-Segment
NET "an<0>" LOC = "D14";
NET "an<1>" LOC = "G14";
NET "an<2>" LOC = "F14";
NET "an<3>" LOC = "E13";

NET "sseg<0>" LOC = "E14";
NET "sseg<1>" LOC = "G13";
NET "sseg<2>" LOC = "N15";
NET "sseg<3>" LOC = "P15";
NET "sseg<4>" LOC = "R16";
NET "sseg<5>" LOC = "F13";
NET "sseg<6>" LOC = "N16";
NET "sseg<7>" LOC = "P16";
```

**Debounce:**

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity debounce is
    port (
        clk, reset : in std_logic;
        sw : in std_logic;
        db_level, db_tick : out std_logic
    );
end debounce;

architecture behavioral of debounce is

    constant N : integer := 21; -- filter of 2'N * 20ns = 40ms
    type state_type is (zero, wait0, one, wait1);
    signal state_reg, state_next : state_type;
    signal q_reg, q_next : unsigned(N - 1 downto 0);
    signal q_load, q_dec, q_zero : std_logic;

begin
    -- FSMD state & data registers
    process (clk, reset)
    begin
        if reset = '1' then
            state_reg <= zero;
            q_reg <= (others => '0');
        elsif (clk'EVENT and clk = '1') then
```

```
                state_reg <= state_next;
                q_reg <= q_next;
        end if;
end process;

-- FSMD data path (counter) next - state logic
q_next <= (others => '1') when q_load = '1' else
      q_reg - 1 when q_dec = '1' else
      q_reg;
q_zero <= '1' when q_next = 0 else '0';

-- FSMD control path next-state logic
process (state_reg, sw, q_zero)
begin
        q_load <= '0';
        q_dec <= '0';
        db_tick <= '0';
        state_next <= state_reg;
        case state_reg is
                when zero =>
                        db_level <= '0';
                        if (sw = '1') then
                                state_next <= wait1;
                                q_load <= '1';
                        end if;
                when wait1 =>
                        db_level <= '0';
                        if (sw = '1') then
                                q_dec <= '1';
                                if (q_zero = '1') then
                                        state_next <= one;
                                        db_tick <= '1';
                                end if;
                        else -- sw= '0'
                                state_next <= zero;
                        end if;
                when one =>
                        db_level <= '1';
                        if (sw = '0') then
                                state_next <= wait0;
                                q_load <= '1';
                        end if;
                when wait0 =>
                        db_level <= '1';
                        if (sw = '0') then
                                q_dec <= '1';
                                if (q_zero = '1') then
                                        state_next <= zero;
```

```
                    end if;
                else -- sw = '1'
                    state_next <= one;
                end if;
            end case;
        end process;
end behavioral;
```

**Disp_Mux:**

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity disp_mux is
    port (
        clk, reset : in std_logic;
        in3, in2, in1, in0 : in std_logic_vector(7 downto 0);
        an : out std_logic_vector(3 downto 0);
        sseg : out std_logic_vector(7 downto 0)
    );
end disp_mux;

architecture behavioral of disp_mux is
    ----------------------------------------------------------------------
    -- TODO: display MUX
    ----------------------------------------------------------------------
    --refresh rate of 800 Hz (50MHz/2^16)
        constant N: integer := 18;
        signal q_reg, q_next: unsigned(N-1 downto 0);
        signal sel: std_logic_vector(1 downto 0);
    begin
        --register
        process(clk, reset)
        begin
                if reset='1' then
                    q_reg <= (others=>'0');
                elsif (rising_edge(clk)) then
                    q_reg <= q_next;
                end if;
        end process;

    --next state logic for counter
    q_next <= q_reg + 1;

    --2 MSBs of counter to control 4-to-1 mux
    -- and generate active-low enable signal
    sel <= std_logic_vector(q_reg(N-1 downto N-2));
```

```
	process(sel, in0, in1, in2, in3)
	begin
		case sel is
			when "00" =>
				--an <= "1110";
				an <= "0001"; --active high
				sseg <= in0;
			when "01" =>
				--an <= "1101";
				an <= "0010";
				sseg <= in1;
			when "10" =>
				--an <= "1011";
				an <= "0100";
				sseg <= in2;
			when others =>
				--an <= "0111";
				an <= "1000";
				sseg <= in3;
		end case;
	end process;

end behavioral;
```

**Rom.psm:**

```
;=====================================================
; Generate Instruction ROM
;=====================================================
VHDL        "ROM_form.vhd", "rom.vhd", "rom"

;=====================================================
; Data Constants
;=====================================================
UP_NIBBLE_MASK    equ    $0F       ;00001111

;=====================================================
; Data RAM Address Aliases
;=====================================================
a_lsb       equ   $00
b_lsb               equ    $02
aa_lsb              equ    $04
aa_msb              equ    $05
bb_lsb              equ    $06
bb_msb              equ    $07
aabb_lsb      equ    $08
aabb_msb    equ    $09
aabb_cout   equ    $0A
led0                equ    $10
```

```
led1                    equ     $11
led2                    equ     $12
led3                    equ     $13

;=====================================================
; Register Aliases
;=====================================================
;Local Variables
data        equ         s0
addr                    equ     s1
i                       equ     s2

;Global Variables
sw_in                   equ   sf
switch_a_b   equ     se

;=====================================================
; Port Aliases
;=====================================================
; Input Ports
rd_flag_port DSIN    $00
sw_port     DSIN   $01

; Output Ports
sseg0_port   DSOUT  $02
sseg1_port   DSOUT  $03
sseg2_port   DSOUT  $04
sseg3_port   DSOUT  $05

;=====================================================
; Main Program
;=====================================================
        call clr_data_mem
main_loop:
        call proc_btn
                call read_switch
                call square
                call write_led
                jump main_loop

;=====================================================
; Routine: clr_data_mem
;   Function: clear data memory
;      Temp Register: data, i
;=====================================================
clr_data_mem:
      load  i,$40               ;Set i=64
      load  data,$00
```

```
clr_mem_loop:
      store data,i
      sub   i,$01                                   ;Decremenet loop index
      jump  nz,clr_mem_loop                 ;Repeat until i=o
      store data,i
      load  switch_a_b,$00
      ret

;=====================================================
; Routine: proc_btn
;   Function: check two buttons and process the display
;      Input Reg:
;      switch_a_b: ram offset (0 for a and 2 for b)
;      Output Register:
;      s3: store input port flag
;      switch_a_b: may be toggled
;   Temp Register: data, addr
;=====================================================
proc_btn:
    in   s3,rd_flag_port

  ;check and process c button
    test s3,$01
    jump z,chk_btns
    call clr_data_mem
    jump proc_btn_done

;=====================================================
; Routine: chk_btns
;   Function: check s button
;      Temp Register: addr
;=====================================================
chk_btns:
    test  s3,$02
    jump  z,proc_btn_done

      in          data,sw_port
      load  addr,a_lsb
      add   addr,switch_a_b
      store data,addr

    xor   switch_a_b,$02

;=====================================================
; Routine: proc_btn_done
;   Function: return to main loop
;=====================================================
proc_btn_done:
```

```
    ret

;=====================================================
; Routine: read_switch
;   Function: obtain two nibbles from input
;      Input Register: sw_in
;      Temp Register: data, addr
;=====================================================
read_switch:
      in    s6,sw_port
      sl0   s6
      comp  s6,$08
      jump  c,sw_ok
      load  s6,$08
sw_ok:
      ;byte 0, lower nibble
      load  addr,a_lsb
      add     addr,s6
      fetch   data,s6
      call    get_lower_nibble
      call    hex_to_led
      store   data,led0
      ;byte 0, upper nibble
      fetch   data,addr
      call    get_upper_nibble
      call    hex_to_led
      store   data,led1
      ;byte 1, lower nibble
      add     addr,$01
      fetch   data,addr
      call    get_lower_nibble
      call    hex_to_led
      store   data,led2
      ;byte 1, upper nibble
      fetch   data,addr
      call    get_upper_nibble
      call    hex_to_led
      ;check sw="100"
      comp  s6,$08
      jump  nz,led_done
      add     addr,$01
      fetch   s6,addr
      test    s6,$01
      jump  z,led_done
      and     data,$7F
led_done:
      store   data,led3
      ret
```

```
;=====================================================
; Routine: write_led
;   Function: output 8 LSBs of result to 8 LEDs
;      Temp Register: data
;=====================================================
write_led:
      fetch data,led0
      out   data,sseg0_port
      fetch data,led1
      out   data,sseg1_port
      fetch data,led2
      out   data,sseg2_port
      fetch data,led3
      out   data,sseg3_port
      ret

;=====================================================
; Routine: get_lower_nibble
;   Function: get lower 4 bits of data
;      Input Register: data
;      Temp Register: data
;=====================================================
get_lower_nibble:
      and data,UP_NIBBLE_MASK     ;Clear upper nibble
      ret

;=====================================================
; Routine: get_upper_nibble
;   Function: get upper 4 bits of data
;      Input Register: data
;      Temp Register: data
;=====================================================
get_upper_nibble:
      sr0 data                ;Right shift 4 times
      sr0 data
      sr0 data
      sr0 data
      ret

;=====================================================
; Routine: square
;   Function: calculate a*a + b*b
;             data/result stored in RAM starting w/
;                           SQ_BASE_ADDR
;      Temp Register: s3,s4,s5,s6,data
;=====================================================
square:
```

```
        ;Calculate a*a
        fetch s3,a_lsb
        fetch s4,a_lsb
        call mult_soft
        store s6,aa_lsb
        store s5,aa_msb

        ;Calculate b*b
        fetch s3,b_lsb
        fetch s4,b_lsb
        call mult_soft
        store s6,bb_lsb
        store s5,bb_msb

        ;Calculate a*a + b*b
        fetch data,aa_lsb
        add   data,s6
        store data,aabb_lsb
        fetch data,aa_msb
        addc  data,s5
        store data,aabb_msb

        ;Get carry out
        load  data,$00
        addc  data,$00
        store data,aabb_cout
        ret

;=======================================================
;routine: mult_soft
;  function: 8-bit unsigned multiplier using
;          shift-and-add algorithm
;  input register:
;     s3: multiplicand
;     s4: multiplier
;  output register:
;     s5: upper byte of product
;     s6: lower byte of product
;  temp register: i
;=======================================================
mult_soft:
    load  s5,$00            ;clear s5
    load  s6,$00                                ;clear s6
    load  i,$08             ;initialize loop index
mult_loop:
    sr0   s4                ;shift lsb to carry
    jump  nc,shift_prod      ;lsb is 0
    add   s5,s3             ;lsb is 1
```

```
shift_prod:
   sra   s5              ;shift upper byte right,
                         ;carry to MSB, LSB to carry
   sra   s6              ;shift lower byte right,
                         ;lsb of s5 to MSB of s6
   sub   i,$01           ;dec loop index
   jump  nz,mult_loop    ;repeat until i=0
   ret

;=====================================================
;routine: hex_to_led
;  function: convert a hex digit to 7-seg led pattern
;  input register: data
;  output register: data
;=====================================================
hex_to_led:
  comp    data,$00
  jump    nz,comp_hex_1
  load    data,$81      ;7seg pattern 0
  jump    hex_done
comp_hex_1:
  comp    data,$01
  jump    nz,comp_hex_2
  load    data,$CF      ;7seg pattern 1
  jump    hex_done
comp_hex_2:
  comp    data,$02
  jump    nz,comp_hex_3
  load    data,$92      ;7seg pattern 2
  jump    hex_done
comp_hex_3:
  comp    data,$03
  jump    nz,comp_hex_4
  load    data,$86      ;7seg pattern 3
  jump    hex_done
comp_hex_4:
  comp    data,$04
  jump    nz,comp_hex_5
  load    data,$CC      ;7seg pattern 4
  jump    hex_done
comp_hex_5:
  comp    data,$05
  jump    nz,comp_hex_6
  load    data,$A4      ;7seg pattern 5
  jump    hex_done
comp_hex_6:
  comp    data,$06
  jump    nz,comp_hex_7
```

```
  load    data,$A0        ;7seg pattern 6
  jump    hex_done
comp_hex_7:
  comp    data,$07
  jump    nz,comp_hex_8
  load    data,$8F        ;7seg pattern 7
  jump    hex_done
comp_hex_8:
  comp    data,$08
  jump    nz,comp_hex_9
  load    data,$80        ;7seg pattern 8
  jump    hex_done
comp_hex_9:
  comp    data,$09
  jump    nz,comp_hex_a
  load    data,$84        ;7seg pattern 9
  jump    hex_done
comp_hex_a:
  comp    data,$0A
  jump    nz,comp_hex_b
  load    data,$88        ;7seg pattern a
  jump    hex_done
comp_hex_b:
  comp    data,$0B
  jump    nz,comp_hex_c
  load    data,$E0        ;7seg pattern b
  jump    hex_done
comp_hex_c:
  comp    data,$0C
  jump    nz,comp_hex_d
  load    data,$B1        ;7seg pattern C
  jump    hex_done
comp_hex_d:
  comp    data,$0D
  jump    nz,comp_hex_e
  load    data,$C2        ;7seg pattern d
  jump    hex_done
comp_hex_e:
  comp    data,$0E
  jump    nz,comp_hex_f
  load    data,$B0        ;7seg pattern E
  jump    hex_done
comp_hex_f:
  load    data,$B8        ;7seg pattern f
hex_done:
  ret
```

**Rom.vhd:**

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
--
-- The Unisim Library is used to define Xilinx primitives. It is also used during
-- simulation. The source can be viewed at %XILINX%\vhdl\src\unisims\unisim_VCOMP.vhd
--
library unisim;
use unisim.vcomponents.all;
--
--
entity rom is
   Port (      address : in std_logic_vector(9 downto 0);
         instruction : out std_logic_vector(17 downto 0);
               clk : in std_logic);
   end rom;
--
architecture low_level_definition of rom is
--
-- Attributes to define ROM contents during implementation synthesis.
-- The information is repeated in the generic map for functional simulation
--
attribute INIT_00 : string;
attribute INIT_01 : string;
attribute INIT_02 : string;
attribute INIT_03 : string;
attribute INIT_04 : string;
attribute INIT_05 : string;
attribute INIT_06 : string;
attribute INIT_07 : string;
attribute INIT_08 : string;
attribute INIT_09 : string;
attribute INIT_0A : string;
attribute INIT_0B : string;
attribute INIT_0C : string;
attribute INIT_0D : string;
attribute INIT_0E : string;
attribute INIT_0F : string;
attribute INIT_10 : string;
attribute INIT_11 : string;
attribute INIT_12 : string;
attribute INIT_13 : string;
attribute INIT_14 : string;
attribute INIT_15 : string;
attribute INIT_16 : string;
attribute INIT_17 : string;
```

attribute INIT_18 : string;
attribute INIT_19 : string;
attribute INIT_1A : string;
attribute INIT_1B : string;
attribute INIT_1C : string;
attribute INIT_1D : string;
attribute INIT_1E : string;
attribute INIT_1F : string;
attribute INIT_20 : string;
attribute INIT_21 : string;
attribute INIT_22 : string;
attribute INIT_23 : string;
attribute INIT_24 : string;
attribute INIT_25 : string;
attribute INIT_26 : string;
attribute INIT_27 : string;
attribute INIT_28 : string;
attribute INIT_29 : string;
attribute INIT_2A : string;
attribute INIT_2B : string;
attribute INIT_2C : string;
attribute INIT_2D : string;
attribute INIT_2E : string;
attribute INIT_2F : string;
attribute INIT_30 : string;
attribute INIT_31 : string;
attribute INIT_32 : string;
attribute INIT_33 : string;
attribute INIT_34 : string;
attribute INIT_35 : string;
attribute INIT_36 : string;
attribute INIT_37 : string;
attribute INIT_38 : string;
attribute INIT_39 : string;
attribute INIT_3A : string;
attribute INIT_3B : string;
attribute INIT_3C : string;
attribute INIT_3D : string;
attribute INIT_3E : string;
attribute INIT_3F : string;
attribute INITP_00 : string;
attribute INITP_01 : string;
attribute INITP_02 : string;
attribute INITP_03 : string;
attribute INITP_04 : string;
attribute INITP_05 : string;
attribute INITP_06 : string;
attribute INITP_07 : string;

```
    --
    -- Attributes to define ROM contents during implementation synthesis.
    --
    attribute INIT_00 of ram_1024_x_18 : label is
"23014300A0000E00F0205408C201F020000002404001003B004B001B000E0006";
    attribute INIT_01 of ram_1024_x_18 : label is
"06085820460806064601A000EE02F01091E001004001501A2302401A00065013";
    attribute INIT_02 of ram_1024_x_18 : label is
"7010E012006A004470108101E011006A00467010E010006A0044706091600100";
    attribute INIT_03 of ram_1024_x_18 : label is
"6012C0036011C0026010A000E013A07F503926017610810154394608006A0046";
    attribute INIT_04 of ram_1024_x_18 : label is
"E505E604005F64006300A000000E000E000E000EA000A00FA000C0056013C004";
    attribute INIT_05 of ram_1024_x_18 : label is
"0500A000E00AA0000000E009B0506005E00890606004E507E606005F64026302";
    attribute INIT_06 of ram_1024_x_18 : label is
"5472400140A70081546E4000A0005462C2010608050895305C65040E02080600";
    attribute INIT_07 of ram_1024_x_18 : label is
"5482400540A700CC547E400440A70086547A400340A700925476400240A700CF";
    attribute INIT_08 of ram_1024_x_18 : label is
"5492400940A70080548E400840A7008F548A400740A700A05486400640A700A4";
    attribute INIT_09 of ram_1024_x_18 : label is
"54A2400D40A700B1549E400C40A700E0549A400B40A700885496400A40A70084";
    attribute INIT_0A of ram_1024_x_18 : label is
"0000000000000000000000000000000A00000B840A700B054A6400E40A700C2";
    attribute INIT_0B of ram_1024_x_18 : label is
"0000000000000000000000000000000000000000000000000000000000000000";
    attribute INIT_0C of ram_1024_x_18 : label is
"0000000000000000000000000000000000000000000000000000000000000000";
    attribute INIT_0D of ram_1024_x_18 : label is
"0000000000000000000000000000000000000000000000000000000000000000";
    attribute INIT_0E of ram_1024_x_18 : label is
"0000000000000000000000000000000000000000000000000000000000000000";
    attribute INIT_0F of ram_1024_x_18 : label is
"0000000000000000000000000000000000000000000000000000000000000000";
    attribute INIT_10 of ram_1024_x_18 : label is
"0000000000000000000000000000000000000000000000000000000000000000";
    attribute INIT_11 of ram_1024_x_18 : label is
"0000000000000000000000000000000000000000000000000000000000000000";
    attribute INIT_12 of ram_1024_x_18 : label is
"0000000000000000000000000000000000000000000000000000000000000000";
    attribute INIT_13 of ram_1024_x_18 : label is
"0000000000000000000000000000000000000000000000000000000000000000";
    attribute INIT_14 of ram_1024_x_18 : label is
"0000000000000000000000000000000000000000000000000000000000000000";
    attribute INIT_15 of ram_1024_x_18 : label is
"0000000000000000000000000000000000000000000000000000000000000000";
```

```
    attribute INIT_16 of ram_1024_x_18 : label is
"000000000000000000000000000000000000000000000000000000000000000000";
    attribute INIT_17 of ram_1024_x_18 : label is
"000000000000000000000000000000000000000000000000000000000000000000";
    attribute INIT_18 of ram_1024_x_18 : label is
"000000000000000000000000000000000000000000000000000000000000000000";
    attribute INIT_19 of ram_1024_x_18 : label is
"000000000000000000000000000000000000000000000000000000000000000000";
    attribute INIT_1A of ram_1024_x_18 : label is
"000000000000000000000000000000000000000000000000000000000000000000";
    attribute INIT_1B of ram_1024_x_18 : label is
"000000000000000000000000000000000000000000000000000000000000000000";
    attribute INIT_1C of ram_1024_x_18 : label is
"000000000000000000000000000000000000000000000000000000000000000000";
    attribute INIT_1D of ram_1024_x_18 : label is
"000000000000000000000000000000000000000000000000000000000000000000";
    attribute INIT_1E of ram_1024_x_18 : label is
"000000000000000000000000000000000000000000000000000000000000000000";
    attribute INIT_1F of ram_1024_x_18 : label is
"000000000000000000000000000000000000000000000000000000000000000000";
    attribute INIT_20 of ram_1024_x_18 : label is
"000000000000000000000000000000000000000000000000000000000000000000";
    attribute INIT_21 of ram_1024_x_18 : label is
"000000000000000000000000000000000000000000000000000000000000000000";
    attribute INIT_22 of ram_1024_x_18 : label is
"000000000000000000000000000000000000000000000000000000000000000000";
    attribute INIT_23 of ram_1024_x_18 : label is
"000000000000000000000000000000000000000000000000000000000000000000";
    attribute INIT_24 of ram_1024_x_18 : label is
"000000000000000000000000000000000000000000000000000000000000000000";
    attribute INIT_25 of ram_1024_x_18 : label is
"000000000000000000000000000000000000000000000000000000000000000000";
    attribute INIT_26 of ram_1024_x_18 : label is
"000000000000000000000000000000000000000000000000000000000000000000";
    attribute INIT_27 of ram_1024_x_18 : label is
"000000000000000000000000000000000000000000000000000000000000000000";
    attribute INIT_28 of ram_1024_x_18 : label is
"000000000000000000000000000000000000000000000000000000000000000000";
    attribute INIT_29 of ram_1024_x_18 : label is
"000000000000000000000000000000000000000000000000000000000000000000";
    attribute INIT_2A of ram_1024_x_18 : label is
"000000000000000000000000000000000000000000000000000000000000000000";
    attribute INIT_2B of ram_1024_x_18 : label is
"000000000000000000000000000000000000000000000000000000000000000000";
    attribute INIT_2C of ram_1024_x_18 : label is
"000000000000000000000000000000000000000000000000000000000000000000";
    attribute INIT_2D of ram_1024_x_18 : label is
"000000000000000000000000000000000000000000000000000000000000000000";
```

```
    attribute INIT_2E of ram_1024_x_18 : label is
"0000000000000000000000000000000000000000000000000000000000000000";
    attribute INIT_2F of ram_1024_x_18 : label is
"0000000000000000000000000000000000000000000000000000000000000000";
    attribute INIT_30 of ram_1024_x_18 : label is
"0000000000000000000000000000000000000000000000000000000000000000";
    attribute INIT_31 of ram_1024_x_18 : label is
"0000000000000000000000000000000000000000000000000000000000000000";
    attribute INIT_32 of ram_1024_x_18 : label is
"0000000000000000000000000000000000000000000000000000000000000000";
    attribute INIT_33 of ram_1024_x_18 : label is
"0000000000000000000000000000000000000000000000000000000000000000";
    attribute INIT_34 of ram_1024_x_18 : label is
"0000000000000000000000000000000000000000000000000000000000000000";
    attribute INIT_35 of ram_1024_x_18 : label is
"0000000000000000000000000000000000000000000000000000000000000000";
    attribute INIT_36 of ram_1024_x_18 : label is
"0000000000000000000000000000000000000000000000000000000000000000";
    attribute INIT_37 of ram_1024_x_18 : label is
"0000000000000000000000000000000000000000000000000000000000000000";
    attribute INIT_38 of ram_1024_x_18 : label is
"0000000000000000000000000000000000000000000000000000000000000000";
    attribute INIT_39 of ram_1024_x_18 : label is
"0000000000000000000000000000000000000000000000000000000000000000";
    attribute INIT_3A of ram_1024_x_18 : label is
"0000000000000000000000000000000000000000000000000000000000000000";
    attribute INIT_3B of ram_1024_x_18 : label is
"0000000000000000000000000000000000000000000000000000000000000000";
    attribute INIT_3C of ram_1024_x_18 : label is
"0000000000000000000000000000000000000000000000000000000000000000";
    attribute INIT_3D of ram_1024_x_18 : label is
"0000000000000000000000000000000000000000000000000000000000000000";
    attribute INIT_3E of ram_1024_x_18 : label is
"0000000000000000000000000000000000000000000000000000000000000000";
    attribute INIT_3F of ram_1024_x_18 : label is
"0000000000000000000000000000000000000000000000000000000000000000";
    attribute INITP_00 of ram_1024_x_18 : label is
"DCDCDCDCDCDB69E0292492B0AC2AA8A22228D1DF2F1BCBC43622437F48B60FFF";
    attribute INITP_01 of ram_1024_x_18 : label is
"0000000000000000000000000000000000000000008CDCDCDCDCDCDCDCDCDC";
    attribute INITP_02 of ram_1024_x_18 : label is
"0000000000000000000000000000000000000000000000000000000000000000";
    attribute INITP_03 of ram_1024_x_18 : label is
"0000000000000000000000000000000000000000000000000000000000000000";
    attribute INITP_04 of ram_1024_x_18 : label is
"0000000000000000000000000000000000000000000000000000000000000000";
    attribute INITP_05 of ram_1024_x_18 : label is
"0000000000000000000000000000000000000000000000000000000000000000";
```

```
    attribute INITP_06 of ram_1024_x_18 : label is
"0000000000000000000000000000000000000000000000000000000000000000";
    attribute INITP_07 of ram_1024_x_18 : label is
"0000000000000000000000000000000000000000000000000000000000000000";
    --
    begin
    --
     --Instantiate the Xilinx primitive for a block RAM
     ram_1024_x_18: RAMB16_S18
     --synthesis translate_off
     --INIT values repeated to define contents for functional simulation
     generic map ( INIT_00 =>
X"23014300A0000E00F0205408C201F020000002404001003B004B001B000E0006",
                INIT_01 =>
X"0608582046080606 4601A000EE02F01091E001004001501A2302401A00065013",
                INIT_02 =>
X"7010E012006A004470108101E011006A00467010E010006A0044706091600100",
                INIT_03 =>
X"6012C0036011C0026010A000E013A07F503926017610810154394608006A0046",
                INIT_04 =>
X"E505E604005F64006300A000000E000E000E000EA000A00FA000C0056013C004",
                INIT_05 =>
X"0500A000E00AA0000000E009B0506005E00890606004E507E606005F64026302",
                INIT_06 =>
X"5472400140A70081546E4000A0005462C2010608050895305C65040E02080600",
                INIT_07 =>
X"5482400540A700CC547E400440A70086547A400340A700925476400240A700CF",
                INIT_08 =>
X"5492400940A70080548E400840A7008F548A400740A700A05486400640A700A4",
                INIT_09 =>
X"54A2400D40A700B1549E400C40A700E0549A400B40A700885496400A40A70084",
                INIT_0A =>
X"0000000000000000000000000000000A00000B840A700B054A6400E40A700C2",
                INIT_0B =>
X"0000000000000000000000000000000000000000000000000000000000000000",
                INIT_0C =>
X"0000000000000000000000000000000000000000000000000000000000000000",
                INIT_0D =>
X"0000000000000000000000000000000000000000000000000000000000000000",
                INIT_0E =>
X"0000000000000000000000000000000000000000000000000000000000000000",
                INIT_0F =>
X"0000000000000000000000000000000000000000000000000000000000000000",
                INIT_10 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
                INIT_11 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
```

```
            INIT_12                                                                 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
            INIT_13                                                                 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
            INIT_14                                                                 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
            INIT_15                                                                 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
            INIT_16                                                                 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
            INIT_17                                                                 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
            INIT_18                                                                 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
            INIT_19                                                                 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
            INIT_1A                                                                 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
            INIT_1B                                                                 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
            INIT_1C                                                                 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
            INIT_1D                                                                 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
            INIT_1E                                                                 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
            INIT_1F                                                                 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
            INIT_20                                                                 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
            INIT_21                                                                 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
            INIT_22                                                                 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
            INIT_23                                                                 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
            INIT_24                                                                 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
            INIT_25                                                                 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
            INIT_26                                                                 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
            INIT_27                                                                 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
            INIT_28                                                                 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
            INIT_29                                                                 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
```

```
            INIT_2A                                                                 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
            INIT_2B                                                                 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
            INIT_2C                                                                 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
            INIT_2D                                                                 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
            INIT_2E                                                                 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
            INIT_2F                                                                 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
            INIT_30                                                                 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
            INIT_31                                                                 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
            INIT_32                                                                 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
            INIT_33                                                                 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
            INIT_34                                                                 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
            INIT_35                                                                 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
            INIT_36                                                                 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
            INIT_37                                                                 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
            INIT_38                                                                 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
            INIT_39                                                                 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
            INIT_3A                                                                 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
            INIT_3B                                                                 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
            INIT_3C                                                                 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
            INIT_3D                                                                 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
            INIT_3E                                                                 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
            INIT_3F                                                                 =>
X"0000000000000000000000000000000000000000000000000000000000000000",
           INITP_00                                                                 =>
X"DCDCDCDCDCDB69E0292492B0AC2AA8A22228D1DF2F1BCBC43622437F48B60FFF",
           INITP_01                                                                 =>
X"0000000000000000000000000000000000000000008CDCDCDCDCDCDCDCDCDC",
```

```
                INITP_02                                                                                =>
X"0000000000000000000000000000000000000000000000000000000000000000",
                INITP_03                                                                                =>
X"0000000000000000000000000000000000000000000000000000000000000000",
                INITP_04                                                                                =>
X"0000000000000000000000000000000000000000000000000000000000000000",
                INITP_05                                                                                =>
X"0000000000000000000000000000000000000000000000000000000000000000",
                INITP_06                                                                                =>
X"0000000000000000000000000000000000000000000000000000000000000000",
                INITP_07                                                                                =>
X"0000000000000000000000000000000000000000000000000000000000000000")
        --synthesis translate_on
        port map(    DI => "0000000000000000",
                    DIP => "00",
                     EN => '1',
                     WE => '0',
                    SSR => '0',
                    CLK => clk,
                   ADDR => address,
                     DO => instruction(15 downto 0),
                    DOP => instruction(17 downto 16));
    --
    end low_level_definition;
    --
    ------------------------------------------------------------------------------
    --
    -- END OF FILE rom.vhd
    --
------------------------------------------------------------------------------
```