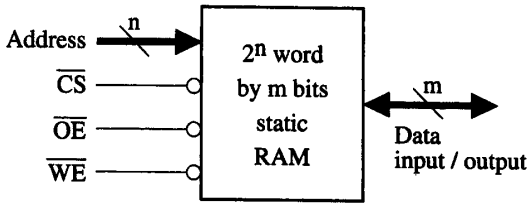
CPE/EE 422/522
Spring 2004
Chapter 9 - VHDL Models for Memories and Busses

Dr. Rhonda Kay Gaede

UAH

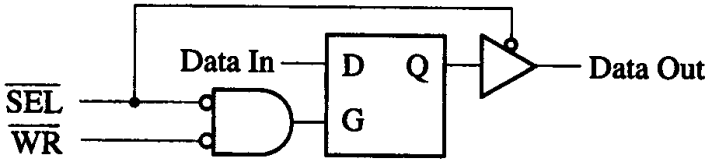
9.1 Static RAM Memory - Block Diagram

- This RAM has n address lines, m data lines, and three control lines.
  - —CS
  - —OE
  - —WE
- Unlike dynamic memory, which requires data refresh, static memory retains data until the power is turned off.



9.1 Static RAM Memory - Static Memory Cell

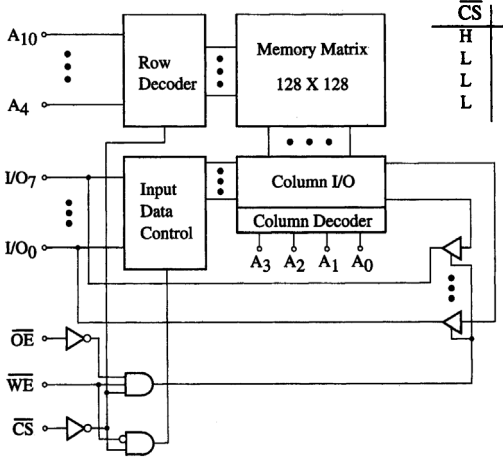
—RAMs contain address decoders and a memory array, each element of which is a memory cell.



G = 1 →Q follows D

G = 0 →data is latched

9.1 Static RAM Memory - Expanded Block Diagram



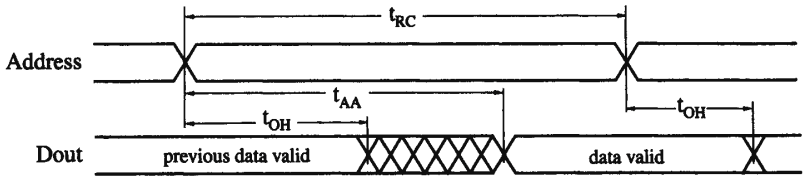
$\overline{CS}$	$\overline{OE}$	$\overline{WE}$	Mode	I/O pins
H	X	X	not selected	high-Z
L	H	H	output disabled	high-Z
L	L	H	read	data out
L	X	L	write	data in

9.1 Static RAM Memory - CMOS RAM Timing Specifications

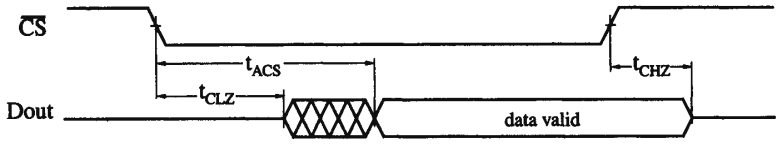
Parameter	Symbol	6116-2		43258A-25	
		min	*max*	*min*	*max*
Read cycle time	t_{RC}	120	—	25	—
Address access time	t_{AA}	—	120	—	25
Chip select access time	t_{ACS}	—	120	—	25
Chip selection to output in low-Z	t_{CLZ}	10	—	3	—
Output enable to output valid	t_{OE}	—	80	—	12
Output enable to output in low-Z	t_{OLZ}	10	—	0	—
Chip deselection to output in high-Z	t_{CHZ}	10*	40	3*	10
Chip disable to output in high-Z	t_{OHZ}	10*	40	3*	10
Output hold from address change	t_{OH}	10	—	3	—
Write cycle time	t_{WC}	120	—	25	—
Chip selection to end of write	t_{CW}	70	—	15	—
Address valid to end of write	t_{AW}	105	—	15	—
Address setup time	t_{AS}	0	—	0	—
Write pulse width	t_{WP}	70	—	15	—
Write recovery time	t_{WR}	0	—	0	—
Write enable to output in high-Z	t_{WHZ}	10*	35	3*	10
Data valid to end of write	t_{DW}	35	—	12	—
Data hold from end of write	t_{DH}	0	—	0	—
Output active from end of write	t_{OW}	10	—	0	—

**Estimated value, not specified by manufacturer.*

9.1 Static RAM Memory - Read Cycle Timing

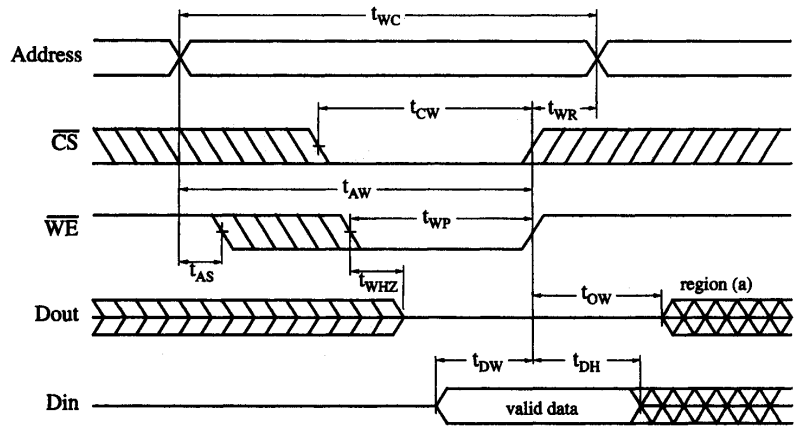


(a) With $\overline{CS}$ = 0, $\overline{OE}$ = 0, $\overline{WE}$ =1

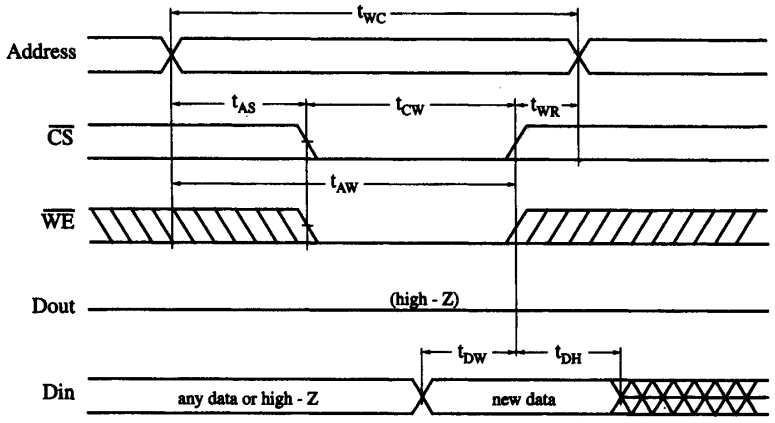


(b) With address stable, $\overline{OE}$ = 0, $\overline{WE}$ =1

9.1 Static RAM Memory - WE-controlled Write Cycle Timing



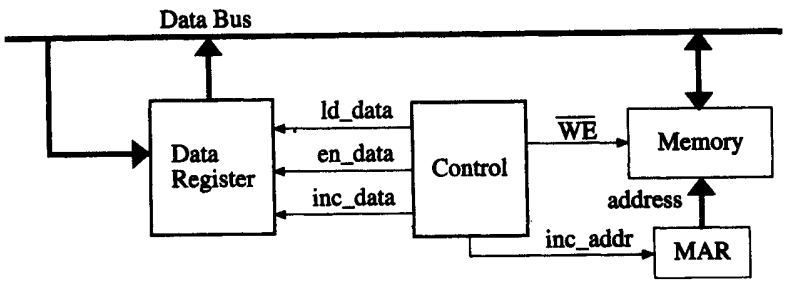
9.1 Static RAM Memory - CS-controlled Write Cycle Timing



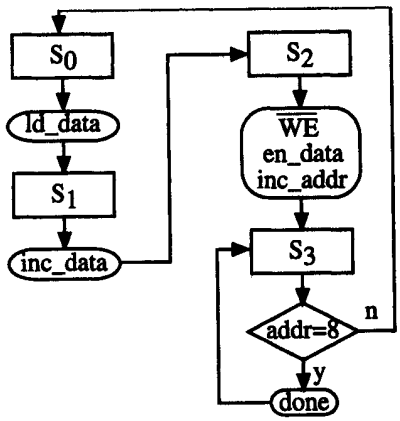
9.1 Static RAM Memory - Simple Memory Model

```
entity RAM6116 is
   port(Cs_b, We_b: in bit;
        Address: in bit_vector(7  downto 0);
        IO: inout std_logic_vector(7 downto 0));
end RAM6116;
architecture simple_ram of RAM6116 is
 type RAMtype is array(0 to 255) of std_logic_vector(7 downto 0);
 signal RAM1: RAMtype:=(others=>(others=>'0')); -- Set all bits to '0'
begin
   process
   begin
     if Cs_b = '1' then IO <= "ZZZZZZZZ";  -- chip not selected
     else
       if We_b'event and We_b = '1' then -- rising-edge of We_b
         RAM1(vec2int(Address'delayed )) <= IO;      -- write
         wait for 0 ns;            -- wait for RAM update
       end if;
       if We_b = '1' then IO <= RAM1(vec2int(Address));     --read
       else IO <= "ZZZZZZZZ";                       --drive high-Z
       end if;
     end if;
     wait on We_b, Cs_b, Address;
   end process;
end simple_ram;
```


9.1 Static RAM Memory - Block Diagram of RAM System



9.1 Static RAM Memory - SM Chart for RAM System



9.1 Static RAM Memory - Tester for Simple Memory Model

```
-- Tester for simple ram model
library ieee;
use ieee.std_logic_1164.all;

entity RAM6116_system is
end RAM6116_system;

architecture RAMtest of RAM6116_system is
   component RAM6116 is
         port(Cs_b, We_b: in bit;
                  Address: in bit_vector(7  downto 0);
                  IO: inout std_logic_vector(7 downto 0));
   end component RAM6116;
   signal state, next_state: integer range 0 to 3;
   signal inc_adrs, inc_data, ld_data, en_data, Cs_b, clk, done: bit;
   signal We_b: bit := '1';                -- initialize to read mode
   signal Data: bit_vector(7 downto 0);              -- data register
   signal Address: bit_vector(7 downto 0); -- address register
   signal IO: std_logic_vector(7 downto 0);         -- I/O bus
begin
  -- Concurrent statements
   clk <= not clk after 100 ns;
   IO <= To_StdLogicVector(data) when en_data = '1'
         else "ZZZZZZZZ";
```

9.1 Static RAM Memory - Tester for Simple Memory Model

```
    RAM1: RAM6116 port map(Cs_b, We_b, Address, IO);
    control: process(state, Address)
    begin
          --initialize all control signals (RAM always selected)
          ld_data<='0'; inc_data<='0'; inc_adrs<='0'; en_data <='0';
           done <= '0'; We_b <='1'; Cs_b <= '0';
          --start SM chart here
          case (state) is
            when 0 =>  ld_data <= '1';  next_state <= 1;
            when 1 =>  inc_data <= '1'; next_state <= 2;
            when 2 =>  We_b <= '0';  en_data <= '1';  inc_adrs <= '1'; next_state <= 3;
            when 3 =>  if (Address = "00001000") then done <= '1';
                       else next_state <= 0;
                       end if;
          end case;
    end process control;
          register_update: process
    begin
      wait until clk = '1';
      state <= next_state;
      if (inc_data = '1') then data <= int2vec(vec2int(data)+1,8); end if;
      if (ld_data = '1') then data <= To_bitvector(IO); end if;
      if (inc_adrs = '1') then
        Address <= int2vec(vec2int(Address)+1,8) after 1 ns;
                   -- delay added to allow completion of memory write
      end if;
    end process register_update;
end RAMtest;
```


9.1 Static RAM Memory - 6116 CMOS RAM VHDL Timing Model

```
-- memory model with timing (OE_b=0)
library ieee;
use ieee.std_logic_1164.all;

entity static_RAM is
generic (constant tAA: time := 120 ns;      -- 6116 static CMOS RAM
   constant tACS:time := 120 ns;
   constant tCLZ:time := 10 ns;
   constant tCHZ:time := 10 ns;
   constant tOH:time := 10 ns;
   constant tWC:time := 120 ns;
   constant tAW:time := 105 ns;
   constant tWP:time := 70 ns;
   constant tWHZ:time := 35 ns;
   constant tDW:time := 35 ns;
   constant tDH:time := 0 ns;
   constant tOW:time := 10 ns);

port (CS_b, WE_b, OE_b: in bit;
  Address: in bit_vector(7 downto 0);
  Data: inout std_logic_vector(7 downto 0) := (others => 'Z'));
end Static_RAM;
```

9.1 Static RAM Memory - 6116 CMOS RAM VHDL Timing Model

```
architecture SRAM of Static_RAM is
  type RAMtype is array(0 to 255) of bit_vector(7 downto 0);
  signal RAM1: RAMtype := (others => (others => '0'));
begin
  RAM: process
  begin
     if (rising_edge(WE_b) and CS_b'delayed = '0')   or (rising_edge(CS_b) and
    WE_b'delayed = '0') then
       RAM1(vec2int(Address'delayed)) <= to_bitvector(Data'delayed); --write
       if CS_b = '0' then
          Data <= transport Data'delayed after tOW;         -- read back after write
       end if;
     end if;
     if falling_edge(WE_b) and CS_b = '0' then    -- enter write mode
       Data <= transport "ZZZZZZZZ" after tWHZ;
     end if;
     if CS_b'event and OE_b = '0' then
       if CS_b = '1' then                                   -- RAM is deselected
         Data <= transport "ZZZZZZZZ" after tCHZ;
       elsif WE_b = '1' then                                --read
         Data <= "XXXXXXXX" after tCLZ;
         Data <= transport to_stdlogicvector(RAM1(vec2int(Address))) after tACS;
       end if;
     end if;
     if Address'event and CS_b ='0' and OE_b ='0' and WE_b ='1' then --read
       Data <= "XXXXXXXX" after tOH;
       Data <= transport to_stdlogicvector(RAM1(vec2int(Address))) after tAA;
     end if;
     wait on CS_b, WE_b, Address;
    end process RAM;
```


9.1 Static RAM Memory - 6116 CMOS RAM VHDL Timing Model

```
     check: process
     begin
       if NOW /= 0 ns then
         if address'event then
           assert (address'delayed'stable(tWC))    -- tRC = tWC assumed
             report "Address cycle time too short"
             severity WARNING;
         end if;
     -- The following code only checks for a WE_b controlled write:
         if rising_edge(WE_b) and CS_b'delayed = '0' then
           assert (address'delayed'stable(tAW))
             report "Address not valid long enough to end of write"
             severity WARNING;
           assert (WE_b'delayed'stable(tWP))
             report "Write pulse too short"
             severity WARNING;
           assert (Data'delayed'stable(tDW))
             report "Data setup time too short"
             severity WARNING;
           wait for tDH;
           assert (Data'last_event >= tDH)
             report "Data hold time too short"
             severity WARNING;
         end if;
       end if;
       wait on WE_b, address, CS_b;
     end process check;
  end SRAM;
```

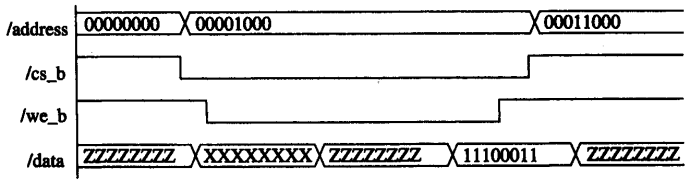
9.1 Static RAM Memory - Test for Timing Model

```
architecture test1 of RAM_timing_tester is
  component static_RAM is
   port (CS_b, WE_b, OE_b: in bit;
         Address: in bit_vector(7 downto 0);
         Data: inout std_logic_vector(7 downto 0));
  end component Static_RAM;
  signal Cs_b, We_b: bit := '1';           -- active low signals
  signal Data: std_logic_vector(7 downto 0) := "ZZZZZZZZ";
  signal Address: bit_vector(7 downto 0);
begin
   SRAM1: Static_RAM port map(Cs_b, We_b, '0', Address, Data);
   process
   begin
         wait for 100 ns;

         Address <= "00001000";                    -- WE-controlled write
         Cs_b <= '0';
         We_b <= transport '0' after 20 ns;
         Data <= transport "11100011" after 140 ns;
         Cs_b <= transport '1' after 200 ns;
         We_b <= transport '1' after 180 ns;
         Data <= transport "ZZZZZZZZ" after 220 ns;
         wait for 200 ns;
```

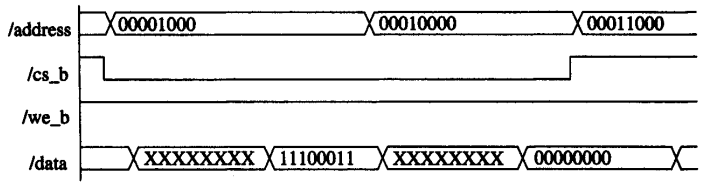

9.1 Static RAM Memory - Test for Timing Model

```
         Address <= "00011000";                    -- RAM deselected
         wait for 200 ns;

         Address <= "00001000";                    -- Read cycles
         Cs_b <= '0';
         wait for 200 ns;
         Address <= "00010000";
         Cs_b <= '1' after 200 ns;
         wait for 200 ns;

         Address <= "00011000";                    -- RAM deselected
         wait for 200 ns;
   end process;
end test1;
```

9.1 Static RAM Memory - Timing Test Results

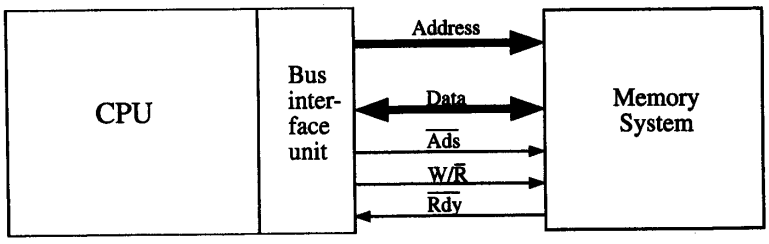


(a) Write cycle

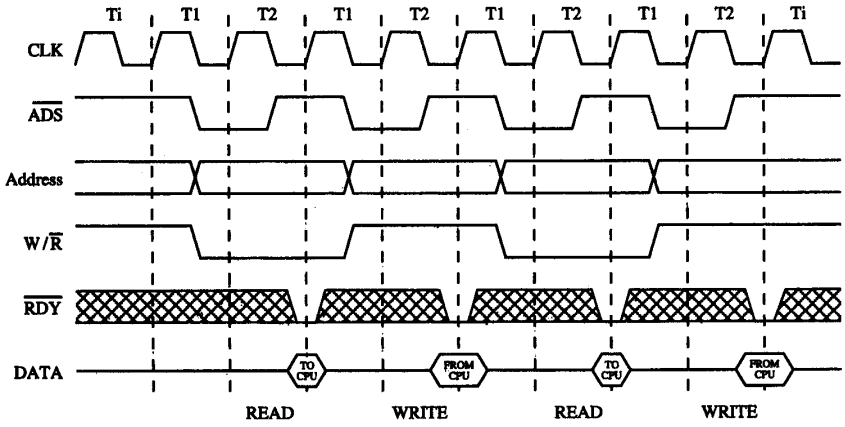


(b) Two read cycles

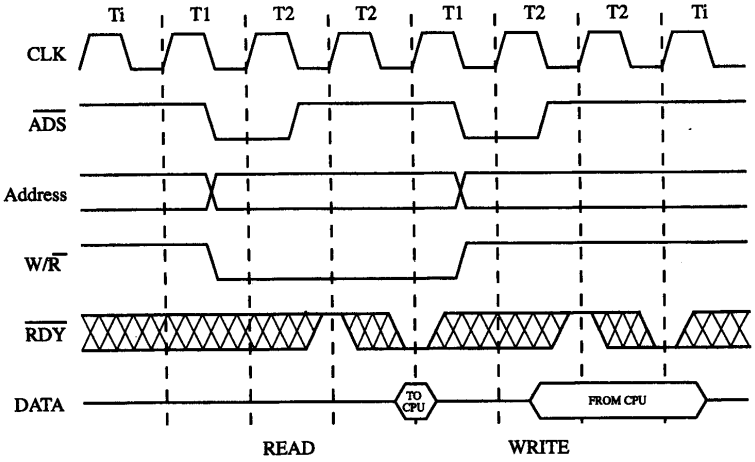
9.2 A Simplified 486 Bus Model - Block Diagram



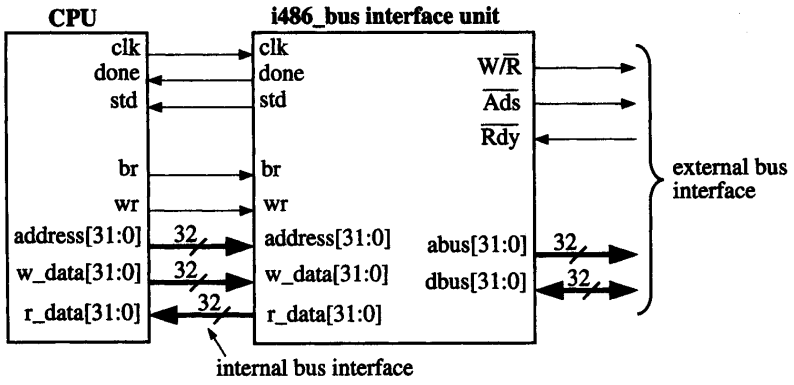
9.2 A Simplified 486 Bus Model - Basic 2-2 Bus Cycle



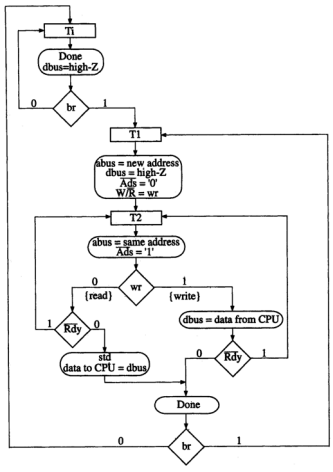
9.2 A Simplified 486 Bus Model - Basic 3-3 Bus Cycle



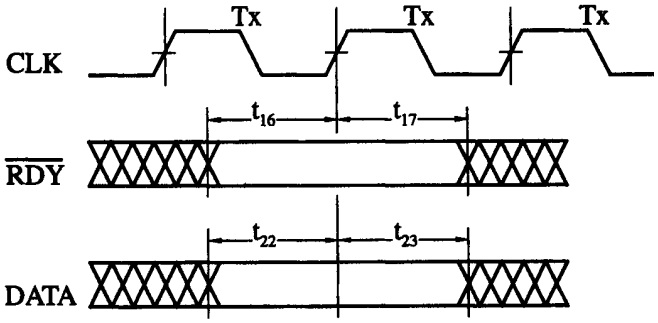
9.2 A Simplified 486 Bus Model - Simplified 486 Bus Interface Unit



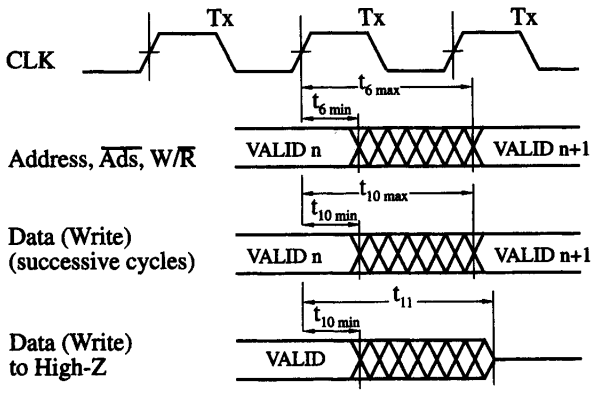
9.2 A Simplified 486 Bus Model - SM Chart for Simplified Bus Interface



9.2 A Simplified 486 Bus Model - 486 Setup and Hold Time Specification



9.2 A Simplified 486 Bus Model - Bus Timing for Address and Data



9.2 A Simplified 486 Bus Model - Bus Interface Unit VHDL Entity

```
entity i486_bus is
    generic (-- These specs are for the i486DX 50
         constant t6_max:time:=12 ns;
         constant t10_min:time:=3 ns;
         constant t10_max:time:=12 ns;
         constant t11_max:time:=18 ns;
         constant t16_min:time:=5 ns;
         constant t17_min:time:=3 ns;
         constant t22_min:time:=5 ns;
         constant t23_min:time:=3 ns);
    port (--external interface
         abus: out bit_vector(31 downto 0);
         dbus: inout std_logic_vector(31 downto 0) := (others => 'Z');
         w_rb, ads_b: out bit := '1';
         rdy_b, clk: in bit;
         --internal interface
         address, w_data: in bit_vector(31 downto 0);
         r_data: out bit_vector(31 downto 0);
         wr, br: in bit;
         std, done:out bit);
end i486_bus;
```


9.2 A Simplified 486 Bus Model - Bus Interface Unit VHDL Entity

```
architecture simple_486_bus of i486_bus is
type state_t is (Ti, T1, T2);
signal state, next_state:state_t:=Ti;
begin
-- The following process outputs the control signals and address of
-- the processor during a read/write operation.The process also drives
-- or tristates the databus depending on the operation type.
-- During the execution of a read/write operation, the done signal
-- is low. When the bus is ready to accept a new request, done is high.
comb_logic: process
begin
    std <= '0';
    case (state) is
      when Ti=>   done<='1';
                    if (br = '1') then next_state <= T1;
                    else next_state <= Ti;
                    end if;
                    dbus <= transport  (others =>'Z') after t10_min;
      when T1=>   done <= '0';
                    ads_b <= transport '0' after t6_max;
                    w_rb <= transport wr after t6_max;
                    abus <= transport address after t6_max;
                    dbus <= transport  (others =>'Z') after t10_min;
                    next_state <= T2;
```

9.2 A Simplified 486 Bus Model - Bus Interface Unit VHDL Entity

```
    when T2=>   ads_b <= transport '1' after t6_max;
                  if (wr = '0') then  -- read
                    if (rdy_b ='0') then
                      r_data <= to_bitvector(dbus);
                      std <= '1';
                      done <= '1';
                      if (br = '0') then next_state <= Ti;
                      else next_state <= T1;
                      end if;
                    else next_state <= T2;
                    end if;
                  else  -- write
                    dbus <= transport to_stdlogicvector(w_data) after t10_max;
                    if (rdy_b = '0') then
                      done<='1';
                      if (br = '0') then next_state <= Ti;
                      else next_state <= T1;
                      end if;
                    else next_state <= T2;
                    end if;
                  end if;
   end case;
   wait on state, rdy_b, br, dbus;
  end process comb_logic;
  seq_logic: process(clk)
  begin
     if (clk = '1') then state <= next_state; end if;
  end process seq_logic;
```


9.2 A Simplified 486 Bus Model - Bus Interface Unit VHDL Entity

```
--The following process checks that all setup and hold times are met for all incoming
-- control signals. Setup and hold times are checked for the data bus during a read only
wave_check: process (clk, dbus, rdy_b)
    variable clk_last_rise:time:= 0 ns;
begin
    if (state=T2) then
       if clk'event and clk = '1' then                    --check setup times
          --The following assert assumes that the setup for RDY
          --  is equal to or greater than that for data
          assert (rdy_b /= '0') OR (wr /= '0') OR (dbus'last_event >= t22_min)
          report "i486 bus:Data setup too short" severity WARNING;
          assert (rdy_b'last_event >= t16_min)
          report "i486 bus:RDY setup too short" severity WARNING;
          clk_last_rise := NOW;
       end if;
       if (dbus'event) then                       --check hold times
          --The following assert assumes that the hold for RDY
          --  is equal to or greater than that for data
          assert (rdy_b /= '0') OR (wr /= '0') OR (now - clk_last_rise >= t23_min)
          report "i486 bus:Data hold too short" severity WARNING;
       end if;
       if (rdy_b'event) then
          assert (now - clk_last_rise >= t17_min)
          report "i486 bus: RDY signal hold too short" severity WARNING;
       end if;
    end if;
end process wave_check;
end simple_486_bus;
```