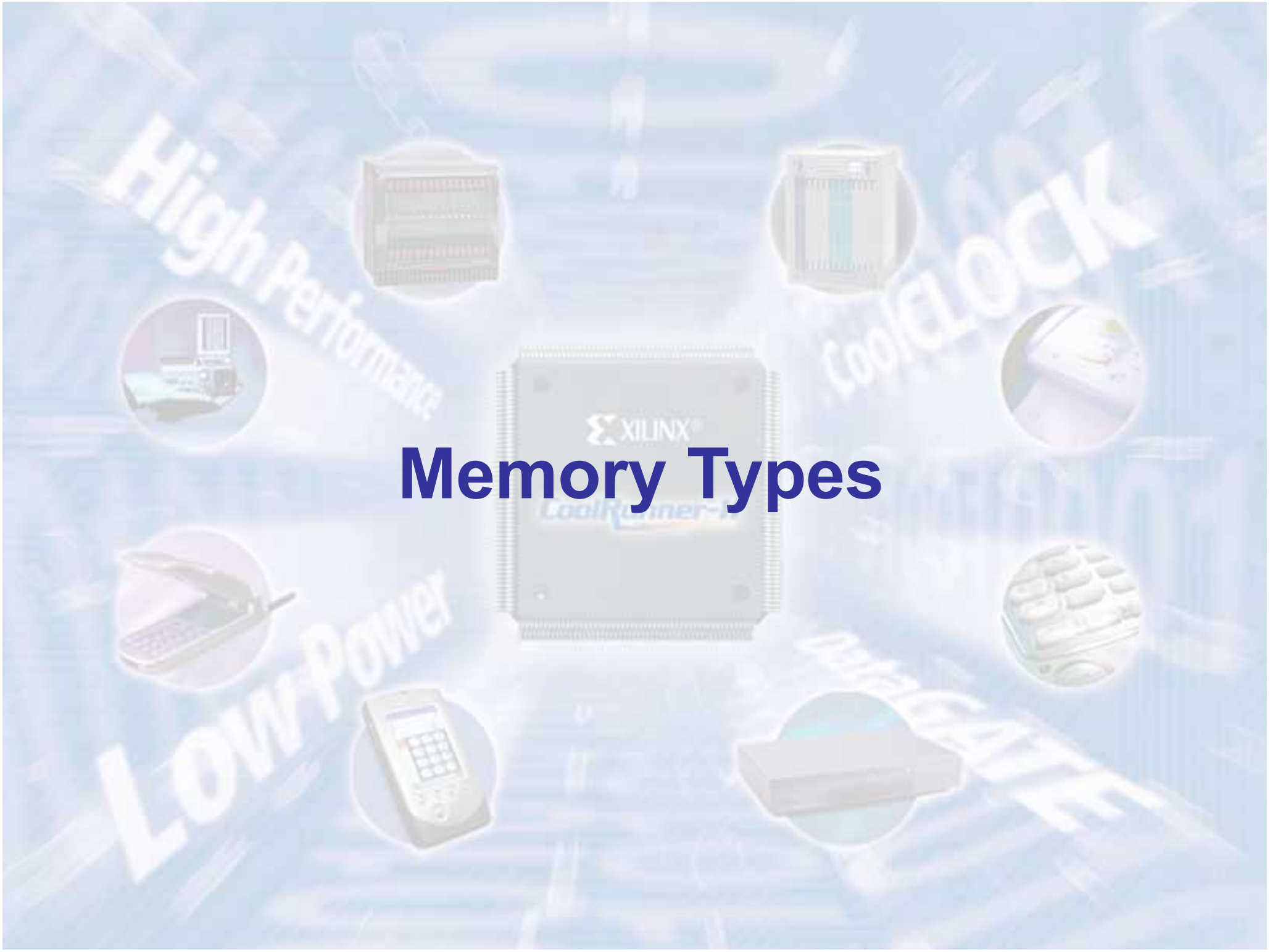
ECE 545: Lecture 9

RAM

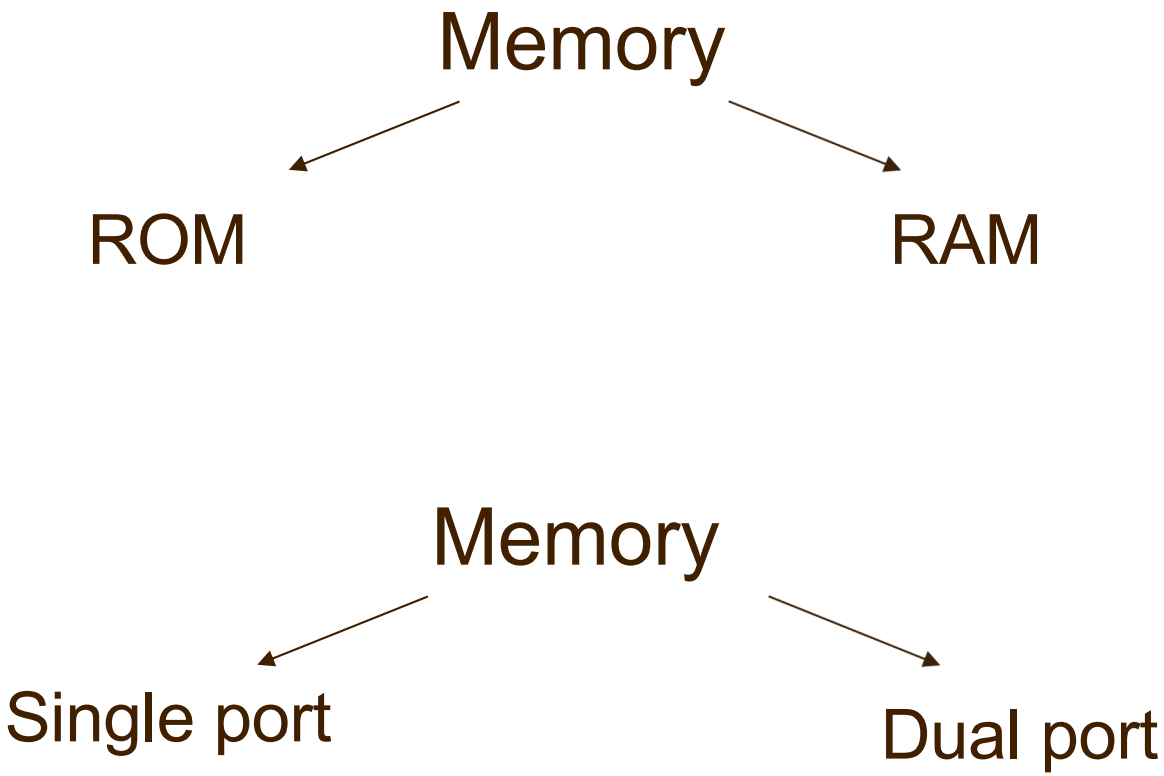
Recommended reading

- Vivado Design Suite User Guide: Synthesis
  Chapter 4
    - RAM HDL Coding Techniques
    - Initializing RAM Contents

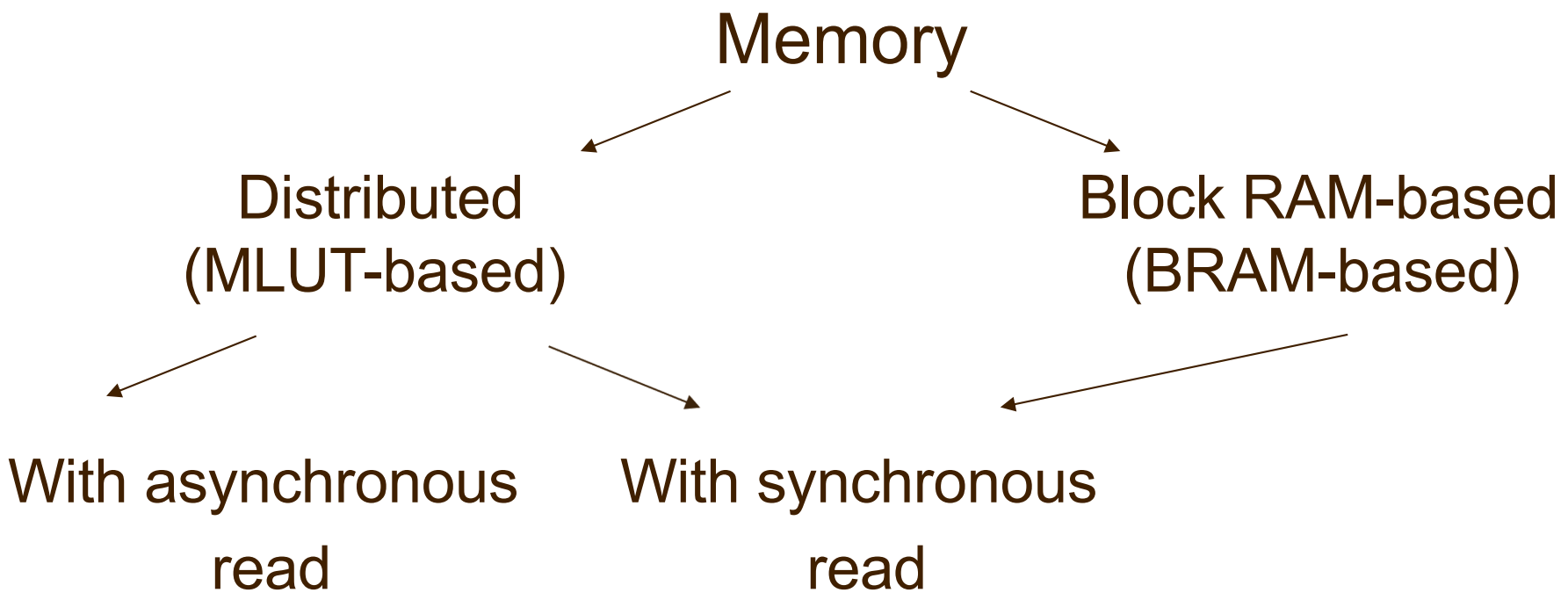


Memory Types

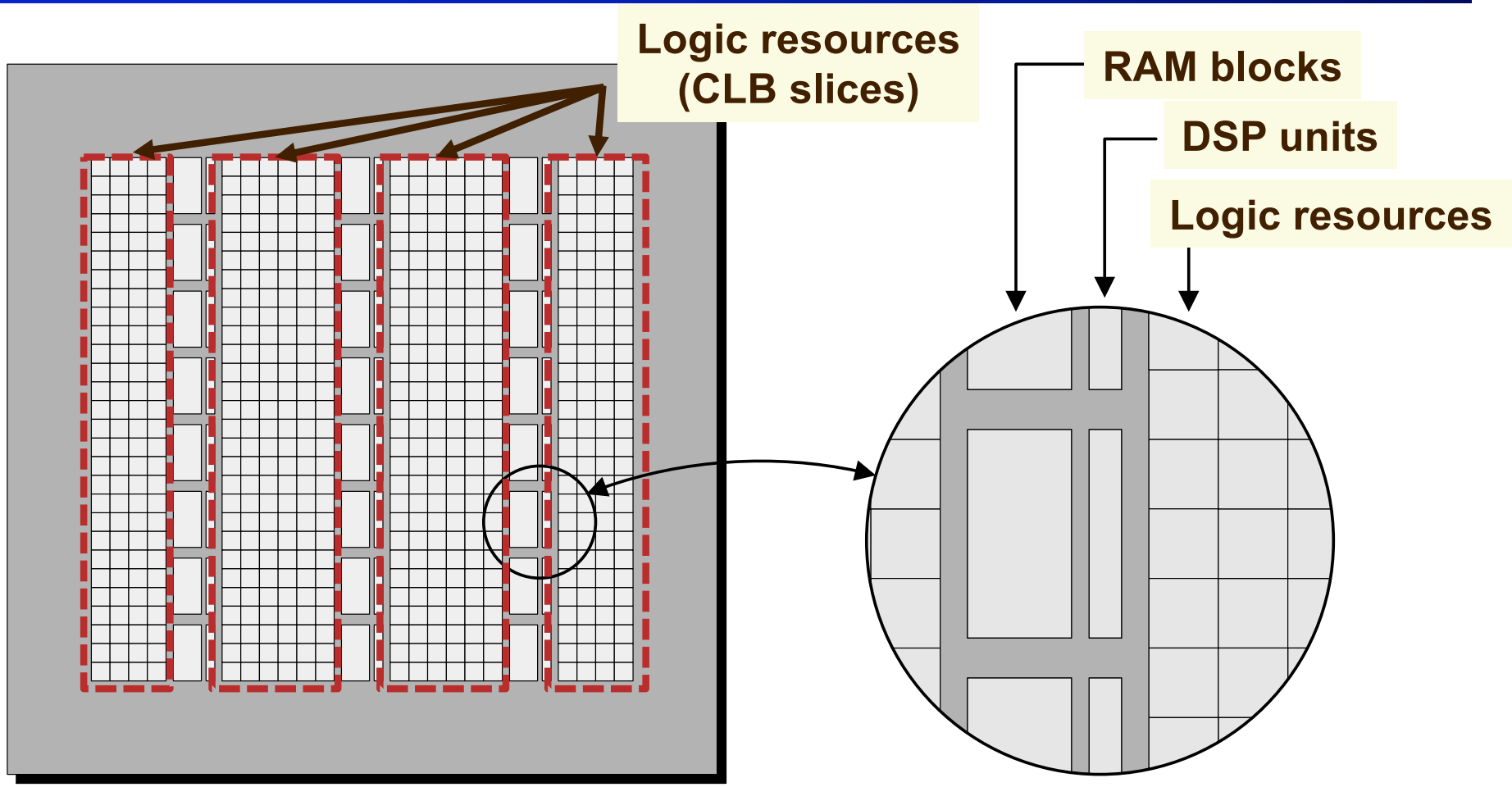
Memory Types



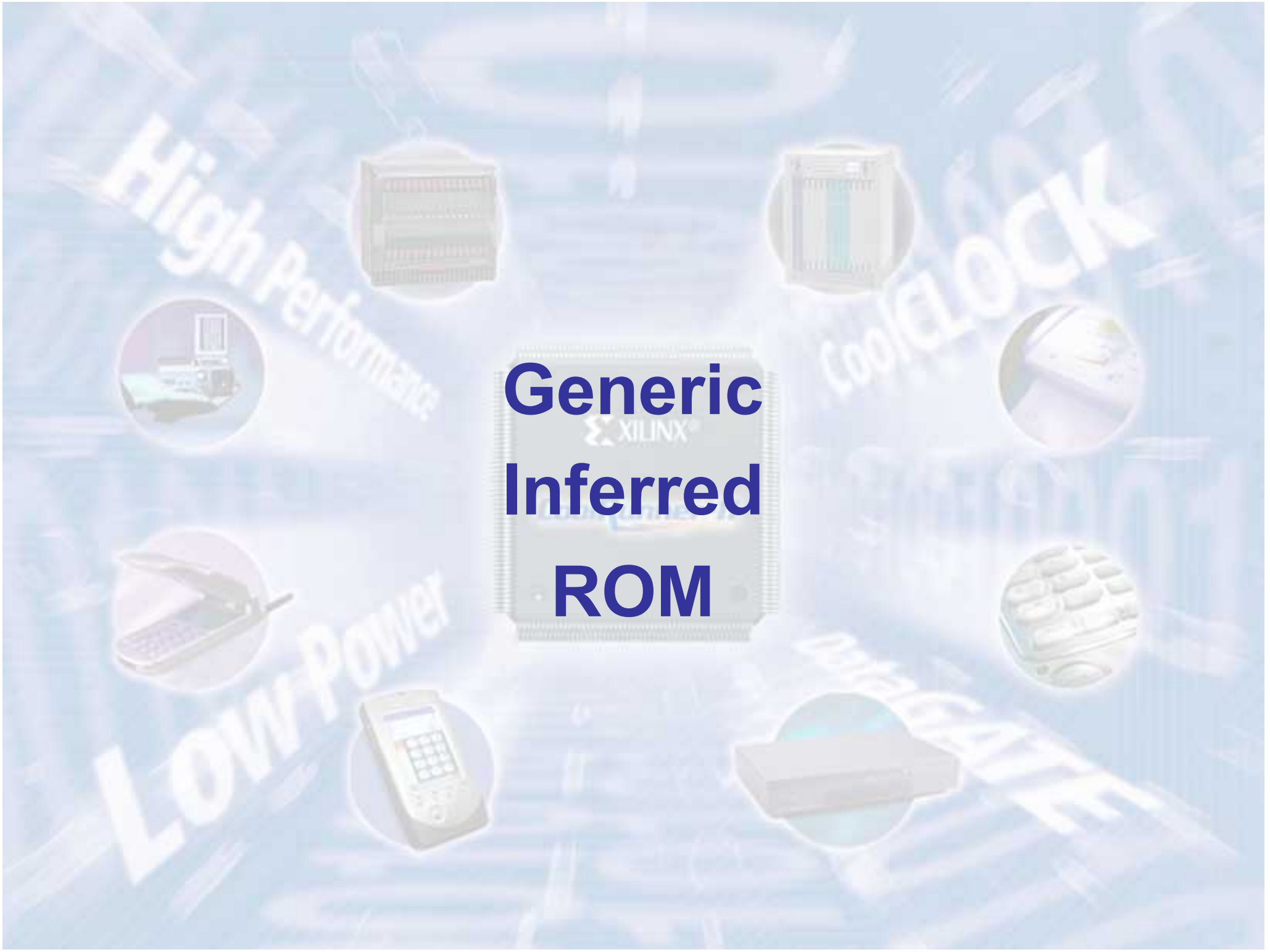
Memory Types specific to Xilinx FPGAs



Location of RAM



Graphics based on The Design Warrior's Guide to FPGAs
Devices, Tools, and Flows. ISBN 0750676043
Copyright © 2004 Mentor Graphics Corp. (www.mentor.com)



Generic Inferred ROM

Single-port ROM with asynchronous read

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

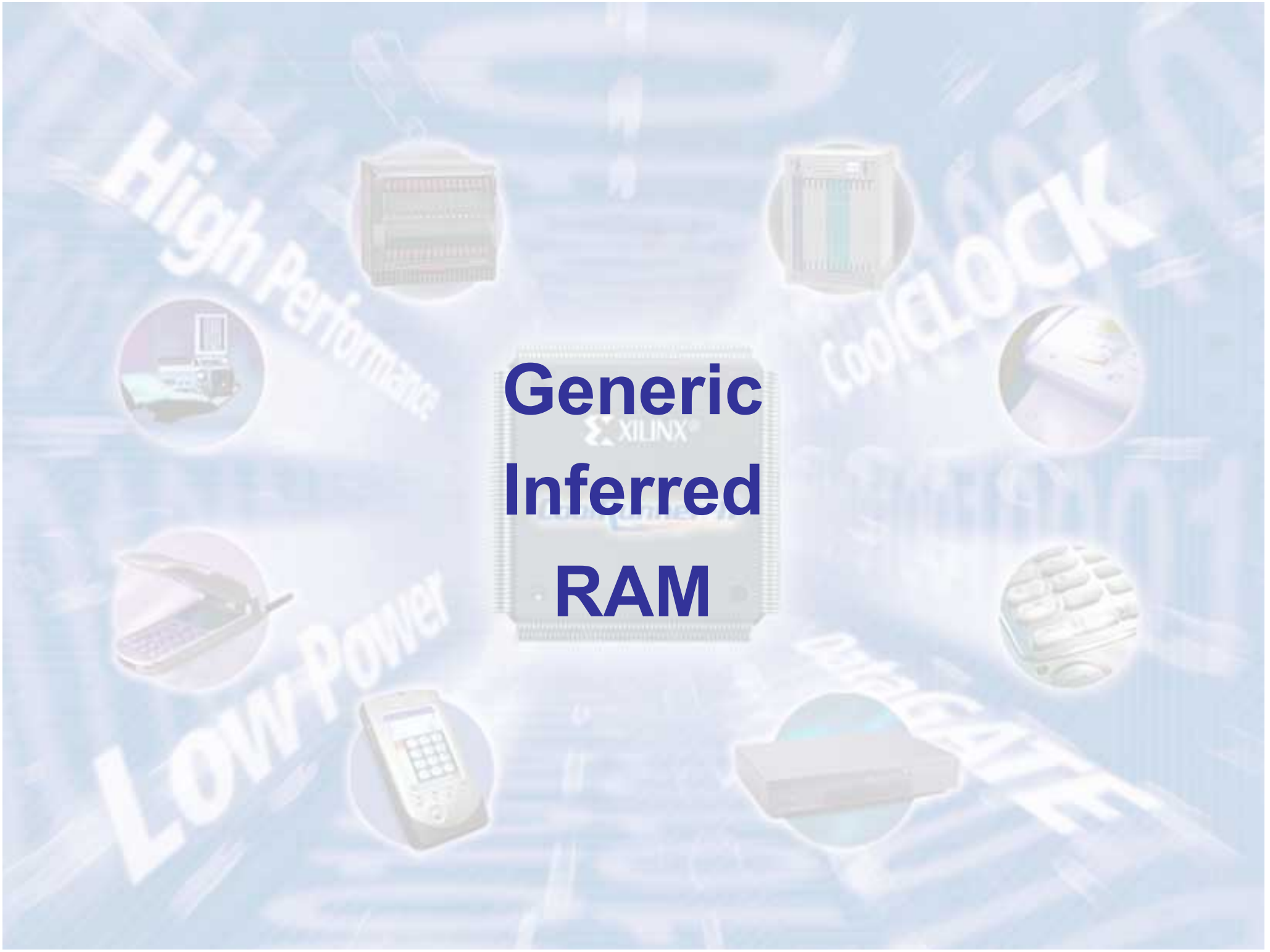
Entity ROM is
  generic ( w : integer := 12;
              -- number of bits per ROM word
            r : integer := 3);
              -- 2^r = number of words in ROM
  port (addr   : in std_logic_vector(r-1 downto 0);
        dout   : out std_logic_vector(w-1 downto 0));
end ROM;
```

Single-port ROM with asynchronous read

```
architecture behavioral of rominfr is
  type rom_type is array (0 to 2**r-1)
    of std_logic_vector (w-1 downto 0);
  constant ROM_array : rom_type :=
  ("000011000100",
   "010011010010",
   "010011011011",
   "011011000010",
   "000011110001",
   "011111010110",
   "010011010000",
   "111110011111");
begin
  dout <= ROM_array(to_integer(unsigned(addr)));
end behavioral;
```

Single-port ROM with asynchronous read

```
architecture behavioral of rominfr is
  type rom_type is array (0 to 2**r-1)
    of std_logic_vector (w-1 downto 0);
  constant ROM_array : rom_type :=
  (X"0C4",
   X"4D2",
   X"4DB",
   X"6C2",
   X"0F1",
   X"7D6",
   X"4D0",
   X"F9F");
begin
  dout <= ROM_array(to_integer(unsigned(addr)));
end behavioral;
```



Generic Inferred RAM

Single-port RAM with asynchronous read

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

entity raminfr is
  generic ( w : integer := 32;
            -- number of bits per RAM word
            r : integer := 6);
            -- 2^r = number of words in RAM
  port (clk : in std_logic;
        we  : in std_logic;
        a   : in std_logic_vector(r-1 downto 0);
        di  : in std_logic_vector(w-1 downto 0);
        do  : out std_logic_vector(w-1 downto 0));
end raminfr;
```

Single-port RAM with asynchronous read

```
architecture behavioral of raminfr is
  type ram_type is array (0 to 2**r-1)
        of std_logic_vector (w-1 downto 0);
  signal RAM : ram_type := (others => (others => '0'));
begin
  process (clk)
  begin
    if rising_edge(clk) then
      if (we = '1') then
        RAM(to_integer(unsigned(a))) <= di;
      end if;
    end if;
  end process;
  do <= RAM(to_integer(unsigned(a)));
end behavioral;
```

Dual-port RAM with asynchronous read

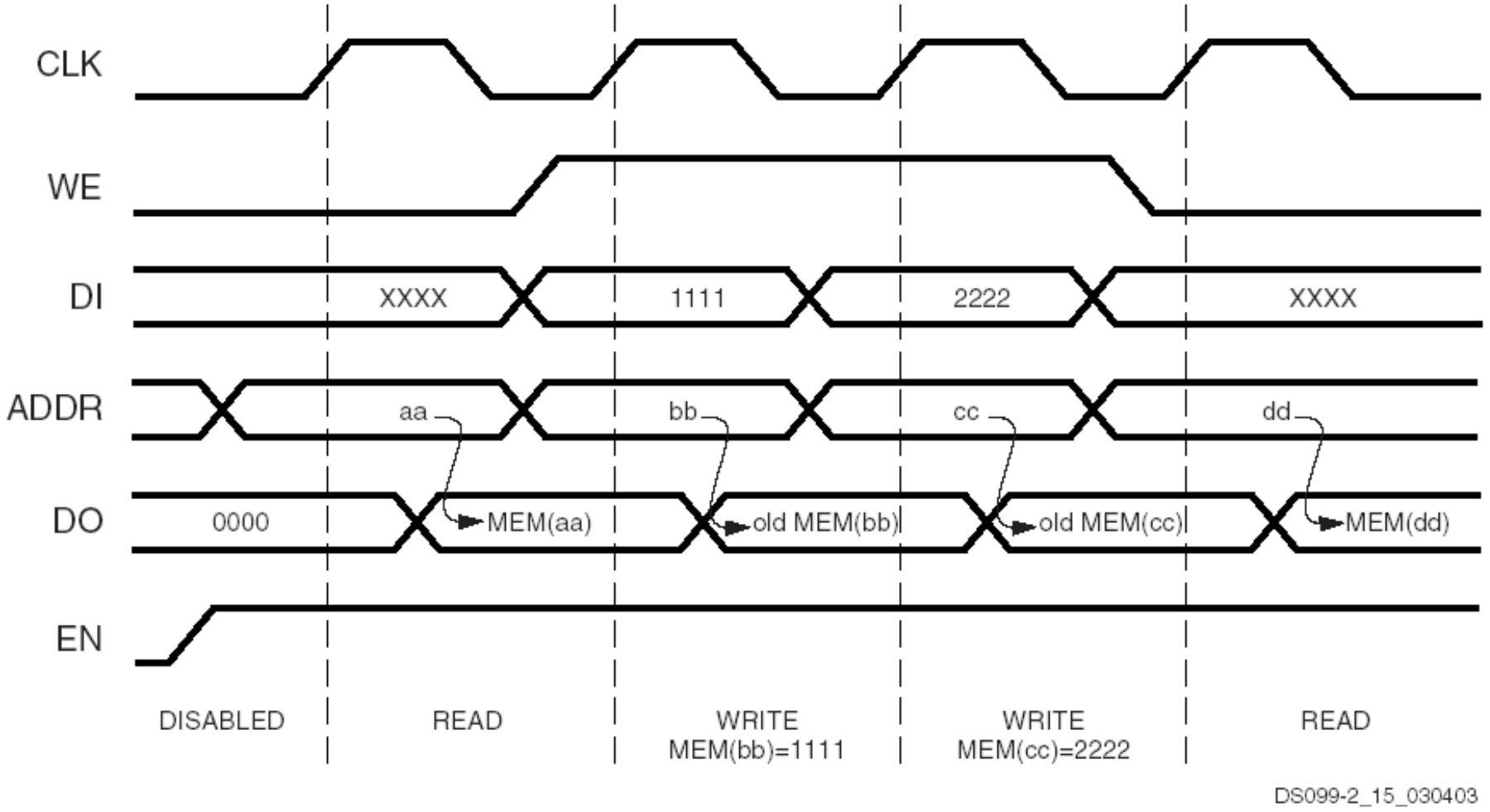
```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity raminfr is
 generic ( w : integer := 32;
             -- number of bits per RAM word
           r : integer := 6);
             -- 2^r = number of words in RAM
  port (clk  : in std_logic;
        we   : in std_logic;
        a    : in std_logic_vector(r-1 downto 0);
        dpra : in std_logic_vector(r-1 downto 0);
        di   : in std_logic_vector(w-1 downto 0);
        spo  : out std_logic_vector(w-1 downto 0);
        dpo  : out std_logic_vector(w-1 downto 0));
end raminfr;
```

Dual-port RAM with asynchronous read

```
architecture syn of raminfr is
  type ram_type is array (0 to 2**r-1) of
            std_logic_vector (w-1 downto 0);
  signal RAM : ram_type := (others => (others => '0'));
begin
  process (clk)
  begin
    if rising_edge(clk) then
      if (we = '1') then
        RAM(to_integer(unsigned(a))) <= di;
      end if;
    end if;
  end process;
  spo <= RAM(to_integer(unsigned(a)));
  dpo <= RAM(to_integer(unsigned(dpra)));
 end syn;
```

Block RAM Waveforms – READ_FIRST mode



Block RAM with synchronous read

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;

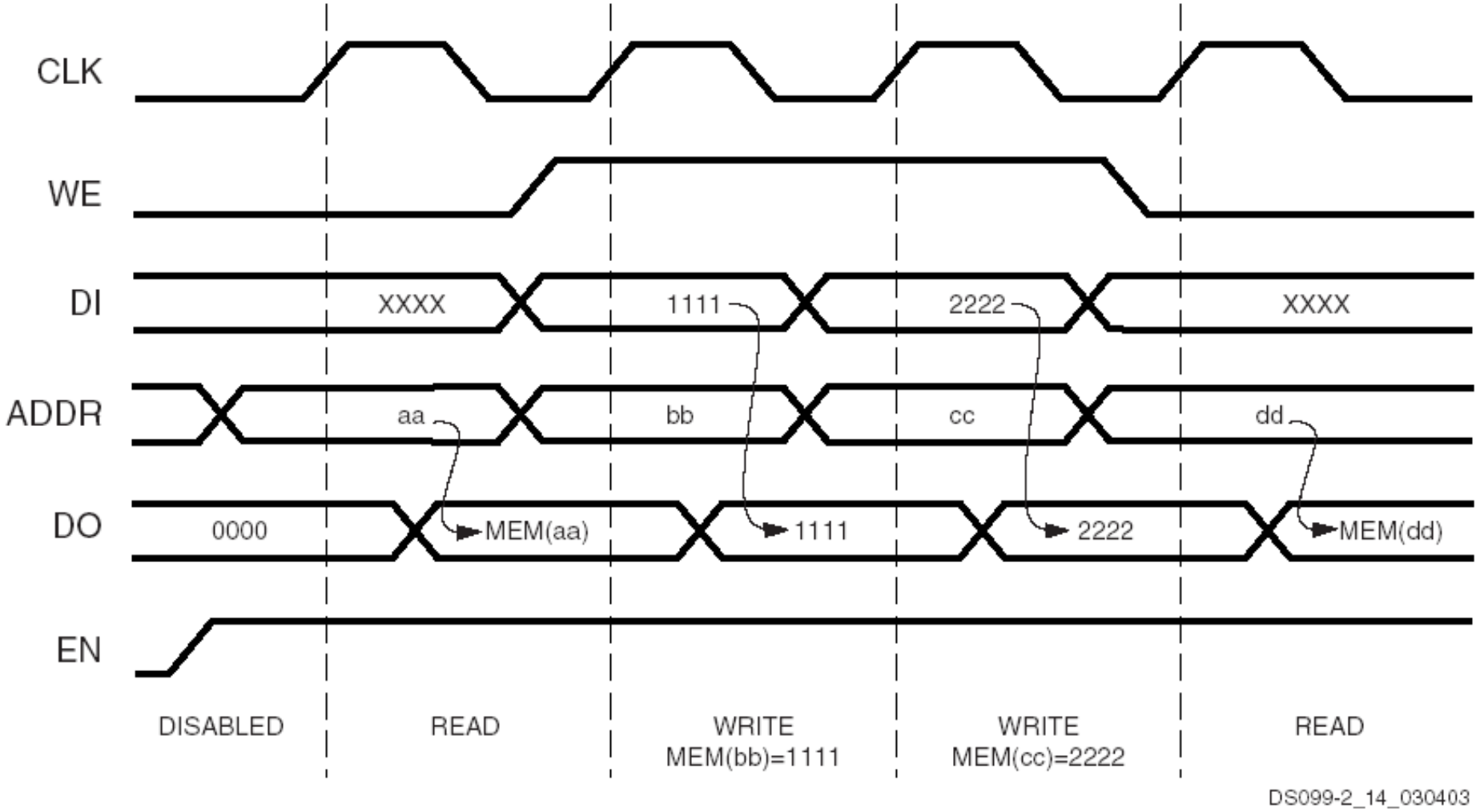
entity raminfr is
  generic ( w : integer := 32;
              -- number of bits per RAM word
            r : integer := 9);
              -- 2^r = number of words in RAM
  port (clk : in std_logic;
        we  : in std_logic;
        en  : in std_logic;
        addr   : in std_logic_vector(r-1 downto 0);
        di  : in std_logic_vector(w-1 downto 0);
        do  : out std_logic_vector(w-1 downto 0));
end raminfr;
```

Block RAM with synchronous read Read-First Mode - cont'd

```
architecture behavioral of raminfr is
  type ram_type is array (0 to 2**r-1) of
                        std_logic_vector (w-1 downto 0);
  signal RAM : ram_type := (others => (others => '0'));

begin
  process (clk)
  begin
    if rising_edge(clk) then
      if (en = '1') then
        do <= RAM(to_integer(unsigned(addr)));
        if (we = '1') then
           RAM(to_integer(unsigned(addr))) <= di;
        end if;
      end if;
    end if;
  end process;
end behavioral;
```

Block RAM Waveforms – WRITE_FIRST mode

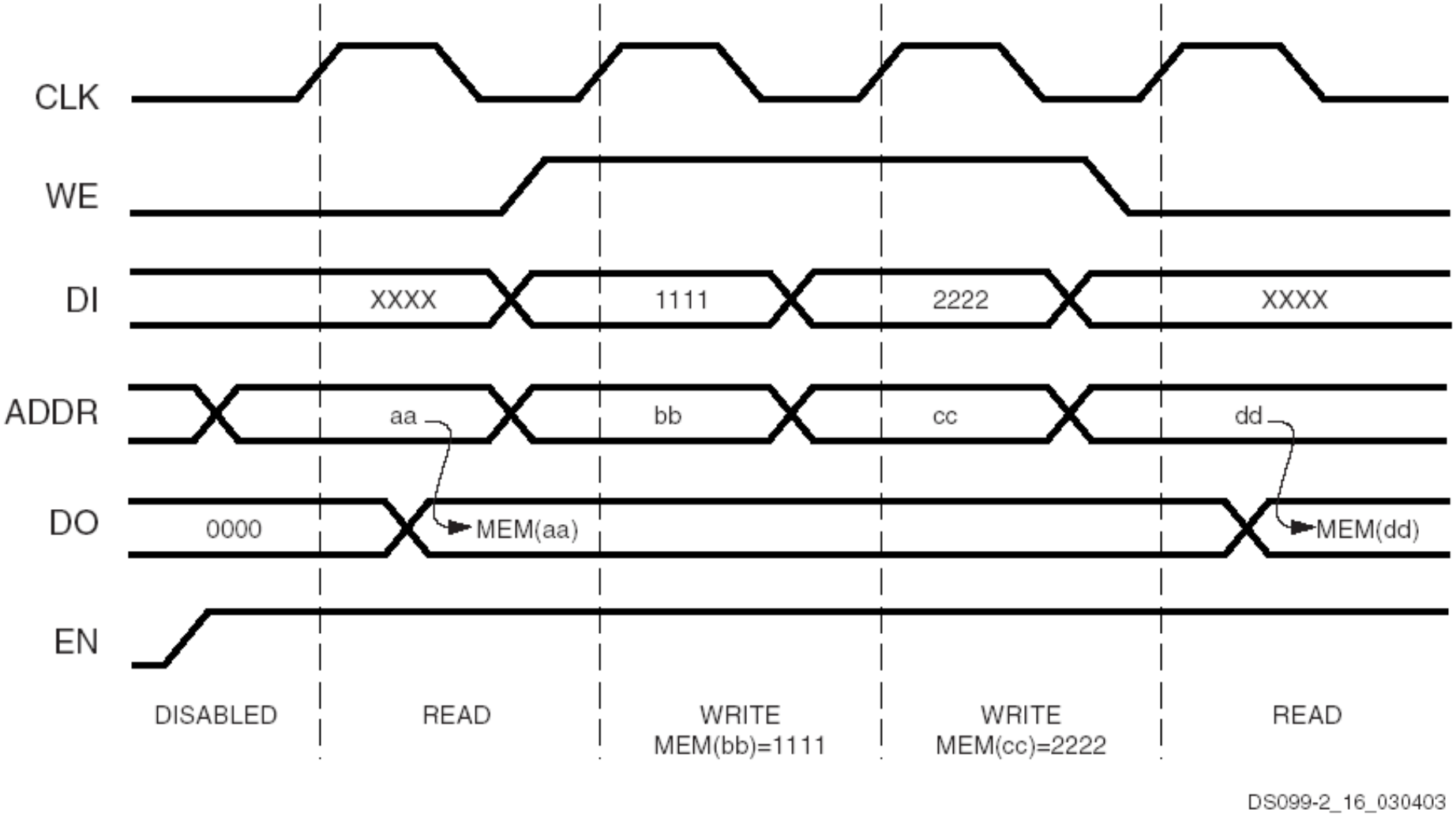


Block RAM with synchronous read Write-First Mode - cont'd

```
architecture behavioral of raminfr is
  type ram_type is array (0 to 2**r-1) of
                        std_logic_vector (w-1 downto 0);
  signal RAM : ram_type := (others => (others => '0'));

begin
  process (clk)
  begin
    if (clk'event and clk = '1') then
      if (en = '1') then
        if (we = '1') then
           RAM(to_integer(unsigned(addr))) <= di;
           do <= di;
        else
           do <= RAM(to_integer(unsigned(addr)));
        end if;
      end if;
    end if;
  end process;
end behavioral;
```

Block RAM Waveforms – NO_CHANGE mode



Block RAM with synchronous read No-Change Mode - cont'd

```
architecture behavioral of raminfr is
  type ram_type is array (0 to 2**r-1) of
                        std_logic_vector (w-1 downto 0);
  signal RAM : ram_type := (others => (others => '0'));

begin
  process (clk)
  begin
    if (clk'event and clk = '1') then
      if (en = '1') then
        if (we = '1') then
           RAM(to_integer(unsigned(addr))) <= di;
        else
           do <= RAM(to_integer(unsigned(addr)));
        end if;
      end if;
    end if;
  end process;
end behavioral;
```