Memory

RAM – Random Access Memory

- Used for storing variables for programs

ROM - Read Only Memory

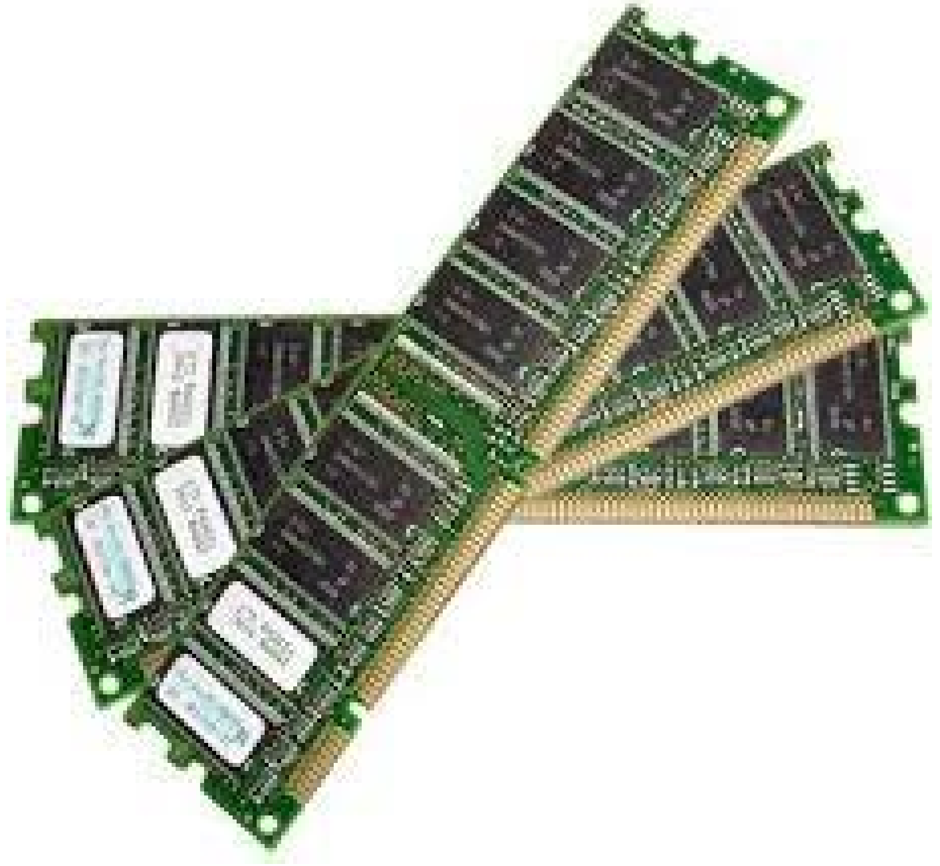
- Typically used for booting

ROM and RAM both allow for random access

- RAM ⇔ read/write RAM
- ROM ⇔ read only RAM

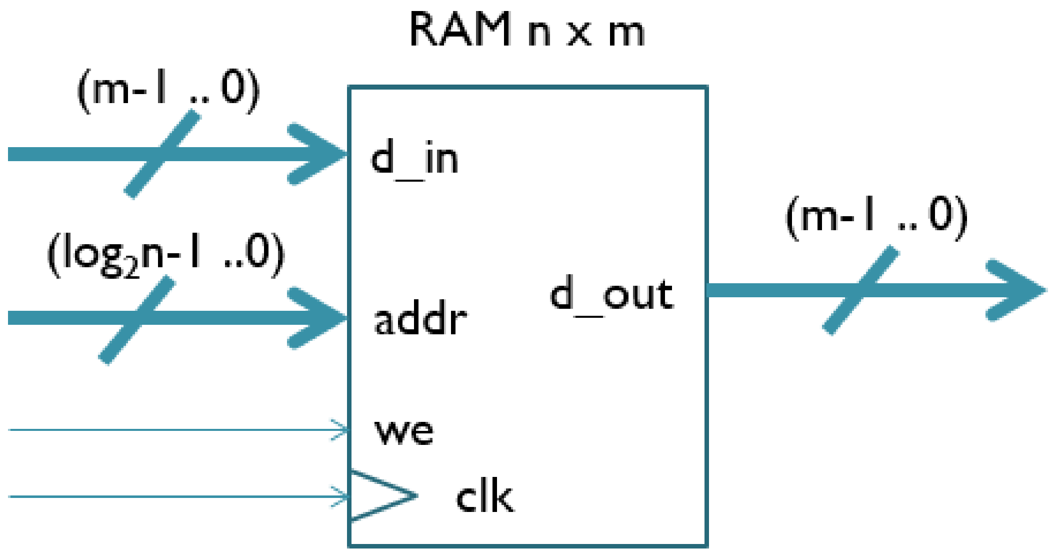
How many gigs of RAM can a 32 bit OS use?

- 4 GB = 2^32 = 4294967296 bytes



RAM Configuration

- An **n x m** RAM has n locations each of size m
  - Example 4K x 128 :
    - 4096 locations (1K = 1024)
    - The data in each location is 128 bits long
  - m is the size of the d_in bus and the d_out bus
  - n and m have no relationship
  - The size of the address bus is $\log_2 n$. Each location has a unique address



VHDL for Simple RAM

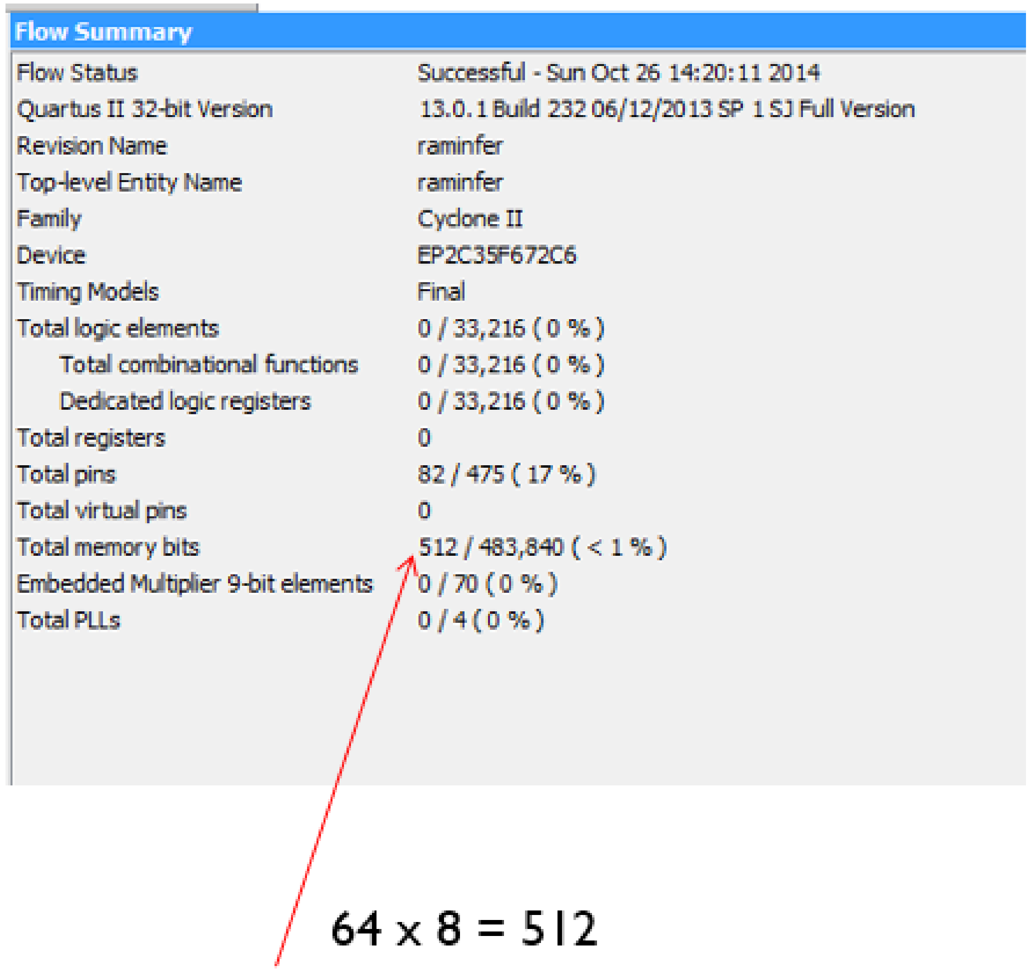
```vhdl
entity raminfer is
   generic (addr_width  : integer := 6;   --default to a 64 x 256 RAM
            data_width  : integer := 8);
   port(
         clk   : in std_logic;
         we    : in std_logic;
         addr  : in std_logic_vector((addr_width - 1) downto 0);
         d_in  : in std_logic_vector(data_width - 1 downto 0);
         d_out : out std_logic_vector(data_width - 1 downto 0));
end raminfer;

architecture rtl of raminfer is

type ram_type is array ((2**addr_width - 1) downto 0) of std_logic_vector (data_width - 1 downto 0);
signal RAM : ram_type;

begin
process(clk)
   begin
   if (clk'event and clk = '1') then
      if (we = '1') then
         RAM(to_integer(unsigned(addr))) <= d_in;
      end if;
      d_out <= RAM(to_integer(unsigned(addr)));
   end if;
 end process;
end rtl;
```

RAM Synthesis


Flow Summary

Flow Status	Successful - Sun Oct 26 14:20:11 2014
Quartus II 32-bit Version	13.0.1 Build 232 06/12/2013 SP 1 SJ Full Version
Revision Name	raminfer
Top-level Entity Name	raminfer
Family	Cyclone II
Device	EP2C35F672C6
Timing Models	Final
Total logic elements	0 / 33,216 (0 %)
Total combinational functions	0 / 33,216 (0 %)
Dedicated logic registers	0 / 33,216 (0 %)
Total registers	0
Total pins	82 / 475 (17 %)
Total virtual pins	0
Total memory bits	512 / 483,840 (< 1 %)
Embedded Multiplier 9-bit elements	0 / 70 (0 %)
Total PLLs	0 / 4 (0 %)

$64 \times 8 = 512$

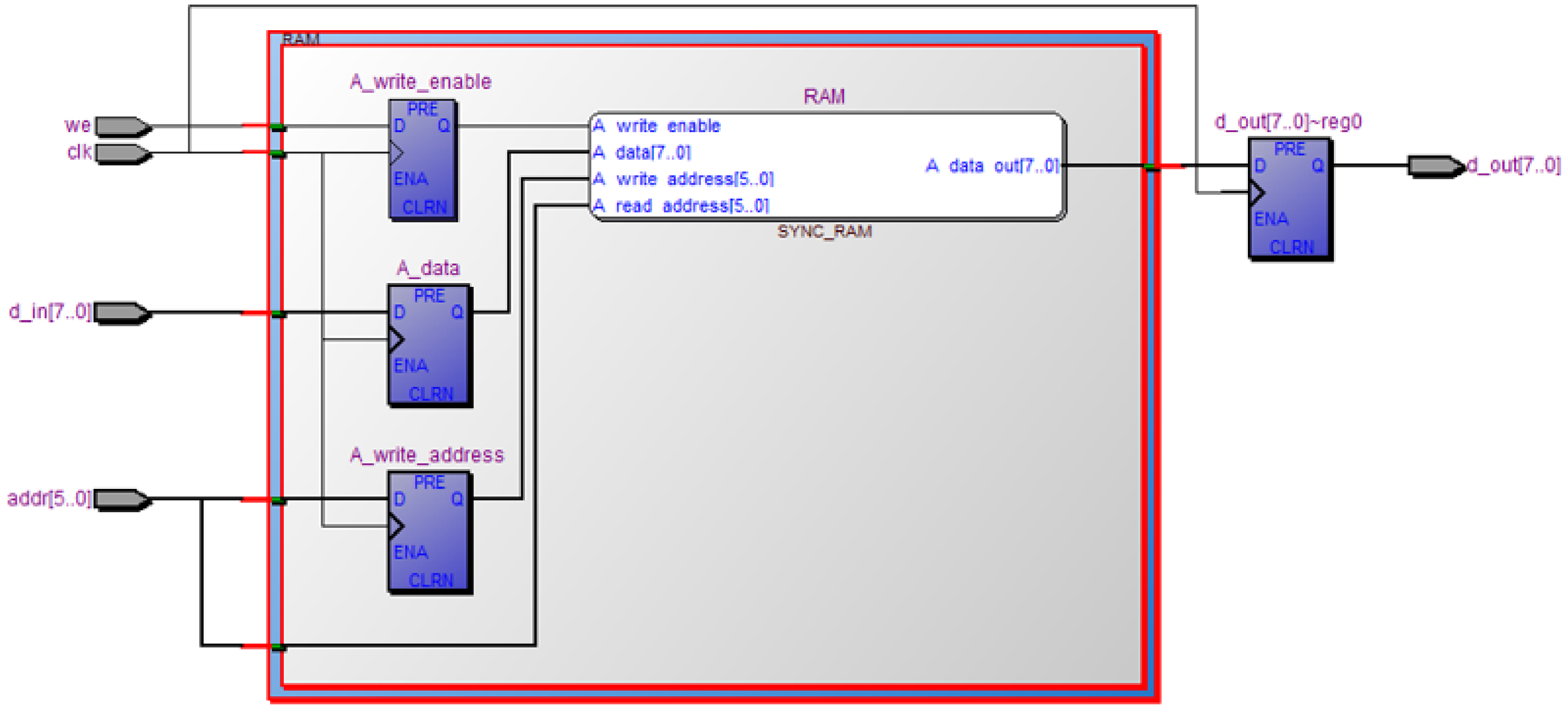
Distributed RAM

- FPGA logic blocks have configurable lookup tables
- Each LUT can be used as a couple bits of RAM
- Many LUTs can be chained together to create a larger RAM block

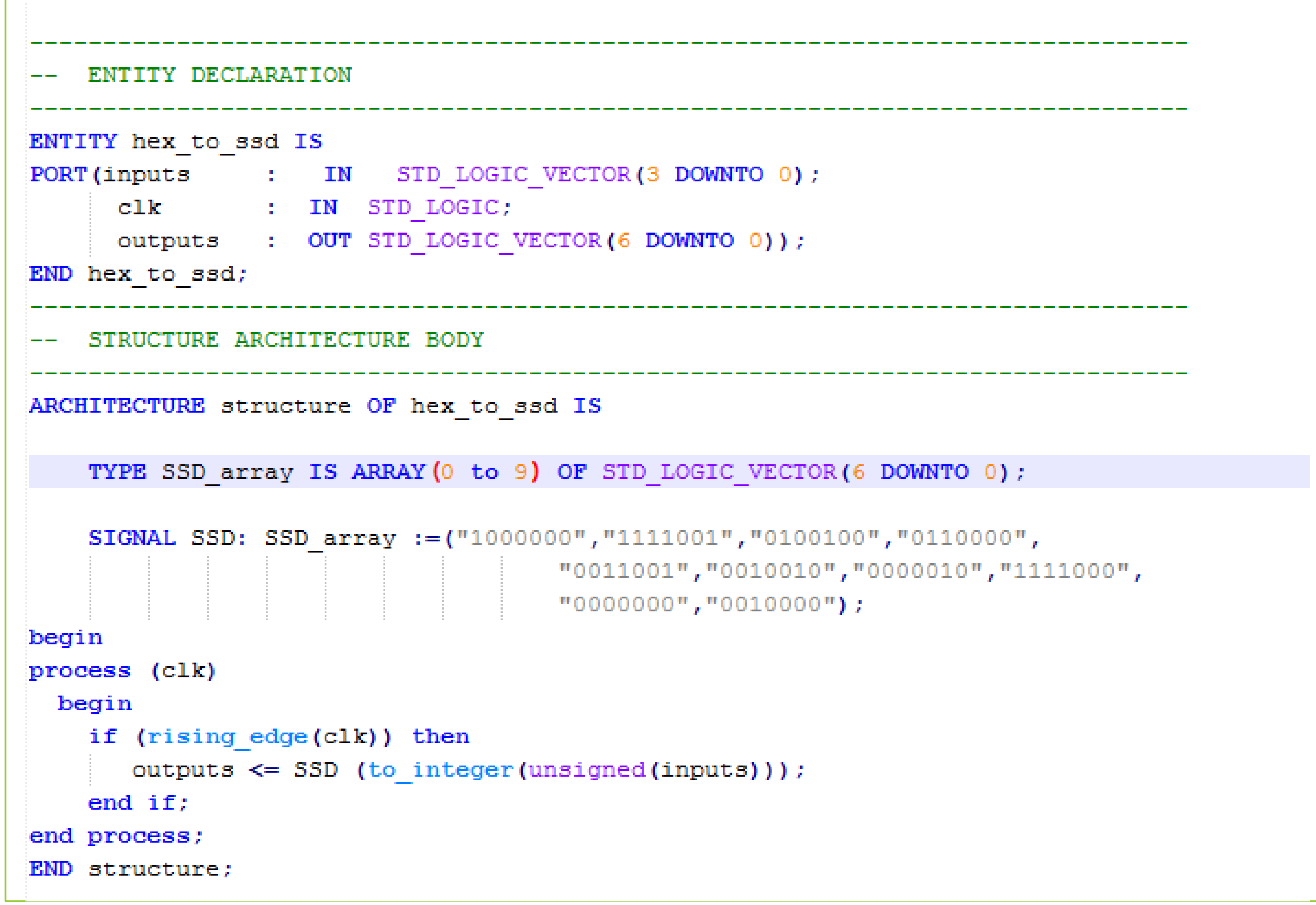
Block RAM

- Dedicated memory containing several kilobytes of RAM

RAM RTL View


RAM
A_write_enable
PRE
D
Q
ENA
CLRN
A_data
A_write_address
RAM
A write enable
A data[7..0]
A write address[5..0]
A read address[5..0]
A data out[7..0]
SYNC_RAM
d_out[7..0]~reg0
we
clk
d_in[7..0]
addr[5..0]
d_out[7..0]

ROM



```
--------------------------------------------------------------------------------
--  ENTITY DECLARATION
--------------------------------------------------------------------------------
ENTITY hex_to_ssd IS
PORT(inputs     :   IN   STD_LOGIC_VECTOR(3 DOWNTO 0);
     clk        :  IN  STD_LOGIC;
     outputs    :  OUT STD_LOGIC_VECTOR(6 DOWNTO 0));
END hex_to_ssd;
--------------------------------------------------------------------------------
--  STRUCTURE ARCHITECTURE BODY
--------------------------------------------------------------------------------
ARCHITECTURE structure OF hex_to_ssd IS

    TYPE SSD_array IS ARRAY(0 to 9) OF STD_LOGIC_VECTOR(6 DOWNTO 0);

    SIGNAL SSD: SSD_array :=("1000000","1111001","0100100","0110000",
                                "0011001","0010010","0000010","1111000",
                                "0000000","0010000");
begin
process (clk)
  begin
    if (rising_edge(clk)) then
       outputs <= SSD (to_integer(unsigned(inputs)));
    end if;
end process;
END structure;
```

How does ROM differ from RAM?

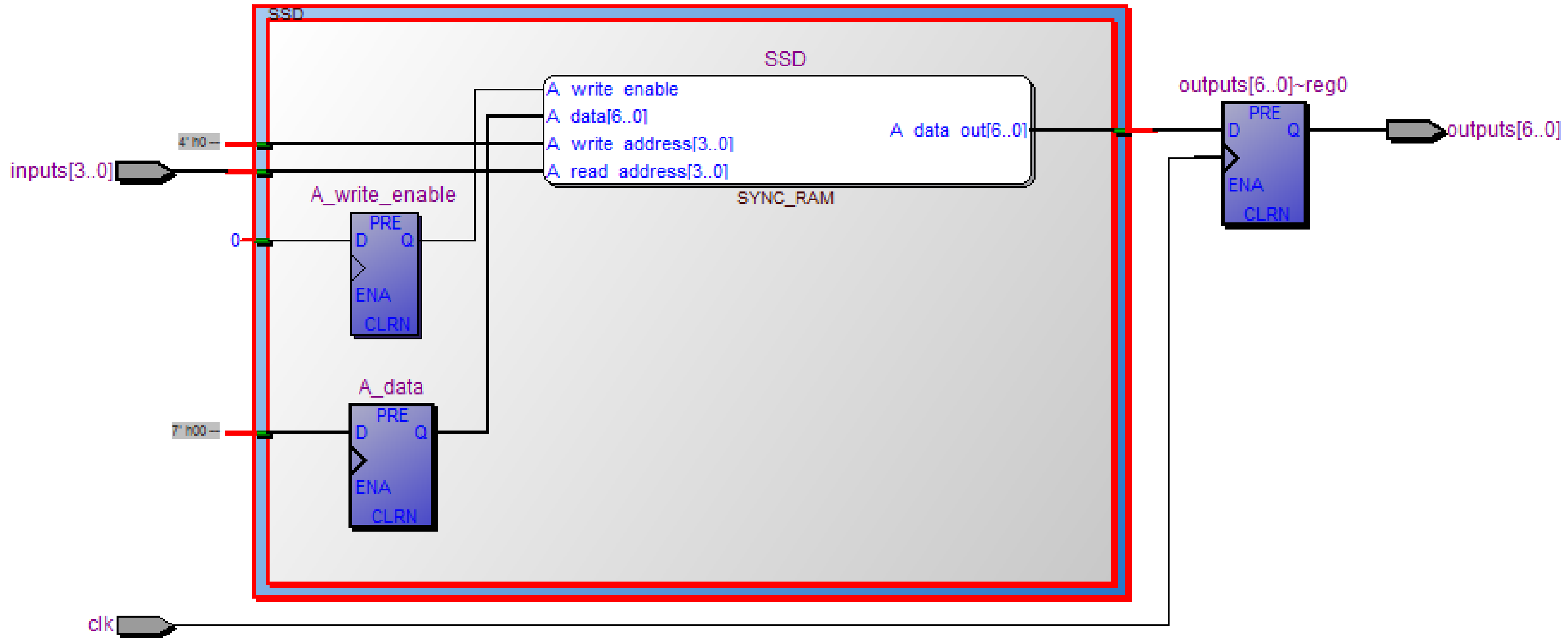
The configuration is the same (n x m)

ROM is inferred when VHDL is written as a look up table

case

array

ROM RTL View


SSD
SSD
A write enable
A data[6..0]
A write address[3..0]
A read address[3..0]
A data out[6..0]
SYNC_RAM
4' h0
inputs[3..0]
0
A_write_enable
PRE
D
Q
ENA
CLRN
7' h00
A_data
PRE
D
Q
ENA
CLRN
outputs[6..0]~reg0
PRE
D
Q
ENA
CLRN
outputs[6..0]
clk