Digital Systems Design

Memory Implementation on Altera CYCLONE V Devices

Embedded Memory

• 10 Kb M10K blocks—blocks of dedicated memory resources
  – The M10K blocks are well suited for larger memory arrays
• 640 bit memory logic array blocks (MLABs)—enhanced memory blocks that are configured from dual-purpose logic array blocks (LABs)
• Memory capacity (Cyclone V, A4)
  – 308 M10K blocks (3080 Kbits)
  – 485 MLAB blocks (303 Kbits)
• Selected embedded memory features
  – Multiple memory modes: single/dual port
  – Port Width Configuration
  – Mixed-width Port Configuration
  – Clocking Modes and Clock Enable
  – Byte Enable
  – Asynchronous Clear
  – Read Enable
Dual-Port Memory Configuration

- True dual port operation
  - Two data input busses
  - Two data output busses
  - Two address busses
  - Two independent clocks
  - The memory blocks also enable mixed-width data ports for reading and writing to the RAM ports in dual-port RAM configuration
  - For example, the memory block can be written in ×1 mode at port A and read out in ×16 mode from port B

Simple Dual-Port & Single-Port Memory Configurations

- Simple Dual-Port Memory
  - One read port
  - One write port
  - Independent clocks

- Single-Port Memory
  - One address bus
  - Separate data input and output busses
  - One clock source
  - Two single-port memory blocks can be implemented in a single M10K block as long as each of the two independent block sizes is equal to or less than half of the M10K block size
Synchronous Memory

- The CYCLONE V memory architecture can implement fully synchronous RAM by registering both the input and output signals to the M10K RAM block.
- All M10K memory block inputs are registered, providing synchronous write cycles.
- The output registers can be bypassed.
  - Asynchronous reading is possible in the simple dual-port mode of M10K blocks by bypassing the output registers.

Implementing Larger and/or Wider Memory

- The Quartus software automatically implements larger memory by combining multiple M10K memory blocks.
  - For example, two 256×16-bit RAM blocks can be combined to form a 256×32-bit RAM block.
- M10K block usage is generally transparent to the designers VHDL code.
  - M10K blocks are used as required by the design specifications (i.e. the memory length and width specified in the VHDL source).
Memory Implementation

- Memory can be implemented by
  - Instantiation
    - Creating VHDL that creates an instance of a particular (predefined) memory component
    - Altera’s `altsyncram` megafuncation will be most commonly used
    - Can write structural VHDL or use Quartus Megawizard Plug-in Manager to generate structural VHDL
  - Inference
    - Write behavioral VHDL that will model the memory
    - VHDL compiler can infer the memory
    - Will need to carefully construct VHDL code for the compiler to do this

MegaWizard Plug-In Manager

- Tools->IP Catalog->Library->Basic Functions->On Chip Memory-RAM: 1-PORT
MegaWizard

Device Family Specification

Memory Size Specification

Port Registering

Input data and address will be registered

Output data will be registered
Read During Write Operation

Memory Initialization

DEPTH = 1024; % Memory Depth %
WIDTH = 8; % Memory Width %
ADDRESS_RADIX = HEX;
% Address and value radices are optional%
DATA_RADIX = HEX;
% Enter BIN, DEC, HEX, or OCT; unless%
% otherwise specified, radices = HEX %
-- Specify values for addresses, which can be
-- single address or range
CONTENT
BEGIN
  0  : 01 ;
  1  : 02 ;
  2  : 03 ;
[3..3FF] : FF ;
% Addresses 0x003-0x3FF contain FF %
END ;
library ieee;
use ieee.std_logic_1164.all;

library altera_mf;
use altera_mf.altera_mf_components.all;

entity memtest is
port
(  
  address : in std_logic_vector (9 downto 0);
  clock   : in std_logic := '1';
  data    : in std_logic_vector (7 downto 0);
  wren    : in std_logic;
  q       : out std_logic_vector (7 downto 0)
);
end memtest;
Memory VHDL Code

```vhdl
architecture syn of memtest is

  signal sub_wire0 : std_logic_vector (7 downto 0);

begin

  q <= sub_wire0(7 downto 0);
  altsyncram_component : altsyncram
  generic map (  
    numwords_a => 1024,
    width_ad_a => 10,
    width_a => 8,
  )
  port map (    
    address_a => address,    
    clock0 => clock,    
    data_a => data,    
    wren_a => wren,    
    q_a => sub_wire0    
  );

end syn;
```
Viewing/Changing/Saving Memory

• Memory contents may be viewed
  – Prior to simulation
    • To verify initial contents
  – After simulation
    • To verify simulation results

• Memory contents may be saved to a new *.mif file
  – Import into other VHDL designs
  – Used by another program to verify simulation results
    • Example: MIF contents analyzed by a C or MATLAB program for verification

Memory Editor

• The Memory Editor allows you to enter, edit, and view the memory contents for a memory block implemented in an Altera device in a
  – Memory Initialization File (.mif)
  – Hexadecimal (Intel-Format) File (.hex)

• You can also use the Memory Editor to view and edit memory cells and their values during simulation.
  – You can create a new MIF or HEX File, and then specify the memory contents for a memory block in the design.
  – You can edit and adjust the memory cells and their values as needed before saving the MIF or HEX File.
Viewing Memory

Inferred Memory

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

entity ram is
  generic(
    address_width : integer := 4;
    data_width : integer := 8
  );
  port(
    clock : in std_logic;
    data : in std_logic_vector(data_width - 1 downto 0);
    write_address : in std_logic_vector(address_width - 1 downto 0);
    read_address : in std_logic_vector(address_width - 1 downto 0);
    we : in std_logic;
    q : out std_logic_vector(data_width - 1 downto 0)
  );
end ram;
```
architecture rtl of ram is
  type ram is array(0 to 2 ** address_width - 1) of
  std_logic_vector(data_width - 1 downto 0);
  signal ram_block : ram;
begin
  process (clock)
  begin
    if (clock'event and clock = '1') then
      if (we = '1') then
        ram_block(to_integer(unsigned(write_address))) <= data;
      end if;
      q <= ram_block(to_integer(unsigned(read_address)));
    end if;
  end process;
end rtl;