Memory Synthesis

- **Approaches:**
  - Random logic using flip-flops or latches
  - Register files in datapaths
  - RAM standard components
  - RAM compilers

- **Computer register files are often just multi-port RAMs**
  - ARM CPU: 32-bit registers R0-R15 => 16 x 32 RAM
  - MIPS CPU: 32-bit registers R0-R31 => 32 x 32 RAM

- **Communications systems often use dual-port RAMs as transmit/receive buffers**
  - FIFO (first-in, first-out RAM)
Technology-independent RAM Models

-- N x K RAM is 2-dimensional array of N K-bit words

library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_numeric_std.all;

entity RAM is
  generic (K: integer:=8; -- number of bits per word
            W: integer:=8); -- number of address bits; N = 2^W
  port ( 
    WR: in std_logic; -- active high write enable
    ADDR: in std_logic_vector (W-1 downto 0); -- RAM address
    DIN: in std_logic_vector (K-1 downto 0); -- write data
    DOUT: out std_logic_vector (K-1 downto 0)); -- read data
end entity RAM;
architecture RAMBEHAVIOR of RAM is
    subtype WORD is std_logic_vector (K-1 downto 0);    -- define size of WORD
    type MEMORY is array (0 to 2**W-1) of WORD;        -- define size of MEMORY
    signal RAM256: MEMORY;                             -- define RAM256 as signal of type MEMORY
begin
    process (WR, DIN, ADDR)
        variable RAM_ADDR_IN: natural range 0 to 2**W-1; -- translate address to integer
        begin
            RAM_ADDR_IN := to_integer(UNSIGNED(ADDR));  -- convert address to integer
            if (WR='1') then -- write operation to RAM
                RAM256 (RAM_ADDR_IN) <= DIN ;
            end if;
            DOUT <= RAM256 (RAM_ADDR_IN); -- always does read operation
        end process;
    end process;
end architecture RAMBEHAVIOR;
Initialize RAM at start of simulation

process (WR, DIN, ADDR)
    variable RAM_ADDR_IN: natural range 0 to 2**W-1; -- to translate address to integer
    variable STARTUP: boolean := true; -- temp variable for initialization
begin

if (STARTUP = true) then -- for initialization of RAM during start of simulation
    RAM256 <= (0 => "00000101", -- initializes first 4 locations in RAM
               1 => "00110100", -- to specific values
               2 => "00000110", -- all other locations in RAM are
               3 => "00011000", -- initialized to all 0s
               others => "00000000");
    DOUT <= "XXXXXXXX"; -- force undefined logic values on RAM output
    STARTUP := false; -- now this portion of process will only execute once

else
    -- “Normal” RAM operations
RAM with bidirectional data bus

**TABLE 8-7: Truth Table for Static RAM**

<table>
<thead>
<tr>
<th>CS</th>
<th>OE</th>
<th>WE</th>
<th>Mode</th>
<th>I/O pins</th>
</tr>
</thead>
<tbody>
<tr>
<td>H</td>
<td>X</td>
<td>X</td>
<td>not selected</td>
<td>high-Z</td>
</tr>
<tr>
<td>L</td>
<td>H</td>
<td>H</td>
<td>output disabled</td>
<td>high-Z</td>
</tr>
<tr>
<td>L</td>
<td>L</td>
<td>H</td>
<td>read</td>
<td>data out</td>
</tr>
<tr>
<td>L</td>
<td>X</td>
<td>L</td>
<td>write</td>
<td>data in</td>
</tr>
</tbody>
</table>
-- Simple memory model
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;

entity RAM6116 is
  port(Cs_b, We_b, Oe_b: in std_logic;
       Address: in unsigned(7 downto 0);
       IO: inout unsigned(7 downto 0));
end RAM6116;

architecture simple_ram of RAM6116 is
  type RAMtype is array(0 to 255) of unsigned(7 downto 0);
  signal RAM1: RAMtype := (others => (others =>'0'));
  -- Initialize all bits to ‘0’
begin
  IO <= "ZZZZZZZZZ" when Cs_b = '1' or We_b = '0' or Oe_b = '1'
   else RAM1(to_integer(Address));  -- read from RAM
process(We_b, Cs_b)
begin
  if Cs_b = '0' and rising_edge(We_b) then  -- rising-edge of We_b
    RAM1(to_integer(Address'delayed)) <= IO;  -- write
  end if;
end process;
end simple_ram;
Synthesizing RAM

- Previous model synthesizes to 1736 Spartan 3 LUTs (16 bits per LUT), but could easily fit into a single “block RAM”

Spartan 3 block RAM = 18K bits

Can instantiate Xilinx block RAM model
library IEEE;
use IEEE.numeric_bit.all;

entity Memory is
  port(Address: in unsigned(6 downto 0);
      CLK, MemWrite: in bit;
      Data_In: in unsigned(31 downto 0);
      Data_Out: out unsigned(31 downto 0));
end Memory;

architecture Behavioral of Memory is
type RAM is array (0 to 127) of unsigned(31 downto 0);
signal DataMEM: RAM; -- no initial values
begin
  process(CLK)
  begin
    if CLK'event and CLK = '1' then
      if MemWrite = '1' then
        DataMEM(to_integer(Address)) <= Data_In; -- Synchronous Write
      end if;
    end if;
  end process;

  Data_Out <= DataMEM(to_integer(Address)); -- Asynchronous Read
end Behavioral;
library IEEE;
use IEEE.numeric_bit.all;

entity Memory is
    port(Address: in unsigned(6 downto 0);
        CLK, MemWrite: in bit;
        Data_In: in unsigned(31 downto 0);
        Data_Out: out unsigned(31 downto 0));
end Memory;

architecture Behavioral of Memory is
type RAM is array (0 to 127) of unsigned(31 downto 0);
signal DataMEM: RAM; -- no initial values
begin
    process(CLK)
    begin
        if CLK'event and CLK = '1' then
            if MemWrite = '1' then
                DataMEM(to_integer(Address)) <= Data_In; -- Synchronous Write
            end if;
            Data_Out <= DataMEM(to_integer(Address)); -- Synchronous Read
        end if;
    end process;
end Behavioral;