BUILDING A ROM WITH XILINX WEBPACK

1. Open your project.
2. Project -> New Source.
   a. Select VHDL module
   b. File Name: “InstROM”
   c. Select Next
   d. Port Name “InstAddress”, in, MSB=7, LSB=0 (for a 256-entry ROM)
   e. Port Name “InstOut”, out, MSB=7, LSB=0
   f. Select Next
   g. Select Finish
3. Edit the VHDL code in InstROM.vhd, so it looks like this:

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity InstROM is
  Port ( InstAddress : in std_logic_vector(7 downto 0); InstOut : out std_logic_vector(7 downto 0));
end InstROM;

architecture Behavioral of InstROM is
  type ROM_Array is array (0 to 255) of std_logic_vector(7 downto 0);
  constant Content: ROM_Array := (
    B"00100100",
    B"00000001",
    B"00010010",
    B"11111111",
    B"00000011",
    others => B"00000000" );

begin
  InstOut <= Content(conv_integer(InstAddress));
end Behavioral;
```

This is for a 256-entry ROM (8-bit PC), 8 bits wide (8 bit instructions). The values in red are those that would change for a different sized ROM. Notice that all the inputs are binary. You should also be able to specify them in hex (see RAM code). Your ROM contents will be different (and much longer!). This puts hex 0x24 (binary 00100100) in location 0, 0x101 in location 1, 0x012 in location 2, etc.

4. In “Sources” Window, click InstROM.vhd.
5. In “Proccesses” window, click “Check Syntax” (in “Synthesize XST”) – correct errors.
6. In “Proccesses” window, click “Create Schematic Symbol” (in “Design Entry Utilities”)
7. Open Schematic in which you want to include your ROM.
8. “instrom” should now be added to your library of symbols. You may want to edit its shape to make it look nicer.

9. Test it alone to make sure it is reading the right values, with the bits in the order you expect, etc.

BUILDING A RAM WITH INITIALIZED VALUES WITH XILINX WEBPACK

- Open your project.
- Project -> New Source.
- Select VHDL module
  - File Name: “DataRAM”
  - Select Next
  - Port Name “DataAddress”, in, MSB=7, LSB=0 (for a 256-entry RAM)
  - Port Name “clk”, in
  - Port Name “ReadMem”, in
  - Port Name “WriteMem”, in
  - Port Name “DataIn”, in, MSB=7, LSB=0
  - Port Name “DataOut”, out, MSB=7, LSB=0
  - Select Next
  - Select Finish
- Edit the VHDL code DataRAM.vhd, so it looks like this:

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity DataRAM is
    Port ( DataAddress : in std_logic_vector(7 downto 0);
           clk : in std_logic;
           ReadMem : in std_logic;
           WriteMem : in std_logic;
           DataIn : in std_logic_vector (7 downto 0);
           DataOut : out std_logic_vector(7 downto 0));
end DataRAM;

architecture Behavioral of DataRAM is

type ram_type is array (0 to 255) of
    std_logic_vector(7 downto 0);

signal tmp_ram: ram_type := (
    1 => X"9e",
    32 => X"00",
    33 => X"00",
    34 => X"80",
    35 => X"08",
    36 => X"5c",
    37 => X"01",
    38 => X"04",
    39 => X"0f",
    40 => X"71",
);```
41 => X"64",
42 => X"2a",
43 => X"2b",
44 => X"2c",
45 => X"2d",
46 => X"2e",
47 => X"2f",
48 => X"50",
49 => X"25",
50 => X"7a",
51 => X"f0",
52 => X"00",
53 => X"3c",
54 => X"a8",
55 => X"90",
56 => X"6e",
57 => X"6f",
58 => X"30",
59 => X"25",
60 => X"1a",
61 => X"c8",
62 => X"00",
63 => X"13",
64 => X"10",
65 => X"11",
66 => X"12",
67 => X"13",
68 => X"14",
69 => X"15",
70 => X"16",
71 => X"07",
72 => X"08",
73 => X"09",
74 => X"0a",
75 => X"00",
76 => X"1f",
77 => X"12",
78 => X"03",
79 => X"01",
80 => X"19",
81 => X"17",
82 => X"01",
83 => X"0a",
84 => X"04",
85 => X"05",
86 => X"06",
87 => X"07",
88 => X"08",
89 => X"09",
90 => X"0a",
91 => X"1b",
92 => X"1c",
93 => X"1d",
94 => X"1e",

others => X"07");
begin
  process(ReadMem, DataAddress)
  begin
    if ReadMem='1' then
      DataOut <= tmp_ram(conv_integer(DataAddress));
    else
      DataOut <= (DataOut'range => 'Z');
    end if;
  end process;

  process(clk, WriteMem)
  begin
    if (clk'event and clk='1') then
      if WriteMem='1' then
        tmp_ram(conv_integer(DataAddress)) <= DataIn;
      end if;
    end if;
  end process;
end Behavioral;

This is for a 256-entry RAM (8-bit address). Your initial RAM contents will be exactly the same as specified here. Your code will change slightly if you have a larger RAM, but these initial values will be the same.

- In “Sources” window, click DataRAM.vhd
- In “Proccesses” window, click “Check Syntax” (in “Synthesize XST”) – correct errors.
- In “Proccesses” window, click “Create Schematic Symbol” (in “Design Entry Utilities”)
- Open Schematic in which you want to include your RAM.
- “dataram” should now be added to your library of symbols. You may want to edit its shape to make it look nicer.
- Test it alone to make sure it is reading the right values, with the bits in the order you expect, etc.