Creating and Simulating Memories in MaxPlus2 and ModelSim

Author: Frank Honoré (honore@mtl.mit.edu), Spring 2004

Altera's Flex10K family of FPGA's supports the efficient implementation of memory blocks such as RAM through the use of Embedded Array Blocks (EABs) in the architecture. To use this feature, a RAM can be defined using the Megafunction Wizard by following the steps outlined below. This example will construct a 256 location by 8-bit wide RAM and simulate it using Modelsim. Detailed timing information for EAB memories is described in the Flex10K datasheet on pages 61-69. (On the lab PC, S:\6.111\AlteraFlex10kDatasheet.pdf)

1. Start MAX+plusII
2. Create the memory model using the MegaWizard
   File -> MegaWizard Plugin Manager
3. Select Create new custom megafunction
4. In the lefthand box, double-click 'storage' and select LPM_RAM_DQ.
5. Check Verilog HDL for output file choice
6. Choose a name and path for the output files (eg. U:\ram\ramdq_8x8)
7. Click Next
8. For this example, create a combinational (non-registered) memory block that has 8 bit input and 8 bit address.
   Uncheck all ports for "Which ports should be registered?"
9. Select 8 for width of 'q' output bus.
10. Select 8 for width of address input bus.
11. Click finish (or next 2 more times and then finish to see what files will be created).

Several files will be created. The .v file is the one of interest that defines the verilog module for the memory. If you look in this file, you'll see it instantiates a parameterized lpm_ram component. Next, we will compile this file in MAX+plusII to generate a more detailed gate-level description of the memory block.

1. File -> Open (browse to ramdq_8x8.v)
2. File -> Project -> Set project to current file
3. Max+plus II -> Compiler
4. Interfaces -> Select Verilog Netlist Writer (this will add a box to the compiler window)
5. Click Start in the compiler window

You now have a ramdq_8x8.vo file in your project directory. This file contains timing information for the memory module using Flex10K primitives with timing parameters.

To simulate in Modelsim, you will need to add an Altera-specific library containing models for the lpm memory that we generated above. For faster simulation, we can simulate the functionality of the memory without timing information (behavioral mode). Or we can simulate with full timing information for accurate results (gate-level mode).

There is a testbench for this example available at S:\6.111\ram Example\test_ram.v. Copy this file to your project directory.

Start Modelsim and create a new project and add ramdq_8x8.v and test_ram.v and compile (see Modelsim tutorial).

Use these commands to map the needed libraries: (See footnote if you are not working from a Digital Lab PC)[1]

```
Modelsim > vmap 220model S:/6.111/Modeltech/altera/verilog/220model
Modelsim > vmap alt_vtl S:/6.111/Modeltech/altera/verilog/alt_vtl
```

Now for functional simulation (without detailed timing) start the simulation with:

```
Modelsim >  vsim -L 220model test_ram
```

Add a waveform display (View -> Wave) and drag the signals to the wave window. Run the simulation, Simulate -> Run -> Run –All which should run for about 600ns and display 3 memory write accesses. Notice that memory output values change instantaneously with the write enable signal going high.

Now let's simulate using the detailed timing model. Replace ramdq_8x8.v with ramdq_8x8.vo and recompile. End previous simulation (quit –sim). Now simulate the design using the alt_vtl library, which contains definitions for primitives instantiated in the .vo file.

```
Modelsim > vsim –L alt_vtl test_ram
```

Let's view the memory accesses one at time so we can see the impact of the timing details. (Be sure to add signals back to the waveform display.)

```
Vsim > run 300
```

As expected, the output q displays the value '5' after a delay.

```
Vsim > run 100
```

Notice that the output is '0' when we wanted to write the value '3'. The width of the write enable pulse is too short in this case.

```
Vsim > run 100
```

Now, the memory model warns us that we have violated the setup time of the address by changing it at the same time as the write enable pulse goes high. Even though we get the correct result in simulation, there is no guarantee the hardware will produce correct results. As an exercise, change the test_ram.v file to correct these timing errors.

---

[1] If not on a Lab PC, the file alt_max2.vo that is also generated when compiling the memory can be included in the Modelsim project directory for timing simulations. Then, simply run 'vsim test_ram'.