DLX CPY and Matrix Multiplier Simulation Uisng Seamless

Introduction

This tutorial uses the DLX processor with the some standard devices such as memory, and virtual screen (I/O) to setup an environment for HW/SW co-simulation. The tutorial will assume that the readers are exposed to HDL (verilog) and design of digital systems in general.

This tutorial will describe the process to add a separate hardware unit to the DLX architectures shown in Fig 1-1. Since the DLX processor described here do not have any multiplication unit, to perform a matrix multiplication, the system needs to do such routine by software. Therefore the purpose of this tutorial is to accelerate the computation of matrix multiplication, and also to demonstrate working environment of Seamless tools.

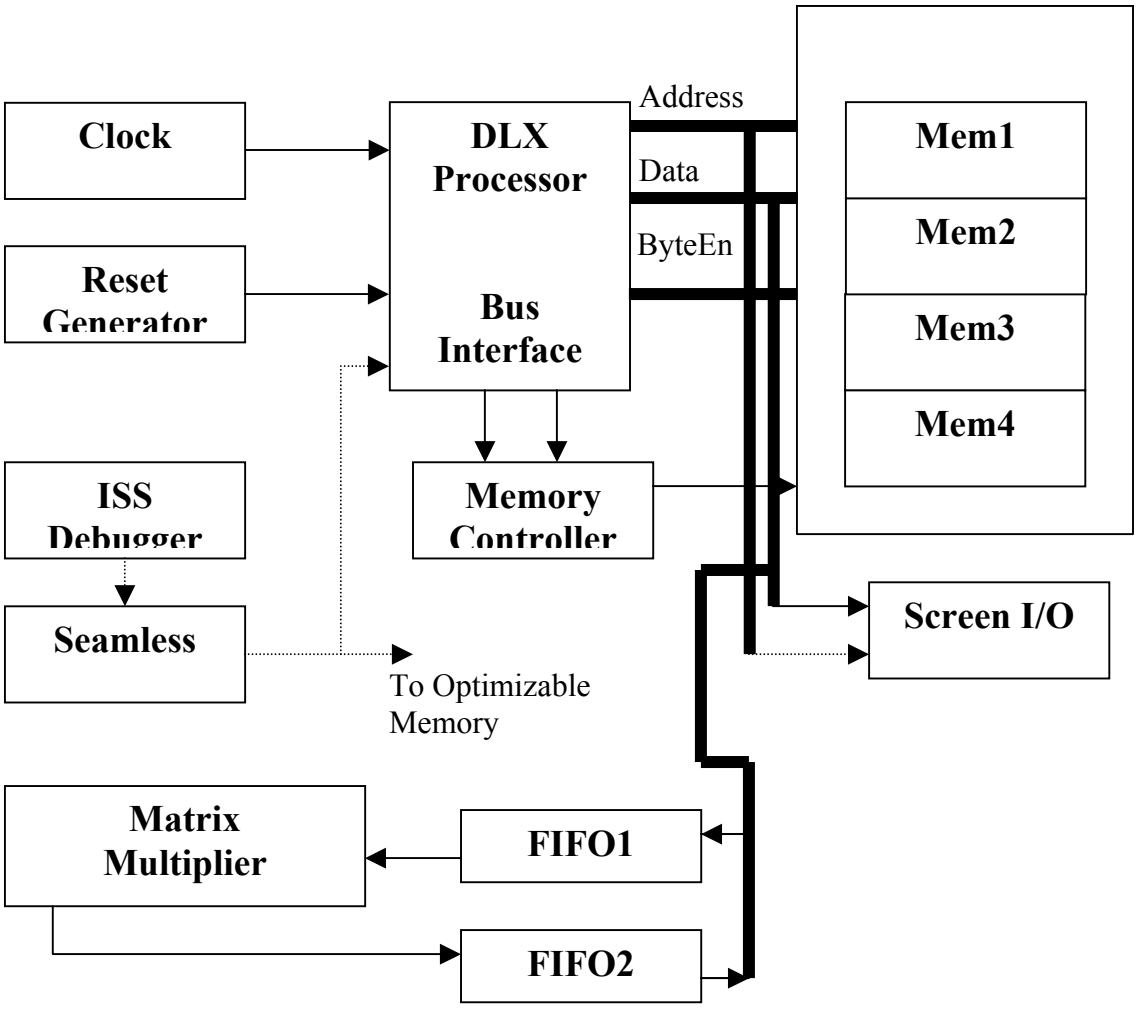

Figure 1-1 DLX Architecture with Matrix Multiplier

Setting up Seamless CVE

Before doing anything, it is important to see if your station is able to run Seamless CVE program. Seamless CVE is very complicated software, and this tutorial just gives you the flavor of it.

1) Log in to your workstation.
2) SSH to genesis
3) Make a separate directory to work in
4) Type "cadenceenv" in the prompt to set up the Cadence environment variables
5) Type "seamlessenv" to setup the Seamless environment variables
6) Type "source var.env" to setup additional environment variables.
7) Type "cve" to run the Seamless CVE software

If you are not able to run the CVE software, please talk to either the systems administrator in your department or contact the author. Also, the source for the last environment files should be given to you, and it should be placed in your working directory. The other two command ("cadenceenv, seamlessenv") should be executable from genesis server.

Setting up the Design Files

Before one can perform the HW/SW co-simulation with Seamless CVE, the design files needs to be set correctly. Design files can be divided in to two categories, hardware descriptive language (HDL), and software (assembly). For Seamless to be able to communicate between the hardware process and software process, each process needs to be compiled and linked to the appropriate software.

First of all lets compile and build the hardware part of the design (DLX processor and its architecture). To make life simpler, a verilog compiling script file can be used, and the source for this script files can be found in appendix.

1) Copy this script files "build_design" into your working directory
2) Type "build_design"

This should compile all of your design files so that it can be simulated with "ncsim" software by Cadence. Again all the design files can be found from appendix, and make sure that all the design files, and other user files are all located in your working directory, and that the script file "build_design" is updated for correct location.

Now, we will compile the software part of the design.

1) type "$CVE_HOME/isms/dlxtools/bin/dlx-gcc –g ~/working_directory/crt_tutorial.S –c –I. –o crt_tutorial.o" to compile the software.

2) Type "$CVE_HOME/isms/dlxtools/bin/dlx-ld crt_tutorial.o –Ttext 0x0000 –o crt_tutorial.elf" to link the software.
3) Copy the debugger "include" file to your tutorial directory: "cp $CVE_HOME/example/tutorial/common/sw/tutorial_int.inc

Now we are ready to start the co-simulation with Seamless CVE.

Invoking the hardware simulator

Lets start the Seamless CVE by following the previous method described in section "Setting up Seamless CVE".

1) When the Seamless CVE is running you will see the menu on the top left of your program. Find Setup Menu, and click Hardware simulator invocation.
2) This will bring up the Setup Logic Simulator window.
3) Type "$CDS_INST_DIR/tools/bin/ncsim –gui –CDSLIB cds.lib work.DLX_design:snap" in the invocation box
4) Make sure that the "Run hardware simulator in a terminal window" button is off
5) Click "OK" to close the window

Invoking the Software simulator

Lets now setup the software simulator process in the Seamless CVE environment.

1) Find the Setup Menu, and click software Invocation setup button.
2) Enter the following simulator invocation string under "Invocation:" in the setup software simulator dialog box: "$CVE_HOME/isms/bin/mdb_dlx crt_tutorial.elf –I tutorial_int.inc"
3) Click the button next to "Run software simulator in a terminal window" in the setup software simulator dialog box. You do this because the software simulator is not GUI driven.
4) Click the OK button. Notice that the software simulation icon in the session window is no longer highlighted, indicating that setup is done.

Setting up the Memory

You need to map the Seamless optimizable memory instances into the processor's address space. Do this by performing the following steps:

1) Click the Map Memory Instances icon in the CVE Session window. This activates the Memory Map dialog box. Each of the four memory instances in the memory module is 1Mbyte long and 8 bits wide. These instances will be mapped across the processor's 32 bit data bus such that each instance occupies 8 bits of the bus.

2) Select the mem0 memory instance by clicking on that instance's entry in the Unmapped Memories list. The instance entry is highlighted when selected. Notice that the base address and bit position entries are both 0.
3) Click the Map button to map mem0 into the processor address space. The entry for mem0 appears in the Memory Map list, as shown below.
4) Select the mem1 memory instance by clicking on that instance's entry in the Unmapped Memories list.
5) Specify the bit position for the mem1 instance by clicking on the up arrow button to increment the value in the box to 8. This maps the mem1 instance into bit 8 through 15 of the processor's 32-bit word.
6) Click the Map button to map mem1 into the processor address space. The entry for mem1 appears in the memory map list with a bit position of 8.
7) Map the mem2 and mem3 instances into bit positions 16 and 24, respectively.
8) Select the crt entry in the Unmapped Memories list. Enter a base address of 3FFFFFF0 and bit position of 0 then click the Map buttons to map the instance to processor address space.
9) Select the FIFO1 entry in the Unmapped Memories list. Enter a base address of 4FFFFFF0 and bit position of 0 then click the Map buttons to map the instance to processor address space.
10) Select the FIFO2 entry in the Unmapped Memories list. Enter a base address of 5FFFFFF0 and bit position of 0 then click the Map buttons to map the instance to processor address space.
11) Click the close button to dismiss the Memory Map dialog box.

The next step is to setup the memory-access ranges for the process as follows:

1) Click the Memory Access Ranges icon in the Seamless CVE Session window. The Memory Ranges dialog box appears. The Address Ranges list box contains four entries. Notice that the first entry, for address 0 through 3FFFFF, is the address range of the memory module in the hardware design. Since theses instances are seamless memory models, this address rage is optimizable. The icon under "access" indicates this.
2) Select the first entry (0 to 3FFFFF) in the address ranges list box. The start and end addresses appear in the start address and end address entry boxes. Notice the icon under the access heading in the list box and the corresponding icon on the access tab in the lower portion of the memory ranges dialog box. Optimized access to this range has not been enabled yet, meaning that all accesses to this memory range generate hardware bus cycles.
3) Create a label for the optimizable address range as follows:
    a) Click the Label tab.
    b) Enter a label of "optimizable" in the range label box, then press return or click the apply button. The label appears in the optimizable entry in the address ranges list box.
4) Setup wait-state values for the optimizable memory:
    a) Click the wait states tab

b) Enter values of 2 in the read and write boxes under "initial," then press return or click the apply button. (Leave the burst boxes blank.) The wait-state values appear in the address ranges list box.

5) Select the screen entry (3FFFFFF0-3FFFFFF3), which is next in the list. Although the screen is actually based on an optimizable register memory (hence the "optimizable" icon under the access column), we will pretend it is hardware-only for the present. Enter a label of "screen" for this range then click apply.
6) Setup the FIFO1 and FIFO2 the Same
7) Setup the illegal address space.
8) Click the close button to close the window. Notice that the memory access ranges icon in the session window is no longer highlighted, indicating that this step has been done.

Saving the configuration

Save the current configuration by performing the following step:

1) Click the save configuration icon in the Seamless CVE Session window, or File menu and Save as item.
2) In the save as dialog box, enter a configuration filename.
3) Click OK to save the configuration.

Starting the Co-simulation

1) Click the Run button in the Seamless CVE main window. The button changes to yellow and remains pressed, indicating that co-simulation can start. In addition, a terminal window appears showing a software simulator startup message: CVE NOTE: waiting to connect to hardware process …
2) Go to the NC verilog simulator's main window, activate the Navigator, and setup a wave form display window. Display the signals at the top level of the design, as well as any other signals you are interested to view.
3) Click the run button on NC verilog main window. This starts the hardware simulation, but the simulation cannot run until its software begins to execute. Notice that the run button turns green, indicating that co-simulation has begun. The Seamless CVE models in the design cause Seamless CVE to start up the software simulator. The software simulator terminal window displays a series of messages followed by this prompt: (mdb)
4) Enter "run" at the (mdb) prompt. The screen window appears and displays the matrix multiplication question with its answer.

At this point you have just finished the basic component of starting the co-simulation. For more advanced information and topics please refer to other Seamless CVE documentations.