CS/ECE 561: Hardware/Software Design of Embedded Systems
Fall 2016

Instructions for Working with SystemC

(File created by Prof. Sudeep Pasricha)

1. Setting up the SystemC environment

Given below are the instructions for accessing and setting up your environment for running SystemC on the Linux Machines in the ECE department (CS students who do not have access to these machines should contact ENS for an account)

(i) (Recommended) The SystemC tool set is installed only on the Linux machines in the ECE department. All the Linux machines available for access are listed at:

http://www.engr.colostate.edu/ens/tools/serverstatus/

You can work on any one of the 64-bit machines. Please make sure you are working on the linux machines and not the sun machines. You do NOT need Xserver for running SystemC, so you can work remotely by using clients like "putty".

(ii) You can also install SystemC locally on your machines (latest version SystemC 2.3.1):

http://www.accellera.org/downloads/standards

You will need to create a (free) account before you can download SystemC. The INSTALL file in the systemc root directory has straightforward instructions on how to install systemc.
Before starting the installation it is recommended to run the following command on your local machine

```
%  sudo apt-get install build-essential
```

After installation, you have to set the LD_LIBRARY_PATH environmental variable to use systemc library. As an example, the following command sets the path to the SystemC library for [ba]sh shells.

`% export LD_LIBRARY_PATH=<path to systemc installation>/lib-linux64` (or just `linux` for 32 bit systems)

The instructions below assume SystemC 2.2 which is the version installed on the ECE servers. Replace references of `systemc-2.2.0 with systemc-2.3.1` if you are working with an installation on your local machine.

(iii) Once you have logged in, set the 'SYSTEMC' environment variable to the directory of the SystemC installation (located at: /usr/local/systemc-2.2.0) Depending on your shell (default in the ECE cluster is the C shell csh but you can find out what you are running through the

‘SHELL’ environment variable, i.e. by typing echo $SHELL), setting the ‘SYSTEMC’ variable is done as follows for [t]csh:

```
% setenv SYSTEMC /usr/local/systemc-2.2.0
```

or, for [ba]sh:

```
% export SYSTEMC=/usr/local/systemc-2.2.0
```

(iv) Once the environment variable is set, you can access the SystemC installation by referring to the ‘SYSTEMC’ variable. The standard GNU C++ compiler ‘g++’ is then used to compile SystemC code and link it against the SystemC libraries:

```
% g++ -I$SYSTEMC/include –L$SYSTEMC/lib-linux64 –lsystemc –lm <source>
```

(v) It is strongly recommended that you create a ‘Makefile’ for compiling your SystemC sources first to object (.o) files and then linking everything together into a final simulation executable. An example of a corresponding ‘Makefile’ that you can use as a starting point is available at: $SYSTEMC/examples/sysc/Makefile.defs.(`If you have installed the systemc-2.3.1, Makefile.defs might be unavailable. So, visit canvas for the file)`

Assuming you are working on a 64 bit ECE machine, adjust this template to your needs by setting the SYSTEMC variable to /usr/local/systemc-2.2.0, the TARGET_ARCH variable to linux64, the MODULE variable to the name of your executable to be generated, and the OBJS variable to the list of files (separated by a single space) that are part of your design with the .cpp being replaced by .o for each file

2. Simple FIFO example

The procedure below walks you through a simple FIFO SystemC example:

(i) Make sure that your SystemC environment is setup so that the environment variable 'SYSTEMC' is set to the installation directory. The example that we are going to use is the simple FIFO producer/consumer example that is part of the standard SystemC installation.

(ii) First create a work directory for the SystemC files.

```
% mkdir sysc_work
```

Change the current working directory to the newly created directory.

```
% cd sysc_work/
```

Copy the SystemC source files for the simple_fifo program from the examples directory into the newly created directory:

```
% cp $SYSTEMC/examples/sysc/simple_fifo/*.cpp .
```

Copy the Makefile template from the examples directory:

```
% cp $SYSTEMC/examples/sysc/Makefile.defs .
% ls
Makefile.defs simple_fifo.cpp
```

(iii) Now you must modify the Makefile.defs file, as described in 1.(v) above. The first few lines of your Makefile.defs should be edited to look like this (remember to not put any spaces after the last character at the end of a line while editing the Makefile, as it can cause errors):

```
## Variable that points to SystemC installation path
SYSTEMC = /usr/local/systemc-2.3.1

TARGET_ARCH = linux64  (or linux for 32 bit machines)

CC = g++

INCDIR = -I. -I.. -I$(SYSTEMC)/include
LIBDIR = -L. -L.. -L$(SYSTEMC)/lib-$(TARGET_ARCH)

LIBS = -lsystemc -lm $(EXTRA_LIBS)

## Change the OBJ variable to reflect your source files
MODULE = simple_fifo
OBJS = simple_fifo.o

EXE = $(MODULE).x

<… the part below this remains unchanged>
```

(iv)You are now ready to compile the example with the Makefile.defs:

```
% make –f Makefile.defs
```

Or alternatively, you can rename Makefile.defs to Makefile and compile the example:

```
% mv Makefile.defs Makefile
% make
```

After a successful compilation, this is what you should have:

```
% ls
Makefile simple_fifo.x simple_fifo.cpp simple_fifo.o
```

Finally run the generated binary to simulate the example design:

```
% ./simple_fifo.x
```

If you make any changes to simple_fifo.cpp, make sure to clean the results of the previous compilation with the following command, before compiling again:

```
% make clean
```