Agilent Technologies

Advanced Design System 2002

ISS Cosimulation

February 2002

Notice

The information contained in this document is subject to change without notice.

Agilent Technologies makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Agilent Technologies shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance, or use of this material.

Warranty

A copy of the specific warranty terms that apply to this software product is available upon request from your Agilent Technologies representative.

Restricted Rights Legend

Use, duplication or disclosure by the U. S. Government is subject to restrictions as set forth in subparagraph (c) (1) (ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013 for DoD agencies, and subparagraphs (c) (1) and (c) (2) of the Commercial Computer Software Restricted Rights clause at FAR 52.227-19 for other agencies.

Agilent Technologies
395 Page Mill Road
Palo Alto, CA 94304 U.S.A.

Contents

Chapter 1: ISS Cosimulation for Texas Instruments DSP

Advanced Design System's Instruction Set Simulator (ISS) cosimulation software enables you to perform hardware/software co-design with Texas Instruments digital signal processing (DSP) chips. The software allows you to link Advanced Design System (ADS) and Texas Instruments Code Composer Studio™ to verify that your DSP code works correctly in the entire system with other digital/analog hardware components.

DSP system development involves both high-level algorithm development and low-level assembly development. Integrating the low-level assembly development in a system-level simulation tool can shorten the development cycle and simplify the simulation and verification of the code. Advanced Design System's Instruction Set Simulator cosimulation software allows you to simulate DSP assembly code along with ADS high-level components. By cosimulating designs specified in the DSP code, you can easily incorporate existing DSP code intellectual property into your new ADS designs.

The TI DSP ISS Cosimulation feature has been implemented to cosimulate with Texas Instruments (TI) DSPs using TI's Code Composer Studio (CCStudio). The cosimulation is seamless and you can interact with both the ADS and TI CCStudio graphical user interfaces (GUIs) for data analysis and visualization during cosimulation.

Basic Requirements

ISS Cosimulation for Texas Instruments DSP is an optional feature of Advanced Design System and is supported on PC/Windows systems *only*. For PC/Windows installation requirements, refer to the *Installation on PC Systems* manual. To use it, you need to have the following:

- Agilent Ptolemy simulator
- TI Code Composer Studio 1.2 or higher along with the desired TI DSP simulator:

  TMS320C6x

  TMS320C54x

Cosimulation Overview

The ADS component provided to perform the cosimulation is called *TICCStudioCosim* and can be found in the ISS Blocks palette or library. The component has one multi-input port and one multi-output port. More than one port can be connected to the input port using the BusMerge component. More than one port can be connected to the output port using the BusSplit component. For cosimulation, each port is mapped to a global variable in the DSP code you specify in the list of component parameters. If the variable is an array data type, the port corresponding to the variable is set to be multirate. The rate depends on the size specified for the variable.

The communication between ADS and TI DSP code is through the global data memory or global variables defined in the TI DSP code. During cosimulation, the global variables you specified are updated by values on the input ports. The DSP code is executed on the target DSP using TI CCStudio. The global variables you specified are read by ADS and are used to update the output ports. This process is continued until the ADS simulation is completed. A break point is used to synchronize the cosimulation.

One single program, with different functions operating on global variables, is used by all the instances of the TICCStudioCosim component in a design. The program or the DSP code used while performing the cosimulation must be in cosimulation format. You must build the code as specified in the section "Building Code In Cosimulation Format" on page 1-2 before cosimulation. You must also configure the TI CCStudio environment to use a single appropriate target TI DSP before cosimulation. Only one target DSP must be specified in the configuration. If more than one target DSP is specified in the configuration, the cosimulation will not proceed.

The cosimulation can invoke TI CCStudio interactively or in the background processing mode. TI DSP code debugging and data visualization is enhanced by invoking the TI CCStudio in the interactive mode.

Building Code In Cosimulation Format

To build the code in cosimulation format, do the following:

1. Using TI Code Composer Studio's Setup tool, configure the CCStudio to use only one DSP target of interest. Select the right endianness to use. In the case of some DSPs, memory mapping has to be set inside CCStudio. For example, the C54x family of TI DSPs require proper memory mapping for simulation. When

you use CCStudio with TI C54x DSPs, the default memory mapping configuration selected from the CCStudio's setup tool will work in most cases.

2. Create a project in TI Code Composer Studio that contains the code or all the functions that you want to cosimulate with. For help on creating a project refer to the Code Composer Studio *User Reference* manual. Functions you want to cosimulate with must have the footprint or declaration of the format "void function()", that is, it must take no parameter nor return anything. The function operates on the global data that has been updated by ADS. If you have a function with a different footprint or declaration, they must be wrapped with a function of the specified declaration type. Each *simulation* function could have a *begin* function and a *wrap-up* function of the same footprint, which will be invoked before and after the cosimulation, respectively.

Note Do not define or use symbols named *ADS_TICCStudio_Cosim*, *ADS_TICCStudio_Cosim_UserFunc* or *ADS_TICCStudio_Cosim_Done* in your DSP code because these are used by the cosimulation framework.

---

3. Copy the cosimulation-specific file from the ADS installation area

   ```
   %HPEESOF_DIR%\hptolemy\lib\ticcscosim\ADS_TICCStudio_Cosim.c
   ```

   to an area where you have write permission. This file contains the "void main()" function defined along with other code required for cosimulation. Do not edit this file. Add this file to your project in the Code Composer Studio. Since this file contains the "main()" function, your code must not have the "main()" function defined.

4. Compile and link the above file, along with the code that has the functions that you want to cosimulate with, in the Code Composer Studio. The generated executable file is now in the cosimulation format.

CCStudio Build Options

For cosimulation, the optimization level can only be up to the function level (i.e., -O2). File level optimization (i.e., -O3) will not work and cosimulation will hang.

The optimization level can be selected as follows:

- From TIC54x CCStudio, choose *Project* > *Options* > *Compiler* > *Optimizer* > *Level*.
- From TIC6x CCStudio, choose *Project* > *Options* > *Compiler* > Basic > *Opt Level*.

When changing endianness for the C6x family of TI DSPs, remember to rebuild the code. While rebuilding the code, you must set the endianness for the compiler and assembler, as follows:

- Compiler *Project* > *Options* > *Compiler* > *Advanced* > *Endianness.*
- Assembler *Project* > *Options* > *Assembler* > *Endianness.*

Also, choose the right run time library (for big endian, the library name ends in *e.lib*, for example, *rts6201e.lib*).

Cosimulation Related Issues

If there are any simulation errors and you are not able to restart the simulation or rebuild the code in CCStudio, then you must manually kill the processes named “CC_APP.EXE” and “HPEESOFSIM.EXE” using the Task Manager in Windows NT/2000 or by pressing Ctrl+Alt+Del and then selecting the process to kill on Windows 98.

After a successful cosimulation, if you want to use the CCStudio user interface, you must stop and release the ADS simulator. From the Schematic window, choose Simulate > Stop and Release Simulator.

DSP Global Variable Specification Syntax

More than one variable of different types can be specified for inputs (InputVars), outputs (OutputVars) and parameters (ParamVars). The global variable specification syntax is similar to C declarations. More than one variable of the same type can be grouped together under one type by separating them with commas “, ”. Variables of different types must be separated using semicolons “; ”. For example,

<type>$_1$ <variable list>$_1$;[<type>$_2$ <variable list>$_2$;] ...

The <type> specified is the *TI DSP C code data type* of the variables specified in <variable list>. Only TI DSP C-basic data types such as *char, uchar, short, ushort, enum, int, uint, long, ulong, float, double, ldouble* are supported. The structure data type is not supported. For the structure data type, there must be global variables

corresponding to each element in the structure. You must update the structure using the global variable values that are updated by ADS.

List of variables <variable list> of type <type> can be of scalar or array types. More than one variables of same type must be separated using comma “, ”. The syntax for scalar variables is:

<name>$_1$[=<value>$_1$][, <name>$_2$[=<value>$_2$]] ...

The syntax for array variables is:

<name>$_1$[<size>][={<value>$_{11}$[, <value>$_{12}$]...}][, <name>$_2$[={<value>$_{21}$ [, <value>$_{22}$]...}]] ...

To initialize an array value from a file, you use a “<“ character. All values will be sequentially read from the file depending on the <size> of the array. The syntax to initialize a value using a file is:

< <filename>.

The value specification will be ignored for InputVars and OutputVars parameters, but will be used by the ParamVars parameter.

Valid examples:

int b, c[5], d[10]={3,5,78}, e=5; float f[13]={1.4, 4.67, 3}, g=0.0; float taps[100]={0.5, <fir.txt, 0.7, 0.5};

Invalid examples:

int b c[5] d[10]={3,5,78} e=5; ==> Variables must be comma separated.

char c=’X’; ==> Value must be an integer or real number.

TICCStudioCosim Component Parameters

Following is a detailed explanation of the parameters for the TICCStudioCosim component:

Program

This parameter is used to specify the TI DSP code to cosimulate with. You must build the TI DSP code in the required format before cosimulation. You must have a set of global variables in the DSP code for communication with ADS.

Same configuration of the TI CCStudio and the target TI DSP used to build the code must be used for cosimulation.

CCStudioGUI

This specifies if the TI CCStudio should be opened interactively or not during the cosimulation. In the interactive mode you can use breakpoints in the DSP code to debug the code or use any other debugging tools provided by TI Code Composer Studio.

Cosimulation is synchronized using breakpoints set at *ADS_TICCStudio_Cosim* and *abort*, so do not set/unset these breakpoints. If you want to break the cosimulation, you must set the breakpoint at some location other than ADS_TICCStudio_Cosim and abort.

In the interactive mode you must initiate the execution of the DSP code after setting all the debugging and verification options. For example press the key F5 (for *Run*)in CCStudio after setting any breakpoints. If the cosimulation stops at any user breakpoint, the cosimulation will wait until you continue the execution of the DSP code past the breakpoint.

In a design that contains Sweep or Optimization components, if you do not set any break points as specified above and once the cosimulation is initiated, the cosimulation will continue without user interaction in the following analyses:

Functions

This is a list of global functions from the TI DSP code specified in the parameter *Program*. There are three functions associated with this parameter:

- **Begin function** is executed once before cosimulation starts.
- **Simulation function** is executed during cosimulation.
- **Wrapup function** is executed after the cosimulation is done.

If you do not have anything to be done either before, during, or after cosimulation, the function specification must be set to “-”. The functions must have the declaration type “void function()”, that is, it must not take any parameters nor return an value. If any of the functions to be cosimulated with are not of the mentioned declaration type, they must be appropriately wrapped inside a function that is of the mentioned declaration type.

Example: In the example project *examples/ticcscosim/isscosim_prj,* the design *addsub.dsn* has only a simulation function. So the Functions parameter has been set to:

```
Functions="- add -"
```

In the same project, the design *viterbi.dsn* has a begin and a simulation function. So the Functions parameter has been set to:

Functions="viterbi_init viterbi -"

In the same project, the design *fir_xdas.dsn* specifies begin, simulation and wrapup functions. So the Functions parameter has been set to:

Functions="fir_cosim_init fir_cosim_apply fir_cosim_exit"

InputsVars

This is a list of global variables declared in the TI DSP code, specified in the parameter Program, which will be updated by the values on the ports connected to the input port during each firing of the TICCStudioCosim component. The specification follows the syntax explained in the section "DSP Global Variable Specification Syntax" on page 1-4. More than one input ports can be connected using a BusMerge component. Each specified variable from left to right corresponds to each port connected to the BusMerge component from bottom to top, respectively.

The port corresponding to an array data type of variable is treated as a multirate port and the rate is dependent on the size specified. For example, if the size specified is 5, which corresponds to a DSP variable of type array and length 5, the rate of the port corresponding to the variable is 5. So, for every firing of the TICCStudioCosim component, 5 values are read on that port and the array corresponding to the variable is filled.

OutputVars

This is a list of global variables declared in the TI DSP code, specified in the parameter *Program*, which will be read from the TI DSP code after a simulation cycle. The specification follows the syntax explained in the section "DSP Global Variable Specification Syntax" on page 1-4. It will update the values on the input ports connected to the output port during each firing of the TICCStudioCosim component. More than one input port can be connected to the output port using a BusSplit component. Each specified variable from left to right corresponds to each port connected to the BusSplit component from bottom to top, respectively.

The port corresponding to an array data type of variable is treated as a multirate port and the rate is dependent on the size specified. For example, if the size specified is 5, which corresponds to a DSP variable of type array and length 5, the rate of the port corresponding to the variable is 5. So, for every firing of the TICCStudioCosim component, 5 values from the array corresponding to the variable are read and the value on the corresponding output port is updated.

ParamVars

This is a list of global variables declared in the TI DSP code, specified in the parameter Program, which will be updated only once at the beginning of the cosimulation. The specification follows the syntax explained in the section "DSP Global Variable Specification Syntax" on page 1-4.

If certain values at the end of the list are missing, the value is assumed to be 0.0.

Examples of ISS Cosimulation for TI DSP

The project *examples/ticcscosim/isscosim_prj* includes a set of three designs for demonstrating TI DSP ISS Cosimulation. The TI CCStudio configuration can use a TI TMS320C6xxx or TI TMS320C54x simulator.

To create the C6xxx configuration, start the Code Composer Setup program. In the Import Configuration window, select the *C62xx Fast Sim Ltl Endian* configuration. Then press *Add To System Configuration*. This will update the System Configuration area. The designs will not execute correctly if this is not done.

You can easily change the configuration to any other configurations provided by TI. But, after every configuration change, the code must be properly compiled using Code Composer Studio.

If the configuration is changed from C6x little endian DSP to a big endian DSP, remember to change the appropriate compiler and assembler flags before building, as explained in the section "CCStudio Build Options" on page 1-3. Also, choose the right run time library (for big endian, the library name ends in *e.lib*, for example, *rts6201e.lib*).

To change the DSP from C6x to C54x, the CCStudio version must be 1.2 or higher. Select the right configuration file from the TI CCStudio 1.2 Setup program. This will map the memory correctly for a particular DSP.

Do the following to run the provided examples:

- From CCS1.2 setup, select the C54x simulator.
- For Board Properties, select the Simulator Config File *sim541.cfg*.

The ADS designs are described in the following sections.

addsub.dsn

This simple example uses two instances of a TICCStudioCosim component. The TI DSP code can be found in *isscosim_prj\isscosim_src\testaddsub.c*. The TI CCStudio project that was used to build is *isscosim_prj\addsub\addsub_c6x*. Before running the cosimulation, please change the Program parameter to point to the area where you have *isscosim_prj\addsub\addsub_c6x.out*.

viterbi.dsn

This example that uses a complex algorithm of Viterbi decoding. The encoder is a convolutional encoder that is built using ADS components.The encoded data is mixed with Gaussian noise. The output is decoded using a TICCStudioCosim component that runs the Viterbi Decoding algorithm on the incoming data. The TI DSP code can be found in the file *isscosim_prj\isscosim_src\viterbi.c*. The TI CCStudio project that was used to build is *isscosim_prj\viterbi\viterbi_c6x*. Before running the cosimulation, please change the Program parameter to point to the area where you have *isscosim_prj\viterbi\viterbi_c6x.out*.

fir_xdas.dsn

This example demonstrates cosimulation with XDAS (eXpress DSP Algorithm Standard) compliant FIR algorithm. The code is shipped as an example with TI XDAS rules and guidelines. The XDAS compliant FIR algorithm implementation can be found under the directory *isscosim_prj\isscosim_src*. The only file that was required to be created is *fir_cosim.c*. This file defines three routines:

- *void fir_cosim_init()*
- *void fir_cosim_apply()*
- *void fir_cosim_exit()*.

The routines take care of instantiating, using and deleting the FIR algorithm. The TI CCStudio project that was used to build is *isscosim_prj\filter_xdas*. To rebuild this project, you must also have the XDAS related headers. These can be obtained from TI (some are shipped as part of this example). Before running the cosimulation, change the Program parameter to point to the area where you have *isscosim_prj\fir_xdas\fir_xdas_c6x.out*.

Index