## TMS320TCI66x

Keystone Multicore Workshop

## Lab Manual

Publication Number: SPRP820 Publication Date: April 2014



Texas Instruments Incorporated 20450 Century Boulevard Germantown, MD 20874 USA

## **Copyright and Contact Information**

#### **Document Copyright**

Publication Title:TMS320TCI66x Keystone Multicore Workshop Lab Manual Publication Number: SPRP820 Revision: A © 2014 Texas Instruments Incorporated All Rights Reserved. Reproduction, adaptation, or translation without prior written permission is prohibited, except as allowed under the copyright laws.

#### Software Copyright

Product Name: TMS320TCI66x Product Release: MCSDK 2.x © 2014 Texas Instruments Incorporated All Rights Reserved.

#### **Contact Information**

#### General

20450 Century Boulevard Germantown, MD 20874 Voice: 301.515.8580 Fax: 301.515.7687 Web: www.ti.com (Broadband/Voice over IP)

#### **Sales Information**

E-mail: mktgsupport@list.ti.com

#### **Applications Engineering**

For Registered Customers Only: E-mail: tech\_support@ti.com

The Telogy Software Applications Engineering group is available to all customers who need technical assistance with a Telogy product, technology, or solution. Inquiries are categorized according to the urgency of the issue, as follows:

**Priority Level 4 (P4)**—You need information or assistance about Telogy product capabilities, product installation, or basic product configuration.

*Priority Level 3 (P3)* — Your network performance is degraded. Network functionality is noticeably impaired, but most business operations continue.

*Priority Level 2 (P2)* — Your production network is severely degraded, affecting significant aspects of business operations. No workaround is available.

**Priority Level 1 (P1)**—Your production network is down, and a critical impact to business operations will occur if service is not restored quickly. No workaround is available.

## **Notices and Trademarks**

#### **Important Notice**

Texas Instruments Incorporated reserves the right to make changes to its products or discontinue any product or service without notice, and to advise customers to obtain the latest version of relevant information to verify, before placing orders, that the information being relied upon is current and complete. All products are sold subject to the terms and conditions of sale supplied at the time of order acknowledgement, including those pertaining to warranty, patent infringement, and limitation of liability.

Customers are responsible for their applications using Texas Instruments Software.

#### **Notice of Proprietary Information**

Information contained herein is subject to the terms of the Non-disclosure Agreement between Texas Instruments Incorporated and your company, and is of a highly sensitive nature and is confidential and proprietary to Texas Instruments Incorporated. It shall not be distributed, reproduced or disclosed orally or in written form, in whole or in part, to any party other than the direct recipients without the express written consent of Texas Instruments Incorporated.

Telogy Software, VLYNQ, PIQUA Software, wONE, PBCC, Uni-DSL, Dynamic Adaptive Equalization, Telinnovation, TurboDSL Packet Accelerator, interOps Test Labs, TurboDOX, and INCA are trademarks of Texas Instruments Incorporated.

All other brand names and trademarks mentioned in this document are the property of Texas Instruments Incorporated or their respective owners, as applicable.

## **Release History**

Release	Date	Chapter/Topic	Description/Comments
SPRP820	April 2014	All	Initial release

## Preface

## **About this Document**

This document is part of a document set prepared in support of the Texas Instruments/Telogy Software TNETVxxxx x.x product release.

## **About the Document Set**

Various books in the document set will be of interest to developers of voice-over-packet products according to their role:

- Project Managers
- Hardware Engineers
- Software Engineers
- Test Engineers
- Application Developers

### **Texas Instruments Silicon and Telogy Software Documents**

The following Texas Instruments/Telogy Software-produced documents are provided with TNETVxxxx Release x.x:

- DSP Module Software Architecture
- MXP Operating Environment Reference GuideMXP Operating Environment User's Manual

## **Document Conventions**

This document uses the following conventions:

- Commands and keywords are in **boldface** font.
- Arguments for which you supply values are in *italic* font.
- Terminal sessions and information the system displays are in screen font.
- Information you must enter is in **boldface screen font**.
- Elements in square brackets ([]) are optional.

Notes use the following conventions:

**NOTE**—Means reader take note. Notes contain helpful suggestions or references to material not covered in the publication.

The information in a caution or a warning is provided for your protection. Please read each caution and warning carefully.

**CAUTION**—Indicates the possibility of service interruption if precautions are not taken.



**WARNING**—Indicates the possibility of damage to equipment if precautions are not taken.

## Contents

	Copyright and Contact Information	
	Notices and Trademarks	
	Release History Preface	
	About this Document	
	About the Document Set	
	Texas Instruments Silicon and Telogy Software Documents	ø-v
	Document Conventions	
	List of Tables	
	List of Figures	
	Preparations	1-1
	1.1 Introduction	1-1
	1.2 Software	
	1.3 EVM Configuration	1-1
Chapter 2		
	CCS Basics (SRIO Loopback)	2-1
	2.1 Purpose	
	2.2 Instructions	
Chapter 3		
	HyperLink Communication	3-1
	3.1 Purpose	
	3.2 Instructions	
Chapter 4		
	SRIO Type 11	4-1
	4.1 Purpose	4-1
	4.2 Project Files.	

### Chapter 5

Optimization 5	
5.1 Purpose	5-1
5.2 Project Files	5-1
5.3 Instructions	5-1
5.3.1 Cache Analysis	
5.3.2 Change the Code to Speed Up to 32K	5-8

## Chapter 6

Using Adva	nced Debug	6-1
6.1 Purpose		6-1
6.1.1 Why	y the Debug Version is Used	6-1
6.2 Instructio	ons	6-2

## Chapter 7

Using MPAX to Define Private Core Memory in DDR	7-1
7.1 Purpose	
7.2 Overview	7-2
7.2.1 Short Description of MPAX (Memory Protection and Extension).	7-2
7.2.2 Coherency Discussion	7-2
7.2.3 Usage of EDMA to Move Data to and from Private Memory	7-2
7.2.4 Platform Configuration and the Memory Map	7-3
7.2.5 MAR Registers	7-4
7.3 Instructions	
7.3.1 Using Trace to Verify the Write Physical Address	7-6
7.3.2 Additional Considerations	7-15

## Chapter 8

STM Library and System Trace	8-1
8.1 Purpose	8-1
8.2 Project Files.	8-1
8.3 Instructions.	8-2
Revision History	

## **List of Tables**

Table 1-1	No Boot Dipswitch Settings	<u>)</u>
Table 1-2	EVM Emulator Types1-2	2
Table 5-1	Clock Values & Cycle Counts	3

## **List of Figures**

Figure 2-1	C6000 Compiler Include Options
Figure 2-2	Launch Target Configuration
Figure 2-3	CCS Debug Window for evm6678Trace.ccxml2-6
Figure 3-1	C6000 Compiler Include Options
Figure 4-1	Verify Successful .out Build
Figure 4-2	CCS Debug: Select and Group Cores
Figure 4-3	SRIOSingleSRIO Run Results
Figure 5-1	CCS Project Settings
Figure 6-1	CCS Debug: IntrinsicCFilters Breakpoint
Figure 6-2	CCS Debug: Show View
Figure 6-3	CCS Debug: IntrinsicCFilters Cache (4K)6-5
Figure 6-4	CCS Debug: IntrinsicCFilters L1D Cache Lines (4K)
Figure 6-5	CCS Debug: IntrinsicCFilters L1D Cache (Last Line, 4K)
Figure 6-6	CCS Debug: IntrinsicCFilters L1D Cache Lines (16K)
Figure 7-1	CCS Debug: MPAX Utilities
Figure 7-2	Trace System Control Setting
Figure 7-3	Trace System Control Settings: Select Receiver
Figure 7-4	Breakpoint Properties: Default Configuration7-9
Figure 7-5	Breakpoint Properties: Example Configuration7-10
Figure 7-6	Breakpoint Properties: Address Mask Export Bits7-11
Figure 7-7	Breakpoint Properties: Example Configuration Complete7-12
Figure 7-8	Choosing the Trace Analyzer
Figure 7-9	Trace Results
Figure 7-10	Trace Display Results: Core Details
Figure 7-11	Trace Display Results: Full Frame
Figure 8-1	CCS Project Properties
Figure 8-2	STM Lib Directory Structure
Figure 8-3	CCS Target Configurations
Figure 8-4	CCS Debug Trace Configuration: Show All Cores
Figure 8-5	Non Debuggable Devices: Not Connected8-6
Figure 8-6	Non Debuggable Devices: Connected
Figure 8-7	Enable Hardware Trace Analyzer
Figure 8-8	Hardware Trace Analysis Configuration
Figure 8-9	Trace Viewer CSSTM_0

## **List of Procedures**

Procedure 1-1	Create a New Target in CCS
Procedure 2-1	Import the Example Project
Procedure 2-2	Verify and Set Project Properties
Procedure 2-3	Build the Project
Procedure 2-4	Connect to the Target EVM
Procedure 2-5	Load and Run the Program
Procedure 3-1	Import the Example Project
Procedure 3-2	Verify and Set Project Properties
Procedure 3-3	Loopback Mode
Procedure 3-4	Build the Project
Procedure 3-5	Connect to the EVM
Procedure 3-6	Load and Run the Program
Procedure 3-7	Board-to-board HyperLink
Procedure 4-1	Import the Project
Procedure 4-2	Build the Project
Procedure 4-3	Connect to the EVM
Procedure 4-4	Load and Run
Procedure 5-1	Build and Run the Project
Procedure 5-2	Connect to the EVM
Procedure 5-3	Load and Run the Program
Procedure 5-4	Compiler Optimization
Procedure 5-5	Enable Software Pipelining
Procedure 5-6	Align the Data
Procedure 5-7	Enable the MUST_ITERATE Pragma
Procedure 5-8	Cache Considerations
Procedure 6-1	View the 4K Case
Procedure 6-2	Looking at the Cache Lines for 4K Case
Procedure 6-3	View the Cache Lines for 16K Case
Procedure 6-4	View the Cache Lines for 8K Case
Procedure 7-1	Run the Example Code
Procedure 7-2	Connect to the Non-debuggable Devices (Esp. CCTMS_0)7-6
Procedure 7-3	Load the Code to the 8 Cores
Procedure 7-4	Configure the CSSTM_0 Trace Control7-7
Procedure 7-5	Add and Configure the Trace Location
Procedure 7-6	Start Display7-12
Procedure 7-7	Enable the Trace Point and Run7-13
Procedure 8-1	Build and Run the Project
Procedure 8-2	Connect to the EVM
Procedure 8-3	Load the Program and Configure the Trace
Procedure 8-4	Run the Program

List of Procedures

## **Chapter 1**

## Preparations

## **1.1 Introduction**

The exercises in this manual were developed for delivery in a classroom environment but have been adapted for use in a personal lab environment. Variations in network configuration, software installation, software version, and Host PC configuration could result in different outcomes.

## 1.2 Software

The exercises in this manual require the following software installations:

- Multicore Software Development Kit (MCSDK) Version 2.x or later
- Code Composer Studio (CCS) Version 5.x or later.

For more information, refer to the BIOS MCSDK Getting Started Guide http://processors.wiki.ti.com/index.php/BIOS\_MCSDK\_2.0\_Getting\_Started\_Guide

## **1.3 EVM Configuration**

All exercises in this manual were designed for use on the TMS320C6678 Lite evaluation modules from Texas Instruments. While these procedures have not been tested on the TMS320C6657 and the EVMK2H, they should also work on those devices.

For labs requiring a mezzanine card with a trace emulator, the TMS320C6678LE is recommended.

### Before you begin, set the EVM to 'no boot' mode as shown in Table 1-1.

#### Table 1-1 No Boot Dipswitch Settings

Boot Mode	DIP SW3 (Pin 1, 2, 3, 4)	DIP SW4 (Pin 1, 2, 3, 4)	DIP SW5 (Pin 1, 2, 3, 4)	DIP SW6 (Pin 1, 2, 3, 4)
No boot	(off, on, on, on)	(on, on, on, on)	(on, on, on, on)	(on, on, on, on)
End of Table 1-1				

**Note**—Additional EVM switch settings are available at the following link: <u>http://processors.wiki.ti.com/index.php/TMDXEVM6678L\_EVM\_Hardware</u> <u>Setup#Boot\_Mode\_Dip\_Switch\_Settings</u>

#### Procedure 1-1 Create a New Target in CCS

#### Step - Action

1 Launch CCS by double-clicking the icon on the desktop.

**Note**—As CCS initializes, a pop-up will appear with a default workspace. Replace the default workspace with "C:/ti/workspace."

- 2 Create a new target configuration:
  - **2a** Select the CCS menu option *View*  $\rightarrow$  *Target Configurations*.
  - 2b Select User Defined.
  - 2c Right-click and select New Target Configuration.
- 3 Enter the name of the new target configuration in the *File Name*: text box.
  - **3a** Set the File name based on the EVM model, <*model*>.*ccxml* For example, 'EVM6678LE.ccxml.'
  - **3b** Leave the *Location* the default value:
    - "C:\Documents\_and\_Settings\student\ti\CCSTargetConfigurations"
  - **3c** Click the *Finish* button. The .ccxml file will now open in a GUI-based view with the *Basic* tab active.
- 4 Define the new target configuration by selecting the connection type in the *Basic* Tab.
  - **4a** Locate your EVM model Table 1-2 and set the properties accordingly.

#### Table 1-2 EVM Emulator Types

EVM Model	/M Model Emulator Type	
EVM6657L	Texas Instruments XDS100v2 USB Emulator	TMS320C6657
EVM6657LE	Blackhawk XDS560v2-USB Mezzanine Card	TMS320C6657
EVM6678L	Texas Instruments XDS100v1 USB Emulator	TMS320C6678
EVM 6678LE     Blackhawk XDS560v2-USB Mezzanine Card     TMS320C6678		TMS320C6678
End of Table 1-2		

- **4b** The *Connection* drop-down menu identifies the emulator type, as shown in the table above. For example, 'Blackhawk XDS560v2-USB Mezzanine Card.'
- **4c** Board or Device identifies the TI processor device, as shown in the table above. For example, 'TMS320C6678.'
- **4d** Under *Save Configuration*, click the *Save* button.

- **5** Configure setup on the *Advanced* Tab
  - 5a Click the Advanced tab at the bottom of the screen.
  - **5b** Select Core 0 on the target device:
    - $TMS320C6657_0 \rightarrow lcePick_C_0 \rightarrow Subpath_1 \rightarrow C66XP_0$ OR
    - $TMS320C6678_0 \rightarrow IcePick_D \rightarrow subpath_0 \rightarrow C66x_0$
  - **5c** You will now see a sub-window called *Cpu Properties* that allows you to choose an *initialization script*.
  - **5d** Locate the appropriate GEL file, then click *Open*:
    - For EVM6657L/LE, select:
      - C:\ti\ccsv5\ccs\_base\emulation\boards\evmc6657I\gel\evmc6657I.gel
    - For EVM6678L/LE, select:
  - **5e** Click the *Save* button.

End of Procedure 1-1

## **Chapter 2**

# **CCS Basics (SRIO Loopback)**

## 2.1 Purpose

The purpose of this exercise is to demonstrate how to build and run a very basic Code Composer Studio (CCS) project on the C6678 EVM. The Direct IO Loopback example delivered with the MCSDK is used to help illustrate these concepts.

## 2.2 Instructions

In this exercise, you will import a sample project from the MCSDK into CCS, build the sample application code, connect to the EVM, load the code, run the application, and verify the results.

The list of processes used in this example are as follows:

- Procedure 2-1 "Import the Example Project"
- Procedure 2-2 "Verify and Set Project Properties"
- Procedure 2-3 "Build the Project"
- Procedure 2-4 "Connect to the Target EVM"
- Procedure 2-5 "Load and Run the Program"

#### Procedure 2-1 Import the Example Project

#### Step - Action

1 Launch CCS by double-clicking the icon on the desktop.

**Note**—As CCS initializes, a pop-up will appear with a default workspace. Replace the default workspace with "C:/ti/workspace".

- 2 Once CCS starts, verify that the perspective is set to CCS Edit.
- 3 Discover the new packages installed in folders other than C:\ti.
  - **3a** Select the CCS menu option *Window*  $\rightarrow$  *Preferences*
  - **3b** In the pop up window that appears, select *Code Composer Studio*  $\rightarrow$  *RTSC*  $\rightarrow$  Products
  - **3c** Add the master folder into *Tool Discover Path* by clicking the *Add* in the upper right corner of the pop-up window and select the master folder.
  - **3d** For example, select C:\ti\MCSDK\_2\_1\_2\_6 to add in the *Tool Discovery Path*.
  - 3e Click Refresh to update the discovered tools list.

- 4 Import the example project as follows:
  - 4a Select the CCS menu option Project → Import Existing CCS Eclipse Project
  - 4b Set Select search directory to locate the example projects available for your EVM:
    - (C6657L/LE) C:\ti\mcsdk\pdk\_C6657\_1\_1\_2\_5\packages\ti\drv\exampleProjects
    - (C6678L/LE) C:\ti\mcsdk\pdk\_C6678\_1\_1\_2\_5\packages\ti\drv\exampleProjects
  - **4c** From the list of *Discovered projects*, place a check mark in the box next to *SRIO\_LoopbackDioIsrexampleproject*.

**Note**—There are multiple SRIO projects with similar names. Verify that the project you import matches exactly with the name as shown above.

- 4d Place a check mark on Copy projects into workspace.
- **4e** Click the *Finish* button.

The *SRIO\_LoopbackDioIsrexampleproject* project should now appear in the CCS *Project Explorer* on the left-hand side of your screen.

#### End of Procedure 2-1

Procedure 2-2	Verify and Set Project Properties
---------------	-----------------------------------

#### Step - Action

- 1 In Project Explorer, right click on SRIO\_LoopbackDiolsrexampleproject and select Properties.
- 2 Select General properties.
- 3 Choose the *Main* tab and set/verify the *Device* properties as follows:
  - 'Family = C6000'
  - 'Variant = Generic C66x Device'
- 4 Select *Build* properties.
- 5 Choose C6000 Compiler → Processor Options and set/verify the following properties:
  - 'Configuration = Debug'
  - 'Target processor version = 6600'
  - 'Application binary interface = eabi'

#### **\_**.

**Note**—Different version of CCS may have slightly different GUI. The Application binary interface tab may be part of the *main* window and not the *processor option* window.

- 6 Select Build properties.
- 7 Choose C6000 Compiler  $\rightarrow$  Optimization and set/verify the following properties:
  - 'Optimization level = 0'
  - 'Optimize for code size = 0'
- 8 Select Build properties
- 9 Choose C6000 Compiler  $\rightarrow$  Debug Options and set/verify the following properties:
  - 'Debugging model = Full symbolic debug'

10 Select Build properties, choose C6000 Compiler → Include Options and ensure that include paths are setup as shown in Figure 2-1.

#### Figure 2-1 C6000 Compiler Include Options

Configuration:	Debug [Active]
"\${CG_TOOL_RC	clude search path (include_path, -I) DOT}/include" _PATH}/ti/drv/srio/example/SRIOLoopbackDioIsr"

11 Click the OK button to save the project properties and close the Properties window.

#### End of Procedure 2-2

#### Procedure 2-3 Build the Project

#### Step - Action

- 1 In Project Explorer, select the SRIO\_LoopbackDiolsrexampleproject project.
- 2 Build the project:
  - Select the CCS menu option  $Project \rightarrow Build Project$
  - OR
  - Right-click on the project in Project Explorer and select Build Project

OR

- Click on the hammer icon
- 3 CCS will now attempt to compile and link the project. This may take a few minutes to complete.
- 4 Please direct your attention to the CCS *Console*. On a successful build, you will see no errors generated in the *Problems* window (NOTE: There may be warnings) and the following message should display in the *Console* window:

'Finished building target: SRIO\_LoopbackDioIsrexampleproject.out'

\*\*\*\* Build Finished \*\*\*\*

#### **QUESTIONS:**

Was the file SRIO\_LoopbackDioIsrexampleproject.out generated?

**Note**—From the *CCS Edit* perspective, check the *Binaries* or *Debug* directory. From the *CCS Debug* perspective, check the *Console*.

#### End of Procedure 2-3

Proce	dure 2-4 Connect to the Target EVM
Step	- Action
1	Click the Open Perspective (available right top corner of the CCS).
2	Switch to the Debug Perspective by selecting the CCS monulantian Window $\rightarrow$ Open Perspective

- Switch to the Debug Perspective by selecting the CCS menu option Window  $\rightarrow$  Open Perspective  $\rightarrow$  CCS Debug.
- **3** Connect the power adapter to your EVM, then connect your laptop to the emulator port on the EVM using the provided USB cable. If you are using the XDS560v2, wait till the solid red light appears before proceeding to the next step.

**Note**—The Windows "Found New Hardware Wizard" may pop-up when you first connect the emulator via USB to the laptop. Select "Yes, this time only"  $\rightarrow$  Next  $\rightarrow$  "Install the software automatically"  $\rightarrow$  Next, and allow the drivers to install on your system. Then click "Finish." You may need to restart CCS at this point.

- 4 Select the CCS menu option *View* → *Target Configurations*. Your newly-created .ccxml target configuration file should be available under *User Defined* target configurations.
- 5 If more than one target is configured, select the target that you defined, right click and set it as default

- **6** Launch the target configuration as shown in Figure 2-2:
  - **6a** Select the target configuration .ccxml file.
  - **6b** Right click and select *Launch Selected Configuration*.

#### Figure 2-2 Launch Target Configuration

CCS Edit - Source not found Coo	le Composer Studio	
File Edit View Navigate Proje	t Run Scripts Window Help	
📬 🖬 🗟 🔍 🔹 🎄 🕶	🔌 🛷 🕶 🔲 🌾 🔶 🗸	⇔ •
🔓 Project Explorer 🕄 Target Confi	igurations 🛛 🗖 🗖	abort() at
	🖹 🛠 😓	Can't find a
type filter text		Locate the f
😑 Projects		View Disass
a 🗁 User Defined		[
EVM6638.ccxml		Locate File
😤 KeplerDSP.ccxml		
😢 KeplerNoGel.ccxml		
ShannonSimulator.ccxm	I	
Shannon_6678.ccxml		
TCI6638.ccxml		
TI_v100.ccxml		
TwoMezzanine6678.ccxn	nl	
c6678MPAX4.ccxml		
evmWithLinux.ccxn	New Target Configuration	
E evitwithEndx.ccxii E		
	Import Target Configuration	
×	Delete	Delete
	Rename	F2
Sec. Sec. Sec. Sec. Sec. Sec. Sec. Sec.	Refresh	F5
۲	Launch Selected Configuration	
	Set as Default	
	Link File To Project	+
	Properties	Alt+Enter

7 This will bring up the *Debug* window as shown in Figure 2-3.

Figure 2-3 CCS Debug Window for evm6678Trace.ccxml

File	Edit	View	Proje	ect	Tools	s R	un	Sc	ripts	W	/indo	w	He	lp									
<b>1</b> 2	•	lē	8	*	s	1	R				59		Ĵ_ (	P	.e	⇒∏	-9		-	勉	₫	- 🔭	
参 [	)ebug	x																		S	ĝ 8		
4 (	🖗 evi	m6678T	race.c	cxml	[Cod	e Co	mpo	oser	Stud	dio -	Devi	ice [	)ebu	uggi	ing]								
	×	Blackh	awk X	DS56	0v2-U	ISB S	yste	m T	race	Em	ulato	or_0/	C66	ixx_0	) (Di	sco	nne	cted	: Un	kno	wn)		
	×	Blackh	awk X	DS56	0v2-U	ISB S	yste	m T	race	Em	ulato	or_0/	C66	ixx_1	L (Di	sco	nne	cted	: Un	kno	wn)		
	@	Blackh	awk X	DS56	0v2-U	ISB S	yste	m T	race	Em	ulato	or_0/	C66	ixx_2	2 (Di	sco	nne	cted	: Un	kno	wn)		
	×	Blackh	awk X	DS56	0v2-U	ISB S	yste	m T	race	Em	ulato	or_0/	C66	ixx_3	3 (Di	sco	nne	cted	: Un	kno	wn)		
	×	Blackh	awk X	DS56	0v2-U	ISB S	yste	m T	race	Em	ulato	or_0/	C66	ixx_4	4 (Di	sco	nne	cted	: Un	kno	wn)		
	@	Blackh	awk X	DS56	0v2-U	ISB S	yste	m T	race	Em	ulato	or_0/	C66	ixx_5	5 (Di	sco	nne	cted	: Un	kno	wn)		
	@	Blackh	awk X	DS56	0v2-U	ISB S	yste	m T	race	Em	ulato	or_0/	C66	ixx_6	5 (Di	sco	nne	cted	: Un	kno	wn)		
	@	Blackh	awk X	DS56	0v2-U	ISB S	yste	m T	race	Em	ulato	or_0/	C66	ixx_7	7 (Di	sco	nne	cted	: Un	kno	wn)		
	~																						

**7a** Select Core 0 (*C66x\_0*)

**7b** Right click and select *Connect Target*.

End of Procedure 2-4

#### Procedure 2-5 Load and Run the Program

Step - Action

- 1 Select Core 0 and load the .out file created earlier in the lab.
  - **1a** Select the CCS menu option  $Run \rightarrow Load \rightarrow Load$  Program.
  - **1b** Click Browse project...
  - **1c** A pop-up will appear with the projects names.
  - 1d Select SRIO\_LoopbackDiolsrexampleproject
  - **1e** Click Debug, then select *SRIO\_LoopbackDiolsrexampleproject.out* and click *OK*.
  - **1f** Click *OK* to load the application to the target (Core 0).
- 2 Run the application by selecting the CCS menu option  $Run \rightarrow Resume$ .

OR

Click on the green arrow.

**3** Once the program completes successfully, you will see the message "Debug(Core 0): DIO with Interrupts example completed successfully."

```
4
    Then select the CCS menu option Run \rightarrow Suspend.
    OR
    Click on the two yellow bars next to the green arrow
    The expected output on the console should appear as follows:
[C66xx 0] Executing the SRIO DIO example on the DEVICE
[C66xx 0] Debug(Core 0): System Initialization for CPPI & QMSS
[C66xx 0] Debug(Core 0): Queue Manager and CPPI are initialized.
[C66xx 0] Debug(Core 0): Host Region 0x8268f0
[C66xx 0] Debug(Core 0): SRIO Driver has been initialized
[C66xx_0] ****** DIO Socket Example with Interrupts (Core 0) *******
[C66xx 0] Debug(Core 0): Starting the DIO Data Transfer - Src(0) 0x@1081b100 Dst(0)
0x@1081b200
[C66xx_0] Debug(Core 0): Starting the DIO Data Transfer - Src(8) 0x@1081c100 Dst(8)
0x@1081c200
[C66xx 0] Debug(Core 0): DIO Socket (0) Send for iteration 0
[C66xx 0] Debug(Core 0): ISR Count: 1
. . .
[C66xx_0] Debug(Core 0): DIO Socket (2) Send for iteration 2
[C66xx 0] Debug(Core 0): ISR Count: 9
[C66xx 0] Debug(Core 0): Transfer Completion without Errors - 9
[C66xx 0] Debug(Core 0): Transfer Completion with Errors
                                                      - 0
[C66xx 0] Debug(Core 0): DIO Transfer Data Validated for all iterations
[C66xx 0] Debug(Core 0): DIO Data Transfer (WRITE) with Interrupts Example Passed
[C66xx 0] ******* DIO Socket Example with Interrupts (Core 0) *******
[C66xx 0] Debug(Core 0): Starting the DIO Data Transfer - Src(0) 0x@1081b100 Dst(0)
0x@1081b200
[C66xx 0] Debug(Core 0): Starting the DIO Data Transfer - Src(8) 0x@1081c100 Dst(8)
0x@1081c200
[C66xx 0] Debug(Core 0): DIO Socket (0) Send for iteration 0
[C66xx_0] Debug(Core 0): ISR Count: 10
[C66xx_0] Debug(Core 0): DIO Socket (1) Send for iteration 0
[C66xx 0] Debug(Core 0): ISR Count: 11
[C66xx 0] Debug(Core 0): DIO Socket (2) Send for iteration 0
[C66xx 0] Debug(Core 0): ISR Count: 12
. . .
[C66xx 0] Debug(Core 0): DIO Socket (0) Send for iteration 2
[C66xx 0] Debug(Core 0): ISR Count: 16
```

```
[C66xx_0] Debug(Core 0): DIO Socket (1) Send for iteration 2
[C66xx_0] Debug(Core 0): ISR Count: 17
[C66xx_0] Debug(Core 0): DIO Socket (2) Send for iteration 2
[C66xx_0] Debug(Core 0): ISR Count: 18
[C66xx_0] Debug(Core 0): Transfer Completion without Errors - 9
[C66xx_0] Debug(Core 0): Transfer Completion with Errors - 0
[C66xx_0] Debug(Core 0): DIO Transfer Data Validated for all iterations
[C66xx_0] Debug(Core 0): DIO Data Transfer (READ) with Interrupts Example Passed
[C66xx_0] Debug(Core 0): Allocation Counter : 81
[C66xx_0] Debug(Core 0): Free Counter : 72
[C66xx_0] Debug(Core 0): DIO with Interrupts example completed successfully.
End of Procedure 2-5
```

#### **QUESTIONS:**

- 1. Using a text editor, look at the CFG file and determine how the project includes the SRIO module.
  - a. What other modules are needed? Hint: QMSS and CPPI are needed for SRIO
  - b. The CFG file specifies CORE 0 and CORE 1. Is this important to the execution of the application?
- 2. 3. Load the OUT file and run it on Core 1.
  - a. Does it run?
  - b. Look at the main function and explain why.

## **Chapter 3**

## HyperLink Communication

## 3.1 Purpose

The purpose of this exercise is to demonstrate how to build an application that uses the HyperLink interface on KeyStone C66x devices.

**Note**—Not all KeyStone devices include HyperLink. Refer to the data manual for your device before proceeding.

**Note**—Not all KeyStone EVMs include a HyperLink interface on the board. An expansion module may be required. Refer to the Quick Start Guide for your EVM before proceeding.

## 3.2 Instructions

Begin by importing HyperLink example code from the MCSDK and running it in loopback mode on a single C66x EVM. In this example, the same C66x EVM acts as both the sender and the receiver of packets. Only one C66x EVM is required for this part of the exercise.

The second part demonstrates the HyperLink connection between two C66x EVMs. As a result, this lab requires a second C66x EVM, a HyperLink cable (HL5CABLE), and an optional EVM breakout card (CI2EVMBOC). One C66xx EVM acts as the sender (Tx) and the other acts as the receiver (Rx).

The list of processes used in this example are as follows:

- Procedure 3-1 "Import the Example Project"
- Procedure 3-2 "Verify and Set Project Properties"
- Procedure 3-3 "Loopback Mode"
- Procedure 3-4 "Build the Project"
- Procedure 3-5 "Connect to the EVM"
- Procedure 3-6 "Load and Run the Program"
- Procedure 3-7 "Board-to-board HyperLink"

#### Procedure 3-1 Import the Example Project

Step - Action

- 1 Move to the CCS Edit Perspective.
- 2 Import the example project as follows:
  - 2a Select the CCS menu option Project → Import Existing CCS Eclipse Project
  - **2b** Set *Select search directory* to locate the example projects available for your EVM:
    - (C6657L/LE) C:\ti\mcsdk\pdk\_C6657\_1\_1\_2\_5\packages\ti\drv\exampleProjects
    - (C6678L/LE) C:\ti\mcsdk\pdk\_C6678\_1\_1\_2\_5\packages\ti\drv\exampleProjects
  - **2c** From the list of *Discovered projects*, place a check mark in the box next to *hypInk\_exampleProject*
  - 2d Place a check mark on Copy projects into workspace.
  - **2e** Click the *Finish* button.
  - **2f** The *hyplnk\_exampleProject* should now appear in the CCS *Project Explorer* on the left-hand side of your screen.

End of Procedure 3-1

#### QUESTIONS:

Expand the *hyplnk\_exampleProject* folder and double click on *hyplnkLLDCfg.h* to view the file and answer the following:

- 1. How many lanes are configured?
- 2. What is the baud rate? (HINT: 01p250 means 1.25GBaud)

#### Procedure 3-2 Verify and Set Project Properties

Step - Action

- 1 In Project Explorer, right click on hyplnk\_exampleProject and select Properties.
- 2 Select General properties.
- 3 Choose the *Main* tab and set/verify the *Device* properties as follows:
  - 'Family = C6000'
    - 'Variant = Generic C66x Device'
- 4 Select *Build* properties.
- 5 Choose C6000 Compiler  $\rightarrow$  Processor Options and set/verify the following properties:
  - 'Configuration = Debug'
  - 'Target processor version = 6600'
  - 'Application binary interface = eabi'

**Note**—In different versions of CCS, the Application binary interface tab is located in the *main* tab

- 6 Select Build properties.
- 7 Choose C6000 Compiler  $\rightarrow$  Optimization and set/verify the following properties:
  - 'Optimization level = 0'
  - 'Optimize for code size = 0'
- 8 Select Build properties
- **9** Choose C6000 Compiler  $\rightarrow$  Debug Options and set/verify the following properties:
  - 'Debugging model = Full symbolic debug'

•

10 Select Build properties, choose C6000 Compiler → Include Options and ensure that include paths are setup as shown in Figure 3-1.



### **Include Options**

Configuration: Debug [ Active ]

Add dir to #include search path (--include\_path, -I)

"\${CG\_TOOL\_ROOT}/include" "\${PDK\_INSTALL\_PATH}"

"\${PDK\_INSTALL\_PATH}/ti/csl/src/intc"

"\${PDK\_INSTALL\_PATH}/ti/drv/hyplnk/example/common"

#### 11 Click the OK button to save the project properties and close the Properties window

#### End of Procedure 3-2

Proc	edure 3-3 Loopback Mode
Step	o - Action
1	Look for the following line in hyplnkLLDCfg.h and verify that it is uncommented.
	<b>#define</b> hyplnk_EXAMPLE_LOOPBACK
	When uncommented, this #define ensures that the example runs in loopback mode on a single EVM.
End o	of Procedure 3-3
Proc	edure 3-4 Build the Project
Step	o - Action
1	In Project Explorer, select the hypInk_exampleProject project.
2	Build the project:
	<b>2a</b> Select the CCS menu option Project $\rightarrow$ Build Project
	OR
	2b Right-click on the project in Project Explorer and select Build Project
3	CCS will now attempt to compile and link the project. This may take a few minutes to complete.
4	Please direct your attention to the CCS <i>Console</i> . On a successful build, you will see no errors generated in the <i>Problems</i> window (NOTE: There may be warnings) and the following message should display in the <i>Console</i> window:
<lin< td=""><td>king&gt;</td></lin<>	king>
'Fin	ished building target: hyplnk exampleProject.out'
****	Build Finished ****

End of Procedure 3-4

Mcycles

#### QUESTIONS:

Was the file hyplnk\_exampleproject.out generated?

\_\_\_\_

**Note**—From the *CCS Edit* perspective, check the *Binaries* or *Debug* directory. From the *CCS Debug* perspective, check the *Console*.

#### Procedure 3-5 Connect to the EVM

#### Step - Action

- 1 Click the *Open Perspective* (available right top corner of the CCS).
- 2 Switch to the Debug Perspective by selecting the CCS menu option
- **3** Window  $\rightarrow$  Open Perspective  $\rightarrow$  CCS Debug.
- 4 Select the CCS menu option *View* → *Target Configurations*. Select the target configuration you created
- 5 Launch the target configuration as follows:
  - **5a** Select the target configuration .ccxml file.
  - **5b** Right click and select Launch Selected Configuration.
- 6 This will bring up the *Debug* window.
  - **6a** Select Core 0 (*C66x\_0*)
  - **6b** Right click and select *Connect Target*.

#### End of Procedure 3-5

Proc	edur	e 3-6 Load and Run the Program
Ste	p - A	tion
1	Sele	ect Core 0 and load the .out file created earlier in the lab.
	1a	Select the CCS menu option $Run \rightarrow Load \rightarrow Load$ Program
	1b	Click Browse project
	1c	A pop-up will appear with the projects names.
	1d	Select hypInk_exampleProject
	1e	Click Debug, then select hypInk_exampleProject.out and click OK.
	1f	Click OK to load the application to the target (Core 0)
2	Rur	the application by selecting the CCS menu option $\mathit{Run}  o \mathit{Resume}.$
3		program attempts to send and receive tokens via the HyperLink interface in loopback de. So the same device acts as both the send and receive side.
4	A su	accessful run should produce the following console output for each iteration.
[C66	5xx_0	Checking statistics
[C66	5xx_0	About to pass 65536 tokens; iteration = 1
[C66	5xx_0	] === this is not an optimized example ===
[C66	5xx_0	Link Speed is 4 * 6.25 Gbps
[C66	5xx_0	Passed 65536 tokens round trip (read+write through hyplnk) in 9278 Mcyc
[C66	5xx_0	Approximately 141574 cycles per round-trip
[C66	5xx_0	] === this is not an optimized example ===

5 Multiple iterations are performed and the program will go on indefinitely until manually stopped. Once you have verified that the program has executed successfully, select the CCS menu option *Run* → *Suspend*.

End of Procedure 3-6

#### Procedure 3-7 Board-to-board HyperLink

#### Step - Action

1

A second C66x EVM is required perform this portion of the lab. You will need a HyperLink cable and (optional) a 2-EVM breakout card.

- Modify the example code for *hyplnk\_exampleProject* so it can be run on two EVMs:
  - 1a Open hypInkLLDCfg.h
  - **1b** Search for #define hyplnk\_EXAMPLE\_LOOPBACK
  - **1c** Comment out this line
  - **1d** Ensure that the baud rate is set to 6.25 Gbaud, i.e. the line #define hyplnk\_EXAMPLE\_SERRATE\_06p250 is uncommented.
- 2 Build the code, load to both targets, and run the generated out file on Core 0 only.
- 3 Modify the example code for hypInk\_exampleProject
  - 3a Open hypInkLLDCfg.h
  - **3b** Change the *Baud Rate* to a higher rate.
- 4 Build the code, load to both targets, and run on Core 0 only.

Note—The two systems must have the same rate!

End of Procedure 3-7

#### **QUESTION:**

- 1. What is the highest transfer rate that can be achieved using this example?
- 2. Look at the errata documentation and identify the theoretical limit of HyperLink transfer. Why is this the case?
- **\_**\_\_

**Note**—The maximum length of the board-to-board connection should be 4 inches. Thus, the physical cable connection is not as efficient as a hard wired connection.

## **Chapter 4**

# SRIO Type 11

## 4.1 Purpose

The purpose of this exercise is to demonstrate usage of Type 11 SRIO using an example application imported into CCS from the MCSDK.

## 4.2 **Project Files**

The following files are used in this lab:

- bioInclude.h
- bioMain.c
- bioUtilityAndGlobals.c
- device\_srio\_loopback.c
- ExampleSRIO.cmd
- fftRoutines.c
- gen\_twiddle\_fft16x16.c
- initialization.c
- masterTask.c
- multicoreLoopback\_osal.c
- requestProcessingData.c
- slaveTask.c
- SRIOMulticore\_fft\_1.cfg

## 4.3 Instructions

First, you will import the example project from MCSDK to CCS. Next, you will build an application from the project and load it to the EVM. Finally, you will run the project and observe the results.

The list of processes used in this example are as follows:

- Procedure 4-1 "Import the Project"
- Procedure 4-2 "Build the Project"
- Procedure 4-3 "Connect to the EVM"
- Procedure 4-4 "Load and Run"

### Procedure 4-1 Import the Project

Step - Action

- 1 Open CCS.
- 2 Once CCS starts, verify that the perspective is set to CCS Edit.
- 3 Import the project.
  - **3a** Select the CCS menu option Project  $\rightarrow$  Import Existing CCS Eclipse Project
  - **3b** Set *Select search directory* to point to where the project files for this exercise are located on your computer.
  - 3c From the list of Discovered projects, place a check mark in the box next to SRIOSingleSRIO
  - 3d Place a check mark on Copy projects into workspace.
  - **3e** Click the *Finish* button.

End of Procedure 4-1

#### Procedure 4-2 Build the Project

#### Step - Action

- 1 In *Project Explorer*, select the *SRIOSingleSRIO* project.
- 2 Clean the project by right-clicking on the project and selecting *Clean Build*.
- **3** Build the project:
  - **3a** Select the CCS menu option  $Project \rightarrow Build Project$ 
    - OR
  - 3b Right-click on the project in Project Explorer and select Build Project
- 4 CCS will now attempt to compile and link the project. This may take a few minutes to complete.

5 Verify that the executable (.out) was built by looking at the debug directory (assuming the build configuration is debug configuration) as shown in Figure 4-1.

Figure 4-1 Verify Successful .out Build

e Edit View Navigate Project Run Scripts Window Help "3 ▼ 🔄 🔞 🔦 ▼ 🏇 ▼ 🥔 ▼ 🗐 🏷 🗢 ▼ → ▼		🖹 🎭 CCS Debug 📴 CCS
) Project Explorer 🔀 🕃 Target Configurations		
🖉 largeFIR_Energy	-	
SRIO_LoopbackDioIsrexampleproject		
SRIOSingleSRIO [Active - Debug]		
Binaries		
A Differences		
ConfigPkg		
bioMain.obj - [C6000/le]		
bioWain.obj = [coood/le]		
fftRoutines.obj - [C6000/le]		
gen_twiddle_fft16x16.obj - [C6000/le]		
initialization.obj - [C6000/le]		
masterTask.obj - [C6000/le]		
multicoreLoopback_osal.obj - [C6000/le]	E	
and a state of the state of		
p gmss_drv.obj - [C6000/le]		
requestProcessingData.obj - [C6000/le]		
slaveTask.obj - [C6000/le]		
srio_drv.obj - [C6000/le]		
SRIOSingleSRIO.out - [C6000/le]		
bioMain.pp		
bioUtilityAndGlobals.pp		
ccsObjs.opt		
ccsSrcs.opt		
cppi_device.pp		🖳 Console 🕸 🛛 🤌 🔂 🔂 🔜 🗛 🛃 🖛 🖳 🕶 😭 🕶
device_srio_loopback.pp		CDT Build Console [SRIOSingleSRIO]
ftRoutines.pp		cmd file="./configPkg/compiler.opt" "/srio drv.c"
gen_twiddle_fft16x16.pp		'Finished building:/srio drv.c'
initialization.pp		
🚡 makefile		'Building target: SRIOSingleSRIO.out'
masterTask.pp		'Invoking: C6000 Linker' "C:/ti/ccs/ccsv5/tools/compiler/c6000/bin/cl6x" -mv6600 -g
multicoreLoopback_osal.pp		define=_LITTLE_ENDIANdefine=_TMS320C6600
🗋 objects.mk		define=TMS320C6600diag_warning=225abi=eabi -z
gmss_device.pp		-m"SRIOSingleSRIO.map"warn_sections
gmss_drv.pp		-i"C:/ti/ccs/ccsv5/tools/compiler/c6000/lib"
requestProcessingData.pp		-i"C:/ti/ccs/dsplib_c66x_3_0_0_8/lib" -i"C:/ti/ccs/ccsv5/tools/compiler/c6000/include"reread libs
slaveTask.pp		rem_model -o "SRIOSingleSRIO.out" -1"./configPkg/linker.cmd"
lo sources.mk		"./srio drv.obj" "./slaveTask.obj" "./requestProcessingData.obj
srio_drv.pp		"./qmss_drv.obj" "./qmss_device.obj"
SRIOMulticore_fft_1.map		"./multicoreLoopback_osal.obj" "./masterTask.obj"
SRIOSingleSRIO.map		"./initialization.obj" "./gen_twiddle_fft16x16.obj"
🚡 subdir_rules.mk		"./fftRoutines.obj" "./device_srio_loopback.obj" "./cppi_device.obj" "./bioUtilityAndGlobals.obj" "./bioMain.obj
🚡 subdir_vars.mk		-1"libc.a" -ldsplib.ae66 "/ExampleSRIO.cmd"
🚡 subdir.mk	-	<linking></linking>

#### End of Procedure 4-2

#### Procedure 4-3 Connect to the EVM

#### Step - Action

- 1 Cycle power on the EVM.
- 2 Click the Open Perspective (available right top corner of the CCS).
- 3 Switch to the Debug Perspective by selecting the CCS menu option Window  $\rightarrow$  Open Perspective  $\rightarrow$  CCS Debug.
- 4 Use the target configuration that you created previously.
- 5 Launch the target configuration as follows:
  - **5a** Select the target configuration .ccxml file.
  - **5b** Right click and select *Launch Selected Configuration*.
- 6 This brings up the *Debug* window.
- 7 Select all cores, right click, and group them by selecting *Group Core(s)* as shown in Figure 4-2. The default name will be *Group 1*

\_.

. .

**~** 1 •

. .

~

File Edit View Search Pro	_	drv.c - Code Composer Stur Tools Run Scripts Wi					
	-	8 8 🖌 - 1 🖾 1 🕍		-		-	🖇 🖏 CCS Debug 🔂 CCS
ॐ Debug ⊠			2. O. R 🔌	• • 6   🕸 · -	□ 🕬 Variables 🛛 🛱	Expressions and Registers	£
mazanin.ccxml [Code C					Name	Туре	Value
		Mezzanine Emulator_0/C66 Mezzanine Emulator_0/C66					
		Mezzanine Emulator_0/C66					
		Mezzanine Emulator_0/C66					
		Mezzanine Emulator_0/C66					
		Morranino Emulator 0/C66					
Blackhawk XDS560	<b>R</b> (	Connect Target	Ctrl+Alt+C	: Unknown)			
Blackhawk XDS560		Disconnect Target	Ctrl+Alt+D	: Unknown)			
		Enable Global Breakpoints					
		Enable Halt On Reset					
		Enable OS Debugging					
		Open GEL Files View					
	×	Hide core(s)					
	1	Show all cores					
		Group core(s)					
E Console 🛛		Ungroup core(s)		·			
mazanin.ccxml		Rename		-			
C66xx_0: GEL Output: Se C66xx_0: GEL Output: Se		Nellallie					
C66xx_1: GEL Output: Se		Remove All Terminated					
C66xx_1: GEL Output: Se	<b>Q</b> 1	Relaunch					
C66xx_2: GEL Output: Se C66xx_2: GEL Output: Se	<b>.</b> -	Terminate and Remove					
C66xx_3: GEL Output: Se		Terminate/Disconnect All					
C66xx_3: GEL Output: Se	-						
C66xx_4: GEL Output: Se C66xx_4: GEL Output: Se		Properties					
C66xx 5: GEL Output: Set		lemory Map		_			
C66xx_5: GEL Output: Set							
		t 80002800 real 8000/				figPkg/compiler.opt" "	/srio_drv.c"
					'Finished buildin	g:/srio_drv.c	
[C66xx_2] fft size 128 o [C66xx_1] fft size 64 ou							
[C66xx_2] fft size 128 o [C66xx_1] fft size 64 ou [C66xx_6] fft size 32 ou	tput		00 imag 80006	800		SRIOSingleSRIO.out'	
[C66xx_2] fft size 128 o [C66xx_1] fft size 64 ou [C66xx_6] fft size 32 ou [C66xx_1] fft size 32 ou	tput	: 80001800 real 800074	00 imag 80006 00 imag 80005	800 400	'Invoking: C6000	Linker'	1
[C66xx_2] fft size 128 o [C66xx_1] fft size 64 ou [C66xx_6] fft size 32 ou [C66xx_1] fft size 32 ou [C66xx_1] fft size 32 ou [C66xx_2] fft size 64 ou	tput tput tput	: 80001800 real 800074 : 80002000 real 800078	00 imag 80006 00 imag 80005 00 imag 80005	800 400 800	'Invoking: C6000 "C:/ti/ccs/ccsv5/	Linker' tools/compiler/c6000/bin/	
[C66xx_2] fft size 128 o [C66xx_1] fft size 64 ou [C66xx_6] fft size 32 ou [C66xx_1] fft size 32 ou	tput tput tput utpu	: 80001800 real 800074 : 80002000 real 800078 : 80001800 real 80007	00 imag 80006 00 imag 80005 00 imag 80005 400 imag 8000	800 400 800 5400	'Invoking: C6000 "C:/ti/ccs/ccsv5/ define=_LITTLE_	Linker'	500
[C66xx_2] fft size 128 o [C66xx_1] fft size 64 ou [C66xx_6] fft size 32 ou [C66xx_1] fft size 32 ou [C66xx_2] fft size 64 ou [C66xx_1] fft size 128 o [C66xx_2] fft size 128 o [C66xx_2] fft size 128 o	tput tput utput utpu utpu	: 80001800 real 800074 : 80002000 real 800078 nt 80001800 real 80007 nt 80003000 real 80008 nt 80002000 real 80007	00 imag 80006 00 imag 80005 00 imag 80005 400 imag 80005 000 imag 8000 800 imag 8000	800 400 800 5400 6000 5800	'Invoking: C6000 "C:/ti/ccs/ccsv5/ define=_LITTLE_ define=TMS320C6 -m"SRIOSingleSRIO	Linker' tools/compiler/c6000/bin/ ENDIANdefine=_TMS320C6 600diag_warning=225: .map"warn_sections	500 abi=eabi -z
[C66xx_2] fft size 128 o [C66xx_1] fft size 64 ou [C66xx_6] fft size 32 ou [C66xx_6] fft size 32 ou [C66xx_2] fft size 64 ou [C66xx_1] fft size 128 o [C66xx_1] fft size 128 o [C66xx_2] fft size 128 o [C66xx_2] fft size 32 ou	tput tput utput utpu utpu utpu	: \$0001800 real \$00074 : \$0002000 real \$00078 it \$0001800 real \$00078 it \$0003000 real \$0008 it \$0002000 real \$00074 : \$0001800 real \$00074	00 imag 80006 00 imag 80005 00 imag 80005 400 imag 80005 000 imag 8000 800 imag 80005 000 imag 80005	800 400 800 5400 6000 5800 400	'Invoking: C6000 "C:/ti/ccs/ccsv5/ define=_LITTLE_ define=TMS320C6 -m"SRIDSingleSRIO -i"C:/ti/ccs/ccsv	Linker' tools/compiler/c6000/bin/ ENDIANdefine=_TMS320C6 600diag_warning=225: .map"warn_sections 5/tools/compiler/c6000/lil	600 abi=eabi -z
[C66xx_2] fft size 128 o [C66xx_1] fft size 64 ou [C66xx_6] fft size 32 ou [C66xx_1] fft size 32 ou [C66xx_2] fft size 64 ou [C66xx_1] fft size 128 o [C66xx_2] fft size 128 o [C66xx_2] fft size 128 o	tput tput utput utpu utpu tput	: 80001800 real 800074 : 80002000 real 800078 tt 80001800 real 80007 tt 80003000 real 80008 tt 80002000 real 80007 : 80001800 real 80007 tt 80002800 real 80007	00 imag 80006 00 imag 80005 00 imag 80005 400 imag 80005 000 imag 8000 800 imag 8000 00 imag 80005 00 imag 80005	800 400 800 5400 6000 5800 400 500	'Invoking: C6000 "C:/ti/ccs/ccsv5/ define=_LITTLE_ define=TMS320C6 -m"SRIOSingleSRIO -i"C:/ti/ccs/ccsv -i"C:/ti/ccs/dspl	Linker' tools/compiler/c6000/bin// ENDIANdefine=_TMS320C66 600diag_warning=225: .map"warn_sections 5/tools/compiler/c6000/lil bi_c66x_3_0_0_8/lib"	500 abi=eabi -z p"
<pre>[C66x2,2] fft size 128 o [C66x2,2] fft size 32 ou [C66x2,1] fft size 32 ou [C66x2,2] fft size 32 ou [C66x2,2] fft size 128 o [C66x2,2] fft size 128 o [C66x2,2] fft size 128 o [C66x2,2] fft size 230 ou [C66x2,1] fft size 250 o [C66x2,1] fft size 512 o [C66x2,1] fft size 512 o</pre>	tput tput utput utpu utpu tput utpu tput	: 80001800 real 800074 : 80002000 real 800078 :t 80003000 real 80007 :t 80003000 real 80007 : 80001800 real 80007 : 80001800 real 80007 :t 80001800 real 80007 : 80001800 real 80007	00 imag 80006 00 imag 80005 00 imag 80005 000 imag 8000 000 imag 8000 800 imag 8000 800 imag 8000 00 imag 8000 00 imag 8000 00 imag 8000	800 400 800 5400 6000 5800 400 5000 5000 5000 5400 800	'Invoking: G6000 "C:/ti/ccs/ccsv5/ define=_LITUE_ define=TM5320C6 -m"SRIOSingleSRIO -i"c:/ti/ccs/ccsv -i"c:/ti/ccs/ccsv -i"C:/ti/ccs/ccsv -rom_model -o "S	Linker <sup>*</sup> tools/compiler/c6000/bin// ENDIANdefine=_TMS320C6 600diag_warning=25: .map <sup>**</sup> warn_sections 5/tools/compiler/c6000/lin 15/tools/compiler/c6000/in S/tools/compiler/c6000/in RIOSingleSRI0.out <sup>**</sup> -1 <sup>**</sup> ./c	500 abi=eabi -z o" clude"reread_libs onfigPkg/linker.cmd"
[C65x,2] fft size 128 o [C65x,2] fft size 64 ou [C65x,2] fft size 32 ou [C65x,2] fft size 32 ou [C65x,2] fft size 128 o [C65x,2] fft size 128 o [C65x,2] fft size 128 o [C65x,2] fft size 128 o [C65x,2] fft size 325 o [C65x,2] fft size 325 o [C65x,2] fft size 325 o [C65x,2] fft size 32 o	tput tput utpu utpu utpu tput utput tput	: 80001800 real 800074 : 80002000 real 800075 tt 80001200 real 800075 tt 80001800 real 80007 tt 80002000 real 80007 tt 80002000 real 80007 tt 80001800 real 800075 : 80002000 real 800076	00 imag 80006 00 imag 80005 00 imag 80005 400 imag 80005 400 imag 8000 800 imag 8000 000 imag 80005 600 imag 80005 400 imag 80005 00 imag 80005	800 400 5400 5400 6000 5800 400 5000 5400 5400 800 800	'Invoking: C6000 "C:/ti/ccs/csv5/ define=IITLE define=TMS320C6 -m"SRIOSingleSRIO -i"C:/ti/ccs/csv -i"C:/ti/ccs/csv -rom_model -o "S "./srio_drv.obj"	Linker <sup>*</sup> tools/compiler/c6000/bin// ENDIANdefine=_TMS32066 600diag_warning=25 map <sup>*</sup> warn_sections 5/tools/compiler/c6000/lin 16_c60X_30_0 0/lin 5/tools/compiler/c6000/in RIOSingleSRI0.out <sup>*</sup> -l <sup>*</sup> /c(r ./slaveTask.obj <sup>**</sup> ./requ	500 abi=eabi -z o" clude"reread_libs onfigPkg/linker.cmd"
<pre>[C66x2,2] fft size 128 o [C66x2,2] fft size 32 ou [C66x2,1] fft size 32 ou [C66x2,2] fft size 32 ou [C66x2,2] fft size 128 o [C66x2,2] fft size 128 o [C66x2,2] fft size 128 o [C66x2,2] fft size 230 ou [C66x2,1] fft size 250 o [C66x2,1] fft size 512 o [C66x2,1] fft size 512 o</pre>	tput tput utpu utpu utpu tput utput tput	: 80001800 real 800074 : 80002000 real 800075 tt 80001200 real 800075 tt 80001800 real 80007 tt 80002000 real 80007 tt 80002000 real 80007 tt 80001800 real 800075 : 80002000 real 800076	00 imag 80006 00 imag 80005 00 imag 80005 400 imag 80005 400 imag 8000 800 imag 8000 000 imag 80005 600 imag 80005 400 imag 80005 00 imag 80005	800 400 5400 5400 6000 5800 400 5000 5400 5400 800 800	'Invoking: C6000 "C:/ti/ccs/ccsv5/ define=LITILE_ define=TMS320C6 -m"SRIOSingleSRIO -i"C:/ti/ccs/ccsv -i"C:/ti/ccs/ccsv -rom_model -o "S "./srio_drv.obj" ./qms_drv.obj"	Linker' tools/compiler/c6000/bin// ENDIANdefine_TM532066 600diag warning=225 .map"warn_sections 5/tools/compiler/c6000/li 10_c60x_3_0_0_0/li 5/tools/compiler/c6000/in KTOSingleSR10.out" -1"./c "./slaveTask.obj" "./requ "./mss_device.obj"	500 abi=eabi -z o" clude"reread_libs onfigPkg/linker.cmd" estProcessingData.obj"
[G65x,2] fft size 128 o [G65x,2] fft size 32 ou [G65x,6] fft size 32 ou [G65x,2] fft size 32 ou [G65x,2] fft size 32 ou [G65x,2] fft size 128 o [G65x,2] fft size 128 o [G65x,2] fft size 128 o [G65x,2] fft size 32 ou [G65x,2] fft size 32 ou	tput tput utpu utpu utpu tput utput tput	: 80001800 real 800074 : 80002000 real 800075 tt 80001200 real 800075 tt 80001800 real 80007 tt 80002000 real 80007 tt 80002000 real 80007 tt 80001800 real 800075 : 80002000 real 800076	00 imag 80006 00 imag 80005 00 imag 80005 400 imag 80005 400 imag 8000 800 imag 8000 000 imag 80005 600 imag 80005 400 imag 80005 00 imag 80005	800 400 5400 5400 6000 5800 400 5000 5400 5400 800 800	<pre>'Invoking: C6000 'C:/fics/ccsv/c -defineITTLE -define_ISI206 -m"SRI0SingleSRI0 -1"C:/fi/ccs/csv -1"C:/fi/ccs/csv -rom.model -0 "S ',/srio_drv.obj" './qmis_drv.obj ''./multicoreLoopb './initialization</pre>	Linker' tools/compler/c6000/bin// ENDIANdefine=_TMS320c6 600diag.warnige:225 .map"warn_sections 5/tools/compiler/c6000/ln 10.c6x3_0_0_8/lib 5/tools/compiler/c6000/ln RtOsingleSRL0.out" -1/.cqu ./qms_device.obj" ack_osal.obj" ./masterTa .obj" ./emt ruidel fftl	500 abi=eabi -z o" clude"reread_libs onfigPkg/linker.cmd" estProcessingData.obj" sk.obj" sk.obj"
[C65x,2] fft size 128 o [C65x,2] fft size 64 ou [C65x,2] fft size 32 ou [C65x,2] fft size 32 ou [C65x,2] fft size 128 o [C65x,2] fft size 128 o [C65x,2] fft size 128 o [C65x,2] fft size 128 o [C65x,2] fft size 325 o [C65x,2] fft size 325 o [C65x,2] fft size 325 o [C65x,2] fft size 32 o	tput tput utpu utpu utpu tput utput tput	: 80001800 real 800074 : 80002000 real 800075 tt 80001200 real 800075 tt 80001800 real 80007 tt 80002000 real 80007 tt 80002000 real 80007 tt 80001800 real 800075 : 80002000 real 800076	00 imag 80006 00 imag 80005 00 imag 80005 400 imag 80005 400 imag 8000 800 imag 8000 000 imag 80005 600 imag 80005 400 imag 80005 00 imag 80005	800 400 5400 5400 6000 5800 400 5000 5400 5400 800 800	<pre>'Invoking: C6000 'C:/fics/ccsv/c -defineITTLE -define_ISI206 -m"SRI0SingleSRI0 -1"C:/fi/ccs/csv -1"C:/fi/ccs/csv -rom.model -0 "S ',/srio_drv.obj" './qmis_drv.obj ''./multicoreLoopb './initialization</pre>	Linker' tools/compler/c6000/bin// ENDIANdefine=_TMS320c6 600diag.warnige:225 .map"warn_sections 5/tools/compiler/c6000/ln 10.c6x3_0_0_8/lib 5/tools/compiler/c6000/ln RtOsingleSRL0.out" -1/.cqu ./qms_device.obj" ack_osal.obj" ./masterTa .obj" ./emt ruidel fftl	500 abi=eabi -z o" clude"reread_libs onfigPkg/linker.cmd" estProcessingData.obj" sk.obj" sk.obj"
[C65x,2] fft size 128 o [C65x,2] fft size 64 ou [C65x,4] fft size 32 ou [C65x,2] fft size 32 ou [C65x,2] fft size 24 ou [C65x,2] fft size 128 o [C65x,2] fft size 128 o [C65x,2] fft size 128 o [C65x,2] fft size 256 o [C65x,2] fft size 256 o [C65x,2] fft size 30 o	tput tput utpu utpu utpu tput utput tput	: 80001800 real 800074 : 80002000 real 800075 tt 80001200 real 800075 tt 80001800 real 80007 tt 80002000 real 80007 tt 80002000 real 80007 tt 80001800 real 800075 : 80002000 real 800076	00 imag 80006 00 imag 80005 00 imag 80005 400 imag 80005 400 imag 8000 800 imag 8000 000 imag 80005 600 imag 80005 400 imag 80005 00 imag 80005	800 400 5400 5400 6000 5800 400 5000 5400 5400 800 800	'Invoking: C6000 "C:/tics/scss/ -define=LITHLE -define=TM5320C6 m"SHOSingleSRU -i"C:/ti/css/cssy -i"C:/ti/css/cssy -rom model -o "S ",/sric_drv.obj" ",/qms_drv.obj" ",/qms_drv.obj" ",/fftRoutines.ob "./fftRoutines.ob	Linker <sup>*</sup> tools/compiler/6600/bin/ ENDIANdefine=_TMS32066 660diag_warning=225: map <sup>*</sup> warn sections 5/tools/compiler/66000/lin tic66x 3.0 g.0%/lbin 5/tools/compiler/6600/in KTOSingleSRIO.out <sup>**</sup> -1 <sup>*</sup> ./cu *./slaveTask.obj <sup>**</sup> /maguration *./agus_device.obj <sup>***</sup> /maguration *./agus_device.obj <sup>****</sup> /maguration *./agus_device.obj <sup>****</sup> /maguration *./agus_device.obj <sup>*****</sup> /maguration *./agus_device.obj <sup>******</sup> /maguration *./agus_device.obj <sup>*********</sup> /maguration *./agus_device.obj <sup>************************************</sup>	500 abi=eabi -z of clude"reread libs onfigPkg/linker.cmd" stProcessingData.obj' sk.obj" sk.05)" sk.obj" .obj" ./bioMain.obj'

End of Procedure 4-3

## Procedure 4-4 Load and Run

## Step - Action

- 1 Select Group1 and connect all cores in the group:
  - **1a** From the CCS *Run* menu, select *Connect Target*. OR
  - **1b** Right click on the group name and choose *Connect Target*. OR
  - **1c** Click the Connect Target icon.
- 2 Load the *SRIOSingleSRIO.out* to all cores in the group:
  - 2a From the *Run* menu, select *Load*.
    - OR
  - 2b Click the Load icon

- **3** Run the code in one of the following ways:
  - 3a Press F8
  - **3b** From the *Run* menu, select *Resume*
  - **3c** Click on the *Resume* icon (green arrow).
    - The output results appear as shown in Figure 4-3:

### Figure 4-3 SRIOSingleSRIO Run Results

т co	CS Deb	ug - C	ode Comp	oser Stud	lio					2				6	
File	Edit	View	Search	Project	Tools	Run	Scripts	Windo	w	Help					
	-	ē	<b>-</b>	🖉 🕶 🖁	3 🕄	• •		<b>*</b>	参	•	N	•			
券□	)ebug	×			× 🕨	88 8	3	<u> </u>	9	_62	۲	- 🕹	🏟		' 🗆
⊿ §	🗿 ma:	zanin.o	cxml [Cod	le Compo	oser Stud	dio - De	vice Deb	ugging]							
	4 🎉	Group	1												
		🔎 Bla	ckhawk X	DS560v2-	USB Me	zzanine	Emulato	r_0/C66	xx_0	(Runn	ing)				
		🔎 Bla	ackhawk X	DS560v2-	USB Me	zzanine	Emulato	r_0/C66	xx_1	(Runn	ing)				
		🔎 Bla	ackhawk X	DS560v2-	USB Me	zzanine	Emulato	r_0/C66	xx_2	(Runn	ing)				
		🔎 Bla	ackhawk X	DS560v2-	USB Me	zzanine	Emulato	r_0/C66	xx_3	(Runn	ing)				
		🔎 Bla	ackhawk X	DS560v2-	USB Me	zzanine	Emulato	r_0/C66	xx_4	(Runn	ing)				
		🔎 Bla	ackhawk X	DS560v2-	USB Me	zzanine	Emulato	r_0/C66	xx_5	(Runn	ing)				
		🔎 Bla	ackhawk X	DS560v2-	USB Me	zzanine	Emulato	r_0/C66	xx_6	(Runn	ing)				
		🔎 Bla	ackhawk X	DS560v2-	USB Me	zzanine	Emulato	r_0/C66	xx_7	(Runn	ing)				

📃 Console 🛛

<pre>[C66xx_0] Debug(Core 0): System Initialization for CPPI &amp; QMSS [C66xx_0] This is Core 0, at this point ALL cores finished create and bind [C66xx_1] fft size 64 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 64 output 80002800 real 80007400 imag 80005800 [C66xx_3] fft size 128 output 80002800 real 80007400 imag 80005800 [C66xx_2] fft size 256 output 80001800 real 80007400 imag 80005400 [C66xx_1] fft size 256 output 80002800 real 80007400 imag 80005400 [C66xx_3] fft size 256 output 80002800 real 80007400 imag 80005400 [C66xx_4] fft size 256 output 80002800 real 80007400 imag 80005400 [C66xx_5] fft size 512 output 80002800 real 80007400 imag 80005400 [C66xx_6] fft size 512 output 80002800 real 80007400 imag 80005400 [C66xx_6] fft size 512 output 80002800 real 80007400 imag 80005800 [C66xx_1] fft size 512 output 80002800 real 80007400 imag 80005800 [C66xx_1] fft size 512 output 80002800 real 80007400 imag 80005800 [C66xx_2] fft size 32 output 80001800 real 80007400 imag 80005800 [C66xx_1] fft size 32 output 80001800 real 80007400 imag 80005800 [C66xx_1] fft size 32 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 128 output 80001800 real 80007400 imag 80005400 [C66xx_1] fft size 128 output 80001800 real 80007400 imag 80005400 [C66xx_1] fft size 128 output 80001800 real 80007400 imag 80005400 [C66xx_1] fft size 128 output 80001800 real 80007400 imag 80005400 [C66xx_1] fft size 128 output 80001800 real 80007400 imag 80005400 [C66xx_1] fft size 128 output 80002000 real 80007400 imag 80005400 [C66xx_1] fft size 128 output 80002000 real 80007400 imag 80005400 [C66xx_2] fft size 128 output 80002000 real 80007400 imag 80005400 [C66xx_2] fft size 128 output 80002000 real 80007400 imag 80005400 [C66xx_2] fft size 256 output 80002000 real 80007400 imag 80005400 [C66xx_2] fft size 266 output 80002000 real 80007400 imag 80005400 [C66xx_2] fft size 128 output 80002000 real 80007400 imag 80005400 [C66xx_2] fft size 128 output 80002000 real 80007800 imag 80005400 [C66xx_2] fft size 128 output 80</pre>	mazanin.ccxn	nl:CIO
[C66xx_0] This is Core 0, at this point ALL cores finished create and bind [C66xx_1] fft size 64 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 64 output 80002800 real 80007600 imag 80005600 [C66xx_3] fft size 128 output 80001800 real 80007400 imag 80005400 [C66xx_1] fft size 256 output 80001800 real 80007400 imag 80005800 [C66xx_1] fft size 256 output 80002800 real 80007400 imag 80005600 [C66xx_3] fft size 256 output 80002800 real 80007400 imag 80005600 [C66xx_4] fft size 256 output 80002800 real 8000700 imag 8000500 [C66xx_4] fft size 256 output 80002800 real 8000700 imag 8000500 [C66xx_3] fft size 512 output 80002800 real 8000700 imag 8000500 [C66xx_4] fft size 512 output 80002800 real 8000700 imag 8000500 [C66xx_4] fft size 512 output 80002800 real 8000700 imag 8000500 [C66xx_2] fft size 512 output 80002800 real 80007800 imag 80005800 [C66xx_1] fft size 32 output 80002000 real 80007400 imag 80005800 [C66xx_1] fft size 32 output 80002000 real 80007400 imag 80005800 [C66xx_2] fft size 32 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 32 output 80001800 real 80007400 imag 80005400 [C66xx_1] fft size 32 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 128 output 80001800 real 80007400 imag 80005800 [C66xx_2] fft size 128 output 80001800 real 80007400 imag 80005800 [C66xx_4] fft size 128 output 80001800 real 80007400 imag 80005800 [C66xx_4] fft size 128 output 80002000 real 80007400 imag 80005400 [C66xx_4] fft size 128 output 80002000 real 80007400 imag 80005400 [C66xx_2] fft size 128 output 80002000 real 80007400 imag 80005400 [C66xx_2] fft size 128 output 80002000 real 80007400 imag 80005400 [C66xx_2] fft size 128 output 80002000 real 80007400 imag 80005400 [C66xx_2] fft size 32 output 80002000 real 80007400 imag 80005400 [C66xx_2] fft size 32 output 80002000 real 80007400 imag 80005400 [C66xx_2] fft size 128 output 80002000 real 80007400 imag 80005400 [C66xx_5] fft size 128 output 80002000 real 80007400 imag 80005400	[C66xx_0]	Debug(Core 0): System Initialization for CPPI & QMSS
[C66xx_1] fft size 64 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 64 output 80002000 real 80007800 imag 80005800 [C66xx_3] fft size 128 output 80002800 real 80007400 imag 80005400 [C66xx_1] fft size 32 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 256 output 80002000 real 80007400 imag 80005400 [C66xx_3] fft size 128 output 80002800 real 80007400 imag 80005400 [C66xx_4] fft size 256 output 80002800 real 80007400 imag 80005400 [C66xx_4] fft size 128 output 80002800 real 80007400 imag 80005400 [C66xx_5] fft size 512 output 80002800 real 80007400 imag 80005400 [C66xx_5] fft size 512 output 80002800 real 80007400 imag 80005400 [C66xx_6] fft size 512 output 80002800 real 80007400 imag 80005800 [C66xx_1] fft size 512 output 80002800 real 80007400 imag 80005800 [C66xx_1] fft size 520 output 80002800 real 80007400 imag 80005800 [C66xx_1] fft size 32 output 80002000 real 80007400 imag 80005800 [C66xx_1] fft size 32 output 80002000 real 80007400 imag 80005800 [C66xx_2] fft size 128 output 80002000 real 80007400 imag 80005400 [C66xx_2] fft size 32 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 128 output 80002000 real 80007400 imag 80005400 [C66xx_2] fft size 128 output 80001800 real 80007400 imag 80005400 [C66xx_1] fft size 128 output 80002000 real 80007400 imag 80005400 [C66xx_2] fft size 128 output 80002000 real 80007400 imag 80005400 [C66xx_2] fft size 128 output 80002000 real 80007400 imag 80005400 [C66xx_1] fft size 128 output 80002000 real 80007400 imag 80005400 [C66xx_2] fft size 128 output 80002000 real 80007400 imag 80005400 [C66xx_2] fft size 128 output 80002000 real 80007400 imag 80005400 [C66xx_2] fft size 128 output 80002000 real 80007400 imag 80005400 [C66xx_2] fft size 32 output 80002000 real 80007400 imag 80005400 [C66xx_2] fft size 128 output 80002000 real 80007400 imag 80005400 [C66xx_2] fft size 128 output 80002000 real 80007400 imag 80005400 [C66xx_5] fft size 128 output 80003000 real 80007400 imag 80005400	[C66xx_0]	
[C66xx_2] fft size 64 output 80002000 real 80007800 imag 80005800 [C66xx_3] fft size 128 output 80002800 real 80007400 imag 80005400 [C66xx_1] fft size 32 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 256 output 80002000 real 80007400 imag 80005800 [C66xx_3] fft size 256 output 80002800 real 80007400 imag 80005600 [C66xx_3] fft size 128 output 80002800 real 80007600 imag 80005600 [C66xx_4] fft size 256 output 80003800 real 80007600 imag 80005600 [C66xx_5] fft size 512 output 80003800 real 80007600 imag 80005400 [C66xx_2] fft size 512 output 80002800 real 80007600 imag 80005400 [C66xx_2] fft size 512 output 80002800 real 80007600 imag 80005400 [C66xx_2] fft size 512 output 80002800 real 80007400 imag 80005400 [C66xx_6] fft size 520 output 80001800 real 80007400 imag 80005400 [C66xx_6] fft size 32 output 80001800 real 80007400 imag 80005400 [C66xx_1] fft size 32 output 80001800 real 80007400 imag 80005800 [C66xx_1] fft size 32 output 80001800 real 80007400 imag 80005800 [C66xx_2] fft size 64 output 80001800 real 80007400 imag 80005800 [C66xx_2] fft size 128 output 80001800 real 80007400 imag 80005800 [C66xx_1] fft size 128 output 80001800 real 80007400 imag 80005800 [C66xx_2] fft size 128 output 80001800 real 80007400 imag 80005800 [C66xx_1] fft size 128 output 80001800 real 80007400 imag 80005800 [C66xx_2] fft size 128 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 128 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 128 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 128 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 128 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 128 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 120 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 120 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 120 output 80001800 real 80007400 imag 80005400 [C66xx_5] fft size 128 output 80003000 real 80007400 imag 80005400	[C66xx_0]	This is Core 0, at this point ALL cores finished create and bind
[C66xx_3] fft size 128 output 80002800 real 80007c00 imag 80005c00 [C66xx_1] fft size 32 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 256 output 80002000 real 80007400 imag 80005800 [C66xx_1] fft size 256 output 80001800 real 8000700 imag 80005000 [C66xx_3] fft size 128 output 80002800 real 8000700 imag 80006000 [C66xx_4] fft size 256 output 80003000 real 80008400 imag 80006000 [C66xx_5] fft size 512 output 80002800 real 8000700 imag 80006000 [C66xx_2] fft size 512 output 80002800 real 8000700 imag 80005000 [C66xx_2] fft size 512 output 80002800 real 8000700 imag 80005000 [C66xx_2] fft size 512 output 80002800 real 80007400 imag 80005400 [C66xx_6] fft size 32 output 80001800 real 80007400 imag 80005400 [C66xx_1] fft size 32 output 80001800 real 80007400 imag 80005400 [C66xx_1] fft size 32 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 64 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 64 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 128 output 80001800 real 80007400 imag 80005400 [C66xx_1] fft size 128 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 128 output 80001800 real 80007400 imag 80005400 [C66xx_1] fft size 128 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 128 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 128 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 128 output 80001800 real 80007400 imag 80005400 [C66xx_1] fft size 128 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 128 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 120 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 120 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 120 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 120 output 80002800 real 80007800 imag 80005400 [C66xx_5] fft size 128 output 80003800 real 80007400 imag 80005400		
[C66xx_1] fft size 32 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 256 output 80002000 real 80007400 imag 80005400 [C66xx_1] fft size 256 output 80001800 real 80007400 imag 80005400 [C66xx_3] fft size 128 output 80002800 real 80007400 imag 80006400 [C66xx_4] fft size 256 output 80003800 real 80008400 imag 80006400 [C66xx_5] fft size 512 output 80002800 real 80007400 imag 80005400 [C66xx_2] fft size 512 output 80002800 real 80007400 imag 80005400 [C66xx_2] fft size 128 output 80002000 real 80007400 imag 80005800 [C66xx_1] fft size 64 output 80001800 real 80007400 imag 80005400 [C66xx_6] fft size 32 output 80004000 real 80007400 imag 80005400 [C66xx_2] fft size 64 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 64 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 128 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 128 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 128 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 128 output 80001800 real 80007400 imag 80005400 [C66xx_3] fft size 128 output 80001800 real 80007400 imag 80005400 [C66xx_4] fft size 128 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 128 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 128 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 128 output 80001800 real 80007400 imag 80005400 [C66xx_3] fft size 256 output 80002000 real 80007400 imag 80005400 [C66xx_4] fft size 512 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 64 output 80002000 real 80007400 imag 80005400 [C66xx_4] fft size 64 output 80002000 real 80007400 imag 80005400 [C66xx_4] fft size 64 output 80002000 real 80007400 imag 80005400 [C66xx_5] fft size 128 output 80003800 real 80007400 imag 80005400		
[C66xx_2] fft size 256 output 80002000 real 80007800 imag 80005800 [C66xx_1] fft size 256 output 80001800 real 80007400 imag 80005400 [C66xx_3] fft size 128 output 80002800 real 80007c00 imag 80005c00 [C66xx_4] fft size 256 output 80003000 real 80008000 imag 80006000 [C66xx_5] fft size 512 output 80002800 real 80007c00 imag 80005c00 [C66xx_3] fft size 512 output 80002800 real 80007c00 imag 80005c00 [C66xx_2] fft size 128 output 80002000 real 80007400 imag 80005800 [C66xx_1] fft size 64 output 80001800 real 80007400 imag 80005400 [C66xx_6] fft size 32 output 80004000 real 80007400 imag 80005400 [C66xx_2] fft size 64 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 64 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 64 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 128 output 80001800 real 80007400 imag 80005400 [C66xx_1] fft size 128 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 128 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 128 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 128 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 128 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 128 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 128 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 128 output 80001800 real 80007400 imag 80005400 [C66xx_3] fft size 256 output 80002800 real 80007400 imag 80005400 [C66xx_2] fft size 512 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 64 output 80002800 real 80007400 imag 80005400 [C66xx_2] fft size 64 output 80002800 real 80007400 imag 80005400 [C66xx_4] fft size 32 output 80003800 real 80007400 imag 80005400 [C66xx_5] fft size 128 output 80003800 real 80007400 imag 80005400		
[C66xx_1] fft size 256 output 80001800 real 80007400 imag 80005400 [C66xx_3] fft size 128 output 80002800 real 80007c00 imag 80005c00 [C66xx_4] fft size 256 output 80003000 real 80008000 imag 80006000 [C66xx_5] fft size 512 output 80002800 real 80007c00 imag 80005c00 [C66xx_3] fft size 512 output 80002800 real 80007c00 imag 80005c00 [C66xx_2] fft size 128 output 80002000 real 80007400 imag 80005400 [C66xx_1] fft size 64 output 80001800 real 80007400 imag 80005400 [C66xx_6] fft size 32 output 80004000 real 80007400 imag 80005400 [C66xx_1] fft size 64 output 80002000 real 80007400 imag 80005400 [C66xx_1] fft size 64 output 80002000 real 80007400 imag 80005400 [C66xx_2] fft size 64 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 128 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 128 output 80001800 real 80007400 imag 80005400 [C66xx_4] fft size 128 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 128 output 80001800 real 80007400 imag 80005800 [C66xx_2] fft size 128 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 128 output 80002000 real 80007400 imag 80005400 [C66xx_2] fft size 128 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 120 output 80001800 real 80007400 imag 80005400 [C66xx_3] fft size 256 output 80002800 real 80007400 imag 80005400 [C66xx_2] fft size 512 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 128 output 80002800 real 80007400 imag 80005400 [C66xx_4] fft size 32 output 80002800 real 80007400 imag 80005400 [C66xx_5] fft size 128 output 80002800 real 80007400 imag 80005400 [C66xx_5] fft size 128 output 80002800 real 80007400 imag 80005400		
[C66xx_3] fft size 128 output 80002800 real 80007c00 imag 80005c00 [C66xx_4] fft size 256 output 80003000 real 80008000 imag 80006000 [C66xx_5] fft size 512 output 80003800 real 80007c00 imag 80005c00 [C66xx_3] fft size 512 output 80002800 real 80007c00 imag 80005c00 [C66xx_2] fft size 128 output 80002000 real 80007400 imag 80005400 [C66xx_1] fft size 64 output 80002000 real 80007400 imag 80005400 [C66xx_6] fft size 32 output 80004000 real 80007400 imag 80005400 [C66xx_1] fft size 32 output 80001800 real 80007800 imag 80005400 [C66xx_2] fft size 64 output 80001800 real 80007800 imag 80005400 [C66xx_2] fft size 64 output 80001800 real 80007800 imag 80005400 [C66xx_2] fft size 128 output 80001800 real 80007800 imag 80005400 [C66xx_2] fft size 128 output 80001800 real 80007800 imag 80005400 [C66xx_2] fft size 128 output 80001800 real 80007800 imag 80005800 [C66xx_3] fft size 128 output 80002000 real 80007800 imag 80005800 [C66xx_2] fft size 128 output 80002000 real 80007800 imag 80005800 [C66xx_1] fft size 128 output 80002000 real 80007800 imag 80005800 [C66xx_2] fft size 128 output 80002800 real 80007400 imag 80005400 [C66xx_2] fft size 256 output 80002800 real 80007400 imag 80005400 [C66xx_2] fft size 512 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 512 output 80002800 real 80007400 imag 80005400 [C66xx_2] fft size 128 output 80002800 real 80007400 imag 80005400 [C66xx_2] fft size 128 output 80002800 real 80007400 imag 80005400 [C66xx_3] fft size 128 output 80002800 real 80007400 imag 80005400 [C66xx_4] fft size 32 output 80002800 real 80007400 imag 80005400 [C66xx_5] fft size 128 output 80003800 real 80007400 imag 80005400		
[C66xx_4] fft size 256 output 80003000 real 80008000 imag 80006000 [C66xx_5] fft size 512 output 80003800 real 80007400 imag 80005400 [C66xx_3] fft size 512 output 80002800 real 80007400 imag 80005800 [C66xx_2] fft size 128 output 80001800 real 80007400 imag 80005400 [C66xx_1] fft size 64 output 80001800 real 80007400 imag 80005400 [C66xx_1] fft size 32 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 32 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 64 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 64 output 80002000 real 80007400 imag 80005400 [C66xx_2] fft size 128 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 128 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 128 output 80002000 real 80007400 imag 80005400 [C66xx_3] fft size 32 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 128 output 80002000 real 80007400 imag 80005400 [C66xx_2] fft size 128 output 80002000 real 80007400 imag 80005400 [C66xx_2] fft size 128 output 80002800 real 80007400 imag 80005400 [C66xx_2] fft size 256 output 80002800 real 80007400 imag 80005400 [C66xx_2] fft size 512 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 512 output 80002800 real 80007400 imag 80005400 [C66xx_2] fft size 512 output 80002800 real 80007400 imag 80005400 [C66xx_2] fft size 64 output 80002800 real 80007400 imag 80005400 [C66xx_4] fft size 32 output 80003800 real 80007400 imag 80005400 [C66xx_5] fft size 128 output 80003800 real 80007400 imag 80005400		
[C66xx_5] fft size 512 output 80003800 real 80008400 imag 80006400 [C66xx_3] fft size 512 output 80002800 real 80007c00 imag 80005c00 [C66xx_2] fft size 128 output 80002000 real 80007800 imag 80005800 [C66xx_1] fft size 64 output 80001800 real 80007400 imag 80005400 [C66xx_6] fft size 32 output 80004000 real 80007400 imag 80005400 [C66xx_1] fft size 32 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 64 output 80001800 real 80007400 imag 80005400 [C66xx_1] fft size 128 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 128 output 80001800 real 80007400 imag 80005400 [C66xx_4] fft size 128 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 128 output 80002000 real 80007400 imag 80005400 [C66xx_3] fft size 128 output 80001800 real 80007400 imag 80005400 [C66xx_3] fft size 32 output 80001800 real 80007400 imag 80005400 [C66xx_1] fft size 32 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 128 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 512 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 512 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 512 output 80002800 real 80007400 imag 80005400 [C66xx_2] fft size 128 output 80002800 real 80007400 imag 80005400 [C66xx_2] fft size 128 output 80002800 real 80007400 imag 80005400 [C66xx_2] fft size 128 output 80002800 real 80007400 imag 80005400 [C66xx_2] fft size 128 output 80002800 real 80007400 imag 80005400 [C66xx_5] fft size 128 output 80003800 real 80007400 imag 80005400		
[C66xx_3] fft size 512 output 80002800 real 80007c00 imag 80005c00 [C66xx_2] fft size 128 output 80002000 real 80007800 imag 80005800 [C66xx_1] fft size 64 output 80001800 real 80007400 imag 80005400 [C66xx_6] fft size 32 output 80004000 real 80007400 imag 80005400 [C66xx_1] fft size 32 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 64 output 80002000 real 80007400 imag 80005400 [C66xx_1] fft size 128 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 128 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 128 output 80002000 real 80007400 imag 80005400 [C66xx_2] fft size 128 output 80002000 real 80007400 imag 80005800 [C66xx_3] fft size 32 output 80001800 real 80007400 imag 80005400 [C66xx_3] fft size 32 output 80001800 real 80007400 imag 80005400 [C66xx_3] fft size 256 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 64 output 80002000 real 80007400 imag 80005400 [C66xx_3] fft size 512 output 80001800 real 80007400 imag 80005400 [C66xx_4] fft size 512 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 128 output 80002000 real 80007400 imag 80005400 [C66xx_4] fft size 512 output 80002000 real 80007400 imag 80005400 [C66xx_5] fft size 64 output 80002000 real 80007400 imag 80005400 [C66xx_5] fft size 128 output 80003000 real 80007400 imag 80005400		
[C66xx_2] fft size 128 output 80002000 real 80007800 imag 80005800 [C66xx_1] fft size 64 output 80001800 real 80007400 imag 80005400 [C66xx_6] fft size 32 output 80004000 real 80007400 imag 80005400 [C66xx_1] fft size 32 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 64 output 80002000 real 80007400 imag 80005400 [C66xx_4] fft size 128 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 128 output 80002000 real 80007400 imag 80005400 [C66xx_2] fft size 128 output 80002000 real 80007400 imag 80005400 [C66xx_2] fft size 128 output 80002000 real 80007400 imag 80005800 [C66xx_3] fft size 32 output 80001800 real 80007400 imag 80005400 [C66xx_3] fft size 256 output 80002800 real 80007400 imag 80005400 [C66xx_1] fft size 512 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 64 output 80002000 real 80007400 imag 80005400 [C66xx_2] fft size 512 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 128 output 80002000 real 80007400 imag 80005400 [C66xx_2] fft size 128 output 80002000 real 80007400 imag 80005400 [C66xx_2] fft size 128 output 80002000 real 80007400 imag 80005400 [C66xx_2] fft size 128 output 80002000 real 80007400 imag 80005400 [C66xx_4] fft size 128 output 80003000 real 80007400 imag 80005400 [C66xx_5] fft size 128 output 80003000 real 8000800 imag 80005400		
[C66xx_1] fft size 64 output 80001800 real 80007400 imag 80005400 [C66xx_6] fft size 32 output 80004000 real 80008800 imag 80006800 [C66xx_1] fft size 32 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 64 output 80001800 real 80007400 imag 80005400 [C66xx_1] fft size 128 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 128 output 80002000 real 80007400 imag 80005800 [C66xx_2] fft size 128 output 80002000 real 80007400 imag 80005800 [C66xx_3] fft size 128 output 80001800 real 80007400 imag 80005800 [C66xx_3] fft size 32 output 80001800 real 80007400 imag 80005400 [C66xx_3] fft size 256 output 80002800 real 80007400 imag 80005400 [C66xx_2] fft size 512 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 526 output 80002800 real 80007400 imag 80005400 [C66xx_3] fft size 512 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 64 output 80002000 real 80007400 imag 80005400 [C66xx_2] fft size 64 output 80002000 real 80007400 imag 80005400 [C66xx_4] fft size 128 output 80003000 real 80007400 imag 80005400 [C66xx_5] fft size 128 output 80003000 real 8000800 imag 80005400		
[C66xx_6] fft size 32 output 80004000 real 80008800 imag 80006800 [C66xx_1] fft size 32 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 64 output 80002000 real 80007400 imag 80005800 [C66xx_1] fft size 128 output 80001800 real 80007400 imag 80005400 [C66xx_4] fft size 128 output 80002000 real 80007400 imag 80005800 [C66xx_2] fft size 128 output 80002000 real 80007800 imag 80005800 [C66xx_1] fft size 32 output 80001800 real 80007400 imag 80005400 [C66xx_3] fft size 256 output 80002800 real 80007400 imag 80005400 [C66xx_1] fft size 512 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 64 output 80002800 real 80007400 imag 80005400 [C66xx_2] fft size 64 output 80002000 real 80007400 imag 80005400 [C66xx_2] fft size 64 output 80002000 real 80007400 imag 80005800 [C66xx_4] fft size 32 output 80002000 real 80007400 imag 80005800 [C66xx_5] fft size 128 output 80003000 real 8000800 imag 80006000 [C66xx_5] fft size 128 output 80003800 real 80008400 imag 80006400		
[C66xx_1] fft size 32 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 64 output 80002000 real 80007800 imag 80005800 [C66xx_1] fft size 128 output 80001800 real 80007400 imag 80005400 [C66xx_4] fft size 128 output 80002000 real 80007400 imag 80006000 [C66xx_2] fft size 128 output 80002000 real 80007800 imag 80005800 [C66xx_1] fft size 32 output 80001800 real 80007400 imag 80005400 [C66xx_3] fft size 256 output 80002800 real 80007400 imag 80005400 [C66xx_1] fft size 512 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 64 output 80002800 real 80007400 imag 80005400 [C66xx_2] fft size 64 output 80002000 real 80007400 imag 80005400 [C66xx_4] fft size 32 output 80002000 real 80007400 imag 80005400 [C66xx_5] fft size 128 output 80003000 real 80008400 imag 80006400		
[C66xx_2] fft size 64 output 80002000 real 80007800 imag 80005800 [C66xx_1] fft size 128 output 80001800 real 80007400 imag 80005400 [C66xx_4] fft size 128 output 80003000 real 80008000 imag 80006000 [C66xx_2] fft size 128 output 80002000 real 80007800 imag 80005800 [C66xx_1] fft size 32 output 80001800 real 80007400 imag 80005400 [C66xx_3] fft size 256 output 80002800 real 80007400 imag 80005400 [C66xx_1] fft size 512 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 64 output 80002000 real 80007400 imag 80005400 [C66xx_4] fft size 64 output 80002000 real 80007400 imag 80005800 [C66xx_5] fft size 128 output 80003000 real 80008000 imag 80006000 [C66xx_5] fft size 128 output 80003800 real 80008400 imag 80006400		
[C66xx_1] fft size 128 output 80001800 real 80007400 imag 80005400 [C66xx_4] fft size 128 output 80003000 real 80008000 imag 80006000 [C66xx_2] fft size 128 output 80002000 real 80007800 imag 80005800 [C66xx_1] fft size 32 output 80001800 real 80007400 imag 80005400 [C66xx_3] fft size 256 output 80002800 real 80007400 imag 80005400 [C66xx_1] fft size 512 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 64 output 80002000 real 80007400 imag 80005800 [C66xx_4] fft size 32 output 80003000 real 80008000 imag 80006000 [C66xx_5] fft size 128 output 80003800 real 80008400 imag 80006400		
[C66xx_4] fft size 128 output 80003000 real 80008000 imag 80006000 [C66xx_2] fft size 128 output 80002000 real 80007800 imag 80005800 [C66xx_1] fft size 32 output 80001800 real 80007400 imag 80005400 [C66xx_3] fft size 256 output 80002800 real 80007c00 imag 80005c00 [C66xx_1] fft size 512 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 64 output 80002000 real 80007400 imag 80005800 [C66xx_4] fft size 32 output 80003000 real 80008000 imag 80006000 [C66xx_5] fft size 128 output 80003800 real 80008400 imag 80006400		
<pre>[C66xx_2] fft size 128 output 80002000 real 80007800 imag 80005800 [C66xx_1] fft size 32 output 80001800 real 80007400 imag 80005400 [C66xx_3] fft size 256 output 80002800 real 80007c00 imag 80005c00 [C66xx_1] fft size 512 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 64 output 80002000 real 80007800 imag 80005800 [C66xx_4] fft size 32 output 80003000 real 80008000 imag 80006000 [C66xx_5] fft size 128 output 80003800 real 80008400 imag 80006400</pre>		
[C66xx_1] fft size 32 output 80001800 real 80007400 imag 80005400 [C66xx_3] fft size 256 output 80002800 real 80007c00 imag 80005c00 [C66xx_1] fft size 512 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 64 output 80002000 real 80007800 imag 80005800 [C66xx_4] fft size 32 output 80003000 real 80008000 imag 80006000 [C66xx_5] fft size 128 output 80003800 real 80008400 imag 80006400		
[C66xx_3] fft size 256 output 80002800 real 80007c00 imag 80005c00 [C66xx_1] fft size 512 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 64 output 80002000 real 80007800 imag 80005800 [C66xx_4] fft size 32 output 80003000 real 80008000 imag 80006000 [C66xx_5] fft size 128 output 80003800 real 80008400 imag 80006400		
[C66xx_1] fft size 512 output 80001800 real 80007400 imag 80005400 [C66xx_2] fft size 64 output 80002000 real 80007800 imag 80005800 [C66xx_4] fft size 32 output 80003000 real 80008000 imag 80006000 [C66xx_5] fft size 128 output 80003800 real 80008400 imag 80006400		
[C66xx_2] fft size 64 output 80002000 real 80007800 imag 80005800 [C66xx_4] fft size 32 output 80003000 real 80008000 imag 80006000 [C66xx_5] fft size 128 output 80003800 real 80008400 imag 80006400		
[C66xx_4] fft size 32 output 80003000 real 80008000 imag 80006000 [C66xx_5] fft size 128 output 80003800 real 80008400 imag 80006400		
[C66xx_5] fft size 128 output 80003800 real 80008400 imag 80006400		
•	[[[00000_3]	110 2176 150 Outhor 00002000 Lear 00000400 TugB 00000400
4		

- 4 Observe the results, then suspend the run:
  - **4a** From the *Run* menu, choose *Suspend*.

OR

**4b** Click on the *Suspend* icon (the yellow "pause" lines)

End of Procedure 4-4

# **Chapter 5**

# Optimization

## 5.1 Purpose

The goal of this exercise is to demonstrate some basic optimization techniques. This exercise works on any KeyStone EVM board. It may also be used with the simulator in conjunction with the estimated cycle count.

## 5.2 Project Files

The following files are used in this lab:

- firMain.c
- intrinsicCFilters.c
- linker.cmd
- naturalCFilters.c
- test.h
- utilities.c

## 5.3 Instructions

In the first part of the exercise, you will build, load and run a project on the EVM without optimization. In the second part, you will enable optimization and analyze the results.

The list of processes used in this example are as follows:

- Procedure 5-1 "Build and Run the Project"
- Procedure 5-2 "Connect to the EVM"
- Procedure 5-3 "Load and Run the Program"
- Procedure 5-4 "Compiler Optimization"
- Procedure 5-5 "Enable Software Pipelining"
- Procedure 5-6 "Align the Data"
- Procedure 5-7 "Enable the MUST\_ITERATE Pragma"
- Procedure 5-8 "Cache Considerations"

Procedure 5-1	Build and Run the Project

Step - Action

1 Open CCS.

- 2 Create new project through the CCS menu item File  $\rightarrow$  New  $\rightarrow$  CCS Project.
- 3 Enter Optimization as a Project Name.
- 4 Click the check box to Use default location.
- 5 Set the Family to C6000 and Variant to Generic C66xxx Device as shown in Figure 5-1:

#### Figure 5-1 CCS Project Settings

	ect		
CCS Project			
Create a new C	CS Project.		
Project name:	optimization		
Output type:	Executable		•
🔽 Use default	location		
Location:	C:\Users\a0270985\WorkSpaces	\temp\optimization	Browse
Device			
Family:	C6000		•
Variant:	<select filter="" or="" text="" type=""></select>	← Generic C66xx Device	•
Connection:			
type filter text	plates and examples	Creates an empty project fully	/ initialized 🔺
	Projects Appty Project (with main.c) The pro	for the selected device.	
En ■ En ■ En ■ Basic I ■ He	npty RTSC Project		~

Note—The screen shots may reflect different location for the project.

- 6 Then press *Finish* to create the new project.
- 7 Then in the *Project Explorer* view, right-click on the newly-created *optimization* project, and click on *Add Files...*
- 8 Browse to the project directory you created for this exercise and select all required files as outlined at the beginning of this exercise, then click *Open*.

- **9** When prompted how files should be imported into the project, leave it as default of *Copy File*. Remove the main.c file that is created by default when you created the new project.
- **10** Examine the code in 'firMain.c' to understand the functions that are being called. The *generateData* function generates the data sets to be operated on. Functions *naturalCFilters* and *intrinsicCfilters* execute filters on the generated data. The former is implemented completely in C, while the latter takes advantage of compiler intrinsics.
- 11 Set the properties for the *Debug* configuration. Right-click on the project. Select *Properties*.
  - **11a** Choose *Build*, click on the *Environment* tab, and click the *Add*...button to add the path to add a variable with *Name* as 'PDK\_ROOT' and *Value* as 'C:\ti\mcsdk\pdk\_C6678\_1\_1\_2\_5'

**11b** Choose *C6000 Compiler* → *Optimization* and set/verify the following properties:

- 'Optimization level = 0'
- 'Optimize for code size = 0'

**11c** Choose *C6000 Compiler* → *Debug Options* and set/verify the following properties:

- 'Debugging model = Full symbolic debug'
- **11d** Choose *C6000 Compiler* → *Include Options*. Under the "Add dir to #include search path" add the following two paths:
  - "\${PDK\_ROOT}/packages/ti/csl"
  - "\${PDK\_ROOT}/packages"

**Note**—This ensures that any include references in the project's source files to header files located at these paths will be interpreted accurately.

- 12 Click the OK button to save the project properties and close the Properties window.
- **13** Right-click on the project and select *Build Project*. A successful build will generate the following output on the console:

• • •

<Linking>

'Finished building target: optimization.out'

\*\*\*\* Build Finished \*\*\*\*

End of Procedure 5-1

## Procedure 5-2 Connect to the EVM

#### Step - Action

- 1 Click the Open Perspective (available right top corner of the CCS).
- 2 Switch to the Debug Perspective by selecting the CCS menu option Window  $\rightarrow$  Open Perspective  $\rightarrow$  CCS Debug.
- 3 Select the CCS menu option *View* → *Target Configurations*. Select the target configuration you created
- 4 Launch the target configuration as follows:
  - **4a** Select the target configuration .ccxml file.
  - **4b** Right click and select *Launch Selected Configuration*.
- 5 This will bring up the *Debug* window.
  - **5a** Select Core 0 (*C66x\_0*)
  - **5b** Right click and select *Connect Target*.

End of Procedure 5-2

		Doc ID: 12222
Proc	edure	e 5-3 Load and Run the Program
Ste	p - A	ction
1	Ena	ble the Clock by selecting the CCS menu option $\mathit{Run}  o \mathit{Clock}  o \mathit{Enable}$
2	Sele	ect Core 0 and load the .out file created earlier in the lab.
	2a	Select the CCS menu option $Run  ightarrow Load  ightarrow Load Program$
	2b	Click Browse project
	2c	Select optimization.out by unwrapping the OptimizationDebug and click OK.
	2d	Click OK to load the application to the target (Core 0).
3	Rur	the application by selecting the CCS menu option $\mathit{Run}  ightarrow \mathit{Resume}.$
4		uccessful run should produce a console output as shown below. Record the cycles time for h natural C and intrinsic C versions:
[C66	5xx_0	natural C code size 32768 time 3889442
[C66	5xx_0	intrinsic C code size 32768 time 2809073
C66	5xx_0	no error was found !!!
C66	5xx 0	
CGE	- 5xx 0]	
	_	
1000	·	] DONE
	edure p - Ae	
1		ve back to the CCS <i>Edit</i> perspective.
2		r will now set the properties for the <i>Release</i> configuration. This suppresses all debug feature I enables the highest time optimization.
	2a	Right-click on the Optimization project. Select Build Configurations $\rightarrow$ Set Active $\rightarrow$ Release
3	Rig	ht-click on the Optimization Project. Select Properties.
	3a	Choose <i>Build</i> , click on the <i>Environment</i> tab, and click the <i>Add</i> button to add the path to add a variable with <i>Name</i> as 'PDK_ROOT' and <i>Value</i> as 'C:\ti\mcsdk\pdk_C6678_1_1_2_5
	Not	t <b>e</b> —The path and platform may differ for your local devices.
	3b	Select C6000 Compiler $\rightarrow$ Optimization and set/verify the following properties:
		• 'Optimization level = 3'
	3c	Choose C6000 Compiler $\rightarrow$ Debug Options and ensure that:
		<ul> <li>'Debugging model = Suppress all symbolic debug generation'</li> </ul>
	3d	beougging model - suppress an symbolic debug generation
		Choose C6000 Compiler →Include Options. Under the "Add dir to #include search path" add
		Choose C6000 Compiler $\rightarrow$ Include Options. Under the "Add dir to #include search path" add the following two paths:
		Choose C6000 Compiler → Include Options. Under the "Add dir to #include search path" add the following two paths: • "\${PDK_ROOT}/packages/ti/csl"
4	loca	<ul> <li>Choose C6000 Compiler → Include Options. Under the "Add dir to #include search path" add the following two paths:</li> <li>"\${PDK_ROOT}/packages/ti/csl"</li> <li>"\${PDK_ROOT}/packages"</li> <li>te—This ensures that any include references in the project's source files to header files</li> </ul>

**5** Right-click on the project and select *Build Project*. A successful build will generate the following output on the console:

· · · ·

'Finished building target: optimization.out'

```
**** Build Finished ****
```

- **6** Enable the Clock by selecting the CCS menu option  $Run \rightarrow Clock \rightarrow Enable$  (if you have done in the previous section of this lab, you can ignore this step).
- 7 Select Core 0 and load the .out file created earlier in the lab.
  - **7a** Select the CCS menu option  $Run \rightarrow Load \rightarrow Load$  Program
  - **7b** Click Browse project...
  - 7c Select optimization.out by unwrapping the OptimizationRelease and click OK.
  - 7d Click OK to load the application to the target (Core 0).
- 8 Run the application by selecting the CCS menu option  $Run \rightarrow Resume$ .
- **9** A successful run should produce a console output as shown below. Record the optimized cycle times for both natural C and intrinsic C versions:
- [C66xx\_0] natural C code size 32768 time 228698

```
[C66xx_0] intrinsic C code size 32768 time 1282213
```

[C66xx\_0] no error was found !!!

[C66xx\_0]

[C66xx\_0]

[C66xx\_0] DONE

End of Procedure 5-4

## **QUESTIONS:**

How much improvement is noted for the natural C code?

How much improvement is noted for the intrinsic code?

What issues exist within the code, if any?



Note—Consider how intrinsic functions utilize the processor.

Procedure 5-5	Enable Software Pipelining	
<u>Ci</u> <u>A</u> i'		

## Step - Action

- 1 In the CCS Project Explorer go to Build  $\rightarrow$  C6000 Compiler  $\rightarrow$  Advanced Options  $\rightarrow$  Assembler Options and check the box that says Keep the generated assembly language (.asm) file
- 2 Rebuild the code.
- **3** The generated assembly file will be located within the *Release* directory since you are building the project's release configuration.

### **QUESTIONS:**

Open the 'intrinsicCFilter.asm' file and answer the following questions:

- Was the compiler able to schedule the software pipeline?
- What are the general reasons that the compiler might not schedule the software pipeline?

Note—Think about cases that can cause randomness in the execution timing.

• What reason can you see that the compiler might not be able to schedule the software pipeline?

## \_\_\_\_\_

**Note**—Think about the inline function.

- 4 Replace the regular function with the intrinsic function in all the loops. Look at the definition of the regular function and see what intrinsic it uses)
- 5 Rebuild the code, load, and run.
- **6** Look at the *intrinsicCFilter.asm*.

#### **QUESTIONS:**

Did the compiler schedule the software pipeline?

Record the optimized project cycles time for natural C function and for intrinsic function with software pipeline.

End of Procedure 5-5

## Procedure 5-6 Align the Data

#### Step - Action

1 In the *intrinsicCFilter.c* code, the data is read from the memory.

## QUESTIONS:

- What is the alignment of the input data?
- What is the alignment of the filter coefficients (in the stack)?

**Note**—Find the pragma that aligns the data. Consider other ways to align the data on a 64-bit boundary.

- 2 Change the code to tell the compiler that the data is loaded from aligned memory. (the \_amemX intrinsic tells the compiler that the data is aligned on 64 bit)
- **3** Rebuild the code, load, and run.
- 4 Record the optimized project cycle time for natural C function and for intrinsic function with software pipeline and aligned load.

#### End of Procedure 5-6

#### Procedure 5-7 Enable the MUST\_ITERATE Pragma

## Step - Action

- 1 Uncomment the code to enable the pragmas that tell the compiler the minimum number of iterations and the divisor.
- 2 Rebuild the code, load, and run
- **3** Record the optimized project cycle time for natural C function and for intrinsic function with software pipeline, aligned load, and MUST\_ITERATE pragma

End of Procedure 5-7.

Procedure 5-8 Cache Considerations

```
Step - Action
```

1 In *test.h*, change the number of elements to 2K, 4K, 8K and 16K

2 In Table 5-1, record the cycle counts for each case.

## Table 5-1 Clock Values & Cycle Counts

Size	Multiply for 32K	Clock Value	Cycles for 32K
32K	1		
16K	2		
8K	4		
4K	8		
2K	16		
End of Table	5-1		

## QUESTION:

Why the non-linear jump in performance?

**Note**—Think about cache trashing.

End of Procedure 5-8

## 5.3.1 Cache Analysis

## QUESTION:

Think about the data size and the fact that float complex requires 8 bytes (single precision). What is the actual size of the data?

**Note**—To understand better the cache issue, you should consider performing the cache debug exercise in this document.

## 5.3.2 Change the Code to Speed Up to 32K

Change the code to increase the speed to 32K and take full advantage of the cache.

## \_\_\_\_

**Note**—Break the data into chunks and call each routine multiple times. Make sure to keep the sum between calls as well as the pointer to the data.

## **Chapter 6**

# **Using Advanced Debug**

## 6.1 Purpose

In the optimization exercise, the number of cycles is non-linear with the number of elements. When the number of elements is 8K, the code is much faster than if the number of elements is 16K; Per element, the numbers are normalized.

Recall that the data resides in L2 memory and that L1 D is configured as all cache. The data is read from L2 memory and is put into the L1 cache for reusability.

The obvious explanation is that when the number of elements is 16K, the cache is trashed during the first filter. So the second filter has a cache miss. The same is true for the third filter and the fourth.

The elements in the optimization exercise are floating-point complex, single-precision numbers. So each element requires 8 bytes; Single-precision is 32-bits, equal to 4 bytes, and floating point complex doubles the number of bytes to 8.

In this exercise, you will use the debug features in Code Composer Studio (CCS) to better understand the cache behavior and determine why the higher results start at 8K and not at 4K.

## 6.1.1 Why the Debug Version is Used

For the purpose of reading data from L1 cache, the optimization is not important. This would not be the case if the optimized code in the release writes out intermediate results (and then reads it later) and the debug version does not. But this is not the case. Both versions read and write the same information. Thus, for this exercise, the debug version is used with non-optimization and full symbolic debug turned on.

## 6.2 Instructions

First, you will look at the cache behavior when the number of elements is 4K. Next, you will do the same with 16K elements. Lastly, you will look at the 8K elements use the information derived from the previous two tasks to draw a conclusion.

The list of processes used in this example are as follows:

- Procedure 6-1 "View the 4K Case"
- Procedure 6-2 "Looking at the Cache Lines for 4K Case"
- Procedure 6-3 "View the Cache Lines for 16K Case"
- Procedure 6-4 "View the Cache Lines for 8K Case"

Proc	edure 6-1	View the 4K Case		
Step	o - Action			
1	Change the	number of elements in the test.h file a	s follows:	
	#define	NUMBER_OF_ELEMENTS	4096	
2	Rebuild the	code.		

- **3** Launch the debugger, connect Core 0 to the emulator and load the code from the debug configuration (the one with no-optimization and full symbolic debug).
- 4 From the *Run* menu in the *Debug* perspective, enable the *Clock*.
- **5** Open the file intrinsicCfilters.c and put a breakpoint after the first filter.

6 Run the code and verify that it stopped at this breakpoint, which should appear as shown in Figure 6-1:

Figure 6-1 CCS Debug: IntrinsicCFilters Breakpoint

📬 🗕 🔚	💼 🛍 🎄 v 📳 % 😃 v 😿 😂 v 🔗 v		
S Debug		10101 Registers (X)= Variable	s 🔍 Brea
	6678Trace.ccxml [Code Composer Studio - Device Debugging]	Identity	Name
🔺 🧽 🔇	Group 1	□	
4 0	Blackhawk XDS560v2-USB System Trace Emulator_0/C66xx_0 (Suspended - SW Breakpoint)	v 🧟 intrinsicCFilter	
	≡ intrinsicC_filters(float *, int, float *)() at intrinsicCFilters.c:47 0x0C0019F4	V Vas Intrinsicer inter	ысакроп
	0x20202020 (no symbols are defined for 0x20202020)		
×	Blackhawk XDS560v2-USB System Trace Emulator_0/C66xx_1 (Disconnected : Unknown)		
~	Blackhawk XDS560v2-USB System Trace Emulator_0/C66xx_2 (Disconnected : Unknown)		
~	Blackhawk XDS560v2-USB System Trace Emulator_0/C66xx_3 (Disconnected : Unknown)		
	Blackhawk XDS560v2-USB System Trace Emulator_0/C66xx_4 (Disconnected : Unknown)		
	Blackhawk XDS560v2-USB System Trace Emulator_0/C66xx_5 (Disconnected : Unknown)		
	Blackhawk XDS560v2-USB System Trace Emulator_0/C66xx_6 (Disconnected : Unknown)		
×	Blackhawk XDS560v2-USB System Trace Emulator_0/C66xx_7 (Disconnected : Unknown)		
		L	
h tect h	C c int000 at /tmp/TL MKLIBIwBz81/SBC/boot c:87 0xc006720	atrinsic/Filters c 32	
		ntrinsicCFilters.c 🛛	
h test.h 35 36	Cint00() at /tmp/TI_MKLIBIwBz8I/SRC/boot.c:87 0xc006720       Image: firMain.c         Image: State	ntrinsicCFilters.c 🛛	
	}	ntrinsicCFilters.c 🛛	
35 36		ntrinsicCFilters.c 🛛	
35 36 37 38 39	} } total_sum = _hif2(sum) * _hif2(sum) + _lof2(sum) * _lof2(sum);	ntrinsicCFilters.c 🛛	
35 36 37 38 39 40	<pre>} } total_sum = _hif2(sum) * _hif2(sum) + _lof2(sum) * _lof2(sum); *output++ = total_sum; /*</pre>	ntrinsicCFilters.c 🛛	
35 36 37 38 39 40 41	} } total_sum = _hif2(sum) * _hif2(sum) + _lof2(sum) * _lof2(sum);	ntrinsicCFilters.c 🛛	
35 36 37 38 39 40	<pre>} } total_sum = _hif2(sum) * _hif2(sum) + _lof2(sum) * _lof2(sum); *output++ = total_sum; /* * End of filter 1</pre>	ntrinsicCFilters.c 🛛	
35 36 37 38 39 40 41 42 43 44	<pre>} } total_sum = _hif2(sum) * _hif2(sum) + _lof2(sum) * _lof2(sum); *output++ = total_sum; /* * End of filter 1 */ /*</pre>		
35 36 37 38 39 40 41 42 43 44 45	<pre>} } total_sum = _hif2(sum) * _hif2(sum) + _lof2(sum) * _lof2(sum); *output++ = total_sum; /* * End of filter 1 */ /* * Second filter, coefficients are (1/4 -1/4) (1/2 -1/2) (-1/4 1/4) (</pre>		
35 36 37 38 39 40 41 42 43 44 45 46	<pre>} } total_sum = _hif2(sum) * _hif2(sum) + _lof2(sum) * _lof2(sum); *output++ = total_sum; /* * End of filter 1 */ /* * Second filter, coefficients are (1/4 -1/4) (1/2 -1/2) (-1/4 1/4) (-: */</pre>		
35 36 37 38 39 40 41 42 43 44 45	<pre>} } total_sum = _hif2(sum) * _hif2(sum) + _lof2(sum) * _lof2(sum); *output++ = total_sum; /* * End of filter 1 */ /* * Second filter, coefficients are (1/4 -1/4) (1/2 -1/2) (-1/4 1/4) (-1/4) p_in = (float2_t *)inputComplex;</pre>		
35 36 37 38 39 40 41 42 43 44 45 46 47	<pre>} } total_sum = _hif2(sum) * _hif2(sum) + _lof2(sum) * _lof2(sum); *output++ = total_sum; /* * End of filter 1 */ /* * Second filter, coefficients are (1/4 -1/4) (1/2 -1/2) (-1/4 1/4) (-: */</pre>		
35 36 37 38 39 40 41 42 43 44 44 45 46 45 46 49 50	<pre> } total_sum = _hif2(sum) * _hif2(sum) + _lof2(sum) * _lof2(sum); *output++ = total_sum;  /* * End of filter 1 */ /* * Second filter, coefficients are (1/4 -1/4) (1/2 -1/2) (-1/4 1/4) (-: */ p_in = (float2_t *)inputComplex; filter_size = 4; x[0] =ftof2(0.25, -0.25); </pre>		
35 36 37 38 39 40 41 42 43 44 43 44 45 46 47 48 49 50 51	<pre> } total_sum = _hif2(sum) * _hif2(sum) + _lof2(sum) * _lof2(sum); *output++ = total_sum; /* * End of filter 1 */ /* * Second filter, coefficients are (1/4 -1/4) (1/2 -1/2) (-1/4 1/4) (-* */ p in = (_float2_t *)inputComplex; filter_size = 4; x[0] = _ftof2(0.25, -0.25); x[1] = _ftof2(0.5, -0.5); </pre>		
35 36 37 38 39 40 41 42 43 44 45 46 45 46 47 48 49 50	<pre> } total_sum = _hif2(sum) * _hif2(sum) + _lof2(sum) * _lof2(sum); *output++ = total_sum;  /* * End of filter 1 */ /* * Second filter, coefficients are (1/4 -1/4) (1/2 -1/2) (-1/4 1/4) (-: */ p_in = (float2_t *)inputComplex; filter_size = 4; x[0] =ftof2(0.25, -0.25); </pre>		
35 36 37 38 39 40 41 42 43 44 45 46 45 46 47 48 49 50 51 51 51 52 €	<pre>} } total_sum = _hif2(sum) * _hif2(sum) + _lof2(sum) * _lof2(sum); *output++ = total_sum; /* * End of filter 1 */ /* * Second filter, coefficients are (1/4 -1/4) (1/2 -1/2) (-1/4 1/4) (-* */ p_in = (_float2_t *)inputComplex; filter_size = 4; x[0] = _ftof2(0.25, -0.25); x[1] = _ftof2(0.5, -0.25); x[2] = _ftof2(-0.25, 0.25);</pre>		
35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 51 52 € 47 Console	<pre> } total_sum = _hif2(sum) * _hif2(sum) + _lof2(sum) * _lof2(sum); *output++ = total_sum; /* * End of filter 1 */ /* * Second filter, coefficients are (1/4 -1/4) (1/2 -1/2) (-1/4 1/4) (-* */ p in = (_float2_t *)inputComplex; filter_size = 4; x[0] = _ftof2(0.25, -0.25); x[1] = _ftof2(0.5, -0.5); </pre>		

End of Procedure 6-1

#### Procedure 6-2 Looking at the Cache Lines for 4K Case

#### Step - Action

1 Left-click on the *View* tab in the *Debug* perspective. In the pull-down menu, choose *Other*. A new window will open, as shown in Figure 6-2.

#### Figure 6-2 CCS Debug: Show View

	View Project Tools Run Scripts Window Help			
📬 🗕 🔡	😨 🖾 🏇 🕶 📳 🏷 🖉 🕶 🕅 🏷 💣 🕶 🔗 🕶			
🌣 Debug 🎖	x 🐘 🕺 🐢 🕫 🖉 🔳 💷 🐗 🐒	>	1010 Registers (×)= Variable	es 🔍 🕒 Breakp
⊿ 😜 evm ⊿ 🤐 ( ∠ 4 0 × × × ×	6678Trace.ccxml [Code Composer Studio - Device Debugging] Group 1 Blackhawk XDS560v2-USB System Trace Emulator_0/C66∞_0 (Suspended - SW B ■ intrinsicC_filters(float *, int, float *)() at intrinsicCFilters.c47 0x0C0019F4 ■ 0x20202020 (no symbols are defined for 0x20202020) Blackhawk XDS560v2-USB System Trace Emulator_0/C66∞_1 (Disconnected)	Breakpoint) Show View type filter tex ▷ ⇔ Gene ▷ ⇔ Analy ▷ ⇔ C/C+ ▲ ⇔ Debu	Identity	Name
		🚺 C	reakpoints ache ontrol Panel	
b         test.h           35         36           37         38           39         40           41         42           43         44           45         5	<pre>[Ec_int00() at /tmp/TI_MKLIBIwBz8I/SRC/boot.c:87 0xc006720</pre>	<ul> <li>□</li> <li>□</li> <li>□</li> <li>○</li> <li>○</li></ul>	ache ontrol Panel	-

## 2 Select *Cache* and double click to open the *Cache* window, as shown in Figure 6-3.

## Figure 6-3 CCS Debug: IntrinsicCFilters Cache (4K)

h test.h	📴 _c_int00() at /tmp/TI_MKLIBIwBz8I/SRC/boot.c:87 0xc006720 🕼 firMain.c 🕼 intrinsicCFilters.c 🛛
35	}
36	
37	total_sum = _hif2(sum) * _hif2(sum) + _lof2(sum) * _lof2(sum);
38	*output++ = total sum;
39	
40	/*
41	* End of filter 1
42	*/
43	
44	/*
45	* Second filter, coefficients are (1/4 -1/4) (1/2 -1/2) (-1/4 1/4) (-1/2 1/2)
46	*/
47	<pre>p in = (_float2 t *)inputComplex;</pre>
48	filter size = 4;
49	-
50	x[0] = _ftof2(0.25, -0.25);
51	x[1] = [ftof2(0.5, -0.5);
52	x[2] = ftof2(-0.25, 0.25);

	- , , , -							
(i) Cache Tag	RAM Read complete: 1536	Cache Lines read fro	om targ	jet (33 a	fter filte	ring)		
Cache	Line Start Adrs	Line End Adrs	Set	Way	Valid	Dirty	LRU Way	Symbols In Cache
L1D cache	0x00800000	0x00803FFF			V	-	-	
L1D cache	0x00808CC0	0x00808DBF		1	V	D	L	
L1D cache	0x00808F00	0x0080903F			V	D		
L1D cache	0x0080A000	0x0080A03F	128	1	V	D	L	
L1D cache	0x0080A1C0	0x0080A1FF	135	1	V	-	L	1 symbols (Double click to view cache line details).
L1D cache	0x0080A280	0x0080A2BF	138	1	V	-	L	I symbols (Double click to view cache line details).
L1D cache	0x0080A300	0x0080A33F	140	0	V	D	L	
L1D cache	0x0080A440	0x0080A47F	145	0	V	D	L	
L1P cache	0x0C0005C0	0x0C0007DF		0	۷			

**3** Double-click on the *Cache* tab to enlarge the window. Then double-click on any line of the L1D to display all L1D lines, as shown in Figure 6-4.

Figure 6-4	CCS Debug: IntrinsicCFilters L1D Cache Lines (4K)

CCS De	bug - Code Composer Stu	dio)						
ile Edit	View Project Tools	Run Scripts Window	/ Help	)				
📬 🖣 🛛	- <b>*</b> -	🔳 🎭 🚇 🕶 🔊	• -	A	•			
Conso	le 🔟 Memory Map 🌐	Table 🚺 Cache 🛛						
Cache	Tag RAM Read complete:	1536 Cache Lines read fro	om targ	et (33 a	fter filte	ring)		
Cache	Line Start Adrs	Line End Adrs	Set	Way	Valid	Dirty	LRU Way	Symbols In Cache
L1D cache	0x00800000	0x0080003F	0	0	V	-	-	inputComplex (0x00800000)
L1D cache	0x00800040	0x0080007F	1	0	V	-	-	inputComplex (+ 1 Line)
L1D cache	0x00800080	0x008000BF	2	0	V	-	-	inputComplex (+ 2 Lines)
L1D cache	0x008000C0	0x008000FF	3	0	V	-	-	inputComplex (+ 3 Lines)
L1D cache	0x00800100	0x0080013F	4	0	V	-	-	inputComplex (+ 4 Lines)
L1D cache	0x00800140	0x0080017F	5	0	V	-	-	inputComplex (+ 5 Lines)
L1D cache	0x00800180	0x008001BF	6	0	V	-	-	inputComplex (+ 6 Lines)
1D cache	0x008001C0	0x008001FF	7	0	V	-	-	inputComplex (+ 7 Lines)
1D cache	0x00800200	0x0080023F	8	0	V	-	-	inputComplex (+ 8 Lines)
.1D cache	0x00800240	0x0080027F	9	0	V	-	-	inputComplex (+ 9 Lines)
.1D cache	0x00800280	0x008002BF	10	0	V	-	-	inputComplex (+ 10 Lines)
1D cache	0x008002C0	0x008002FF	11	0	V	-	-	inputComplex (+ 11 Lines)
1D cache	0x00800300	0x0080033F	12	0	v	-	-	inputComplex (+ 12 Lines)
.1D cache	0x00800340	0x0080037F	13	0	V	-	-	inputComplex (+ 13 Lines)
1D cache	0x00800380	0x008003BF	14	0	V	-	-	inputComplex (+ 14 Lines)
1D cache	0x008003C0	0x008003FF	15	0	V	-	-	inputComplex (+ 15 Lines)
1D cache	0x00800400	0x0080043F	16	0	v	-	-	inputComplex (+ 16 Lines)
1D cache	0x00800440	0x0080047F	17	0	v	-	-	inputComplex (+ 17 Lines)
1D cache	0x00800480	0x008004BF	18	0	V	-	-	inputComplex (+ 18 Lines)
1D cache	0x008004C0	0x008004FF	19	0	v	-	-	inputComplex (+ 19 Lines)
1D cache	0x00800500	0x0080053F	20	0	v	-	-	inputComplex (+ 20 Lines)
L1D cache	0x00800540	0x0080057F	21	0	v	-	-	inputComplex (+ 21 Lines)
L1D cache	0x00800580	0x008005BF	22	0	v	-	-	inputComplex (+ 22 Lines)
L1D cache		0x008005FF	23	0	V	-	-	inputComplex (+ 23 Lines)
L1D cache			24	0	v	-	-	inputComplex (+ 24 Lines)
L1D cache		0x0080067F	25	0	v	-	-	inputComplex (+ 25 Lines)
L 1D cache		0x008006BF	26	0	V	-	-	inputComplex (+ 26 Lines)
L 1D cache			27	0	V	-	-	inputComplex (+ 27 Lines)
1D cache			28	0	V	-	-	inputComplex (+ 28 Lines)
1D cache		0x0080077F	29	0	V	-	-	inputComplex (+ 29 Lines)
1D cache			30	0	v	-	-	inputComplex (+ 30 Lines)
L 1D cache			31	0	v	-		inputComplex (+ 31 Lines)
L 1D cache			32	0	v	-		inputComplex (+ 32 Lines)
L 1D cache			33	0	v	-	-	inputComplex (+ 33 Lines)
I 1D cache			34	0	v			inputComplex (+ 34 Lines)

4 Examine the first L1D line. The address of the first L1D line is 0x0080 0000, which is the first line of the L2 memory and where the input vector resides. Notice that the valid flag is set for this cache line.

#### 5 Next, look at the last L1D cache line as shown in Figure 6-5:

- The last line of the vector inputComples is line 255 with a starting address of 0x0080 3fC0.
- After this, there are several DIRTY lines, which indicate where the code changed some values.
- Remember that L1D cache line has 64 bytes (0x40). So the last byte of *inputComplex* in the cache is byte 0x00803fc0 +0x40 -1 = 0x00803fff. This is the Line End address of line 255.

## Figure 6-5 CCS Debug: IntrinsicCFilters L1D Cache (Last Line, 4K)

ile Edit Viev	Code Composer Studio w Project Tools Run	Scripts Window	Hel					
	,							
📑 🗕 🖪	🖾 🏇 🕶 🔜	🎭 🚇 👻 🔊	<b>*</b> -	Ň	•			
Console	Memory Map 🌐 Table	🚦 Cache 🔀						
Cache Tag R	AM Read complete: 1536	Cache Lines read fro	m targ	et (33 af	fter filte	ring)		
ache	Line Start Adrs	Line End Adrs	Set	Way	Valid	Dirty	LRU Way	Symbols In Cache
.1D cache	0x00803EC0	0x00803EFF	251	0	V	-	-	inputComplex (+ 251 Lines)
.1D cache	0x00803F00	0x00803F3F	252	0	V	-	-	inputComplex (+ 252 Lines)
.1D cache	0x00803F40	0x00803F7F	253	0	V	-	-	inputComplex (+ 253 Lines)
.1D cache	0x00803F80	0x00803FBF	254	0	V	-	-	inputComplex (+ 254 Lines)
.1D cache	0x00803FC0	0x00803FFF	255	0	V	-	-	inputComplex (+ 255 Lines)
.1D cache	0x00808CC0	0x00808CFF	51	1	V	D	L	
.1D cache	0x00808D00	0x00808D3F	52	1	V	D	L	
.1D cache	0x00808D40	0x00808D7F	53	1	V	D	L	
.1D cache	0x00808D80	ache line starts at 0x	008081	101	V	D	L	
.1D cache	0x00808F00	UX00808F3F	60	1	V	-	-	
.1D cache	0x00808F40	0x00808F7F	61	0	V	D	L	
.1D cache	0x00808F80	0x00808FBF	62	0	V	D	L	
.1D cache	0x00808FC0	0x00808FFF	63	0	V	D	L	
.1D cache	0x00809000	0x0080903F	64	1	V	D	L	_sys_memory (0x00809000)
.1D cache	0x0080A000	0x0080A03F	128	1	V	D	L	_ftable (0x0080A000)
.1D cache	0x0080A1C0	0x0080A1FF	135	1	V	-	L	_ftable (+ 7 Lines, End: 0x0080A1DF),_stream (0x
.1D cache	0x0080A280	0x0080A2BF	138	1	V	-	L	_device (0x0080A280)
.1D cache	0x0080A300	0x0080A33F	140	0	V	D	L	TI_enable_exit_profile_output (0x0080A300),_r
.1D cache	0x0080A440	0x0080A47F	145	0	V	D	L	_tmpnams (+ 5 Lines, End: 0x0080A45F),parmbuf
.1P cache	0x0C0005C0	0x0C0005DF	46	0	V			_getarg_diouxp (0x0C0005C0)
.1P cache	0x0C0005E0	0x0C0005FF	47	0	V			_getarg_diouxp (+ 1 Line)
.1P cache	0x0C000600	0x0C00061F	48	0	V			_getarg_diouxp (+ 2 Lines)
.1P cache	0x0C000620	0x0C00063F	49	0	V			_getarg_diouxp (+ 3 Lines)
L 1P cache	0x0C000640	0x0C00065F	50	0	V			_getarg_diouxp (+ 4 Lines)

## End of Procedure 6-2

#### **QUESTIONS:**

- What is the last cache line that has the *inputComplex* vector?
- How many bytes were read from the *inputComplex* vector?
- What is the number of elements that were read from the *inputComplex* vector?

Proc	edure 6-3	View the Cache L	ines for 16K	Case					
Ste	o - Action								
1	Change the	e number of elements	in the test h	filo to	16K				
•	change the		in the test.in		101				
	#define	e NUMBER OF H	ELEMENTS		1	L638	34		
2	Repeat all t	he previous steps as c	lefined in Tas	k1a	nd Ta	sk 2.			
-	•								
3		and run the code to t own in Figure 6-6:	he break poir	nt. Th	ie cac	he lir	nes sh	ould be	e the same or simi
Figu	re 6-6 C	CCS Debug: Intrinsic	Filters L1D	Cach	e Lin	es (10	5K)		
	🐨 CCS De	bug - Code Composer Studio							A
	File Edit	View Project Tools Run	Scripts Windov	v Hel	p				
	i 📬 👻 🛛		🎭 🛺 🗕 粉 🔡	<b>a</b> -	1	•			
				· ·	1.14				
		ile 🔟 Memory Map 🌐 Table			. (000	0 0			
		Tag RAM Read complete: 1536							
	Cache	Line Start Adrs	Line End Adrs	Set	Way	Valid	Dirty	LRU Way	Symbols In Cache
	L1D cache		0x0080803F	0	0	V	-	L	inputComplex (+ 512 Line
	L1D cache		0x0080807F	1	0	V	-	L	inputComplex (+ 513 Line
	L1D cache		0x008080BF	2	0	V	-	L	inputComplex (+ 514 Line
	L1D cache		0x008080FF	3	0	V	-	L	inputComplex (+ 515 Line
	L1D cache		0x0080813F	4	0	V	-	L	inputComplex (+ 516 Line
	L1D cache	0x00808140	0x0080817F	5	0	V	-	L	inputComplex (+ 517 Line
	L1D cache	0x00808180	0x008081BF	6	0	V	-	L	inputComplex ( 1 E18 Line Symbols:
	L1D cache	0x008081C0	0x008081FF	7	0	V	-	L	inputComplex (+ 5
	L1D cache	0x00808200	0x0080823F	8	0	V	-	L	inputcomplex (+ 520 Line
	L1D cache	0x00808240	0x0080827F	9	0	V	-	L	inputComplex (+ 521 Line
	L1D cache	0x00808280	0x008082BF	10	0	V	-	L	inputComplex (+ 522 Line
	L1D cache	0x008082C0	0x008082FF	11	0	V	-	L	inputComplex (+ 523 Line
	L1D cache	0x00808300	0x0080833F	12	0	V	-	L	inputComplex (+ 524 Line
	L1D cache	0x00808340	0x0080837F	13	0	V	-	L	inputComplex (+ 525 Line
	L1D cache	0x00808380	0x008083BF	14	0	V	-	L	inputComplex (+ 526 Line
	L1D cache	0x008083C0	0x008083FF	15	0	V	-	L	inputComplex (+ 527 Line
	L1D cache	0x00808400	0x0080843F	16	0	V	-	L	inputComplex (+ 528 Line
	L1D cache	0x00808440	0x0080847F	17	0	V	-	L	inputComplex (+ 529 Line
	L1D cache	0x00808480	0x008084BF	18	0	V	-	L	inputComplex (+ 530 Line
	L1D cache	0x008084C0	0x008084FF	19	0	V	-	L	inputComplex (+ 531 Line
	L1D cache	0x00808500	0x0080853F	20	0	V	-	L	inputComplex (+ 532 Line
	L1D cache	0x00808540	0x0080857F	21	0	V	-	L	inputComplex (+ 533 Line
	L1D cache	0x00808580	0x008085BF	22	0	V	-	L	inputComplex (+ 534 Line
	L1D cache		0x008085FF	23	0	V	-	L	inputComplex (+ 535 Line
	L1D cache	0x00808600	0x0080863F	24	0	V	-	L	inputComplex (+ 536 Line
	L1D cache		0x0080867F	25	0	٧	-	L	inputComplex (+ 537 Line
	L1D cache		0x008086BF	26	0	٧	-	L	inputComplex (+ 538 Line
	L1D cache		0x008086FF	27	0	٧	-	L	inputComplex (+ 539 Line
	L1D cache		0x0080873F	28	0	٧	-	L	inputComplex (+ 540 Line
	L1D cache		0x0080877F	29	0	V	-	L	inputComplex (+ 541 Line
			0x008087BF	30	0	v	-	L L	inputComplex (+ 542 Line
	L1D cache								

#### End of Procedure 6-3

## **QUESTIONS:**

- Why does the first entry address start with 0x00808000?
- What is the last cache line that has the inputComplex vector?
- How many bytes were read from the inputComplex vector? How do you know?
- What is the number of elements that were read from the inputComplex vector?

#### Step - Action

- 1 Change the number of elements in the test.h file to 8K.
- 2 Repeat all the previous steps as defined in Task 1 and Task 2.
- **3** Build, load, and run the code to the break point.

End of Procedure 6-4

## **QUESTIONS:**

Examine the cache.

- What is the first entry address? What does it mean?
- What is the last cache line that has the *inputComplex* vector?
- How many bytes were read from the *inputComplex* vector? How do you know?
- What is the number of elements that were read from the *inputComplex* vector?

## **Finding the Bug**

You should now understand that the number of elements as defined in *test.h* is not the number of elements that are actually read from the input vector.

Can you suggest a bug fix in *firMain.c* that will fix the problem? Write your answer below.

## **Chapter 7**

# **Using MPAX to Define Private Core Memory in DDR**

## 7.1 Purpose

A typical multicore scenario uses multiple cores to run the same code, but requires different data and configuration structures.

Downloading the same execution code for multiple cores is very efficient. However, since each core has its own configuration and data structures, these values must reside in the private memory of each core; Namely, L2 SRAM or in L1D SRAM (if the complete L1 D is not configured as cache).

However, the size of L2 is limited to 512K bytes for C6678, and 1024K bytes for C6670. In addition, L2 is usually partitioned into L2 cache and L2 RAM, so that the available L2 RAM size is smaller. In many cases, the amount of private data does not fit into the available L2 RAM.

This exercise presents a simple way to configure part of the external DDR as a private memory such that each core sees only its own data and structures.

## 7.2 Overview

## 7.2.1 Short Description of MPAX (Memory Protection and Extension)

<u>The C66x CorePac User's Guide</u> (Chapter 7.3) describes the MPAX unit. The MPAX registers are used to translate a logical address (32-bit address) into a physical address (36-bit address). In this example, translation registers are used to assign the different physical address of each core to the same logical address. This that the same code will load the same structures to different physical locations for each core.

The MPAX registers manipulations used to achieve the different translation will be done in the main () function. The initialization of pre-initialized global variables (for example, a declaration of global variable int x=6) is done before the start of main (). Thus, pre-initialized global variables are not allowed in the DDR private memory.

\_\_\_

**Note**—There may be way to configure the MPAX registers during boot, but this exercise does not consider it.

## 7.2.2 Coherency Discussion

For cases where L1D is configured as a cache, there is hardware coherency between L2 data and L1 D inside the C66x CorePac. The hardware coherency guarantees that a variable that is defined in L2 and is cache by L1D has the same value in both places.

This guarantee does not hold true with private memory in DDR. There is no hardware coherency between DDR and the internal memory of the core – L2 and L1D. Thus the user must maintain the coherency (using invalidate, writeback and writeback invalidate).

This example solves the cache coherency problem by disabling the cache. Disabling the cache is done using the MAR registers. The MAR registers are described in the <u>C66</u> <u>CorePac User's Guide</u> Chapter 4.4.

## 7.2.3 Usage of EDMA to Move Data to and from Private Memory

Data can be moved in and out of private memory using the EDMA. When a core reads or writes to the DDR, the data goes via the master port of the core into the MSMC through the core MPAX registers. When EDMA reads or writes data to the DDR, the data goes via the TeraNet port. The TeraNet port that connects to the DDR has 16 sets of MPAX registers that correspond to 16 privilege IDs. Each set has 8 registers. 8 more MPAX registers are connected to the shared L2 memory inside the MSMC.

The privilege ID of EDMA transform is inherited from the master who initiates the transform. So if Core 0 (with Privilege ID = 0) initiates EDMA transform, the privilege ID is 0. If Core 7 initiates the EDMA transform, the EDMA privilege ID is 7.

If the EDMA is used with the DDR private memory, then the MPAX registers of each privilege ID should be configured similarly to the core MPAX. In this example, EDMA is not used.

## 7.2.4 Platform Configuration and the Memory Map

Projects that use RTSC must define the platform. The platform defines what memories are used in the execution. In addition to L1, L2 and the MSMC memory, the external memory DDR is defined.

Usually DDR is defined as 2G bytes memory (for the 6678 EVM). In the example, you will reduce the size of DDR to 1G. You will then use the other 1G for private memories.

The default settings of the MPAX registers for all cores are as follows:

- Register 0 value is 0000001E 000000BF -> correspondent to 2G mapping of internal memory into itself
- Register 1 value is 8000001E 800000BF -> maps the 2G external memory 8000 0000 to ffff ffff into the 36 bit range 8 0000 0000 to 8 7fff ffff
- Register 2 value is 2100000b 100000ff -> Maps (again) 4K starting at address 0x21000000 to address 1 0000 0000. This is the DDR EMIF configuration registers. Note that if the same memory range is defined in multiple MPAX registers, the higher MPAX register translation is used.
- Registers 3 to 15 values are 00000000 00000080 which basically points to empty memory regions.

In this example, the first DDR are divided into two parts:

- A shared 256M DDR starting at logical address 0x8000 0000 to address 0x8fff ffff
   – physical address 8 0000 0000 to address 8 0fff ffff
- A private 16M DDR for each core. The logical address starts at 0x9000 0000 to 0x97ff ffff with physical address depends on the core number as follows:
  - Core 0 -> 8 1000 0000 to 8 10ff ffff
  - Core 1 -> 8 1100 0000 to 8 11ff ffff
  - Core 2 -> 8 1200 0000 to 8 12ff ffff
  - Core 3 -> 8 1300 0000 to 8 13ff ffff
  - Core 4 -> 8 1400 0000 to 8 14ff ffff
  - Core 5 -> 8 1500 0000 to 8 15ff ffff
  - Core 6 -> 8 1600 0000 to 8 16ff ffff
  - Core 7 -> 8 1700 0000 to 8 17ff ffff

To perform the translation, MPAX Register 4 will be used. The high 32 bits of the register has the base logical address (5 bytes) and one minus the log (base 2) of the size. Since the size of the private memory is 16M ( $2^{**}24$ ), the size is 23 or 10111b = 0x17. Thus the high 32-bit of MPAX register number 3 should be 0x90000017 (bit 5 to 11 are reserved).

The law 32-bit of MPAX 3 has the physical address (6 bytes) and 2 bytes of the permission value. In this example, full permission values are given to the user and Supervisor. Thus, the permission value is 0011 1111b or 0x3f.

The physical address part of the law 32-bit depends on the core number. Based on the values from above, the law 32-bits are

- Core 0 -> 0x810000 3f
- Core 1-> 0x811000 3f
- Core 2-> 0x812000 3f
- Core 3-> 0x813000 3f
- Core 4-> 0x814000 3f
- Core 5-> 0x815000 3f
- Core 6-> 0x816000 3f
- Core 7-> 0x817000 3f

**Note**—Bits 6 and 7 are not used and may read as non zero.

The example configures the MPAX registers in two ways, either with CSL functions or with direct register manipulations. The user can switch between the two ways (or even use both of them). A printf shows the values of the MPAX registers.

## 7.2.5 MAR Registers

Next, consider the MAR registers. For this example, the logical memory starts on 16M boundary and the size is 16M. So one MAR register controls the cache-ability of the complete private memory (MAR registers control the logical addresses, not the physical address). If the private memory was larger, multiple MAR registers will be used. And if the logical memory boundary was not aligned on 16M bytes, a more complex scheme has to be developed.

For the example case, the MAR registers disable the cache and prefetch for all addresses starting in MAR 140 (0x9000 0000) to MAR 159 (0xa000 0000).

## 7.3 Instructions

The list of processes used in this example are as follows:

- Procedure 7-1 "Run the Example Code"
- Procedure 7-2 "Connect to the Non-debuggable Devices (Esp. CCTMS\_0)"
- Procedure 7-3 "Load the Code to the 8 Cores"
- Procedure 7-4 "Configure the CSSTM\_0 Trace Control"
- Procedure 7-5 "Add and Configure the Trace Location"
- Procedure 7-6 "Start Display"
- Procedure 7-7 "Enable the Trace Point and Run"

**Note**—This exercise requires a mezzanine card with a trace emulator on the target platform.

## Procedure 7-1 Run the Example Code

## Step - Action

- 1 Load MPAX\_registerDemo from c:/temp/lab directory and look at its properties. Make sure that all the relative paths exist in your system symbols.
- 2 Choose how you want to configure the MPAX register: using CSL or direct register manipulation or both. Change the #if in two places to 0 or 1.
- **3** Rebuild the project.
- 4 Connect EVM6678 and start the debugger. Connect all the cores and load the code to all the cores.
- 5 Run all the cores. The printing will tell you when each core is done. A delay function staggers the cores.
- 6 When all cores print the done statement they are all in infinite loop. Pause all the cores.
- 7 Repeat the following for each of the cores:
  - 7a Select a single core.
  - **7b** Open a memory view window (View-> memory).
  - 7c Look at address 0x9000 0000 for all cores (This is external memory).
  - 7d Core zero will have values start at 0, 1, 2, etc.
  - 7e Core 1 will have values 0x0002 0000 and incrementing by 1.
  - 7f Core 2 will have the first value 0x0004 0000 and incrementing by 1.
  - **7g** Core 3, 4, 5, 6, and 7 will have first values 0x0006 0000, 0x0008 0000, 0x000a 0000, 0x000c 0000 and 0x000e 0000, respectively.

**Conclusion:** The same logical addresses (0x9000 0000 and on) have a different physical address for each core.

End of Procedure 7-1

## 7.3.1 Using Trace to Verify the Write Physical Address

The next section describes how to use the trace facility to verify that each core actually writes to a different physical address.

#### Procedure 7-2 Connect to the Non-debuggable Devices (Esp. CCTMS\_0)

## Step - Action

- After launching the target, right-click on the upper line in the debug window (for example, 1 evm6678Trace.ccsml) and choose show all cores.
- 2 Go to non-debuggable devices (the bottom of the window), right-click, and connect to target.
- 3 Connect all the cores as well.
- 4 The debug window should appear as shown in Figure 7-1.

#### **CCS Debug: MPAX Utilities** Figure 7-1

CCS Debug - MPAX1/MAPX_utilities.c - Code Composer Studio
File Edit View Project Tools Run Scripts Window Help
🔁 🕶 🔚 🕼 🗧 🕸 🕶 🔜 🎭 🚇 🕶 🔊 😂 🕶 🛛 🔗 🕶
🎋 Debug 🛛 💫 🦓 🐨 💷 📄 🔍 🕾 🕾 🖄 👘 🏹 🖱 🗖
⊿ 💱 evm6678Trace.ccxml [Code Composer Studio - Device Debugging]
a 🎲 Group 1
Blackhawk XDS560v2-USB System Trace Emulator_0/C66xx_0 (Suspended)
0x20B01130 (no symbols are defined for 0x20B01130)
a 👩 Blackhawk XDS560v2-USB System Trace Emulator_0/C66xx_1 (Suspended)
0x20B002A8 (no symbols are defined for 0x20B002A8)
a 👩 Blackhawk XDS560v2-USB System Trace Emulator_0/C66xx_2 (Suspended)
0x20B002A8 (no symbols are defined for 0x20B002A8)
a 👘 Blackhawk XDS560v2-USB System Trace Emulator_0/C66xx_3 (Suspended)
0x20B002A8 (no symbols are defined for 0x20B002A8)
Blackhawk XDS560v2-USB System Trace Emulator_0/C66xx_4 (Suspended)
0x20B002A8 (no symbols are defined for 0x20B002A8)
Blackhawk XDS560v2-USB System Trace Emulator_0/C66xx_5 (Suspended)
0x20B002A8 (no symbols are defined for 0x20B002A8)
Blackhawk XDS560v2-USB System Trace Emulator_0/C66xx_6 (Suspended)
0x20B002A8 (no symbols are defined for 0x20B002A8)
Blackhawk XDS560v2-USB System Trace Emulator_0/C66xx_7 (Suspended)
0x20B002A8 (no symbols are defined for 0x20B002A8)
Non Debuggable Devices
Blackhawk XDS560v2-USB System Trace Emulator_0/IcePick_D
Blackhawk XDS560v2-USB System Trace Emulator_0/CS_DAP_DebugSS
Blackhawk XDS560v2-USB System Trace Emulator_0/CSSTM_0
Blackhawk XDS560v2-USB System Trace Emulator_0/TETB_STM

**End of Procedure 7-2** 

#### Procedure 7-3 Load the Code to the 8 Cores

Step - Action

- 1 To make the results easier, change the number of elements (Line 171 in testMPAX1.c) to 10.
- 2 Build the project again and load it o the 8 cores.

End of Procedure 7-3

#### Procedure 7-4 Configure the CSSTM\_0 Trace Control

#### Step - Action

**Note**—These instructions are for CCS V5.3. For CCS V5.4 follow the instructions in Chapter 8 on page 8-1.

- 1 From the *Tools* menu (in the debug prospective), choose *Trace control*.
- 2 The Trace System Control window (see below) opens. Click on the CSSTM\_0 tab.
- **3** As shown in Figure 7-2, set *Port width* to 4 pin, place a check mark next to *Synchronize with target*, set the buffer size (512kB is sufficient for this example), and select *Circular* buffer.

Figure 7-2 Trace System Control Setting

Trace System Control					<b>X</b>
CSSTM_0 C66xx_0 C66xx_1 C66xx_2	C66xx_3 C66x	∝_4   C66∞_5	C66xx_6   C66xx_	.7	
4 pin  Port width 512 kB	<ul> <li>Buffer size</li> </ul>	C Stop-on-full			
Synchronize with target		Circular			
About	Data Viewer	Receiver	ОК	Cancel	Apply

- 4 Click on the *Receiver*... button to open the Select Receiver window as shown in Figure 7-3
  - 4a Choose where the trace data will go. Choose 560 V2 Trace.

**Note**—You can use the EB as well, but the ETB is small (32K only) and then you have to read it from the ETB.

**4b** Click *OK* on the dialogue box and then click *Apply* and wait for the programming to be done.

Figure 7-3 Trace System Control Settings: Select Receiver

CSSTM_0 C66xx_0 C66xx_ Trace Settings 4 pin  Port width Synchronize with target	512 kB 💌 Buffer size	x_4 C66xx_5 C C Stop-on-full C Circular	66xx_6   C66xx_	7]	
	Select Receiver Blackhawk XDS560v2-USB S ETB 560 V2 Trace 560 Trace Pod Pro Trace None OK	System Trace Emula	ator_0/CSSTM_0		
	About Data Viewer	Receiver	ок	Cancel	Apply

End of Procedure 7-4

#### Procedure 7-5 Add and Configure the Trace Location

Step - Action

1 From the *View* tab, open a *Breakpoint* window.

**Note**—Trace points are currently (CCS5 version 5.3.x) defined from the *Breakpoint* window.

- 2 Load the code to all the cores.
- **3** From the *Debug* menu, go back and select the CSSTM\_0 trace.
- 4 Go to the *Breakpoint* window, right-click, and select *Breakpoint*.
- **5** You can define either a breakpoint or a trace point. Select *Trace point*. A trace breakpoint will be added to the window.
- **6** To configure the trace, right-click on the trace and choose *Properties*. The *Breakpoint Properties* window is opened as shown in Figure 7-4:

#### Figure 7-4 Breakpoint Properties: Default Configuration

🖻 . 🖻 🖻 🖕 🔌 .					
N   O   N   N   N   N	🍫 ▾ 🕹 ▽ 🗖 🗖	1010 Registers (×)= Variables	● Breakpoints 🛛	6	) 🗸 😹 💥 🍪 🖂 🗙
r Studio - Device Debugging]		Identity	Name	Condition	Count
		🕅 🌧 Trace	Trace		
Properties for					
Breakpoint Properties	💣 Breakpoir	nt Properties			↓ ↓ ↓ ↓ ↓
	Properties		Values		
	⊿ Hardware	Configuration			
	⊿ Type		Trace		
	⊿ STI	M Trace Type	CP_Tracer		
		Transaction Monitor	MSMC_0		
	4	Function	Transaction Statistics	5	
		<ul> <li>Statistics Type</li> <li>Address Range Filter</li> </ul>	☐ false		
		EMU Trigger Filter	false		
		Sample Window	65535		
		Display Settings			
	⊿ Miscellane				
	Group		Default Group		
	Name		Trace		
	All settings u	nder this are handled by the	target without intruding	) on the target's execut	Edit Property
0				ОК	Cancel

- 7 To configure a Breakpoint Properties value, click on the value line to the right of the each value to access a pull-down menu of options. The trace for this example is configured as follows:
  - 7a STM Trace Type: CP\_Tracer
  - **7b** Transaction Monitor: DDR3
  - 7c Function: Transaction Logging (This will open a new dialogue box to choose what to log)
  - **7d** Transaction Master ->GEM (Select only the GEM tab, this will open the next level which GEM to follow)
  - 7e All GEMs should remain active. So select all the GEM from 0 to 7
  - 7f At this point, the window appears as shown in Figure 7-5.

#### Figure 7-5 Breakpoint Properties: Example Configuration

Breakpoint Properties	Breakpoint Properties		$\Leftrightarrow \bullet \Rightarrow \bullet$
	Properties	Values	
	Transaction Monitor	DDR3	
	⊿ Function	Transaction Logging	
	⊿ Transaction Master		
	SRIO_CPPI	☐ false	
	⊿ GEM	✓ true	
	GEM0	✓ true	
	GEM1	✓ true	
	GEM2	✓ true	
	GEM3	✓ true	
	GEM4	✓ true	
	GEM5	✓ true	
	GEM6	✓ true	
	GEM7	✓ true	
	GEM0 CFG	☐ false	
	GEM1 CFG	□ false	
	GEM2 CFG	□ false	
	GEM3 CFG	□ false	
	٠		,
	All settings under this are handled by th	e target without intruding on the target's execution	Edit Prope

- 8 Scroll down in the window to Event Filter, Transaction Type, and set Only CPU access to true.
- 9 Export Configuration should be set to true for Status and New Requests.

10 Address mask: In this example, you want to see the physical addresses. The trace can show only 10 bits of address. So you want to choose the right bits.

Of course, you can run the trace multiple times with each trace covering a different set of bits. All of the writes are into DDR3 and that the physical addresses always start at 8 1N00 0000 (where N is the core number). So the 10 bits that you are going to choose for this example are as follows:

- Bit 29: Expected to be always 0.
- Bit 28: Expected to be always 1.
- Bits 27 to 24: Expected to be the core number.
- Bits 23 to 20: Expected to be all zeros.

Click on *Export Bits*. In the pull-down menu (see Figure 7-6), choose 29:20:

## Figure 7-6 Breakpoint Properties: Address Mask Export Bits

Breakpoint Properties	💣 Breakpoint Properties		$\langle \neg \bullet \neg \neg \bullet \bullet \rangle$	
	Properties	Values		
	⊿ Transaction Type			
	CPU Data Access	✓ true		
	CPU Instruction Ad	c 🗌 false		
	DMA Access	false		
	Access Type	Any		
	▲ Export Configuration			
	Export Access Status	✓ true		
	Export New Request Export New Re			
	Export Last Burst Even			
	Address Mask	Export Bits 29:20		
	▲ Address Range Filter	Export Bits 26:17		
	Address Kange Filter b Start Location	Export Bits 20:17		
	-	Export Bits 28:19		
	Address Extension	Export Bits 29:20		
	End Location	Export Bits 30:21		
	Exclusive Address	Export Bits 31:22		
	EMU Trigger Filter	Export Bits 32:23		
	Miscellaneous	Export Bits 33:24		
	•	Export Bits 34:25 Export Bits 35:26		
	Set the 10 bit of the address associated with	Export Bits 37:28		
		Export Bits 38:29		
		Export Bits 39:30		
		Export Bits 40:31		
		Export Bits 41:32		
?		Export Bits 42:33		
		Export Bits 43:34		
		Export Bits 44:35 Export Bits 45:36		
ration: Summary - ALL 🚦 Cache		Export Bits 45:50 Export Bits 46:37		
		Export Bits 40.37		

TMS320TCl66x Keystone Multicore Workshop Lab Manual (BookID: SPRP820/A) Product Release: MCSDK 2.x

- 11 No other filtering will be used. So both the Address Range Filter and EMU Trigger Filter should be marked false. Click OK.
- 12 The Breakpoint Properties window should now be configured as shown in Figure 7-7

Figure 7-7 Breakpoint Properties: Example Configuration Complete

Breakpoint Properties	Breakpoint Properties		⇔ • ⇔ •
	Properties	Values	
	TE_SS	□ false	
	⊿ Event Filter		
	⊿ Transaction Type		
	CPU Data Access	🗹 true	
	CPU Instruction A	🗆 false	
	DMA Access	false	
	Access Type	Any	
	⊿ Export Configuration		
	Export Access Status	🗹 true	
	Export New Request E		
	Export Last Burst Even		
	Address Mask	Export Bits 29:20	
	Address Range Filter	alse false	
	EMU Trigger Filter	alse false	
	Miscellaneous		
	Group	Default Group	
	Name	Trace	
	•	III	
	All settings under this are handled by the ta	rget without intruding on the target's execution	Edit Prop

End of Procedure 7-5

## Step - Action

1 From the *Tools* menu on the *Debug* perspective, choose Trace Analyzer, then *Open Trace Connection*, and select *Blackhawk XDS560v2-USB System Trace* (or your emulator) as shown in Figure 7-8:



	Is Run Scripts Window Pin Connect	🗄 🛷 🕶 🖽 🕅 🖓 CCS Debug) 🗄
	Port Connect	💿 🗜 🌰 👻 🤣 🥙 🍈 🗆 💷 Registers 🖉 Variables 💁 Breakpoints 🖾 🛛 💣 👻 💥 🎇 🊱 🖓 🔌 🕀
	Save Memory Load Memory Fill Memory	ging)        6xx_0 (Suspended)
₽ ₩	RTOS Object View (ROV) RTOS Analyzer	b t was reached) b(∞_1 (Suspended)
ííí.	System Analyzer	<pre>&gt; voo_a (usperioed)</pre>
<u>ііі</u> 8	Graph Image Analyzer	t was reached) fox_2 (Suspended)
	Profile	•
	Trace Analyzer	Open Trace Connection In New View   Open Trace Connection In New View
	Trace Control	Open Trace File In New View Blackhawk XDS560v2-USB System Trace Emulator_0/CSSTM_0
	XDAIS Tools	Open Trace Connection

2 The Trace window will appear.

End of Procedure 7-6

Step - Action

- 1 If you have not already done so, enable the *Trace point* in the *Breakpoint* window.
- 2 Run the code on all 8 cores.
- **3** Wait for the *printf* to verify that all cores are done.
- 4 Stop Run on all of the cores.

It may take several seconds for all the data to get into the CCS. Once it is there, the results will appear as shown in Figure 7-9.

## Figure 7-9 Trace Results

e - 🦽 -							19 95 0	S Debug	CCS Edit
5 3. 5 .e 🔕 • & 🗠 7 °	° 🗆	IIII Registers (4)-	/ariables	<sup>0</sup> o Breakpoints	83		- * * *	X D D	3 🗞   🝰 🔨
e Debugging] htor_0/iC66x_0 (Suspended) ntry point was reached) htor_0/iC66x_1 (Suspended) ntry point was reached) htor_0/iC66x_2 (Suspended) ntry point was reached)		v * ···							
nor_o/coom_o (suspended)									
0		A Mamoru Reau	ene 🗖 Te	ace Dicelary - Ria	shhauk YOS	560-2-1158 Surk	em Trace Emulator 0/		1
	<u>^</u>	showing tor sam	pres (all uai						a Nate: 200.01436
				🛷 •	- 🔠 🗰	🔅 = 🚮 🕺	art Stop Resume	All & C	i kə cəl 🖽
;		Dalta Time	Time	Euroction Start	Marter ID	Marter Name	Channel Number	Modula	Data Message
		wend time			master ID	master mame	summer number	mousie	ereta mitsoage
		2		*	120	COTINICAL		COT	New Request: N
		-	-						
									New Request: N
		-							
	-								New Request: N
ss 0x9000 0000 to 0x9fff ffff i	E 8	-		0					New Request; N New Request; N
									New Request: N
00020-		-		0					New Request; N
000103		-			140	Cr meter		CPT	receiveduese, o
mary - ALL 🚦 Cache								• × 🏭 🗋	•••••
	Constraints     Constrain	Construction     C	S. S. C. (A Constraint of the second seco	A A Constraint of the second of the sec	• • • • • • • • • • • • • • • • • • •	Image: Solution of the second seco	Image: Source of the second	III Reports       III Reports	Image: Solution of the second of the seco

5 Double-click on the *Trace Display* window to enlarge it as shown in Figure 7-10. Focus on the central part of the window and notice how each core writes to a different physical address:

Figure 7-10	Trace Display Results: Core Details	

Master=GEM0; Write Access; XID=0x0; Addr=0x100 Master=GEM0; Write Access; XID=0x1; Addr=0x100 Master=GEM0; Write Access; XID=0x2; Addr=0x100 Master=GEM0; Write Access; XID=0x3; Addr=0x100 Master=GEM0; Write Access; XID=0x4; Addr=0x100 Master=GEM0; Write Access; XID=0x5; Addr=0x100 Master=GEM0; Write Access; XID=0x6; Addr=0x100 Master=GEM0; Write Access; XID=0x7; Addr=0x100 Master=GEM0; Write Access; XID=0x8; Addr=0x100 Master=GEM0; Write Access; XID=0x9; Addr=0x100 Master=GEM0; Write Access; XID=0xa; Addr=0x100 Master=GEM0; Write Access; XID=0xb; Addr=0x100 Master=GEM0; Write Access; XID=0xc; Addr=0x100 Master=GEM0; Write Access; XID=0xd; Addr=0x100 Master=GEM0; Write Access; XID=0xe; Addr=0x100 Master=GEM0; Write Access; XID=0xf; Addr=0x100 Master=GEM0; Write Access; XID=0x0; Addr=0x100 Master=GEM0; Write Access; XID=0x1; Addr=0x100 Master=GEM0; Write Access; XID=0x2; Addr=0x100 Master=GEM0; Write Access; XID=0x3; Addr=0x100 Master=GEM1; Write Access; XID=0x2; Addr=0x110 Master=GEM1; Write Access; XID=0x3; Addr=0x110 Master=GEM1; Write Access; XID=0x4; Addr=0x110 Master=GEM1; Write Access; XID=0x5; Addr=0x110 Master=GEM1; Write Access; XID=0x6; Addr=0x110 Master=GEM1; Write Access; XID=0x7; Addr=0x110 Master=GEM1; Write Access; XID=0x8; Addr=0x110 Master=GEM1; Write Access; XID=0x9; Addr=0x110 Master=GEM1; Write Access; XID=0xa; Addr=0x110 Master=GEM1; Write Access; XID=0xb; Addr=0x110 Master=GEM1; Write Access; XID=0xc; Addr=0x110 Master=GEM1; Write Access; XID=0xd; Addr=0x110

## 6 The full Trace Display should appear as shown in Figure 7-11:

## Figure 7-11 Trace Display Results: Full Frame

CCS Debug -			<u> </u>		<u> </u>			<b>L</b> .	<u>v</u>	
le Edit Vie	w Project	Tools Run	scripts Wi	ndow Help						
📬 🕶 🔛 😳		🌣 🕶 🗔 👒	🚇 🕶 🔊	1 🐑 💣 👻 📝	4 -				🥵 CCS Debu	ig) 🖽 CCS Edit
Memory Bro	wser 🗖 Te	ace Display - Bla	ckhawk XDS	S60v2-USB Syste	m Trace Emulator_0/0	SSTM_0 88				-
howing 162 Sar	mples (all dat	a)   Stopped by	user : uploa	ad complete   R	eceiver Clock: 87.5 MH	iz, Trace Dat	a Rate: 200.0145875 MHz 🔗 🗸 🛐 🏥 🌸 👻 🚮 Start Stop	Resume	ALL Q. (F. K	୦ କା 🖪 🛯
Delta Time	Time	Function Start	Marter ID	Marter Name	Channel Number	Module	Data Message		Domain	Class
cierta tittie	0	Punction start	master ap	iviaster ivarrie	channel woniber	MODUR	bata message	Crece	Comain	Cidos
3	3	0	128	CPTracer		COT	New Request: Master= GEM0: Write Access: XID=0x0: Addr=0x100		DDR	New Request (
5	8	0	128	CPTracer		CPT	New Request; Master=GEM0; Write Access; XID=0x0; Add=0x000 New Request; Master=GEM0; Write Access; XID=0x1; Addr=0x100		DDR	New Requests
5	13	0	128	CPTracer		CPT	New Request; Master-GEM0; Write Access; XID=0x2; Addr=0x100		DDR	New Request
6	19	0	128	CPTracer		CPT	New Request: Master=GEM0; Write Access; XID=0x2; Addr=0x100 New Request: Master=GEM0; Write Access; XID=0x3; Addr=0x100		DDR	New Requests
5	24	0	128	CPTracer		CPT	New Request; Master=GEM0; Write Access; XID=0x3; Addr=0x100 New Request; Master=GEM0; Write Access; XID=0x4; Addr=0x100		DDR	New Request
3	27	0	128	CPTracer		CPT	New Request: Master=GEM0; Write Access; XID=0x5; Addr=0x100 New Request: Master=GEM0; Write Access; XID=0x5; Addr=0x100		DDR	New Request
6	33	0	128	CPTracer		CPT	New Request; Master=GEM0; Write Access; XID=0x5; Addr=0x100 New Request; Master=GEM0; Write Access; XID=0x6; Addr=0x100		DDR	New Request
5	33	0	128	CPTracer		CPT	New Request: Master=GEM0; Write Access; XID=0x0; Addr=0x100 New Request; Master=GEM0; Write Access; XID=0x7; Addr=0x100		DDR	New Request
5	43	0	128	CPTracer		CPT	New Request; Master=GEM0; Write Access; XID=0x8; Addr=0x100 New Request; Master=GEM0; Write Access; XID=0x8; Addr=0x100		DDR	New Request
4	47	0	128	CPTracer		CPT			DDR	New Request
5	52	0	128	CPTracer		CPT	New Request; Master=GEM0; Write Access; XID=0x9; Addr=0x100 New Request; Master=GEM0; Write Access; XID=0xa; Addr=0x100		DDR	New Request
5	57	0	128	CPTracer		CPT				
5	62	0	128	CPTracer		CPT	New Request; Master=GEM0; Write Access; XID=0xb; Addr=0x100		DDR	New Request
4	66	0					New Request; Master=GEM0; Write Access; XID=0xc; Addr=0x100		DDR DDR	New Request
		0	128	CPTracer		CPT	New Request; Master=GEM0; Write Access; XID=0xd; Addr=0x100			New Request
-	71		128	CPTracer		CPT	New Request; Master=GEM0; Write Access; XID=0xe; Addr=0x100		DDR	New Request
5	82	0	128	CPTracer		CPT	New Request; Master=GEM0; Write Access; XID=0xf; Addr=0x100		DDR	New Request
6		0	128	CPTracer		CPT	New Request; Master= GEM0; Write Access; XID=0x0; Addr=0x100		DDR	New Request
3	85	0	128	CPTracer		CPT	New Request; Master=GEM0; Write Access; XID=0x1; Addr=0x100		DDR	New Request
5	90 96	0	128	CPTracer		CPT	New Request; Master:: GEM0; Write Access; XID:: 0x2; Addr:: 0x100		DDR	New Request
*		0	128	CPTracer		CPT	New Request; Master=GEM0; Write Access; XID=0x3; Addr=0x100		DDR	New Request
3071128	3071224		128	CPTracer		CPT	New Request; Master=GEM1; Write Access; XID=0x2; Addr=0x110		DDR	New Request
5	3071229	0	128	CPTracer		CPT	New Request; Master=GEM1; Write Access; XID=0x3; Addr=0x110		DDR	New Request:
6	3071235	0	128	CPTracer		CPT	New Request; Master=GEM1; Write Access; XID=0x4; Addr=0x110		DDR	New Request
5	3071240	0	128	CPTracer		CPT	New Request; Master=GEM1; Write Access; XID=0x5; Addr=0x110		DDR	New Request
5	3071245	0	128	CPTracer		CPT	New Request; Master=GEM1; Write Access; XID=0x6; Addr=0x110		DDR	New Request
4	3071249	0	128	CPTracer		CPT	New Request; Master=GEM1; Write Access; XID=0x7; Addr=0x110		DDR	New Request
5	3071254	0	128	CPTracer		CPT	New Request; Master=GEM1; Write Access; XID=0x8; Addr=0x110		DDR	New Request
5	3071259	0	128	CPTracer		CPT	New Request; Master:: GEM1; Write Access; XID:: 0x9; Addr:: 0x110		DDR	New Request
5	3071264	0	128	CPTracer		CPT	New Request; Master=GEM1; Write Access; XID=0xa; Addr=0x110		DDR	New Request:
4	3071268	0	128	CPTracer		CPT	New Request; Master=GEM1; Write Access; XID=0xb; Addr=0x110		DDR	New Request
5	3071273	0	128	CPTracer		CPT	New Request; Master=GEM1; Write Access; XID=0xc; Addr=0x110		DDR	New Request
5	3071278	0	128	CPTracer		CPT	New Request; Master=GEM1; Write Access; XID=0xd; Addr=0x110		DDR	New Request
6	3071284	0	128	CPTracer		CPT	New Request; Master=GEM1; Write Access; XID=0xe; Addr=0x110		DDR	New Request
3	3071287	0	128	CPTracer		CPT	New Request; Master=GEM1; Write Access; XID=0xf; Addr=0x110		DDR	New Request:
5	3071292	0	128	CPTracer		CPT	New Request; Master=GEM1; Write Access; XID=0x0; Addr=0x110		DDR	New Request
6	3071298	0	128	CPTracer		CPT	New Request; Master=GEM1; Write Access; XID=0x1; Addr=0x110		DDR	New Request
5	3071303	0	128	CPTracer		CPT	New Request; Master=GEM1; Write Access; XID=0x2; Addr=0x110		DDR	New Request
3	3071306	0	128	CPTracer		CPT	New Request; Master=GEM1; Write Access; XID=0x3; Addr=0x110		DDR	New Request:
6	3071312	0	128	CPTracer		CPT	New Request; Master=GEM1; Write Access; XID=0x4; Addr=0x110		DDR	New Request

End of Procedure 7-7

## 7.3.2 Additional Considerations

- The time unit in the tracer is in tracer ticks driven from an 87.5MHZ clock. So each time unit is 1/87.5 microsecond.
- If the load is done when the trace is open, the trace will log the loading procedure. To prevent this, load the code before configuring the CCSTM. If you want to run the code again, load it again, and re-configure the CCSTM\_0.
- Play a little more with the CCSTM configuration to see what other values can be traced.
- Play with the address bits to see how they can affect the trace values.
- If the trace file is large, you have to wait until after the system is stopped for the data to reach the console. A percentage value will indicate how much of the trace is already moved to the console (You will not see this in the previous example, since the trace is short).

# **Chapter 8**

# **STM Library and System Trace**

## 8.1 Purpose

The goal of this exercise is to demonstrate the usage of STM library to collect real-time information into system trace and present it on CCS.

## 8.2 Project Files

The following files are used in this exercise:

- Initialization.c
- StmMain.c
- System\_trace.c
- System\_trace.h
- TraceNoRTSC\_l2.cmd

## 8.3 Instructions

The list of processes used in this example are as follows:

- Procedure 8-1 "Build and Run the Project"
- Procedure 8-2 "Connect to the EVM"
- Procedure 8-3 "Load the Program and Configure the Trace"
- Procedure 8-4 "Run the Program"

**Note**—This exercise requires a mezzanine card with a trace emulator on the target platform.

Procedure 8-1	Build and Run the Project
---------------	---------------------------

## Step - Action

- 1 Open CCS.
- 2 Create new project through the CCS menu item
- **3** File  $\rightarrow$  New  $\rightarrow$  CCS Project.
- **4** Enter *stm\_example* as a *Project Name*.
- **5** Click the check box to Use default location.
- 6 Set the Family to C6000 and Variant to Generic C66xxx Device as shown in Figure 8-1:

Figure 8-1	CCS Proje	ct Properties
	New CCS Pro	ject
	CCS Project Create a new C	CCS Project.
	Device Family: Variant: Connection: Advanced s Project tem type filter text Project tem type filter text Empty En En En En En En En En En En	C:\Users\a0270985\WorkSpaces\temp Browse  C6000 <select filter="" or="" text="" type="">  Generic C66xx Device     <select filter="" or="" text="" type="">       ettings plates and examples   (Projects  (project fully initialized for the selected device.  (with main.c) (mpty Project (with main.c)) (mpty Project (with main.c)) (mpty Assembly-only Project fully P</select></select>
	?	< Back Next > Finish Cancel
		<pre>&lt; dock ivext &gt; rinish Cancel</pre>

- 7 Press *Finish* to create the new project.
- 8 In the *Project Explorer* view, right-click on the newly-created *stm\_example* project, and click on *Add Files...*
- **9** Browse to 'C:\ti\labs\code\STM,' select all the files in this directory, and click *Open*. When prompted how files should be imported into the project, leave it as default of *Copy File*.
- 10 If a file main.c was generated when you created the new project, remove the main.c file
- **11** Examine the code in 'StmMain.c' to understand the code.

#### 12 Location of the STM library

 The STM library is part of MCSDK 3 release. If you have not installed MCSDK 3, you can get the STM and all advanced debug library from the following address: https://gforge.ti.com/gf/project/ctoolslib/frs/?action=FrsReleaseBrowse&frs\_package\_id

=92

- If you do not have MCSDK release, load the library from gforge address from above and put it in directory 'C:\tiMCSDK\_3\_0\_0\_11\ctoolslib\_1\_0\_0\_3\packages\ti\STMLib'
- After installing the STM library, the directory 'C:\tiMCSDK\_3\_0\_0\_11\ctoolslib\_1\_0\_0\_3\packages\ti\STMLib' includes multiple sub-directories, as shown in Figure 8-2.

## Figure 8-2 STM Lib Directory Structure

► OSDisk (C:) ► ti ► MCSDK_3_01_12	▶ ctoolslib_1_0_2_0 ▶ packages	▶ ti ▶ STMLib ▶
brary 🔻 Share with 🔻 Burn	New folder	
Name	Date modified	Туре
🎍 doc	8/1/2013 3:40 PM	File folder
퉬 include	8/1/2013 3:40 PM	File folder
퉬 lib	8/1/2013 3:40 PM	File folder
퉬 projects	8/1/2013 3:40 PM	File folder
퉬 src	8/1/2013 3:40 PM	File folder
STMLib_5.0_Manifest.htm	2/5/2013 10:38 PM	HTML Document

- 13 Set the properties for the *Debug* configuration. Right-click on the project. Select *Properties*.
  - **13a** Choose *Build*, click on the *Environment* tab, and click the *Add*...button to add the path to add a variable with *Name* as 'STM\_ROOT' and *Value* as 'C:\tiMCSDK\_3\_0\_0\_11\ctoolslib\_1\_0\_0\_3\packages\ti\STMLib'
  - **13b** Choose *C6000 Compiler*  $\rightarrow$  *Optimization* and set/verify the following properties:
    - 'Optimization level = 0'
    - 'Optimize for code size = 0'

**13c** Choose *C6000 Compiler* → *Debug Options* and set/verify the following properties:

- 'Debugging model = Full symbolic debug'
- **13d** Choose *C6000 Compiler* → *Include Options*. Under the "Add dir to #include search path" add the following two paths:
  - "\${STM\_ROOT}/src"
  - "\${STM\_ROOT}/include"

**Note**—This ensures that any include references in the project's source files to header files located at these paths will be interpreted accurately.

- 14 In the linker tab, open file search path.
- **15** Add the following path \${STM\_ROOT}\lib to the search path and the library stm.c66xx\_elf.lib to the library list
- 16 Click the OK button to save the project properties and close the Properties window.

**17** Right-click on the project and select *Build Project*. A successful build will generate the following output on the console:

```
<Linking>
```

. .

'Finished building target: .out'

\*\*\*\* Build Finished \*\*\*\*

End of Procedure 8-1

## Procedure 8-2 Connect to the EVM

#### Step - Action

- 1 Click the *Open Perspective* (available right top corner of the CCS).
- 2 Switch to the Debug Perspective by selecting the CCS menu option Window  $\rightarrow$  Open Perspective  $\rightarrow$  CCS Debug.
- 3 Select the CCS menu option View  $\rightarrow$  Target Configurations as shown in Figure 8-3.

### Figure 8-3 CCS Target Configurations

CCS Debug - Code Composer Studio			
File Edit View Project Tools Run	Scripts Window Help		
📑 • 🗟 🕲 👘 • 🔪	B = 21   3. ○ .c = 1 = 1 = 8 @ .	· 10 10 💣 • 🛛 🛷 •	📑 🔯 C/C++ 🥵 CCS Debug) 🖥
🏇 Debug 😵 Target Configurations 😂		🕅 🗶 🤌 🖻 🖓 🗖	00= Variables 🕴 🏤 Expressions 💵 Registers
type filter text			▲ ○ ○ ○ × 当
<ul> <li>Projects</li> <li>User Defined</li> </ul>			

- **4** Use the target configuration that you created in "Preparations" on page 1-1. To create a new target configuration, follow "Create a New Target in CCS" on page 1-2.
- 5 Launch the target configuration as follows:
  - **5a** Select the target configuration .ccxml file.
  - **5b** Right click and select *Launch Selected Configuration*.
  - This will bring up the Debug window.
    - **6a** Select Core 0 (*C66x\_0*)
    - **6b** Right click and select *Connect Target*.

#### End of Procedure 8-2

Procedure 8-3	Load the Program and	Configure the Trace
---------------	----------------------	---------------------

#### Step - Action

6

**Note**—These instructions are for CCS V5.4. For CCS V5.3 follow the instructions in Chapter 7 "Using MPAX to Define Private Core Memory in DDR" on page 7-1.

- 1 Select Core 0 and load the .out file created earlier in this exercise.
  - **1a** Select the CCS menu option  $Run \rightarrow Load \rightarrow Load$  Program
  - **1b** Click Browse project...
  - **1c** Select by unwrapping the *stmSimpleExample* $\rightarrow$ *Debug* and click *OK*.
  - 1d Click OK to load the StmSimpleExample.out application to the target (Core 0).

Figure 8-4

2 As shown in Figure 8-4 select the top line in the debug window, right click, and select *Show all cores*.

CCS Debug - StmSimpleExample/StmMain		
File Edit View Project Tools Run	Scripts Window Help	
📬 🖬 🕼 🐔 🏘 🗙 🔪 🛙	• II = 24   2. O. 16 = 75   2. *	\$
称 Debug 🛛		Ş
evm6678Trace.ccxml [Code Core     Plackbauck XDS560:2 USP S	Connect Target Ctrl+Alt+C	٦
■ 00 DIACKIIAWK ADSJOUVZ-USD S	Disconnect Target Ctrl+Alt+D	
■ main(int)() at StmMain.e ■ _c_int00() at boot.c:87 0>	Enable Global Breakpoints	
Blackhawk XDS560v2-USB S	Enable Halt On Reset	
Blackhawk XDS560v2-USB S		
Blackhawk XDS560v2-USB S	Enable OS Debugging	
🔎 Blackhawk XDS560v2-USB S	Open GEL Files View	
🖉 Blackhawk XDS560v2-USB S	Hide core(s)	
Blackhawk XDS560v2-USB S	Show all cores	
Blackhawk XDS560v2-USB S	Group core(s)	"
	Sync group core(s)	
	Ungroup core(s)	
	Rename	
	Kendme	
- Alexandre - A	Remove All Terminated	
Q <sub>5</sub>	Relaunch	
💽 0x20b01130 🛛 🚺 StmMain.c 🖾 🜍	Edit evm6678Trace.ccxml	
1/**	Edit Source Lookup	
2 *	Terminate and Remove	
3*/	Terminate/Disconnect All	
5		
6 //#include /vdc/puntime/ld	Droperties	

3 Non debuggable devices will appear as shown in Figure 8-5.

## Figure 8-5 Non Debuggable Devices: Not Connected

- a 🔐 Non Debuggable Devices
  - Blackhawk XDS560v2-USB System Trace Emulator\_0/IcePick\_D
  - Blackhawk XDS560v2-USB System Trace Emulator\_0/CS\_DAP\_DebugSS (Disconnected)
  - Blackhawk XDS560v2-USB System Trace Emulator\_0/CSSTM\_0 (Disconnected)
  - Blackhawk XDS560v2-USB System Trace Emulator\_0/TETB\_STM (Disconnected)
- 4 Select the Non Debuggable device group, right click, and connect Target. You notice that all devices are connected as shown in Figure 8-6.

## Figure 8-6 Non Debuggable Devices: Connected

- a 🦃 Non Debuggable Devices
  - Blackhawk XDS560v2-USB System Trace Emulator\_0/IcePick\_D
  - Blackhawk XDS560v2-USB System Trace Emulator\_0/CS\_DAP\_DebugSS
  - Blackhawk XDS560v2-USB System Trace Emulator\_0/CSSTM\_0
  - Blackhawk XDS560v2-USB System Trace Emulator\_0/TETB\_STM

- Next, enable the Hardware Trace Analyzer as shown in Figure 8-7 5
  - Note—These operations are different in CCSV5.3. The screen shots in this example are from CCSv5.4

Figure 8-7	Enable Hard	lware Trace Analyzer
------------	-------------	----------------------

:	Memory Map	_0	2 =1 =5 📳 🎭 🚇 🕶 🔊 论 🖝 🛷 🕶
7	GEL Files		🍇 🕨 🗉 🔳 🤉 א 🖓 א 🐇
n	Debugger Options	> jing	
e	Pin Connect		-
a	Port Connect		.k_D
a	Save Memory		AP_DebugSS M_0 (Disconnected)
a a 51	Load Memory		STM
5			Suspended - SW Breakpoint)
5	Fill Memory	1 (	Disconnected : Unknown)
5 🐴	RTOS Object View (ROV)	F *	Disconnected : Unknown)
j 🛍	RTOS Analyzer	· · · · ·	Disconnected : Unknown)
5	System Analyzer	► 1 <sup>4</sup> (	Disconnected : Unknown)
1	System Analyzer	ناد ز	Discourse sheet - Unite stress)
	Hardware Trace Analyzer	•	Function Profiling - Not applicable on selected CPUs
	Hardware Trace Analyzer	) )	Function Profiling - Not applicable on selected CPUs Stall Profiling - Not applicable on selected CPUs
	Hardware Trace Analyzer Graph	)    _ /	
	Hardware Trace Analyzer Graph Image Analyzer	)    c /	Stall Profiling - Not applicable on selected CPUs
	Hardware Trace Analyzer Graph	> > >	Stall Profiling - Not applicable on selected CPUs Cache Analysis - Not applicable on selected CPUs
5 kii 5 kii 1 kii 1 kii	Hardware Trace Analyzer Graph Image Analyzer	•	Stall Profiling - Not applicable on selected CPUs Cache Analysis - Not applicable on selected CPUs Memory Throughput and Access Analysis
	Hardware Trace Analyzer Graph Image Analyzer Profile	•	Stall Profiling - Not applicable on selected CPUs Cache Analysis - Not applicable on selected CPUs Memory Throughput and Access Analysis PC Trace - Not applicable on selected CPUs
5 kii 5 kii 1 kii 1 kii	Hardware Trace Analyzer Graph Image Analyzer Profile XDAIS Tools	•	Stall Profiling - Not applicable on selected CPUs Cache Analysis - Not applicable on selected CPUs Memory Throughput and Access Analysis PC Trace - Not applicable on selected CPUs Custom PC Trace - Not applicable on selected CPUs
5 kii 5 kii 1 kii 1 kii	Hardware Trace Analyzer Graph Image Analyzer Profile XDAIS Tools	•	Stall Profiling - Not applicable on selected CPUs Cache Analysis - Not applicable on selected CPUs Memory Throughput and Access Analysis PC Trace - Not applicable on selected CPUs Custom PC Trace - Not applicable on selected CPUs Custom System Trace

/nuntime/Diago hy

## 6 The Hardware Trace Analysis Configuration window appears, as shown in Figure 8-8.

Figure 8-8 H	lardware Trace Anal	ysis Configuration
--------------	---------------------	--------------------

emory Throughput and Access Ana raphs the memory access throughput (M	1B/s) and/or		
e number of bus cycles spent waiting to	access memory.		
Vame	Instrumentation Type	Cores	Transport Type
Memory Throughput and Access Analysi	s System Trace	CSSTM_0	560 V2 Trace 👻
Data Collection Settings			Advanced Sett
	Open File Analysis Dashboard		

7 Chose *560 V2 Trace* as the transport type, and click *Start*. Shortly, several windows will open. Enlarge the *Trace Viewer CSSTM\_0* window.

End of Procedure 8-3

## Procedure 8-4 Run the Program

## Step - Action

- 1 Run the application by selecting the CCS menu option  $Run \rightarrow Resume$ .
- 2 The *Trace Viewer CSSTM\_0* window appears as shown in Figure 8-9.

## Figure 8-9 Trace Viewer CSSTM\_0

stopped	by buffer full	: upload com	piete			Analyze	•
	Time	Micro Secs	Master Name	Data Message	Data	Class	
1	0	0.0000					
2	2	0.0229	C66X_0	i value 2		Target function: PutMSG()	
3	271	3.0971	C66X_0	a1 344 a2 86		Target function: PutMSG()	
4	422	4.8229	C66X_0	a3 258		Target function: PutMSG()	
5	546	6.2400	C66X_0	End a IF changes		Target function: PutMSG()	
6	752	8.5943	C66X_0	DONE number 0 !!!!		Target function: PutMSG()	
7	922	10.5371	C66X_0	i value 4		Target function: PutMSG()	
8	1188	13.5771	C66X_0	a1 144 a2 36		Target function: PutMSG()	
9	1337	15.2800	C66X_0	a3 108		Target function: PutMSG()	
10	1461	16.6971	C66X_0	End a IF changes		Target function: PutMSG()	
11	1673	19.1200	C66X_0	DONE number1 !!!!		Target function: PutMSG()	
12	1841	21.0400	C66X_0	i value 6		Target function: PutMSG()	
13	2107	24.0800	C66X_0	al 184 a2 46		Target function: PutMSG()	
14	2255	25.7714	C66X_0	a3 138		Target function: PutMSG()	

End of Procedure 8-4

# **Revision History**

**Revision History** 

## **IMPORTANT NOTICE**

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, enhancements, improvements and other changes to its semiconductor products and services per JESD46, latest issue, and to discontinue any product or service per JESD48, latest issue. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All semiconductor products (also referred to herein as "components") are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its components to the specifications applicable at the time of sale, in accordance with the warranty in TI's terms and conditions of sale of semiconductor products. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by applicable law, testing of all parameters of each component is not necessarily performed.

TI assumes no liability for applications assistance or the design of Buyers' products. Buyers are responsible for their products and applications using TI components. To minimize the risks associated with Buyers' products and applications, Buyers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI components or services are used. Information published by TI regarding third-party products or services does not constitute a license to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of significant portions of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI components or services with statements different from or beyond the parameters stated by TI for that component or service voids all express and any implied warranties for the associated TI component or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Buyer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products, and any use of TI components in its applications, notwithstanding any applications-related information or support that may be provided by TI. Buyer represents and agrees that it has all the necessary expertise to create and implement safeguards which anticipate dangerous consequences of failures, monitor failures and their consequences, lessen the likelihood of failures that might cause harm and take appropriate remedial actions. Buyer will fully indemnify TI and its representatives against any damages arising out of the use of any TI components in safety-critical applications.

In some cases, TI components may be promoted specifically to facilitate safety-related applications. With such components, TI's goal is to help enable customers to design and create their own end-product solutions that meet applicable functional safety standards and requirements. Nonetheless, such components are subject to these terms.

No TI components are authorized for use in FDA Class III (or similar life-critical medical equipment) unless authorized officers of the parties have executed a special agreement specifically governing such use.

Only those TI components which TI has specifically designated as military grade or "enhanced plastic" are designed and intended for use in military/aerospace applications or environments. Buyer acknowledges and agrees that any military or aerospace use of TI components which have *not* been so designated is solely at the Buyer's risk, and that Buyer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI has specifically designated certain components as meeting ISO/TS16949 requirements, mainly for automotive use. In any case of use of non-designated products, TI will not be responsible for any failure to meet ISO/TS16949.

Products		Applications	
Audio	www.ti.com/audio	Automotive and Transportation	www.ti.com/automotive
Amplifiers	amplifier.ti.com	Communications and Telecom	www.ti.com/communications
Data Converters	dataconverter.ti.com	Computers and Peripherals	www.ti.com/computers
DLP® Products	www.dlp.com	Consumer Electronics	www.ti.com/consumer-apps
DSP	dsp.ti.com	Energy and Lighting	www.ti.com/energy
Clocks and Timers	www.ti.com/clocks	Industrial	www.ti.com/industrial
Interface	interface.ti.com	Medical	www.ti.com/medical
Logic	logic.ti.com	Security	www.ti.com/security
Power Mgmt	power.ti.com	Space, Avionics and Defense	www.ti.com/space-avionics-defense
Microcontrollers	microcontroller.ti.com	Video and Imaging	www.ti.com/video
RFID	www.ti-rfid.com		
OMAP Applications Processors	www.ti.com/omap	TI E2E Community	e2e.ti.com
Wireless Connectivity	www.ti.com/wirelessconne	ectivity	

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265 Copyright © 2014, Texas Instruments Incorporated