## UG0638 User Guide SmartFusion2, IGLOO2, RTG4 SmartDebug Software v12.3 and later

NOTE: PDF files are intended to be viewed on the printed page; links and cross-references in this PDF file may point to external files and generate an error when clicked. View the online help included with software to enable all linked content.







#### Microsemi Headquarters

One Enterprise, A<sup>l</sup>iso Viejo, CA 92656 USA Within the USA: +1 (800) 713-4113 Outside the USA: +1 (949) 380-6100 Fax: +1 (949) 215-4996 Email:

#### sales.support@microsemi.com www.microsemi.com

©2019 Microsemi, a wholly owned subsidiary of Microchip Technology Inc. All rights reserved. Microsemi and the Microsemi logo are registered trademarks of Microsemi Corporation. All other trademarks and service marks are the property of their respective owners. Microsemi makes no warranty, representation, or guarantee regarding the information contained herein or the suitability of its products and services for any particular purpose, nor does Microsemi assume any liability whatsoever arising out of the application or use of any product or circuit. The products sold hereunder and any other products sold by Microsemi have been subject to limited testing and should not be used in conjunction with mission-critical equipment or applications. Any performance specifications are believed to be reliable but are not verified, and Buyer must conduct and complete all performance and other testing of the products, alone and together with, or installed in, any end-products. Buyer shall not rely on any data and performance specifications or parameters provided by Microsemi. It is the Buyer's responsibility to independently determine suitability of any products and to test and verify the same. The information provided by Microsemi hereunder is provided "as is, where is" and with all faults, and the entire risk associated with such information is entirely with the Buyer. Microsemi does not grant, explicitly or implicitly, to any party any patent rights, licenses, or any other IP rights, whether with regard to such information itself or anything described by such information. Information provided in this document is proprietary to Microsemi, and Microsemi reserves the right to make any changes to the information in this document or to any products and services at any time without notice.

#### About Microsemi

Microsemi, a wholly owned subsidiary of Microchip Technology Inc. (Nasdaq: MCHP), offers a comprehensive portfolio of semiconductor and system solutions for aerospace & defense, communications, data center and industrial markets. Products include high-performance and radiation-hardened analog mixed-signal integrated circuits, FPGAs, SoCs and ASICs; power management products; timing and synchronization devices and precise time solutions, setting the world's standard for time; voice processing devices; RF solutions; discrete components; enterprise storage and communication solutions, security technologies and scalable anti-tamper products; Ethernet solutions; Power-over-Ethernet ICs and midspans; as well as custom design capabilities and services. Learn more at www.microsemi.com.

5-02-00638-6/12.19



## **Table of Contents**

Table of Contents	2
Welcome to SmartDebug	4
Introduction to SmartDebug	
Use Models	
Supported Families, Programmers, and Operating Systems	
Supported Tools	
Getting Started with SmartDebug	6
Using SmartDebug	6
Running SmartDebug in Demo Mode	6
Create Standalone SmartDebug Project	7
Import from DDC File (created from Libero)	8
SmartDebug User Interface	
Programming Connectivity and Interface	11
Selecting Devices for Debug	14
View Device Status	14
Embedded Flash Memory (NVM) Content Dialog Box (SmartFusion2 and IGLOO2 Only)	16
Debugging	19
Debug FPGA Array	19
Hierarchical View.	
Netlist View	21
Live Probes	22
Active Probes	24
Probe Grouping (Active Probes Only)	26
Memory Blocks	30
Probe Insertion (Post-Layout)	35
Event Counter	38
Frequency Monitor	41
FPGA Hardware Breakpoint Auto Instantiation	46
User Clock Frequencies	52
Pseudo Static Signal Polling	53
Debug SERDES (SmartFusion2, IGLOO2, and RTG4)	56
Debug SERDES – Loopback Test	57
Debug SERDES – PRBS Test	
Debug SERDES – PHY Reset	
SmartDebug Tcl Support	66
Frequently Asked Questions	67



	07
Embedded Flash Memory (NVM) - Failure when Programming/Verifying	
Analog System Not Working as Expected	
ADC Not Sampling the Correct Value	
How do I unlock the device security so I can debug?	
How do I export a report?	
How do I generate diagnostic reports for my target device?	
How do I monitor a static or pseudo-static signal?	
How do I force a signal to a new value?	
How do I count the transitions on a signal?	
How do I monitor or measure a clock?	
How do I perform simple PRBS and loopback tests?	
How do I read LSRAM or USRAM content?	
How do I change the content of LSRAM or USRAM?	74
How do I read the health check of the SERDES?	76
Where can I find files to compare my contents/settings?	76
What is a UFC file? What is an EFC file?	76
Is my FPGA fabric enabled?	
Is my Embedded Flash Memory (NVM) programmed?	77
How do I display Embedded Flash Memory (NVM) content in the Client partition?	77
How do I know if I have Embedded Flash Memory (NVM) corruption?	77
Why does Embedded Flash Memory (NVM) corruption happen?	77
How do I recover from Embedded Flash Memory corruption?	77
What is a JTAG IR-Capture value?	78
What does the ECC1/ECC2 error mean?	78
What happens if invalid firmware is loaded into eNVM in SmartFusion2 devices?	78
Can I compare serialization data?	78
Can I tell what security options are programmed in my device?	78
How do I interpret data in the Device Status report?	78
How do I interpret data in the Flash Memory (NVM) Status Report?	79
Device Status Report	80
IDCode	80
User Info	80
Device State	81

User Info	
Device State	
Factory Data	
Security	
-	



## **Welcome to SmartDebug**

## Introduction to SmartDebug

Design debug is a critical phase of FPGA design flow. Microsemi's SmartDebug tool complements design simulation by allowing verification and troubleshooting at the hardware level. SmartDebug provides access to non-volatile memory (eNVM), SRAM, SERDES, and probe capabilities. Microsemi SmartFusion2 System-on-chip (SoC) field programmable gate array (FPGA), IGLOO2 FPGA, and RTG4 FPGA devices have built-in probe logic that greatly enhance the ability to debug logic elements within the device. SmartDebug accesses the built-in probe points through the Active Probe and Live Probe features, which enables designers to check the state of inputs and outputs in real-time without re-layout of the design.

## **Use Models**

SmartDebug can be run in the following modes:

- · Integrated mode from the Libero Design Flow
- Standalone mode
- Demo mode

### **Integrated Mode**

When run in integrated mode from Libero, SmartDebug can access all design and programming hardware information. No extra setup step is required. In addition, the Probe Insertion feature is available in Debug FPGA Array.

To open SmartDebug in the Libero Design Flow window, expand **Debug Design** and double-click **SmartDebug Design**.

### **Standalone Mode**

SmartDebug can be installed separately in the setup containing FlashPro Express and Job Manager. This provides a lean installation that includes all the programming and debug tools to be installed in a lab environment for debug. In this mode, SmartDebug is launched outside of the Libero Design Flow. When launched in standalone mode, you must to go through SmartDebug project creation and import a Design Debug Data Container (DDC) file, exported from Libero, to access all debug features in the supported devices.

**Note**: In standalone mode, the Probe Insertion feature is not available in FPGA Array Debug, as it requires incremental routing to connect the user net to the specified I/O.

### **Demo Mode**

Demo mode allows you to experience SmartDebug features (Active Probe, Live Probe, Memory Blocks, SERDES) without connecting a board to the system running SmartDebug.

**Note:** SmartDebug demo mode is for demonstration purposes only, and does not provide the functionality of integrated mode or standalone mode.

Note: You cannot switch between demo mode and normal mode while SmartDebug is running.

### Standalone Mode Use Model Overview

In the main use model for standalone SmartDebug, the DDC file must be generated from Libero and imported into a SmartDebug project to obtain full access to the device debug features. Alternatively, SmartDebug can be used without a DDC file with a limited feature set.



## Supported Families, Programmers, and Operating Systems

Programming and Debug: SmartFusion2, IGLOO2, and RTG4
Programmers: FlashPro3, FlashPro4, FlashPro5, FlashPro6
Operating Systems: Windows 7, Windows 10, RHEL 6.x, RHEL 7.x, Cent OS 6, and Cent OS 7

## **Supported Tools**

The following table lists device family support for SmartDebug tools.

SmartDebug Support per Device Family	SmartFusion2	IGLOO2	RTG4
Live Probes	х	х	х
Active Probes	х	х	х
Memory Debug	х	х	х
Probe Insertion (available only through Libero flow)	х	х	х
View Flash Memory Content	х	х	
Debug SERDES	х	х	Х
FPGA Hardware Breakpoint (Needs FHB Auto Instantiation)	х	х	х
Event Counter (Needs FHB Auto Instantiation)	х	х	Х
Frequency Monitor (Needs FHB Auto Instantiation)	Х	Х	х

Note: "X" indicates the tool is supported.



## **Getting Started with SmartDebug**

This topic introduces the basic elements and features of SmartDebug. If you are already familiar with the user interface, proceed to the Solutions to Common Issues Using SmartDebug or Frequently Asked Questions sections.

SmartDebug enables you to use JTAG to interrogate and view embedded silicon features and device status (FlashROM, Security Settings, Embedded Flash Memory (NVM)).

See Using SmartDebug for an overview of the use flow.

You can use the debugger to:

- Get device status and view diagnostics
- Use the Embedded Flash Memory Debug GUI to read out and compare your content with your original files

### Using SmartDebug

The most common flow for SmartDebug is:

- 1. Create your design. You must have a FlashPro programmer connected to use SmartDebug.
- 2. Expand **Debug Design** and double-click **Smart Debug Design** in the Design Flow window. SmartDebug opens for your target device.
- 3. Click View Device Status to view the device status report and check for issues.
- 4. Examine individual silicon features, such as FPGA debug.

## Running SmartDebug in Demo Mode

Demo mode allows you to experience SmartDebug features (Active Probe, Live Probe, Memory Blocks, SERDES) without connecting a board to the system running SmartDebug.

**Note:** SmartDebug demo mode is for demonstration purposes only, and does not provide the functionality of integrated mode or standalone mode.

Note: You cannot switch between demo mode and normal mode while SmartDebug is running.

If programming hardware is not detected when you invoke SmartDebug, you will see the following.



SmartDebug (DEMO MODE)	- 0	×
<u>File View H</u> elp		
Device: M2S/M2GL010(T S TS) (M2GL010T)	Programmer: simulation (simulation)	~
* SMARTDEBUG IS RU	NNING IN DEMO MODE *	
ID code read from device: HARDWARE NOT C	CONNECTED	
View Device Status	Debug FPGA Array	
View Flash Memory Content	Debug SERDES	
Log		
		1
🔳 Messages 🔞 Errors 🗼 Warnings 🌐 Info		

Figure 1 · SmartDebug in Demo Mode Example

### See Also

Active Probes Live Probes Memory Blocks Debug SERDES - Loopback Test Debug SERDES - PRBS Test

## Create Standalone SmartDebug Project

A Standalone SmartDebug project can be configured in two ways:

- Import DDC files exported from Libero
- Construct Automatically

From the SmartDebug main window, click **Project** and choose **New Project**. The Create SmartDebug Project dialog box opens.



Cleate	SmartDebug Projec				×
lame:	sdebug 1				
ocation:	C:/Users				
Constru	ct JTAG chain for the	roject			
Conn	ected programmers:	S201YQST1V		Refresh	
		7: CA (12/2/10/2/14/06/06)			1
<b>@</b> I	mport DDC File: ide_	/negedgedk/2048_18_102	4_36_v/srcs/RAM_Logical_	View.ddc	
		/negedgedk/2048_18_102		View.ddc	
L	Design debug data wili	be imported with JTAG chain		View.ddc	
L		be imported with JTAG chain		View.ddc	
	Design debug data wili	be imported with JTAG chain		View.ddc	]
	Design debug data wili	be imported with JTAG chain		View.ddc	]
	Design debug data wili	be imported with JTAG chain		View.ddc	]
1	Design debug data wili	be imported with JTAG chain		View.ddc	

Figure 2 · Create SmartDebug Project Dialog Box

## Import from DDC File (created from Libero)

When you select the **Import from DDC File** option in the Create SmartDebug Project dialog box, the Design Debug Data of the target device and all hardware and JTAG chain information present in the DDC file exported in Libero are automatically inherited by the SmartDebug project. The programming file information loaded onto other Microsemi devices in the chain is also transferred to the SmartDebug project.

Debug data is imported from the DDC file (created through Export SmartDebug Data in Libero) into the debug project, and the devices are configured using data from the DDC file.

If the DDC version and software version are not compatible, project creation is not allowed, and you must run **Generate SmartDebug FPGA Array Data**. Then click **Export SmartDebug Data** to export a new DDC file and use it for project creation.

### **Construct Automatically**

When you select the **Construct Automatically** option, a debug project is created with all the devices connected in the chain for the selected programmer. This is equivalent to Construct Chain Automatically in FlashPro.

### **Configuring a Generic Device**

For Microsemi devices having the same JTAG IDCODE (i.e., multiple derivatives of the same Die—for example, M2S090T, M2S090TS, and so on), the device type must be configured for SmartDebug to enable relevant features for debug. The device can be configured by loading the programming file, by manually selecting the device using Configure Device, or by importing DDC files through Programming Connectivity and Interface. When the device is configured, all debug options are shown.

For debug projects created using Construct Automatically, you can use the following options to debug the devices:

- Load the programming file Right-click the device in Programming Connectivity and Interface.
- Import Debug Data from DDC file Right-click the device in Programming Connectivity and Interface.



The appropriate debug features of the targeted devices are enabled after the programming file or DDC file is imported.

### **Connected FlashPRO Programmers**

The drop-down lists all FlashPro programmers connected to the device. Select the programmer connected to the chain with the debug device. At least one programmer must be connected to create a standalone SmartDebug project.

Before a debugging session or after a design change, program the device through Programming Connectivity and Interface.

### See Also

Programming Connectivity and Interface View Device Status Export SmartDebug Data (from Libero)

## SmartDebug User Interface

You can start standalone SmartDebug from the Libero installation folder or from the FlashPRO installation folder. **Windows:** 

<Libero Installation folder>/Designer/bin/sdebug.exe

<FlashPRO Installation folder>/bin/sdebug.exe

### Linux:

<Libero Installation folder>/ bin/sdebug

<FlashPRO Installation folder>/bin/sdebug



SmartDebug	
Project View Tools	
ि 🚰 म 🕭	
SmartDebug Projects         New         Open         Recent Projects         C:\SmartDebug\test\test5         C:\SmartDebug\test\test4         C:\SmartDebug\test\test2         C:\SmartDebug\test\test1	
Log	ē ×
Trors 🛦 Warnings 👔 Info	

Figure 3 · Standalone SmartDebug Main Window

### Project Menu

The Project menu allows you to do the following:

- Create new SmartDebug projects (Project > New Project)
- Open existing debug projects (Project > Open Project)
- Execute SmartDebug-specific Tcl scripts (Project > Execute Script)
- Export SmartDebug-specific commands to a script file (Project > Export Script File)
- See a list of recent SmartDebug projects (Project > Recent Projects).

### Log Window

SmartDebug displays the Log window by default when it is invoked. To suppress the Log window display, click the View menu and toggle **View Log**.

The Log window has four tabs:

Messages - displays standard output messages

**Errors** – displays error messages

Warnings - displays warning messages

Info - displays general information

#### **Tools Menu**

The Tools menu includes Programming Connectivity and Interface and Programmer Settings options, which are enabled after creating or opening a SmartDebug project.



## **Programming Connectivity and Interface**

To open the Programming Connectivity and Interface dialog box, from the standalone SmartDebug Tools menu, choose **Programming Connectivity and Interface**. The Programming Connectivity and Interface dialog box displays the physical chain from TDI to TDO.

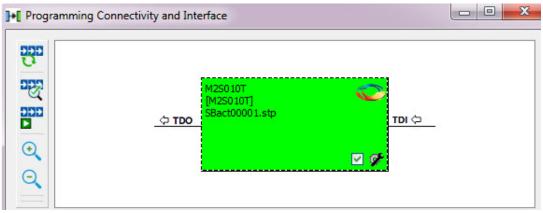


Figure 4 · Programming Connectivity and Interface Dialog Box – Project created using Import from DDC File All devices in the chain are disabled by default when a standalone SmartDebug project is created using the **Construct Automatically** option in the Create SmartDebug Project dialog box.

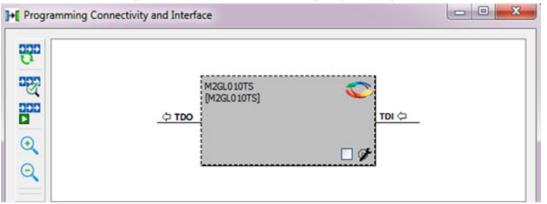


Figure 5 · Programming Connectivity and Interface window – Project created using Construct Automatically The Programming Connectivity and Interface dialog box includes the following actions:

• Construct Chain Automatically - Automatically construct the physical chain.

Running Construct Chain Automatically in the Programming Connectivity and Interface removes all existing debug/programming data included using DDC/programming files. The project is the same as a new project created using the Construct Chain Automatically option.

- Scan and Check Chain Scan the physical chain connected to the programmer and check if it matches the chain constructed in the scan chain block diagram.
- Run Programming Action Option to program the device with the selected programming procedure.

When two devices are connected in the chain, the programming actions are independent of the device. For example, if M2S090 and M2GL010 devices are connected in the chain, and the M2S090 device is to be programmed and t

he M2GL010 device is to be erased, both actions can be done at the same time using the Run Programming Action option.

- Zoom In Zoom into the scan chain block diagram.
- **Zoom Out** Zoom out of the scan chain block diagram.



### **Hover Information**

The device tooltip displays the following information if you hover your cursor over a device in the scan chain block diagram:

- **Name:** User-specified device name. This field indicates the unique name specified by the user in the Device Name field in Configure Device (right-click **Properties**).
- Device: Microsemi device name.
- Programming File: Programming file name.
- **Programming action:** The programming action selected for the device in the chain when a programming file is loaded.
- IR: Device instruction length.
- **TCK:** Maximum clock frequency in MHz to program a specific device; standalone SmartDebug uses this information to ensure that the programmer operates at a frequency lower than the slowest device in the chain.

(2) M2S050T (2)	Č	M2S050T	
	Name:	M2S050T (2)	e 🗹 🌽
	Device: File:	M2S050T	
	Programming ac	tion:	
	IR:	8	
	TCK:	1000000	

### **Device Chain Details**

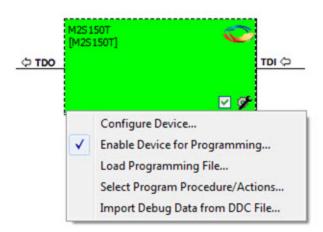
The device within the chain has the following details:

- User-specified device name
- Device name
- Programming file name
- Programming action Select **Enable Device for Programming** to enable the device for programming. Enabled devices are green, and disabled devices are grayed out.

### **Right-click Properties**

The following options are available when you right-click a device in the Programming Connectivity and Interface dialog box.





**Set as Libero Design Device** - The user needs to set Libero design device when there are multiple identical Libero design devices in the chain.

Configure Device - Ability to reconfigure the device.

- Family and Die: The device can be explicitly configured from the Family, Die drop-down.
- Device Name: Editable field for providing user-specified name for the device.

**Enable Device for Programming** - Select to enable the device for programming. Enabled devices are shown in green, and disabled devices are grayed out.

Load Programming File - Load the programming file for the selected device.

**Select Programming Procedure/Actions-** Option to select programming action/procedures for the devices connected in the chain.

- Actions: List of programming actions for your device.
- **Procedures**: Advanced option; enables you to customize the list of recommended and optional procedures for the selected action.

Import Debug Data from DDC File - Option to import debug data information from the DDC file.

**Note:** This option is supported when SmartDebug is invoked in standalone mode.

The DDC file selected for import into device must be created for a compatible device. When the DDC file is imported successfully, all current device debug data is removed and replaced with debug data from the imported DDC file.

The JTAG Chain configuration from the imported DDC file is ignored in this option.

If a programming file is already loaded into the device prior to importing debug data from the DDC file, the programming file content is replaced with the content of the DDC file (if programming file information is included in the DDC file).

### **Debug Context Save**

Debug context refers to the user selections in debug options such as Debug FPGA Array, Debug SERDES, and View Flash Memory Content. In standalone SmartDebug, the debug context of the current session is saved or reset depending on the user actions in Programming Connectivity and Interface.

The debug context of the current session is retained for the following actions in Programming Connectivity and Interface:

- Enable Device for Programming
- Select Programming Procedure/Actions
- Scan and Check Chain
- Run Programming Action

The debug context of the current session is reset for the following actions in Programming Connectivity and Interface:

Auto Construct – Clears all the existing debug data. You need to reimport the debug data from DDC file.



- Import Debug Data from DDC file
- Configure Device Renaming the device in the chain
- Configure Device Family/Die change
- Load Programming File

## Selecting Devices for Debug

Standalone SmartDebug provides an option to select the devices connected in the JTAG chain for debug. The device debug context is not saved when another debug device is selected.

s/M2GL090(T TS TV) (M2GL090TS)	Programmer: 93536 (usb93536)
S/M2GL090(T TS TV) (M2GL090TS)	
5/M2GL010(T S TS) (M2GL010TS) ode read from device: 1F8071CF	
View Device Status	Debug FPGA Array
View Flash Memory Content	Debug SERDES

### **View Device Status**

Click **View Device Status** in the standalone SmartDebug main window to display the Device Status Report. The Device Status Report is a complete summary of IDCode, device certificate, design information, programming information, digest, and device security information. Use this dialog box to save or print your information for future reference.



Device Status:	de (read from the device) (HEX):	1f8071cf		
		1.007.10.		
Devie	ce Certificate Family:	SmartFusio	n7	
	Die:	M25090		
Desid	an Information			
	Design Name:		SYS_SERDES	
	Design checksum (HEX):	53AA		
	Design Version:		0	
	Back Level: Operating voltage:	0 1.2V		
	Internal Oscillator:	50MHz		
Dige	st Information			
	Fabric Digest (HEX):		4b094bc5aa06677f5f34 0fa81a31dcb45cbb11cf	
	eNVM_0 Digest (HEX):		0b662a86aea6ab52c0db 344d1a2662418072850	
	ARM CortexM3 access to ARM CortexM3 access to Factory test mode access Power on reset delay: System Controller Suspe	eSRAM mode eSRAM mode eSRAM mode eNVM_0 read eNVM_0 writ DDR bridge DDR bridge s: Allowed. 100ms	le 0 write is protected. le 1 read is protected. le 1 write is protected. l is protected. e is protected. read is protected. write is protected.	
Prog	ramming Information			
	Cycle count:	333		
	VPP Range: Temp Range:	HIGH ( VPP HOT	>= 3.3V )	
	*Algorithm Version:	2	Flack Dec 5	
	* Programmer: * Software Version:	FlashPro v1	FlashPro 5	
	* Programming Software		1.0	
	* Programming Interface	Protocol: JTA	G	
	* Programming File Type:	STAPL		

Figure 6 · Device Status Report

### **IdCode**

IDCode read from the device under debug.

### **Device Certificate**

Device certificate displays Family and Die information if device certificate is installed on the device.

If the device certificate is not installed on the device, a message indicating that the device certificate may not have been installed is shown.



### **Design Information**

Design Information displays the following:

- Design Name
- Design Checksum
- Design Version
- Back Level (SmartFusion2 and IGLOO2 only)
- Operating Voltage (SmartFusion2 and IGLOO2 only)
- Internal Oscillator (SmartFusion2 and IGLOO2 only)

### **Digest Information**

Digest Information displays Fabric Digest, eNVM\_0 Digest and eNVM\_1 Digest (for M2S090 and M2S150 devices only) computed from the device during programming. eNVM Digest is shown when eNVM is used in the design.

### **Device Security Settings**

Note: For RTG4 devices, only Lock Bit information is displayed.

Device Security Settings indicate the following:

- Factory test mode access
- Power on reset delay
- System Controller Suspend Mode
- In addition, if custom security options are used, Device Security Settings indicate:
- User Lock segment is protected
- User Pass Key 1/2 encrypted programming is enforced for the FPGA Array
- User Pass Key 1/2 encrypted programming is enforced for the eNVM\_0 and eNVM\_1
- SmartDebug write access to Active Probe and AHB mem space
- SmartDebug read access to Active Probe, Live Probe & AHB mem space
- UJTAG access to fabric

### **Programming Information**

Programming Information displays the following:

- Cycle Count
- VPP Range
- Temp Range
- Algorithm Version
- Programmer
- Software Version
- Programming Software
- Programming Interface Protocol
- Programming File Type

# Embedded Flash Memory (NVM) Content Dialog Box (SmartFusion2 and IGLOO2 Only)

The NVM content dialog box is divided into two sections:

- View content of Flash Memory pages (as shown in the figure below)
- Check page status and identify if a page is corrupted or if the write count limit has exceeded the 10-year retention threshold



Choose the eNVM page contents to be viewed by specifying the page range (i.e., start page and the end page) and click **Read from Device** to view the values.

You must click Read from Device each time you specify a new page range to update the view.

Specify a page range if you wish to examine a specific set of pages. In the Retrieved Data View, you can enter an Address value (such as 0010) in the Go to Address field and click the corresponding button to go directly to that address. Page Status information appears to the right.

### **Contents of Page Status**

- ECC1 detected and corrected
- ECC2 detected
- Write count of the page
- If write count has exceeded the threshold
- If the page is used as ROM (first page lock)
- Overwrite protect (second page lock)
- Flash Freeze state (deep power down)

Retrieve Flash Memory Con	tent from Dev	rices																			
Select <page range=""> •</page>	Ľ.		23	Read from	Device																
Start Page: 10			(address 0)	x500)																	
End Page: 20			(11 pages,	1408 byte	s)																
atest Content Retrieved for		, 1408 b	ytes startin	g from add	ress 0x5	00												Men	Jan 18 16:4	5/55 2016	
les: Al Page Status		Go	to Address	s (hex):				Go													
tatus for Page #10:		1	and the second second	r   Address								Co	ntent								
			agenumber	1 Hourson	0		2	1 1	4	5	6	7	8	0	1 4	8	6	0	6		
			10	00500	0	1 F6	02	02	91	42	6	D1	8 D3	9 F8	A 98	20	42	P F4	80	F 71	
ion recoverable data error del	ected: False				4	F6 F8	-	-	-	-	1.0000			and the second division of	-	the second data	and in case of the local division of the loc	and property lies	of the party little	The subscription of the su	
ion recoverable data error del Nitle counter over threshold: Nitle count:	ected: False False 152		10	00500	4F C3		02	02	91	42	06	D1	03	F8	98	20	42	F4	80	71	
Von recoverable data error del Nrite counter over threshold: Nrite count: Joe as RCM: Dverwrite Protect:	ected: False False 152 Off Not set		10 10	00500	0 ∉ €	F8	02 98	02 10	91 D3	42 F8	06 98	D1 60	D3 26	F8 F0	98 80	20 65	42 C3	F4 F8	80 98	71 50	
Recoverable BCC1 error detect kon recoverable data error det Write counter over threshold: Write counts use as ROM: use as ROM: Deerwrite Protect: FlashFreeze state:	ected: False False 152 Off		10 10 10	00500	0 ∉ € ₽₽	F8 F6	02 98 00	02 10 53	91 D3 CE	42 F8 F2	06 98 00	D1 60 03	D3 26 50	F8 F0 69	98 80 45	20 65 F4	42 C3 00	F4 F8 70	80 98 58	71 50 61	
Non recoverable data error del Write counter over threshold: Write count: Use as RCM: Overwrite Protect:	ected: False False 152 Off Not set		10 10 10 10	00500 00510 00520 00530	0 4F C3 4E FF D6	F8 F6 F7	02 98 00 2E	02 10 53 ##	91 D3 CE 40	42 Fil F2 F2	06 98 00 84	D1 60 03 63	D3 25 50 C0	F8 F0 69 F2	98 80 45 00	20 65 F4 03	42 C3 00 00	F4 F8 70 22	80 98 58 D1	71 50 61 18	
Non recoverable data error del Write counter over threshold: Write count: Use as RCM: Overwrite Protect:	ected: False False 152 Off Not set		10 10 10 10	00500 00510 00520 00530 00540	0 4F C3 4E FF D6 82	F8 F6 F7 58	02 98 00 2E 40	02 10 53 FF 68	91 D3 CE 40 08	42 78 72 72 72 32	06 98 00 84 00	D1 60 03 63 18	D3 25 50 C0 35	F8 F0 69 F2 60	98 80 45 00 D1	20 65 F4 03 58	42 C3 00 00 46	F4 F8 70 22 68	80 98 58 D1 08	71 50 61 18 32	
ion recoverable data error del Nite counter over threshold: Nite count: Joe as RCM: Overwrite Protect:	ected: False False 152 Off Not set		10 10 10 10 10 10	00500 00510 00520 00530 00540 00550	0 4 € 7 6 7 6 82 00	F8 F6 F7 58 F5	02 98 00 2E 40 A0	02 10 53 FF 68 7F	91 D3 CE 40 08 0E	42 F8 F2 F2 32 60	06 98 00 84 00 F2	D1 60 03 63 18 D1	D3 25 50 C0 35 84	F8 F0 69 F2 60 F5	98 80 45 00 01 E0	20 65 F4 03 58 2F	42 C3 00 00 46 09	F4 F8 70 22 68 D3	80 98 58 D1 08 42	71 50 61 18 32 F2	
Non recoverable data error del Write counter over threshold: Write count: Use as RCM: Overwrite Protect:	ected: False False 152 Off Not set		10 10 10 10 10 10 10 10	00500 00510 00530 00540 00540 00550	0 4F C3 4E PF D6 82 00 FB	F8 F6 F7 58 F5 01	02 98 00 2E 40 A0 C4	02 10 53 FF 68 7F F2	91 D3 CE 40 08 08 02	42 F8 F2 F2 32 60 01	06 98 00 84 00 F2 01	D1 60 03 63 18 D1 24	D3 26 50 C0 35 84 0C	F8 F0 69 F2 60 F5 60	98 80 45 00 01 E0 44	20 65 F4 03 58 2F 68	42 C3 00 00 46 09 12	F4 F8 70 22 68 D3 F0	80 98 58 D1 08 42 02	71 50 61 18 32 F2 0F	

Figure 7 · Flash Memory Dialog Box for a SmartFusion2 Device (SmartDebug)

The page status gets updated when you:

- Click Page Range
- Click a particular cell in the retrieved eNVM content table
- Scroll pages from the keyboard using the Up and Down arrow keys
- Click Go to Address (hex)

The retrieved data table displays the content of the page range selection. If content cannot be read (for example, pages are read-protected, but security has been erased or access to eNVM private sectors), Read from Device reports an error.

Click View Detailed Status for a detailed report on the page range you have selected.

For example, if you want to view a report on pages 1-3, set the Start Page to 1, set the End Page to 3, and click **Read from Device**. Then click **View Detailed Status**. The figure below is an example of the data for a specific page range.



from Page 1 to Page 3, 384	bytes starting from address 0x80 as of Th	u Jan 07 14:49:29 2016 Save Arint
Page Status Summary [ Page	1 to 3]	
	of pages with ECC2 errors: 0 of pages with write count out of range: 0 or [Page 1 to 3]	
Flash Memory Page Status [P	age 1 to 3]	
FlashMemory		
FlashMemory FlashMemory	Recoverable ECC1 error detected: Non recoverable data error detected: Write counter over threshold: Write count: Use as ROM: Overwrite Protect: FlashFreeze state:	False 38 *** This value may be incorrect due to OEPB is not set. Off Not set False False False S8 **** This value may be incorrect due to OEPB is not set. Off Not set False
Help		Close

Figure 8 · Flash Memory Details Dialog Box (SmartDebug)



## Debugging

## **Debug FPGA Array**

In the Debug FPGA Array dialog box, you can view your Live Probes, Active Probes, Memory Blocks, and Insert Probes (Probe Insertion).

The Debug FPGA Array dialog box includes the following four tabs:

- Live Probes
- <u>Active Probes</u>
- Memory Blocks
- Probe Insertion

It also includes the FPGA Hardware Breakpoint (FHB) controls, consisting of the following tabs:

- "Event Counter" on page 38
- "Frequency Monitor" on page 41
- "User Clock Frequencies" on page 52

### **Hierarchical View**

The Hierarchical View lets you view the instance level hierarchy of the design programmed on the device and select the signals to add to the Live Probes, Active Probes, and Probe Insertion tabs in the Debug FPGA Array dialog box. Logical and physical Memory Blocks can also be selected.

- Instance Displays the probe points available at the instance level.
- **Primitives** Displays the lowest level of probeable points in the hierarchy for the corresponding component —i.e., leaf cells (hard macros on the device).

You can expand the hierarchy tree to see lower level logic.

Signals with the same name are grouped automatically into a bus that is presented at instance level in the instance tree.

The probe points can be added by selecting any instance or the leaf level instance in the Hierarchical View. Adding an instance adds all the probe able points available in the instance to Live Probes, Active Probes, and Probe Insertion.



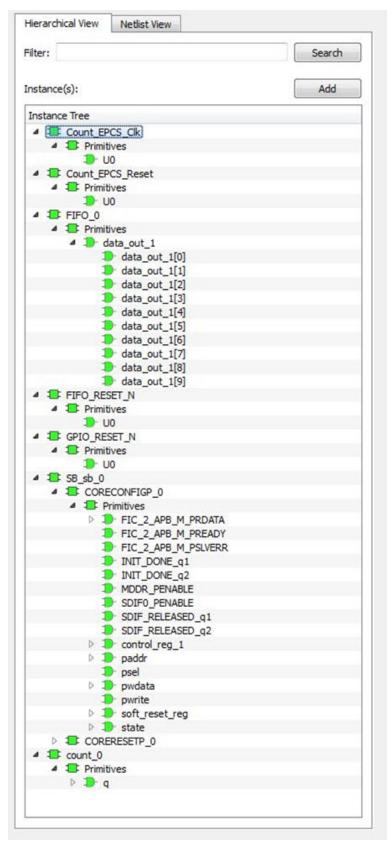


Figure 9 · Hierarchical View



### Search

In Live Probes, Active Probes, Memory Blocks, and the Probe Insertion UI, a search option is available in the Hierarchical View. You can use wildcard characters such as \* or ? in the search column for wildcard matching. Probe points of leaf level instances resulting from a search pattern can only be added to Live Probes, Active Probes, and the Probe Insertion UI. You cannot add instances of search results in the Hierarchical View.

### **Netlist View**

The Netlist View displays a flattened net view of all the probe-able points present in the design, along with the associated cell type.



Filter:	Sear	ch
Net(s):	Ad	d
Name	Туре	•
count_0_q[0]:count_0/q[0]:Q	DFF	
count_0_q[10]:count_0/q[10]:Q	DFF	
count_0_q[11]:count_0/q[11]:Q	DFF	
count_0_q[12]:count_0/q[12]:Q	DFF	
count_0_q[13]:count_0/q[13]:Q	DFF	
count_0_q[14]:count_0/q[14]:Q	DFF	
count_0_q[15]:count_0/q[15]:Q	DFF	
count_0_q[16]:count_0/q[16]:Q	DFF	
count_0_q[17]:count_0/q[17]:Q	DFF	
count_0_q[18]:count_0/q[18]:Q	DFF	
count_0_q[19]:count_0/q[19]:Q	DFF	
count_0_q[1]:count_0/q[1]:Q	DFF	
count_0_q[2]:count_0/q[2]:Q	DFF	
count_0_q[3]:count_0/q[3]:Q	DFF	
count_0_q[4]:count_0/q[4]:Q	DFF	
count_0_q[5]:count_0/q[5]:Q	DFF	
count_0_q[6]:count_0/q[6]:Q	DFF	
count_0_q[7]:count_0/q[7]:Q	DFF	
count_0_q[8]:count_0/q[8]:Q	DFF	
count_0_q[9]:count_0/q[9]:Q	DFF	4

Figure 10 · Netlist View

### Search

A search option is available in the Netlist View for Live Probes, Active Probes, and Probe Insertion. You can use wildcard characters such as \* or ? in the search column for wildcard matching.

## **Live Probes**

Live Probes is a design debug option that uses non-intrusive real time scoping of up to two probe points with no design changes.



The Live Probes tab in the Debug FPGA Array dialog box displays a table with the probe names and pin types.

Note: SmartFusion2 and IGLOO2 support two probe channels, and RTG4 supports one probe channel.

### SmartFusion2 and IGLOO2

Two probe channels (ChannelA and ChannelB) are available. When a probe name is selected, it can be assigned to either ChannelA or ChannelB.

You can assign a probe to a channel by doing either of the following:

- Right-click a probe in the table and choose Assign to Channel A or Assign to Channel B.
- Click the **Assign to Channel A** or **Assign to Channel B** button to assign the probe selected in the table to the channel. The buttons are located below the table.

When the assignment is complete, the probe name appears to the right of the button for that channel, and SmartDebug configures the ChannelA and ChannelB I/Os to monitor the desired probe points. Because there are only two channels, a maximum of two internal signals can be probed simultaneously.

Click the **Unassign Channels** button to clear the live probe names to the right of the channel buttons and discontinue the live probe function during debug.

Note: At least one channel must be set; if you want to use both probes, they must be set at the same time.

Active Probes Selection 🛛 🕏 🗙	FPGA Array debug data	
View Netlist View	Live Probes Active Probes Memory Blocks	
ilter: Search		Delete Delete All
et(s): Add		Name
Inst_CLK0_Top/Inst_CLK(	:Inst_CLK0_Top/Inst_CLK0_B2/Inst_CLK0_B3/Inst_CLK0_B4/Inst_CLK0	Assign to Channel A Assign to Channel B
Inst_CLK0_Top/Inst_CLK( Inst_CLK0_Top/Inst_CLK(		
Inst_CLK0_Top/Inst_CLK(	۲. III	•
Inst_CLK0_Top/Inst_CLK(	Assign to Channel A ->	
Inst_CLK0_Top/Inst_CLK(	Assign to Channel B ->	
Inst_CLK0_Top/Inst_CLK(	Unassign Channels	

Figure 11 · Live Probes Tab (SmartFusion2 and IGLOO2) in SmartDebug FPGA Array Dialog Box

### RTG4

One probe channel (Probe Read Data Pin) is available for RTG4 for debug. When a probe name is selected, it can be assigned to the Probe Channel (Probe Read Data Pin).

You can assign a probe to a channel by doing either of the following:

- Right-click a probe in the table and choose Assign to Probe Read Data Pin.
- Click the **Assign to Probe Read Data Pin** button to assign the probe selected in the table to the channel. The button is located below the table.

Click the **Unassign probe read data pin** button to clear the live probe name to the right of the channel button and discontinue the live probe function during debug.

The Active Probes READ/WRITE overwrites the settings of Live Probe channels (if any).



Active Probes Selection 🛛 🗗 🗙	FPGA Array debug data	
lierarchical View Netlist View	Live Probes Active Probes Memory Blocks	
Filter: Search		Delete Delete All
nstance(s): Add	Name	Туре
Instance Tree	LED_ctrl_0/pb1_reg1:LED_ctrl_0/pb1_reg1:Q	DFF
LED_ctrl_0	LED_ctrl_0/pb1_reg2:LED_ctrl_0/pb1_reg2:Q Assign to	o probe read data pin
b D counter pb1 reg1	LED_ctrl_0/pb2_reg1:LED_ctrl_0/pb2_reg1:Q	DFF
<pre>pb1_reg2 pb2_reg1</pre>	LED_ctrl_0/pb2_reg2:LED_ctrl_0/pb2_reg2:Q	DFF
▶ pb2_reg2 ▶ Tot_lft ▶ Tot_rgt		
	Assign to probe read data pin ->	
	Unassign probe read data pin	

Figure 12 · Live Probes Tab (RTG4) in SmartDebug FPGA Array Dialog Box

### **Supported Families**

SmartFusion2 IGLOO2 RTG4

### Live Probes in Demo Mode

You can assign and unassign Live Probes ChannelA and ChannelB in Demo Mode.

### **Active Probes**

Active Probes is a design debug option to read and write to one or many probe points in the design through JTAG.

In the left pane of the Active Probes tab, all available Probe Points are listed in instance level hierarchy in the Hierarchical View. All Probe Names are listed with the Name and Type (which is the physical location of the flip-flop) in the Netlist View.

Select probe points from the Hierarchical View or Netlist View, right-click and choose **Add** to add them to the Active Probes UI. You can also add the selected probe points by clicking the **Add** button. The probes list can be filtered with the Filter box.



e/Active Probes Selection	8 ×	FPGA Array debu	n data				
			10000				
Hierarchical View Netlist View		Live Probes	Active Probes	Memory Bi	ocks Probe D	nsertion	
Filter:	Search	• - •	t 🕴 📑	ave	Load	Delete	Delete All
Net(s):	Add	Name		5	pe	Read Value	Write Value
	~~~	SERDES	Debug_0_MS_RE	ADY_int:Q	DFF	1	
Name		SERDES	Debug_0.t_n_d	base:Q	DFF	1	
B_DOUT_c[7:0]		SERDES	Debug_0eset_n	_rcosc:Q	DFF	1	
Fabric_Debug_0(count_0_countA[7:0]     Fabric_Debug_0(count_0_countS[7:0]     Fabric_Debug_0(count_dbk_0(rin dbk[7:0])		Fabric_De	bug_0/count_0_	coutA[7:0]	DFF	8hE7	8ħ
		Fabric_De	bug_0/count_0_	cout8[7:0]	DFF	8764	8h
Fabric, Debug, O(xount, drk, O/sync; fra SERDES, Debug, ON, 540:SERDES, D. SERDES, Debug, ON, 540:SERDES, D. SERDES, Debug, O(X), 540:SERDES, D. SERDES, Debug, O(X), DEMO, 0. CORE SERDES, Debug, O(X), DEMO, 0. CORE	<pre>webug_0/SD_DEMO_0_CORERI webug_0/SD_DEMO_0_CORERI CONFIGP_0.INIT_DONE_q1:S CONFIGP_0.INIT_DONE_q1:S CONFIGP_0.INIT_DONE_q2:S CONFIGP_0.INIT_PRELEASED_ CONFIGP_0.SDIF_RELEASED_ CONFIGP_0.soft_reset_reg[1 CONFIGP_0.soft_reset_reg[1 CONFIGP_0.comFiGI_DONE</pre>		bu <u>g 0/c k 0/c</u> n		DFF	81194	8h
	MARTING A CHET DECET ET	R	ead Active Prober	Save A	ctive Probes' Da	Vinte Ac	tive Probes

Figure 13 · Active Probes Tab in SmartDebug FPGA Array Dialog Box

When you have selected the desired probe, points appear in the Active Probe Data chart and you can read and write multiple probes (as shown in the figure below).

You can use the following options in the Write Value column to modify the probe signal added to the UI:

- Drop-down menu with values '0' and '1' for individual probe signals
- Editable field to enter data in hex or binary for a probe group or a bus



Fabric_Debug_0/count_0_coutB[7:0]     DFF     8'hB4     8'h	•	- + + Save	Load	Delete	Delete All
SERDES_Debug_0t_n_clk_base:Q         DFF         1         0           SERDES_Debug_0eset_n_rcosc:Q         DFF         1         1           Fabric_Debug_0/count_0_coutA[7:0]         DFF         8'hE7         8'h           Fabric_Debug_0/count_0_coutB[7:0]         DFF         8'hB4         8'h	ar	me	Туре	Read Value	Write Value
SERDES_Debug_0eset_n_rcosc;Q         DFF         1         0           Fabric_Debug_0/count_0_coutA[7:0]         DFF         8'hE7         8'h           Fabric_Debug_0/count_0_coutB[7:0]         DFF         8'hB4         8'h		SERDES_Debug_0MS_READY_int:Q	DFF	1	
SERDES_Debug_0eset_n_rcosc:Q         DFF         1         1           Fabric_Debug_0/count_0_coutA[7:0]         DFF         8'hE7         8'h           Fabric_Debug_0/count_0_coutB[7:0]         DFF         8'hB4         8'h		SERDES_Debug_0t_n_dk_base:Q	DFF	1	
Fabric_Debug_0/count_0_coutA[7:0]         DFF         8'hE7         8'h           Fabric_Debug_0/count_0_coutB[7:0]         DFF         8'hB4         8'h		SERDES_Debug_0eset_n_rcosc:Q	DFF	1	
	>	Fabric_Debug_0/count_0_coutA[7:0]	DFF	8'hE7	
Fabric_Debug_0/ck_0/cin_chk[7:0] DFF 8'h94 8'h	>	Fabric_Debug_0/count_0_coutB[7:0]	DFF	8'hB4	8'h
	>	Fabric_Debug_0/ck_0/cin_chk[7:0]	DFF	8'h94	8'h

Figure 14 · Active Probes Tab - Write Value Column Options

### **Supported Families**

SmartFusion2 IGLOO2 RTG4

### **Active Probes in Demo Mode**

In demo mode, a temporary probe data file with details of current and previous values of probes added in the active probes tab is created in the designer folder. The write values of probes are updated to this file, and the GUI is updated with values from this file when you click Write Active Probes. Data is read from this file when you click Read Active Probes. If there is no existing data for a probe in the file, the read value displays all 0s. The value is updated based on your changes.

## Probe Grouping (Active Probes Only)

During the debug cycle of the design, designers often want to examine the different signals. In large designs, there can be many signals to manage. The Probe Grouping feature assists in comprehending multiple signals as a single entity. This feature is applicable to Active Probes only. Probe nets with the same name are automatically grouped in a bus when they are added to the Active Probes tab. Custom probe groups can also be created by manually selecting probe nets of a different name and adding them into the group.

The Active Probes tab provides the following options for probe points that are added from the Hierarchical View/Netlist View:

• Display bus name. An automatically generated bus name cannot be modified. Only custom bus names can be modified.



- Expand/collapse bus or probe group
- Move Up/Down the signal, bus, or probe group
- Save (Active Probes list)
- Load (already saved Active Probes list)
- Delete (applicable to a single probe point added to the Active Probes tab
- Delete All (deletes all probe points added to the Active Probes tab)
  - In addition, the context (right-click) menu provides the following operations:
    - o Create Group, Add/Move signals to Group, Remove signals from Group,
    - o Ungroup
    - o Reverse bit order, Change Radix for a bus or probe group
    - o Read, Write, or Delete the signal or bus or probe group

FPGA Array debug data Live Probes Active Probes Memory Blocks Probe Insertion ŧ + ŧ Load... Delete Delete All -Save... Name Read Value Write Value Туре SERDES\_Debug\_0...MS\_READY\_int:Q DFF 1 1 SERDES\_Debug\_0...t\_n\_dk\_base:Q DEE 0 • 1 SERDES\_Debug\_0...eset\_n\_rcosc:Q DFF 1 • Fabric\_Debug\_0/c...k\_0/cin\_chk[7:0] DFF 8'hBE 8h Fabric\_Debug\_0...0/cin\_chk[7]:Q DFF 1 • • Fabric\_Debug\_0...0/cin\_chk[6]:Q DFF 0 Fabric\_Debug\_0...0/cin\_chk[5]:Q DFF 1 \* Fabric\_Debug\_0...0/cin\_chk[4]:Q DFF 1 \* Fabric\_Debug\_0...0/cin\_chk[3]:Q DFF 1 Fabric\_Debug\_0...0/cin\_chk[2]:Q DFF • 1 Fabric\_Debug\_0...0/cin\_chk[1]:Q DFF 1 • Fabric\_Debug\_0...0/cin\_chk[0]:Q DFF 0 • group1[1:0] 2'h2 2'h Fabric\_Debug\_0...0/cin\_chk[1]:Q DFF 1 • \* Fabric\_Debug\_0...0/cin\_chk[0]:Q DFF 0 4 group2[1:0] 2'h3 2h Fabric\_Debug\_0...0/cin\_chk[5]:Q DFF \* 1 Fabric\_Debug\_0...0/cin\_chk[4]:Q DFF 1 • **Read Active Probes** Save Active Probes' Data... Write Active Probes

#### Figure 15 · Active Probes Tab

- Green entries in the "Write Value" column indicate that the operation was successful.
- Blue entries in the "Read Value" column indicate values that have changed since the last read.



### Context Menu of Probe Points Added to the Active Probes UI

When you right-click a signal or bus, you will see the following menu options: *For individual signals that are not part of a probe group or bus:* 

- Read
- Write
- Delete
- Poll
- Create Group
- Add to Group
- Move to Group

SYS_SERDES_sb_0_SDIF0_I	NIT ADR DWIDTTE-SVS SEDDES - 0/CC	RECONFIGP_0/pwrite:Q
	Read	
	Delete	
	Poll	
	Create Group	
	Add to Group	
	Move to Group	

For individual signals in a probe group:

- Read
- Delete
- Poll
- Create Group
- Add to Group
- Move to Group
- Remove from Group

	count_0_q[7]:count_0/q[7]:Q		
	count_0_q[6]:count_0/q[6]:Q	Read	
	count_0_q[5]:count_0/q[5]:Q	Delete	
	count_0_q[4]:count_0/q[4]:Q	D-11	
	SYS_SERDES_sb_0/CORECONFIGP_0/soft_reset_reg[12SERDES_	Poll	_reg[12]:
⊿ SY	S_SERDES_sb_0/CORECONFIGP_0/soft_reset_reg[14:8,6:2,0]	Create Group	
	SYS_SERDES_sb_0/CORECONFIGP_0/soft_reset_reg[14SERDES_	Add to Group	_reg[14]:
	SYS_SERDES_sb_0/CORECONFIGP_0/soft_reset_reg[13SERDES_	Move to Group	reg[13]:
	SYS_SERDES_sb_0/CORECONFIGP_0/soft_reset_reg[12SERDES_	Remove from Group	reg[12]:

For individual signals in a bus:

- Read
- Delete
- Poll
- Create Group
- Add to Group



count_0_q[19]:count_0/q[19]:Q	
count_0_q[18]:count_0/q[18]:Q	
count_0_q[17]:count_0/q[17]:Q	Read
count_0_q[16]:count_0/q[16]:Q	Delete
count_0_q[15]:count_0/q[15]:Q	Poll
count_0_q[14]:count_0/q[14]:Q	Create Group
count_0_q[13]:count_0/q[13]:Q	· · · · · · · · · · · · · · · · · · ·
count_0_q[12]:count_0/q[12]:Q	Add to Group
count_0_q[11]:count_0/q[11]:Q	
count_0_q[10]:count_0/q[10]:Q	

### For a bus:

- Delete
- Reverse Bit Order
- Change Radix to Binary
- Poll
- Create Group

count_0_q[19]:count_0/q[19]:Q	Delete
count_0_q[18]:count_0/q[18]:Q	Reverse Bit Order
count_0_q[17]:count_0/q[17]:Q	Change Radix to Binary
count_0_q[16]:count_0/q[16]:Q	Change Radix to binary
ount_0_q[15]:count_0/q[15]:Q	Poll
ount_0_q[14]:count_0/q[14]:Q	Create Group
:ount_0_q[13]:count_0/q[13]:Q	
:ount_0_q[12]:count_0/q[12]:Q	
count_0_q[11]:count_0/q[11]:Q	

For a probe group:

- Delete
- Reverse Bit Order
- Change Radix to Binary
- Poll
- Create Group
- Ungroup



1.000	count_0_q[2]:count_0/q[2]:Q	Delete			
	count_0_q[1]:count_0/q[1]:Q	Reverse Bit Order			
count_0_q[0]:count_0/q[0]:Q		Change Radix to Binary			
	count_0_q[6]:count_0/q[6]:Q				
count_0_q[5]:count_0/q[5]:Q		Poll			
A SY	S_SERDES_sb_0/CoreAHBLite_0/ma	Create Group	[15:7,		
	SYS_SERDES_sb_0/CoreAHBLite_0	Ungroup	16/ma		
	SYS_SERDES_sb_0/CoreAHBLite_0/matrix4x16/mastersite_0/matrix4x16/ma				
	SYS_SERDES_sb_0/CoreAHBLite_0/ma	atrix4x16/mastersite_0/matrix	(4x16/ma		

### Differences Between a Bus and a Probe Group

A bus is created automatically by grouping selected probe nets with the same name into a bus. A bus *cannot* be ungrouped.

A Probe Group is a custom group created by adding a group of signals in the Active Probes tab into the group. The members of a Probe Group are not associated by their names. A Probe Group *can* be ungrouped.

In addition, certain operations are also restricted to the member of a bus, whereas they are allowed in a probe group.

The following operations are not allowed in a bus:

• Move to Group: Moving a signal to a probe group

• Remove from Group: Removing a signal from a probe group

### **Memory Blocks**

The Memory Blocks tab in the Debug FPGA Array dialog box shows the hierarchical view of all memory blocks in the design. The depth and width of blocks shown in the logical view are determined by the user in SmartDesign, RTL, or IP cores using memory blocks.

#### Notes:

- RAM is not accessible to the user when SmartDebug is accessing RAM blocks.
- RAM is not accessible to the user during a read or write operation.
  - o During a single location write, the RAM block is not accessible. If multiple locations are written, the RAM block is accessed and released for each write.
  - o When each write is completed, access returns to the user, so the access time is a single write operation time.

The example figure that follows shows the hierarchical view of the Memory Blocks tab. You can view logical

blocks and physical blocks. Logical blocks are shown with an L (<sup>11</sup>), and physical blocks are shown with a P (



Terrera Placka Salatina	A Y	
temory Blocks Selection  Filter:  Memory Blocks:  Instance Tree  Set Barnen  Bernen, men, 0,0  Bernen	5 × Search Select	FPGA Array debug data     Live Probes   Active Probes Memory Blocks   Probe Insertion       User Design Memory Block:     Data Width:     Port Used:
B 28 mem_mem_0_1 B 28 mem_mem_0_2 B 28 mem_mem_0_3		
нер		Close

Figure 16 · Memory Blocks Tab - Hierarchical View

You can only select one block at a time. You can select and add blocks in the following ways:

• Right-click the name of a memory block and click Add as shown in the following figure.

Filter:     Search       Memory Blocks:     Select       Instance Tree     Oracle Tree       This Tree     Point Tree       This Tree     Point Tree       This Tree     Point Tree       This Tree     Point Tree       The Tree<	
	•

Figure 17 · Adding a Memory Block

- Click on a name in the list and then click Select.
- Select a name, drag it to the right, and drop it into the Memory Blocks tab.
- Enter a memory block name in the Filter box and click Search or press Enter. Wildcard search is supported.

Note: Only memory blocks with an L or P icon can be selected in the hierarchical view.

### **Memory Block Fields**

The following memory block fields appear in the Memory Blocks tab.

### **User Design Memory Block**

The selected block name appears on the right side. If the block selected is logical, the name from top of the block is shown.



### **Data Width**

If a block is logical, the depth and width is retrieved from each physical block, consolidated, and displayed. If the block is physical, the width is 9-bits, and the depth is 128 for uSRAM blocks and 2048 for LSRAM blocks.

### **Port Used**

This field is displayed only in the logical block view. Because configurators can have asymmetric ports, memory location can have different widths. The port shown can either be Port A or Port B. For TPSRAM, where both ports are used for reading, Port A is used. This field is hidden for physical blocks, as the values shown will be irrespective of read ports.

The following figure shows the Memory Blocks tab fields for a logical block view.

≠ mory Blocks Selection ♂ ×	EDCA Array dahun data
Filter:         Search           Filter:         Search           Instance Tree         Select           Instance Tree         Select           Image: Select Select         Select           Image: Select Select Select         Select           Image: Select Se	FPGA Array debug data       Live Probes     Active Probes       Memory Block:     mem       User Design Memory Block:     mem       Depth X Width:     2048 X 32       Port Used:     Port A
Help	Read Block Data Write Block

Figure 18 · Memory Blocks Tab Fields for Logical Block View

The following figure shows the Memory Blocks tab fields for a physical block view.

mory Blocks Selection	8 ×	FPGA Array debug data
Filter: Memory Blocks:	Search Select	Uve Probes         Active Probes         Memory Blocks         Probe Insertion           User Design Memory Block:         mem/mem_mem_0_0/INST_RAM1K20_JP           Depth X Width:         2048 X 10
Instance Tree		

Figure 19 · Memory Blocks Tab Fields for Physical Block View

### **Read Block**

Memory blocks can be read once they are selected. If the block name appears on the right-hand side, the Read Block button is enabled. Click **Read Block** to read the memory block.



### Logical Block Read

A logical block shows three fields. User Design Memory Block and Depth X Width are read only fields, and the Port Used field has options. If the design uses both ports, Port A and Port B are shown under options. If only one port is used, only that port is shown.

ary Blocks Selection 🗗 🛪	1																	
a y block selectori	FPGA Arr	ay debug da	ita															
Iter: Search	Live P	obes   A	tive Probes	Мето	ry Blocks	Probe Ins	ertion											
	Line 7	and the later of the second	and a second	mem														
emory Blocks: Select		esign Memo X Width:	A DIDOC	2048 X 32														
	PortU			Port A	•													
Instance Tree	Porto	yeu.		India	-													
B S mem B S mem_mem_0_0		0	1	2	3	4	5	6	7	8	9	A	8	C	D	E	F	-
B Primitives     INST RAMIK20 IP	0000	AFREADS	RECERCUT	DESDERET	F4823A29	7686887	TEFESSA	CB35E450	55464785	27180084	20750603	89681940	D30E72C7	30007206	62312005	20486950	A116204	-
B S mem mem 0 1 B S Primtives				100000000		1		100000000		CONCERNS OF	10070725	10.1 10.1 10.2 10.1	Part of the local division of	Contractor		Conception of the	-	
INST_RAM1K20_IP	0010	AA1E348	7 10130DEC	803128E	FOB4EBBC	89CF8849	9AEASA60	59E2F88A	33AD6DCF	F 49988804	E4A8F39E	240382FE	F76CF380	29EE03C8	883F6516	68D0CEEE	77E2586	0
B mem_mem_0_2     B Primitives	0020	012A7775	8281438F	DESSESE	ED013E69	C541FBD6	AF 38ABCO	DFEBC883	44F2A410	36716108	D986E8DC	ADECE522	D95DED 10	50F707C8	78883368	2372881A	62CD-401	E
INST_RAM1K20_IP													-					
E S men_men_0_3 E S Primitives	0030	D6F6EC9	OEDAC 15	C-E 126.2	0AE257A4	38184604	91906028	D689P281	01-4CED 76	08CC13EB	SABCBOOK	P02E8870	47860346	BAA4A57E	/PA/PB9E	322 68 70	0843715	0
INST_RAM1K20_IP	0040	3E130060	22169358	D3A1CEA	0 88 1D 28D8	AE4FA47C	DE703809	SOAS28F1	01811160	E5637FB3	A2F5C501	C8756D70	D846A170	1E3FBC9E	8E994EDF	2EA72120	61F2E41	7 -
												. 1						
							Read	Block	Save Block	Data	Write Blo	OK						
																		_

### Figure 20 · Logical Block Read

The data shown is in Hexadecimal format. In the example figure above, data width is 32. Because each hexadecimal character has 4 bits of information, you can see 8 characters corresponding to 32 bits. Each row has 16 locations (shown in the column headers) which are numbered in hexadecimal from 0 to F.

**Note:** For all logical blocks that cannot be inferred from physical blocks, the corresponding icon does not contain a letter.

### Physical Block Read

When a Physical block is selected, only the User Design Memory Block and Depth X Width fields are shown.

emory Blocks Selection	5×																	
	FPGA Arra	y debug da	ta															
Filter: Search	Live Pro	bes   A	tive Probes	Memor	ry Blocks	Probe Insi	ertion											
Memory Blocks: Select	User De Depth X		ry Block:	mem/mem, 2048 X 10		INST_RAM1	K20_JP											
Instance Tree		0	1	2	3	4	5	6	7	8	9	A	B	c	D	E	F	•
B & men B & men men 0 0	0000	OAD	000	3F1	259	177	11A	2A0	315	17A	183	14C	387	386	185	360	244	-
E S Primitives DINST_RAM1K20_IP	0010	107	300	3CA	370	299	006	-	39F	214	12E	1FE	370	398	036	3CE	000	
B men men 0 1     B Primbves						279		30A				1.000				1.000		
INST_RAMIK20_IP     B mem_mem_0_2	0020	0E9	31F	308	209	146	IAD	163	020	3AB	3AC	252	03C	398	2CE	22A	23E	
E Primitives	0030	323	28A	048	354	18F	048	171	0E6	306	346	2E0	09E	2EE	12E	0EC	320	
INST_RAMIK20_IP     B    mem_mem_0_3     B	0040	0DC	088	340	388	OFC	009	1F1	000	363	011	OFC	2ED	13E	18F	05D	027	
- INST_RAMIK20_IP	0050	159	158	15C	135	104	066	211	OOF	2AC	OCA	0F3	0A7	194	0E6	2FA	358	
							Read	Block	Save Block	Data	Write Blo	dk						
												_						_
							Read	Block	Save Block	Data	Write Blo	dk 📘						

Figure 21 · Physical Block Read



### Write Block

### **Logical Block Write**

A memory block write can be done on each location individually. A logical block shows each location of width. The written format is hexadecimal numbers from 0 to F. Width is shown in bits, and values are shown in hexadecimal format. If an entered value exceeds the maximum value, SmartDebug displays a pop-up message showing the range of allowed values.

emory Blocks Selection & X	-	ay debug da																
	1000000						- 21											
Filter: Search	Live Pr	obes A	tive Probes	Memor	ry Blocks	Probe Ins	ertion											
Memory Blocks: Select		esign Memo X Width:	ry Block:	mem 2048 X 32														
Instance Tree	Port U	ved:		Port A	•													
S men     B men_men_0_0     B    S Primitives		0	- 1	2	3	4	5	6	7	8	9	A	В	С	D	E	F	•
B Printives     INST_RAM1K20_IP     B men_men_0_1	0000	AEB66D St	88258000	DF30E8F1	F4823A29	76686887	7EEF558A	CB35E450	55A6A785	2718008A	297E96D3	896819AC	D30E77C7	30007206	62317005	39A868F0	A1162D5	A
Primitives     INST RAMIK20 IP	0010	AA 1E 348	FFFFFFFF	803128EA	FOB4EBBC	B9CF8849	9AEASA66	59E2F88A	33AD6DCF	49988804	E4A8F39E	240382FE	F76CF38D	29EE03CB	883F6516	68D0CEEE	77E2586	5
B S mem_mem_0_2 B S Primitives	0020	012A7779	8281438F	DBSSFSEB	ED013E69	C541FBD6	AF38ABOC	DFEBC883	4#2A410	367161D8	D986E8DC	ADECE522	D9506010	50F707C8	7888336E	237288 1A	62CD401	E
INST_RAMIK20_IP     E mem_mem_0_3     R    Rrimitives	0030	D6F6EC9	0EDAC15	C4E 1E628	0AE257A4	3818F60F	91FD6028	D689F281	014CED76	08CC13E8	3ABCEDD6	F02EB870	4F86D34€	BAA44678	7F47FB9E	322FE870	D843719	D
<ul> <li>Primtives</li> <li>INST_RAM1K20_IP</li> </ul>	0040	3E 130060	22169358	DIAICEA	88 10 2806	AE4FA470	DE 703809	SOAS2BF1	0181116D	E5637FB3	A2F5C501	C8756D70	D846A170	1E3FBC9E	8E994EDF	2EA7212	61F2E41	
							Read	Block	Save Block	Data	Write Blo	dk 🔤						
																		_
		1					Read	Block	Save Block	Data	Write Blo	dk						

Figure 22 · Logical Block Write

### **Physical Block Write**

Physical blocks have a fixed width of 9 bits. The maximum value that can be written in hexadecimal format is 1FF. If an entered value exceeds the limit, SmartDebug displays a popup message showing the range of values that can be entered.

mory Blocks Selection	e x	FPGA Arra		1															
			1.1.1.1.1																
Filter:	Search	Live Pro	bes A	tive Probes	Memor	y Blocks	Probe Ins	ertion											
Memory Blocks:	Select	User Design Memory Blok: mem/mem_ge_0_0/b/ST_RAM 3/20_1P Depen X Width: 2048 X 10																	
Instance Tree			0	1	2	3	4	5	6	1	8	9	A	B	c	D	E	F	-
	0	0000	0AD	000	₹1	259	177	11A	2A0	315	17A	183	14C	387	38E	185	3ED	ZAA	
INST_RAMIX20_IP     S mem.mem.0_1     S Primitives     INST_RAMIX20_IP     INST_RAMIX20_IP		0010	107	300	3CA	37C	299	006	30A	39F	214	12E	IFE	370	398	036	3CE	000	
		0020	0E9	31F	3FF	209	146	1AD	163	020	348	3AC	252	03C	398	2CE	22A	23E	
B S mem_mem_0 B S Primitives	_2 _RAM1K20_IP	0030	323	28A	048	354	18F	048	171	0E6	308	3A6	2E0	09E	ZEE	12E	OEC	320	
B # mem_mem_0 B # Primitives	3	0040	000	088	340	388	OFC	009	1F1	000	363	011	OFC	2ED	13E	15F	05D	027	
- DIST	RAM1K20_IP	0050	159	158	15C	135	104	066	211	OOF	2AC	OCA	OF3	0A7	194	OE6	2FA	358	•
								Read	Block	Save Block	Data	Write Blod	k						
	_											1	- 524						_

Figure 23 · Physical Block Write

### **Unsupported Memory Blocks**

If RTL is used to configure memory blocks, it is recommended that you follow RAM block inference guidelines provided by Microsemi. See <u>Inferring Microsemi SmartFusion2 RAM Blocks</u> for more information.



SmartDebug may or may not be able to support logical view for memory blocks that are inferred using RTL coding not specified in the above document.

### **Memory Blocks in Demo Mode**

A temporary memory data file is created in the designer folder for each type of RAM selected. All memory data of all instances of USRAM, LSRAM, and other RAM types is written to their respective data files. The default value of all memory locations is shown as 0s, and is updated based on your changes.

Both physical block view and logical block view are supported.

## Probe Insertion (Post-Layout)

### Introduction

Probe insertion is a post-layout debug process that enables internal nets in the FPGA design to be routed to unused I/Os. Nets are selected and assigned to probes using the Probe Insertion window in SmartDebug. The rerouted design can then be programmed into the FPGA, where an external logic analyzer or oscilloscope can be used to view the activity of the probed signal.

Note: This feature is not available in standalone mode because of the need to run incremental routing.

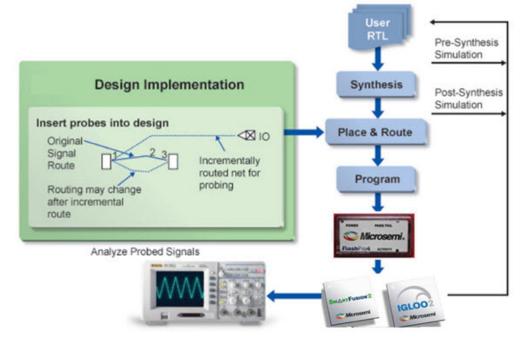


Figure 24 · Probe Insertion in the Design Process

The Probe Insertion debug feature is complementary to Live Probes and Active Probes. Live Probes and Active Probes use a special dedicated probe circuitry.

### **Probe Insertion**

- Double-click SmartDebug Design in the Design Flow window to open the SmartDebug main window. Note: FlashPro Programmer must be connected for SmartDebug.
- 2. Select Debug FPGA Array and then select the Probe Insertion tab.



e Insertion Data Selecti	on	8 ×	FPGA Array debu	ug data					
Hierarchical View Ne	tlist View		Live Probes	Active Probes	Memory Blocks	Probe Insertion			
Filter:		Search						Delete	Delete All
nstance(s):	ſ	Add	Net		Driver		Package Pin	Po	ort Name
Instance Tree	-		AND2_0_Y	AND2_0/	J0:Y		Unassigned	Probe_Insert	D
Image: Second state         Image: Second state           Image: Second state         Image: Second state <td></td> <td></td> <td>D_¢</td> <td>UJTAG_0</td> <td>INST_UJTAG_SYSRE:</td> <td>SET_FF_IP:UDRUPD</td> <td>Unassigned</td> <td>Probe_Insert</td> <td>1</td>			D_¢	UJTAG_0	INST_UJTAG_SYSRE:	SET_FF_IP:UDRUPD	Unassigned	Probe_Insert	1
<ul> <li>Stop</li> <li>UJTAG_0</li> <li>USer_CLK</li> <li>Count_0</li> </ul>						Inse	ert probe(s) and pr	ogram the device	Run

Figure 25 · Probe Insertion Tab

In the left pane of the Probe Insertion tab, all available Probe Points are listed in instance level hierarchy in the Hierarchical View. All Probe Names are shown with the Name and Type in the Netlist View.

 Select probe points from the Hierarchical View or Netlist View, right-click and choose Add to add them to the Active Probes UI. You can also add the selected probe points by clicking the Add button. The probes list can be filtered with the Filter box.

Each entry has a Net and Driver name which identifies that probe point.

The selected net(s) appear in the Probes table in the Probe Insertion tab, as shown in the figure below. SmartDebug automatically generates the Port Name for the probe. You can change the Port Name from the default if desired.

4. Assign a package pin to the probe using the drop-down list in the Package Pin column. You can assign the probe to any unused package pin (spare I/O).



be Insertion Data Selection	ē ×	FPGA Array debu	ug data				
Hierarchical View Netlist View		Live Probes	Active Probes	Memory Blocks	Probe Insertion		
Filter:	Search					Delete	Delete All
Instance(s):	Add	Net		Driver	Package Pin	Port	Name
Instance Tree		q_c[0]	count_0/q[	[0]:Q	Н5 -	Probe_Insert0	
<ul> <li>Primitives</li> <li>AND2_0</li> </ul>		q_c[1]	count_0/q[	[1]:Q	H6	Test2	
		q_c[3]	count_0/q[	[3]:Q	36 •	Probe_Insert2	
<ul> <li>Stop</li> <li>UJTAG_0</li> <li>UUTAG_0</li> <li>UUTAG_0</li> <li>Ster_CLK</li> <li>Count_0</li> <li>Primitives</li> <li>QRNO</li> <li>QRNO</li> <li>Q_RNO</li> <li>Q_roy</li> <li>Q_s</li> </ul>					Insert probe(s) a	and program the devic	ve Run

Figure 26 · Debug FPGA Array > Probe Insertion > Add Probe

5. Click Run.

This triggers Place and Route in incremental mode, and the selected probe nets are routed to the selected package pin. After incremental Place and Route, Libero automatically reprograms the device with the added probes.

The log window shows the status of the Probe Insertion run.

#### **Probe Deletion**

To delete a probe, select the probe and click **Delete**. To delete all probes, click **Delete All**.

**Note**: Deleting probes from the probes list without clicking **Run** does not automatically remove the probes from the design.

#### **Reverting to the Original Design**

To revert to the original design after you have finished debugging:

- 1. In SmartDebug, click **Delete All** to delete all probes.
- 2. Click Run.
- 3. Wait until the action has completed by monitoring the activity indicator (spinning blue circle). Action is completed when the activity indicator disappears.
- 4. Close SmartDebug.



# **Event Counter**

The Event Counter counts the signals that are assigned to Channel A through the Live Probe feature. This feature can track events from the MSS or the board. When the Event Counter is activated, and a signal is assigned to Channel A, the counter starts counting the rising edge transitions. The counter must be stopped to get the final signal transition count. During the count, you cannot assign another signal to Channel A/Channel B or go to any other tab on the window.

Active Probes Selection	₫ × FPGA Array debug data
Hierarchical View Netlist View	Live Probes Active Probes Memory Blocks Probe Insertion
Filter:	rdh Delete All
instance(s):	id Name Type
Primitives	A_DOUT_0_c[8]:URAM_3\/sd_URAM_3_URAM_R0C4/INST_RAM64x18_IP:A_DOUT[0] RAM64x18
▷ == URAM_0\ ▷ == URAM_1\	A_DOUT_0_c[7]:URAM_3\vd_URAM_3_URAM_R0C3/INST_RAM64x18_JP:A_DOUT[1] RAM64x18
> == URAM_2\ > == URAM_3\	A_DOUT_0_c[6]:URAM_3\/sd_URAM_3_URAM_R0C3/INST_RAM64x18_IP:A_DOUT[0] RAM64x18
⇒ == count 6 0\	A_DOUT_0_c[5]:URAM_3\sd_URAM_3_URAM_R0C2/INST_RAM64x18_IP:A_DOUT[1] RAM64x18
vent Counter/Frequency Monitor	A_DOUT_0_c[4];URAM_3\/sd_URAM_3_URAM_R0C2/JINST_RAM64x18_JP;A_DOUT[0] RAM64x18
	A_DOUT_0_c[3]:URAM_3\/sd_URAM_3_URAM_ROC1/INST_RAM64x18_IP:A_DOUT[1] RAM64x18
	t A_DOUT_0_c[2]:URAM_3\/sd_URAM_3_URAM_ROC1/INST_RAM64x18_IP:A_DOUT[0] RAM64x18
dge Selected: Rising me elapsed (s):	A_DOUT_0_c[1]:URAM_3Vsd_URAM_3_URAM_R0C0/INST_RAM64x18_IP:A_DOUT[1] RAM64x18
otal Events: 598355545	A_DOUT_0_c[0]:URAM_3\/sd_URAM_3_URAM_R0C0/INST_RAM64x18_IP:A_DOUT[0] RAM64x18
Signal : A_DOUT_0_c(8):URAM_3\/sd_URAM_3_URAM	<
	Assign to Channel A -> A_DOUT_0_c[8]:URAM_3\/sd_URAM_3_URAM_R0C4/INST_RAM64x18_
	Assign to Channel B ->
	Unassign Channels
Event Counter Frequency Monitor User Clock Freque	

Figure 27 · Event Counter Tab/UI

#### **Activating the Event Counter**

You can activate the Event Counter in either of the following two ways:

• Click Activate Event Counter and then assign a signal to Live Probe Channel A.



<b>v</b>											
e/Active Probes Selection	e x	FP	GA Array debu	ig data							
Herarchical View Netlist View			Live Probes	Active Pro	bes	Memory Blo	cks Prob	e Insertion			
Filter:	Search								Delete	Delete	Al
Instance(s):	Add				aras			Name			*
<pre>&gt; \$ sd1_0\ &gt; \$ sd1_1\</pre>	* E		sd1_2\/count	er_top_0Vg	_cnt[19	9]_counterV	reg_counter	[9]_net_1:	sd1_2\/counter	_top_0Vg_cnt	t[15
4 1 sd1_2	(i)		sd1_2\/count	er_top_0Vg	_cnt[19	(8]_counterV	reg_counter	[9]_net_1:	sd1_2\/counter	_top_0Vg_cnt	t[15
counter_top_0\ counter_top_1\			sd1_2\/country	er_top_0Vg	_ont[19	7]_counterV	reg_counter	[9]_net_1::	sd1_2\/counter	_top_0Vg_cnt	£[15
Image: Second			sd1_2\/countr	er_top_0Vg	_ont[19	6]_counterV	reg_counter	[9]_net_1::	sd1_2Vcounter	_top_0Vg_cnt	4[15
Event Counter/Frequency Monitor			sd1_2\/count	er_top_0Vg	_ont[19	(5]_counterV	reg_counter	[9]_net_1::	sd1_2\/counter	_top_0Vg_cnt	đ[15
		5	sd1_2\/counti	er_top_0Vg	_	Assign to C	hannel A	_net_to	sd1_2\/counter	_top_0Vg_cnt	15
Activate Event Counter	Reset	1	sd1_2\/count	er_top_0Vg		Assign to C		_net_la	sd1_2\/counter	_top_0Vg_cnt	t[15
Edge Selected: Rising	Stop		sd1_2\/count	er_top_0Vg	_ont[19	2]_counterV	reg_counter	[9]_net_1:	sd1_2\/counter	_top_0Vg_ont	t[15
Total Events: 0			sd1_2\/count	er_top_0Vg	_ont[19	1]_counterV	reg_counter	[9]_net_1::	sd1_2\/counter	_top_0Vg_cnf	t[15
Signal : sd1_2\/counter_top_0\/g_cnt[194]_coun	nter Vr		sd1_2\/count	er_top_0Vg	_ont[19	0]_counterV	reg_counter	[9]_net_1::	sd1_2Vcounter	_top_0Vg_cnt	t[15
			sd1_2\/count	er_top_0Vg	_ont[18	9]_counterV	reg_counter	[9]_net_1a	sd1_2\/counter	_top_0Vg_cnt	t[18
			sd1_2\/count	er_top_0Vg	_ont[18	8]_counterV	reg_counter	[9]_net_1::	sd1_2\/counter	_top_0Vg_on	118 _
			Assign to Ch	annel A	-> sc	1_2Vcounte	_top_0Vp_c	nt[194]_co	unter Vreg_co	unter[9]_net_	1:sd1_;
			Assign to Ch	annel B	->						
			Unassign Ch	anneis							
Event Counter Frequency Monitor User	Clock Frequencies										

Figure 28 · Activating the Event Counter

• Assign a signal to Probe Channel A and then click Activate Event Counter.

Active Probes Selection & X	FPGA Array debug data
Herarchical View Netlist View	Live Probes Active Probes Memory Blocks Probe Insertion
Filter: Search	Delete Al
Instance(s): Add	Name
> \$ sd1_0\       > \$ sd1_2\       > \$ counter_top_0\       > \$ counter_top_1\       > \$ counter_top_2\	sd1_2\/counter_top_0\/g_cnt[199]_counter\/teg_counter[9]_net_1:sd1_2\/counter_top_0\/g_cnt[15 sd1_2\/counter_top_0\/g_cnt[197]_counter\/teg_counter[9]_net_1:sd1_2\/counter_top_0\/g_cnt[15 sd1_2\/counter_top_0\/g_cnt[197]_counter\/teg_counter[9]_net_1:sd1_2\/counter_top_0\/g_cnt[15 sd1_2\/counter_top_0\/g_cnt[196]_counter\/teg_counter[9]_net_1:sd1_2\/counter_top_0\/g_cnt[15 sd1_2\/counter_top_0\/g_cnt[196]_counter\/teg_counter[9]_net_1:sd1_2\/counter_top_0\/g_cnt[15 sd1_2\/counter_top_0\/g_cnt[16 Assign to Channel A sd1_2\/counter_top_0\/g_cnt[15
Activate Event Counter   Edge Selected: Rising  Time elapsed (s):  Total Events:  Signal : sd1_2V(counter_top_0V)g_cnt[194]_counterV/r	<pre>edi_2V(counter_top_0Vg_cnt[193]_counter\/reg_counter[9]_net_1:edi_2V(counter_top_0Vg_cnt[15 edi_2V(counter_top_0Vg_cnt[192]_counter\/reg_counter[9]_net_1:edi_2V(counter_top_0Vg_cnt[15 edi_2V(counter_top_0Vg_cnt[191]_counter\/reg_counter[9]_net_1:edi_2V(counter_top_0Vg_cnt[15 edi_2V(counter_top_0Vg_cnt[190]_counter\/reg_counter[9]_net_1:edi_2V(counter_top_0Vg_cnt[15 edi_2V(counter_top_0Vg_cnt[189]_counter\/reg_counter[9]_net_1:edi_2V(counter_top_0Vg_cnt[18 edi_2V(counter_top_0Vg_cnt[189]_counter\/reg_counter[9]_net_1:edi_2V(counter_top_0Vg_cnt[18 edi_2V(counter_top_0Vg_cnt[188]_counter\/reg_counter[9]_net_1:edi_2V(counter_top_0Vg_cnt[18 edi_2V(counter_top_0Vg_cnt[188]_counter\/reg_counter[9]_net_1:edi_2V(counter_top_0Vg_cnt[18 edi_2V(counter_top_0Vg_cnt[188]_counter\/reg_counter[9]_net_1:edi_2V(counter_top_0Vg_cnt[18 edi_2V(counter_top_0Vg_cnt[188]_counter\/reg_counter[9]_net_1:edi_2V(counter_top_0Vg_cnt[18 edi_2V(counter_top_0Vg_cnt[188]_counter\/reg_counter[9]_net_1:edi_2V(counter_top_0Vg_cnt[18] edi_2V(counter_top_0Vg_cnt[188]_counterVeg_counter[9]_net_1:edi_2V(counter_top_0Vg_cnt[18]); edi_2V(counter_top_0Vg_cnt[188]_counterVeg_counter[9]_net_1:edi_2V(counter_top_0Vg_cnt[18]); edi_2V(counter_top_0Vg_cnt[18]); edi_2V(counter_top_0Vg_cnt[18]); edi_2V(counter_top_0Vg_cnt[18]); edi_2V(counter_top_0Vg_cnt[18]); edi_2V(counter_top_0Vg_cnt[18]); edi_2V(counter_top_0Vg_cnt[18]); edi_2V(counter_top_0Vg_cnt[18]); edi_2V(counter_top_0Vg_cnt[18]); edi_2V(counter_top_0Vg_cnt[18]); edi_2V(counter_top_0Vg_cnt[18]); edi_2V(counter_top_0Vg_cnt[18]); edi_2V(counter_top_0Vg_cnt[18]); edi_2V(counter_top_0Vg_cnt[18]); edi_2V(counter_top_0Vg_cnt[18]); edi_2V(counter_top_0Vg_cnt[18]); edi_2V(counter_top_0Vg_cnt[18]); edi_2V(counter_top_0Vg_cnt[18]); edi_2V(counter_top_0Vg_cnt[18]); edi_2V(counter_top_0Vg_cnt[18]); edi_2V(counter_top_0Vg_cnt[18]); edi_2V(counter_top_0Vg_cnt[18]); edi_2V(counter_top_0Vg_cnt[18]); edi_2V(counter_top_0Vg_cnt[18]); edi_2V(counter_top_0Vg_cnt[18]); edi_2V(counter_top_0Vg_cnt[18]); edi_2V(counter_to</pre>
Event Counter Frequency Monitor User Clock Frequencies	Assign to Channel A       -> sd1_2\/counter_top_0\/g_cnt[194]_counter/\/reg_counter[9]_net_1:sd1_;         Assign to Channel B       ->         Unassign Channels       ->

Figure 29 · Activating the Event Counter - Assign Probe Channel



#### **Running the Event Counter**

Event Counter automatically runs the counter, which is indicated by a green LED. The counts are updated every second, and are shown next to Total Events. FPGA Array debug data and the control tabs in the Event Counter panel are disabled while Event Counter is running. When a signal is assigned, the signal name appears next to Signal.

Ŧ			
ve/Active Probes Selection	8 ×	FPGA Array debug data	
Herarchical View Netlist View		Live Probes Active Probes Memory Blocks Probe Insertion	
Filter:	Search	Delete Delete	a Al
Instance(s):	Add	Name	1
> 1 sd1_0		sd1_2\/counter_top_0\/g_cnt[199]_counter\/reg_counter(9]_net_1:sd1_2\/counter_top_0\/g_c	nt[15
> = sd1_1\ = = sd1_2\	E	sd1_2\/counter_top_0\/g_cnt[198]_counter\/reg_counter(9]_net_1:sd1_2\/counter_top_0\/g_c	nt[15
E counter_top_0\ Counter_top_1\		sd1_2\/counter_top_0\/g_cnt[197]_counter\/reg_counter[9]_net_1:sd1_2\/counter_top_0\/g_c	nt[15
b E counter too 21		sd1_2\/counter_top_0\/g_cnt[196]_counter\/reg_counter[9]_net_1:sd1_2\/counter_top_0\/g_c	nt[15
Event Counter/Frequency Hon	itor	sd1_2\/counter_top_0\/g_cnt[195]_counter\/reg_counter[9]_net_1:sd1_2\/counter_top_0\/g_c	nt[15
	-	sd1_2\/counter_top_0\/g_cnt[194]_counter\/reg_counter[9]_net_1:sd1_2\/counter_top_0\/g_c	nt[15
Activate Event Counter	Reset	sd1_2\/counter_top_0\/g_cnt(193)_counter\/reg_counter(9)_net_1:sd1_2\/counter_top_0\/g_c	nt[15
Edge Selected: Rising Time elapsed (s):	Stop	sd1_2V/counter_top_0V/g_cnt[192]_counter_Vreg_counter[9]_net_1:sd1_2V/counter_top_0V/g_c	nt[15
Total Events: 527635	]	sd1_2Vcounter_top_0Vg_cnt[191]_counterVreg_counter[9]_net_1:sd1_2Vcounter_top_0Vg_c	nt[15
Signal : sd1_2\/counter_top_0\/g_c	nt[194]_counter\/r	sd1_2Vcounter_top_0Vg_cnt[190]_counterVreg_counter[9]_net_Ltsd1_2Vcounter_top_0Vg_c	nt[15
		sd1_2Vcounter_top_0Vg_cnt[189]_counterVreg_counter[9]_net_1:sd1_2Vcounter_top_0Vg_c	nt[18
		sd1_2V/counter_top_0Vg_cnt(188)_counterVreg_counter(9)_net_1:sd1_2V/counter_top_0Vg_c	nt[18_
		<	•
		Assign to Channel A -> sd1_2\/counter_top_0\/g_cnt[194]_counter\/reg_counter[9]_net	_1:sd1_3
		Assign to Channel B ->	
		Unassign Channels	
Event Counter Prequency Monit	tor User Clock Prequencies		
Tabs disa		Window disable d	

Figure 30 · Running the Event Counter

#### **Stopping the Event Counter**

The only button enabled when Event Counter is running is the "Stop" button. Click button to stop counting. A red LED is shown to indicates the Event Counter has stopped. FPGA Array debug data and the control tabs in the Event Counter panel are enabled when Event Counter is not running.

ug FPGA Array	And in case of the local division of the loc	
tive Probes Selection	8 ×	FPGA Array debug data
rarchical View Netlist View		Live Probes Active Probes Memory Blocks Probe Insertion
81	Search	Delete Delete All
tance(s):	Add	Name
5 sd1.0\		sd1_2\/counter_top_0\/g_ant[199]_counter\/reg_counter[9]_net_1:sd1_2\/counter_top_0\/g_ant[19
<pre>sd1_1\ sd1_2\</pre>	1	sd1_2\/counter_top_0\/g_cnt[198]_counter\/reg_counter[9]_net_1:sd1_2\/counter_top_0\/g_cnt[15
counter_top_0\		sd1_2\/counter_top_0\/g_cnt[197]_counter\/reg_counter[9]_net_1:sd1_2\/counter_top_0\/g_cnt[15
counter_top_1     counter_top_2	-	
		sd1_2\/counter_top_0\/g_cnt[196]_counter\/reg_counter[9]_net_1:sd1_2\/counter_top_0\/g_cnt[15
nt Counter/Frequency Monitor		sd1_2\/counter_top_0\/g_cnt[195]_counter\/reg_counter[9]_net_1:sd1_2\/counter_top_0\/g_cnt[15
		sd1_2Vcounter_top_0Vg_ant[194]_counterVreg_counter[9]_net_1:sd1_2Vcounter_top_0Vg_ant[15
Activate Event Counter ●	Reset	sd1_2Vcounter_top_0Vg_cnt[193]_counter\/reg_counter[9]_net_1:sd1_2Vcounter_top_0Vg_cnt[15
ge Selected: Rising		
e elapsed (s):	Stop	sd1_2\/counter_top_0\/g_cnt[192]_counter\/reg_counter[9]_net_1:sd1_2\/counter_top_0\/g_cnt[15
tal Events: 3373309		sd1_2Vcounter_top_0Vg_ant[191]_counterVreg_counter[9]_net_1:sd1_2Vcounter_top_0Vg_ant[15
nal : sd1_2\/counter_top_0\/g_cnt[194]_counter\/r	2	sd1_2\/counter_top_0\/g_ont[190]_counter\/reg_counter[9]_net_1:sd1_2\/counter_top_0\/g_ont[15
		sd1_2Vcounter_top_0Vg_cnt[189]_counterVreg_counter[9]_net_1:sd1_2Vcounter_top_0Vg_cnt[18
		sd1_2\/counter_top_0\/g_ont[188]_counter\/reg_counter[9]_net_1:sd1_2\/counter_top_0\/g_ont[18_
		1 III i stolene problem for the former for the former problem for the former former for the former for the former for the former for the f
		Assign to Channel A -> sd1_2Vcounter_top_0Vg_cnt[194]_counter\/reg_counter[9]_net_1:sd1_
		Levels as causes and as r_r Annune _ab_oAAFout sat_mane AeAFonue [a]_ler_rant_
		Assign to Channel 8 ->
[		Unassign Channels
nt Counter Frequency Monitor User Clock	Frequencies	

Figure 31 · Stopping the Event Counter

**Note:** When a DC signal (signal tied to logic '0') is assigned to Live Probe Channel A, or if there are no transitions on the signal assigned to Live Probe Channel A with initial state '0', the Event Counter value is updated as '1' when the counter is stopped. This is a limitation of the FHB IP, and will be fixed in upcoming releases.

#### See Also

"Frequency Monitor" on page 41 "User Clock Frequencies" on page 52

# **Frequency Monitor**

The Frequency Monitor calculates the frequency of any signal in the design that can be assigned to Live Probe channel A. The Frequency Monitor must be activated before or after the signal is assigned to Live Probe Channel A. You can enter the time to monitor the signal. The accuracy of results increases as the monitor time increases. The unit of measurement is displayed in Megahertz (MHz). During the run, progress is displayed in the pane.

Microsemi

■ e/Active Probes Selection	FPGA Array debug data
Hierarchical View Netlist View	Live Probes Active Probes Memory Blocks Probe Insertion
Filter: Search	Delete All
Instance(s): Add	Name Туре
Primitives	A_DOUT_0_c[8]:URAM_3\vsd_URAM_3_URAM_R0C4/INST_RAM64x18_IP:A_DOUT[0] RAM64x18
▷ IRAM_0\ ▷ IRAM_1\	A_DOUT_0_c[7]:URAM_3\/sd_URAM_3_URAM_R0C3/INST_RAM64x18_IP:A_DOUT[1] RAM64x18
▷ == URAM_2\ ▷ == URAM_3\	A_DOUT_0_c[6]:URAM_3\sd_URAM_3_URAM_R0C3/INST_RAM64x18_IP:A_DOUT[0] RAM64x18
▷ III count 6 0\	A_DOUT_0_c[5]:URAM_3\/sd_URAM_3_URAM_R0C2/INST_RAM64x18_IP:A_DOUT[1] RAM64x18
Event Counter/Frequency Monitor	A_DOUT_0_c[4]:URAM_3\vd_URAM_3_URAM_R0C2/INST_RAM64x18_IP:A_DOUT[0] RAM64x18
	A_DOUT_0_c[3]:URAM_3\/sd_URAM_3_URAM_R0C1/INST_RAM64x18_IP:A_DOUT[1] RAM64x18
Activate Frequency Meter	A_DOUT_0_c[2]:URAM_3\/sd_URAM_3_URAM_ROC1/INST_RAM64x18_IP:A_DOUT[0] RAM64x18
Monitor time (s): 5	A_DOUT_0_c[1]:URAM_3\/sd_URAM_3_URAM_R0C0/INST_RAM64x18_IP:A_DOUT[1] RAM64x18
Frequency (MHz): 0 Signal: A_DOUT_0_c[8]:URAM_3\/sd_URAM_3_URAM_R0C4/INST_RAT	A_DOUT_0_c[0]:URAM_3\/sd_URAM_3_URAM_R0C0/INST_RAM64x18_IP:A_DOUT[0] RAM64x18
Event Counter - Even we Market	Assign to Channel A     -> A_DOUT_0_c[8]:URAM_3\/sd_URAM_3_URAM_R0C4/INST_RAM64x18_IP     Assign to Channel B     ->     Unassign Channels
Event Counter Frequency Monitor User Clock Frequencies	Unassign Ghannels

Figure 32 · Frequency Monitor Tab/UI

In the Frequency Monitor tab, you can activate the Frequency Monitor, change the monitor time (delay to calculate frequency), reset the monitor, and set the frequency in megahertz (MHz). Click the drop-down list to select monitor time value. During the frequency calculation, all tabs on the right side of the window are disabled, as well as the tabs in the FPGA Hardware Breakpoint (FHB) pane.

#### **Activating the Frequency Monitor**

You can activate the Frequency Monitor in either of the following two ways:

• Click Activate Frequency Monitor, and then click the Live Probe tab and assign a signal to Channel A (Channel B is not configured for spatial debug operations).

licrosemi

**Міскоснір** сотралу

ve/Active Probes Selection & X	FPGA Array debug data
Hierarchical View Netlist View	Uve Probes Active Probes Memory Blocks Probe Insertion
Filter: Search	Delete Delete All
Instance(s): Add	Name
> = /0_tbe = 4	sd1_2Vcounter_top_0Vg_cnt[199]_counterVreg_counter[9]_net_1:sd1_2Vcounter_top_0Vg_cnt[15
> = sdi_1\ # = sdi_2\	sd1_2V/counter_top_0Vg_ont[198]_counterVreg_counter[9]_net_1:sd1_2V/counter_top_0Vg_ont[15
✓ 4 sd1_2\	
E counter_top_1\	sd1_2Vcounter_top_0Vg_cnt[197]_counterVreg_counter[9]_net_1:sd1_2Vcounter_top_0Vg_cnt[15
▷ ■ counter ton 2\	sd1_2\/counter_top_0\/g_ant[196]_counter\/reg_counter[9]_net_1:sd1_2\/counter_top_0\/g_ant[15
Event Counter/Frequency Honitor	sd1_2Vcounter_top_0Vg_cnt[195]_counterVreg_counter[9]_net_1:sd1_2Vcounter_top_0Vg_cnt[15
	sd1_2Vcounter_top_0Vg_c Assign to Channel A tet_1:sd1_2Vcounter_top_0Vg_cnt[15
Activate Frequency Meter RESET	sd1_2/counter_top_0/g_c Assign to Channel B tet_1:sd1_2/counter_top_0/g_cnt[15
Monitor time (s): 0.1 •	
Frequency (MHz): 0	sd1_2Vcounter_top_0Vg_cnt[192]_counterVreg_counter[9]_net_1:sd1_2Vcounter_top_0Vg_cnt[15
Signal : sd1_2\/counter_top_0\/g_cnt[192]_counter\/reg_counter[9]_n	sd1_2Vcounter_top_0Vg_cnt[191]_counterVreg_counter[9]_net_1:sd1_2Vcounter_top_0Vg_cnt[15
	sd1_2Vcounter_top_0Vg_cnt[190]_counterVreg_counter[9]_net_1:sd1_2Vcounter_top_0Vg_cnt[15
	an "s Amaga Tahla AB" and sad "maga tab" maga tabla s "s Amaga Tahla AB" and sa
	sd1_2\/counter_top_0\/g_cnt[189]_counter\/reg_counter[9]_net_1:sd1_2\/counter_top_0\/g_cnt[18
	sd1_2\/counter_top_0\/g_cnt[188]_counter\/reg_counter[9]_net_1:sd1_2\/counter_top_0\/g_cnt[18
	4 M
	Assign to Channel A -> sd1_2\/counter_top_0\/g_cnt[192]_counter\/reg_counter[9]_net_1:sd1_;
	Assign to Channel B ->
	Unassign Channels
Event Counter Frequency Monitor User Clock Frequencies	Contraction of the second se

Figure 33 · Activating the Frequency Monitor - Assign a Signal

• Click the Live Probe tab and assign a signal to Channel A, and then click the Frequency Monitor tab and check the Activate Frequency Monitor checkbox.

Microsemi

а 🔨 Міскоснір company

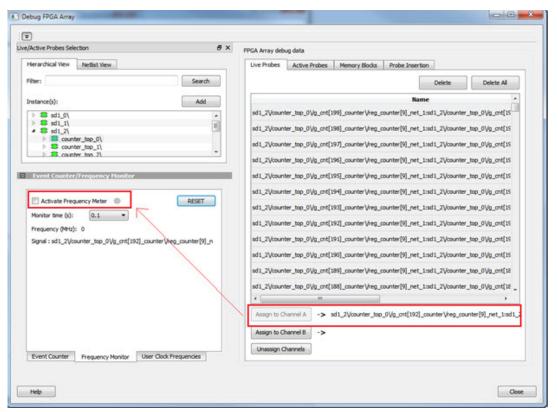


Figure 34 · Activating the Frequency Monitor

#### **Running the Frequency Monitor**

The Frequency Monitor runs automatically, and is indicated by a green LED. While it is running, FPGA Array debug data and the control tabs in the panel are disabled. A progress bar shows the monitor time progress when it is 1 second and above (as shown in the following figure). The Reset button is also disabled during the run. When a signal is assigned, the signal name appears next to Signal.

Microsemi

**Міскоснір** company

ve/Active Probes Selection	
	FPGA Array debug data
Herarchical Vew Netlist View	Live Probes Active Probes Memory Blocks Probe Insertion
Filter: Search	Delete All
Instance(s): Add	Name
> 2 /0_lbe 2 4	sd1_2\/counter_top_0\/g_cnt[199]_counter\/reg_counter[9]_net_1:sd1_2\/counter_top_0\/g_cnt[15
▷ \$\$ sd1_1\ \$ sd1_2\ \$ sd1_2\	sd1_2Vcounter_top_0Vg_ant[198]_counterVreg_counter[9]_net_1:sd1_2Vcounter_top_0Vg_ant[15
E counter_top_0\ E counter_top_1\	sd1_2Vcounter_top_0Vg_ant[197]_counter\keg_counter[9]_net_1:sd1_2Vcounter_top_0Vg_ant[15
D a counter too 21	sd1_2Vcounter_top_0Vg_cnt[196]_counterVreg_counter[9]_net_1:sd1_2Vcounter_top_0Vg_cnt[15
Event Counter/Frequency Monitor	sd1_2\/counter_top_0\/g_cnt[195]_counter\/reg_counter[9]_net_1:sd1_2\/counter_top_0\/g_cnt[15
	sd1_2\/counter_top_0\/g_cnt(194)_counter\/reg_counter(9)_net_1:sd1_2\/counter_top_0\/g_cnt(19
Activate Frequency Meter     RESET	sd1_2\/counter_top_0\/g_cnt[193]_counter\/reg_counter[9]_net_1:sd1_2\/counter_top_0\/g_cnt[15
Monitor time (s): 5 -	sd1_2Vcounter_top_0Vg_ont[192]_counterVreg_counter[9]_net_1:sd1_2Vcounter_top_0Vg_ont[15
Frequency (MHz): 0	sd1_2Vcounter_top_0Vg_cnt[191]_counterVreg_counter[9]_net_1:sd1_2Vcounter_top_0Vg_cnt[15
Signal : sd1_2Vcounter_top_0Vg_cnt[192]_counterVreg_counter[9]_n	sd1_2\/counter_top_0\/g_cnt[190]_counter\/reg_counter[9]_net_1:sd1_2\/counter_top_0\/g_cnt[15
	sd1_2\/counter_top_0\/g_cnt[189]_counter\/reg_counter[9]_net_1:sd1_2\/counter_top_0\/g_cnt[18
	sd1_2Vcounter_top_0Vg_ont[188]_counterVreg_counter[9]_net_1:sd1_2Vcounter_top_0Vg_ont[18
	4 [
	Assign to Channel A -> sd1_2\/counter_top_0\/g_cnt[192]_counter\/reg_counter[9]_net_1:sd1_;
	Assign to Channel 8 ->
	Unassign Channels
Event Counter Frequency Monitor User Clock Frequencies	
Tabs disabled	Window disabled

Figure 35 · Running the Frequency Monitor

#### **Stopping the Frequency Monitor**

The Frequency Monitor stops when the specified monitor time has elapsed. This is indicated by a red LED. The result appears next to Frequency. The window and the tabs on the control panel are enabled. The Reset button is also enabled to reset the Frequency to 0 to start over the next iteration. The progress bar is hidden when the Frequency Monitor stops.

*licrosemi* 

а 🔨 Міскоснір company

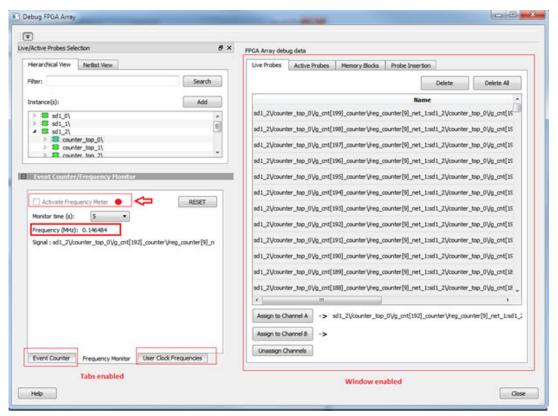


Figure 36 · Stopping the Frequency Monitor

#### See Also

"Event Counter" on page 38 "User Clock Frequencies" on page 52

# FPGA Hardware Breakpoint Auto Instantiation

The FPGA Hardware Breakpoint (FHB) Auto Instantiation feature automatically instantiates an FHB instance per clock domain that is using gated clocks (GL0/GL1/GL2/GL3) from an FCCC instance. The FHB instances gate the clock domain they are instantiated on. These instances can be used to force halt the design or halt the design through a live probe signal. Once a selected clock domain or all clock domains are halted, you can play or step on the clock domains, either selectively or all at once. The FPGA Hardware Breakpoint controls in the SmartDebug UI allow you to control the debugging cycle.

**Note:** The FHB Auto-Instantiation for RTG4 will instantiate a CCC/PLL which does not enable the insertion of PLL Loss of Lock auto-reset logic. This auto-reset logic is recommended when using the RTG4 in production/flight applications per Customer Notifications CN19009 and PCN18009.7 on the Microsemi website. The PLL auto-reset logic is not required in lab environments where the RTG4 is not exposed to thermal cycling or radiation environments. Therefore, it is not recommended to generate a flight bitstream of an RTG4 design which includes the FHB instantiation.

To enable this option, select the Enable FHB Auto Instantiation check box in the Design flow tab of the Project Settings dialog box (Libero > File > Project Settings).

Note: FHB auto-instantiation can also be done in the "Import netlist as VM file" flow.

See the example figure that follows.

Nicrosen

MICROCHIP company



Project settings		00
ngiect settings Device selection Device selection Device settings Design flow Design flow Do fle Waveforms View commands Timescale Simulation Rearies Simulation Rearies	HOL source fles language options         Libers Soft supports mixed HOL language designs; you can inport liveling and VHOL in the same project. For VHOL fles, you may choose between HOL 2008 and VHOL 49.         Verified         © System Variance         @ VHOL         @ VHOL ADD         WHOL         @ VHOL ADD         WHOL         @ VHOL ADD         @ VHOL ADD         @ VHOL ADD         @ VHOL ADDD         WHOL         @ VHOL ADDD         @ VHOL ADDD         WHOL         @ VHOL ADDD         WHOL         @ VHOL ADDD         @ VHOL ADDD         WHOL         @ VHOL ADDD         @ VHOL ADDD         WHOL         @ VHOL ADDD         @ VHOL ADDD         @ VHOL ADDD         @ VHOL ADDD         @ VHOL         @ VHOL         @ VHOL	Sare Decard
	Working     Working     Book flow     Devalue block creation     Root ad I     Devalue synthesis     Devalue Frick Hardware Breakpoint Auto Instantiation	
	Synthesis gate level netlist format  Werlog netlist Design nethodology  Use standalone initialization for MDDR/PDDR/SERDES pergherals Reports	
Help	Maximum number of high fanout nets is be displayed: 10           IV         Abort flow if errors are found in Physical Design-Constraints (POC)         IV           IV         Abort flow if errors are found in Triang Constraints (POC)	Gee

Figure 37 · Enable FHB Auto Instantiation in Project Settings Dialog Box: Design flow Tab

FPGA Hardware Breakpoint (FHB) controls appear in the Debug FPGA Array dialog box when there is an autoinstantiated FHB instance in the design. See the example figure that follows.

<b>v</b>					
Active Probes Selection	8 ×	FPGA Array debug data			
Herarchical View Netlist View		Live Probes Active Probes Memo	ory Blocks		
Filter:	Search	+ - + + Save	Load	Delete	Delete All
Instance(s):	Add	Name	Type	Read Value	Write Value
	AOU	FCCC_0_Count_c[19:0]	DFF	20'hDD849	20'h
E count_epcs_0\ E count_epcs_1\		FCCC_1_Count_c[19:0]	DFF	20'h47D45	20h
S count_epcs_2\		FCCC_2_Count_c[19:0]	DFF	20'hC1548	20'h
FPGA Hardware BreakPoint     Operate on All Clock Domains      Operate os     Select Clock Domain :      [FCCC_0/GL0_INST     Trigger Setup     Trigger Setup     Trigger Signal : Not Connected	n Selected Clock Domain				
Operate on All Clock Domains      Operate of Select Clock Domain :      FCCC_0/GL0_PKST     Trigger Setup		Read Active Probes S	ave Active Probes' Di		tive Probes

Figure 38 · FPGA Hardware Breakpoint (FHB) Controls

You can choose **Operate on All Clock Domains** or **Operate on Selected Clock Domain** by selecting the appropriate radio button. Selecting either of these modes sets the FHB instances to the respective mode. Once you assign the Live Probe PROBE\_A connection and click **Arm Trigger**, the DUT halts on the next positive edge that occurs on the signal connected to Live Probe PROBE\_A.



When you choose Operate on Selected Clock Domain mode, the Select Clock Domain combo box is enabled,

and all available clock domains are listed. The Halt (Pause) , Play , and Step buttons are associated for that clock domain. If you switch between clock domains in this mode, previous clock domain settings are not retained.

Note: The Operate on Selected Clock Domain mode is not supported for RTG4 devices.

When you choose **Operate on All Clock Domains** mode, the Select Clock Domain combo box is disabled. The Halt, Play, and Step buttons are associated for all clock domains.

The Trigger Signal is shown as Not Connected until a live probe is assigned. See the example figure that follows.

8 ×	FPGA Array debug data			
	Live Probes Active Probes Memory Blocks Probe Insertion			
Search	+ - + + Save	Load	Delete	Delete All
Add	Name	Туре	Read Value	Write Value
	▷ q_0_c[19:0]	DFF	20'h7FCEC	20'h
	▷ q_2_c[19:0]	DFF	20'h1AF18	20h
		DFF	20'hF3398	20'h
	▷ q_3_c[19:0]	DFF	20'h0DC4C	207h
Arm Trigger				
Arm Trigger				
Arm Trigger				
	Search Add	Search         Image: Constraint of the search of the	Search         Add           Add         Name         Type                Q_0_c(19:0)         DFF                Q_1_c(19:0)         DFF                Q_1_c(19:0)         DFF                Q_1_c(19:0)         DFF                Q_1_c(19:0)         DFF                Q_1_c(19:0)         DFF                Q_1_c(19:0)         DFF	Search         Image: Save         Load         Delete           Add         Image: Save         Load         Delete           Name         Type         Read Value         Image: Save         Delete           Name         Type         Read Value         Image: Save         Delete           Name         Type         Read Value         Image: Save         Delete           Image: Save         Delete         Delete         Image: Save         Delete           Image: Save         Delete         Type         Read Value         Image: Save         Delete           Image: Save         Delete         Type         Read Value         Image: Save         Delete           Image: Save         Delete         Type         Read Value         Image: Save         Delete           Image: Save         Delete         Type         Read Value         Image: Save         Delete           Image: Delete         Q.Sc(19:0)         DFF         Zohr/Mage: Save         Delete         Image: Save           Image: Delete         Delete         Delete         Delete         Zohr/Mage: Save         Delete           Image: Delete         Delete         Delete         Delete

When a probe is assigned to Live Probe PROBE\_A, the Trigger Signal updates.

If you require a certain number of clock cycles before halting the clock domain after triggering, a value between 0 and 255 must be entered for Delay Cycles Before Halt before you click **Arm Trigger**. This sets the FHBs to trigger after the specified delay from the rising edge trigger.

Delay is not applied to a forced Halt. See the example figure that follows.



•					
ve/Active Probes Selection	8 X	FPGA Array debug data			
Hierarchical View Netlist View		Live Probes Active Probes	Memory Blocks Probe	Insertion	
Filter:	Search	+ - + + Save	Load	Delete	Delete Al
Instance(s):	Add	Name	Туре	Read Value	Write Value
		> q_0_c[19:0]	DFF	20'h7FCEC	20'h
count_0\ \$\$ count_1\		▷ q_2_c[19:0]	DFF	20'h1AF18	20'h
D == count_2\		≥ q_1_c[19:0]	DFF	20'hF3398	201h
Image: Second State S		▷ q_3_c[19:0]	DFF	20'h0DC4C	20%
PEGA Hardware BreakPoint     Operate on Al Clock Domains      Operate     Select Clock Domain :      PCCC_0/GL0_INST     Trigger Setup     Trigger Setup     Trigger Setup	e on Selected Clock Domain				
Operate on Al Clock Domains     Operate     Select Clock Domain :      FCCC_0/GLO_INST     Trigger Setup					
Select Clock Domain : [FCCC_0/R.0_INST Trigger Setup Trigger Signal : Not Connected Edge Selected: Rising					
Operate on Al Clock Domains Operat Select Clock Domain : FCCC_0/GL0_INST Trigger Setup Trigger Signal : Not Connected Edge Selected: Rising Delay Cycles Before Halt : 240	* 	Red Active Probes	Save Active Probes' D	Marka a	ctive Probes

When a live probe connection is made and you click **Arm Trigger**, FPGA Hardware Breakpoint functionality is disabled until the trigger is disarmed automatically or the design is force halted.

e/Active Probes Selection	8 ×	FPGA Array debug data			
Herarchical View Netlist View		Live Probes Active Probes Memory Blocks Probe Insertion			
Filter:	Search	+ - + + Save	Load	Delete	Delete All
Instance(s):	Add	Name	Туре	Read Value	Write Value
		▶ q_0_c[19:0]	DFF	20'h7FCEC	20%
E count_0\ E count_1\		▷ q_2_c[19:0]	DFF	20'h1AF18	201h
D = count_1		▷ q_1_c[19:0]	OFF	20'hF3398	201h
Image: Second 3\		▷ q_3_c[19:0]	DFF	20'h0DC4C	20%
PEGA Hardware BreakPoint     @ Operate on All Clock Domains Operat     Select Clock Domain : [FCCC_0/GL0_INST     Trigger Setup     Trigger Setup     Trigger Signal : q_0_c[17]:count_0Vq[17]     Edge Selected: Rising	*	L,			
Operate on All Clock Domains Operate Select Clock Domain : PCCC_0/GL0_RIST Trigger Setup Trigger Signal : q_0_(17):count_0/Q(17)	*				
Operate on All Clock Domains Operate Select Clock Domain : FCCC_0/GL0_RIST Trigger Setup Trigger Signal :0_c[17]:count_0/va[17] Edge Selected: Rising	*				
Coperate on Al Clock Domains Coperate Select Clock Domains FCCC_0/R40_RKST Trigger Setup Trigger Setup Edge Selected: Rising Delay Cycles Before Halt: 240 RESET	*) :Q				
Select Clock Domain : [PCCC_0/GL0_INST Trigger Setup Trigger Signal : q_0_c[17]:count_0Vq[17] Edge Selected: Raing Delay Cycles Before Halt : 240 RESET	*) :Q	Read Active Probes 5	ive Active Probes' D		tive Probes



#### **Trigger Input**

You can use the trigger input signal if you want an event in the DUT to trigger the FHB IP (for example, a particular state in the FSM or counter value, and so on) when this signal is asserted. If the trigger signal is already asserted (or HIGH) at the time of arming the FHB, the DUT is halted immediately.

Force Halt/Play/Step is done using the FPGA Hardware Breakpoint controls (see the example figure that follows). Once the clock domain is halted, you can either force Play the clock domain or Step the clock domain by 1 clock cycle.

You can save the waveform view of the selected active probes using Export Waveform by specifying the number of clock cycles to capture. The waveform is saved to a .vcd file.

e/Active Probes Selection	8 ×	FPGA Array debug data			
Hierarchical View Netlist View		Live Probes Active Probes Memory Blocks Probe Insertion			
Filter:	Search	+ - + + Save	Load	Delete	Delete All
Instance(s):	Add	Name	Type	Read Value	Write Value
			DFF	20h201E2	20'h
count_0\ secont_1\		q_2_c[19:0]	DFF	20h201E2	20'h
E count_2\		▷ q_1_c[19:0]	DFF	20h201F2	20'h
Image: Second State S		▷ q_3_c[19:0]	OFF	20h201F2	207h
Operate on All Clock Domains      Operate on All Clock Domains     Operate on All Clock Domains     PCCC_0/RL0_INST     Trigger Setup     Trigger Setup     Trigger Signal : q_0_c[17]:count_0/va[	•				
Operate on All Clock Domains      Ope Select Clock Domain : [PCCC_0/GL0_INST Trigger Setup	•				
Select Clock Domain : [FCCC_0/GL0_INST Trigger Setup Trigger Signal : q_0_c[17]:count_0\q[ Edge Selected: Rising	•				

#### **FPGA Hardware Breakpoint Operations**

#### **Live Probe Halt**

You can halt a selected clock domain or all clock domains in Live Probe Halt mode based on the mode selection (**Operate on All Clock Domains** or **Operate on Selected Clock Domain**).

Assign a signal to Live Probe PROBE\_A in the **Live Probes** tab of the UI, and then click the **Active Probe** tab to see the FPGA Hardware Breakpoint controls.

Click Arm Trigger to arm the FHBs to look for a trigger on the signal connected to Live Probe PROBE\_A.

Once the trigger occurs, the clock domains are halted.

**Note:** If only one clock domain is halted, other clock domains continue to run, and you should anticipate results accordingly.

Note: Live Probe Halt can be delayed for a maximum of 255 clock cycles.

The actual delay realized on hardware is calculated by the following equation:

Actual delay cycles on hardware =

```
#Delay clock cycles before halt mentioned in smartdebug * (DUT clock frequency/FHB
clock frequency)
```



FHB clock frequency is device specific: SmartFusion2: 50MHz IGLOO2: 50MHz RTG4: 100MHz See Assumptions and Limitations for more information.

#### **Force Halt**

You can force halt a selected clock domain or all clock domains based on mode selection without having to wait for a trigger from a live probe signal. Click the **Halt** button in the FPGA Hardware Breakpoint (FHB) controls.

In **Operate on Selected Clock Domain** mode, the state of the Halt button is updated based on the state of the clock domain selected.

In **Operate on all Clock Domains** mode, the Halt button is disabled only when all clock domains are halted. Each clock domain is halted sequentially in the order shown in the Select Clock Domain combo box.

**Note:** If only one clock domain is halted, other clock domains continue to run, and you should anticipate results accordingly.

#### Play

Once the clock domain is in a halted state (live probe halt or force halt), you can click **Play** in the FPGA Hardware Breakpoint controls. This resumes the clock domain from the halted state.

In **Operate on all Clock Domains** mode, each clock domain runs sequentially in the order shown in the Select Clock Domain combo box.

#### Step

Once the clock domain is in a halted state (live probe halt or force halt), you can click the **Step** button in the FPGA Hardware Breakpoint controls. This advances the clock domain by one clock cycle and holds the state of the clock domain.

In **Operate on All Clock Domains** mode, each clock domain steps sequentially in the order shown in the Select Clock Domain combo box.

#### Waveform Capture

You can save the waveform view of the selected active probes using Export Waveform by specifying the number

of clock cycles to capture in text box and then clicking **Capture Waveform**. The waveform is saved to a .vcd file.

You can view the waveforms by importing the .vcd file. The waveform file can be viewed in any waveform viewer that supports vcd format.

#### Reset

You can reset a selected clock domain or all clock domains (based on the mode selection) by clicking **RESET** at any time. This resets the FHBs on clock domains and instructs FHB muxes not to look for a trigger.

#### **Assumptions and Limitations**

- If you select the auto instantiation option in Libero, you need to rerun Synthesis (if already run) to get the FHB related functionality.
- Supported for FCC driven gated clocks (GL0/GL1/GL2/GL3) only.
- CLKINT\_PRESERVE FHB is not auto-instantiated if the user design contains this macro.
- Designs that have Encrypted IPs are not supported.
- EDIF using constraints flow is not supported.
- Live Probe triggering occurs on the Positive Edge only.



- For imported verilog netlist files (.vm files), you must rerun synthesis to get FHB-related functionality. If synthesis is disabled and the netlist is compiled directly, FHB functionality is not inferred.
- If only one clock domain is halted during operations, other clock domains continue to run, and you should anticipate results accordingly.
- FHB performance can only be characterized against the clock which it is running at (i.e. 50MHz).
  - o If the DUT clock is running at or less than 50MHz, the DUT clock will halt within one clock cycle (1 or less).
  - o For frequencies higher than 50MHz, the point at which the DUT halts cannot be guaranteed.

# **User Clock Frequencies**

The User Clock Frequencies tab shows the frequencies that have been configured from the FCCC block. If assigned, live probe channels are temporarily unassigned, and reassigned after user clock frequencies have been calculated. The Refresh button recalculates frequencies if clocks have been changed.

Active Probes Selection		S × FPGA Array debug data	
Hierarchical View Netlist Vi	ew	Live Probes Active Probes Memory Blocks Probe Insertion	
Filter:	Search	Delete	Delete All
Instance(s):	Add	Name	Туре
Primitives		A_DOUT_0_c[8]:URAM_3\/sd_URAM_3_URAM_R0C4/INST_RAM64x18_IP:A_DOUT[0]	RAM64x18
URAM_0\ URAM_1\		A DOUT 0 c[7]:URAM_3\/sd_URAM_3 URAM_R0C3/INST_RAM64x18_IP:A_DOUT[1]	RAM64x18
> = URAM_1			
URAM_3\ E count 6 0\		A_DOUT_0_c[6]:URAM_3\/sd_URAM_3_URAM_R0C3/INST_RAM64x18_IP:A_DOUT[0]	RAM64x18
		A_DOUT_0_c[5]:URAM_3\/sd_URAM_3_URAM_R0C2/INST_RAM64x18_IP:A_DOUT[1]	RAM64x18
Event Counter/Frequence	cy Monitor	A_DOUT_0_c[4]:URAM_3\/sd_URAM_3_URAM_R0C2/INST_RAM64x18_IP:A_DOUT[0]	RAM64x18
		A_DOUT_0_c[3]:URAM_3\/sd_URAM_3_URAM_R0C1/INST_RAM64x18_IP:A_DOUT[1]	RAM64x18
User Clocks	Frequency (MHz)	A_DOUT_0_c[2]:URAM_3\/sd_URAM_3_URAM_R0C1/INST_RAM64x18_IP:A_DOUT[0]	RAM64x18
1 FCCC_0_GL0	~24.5	A_DOUT_0_c[1]:URAM_3\/sd_URAM_3_URAM_R0C0/JINST_RAM64x18_IP:A_DOUT[1]	RAM64x18
2 FCCC_0_GL1	~48.7	A_DOUT_0_c[0];URAM_3\/sd_URAM_3_URAM_R0C0/INST_RAM64x18_IP;A_DOUT[0]	RAM64x18
3 FCCC_0_GL2	~97.4	Higher Collaboration Star Street Stre	10110 1010
4 FCCC_0_GL3	~194.6		
		۲. m	•
		Assign to Channel A -> A_DOUT_0_c[8]:URAM_3\/sd_URAM_3_URAM_R0C4/INS	T_RAM64x18_IP
2		Assign to Channel B ->	
Event Counter Frequence	y Monitor User Clock Frequencies	Unassign Channels	

Figure 39 · User Clock Frequencies Tab/UI

#### See Also

"Event Counter" on page 38 "Frequency Monitor" on page 41 <u>UG0449- SmartFusion2 and IGLOO2 Clocking Resources User Guide</u> UG0586- RTG4 FPGA Clocking Resources User Guide



# **Pseudo Static Signal Polling**

With Active Probes you can check the current state of any probe in the design. However, in most cases, you will not able to time the active probe read to capture its intended value. For these cases, you can use Pseudo Static Signal Polling, in which the SmartDebug software polls the signal at intervals of one second to check if the probe has the intended value. This feature is useful in probing signals which reach the intended state and stay in that state.

From the Active Probes tab in the Debug FPGA Array dialog box, right-click a signal, bus, or group and choose **Poll...** See the example figure that follows.

/Active Probes Selection	ð×	FPGA Array debug data			
Herarchical View Netlist View			ory Blocks Probe	Insertion	
Filter: Se	earch	+ - + + Save	Load	Delete	Delete All
Instance(s):	Add	Name	Туре	Read Value	Write Value
		Shift_Reg_0/shft_reg[13:0]	DFF	14h0001	14h
B D_FF_0     A B Primitives	^	D_FF_0/q_0:D_FF_0/q:Q	DFF	0	· ·
Primoves     g		And the second sec		Read	
A Shift Reg 0				Delete	
Primitives					
A > shft_reg				Poll	
shft_reg[0]				Create Group	
shft_reg[1] shft_reg[2]					
shft_reg[3]					
shft_reg[4]					
shft_reg[5]					
shft_reg[6] shft_reg[7]					
sht_reg[3]		Read Active Probes	Save Active Probes' D	ata Write Act	tive Probes
and editor					

Figure 40 · Debug FPGA Array Dialog Box - Poll Option

The Pseudo-static signal polling dialog box opens.

#### **Scalar Signal Polling**

#### **Polling Setup**

To poll scalar signals, select Poll for 0 or Poll for 1.

The selected signal is polled once per second. It should be used for pseudo-static signals that do not change frequently. The elapsed time is shown next to **Time Elapsed in seconds**.

To begin polling, click **Start Polling**. See the following example figure.

Pseudo-static signal polling		? ×
Signal : D_FF_0/q_0:D_FF_0/q:C		
Polling Setup		
Poll for 0	Poll for 1	
	d once per second. It should be used for pseudo-static signals th it pseudo-static signal polling, click the Help button. Start Polling Stop Polling	aat do not change frequently.

Figure 41 · Pseudo-static signal polling Dialog Box (Scalar Signal Polling) - Start Polling



To end polling, click Stop Polling. See the following example figure.

Pseudo-static signal poll	ng	8 ×
Signal : D_FF_0/q_0:D_FF_0	/q:Q	
Polling Setup		1
Poll for 0	O Poll for 1	
	polled once per second. It should be used for pseudo-static signals that d about pseudo-static signal polling, click the Help button. Start Polling Stop Polling	o not change frequently.
Help		Close

Figure 42 · Pseudo-static signal polling Dialog Box (Scalar Signal Polling) - Stop Polling

Note: You cannot change the poll value or close the polling dialog box while polling is in progress.

The elapsed time is updated in seconds until the polled value is found. When the polled value is found, **User value matched** is displayed in green in the dialog box. See the following example figure.

Pseudo-static signal polling		S ≤ ×
Signal : D_FF_0/q_0:D_FF_0/q:Q		
Polling Setup		
Poll for 0	Poll for 1	
	once per second. It should be used for pseudo-static signals t pseudo-static signal polling, click the Help button.	hat do not change frequently.
Help	Stop Politing	Close

Figure 43 · Pseudo-static signal polling Dialog Box (Scalar Signal Polling) - User Value matched

#### **Vector Signal Polling**

To poll vector signals, enter a value in the text box. The entered value is checked and validated. If an invalid value is entered, start polling is disabled, and an example displays showing the required format. See the following example figures.



Pseudo-s	tatic signal polling			<u>ଟ</u> ×
Signal : Shir	ft_Reg_0/shft_reg			
Polling Set.	up			
Poll for	14'h0			
				als that do not change frequently.
	and the second states of the	eudo-static signal polling, d	ick the Help button.	
Time Elapse	ed in seconds: 0			
		Start Polling	Stop Polling	
Help				Close

Figure 44 · Pseudo-static signal polling Dialog Box (Vector Signal Polling)

Pseudo-st	tatic signal polling		ି <mark>×</mark>
Signal : Shi	ft_Reg_0/shft_reg		
Polling Set.	up		
Poll for	14'h		
	lid hex value. Eg: 14'h0		
Time Elapse	ed in seconds: 0	Start Polling Stop Polling	
		[ start to my]	
Help			Close
Help			Close

Figure 45 · Pseudo-static signal polling Dialog Box (Vector Signal Polling) -- After Validation When you enter a valid value and click **Start Polling** is clicked, polling begins.

To end polling, click Stop Polling.

**Note:** You cannot change the poll value or close the polling dialog box while polling is in progress. The elapsed time is updated in seconds until the polled value is found. When the polled value is found, **User value matched** is displayed in green in the dialog box.



# Debug SERDES (SmartFusion2, IGLOO2, and RTG4)

You can examine and debug the SERDES blocks in your design in the Debug SERDES dialog box (shown in the figure below).

To Debug SERDES, expand SmartDebug in the Design Flow window and double-click Debug SERDES.

Debug SERDES Configuration is explained below. See the <u>PRBS Test</u> and <u>Loopback Test</u> topics for information specific to those procedures.

**SERDES Block** identifies which SERDES block you are configuring. Use the drop-down menu to select from the list of SERDES blocks in your design.

#### **Debug SERDES - Configuration**

#### **Configuration Report**

The Configuration Report output depends on the options you select in your <u>PRBS Test</u> and <u>Loopback Tests</u>. The default report lists the following for each Lane in your SERDES block:

Lane mode - Indicates the programmed mode on a SERDES lane as defined by the SERDES system register.

**PMA Ready** - Indicates whether PMA has completed its internal calibration sequence for the specific lane and whether the PMA is operational. See the <u>SmartFusion2</u> or <u>IGLOO2</u> High Speed Serial Interfaces User Guide on the Microsemi website for details.

**TxPII status** - Indicates the loss-of-lock status for the TXPLL is asserted and remains asserted until the PLL reacquires lock.

**RxPLL status** - Indicates the CDR PLL frequency is not grossly out of range of with incoming data stream.

Click **Refresh Report** to update the contents of your SERDES Configuration Report. Changes to the specified SERDES register programming can be read back to the report.

#### SERDES Register Read or Write

**Script** - Runs Read/Write commands to access the SERDES control/status register map using a script. Enter the full pathname for the script location or click the browse button to navigate to your script file. Click **Execute** to run the script.



	SERDES Block: SERDESIF_0	
Debug SERDES	Configuration Report:	
Configuration Tests PRBS Test Loopback Test	Serdes Block SERDESIF_0: Lane 0: Lane mode : EPCS (custom) PMA Ready : True TxPLL status : Locked RxPLL status : Locked Lane 1: Lane mode : EPCS (custom) PMA Ready : True TxPLL status : Locked Lane 2: Lane mode : EPCS (custom) PMA Ready : True TxPLL status : Locked RxPLL status : Locked RxPLL status : Locked Lane 3: Lane mode : EPCS (custom) PMA Ready : True TxPLL status : Locked Lane 3: Lane mode : EPCS (custom) PMA Ready : True TxPLL status : Locked Lane 3: Lane mode : EPCS (custom) PMA Ready : True TxPLL status : Locked RxPLL status : Locked	<ul> <li>Refresh Report</li> <li>E</li> </ul>
	SERDES Register Read or Write: Script:	Execute

Figure 46 · Debug SERDES - Configuration

Note: The PCIe and XAUI protocols only support PRBS7. The EPCS protocol supports PRBS7/11/23/31.

## Debug SERDES – Loopback Test

Loopback data stream patterns are generated and checked by the internal SERDES block. These are used to self-test signal integrity of the device. You can switch the device through predefined tests.

See the PRBS Test topic for more information about the PRBS test options.

**SERDES Block** identifies which SERDES block you are configuring. Use the drop-down menu to select from the list of SERDES blocks in your design.

#### **SERDES Lanes**

Select the **Lane** and **Lane Status** on which to run the Loopback test. Lane mode indicates the programmed mode on a SERDES lane as defined by the SERDES system register.

#### **Test Type**

**PCS Far End PMA RX to TX Loopback-** This loopback brings data into the device and deserializes and serializes the data before sending it off-chip. This loopback requires 0PPM clock variation between the TX and RX SERDES clocks.

See the <u>SmartFusion2</u> or <u>IGLOO2</u> High Speed Serial Interfaces User's Guide on the Microsemi website for details.



**Near End Loopback (On Die)** - To enable, select the Near End Loopback (On Die) option and click **Start**. Click **Stop** to disable. Using this option allows you to send and receive user data without sending traffic off-chip. You can test design functionality without introducing other issues on the PCB.

See the <u>SmartFusion2</u> or <u>IGLOO2</u> High Speed Serial Interfaces User's Guide on the Microsemi website for details.

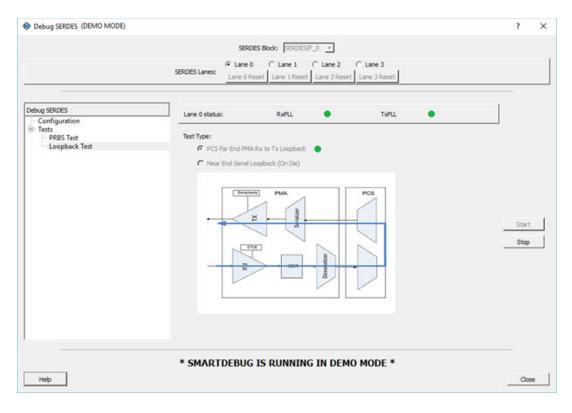
S	SERDES Block: SERDESIF_0   SERDES Lane 0  Lane 1  Lane 2  Lane 3 Lane 0 Reset Lane 1 Reset Lane 2 Reset Lane 3 Reset Lane	
Debug SERDES Configuration Tests PRBS Test Loopback Test	Lane 0 status: RxPLL TxPLL Test Type: PCS Far End PMA Rx to Tx Loopback Near End Serial Loopback (On Die)	Stop

Figure 47 · Debug SERDES - Loopback Test

#### **Running Loopback Tests in Demo Mode**

You can run Loopback tests in demo mode. The SERDES demo mode is provided to demonstrate the GUI features of SERDES. All channels are enabled. Properly working channels and channels with connectivity issues are shown so you can see the available GUI options. See the following example figure.





# Debug SERDES – PRBS Test

PRBS data stream patterns are generated and checked by the internal SERDES block. These are used to self-test signal integrity of the device. You can switch the device through several predefined patterns.

View Loopback Test settings in the Debug SERDES - Loopback Test topic.

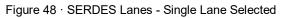
**SERDES Block** identifies which SERDES block you are configuring. Use the drop-down menu to select from the list of SERDES blocks in your design.

#### **SERDES Lanes**

Check the box or boxes to select the lane(s) on which to run the PRBS test. Then select the Lane Status, test type, and pattern for each lane you have selected. Lane mode indicates the programmed mode on a SERDES lane as defined by the SERDES system register. See the examples below.



Debug SERDES					Ŀ	8
	SERDES Lanes: 📝	SERDES Block: SER	set Selected Lanes			
ebug SERDES Configuration Tests PRBS Test Loopback Test	Lane 0 Status: Lane Number Lane 0	Near End Serial Loopb Cumulative Error Count 0	PRBS7   RxPLL  Bit Error Rate NA	Reset Error Count	Lock to data	•
						Start Stop
Help						Close



	SERDES Lanes: 🔽	Lane 0 🗹 Lane 1 📝 La	ne 2 🕅 Lane	3 Re	set Selected	Lanes			
ebug SERDES Configuration	Lane 0 Status:	Near End Serial Loopb	ack (On-Die)	•	PRBS7	RXPLL	TXPLL	Lock to data	•
PRBS Test	Lane 1 Status:	Near End Serial Loopb	ack (On-Die)	•	PRBS7	RXPLL	TxPLL	Lock to data	•
Loopback Test	Lane 2 Status:	Near End Serial Loopb	ack (On-Die)	•	PRBS7	RXPLL	TxPLL	Lock to data	•
	Lane Number	Cumulative Error Count	Data Rate		Bit Error P	ate	Reset Error Count		
	Lane 0	0		Gbps	NA		<b>E</b>		
	Lane 1	0		Gbps	NA				
	Lane 2	0		Gbps	NA				
									Stop
Help									Close

Figure 49 · SERDES Lanes - Multiple Lanes Selected



#### **Test Type**

**Near End Serial Loopback (On-Die)** enables a self-test of the device. The serial data stream is sent internally from the SERDES TX output and folded back onto the SERDES RX input.

**Serial Data (Off-Die)** is the normal system operation where the data stream is sent off chip from the TX output and must be connected to the RX input via a cable or other type of electrical interconnection.

If more than one SERDES Lane has been selected, the test type can be selected per lane. In the following example, Near End Serial Loopback (On-Die) has been selected for Lane 0 and Lane 3, and Serial Data (Off-Die) has been selected for Lane 1 and Lane 2.

Debug SERDES									3
	SERDES Lanes: 👿 L	SERDES Block: SER		3 Re	set Selecte	d Lanes			
ebug SERDES	1.00								
Configuration	Lane 0 Status:	Near End Serial Loopb	ack (On-Die)	*	PRBS7	* RxPLL	TXPLL	Lock to data	•
PRBS Test	Lane 1 Status:	Serial Data (Off-Die)		-	PR857	- RxPLL	TXPLL	Lock to data	•
Loopback Test	Lane 2 Status:	Serial Data (Off-Die)		*	PRBS7	* RXPLL	TXPLL	Lock to data	•
	Lane 3 Status:	Near End Serial Loopb	ack (On-Die)	*	PRBS7	RxPLL	TXPLL	Lock to data	•
	Lane Number	Cumulative Error Count	Data Rate		Bit Error	Rate	Reset Error Count		
	Lane 0	5		Gbps	NA				
	Lane 1	0		Gbps	NA				
	Lane 2	0		Gbps	NA				
	Lane 3	0		Gbps	NA				Start
								_	
									Stop
Help								F	Close

Figure 50 · Test Type Example

#### Pattern

The SERDESIF includes an embedded test pattern generator and checker used to perform serial diagnostics on the serial channel, as shown in the table below. If more than one lane is selected, the PRBS pattern can be selected per lane.

Pattern	Туре
PRBS7	Pseudo-Random data stream of 2^7 polynomial sequences
PRBS11	Pseudo-Random data stream of 2^11 polynomial sequences
PRBS23	Pseudo-Random data stream of 2^23 polynomial sequences
PRBS31	Pseudo-Random data stream of 2^31 polynomial sequences



#### **Cumulative Error Count**

Lists the number of cumulative errors after running your PRBS test. To reset the error count to zero, select the lane(s) and click **Reset**. By default, Cumulative Error Count = 0, the Data Rate text box is blank, and Bit Error Rate = NA.

ebug SERDES Configuration	Lane 0 Status:	Near End Serial Loopb	ack (On-Die)	*	PRBS7 +	RXPLL	TXPLL	Lock to data	
Tests     PRBS Test	Lane 1 Status:	Serial Data (Off-Die)		w)	PRBS11 *	RxPLL	TXPLL	Lock to data	•
Loopback Test	Lane 2 Status:	Serial Data (Off-Die)		*	PRBS23 *	RXPLL	TXPLL	Lock to data	•
	Lane 3 Status:	Near End Serial Loopb	ack (On-Die)	*	PRBS31 *	RxPLL	TXPLL	Lock to data	•
	Lane Number	Cumulative Error Count	Data Rate		Bit Error Ra	te 🗍	Reset Error Count		
	Lane 0	0	1	Gbps	2.00e-10	100	8	100	
	Lane 1	0	2	Gbps	1.00e-10		63		
	Lane 2	0	3	Gbps	6.67e-11		8		
	Lane 3	0	4	Gbps	5.00e-11		13		Start
								_	
									Stop

Figure 51 · Debug SERDES - PRBS Test

Note: If the design uses SERDES PCIe, PRBS7 is the only available option for PRBS tests.

#### **Bit Error Rate**

The Bit Error Rate is displayed per lane. If you did not specify a Data Rate, the Bit Error Rate displays the default NA. When the PRBS test is started, the Cumulative Error Count and Bit Error Rate are updated every second. You can select specific lanes and click **Reset Error Count** to clear the Cumulative Error Count and Bit Error Rate fields of the selected lanes.

In the example below, the Bit Error Rate is displayed for all lanes.



	Lane 0 📝 Lane 1 📝 Lar							
Lage 0 Status:	Near End Serial Looph	ack (On-Die)	-	PRRS7 +	RVPL	Typu	Lock to data	
			1					
Lane 1 Status:	Serial Data (Off-Die)		*	PRES11 *	RMPLL	TXPLL	Lock to data	•
Lane 2 Status:	Serial Data (Off-Die)		*	PRBS23 *	RXPLL	TxPLL 🔵	Lock to data	•
Lane 3 Status:	Near End Serial Loopb	ack (On-Die)	•	PRBS31 *	RXPLL	TXPLL	Lock to data	•
Lane Number	Cumulative Error Count	Data Rate		Bit Error Rat	e 🕞	Reset Error Count		
Lane 0	0	1 9	bos	2.00e-10	-	27	-	
Lane 1	0			1.00e-10				
Lane 2	0		1.5					
Carle D		- C.	op o	Divic II				Start
								Stop
								- ap
	Lane 3 Status:	Lane 1 Status: Serial Data (Off-Oie) Lane 2 Status: Serial Data (Off-Oie) Lane 3 Status: Near End Serial Loopb Lane Number Cumulative Brror Count Lane 0 0 Lane 1 0 Lane 2 0	Lane 1 Status: Serial Data (Off-Die) Lane 2 Status: Serial Data (Off-Die) Lane 3 Status: Niear End Serial Loopback (On-Die) Lane Number Cumulative Error Count Data Rate Lane 0 0 1 0 0 Lane 1 0 2 0	Lane 1 Status:     Serial Data (Off-Die)       Lane 2 Status:     Serial Data (Off-Die)       Lane 3 Status:     Flear End Serial Loopback (On-Die)       Lane Number     Cumulative Error Count     Data Rate       Lane 0     0     1     Gops       Lane 1     0     2     Gops       Lane 2     0     3     Gops	Lane 1 Status:         Serial Data (Off-Die)         *         PRBS11         *           Lane 2 Status:         Serial Data (Off-Die)         *         PRBS23         *           Lane 3 Status:         Hear End Serial Loopback (On-Die)         *         PRBS31         *           Lane 1 Status:         Hear End Serial Loopback (On-Die)         *         PRBS31         *           Lane 0         0         1         Gbps         2.00e-10           Lane 1         0         2         Gbps         1.00e-10           Lane 2         0         3         Gbps         6.67e-11	Lane 1 Status:       Serial Data (Off-Die) <ul> <li>PRBS 11</li> <li>RxPLL</li> </ul> Lane 2 Status:       Serial Data (Off-Die) <ul> <li>PRBS 23</li> <li>RxPLL</li> <li>Lane 3 Status:</li> <li>Near End Serial Loopback (On-Die)</li> <li>PRBS 31</li> <li>RxPLL</li> </ul> Lane 1 Status:       Near End Serial Loopback (On-Die) <ul> <li>PRBS 31</li> <li>RxPLL</li> </ul> Lane Number       Cumulative Error Count       Data Rate       Bit Error Rate         Lane 0       0 <ul> <li>Gbps</li> <li>2.00e-10</li> <li>Lane 2</li> <li>Gbps</li> <li>6.67e-11</li> </ul>	Lane 1 Status:       Serial Data (Off-Die) <ul> <li>PRES11</li> <li>RxPLL</li> <li>TxPLL</li> <li>Lane 2 Status:</li> <li>Serial Data (Off-Die)</li> <li>PRES23</li> <li>RxPLL</li> <li>TxPLL</li> <li>Lane 3 Status:</li> <li>Near End Serial Loopback (Dn-Die)</li> <li>PRES31</li> <li>RxPLL</li> <li>TxPLL</li> <li>Lane 3 Status:</li> <li>Near End Serial Loopback (Dn-Die)</li> <li>PRES31</li> <li>RxPLL</li> <li>TxPLL</li> <li>Lane 0</li> <li>0</li> <li>1</li> <li>Gbps</li> <li>2.00e-10</li> <li>Lane 1</li> <li>0</li> <li>2</li> <li>Gbps</li> <li>1.00e-10</li> <li>Lane 2</li> <li>3</li> <li>Gbps</li> <li>6.67e-11</li> <li></li> <!--</td--><td>Lane 1 Status:       Serial Data (Off-Die)       PRBS11       RxPLL       TxPLL       Lock to data         Lane 2 Status:       Serial Data (Off-Die)       PRBS22       RxPLL       TxPLL       Lock to data         Lane 3 Status:       Near End Serial Loopback (On-Die)       PRBS31       RxPLL       TxPLL       Lock to data         Lane 3 Status:       Near End Serial Loopback (On-Die)       PRBS31       RxPLL       TxPLL       Lock to data         Lane 0       0       1       Gops       2.00e-10       Image: Commutative Error Count       Lane 1       0       2       Gops       1.00e-10       Image: Count       Lane 2       0       3       Gops       6.67e-11       Image: Count       Lane 3       0       4       Gops       5.00e-11       Image: Count       Image: Count</td></ul>	Lane 1 Status:       Serial Data (Off-Die)       PRBS11       RxPLL       TxPLL       Lock to data         Lane 2 Status:       Serial Data (Off-Die)       PRBS22       RxPLL       TxPLL       Lock to data         Lane 3 Status:       Near End Serial Loopback (On-Die)       PRBS31       RxPLL       TxPLL       Lock to data         Lane 3 Status:       Near End Serial Loopback (On-Die)       PRBS31       RxPLL       TxPLL       Lock to data         Lane 0       0       1       Gops       2.00e-10       Image: Commutative Error Count       Lane 1       0       2       Gops       1.00e-10       Image: Count       Lane 2       0       3       Gops       6.67e-11       Image: Count       Lane 3       0       4       Gops       5.00e-11       Image: Count       Image: Count

#### Figure 52 · Bit Error Rate Example - All Lanes

In the example below, Lane 1 and Lane 2 are selected and Reset Error Count is clicked.

	SERDES Lanes: 💟 L	ane 0 🔽 Lane 1 🔽 Lar	ne 2 🔽 Lane :	3 Re	set Selected La	nes			
ebug SERDES	Lane 0 Status:	and the state of t	1.0.0.0	¥	PRBS7 *			Lock to data	
Configuration Tests	Lane 1 Status:	Near End Serial Loopb Serial Data (Off-Die)	aox (On-Die)			RxPLL	TXPLL	Lock to data	
PRBS Test Loopback Test	Lane 2 Status:	Serial Data (Off-Die)			PRBS23 ¥		TXPLL	Lock to data	
	Lane 3 Status:	Near End Serial Loopb	ack (On-Die)	•	PRBS31 *	RxPLL	TXPLL	Lock to data	•
	Lane Number	Cumulative Error Count	Data Rate		Bit Error Rat	e 📑	Reset Error Count		
	Lane 0	5	1	Gbps	1.82e-11				
	Lane 1	D	2	Gbps	NA				
	Lane 2	0	3	Gbps	NA				
	Lane 3	D	4	Gbps	4.55e-12				Start
									Stop

Figure 53 · Reset Error Count Example



#### **Running PRBS Tests in Demo Mode**

You can run Multi Lane PRBS tests in demo mode. The SERDES demo mode is provided to demonstrate the GUI features of SERDES. All channels are enabled. Properly working channels and channels with connectivity issues are shown so you can see the available GUI options. See the following example figure.

SERDES Lanes:	V Lane 0 V Lane 1 V La	ne 2 🔽 Lan	e 3 Reset Selected Lanes			
Lane 0 Status:	Near End Serial Loopbac	k (On-Die)	✓ PRBS7 ✓ RxPLL●	TxPLL	Lock to data	٠
Lane 1 Status:	Near End Serial Loopbac	k (On-Die)	PRESII - RAPLL	TXPLL	Lock to data	
Lane 2 Status:	Near End Serial Loopbac	k (On-Die)	PRBS23 Y RxPLL	TXPLL	Lock to data	•
Lane 3 Status:	Near End Serial Loopbac	k (On-Die)	T PR8531 T RxPLL	TXPLL	Lock to data	•
					2	
Lane Number	Cumulative Error Count	Data Rate	Bit Error Rate	Reset Error Count		
Lane 0	0	2	Gbps 1.67e-10	Г		
Lane 1	NA	3	Gbps NA	Г		
Lane 2	NA	4	Gbps NA	Г		
Lane 3	0	5	Gbps 6.67e-11	Г		Start
- 10 m						Stop
	Lane 0 Status: Lane 1 Status: Lane 2 Status: Lane 3 Status: Lane 3 Status: Lane 0 Lane 1 Lane 1 Lane 2 Lane 3	Lane 0 Status: Near End Serial Loopbac Lane 1 Status: Near End Serial Loopbac Lane 2 Status: Near End Serial Loopbac Lane 3 Status: Near End Serial Loopbac Lane 3 Status: Near End Serial Loopbac Lane 0 0 Lane 1 NA Lane 2 NA Lane 3 0	Lane 0 Status:       Near End Serial Loopback (On-Die)         Lane 1 Status:       Near End Serial Loopback (On-Die)         Lane 2 Status:       Near End Serial Loopback (On-Die)         Lane 3 Status:       Near End Serial Loopback (On-Die)         Lane 1 NA       3         Lane 2 NA       4         Lane 3 0       5	Lane 1 Status:       Near End Serial Loopback (On-Die)       PRBS11       RxPLL         Lane 2 Status:       Near End Serial Loopback (On-Die)       PRBS23       RxPLL         Lane 3 Status:       Near End Serial Loopback (On-Die)       PRBS31       RxPLL         Lane 3 Status:       Near End Serial Loopback (On-Die)       PRBS31       RxPLL         Lane 3 Status:       Near End Serial Loopback (On-Die)       PRBS31       RxPLL         Lane 0       0       Image: Data Rate       Bt Error Rate         Lane 1       NA       3       Gbps       NA         Lane 2       NA       4       Gbps       NA	Lane 0 Status:       Near End Serial Loopback (On-Die)       PRBS7       RxPLL       TxPLL         Lane 1 Status:       Near End Serial Loopback (On-Die)       PRBS11       RxPLL       TxPLL         Lane 2 Status:       Near End Serial Loopback (On-Die)       PRBS23       RxPLL       TxPLL         Lane 2 Status:       Near End Serial Loopback (On-Die)       PRBS23       RxPLL       TxPLL         Lane 3 Status:       Piesr End Serial Loopback (On-Die)       PRBS33       RxPLL       TxPLL         Lane 3 Status:       Piesr End Serial Loopback (On-Die)       PRB533       RxPLL       TxPLL         Lane 1 Status:       Piesr End Serial Loopback (On-Die)       PRB533       RxPLL       TxPLL         Lane 1 NA       3       Gbps       NA       I         Lane 2 NA       4       Gbps       NA       I         Lane 3       0       5       Gbps       6.67e-11       I	Lane 0 Status:       Near End Serial Loopback (On-Die)       PRBS7       RxPLL       TxPLL       Lock to data         Lane 1 Status:       Near End Serial Loopback (On-Die)       PRBS11       RxPLL       TxPLL       Lock to data         Lane 2 Status:       Near End Serial Loopback (On-Die)       PRBS23       RxPLL       TxPLL       Lock to data         Lane 2 Status:       Near End Serial Loopback (On-Die)       PRBS23       RxPLL       TxPLL       Lock to data         Lane 3 Status:       Piesr End Serial Loopback (On-Die)       PRBS31       RxPLL       TxPLL       Lock to data         Lane 3 Status:       Piesr End Serial Loopback (On-Die)       PRBS31       RxPLL       TxPLL       Lock to data         Lane 1       NA       3       Gbps       NA       T         Lane 2       NA       3       Gbps       NA       T         Lane 3       0       5       Gbps       6.47e-11       T

#### Notes:

The formula for calculating the BER is as follows:

BER = (#bit errors+1)/#bits sent

#bits sent = Elapsed time/bit period

When clicked on Start:

- The BER is updated every second for the entered data rate and errors observed.
- If no data rate is entered by the user, the BER is set to the default NA.

When clicked on Stop:

• The BER resets to default.

When clicked on Reset:

- The BER resets to default.
- If no test is in progress, the BER remains in the default value.
- If the PRBS test is in progress, the BER calculation restarts.



# Debug SERDES – PHY Reset

SERDES PMA registers (for example, TX\_AMP\_RATIO) modified using a TCL script from the Configuration tab require a soft reset for the new values to be updated. Lane Reset for individual lanes achieves this functionality. Depending on the SERDES lanes used in the design, the corresponding Lane Reset buttons are enabled.

#### Lane Reset Behavior for SERDES Protocols Used in the Design

- EPCS: Reset is independent for individual lanes. Reset to Lane X (where X = 0,1,2,3) resets the Xth lane.
- PCIe: Reset to Lane X (where X = 0,1,2,3) resets all lanes present in the PCIe link and PCIe controller.

For more information about soft reset, refer to the <u>SmartFusion2 and IGLOO2 High Speed Serial Interfaces User</u> <u>Guide</u>.



# **SmartDebug Tcl Support**

Refer to the <u>SmartFusion2, IGLOO2, RTG4 Tcl Commands Reference Guide</u> for information about the Tcl commands supported by SmartDebug.



# **Frequently Asked Questions**

# Embedded Flash Memory (NVM) - Failure when Programming/Verifying

If the Embedded Flash Memory failed verification when executing the PROGRAM\_NVM, VERIFY\_NVM or PROGRAM\_NVM\_ACTIVE\_ARRAY action, the failing page may be <u>corrupted</u>. To confirm and address this issue:

- 1. In the Inspect Device window click View Flash Memory Content.
- 2. Select the Flash Memory block and client (or page range) to retrieve from the device.
- 3. Click **Read from Device**; the retrieved data appears in the lower part of the window.
- 4. Click View Detailed Status.

Note: Note: You can use the check\_flash\_memory and read\_flash\_memory Tcl commands to perform diagnostics similar to the commands outlined above.

5. To reset the corrupted NVM pages, either re-program the pages with your original data or 'zero-out' the pages by using the Tcl command <u>recover\_flash\_memory</u>.

If the Embedded Flash Memory failed verification when executing a VERIFY\_NVM or VERIFY\_NVM\_ACTIVE\_ARRAY action, the failure may be due to the change of content in your design. To confirm this, repeat steps 1-3 above.

Note: NVM corruption is still possible when writing from user design. Check NVM status for confirmation.

# Analog System Not Working as Expected

If the Analog System is not working correctly, it may be due the following:

- 1. System supply issue. To troubleshoot:
- Physically verify that all the supplies are properly connected to the device and they are at the proper level. Then confirm by running the Device Status.
- Physically verify that the relevant channels are correctly connected to the device.
- 2. Analog system is not properly configured. You can confirm this by examining the Analog System.

# ADC Not Sampling the Correct Value

If the ADC is sampling all zero values then the wrong analog pin may be connected to the system, or the analog pin is disconnected. If that is not the case and the ADC is not sampling the correct value, it may be due to the following:

- 1. System supply issues Run the device status to confirm.
- Analog system is not configured at all To confirm, <u>read out the ACM configuration</u> and verify if the ACM content is all zero.
- 3. Analog system is not configured correctly To confirm, <u>read out the ACM configuration</u> and verify that the configuration is as expected.

Once analog block configuration has been confirmed, you can use the <u>sample analog channel</u> Tcl command for debug sampling of the analog channel with user-supplied sampling parameters.

If you have access to your Analog System Builder settings project (<Libero IDE project>/Smartgen/AnalogBlock), you may use the <u>compare function provided by the tool</u>.



# How do I unlock the device security so I can debug?

You must provide the PDB file with a User Pass Key in order to unlock the device and continue debugging. If you do not have a PDB with User Pass Key, you can <u>create a PDB file in FlashPro</u> (if you know the Pass Key value).

# How do I export a report?

You can export three reports from the SmartDebug GUI: Device Status, Client Detailed Status from the NVM, or the Compare Client Content report from the NVM. Each of those reports can be saved and printed.

For more information about Tcl commands supported by SmartDebug, see SmartDebug Tcl Commands.

# How do I generate diagnostic reports for my target device?

A set of diagnostic reports can be generated for your target device depending on which silicon feature you are debugging. A set of Tcl commands are available to export those reports. The following is a summary of those Tcl commands based on the silicon features.

When using the –file parameter, ensure that you use a different file name for each command so you do not overwrite the report content. If you do not specify the –file option in the Tcl, the output results will be directed to the FlashPro log window.

#### For the overall device:

read device status

read id code

#### For FlashROM:

compare flashrom client
read flashrom

#### For Embedded Flash Memory (NVM):

compare memory client
check flash memory
read flash memory

#### For Analog Block:

read analog block config compare analog config sample analog channel

To execute the Tcl command, from the File menu choose Run Script.

## How do I monitor a static or pseudo-static signal?

To monitor a static or pseudo-static signal:

- 1. Add the signal to the Active Probes tab.
- 2. Select the signal in the Active Probes tab, right-click, and choose Poll....

#### SmartDebug User Guide



Active Probes Selection	8 ×	FPGA Array debug data			
Hierarchical View Netlist View		Live Probes Active Probes Mer	mory Blocks Probe 1	Insertion	
Filter:	Search	◆ - ◆ ◆ Save	Load	Delete	Delete All
Instance(s):	Add	Name	Туре	Read Value	Write Value
	100	Shift_Reg_0/shft_reg[13:0]	DFF	14'h0001	14h
D_FF_0     A      Primitives	<u>^</u>	D_FF_0/q_0:D_FF_0/q:Q	DFF	n	<u> </u>
				Read	
▲ I Shift_Reg_0				Delete	
4 🎩 Primitives					
4 🐌 shft_reg				Poll	
shft_reg[0] shft_reg[1]				Create Group	
sht_reg[1]				The second second second second	
shft_reg[3]					
shft_reg[4]					
shft_reg[5]					
shft_reg[6]					
shft_reg[7]		Read Active Probes	Save Active Probes' Da	ta Write Act	ive Probes
shft_reg[8]	· · · ·				

3. In the Pseudo-static Signal Polling dialog box, choose a value in Polling Setup and click Start Polling.

gnal: D_FF_0/q_0:D_FF_0/q:Q		
Polling Setup		
Poll for 0	Poll for 1	
	nce per second. It should be used for pseudo-static signals the seudo-static signal polling, click the Help button.	hat do not change frequently.
	Start Polling Stop Polling	
Help	Start Polling Stop Polling	Close
Help	Start Polling Stop Polling	Close

# How do I force a signal to a new value?

To force a signal to a new value:

- 1. In the SmartDebug window, click Debug FPGA Array.
- 2. Click the Active Probes tab.
- 3. Select the signal from the selection panel and add it to Active Probes tab.



Active Probes Selection	8 ×	FPGA Array debug data
Hierarchical View Netlist View		Live Probes Active Probes Memory Blocks Probe Insertion
ilter:	Search	+ - + Save Load Delete All
Net(s):	Add	Name Type Read Value Write Value
B_DOUT_1_(cfs:0)     B_DOUT_2_(cfs:0)     B_DOUT_2_(cfs:0)     DFN1_0_QOPFN1_0:Q     DFN1_1_2QDFN1_0:Q     DFN1_1_2QDFN1_0:Q     DFN1_1_2QDFN1_0:Q     DFN1_1_2QDFN1_0;Q     DFN1_0_0(cfs:0)     Count_6_0_0(fs:0)     Count_6_2_0_0(fs:0)     count_7_0_0(fs:0)     count_7_0_0(fs:0)		
< [	•	Read Active Probes Save Active Probes' Data Write Active Probes

- 1. Click Read Active Probe to read the value.
- 2. In the Write Value column, enter the value to write to the signal and then click Write Active Probes.

Active Probes Selection		₽×	FPGA A	rray deb	ug data					
Hierarchical View Netlist View			_	Probes		e Probes	Memory	y Blocks Probe In	nsertion	
Filter:	Search		+	-	+ •	S	ave	Load	Delete	Delete All
(s):	Add		Name			Туре	Read Value	Write Value		
	Huu			DFN1_0_Q:DFM		N1_0:Q		DFF	1	0 -
Name	Туре	*		B_DOUT_c[5:0]		RAM64x18	6'h0E	6'h9		
<ul> <li>B_DOUT_2_(7:0)</li> <li>B_DOUT_c(5:0)</li> <li>DFN1_0_Q:DFN1_0:Q</li> <li>DFN1_1_2:DFN1_1:Q</li> </ul>	RAM64x18 RAM64x18 DFF DFF									
<ul> <li>□ URAM_0\vd_URAM_0_URAM_RCC0[A_ADDR_net[9:0]</li> <li>□ URAM_0\vd_URAM_0_URAM_RCC0[B_ADDR_net[9:0]</li> <li>□ count_6_0_0[5:0]</li> <li>□ count_7_0_0[6:0]</li> <li>□ count_7_0_0[6:0]</li> <li>□ count_7_0_0[6:0]</li> </ul>						ive Probes		/e Active Probes' Dat		tive Probes

# How do I count the transitions on a signal?

If FHB IP is auto-instantiated in the design, you can use the Event Counter in the **Live Probes** tab to count the transitions on a signal.

To count the transitions on a signal:

- 1. Assign the desired signal to Live Probe Channel A.
- 2. Click the Event Counter tab and check the Activate Event Counter checkbox.



/Active Probes Selection & X	FPGA Array debug data						
Hierarchical View Netlist View	Live Probes Active Probes Memory Blocks Probe Insertion						
Filter: Search	Delete	Delete All					
Instance(s): Add	Name	Туре					
Primitives	A_DOUT_0_c[8]:URAM_3\/sd_URAM_3_URAM_R0C4/INST_RAM64x18_IP:A_DOUT	0] RAM64x18					
URAM_0\ URAM_1\	A_DOUT_0_c[7]:URAM_3\/sd_URAM_3_URAM_R0C3/INST_RAM64x18_IP:A_DOUT	1] RAM64x18					
URAM_2\ URAM_3\	A_DOUT_0_c[6]:URAM_3\/sd_URAM_3_URAM_R0C3/INST_RAM64x18_IP:A_DOUT	0] RAM64x18					
> 🖀 count 6 0\	A_DOUT_0_c[5]:URAM_3\/sd_URAM_3_URAM_R0C2/INST_RAM64x18_IP:A_DOUT	1] RAM64x18					
Event Counter/Frequency Monitor	A_DOUT_0_c[4];URAM_3\sd_URAM_3_URAM_R0C2/INST_RAM64x18_IP;A_DOUT	0] RAM64x18					
	A_DOUT_0_c[3]:URAM_3\/sd_URAM_3_URAM_ROC1/INST_RAM64x18_IP:A_DOUT	1] RAM64x18					
Activate Event Counter     Reset	A_DOUT_0_c[2];URAM_3\/sd_URAM_3_URAM_ROC1/INST_RAM64x18_IP;A_DOUT	0] RAM64x18					
Edge Selected: Rising Time elapsed (s):	A_DOUT_0_c[1]:URAM_3\/sd_URAM_3_URAM_R0C0/INST_RAM64x18_IP:A_DOUT	1] RAM64x18					
Time elapsed (s):	A DOUT 0 c[0]:URAM 3\/sd URAM 3 URAM ROC0/INST RAM64x18 IP:A DOUT	0] RAM64x18					
Signal : A_DOUT_0_c(8):URAM_3\/sd_URAM_3_URAM	Assign to Channel A     -> A_DOUT_0_c(8):URAM_3\/sd_URAM_3_URAM_R0C4     Assign to Channel B     ->	► /INST_RAM64x18_IP					
Event Counter Frequency Monitor User Clock Frequencies	Unassign Channels						

#### See Also

"Event Counter" on page 38

## How do I monitor or measure a clock?

You can monitor a clock signal from the **Live Probe** tab when the design is synthesized and compiled with FHB Auto Instantiation turned on in Project Settings dialog box.

In the Live Probe tab, SmartDebug allows you to:

1. Measure all the FABCCC GL clocks by clicking the **User Clock Frequencies** tab, as shown in the figure below.

	ē ×	FPGA Array debug data	
ierarchical View Netlist Vi	ew	Live Probes Active Probes Memory Blocks Probe Insertion	
lter:	Search	Delete	Delete All
stance(s):	Add	Name	Туре
Primitives		A_DOUT_0_c[8]:URAM_3\/sd_URAM_3_URAM_R0C4/INST_RAM64x18_IP:A_DOUT[0]	RAM64x18
> 1 URAM_0\ > 1 URAM_1\	=	A_DOUT_0_c[7]:URAM_3\/sd_URAM_3_URAM_R0C3/INST_RAM64x18_IP:A_DOUT[1]	RAM64x18
> == URAM_2\ > == URAM_3\		A_DOUT_0_c[6]:URAM_3\/sd_URAM_3_URAM_R0C3/INST_RAM64x18_IP:A_DOUT[0]	RAM64x18
= count 6 0\	•	A_DOUT_0_c[5]:URAM_3\/sd_URAM_3_URAM_R0C2/INST_RAM64x18_IP:A_DOUT[1]	RAM64x18
Event Counter/Frequence	cy Monitor	A_DOUT_0_c[4]:URAM_3\/sd_URAM_3_URAM_R0C2/INST_RAM64x18_IP:A_DOUT[0]	RAM64x18
		A_DOUT_0_c[3]:URAM_3\/sd_URAM_3_URAM_R0C1/INST_RAM64x18_IP:A_DOUT[1]	RAM64x18
User Clocks	Frequency (MHz)	A_DOUT_0_c[2]:URAM_3\/sd_URAM_3_URAM_R0C1/INST_RAM64x18_IP:A_DOUT[0]	RAM64x18
FCCC_0_GL0	~24.5	A DOUT 0 c[1]:URAM 3\/sd URAM 3 URAM R0C0/INST RAM64x18 IP:A DOUT[1]	RAM64x18
2 FCCC_0_GL1	~48.7	A_DOUT_0_c[0]:URAM_3Vsd_URAM_3_URAM_R0C0/INST_RAM64x18_IP:A_DOUT[0]	RAM64x18
3 FCCC_0_GL2	~97.4	HT0001_0_1010000-10400_010000-1000041001_00004440_0140101101	101101110
4 FCCC_0_GL3	~194.6		
		<	
		Assign to Channel A -> A_DOUT_0_c[8]:URAM_3\/sd_URAM_3_URAM_R0C4/INS	T DAM64v18 TD
			1_1041101X 10_1F
2		Assign to Channel B ->	
		Unassign Channels	

- 2. Monitor frequencies of any probe points by:
  - a. Assigning the desired signal to Live Probe Channel A.
  - b. Selecting the **Frequency Monitor** tab as shown in the following figure and checking the Activate Frequency Meter checkbox.

Microsemi

а 🔨 Міскоснір company



₽ /e/Active Probes Selection	ē×	FPGA Array debu	n data				
Hierarchical View Netlist View		Live Probes	Active Probes	Memory Blocks	Probe Inser	tion	
Filter:	Search					Delete	Delete All
Instance(s):	Add			Name			Туре
Primitives		A_DOUT_0_c	[8]:URAM_3\/sd_	URAM_3_URAM_RO	4/INST_RAM6	4x18_IP:A_DOUT[0]	RAM64x18
URAM_0\ URAM_1\	E	A_DOUT_0_c	[7]:URAM_3\/sd_	URAM_3_URAM_ROO	3/INST_RAM6	4x18_IP:A_DOUT[1]	RAM64x18
URAM_2\ URAM_3\		A_DOUT_0_c	[6]:URAM_3\/sd_	URAM_3_URAM_ROO	3/INST_RAM6	4x18_IP:A_DOUT[0]	RAM64x18
D = count 6 0\	•	A_DOUT_0_c	[5]:URAM_3\/sd_	URAM_3_URAM_ROO	2/INST_RAM6	4x18_IP:A_DOUT[1]	RAM64x18
Event Counter/Frequency Monitor		A_DOUT_0_c	[4]:URAM_3\/sd_	URAM_3_URAM_RO	2/INST_RAM6	4x18_IP:A_DOUT[0]	RAM64x18
	]	A_DOUT_0_c	[3]:URAM_3\/sd_	URAM_3_URAM_ROO	1/INST_RAM6	4x18_IP:A_DOUT[1]	RAM64x18
🕢 Activate Frequency Meter 🛛 🔵	RESET	A_DOUT_0_c	[2]:URAM_3\/sd_	URAM_3_URAM_RO	1/INST_RAM6	4x18_IP:A_DOUT[0]	RAM64x18
Monitor time (s): 5		A_DOUT_0_c	[1]:URAM_3\/sd_	URAM_3_URAM_RO	CO/INST_RAM6	4x18_IP:A_DOUT[1]	RAM64x18
Frequency (MHz): 0 Signal : A_DOUT_0_c[8]:URAM_3\/sd_U	DAM & LOAN DOCUMET DA	A DOUT 0 cl	0]:URAM 3\/sd	URAM 3 URAM ROO	0/INST RAM6	4x18_IP:A_DOUT[0]	RAM64x18
		Assign to Ch	annel A ->	A_DOUT_0_c[8]:UR	] AM_3\/sd_URA	M_3_URAM_ROC4/INS	► T_RAM64x18_IP
Event Counter Frequency Monitor	User Clock Frequencies	Assign to Ch Unassign Ch					

# How do I perform simple PRBS and loopback tests?

You can perform PRBS and loopback tests using the Debug SERDES option in SmartDebug.

To perform a PRBS test, in the Debug SERDES dialog box, select **PRBS Test** to run a PRBS test on-die or offdie For more information, see "Debug SERDES – PRBS Test" on page 59.

To perform a PRBS test, in the Debug SERDES dialog box, select PRBS Test to run a PRBS test on-die or offdie. For more information, see "Debug SERDES – PRBS Test" on page 59.

To perform a loopback test, in the Debug SERDES dialog box, select **Loopback Test** to run a near end serial loopback /far end PMA Rx to Tx loopback test. For more information, see "Debug SERDES – Loopback Test" on page 57.

## How do I read LSRAM or USRAM content?

To read RAM content:

- 1. In the Debug FPGA Array dialog box, click the Memory Blocks tab.
- 2. Select the memory block to be read from the selection panel on the left of the window.



/ Blocks Selection	8×	FPGA Array debug data
n	Search	Live Probes Active Probes Memory Blocks Probe Insertion
nory Blocks:	Select	User Design Memory Blodc: Data Width:
itance Tree		Port Used:
Brans Add		
* * mem_mem_0_2     * * * mem_mem_0_3		
		L
		Read Block Save Block Data Write Block
		had 1

An "L" in the icon next to the block name indicates that it is a logical block, and a "P" in the icon indicates that it is a physical block. A logical block displays three fields in the Memory Blocks tab: User Design Memory Blocks, Data Width, and Port Used. A physical block displays two fields in the Memory Blocks tab: User Design Memory Block and Data Width.

- 3. Add the block in one of the following ways:
  - a. Click Select.
  - b. Right-click and choose Add.
  - c. Drag the block to the Memory Blocks tab.
- 4. Click Read Block to read the content of the block.

emory Blocks Selection	₽×	FPGA Arr	ay debu	g data														
Filter: Sea	dh	Live Pr	obes	Active	Probes	Me	mory Bl	ocks	Probe	Insertio	n							
Memory Blocks: Sele	ct	User D Data W	esign Me /idth:	mory Bl	ock:	Fabric_ 18-bit	Logic_0	/U3/F_0	_F0_U1	L								
Instance Tree	^	Port Us	ed:			Port A	8	-										
Fabric_Logic_0 U3 F_0_F0_U1			0	1	2	3	4	5	6	7	8	9	A	В	с	D	E	F
		0000	00A83	08809	09008	14500	00010	00381	12028	00040	12080	04000	20214	02000	11080	20040	1C220	0A020
		0010	02700	04451	04001	08000	05000	32500	00120	00000	00080	00420	04019	1C800	00052	00106	00C22	10058
<ul> <li>ramtmp_ramtmp_0_0</li> <li>Primitives</li> </ul>		0020	10400	00010	10000	14044	1C040	0810E	39425	0D990	10C14	00004	04001	10000	00100	00042	20100	08002
<ul> <li>INST_RAM64x18_IP</li> <li>F_11_F1_U2</li> </ul>		0030	000 1B	00000	20808	0008A	00 1E0	28100	02883	00770	10020	04000	00000	00200	20004	22400	04006	0A090
<ul> <li>4 12: ramtmp_ramtmp_0_0</li> <li>4 12: Primitives</li> </ul>																		
INST_RAM64x18_I F_12_F1_U2	·						Read B	lock	Save	Block D	ata	Wri	te Block					
Image: A second seco	-																	

#### See Also

"Memory Blocks (SmartFusion2, IGLOO2, RTG4)" on page 30"Memory Blocks (SmartFusion2, IGLOO2, RTG4)" on page 30

### How do I change the content of LSRAM or USRAM?

To change the content of LSRAM or USRAM:

1. In the SmartDebug window, click **Debug FPGA Array**.



- 2. Click the Memory Blocks tab.
- 3. Select the memory block from the selection panel on the left of the window.

Filter: Search Live	
E S ranvers B INST RANK20 JP E S mem_mem_0.1 E S mem_mem_0.2 E S mem_mem_0.3 E S mem_mem_0_3	Read Block Save Block Data Write Block

An "L" in the icon next to the block name indicates that it is a logical block, and a "P" in the icon indicates that it is a physical block. A logical block displays three fields in the Memory Blocks tab: User Design Memory Blocks, Data Width, and Port Used. A physical block displays two fields in the Memory Blocks tab: User Design Memory Block and Data Width.

- 4. Add the memory block in one of the following ways:
  - a. Click Select.
  - b. Right-click and choose Add.
  - c. Drag the block to the **Memory Blocks** tab.
- 5. Click Read Block. The memory content matrix is displayed.
- 6. Select the memory cell value that you want to change and update the value.
- 7. Click **Write Block** to write to the device.

emory Blocks Selection	8	×	FPGA Arra	ay debug	data														
Filter:	Search		Live Pro	obes	Active	Probes	Men	nory Blo	cks	Probe I	nsertion								
Memory Blocks: Select		User Design Memory Block: Fabric_Logic_0/U3/F_12, Data Width: 18-bit							2_F1_U2	2									
Instance Tree		^	Port Us	ed:			Port A		-										
Fabric_Logic_0																			
▷ 💶 U2	1			0	1	2	3	4	5	6	7	8	9	A	в	с	D	E	F
✓ I U3 ▷ I F_0_F0_U1						-					-								-
F 10 F1 U2			0000	00083	3FFFF	00102	00088	01200	00824	00004	00304	00200	00E00	0006A	20001	00060	00050	00300	00000
▶ 1 F 11 F1 U2			0010			20002													
4 🎩 F_12 F1 U2			0010	00000	20410	20002	02101	00080	08016	02000	00200	UUUAU	00002	08000	10020	05004	00018	20008	08300
⊿ I ramtmp_ramtm	np_0_0		0020	00200	00000	00000	00084	00080	02408	00001	02080	20000	00000	20000	00005	02000	02012	00C01	00454
A IP Primitives			0020								02000	20000		20000			OLO IL		
	RAM64x18_IP		0030	02400	10001	00001	04000	00400	00002	01201	00004	00020	01C00	02040	10008	07242	18102	24041	02044
F_13_F1_U2																			1000100
F_14_F1_U2 F_15_F1_U2																			
F_15_F1_02 F_16_F1_U2							_						_		_				
F_17_F1_U2								Read B	ock	Save	Block Da	ta	Writ	te Block					
> 4 F 18 F1 U2																			
5 JE E 19 E1 U2	1	-	-																

#### See Also

"Memory Blocks (SmartFusion2, IGLOO2, RTG4)" on page 30"Memory Blocks (SmartFusion2, IGLOO2, RTG4)" on page 30



# How do I read the health check of the SERDES?

You can read the SERDES health check using the following Debug SERDES options:

 Review the Configuration Report, which returns PMA Ready, TxPLL status, and RxPLL status. For SERDES to function correctly, PMA ready should be true, and TxPLL and RxPLL status should be locked. The Configuration Report can be found in the Debug SERDES dialog box under Configuration. See Debug SERDES (SmartFusion2, IGLOO2, and RTG4).

Debug SERDES		Designation of the local division of the loc	2 ×
	SERDES Blods: SERDES F_0 • SERDES Lanes: © Lane 0 © Lane 1 © Lane 2 © Lane 3 Lane 0 Reset Lane 1 Reset Lane 2 Reset Lane 3 Reset		
Debug SERDES	Configuration Report:		
Configuration First PRBS Test Loopback Test	Serdes Bok SERDESIP_0: Lare SO: PMA Ready: True TrolL status: Locked RoUL status: Locked Lane mode: EPCS (custom) PMA Ready: True TrolL status: Locked Lane mode: EPCS (custom) PMA Ready: True TrolL status: Locked RoUL status: Locked RoUL status: Locked Lare 3: Lare mode: EPCS (custom) PMA Ready: True TrolL status: Locked RoUL status: Locked RoUL status: Locked RoUL status: Locked RoUL status: Locked RoUL status: Locked		<ul> <li>Refresh Report</li> <li>E</li> </ul>
	SERDES Register Read or Write: Script:		Execute
Hep			Close

2. Run the **PRBS Test**, which is a Near End Serial Loopback tests on selected lanes. This should result in 0 errors in the Cumulative Error Count column. See "Debug SERDES – PRBS Test" on page 59.

#### Where can I find files to compare my contents/settings?

#### **FlashROM**

You can compare the FlashROM content in the device with the data in the PDB file. You can find the PDB in the <Libero IDE project>/Designer/Impl directory.

#### **Embedded Flash Memory (NVM)**

You can compare the Embedded Flash Memory content in the device with the data in the PDB file. You can find the PDB in the <Libero IDE project>/Designer/Impl directory.

### What is a UFC file? What is an EFC file?

UFC is the User FlashROM Configuration file, generated by the FlashROM configurator; it contains the partition information set by the user. It also contains the user-selected data for region types with static data. However, for AUTO\_INC and READ\_FROM\_FILE, regions the UFC file contains only:

- Start value, end value, and step size for AUTO INC regions, and
- File directory for READ\_FROM\_FILE regions

EFC is the Embedded Flash Configuration file, generated by the Flash Memory Builder in the Project Manager <u>Catalog</u>; it contains the partition information and data set by the user.

Both UFC and EFC information is embedded in the PDB when you generate the PDB file.



## Is my FPGA fabric enabled?

When your FPGA fabric is programmed, you will see the following statement under Device State in the Device Status report:

FPGA Array Status: Programmed and Enabled

If the FPGA fabric is not programmed, the Device State shows:

FPGA Array Status: Not Enabled

# Is my Embedded Flash Memory (NVM) programmed?

To figure out if your NVM is programmed, read out and view the NVM content or perform verification with the PDB file.

To examine the NVM content, see the <u>FlashROM Memory Content Dialog Box</u>.

# How do I display Embedded Flash Memory (NVM) content in the Client partition?

You must load your PDB into your FlashPro project in order to view the Embedded Flash Memory content in the Client partition. To view NVM content in the client partition:

- 1. Load your PDB into your FlashPro project.
- 2. Click Inspect Device.
- 3. Click View Flash Memory Content.
- 4. Choose a block from the drop-down menu.
- 5. Select a client.
- 6. Click **Read from Device**. The Embedded Flash Memory content from the device appears in the Flash Memory dialog box.

# How do I know if I have Embedded Flash Memory (NVM) corruption?

When Embedded Flash Memory is <u>corrupted</u>, <u>checking Embedded Flash Memory</u> may return with any or all of the following page status:

- ECC1/ECC2 failure
- Page write count exceeds the 10-year retention threshold
- Page write count is invalid
- Page protection is set illegally (set when it should not be)

See the How do I interpret data in the Flash Memory (NVM) Status Report? topic for details.

If your Embedded Flash Memory is corrupted, you can recover by reprogramming with original design data. Alternatively, you can 'zero-out' the pages by using the Tcl command recover flash memory.

# Why does Embedded Flash Memory (NVM) corruption happen?

Embedded Flash Memory corruption occurs when Embedded Flash Memory programming is interrupted due to:

- Supply brownout; monitor power supplies for brownout conditions. For SmartFusion monitor the VCC\_ENVM/VCC\_ROSC voltage levels; for Fusion, monitor VCC\_NVM/VCC\_OSC.
- Reset signal is not properly tied off in your design. Check the Embedded Memory reset signal.

## How do I recover from Embedded Flash Memory corruption?

Reprogram with original design data or 'zero-out' the pages by using the Tcl command recover flash memory.



## What is a JTAG IR-Capture value?

JTAG IR-Capture value contains private and public device status values. The public status value in the value read is ISC\_DONE, which indicates if the FPGA Array is programmed and enabled.

The ISC\_DONE signal is implemented as part of IEEE 1532 specification.

## What does the ECC1/ECC2 error mean?

ECC is the Error Correction Code embedded in each Flash Memory page.

ECC1 - One bit error and correctable.

ECC2 – Two or more errors found, and not correctable.

# What happens if invalid firmware is loaded into eNVM in SmartFusion2 devices?

When invalid firmware is loaded into eNVM in SmartFusion2 devices, Cortex-M3 will not be able to boot and issues reset to MSS continuously. eNVM content using View Flash Memory content will read zeroes in SmartDebug.

To verify that your FlashROM is programmed, <u>read out and view the FlashROM content</u> or perform verification with the PDB file by selecting the <u>VERIFY</u> or <u>VERIFY\_FROM</u> action in FlashPro.

## Can I compare serialization data?

To compare the serialization data, you can read out the FlashROM content and visually check data in the serialization region. Note that a serialization region can be an AUTO\_INC or READ\_FROM\_FILE region.

For serialization data in the AUTO\_INC region, check to make sure that the data is within the specified range for that region.

For READ FROM FILE region, you can search for a match in the source data file.

#### Can I tell what security options are programmed in my device?

To determine the programmed security settings, run the Device Status option from the Inspect Device dialog and examine the Security Section in the report.

This section lists the security status of the FlashROM, FPGA Array and Flash Memory blocks.

## How do I interpret data in the Device Status report?

The Device Status Report generated from the FlashPro SmartDebug Feature contains the following sections:

- IDCode (see below)
- User Information
- Device State
- Factory Data
- Security Settings



# How do I interpret data in the Flash Memory (NVM) Status Report?

The Embedded Flash Memory (NVM) Status Report generated from the FlashPro SmartDebug feature consists of the page status of each NVM page. For example:

```
Flash Memory Content [ Page 34 to 34 ]
FlashMemory Page #34:
Status Register(HEX): 00090000
Status ECC2 check: Pass
```

#### Data ECC2 Check: Pass

```
Write Count: Pass (2304 writes)
Total number of pages with status ECC2 errors: 0
Total number of pages with data ECC2 errors: 0
Total number of pages with write count out of range: 0
FlashMemory Check PASSED for [ Page 34 to 34 ]
The 'check_flash_memory' command succeeded.
The Execute Script command succeeded.
```

#### Table 1 · Embedded Flash Memory Status Report Description

Flash Memory Status Info	Description
Status Register (HEX)	Raw page status register captured from device
Status ECC2 Check	Check for <u>ECC2 issue</u> in the page status
Data ECC2 Check	Check for <u>ECC2 issue</u> in the page data
Write Count	Check if the page-write count is within the expected range. The expected write count is greater than or equal to: 6,384 - SmartFusion devices 2,288 - Fusion devices Note: Write count, if corrupted, cannot be reset to a valid value within the customer flow;invalid write count will not prevent device from being programmed with the FlashPro tool. The write count on all good eNVM pages is set to be 2288 instead of 0 in the manufacturing flow. The starting count of the eNVM is 2288. Each time the page is programmed or erased the count increments by one. There is a Threshold that is set to 12288, which equals to 3 * 4096. Since the threshold can only be set in multiples of 4096 (2^12), to set a 10,000 limit, the Threshold is set to 12288 and the start count is set to 2288; and thus the eNVM has a 10k write cycle limit. After the write count exceeds the threshold, the STATUS bit goes to 11 when attempting to erase/program the page.



# **Device Status Report**

### IDCode

The IDCode section shows the raw IDCode read from the device. For example, in the Device Status report for an AFS600 device, you will find the following statement:

IDCode (HEX): 233261cf

The IDCode is compliant to IEEE 1149.1. The following table lists the IDCode bit assignments:

Bit Field (little endian)	Example Bit Value for AFS600 (HEX)	Description
Bit [31-28] (4 bits)	2	Silicon Revision
Bit [27-12] (16 bits)	3326	Device ID
Bit [11-0] (12 bits)	1cf	IEEE 1149.1 Manufacturer ID for Microsemi

#### **User Info**

The User Information section reports the information read from the User ROW (UROW) of IGLOO, ProASIC3, SmartFusion and Fusion devices. The User Row includes user design information as well as troubleshooting information, including:

- Design name (10 characters max)
- Design check sum (16-bit CRC)
- Last programming setup used to program/erase any of the silicon features.
- FPGA Array / Fabric programming cycle count

#### For example:

```
User Information:
UROW data (HEX): 603a04e0a1c2860e59384af926fe389f
Programming Method: STAPL
Programmer: FlashPro3
Programmer Software: FlashPro vX.X
Design Name: ABCBASICTO
Design Check Sum: 603A
Algorithm Version: 19
Array Prog. Cycle Count: 19
```

#### Table 3 · Device Status Report User Info Description

Category	Field	Description
User Row Data	(Example) UROW data (HEX): 603a04e0a1c2860e59384af926fe389f	Raw data from User Row (UROW)



Category	Field	Description
Programming Troubleshooting Info	(Example) Programming Method: STAPL Programmer: FlashPro3 Programmer Software: FlashPro v8.6 Algorithm Version: 19	Known programming setup used. This includes: Programming method/file, programmer and software. It also includes programming Algorithm version used.
Design Info	(Example) Design Name: ABCASICTO Design Check Sum: 603A	Design name (limited to 10 characters) and check sum. Design check sum is a 16-bit CRC calculated from the fabric (FPGA Array) datastream generated for programming. If encrypted datastream is generated selected, the encrypted datastream is used for calculating the check sum.

### **Device State**

The device state section contains:

- IR-Capture register value, and
- The FPGA status

The IR-Capture is the value captured by the IEEE1149.1 instruction register when going through the IR-Capture state of the IEEE 1149.1 state machine. It contains information reflecting some of the states of the devices that is useful for troubleshooting.

One of the bits in the value captured is the ISC\_DONE value, specified by IEEE 1532 standard. When the value is '1' it means that the FPGA array/fabric is programmed and enabled. This is available for IGLOO, ProASIC3, SmartFusion and Fusion devices.

For example:

Device State: IRCapture Register (HEX): 55 FPGA Array Status: Programmed and enabled

#### For a blank device:

Device State: IRCapture Register (HEX): 51 FPGA Array Status: Not enabled

## **Factory Data**

The Factory Data section lists the Factory Serial Number (FSN). Each of the IGLOO, ProASIC3, SmartFusion and Fusion devices has a unique 48-bit FSN.

#### Security

The security section shows the security options for the FPGA Array, FlashROM and Flash Memory (NVM) block that you programmed into the device.

For example, using a Fusion AFS600 device:

Security:

#### SmartDebug User Guide



```
Security Register (HEX): 000000088c01b
FlashROM
Write/Erase protection: Off
Read protection: Off
Encrypted programming: Off
FPGA Array
Write/Erase protection: Off
Verify protection: Off
Encrypted programming: Off
FlashMemory Block 0
Write protection: On
Read protection: On
Encrypted programming: Off
FlashMemory Block 1
Write protection: On
Read protection: On
Encrypted programming: Off
```

#### Table 4 · Device Status Report - Security Description

Security Status Info	Description
Security Register (HEX)	Raw data captured from the device's security status register
Write/Erase Protection	Write protection is applicable to FlashROM, FPGA Array (Fabric)and Flash Memory (NVM) blocks. When On, the Silicon feature is write/erase protected by user passkey.
Read Protection	Read protection is applicable to FlashROM and Flash Memory (NVM) blocks. When On, the Silicon feature is read protected by user passkey.
Verify Protection	Verify Protection is only applicable to FPGA Array (Fabric) only. When On, the FPGA Array require user passkey for verification.
	Reading back from the FPGA Array (Fabric) is not supported.
	Verification is accomplished by sending in the expected data for verification.
Encrypted Programming	Encrypted Programming is supported for FlashROM, FPGA Array (Fabric) and Flash Memory (NVM) blocks. When On, the silicon feature is enable for encrypted programmed. This allows field design update with encrypted datastream so the user design is protected.

#### **Encrypted Programming**

To allow encrypted programming of the features, the target feature cannot be Write/Erase protected by user passkey.

The security settings of each silicon feature when they are enabled for encrypted programming are listed below.

#### **FPGA Array (Fabric)**

Write/Erase protection: Off Verify protection: Off Encrypted programming: On



Set automatically by Designer or FlashPro when you select to enable encrypted programming of the FPGA Array (Fabric). This setting allows the FPGA Array (Fabric) to be programmed and verified with an encrypted datastream.

#### **FlashROM**

```
Write/Erase protection: Off
Read protection: On
Encrypted programming: On
```

Set automatically by Designer or FlashPro when you select to enable encrypted programming of the FlashROM. This setting allows the FlashROM to be programmed and verified with an encrypted datastream.

FlashROM always allows verification. If encrypted programming is set, verification has to be performed with encrypted datastream.

Designer and FlashPro automatically set the FlashROM to be read protected by user passkey when encrypted programming is enabled. This protects the content from being read out of the JTAG port after encrypted programming.

#### Flash Memory (NVM) Block

Write/Erase protection: Off Read protection: On Encrypted programming: On

The above setting is set automatically set by Designer or FlashPro when you select to enable encrypted programming of the Flash Memory (NVM) block. This setting allows the Flash Memory (NVM) block to be programmed with an encrypted datastream.

The Flash Memory (NVM) block does not support verification with encrypted datastream.

Designer and FlashPro automatically set the Flash Memory (NVM) block to be read protected by user passkey when encrypted programming is enabled. This protects the content from being read out of the JTAG port after encrypted programming.