UG0888
User Guide
PolarFire SoC FPGA Trace and Debug

a MICROCHIP company

Microsemi Headquarters
One Enterprise, Aliso Viejo,
CA 92656 USA
Within the USA: +1 (800) 713-4113
Outside the USA: +1 (949) 380-6100
Sales: +1 (949) 380-6136
Fax: +1 (949) 215-4996
Email: sales.support@microsemi.com
www.microsemi.com

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.

50200888. 1.0 12/19

Contents

Figures

Tables

1 Acronyms

The following acronyms are used in this document.

Table 1 • **List of Acronyms**

Acronym	Expanded
AXI	Advanced Extensible Interface
IDE	Integrated Development Environment
JTAG	Joint Test Action Group
JPAM	JTAG Processor Analytic Module
MSS	Micro-Processor Subsystem
SMB	System Memory Buffer
TAP	Test Access Port

2 PolarFire SoC Trace

PolarFire SoC devices contain industry's first RISC-V multi-core Microprocessor Subsystem (MSS) with 5x 64-bit RISC-V processor cores. The PolarFire SoC MSS contains a Trace block to enable trace functionalities. This chapter describes the features, components, and use models of Trace.

As a prerequisite, see *UG0880: PolarFire SoC FPGA MSS User Guide* to know MSS buses/blocks that are tapped by the Trace block.

2.1 Trace Features

The Trace block implements a message-based protocol between a Trace Integrated Development Environment (IDE) and the Trace block via JTAG. The Trace block provides the following features:

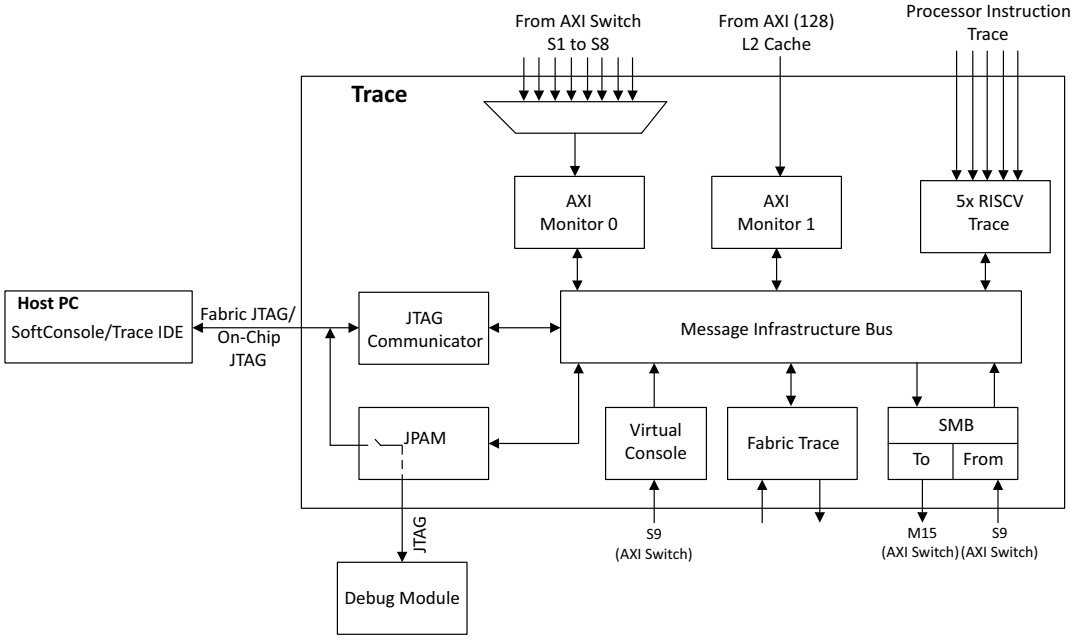
- Instruction trace per processor core
- Full AXI (64) trace of a selectable single slave Interface on the AXI Switch
- AXI transaction (no-data) trace of AXI (128) bus between L2 cache to DDR
- Status monitoring of up to 40 fabric signals

The Trace block collects the trace data and sends it to a Trace IDE running on a Host PC. The trace data can be used to identify performance and fault points during program execution.

2.2 Trace Architecture

The following figure shows the high-level architecture and components of the Trace block.

Figure 1 • **Trace Block Diagram**



2.3 Trace Components

The Trace contains the following components:

- JTAG Communicator
- JPAM
- Message Infrastructure Bus
- AXI Monitor 0
- AXI Monitor 1
- Virtual Console
- System Memory Buffer (SMB)
- RISC-V Trace
- Fabric Trace

2.3.1 JTAG Communicator

JTAG Communicator connects a Host to the Trace block via JTAG. The JTAG Communicator Test Access Point (TAP) contains an 8-bit instruction register (IR) and supports the JTAG instructions.

2.3.2 JPAM

JTAG Processor Analytic Module (JPAM) provides access to the JTAG debug module of the CPU Core Complex. This debug module enables the debugging of processor cores. JPAM can connect to the fabric JTAG controller or the On-Chip JTAG controller.

2.3.3 Message Infrastructure Bus

The message infrastructure bus provides a basic message and event routing function. This component enables message exchange between JTAG Communicator and analytic modules, and vice versa.

The message infrastructure bus contains:

- A 32-bit bus configured for downstream messages for data trace
- An 8-bit bus for upstream messages (control)

These two buses operate using the MSS AXI clock.

2.3.4 AXI Monitor 0

AXI Monitor 0 is an analytic module that provides full address and data trace on a selectable single slave interface of the AXI Switch (S1 to S8). This module also provides an 3-bit GPIO control unit to enable the trace of slave port from S1:S8. For example, setting GPIO_0 enables the trace of S1 port on the AXI switch.

2.3.5 AXI Monitor 1

AXI Monitor 1 is an analytic module that provides full address trace on the AXI4-128 bus between the CPU Core Complex L2 Cache and DDR. AXI Monitor 1 does not provide data trace ability. This component enables the trace of effectiveness of the L2 Cache and DDR response rates.

2.3.6 Virtual Console

Virtual Console is an analytic module that provides an AXI3/AXI4 slave interface to enable communication between the Debug module and the Trace IDE. This peripheral interface enables the software to communicate with the Debug module via the Message Infrastructure Bus sub-block of Trace.

2.3.7 System Memory Buffer (SMB)

System Memory Buffer (SMB) is a communicator module that provides buffering and storing of messages in a region of shared system memory. The SMB connects to the system memory via AXI Switch and to the Message Infrastructure Bus sub-block via input and output message interfaces.

2.3.8 RISC-V Trace

RISC-V trace module is a processor analytic module that provides instruction trace from a processor core. Optional statistics counters are also available. The five identical RISC-V trace modules support the RISC-V ISA enabling the trace of E51 and four U54 processor cores. These modules support filtering which can be used to specify the attributes to be traced and when to be traced.

2.3.9 Fabric Trace

Fabric Trace is a Status Monitor analytic module that provides a 40 channel logic analyzer required for hardware tracing of the FPGA fabric design concurrently with CPU and AXI trace functions. It also provides an 8-bit GPIO control unit enabling the Trace block to control internal FPGA fabric functions. One of these GPIO connections can be used to control a 2:1 MUX allowing greater than 32 channels to be traced (32 at a time) without reprogramming the PolarFire SoC device.

Table 2 lists the interfaces ports of Fabric Trace.

Table 2 • **Fabric Trace IO Ports**

EIP Connection	MSS Direction	Function
USOC_TRACE_CLOCK_F2M	Input	Clock input to Fabric Trace
USOC_TRACE_VALID_F2M	Input	Valid input to Fabric Trace
USOC_TRACE_DATA_F2M[39:0]	Input	32-bit trace input to Fabric Trace
USOC_CONTROL_DATA_M2F[7:0]	Output	8-bit GPIO to the fabric

2.4 Use Models

This section describes the following use models of the Trace block:

- Processor Trace
- Data Trace on AXI Switch Slave Port
- Address and Data Trace on DDR Controller

Note: Future versions of SoftConsole will include integrated Trace capabilities, which will enable the user to collect the Trace data.

2.4.1 Processor Trace

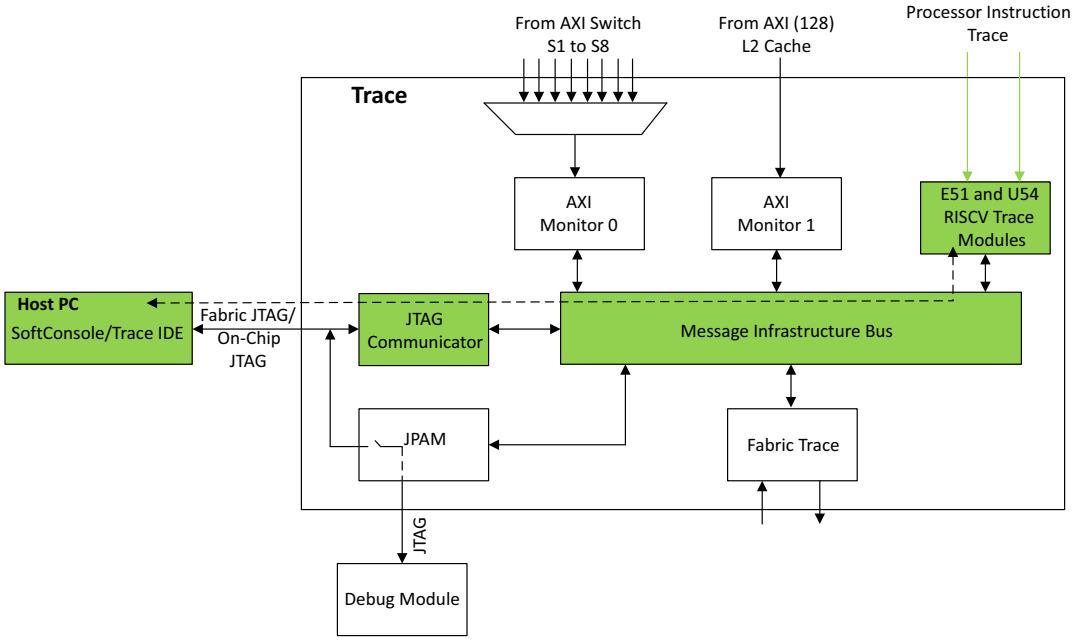
Processor trace involves the following steps:

1. Programming the design on the PolarFire SoC Device using FlashPro Express.
2. Running the firmware on the required processors (E51 and U54) using SoftConsole.
3. Discovering Trace modules using the Trace IDE.
4. Configuring the required RISC-V trace analytic modules using the Trace IDE.
5. Running and monitoring the trace data using the Trace IDE.

Note: Processor trace data contains the assembly code which must be same as in the debugger's Disassembly view.

The user can configure one or more RISC-V analytic modules for multi-processor trace. Figure 2 shows the required trace modules and the flow involved in this trace use model.

Figure 2 • **Processor Trace Model**



2.4.2 Data Trace on AXI Switch Slave Port

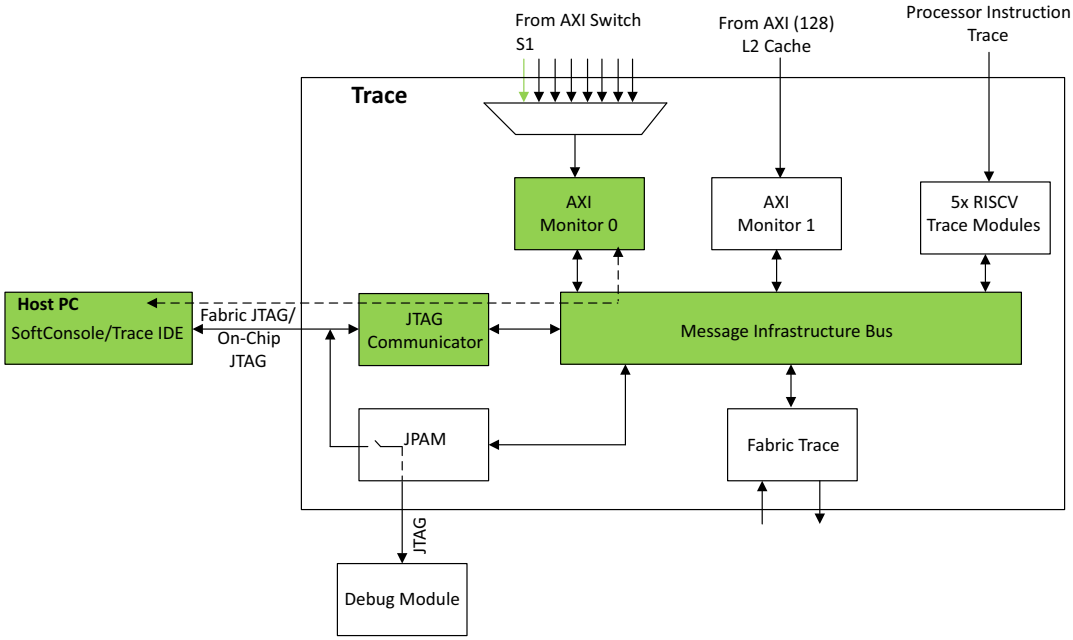
Data trace on a slave port involves the following steps:

1. Programming the design on the PolarFire SoC Device using FlashPro Express.
2. Running the firmware to create traffic on S1 slave port using SoftConsole.
3. Discovering Trace modules using the Trace IDE.
4. Configuring the AXI Monitor 0 bus monitor using the Trace IDE.
5. Running and monitoring the AXI(64) trace data using the Trace IDE.

Note: The monitored trace data must match with the data sent/received on S1 slave.

Figure 3 shows the required trace modules and the flow involved in this trace use model.

Figure 3 • **AXI Switch Data Trace User Model**



Note: The configuration of AXI Monitor 0 involves setting GPIO_0 to enable trace on the S1 port connected to FIC0.

2.4.3 Address and Data Trace on DDR Controller

The Trace block can be used to monitor the following:

- Address trace on the AXI 128 bus connected to DDR cached region.
- Data trace on S7 slave (AXI 64) connected to DDR non-cached region.

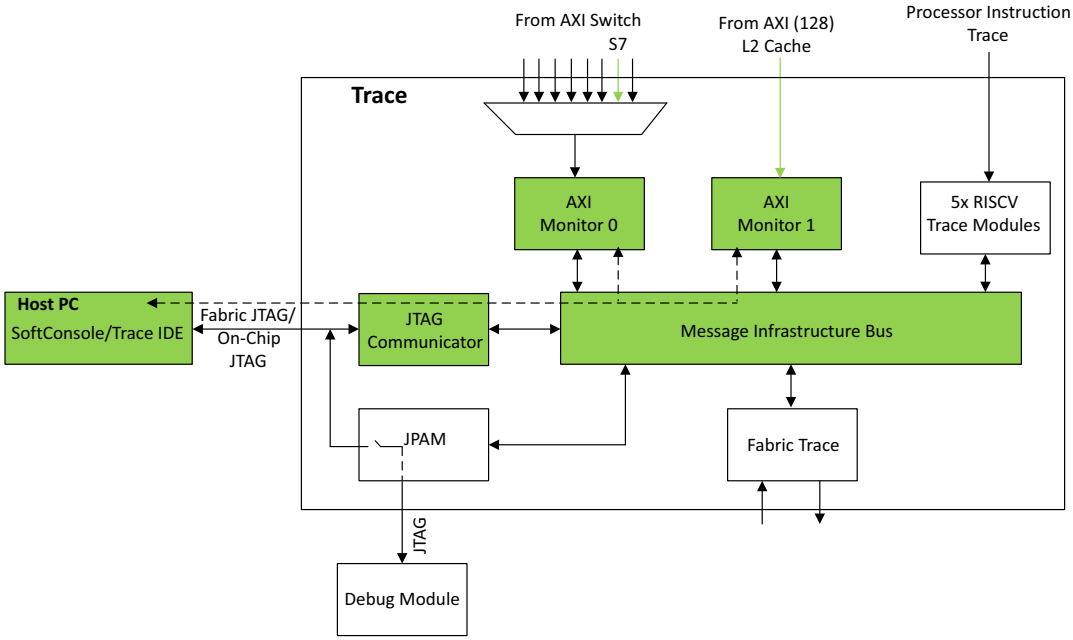
Address and data trace for DDR controller involves the following steps:

1. Programming the design on the PolarFire SoC Device using FlashPro Express.
2. Running the firmware to create traffic on S7 (AXI 64) and AXI(128) buses using SoftConsole.
3. Discovering the required Trace modules using the Trace IDE.
4. Configuring the AXI Monitor 0 and AXI Monitor 1 using the Trace IDE.
5. Running and monitoring the data and address trace data using the Trace IDE.

Note: The monitored address trace from AXI Monitor 1 module must match with the addresses used in the firmware. Also, the monitored data trace from AXI Monitor 0 module must match with the data sent/received on S7.

Figure 4 shows the required trace modules and the flow involved in this trace use model.

Figure 4 • **Data and Address Trace User Model**



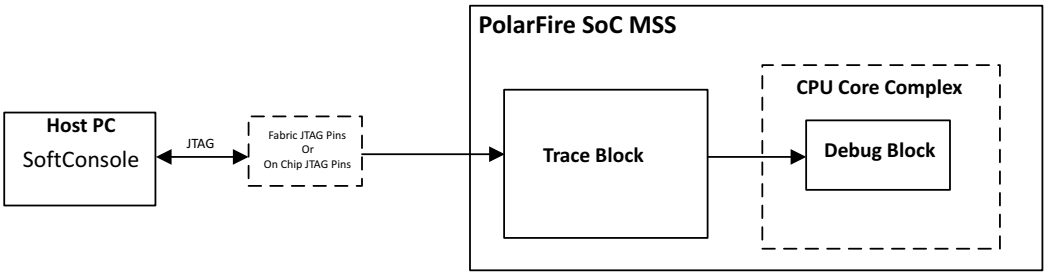
3 PolarFire SoC Debug

PolarFire SoC MSS contains a Debug block that allows an external host PC to initiate debug operations on MSS processor cores via JTAG. This chapter describes the debug architecture and features including multi-processor debugging.

3.1 Debug Architecture

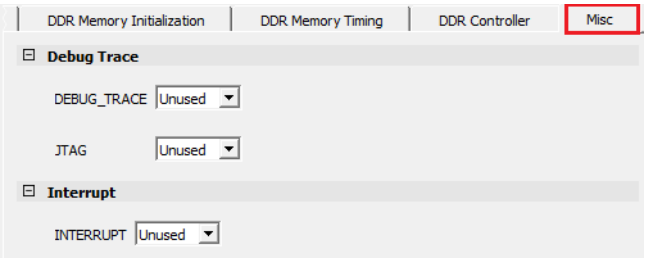
Debugging of MSS processor cores can be performed via fabric JTAG I/Os or on-chip JTAG I/Os as shown in Figure 5. This option can be selected in Libero SoC->PolarFire SoC MSS Configurator as shown in Figure 6.

Figure 5 • **Debug Connectivity**



The Trace and Debug options can be configured using the MSS Configurator as shown in Figure 6.

Figure 6 • **Trace and Debug Options**



Note: Future versions of SoftConsole will include multi-CPU debug capabilities.

4 Revision History

The revision history describes the changes that were implemented in the document. The changes are listed by revision, starting with the most current publication.

4.1 Revision 1.0

The first publication of this document.