

Arm® CoreSight™ ELA-600 Embedded Logic Analyzer

Revision: r0p0

Technical Reference Manual

The logo for Arm, consisting of the lowercase letters 'arm' in a bold, sans-serif font.

Arm® CoreSight™ ELA-600 Embedded Logic Analyzer

Technical Reference Manual

Copyright © 2017, 2018 Arm Limited (or its affiliates). All rights reserved.

Release Information

Document History

Issue	Date	Confidentiality	Change
0000-00	30 November 2017	Confidential	First release for r0p0
0000-01	17 February 2018	Non-Confidential	Second release for r0p0

Non-Confidential Proprietary Notice

This document is protected by copyright and other related rights and the practice or implementation of the information contained in this document may be protected by one or more patents or pending patent applications. No part of this document may be reproduced in any form by any means without the express prior written permission of Arm. **No license, express or implied, by estoppel or otherwise to any intellectual property rights is granted by this document unless specifically stated.**

Your access to the information in this document is conditional upon your acceptance that you will not use or permit others to use the information for the purposes of determining whether implementations infringe any third party patents.

THIS DOCUMENT IS PROVIDED “AS IS”. ARM PROVIDES NO REPRESENTATIONS AND NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY, SATISFACTORY QUALITY, NON-INFRINGEMENT OR FITNESS FOR A PARTICULAR PURPOSE WITH RESPECT TO THE DOCUMENT. For the avoidance of doubt, Arm makes no representation with respect to, and has undertaken no analysis to identify or understand the scope and content of, third party patents, copyrights, trade secrets, or other rights.

This document may include technical inaccuracies or typographical errors.

TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL ARM BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF ARM HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

This document consists solely of commercial items. You shall be responsible for ensuring that any use, duplication or disclosure of this document complies fully with any relevant export laws and regulations to assure that this document or any portion thereof is not exported, directly or indirectly, in violation of such export laws. Use of the word “partner” in reference to Arm’s customers is not intended to create or refer to any partnership relationship with any other company. Arm may make changes to this document at any time and without notice.

If any of the provisions contained in these terms conflict with any of the provisions of any click through or signed written agreement covering this document with Arm, then the click through or signed written agreement prevails over and supersedes the conflicting provisions of these terms. This document may be translated into other languages for convenience, and you agree that if there is any conflict between the English version of this document and any translation, the terms of the English version of the Agreement shall prevail.

The Arm corporate logo and words marked with ® or ™ are registered trademarks or trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. All rights reserved. Other brands and names mentioned in this document may be the trademarks of their respective owners. Please follow Arm’s trademark usage guidelines at <http://www.arm.com/company/policies/trademarks>.

Copyright © 2017, 2018 Arm Limited (or its affiliates). All rights reserved.

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

LES-PRE-20349

Confidentiality Status

This document is Non-Confidential. The right to use, copy and disclose this document may be subject to license restrictions in accordance with the terms of the agreement entered into by Arm and the party that Arm delivered this document to.

Unrestricted Access is an Arm internal classification.

Product Status

The information in this document is for a Beta product, that is a product under development.

Web Address

<http://www.arm.com>

Contents

Arm® CoreSight™ ELA-600 Embedded Logic Analyzer Technical Reference Manual

Preface

<i>About this book</i>	7
<i>Feedback</i>	10

Chapter 1

Introduction

1.1	<i>About the ELA-600 Embedded Logic Analyzer</i>	1-12
1.2	<i>Definitions of terms used in this book</i>	1-13
1.3	<i>Compliance</i>	1-14
1.4	<i>Features</i>	1-15
1.5	<i>Interfaces</i>	1-16
1.6	<i>Configuration options</i>	1-17
1.7	<i>Test features</i>	1-18
1.8	<i>Product documentation and design flow</i>	1-19
1.9	<i>Product revisions</i>	1-21

Chapter 2

Functional description

2.1	<i>About the functions</i>	2-23
2.2	<i>Interfaces</i>	2-25
2.3	<i>Clocking and reset</i>	2-27
2.4	<i>Trace control and capture</i>	2-29
2.5	<i>Triggering</i>	2-37
2.6	<i>Authentication interface</i>	2-42

2.7	Parameter summary	2-43
-----	-------------------------	------

Chapter 3

Programmers model

3.1	Access permissions	3-45
3.2	Programming sequence	3-46
3.3	Control register summary	3-47
3.4	Control register descriptions	3-48
3.5	Current State register summary	3-53
3.6	Current State register descriptions	3-54
3.7	RAM register summary	3-56
3.8	RAM register descriptions	3-57
3.9	Trigger State register summary	3-60
3.10	Trigger State register descriptions	3-68
3.11	Integration Mode register summary	3-81
3.12	Integration Mode register descriptions	3-82
3.13	Authentication register summary	3-86
3.14	Authentication register descriptions	3-87
3.15	Device register summary	3-88
3.16	Device register descriptions	3-89
3.17	ID register summary	3-92
3.18	ID register descriptions	3-93

Appendix A

Signal descriptions

A.1	Clocks and reset	Appx-A-98
A.2	Debug APB signals	Appx-A-99
A.3	Observation interface signals	Appx-A-100
A.4	Timestamp interface signals	Appx-A-102
A.5	Authentication interface signals	Appx-A-103
A.6	DFT and MBIST interface signals	Appx-A-104
A.7	Q-Channel Low-Power Interface signals	Appx-A-105
A.8	Output Action signals	Appx-A-106
A.9	External Trigger Input signals	Appx-A-107
A.10	AMBA ATBv4 Interface	Appx-A-108

Appendix B

Revisions

B.1	Revisions	Appx-B-110
-----	-----------------	------------

Preface

This preface introduces the *Arm® CoreSight™ ELA-600 Embedded Logic Analyzer Technical Reference Manual*.

It contains the following:

- *About this book* on page 7.
- *Feedback* on page 10.

About this book

This book is for the Arm® CoreSight™ ELA-600 Embedded Logic Analyzer.

Product revision status

The *rm**pn* identifier indicates the revision status of the product described in this book, for example, *r1p2*, where:

rm Identifies the major revision of the product, for example, *r1*.

pn Identifies the minor revision or modification status of the product, for example, *p2*.

Intended audience

This book is written for system designers, system integrators, and programmers who are designing or programming a *System-on-Chip* (SoC) that uses the ELA-600.

Using this book

This book is organized into the following chapters:

Chapter 1 Introduction

This chapter describes the ELA-600 Embedded Logic Analyzer.

Chapter 2 Functional description

This chapter describes the functionality of the ELA-600.

Chapter 3 Programmers model

This chapter describes the programmers model.

Appendix A Signal descriptions

This appendix describes the external signals of the ELA-600.

Appendix B Revisions

This appendix describes the technical changes between released issues of this book.

Glossary

The Arm® Glossary is a list of terms used in Arm documentation, together with definitions for those terms. The Arm Glossary does not contain terms that are industry standard unless the Arm meaning differs from the generally accepted meaning.

See the *Arm® Glossary* for more information.

Typographic conventions

italic

Introduces special terminology, denotes cross-references, and citations.

bold

Highlights interface elements, such as menu names. Denotes signal names. Also used for terms in descriptive lists, where appropriate.

monospace

Denotes text that you can enter at the keyboard, such as commands, file and program names, and source code.

monospace

Denotes a permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name.

monospace italic

Denotes arguments to monospace text where the argument is to be replaced by a specific value.

monospace bold

Denotes language keywords when used outside example code.

<and>

Encloses replaceable terms for assembler syntax where they appear in code or code fragments. For example:

```
MRC p15, 0, <Rd>, <CRn>, <CRm>, <Opcode_2>
```

SMALL CAPITALS

Used in body text for a few terms that have specific technical meanings, that are defined in the *Arm® Glossary*. For example, IMPLEMENTATION DEFINED, IMPLEMENTATION SPECIFIC, UNKNOWN, and UNPREDICTABLE.

Timing diagrams

The following figure explains the components used in timing diagrams. Variations, when they occur, have clear labels. You must not assume any timing information that is not explicit in the diagrams.

Shaded bus and signal areas are undefined, so the bus or signal can assume any value within the shaded area at that time. The actual level is unimportant and does not affect normal operation.

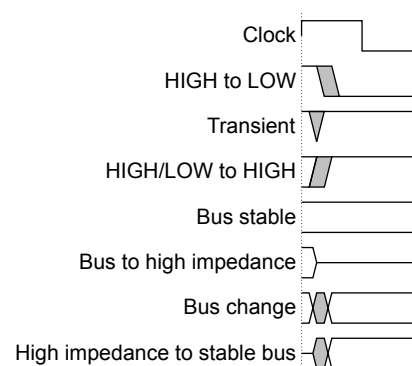


Figure 1 Key to timing diagram conventions

Signals

The signal conventions are:

Signal level

The level of an asserted signal depends on whether the signal is active-HIGH or active-LOW. Asserted means:

- HIGH for active-HIGH signals.
- LOW for active-LOW signals.

Lowercase n

At the start or end of a signal name denotes an active-LOW signal.

Additional reading

This section lists publications by ARM and by third parties.

See *Infocenter* <http://infocenter.arm.com>, for access to ARM documentation.

ARM publications

This book contains information that is specific to this product. See the following documents for other relevant information:

- *Arm® Low Power Interface Specification, Q-Channel and P-Channel Interfaces* (ARM IHI 0068).
- *Arm® AMBA® AXI and ACE Protocol Specification* (ARM IHI 0022).
- *Arm® AMBA® APB Protocol Specification* (ARM IHI 0024).
- *Arm® AMBA® ATBv4 ATB Protocol Specification* (ARM IHI 0032).
- *Arm® CoreSight™ Architecture Specification* (ARM IHI 0029).

The following confidential books are only available to licensees:

- *Arm® CoreSight™ ELA-600 Embedded Logic Analyzer Configuration and Integration Manual* (ARM 101089).

Other publications

This section lists relevant documents published by third parties:

- *JEDEC Standard Manufacturer's Identification Code, JEP106* <http://www.jedec.org>.

Feedback

Feedback on this product

If you have any comments or suggestions about this product, contact your supplier and give:

- The product name.
- The product revision or version.
- An explanation with as much information as you can provide. Include symptoms and diagnostic procedures if appropriate.

Feedback on content

If you have comments on content then send an e-mail to errata@arm.com. Give:

- The title *Arm CoreSight ELA-600 Embedded Logic Analyzer Technical Reference Manual*.
- The number 101088_0000_01_en.
- If applicable, the page number(s) to which your comments refer.
- A concise explanation of your comments.

Arm also welcomes general suggestions for additions and improvements.

————— **Note** —————

Arm tests the PDF only in Adobe Acrobat and Acrobat Reader, and cannot guarantee the quality of the represented document when used with any other PDF reader.

Chapter 1

Introduction

This chapter describes the ELA-600 Embedded Logic Analyzer.

It contains the following sections:

- *1.1 About the ELA-600 Embedded Logic Analyzer* on page 1-12.
- *1.2 Definitions of terms used in this book* on page 1-13.
- *1.3 Compliance* on page 1-14.
- *1.4 Features* on page 1-15.
- *1.5 Interfaces* on page 1-16.
- *1.6 Configuration options* on page 1-17.
- *1.7 Test features* on page 1-18.
- *1.8 Product documentation and design flow* on page 1-19.
- *1.9 Product revisions* on page 1-21.

1.1 About the ELA-600 Embedded Logic Analyzer

The ELA-600 Embedded Logic Analyzer is a component for debugging hardware-related issues.

Debug signals are connected from the IP being debugged to the ELA-600, which compares the signals with a target value and drives actions. There is an optional trace capability that can be used to generate a history of the debug signals at any point in time.

The ELA-600 can be configured to trace to an integrated SRAM or over an ATBv4 interface.

1.2 Definitions of terms used in this book

This Technical Reference Manual uses terms that are specific to the ELA-600 Embedded Logic Analyzer.

The following terms have specific meanings within the context of this Technical Reference Manual. Wherever they are used throughout the book they are shown in *italics* and have the meanings that are shown here:

Trigger State

One of the eight states that the ELA-600 trigger logic can be in. The *Trigger State* controls which *Signal Group* signals are routed to the comparison logic, target comparison values, comparison and counter control, and output actions. The ELA-600 advances to the next *Trigger State* when its *Trigger Condition* is met.

————— **Note** —————

The sequence of *Trigger States* is programmable and does not depend on implementation.

Trigger Signal Comparison

The comparison of the *External Trigger Input Signals* and selected *Signal Group* with a target value and mask that is determined by the current *Trigger State*.

Trigger Counter Comparison

The comparison of the up-counter of the current *Trigger State* with its target value. The counter can be incremented by **ELACLK** or by *Trigger Signal Comparison* matches. The counter can be reset by a *Trigger Signal Comparison* match.

Trigger Signal Alternative Comparison

An alternative comparison of the *External Trigger Input Signals* and selected *Signal Group* with a target value and mask that is determined by the current *Trigger State*.

Trigger Condition

When the *Trigger Condition* is met, the ELA-600 generates an *Output Action* and transitions to the next *Trigger State*. If *Trigger Counter Comparison* is enabled, the *Trigger Condition* is met when the *Trigger Counter Comparison* is true. If *Trigger Counter Comparison* is disabled, the *Trigger Condition* is met when the *Trigger Signal Comparison* is met.

External Trigger Input Signals

The ELA-600 supports eight input signals that can form part of the *Trigger Signal Comparison*. The *External Trigger Input Signals* can come from other ELA-600 instances, a CoreSight Cross Trigger Interface, or other logic in the SoC.

Signal Group

A group of input signals from the Observation interface. The ELA-600 supports up to 12 *Signal Groups*, each of which is 64 bits, 128 bits, or 256 bits wide, determined by the `GRP_WIDTH` parameter.

Output Action

The ELA-600 generates an *Output Action* when the *Trigger Condition* is met.

The *Output Action* can:

- Drive the **STOPCLOCK** output for scan-dump analysis.
- Drive a CoreSight Embedded Cross Trigger through **CTTRIGOUT[1:0]** to a CoreSight *Cross Trigger Interface* (CTI).
- Drive other logic through **ELAOUTPUT[3:0]**.

1.3 Compliance

The ELA-600 implements the Arm CoreSight Architecture Specification. It complies with the AMBA APB and ATB Protocol and the Arm Low Power interface Q-Channel specification.

This Technical Reference Manual complements architecture reference manuals, architecture specifications, protocol specifications, and relevant external standards. It does not duplicate information from these sources.

See the *Arm® AMBA® APB Protocol Specification*, the *Arm® AMBA® 4 ATB Protocol Specification*, the *Arm® CoreSight™ Architecture Specification*, and the *Arm® Low Power Interface Specification, Q-Channel and P-Channel Interfaces* for more information.

1.4 Features

The ELA-600 supports ELA-500 features and several new features to assist with debug. Some of the key features are programmable *Trigger States*, programmable *Output Actions* for each *Trigger State*, ATB trace, and additional *External Trigger Input Signals*.

The ELA-600 has the following key features:

- Five or eight programmable *Trigger States*.
- Eight programmable actions, to allow each *Trigger State* to control:
 - Stop clock.
 - Trace control.
 - CoreSight cross-trigger.
 - Four general-purpose trigger outputs.
- A programmable 32-bit counter for each *Trigger State* that can be used to delay output actions, count events, or as a watchdog timer.
- ATB trace supports:
 - A configuration that can trace over a 128-bit ATB interface to allow a greater amount of data to be traced without an integrated SRAM.
 - Data compression, byte packing, and selective trace of bytes in a SIGNALGRP to reduce the amount of trace data.
- Improved trigger and trace capture for both SRAM and ATB trace configurations:
 - SRAM configuration supports a 256-bit comparator.
 - Each trigger state comparator can be segmented into multiple 32-bit comparators to allow greater comparison capabilities for logic analysis and tracing capability.
 - Supports SIGQUAL qualifier signals for each SIGNALGRP for signals that can be used to qualify valid data without needing to be traced.
 - Supports trace of trigger state counters to support measurement of events for performance measurements such as latency.
 - Supports simultaneous trace of two SIGNALGRPs on the same clock cycle with configurable FIFOs.
- An Observation interface consisting of 12 *Signal Groups* with a configurable width of 64, 128, or 256 debug signals.
- Eight *External Trigger Input Signals* that can be masked and compared against a target value for each *Trigger State*. An *Output Action* from one logic analyzer can be connected to these inputs on a second logic analyzer to cause a direct cross-trigger, independently of the CoreSight *Embedded Cross Trigger* (ECT) infrastructure. This feature enables users to have a lower-latency *Embedded Logic Analyzer* (ELA) cross-trigger mechanism that does not rely on the correct operation of software-debug cross-triggering components.
- Programmable *Trigger Condition* comparison with the ability to change the target comparison by selectively masking signals, and selecting =, !=, <, <=, >, >= for comparison of the masked signals and counter target value comparisons.
- Programmable *Trigger Alternative Condition* comparison with the ability to change the target comparison by selectively masking signals, and selecting =, !=, <, <=, >, >=, for comparison of the masked signals and counter target value comparisons.
- Optional support of signal trace using an integrated SRAM or ATB interface. The SRAM trace depth is configurable. Timestamp trace capture is possible and enables correlation of ELA trace with other CoreSight trace sources for both SRAM and ATB configurations.

1.5 Interfaces

The ELA-600 has numerous external interfaces, including interfaces for debug signals, trigger inputs, and authentication permissions.

The ELA-600 has the following external interfaces:

- An Observation interface to capture signals from the IP being debugged.
- An *External Trigger Input Signals* interface that enables the ELA-600 to be triggered by external logic.
- An Authentication interface that determines the type of accesses permitted.
- A debug APB slave interface that enables access to the configuration and status registers.
- A configurable ATB interface to allow trace to a CoreSight trace infrastructure.
- A timestamp interface to provide timestamp information with captured trace data.
- A Low-Power Q-Channel interface to determine when **ELACLK** can be stopped.
- A *Memory Built-In Self-Test* (MBIST) interface for testing SRAM.

1.6 Configuration options

This section describes the configuration options available in the ELA-600.

1.6.1 Configurable parameters

There are several configurable options available in the ELA-600.

See [2.7 Parameter summary on page 2-43](#).

1.6.2 Static parameters

There are no configurable static parameters in the ELA-600.

1.6.3 Tie-off signals

There are no configurable tie-off signals in the ELA-600.

1.7 Test features

The ELA-600 has several test features.

See the *Arm® CoreSight™ ELA-600 Embedded Logic Analyzer Configuration and Integration Manual* (ARM 101089) for information about the test features.

1.8 Product documentation and design flow

The ELA-600 documentation includes a Technical Reference Manual and a Configuration and Integration Manual. These books relate to the ELA-600 design flow.

Documentation

The ELA-600 documentation includes the following books:

Technical Reference Manual

The *Technical Reference Manual* (TRM) describes the functionality and the effects of functional options on the behavior of the ELA-600. It is required at all stages of the design flow. The choices that you make in the design flow can mean that some behavior described in the TRM is not relevant. If you are programming the ELA-600 then contact:

- The implementer to determine:
 - What integration, if any, was performed before implementing the ELA-600.
 - The build configuration of the implementation.

————— **Note** —————

Build configuration information is also readable from the DEVID registers.

- The integrator to determine the pin configuration of the device that you are using.

Configuration and Integration Manual

The *Configuration and Integration Manual* (CIM) describes:

- The available build configuration options and related issues in selecting them.
- How to configure the *Register Transfer Level* (RTL) with the build configuration options.
- How to integrate the ELA-600 into a SoC. This includes a description of the integration kit and describes the pins that the integrator must tie off to configure the macrocell for the required integration.
- How to implement the ELA-600 into your design. This includes floorplanning guidelines, *Design for Test* (DFT) information, and how to perform netlist dynamic verification on the ELA-600.
- The processes to sign off the integration and implementation of the design.

The Arm product deliverables include reference scripts and information about using them to implement your design.

Reference methodology documentation from your EDA tools vendor complements the CIM.

The CIM is a confidential book that is only available to licensees.

Design flow

The ELA-600 is delivered as synthesizable RTL. Before it can be used in a product, it must go through the following processes:

Implementation

The implementer configures and synthesizes the RTL to produce a hard macrocell. This includes integrating RAMs into the design.

Integration

The integrator connects the implemented design into a SoC. This includes connecting it to a memory system and peripherals.

Programming

This is the final process. The system programmer develops the software that is required to configure and initialize the ELA-600, and tests the required application software.

Each process:

- Can be performed by a different party.
- Can include implementation and integration choices that affect the behavior and features of the ELA-600.

The operation of the final device depends on:

Build configuration

The implementer chooses the options that affect how the RTL source files are pre-processed. These options usually include or exclude logic that affects one or more of the area, maximum frequency, and features of the resulting macrocell.

Software configuration

The programmer configures the ELA-600 by programming particular values into registers. This affects the behavior of the ELA-600.

Note

This Technical Reference Manual refers to implementation-defined features that are applicable to build configuration options. Reference to a feature that is included means that the appropriate build and pin configuration options are selected. Reference to an enabled feature means a feature that has been configured by software.

1.9 Product revisions

This section describes the differences in functionality between product revisions of the ELA-600.

r0p0 First release.

Chapter 2

Functional description

This chapter describes the functionality of the ELA-600.

It contains the following sections:

- *2.1 About the functions* on page 2-23.
- *2.2 Interfaces* on page 2-25.
- *2.3 Clocking and reset* on page 2-27.
- *2.4 Trace control and capture* on page 2-29.
- *2.5 Triggering* on page 2-37.
- *2.6 Authentication interface* on page 2-42.
- *2.7 Parameter summary* on page 2-43.

2.1 About the functions

This section describes the functional blocks in the ELA-600.

The ELA-600 can support one of three optional configurations:

- No trace
- SRAM trace
- ATB trace

Note

See the *Arm® CoreSight™ ELA-600 Embedded Logic Analyzer Configuration and Integration Manual* for more information about setting the trace options.

The ELA-600 can be configured to have 64, 128, or 256 debug signals in a *Signal Group*. There are eight *External Trigger Input Signals* that can be used for cross-triggering from other CoreSight components including another ELA-600.

The ELA-600 is programmed from an APB bus and has architectural registers that enable identification in the CoreSight topology.

The ELA-600 also provides support for:

- A CoreSight authentication interface.
- An optional SRAM trace interface with configurable trace depth.
- Insertion of timestamps into the trace data.

There are seven output actions that can be used for various functions, such as stopping the clock to enable the system state to be extracted using a scan chain, cross triggering to a CoreSight debug subsystem, and other system-specific actions.

The following figure shows the functional blocks of the ELA-600:

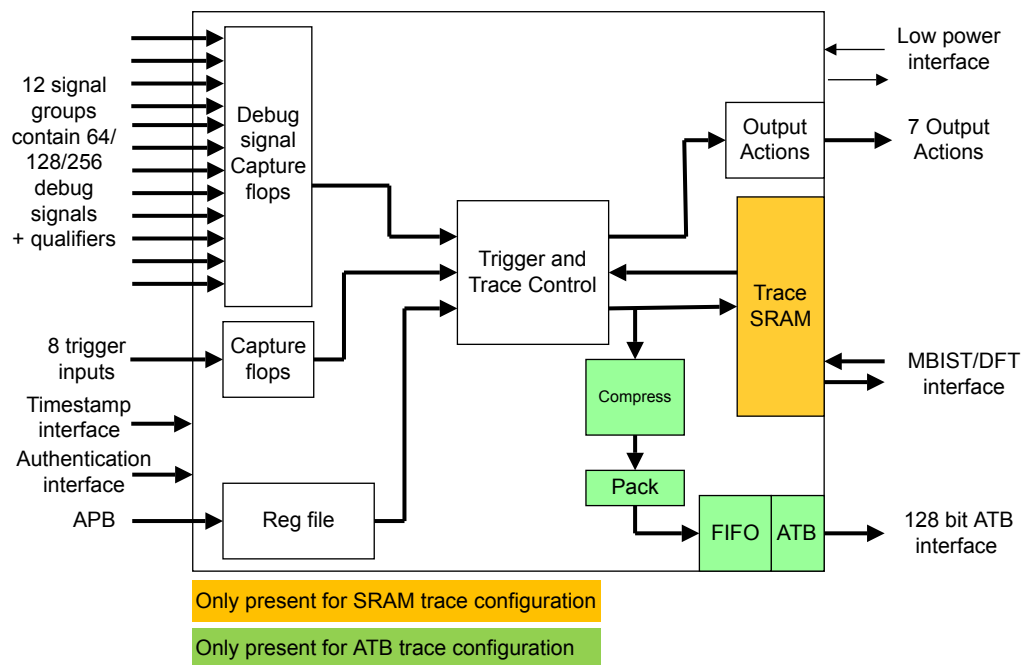


Figure 2-1 ELA-600 processor block diagram

The following figure shows how to use a ELA-600 in a system:

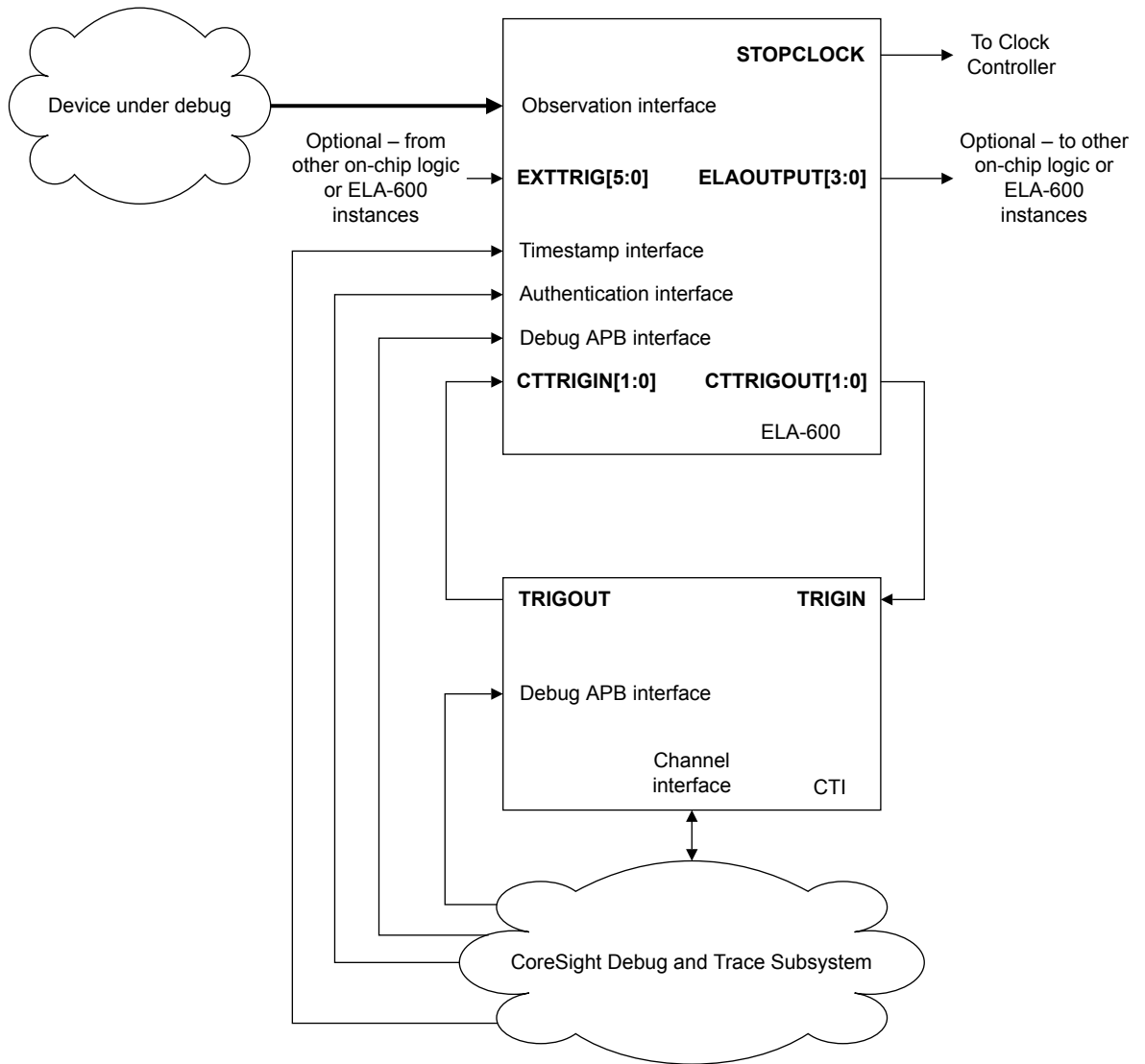


Figure 2-2 How to use the ELA-600 in a system

2.2 Interfaces

The ELA-600 has numerous external interfaces. Some of these include interfaces to debug signals, trigger inputs, and authentication permissions.

The ELA-600 has the following interfaces:

Debug APB slave interface

This interface provides access to the ELA-600 configuration register and status registers. See the *Arm® AMBA® APB Protocol Specification* and the *Arm® CoreSight™ Architecture Specification* for more information about the debug APB signals.

Observation Interface

This consists of 12 *Signal Group* buses of 64, 128, or 256 bits, depending on the configuration parameter `GRP_WIDTHH`.

External Trigger inputs

There are eight trigger inputs that can be used as trigger conditions. These inputs can be sourced from signals from a CoreSight CTI, or other on-chip signals, such as interrupts and debug requests, or can be an output signal from another ELA-600.

Timestamp interface

This interface accepts a 64-bit natural binary value from a timestamp generator in the system. Timestamp values are inserted into the ELA-600 trace output.

Authentication interface

The **DBGEN**, **NIDEN**, **SPIDEN**, and **SPNIDEN** signals are supported as described in the *Arm® CoreSight™ Architecture Specification*.

SRAM trace interface

If present, the SRAM trace interface connects to the SRAM that is used to store the captured trace data.

The following figure shows the SRAM read access timing:

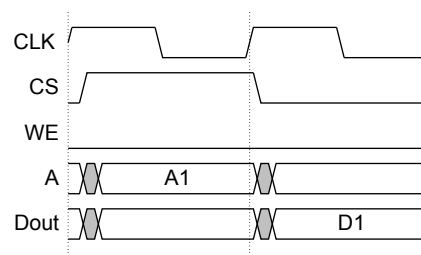


Figure 2-3 SRAM read access timing

The following figure shows the SRAM write access timing:

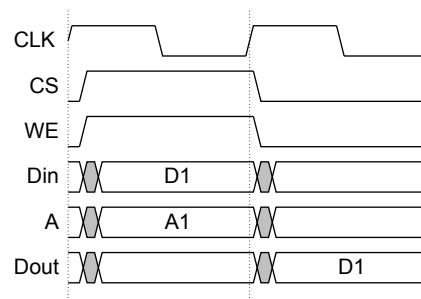


Figure 2-4 SRAM write access timing

SRAM MBIST interface

The *Memory Built-In Self-Test* (MBIST) interface provides a functional access path to the memories for self-test purposes.

Q-Channel interface

The ELA-600 provides a Q-Channel interface that can be used by a clock controller to control stopping of ELACLK:

- ELA-600 signals opportunities for clock stopping to the clock controller by driving **ELAQACTIVE** LOW.
- While **ELAQACTIVE** is LOW, the clock controller can request clock stopping by driving **ELAQREQn** LOW.
- ELA-600 can accept ($\text{ELAQACCEPTn} = 0$) or deny ($\text{ELAQDENY} = 1$) the clock stop request.
- If ELA-600 accepts the clock stop request, then the clock can be stopped.
- When ELA-600 requires a clock again, it will drive **ELAQACTIVE** HIGH.
- In response to **ELAQACTIVE** asserting HIGH, the clock controller must restart clock and drive **ELAQREQn** HIGH.

The **ELAQACTIVE** signal is driven by an OR of the following signals:

- **PSELDBG**.
- $\text{CTRL.TRACE_BUSY} = 1$, that indicates a trace write is in progress.
- $\text{CTRL.RUN} = 1$.
- $\text{ITCTRL.IME} = 1$.

$\text{ITCTRL.IME} = 1$ is used to assert **ELAQACTIVE** so that **ELACLK** continues to run when integration mode is enabled, to avoid clock gating during integration test of the pulsed signals on **CTTRIGIN** and **EXTTRIGIN**.

An internal active signal is generated with synchronized **PSELDBG**. This internal active signal is used when **ELAQREQn** requests are asserted, to move to the Q_STOPPED state when internal active is LOW, or to the Q_DENIED state when internal active is HIGH.

AMBA ATB interface

- Supports the *Arm® AMBA® ATBv4 ATB Protocol Specification* (ARM IHI 0032).
- Supports a 128-bit ATB interface for both $\text{GRP_WIDTH} = 64$ and 128 . $\text{GRP_WIDTH} = 256$ is not supported. An ATB upsizer or downsizer is required to connect to ATB trace infrastructures that are not 128-bits wide.
- **ATID[6:0]** is programmable in $\text{ATBCTRL.ATID_VALUE}[14:8]$
- Supports ATB trace trigger generation ($\text{ATID}[6:0] = 0x7D$) when a trigger state branches to Final State. Programming $\text{ATBCTRL}[15].\text{ATID_TRIG_EN} = 1$ enables ATID trigger.
- Integration test is supported with **ITATBCTR** registers.
- Uses **ELACLK**, so an external bridge may be needed.
- Does not support **ATCLKEN**. The Q-Channel interface should be used for clock gating and power management.
- Does not support **SYNCREQ**. ASYNCS are automatically generated when the trace stream changes, the trace packet type changes to a timestamp or counter packet, or to identify the basis for delta compression data packets.

Related references

[A.6 DFT and MBIST interface signals on page Appx-A-104.](#)

[A.7 Q-Channel Low-Power Interface signals on page Appx-A-105.](#)

[A.10 AMBA ATBv4 Interface on page Appx-A-108.](#)

2.3 Clocking and reset

This section describes the clock and reset signals and procedures for the ELA-600.

This section contains the following subsections:

- [2.3.1 Clocking on page 2-27.](#)
- [2.3.2 Reset on page 2-27.](#)

2.3.1 Clocking

The ELA-600 has a single clock domain **ELACLK**.

Clock domain synchronization

The ELA-600 performs synchronization of some input signals.

Software must take into account the following restrictions:

- The debug APB registers must only be written when the ELA-600 is stopped, that is, when CTRL.RUN is low.
- The debug APB registers can be read at any time.

When implementing the ELA-600 in a system, you must be aware of the following points:

- All the debug signals in the Observation interface are sampled by **ELACLK**. These signals are not synchronized to **ELACLK** inside the ELA-600, and must be driven from logic clocked by **ELACLK** outside the ELA-600.
- The Authentication interface signals, **SPIDEN**, **DBGEN**, **NIDEN**, and **SPNIDEN**, must be synchronized to **ELACLK** outside the ELA-600.
- To aid debug, it is advantageous to have the logic analyzer operational during reset or during power management events of the sampled IP, such as powering down one of multiple cores. Take care with debug signals that originate from processors or other IP. The sampled IP could be power-gated or have asynchronous resets that could cause glitching or false sampling by the logic analyzer. De-assert **SIGCLKEN<n>** when debug inputs are changing because of:
 - Assertion or de-assertion of isolation logic for debug signals that cross power domains.
 - Assertion or de-assertion of asynchronous resets to the sampled IP.
- The **STOPCLOCK** output must not affect the initialization of the SoC following a Cold reset.
- The **TSVALUE** signals in the Timestamp interface must be synchronized to **ELACLK**.
- Synchronizer cell models are connected to **ELAQREQn** and **CTTRIGOUTACK[1:0]** inputs, and can be replaced with library cells during implementation.

Related references

[3.6 Current State register descriptions on page 3-54.](#)

2.3.2 Reset

The ELA-600 has one functional reset.

RESETn resets the logic in the **ELACLK** domain.

RESETn can be asserted asynchronously to **ELACLK**, and must be synchronously deasserted.

RESETn must only be asserted when powering up the **ELACLK** domain, or when a reset of system debug logic is required. Assertion of **RESETn** when the CTRL.RUN bit is set disables the ELA-600.

Note

Arm recommends that **RESETn** is only asserted when the ELA-600 is already disabled with CTRL.RUN set to 0b0.

A third reset signal, **nMBISTRESET**, which is generated with the SRAM trace configuration, is an asynchronous test mode reset for **ELACLK**. **nMBISTRESET** must be driven HIGH for functional

operation and reset. See the *Arm® CoreSight™ ELA-600 Embedded Logic Analyzer Configuration and Integration Manual* (ARM 101089) for more information on test mode.

2.4 Trace control and capture

The SRAM trace interface on the ELA-600 is configurable using the parameters `RAM_ADDR_SIZE` and `GRP_WIDTH`, when the SRAM trace configuration is generated.

The value of `RAM_ADDR_SIZE` represents the number of SRAM address bits. Software can read the `DEVID` register to determine trace depth. The trace SRAM acts as a circular buffer when trace is being captured, with the SRAM address incrementing automatically.

This section contains the following subsections:

- [2.4.1 Trace control on page 2-29.](#)
- [2.4.2 Trace capture on page 2-29.](#)
- [2.4.3 Simultaneous trace using the highest Trigger State number on page 2-29.](#)
- [2.4.4 SRAM trace format on page 2-30.](#)
- [2.4.5 Counter trace on page 2-31.](#)
- [2.4.6 ATB trace format on page 2-32.](#)
- [2.4.7 Timestamp control on page 2-34.](#)
- [2.4.8 Debug APB registers and interface to SRAM on page 2-35.](#)

2.4.1 Trace control

Trace is controlled by the `CTRL`, `CNTSEL`, `TRIGCTRL`, `PTACTION`, and `ACTION<n>` registers for both SRAM and ATB trace configurations.

The ATB interface is configured with the `ATB_FIFO_DEPTH` parameter, which is set based on the `ATREADY` delays that can occur at the trace infrastructure.

ATB trace also has `TWBSSEL` to select `SIGNALGRP` bytes to trace, `ATBCTRL` to control compression and `ATID`, and `AUXCTRL` to control flush.

Trace can only be active when the ELA-600 is running, that is when `CTRL.RUN` = 1. If `PTACTION.TRACE` is set, trace becomes active when `CTRL.RUN` is set.

When the ELA-600 is running, trace is controlled by `ACTION<n>.TRACE`. It is therefore possible to enable or disable trace at each *Trigger State* transition.

2.4.2 Trace capture

When trace is active, trace capture is controlled in each *Trigger State* by the *Trigger Control Register* (`TRIGCTL<n>`).

The following trace capture options are available:

- Capture `SIGNALGRP` data on every `ELACLK`.
- Capture `SIGNALGRP` data on a *Trigger Signal Comparison* match.
- Capture `SIGNALGRP` data on a *Trigger Counter Comparison* match.
- Capture counter data on a *Trigger Signal Comparison* match.

Related references

[3.10.2 Trigger Control registers on page 3-69.](#)

2.4.3 Simultaneous trace using the highest Trigger State number

The highest *Trigger State* (which is 4 or 7 depending on the `NUM_TRIGSTATE` parameter) includes additional capability that allows simultaneous trace to trace `SIGNALGRP<n>` data while the remaining *Trigger States* are programmed for comparisons and trace writes. *Trigger State* 4 is generated by setting the `NUM_TRIG_STATES` parameter to 5. *Trigger State* 7 is generated by setting the `NUM_TRIG_STATES` parameter to 8.

The `ST_FIFO_DEPTH` parameter can be set so that two FIFOs are generated, which allows tracing to occur on the same cycle by the highest *Trigger States* (`TSSR` register) as well as the other active *Trigger States*.

For example, *Trigger States* 4 or 7 can filter and write trace data on the same cycle as any lower numbered *Trigger States*. The FIFOs are emptied using a fairness arbitration.

If multiple trace writes for *Trigger States* occur on the same clock cycle, trace data from SIGNALGRP<n> or CNTSEL from the highest numbered *Trigger State* takes priority when the parameter ST_FIFO_DEPTH = 0. The trace write from the lower numbered *Trigger States* is dropped and the trace data overwrite bit[5] in the trace header byte is set. In this implementation, where DEVID2 = 4, the trace data from *Trigger State* 4 will take precedence when there is a simultaneous write from another *Trigger State*. This feature allows a prioritized trace of two SIGNALGRPs at the same time, but not the same clock cycle. Setting ST_FIFO_DEPTH = 4 or = 8 allows simultaneous trace for up to four or eight clock cycles.

The *Trigger State Select Register*, controls the second trace comparator. Setting bit[NUM_TRIG_STATES - 1], that is ALTTS[NUM_TRIG_STATES - 1] = 1, enables the independent trace capability of *Trigger State* 4 or 7. When this bit is set, *Trigger State* 4 or 7 cannot be used in a loop with other *Trigger States* using NEXTSTATE<n> and ALTNEXTSTATE<n> with *Trigger State* 4 or 7 as a destination. Also, *Trigger State* 4 or 7 cannot update or drive an ACTION or ALTACTION.

The TSSR *Trigger State* has the following characteristics:

- Alternative comparisons do not function.
- The TSSR *Trigger State* runs in a continuous trace loop as if the NEXTSTATE is programmed to go back to the TSSR *Trigger State*.
- The TSSR *Trigger State* 4 trace stops when the other *Trigger States* reach a final state or CTRL.RUN is set to 0.
- The counters and trace filtering options are functional for the TSSR *Trigger State*. The following trigger states are available when ALTTS4 = 1:
 - TRIGCTRL[NUM_TRIG_STATES - 1].COMPSRC
 - TRIGCTRL[NUM_TRIG_STATES - 1].WATCHRST
 - TRIGCTRL[NUM_TRIG_STATES - 1].COUNTSRC
 - TRIGCTRL[NUM_TRIG_STATES - 1].TRACE
 - TRIGCTRL[NUM_TRIG_STATES - 1].COUNTCLR

The following trigger states are not available:

- TRIGCTRL[NUM_TRIG_STATES - 1].CAPTID
- TRIGCTRL[NUM_TRIG_STATES - 1].ALTCOMP
- TRIGCTRL[NUM_TRIG_STATES - 1].COUNTBRK
- TRIGCTRL[NUM_TRIG_STATES - 1].ALTCOMP_SRC
- Loops that are based on counter reset add an extra clock for the reset. For example:
 - Setting COUNTCOMP4 = 5 with TRIGCTRL4.COUNTSRC = 1, TRIGCTRL4.TRACE = 1 and TRIGCTRL4.COMP = 0b001, results in trace capture when the counter reaches five trigger signal comparisons. The trace capture stays asserted until final_state is reached.
 - Setting TRIGCTRL4.COUNTCLR = 1 resets the counter after five trigger signal comparisons are counted.
 - A trace is captured after every five trigger signal comparisons, with the *Trigger State* going to final_state one ELACLK cycle later, or if CTRL.RUN is cleared.
 - If TRIGCTRL4.TRACE = 0, a trace is captured every six ELACLK cycles when COUNTCOMP4 = 5.

2.4.4 SRAM trace format

The trace SRAM can capture a full *Signal Group* on every ELACLK cycle. The width of the trace SRAM is GRP_WIDTH + 8, with the additional 8 bits being used to record a header byte that identifies the data as either a timestamp or a capture of the *Signal Group*.

Each trace SRAM word contains a data payload and a header byte. The header byte is located at the least significant byte of the SRAM word. The payload data is located in the upper bytes of the SRAM word.

For example, for a 64-bit *Signal Group* configuration:

```
GRP_WIDTH = 64
SRAM data[71:0] = {payload[63:0], header[7:0]}
```

For a 128-bit *Signal Group* configuration:

```
GRP_WIDTH = 128
SRAM data[135:0] = {payload[127:0], header[7:0]}
```

For a 256-bit *Signal Group* configuration:

```
GRP_WIDTH = 256
SRAM data[263:0] = {payload[255:0], header[7:0]}
```

The following table shows the header byte format.

Table 2-1 Header byte format

Bits	Name	Function
[7:6]	Trace counter[1:0]	Two selected bits from a 16-bit cycle counter in the trace unit, used to identify a fine-grain time associated with the trace capture. The Timestamp Control Register controls the selection of the counter bits that are used.
[5]	Trace data overwrite or overflow	When ST_FIFO_DEPTH = 0, identifies that <i>Trigger State</i> NUM_TRIG_STATES -1 has overwritten data that was being written at the same time from a different <i>Trigger State</i> , that is <i>Trigger States</i> 0-(NUM_TRIG_STATES -2). When ST_FIFO_DEPTH != 0, this bit indicates that there was a simultaneous trace FIFO overrun for the <i>Trigger State</i> bits[4:2] of the header. Requires that TSSR.ALTTTS[NUM_TRIG_STATES -1] = 1.
[4:2]	Trigger state	Current <i>Trigger State</i> . Software can use the <i>Trigger State</i> to determine which SIGNALGRP<n> was traced, by reading SIGSEL<n>.
[1:0]	Type	Returns the type of data that follows the header byte: <ul style="list-style-type: none"> 0b00 Counter trace data. For this packet type, the <i>Trigger State</i> bits represent the counter trace position in CNTSEL. 0b01 A 64-bit, 128-bit, or 256-bit data payload follows the header based on the GRP_WIDTH parameter value. 0b10 A timestamp value follows the header. <p style="text-align: center;">————— Note —————</p> <p>Timestamps are optional and can be enabled using the Timestamp Control Register. A timestamp payload contains the full 64-bit timestamp value. If the ELA-600 is configured with GRP_WIDTH > 64, the payload is zero-extended above the 64-bit timestamp value.</p> <ul style="list-style-type: none"> 0b11 Reserved.

Related references

[3.4.2 Timestamp Control register on page 3-48.](#)

2.4.5 Counter trace

Counter trace is used to measure the cycle time between signal comparisons.

Use this to measure the latency between events, such as request and response, or to measure events that require a greater resolution than available from the timestamp interface. The programming steps are:

Procedure

1. Program TRIGCTRLn of the first trigger state "n" to use either a signal or counter comparison that triggers a move to the next trigger state based on the event that will be used to start the measurement. The NEXTSTATEn or ALTNEXTSTATEn needs to be programmed to move to a different trigger state that will enable the counter and trace the value based on a signal comparison.
2. Program the CNTSEL register with the number of the second trigger state in one of the four counter positions to generate a counter trace. Otherwise a SIGNALGRP trace will be written.
3. The second trigger state "m" needs to be programmed so that:
 - a. TRIGCTRLm.COUNTSRC = 0 cycle count.
 - b. COUNTCOMPm = 0xFFFFFFFF, which allows the counter to count to its maximum value before stopping.
 - c. TRIGCTRLm.COMP or TRIGCTRLm.ALTCOMP need to be set for a signal comparison that will trigger the counter trace.
 - d. TRIGCTRLm.TRACE = 0b11 enables counter trace.

Note

The counter continues to run if TRIGCTRLm.COUNTCLR = 0 when advancing to the third trigger state after the signal comparison. A third trigger state can be used to trace the same counter if its number is in one of the four CNTSEL counter positions.

2.4.6 ATB trace format

The ATB trace format supports compression by removing zero bytes from the uncompressed raw data and uses a form of delta or difference compression to create additional zero value bytes which can be removed to further reduce the amount of trace data.

In the format, a zero byte value is used for an ASYNC so that trace decompression software can find a start or end point in the memory used for trace capture. A zero byte value cannot appear as a header or data payload byte, so two consecutive zero byte values in a trace memory are not valid. Each trace packet has the following format (with options in "[]" MSB to LSB { [Payload bytes], [Zero Byte identifier 1], [Zero Byte identifier 0], Header byte, [ASYNC]}):

- Payload bytes are not included when all the trace data are zero value bytes from raw data or after compression for data packet types.
- Zero Byte identifier 1 is not included with 64-bit trace (GRP_WIDTH = 64), or when there are no zero value bytes in the upper 8 bytes of 128-bit trace. Header bit 7 indicates the presence of zero byte values in the upper 8 bytes.
- Zero Byte identifier 0 is not included when there are no zero value bytes in the lower 8 bytes of the trace data. Header bit 6 indicates the presence of zero byte values in the lower 8 bytes of trace data.

Examples:

128-bit trace with all zeros in a data packet:

{0xFF, 0xFF, 0xC1} where 0xC1 is the header byte and 0xFF are the Zero Byte Identifiers with each bit set representing the zero byte position within the data payload.

64-bit trace with all zeros in the data packet:

{0xFF, 0x41} where 0x41 is the header and 0xFF is the Zero Byte position identifier.

Difference compression can further increase the number of zero bytes and therefore reduce the number of data bytes in the payload. Difference compression is only supported with data packets, but counter or timestamp packets still have the zero bytes removed from the raw data payload. Compression captures a data payload as a difference basis, then XORs subsequent data packets with the basis. A basis packet is always preceded with an ASYNC when compression is enabled by setting ATBCTRL.INTERVAL to a nonzero value.

For example:

```

Uncompressed trace data 0 = 0x12_F1_99_77_00_55_00_6C
Uncompressed trace data 1 = 0x12_F1_99_66_00_54_01_70

Trace data 0 header = 0x41
Trace data 0 Zero Byte position identifier = 0x0A
Trace data 0 with zero value bytes removed, zero byte position identifier and header:
0x12_F1_99_77_55_6C_0C_41

Trace data 1 header = 0x41
Trace data 1 Zero Byte position identifier = 0xE8
Difference calculation:
Basis = 0x12_F1_99_77_00_55_00_6C
Difference = 0x12_F1_99_66_00_54_01_70
            XOR 0x12_F1_99_77_00_55_00_6C
            = 0x00_00_00_11_00_01_01_1C

Compressed Trace data 1 with zero value removed, zero byte position identifier and header:
0x11_01_01_1C_E8_41

ATB Data which includes ASYNC to identify difference basis:
0x11_01_01_1C_E8_41_12_F1_99_77_55_6C_0C_41_00
    
```

In this example, ATBYTESM = 0xE for a 15 byte transfer. A transfer less than 16 bytes only occurs when there is an ATB flush with AFVALIDM, or CTRL.RUN = 0 was written, which stops trace and causes an automatic flush of the remaining bytes in FIFOs.

When trace overruns occur in the ATB byte packing FIFO, a trace packet is lost. Header bit 5 indicates that an overrun occurred earlier. If compression is enabled in **ATBCTRL[7:0]**, then there is a chance that the basis was lost. If the overrun bit is set without a preceding ASYNC and the trace stream changes to a different trigger state, then the basis packet was dropped. If the trace stream remains the same without a preceding ASYNC, then the data may still be using the same basis. Overruns in the packing FIFO cause a basis update in the compression so that a new basis is sent as early as possible. Header bit 5 is also set when ST_FIFO_DEPTH !=0 and there is a simultaneous trace FIFO overrun.

Note

Decompression software determines trace data packet payload size by reading DEVID1.SIGGRPWIDTH. Use of trace compression can be determined by reading ATBCTRL.INTERVAL.

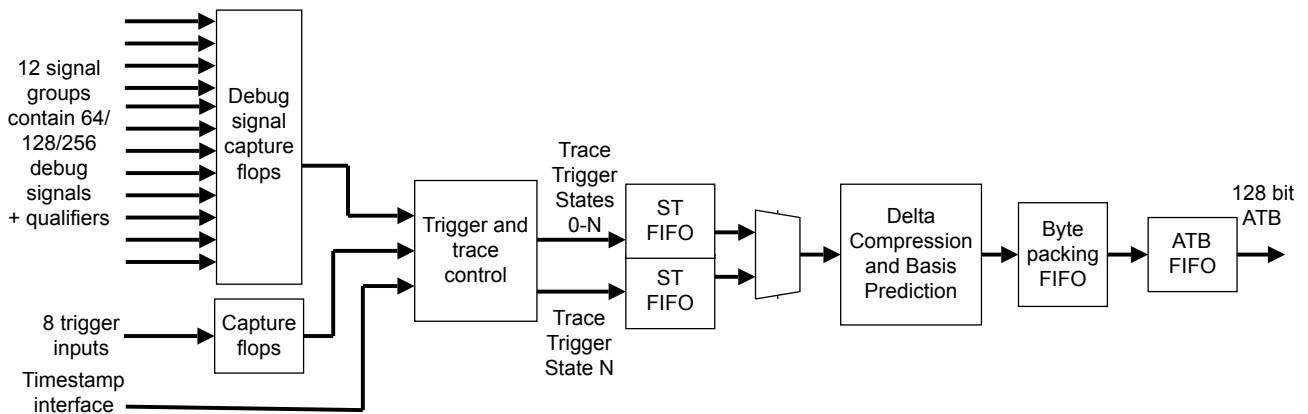
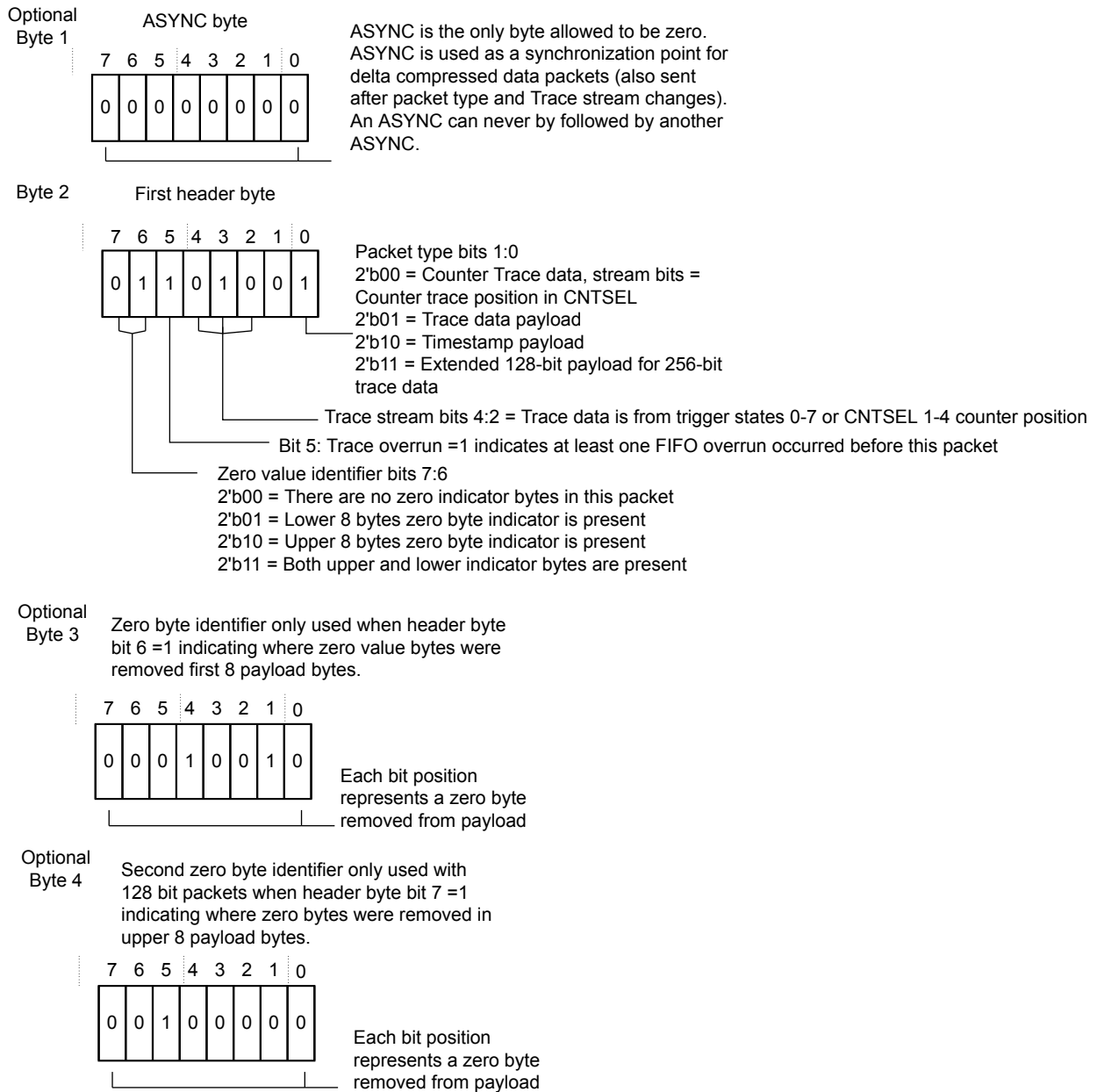


Figure 2-5 ELA-600 ATB trace data path



Payload Bytes : LSB least significant byte first to MSB. There will not be a payload if all bytes are zero.

Data Payload contains has zero value bytes removed based on identifier byte position bits
Packet type signal trace data, timestamp or counter trace data:

Note that header bytes support a maximum 128 bit data payload. 256 bit traces greater will require a second packet.

Figure 2-6 ELA-600 ELA trace format

2.4.7 Timestamp control

Timestamps enable correlation of ELA-600 trace with trace from other CoreSight trace sources.

Timestamps in the ELA-600 have the following features:

- The Timestamp Control Register is used to enable writing of timestamps into the trace SRAM.
- Timestamps plus the associated header byte occupy 72, 136, or 264 bits of data. If the ELA-600 is configured with `GRP_WIDTH > 64`, the timestamp value is zero-extended to 128 or 256 bits.
- Trace filtering that does not capture debug signal data on every **ELACLK** cycle enables timestamps to be written into the trace SRAM configuration trace header byte based on the interval set in the Timestamp Control Register. When the programmed timestamp interval is reached, a request is generated to insert a timestamp in the next available cycle that does not have a debug signal trace capture.
- Timestamps can be written into the trace SRAM or over the ATB interface after the trace active action is deasserted when `TIMECTRL.TSINT = 0`. This guarantees that at least one timestamp is present in the circular SRAM buffer.
- When `CTRL.RUN` is cleared, a timestamp is written if the previous trace write contained a data payload.

2.4.8 Debug APB registers and interface to SRAM

When configured with `TRACE_GEN = 1`, the SRAM is accessible through the debug APB registers.

The SRAM is 72 bits, 136 bits, or 264 bits wide, depending on the *Signal Group* width configuration parameter `GRP_WIDTH`.

Four registers enable the SRAM to be accessed through the 32-bit debug APB interface:

- *RAM Read Address Register* (RRAR).
- *RAM Read Data Register* (RRDR).
- *RAM Write Address Register* (RWAR).
- *RAM Write Data Register* (RWDR).

The RAM Read registers are provided to enable a debugger to read out captured trace data from the ELA-600. The RAM Write registers are provided to support integration testing.

The RRAR and RWAR address single 72-bit, 136-bit, or 264-bit words within the SRAM. Multiple RRDR or RWDR accesses are required for each SRAM word. An internal holding register is used to transfer data between the SRAM and RAM Data registers.

SRAM reads

When the RRAR is updated, either by a debug APB write or by an automatic increment, the SRAM data at that address is copied to the holding register.

Reads to the RRDR return the data from the holding register. The first read of the RRDR after an RRAR update returns the trace data header byte value, zero-extended to 32-bits. Subsequent reads of the RRDR return 32-bit chunks of the trace data payload, starting with the least significant chunk. This continues until all the payload data has been read, that is, two chunks if `GRP_WIDTH = 64`, four chunks if `GRP_WIDTH = 128`, and eight chunks if `GRP_WIDTH = 256`.

When the final 32 bits of the payload have been read, the RRAR is incremented automatically. Then, the next word of SRAM data is copied into the holding register. This enables the SRAM data content to be read out efficiently.

The RRAR wraps to address zero if it is incremented beyond the maximum depth of the SRAM.

The trace read data scrambler protects the exposure of designs that are sensitive to signal trace. One or more signal groups might be scrambled. You must know the details of the specific scrambling system that was used to be able to unscramble the trace data.

For more information, see *Arm® CoreSight™ ELA-600 Embedded Logic Analyzer Configuration and Integration Manual* (ARM 101089).

SRAM writes

Writes to the SRAM are supported for integration-testing purposes.

A write to the RWAR sets the SRAM address for the data that is then written to the RWDR. Writes to the RWDR update the internal holding register.

The first write to the RWDR sets the header byte value with the least significant byte written. Subsequent writes to the RWDR set 32-bit chunks of the payload, starting with the least significant chunk. When the final 32 bits of the payload have been written, the content of the holding register is copied into the SRAM and the RWAR is incremented automatically.

The RWAR wraps to address zero if it is incremented beyond the maximum depth of the SRAM.

2.5 Triggering

Triggering is the process of causing an *Output Action* signal to be driven, or advancing to the next *Trigger State*, when a *Trigger Signal Comparison* or a *Trigger Counter Comparison* match occurs.

Trigger Signal Comparisons are based on the comparison of *Signal Groups* and *External Trigger Input Signals*. The masked *Signal Group* values and target values in the *Signal Compare* (SIGCOMP<n>) registers are compared, and logically ANDed, with the comparison of the masked *External Trigger Input Signals* and target values in the *External Compare* register (EXTCOMP<n>), where $n = 0-7$ and denotes one of the eight *Trigger States*. The *Trigger Condition* is only met if both the *Signal Group* and *External Trigger Input Signal* comparisons succeed.

Trigger state comparators are sized the same as GRP_WIDTH. Also each trigger state comparator can be used as multiple 32-bit comparators using the COMPCTRLn and ALTCOMPCTRLn registers. Each 32-bit comparator can be removed from the TRIGCTRLn comparison and programmed with a different comparison in the COMPCTRLn and ALTCOMPCTRLn registers.

The mask registers can also support *Trigger Conditions* that only use a single set of signals. They are:

Debug signals only

This is achieved by masking out the eight *External Trigger Input Signals* and SIGQUAL signals, and writing the External Compare and QUALCOMP registers with zeros. This causes an always true condition.

External Trigger Input Signals only

This is achieved by masking out all the debug and SIGQUAL signals in the *Signal Group* and writing zeros into the Signal Compare and QUALCOMP registers.

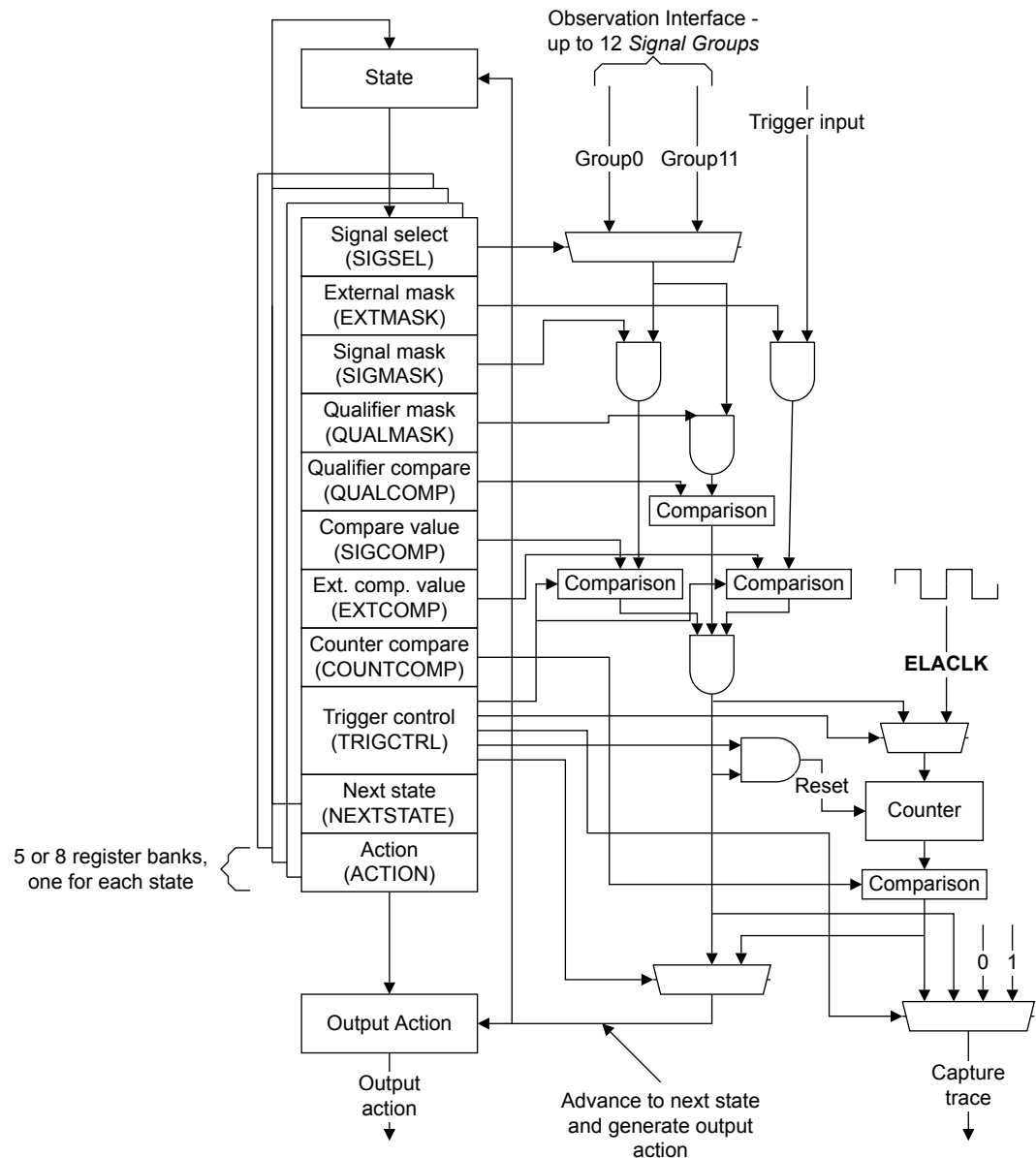
Qualifier Input Signals only

This is achieved by masking out all the debug signals in the *Signal Group* and *External Compare registers* and writing zeros into the Signal Compare and External Compare registers.

Trigger Counter Comparisons are made when the *Trigger State* counter counts from zero to the target value set in the *Counter Compare* register (COUNTCOMP<n>). A *Trigger Condition* can only be caused by either a *Trigger Signal Comparison* or a *Trigger Counter Comparison*.

Trigger States can be placed into loops by programming the NEXTSTATE<n> register to move to a previous *Trigger State*. *Trigger State* loops can be useful for trace filtering that is based on repeated *Trigger Signal Comparison* and *Trigger Counter Comparison* conditions. There is a *Trigger State* loop counter capability that can be enabled by programming TRIGCTRL.COUNTBRK = 1 and TRIGCTRL.COUNTCLR = 0. The loop counter can be used to break the loop and stop trace before the SRAM is full, or after a count of *Trigger Signal Comparison* events or a count of cycles. The *Trigger State* loop counter can also be used to control the number of ACTIONS by toggling ELA outputs **ELAOUTPUT** and **CTTRIGOUT**, which can be connected as interrupts or other CTI events.

The following figure shows the triggering mechanism within the ELA-600.



Note:
Comparison represents a comparison function which can be one of:
 =, >, >=, !=, <, <=.

Figure 2-7 Triggering mechanism within the ELA-600

This section contains the following subsections:

- [2.5.1 Conditional trigger states](#) on page 2-38.
- [2.5.2 Transaction ID capture](#) on page 2-40.

2.5.1 Conditional trigger states

Conditional *Trigger States* adds an optional second trigger signal comparison to each *Trigger State*.

DEVID[19:16] = 1 indicates to software that the ELA-600 supports conditional *Trigger States*.

The following pseudocode examples show *Trigger State 0* comparisons using conditional *Trigger State* support, which adds the options that are highlighted in *italic text*:

Example 1: Both *Trigger State* conditions are programmed for trigger signal comparisons.

```
IF ((SIGNALGRP<n> < SIGCOMP0) && (TRIGCTRL0.COMPSEL == 0))
    drive ACTION0 and goto NEXTSTATE0
ELSE IF ((SIGNALGRP<n> > SIGCOMP0) && (TRIGCTRL0.ALTCOMPSEL == 0))
    drive ALTACTION0 and goto ALTNEXTSTATE0
ELSE do nothing
```

Example 2: The first condition is programmed for trigger signal comparison, and the alternative condition is programmed for counter cycle count comparison. The event count can be programmed for the first IF condition.

```
IF ((SIGNALGRP<n> < SIGCOMP0) && (TRIGCTRL0.COMPSEL == 0))
    drive ACTION0 and goto NEXTSTATE0
ELSE IF ((counter == COUNTCOMP0) && (TRIGCTRL0.ALTCOMPSEL == 1) && (TRIGCTRL0.COUNTSRC == 0))
    drive ALTACTION0 and goto ALTNEXTSTATE0
ELSE do nothing
```

Example 3: The first condition is programmed for signal comparison, and the alternative condition is programmed for counter comparison using loop counter COUNTBRK.

```
IF ((SIGNALGRP<n> < SIGCOMP0) && (TRIGCTRL0.COMPSEL == 0))
    drive ACTION0 and goto NEXTSTATE0
ELSE IF ((counter == COUNTCOMP0) && (TRIGCTRL0.ALTCOMPSEL == 1) && (TRIGCTRL0.COUNTBRK == 1))
    IF((counter_match != 1) && (event_or_cycle_count_increment))
        goto ALTNEXTSTATE0
    ELSE IF (counter_match == 1)
        goto final_state
ELSE // COUNTBRK will default to the first IF condition branch
    goto NEXTSTATE0
```

Example 4: The first condition is programmed for counter comparison, and the alternative condition is programmed for signal comparison. In this case, the first condition, the counter match, dominates when there is a simultaneous signal comparison match.

```
IF ((counter == COUNTCOMP0) && (TRIGCTRL0.ALTCOMPSEL == 1) && (TRIGCTRL0.COUNTSRC == 0))
    drive ACTION0 and goto NEXTSTATE0
ELSE IF ((SIGNALGRP<n> < SIGCOMP0) && (TRIGCTRL0.COMPSEL == 0))
    drive ALTACTION0 and goto ALTNEXTSTATE0
ELSE do nothing
```

Example 5: The first condition is programmed for counter comparison using loop counter COUNTBRK, and the alternative condition is programmed for signal comparison.

```
IF ((counter == COUNTCOMP0) && (TRIGCTRL0.COMPSEL == 1) && (TRIGCTRL0.COUNTBRK == 1))
    IF ((counter_match != 1) && event_or_cycle_count_increment)
        goto NEXTSTATE0
    ELSE IF (counter_match == 1)
        goto final_state
ELSE IF ((SIGNALGRP<n> < SIGCOMP0) && (TRIGCTRL0.ALTCOMPSEL == 0))
    drive ALTACTION0 and goto ALTNEXTSTATE0
ELSE // COUNTBRK will default to the first IF condition branch
    goto NEXTSTATE0
```

Example 6: Counting trigger signal comparisons as events does not need to use an alternative condition.

```
Trigger State 0:
Signal Comparison:
If (SIGNALGRP<n> == SIGCOMP0)
    trace SIGNALGRP

Counter Comparison: // use the counter to count Signal Comparison events
If (counter == COUNTCOMP0) && (TRIGCTRL0.COMPSEL == 1) && (TRIGCTRL0.COUNTSRC == 1) &&
(TRIGCTRL0.COMP == 3'b001)
    goto final_state
ELSE IF (SIGNALGRP<n> == SIGCOMP0)
    counter = counter + 1
ELSE do nothing
```

————— Note —————

A single state comparator can be used as separate 32-bit comparisons by programming COMPCTRL and ALTCOMPCTRL. For example, 32-bit addresses on buses A and B can be separately compared as IF ((ADDRESS_A == value_x) || (ADDRESS_B == value_y))

Filtered trace that is based on signal comparisons or counter comparisons is still independently controlled by TRIGCTRL<n>.TRACE.

If both the first IF and alternative ELSE IF trigger conditions match on the same clock cycle comparison, the first IF condition dominates and drives ACTION0 and goes to NEXTSTATE0.

TRIGCTRLn.COUNTBRK = 1 loops to the primary IF condition branch if the primary IF or alternative ELSE IF conditions do not match in the comparison clock cycle.

Two additional registers are used to provide support for the conditional *Trigger States*.

Related references

[3.10.5 Alt Next State registers on page 3-72.](#)

[3.10.6 Alt Action registers on page 3-73.](#)

2.5.2 Transaction ID capture

Captures a transaction ID when there is a trigger signal match of an address request in a *Trigger State*.

The ID bits are based on debug signal wiring to the LSB bits of the SIGNALGRP and the parameter ID_CAPTURE_SIZE which identifies the MSB of the ID, such that transaction_ID[x:0] = SIGNALGRP<n>[x:0].

Stacked IDs can be wired to a SIGNALGRP such that {transaction_ID1[y:x+1], transaction_ID0[x:0]} = SIGNALGRP<n>[y:0].

SIGMASK<n> can be used to select the transaction ID in a subsequent trigger state.

The following figure illustrates a transaction ID capture from a read address channel on the first *Trigger State*, and use of the ID to compare with other debug signals connected to the SIGNALGRP<n> port in the second *Trigger State*.

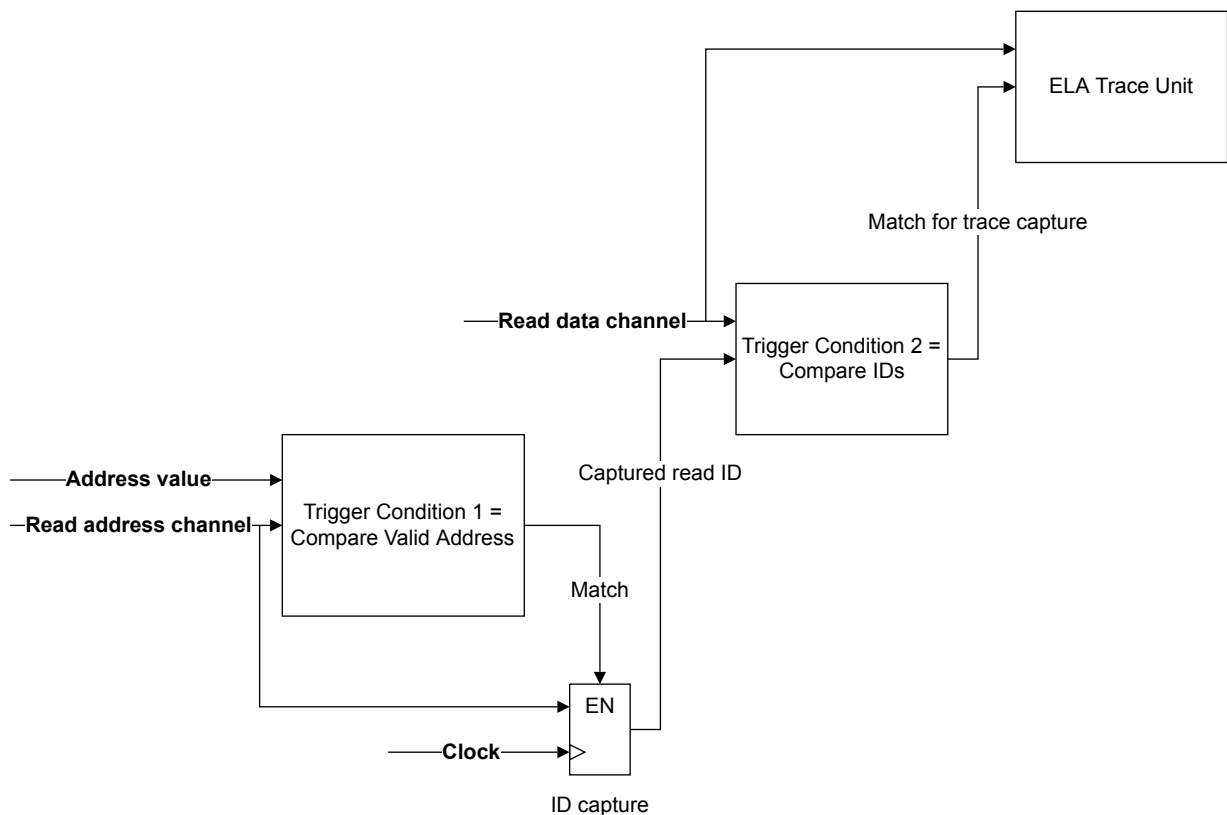


Figure 2-8 Transaction ID capture from read address channel

The parameters `ID_CAPTURE_GEN = 1` and `COND_TRIG = 1` must be set to enable generation of the ID capture feature. The `ID_CAPTURE_SIZE` parameter must be set to the number of bits in the ID tag.

If `DEVID1` bits[24:20] = 0 then it indicates that the Transaction ID capture feature is disabled, otherwise `DEVID1` bits[24:20] are set to `ID_CAPTURE_SIZE`. The parameter `ID_CAPTURE_SIZE` sets the size of the ID capture register which can be 2-30 bits.

Related references

[3.16.4 Device Configuration register on page 3-90.](#)

2.6 Authentication interface

The ELA-600 supports the authentication signals **DBGEN**, **NIDEN**, **SPIDEN**, and **SPNIDEN** through the Authentication interface.

When the ELA-600 is configured for Secure visibility:

- The ELA-600 *Trigger State* and trace operation are enabled when Secure non-invasive debug is enabled. When Secure non-invasive debug is disabled, the ELA-600 is stopped and does not move between *Trigger States*, assert any outputs, or capture any trace.
————— **Note** —————
 - The ELA-600 does not operate if authentication signals are set to Non-secure debug mode.
 - The CoreSight Authentication rules require that Non-secure debug is enabled for Secure debug to be enabled.
- The **STOPCLOCK** output is only asserted when Secure invasive debug is enabled.
- If Secure non-invasive debug becomes disabled dynamically, CTRL.RUN is internally gated, disabling the ELA-600. APB reads of CTRL.RUN do not indicate the state of the internally-gated run signal.

For more information on the Authentication interface, including the permitted values of the Authentication signals, and the CoreSight debug states, see the *Arm® CoreSight™ Architecture Specification*.

2.7 Parameter summary

The functionality of the ELA-600 is determined by configurable parameters.

The following table shows the parameters that control the configuration of the ELA-600:

Table 2-2 ELA-600 configuration parameters

Parameter	Values	Default	Description
GRP_WIDTH	64, 128, or 256	128	64, 128, or 256 debug signals are allowed in a <i>Signal Group</i> . The number of SIGMASK and SIGCOMP registers for each trigger state are extended based on GRP_WIDTH/32. The ATB configuration does not support GRP_WIDTH = 256.
RAM_ADDR_SIZE	2-25 inclusive	9	Number of address bits in the SRAM. This parameter is only available on the SRAM configuration. Number of entries in the SRAM = $2^{\text{RAM_ADDR_SIZE}}$ For example, when RAM_ADDR_SIZE is 9, the SRAM has 512 entries. Arm recommends at least 512 entries, where an entry is data payload plus the header byte. When GRP_WIDTH = 64, each entry is 9 bytes. When GRP_WIDTH = 128, each entry is 17 bytes. When GRP_WIDTH = 256, each entry is 33 bytes.
ID_CAPTURE_SIZE	2-30	10	Sets the number of bits used for the captured transaction ID.
ID_CAPTURE_GEN	0 or 1	1	0: Captured transaction ID is not generated. 1: Captured transaction ID is generated. ————— Note ————— When ID_CAPTURE_GEN = 1, you must also set COND_TRIG = 1.
NUM_TRIG_STATES	5 or 8	5	Sets the number of <i>Trigger States</i> .
TRIGIN_EDGE	0 or 1	1	Used to determine the method by which CTTRIGIN and EXTRIG are detected. When TRIGIN_EDGE = 0, CTTRIGIN and EXTRIG are sampled when the <i>Trigger State</i> signal comparison occurs. When TRIGIN_EDGE = 1, assertion of CTTRIGIN and EXTRIG are detected if CTRL.RUN = 1, and their assertions are latched until CTRL.RUN = 0.
ATB_FIFO_DEPTH	4 or 8	4	Sets the ATB interface FIFO Depth based on the ATREADY response latency from the trace infrastructure. This is only available on an ATB configuration.
ST_FIFO_DEPTH	0, 4, or 8	0	Sets the depth of the simultaneous trace FIFOs for the TSSR trigger state and the remaining trigger states. A value of 0 does not generate a FIFO, so a trace that occurs on the same cycle for not trigger states writes the TSSR FIFO.

Chapter 3

Programmers model

This chapter describes the programmers model.

Note

Any register bit position that is not listed in the description tables is reserved.

It contains the following sections:

- [3.1 Access permissions](#) on page 3-45.
- [3.2 Programming sequence](#) on page 3-46.
- [3.3 Control register summary](#) on page 3-47.
- [3.4 Control register descriptions](#) on page 3-48.
- [3.5 Current State register summary](#) on page 3-53.
- [3.6 Current State register descriptions](#) on page 3-54.
- [3.7 RAM register summary](#) on page 3-56.
- [3.8 RAM register descriptions](#) on page 3-57.
- [3.9 Trigger State register summary](#) on page 3-60.
- [3.10 Trigger State register descriptions](#) on page 3-68.
- [3.11 Integration Mode register summary](#) on page 3-81.
- [3.12 Integration Mode register descriptions](#) on page 3-82.
- [3.13 Authentication register summary](#) on page 3-86.
- [3.14 Authentication register descriptions](#) on page 3-87.
- [3.15 Device register summary](#) on page 3-88.
- [3.16 Device register descriptions](#) on page 3-89.
- [3.17 ID register summary](#) on page 3-92.
- [3.18 ID register descriptions](#) on page 3-93.

3.1 Access permissions

Individual registers and register groups have different availability, depending on whether the ELA-600 is running or not.

The following table lists the software access permissions by register group or individual register where appropriate.

Table 3-1 Register access permissions

Register or register group	Access when CTRL.RUN=1 ^a	Access when CTRL.RUN=0 ^a
<i>Logic Analyzer Control register on page 3-48</i>	Can be accessed by software.	Can be accessed by software.
<i>Current state registers on page 3-53</i>		
<i>Authentication registers on page 3-86</i>		
<i>Device registers on page 3-88</i>		
<i>ID registers on page 3-92</i>		
<i>RAM registers on page 3-56</i>	Must not be accessed by software.	
<i>Integration mode registers on page 3-81</i>		
<i>Trigger state registers on page 3-60</i>	Can be read by software. Must not be written.	
<i>Timestamp Control register on page 3-48</i>		
<i>Pre-trigger Action register on page 3-50</i>		

^a Access means reads or writes, according to the Type field in the relevant register summary table.

3.2 Programming sequence

Programming the ELA-600 requires you to perform several steps.

Note

- In the following sequence of steps, a lowercase $\langle n \rangle$, where $n = 0$ to 7 , denotes one of the eight *Trigger States*.
 - The order of steps 2-8 is not important.
 - For ELA compatibility, initialize all registers to zero before programming registers with ELA-500 settings.
-

Procedure

1. Set CTRL.RUN = 0 to stop the logic analyzer and allow the *Trigger State* and conditions to be programmed.
2. Select the bus to be used for comparisons for each *Trigger State* by writing to the appropriate SIGSEL $\langle n \rangle$ register.
3. Program the mask and compare registers for each *Trigger State* by writing to SIGMASK $\langle n \rangle$ and SIGCOMP $\langle n \rangle$, respectively.
4. Program the *Output Action* for each *Trigger State* by writing to the ACTION $\langle n \rangle$ registers. Program an initial *Output Action* by writing to the PTACTION register.
Trigger State sequences and actions result in either a level or a pulse on the logic analyzer output signals, depending on the value in the *Output Action* registers for subsequent *Trigger States*.
5. Program the next *Trigger State* sequence in the NEXTSTATE $\langle n \rangle$ registers.
Trigger State 0 is the first enabled state after reset.
6. If the counters are used for a *Trigger State*, then write the compare value for each *Trigger State* counter to COUNTCOMP $\langle n \rangle$.
7. Program the Trigger Control, COMPCTRL n , and ALTCOMPCTRL n registers to select the comparison type and counter options for each *Trigger State* by writing to TRIGCTRL $\langle n \rangle$.
8. Write to the RWAR to set the first RAM address to be written and to clear the WRAP bit for the SRAM configuration. If configured for ATB trace, then the TWBSEL n registers must have a "1" written for each SIGNALGRP n byte that will be traced. At least one byte needs to be traced.
9. Set CTRL.RUN = 1 to enable the ELA-600.

The Current *Trigger State*, Counter, and Actions registers can be read when CTRL.RUN = 0 or 1.

Related references

[Chapter 3 Programmers model on page 3-44.](#)

3.3 Control register summary

This section gives a summary of the ELA-600 Control registers.

The following table shows the Control registers in offset order from the base address of the ELA-600.

Table 3-2 Control registers summary

Offset	Name	Type	Reset	Description
0x000	CTRL	RW	0x00000000	3.4.1 Logic Analyzer Control register on page 3-48
0x004	TIMECTRL	RW	0x00000000 ^b	3.4.2 Timestamp Control register on page 3-48
0x008	TSSR	RW	0x00000000	3.4.3 Trigger State Select Register on page 3-49
0x00C	ATBCTRL	RW	0xFFFFFFFF	3.4.4 ATB Control Register on page 3-49 The ATB Control register is only available in the ATB configuration.
0x010	PTACTION	RW	0x00000000	3.4.5 Pre-trigger Action register on page 3-50
0x014	AUXCTRL	RW	0x00000000	3.4.6 Auxiliary Control register on page 3-51 The Auxiliary Control register is only available in the ATB configuration.
0x018	CNTSEL	RW	0xFFFFFFFF	3.4.7 Counter Select register on page 3-51 The Counter Select register is for counter trace control.

^b Must be initialized by software before writing CTRL.RUN=1.

3.4 Control register descriptions

This section describes the ELA-600 Control registers.

[Table 3-2 Control registers summary on page 3-47](#) provides cross-references to individual registers.

This section contains the following subsections:

- [3.4.1 Logic Analyzer Control register on page 3-48.](#)
- [3.4.2 Timestamp Control register on page 3-48.](#)
- [3.4.3 Trigger State Select Register on page 3-49.](#)
- [3.4.4 ATB Control Register on page 3-49.](#)
- [3.4.5 Pre-trigger Action register on page 3-50.](#)
- [3.4.6 Auxiliary Control register on page 3-51.](#)
- [3.4.7 Counter Select register on page 3-51.](#)

3.4.1 Logic Analyzer Control register

The ELA-600 Logic Analyzer Control register enables and disables the ELA-600.

The CTRL register characteristics are:

- Usage constraints** No usage constraints.
- Configurations** Available in all configurations.
- Attributes** See [3.3 Control register summary on page 3-47.](#)

The following table shows the bit assignments.

Table 3-3 CTRL register bit assignments

Bits	Name	Function
[1]	TRACE_BUSY	<p>Read only. Reads the trace busy status. Needs to be polled after CTRL.RUN = 0 is written and before any other APB registers are written to insure that trace has stopped after draining the simultaneous trace FIFO (ST_FIFO_DEPTH != 0), and the ATB FIFO has drained when configured for ATB trace.</p> <p>————— Note —————</p> <p>TRACE_BUSY does not need to be read when RUN = 1.</p> <hr/> <p>0> If both TRACE_BUSY and RUN = 0, then ELA-600 registers can be written.</p> <p>1 Trace is busy and ELA-600 registers cannot be written.</p>
[0]	RUN	<p>Run control.</p> <p>0 ELA-600 disabled. Register programming permitted when TRACE_BUSY = 0.</p> <p>1 ELA-600 enabled.</p>

3.4.2 Timestamp Control register

The ELA-600 Timestamp Control register enables insertion of timestamps in trace, programming of the timestamp request interval, and determination of which 2 bits from the 16 trace counter bits are written to the upper 2 bits of the trace header byte.

The TIMECTRL register characteristics are:

- Usage constraints** Writing when CTRL.RUN = 1 results in improper operation. Registers cannot be written when CTRL.TRACE_BUSY = 1 when configured for SRAM or ATB trace.
- Configurations** Available in the SRAM and ATB configurations.

Attributes See [3.3 Control register summary on page 3-47](#).

The following table shows the bit assignments.

Table 3-4 TIMECTRL register bit assignments

Bits	Name	Function
[16]	TSEN	Timestamp Enable.
[15:12]	TSINT	Timestamp Interval. When Timestamps are enabled, TSINT specifies the bit number of the 16-bit trace counter that causes a timestamp packet to be requested. The trace counter runs from ELACLK . When the specified bit changes, a timestamp packet is requested to be inserted into the trace SRAM when there is an ELACLK cycle during which trace data is not being captured. The ELA-600 does not insert back-to-back timestamps in the SRAM, even when TSINT causes multiple requests to be made. When TSINT = 0, a timestamp is written when ACTION.TRACE disables trace. Looping <i>Trigger States</i> enable and then disable trace, causing timestamp writes. A timestamp is always written when CTRL.RUN is cleared and the previous trace write contained a data payload.
[11:8]	Reserved	-
[7:4]	TCSEL1	Trace Counter 1 select. Only available in the SRAM configuration. Selects the bit number of the 16-bit trace counter that is presented as Trace Counter[1] in the SRAM header byte.
[3:0]	TCSEL0	Trace Counter 0 select. Only available in the SRAM configuration. Selects the bit number of the 16-bit trace counter that is presented as Trace Counter[0] in the SRAM header byte.

Related references

[2.4.4 SRAM trace format on page 2-30](#).

3.4.3 Trigger State Select Register

The Trigger State Select Register enables and disables independent trace for *Trigger State 4* or *7*.

The TSSR characteristics are:

Usage constraints Registers cannot be written when CTRL.TRACE_BUSY = 1 when configured for SRAM or ATB trace.

Configurations Only available with the SRAM and ATB configurations.

Attributes See [3.3 Control register summary on page 3-47](#).

The following table shows the bit assignments.

Table 3-5 TSSR register bit assignments

Bits	Name	Function
[7:0]	ALTTS	Each bit identifies the trigger state that enables independent trace. Only <i>Trigger State</i> NUM_TRIG_STATES -1 supports independent trace. ALTTS[NUM_TRIG_STATES -1]=0 <i>Trigger State</i> NUM_TRIG_STATES -1 independent trace disabled. ALTTS[NUM_TRIG_STATES -1]=1 <i>Trigger State</i> NUM_TRIG_STATES -1 independent trace enabled. All other bits read zero.

3.4.4 ATB Control Register

ATBCTRL controls ATB compression, ATID value, and ATID trigger.

The ATBCTRL register characteristics are:

- Usage constraints** Writing when CTRL.RUN = 1 results in improper operation. Registers cannot be written when CTRL.TRACE_BUSY = 1 when configured for SRAM or ATB trace.
- Configurations** Available only with the ATB configuration.
- Attributes** See [3.3 Control register summary on page 3-47](#).

The following table shows the bit assignments.

Table 3-6 ATB register bit assignments

Bits	Name	Function
[31]	PREDICT	Run control. 0 Prediction logic disabled. 1 Enable Delta basis changes using prediction logic. Prediction can reduce the number of trace bytes for repeated data that is generated from loops. The reset value is X.
[30:16]	-	Reserved, RAZ.
[15]	ATID_TRIG_EN	ATID trigger transaction. 0 Normal operation. 1 Causes an ATID = 0x7D trigger transaction with ATBYTES = 0 and the lower byte set to ATID_VALUE when a trigger state reaches Final State, trace has stopped and autoflush has transferred all FIFO contents over ATB. This trigger can be used to stop trace from other CoreSight trace sources that are connected to the Trace Memory Controller. See the Arm AMBA ATBv4 ATB Protocol Specification for more information about the ATID trigger operation.
[14:8]	ATID_VALUE	Reserved, RAZ.
[7:0]	ASYNC_INTERVAL	Sets the maximum interval between ASYNCS and also enables Delta compression. A nonzero ASYNC_INTERVAL value enables Delta compression and sets the interval for the maximum number of trace writes that can occur to a trace memory before an ASYNC and compression basis update is written. Set this to a value that assures that ASYNCS occur in a circular memory or to the largest value. The reset value is X.

3.4.5 Pre-trigger Action register

PTACTION sets a level on the Action outputs immediately after CTRL.RUN is set, and before the first *Trigger Condition* has been met.

The PTACTION register characteristics are:

- Usage constraints** Writing when CTRL.RUN = 1 results in improper operation. Registers cannot be written when CTRL.TRACE_BUSY = 1 when configured for SRAM or ATB trace.
- Configurations** Available in all configurations.
- Attributes** See [3.3 Control register summary on page 3-47](#).

The following table shows the bit assignments.

Table 3-7 PTACTION register bit assignments

Bits	Name	Function
[7:4]	ELAOUTPUT	Sets the value to drive on ELAOUTPUT[3:0].
[3]	TRACE	Enables trace.
[2]	STOPCLOCK	Sets the level to drive on STOPCLOCK.
[1:0]	CTTRIGOUT	Sets the value to drive on CTTRIGOUT[1:0].

3.4.6 Auxiliary Control register

AUXCTRL controls ATB flush.

The AUXCTRL register characteristics are:

Usage constraints This register is only supported when TRACE_GEN = 2. Registers cannot be written when CTRL.TRACE_BUSY = 1 when configured for SRAM or ATB trace.

Configurations Available in all configurations.

Attributes See [3.3 Control register summary on page 3-47](#).

The following table shows the bit assignments.

Table 3-8 AUXCTRL register bit assignments

Bits	Name	Function
[0]	FLUSH_DIS	0 ATB AFVALID flush requests enabled. This is the reset value. 1 Disables ATB AFVALID flush requests and as a result, AFREADY will always be set.

3.4.7 Counter Select register

The Counter Trace Select register selects the trigger state counters that are traced when TRIGCTRL.TRACE is programmed for counter trace.

The CNTSEL register characteristics are:

Usage constraints Writing when CTRL.RUN = 1 results in improper operation. Registers cannot be written when CTRL.TRACE_BUSY = 1 when configured for SRAM or ATB trace.

Configurations Available in both SRAM and ATB trace configurations.

Attributes See [3.3 Control register summary on page 3-47](#).

Selects the *Trigger State* 32-bit counter that is traced for each position in the trace packet for *Trigger States* 0-7.

————— **Note** —————

Trigger state 7, 6, and 5 are only available when NUM_TRIG_STATES = 8.

The trace packet header stream bits equal the position number, which can be used to determine the trigger state that caused the trace write.

————— **Note** —————

TRIGCTRLn.TRACE = 11 must be set to cause a counter write.

POSITION1 = [31:0]

POSITION2 = [63:32]

POSITION3 = [95:64]

POSITION4 = [127:96]

The following table shows the bit assignments.

Table 3-9 CNTSEL register bit assignments

Bits	Name	Function
[11:9]	-	Trigger state counter selected for POSITION4 when GRP_WIDTH > 64.
[8:6]	-	Trigger state counter selected for POSITION3 when GRP_WIDTH > 64.
[5:3]	-	Trigger state counter selected for POSITION2.
[2:0]	-	Trigger state counter selected for POSITION1.

3.5 Current State register summary

This section gives a summary of the ELA-600 Current State registers.

The following table shows the Current State registers in offset order from the base address of the ELA-600.

Table 3-10 Current State registers summary

Offset	Name	Type	Reset	Description
0x020	CTSR	RO	0b0001	3.6.1 Current Trigger State Register on page 3-54
0x024	CCVR	RO	0x00000000	3.6.2 Current Counter Value Register on page 3-54
0x028	CAVR	RO	0x00	3.6.3 Current Action Value Register on page 3-55
0x02C	RDCAPTID	RO	-	3.6.4 Read Captured Transaction ID register on page 3-55

3.6 Current State register descriptions

This section describes the ELA-600 Current State registers.

Table 3-10 Current State registers summary on page 3-53 provides cross-references to individual registers.

This section contains the following subsections:

- *3.6.1 Current Trigger State Register on page 3-54.*
- *3.6.2 Current Counter Value Register on page 3-54.*
- *3.6.3 Current Action Value Register on page 3-55.*
- *3.6.4 Read Captured Transaction ID register on page 3-55.*

3.6.1 Current Trigger State Register

The Current Trigger State Register takes a snapshot of the current *Trigger State*. When the CTSR is read, the current values of the *Current Counter Value Register (CCVR)* and *Current Action Value Register (CAVR)* are also captured.

The CTSR characteristics are:

- Usage constraints** No usage constraints.
- Configurations** Available in all configurations.
- Attributes** See *3.5 Current State register summary on page 3-53.*

The following table shows the bit assignments.

Table 3-11 CTSR bit assignments

Bits	Name	Function
[31]	FINALSTATE	<p>0 ELA-600 is still tracing.</p> <p>1 Indicates that the ELA-600 has stopped advancing <i>Trigger States</i> and stopped trace.</p> <p>FINALSTATE can be set by TRIGCTRL<n>.COUNTBRK reaching the final loop count, or by programming NEXTSTATE<n> or ALTNEXTSTATE<n> to zero.</p>
[30:NUM_TRIG_STATES]	Reserved	-
[NUM_TRIG_STATES-1:0]	CTSR	<p>Reads current <i>Trigger State</i>. This is a one-hot encoded field.</p> <p>When CTRL.RUN:</p> <p>0 RAZ.</p> <p>1 Returns current <i>Trigger State</i>.</p> <p>If FINALSTATE is 1, then the CTSR field gives the <i>Trigger State</i> when FINALSTATE became 1.</p>

3.6.2 Current Counter Value Register

The Current Counter Value Register returns the counter value that was captured when the *Current Trigger State Register (CTSR)* was read.

The CCVR characteristics are:

- Usage constraints** No usage constraints.
- Configurations** Available in all configurations.
- Attributes** See *3.5 Current State register summary on page 3-53.*

The following table shows the bit assignments.

Table 3-12 CCVR bit assignments

Bits	Name	Function
[31:0]	CCVR	Returns the counter value when the CTSR was last read. If the CTSR has never been read, then the value in the CCVR is UNDEFINED.

3.6.3 Current Action Value Register

The Current Action Value Register returns the Action value that was captured when the *Current Trigger State Register* (CTSR) was read.

The CAVR characteristics are:

Usage constraints No usage constraints.

Configurations Available in all configurations.

Attributes See [3.5 Current State register summary on page 3-53](#).

The following table shows the bit assignments.

Table 3-13 CAVR bit assignments

Bits	Name	Function
[7:4]	ELAOUTPUT	Value driven on ELAOUTPUT[3:0] .
[3]	TRACE	Trace active. 0b0 Trace is not active. 0b1 Trace is active.
[2]	STOPCLOCK	Level driven on STOPCLOCK . 0b0 0 Driven on STOPCLOCK . 0b1 1 Driven on STOPCLOCK .
[1:0]	CTTRIGOUT	Value driven on CTTRIGOUT[1:0] .

3.6.4 Read Captured Transaction ID register

The RDCAPTID register captures a transaction ID on the trigger signal match of an address request in a *Trigger State*. The ID can then be used by a subsequent *Trigger State* to trace the response to the original request.

The RDCAPTID register characteristics are:

Usage constraints Can only be read when CTRL.RUN = 0 and ID_CAPTURE_GEN = 1.

Configurations Available in all configurations.

Attributes See [3.5 Current State register summary on page 3-53](#).

The following table shows the bit assignments.

Table 3-14 RDCAPTID bit assignments

Bits	Name	Function
[ID_CAPTURE_SIZE-1:0]	RDCAPTID	Returns the captured transaction ID.

3.7 RAM register summary

This section gives a summary of the ELA-600 RAM registers.

The following table shows the RAM registers in offset order from the base address of the ELA-600.

Table 3-15 RAM registers summary

Offset	Name	Type	Reset	Description
0x040	RRAR	RW	-	3.8.1 RAM Read Address Register on page 3-57
0x044	RRDR	RO	-	3.8.2 RAM Read Data Register on page 3-57
0x048	RWAR	RW	-	3.8.3 RAM Write Address Register on page 3-58
0x04C	RWDR	WO	-	3.8.4 RAM Write Data Register on page 3-58

3.8 RAM register descriptions

This section describes the ELA-600 RAM registers.

[Table 3-15 RAM registers summary on page 3-56](#) provides cross-references to individual registers.

This section contains the following subsections:

- [3.8.1 RAM Read Address Register on page 3-57.](#)
- [3.8.2 RAM Read Data Register on page 3-57.](#)
- [3.8.3 RAM Write Address Register on page 3-58.](#)
- [3.8.4 RAM Write Data Register on page 3-58.](#)

3.8.1 RAM Read Address Register

The RAM Read Address Register is used to select the address that is read from the trace SRAM into a holding register.

The RRAR characteristics are:

- Usage constraints** No access when CTRL.RUN = 1.
- Configurations** Only available in configurations with TRACE_GEN = 1.
- Attributes** See [3.7 RAM register summary on page 3-56.](#)

The following table shows the bit assignments.

Table 3-16 RRAR bit assignments

Bits	Name	Function
[RAM_ADDR_SIZE-1:0]	RRA	<p>RAM Read Address.</p> <p>Writes to the RRA cause the trace SRAM data at that address to be transferred into the holding register.</p> <p>After the SRAM read data is transferred to the holding register, RRA increments by one. This prepares the RRA address for sequential RRDR reads.</p> <p>The RRA automatically increments after APB reads from the RRDR have read the contents of the holding register. An RRDR read of the last data in the holding register initiates a read to SRAM at the address pointed to by the RRA. The holding register is filled with the data at this address, then the RRA increments.</p>

Related references

[3.8.2 RAM Read Data Register on page 3-57.](#)

3.8.2 RAM Read Data Register

The RAM Read Data Register is a read-only register that reads data from the SRAM read holding register.

The RRDR characteristics are:

- Usage constraints** No access when CTRL.RUN = 1.
- Configurations** Only available in configurations with TRACE_GEN = 1.
- Attributes** See [3.7 RAM register summary on page 3-56.](#)

The following table shows the bit assignments.

Table 3-17 RRDR bit assignments

Bits	Name	Function
[31:0]	RRD	<p>Reads SRAM data from the holding register.</p> <p>Reads from the RRD return the SRAM data from the holding register. The first read of the RRD after an RRAR update returns the trace data header byte value, zero-extended to 32 bits. Subsequent reads of the RRD return 32-bit chunks of the trace data payload, starting with the least significant word, until all the payload data has been read, that is, two words if GRP_WIDTH = 64, four words if GRP_WIDTH = 128, and eight words if GRP_WIDTH = 256.</p> <p>When the final 32 bits of the payload have been read, the RRA is incremented automatically, and the next word of SRAM data is copied into the holding register. This enables the SRAM data content to be read out efficiently.</p> <p>The RRA wraps to address zero if it is incremented beyond the maximum depth of the SRAM.</p>

3.8.3 RAM Write Address Register

The RAM Write Address Register is used to select the SRAM address that the data from the write holding register is written to.

The RWAR characteristics are:

Usage constraints No access when CTRL.RUN = 1.

Configurations Only available in configurations with TRACE_GEN = 1.

Attributes See [3.7 RAM register summary on page 3-56](#).

The following table shows the bit assignments.

Table 3-18 RWAR bit assignments

Bits	Name	Function
[31]	WRAP	The WRAP bit is set when the RAM Write Address is incremented beyond $2^{\text{RAM_ADDR_SIZE}}$ while the ELA-600 is capturing trace data. The WRAP bit is not set by writes to the RWDR that cause the RAM Write Address to roll over. Software must clear the WRAP bit when writing to the RWAR.
[RAM_ADDR_SIZE-1:0]	RWA	<p>RAM Write Address.</p> <p>Writes to the RWA set the SRAM address for data that is then written through the RWDR.</p> <p>Reads from the RWA return the address of the SRAM location that is to be written next, either by writes to the RWDR, or by the trace unit.</p> <p>When trace is stopped, the RWA contains the address of the last SRAM location that was written plus one. If the RAM Write Address was incremented beyond the depth of the RAM while the ELA-600 was capturing trace data, the WRAP bit is set.</p> <p>The RWAR is automatically incremented by APB writes to the SRAM through the RWDR.</p>

Related references

[3.8.4 RAM Write Data Register on page 3-58](#).

3.8.4 RAM Write Data Register

The RAM Write Data Register is a write-only register that writes data to the SRAM write holding register.

The RWDR characteristics are:

Usage constraints No access when CTRL.RUN = 1.

Configurations Only available in configurations with TRACE_GEN = 1.

Attributes See [3.7 RAM register summary on page 3-56](#).

The following table shows the bit assignments.

Table 3-19 RWDR bit assignments

Bits	Name	Function
[31:0]	RWDR	<p>Writes data to the write holding register and initiates an SRAM write when the write holding register is full.</p> <p>Writes to the RWD update the internal write holding register.</p> <p>The first write to the RWD sets the header byte value from the least significant byte written. Subsequent writes to the RWD set 32-bit chunks of the payload, starting with the least significant chunk. When the final 32 bits of the payload have been written, the content of the holding register is copied into the SRAM and the RWA is incremented automatically.</p>

3.9 Trigger State register summary

This section gives a summary of the ELA-600 *Trigger State* registers.

The following table shows the trigger state registers in offset order from the base address of the ELA-600.

Note

All the *Trigger State* registers must be initialized by software before writing CTRL.RUN=1.

Table 3-20 Trigger state registers summary

Offset	Name	Type	Reset	Description
<i>Trigger State 0 registers</i>				
0x100	SIGSEL0	RW	-	3.10.1 Signal Select registers on page 3-68
0x104	TRIGCTRL0	RW	-	3.10.2 Trigger Control registers on page 3-69
0x108	NEXTSTATE0	RW	-	3.10.3 Next State registers on page 3-71
0x10C	ACTION0	RW	0x00	3.10.4 Action registers on page 3-72
0x110	ALTNEXTSTATE0	RW	-	3.10.5 Alt Next State registers on page 3-72
0x114	ALTACTION0	RW	0x00	3.10.6 Alt Action registers on page 3-73
0x118	COMPCTRL0	RW	-	3.10.7 Comparator Control registers on page 3-74
0x11C	ALTCOMPCTRL0	RW	-	3.10.8 Alternate Comparator Control registers on page 3-75
0x120	COUNTCOMP0	RW	-	3.10.9 Counter Compare registers on page 3-76
0x128	TWBSEL0	RW	-	3.10.10 Trace Write Byte Select registers on page 3-76
0x130	EXTMASK0	RW	-	3.10.11 External Mask registers on page 3-77
0x134	EXTCOMP0	RW	-	3.10.12 External Compare registers on page 3-77
0x138	QUALMASK0	RW	-	3.10.13 Qualifier Mask registers on page 3-78
0x13C	QUALCOMP0	RW	-	3.10.14 Qualifier Compare registers on page 3-78
0x140	SIGMASK0[31:0]	RW	-	3.10.15 Signal Mask registers on page 3-79
0x144	SIGMASK0[63:32]	RW	-	3.10.15 Signal Mask registers on page 3-79
0x148	SIGMASK0[95:64]	RW	-	3.10.15 Signal Mask registers on page 3-79
0x14C	SIGMASK0[127:96]	RW	-	3.10.15 Signal Mask registers on page 3-79
0x150	SIGMASK0[159:128]	RW	-	3.10.15 Signal Mask registers on page 3-79
0x154	SIGMASK0[191:160]	RW	-	3.10.15 Signal Mask registers on page 3-79
0x158	SIGMASK0[223:192]	RW	-	3.10.15 Signal Mask registers on page 3-79
0x15C	SIGMASK0[255:224]	RW	-	3.10.15 Signal Mask registers on page 3-79
0x180	SIGCOMP0[31:0]	RW	-	3.10.16 Signal Compare registers on page 3-80
0x184	SIGCOMP0[63:32]	RW	-	3.10.16 Signal Compare registers on page 3-80
0x188	SIGCOMP0[95:64]	RW	-	3.10.16 Signal Compare registers on page 3-80
0x18C	SIGCOMP0[127:96]	RW	-	3.10.16 Signal Compare registers on page 3-80

Table 3-20 Trigger state registers summary (continued)

Offset	Name	Type	Reset	Description
0x190	SIGCOMP0[159:128]	RW	-	3.10.16 Signal Compare registers on page 3-80
0x194	SIGCOMP0[191:160]	RW	-	3.10.16 Signal Compare registers on page 3-80
0x198	SIGCOMP0[223:192]	RW	-	3.10.16 Signal Compare registers on page 3-80
0x19C	SIGCOMP0[255:224]	RW	-	3.10.16 Signal Compare registers on page 3-80
<i>Trigger State 1 registers</i>				
0x200	SIGSEL1	RW	-	3.10.1 Signal Select registers on page 3-68
0x204	TRIGCTRL1	RW	-	3.10.2 Trigger Control registers on page 3-69
0x208	NEXTSTATE1	RW	-	3.10.3 Next State registers on page 3-71
0x20C	ACTION1	RW	0x00	3.10.4 Action registers on page 3-72
0x210	ALTNEXTSTATE1	RW	-	3.10.5 Alt Next State registers on page 3-72
0x214	ALTACTION1	RW	0x00	3.10.6 Alt Action registers on page 3-73
0x218	COMPCTRL1	RW	-	3.10.7 Comparator Control registers on page 3-74
0x21C	ALTCOMPCTRL1	RW	-	3.10.8 Alternate Comparator Control registers on page 3-75
0x220	COUNTCOMP1	RW	-	3.10.9 Counter Compare registers on page 3-76
0x228	TWBSEL1	RW	-	3.10.10 Trace Write Byte Select registers on page 3-76
0x230	EXTMASK1	RW	-	3.10.11 External Mask registers on page 3-77
0x234	EXTCOMP1	RW	-	3.10.12 External Compare registers on page 3-77
0x238	QUALMASK1	RW	-	3.10.13 Qualifier Mask registers on page 3-78
0x23C	QUALCOMP1	RW	-	3.10.14 Qualifier Compare registers on page 3-78
0x240	SIGMASK1[31:0]	RW	-	3.10.15 Signal Mask registers on page 3-79
0x244	SIGMASK1[63:32]	RW	-	3.10.15 Signal Mask registers on page 3-79
0x248	SIGMASK1[95:64]	RW	-	3.10.15 Signal Mask registers on page 3-79
0x24C	SIGMASK1[127:96]	RW	-	3.10.15 Signal Mask registers on page 3-79
0x250	SIGMASK1[159:128]	RW	-	3.10.15 Signal Mask registers on page 3-79
0x254	SIGMASK1[191:160]	RW	-	3.10.15 Signal Mask registers on page 3-79
0x258	SIGMASK1[223:192]	RW	-	3.10.15 Signal Mask registers on page 3-79
0x25C	SIGMASK1[255:224]	RW	-	3.10.15 Signal Mask registers on page 3-79
0x280	SIGCOMP1[31:0]	RW	-	3.10.16 Signal Compare registers on page 3-80
0x284	SIGCOMP1[63:32]	RW	-	3.10.16 Signal Compare registers on page 3-80
0x288	SIGCOMP1[95:64]	RW	-	3.10.16 Signal Compare registers on page 3-80
0x28C	SIGCOMP1[127:96]	RW	-	3.10.16 Signal Compare registers on page 3-80
0x290	SIGCOMP1[159:128]	RW	-	3.10.16 Signal Compare registers on page 3-80
0x294	SIGCOMP1[191:160]	RW	-	3.10.16 Signal Compare registers on page 3-80
0x298	SIGCOMP1[223:192]	RW	-	3.10.16 Signal Compare registers on page 3-80
0x29C	SIGCOMP1[255:224]	RW	-	3.10.16 Signal Compare registers on page 3-80

Table 3-20 Trigger state registers summary (continued)

Offset	Name	Type	Reset	Description
<i>Trigger State 2 registers</i>				
0x300	SIGSEL2	RW	-	3.10.1 Signal Select registers on page 3-68
0x304	TRIGCTRL2	RW	-	3.10.2 Trigger Control registers on page 3-69
0x308	NEXTSTATE2	RW	-	3.10.3 Next State registers on page 3-71
0x30C	ACTION2	RW	0x00	3.10.4 Action registers on page 3-72
0x310	ALTNEXTSTATE2	RW	-	3.10.5 Alt Next State registers on page 3-72
0x314	ALTACTION2	RW	0x00	3.10.6 Alt Action registers on page 3-73
0x318	COMPCTRL2	RW	-	3.10.7 Comparator Control registers on page 3-74
0x31C	ALTCOMPCTRL2	RW	-	3.10.8 Alternate Comparator Control registers on page 3-75
0x320	COUNTCOMP2	RW	-	3.10.9 Counter Compare registers on page 3-76
0x328	TWBSEL2	RW	-	3.10.10 Trace Write Byte Select registers on page 3-76
0x330	EXTMASK2	RW	-	3.10.11 External Mask registers on page 3-77
0x334	EXTCOMP2	RW	-	3.10.12 External Compare registers on page 3-77
0x338	QUALMASK2	RW	-	3.10.13 Qualifier Mask registers on page 3-78
0x33C	QUALCOMP2	RW	-	3.10.14 Qualifier Compare registers on page 3-78
0x340	SIGMASK2[31:0]	RW	-	3.10.15 Signal Mask registers on page 3-79
0x344	SIGMASK2[63:32]	RW	-	3.10.15 Signal Mask registers on page 3-79
0x348	SIGMASK2[95:64]	RW	-	3.10.15 Signal Mask registers on page 3-79
0x34C	SIGMASK2[127:96]	RW	-	3.10.15 Signal Mask registers on page 3-79
0x350	SIGMASK2[159:128]	RW	-	3.10.15 Signal Mask registers on page 3-79
0x354	SIGMASK2[191:160]	RW	-	3.10.15 Signal Mask registers on page 3-79
0x358	SIGMASK2[223:192]	RW	-	3.10.15 Signal Mask registers on page 3-79
0x35C	SIGMASK2[255:224]	RW	-	3.10.15 Signal Mask registers on page 3-79
0x380	SIGCOMP2[31:0]	RW	-	3.10.16 Signal Compare registers on page 3-80
0x384	SIGCOMP2[63:32]	RW	-	3.10.16 Signal Compare registers on page 3-80
0x388	SIGCOMP2[95:64]	RW	-	3.10.16 Signal Compare registers on page 3-80
0x38C	SIGCOMP2[127:96]	RW	-	3.10.16 Signal Compare registers on page 3-80
0x390	SIGCOMP2[159:128]	RW	-	3.10.16 Signal Compare registers on page 3-80
0x394	SIGCOMP2[191:160]	RW	-	3.10.16 Signal Compare registers on page 3-80
0x398	SIGCOMP2[223:192]	RW	-	3.10.16 Signal Compare registers on page 3-80
0x39C	SIGCOMP2[255:224]	RW	-	3.10.16 Signal Compare registers on page 3-80
<i>Trigger State 3 registers</i>				
0x400	SIGSEL3	RW	-	3.10.1 Signal Select registers on page 3-68
0x404	TRIGCTRL3	RW	-	3.10.2 Trigger Control registers on page 3-69
0x408	NEXTSTATE3	RW	-	3.10.3 Next State registers on page 3-71

Table 3-20 Trigger state registers summary (continued)

Offset	Name	Type	Reset	Description
0x40C	ACTION3	RW	0x00	3.10.4 Action registers on page 3-72
0x410	ALTNEXTSTATE3	RW	-	3.10.5 Alt Next State registers on page 3-72
0x414	ALTACTION3	RW	0x00	3.10.6 Alt Action registers on page 3-73
0x418	COMPCTRL3	RW	-	3.10.7 Comparator Control registers on page 3-74
0x41C	ALTCOMPCTRL3	RW	-	3.10.8 Alternate Comparator Control registers on page 3-75
0x420	COUNTCOMP3	RW	-	3.10.9 Counter Compare registers on page 3-76
0x428	TWBSEL3	RW	-	3.10.10 Trace Write Byte Select registers on page 3-76
0x430	EXTMASK3	RW	-	3.10.11 External Mask registers on page 3-77
0x434	EXTCOMP3	RW	-	3.10.12 External Compare registers on page 3-77
0x438	QUALMASK3	RW	-	3.10.13 Qualifier Mask registers on page 3-78
0x43C	QUALCOMP3	RW	-	3.10.14 Qualifier Compare registers on page 3-78
0x440	SIGMASK3[31:0]	RW	-	3.10.15 Signal Mask registers on page 3-79
0x444	SIGMASK3[63:32]	RW	-	3.10.15 Signal Mask registers on page 3-79
0x448	SIGMASK3[95:64]	RW	-	3.10.15 Signal Mask registers on page 3-79
0x44C	SIGMASK3[127:96]	RW	-	3.10.15 Signal Mask registers on page 3-79
0x450	SIGMASK3[159:128]	RW	-	3.10.15 Signal Mask registers on page 3-79
0x454	SIGMASK3[191:160]	RW	-	3.10.15 Signal Mask registers on page 3-79
0x458	SIGMASK3[223:192]	RW	-	3.10.15 Signal Mask registers on page 3-79
0x45C	SIGMASK3[255:224]	RW	-	3.10.15 Signal Mask registers on page 3-79
0x480	SIGCOMP3[31:0]	RW	-	3.10.16 Signal Compare registers on page 3-80
0x484	SIGCOMP3[63:32]	RW	-	3.10.16 Signal Compare registers on page 3-80
0x488	SIGCOMP3[95:64]	RW	-	3.10.16 Signal Compare registers on page 3-80
0x48C	SIGCOMP3[127:96]	RW	-	3.10.16 Signal Compare registers on page 3-80
0x490	SIGCOMP3[159:128]	RW	-	3.10.16 Signal Compare registers on page 3-80
0x494	SIGCOMP3[191:160]	RW	-	3.10.16 Signal Compare registers on page 3-80
0x498	SIGCOMP3[223:192]	RW	-	3.10.16 Signal Compare registers on page 3-80
0x49C	SIGCOMP3[255:224]	RW	-	3.10.16 Signal Compare registers on page 3-80
<i>Trigger State 4 registers</i>				
0x500	SIGSEL4	RW	-	3.10.1 Signal Select registers on page 3-68
0x504	TRIGCTRL4	RW	-	3.10.2 Trigger Control registers on page 3-69
0x508	NEXTSTATE4	RW	-	3.10.3 Next State registers on page 3-71
0x50C	ACTION4	RW	0x00	3.10.4 Action registers on page 3-72
0x510	ALTNEXTSTATE4	RW	-	3.10.5 Alt Next State registers on page 3-72
0x514	ALTACTION4	RW	0x00	3.10.6 Alt Action registers on page 3-73
0x518	COMPCTRL4	RW	-	3.10.7 Comparator Control registers on page 3-74

Table 3-20 Trigger state registers summary (continued)

Offset	Name	Type	Reset	Description
0x51C	ALTCOMPCTRL4	RW	-	3.10.8 Alternate Comparator Control registers on page 3-75
0x520	COUNTCOMP4	RW	-	3.10.9 Counter Compare registers on page 3-76
0x528	TWBSEL4	RW	-	3.10.10 Trace Write Byte Select registers on page 3-76
0x530	EXTMASK4	RW	-	3.10.11 External Mask registers on page 3-77
0x534	EXTCOMP4	RW	-	3.10.12 External Compare registers on page 3-77
0x538	QUALMASK4	RW	-	3.10.13 Qualifier Mask registers on page 3-78
0x53C	QUALCOMP4	RW	-	3.10.14 Qualifier Compare registers on page 3-78
0x540	SIGMASK4[31:0]	RW	-	3.10.15 Signal Mask registers on page 3-79
0x544	SIGMASK4[63:32]	RW	-	3.10.15 Signal Mask registers on page 3-79
0x548	SIGMASK4[95:64]	RW	-	3.10.15 Signal Mask registers on page 3-79
0x54C	SIGMASK4[127:96]	RW	-	3.10.15 Signal Mask registers on page 3-79
0x550	SIGMASK4[159:128]	RW	-	3.10.15 Signal Mask registers on page 3-79
0x554	SIGMASK4[191:160]	RW	-	3.10.15 Signal Mask registers on page 3-79
0x558	SIGMASK4[223:192]	RW	-	3.10.15 Signal Mask registers on page 3-79
0x55C	SIGMASK4[255:224]	RW	-	3.10.15 Signal Mask registers on page 3-79
0x580	SIGCOMP4[31:0]	RW	-	3.10.16 Signal Compare registers on page 3-80
0x584	SIGCOMP4[63:32]	RW	-	3.10.16 Signal Compare registers on page 3-80
0x588	SIGCOMP4[95:64]	RW	-	3.10.16 Signal Compare registers on page 3-80
0x58C	SIGCOMP4[127:96]	RW	-	3.10.16 Signal Compare registers on page 3-80
0x590	SIGCOMP4[159:128]	RW	-	3.10.16 Signal Compare registers on page 3-80
0x594	SIGCOMP4[191:160]	RW	-	3.10.16 Signal Compare registers on page 3-80
0x598	SIGCOMP4[223:192]	RW	-	3.10.16 Signal Compare registers on page 3-80
0x59C	SIGCOMP4[255:224]	RW	-	3.10.16 Signal Compare registers on page 3-80
<i>Trigger State 5 registers</i>				
0x600	SIGSEL5	RW	-	3.10.1 Signal Select registers on page 3-68
0x604	TRIGCTRL5	RW	-	3.10.2 Trigger Control registers on page 3-69
0x608	NEXTSTATE5	RW	-	3.10.3 Next State registers on page 3-71
0x60C	ACTION5	RW	0x00	3.10.4 Action registers on page 3-72
0x610	ALTNEXTSTATE5	RW	-	3.10.5 Alt Next State registers on page 3-72
0x614	ALTACTION5	RW	0x00	3.10.6 Alt Action registers on page 3-73
0x618	COMPCTRL5	RW	-	3.10.7 Comparator Control registers on page 3-74
0x61C	ALTCOMPCTRL5	RW	-	3.10.8 Alternate Comparator Control registers on page 3-75
0x620	COUNTCOMP5	RW	-	3.10.9 Counter Compare registers on page 3-76
0x628	TWBSEL5	RW	-	3.10.10 Trace Write Byte Select registers on page 3-76
0x630	EXTMASK5	RW	-	3.10.11 External Mask registers on page 3-77

Table 3-20 Trigger state registers summary (continued)

Offset	Name	Type	Reset	Description
0x634	EXTCOMP5	RW	-	3.10.12 External Compare registers on page 3-77
0x638	QUALMASK5	RW	-	3.10.13 Qualifier Mask registers on page 3-78
0x63C	QUALCOMP5	RW	-	3.10.14 Qualifier Compare registers on page 3-78
0x640	SIGMASK5[31:0]	RW	-	3.10.15 Signal Mask registers on page 3-79
0x644	SIGMASK5[63:32]	RW	-	3.10.15 Signal Mask registers on page 3-79
0x648	SIGMASK5[95:64]	RW	-	3.10.15 Signal Mask registers on page 3-79
0x64C	SIGMASK5[127:96]	RW	-	3.10.15 Signal Mask registers on page 3-79
0x650	SIGMASK5[159:128]	RW	-	3.10.15 Signal Mask registers on page 3-79
0x654	SIGMASK5[191:160]	RW	-	3.10.15 Signal Mask registers on page 3-79
0x658	SIGMASK5[223:192]	RW	-	3.10.15 Signal Mask registers on page 3-79
0x65C	SIGMASK5[255:224]	RW	-	3.10.15 Signal Mask registers on page 3-79
0x680	SIGCOMP5[31:0]	RW	-	3.10.16 Signal Compare registers on page 3-80
0x684	SIGCOMP5[63:32]	RW	-	3.10.16 Signal Compare registers on page 3-80
0x688	SIGCOMP5[95:64]	RW	-	3.10.16 Signal Compare registers on page 3-80
0x68C	SIGCOMP5[127:96]	RW	-	3.10.16 Signal Compare registers on page 3-80
0x690	SIGCOMP5[159:128]	RW	-	3.10.16 Signal Compare registers on page 3-80
0x694	SIGCOMP5[191:160]	RW	-	3.10.16 Signal Compare registers on page 3-80
0x698	SIGCOMP5[223:192]	RW	-	3.10.16 Signal Compare registers on page 3-80
0x69C	SIGCOMP5[255:224]	RW	-	3.10.16 Signal Compare registers on page 3-80
<i>Trigger State 6 registers</i>				
0x700	SIGSEL6	RW	-	3.10.1 Signal Select registers on page 3-68
0x704	TRIGCTRL6	RW	-	3.10.2 Trigger Control registers on page 3-69
0x708	NEXTSTATE6	RW	-	3.10.3 Next State registers on page 3-71
0x70C	ACTION6	RW	0x00	3.10.4 Action registers on page 3-72
0x710	ALTNEXTSTATE6	RW	-	3.10.5 Alt Next State registers on page 3-72
0x714	ALTACTION6	RW	0x00	3.10.6 Alt Action registers on page 3-73
0x718	COMPCTRL6	RW	-	3.10.7 Comparator Control registers on page 3-74
0x71C	ALTCOMPCTRL6	RW	-	3.10.8 Alternate Comparator Control registers on page 3-75
0x720	COUNTCOMP6	RW	-	3.10.9 Counter Compare registers on page 3-76
0x728	TWBSEL6	RW	-	3.10.10 Trace Write Byte Select registers on page 3-76
0x730	EXTMASK6	RW	-	3.10.11 External Mask registers on page 3-77
0x734	EXTCOMP6	RW	-	3.10.12 External Compare registers on page 3-77
0x738	QUALMASK6	RW	-	3.10.13 Qualifier Mask registers on page 3-78
0x73C	QUALCOMP6	RW	-	3.10.14 Qualifier Compare registers on page 3-78
0x740	SIGMASK6[31:0]	RW	-	3.10.15 Signal Mask registers on page 3-79

Table 3-20 Trigger state registers summary (continued)

Offset	Name	Type	Reset	Description
0x744	SIGMASK6[63:32]	RW	-	3.10.15 Signal Mask registers on page 3-79
0x748	SIGMASK6[95:64]	RW	-	3.10.15 Signal Mask registers on page 3-79
0x74C	SIGMASK6[127:96]	RW	-	3.10.15 Signal Mask registers on page 3-79
0x750	SIGMASK6[159:128]	RW	-	3.10.15 Signal Mask registers on page 3-79
0x754	SIGMASK6[191:160]	RW	-	3.10.15 Signal Mask registers on page 3-79
0x758	SIGMASK6[223:192]	RW	-	3.10.15 Signal Mask registers on page 3-79
0x75C	SIGMASK6[255:224]	RW	-	3.10.15 Signal Mask registers on page 3-79
0x780	SIGCOMP6[31:0]	RW	-	3.10.16 Signal Compare registers on page 3-80
0x784	SIGCOMP6[63:32]	RW	-	3.10.16 Signal Compare registers on page 3-80
0x788	SIGCOMP6[95:64]	RW	-	3.10.16 Signal Compare registers on page 3-80
0x78C	SIGCOMP6[127:96]	RW	-	3.10.16 Signal Compare registers on page 3-80
0x790	SIGCOMP6[159:128]	RW	-	3.10.16 Signal Compare registers on page 3-80
0x794	SIGCOMP6[191:160]	RW	-	3.10.16 Signal Compare registers on page 3-80
0x798	SIGCOMP6[223:192]	RW	-	3.10.16 Signal Compare registers on page 3-80
0x79C	SIGCOMP6[255:224]	RW	-	3.10.16 Signal Compare registers on page 3-80
<i>Trigger State 8 registers</i>				
0x800	SIGSEL7	RW	-	3.10.1 Signal Select registers on page 3-68
0x804	TRIGCTRL7	RW	-	3.10.2 Trigger Control registers on page 3-69
0x808	NEXTSTATE7	RW	-	3.10.3 Next State registers on page 3-71
0x80C	ACTION7	RW	0x00	3.10.4 Action registers on page 3-72
0x810	ALTNEXTSTATE7	RW	-	3.10.5 Alt Next State registers on page 3-72
0x814	ALTACTION7	RW	0x00	3.10.6 Alt Action registers on page 3-73
0x818	COMPCTRL7	RW	-	3.10.7 Comparator Control registers on page 3-74
0x81C	ALTCOMPCTRL7	RW	-	3.10.8 Alternate Comparator Control registers on page 3-75
0x820	COUNTCOMP7	RW	-	3.10.9 Counter Compare registers on page 3-76
0x828	TWBSEL7	RW	-	3.10.10 Trace Write Byte Select registers on page 3-76
0x830	EXTMASK7	RW	-	3.10.11 External Mask registers on page 3-77
0x834	EXTCOMP7	RW	-	3.10.12 External Compare registers on page 3-77
0x838	QUALMASK7	RW	-	3.10.13 Qualifier Mask registers on page 3-78
0x83C	QUALCOMP7	RW	-	3.10.14 Qualifier Compare registers on page 3-78
0x840	SIGMASK7[31:0]	RW	-	3.10.15 Signal Mask registers on page 3-79
0x844	SIGMASK7[63:32]	RW	-	3.10.15 Signal Mask registers on page 3-79
0x848	SIGMASK7[95:64]	RW	-	3.10.15 Signal Mask registers on page 3-79
0x84C	SIGMASK7[127:96]	RW	-	3.10.15 Signal Mask registers on page 3-79
0x850	SIGMASK7[159:128]	RW	-	3.10.15 Signal Mask registers on page 3-79

Table 3-20 Trigger state registers summary (continued)

Offset	Name	Type	Reset	Description
0x854	SIGMASK7[191:160]	RW	-	3.10.15 Signal Mask registers on page 3-79
0x858	SIGMASK7[223:192]	RW	-	3.10.15 Signal Mask registers on page 3-79
0x85C	SIGMASK7[255:224]	RW	-	3.10.15 Signal Mask registers on page 3-79
0x880	SIGCOMP7[31:0]	RW	-	3.10.16 Signal Compare registers on page 3-80
0x884	SIGCOMP7[63:32]	RW	-	3.10.16 Signal Compare registers on page 3-80
0x888	SIGCOMP7[95:64]	RW	-	3.10.16 Signal Compare registers on page 3-80
0x88C	SIGCOMP7[127:96]	RW	-	3.10.16 Signal Compare registers on page 3-80
0x890	SIGCOMP7[159:128]	RW	-	3.10.16 Signal Compare registers on page 3-80
0x894	SIGCOMP7[191:160]	RW	-	3.10.16 Signal Compare registers on page 3-80
0x898	SIGCOMP7[223:192]	RW	-	3.10.16 Signal Compare registers on page 3-80
0x89C	SIGCOMP7[255:224]	RW	-	3.10.16 Signal Compare registers on page 3-80

3.10 Trigger State register descriptions

This section describes the ELA-600 *Trigger State* registers.

[Table 3-20 Trigger state registers summary on page 3-60](#) provides cross-references to individual registers.

Note

In the following register descriptions, a lowercase $\langle n \rangle$, where $n = 0$ to 7 , denotes one of the eight *Trigger States*. For example, the SIGSEL2 register selects which input bus is used when the ELA-600 is in *Trigger State 2*.

This section contains the following subsections:

- [3.10.1 Signal Select registers on page 3-68.](#)
- [3.10.2 Trigger Control registers on page 3-69.](#)
- [3.10.3 Next State registers on page 3-71.](#)
- [3.10.4 Action registers on page 3-72.](#)
- [3.10.5 Alt Next State registers on page 3-72.](#)
- [3.10.6 Alt Action registers on page 3-73.](#)
- [3.10.7 Comparator Control registers on page 3-74.](#)
- [3.10.8 Alternate Comparator Control registers on page 3-75.](#)
- [3.10.9 Counter Compare registers on page 3-76.](#)
- [3.10.10 Trace Write Byte Select registers on page 3-76.](#)
- [3.10.11 External Mask registers on page 3-77.](#)
- [3.10.12 External Compare registers on page 3-77.](#)
- [3.10.13 Qualifier Mask registers on page 3-78.](#)
- [3.10.14 Qualifier Compare registers on page 3-78.](#)
- [3.10.15 Signal Mask registers on page 3-79.](#)
- [3.10.16 Signal Compare registers on page 3-80.](#)

3.10.1 Signal Select registers

The Signal Select registers control the selection of the debug signal bus that is masked using the Signal Mask, and compared to the Signal Compare registers for all eight *Trigger States*.

The SIGSEL $\langle n \rangle$ register characteristics are:

Usage constraints	Writing when CTRL.RUN = 1 results in improper operation.
Configurations	Available in all configurations. SIGSEL5, SIGSEL6, and SIGSEL7 are only available when NUM_TRIG_STATES = 8.
Attributes	See 3.9 Trigger State register summary on page 3-60 .

The following table shows the bit assignments.

Table 3-21 SIGSEL<n> register bit assignments

Bits	Name	Function
[31:12]	Reserved	-
[11:0]	SIGSEL<n>	<p>Selects <i>Signal Group</i>.</p> <p>0x1 Selects <i>Signal Group 0</i>.</p> <p>0x2 Selects <i>Signal Group 1</i>.</p> <p>0x4 Selects <i>Signal Group 2</i>.</p> <p>0x8 Selects <i>Signal Group 3</i>.</p> <p>0x10 Selects <i>Signal Group 4</i>.</p> <p>0x20 Selects <i>Signal Group 5</i>.</p> <p>0x40 Selects <i>Signal Group 6</i>.</p> <p>0x80 Selects <i>Signal Group 7</i>.</p> <p>0x100 Selects <i>Signal Group 8</i>.</p> <p>0x200 Selects <i>Signal Group 9</i>.</p> <p>0x400 Selects <i>Signal Group 10</i>.</p> <p>0x800 Selects <i>Signal Group 11</i>.</p>

3.10.2 Trigger Control registers

The Trigger Control registers are used to select the comparison type for each *Trigger State*. The comparisons are between the input *Signal Group* that is masked by the Signal Mask register, and the Signal Compare Registers.

The TRIGCTRL<n> register characteristics are:

Usage constraints Writing when CTRL.RUN = 1 results in improper operation.

Configurations Available in all configurations. TRIGCTRL5, TRIGCTRL6, and TRIGCTRL7 are only available when NUM_TRIG_STATES = 8.

Attributes See [3.9 Trigger State register summary](#) on page 3-60.

The following table shows the bit assignments.

Table 3-22 TRIGCTRL<n> register bit assignments

Bits	Name	Function
[15]	ALTCOMPSEL	<p>Selects the alternative comparison mode:</p> <p>0b0 <i>Trigger Signal Alternative Comparisons</i> selected.</p> <p>0b1 <i>Trigger Counter Alternative Comparisons</i> selected.</p>
[14:12]	ALTCOMP	<p><i>Trigger Signal Alternative Comparison</i> type select:</p> <p>0b000 <i>Trigger Signal Alternative Comparisons</i> disabled.</p> <p>0b001 Alternative compare type is <i>equal</i> (==).</p> <p>0b010 Alternative compare type is <i>greater than</i> (>).</p> <p>0b011 Alternative compare type is <i>greater than or equal</i> (>=).</p> <p>0b101 Alternative compare type is <i>not equal</i> (!=).</p> <p>0b110 Alternative compare type is <i>less than</i> (<).</p> <p>0b111 Alternative compare type is <i>less than or equal</i> (<=).</p>

Table 3-22 TRIGCTRL<n> register bit assignments (continued)

Bits	Name	Function
[11:10]	CAPTID	<p>0b00 Disable use of the captured ID for signal comparisons.</p> <p>0b01 Capture ID when trigger signal condition matches. The ID is captured, from SIGNALGRP<n>[ID_CAPTURE_SIZE-1:0].</p> <p>0b10 Use the captured ID instead of the target value in SIGCOMP<n>[ID_CAPTURE_SIZE-1:0] for comparison of SIGNALGRP<n>[ID_CAPTURE_SIZE-1:0].</p> <p>0b11 Use the captured ID instead of the SIGNALGRP<n>[ID_CAPTURE_SIZE-1:0] for a comparison against SIGCOMP<n>[ID_CAPTURE_SIZE-1:0].</p>
[9]	COUNTBRK	<p>Loop counter break.</p> <p>The loop counter break uses the <i>Trigger State</i> counter to break loops between <i>Trigger States</i> after a <i>Trigger Counter Comparison</i>. When the counter comparison matches, the <i>Trigger State</i> goes into a final state, which stops trace writes and leaves the output actions at the previous <i>Trigger State</i> ACTION value.</p> <p>0b0 Normal operation.</p> <p>0b1 Break <i>Trigger State</i> loop: A counter comparison match causes a transition to the final state, otherwise go to the NEXTSTATE<n> <i>Trigger State</i> as the counter increments.</p>
[8]	COUNTCLR	<p>Counter clear.</p> <p>0b0 Do not clear the counter value when moving to a different NEXTSTATE<n>.</p> <p>0b1 Clear the counter value when moving to a different NEXTSTATE<n>.</p> <p>————— Note —————</p> <p>TRIGCTRL.WATCHRST must be 0b0 when using this feature.</p> <p>—————</p>
[7:6]	TRACE	<p>Trace capture control.</p> <p>0b00 Trace is captured when <i>Trigger Signal Comparison</i> succeeds.</p> <p>0b01 Trace is captured when <i>Trigger Counter Comparison</i> succeeds.</p> <p>0b10 Trace is captured every ELACLK cycle.</p> <p>0b11 <i>Trigger State</i> counters are enabled with COUNTSRC = 0 and traced when there is a <i>Trigger Signal Comparison</i>. Counters must also be selected in CNTSEL.</p>
[5]	COUNTSRC	<p>Counter source select.</p> <p>0b0 Counter is incremented every ELACLK cycle.</p> <p>0b1 Counter is incremented when <i>Trigger Signal Comparison</i> matches.</p>
[4]	WATCHRST	<p>Counter reset.</p> <p>0b0 Do not reset the counter after a <i>Trigger Signal Comparison</i> match.</p> <p>0b1 Reset the counter after a <i>Trigger Signal Comparison</i> match.</p> <p>The counter acts like an activity watchdog timer, only allowing advancement to the next <i>Trigger State</i> when the <i>Trigger Counter Comparison</i> is reached. The counter is reset by a signal comparison.</p>

Table 3-22 TRIGCTRL<n> register bit assignments (continued)

Bits	Name	Function
[3]	COMPSEL	Comparison mode. Acts as both a counter enable and a select for the comparison mode. 0b0 Disable counters and select <i>Trigger Signal Comparison</i> mode. 0b1 Enable counters and select <i>Trigger Counter Comparison</i> mode.
[2:0]	COMP	<i>Trigger Signal Comparison</i> type select. 0b000 <i>Trigger Signal Comparisons</i> disabled. The enabled counters count clocks immediately after the <i>Trigger State</i> has been entered and generate a programmable <i>Output Action</i> and transition to the next <i>Trigger State</i> when the Counter Compare Register count is reached, that is when a <i>Trigger Counter Comparison</i> match occurs. 0b001 Compare type is <i>equal</i> (==). 0b010 Compare type is <i>greater than</i> (>). 0b011 Compare type is <i>greater than or equal</i> (>=). 0b101 Compare type is <i>not equal</i> (!=). 0b110 Compare type is <i>less than</i> (<). 0b111 Compare type is <i>less than or equal</i> (<=).

3.10.3 Next State registers

The Next State registers are zero-one-hot encoded registers that point to the next *Trigger State* that is entered after the *Trigger Condition* is met.

The NEXTSTATE<n> register characteristics are:

Usage constraints Writing when CTRL.RUN = 1 results in improper operation.

Configurations Available in all configurations. NEXTSTATE5, NEXTSTATE6, and NEXTSTATE7 are only available when NUM_TRIG_STATES = 8.

Attributes See [3.9 Trigger State register summary on page 3-60](#).

The following table shows the bit assignments.

Table 3-23 NEXTSTATE<n> register bit assignments

Bits	Name	Function
[NUM_TRIG_STATES-1:0]	NEXTSTATE<n>	Selects the next state to move to after the <i>Trigger Condition</i> has been met in the current state. 0x0 Do not change state. This is the final <i>Trigger State</i> . 0x1 Selects <i>Trigger State</i> 0. 0x2 Selects <i>Trigger State</i> 1. 0x4 Selects <i>Trigger State</i> 2. 0x8 Selects <i>Trigger State</i> 3. 0x10 Selects <i>Trigger State</i> 4, when NUM_TRIG_STATES = 5. 0x20 Selects <i>Trigger State</i> 5, when NUM_TRIG_STATES = 8. 0x40 Selects <i>Trigger State</i> 6, when NUM_TRIG_STATES = 8. 0x80 Selects <i>Trigger State</i> 7, when NUM_TRIG_STATES = 8.

3.10.4 Action registers

The ACTION registers enable and disable trace, and control the level of the logic analyzer outputs on the ELAOUTPUT[3:0], STOPCLOCK, and CTTRIGOUT[1:0] pins.

The ACTION<n> register characteristics are:

Usage constraints Writing when CTRL.RUN = 1 results in improper operation.

Configurations Available in all configurations. ACTION5, ACTION6, and ACTION7 are only available when NUM_TRIG_STATES = 8.

Attributes See 3.9 *Trigger State register summary* on page 3-60.

The following table shows the bit assignments.

Table 3-24 ACTION<n> register bit assignments

Bits	Name	Function
[7:4]	ELAOUTPUT	Value to drive on ELAOUTPUT[3:0].
[3]	TRACE	Trace active. 0b0 Trace is not active. 0b1 Trace is active.
[2]	STOPCLOCK	Level to drive on STOPCLOCK. 0b0 Drive 0 on STOPCLOCK. 0b1 Drive 1 on STOPCLOCK.
[1:0]	CTTRIGOUT	Value to drive on CTTRIGOUT[1:0].

————— **Note** —————

Arm recommends the following ELAOUTPUT connections:

- **CTTRIGOUT[1:0]** - Connect to a *Cross Trigger Interface (CTI)* to enable the ELA-600 to trigger devices in the CoreSight system.
- **STOPCLOCK** - Connect to clocks unit to stop all clocks for serial scan dump.
- **ELAOUTPUT[3:0]** - Connect to any external device, for example a *Generic Interrupt Controller (GIC)* as an *Interrupt Request (IRQ)* input, an oscilloscope, or another ELA-600.

3.10.5 Alt Next State registers

The Alt Next State registers are zero-one-hot encoded registers that point to the next *Trigger State* that is entered after the conditional *Trigger Condition* is met.

The ALTNEXTSTATE<n> register characteristics are:

Usage constraints Writing when CTRL.RUN = 1 results in improper operation.

Configurations Available when COND_TRIG = 1. ALTNEXTSTATE5, ALTNEXTSTATE6, and ALTNEXTSTATE7 are only available when NUM_TRIG_STATES = 8.

Attributes See 3.9 *Trigger State register summary* on page 3-60.

The following table shows the bit assignments.

Table 3-25 ALTNEXTSTATE<n> register bit assignments

Bits	Name	Function
[NUM_TRIG_STATES-1:0]	ALTNEXTSTATE<n>	<p>Selects the next state to move to after the conditional <i>Trigger Condition</i> has been met in the current state.</p> <p>0x0 Do not change state. This is the final <i>Trigger State</i>.</p> <p>0x1 Selects <i>Trigger State 0</i>.</p> <p>0x2 Selects <i>Trigger State 1</i>.</p> <p>0x4 Selects <i>Trigger State 2</i>.</p> <p>0x8 Selects <i>Trigger State 3</i>.</p> <p>0x10 Selects <i>Trigger State 4</i>, when NUM_TRIG_STATES = 5.</p> <p>0x20 Selects <i>Trigger State 5</i>, when NUM_TRIG_STATES = 8.</p> <p>0x40 Selects <i>Trigger State 6</i>, when NUM_TRIG_STATES = 8.</p> <p>0x80 Selects <i>Trigger State 7</i>, when NUM_TRIG_STATES = 8.</p>

3.10.6 Alt Action registers

The Alt Action registers enable and disable trace and control the level of the logic analyzer outputs on the ELAOUTPUT[3:0], STOPCLOCK, and CTTRIGOUT[1:0] pins.

The ALTACTION<n> register characteristics are:

Usage constraints Writing when CTRL.RUN = 1 results in improper operation.

Configurations Available when COND_TRIG = 1. ALTACTION5, ALTACTION6, and ALTACTION7 are only available when NUM_TRIG_STATES = 8.

Attributes See 3.9 *Trigger State register summary* on page 3-60.

The following table shows the bit assignments.

Table 3-26 ALTACTION<n> register bit assignments

Bits	Name	Function
[7:4]	ELAOUTPUT	Value to drive on ELAOUTPUT[3:0].
[3]	TRACE	<p>Trace active.</p> <p>0b0 Trace is not active.</p> <p>0b1 Trace is active.</p>
[2]	STOPCLOCK	<p>Level to drive on STOPCLOCK.</p> <p>0b0 Drive 0 on STOPCLOCK.</p> <p>0b1 Drive 1 on STOPCLOCK.</p>
[1:0]	CTTRIGOUT	Value to drive on CTTRIGOUT[1:0].

Note

Arm recommends the following ELAOUTPUT connections:

- **CTTRIGOUT[1:0]** - Connect to a *Cross Trigger Interface (CTI)* to enable the ELA-600 to trigger devices in the CoreSight system.
- **STOPCLOCK** - Connect to clocks unit to stop all clocks for serial scan dump.
- **ELAOUTPUT[3:0]** - Connect to any external device, for example a *Generic Interrupt Controller (GIC)* as an *Interrupt Request (IRQ)* input, an oscilloscope, or another ELA-600.

3.10.7 Comparator Control registers

The Comparator Control registers support 32-bit comparisons.

————— **Note** —————

The number of bits in the COMPCTRLn and ALTCOMPCTRLn registers depend on the GRP_WIDTH (GRP_WIDTH/32 comparators x 4 CTRL bits per 32-bit comparator). Each trigger state has a COMPCTRLn register with a 4-bit field to control GRP_WIDTH/32 comparators.

The COMPCTRL<n> register characteristics are:

Usage constraints Writing when CTRL.RUN = 1 results in improper operation.

Configurations Available with SRAM or ATB trace configurations.

Attributes See [3.9 Trigger State register summary on page 3-60](#).

4-bit CTRL Field in COMPCTRLn:

x000	32-bit Signal Comparison masked, use TRIGCTRLn register comparators (ELA-500 compatible) if all comparators are masked.
x001	Compare type is equal (==).
x010	Compare type is greater than (>).
x011	Compare type is greater than or equal (>=).
x101	Compare type is not equal (!=).
x110	Compare type is less than (<).
x111	Compare type is less than or equal (<=).
0xxx	AND this comparison with others.
1xxx	OR this comparison with others.

Simultaneous signal and alternate signal comparisons can be made by using either the larger TRIGCTRL.COMP or TRIGCTRL.ALTCOMP comparators or separate 32-bit comparators which will be removed from the larger comparator.

————— **Note** —————

If all the 32-bit comparators are enabled in either COMPCTRLn or ALTCOMPCTRLn, then the larger comparator must be disabled in TRIGCTRLn.

A 32-bit comparator is enabled and masked from the TRIGCTRL.COMP or TRIGCTRL.ALTCOMP comparators when COMPCTRL[3:0] or ALTCOMPCTRL[3:0] bits are written with a nonzero value. The TRIGCTRL.COMP or TRIGCTRL.ALTCOMP comparator masks the output from this comparator. The TRIGCTRL.COMP or TRIGCTRL.ALTCOMP comparator can therefore support comparisons from 32-bit to GRP_WIDTH in 32-bit increments.

COMPCTRL[3] or ALTCOMPCTRL[3] is either ORed or ANDed for each comparator. Each 32-bit comparator with COMPCTRL[3] = 0 or ALTCOMPCTRL[3] = 0 is first selectively ANDed, then the final ANDed output is ORed with the comparators that have COMPCTRL[3] = 1 or ALTCOMPCTRL[3] = 1.

TRIGCTRL.COMP or TRIGCTRL.ALTCOMP is always ORed with the 32 comparator output.

Backward compatibility with ELA-500 greater-than comparisons is supported with multiple 32-bit comparators instead of a single 64-, 128-, or 256-bit comparator. The following is an example of a GRP_WIDTH= 64 comparison:

```
Compare_64 = (SIGNALGRP[63:32] > SIGCOMP[63:32]) | ( (SIGNALGRP[63:32] == SIGCOMP[63:32]) &
(SIGNALGRP[31:0] > SIGCOMP[31:0]) )
```

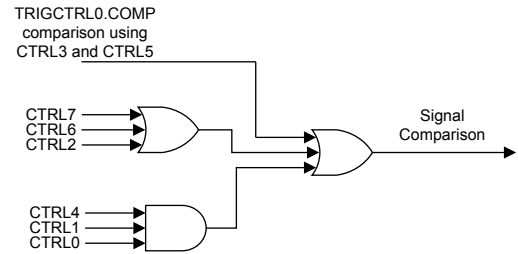


Figure 3-1 Example of COMPCTRLn and ALTCOMPCTRLn AND and OR operation

The following table shows the bit assignments.

Table 3-27 COMPCTRL<n> register bit assignments

Bits	Name	Function
[31:28]	CTRL7	Control comparison of SIGNALGRP, SIGMASK, and SIGCOMP bits [255:224] when GRP_WIDTH=256
[27:24]	CTRL6	Control comparison of SIGNALGRP, SIGMASK, and SIGCOMP bits [223:192] when GRP_WIDTH=256
[23:20]	CTRL5	Control comparison of SIGNALGRP, SIGMASK, and SIGCOMP bits [191:160] when GRP_WIDTH=256
[19:16]	CTRL4	Control comparison of SIGNALGRP, SIGMASK, and SIGCOMP bits [159:128] when GRP_WIDTH=256
[15:12]	CTRL3	Control comparison of SIGNALGRP, SIGMASK, and SIGCOMP bits [127:96] when GRP_WIDTH>64
[11:8]	CTRL2	Control comparison of SIGNALGRP, SIGMASK, and SIGCOMP bits [95:64] when GRP_WIDTH>64
[7:4]	CTRL1	Control comparison of SIGNALGRP, SIGMASK, and SIGCOMP bits [63:32]
[3:0]	CTRL0	Control comparison of SIGNALGRP, SIGMASK, and SIGCOMP bits [31:0]

3.10.8 Alternate Comparator Control registers

The Alternate Comparator Control registers support 32-bit comparisons.

The ALTCOMPCTRL<n> register characteristics are:

Usage constraints Writing when CTRL.RUN = 1 results in improper operation.

Configurations Available with SRAM or ATB trace configurations.

Attributes See [3.9 Trigger State register summary on page 3-60](#).

See [3.10.7 Comparator Control registers on page 3-74](#) for more information.

The ALTCOMPCTRL.ACTRLn bit positions are programmed the same way as COMPCTRL.CTRLn. See [3.10.7 Comparator Control registers on page 3-74](#)

The following table shows the bit assignments.

Table 3-28 ALTCOMPCTRL<n> register bit assignments

Bits	Name	Function
[31:28]	ACTRL7	Control comparison of SIGNALGRP, SIGMASK, and SIGCOMP bits [255:224] when GRP_WIDTH=256
[27:24]	ACTRL6	Control comparison of SIGNALGRP, SIGMASK, and SIGCOMP bits [223:192] when GRP_WIDTH=256
[23:20]	ACTRL5	Control comparison of SIGNALGRP, SIGMASK, and SIGCOMP bits [191:160] when GRP_WIDTH=256
[19:16]	ACTRL4	Control comparison of SIGNALGRP, SIGMASK, and SIGCOMP bits [159:128] when GRP_WIDTH=256
[15:12]	ACTRL3	Control comparison of SIGNALGRP, SIGMASK, and SIGCOMP bits [127:96] when GRP_WIDTH>64
[11:8]	ACTRL2	Control comparison of SIGNALGRP, SIGMASK, and SIGCOMP bits [95:64] when GRP_WIDTH>64

Table 3-28 ALTCOMPCTRL<n> register bit assignments (continued)

Bits	Name	Function
[7:4]	ACTRL1	Control comparison of SIGNALGRP, SIGMASK, and SIGCOMP bits [63:32]
[3:0]	ACTRL0	Control comparison of SIGNALGRP, SIGMASK, and SIGCOMP bits [31:0]

3.10.9 Counter Compare registers

The Counter Compare registers are used when *Trigger Counter Comparison* is selected in the appropriate TRIGCTRL<n> register, that is when TRIGCTRL<n>.COUNTEN = 1.

The COUNTCOMP<n> register characteristics are:

Usage constraints Writing when CTRL.RUN = 1 results in improper operation.

Configurations Available in all configurations. COUNTCOMP5, COUNTCOMP6, and COUNTCOMP7 are only available when NUM_TRIG_STATES = 8.

Attributes See [3.9 Trigger State register summary on page 3-60](#).

The following table shows the bit assignments.

Table 3-29 COUNTCOMP<n> register bit assignments

Bits	Name	Function
[31:0]	COUNTCOMP<n>	A value that, when reached in the associated up-counter for this <i>Trigger State</i> , causes a <i>Trigger Counter Comparison</i> match to occur.

3.10.10 Trace Write Byte Select registers

Trace write byte select register bits to select trace byte writes within a SIGNALGRP. Selectively tracing bytes can be used to reduce trace data or trace overruns.

The TWBSEL<n> register characteristics are:

Usage constraints Writing when CTRL.RUN = 1 results in improper operation.

Configurations Available only for the ATB trace configuration.

Attributes See [3.9 Trigger State register summary on page 3-60](#).

Each bit position in this register enables trace write from the SIGNALGRPn that is selected in the SIGSEL<n> register for each n trigger state. TWBSEL has (GRP_WIDTH/8) bits. The following table shows the bit assignments.

Table 3-30 NEXTSTATE<n> register bit assignments

Bits	Name	Function
[15]	TRACE_BYTE15	1 Trace write byte from SIGNALGRPn[127:120]
[14]	TRACE_BYTE14	1 Trace write byte from SIGNALGRPn[119:112]
...		
[0]	TRACE_BYTE0	1 Trace write byte from SIGNALGRPn[7:0]

3.10.11 External Mask registers

The External Mask registers are used to mask out specific *External Trigger Input Signals* for *Trigger Signal* comparisons.

The EXTMASK<n> register characteristics are:

Usage constraints Writing when CTRL.RUN = 1 results in improper operation.

Configurations Available in all configurations. EXTMASK5, EXTMASK6, and EXTMASK7 are only available when NUM_TRIG_STATES = 8.

Attributes See [3.9 Trigger State register summary on page 3-60](#).

The following table shows the bit assignments.

Table 3-31 EXTMASK<n> register bit assignments

Bits	Name	Function
[7:2]	EXTTRIG	Mask EXTTRIG[5:0] signals. Each signal is masked by clearing the appropriate bit. 0b0 <i>External Trigger Input Signal</i> is masked and is not used in comparisons. 0b1 <i>External Trigger Input Signal</i> is not masked.
[1:0]	CTTRIGIN	Mask CTTRIGIN[1:0] signals. Each signal is masked by clearing the appropriate bit. 0b0 <i>External Trigger Input Signal</i> is masked and is not used in comparisons. 0b1 <i>External Trigger Input Signal</i> is not masked.

Related concepts

[2.5 Triggering on page 2-37](#).

Related references

[3.10.15 Signal Mask registers on page 3-79](#).

[3.10.16 Signal Compare registers on page 3-80](#).

3.10.12 External Compare registers

The External Compare registers provide the data values with which the *External Trigger Input Signals* are compared.

The EXTCOMP<n> register characteristics are:

Usage constraints Writing when CTRL.RUN = 1 results in improper operation.

Configurations Available in all configurations. EXTCOMP5, EXTCOMP6, and EXTCOMP7 are only available when NUM_TRIG_STATES = 8.

Attributes See [3.9 Trigger State register summary on page 3-60](#).

The following table shows the bit assignments.

Table 3-32 EXTCOMP<n> register bit assignments

Bits	Name	Function
[7:2]	EXTTRIG	Compare value for EXTTRIG[5:0] signals.
[1:0]	CTTRIGIN	Compare value for CTTRIGIN[1:0] signals.

Related concepts

[2.5 Triggering on page 2-37](#).

Related references

[3.10.15 Signal Mask registers on page 3-79.](#)

[3.10.16 Signal Compare registers on page 3-80.](#)

3.10.13 Qualifier Mask registers

The QUALMASK registers are used to mask out specific QUALMASK Qualifier Input Signals for Trigger Signal comparisons.

The QUALMASK<n> register characteristics are:

Usage constraints Writing when CTRL.RUN = 1 results in improper operation.

Configurations Available only in SRAM and ATB trace configurations. QUALMASK5, QUALMASK6, and QUALMASK7 are only available when NUM_TRIG_STATES = 8.

Attributes See [3.9 Trigger State register summary on page 3-60.](#)

The following table shows the bit assignments.

————— **Note** —————

Each signal is masked by clearing the appropriate bit.

0 Input Signal is masked and is not used in comparisons.

1 Qualifier Input Signal is not masked.

Table 3-33 QUALMASK<n> register bit assignments

Bits	Name	Function
[7]	QUALMASK[7]	Mask SIGQUAL[7]. These bit is only available if GRP_WIDTH = 256.
[6]	QUALMASK[6]	Mask SIGQUAL[6]. These bit is only available if GRP_WIDTH = 256.
[5]	QUALMASK[5]	Mask SIGQUAL[5]. These bit is only available if GRP_WIDTH = 256.
[4]	QUALMASK[4]	Mask SIGQUAL[4]. These bit is only available if GRP_WIDTH = 256.
[3]	QUALMASK[3]	Mask SIGQUAL[3]. These bit is only available if GRP_WIDTH > 64.
[2]	QUALMASK[2]	Mask SIGQUAL[2]. These bit is only available if GRP_WIDTH > 64.
[1]	QUALMASK[1]	Mask SIGQUAL[1].
[0]	QUALMASK[0]	Mask SIGQUAL[0].

3.10.14 Qualifier Compare registers

The QUALCOMP Compare registers provide the data values with which the Input Qualifier Signals are compared.

The QUALCOMP<n> register characteristics are:

Usage constraints Writing when CTRL.RUN = 1 results in improper operation.

Configurations Available only in SRAM and ATB trace configurations. QUALCOMP5, QUALCOMP6, and QUALCOMP7 are only available when NUM_TRIG_STATES = 8.

Attributes See [3.9 Trigger State register summary on page 3-60.](#)

Qualifier signals include signals such as valid or ready handshakes. The comparison is always a match to value [GRP_WIDTH-32-1:0] QUALVALUE compare value for SIGQUAL[GRP_WIDTH-32-1:0]

Table 3-34 QUALCOMP<n> register bit assignments

Bits	Name	Function
[7]	QUALVALUE[7]	Compare value for SIGQUAL[7]. These bit is only available if GRP_WIDTH = 256.
[6]	QUALVALUE[6]	Compare value for SIGQUAL[6]. These bit is only available if GRP_WIDTH = 256.
[5]	QUALVALUE[5]	Compare value for SIGQUAL[5]. These bit is only available if GRP_WIDTH = 256.
[4]	QUALVALUE[4]	Compare value for SIGQUAL[4]. These bit is only available if GRP_WIDTH = 256.
[3]	QUALVALUE[3]	Compare value for SIGQUAL[3]. These bit is only available if GRP_WIDTH > 64.
[2]	QUALVALUE[2]	Compare value for SIGQUAL[2]. These bit is only available if GRP_WIDTH > 64.
[1]	QUALVALUE[1]	Compare value for SIGQUAL[1].
[0]	QUALVALUE[0]	Compare value for SIGQUAL[0].

3.10.15 Signal Mask registers

The Signal Mask registers are used to mask out specific *Signal Group* signals for *Trigger Signal* comparisons.

The SIGMASK<n> register characteristics are:

Usage constraints Writing when CTRL.RUN = 1 results in improper operation.

Configurations Available in all configurations. SIGMASK5, SIGMASK6, and SIGMASK7 are only available when NUM_TRIG_STATES = 8.

Attributes See [3.9 Trigger State register summary on page 3-60](#).

The following table shows the bit assignments.

————— **Note** —————

Each signal in the *Signal Group* is masked by clearing the appropriate bit.

Table 3-35 SIGMASK<n> register bit assignments

Bits	Name	Function
[31:0]	SIGMASK[31:0]	Mask bits from SIGCOMP[31:0].
[63:32]	SIGMASK[63:32]	Mask bits from SIGCOMP[63:32].
[95:64]	SIGMASK[95:64]	Mask bits from SIGCOMP[95:64]. These bits are only used if GRP_WIDTH = 128 or 256.
[127:96]	SIGMASK[127:96]	Mask bits from SIGCOMP[127:96]. These bits are only used if GRP_WIDTH = 128 or 256.
[159:128]	SIGMASK[159:128]	Mask bits from SIGCOMP[159:128]. These bits are only used if GRP_WIDTH = 256.
[191:160]	SIGMASK[191:160]	Mask bits from SIGCOMP[191:160]. These bits are only used if GRP_WIDTH = 256.
[223:192]	SIGMASK[223:192]	Mask bits from SIGCOMP[223:192]. These bits are only used if GRP_WIDTH = 256.
[255:224]	SIGMASK[255:224]	Mask bits from SIGCOMP[255:224]. These bits are only used if GRP_WIDTH = 256.

Related concepts

[2.5 Triggering on page 2-37](#).

Related references

[3.10.16 Signal Compare registers on page 3-80](#).

3.10.16 Signal Compare registers

The Signal Compare registers provide the data values with which the *Signal Group* signals are compared.

The SIGCOMP<n> register characteristics are:

Usage constraints Writing when CTRL.RUN = 1 results in improper operation.

Configurations Available in all configurations. SIGCOMP5, SIGCOMP6, and SIGCOMP7 are only available when NUM_TRIG_STATES = 8.

Attributes See [3.9 Trigger State register summary on page 3-60](#).

The following table shows the bit assignments.

Table 3-36 SIGCOMP<n> register bit assignments

Bits	Name	Function
[31:0]	SIGCOMP[31:0]	Compare value for <i>Signal Group</i> signals[31:0].
[63:32]	SIGCOMP[63:32]	Compare value for <i>Signal Group</i> signals[63:32].
[95:64]	SIGCOMP[95:64]	Compare value for <i>Signal Group</i> signals[95:64]. These bits are only used if GRP_WIDTH = 128 or 256.
[127:96]	SIGCOMP[127:96]	Compare value for <i>Signal Group</i> signals[127:96]. These bits are only used if GRP_WIDTH = 128 or 256.
[159:128]	SIGCOMP[159:128]	Compare value for <i>Signal Group</i> signals[159:128]. These bits are only used if GRP_WIDTH = 256.
[191:160]	SIGCOMP[191:160]	Compare value for <i>Signal Group</i> signals[191:160]. These bits are only used if GRP_WIDTH = 256.
[223:192]	SIGCOMP[223:192]	Compare value for <i>Signal Group</i> signals[223:192]. These bits are only used if GRP_WIDTH = 256.
[255:224]	SIGCOMP[255:224]	Compare value for <i>Signal Group</i> signals[255:224]. These bits are only used if GRP_WIDTH = 256.

Related concepts

[2.5 Triggering on page 2-37](#).

Related references

[3.10.15 Signal Mask registers on page 3-79](#).

3.11 Integration Mode register summary

This section gives a summary of the ELA-600 Integration Mode registers.

The following table shows the integration mode registers in offset order from the base address of the ELA-600.

Table 3-37 Integration mode registers summary

Offset	Name	Type	Reset	Description
0xEE8	ITTRIGOUT	WO	0x00	3.12.1 Integration Mode Action Trigger Output register on page 3-82
0xEEC	ITATBDATA	WO	0bx	3.12.2 Integration Test ATB Data register on page 3-82
0xEF0	ITATBCTR1	RO	0bx	3.12.3 Integration Test ATB Control register 1 on page 3-83
0xEF4	ITATBCTR0	WO	0bx	3.12.4 Integration Test ATB Control register 0 on page 3-83
0xEF8	ITTRIGIN	RO	-	3.12.5 Integration Mode External Trigger Input register on page 3-84
0xF00	ITCTRL	RW	0b0	3.12.6 Integration Mode Control register on page 3-84

3.12 Integration Mode register descriptions

This section describes the ELA-600 Integration Mode registers.

[Table 3-37 Integration mode registers summary on page 3-81](#) provides cross-references to individual registers.

This section contains the following subsections:

- [3.12.1 Integration Mode Action Trigger Output register on page 3-82.](#)
- [3.12.2 Integration Test ATB Data register on page 3-82.](#)
- [3.12.3 Integration Test ATB Control register 1 on page 3-83.](#)
- [3.12.4 Integration Test ATB Control register 0 on page 3-83.](#)
- [3.12.5 Integration Mode External Trigger Input register on page 3-84.](#)
- [3.12.6 Integration Mode Control register on page 3-84.](#)

3.12.1 Integration Mode Action Trigger Output register

The Integration Mode Action Trigger Output register drives values on the *Output Actions*.

The ITTRIGOUT register characteristics are:

Usage constraints No access when CTRL.RUN = 1.

Configurations Available in all configurations.

Attributes See [3.11 Integration Mode register summary on page 3-81](#).

The following table shows the bit assignments.

Table 3-38 ITTRIGOUT register bit assignments

Bits	Name	Function
[7:4]	ELAOUTPUT	Value to drive on ELAOUTPUT[3:0] when ITCTRL.IME = 1.
[3]	Reserved	-
[2]	STOPCLOCK	Level to drive on STOPCLOCK when ITCTRL.IME = 1. 0b0 Drive 0 on STOPCLOCK. 0b1 Drive 1 on STOPCLOCK.
[1:0]	CTTRIGOUT	Value to drive on CTTRIGOUT[1:0] when ITCTRL.IME = 1.

3.12.2 Integration Test ATB Data register

The ITATBDATA register controls signal outputs when ITCTRL.IME is set. This register is only supported when TRACE_GEN = 2.

The ITATBDATA register characteristics are:

Usage constraints None.

Configurations Available only in the ATB configuration.

Attributes See [3.11 Integration Mode register summary on page 3-81](#).

The following table shows the bit assignments.

Table 3-39 ITATBCTR0 register bit assignments

Bits	Name	Function
[31:17]	Reserved	-
[16]	ATDATAM[127]	Drives the ATDATAM[127] output.

Table 3-39 ITATBCTR0 register bit assignments (continued)

Bits	Name	Function
[15]	ATDATAM[119]	Drives the ATDATAM[119] output.
[14]	ATDATAM[111]	Drives the ATDATAM[111] output.
[13]	ATDATAM[103]	Drives the ATDATAM[103] output.
[12]	ATDATAM[95]	Drives the ATDATAM[95] output.
[11]	ATDATAM[87]	Drives the ATDATAM[87] output.
[10]	ATDATAM[79]	Drives the ATDATAM[79] output.
[9]	ATDATAM[71]	Drives the ATDATAM[71] output.
[8]	ATDATAM[63]	Drives the ATDATAM[63] output.
[7]	ATDATAM[55]	Drives the ATDATAM[55] output.
[6]	ATDATAM[47]	Drives the ATDATAM[47] output.
[5]	ATDATAM[39]	Drives the ATDATAM[39] output.
[4]	ATDATAM[31]	Drives the ATDATAM[31] output.
[3]	ATDATAM[23]	Drives the ATDATAM[23] output.
[2]	ATDATAM[15]	Drives the ATDATAM[15] output.
[1]	ATDATAM[7]	Drives the ATDATAM[7] output.
[0]	ATDATAM[0]	Drives the ATDATAM[0] output.

3.12.3 Integration Test ATB Control register 1

The ITATBCR1 register enables the values of signal inputs to be read when bit[0] of the Integration Mode Control Register is set. This register is only supported when TRACE_GEN = 2.

The ITATBCR1 register characteristics are:

Usage constraints None.

Configurations Available only in the ATB configuration.

Attributes See [3.11 Integration Mode register summary](#) on page 3-81.

The following table shows the bit assignments.

Table 3-40 ITATBCTR0 register bit assignments

Bits	Name	Function
[31:2]	Reserved	-
[1]	AFVALIDM	Returns the value of the AFVALIDM input.
[0]	ATREADYM	Returns the value of ATREADYM input.

3.12.4 Integration Test ATB Control register 0

The ITATBCTR0 register controls signal outputs when TRCITCTRL.IME is set. This register is only supported when TRACE_GEN = 2.

The ITATBCTR0 register characteristics are:

Usage constraints None.

Configurations Available only in the ATB configuration.
Attributes See [3.11 Integration Mode register summary](#) on page 3-81.

The following table shows the bit assignments.

Table 3-41 ITATBCTR0 register bit assignments

Bits	Name	Function
[31:12]	Reserved	-
[11:8]	ATBYTES[3:0]	Drives the ATBYTESM outputs [3:0].
[7:2]	Reserved	-
[1]	AFREADYM	Drives the AFREADYM output.
[0]	ATVALIDM	Drives the ATVALIDM output.

Note

ATBCTRL.ATID_VALUE is driven on **ATID[6:0]** after **IME=1** is written. The reset value for ATBCTRL.ATID_VALUE is 0bxxxxxx. **AFREADY** and **ATVALID** must be written back to zero before writing **IME=0** to resume functional operation or false data transactions could occur or ATB flush may not operate.

3.12.5 Integration Mode External Trigger Input register

The Integration Mode External Trigger Input register captures the values on the eight trigger inputs.

The ITTRIGIN register characteristics are:

Usage constraints No access when CTRL.RUN = 1.
Configurations Available in all configurations.
Attributes See [3.11 Integration Mode register summary](#) on page 3-81.

The following table shows the bit assignments.

Table 3-42 ITTRIGIN register bit assignments

Bits	Name	Function
[7:2]	EXTTRIG	Captures the value on EXTTRIG[5:0] when ITCTRL.IME = 1.
[1:0]	CTTRIGIN	Captures the value on CTTRIGIN[1:0] when ITCTRL.IME = 1.

Note

If the parameter TRIGIN_EDGE = 0, then **CTTRIGIN** and **EXTTRIG** must be held steady when reading ITTRIGIN. If TRIGIN_EDGE = 1, and if ICTRL.IME = 1, a rising-edge on **CTTRIGIN** or **EXTTRIGIN** is latched until ICTRL.IME = 0.

3.12.6 Integration Mode Control register

The Integration Mode control register enables testing of the eight external trigger inputs and the eight output actions.

The ITCTRL register characteristics are:

Usage constraints No access when CTRL.RUN = 1.

Configurations Available in all configurations.

Attributes See [3.11 Integration Mode register summary](#) on page 3-81.

The following table shows the bit assignments.

Table 3-43 ITCTRL register bit assignments

Bits	Name	Function
[0]	IME	Integration Mode enable. 0b0 Integration Mode disabled. The ELA-600 operates normally. 0b1 Integration Mode enabled when CTRL.RUN = 0.

3.13 Authentication register summary

This section gives a summary of the ELA-600 Authentication registers.

The following table shows the Authentication registers in offset order from the base address of the ELA-600.

Table 3-44 Authentication registers summary

Offset	Name	Type	Reset	Description
0xFB8	AUTHSTATUS	RO	-	3.14.1 Authentication Status register on page 3-87

3.14 Authentication register descriptions

This section describes the ELA-600 Authentication registers.

[Table 3-44 Authentication registers summary on page 3-86](#) provides cross-references to individual registers.

3.14.1 Authentication Status register

The Authentication Status register returns the status of the authentication signals **DBGEN**, **NIDEN**, **SPIDEN**, and **SPNIDEN**.

The AUTHSTATUS register characteristics are:

Usage constraints No usage constraints.

Configurations Available in all configurations.

Attributes See [3.13 Authentication register summary on page 3-86](#).

The following table shows the bit assignments.

Table 3-45 AUTHSTATUS bit assignments

Bits	Name	Function
[7:6]	SNID	Secure, non-invasive debug. 0b10 Debug disabled. 0b11 Debug enabled.
[5:4]	SID	Secure, invasive debug. 0b10 Debug disabled. 0b11 Debug enabled.
[3:2]	NSNID	Non-secure, non-invasive debug. 0b10 Debug disabled. 0b11 Debug enabled.
[1:0]	NSID	Non-secure, invasive debug. 0b10 Debug disabled. 0b11 Debug enabled.

See the *Arm® CoreSight™ Architecture Specification* for more information.

Related concepts

[2.6 Authentication interface on page 2-42](#).

3.15 Device register summary

This section gives a summary of the ELA-600 Device registers.

The following table shows the device registers in offset order from the base address of the ELA-600^c.

Table 3-46 Device registers summary

Offset	Name	Type	Reset	Description
0xFBC	DEVARCH	RO	0x47710A75	3.16.1 Device Architecture register on page 3-89
0xFC0	DEVID2	RO	Configuration-dependent	3.16.2 Device Configuration register 2 on page 3-89
0xFC4	DEVID1	RO	Configuration-dependent	3.16.3 Device Configuration register 1 on page 3-90
0xFC8	DEVID	RO	Configuration-dependent	3.16.4 Device Configuration register on page 3-90
0xFCC	DEVTYPE	RO	0x75	3.16.5 Device Type Identifier register on page 3-91

^c Configuration-dependent.

3.16 Device register descriptions

This section describes the ELA-600 Device registers.

[Table 3-46 Device registers summary on page 3-88](#) provides cross-references to individual registers.

This section contains the following subsections:

- [3.16.1 Device Architecture register on page 3-89](#).
- [3.16.2 Device Configuration register 2 on page 3-89](#).
- [3.16.3 Device Configuration register 1 on page 3-90](#).
- [3.16.4 Device Configuration register on page 3-90](#).
- [3.16.5 Device Type Identifier register on page 3-91](#).

3.16.1 Device Architecture register

The Device Architecture register returns the architect and architecture of the ELA-600.

The DEVARCH register characteristics are:

Usage constraints	No usage constraints.
Configurations	Available in all configurations.
Attributes	See 3.15 Device register summary on page 3-88 .

The following table shows the bit assignments.

Table 3-47 DEVARCH bit assignments

Bits	Name	Function
[31:21]	ARCHITECT	The architect of the device. 0x23B Arm.
[20]	PRESENT	Indicates that the register is present. 1 Register present.
[19:16]	REVISION	Architecture revision. 1 First revision.
[15:0]	ARCHID	The architecture of the device. 0x0A75 CoreSight ELA.

See the *Arm® CoreSight™ Architecture Specification* for more information.

3.16.2 Device Configuration register 2

The Device Configuration register 2 provides configuration information about the ELA-600.

The DEVID2 register characteristics are:

Usage constraints	No usage constraints.
Configurations	Available in all configurations.
Attributes	See 3.15 Device register summary on page 3-88 .

The following table shows the bit assignments.

Table 3-48 DEVID2 bit assignments

Bits	Name	Function
[31:20]	-	Reserved
[19:16]	TRIGIN_EDGE	0 Level detect of CTTRIGIN and EXTTRIG . 1 Single edge detect of CTTRIGIN and EXTTRIG .
[15:8]	COMP_WIDTH	Indicates the comparator width. 0 Comparator width = GRP_WIDTH. >0 Comparator width = (COMP_WIDTH + 1) x 8. For example, if COMP_WIDTH = 15, then comparator width = 128.
[7:0]	ALTTS	Indicates whether <i>Alternate Trace Select</i> (ALTTS) is implemented or not. 0x00 ALTTS is not implemented. 0x10 ALTTS is implemented and can be used for independent trace of <i>Trigger State 4</i> . 0x80 ALTTS is implemented and can be used for independent trace of <i>Trigger State 7</i> . All other encodings are reserved and read as 0x00. See 2.4.3 Simultaneous trace using the highest Trigger State number on page 2-29 for more information.

3.16.3 Device Configuration register 1

The Device Configuration register 1 provides configuration information about the ELA-600.

The DEVID1 register characteristics are:

Usage constraints No usage constraints.

Configurations Available in all configurations.

Attributes See [3.15 Device register summary](#) on page 3-88.

The following table shows the bit assignments.

Table 3-49 DEVID1 bit assignments

Bits	Name	Function
[31:24]	COUNTWIDTH	Counter width in bits. Fixed at 32.
[23:16]	NUMTRIGSTATES	Number of <i>Trigger States</i> . Five or eight.
[15:8]	SIGGRPWIDTH	<i>Signal Group</i> width. The field value is (<i>Signal Group</i> width/8) - 1. For example, 7 if GRP_WIDTH = 64, 15 if GRP_WIDTH = 128, and 31 if GRP_WIDTH = 256.
[7:0]	NUMSIGGRPS	Number of <i>Signal Groups</i> . Fixed at 12.

3.16.4 Device Configuration register

The Device Configuration register provides configuration information about the ELA-600.

The DEVID register characteristics are:

Usage constraints No usage constraints.

Configurations Available in all configurations.

Attributes See [3.15 Device register summary](#) on page 3-88.

The following table shows the bit assignments.

Table 3-50 DEVID bit assignments

Bits	Name	Function
[31:29]	-	Reserved
[28:25]	SCRAMBLER	0 Trace read data scrambler not present. 1 Trace read data scrambler present. Only available with GRP_WIDTH=256 and SRAM configuration.
[24:20]	ID_CAPTURE_SIZE	2-30 bits when ID_CAPTURE_GEN = 1, or 0 otherwise.
[19:16]	COND_TRIG	Shows the value of the COND_TRIG parameter. 1 Fixed at 1. Conditional trigger states are always enabled on the ELA-600.
[15:8]	SRAM_ADDR_SIZE	SRAM address width in bits.
[7:4]	TRACEFORMAT	Trace implementation: 0 Not supported. 1 Indicates SRAM trace configuration using trace header format revision 1. 2 Indicates ATB trace format that supports compression and zero byte removal.
[3:0]	TRACETYPE	ATB trace: 0 ATB trace not implemented. 1 ATB trace is implemented.

3.16.5 Device Type Identifier register

The Device Type Identifier register returns the device type.

The DEVTYPE register characteristics are:

Usage constraints No usage constraints.

Configurations Available in all configurations.

Attributes See [3.15 Device register summary](#) on page 3-88.

The following table shows the bit assignments.

Table 3-51 DEVTYPE bit assignments

Bits	Name	Function
[7:0]	DEVTYPE	0x75. SUB type = 0x7. MAJOR type = 0x5.

3.17 ID register summary

This section gives a summary of the ELA-600 ID registers.

The following table shows the device registers in offset order from the base address of the ELA-600.

Table 3-52 ID registers summary

Offset	Name	Type	Reset	Description
0xFD0	PIDR4	RO	0x04	3.18.1 Peripheral ID4 Register on page 3-93
0xFD4	PIDR5	RO	0x00	3.18.2 Peripheral ID5 Register on page 3-93
0xFD8	PIDR6	RO	0x00	3.18.3 Peripheral ID6 Register on page 3-93
0xFDC	PIDR7	RO	0x00	3.18.4 Peripheral ID7 Register on page 3-94
0xFE0	PIDR0	RO	0xD0	3.18.5 Peripheral ID0 Register on page 3-94
0xFE4	PIDR1	RO	0xB9	3.18.6 Peripheral ID1 Register on page 3-94
0xFE8	PIDR2	RO	0x0B	3.18.7 Peripheral ID2 Register on page 3-95
0xFEC	PIDR3	RO	0x00	3.18.8 Peripheral ID3 Register on page 3-95
0xFF0	CIDR0	RO	0x0D	3.18.9 Component ID0 Register on page 3-95
0xFF4	CIDR1	RO	0x90	3.18.10 Component ID1 Register on page 3-96
0xFF8	CIDR2	RO	0x05	3.18.11 Component ID2 Register on page 3-96
0xFFC	CIDR3	RO	0xB1	3.18.12 Component ID3 Register on page 3-96

3.18 ID register descriptions

This section describes the ELA-600 ID registers.

Table 3-52 ID registers summary on page 3-92 provides cross-references to individual registers.

This section contains the following subsections:

- *3.18.1 Peripheral ID4 Register on page 3-93.*
- *3.18.2 Peripheral ID5 Register on page 3-93.*
- *3.18.3 Peripheral ID6 Register on page 3-93.*
- *3.18.4 Peripheral ID7 Register on page 3-94.*
- *3.18.5 Peripheral ID0 Register on page 3-94.*
- *3.18.6 Peripheral ID1 Register on page 3-94.*
- *3.18.7 Peripheral ID2 Register on page 3-95.*
- *3.18.8 Peripheral ID3 Register on page 3-95.*
- *3.18.9 Component ID0 Register on page 3-95.*
- *3.18.10 Component ID1 Register on page 3-96.*
- *3.18.11 Component ID2 Register on page 3-96.*
- *3.18.12 Component ID3 Register on page 3-96.*

3.18.1 Peripheral ID4 Register

The Peripheral ID4 Register returns byte[4] of the Peripheral ID.

The PIDR4 characteristics are:

Usage constraints	No usage constraints.
Configurations	Available in all configurations.
Attributes	See <i>3.17 ID register summary on page 3-92.</i>

The following table shows the bit assignments.

Table 3-53 PIDR4 bit assignments

Bits	Name	Function
[7:4]	SIZE	0x0. One 4KB count.
[3:0]	DES_2	0x4. JEP continuation code for Arm.

3.18.2 Peripheral ID5 Register

The Peripheral ID5 Register returns byte[5] of the Peripheral ID.

The PIDR5 characteristics are:

Usage constraints	No usage constraints.
Configurations	Available in all configurations.
Attributes	See <i>3.17 ID register summary on page 3-92.</i>

The following table shows the bit assignments.

Table 3-54 PIDR5 bit assignments

Bits	Name	Function
[7:0]	PIDR5	0x00. Reserved.

3.18.3 Peripheral ID6 Register

The Peripheral ID6 Register returns byte[6] of the Peripheral ID.

The PIDR6 characteristics are:

- Usage constraints** No usage constraints.
- Configurations** Available in all configurations.
- Attributes** See [3.17 ID register summary on page 3-92](#).

The following table shows the bit assignments.

Table 3-55 PIDR6 bit assignments

Bits	Name	Function
[7:0]	PIDR6	0x00. Reserved.

3.18.4 Peripheral ID7 Register

The Peripheral ID7 Register returns byte[7] of the Peripheral ID.

The PIDR7 characteristics are:

- Usage constraints** No usage constraints.
- Configurations** Available in all configurations.
- Attributes** See [3.17 ID register summary on page 3-92](#).

The following table shows the bit assignments.

Table 3-56 PIDR7 bit assignments

Bits	Name	Function
[7:0]	PIDR7	0x00. Reserved.

3.18.5 Peripheral ID0 Register

The Peripheral ID0 Register returns byte[0] of the Peripheral ID.

The PIDR0 characteristics are:

- Usage constraints** No usage constraints.
- Configurations** Available in all configurations.
- Attributes** See [3.17 ID register summary on page 3-92](#).

The following table shows the bit assignments.

Table 3-57 PIDR0 bit assignments

Bits	Name	Function
[7:0]	PART_0	0xD0. Bits[7:0] of part number 0x9D0.

3.18.6 Peripheral ID1 Register

The Peripheral ID1 Register returns byte[1] of the Peripheral ID.

The PIDR1 characteristics are:

- Usage constraints** No usage constraints.
- Configurations** Available in all configurations.
- Attributes** See [3.17 ID register summary on page 3-92](#).

The following table shows the bit assignments.

Table 3-58 PIDR1 bit assignments

Bits	Name	Function
[7:4]	DES_0	0x8. Bits[3:0] of JEP106 identification code for Arm 0x3B.
[3:0]	PART_1	0x9. Bits[11:8] of part number 0x9D0.

3.18.7 Peripheral ID2 Register

The Peripheral ID2 Register returns byte[2] of the Peripheral ID.

The PIDR2 characteristics are:

Usage constraints No usage constraints.

Configurations Available in all configurations.

Attributes See [3.17 ID register summary on page 3-92](#).

The following table shows the bit assignments.

Table 3-59 PIDR2 bit assignments

Bits	Name	Function
[7:4]	REVISION	0x0. Revision number. Indicates revision r0p0.
[3]	JEDEC	0b1. Fixed at 0b1.
[2:0]	DES_1	0b011. Bits[6:4] of JEP106 identification code for Arm 0x3B.

3.18.8 Peripheral ID3 Register

The Peripheral ID3 Register returns byte[3] of the Peripheral ID.

The PIDR3 characteristics are:

Usage constraints No usage constraints.

Configurations Available in all configurations.

Attributes See [3.17 ID register summary on page 3-92](#).

The following table shows the bit assignments.

Table 3-60 PIDR3 bit assignments

Bits	Name	Function
[7:4]	REVAND	0x0. RevAnd.
[3:0]	CMOD	0x0. Indicates whether the customer has modified the behavior of the component. Usually, this field is 0000. You can change this value when you make authorized modifications to this component.

3.18.9 Component ID0 Register

The Component ID0 Register returns byte[0] of the Component ID.

The CIDR0 characteristics are:

Usage constraints No usage constraints.

Configurations Available in all configurations.

Attributes See [3.17 ID register summary on page 3-92](#).

The following table shows the bit assignments.

Table 3-61 CIDR0 bit assignments

Bits	Name	Function
[7:0]	PRMBL_0	0x0D. Preamble.

3.18.10 Component ID1 Register

The Component ID1 Register returns byte[1] of the Component ID.

The CIDR1 characteristics are:

- Usage constraints** No usage constraints.
- Configurations** Available in all configurations.
- Attributes** See [3.17 ID register summary on page 3-92](#).

The following table shows the bit assignments.

Table 3-62 CIDR1 bit assignments

Bits	Name	Function
[7:4]	CLASS	0x9. Indicates a CoreSight component.
[3:0]	PRMBL_1	0x0. Preamble.

3.18.11 Component ID2 Register

The Component ID2 Register returns byte[2] of the Component ID.

The CIDR2 characteristics are:

- Usage constraints** No usage constraints.
- Configurations** Available in all configurations.
- Attributes** See [3.17 ID register summary on page 3-92](#).

The following table shows the bit assignments.

Table 3-63 CIDR2 bit assignments

Bits	Name	Function
[7:0]	PRMBL_2	0x05. Preamble.

3.18.12 Component ID3 Register

The Component ID3 Register returns byte[3] of the Component ID.

The CIDR3 characteristics are:

- Usage constraints** No usage constraints.
- Configurations** Available in all configurations.
- Attributes** See [3.17 ID register summary on page 3-92](#).

The following table shows the bit assignments.

Table 3-64 CIDR3 bit assignments

Bits	Name	Function
[7:0]	PRMBL_3	0x81. Preamble.

Appendix A

Signal descriptions

This appendix describes the external signals of the ELA-600.

It contains the following sections:

- *A.1 Clocks and reset* on page Appx-A-98.
- *A.2 Debug APB signals* on page Appx-A-99.
- *A.3 Observation interface signals* on page Appx-A-100.
- *A.4 Timestamp interface signals* on page Appx-A-102.
- *A.5 Authentication interface signals* on page Appx-A-103.
- *A.6 DFT and MBIST interface signals* on page Appx-A-104.
- *A.7 Q-Channel Low-Power Interface signals* on page Appx-A-105.
- *A.8 Output Action signals* on page Appx-A-106.
- *A.9 External Trigger Input signals* on page Appx-A-107.
- *A.10 AMBA ATBv4 Interface* on page Appx-A-108.

A.1 Clocks and reset

The ELA-600 has a single clock and reset signal.

The following table shows the ELA-600 clock and reset signals.

Table A-1 ELA-600 clock and reset signals

Signal name	Type	Description
ELACK	Input	Logic analyzer clock for triggering and trace.
RESETn		Reset for ELACK domain including <i>Output Actions</i> .

A.2 Debug APB signals

The following table shows the ELA-600 debug AMBA3 APB signals. All the signals are in the **ELACLK** clock domain.

Table A-2 ELA-600 debug APB signals

Signal name	Type	Description	Connection information
PSELDBG	Input	Select.	Connect to the CoreSight debug subsystem. An ADB bridge is needed if the APB interface runs on a clock that is different from ELACLK .
PENABLEDBG		Enable.	
PWRITEDBG		Peripheral write.	
PADDRDBG[11:2]		Address. The ELA-600 only supports word-aligned addresses.	
PWDATADBG[31:0]		Write data.	
PREADYDBG	Output	Peripheral ready.	
PRDATADBG[31:0]		Read data.	

A.3 Observation interface signals

The following table shows the ELA-600 Observation interface signals. All the signals are in the ELACLK clock domain.

————— **Note** —————

The number of signals in each SIGNALGRP can be 64, 128, or 256, depending on GRP_WIDTH.

Table A-3 ELA-600 Observation interface signals

Signal name	Type	Description	Connection information
SIGNALGRP0[GRP_WIDTH-1:0]	Input	Signal Group 0 debug signals.	Connect to IP debug signals.
SIQUAL0[GRPWIDTH/32-1:0]	Input	Signal Group 0 qualifier signals.	
SIGCLKEN0	Input	ELACLK clock enable for SIGNALGRP0.	
SIGNALGRP1[GRP_WIDTH-1:0]	Input	Signal Group 1 debug signals.	
SIQUAL1[GRPWIDTH/32-1:0]	Input	Signal Group 1 qualifier signals.	
SIGCLKEN1	Input	ELACLK clock enable for SIGNALGRP1.	
SIGNALGRP2[GRP_WIDTH-1:0]	Input	Signal Group 2 debug signals.	
SIQUAL2[GRPWIDTH/32-1:0]	Input	Signal Group 2 qualifier signals.	
SIGCLKEN2	Input	ELACLK clock enable for SIGNALGRP2.	
SIGNALGRP3[GRP_WIDTH-1:0]	Input	Signal Group 3 debug signals.	
SIQUAL3[GRPWIDTH/32-1:0]	Input	Signal Group 3 qualifier signals.	
SIGCLKEN3	Input	ELACLK clock enable for SIGNALGRP3.	
SIGNALGRP4[GRP_WIDTH-1:0]	Input	Signal Group 4 debug signals.	
SIQUAL4[GRPWIDTH/32-1:0]	Input	Signal Group 4 qualifier signals.	
SIGCLKEN4	Input	ELACLK clock enable for SIGNALGRP4.	
SIGNALGRP5[GRP_WIDTH-1:0]	Input	Signal Group 5 debug signals.	
SIQUAL5[GRPWIDTH/32-1:0]	Input	Signal Group 5 qualifier signals.	
SIGCLKEN5	Input	ELACLK clock enable for SIGNALGRP5.	
SIGNALGRP6[GRP_WIDTH-1:0]	Input	Signal Group 6 debug signals.	
SIQUAL6[GRPWIDTH/32-1:0]	Input	Signal Group 6 qualifier signals.	
SIGCLKEN6	Input	ELACLK clock enable for SIGNALGRP6.	
SIGNALGRP7[GRP_WIDTH-1:0]	Input	Signal Group 7 debug signals.	
SIQUAL7[GRPWIDTH/32-1:0]	Input	Signal Group 7 qualifier signals.	
SIGCLKEN7	Input	ELACLK clock enable for SIGNALGRP7.	
SIGNALGRP8[GRP_WIDTH-1:0]	Input	Signal Group 8 debug signals.	
SIQUAL8[GRPWIDTH/32-1:0]	Input	Signal Group 8 qualifier signals.	
SIGCLKEN8	Input	ELACLK clock enable for SIGNALGRP8.	
SIGNALGRP9[GRP_WIDTH-1:0]	Input	Signal Group 9 debug signals.	

Table A-3 ELA-600 Observation interface signals (continued)

Signal name	Type	Description	Connection information
SIQUAL9[GRPWIDTH/32-1:0]	Input	<i>Signal Group 9</i> qualifier signals.	
SIGCLKEN9	Input	ELACLK clock enable for SIGNALGRP9 .	
SIGNALGRP10[GRP_WIDTH-1:0]	Input	<i>Signal Group 10</i> debug signals.	
SIQUAL10[GRPWIDTH/32-1:0]	Input	<i>Signal Group 10</i> qualifier signals.	
SIGCLKEN10	Input	ELACLK clock enable for SIGNALGRP10 .	
SIGNALGRP11[GRP_WIDTH-1:0]	Input	<i>Signal Group 11</i> debug signals.	
SIQUAL11[GRPWIDTH/32-1:0]	Input	<i>Signal Group 11</i> qualifier signals.	
SIGCLKEN11	Input	ELACLK clock enable for SIGNALGRP11 .	

A.4 Timestamp interface signals

The following table shows the ELA-600 timestamp interface signals. The input value must be synchronized into the **ELACLK** clock domain. This signal is only available with the SRAM or ATB trace configurations.

Table A-4 ELA-600 timestamp interface signals

Signal name	Type	Description	Connection information
TSVALUE[63:0]	Input	Timestamp value, encoded as a natural binary number.	From Timestamp Generator or decoder.

A.5 Authentication interface signals

The following table shows the ELA-600 Authentication interface signals. All the signals must be synchronized into the **ELACLK** clock domain.

Table A-5 ELA-600 Authentication interface signals

Signal name	Type	Description	Connection information
DBGEN	Input	Invasive debug enable.	From CoreSight debug subsystem authentication control signals.
NIDEN		Non-invasive debug enable.	
SPIDEN		Secure invasive debug enable.	
SPNIDEN		Secure non-invasive debug enable.	

A.6 DFT and MBIST interface signals

The following table shows the ELA-600 SRAM BIST interface signals. All the signals are in the **ELACLK** clock domain.

Note

The signals are only available in the SRAM trace configuration..

Table A-6 ELA-600 DFT and MBIST interface signals

Signal name	Type	Description	Connection information
DFTRAMHOLD		Disables the RAM chip select during scan shift.	Connect to scan DFT logic.
MBISTREQ		MBIST Mode Request. Enables MBIST testing.	Connect to MBIST controller.
nMBISTRESET		Resets functional logic to enable MBIST operations.	
MBISTADDR[RAM_ADDR_SIZE-1:0]	Input	Logical RAM address.	
MBISTINDATA[(GRP_WIDTH+8)-1:0]		RAM Write data.	
MBISTWRITEEN		Write enable control. A no-op occurs if write and read enables are both zero.	
MBISTREADEN		Read enable control. A no-op occurs if write and read enables are both zero.	
MBISTACK		Output	
MBISTOUTDATA[(GRP_WIDTH+8)-1:0]	RAM Read data.		

A.7 Q-Channel Low-Power Interface signals

The following table shows the ELA-600 Q-Channel Low-Power Interface signals.

All the signals are in the **ELACLK** clock domain.

Table A-7 ELA-600 Q-Channel Low-Power Interface signals

Signal name	Type	Description	Connection information
ELAQREQn	Input	Quiescence request from the clock controller to the ELA-600.	Connect to clock controller or power controller.
ELAQACCEPTn	Output	Quiescence request accept from the ELA-600.	
ELAQDENY		Quiescence request deny from the ELA-600.	
ELAQACTIVE		Indicates that the ELA-600 is active.	

A.8 Output Action signals

The following table shows the ELA-600 *Output Action* signals. All the signals are in the **ELACLK** clock domain.

Table A-8 ELA-600 *Output Action* signals

Signal name	Type	Description	Connection information
CTTRIGOUT[1:0]	Output	Trigger outputs.	Connect to external CTI inputs.
STOPCLOCK		Used to stop SoC clocks.	Connect to SoC clock control.
ELAOUTPUT[3:0]		General-purpose outputs.	Connect to GIC, external I/O, or register.

A.9 External Trigger Input signals

The following table shows the ELA-600 *External Trigger Input* Signals. All **CTTRIGIN** and **CTTRIGOUTACK** signals are internally synchronized into the **ELACLK** clock domain. All outputs are driven with **ELACLK**.

Table A-9 ELA-600 External Trigger Input signals

Signal name	Type	Description	Connection information
CTTRIGIN[1:0]		Trigger inputs.	From external CTI or CoreSight EVENT interface.
EXTTRIG[5:0]	Input	External inputs for non-debug signal <i>Trigger Condition</i> .	From other ELA-600 ELAOUTPUT signals or user-defined synchronized signals.

————— **Note** —————

CTTRIGIN and **EXTTRIGIN** are compared with *Trigger Signal Comparisons*. Arm recommends that a level is driven from the source connections to support the timing of the *Trigger Signal Comparison*. However, some cross-triggering sources might require detection of a one-time pulse.

Setting the parameter `TRIGIN_EDGE = 1` configures **CTTRIGIN** and **EXTTRIGIN** for detection of a single rising edge on each input when `CTRL.RUN = 1` or `ICTRL.IME = 1`. When a rising-edge is detected, **CTTRIGIN** and **EXTTRIGIN** can only be cleared when `CTRL.RUN = 0` or `ICTLR.IME = 0` is written.

A.10 AMBA ATBv4 Interface

The following table shows the ELA-600 *AMBA ATBv4 Interface* Signals.

Table A-10 ELA-600 AMBA ATBv4 Interface signals

Signal name	Type	Description	Connection information
ATREADYM	Input	Trace slave ready.	Connect to ATB trace infrastructure.
AFVALIDM	Input	Flush valid request.	Connect to ATB trace infrastructure.
ATVALIDM	Output	ATB master valid.	Connect to ATB trace infrastructure.
ATBYTES[3:0]	Output	ATBYTE transfer.	Connect to ATB trace infrastructure.
ATDATAM[127:0]	Output	Trace data.	Connect to ATB trace infrastructure.
ATID[6:0]	Output	Trace ID.	Connect to ATB trace infrastructure.
AFREADYM	Output	Flush ready.	Connect to ATB trace infrastructure.

————— **Note** —————

An ATB bridge is needed if the ATB trace infrastructure is not clocked by **ELACLK** source.

An ATB upsizer or downsizer is required to connect to ATB trace infrastructures that are not 128-bits wide.

—————

Appendix B

Revisions

This appendix describes the technical changes between released issues of this book.

It contains the following section:

- [B.1 Revisions on page Appx-B-110.](#)

B.1 Revisions

Each table lists the technical differences between successive issues of the document.

Table B-1 Issue 0000-00

Change	Location	Affects
First release	-	-

Table B-2 Issue 0000-01

Change	Location	Affects
Changed product name to ELA-600	Whole document	Whole document