# Ncore Trace and Debug Architecture Specification

Release: 1.0

Rev: 0.86, December 16, 2021

**Release Information**

| *Version* | *Editor* | *Change* | *Date* |
|---|---|---|---|
| **0.3** | MK | Initial Document from original slides form Michael Frank | 05/24/2021 |
| **0.5** | MK | First release for review | 06/06/2021 |
| **0.6** | MK/MF | Added user bits mask register and fixed typos (Tarce) | 06/11/2021 |
| **0.7** | MK | Updated the main overview figure.<br>Added clarification on register AND and OR operations<br>Added circular buffer setting<br>Added buffer threshold setting<br>Updated DTW capture format | 06/18/2021 |
| **0.8** | MK | Added DtwDbgReq and DtwDbgRsp details<br>Added CSR offsets | 07/25/2021 |
| **0.81** | MK | Fixed empty reset value in CSRs<br>Clarification on NDP fields captured<br>Timeout at unit level for building DTW<br>Added capture and drop events to capture and accumulate blocks<br>Added clarification on Native interface Trace signaling and usage<br>Removed unit level buffer threshold<br>Clarification on SysReq and SysRsp tracing | 08/05/2021 |
| **0.82** | MK | Removed circular buffer option for capture block | 08/06/2021 |
| **0.83** | MK | Updated the time stamp logic and the accumulation buffer read part | 08/15/2021 |
| **0.84** | MK | Changed TOPCR from single to 2 registers | 08/24/2021 |
| **0.85** | MK | Updated description of user bit setting<br>Updated rsp encoding | 11/23/2021 |
| **0.86** | MK | Update for the trace en  for NCAIU with AXI | 12/16/2021 |
| | | | |
| **Legend:** | MK<br>MF<br>Xx | Mohammed<br>Michael Frank<br>Whoever else edited this document | |

**Confidential Proprietary Notice**

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: (i) for the purposes of determining whether implementations infringe any third party patents;(ii) for developing technology or products which avoid any of Arteris IP's intellectual property; or (iii) as a reference for modifying existing patents or patent applications or creating any continuation, continuation in part, or extension of existing patents or patent applications; or (iv) for generating data for publication or disclosure to third parties, which compares the performance or functionality of the Arteris IP technology described in this document with any other products created by you or a third party, without obtaining Arteris IP's prior written consent.

THIS DOCUMENT IS PROVIDED "AS IS". ARTERIS IP 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, Arteris IP 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. Arteris IP makes no representations or warranties against the risk or presence of same.

TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL ARTERIS IP 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 ARTERIS HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

This document consists solely of commercial items. You shall be solely 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 Arteris IP's customers is not intended to create or refer to any partnership relationship with any other company. Arteris IP 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 Arteris IP, 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 the document and any translation, the terms of the English version of the agreement shall prevail.



**Confidentiality Status**

This document is Confidential and Proprietary. This document may only be used and distributed in accordance with the terms of the agreement entered into by Arteris IP and the party that Arteris IP delivered this document to.

**Product Status**

The information in this document is ***Preliminary***.

**Web Address**

http://www.arteris.com

# Table of Contents

# Table of Figures

# Table of Tables

# Preface

This preface introduces the Arteris® Network-on-Chip Hierarchical Coherency Engine Architecture  Specification.

**About this document**

This technical document is for the Arteris Network-on-Chip Hierarchical Coherency Engine Architecture. It describes the subsystems and their function along with the system's interactions with the external subsystems. It also provides reference documentation and contains programming details for registers.

**Product revision status**

*TBD*

**Intended audience**

This manual is for system designers, system integrators, and programmers who are designing or programming a System-on-Chip (SoC) that uses or intend to use the Arteris Network-on-Chip Hierarchical Coherency System (ANoC-HCS).

**Using this document**

*TBD*

**Glossary**

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

**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 italic*

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.

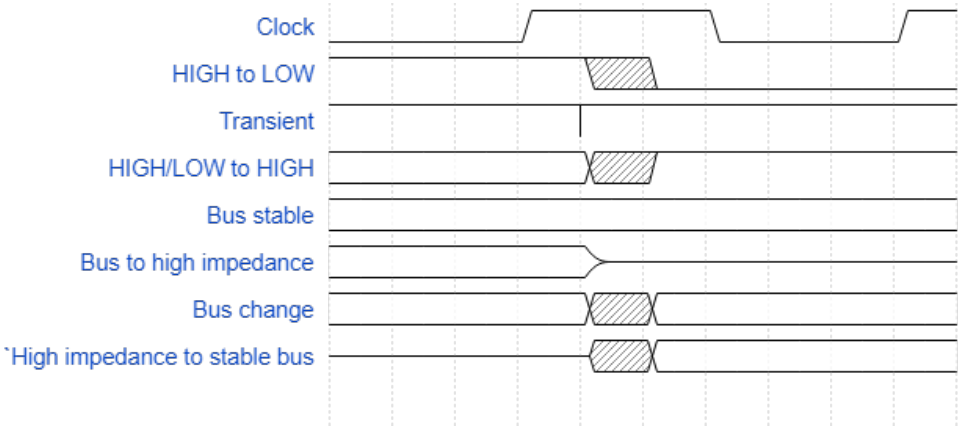
SMALL CAPITALS

Used in body text for a few terms that have specific technical meanings, that are defined in the Arteris® 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.




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

History of the World II, Mel Brooks.

# 1 Introduction

This specification describes the trace and debug architecture for Ncore. Tracing can help localizing problems and errors in a NoC which may result from functional and performance anomalies. It requires taking a snapshot of traffic in a live system with temporal order to understand what is going on. A system in general will need to implement following functionality:

- Label: Identify and select transactions based on a set criterion like address, target etc.
- Order: Identify temporal ordering of transactions, this requires a time stamp.
- Capture: Take a snapshot of the labeled transaction at a feasible location.
- Aggregate and access: Accumulate all captured transactions at a central location and provide SW access.
- Analysis: Software is required to access the captured information and analyze it.

This specification is specified with restrictions to minimize Maestro software changes and not to add any new wires to the network, this is more of an add on to current Ncore system which does have some limitations.

## 1.1 Parameters

Three new system level parameters are defined for trace and debug. Note that trace and debug feature is always present and cannot be disabled.

**Name:** nMainTraceBufSize **Type:** Int **Visibility:** User Settable

| | **Architecture** | | **Release** | | **Default** |
|---|---|---|---|---|---|
| | *Min* | *Max* | *Min* | *Max* | |
| **Value** | 32 | 4096 | 32 | 1024 | 64 |
| **Constraint** | | | | | |
| **Customer Description** | Specifies the number of trace entries in the buffer each entry is 72 bytes. | | | | |
| **Engineering Description** | | | | | |

TABLE 1 NMAINTRACEBUFSIZE PARAMETER

**Name:** nUnitTraceBufSize **Type:** Int **Visibility:** Engg

| | **Architecture** | | **Release** | | **Default** |
|---|---|---|---|---|---|
| | *Min* | *Max* | *Min* | *Max* | |
| **Value** | 8 | 32 | 8 | 16 | 8 |
| **Constraint** | | | | | |
| **Customer Description** | Specifies the number of trace entries in each Ncore unit, every is 64 bytes. | | | | |
| **Engineering Description** | | | | | |

TABLE 2 NUNITTRACEBUFSIZE PARAMETER

**Name:** nTraceRegisters **Type:** Int **Visibility:** User Settable

| | **Architecture** | | **Release** | | **Default** |
|---|---|---|---|---|---|
| | *Min* | *Max* | *Min* | *Max* | |
| **Value** | 1 | 8 | 1 | 4 | 1 |

| | |
|---|---|
| **Constraint** | |
| **Customer Description** | Specifies the number of trace register sets present in the system per AIU |
| **Engineering Description** | |

TABLE 3 NUNITTRACEBUFSIZE PARAMETER

### 1.1.1 Register offset

Debug and trace registers described in this document are located at 0x900 offset within each Ncore unit.

### 1.1.2 Ncore Debug message fields and format

Debug data is carried over DtwDbgReq and corresponding response is DtwDbgRsp. These message fields are shown in Table 4 & Table 5.

| Field | Paramter | Description |
|---|---|---|
| **Header (SMI header)** | | |
| Target id | wTargetId | Funit ID for DVE |
| Initiator id | wInitiatorId | Funit ID of the initialing Ncore unit |
| cm type | wCmType | 0xA0 |
| Message id | wMsgId | Unique message ID |
| H prot | wHProt | Header protection Parity or ECC |
| T tier | wTTier | Tier (not used currently) |
| Steering | wSteering | Steering (not used currently) |
| Priority | wPriority | Set to lowest priority (not used currently) |
| Ql | wQl | QoS label (not used currently) |
| **Non data payload (concerto header)** | | |
| M prot | wMProt | Message protection Parity or ECC |
| Aux | wNdpAux | Optional auxiliary bits |
| Rl | wRL | Response level set as [01] |
| Cm status | wCMStatus | CM status set as zeros |
| Tm | wTM | Trace me bit always set to zero |
| **Data (data payload)** | | |
| Data | wData | Data width set based on Ncore unit data width 64/128 or 256 |
| D bad | wDBad | DW bad data indicator (set if storage memory was corrupted) |
| D wid | wDwId | DW identifier |
| D prot | wDProt | DW Data protection Parity or ECC |
| Be | wBE | DW Byte enables |

TABLE 4 DTWDBGREQ FIELDS

| Field | Paramter | Description |
|---|---|---|
| **Header (SMI header)** | | |
| Target id | wTargetId | Funit ID of the Ncore unit Initiator Id from the DtwDbgReq |
| Initiator id | wInitiatorId | Funit ID of the initialing DVE |
| cm type | wCmType | 0xFF |
| Message id | wMsgId | Unique message ID |
| H prot | wHProt | Header protection Parity or ECC |
| T tier | wTTier | Tier (not used currently) |
| Steering | wSteering | Steering (not used currently) |
| Priority | wPriority | Set to lowest priority (not used currently) |
| Ql | wQl | QoS label (not used currently) |
| **Non data payload (concerto header)** | | |
| M prot | wMProt | Message protection Parity or ECC |
| Rl | wRL | Response level set as [00] |
| Cm status | wCMStatus | CM status set as described in Table 6 |
| R message ID | wMsgId | Set as message ID form DtwDbgReq |

TABLE 5 DTWDBGRSP FIELDS

## 1.2 Detailed Description

A top-level overview of the system with trace and debug capability is shown in Figure 1. Existing data network is used to transport trace information. The Trace trigger block is present in all the AIUs. The capture block is present in all AIUs, DMIs and DIIs. DCE and DVE do not participate in Trace capture. Note that DCE does not connect to the data network. Trace information from other units can be used to post process and build DCE and DVE trace. The trace trigger and capture blocks together fill in the functionality of label, order, and capture. Trace accumulate block is centrally located in DVE; it can be accessed via CSRs. It fills in the functionality of aggregate and access, note that it also participates in ordering transactions. The Analysis functionality falls within the software domain and is not specified in this document for now.

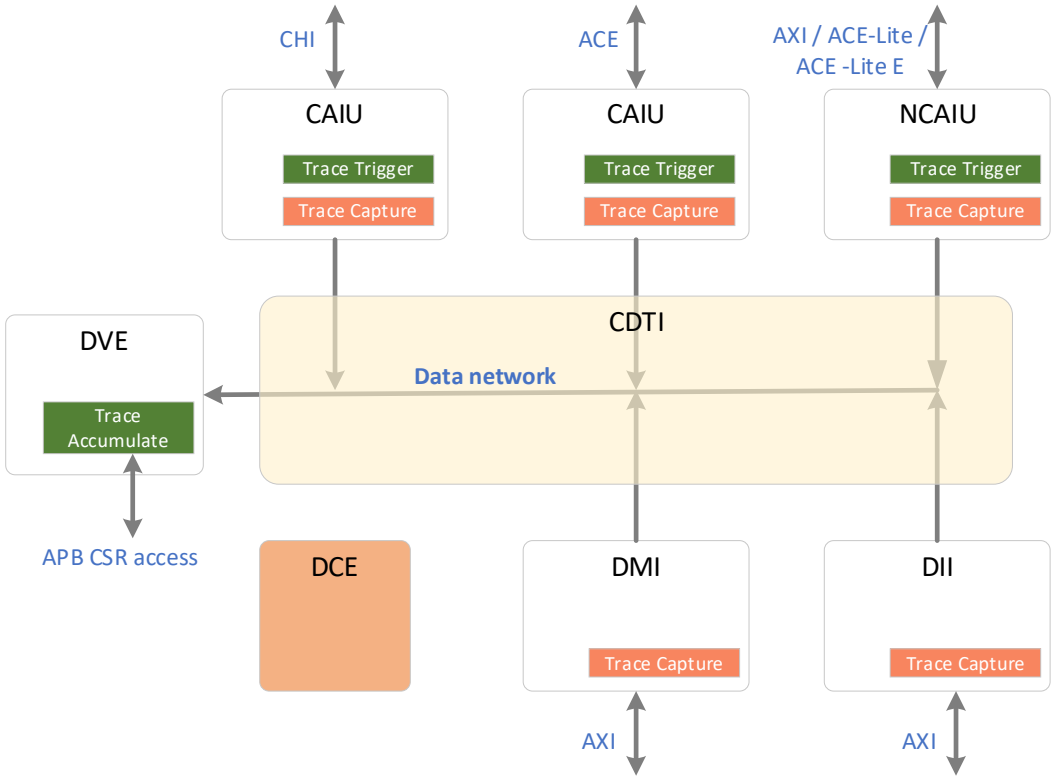



FIGURE 1 TRACE AND DEBUG OVERVIEW

### 1.2.1 Trace Trigger

The trace trigger block is responsible for identifying transactions to be captured and marking them. Ncore protocol has a "Trace Me" field in its messages. Ncore units shall propagate this trace signaling appropriately in all derived messages originating from a transaction that is marked to be traced (note that SysReq and SysRsp are not traced and will have their trace me field tied to zero). Furthermore, the trace signal shall be propagated on interfaces that support trace signaling based on the specific requirements of the interface (today this includes CHI-B and ACE-Lite E/ ACE5-Lite).

There are two classification of marking a transaction to be traced.

1. Master initiated trace, this applies to CHI and ACE-Lite E (or ACE5-Lite) interfaces, which have trace signal capability on the interface.
2. Ncore initiated trace, this applies to all AIUs in Ncore, here Ncore provides CSRs which can be configured to mark desired transactions as trace transactions.

#### *1.2.1.1 Master Initiated Trace*

CHI and ACE-Lite E (ACE5-Lite) interfaces have trace signaling capability, the master can identify transactions that are to be traced by appropriate interface signaling. Ncroe shall honor this signaling and meet the interface requirements. In the case of ACE5-Lite the more conservative approach must be taken where all responses on the native interface must

propagate the trace signal, furthermore in the case of write channel the trace signaling on W cannel is ignored and trace signaling on AW is taken into account.

#### 1.2.1.2 Ncore Initiated Trace

All AIUs in the Ncore system including those that support trace signaling shall have the capability to initiate transaction tracing using internal CSRs. An incoming transaction on the native interface is compared with the trace CSR settings, if there is a match the transaction is marked to be traced. Multiple number of CSR sets can be present as specified at build time by the parameter nTraceRegisters. Different trace options are described below, in a single set of register all the options below can be enabled together, they are checked with an AND option i.e. that is all enabled options should match the incoming transaction. Between register sets it is an OR operation.

- **Trace Signal:** Trace transactions that are marked by the native interface, if trace signaling capability is available on the Native interface.
- **Address Range:** Trace transactions that match the specified address range. One address range can be specified per register set.
- **Op Code:** Trace transactions that match the specified Op codes. Upto 4 op codes can be specified at a time per register set.
- **Memory Attribute:** Trace transactions that match the specified memory attributes on the native interface. This is MemAttr for Chi and AxCache for AXI/ACE/ACE-Lite/ACE-Lite E. One memory attribute can be specified per register set.
- **User Bits:** Trace transactions that match the specified user bits on the native interface. One user bit value can be specified per register set.
- **Target Type:** Trace transactions that match the specified target type. The target type is specified using HUT and HUI. In the case of DII, HUI indicates DII NunitID and in the case of DMI, HUI indicates MIG number. One target type can be specified per register set.

#### 1.2.1.3 Trace Control Register (xTCTRLR)

| Bits | Name | Access | Reset | Description |
|---|---|---|---|---|
| 0 | Trace Signal | R/W | 0x1 | Trace using the native interface trace signaling (only present if the Native interface supports trace else it is reserved) |
| 1 | Address Range | R/W | 0x0 | Trace if there is a match on the address range specified |
| 2 | Op code | R/W | 0x0 | Trace if there is a match on the op codes specified. (Does not apply to AXI NCAIU, must be reserved in the case) |
| 3 | Memory attribute | R/W | 0x0 | Trace if there is a match on the memory attribute specified |

| | | | | |
|---|---|---|---|---|
| 4 | User bits | R/W | 0x0 | Trace if there is a match on the user bits specified (Reserved if user bits do not exist in the configuration) |
| 5 | Target type | R/W | 0x0 | Trace if there is a match on the target type |
| 6 | HUT | R/W | 0x0 | Specifies target type DMI or DII |
| 11:7 | HUI | R/W | 0x0 | Specifies the HUI (NunitID for DII and MIG number of DMI) |
| 18:12 | Reserved | RO | 0x0 | |
| 23:19 | Size | R/W | 0x0 | Address range size, this field indicates a binary number from 0 to 31 from which the region's Size is calculated as (Size of IG) * 2 ^(Size +12) bytes |
| 31:24 | Memory Attribute | R/W | 0x0 | Specifies the memory attribute:<br>**CHI:**<br>bits 31:28 are MemAttr field.<br>bits 27: 24 are reserved.<br>**AXI/ACE/ACE-Lite/ACE-Lite E :**<br>Bits 31:28 are AxCache field.<br>Bits 27:26 are for AR and AW respectively, if both are set match is done on both AR and AW.<br>Bits 25:24 are reserved |

#### 1.2.1.4 Trace Base Address Low Register (xTBALR)

| Bits | Name | Access | Reset | Description |
|---|---|---|---|---|
| 31:0 | Base address LO | R/W | 0x0 | Lower order bits 43:12 of the base address of the region |

#### 1.2.1.5 Trace Base Address High Register (xTBAHR)

| Bits | Name | Access | Reset | Description |
|---|---|---|---|---|
| 7:0 | Base address Hi | R/W | 0x0 | Higher order bits 51:44 of the base address of the region |
| 31:8 | Reserved | | | |

#### 1.2.1.6 Trace Op Code Register (xTOPCR0)

| Bits | Name | Access | Reset | Description |
|---|---|---|---|---|
| 14:0 | Op code 1 | R/W | 0x0 | Op code number 1 (mapping defined based on Native interface, MSB unused bits are RO reserved) |
| 15 | Valid 1 | R /W | 0x0 | Valid bit for op code 1 |
| 30:16 | Op code 2 | R/W | 0x0 | Op code number 2 (mapping defined based on Native interface, MSB unused bits are RO reserved) |
| 31 | Valid 2 | R /W | 0x0 | Valid bit for op code 2 |

#### 1.2.1.7 Trace Op Code Register (xTOPCR1)

| Bits | Name | Access | Reset | Description |
|---|---|---|---|---|
| 14:0 | Op code 3 | R/W | 0x0 | Op code number 3 (mapping defined based on Native interface, MSB unused bits are RO reserved) |
| 15 | Valid 3 | R /W | 0x0 | Valid bit for op code 3 |
| 30:16 | Op code 4 | R/W | 0x0 | Op code number 4 (mapping defined based on Native interface, MSB unused bits are RO reserved) |
| 31 | Valid 4 | R /W | 0x0 | Valid bit for op code 4 |

#### 1.2.1.8 Trace User Bits Register (xTUBR)

| Bits | Name | Access | Reset | Description |
|---|---|---|---|---|
| 31:0 | User bits | R/W | 0x0 | Configured number of user bits rest of the field is reserved |

#### 1.2.1.9 Trace User Bits Mask Register (xTUBMR)

| Bits | Name | Access | Reset | Description |
|---|---|---|---|---|
| 31:0 | User bits mask | R/W | 0x0 | User bits mask register, set 1 for bits that need to be used for comparison. |

### 1.2.2 Trace Capture

All AIUs, DMIs and DIIs shall support trace capturing capability. The block snoops SMI interface and captures messages that have the TraceMe field set. It captures non-data request, response and data messages (this includes all NDP header fields that are of non-zero width and NDP payload); actual data within the data message is not captured. Which SMI ports are to be snooped and captured is configurable via CSR settings. Multiple SMI ports can be snooped and captured simultaneously. The capture block has a capture buffer that is sized based on the parameter nUnitTraceBufSize, this parameter specifies the number of 64-byte entries in the buffer. The captured data packed into DTWs, once enough data is captured to build a single DTW, it is sent out on the SMI data network to the DVE in the system. If enough data is not accumulated to build a single DTW then a DTW with at-least a single concerto message and padding must be sent out once a timeout is reached (the timeout value is implementation defined example 12bit counter). If the capture buffer is full then no more messages are captured until the buffer drains to accommodate at least one more DTW.

The DTW data format example is shown in Figure 2. The example is for a 128-bit bus width. The packet always starts with a 32-bit free running counter time stamp, followed by the captured concerto message. As shown multiple full concerto

messages can be packed in a single a DTW. The DTW may end with padding of 0s if a full message cannot be packed into the DTW.

The capture block will give out two events. These events indicate SMI messages that were dropped and captured in a clock cycle respectively. As there can be upto 8 SMI messages that are dropped or captured, these event maybe 3 bits wide each.

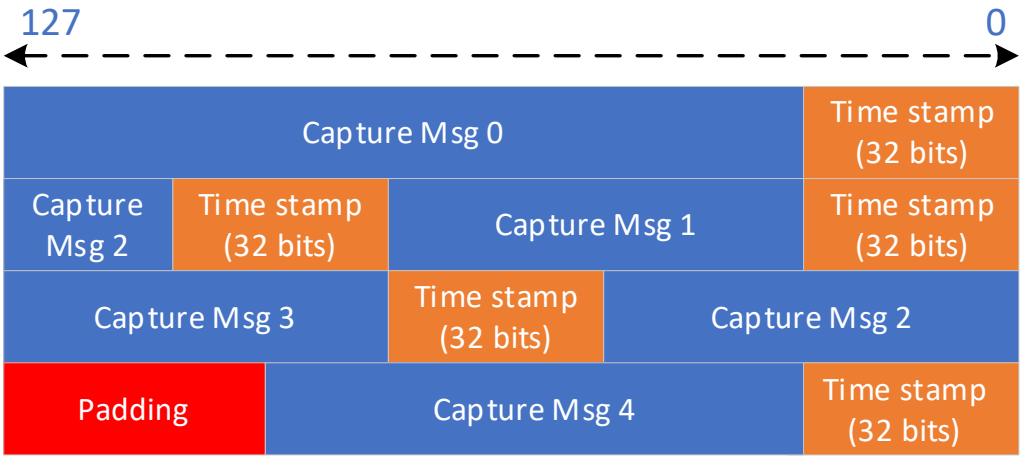



FIGURE 2 TRACE DTW DATA BUILD FORMAT

The DtwRsp cm_status field is used to synchronize the Ncore unit local time stamp counter with the DVE time stamp counter. This filed is as described in Table 6.

| Field | Bits | Name | Description |
|---|---|---|---|
| cm_status | 7 | Signe | Positive (0) or negative (1) |
| | 6:0 | Value | Value by which the counter needs to be adjusted, this maps to counter bits [10:4] of the local counter, bits [3:0] are zero. |

TABLE 6 DTWRSP CM_STATUS DESCRIPTION

The general concept is to have a feedback loop with a small correction and programable increment, as shown in Figure 3 master timeout stamp counter (MTSP) is compared with the unit timeout stamp counter (UTSC). The resultant error is multiplied with a gain and the UTSC increment value is updated accordingly.

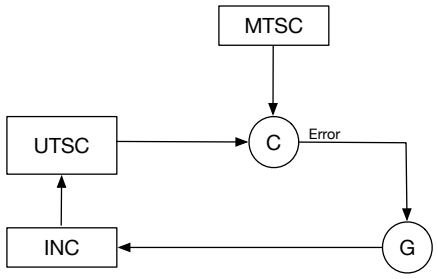



FIGURE 3 COUNTER SYNC FEEDBACK LOOP

C in the comparator of MTSC (counter in DVE) and UTSC (the local counter); comparison happens in DVE and error is sent back in DtwDbgRsp. G is the gain factor which is multiplied with the error and is used to correct INC (increment value of UTSC)

In this implementation INC is a register of 12 bits (4 bits of integer and 8 bits of fraction). This value initially represents the clock frequency difference between DVE and the Ncore unit, it gets updated based on the feedback error. Example values are shown in Table 1. The error is represented by 8 bits in singed magnitude format. where bit 8 is the sign and lower 7 bits is magnitude within DtwDbgRsp. The 7 magnitude bits are multiplied with the 4-bit gain from CSRs to give an 11-bit number. MSB 8-bits of this 11-bit number are treated as fractional value that is either added or subtracted to the INC value based on the sign of the correction.

| Clock ratio with DVE | Value in INC register |
|---|---|
| 2 | {4'b0010, 8'b0000_0000} |
| 1 | {4'b0001, 8'b0000_0000} |
| 0.5 | {4'b0000, 8'b1000_0000} |
| 0.25 | {4'b0000, 8'b0100_0000} |
| 0.33 | {4'b0000, 8'b1100_0000} |

TABLE 7 INCREMENT VALUE EXAMPLES

#### 1.2.2.1 Capture Control Register (xCCTRLR)

| Bits | Name | Access | Scope | Reset | Description |
|---|---|---|---|---|---|
| 0 | Ndn0 SMI TX | R/W | All | 0x0 | Ndn0 SMI Tx snoop and capture enable |
| 1 | Ndn0 SMI RX | R/W | All | 0x0 | Ndn0 SMI Rx snoop and capture enable |
| 2 | Ndn1 SMI TX | R/W | All | 0x0 | Ndn1 SMI Tx snoop and capture enable |
| 3 | Ndn1 SMI RX | R/W | All | 0x0 | Ndn1 SMI Rx snoop and capture enable |
| 4 | Ndn2 SMI TX | R/W | All | 0x0 | Ndn2 SMI Tx snoop and capture enable |
| 5 | Ndn2 SMI RX | R/W | All | 0x0 | Ndn2 SMI Rx snoop and capture enable |
| 6 | Dn0 SMI TX | R/W | All | 0x0 | Dn0 SMI Tx snoop and capture enable |
| 7 | Dn0 SMI RX | R/W | All | 0x0 | Dn0 SMI Rx snoop and capture enable |
| 15:8 | Reserved | RO | All | 0x0 | |
| 19:16 | gain value | R/W | All | 0x2 | 4-bit gain value |

| | | | | | |
|---|---|---|---|---|---|
| 31:20 | Inc value | R/W | All | 0x100 | Time stamp counter increment value: top 4 bits are integer and lower 8 bits are fractional |

## 1.2.3 Trace Accumulate

This block is present only in DVE and the main functionality is to accumulate incoming trace DTWs from different Ncore capture units. The capture buffer is sized based on the parameter nMainTraceBufSize. Each entry in the buffer is organized as shown in Figure 4. It is a 72-byte entry with 64-byte payload (DtwReq data received from the capture block).

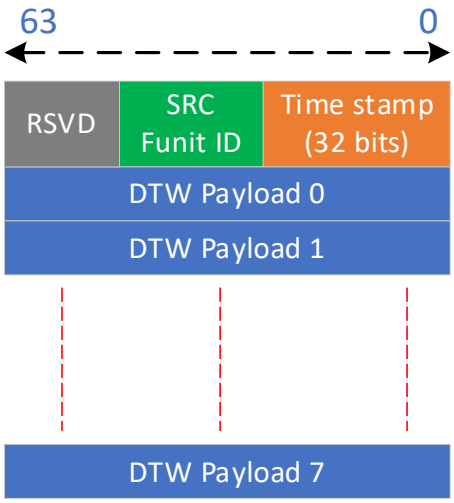



FIGURE 4 TRACE FORMAT

The time stamp here is the local free running counter at DVE. This counter is used to synchronize other Ncore unit counters. This is done by taking the first-time stamp from the received DtwReq and comparing it with bits [31:4] of the local counter, if the received time stamp value is bigger, then the cm status bit 7 of the DtwRsp is set to 1 else set to 0. Bits [10:4] of the local counter are placed into the cm status bits [6:0]. If the counter values match, then the cm status bits [6:0] are set to zeros.

The trace buffer shall be accessible via CSRs.

The accumulate block will give out an event that indicates DTW messages that are dropped per clock cycle.

### *1.2.3.1 Trace Accumulate Status & Control Register (xTASCR)*

| Bits | Name | Access | Reset | Description |
|---|---|---|---|---|
| 0 | Buffer empty | RO | 0x1 | Set when all the entries within the buffer are read and the buffer is empty |
| 1 | Buffer full | RO | 0x0 | Set when the buffer is full. |

| | | | | |
|---|---|---|---|---|
| 2 | Circular buffer | R/W | 0x0 | When set, local buffer acts as a circular buffer, older messages are dropped once the buffer is full. When not set new messages are dropped when the buffer is full |
| 3 | Buffer clear | WSC | 0X0 | Write 1 to clear the complete buffer |
| 4 | Buffer read | WSC | 0x0 | Write one to trigger a read |
| 31:5 | Reserved | RO | 0x0 | |

#### 1.2.3.2 TTracrace Accumulate data header Register (xTADHR)

| Bits | Name | Access | Reset | Description |
|---|---|---|---|---|
| 7:0 | Funit ID | RO | 0x0 | F unit ID of the trace capture Ncroe unit |
| 30:8 | Reserved | RO | 0x0 | |
| 31 | Valid | RSC | 0x0 | This is set once read data is valid, self clears on read |

#### 1.2.3.3 Trace Accumulate data Time stamp Register (xTADTSR)

| Bits | Name | Access | Reset | Description |
|---|---|---|---|---|
| 31:0 | Time stamp | RO | 0x0 | Time stamp value from DVE free running counter |

#### 1.2.3.4 Trace Accumulate data (0-15) Register (xTAD0R)

These are 16 registers, where register 0 maps to data [31:0] and so on so forth.

| Bits | Name | Access | Reset | Description |
|---|---|---|---|---|
| 31:0 | Capture data | RO | 0x0 | Capture data |

# 2 Opens

Questions/Feedback/Need to discuss:

# 3 Glossary

Arteris

A NoC Company

NCore3

A coherent NoC provided by Arteris with AMBA interfaces and built-in caches.

# 4 Notes

Notes ……