|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Change Request** | | | | | | | |
| **Document** | **ORAN-WG6.AAL-GAnP** | **ver** | **00.01.01** | **CR** | **NVD-001** | **rev** | **3** |

|  |  |  |  |
| --- | --- | --- | --- |
| ***Title:*** | AAL Greenfield Agreements-Cat 1 | | |
| ***Source to WG:*** | NVIDIA | | |
| ***Target WG :*** | **WG6** | | |
| ***Category:*** | **B** | ***CR Creation Date*** | October 18, 2021 |
|  | *Use one of the following* ***categories****:* ***A*** *(mirror corresponding to a change in an earlier release)* ***B*** *(addition of feature),* ***C*** *(functional modification of feature)* ***D*** *(editorial modification)* ***F*** *(correction)*  Detailed explanations of the above categories can be found in 3GPP [TR 21.900](http://www.3gpp.org/ftp/Specs/html-info/21900.htm). | | |

|  |  |
| --- | --- |
| ***Reason for Change:*** | To include AAL Greenfield Agreements in Sec. 2.5 (General Interface Principles) |
| ***Summary of change:*** | New text is proposed and can be reviewed by track change in the text below |
| ***Consequences if not aproved:*** | If not included, AAL Greenfield agreements will not be adopted in AAL GA&P Specification |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***Clauses affected:*** | <list specific document sections impacted by the CR> | | | | |
|  | **Y** | **N** |  | |  |
| ***Other specs*** |  | **X** | Other core specifications: | <fill in related CRs if “Y”> | |
| ***affected:*** |  | **x** | Test specifications: | <fill in related CRs if “Y”> | |
| ***(show related CRs)*** |  | **X** | O&M Specifications: | <fill in related CRs if “Y”> | |
| ***Supporting material:***  ***Other comments:*** | <provide file name or URL of any material supporting this CR> | | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| ***Status:*** |  | ***CR Closed Date:*** |  |
| ***Outcome:*** |  | ***Duplication:*** |  |
| ***Outcome explanation:*** |  | | |

The proposed changes are indicated by Track Changes in the text below.

## General Interface Principles

The set of generic and profile-specific features of AALI described in the following subsections are defined from an application (i.e. AALI consumer) point of view.

### Generic Principles

#### Extensibility

O-RAN has defined the functions that can be accelerated by the cloud platform based on 3GPP specifications and O-RAN deployment scenarios. However, the AALI should not limit innovation of future implementations and should evolve as the specification requires. To that end, the AALI shall be extensible to accommodate future revisions of the specification.

#### Implementation Independence

AALI shall be independent of the underlying AALI implementation.

#### Interrupt and Poll Mode

AALI shall allow multiple design choices for application vendors and shall not preclude an application/ HW accelerator vendor from adopting/supporting an interrupt-driven design or poll-mode design or any combination of both. As such, the AALI shall support both interrupt mode, poll mode and any combination of interrupt and poll modes for the data-path application interface.

#### Discovery and Configuration

AALI shall support application to discover and configure AAL-LPU(s). AALI shall allow an application to discover what physical resources have been assigned to it from the upper layers, and then to configure said resources for offload operations.

#### Multiple AAL-LPU Support

There may be scenarios where multiple AAL-LPUs (either implementing the same or different AAL profile(s)) are assigned to a single application, which uses one or more of these AAL-LPU(s) as needed. The AALI shall support an application using one or more AAL-LPU(s) at the same time, as shown in Figure 2.6

Graphical user interface, text, application, chat or text message

Description automatically generated

Figure 2.6. Logical Representation of AALI support for multiple AAL-LPUs

#### AALI offload capabilities

The AALI in supporting different AAL profiles and AALI implementations shall support different offload architectures including look-aside, inline, and any combination of both. An AALI implementation shall support one or more of these offload architectures depending on the supported AAL profile(s).

#### Look-aside Acceleration Model

The AALI shall support look-aside acceleration model where the host CPU invokes a HW accelerator for data processing and receives the result after processing is complete. A look-aside architecture, illustrated in Figure 2.7, allows the application to offload AAL profile(s) to a HW accelerator and continue to perform other work in parallel—this could be to continue to execute other software tasks in parallel or to sleep and wait for the HW accelerator to complete. This model requires the AALI to support two operations, one for initiating the offload and another for retrieving the output data once complete.

Diagram, schematic

Description automatically generated

Figure 2.7. AALI look-aside acceleration model

#### Inline Acceleration Model

The AALI shall support inline acceleration model where the host CPU, after invoking a HW accelerator for offloading AAL profile(s), does not necessarily retrieve the post processed data. Unlike the look-aside acceleration model where data source/sink is always the host CPU (i.e., the HW accelerator always receives the data to be processed from the host CPU and returns the post processed data to the same), a HW accelerator operating in inline acceleration mode may receive/return data from/to a different source/destination node than the host CPU, depending on the direction of data flow (e.g., in downlink (DL) direction versus uplink (UL) direction). Figure 2.8. AALI inline acceleration model shows one possible implementation of an inline acceleration model.

Diagram

Description automatically generated

Figure 2.8. AALI inline acceleration model

In Figure 2.8. AALI inline acceleration model, “Tx” refers to the transmission of the data from the HW accelerator through an egress port (e.g., an Ethernet interface) to a destination node (e.g. O-RU), while “Rx” refers to the reception of data through an ingress port (e.g., an Ethernet interface) to the HW accelerator from a source node (e.g. O-RU).

While the look-aside architecture (in DL) shall support dataflow from the CPU to the HW accelerator and back to the CPU before being sent to the egress port (e.g., front-haul interface), the inline architecture (in DL) shall support data flow from the CPU to the accelerator and directly from the accelerator to the egress port (e.g., front-haul interface), instead of being sent back to the CPU. The typical user plane data flows for accelerating the O-DU high-PHY functions for the look-aside and inline architectures are as follows.

Look-aside architecture user plane dataflow

CPU ↔ HW accelerator ↔ CPU ↔ front-haul: for a set of consecutive high-PHY functions offload (e.g., FEC)

CPU ↔ HW accelerator ↔ CPU ↔ HW accelerator ↔…↔ CPU ↔ front-haul: for a set of non-consecutive high-PHY functions offload

Inline architecture user plane dataflow

CPU ↔ HW accelerator ↔ front-haul: for a set of consecutive high-PHY functions offload (up to the end of the high-PHY pipeline)

Figure 2.9 illustrates one possible implementation of the look-aside and inline architectures. While a set of PHY-layer functions are offloaded to the HW accelerator for look-aside acceleration, the entire end-to-end high-PHY pipeline is offloaded to the HW accelerator for inline acceleration.



Figure 2.9. User plane dataflow paths in look-aside and inline acceleration architectures.

#### AALI API Concurrency and Parallelism

To enable greater flexibility and design choice by application vendors, the AALI shall support multi-threading environment allowing an application to offload acceleration requests in parallel from several threads.

#### Separation of Control and User Plane AALI APIs

For efficiency and flexibility of AALI implementation, AALI shall support separation of control and user plane APIs with appropriate identifiers, as required by different AAL profiles.

#### Support of Versatile Acceleration Payload

Range of payload sizes can vary widely, depending on the specific layer of the RAN protocol stack from which the workload for AAL profile(s) is offloaded to a HW accelerator. AALI API shall be flexible to support various ranges of payload sizes as required by different AAL profiles.

#### Support of Different Transport Mechanisms

The transport between an application and an AALI implementation can be of different types (e.g., based on shared memory, PCIe interconnect, over ethernet etc.). AALI shall support abstraction of these various transport mechanisms between the ‘Application’ and the ‘AALI implementation’.

#### AALI API namespace

For convenience of AALI implementation, AALI shall follow a unique name space for all AALI API functions.

### High-PHY Profile Specific Principles

The set of features of AALI described in the following subsections are relevant for inline high-PHY AAL profiles (profile names with suffix ‘\_HIGH-PHY’) defined in Chapter 5

#### Separation of Cell and Slot Level Parameter Configurations

In general, “cell-specific” (typically static or semi-static in nature) parameters change less frequently than “slot-specific” (typically dynamic in nature, specific to PHY channels/signals) parameters associated with inline, high-PHY AAL profiles. Hence, for optimizing signalling overhead, AALI shall support configuration of “cell-specific” and “slot-specific” parameters to AALI implementation using separate AALI API functions. It is noteworthy that the cell/slot specific configurations can include both control and user planes.

#### Timing Synchronization

To maintain consistent timing (with respect to a given over-the-air (OTA) slot) AALI shall support system frame number (SFN) based or slot-based synchronization between the application and the AALI implementation supporting inline, high-PHY AAL profiles.

#### Compatibility with O-RAN FH interface

AALI API shall be compatible with O-RAN FH interface (7.2-x split) to enable communication between the O-DU application and O-RU via AALI implementation as required by inline, high-PHY AAL profile(s).