|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Change Request** | | | | | | | |
| **Document** | **ORAN-WG6.AAL-GAnP** | **ver** | **00.01.01** | **CR** | **NVD-003** | **rev** | 1 |

|  |  |  |  |
| --- | --- | --- | --- |
| ***Title:*** | AAL GAnP Chapter 2 Revision – to align with Cat 2 Greenfield Agreements, NVD CR-001 and NVD CR-002 | | |
| ***Source to WG:*** | NVIDIA | | |
| ***Target WG :*** | **WG6** | | |
| ***Category:*** | **B** | ***CR Creation Date*** | October 31, 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 align AAL GAnP Chapter 2 with Cat 2 Greenfield Agreements and the CRs NVD-001 and NVD-002 |
| ***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 GAnP Chapter 2 will not be consistent with AAL Greenfield Agreements and related CRs NVD-001 and NVD-002 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***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 Aspects

## Hardware Acceleration

In the design of digital computing systems, ranging from general-purpose processors to fully customized hardware, there is a tradeoff between flexibility and efficiency, with efficiency increasing by orders of magnitude when any given application is implemented in hardware. The range of implementation options includes general-purpose processors such as CPUs, more specialized processors such as GPUs, functions implemented on field-programmable gate arrays (FPGAs), and fixed-functions implemented on application-specific integrated circuits (ASICs). Hardware accelerator is a specialized HW implementation that can offload processing from application(s) running on the General-Purpose Processor. Any transformation of data or routine that can be computed, can be processed purely in software running on a generic CPU, purely in a specialized hardware, or using a combination of both. The implementation of computing tasks in hardware to improve performance is known as hardware acceleration. The hardware acceleration can be implemented in the form of lookaside or inline mode where in the former case, the host CPU invokes an accelerator for data processing and receives the result after processing is complete, while in the latter case, the accelerator, after being invoked by the host CPU with the request for data processing, completes the processing request of data received from a source node and directly transfers the post-processed data to a destination node, where the source or destination node can be different than host CPU (e.g., an Ethernet Interface). The principle of hardware acceleration and functional offloading in lookaside mode is illustrated in Figure 2.1, allowing the application to offload workload to a hardware accelerator and to continue performing other work in parallel- this could be to continue to execute other software tasks in parallel or to sleep and wait for the accelerator hardware to complete. The hardware acceleration boosts application performance in environments with compute-intensive, deeply pipelined, massively parallel operations as shown in Figure 2.1. This model requires the API to support two operations, one for initiating the offload and another for retrieving the operation once complete.

Graphical user interface, application, Teams

Description automatically generated

Figure 2.1 Example illustration of the effect of hardware acceleration on functional compute performance

## AAL Architecture

The goal of the acceleration abstraction layer (AAL) is to specify a common and consistent interface for HW accelerators to the applications which facilitates decoupling of an application, e.g., O-RAN Cloudified Network Function, from a specific HW implementation. In order to accommodate the many different combinations of HW and SW implementations and also many different network deployment scenarios, the AAL introduces the concept of an AAL profile which is used to distinguish between the different combinations of accelerated functions (AFs) to be offloaded. The end-to-end high-level AAL architecture block diagram is shown in Figure 2.2.

Graphical user interface

Description automatically generated

Figure 2.2 High Level AAL Architecture Diagram

Figure 2.3 shows the relationship and cardinality between the components that constitute the AAL architecture.

@startuml oran\_aal

skinparam component {

FontSize 10

BorderColor black

BackgroundColor lightgrey

ArrowFontName Ariel

ArrowColor black

ArrowFontColor #777777

}

component "AAL Logical Processing Unit" as aal\_lpu

component “AAL Queue” as q

component "AAL Profile" as profile

component "Application" as app

component "Hardware Accelerator" as hw

component "Accelerated Function" as af

component “HW Accelerator Manager” as hwM

aal\_lpu o-left- "N" q

aal\_lpu "M" --o hw

aal\_lpu "A" -up-o app

aal\_lpu o-- "P" profile

hw o-up- "B" profile

hw o-right- hwM

q o--o profile

profile o–- “C” af

@enduml

Figure 2.3 AAL Component Relationship and Cardinality

### AAL Deployed in Cloud environments

Graphical user interface, application

Description automatically generated

Figure 2.4 Accelerator APIs/Libraries in Container and Virtual Machine Implementations

The AAL specification shall define the AAL interface (AALI) and the AAL profiles that may be supported by that interface. The AALI is the application interface the O-RAN Cloudified Network Functions shall use to access the underlying AALI implementation encompassing HW accelerator and associated SW libraries, drivers etc. In Figure 2.4, two deployment scenarios are shown- one with Containers, and the other with Virtual Machines. In both instances the AALI is the interface between the Network Function (NF) application and the AALI implementation that are exposed to the Network Function application.

Figure 2.4 also shows the O-Cloud Infrastructure Management Services and Accelerator management. The AAL Specification shall define the requirements for managing the hardware accelerator in the O-Cloud instance. The orchestration of the HW Accelerator Manager is outside the scope of the AAL and shall be specified in the ORAN WG6 O2 specification [12].

## AAL Specification Objectives

The AAL specification facilitates the following:

Deployment of O-RAN Cloudified Network Functions with AALI implementations from different vendors.

## Scope of the AALI

The AAL specification shall define the AALI – the interface between the application and AALI implementation in the O-Cloud instance. This includes the APIs and information models, operations and input/outputs used by the application to interface with the AALI implementation. The AALI implementation itself shall not be defined by the AAL GAnP specification. ETSI GS NFV-IFA 002 [5] defines several abstraction models including pass through and abstracted models that can be used to realize an AALI implementation.

Diagram

Description automatically generated

Figure 2.5 AAL Specification Scope