Using MARTE with SysML: Multimedia Example

Table of Content

Table of Figures

Abstract: This paper demonstrates how to use MARTE profile with SysML plugin, through a Multimedia example: an audio-video decoder running on a generic platform.

Keywords: SysML, MARTE

1. Introduction

This paper shows how to use MARTE profile with SysML plugin to model a high-level description of an audio-video decoder running on a generic platform (processor + hardware accelerator), in **MultiMediaExample.mdzip** SysML project file (located in *<install.dir>/samples/SysML/MARTE sample* directory).

2. Model Packaging

The model packaging reflects the separation between application (or functional) model and platform model. The application model is then mapped onto the platform model to obtain a refined virtual system model.

- Functional architecture describes applicative structures with timed behaviors synchronizing through events and exchanging data.
- Physical architecture describes a generic execution platform (processing units like CPU or IC, physical links like bus or routing network, storage units like memories, firmware like schedulers).
- Virtual system architecture is the result of the mapping step (mapping or allocation itself can be seen as a model) which consists in allocating functions to processing units and data to storage units, and routing inter-processor data on physical links.

Figure 1 shows the package hierarchy of the model.

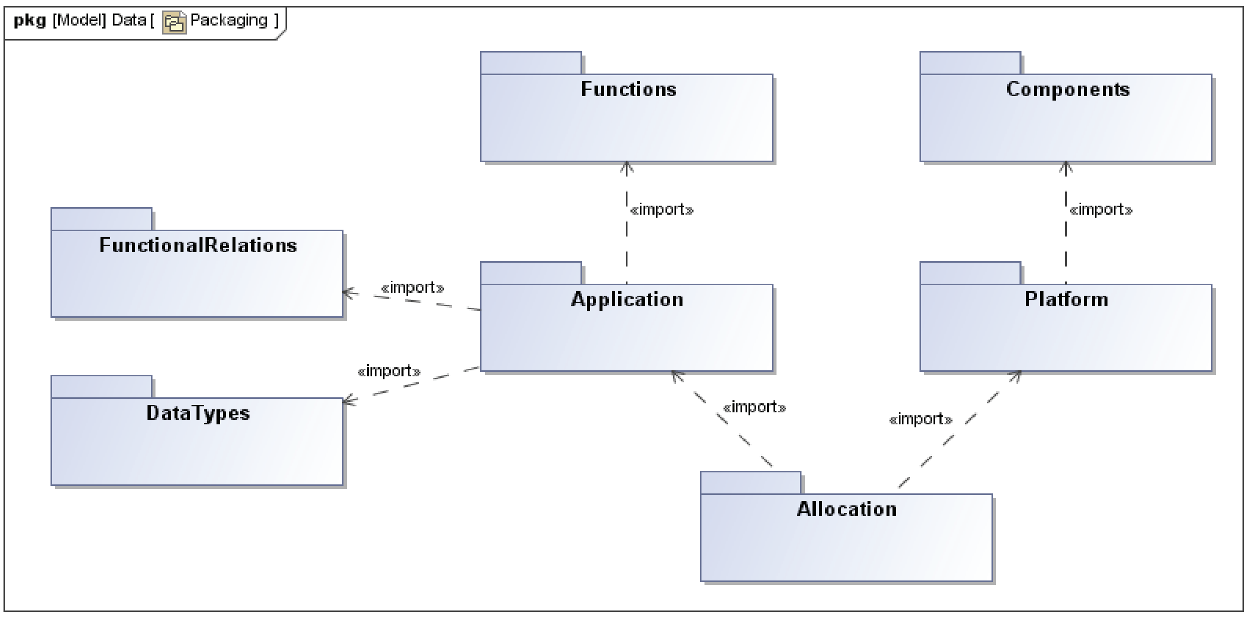

Figure 1: Package Diagram of the Model

- The `Functions` package contains the functions of the application model.
- The `Components` package contains the components of the platform model.
- The `Application` and `Platform` packages contain the models themselves.
- The `Allocation` package contains the allocation model, using elements from application and platform models.

- The DataTypes and FunctionalRelations packages are defined containing the data types and the functional relations (data communication and synchronization elements between functions).

3. Application Model

The application model is a set of functions exchanging data and/or synchronizing through events and relations. There are two different aspects in the application model:

- The structural aspect is the encapsulation of the different functions and the relations between them (data paths).
- The behavioral aspect is the control flow used to process data and synchronize functions.

3.1 Structure

A SysML Internal Block Diagram (IBD) is used to represent the applicative structure of the system. A top level SysML block is used as a root container.

Functions encapsulating other functions are called "container functions" whereas functions defined at the lowest hierarchical levels are called "leaf functions".

Functions of the lower hierarchical level are all Properties of the top-level Blocks. Types of these Properties are defined by Blocks, appearing in the Functions package. Sub-functions of these Blocks are also represented as Properties, and so on.

Figure 2 shows the SysML Block Definition Diagram (BDD), demonstrating the functional hierarchy.

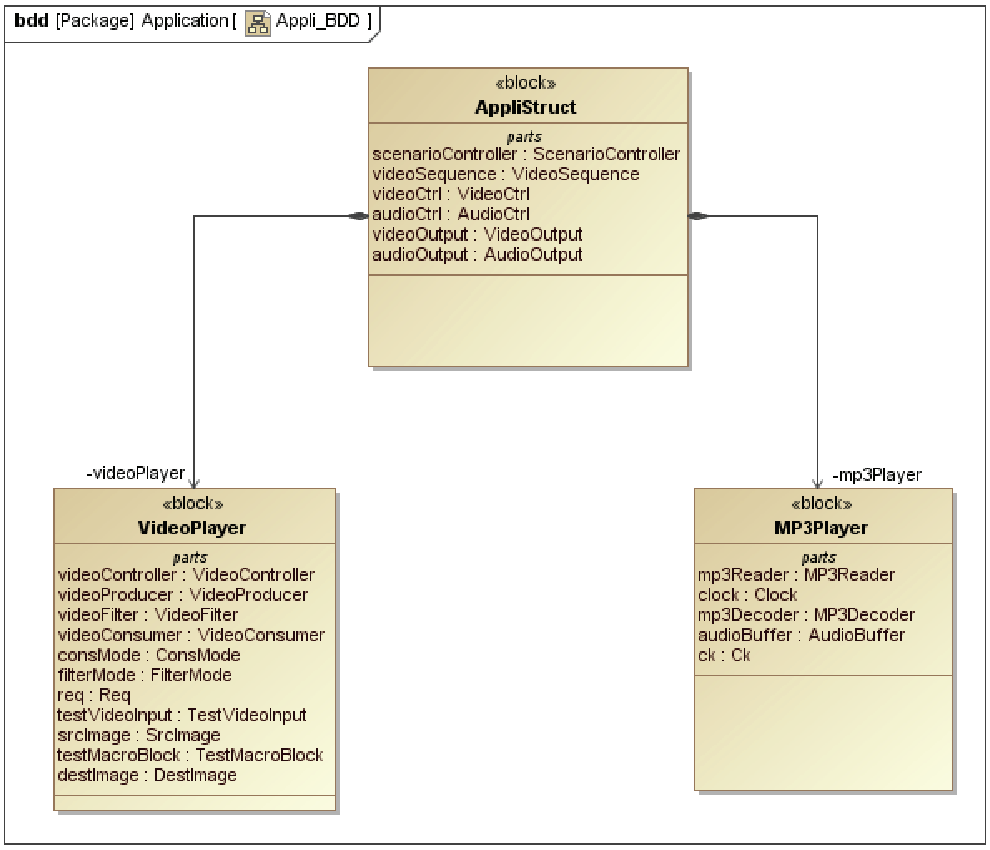
bdd [Package] Application [Appli_BDD]
«block»
AppliStruct
parts
scenarioController : ScenarioController
videoSequence : VideoSequence
videoCtrl : VideoCtrl
audioCtrl : AudioCtrl
videoOutput : VideoOutput
audioOutput : AudioOutput
-videoPlayer
-mp3Player
«block»
VideoPlayer
parts
videoController : VideoController
videoProducer : VideoProducer
videoFilter : VideoFilter
videoConsumer : VideoConsumer
consMode : ConsMode
filterMode : FilterMode
req : Req
testVideoInput : TestVideoInput
srcImage : SrcImage
testMacroBlock : TestMacroBlock
destImage : DestImage
«block»
MP3Player
parts
mp3Reader : MP3Reader
clock : Clock
mp3Decoder : MP3Decoder
audioBuffer : AudioBuffer
ck : Ck

Figure 2: BDD of the Application, Leaf Functions and Relations

Figure 3 shows the leaf functions (lowest hierarchical level functions) defined in Functions package of the SysML model.

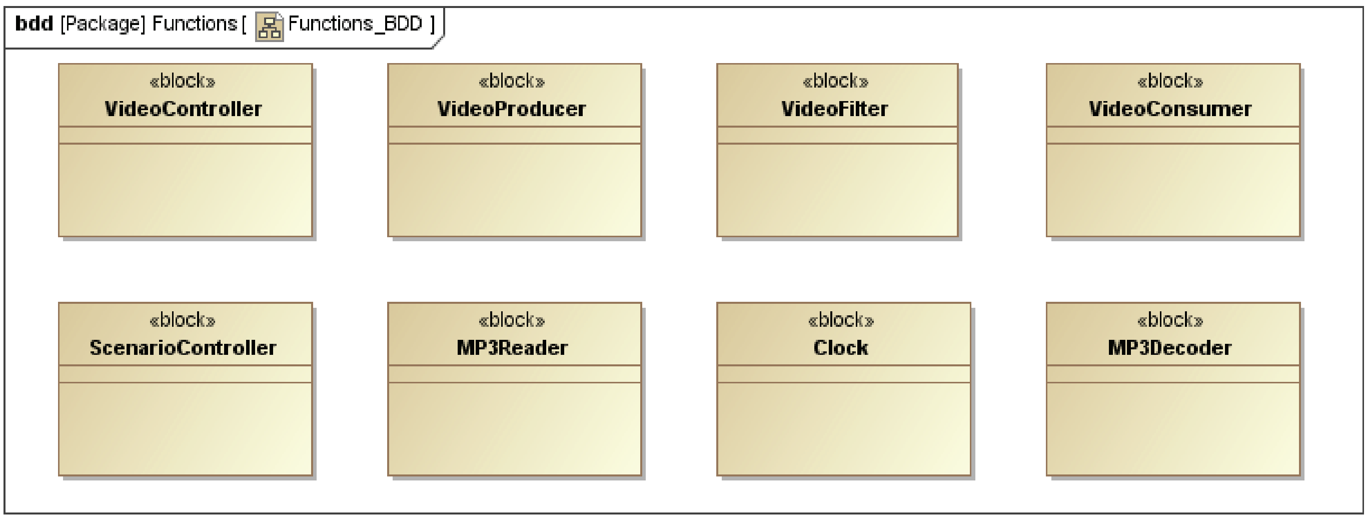

Figure 3: Leaf Functions Defined in Functions Package

The connections between application elements are represented in the IBD of the top level Block (named `AppliStruct`):

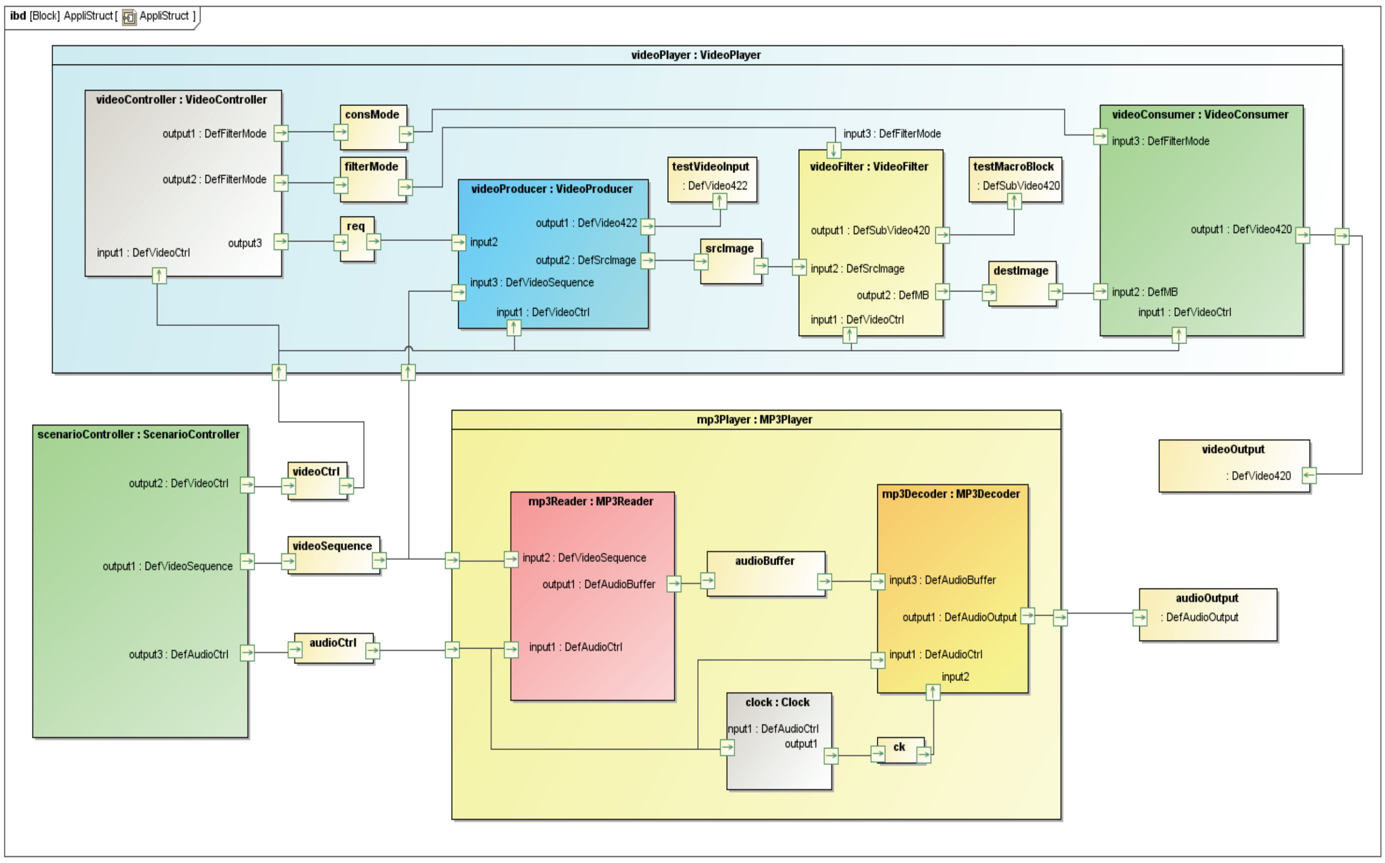

Figure 4: IBD of the Application

Relational elements are represented by Properties with two Ports (input and output). Types of these Properties are SysML Blocks (defined in the `FunctionalRelations` package) with the following stereotypes from MARTE profile:

- `NotificationResource`: event enabling synchronization between functions.
- `SwMutualExclusionResource` with `SharedDataComResource`: shared variable enabling asynchronous data read/write.

- `MessageComResource`: enables synchronous buffered data exchanges.

Between Ports of different Properties, there are *Assembly* and *Delegation* Connectors for Ports forwarding between external and internal functions.

Data Types are defined by SysML Blocks in the `DataTypes` package. The Data Types are carried by the Ports. Such Ports are in fact MARTE FlowPorts, with directions (in, out, inout).

As shown on the IBD, this application is composed of the three main parts:

1. `scenarioController` (bottom left corner) provides the `videoSequence` and controls execution of video and `mp3Player`.
2. `videoPlayer` (top) decodes `videoSequence`.
3. `mp3Player` (bottom) decodes the MP3-encoded sound, contained in `videoSequence`.

3.2 Behavior

Leaf function types are Blocks that contain a Behavior. This Behavior is described by an Activity diagram.

Inputs and outputs of the Activity are Activity Parameter Nodes, allocated to corresponding Ports of the application structure via the MARTE *Allocate* stereotype.

Input and Output Pins allow read/write on Ports in Behaviors. Opaque Actions model those Operations to be performed, i.e. data read/write and processing.

For Opaque Actions, the MARTE stereotype; TimedProcessing, is sometime used in order to set execution duration.

4. Platform model

The platform is a set of computation and storage resources connected by physical links. It is represented using `Hardware Resource Model` from MARTE profile.

The platform structure is described by an IBD. A top-level SysML Block is used as a root container.

Components of the lower hierarchical level are all Properties of the top-level Blocks. Types of these Properties are defined by MARTE Stereotyped Blocks, appearing in the `Components` package. Sub-functions of these Blocks are also represented as Properties, and so on.

Schedulers can be allocated to MARTE `HwProcessor` using the MARTE `Assign` stereotype. The schedulers are defined as Blocks stereotyped with MARTE `GaExecHost`.

Figure 5 shows Blocks defined in the imported `Components` package. The MARTE stereotype used for each one is visible. The package contains computation resources (`UserComputer, CoProcessor`), communication resources (`Bus, USB, SPI`), memories (`DisplayEnv, Source, Mem`) and a scheduler (`Scheduler`).

Figure 6 shows the IBD of the platform. Ports are MARTE `hwEndPoints`. Assembly Connectors are used for connecting `end points` to `Buses`, while Delegation Connectors are used for ports forwarding between different hierarchical levels. `Schedulers` are allocated to hardware processors using `Assign` stereotype from MARTE.

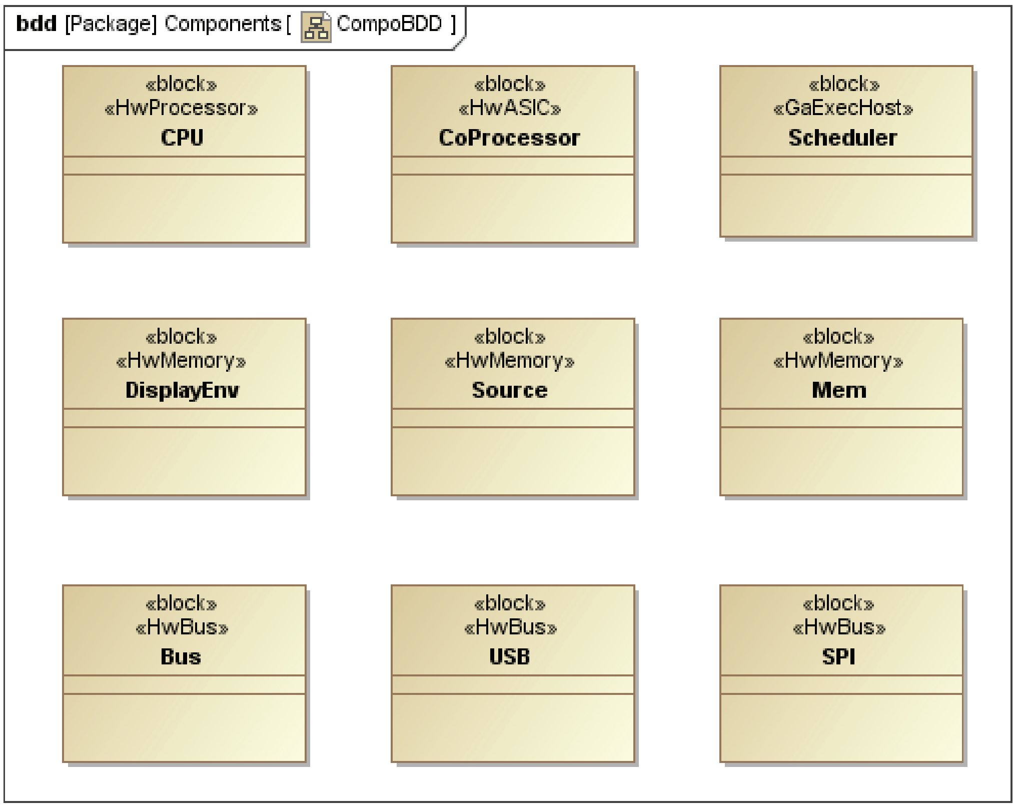
bdd [Package] Components [CompoBDD]
«block» «HwProcessor» CPU
«block» «HwASIC» CoProcessor
«block» «GaExecHost» Scheduler
«block» «HwMemory» DisplayEnv
«block» «HwMemory» Source
«block» «HwMemory» Mem
«block» «HwBus» Bus
«block» «HwBus» USB
«block» «HwBus» SPI

Figure 5: Blocks Defined in Components Package

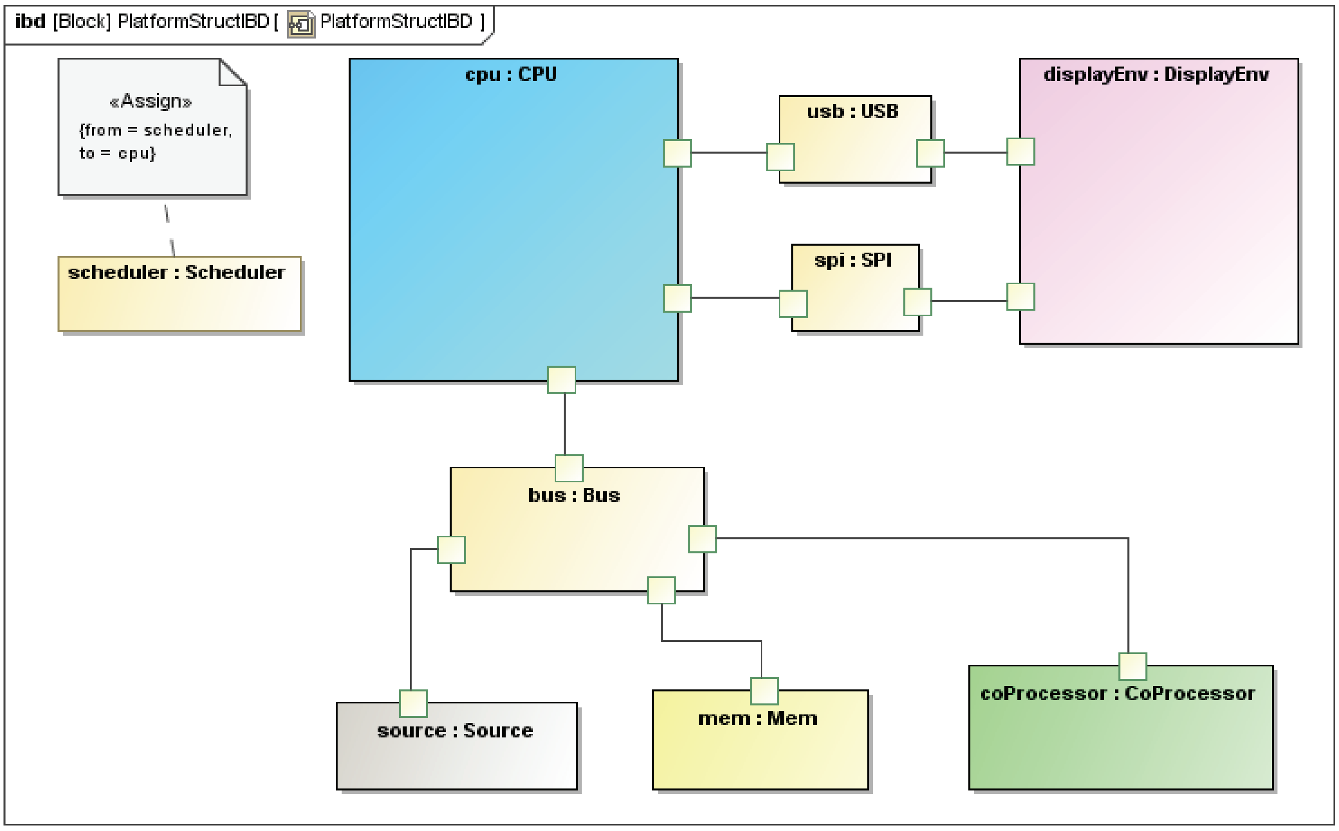
ibd [Block] PlatformStructIBD [PlatformStructIBD]
«Assign» {from = scheduler, to = cpu}
scheduler : Scheduler
cpu : CPU
usb : USB
spi : SPI
displayEnv : DisplayEnv
bus : Bus
source : Source
mem : Mem
coProcessor : CoProcessor

Figure 6: IBD of the Platform

5. Allocation

To obtain an allocated virtual system model, the functions of the application are mapped to the computation resources of the platform, while the shared variables and exchanged data can be mapped onto the memories and the physical links. `Assign` stereotype from MARTE profile is used to model this mapping or allocation.

Software mapping is achieved by assigning functions from the application to tasks (Blocks stereotyped with MARTE `schedulable resource`). Tasks are scheduled by the schedulers of the platform.

Hardware mapping is done by directly assigning functions to ICs of the platform.

Link mapping is the allocation of functional relations to the physical links or the memories of the platform.

Figure 7 shows the different assignments for the example model.

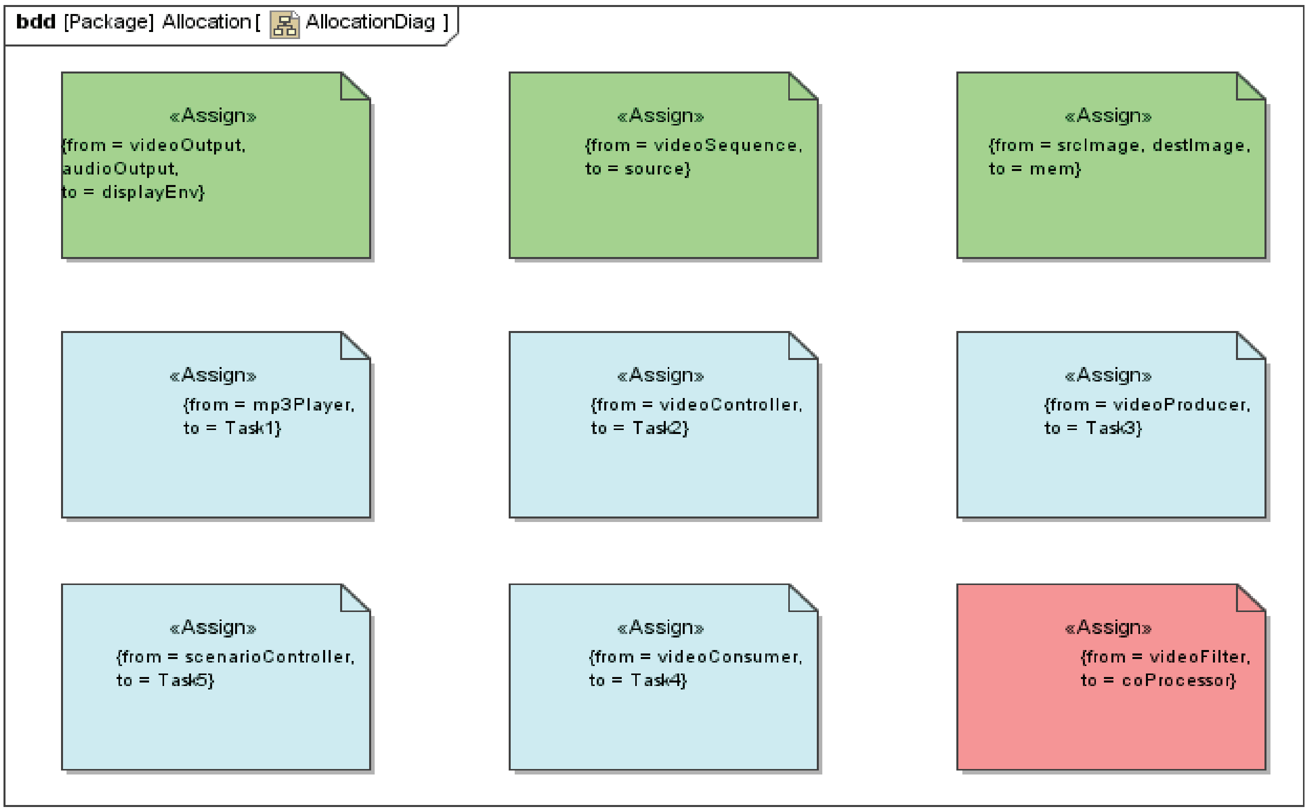
bdd [Package] Allocation [AllocationDiag]
«Assign» {from = videoOutput, audioOutput, to = displayEnv}
«Assign» {from = videoSequence, to = source}
«Assign» {from = srcImage, destImage, to = mem}
«Assign» {from = mp3Player, to = Task1}
«Assign» {from = videoController, to = Task2}
«Assign» {from = videoProducer, to = Task3}
«Assign» {from = scenarioController, to = Task5}
«Assign» {from = videoConsumer, to = Task4}
«Assign» {from = videoFilter, to = coProcessor}

Figure 7: Assignments for Application to Platform Mapping

6. Conclusions

This paper shows how to use MARTE profile with SysML plugin to model a high-level description of an audio-video decoder running on a generic platform (processor + hardware accelerator), in **MultiMediaExample.mdzip** SysML project file (located in *<install.dir>/samples/SysML/MARTE sample* directory). This paper helps you navigate and understand the model in **MultiMediaExample.mdzip**. In addition, it demonstrates how to use MARTE stereotypes with SysML models.