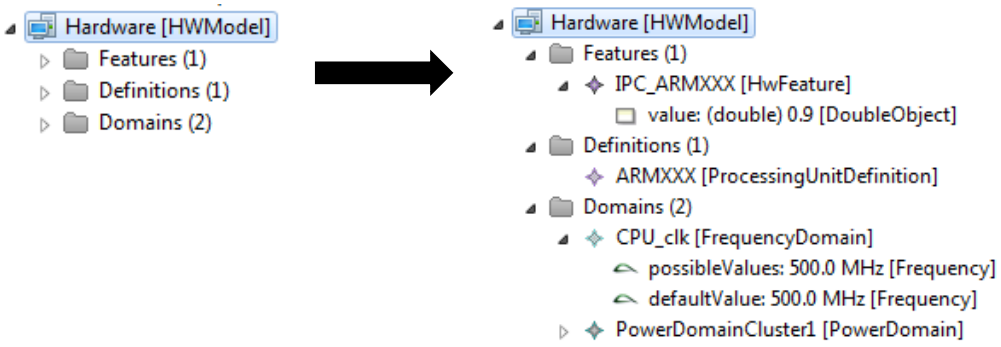
Amalthea HW Examples

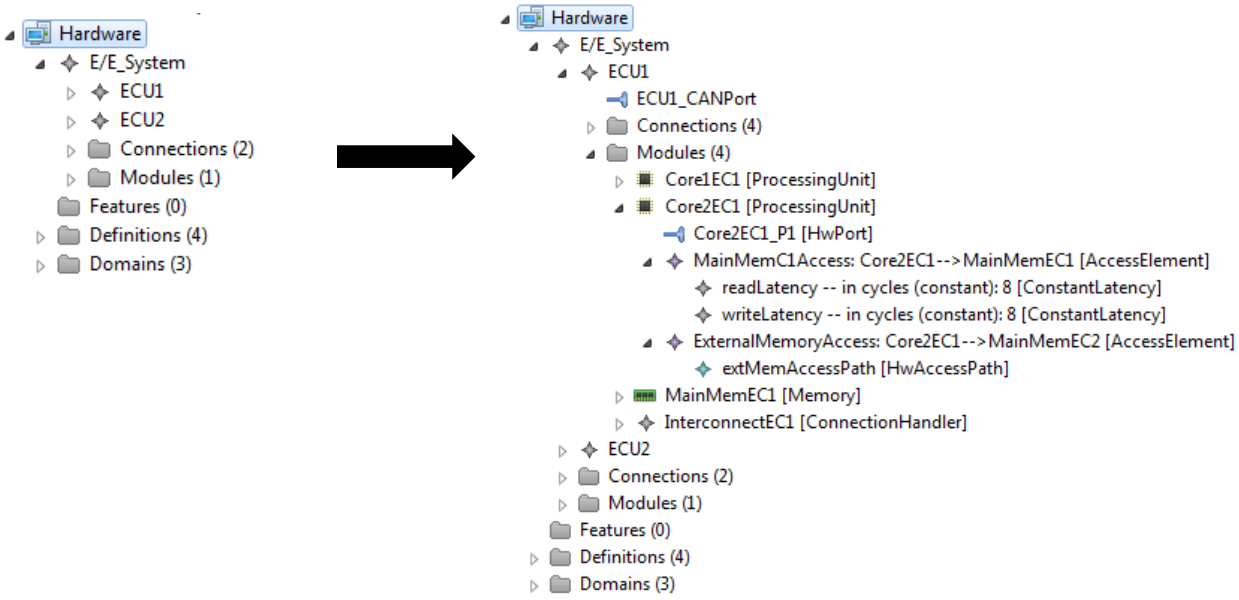
In this documents a short description of the model is given followed by a short description of the two examples of the Amalthea hardware model proposal. This document is intended for users who start work with the new hardware model.

General model and editor overview:

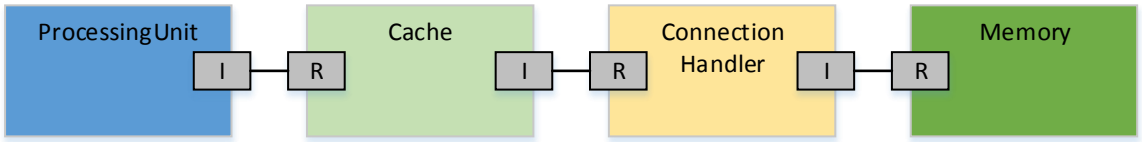
In the following figure the three folders are always present after creating a new *HwModel* in Amalthea. In these folders all *Features*, *Definitions* and *Domains* are collected, which are created by the user. On the right side they are unfolded with some simple examples inside. A very simple feature could be an IPC (Instructions Per Cycle) value which was also used in the previous Amalthea hardware model. To create it a *HwFeature* is used which contains a double value with the corresponding IPC value. This feature can then be referenced by a definition e.g. for a processor. The definitions can be created for every *HwModule* (*ProcessingUnits*, *Caches*, *ConnectionHandlers* and *Memories*). These definitions can later be referenced by the instances of the modules. That means that it is only necessary to define the type of multiples instances just once. This principle is not new it was also included in the old Amalthea HwModel named as types. The last folder contains domains, there are two different kind of domains, *FrequencyDomains* and *PowerDomains*. They can always contain a default value, which includes the value itself and the unit e.g. MHz. They can also contain one or multiple possible values e.g. for a power safe mode.



To create instances of different cores, memories, etc. a so called *Structure* is necessary. Every structure contains two folders (*Modules* and *Connections*) and the possibility to contain further structures. The modules folder include all instances of *Memories*, *ProcessingUnits*, *ConnectionHandler* and *Caches*. The connections include the static *HwConnections* to connect two different ports. Ports can be contained by every module and by structures, the only difference is that in case of a structure it is only a hierarchical port which is represented with help of the port specific attribute “delegated”. The references to the domains and definitions are always set within the modules. In case of a *ProcessingUnit* an access to a *Memory* or other *ProcessingUnit* can be specified with help of a *HwAccessElement*. Dependant of the abstraction level of the module logical paths with help of latencies and data rates can be created or even more detailed *HwAccessPath* were all the static *HwConnections* and *ConnectionHandlers* can be referenced.

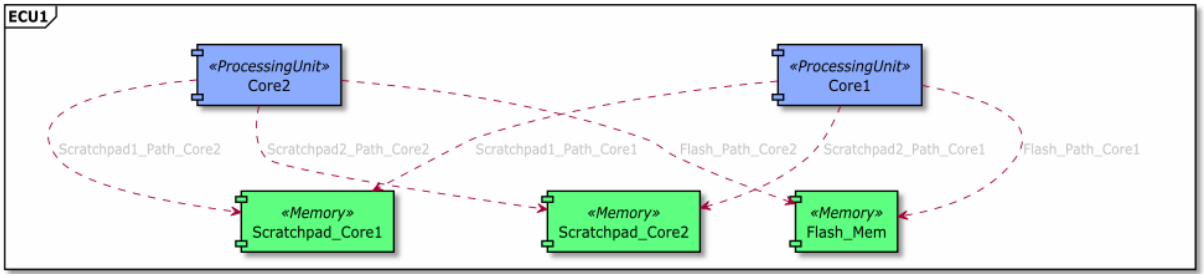


For *HwPorts* it's always possible to select if the port is an initiator or a responder port. The Following example shows our recommendation how they should be used.



I = Initiator
R = Responder

Example1: Simple_ECU

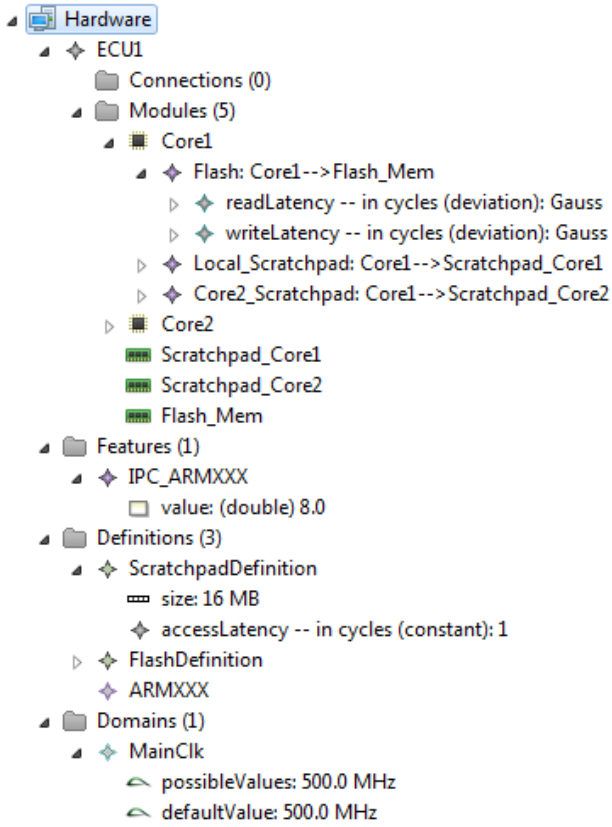


The example shows a single structure with two identical cores with such an Instruction per cycle feature. The model includes only one freuency domain and no power domain. Each core ist connected to all three different memory units via an *HwAccessElement* with read and write latencies. The memory components include an access lateny. This means the total latency for an read and a write access are calculated in the following way:

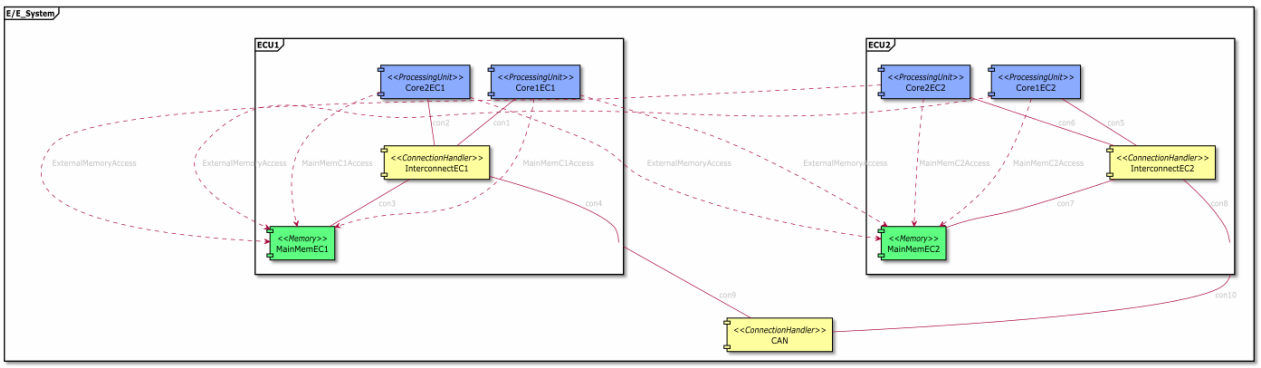
$$TotalReadLatency = readLatency + accessLatency$$

$$TotalWriteLatency = writeLatency + accessLatency$$

This example shows a very simple hardware modelling approach where no interconnects and ports are necessary. Such a model can be used in a very early design phase where only rough esimations or a limited amount of informations about the system are availible.



Example2: Simple_E_E_Architecture



The second example shows a simple E/E-Architecture out of two identical ECUs. Each ECU contains two cores, one interconnect and a memory component. Both ECUs are connected via a CAN-Bus. In this example both possibilities for a *HwAccessElement* is shown. The local memory is just connected with read and write latencies and the external memory of the other ECU is connected with help of a *HwAccessPath*. To use access paths hardware ports and connections between those ports are mandatory. The access paths itself is an ordered list of elements which are used for the connection. As

an example for the access path between Core1EC1 and MainMemEC2 following access path elements are referred:

con1 → internalCan_ECU1 → con4 → con9 → CAN → con10 → con8 → InterconnectEC2 → con7.

That means the complete access paths includes:
3xConnectionHandler and 6xHwConnections.

The latency in this case is the sum of all elements of the path plus the access latency of the memory. However latencies at connections are usually only used to account an offset for a specific component. In case a data rate is used the maximum data rate is limited by the lowest data rate in the path. In case of a *ConnectionHandler* the data rate is usually shared between different accesses.

The hierarchical ports from both ECUs to connect the CAN Bus with the ECUs as block boxes are not mandatory but recommended. This concept of hierarchical ports increases the number of *HwConnections* but allows also the structuring of all internal modules within a *HwStructure* and only connect the hierarchical ports with the rest of the system.