# Reference of the JSon format of Placer format: PlacerBeta5 and PlacerBeta5Out

TANGO H2020

Renaud De Landtsheer

October 2018

CETIC ASBL – Centre d'Excellence en Technologies de l'Information et de la Communication
Aéropole – Avenue Jean Mermoz 28 – B-6041 Charleroi
e-mail : info@cetic.be - TVA BE 0474 549 932

## Table of content

# 1 License

This document is about the alpha release of Placer (with the tag alpha2). The source code for Placer is available at this link: https://github.com/TANGO-Project/placer

# 2 Input

This document defines the format of the JSon file input by Placer. It is the version PlacerBeta5

The JSon input format of Placer is defined in the finitio language (see `http://www.finitio.io/`). This format is further detailed in this document.

Finitio provides a validation tooling that can validate a given JSon fila against a finitio definition. A finitio definition of the JSon format is joined to the placer executable file.

The input specifies lots of metrics, regarding timing, energy, power, etc. Placer does not perform any check of the unit in use, but you must be coherent in the whole JSon. For instance, timing can be specified in millisecond, energy in millijoules, power in milliwatt, etc. All formulas included in the JSon file must be coherent with the selected convention. Similarly, data size is undefined in the format itself.

To help the user remember the units for data size and for time, they are to be defined in natural language in the JSon file.

The input format is made of three parts: the processing element classes, the hardware model, the software model and the mapping goal. Three fields are also provided to define the time unit (**timeUnit**) and the data unit (**dataUnit**) in use and well as any info related to the problem (**info**). They are solely for documentation purpose and are not processed by Placer. The **JsonFormat** specifies the format in use and is checked by Placer at startup. This is the format mentioned in the title of this document.

```
{
    timeUnit:String
    dataUnit:String
    info:String
    jsonFormat:String
    processingElementClasses:[ProcessingElementClass]
    hardwareModel:HardwareModel
    softwareModel:SoftwareModel
    goal:MappingGoal
    properties:?[NameValue]
    constraints:[Constraint]

}
```

## 2.1 Class model

```
ProcessingElementClass =
    {monoTaskSwitchingTask:MonoTaskSwitchingTaskClass}
    | {multiTaskPermanentTasks:MultiTaskPermanentTasksClass}
```

*Processing element classes* define the available class in the model, namely it defines what is a CPU, an FPGA, etc. There are two possible classes:

1. MonoTaskSwitchingTaskClass

```
SwitchingTask = {
    name:String
    resources:[String]
    properties:[String]
    switchingDelay:?Integer
}
```

This corresponds to a processor core: it can execute only one task at a time, and there is a switching delay between tasks.
*Resources*: used by tasks allocated to the PE (Processing Elements) and released when task is completed. A reference to a resource in a formula resolves to its level of usage.
*Properties*: it defines the set of properties that should be defined for each instance of the declared ProcessingElementClass.
*SwitchingDelay* is the delay that all PE of this class need in order to start a task once they completed one.

2. MultiTaskPermanentTasksClass

```
MultiTaskPermanentTasks = {
    name:String
    resources:[String]
    properties:[String]
}
```

The fields of MultiTaskPermanentTasksClass have the same definitions as the ones of MonoTaskSwitchingTaskClass

## 2.2 Hardware model

```
HardwareModel = {
    name:String
    processingElements:[ProcessingElement]
    busses:[Bus]
    properties:[NameValue]
    Constraints:[Constraint]
}
```

A hardware model has a *name* (which is needed in case of multi-hardware optimization).

*Properties* defines a set of name-value properties that can be referenced in the whole hardware model, notably to test several variations of hardware without having to write separated JSon files.

*Constraints* list a set of constraints that refer only to hardware. Check the constraint section below for the supported constraints. This list is used for multi-hardware optimization, otherwise, you can simply list the constraints in section **Erreur ! Source du renvoi introuvable.**.

### 2.2.1 Processing element

```
ProcessingElement = {
    processorClass:ProcessingElementClassRef
    name:String
    resources:[NameValue]
    properties:[NameValue]
    memSize:Integer
    powerModel:Formula
    multiCore:?Int
}
```

*Processing elements* define instances of the declared classes, and define the value for their available resources and their properties. The defined *resources* and *properties* should match the ones defined in the processing element class.

*memSize* is added as an extra resource. The memory is a special resource that is used both as computation memory of tasks, and as data buffer around transmissions, that's why it it is defined separately to other resources.

The *power model* is a formula that can refer to resources and properties of the processing element, including memSize (a default system attribute) and buzy for MonoTaskSwitchingTask processing elements. Buzy is a variable that is zero when the processing element does not perform a task, and 1 when the processing element performs a task. It is only available on MonoTaskSwitchingTask processing elements. More details in Section 2.5.4.

*multiCore* can be used solely for Switching task processing elements. This attribute was proposed to represent GPGPU for instance. It specifies how many cores there are in the processing element. Notice that for such multi-core, we do not consider any data transfer between cores, so if you need this detail, you should instead specify one PE per core and represent the bus that binds these cores together.

### 2.2.2 Busses

There are two types of busses

```
Bus = {halfDuplexBus:HalfDuplexBus}
      |{singleWayBus:SingleWayBus}
```

#### 2.2.2.1 Half duplex busses

```
HalfDuplexBus = {
    name:String
    relatedProcessors:[ProcessingElementRef]
    timeUnitPerDataUnit:Integer
    latency:Integer
}
```

*Half duplex* busses are shared communication media links like Ethernet bus: everyone connected to the bus can talk with everyone, but at any time the bus is at most one message flowing on the bus in only one direction. In other words, when a message is sent on the bus, it prevents other messages from access to the bus until the message is fully transmitted and the bus free again.

They are characterized in speed by their throughput in *timeUnitPerDataUnit*, and their *latency*.

#### 2.2.2.2 Single way busses

```
SingleWayBus = {
    name:String
    from:[ProcessingElementRef]
    to:[ProcessingElementRef]
    timeUnitPerDataUnit:Integer
    latency:Integer
}
```

*Single way busses* are simpler busses with fixed direction for the data flowing on them: A processing element included in “*from*” can transmit to any processing element included in “*to*”. Their purpose is to represent transmission lines with asymmetric speed, such as ADSL.

## *2.3 Software model*

```
SoftwareModel = {
    softwareClass:SoftwareClass
    sharedPermanentFunctions:[Implementation]
    tasks:[Task]
    transmissions:[Transmission]
    properties:[NameValue]
}
```

*A software model* is made of the *tasks*, the *transmissions* between these tasks, and the *software class*.

We can define a set of *sharedPermanentFunctions*. These are meant to be instantiated on multiTaskPermanentTask processing elements. If one such sharedFunction is instantiated, all task that can be implemented by it can execute it, but only one at a time.

```
SoftwareClass = “oneShotSoftware” | “iterativeSoftware”
```
*Software classes* define how the set of tasks must be executed.

- *One shot software* must execute the tasks as quickly as possible.
- *Iterative software* must execute the tasks repeatedly.

As such they support an additional vocabulary for expressing constraints, see section 2.4

### 2.3.1 Tasks and implementations

```
Task = {
    name:String
    implementations:[Implementation]
    sharedImplementation:[ImplementationRef]
}
```

A *simple task* has a *name*, and a set of *implementations*. *sharedImplementations* refer to the ones that are declared in the software model. Placer has to select one implementation among the set of implementation and shared mplementation.

```
Implementation = {
    name:String
    target:ProcessingElementClassRef
    parameters:[ParameterDefinition]
    resourceUsage:[NameFormula]
    computationMemory:Formula
    duration:Formula
    nbThreads:?Formula,
}
```

An *implementation* has a *name*, a defined *target* of which it can run.

Implementations can be parameterized. For instance, matrix multiplication can be performed by block where the size of the block. The *parameters* can be referenced in the formula defining *resource usage*, *computation memory* and *duration* of the implementation.

*resourceUsage* defines the resource usage for each resource that is supported by the target processing element class.

*nbThreads* specifies how many threads there are in the implementation. These threads will be scheduled on the cores of a single multicore processing element. They can solely be specified for implementation targeting switching processing elements. The number of treads can depend on the parameter of the implementation.

```
ParameterDefinition = {
   name:String
   values:[Integer]
}
```

### 2.3.2 Transmissions

```
Transmission = {
   name:String
   source:TaskRef
   target: TaskRef
   size:Integer
   timing:TransmissionTimingConstraint
}
```

*transmissions* occur from a task to another task. They are quantified by their *size*.

```
TransmissionTimingConstraint = String
```

Transmissions have a *timing* constraint. It can be :

- *Asap*: the transmission must occur as soon as the transmissing process terminates
- *Alap*: it must occur such that the target process starts immediately after completion of the transmission.
- *Free*: Placer will chose the best timing for the transmission to occur

- *Sticky*: Places selects between Alap and Asap

## 2.4 Constraints and objectives

Additional constraints can be specified to Placer in the constraints field.

TaskPE defines a couple (task and PE based on their names and is used in the constraint definition herebelow.

```
TaskPE = {
    task:TaskRef
    processingElement:ProcessingElementRef
}
```

The supported constraints are as follows:

```
Constraint =
     {mustBeUsed:ProcessingElementRef}
   | {mustNotBeUsed:ProcessingElementRef}
   | {samePE:[TaskRef]}
   | {notSamePE:[ProcessingElementRef]}
   | {symmetricPE:[ProcessingElementRef], breaking:”longTask”|”workload”}
   | {symmetricTasks:[TaskRef]}
   | {runOn:TaskPE}
   | {notRunOn:TaskPE}
   | {powerCap:Int}
   | {energyCap:Int}
   | {maxMakespan:Int}
   | {maxWidth:Int}
   | Objective
```

A constraint can be:
- *mustBeUsed*: specifies that a processing element must be used by at least one task
- *mustNotBeUsed*: specifies that a processing element must not be used by any task
- *samePE*: specifies a list of task that should be executedd o nthe sameprocessing element
- *notSamePE*: specifies that all the speccified tasks must execute on different processing elements
- *symmetricPE*: specifies that a set of processing element are symmetric. Placer will use this infromation to reduce its search space. It can use it in two different ways, either it breaks on the workload of the PE or it breaks on the lognest task located on these PE. You can specify these two possibilities throughthe *breaking* field
- *symmetricTasks:* specifies that a set of tasks are symmetric. Placer uses this information to speed up its searh.
- *runOn*: specifies that some task must execute on some processing element
- *notRunOn*: specifies that some task must not execute on some processing element
- *powerCap:* specifies a maximal power consumption that the hardware platform cannot exceed at any time
- *energyCap:* sets the maximal amoiunt of energy that the hardware can use to execute all tasks
- *maxMakespan:* specified the maximal value to the end of all tasks

- *maxWidth:* specifies the maximal delay for each processing element of the start of the first task executing on it, and the end of the las task executing on it. This is only accessible for iterative software classes
- *objective:* specify an objective that Placer has to optimize. Only one of such objective can be specified.

### 2.4.1 Mapping objective

The *mapping goal* defines what must be minimized when computing the mapping.

```
objective =
    {multiObjective:MultiObjectiveGoal}
    | {simpleObjective:SimpleGoal}
```

The goal can be mono-objective, so it minimizes one simple goal, taken among the supported goals, or it can also be multi-objective.

```
SimpleGoal = String
```

Simple goal can be:

- *minMakespan* : Minimize the makespan: the duration between the start of the first task, and the end of the last task.
- *minEnergy* : Minimize the energy consumed to execute the whole schedule once (considering one shot applications)
- *minFrame* : for iterative software: minimize the amount of time before the same schedule can be started again in a pipe-line fashion

```
MultiObjectiveGoal = {
    a:SimpleGoal
    b:SimpleGoal
}
```

Multi-objective optimization will compute the optimal trade-offs minimizing two simple optimization goals.

## *2.5 Auxiliary fields*

### 2.5.1 References

```
ProcessingElementRef = String
```

This is a reference to a processing element declared in the hardware model. The reference uses the name as a reference key.

```
ProcessingElementClassRef = String
```

This refers to a processing element class defined in the hardware model and uses the name of the class as a key.

```
ImplementationRef = String
```

An *ImplementationRef* refers to an implementation by its name. such refered implementations are typically shared implementations defined in the software model

```
TaskRef = String
```

A *taskRef* refers to a simple process declared in the software model. It uses the name of the simple process as a key.

### 2.5.2 NameValue

```
NameValue = {
    name:String
    value:Integer
}
```

This defines a value for a given name. It is used throughout the JSon file when an attribute needs to be given a value.

### 2.5.3 NameFormula

NameFormula = {
  name:String
  formula:Formula
}

This defines a formula for a given name. It is used throughout the JSon file when an attribute needs to be given a value.

### 2.5.4 Formulas

```
Formula = String
```

*Formulas* are built based on identifiers and operators.

- identifiers are all identifiers in the scope of the formula:
- if the formula is defined in an implementation, it can refer to attributes of the implementation, attributes of the target processing element class and/or global properties of the hardware.
- if the formula is defined in a ProcessingElement, it can refer to resources and properties of the processing element, including memSize (a default system attribute) and buzy for MonoTaskSwitchingTask processing elements

*operators* include: + - * / () and integer literals.

This is an example of formula: *buzy*mflops*30+5*

# 3 Output

```
{
    timeUnit:String
    dataUnit:String
    info:String
    jsonFormat:String
    mapping:[Mappig]
}
```

*timeUnit*, *dataUnit* and *info* are coopied fro the input file. *jsonFormat* is set to the output format defined in this document.

```
Mapping = {
    hardwareName:String
    makespan:Int
    width:?Int
    energy:Int
    sharedFunctions:[SharedFunctionMapping]
    taskMapping:[TaskMapping]
    transmissionMapping:[TransitionMappig]
}
```

*hardwareName* in the name of the hardware platform.
*Makespan* is the makespan of the mapping
*Width* is the width of the mapping, as defined in the input file. This is only present if the software is an iterative software.
*Energy* is the energy needed to execute this mapping.

### 3.1 sharedFunctions

```
SharedFunctionMapping = {
   sharedImplem:SharedFunctionRef
   parameters:[NameValue]
   nbInstance:Int
   host:ProcessingElementRef
}
```

### 3.2 taskMapping

```
{
    task:TaskRef
    processingElement:ProcessingElementRef
    implementation:ImplementationRef
    parameters:NameValue
    start:Int
    duration:Int
    end:Int
}
```

### *3.3 transmissionMapping*

```
{
    transmission:TransmissionRef
    bus:BusRef
    start:Int
    duration:Int
    end:Int
}
```

## 4 Annex: Example

### *4.1 Input*

```
{
  "timeUnit":"ms",
  "dataUnit":"bit",
  "jsonFormat":"PlacerBeta5",
  "processingElementClasses":[
    {
      "switchingTask":{
        "name":"cpu",
        "resources":[],
        "properties":[
          "mflops",
          "mips"
        ],
        "switchingDelay":1
      }
    },
    {
      "multiTaskPermanentTasks":{
        "name":"fpga",
        "resources":[
          "kgate",
          "multiplier"
        ],
        "properties":[
          "frequency"
        ]
      }
    }
  ],
  "hardwareModel":{
    "name":"ExampleHardware1",
    "processingElements":[
      {
        "processorClass":"cpu",
        "name":"procA",
        "memSize":1000,
        "resources":[],
        "properties":[
          {
            "name":"mflops",
            "value":10
          },
          {
            "name":"mips",
          "value":1000
        }
      ],
      "powerModel":"buzy*mflops*30+5"
    },
    {
      "processorClass":"cpu",
      "name":"procB",
      "memSize":1000,
      "resources":[],
      "properties":[
        {
          "name":"mflops",
          "value":6
        },
        {
          "name":"mips",
          "value":100
        }
      ],
      "powerModel":"buzy*mflops*20+1"
    },
    {
      "processorClass":"fpga",
      "name":"FPGA1",
      "memSize":1000,
      "resources":[
        {
          "name":"kgate",
          "value":110
        },
        {
          "name":"multiplier",
          "value":500
        }
      ],
      "properties":[
        {
          "name":"frequency",
          "value":100
        }
      ],
      "powerModel":"frequency*frequency*(kgate+multiplier*20)/1000+2"
    }
  ],
  "busses":[
    {
      "halfDuplexBus":{
        "name":"globalBus",
        "relatedProcessors":[
          "procA",
          "procB",
          "FPGA1"
        ],
        "timeUnitPerDataUnit":3,
        "latency":1
      }
    },
    {
      "halfDuplexBus":{
        "name":"busAToFPGA",
          "relatedProcessors":[
            "procA",
            "FPGA1"
          ],
          "timeUnitPerDataUnit":1,
          "latency":3
        }
      },
      {
        "halfDuplexBus":{
          "name":"globalBus2",
          "relatedProcessors":[
            "procA",
            "procB",
            "FPGA1"
          ],
          "timeUnitPerDataUnit":4,
          "latency":1
        }
      }
    ],
    "properties":[
      {
        "name":"hardwareSize",
        "value":4
      }
    ]
  },
  "softwareModel":{
    "tasks":[
      {
        "name":"inputting",
        "implementations":[
          {
            "name":"cpu_standard",
            "target":"cpu",
            "parameters":[],
            "resourceUsage":[],
            "computationMemory":"10",
            "duration":"100/mflops+1000/mips"
          },
          {
            "name":"fpga_standard",
            "target":"fpga",
            "parameters":[],
            "resourceUsage":[
              {
                "name":"kgate",
                "formula":"100"
              },
              {
                "name":"multiplier",
                "formula":"30"
              }
            ],
            "computationMemory":"100",
            "duration":"500/frequency"
          }
        ]
      },
      {
```

```
    "name":"decoding",
    "implementations":[
      {
        "name":"PI_cpu_supplier1",
        "target":"cpu",
        "parameters":[],
        "resourceUsage":[],
        "computationMemory":"100",
        "duration":"10/mflops+100000/mips"
      },
      {
        "name":"PI_cpu_supplier2",
        "target":"cpu",
        "parameters":[],
        "resourceUsage":[],
        "computationMemory":"10",
        "duration":"100/mflops+1000/mips"
      },
      {
        "name":"fpga_standard",
        "target":"fpga",
        "parameters":[],
        "resourceUsage":[
          {
            "name":"kgate",
            "formula":"100"
          },
          {
            "name":"multiplier",
            "formula":"20"
          }
        ],
        "computationMemory":"10",
        "duration":"1000/frequency"
      }
    ]
  },
  {
    "name":"Transforming",
    "implementations":[
      {
        "name":"cpu_standard",
        "target":"cpu",
        "parameters":[],
        "resourceUsage":[],
        "computationMemory":"10",
        "duration":"110/mflops+1000/mips"
      }
    ]
  },
  {
    "name":"Transforming2",
    "implementations":[
      {
        "name":"cpu_standard",
        "target":"cpu",
        "parameters":[],
        "resourceUsage":[],
        "computationMemory":"10",
        "duration":"110/mflops+1000/mips"
      }
      ]
    },
    {
      "name":"watermarking",
      "implementations":[
        {
          "name":"cpu_standard",
          "target":"cpu",
          "parameters":[],
          "resourceUsage":[],
          "computationMemory":"0",
          "duration":"100/mflops+1000/mips"
        },
        {
          "name":"PI_fpga_supplier1",
          "target":"fpga",
          "parameters":[],
          "resourceUsage":[
            {
              "name":"kgate",
              "formula":"20"
            },
            {
              "name":"multiplier",
              "formula":"30"
            }
          ],
          "computationMemory":"10",
          "duration":"100/frequency"
        },
        {
          "name":"PI_fpga_supplier2",
          "target":"fpga",
          "parameters":[],
          "resourceUsage":[
            {
              "name":"kgate",
              "formula":"10"
            },
            {
              "name":"multiplier",
              "formula":"35"
            }
          ],
          "computationMemory":"10",
          "duration":"101/frequency"
        }
      ]
    },
    {
      "name":"encoding",
      "implementations":[
        {
          "name":"cpu_standard",
          "target":"cpu",
          "parameters":[],
          "resourceUsage":[],
          "computationMemory":"10",
          "duration":"100+100/mflops+1000/mips"
        },
        {
        "name":"cpu_param",
        "target":"cpu",
        "parameters":[
          {
            "name":"toto",
            "values":[
              0,
              1,
              2
            ]
          }
        ],
        "resourceUsage":[],
        "computationMemory":"5*(toto+1)",
        "duration":"100+100/mflops+1000/mips-10*toto"
      }
    ]
  }
],
"transmissions":[
  {
    "name":"inputToDecode",
    "source":"inputting",
    "target":"decoding",
    "size":50,
    "timing":"Free"
  },
  {
    "name":"decodeToTransform",
    "source":"decoding",
    "target":"Transforming",
    "size":2,
    "timing":"Sticky"
  },
  {
    "name":"transformToWatermark",
    "source":"Transforming",
    "target":"watermarking",
    "size":2,
    "timing":"Asap"
  },
  {
    "name":"decodeToTransform2",
    "source":"decoding",
    "target":"Transforming2",
    "size":2,
    "timing":"Sticky"
  },
  {
    "name":"transform2ToWatermark",
    "source":"Transforming2",
    "target":"watermarking",
    "size":2,
    "timing":"Asap"
  },
  {
    "name":"watermarkToEncode",
    "source":"watermarking",
    "target":"encoding",
    "size":20,
    "timing":"Asap"
      },
      {
        "name":"side_comm",
        "source":"inputting",
        "target":"encoding",
        "size":5,
        "timing":"Free"
      }
    ],
    "softwareClass":"iterativeSoftware"
  },
  "constraintsAndObjectives":[
    {
      "multiObjective": {
        "a": "minMakespan",
        "b": "minEnergy"
      }
    },
    {
      "powerCap":700
    },
    {
      "energyCap":500000
    },
    {
      "maxWidth":200
    }
  ]
}
```

## 4.2 Output

```
{
  "timeUnit":"ms",
  "dataUnit":"bit",
  "info":"",
  "mappings":[
    {
      "hardwareName":"ExampleHardware1",
      "makeSpan":189,
      "width":160,
      "energy":30912,
      "sharedFunctions":[],
      "taskMapping":[
        {
          "task":"inputting",
          "processingElement":"procA",
          "implementation":"cpu_standard",
          "parameters":[],
          "start":0,
          "duration":11,
          "end":11
        },
        {
          "task":"decoding",
          "processingElement":"procA",
          "implementation":"PI_cpu_supplier2",
          "parameters":[],
          "start":11,
          "duration":11,
        "end":22
      },
      {
        "task":"Transforming",
        "processingElement":"procA",
        "implementation":"cpu_standard",
        "parameters":[],
        "start":22,
        "duration":12,
        "end":34
      },
      {
        "task":"Transforming2",
        "processingElement":"procB",
        "implementation":"cpu_standard",
        "parameters":[],
        "start":29,
        "duration":28,
        "end":57
      },
      {
        "task":"watermarking",
        "processingElement":"procB",
        "implementation":"cpu_standard",
        "parameters":[],
        "start":57,
        "duration":26,
        "end":83
      },
      {
        "task":"encoding",
        "processingElement":"procB",
        "implementation":"cpu_param",
        "parameters":[
          {
            "name":"toto",
            "value":2
          }
        ],
        "start":83,
        "duration":106,
        "end":189
      }
    ],
    "transmissionMapping":[
      {
        "transmission":"transformToWatermark",
        "bus":"globalBus",
        "start":34,
        "duration":7,
        "end":41
      },
      {
        "transmission":"decodeToTransform2",
        "bus":"globalBus",
        "start":22,
        "duration":7,
        "end":29
      },
      {
        "transmission":"side_comm",
          "bus":"globalBus",
          "start":41,
          "duration":16,
          "end":57
        }
      ]
    },
    {
      "hardwareName":"ExampleHardware1",
      "makeSpan":166,
      "width":166,
      "energy":45488,
      "sharedFunctions":[],
      "taskMapping":[
        {
          "task":"inputting",
          "processingElement":"procA",
          "implementation":"cpu_standard",
          "parameters":[],
          "start":0,
          "duration":11,
          "end":11
        },
        {
          "task":"decoding",
          "processingElement":"procA",
          "implementation":"PI_cpu_supplier2",
          "parameters":[],
          "start":11,
          "duration":11,
          "end":22
        },
        {
          "task":"Transforming",
          "processingElement":"procA",
          "implementation":"cpu_standard",
          "parameters":[],
          "start":22,
          "duration":12,
          "end":34
        },
        {
          "task":"Transforming2",
          "processingElement":"procB",
          "implementation":"cpu_standard",
          "parameters":[],
          "start":29,
          "duration":28,
          "end":57
        },
        {
          "task":"watermarking",
          "processingElement":"procA",
          "implementation":"cpu_standard",
          "parameters":[],
          "start":64,
          "duration":11,
          "end":75
        },
        {
          "task":"encoding",
          "processingElement":"procA",
          "implementation":"cpu_param",
          "parameters":[
            {
              "name":"toto",
              "value":2
            }
          ],
          "start":75,
          "duration":91,
          "end":166
        }
      ],
      "transmissionMapping":[
        {
          "transmission":"decodeToTransform2",
          "bus":"globalBus",
          "start":22,
          "duration":7,
          "end":29
        },
        {
          "transmission":"transform2ToWatermark",
          "bus":"globalBus",
          "start":57,
          "duration":7,
          "end":64
        }
      ]
    },
    {
      "hardwareName":"ExampleHardware1",
      "makeSpan":148,
      "width":148,
      "energy":45584,
      "sharedFunctions":[],
      "taskMapping":[
        {
          "task":"inputting",
          "processingElement":"procA",
          "implementation":"cpu_standard",
          "parameters":[],
          "start":0,
          "duration":11,
          "end":11
        },
        {
          "task":"decoding",
          "processingElement":"procA",
          "implementation":"PI_cpu_supplier2",
          "parameters":[],
          "start":11,
          "duration":11,
          "end":22
        },
        {
          "task":"Transforming",
          "processingElement":"procA",
          "implementation":"cpu_standard",
          "parameters":[],
          "start":22,
          "duration":12,
          "end":34
        },
        {
          "task":"Transforming2",
          "processingElement":"procA",
          "implementation":"cpu_standard",
          "parameters":[],
          "start":34,
          "duration":12,
          "end":46
        },
        {
          "task":"watermarking",
          "processingElement":"procA",
          "implementation":"cpu_standard",
          "parameters":[],
          "start":46,
          "duration":11,
          "end":57
        },
        {
          "task":"encoding",
          "processingElement":"procA",
          "implementation":"cpu_param",
          "parameters":[
            {
              "name":"toto",
              "value":2
            }
          ],
          "start":57,
          "duration":91,
          "end":148
        }
      ],
      "transmissionMapping":[]
    },
    {
      "hardwareName":"ExampleHardware1",
      "makeSpan":185,
      "width":132,
      "energy":31120,
      "sharedFunctions":[],
      "taskMapping":[
        {
          "task":"inputting",
          "processingElement":"procA",
          "implementation":"cpu_standard",
          "parameters":[],
          "start":0,
          "duration":11,
          "end":11
        },
        {
          "task":"decoding",
          "processingElement":"procA",
          "implementation":"PI_cpu_supplier2",
          "parameters":[],
          "start":11,
          "duration":11,
          "end":22
        },
        {
          "task":"Transforming",
          "processingElement":"procA",
          "implementation":"cpu_standard",
          "parameters":[],
          "start":22,
          "duration":12,
          "end":34
        },
        {
          "task":"Transforming2",
          "processingElement":"procA",
          "implementation":"cpu_standard",
          "parameters":[],
          "start":34,
          "duration":12,
          "end":46
        },
        {
          "task":"watermarking",
          "processingElement":"procB",
          "implementation":"cpu_standard",
          "parameters":[],
          "start":53,
          "duration":26,
          "end":79
        },
        {
          "task":"encoding",
          "processingElement":"procB",
          "implementation":"cpu_param",
          "parameters":[
            {
              "name":"toto",
              "value":2
            }
          ],
          "start":79,
          "duration":106,
          "end":185
        }
      ],
      "transmissionMapping":[
        {
          "transmission":"transformToWatermark",
          "bus":"globalBus",
          "start":34,
          "duration":7,
          "end":41
        },
        {
          "transmission":"transform2ToWatermark",
          "bus":"globalBus",
          "start":46,
          "duration":7,
          "end":53
        },
        {
          "transmission":"side_comm",
          "bus":"globalBus",
          "start":11,
          "duration":16,
          "end":27
        }
      ]
    },
    {
      "hardwareName":"ExampleHardware1",
      "makeSpan":217,
      "width":188,
      "energy":30896,
      "sharedFunctions":[],
      "taskMapping":[
        {
          "task":"inputting",
          "processingElement":"procA",
          "implementation":"cpu_standard",
          "parameters":[],
          "start":0,
          "duration":11,
          "end":11
        },
        {
          "task":"decoding",
          "processingElement":"procA",
          "implementation":"PI_cpu_supplier2",
          "parameters":[],
          "start":11,
          "duration":11,
          "end":22
        },
        {
          "task":"Transforming",
          "processingElement":"procB",
          "implementation":"cpu_standard",
          "parameters":[],
          "start":29,
          "duration":28,
          "end":57
        },
        {
          "task":"Transforming2",
          "processingElement":"procB",
          "implementation":"cpu_standard",
          "parameters":[],
          "start":57,
          "duration":28,
          "end":85
        },
        {
          "task":"watermarking",
          "processingElement":"procB",
          "implementation":"cpu_standard",
          "parameters":[],
          "start":85,
          "duration":26,
          "end":111
        },
        {
          "task":"encoding",
          "processingElement":"procB",
          "implementation":"cpu_param",
          "parameters":[
            {
              "name":"toto",
              "value":2
            }
          ],
          "start":111,
          "duration":106,
          "end":217
        }
      ],
      "transmissionMapping":[
        {
          "transmission":"decodeToTransform",
          "bus":"globalBus",
          "start":22,
          "duration":7,
          "end":29
        },
        {
          "transmission":"decodeToTransform2",
          "bus":"globalBus",
          "start":50,
          "duration":7,
          "end":57
        },
        {
          "transmission":"side_comm",
          "bus":"globalBus",
          "start":29,
          "duration":16,
          "end":45
        }
      ]
    }
  ]
}
```