Placer: a Design-time Model-based Tool for Mapping Task-based SW onto Heterogeneous HW

CPS Summer School Sept. 2018

Renaud De Landtsheer

rdl@cetic.be

Content

- Context
  - About research, PhD, industry, etc.
- Placer
  - Heterogeneity
  - What?
  - Why?
  - An example
  - Play with
- Lessons learnt

Ongoing work
To be released by Jan 2019

OscaR as a tech transfer project

- Oscar
  - Open source framework for combinatorial optimization
  - Started in 2011
  - Open source LGPL license
  - https://bitbucket.org/oscarlib/oscar
  - Scala

- Two engines
  - OscaR.CP by UCL Belgium
    - The one we use here
  - OscaR.CBLS by CETIC Belgium
    - My main work
    - Very good scalability
    - No scheduling engine in 2016, start of TANGO
    - No time to build one

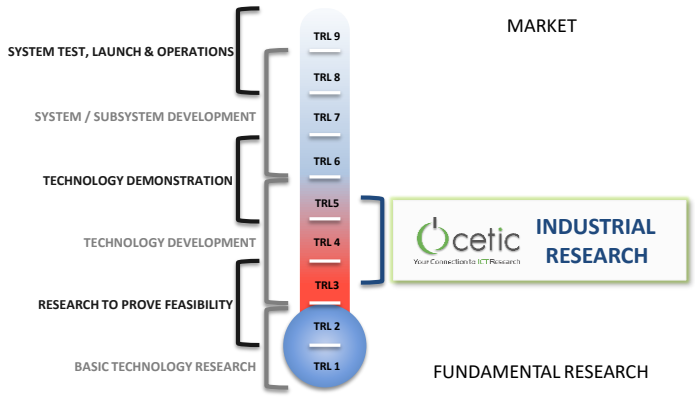
What is CETIC?

- Not an industry
- Not a university

- A bridge between university and industry
  - Technology transfer
  - Push mode: we have some technology
    - from academia, from our own developments
  - Pull mode: industry comes with a problem
    - Requires technical skills, selected knowledge, considered as risky
  - Also a bridge for people

- A « private » research centre
  - May not compete directly against regional companies
    - Must offer more innovative service
  - Research projects; seldom get 100% funding
  - Consulting missions to reach 100% salary pay

Technology Readiness Level



Content

- Hello world
- TANGO
  - Heterogeneity
- Placer
  - What?
  - Why?
  - An example
  - Play with it
  - GUI
  - Under the Hood
    - CP solver
    - Modelling into CP
    - Guiding CP search
    - LNS
- Lessons learnt

TANGO project

- One of my research project
- H2020 research project
  - Start: Jan 2016
  - End: Jan 2019
- Today's topic:
  - Placer tool
  - My contribution to TANGO

Heterogeneity in processing elements

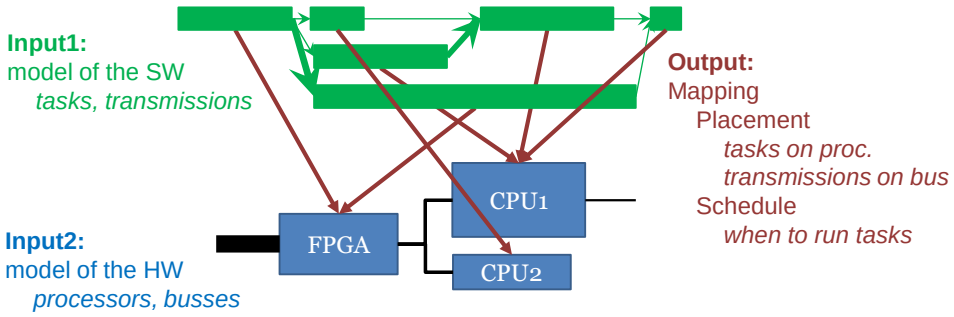
- CPU core
  - Executes one task at a time
  - Switch from one task to another one
- FPGA
  - A large set of configurable hardware gates
  - Can hosts functional blocks
    - Matrix multiplication, FFT, etc.
    - Also several blocks at the same time, provided it has enough gates
  - Each functional blocks
    - Is permanently allocated some gates
    - Can be running independently of the others
  - Configuration
    - Must be performed at start up (or burnt in)
    - Cannot be changed during execution (simplifying assumption here)
- GPGPU
  - A set of very small cores (like CPU cores)
  - Can run several tasks at the same time
  - Each executing tasks uses a set of cores allocate to it
  - Upon completion, the cores are free for another task

Content

- Hello world
- TANGO
  - Heterogeneity
- Placer
  - What?
  - Why?
  - An example
  - Play with it
  - GUI
  - Under the Hood
    - CP solver
    - Modelling into CP
    - Guiding CP search
    - LNS
- Lessons learnt

What is Placer?

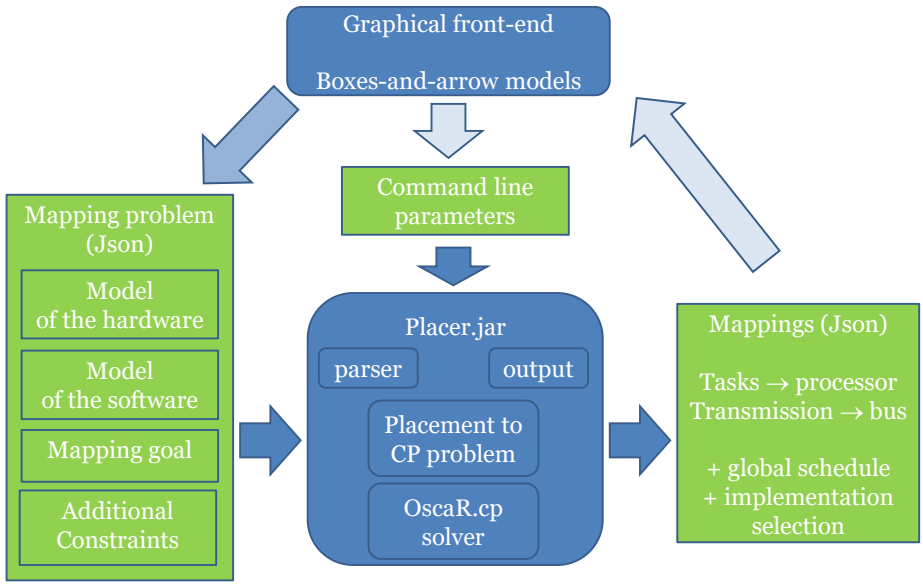
- Model-based Design time tool
- Find the best mapping of software onto hardware



Specification of Placer

- **Given**
  - Meta info
    - Relevant processor classes and their resources
  - Model of Software
    - Tasks
      - Implementations (several possible, for declared targets classes)
    - Transmissions between tasks
      duration = throughput * dataSize + latency
  - Model of Hardware
    - Processing elements (of the declared classes)
    - Busses
- **Find** mapping and schedule
  - Task → (processing element, implementation, timing)
  - Transmission → (bus, timing)
- **Such that**
  - Capacities are not exceeded
  - Minimize: Timing or Energy (or both)

Architecture of Placer



About Placer

- Written in Scala
- Open source LGPL
  https://github.com/TANGO-Project/placer
- Developed as part of the TANGO H2020
  www.tango-project.eu/
- Ongoing work
  - To be released by end 2018
  - Developed by me, myself and RDL
    - 20%-time job over two years
  - Graphical modelling tool by ongoing internship

Why Placer?

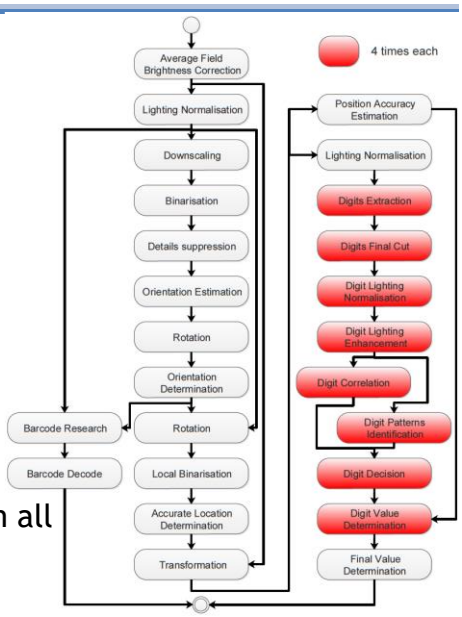
- External reasons
  - Design-time mapping tool
    - Specifically targeting heterogeneous platforms
      - Chose between FPGA or CPU or GPGPU
      - Make a global decision: mapping and timing
  - Distributed in open source
    no closed source dependencies, no licence key
    (except the GUI that relies on a community edition JAR)
- Internal reasons
  - Validate that CP can be useful to solve HW SW mapping of industry use case in the world of embedded systems
    - (Not too many tasks)
  - Learn about the applicability of CP to multi-modal problems
  - Bring the HW SW mapping problem to the (local) CP community
    - Known as a “flexible job-shop problem”
    - So far considered as an open problem

Content

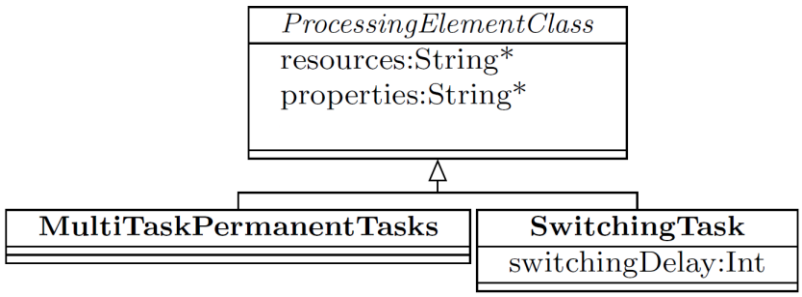
- Hello world
- TANGO
  - Heterogeneity
- Placer
  - What?
  - Why?
  - An example
  - Play with it
  - GUI
  - Under the Hood
    - CP solver
    - Modelling into CP
    - Guiding CP search
    - LNS
- Lessons learnt

Embedded use case: AquaScan

- Software
  - nbTasks:51
  - nbTransmissions:62
  - Realistic
    - Durations
    - Data transfer
  - Lacking
    - Energy data
- Hardware:
  - 4 identical cores
  - A single bus connecting them all
- Intuitive parallelization …



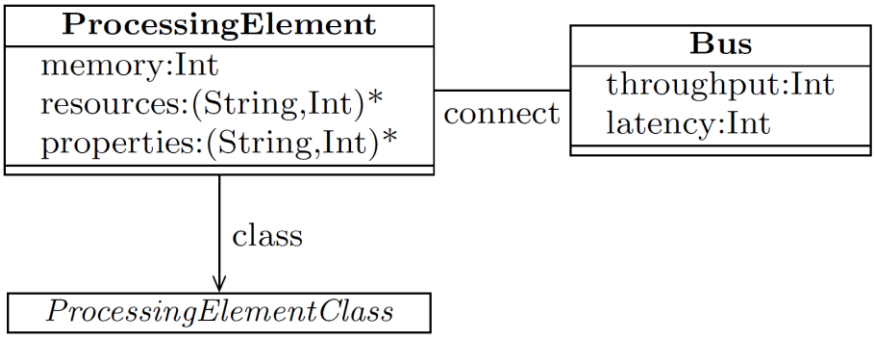
Declaring the meta info



Example of meta info

```
"processingElementClasses":
[
    {
        "switchingTask":
        {
            "name":"cpu",
            "resources":[],
            "properties":[],
            "switchingDelay":1
        }
    },
    {
        "multiTaskPermanentTasks":
        {
            "name":"fpga",
            "resources":["gates"],
            "properties":[]
        }
    }
],
```

Modelling the hardware



Example of processing element

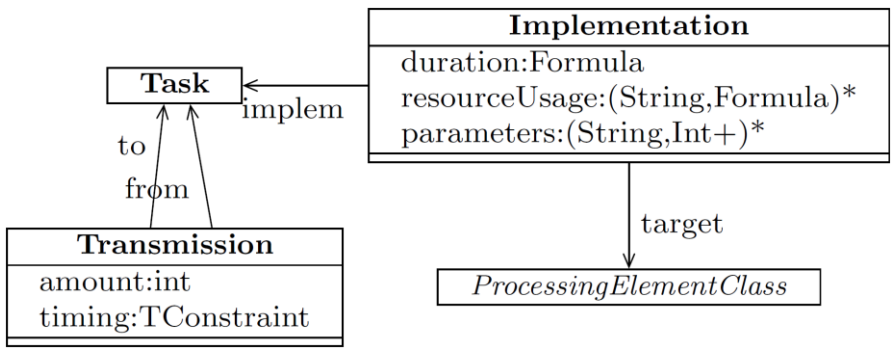
```
"hardwareModel":
{
    "name":"ExampleHardware1",
    "processingElements":
    [
        {
            "processorClass":"cpu",
            "name":"Core1",
            "memSize":32768
        },
        {
            "processorClass":"fpga",
            "name":"Fpga1",
            "memSize":32768,
            "resources":[{"name":"gates", "value":1000}]
        }
    ]
}
```

Memory used for
- computation
- data buffering around transmissions

Example of bus

```
"busses":
[
    {
        "halfDuplexBus":
        {
            "name":"globalBus",
            "relatedProcessors":
            [
                "Core1",
                "Core2",
                "Core3",
                "Core4"
            ],
            "timeUnitPerDataUnit":1,
            "latency":1
        }
    }
```

Modelling the software



Example of task

```
{
    "name":"Digit1Correlation",
    "implementations":
    [
        {
            "name":"CpuDigit1Correlation",
            "target":"cpu",
            "computationMemory":"2",
            "duration":"43205"
        },
        {
            "name":"FpgaDigit1Correlation",
            "target":"fpga",
            "resourceUsage":[{"name":"gates", "formula":"300"}],
            "computationMemory":"2",
            "duration":"8640"
        }
    ]
},
```

Example of transmission

```
"transmissions":
[
    {
        "name":"InputToAvgFieldBrightnessCorrection",
        "source":"Input",
        "target":"AvgFieldBrightnessCorrection",
        "size":8192,
        "timing":"Sticky"

    },
```

Timing contraint on transmission can be:
Free, ASAP, ALAP, Sticky

Constraints and objective

- Objective
  - minMakespan
  - minEnergy
  - pareto(minMakespan,minEnergy)
- Constraints
  - samePE(tasks)
  - notSamePE(tasks)
  - runOn(task,pe)
  - notRunOn(task,pe)
  - powerCap(maxPower)
  - energyCap(maxEnergy)
  - maxMakespan(deadline)

Content

- Hello world
- TANGO
  - Heterogeneity
- Placer
  - What?
  - Why?
  - An example
  - Play with it
  - GUI
  - Under the Hood
    - CP solver
    - Modelling into CP
    - Guiding CP search
    - LNS
- Lessons learnt

Let's min the makespan (no FPGA)

- Placer does not parallelize the intuitive way

```
makeSpan:82889 us
Core1:
   Digit3PatternsIdentification
   Digit4Correlation
Core2:
   PositionAccuracyEstimation
   Digit1Correlation
Core3:
   BarcodeResearch
   Digit3Correlation
   ...
Core4:
   Input
   AvgFieldBrightnessCorrection
    ...
```

- More info:
  - `Digit`**`X`**`Correlation` lasts for 50% of the make span
  - Core4 is used at 99,3% (it waits vey little for data)
- Not happy with parallelization: not the expected one

Specifying constraints

- Grouping the tasks by stream to be the same core

```
{
  "samePE":[
    "Digit"X"Extraction",
    "Digit"X"FinalCut",
    "Digit"X"LightingNormalisation",
    "Digit"X"LightingEnhancement",
    "Digit"X"Correlation",
    "Digit"X"PatternsIdentification",
    "Digit"X"Decision",
    "Digit"X"ValueDetermination"
  ]
}
```

With X in 1..4

Optimizing with constraints

- Desired parallelization is obtained
- Schedule is slightly longer, by 1.2%

```
makeSpan:83911 us
Core1:
    Digit4…
Core2
    Digit3…
Core3
    Digit2…
Core4
    Input
    …
    Digit1 …
    Output
```

(original makeSpan: 82889 us)

Min make span with FPGA no constraint

- Some tasks are located on FPGA
- Only 3 out of 4 can fit on FPGA, so speedup is only 2%

```
makeSpan: 81017 us
Core1:
    ...
Core2:
    ...
Core3:
    ...
Core4:
    Input
    ...
Fpga1:9615:
    Digit2Correlation
    Digit3Correlation
    Digit4Correlation
```

(original makeSpan: 82889 us)

Try out a bigger FPGA

- From 1000 to 1200 gates

```
{
    "processorClass":"fpga",
    "name":"Fpga1",
    "memSize":32768,
    "resources":[{"name":"gates", "value":1200}],
    "properties":[],
    "powerModel":"0"
}
```

- Now all tasks fit

makeSpan:46602

- Good, but FPGA is under-used in time

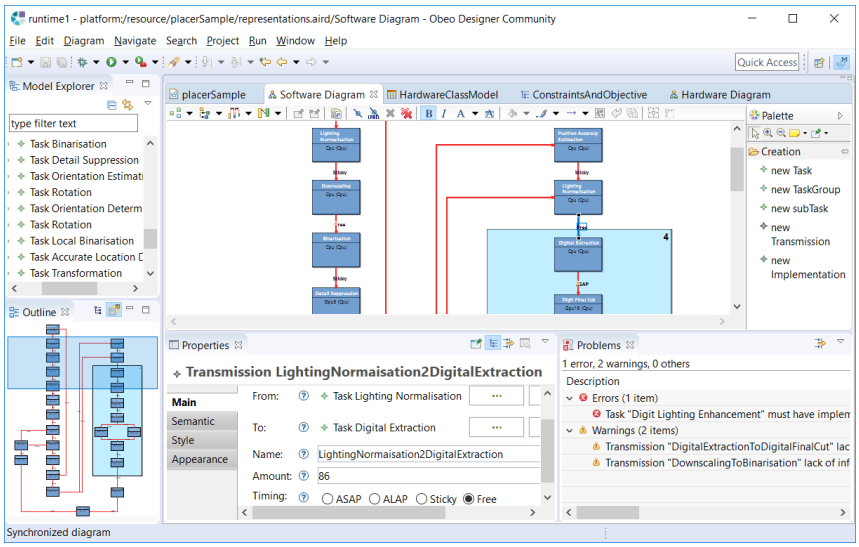
Shared implementation on FPGA

- Instantiate an implementation once, so that several tasks can be executed by this instantiated implementation
- Placer has to decide
  - how many implementation to instantiate
    - which what parameters
  - which task use which instantiation
    - In case they have different parameters
  - and to schedule the tasks accordingly

Content

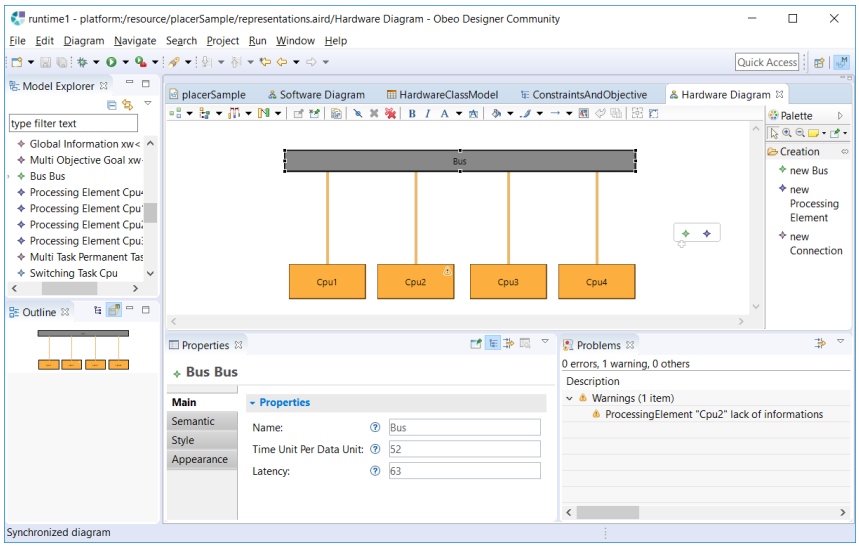
- Hello world
- TANGO
  - Heterogeneity
- Placer
  - What?
  - Why?
  - An example
  - Play with it
  - GUI
  - Under the Hood
    - CP solver
    - Modelling into CP
    - Guiding CP search
    - LNS
- Lessons learnt

Graphical editor (software view)



Eclipse plug-in based on Obeo designer, community edition
By Romain Launay, ESEO engineering school, FR, intern at CETIC

Graphical editor (hardware view)



Content

- Hello world
- TANGO
  - Heterogeneity
- Placer
  - What?
  - Why?
  - An example
  - Play with it
  - GUI
  - Under the Hood
    - CP solver
    - Modelling into CP
    - Guiding CP search
    - LNS
- Lessons learnt

Under the hood: OscaR.cp

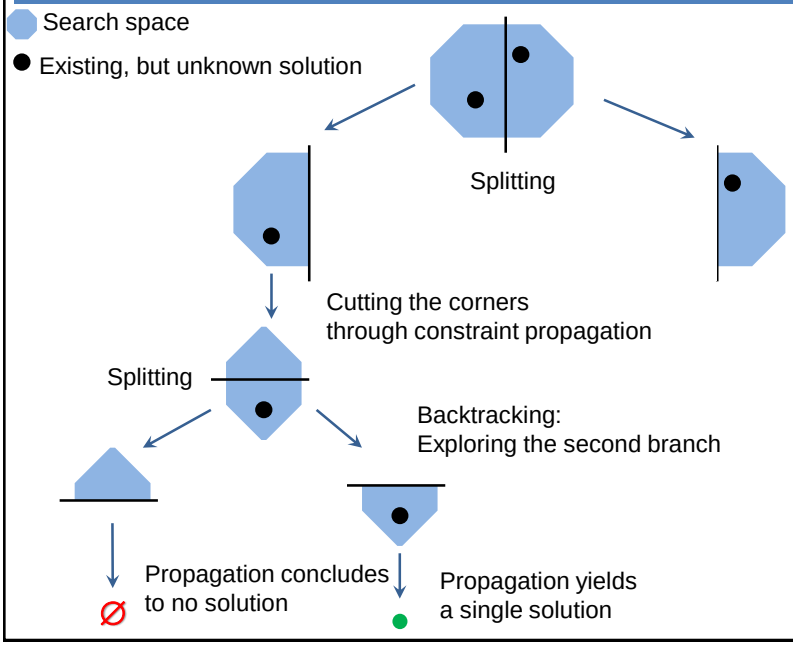
- Oscar
  - Open source framework for combinatorial optimization
  - Started in 2011
  - Open source LGPL license
  - https://bitbucket.org/oscarlib/oscar
  - Scala

OscaR
OPERATIONAL RESEARCH IN SCALA

- Two engines
  - OscaR.CP — by UCL Belgium
  - OscaR.CBLS — by CETIC Belgium

	Completeness	Scalability	Expressivity
CLBS	No	+++	++
CP	Yes	+	+++
MIP	Yes	++	+

Constraint programming in one slide



An example of CP script

```
val nQueens = 1000 // Number of queens
val Queens  = 0 until nQueens

// Variables
val queens = Array.fill(nQueens)(CPIntVar(Queens))

// Constraints
add(allDifferent(queens))
add(allDifferent(Queens.map(i => queens(i) + i)))
add(allDifferent(Queens.map(i => queens(i) - i)))

// Specifying the search strategy
search(binaryFirstFail(queens))

val stats = start(nSols = 1)

println(stats)
```

White magics

Black magics

Both the model and the search strategy
- **are critical to have good overall efficiency**
- **have no direct impact on completeness**

Content

- Hello world
- TANGO
  - Heterogeneity
- Placer
  - What?
  - Why?
  - An example
  - Play with it
  - GUI
  - Under the Hood
    - CP solver
    - Modelling into CP
    - Guiding CP search
    - LNS
- Lessons learnt

Translating Tasks to CP

- Variables are
  - start, duration, end

```
val start: CPIntVar = CPIntVar(0, maxHorizon)
```

  - processorID
  - implementationID
- Constraints on Tasks
  - (duration,target,implementation) ∈ …

```
table(implementationID, processorID, duration,
                implemAndProcessorAndDurations)
```

  - timing

```
end === (start + duration)
```

Lessons learnt

- Table constraint does not work on large domains
  - Problem
    - From industrial case, duration of transmission and tasks can be very different depending on target
      From 0 to 8.000.000
    - Table constraint iterate over full domain of variable

```
table(implementationID, processorID, duration,
                implemAndProcessorAndDurations))
```

  - Solution
    - Do not put duration in the table constraint
    - Use a durationID in the table, and map the duration ID to the actual duration using a constraint that does not iterate over the domain

```
table(implementationID, processorID, durationID ,
                implemAndProcessorAndIndice)
element(durationIDToActualDurationArray,
        durationID,duration)
```

Translating Transmissions to CP

- Variables are
  - start, duration, end
  - busID
- Constraints on Transmissions
  - (EmittingTask.target,target,ReceivingTask.target) ∈ …

```
table(originProcessorID, busID, destinationProcessorID,
                processorToBusToProcessorAdjacency))
```

  - (duration,target) ∈ …
  - EmittingTask.end <= start
  - End <= ReceivingTask.start

Translating Busses and processors to CP

- Generally posted as resource constraints

```
maxCumulativeResource(startTimeArray, durationArray,
                              endArray, amountArray,
                              maxResource)
```

- FPGA posted as bin-packing constraints, on each resource dimension

```
weightedSum(concatenatedRequirementsArray,
              concatenatedUsageArray,resourceToUsage(resource))
```

Content

- Hello world
- TANGO
  - Heterogeneity
- Placer
  - What?
  - Why?
  - An example
  - Play with it
  - GUI
  - Under the Hood
    - CP solver
    - Modelling into CP
    - Guiding CP search
    - LNS
- Lessons learnt

Search strategies

- What makes a CP search strategy a good one?
- CP performs a depth-first-search, left to right exploration
- General principle:
  If an incorrect decision is taken,
  fail as quickly as possible
  - So find the key decision that produce the most constraint on the overall solution, and set these first
    - If the decision is wrong, it will fail quickly
    - If the decision is right, you win
  - Use a “good enough” heuristic to make the correct decision

Search Strategies

- Pure scheduling can be solved efficiently

```
conflictOrderingSearch(
  taskAndTransmissionStarts,
  taskAndTransmissionStarts(_).size,
  taskAndTransmissionStarts(_).min)
```

  - Performs some learning about the decision that are to be taken high in the tree

- … but we do not have a pure scheduling problem, placement greatly impacts the scheduling constraints

[COS2015] Conflict Ordering Search for Scheduling Problems, Steven Gay, Renaud Hartert, Christophe Lecoutre, Pierre Schaus, 2005

Search strategies

- What constraints the overall solution?
  - In our case, the placement of the longest tasks
  - Distribute on task placement, the longest tasks first

```
conflictOrderingSearch(
        processorIDChoices,
        taskMaxDurations(_),
        processorIDChoices(_).minBy(procID => processorLoadArray(procID)))
```

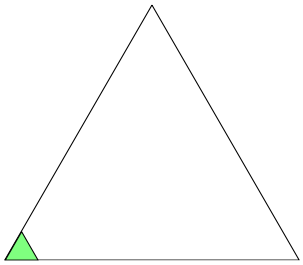
- Overall search strategy is a composite

```
distributeOnTaskPlacementLessBuzyProcFirst
++ distributeOnTransmissionRouting
++ distributeOnTaskAndTransmissionStarts
```

- Finally, you can also set it through the command line

Large Neighbourhood Search (LNS)

- CP search explores a search tree
- Revising first decisions
  - requires exploring the full subtree rooted at the decision
  - This can still take time
- What if some initial decisions are not good?
  - Search might be stuck with this decision for long time



- LNS : repeatedly perform CP search on partially decided problem

```
currentSol = findFeasibleSolution(problem)
While(…){
   partialAssignment = selectSomeAssigments(currentSol)
   currentSol = findBest(problem
                         ∧ partialAssignment
                         ∧ obj < currentSolution.obj)
}
```

Selecting assignments in LNS

- What is a good assignment selection?
  - Not restricted to assignments performed at the bottom of the search tree
  - Enable the revision of assignments made early in the search tree
  - Identify linked assignments
    - Ex: sameCore Constraints
      - constraints two tasks to be on the same processing element
      - You need to release all placement of these tasks together

Relaxation procedure

- Release all schedule, and 90% of placement
  - Works great except when dealing with shared parametric implementations on FPGA
- Release schedule and [illegible] sks
- Rele[illegible] influence zone
- Re[illegible] placed on a subset of the processing ele[illegible]
- Combination of the above

Ongoing work

Content

- Hello world
- TANGO
  - Heterogeneity
- Placer
  - What?
  - Why?
  - An example
  - Play with it
  - GUI
  - Under the Hood
    - CP solver
    - Modeling into CP
    - Guiding CP search
    - LNS
- Lessons learnt

Lessons learnt

- Overflow
  - Problem
    - Industrial used picosecond as a time unit = 10^{-12} s
    - All numbers went huge, since tasks can take milliseconds
    - Overflow happened within solver, not detected, silent error
      - Common issue in combinatorial solvers (Gecode, GoogleCP, OscaR)
  - Solution
    - Add overflow detection in input data,
      check the upper bound of biggest values against MaxInt
      (and perform these checks using long)

Lessons learnt

- Several models with subtle differences
  - Problem
    - Different models with subtle differences
    - Quickly got lost in what represents what, waste of time
  - Solution
    - Add an « info » field in the file
    - Placer forwards this field to the solution file

Lessons learnt

- Several versions of Placer
  - Problem
    - Input language of Placer has evolved over time
    - Json parsers are not always user friendly
      - Non-mentioned fields are not considered as errors, just empty lists
      - Unexpected fields are not an error either
  - Solution
    - Add version number to the file format of Placer
    - Check it when reading the files

Lessons learnt

- Non-satisfiable models
  - Problem
    - Input problem might be unsatisfiable because of a conjunction of constraints
    - Ex: specified deadline is too tight
    - Ex: two outgoing transmissions in ASAP mode and only one bus available
  - Solutions
    - Trigger propagation every time a constraint is added, and report failure mentioning the origin of the latest constraint
    - Analyse the input model for some errors, and report them before start up the solver

Lessons learnt

When you ask for a constant/config parameter

- Or you have hidden it somewhere without telling anyone

… it means that you did not manage to get rid of it

Lessons learnt

All solvers are based on some black magics

You need to have some insight of the inner guts
to use it properly

No matters the level of documentation of the API