Innovation and Evolution of Hepsycode Framework: an Extended Methodology for HW/SW Co-Design of Mixed-Criticality and Real-Time Embedded Systems

Authors:

Vittoriano Muttillo, Giacomo Valente, Luigi Pomante

vittoriano.muttillo@graduate.univaq.it, giacomo.velente@univaq.it, luigi.pomante@univaq.it

University of L'Aquila
Center of Excellence DEWS
Department of Information Engineering, Computer Science and Mathematics (DISIM)

OUTLINE

1. Introduction
2. Hepsycode Methodology
   1. System Description
   2. Metrics Evaluation
   3. Design Space Exploration
3. Megam@rt2
4. AQUAS
5. Hepsycode Ecosystem
6. Hepsycode Reference

Hepsy Code

HW/SW CO-DEsign of HEterogeneous Parallel dedicated SYstems

www.hepsycode.com

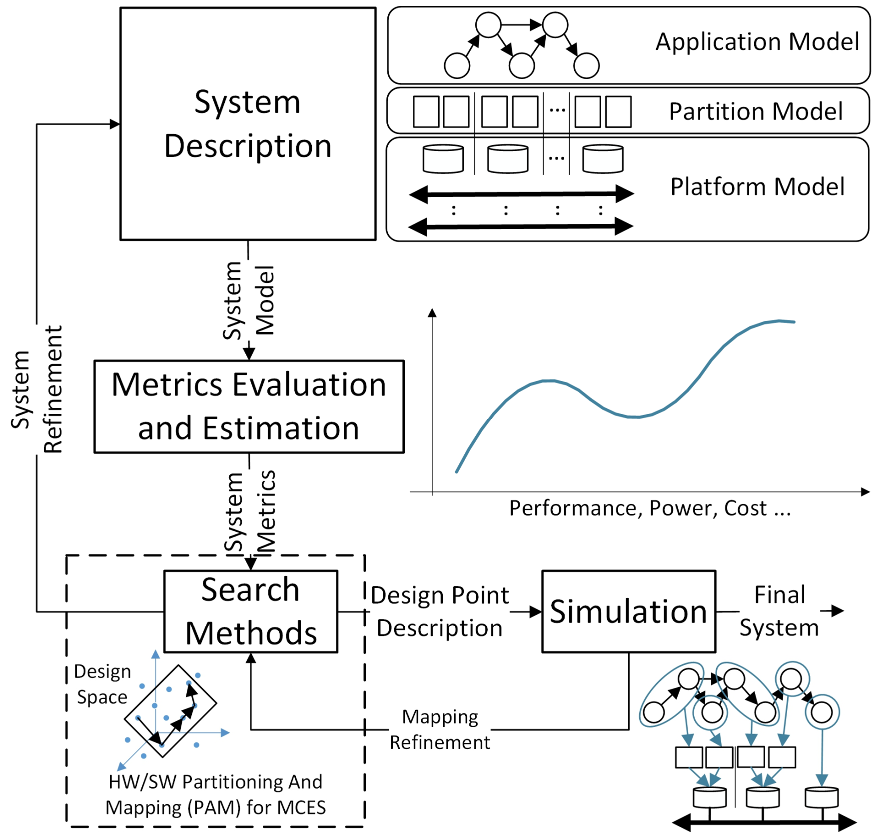
INTRODUCTION

- Early embedded system design activities
  - ✓ Modeling F/NF requirements and validate them before final implementation.
  - ✓ Using system-level models to identify best HW/SW resources allocation by simulating system behavior.
  - ✓ Reduce costs and overall complexity of systems development using proper SW tools.

- Development of a framework for **modeling**, **analysis** and **validation** of mixed-criticality embedded systems, through the use of software tools for "***Model-Based ESL HW/SW Co-design***“.
  - ✓ Electronic System-Level HW/SW Co-Design methodology, and related tools, to design “Heterogeneous Parallel Embedded Systems” (e.g. multi-core systems, multi-processor systems, network-on-chip) for Mixed-Criticality applications has been proposed.
  - ✓ Starting (at least) from the ***System Behavior Specification***, ***Timing*** and ***Mixed-Criticality*** constraints, the proposed approach aims to suggest the ***HW/SW partitioning***, the ***Architecture*** and the ***Mapping*** of the partitioned entities onto the HW components by means of the Design Space Exploration step considering ***Hypervisor-based SW Partitions***.
  - ✓ Drive the DSE to avoid having processes with different criticality levels allocated on the same (shared) partition/processor/core, considering different objective and constraints (timing, power/energy, concurrency…)

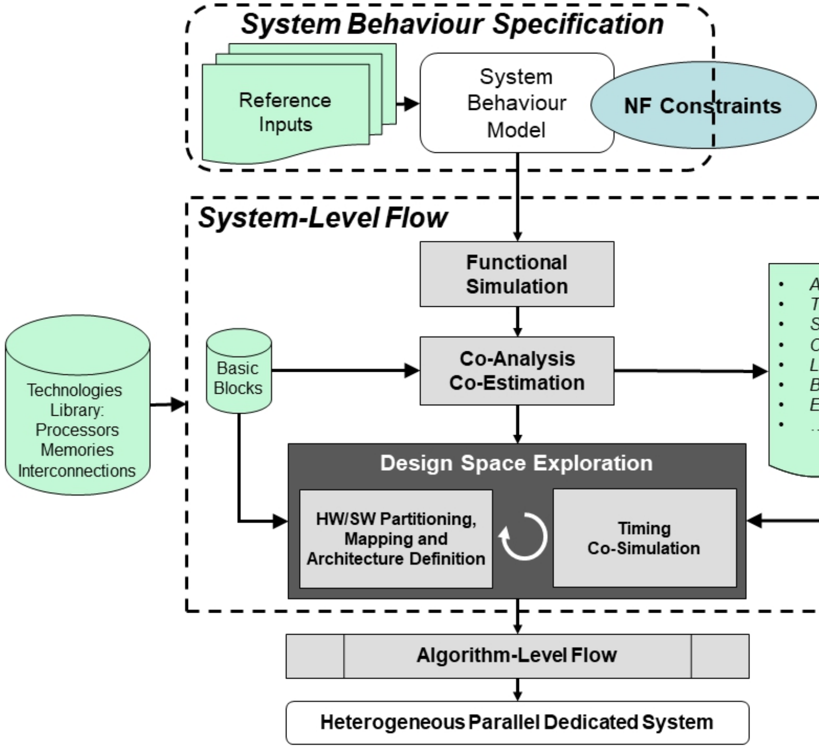
UNIVERSITÀ
DEGLI STUDI
DE L’AQUILA

HEPSYCODE METHODOLOGY

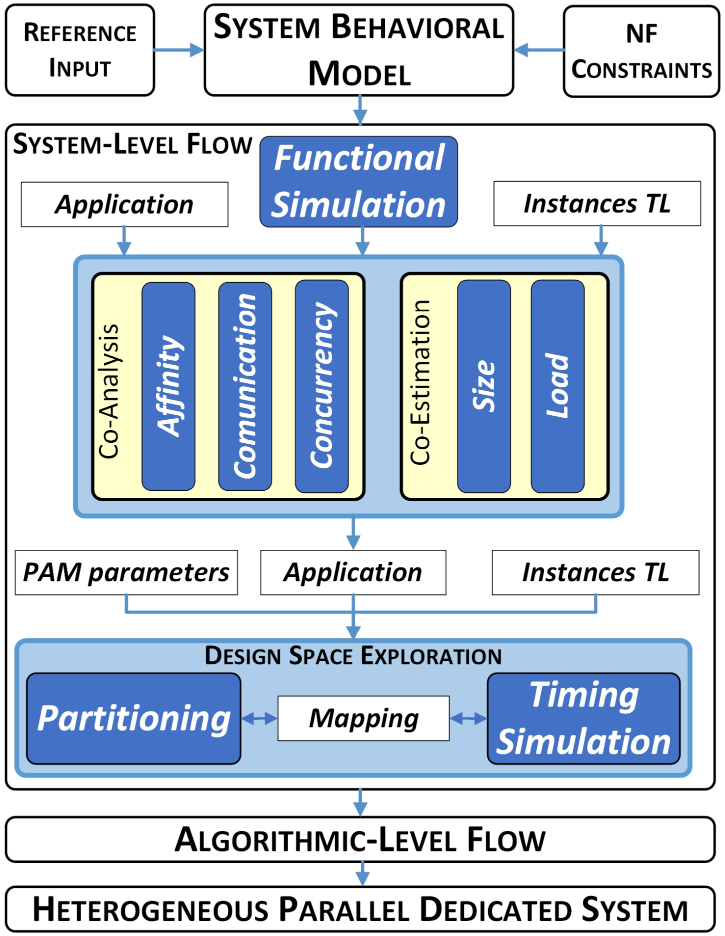


UNIVERSITÀ DEGLI STUDI DE L'AQUILA

HEPSYCODE DESIGN FLOW

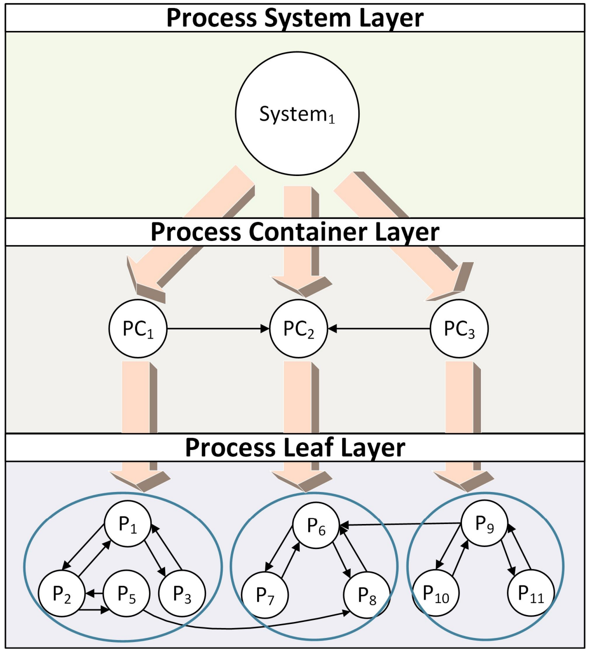


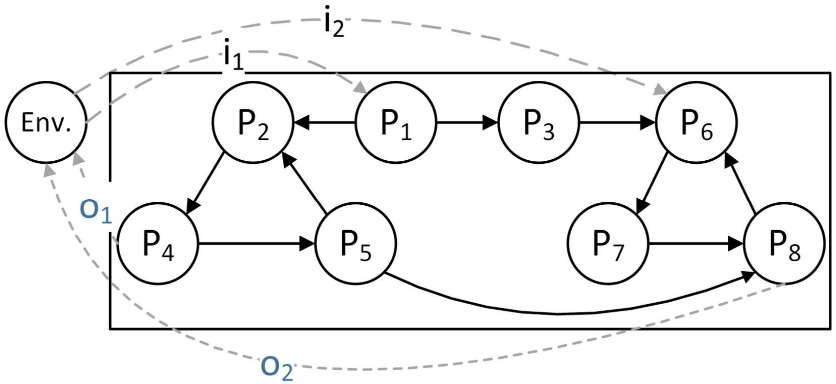
HEPSYCODE FRAMEWORK



UNIVERSITÀ DEGLI STUDI DE L'AQUILA

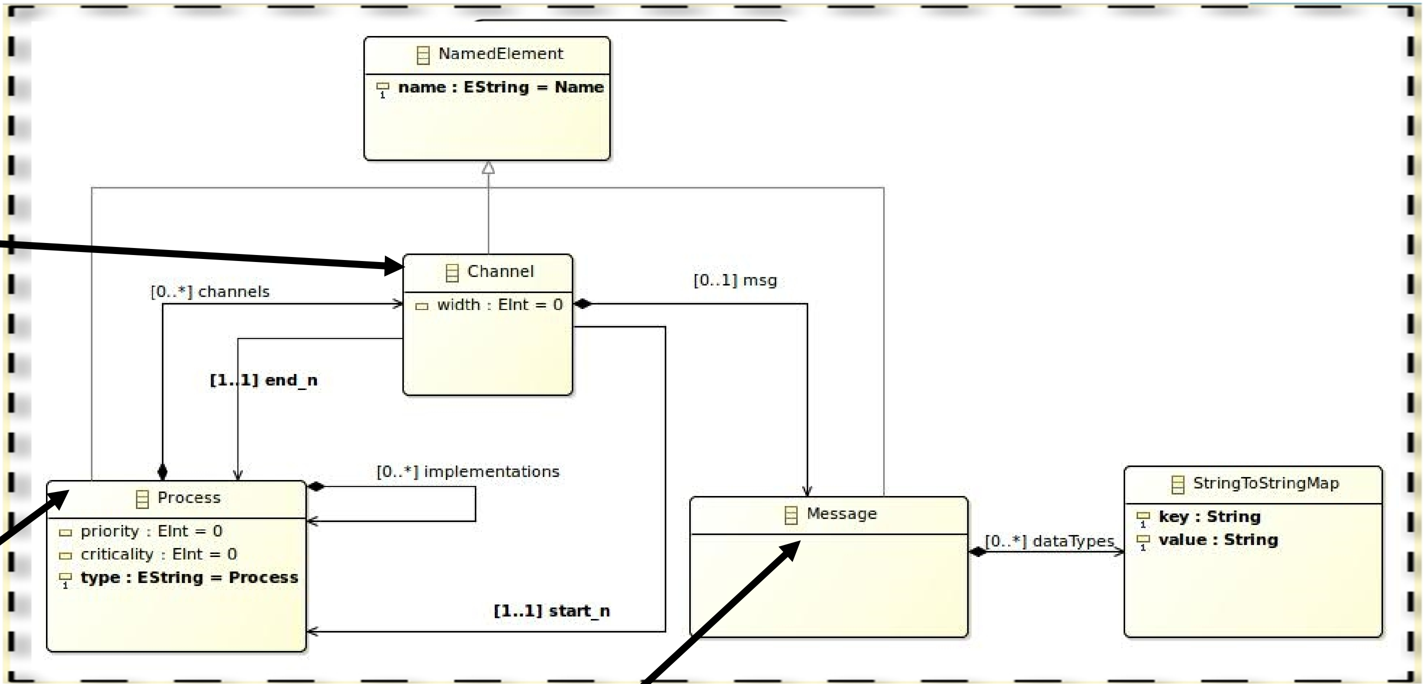
SYSTEM DESCRIPTION





- *System-Level Specification Languages that allows to realize the Hepsycode Model of Computation as a Process Network connected via synchronous channels. Our reference languages is the SystemC, a C++ class library able to capture and define system specification*

HEPSYCODE METAMODEL



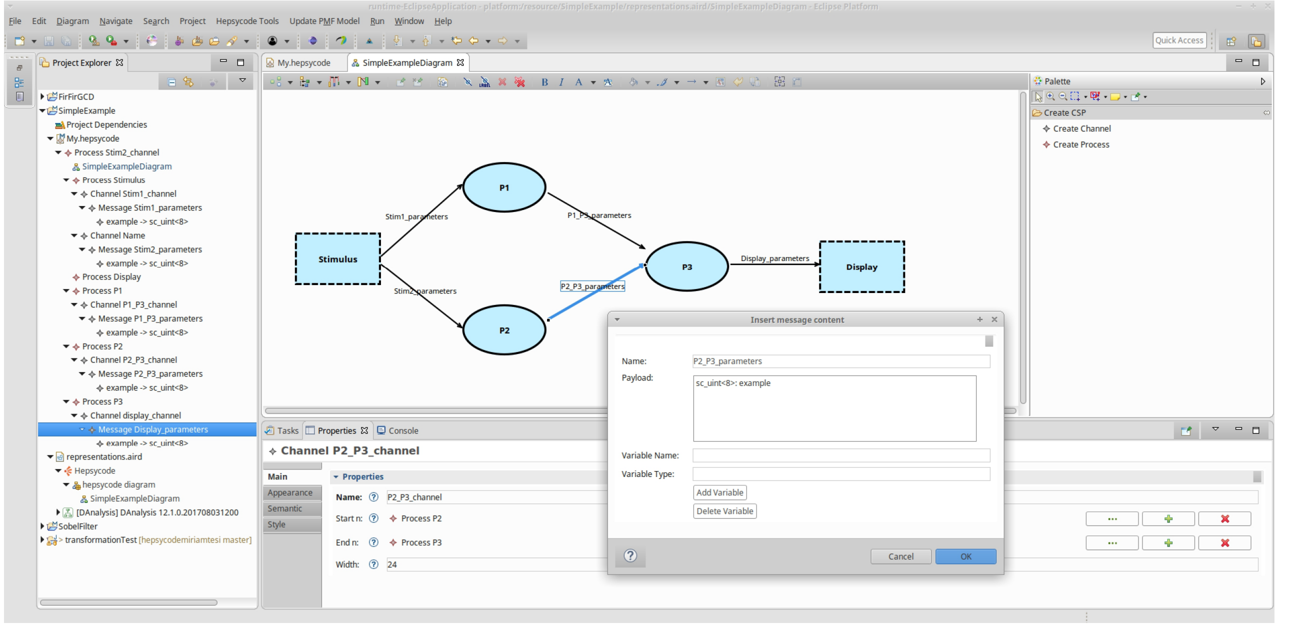
Channels (CSP Channels): unidirectional, rendezvous and blocking point-to-point channel

Process (CSP process): a functional behavior implementation, basic part of a CSP behavior model. A process is formed by an "init" and a body where the behavior can be modelled by a set of statement C or by a Finite-State Machine where the transition between state allow process to exchange data using CSP channel. Each process has a priority and criticality attributes

Messages: Data exchanged by process, with a specific length depending on data types of content.

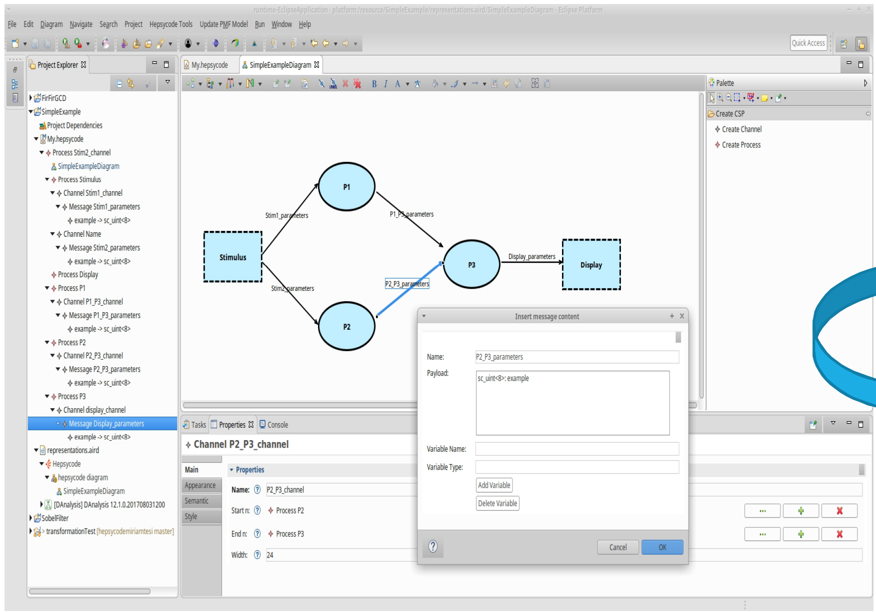
UNIVERSITÀ DEGLI STUDI DE L'AQUILA

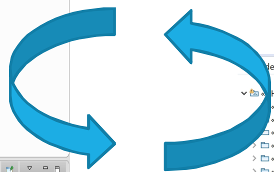
HEPSYCODE GUI

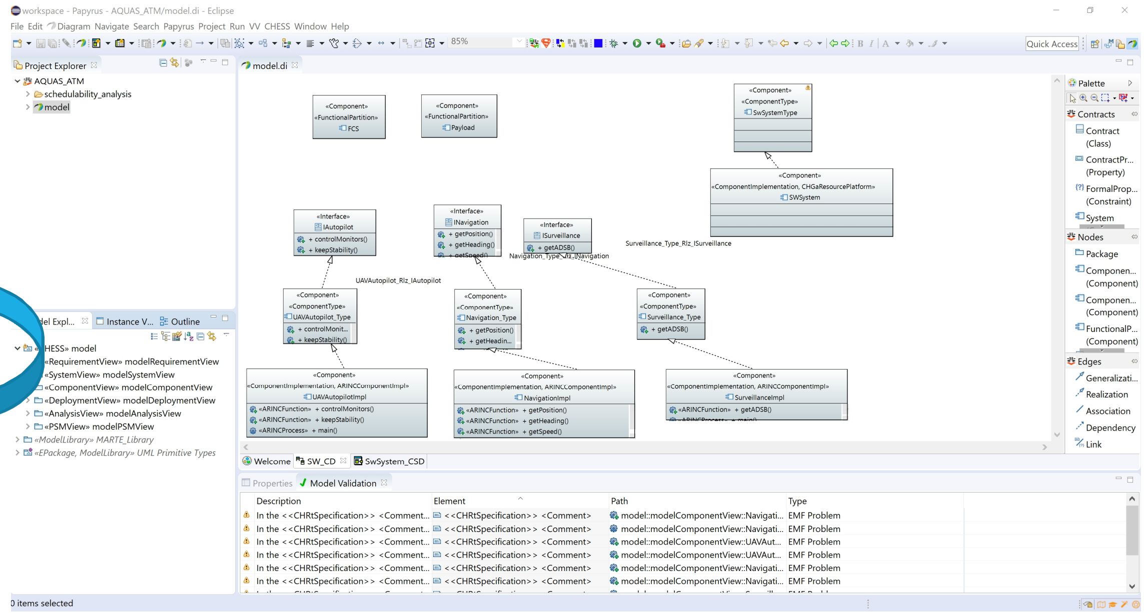


UNIVERSITÀ DEGLI STUDI DE L'AQUILA

HEPSYCODE MODEL TRANSFORMATION



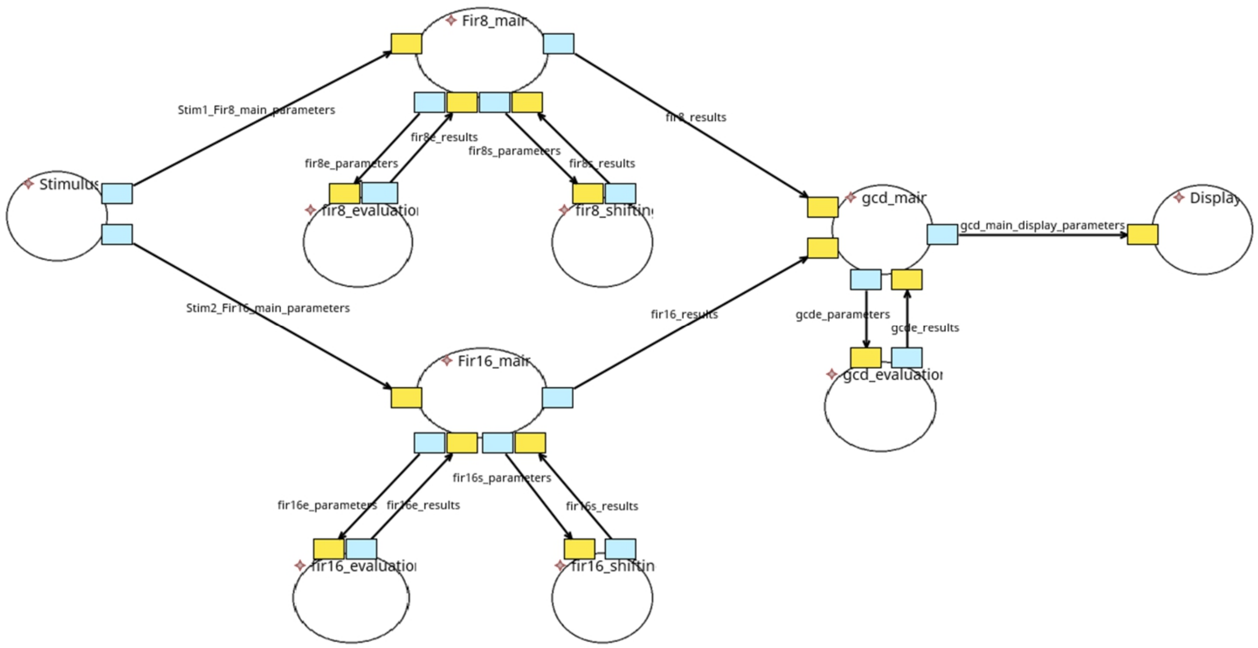




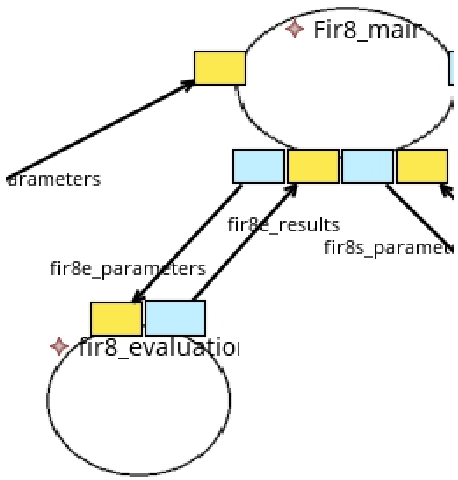
CHESS

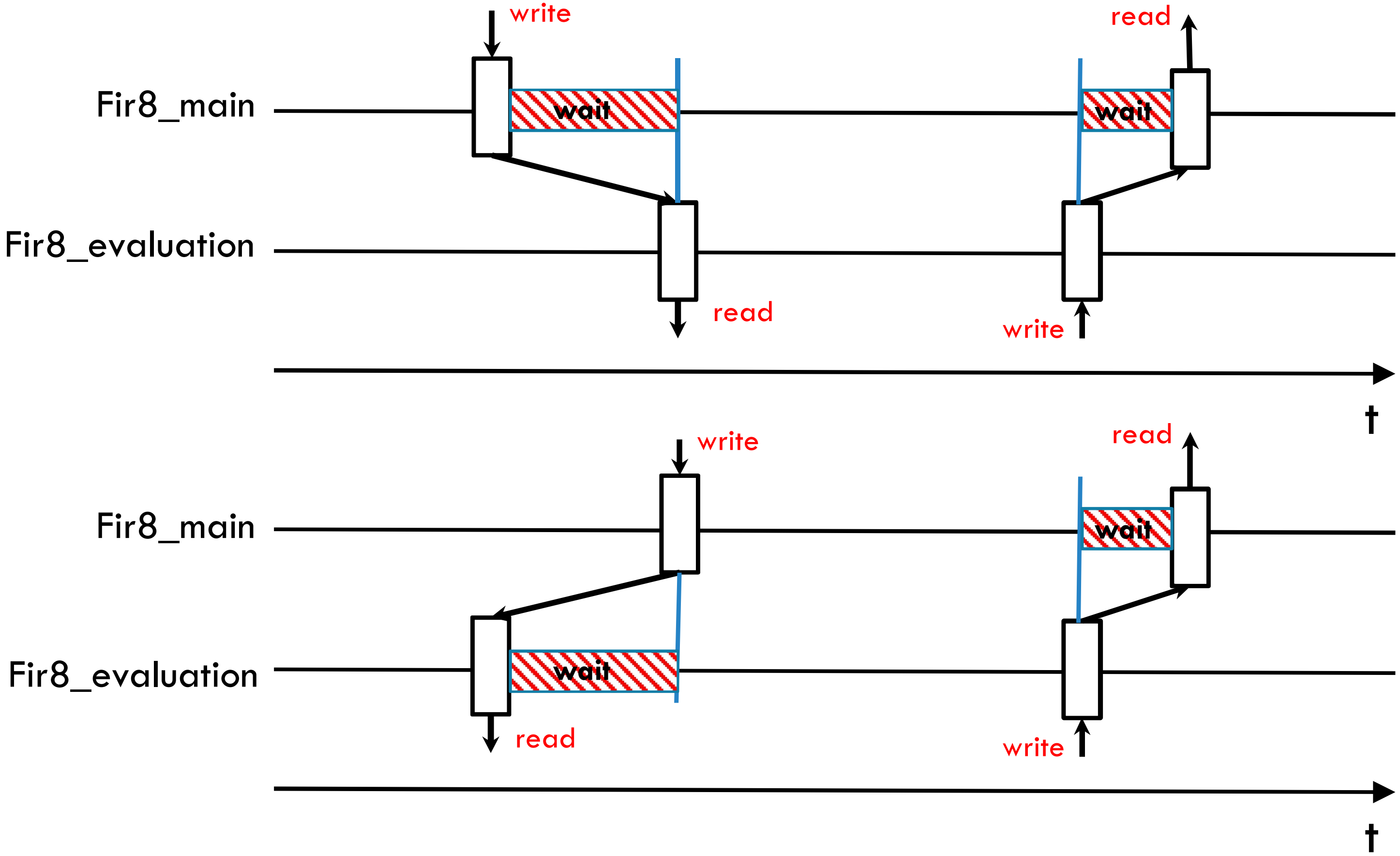
UNIVERSITÀ DEGLI STUDI DE L'AQUILA

HEPSYCODE EXAMPLE - FIRFIRGCD

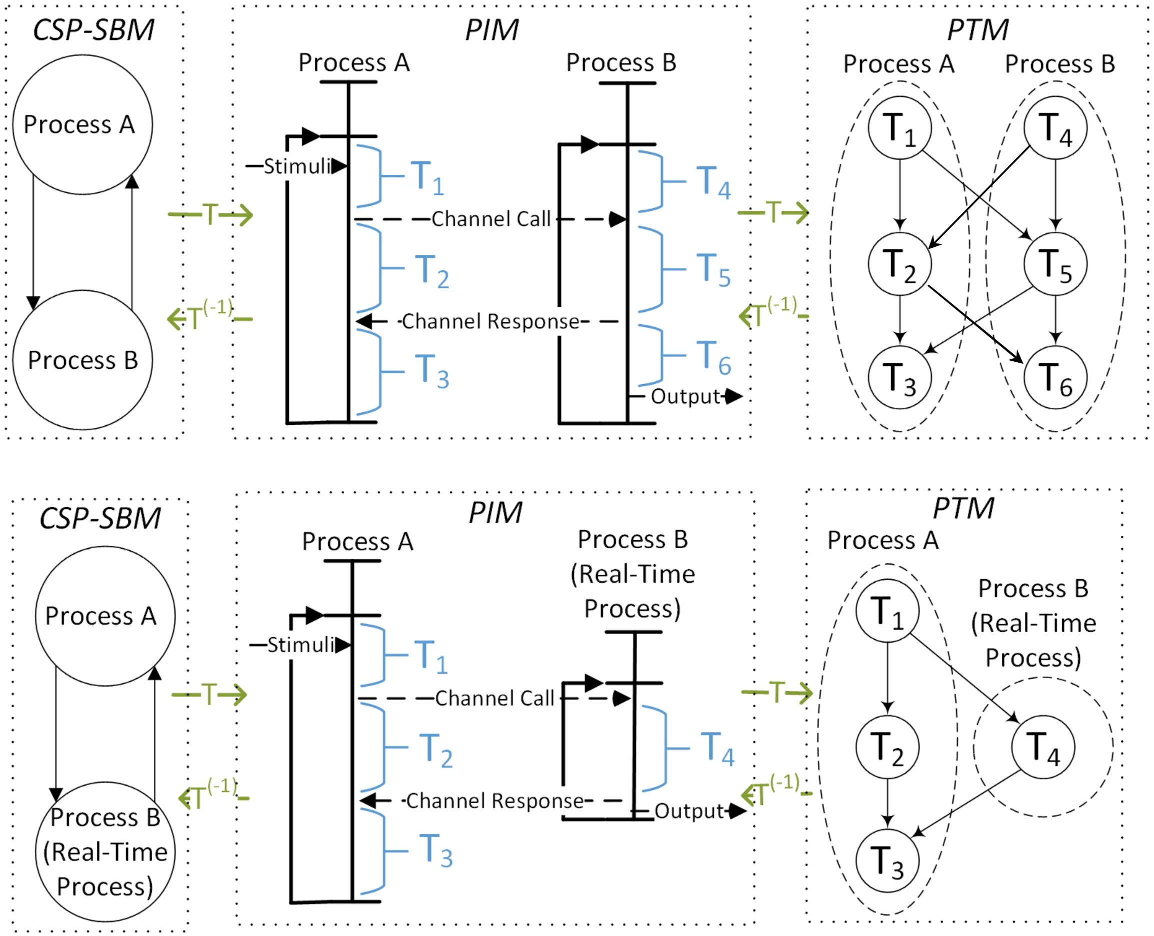


HEPSYCODE CSP

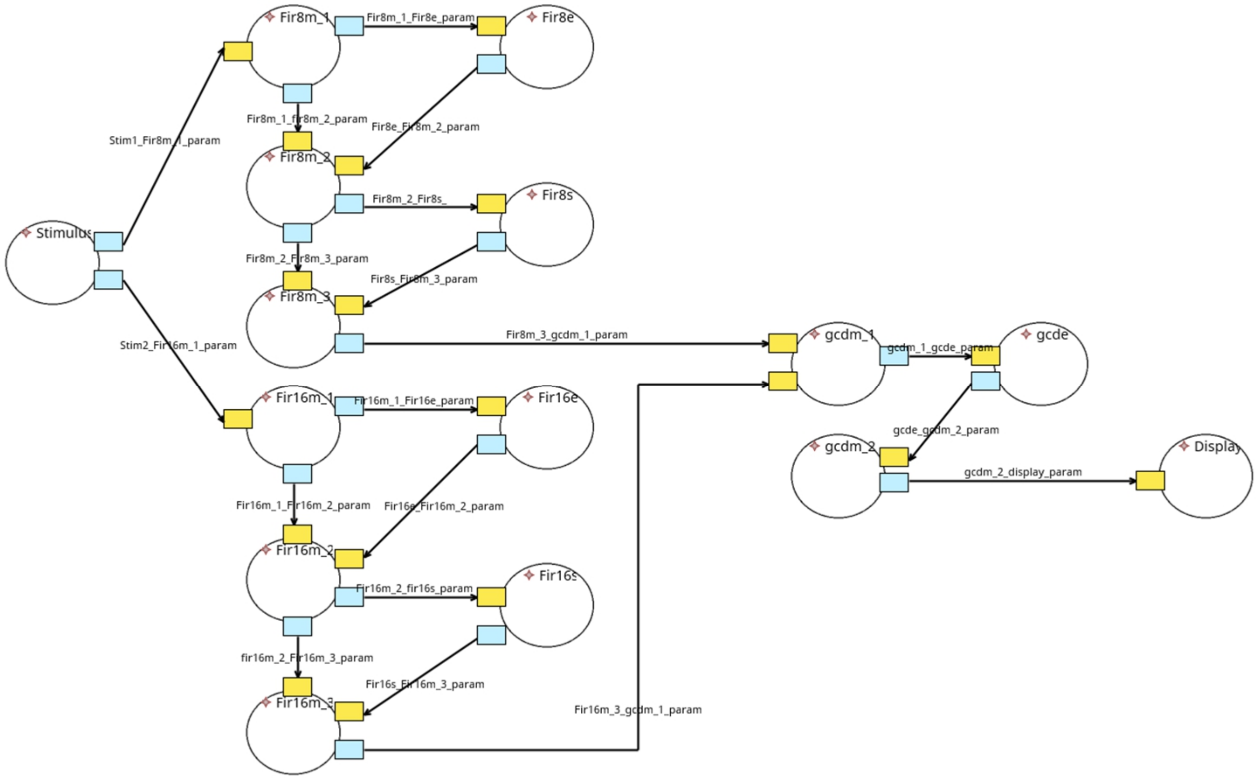




HEPSYCODE REAL-TIME



HEPSYCODE EXAMPLE – FIRFIRGCD-RT



HEPSYCODE NF-CONSTRAINT

- *Non–Functional Constraints*
  - Timing Constraints (TC)
    - **Time-To-Completion Constraint (TTC)**
  - Real-Time Constraints (RTC)
    - **Time-To-Reaction Constraint (TTR)**
    - **Soft and Hard-Real-time Constraints (S/HRT)**
  - Mixed-Criticality Constraints (MCC)
    - **Constraint in the DSE cost function**
    - **Schedulability analysis**
  - Architectural Constraints
    - **Target Form Factor (TFF)**
      - On-chip: ASIC, FPGA, SO(P)C
      - On-Board: SOB (PCB)
    - **Target Template Architecture (TTA)** (related to type of available Basic Blocks BB)
  - Scheduling Directives (SD) - Available scheduling policies for SW processors:
    - **First-Come First-Served (FCFS),** FCFS(no overhead), FCFS (Time Stretching)
    - **Fixed Priority (FP)**
    - **Hypervisor (HVP - WIP)**

UNIVERSITÀ DEGLI STUDI DE L'AQUILA

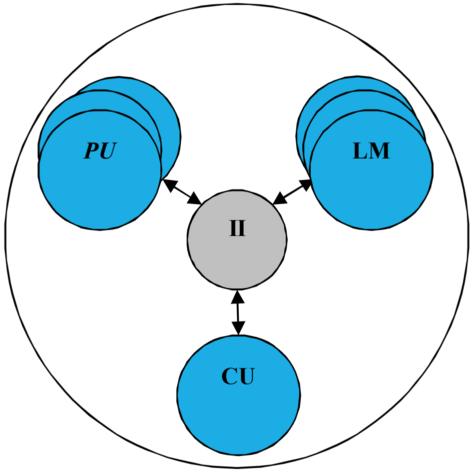
HEPSYCODE TARGET ARCHITECTURE

- The target HW architectures are composed of different basic HW components. This components are collected into a **Technologies Library** (*TL*). TL can be considered a generic "database" that provides the characterization of all the available technologies used in industry and academic world.

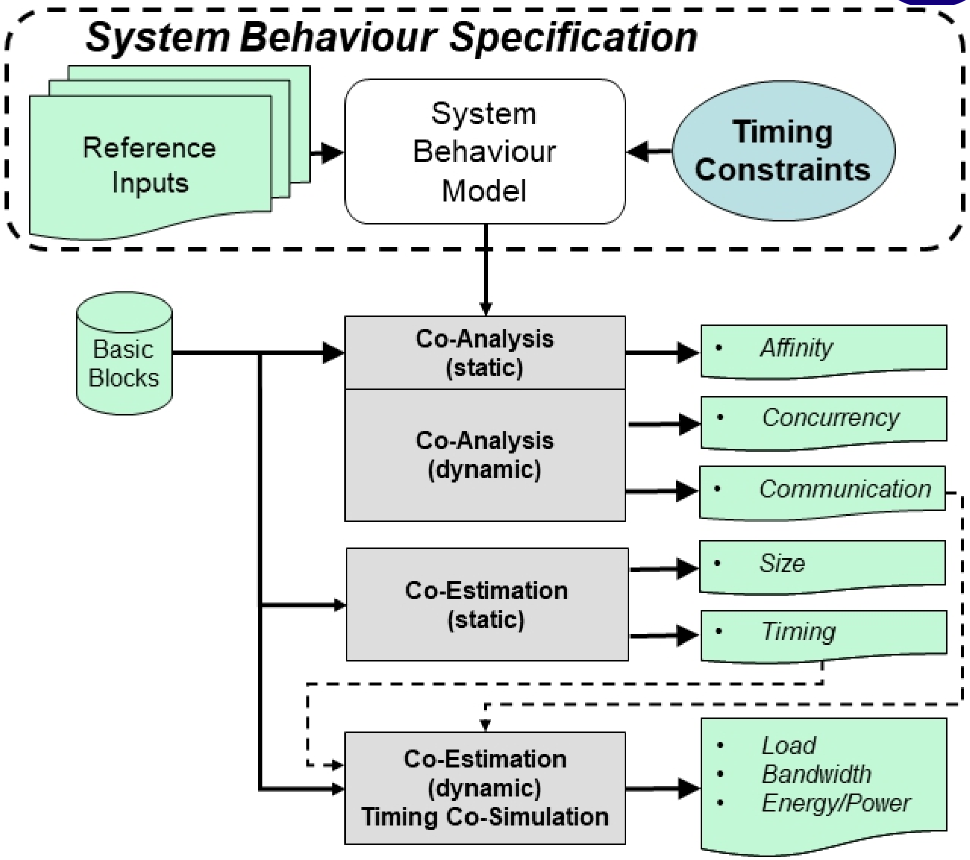
- $TL = \{PU, MU, EIL\}$, where $PU = \{pu_1, pu_2, .. , pu_p\}$ is a set of *Processing Units*, $MU = \{mu_1, mu_2, .. , mu_m\}$ is a set of *Memory Units* and $EIL = \{il_1, il_2, .. , il_c\}$ is a set of *External Interconnection Links*.

- Blocks built by the designer starting from the TL are called **Basic Blocks** (*BB*)

- They are the basic components available during DSE step to automatically define the HW architecture. A generic BB is composed of a set of Processing Units (*PU*), a set of Memories Units (*MU*), an Internal Interconnection (*II*) and a Communication Unit (CU). and a *Communication Unit* (*CU*). *CU* represents the set of *EIL* that can be managed by a BB.



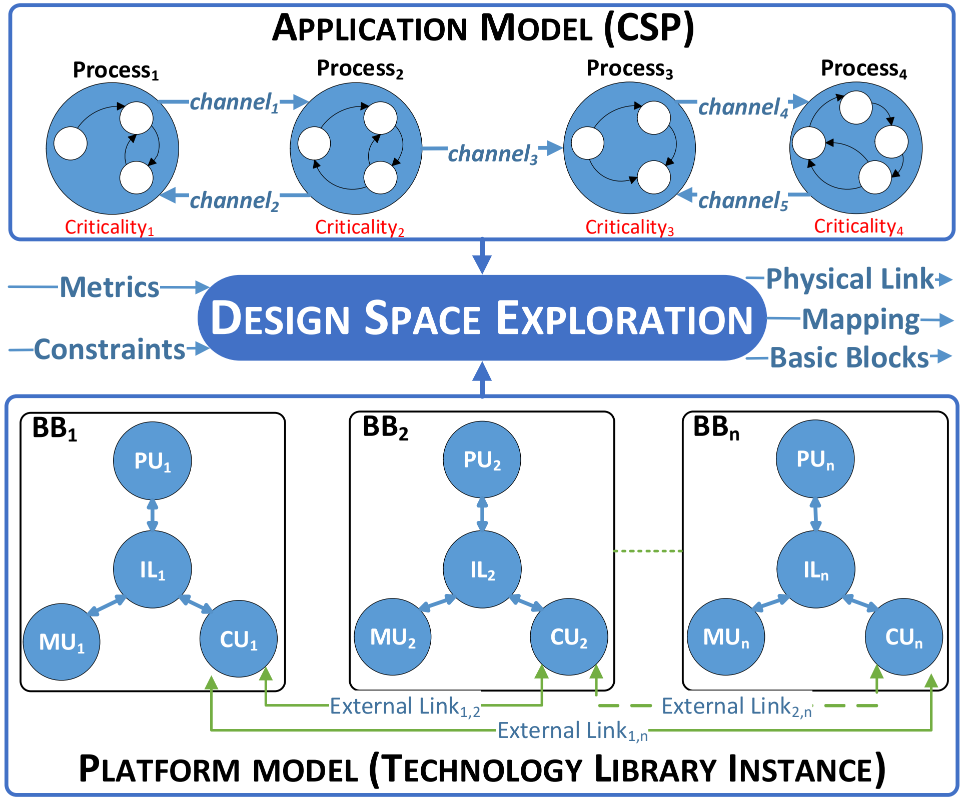
METRICS EVALUATION & ESTIMATION



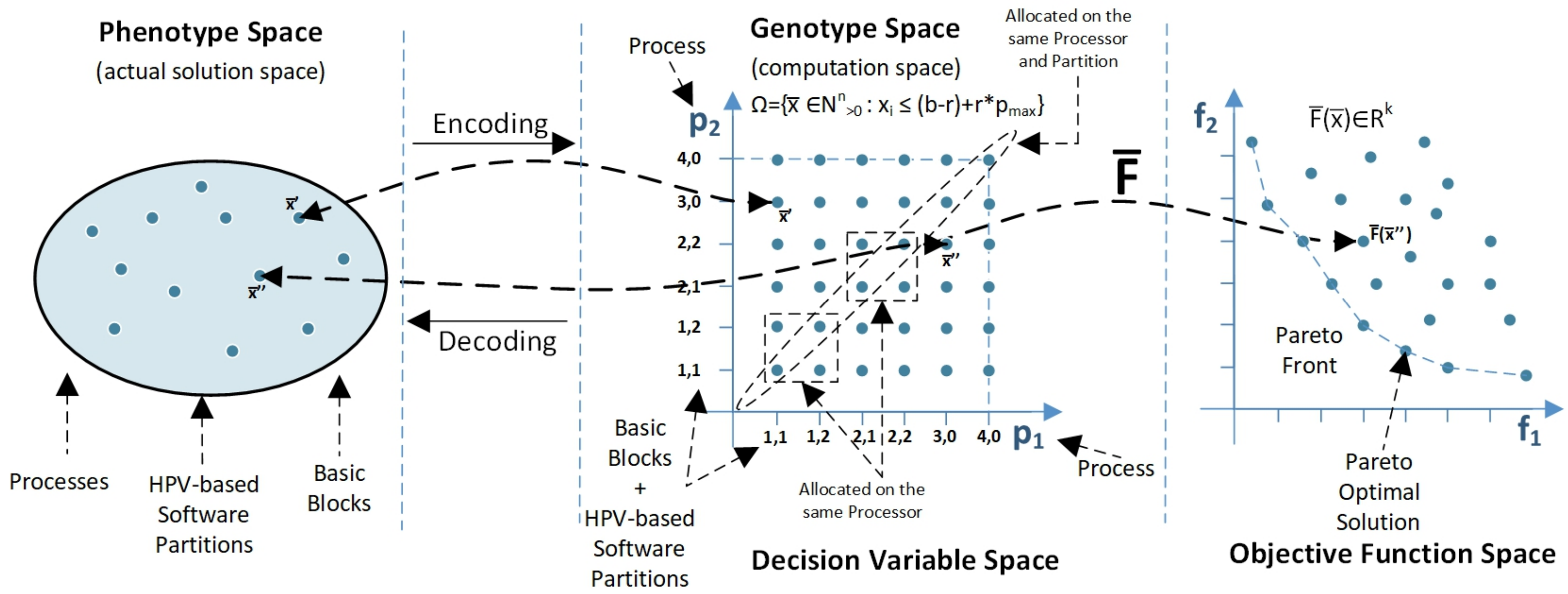
HEPSY CODE

UNIVERSITÀ DEGLI STUDI DE L'AQUILA

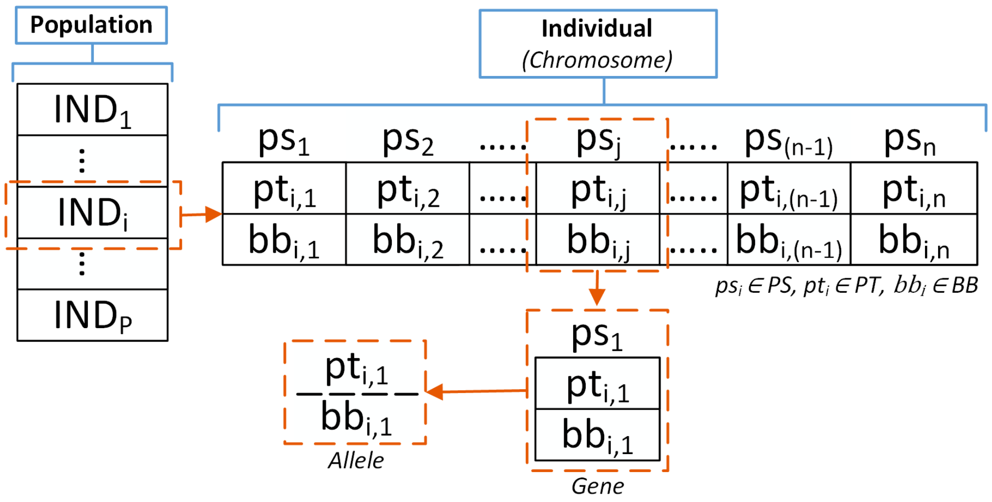
HEPSYCODE DSE

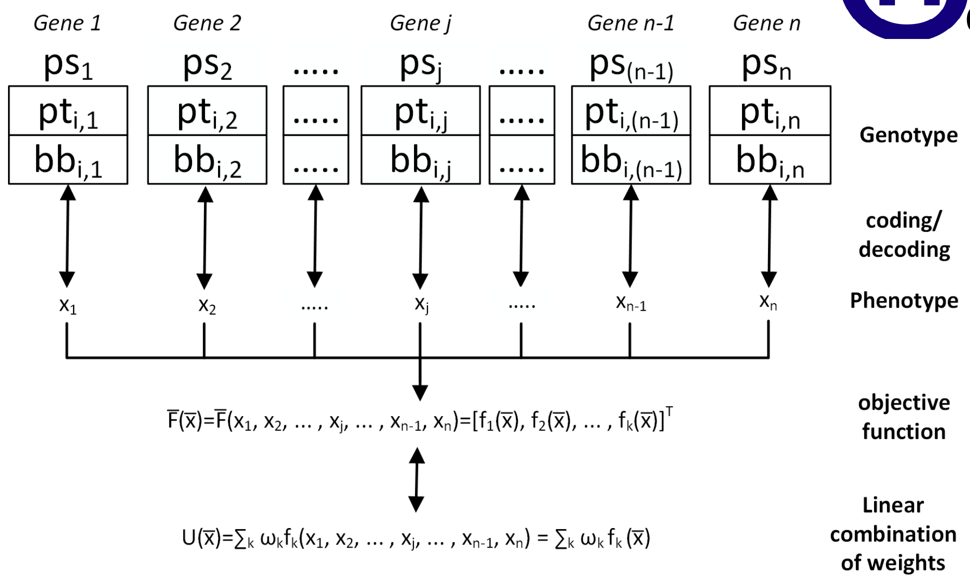


UNIVERSITÀ DEGLI STUDI DE L'AQUILA



HEPSYCODE DSE





HEPSYCODE DSE INDIVIDUAL

Definition 11.1. (*Linearization of Multi-objective Design Space Exploration Optimization Problem*).

$$\begin{aligned} \min_{\bar{x}} \quad & U(\bar{x}) = \sum_k \omega_k \cdot f_k(\bar{x}) = \sum_k \omega_k \cdot f_k(x_1, x_2, \ldots, x_n) \\ \text{subject to} \quad & \bar{x} \in \Omega = \{\bar{x} \in \mathbb{N}^n_{>0} : \ 0 < x_i \leq (b-r) + r * p_{max}\} \end{aligned} \tag{42}$$

$U(\bar{x})$ is the utility function evaluated at each iteration of the GA for each individual $\bar{x} \in \Omega$. f_k represents the value of the objective function (or metric) k for each individual $\bar{x}$, while ω_k is the weight associated to each objective function or metric.

Definition 9.1. (*Multi-Objective Design Space Exploration Optimization Problem*).

$$\begin{aligned} \min_{\bar{x}} \quad & \bar{F}(\bar{x}) = [f_1(\bar{x}), f_2(\bar{x}), \ldots, f_k(\bar{x})]^\intercal \\ \text{subject to} \quad & \bar{x} \in \Omega = \{\bar{x} \in \mathbb{N}^n_{>0} : x_i \leq (b-r) + r * p_{max}\} \end{aligned} \tag{39}$$

where $\bar{x} = \{x_1, \ldots, x_n\}$ is an n-dimensional decision variable vector representing processes in the solution space Ω (which refers to a feasible search space, feasible set of decision vectors), $\mathbb{R}^k$ refers to the objective space. The value b is the total number of BBs, r is the number of BBs that have processor type egual to GPP, and p_{max} is the maximum number of HPV-based SW Partition instances for each GPP processor. $\bar{F}(\bar{x}) = [f_1(\bar{x}), f_2(\bar{x}), \ldots, f_k(\bar{x})]^\intercal \rightarrow \mathbb{R}^k$ consists of k $\geq$ 2 real-valued objective functions.

HEPSYCODE DSE

UNIVERSITÀ
DEGLI STUDI
DE L'AQUILA

HEPSYCODE DSE - AFFINITY

11.1 Affinity Index

The ***Affinity Index*** is a metric based on two matrixes that consider each individual $\bar{x}$. The first matrix is the *Affinity Matrix* $A = \{\ [a_1,\ a_2,\ ..\ ,a_j,\ ..\ ,\ a_n]^\intercal\ :\ a_j = [a_{j,1}(GPP),\ a_{j,2}(DSP),\ a_{j,3}(SPP)]^\intercal\} \in \mathbb{R}^{n\times 3}$. a_j is an array of a triples in the interval [0,1] that provides a quantification of the matching among the structural and functional features of the functionality implemented by a process ps_j and the architectural features of each one of the following processor types: *GPP, DSP, SPP*. Higher the *Affinity Matrix* element value, more suitable the corresponding processor type. The second matrix is the *Affinity Selection Matrix* $ASM(\bar{x}) = \{\ [asm_1(\bar{x}),\ asm_2(\bar{x}),\ ..\ ,asm_j(\bar{x}),\ ..\ ,\ asm_n(\bar{x})]\ \} \in \mathbb{R}^{3\times n}$, where the array $asm_j(\bar{x}) = [asm_{j,1}(GPP),\ asm_{j,2}(DSP),\ asm_{j,3}(SPP)]^\intercal \in \mathbb{R}^3$ assume the value 0 or 1, respectively, if the process ps_j is allocated or not to the associated type of processor. So, it is possible to evaluate the *Total Degree of Affinity (TDA) Index* as:

$$f_{TDA}(\bar{x}) = 1 - \frac{\mathrm{tr}[A \cdot ASM(\bar{x})]}{n} = 1 - \frac{\sum_{j=1}^{n}\sum_{k=1}^{3} a_{j,k} \cdot asm_{k,j}(\bar{x})}{n} \tag{43}$$

It is worth noting that the *Affinity Matrix* A is independent from the specific iteration and individual $\bar{x}$, since it is a unique and fixed matrix evaluated in the Co-Estimation Design-Flow step.

UNIVERSITÀ DEGLI STUDI DE L'AQUILA

11.2 Processes Concurrency Index

The ***Processes Concurrency Index*** is based on a *Concurrency Matrix*, calculated in the Co-Estimation step:

$$CON = \begin{bmatrix} con_{1,1} & con_{1,2} & \cdots & con_{1,n} \\ con_{2,1} & con_{2,2} & \cdots & con_{2,n} \\ \vdots & \vdots & \vdots & \vdots \\ con_{n,1} & con_{n,2} & \cdots & con_{n,n} \end{bmatrix} \quad (44)$$

CON provides information about how much processes pairs can be potentially concurrently "working", where $CON = \{\ con_{i,j} \neq 0\ :\ ps_i \wedge ps_j$ can be potentially executed concurrently$\} \in \mathbb{R}^{n\times n}$. Starting from each individual $\bar{x}$, it is possible to define a *Processes Concurrency Selection Matrix*, $S^{con}(\bar{x}) \in \mathbb{R}^{n\times n}$, as listed below:

$$S^{con}(\bar{x}) = \begin{cases} s^{con}_{i,j}(\bar{x}) = 1, & if\ ps_i \in pu_x \wedge ps_j \in pu_y \wedge pu_x \neq pu_y \\ s^{con}_{i,j}(\bar{x}) = 0, & \text{otherwise} \end{cases} \in \mathbb{R}^{n\times n} \quad (45)$$

So, for each individual $\bar{x}$, the *Exploited Inter Cluster Parallelism* matrix, $EIPC(\bar{x}) \in \mathbb{R}^{n\times n}$, indicates how much an individual can exploit the potential concurrency:

$$EIPC(\bar{x}) = CON \cdot S^{con}(\bar{x}) \quad (46)$$

Starting from $EICP(\bar{x})$ matrix function, the *Exploited Parallelism (EP)* index is equal to:

$$f_{EP}(\bar{x}) = \frac{\sum_{j=1}^{n} \sum_{k=1}^{n} eipc_{j,k}(\bar{x})}{max_{EP}} \qquad max_{EP} = \sum_{j=1}^{n} \sum_{k=1}^{n} con_{j,k} \quad (47)$$

HEPSYCODE DSE – PROCESS CONCURRENCY

HepsyCode

UNIVERSITÀ DEGLI STUDI DE L'AQUILA

HEPSYCODE DSE – PROCESS COMMUNICATION

11.3 Processes Communication Index

The ***Processes Communication Index*** is based on the *Communication Matrix*, calculated in the Co-Estimation step:

$$CM = \begin{bmatrix} cm_{1,1} & cm_{1,2} & \cdots & cm_{1,n} \\ cm_{2,1} & cm_{2,2} & \cdots & cm_{2,n} \\ \vdots & \vdots & \vdots & \vdots \\ cm_{n,1} & cm_{n,2} & \cdots & cm_{n,n} \end{bmatrix} \tag{48}$$

CM is expressed by the number of bits sent/received over each channel. So, for each individual $\bar{x}$, it is possible to define a *Processes Communication Selection Matrix*, $S^{cm}(\bar{x}) \in \mathbb{R}^{n \times n}$, as listed below:

$$S^{cm}(\bar{x}) = \begin{cases} s^{cm}_{i,j}(\bar{x}) = 1, & if\ ps_i \in pu_x \wedge ps_j \in pu_y \wedge pu_x \neq pu_y \\ s^{cm}_{i,j}(\bar{x}) = 0.5, & if\ ps_i \in pt_x \wedge ps_j \in pt_y \wedge pt_x \neq pt_y \\ s^{cm}_{i,j}(\bar{x}) = 0, & \text{otherwise} \end{cases} \in \mathbb{R}^{n \times n} \tag{49}$$

So, for each individual $\bar{x}$, the *Inter Cluster Communication Cost*, $ICCC(\bar{x}) \in \mathbb{R}^{n \times n}$, represents the cost associated to process communication if processes are allocated on different processors:

$$ICCC(\bar{x}) = CM \cdot S^{cm}(\bar{x}) \tag{50}$$

Starting from ICCC matrix, the *Normalized Total Communication Cost* index is:

$$\begin{aligned} f_{NTCC}(\bar{x}) &= \frac{\sum_{j=1}^{n} \sum_{k=1}^{n} iccc_{j,k}(\bar{x})}{max_{NTCC}} \\ max_{NTCC} &= \sum_{j=1}^{n} \sum_{k=1}^{n} cm_{j,k} \end{aligned} \tag{51}$$

UNIVERSITÀ DEGLI STUDI DE L'AQUILA

11.4 Load Index

The ***Load Index*** is based on the *Load Matrix* $L = \{\ [l_1,\ l_2,\ ..\ ,l_j,\ ..\ ,\ l_n]\ :\ l_j = [l_{1,j},\ l_{2,j},\ ..\ ,\ l_{k,j},\ ..\ ,l_{s,j}]^\intercal\} \in \mathbb{R}^{s\times n}$, where each matrix element represents the load that each process ps_j would impose to each s *non-SPP* processor pu_k (used in at least one BB, s = #PU - #HW_PU) to satisfy TTC. L is estimated by allocating all the n processes to a single-instance of each software processor pu_k and performing some simulations into the Co-Estimation step. Three parameters have to be computed: FRT_k *(Free Running Time)*, i.e. the total application simulated time on processor pu_k; $t_{k,j}$, the simulated time for each process ps_j on processor pu_k; $N_{k,j}$, the number of executions of each process ps_j on processor pu_k. Starting from these estimated parameters, it is possible to define the *Free Running Load Matrix* FRL:

$$FRL = \begin{bmatrix} frl_{1,1} & frl_{1,2} & \cdots & frl_{1,j} & \cdots & frl_{1,n} \\ frl_{2,1} & frl_{2,2} & \cdots & frl_{2,j} & \cdots & frl_{2,n} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ frl_{k,1} & frl_{k,2} & \cdots & frl_{k,j} & \cdots & frl_{k,n} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ frl_{s,1} & frl_{s,2} & \cdots & frl_{s,j} & \cdots & frl_{s,n} \end{bmatrix} \tag{52}$$

$$where\ frl_{k,j} = \frac{(t_{k,j} \cdot N_{k,j})}{FRT_k}\ \forall k = 1..s,\ j = 1..n$$

where $FRT_k/N_{k,j}$ is the average period of each processes ps_j on processor pu_k. By imposing that the simulated time shall be equal to TTC, it is possible to evaluate the Load $l_{k,j}$ that processes ps_j would impose to the SW processor pu_k to satisfy TTC itself. In fact, setting:

$$TTC = x_k \cdot (FRT_k + OH_k)\ \ with\ 0 \le x_k \le 1 \tag{53}$$

while OH_k is the overhead introduced by a given scheduling policy. The value of estimated Load $l_{k,j}$ that the system imposes to processor pu_k to satisfy TTC is equal to:

$$\begin{aligned} l_{k,j} &= \frac{(t_{k,j} \cdot N_{k,j})}{TTC} = \frac{(t_{k,j} \cdot N_{k,j})}{FRT_k} \cdot \frac{FRT_k}{TTC} = \\ &= frl_{k,j} \cdot \frac{FRT_k}{TTC} = \frac{frl_{k,j}}{x_k} \cdot \frac{FRT_k}{(FRT_k + OH_k)} \\ &\forall k = 1..s,\ j = 1..n \end{aligned} \tag{54}$$

From a DSE perspective, by considering the sum of the Load $l_{k,j}$ of all the processes allocated to a GPP/ASP pu_k processor, it is possible to check if the total imposed Load is acceptable. Considering [23], the least load upper bound is on the order of $\simeq 70\%$. In this work, the introduction of HPV-based SW partitions add a second level scheduling, introducing hierarchical scheduling issues, so the new load upper bound became $\simeq 36\%$ [28]. So it is possible to define the *Load Index* as:

$$\begin{aligned} f_L(\bar{x}) &= 1 - \mathrm{tr}\,[L \cdot ALL^L(\bar{x})] = 1 - \frac{\sum_{k=1}^{s}\sum_{j=1}^{n} l_{k,j} \cdot all^L_{j,k}(\bar{x})}{s} \\ L &= \begin{cases} \frac{frl_{k,j}}{x_k} \cdot \frac{FRT_k}{(FRT_k+OH_k)} & if\ TTC\ \le (FRT_k + OH_k) \\ frl_{k,j} & otherwise \end{cases} \in \mathbb{R}^{s\times n} \\ ALL^L(\bar{x}) &= \begin{cases} all^L_{j,k}(\bar{x}) = 1 & if\ ps_j\ \in\ pu_k \\ all^L_{j,k}(\bar{x}) = 0 & otherwise \end{cases} \in \mathbb{R}^{n\times s} \end{aligned} \tag{55}$$

HEPSYCODE DSE – LOAD

UNIVERSITÀ DEGLI STUDI DE L'AQUILA

11.5 Cost Index

The ***Cost Index*** is a metric related to the monetary cost $C = [c_1, c_2, .., c_k, .., c_b]$ associated to each bb_k considered in the specific $\bar{x}$ (considering PU, MU and CU):

$$
\begin{aligned}
f_C(\bar{x}) &= 1 - C \cdot ALL^C(\bar{x}) = 1 - \frac{\sum_{k=1}^{b} c_k \cdot all_k^C(\bar{x})}{max_C} \\
ALL^C(\bar{x}) &= \begin{cases} all_k^C(\bar{x}) = 1 & if \ \exists \ ps_j \ : \ ps_j \ \in \ pu_k, \ \forall j = 1..n \\ all_k^C(\bar{x}) = 0 & otherwise \end{cases} \in \mathbb{R}^b \qquad (56) \\
max_C &= b \cdot max(c_k), \ \forall k = 1..b
\end{aligned}
$$

HEPSYCODE DSE – COST

HEPSYCODE DSE – SIZE

11.6 Size Index

The ***Size Index*** is a set of estimations for each statement of each process with respect to each available processor. It is related to number of bytes or area/resources metrics depending on SW or HW implementations. It is possible to define three matrix: $RAM = \{[ram_1, ram_2, .., ram_j, .., ram_n] : ram_j = [ram_{j,1}, ram_{j,2}, .., ram_{j,k}, .., ram_{j,b}]^\intercal\} \in \mathbb{R}^{n\times b}$, where $ram_{j,k}$ is the RAM size value of each process ps_j allocated on SW processor pu_k defined into the BB, $ROM = \{[rom_1, rom_2, .., rom_j, .., rom_n] : rom_j = [rom_{j,1}, rom_{j,2}, .., rom_{j,k}, .., rom_{j,b}]^\intercal\} \in \mathbb{R}^{n\times b}$, where $rom_{j,k}$ is the ROM size value of each process ps_j allocated on SW processor pu_k defined into the BB, $EQG = \{[eqg_1, eqg_2, .., eqg_j, .., eqg_n] : eqg_j = [eqg_{j,1}, eqg_{j,2}, .., eqg_{j,k}, .., eqg_{j,b}]^\intercal\} \in \mathbb{R}^{n\times b}$, where $eqg_{j,k}$ is the equivalent gate value associated to each process ps_j allocated on SW processor pu_k defined into the BB. Starting from this matrix, it is possible to calculate the *Size Index*:

$$f_S(\bar{x}) = f_{SW}(\bar{x}) + f_{HW}(\bar{x}) \tag{57}$$

$$f_{SW}(\bar{x}) = \frac{\mathrm{tr}\left[(RAM+ROM)\cdot ALL^{SW}(\bar{x})\right] - max_{SIZE\_SW}}{max_{SIZE\_SW}} = \frac{\left[\sum_{j=1}^{n}\sum_{k=1}^{b}(ram_{j,k}+rom_{j,k})\cdot all_{k,j}^{SW}(\bar{x})\right] - max_{SIZE\_SW}}{max_{SIZE\_SW}} \tag{58}$$

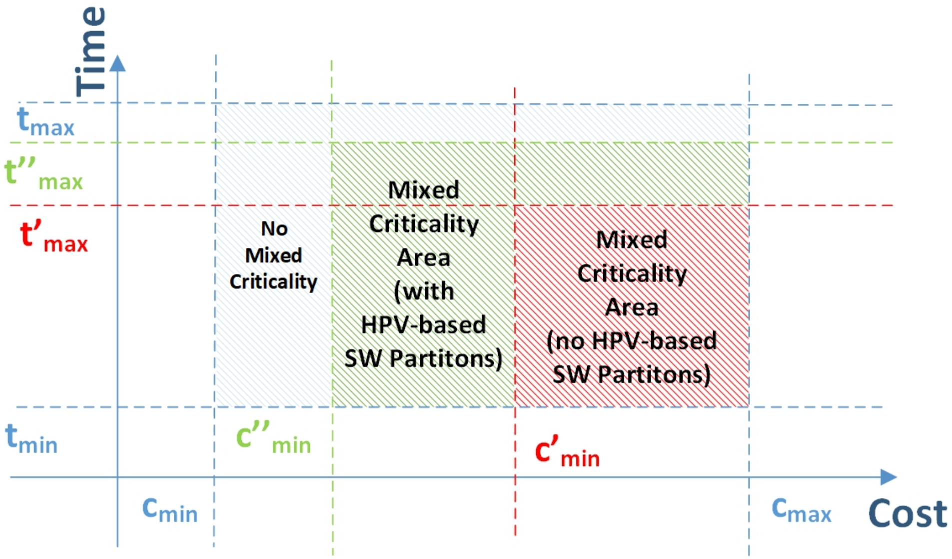
$$ALL^{SW}(\bar{x}) = \begin{cases} all_{k,j}^{SW}(\bar{x}) = 1 & if\ ps_j \in pu_k\ SW\_PU \\ all_{k,j}^{SW}(\bar{x}) = 0 & otherwise \end{cases} \in \mathbb{R}^{b\times n} \tag{59}$$

$$f_{HW}(\bar{x}) = \frac{\mathrm{tr}\left[EQG\cdot ALL^{HW}(\bar{x})\right] - max_{SIZE\_HW}}{max_{SIZE\_HW}} = \frac{\left[\sum_{j=1}^{n}\sum_{k=1}^{b} eqg_{j,k}\cdot all_{k,j}^{HW}(\bar{x})\right] - max_{SIZE\_HW}}{max_{SIZE\_HW}} \tag{60}$$

$$ALL^{HW}(\bar{x}) = \begin{cases} all_{k,j}^{HW}(\bar{x}) = 1 & if\ ps_j \in pu_k\ HW\_PU \\ all_{k,j}^{HW}(\bar{x}) = 0 & otherwise \end{cases} \in \mathbb{R}^{b\times n} \tag{61}$$

max_{SIZE_SW} considered into the BBs depends on OS technologies, in order to reduce the SW memory size available for allocated processes in terms of OS size (and HPV-based SW partition size, where the memory is allocated and reduced respect to the size and memory allocated to each partition and each HPV solution). It is worth nothing that $ALL(\bar{x})$ is equal for load and size indexes.

UNIVERSITÀ DEGLI STUDI DE L'AQUILA



11.7 Criticality Index

The metric specifically introduced in [29] [30] and extended in this paper to consider HPV-based SW partition is the ***Criticality Index***, related to the criticality level associated to each process ps_j. In particular, defined the array $CRIT = \{[crit_1, crit_2, .., crit_j, .., crit_n] : crit_j \in \mathbb{R}$ is the criticality level associated to process $ps_j\}$, then it is possible to define the *Criticality Index* as:

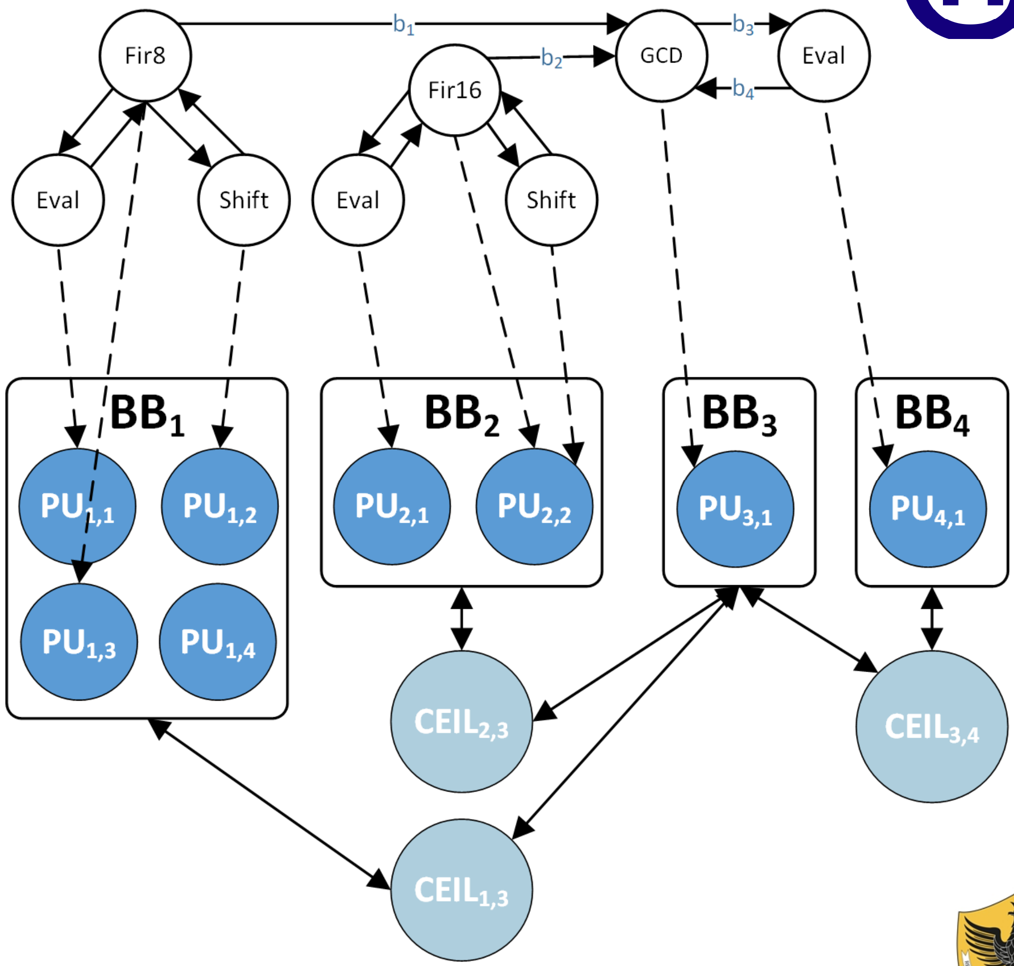
$$
\begin{aligned}
f_{CRIT}(\bar{x}) &= \frac{\sum_{j=1}^{n}\sum_{k=j+1}^{n} mc_{j,k}(\bar{x})}{\frac{n\cdot(n-1)}{2}} \\
MC(\bar{x}) &= \begin{cases} mc_{j,k}(\bar{x}) = 1 & if\ |crit_j - crit_k| > 0 \wedge ps_j \in pu_x \wedge ps_k \in pu_y \wedge pu_x = pu_y \\ mc_{j,k}(\bar{x}) = 1 & if\ |crit_j - crit_k| > 0 \wedge ps_j \in pt_j \in pu_x \wedge ps_k \in pt_k \in pu_y \wedge pt_j = pt_k \wedge pu_x = pu_y \\ mc_{j,k}(\bar{x}) = 0 & otherwise \end{cases}
\end{aligned}
\tag{62}
$$

The goal behind this metric is to avoid having processes with different criticality levels on the same (shared) partition/processor/core resource. If the constraint is not satisfied, the index value becomes 1, so the final cost function has a higher value (in term of utility function) if an individual doesn't satisfy criticality constraint.

HEPSYCODE DSE – CRITICALITY

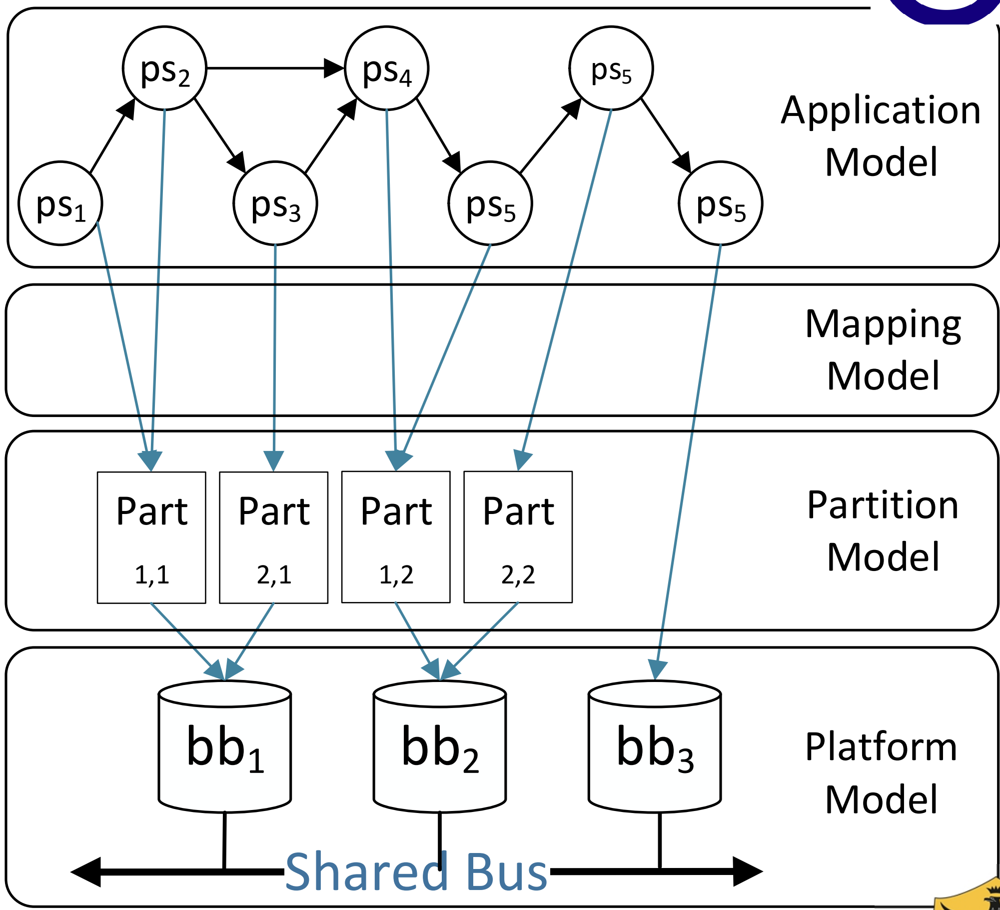
UNIVERSITÀ DEGLI STUDI DE L'AQUILA

HEPSYCODE DSE OUTPUT



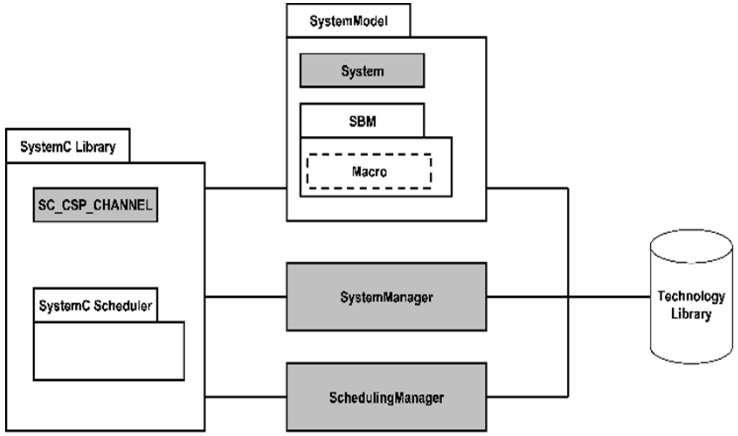
UNIVERSITÀ DEGLI STUDI DE L'AQUILA

HEPSYCODE DSE OUTPUT

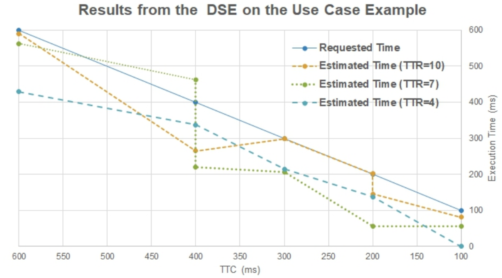


UNIVERSITÀ DEGLI STUDI DE L'AQUILA

HEPSYCODE TIMING SIMULATION



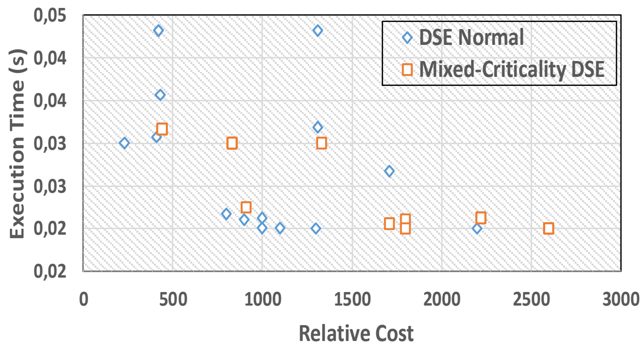
TTC (s)	Allocation	FCFS (s)	OH FCFS (s)	FP (s)	OH FP (s)
0,2	All (1)	0,11	0,02 (1)	0,23	0,14 (1)
0,2	All (3)	0.06	0.01 (3)	0.13	0.09 (3)
0,1	23489 (1) 567 (2)	0.06	0.01 (1) 0.02 (2)	0,11	0.06 (1) 0.13 (2)
0,1	2349 (1) 5678 (3)	0.05	0.01 (1) 0.03 (3)	0.09	0.05 (1) 0.07 (3)
0,05	234 (1) 5678 (3)	0.04	0.01 (1) 0.01 (3)	0.08	0.05 (1) 0.04 (3)
0,05	234 (1) 567 (2) 89 (3)	0,05	0.02 (1) 0.01 (2) 0.04 (3)	0,05	0.03 (1) 0.02 (2) 0.04 (3)
0,05	234 (1) 67 (4) 589 (3)	0,03	0.01 (1) 0.02 (3)	0,04	0.01 (1) 0.02 (3)
0,02	All (4)	0.001	-	0.001	-



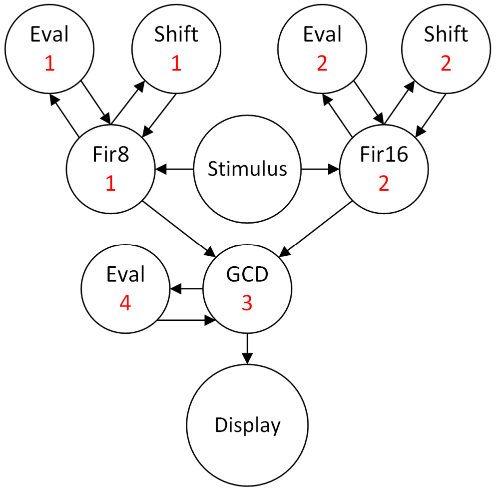
UNIVERSITÀ DEGLI STUDI DE L'AQUILA

HEPSYCODE MIXED-CRITICALITY EXAMPLE

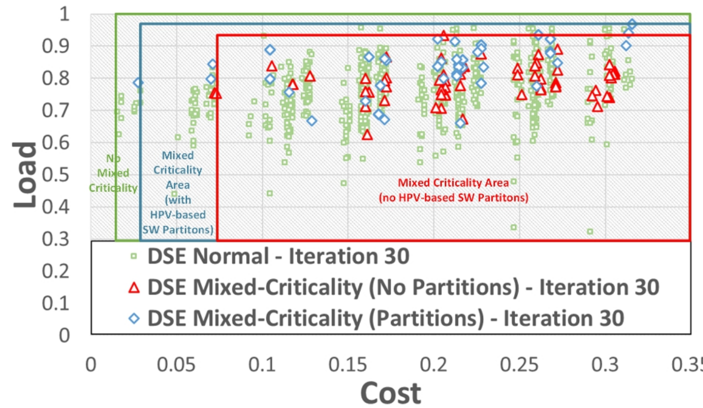
Parameters	Nr.	Values
BBs	≤ 8	2 8051, 2 DSPIC, 2 LEON3, 2 Spartan3an, 2 Virtex-7
App. processes	8	CSP processes
App. Channels	15	CSP channels
GA Selection	1	Random
GA Crossover (C)	1	One-Point
C probability (pc)	1	0.3
GA Mutation (M)	1	Random
M probability (pm)	1	0.1
Survival Selection (S)	1	Fitness-Based
S probability (ps)	1	0.15
Search Iteration (I)	40	-
Initial Population Size (P)	1000	Number of Starting individuals



HEPSYCODE MIXED-CRITICALITY EXAMPLE

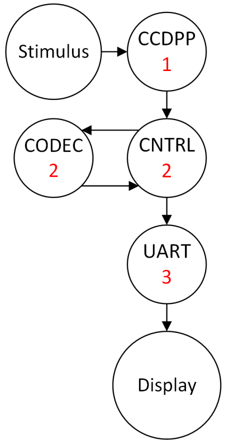


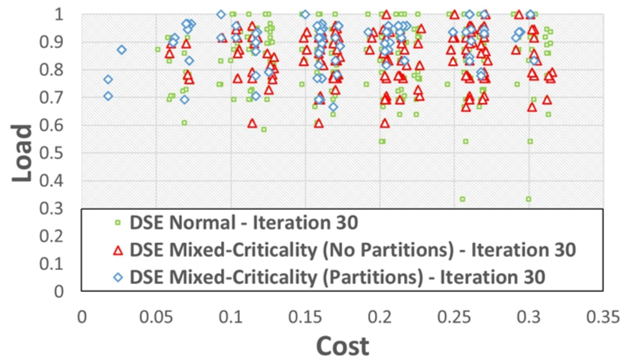
Fir-Fir-GCD is a synthetic application that takes in input two values (triggered by Stimulus), makes two filtering actions (Fir8 and Fir16) and then makes the greatest common divisor (GCD) and displays the result.



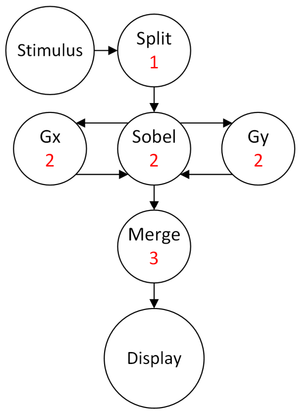
L. Pomante and P. Serri, "*SystemC-based HW/SW Co-Design of Heterogeneous Multiprocessor Dedicated Systems*", International Journal of Information Systems, Volume 1, July 30, 2014

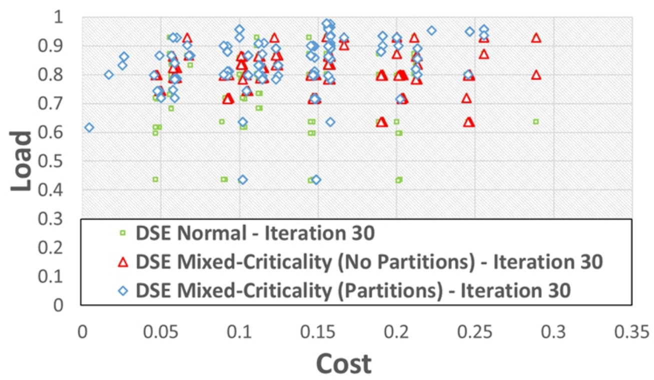
HEPSYCODE MIXED-CRITICALITY EXAMPLE





Digital Camera is an academic use case that represents a simple digital camera. The application captures images, stores images in digital format and downloads images to PC. It is possible to extend it with variable size images, image deletion, digital stretching, zooming in and out, etc.

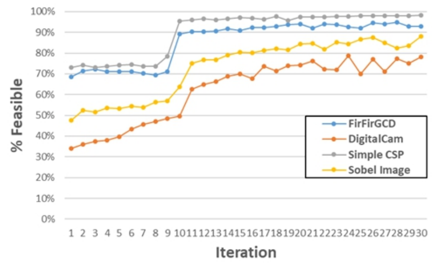




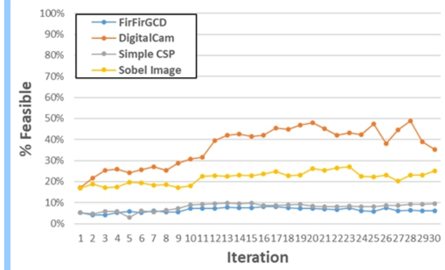
Sobel Image is an application that performs the Sobel filter on sample input image.

UNIVERSITÀ DEGLI STUDI DE L'AQUILA

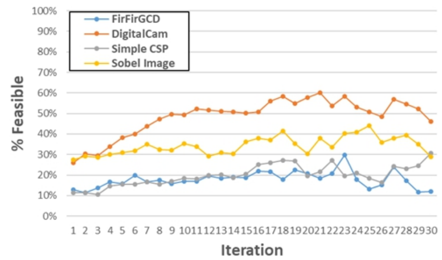
HEPSYCODE MIXED-CRITICALITY EXAMPLE



Normal



No HPV-Based SW Partition



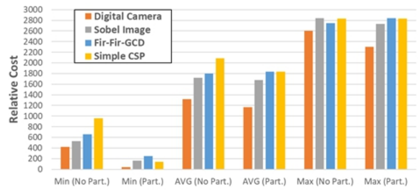
With HPV-Based SW Partition

TABLE I
DSE PERCENTAGE REDUCTION OF FEASIBLE SOLUTION

Use Cases	AVG Red. No Part. - Nor.	AVG Red. Part. - Nor.	AVG No Part. - Part.
Fir-Fir-GCD	79%	68%	11%
Digital Camera	25%	14%	12%
Simple CSP	82%	70%	12%
Sobel Image	51%	39%	13%
AVG	59.25%	32.75%	12%

TABLE II
DSE MINIMUM (RELATIVE) COST ANALYSIS

Use Cases	Min No Part.	Min Part.	% Reduction No Part. - Part.
Fir-Fir-GCD	660	250	62.1%
Digital Camera	420	40	90.5%
Simple CSP	530	160	69.8%
Sobel Image	960	140	85.4%
AVG	642.5	147.5	76.95%



DSE Minimum (Relative) Cost Analysis

UNIVERSITÀ DEGLI STUDI DE L'AQUILA

HEPSYCODE IN MEGAM@RT2

Overview of the approach

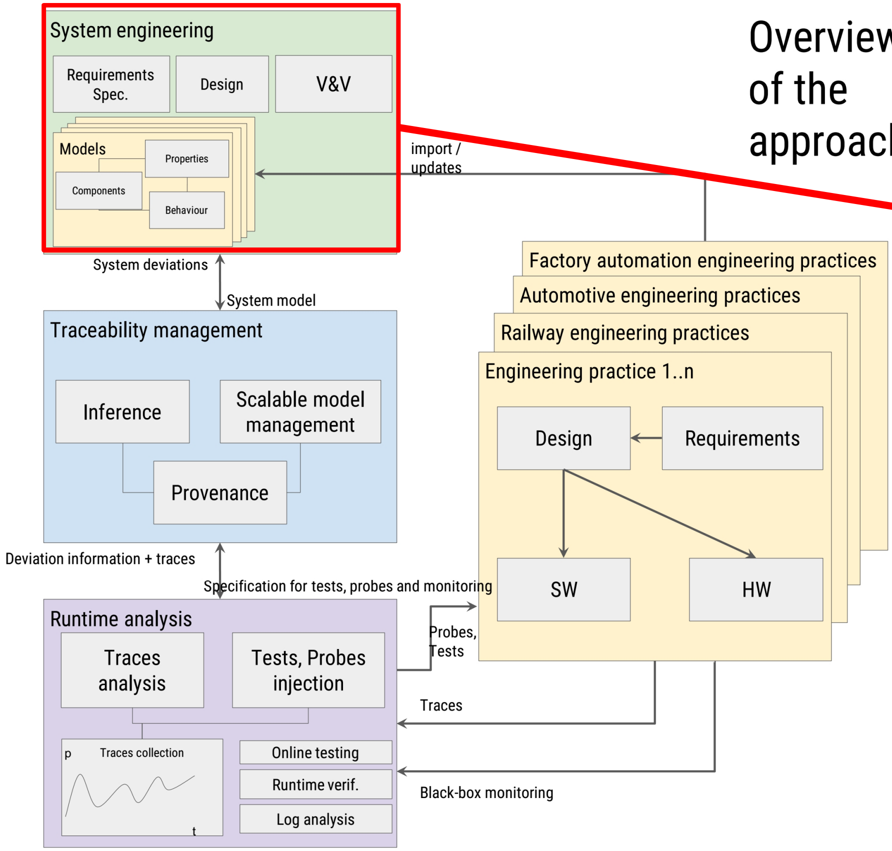
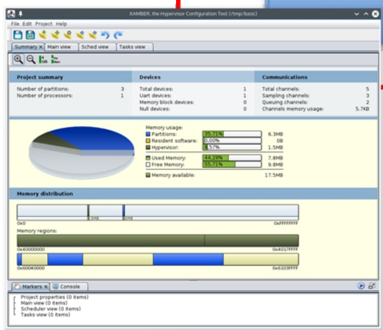


Figure 1.1: MegaM@Rt overall approach

+



HEPSYCODE – XAMBER INTEGRATION

Metamodel and ancillary support

The following set of requirements derive from the SYS-0100 basic requirement. They specify the metamodel properties and the characteristics of the environment that support the modelling activities.

ID	Definition	Refines
SYS-010100	The SE must support standard modelling languages, standard profiles (i.e. AADL, UML, SysML, MARTE, fUML, UTP) and profile customisation capability.	CSY_01, CSY_02, IKER_01, NOK_01, TEK_01, TEK_02.

Requirement Modelling

The following set of requirements derive from the SYS-0200 basic requirement. They specify the support that the System Engineering Tool Set must provide to the system requirements specification activities.

ID	Definition	Refines
SYS-020202	The SE must allow modelling the non-functional/extra-functional requirements and constraints (e.g. execution delay, power consumption, etc.)	CSY_01, NOK_02.

System Architecture & Design

The following set of requirements derive from the SYS-0300 basic requirement. They specify the methodologies and tools characteristics required to support the system design modelling.

ID	Definition	Refines
SYS-030101	The SE must support the architectural views definition and modelling.	NOK_16, NOK_19, NOK_20.
SYS-030402	The SE must support the reuse of existing models or patterns	BT_04, BT_05, BT_06, CAM_01, NOK_04, NOK_12

Hepsycode & Xamber

- CSY: Railway – Platform screen doors control
- IKER: Smart Warehouse – Deployment and supervision of agents
- NOK: Telecom – Base Transceiver Station
- TEK: Short range communication – Indoor positioning

Hepsycode

- BT: Transportation – Train Control and Management System
- CAM: Traffic Monitoring – Intelligent Surveillance System

Xamber

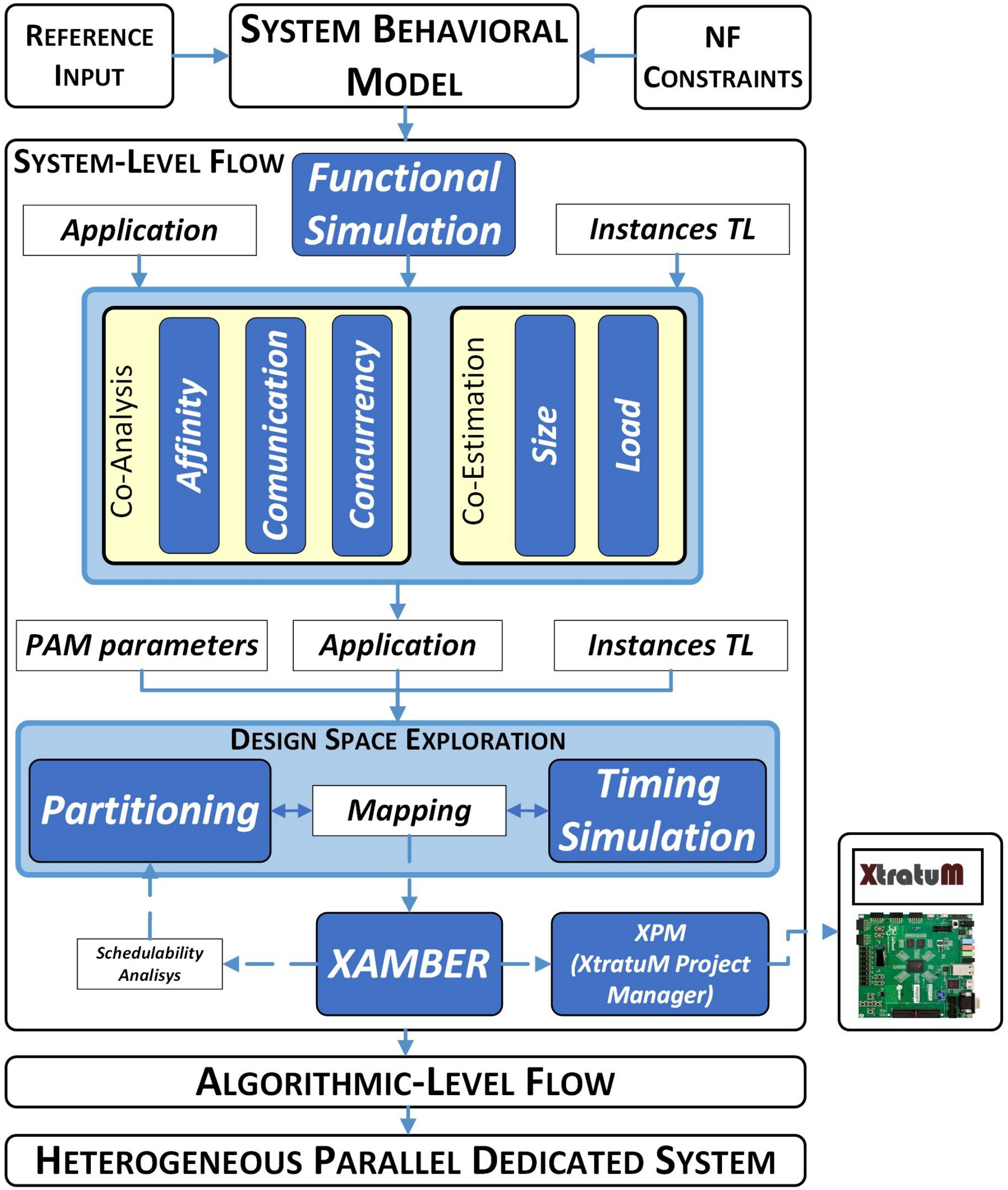
- BT: Transportation – Train Control and Management System
- TRT: Avionics – Flight Management System

Hepsycode Interest (No System Requirement Covered)

- TRT: Avionics – Flight Management System

UNIVERSITÀ DEGLI STUDI DE L'AQUILA

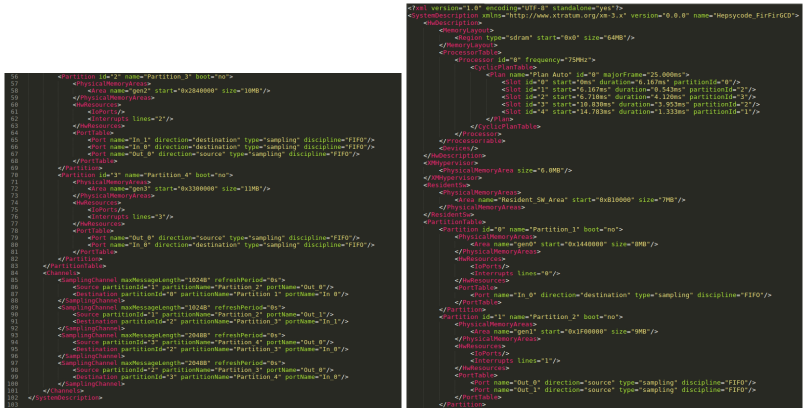
HEPSYCODE – XAMBER INTEGRATION

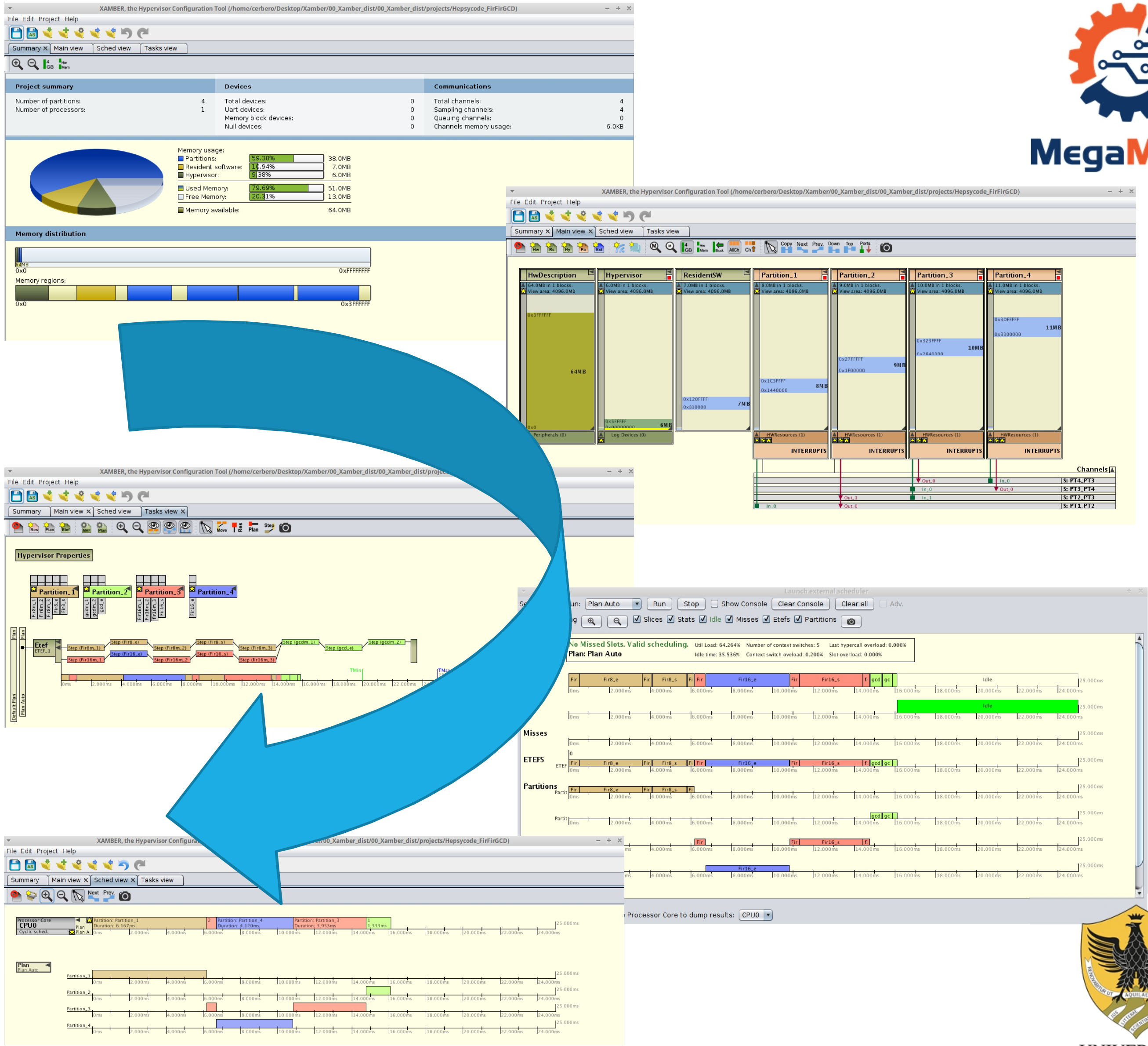


MegaM@Rt2

UNIVERSITÀ DEGLI STUDI DE L'AQUILA

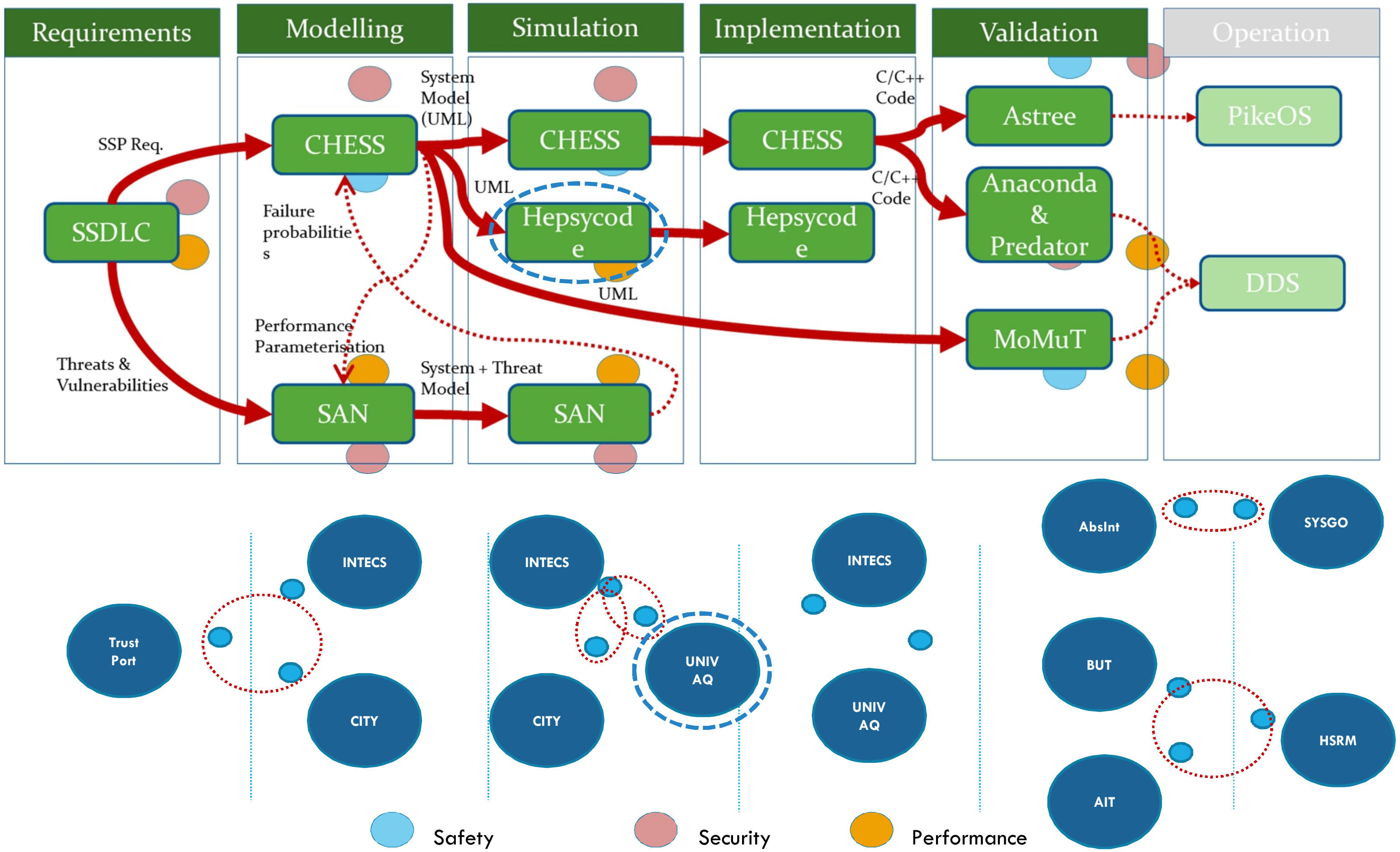
HEPSYCODE – XAMBER INTEGRATION

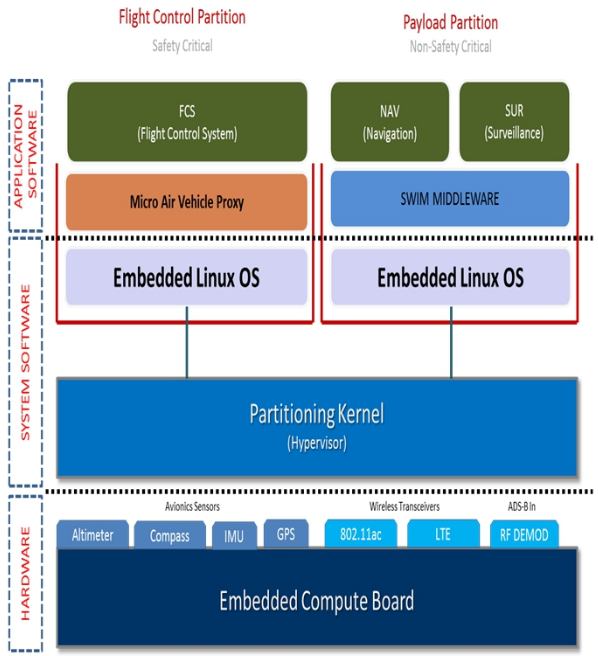




UNIVERSITÀ DEGLI STUDI DE L'AQUILA

HEPSYCODE IN AQUAS – UC1

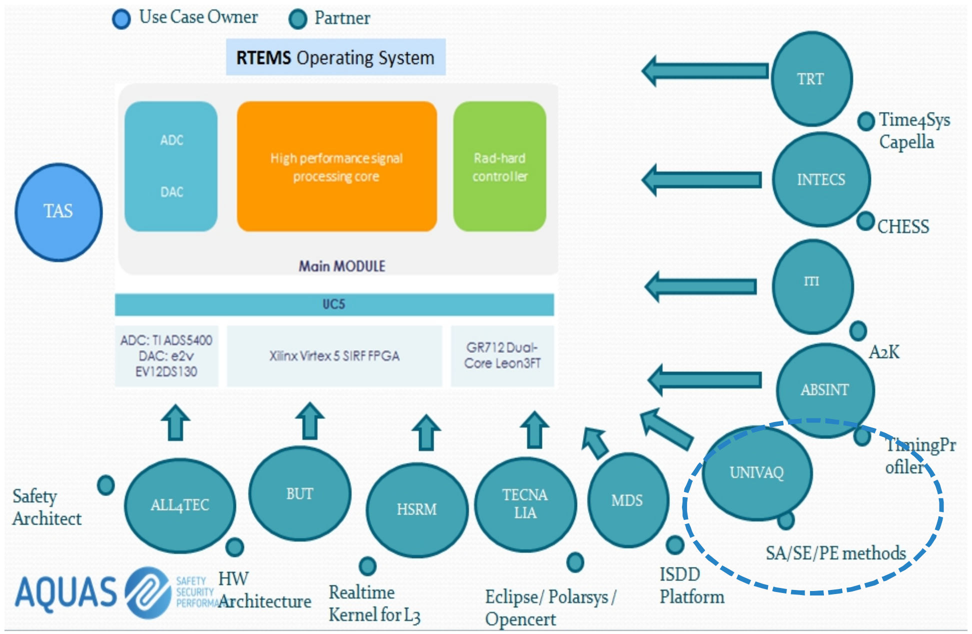




UNIVERSITÀ DEGLI STUDI DE L'AQUILA

HEPSYCODE IN AQUAS – UC5

AQUAS SAFETY SECURITY PERFORMANCE



Interfaces to different tools	Application model with criticality requirement	Hepsycode methodology	Technology definition
art2kitect CHESS TimingProfiler CODEO		Model PikeOS on Leon 3 environment Criticality systems design specification Design space exploration with partitioning application model PikeOS Benchmarking Activities	HW architecture SW architecture Mapping

UNIVERSITÀ DEGLI STUDI DE L'AQUILA

HEPSYCODE ECOSYSTEM

European Projects: EMC² · MegaM@Rt² · AQUAS (Safety Security Performance) · Fit Opti Vis · …

Modeling Tools: eSSYN (embedded software synthesis) · DEADALUS FRAMEWO RK · ForSyDe · PREESM · …

Hypervisor: XtratuM · PikeOS · …

Hepsy Code

UNIVERSITÀ DEGLI STUDI DE L'AQUILA

HEPSYCODE REFERENCE

HW/SW CO-DEsign of HEterogeneous Parallel dedicated SYstems (Hepsycode)

WEBSITE

www.hepsycode.com

DOWNLOAD

Official git repository: https://bitbucket.org/vittorianomuttillo87/tool-hepsycode/src/master/

SYSTEM REQUIREMENTS

- Ubuntu 16.04.3 LTS (Xenial Xerus);
- SystemC Libraries version 2.3.0;
- Eclipse Oxygen Modelling Tools with the following plugins in place:
- Oxygen.3a Release (4.7.3a)

RELEASE NOTES

Latest Release: 1.0.0

LICENSE

GNU GENERAL PUBLIC LICENSE Version 2, June 1991 (see https://www.gnu.org/licenses/old-licenses/gpl-2.0.html)

DEVELOPER RESOURCES

Source Repositories: https://bitbucket.org/vittorianomuttillo87/tool-hepsycode/src/master/

- Clone:
  - ssh: git@bitbucket.org:vittorianomuttillo87/tool-hepsycode.git
  - https: https://vittorianomuttillo87@bitbucket.org/vittorianomuttillo87/tool-hepsycode.git

You can use the code from these repositories to experiment, test, build, and create patches, issue pull requests (only by request).

SUPPORT

We currently support:

1. Email:
   - Luigi Pomante, luigi.pomante@univaq.it
   - Vittoriano Muttillo, vittoriano.muttillo@graduate.univaq.it
   - Giacomo Valente, giacomo.valente@graduate.univaq.it
   - (please take care to use [HEPSYCODE SUPPORT] as object
2. Issues on bitbucket.org

UNIVERSITÀ DEGLI STUDI DE L'AQUILA

THANKS!

Questions?