CPSIoT 2019
Summer School on Cyber-Physical Systems and Internet-of-Things
*Budva, Montenegro, June 11, 2019*

# Design space exploration for Hypervisor-based mixed-criticality systems

Authors:
Vittoriano Muttillo, Luigi Pomante
vittoriano.muttillo@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. Safety Assurance Standards
3. Mixed Criticality Systems Analysis
4. Mixed-Criticality Classification
5. Mixed-Criticality HW/SW Co-Design
6. Proposed Methodology
7. ESL Methodology Main Elements
8. HepsyCode-MC
9. Case Studies
10. Hepsycode Ecosystem
11. Publications and European Projects
12. Conclusion and Future Works

# 1 Introduction

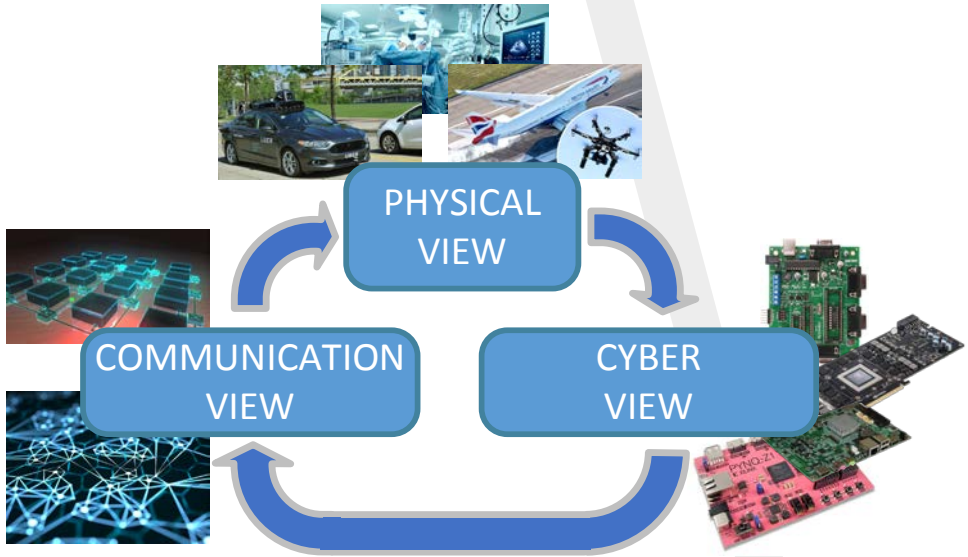
"Brief Introduction to Mixed Criticality and Cyber Physical Systems"

# Context: Cyber-Physical System

- A cyber-physical system (CPS) is an integration of computation with physical processes whose behavior is defined by both cyber and physical parts of the system.

- Embedded computers and networks monitor and control the physical processes, usually with feedback loops where physical processes affect computations and vice versa.

- As an intellectual challenge, CPS is about the intersection, not the union, of the physical and the cyber.




[1] Lee, E. A., Seshia, S. A.,: Introduction to Embedded Systems, a Cyber-Physical Systems approach, Second Edition, LeeSeshia.org, 2015

# Mixed-Criticality Embedded Systems

- The growing complexity of embedded digital systems based on modern System-on-Chip (SoC) adopting explicit heterogeneous parallel architectures has radically changed the common design methodologies.

- HW/SW co-design methodologies are of renovated relevance

- A growing trend in embedded systems domain is the development of mixed-criticality systems where multiple embedded applications with different levels of criticality are executed on a shared hardware platform (i.e. Mixed-Criticality Embedded Systems)



Proceedings of CPS&IoT2019 page 439

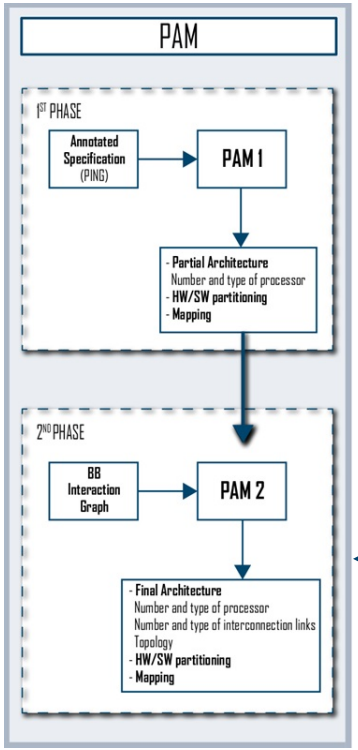
# Mixed-Criticality Systems

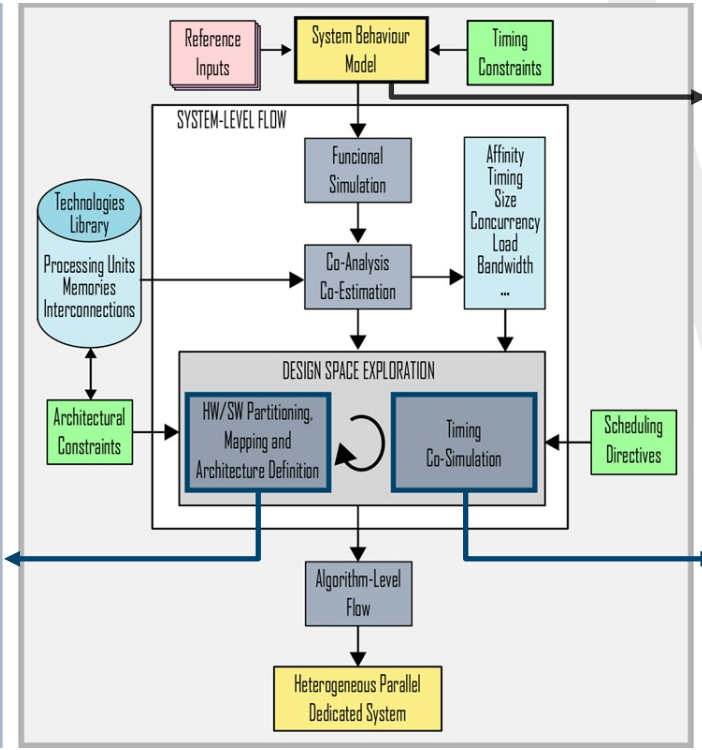

- A mixed criticality system is "*an integrated suite of HW, OS, middleware services and application software that supports the concurrent execution of safety-critical, mission-critical, and noncritical software within a single, secure computing platform*"
- **MAIN GOALS**: development of EDA tools, mainly oriented to support the designer of Mixed-Criticality and Cyber-Physical systems based on heterogeneous multi/many-core platforms, considering **Hypervisor-Based SW partitions**

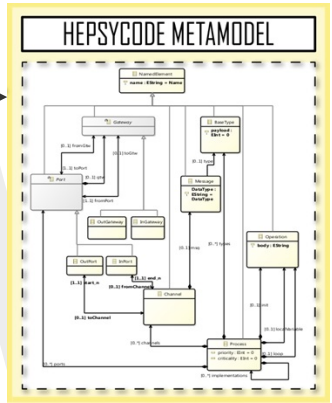
# MECO education & training Goals

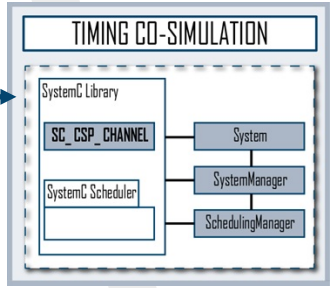
- In the context of real-time embedded systems design, this work starts from a specific methodology (called **HEPSYCODE**: HW/SW **CO**-**DE**sign of **HE**terogeneous **P**arallel Dedicated **SY**stems), based on an existing System-Level HW/SW Co-Design methodology, and introduces the possibility to add real-time and mixed-criticality requirements to the set of non-functional ones

- Focus on **Design Space Exploration** considering **HPV-based SW Partitions**




Official Web Site:
**www.hepsycode.com**

# 2.

# Safety Assurance Standards

"Criticality is a designation of the level of assurance against failure needed for a system component"



Proceedings of CPS&IoT2019 page 442

# Safety Related Standards

- Industry has shown a growing interest in integrating and running independently-developed applications of different "criticalities" in the same (often multicore) platform. Such integrated systems are commonly referred to as **mixed-criticality systems (MCS)**.

- Most of the MCS-related research cite the **safety-related standards** associated to each application domain (e.g. aeronautics, space, railway, automotive) to justify their methods and results. However, those standards are not freely available and do not always clearly and explicitly specify the requirements for mixed-criticality

- New MC task model is in essence the result of combining the **standard hard real-time requirements** (studied by the real-time research community since the 70's) with the **notion of "criticality" of execution**.



Proceedings of CPS&IoT2019 page 443

# System Design and Development Assurance Process

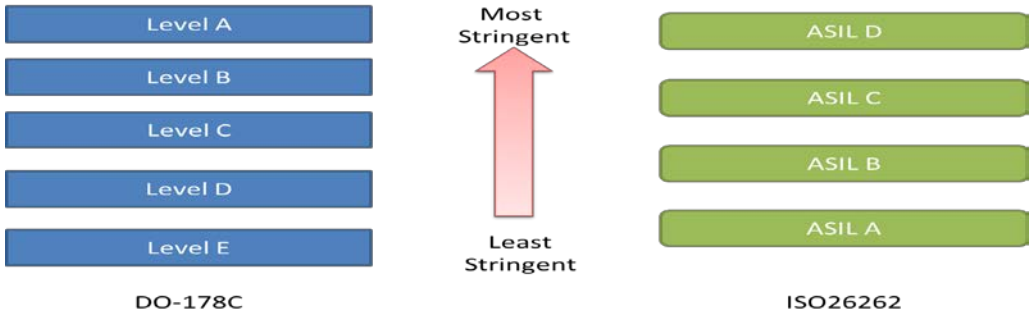
- During a typical development life cycle of a safety-critical system, the behavior and characteristics that are expected from the system are expressed in the form of a list of requirements
  - based on the system operational requirements (what the system is expected to do) and also considering non-functional properties related to safety, security and performance, including timing and energy constraints.

- System safety assessment process must be carried out as part of the development life cycle to determine and categorize the failure conditions of the system (e.g. through a hazard analysis).
  - safety-related requirements are derived as a result of the system safety assessment process, which may include functional, integrity, dependability requirements and design constraints.

- Safety-related requirements are allocated to hardware and software components, thereby specifying the mechanisms required to prevent the faults or to mitigate their effects and avoid the propagation of failures.

# Integrity Level

- Most safety standards use the concept of an **integrity level**, which is assigned to a system or a function. This level will be based on an initial analysis of the consequences of software going wrong. Both standards have clear guidance on how to identify integrity level.
  - **DO-178C** has **Software Development Assurance Level (DAL)**, which are assigned based on the outcome of "anomalous behavior" of a software component – **Level A** for "Catastrophic Outcome", **Level E** for "No Safety Effect".
  - **ISO26262** has **ASIL (Automotive Safety Integrity Level)**, based on the exposure to issues affecting the controllability of the vehicle. ASILs range from **D** for the highest severity/most probable exposure, and **A** as the least.



# Safety Standards

- **GENERAL** (IEC-61508) based on **SIL (Safety Integrity Level)**: Functional safety standards (of electrical, electronic, and programmable electronic)
  - **AUTOMOTIVE** (ISO26262) based on **ASIL (Automotive Safety Integrity Level)** (Road vehicles - Functional safety)
  - **NUCLEAR POWER** (IEC 60880-2)
  - **MEDICAL ELECTRIC** (IEC 60601-1)
  - **PROCESS INDUSTRIES** (IEC 61511)
  - **RAILWAY** (CENELEC EN 50126/128/129])
  - **MACHINERY** (IEC 62061)
- **AVIONIC** based on **DAL (Development Assurance Level )** related to ARP4761and ARP4754
  - DO-178B (Software Considerations in Airborne Systems and Equipment Certification)
  - DO-178C (Software Considerations in Airborne Systems and Equipment Certification, replace DO-178B)
  - DO-254 (Airborne - Design), similar to DO-178B, but for hardware
  - DO-160F (Airborne - Test)
- **MEDICAL DEVICE**
  - FDA-21CFR
  - IEC-62304

# 3.

# Mixed Criticality Systems Analysis

"The more confidence one needs in a task execution time bound (the less tolerant one is of missed deadlines), the larger and more conservative that bound tends to become in practice"



Proceedings of CPS&IoT2019 page 447

# MCS state-of-the-art Model (1)

- Almost 200 papers treating of the scheduling of MCS have been referenced in Burns and Davis* paper, and tens of related papers are still published every year. Most of the works about MCS published by the real-time scheduling research community are based on a **model proposed by Vestal***
  - System has several modes of execution, say **modes $\{1, 2, \dots, L\}$**. The application system is a **set of real-time tasks**, where each task $\tau_i$ is characterized by a period $T_i$ and a deadline $D_i$ (as in the usual real-time task model), an **assurance level $l_i$** and a set of **worst-case computational estimates $\{C_{i,1}, C_{i,2}, \dots, C_{i,l_i}\}$**, under the assumption that $C_{i,1} \leq C_{i,2} \leq \dots \leq C_{i,l_i}$
- The different WCET estimates are meant to model estimations of the **WCET at different assurance levels**. The worst time observed during tests of **normal operational scenarios** might be used as $C_{i,1}$ whereas at **each higher assurance level** the subsequent estimates $C_{i,2}, \dots, C_{i,l_i}$ are assumed to be obtained by more conservative **WCET analysis techniques**.

** Burns, A, Davis, R.I.: "Mixed Criticality Systems - A Review", University of York, 4 March 2016.*
*** S. Vestal, "Preemptive Scheduling of Multi-criticality Systems with Varying Degrees of Execution Time Assurance," Real-Time Systems Symposium (RTSS) 28th IEEE International on, Tucson, AZ, 2007, pp. 239-243.*

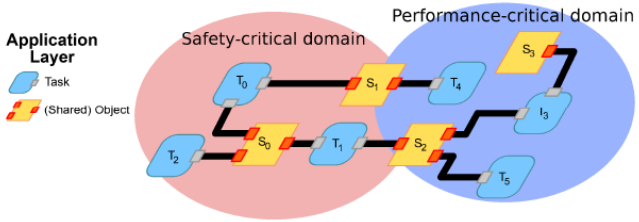
# MCS state-of-the-art Model (2)

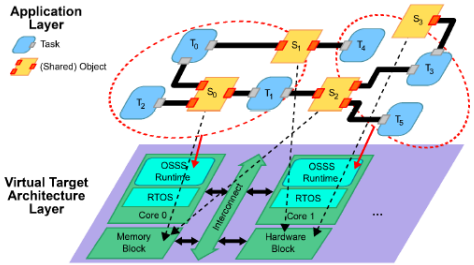
- The system starts its execution in mode 1 and all tasks are scheduled to execute on the core[s]. Then at runtime, if the system is running in mode $k$ then each time the execution budget $C_{i,k}$ of a task $\tau_i$ is overshot, the system switches to mode $k+1$ It results from this transition from mode $k$ to mode $k+1$ that all the tasks of criticality not greater than $k$ (i.e., $l_i \geq k$) are suspended. Mechanisms have also been proposed to eventually re-activate the dropped tasks at some later points in time*.
  - one of the simplifications of this model is the Vestal's model with only two modes, usually referred to as LO and HI modes (which stand for Low- and High-criticality modes).
- Multiple variations of that scheduling scheme exist, some for single-core, others for multicore architectures. In the case of multicore, both global and partitioned scheduling techniques have been studied and solutions for fixed priority scheduling (RM), Earliest Deadline First (EDF) and time triggered scheduling have been proposed in literature.
  - some works also propose to change the priorities or the periods of the tasks during a mode change rather than simply stopping the less critical ones.
  - Note that some works also propose to change the priorities or the periods of the tasks during a mode change rather than simply stopping the less critical ones.

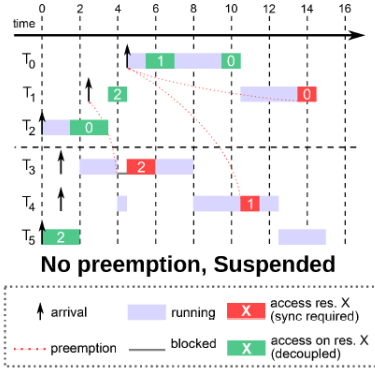
** F. Santy, G. Raravi, G. Nelissen, V. Nelis, P. Kumar, J. Goossens, and E. Tovar. Two protocols to reduce the criticality level of multiprocessor mixed-criticality systems. In RTNS 2013, RTNS '13, pages 183–192. ACM, 2013.*

# MCS Design - OFFIS

Mixed-criticality principles applied to application layer modelling objects:




- Safety-critical domain $T_0, T_1, T_2, S_0$: high-critical tasks and shared objects
- Performance-critical domain $T_3, T_4, T_5, S_3$: low-critical tasks and shared objects
- mixed-critical shared objects: $S_1, S_2$

- Ports
  - Connections to shared object **interfaces**
- Deadline
  - might affect scheduling decisions

**Mixed-Criticality** aware properties:

- Task **periods**
  - Safety-crititcal tasks might have increased **activation frequencies** in high criticality levels
- **Vector** of computation times
  - platform-dependent, → later
  - One for each criticality level (e.g. *LO*, *HI*)
- **Criticality level**
  - statically defined at design time


**Task Definition**

Task $T_i \in \tau$,
$T_i = \left(\vec{T}_i, D_i, \vec{C}_i, \pi_i, L_i\right)$

- a **vector** of periods $\vec{T}_i$ (minimum arrival interval)
- $D_i$: deadline
- $\vec{C}_i$: **vector of computation times** (one for each criticality level)
- $\pi_i$: **ports** for connecting to communication objects
- $L_i$: **criticality** level (e.g. *LO*, *HI*)

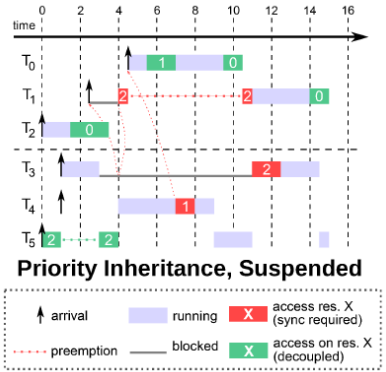
Possible Shared Object realisation should include:


- Cached **internal** state $\Sigma, \Sigma'$
  - before performing the call, create copy: $\Sigma' \leftarrow \Sigma$
  - active calls work on **state copy** $\Sigma'$
  - after the call, **update** original state: $\Sigma \leftarrow \Sigma'$
- **Preemptible scheduling** of access, policy defined by attribute $\Phi$ (e.g. round-robin, fixed-prio)
  - if criticality level $L$ increases, **abort** active calls $< L$
  - discard $\Sigma'$
  - only **allow** calls from tasks with criticality level $\geq L$

**Shared Object**

$S_i = (\Sigma, \Sigma', L, M, I, \Phi)$

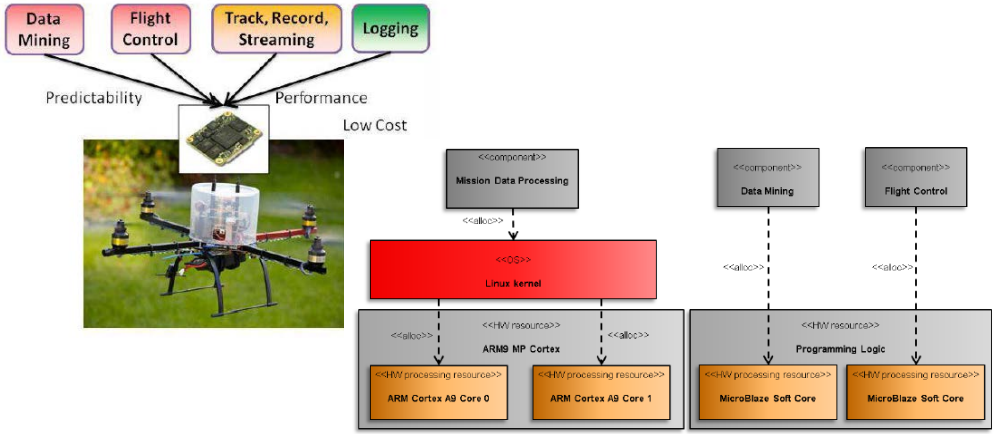
- $\Sigma_{0,1}$: **inner states** (containing abstract data types),
- $L$: current criticality level (e.g. *LO*, *HI*)
- $M \subseteq \Sigma \times \Sigma$: a set of **methods** or **services** (e.g. *read*(), *write*())
- $I \subseteq \mathcal{P}(M)$: Interfaces for grouping methods
- $\Phi$: **resource arbitration policy**






Proceedings of CPS&IoT2019 page 450

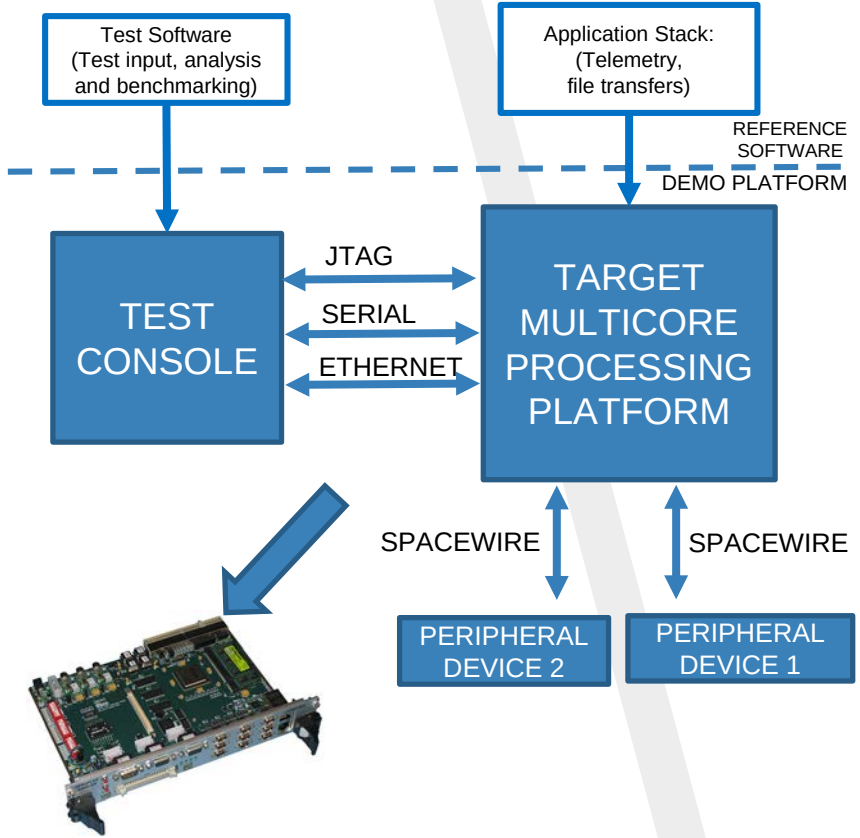
# Industrial and Academic MCS Case Study




**Safety critical tasks**: All tasks which are needed for a stable and safety flight of the multi-rotor system, e.g. the flight and navigation controllers. An error, like missing a deadline, will cause a crash-landing!

**Mission critical tasks**: All tasks which are not needed for a safe flight, but may also have defined deadlines, e.g. tasks which are belonging to the payload processing, like video processing.

**Uncritical tasks**: All tasks which are not needed either for a safe flight or a correct execution of the mission task, e.g. control of the debug LEDs or transmission of telemetry data.



# MCS state-of-the-art Model (2)

## Research – Industrial Domain Misunderstandings

- Academic papers: "system criticality" as a mode of execution of software tasks (e.g. high or low criticality). Mode change allowed
- Industrial domain: "system criticality" refers to the level of assurance (e.g. DAL, SIL or ASIL) applied in the development of a software application that implements critical system functionalities (i.e. safety functions)

## Mixed-Criticality Challenges

- Scheduling (Priority Vs Safety), Partition (Isolation), Performance (WCET estimation), Predictability (Graceful Degradation), Manufactory Cost, Fault-tolerance, Power-consumption, Networking



Proceedings of CPS&IoT2019 page 452

MECO education & training

# 4.

# Mixed-Criticality Classification

*"A major industrial challenge arises from the need to face cost efficient integration of different applications with different levels of safety and security on a single computing platform in an open context"*



Proceedings of CPS&IoT2019 page 453

# MCS Architectures

## Separation technique:

- **Timing separation**: scheduling policy, temporal partitioning with HVP, NoC
- **Spatial separation**: one task per core, one task on HW ad hoc (DSP, FPGA), spatial partition with HVP, NoC, MMU, MPU etc.

## ➢HW:

- **Temporal isolation**: Scheduling HW
- **Spatial isolation**: separated Task on dedicated components (HW ad hoc, FPGA etc.)

## ➢Single core:

- **Temporal isolation**: Scheduling policy with SO o RTOS, Scheduling policy with HVP
- **Spatial isolation** : MMU, MPU, HVP Partitioning

## ➢Multi-core

- **Architecture**: shared memory systems, Uniform Memory Architecture, UMA (SMP), Not Uniform Memory Architecture, NUMA, distributed systems, NoC
- **Temporal isolation**: Scheduling policy with SO o RTOS, Scheduling policy with HVP
- **Spatial isolation**: MMU, MPU, HVP partitioning

## ➢Many-core

- **Work in progress**

# MCS Technologies

## Tecnologies:

- **Hardware**: HW ad hoc, FPGA, DSP, Processor
  - **Processor**: LEON3, ARM, MICROBLAZE etc.
- **Software**: Bare-metal, OS, RTOS, HVP
  - **OS**: Linux etc.
  - **RTOS**: eCos, RTEMS, FreeRTOS, Threadx, VxWorks, EriKa etc.
  - **HVP**: PikeOS, Xtratum, Xen etc.

HVP

PikeOS

XtratuM

Xen

RTOS

THREADX

eCOS

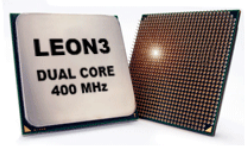
VxWorks

RTEMS

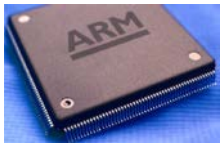
ERIKA ENTERPRISE

freeRTOS

Hardware





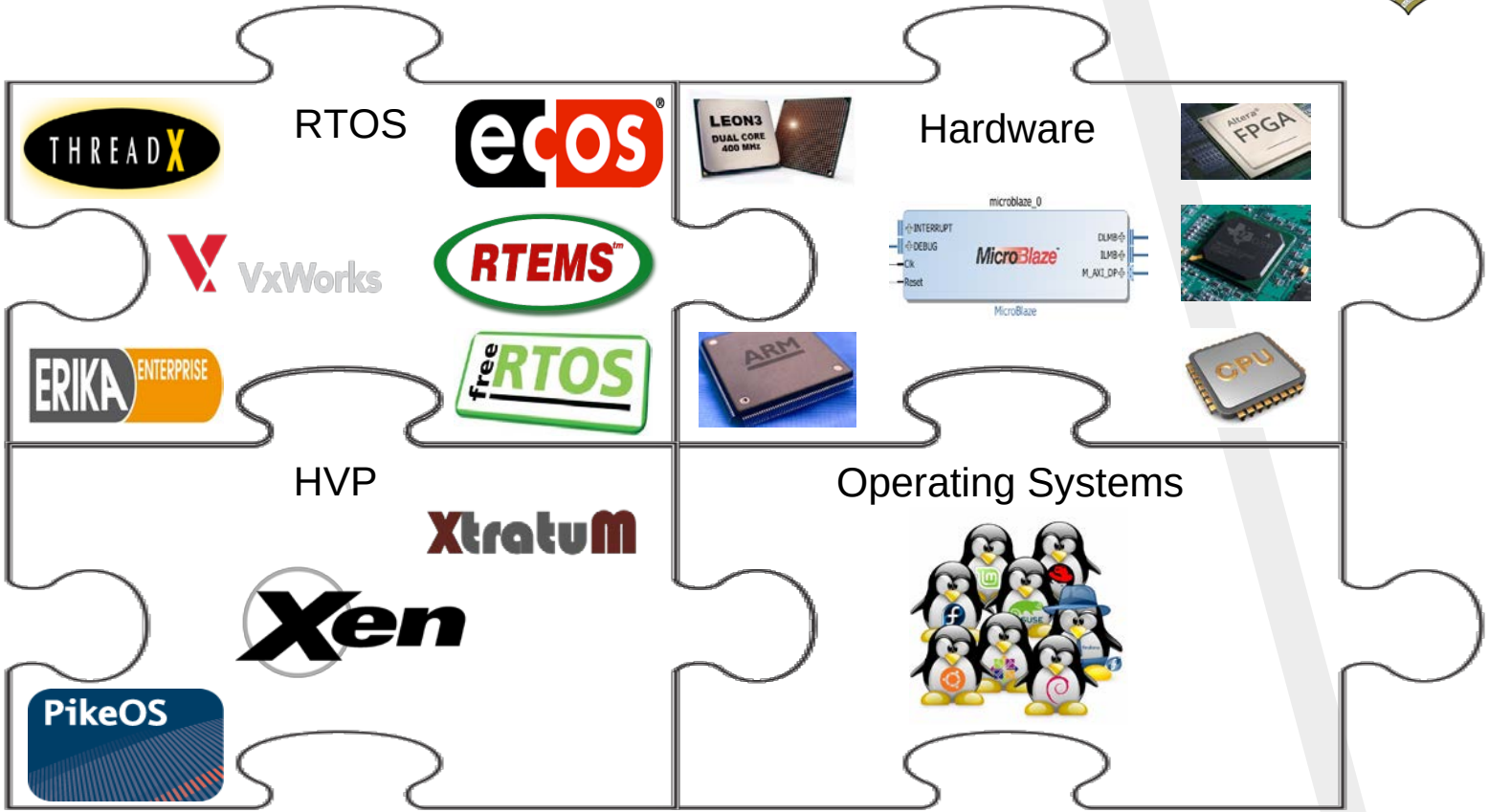


OS

# MCS Implementations

## Scheduling:

- 0-Levels
- 1-Level
- 2-Levels
- 3-Levels?






Proceedings of CPS&IoT2019 page 456

# MC Implementation Solutions (M-CIS)

## Tecnologies:

- **Hardware**: HW ad hoc, FPGA, DSP, Processor.
  - **Processor**: LEON3, ARM, MICROBLAZE, etc.
- **Software**: Bare-metal, OS, RTOS, HVP
  - **OS**: Linux etc.
  - **RTOS**: eCos, RTEMS, FreeRTOS, Threadx, VxWorks, EriKa etc.
  - HVP: PikeOS, Xtratum, Xen etc
- **Many-core**: WIP

| Separation Technique | HW | Single core | Multi-core | Many-core |
|---|---|---|---|---|
| **Spatial** | **0-levels scheduling** [125],[129],[130] | **0-levels scheduling** [7],[129],[130],[131] | **0-levels scheduling** [132],[124],[125],[131] | [124],[125], [126],[127],[128] |
| | | **1-level scheduling** [97],[133],[134],[135] [141],[142],[143] | **1-level scheduling** [7],[136],[137],[125],[138],[139],[140] ,[133],[144],[141],[145],[146],[61],[14 2],[147],[131] | |
| | | **2-levels scheduling** [141],[148] | **2-levels scheduling** [149],[125],[126],[150],[151],[128] | |
| | | **3-levels scheduling** [154] | **3-levels scheduling** [154] | |
| **Temporal** | **0-levels scheduling** [125],[129],[130] | **0-levels scheduling** [7],[129],[130],[131] | **0-levels scheduling** [132],[124],[125],[129],[130],[128] | [124],[125], [126],[127],[128] |
| | | **1-level scheduling** [133],[134],[135], [142],[143],[131] | **1-level scheduling** [7],[136],[137],[155],[125],[138],[140] [133],[144],[146],[61],[156],[142] | |
| | | **2-levels scheduling** [141],[148] | **2-levels scheduling** [149],[125],[126],[152],[141],[145], [153],[148],[49],[128] | |
| | | **3-levels scheduling** [154] | **3-levels scheduling** [154] | |

# References

[7] Henning Schlender, S¨oren Schreiner, Malte Metzdorf, Kim Gr¨uttner, and Wolfgang Nebel. Teaching mixed-criticality: Multi-rotor flight control and payload processing on a single chip. In Proceedings of the WESE'15: Workshop on Embedded and Cyber-Physical Systems Education, WESE'15, pages 9:1–9:8, New York, NY, USA, 2015. ACM.

[49] Gernot Heiser and Ben Leslie. The OKL4 microvisor: Convergence point of microkernels and hypervisors. In Asia-Pacific Workshop on Systems (APSys), pages 19–24, New Delhi, India, August 2010

[61] Vittoriano Muttillo, Giacomo Valente, and Luigi Pomante. Criticality-driven design space exploration for mixed-criticality heterogeneous parallel embedded systems. In Proceedings of the 9th Workshop and 7th Workshop on Parallel Programming and RunTime Management Techniques for Manycore Architectures and Design Tools and Architectures for Multicore Embedded Computing Platforms, PARMA-DITAM '18, pages 63–68, New York, NY, USA, 2018. ACM.

[97] S. Vestal. Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance. In 28th IEEE International Real-Time Systems Symposium (RTSS 2007), pages 239–243, Dec 2007.

[124] H. Isakovic and R. Grosu. Aheterogeneous time-triggered architecture on a hybrid system-on-a-chip platform. In 2016 IEEE 25th International Symposium on Industrial Electronics (ISIE), pages 244–253, June 2016.

[125] Haris Isakovic, Radu Grosu, Denise Ratasich, Jiri Kadlec, Zdenek Pohl, Steve Kerrison, Kyriakos Georgiou, Kerstin Eder, Norbert Druml, Lillian Tadros, Flemming Christensen, Emilie Wheatley, Bastian Farkas, Rolf Meyer, and Mladen Berekovic. Asurvey of hardware technologies for mixed-critical integration explored in the project emc2. In Stefano Tonetta, Erwin Schoitsch, and Friedemann Bitsch, editors, Computer Safety, Reliability, and Security, pages 127–140, Cham, 2017. Springer International Publishing.

[126] S. Saidi, R. Ernst, S. Uhrig, H. Theiling, and B. D. de Dinechin. The shift to multicores in real-time and safety-critical systems. In 2015 International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS), pages 220–229, Oct 2015.

[127] F. Federici, M. Micozzi, V. Muttillo, L. Pomante, and G. Valente. Simulationbased analysis of a hardware mechanism to support isolation in mixed-criticality network on chip. In 2017 European Modelling Symposium (EMS), pages 185– 190, Nov 2017.

[128] Martin Schoeberl, Sahar Abbaspour, Benny Akesson, Neil Audsley, Ra↵aele Capasso, Jamie Garside, Kees Goossens, Sven Goossens, Scott Hansen, Reinhold Heckmann, Stefan Hepp, Benedikt Huber, Alexander Jordan, Evangelia Kasapaki, Jens Knoop, Yonghui Li, Daniel Prokesch, Wolfgang Pu"tsch, Peter Puschner, Andr Rocha, Cludio Silva, Jens Spars, and Alessandro Tocchi. Tcrest: Time-predictable multi-core architecture for embedded systems. Journal of Systems Architecture, 61(9):449 – 471, 2015.

[129] Rodolfo Pellizzoni, Patrick Meredith, Min-Young Nam, Mu Sun, Marco Caccamo, and Lui Sha. Handling mixed-criticality in soc-based real-time embedded systems. In Proceedings of the Seventh ACM International Conference on Embedded Software, EMSOFT '09, pages 235–244, New York, NY, USA, 2009. ACM

[130] M. Zimmer, D. Broman, C. Shaver, and E. A. Lee. Flexpret: Aprocessor platform for mixed-criticality systems. In 2014 IEEE 19th Real-Time and Embedded Technology and Applications Symposium (RTAS), pages 101–110, April 2014.

MECO education & training

# References

[131] W. Weber, A. Hoess, J. v. Deventer, F. Oppenheimer, R. Ernst, A. Kostrzewa, P. Dor, T. Goubier, H. Isakovic, N. Druml, E. Wuchner, D. Schneider, E. Schoitsch, E. Armengaud, T. Sderqvist, M. Traversone, S. Uhrig, J. C. Prez-Corts, S. Saez, J. Kuusela, M. v. Helvoort, X. Cai, B. Nordmoen, G. Y. Paulsen, H. P. Dahle, M. Geissel, J. Salecker, and P. Tummeltshammer. The emc2 project on embedded microcontrollers: Technical progress after two years. In 2016 Euromicro Conference on Digital System Design (DSD), pages 524–531, Aug 2016.

[132] I. Sourdis, D. A. Khan, A. Malek, S. Tzilis, G. Smaragdos, and C. Strydis. Resilient chip multiprocessors with mixed-grained reconfigurability. IEEE Micro, 36(1):35–45, Jan 2016.

[133] M. Paulitsch, O. M. Duarte, H. Karray, K. Mueller, D. Muench, and J. Nowotsch. Mixed-criticality embedded systems – a balance ensuring partitioning and performance. In 2015 Euromicro Conference on Digital System Design, pages 453–461, Aug 2015.

[134] Andreas Gerstinger, Heinz Kantz, and Christoph Scherrer. Tas control platform: Aplatform for safety-critical railway applications. ERCIM News, 2008, 2008.

[135] M. G. Hill and T. W. Lake. Non-interference analysis for mixed criticality code in avionics systems. In Proceedings ASE 2000. Fifteenth IEEE International Conference on Automated Software Engineering, pages 257–260, Sept 2000.

[136] J. Steinbaeck, A. Tengg, G. Holweg, and N. Druml. A3d time-of-flight mixed criticality system for environment perception. In 2017 Euromicro Conference on Digital System Design (DSD), pages 368–374, Aug 2017.

[137] G. Macher, A. Hller, E. Armengaud, and C. Kreiner. Automotive embedded software: Migration challenges to multi-core computing platforms. In 2015 IEEE 13th International Conference on Industrial Informatics (INDIN), pages 1386–1393, July 2015.

[138] T. Bijlsma, M. Kwakkernaat, and M. Mnatsakanyan. Areal-time multi-sensor fusion platform for automated driving application development. In 2015 IEEE 13th International Conference on Industrial Informatics (INDIN), pages 1372– 1377, July 2015.

[139] D. Muench, O. Isfort, K. Mueller, M. Paulitsch, and A. Herkersdorf. Hardwarebased i/o virtualization for mixed criticality real-time systems using pcie sr-iov.
In 2013 IEEE 16th International Conference on Computational Science and Engineering, pages 706–713, Dec 2013.

[140] Z. Mynar, L. Vesely, and P. Vaclavek. Pmsm model predictive control with field-weakening implementation. IEEE Transactions on Industrial Electronics, 63(8):5156–5166, Aug 2016

[141] J. Kim, M. Yoon, S. Im, R. Bradford, and L. Sha. Optimized scheduling of multi-ima partitions with exclusive region for synchronized real-time multi-core systems. In 2013 Design, Automation Test in Europe Conference Exhibition (DATE), pages 970–975, March 2013.

[142] M. Pagani, A. Balsini, A. Biondi, M. Marinoni, and G. Buttazzo. Alinux-based support for developing real-time applications on heterogeneous platforms with dynamic fpga reconfiguration. In 2017 30th IEEE International System-on-Chip Conference (SOCC), pages 96–101, Sept 2017.

# References

[143] Kees Goossens, Arnaldo Azevedo, Karthik Chandrasekar, Manil Dev Gomony, Sven Goossens, Martijn Koedam, Yonghui Li, Davit Mirzoyan, Anca Molnos, Ashkan Beyranvand Nejad, Andrew Nelson, and Shubhendu Sinha. Virtual execution platforms for mixed-time-criticality systems: The compsoc architecture and design flow. SIGBED Rev., 10(3):23–34, October 2013.

[144] Ankit Agrawal, Gerhard Fohler, Johannes Freitag, Jan Nowotsch, Sascha Uhrig, and Michael Paulitsch. Contention-aware dynamic memory bandwidth isolation with predictability in cots multicores: An avionics case study. In 29th Euromicro Conference on Real-Time Systems (ECRTS 2017), volume 76 of Leibniz International Proceedings in Informatics (LIPIcs), pages 2:1–2:22, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[145] J. Kim, M. Yoon, R. Bradford, and L. Sha. Integrated modular avionics (ima) partition scheduling with conflict-free i/o for multicore avionics systems. In 2014 IEEE 38th Annual Computer Software and Applications Conference, pages 321–331, July 2014.

[146] B. Huber, C. El Salloum, and R. Obermaisser. Aresource management framework for mixed-criticality embedded systems. In 2008 34th Annual Conference of IEEE Industrial Electronics, pages 2425–2431, Nov 2008.

[147] M. S. Mollison, J. P. Erickson, J. H. Anderson, S. K. Baruah, and J. A. Scoredos. Mixed-criticality real-time scheduling for multicore systems. In 2010 10th IEEE International Conference on Computer and Information Technology, pages 1864–1871, June 2010.

[148] Y. D. Bock, J. Broeckhove, and P. Hellinckx. Hierarchical real-time multicore scheduling through virtualization: Asurvey. In 2015 10th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), pages 611–616, Nov 2015.

[149] C. E. Salloum, M. Elshuber, O. Hftberger, H. Isakovic, and A. Wasicek. The across mpsoc – a new generation of multi-core processors designed for safetycritical embedded systems. In 2012 15th Euromicro Conference on Digital System Design, pages 105–113, Sept 2012.

[150] S. Avramenko, S. Esposito, M. Violante, M. Sozzi, M. Traversone, M. Binello, and M. Terrone. An hybrid architecture for consolidating mixed criticality applications on multicore systems. In 2015 IEEE 21st International On-Line Testing Symposium (IOLTS), pages 26–29, July 2015.

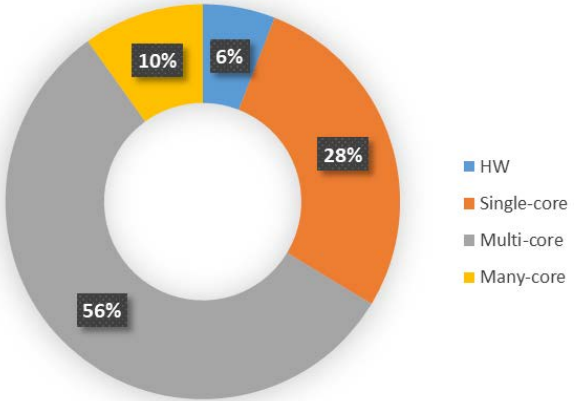
[151] S. Esposito, S. Avramenko, and M. Violante. On the consolidation of mixed criticalities applications on multicore architectures. In 2016 17th Latin-American Test Symposium (LATS), pages 57–62, April 2016.

[152] F. Federici, V. Muttillo, L. Pomante, G. Valente, D. Andreetti, and D. Pascucci. Implementing mixed-critical applications on next generation multicore aerospace platforms. In EMC2 Summit at CPS Week, CPS Week, 2016.

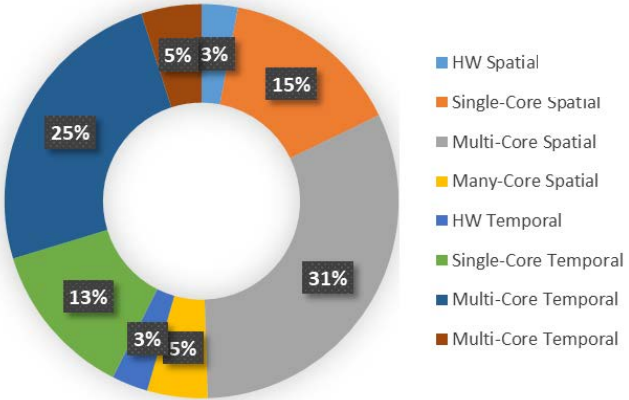
[153] P. Modica, A. Biondi, G. Buttazzo, and A. Patel. Supporting temporal and spatial isolation in a hypervisor for arm multicore platforms. In 2018 IEEE International Conference on Industrial Technology (ICIT), pages 1651–1657, Feb 2018.

[154] G. Cicero, A. Biondi, G. Buttazzo, and A. Patel. Reconciling security with virtualization: Adual-hypervisor design for arm trustzone. In 2018 IEEE International Conference on Industrial Technology (ICIT), pages 1628–1633, Feb 2018.

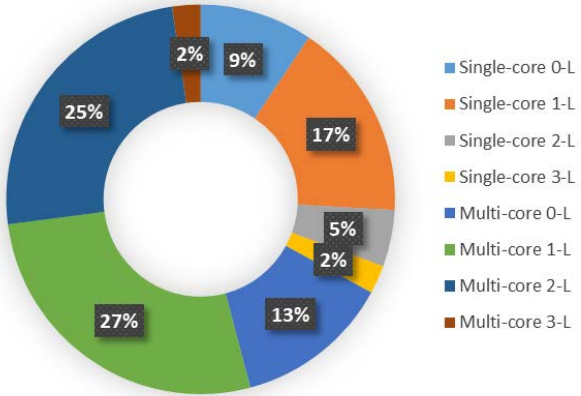
# MC Implementation Solutions (M-CIS)




## ISOLATION TECHNIQUES




## SCHEDULING LEVELS






Proceedings of CPS&IoT2019 page 461

MECO education & training

# 5.

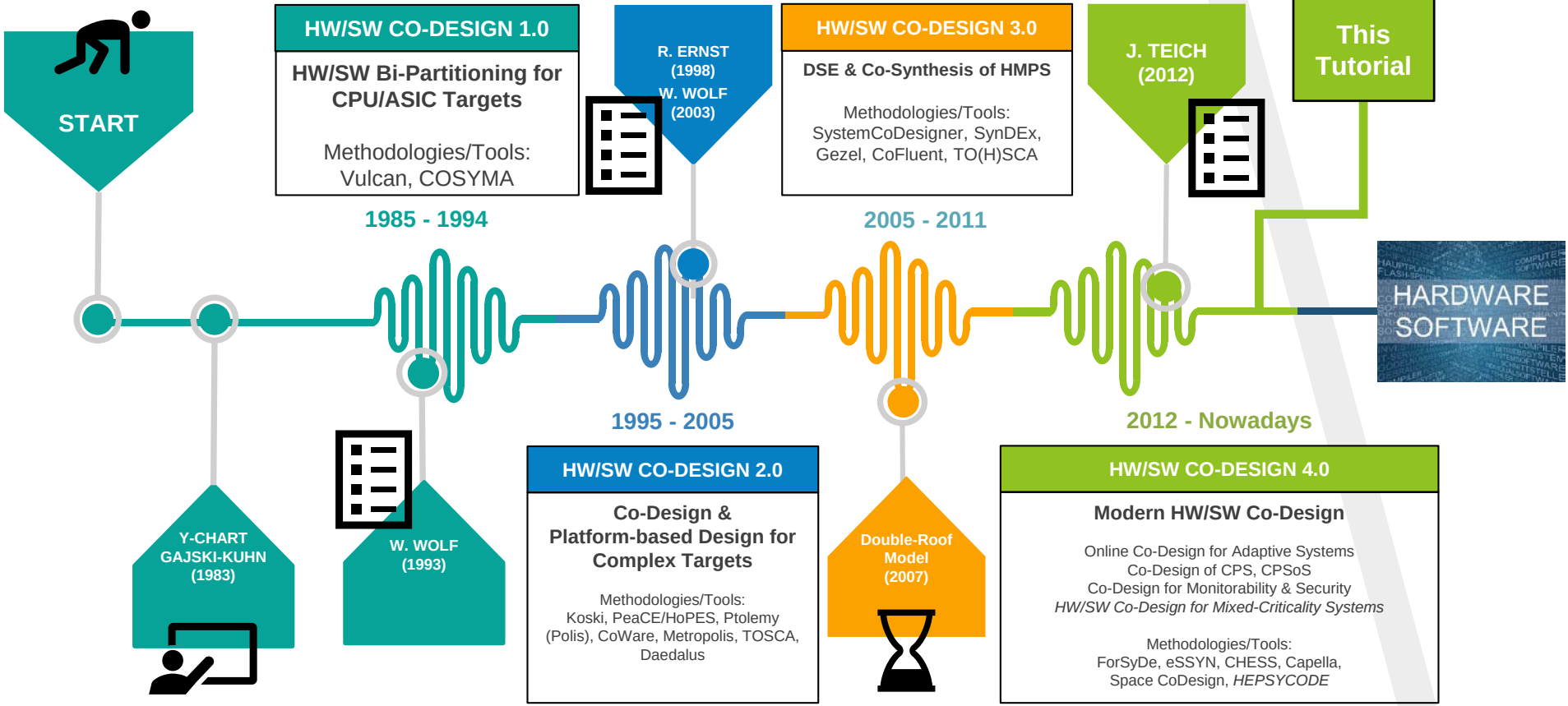
# Mixed-Criticality HW/ SW Co-Design

*"Multi-core and many-core computing platforms have to significantly improve system (and application) integration, efficiency and performance"*



Proceedings of CPS&IoT2019 page 462

# HW/ SW Co-Design Methodologies

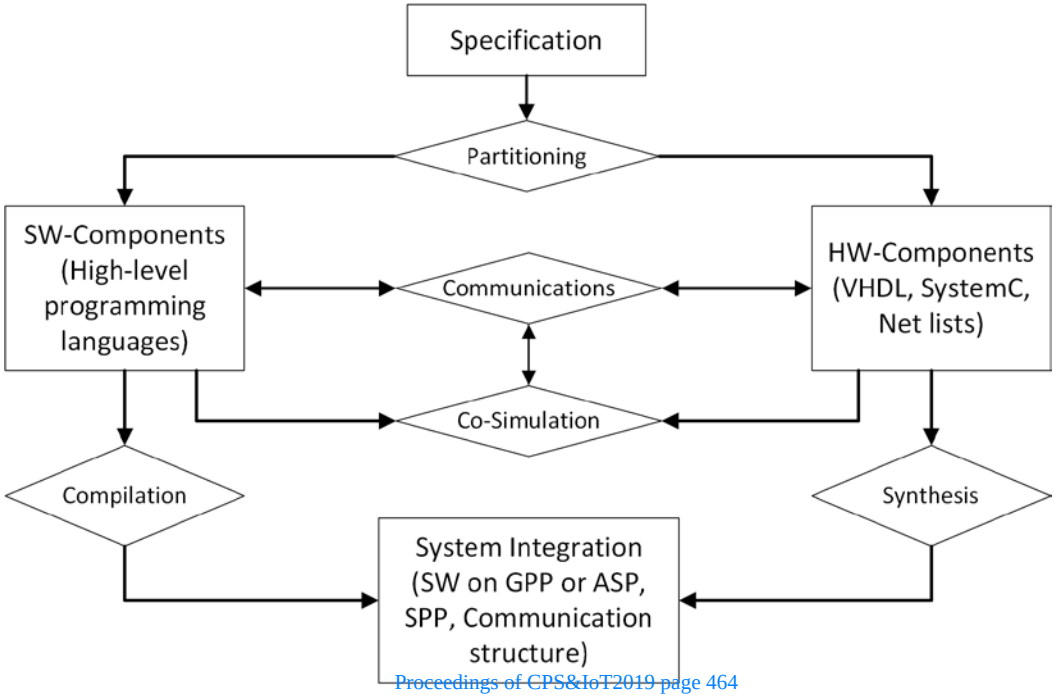





Proceedings of CPS&IoT2019 page 463

# Classical ELS HW/ SW Co-Design Approach

*Electronic System Level (ESL) is the utilization of appropriate abstractions in order to increase comprehension about a system, and to enhance the probability of a successful implementation of functionality in a cost-effective manner, targeting design methodologies for electronic digital HW/SW systems*






Proceedings of CPS&IoT2019 page 464

# Mixed-Criticality HW/ SW Co-Design

## Classification of different ESL Methodology Approaches (considering Mixed-Criticality Issues)

| | | Specification | | Specification | Implementation | | | Decision Making | | Refinement | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **ESL Approach** | **Gen.**[1] | Appl.[2] | Arch.[3] | **Language** | MoS[4] | MoP[5] | **DSE**[6] | Comp.[7] | Comm.[8] | Comp.[7] | Comm.[8] |
| AUTOFOCUS3 | 4.0 | PN | HeMPES | Custom | Component | TAPM | ● | ● | ○ | ● | - |
| CONTREP | 4.0 | UML, SDF | HeMPES | MARTE & SysML & SystemC | TLM | TAPM | ● | ● | ○ | ● | ○ |
| DeSyDe | 3.0/4.0 | SDF | HeMPSoC | XML | - | - | ● | ● | ○ | ● | ○ |
| Combined-DSE | 4.0 | CP | HeMPES | MiniZinc | TLM | TAPM | ● | ● | - | ● | - |
| OSSS-MC | 4.0 | OSSS/MC | HeMPSoC | SystemC | TLM | T/ISAPM | - | ● | - | ● | - |
| MultiPARTES | 4.0 | UML | HoMPES | MARTE | TLM | TAPM | ● | ● | - | ● | - |
| HEPSYCODE | 4.0 | CSP | HeMPES | SystemC | TLM | T/ISAPM | ● | ● | ● | ● | ○ |

[1] Gen.: HW/SW Co-Design Generation; [2] Appl.: Application Model; [3] Arch.: Architecture Model;
[4] MoS: Model of Structure; [5] MoP: Model of Performance; [6] DSE: Design Space Exploration;
[7] Comp.: Computation; [8] Comm.: Communication; CP: Constraint Programming;
PN: Process Network; CSP: communicating sequential processes; UML: Unified Modeling Language; SDF: Synchronous Dataflow;
He/HoMPS: Heterogeneous/Homogeneous Multi-processor Systems;
He/HoMPSoC: Heterogeneous/Homogeneous Multi-processor System on chip;
He/HoMPES: Heterogeneous/Homogeneous Multi-processor Embedded System;
TLM: Transaction-Level Model;
TAPM: Task Accurate Performance Model; ISAPM: Instruction Set Accurate Performance Model; CAPM: Cycle-Accurate Performance Model;



Proceedings of CPS&IoT2019 page 465

# 6.

# Proposed Methodology

"You will never strike oil by drilling through the map! - *Solomon Wolf Golomb*"

# Proposed HW/ SW Co-Design Methodology

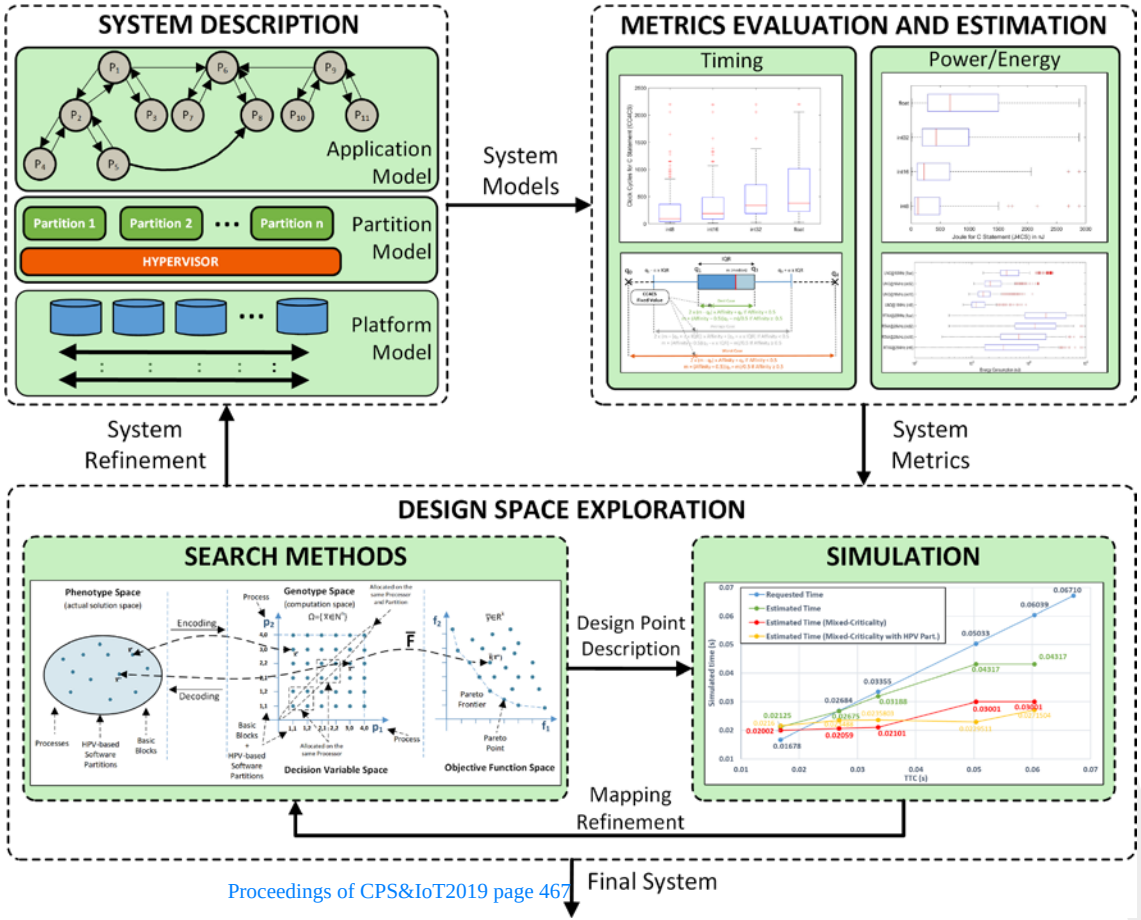
DEFINITION OF A GENERAL METHODOLOGY ABLE TO ABSTRACT CLASSICAL SYSTEM DESIGN FLOW (APPLICABLE TO DIFFERENT HW/SW CO-DESIGN FLOW)

***System Description***: Introduction of a partition layer to model HPV SW partitions

***Metrics Evaluation and Estimation***: Definition of different metrics (with related benchmarking arctivites in order to extract as-much-as-possible system informations)

***Search Methods:*** Meta-heuristic algorithm refinement (GA improvements)

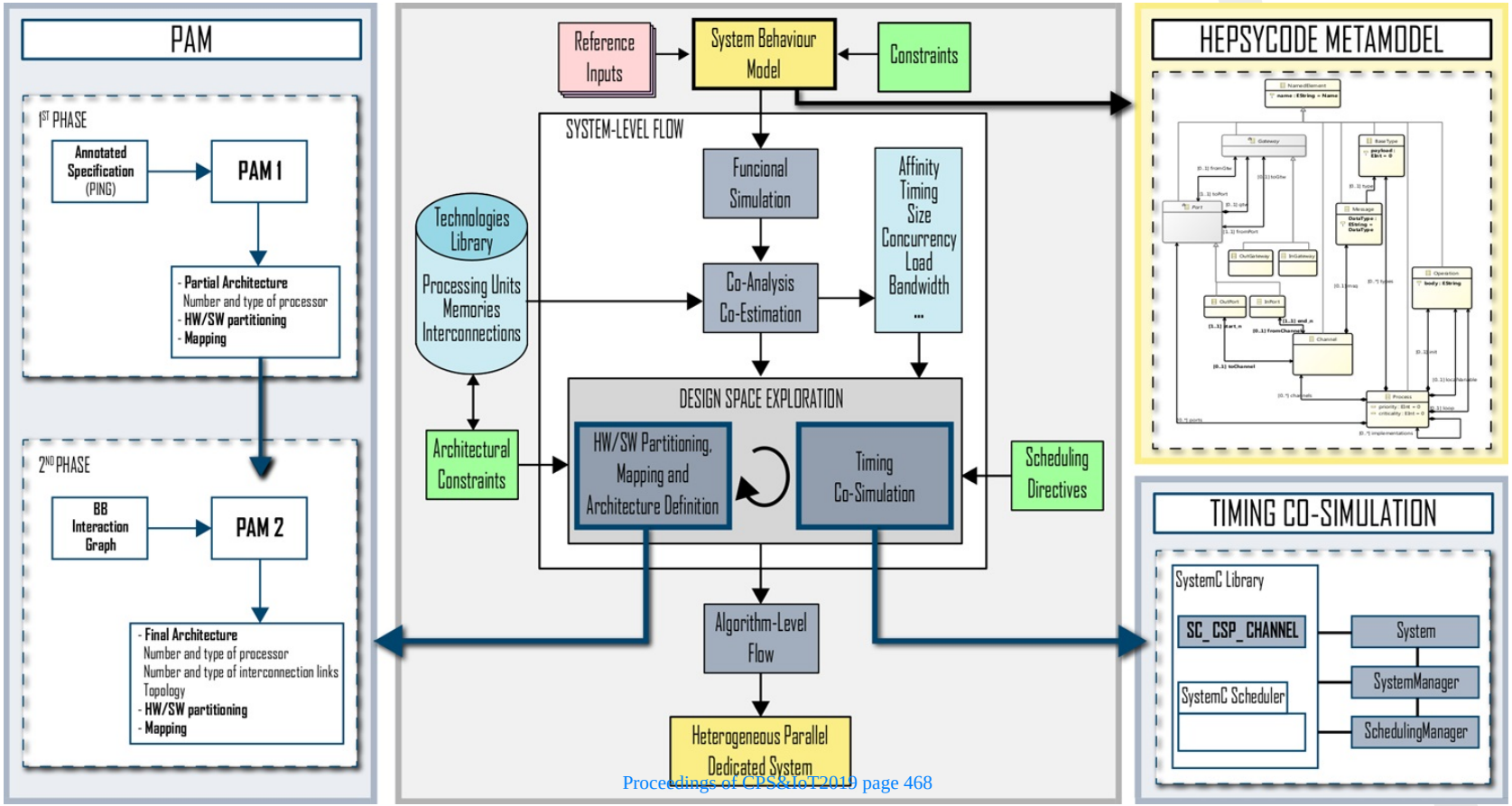
***Timing Simulator:*** improvement introducing Hierarchical scheduling feature






Proceedings of CPS&IoT2019 page 467

# HEPSYCODE HW/ SW Co-Design Flow



# HW/ SW Co-Design Framework

Definition of a general framework able to automate system design flow (Implemented using different SW technologies)

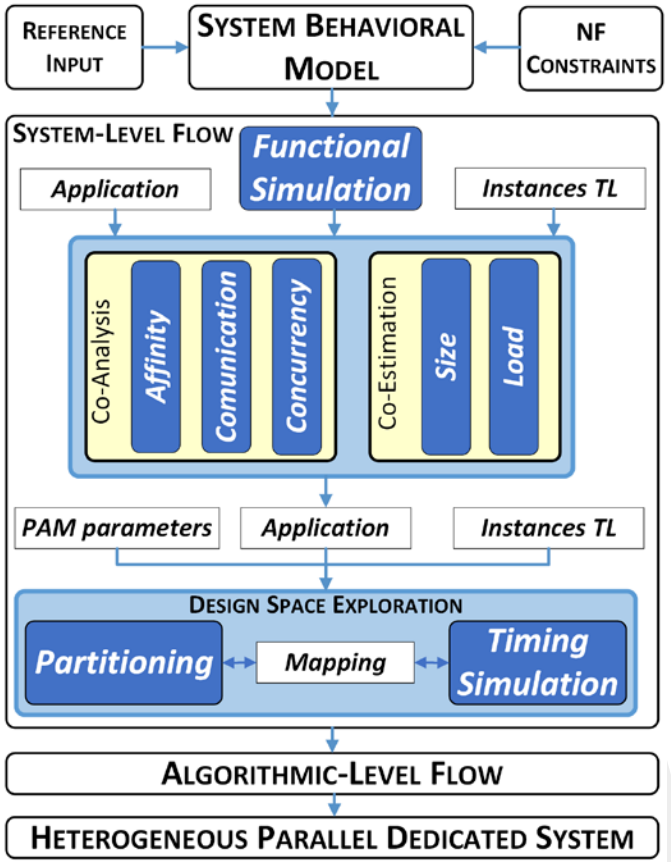
**System Behavioral model:** Realization of a GUI to model application using the specification language defined in the system behavioral specification step

**Functional Simulation:** automatic generation of a SystemC code implementing Hoare's CSP model of Computation from GUI

**Co-Analysis&Co-Estimation:** definition of a extensible activity step to evaluate system metrics:

- Affinity
- Concurrency
- Communication
- Size
- Load
- Power (WIP)

**DSE:** implementation of an automatic (extensible) DSE to make analysis and propose solutions in an HW/SW Co-simulation environment (**HEPSIM**)






Proceedings of CPS&IoT2019 page 469

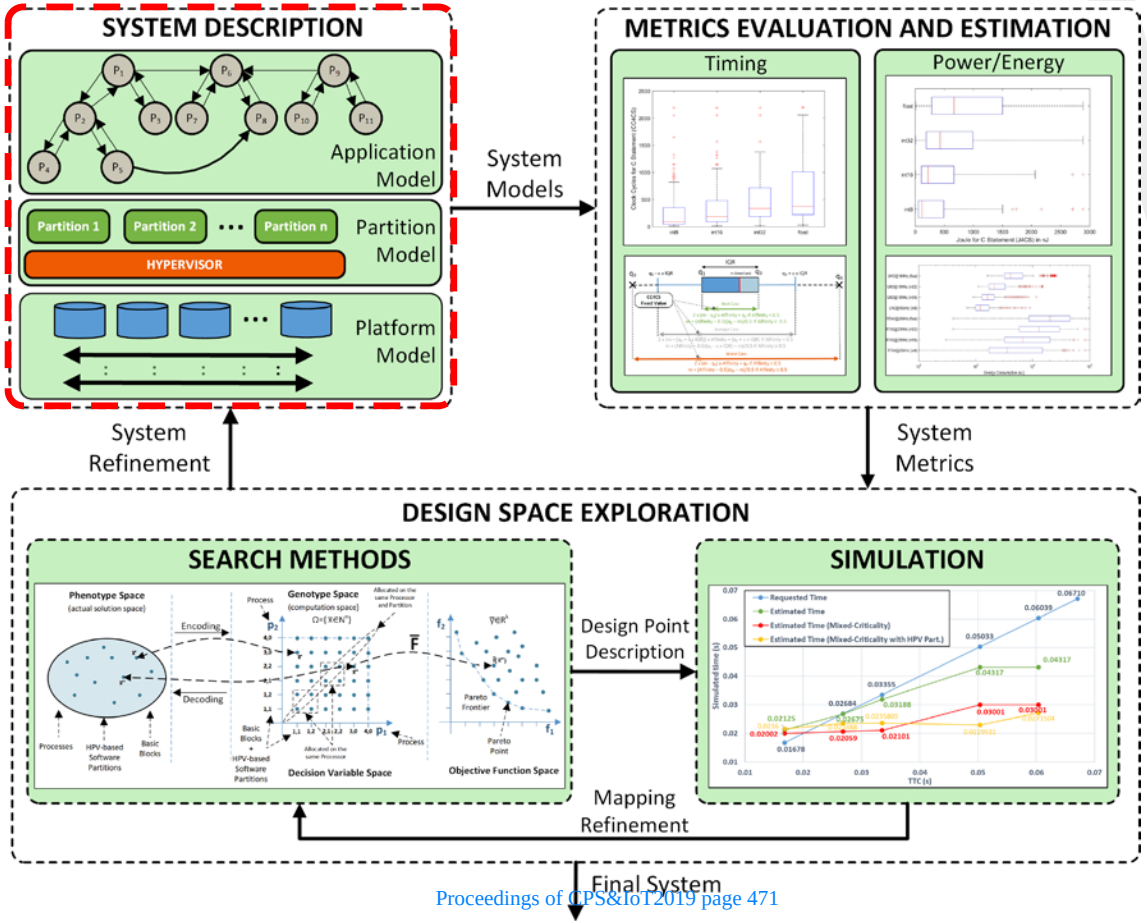
MECO education & training

# 7.

# ESL Methodology Main Elements

"All models are wrong but some are useful- *George E P. Box*"

# Proposed HW/ SW Co-Design Methodology



# Concurrency

- Concurrency is the **decomposability property** of a program, algorithm, or problem into order-independent or partially-ordered components or units.
- Even if the concurrent units of the program, algorithm, or problem are executed out-of-order or in partial order, **the final outcome will remain the same**. This allows for parallel execution of the concurrent units, which can significantly improve overall speed of the execution in multi-processor and multi-core systems.
- A number of **mathematical models** have been developed for general concurrent computation (Petri nets, process calculi, the Parallel Random Access Machine model, the Actor model etc.).



Proceedings of CPS&IoT2019 page 472

# Model Of Computation

- A *Model of Computation (MoC)* is a set of operational elements used to describe the behavior of an application (or a system). The set of operational elements and the set of relations among them are called the semantics of a MoC.
- MoC can be classified into Timed or Untimed, when introducing a totally or partially ordered events respectively.
  - **Untimed MoC:**
    - **Rendezvous of Sequential Processes**: applications are modeled with sequential processes that reach a particular point at which they have to synchronize each other (i.e., CSP by Hoare, 1975).
    - **Kahn Process Networks**: a process network where processes communicates using channels, which are unbounded point-to-point FIFO queues, sending fixed amount of data, called tokens
    - **Dataflow**: a special case of Kahn process networks, where processes (called actors) consume data exchanged between channels with a fixed firing rate



Proceedings of CPS&IoT2019 page 473

MECO education & training

# Process Calculi and CSP

- Process Calculi (or Process Algebras) are a diverse family of related approaches for formally modelling concurrent systems. Process calculi provide a tool for the high-level description of interactions, communications, and synchronizations between a collection of independent agents or processes. They also provide algebric laws that allow process descriptions to be manipulated and analyzed, and permit formal reasoning about equivalences between processes (e.g., using bisimulation).

- Process Calculi include the Communicating Sequential Processes (CSP), the Calculus of Communicating Systems (CCS), the Algebra of Communicating Processes (ACP) and so on.

- CSP is based on message passing via channels and was highly influential in the design of the OCCAM programming language.

- CSP was first described in a 1978 paper by Tony Hoare [8], but has since evolved substantially. CSP has been practically applied in industry as a tool for specifying and verifying the concurrent aspects of a variety of different systems as well as a secure ecommerce system. The theory of CSP itself is also still the subject of active research, including work to increase its range of practical applicability (e.g., increasing the scale of the systems that can be tractably analyzed).

# MECO education & training Modelling Language

- The system behavior modeling language introduced in *HEPSYCODE*, named **HML** (**HEPSY Modeling Language**), is based on the well-known **Communicating Sequential Processes** (**CSP**) Model of Computation (MoC)

- By means of HML it is possible to specify the System Behavior Model (SBM)

*SBM = {PS, CH} is a CSP-based executable/simulatable model of the system behaviour based on a Concurrent Processes MoC that explicitly defines also a model of communication) among processes (PS) using unidirectional point-to-point blocking channels (CH) for data exchange (i.e. CSP channels).*

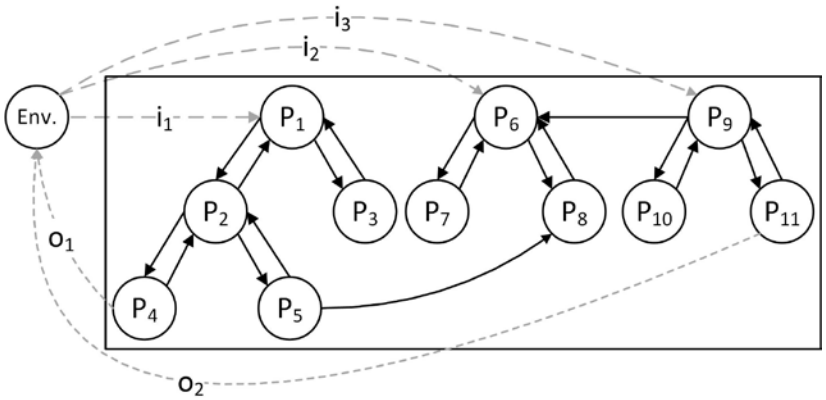
*PS = {$ps_1$, $ps_2$, .. , $ps_n$} is a set of concurrent processes that communicate each others exclusively by means of channels and use only local variables. Each process is described by means of a sequence of statements (an init section followed by a neverending loop) by using a suitable modeling language. Each process can have a priority p: 1(lower) to 100 (higher) imposed by the designer*

*CH = {$ch_1$, $ch_2$, .. , $ch_n$} is a set of channels where each channel is characterized by source and destination processes, and some details (i.e. size, type) about transferred data. Each channel can have also a priority p: 1(lower) to 100 (higher) imposed by the designer*

# System Description Models

- HEPSYCODE Modeling Language (HML):
  - Process Network connected via synchronous channels

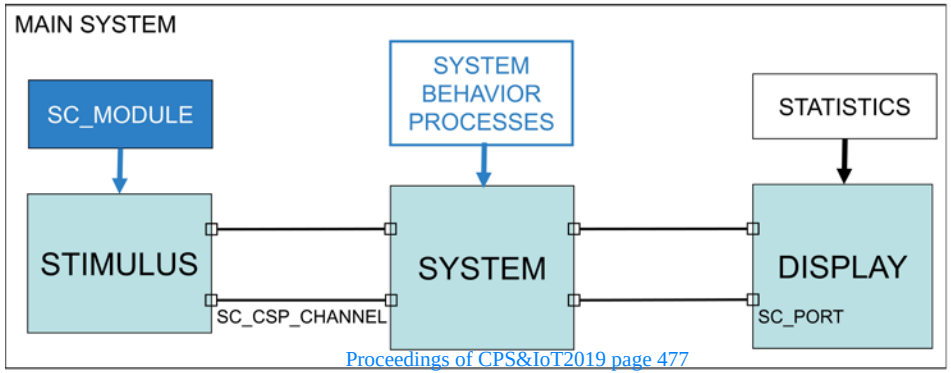
- G = {PS; CH} is the graph of the specification, where the graph nodes are the processes and the graph edges are the channel




- The initial HML model is then transformed into an executable SystemC model based on the Communicating Sequential Processes (CSP) Model of Computation (MoC)

# HEPSYCODE Functional Language

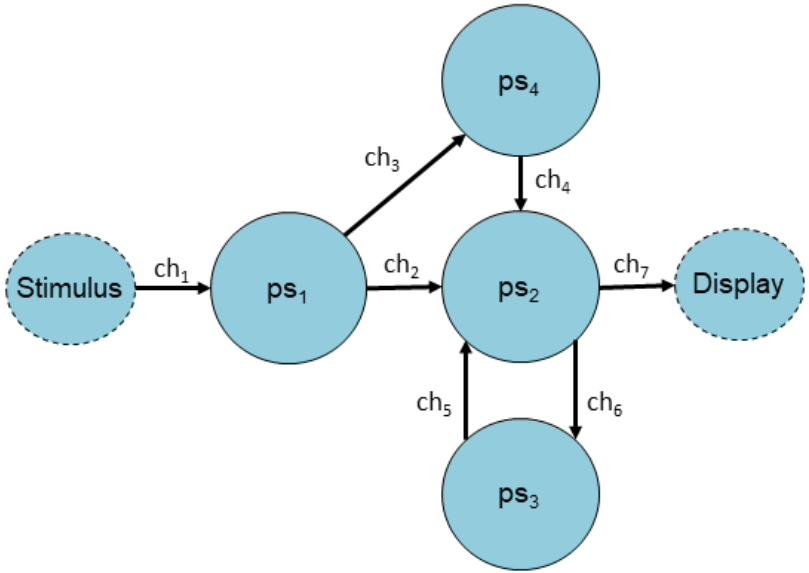
- HEPSYCODE Functional Language:
  - Reference languages is the SystemC, C++ class library able to capture and define system specifications
  - CSP processes are modelled by exploiting basic SC_THREAD (implemented as an infinite loop with an initialization step)
  - CSP channels have been modeled by introducing a proper SC_CSP_CHANNEL in the SystemC library.
  - System behavior is enclosed into a single SC MODULE, containing all the CSP processes and channels
  - Other SC MODULE and SC CSP CHANNEL are then used to model the Test-Bench and connected to the system by means of proper SC PORT




Proceedings of CPS&IoT2019 page 477

# System Behaviour

- An example of a possible SBM in shown in Figure, where the process PS = {$ps_1$, .. , $ps_4$} exchange data using channel CH = {$ch_1$, .. , $ch_7$}



# Constraints

- *Non–Functional Constraints*
  - ✓ Timing Constraints (TC)
    - **Time-To-Completion Constraint (TTC)**
  - ✓ Real-Time Constraints (RTC)
    - **Time-To-Reaction Constraint (TTR)**
  - ✓ Mixed-Criticality Constraints (MCC)
    - **Constraint in the DSE cost function**
  - ✓ 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)**



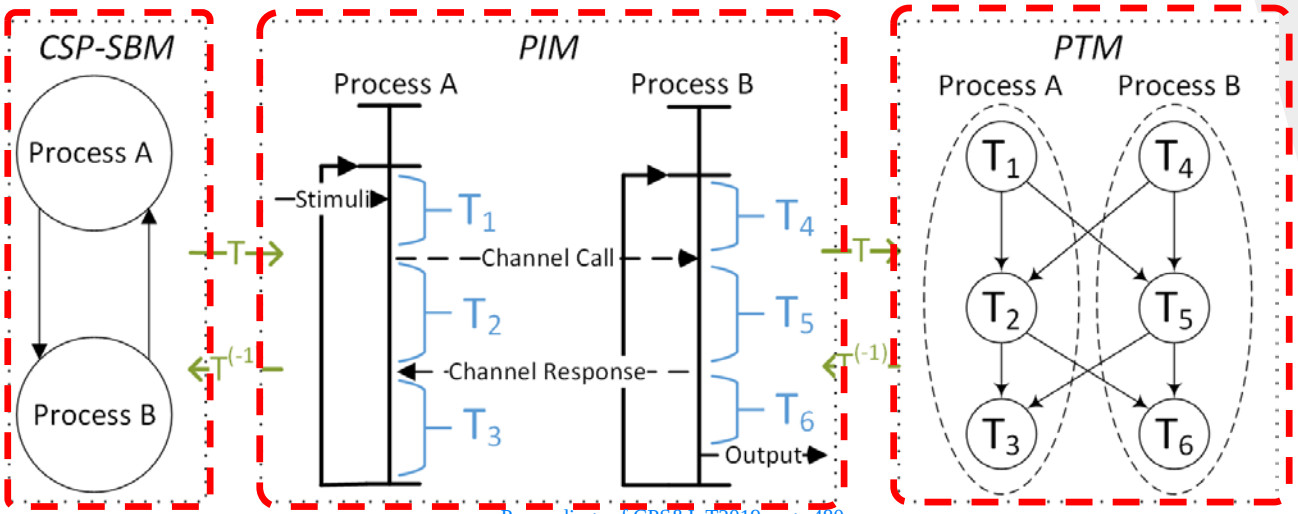
Proceedings of CPS&IoT2019 page 479

# SBM with Real-Time Constraints

**Communicating sequential processes (CSP) System Behavioral Model (CSP-SBM)**: network of concurrent processes model

**Process Implementation Model (PIM)**: split processes into several "dependent" tasks.

**Process Task Graph Model (PTM)**: Directed Acyclic Graph Task model (data flow)

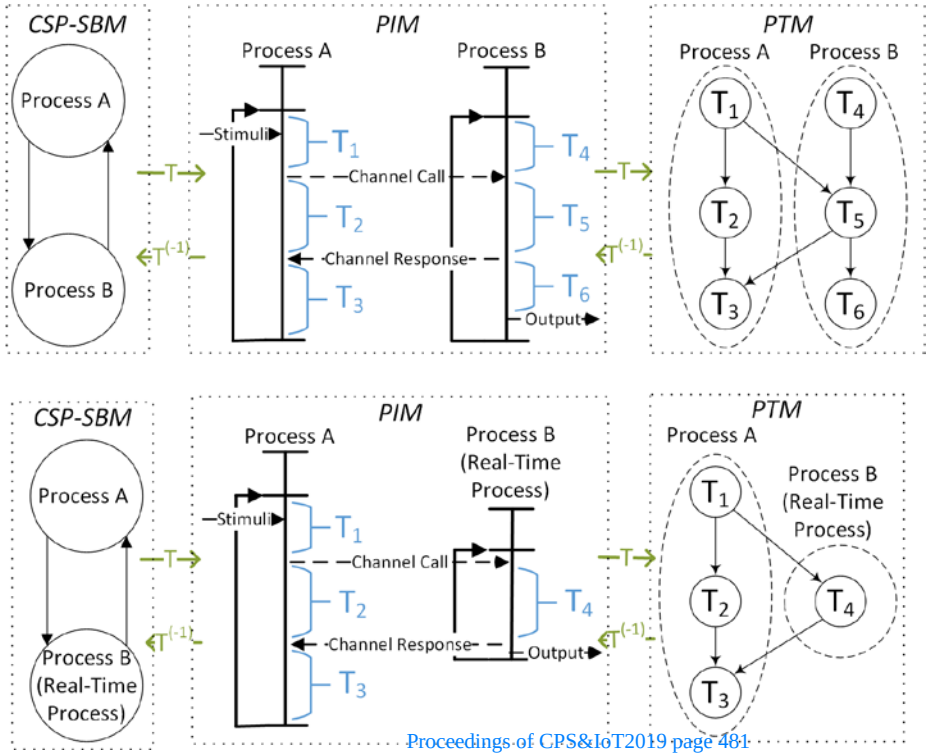





Proceedings of CPS&IoT2019 page 480

# SBM with Real-Time Constraints

- With respect to the SBM model, it is now possible to identify two class of CSP processes: *classical CSP process* and *real-time CSP processes*



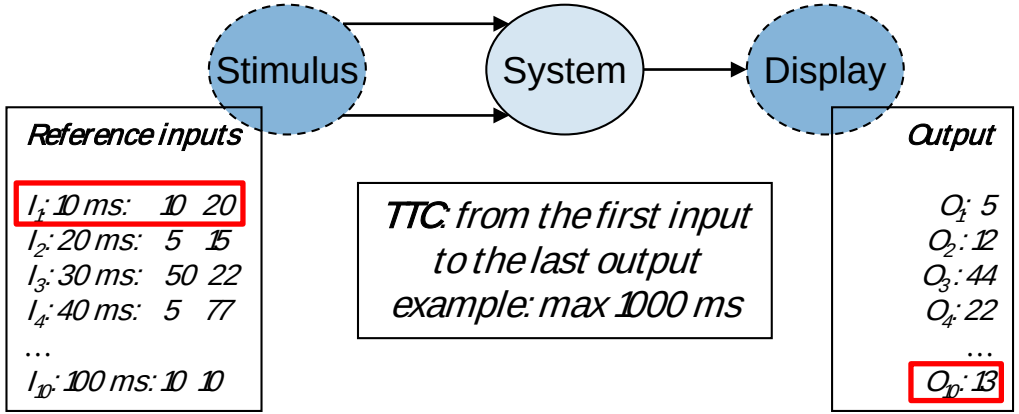


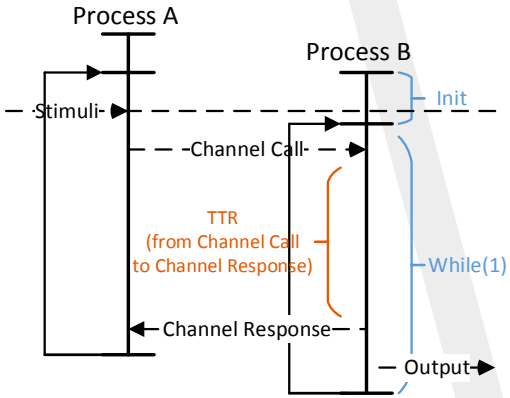

Proceedings of CPS&IoT2019 page 481

# Timing Constraints

- ***Time-to-Completion (TTC)**: time available to complete the SBM execution from the first input trigger to complete output generation. This constrain should be satisfied by each ($input_i$, $output_i$) couple.*

- ***Time-to-Reaction (TTR)**: real-time constraints related to the time available for the execution of leaf CSP processes (i.e. the time available to execute the statements inside the input/output pair that delimits the never-ending loop of a CSP process). This constrain should be satisfied by each input and output*






Proceedings of CPS&IoT2019 page 482

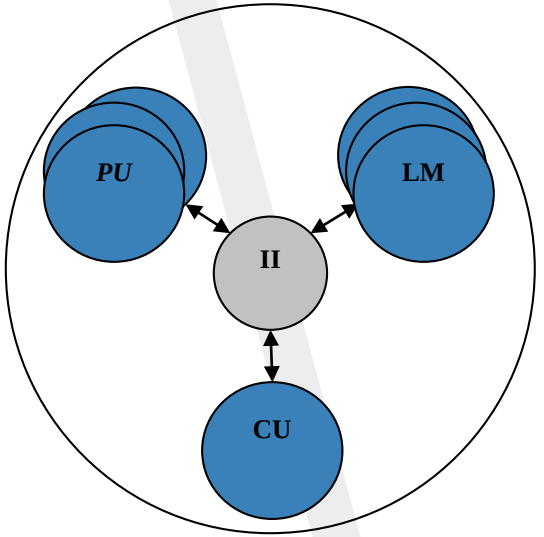
# Target Architecture

- 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.

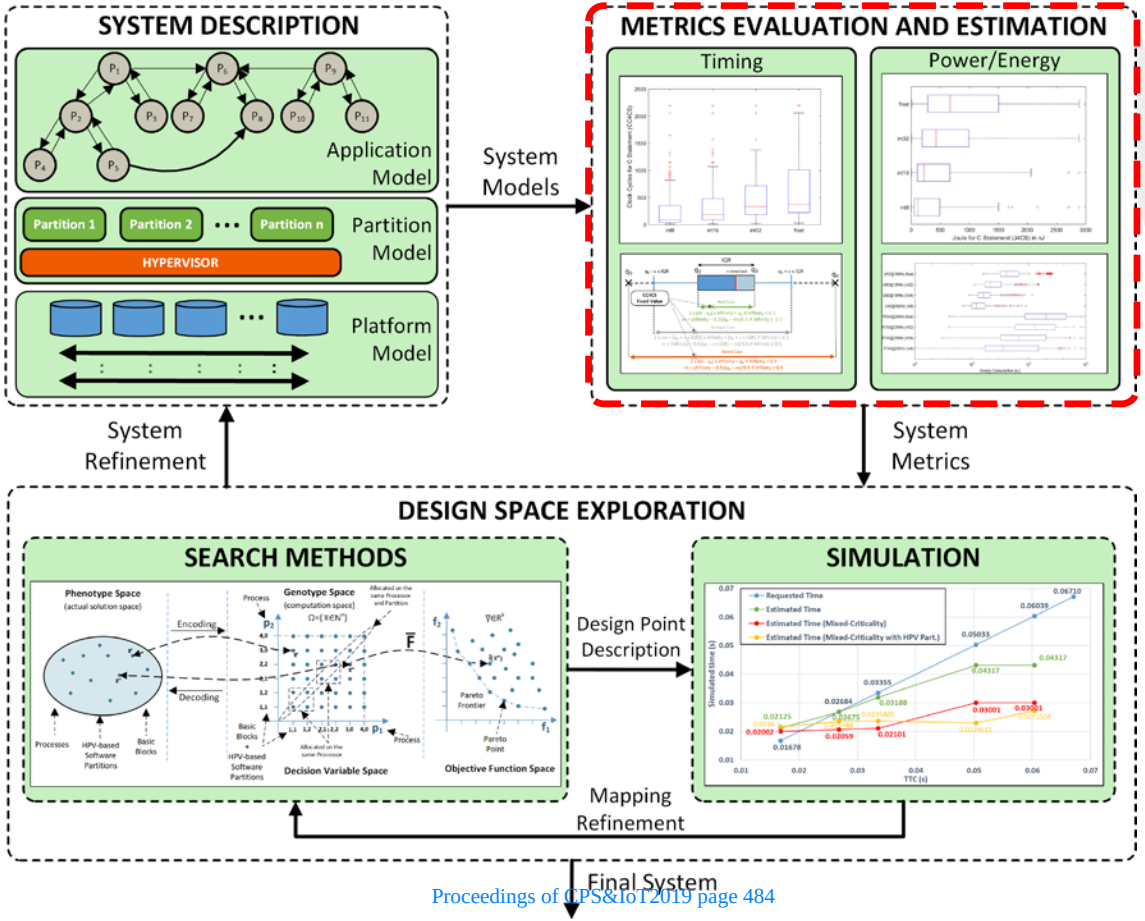
- $TL = \{PU, MU, EIL\}$, where $PU = \{pu_1, pu_2, .., pu_p\}$ is a set of *Processing Units*, $MU = \{mu_1, mu_2, .., mu_m\}$ is a set of *Memory Units* and $EIL = \{il_1, il_2, .., il_c\}$ is a set of *External Interconnection Links*.

- 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.



# Proposed HW/ SW Co-Design Methodology



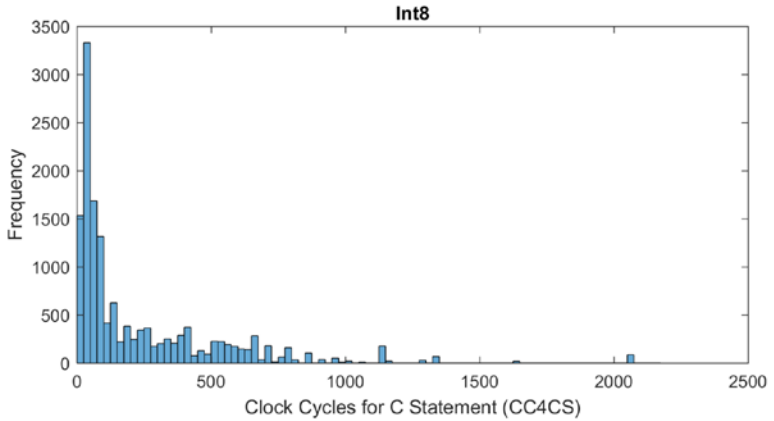
# HEPSYCODE Metrics

**CC4CS**: early stage metric to estimate in a HW/SW unified view process execution time

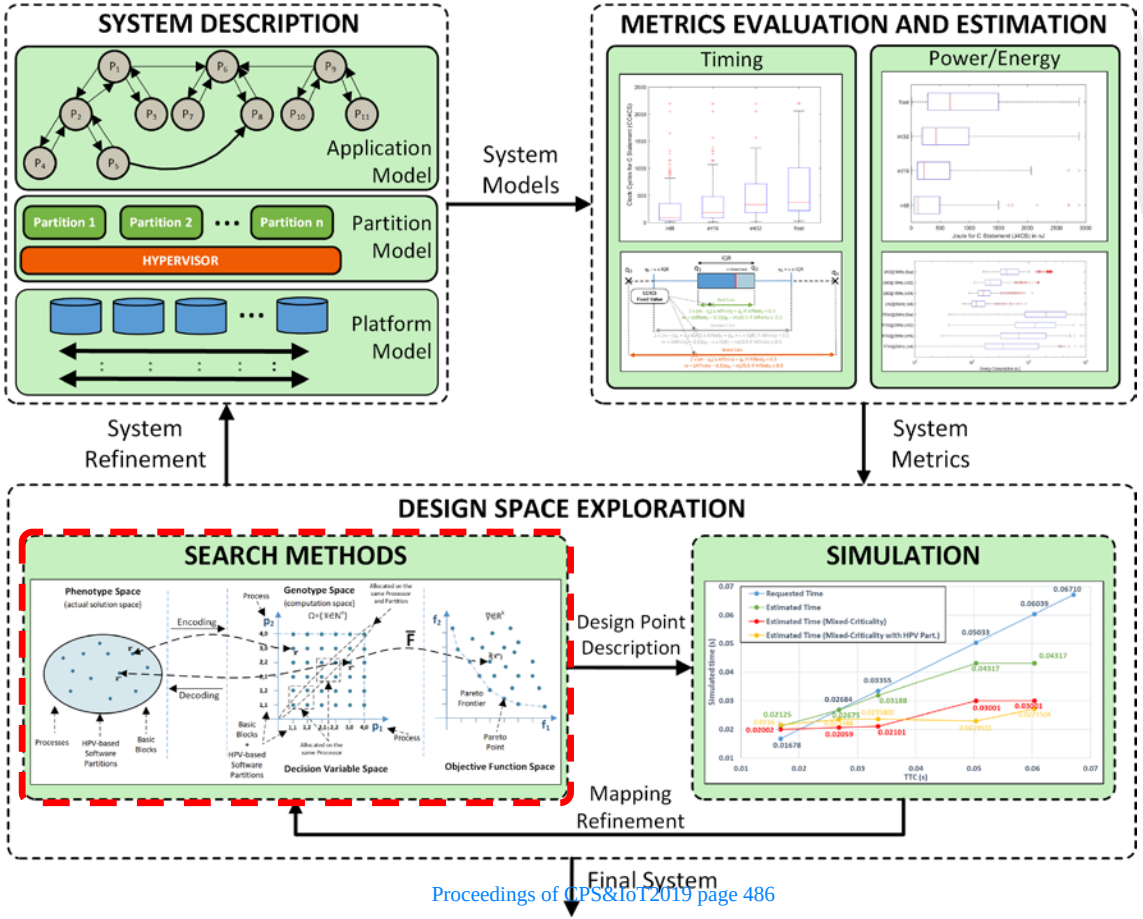
**Statistical Analisys**: Evaluate metric accuracy

**S4CS** (HW/SW): size metric to evaluate software (in terms of bytes in RAM/ROM) and hardware (in terms of gate or LUT count) size (WIP)

Other WIP metrics: **Affinity, Power/Energy, Monitorability, Security**



# Proposed HW/ SW Co-Design Methodology

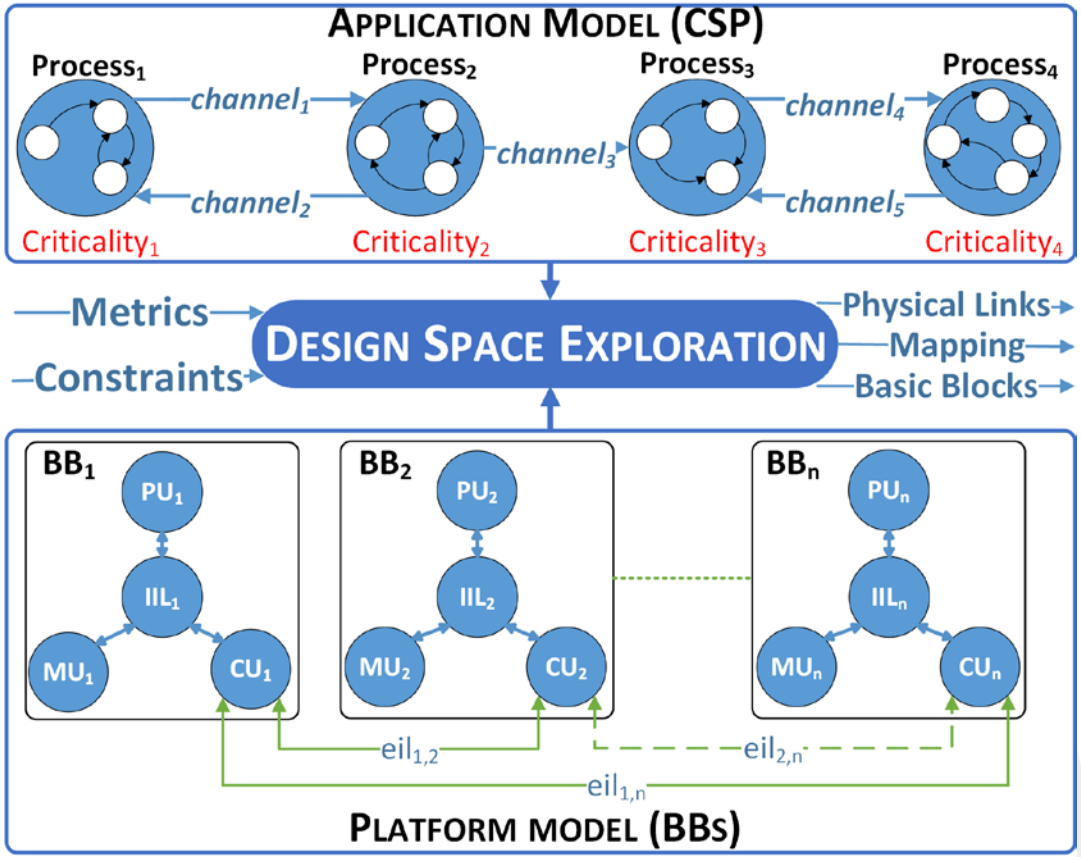





Proceedings of CPS&IoT2019 page 486

# HEPSYCODE Design Space Exploration

- INPUT:
  - **Application Model:** CSP model injected with safety requirements.
  - **Platform Model:** subset of HW solution (also in a multi-core scenario)
  - **Metrics:** results from the Evaluation&Estimation activity
  - **Constraints:** F/NF constraints (depending on application domain)
- OUTPUT:
  - **Physical Links:** Possible optimal links and topology.
  - **Mapping:** Process to BBs.
  - **Basic Blocks:** Processors, architecture and number of cores.

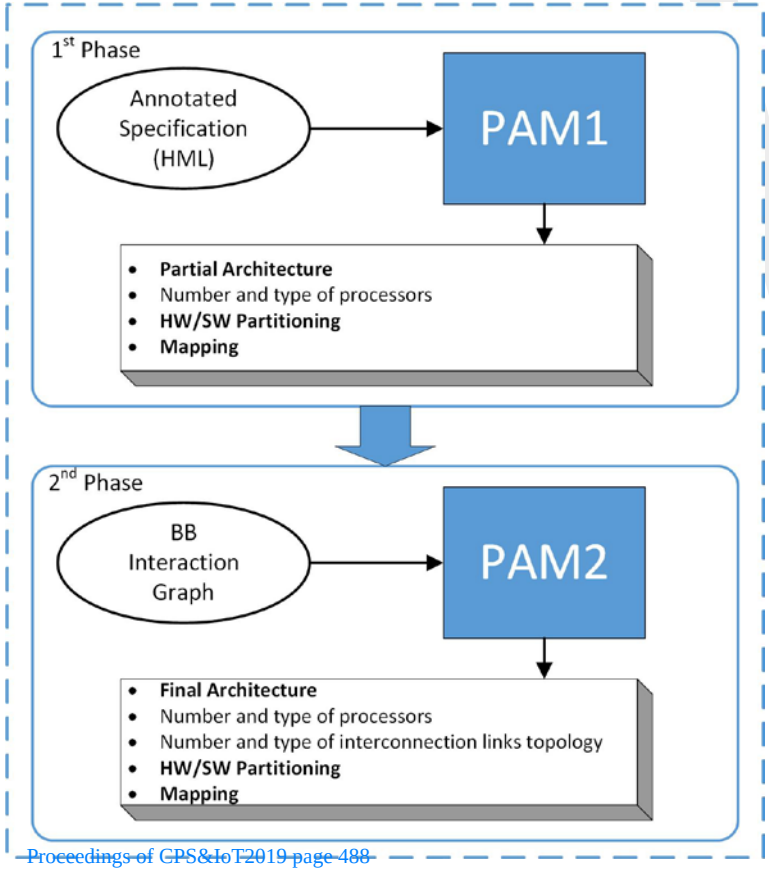





Proceedings of CPS&IoT2019 page 487

# HEPSYCODE Design Space Exploration

- INPUT:
  - **Application Model:** CSP model injected with safety requirements.
  - **Platform Model:** subset of HW solution (also in a multi-core scenario)
  - **Metrics:** results from the Evaluation&Estimation activity
  - **Constraints:** F/NF constraints (depending on application domain)
- OUTPUT:
  - **Physical Links:** Possible optimal links and topology.
  - **Mapping:** Process to BBs.
  - **Basic Blocks:** Processors, architecture and number of cores.



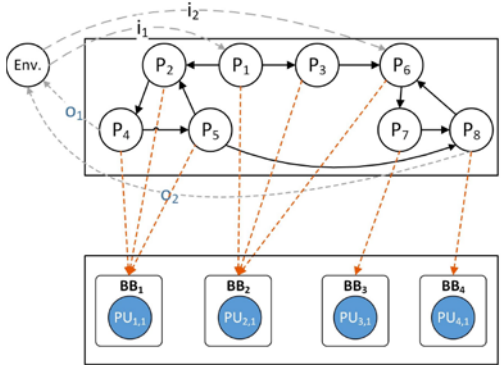

Proceedings of GPS&IoT2019 page 488

# HEPSYCODE Design Space Exploration

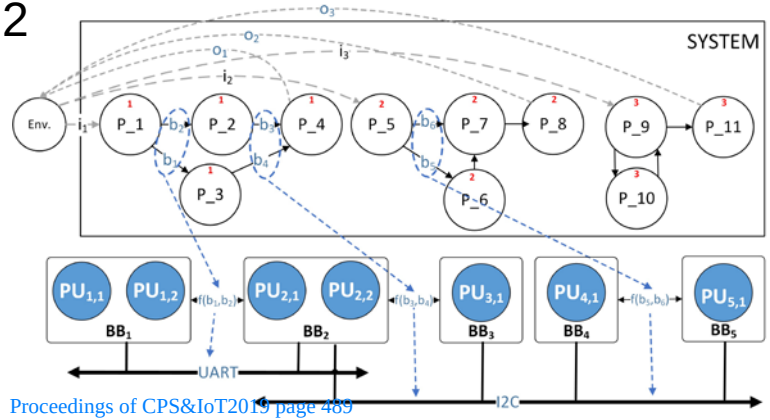
- INPUT:
  - **Application Model:** CSP model injected with safety requirements.
  - **Platform Model:** subset of HW solution (also in a multi-core scenario)
  - **Metrics:** results from the Evaluation&Estimation activity
  - **Constraints:** F/NF constraints (depending on application domain)

- OUTPUT:
  - **Physical Links:** Possible optimal links and topology.
  - **Mapping:** Process to BBs.
  - **Basic Blocks:** Processors, architecture and number of cores.

PAM1



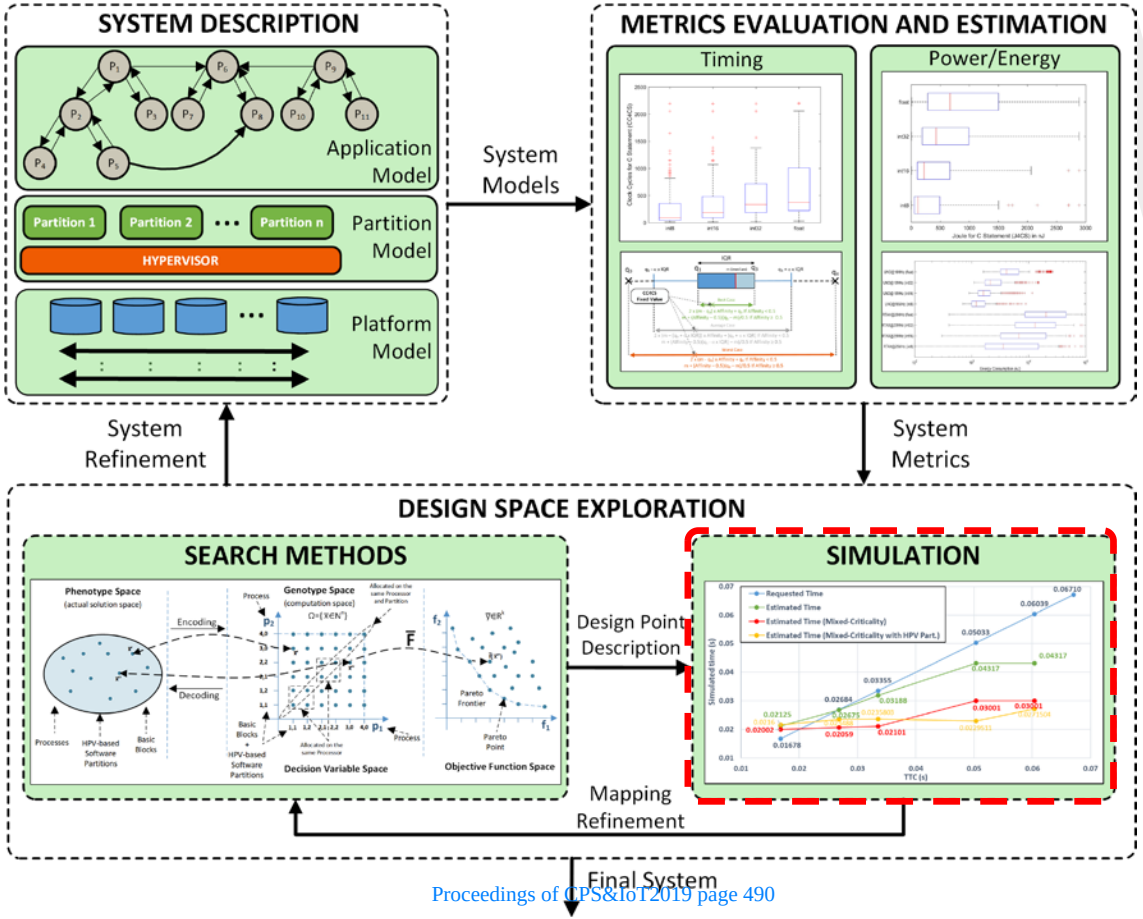

PAM2





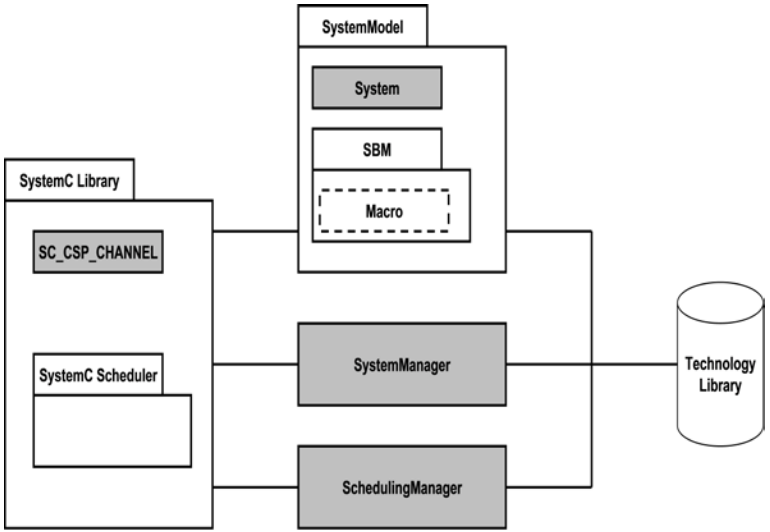

Proceedings of CPS&IoT2019 page 489

# Proposed HW/ SW Co-Design Methodology



# System Description Models

- HEPSIM (HEPSYCODE SIMulator), the extended SystemC simulator used for HW/SW Co-Simulations in HEPSYCODE:
  - The SystemC library has been extended with a SC_CSP_CHANNEL template class to implement the point-to-point CSP channel semantic

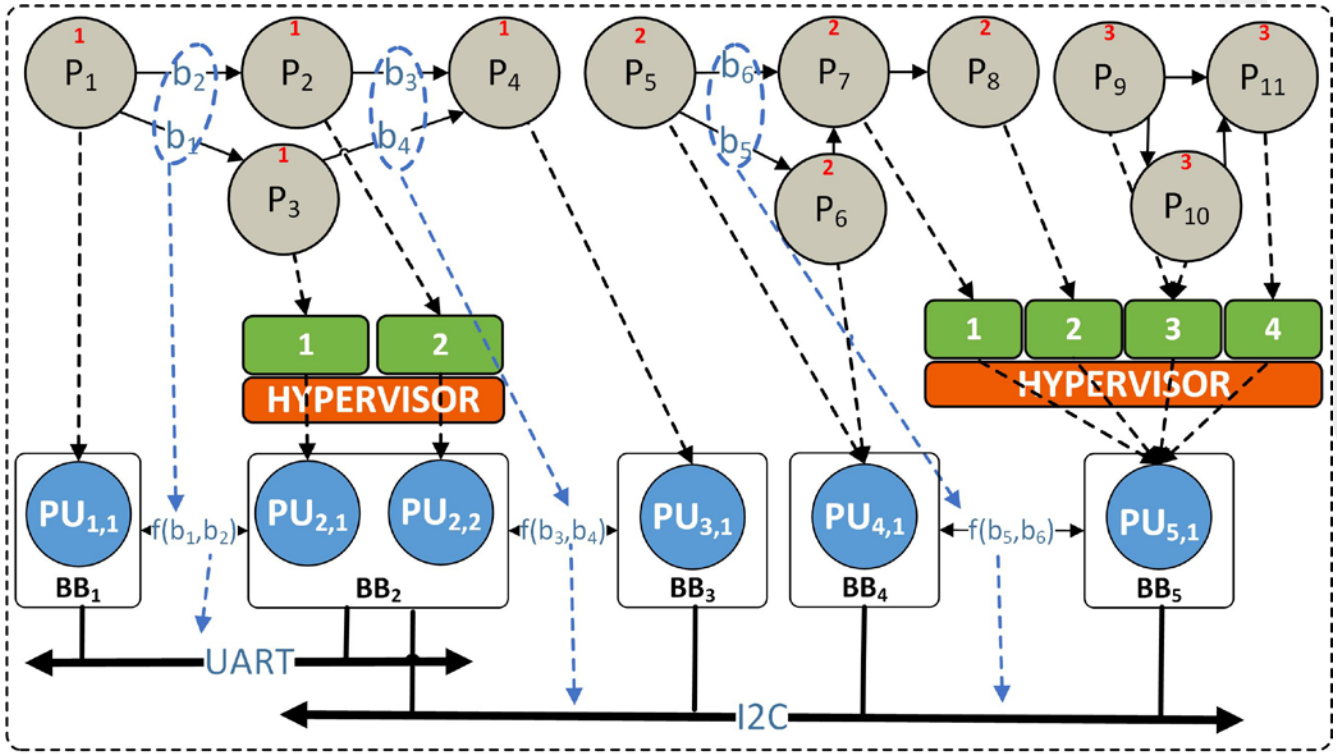



```
/* Macro S */
#define S(X) \
        pSystemManager->Increase(X); \
        if(!pSystemManager->checkSPP(X)) \
                wait(pSchedulingManager->schedule[X]);
.............
/* HEPSCHED */
if(ready[ps.id]==true){
        schedule[ps.id].notify(SC_ZERO_TIME);
        wait(release[ps.id]);
}
.............
/* Macro S */
        wait(pSystemManager->upSimTime(X)); \
        if(!pSystemManager->checkSPP(X)) \
                pSchedulingManager->release[X]
                .notify(SC_ZERO_TIME);
        #endif
.............
/* The handle goes to HEPSCHED */
```

- Implemented an Hierarchical Scheduling manager (2-Levels Scheduling)

# System Description Models

- Example of a Possible Deployment platform






Proceedings of CPS&IoT2019 page 492

# 8.

# HepsyCode-MC

"The fundamental issue with MCS is how to reconcile the differing needs of separation (for safety) and sharing (for efficient resource usage)"

# HEPSYCODE Multi-Objective Optimization Problem

**Multi-Objective Design Space Exploration Optimization Problem**

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

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

**Linearization of Multi-objective Design Space Exploration Optimization Problem**

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

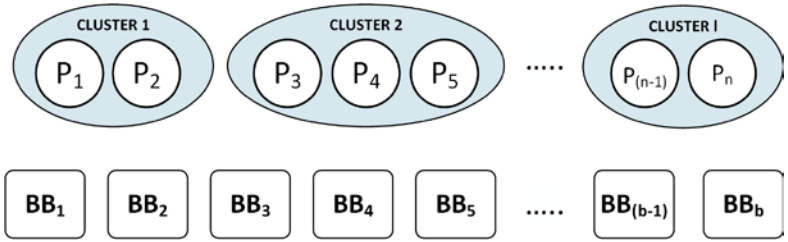
$U(\bar{x})$ is the utility function evaluated at each iteration of the GA for each individual $\bar{x} \in \Omega$. $f_k$ represents the value of the objective function (or metric) $k$ for each individual $\bar{x}$, while $\omega_k$ is the weight associated to each objective function or metric.



Proceedings of CPS&IoT2019 page 494

# HEPSYCODE Multi-Objective Optimization Problem

- The introduction of mixed-criticality requirements reduces the decision space due to the fact that applications with different criticality can not share the same HW block or the same SW partition
  - Introducing *l* criticality levels assigned to each process (with some kind of risk analysis driven by standards and certifications), the decision variable space (without HPV-based software partition) could be divided into different clusters




  - *Decision Variable Space Size (No HPV SW Partitions):* $P(b,l) = \frac{b!}{(b-l)!},\ l \leq b$

  - *r* processors able to support HPV, *t* processors not able to support HPV
    - Decision Variable Space Size: $P([t + r \cdot p_{max}], l) = \frac{[t + r \cdot p_{max}]!}{([t + r \cdot p_{max}] - l)!},\ l \leq [t + r \cdot p_{max}]$

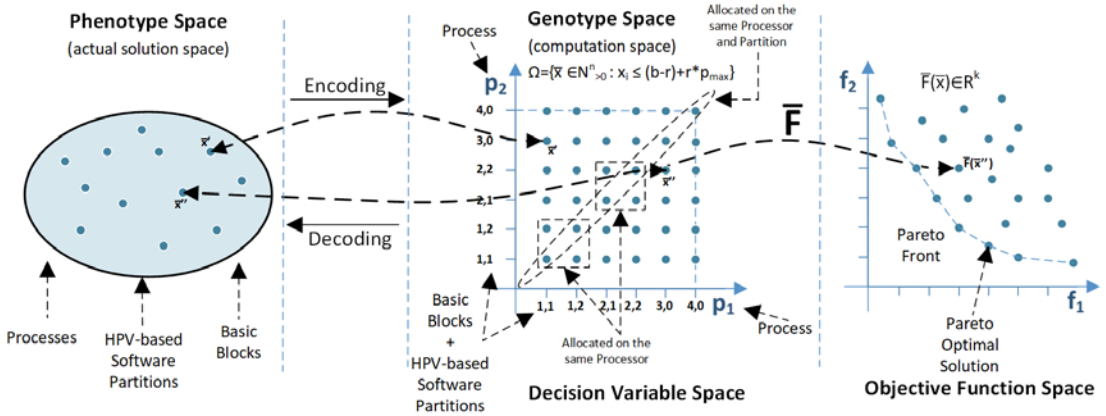


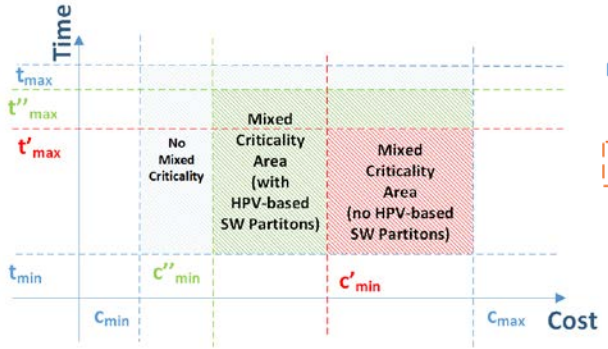
Proceedings of CPS&IoT2019 page 495

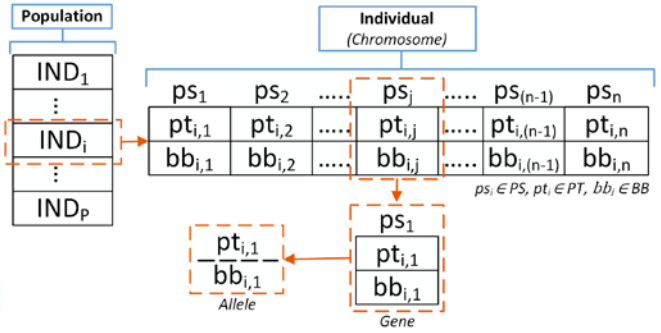
# HEPSYCODE Multi-Objective Optimization Problem

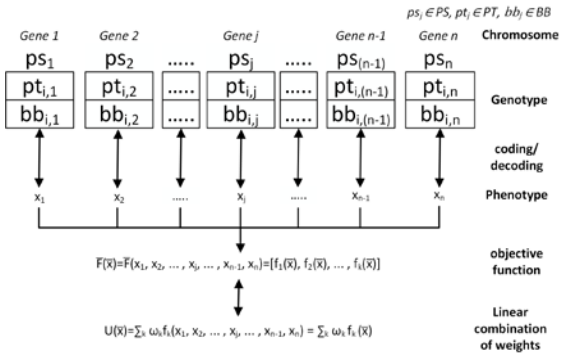
**Multi-objective optimization problem:** Introduction of HPV-based software partitions into the decision variable space.

**Pareto analysis with mixed-criticality constraints:** The introduction of Hypervisor technologies increase the number of feasible solutions decreasing the global cost.



# DSE Approach

## 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:

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

## 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:

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

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.



Proceedings of CPS&IoT2019 page 497

# 9.

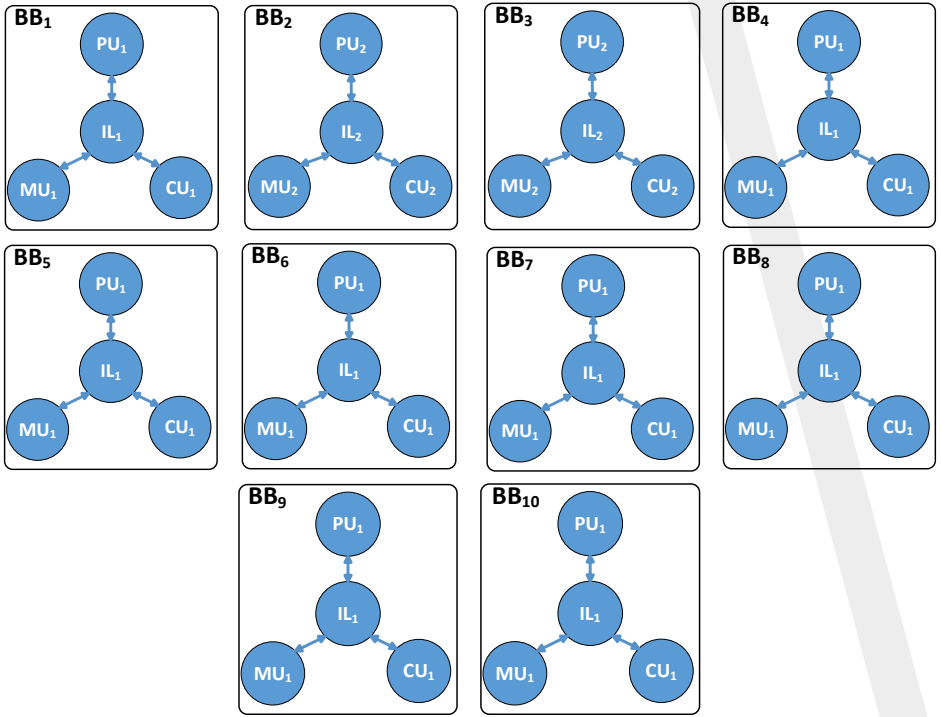
# Case Studies

"In the future, everyone will be world-famous for 15 minutes – Andy Warhol"

# TL Instances

## Processors

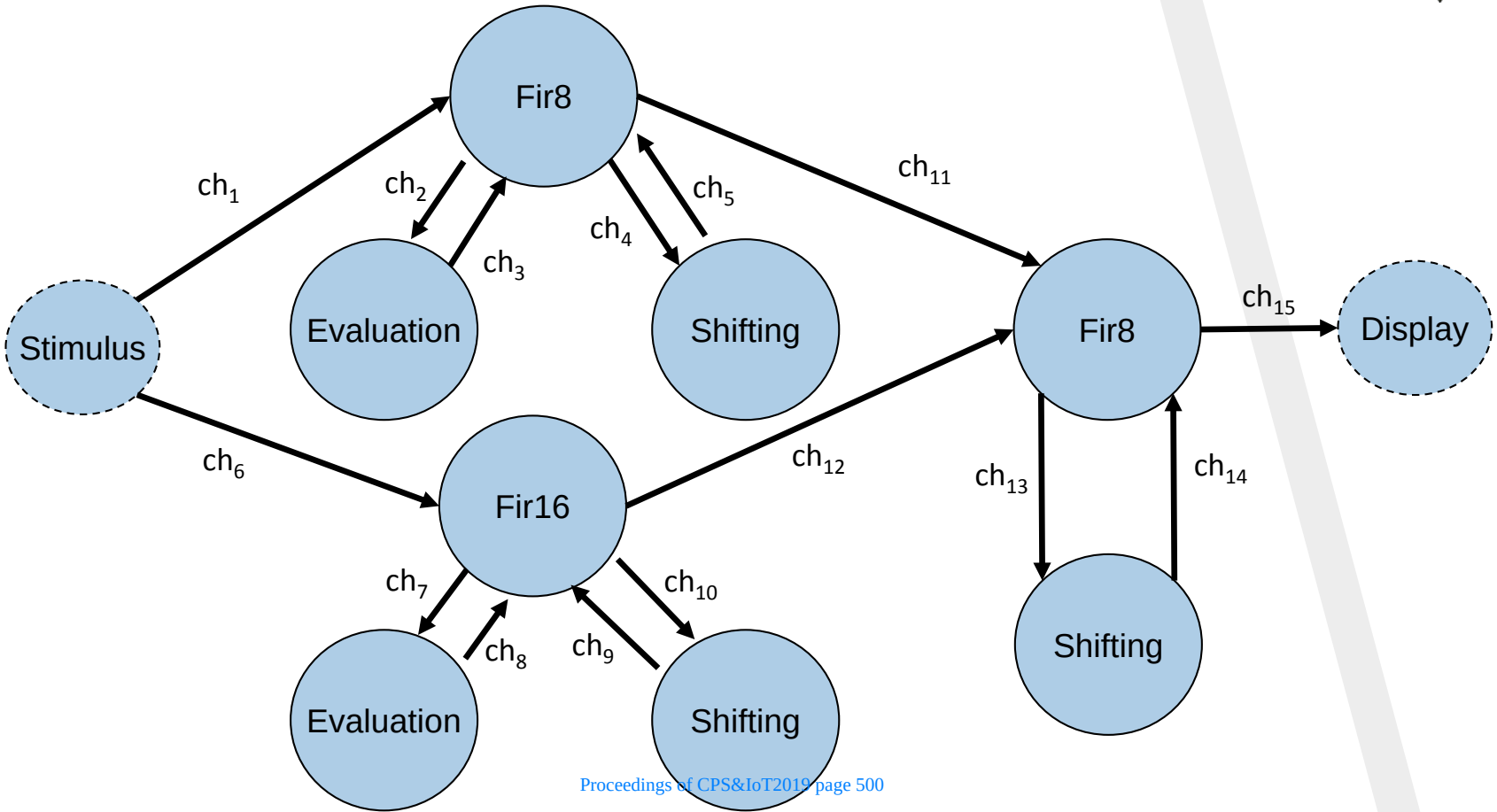
| | | |
|---|---|---|
| Intel **MPU 8051** with frequency 20 MHz | ID 0  | ID 1  |
| Microchip DSPIC or **PIC24** with frequency 20 MHz | ID 2  | ID 3  |
| Gaisler **LEON3** with frequency 75 MHz | ID 4   | ID 5   |
| Xilinx **Spartan3AN** with frequency 50 MHz | ID 6  | ID 7  |
| Xilinx **Virtex7** with frequency 250 MHz | ID 8   | ID 9   |






Proceedings of CPS&IoT2019 page 499

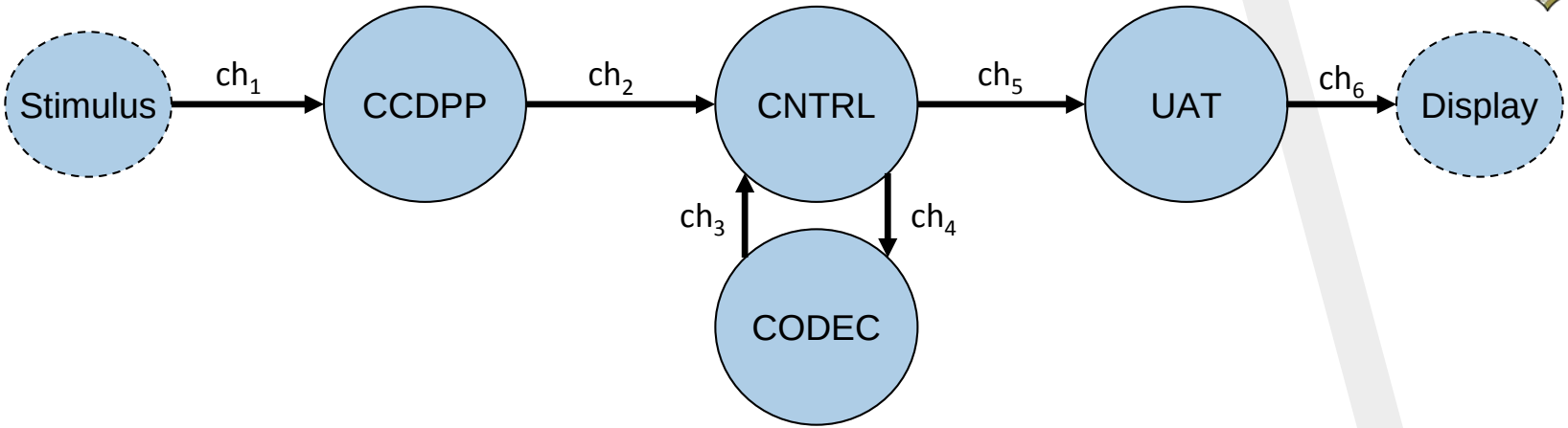
# Case Study 1: FirFirGCD





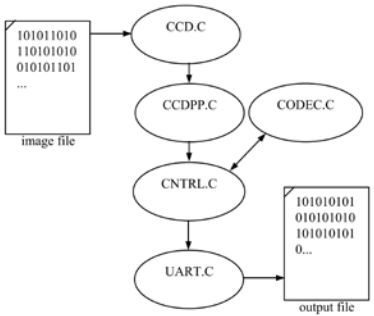

Proceedings of CPS&IoT2019 page 500

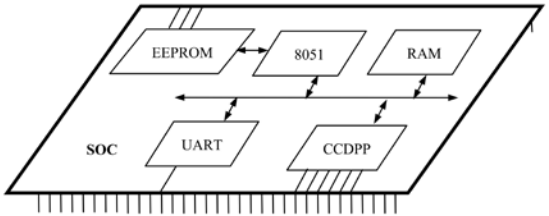
# Case Study 2: DigitalCam



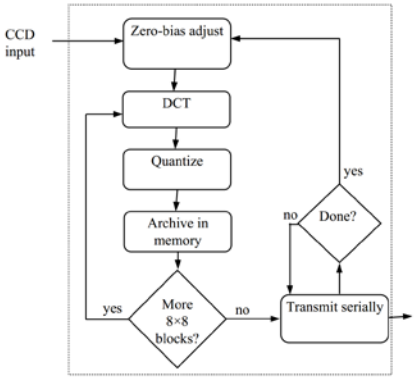

Embedded Systems Design: A Unified Hardware/Software Introduction: Digital Camera Example

Executable model of digital camera



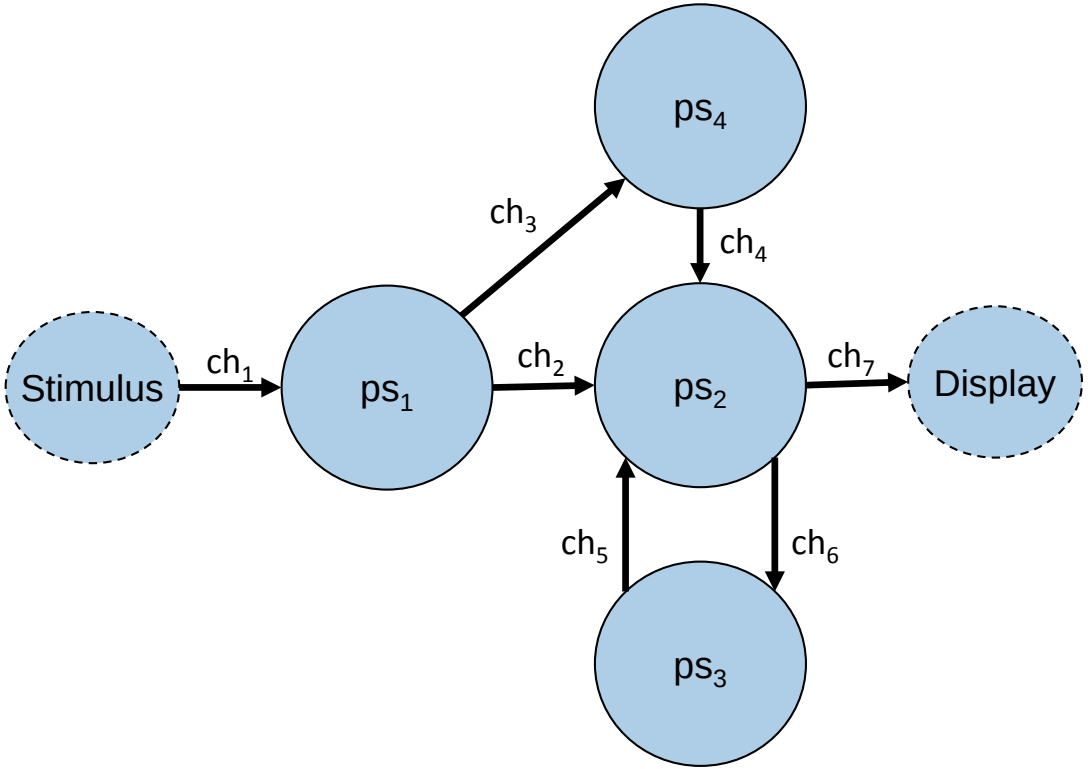




| | Implementation 2 | Implementation 3 | Implementation 4 |
|---|---|---|---|
| Performance (second) | 9.1 | 1.5 | 0.099 |
| Power (watt) | 0.033 | 0.033 | 0.040 |
| Size (gate) | 98,000 | 90,000 | 128,000 |
| Energy (joule) | 0.30 | 0.050 | 0.0040 |




Proceedings of CPS&IoT2019 page 501

# Case Study 3: Hepsy-RT

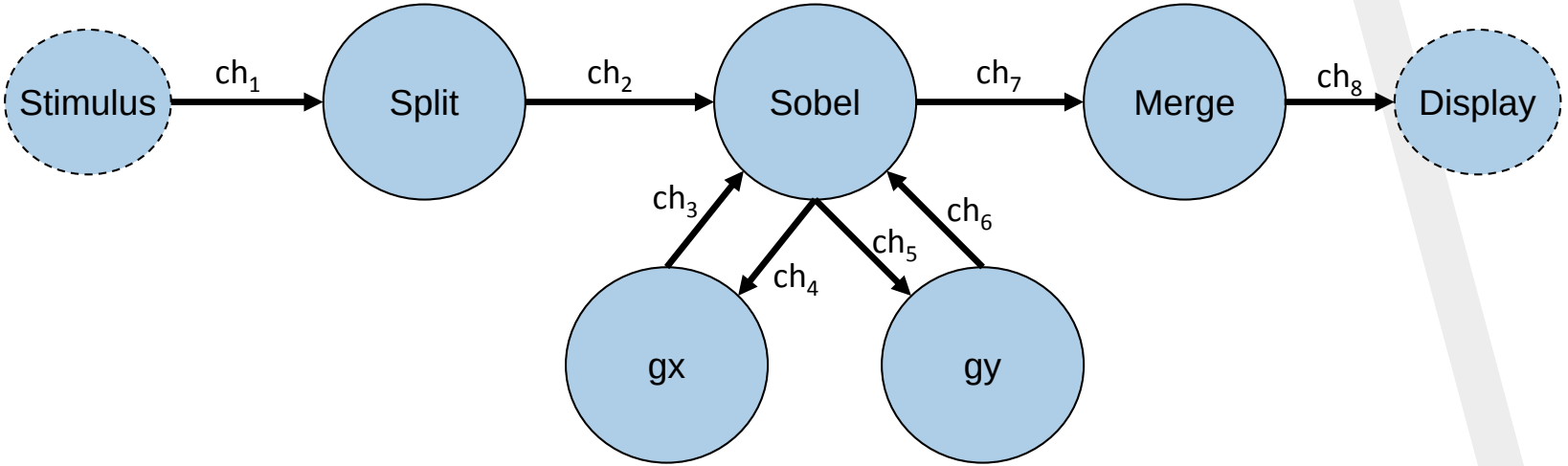


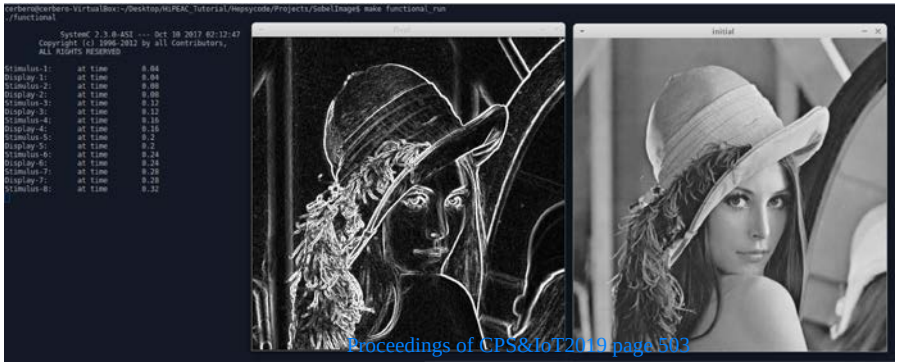

$L_1 = t_1 / ([x_1 * FTR_1] / N)$
$L_2 = t_2 / ([x_2 * FTR_1] / N)$
$L_3 = t_3 / TTR_3$ (real-time process load)
$L_4 = t_4 / ([x_4 * FTR_1] / N)$

V. Muttillo, G. Valente, D. Ciambrone, V. Stoico, and L. Pomante. **HEPSYCODE-RT: a Real-Time Extension for an ESL HW/SW Co-Design Methodology**. In Proceedings of the 10th Workshop on Rapid Simulation and Performance Evaluation: Methods and Tools (RAPIDO '18). ACM, New York, NY, USA, 2018

# Case Study 4: Sobel Image

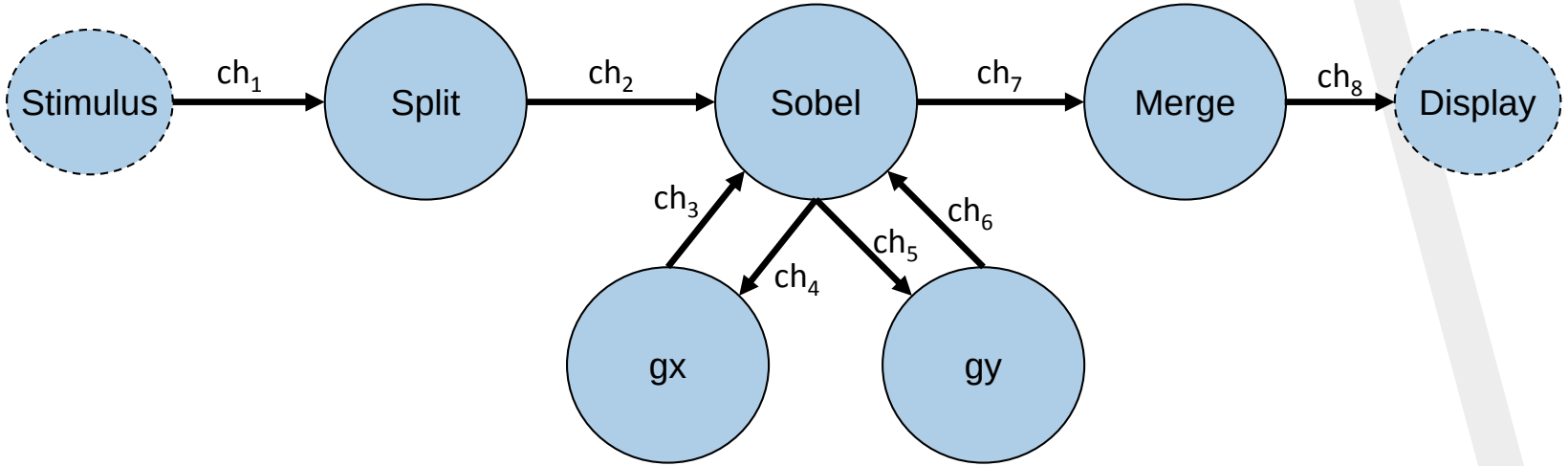
## (OpenCV)

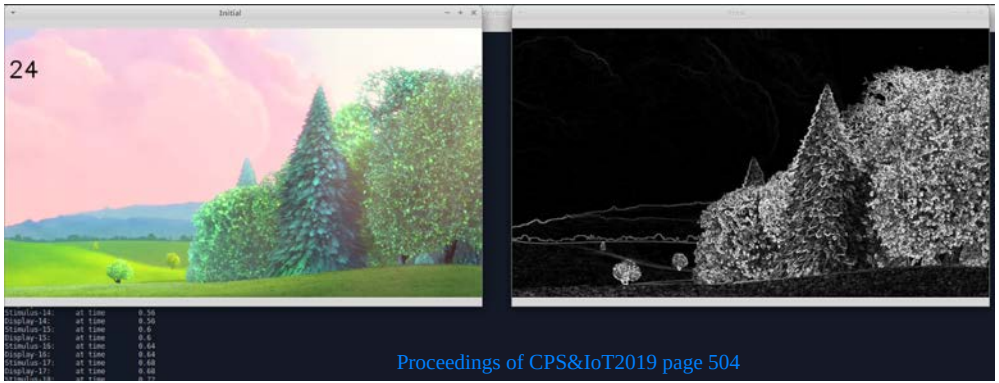


# Case Study 5: Sobel Video

## (OpenCV)






Proceedings of CPS&IoT2019 page 504

MECO education & training

# 10.

# Hepsycode Ecosystem

"In the long history of humankind (and animal kind, too) those who learned to collaborate and improvise most effectively have prevailed – *Charles Darwin*"

# Hepsycode Ecosystem

# 11

# Publications and European Projects

*"Framework Programmes for Research and Technological Development"*

# Pubblications

## Journals

1. Muttillo, V., Valente, G., Federici, F., Pomante, L., Faccio, M., Tieri, C., Ferri, S.: “A design methodology for soft-core platforms on FPGA with SMP Linux, OpenMP support, and distributed hardware profiling system”, In: EURASIP Journal on Embedded Systems
2. Pomante, L., Muttillo, V., Krena, B., Vojnar, T., Veljkovic, F., Pacome, M., Matschnig, M., Fischer, B., Martinez, J., Gruber, T.: “The AQUAS ECSEL Project - Aggregated Quality Assurance for Systems: Co-Engineering Inside and Across the Product Life Cycle”, In Microprocessors and Microsystems: Embedded Hardware Design (MICPRO) **MINOR REVISION**
3. Pomante, L., Muttillo, V., Santic, M., Serri, P.: “SystemC -based Electronic System-Level Design Space Exploration Environment for Dedicated Heterogeneous Multi-Processor Systems”, In: Microprocessors and Microsystems: Embedded Hardware Design (MICPRO) **SUBMITTED**
4. Muttillo, V., Tiberi, L., Pomante, L.: “ Benchmarking Analysis of Hypervisor Technologies for Aerospace Multi-core Systems”, In: Journal of Aerospace Information Systems (JAIS) **SUBMITTED**

## Conferences

1. V. Muttillo, L. Pomante, P. Balbastre, J. Simo. HW/SW Co-Design Framework for Mixed-Criticality Embedded Systems considering Xtratum-based SW Partitions. Euromicro Conference on Digital System Design. 2019 **SUBMITTED**
2. V. Muttillo. Hw/sw co-design methodology for mixed-criticality and real-time embedded systems. In Design, Automation and Test in Europe (DATE 2019), Ph.D. Forum, Florence, Italy, Mar. 2019.
3. V. Muttillo, G. Fiorilli, . Di Mascio. Tuning dse for heterogeneous multi-processor embedded systems by means of a self-equalized weighted sum method. In Workshop on Parallel Programming and RunTime Management Techniques for Manycore Architectures and Design Tools and Architectures for Multicore Embedded Computing Platforms, 2019
4. V. Muttillo, G. Valente and L. Pomante, “Design Space Exploration for Mixed-Criticality Embedded Systems considering Hypervisor-based SW Partitions”, Euromicro Conference on Digital System Design, Prague, 2018. **BEST POSTER AWARD**
5. D. Ciambrone, V. Muttillo, L. Pomante and G. Valente, "HEPSIM: An ESL HW/SW co-simulator/analysis tool for heterogeneous parallel embedded systems" 7th Mediterranean Conference on Embedded Computing, Budva, 2018. **BEST PAPER AWARD**



Proceedings of CPS&IoT2019 page 508

# Pubblications

## Conferences


6. V. Muttillo and G. Valente, "Injecting hypervisor-based software partitions into Design Space Exploration activities considering mixed-criticality requirements", 2018 7th Mediterranean Conference on Embedded Computing (MECO), Budva, 2018
7. V. Muttillo, G. Valente, L. Pomante. “Criticality-aware Design Space Exploration for Mixed Criticality Embedded Systems”. In Proceedings of the 9th ACM/SPEC on International Conference on Performance Engineering (ICPE ’18), ACM, New York, NY, USA, 2018
8. V. Muttillo, G. Valente, L. Pomante, V. Stoico, F. D’Antonio, F. Salice. “CC4CS: an Off-the-Shelf Unifying Statement-Level Performance Metric for HW/SW Technologies”. In ACM/SPEC International Conference on Performance Engineering (ICPE '18), 2018, pp. 119-122
9. V. Muttillo, G. Valente, L. Pomante. “Criticality-driven Design Space Exploration for Mixed-Criticality Heterogeneous Parallel Embedded Systems”. In 9th Workshop and 7th Workshop on Parallel Programming and RunTime Management Techniques for Many-core Architectures and Design Tools and Architectures for Multicore Embedded Computing Platforms (PARMA-DITAM ’18), 2018
10. V. Muttillo, G. Valente, D. Ciambrone, V. Stoico, L. Pomante. “HEPSYCODE-RT: a Real-Time Extension for an ESL HW/SW Co-Design Methodology. 10th Workshop on Rapid Simulation and Performance Evaluation: Methods and Tools (RAPIDO’18), 2018
11. Di Pompeo, D., Incerto, E., Muttillo, V., Pomante, L., Valente, G.: “An Efficient Performance-Driven Approach for HW/SW Co-Design”, In: Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering (ICPE)., L’Aquila, Italy, 22-27 Apr. 2017
12. Faccio, M., Federici, F., Marini, G., Muttillo, V., Pomante, L., Valente, G.: "Design and validation of multi-core embedded systems under time-to-prototype and high-performance constraints", In: Research and Technologies for Society and Industry (RTSI), Bologna, Italy, 7-9 Sep. 2016
13. Valente, G., Muttillo, V., Pomante, L., Federici, F., Faccio, M., Moro, A., Ferri, S., Tieri, C.: "A Flexible Profiling Sub-System for Reconfigurable Logic Architectures", In: Parallel, Distributed, and Network-Based Processing (PDP), pp. 373-376, Heraklion Crete, Greece, 17-19 Feb 2016

# European Projects

**ARTEMIS-JU AIPP 2013-621429 EMC2 (Embedded Multi-Core systems for Mixed Criticality applications in dynamic and changeable real-time environments):**

Deliverable D2.3: Design, implementation, prototyping and verification approach for mixed-critical and parallel applications, Sep. 2015
Deliverable D2.4: Intermediate validation report based on selected living labs scenarios, Mar. 2016
Deliverable D2.5: Complete modelling and analysis framework, Oct. 2016
Deliverable D2.6: Comprehense validation report for the modelling frameworks and offline tools, based on refined living labs results, Apr. 2017

**H2020 ECSEL RIA 2016-737494 MegaM@rt2 (MegaModelling at Runtime - scalable model-based framework for continuous development and runtime validation of complex systems):**

Deliverable D1.2: Architecture specification and roadmap - initial version, Oct. 2017
Deliverable D1.4: Architecture specification and roadmap – final Version, Jun. 2018
Deliverable D2.2: Design Tool Set Specification, Feb. 2018
Deliverable D2.3: Design Tool Set – Initial Version, Jun. 2018
Deliverable D6.3: Dissemination and Exploitation Report – initial release, Feb. 2018

**H2020 ECSEL RIA 2016-737475 AQUAS (Aggregated Quality Assurance for Systems):**

Deliverable D2.1.1: Domain Environment – Air Traffic Management, Oct. 2017
Deliverable D2.1.5: Domain Environment – Space Multicore Architecture, Oct. 2017
Deliverable D2.2.1: Demonstrator Architecture - Air Traffic Management, Apr. 2018
Deliverable D2.2.5: Demonstrator Architecture - Space Multicore Architecture, Apr. 2018
Deliverable D3.1: Specification of Safety, Security and Performance Analysis and Assessment Techniques, Apr. 2018

# HEPSYCODE Repository

HW/SW CO-DEsign of HEterogeneous Parallel dedicated SYstems:

Tool available for free on a git repository under GPL2 for testing, improvements, collaborations etc.
Web Site: **www.hepsycode.com**

You can download the HEPSYCODE tool on this page:
https://bitbucket.org/vittorianomuttillo87/tool-hepsycode/src/master/

MECO education & training

# 12.

# Conclusions and Future Work

*"Embedded systems are the key innovation driver to improve mechatronic products with cheaper and even new functionalities. They support today's information society as inter-system communication enabler. Consequently, boundaries of application domains are alleviated and ad-hoc connections and interoperability play an increasing role"*

# Conclusions and future work

- ESL HW/SW Co-Design approach able to take into account mixed-criticality constraints
- The methodology, design flow and framework drive the designer from the input specification to the final implementation solution, while offering timing simulation capabilities, design space exploration activities with the support of analysis tool
- It is possible to integrate this approach with other external tools (like Xamber, but other tools are under evaluation)
- FUTURE WORKS:
  - Consider multi-core scenario while introducing schedulability and RT analysis
  - Combine PAM1 and PAM2 activities into a unique DSE approach
  - Exploit parallel programming techniques (parallel meta-heuristics)
  - Analysis and tests in PAM2 considering also mixed-criticality index
  - Introduce fixed WCET values (taken from external tools)
  - Integrate other external tools to enhance HEPSYCODE functionality
  - Improve the hierarchical scheduling implementation



Proceedings of CPS&IoT2019 page 513

# Conclusions and future work

- Model GR-CPCI-LEON4-N2X Quad-Core 32-bit LEON4 SPARC V8 processor with MMU, IOMMU
- Model TASI/UNIVAQ Satellite Applications
- Contributed to benchmarking of fully-open Aeroflex Gaisler quad-LEON3 system on FPGA with Xtratum and PikeOS
- Improve case studies example, exploit works into some European Projects

# HW/ SW Co-Design Methodologies

START

**HW/SW CO-DESIGN 1.0**

**HW/SW Bi-Partitioning for CPU/ASIC Targets**

Methodologies/Tools:
Vulcan, COSYMA

1985 - 1994

Y-CHART GAJSKI-KUHN (1983)

W. WOLF (1993)

R. ERNST (1998)
W. WOLF (2003)

1995 - 2005

**HW/SW CO-DESIGN 2.0**

**Co-Design & Platform-based Design for Complex Targets**

Methodologies/Tools:
Koski, PeaCE/HoPES, Ptolemy (Polis), CoWare, Metropolis, TOSCA, Daedalus

**HW/SW CO-DESIGN 3.0**

**DSE & Co-Synthesis of HMPS**

Methodologies/Tools:
SystemCoDesigner, SynDEx, Gezel, CoFluent, TO(H)SCA

2005 - 2011

Double-Roof Model (2007)

J. TEICH (2012)

2012 - Nowadays

**HW/SW CO-DESIGN 4.0**

**Modern HW/SW Co-Design**

Online Co-Design for Adaptive Systems
Co-Design of CPS, CPSoS
Co-Design for Monitorability & Security
*HW/SW Co-Design for Mixed-Criticality Systems*

Methodologies/Tools:
ForSyDe, eSSYN, CHESS, Capella, Space CoDesign, *HEPSYCODE*

**HW/SW CO-DESIGN 5.0**

**New HW/SW Co-Design Generation:**
Model driven, Approximate Computing, Machine Learning, Quantum Computing, 5G, Smart Cities, IoT, DSE at runtime

Towards the Future




Proceedings of CPS&IoT2019 page 515

MECO education & training

THANKS!

# Any questions?



Proceedings of CPS&IoT2019 page 516