Designing Network-on-Chip based multi-core heterogeneous System-on-Chip: the MAGALI experience

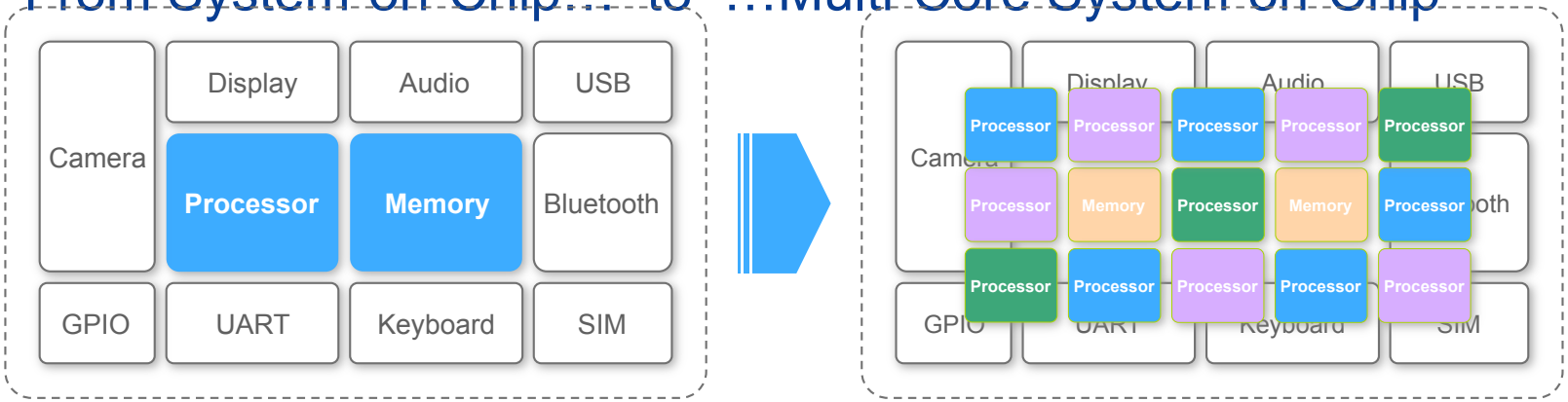
F. Clermidy, Ph'D
CEA-LETI, Minatec Campus

http://www.artist-embedded.org/

energie atomique • energies alternatives

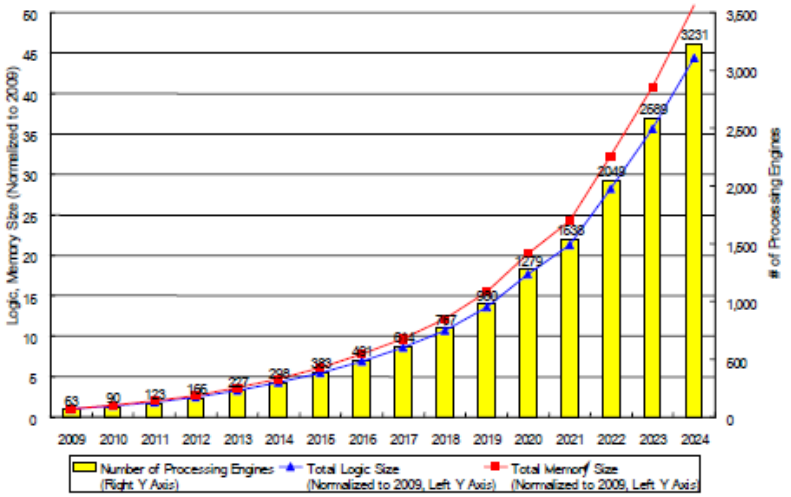
A well-known evolution: Multi-core SoC

From System-on-Chip… to …Multi-Core System-on-Chip



Complexity is exploding!

>1000 Processing Elements in 2020!!



From ITRS 2009 [www.itrs.net]

In embedded systems: ITRS 2009

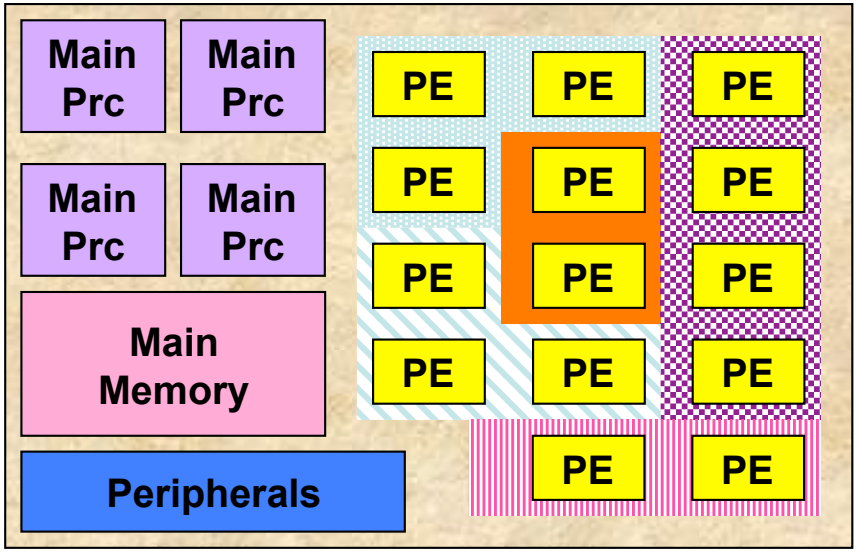
- *« SOC-Consumer Portable Drivers »*

*Performance * 1000 in 15 years*

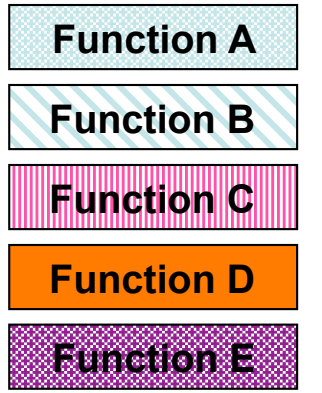
Power consumption objective 500 mW

PE = dedicated accelerators, 250 kG/64 Kbits

Same design effort



Architecture



Software View

Two possible paths

- *Homogeneous = replication of identical resources*
  - *+ Programming simplicity*
  - *+ Fault and variability-tolerance*
  - *+ Flexibility*
  - *- Area*
  - *- Power consumption / performance*
- *Heterogeneous = each resource has its own dedicated function*
  - *+ Area*
  - *+ Power consumption / performance*
  - *- Each resource is critical*
  - *- Programming is more complex*

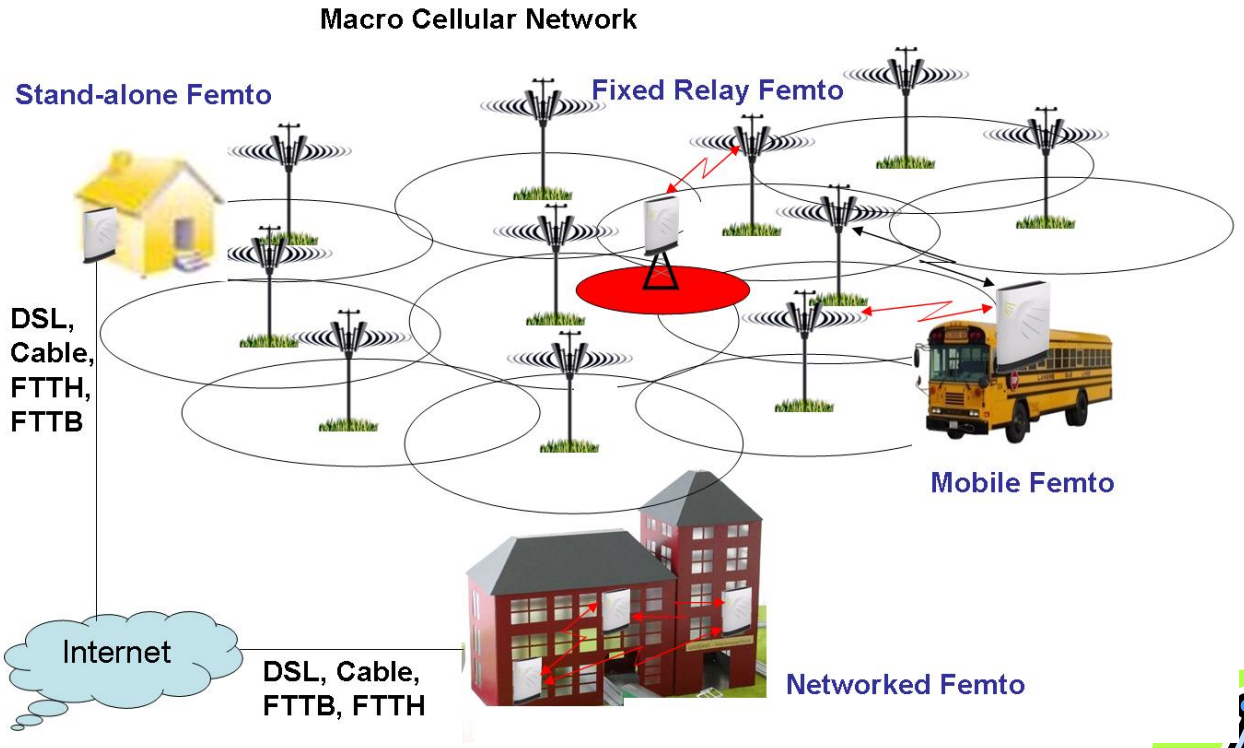
Towards regular and adaptable architectures

- *Quiet evolution: mixing heterogeneous and homogeneous*
  - *Communications are key: Network-on-Chip (NoC)*
  - *Control distribution*
- *Revolution ?*
  - *Dynamic adaptation through reconfiguration*
  - *Distributing decisions*

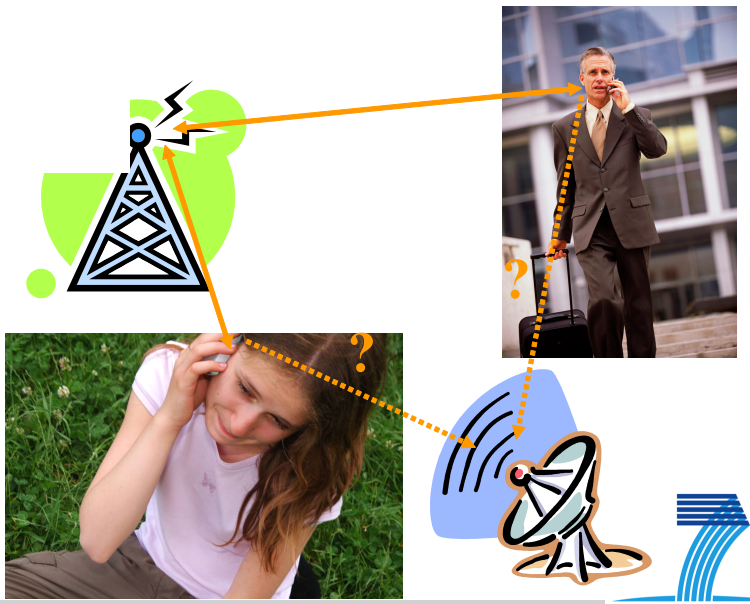
Outline

- Context
- MAGALI overview
- NoC, GALS and Low-Power
- Dynamic reconfiguration
- Distributed decisions
- Configuring & programming
- Conclusion

An application starting point



"Software Defined Radio"
Femtocells
MIMO
(ICT projects Befemto & ARTIST4G)



"Cognitive Radio"
TERROP
NEWCOM

What are the problems?

- Increasing complexity
  - MIMO scheme
  - Spectral efficiency increase

  => 1 Tops needed in 2015
- Increasing flexibility
  - Software Defined Radio, Cognitive Radio

  ➔ more control, more configurations
- Strict constraints
  - Hard real-time: frame = 1 ms
  - Mastering computing latency mandatory

  (latency => memory => real estate => cost)
  - Power consumption under 500 mW

C.H. Van Berkel, "**Multi-Core for Mobile Phones**", DATE'09

LETI's NoC

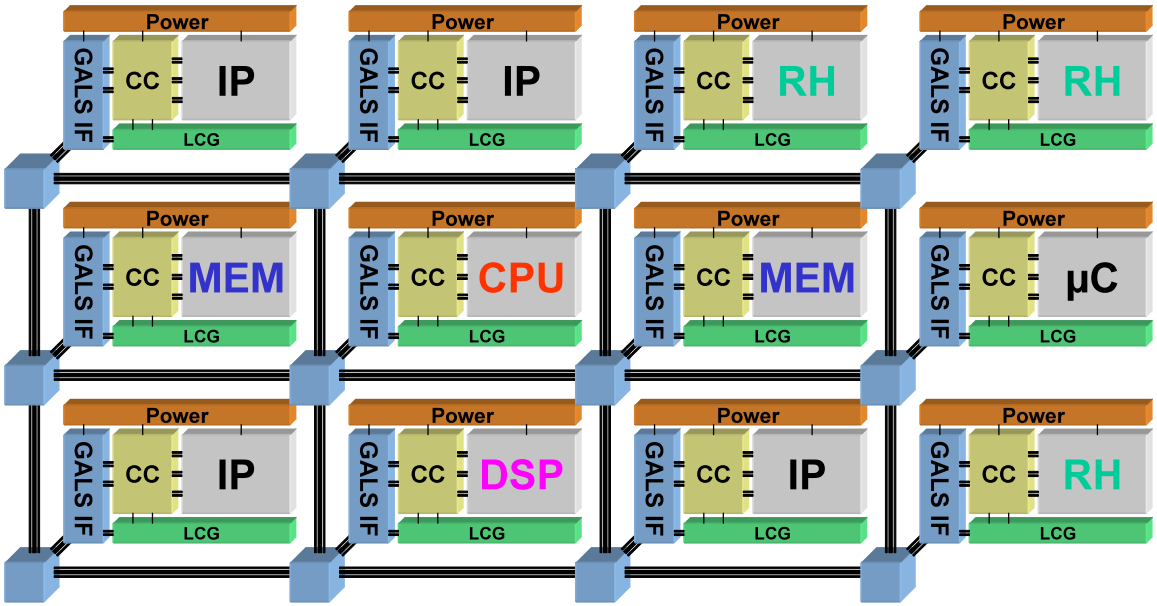
GALS GALS interface

LCG Local Clock Generator

Power Power Control

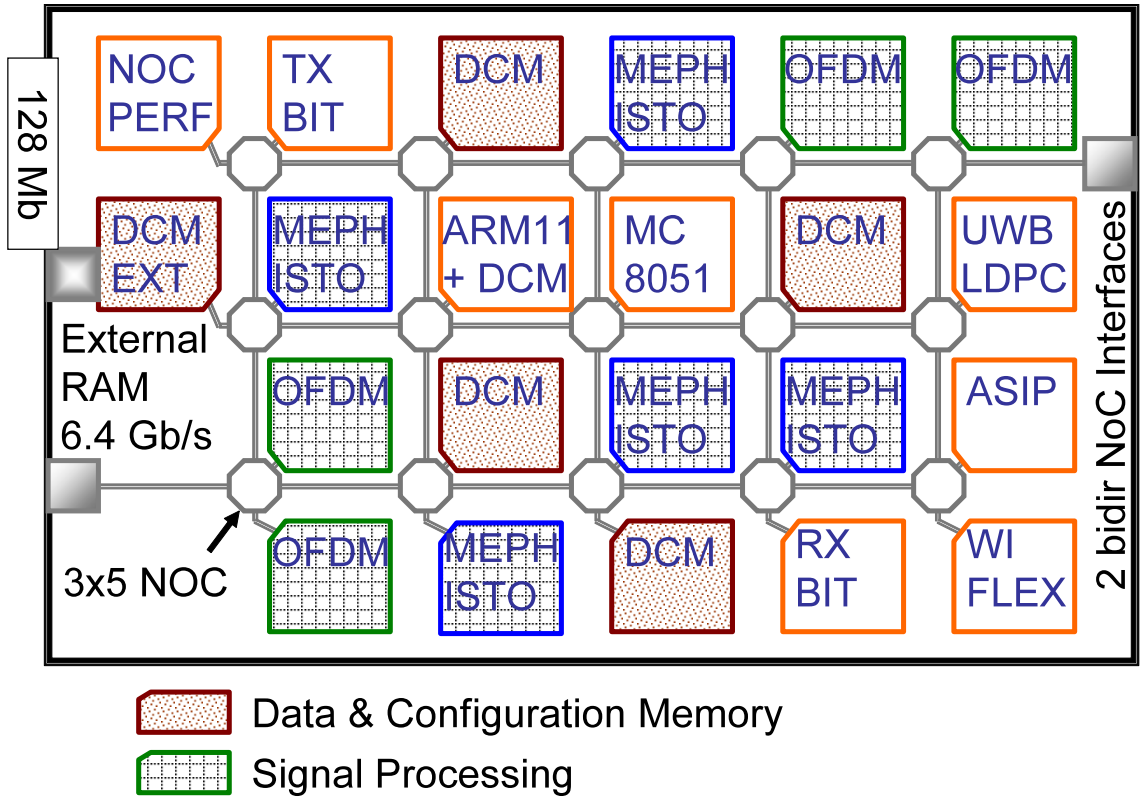
CC Communication/Configuration controller

IP IP Core



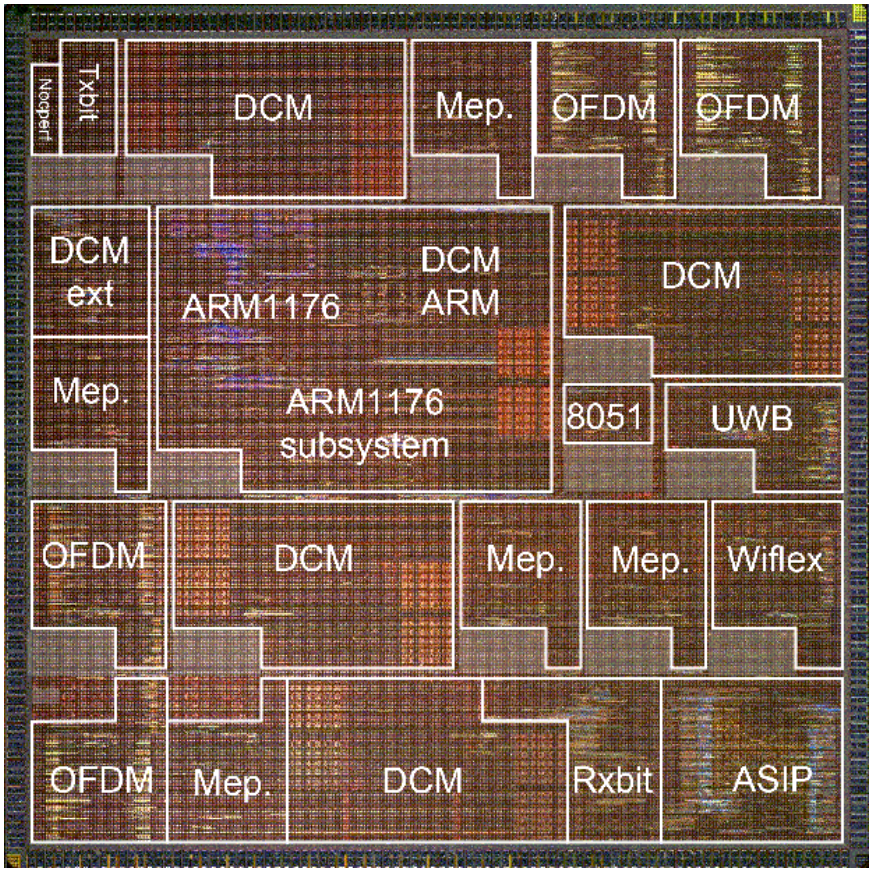
- 2-D mesh based NoC
- Support heterogeneous tiles: IP, memory blocs, programmable cores, reconfigurable hardware
- Data-flow homogeneous programming model
- Communication/Configuration (CC) controller
- GALS implementation for advanced power management

MAGALI Chip



- TRX_OFDM: 32-2048 FFT/iFFT, GI insertion, framing/deframing, power normalization
- DCM: Fully programmable memory cores for data storage and manipulation (32Kwords 32bits), Configuration server
- MEPHISTO: VLIW cores for complex matrices computation (8GMAC/s)
- BIT cores: Support for mapping / interleaving / puncturing (TX and RX)
- FEC decoders: reconfigurable channel decoders supporting LDPC, Viterbi and Turbo decoders

MAGALI Chip layout



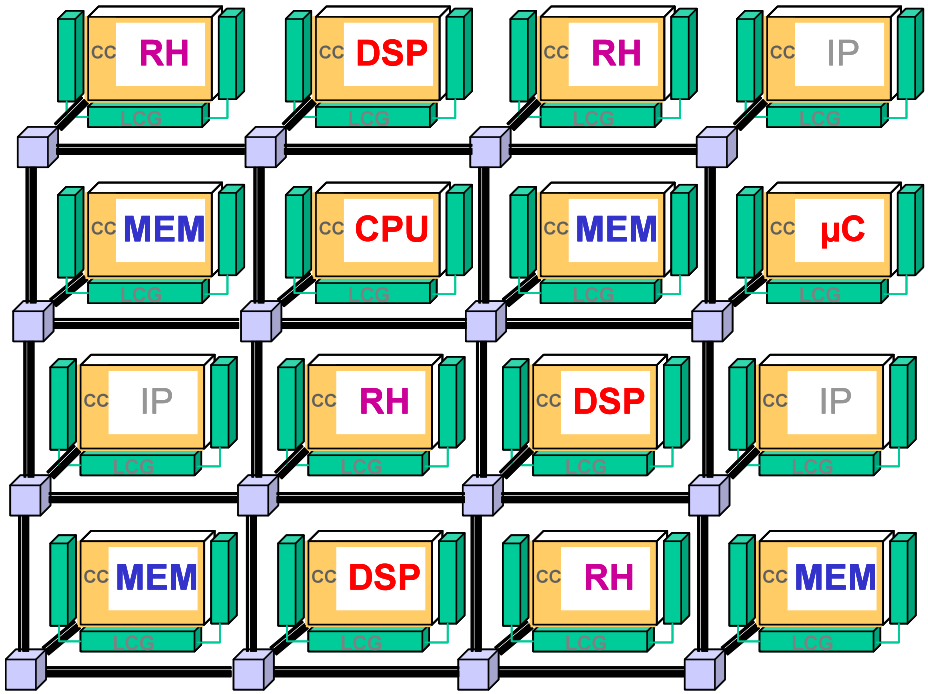
- ST 65nm LP technologie, 5400µm x 5400µm, 30mm²
- Total power < 500mW
- NoC area (15 Routers + 20 GALS interfaces + NoC links) : 11% overall chip area

Outline

- Context
- MAGALI overview
- NoC, GALS and Low-Power
- Dynamic reconfiguration
- Distributed decisions
- Configuring & programming
- Conclusion

What is a NoC?

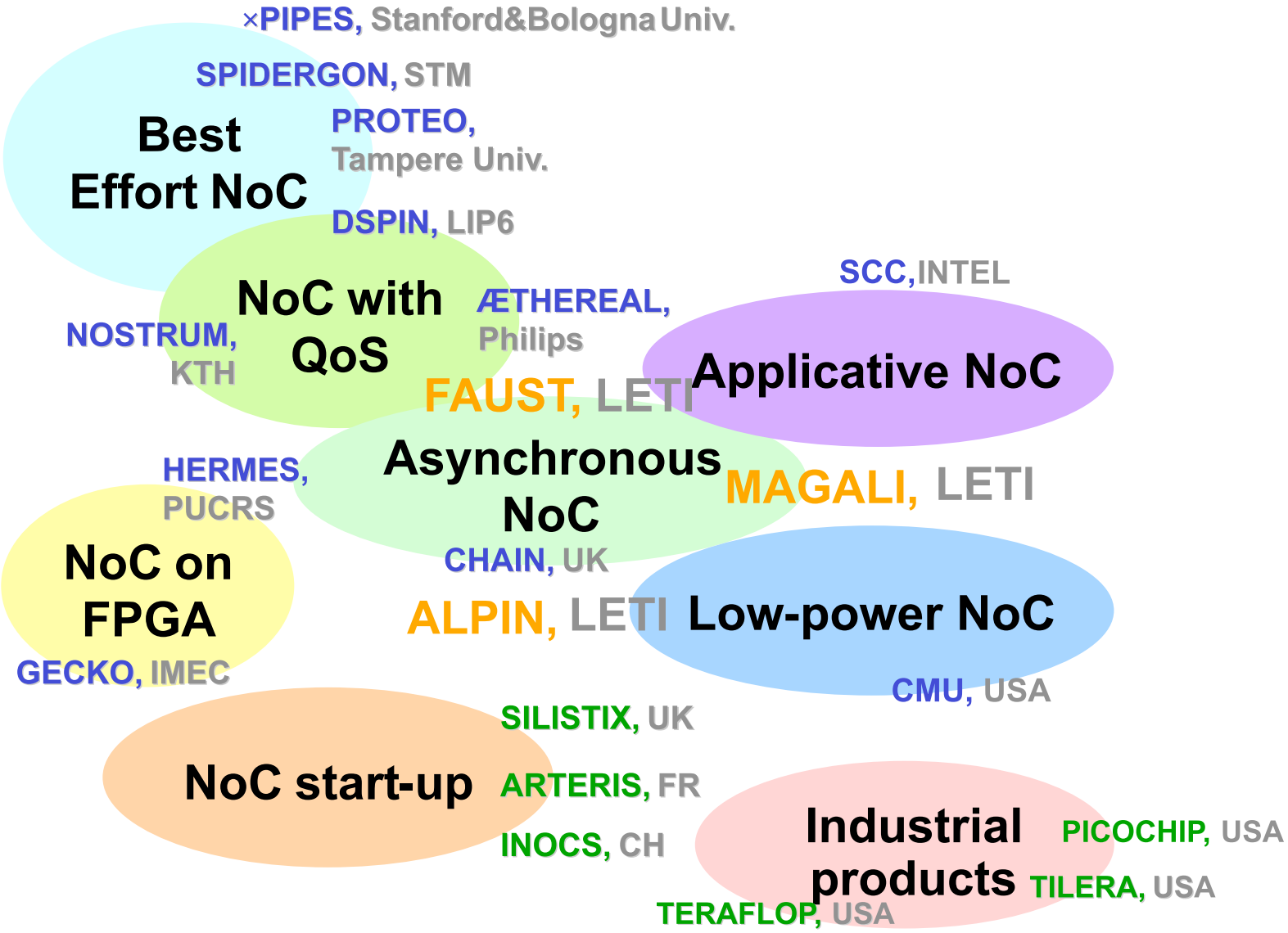
- "NoC is an interconnection structure for exchanging information on a chip between heterogeneous or homogeneous HW/SW resources"



Some history

- 1980 to 2000 : Multiprocessors networking
- 2000, Jantsch et Al. « NoC: an architecture for billion transistor area »
- 2000 : A. Greiner et Al. « SPIN, a fat-tree topology for IP communications »
- 2001, Dally et Al. : « Route packets, not wires »
- 2002, Benini and De Micheli: « NoC: a new SoC paradigm »

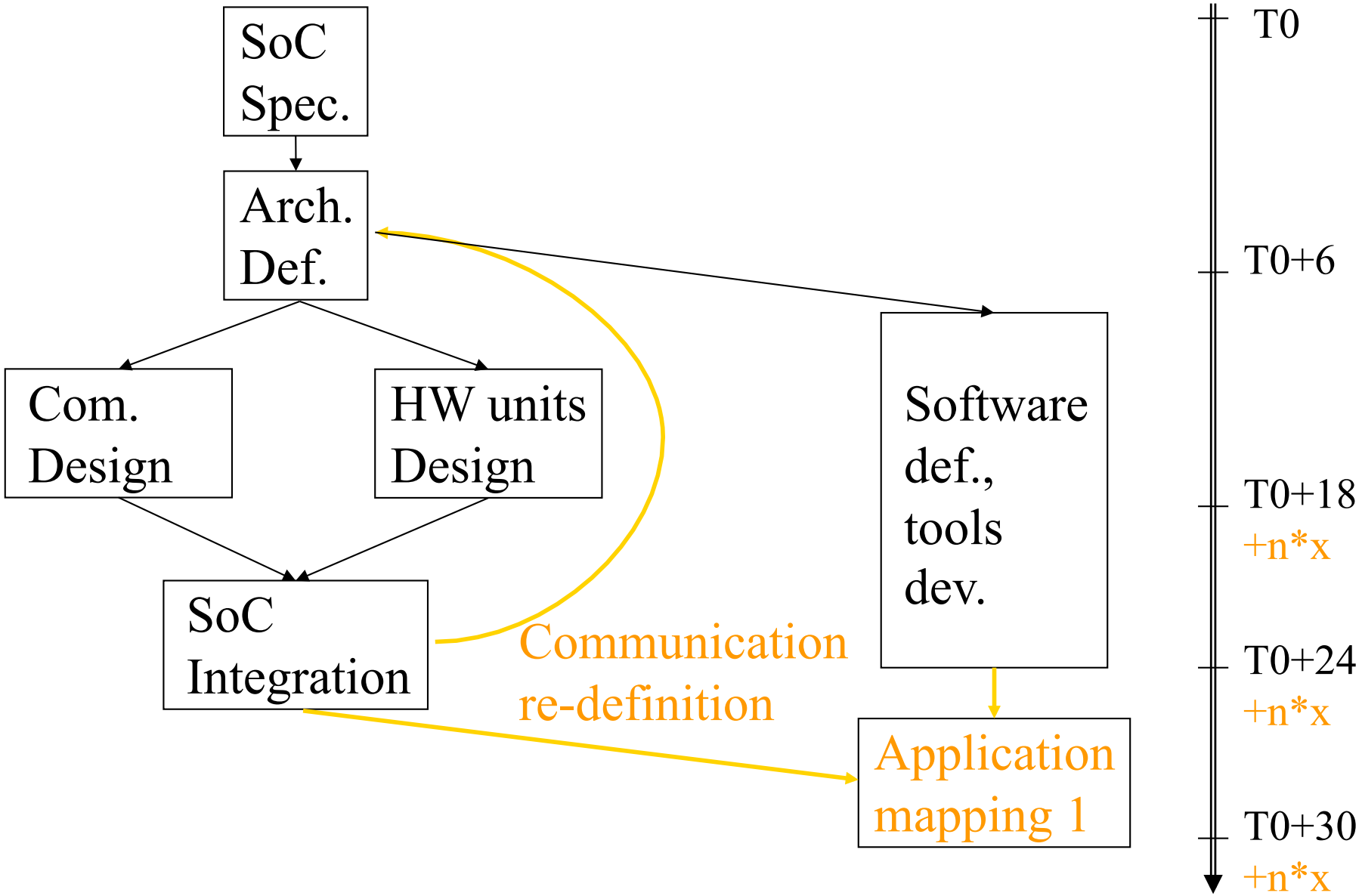
NoC research worldwide



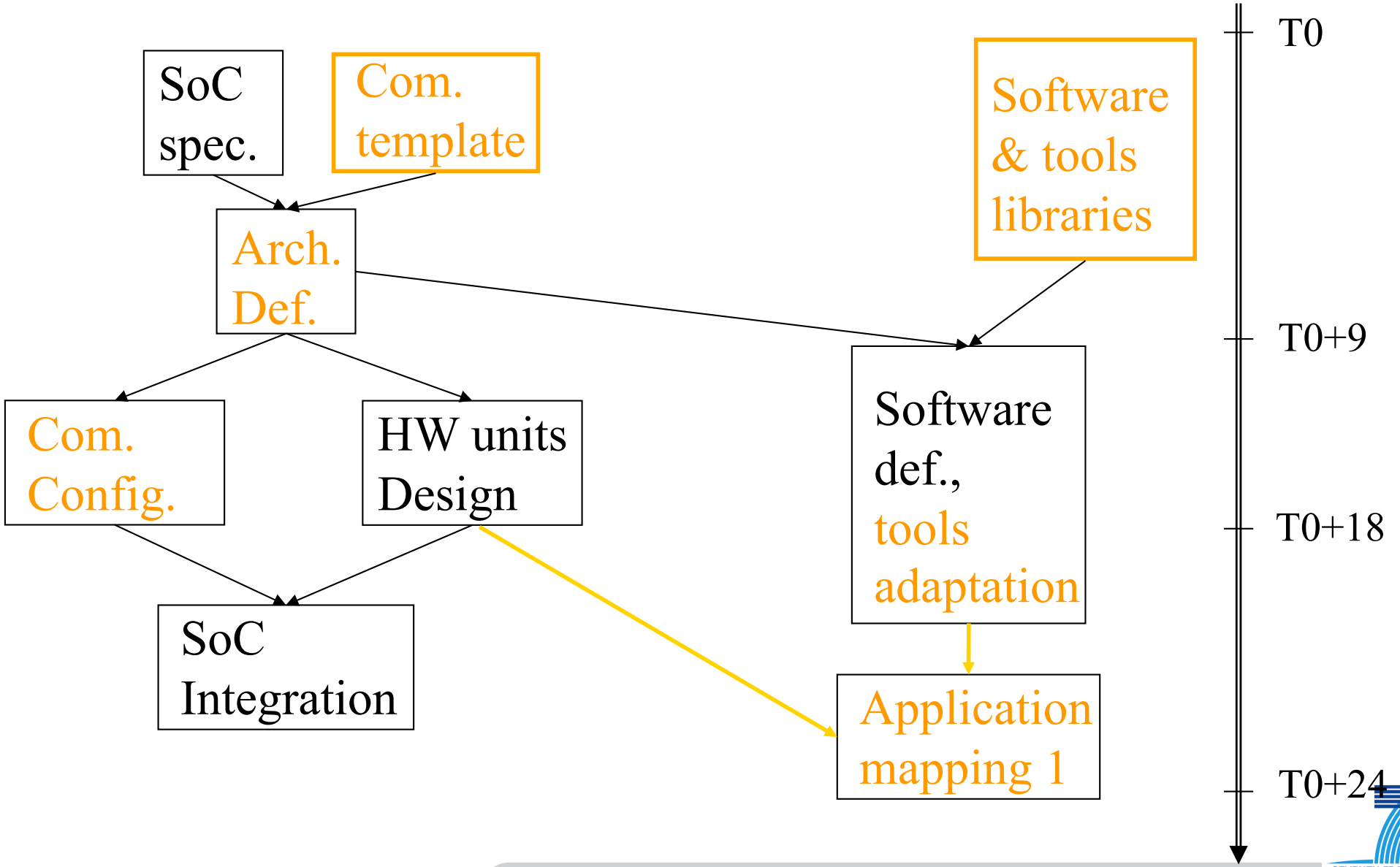
Communication-Centric platform

- Concepts
  - Architecture platform articulated around a Network-on-Chip
  - Network-on-Chip with QoS for high throughput communications, low latency, deadlock or live-lock free, reliability
  - Efficient implementation with GALS techniques
    - Key element for power management and isolation of faulty elements
  - Need for:
    - Efficient programming model
    - Associated tools:
      - Development possible thanks to platform concept

SoC standard methodology



SoC communication-centric methodology



NoC topics

- NoC is in the heart of programming model
  - What are the functions of a NoC ?
    - Just an efficient interconnection medium
    - Added Quality of Service
    - Partial/full Support for programming model
  - Communication protocol stack implemented

- NoC is in the heart of parallel and distributed computing
  - New tools for application mapping are needed

- NoC is in the heart of implementation issues
  - Globally Asynchronous, Locally Synchronous structures
  - NoC is a potential weakness point for reliability, variability

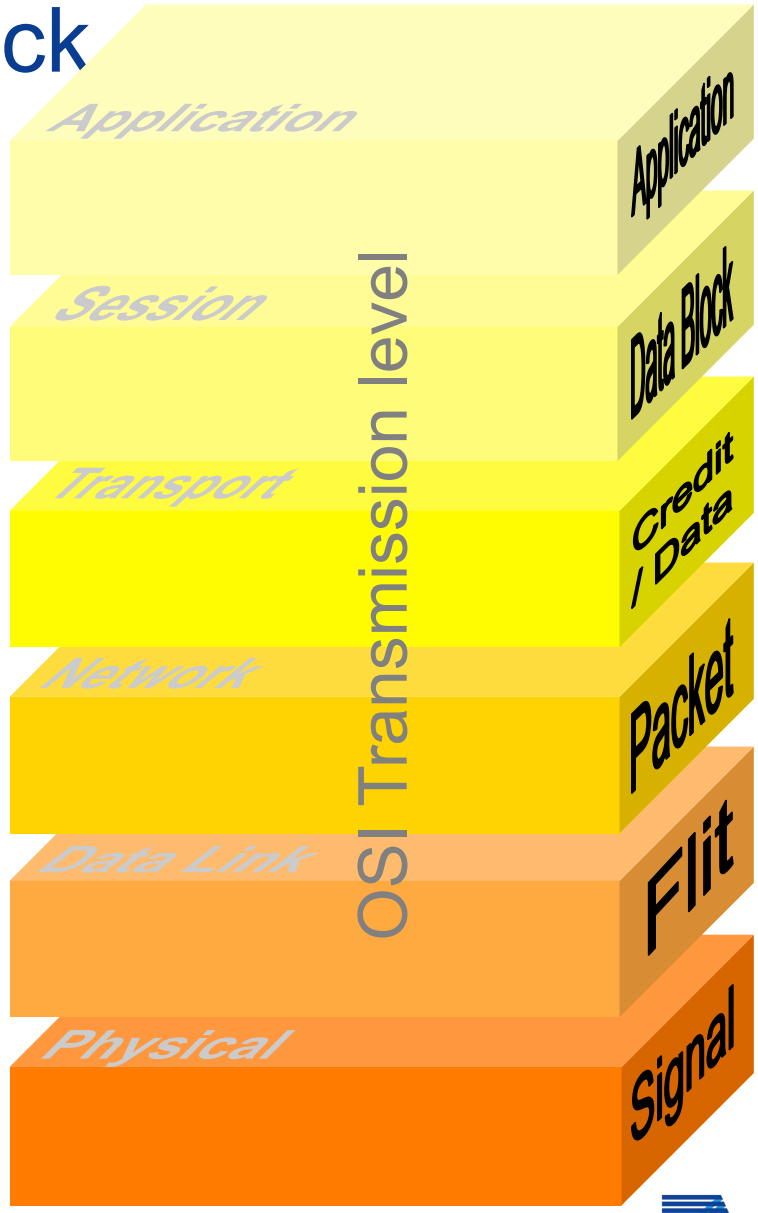
NoC topics

- NoC is in the heart of power consumption issues
  - NoC itself can be power hungry
  - NoC can open new solutions for smart management of power for the whole structure
- NoC is a new paradigm shifting from IP re-use to platform re-use
  - Need new design tools (exploration, construction)
- NoC arises new questions on classical topics
  - Testability of the NoC itself, and its associated IPs
  - Debug is a difficult issue : determinism is often required by industrials, but difficult to achieve…with GALS, parallel and distributed structures

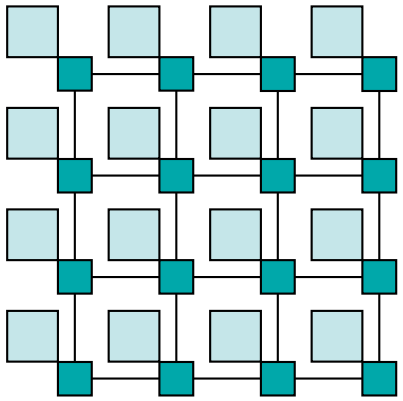
NoC Protocol Stack

Programming model of the NoC-based platform is essential. It can determine :

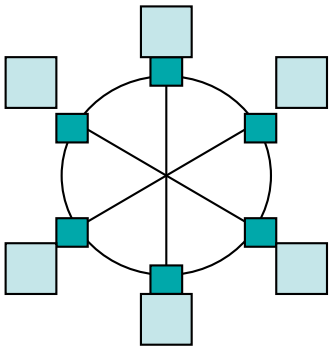
- **Reconfiguration management**
- **Task synchronization**
- **Power management**
- **Bandwidth allocation**
- **End-to-end flow control**
- **Protocol wrappers**
- **Packet routing**
- **GALS strategy**



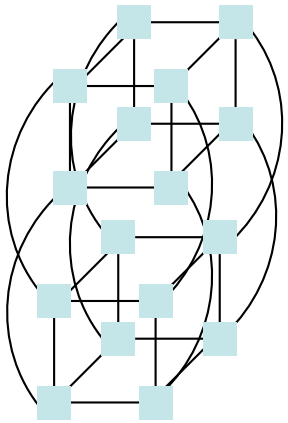
An example of NoC particularities: Topology



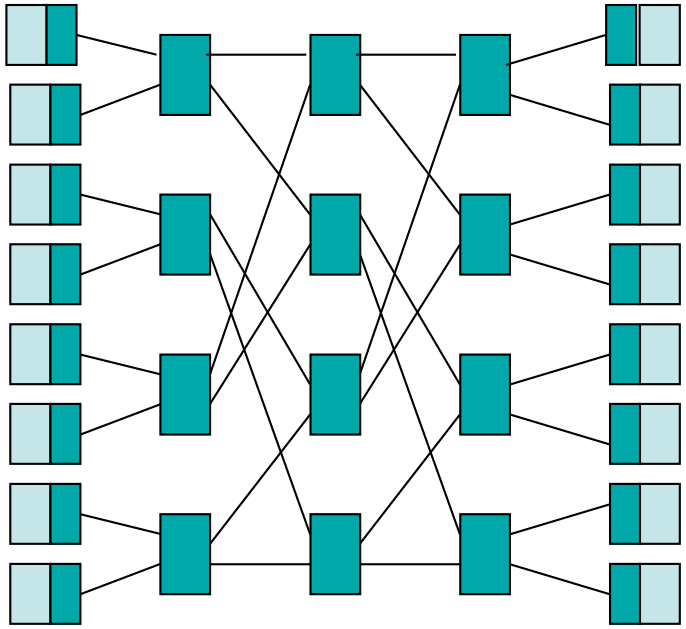
Mesh



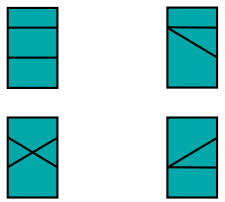
Chordal ring



Hypercube



Omega Network

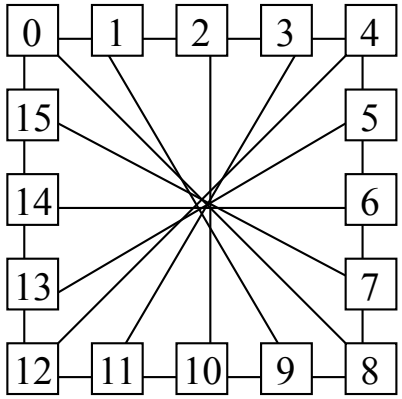


Switches Configurations

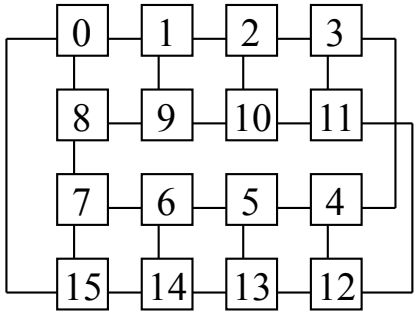
Scalability and implementation

- **Topology: Chordal ring**
- **Implementation:**
  - **NODE 4*4 only** => 20 to 25 % area gain, < 5% performance gain compared to 5*5 needed for mesh
  - Is it a **good layout ?**
  - Long wires ?

Topology vs 2-D layout : scaling



OK

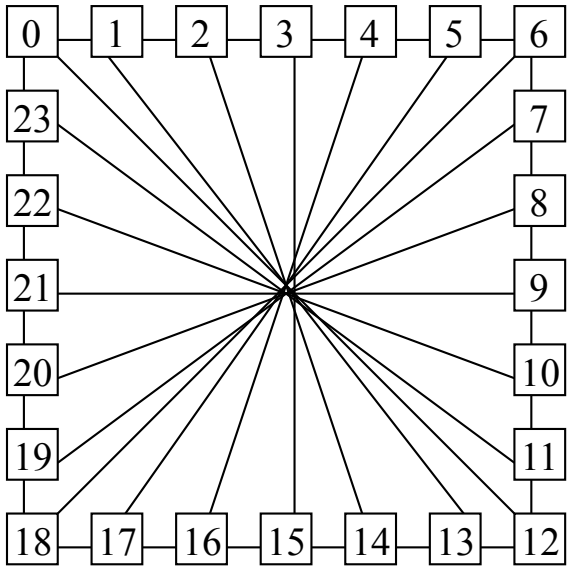


16 nodes

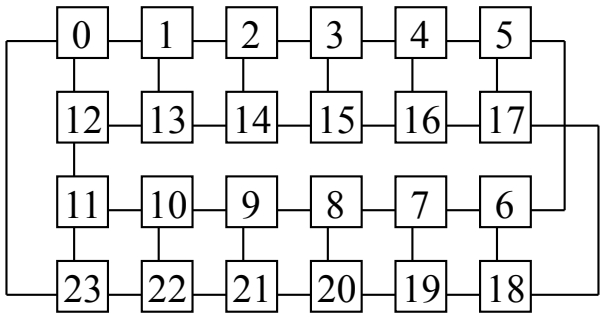
Topology : 4*4

Layout 1:1

2 medium wires
1 long
1 cross



OK



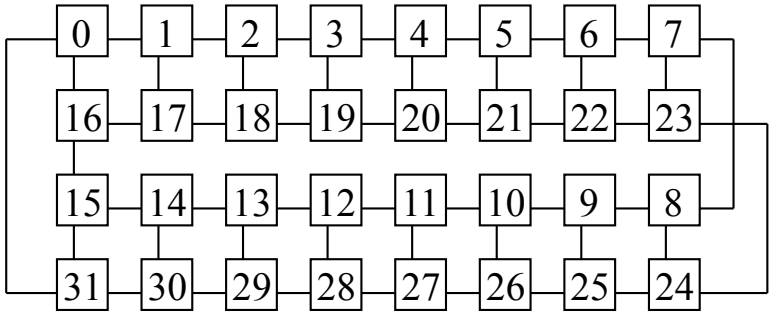
24 nodes

Topology : 6*4

Layout 3:2

2 medium wires
1 long
1 cross

Topology vs 2-D layout: 32 units case



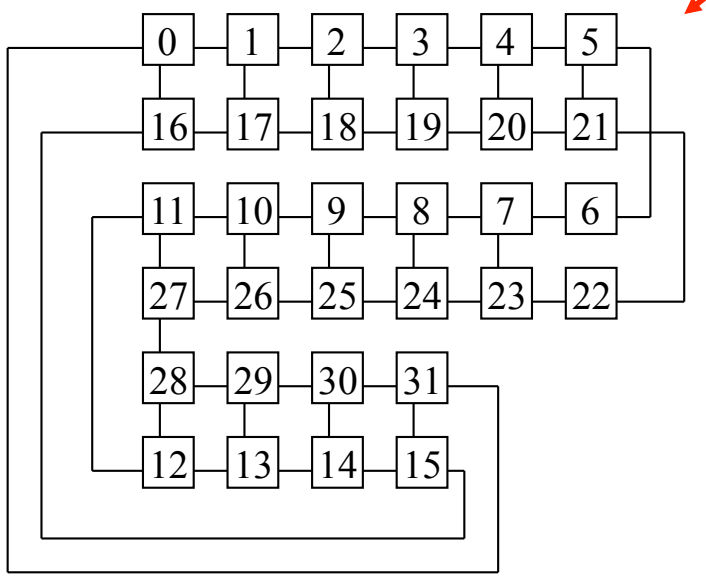
Topologie : 8*4

Layout 2:1

NOK

2 average wires
1 long
1 cross

New layout



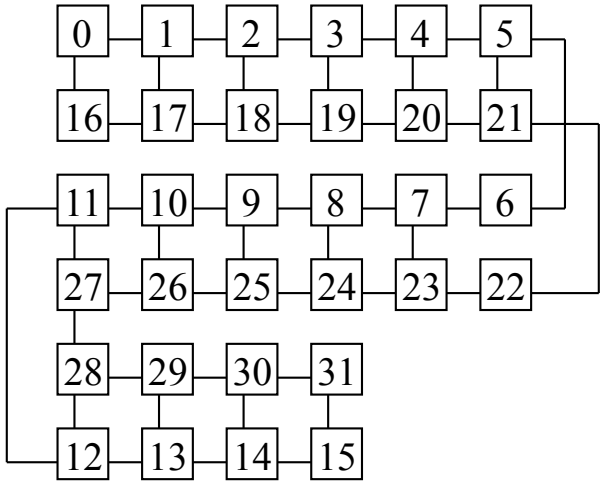
OK, but
Long wires

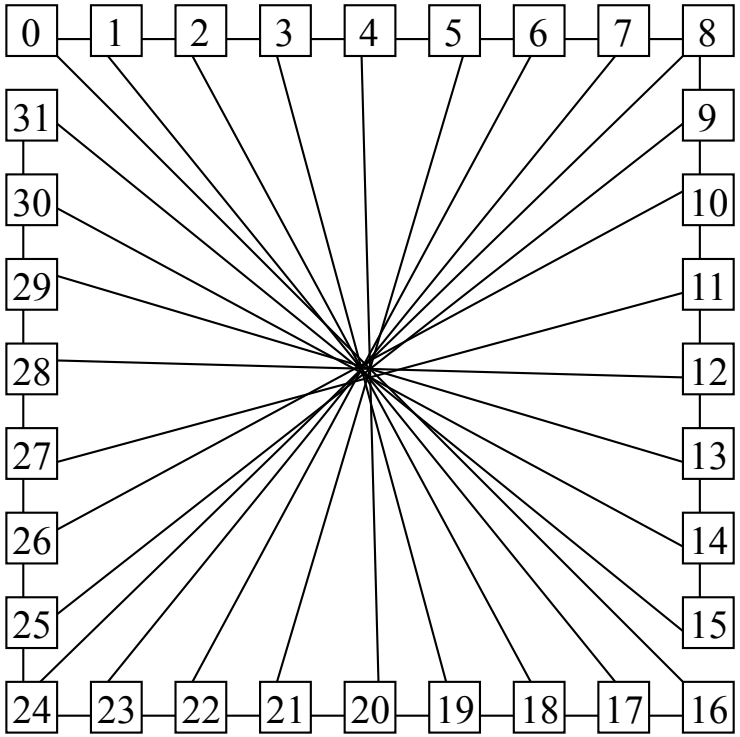
Topologie : 8*4

Layout 1:1

3 average wires
2 long wires
1 cross

Deleting the long wires ?





Topology : 8*4

Layout 1:1

4 average wires
1 wire crossing
Long lines deleted

Diameter = 16

Equivalent mesh diameter = 10

Final comparison mesh/chordal ring

MESH

Diameter = 2*(SQRT(N)-1)
N = 16, D = 6
N = 32, D = 10
N = 64, D = 14
N = 79, D = 16

With layout view :

Layout 1:1 short lines only

Equivalent diameter = real diameter

CHORDAL RING

Diameter = N/4 (if all the wires)
N = 16, D = 4
N = 32, D = 8
N = 64, D = 16
N = 79, D = 20

With layout view :

Layout 1:1 with long lines

If long lines deleted :
Equivalent diameter = real diameter *2
+ p*mean lines costs

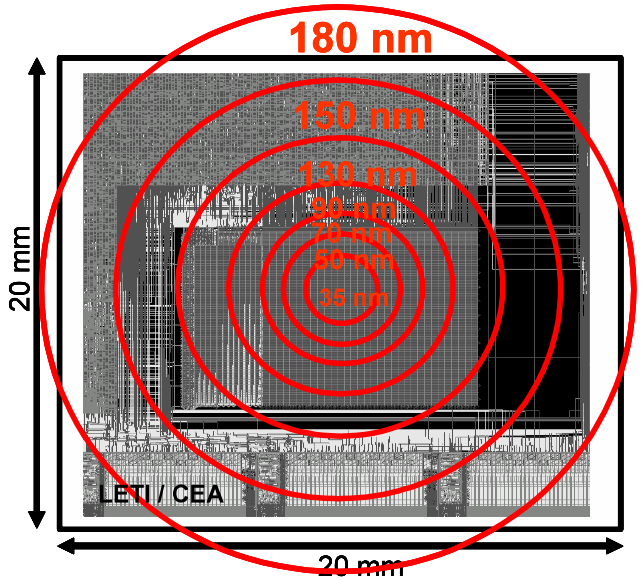
Else
Equivalent diameter = real diameter +
1*long line costs + p*mean line costs
If cost ~ real distance :
= real diameter *2 + p*mean lines costs

Implementation challenges

- Globally Asynchronous Locally Synchronous (GALS) architecture
  - NoC is in the center of such issues
- Low-power schemes
  - Communication is power-consuming
  - NoC implementation influences low-power policies
- Test & Debug
  - Mandatory for industrial acceptance
  - Distributed systems induced by NoC are difficult to debug (lost of pure determinism in many cases)
- Tools
  - Mandatory for NoC-based architecture design
- Other challenges :
  - Optical NoC, 3-D implementation

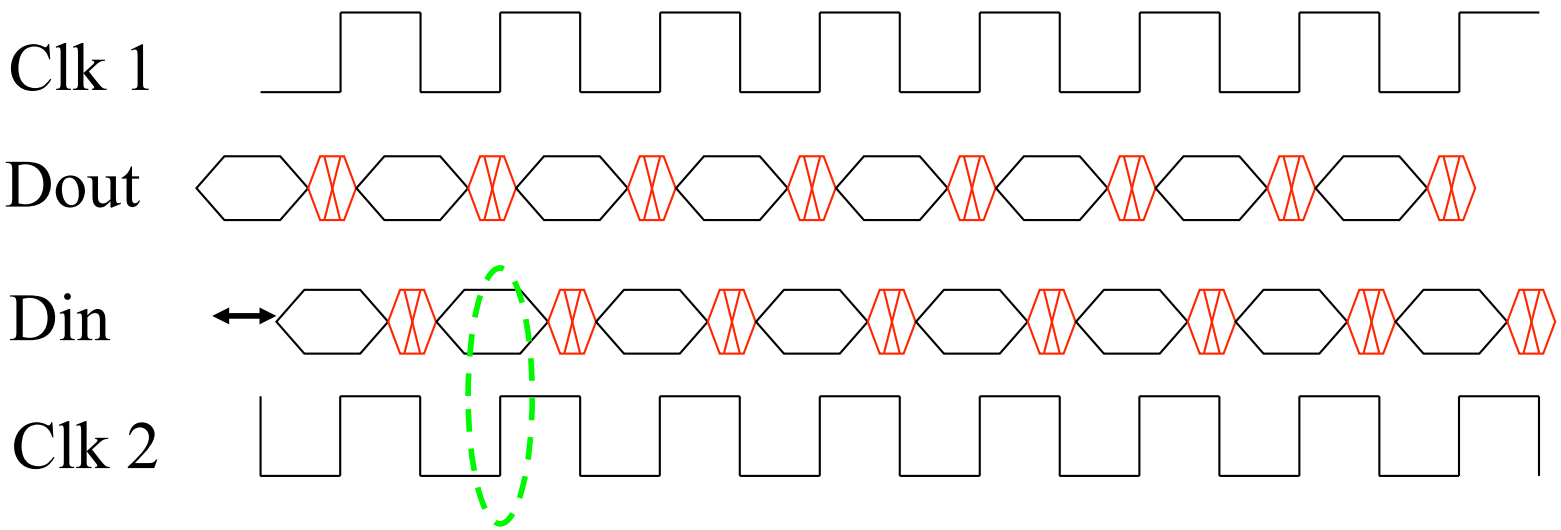
GALS Architectures

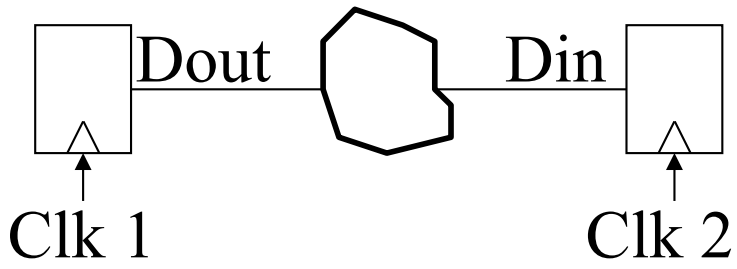
- With technology shrinks
  - Timing closure & Clock tree synthesis problems, even when using Physical Synthesis
  - Reliability issues
  - Communication Power Consumption (due to long wire loads)



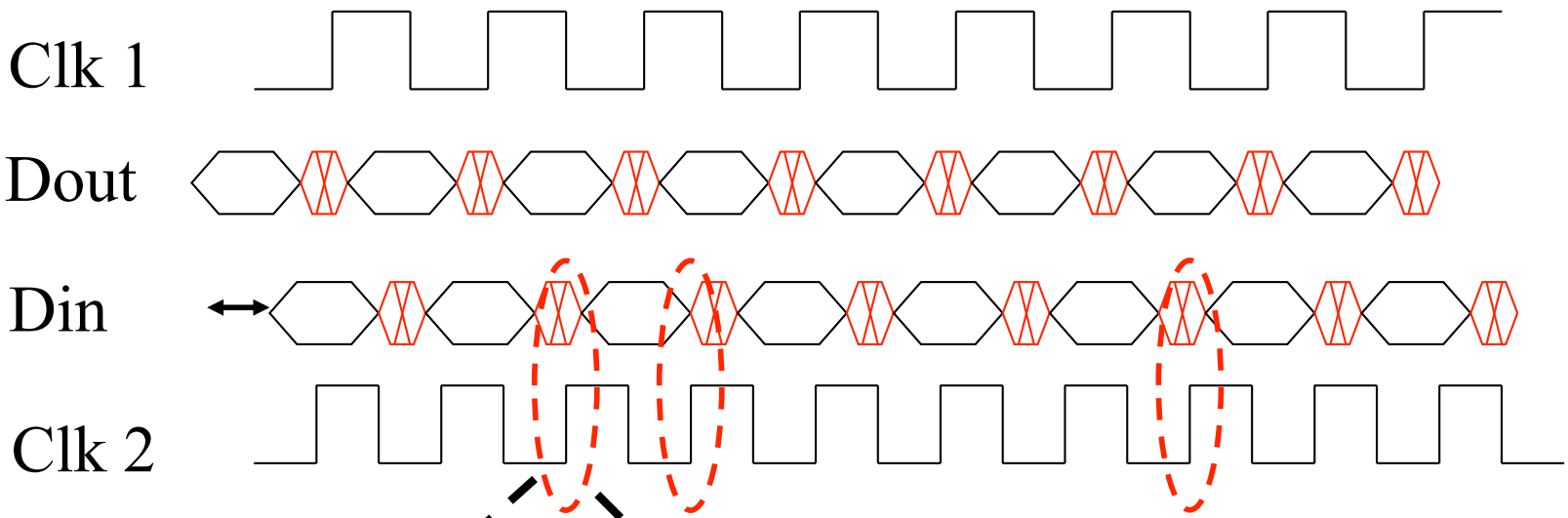
- Globally Asynchronous Locally Synchronous (GALS) architecture
  - IPs are synchronous islands
  - System communications are asynchronous

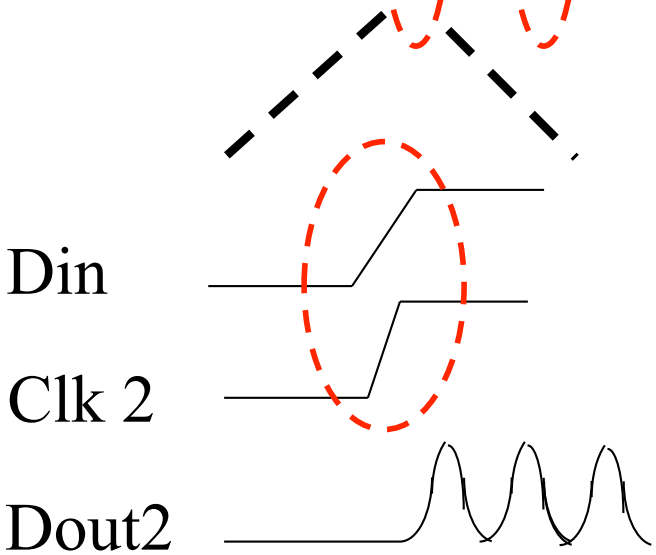
Metastability issue (1)





Metastability issue (2)



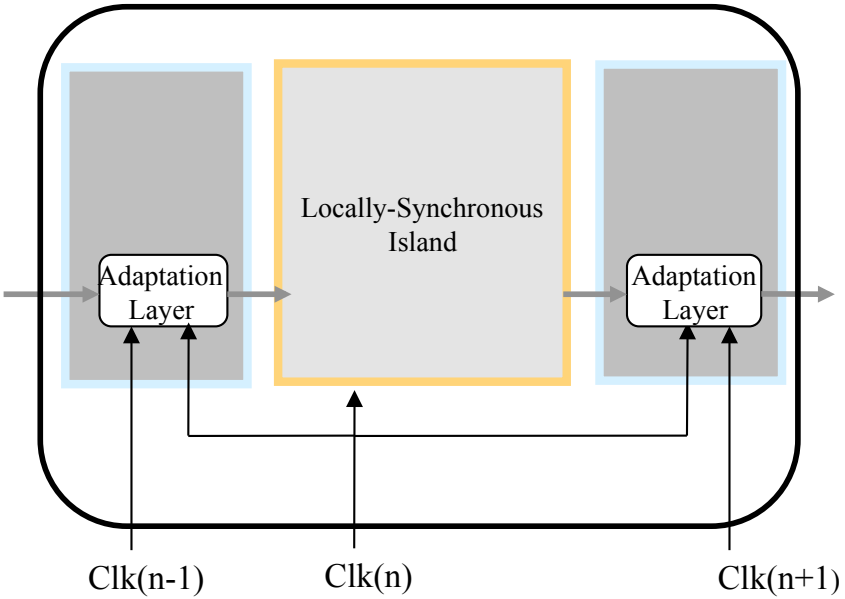


Multiple flip-flop

Can “solve” the problem

Boundary Synchronization (mesochronous)

- ✓ Low area overhead
- ✓ Power consumption
- ✗ Verification
- ✗ Throughput
- ✗ Latency

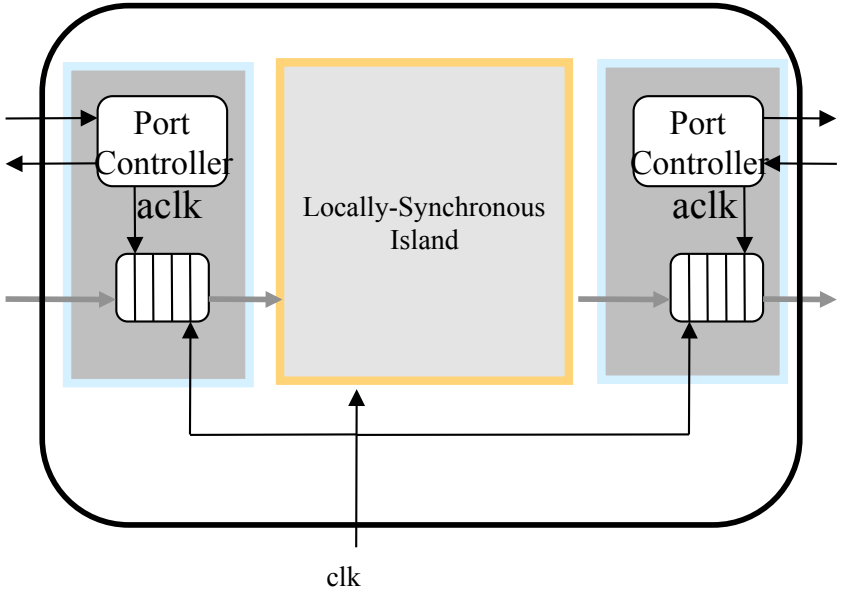


R. Dobkin, R. Ginosar, C. Sotiriu, Data Synchronization Issues in GALS SoCs, *Proceedings of the 10th International Symposium on Asynchronous Circuits and Systems*, pp. 170-179, Crete, Greece, 19 - 23 April 2004.

T. Bjerregaard, S. Mahadevan, R. Grøndahl Olsen and J. Sparsø, An OCP Compliant Network Adapter for GALS-based SoC Design Using the MANGO Network-on-Chip, *Proceedings of the International Symposium on System-on-Chip (SoC'05)*, pp. 171-174, 2005.

Bi-synchronous Gray FIFO based

- ✓ Simple solution,
- ✓ no additional cells
- ✓ high throughput
- ✗ area cost
- ✗ power consumption



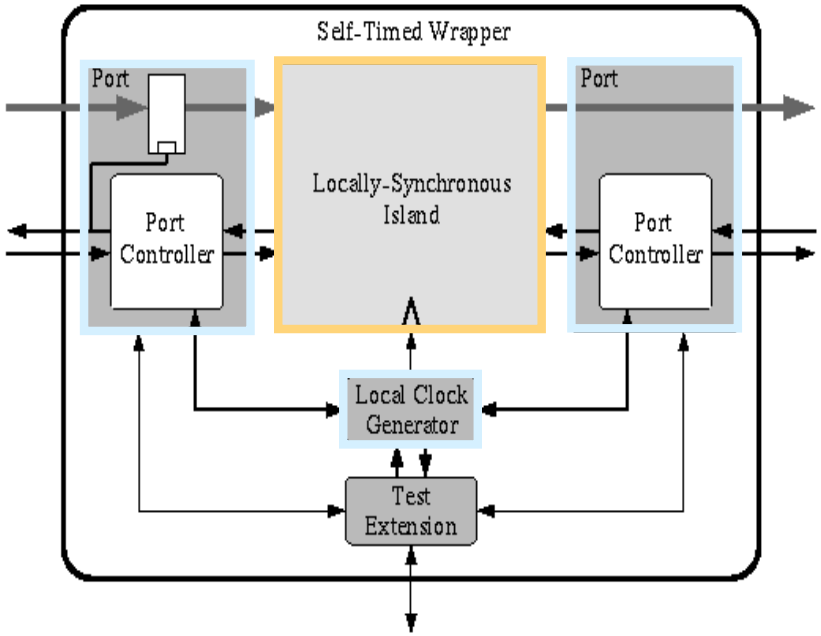
T. Chelcea, S. Nowick, Low-latency asynchronous FIFO's using token rings, *Proceedings of International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pp. 210-220, April 2000.

A. Chakraborty, M. Greenstreet, Efficient Self-Timed Interfaces for Crossing Clock Domains, *Proceedings of 9th International Symposium on Asynchronous Circuits and Systems (ASYNC'2003)*, pp. 78-88, Vancouver, Canada, 2003.

E. Beigne, P. Vivet, Design of On-chip and Off-chip Interfaces for a GALS NoC Architecture, *Proceedings of 12th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC'06)*, Grenoble, France, pp. 172-181, March 2006.

Pausable (or stretchable) clocks

- ✓ Low area overhead
- ✓ Low consumption
- ✓ Adaptable to DFS
- ✗ Need local clock generator & specialized cells
- ✗ Throughput lowered



K. Yun, R. Donohue, Pausible Clocking: A first step toward heterogeneous systems, Proceedings of International Conference on Computer Design (ICCD), October 1996.

J. Muttersbach, T. Villiger, W. Fichtner: "Practical Design of Globally-Asynchronous Locally-Synchronous Systems", Proceedings of the Sixth International Symposium on Advanced Research in Asynchronous Circuits and Systems, ASYNC'2000, Eilat, Israel, pp. 52-59, April 2-6, 2000.

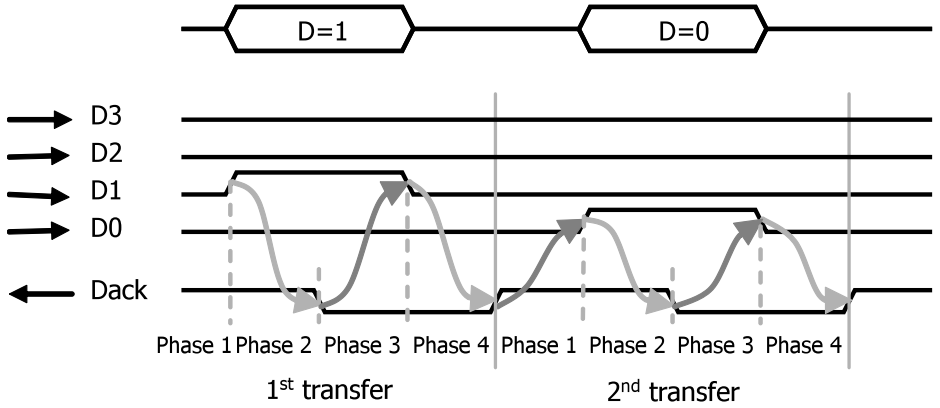
GALS interfaces: conclusion

- Mesochronous is simple BUT limited
- Pausable Clock has intrinsic defaults for industrialization
- GALS FIFO are the best way. Gray code is not optimal => other code

- Ex: 65 nm, MAGALI chip - Johnson Code
  - 500 Mhz
  - 0.014 mm2
  - 10 µW leakage
  - 3 pJ/flit

Asynchronous NoC nodes and links

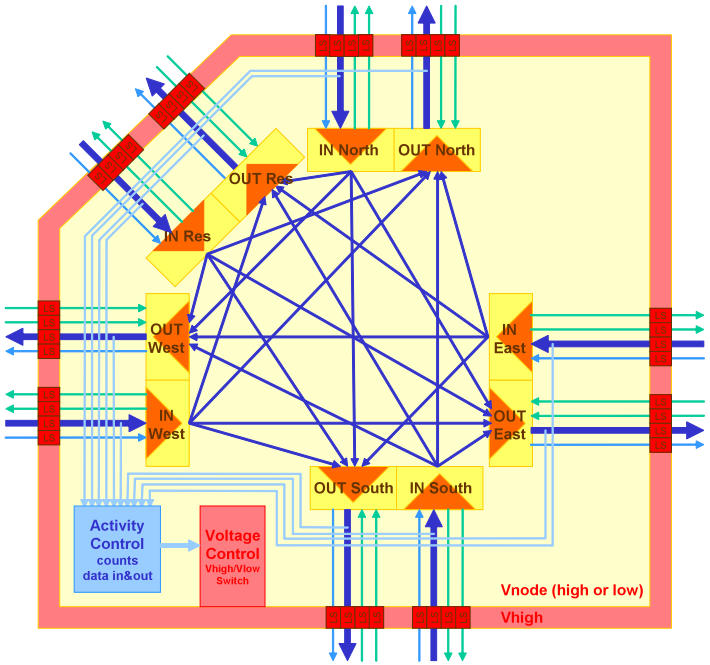
- 5x5 network router, mesh topology
- Delay Insensitivity



QDI 4-rail pipeline stage

- High Robustness to process variations and external conditions
  - temperatures, voltage drop…
- Natural enabler for Dynamic Voltage Scaling
  - no need for clock frequency scaling during transitions

Async. Node Architecture & Performance



Architecture

- Fully decentralized arbitration
- 5 Input Controllers : flits routing
- 5 Output Controllers : flits arbitration
- 2 Virtual Channels

Techno : CMOS 65nm

Throughput: 550 Mflits/s – 17.6 Gb/s

Leakage: 210 µA

Energy: 30 pJ/flit

Area 0.17 mm²

LETI's NoC

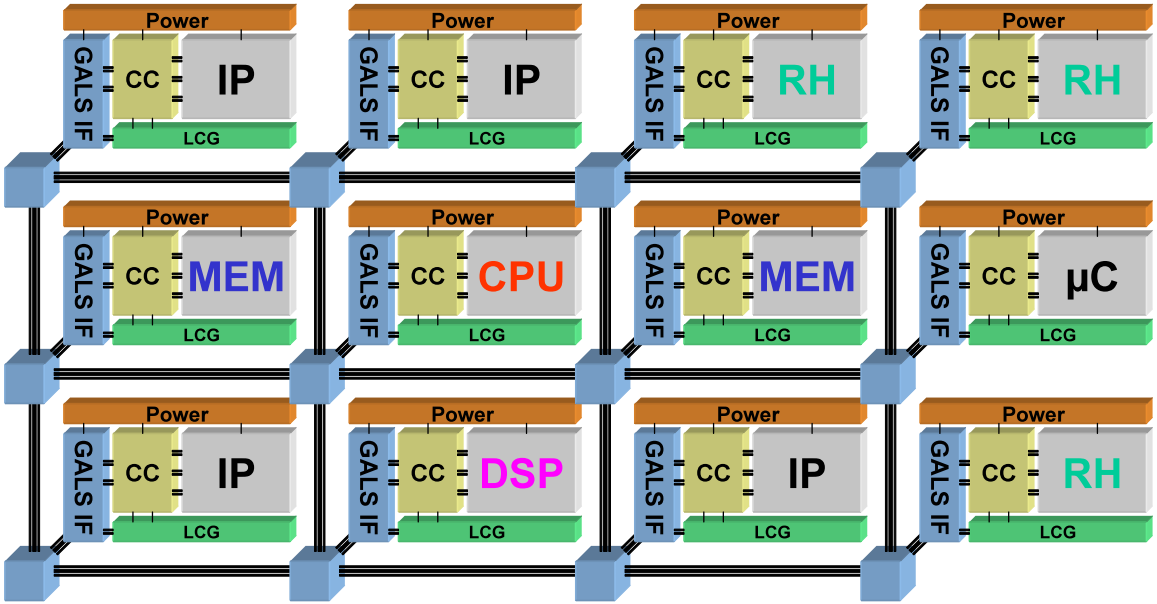
GALS GALS interface

LCG Local Clock Generator

Power Power Management

CC Communication/Configuration controller

IP IP Core



- 2-D mesh based NoC
- Communication/Configuration (CC) controller
- Support heterogeneous tiles : IP, memories (MEM), programmable cores, reconfigurable hardware (RH)
- GALS implementation
- Tools for NoC-based design and exploitation

Low-Power & NoC

- Transmission lines
- **Local DVFS**
- Partial activation of routers
- Data coding
- Routing algorithms
- Topology choice
- **Programming model**
- **Application**

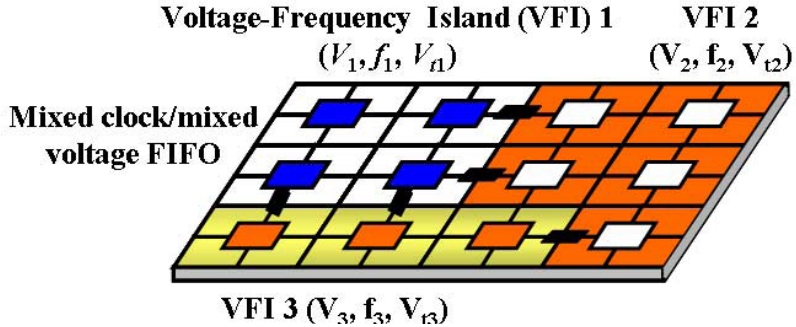
Transistors

System

Power Gain

Local DVFS

- Always associated with GALS techniques



- Island partitioning
  - NoC regions are at different voltages

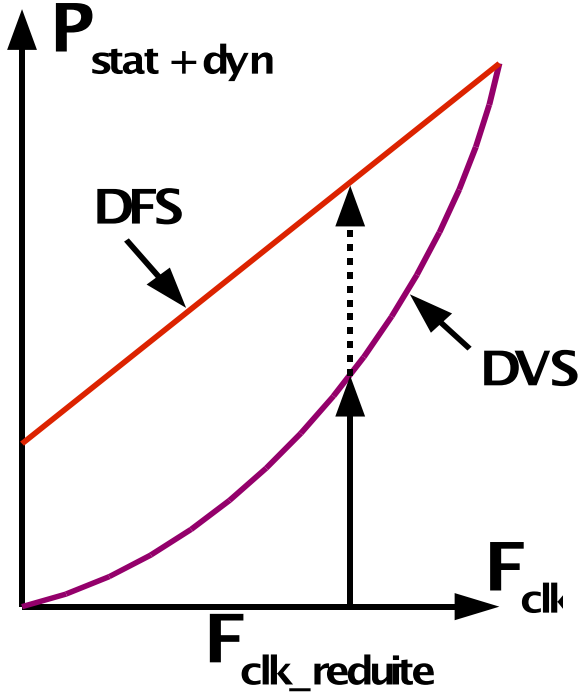
U. Y. Ogras, R. Marculescu, P. Choudhary, D. Marculescu, "Voltage-Frequency Island Partitioning for GALS-based Networks-on-Chip" Proceedings of *DAC 2007, June 4–8, 2007, San Diego, California, USA*

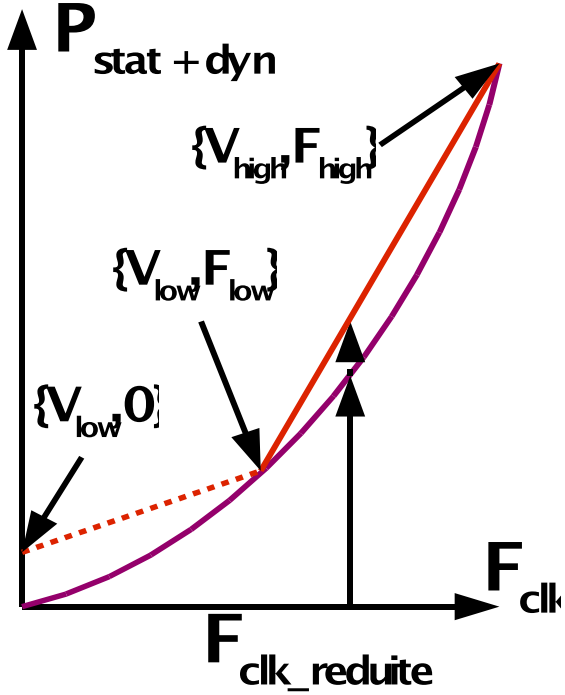
- Each Unit with its local voltage/frequency

E. Beigné, F. Clermidy, S. Miermont, P. Vivet, "Dynamic Voltage and Frequency Scaling Architecture for Units Integration within a GALS NoC", Proceedings of the 2nd IEEE International Symposium on Networks-on-Chip, NOCS'2008, New-Castle, UK, April 2008.

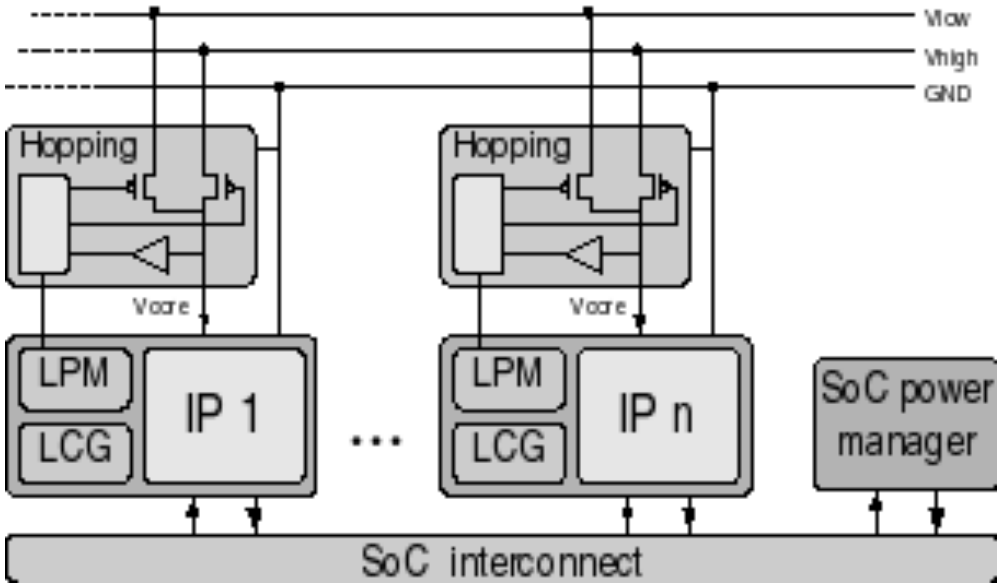
VDD Hopping : Principle

- Energy per operation scales with V^2
- Use of two PMOS power switches
  - Vhigh, Vlow : a discrete DVS
  - Switch between Vhigh and Vlow :
    - Smooth and fast transitions (less than 100 ns)
    - Programmable Duty Ratio





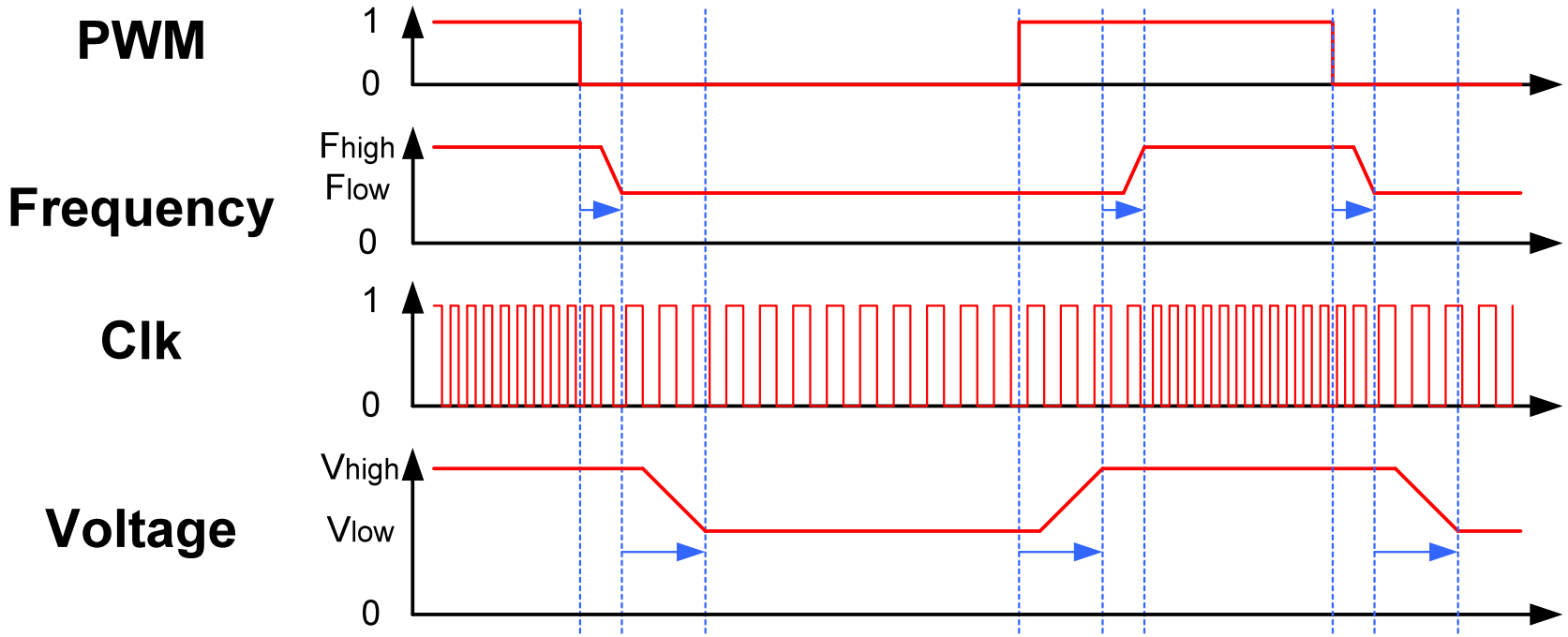
VDD-Hopping: distribution



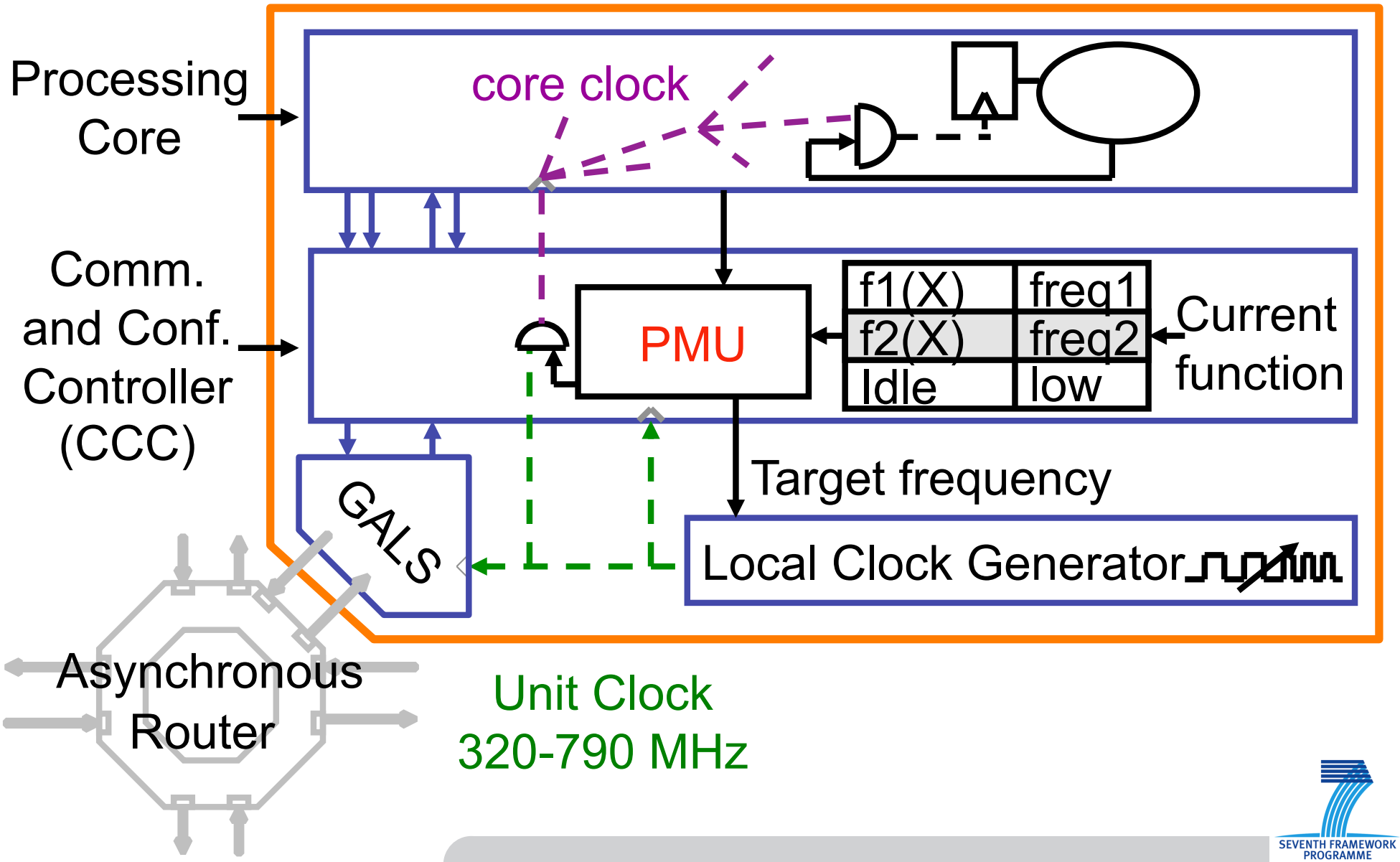
LPM : Local Power Manager
LCG : Local Clock Generator

- VDD-Hopping offers DVFS ***at IP level***
  - No need of inductor, capacitor, charge pump
  - Fully integrable
  - Low area (3% of IP area),
  - High Power Efficiency (95%)
  - Only requires two external supplies per IP :
    - Vhigh (nominal voltage) & Vlow (set wrt. to logic & SRAMs constraints)

VDD Hopping: clock management

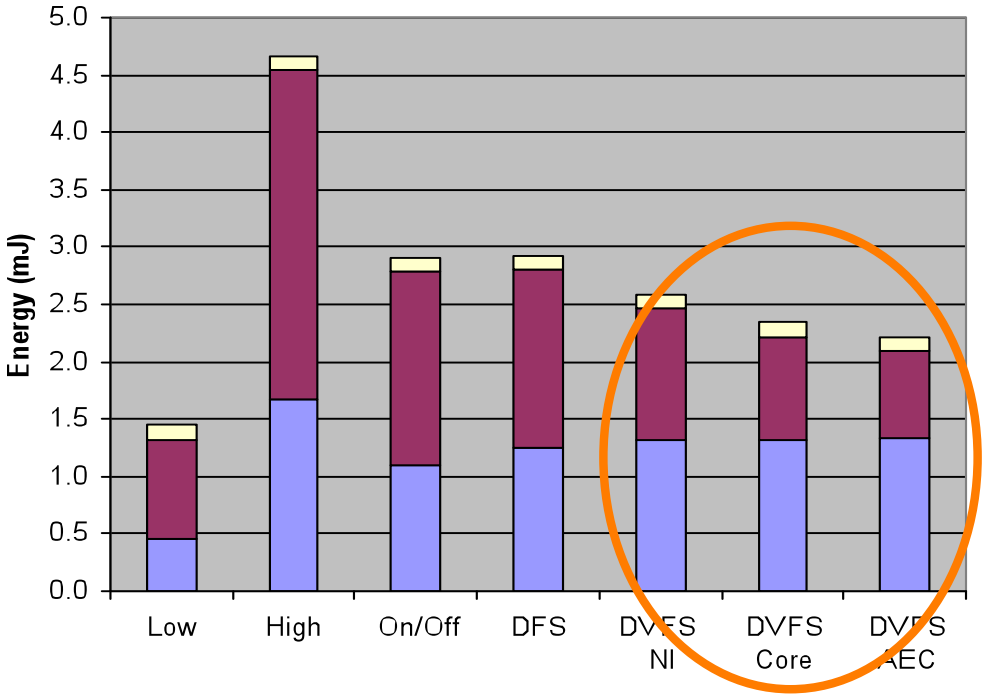


Resource power control



Exploration of VDD-Hopping benefits

- VDD-Hopping power reduction capabilities :
  - On-line dynamic slack time optimization : 30% gain wrt. static DVFS
  - DVFS compared to On/Off mode : 45% gain
  - Total chip budget : reduction from 340mW downto 160mW



3GPP-LTE Application (MAGALI)

SYSTEMC-TLM power simulation

Outline

- Context
- MAGALI overview
- NoC, GALS and Low-Power
- Dynamic reconfiguration
- Distributed decisions
- Configuring & programming
- Conclusion

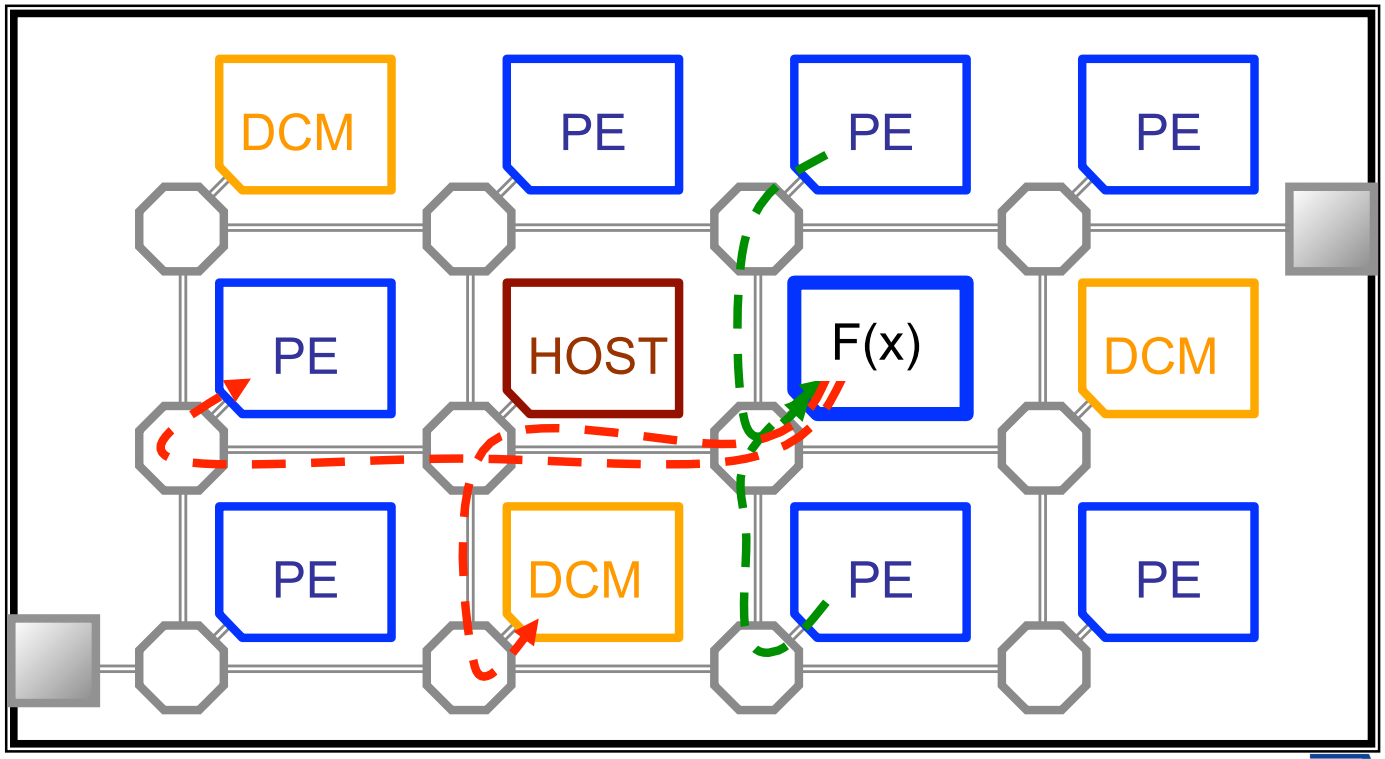
Semi-distributed control

- Data-flow directed synchronization (fork, join, loop) through each PE associated CC
- Complex data and flow mixing performed in DCM
- If more complex control => host control

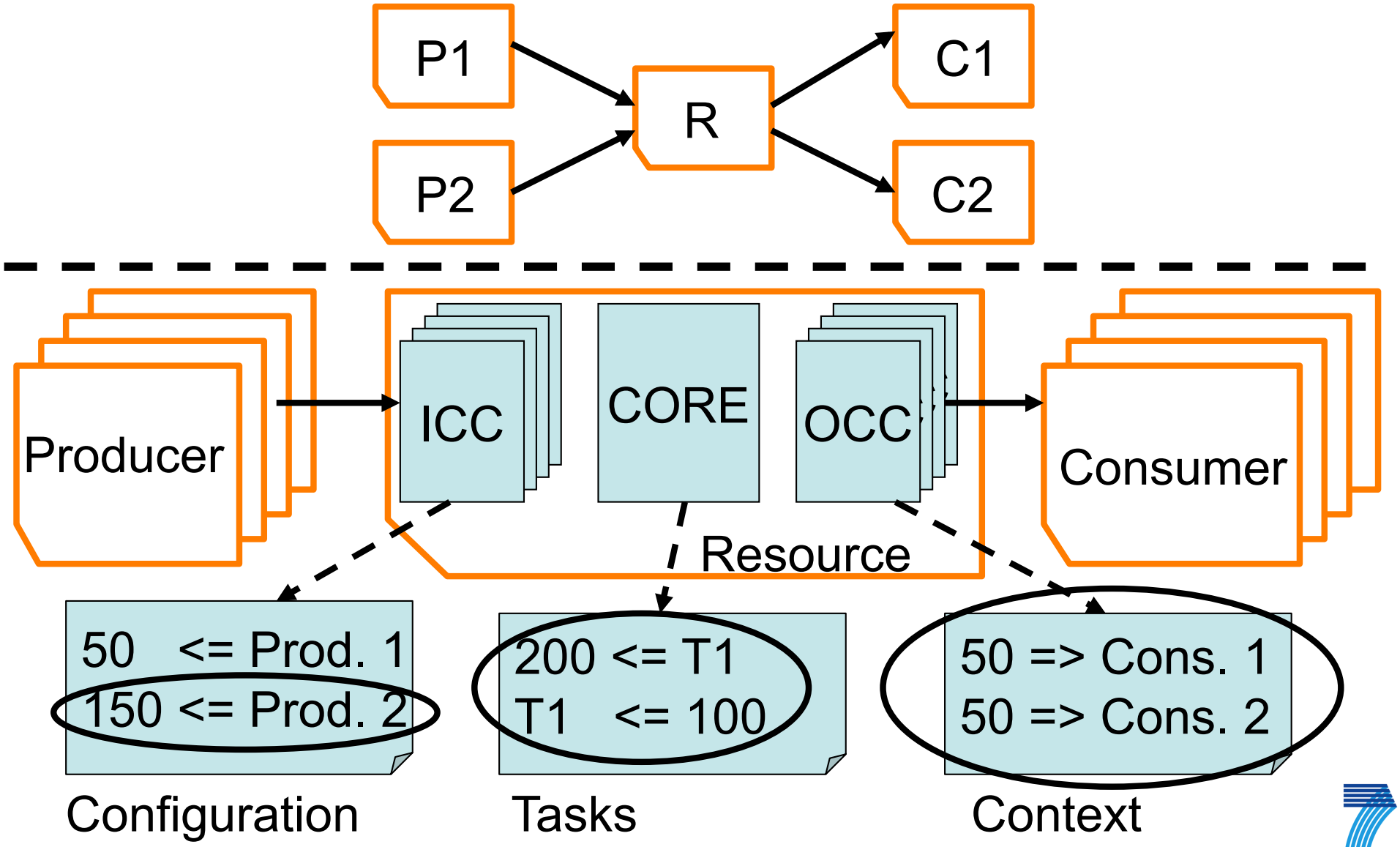
DCM = Data and Configuration Memory

CC = Communi-cation & Configuration controller

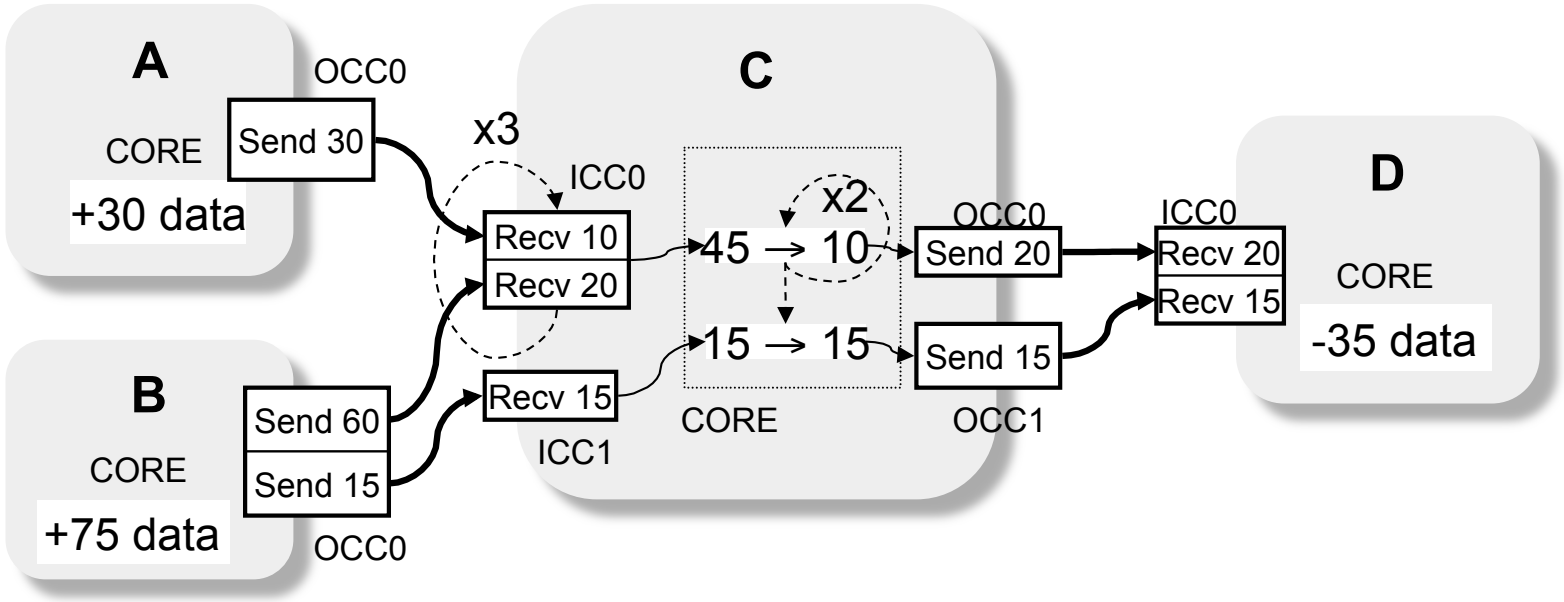
J. Martin et al., “A Microprogrammable Memory Controller for High-Performance Dataflow Applications”, ESSCIRC’09



Communication scheme



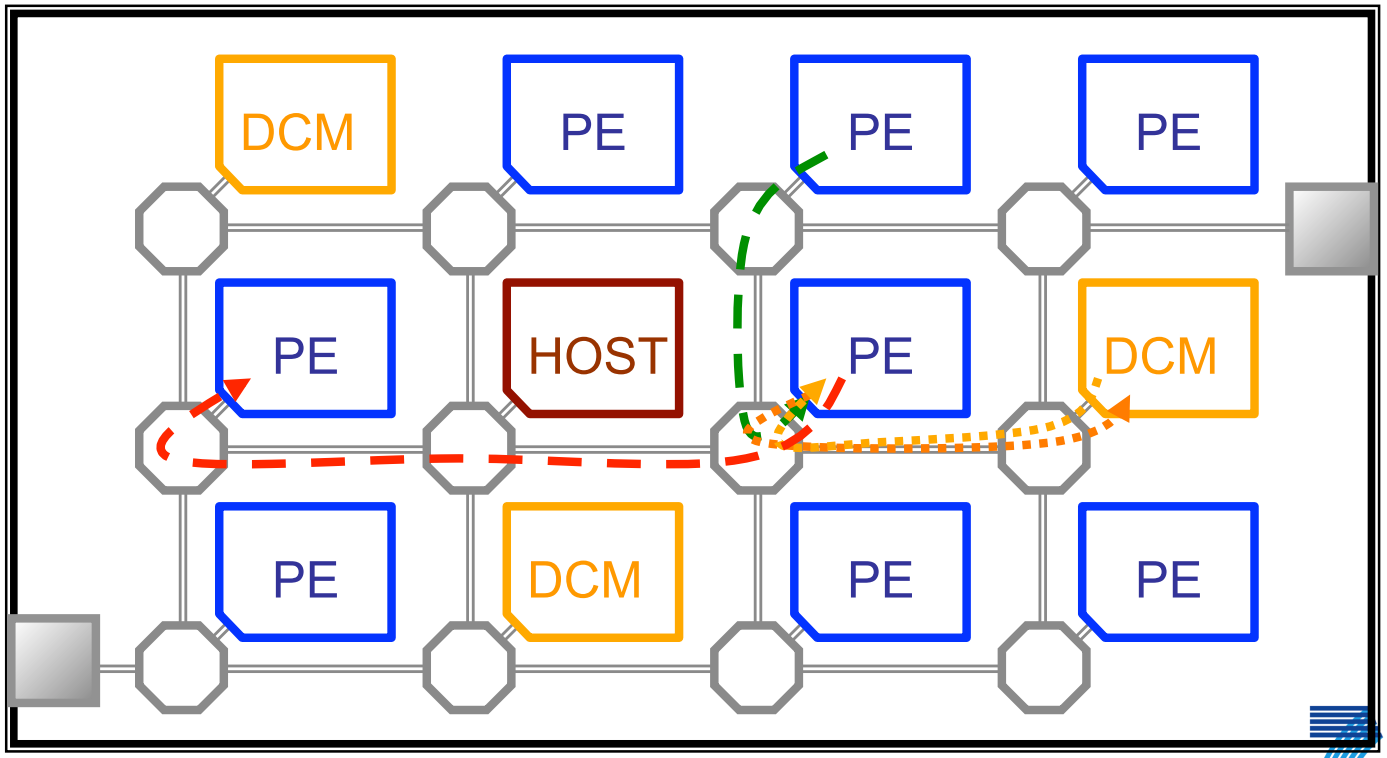
µProgrammed data synchronization



Mnemonic, operand(s)	Description
RC c s	Request configuration
RCL c s	Request configuration + Loop pointer
LL n	Go back to stored loop position. Loop **n** times
GL n	Go back to first instruction. Loop **n** times
LLi r	Go back to stored loop position. Loop number in register **r**
GLi r	Go back to first instruction. Loop number in register **r**
STOP	End of micro-program

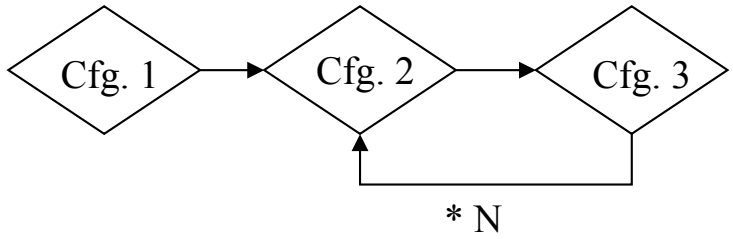
Dynamic reconfiguration

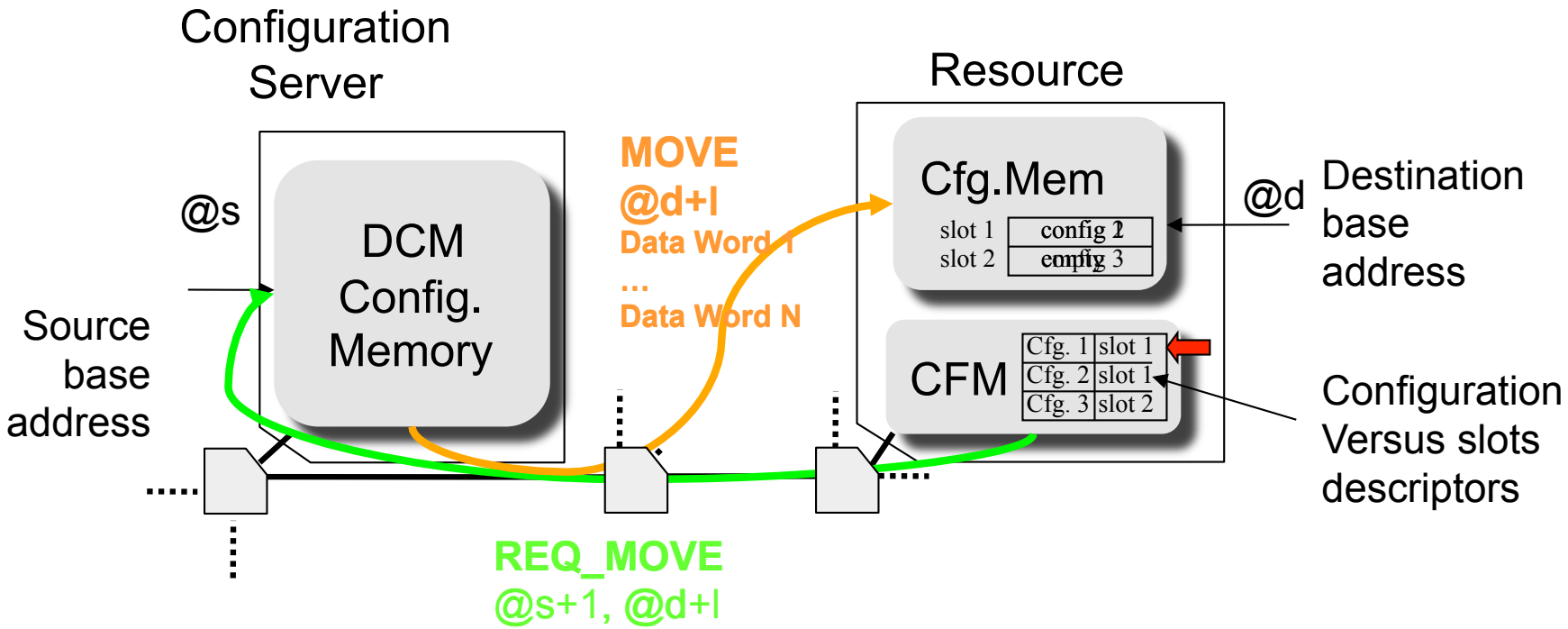
- PE configurations are stored in DCM memories
- When a PE has to run a configuration not loaded => ask to associated DCM
- Configurations can be modified online by the host



F. Clermidy et al. “A Communication and Configuration Controller for NoC based Reconfigurable Data Flow Architecture”, NOCS'09

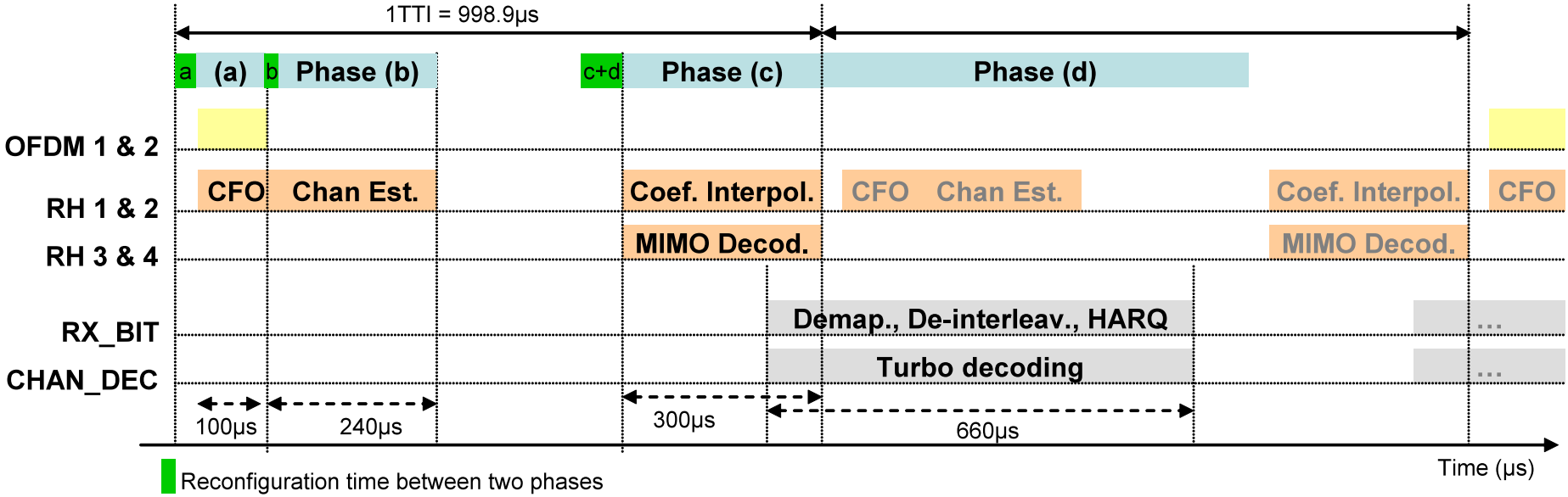
Self-configuration protocol





Some results: reconfiguration time

- 3GPP-LTE : RT-constraints 1 ms
  - 4 configuration phases
  - Most configuration time hidden by computation time



=> 4 µs reconfiguration time

Outline

- Context
- MAGALI overview
- NoC, GALS and Low-Power
- Dynamic reconfiguration
- Distributed decisions
- Configuring & programming
- Conclusion

Why distributing decisions?

- Number of cores is increasing => central decision is slow
- Process variations
- Increasing flexibility demand (applicationS)
- Individual optimization required
  - Power
  - Variability
  - Thermal
  - Real-time (reducing buffering needs)
- And at run-time!

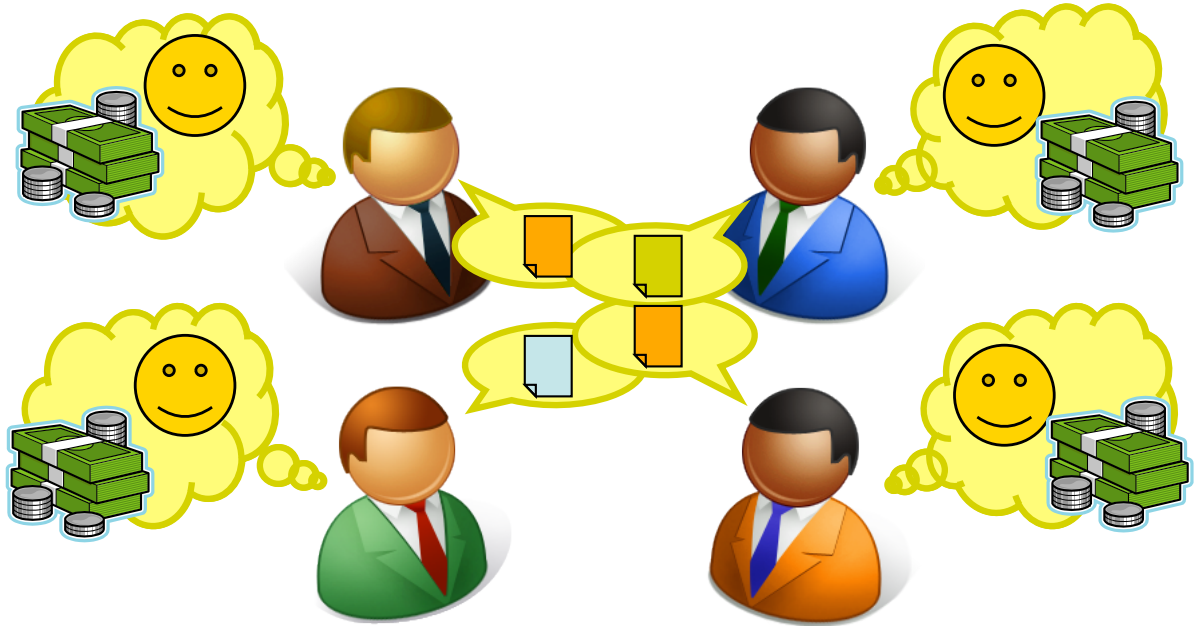
Options?

- Design-time optimization
  - Greedy algorithm, tabu search, simulated annealing, Genetic algorithms, Linear model optimization
  - ⇒ Processing requirement is too high for run-time usage
- Run-time optimization
  - Convex optimization, Non-linear lagrange optimization, Integer linear programming, Off-line exploration + on-line manager
  - ⇒ Centralized method: scalability of processing and communication?

=> So? Distributing centralized methods or optimizing distributed algorithms?

Distributed Scheme: Game Theory

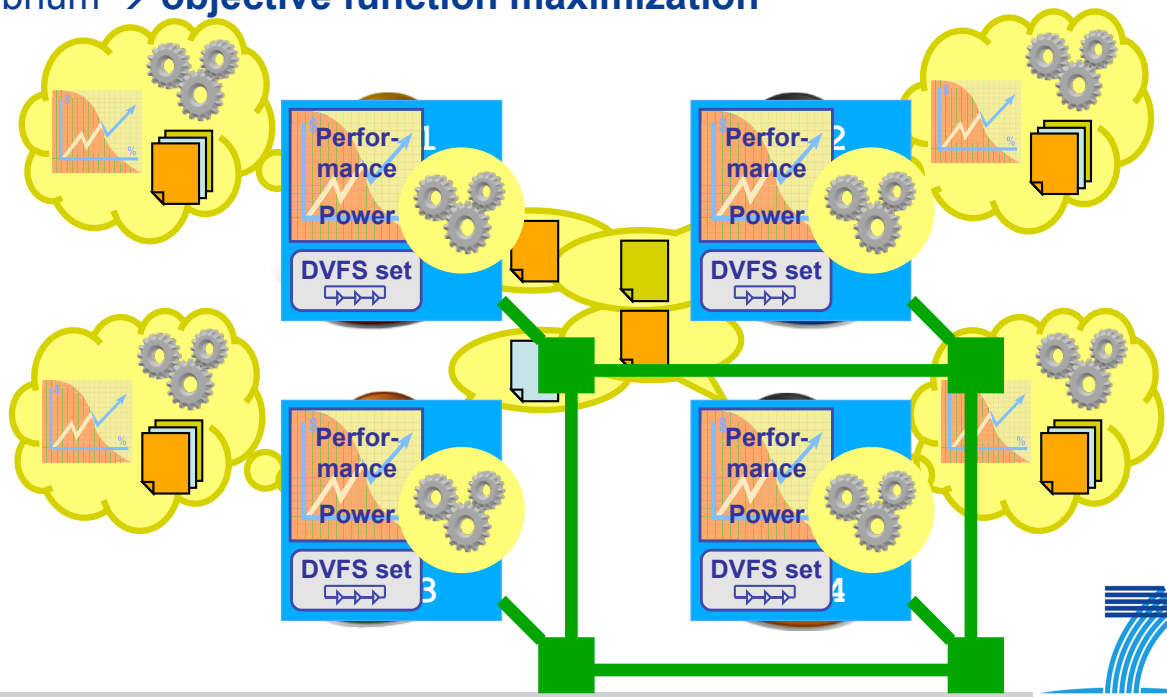
- Game Theory models:
  - **Players**
  - **Interacting** through actions
  - **Makes decisions** (distributed & parallel)
  - Maximizing **individual gain** (Objective Function)
  - Solution: **nobody can unilaterally improve his gain** (Nash Equilibrium)



Game Theory in MP-SoC

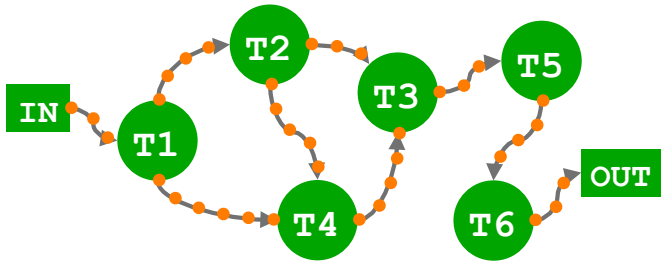
- Game Theory → MP-SoC
  - Players → **PE**
  - Actions → PE **parameters** (e.g.: frequency)
  - Decision making → **actuators** in PE (e.g.: DVFS)
  - Individual gain → **objective function** per PE (e.g.: performance, power)
  - Solution: Nash Equilibrium → **objective function maximization**

- So... what do we need?
  - Distributed Objective Function
  - Local Maximization Algorithm



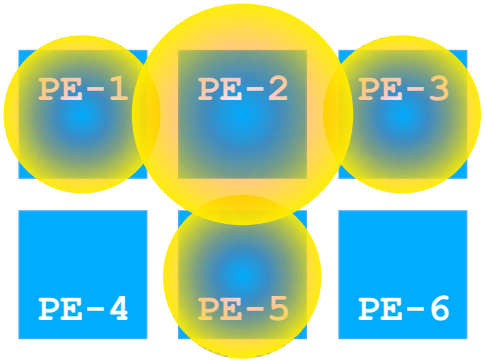
Thermal management example

Applicative



Synchronization → Frequency

Technological



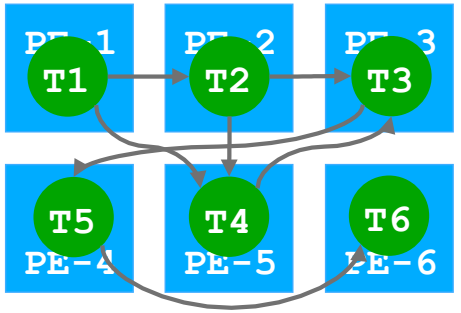
Temperature → Frequency

Applicative + Technological

How do we set **Frequencies**

to optimize **Synchronization** +

Temperature?



Temperature optimization

Best latency → Best temperature

	Configuration 1 ($w_{syn} = 1, w_{temp} = 0$)		Configuration 2 ($w_{syn} = 0.75, w_{temp} = 0.25$)		Configuration 3 ($w_{syn} = 0.5, w_{temp} = 0.5$)		Configuration 4 ($w_{syn} = 0.25, w_{temp} = 0.75$)	
PE	GT Algo.	Matlab	GT Algo.	Matlab	GT Algo.	Matlab	GT Algo.	Matlab
1	35.0°C	35.9°C	35.0°C	34.2°C	35.0°C	33.2°C	34.0°C	33.2°C
2	48.1°C	49.5°C	45.7°C	45.8°C	42.7°C	42.1°C	41.4°C	38.2°C
3	38.6°C	39.6°C	36.4°C	36.2°C	34.3°C	32.4°C	33.3°C	27.0°C
4	37.1°C	38.1°C	35.0°C	35.9°C	32.0°C	34.2°C	31.7°C	31.9°C
5	53.7°C	55.2°C	47.3°C	50.7°C	38.2°C	45.8°C	37.9°C	38.8°C
6	45.3°C	46.6°C	39.6°C	42.4°C	31.7°C	37.5°C	31.4°C	29.8°C
T. Avg	43.0°C	44.1°C	39.8°C	40.9°C	35.6°C	37.5°C	34.9°C	33.1°C
T. Gain	-	-	7.95%	7.99%	23.94%	17.63%	23.02%	33.11%

Different trade-offs between application latency and temperature

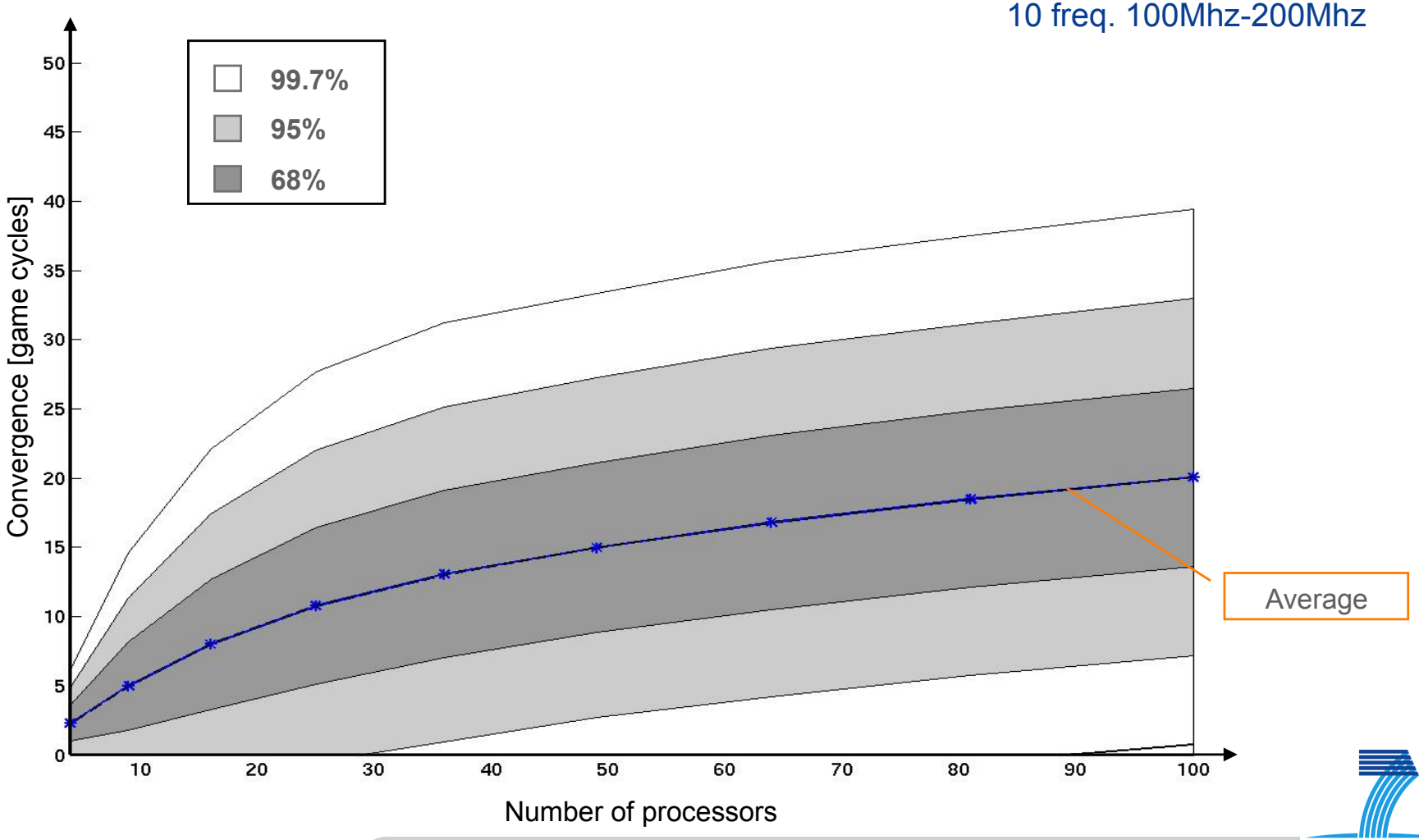
Convergence & Scalability

Convergence does not explode with the number of processors!

300000 scenarios

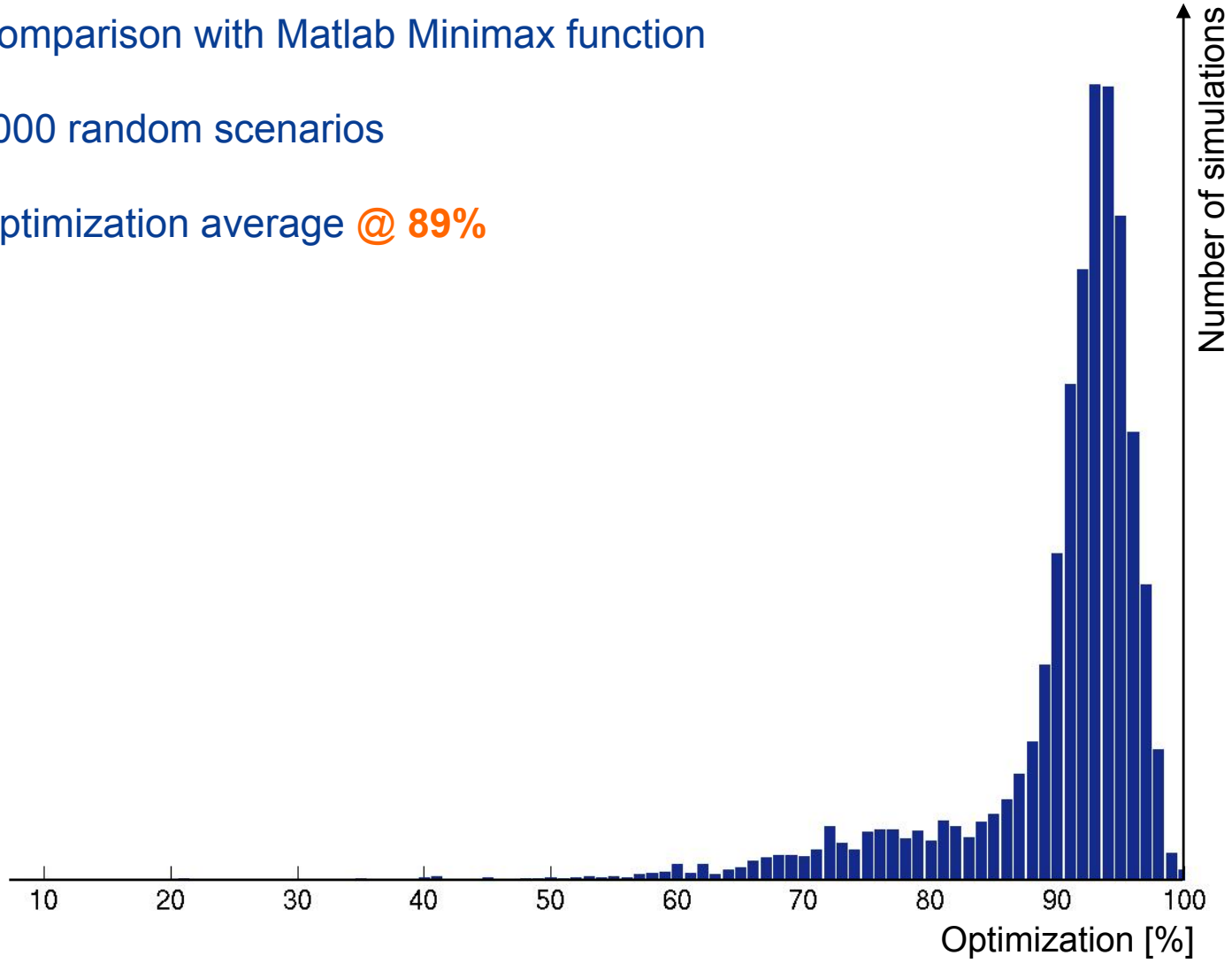
Synthetic applications

10 freq. 100Mhz-200Mhz



Optimality study

- Comparison with Matlab Minimax function
- 8000 random scenarios
- Optimization average @ **89%**



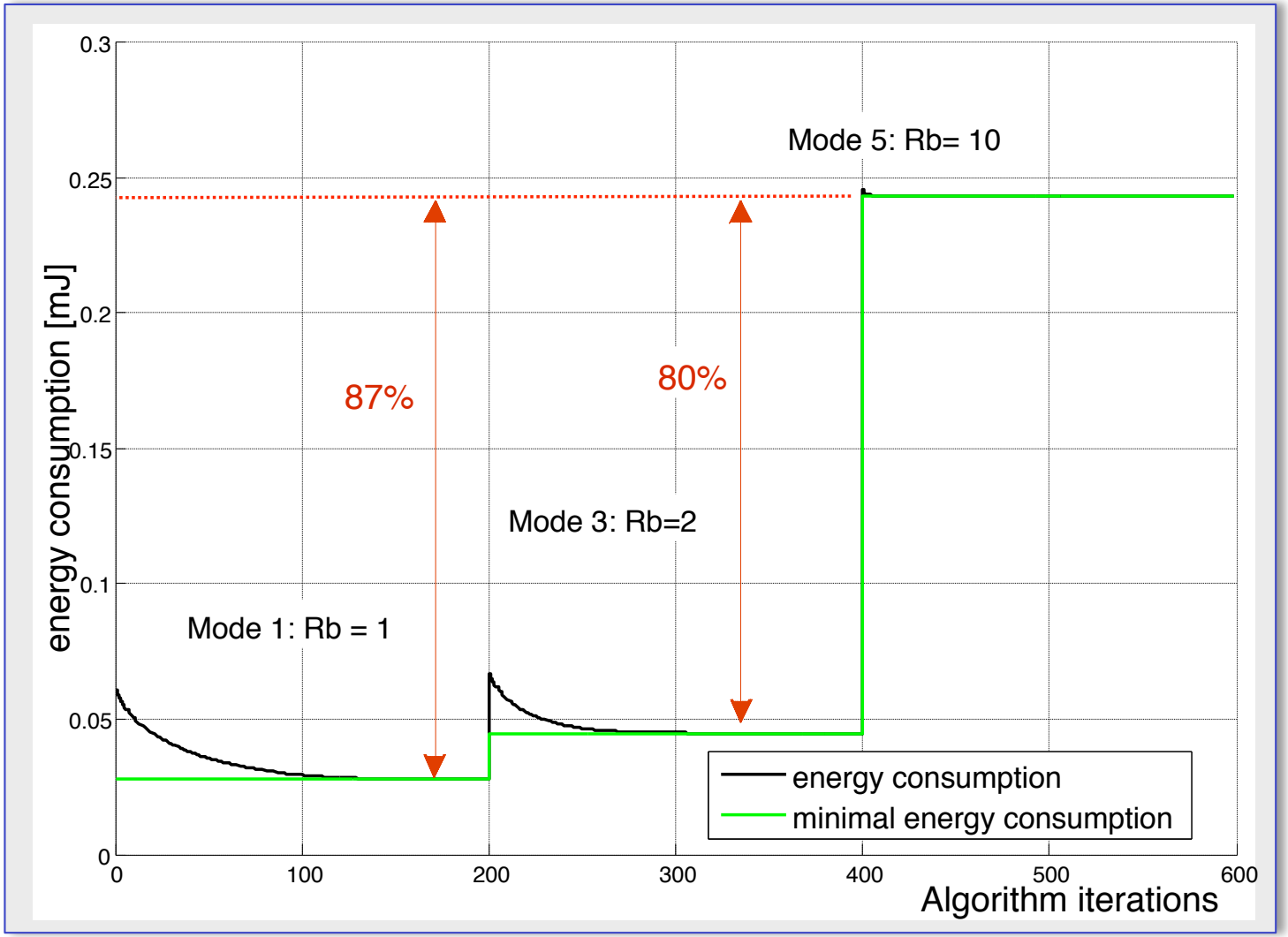
Implementation (65nm)

- Local Decision Maker (LDM)

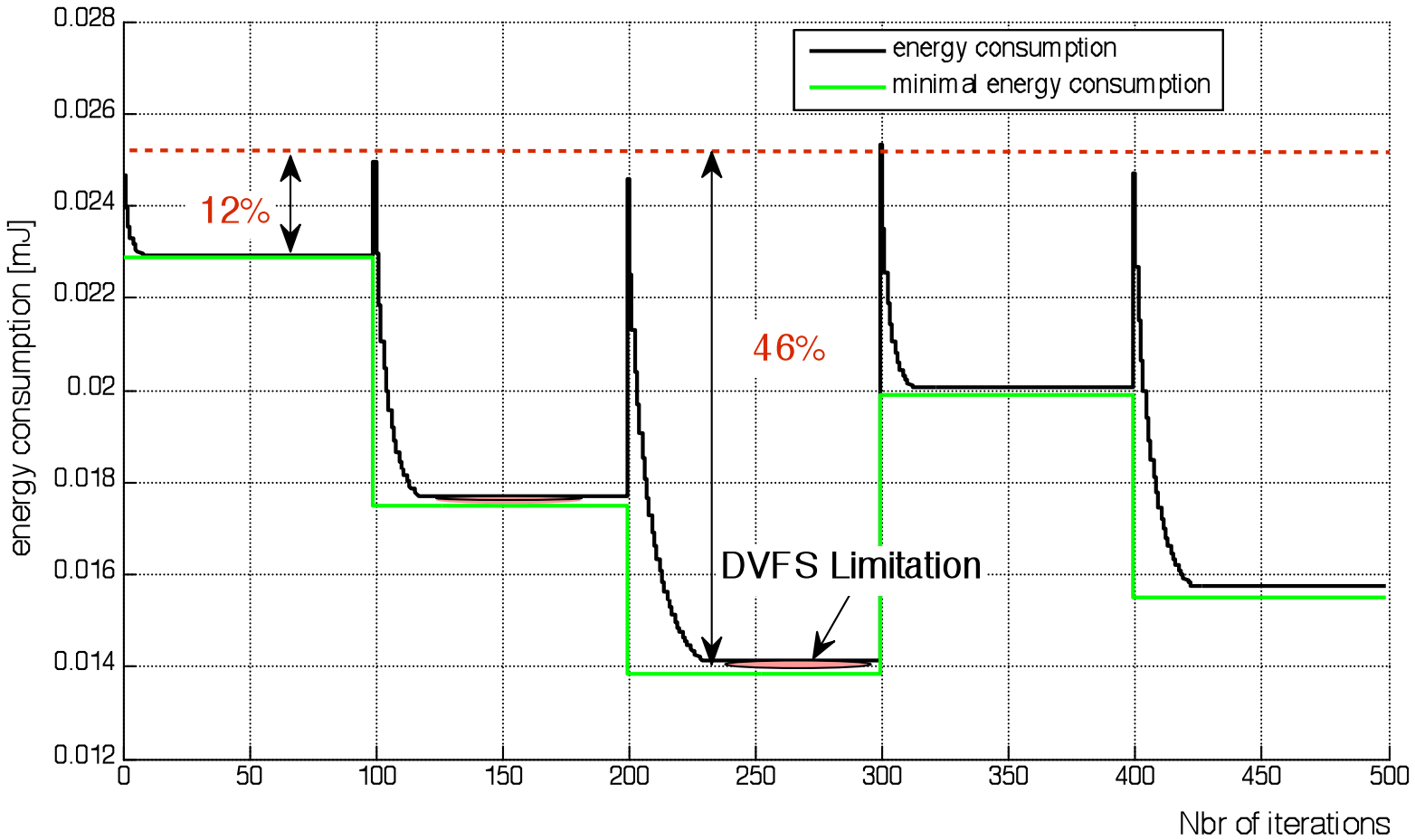
Criteria	µprog MIPS	HW Matlab model	HW Optimized model
Frequency [MHz]	400	25	100
Performance overhead [Game cycle duration]	2420	461	752
Area overhead (mm^2)	0.122	0.061	0.014
Com.overhead (clk cycles)	58	7	7

- Reactivity time of the controller is about **5ms**
- Throughput degradation: **0.17%**

Power Management using Consensus



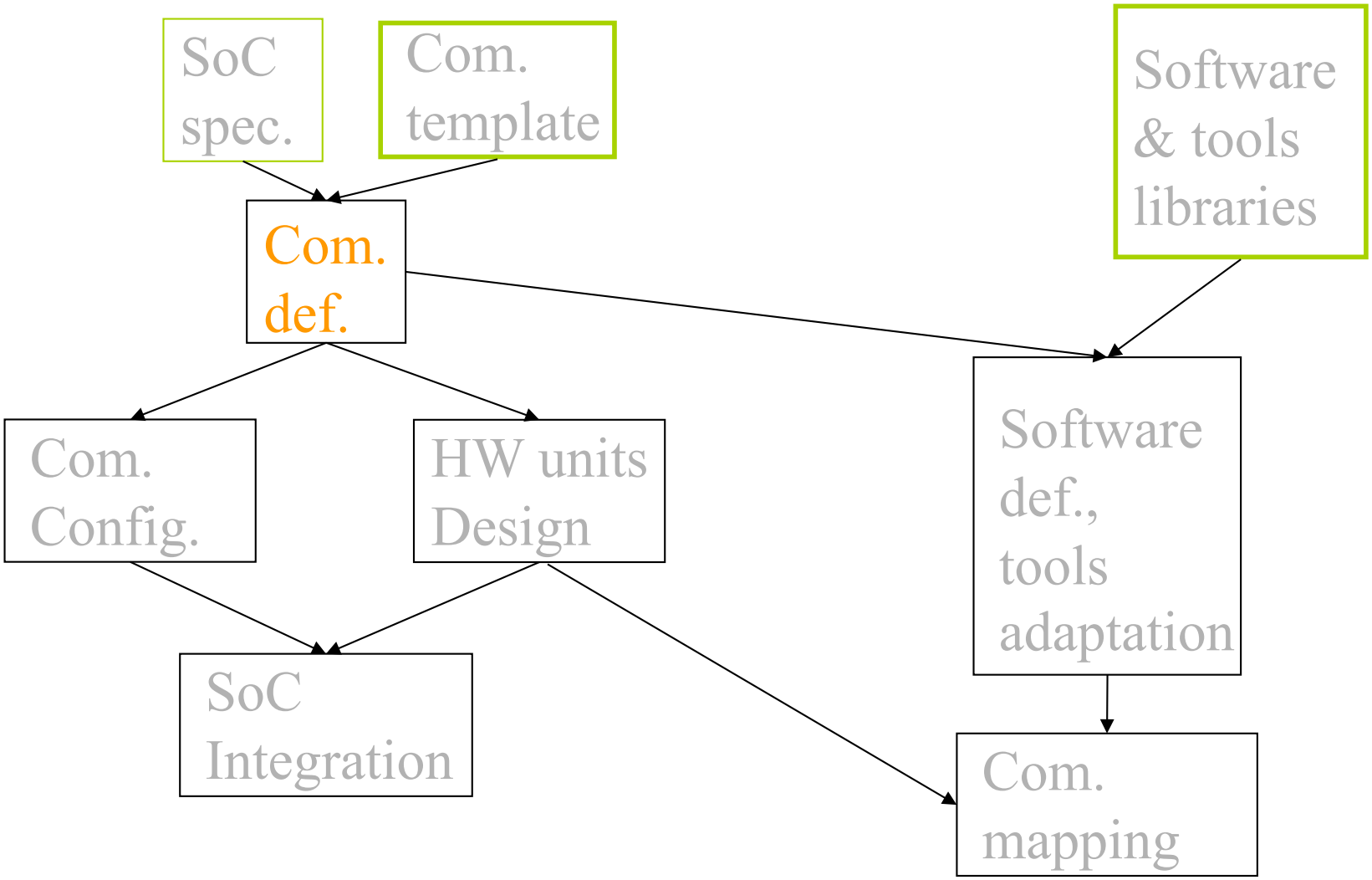
Modifying latency constraints on-line



Outline

- Context
- MAGALI overview
- NoC, GALS and Low-Power
- Dynamic reconfiguration
- Distributed decisions
- Configuring & programming
- Conclusion

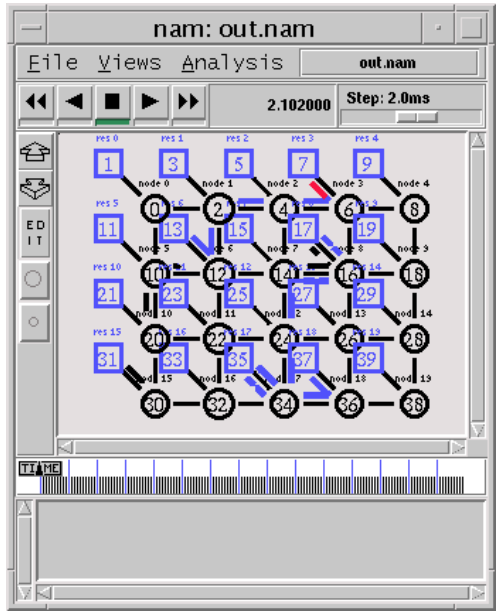
Programming Steps



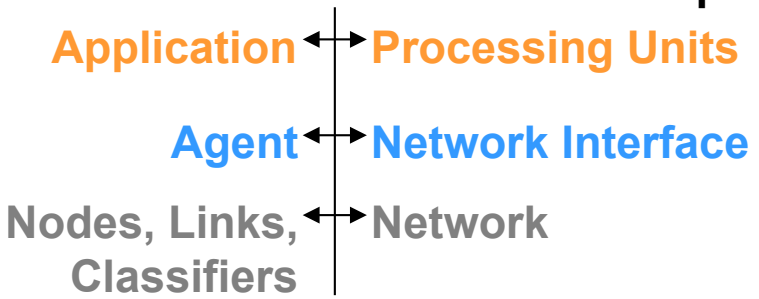
NS-2 Modeling

- NS-2 components adaptation
- **Network design**
  - 2D-mesh, packet-switching
- **Units + Network Interface design**
  - **Network Interface**: use of **Agent** component for modeling the protocol
  - Generic **processing units**: dataflows modeling (**Application** component)
  - Configuration parameters

NAM view

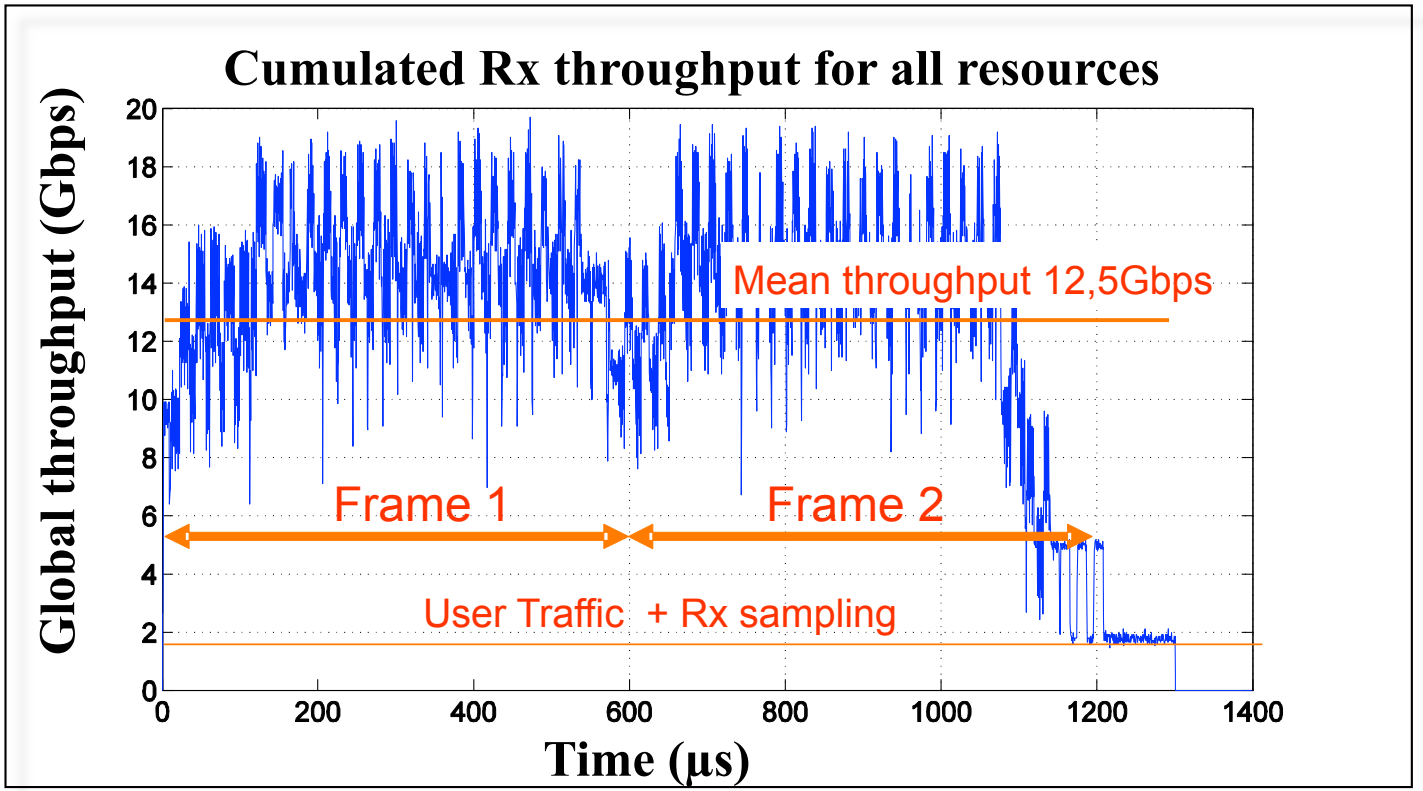


NS-2 / NoC Relationships



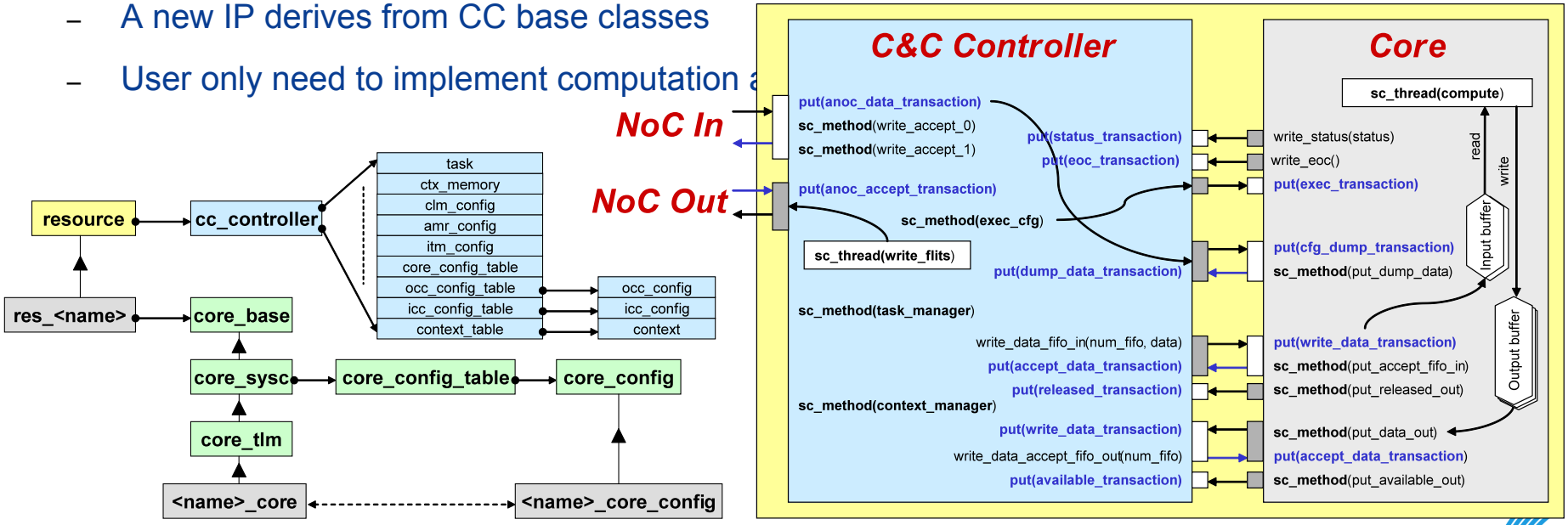
Results

- **Applicative throughput**
  - Cumulate throughput for each resource
  - Maximum value: 20 resources × 3,2Gbps (100MHz) → 64Gbps
  - Simulation: maximum throughput → 20Gbps
  - NoC is needed for such application



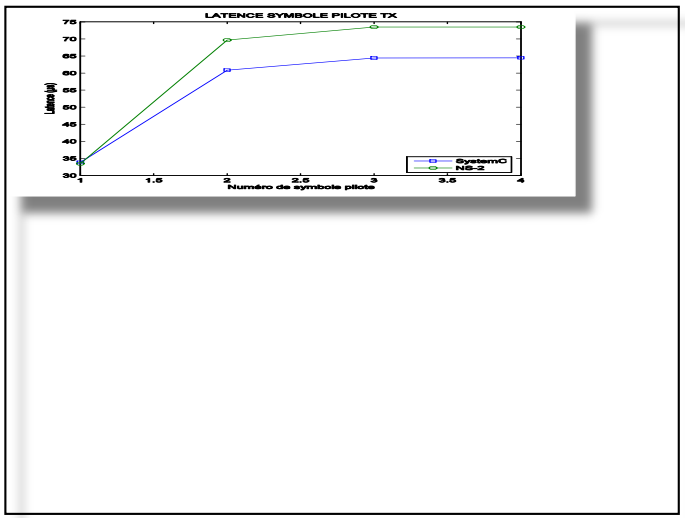
SystemC-TLM environment

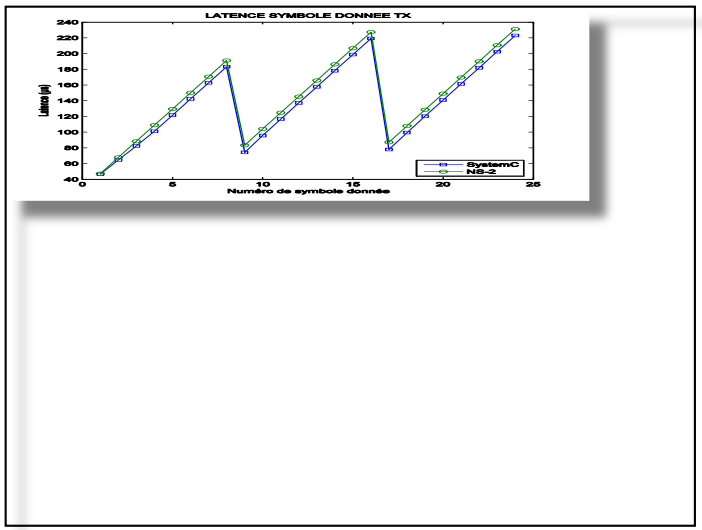
- Generated from the IP-XACT Magillem tool
- Complete NoC SystemC/TLM Platform
  - Based on SystemC 2.1 + TLM OSCI 2.0 draft + ST TLM devkit
  - Include NoC nodes + CC controller
- IP integration within NoC ?
  - A new IP derives from CC base classes
  - User only need to implement computation a



NS-2 / SystemC Comparison

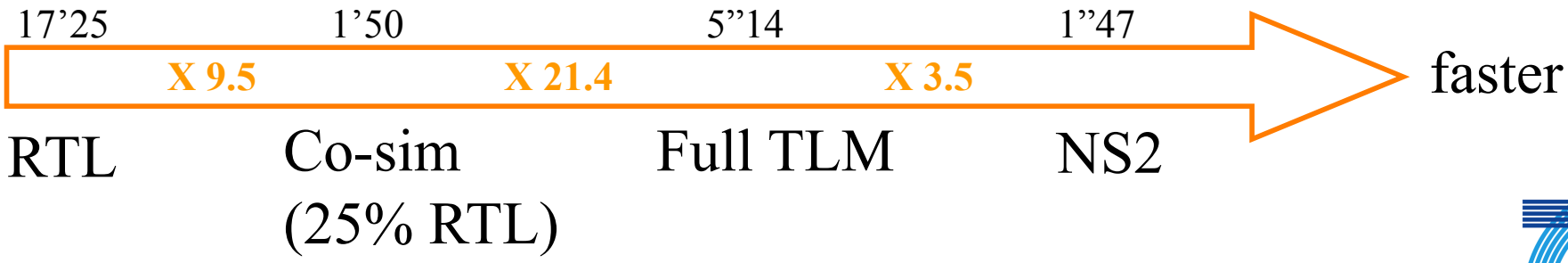
- **Comparison NS-2 with SystemC model (behavioral)**



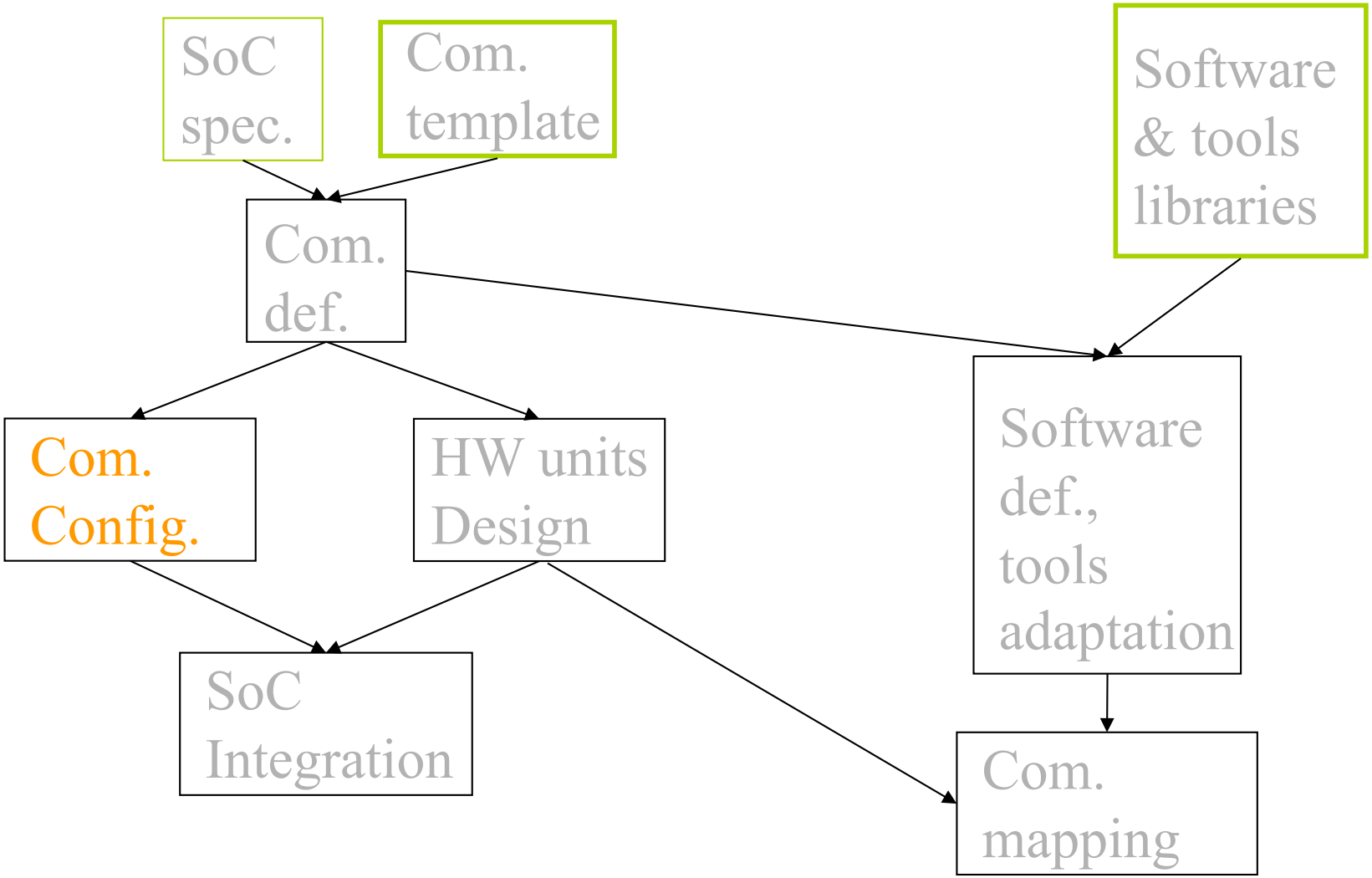


  - 15 % differences. Due to switching mode modeling in NS2

- **Simulation time: time needed to decode a 3GPP-LTE Frame**



Programming Steps



NI automatic generation

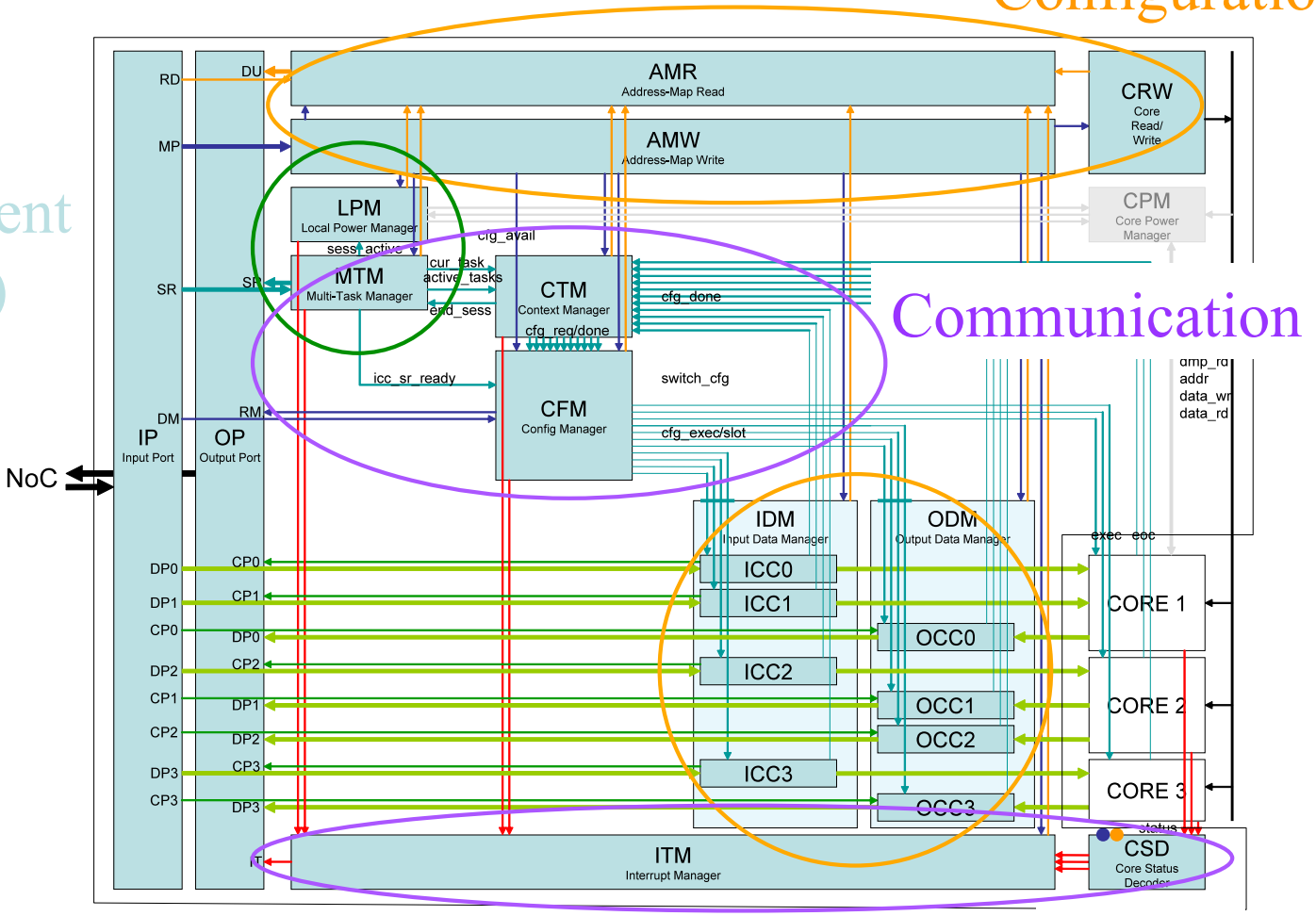
- Communication & Configuration (CC) controller = NI + high level communication and configuration primitives

- Numerous parameters
  - Fundamental
    - Cores number
    - Input/output flows
  - Level of functionalities
    - Context size
    - Number of configuration
  - Power management
    - Global gated clock enable

CC Micro-Architecture and Design

Configuration

Power Management (DVFS)



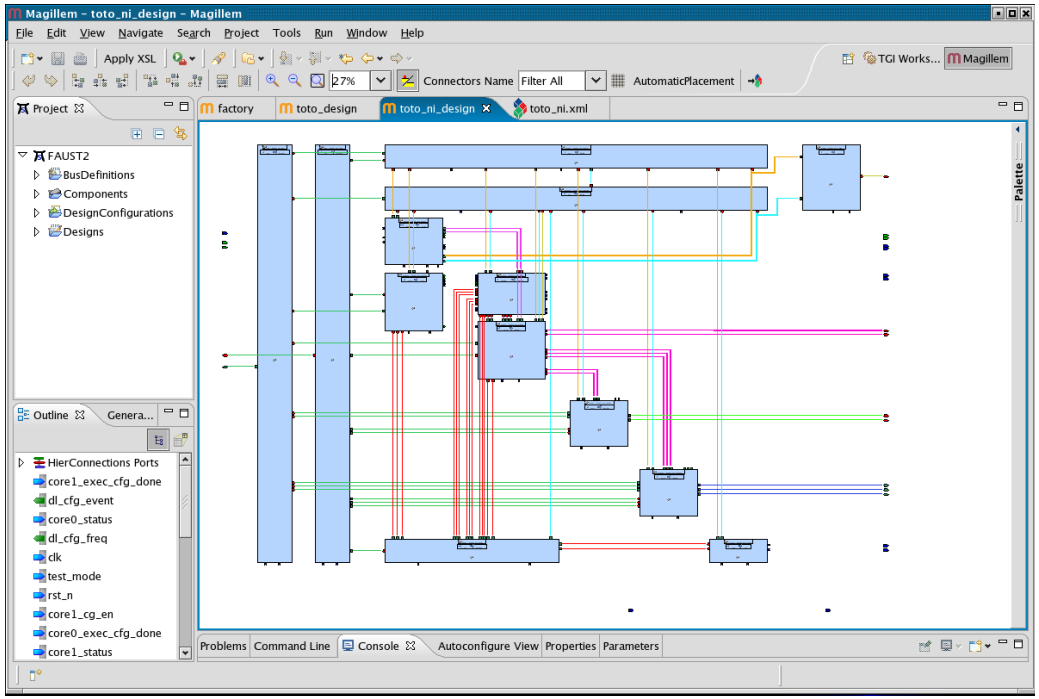
Communication

QoS

Debug

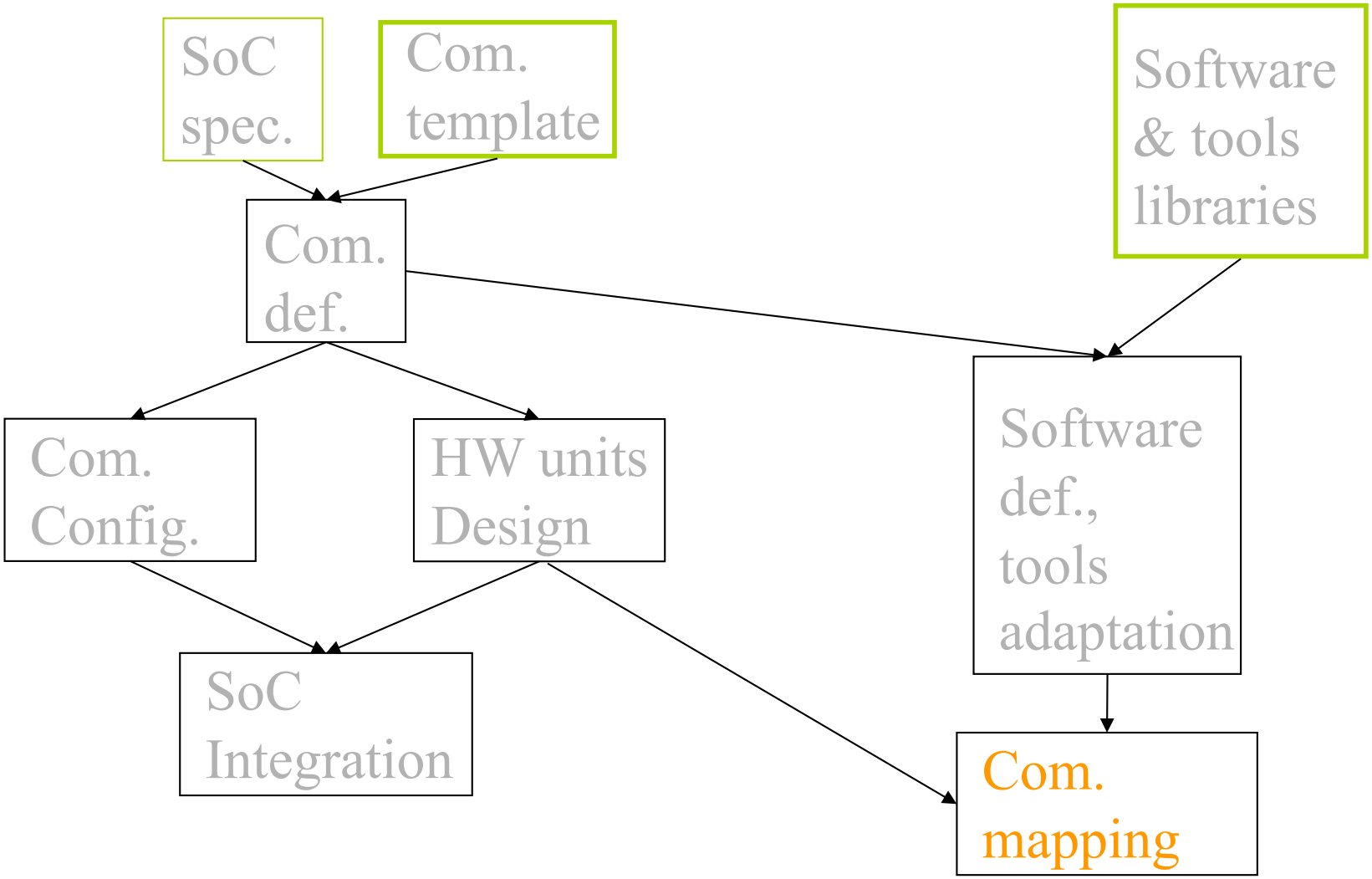
CC controller generation

- All CC blocs are IP-XACT compliant
- Magillem tool (MDS collaboration)
  - Generator to create a CC: TGI interface

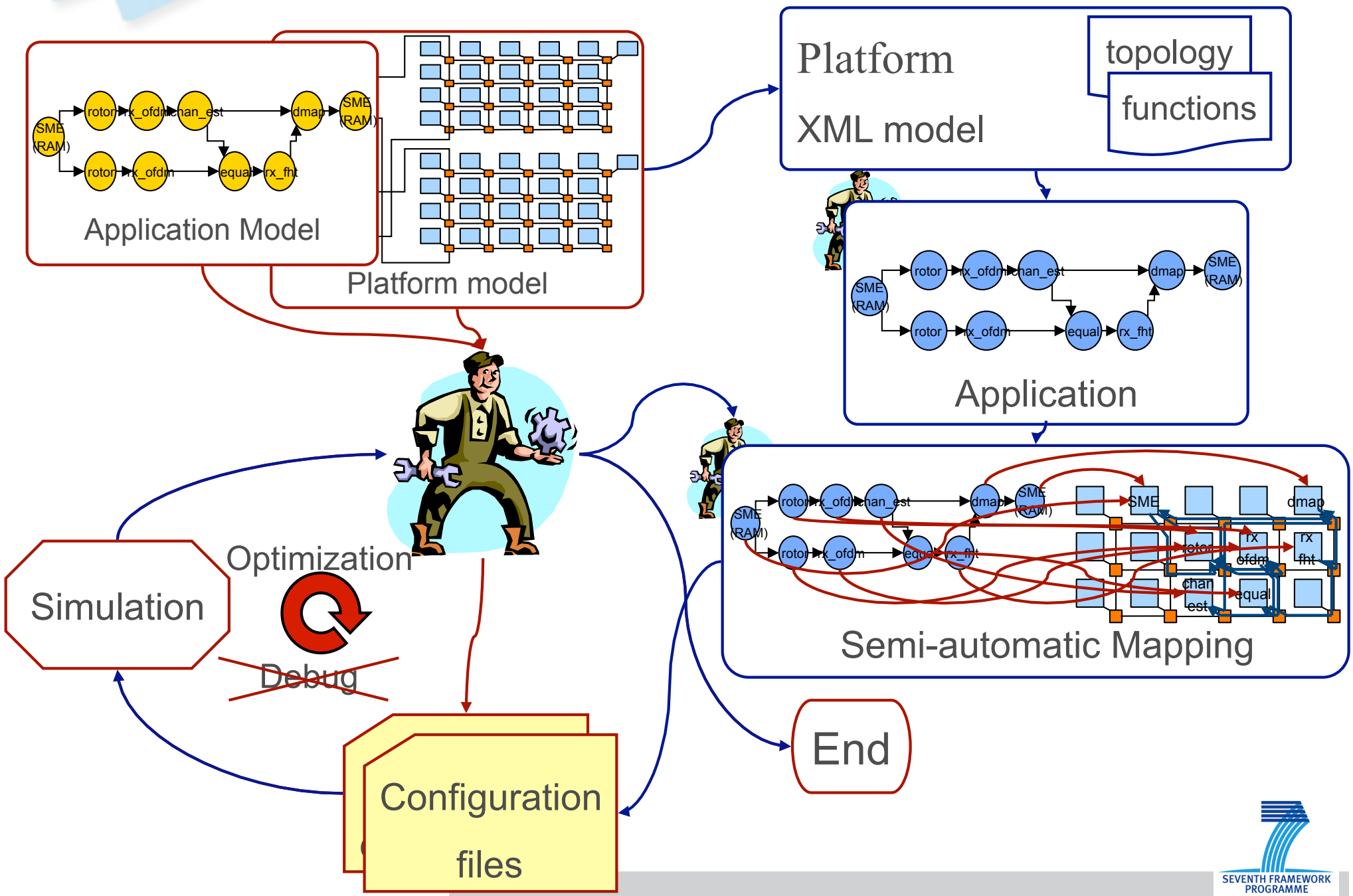



```
# Parameter  = Value   ; # Range or values
$unit_name = trx_ofdm ;
$nb_cores       = 1       ; # 1 .. 4
$nb_fifo_in     = 2       ; # 1 .. 4
$nb_fifo_out    = 2       ; # 1 .. 4
$nb_cfg_icc     = 6       ; # 1 .. 2^nb_bits_slot_id
$nb_cfg_occ     = 6       ; # 1 .. 2^nb_bits_slot_id
$default_size_available_fifo_in[0]     = 16      ; # <2^16 (default 9)
$nb_bits_size_available_fifo_in[0]     = 8       ; # 1 .. 16 (default 8)
$nb_bits_size_released_fifo_in[0]      = 8       ; # 1 .. 16 (default 9)
$default_size_available_fifo_in[1]     = 16      ; # 1 .. 16 (default 9)
$nb_bits_size_available_fifo_in[1]     = 8       ; # 1 .. 16 (default 8)
$nb_bits_size_released_fifo_in[1]      = 8       ; # 1 .. 16 (default 9)
$nb_bits_size_available_fifo_out[0]    = 5       ; # 1 .. 16 (default 9)
$nb_bits_size_available_fifo_out[1]    = 5       ; # 1 .. 16 (default 9)
$core_name[0]    = trx_ofdm ;
$core_binding_fifo_in[0]        = [0,1]   ;
$core_binding_fifo_out[0]       = [0,1]   ;
$nb_bits_core_status[0]         = 16      ; # 1 .. 32
$core_cfg_begin[0]              = 0       ; # 0 .. 2^nb_bits_core
$nb_bits_core_addr[0]           = 10      ; # 1 .. 21 (default 8)
$core_cfg_size[0]               = 8       ; # 1 .. nb_bits_core_addr
$nb_cfg_core[0]                 = 3       ; # 1 .. 2^nb_bits_slot_id
$has_gc_en_core[0]              = 1       ; # 0 1
$nb_bist_elements[0]            = 14      ; # user-def
$scan_counter_width             = 9       ; # user-def
```

Programming Steps



NoC programming general scheme



Programming choices rational

- Limited Memory
  - Off-line computing of communications
  - On-line full programming
- Numerous parameters for one application
  - But few local adaptations
- Fast Reconfiguration

=> Off-line computing with On-line adaptation

Mapping and configuration manipulation

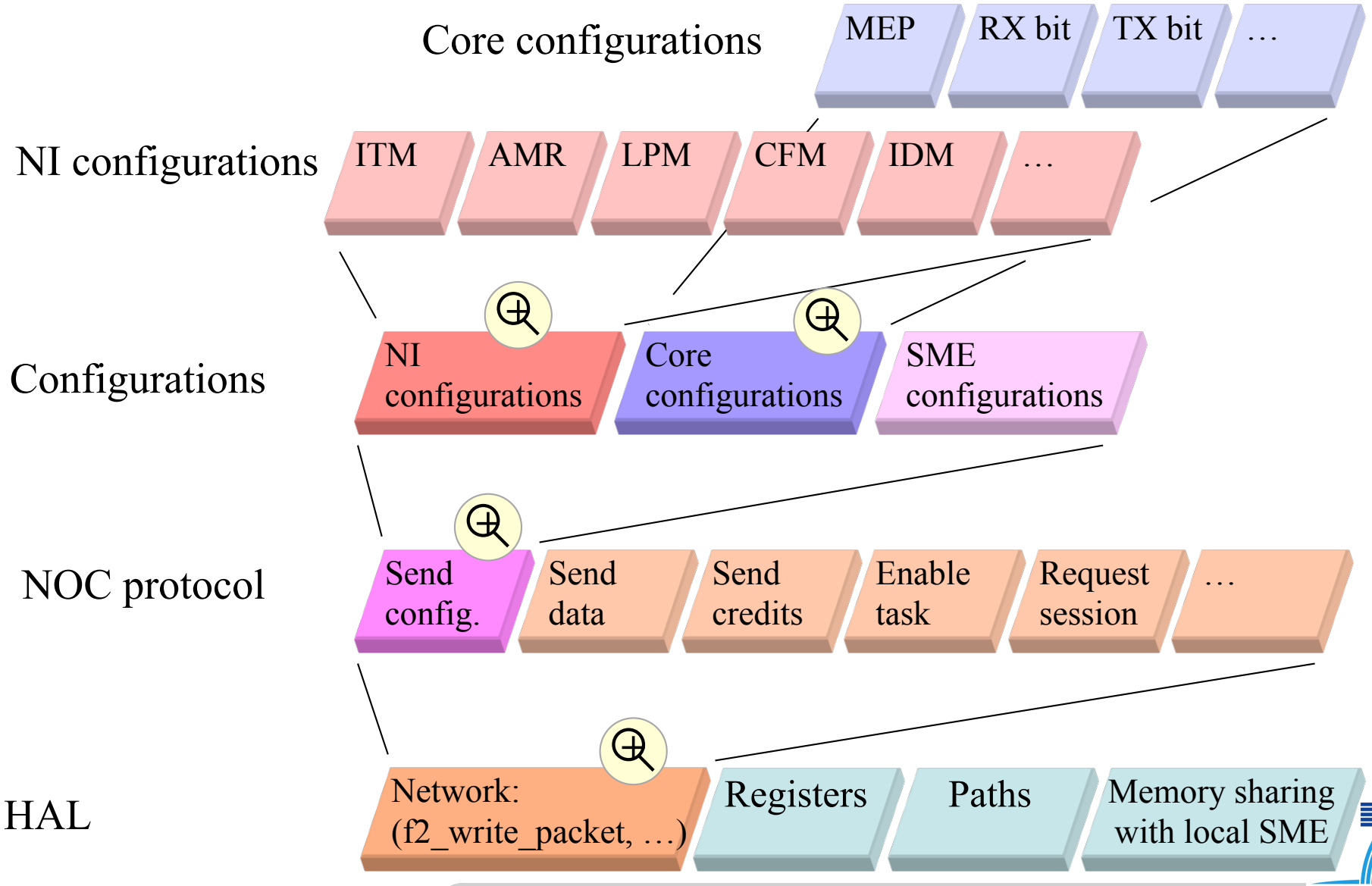
- Bottom up view
  - SW libraries for programming the communications, the HW IP, …, at several levels.

⇒ Communication and configuration APIs (F2 APIs)

- Top down view
  - High-level models
  - Tools for mapping the application on the hardware

⇒ Communication compiler (Comc)

F2 APIs : layered architecture



F2 APIs : description of the register map

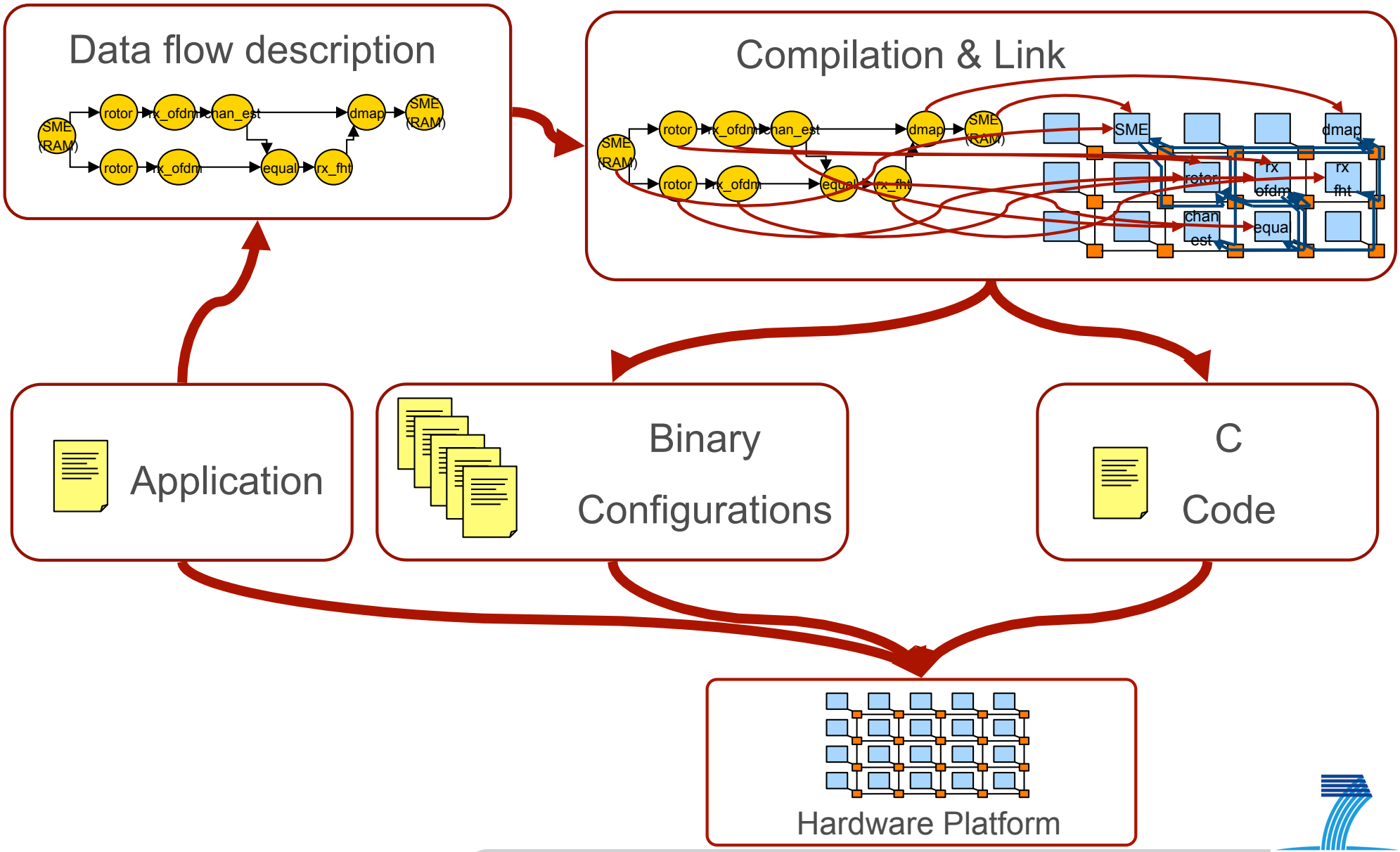
- Macros generated from IP-XACT
- Example for the NI (partial):

```
/* Definitions for block ITM_CONFIG */
#define ITM_CONFIG_OFFSET                                   0x00
#define ITM_CONFIG_RANGE                                    2

  /* Definitions for register CONFIG_1 */
  #define ITM_CONFIG_CONFIG_1_OFFSET                        0x00

    /* Definitions for register field CHANNEL */
    #define ITM_CONFIG_CONFIG_1_CHANNEL_OFFSET              31
    #define ITM_CONFIG_CONFIG_1_CHANNEL_SIZE                1
    #define ITM_CONFIG_CONFIG_1_CHANNEL_SET(_val_)      \
            SET_VAL(_val_,ITM_CONFIG_CONFIG_1_CHANNEL)

    /* Definitions for register field SOURCE_ID */
    #define ITM_CONFIG_CONFIG_1_SOURCE_ID_OFFSET            18
    #define ITM_CONFIG_CONFIG_1_SOURCE_ID_SIZE              7
    #define ITM_CONFIG_CONFIG_1_SOURCE_ID_SET(_val_)    \
            SET_VAL(_val_,ITM_CONFIG_CONFIG_1_SOURCE_ID)

    /* Definitions for register field PATH_TO_TARGET */
    #define ITM_CONFIG_CONFIG_1_PATH_TO_TARGET_OFFSET       0
    #define ITM_CONFIG_CONFIG_1_PATH_TO_TARGET_SIZE         18
    #define ITM_CONFIG_CONFIG_1_PATH_TO_TARGET_SET(_val_)   \
            SET_VAL(_val_,ITM_CONFIG_CONFIG_1_PATH_TO_TARGET)
```

Comc : goals

- Ease the tasks of the SW developer, by using a functional description of the data flow
- Hide the complexity due to the architectural concepts
- Allow to describe parameterized configurations

Communication mapping workflow



Conclusion

- NoC-Based embedded system is a paradigm shift
  - Communication-centric scheme
  - Large choices and optimization possibilities
  - Implementation (GALS, Low-Power)
- Scalability leads smartness to go to lower levels
  - Control
  - Reconfiguration
  - Decisions
- Programmability of heterogeneous platforms is key

Leti

LABORATOIRE D'ÉLECTRONIQUE
ET DE TECHNOLOGIES
DE L'INFORMATION

CEA-Leti
MINATEC Campus, 17 rue des Martyrs
38054 GRENOBLE Cedex 9
Tel. +33 4 38 78 36 25

www.leti.fr

Thank you

cea
énergie atomique • énergies alternatives

MINATEC

INSTITUT CARNOT CEA LETI