

Link between programming model and NoC design

GIPSA-Lab/Grenoble-INP

D. Houzet, S. Mancini

ENSTA-Telecom/ParisTECH

Y. Mathieu, O. Hammami

PhD Students: Z. Larabi, Z. Wang, G. Oshiro

SEMBA 2009

Complex Digital Chip

Processing:

- task mapping
- scheduling

HW /
SW

Communication:

- routing
- scheduling
- caching
- synchro
-

NoC

Memory:

- data mapping

On / Off
chip

20 yrs experience on multiprocessors design and
NoCs since 2000 :

μSpider with LESTER

FAUST with CEA (4more project)

1st ARTERIS implementation on 45nm

Hermes with Porto Allegre Univ.

PCI-express on Chip experience

Spacewire, 1553 with M3 Systems

Outline

- SoC Programming Models
- 1) NoC features
- 2) Data accesses
- 3) Exploration
- Conclusion

Programming Model

- Deals with:
 - Communication
 - Synchronization
 - Data access
 - Scheduling
- Big Gap between Programming models for general purpose multiprocessors and hardware (general purpose NoC + HW/SW interfaces)

Programming Model

- cost & performances issues !!!!
 - Non optimized software increases embedded memory size
 - Non optimized HW/SW interface and NoC increase global latencies
- Why not a dedicated NoC for a dedicated programming model ?????

Dedicated Programming Model

Example 1: GPU

- Example of dedicated hardware for a dedicated programming model:
 - Programming model for graphics data parallelism:
 - CUDA (Nvidia), OpenCL:
 - tex2D() for 2D cache acces
 - Coalesced memory acces by SIMD blocks of threads
 - Dedicated Hardware:
 - 2D/3D Read-Only caches
 - Scheduling of contiguous threads -> DMA
 - Crossbars
 - Local shared Memory barriers

Dataflow Programming Model (message passing)

- Dataflow model:
 - Explicit parallelism
 - Decoupling of scheduling and communication
- Properties:
 - Parallelism Scalability
 - Modularity (reuse)
 - Portability (stream asynchrony : GALS!)
 - Adaptivity (preemption, migration...)

Dataflow Programming Model

- Examples of front-end:
 - Simulink, SystemC, CAL-openDF, ...
- features:
 - Scheduling policies
 - Communication mechanisms (blocking or non blocking)
 - synchronization
 - Shared data structures accesses (RO, RW, hierarchy,...)

Dataflow Programming Model

- Dataflow Applications use mainly double-buffering and multi-buffering to benefit from separation and pipelining between communication and treatment
- SystemC example:
 - Non blocking access with double-buffering
= `sc_signal`
 - Blocking access with multi-buffering
= `sc_fifo` (from a single thread)

Dedicated Programming Model

Example 2: Crossbus

- Subset of SystemC frontend
 - Communication: `sc_signal`, `sc_fifo`
 - Synchronization: `wait()`, `sc_event`
 - Global Data structures acces: `tex2D` like library
- NoC Hardware features:
 - Irregular X-Y GALS mesh with HW deviation tables
 - Dedicated caches (2D/3D/tree/octree...)
 - Dedicated synchronization network
 - Dedicated DMA and packets with sync information, data structure ...

SystemC Programming Model

```
class producer1 : public sc_module {
public:
    sc_out <int> a1;
    sc_in <bool> clk;

    SC_HAS_PROCESS(producer1);

    producer1(sc_module_name name) : sc_module(name) {
        SC_THREAD(main22);

        sensitive_pos << clk;    }

    void main22() {
        int  nb1;

        nb1=1;

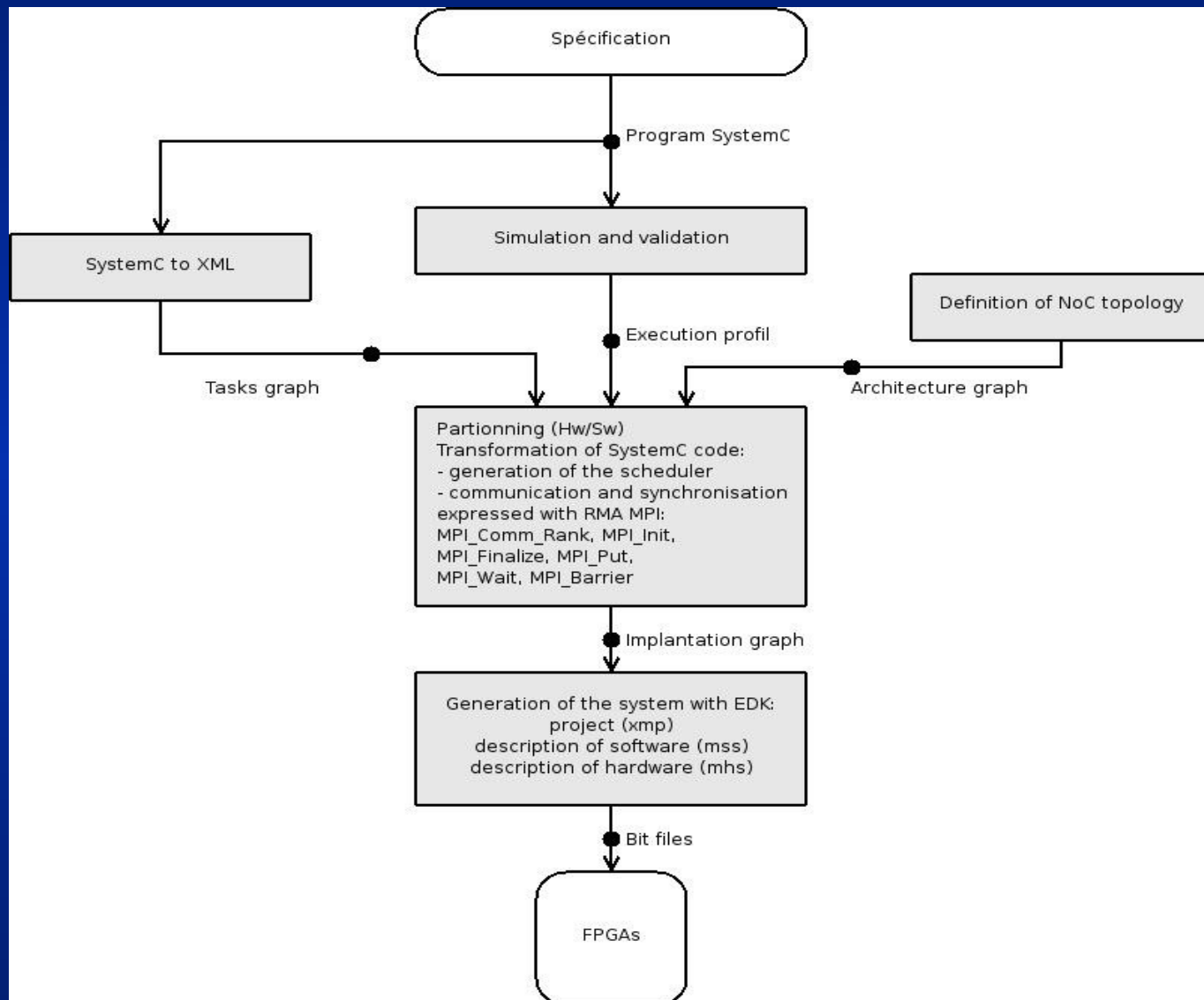
        while(1) {
            if ((nb1%2)==0) {
                a1.write(nb1++);

                wait();
            }
            else {
                a1.write(nb1);

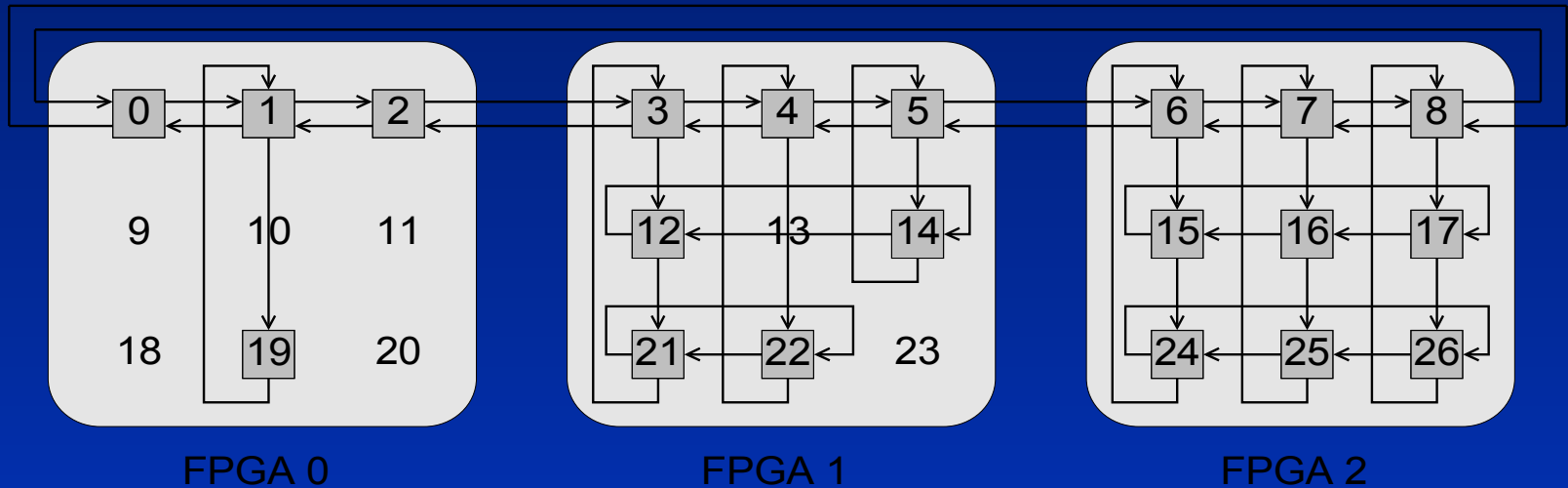
                nb1=nb1+3;

                wait();
            }
        }
    }
};
```

Design Flow

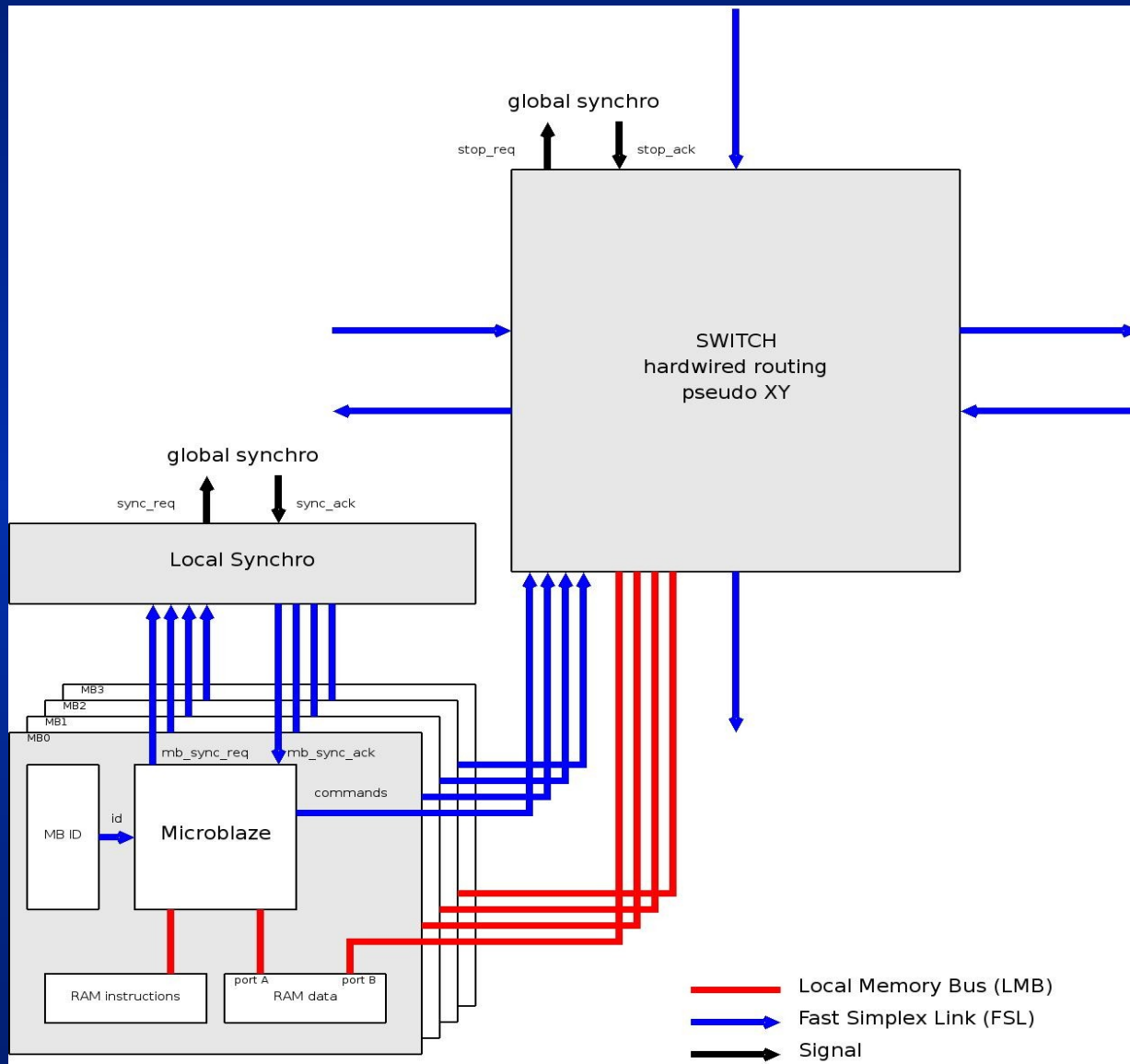


NoC Topology (multi-chip, 3D)



- **Hardwired MPI RMA Subset**
 - **MPI_Comm_Rank:** Read of Id
 - **MPI_put:** Write of a DMA command
 - **MPI_Barrier:** Blocking Write of a sync request

NoC Synchro



Crossbus Area and Timing Performances

- 32bit Crossbus area: almost same size as simple Hermes NoC (8bits)

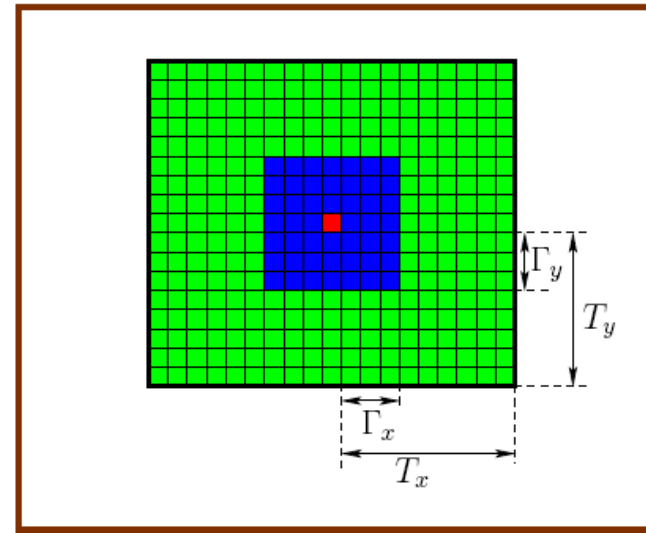
	IOs	GLOBAL BUFFERS	FUNCTION GENERATORS	CLB SLICES	DFFS or LATCHES	FREQUENCY (MHz)
ROUTER avec 1 MB	497	0	2987	1494	628	173.3
ROUTER avec 2 MB	582	0	3277	1639	609	178.6
ROUTER avec 4 MB	743	0	3987	1994	668	203.3

- Timing performances of a producer/consumer case study according to the size of the data paquets :

	4 mots	8 mots	16 mots
Initialisation du système	1585	3281	4057
MPI_init()	426	426	426
MPI_Consom_Rank()	1	1	1
Ordonnement	60	60	60
Ecriture signal	86	133	178
MPI_Barrier()	1	1	1

2) Data accesses (power issue) (dedicated nD-AP cache)

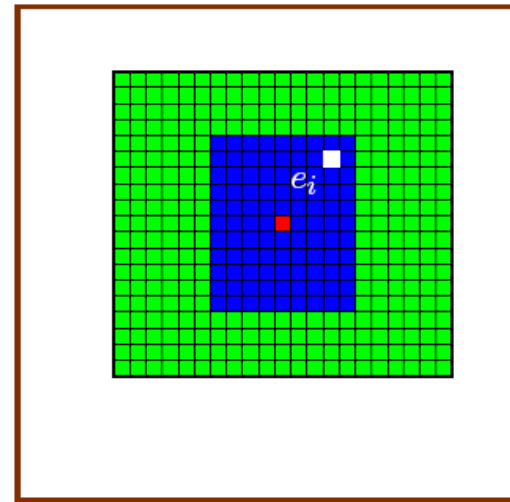
- 2D block elements
- Measures the mean of the addresses
- 'c' is the center of the cached area
- The cache is updated each time the mean value drifts too much from the cache center



The cache parameters are: T_x , Γ_x , Δ_x , T_y , Γ_y and Δ_y

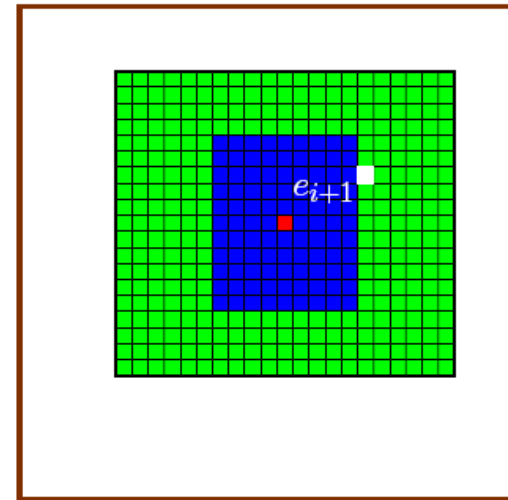
2D-AP cache behaviour

- e_i : The estimated center of the cache at iteration i
- Δ_x : The cache step displacement in 'X' direction



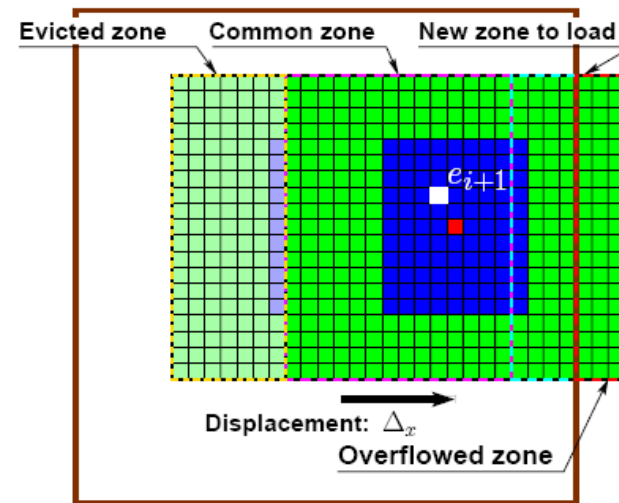
2D-AP cache behaviour

- e_i : The estimated center of the cache at iteration i
- Δ_x : The cache step displacement in 'X' direction



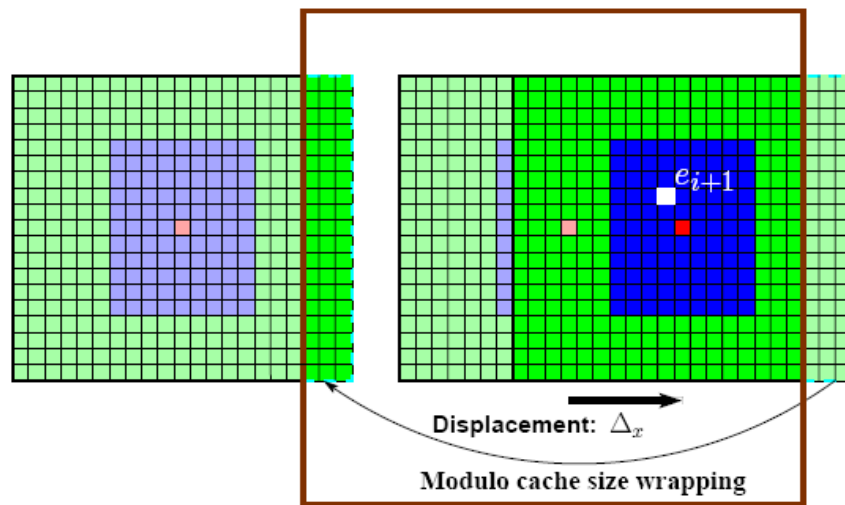
2D-AP cache behaviour

- e_i : The estimated center of the cache at iteration i
- Δ_x : The cache step displacement in 'X' direction



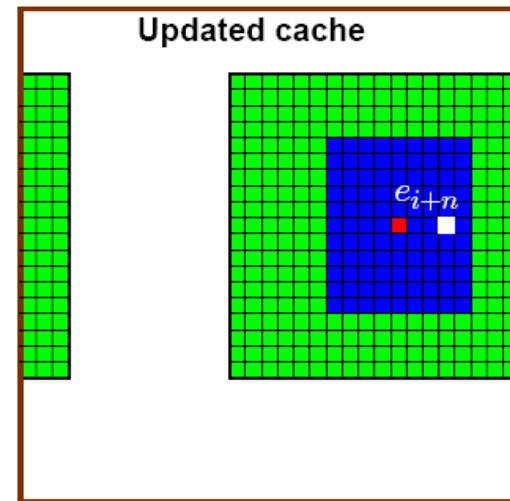
2D-AP cache behaviour

- e_i : The estimated center of the cache at iteration i
- Δ_x : The cache step displacement in 'X' direction

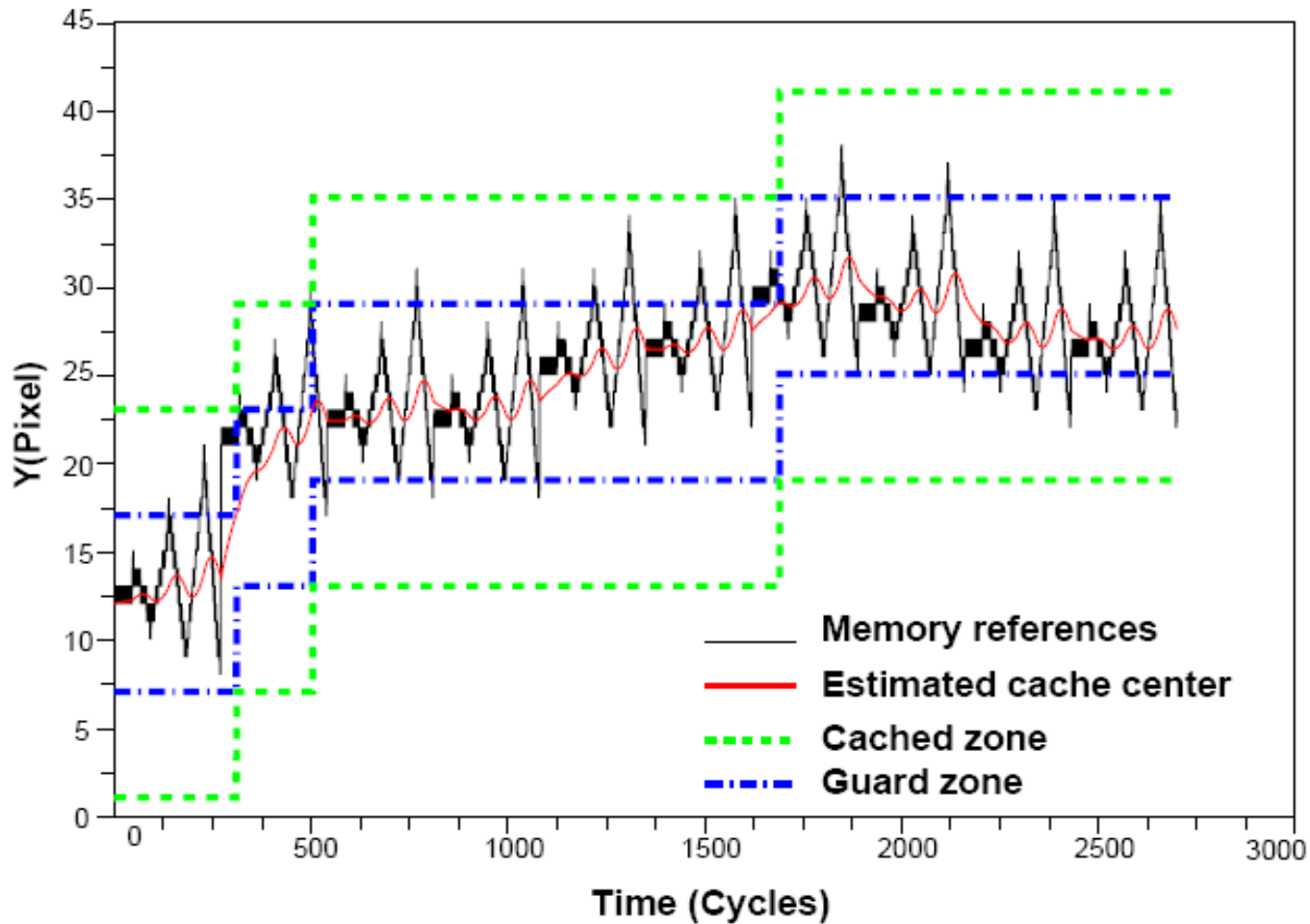


2D-AP cache behaviour

- e_i : The estimated center of the cache at iteration i
- Δ_x : The cache step displacement in 'X' direction



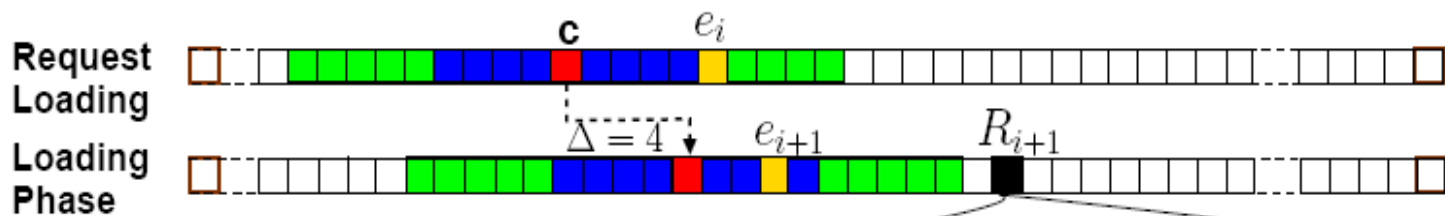
Cache in action



Memory accesses against time.

Computation of cache parameters

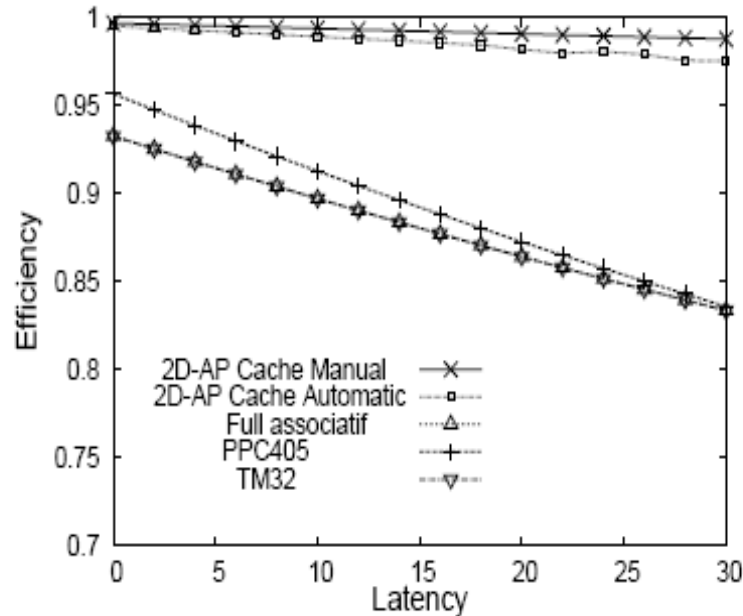
- ① To avoid loading conflicts: $\Delta > v(Lat + \frac{\Delta}{m})$
- ② To avoid cache oscillations: $\Gamma > \frac{\Delta}{2}$
- ③ To avoid waiting time: $T > \Gamma + v(Lat + \frac{\Delta}{m}) + A$



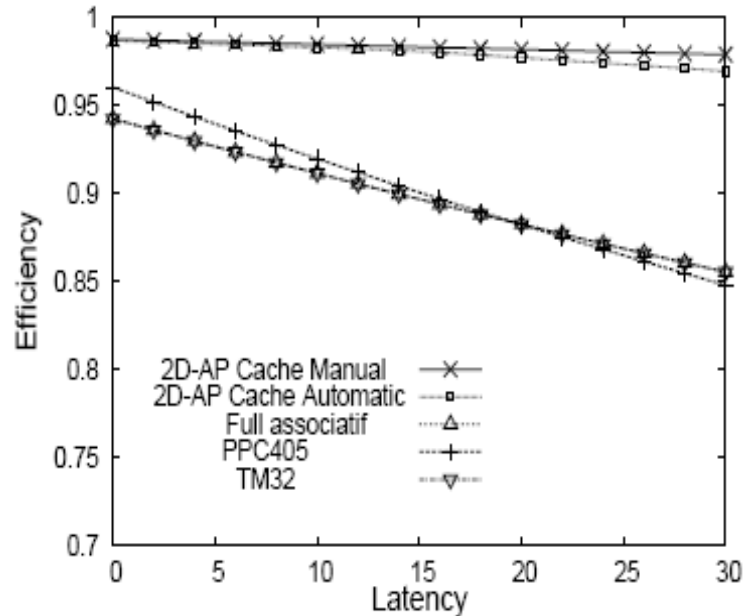
- Cache center "c"
- Guard zone 2Γ
- Cached zone $2T$
- Cache maximum size

Bad case
The referenced address is out of the cached zone.
This causes a cache miss.

Cache Results



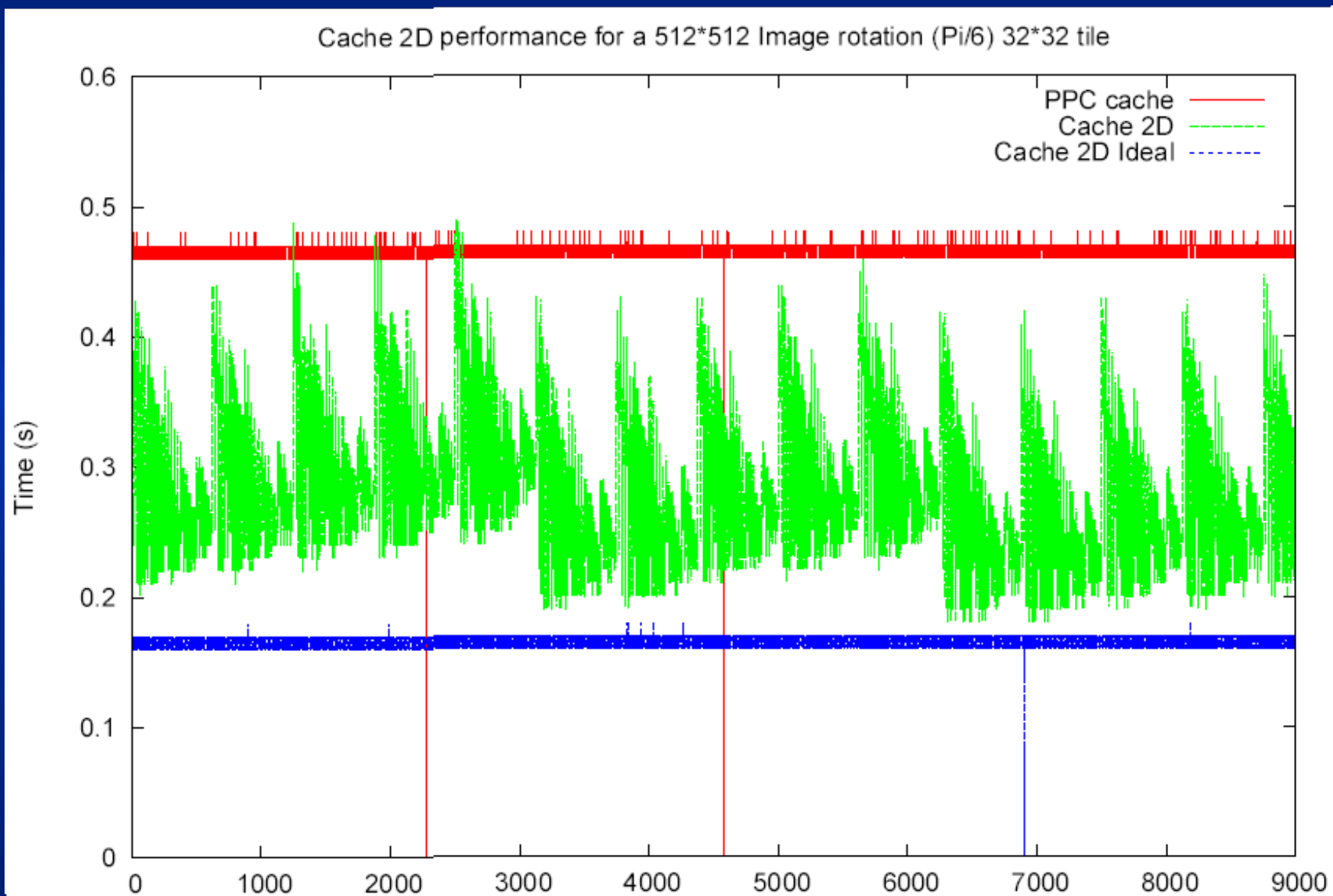
(c) 2D Backprojection: cache size=0.5 KB



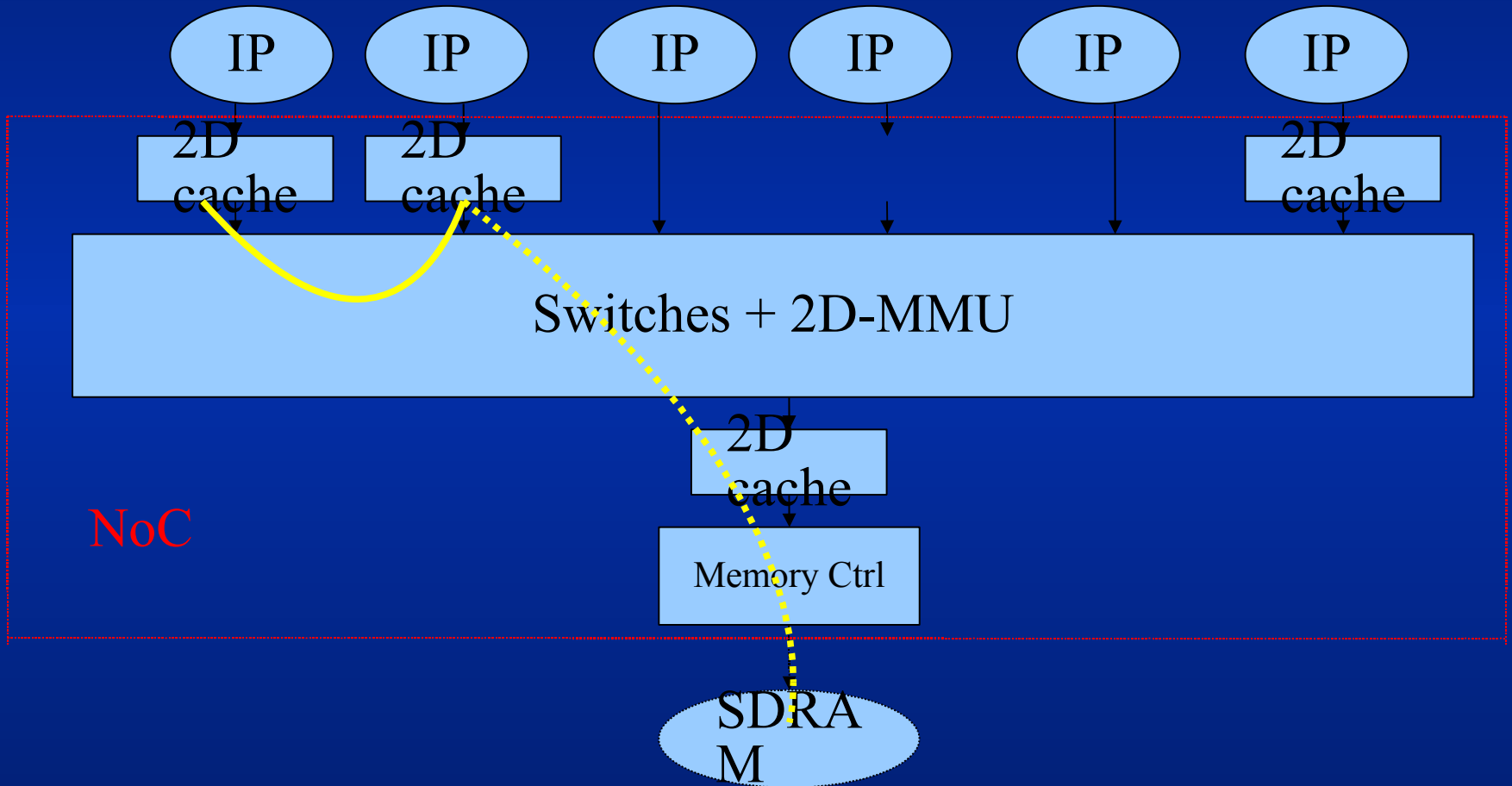
(d) Ray Casting : cache size=1 KB

- 2D Backprojection: $Lat_{max} \geq 30$, ideal prefetch and high data reuse. Memory gain: 32.
- 3D Ray Casting : $Lat_{max} \geq 30$, ideal prefetch and high data reuse. Memory gain: 16.

2D Cache parameters exploration



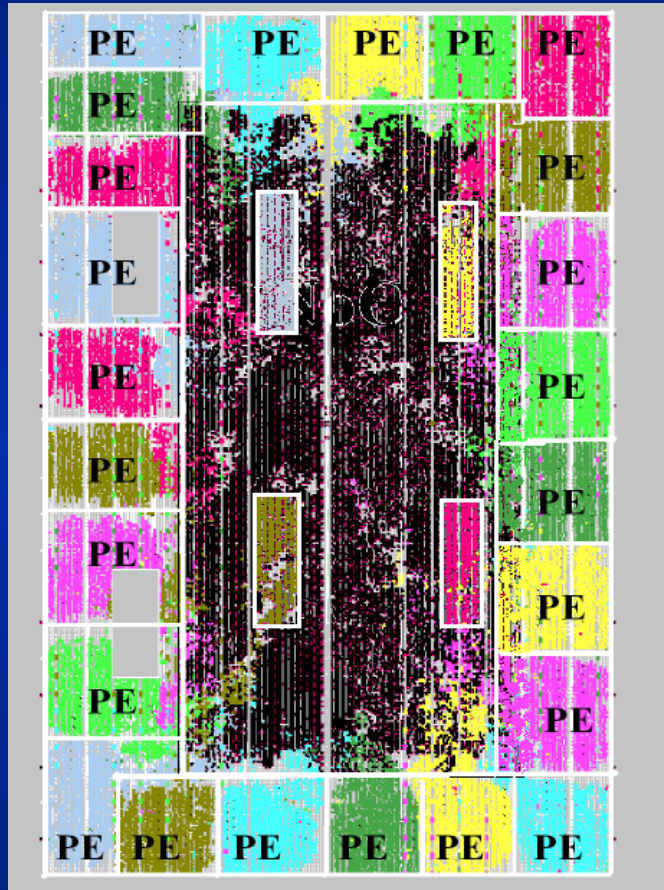
NoC with hierarchical distributed dedicated caches (2D)



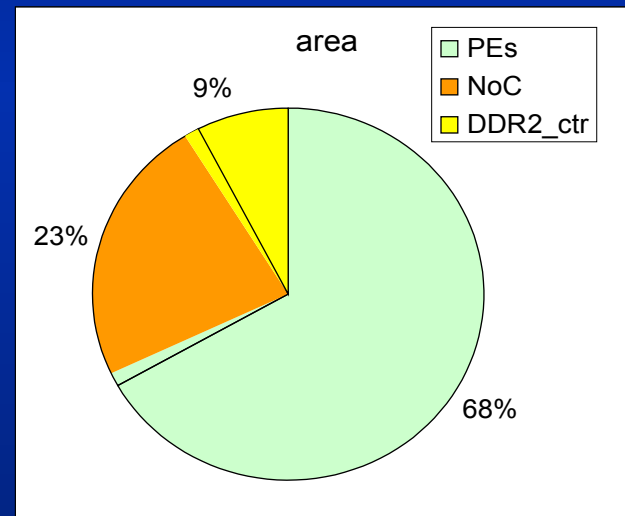
NoC Exploration

- High level simulation (SoCLiB project)
 - Need at some point of low level information
 - --> meet the « simulation wall »
 - Parallel execution on multicore processors (GPU, Cell...): pb of global com!
 - --> easier with next generation
- Emulation on multi-FPGA
 - Brings low level accurate results
 - Transposable to ASIC!

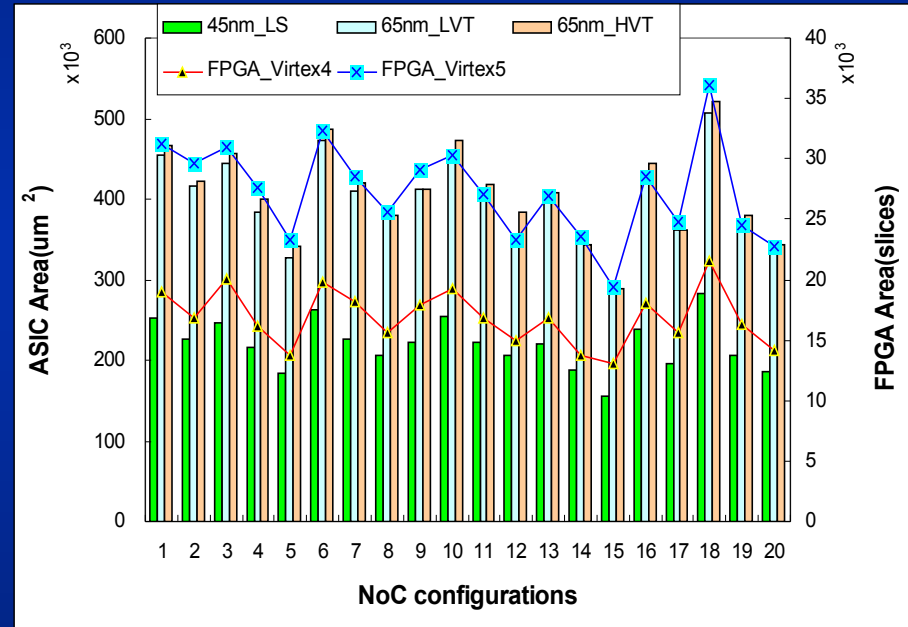
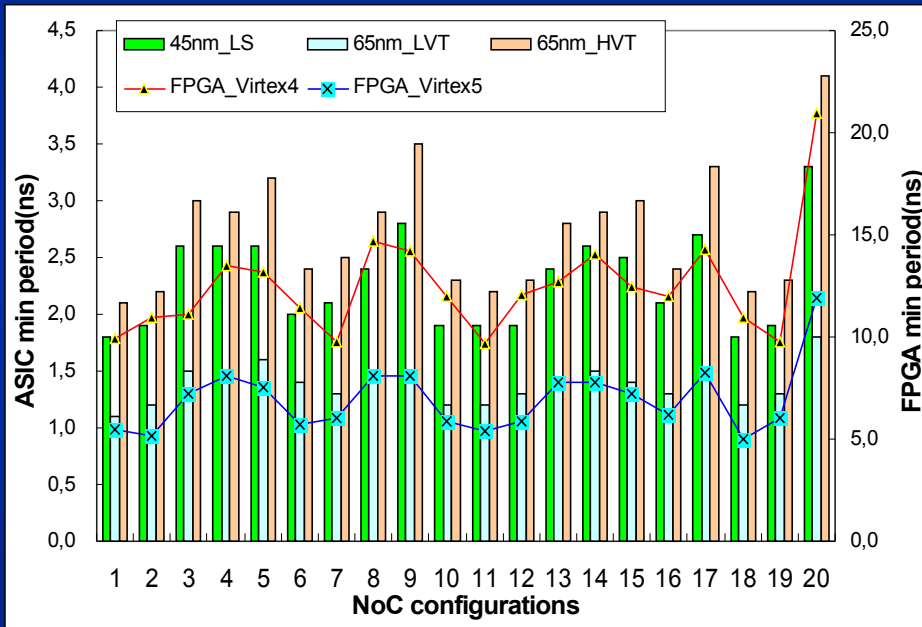
V4FX140 FPGA 24-PE NoC (Arteris)



Resource	Number	Percentages
RAMB16s	384 / 552	69%
DSP48s	72 / 192	37%
4 input LUTs:	94883 / 126336	75%
Slices	55266 / 63168	87%



FPGA/ASIC Comparison



Conclusion

- SystemC: Programming Model for Dataflow application
- Communication Paradigm (NoC) optimized for the Programming Model (SystemC):
 - Low cost efficient NoC enabling FPGA targeting
 - Dedicated parametrizable caches
 - Automated NoC synthesis and exploration

Thank You !
Questions ?