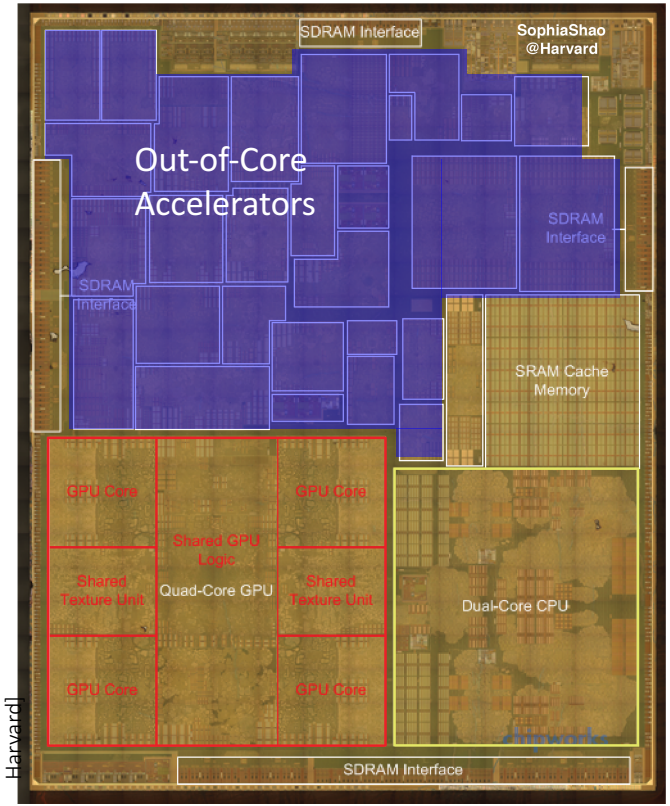
Horizon 2020
European Union funding
for Research & Innovation

From high-level specification down to hardware

Christian Pilato (Università della Svizzera Italiana)
and
Francesca Palumbo (Università degli Studi di Sassari)

Heterogeneous Systems-on-Chip



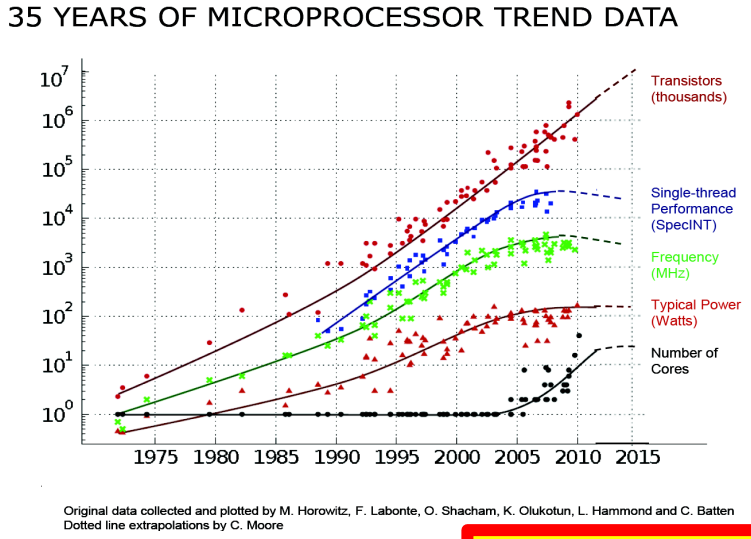
[Die photo from Chipworks - Accelerators annotated by Y.S. Shao @ Harvard]

iPhone 6 features the A8 SoC

- 8.47 x 10.50 mm (20nm by TSMC)
  - 13% smaller than A7 (28nm)
- dual-core ARM CPU at 1.40 GHz
  - 25% more CPU performance
- four-cluster PowerVR GPU
  - 50% more graphics performance
- 2 billions of transistors
  - twice the number of transistors compared to the A7
- **almost 30 out-of-core accelerators**
  - **50% of the power compared to A7 (~20 out-of-core accelerators)**

Dark Silicon is the Next Big Issue

"MOORE'S LAW: The number of transistors on an affordable CPU would double every two years" (G.E. Moore, 1976)



20+ years of CPU improvements (pipeline stages, branch predictions, multicore, etc.), but we hit the **utilization wall**!

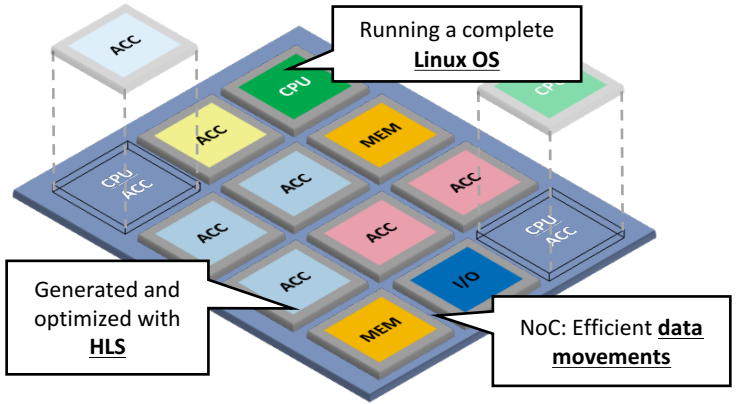
"With each successive generation, the percentage of a chip that can actively switch drops exponentially due to power constraints"

dark silicon is pushing towards **heterogeneous computing**

ESP: Embedded Scalable Platform

Embedded platform with:

- Leon3 processor
- Possibility to plug-and-play a variety of accelerators
- On-chip communication with DRAM controllers
  - Latency-insensitive NoC



FPGA prototype for full-system evaluation

- 100 MHz
- 1 GB of DDR3 memory with two physically-separated address spaces

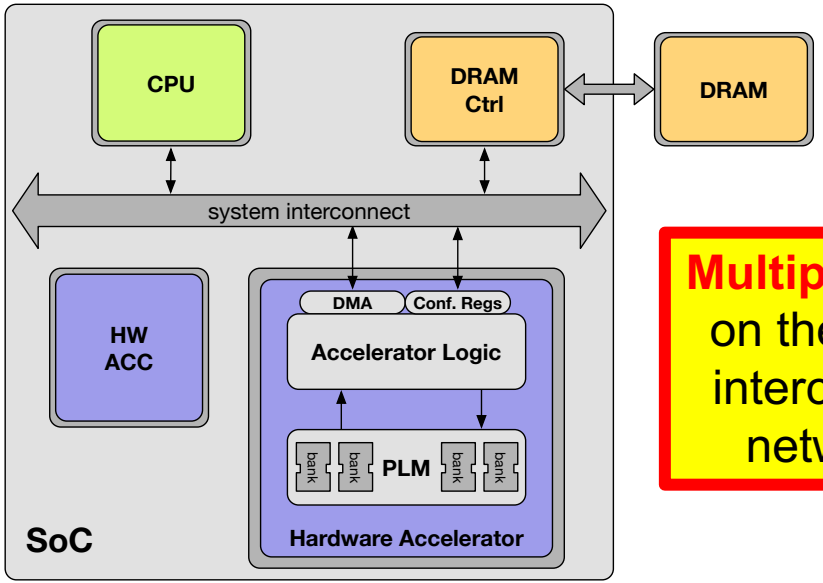
Mantovani *et al.*
ASPDAC 2016

Abstracting Accelerator-based Systems

Specialized microarchitecture for both computation and storage

- For delivering energy-efficient high performance

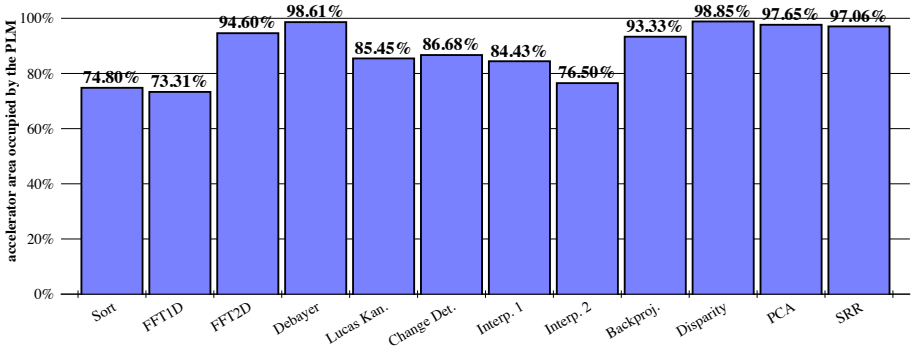
CPU to prepare the data in DRAM and control the accelerator (e.g., device driver)



Multiple accelerators on the same system interconnect (bus or network-on-chip)

PLM and DRAM: A Huge Gap

Bench.	DRAM Data Size (MB)	PLM Data Structures		Bench.	DRAM Data Size (MB)	PLM Data Structures	
		(#)	(MB)			(#)	(MB)
Sort	4.000	6	0.024	FFT1D	0.250	10	0.040
FFT2D	64.000	4	0.128	Debayer	16.000	4	0.096
Lucas Kan.	32.000	11	0.020	Change Det.	320.000	10	0.062
Interp. 1	32.040	6	0.048	Interp. 2	64.010	7	0.640
Backproj.	256.040	8	0.099	Diparity	15.820	11	0.146
PCA	20.190	3	0.117	SRR	4.760	21	0.076



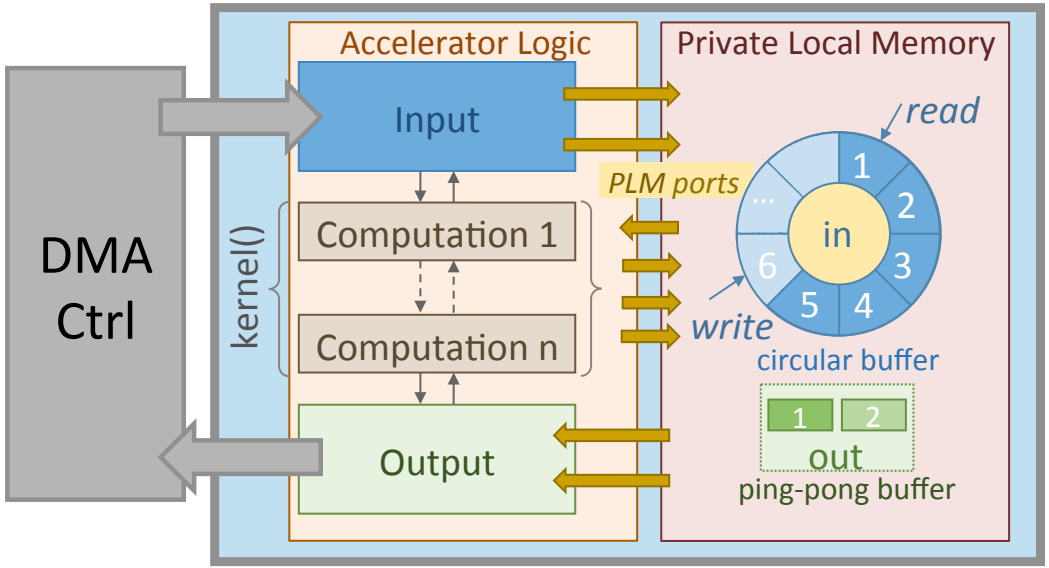
Each accelerator is ~1mm^2
- Comparable to Apple A8 accelerators

PLM occupies **75% to 98%** of accelerator area
- Still a lot of data transfers

Why Out-of-core Accelerators?

Loosely-coupled accelerators are more efficient in case of large data sets to elaborate

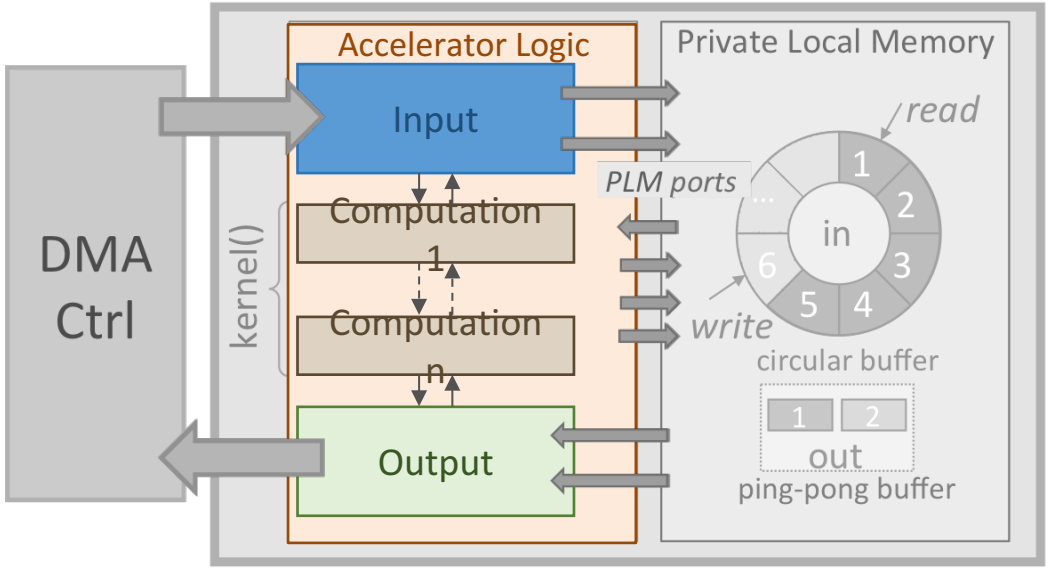
- Part of the data can be locally stored in **Private Local Memories** for fast access



Cota *et al.*
DAC 2015

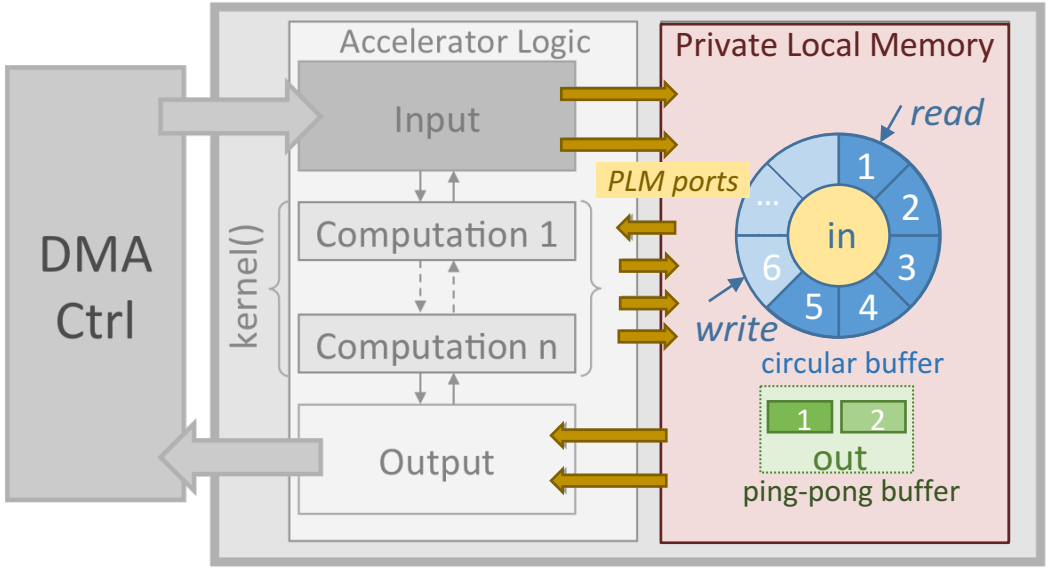
Out-of-core Accelerators

Accelerator logic with specialized micro-architecture to exploit hardware parallelism



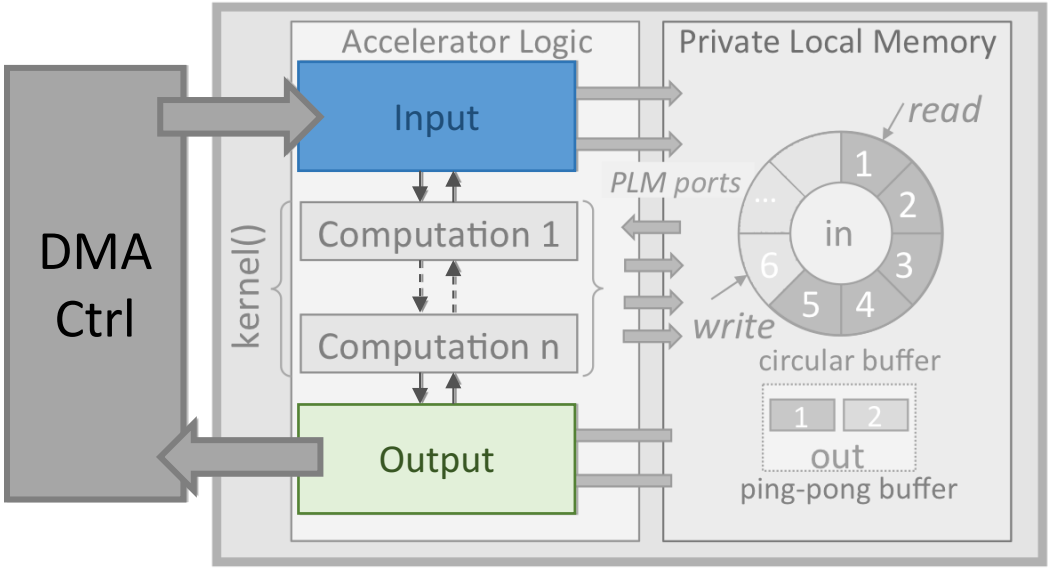
Out-of-core Accelerators

Private Local Memory with multiple ports to offer more hardware parallelism



Out-of-core Accelerators

Communication processes and DMA controller to exchange data with the rest of the system

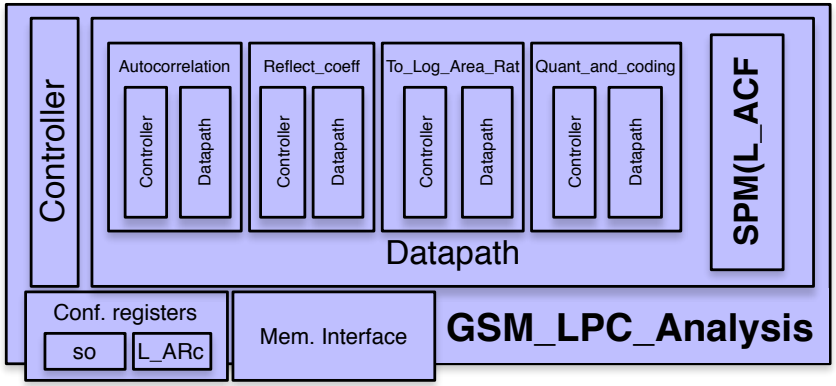


From Programmer's View to RTL

C language allows us to easily specify, design, and optimize accelerators for irregular applications

- pointer-based operations (arithmetic, dynamic resolutions, accesses to external memory, …)

```
void Gsm_LPC_Analysis(word* so, word* LARc)
{
   longword L_ACF[9];
   Autocorrelation(so, L_ACF);
   Reflect_coeff(L_ACF, LARc);
   To_Log_Area_Rat(LARc);
   Quant_and_coding(LARc);
}
```

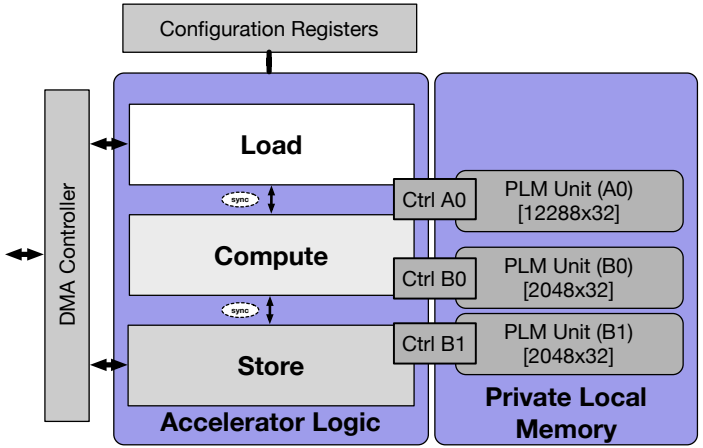


From Programmer's View to RTL

SystemC language allows us to easily specify, design, and optimize data-intensive accelerators

- DMA transfers to exchange data blocks with main memory

```
SC_MODULE(debayer) {
  sc_in<bool> clk, rst;
private:
  int A0[6][2048];
  int B0[2048], B1[2048];
public:
  SC_CTOR(debayer) {
    SC_CTHREAD(Load, clk.pos());
    reset_signal_is(rst, false);
    SC_CTHREAD(Compute, clk.pos());
    reset_signal_is(rst, false);
    SC_CTHREAD(Store, clk.pos());
    reset_signal_is(rst, false);
//...
```



Good News and Bad News

The Good News:

- **Abundant data parallelism enables high performance**
  - **Hardware parallelism**
- **Specialized micro-architecture enables energy efficiency**
  - **Possibility to turn off the component when inactive**
- **High-Level Synthesis (HLS) tools come in handy**

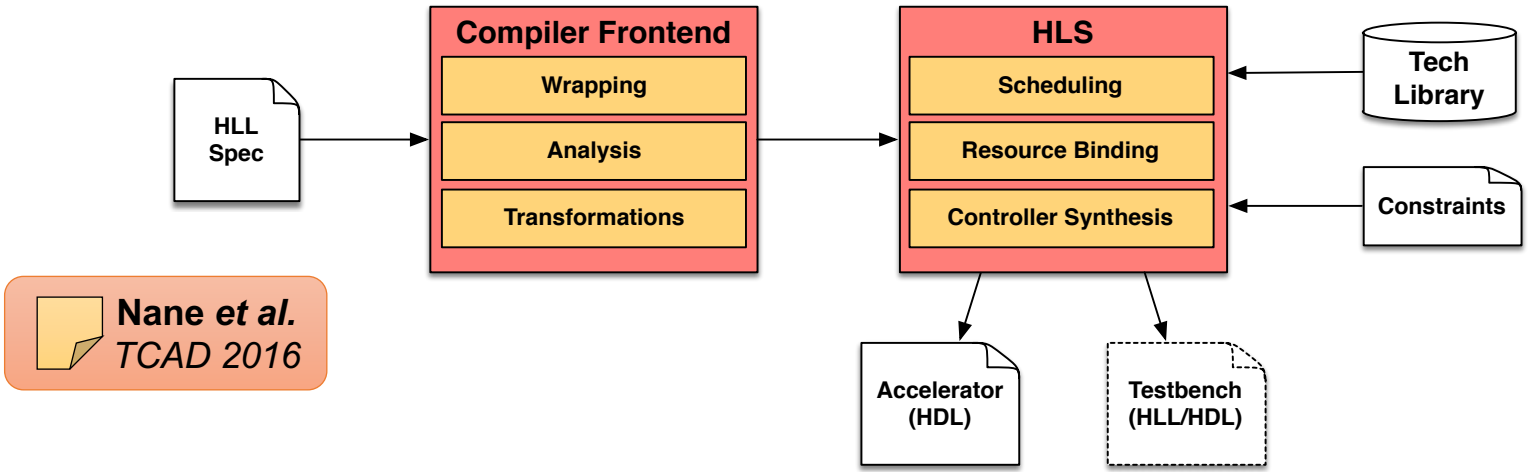
The Bad News:

- **Lack of unified design and programming models**
  - **Limited reusability**
- **Communication overheads**
  - **Especially in case of many accelerators running together and competing for the memory**
- **Hardware parallelism requires a specialized local memory**
  - **Many data to process in the same clock cycle**

HLS-based Design Flow

High-Level Synthesis (HLS) tools are pretty good to design the accelerator logic

- Description in a high-level language (e.g., C/C++/SystemC)
- Several compiler-based, technology-aware optimizations



Generating Optimized Heterogeneous SoCs

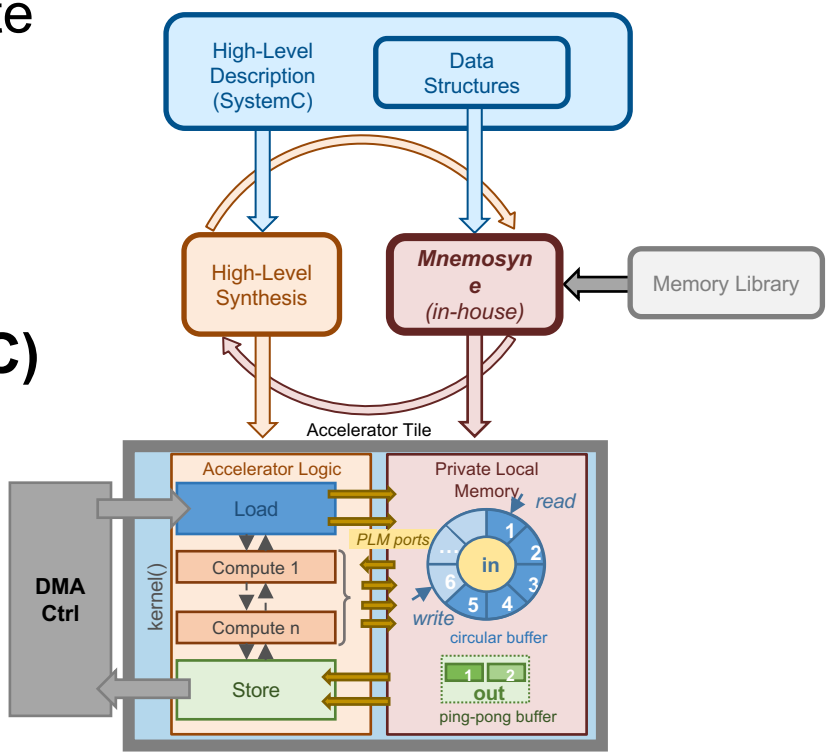
High-Level Synthesis (HLS) to create the accelerator logic

- Definition of memory-related parameters (e.g. number of process interfaces)

System-level sharing of resources with **Multi-Dataflow Composer (MDC)**

Generation of **specialized PLMs**

- Technology-related optimizations
- Possibility of system-level optimizations across accelerators



Target & Technological Challenges

- **DATAFLOW MODEL OF COMPUTATION**
  - Modularity and parallelism → *EASIER INTEGRATION AND FAVOURED RE-USABILITY*
- **COARSE-GRAINED RECONFIGURABILITY**
  - Flexibility and resource sharing → *MULTI-APPLICATION PORTABLE DEVICES*

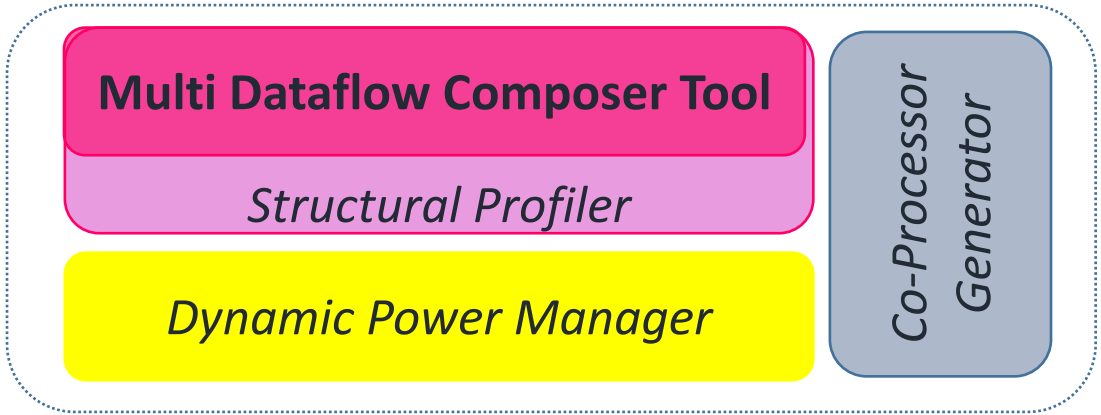
Target & Technological Challenges

- **DATAFLOW MODEL OF COMPUTATION**
  - Modularity and parallelism → *EASIER INTEGRATION AND FAVOURED RE-USABILITY*
- **COARSE-GRAINED RECONFIGURABILITY**
  - Flexibility and resource sharing → *MULTI-APPLICATION PORTABLE DEVICES*

Reconfigurable Platform Composer Tool Project

Automated **DESIGN FLOW** are fundamental to guarantee **SHORTER TIME-TO-MARKET**. Dealing with **APPLICATION SPECIFIC MULTI-CONTEXT** systems, in particular for **KERNEL ACCELERATORS**, state of the art still lacks in providing a broadly accepted solution.

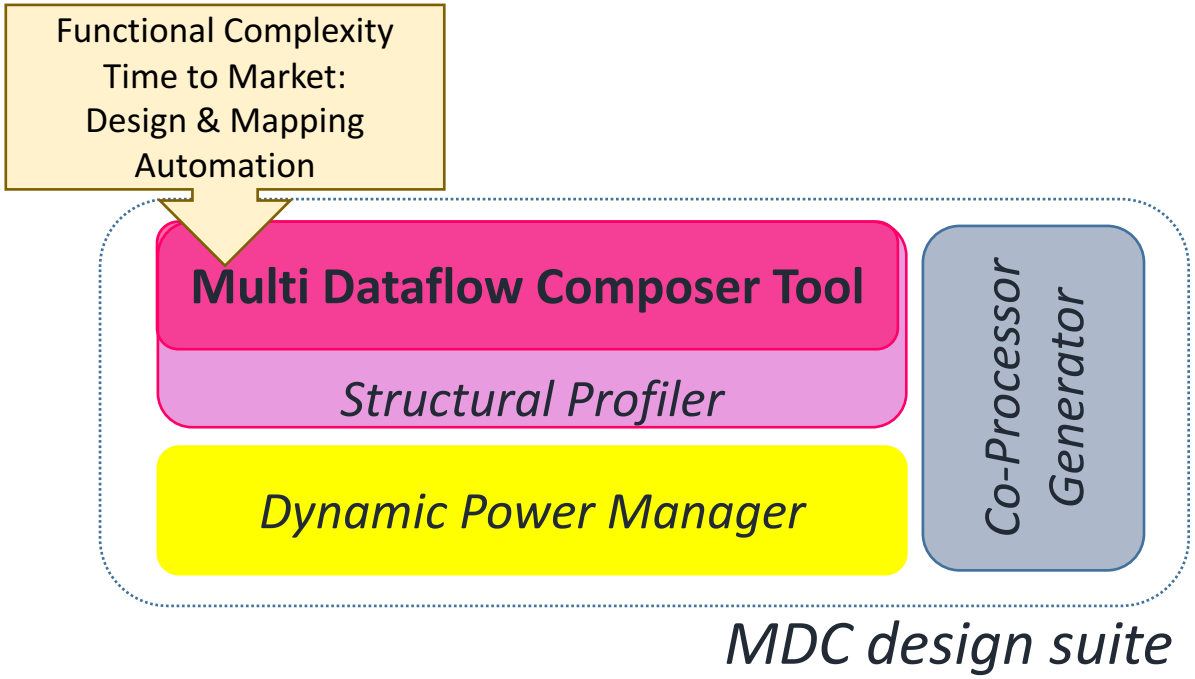
Design Suite & Targeted Challenges



MDC design suite

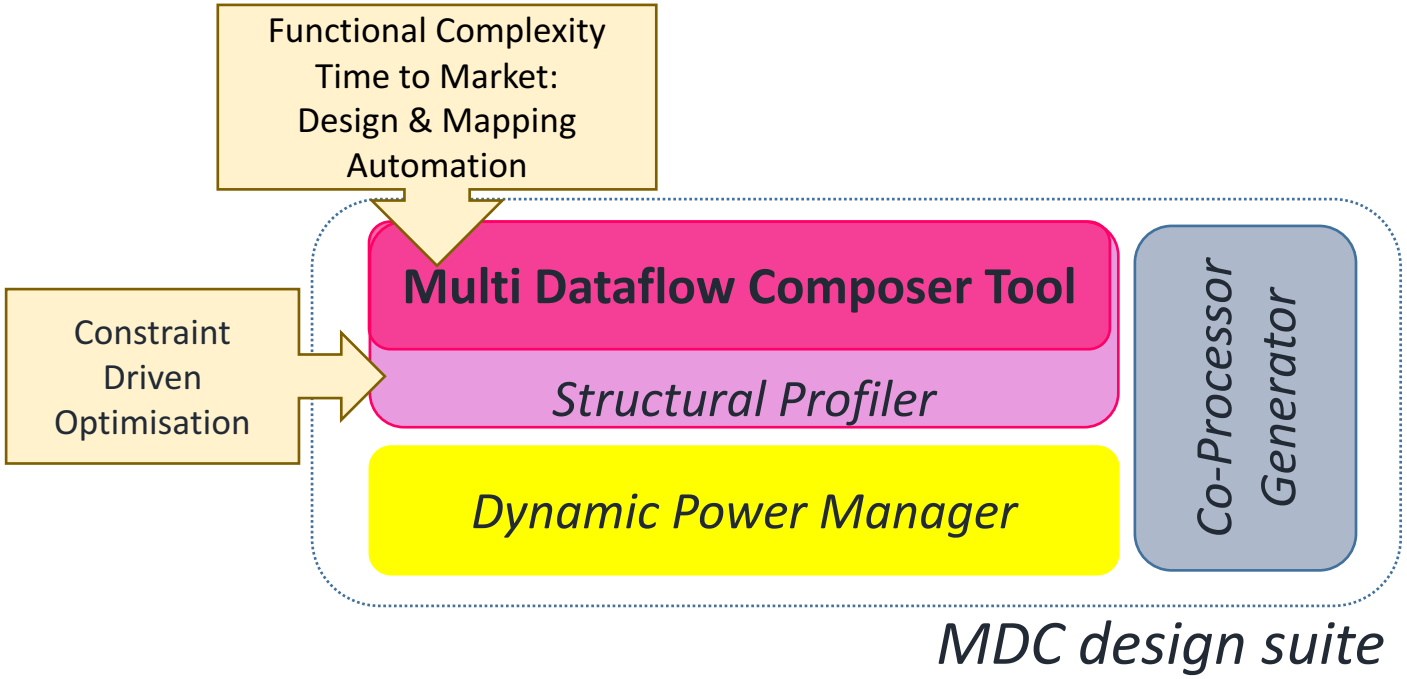
http://sites.unica.it/rpct/

Design Suite & Targeted Challenges



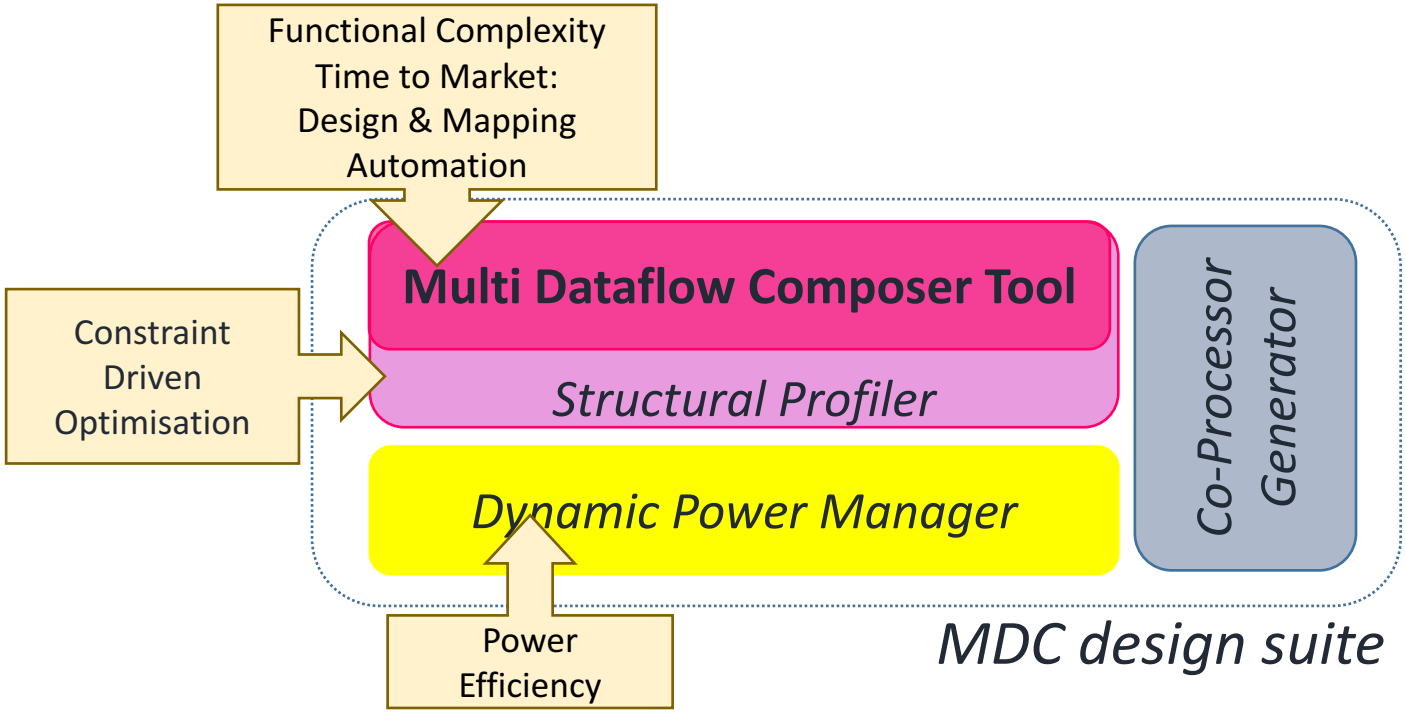
http://sites.unica.it/rpct/

Design Suite & Targeted Challenges



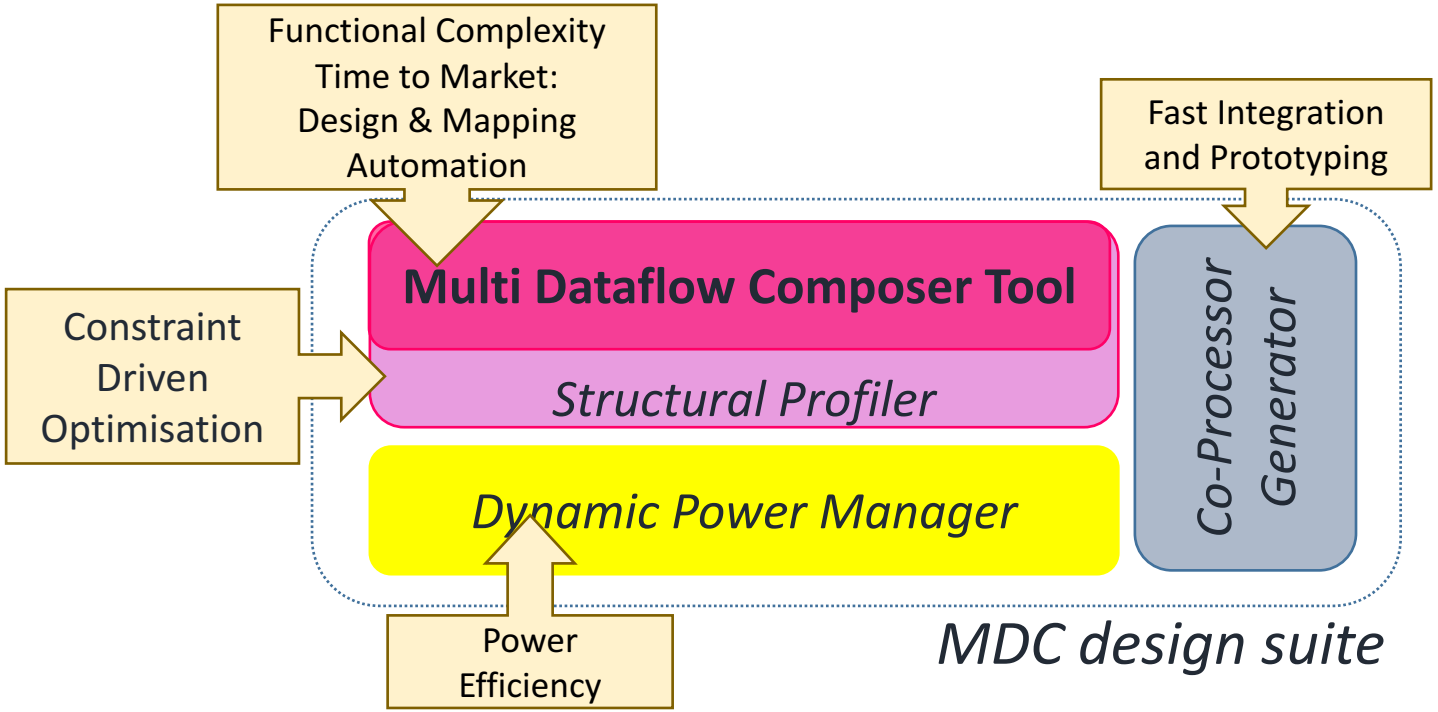
http://sites.unica.it/rpct/

Design Suite & Targeted Challenges



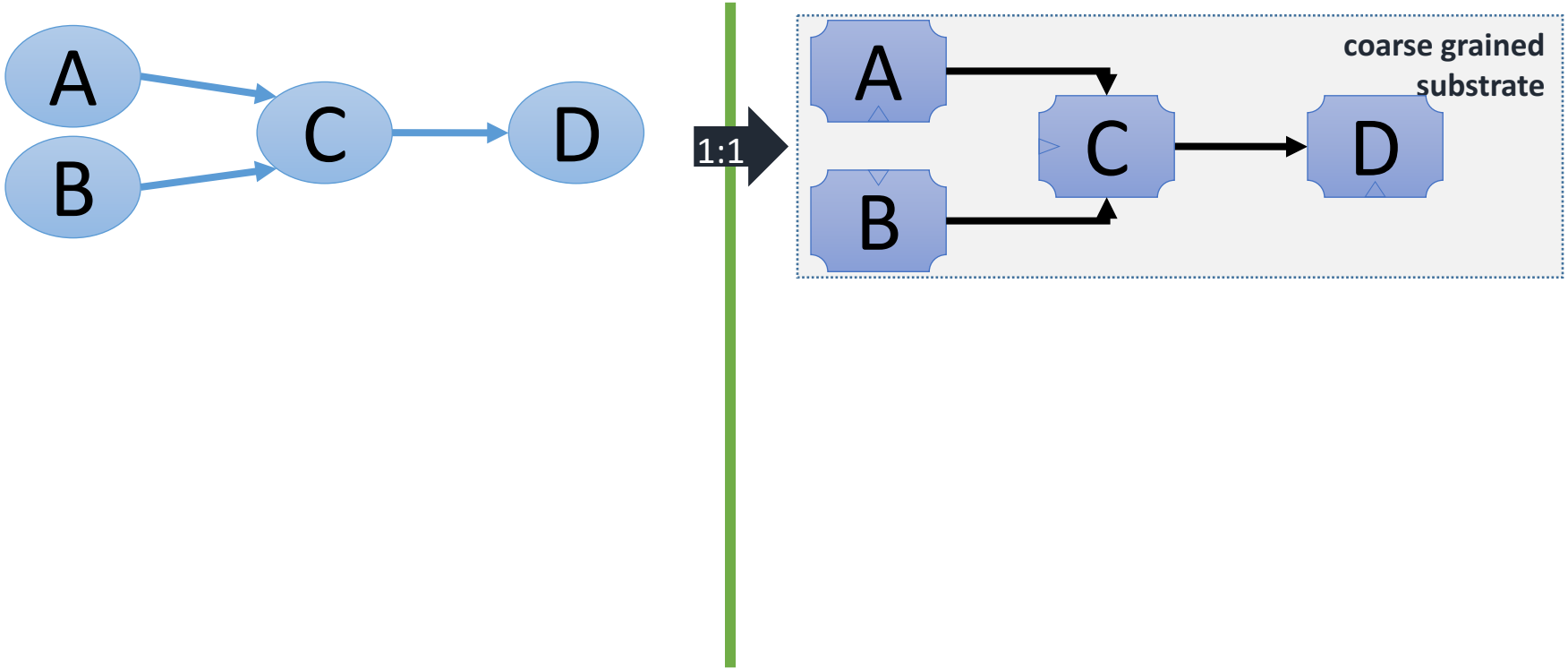
http://sites.unica.it/rpct/

Design Suite & Targeted Challenges

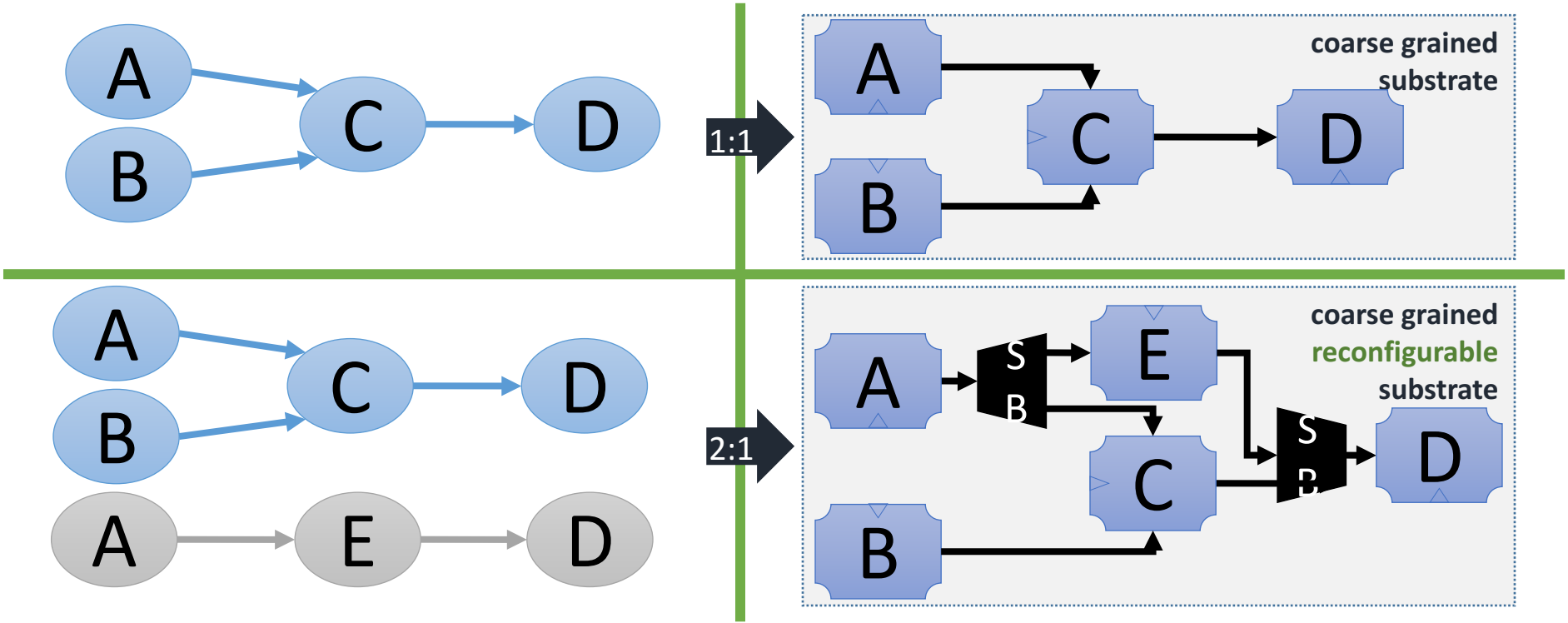


http://sites.unica.it/rpct/

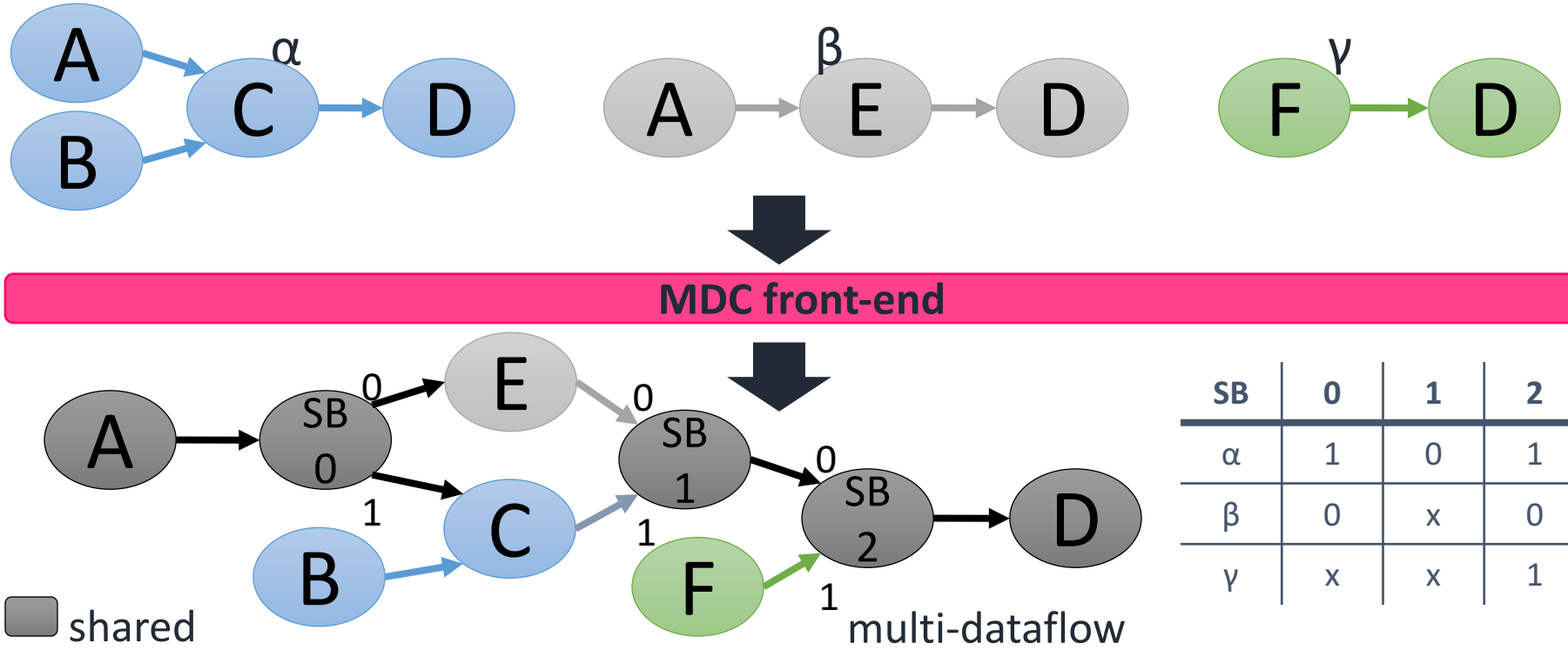
Baseline: From Dataflow to Hardware



Baseline: From Dataflow to Hardware

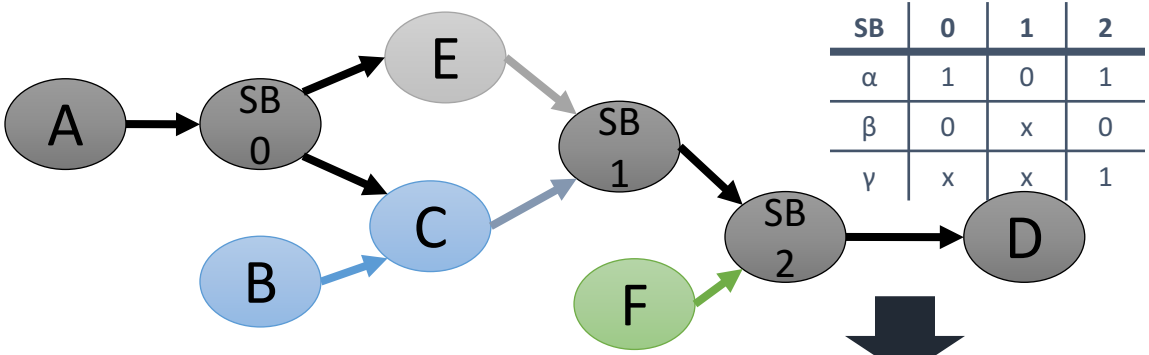


MDC Frontend: Datapath Merging

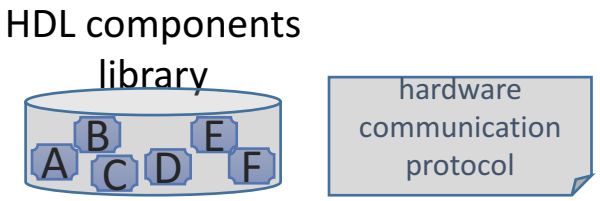


SB	0	1	2
α	1	0	1
β	0	x	0
γ	x	x	1

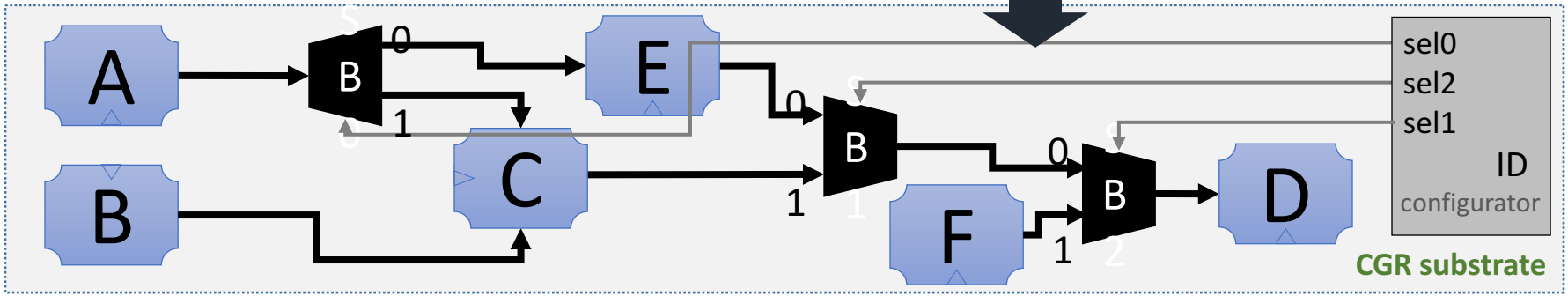
MDC Backend: HW System Implementation



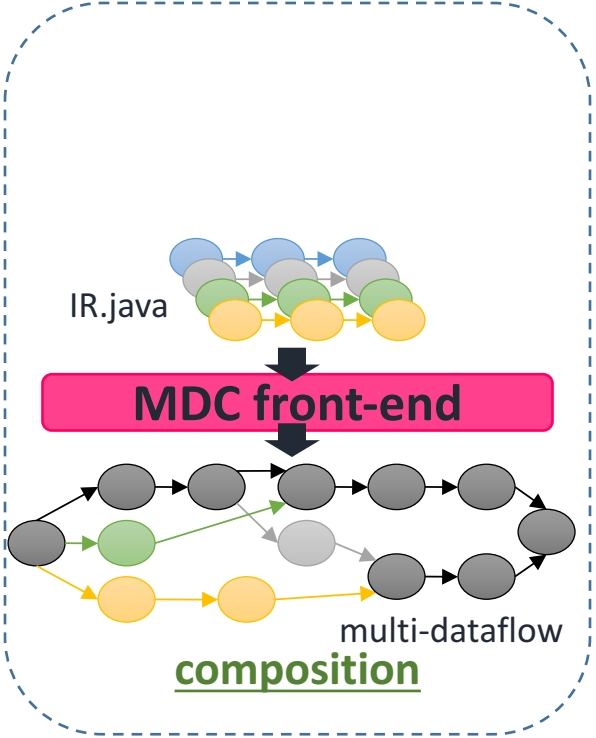
SB	0	1	2
α	1	0	1
β	0	x	0
γ	x	x	1



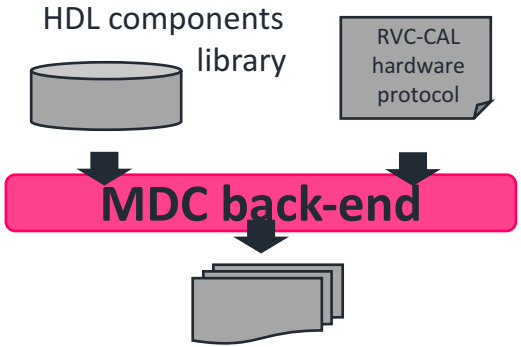
MDC back-end



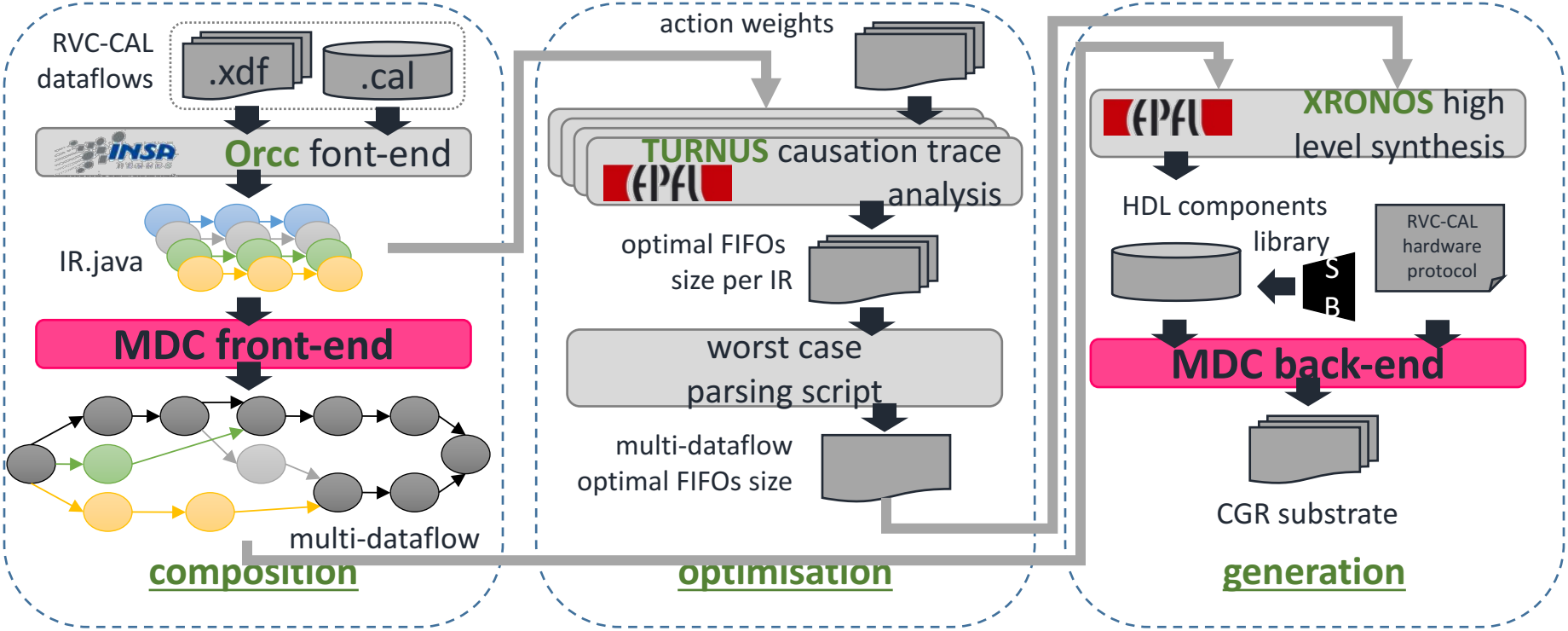
Integration within the MPEG-RVC Framework



optimisation



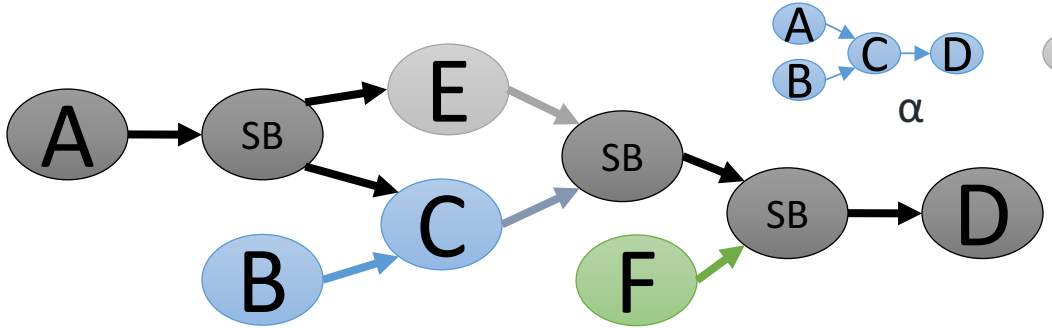
Integration within the MPEG-RVC Framework

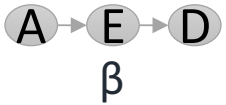


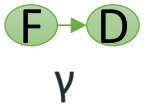
Structural Profiler

What are the topological characteristics impacting on the CGR substrate?

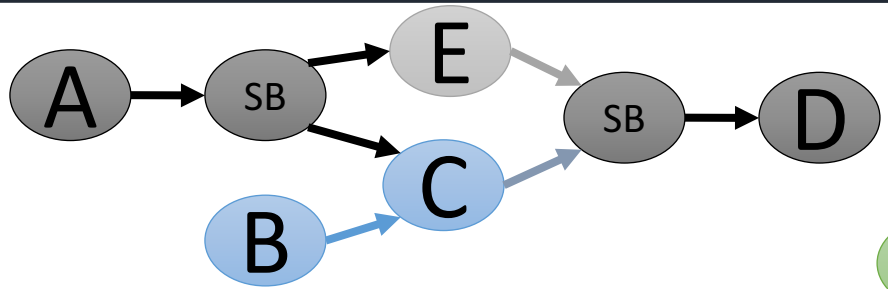
1. Number of merged dataflow specifications

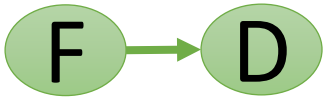






α+β+γ



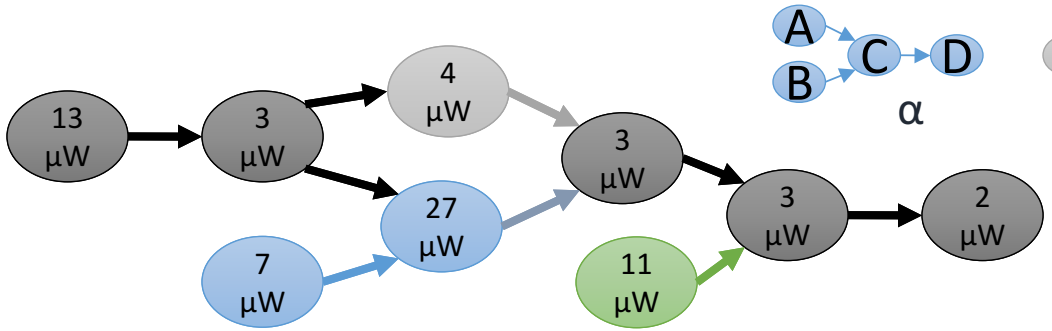


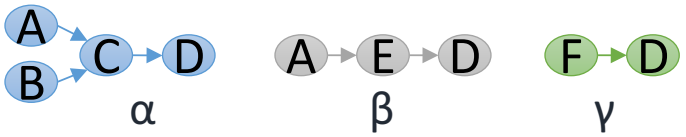
α+β | γ

Structural Profiler

What are the topological characteristics impacting on the CGR substrate?

1. Number of merged dataflow specifications



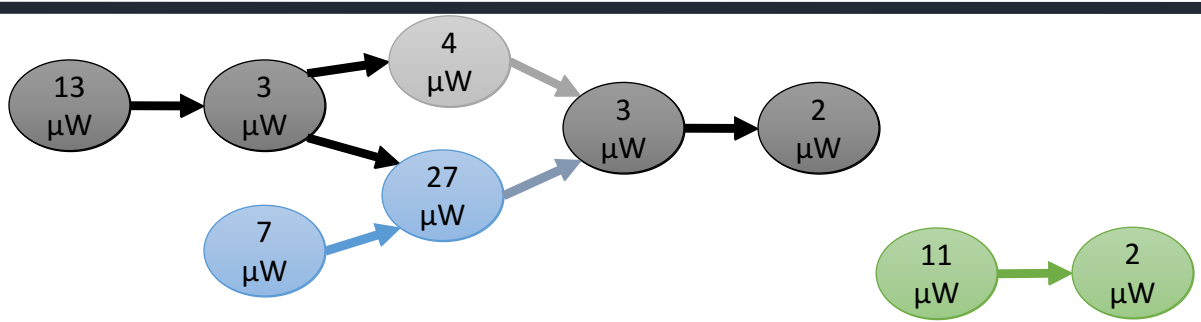


α+β+γ

tot static power

73 µW

☹



α+β|γ

tot static power

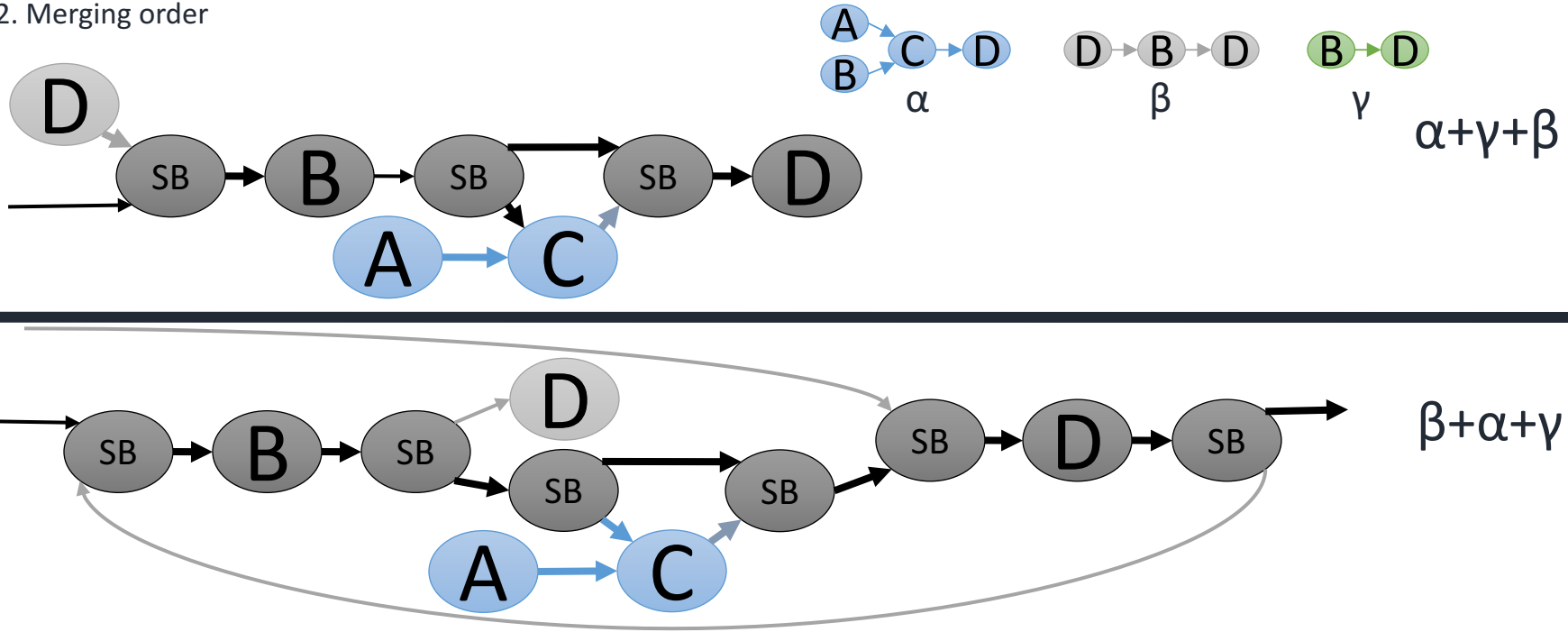
72 µW

☺

Structural Profiler

What are the topological characteristics impacting on the CGR substrate?

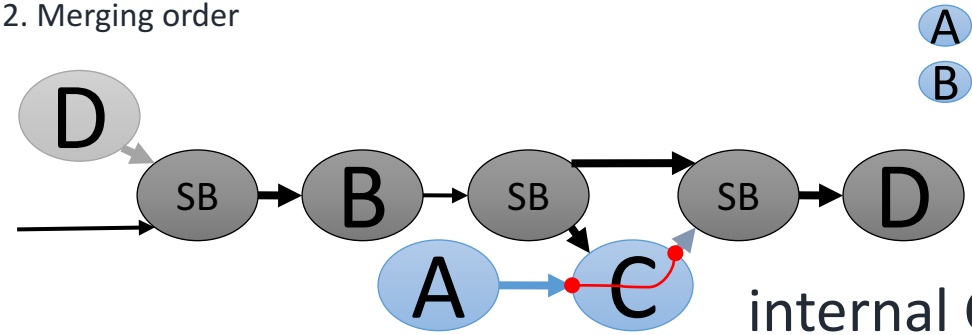
2. Merging order



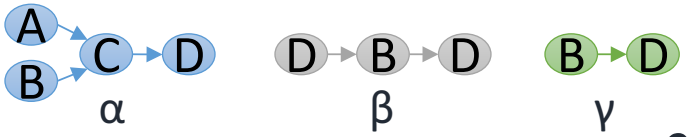
Structural Profiler

What are the topological characteristics impacting on the CGR substrate?

2. Merging order



internal CP



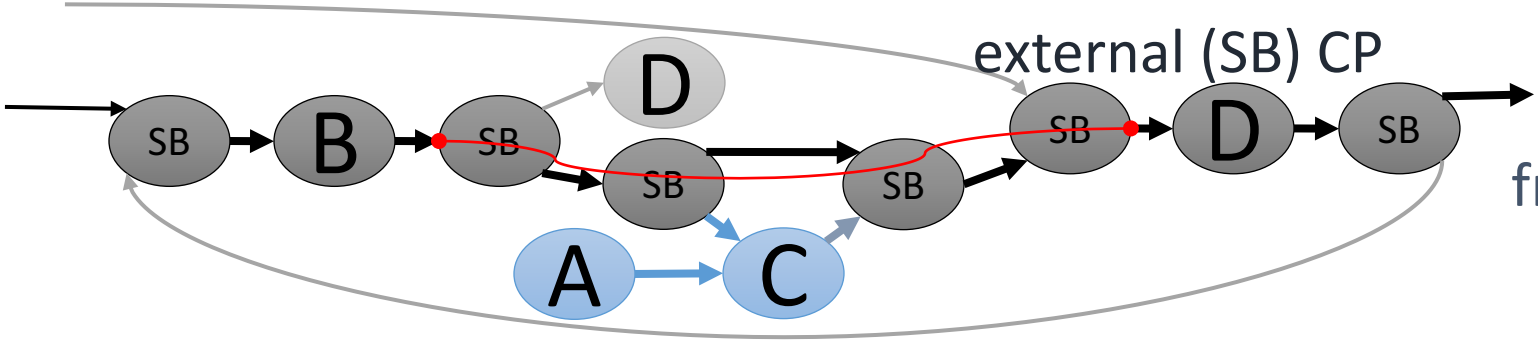
α+γ+β

frequency

45 MHz

☺

external (SB) CP



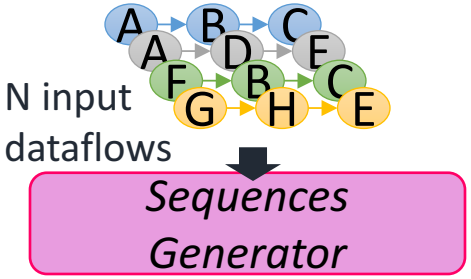
β+α+γ

frequency

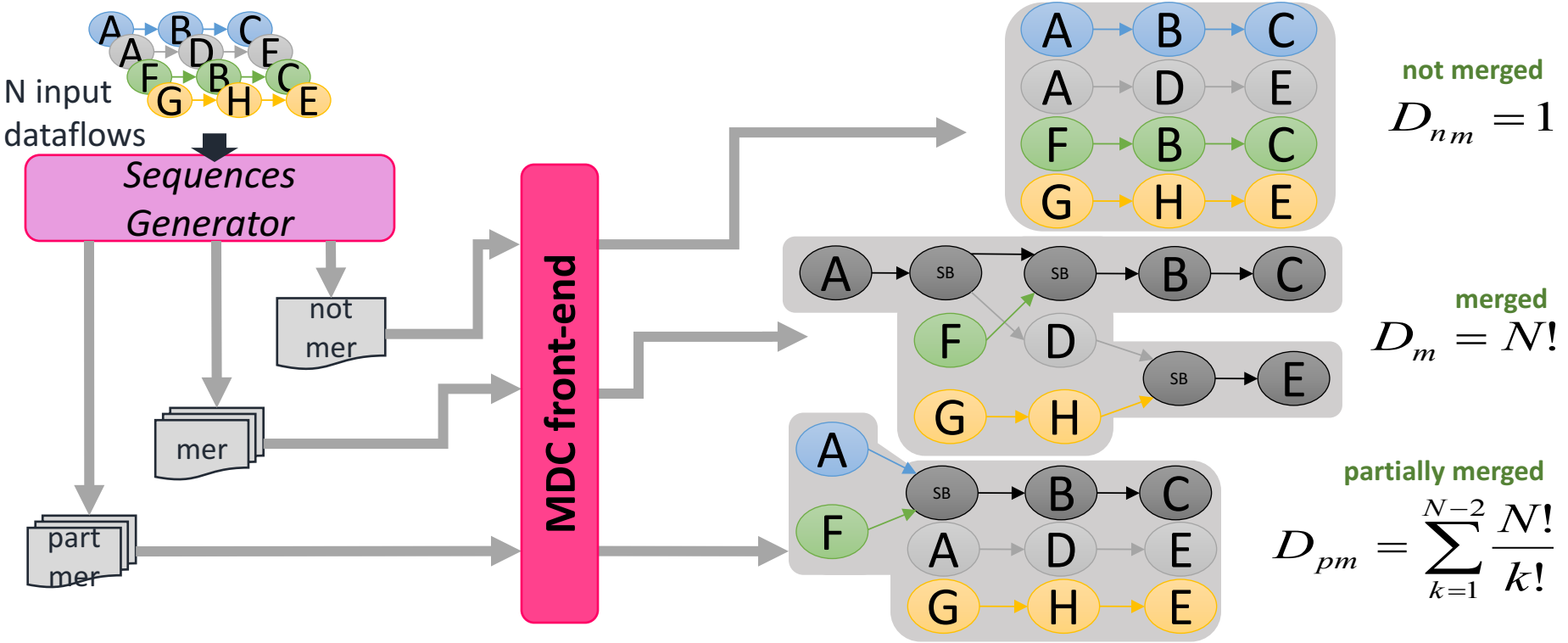
42 MHz

☹

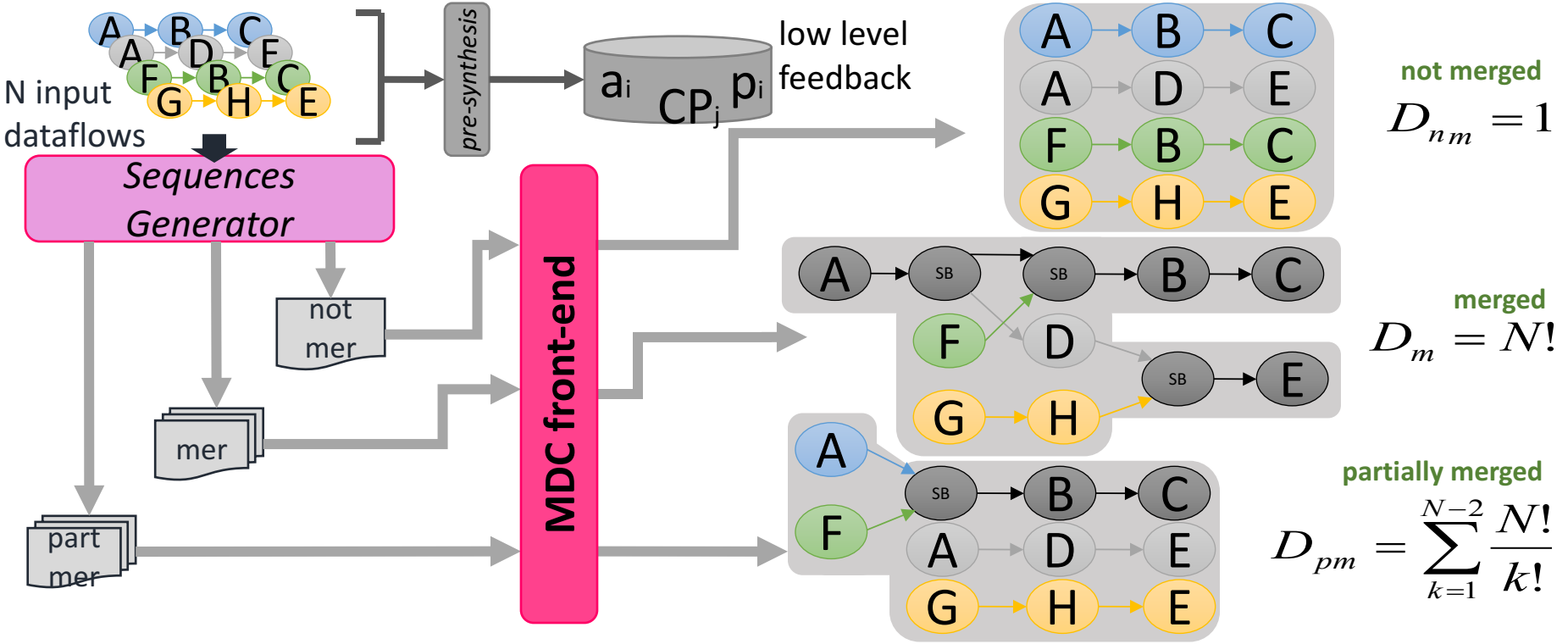
Structural Profiler



Structural Profiler



Structural Profiler

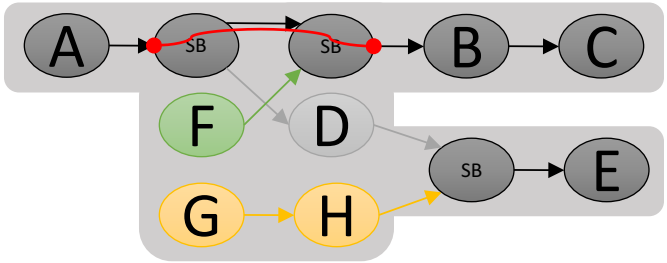


Structural Profiler

$$Area = \sum_{i=1}^{M} a_i$$

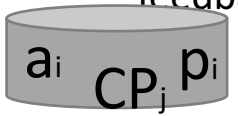
$$Power = \sum_{i=1}^{M} p_i$$

number of actors involved in the DP



current design point (DP)

low level feedback



a_i/ p_i = actor area/power

CP_j = input dataflow critical path

$$Frequency = \frac{1}{CP} = \frac{1}{\max(CP_{in}, CP_{SB})}$$

longest SB chain within the DP

$$CP_{in} = \max(CP_j)$$

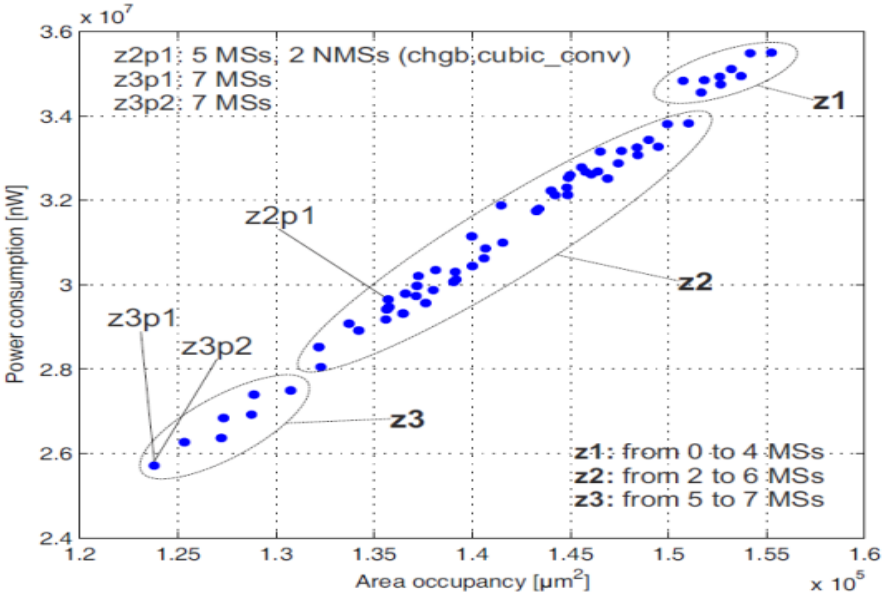
number of SBs in the DP chain

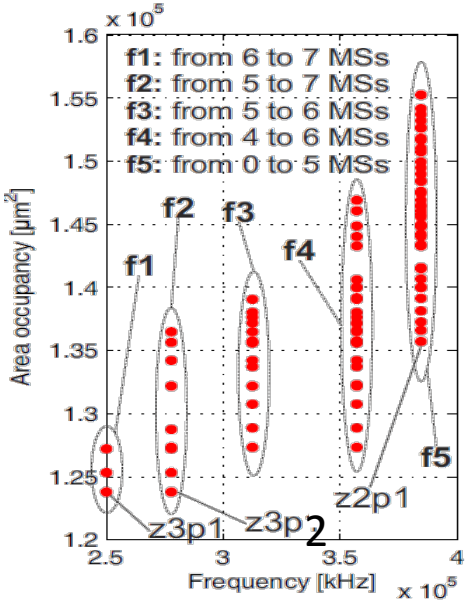
$$CP_{SB} = f(b) * \ln(N_{SB}) + g(b)$$

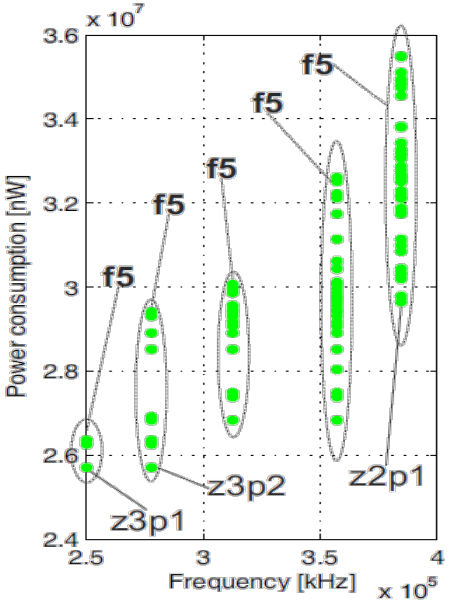
empirical functions of the SB size in bits *b*

Structural Profiler

Automated Pareto Analysis



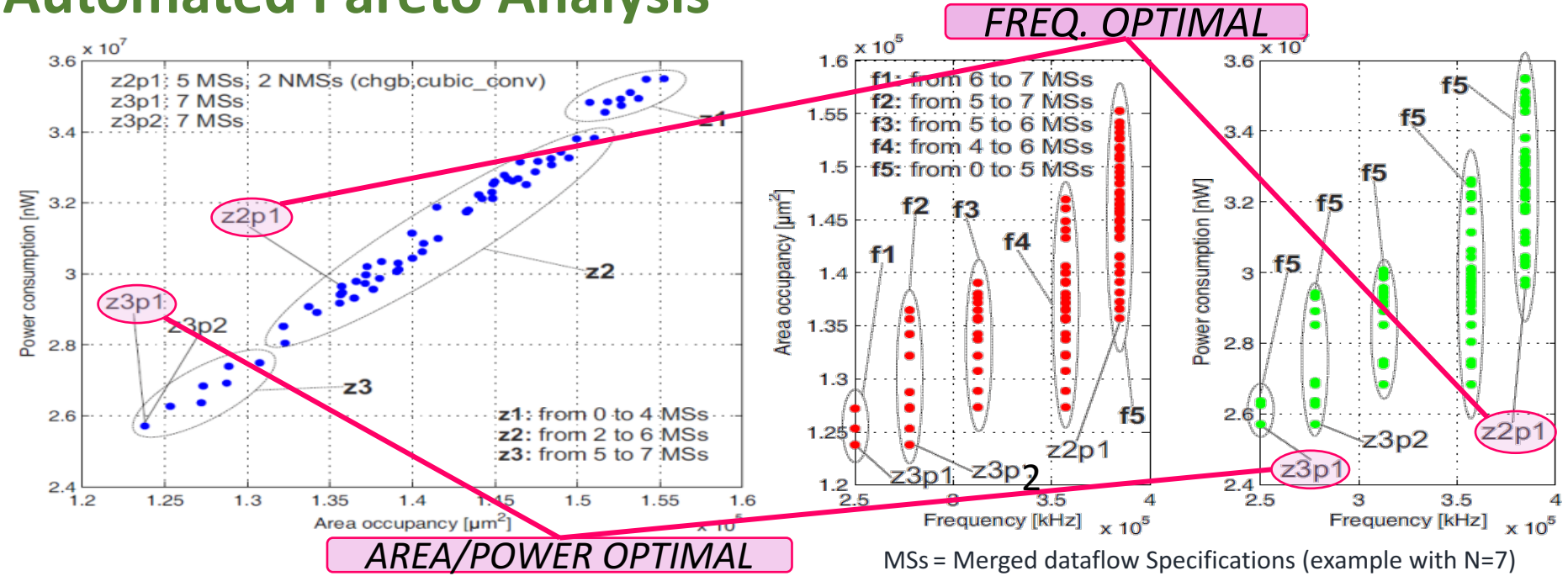




MSs = Merged dataflow Specifications (example with N=7)

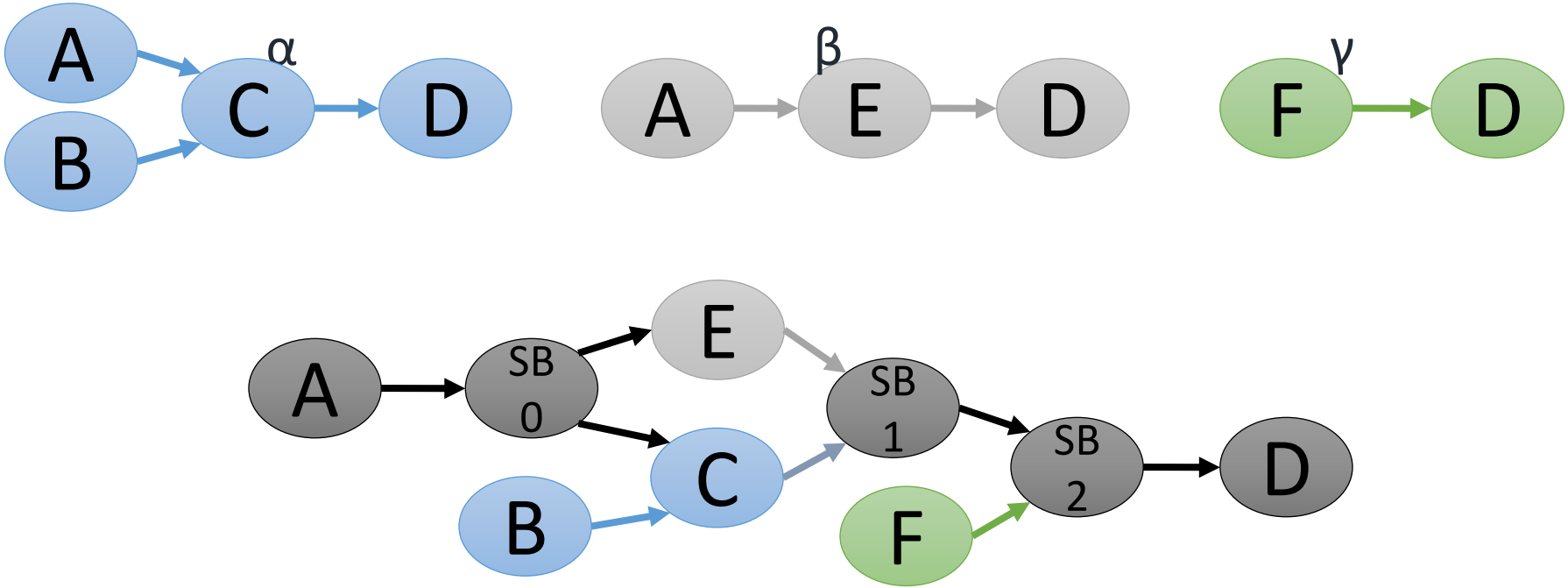
Structural Profiler

Automated Pareto Analysis

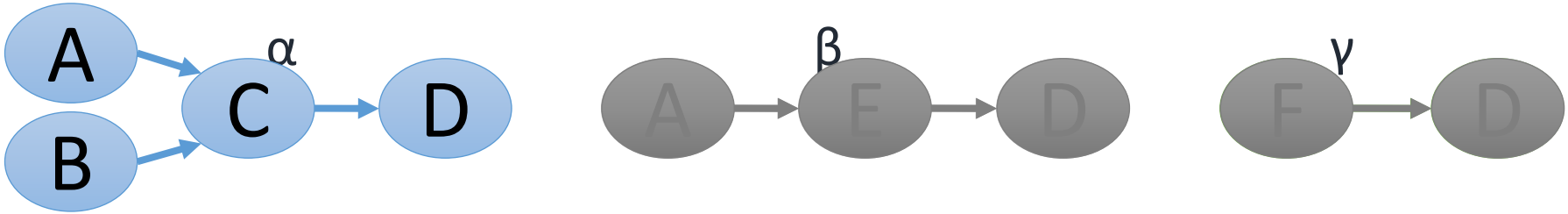


MSs = Merged dataflow Specifications (example with N=7)

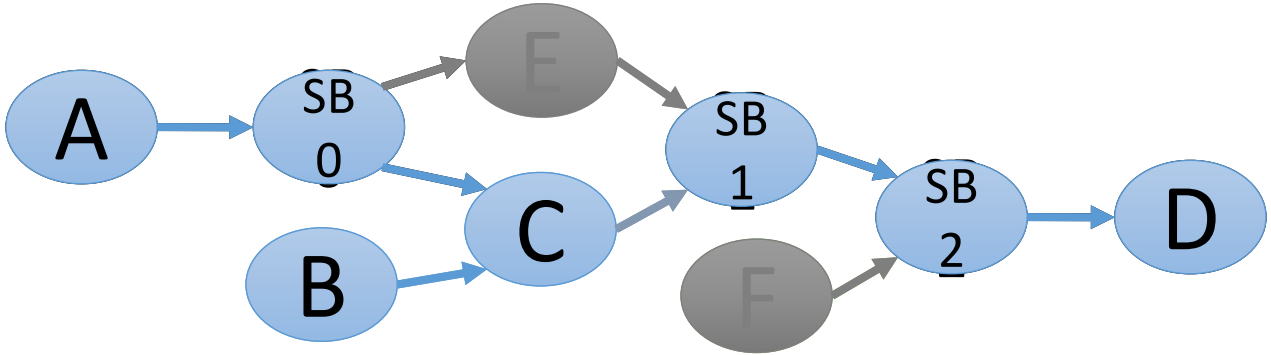
Dynamic Power Management



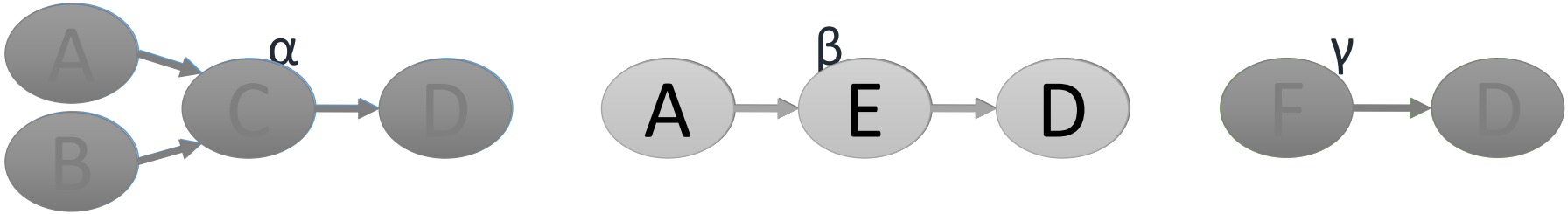
Dynamic Power Management



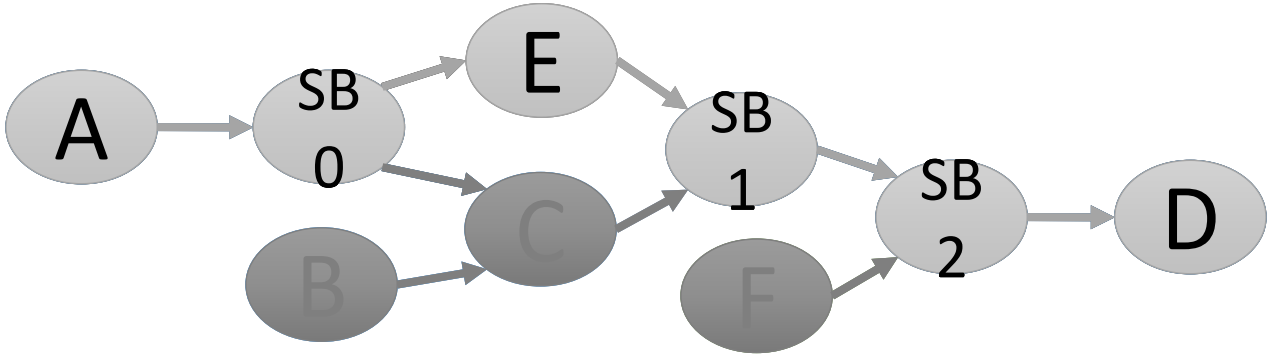
α execution: E and F are wasting power!



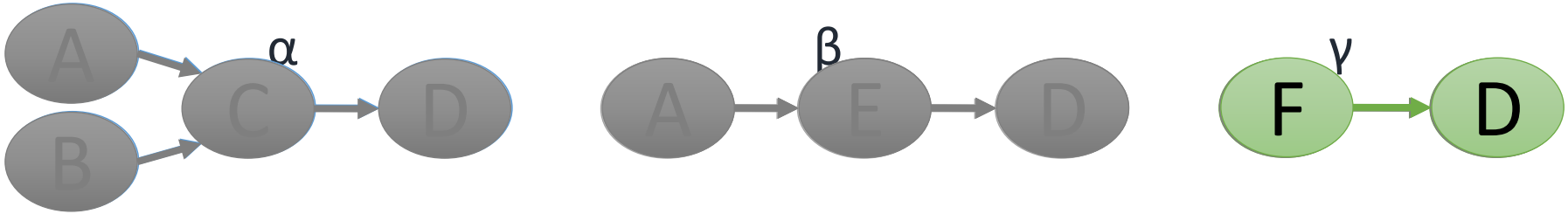
Dynamic Power Management



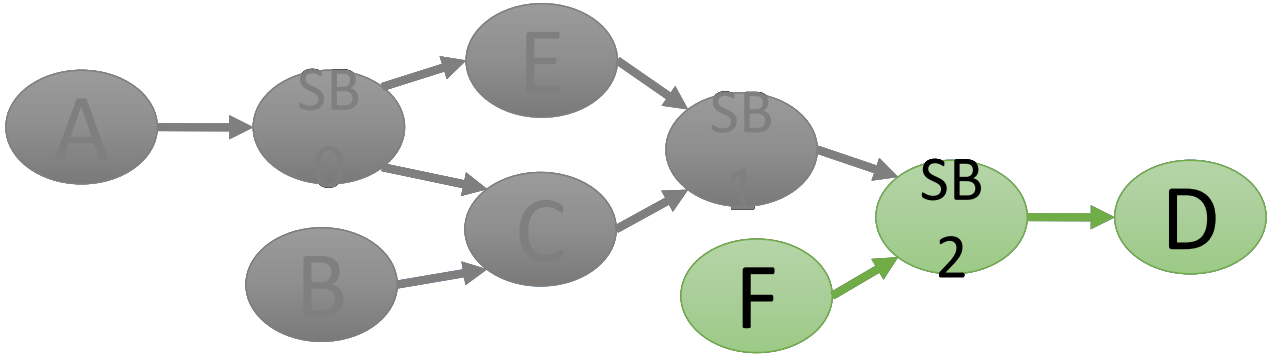
β execution: B, C and F are wasting power!



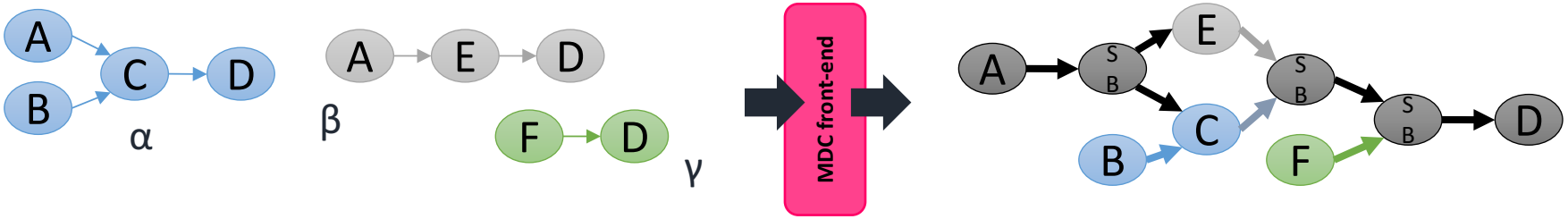
Dynamic Power Management



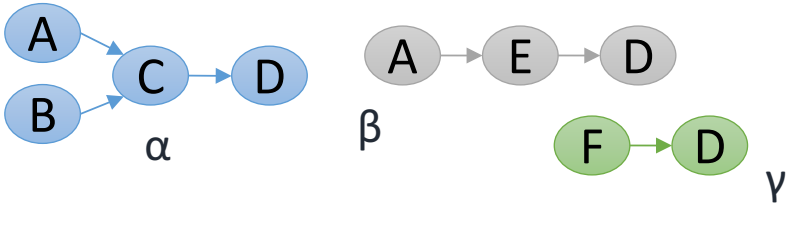
γ execution: A, B, C, E, SB0 and SB1 are wasting power!

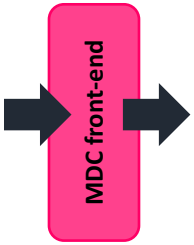


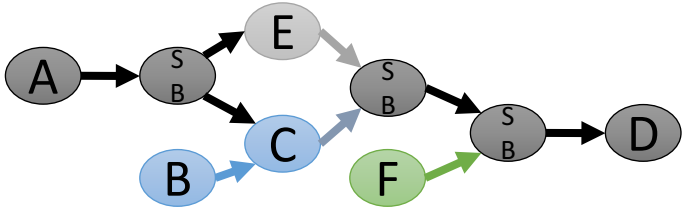
Dynamic Power Management



Dynamic Power Management

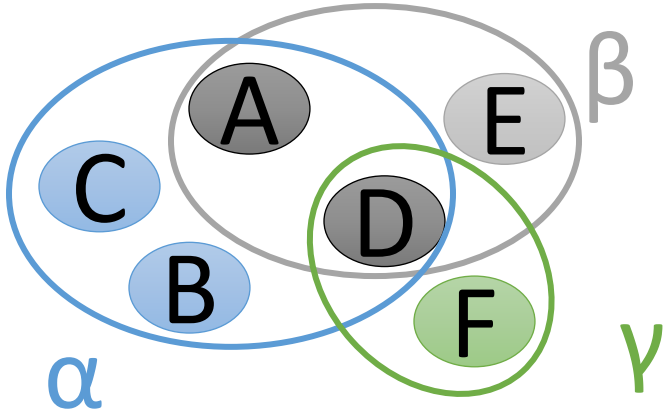






LR	1	2	3	4	5
actors	A	B,C	D	E	F
α	1	1	1	0	0
β	1	0	1	1	0
γ	0	0	1	0	1

Logic Regions (LRs) Identification



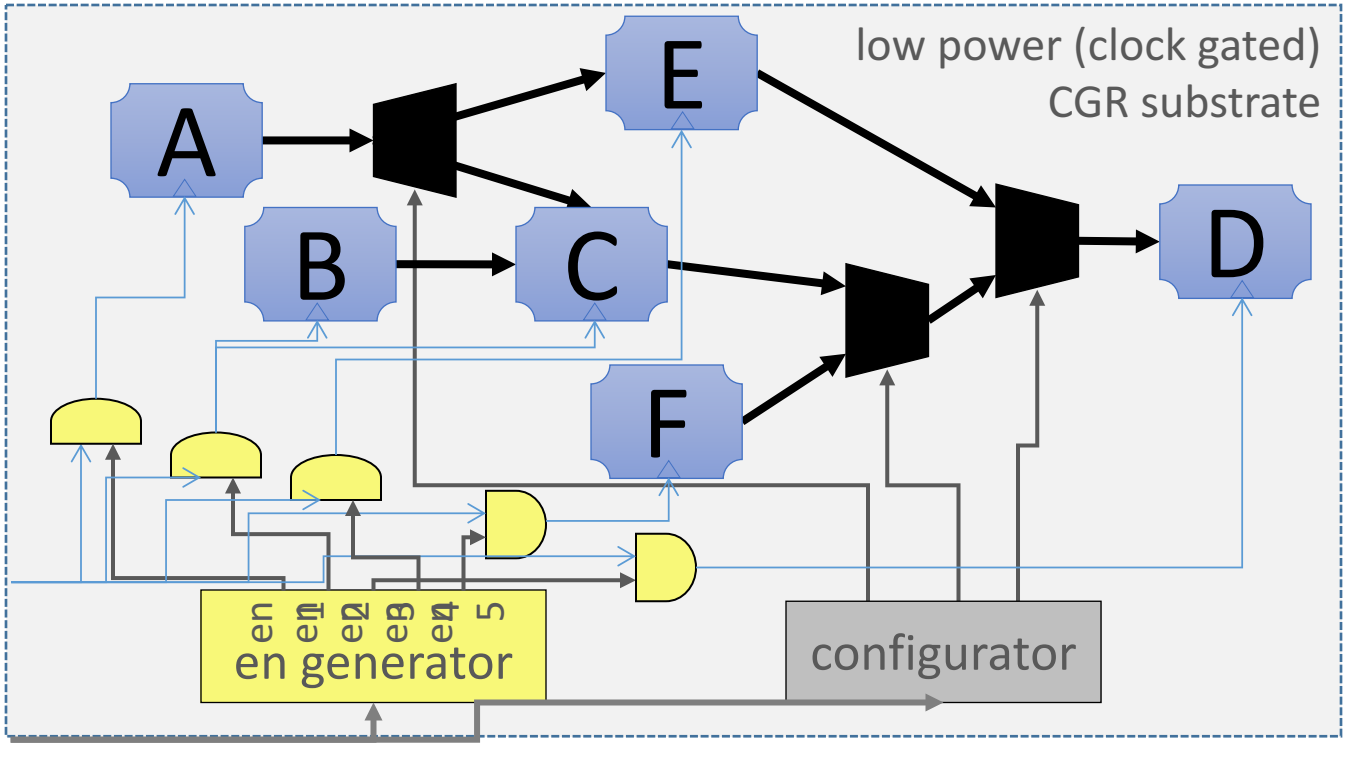
Dynamic Power Management

LR	actors	α	β	γ
1	A	1	1	0
2	B,C	1	0	0
3	D	1	1	1
4	E	0	1	0
5	F	1	0	1

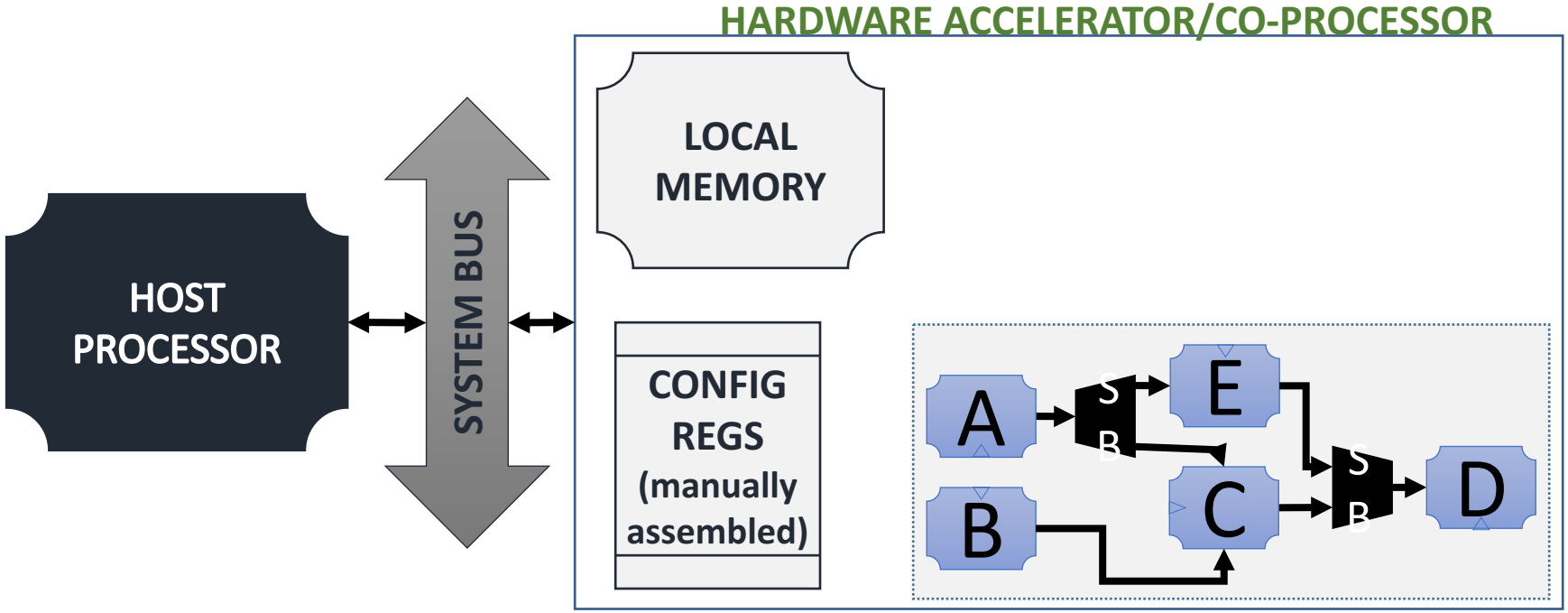
MDC back-end

clk

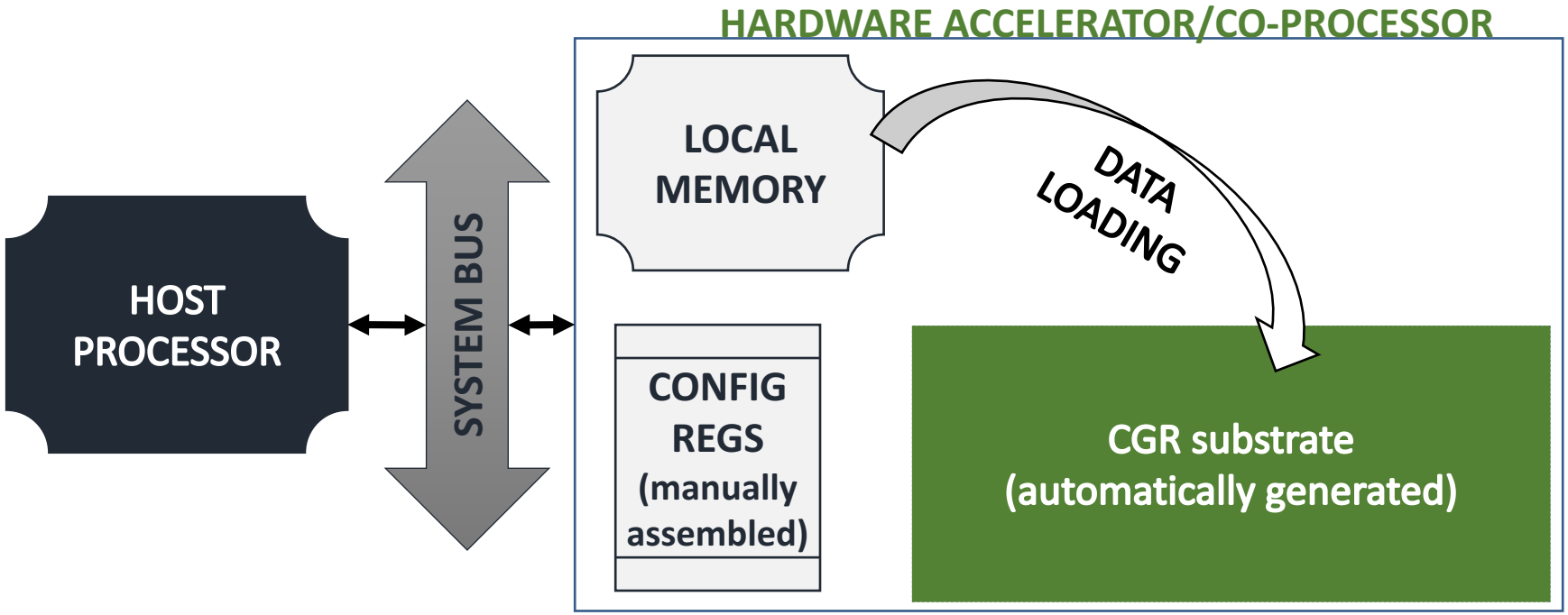
ID



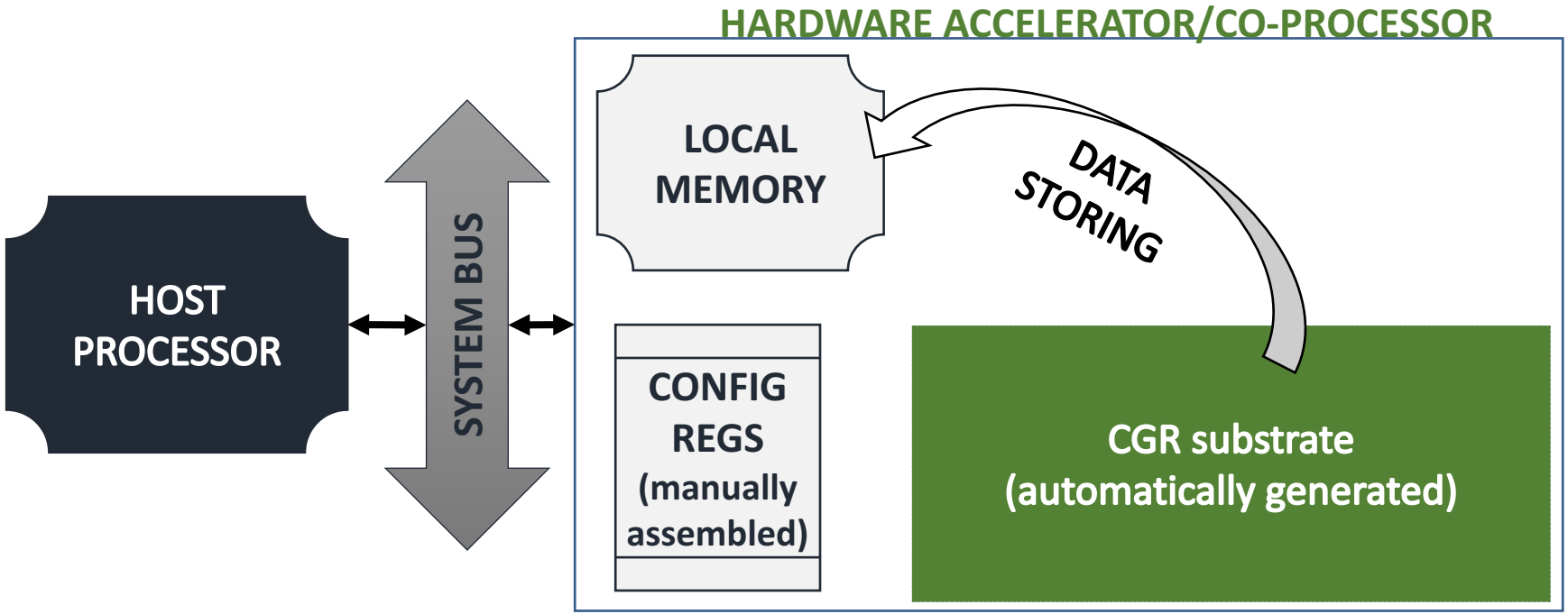
Co-processor Generator



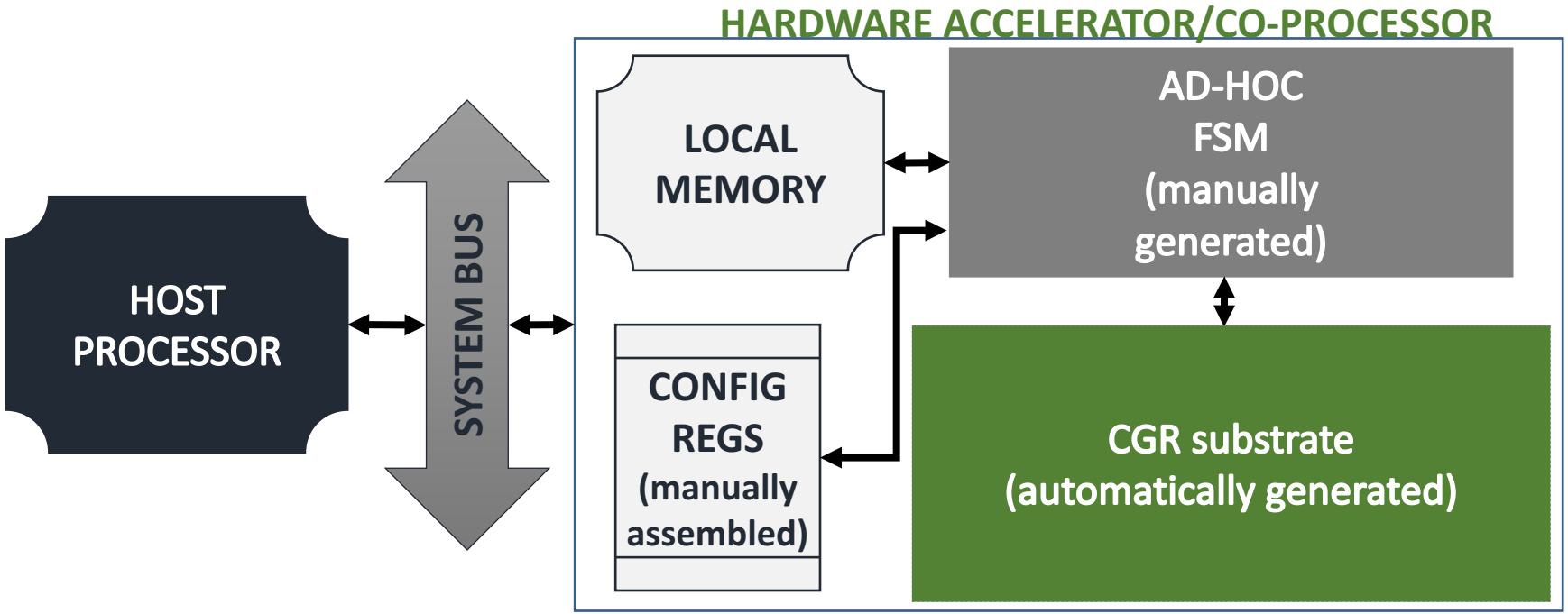
Co-processor Generator



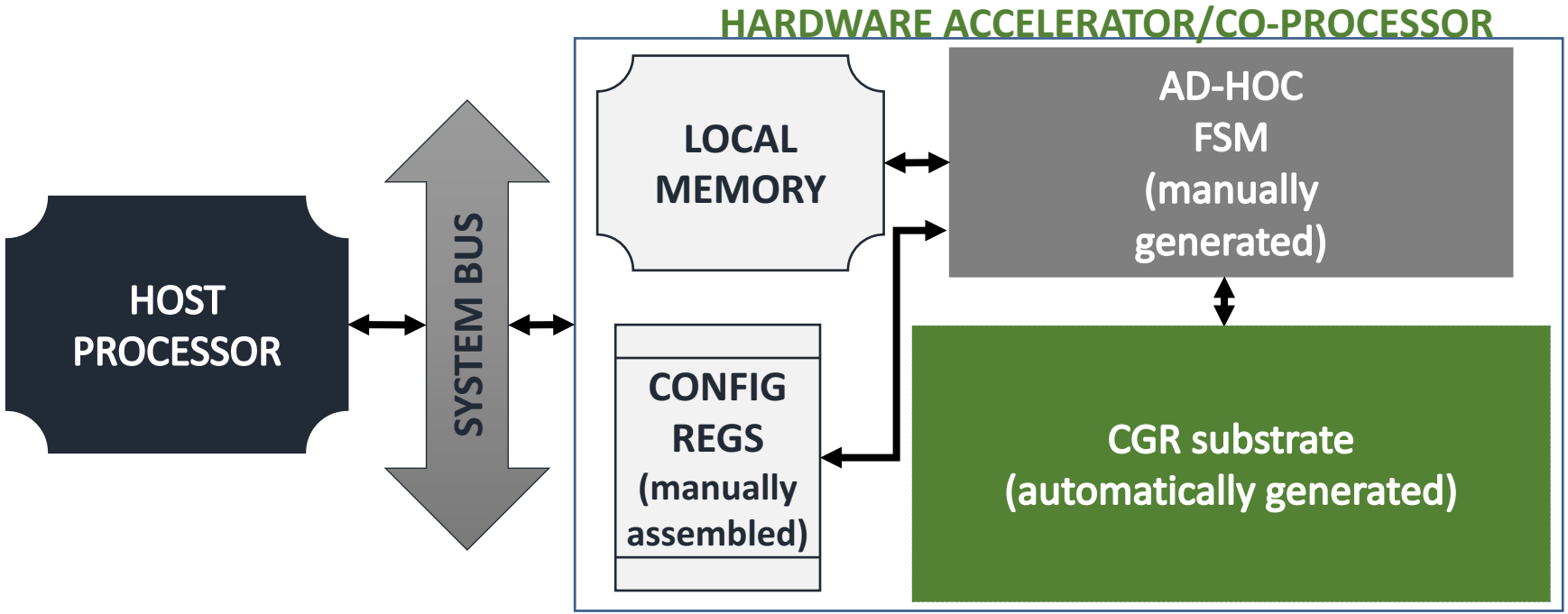
Co-processor Generator



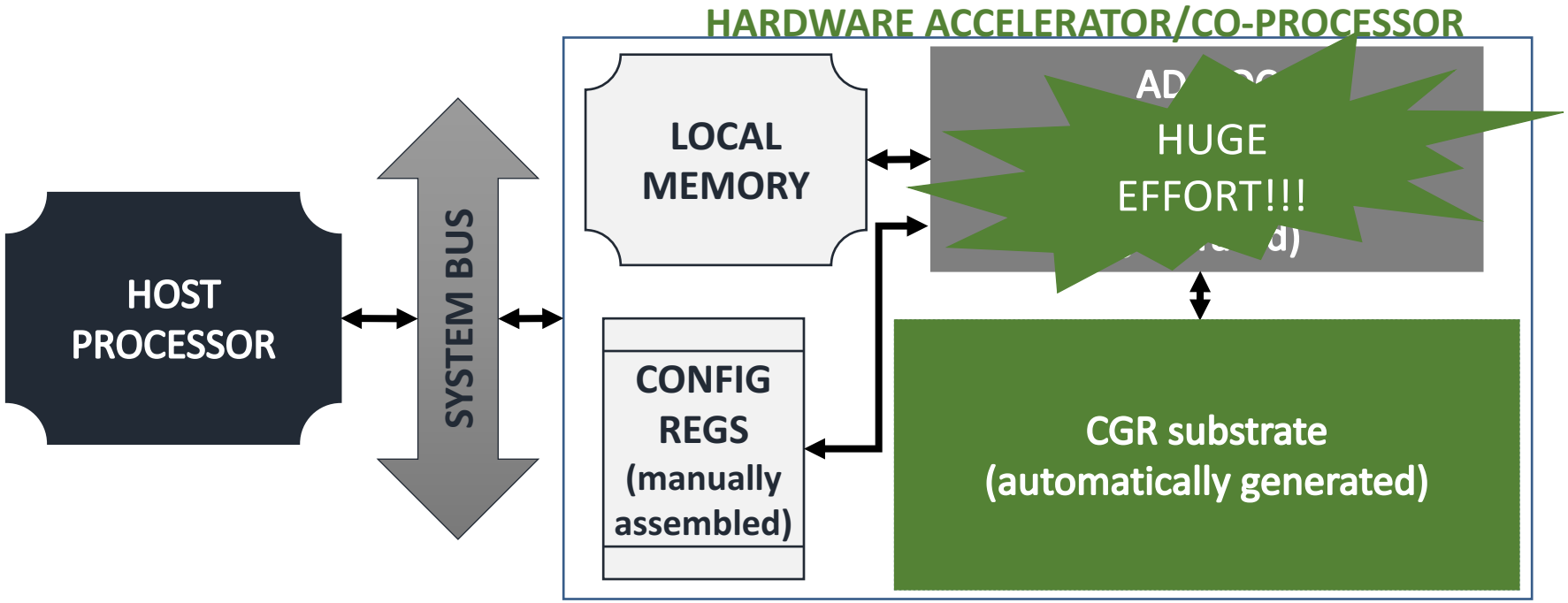
Co-processor Generator



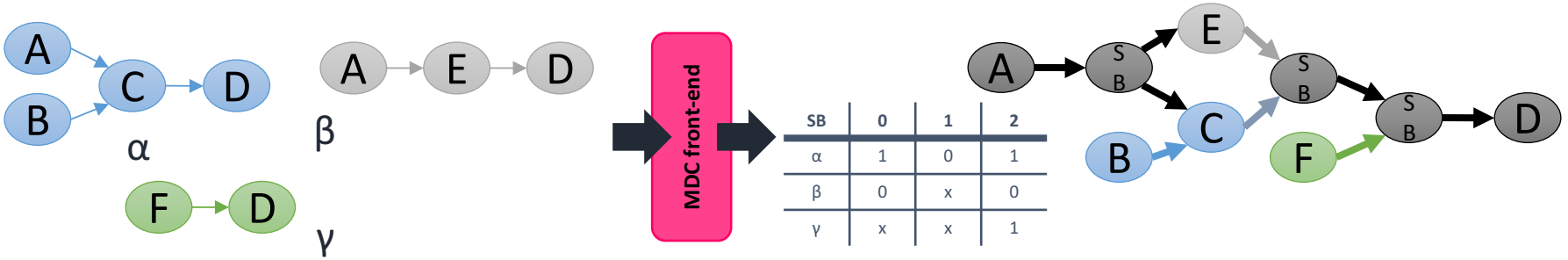
Co-processor Generator



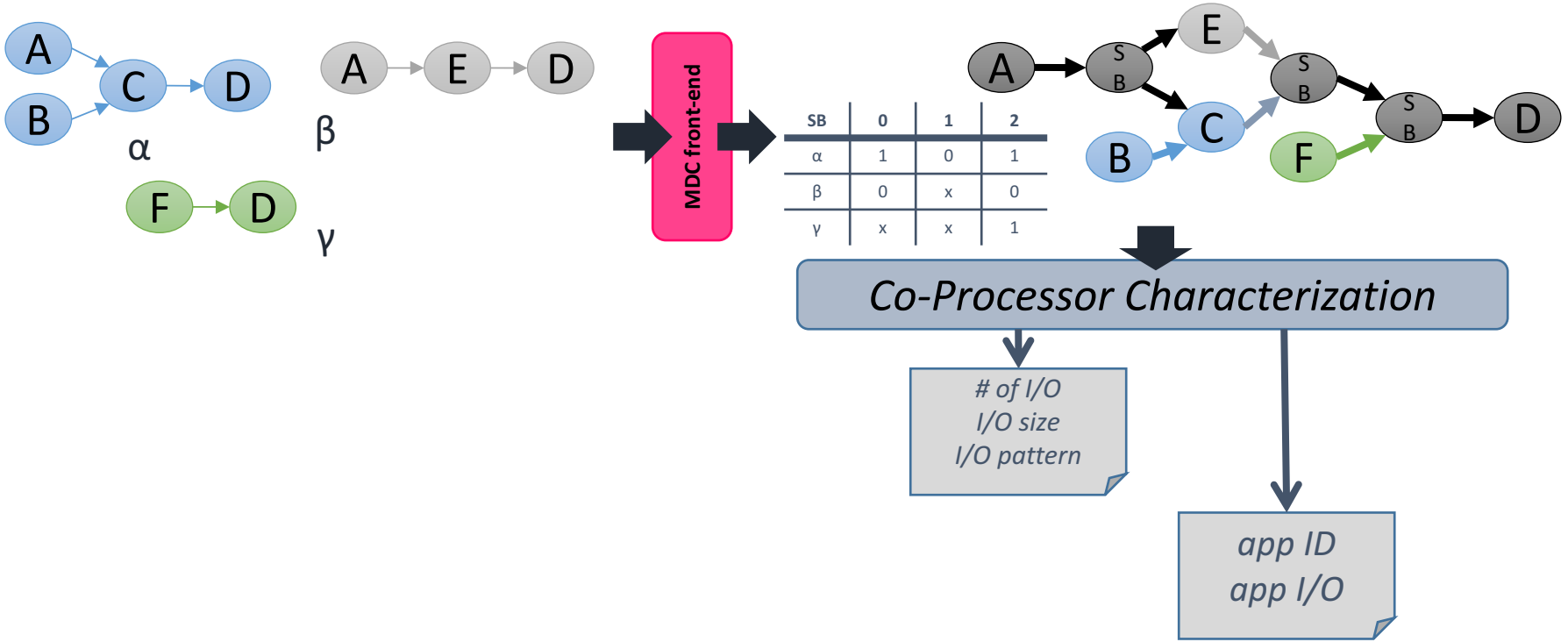
Co-processor Generator



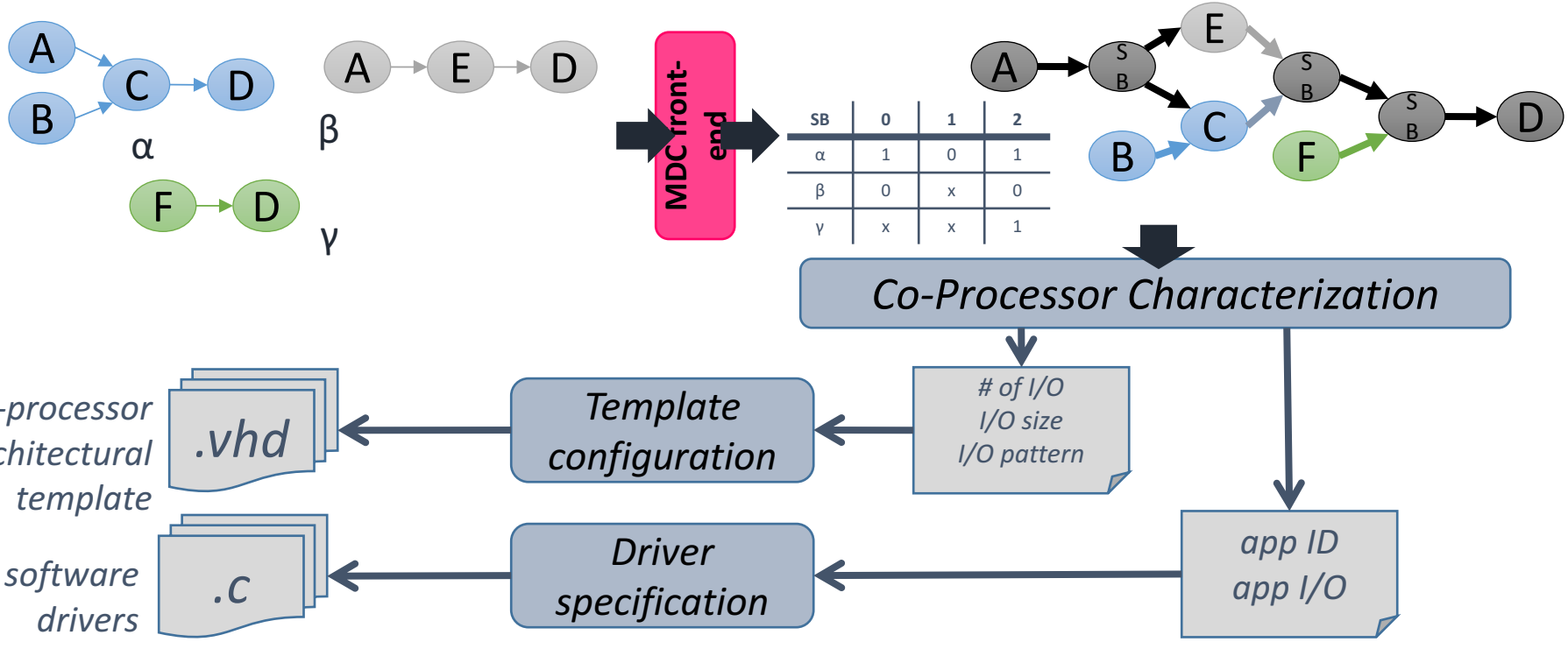
Co-processor Generator



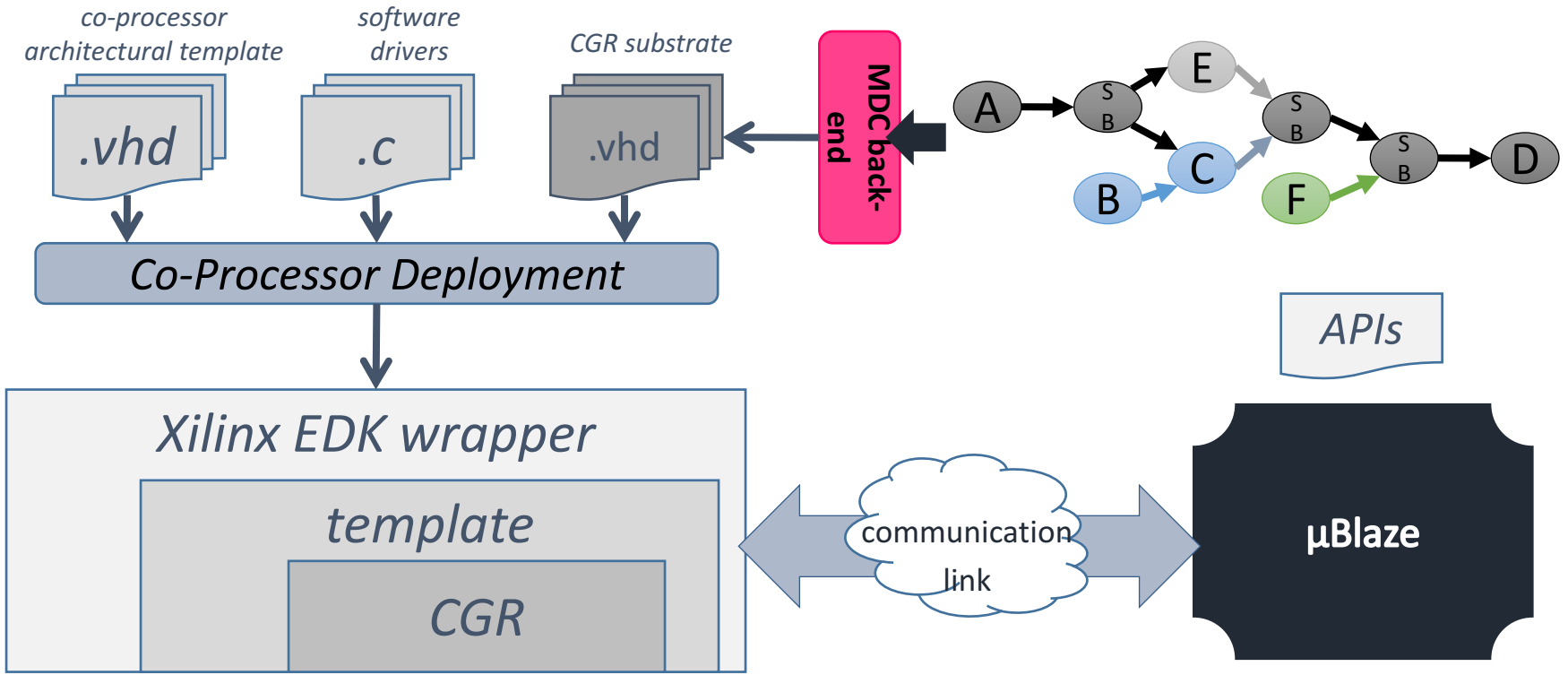
Co-processor Generator



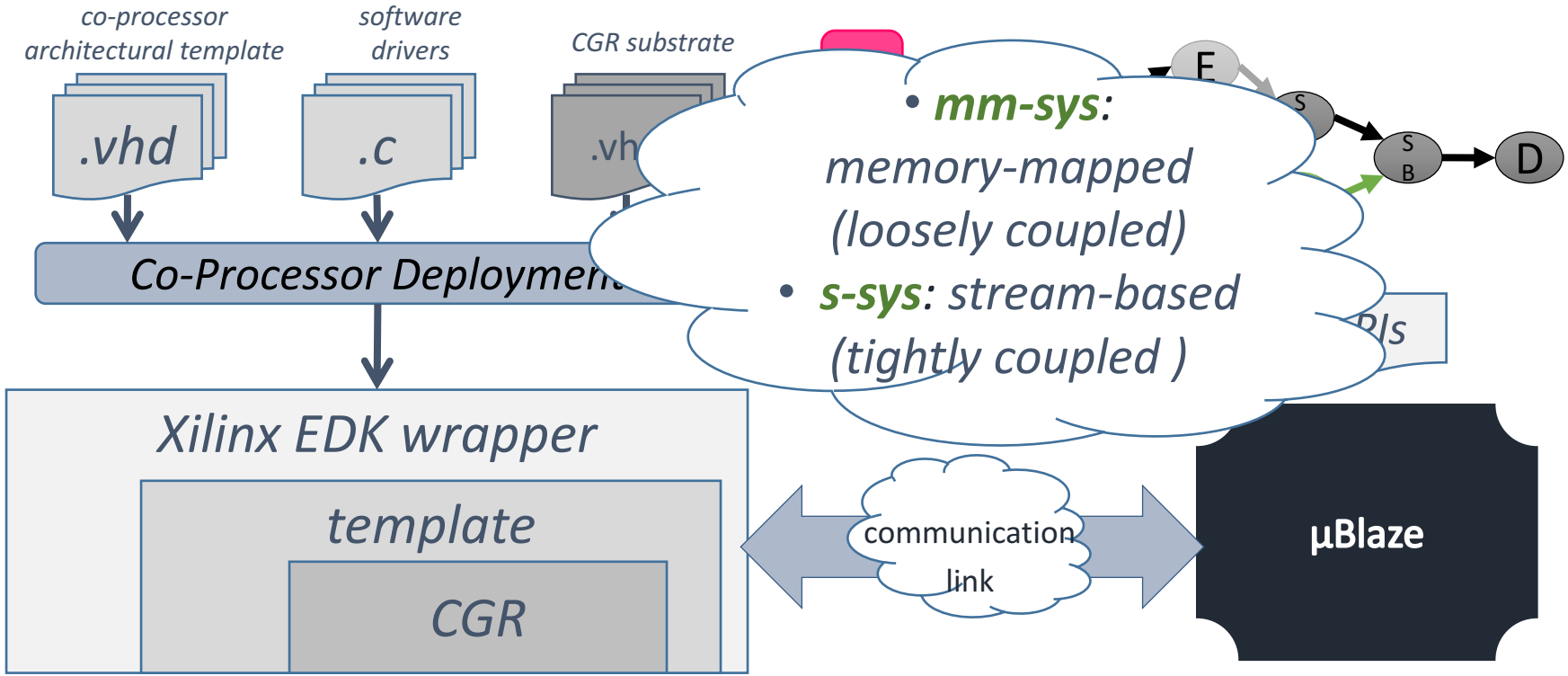
Co-processor Generator



Extension: Co-processor Generator



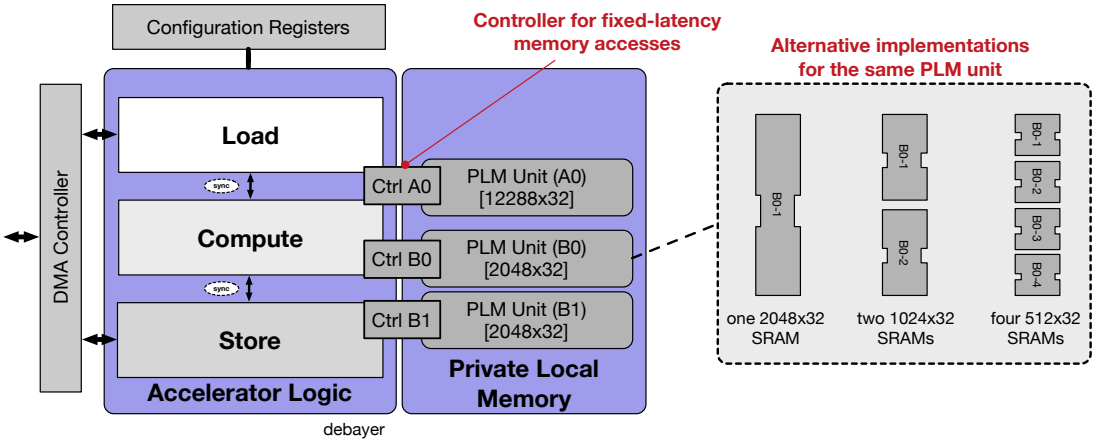
Extension: Co-processor Generator



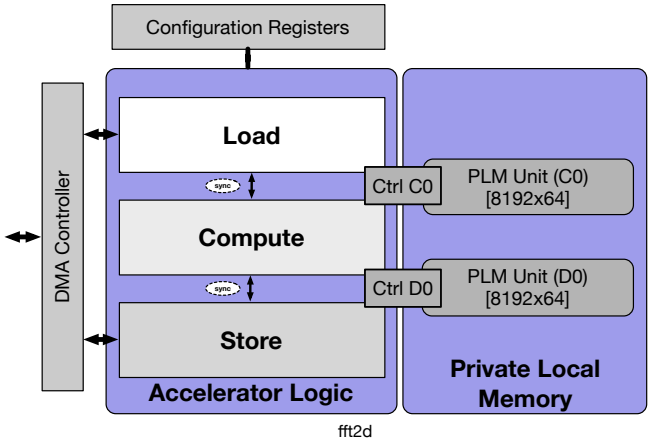
How to Design Private Local Memories?

Dedicated (multi-port) local memories for storing part of the data

- memory design transparent to accelerator logic
- alternative implementations with **block/cyclic partitioning**



debayer

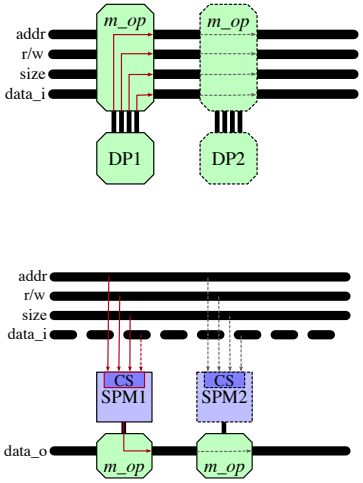


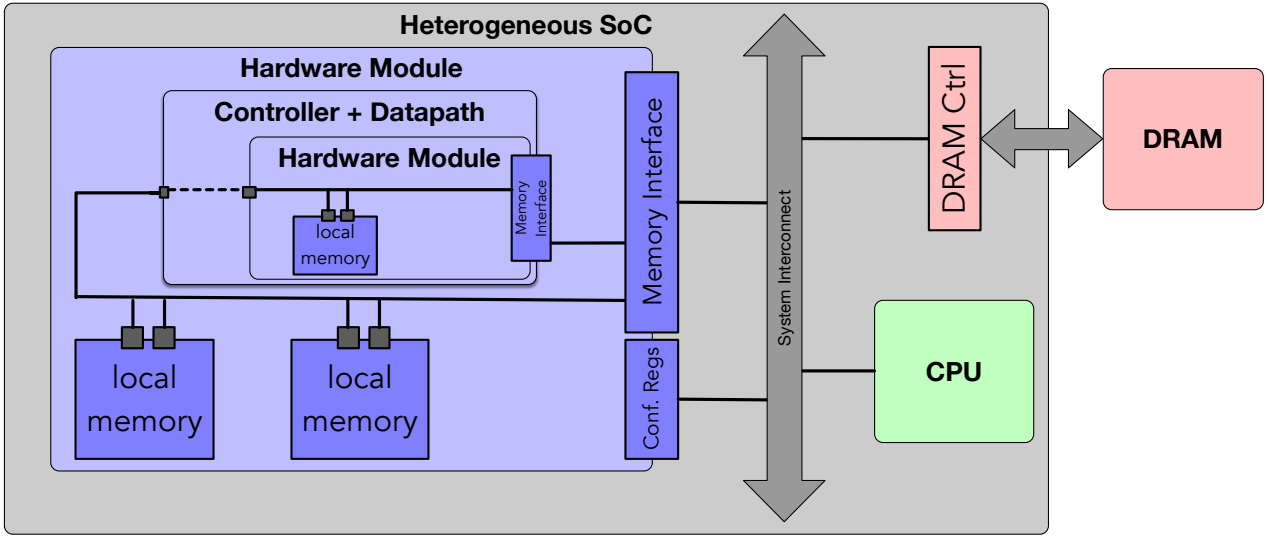
fft2d

RTL Architecture for Irregular Applications

Internal memory bus where the pointer is dynamically resolved

- Daisy-chain architecture with possibility of accessing the external memory





Pilato *et al.*
CODES 2011

Multi-port Memories are not for Free

Distributed registers (e.g. flip-flops)

- Many ports at the cost of more area
- Good for small to medium data structures

1024x32 array in an industrial CMOS 32nm technology

Distributed registers
145,707.5 um^2

Memory Intellectual Property (IP) blocks

- Area-efficient macro blocks provided by the technology vendors
  - SRAMs for CMOS and BRAMs for FPGA
- Good for medium to large arrays
- Limited number of ports (usually no more than two!)

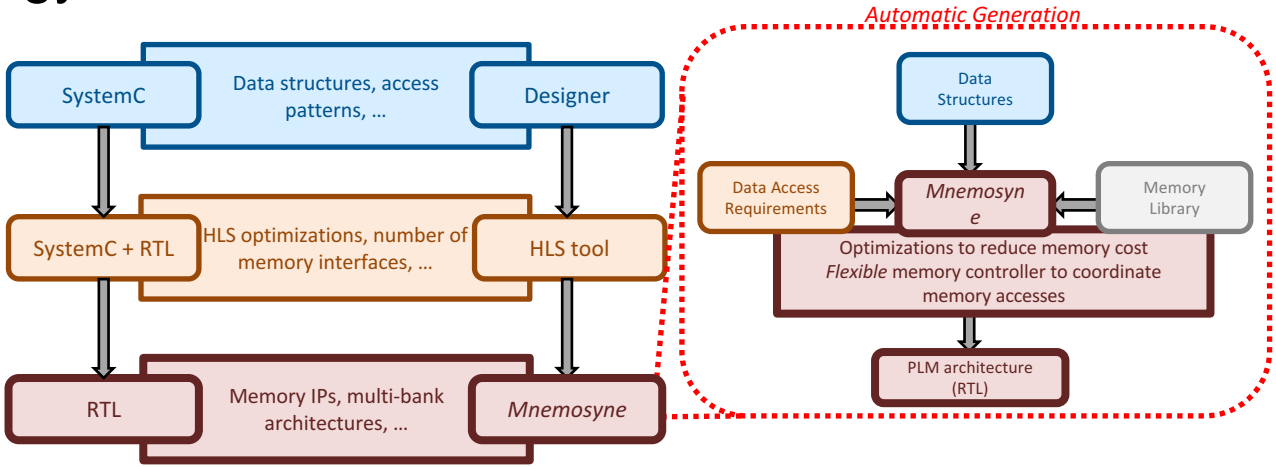
Memory IP block
35,106.6 um^2

(4x area reduction)

Multi-bank architectures based on memory IPs

Mnemosyne

Prototype CAD tool that implements a complete system-level methodology for **PLM customization**



Performance optimization: *HLS* defines how the accelerator logic accesses the data structures (e.g. number of parallel accesses)

Cost optimization: *Mnemosyne* defines the best memory architecture able to guarantee the desired performance (e.g. number of banks, data allocation)

Reuse What is not Used

Generally we can use one **PLM unit** (eventually composed of many banks) for each data structure

“Two data structures are compatible if they can be allocated to the same PLM unit (memory IPs)”

Reuse the same memory IPs
for several data structures

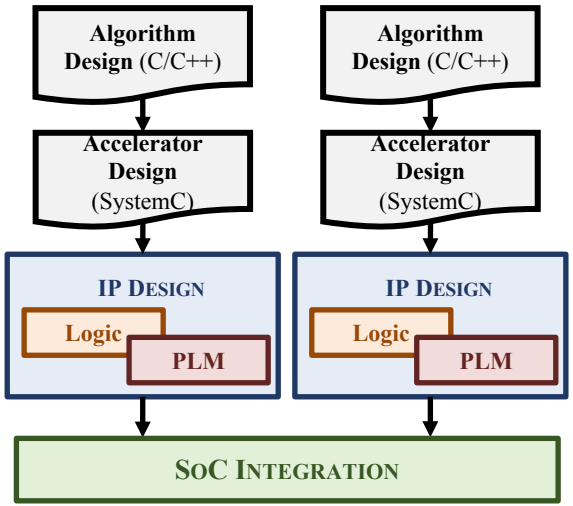
A common case: accelerators never executed at the same time

- Possible only at the system-level, when integrating the components
- Optimizations of accelerator logic and memory subsystem are independent

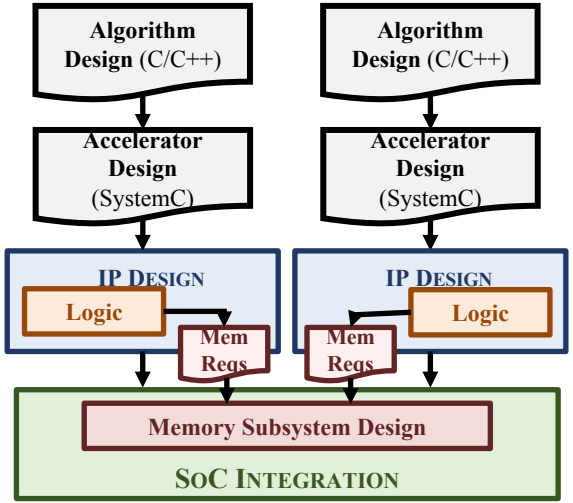
Optimization only at the System-Level

Accelerator(s) memory subsystem is defined during SoC integration

- Possibility for more optimizations

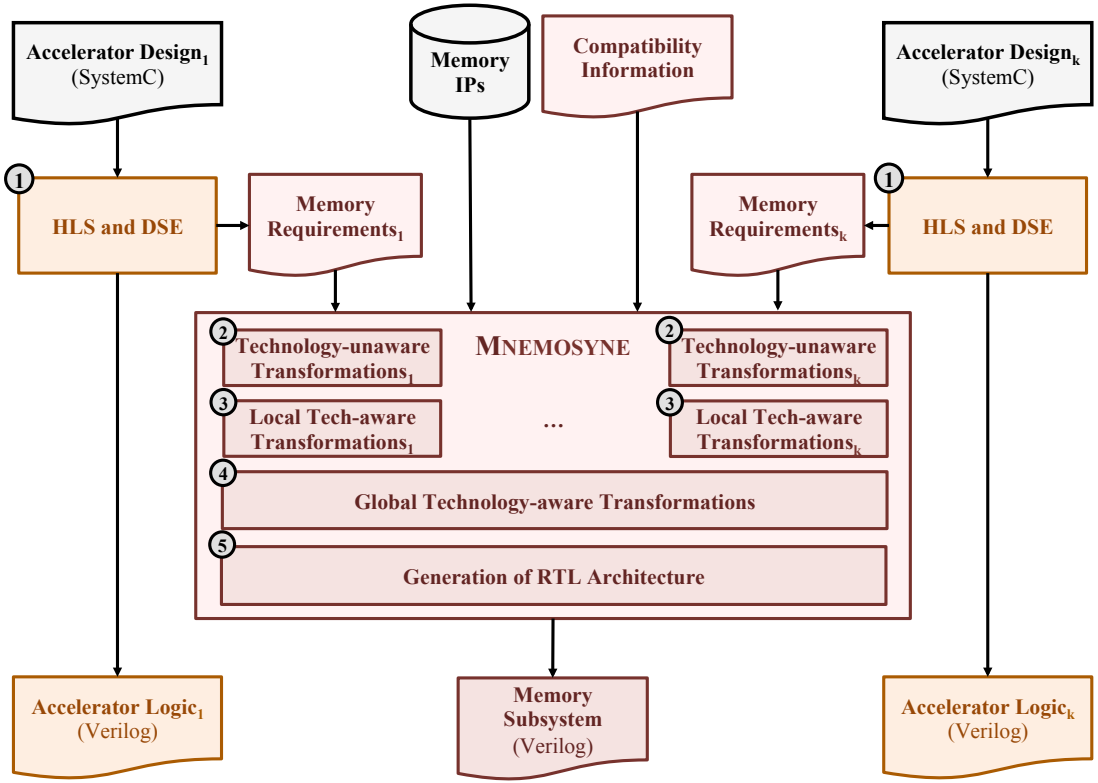


Traditional Approach



Our Approach

Optimization of Multiple Accelerators

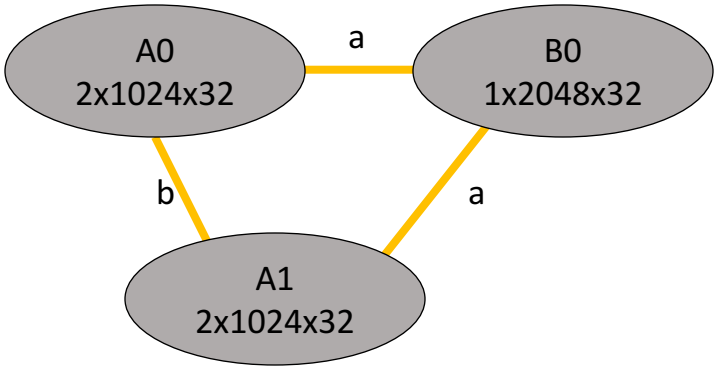


Pilato *et al.*
TCAD 2017

Memory Compatibility Graph

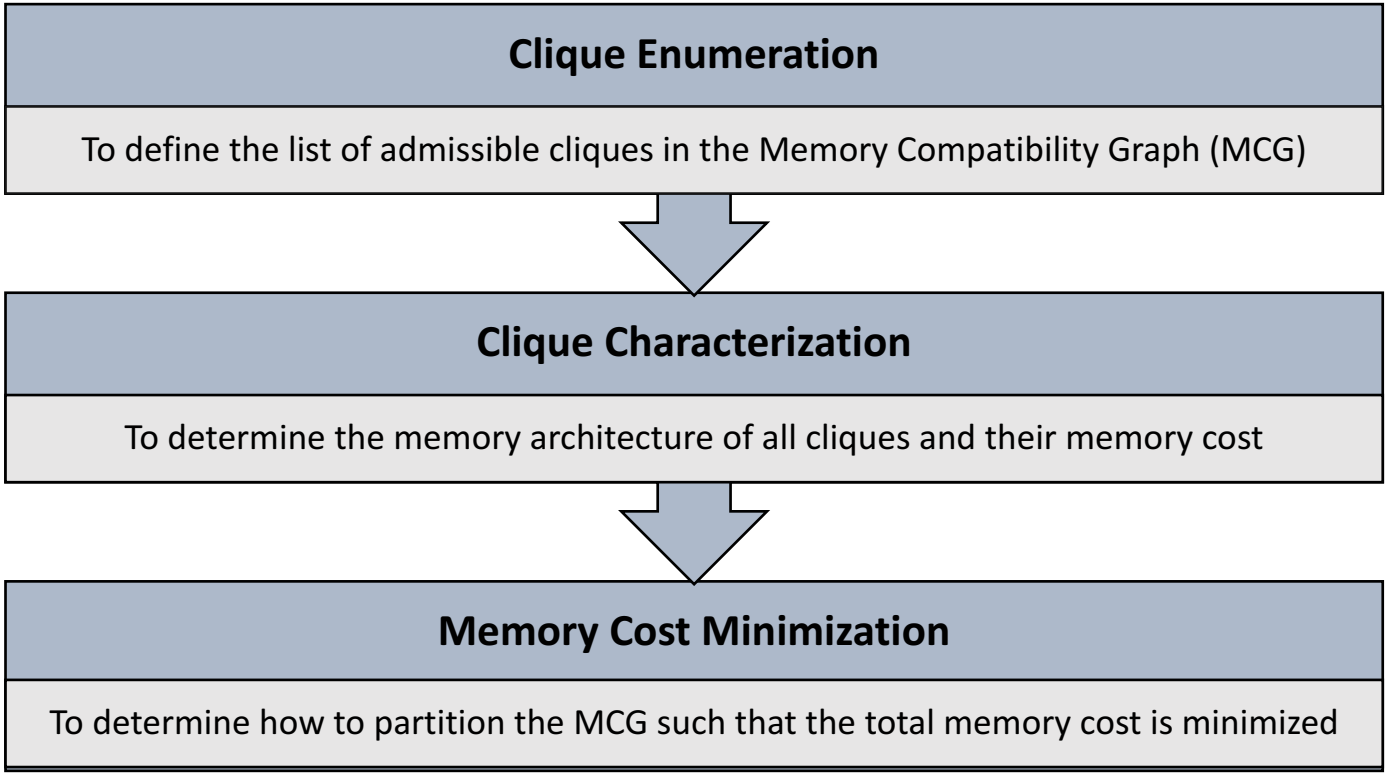
Graph to represent the possibilities for optimizing the data structures

- Each node represents a data structure to be allocated, annotated with its data footprint (after data allocation)
- Each edge represents compatibility between the two data structures



a) **Address-space compatibility**: the data structures are compatible and can use the same memory IPs
b) **Memory-interface compatibility**: the ports are never accessed at the same time and the data structures can stay in the same memory IP

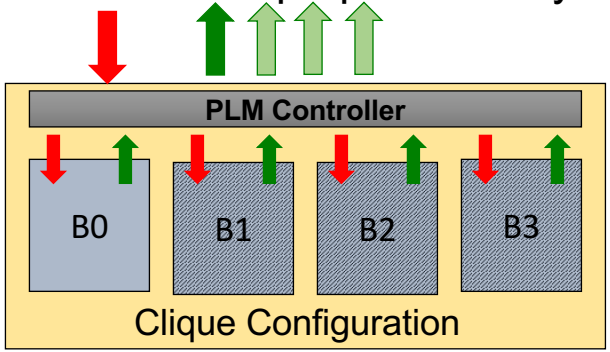
How to Determine the Memory Subsystem

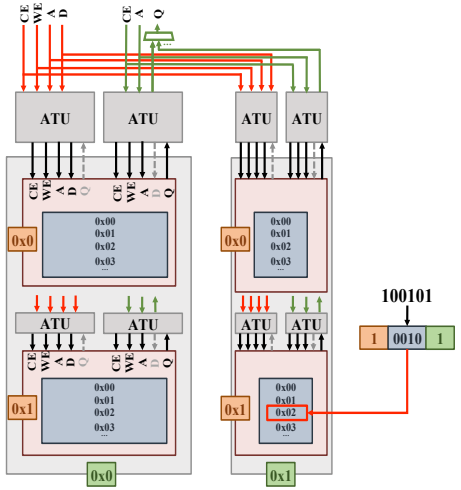


PLM Controller Generation

A **lightweight PLM controller** is created for each clique based on the bank configuration

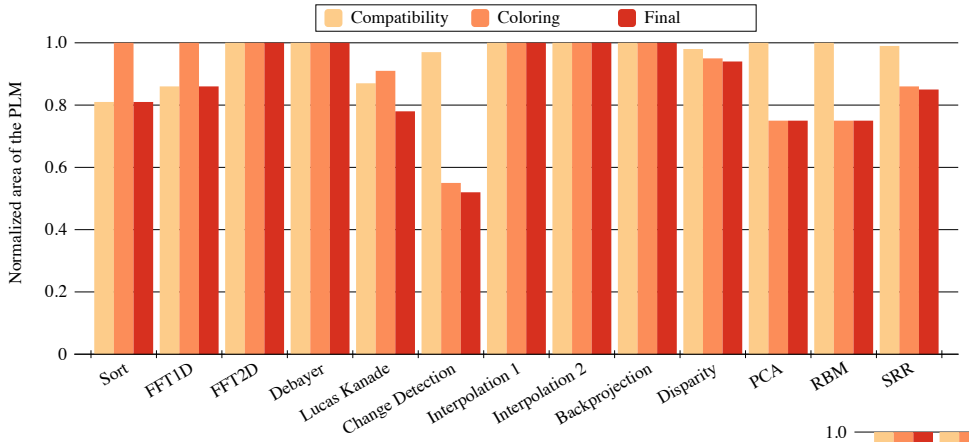
- Accelerator logic is not aware of the actual memory organization
- Array offsets need to be translated into proper memory addresses





Custom logic with negligible overhead, especially when the number of banks and their size is a power of two

Impact of Optimizations

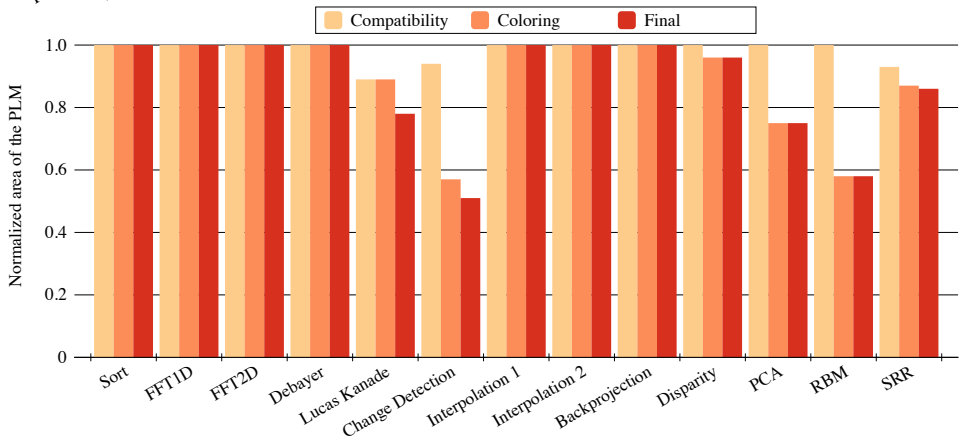


Industrial 32nm CMOS technology

- Memory library with 18 SRAMs

Xilinx Virtex-7 FPGA

- Memory library with 6 BRAM configurations



Multiple Accelerators in Time Multiplexing

We created four realistic scenarios:

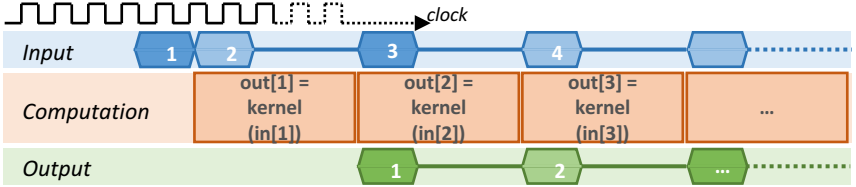
- **Required**: Sort, FFT1D, FFT2D
- **WAMI**: Debayer, Lucas Kanade, Change Detection
- **SAR**: Interpolation 1, Interpolation 2, Backprojection
- **Cortex**: Disparity, SRR, PCA, RBM

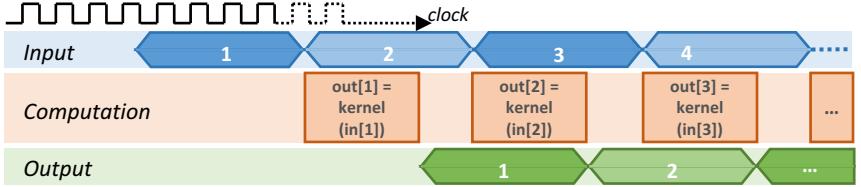
Bench.	Data Structures		CMOS			FPGA		
	(#)	(KB)	#Ctrl.	(KB)	Diff (%)	#Ctrl.	(KB)	Diff (%)
Required	13	192.00	4	140.00	-32.05	4	192.00	-29.17
WAMI	25	178.21	8	131.00	-43.93	8	212.00	-57.55
SAR	21	211.10	8	100.00	-55.07	8	216.00	-53.07
Cortex	54	404.23	32	653.50	-36.42	32	690.00	-46.67

Accelerators Executing Concurrently

Balancing communication and computation is crucial for performance optimization

- Optimizing microarchitecture reduces the **computation latency**
  - Combination of HLS transformations and PLM customization





- **Input and output phases** interact with the rest of the system
  - Backpressure due to congestion may increase the latency

Reduce the congestion or **exploit the congestion** to optimize the execution at the system level

Optimizing the System Execution

Reduce congestion: Smart data allocation on multiple controllers

- Partition the data set across multiple memory spaces (1-4 MB pages) with a custom Linux module
- Configure hardware TLB with virtual-to-physical addresses
- DMA controller generates transactions to the proper memory controller

Mantovani *et al.*
CASES 2016

Exploit congestion: dynamic power management with DVFS

- Vary the execution mode of the accelerators (voltage/frequency) based on the workload
- Only requires local probes to determine when an accelerator is stalling after a data request

Mantovani *et al.*
DAC 2016

Accelerators are Becoming More and More Complex

Complex accelerators with large PLMs are becoming an important source of **power consumption**

- Leakage is becoming more and more critical (45nm and below)
  - almost 70% of total power consumption
  - SRAM leakage contributes for more than 50% to the total leakage
  - can be reduced by more than 70% by reducing supply voltage
  - Dual-rail SRAMs not sufficiently exploited

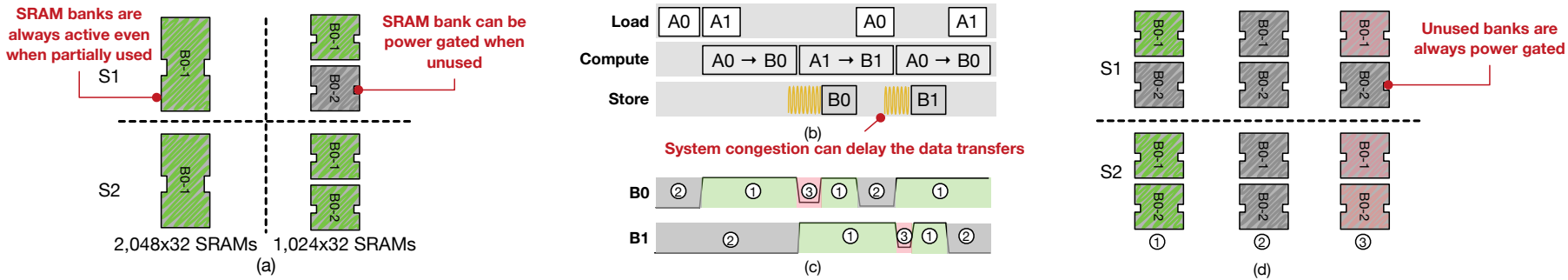
Fine-grained power management is gaining a lot of attention

- Number of accelerators is usually larger than the number of memory controllers (communication bottleneck)

During congestion we can reduce the power consumption of the accelerators

How to Dynamically Control the Banks?

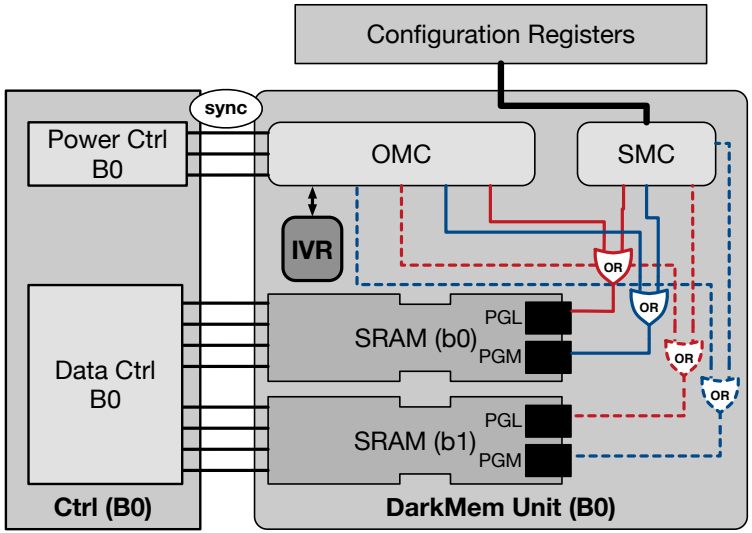
- PLM units are not entirely used in all configurations (scenarios)
  - **Scenario-based optimization**: partitioning of the banks to maximize the banks that are power gated

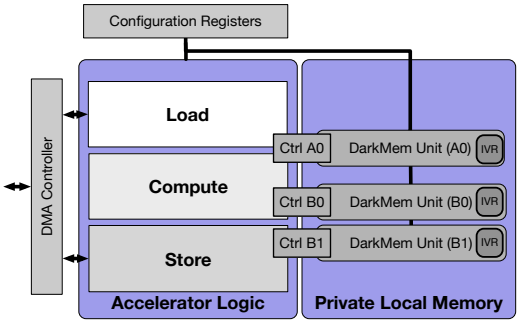


  - **Workload-based optimization**: dynamic control of the logic/cell power gating based on the execution phases (e.g., during congestion)

DarkMem Architecture

- Each PLM unit is extended with power-control logic
  - **SMC** identifies the current execution scenario
  - **OMC** manages the SRAM operating modes based on signals from the accelerator logic





- Fine-grained control of each SRAM power pin (PGL and PGM)

Pilato *and Carloni*
ASPDAC 2018

Determining the Bank Configuration

- **ILP formulation** to determine the number and type of banks for each PLM unit, based on:
  - Data to be stored (bitwidth and number of words) in each scenario
  - List of scenarios and frequency of execution
  - List of available memory IPs and corresponding active/gated static power

$$PLM_{static} = \sum_{s \in S} \left(PLM^{s}_{static} \cdot freq(s) \right)$$

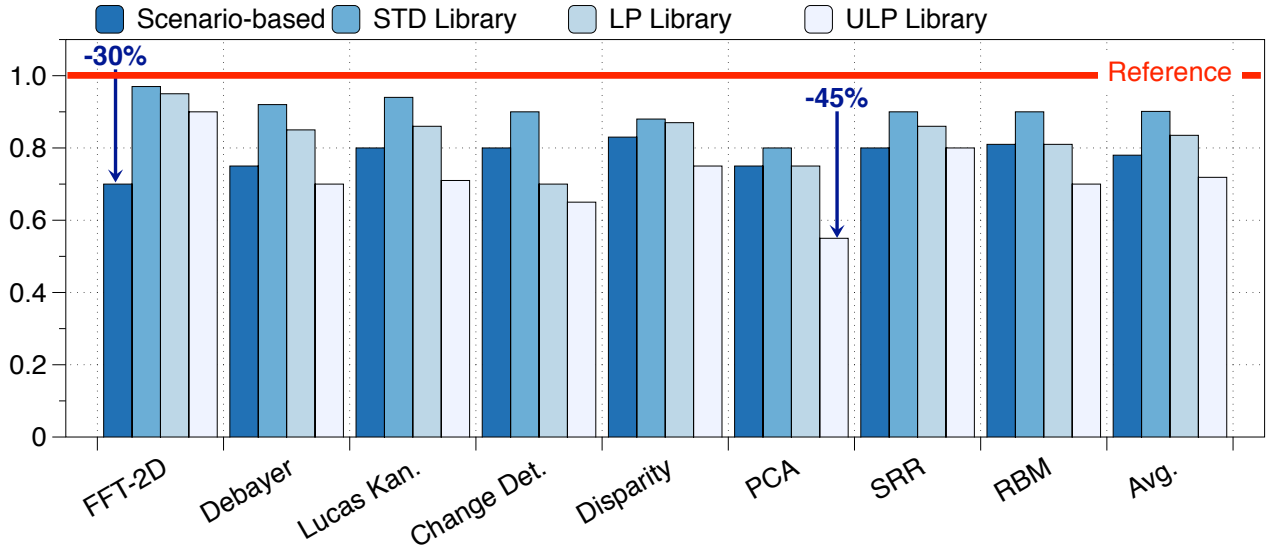
- Used to configure the SMC modules to generate the proper masks

Experimental Results

- We improved the design of eight accelerators
  - SystemC specification extended with DarkMem API
  - PLM generator extended with generation of the DarkMem units
- Industrial 32nm CMOS technology at 1GHz
  - Cadence C-to-Silicon for HLS
- Memory library with 18 dual-rail SRAMs
  - Customized to have different power-gating characteristics

Impact of Single Optimizations

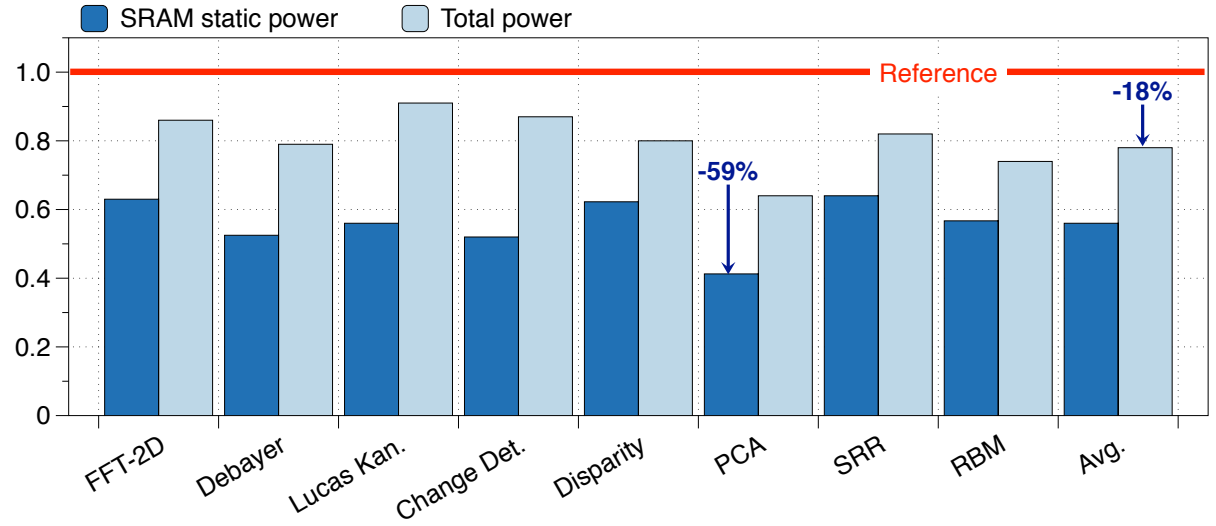
- **Reference** designs: with no optimizations



- Performance overhead is minimal (less than 1%)

Combined Results

- SRAM static power can be reduced up to 60%
  - In average, the total power is reduced by around 18%

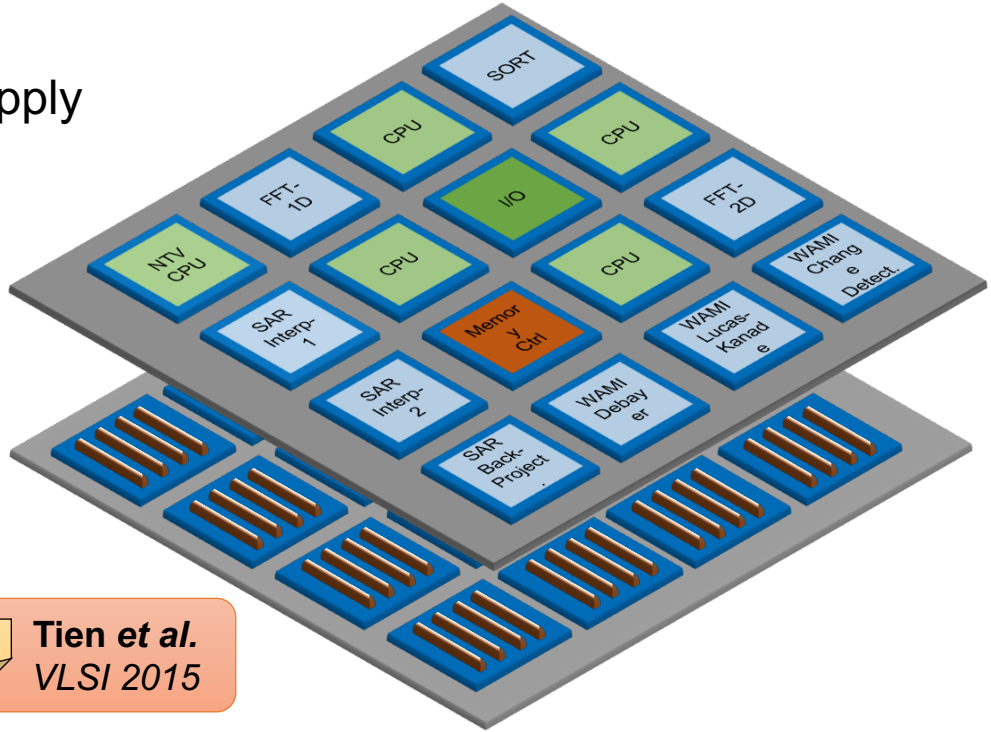


Fine-grained DVFS in Heterogeneous SoCs

Modern technology for **integrated voltage regulators** (IVRs) enables fine-grained DVFS

- Fast response (sub-µs) to dynamic supply voltage scaling
- Possibility to create NoC-based SoCs where each tile has its own voltage domain
  - 3D-Stacked Switched-Inductor Voltage Regulators

How many voltage domains?
How to aggregate accelerators?
How to change operating point?



Tien *et al.*
VLSI 2015

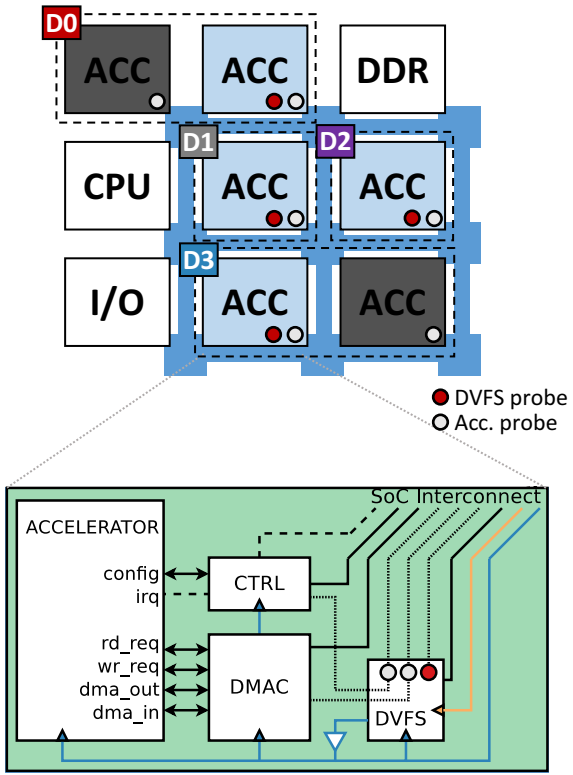
Enabling Pre-Silicon Evaluation of Fine-Grained DVFS

Design time modeling

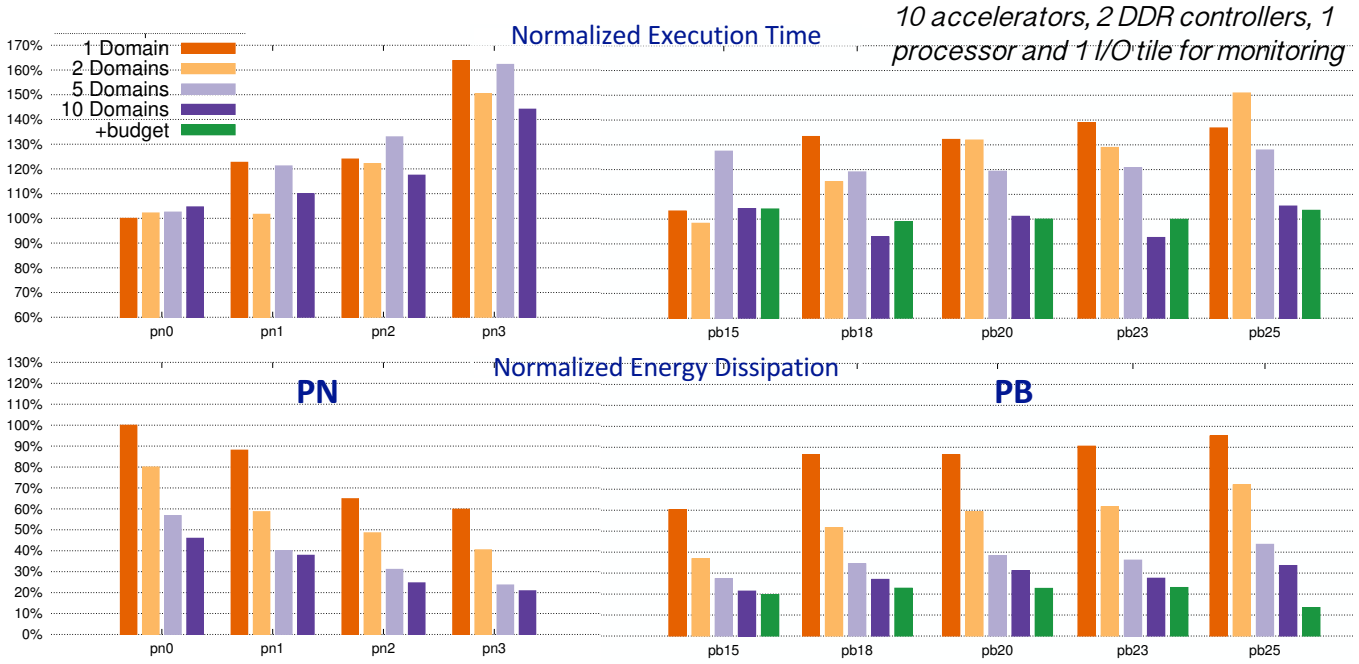
- Accurate power modeling of accelerators
- Actual frequency scaling implementation
  - Up to 12 independent clock regions dedicated to accelerators on each FPGA
  - Configurable transient (~10s cycles)
  - Configurable hardware policies and software supervisor

Runtime monitoring

- Distributed system and accelerator customized probes



Heterogeneous SoC with Fine-grained DVFS



Up to **85% energy savings** with a performance penalty of less than 10%

Conclusions

Specialized accelerators are key elements in SoC design

- High performance and energy efficiency
- (Quite) mature HLS tools can support the generation of complex systems

PLM customization is crucial to achieve high performance

- Enables more hardware parallelism
- High static/dynamic power consumption to be addressed with specific solutions

With **Mnemosyne** and **fine-grained power management**:

- We create complex heterogeneous SoCs with optimized accelerator memory subsystems (up to 55% of area savings, and 60% of power saving)
- We reduce energy consumption (>80%) with almost no performance overhead (<10%)

What's Next?

Latency-insensitive memories can relax the conflict-free requirements and enable additional memory optimizations

- Memory subsystem can be designed in parallel to accelerators
  - physically-aware optimizations
  - trading-off performance overhead and area savings

Analysis of **emerging memory technologies**

- How to *design* and *program* systems with a combination of different memory technologies?

References

- C. Pilato, F. Ferrandi, D. Sciuto, “A design methodology to implement memory accesses in High-Level Synthesis”, CODES+ISSS 2011
- E. Cota, P. Mantovani, G. Di Guglielmo, L.P. Carloni, “An Analysis of Accelerator Coupling in Heterogeneous Architectures”, DAC 2015
- P. Mantovani, G. Di Guglielmo, L.P. Carloni, “High-level synthesis of accelerators in embedded scalable platforms”, ASPDAC 2016
- R. Nane, V. M. Sima, C. Pilato, J. Choi, B. Fort, A. Canis, Y. T. Chen, H. Hsiao, S. Brown, F. Ferrandi, J. Anderson, K. Bertels, “A Survey and Evaluation of FPGA High-Level Synthesis Tools”, TCAD 2016
- P. Mantovani, E. Cota, C. Pilato, G. Di Guglielmo, L.P. Carloni, “Handling large data sets for high-performance embedded applications in heterogeneous systems-on-chip”, CASES 2016
- P. Mantovani, E. Cota, K. Tien, C. Pilato, G. Di Guglielmo, K.L. Shepard, L.P. Carloni, “An FPGA-based Infrastructure for Fine-grained DVFS Analysis in High-performance Embedded Systems”, DAC 2016
- C. Pilato, P. Mantovani, G. Di Guglielmo, L.P. Carloni, “System-Level Optimization of Accelerator Local Memory for Heterogeneous Systems-on-Chip”, TCAD 2017
- C. Pilato, L.P. Carloni, “DarkMem: Fine-Grained Power Management of Local Memories for Accelerators in Embedded Systems”, ASPDAC 2018