There is no **Press & Go** procedure to refactor SW to HW,

…or there is?

introducing

Dynamic Memory Management in Vivado-HLS for Scalable Many-Accelerator Architectures

Dionysios Diamantopoulos, Sotirios Xydis, Kostas Siozios and Dimitrios Soudris

School of Electrical and Computer Engineering, National Technical University of Athens, Greece

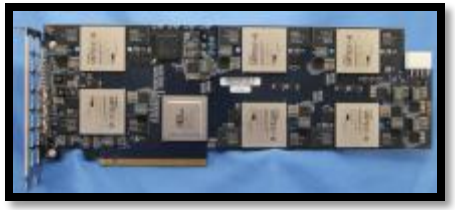
ARC 2015, 13 - 17 April 2015, Bochum, Germany

A promising FPGA-coupled accelerators' world!

From research wise vision to industry adoption

Microsoft Catapult

Bing Search Acceleration
2x throughput, 30% of the cost



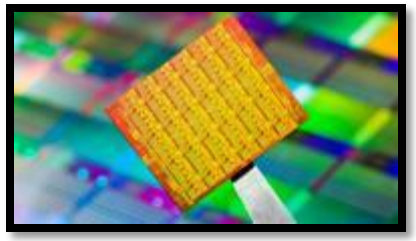
Search Engine

Cadence Protium

EDA Simulation

EDA Simulation Acceleration
design time reduction by up to 70%.

Intel unveils new Xeon+FPGA chip



Server Workload

Workload-aware customization
20x performance boost

HPC

Maxeler MPC-C Series

Large-scale risk analysis
284x speedup

Convey Computer

Graph analytics

Graph traversal
Breadth-First Search
12x on Graph500

Many-accelerator Systems on FPGAs

A promising FPGA-coupled accelerators' world!

From research wise vision to industry adoption

Once-in-a-lifetime opportunity for architects:
Beyond general-purpose, specialization and acceleration go mainstream.

Limitations to FPGA many-accelerators (1/2)

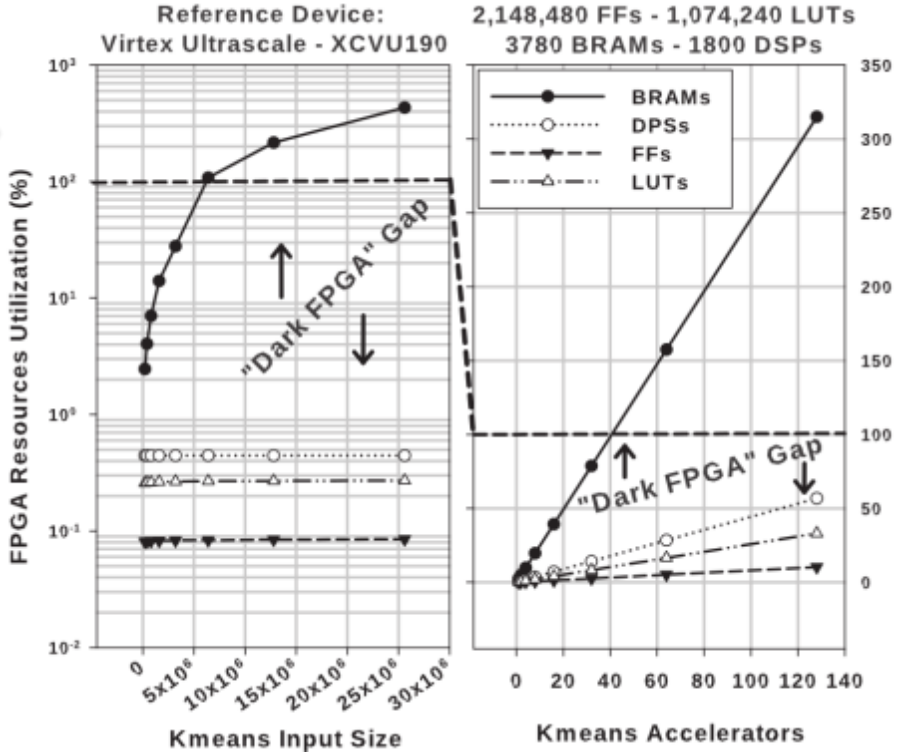
The expected-one

PC-FPGA Bandwidth: Adopted Technology Solutions i) Advanced multi-lanes interconnection (i.e. InfiniBand), ii) Photonics.

The surprising-one

Memory resources: Not yet an efficient solution. Memory subsystem is not only a performance bottleneck but it rather forms the **main limiting factor** regarding the **scalability** of many-accelerator architectures for FPGA devices.

A real-world case study
Kmeans on Ultrascale

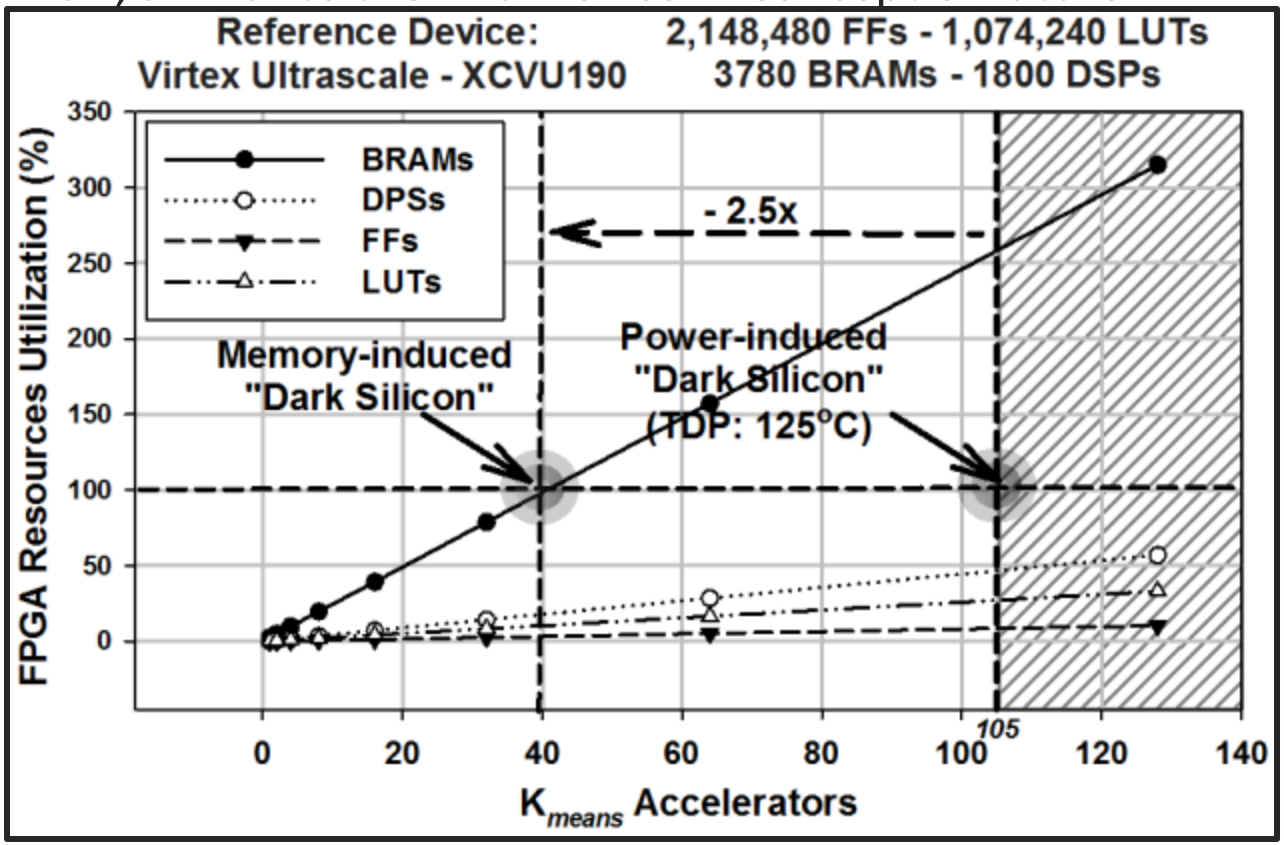


Source of the memory scalability bottleneck

Modern FPGA design tools allow only **static memory allocation**, which dictates the reservation of the **maximum memory** requirements, for the **entire execution window**.

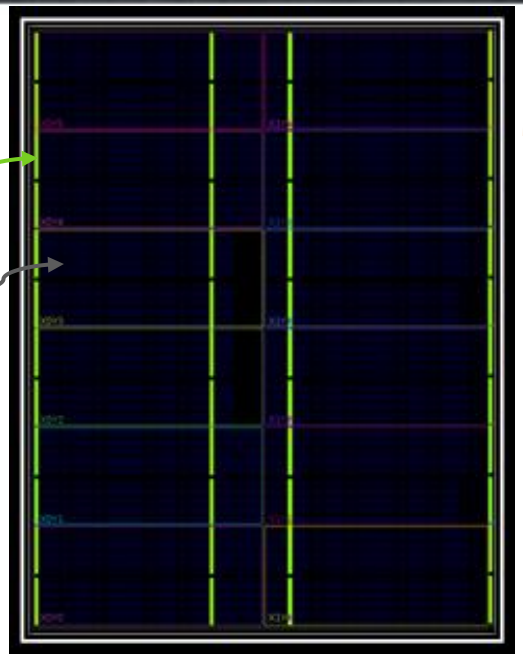
Limitations to FPGA many-accelerators (2/2)

- Rapid starvation of the available on-chip memory leads in severe resource under-utilization of the FPGA, similar to the "Dark Silicon" concept of future many-core chips
- Modern FPGA CAD tools (both at the RTL or HLS-level) allow only static memory allocation, which dictates the reservation of the maximum memory that an accelerator needs, for the entire execution window



- Considering an ambient temperature of 50°C, the power-induced "Dark Silicon" manifests itself with an allocation scenario of 105 accelerators consuming around 20 Watts while memory induced "Dark Silicon" poses a stricter constraint in accelerators' count, i.e. up to 2.5 × less accelerators

Why this matters ?

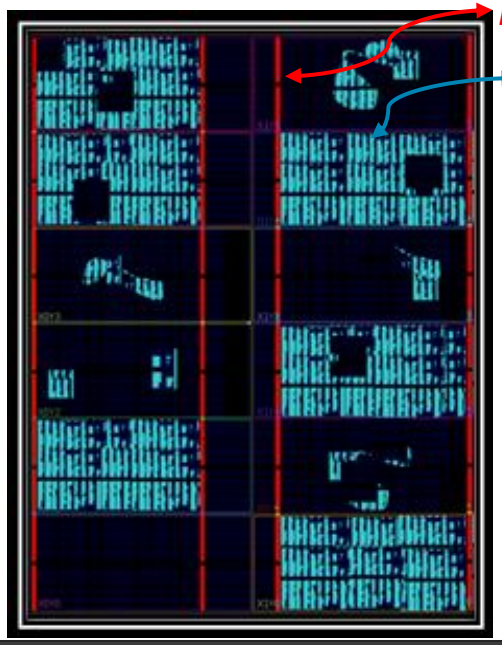


Today

5 x Kmeans

100% Memory utilization

32% Logic utilization



Our Strategy

13 x Kmeans

100% Memory utilization

96% Logic utilization

What do we achieve (21.4 avg. throughput) by alleviating memory starvation

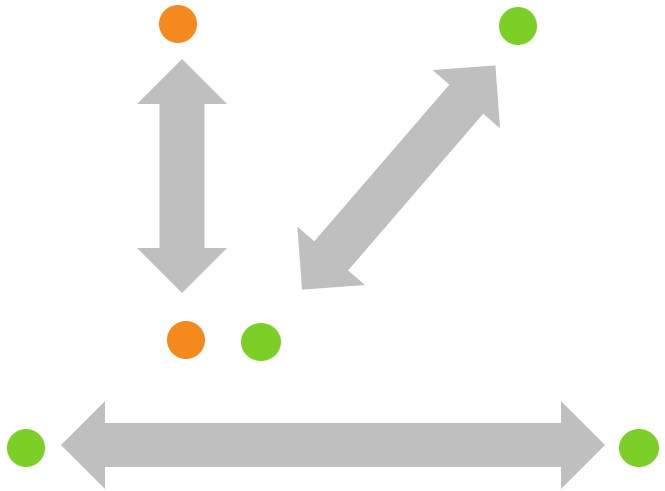
What do we lose (30x underutilization) if we don't resolve the memory scalability bottleneck

Our Contribution

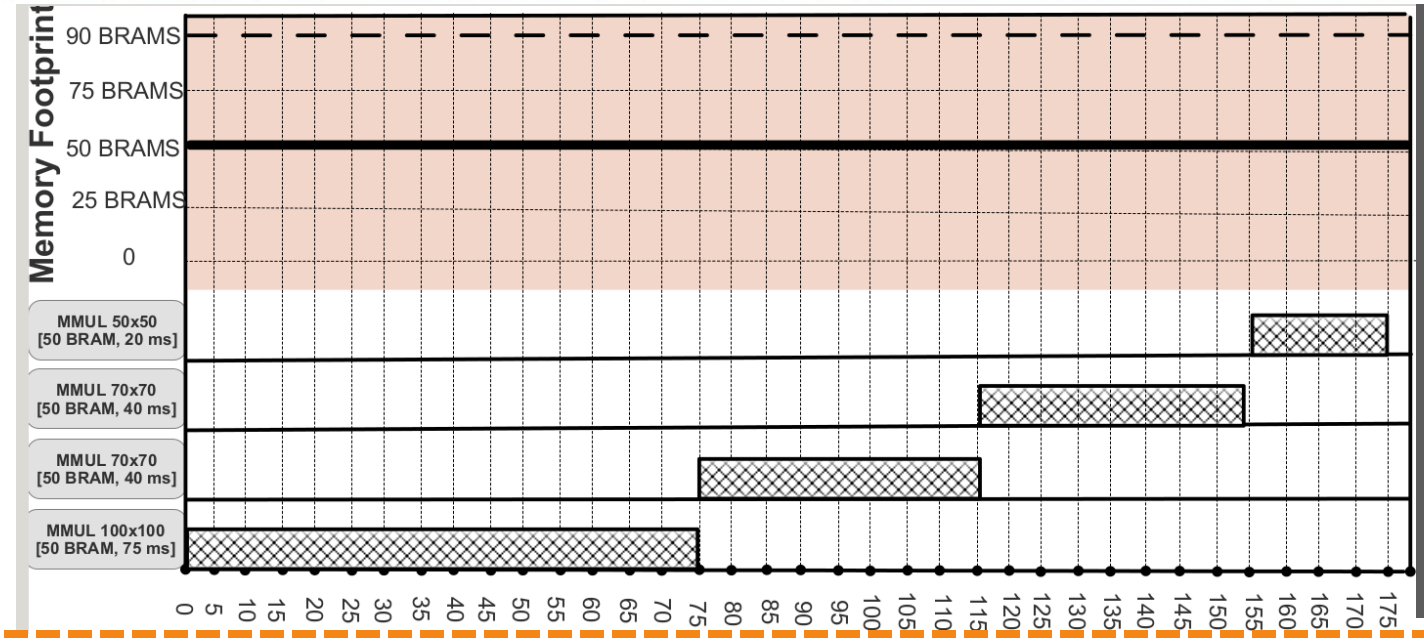
The novel DMM-HLS framework/API providing:

- **Synthesizable DMM Architecture for Vivado-HLS**
  extends typical HLS with DMM mechanisms
- **Design Methodology & Tool-flow**
  provides API function calls which enable source-to-source modifications that transform the statically allocated code to dynamic one, similar to malloc/free API of glibc
- **System-level Architecture Exploration**
  Early exploration of solutions that trade-off Quality of Report (QoR)

DMM-HLS for Memory-induced FPGAs

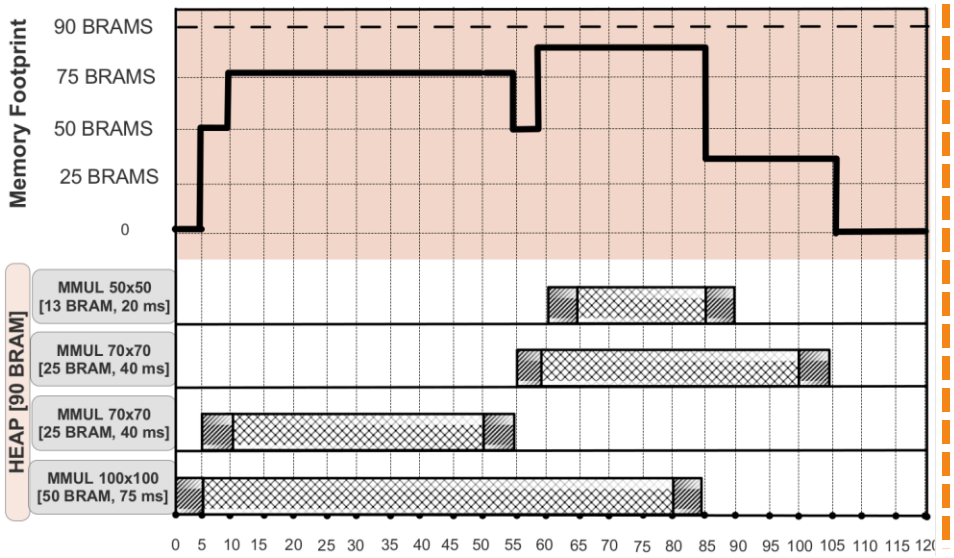


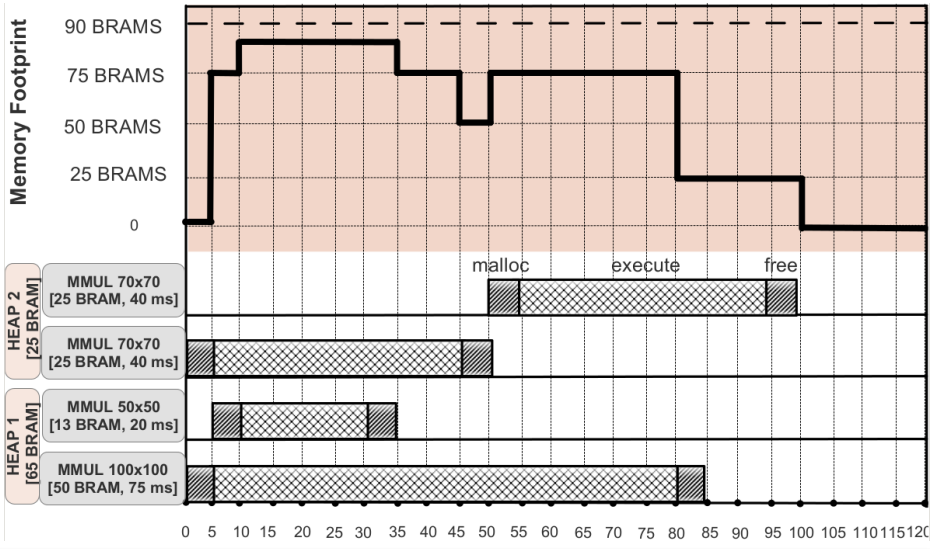
Instead of how-to: An explanatory test-case



Static : 175 ms
DMM-1 : 105 ms
DMM-2 : 100 ms

Accelerator parallelism and overlapping, for the execution of 4 MMUL applications with differing workload characteristics onto a FPGA with maximum 90 BRAMs – St./DMM1-2.





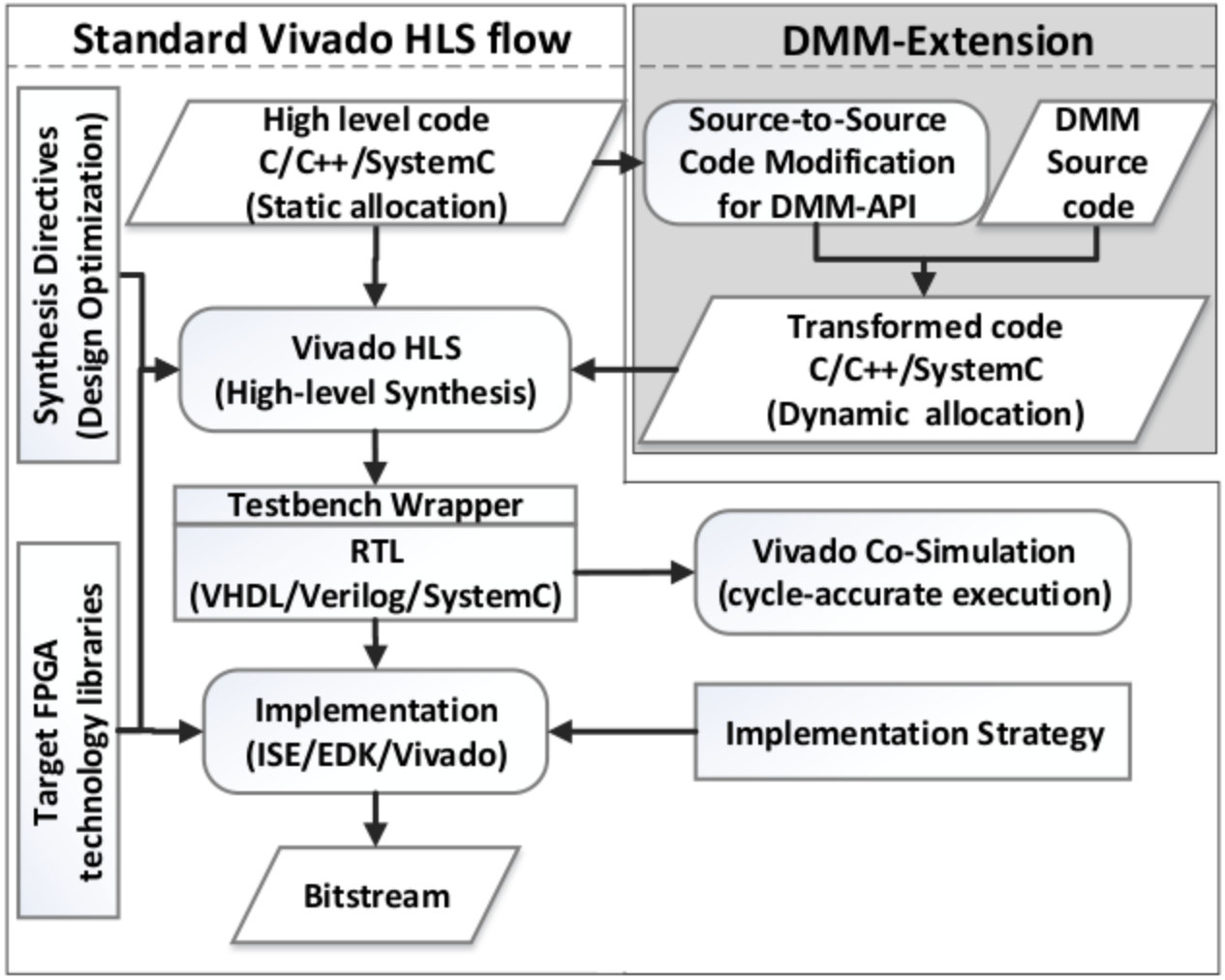
Design Methodology

Two-steps design methodology

- The generation of high performance hardware accelerators
- The integration of the generated accelerators together with a reference SoC architecture.

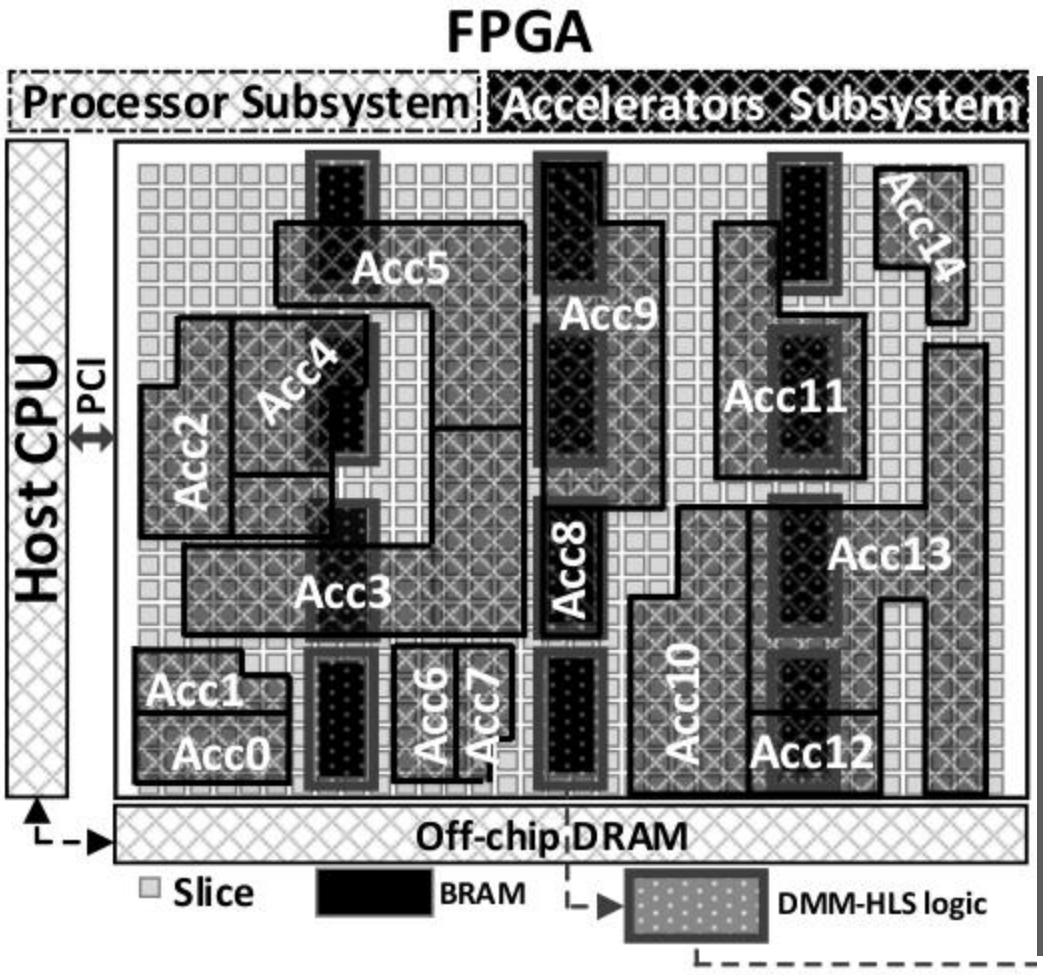
Toolflow

- *Xilinx Vivado HLS* -> design of a AXI slave-based accelerators
- *Xilinx XPS* -> top-down SoC design



- The original code is transformed from statically allocated to dynamically allocated using specific function calls from the proposed DMM-HLS API

Design Methodology



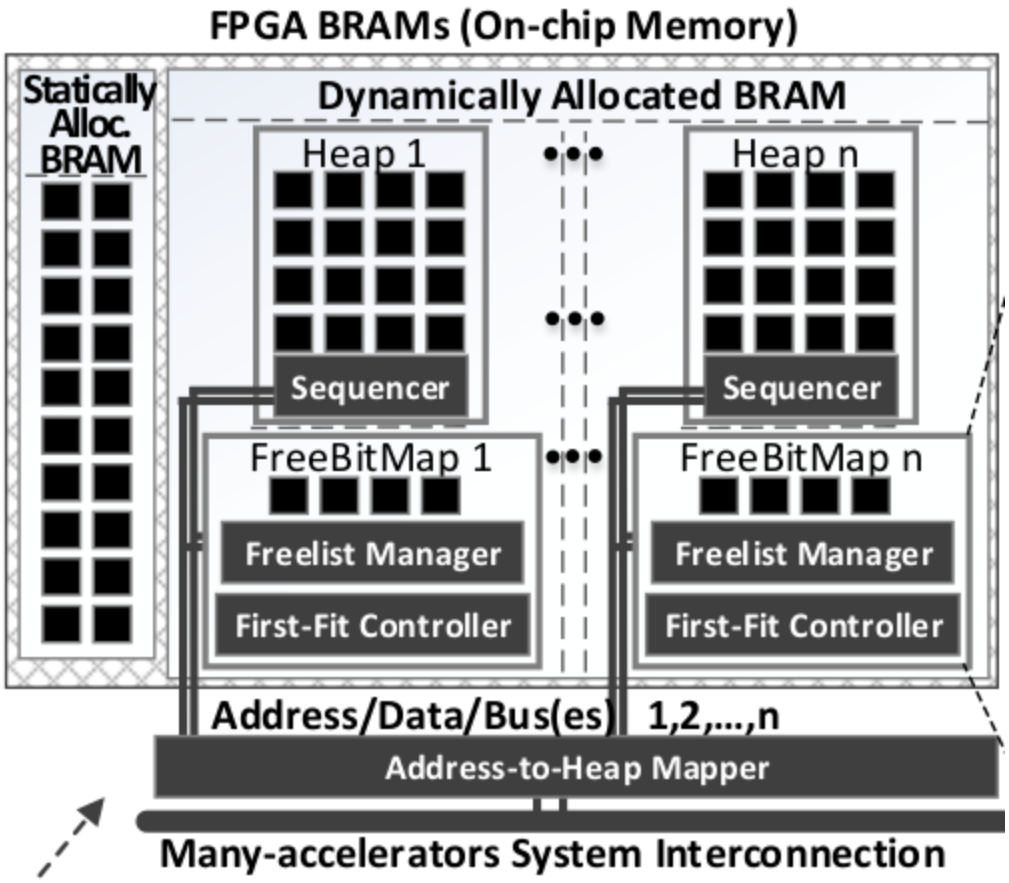
The accelerators are modeled in high-level language and synthesized using Vivado-HLS. The on-chip BRAMs are considered as a shared-memory communication link between processor and accelerators datapaths.

Moving from static to dynamic allocation using an aggressive approach could lead to the allocation of all BRAMs under the same memory module so that all allocation/deallocation requests occurs on this unique module. This implies serialization bottleneck in case of hundreds of accelerators, thus memory level parallelism is required.

Architecture template for memory efficient many accelerator FPGA-based systems

- Processor subsystem: executing the application control flow
- Accelerators subsystem: holding the computationally intensive kernels

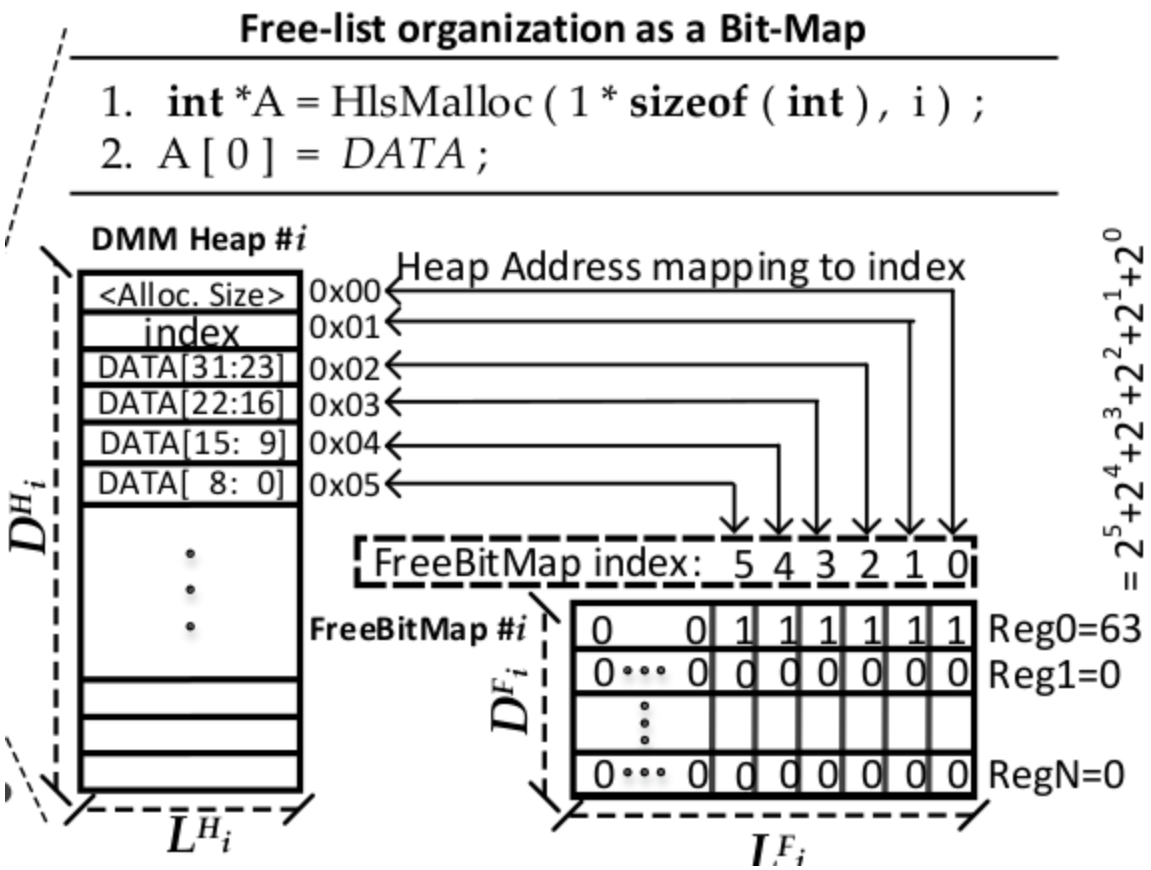
Design Methodology



Each heap implements its own allocator consisting of two major hardware components, i) the free-list memory structure holding the freed and allocated memory blocks and ii) the fit allocation algorithm that searches over the free-list and allocates memory in a first fit manner. The maximum number of heaps controls the supported memory level parallelism of the dynamically allocated data.

- DMM-HLS supports parallel memory access paths, by grouping BRAM modules into memory banks (heaps).
- More than one accelerators can be bound to a specific memory heap for allocating data.

Design Methodology



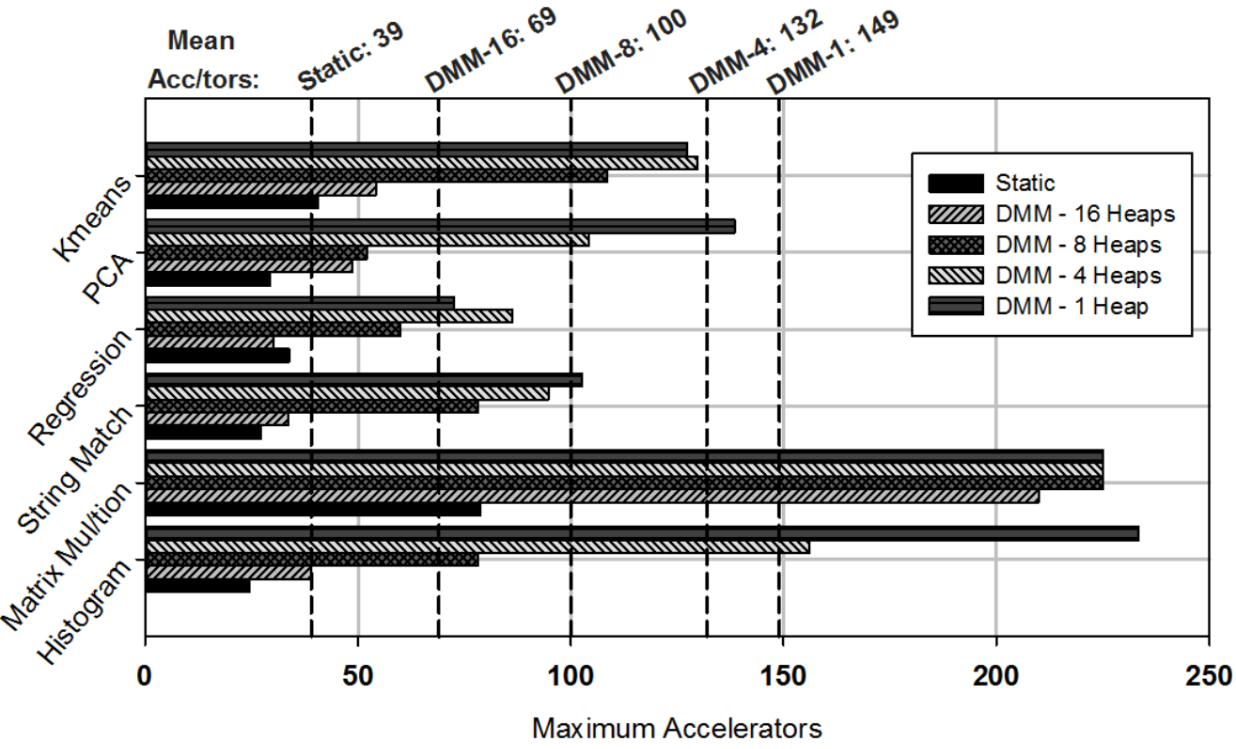
Each *heap-i* is highly parametric on a number of design options :

(i) the heap depth
(ii) the heap word length,
(iii) the allocation alignment
(iv) the meta-data header size

In a similar manner we define the free depth and free word length for the FreeBitMap memory.

- The proposed DM allocator for HLS supports arbitrary size allocation/ deallocation, i.e. enables allocation on the same heap of any simple data type (integers, floats, doubles etc.), as well as more complex data types (structs of the same or combined simple data types and 1-D arrays).

Experimental Results 1/3 : Accelerators density



The proposed DMM-HLS framework delivers many-accelerator architectures with 3.8× more accelerators in average, (vs static). On some kernels the gains may be up to 9.7×, i.e. Histogram application.
The high gains of DMM-HLS on accelerator's density come from the usage of one single heap (DMM-1).

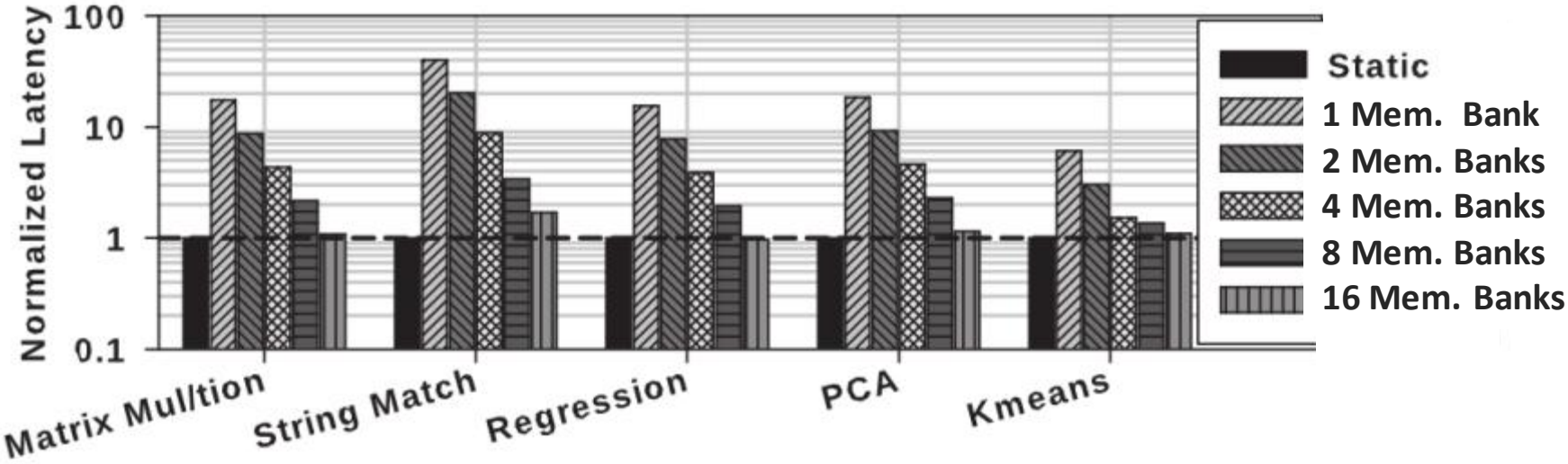
DMM-1 configuration has the least possible overhead regarding the resources consumed by the DM manager. As long as configurations with more heaps are adopted (e.g. DMM-4 etc), the extra resources needed to implement the corresponding allocators decrease the maximum number of accelerators, i.e. the instantiation of 16 heaps delivers an average gain of 1.7×.

Per Accelerator Analysis:
Tunable Performance Overheads

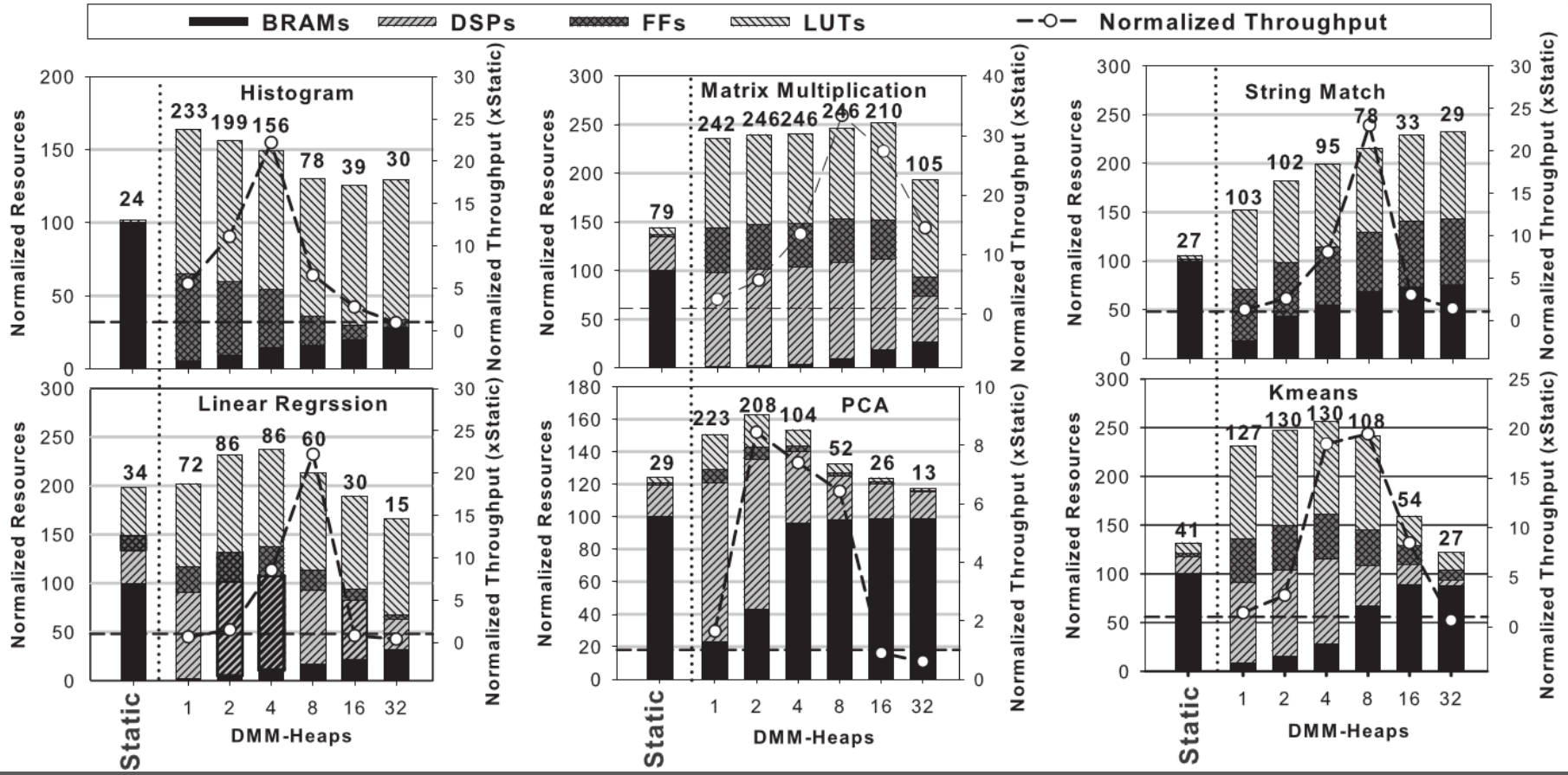
- Tunable memory level parallelism can be used to minimize performance overhead per accelerator.

Memory Banks	Av. Latency Overhead vs Vivado HLS
1	19 x
2	10 x
4	4.7 x
8	2.3 x
16	1.2 x

The average overhead for all accelerators is 19.9× when only one heap is employed.
It drops to 10×, 4.7×, 2.3× and 1.2× as long as the heaps are doubled.
The increase of heaps allows higher memory level parallelism to be achieved, since less accelerators are sharing the same heap.



Experimental Results 3/3: System Throughput



- We measured an average throughput increase of 21.4 × with private memory initialization (3.1 × with shared memory) over the static allocation of the conventional Vivado-HLS.
- Area overheads is due to the heap allocator modules and the extended interface of the accelerators. For single heap implementations: 0.3% FFs, 1.2% LUTs, which scales up to +10.7% FFs and +55.2% LUTs for the case of 32 heaps.

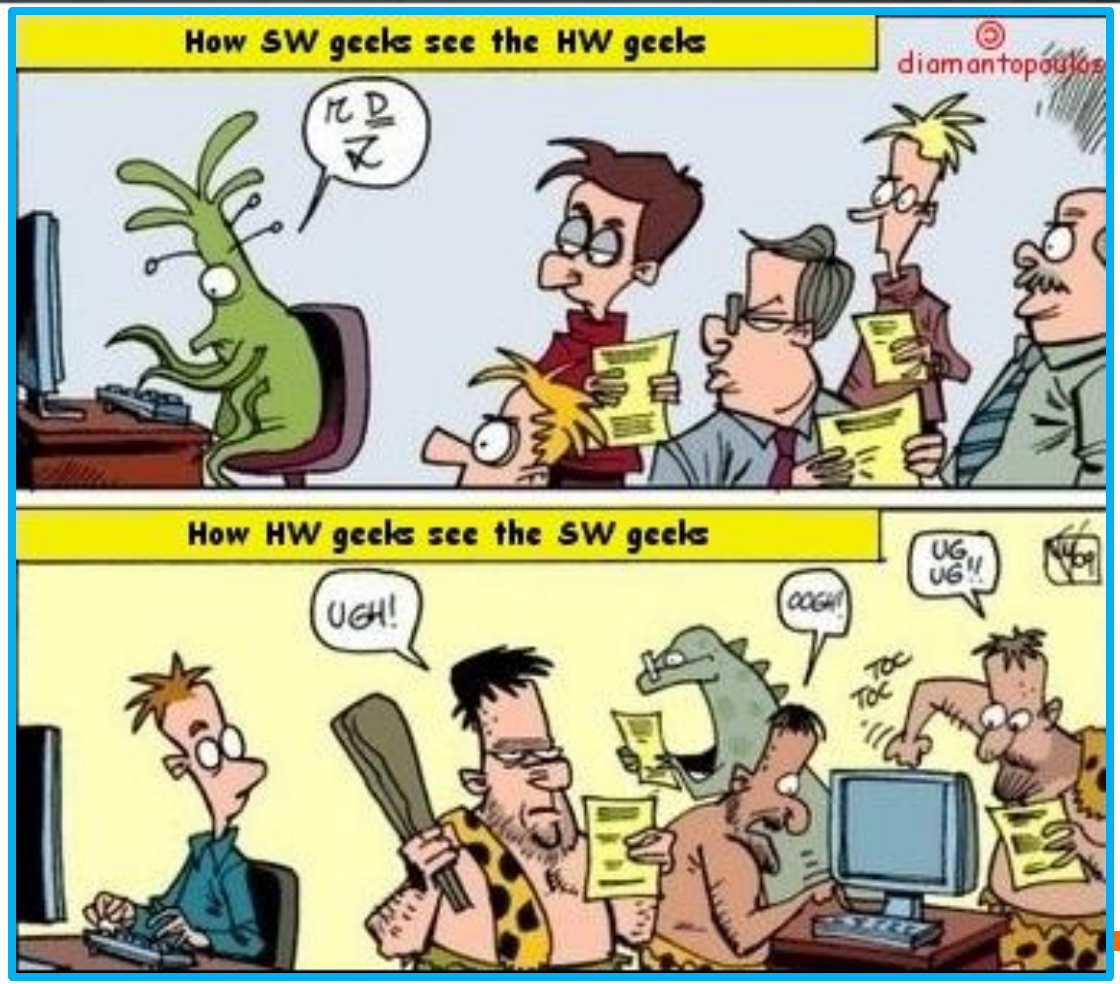
Future directions

- Customized heaps build-up
- Extend DMM-HLS with memory paging capabilities for external DRAM/FLASH – test it with real Big Data
- Make DMM-HLS open to HLS community

Conclusions

- A ESL **methodology** and the corresponding **tool flow** to build efficient many-accelerator architectures for FPGAs has been presented
- Our framework already tested on 6 critical kernels (Phoenix suite targeting **HPC**).
- It succeeded significant increase in FPGA's accelerators density (**3.8× more accelerators**).
- DMM-HLS framework aided the instantiation of many-accelerator systems that exhibit an **average throughput increase of 21.4×** over the static allocation obtained by the conventional Vivado-HLS flow.

Thank you!



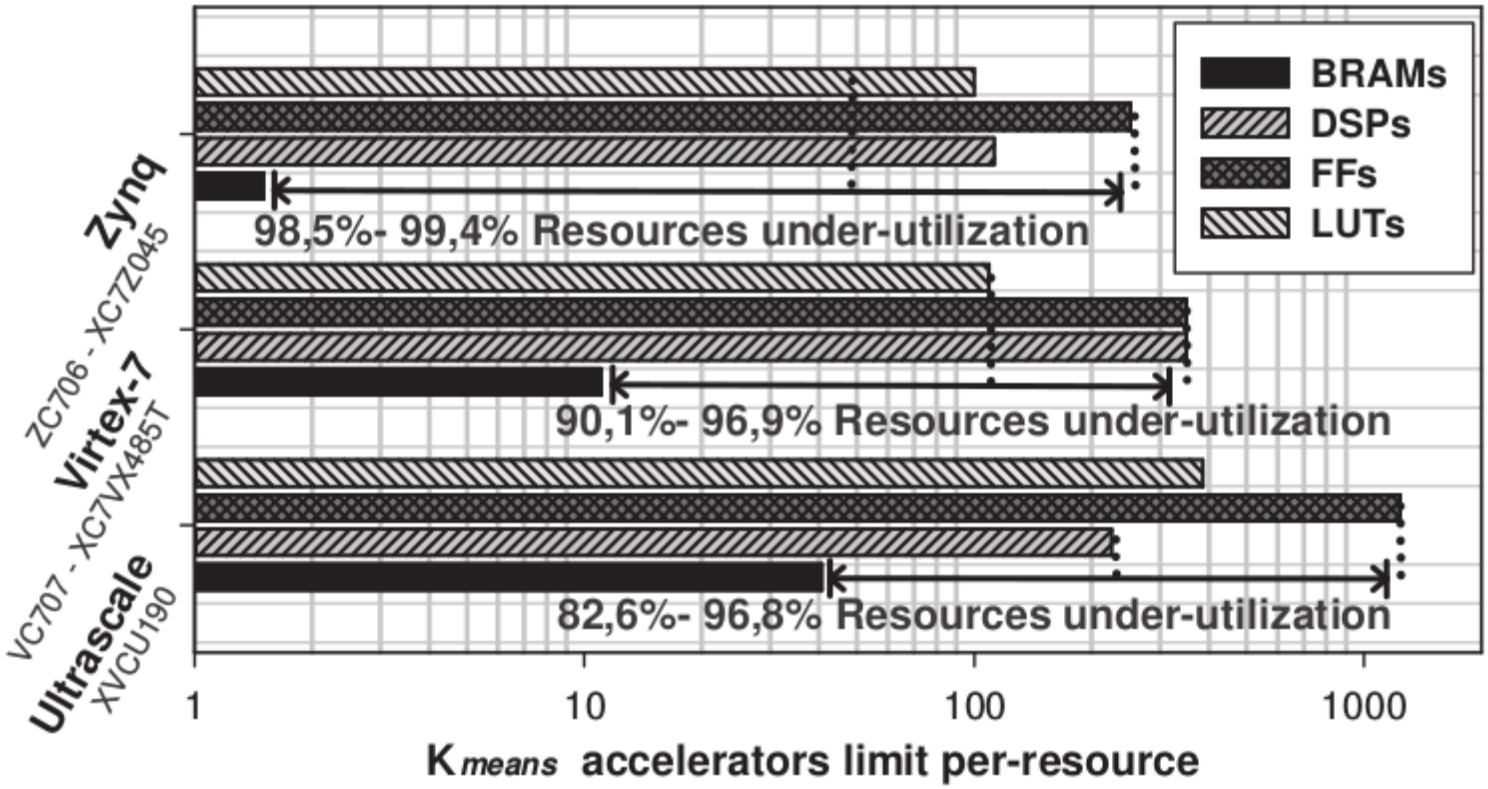
more info at
diamantd@microlab.ntua.gr

Backup Slides

more than More slides…

Backup 1/2: Device-universal under-utilization

- The on-chip memory (BRAM type) is the resource type that starves faster
- The size of the on-chip memory has a significant impact on the maximum number of allocated accelerators



- The memory-induced resource under-utilization ranges between 98.5% - 99.4%, 90.1% - 96.9% and 82.6% - 96.8% for the Zynq, Virtex-7 and Ultrascale FPGA devices, respectively

Backup 2/2: Employed Applications

APPLICATIONS CHARACTERIZATION

Application Domain	Kernel	Description	Parameters
Image Processing	Histogram	Determine frequency of RGB channels in image.	M_{size} = 640×480 pixels
Scientific Computing	Matrix Multiplication	Dense integer matrix multiplication.	M_{size} = 100×100
Enterprise Computing	String Match	Search file with keys for an encrypted word	$N_{file-keys}$ = 307,200, M_{words} = 4
Artificial Intelligence	Linear Regression	Compute the best fit line for a set of points.	N_{points} = 100,000
Artificial Intelligence	PCA	Principal components analysis on a matrix.	M_{size} = 250×250
Artificial Intelligence	K_{means}	Iterative clustering algorithm to classify n-D data points into groups.	N_{points} = 20,000, $P_{clusters}$ = 10, $n_{dimensions}$ = 3