PARMA 2020: 11th Workshop on Parallel Programming and
Run-Time ManagementTechniques for Many-core Architectures
DITAM 2020: 9th Workshop on Design Tools and Architectures
for Multi-Core EmbeddedComputing Platforms
PARMA-DITAM 2020

An OpenMP Parallel Genetic Algorithm for Design Space Exploration of Heterogeneous Multi-processor Embedded Systems

Author: Vittoriano Muttillo, Paolo Giammatteo, Giuseppe Fiorilli and Luigi Pomante

{vittoriano.muttillo, paolo.giammatteo, luigi.pomante}@univaq.it, {giuseppe.fiorilli}@student.univaq.it

University of L'Aquila
*Center of Excellence **DEWS***
*Department of Information Engineering, Computer Science and Mathematics (**DISIM**)*

Outline

1. Introduction
2. Design Space Exploration Context
3. Research Questions
4. Proposed Parallel Genetic Algorithm
5. Methodology and Scenarios
6. Experimental Results
7. Conclusion And Future Work

1. Introduction

Introduction

- The most critical issues into an HW/SW Co-Design methodology is always related to Design Space Exploration (DSE) activities.
- DSE is related to the approach, whether automated or not, used in order to find the best HW/SW partitioning and mapping for the final system implementation.
- These approaches usually rely on optimization problems, more than one objective function that can be considered (i.e., minimize cost, power consumption, maximize performance, throughput, etc.)
- A classical approach to solve an Multi-objective optimization problems (MOOP) is the Weighted Sum Method (WSM), which assigns a weight to each objective function

2.
Design Space Exploration Context

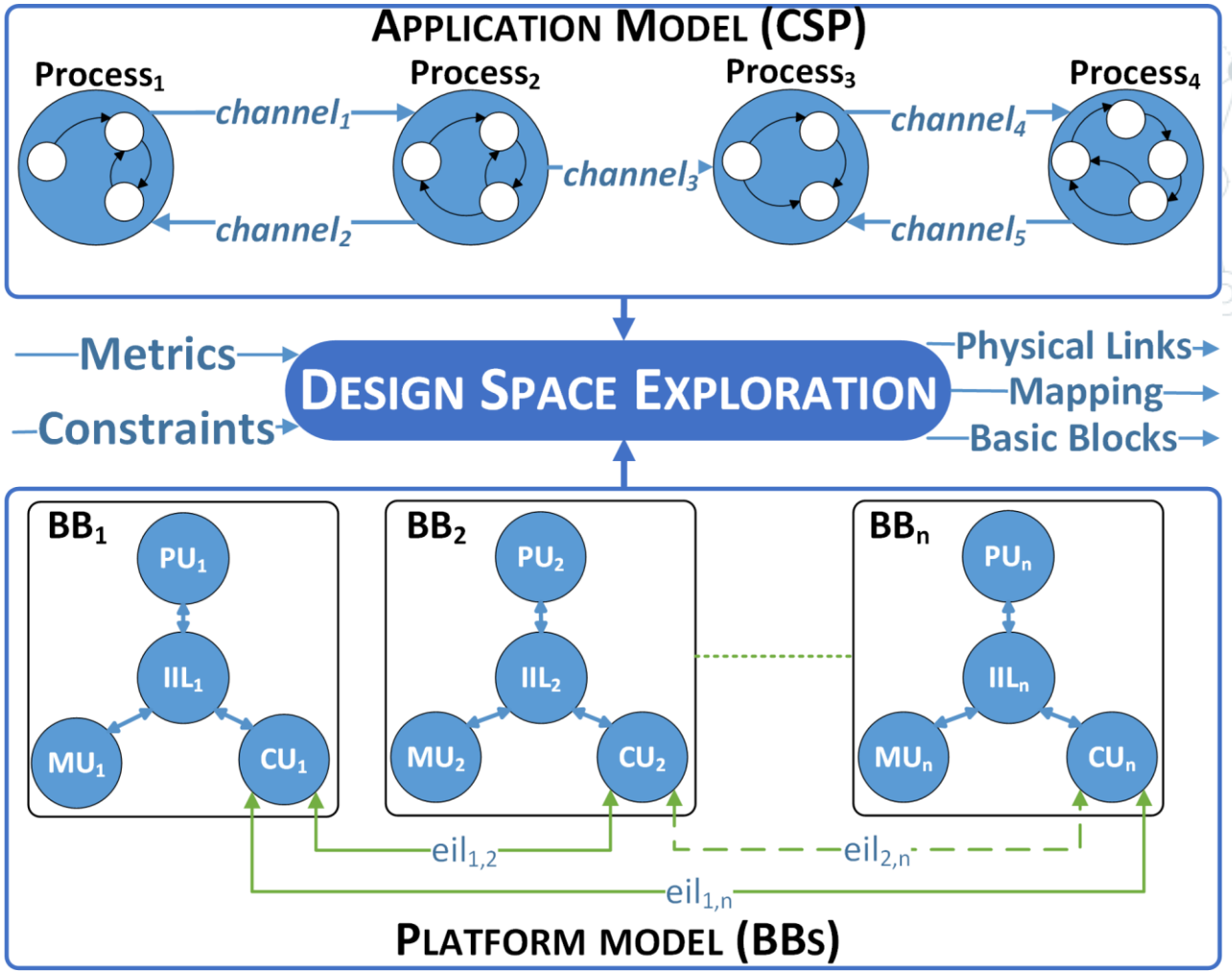
Design Space Exploration Context

INPUT:

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

OUTPUT:

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



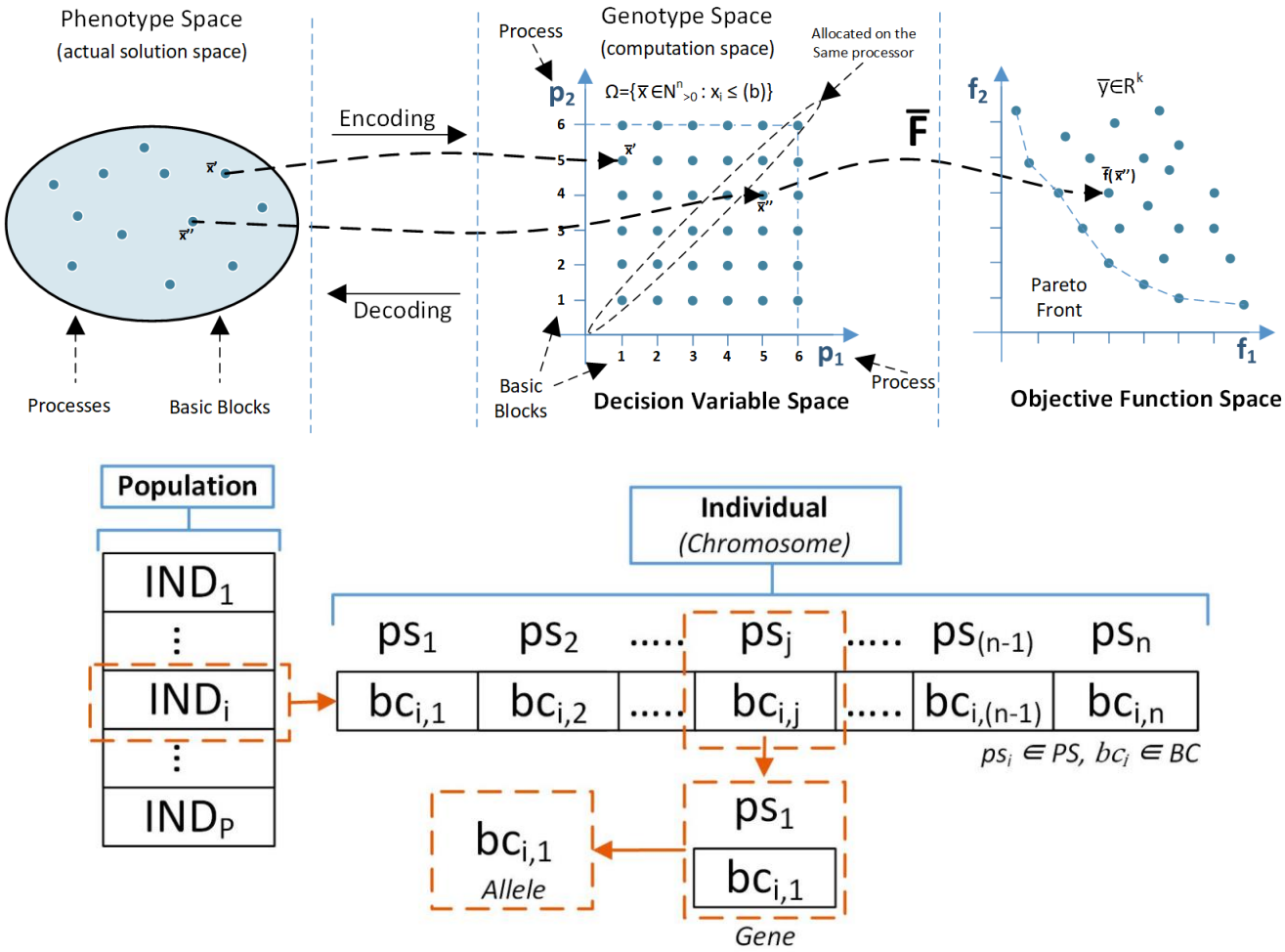
Design Space Exploration Context

INPUT:

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

OUTPUT:

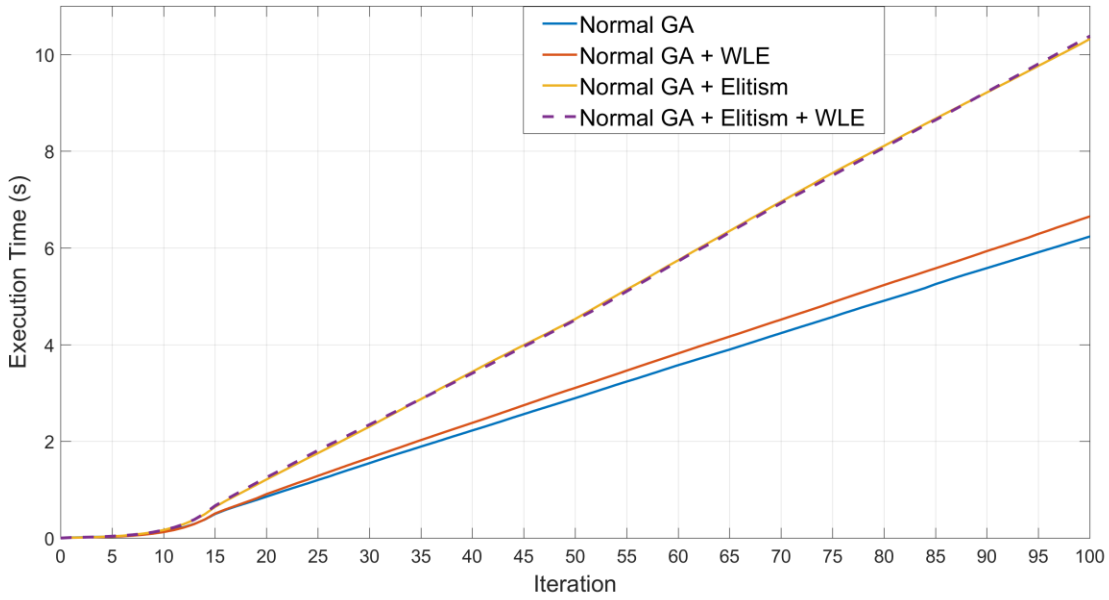
- **Physical Links:** Possible optimal links and topology.
- **Mapping:** Process to BBs.
- **Basic Blocks:** Processors, architecture and number of cores.



3. Research Questions

Research Questions

- **RQ1**: How is possible to reduce automatic DSE-GA execution time without loss of accuracy and diversity?
- **RQ2**: Which paradigm can be used in order to guarantee bounded timing behavior?
- **RQ3**: In which manner the automatic DSE configuration drives the choice of the possible algorithm implementations?



Research Questions

- **RQ1**: How is possible to reduce automatic DSE-GA execution time without loss of accuracy and diversity?
- **RQ2**: Which paradigm can be used in order to guarantee bounded timing behavior?
- **RQ3**: In which manner the automatic DSE configuration drives the choice of the possible algorithm implementations?

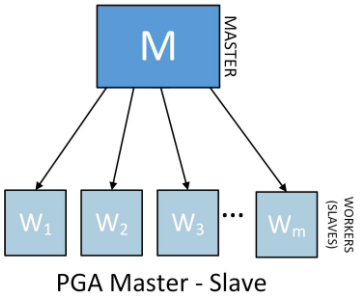
PARALLEL GENETIC ALGORITHM (PGA)

4.

Proposed Parallel Genetic Algorithm

Proposed Parallel Genetic Algorithm

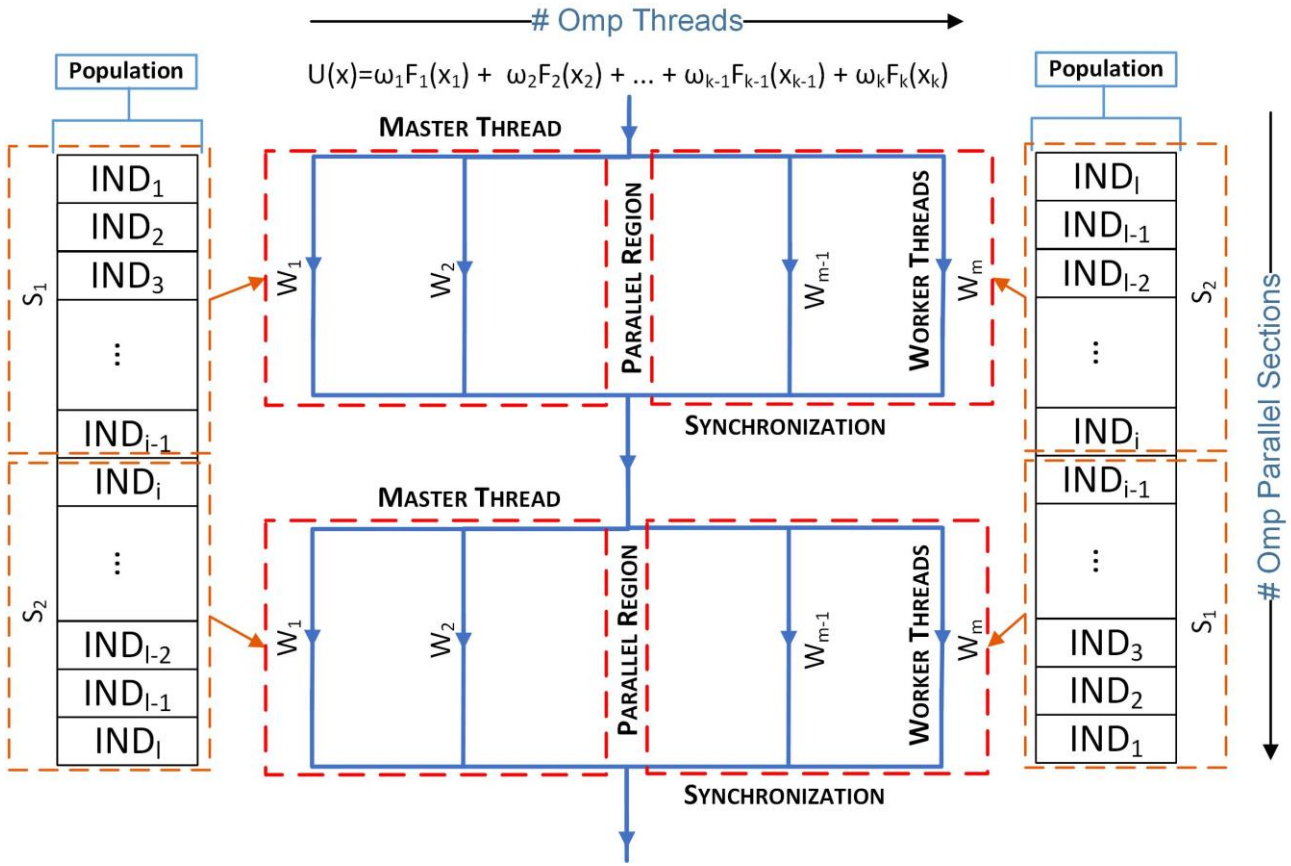
Global parallelization has been implemented, with a master-slave PGA approach.



PGA Master - Slave

3 types of PGA implementation:

- **Type 1**: Parallel evaluation of objective functions for each individual (evaluates objective functions on different threads);
- **Type 2**: Parallel evaluation of utility function (split the population into subsets);
- **Type 3**: Hybrid approach, combining the previous two approaches.



5.
Methodology and Scenarios

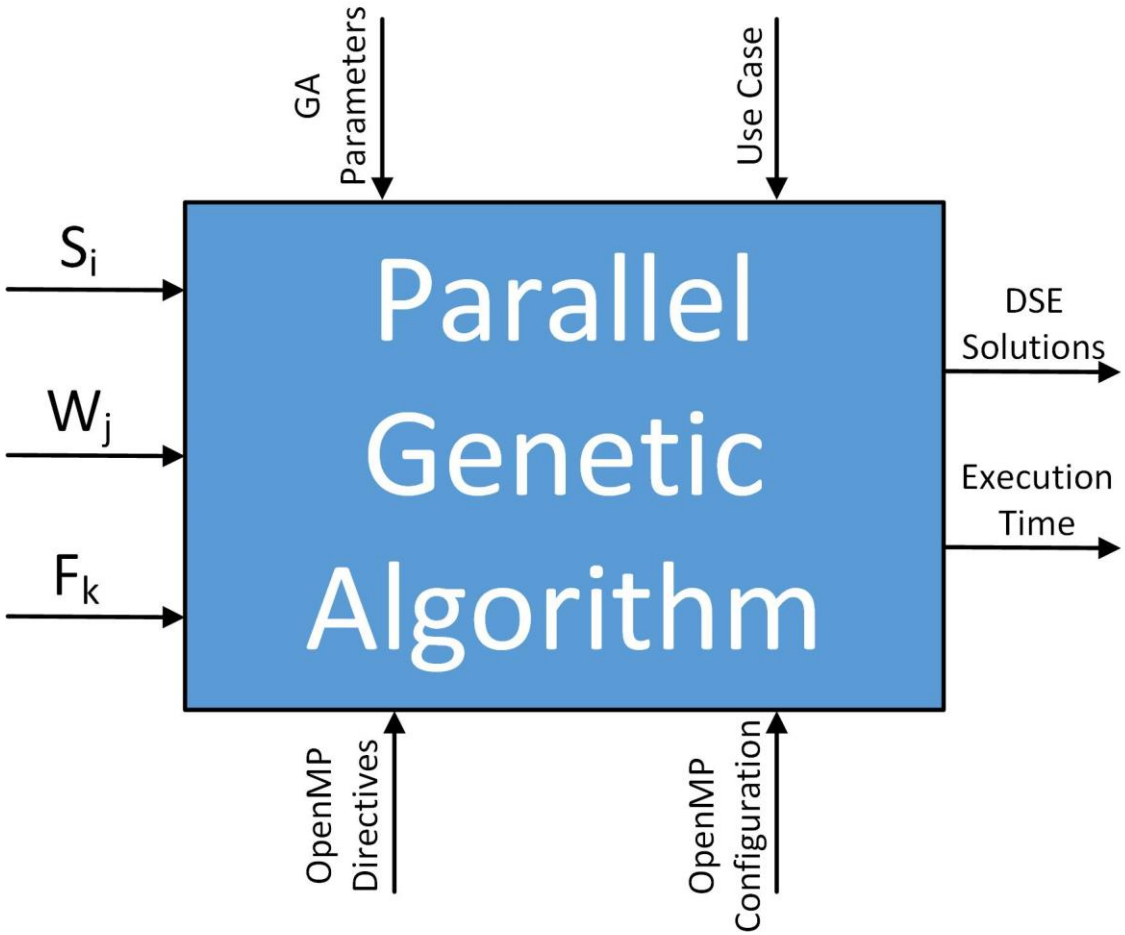
Methodology and Scenarios

INPUT:

- **S_i:** sub-population size
- **W_j:** number of workers/threads considered
- **F_k:** number of objective functions evaluated in parallel
- **GA Parameters:** mutation, crossover, selection, etc.
- **Use Cases**
- **OpenMP Directive**
- **OpenMP Configuration**

OUTPUT:

- **DSE solutions**
- **Execution Time**



Methodology and Scenarios

Different tests and configurations:

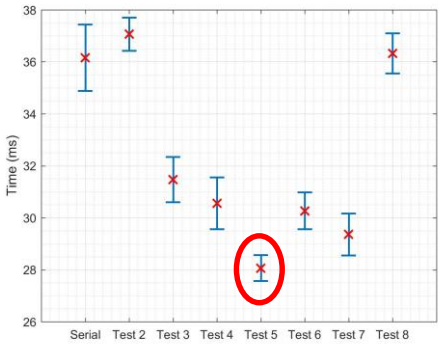
1. Serial;
2. Type 1 with parallel evaluation of objective functions for each individuals with 2 threads;
3. Type 2 with parallel evaluation of utility function using subpopulation sets and the *omp parallel for* with 2 threads;
4. Type 2 with parallel evaluation of utility function using subpopulation sets and the *omp parallel for* with 4 threads;
5. Type 2 with parallel evaluation of utility function using sub-population sets and the *omp parallel for* with 2 threads, schedule guided;
6. Type 2 with parallel evaluation of utility function using subpopulation sets and *omp parallel sections* with 2 sections and 2 threads;
7. Type 2 with parallel evaluation of utility function using subpopulation sets and omp parallel sections with 4 sections and 4 threads;
8. Type 3 (i.e., hybrid approach) with parallel evaluation of objective functions using sub-population and omp parallel sections with 2 parallel flows and 2 sections

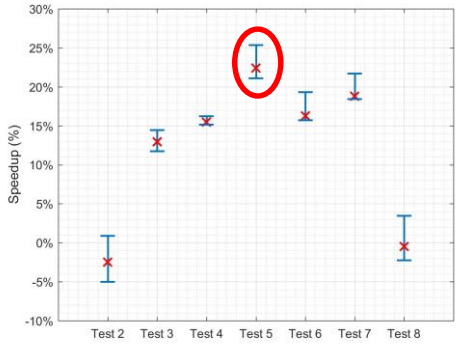
6. Experimental Results

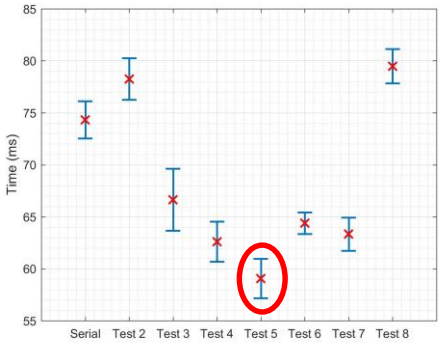
Experimental Results

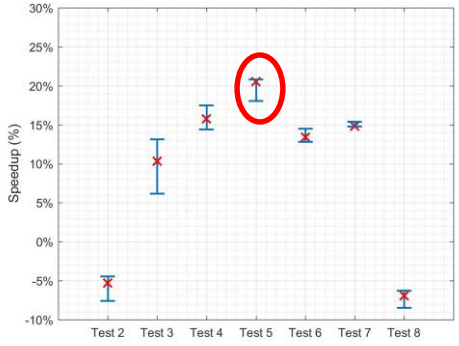
Parallelism performance table

	500000 individuals				1000000 individuals			
Impl.	Best Case (s)	Speed-up	Worst Case (s)	Speed-up	Best Case (s)	Speed-up	Worst Case (s)	Speed-up
1.	34	-	38	-	71.5	-	76.5	-
2.	36	-5.56%	38	0	74.7	-4.28%	82.3	-7.05%
3.	30,1	11,47%	32,6	14,21%	62,1	13,15%	71,8	6,14%
4.	29	14,7%	32	15,79%	59	17,48%	65,5	14,38%
5.	27,4	19,41%	29	23,68%	57	20,28%	62,7	18,03%
6.	29,5	13,23%	31,6	16,84%	62,7	12,31%	65,8	13,99%
7.	28,6	15,88%	31	18,42%	60,5	15,38%	65,2	14,77%
8.	35,5	-4,22%	37,5	1,32%	76	-5,92%	83	-7,83%









7.
Conclusion and Future Work

Conclusion and Future Work

- This work has presented a DSE approach extended to implement a parallel genetic algorithm able to reduce execution time while it does not degrade the goodness of the final DSE solution founded
- An approach to finding the best PGA configuration is presented, where the designer can changes input parameters to achieve desired performances
- Results show the need to do not fix the PGA configuration using OpenMP, while the performances depend on population size and GA configuration
- FUTURE WORKS:
  - Exploit PGA on different technologies and parallel programming languages (GPU/CUDA, multi/many cores/OpenMPI, etc..)
  - Try to implement also fine-grained or coarse-grained PGA
  - Switch from an offline DSE to a runtime DSE able to self-adapt the system respect to several constraints (performance, power/energy, area, etc.)

Thanks!

Questions?