HETEROGENEOUS MULTIPROCESSOR MAPPING FOR REAL-TIME STREAMING SYSTEMS

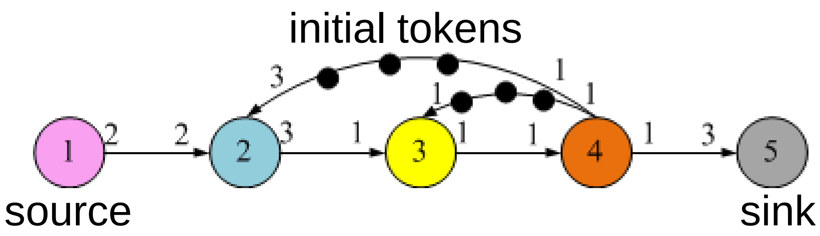
Jing Lin, Akshaya Srivasta,
Prof. Andreas Gerstlauer, and Prof. Brian L. Evans

Department of Electrical and Computer Engineering
The University of Texas at Austin

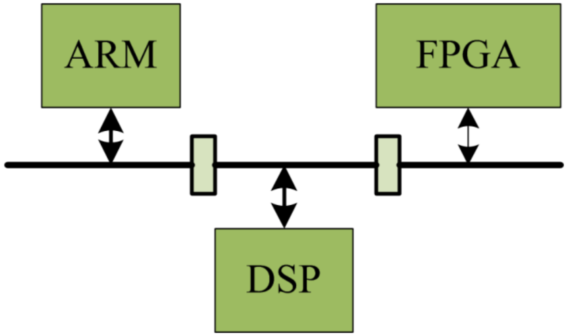
May 27, 2011

Introduction

- Synchronous data flow (SDF) models

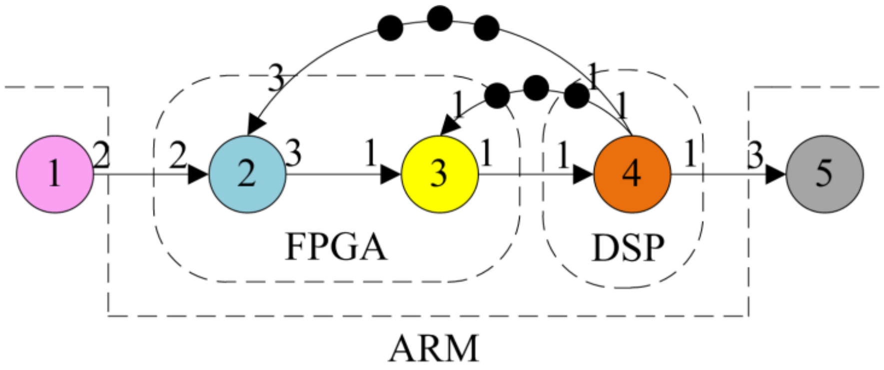


  - Static schedule: 1-2-3-4-3-4-3-4-5
  - Model for many real-time streaming applications, which desire high throughput and low latency
- Multiprocessor system-on-chips (MPSoCs)

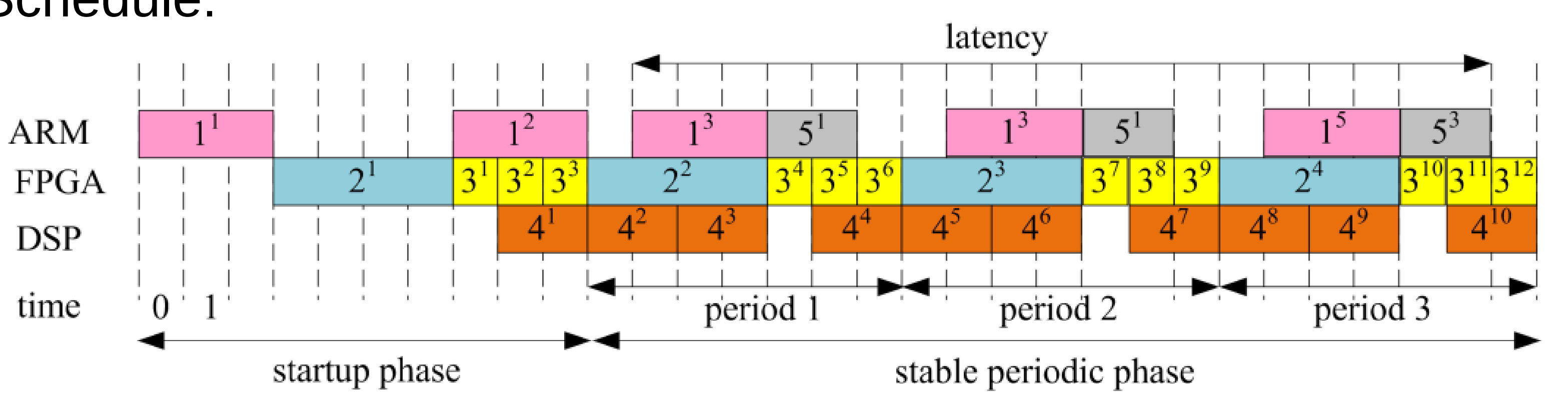


Problem Definition

- Mapping SDF models to MPSoCs
  - Partition:

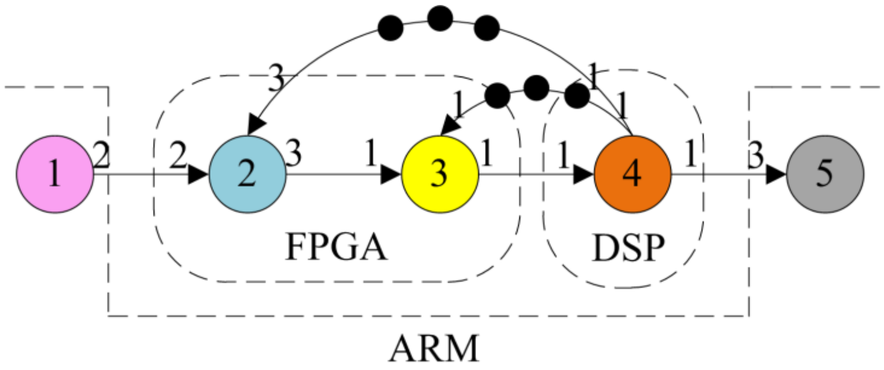


  - Schedule:

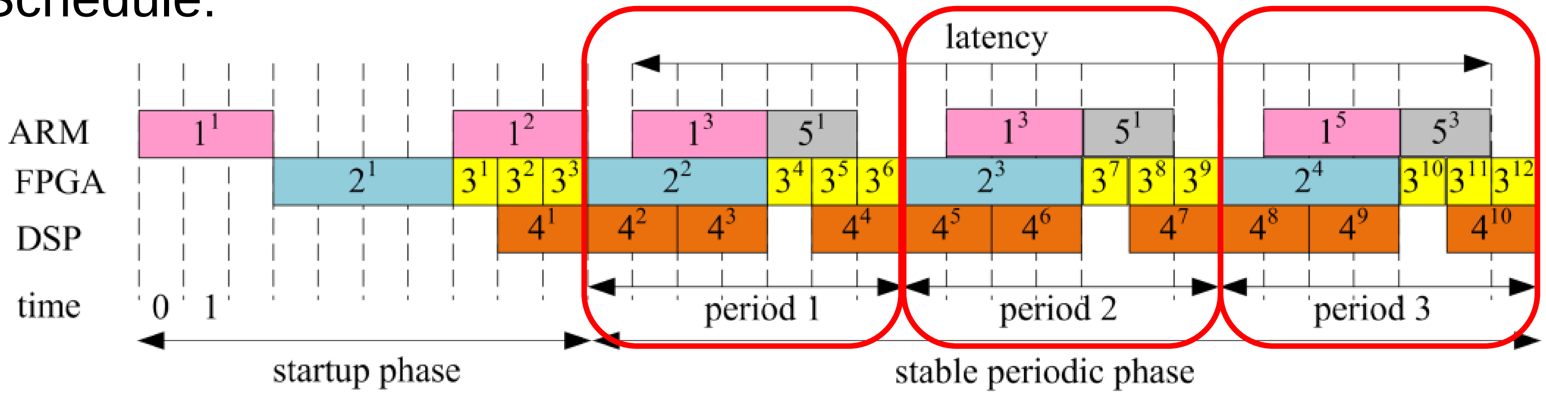


Problem Definition

- Mapping SDF models to MPSoCs
  - Partition:



  - Schedule:



Period = 1 / Throughput

Latency = (End of the *n*-th exec. of Sink) – (Start of the *n*-th exec. of Source)

Prior Work

Publication	General SDF	Processor Heterogeneity	Objectives	Solution Form	Main Approach
[Lee1987]	Yes	No	Throughput	Single solution	Linear programming
[Bha1996]	Yes	No	Throughput	Single solution	Linear programming
[Zhu2009]	Yes	Yes	Buffer size	Schedule only**	Constraint programming
[Bon2010]	No*	Yes	Throughput	Single solution	Graph-based solution
[Zit2000]	Yes	No	Multiple	Pareto front***	Evolutionary algorithm

* Homogeneous SDF graph only
** Partition is assumed to be given
*** A set of points that are Pareto optimal

Our Goal

- Mapping general SDF to heterogeneous MPSoCs
- Multi-objective optimization
  - Throughput
  - Latency
  - Processor cost (e.g. price, area)

A multi-objective optimization framework that jointly optimizes throughput, latency and processor cost for general multiprocessor SDF mapping.

Global Optimization

- An integer linear programming (ILP) Model
- Optimize partitioning and scheduling simultaneously
- Objective:

$$\text{Minimize } \lambda_1 \cdot Period + \lambda_2 \cdot Latency + \lambda_3 \bullet Cost$$

- Constraints
  - SDF semantics
  - Static partitioning
  - Execution time profile
  - Sequential execution of actors mapped to the same processor
  - Stable periodic schedule

Global Optimization

- Actor - i; Processor - j; Time - $t \in \{0,1,\ldots T\}$
- Decision variables
  - $S_i(t)$, $E_i(t)$: Number of started/ended executions of actor i up to time t
  - A_{ij}: Indicator of whether actor i is bound to processor j
  - $start(t)$: Indicator of the start of stable periodic phase
- Constraints
  - Execution precedence: $c_{i_1,i_2} S_{i_2}(t) \leq p_{i_1,i_2} E_{i_1}(t) + o_{i_1,i_2}$
  - Execution time: $S_i(t) = \sum_j A_{ij} E_i(t + d_{ij})$
  - Sequential execution: $\sum_j A_{ij}(S_i(t) - E_i(t)) \leq 1$
  - Periodicity of the schedule: $W_i(T) - \sum_t W_i(t) start(t) = n_i \sum_j A_{ij} d_{ij}$
  - Definition of objectives: $Period = T - \sum_t t \cdot start(t);\ Cost = \sum_j Alloc_j \cdot pc_j$

$$Latency = \underbrace{\sum_t (U(t) - V(t)) + \sum_j A_{Ij} d_{Ij}}_{\text{Time interval between Source's 1st start and Sink's 1st end in the periodic phase}} + \underbrace{(S_1(T) - S_I(T))}_{\text{Difference in iteration numbers}} \cdot Period$$

Global Optimization

- Actor - i; Processor - j; Time - $t \in \{0,1,\ldots T\}$
- Decision variables
  - $S_i(t)$, $E_i(t)$: Number of started/ended executions of actor i up to time t
  - A_{ij}: Indicator of whether actor i is bound to processor j
  - $start(t)$: Indicator of the start of stable periodic phase
- Constraints
  - Execution precedence: $c_{i_1,i_2} S_{i_2}(t) \le p_{i_1,i_2} E_{i_1}(t) + o_{i_1,i_2}$
  - Execution time: $S_i(t) = \sum_j A_{ij} E_i(t + d_{ij})$
  - Sequential execution: $\sum_j A_{ij}(S_i(t) - E_i(t)) \le 1$
  - Periodicity of the schedule: $W_i(T) - \sum_t W_i(t) start(t) = n_i \sum_j A_{ij} d_{ij}$
  - Definition of objectives: $Period = T - \sum_t t \cdot start(t)$; $Cost = \sum_j Alloc_j \cdot pc_j$

$$Latency = \sum_t (U(t) - V(t)) + \sum_j A_{Ij} d_{Ij} + (S_1(T) - S_I(T)) \cdot Period$$

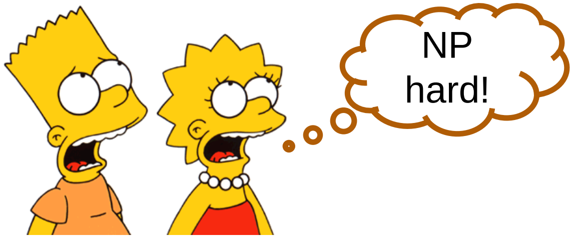
Time interval between Source's 1st start and Sink's 1st end in the periodic phase

Difference in iteration numbers

Linearize product terms: add one variable and three constraints

Express indicator functions with two constraints

Global Optimization



- Actor - i; Processor - j; Time - $t \in \{0,1,\ldots T\}$
- Decision variables
  - $S_i(t)$, $E_i(t)$: Number of started/ended executions of actor i up to time t
  - A_{ij}: Indicator of whether actor i is bound to processor j
  - $start(t)$: Indicator of the start of stable periodic phase
- Constraints
  - Execution precedence: $c_{i_1,i_2} S_{i_2}(t) \le p_{i_1,i_2} E_{i_1}(t) + o_{i_1,i_2}$
  - Execution time: $s_i(t) = \sum_j A_{ij} E_i(t + d_{ij})$
  - Sequential execution: $\sum_j A_{ij}(S_i(t) - E_i(t)) \le 1$
  - Periodicity of the schedule: $W_i(T) - \sum_t W_i(t) start(t) = n_i \sum_j A_{ij} d_{ij}$
  - Definition of objectives: $Period = T - \sum_t t \cdot start(t)$; $Cost = \sum_j Alloc_j \cdot pc_j$

$$Latency = \sum_t (U(t) - V(t)) + \sum_j A_{Ij} d_{Ij} + (S_1(T) - S_I(T)) \cdot Period$$

Time interval between Source's 1st start and Sink's 1st end in the periodic phase

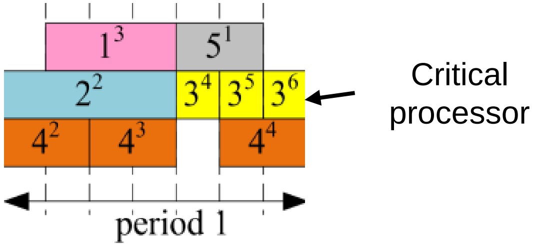
Difference in iteration numbers

Linearize product terms: add one variable and three constraints

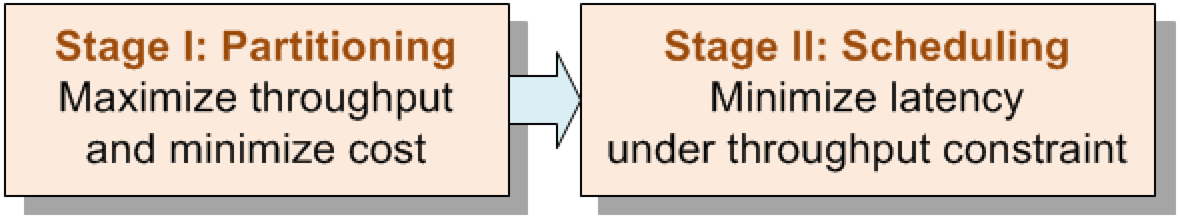
Express indicator functions with two constraints

Heuristic Optimization

- Maximum throughput partition
  - For fixed partition, the best throughput is determined by the critical processor
  - Empirically, the best throughput is achievable given long enough startup phase and proper scheduling
  - Just optimize partitioning for the best throughput and cost



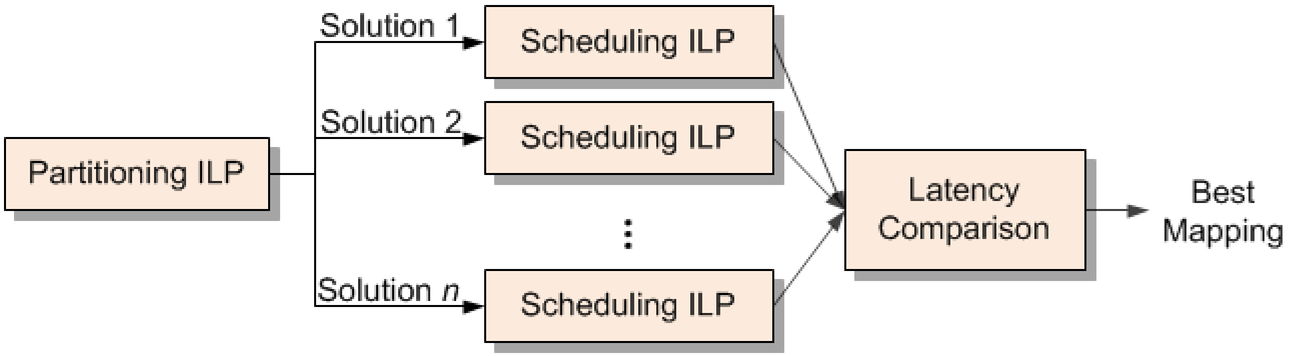
- Two-stage optimization process



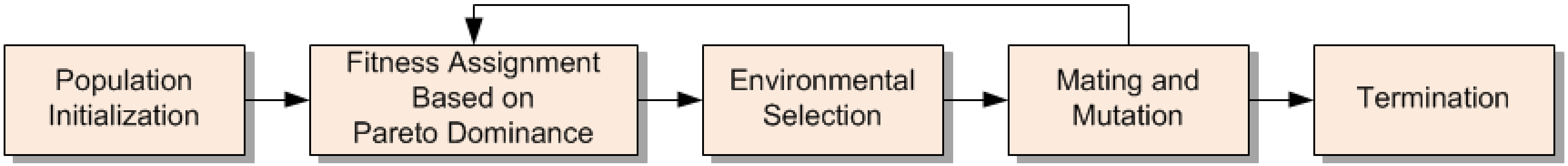
- Throughput and cost are prioritized over latency

Heuristic Optimization

- Two ILPs



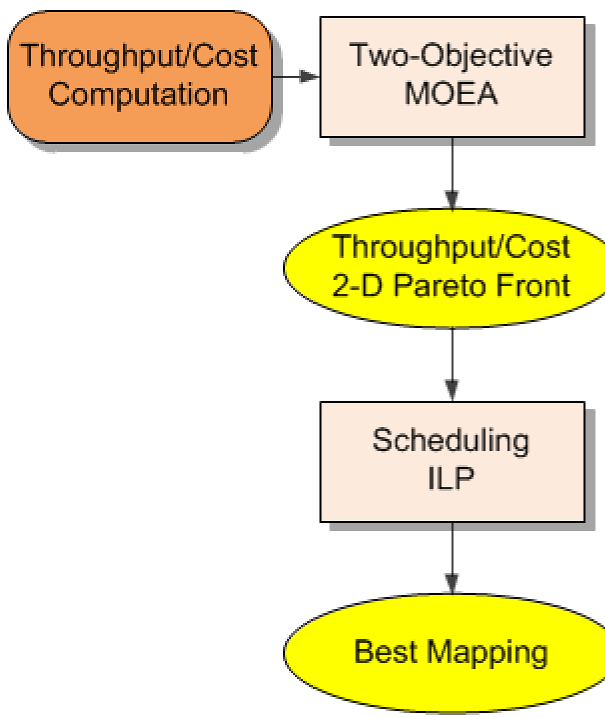
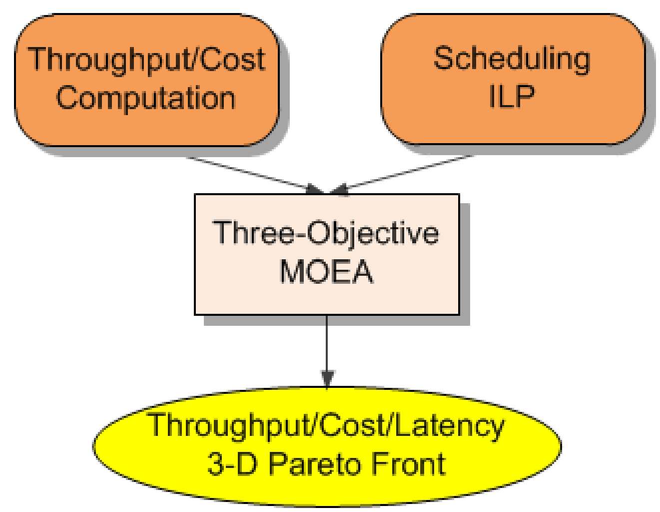
- Multi-Objective Evolutionary Algorithm (MOEA)



  - The population consists of a subset of all possible partitions
  - Converge to a set of multi-objective optimal partitions, i.e. Pareto front

Heuristic Optimization

- MOEA with Scheduling ILP

Generate a single solution	Generate a Pareto front
	

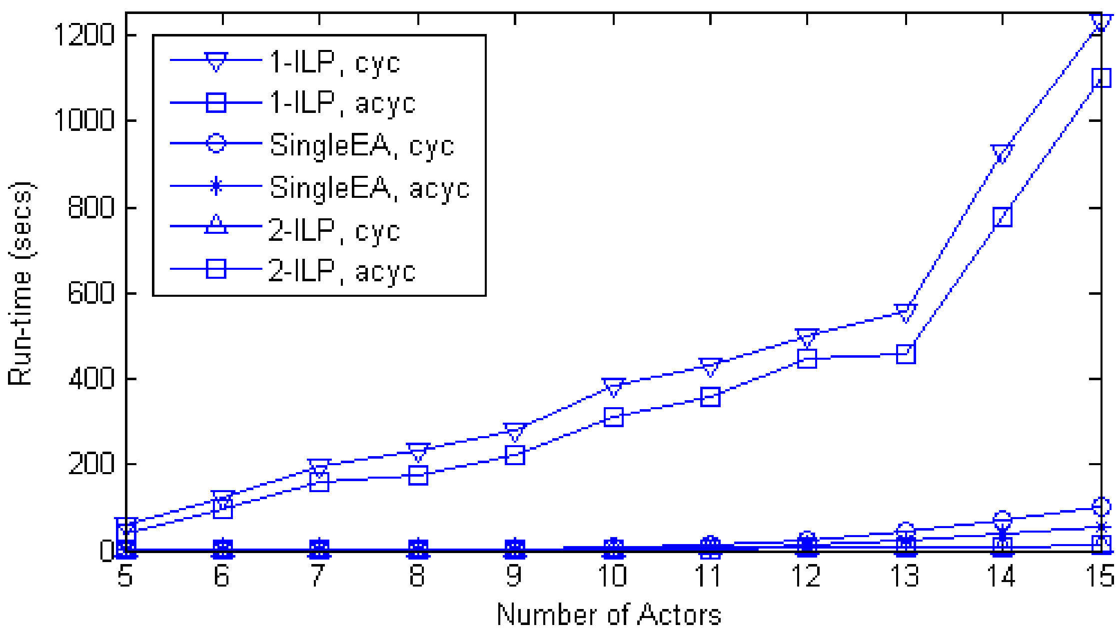
Global vs. Heuristic Optimization

	Global Optimization	Heuristic Optimization
Optimality	Global optimal	Sub-optimal
Computational Complexity	NP hard	MOEA: $O(M^2 logM)$ per iteration* Scheduling ILP: polynomial time
Design Space Exploration	Generate a single mapping; Generate a Pareto-front by fine tuning the weights	Generate a single mapping or a three-objective Pareto front

* M is the population size

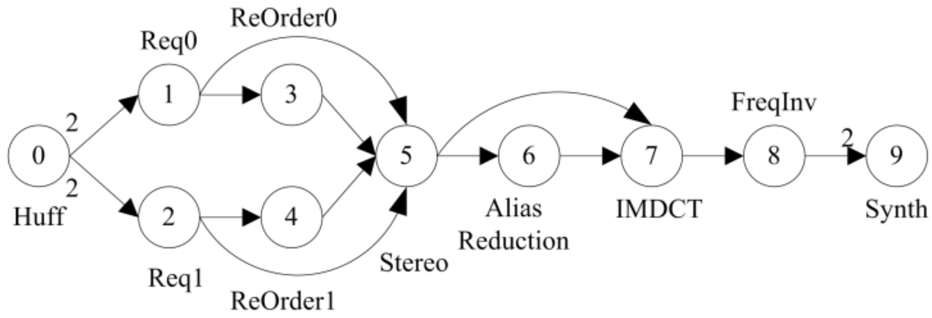
Experimental Results

- Programming Tools
  - ILP: CPLEX Concert Technology for C++
  - MOEA: MOGALib framework in C++
- Run-time comparison
  - random cyclic/acyclic SDF graphs mapped to 3 processors

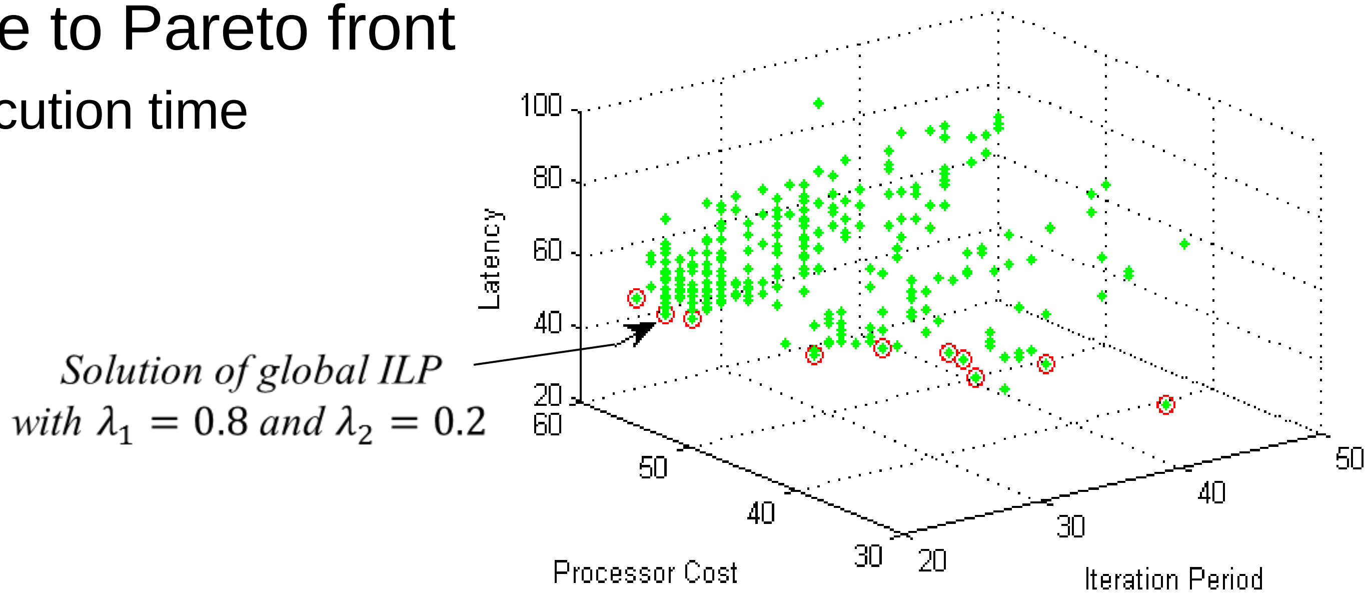


Experimental Results

- Design space exploration for an MP3 decoder



- Convergence to Pareto front
  - ~1 hour execution time



Conclusion

- Mapping SDF models onto heterogeneous MPSoCs
- Global ILP
- Heuristics by MOEA
- Generate a single mapping or a Pareto front

References

[1] S.S. Bhattacharyya, P.K. Murthy, and E.A. Lee, Software synthesis from dataflow graphs, Springer, 1996.

[2] E.A. Lee and D.G. Messerschmitt, “Static scheduling of synchronous data flow programs for digital signal processing,” IEEE Trans. on Computers, vol. 36, no. 1, pp. 24–35, 1987.

[3] J.L. Pino, T.M. Parks, and E.A. Lee, “Automatic code generation for heterogeneous multiprocessors,” in Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing, 1994, pp. 445–448.

[4] J. Zhu, I. Sander, and A. Jantsch, “Buffer minimization of realtime streaming applications scheduling on hybrid CPU/FPGA architectures,” in Proc. IEEE Conf. on Design, Automation and Test in Europe, 2009, pp. 1506–1511.

[5] A. Bonfietti, L. Benini, M. Lombardi, and M. Milano, “An efficient and complete approach for throughput-maximal SDF allocation and scheduling on multi-core platforms,” in Proc. IEEE Conf. on Design, Automation and Test in Europe, 2010, pp. 897–902.

[6] N. Aslam, T. Arslan, and A. Erdogan, “Algorithmic level design space exploration tool for creation of highly optimized synthesizable circuits,” in Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing, 2007, vol. 2.

[7] E. Zitzler, J. Teich, and S.S. Bhattacharyya, “Evolutionary algorithms for the synthesis of embedded software,” IEEE Trans. on VLSI Systems, vol. 8, no. 4, pp. 452–455, 2000.

[8] F. Glover, “Improved linear integer programming formulations of nonlinear integer problems,” Management Science, pp. 455–460, 1975.

[9] E. Zitzler, M. Laumanns, and L. Thiele, “SPEA2: Improving the strength Pareto evolutionary algorithm,” in Eurogen, 2001, vol. 3242.

[10] S. Stuijk, M. Geilen, and T. Basten, “SDFˆ3: SDF For Free,” in Proc. IEEE Int. Conf. on Application of Concurrency to System Design, 2006, pp. 276–278.

[11] J.T. Alander, “On optimal population size of genetic algorithms,” in Proc. IEEE Int. Conf. on Comp. Sys. and Software Eng., 2002, pp. 65–70.

Thank you for your attention!