Move Based Algorithm for Runtime Mapping of Dataflow Actors on Heterogeneous MPSoCs

Kevin J. M. Martin Thanh Dinh Ngo Jean-Philippe Diguet

Univ. Bretagne-Sud
UMR CNRS 6285, Lab-STICC
Lorient, France

20/10/2017

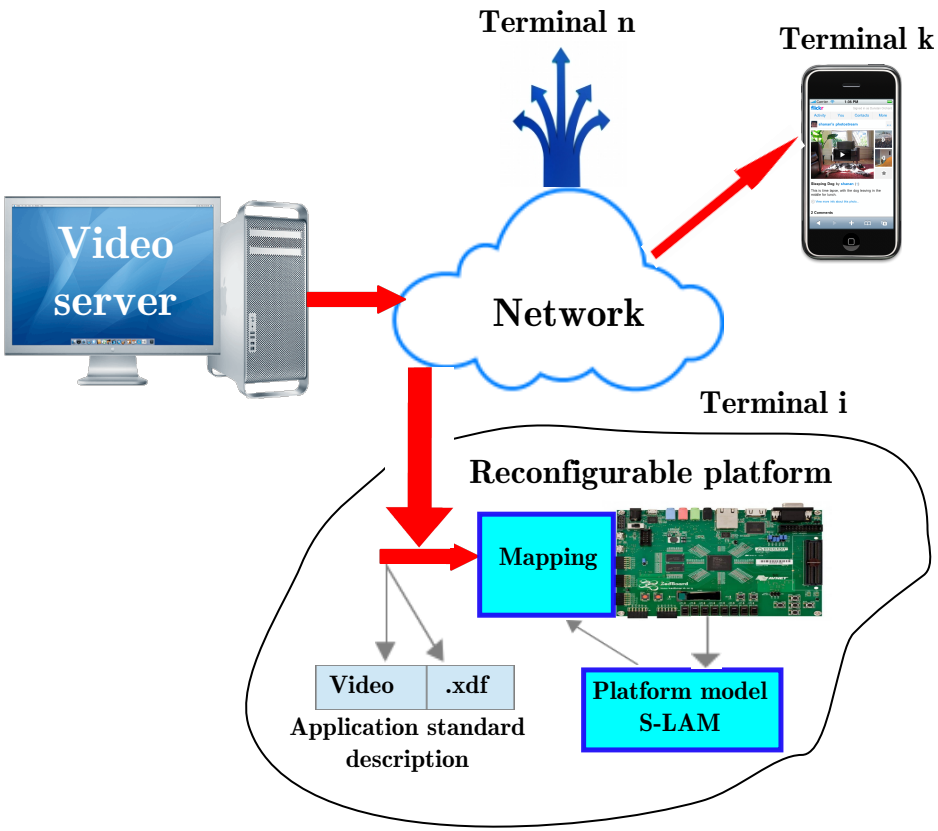
Groupe de Travail Optimisation pour les Systèmes Intégrés

Outline

Outline

Introduction

Video Streaming

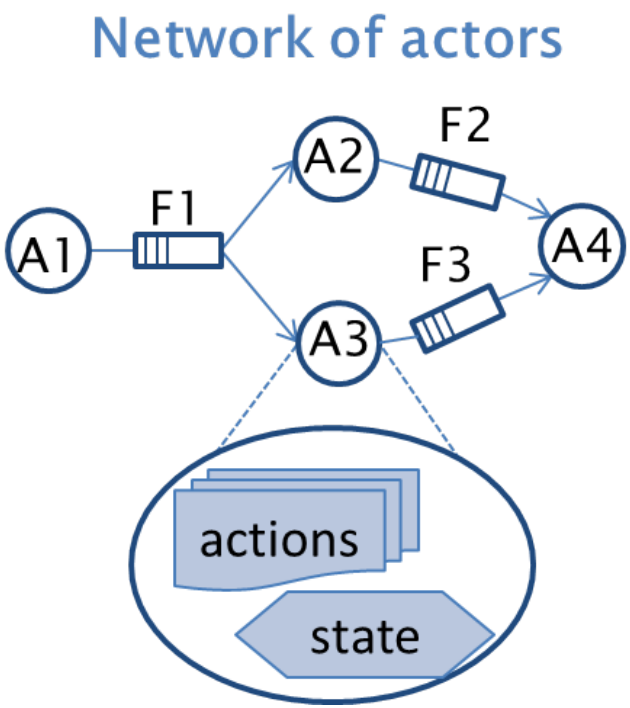


Video coding

1. Standards
2. Evolution and profiles

Introduction

Dataflow

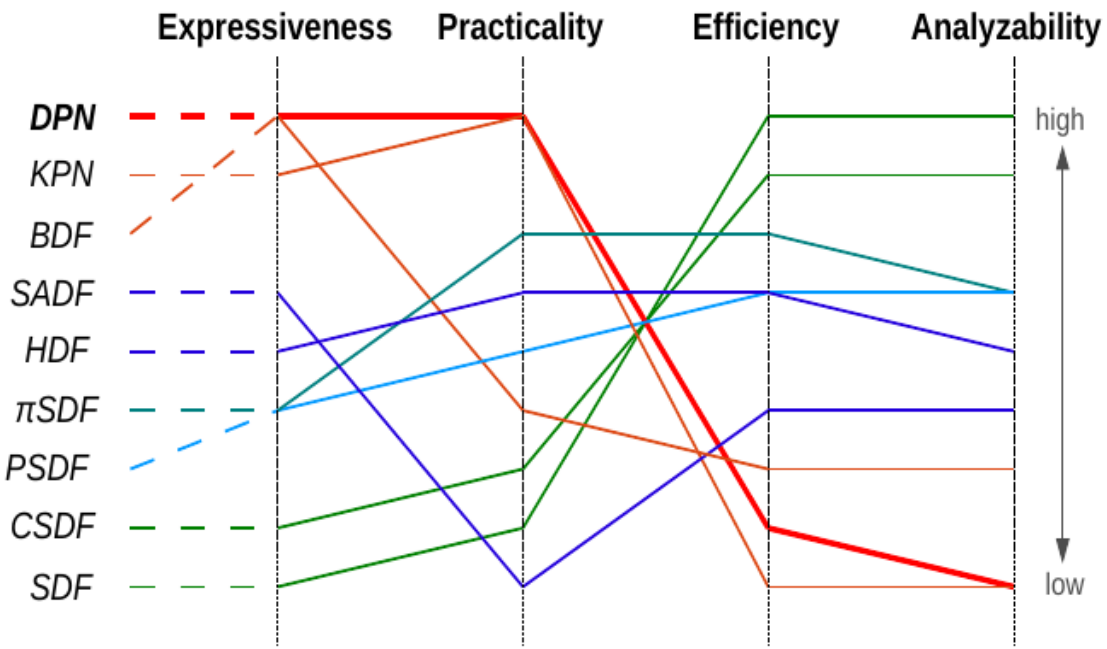


Dataflow

- Formal Model of Computation (MoC)
- Explicit spatial and temporal parallelism
- Static or dynamic actors
  - Execute actions (“fire” actions)
- Firing rule
  - Enough tokens in input FIFOs
  - Enough space in output FIFOs

Introduction

Dataflow Model of Computations



Dynamic MoC

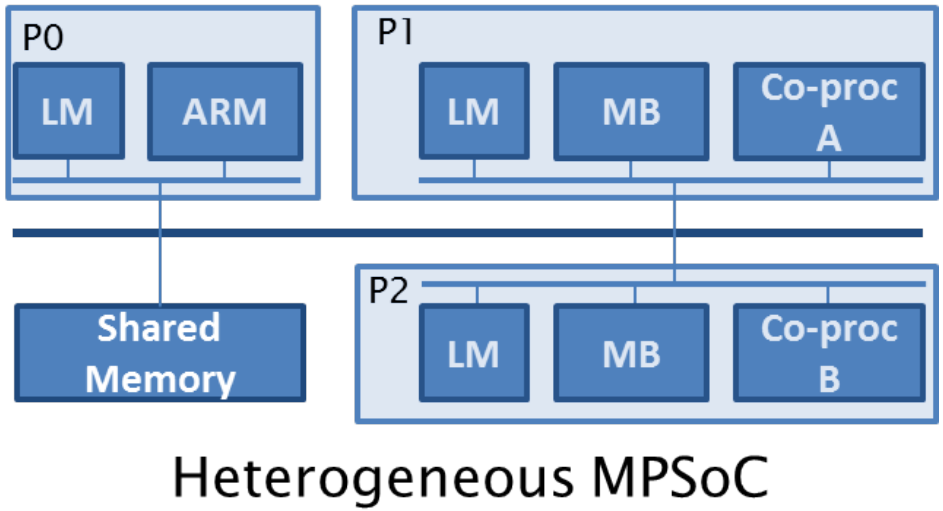
- ✓ Expressivity
- × Analyzability

Static MoCs

- × Expressivity
- ✓ Analyzability

Introduction

Heterogeneous Multi-processor platform



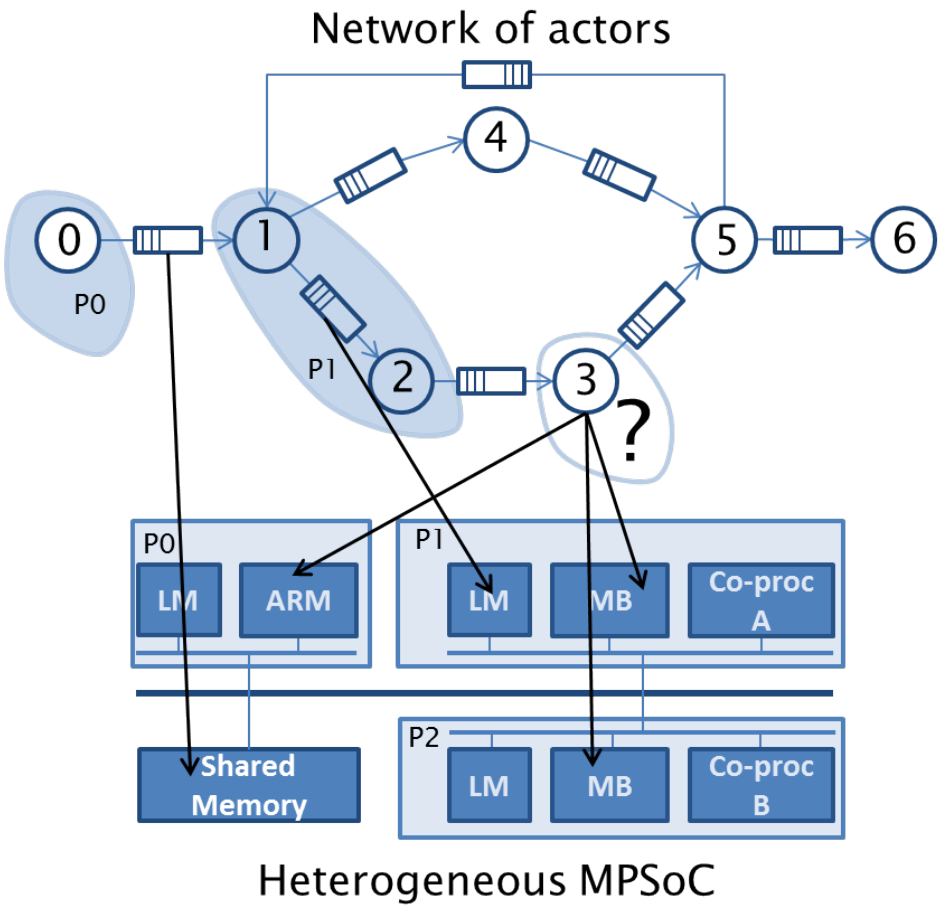
Heterogeneous MPSoC

Platform components

- General purpose processors
- Custom processors (ASIP)
- Hardware accelerators

Introduction

Dataflow Mapping and scheduling



Actor mapping

- Equivalent to graph partitioning problem
- NP hard
- Computational load over processors
- Communication load / connectivity of actors
- Platform characteristics

Introduction

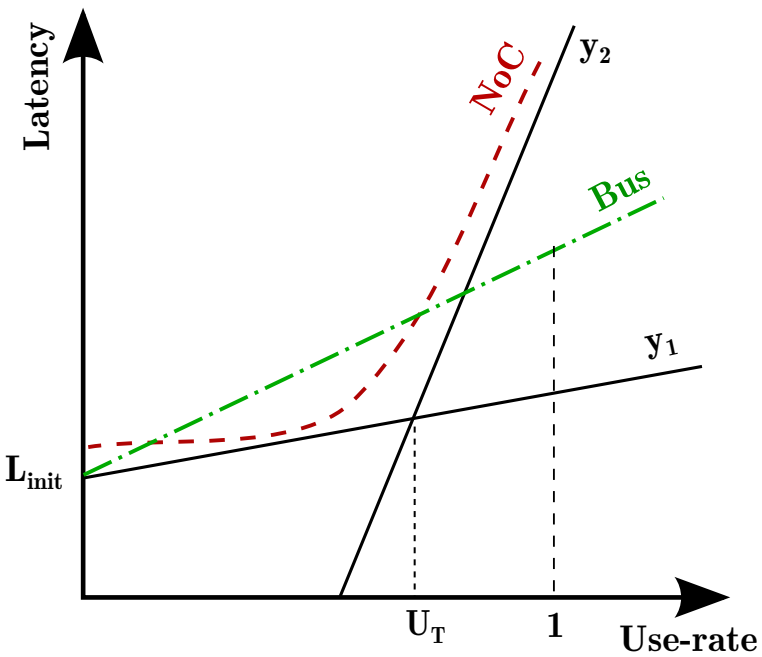
Dataflow Mapping and scheduling

Different kinds of mapping

- Static mapping (at compile time)
- On-the-fly mapping (at run-time)
- Hybrid mapping (mix of two above)
- Runtime mapping
- Runtime remapping
  - Ability to change the mapping during the execution of the application

Introduction

Communication model



Analytical model

- Generic and parametric communication model
- Link between use-rate and latency
- Communications of a NoC or a bus

$$f_{cl}(x) = \begin{cases} y_1 & \text{if } x \leq threshold(U_T) \\ y_2 & \text{otherwise} \end{cases} \quad (1)$$

Outline

Related work

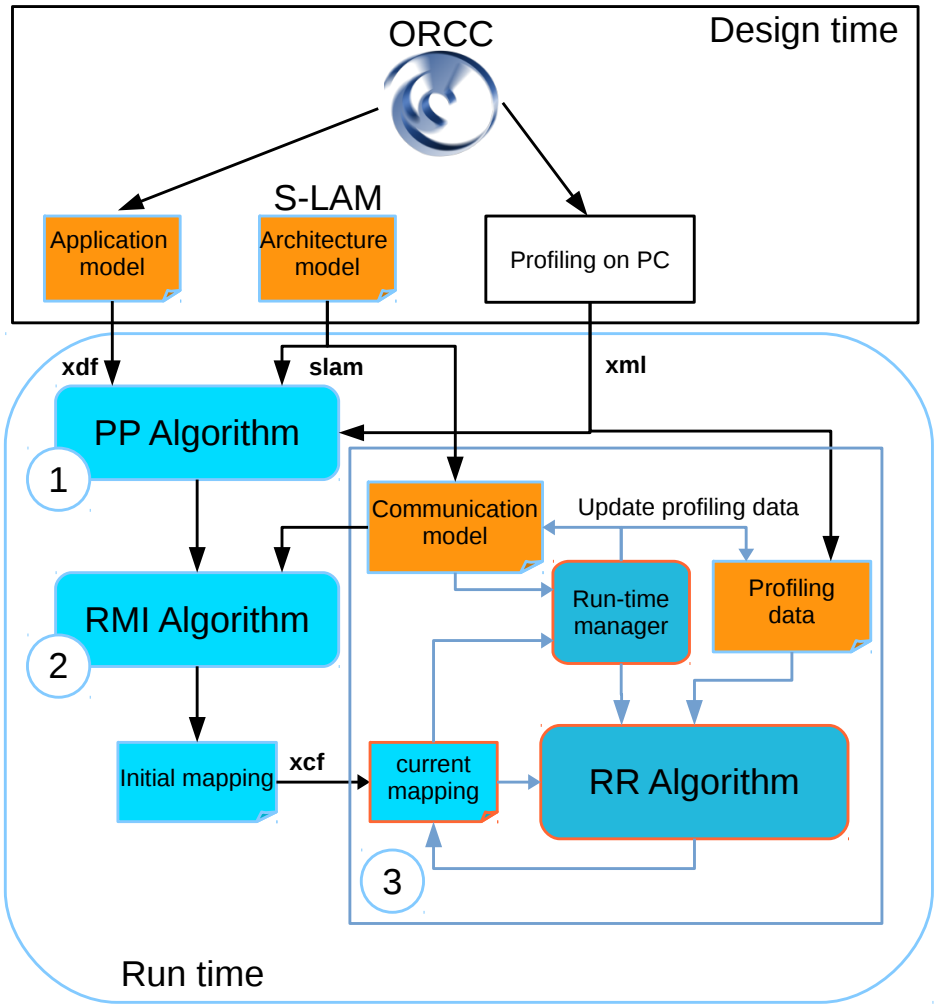
Table: Various approaches for the mapping of streaming applications

Ref.	MoC	Platform	Comm.	Map.
[18], [8]-2008	SDF	Fixed-Homo	N/A	Static
[3]-2010	N/A	Generic-Heter	NoC	On-the-fly
[10]-2008	N/A	Fixed-Heter	Constant	On-the-fly
[17]-2013	SDF	Generic-Heter	NoC	Hybrid
[15]-2012	KPN	Fixed-Homo	NoC	Hybrid
[20]-2011	SADF	Generic-Heter	N/A	N/A
[21]-2013	DPN	Generic-Homo	Constant	Hybrid
[1]-2013	KPN	Generic-Heter	Yes	Hybrid
[4]-2011	N/A	Fixed-Homo	NoC	Hybrid
[2]-2012	KPN	Fixed-Heter	Yes	N/A
[16]-2010	N/A	Fixed-Heter	N/A	Hybrid
[7]-2012	SDF	Generic-Heter	IPC	Hybrid
[13]-2013	KPN	Generic-Homo	Constant	Hybrid
[9]-2014	DPN	generic-Heter	Yes	Hybrid
[14]-2015	KPN	Generic-Heter	Constant	Hybrid+R
[5]-2013	SDF	Generic-Homo	NoC	Hybrid+R
Ours	**DPN**	**Generic-Heter**	**Yes**	**Hybrid+R**

Outline

Move-based algorithm

Overview



At design time

- Application model
- Architecture and communication model
- Application profiling

At run time

1. Pre-Processing (PP)
2. Runtime Mapping Initialisation (RMI)
3. Runtime Remapping (RR)

Move-based algorithm

Parameters and evaluation metrics

$$W^i = \sum_{j \in \mathbb{P}} R_i * T_{ij} \qquad \forall i \in \mathbb{A} \qquad (2)$$

$$memUsage_j = \sum_{i:\mathbb{P}[i]=j} Cs^i < ICache_j \qquad \forall j \in \mathbb{P} \qquad (3)$$

$$compT_j = \sum_{i:\mathbb{P}[i]=j} R_i * T_{ij} \qquad \forall j \in \mathbb{P} \qquad (4)$$

$$commT_j = \sum_{i:\mathbb{P}[i]=j} 0.5 * f_{cl}(x) * comm(i) * T_c \qquad (5)$$

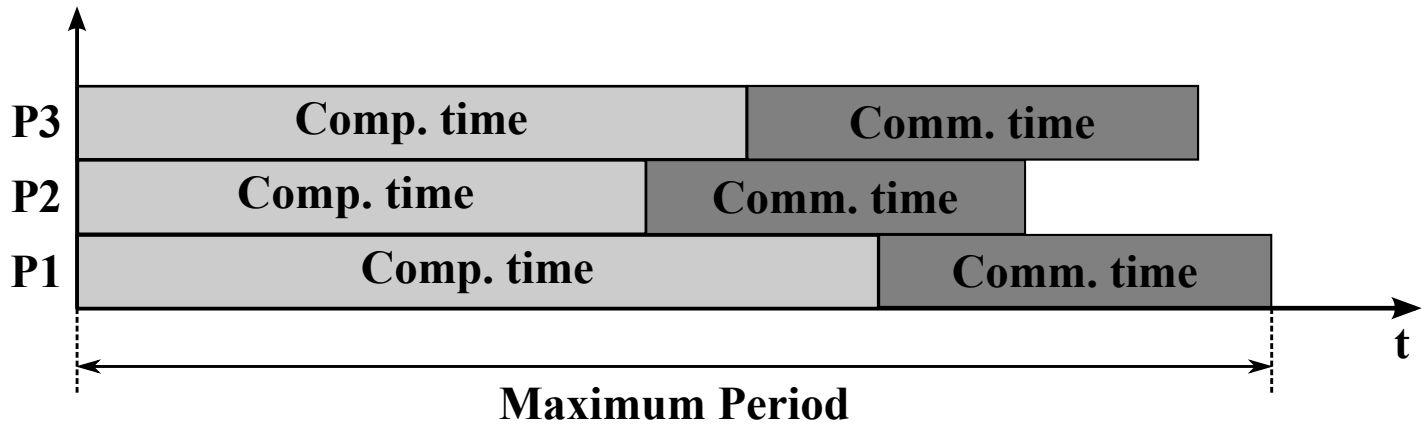
$$Period_j = compT_j + commT_j \qquad \forall j \in \mathbb{P} \qquad (6)$$

$$Th = \frac{1}{\max_{j \in \mathbb{P}}(Period_j)} \qquad (7)$$

Parameter	Definition
DPN application graph (DPNapp)	
$\lvert\mathbb{A}\rvert$	Number (Nb) of actors
$\lvert\mathbb{F}\rvert$	Nb of FIFO channels
Architecture graph (arch)	
$\lvert\mathbb{P}\rvert$	Nb of processors
$ICache_j$	Instruction cache size of processor j
Profiling data (profile)	
R_i	Mean number of firings of actor i
W^i	Total computation cost of actor i
Cs^i	Instruction code size of actor i
T_{ij}	Mean execution time of actor i on processor j
T_c	Communication time per token
Evaluation metrics	
$compT_j$	computation time of processor j
$commT_j$	communication time of processor j
$comm(i)$	total number of tokens transferred on the communication media of actor i

Move-based algorithm

Parameters and evaluation metrics



Objective

Minimize the maximum period

Move-based algorithm

Pre-processing - PP

Objective

Find the *lower bound period*, the unreachable lowest period to define the processing budget for all processors

Assumptions

- Profiling of the execution time of each actor on each processor

Move-based algorithm

Runtime mapping initialization - RMI

Objective

Find rapidly a *good* initial mapping

Greedy algorithm + speculative approach

- relies on a factor called alpha (α), a ratio of processor cycle budget considered during the first step (computing-oriented) mapping
- takes into account both computation and communication workloads
- performs both actor and data mappings

Move-based algorithm

Runtime mapping initialization - RMI

Two phases algorithm

1. Computation phase
2. Communication phase

1) Computation phase

1. Sorting actors according to their total **computation** cost (bubble sort)
2. Map the first actor to its best processor
3. Update the processing use until α

2) Communication phase

1. Sorting remaining actors according to their total **communication** use
   - actors on same processor $\Rightarrow$ communication time = 0
   - actors on different processors $\Rightarrow$ "bet" with communication model
2. Map the first actor to the processor with least communication
3. Update the processing use

Move-based algorithm

Algorithm principle

During application execution

- System monitoring and application profiling (workload)
- Unbalancing load over the processors $\Rightarrow$ Remapping needed?

Runtime remapping

- Inspired by the Fiduccia and Mattheyses algorithm (FM), a famous partitioning algorithm used in VLSI [6]
- Only one actor is allowed to move from one processor to another
- Actor move $\Rightarrow$ Gain but also Cost
- Balance between migration cost and performance improvement

$$Cost_{remap} = max(gainT(i)) \tag{8}$$

Runtime remapping - RR

Finding the possible moves

Modified FM algorithm

FM algorithm	Our algorithm
Cell move	Actor move
2 partitions	*n* processors
All cells	Only actors on the processor with maximum period
Off-line	At runtime

Consider the processor with the maximum period

Runtime remapping - RR

Trade-off between migration cost and performance improvement

Actor move decision

1. Unlock all actors in the list of actor move candidates
2. For each actor
   - Compute the gain of performance and a migration cost
   - lock actor
3. Find the maximum gain

Migration cost

- All the binary code of actors are contained in a shared memory
- Migration cost = function of cache miss that happens when moving actor from one processor to another one

Move-based algorithm

Runtime remapping - RR

$$Cost_{mig}(i) = f_{cl}(x) * Cs^i \quad (9)$$

$$PerG(i) = maxPerI - maxPerN(i) \quad (10)$$

$$gainT(i) = N * PerG(i) - Cost_{mig}(i) \quad (11)$$

Where

- Cs^i: size of the binary code of actor i
- $f_{cl}(x)$: communication latency
- maxPerI: maximum period before actor move
- maxPerN: maximum new period
- The cost is smoothed over N periods

Outline

Results

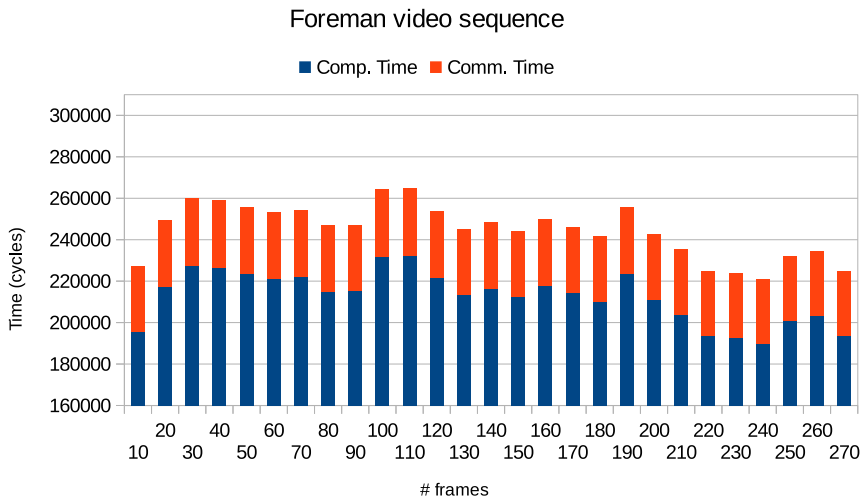
Setup environment

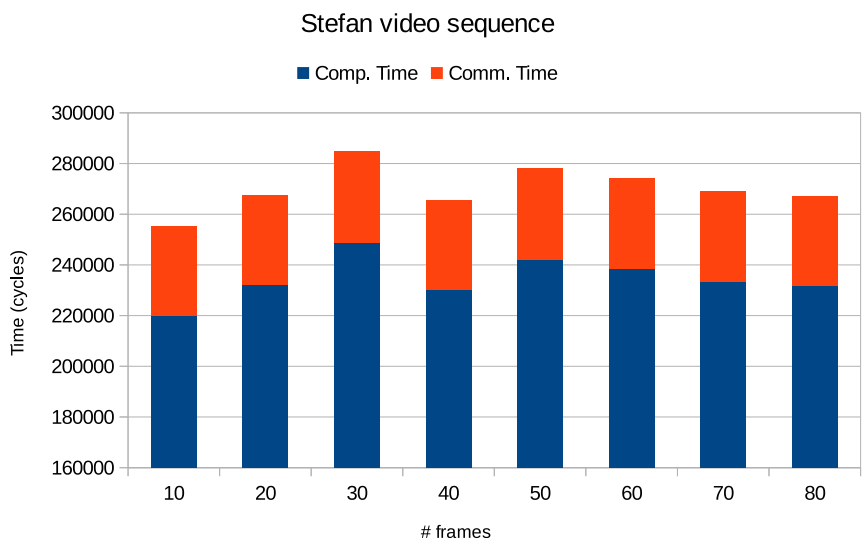
Tools

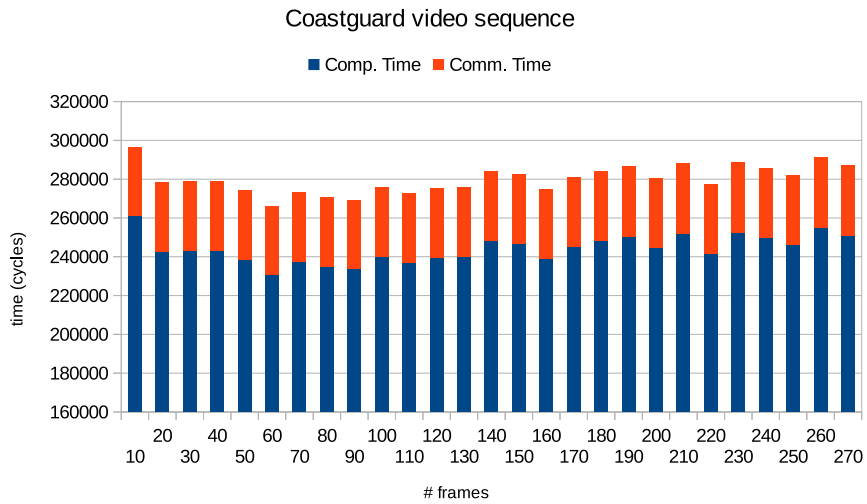
- SDF3 tool [19]: generated dynamic dataflow applications
- ORCC [11]: real dataflow applications
  - MPEG4 Part 2 Simple Profile (MPEG4-SP)
- System-Level Architecture Model (S-LAM) [12]

Results

The need of runtime remapping







Results

Results on MPEG4-SP

Properties of MPEG4-SP video decoder

Decoder	Profile	YUV	#Actors	#FIFOs
MPEG4 Part 2	SP	yes	41	104

Accelerators used in platform 7.1

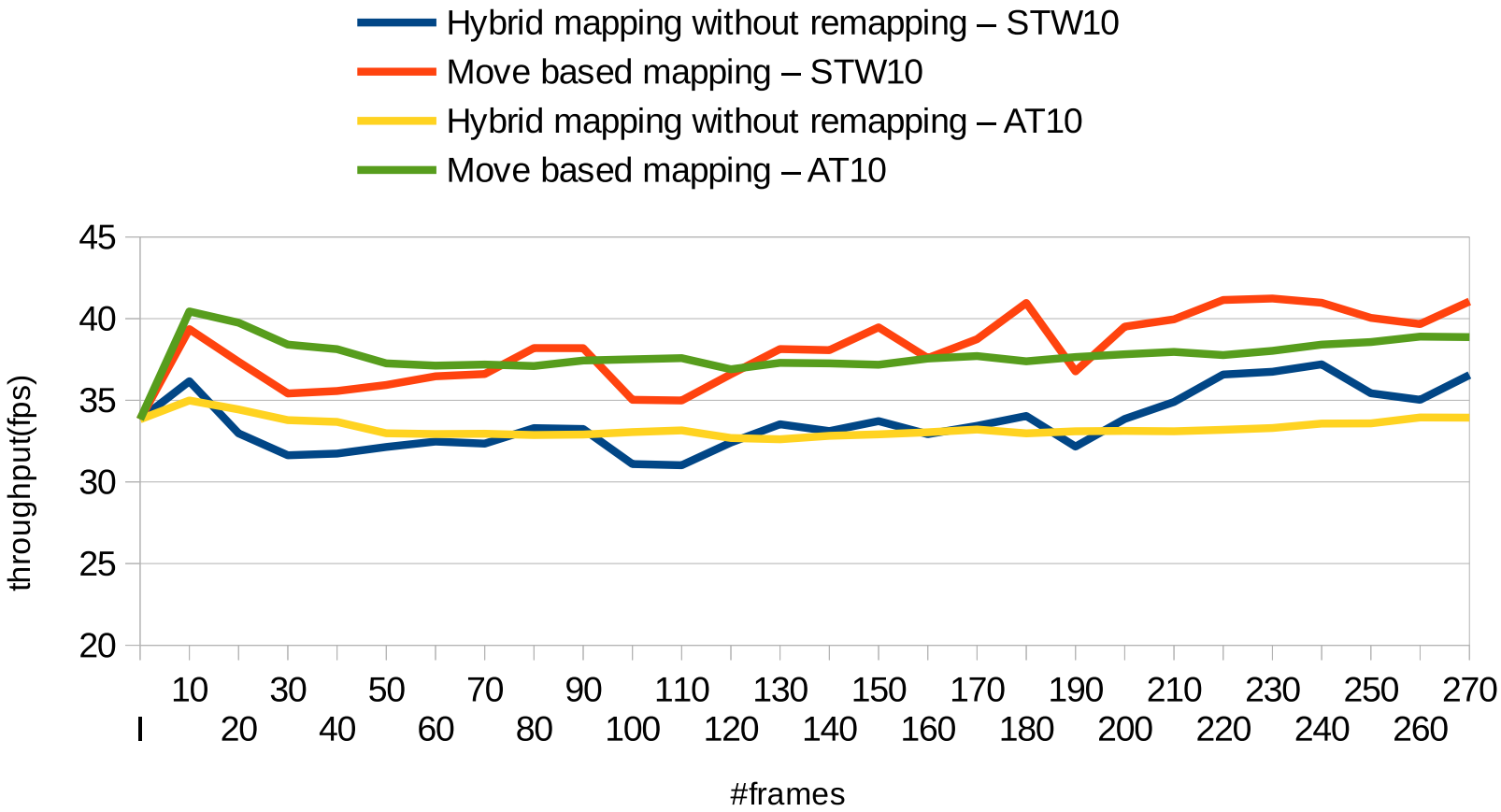
Platform	MB1	MB2	MB3	MB4	MB5	MB6	MB7
7.1	Merger	IDCT	Parser	Inter	IQ+IAP	Add	IDCT

Profiling workload

- AT10: Average time updated after 10 frames
- SWT10: Sliding Window Time, last 10 frames

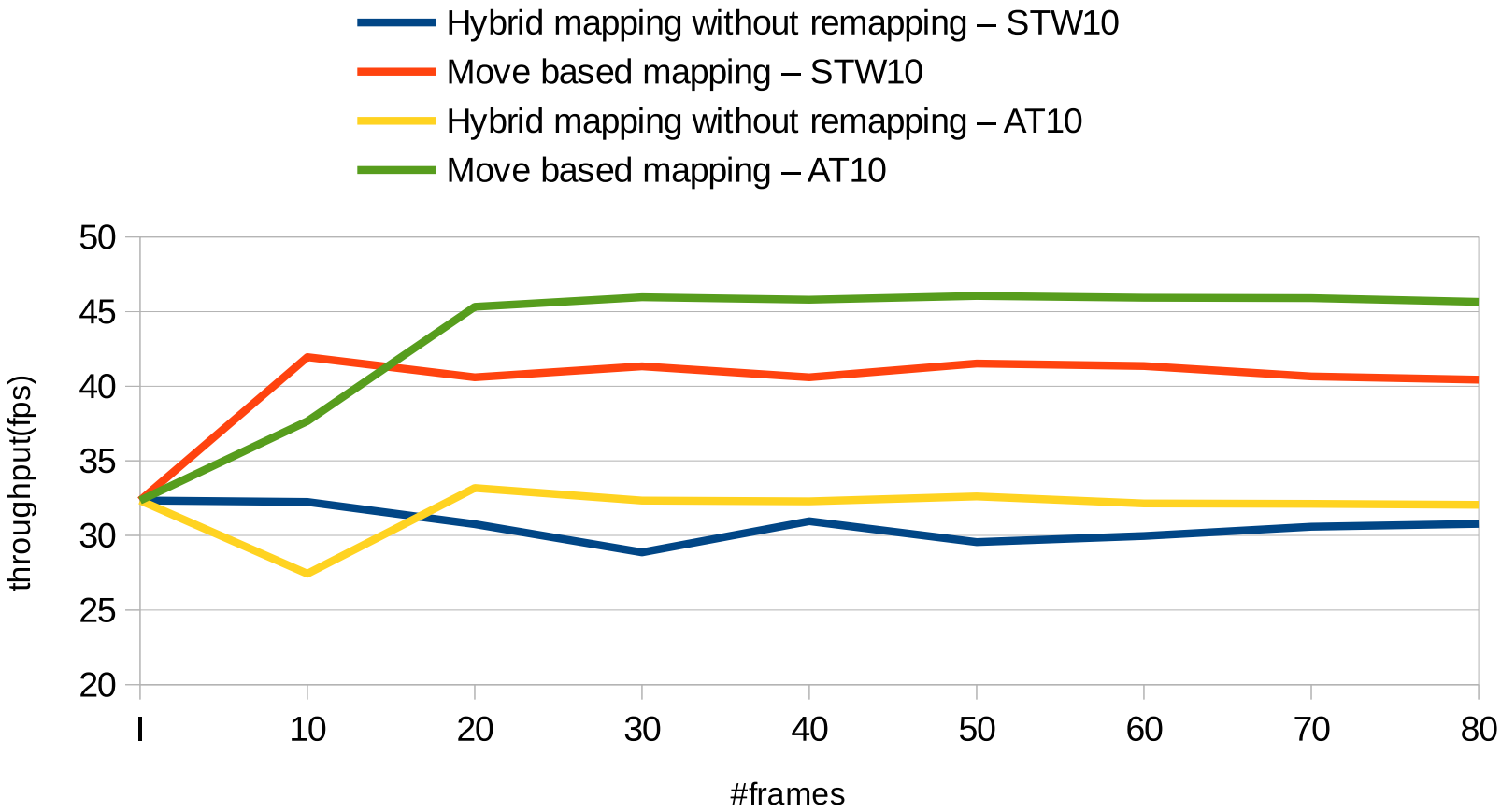
Results

Results on MPEG4-SP/ Foreman



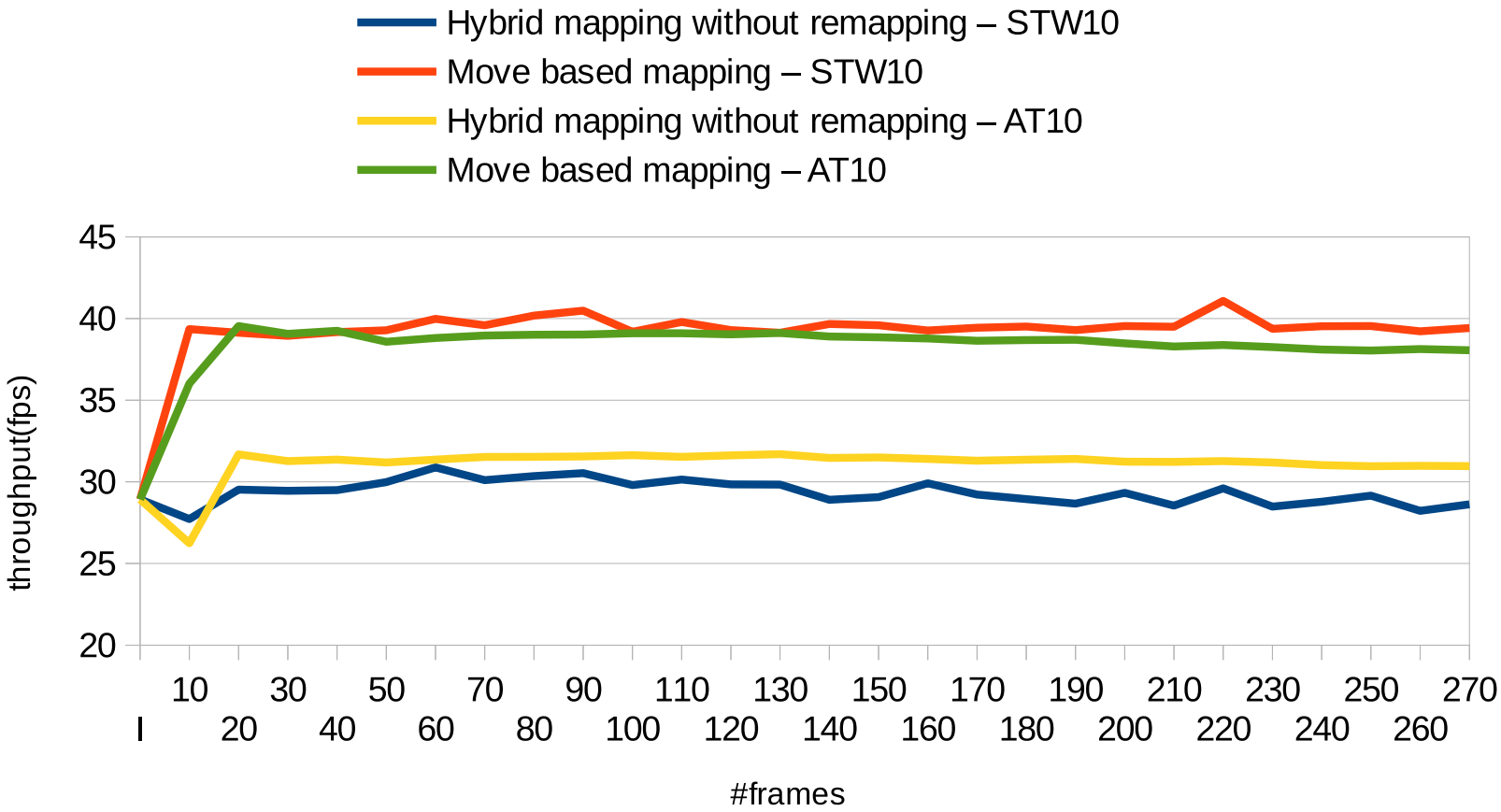
Results

Results on MPEG4-SP/ Stefan



Results

Results on MPEG4-SP/ Coastguard



Summary

- Move based algorithm for runtime (re)mapping of dataflow actors
  - Profiling / monitoring of the application
  - Fast algorithm
- Take into account migration cost when moving actors

References I

[1] J. Castrillon, R. Leupers, and G. Ascheid.
MAPS: Mapping Concurrent Dataflow Applications to Heterogeneous MPSoCs.
Industrial Informatics, IEEE Transactions on, 9(1):527–545, Feb 2013.

[2] J. Castrillon, A. Tretter, R. Leupers, and G. Ascheid.
Communication-aware mapping of KPN applications onto heterogeneous MPSoCs.
In *Proceedings of the 49th Annual Design Automation Conference*, DAC '12, pages 1266–1271, New York, NY, USA, 2012. ACM.

[3] E. de Souza Carvalho, N. Calazans, and F. Moraes.
Dynamic task mapping for MPSoCs.
Design Test of Computers, IEEE, 27(5):26–35, Sept 2010.

[4] S. Kaushik, A. Singh, and T. Srikanthan.
Computation and communication aware run-time mapping for NoC-based MPSoC platforms.
In *SOC Conference (SOCC), 2011 IEEE International*, pages 185–190, Sept 2011.

[5] C. Lee, S. Kim, and S. Ha.
Efficient run-time resource management of a manycore accelerator for stream-based applications.
In *Embedded Systems for Real-time Multimedia (ESTIMedia), 2013 IEEE 11th Symposium on*, pages 51–60, Oct 2013.

[6] S. K. Lim.
Practical Problems in VLSI Physical Design Automation.
Springer Publishing Company, Incorporated, 1 edition, 2008.

[7] J. Lin, A. Gerstlauer, and B. L. Evans.
Communication-aware heterogeneous multiprocessor mapping for real-time streaming systems.
Journal of Signal Processing Systems, 69(3):279–291, Dec. 2012.

[8] W. Liu, M. Yuan, X. He, Z. Gu, and X. Liu.
Efficient sat-based mapping and scheduling of homogeneous synchronous dataflow graphs for throughput optimization.
In *Real-Time Systems Symposium*, 2008.

[9] D.-T. Ngo, J.-P. Diguet, K. Martin, and D. Sepulveda.
Communication-model based Embedded Mapping of Dataflow Actors on Heterogeneous MPSoC.
In *Proceedings of the Conference on Design and Architectures for Signal and Image Processing (DASIP)*, 2014.

References II

[10] V. Nollet, P. Avasare, H. Eeckhaut, D. Verkest, and H. Corporaal.
Run-Time Management of a MPSoC Containing FPGA Fabric Tiles.
Very Large Scale Integration (VLSI) Systems, IEEE Transactions on, 16(1):24–33, Jan 2008.

[11] ORCC.
The Open RVC-CAL Compiler : A Development Framework for Dataflow Programs.

[12] M. Pelcat, J. F. Nezan, J. Piat, J. Croizer, and S. Aridhi.
A System-Level Architecture Model for Rapid Prototyping of Heterogeneous Multicore Embedded Systems.
In *Conference on Design and Architectures for Signal and Image Processing (DASIP) 2009*, page 8 pages, nice, France, Sept. 2009.

[13] W. Quan and A. D. Pimentel.
A scenario-based run-time task mapping algorithm for MPSoCs.
In *Proceedings of the 50th Annual Design Automation Conference*, DAC '13, pages 131:1–131:6, New York, NY, USA, 2013. ACM.

[14] W. Quan and A. D. Pimentel.
A Hybrid Task Mapping Algorithm for Heterogeneous MPSoCs.
ACM Trans. Embed. Comput. Syst., 14(1):14:1–14:25, Jan. 2015.

[15] L. Schor, I. Bacivarov, D. Rai, H. Yang, S.-H. Kang, and L. Thiele.
Scenario-based design flow for mapping streaming applications onto on-chip many-core systems.
In *Proceedings of the 2012 International Conference on Compilers, Architectures and Synthesis for Embedded Systems*, CASES '12, New York, NY, USA, 2012. ACM.

[16] A. Schranzhofer, J.-J. Chen, L. Santinelli, and L. Thiele.
Dynamic and adaptive allocation of applications on MPSoC platforms.
In *Design Automation Conference (ASP-DAC), 2010 15th Asia and South Pacific*, pages 885–890, Jan 2010.

[17] A. K. Singh, A. Kumar, and T. Srikanthan.
Accelerating throughput-aware runtime mapping for heterogeneous MPSoCs.
ACM Trans. Des. Autom. Electron. Syst., 18(1), 2013.

[18] S. Stuijk, M. Geilen, and T. Basten.
Exploring trade-offs in buffer requirements and throughput constraints for synchronous dataflow graphs.
In *43rd ACM/IEEE DAC*, pages 899–904, 2006.

References III

[19] S. Stuijk, M. Geilen, and T. Basten.
Sdf3: Sdf for free.
In *6th Int. Conf. on Application of Concurrency to System Design,*, 2006.

[20] S. Stuijk, M. Geilen, B. Theelen, and T. Basten.
Scenario-aware dataflow: Modeling, analysis and implementation of dynamic applications.
In *Int. Conf on Embedded Computer Systems (SAMOS)*, 2011.

[21] H. Yviquel, E. Casseau, M. Raulet, P. Jääskeläinen, and J. Takala.
Towards run-time actor mapping of dynamic dataflow programs onto multi-core platforms.
In *Int. Symp. on Image and Signal Processing and Analysis (ISPA)*, France, 2013.