
Initial Performance Evaluation of a Novel Execution Model (DTA) mapping to Haskell PM

Cluster: Programming Models & OS

Adrian Cristal (BSC, HiPEAC member)

Roberto D'Aprile (UPC PhD Student, HiPEAC member)

Roberto Giorgi (UNISI, HiPEAC member)

Nikola Puzović (UNISI, PhD Student, HiPEAC member)

Osman Unsal (BSC, HiPEAC member)

Mateo Valero (BSC, HiPEAC member)

Institutions:

BSC=Barcelona Supercomputing Center

UNISI=University of Siena

Decoupled Threaded Architecture

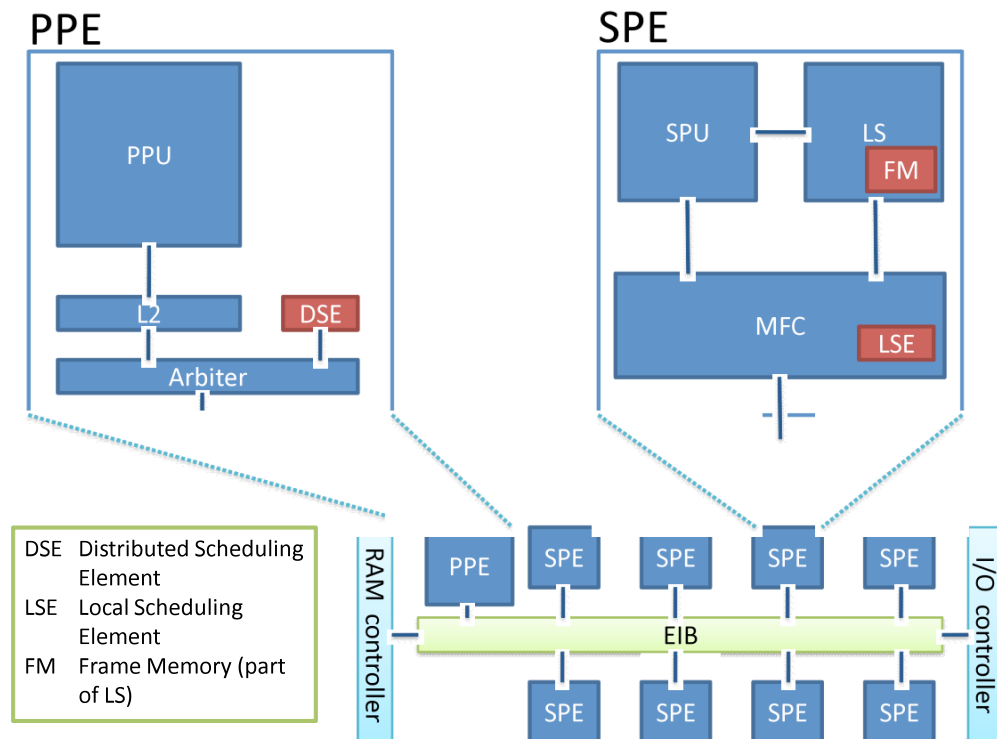
- Developed in SARC (Scalable ARChitectures) Integrated Project

- DTA provides support for *fast* TLP management

- ISA support for TLP and (HW) scalable distributed scheduler

- Implementation of DTA in UNISIM

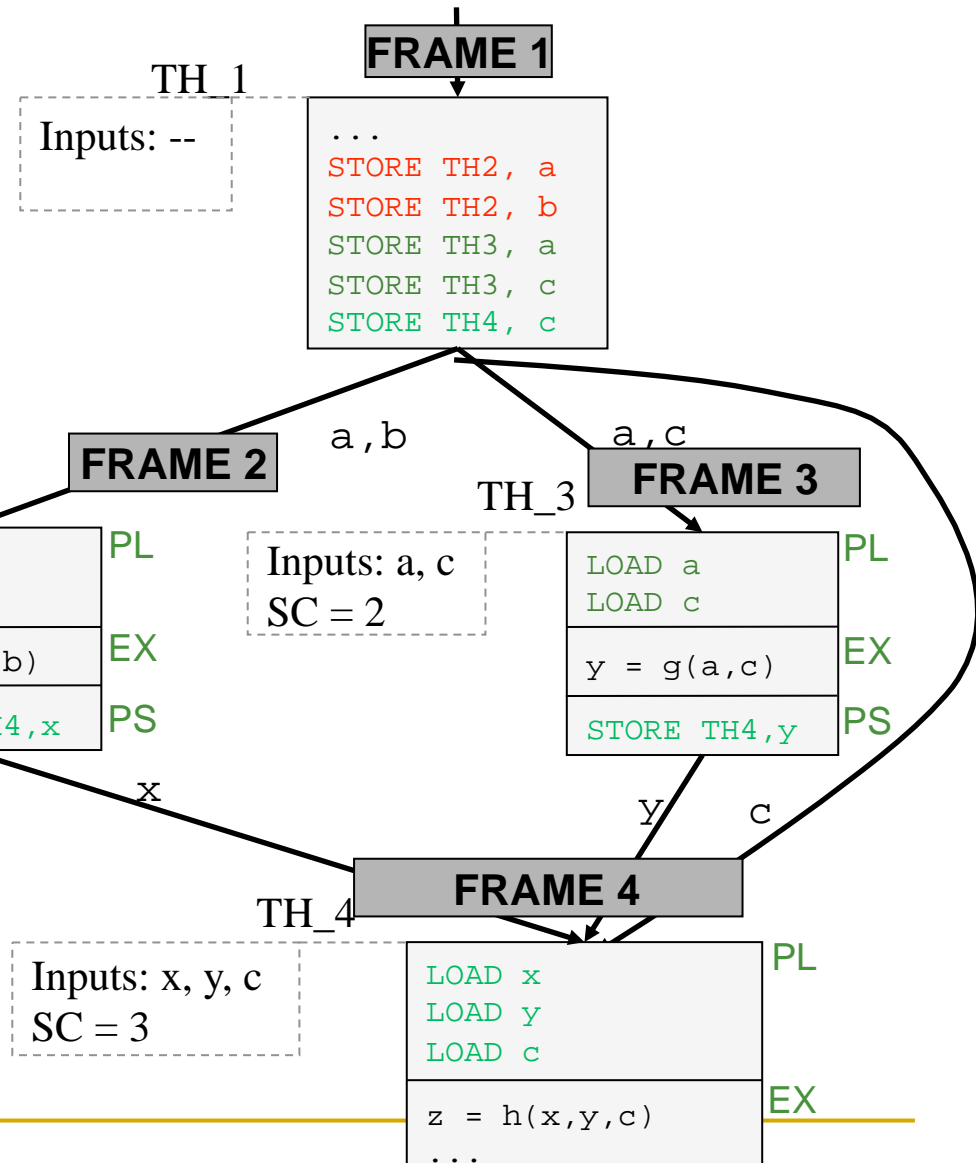
- Based on Cell processor model
- CellSim/SARCSim
- Kept SPU ISA with DTA-specific additions



Threads in DTA

Thread TH_1

- Has no input data (hence SC = 0)
- Sends data to threads 2, 3 and 4
 - STORE instruction is used for storing data in per-thread frames



Thread TH_2

- Needs *a* and *b* as its input (hence SC = 2)
- When they arrive from thread TH_1 SC will reach zero
 - Thread TH_2 can start executing
 - LOAD instructions are used for getting data from the frame
- Once the execution finishes
 - Thread can write data to other threads

Application of TM to a Novel Execution Model (DTA)

- Is TM more efficient in TLP-oriented Architectures ?
 - *Recent implementations bet on that (e.g. SUN Rock)*
 - GHC (compiler) already provides a software TM implementation
 - in SARC (FP6-IP) (we) implemented a TLP infrastructure (DTA) for the UNISIM (simulator)
 - HIPEAC-1 “mini-cluster” (UPC+SIENA) explored GCH-IR (“core”) to DTA mapping
 - Direction of collaboration/investigation:
 - Find reasonable tradeoff between HW and SW support
 - Synchronize among threads using coarse-grained dataflow
 - Overhead reduction through read/write set prediction
-

“Counter” example

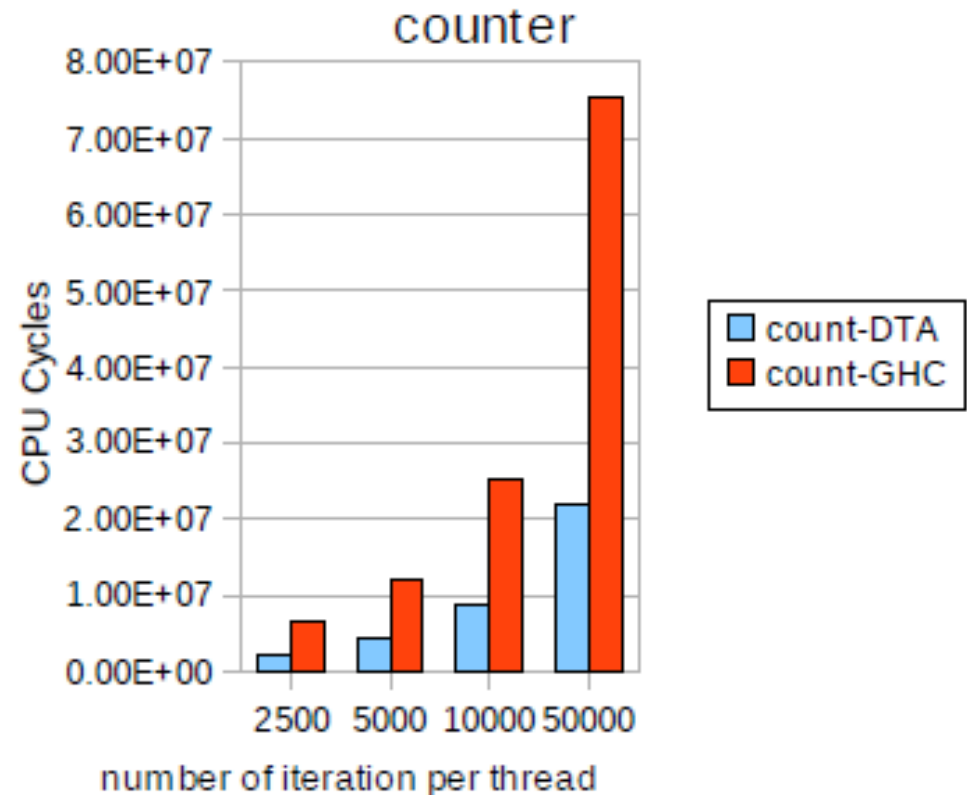
```
module Main where

import GHC.Conc

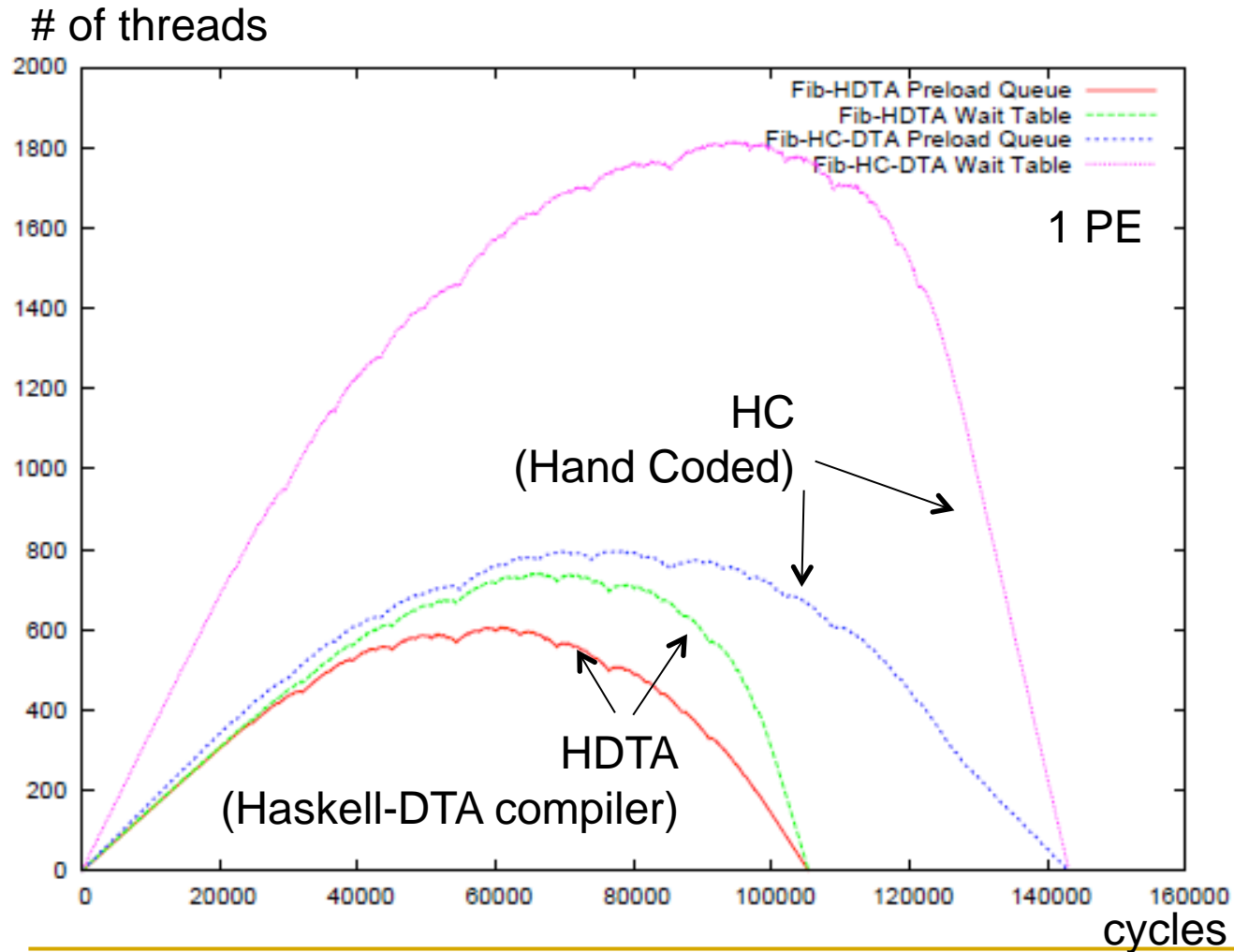
incTVar :: TVar Int -> STM ()
incTVar a = do
  tmp <- readTVar a
  writeTVar a (tmp+1)

loop n f = mapM_ (\_ -> f) [1..n]

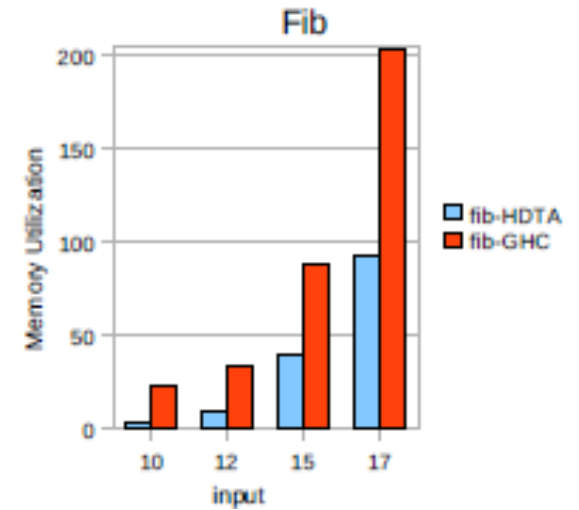
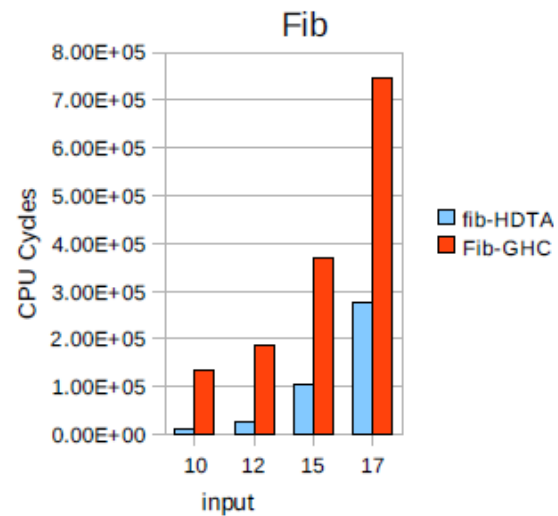
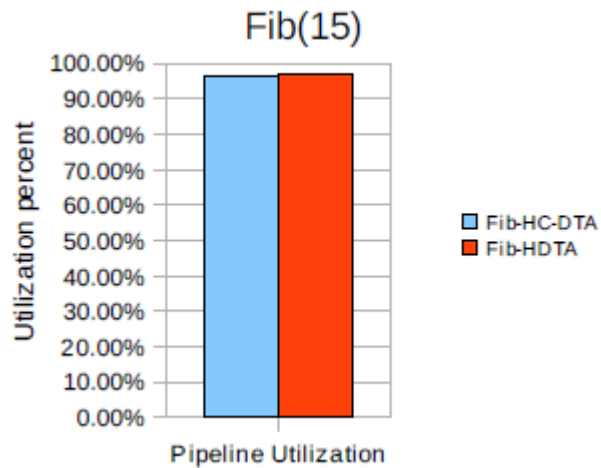
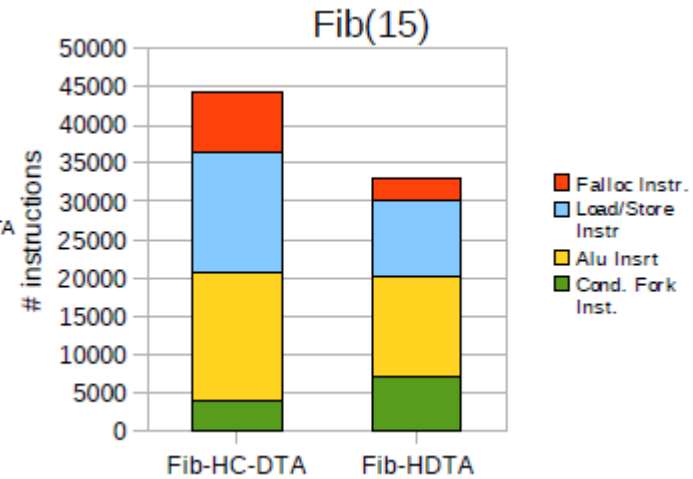
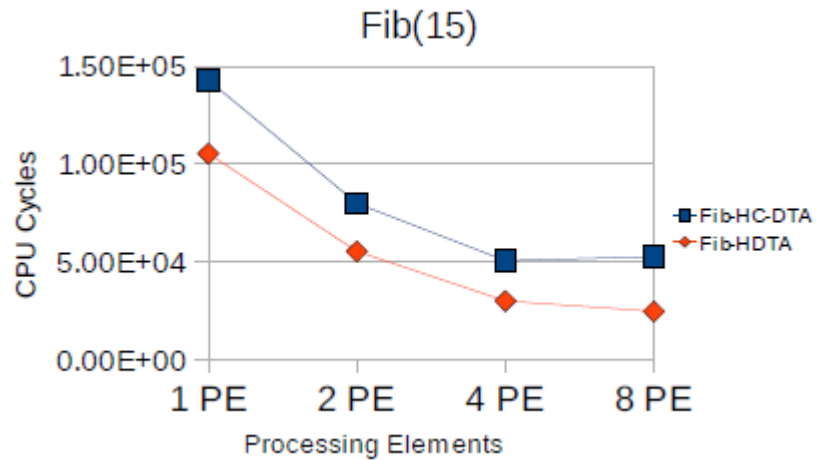
main = do
  x <- newTVarIO 0
  n <- return 2500
  join1 <- newEmptyMVar
  join2 <- newEmptyMVar
  forkIO ( do
    loop n (atomically (incTVar x))
    putMVar join1 ())
  forkIO $ do
    loop n $ do
      atomically (incTVar x)
      putMVar join2 ()
  takeMVar join1
  takeMVar join2
  tmp <- atomically (readTVar x)
  print tmp
```



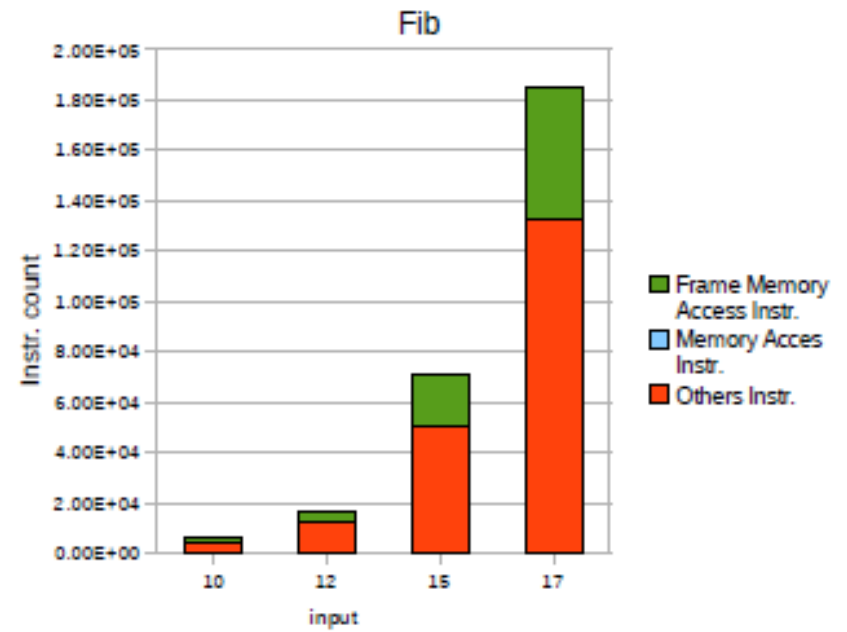
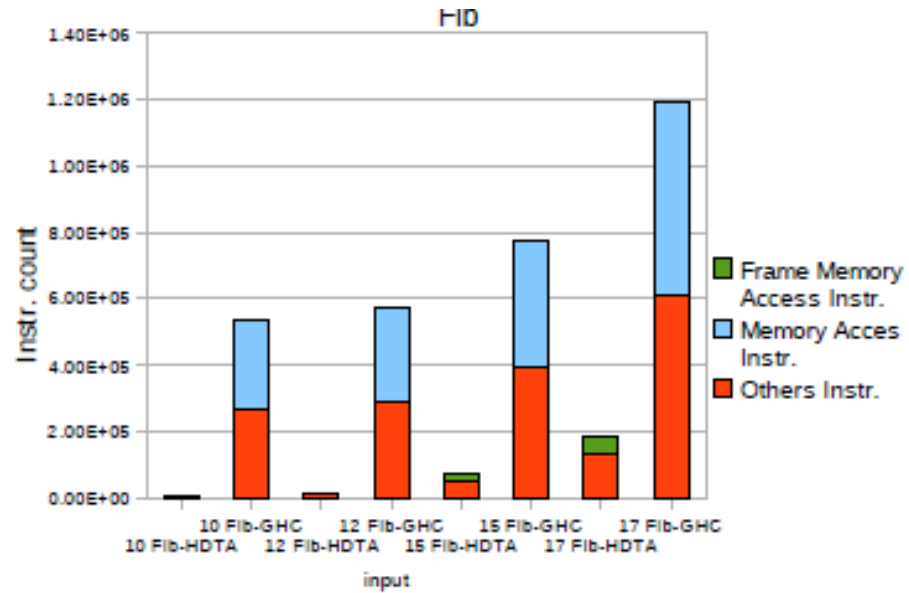
Fibonacci(15): Queued/Waiting Threads



“Fibonacci” example



“Fibonacci” example



References

- [Kavi01a] K. M. Kavi, R. Giorgi, J. Arul, "Scheduled Dataflow: Execution Paradigm, Architecture, and Performance Evaluation", IEEE Trans. Computers, ISSN:0018-9340, Los Alamitos, CA, USA, vol. 50, no. 8, Aug. 2001, pp. 834-846, doi [10.1109/12.947003](https://doi.org/10.1109/12.947003).
- [Giorgi07a] R. Giorgi, Z. Popovic, N. Puzovic, "DTA-C: A Decoupled multi-Threaded Architecture for CMP Systems", Proc. IEEE SBAC-PAD, ISBN:0-7695-23014-1, Gramado, Brasil, Oct. 2007, pp. 263-270
- [Giorgi07c] R. Giorgi, Z. Popovic, N. Puzovic, "Memory access decoupling in a multithreaded architecture", (First Italian Workshop on Real Time Embedded Systems) WIRTES 2007, Pisa, July 2007
- [Giorgi09a] R. Giorgi, Z. Popovic, N. Puzovic, "Introducing hardware TLP support for the Cell processor", *Proc. IEEE Int.I Workshop on Multi-Core Computing Systems*, Fukuoka, Japan, March 2009
- [Giorgi09b] R. Giorgi, Z. Popovic, N. Puzovic, "Exploiting DMA mechanisms to enable non-blocking execution in Decoupled Threaded Architecture ", *Proc. IEEE Int.I Workshop on Multithreaded Architectures and Applications*, Rome, Italy, May 2009

- More to come on <http://www.dii.unisi.it/~giorgi/papers>

Thanks for your attention!