Core Design and Scalability of Tiled SDF Architecture

Roberto Giorgi, Zdravko Popovic

Dept. Of Information Engineering, University of Siena, Via Roma 56, 53100 Siena, Italy

Abstract

Embedded systems are using more extensively multi-core chips to reach high performance goals. While current systems contain only a few cores, present trends and commercial/research roadmaps foresee that in a near future many cores will be integrated on the same chip to achieve the best tradeoff between power consumption and performance. At the same time, centralized designs are progressively abandoned in favour of more modular and scalable approaches that address explicitly wire delay problem and aim to exploit application parallelism. Such designs are often referred as tiled architectures. Here we present our idea how the tiled paradigm can be applied on the SDF architecture.

KEYWORDS: multiprocessor architectures, scalability, wire delay, chip multiprocessors

1 Introduction and motivation

The main reason for creating tiled architectures is to achieve better scalability than current modern Chip Multiprocessors, while addressing wire-delay [1] [2] [3]. One of the main limitations to scalability is wire delay. In the recent past there were several proposals for tiled architectures, such as: Raw [4], Smart Memories [5], Synchroscalar [6], TRIPS [7], and WaveScalar [8]. These proposals are different in terms of number and type of tiles, computational power of processing elements, memory organization, interconnection network and programming model.
Our proposal is based on the SDF architecture [9] [10]. SDF exploits a simple paradigm that is based on dataflow and multithreading. Program is partitioned into non-blocking execution threads and all memory accesses are decoupled from program execution. Each thread is partitioned into preload phase (data that is needed in execution is fetched into the local registers), execution phase (thread is executed without any memory access) and post-store phase (results are stored back into the memory). In this way, decoupling memory access from execution is achieved. There are two pipeline types in the architecture, one for preload and post-store thread phases (synchronization pipeline) and one for execution phase (execution pipeline).
We have performed some tests in order to demonstrate scaling possibilities of the SDF with large number of pipelines, and used two hand-coded benchmarks for that: bit-count from MiBench and radix sort from SPLASH2. They showed that when working with up to 256 pipelines, SDF can achieve almost linear scaling (figure 1). The purpose of these experiments was to see if limitations to scalability were coming from the proposed

architecture or from the program. We found that both for Mibench-bitcnt and SPLASH2-radix the proposed architecture can scale of a factor at least 100x.
Saturation for bitcount occurs because the loop unrolling was done for 64 registers in a register set. If we have supposed bigger number of registers, even higher scalability could have been achieved. The scalability of radix sort algorithm can be improved by forking more threads for sorting and merging the array. In this case we forked 1024 threads, and saturation is reached for this number of pipeline pairs (it is almost linear up to 200, and above it's not because there are prts of algorithm that can't be parallelized).

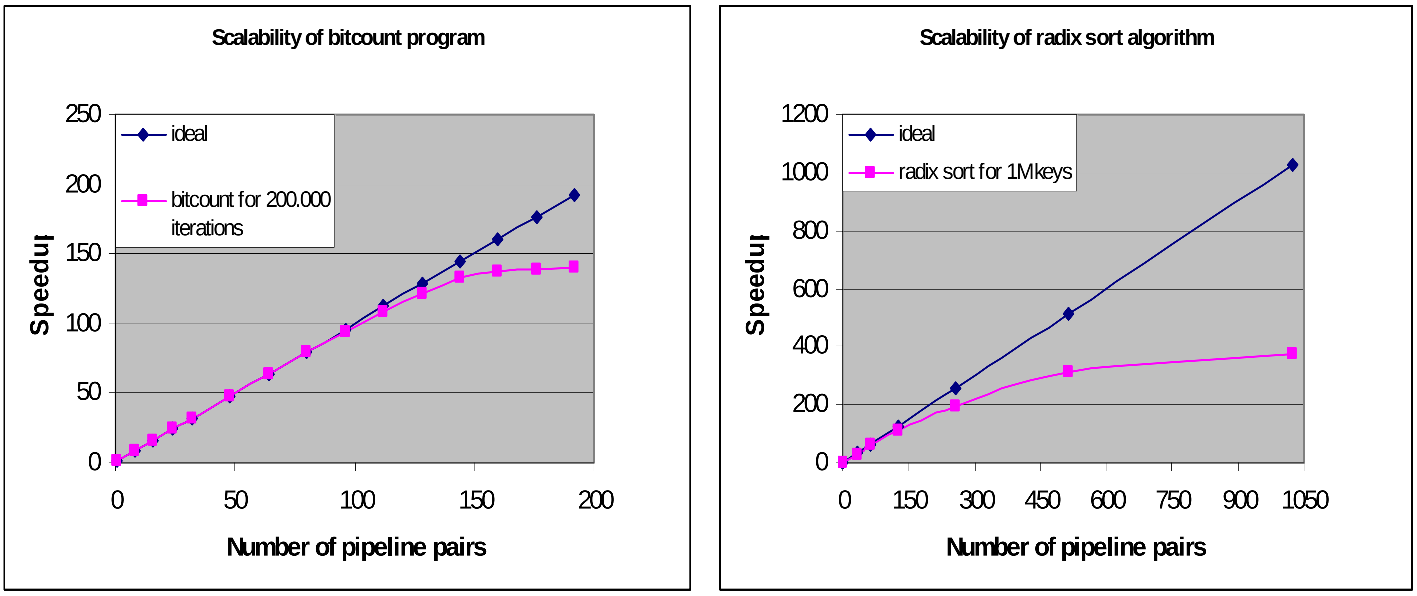

Figure 1. - Scalability of the SDF architecture (speedup is calculated by measuring the number of cycles needed for program execution)

2 The proposal

In the basic SDF architecture if we have a lot of pipelines, they can't communicate with the memory all in one cycle. That is one of the reasons why we have to clusterize the resources in some way to exploit the parallel potential of the architecture, besides the classical wire-delay problem. We propose a new tiled architecture based on the SDF architecture [11]. We want to have a lot of threads running in parallel, while keeping most of the data accesses to local resources, not the shared one. Programs can remain the same as in the basic SDF architecture.
The whole chip is divided into clusters (figure 2). Every cluster is the same and can be considered as high level tile. Inside each cluster there are two (low level) tile types, one execution and one control. Control tile, we call it the **D**istributed **S**cheduler (DS), takes care of scheduling the threads onto relatively small group of execution tiles, and everything else is done in the execution tile. Control tile consists of **F**alloc **R**equest **Q**ueue (FRQ) with all the threads that wait to be assigned for execution onto some tile, **F**ree **F**rame **T**able (FFT) with the number of available frames in each execution tile of the cluster and control logic. This tile can communicate also with other DS tiles, and eventually schedule some threads on the other clusters.

Execution tiles are similar to SDF processors, and they are able to completely execute threads. In each of the execution tiles there is a local scheduler, which takes care about continuation management once the thread is scheduled to the tile. Each thread has its dedicated frame (portion o f memory in execution tile) from which it reads the input data. New thread is created with the FALLOC instruction.

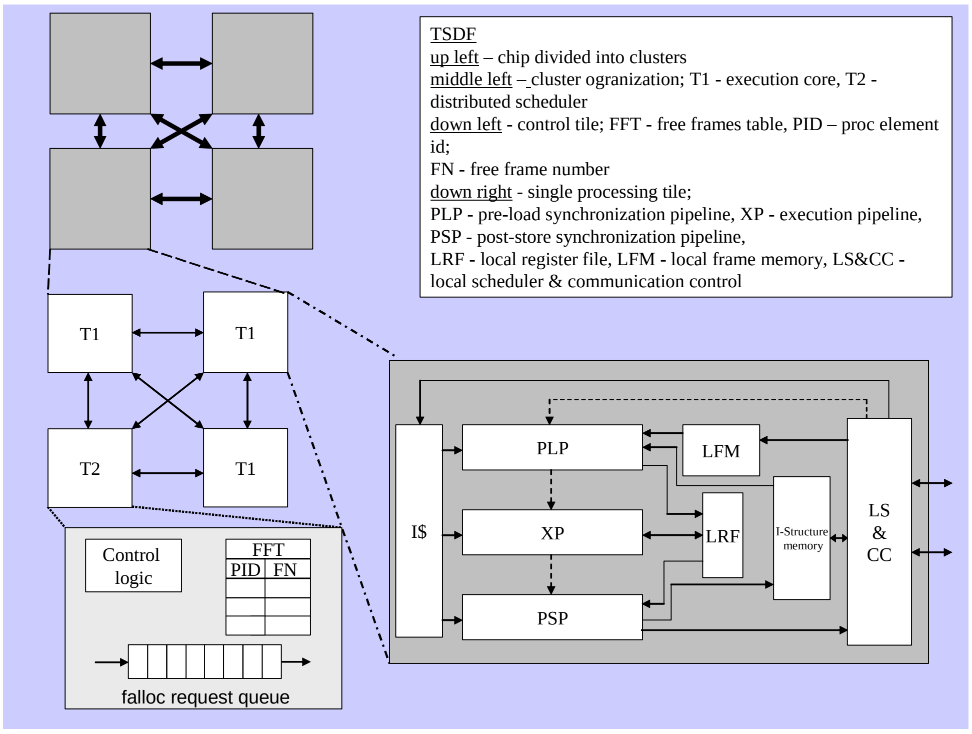
TSDF
up left – chip divided into clusters
middle left – cluster ogranization; T1 - execution core, T2 - distributed scheduler
down left - control tile; FFT - free frames table, PID – proc element id;
FN - free frame number
down right - single processing tile;
PLP - pre-load synchronization pipeline, XP - execution pipeline, PSP - post-store synchronization pipeline,
LRF - local register file, LFM - local frame memory, LS&CC - local scheduler & communication control
T1
T1
T2
T1
Control logic
FFT
PID
FN
falloc request queue
I$
PLP
XP
PSP
LFM
LRF
I-Structure memory
LS & CC

Figure 2. – Architecture

3 How the machine works

Here we briefly describe algorithms of both Distributed and Local schedulers.
Distributed scheduler serves to forward messages from one processing element to another and to make decision where the new thread is going to be executed. When the DS receives new Falloc Request Message it
looks up for available frames in the local cluster. If there is any, forwards the message to the chosen processing element. In all of them are busy, it puts the message in FRQ and send broadcast message to all other clusters. Message can be removed from the queue in two cases: 1) when Ffree Message arrives; this means that one of the local PEs has finished the execution of some thread and therefore has the resources available and 2) when Broadcast Response Message arrives from some other cluster that can execute new thread. In both cases DS just forwards the message from the FRQ. Other messages (Data Response Message and Falloc Response Message) are simply forwarded to their destination.
Local scheduler operates in a simple manner. When it receives a Falloc Request Message, dedicates one free frame for the execution of the new thread and sends Falloc Response Message (that contains frame number of dedicated frame) to the processing element that issued this request. When Falloc Response Message is received it stores the frame number in the destination register of the Falloc instruction. Data Response Message is sent for each store instruction (can be grouped for the same destination) and upon the reception of this

message sent data is stored in the destination frame. If the destination frame is in the same processing element, no message is sent and the store is done locally. When all the data is stored, thread can start execution.

Each tile in the architecture has its unique address which consists of pair (cluster id – CID, processing element id – PID). Also, each frame number in the architecture is unique, and defined by the triplet (CID, PID, frame number – FN). This simplifies the routing of the messages in the way that the messages are sent to the intra-cluster network just if CID field is different from the local CID.

4 Future work

Further research will proceed on several issues. One is to investigate the impact of the structure of the execution tiles regarding the number of pipelines inside, size of frame memory and other structures. Other is to search for optimal number, structure and usage of the messages in the system. Also, we will try to find the placement of elements inside a single core which minimizes used area.

References

[1] R. Ho, K. Mai, M. Horowitz, "The future of wires", Proceedings of the IEEE, Vol. 89, No.4, pp. 490--504, 2001.

[2] B. M. Beckmann, D. A. Wood, "Managing Wire Delay in Large Chip-Multiprocessor Caches," Proceedings of the 37th annual IEEE/ACM International Symposium on Microarchitecture, pp. 319-330, 2004.

[3] S.W. Keckler, Doug Burger, C.R. Moore, R. Nagarajan, K. Sankaralingam, V. Agarwal, M.S. Hrishikesh, N. Ranganathan, P. Shivakumar, "A Wire-Delay Scalable Microprocessor Architecture for High Performance Systems," *International Solid-State Circuits Conference (ISSCC)*, pp. 1068-1069, February, 2003.

[4] M. B. Taylor, J. Kim, J. Miller, D. Wentzlaff, F. Ghodrat, B. Greenwald, H. Hoffmann, P. Johnson, Jae-Wook Lee, W. Lee, A. Ma, A. Saraf, M. Seneski, N. Shnidman, V. Strumpen, M. Frank, S. Amarasinghe, A. Agarwal, "The Raw Microprocessor: A Computational Fabric for Software Circuits and General Purpose Programs," IEEE Micro vol.22 Issue 2, pp. 25-35, Mar/Apr 2002.

[5] K. Mai, T. Paaske, N. Jayasena, R. Ho, W. J. Dally, M. Horowitz, "Smart Memories: A Modular Reconfigurable Architecture," 27th Int'l Symp. on Computer Architecture, pp. 161-171, ACM Press, June 2000.

[6] J. Oliver, R. Rao, P. Sultana, J. Crandall, E. Czemikowski, L. W. Jones IV, D. Franklin, V. Akella, F. T. Chong, "Synchroscalar: A Multiple Clock Domain, Power-aware, Tile-based Embedded Processor," 31st Int'l Symp. on Computer Architecture, pp. 150-161. IEEE CS, June 2004.

[7] K. Sankaralingam, R. Nagarajan, H. Liu, C. Kim, J. Huh, D. Burger, S. W. Keckler, C. R. Moore, "Exploiting ILP, TLP and DLP with the Polymorphous TRIPS Architecture," 30th Int'l Symp. on Computer Architecture, pp. 422-433. ACM Press, June 2003.

[8] S. Swanson, K. Michelson, A. Schwerin, M. Oskin, "WaveScalar," in the 36th Proc. Int'l Symp. on Microarchitecture (MICRO-36), pp. 291-302. IEEE Press, Dec. 2003.

[9] Krishna M. Kavi, Roberto Giorgi, Joseph 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.

[10] K. Kavi, J. Arul, R. Giorgi, "Execution and Cache Performance of the Scheduled Dataflow Architecture," *SPRINGER Journal of Universal Computer Science*, ISSN:0948-6968, New York, NY, (USA), vol. 6, no. 10, Oct. 2000, pp. 948-967, Special Issue on Multithreaded Processors and Chip Multiprocessors.

[11] S. Bartolini, R. Giorgi, E. Martinelli, Z. Popovic, "Recent Proposals for Tiled Architectures," ACACES 2005 POSTER ABSTRACTS, L'Aquila, Italy, July 2005, pp. 47-50.