Recent Proposals for Tiled Architectures

Sandro Bartolini, Roberto Giorgi,
Enrico Martinelli, Zdravko Popovic

Dpt. Ingegneria delll'Informazione, University of Siena, Via Roma 56, 53100, Siena, Italy

ABSTRACT

One of the most interesting research questions for the microarchitecture community is: "How can we effectively use the increasing number of transistors available on a single chip while avoiding wire delay problem?" The time needed for a signal to reach the opposite edge of a chip is becoming longer than one cycle, and because of this, it becomes hard to gain more performance improvement with the scaling of superscalar architectures. One viable solution for using all the available transistors efficiently and effectively, while hiding wire delay as much as possible is to parallelize resource usage through resource clustering and decoupling. Recently a good number of tiled/clustered architectures have been proposed, indicating that this field is gathering high interest from both academia and industry.

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

1 Introduction

The natural way for improving the uniprocessor systems is connecting several uniprocessor systems with some kind of interconnection to work together. This is the main idea of all multiprocessor systems. As the technology was improving, it became possible to place more than one processor on a single chip, and therefore chip multiprocessors became reality. One type of chip multiprocessors are tiled architectures. Tiled architectures have some number of tiles that are replicated on the chip and connected with on-chip network. Using this approach, design is more simpler, because with technology improving just new tiles are added, and everything else (computational model, programing model, interconnection, memory organization) can stay the same.

The focus of this poster is to present some recent proposals that employ the tiling paradigm at different extents, in a comparative fashion, and highlight their main features and advantages. Architectures that are presented in this poster are (in alphabetical order): Raw [1] (MIT), Smart Memories [2] (Stanford University), Synchroscalar [3] (University of California, Davis and Polytechnic State University, San Luis Obispo), TRIPS [4] (University of Texas at Austin) and WaveScalar [5] (University of Washington), CDE [6] (Universitat Politecnica de Catalunya). In the second part, we present our idea how to design tiled architecture based on the SDF [7] architecture.

2 Main Idea of the Proposed Architectures

The Raw architecture tries to overcome wire delay problem by tiling resources on the chip into some number of equal processing cores and making that no wire is longer than the

length or width of a tile. Its ISA allows programmer to have an effective control over the communication hardware between tiles and towards off-chip modules.
Smart Memories is a modular reconfigurable architecture that can alter its wires, memory and computational model to adapt to the type of application it executes. It's tiles can be processing or memory.
Synchroscalar performs non-homogeneous voltage and frequency scaling of different tile sets to achieve the lowest power consumption, while reaching the performance targets. There are processing and control tiles.
TRIPS tries to avoid wire and memory latency problems by tiling processing cores and placing more memory on the chip, and to adapt its architecture in order to exploit different types of parallelism like, instruction, thread and data level parallelism. Processing cores and on-chip memory are reconfigurable to achieve this.
WaveScalar has a huge number of simple processing elements which communicate operands in a way to employ dataflow execution model. They are grouped in tiles, with the memory on the edges of the chip. It exploits dataflow locality through static and dynamic prediction of instruction dependencies in the dynamic trace of the application.
CDE consists of one Epoch Control Processor (EPC) and a grid of MIPS R2000-like processing tiles. Compiler partitions program into two levels of hierarchy - Control Epoch which are large code segments, each containing several Dependence Clusters, which are chains of dependent instructions. Processing tiles execute individual DC threads, and EPC fires these DCs by processing epochs. This architecture is still in the early stage of development.

3 Advantages and Disadvantages

Common advantage of all presented architectures is good scalability.
Synchroscalar architecture addresses mainly communication and multimedia applications, while others are designed with idea to be used as a general purpose architectures. Synchroscalar has good power saving capabilities that are close to ASIC design and still it provides the flexibility of the DSP. It doesn't achieve the highest performance possible because of the power efficiency, but in embedded systems power characteristics can be of bigger importance.
Raw and TRIPS have shown good performance over wide range of workloads with different types of available parallelism. The advantage of Raw is that it doesn't need any hardware reconfiguration to achieve this. TRIPS, like Smart Memories, require its hardware to be reconfigured, but thanks to this they can adapt to many different workloads and to have maximum performance. Through its ISA, Raw allows programmers to access gates, wires and pins and to try to achieve better performance and power efficiency. In TRIPS there are still open questions about interface between software and reconfigurable hardware. Smart Memories chip can be connected with other chips of same kind as a part of a wider multiprocessor system.
WaveScalar has also shown good initial performance, but it is still in the early phase of development and there are some open questions (interrupts handling, I/O). Its main advantage is that it is a dataflow hardware that runs programs that are written in standard programming languages.

4 Tiled SDF (TSDF)

We propose a new tiled architecture based on the SDF architecture. Our idea is to apply tiling paradigm to the SDF execution model. We want to have several threads running in parallel, while keeping most of the data accesses to local resources, not the shared one. Programs can be the same as in basic SDF architecture.

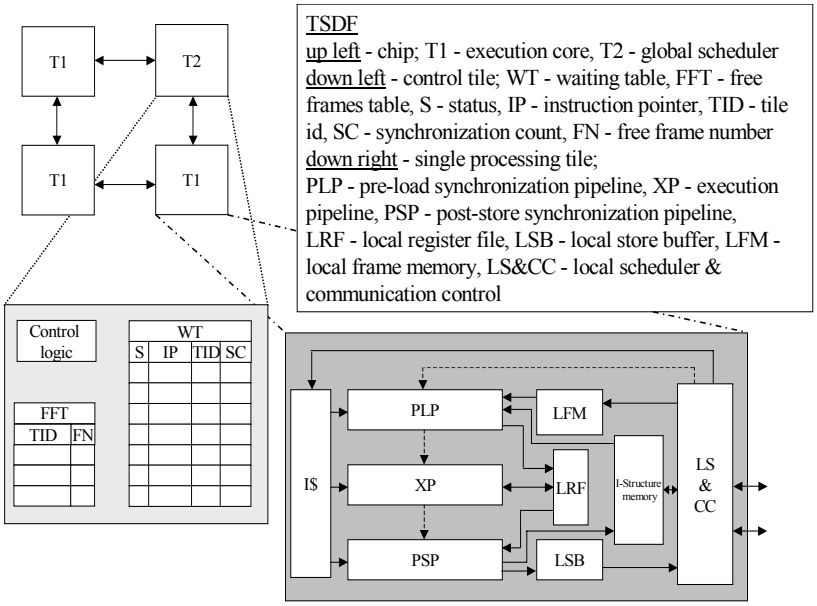

Figure 1. – TSDF architecture

There are two tile types, one execution and one control. Control tile, we call it the Global Scheduler, takes care of scheduling all the threads onto execution tiles, and everything else is done in the execution tile. 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.

Tiles communicate data through the network, but its organization is still an open issue. Control tile can send broadcast message to all execution tiles.

Algorithm of Global Scheduler:

1 GS issues a thread of the program to the first available tile, by looking up Free Frames Table, and assigns an entry in the Waiting Table (Thread_Id, Status, Instruction_Pointer, Synchronization_Counter, Tile_Id).
2 For each FALLOC request (IP, SC) assigns new entry in the WT with Status = Waiting, and Tile_id = Unknown. Thread_Id is equal to the number of entry. As a FALLOC response sends, Thread_Id and IP.
3 On each STM (STORE Message, see bellow) decrements SC of the entry with matching Thread_Id.
4 If some SC reaches zero, GS decides on which tile to map the thread, sends message to that tile (Tile_Id, Thread_Id, IP, SC), changes status to Executing, and writes Tile_Id in the field of the WT entry. Also, sends broadcast message to all the tiles with Tile_Id and Thread_Id.
5 For each FFREE instruction, changes the Status of the WT entry with matching Thread_Id to FREE.

Algorithm for Local Scheduler:

1 Forwards each FALLOC from execution stage to the GS, and waits for the Thread_Id in response.
2 STORE instructions in the post-store stage are grouped by the Thread_Id, and for each Thread_Id one STM is sent to the GS with Tile_Id, and SC (number of data stored for

that thread). Also, every STORE is stored in the Local Store Buffer (Thread_Id, Offset, Data).

3 When receives thread assign it allocates one frame in the Local Frame Memory for that thread, loads the code block into I$, waits for the number of received data to reach SC (some data can be local), and then enables the thread. Thread then goes to pre-load, execute and post-store stage.

4 If the tile receives data request message it checks if there are data with the same Thread_Id in the LSB, and if there are, sends the data, and frees the LSB entry.

5 After the thread is executed FFREE message is sent to GS.

6 If there is I-Structure access to the remote tile, it sends the request.

7 On I-Structure request, checks if the address is local and if it is, send the data response.

From these algorithms we can see that all the loads in the pre-load stage are to the local memory. Stores in the post-store stage are to the LSB, and all these data, at some point, are sent to the LFM of the local or remote tiles.

5 Future work

Here we presented our idea how the TSDF architecture can be organized. Next step will be evaluation. With this approach we try to achieve good scalability and avoid wire delay problem, while keeping the SDF execution model. There are still some things that are not precisely defined, like network connection between tiles. Also there is a lot of space for optimization and improvement, like algorithm for global scheduler that would minimize communication, or optimal number of pipelines in each tile.

References

{1] 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

[2] 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

[3] 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

[4] 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

[5] 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

[6] Carmelo Acosta, Sriram Vajapeyam, Alex Ramirez, and Mateo Valero. CDE: A Compiler-driven, Dependence-Centric, Eager-executing Architecture for the Billion Transistors Era. Intl. Workshop on Complexity-Effective Design (WCED'03). Tokyo (Japan), June 2003.

[7] Krishna M. Kavi, Roberto Giorgi, Joseph Arul, "Scheduled Dataflow: Execution Paradigm, Architecture, and Performance Evaluation", IEEE Trans. Computers, vol 50, no.8, Aug. 2001, pp. 834-846.