One-Chip *TeraArchitecture*

Gheorghe Ştefan
BrightScale, Sunnyvale, CA
gstefan@brightscale.com

Abstract

The distinction between the *complex* computation and the *intense* computation is presented in order to introduce **TeraArchitecture**, a high-performance architecture which integrates all kinds of parallelism in order to optimize area & power. Multi-processors are efficient for multi-threaded complex computations, while many-processors are efficient in data intense computations. Combining them results an efficient solution for the *"Tera Computing Era"*.

Keywords: parallel computing, computer architecture, multi-processors, many-processors, power-aware design.

1 Introduction

Computation manifests in two very distinct ways: as **complex** computation and as **intense** computation. In the mono-processor era this distinction generated two kinds of very different machines: the general purpose complex processors and the simpler specialized, but very efficient, signal processors. In the parallel computing era the relation between the complex computation and intense computation changes, and more, it can and must be used to optimize the parallel computing machines.

We introduce here an optimal parallel architecture for the *"Tera Computing Era"*, based on the **strongest segregation** between the *complex parallel computation* component and the *intense parallel computation* component. Tera Power means: the *sum of floats and integer* (word, half-word, byte) *instructions* (mean weight of floats are 10%, but no bigger than 25%) to be in the range of $2^{40} \simeq 1.1 \times 10^{12}$.

We start emphasizing the distinctions between "complex" and "intense" parallel computation and continue with the new proposed architecture and the associated organization. Results a computer organization which optimizes *area, power, price, performance* and *programmability*. The last section will present details about the intense part of the architecture and the associated organization.

2 Complex vs. Intense

The actual trend in the parallel computing R&D must make the fundamental distinction between **multi-processors** and **many-processors** [2]. Roughly speaking:

- a multi-processor consists in more than one ***big*** *& complex* processor (no more than few dozens) performing *multi-threaded executions*
- a many-processor is an array of thousands of ***small*** *& simple* machines performing efficiently *vector and/or stream processing.*

Multi-processors are efficient performing complex parallel computations, while many-processors are good in solving data intensive problems. In [8] is presented how the size of the processors decrease when the number of processors increase in order to keep area efficient a network of processing machines.

Complex computation is characterized by the following features and figures (we use as a case study a $65nm$ ***Intel***-like implementation):

- mono or multi *big & complex* processor organization
- multi-threaded programming model
- operating system oriented design
- memory hierarchy is cache-based
- peak performance[1] for a super-scalar mono-processor (with: $1.5cm^2$, $50Watt$, $\sim 2GHz$): *4 GIPS + 2 GFLOPS*

[1] GIPS stands for **G***iga 32-bit* **I***nstruction* **P***er* **S***econd*, and GFLOPS stands for **G***iga 32-bit* **FL***oating point* **O***perations* **P***er* **S***econd*

- *(0.027 GIPS + 0.013 GFLOPS)/mm^2*
- *(0.08 GIPS + 0.04 GFLOPS)/Watt*

Intense computation is characterized by the following features and figures (based on the $65nm$ ***BrightScale*** implementation):

- many *small & simple* cell organization
- array (vector and/or stream) computing
- high-latency functional pipe oriented system
- the memory hierarchy is multi-buffer oriented
- peak performance[2] (with: $0.64cm^2$, $10Watt$, $0.4\,GHz$):
  400 GOPS
  which can be distributed between integer operations and floating-point operations (floating-point operations are performed executing 30 16-bit operations), *for example* as:
  (220 GOPS + 6 GFLOPS)
  for the low-intensive float (the weight of floating-point operations is 3%), or as
  (40 GOPS + 12 GFLOPS)
  for the high-intensive float (the weight of floating-point operations is 23%)
- 6.25 *GOPS/mm^2*
  or
  (3.44 GOPS + 0.094 GFLOPS)/mm^2
  for the low-intensive float, or as
  (0.625 GOPS + 0.19 GFLOPS)/mm^2
  for the high-intensive float
- *40 GOPS/Watt*
  or
  (22 GOPS + 0.6 GFLOPS)/Watt
  for the low-intensive float, or as
  (4 GOPS + 1.2 GFLOPS)/Watt
  for the high-intensive float

Intense vs. complex by numbers It is obvious that the two kinds of computation are separated by more than one order of magnitude when performances, per mm^2 or per *Watt*, are compared. See Table 1 for the area use by an intense architecture versus the area use by a complex architecture. See Table 2 for the power use by an intense architecture versus the power use by a complex architecture.

[2](where: GOPS stands for **G***iga 16-bit* **O***perations* **P***er* **S***econd)*

Type	(intPerf/mm^2)/(compPerf/mm^2)
no float	116
low float	52
high float	13

Table 1: **Intense computation vs. complex computation in area use.** The performance per mm^2 for intense computation (intPerf/mm^2) is compared against the performance per mm^2 for complex computation (compPerf/mm^2).

Type	(intPerf/Watt)/(compPerf/Watt)
no float	250
low float	110
high float	26

Table 2: **Intense computation vs. complex computation in power use.** The performance per *Watt* for intense computation (intPerf/Watt) is compared against the performance per *Watt* for complex computation (compPerf/Watt).

Because the "price" of complexity is very high, **separating "the complex" from "the intense" results to be the best way to optimize a computing system**. Reducing "the complex" to "the intense" is not possible because of the big difference in the programming model (multi-threaded compared against array-oriented). Reducing "the intense" to "the complex" is already performed with very costly (big area and huge power) solutions (see, for example, the *Intel & NVIDIA* mixture).

Because, by programming, GOPS can be converted in GFLOPS, a many-processor architecture can be adapted to different weights of flops. The relation

$$n = t(1 + w(k - 1))$$

tells us how many GOPS, n, are needed to a targeted sum t, of flops and simple operations, when the weight of flops is w, and a float operation is performed by k simple, one cycle, operations.

Example 1 *Let be an intense float application with the weight of floats $w = 0.2$, in a many-processor executing only simple, one-cycle operations. For a floating-point operation, each processing element uses a sequence of $k = 20$ simple operations. How big must be the performance, n, expressed in simple, one-cycle TOPS in order to obtain an overall performance of $t = 1\ T(OP + FLOP)S$, composed*

by 0.8 TOPS of one-cycle simple operations and 0.2 TFLOPS?

$$n = 1(1 + 0.2(20 - 1))\ TOPS = 4.8\ TOPS$$

Therefore, if the application domain is float-intense, then a Tera Machine must have the "brute force" performance of 4.8 TOPS.
◇

Example 2 *Let us evaluate w and t for the intense float application domain of* **graphics**, *using the well known "graphics Bible" of* Foley & van Dam [5] *(see* section 18.3.9*). We consider the following case: average coverage per triangle: 100 pixels; one-half of the pixels of all the triangles are obscured by some other triangles; we assume (together with* [5]*) no primitives needed for clipping (minimizes the calculation for clipping but maximize the work required in the succeeding stages); ambient and diffuse illumination models; 1920 by 1080 display; 30 frames per second.*

The resulting requirements are: $(370\ flop + 4 \times 370\ op)/triangle + (4.25\ add + 4 \times 4.25\ op)/pixel + 2\ memOp/pixel$, *where:* flop *stands for floating-point operations,* op *stands for simple, one-cycle arithmetic, logic or control operations,* add *stands for one-cycle integer addition, and* memOp *stands for the frame-buffer access operation.*

For these requirements results: $w = 0.195$, *for* 10^6 *triangles. If the many-processor architecture asks for 30 cycle/float* ($k = 30$), *and the graphic application manages* 10^6 *triangles per frame, results*

$$n = 366\ GOPS.$$

Then, with a 400 GOPS machine the graphics above defined can be performed.
◇

3 TeraArchitecture

The proposed parallel computing machine consists of:

- a coarse grain network of m big & complex processors (P), called **C**oarse-Grain Multi-**P**rocessor (CiP); that performs multi-threaded execution;
- a fine grain cellular machine, having n small and simple execution units (EU) or processing elements (PE), called **F**ine-Grain Man**y**-**P**rocessor (FyP), working under the control of at least one of the following mechanisms:
  - centralized control: the n EUs are controlled by an instruction sequencer (IS)
  - global control: the EUs or PEs are included in a simple global loop
  - local control: all or some of the n PEs are controlled by their own programs

  with:
  - no operating system (the network performs a pipe of functions)
  - no cache memory system (the network uses a multiple buffers system)

  performing array and/or stream processing
- an interconnection fabric & memory & peripherals controllers

interconnected as in Figure 1, where: $n >> m$.

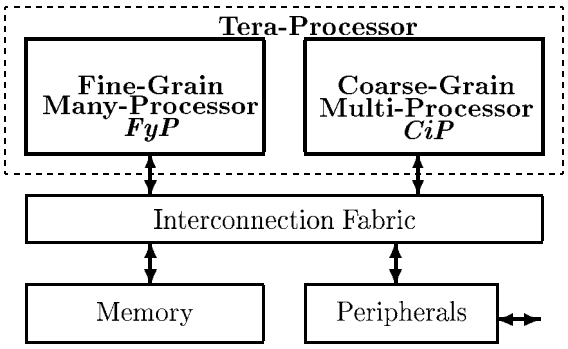

Figure 1: **The Tera-Processor Organization.**

Because $n >> m$, the few P processors are designed to have good or very good *performance*, while the many EU and PE machines are designed to be only *competent*. In P is usual to have multipliers, floating-point units, while EU and PE has hardware resources only for the most frequently used operations, because all the complex operations (multiplication, floats, ...) are done by sequencing simple operations. This means: *keep small and simple what is many times multiplied.*

The network FyP works like an accelerator for the network CiP. FyP is a functionally oriented machine, better programmed using a functional language, while CiP is an object-oriented structure designed to manage big complexities.

Both, general purpose computation and embedded computation are best supported by the previously described architecture, let us call it ***TeraArchitecture***. The associated organization is a ***TeraProcessor*** (TP).

We estimate that in the 65 *nm* technology a TP organization with $n = 1024$ and $m = 4$ can be implemented on a $2.5cm^2$ silicon die. Because FyP is on the same die with CiP the size and complexity of each of the 4 Ps can be reduced. For the same reason the clock frequency can be 1 GHz for the CiP network and 0.4 GHz for the FyP network, resulting an *area & energy* saving design.

4 The Preferred Embodiment

In the current stage of technological development the preferred embodiment for TP is:

- for CiP an *Intel*-like multi-core, with 2 to 8 cores, is the most appropriate
- for FyP there are three *BrightScale* solutions:
  - a linear array, of 256-4096 EUs with a centralized control performed by an IS, an IO Controller, and global control, if needed (EU contains: data memory, file register, arithmetic and logic unit, and a Boolean machine)
  - besides the linear array of EUs, a pipe of PEs is added for stream processing (PE is an EU with its own program memory)
  - a linear array of 256-1024 PEs for both, vector processing and stream processing.

5 Concluding Remarks

1. The functional aspects of computation in the *"Tera Computing Era"* are defined by *recognition, mining and synthesis* (RMS) [3]. *Intel* introduced the terminology of "multi" and "many" [2], but their plan is to promote many-processors only as followers for multi-processors. Our way to TOPS uses the symbiotic approach which integrates "many" with "multi". The theoretical aspects which cover RMS are synthesized in the Berkeley's *13 motifs* [1].

2. TP is a solution for TOPS, due to of the segregation between the complex computation and intense computation [7]. Indeed, using an array of $n = 4096$ EUs for FyP and an array of $m = 4$ cores for CiP, in the 45 nm technology at 1 GHz, results a machine strong enough to open the *"Tera Computing Era"* on one-chip.

3. The complex computation is supported by the standard Turing model used by Flynn [4] to provide its taxonomy. This taxonomy works pretty well in classifying CiP as a MIMD machine.

The intense computation is described by the Kleene's [6] model [7]. FyP contains a data-parallel machine, sometimes a time-parallel machine (featured with speculative-parallel resources) is added, and an array of cells able to perform all types of intense parallel computations can be also possible.

Acknowledgments The authors got a lot of support from the main technical contributors to the development of the *BrightScale* technology, the *BA1024* chip, the associated language, and its first application (frame rate conversion): Emanuele Altieri, Frank Ho, Bogdan Mîţu, Marius Stoian, Dominique Thiebaut, Dan Tomescu, Tom Thomson.

References

[1] K. Asanovic, *et al.*: "The Landscape of Parallel Computing Research: A View from Berkeley," *Technical Report No. UCB/EECS-2006-183*, 2006.

[2] Shekar Y. Borkar, *et al.*: *Platform 2015: Intel Processor and Platform Evolution for the Next Decade*, edited by R. M. Ramanathan and Vince Thomas, *Intel Corporation*, 2005.

[3] Pradeep Dubey: "A Platform 2015 Workload Model: Recognition, Mining and Synthesis Moves Computers to the Era of Tera", *Technology@Intel Magazine*, Feb. 2005.

[4] Michael J. Flynn: "Very High-Speed Computing Systems", in *Proceeding of the IEEE*, 54(12), December 1966, p. 1901-1909

[5] James D. Foley, *et al.*: *Computer Graphics. Principles and Practice*, Addison-Wesley, 1997.

[6] Stephen C. Kleene: "General Recursive Functions of Natural Numbers", in *Math. Ann.*, 112, 1936.

[7] Mihaela Malita, Gheorghe Ştefan: "On the Many-Processor Paradigm", *Proceedings of the 2008 World Congress in Computer Science, Computer Engineering and Applied Computing*, 2008.

[8] Gheorghe Ştefan, M. Malita: "Granularity and Complexity in Parallel Systems", in *Proceedings of the 15 IASTED International Conf, 2004, Marina Del Rey*, CA, ISBN 0-88986-391-1, pp.442-447.