9.6 A Wire-Delay Scalable Microprocessor Architecture for High Performance Systems

Stephen W. Keckler[+], Doug Burger[+], Charles R. Moore[+], Ramadass Nagarajan[+], Karthikeyan Sankaralingam[+], Vikas Agarwal[*], M.S. Hrishikesh[*], Nitya Ranganathan[+], and Premkishore Shivakumar[+]

[+]Department of Computer Sciences, The University of Texas at Austin, Austin, TX
[*]Department of Electrical and Computer Engineering, The University of Texas at Austin, Austin, TX

Contact Author: Stephen W. Keckler
                         1 University Station C0500
                         Department of Computer Sciences
                         The University of Texas at Austin
                         Austin, TX 78712-1188
                         TEL: 512-471-9763
                         FAX: 512-232-1413
                         skeckler@cs.utexas.edu

Speaker: Charles R. Moore
                         1 University Station C0500
                         Department of Computer Sciences
                         The University of Texas at Austin
                         Austin, TX 78712-1188
                         TEL: 512-232-7468
                         FAX: 512-232-1413
                         crmoore@cs.utexas.edu

While microprocessor pipeline depths have increased dramatically over the last decade, they are fast approaching their optimal depth. As shown in Figure 9.6.1, the number of logic levels in modern processors is nearing 10 fanout-of-4 (FO4) inverter delay. Substantial further reductions will be undesirable due to pipeline overheads and power consumption [1]. Technology trends also show that global on-chip wire delays are growing significantly, eventually increasing cross-chip communication latencies to tens of cycles and rendering the expected chip area reachable in a single cycle to be less than 1% in a 35nm technology, as shown in Figure 9.6.2. The challenge for architects is to design new architectures that achieve both a fast clock rate (low FO4) and high concurrency, despite slow global wires. Because existing superscalar microarchitectures rely on global communication, they are poorly matched to the technology challenges of the coming decade.

The Grid Processor architecture (GPA) is designed to address these technology challenges [2]. As shown in Figure 9.6.3, each GPA implementation consists of a 2-D array (4x4 in this example but scalable to larger dimensions) of ALUs connected via a routed operand network, with L1 I-cache, D-cache, and register file banks around the periphery of the ALU array. Each ALU includes an integer unit, a floating point unit, instruction buffers, operand buffers, and an operand router. While [3] proposes ALU chaining similar to the GPA and clustered VLIW architectures have similar partitioning strategies, GPAs permit out-of-order execution and fast clock rates, achieving performance far higher than conventional architectures.

In a GPA program, spatial instruction placement is static but execution order is dynamic. The compiler forms large single-entry, multiple exit regions (hyperblocks) and schedules them to the ALU array. Current hyperblock generation techniques yield instruction blocks consisting of 14-119 (average of 47) useful instructions, each with typically fewer than 10 input and 10 output registers, for the benchmarks shown in Figure 9.6.5 (SPECINT and SPECFP). The instructions along the critical path of the block are mapped to the grid to minimize communication latency by using the short physical paths between adjacent ALUs (Figure 9.6.4) and the bypass path within an ALU. At runtime, the instructions of a block are fetched en masse from the multi-ported instruction cache and distributed horizontally into the grid. Instructions execute in dataflow order, dictated by the arrival time of the operands at each ALU. Intermediate values are routed directly from the producing ALU to the consuming ALU without being written back to the register file. Block outputs are written to the register file at block completion and are dynamically bypassed directly to

instructions in the next block. A next-block predictor speculatively selects subsequent blocks to be mapped and executed while the current block is being executed. Mis-speculations and exceptions cause rollback to the last committed block boundary. Figure 9.6.5 shows the instructions per clock (IPC) currently achieved on an 8x8 GPA, comparing it to the Alpha 21264.

The GPA offers specific technology scaling advantages over conventional architectures. First, it facilitates partitioning in the ALU array, instruction caches, and register files, providing both faster access and higher bandwidth to the memory structures. Second, communication delays in the ALU array are exposed to the compiler for optimization, reducing the need for broadcast communication networks within the core. Third, GPAs enable block-atomic state tracking and orchestration, as opposed to the conventional instruction-oriented approaches. This block-atomic model serves to eliminate many per-instruction overheads and centralized structures associated with instruction fetch, rename, register read, commit, and exception handling. Finally, the instruction buffers associated with each ALU serve as a set of distributed reservation stations, enabling an effective dynamic scheduling window of hundreds to thousands of instructions.

GPA-based systems provide unique opportunities for power efficiency. The elimination of structures dedicated to instruction-level register renaming, associative operand comparisons, and state tracking reduce the overhead circuitry and power on a per-ALU basis. ALU chaining dramatically reduces the number of global register file accesses in exchange for short point-to-point connections. The dynamic power of the ALU array and banked memory structures can be actively managed to reduce consumption during periods of lighter utilization. The dataflow execution model of the GPA is also amenable to power-efficient asynchronous design techniques.

In addition to high ILP, a secondary design goal of the GPA is *polymorphism*, or the ability to adapt the hardware to the execution characteristics of the application. Grid Processors can be easily sub-divided into sub-processors, allowing discrete threads to be assigned to different sub-processors for high thread-level parallelism (TLP). Grid Processors can also be configured to target data-level parallelism (DLP), often exhibited in media, streaming, and scientific codes. For DLP applications, the same GPA hardware employs a different execution model in which instructions for kernels or inner loops are mapped to the ALUs and stay resident for multiple iterations. In addition, each access to a data cache bank provides multiple values that are distributed to the ALUs in each row. Initial results on a set of 7 signal processing kernels show that an 8x8 GPA can average 48 compute instructions per cycle. Assuming an 8-GHz clock in 50nm CMOS, this configuration would achieve a performance level of 384 GFlops.

As shown in Figure 9.6.6, we are planning a prototype chip that will consist of four 4x4 cores, a shared L2 cache structure built from an array of 128KB memory banks connected by a routed network, and a set of distributed memory controllers with channels to external memory. The prototype will be built using a 130nm process and is targeted for completion in 2005. Future technology generations will enable similar chips with even more powerful 8x8 cores.

References:
[1] M.S. Hrishikesh, N.P. Jouppi, K.I. Farkas, D. Burger, S.W. Keckler, and P. Shivakumar, "The Optimal Logic Depth Per Pipeline Stage is 6 to 8 FO4 Inverter Delays," *ISCA-29*, pp. 14-24, May, 2002.
[2] R. Nagarajan, K. Sankaralingam, D. Burger, and S.W. Keckler, "A Design Space Evaluation of Grid Processor Architectures," *MICRO-34*, pp. 40-51, December, 2001.
[3] M. Ozawa, M. Imai, Y. Ueno, H. Nakamura, and T. Nanya, "Performance Evaluation of Cascade ALU Architecture for Asynchronous Super-Scalar Processors," *ASYNC 2001*, pp. 162-172, March, 2001.

List of figure captions:

Figure 9.6.1: Historical reduction in cycle time driven by pipelining.

Figure 9.6.2: Projected fraction of chip reachable in one cycle with an 8FO4 clock period.

Figure 9.6.3: Grid Processor block diagram.

Figure 9.6.4: Mapping of dataflow critical path to physical ALUs.

Figure 9.6.5: An 8x8 GPA achieves 1.1-14x greater instructions per clock (IPC) than a conventional out-of-order core.

Figure 9.6.6: Diagram of proposed chip-multiprocessor (CMP) prototype with four GPA cores.

Figure 9.6.1 (top-left chart):

Clock Period (FO4) — y-axis: 0, 10, 20, 30, 40, 50, 60, 70, 80

x-axis: 1992, 1994, 1996, 1998, 2000, 2002

Legend:
- ● Intel x86
- ■ IBM Power
- ◆ HP PA
- ▲ Sun Sparc
- ✕ DEC Alpha
- + AMD K-series

Cray 1-S

Figure 9.6.1: Historical reduction in cycle time driven by pipelining.

Figure 9.6.2 (top-right):

180nm
150nm
130nm
90nm
70nm
50nm
35nm

20mm

Figure 9.6.2: Projected fraction of chip reachable in one cycle with an 8FO4 clock period.

Figure 9.6.3 (middle-left):

Prototype specifications:
- 5 8KB I-Cache banks
- 4 8KB D-Cache banks
- 128 instruction buffers per ALU
- 4 256 register 2-ported register file banks

Register File

I-Cache

D-Cache

Grid Control

Secondary Cache Interface

Input ports
Operand buffers
Instruction buffers
FP
Integer
Router
Output ports

Figure 9.6.3: Grid Processor block diagram.

Figure 9.6.4 (middle-right chart):

Instructions per Clock (IPC) — y-axis: 0, 1, 2, 3, 4, 5, 6, 7

x-axis: ammp, art, bzip2, compr, gzip, m88ksim, mcf, parser, twolf, vortex, mean

Legend:
- ■ 8x8 GPA
- □ Alpha 21264

Figure 9.6.4: An 8x8 GPA achieves 1.1-14x greater instructions per clock (IPC) than a conventional out-of-order core.

Figure 9.6.5 (bottom-left):

move t2,r2
move t3,r3

I1 add t1,t2,t3
I2 add r2,t2,t1
I3 ld t4,(t1)
I4 add t5,t4,1
I5 beqz t5,0xdeac

Dataflow Graph

Register File

move t2,r2
move t3,r3

I2 I1
I3
I4
I5

ALU grid

Critical Path Mapping

Figure 9.6.5: Mapping of dataflow critical path to physical ALUs.

Figure 9.6.6 (bottom-right):

M M
M M
DRAM Interface

M M
M M
DRAM Interface

Figure 9.6.6: Diagram of proposed chip-multiprocessor (CMP) prototype with four GPA cores.