Proceedings of Hotchips 13, August 2001.

The Raw Processor: A Composeable 32-Bit Fabric for Embedded and General Purpose Computing

Michael Taylor, Jason Kim, Jason Miller, Fae Ghodrat,
Ben Greenwald, Paul Johnson, Walter Lee, Albert Ma,
Nathan Shnidman, David Wentzlaff, Matt Frank,
Saman Amarasinghe and Anant Agarwal

MIT Laboratory for Computer Science
{`mtaylor, jkim, jasonm, fghodrat, beng, prj, walt, ama, naters, wentzlaff, mfrank, saman, agarwal`}`@lcs.mit.edu`

The Raw project is attempting to create a scalable processor architecture that is suitable for both general purpose and embedded computations. Current general purpose processors differ from embedded devices in that they provide large amounts of hardware support to discover and manipulate instruction-level parallelism and unstructured memory accesses. Because the parallelism in embedded computations is much more predictable, embedded devices such as DSPs do not offer a rich set of mechanisms, rather they devote their area to computational resources such as pipelined floating point, thereby achieving significantly better area and energy efficiency. However, their best performance is achieved for regular data access patterns such as streams, and they often require assembly code manipulation. Embedded FPGAs and ASICs go one step further, and can offer even better results for many classes of computations, but require a hardware design step in mapping their applications into silicon.

Raw will support many classes of computations that traditionally have run on microprocessors, DSPs, FPGAs and ASICs. Raw implements a simple, highly parallel, tiled architecture, and exposes its interconnect, I/O, memory and computational elements to the compiler [5]. This exposure allows the software system to allocate resources and coordinate data flow within the chip in an application-specific manner. Furthermore, the tiled, replicated architecture of Raw allows it to scale with increasing silicon densities.

As depicted in Figure 1, the Raw processor is a single chip containing 16 identical processor-sized tiles connected in a 4-by-4 mesh configuration by four nearest neighbor point-to-point pipelined high-speed networks (two static, two dynamic). Implemented in the .15 micron IBM SA-27E ASIC process, the design occupies an 18.2x18.2mm die, has 1080 HSTL signal I/Os, consumes 45W[1], and runs at a target frequency of 250 MHz. Because tiles are only connected to their nearest neighbors, the longest wire on the chip runs only the length of a single tile.

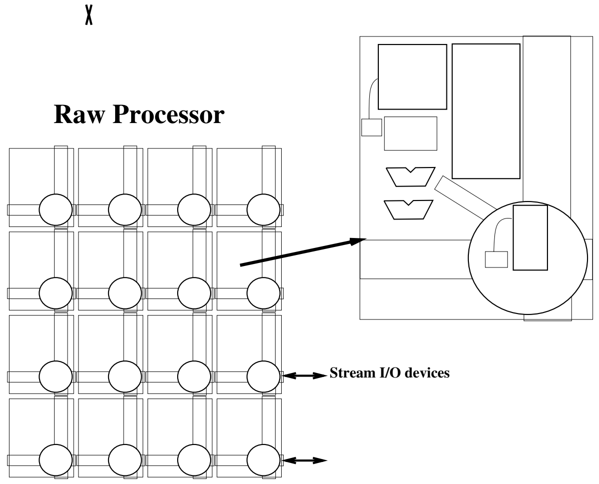

Fig. 1. Raw processor composition. A typical Raw system might include a Raw processor coupled with off-chip RDRAM and stream-IO devices.

Each tile contains a general-purpose processor, which is connected to its neighbors by a static router and a dynamic router. The processor is an eight stage single-issue MIPS-style pipeline. It has a four stage pipelined FPU, a 32 KByte two-way associative SRAM data cache and 32 KBytes of instruction SRAM that is virtualized via a binary rewriting system. When the data access patterns are known at compile time, the software implements software data caching for predictable memory access by using a portion of the instruction SRAM memory.

The static router controls two independent 32-bit pipelined channels in each direction (North, East, South, West, and Processor). It

[1] To reduce project risk, we have not focussed on low-power design in our first experimental Raw prototype. However, its tiled, pipelined nature allows application-specific orchestration of power, and we plan to undertake a low-power design in a follow-on project.

sequences a 64-bit instruction that simultaneously specifies an operation (branch, no-op, or move) and 12 routes between these channels. The static router's local SRAM contains 8K of these instructions, and is also virtualized in software. Like the instruction SRAM, the software can also use a portion of the static router SRAM to store data with predictable access patterns.

In order to route a value between two tiles on this network, one inserts instructions on each intermediate node specifying the appropriate route. The static router allows single word messages to be sent between tiles with a guaranteed relative ordering. The purpose of this network is to connect the Raw tiles in a manner that can exploit both ILP (instruction level parallelism) and streaming data parallelism. The parallelizing Raw compiler, RAWCC [4, 3, 2, 1], uses these routers as an operand network between the ALUs of the processors to parallelize SpecFP 95 and multimedia applications.

The dynamic router uses a dimensioned-ordered wormhole routing protocol to control two independent 32-bit pipelined channels in each direction (North, East, South, West, and Processor). These channels allow messages of up to 31 words (plus a header specifying the destination tile number, source tile number, message length, and type) to be sent between tiles and outside of the chip. These messages take one cycle per hop when going straight, and two cycles on turns. The processor dedicates one of these two dynamic channels to external communication: memory (i.e., cache misses), interrupts, and PCI transactions. The other channel is utilized for user-level messaging between tiles. The processor supports OS-level transparent deadlock recovery in the event that the user over-commits the buffer resources of the network.

The pins of the chip are connected directly to the edges of the mesh networks, and run at the speed of the chip. When a message is routed off the side of the chip, it appears on the pins. Because the number of network signals (38 signals x 4 networks x 16 tile-sides x 2 directions = 4864) exceeds the number of pins available, we transparently multiplex the 2 static and 2 dynamic networks on each port down to one physical channel. We also multiplex the middle two channels in the vertical direction, which results in 14 channels, or 112 Gbits of bandwidth in each of the output and input directions.

The Raw chip can be composed into power-of-two dimensioned meshes, for systems of up to 64 chips. The system is virtualized in such a way that this system will appear to the programmer to be exactly like a single Raw chip of 64 times the size, except that certain hops on the networks have an extra cycle (or two, for the shared middle channel) of latency. This allows us a glimpse into the future: we can ascertain the scalability of our architecture as well as run applications that are beyond the capabilities of modern day microprocessors.

Acknowledgments: The Raw project is funded by Darpa, NSF, and the Oxygen Alliance. IBM is supporting the ASIC fabrication of Raw.

References

1. Anant Agarwal, Saman Amarasinghe, Rajeev Barua, Matt Frank, Walter Lee, Vivek Sarkar, and Michael Taylor. The raw compiler project. In *Proceedings of the Second SUIF Compiler Workshop*, Stanford CA, 1997.
2. Rajeev Barua, Walter Lee, Saman Amarasinghe, and Anant Agarwal. Memory Bank Disambiguation using Modulo Unrolling for Raw Machines. In *Proceedings of the ACM/IEEE Fifth Int'l Conference on High Performance Computing(HIPC)*, Dec 1998. Also http://www.cag.lcs.mit.edu/raw/.
3. Rajeev Barua, Walter Lee, Saman Amarasinghe, and Anant Agarwal. Maps: A Compiler-Managed Memory System for Raw Machines. In *Proceedings of the 26th International Symposium on Computer Architecture*, Atlanta, GA, May 1999.
4. Walter Lee, Rajeev Barua, Matthew Frank, Devabhatuni Srikrishna, Jonathan Babb, Vivek Sarkar, and Saman Amarasinghe. Space-Time Scheduling of Instruction-Level Parallelism on a Raw Machine. In *Proceedings of the Eighth ACM Conference on Architectural Support for Programming Languages and Operating Systems*, pages 46–57, San Jose, CA, October 1998.
5. Elliot Waingold, Michael Taylor, Devabhaktuni Srikrishna, Vivek Sarkar, Walter Lee, Victor Lee, Jang Kim, Matthew Frank, Peter Finch, Rajeev Barua, Jonathan Babb, Saman Amarasinghe, and Anant Agarwal. Baring It All to Software: Raw Machines. *IEEE Computer*, 30(9):86–93, September 1997. Also available as MIT-LCS-TR-709.