AsAP: A Fine-Grained Many-Core Platform for DSP Applications

MANY EMERGING AND FUTURE APPLICATIONS REQUIRE SIGNIFICANT LEVELS OF COMPLEX DIGITAL SIGNAL PROCESSING AND OPERATE WITHIN LIMITED POWER BUDGETS. MOREOVER, DRAMATICALLY RISING VLSI FABRICATION AND DESIGN COSTS MAKE PROGRAMMABLE AND RECONFIGURABLE SOLUTIONS INCREASINGLY ATTRACTIVE. THE ASAP PROJECT ADDRESSES THESE CHALLENGES WITH A CHIP MULTIPROCESSOR COMPOSED OF SIMPLE PROCESSORS WITH SMALL MEMORIES, ACHIEVING HIGH ENERGY EFFICIENCY AND THROUGHPUT IN A SMALL CHIP AREA.

Bevan Baas
Zhiyi Yu
Michael Meeuwsen
Omar Sattari
Ryan Apperson
Eric Work
Jeremy Webb
Michael Lai
Tinoosh Mohsenin
Dean Truong
Jason Cheung
University of California, Davis

Applications that require the computation of complex, multitask digital-signal processing (DSP) workloads are becoming increasingly commonplace. Such applications include wired and wireless communications, multimedia, sensor signal processing, and medical devices. Many are embedded and are key components in their host systems—which commonly exist in high-volume products.

In developing a suitable computing platform for these target applications, we adopted the following characteristics as goals. The platform should be

- highly energy efficient under all operating conditions;
- capable of high performance (to serve the broadest range of applications, we target workloads varying in requirements from low to very high rates);
- programmable or reconfigurable and easy to program using a high-level language; and
- well-suited for future fabrication technologies. (The key relevant differences in deep-submicron CMOS technologies include extremely large numbers of available transistors, very expensive one-time fabrication and design costs, relatively slow ''long'' wires, large variations in both transistors and wires, and high leakage currents.)

The result of our work on these goals is AsAP, the *Asynchronous Array of simple Processors*. We geared AsAP's design toward both handling the task-level parallelism inherent in DSP applications and making best use of the capabilities of deep-submicron technology while avoiding many of its limitations. Organized as a chip-

Published by the IEEE Computer Society.

multiprocessor composed of many small and simple processing elements, with globally asynchronous, locally synchronous (GALS) clocking, and a nearest-neighbor mesh interconnect, AsAP achieves greater performance per area and better energy-efficiency than current DSPs. It is also many times smaller.

Motivating observations

> *You know you have achieved perfection in design, not when you have nothing more to add, but when you have nothing more to take away.*
> *—Antoine de Saint Exupéry*

Several observations about our target applications and fabrication technologies motivated the many design decisions that resulted in AsAP. Here, we discuss four of the most significant points.

Modern DSP applications often consist of multiple cascaded simple tasks. An example of a modern demanding application is the digital baseband processor for an IEEE 802.11a/11g wireless LAN transmitter. Figure 1 shows a simplified block diagram of the transmitter and the cascading flow of data between tasks. Clearly, the computation of these tasks on a traditional architecture consisting of a very fast CPU fed by a multilevel memory hierarchy would result in significant memory traffic. It is reasonable that an architecture that more closely models the application composition would require less data movement and likely less total memory as well.

Task-level parallelism in modern DSP applications is often easy to find; in fact, the IEEE specification contains a diagram very similar to Figure 1. This motivated us to pursue task-level parallelism in our design ahead of data- and instruction-level parallelism.

We expect the shift of application structures to ones with high levels of task-level parallelism to not only continue, but to intensify. Recent generations of applications are often not merely faster versions of previous generations but are typically far more complex in structure and contain a wider variety of subtasks (for example, 802.11n vis-à-vis 802.11b, and H.264 vis-à-vis MPEG-2), which makes their computation with traditional architecture processors even less efficient.

Common DSP tasks often require less than several hundred words of memory for both instructions and data. Memory occupies much of the area in modern processors—often in the range of 55 to 75 percent of the processor's area.[1] This has the unfortunate side effect of reducing the area available for datapath circuits, which are really the reason for the processor's existence, especially in the DSP domain.

As Table 1 shows, the numbers of instructions and data words required to compute several common DSP tasks are far smaller than the amount of memory found in modern processors, which typically ranges from 10 Kbytes to 10 Mbytes. Reducing memory sizes can result in significant area and power reductions.

Large wire delays and large variations in device and wire performance in future fabrication technologies increasingly complicate the distribution of high-speed and low-

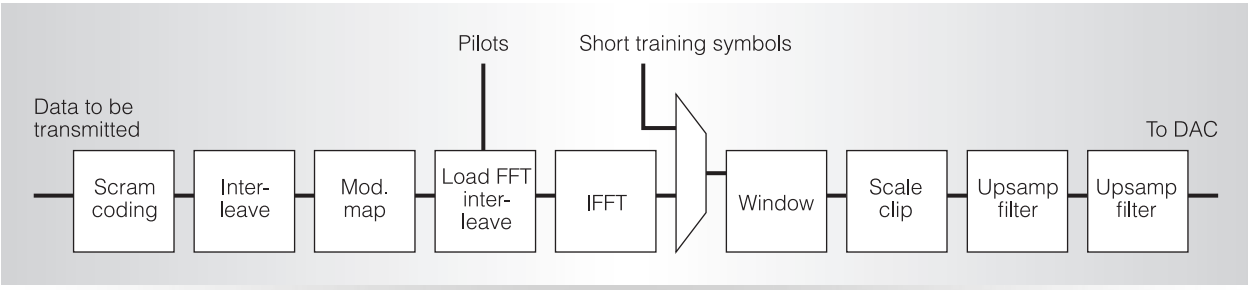


Figure 1. Block diagram of an IEEE 802.11a/11g wireless LAN baseband transmitter, DAC: digital-to-analog converter.

Table 1. Instruction and data words required for common DSP tasks for a simple memory-to-memory processor with a basic ALU and hardware multiplier.

Task	Instruction memory required (no. of words)	Data memory required (no. of words)
N-point FIR filter	6	2*N*
8-point DCT	40	16
8×8 2D DCT	154	72
Convolutional coder ($k = 7$)	29	14
Huffman encoder	200	330
N-point convolution	29	2*N*
64-point complex FFT	97	192
Bubble sort	20	1
N-element merge sort	50	*N*
Square root	62	15
Exponential (e^x)	108	32

FIR: finite impulse response; DCT: discrete cosine transform; FFT: fast Fourier transform

skew clock and data signals. It is well known that long on-chip wires scale poorly in deep-submicron technologies. Although architectures can compensate for long wire delays to some extent, large variations in these wire delays provide a considerable challenge to achieving correct functionality in processors. Thus, a clocking and processor interconnect methodology that avoids long wires will likely require less design time and will scale to larger sizes far more easily.

The most efficient processing element contains sufficient resources to capture the computational kernels of the targeted workloads, and no more. It is clear that a processing element with more circuit resources than necessary will be slower and consume more power and chip area than a processor without excess resources. It also makes sense that a design with very small processing elements (for example, FPGA logic cells) will also be slower, more power hungry, and larger if it requires many routing resources to map a computation kernel.

The AsAP architecture

Inspired by these and other observations, four key features enable AsAP to compute the targeted workloads with high energy efficiency, high throughput, and small chip area.

Chip multiprocessor

The traditional method of achieving high performance has been to rely mainly on high clock rates at the expense of deeper pipelines and increased design complexity. Especially for DSP workloads, in which data-level and task-level parallelism are abundant, considering highly parallel architectures makes sense. AsAP takes full advantage of parallelism with very fine-grained processing elements that permit large numbers of processors per die.

Small memories and simple processors

Each AsAP processor is a simple, single-issue processor with a 64-word × 32-bit instruction memory, a 128-word × 16-bit data memory, a 16-bit ALU, a 16 × 16-bit multiplier with a 40-bit accumulator, and four programmable address generators. Each processor uses memory-to-memory instructions with no register file, and provides no support for branch prediction, out-of-order execution, or speculative operations.

Each processor supports 54 very general instructions, among which only the bit-reverse is algorithm specific (used for FFT address calculations). This is in line with our philosophy that the greatest usefulness of programmable processors is in computing future, presently unknown workloads. Although this choice hinders AsAP's performance on common benchmarks, it makes the platform suitable for a much broader spectrum of workloads.

Independent and haltable GALS clocking

Each processor is a completely standard digital machine clocked by a single synchronous clock signal. However, each processor also has its own digitally programmable clock oscillator that is fully independent from all other oscillators. There are no clock crystals, phase-locked loops, delay-locked loops, or global frequency or phase-related signals; and the system is globally asynchronous, locally synchronous (GALS). The oscillators are truly independent—they can change their frequency over their full range (1.66 to 600+ MHz in our

fabricated chip), halt, and restart arbitrarily. Each oscillator occupies less than 1 percent of its processor's area.

Independent clocking lets each oscillator halt when its host processor is idle, thus reducing power dissipation. Oscillators fully halt in nine cycles when there is no work to do, and restart at full speed in less than one cycle after work is available. Processors stall either when they execute a FIFO read and the FIFO buffer is empty, or when they execute an output port write and the port is full. Processors dissipate zero active power (leakage only) while stalled.

A second benefit of GALS clocking is that processor clock speeds can be precisely adjusted to match the workloads of individual processors.[2] Although the first AsAP chip does not implement per-processor voltage scaling, a design using joint voltage and frequency scaling could achieve even greater power reductions.

Unfortunately, independent clocking comes at a cost—namely, the circuits required to reliably transfer data across independent clock domains. AsAP's solution is to use a special dual-clock FIFO design with its write port clocked in one clock domain and its read port clocked in another domain. Address pointers are gray-coded (a number coding scheme where adjacent codes are guaranteed to differ by only one bit), so the worst-case failure is a single metastable bit. The design reduces the probability of a metastable bit to extremely low levels at the expense of additional FIFO latency through a configurable number of synchronization registers. We have operated multiple processors with only one synchronization register for several weeks without observing a single failure.

Nearest-neighbor interprocessor communication

Because of its excellent match with planar fabrication technologies, high scalability, and low overhead, AsAP uses a circuit-switched, nearest-neighbor network to connect processors. Whereas packet-switched networks generally work better for random and dynamically changing communication patterns, most DSP and embedded tasks exhibit fixed communication patterns that can be largely localized by careful task placement, and are efficiently handled by a circuit-switched network. The network is reconfigurable in the sense that each processor's two inputs can map to any two of its four neighbors (north, south, east, or west). Each processor also has four outputs to the same four neighbors and can write to any dynamically changing number or pattern of them under program control.

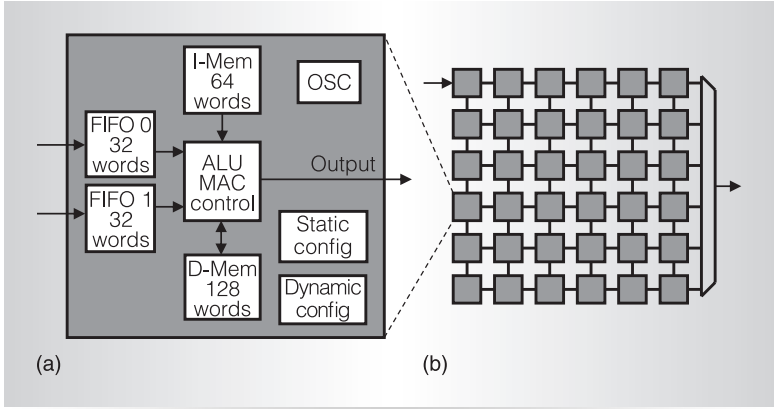


Figure 2. Major blocks inside a single AsAP processor (a), and the AsAP chip containing 36 processors (b). Despite having an input and output port on each processor's edge (24 in total), the array has one input port and one multiplexed output port, an arrangement due solely to chip package restrictions.

The AsAP chip

We designed an AsAP chip containing 36 (6 × 6) processors and it was fabri-

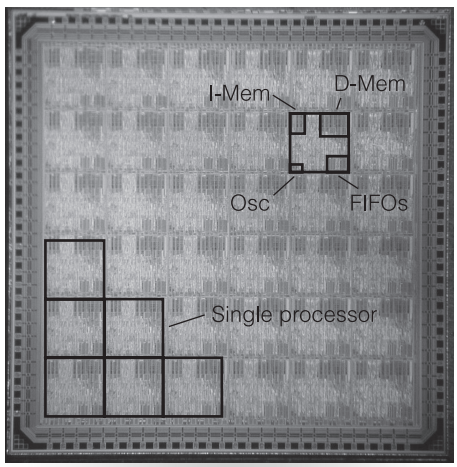


Figure 3. Die micrograph of the 36-processor AsAP chip.

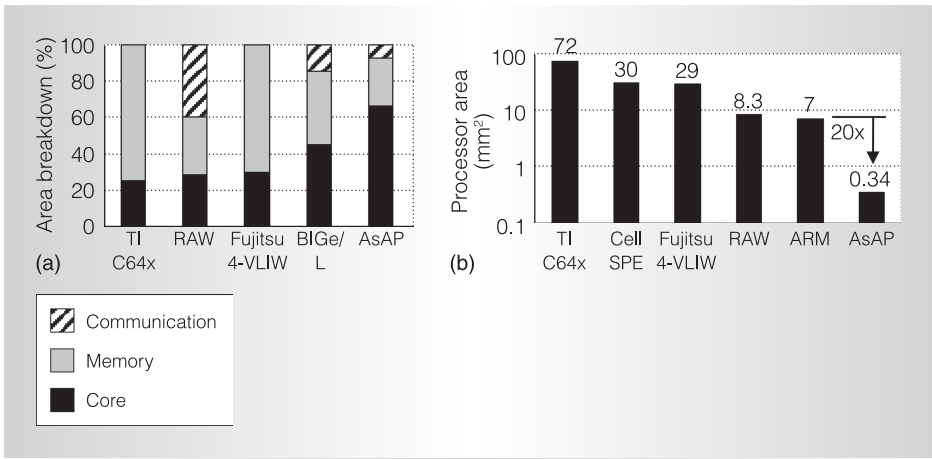


Figure 4. Comparisons of AsAP with other processors in terms of area breakdown (a) and absolute area (b). All processors are scaled to 0.13 μm.[1,4–8]

cated in TSMC's 0.18-micron CMOS technology using standard cells, and it is fully functional.[3] Figure 2 shows the block diagram. These are a few of the key measurements:

- 520–540 MHz clock rates at 1.8 V (an improved test environment allows us to increase the maximum clock rate from our previously reported 475 MHz[3]) and clock rates over 600 MHz at 2.0 V for most processors;
- 32 mW average power for each processor while executing applications at 475 MHz;
- 84 mW power for each processor while 100 percent active at 475 MHz;
- 2.4 mW average application power per processor at 0.9 V and 116 MHz, and
- 0.66 mm^2 die area for each processor.

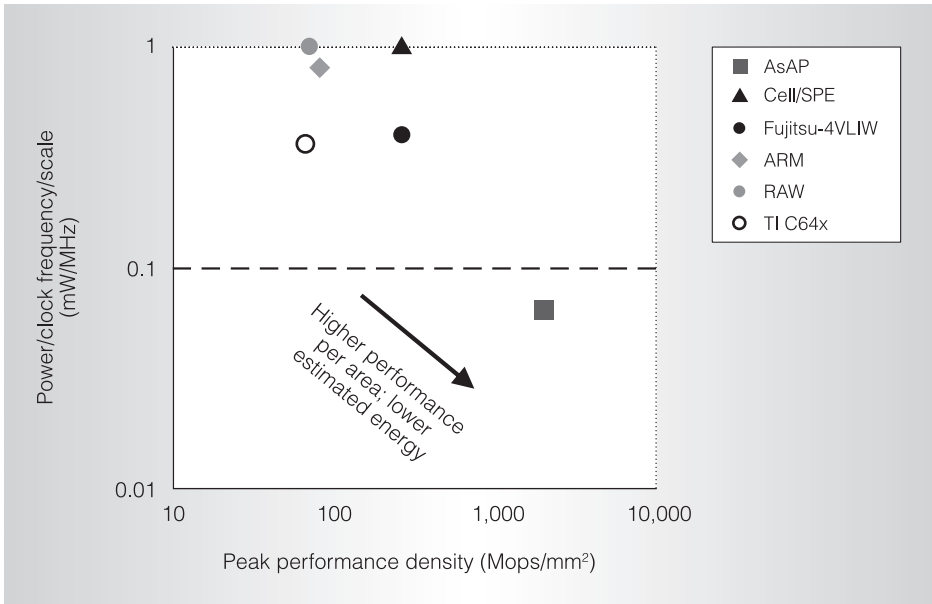


Figure 5. Comparisons of approximate peak performance per area versus energy per operation. Word widths and workload not factored. All processors are scaled to 0.13 μm. We assume 2 operations/cycle for Cell/SPE and 3.3 operations/cycle for TI C64x.[1,4–8]

If this exact design were implemented in a 13-mm × 13-mm chip in 90-nm CMOS technology, it would contain more than 1,000 processors, operate at approximately 1 GHz, have a peak throughput of 1.0 tera-operations per second, and dissipate only 10 to 15 W plus leakage when executing loads equivalent to our target applications.

Figure 3 shows the abutted processor tile structure that gives AsAP its great scalability. There is an interesting story about this chip's history. The first AsAP design was a 9-processor, 3 × 3 array. The week before tapeout, we learned that additional space was available on the run. So, we redesigned the chip-level circuits, I/O interfaces, pad ring, and power grid, and moved the new 6 × 6 design through the entire flow, all the way to GDSII (the database format for an IC layout), all in just 10 hours. This GDSII was actually fabricated. Our unexpected opportunity provided a real-life demonstration of how easy it is to scale a processing array without any global wires.

AsAP compares well with other processors, with 66 percent of each processor's area for the core, and also in terms of its absolute processor area, as Figure 4 shows. AsAP is 20 to 210 times smaller than the other processors in the comparison.

AsAP also compares well in terms of its energy efficiency and performance per chip area. Figure 5 plots these two values against each other—without considering, because of a lack of available data, important factors such as word widths and workload. When scaled to the same technology, AsAP achieves energy efficiencies very approximately on the order of 5.5 to several dozen times better than the processors shown, and performance densities on the order of 7.5 to several dozen times better.

```
int atan[14];
int A = 26981, X = 16384;
int Y = 0, Z = 0;
int Xshft, Yshft, i;

Acc = A * Ibuf0;
Acc = Acc >> Ibuf0;

for (i=0; i<14; i=i+1) {
    Xshft = X >> i;
    Yshft = Y >> i;
    if (Y < Acc) {
        X = X - Yshft;
        Y = Y + Xshft;
        Z = Z - atan[i]; }
    else {
        X = X + Yshft;
        Y = Y - Xshft;
        Z = Z + atan[i]; }
    }
Obuf = 0 - Z;
Obuf = Z + 12868;
```

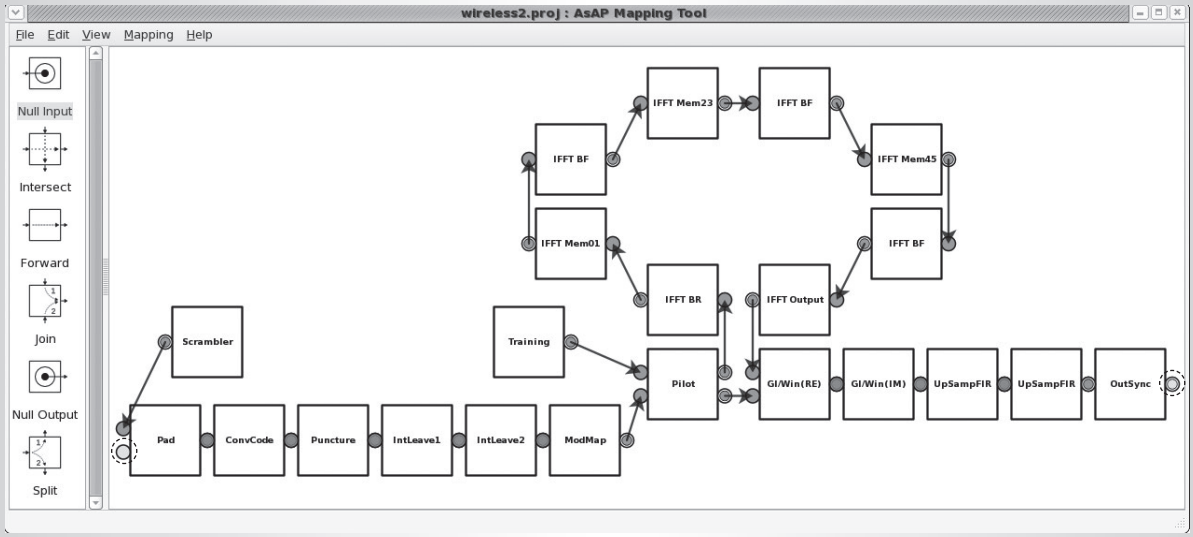
Figure 6. AsAP C program of a combined arcsin-arccos generator.

Programming the array

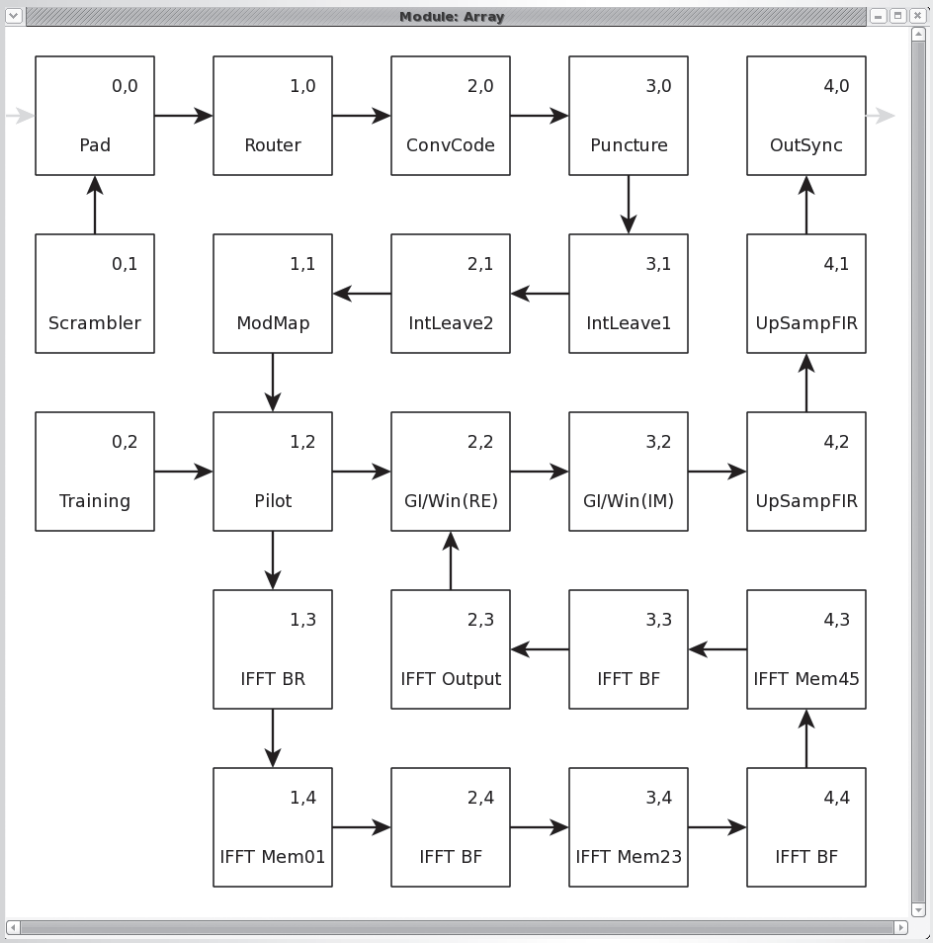
We have programmed many DSP and general tasks on a varying number of AsAP processors, including filtering; convolutional coding; interleaving; sorting; square-root calculations; CORDIC sin, cos, arcsin, and arccos (coordinate rotation digital calculations); matrix multiplication; pseudorandom number generation; and complex FFTs of lengths 32 to 1,024. We have also programmed a complete $k = 7$ viterbi decoder, a JPEG encoder, and a complete, fully compliant IEEE 802.11a/11g wireless LAN baseband transmitter. Blocks composed of any number of processors plug directly together with no modifications required whatsoever.

Programming the AsAP array follows three basic steps:

1. Individual programs are written for each task in either C or assembly language. Figure 6 shows a program that computes the CORDIC arcsin-arccos. Reserved variables Ibuf0, Ibuf1, and Obuf indicate input FIFO 0, input FIFO 1, and the output port, respectively.
2. Tasks are combined using either manual configuration code or by linking inputs and outputs of each task using the mapping tool's GUI shown in Figure 7a. Each task processed by the GUI is stored in a flexible XML format which allows small programs to be included directly in larger applications.



(a)



(b)

Figure 7. Screen snapshots of the auto-mapper's hand-drawn processor graph (a) and automatically mapped output graph (b), for the 22-processor IEEE 802.11a/11g wireless LAN transmitter shown in Figure 9. The tool was constrained to use a 5 × 5 array, and the ''Router'' processor in location 1,0 was automatically inserted by the auto-mapper to complete the mapping.

3. The AsAP C compiler, assembler, and automatic mapping tool compile programs, map them to the 2D mesh of processors, and produce a single bit file that programs and configures the array when loaded into the chip.

We have found AsAP's memory restrictions to be less problematic than they might appear at first. There is a strong analogy between writing functions or subroutines in a large single-threaded program and writing programs for subtasks as part of a complex application. Individual task programs encapsulate complexity and make the overall application simpler to write.

Figure 8 shows our simplest (and slowest) JPEG encoder implementation. The five component tasks are spread across nine processors; the wide gray arrows indicate the main data-flow path. The encoder is fully functional on the chip, and at a clock rate of 300 MHz has similar performance with Texas Instruments' eight-way very long instruction word (VLIW) TI C62x, but with approximately 11 times lower energy dissipation.[9,10] Higher-performance implementations with more processors are clearly possible.

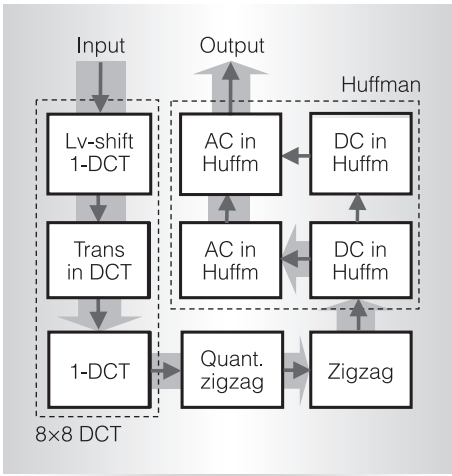


Figure 8. Nine-processor AsAP implementation of a JPEG encoder.

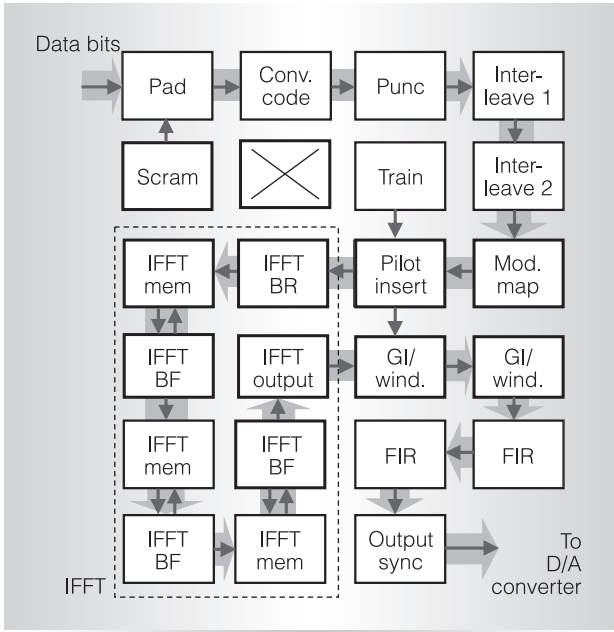


Figure 9. AsAP implementation of an IEEE 802.11a/11g wireless LAN transmitter, using 22 processors.

One of our most complex applications to date is the IEEE 802.11a/11g baseband transmitter;[11] Figure 9 shows the implementation. It is fully compliant with the IEEE standard—including training symbols, pilot subcarriers, and operation over all eight rates—and it even includes upsampling and filters that the standard does not require. This application was written in three months by one graduate student with no prior knowledge of the standard and a full load of classes, and it is lightly optimized. The transmitter dissipates 407 mW at 300 MHz, which lets it operate at 30 percent of full speed at the highest rate of 54 Mbps. In comparison to a noncomprehensive implementation on a Texas Instruments eight-way VLIW C62x,[12] this implementation has 5 to 10 times higher performance and 35 to 75 times lower energy dissipation.

Although we took the JPEG and 802.11a/11g data with all processors operating at 300 MHz, the power dissipation is only slightly less when we reduce the clock frequencies of lightly loaded processors. This is because the energy per operation is not a function of the clock frequency, since voltage scaling was not implemented—except for the second-order effect of additional overhead cycles to flush the pipeline when entering a stall. Reducing the clock frequencies of lightly loaded

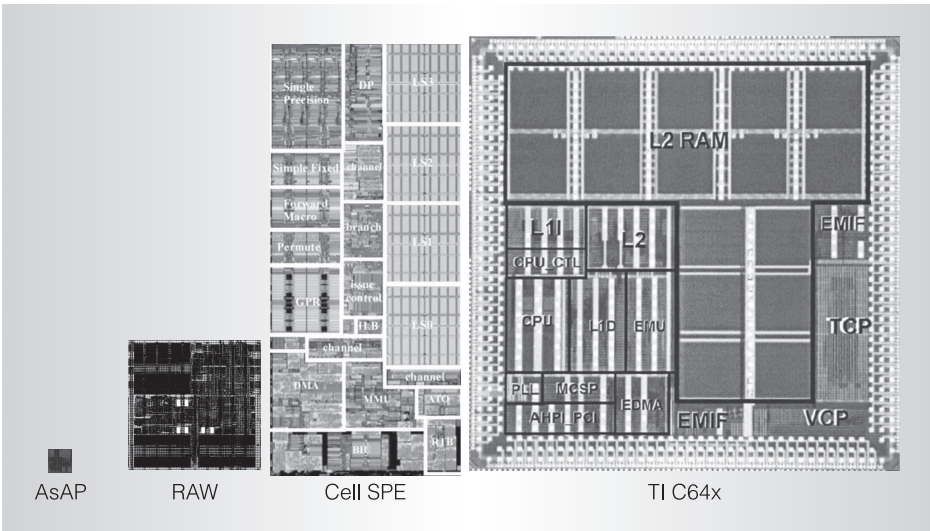


Figure 10. Approximate scale die micrographs of the multicore processors AsAP, RAW,[4] and Cell SPE,[7] and the TI C64x[1] DSP processor scaled to the same technology.

processors makes virtually no change in throughput or latency.

Counter to intuition, the independently clocked processors are actually well-suited for many real-time applications. The final processor in Figure 9, labeled Output sync, is clocked by an external clock (the digital/analog converter clock in this example) and executes a trivial program moving data from its input to its output. As long as processors in the array are fast enough, hardware flow control throughout the array prevents overflow and underflow, and the D/A converter never misses a sample.

Automatic application graph mapping onto the homogeneous AsAP array enables some very interesting possibilities of bypassing faulty or poorly performing processors at the time of manufacture for yield enhancement, or for self-healing when processors fail over time—for example, because of negative bias temperature instability or hot carrier injection effects.

"Throw-away" processors: A new paradigm

AsAP's processors' low power when unused (zero active power) and extremely small size make processors so cheap that they can be used for trivial tasks such as serving as memories or routing data between distant processors.[13] In this paradigm, load balancing is far less important than low idle-processor power and efficient application graph mappings.

Using processors for trivial tasks runs counter to many modern processor designs, which often use very large circuit structures to keep datapaths busy. Our approach makes even greater sense in very deep-submicron technologies, in which transistors are even more abundant. Figure 10 puts AsAP's area in perspective and illustrates the reason for its high efficiency and great numbers.

Although AsAP's key features of parallel processors, small and simple processing elements, GALS clocking, and nearest-neighbor mesh interconnect distinguish it from other previous and current parallel processors, several processors share common features. Examples include Transputer,[14] systolic architecture,[15] wavefront architecture,[16] PADDI,[17] RAW,[4] Synchroscalar,[18] and the heterogeneous Picochip[19] and Cradle[20] processors.

One of AsAP's most significant limitations is its inefficiency in computing workloads with active memory requirements far larger than a few thousand words, such as video applications and long-code, low-density, parity-check (LDPC) decoders. We are actively working on efficient archi-

tectures for including large shared memories on chip. Our other ongoing efforts include work on techniques to reduce leakage power and on shared special-purpose accelerators. We are also interested in expanding the application base, including finding important applications that could map to a floating-point version of AsAP.

AsAP's high performance and high energy efficiency compare well with modern DSP processors. We believe architectures like AsAP—that make good use of billions of transistors, reduce design effort through a tileable and homogeneous architecture, eliminate long high-speed wires, and intrinsically accommodate some types of process variations through clock tuning and some circuit faults through remapping—will become increasingly attractive in deep-submicron fabrication technologies. MICRO

Acknowledgments

The authors gratefully acknowledge support from Intel, UC MICRO, NSF grant 0430090 and CAREER award 0546907, MOSIS, Artisan, and a University of California, Davis, Faculty Research Grant. We also thank M. Singh, R. Krishnamurthy, M. Anders, S. Mathew, S. Muroor, and W. Li.

References

1. S. Agarwala et al., ''A 600-MHz VLIW DSP,'' *J. Solid-State Circuits*, vol. 37, no. 11, Nov. 2002, pp. 1532-1544.
2. Z. Yu and B.M. Baas, ''Performance and Power Analysis of Globally Asynchronous Locally Synchronous Multi-Processor Systems,'' *Proc. IEEE Computer Soc. Ann. Symp. Emerging VLSI Technologies and Architectures* (ISVLSI 06), IEEE CS Press, 2006, pp. 378-384.
3. Z. Yu et al., ''An Asynchronous Array of Simple Processors for DSP Applications,'' *Proc. Int'l Solid-State Circuits Conf.* (ISSCC 06), IEEE Press, 2006, pp. 428-429.
4. M. Taylor et al., ''A 16-Issue Multiple-Program-Counter Microprocessor with Point-to-Point Scalar Operand Network,'' *Proc. Int'l Solid-State Circuits Conf.* (ISSCC 03), IEEE Press, 2003, pp. 170-171.
5. A. Suga et al., ''A 4-Way VLIW Embedded Multimedia Processor,'' *Proc. Int'l Solid-State Circuits Conf.* (ISSCC 00), IEEE Press, 2000, pp. 240-241.
6. A.A. Bright et al., ''Creating the BlueGene/L Supercomputer from Low-Power SoC ASICs,'' *Proc. Int'l Solid-State Circuits Conf.* (ISSCC 05), IEEE Press, 2005, pp. 188-189.
7. B. Flachs et al., ''A Streaming Processing Unit for a CELL Processor,'' *Proc. Int'l Solid-State Circuits Conf.* (ISSCC 05), IEEE Press, 2005, pp. 134-135.
8. R. Witek and J. Montanaro, ''StrongARM: A High-Performance ARM Processor,'' *Proc. IEEE Int'l Computer Conf.* (Compcon 96), IEEE CS Press, 1996, pp. 188-191.
9. C. Kozyrakis and D. Patterson, ''Vector vs. Superscalar and VLIW Architectures for Embedded Multimedia Benchmarks,'' *Proc. IEEE/ACM Int'l Symp. Microarchitecture* (Micro 35), IEEE CS Press, 2002, pp. 283-289.
10. T. Lin and C. Jen, ''Cascade—Configurable and Scalable DSP Environment,'' *Proc. Int'l Symp. Circuits and Systems* (ISCAS 02), IEEE Press, 2002, pp. 26-29.
11. M.J. Meeuwsen, O. Sattari, and B.M. Baas, ''A Full-Rate Software Implementation of an IEEE 802.11a Compliant Digital Baseband Transmitter,'' *Proc. IEEE Workshop Signal Processing Systems* (SIPS 04), IEEE Press, 2004, pp. 124-129.
12. M.F. Tariq et al., ''Development of an OFDM Based High Speed Wireless LAN Platform Using the TI C6x DSP,'' *Proc. IEEE Int'l Conf. Comm.* (ICC 02), vol. 1, IEEE Press, 2002, pp. 522-526.
13. B.M. Baas, ''A Parallel Programmable Energy-Efficient Architecture For Computationally-Intensive DSP Systems,'' *Conf. Record 37th Asilomar Conf. Signals, Systems, and Computers*, IEEE Press, 2003, pp. 2185-2189.
14. C. Whitby-Strevens, ''Transputers—Past, Present and Future,'' *IEEE Micro*, vol. 10, no. 6, Nov.–Dec. 1990, pp. 16-19, 76-82.
15. H.T. Kung, ''Why Systolic Architecture?'' *Computer*, vol. 15, no. 1, Jan. 1982, pp. 37-46.
16. S.Y. Kung et al., ''Wavefront Array Processor: Language, Architecture, and Appli-

cations,'' *IEEE Trans. Computers*, vol. 31, no. 11, Nov. 1982, pp. 1054-1066.
17. R.A. Sutton, V.P. Srini, and J.M. Rabaey, ''A Multiprocessor DSP System using PADDI-2,'' *Proc. 35th Design Automation Conf.* (DAC 98), ACM Press, 1998, pp. 62-65.
18. J. Oliver et al., ''Synchroscalar: A Multiple Clock Domain, Power-Aware, Tile-Based Embedded Processor,'' *Proc. Int'l Symp. Computer Architecture* (ISCA 04), IEEE CS Press, 2004, pp. 150-161.
19. R. Baines and D. Pulley, ''A Total Cost Approach to Evaluating Different Reconfigurable Architecture Architectures for Baseband Processing in Wireless Receivers,'' *IEEE Comm. Magazine*, vol. 41, no. 1, Jan. 2003, pp. 105-113.
20. *Multiprocessor DSPs: Next Stage in the Evolution of Media Processor DSPs*, tech. report, Cradle Technologies; http://www.ed-china.com/ARTICLES/2006MAY/5/2006MAY29_CP_EMS_TS_1.PDF.

Bevan Baas is an assistant professor in the Department of Electrical and Computer Engineering at the University of California, Davis. He currently leads projects in architecture, hardware, software tools, and applications for VLSI computation with an emphasis on DSP workloads. Baas has a PhD and MS in electrical engineering from Stanford University and a BS in electronic engineering from California Polytechnic State University, San Luis Obispo.

Zhiyi Yu is a PhD candidate in electrical and computer engineering at the University of California, Davis. His research interests include high-performance and energy-efficient digital VLSI design with an emphasis on many-core GALS clocking and efficient processor interconnects. Yu has an MS and BS in electrical engineering from Fudan University.

Michael Meeuwsen is a hardware engineer with Intel Digital Enterprise Group. His research interests include digital circuit design and 802.11a/g algorithm mapping. Meeuwsen has an MS in electrical and computer engineering from the University of California, Davis, and BS degrees in electrical engineering and computer engineering from Oregon State University.

Omar Sattari is a software engineer at CornerTurn. His research interests include FFT and DSP algorithms and digital hardware design. Sattari has an MS and BS in electrical and computer engineering, both from the University of California, Davis.

Ryan Apperson is an IC design engineer at Boston Scientific CRM Division. His research interests include multiclock domain systems and SRAM design. Apperson has an MS in electrical and computer engineering from the University of California, Davis, and a BS in electrical engineering from the University of Washington.

Eric Work is an MS student in electrical and computer engineering at the University of California, Davis. His research interests include the mapping of arbitrary task graphs to processor networks and software tool flow. Work has a BS from the University of Washington.

Jeremy Webb is an MS student in electrical and computer engineering at the University of California, Davis, and a hardware engineer at Agilent. His research interests include high-speed board design and system interfacing. Webb has a BS from the University of California, Davis.

Michael Lai is a design engineer at Altera. His research interests include the design of high-speed arithmetic units and control. Lai has an MS and BS in electrical and computer engineering from the University of California, Davis.

Tinoosh Mohsenin is a PhD candidate in electrical and computer engineering at the University of California, Davis. Her research interests include Low Density Parity Check (LDPC) algorithms and architectures, and processor design. Mohsenin has an MS in electrical and computer engineering from Rice University and a BS in

electrical engineering from Sharif University.

Dean Truong is a PhD student in electrical and computer engineering at the University of California, Davis. His research interests include high-speed processor architectures and VLSI design. Truong has a BS in electrical and computer engineering from the University of California, Davis.

Jason Cheung is a software developer at Seven Networks. His research interests are in many-core compilers. Cheung has a BS in computer science and engineering from the University of California, Davis.

Direct questions and comments about this article to Bevan Baas, Department of Electrical and Computer Engineering, University of California, Davis, California 95616; bbaas@ucdavis.edu.