AsAP: An Asynchronous Array of Simple Processors

Zhiyi Yu, Michael J. Meeuwsen, Ryan W. Apperson, Omar Sattari, Michael Lai, Jeremy W. Webb, Eric W. Work, Dean Truong, Tinoosh Mohsenin, and Bevan M. Baas

***Abstract*—An array of simple programmable processors is implemented in 0.18 μm CMOS and contains 36 asynchronously clocked independent processors. Each processor occupies 0.66 mm^2 and is fully functional at a clock rate of 520–540 MHz at 1.8 V and over 600 MHz at 2.0 V. Processors dissipate an average of 32 mW under typical conditions at 1.8 V and 475 MHz, and 2.4 mW at 0.9 V and 116 MHz while executing applications such as a JPEG encoder core and a fully compliant IEEE 802.11a/g wireless LAN baseband transmitter.**

***Index Terms*—Array processor, chip multi-processor, digital signal processing, DSP, GALS, globally asynchronous locally synchronous, many-core, MIMD, multi-core.**

I. Introduction

APPLICATIONS that require the computation of complex DSP workloads are becoming increasingly commonplace. These applications often comprise multiple DSP tasks and are frequently key components in many systems such as: wired and wireless communications, multimedia, remote sensing and processing, and biomedical applications. Many of these applications are embedded and are strongly energy and cost-constrained. In addition, many of them require very high throughputs, often dissipate a significant portion of the system power budget, and are therefore of considerable interest. Therefore, the key challenges in DSP and embedded processor design are in maximizing performance (often throughput), minimizing energy dissipation per operation, and minimizing silicon area (cost).

Increasing clock frequencies and increasing numbers of circuits per chip has resulted in modern chip performance being limited by power dissipation rather than circuit constraints. This implies a new era of high-performance design that must now focus on energy-efficient implementations. Furthermore, future fabrication technologies are imposing new challenges such as large circuit parameter variations, and wire delays which may significantly reduce maximum clock rates. Therefore, architectures that address the challenges and exploit the advantages of future fabrication technologies are worthy of special consideration.

There are several design approaches for DSP processors. ASICs can provide very high performance and very high energy efficiency, but they have little programming flexibility. On the other hand, programmable DSPs are easy to program but their performance and energy efficiency are much lower. FPGAs fall somewhere in between. The goal of the Asynchronous Array of simple Processors (AsAP) project is to develop a system that computes complex DSP application workloads with high performance and high energy efficiency, is well suited for implementation in future fabrication technologies, and maintains the flexibility and programming ease of a programmable processor.

The AsAP system comprises a 2-D array of simple programmable processors interconnected by a reconfigurable mesh network [1]. Processors are each clocked by fully independent haltable oscillators in a Globally Asynchronous Locally Synchronous (GALS) [2] scheme. The multi-processor architecture efficiently makes use of task level parallelism in many complex DSP applications, and also efficiently computes many large DSP tasks through fine-grain parallelism (i.e., large numbers of processors computing a fine-grain partitioning of the workload) to achieve high performance.

AsAP uses a simple processor architecture with small memories to dramatically increase energy efficiency. The flexible programmable processor architecture broadens the target application domain and allows high one-time fabrication costs to be shared among a variety of applications. The GALS clocking style and nearest-neighbor communication greatly enhance scalability, and provide opportunities to mitigate effects of device variations, global wire limitations, and processor failures. A prototype 6×6 AsAP chip has been implemented in 0.18 μm CMOS and is fully functional [3].

This paper is organized as follows. Section II introduces the key AsAP processor features and Section III details the AsAP design. Section IV presents measured results from the fabricated chip, and Sections V and VI discuss related work and conclude the paper.

II. Motivation and Key Features of AsAP Processor

Several key features distinguish the AsAP processor. These features and the resulting benefits are illustrated in Fig. 1 and are discussed in greater detail in the following subsections.

A. Chip Multiprocessor and Task Level Parallelism

Increasing the clock frequency of processors has worked well for increasing performance but recently has become significantly more challenging. Advanced CPUs already consume more than 100 W operating over 2 GHz [4]. Cooling costs for such chips limit the acceptable power consumption and also the achievable clock frequency and processor performance [5]. The technique of increasing the clock frequency by deeper pipeline stages is also reaching its limit, since it requires more registers

Manuscript received June 20, 2006; revised October 31,2007. This work was supported by Intel, UC Micro, National Science Foundation Grant 0430090 and CAREER Award 0546907, Semiconductor Research Corporation GRC Grant 1598, IntellaSys, S Machines, MOSIS, Artisan, ST Microelectronics, and a UCD Faculty Research Grant.

The authors are with the Department of Electrical and Computer Engineering, University of California, Davis, CA 95616 USA (e-mail: zhyyu@ece.ucdavis.edu).

Digital Object Identifier 10.1109/JSSC.2007.916616

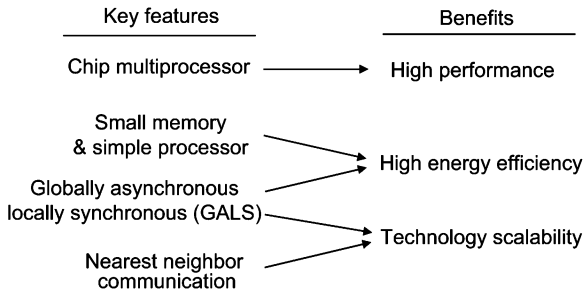


Fig. 1. Key features of AsAP and resulting benefits.

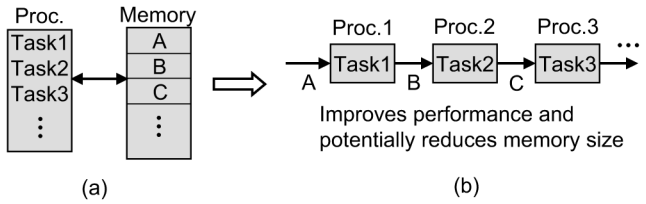


Fig. 2. Multi-task application executing on (a) a traditional architecture and (b) a stream-oriented multi-processor well suited for task level parallelism.

and control logic, thereby further increasing design difficulty and lowering energy efficiency.

Parallelization, rather than increased clock frequency, is another strategy to increase system performance with techniques such as instruction level parallelism, data level parallelism and task level parallelism.

Task level parallelism is especially well suited for many DSP applications, but unfortunately it can not be easily used on traditional sequentially executing processors. As shown in Fig. 2(a), the traditional system normally contains a powerful processor with a large memory, and executes the tasks of the application in sequence and stores temporary results into memory. The same application may be able to run on multiple processors using task level parallelism more efficiently as shown in Fig. 2(b), where different processors handle different tasks of the application. Normally the data input of DSP applications is considered of infinite length, so these processors can execute in parallel and achieve high performance. Also, the temporary results from each processor can be sent to the following processor directly and do not need to be stored in a large global memory, so less memory is necessary compared to the traditional method.

Task level parallelism is widely available in many DSP applications. Fig. 3 shows an example of a modern complex application that exhibits abundant task-level parallelism—the transmit chain of an IEEE 802.11a/g wireless LAN transmitter. It contains more than 10 tasks, and each of them can be directly mapped to separate processors to take advantage of the available task level parallelism.

B. Memory Requirements of the Targeted Tasks

With an ever increasing number of transistors possible per die, modern programmable processors typically use not only an increasing amount of on-chip memory, but also an increasing percentage of die area for memory. Fig. 4 shows the area breakdown of four modern processors [6], [4], [7], [8] with memories that occupy 55% to 75% of the processor's area. Large memories reduce the area available for execution units, consume significant power, and require larger delays per memory transaction. Therefore, architectures that minimize the need for memory and keep data near or within processing elements can increase area efficiency, performance, and energy efficiency.

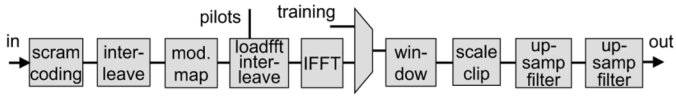


Fig. 3. IEEE 802.11a/g wireless LAN (54 Mb/s, 5/2.4 GHz) baseband transmit path.

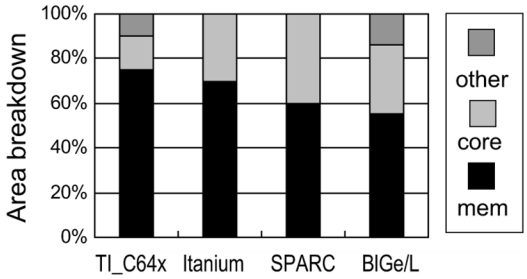


Fig. 4. Area breakdown for four modern processors.

TABLE I
MEMORY REQUIREMENTS FOR COMMON DSP TASKS ASSUMING A SIMPLE SINGLE-ISSUE PROCESSOR

Task	Instruction Mem requirement (words)	Data Mem requirement (words)
N-point FIR	6	$2N$
8-point DCT	40	16
8×8 2-D DCT	154	72
Conv. coding ($k = 7$)	29	14
Huffman encoder	200	350
N-point convolution	29	$2N$
64-point complex FFT	97	192
Bubble sort	20	1
N merge sort	50	N
Square root	62	15
Exponential	108	32

A notable characteristic of the targeted DSP and embedded tasks is that many have very limited memory requirements compared to general-purpose tasks. The level of *required* memory must be differentiated from the amount of memory that *can* be used or *is* typically used to calculate these kernels. Table I lists the actual amounts of instruction and data memory required for several tasks commonly found in DSP applications. These numbers assume a simple single-issue fixed-point processor. The data show that several hundred words of memory are enough for many DSP and embedded tasks—far smaller than the 10 KBytes to 10 MBytes per processing element typically found in modern DSP processors. Reducing memory sizes can result in significant area and power savings.

C. GALS Clocking Styles

A globally synchronous clock style is normally used in modern integrated circuits. But with the larger relative wire delays and larger parameter variations of deep-submicron

technologies, it has become increasingly difficult to design both large chips, and chips with high clock rates. Additionally, high speed global clocks consume a significant portion of power budgets in modern processors. For example, 1/4 of the total power dissipation in the recent 2-core Itanium [4] is consumed by clock distribution circuits and the final clock buffer. Also, the synchronous style lacks the flexibility to independently control the clock frequency among system sub-components to achieve increased energy efficiency.

The opposite clock style of globally synchronous—fully asynchronous—has the potential for speed and power improvements, but currently lacks EDA tool support, is difficult to design, and has large circuit overhead which reduces its efficiency.

The GALS [2] clocking style separates processing blocks such that each part is clocked by an independent clock domain. Its use enables the possibility of eliminating global clock distribution completely which brings power and design complexity benefits. Another significant benefit of GALS is the opportunity to easily and completely shut off a circuit block's clock (not just portions of the clock tree as with clock gating) when there is no work to do. Additionally, independent clock oscillators permit independent clock frequency scaling, which can dramatically reduce power dissipation in combination with supply voltage scaling [9].

D. Wires and On Chip Communication

A considerable challenge is presented by the increasing power dissipation and delay caused by on-chip communication (wires). As Ho *et al.* report [10], global chip wires will dramatically limit performance in future fabrication technologies if not properly addressed since their delay is roughly constant with technology scaling—which leads to an increasing percentage of clock cycle time. A number of architectures have specifically addressed this concern [11]–[13]. Therefore, architectures that enable the elimination of long high-speed wires will likely be easier to design and may operate at higher clock rates [10].

There are several methods to avoid global wires. Networks on Chip (NoC) [14] treat different modules in a chip as different nodes in a network and use routing techniques instead of simple wire links and buses to transfer data. NoCs provide a powerful communication method, but often consume large amounts of area and power. Another method is local communication, where each processor connects only to processors within a local domain. One of the simplest examples is nearest neighbor communication, where each processor directly connects and communicates only to immediately adjacent processors. This architecture has high area and energy efficiency per processor, and can also provide sufficient communication for many DSP applications, especially those that are stream-like [15]. The greatest challenge when using nearest-neighbor interconnects is efficiently mapping applications that exhibit significant long-distance communication. Fortunately, for many applications—including embedded and complex DSP applications—nearest neighbor inter-processor communication is highly effective.

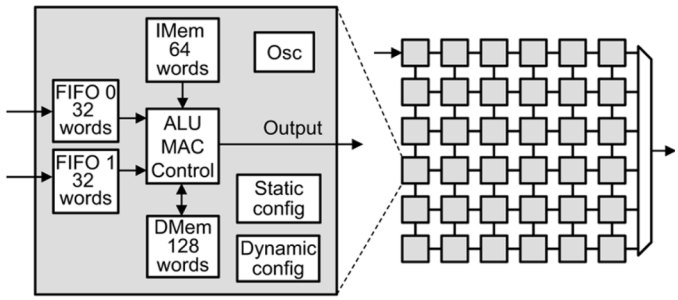


Fig. 5. Block diagram of an AsAP processor.

III. The AsAP Processor System

The AsAP system comprises a 2-D array of simple programmable processors. Each processor is clocked in a GALS fashion and interconnected by a reconfigurable mesh network. AsAP processors are optimized to efficiently compute DSP algorithms individually as well as in conjunction with neighboring processors.

Fig. 5 shows a block diagram of an AsAP processor and the fabricated processing array. The array is organized as a 6 by 6 mesh. Data enters the array through the top left processor and exits through one of the right column processors, selected by a mux. Input and output circuits are available on each edge of all periphery processors but most are unconnected in this test chip due to package I/O limitations.

Each processor is a simple single-issue processor, and contains: a local clock oscillator; two dual-clock asynchronous interfaces to provide communication with other processors; and a simple CPU including ALU, MAC, and control logic. Each processor contains a 64-word instruction memory and a 128-word data memory. They also contain static and dynamic configuration logic to provide configurable functions such as addressing modes and interconnections with other processors. Each processor can receive data from any two neighbors and can send data to any combination of its four neighbors. Each processor contains two input ports because it meshes well with the data flow graphs of the applications we have studied. Clearly, two or more input ports are required to support graph fan-in and we found a third input port was not frequently used. AsAP supports 54 RISC style instructions. Other than the bit-reverse instruction which is useful for the calculation of the Fast Fourier Transform (FFT), no algorithm-specific instructions are implemented.

The AsAP processor system utilizes a GALS clocking style with a local clock oscillator inside each processor. The maximum span of the clock tree is less than 1 mm in 0.18 μm technology—the distance across a single processing element. This approach has excellent scalability and allows the simple addition of more processors to the array. In a synchronous system, the global clock tree must be redesigned when adding more processors, which can be very difficult for large chips.

A. Single AsAP Processor Design

1) Pipelining and Datapath: Each AsAP processor has a nine stage pipeline as shown in Fig. 6. The *IFetch* stage fetches instructions according to the program counter (PC). No branch prediction circuits are implemented. All control signals are

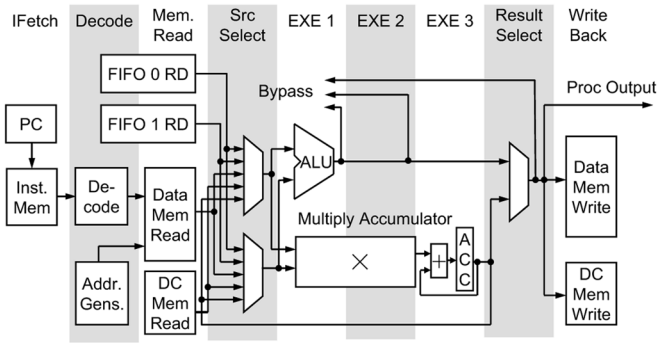


Fig. 6. AsAP 9-stage pipeline.

generated in the *Decode* stage and pipelined appropriately. The *Mem Read* and *Src Select* stages fetch data from the data memory (DMEM), immediate field, the asynchronous FIFO interface from other processors, dynamic configuration memory (DCMEM), the ACC accumulator register, or ALU/MAC forwarding logic. The execution stages occupy three cycles, and bypass logic is used for the ALU and MAC to alleviate data hazard pipeline penalties. The *Result Select* and *Write Back* stages select results from the ALU or MAC unit, and write the result to data memory, DC memory, or neighboring processors. To simplify pre-tapeout verification, pipeline interlocks are not implemented in hardware, and all code is scheduled prior to execution by the programmer or compiler.

The MAC unit is divided into three stages to enable a high clock rate as well as the capability of issuing MAC and multiply instructions every cycle. Fig. 7 shows a block diagram of the MAC unit. The first stage generates the partial products of the 16×16 multiplier. The second stage uses carry-save adders to compress the partial products into a single 32-bit carry-save output. The final stage contains a 40-bit adder to add the results from the second stage to the 40-bit accumulator register (ACC). Because the ACC is normally read infrequently, only the least-significant 16 bits of the ACC are readable. More significant ACC bits are read by shifting those bits into the 16 LSBs. This simplification reduces hardware and relaxes timing in the final MAC unit stage which is the block's critical pipeline stage.

2) Local Oscillator: Fig. 8 contains a simplified schematic of the programmable local oscillator which provides the clock to each processor. The oscillator is an enhanced ring oscillator and is built entirely with standard cells. Three methods are used to configure the frequency of the oscillator. First, the ring size can be configured to 5 or 9 stages using the configuration signal *stage_sel*. Second, seven tri-state inverters are connected in parallel with each inverter. When a tri-state inverter is turned on, that stage's current drive increases, and the ring's frequency increases [16]. Third, a clock divider at the output divides the clock from 1 to 128 times. The *halt* signal and the SR latch allow the oscillator to cleanly halt when the processor stalls without any partial clock pulses.

Processors must stall when they attempt to read data from an empty FIFO or write data to a full FIFO, to maintain correct operation. The clock oscillator can be configured to halt during stalls so that the processor consumes no power whatsoever except leakage while stalled. Fig. 9 shows an example waveform

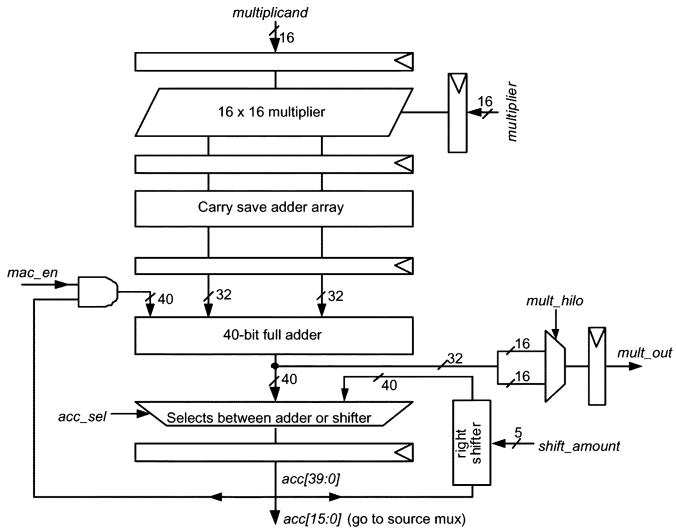


Fig. 7. Block diagram for the three-stage MAC unit.

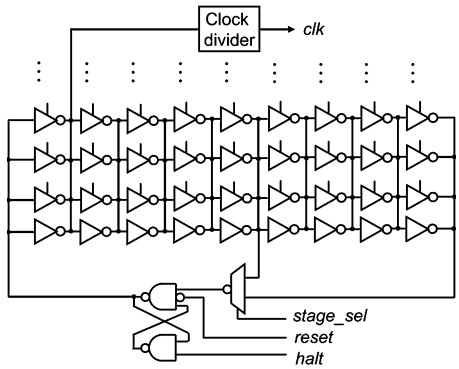


Fig. 8. Programmable clock oscillator: an inverter ring with configurable tri-state inverters, ring size and frequency divider.

for clock halting and restarting. Signal *stall_fifo* is asserted when FIFO access is stalled due to either an empty input or full output condition. After a nine clock cycle period, during which the processor's pipeline is flushed, the signal *halt_clk* (same as *halt* in Fig. 8) goes high which halts the clock oscillator. The signal *stall_fifo* returns low when the cause of the stall has been resolved (when data is put into the empty input FIFO or data is removed from the full output FIFO); then *halt_clk* restarts the oscillator at full speed in less than one clock period. Using this method, power is reduced by 53% and 65% for a JPEG encoder and a 802.11a/g transmitter application respectively, by making active power dissipation equal to zero during periods when processors have no work to perform.

Figs. 10(a) and (b) show measured oscillator frequencies for the 5-inverter and 9-inverter rings respectively. Over all possible configurations, the oscillator has 524 288 frequency settings and its frequency range is 1.66 MHz to 702 MHz as shown in Fig. 10(c). Figure 10(d) shows the number of occurrences of different frequency gaps between settings within the useful frequency range of 1.66 to 500 MHz. In this useful range, approximately 99% of the frequency gaps are smaller than 0.01 MHz, and the largest gap is 0.06 MHz.

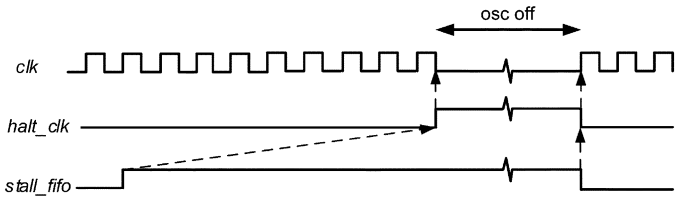


Fig. 9. Example waveform of clock halting and restarting.

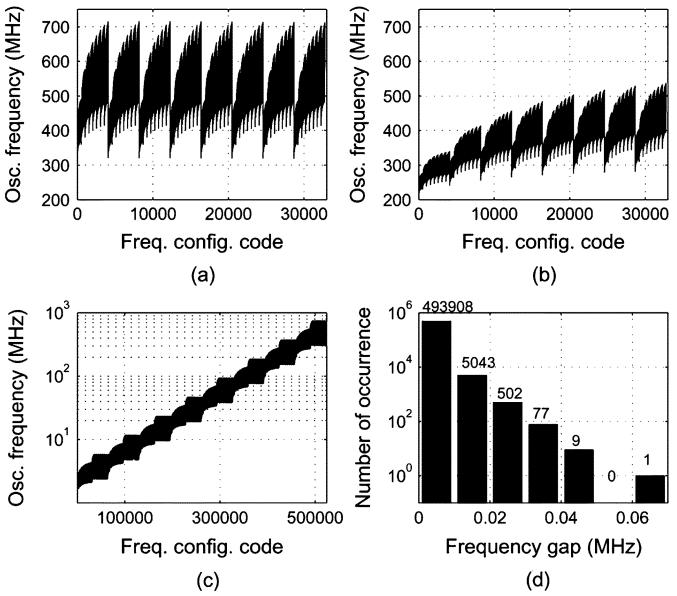


Fig. 10. Measured oscillator data for a single processor: (a) frequencies for the 5-inverter ring, (b) frequencies for the 9-inverter ring, (c) frequencies over all configuration possibilities, and (d) number of occurrences at different frequency gaps.

Despite the fact that the layout for all processors is exactly the same, process variation causes different processors on the same die to perform differently than others. Fig. 11 shows measured oscillator frequencies taken at the same configuration setting on the same chip. The oscillator located in the bottom right processor has a frequency greater than 540 MHz, while several oscillators in the top row have frequencies less than 500 MHz. By its very nature of being a truly GALS processor array, this design is free from any new functionality issues whatsoever not present in standard clocked digital systems (other than possible metastability at clock domain crossings which can be reduced to extremely low levels through configurable synchronization registers) in either hardware (including the FIFO buffers) or software due to processor clock frequency variations from jitter, skew, halting, restarting, frequency changes, or process variations. The only impact of variations is a possible change in throughput.

B. Design of Inter-Processor Communication

The AsAP architecture connects processors via a configurable 2-D mesh as shown in Fig. 12. To maintain link communication at full clock rates, inter-processor connections are made to nearest-neighbor processors only. Each processor has two asynchronous input data ports and can connect each port to any of its four nearest neighboring processors. The

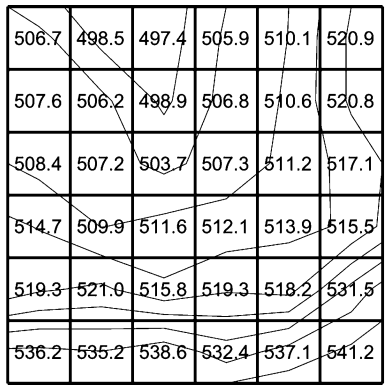


Fig. 11. Physical distribution of measured oscillator frequencies across different processors with the same configuration. Data are given in MHz with contour lines from 500 MHz to 540 MHz in 5 MHz steps.

input connections of each processor are normally defined during the configuration sequence after powerup. The output port connections can be changed among any combination of the four neighboring processors at any time through software. Input ports are read and output ports written through reserved program variables and inter-processor timing is in fact invisible to programs without explicit software synchronization. AsAP's nearest neighbor connections result in no high-speed wires with a length greater than the linear dimension of a processing element. Since inter-processor links are extremely short local wires and not global wires, inter-processor delay decreases with advancing fabrication technologies and allows clock rates to scale upward. Data transfers between distant AsAP processors are handled by routing through intermediary processors—physical links are still only among nearest neighbors, however.

The reliable transfer of data across unrelated asynchronous clock domains is accomplished by mixed-clock-domain FIFOs. The dual-clock FIFOs read and write in fully independent and haltable clock domains without restrictions other than minimum cycle times. No special circuits are required other than the ones shown. The FIFO block was entirely synthesized, and automatically placed and routed. A block diagram of the FIFO's major components is shown in Fig. 13. The FIFO's write clock and write data are supplied in a source-synchronous fashion by the upstream processor and the FIFO's read clock is supplied by the downstream processor. In AsAP, the dual-clock FIFO resides in the downstream processor to simplify the physical design. Values are read from only the head of the FIFO, and this is presented to software as a single operand source. The read and write addresses are transferred across the asynchronous interface, and are used to decide if the FIFO is full or empty. Configurable synchronization registers are inserted in the asynchronous interface to alleviate metastability. In order to avoid changing multiple address bit values at the same time across the asynchronous interface, the addresses are gray coded when transferred across the clock domain boundary [17].

C. Implementation of AsAP

The AsAP processor is implemented in 0.18 μm TSMC standard CMOS using Artisan standard cells. The chip is fully synthesized from verilog, except the clock oscillator which was designed by hand from standard cells. The IMem, DMem, and two

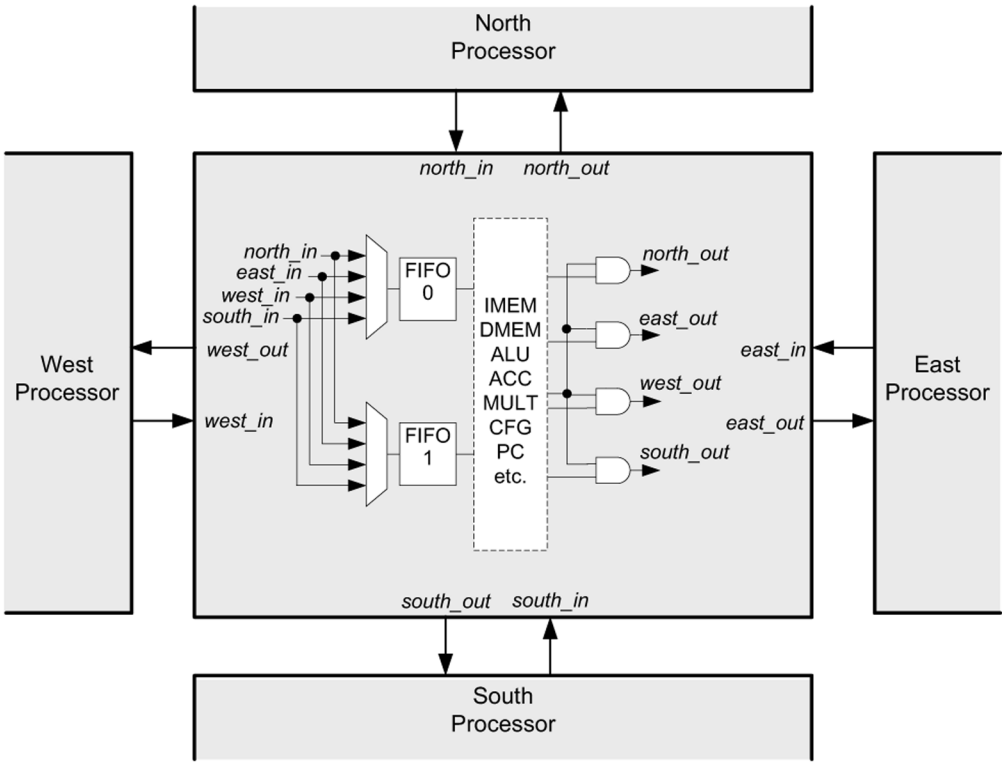


Fig. 12. Nearest neighbor inter-processor communication diagram.

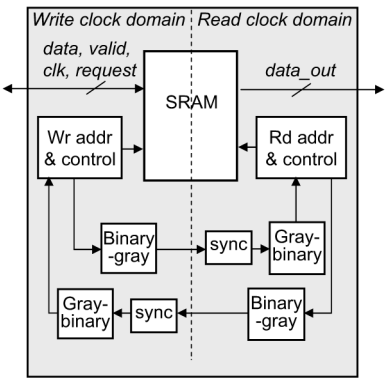


Fig. 13. Block diagram of the dual-clock FIFO used for asynchronous boundary communication.

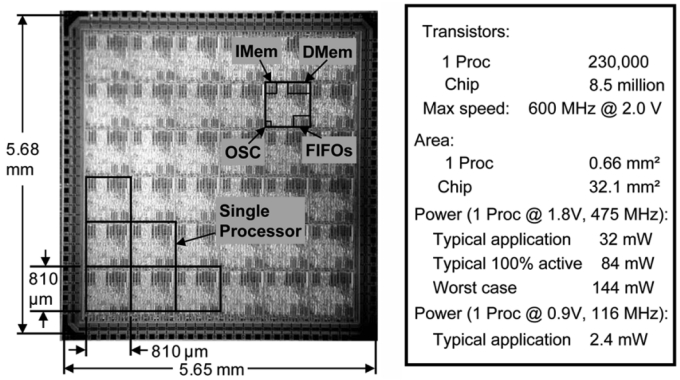


Fig. 14. Micrograph of the 6 × 6 AsAP chip.

FIFO memories are built using memory macro blocks. The design process was done in two phases. First, a single processor was auto placed and routed including power grid design, clock tree insertion, and several passes of in-place optimization to increase speed and reduce wiring congestion. Second, processors were arrayed across the chip, and the small amount of global circuitry was auto placed and routed around the array. Processors nearly abut with very short wires between them. Fig. 14 shows the die micrograph of the first generation 6 × 6 AsAP array processor. Each processor contains 230 000 transistors.

For testing purposes, 27 critical signals from each of the 36 processors can be selectively routed to eight chip pads for real-time viewing of these key signals which include: clocks, stall signals, FIFO signals, program counter, etc. Figure 15 shows the test environment for the AsAP prototype including a printed circuit board hosting an AsAP processor and a supporting FPGA board to interface between AsAP's configuration and data ports and a host computer. There is one SPI style serial port designed in the AsAP processor which receives external information and commands for configuration and programs.

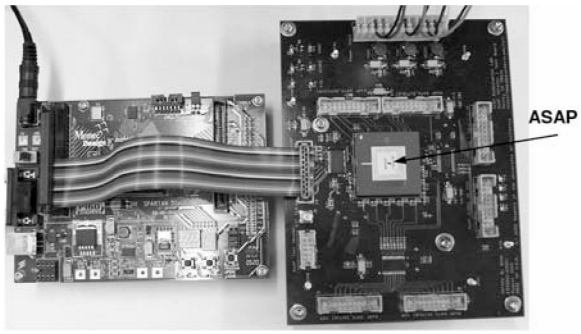


Fig. 15. AsAP board and supporting FPGA-based test board.

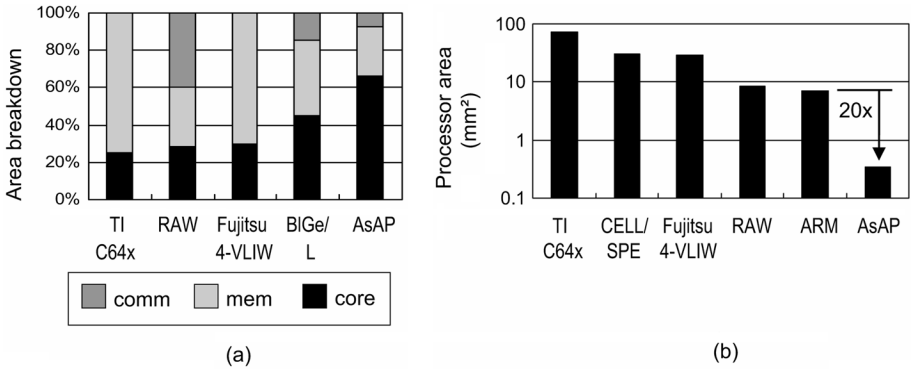


Fig. 16. Area evaluation of AsAP processor and several other processors; with technology scaled to 0.13 μm.

TABLE II
AREA BREAKDOWN IN A SINGLE PROCESSOR

	Area (μm^2)	Area percentage
Core and others	433,300	66.0 %
Data memory	115,000	17.5 %
Instruction mem.	56,500	8.6 %
Two FIFOs	48,000	7.4 %
Oscillator	3,300	0.5 %
Single processor	656,100	100.0 %

IV. MEASUREMENT AND EVALUATION

A. Testing and Measurement Results

1) Area: Due to its small memories and simple communication scheme, each AsAP processor devotes most of its area to the execution core and thereby achieves a high area efficiency. Table II shows the area breakdown for each AsAP processor. Each one dedicates 8% to communication circuits, 26% to memory circuits, and a favorable 66% to the core. These data compare well to other processors [6], [12], [18], [8], [19], [20], as shown in Fig. 16(a), since the other processors use 20% to 45% of their area for the core. Each AsAP processor occupies 0.66 mm^2 and the 6×6 array occupies 32.1 mm^2 including pads, global power rings, and a small amount of chip-level circuits. Fig. 16(b) compares the area of several processors scaled to 0.13 μm, assuming area reduces as the square of the technology's minimum feature size. The AsAP processor is 20 to 210 times smaller than these other processors.

2) Power and Performance: The fabricated processors run at 520–540 MHz at a supply voltage of 1.8 V and over 600 MHz at 2.0 V. The shmoo plot of Fig. 17 shows processor operation as a function of supply voltage and clock speed. Since AsAP processors dissipate zero active power when idle for even brief periods of time (such as is common in complex applications), and because different instructions dissipate varying amounts of power, it is useful to consider several power measurements. The average processor power while executing the JPEG encoder and 802.11a/g transmitter applications is 32 mW at 475 MHz. Processors that are 100% active and executing a "typical application" mix of instructions dissipate 84 mW each at 475 MHz. The absolute worst case power per processor at 475 MHz is 144 mW and occurs when using the MAC instruction with all memories active every cycle.

At a supply voltage of 0.9 V, processors run at 116 MHz and the typical application power is only 2.4 mW.

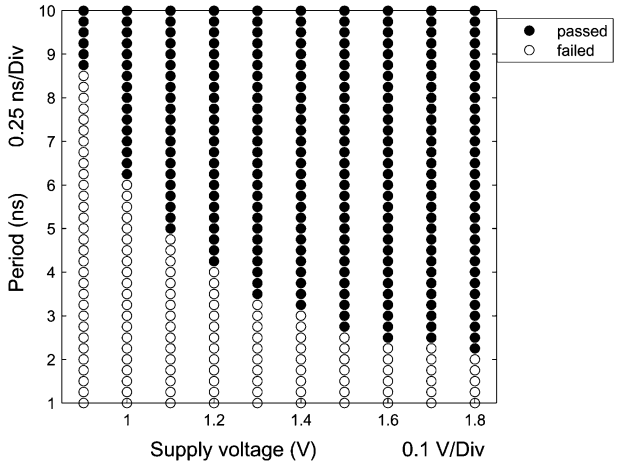


Fig. 17. Processor shmoo: voltage versus speed.

Approximately 2/3 of AsAP's power is dissipated in its clocking system. This is largely due to the fact that clock gating is not implemented in this first design, so the future addition of even coarse levels of clock gating (distinguished from oscillator halting) are expected to significantly reduce power consumption further.

Fig. 18 compares the peak performance density and energy efficiency of several processors [9], [18], [20], [12], [6]. All data are scaled to 0.13 μm technology. Energy efficiency is defined as the power divided by the clock frequency with a scale factor to compensate for multiple issue architectures. These processors have large differences that are not taken into account by these simple metrics—such as word width and workload—so this comparison is only approximate. The AsAP processor has a high peak performance density that is 7 to 30 times higher than the others. Also, the AsAP processor has a low power per operation that is 5 to 15 times lower than the others.

If the exact AsAP design presented here were scaled to a 90 nm technology, a 13 mm by 13 mm chip would yield over 1000 processors, have a peak computation rate of 1 TeraOp/sec, but dissipate only 10 W typical application power in addition to leakage. The fine-grain structure of the AsAP design could provide opportunities for leakage and active power reduction in the common case when loads are unbalanced across processors, and would be able to leverage techniques commonly used

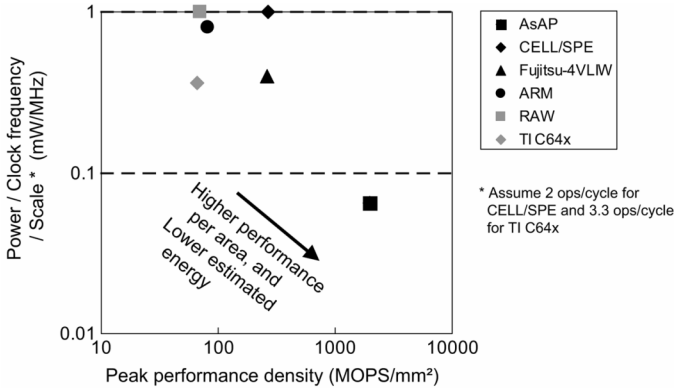


Fig. 18. Power and performance evaluation of AsAP processor and several other processors; with technology scaled to 0.13 μm.

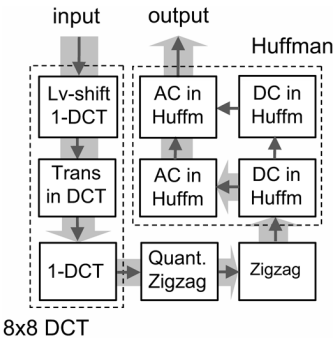


Fig. 19. JPEG encoder core using nine processors; thin arrows show all paths and wide arrows show the primary data flow.

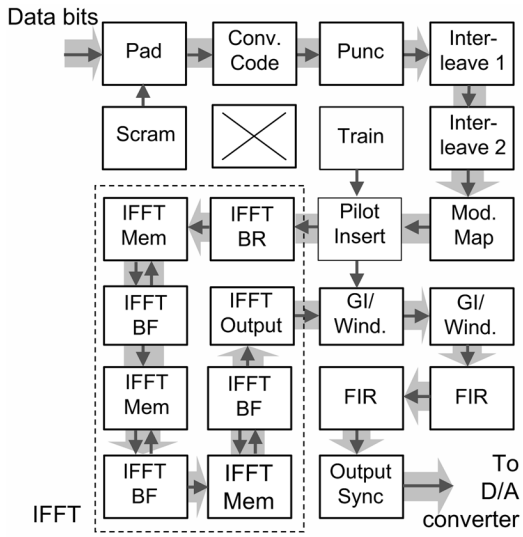


Fig. 20. 802.11a/g transmitter implementation using 22 processors; thin arrows show all paths and wide arrows show the primary data flow.

TABLE III
COMPARISON OF JPEG ENCODER (9 ASAP PROCESSORS) AND IEEE 802.11A/11G WIRELESS LAN TRANSMITTER (TI C62X NON-COMPLETE, 22 ASAP PROCESSORS COMPLETE IMPLEMENTATION) SOFTWARE IMPLEMENTATIONS

	TI C62x		AsAP	
	Performance	Energy	Performance	Energy
JPEG encoder	3.90 μsec/blk	12.4 μJ/blk	4.65 μsec/blk	1.04 μJ/blk
IEEE 802.11a/g tx	1.7 Mbps	1880 nJ/bit	16.2 Mbps	25.1 nJ/bit

in much larger processors (e.g., by reducing the supply voltage to a region of lightly used AsAP processors). While this design contains no special circuits or architectural features to reduce leakage, assuming a constant transistor density, AsAP's leakage power would scale lower roughly at the same rate as its 20 to 210 times lower area compared to the processors shown in Fig. 16, when implemented in deep-submicron technologies.

B. Software and Application Implementations

While the previous benchmark results are useful, most are based on simple metrics and do not take into account application performance. In addition to the tasks listed in Table I, we also implemented and further analyzed several complex applications, including a JPEG encoder core and an 802.11a/g baseband transmitter [21] which were written in assembly code by hand, were lightly optimized, and had to be written in parallel tasks to map to the array.

1) JPEG Encoder: Fig. 19 shows a JPEG encoder core using nine processors. Three processors compute the Level Shift and an 8×8 DCT, and four processors implement a Huffman encoder. All processors consume 224 mW at a clock rate of 300 MHz. Processing each 8×8 block requires approximately 1400 clock cycles. Compared to one implementation on a TI C62x 8-way VLIW DSP processor, AsAP has similar performance, and 11 times lower power consumption [22], [23].

2) 802.11a/g Transmitter: Fig. 20 shows a fully-compliant IEEE 802.11a/g wireless baseband transmitter using 22 processors [24]. Data enters the transmitter in the upper left, flows through a number of frequency-domain tasks, the OFDM 64-point complex IFFT, a number of time-domain tasks, and finally to a synchronization processor whose only purpose is to enable the asynchronous array to be directly connected to a digital-to-analog (D/A) converter with no other necessary logic.

The implementation consumes 407 mW at 300 MHz and achieves 30% of full rate at 54 Mb/s despite that fact the code is lightly optimized and unscheduled. This result compares well to a previously published non-complete implementation on a TI C62x 8-way VLIW DSP [25], [23]—AsAP achieves 5 to 10 times higher performance with 35 to 75 times lower energy dissipation—depending on details of the application conditions. Data are summarized in Table III.

V. RELATED WORK

Most parallel processor systems can be easily differentiated by their processing element architectures. They can be categorized into three broad types: 1) heterogeneous such as Pleiades [26], 2) SIMD or SIMD-like such as Imagine [15] and PipeRench [27], and 3) homogeneous MIMD. AsAP is a homogeneous MIMD-style machine and can be distinguished from other MIMD systems such as Smart Memories [28], RAW [12], TRIPS [13], and Synchroscalar [29], by processing element granularity alone. Smart Memories contains 64-bit processors with two integer clusters, one FPU cluster, and 128 KB of memory. Each RAW processor contains 32 KB instruction memory and 32 KB Dcache. The TRIPS processor contains

large multiple-issue processors. Individual Synchroscalar processing elements are SIMD processors.

Clocking style is another feature that distinguishes AsAP from other projects. Most of the other implementations use globally synchronous clocking styles. Pleiades and FAUST [30] use a handshaking GALS clocking style, which requires acknowledgment of each transaction and is therefore significantly different from the source-synchronous interprocessor communication used in AsAP—which is able to sustain a full-rate communication of one word per clock cycle, albeit with longer latencies. The Intel 80-core processing array [31] employs mesochronous clocking where each processor operates at the same clock frequency with a varying clock phase.

The inter-processor network is another key feature in AsAP. Smart Memories and RAW use mesh-connected structures, but contain sophisticated networks to route data. The Transputer [32] and Systolic processors [11] share the same idea of nearest-neighbor communication with AsAP, but Transputer uses a bit serial communication channel, and systolic processors send and receive data in a highly regular manner. Many other architectures have used the 2-D mesh or 2-D toroidal mesh such as the MasPar MP-1 [33].

VI. Conclusion

The AsAP scalable programmable processor array targets DSP and embedded applications and features a simple chip multiprocessor architecture with small memories, GALS clocking style, and nearest neighbor communication. These and other features make AsAP well-suited for future fabrication technologies, and for the computation of complex multi-task DSP workloads.

The AsAP processing array is implemented in 0.18 μm CMOS, and runs at 520–540 MHz at 1.8 V and over 600 MHz at 2.0 V. Each highly energy-efficient processor dissipates 32 mW while executing applications, and 84 mW when 100% active at 475 MHz. It achieves a high performance density of over 910 peak MOPS per mm^2.

Acknowledgment

The authors thank M. Singh, R. Krishnamurthy, M. Anders, S. Mathew, S. Muroor, W. Li, and C. Chen.

References

[1] B. M. Baas, "A parallel programmable energy-efficient architecture for computationally-intensive DSP systems," in *37th Asilomar Conf. Signals, Systems and Computers*, Nov. 2003.

[2] D. M. Chapiro, "Globally-asynchronous locally-synchronous systems," Ph.D. dissertation, Stanford Univ., Stanford, CA, Oct. 1984.

[3] Z. Yu, M. Meeuwsen, R. Apperson, O. Sattari, M. Lai, J. Webb, E. Work, T. Mohsenin, M. Singh, and B. Baas, "An asynchronous array of simple processors for DSP applications," in *IEEE ISSCC Dig. Tech. Papers*, 2006, pp. 428–429.

[4] S. Naffziger, T. Grutkowski, and B. Stackhouse, "The implementation of a 2-core multi-threaded Itanium family processor," in *IEEE ISSCC Dig. Tech. Papers*, 2005, pp. 182–183, 592.

[5] M. Horowitz and W. Dally, "How scaling will change processor architecture," in *IEEE ISSCC Dig. Tech. Papers*, 2004, pp. 132–133.

[6] S. Agarwala, M. D. Ales, R. Damodaran, P. Wiley, S. Mullinnix, J. Leach, A. Lell, M. Gill, A. Rajagopal, A. Chachad, M. Agarwala, J. Apostol, M. Krishnan, D. Bui, Q. An, N. S. Nagaraj, T. Wolf, and T. T. Elappuparackal, "A 600-MHz VLIW DSP," *IEEE J. Solid-State Circuits*, vol. 37, no. 11, pp. 1532–1544, Nov. 2002.

[7] J. Hart, S. Choe, L. Cheng, C. Chou, A. Dixit, K. Ho, J. Hsu, K. Lee, and J. Wu, "Implementation of a 4th-generation 1.8 GHz dual-core SPARC v9 microprocessor," in *IEEE ISSCC Dig. Tech. Papers*, 2005, pp. 186–187.

[8] A. Bright, M. Ellavsky, A. Gara, R. Haring, G. Kopcsay, R. Lembach, J. Marcella, M. Ohmacht, and V. Salapura, "Creating the Bluegene/L supercomputer from low-power SoC AISCs," in *IEEE ISSCC Dig. Tech. Papers*, 2005, pp. 188–189.

[9] G. Semeraro, G. Magklis, R. Balasubramonian, D. H. Albonesi, S. Dwarkadas, and M. L. Scott, "Energy-efficient processor design using multiple clock domains with dynamic voltage and frequency scaling," in *Proc. Int. Symp. High-Performance Computer Architecture*, 2002, pp. 29–40.

[10] R. Ho, K. W. Mai, and M. A. Horowitz, "The future of wires," *Proc. IEEE*, vol. 89, no. 4, pp. 490–504, Apr. 2001.

[11] S. Y. Kung, "VLSI array processors," *IEEE ASSP Mag.*, vol. 2, no. 3, pp. 4–22, Jul. 1985.

[12] M. B. Taylor, J. Kim, J. Miller, D. Wentzlaff, F. Ghodrat, B. Greenwald, H. Hoffman, P. Johnson, W. Lee, A. Saraf, N. Shnidman, V. Stumpen, S. Amarasinghe, and A. Agarwal, "A 16-issue multiple-program-counter microprocessor with point-to-point scalar operand network," in *IEEE ISSCC Dig. Tech. Papers*, 2003, pp. 170–171.

[13] S. W. Keckler, D. Burger, C. R. Moore, R. Nagarajan, K. Sankaralingam, V. Agarwal, M. S. Hrishikesh, N. Ranganathan, and P. Shivakumar, "A wire-delay scalable microprocessor architecture for high performance systems," in *IEEE ISSCC Dig. Tech. Papers*, 2003, pp. 168–169.

[14] W. Dally and B. Towles, "Route packets, not wires: On-chip interconnection networks," in *IEEE Int. Conf. Design Automation*, Jun. 2001, pp. 684–689.

[15] J. H. Ahn, W. J. Dally, B. Khailany, U. J. Kapasi, and A. Das, "Evaluating the Imagine stream architecture," in *Proc. Int. Symp. Computer Architecture*, Jun. 2004, pp. 19–23.

[16] T. Olsson and P. Nilsson, "A digitally controlled PLL for SOC applications," *IEEE J. Solid-State Circuits*, vol. 39, no. 5, pp. 751–760, May 2004.

[17] R. W. Apperson, Z. Yu, M. J. Meeuwsen, T. Mohsenin, and B. M. Baas, "A scalable dual-clock FIFO for data transfers between arbitrary and haltable clock domains," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 15, no. 10, pp. 1125–1134, Oct. 2007.

[18] A. Suga, T. Sukemura, H. Wada, H. Miyake, Y. Nakamura, Y. Takebe, K. Azegami, Y. Himura, H. Okano, T. Shiota, M. Saito, S. Wakayama, T. Ozawa, T. Satoh, A. Sakutai, T. Katayama, K. Abe, and K. Kuwano, "A 4-way VLIW embedded multimedia processor," in *IEEE ISSCC Dig. Tech. Papers*, 2000, pp. 240–241.

[19] D. Pham, S. Asano, M. Bolliger, M. N. Day, H. P. Hofstee, C. Johns, J. Kahle, A. Kameyama, J. Keaty, Y. Masubuchi, M. Riley, D. Shippy, D. Stasiak, M. Suzuoki, M. Wang, J. Warnock, S. Weitzel, D. Wendel, T. Yamazaki, and K. Yazawa, "The design and implementation of a first-generation CELL processor," in *IEEE ISSCC Dig. Tech. Papers*, 2005, pp. 184–185.

[20] R. Witek and J. Montanaro, "StrongARM: A high-performance arm processor," in *Proc. IEEE Computer Society Int. Conf.: Technologies for the Information Superhighway (COMPCON)*, Feb. 1996, pp. 188–191.

[21] "Wireless LAN medium access control (MAC) and physical layer (PHY) specifications: High speed physical layer in the 5 GHz band," Standard for Information Technology, LAN/MAN Standard Committee of the IEEE Computer Society, New York, NY, 1999.

[22] T. Lin and C. Jen, "Cascade—configurable and scalable DSP environment," in *Proc. IEEE ISCAS*, 2002, pp. 26–29.

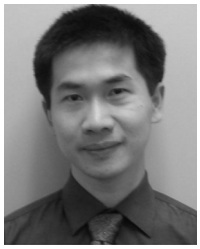
[23] C. Kozyrakis and D. Patterson, "Vector versus superscalar and VLIW architectures for embedded multimedia benchmarks," in *Proc. IEEE/ACM MICRO*, Nov. 2002, pp. 283–289.

[24] M. Meeuwsen, O. Sattari, and B. Baas, "A full-rate software implementation of an IEEE 802.11a compliant digital baseband transmitter," in *Proc. IEEE Workshop on Signal Processing Systems*, Oct. 2004, pp. 297–301.

[25] M. F. Tariq, Y. Baltaci, T. Horseman, M. Butler, and A. Nix, "Development of an OFDM based high speed wireless LAN platform using the TI C6x DSP," in *IEEE Int. Conf. Communications*, Apr. 2002, pp. 522–526.

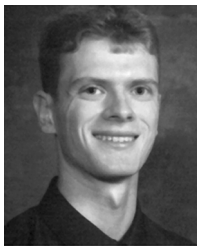
[26] H. Zhang, V. Prabhu, V. George, M. Wan, M. Benes, A. Abnous, and J. M. Rabaey, "A 1-V heterogeneous reconfigurable DSP IC for wireless baseband digital signal processing," *IEEE J. Solid-State Circuits*, vol. 35, no. 11, pp. 1697–1704, Nov. 2000.

[27] H. Schmit, D. Whelihan, A. Tsai, M. Moe, B. Levine, and R. R. Taylor, "PipeRench: A virtualized programmable datapath in 0.18 micron technology," in *Proc. IEEE Custom Integrated Circuits Conf.*, 2002, pp. 63–66.
[28] K. Mai, T. Paaske, and N. Jayasena *et al.*, "Smart memories: A modular reconfigurable architecture," in *Proc. Int. Symp. Computer Architecture*, Jun. 2000, pp. 161–171.
[29] J. Oliver, R. Rao, P. Sultana, J. Crandall, E. Czernikowski, L. W. Jones, D. Franklin, V. Akella, and F. T. Chong, "Synchroscalar: A multiple clock domain, power-aware, tile-based embedded processor," in *Proc. Int. Symp. Computer Architecture*, Jun. 2004.
[30] D. Lattard, E. Beigne, and C. Bernard *et al.*, "A telecom baseband circuit based on an asynchronous network-on-chip," in *IEEE ISSCC Dig. Tech. Papers*, 2007, pp. 258–259.
[31] S. Vangal, J. Howard, and G. Ruhl *et al.*, "An 80-tile 1.28 TFLOPS network-on-chip in 65 nm CMOS," in *IEEE ISSCC Dig. Tech. Papers*, 2007, pp. 98–99.
[32] C. Whitby-Strevens, "Transputers–past, present and future," *IEEE Micro*, vol. 10, no. 6, pp. 16–19, Dec. 1990.
[33] T. Blank, "The MasPar MP-1 architecture," in *Proc. 35th IEEE Computer Society Int. Conf.: Intellectual Leverage (COMPCON Spring'90)*, 1990, pp. 20–24.



Zhiyi Yu received the B.S. and M.S. degrees in electrical engineering from Fudan University, Shanghai, China, in 2000 and 2003, respectively, and the Ph.D degree in electrical and computer engineering from the University of California, Davis, in 2007.

Dr. Yu is currently a Hardware Engineer with IntellaSys Corporation, headquartered in Cupertino, CA. His research interests include high-performance and energy-efficient digital VLSI design, architectures, and processor interconnects, with an emphasis on many-core processors. He was a key designer of the 36-core Asynchronous Array of simple Processors (AsAP) chip, and one of the designers of the 150+ core second generation computational array chip.



Michael J. Meeuwsen received the B.S. degrees with honors in electrical engineering and computer engineering (both *summa cum laude*) from Oregon State University, Corvallis, and the M.S. in electrical and computer engineering from the University of California, Davis.

He is currently a Hardware Engineer with Intel Digital Enterprise Group, Hillsboro, OR, where he works on CPU hardware design. His research interests include digital circuit design and IEEE 802.11a/g algorithm mapping.

Ryan W. Apperson received the B.S. in electrical engineering (*magna cum laude*) from the University of Washington, Seattle, and the M.S. degree in electrical and computer engineering from the University of California, Davis.

He is currently an IC Design Engineer with Boston Scientific CRM Division, Redmond, WA. His research interests include multiclock domain systems and SRAM design.

Omar Sattari received the B.S. and M.S. degrees in electrical and computer engineering from the University of California, Davis.

He is currently a software engineer at CornerTurn. His research interests include FFT and DSP algorithms and digital hardware design.

Michael Lai received the B.S. and M.S. degrees in electrical and computer engineering from the University of California, Davis.

He is currently a Design Engineer at Altera Corporation working on the next generation transceiver product. His research interests include the design of high-speed arithmetic units and control.

Jeremy W. Webb received the B.S. degree in electrical and computer engineering from the University of California, Davis.

He is currently an M.S. student in electrical and computer engineering at the University of California, Davis, and a hardware engineer at Centellax. His research interests include high-speed board design and system interfacing.

Eric W. Work received the B.S. degree from the University of Washington, and the M.S. degree in electrical and computer engineering from the University of California, Davis.

He is currently an engineer at S Machine corporation. His research interests include the mapping of arbitrary task graphs to processor networks and software tool flow.

Dean Truong received the B.S. degree in electrical and computer engineering from the University of California, Davis.

He is currently a Ph.D. student in electrical and computer engineering at the University of California, Davis. His research interests include high-speed processor architectures and VLSI design.

Tinoosh Mohsenin received the B.S. degree in electrical engineering from Sharif University, Tehran, Iran, and the M.S. degree in electrical and computer engineering from Rice University, Houston, TX. She is currently pursuing the Ph.D. degree in electrical and computer engineering from the University of California, Davis.

She is the designer of the Split-Row and Multi-Split-Row Low Density Parity Check (LDPC) decoding algorithms. Her research interests include energy-efficient and high-performance signal processing and error correction architectures including multi-gigabit full-parallel LDPC decoders and many-core processor architecture design.

Bevan M. Baas received the B.S. degree in electronic engineering from California Polytechnic State University, San Luis Obispo, in 1987, and the M.S. and Ph.D. degrees in electrical engineering from Stanford University, Stanford, CA, in 1990 and 1999, respectively.

In 2003 he became an Assistant Professor with the Department of Electrical and Computer Engineering, University of California, Davis. He leads projects in architecture, hardware, software tools, and applications for VLSI computation with an emphasis on DSP workloads. Recent projects include the Asynchronous Array of simple Processors (AsAP) chip, applications, and tools; low density parity check (LDPC) decoders; FFT processors; viterbi decoders; and H.264 video codecs.

From 1987 to 1989, he was with Hewlett-Packard, Cupertino, CA, where he participated in the development of the processor for a high-end minicomputer. In 1999, he joined Atheros Communications, Santa Clara, CA, as an early employee and served as a core member of the team which developed the first IEEE 802.11a (54 Mb/s, 5 GHz) Wi-Fi wireless LAN solution. During the summer of 2006 he was a Visiting Professor in Intel's Circuit Research Lab.

Dr. Baas was a National Science Foundation Fellow from 1990 to 1993 and a NASA Graduate Student Researcher Fellow from 1993 to 1996. He was a recipient of the National Science Foundation CAREER Award in 2006 and the Most Promising Engineer/Scientist Award by AISES in 2006. He is an Associate Editor for the IEEE Journal of Solid-State Circuits and has served as a member of the Technical Program Committee of the IEEE International Conference on Computer Design (ICCD) in 2004, 2005, and 2007. He also serves as a member of the Technical Advisory Board of an early stage technology company.