Synchroscalar: Initial Lessons in Power-Aware Design of a Tile-Based Embedded Architecture

John Oliver[1], Ravishankar Rao[1], Paul Sultana[1], Jedidiah Crandall[1], Erik Czernikowski[1], Leslie W. Jones IV[2], Dean Copsey[1], Diana Keen[2], Venkatesh Akella[1], and Frederic T. Chong[1]

[1] University of California at Davis
[2] California Polytechnic State University, San Luis Obispo

Abstract. Embedded devices have hard performance targets and severe power and area constraints that depart significantly from our design intuitions derived from general-purpose microprocessor design. This paper describes our initial experiences in designing Synchroscalar, a tile-based embedded architecture targeted for multi-rate signal processing applications.

We present a preliminary design of the Synchroscalar architecture and some design space exploration in the context of important signal processing kernels. In particular, we find that synchronous design and substantial global interconnect are desirable in the low-frequency, low-power domain. This global interconnect enables parallelization and reduces processor idle time, which are critical to energy efficient implementations of high bandwidth signal processing. Furthermore, statically-scheduled communication and SIMD computation keep control overheads low and energy efficiency high.

Keywords: Low Power Processor, 802.11(a), Programmable DSP Processor, tiled-based architectures, embedded processors.

1 Introduction

Next-generation embedded applications demand high throughput with low power consumption. Current approaches often use Application-Specific Integrated Circuits (ASICs) to satisfy these constraints. However, rapidly evolving application protocols, multi-protocol embedded devices, and increasing chip NRE costs all argue for a more flexible solution. In other words, we want the flexibility of a programmable DSP with energy efficiency more similar to an ASIC. We propose the Synchroscalar architecture, a tile-based DSP designed to efficiently meet the throughput targets of applications with multi-rate computational subcomponents.

In designing Synchroscalar, we focused on three key features of ASICs that lead to their energy efficiency – high parallelism, custom interconnect, and low control overhead. Parallelism is important in that it allows the frequency of an

architecture to be reduced linearly with investment in logic, modulo communication. This linear reduction, when coupled with voltage scaling, yields a quadratic decrease in power and a linear decrease in system energy. Low communication latency, however, is important in maintaining the parallelism necessary for these energy gains. ASICs accomplish low latency through custom interconnect. We find that, in the low frequency domain, a tile-based processing architecture can use segmentable global busses to achieve low latency with high energy efficiency. Control overhead of the busses is kept low by using statically scheduled segmentation and data motion. Control overhead of the tiles can be reduced by grouping columns into SIMD execution units.

In the remainder of this paper, we provide an overview of the Synchroscalar architecture to establish the context of our study. Then we provide some simple tile and interconnect models which we used to guide our design. We use these models to conduct an analysis of FIR, FFT, Viterbi, and AES kernels running on different points in the design space. We discuss our intuitions from this analysis and conclude with future work for our project.

2 Synchroscalar Architecture

In this section, we introduce the proposed Synchroscalar architecture and the rationale behind it. As noted in the previous section, we were motivated by the need for an embedded architecture with the flexibility of a general purpose processor (DSP) and the power efficiency of an application specific integrated circuit. We examined ASIC implementations of Viterbi, FFT, AES, FIR and found that the key sources of the power efficiency of an ASIC are

- Parallelism, multiple clock and voltage domains
- Customized interconnect mirroring the dataflow inherent in the computation
- Distributed memory to provide high bandwidth
- Customized functional blocks to implement the computation
- Absence of instructions, removing instruction cache accesses and decode logic

If we want to approach the efficiency of an ASIC, our architecture should retain as many of the key strengths of an ASIC as possible. This directs us towards a tiled-based multiprocessor architecture with multiple clock and voltage domains, reconfigurable interconnect, and low-overhead SIMD control.

Abstractly, Synchroscalar is a two dimensional array of processing elements (PEs), each column potentially operating at different fixed frequencies and hence voltage. There is a single vertical bus connecting the elements in a column, and these vertical buses are connected by a single horizontal bus for communication between columns. In reality, in order to reduce the distance between PEs in a single column, the column is folded over. There are PEs on both sides of the vertical bus. That is the basis for Synchroscalar, as shown in Figure1 (we do not plan to support dynamic frequency/voltage scaling at present). Because of the data-parallel nature of computation, each PE can be viewed as one functional unit of a SIMD machine. There is a SIMD controller for each pair of columns.

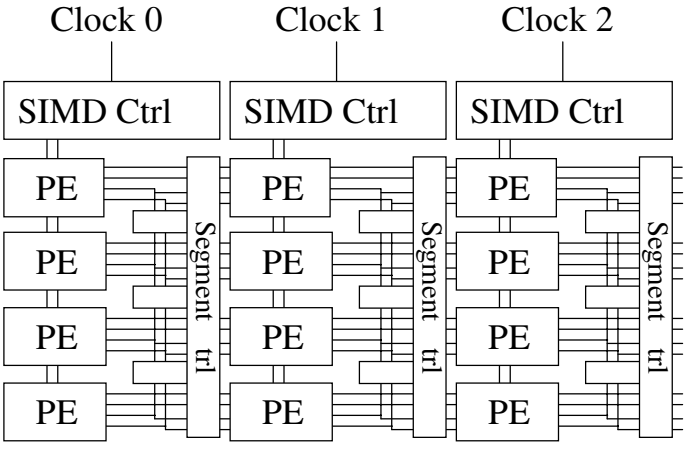

Fig. 1. The Synchroscalar Architecture

Each PE (tile) has a single DSP engine with two functional units, SRAM, register file, and communication interfaces. For brevity, we will refer to this cluster of bus, two columns, and SIMD controller as a single column. Although the tiles are SIMD, the communication patterns are not identical, so programmable engines are required for controlling communication.

2.1 Programming Model

The architecture of Synchroscalar is motivated by Synchronous Dataflow (SDF) model of computation [2, 3, 4]. DSP design environment tools created by Synopsys and Cadence use this model.

SDF is a subset of general purpose dataflow that restricts the number of data values produced and consumed by an actor to be a constant. The restriction imposed by the SDF model offers the advantage of static scheduling and decidability of key verification problems such as bounded memory requirements and deadlock avoidance [8] Synchroscalar can be viewed as a architecture to support SDF computation model efficiently. This predictability is crucial to providing the generality of programming units while retaining much of the efficiency of ASICs.

2.2 Clock and Voltage Domains

Clock and voltage domains are per-column, with the task parallelized within the column. Tasks can be mapped to different columns depending on their computational requirements. This mapping is crucial to performance, because once set, the voltage and frequency of a column may not change. Mapping algorithms must be developed to provide minimize communication and maximize power savings. Computationally-intensive tasks are performed at the best available

frequency and voltage that meets the performance requirements. Other tasks can be mapped to columns operated at lower frequency and voltage.

We employ *rational clocking*[15] for the frequencies of different columns. If f_m and f_n are the frequencies of two columns of PEs then $f_m/f_n = M/N$ where M and N are integers. While this allows a wide range of selection of frequencies, the relation between the two frequencies provides the predictable communication points between the domains required for statically scheduled communication. Rational clocking eliminates the synchronization overhead with asynchronous or GALS systems while still giving us the flexibility of different frequency domains.

ASICs benefit from high-bandwidth, low-latency communication provided by custom interconnects. We exploit low clock frequencies and static scheduling to maximize throughput while minimizing latency. Static scheduling is required to maintain guaranteed performance. Although the clock frequencies are low enough to traverse a column in a single cycle, we segment the bus in order to increase the usable bandwidth. Segment controllers are turned on or off by signals from a central per-column segment controller. As shown in Fig.1, the bus connecting two columns of PEs is partitioned into segments [23] by segment controllers.

The column segment controllers are small state machines which can be reconfigured for each algorithm. By suitably controlling the segment controllers, the bus can perform several parallel communications. For instance, if all the controllers are turned off, the bus becomes a broadcast bus, all PEs able to receive the same data. Alternatively, two messages can pass between neighboring columns using the same wires in different segments if the segment controller between them is on. The tasks are mapped to the tile architecture such that the communication between the PEs is minimized. Highly communicating tasks are assigned to neighboring PEs. This reduces the number of segments the data has to travel, and hence saves power.

2.3 SIMD Control

In order to reduce the cost of instruction fetch and decode, a single SIMD controller sends instructions to the PEs in a column. The SIMD controller performs all control instructions, only forwarding computation instructions to the PEs. To communicate data (used for conditional branches), the SIMD controller is connected to the segmented bus with the PEs.

In order to use branch prediction, there needs to be a mechanism to squash instructions that have already been sent to the processing elements. Instead, we provide a short pipeline in the control unit to calculate branches quickly, and delay instructions from reaching the processing elements. This introduces a single-cycle stall for each conditional branch. For zero-overhead loops, there is still no delay, because the PC is used for decision-making, not the actual instruction. Our implementation incurs no extra overhead for these loops which are critical to DSP performance.

3 Framework

With the Synchroscalar architecture and motivation for context, we now present a general framework within which to evaluate the surrounding design space. The framework will use some simple first-order models of tile and interconnect power, validated with datapoints in the literature and VHDL designs of custom Synchroscalar elements. Although our models are by necessity abstract enough to cover the design space, we argue that the important scaling effects are captured and that our qualitative conclusions are valid.

3.1 Tile Model

We use the voltage frequency scaling given by the Newton's alpha law f= k$*\frac{(V_{dd}-V_t)^\alpha}{Vdd}$ [14]. This equation gives the voltage-frequency scaling for a given technology. We have modeled a ring oscillator in SPICE using the Berkeley Predictive Technology Model (http://www-device.eecs.berkeley.edu/ ptm/) to get a better feel for the acceptable range of supply voltage and threshold voltage. This enables us to project the models into 90 nm and 45 nm technology.

Our tile is based on the low power 16-bit VLIW DSPs similar to the Intel-ADI MSA-based Blackfin[7] and the SPXK5 from NEC [19]. The minimum core power is assumed to be 0.07mW/MHz similar to [19]. (We are in the process of finishing a detailed VHDL model for the tile and and validating this assumption). The SRAM power is given 0.02mA/MHz for 32kB of memory. This number was obtained from the circuit given in [12], by scaling for technology and size.

3.2 Interconnect Model

The interconnect model is largely based on the data given in [6]. We find that the gate and drain capacitances are orders of magnitude smaller than the wire capacitance per unit length. We thus model only the wire capacitance. The drain-source capacitance of the segmenters and the gate and drain capacitances of the drivers are ignored. In 0.18u tech, the gate capacitance of a minimum sized transistor is about 1-2fF [6]. This value is expected to remain constant over shrinking process technologies. The projected value, in 0.13u tech, of wire capacitance of a semi-global wire, per unit length is 387fF/mm. The chip length is about 10mm and hence the wire capacitance is about 3870fF. This suggests that even if the drivers and repeaters are 10-times the minimum size, their capacitance is about 20fF. If there are 8 drivers for each bus, it adds only 160fF to the wire capacitance.

We are in the process of completing VHDL models for the segment controllers, SIMD controller and the communication interfaces. We plan to augment our results with this in the future, but we believe that they are unlikely to change the major trends in the results reported here.

4 Applications

The main objective of this paper is an exploration of the design space defined by the goals of the Synchroscalar architecture. Specifically, we are interested in the impact of various architectural parameters such as the number of tiles, the interconnect structure, the width of the buses on the power while meeting the performance constraints of an application.

For an initial driving application, we choose the 54 Mbps 802.11(a) wireless LAN physical layer. This is currently outside the scope of DSP processors and is currently done with ASICs or DSPs with co-processors for the computationally intensive applications. The computationally challenging aspects of 802.11(a) are Viterbi decoder, FFT, and large FIR filters for equalization. We will evaluate each of these function on the Synchroscalar architecture. We derive the performance (throughput) targets for each function so that we can meet the 54 Mbps data rate. In addition we also use the Advanced Encryption Standard (AES) as a benchmark as it contains very different kind of computation, intensive on bit manipulation and table look-ups, to see how our architecture fares on such workloads.

The FIR filter is used in the equalization function in the OFDM receiver. We model a 128-tap FIR filter and assume that the data rate is 64 Mbps. We also model a 128 point FFT and assume the data rate is 256 Mbps. FFT and IFFT are key components of the OFDM receiver. For the Viterbi Decoderwe assume the constraint length for the decoder K=7 and the data rate is 54 Mbps. This is the most computation intensive part of the OFDM receiver.

Our initial experimental procedure is as follows:

1. Write the function in C and verify using Blackfin Visual DSP simulation environment
2. Replace the performance critical sections of the code with Blackfin assembly code, to achieve optimal performance. This corresponds to the implementation on a single tile.
3. Next map the application into multiple tiles and using a homebrewed tool to assist in pruning the search space.
4. Manually insert the communication instructions
5. Estimate the clock cycle count for the application.
6. Using the power model for the interconnect and the tile described in the previous section, estimate the power. The parameterized power models were described in Excel and that was used to generate the graphs reported in the next section.

While an extensive cycle-level simulation infrastructure is currently under development, we felt that hand-counts were appropriate for guiding the early design of the architecture. In particular, our driving signal processing applications are very amenable to hand-analysis as their computations are focused on a small number of kernels.

5 Results

Our results focus on several key design questions. We explore the parallelism available in each algorithm by varying the number of processing tiles, the communication bandwidth necessary through varying global bus widths, and the power efficiency of communication by exploring segmented buses.

5.1 Architectural Configurations

Figure 2 shows that as the number of tiles increases, there is the traditional tradeoff between computation and communication, but performance is not our goal. As the performance increases, we lower the clock frequency to maintain

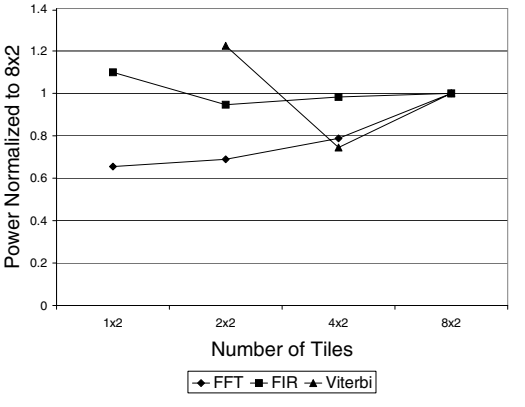

Fig. 2. Power required as the number of tiles increases

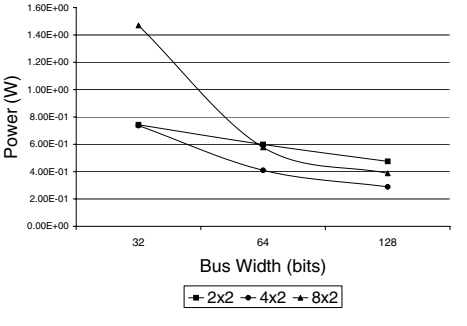

Fig. 3. Viterbi Decoder power as bus width increases for various tile configurations

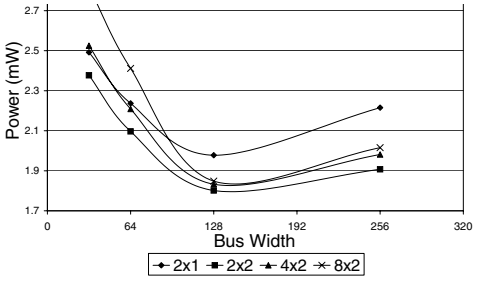

Fig. 4. FIR Filter power as the bus width increases for various tile configurations

a constant performance target, allowing a decrease in voltage. Note that this is not done dynamically. Each experiment with a different number of tiles is a completely different instance of the program. So, for each instance, we provide the lowest frequency / voltage to maintain the same performance. The tradeoff is then between adding processors, providing a constant increase in-power consumption, and reducing the voltage, providing a quadratic decrease in performance.

All three applications observe an initial decrease in total power, but by the 8x2 tile configuration, the decreasing returns of parallelization is outweighing the benefits voltage scaling. Thus we should provide either 2x2 or 4x2 tiles in each column.

5.2 Impact of Bus Width

We then vary bus width. Data dependencies prevent effective overlap of communication and computation. This makes fast communication critical to efficiency, else processor idle time will lead to wasted power. We note that processor power accounts for the majority of our system power and that it is impractical to turn processors on and off for periods on the order of a dozen cycles. Consequently, we see in Figures 3 - 5 that increasing bus width decreases processor idle time, which decreases system power. For FIR, the power begins increasing again at 256 bits because FIR can not take advantage of the increased width.

We further note that Amdahl's law comes into play, and we see the greatest power savings as we initially double bus width, cutting communication latencies in half. As we continue to invest in bus bandwidth, processor idle time becomes a smaller fraction of total run time. With cost as a concern, an area-conscious design philosophy would be to choose a bus width of 64 or 128 bits, where we get the most bang for the buck.

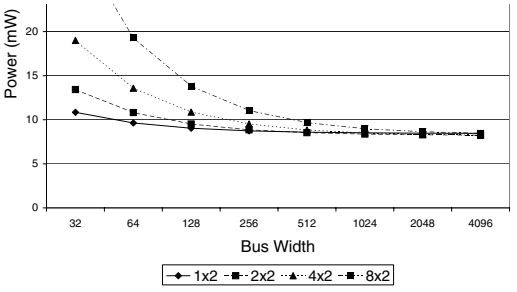

Fig. 5. FFT power as the bus width increases for various tile configurations

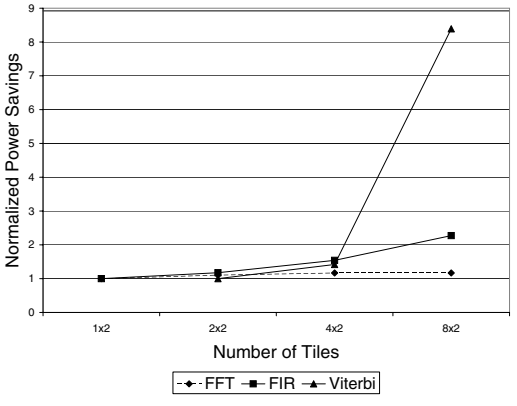

Fig. 6. Power Savings when using Segmented Buses over Unsegmented Buses

5.3 Impact of Segmented Buses

Segmenting the bus allows two simultaneous, short-distance messages to use the same bits in the wire. At the low frequencies of the Synchroscalar system, segmenters are simple transmission gates with little signal restoration or latency involved. Figure 6 shows that as the number of tiles in the column increases, the savings from segmentation also increases, because there are more messages that can traverse the bus at once. Dramatic savings are seen in Viterbi with 8x2 tiles. Even at 4x2, the applications observe 17-54% power savings.

5.4 Discussion

Our simple design-space exploration has revealed several results that challenge our intuitions of microprocessor design. Primarily, substantial global intercon-

nect makes sense in this domain. Low operating frequencies allow signals to traverse global buses in a single cycle. Data dependencies and tile power make the latency of global communication critical. Furthermore, statically-scheduled segmented buses allow the power and utilization of our interconnect to approximate more specialized interconnects as used in ASICs.

6 Related Work

The challenges presented by next generation applications in terms of higher data rates, lower power requirements, shrinking time-to-market requirements, and lower cost has resulted in a tremendous interest in architectures and platforms for embedded communication appliances in the past few years. Researchers have approached the problem from several different angles. The DSP architecture companies have proposed highly parallel VLIW machines coupled with hardware accelerators or co-processors for the computation-intensive functions. The TI OMAP is a good example of this category of solutions. The programmable logic community has been very active in this area, as well, and there are numerous architectural proposals that are derivatives of the standard FPGA. The SCORE project at UC Berkeley [5] and the PipeRench project at CMU [16] are especially noteworthy. They use the dynamic reconfigurability of field-programmable gate arrays to exploit power and performance efficiency. The PLEIADES project at UC Berkeley [21] proposes an interconnection of a low power FPGA, datapath units, memory, and processors, optimized for different application domains. The Pleiades researchers conclude that a hierarchical generalized mesh interconnect structure [22] is most appropriate for their architecture because it balances both the global and the local interconnect. Our results are in agreement with this conclusion in general but given that we are targetting streaming computations such as those encountered in a wirless transceiver, we have greater emphasis on near-neighbor communication, so we have stayed away from a general mesh.

The adaptive SOC project at University of Massachussets [10] advocates an array of processors connected by a statically scheduled communication fabric. They allow different processors to operate at different clock frequencies and demonstrate significant power savings on video processing benchmarks. The key differences between this work and Synchronscalar are in the structure and contents of the tiles and the memory architecture. In aSOC the tiles are hardwired functional blocks such as Viterbi decoder, FFT, DCT etc., while in Synchroscalar we assume programmable DSPs as the building blocks for the tiles. As a result, the memory architecture of the system is radically different, changing the data transfer and communication scheduling problem as well. But, it would be interesting to compare the results between the Synchroscalar and aSOC approaches.

Recently, there has been a revival of interest in locally synchronous and globally asynchronous (GALS) approach to processor implementation [1] including the use of multiple clock domains and multiple voltages [11] [17]. The key difference between GALS approach and the Synchroscalar approach is the restriction of using only rationally related frequencies between different columns. This

avoids the use of asynchronous FIFOs with their synchronization overhead. So, synchroscalar is similar to Numesh [18], rather than the GALS approach.

Synchroscalar's use of spatial rather than temporal flexibility is somewhat inspired by the MIT RAW project [20] [9], but our focus on low power and embedded applications is significantly different. Nevertheless, we expect to be further inspired by the extensive compiler work from the RAW group. Although their compiler algorithms are geared towards dynamic general microprocessor algorithms such as speculation and caching, we expect to leverage their experiences with program analysis and resource allocation.

Another project with a less embedded focus is the Imagine stream processor, a tile architecture at Stanford [13]. Their experience with streaming applications will also be invaluable to the design of our high-level software. Our emphasis on Synchroscalar regions for power reduction and static scheduling of rationally-clocked communication, however, will add significant challenges to our software solutions. Furthermore, both Imagine and RAW are focused on large-system scalability rather than the inexpensive design points of small, embedded systems. We believe that Synchroscalar's differing focus in cost and power will lead to significantly new tradeoffs and design decisions.

7 Conclusion

The goal of this work was to guide the initial design of tile-based embedded architecture. Through simple power models, we found that our original intuitions regarding interconnect did not apply to the low-frequency, data-dependent nature of our application domain. We found that wide, segmented global buses give us some of the low latency and flexibility that conventional DSPs lack. We plan to continue our evaluation of the Synchroscalar architecture through extensive design and simulation of end-to-end applications. We are confident that a novel architecture can meet the challenges of tomorrow's embedded applications.

Acknowledgements

This work is supported by NSF ITR grants 0312837 and 0113418, and NSF CAREER and UC Davis Chancellor's fellowship awards to Fred Chong.

References

1. B. M. Baas. A parallel programmable energy-efficient architecture for computationally intensive DSP systems. In *Conference Record of the Thirty-Seventh Asilomar Conference on Signals, Systems, and Computers*, Nov 2003.
2. S. Bhattacharya, P. Murthy, and E. Lee. Software synthesis from dataflow graphs, 1996.

3. S. Bhattacharya, P. Murthy, and E. Lee. Synthesis of embedded software from synchronous dataflow specifications. *Journal of VLSI Signal Processing*, (21):151–166, June 1999.
4. J. Buck, S. Ha, E. A. Lee, and D. G. Messerschmitt. Ptolemy: A framework for simulating and prototyping heterogenous systems. *Int. Journal in Computer Simulation*, 4(2):0–, 1994.
5. E. Caspi, M. Chu, R. Huang, J. Yeh, J. Wawrzynek, and A. DeHon. Stream computations organized for reconfigurable execution (SCORE). In *Field-Programmable Logic and Applications, FPL-2000*, pages 605–614, 2000.
6. R. Ho, K. Mai, and M. Horowitz. The future of wires. In *Proceedings of the IEEE*, volume 89, pages 490–504, April 2001.
7. R. Kolagotla, J. Fridman, B. Aldrich, M. Hoffman, W. Anderson, M. Allen, D. Witt, R. Dunton, and L. Booth. High Performance Dual-MAC DSP Architecture. *IEEE Signal Processing Magazine*, July 2002.
8. E. A. Lee and D. G. Messerschmitt. Static scheduling of synchronous dataflow programs for digital signal processing. *IEEE Transactions on Computers*, C-36(1), January 1999.
9. W. Lee, R. Barua, M. Frank, D. Srikrishna, J. Babb, V. Sarkar, and S. P. Amarasinghe. Space-time scheduling of instruction-level parallelism on a raw machine. In *Architectural Support for Programming Languages and Operating Systems*, pages 46–57, 1998.
10. J. Liang, S. Swaminathan, and R. Tessier. aSOC: A scalable, single-chip communications architecture. In *IEEE PACT*, pages 37–46, 2000.
11. D. Marculescu and A. Iyer. Power and performance evaluation of globally asynchronous locally synchronous processors. In D. DeGroot, editor, *Proceedings of the 29th International Symposium on Computer Architecture (ISCA-02)*, volume 30, 2 of *Computer Architectuer News*, pages 158–170, New York, May 25–29 2002. ACM Press.
12. T. Mori, B. Amrutur, M. Horowitz, I. Fukushi, T. Izawa, and S. Mitarai. A 1v 0.19mw at 100 mhz 2kx16b sram utilizing a half-swing pulsed-decoder and write-bus architecture in 0.25 μm dual-vt cmos. In *Solid-State Conference, 1998, Digest of Technical Papers, 45th ISSCC 1998 IEEE International*, 1998.
13. S. Rixner, W. J. Dally, U. J. Kapasi, B. Khailany, A. Lopez-Lagunas, P. R. Mattson, and J. D. Owens. A bandwidth-efficient architecture for media processing. In *International Symposium on Microarchitecture*, pages 3–13, 1998.
14. T. Sakurai and R. Newton. Alpha-Power Law MOSFET Model and Its Application to CMOS Inverter Delay and Other Formulas. *IEEE Journal of Solid State Circuits*, 25:584–594, April 1990.
15. L. Sarmenta, G. A. Pratt, and S. Ward. Rational clocking. In *International Conference on Computer Design*, pages 271–278, 1995.
16. H. Schmit, S. Cadambi, M. Moe, and S. Goldstein. Pipeline reconfigurable FPGA. *Journal of VLSI Signal Processing Systems for Signal, Image and Video Technology*, 24(2):129–146, March 2000.
17. 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 *HPCA*, pages 29–42, 2002.
18. D. Shoemaker, F. Honore, C. Metcalf, and S. Ward. Numesh: An architecture optimized for scheduled communication. *Journal of Supercomputing*, 10(3), 1996.
19. M. Y. T. Kumura, M. Ikekawa and I. Kuroda. VLIW DSP for Mobile Applications. *IEEE Signal Processing Magazine*, July 2002.

20. M. B. Taylor, J. Kim, J. Miller, D. Wentzlaff, F. Ghodrat, B. Greenwald, H. Hoffman, P. Johnson, J.-W. Lee, W. Lee, A. Ma, A. Saraf, M. Seneski, N. Shnidman, V. Strumpen, M. Frank, S. Amarasinghe, and A. Agarwal. The Raw microprocessor: A computational fabric for software circuits and general-purpose programs. *IEEE Micro*, 22(2):25–35, Mar./Apr. 2002.
21. H. Zhang, V. Prabhu, V. George, M. Benes, A. Abnous, and J. Rabaey. A 1-V heterogenous reconfigurable DSP IC for wireless baseband digital signal processing. *IEEE Journal of Solid State Circuits*, 35:1697–1704, November 2000.
22. H. Zhang, M. Wan, V. George, and J. Rabaey. Interconnect Architecture Exploration for Low Energy Reconfigurable Single-Chip DSP. In *Proceedings of the Workshop on VLSI, Orlando, Florida*, April 1999.
23. Y. Zhang, W. Ye, and M. J. Irwin. An alternative for on-chip global interconnect: Segmented bus power modeling. In *Conference Record of the Thirty-Second Asilomar Conrference on Signals, Systems and Computers*, pages 1062–1065, 1998.