A Design Environment for High-Throughput Low-Power Dedicated Signal Processing Systems

W. Rhett Davis, *Member, IEEE*, Ning Zhang, *Student Member, IEEE*, Kevin Camera, *Student Member, IEEE*, Dejan Marković, *Student Member, IEEE*, Tina Smilkstein, *Student Member, IEEE*, M. Josie Ammer, Engling Yeo, *Student Member, IEEE*, Stephanie Augsburger, *Student Member, IEEE*, Borivoje Nikolić, *Member, IEEE*, and Robert W. Brodersen, *Fellow, IEEE*

***Abstract*—A hierarchical automated design flow for low-energy direct-mapped signal processing integrated circuits is presented. A modular framework based on a combined dataflow graph and floorplan description drives automatic layout generation with commercial CAD tools. Automatic characterization of layout improves system-level estimates. Simplified physical design methodologies for low supply voltages are discussed. The flow is demonstrated on a 300-k transistor test-chip, a time-division multiple-access baseband receiver, and a soft-output Viterbi decoder. An example of architectural comparison of energy efficiency is presented.**

***Index Terms*—Application specific integrated circuits, design automation, design methodology, integrated circuit design, parallel architectures, system analysis and design.**

I. Introduction

THE architectures commonly used to implement signal-processing algorithms in hardware differ most significantly in terms of efficiency and flexibility. General purpose processors are the least energy- and area-efficient, while slightly more specialized architectures, such as programmable digital signal processors, can often accomplish the same task with an order of magnitude less energy. The most efficient architectures in terms of power and area can be obtained by directly mapping the algorithms into hardware. Computational energy and area efficiencies that can be achieved with this approach are 100–1000 MOPS/mW and 100–1000 MOPS/mm^2. These efficiencies can be two to three orders of magnitude higher than the efficiency achieved by software processors [1].

A direct-mapped architecture can be obtained by mapping the operations of a dataflow graph directly into functional units and hard-wiring the connections between them. In this way, the maximum parallelism can be obtained, allowing the minimum clock rate and supply voltage to be used, resulting in reduced energy per operation [2]. The ability to exploit a high level of parallelism allows computational rates that far exceed uniprocessors without requiring high clock rates. For example, a direct-mapped implementation of the three-tap finite-impulse response (FIR) filter graph shown in Fig. 1(a) would contain a delay line, three multipliers, and two adders as shown in Fig. 1(b). In contrast, a resource-shared architecture such as the one shown in Fig. 1(c) alters the dataflow graph in order to reduce the design to a single multiplier and adder. The energy required for the computation can be modeled with the equation

$$E = CV^2 \tag{1}$$

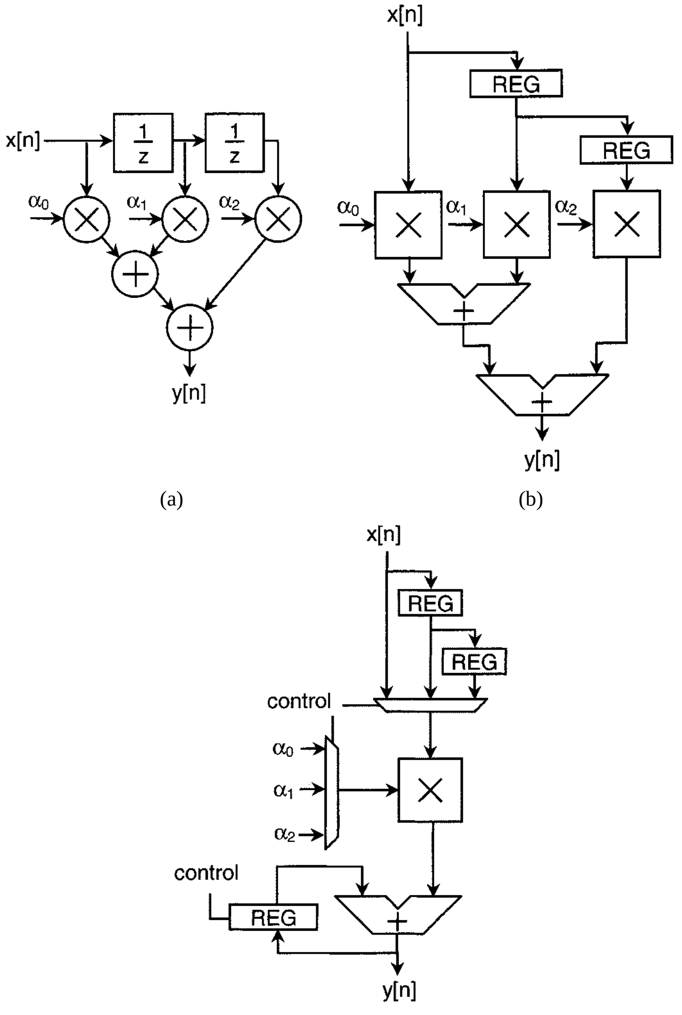


Fig. 1. A simple data-flow graph for: (a) a three-tap FIR filter, (b) a direct-mapped implementation, and (c) a resource-shared implementation.

Manuscript received July 24, 2001; revised October 22, 2001. This work was supported by DARPA and the member companies of the Berkeley Wireless Research Center.

W. R. Davis is with the Berkeley Wireless Research Center, Berkeley, CA 94704 USA (e-mail: wrdavis@eecs.berkeley.edu).

N. Zhang is with Atheros Communications, Inc., Sunnyvale, CA 94085 USA.

K. Camera is with Atheros Communications, Inc., Sunnyvale, CA 94085 USA. He is also with the Department of Electrical Engineering and Computer Science, University of California, Berkeley, CA 94704 USA.

D. Markocić, T. Smilkstein, M. J. Ammer, E. Yeo, S. Augsburger, B. Nikolić, and R. W. Brodersen are with the Department of Electrical Engineering and Computer Science, University of California, Berkeley, CA 94704 USA.

Publisher Item Identifier S 0018-9200(02)01695-5.

TABLE I
ARCHITECTURAL COMPARISON OF ENERGY EFFICIENCY FOR COMMON METHODS OF ALGORITHM IMPLEMENTATION, SCALED TO 0.18-μm TECHNOLOGY

Architecture	64-point FFT Energy per Transform (pJ)	16-State Viterbi Decoder Energy per Decoded bit (pJ)
Direct-Mapped Hardware	1.78	0.022
FPGA	683	5.5
Low-Power DSP	436	19.6
High-Performance DSP	1700	108

where C is the intrinsic amount of capacitance that must be switched and V is the supply voltage. The serial, resource-shared architecture has reduced area but does not reduce the capacitance to be switched for the FIR computation. Furthermore, the multiplier and adder must be clocked three times as fast, requiring a higher supply voltage and ultimately more energy than the parallel, direct-mapped architecture for the same throughput.

The efficiency of direct-mapped architectures makes them especially attractive for many digital signal processing (DSP) applications. DSP algorithms can be extremely complex with very high processing rates but are highly parallel. Consider the performance of direct-mapped architectures compared to FPGA and programmable DSP implementations of the fast Fourier transform (FFT) and Viterbi decoder algorithms, two important parts of a wireless orthogonal frequency-division multiplexing (OFDM) system [3]. Table I shows the comparison between vendor-published benchmark data for the industry-leading high-performance and low-power programmable DSPs[1] and FPGA[2] and post-layout simulations of direct-mapped hardware [4]. The results were calculated for constant throughput rates of 50 Ms/s for the FFT and 100 Mb/s for the Viterbi decoder and have been scaled to a common technology (0.18 μm) to support a meaningful architectural comparison. The table shows roughly a three-orders-of-magnitude energy penalty for the high-performance programmable approach and more than two orders of magnitude for the low-power approaches.

In spite of the enormous advantage of direct-mapped architectures, they are not commonly used unless the application cannot be accomplished by any other means. Direct-mapped IP cores for some communication system components (such as Viterbi decoders) are readily available, but most of today's wireless devices are based on programmable solutions. It is doubtful that programmable solutions will ever "catch up" to the energy efficiency of direct mapping, because sequential execution destroys the parallelism inherent in algorithms and wastes the opportunity to save power by lowering the supply voltage. Furthermore, because of the poor mapping between the algorithm and the architecture, it is becoming difficult for programmable architectures to even meet the performance requirements of today's wireless systems. For example, even

[1]Numbers taken for the Texas Instruments TMS320C6x and the TMS320C5x architectures using assembly code, posted benchmarks for the C5000 and C6000 platforms [5], Viterbi decoder application notes [6] and measured power consumption data [7].

[2]Numbers taken for the Xilinx Vertex-E using the Xilinx LogiCORE product specifications [8] and online Vertex Power Estimation Worksheet [9].

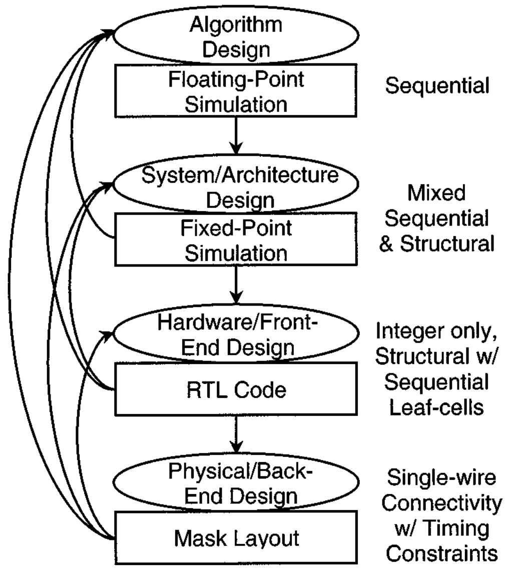

Algorithm Design
Floating-Point Simulation
Sequential
System/Architecture Design
Fixed-Point Simulation
Mixed Sequential & Structural
Hardware/Front-End Design
RTL Code
Integer only, Structural w/ Sequential Leaf-cells
Physical/Back-End Design
Mask Layout
Single-wire Connectivity w/ Timing Constraints

Fig. 2. Standard design flow for hardware implementation of algorithms.

the highest performance programmable DSP today would need to spend roughly 50% of its cycles on a 64-point FFT to meet the throughput requirements of the IEEE 802.11a wireless networking standard, leaving very little room to implement the remainder of the algorithm. Programmable architectures, like direct-mapped, must seek to exploit parallelism to close the performance gap.

Direct-mapped architectures are seen as unattractive primarily because the tremendous design effort involved is not economically viable given the lack of flexibility of the final hardware. This paper presents our solution for achieving the benefits of direct mapping with drastically reduced design effort in order to make hardware flexibility less of an issue. The paper begins with an examination of existing methodologies and the factors that frustrate the design of direct-mapped architectures. Next, our design methodology will be presented, followed by a discussion of physical issues which enable designs in the low-power domain. Lastly, we present several design examples that use this flow.

II. CURRENT METHODS FOR ALGORITHM IMPLEMENTATION

A standard design flow for hardware implementation of algorithms has four phases which are typically handled by four different designers, as shown in Fig. 2. Algorithm designers conceive the chip and deliver a specification to system designers, often in the form of a floating-point simulation. This simulation can be used to generate system characterizations such as a bit-error-rate (BER) versus SNR curve for a communications chip. The system or architecture designers begin to add structure to this simulation, partitioning the design into functional units. They must also convert the data types from floating to fixed-point and verify that finite word-length effects and pipeline depth do not compromise the algorithm. The hardware (or front-end) designers map the simulation to register-transfer level (RTL) code (usually VHDL or Verilog) and verify that the code matches the specified functionality and pipeline depth. Physical designers take standard-cell netlists synthesized from the RTL code and use place-and-route tools to generate layout mask patterns for fabrication while verifying that all timing constraints are met, commonly referred to as reaching timing closure. This flow requires three translations of the design, expressing the functionality as gradually less sequential and more structural with requirements for reverification at each stage. Opportunities for algorithmic modifications to reduce power and area are often lost due to the separation of engineering decisions. Performance bottlenecks discovered during the physical design phase are unknown to the algorithm designer. Aggressive system requirements may require new and unusual architectures, which can stall the flow, leading to uncontrolled looping back to earlier stages of the design process and extending the design time indefinitely.

The main problem with this flow is that it attempts to avoid feeding back information to algorithm designers. The technique of reducing power through algorithmic transformations to permit voltage reduction [2] is well understood. However, today's CAD environments do not support this kind of design. The flow we need would allow algorithm designers to explore the design space as thoroughly as possible by creating mask layout and obtaining performance estimates. This exploration should allow refinement of fixed-point types, be constrained by libraries of efficient hardware blocks, and be carried out by an automated design flow. This encourages feedback of physical design issues to algorithm designers by allowing them to maintain ownership of the design data at all times. It also would encourage interaction with system, hardware, and physical designers by reducing the design process to a single phase.

Recent efforts have identified the gaps between algorithm, system, hardware, and physical design but have yet to encompass the complete problem. Some attempt to close the gap between algorithm and hardware design by basing synthesis tools on C/C++ descriptions [10]. However, these solutions require a style of C code that is very similar to RTL code and is unattractive to algorithm designers. Commercial tools from design automation companies offer RTL code generation solutions from block diagrams. However, these tools are targeted mostly for hardware designers and obscure the information about the algorithm and architecture through the code generation process. Much attention has been given also to the problem of closing the gap between hardware and physical design by giving the hardware designer control over physical structure [11] and floorplanning early in the design process [12]. However, these concepts need to be pushed into the algorithm design phase as well. To accomplish this, a design environment is needed which offers fast, automatic generation of physical information from a system-level description.

Work on automating the design process began with silicon compilers in the early 1980s [13], leading to research projects such as FIRST [14], LAGER [15], and many others and commercial products such as Genesil [16]. The limitations of process technology limited their usefulness, however, since it was easy to create large designs but expensive to fabricate them. Furthermore, these systems required extensive physical libraries which were difficult to create and maintain. However, the efficiency of the present-day standard-cell methodology allows the designer to work above the physical level during library creation, and current process technologies allow fabrication of much larger designs. Another limitation of the silicon compilers of the past is that they were driven by input languages which were structural and not easy to simulate, making it difficult to drive the flow from the algorithm designer's perspective. The VOV system [17] took an entirely different approach to design by automating flows comprised of generic tools. VOV was focused mainly on repeating flow traces generated by a single user on a single design. We need a system that automates the flow for all users on any design.

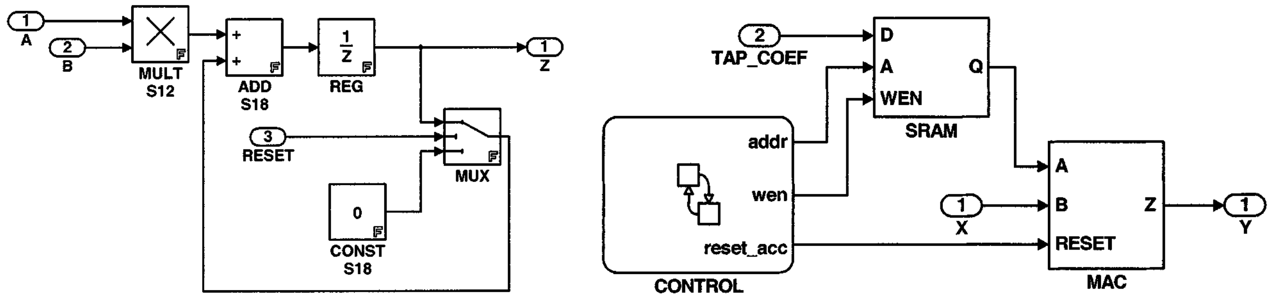
III. Chip-in-a-Day Design Flow

A. Capturing Design Decisions

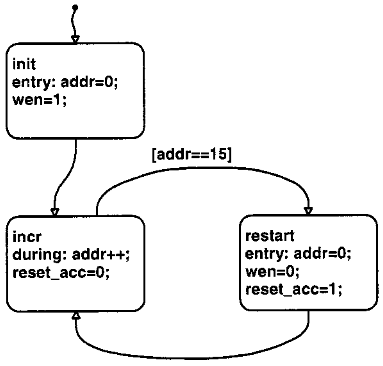
In order to provide a well-integrated design flow, we must be very specific about how we make and capture design decisions. Our goal with this flow was to get algorithm designers to drive the entire design process from the same description they used to develop the algorithms. This goal therefore requires them to use dataflow graphs instead of writing C or Matlab code. By starting with a dataflow graph, we avoid the difficulty of inferring parallelism from a procedural description and can derive a parallel architecture directly from the graph. All of the decisions necessary to create mask layout and get performance estimates would be entered refinements of the original dataflow graph. These design decisions are divided into the following types:

- *Function decisions*: input–output behavior of the dataflow graph, specified by the simulator associated with dataflow graph editor.
- *Signal decisions*: physical signals, such as word lengths, specified as edge properties.
- *Circuit decisions*: transistors to implement each block, specified as node properties.
- *Floorplan decisions*: physical locations of functional units, specified with a companion floorplan view.

The choice of a dataflow graph editor depends mainly on the model of computation used to express function decisions [18]. We chose a discrete-time computation model because it can be made cycle-accurate and bit-true with respect to the hardware, which is necessary to verify that hardware generated by the flow faithfully represents the functional description. There are a number of dataflow graph editors that support the discrete-time model, and we chose MathWorks' Simulink [19] because of its familiarity to algorithm designers, due to its close integration with Matlab. It is important to make it as easy as possible for al-



(a) (b)



(c)

Fig. 3. (a) Dataflow graphs of a time-multiplexed FIR filter with (b) a detail of the multiply-accumulate block and (c) detail of the control logic finite state-machine.

gorithm developers to approach the design environment in order to ease the use of hardware dependent optimization early in the design process. Fig. 3 illustrates a cycle-accurate dataflow graph example of a time-multiplexed FIR filter. Fig. 3(a) shows a multiply-accumulate block being fed by an input data stream, tap coefficients from an SRAM, and some control logic.

In signal processing architectures, datapath logic typically amounts for more than 90% of the area and power. The dataflow graphs are therefore built on primitives which are common in digital datapaths. Floating-point primitives are swapped with fixed-point primitives to explore finite word-length effects. These fixed-point types express the signal decisions for the automated flow and are needed in order for the simulation to be bit-true. Fig. 3(a) shows a multiply-accumulate block with a fixed-point multiplier, adder, register, and multiplexer. The goal is to specify functionality and signals with this dataflow graph so completely that RTL simulation is not a necessary part of the design process.

Since it is inefficient to specify control with dataflow graphs, we chose a different strategy for these functions by adding an extended finite state-machine primitive. These state-machines are co-simulated with the dataflow graphs but require a different editor. Fig. 3 shows our approach to this specification with an example of a time-multiplexed FIR filter. Fig. 3(b) shows a multiply-accumulate block being fed by an input data stream, tap coefficients from an SRAM, and control signals from a state-machine. Fig. 3(c) shows the address generator and MAC reset control state-machine. This chart has an initial loop to load tap coefficients, with successive loops reading the coefficients and resetting the accumulator.

The leaf-nodes (primitives) of the dataflow graph hierarchy will correspond to functional units in the hardware and are thus called "macros." Circuit decisions are encapsulated through the specification of a parameterized generator for each macro. Fixed-point types and other parameters can be passed to the macro generators, thus allowing exploration of design tradeoffs from the dataflow graph representation. Each generator produces a netlist of cells for which layout and schematic views exist. There are four main types of macros currently supported by our flow:

- *datapath macros*: ranging in complexity from arithmetic blocks (such as adders, registers, and multpliers) to complex pipelines such as an FFT or Viterbi decoder, implemented with VHDL code, datapath generators or semi-custom tiled layout modules;
- *state-machine macros*: VHDL code generated from each state-machine which is synthesized;
- *black-box macros*: cells which have schematic and layout-views already defined, such as a vendor-supplied SRAM;
- *composite macros*: portions of a design hierarchy that have been routed.

Though several mechanisms can be used for macro implementation, the most popular method has been the use of a commercial datapath generator [20]. The description language for this generator has a highly restrictive set of operators that is very similar to the set of primitives used by the dataflow graph simulator, which minimizes the effort involved in creating new library elements. This approach does result in a second functional specification for a block that must be made equivalent to its corresponding dataflow graph model. At present, this is accomplished with cross-check simulations between the dataflow graph and RTL code created by the datapath generator. Equivalence could also be checked with formal methods. Equivalence checking is considered to be part of the library creation process

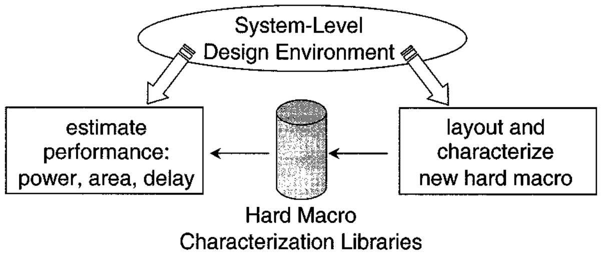

System-Level
Design Environment
estimate performance: power, area, delay
layout and characterize new hard macro
Hard Macro Characterization Libraries

Fig. 4. Hardening the hierarchy.

TABLE II
RESULTS OF HARDENING BASEBAND RECEIVER SYSTEM MACROS IN A 0.25-μm TECHNOLOGY

macro statistics		decimation filter	adaptive pilot detect	frequency offset estimator
area in 0.25 µm		1.4 mm^2	1.2 mm^2	0.22 mm^2
power @ 25 MHz	2.5 V	120 mW	88 mW	5.2 mW
	1.0 V	13 mW	7.6 mW	0.47 mW
critical-path delay	2.5 V	5.6 ns	10 ns	5.2 ns
	1.0 V	18 ns	32 ns	17 ns
	cells	21 k	15 k	3500
	transistors	240 k	220 k	37 k
exec. time	synth / route	3 hours	4 hours	1 hour
	characterize	9 hours	16 hours	1.5 hours
disk space	synth / route	180 MB	160 MB	67 MB
	characterize	1.5 GB	980 MB	190 MB

and facilitates reuse of complex functional blocks. However, this task is difficult, and a much better approach would be to ensure equivalence through automatic translation of datapath generator code from the dataflow graph.

B. Hardening the Hierarchy

One of the most important parts of our environment is the ability to quickly route and characterize these macros, turning them into "hard macros" as illustrated in Fig. 4. The performance of these "hard macros" is well understood, meaning that estimates for the entire system performance will have less variance as more macros are hardened. The higher levels of the hierarchy can then be adjusted to compensate for any incorrect assumptions. In using this flow, we find that the design process tends to progress by routing and characterizing the entire hierarchy from the macros to the top-level. We call this process "hierarchy hardening," and once the entire hierarchy is hardened, the design is done. This contrasts sharply with the standard flow in Fig. 2 where the phases of the design process are determined by which designer's expertise is currently being used rather than which level of the functional hierarchy is currently being hardened. Table II shows a sample of the types of data obtained from the process of hardening three datapath generator macros from the basband receiver design example discussed later. These estimates are obtained after the flow has synthesized, routed, performed parasitic extraction, and run switch-level static-timing and transient simulations on each macro using test vectors generated from the dataflow graph. The essential price of push-button automation is execution time and disk space, shown in the table for a 400-MHz UltraSPARC-II system with 2 MB of L2 cache, 4 GB of RAM, 8 GB of swap, and a NetApp F630 filer available over Gigabit Ethernet. This information is stored in order to provide fast estimates of performance and required resources and ultimately should be available from within the dataflow graph editor to provide information for reuse.

C. Signal Optimizations

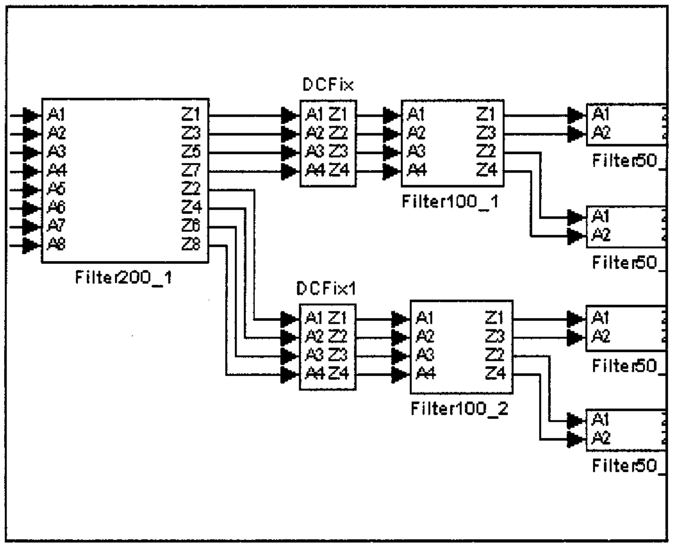
The flow supports a number of special signal-level optimizations in the dataflow graph which can reduce the power and area of the design. Certain primitives are recognized to be reordering of wires rather than circuit macros. For example, the use of the "Enable" block, which can be used to turn off blocks when not in use, corresponds to a gated clock in the physical design. Another approach would be to translate multiple sample times for unit-delay primitives into synchronized clock-trees with multiple rates. Simpler optimizations which are supported include multiplication by a constant power of 2 (a hard-wired shift) and comparison to zero (the sign bit). In addition, the flow permits certain signals to be denoted as "simulation only," meaning that they will not appear in the final circuit. This allows the algorithm designer to create debugging signals in their dataflow graphs freely without worrying about how they affect the hardware and eliminates the need to translate the design when creating mask layout. The critical requirement to facilitate our goal is that the same description be used for both algorithmic and hardware exploration.

D. Physical Design

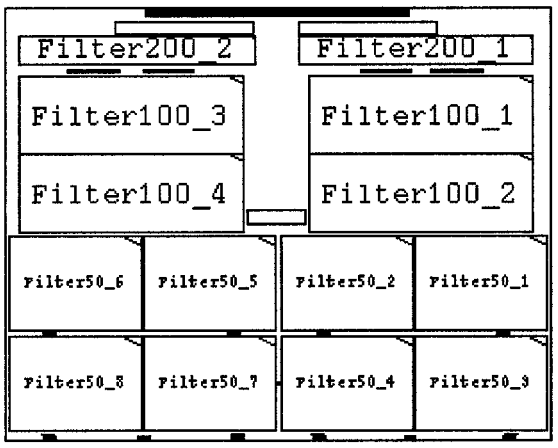
In order to provide scalability to large designs and to allow the use of heterogeneous macros, it was found that a floorplan was required in addition to the functional description. Floorplan decisions are captured with commercial physical design tools. The initial skeleton floorplan is generated by the automated flow, and then the algorithm designers edit the floorplan, placing instances and boundary pins using simplified commands. Instance names in the floorplan are constrained to match the block names in the dataflow graph to minimize confusion from the translation of design data. Furthermore, the floorplan hierarchy matches the hierarchy of the dataflow graph. Blocks which are repeated in the dataflow graph become repeated in the floorplan. This minimizes the floorplanning effort and the execution time of the later routing steps. Fig. 5 shows a portion of the top-level dataflow graph and floorplan for a parallel, pipelined FIR decimation filter which reuses large filter blocks. Once the floorplan is complete, the design can be routed and characterized by the automated flow. This floorplan could also be used to improve fast performance estimates from the dataflow graph editor by predicting the parasitics of global wires with Manhattan distances.

E. Design Flow Automation

An abstract view of the automated design flow is shown in Fig. 6. The "elaboration" step translates the dataflow graph into an electronic design framework. This step also invokes macro generators and stitches their output into a single netlist of



(a)



(b)

Fig. 5. (a) Dataflow graph and (b) floorplan for a parallel pipelined FIR filter.

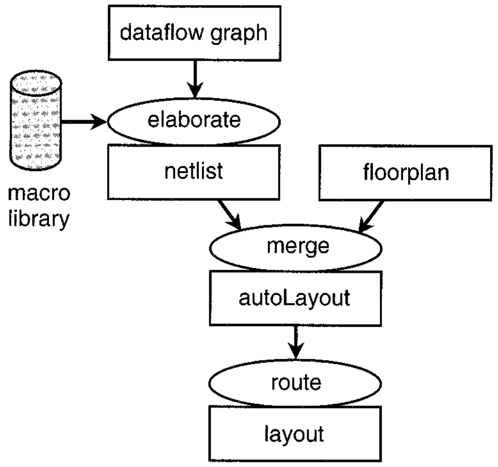


Fig. 6. High-level dependency graph for the automated design flow.

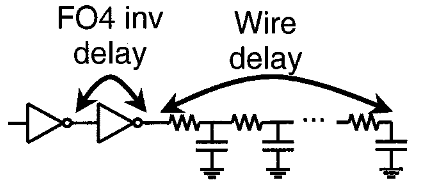
routable objects. The next step merges placement information from the floorplan views with the netlist, creating "autoLayout" views. Designers modify these autoLayout views and save them as floorplans for the next invocation of the flow. This allows the dataflow graph and floorplan to be developed side by side. As new blocks are added to the dataflow graph, new unplaced blocks appear in the floorplan after the merge step. After merging, the flow proceeds to a series of steps which route the hierarchy from the macros to the top level. As illustrated in the figure, the design flow is described as a dependency graph and uses a method of automation similar to the UNIX MAKE program.

Several new programs were developed to support the design flow. First, we wrote a new program to translate the hierarchical dataflow graphs into EDIF files. Translation is accomplished by tagging different portions of dataflow graph as "macros" and annotating fixed-point types for the edges between the macros. Each macro then defines an instance and each edge defines a group of nets. Handling hierarchy was the main challenge in writing this translator, because it both expands the hierarchy and preserves block references. The hierarchy must be expanded to ensure that primitives with different parameters become different cells in the physical hierarchy. For example, 8-bit and 12-bit adders are instances of the same primitive in the dataflow graph but must be instances of different master cells in the physical design. On the other hand, the translator must preserve block references to allow repetition of complex blocks. For example, if an FIR filter block is to be routed once and repeated several times in the layout, each instance must have the same master cell. Our EDIF translator preserves the block references and expands the hierarchy only if the macros have different parameters.

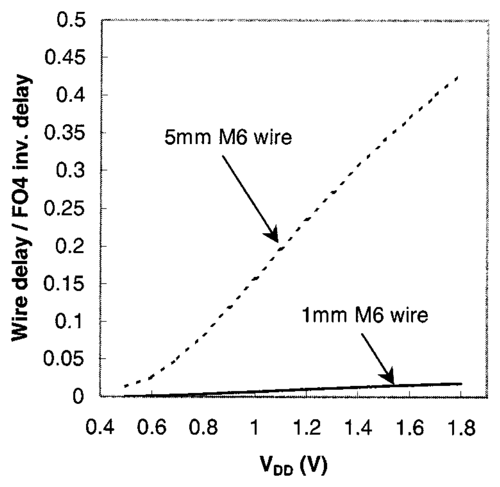
Next, we developed a program to translate designs from Stateflow [19], the finite state-machine editor bundled with Simulink, into synthesizable VHDL code. Stateflow allows the description of transition behavior with action statements annotated as labels. The action syntax is almost identical to C and also follows the same sequential execution model. The primary goal of our translator is to systematically generate hardware with the same cycle-accurate behavior as the software-based model. This helps us to meet our goal of eliminating RTL simulations from the design flow. To make the hardware more efficient, our translator maps the data types from the action syntax into IEEE standard logic vectors with word lengths based on minimum and maximum range limits specified in the chart. Once the designer has chosen these limits and verified them in simulation, the resulting hardware is guaranteed to have adequate precision and range. Comparisons of synthesized area and speed for generated code and hand-authored code show that the tool produces efficient hardware when used on large, complex state-machines. Charts with fewer than 10 states tend to be inefficient and slow due to the overhead of maintaining cycle-accuracy with the dataflow graph simulator. Due to the small ratio of control logic to datapath logic in our designs of interest, this inefficiency is negligible.

IV. Enabling Physical Design Issues

One of the greatest difficulties of standard ASIC design flows is timing closure. Timing closure is the process of verifying that the layout meets the timing constraints assumed in the initial description. The difficulty arises from the fact that interconnect



(a)



(b)

Fig. 7. (a) FO4 inverter and wire delay measurement setup and (b) simulated results of voltage scaling for isolated 1-mm and 5-mm metal 6 wires in a 0.18-μm technology.

capacitance is unknown at design time. The most common problems that arise during hardening are failure to meet cycle-time goals and generation of clock-trees with excessive skew. The low-voltage and low-power nature of direct-mapped architectures simplifies these issues and allows us to deal with them in ways that are easy to automate.

A. *Reduced Impact of Interconnect*

The impact of interconnect on the design process is reduced at low supply voltages, since the logic speed has been decreased. This effect can be illustrated by measuring the ratio of logic delay to wire delay as illustrated in Fig. 7(a). Wire delay is measured from input to output nodes of a distributed *RC* wire while logic delay is measured from input to output nodes of an inverter. The inverter is sized so that the lumped capacitance of the wire is a fan-out of 4 (FO4) load. The graph in Fig. 7(b) shows that, at the standard supply voltage for the technology, the wire delay is close to half of the logic delay for a 5-mm isolated metal 6 wire. As the supply voltage drops, however, the wire delay does not change while the logic delay rises, causing the wire/logic delay ratio to drop considerably. This means that, at low voltages, long wires can be accurately modeled as lumped capacitances, making it easier to predict delay from simple Manhattan distances measured in the floorplan. It also means that we can design much larger blocks without worrying about repeater insertion, making it considerably easier to design large systems.

B. *Race-Immune Clock-Tree Generation*

Clock-tree design in the standard ASIC flow typically consists of running automated clock-tree insertion tools on a flat netlist. If the resulting clock-tree violates hold-time constraints, then it can be back-annotated into the synthesis tool and resynthesized to add delay to paths with excessive skew. Once hierarchy is added to the physical design, it becomes harder for clock-tree insertion tools to control the skew, and the back-annotation/resynthesis flow becomes problematic. Our design flow avoids this problem by exploiting the low supply voltages to pursue race-immune clock-tree synthesis. We define the quantity "race margin" for a given technology to be the minimum clock-to-Q delay of all clocked elements minus the maximum hold time

$$t_{\text{skew(max)}} < t_{\text{clk}-Q(\text{min})} - t_{\text{hold(max)}}. \tag{2}$$

If the absolute skew of the clock tree is less than the race margin, then no back-annotation and resynthesis flow is required to prevent races. Transmission-gate flip-flops (TGFFs), also called master–slave latch pairs, are the preferable clocked element for this methodology because they are the most energy efficient and have large internal race margins [21].

The gating of the clock is determined in the dataflow graph by disabling blocks of the algorithm. Because the physical hierarchy matches the dataflow graph hierarchy, it is sufficient to build balanced clock trees for the subblocks of the design and gate clocks at the top level of routing hierarchy. Our methodology is to use a commercial tool to build a clock tree for the largest subblock with a bounded clock slope and skew less than the race margin. Then, trees for the other subblocks are generated to match the first tree. The clock-tree insertion and characterization process is automated by the flow, however, the user must choose which block's clock tree is to be matched by the other blocks' trees. Table III shows the results of this methodology for a design with three subblocks of different sizes in a 0.18-μm technology. The race margin was simulated to be 580 ps, which is safely larger than the 340 ps of skew between the blocks. Table III also shows the clock-tree power relative to the logic power at 1 V, indicating that only limited additional power is required to achieve race immunity.

C. *Simplified Power Routing*

Another barrier to full layout automation is power routing, since supply and ground rails are typically hand routed. The floorplan view for our flow allows specification of hand-routed critical net segments in addition to placement information. However, since the power dissipation of direct-mapped architectures is very low (100 mW can provide 10-100 GOPS/s), the standard-cell row rails have sufficient capacity to support a much larger area than with high-speed ASICs. Also, the standard-cell junction capacitances provide sufficient decoupling capacitance to suppress rail-bouncing. In order to take advantage of this simplification, supply rings are not automatically drawn around the lower levels of hierarchy. Instead, the flow inserts filler cells to abut the lower levels of hierarchy and connect power through the standard-cell rows.

V. Design Examples

A. *FIR Filter Test Chip*

To verify the approach of this flow, a test chip was developed based on the parallel, pipelined FIR decimation filter shown in

TABLE III
EXAMPLE HIERARCHICAL CLOCK-TREE STATISTICS IN A 0.18-μm TECHNOLOGY

Block	Routed Area (mm^2)	Clock Sinks	Tree Stages	Insertion Delay, 1V typ		Power @ 25 MHz, 1 V	
				min (ns)	max (ns)	Clock (mW)	Logic (mW)
1	0.59	4246	6	2.266	2.371	0.77	1.49
2	0.39	607	10	2.214	2.270	0.14	0.65
3	0.29	456	12	2.381	2.455	0.12	0.28

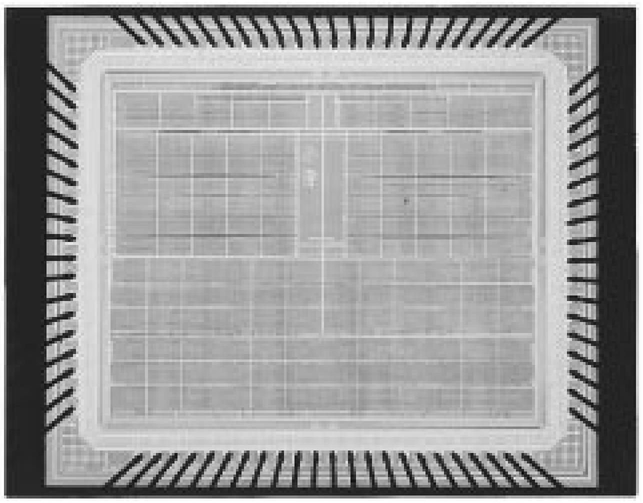


Fig. 8. Die photo of the decimation filter test chip.

TABLE IV
COMPARISON OF TEST-CHIP PERFORMANCE ESTIMATES IN A 0.25-μm TECHNOLOGY

estimate	dataflow-graph-based	layout	actual
area	2.0 mm^2	6.8 mm^2	6.8 mm^2
power	16 mW	17 mW	14 mW
criti.-path delay	19 ns	21 ns	19 ns

Fig. 5. This chip investigated another approach for race immunity through the use of a two-phase clock. This was found to be inefficient and unnecessary, however, and the strategy previously described has been used in all other designs. A die photo is shown in Fig. 8.

The test chip was designed with several hundred hard macros with roughly the complexity of adders and registers. These macros were placed manually in three levels of physical hierarchy. This was more levels than necessary, but it allowed us to exercise our hierarchical place-and-route flow more thoroughly. The design contained 307 K transistors and was fabricated in a 0.25-μm technology. Table IV shows the performance evaluation of the chip at 1.0 V and 25 MHz. The dataflow-graph-based estimates are very close to the layout estimates due to the fact that power and delay numbers from characterized hard macros were used. The area discrepancy between dataflow-graph-based estimates and layout arises from the fact that the hard macros could not be abutted in this version of the flow, leading to a wasted space. From this design, it was determined that the multiplier and adder macro granularity is too high. The number of these objects quickly becomes unmanageable in submicrometer technologies (a multiplier requires only 0.05 mm^2 in a 0.25-μm). In contrast, a version of this decimation filter created with the standard-cell datapath generator (shown in Table II) implements the same function with the same critical path delay but is much smaller because it was placed as a flat netlist. Also, the power consumption of the flat design is lower due to reduced wire length and transistor count. The performance of the test chip relative to the datapath generator macro indicates that such detailed floorplanning is not necessary for this particular structure. More recent versions of the flow allow selective flattening of the physical hierarchy to improve routing density.

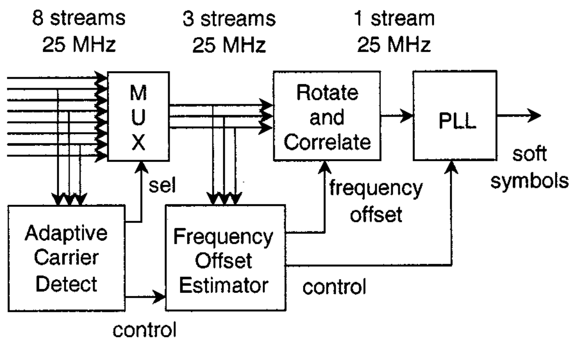


Fig. 9. Block diagram for the 1.6-Mb/s TDMA-DSSS digital baseband timing recovery chip.

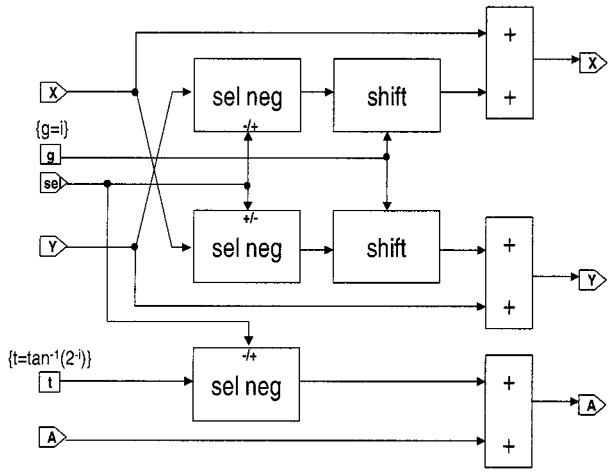


Fig. 10. Reusable CORDIC slice.

B. Baseband Receiver

A 1.6-Mb/s TDMA-DSSS digital baseband timing recovery unit for use in a low-power wireless network [22] has been designed using this flow. A block diagram for the system is shown in Fig. 9. This synchronization system is intended to provide coherent timing recovery and code acquisition for a stream of

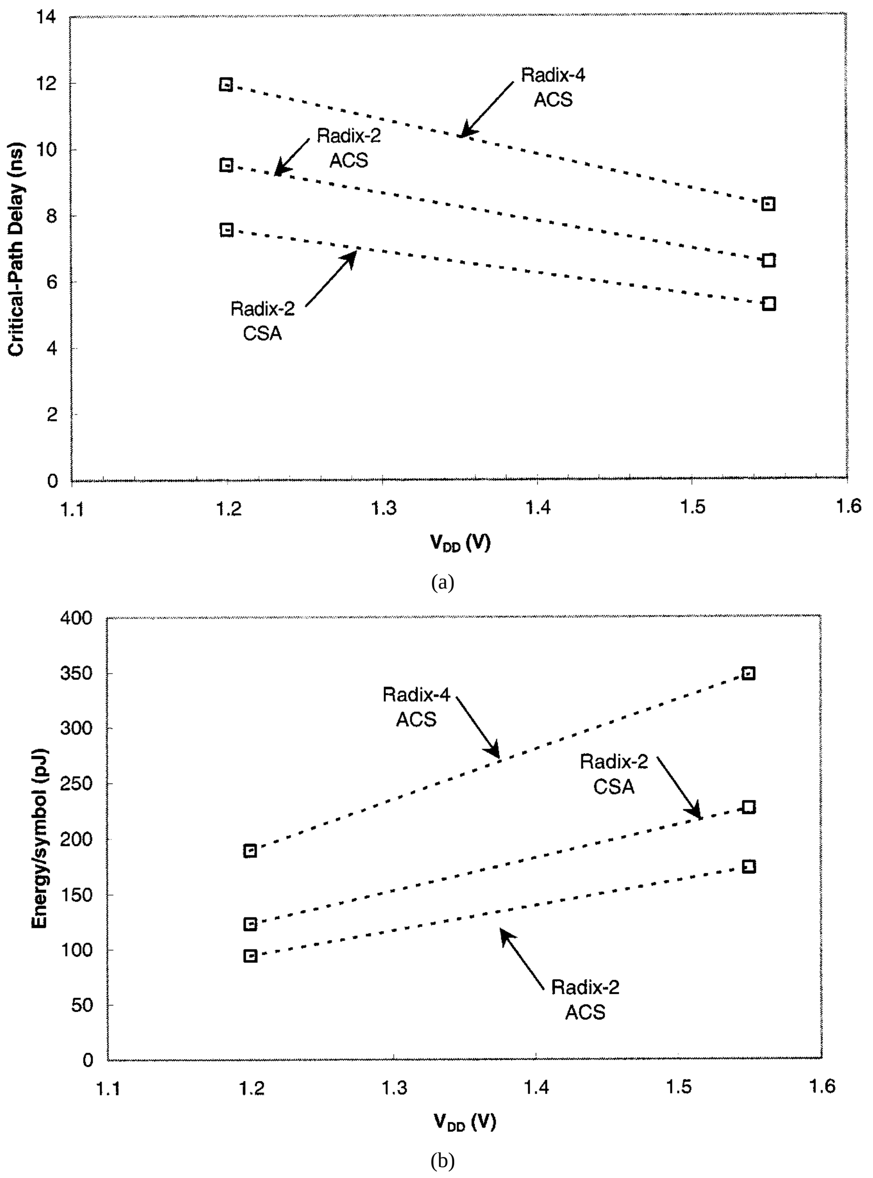


Fig. 11. Comparisons of: (a) critical-path delay and (b) energy/symbol for post-layout characterizations of three SOVA architectures.

soft symbols. The white blocks in the figure correspond to the hardened macros with results given in Table II.

This design uses a single-phase clock with five gated clock domains to turn off large sections of the chip when not in use. The algorithm proceeds in stages with each stage's estimate of timing being used by the subsequent stage. Each stage of the algorithm needs to operate on only a fraction of the total number of symbols in a packet. The carrier detection stage, for example, needs to be active for only two symbols out of every 500. Rather than developing an architecture that supports all stages of the algorithm, it was much easier from a design standpoint to simply map each algorithmic stage directly into a hardware block and then shut off the clock to each block when not in use. While this approach could be considered wasteful of area, it is optimal from an energy/symbol standpoint and provides the algorithm designer with the ability to make a cost-energy tradeoff. This approach also frees the system designer from the difficulty of translating the algorithm architecture to a different circuit architecture when optimizing the algorithm for energy.

The design of this system was accelerated by a parameterized CORDIC-slice macro shown in Fig. 10. The slice macro was parameterized in terms of the bit-widths of inputs X, Y, and A, the constant shift value G, and the constant arctangent value T. The slice was then used 27 times with different parameters to implement CORDIC angle rotation and polar-to-rectangular conversion blocks. This indicates the level of reuse which is possible, since once the CORDIC function was debugged and verified, no further debugging at the RTL level was necessary.

The first routing pass of this design was performed flat and ran through the automated flow in 13 h. A level of physical hierarchy was later added to the design, yielding five blocks in the top level. This simplification of the physical design data reduced the overall flow execution time to 2.5 h. Furthermore, the total area was reduced by 3% due to the increased density allowed by the simplified blocks.

This design also demonstrates our state-machine translator with three state machines with 30 total states which requires 1.2% of the total cell-area of the chip. The critical path of the

TABLE V
COMPARISON OF ENERGY EFFICIENCY OF SOVA ARCHITECTURES IN A 0.18-μm TECHNOLOGY

architecture	simulated			scaled		
	supply	throughput	energy/sym.	supply	throughput	energy/sym.
radix-2 ACS	1.2 V	105 Msym/s	94 pJ	1.2 V	105 Msym/s	94 pJ
radix-2 CSA	1.2 V	132 Msym/s	123 pJ	1.0 V	105 Msym/s	85 pJ
radix-4 ACS	1.2 V	168 Msym/s	189 pJ	0.94 V	105 Msym/s	116 pJ

generated control logic is 15 ns at 1.2 V which is considerably less than the overall critical path of 31 ns in the datapath blocks. This supports our assumption that the control logic is not the limitation of system performance.

C. SOVA Design Exploration

In this section, we present an example of the type of design exploration made possible by our flow. The soft-output Viterbi algorithm (SOVA) has been recently examined as a building block in high-throughput iterative decoders. Iterative decoders promise significant SNR performance improvement over conventional decoders at the expense of increased complexity. An implementation of this algorithm that uses a modified register-exchange method for calculating survivor paths [23] was fabricated in a 0.18-μm technology with low-threshold standard cells and a single-phase clock.

With our flow, resynthesizing the design with high-threshold cells and rehardening the hierarchy is essentially automated. This made it possible for us to examine the performance of various SOVA microarchitectures for both high-throughput and low-power applications, using the traditional radix-2 add–compare–select (ACS) architecture (with a 0.30 mm^2 area) as a baseline.

The traditional bottleneck in Viterbi decoders, the ACS recursion, can be transformed and retimed as compare–select–add (CSA) operations in order to improve the critical path delay. Each of the CSA units has one fewer addition in the critical path [24]. This modification increases the speed at the expense of doubled numbers of adders and multiplexers, as well as increased routing complexity. This change is quantified with a few changes to the datapath generator code and re-execution of the flow. As shown in Fig. 11, a speed increase of 26% is accompanied by 31% increase in energy, and 19% increase in area (0.35 mm^2. The increased routing complexity was evident from the fact that the router needed 30% more time to complete with the same cell density (95%).

Another common transformation for increasing throughput of the ACS unit is to perform two steps of the algorithm at once, resulting in a radix-4 ACS Viterbi decoder [25]. Critical-path delay increases but the overall throughput improves. This modification triples the area of the ACS unit but also increases the size of the register-exchange unit, making it difficult to predict the overall change without carrying out the design to completion. With approximately a week of modification to the dataflow graphs and datapath generator code and a day of re-executing the flow, we hardened and characterized a radix-4 version of SOVA design. Fig. 11 shows that the critical-path delay relative to the radix-2 ACS design was increased by 25%. Area increased by a factor of 2.4 to 0.72 mm^2, and power consumption quadrupled, causing the energy per symbol to double since two symbols are handled in each cycle.

TABLE VI
SUMMARY OF OTHER BLOCKS DEVELOPED WITH OUR FLOW

block	area in 0.25 μm	clock frequency
OFDM synchronization	0.40 mm^2	20 MHz
FFT	1.4 mm^2	25 MHz
Viterbi decoder	0.71 mm^2	25 MHz
3G MAP Turbo decoder	19 mm^2	2 MHz
LDPC iterative decoder	0.21 mm^2	350 MHz
Polyphase filter bank	12 mm^2	36.1 MHz
RAKE receiver	0.45 mm^2	25 MHz
DFT	60,000 μm^2	300 KHz
Huffman decoder	16,000 μm^2	150 MHz
Run-length decoder	6,200 μm^2	467 MHz

Lastly, we would like to know which of these three architectures would be the most energy-efficient if we were given complete freedom when scaling supply voltages. The answer can be approximated by using the voltage-delay characteristic of a ring-oscillator (obtained with a transistor-level simulation) to scale the voltage of the CSA and radix-4 designs down until they match the throughput of the original ACS design. The results of this scaling are shown in Table V. The results show that the voltage could be dropped by 0.2 V for the CSA design but only an additional 0.06 V for the radix-4 design due to the rapidly increasing delay below 1 V. The energy per symbol was minimized by the CSA, leading us to conclude that for this range of throughputs, the radix-2 CSA is the most energy efficient architecture.

D. Other Examples

A considerable number of design examples have been developed to determine the relationship between algorithm and architecture. Building blocks have been developed which include equalizers, polyphase filters, correlators, MAP and LDPC decoders, Huffman, Lempel–Ziv decoders, DFT, and FFT blocks. These macros were used to build communications and signal processing systems, such as iterative decoders for high throughput or low power, data handling for maskless lithography, polyphase filter banks, CDMA/TDMA baseband receivers with RAKE processing, an OFDM receiver with multi-antenna support, and signal processing for image-reject mixers. Table VI summarizes the blocks developed with area and clock frequency as rough measures of complexity [26].

VI. Conclusion

The success of the test chips and ease of the macro hardening flow are encouraging. The next step is to apply the flow to the design of systems in the 1-M to 10-M transistor range. The most difficult aspect of this flow is the verification of functional equivalency of macro generators and their dataflow graph models. As macros become more complex, more opportunities for discrepancy arise, leading to potential problems when macros are combined. Future work will focus on comparisons of the estimates gained from this approach to estimates made with other system-level design methods. Also, much more investigation is needed into the level of detail needed during floorplanning and into which macro granularities scale best to future process generations.

Acknowledgment

The authors would like to thank ST Microelectronics for fabrication of the test chips, H. So, F. Chen, B. Coates, D. Wang, and N. Chan for their help developing the flow, and P. Husted and M. Sheets for using the flow and providing valuable feedback.

References

[1] R. Brodersen, “The network computer and its future,” in *ISSCC Dig. Tech. Papers*, Feb. 1997, pp. 32–36.

[2] A. P. Chandrakasan and R. W. Brodersen, “Minimizing power consumption in digital CMOS circuits,” *Proc. IEEE*, vol. 83, pp. 498–523, April 1995.

[3] N. Weste and D. J. Skellern, “VLSI for OFDM,” *IEEE Commun. Mag.*, vol. 36, pp. 127–131, Oct. 1998.

[4] N. Zhang and R. W. Brodersen, “Architectural evaluation of flexible digital signal processing for wireless receivers,” in *Proc. Asilomar Conf. Signals, Systems and Computers*, Oct. 2000, pp. 78–83.

[5] C5000 and C6000 Platform Benchmarks (2002). [Online]. Available: http://dspvillage.ti.com

[6] H. Hendrix. (1996, Jan.) Viterbi decoding techniques for the TMS320C54x DSP generation. Application Report, Texas Instruments, Inc. [Online]. Available: http://www-s.ti.com/sc/psheets/spra071a/spra071a.pdf.

[7] K. Castille. (1999, Nov.) TMS320C6000 power consumption summary. Application Report, Texas Instruments, Inc.. [Online]. Available: http://www-s.ti.com/sc/psheets/spra486b/spra486b.pdf.

[8] (2002) XtremeDSP LogiCORE product datasheets. Xilinx, Inc. [Online]. Available: http://www.xilinx.com

[9] Vertex Power Estimate Worksheet Version 1.5 (2002). [Online]. Available: http://www.xilinx.com/cgi-bin/powerweb.pl.

[10] (2002). Open SystemC Initiative. [Online]. Available: http://www.systemc.org.

[11] W. J. Dally and A. Chang, “The role of custom design in ASIC Chips,” in *Proc. Design Automation Conf.*, June 2000, pp. 643–647.

[12] J. S. Yim, S. O. Bae, and C. M. Kyung, “A floorplan-based planning methodology for power and clock distribution in ASIC’s,” in *Proc. Design Automation Conf.*, June 1999, pp. 766–771.

[13] C. A. Mead, “Structural and behavioral composition of VLSI,” in *Proc. Int. Conf. on VLSI*, Aug. 1983, pp. 3–8.

[14] N. Bergmann, “A case study of the FIRST silicon compiler,” in *Proc. Caltech Conf. on VLSI*, Mar. 1983, pp. 413–430.

[15] C. B. Shung *et al.*, “An integrated CAD system for algorithm-specific IC design,” *IEEE Trans. Computer-Aided Design*, vol. 10, pp. 447–463, Apr. 1991.

[16] S. C. Johnson and S. Mazor, “Silicon compiler lets system makers design their own VLSI chips,” *Electronic Design*, vol. 32, pp. 167–181, Oct. 1984.

[17] A. Casotto and A. Sangiovanni-Vincentelli, “Automated design management using traces,” *IEEE Trans. Computer-Aided Design*, vol. 12, pp. 1077–1095, Aug. 1993.

[18] E. A. Lee and A. Sangiovanni-Vincentelli, “A framework for comparing models of computation,” *IEEE Trans. Computer-Aided Design*, vol. 17, pp. 1217–1229, Dec. 1998.

[19] The MathWorks, Inc. Simulink and stateflow. [Online]. Available: http://www.mathworks.com.

[20] Synopsys, Inc. Module compiler. [Online]. Available: http://www.synopsys.com.

[21] D. Marković, B. Nikolić, and R. W. Brodersen, “Analysis and design of low-energy flip-flops,” in *Proc. Int. Symp. on Low Power Electronics and Design*, Aug. 2001, pp. 52–55.

[22] J. L. da Silva *et al.*, “Design methodology for PicoRadio networks,” in *Proc. Design, Automation and Test in Europe*, March 2001, pp. 314–323.

[23] E. Yeo, P. Pakzad, B. Nikolić, and V. Anantharam, “VLSI architectures for iterative decoders in magnetic recording channels,” *IEEE Trans. Magn.*, vol. 37, pp. 748–755, Mar. 2001.

[24] I. Lee and J. L. Sonntag, “A new architecture for the fast Viterbi algorithm,” in *Proc. IEEE Global Telecommunications Conf.*, Nov.–Dec. 2000, pp. 1664–1668.

[25] P. J. Black and T. Meng, “A 1-Gb/s, four-state, sliding block Viterbi decoder,” *IEEE J. Solid-State Circuits*, vol. 32, pp. 797–805, June 1997.

[26] University of California at Berkeley. (2000, Dec.) VLSI signal processing class web page. [Online]. Available: http://bwrc.eecs.berkeley.edu/Classes/EE225C/projects.htm.

W. Rhett Davis (S’92–M’02) received B.S. degrees in electrical and computer engineering from North Carolina State University, Raleigh, in 1994 and the M.S. degree in electrical engineering from the University of California at Berkeley in 1997. He recently received the Ph.D. degree from the University of California at Berkeley.

He worked two years with the Center for Advanced Electronic Materials Processing. After working briefly with Hewlett-Packard in Boeblingen, Germany, he came to the University of California at Berkeley, where his doctoral research concerns digital IC design, communications, and computer-aided design with the Berkeley Wireless Research Center.

Ning Zhang (S’97) received B.S. degrees in applied physics (with honors) from the California Institute of Technology, Pasadena, in 1996 and the M.S. and Ph.D. degrees in electrical engineering from the University of California at Berkeley in 1998 and 2001, respectively.

She worked at Wireless Research Laboratory, Lucent Technologies, in 1999. She is currently working at Atheros Communications, Inc. Her research interests include communications systems and architectures for wireless applications, digital signal processing (DSP) for communications, and low-power DSP architectures.

Kevin Camera (S’01) received the B.S. and M.S. degrees from the University of California at Berkeley in 1998 and 2001, respectively. He is currently working toward the Ph.D. degree in electrical engineering at the same university.

Recently, he has also been an employee at Atheros Communications, working on improvements to the algorithm verification environment used for the development of 802.11a wireless LAN chipsets. His present research goals focus on the investigation and creation of new hardware/software architectures for highly energy-efficient signal processing applications.

Dejan Marković (S'96) received the Dipl.Ing. degree in electrical engineering from the University of Belgrade, Yugoslavia, in 1998 and the M.S. degree in electrical engineering from the University of California at Berkeley in 2000. He is currently working toward the Ph.D. degree at the University of California at Berkeley focusing on energy-efficient digital integrated circuits and architectures for wireless communication receivers.

He was a Visiting Scholar at the University of California at Davis, in 1998, where he conducted research in the area of pass-transistor logic. He held internship positions at Lawrence Berkeley National Laboratory in 1999 where he worked on pixel-array IC for X-ray spectroscopy, and Intel Corporation in 2001 where he worked on low-energy clocked storage elements.

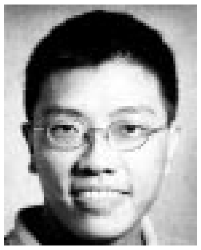
Mr. Marković received the 2001–2002 CalVIEW (Video Instruction for the Engineering World) Fellow Award for excellence in teaching and mentoring of industry engineers through the CalVIEW distance learning program.

Tina Smilkstein (S'01) received the B.A. degree in business administration with an emphasis on information management from Nanzan University, Nagoya, Japan, in 1989. After completing a three-year computer science re-entry program at the University of California at Berkeley in 1999, she entered the Electrical Engineering Graduate Program also at the University of California at Berkeley where she is presently researching automated low-power clock tree generation.

M. Josie Ammer received the B.S. and M.Eng. degrees in electrical engineering and computer science from the Massachusetts Institute of Technology (MIT), Cambridge, in 1997 and 1999, respectively. She is currently working toward the Ph.D. degree at the University of California at Berkeley studying with Prof. J. Rabaey.

Her research interests include low power digital integrated circuits for wireless communication.

Ms. Ammer received the Robert M. Fano UROP (Undergraduate Research) Award in 1997 and the Ernst A. Guillemin Masters Thesis Award, First Prize, in 1999, while at MIT. She is most recently funded by an Intel Ph.D. Fellowship.



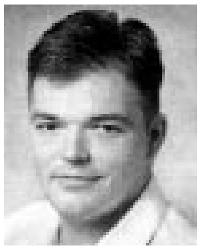
Engling Yeo (S'96) received the B.S. and M.S. degrees in electrical engineering and computer science from the University of California at Berkeley in 1994 and 1995, respectively. He is currently working toward the Ph.D. degree in electrical engineering at the same university.

He worked at the DSO National Laboratories, Singapore, from 1996 to 1999 as a Senior Member of Technical Staff in radar and signal processing systems. His primary interests are VLSI architectures for communication and storage systems.

Stephanie A. Augsburger (S'01) received the B.S. degree in computer engineering (with highest honor) from Georgia Institute of Technology, Atlanta, in 2000. She is currently working toward the M.S. degree in electrical engineering at the University of California at Berkeley.

As a Research Assistant, she has been focused on power optimization for digital integrated circuits.

Ms. Augsburger is funded by a Semiconductor Research Corporation Master's Scholarship.



Borivoje Nikolić (S'93–M'99) received the Dipl.Ing. and M.Sc. degrees in electrical engineering from the University of Belgrade, Yugoslavia, in 1992 and 1994, respectively, and the Ph.D. degree from the University of California at Davis in 1999.

He was on the faculty of the University of Belgrade from 1992 to 1996. He spent two years with Silicon Systems, Inc., Texas Instruments Storage Products Group, San Jose, CA, working on disk-drive signal processing electronics. In 1999, he joined the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley as an Assistant Professor. His research activities include high-speed and low-power digital integrated circuits and VLSI implementation of communications and signal-processing algorithms.

Dr. Nikolić received the College of Engineering Best Doctoral Dissertation Prize and Anil K. Jain Prize for the Best Doctoral Dissertation in Electrical and Computer Engineering at University of California at Davis in 1999, as well as the City of Belgrade Award for the Best Diploma Thesis in 1992.

Robert W. Brodersen (M'76–SM'81–F'82) received the Ph.D. degree from the Massachusetts Institute of Technology, Cambridge, in 1972.

He was then with the Central Research Laboratory at Texas Instruments for three years. Following that, he joined the Electrical Engineering and Computer Science faculty of the University of California at Berkeley, where he is now the John Whinnery Chair Professor. His research is focused in the areas of low-power design and wireless communications and the CAD tools necessary to support these activities.

Prof. Brodersen has won best paper awards for a number of journal and conference papers in the areas of integrated circuit design, CAD and communications, including in 1979 the W.G. Baker Award. In 1983, he was co-recipient of the IEEE Morris Liebmann Award. In 1986, he received the Technical Achievement Awards in the IEEE Circuits and Systems Society and in 1991 from the Signal Processing Society. In 1988, he was elected to be member of the National Academy of Engineering. In 1996, he received the IEEE Solid-State Circuits Society Award and in 1999 received an honorary doctorate from the University of Lund in Sweden. In 2000, he received a Millennium Award from the Circuits and Systems Society, the Golden Jubilee Award from the IEEE, and was co-recipient of the Lewis Winner Best Paper Award in the ISSCC.