Power and Area Minimization for Multidimensional Signal Processing

Dejan Marković, *Member, IEEE*, Borivoje Nikolić, *Senior Member, IEEE*, and Robert W. Brodersen, *Fellow, IEEE*

***Abstract*—Sensitivity-based methodology is applied to optimization of performance, power and area across several levels of design abstraction for a complex wireless baseband signal processing algorithm. The design framework is based on a unified, block-based graphical description of the algorithm to avoid design re-entry in various phases of chip development. The use of architectural techniques for minimization of power and area for complex signal processing algorithms is demonstrated using this framework. As a proof of concept, an ASIC realization of the MIMO baseband signal processing for a multi-antenna WLAN is described. The chip implements a 4×4 adaptive singular value decomposition (SVD) algorithm with combined power and area minimization achieving a power efficiency of 2.1 GOPS/mW (12-bit add equivalent) in just 3.5 mm^2 in a standard 90 nm CMOS process. The computational throughput of 70 GOPS is implemented with 0.5 M cells at a 100 MHz clock and 385 mV supply, dissipating 34 mW of power. With optimal channel conditions the algorithm implemented can deliver up to 250 Mb/s over 16 sub-carriers.**

***Index Terms*—Circuit optimization, CMOS digital integrated circuits, design methodology, field-programmable gate arrays, matrix decomposition, MIMO systems, multidimensional signal processing, parallel architectures, pipelines, sensitivity.**

I. Introduction

THE SCALING OF CMOS has been enabling integration of higher complexity systems in each technology generation. Feature scaling by a factor of 0.7 doubles the density of digital logic. This benefit can be exploited in two ways: either to decrease the chip size (thus its cost) or to increase the amount of functionality while maintaining the same chip size. In wireless and wireline communications applications the scaling of technology has traditionally been used for implementing more sophisticated signal processing in order to achieve higher data rates and better spectrum utilization.

Scaling of technology has imposed very tight power budgets for communications chips. Increased device leakages and decreased rate of supply voltage scaling, along with the demand for higher data rates and more functionality, have been steadily increasing the chip power consumption. Nowadays, power consumption of both fixed and mobile communication devices is limited either by the heat dissipation and packaging, or by the battery life. The requirements for high data rate and increased algorithmic complexity in next-generation wireless devices present a difficulty for meeting the power budgets. Therefore, in designing next-generation wireless systems, algorithm designers, system architects and circuit designers face a challenge of how to optimally utilize the benefits of technology scaling in a short development cycle.

The problem that needs to be solved is complex: which algorithm optimally uses the underlying technology to achieve the desired data rate, while staying within the power and die size limits? To answer this, every candidate algorithm has to be mapped into an architecture that is optimal for a particular technology. The architecture choice strongly depends on the required throughput, but also on the underlying technology options, usually defined by the choice of supply and threshold voltages. Technologies with high off-state leakage "prefer" deeper pipelined architectures, while the technologies with less leakage prefer memory-oriented and highly parallel architectures. To meet the deployment targets, design teams will often settle to a first architecture that meets the goals. Even when experimenting with implementing multiple algorithms there is no proof of a clear advantage of one to another; the final choice is ultimately dependent on a designer that achieves a smaller or lower power implementation with shorter design time.

At the lower level, problem of mapping architecture into silicon can be formulated in many ways. Commonly, there is a required performance level, with acceptable power and area budget. Since the performance of a communication system is often dictated by the standard, small power or area are commonly distinguishing features of a particular implementation. The overall design process, therefore, can be viewed as a constrained optimization problem, where the power and/or area are minimized under performance constraints.

This paper presents the methodology to formulate the system design problem as a performance-constrained and area-constrained power minimization. The problem will be solved using sensitivity-based optimization at multiple levels of design abstraction. Design at higher abstraction layers provides more room for power or area minimization then design at the circuit-level. However, architectural optimization techniques have been largely heuristic, and there is no established systematic way for trading off throughput, power and area. In our approach, optimal energy-delay tradeoffs from the circuit-level are used to characterize multiple architectural realizations and thus guide architectural mapping for an algorithm.

Section II introduces multi-antenna signal processing for decoupling of multi-path wireless channels, to demonstrate the level of complexity that can be supported with our design

Manuscript received August 25, 2006; revised December 19, 2006. This work was supported by C2S2 under MARCO Contract 2003-CT-888 and BWRC member companies. Chip fabrication was provided by STMicroelectronics.

D. Marković is with the Department of Electrical Engineering, University of California, Los Angeles, CA 90095 USA (e-mail: dejan@ee.ucla.edu).

B. Nikolić and R. W. Brodersen are with the Berkeley Wireless Research Center, University of California, Berkeley, CA 94704 USA.

Digital Object Identifier 10.1109/JSSC.2007.892191

methodology. The methodology for algorithm-to-architecture mapping is described in Section III, including sensitivity based optimization framework, Simulink design description, and ASIC design flow. The proposed approach enables power and area optimizations across the boundary of algorithm, architecture, and circuits, as presented in Section IV. Architectural exploration of adaptive SVD algorithm is demonstrated in Section V. The power/area reduction concepts are experimentally validated as described in Section VII for an architecture optimized for 0.4 V operation. Section VII concludes the paper.

II. Multi-Antenna Signal Processing

The explosive growth of wireless communications, spurred by the availability of unlicensed spectrum bands, has fueled the deployment of complex signal processing and coding schemes for achieving higher data rates and more robust transmission. One direction in algorithms and systems research is in the use of MIMO systems to exploit degrees of freedom in the wireless channel and effectively increase its capacity. MIMO technology has been already adopted for use in next-generation WLANs. For example, to satisfy a growing need for higher capacity and extended range, OFDM-based WLAN devices [1] are moving to using MIMO algorithms as being defined in the IEEE 802.11n standard [2].

The computational requirements of baseband signal processing for MIMO algorithms are significantly higher than those of currently deployed systems. Ideally, a complete MIMO channel decomposition would be performed independently in each of the narrowband sub-carriers, which would produce a computational increase that outstrips the improvements provided by scaling of technology alone. To illustrate the complexity, we select a multi-antenna algorithm that can deliver up to 250 Mb/s in 16 frequency sub-channels.

A. Multi-Path MIMO Channel

Multi-antenna technology can be used to improve robustness or increase capacity of a wireless link [3]. Multiple antennas have traditionally been used as a method to overcome channel fading through increased diversity [4]. Link robustness is improved by multi-path averaging as shown in Fig. 1. By sending the same information over different paths and averaging over multiple independently-faded replicas at the receiver, more reliable communication is achieved. The number of averaging paths can be artificially increased by sending the same signal over multiple antennas. For example, if the transmitted signal is replicated over n different paths, which correspond to n transmit antennas, the average error probability decays as $1/\mathrm{SNR}^n$ as opposed to $1/\mathrm{SNR}$ for the single-antenna case. Under these conditions, when the error probability decays as $1/\mathrm{SNR}^n$, the diversity gain is equal to n [3]. Furthermore, having both multiple transmit and multiple receive antennas improves the link reliability by essentially averaging over multiple independently-faded paths. For a system with m transmit and n receive antennas, the maximum achievable diversity gain is $m \cdot n$.

Another potential of MIMO systems is increased capacity. The goal is to spatially localize transmission beams, so that independent data streams can be transmitted over multiple antennas

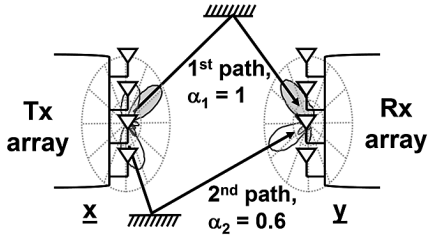


Fig. 1. Multi-path MIMO wireless channel increases robustness through multi-path averaging or increases capacity through spatial multiplexing. MIMO channel is a complex matrix formed of transfer functions between individual antenna pairs.

for increased capacity. This method, known as spatial multiplexing, takes advantage of fading by constructively combining paths that fade independently, which increases the degrees of freedom available for communication. The spatial-multiplexing gain can be quantified with the channel capacity increase. Foschini [4] showed that for an $m \times n$ system, under certain channel conditions, the capacity of the channel increases linearly with the number of spatial channels. The number of available spatial channels is the minimum between the number of transmit antennas m and the number of receive antennas n. In a MIMO system, channel is a complex matrix $\boldsymbol{H}$ formed of transfer functions between individual antenna pairs. Vectors $\underline{\boldsymbol{x}}$ and $\underline{\boldsymbol{y}}$ in Fig. 1 represent transmit and receive symbols, respectively. The goal is thus to estimate gains of the spatial sub-channels from $\underline{\boldsymbol{x}}$ and $\underline{\boldsymbol{y}}$ information.

MIMO channel decoding can be performed in several different ways. The pioneering work in MIMO decoding [4], [5] by Foschini resulted in formulation of Bell Labs Layered Space Time (BLAST) algorithms. In BLAST, codewords are arranged in a space-time grid. The data is encoded over multiple transmit antennas and distributed in time. Depending on how the data is organized in the space-time grid, there are diagonal [4] and vertical [5] architectures. Another approach to channel decoding is QR decomposition, which for a channel matrix $\boldsymbol{H}$ is a factorization $\boldsymbol{H} = \boldsymbol{Q} \cdot \boldsymbol{R}$, where $\boldsymbol{Q}$ is a unitary matrix and $\boldsymbol{R}$ is an upper triangular matrix. This method is traditionally used to compute sub-spaces in various beam-forming algorithms [6]. Full channel decoupling can be done using more computationally demanding methods such as singular value decomposition (SVD).

B. Singular Value Decomposition

Singular value decomposition is an optimal way to extract spatial multiplexing gains in MIMO channels [7]. Concept of the SVD-based channel decoupling is illustrated in Fig. 2. Channel matrix $\boldsymbol{H}$ can be represented as a product of $\boldsymbol{U}$, $\boldsymbol{\Sigma}$, and $\boldsymbol{V}^{\dagger}$, where $\boldsymbol{U}$ and $\boldsymbol{V}$ are unitary, and Σ is a diagonal matrix. SVD can be also viewed as a composition of three operations: a rotation $(\boldsymbol{U})$, a scaling operation $(\boldsymbol{\Sigma})$, and another rotation $(\boldsymbol{V}^{\dagger})$. When the channel matrix $\boldsymbol{H}$ is partially known to the transmitter, the optimal strategy is to transmit independent streams in the directions of the eigenvectors of $\boldsymbol{H}^{\dagger}\boldsymbol{H}$ [3]. Projection of modulated symbols $\underline{\boldsymbol{x}}'$ onto $\boldsymbol{V}$ matrix essentially directs the transmit symbols along eigen-modes of the fading channel. If the received signal $\underline{\boldsymbol{y}}$ is post-processed by rotating

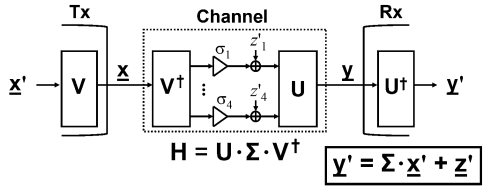


Fig. 2. Concept of singular value decomposition applied to MIMO channel matrix H. The goal is to estimate gains of spatial MIMO channels, which are given by the Σ matrix.

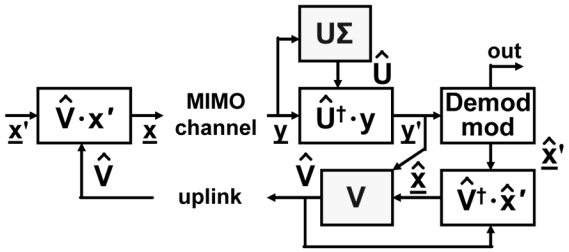


Fig. 3. Adaptive blind-tracking SVD algorithm. Estimation of spatial channel gains is done at the receiver in the $U\Sigma$ block.

$\underline{y}$ along $U^{\dagger}$ matrix, the channel between $\underline{x}'$ and $\underline{y}'$ appears fully orthogonal. Then, independent data streams can be sent through the spatial sub-channels with gains corresponding to the entries in the Σ matrix. At the receiver, data streams arrive orthogonally without interference between the streams. The SVD matrices have to be adaptively estimated to track time-varying channel conditions.

Finding SVD of a channel matrix is a multidimensional signal processing task dealing with vector and matrix arithmetic. This sort of computation involves hundreds of add and multiply operations, and may also need divide and square root. The overall complexity is well beyond the complexity of standard communication blocks such as an FFT or a Viterbi decoder, so the choice of architecture that minimizes power and area of the SVD becomes an interesting question.

C. Adaptive Blind-Tracking SVD Algorithm

Fig. 3 is a block diagram of an adaptive blind-tracking SVD algorithm proposed by Poon [8]. The core of the algorithm is in the $U\Sigma$ and V blocks, which estimate corresponding matrices. Hat symbol $(\hat{})$ is used to indicate estimates. Channel decoupling is done at the receiver. As long as there is a sizable number of received symbols $\underline{y}$ within a fraction of the coherence time, the receiver can estimate U and Σ from the received data alone. Tracking of the V matrix is based on decision-directed estimates of the transmitted symbols $\underline{\hat{x}}$. V matrix is periodically sent to the transmitter through the feedback channel.

MIMO decoder specifications for the chip we implemented are derived from the following system: 4×4 antenna system with variable-constellation PSK modulation, 16 MHz of channel bandwidth utilized by 16 sub-carriers (corresponding to the narrowband frequency sub-channels). The chip presented in this paper integrates the $U\Sigma$ algorithm and rotation along $U^{\dagger}$ matrix, which is over 80% of complexity of the entire SVD. Back-channel information about the V matrix is used at the transmitter, so integration of the V tracking algorithm would be required in the complete system realization.

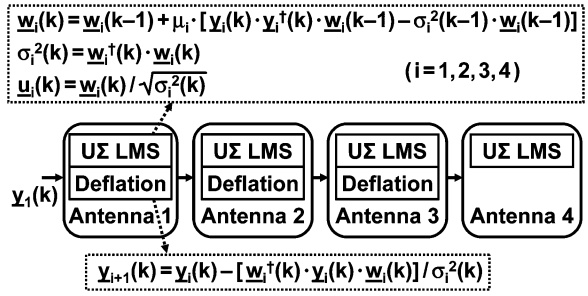


Fig. 4. LMS-based estimation of gains in spatial channels. The arithmetic complexity is about 300 adders, 400 multipliers, and 8 square roots and dividers. This level of complexity is hard to optimize in RTL.

The $U\Sigma$ block performs sequential estimation of the eigen-pairs, eigenvectors and eigenvalues, using the adaptive MMSE-based algorithm from [8]. Traditionally, this kind of tracking is done by evaluating the eigen-pairs of the autocorrelation matrix. The algorithm shown in Fig. 4 reduces matrix arithmetic to vector-based arithmetic with additional square root and division, which greatly reduces computational complexity. Estimation of eigen-pairs starts by finding the dominant eigenvalue σ_1, using the full rank information $\underline{y}_1$. The $U\Sigma$ LMS recursive algorithm in Fig. 4 estimates $\underline{w}$, which is the product of the eigenvector $\underline{u}$ and the eigenvalue σ. Each symbol period, complex vector $\underline{w}$ is updated with the LMS correction factor. Based on $\underline{w}$, the algorithm computes σ and $\underline{u}$. Then, the projection of $\underline{y}_1$ on the direction of the tracked dominant eigenvector is removed, and the algorithm is applied on the residue to track the second dominant eigen-mode. The cancellation and tracking process proceeds until all eigen-pairs are estimated. Step-size is adaptively adjusted based on the estimated eigenvalues. The square root and division are implemented using Newton–Rhapson iterative formulas, which have quadratic convergence [9]. In implementation terms, this means that each iteration resolves two bits of accuracy [10]. Under slowly varying input, which is the case with MIMO channel data, the result can be taken as the initial condition for the next iteration to obtain the answer in a single iteration.

Computational complexity of the algorithm can be inferred from formulas in Fig. 4 by decomposing complex-valued vector operands into equivalent real-valued scalar operands. Overall, the algorithm requires about 300 adders, 400 multipliers, and 8 square roots and dividers. Additionally, the operations are distributed around nested feedback loops due to recursive nature of the algorithm.

III. Design Methodology

The design methodology is based on balancing sensitivities at multiple levels of design description. Basic building blocks, such as adders and multipliers are characterized at circuit-level for power and delay, over varying wordlenghts and switching activities. The algorithm is mapped to architecture in a top-down process, with physical-level data mapped back into a high-level design description. An energy and area optimal mapping of an

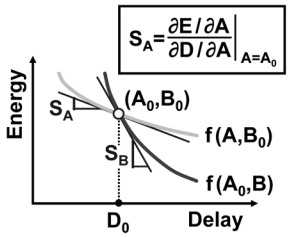


Fig. 5. Sensitivity-based optimization. Energy-delay sensitivity is the slope of tuning each variable in the energy-delay space. The highest energy efficiency is achieved by equalizing sensitivity to all variables.

algorithm into architecture and a circuit with a required performance is achieved when all energy-performance and area-performance sensitivities for various blocks in the same level of hierarchy as well as across different levels of hierarchy are balanced.

The high-level design description is created in Simulink. The same description is used for functional simulation, rapid system evaluation on an FPGA array, and ASIC synthesis. This unified design description also allows for a seamless FPGA-assisted verification of the fabricated ASIC. The design infrastructure enables large design space exploration including algorithms, architectures, and circuits, thus enabling for optimization across various abstraction layers.

A. Sensitivity-Based Optimization Framework

The goal of mapping of an algorithm into architecture is to achieve minimum energy under performance and area constraints. The plot in Fig. 5 illustrates energy-delay sensitivity, which is simply the slope of a curve in the energy-delay space that corresponds to tuning of a circuit-level design variable. The variables at the circuit-level can be gate sizes, transistor thresholds and supply voltages, while they can correspond to logic depth at a higher level. The most energy-efficient design is achieved when the sensitivities to all variables are equal, or when the variables reach their limits [11]. Similarly, the most area-efficient design is achieved when the area-performance sensitivities are balanced. Area optimization for dedicated DSP algorithms differs from the optimization for microprocessors [12], [13].

The results from [11] were obtained on simple datapath blocks, using continuous tuning of transistor sizes, supply and threshold voltage. It has been demonstrated that 65% of energy can be saved without delay penalty or, equivalently, the delay can be improved by 25% without energy increase, compared to the design sized for minimum delay at nominal V_{DD} and V_{TH}. In this work, we demonstrate large energy savings on a complex design using the same principles of optimization with small changes in a conventional ASIC design methodology.

The key observations from prior work [11] are used as guidelines for the ASIC design: 1) gate sizing is the most effective in reducing the energy near (up to 20%) the minimum delay point; 2) voltage scaling is the most effective for large incremental delays; and 3) threshold adjustment is the least effective due to broad energy minimum as a function of leakage-to-switching ratio. In order to apply the sensitivity-based optimization on a hierarchical, complex design, these principles are applied to a timing-driven logic synthesis with discrete gate sizes.

B. Block Characterization

Circuit-level characterization augments basic design library blocks with technology-dependent information for power, speed, and area, thus providing complete information for block-based architecture design. Basic primitives include simple operators such as add, multiply, shift, mux, register and so on, so a full characterization over a range of latency and word-size parameters is possible. The logic depth is chosen such that the registers are inserted between these logic primitives when composing a larger function.

Plot in Fig. 6(a) shows typical curves for add and multiply blocks, for a fixed number of bits, where all points on the curves are characterized for power and area. For a range of interest, the slope of the *E–D* curves as a function of logic depth for adders and multipliers are similar, and can be matched to the sensitivity to lower-level parameters (supply and sizing). The difference in gate complexity of these blocks translates into different latencies of their implementations, for the equal cycle time.

Concept of balancing sensitivity to supply (V_{DD}) and gate size (W) is graphically illustrated in Fig. 6(b). Design sized for minimum delay at nominal supply has very large sensitivity to gate sizing. The sensitivity to V_{DD} and W is balanced by downsizing the logic, which results in some delay increase. As V_{DD} scales down to reach the target delay, W has to adjust to track reduced sensitivity to V_{DD}. Combining results of energy-delay and latency versus cycle time characterization provides insight into optimal supply voltage, gate size, and logic depth (L_d). Pipeline balancing at the block level thus enables hierarchical expansion for full algorithm realization.

C. Simulink Design Environment

Design re-entry in various phases of system development is a standard practice today. A design is entered in various forms by different designers or different teams, resulting in heterogeneous design descriptions. Algorithm developers tend to work in C or Matlab environment, which has an array of built-in functions convenient for quick algorithm modeling and verification. SystemC [14] is another behavioral entry, closer to traditional hardware descriptions like Verilog or VHDL, but requires more sophisticated programming skills. The architectural description is then created separately by hardware designers who have to completely re-enter the design in a hardware description language (HDL). This dramatically limits the possibility of exploring various architectures for a given algorithm or modifying the algorithm to better map onto the underlying technology.

We use unified graphical Simulink design environment [15], which is widely adopted by algorithm developers. An algorithm is entered only once is a graphical block form, that easily maps onto a corresponding dataflow architecture. This approach enables algorithm verification or hardware emulation, and also provides abstract view of design architecture, thereby avoiding costly design re-entry. The methodology relies on timed-dataflow block-level representation of an algorithm. Fixed-point block set includes basic operators such as add,

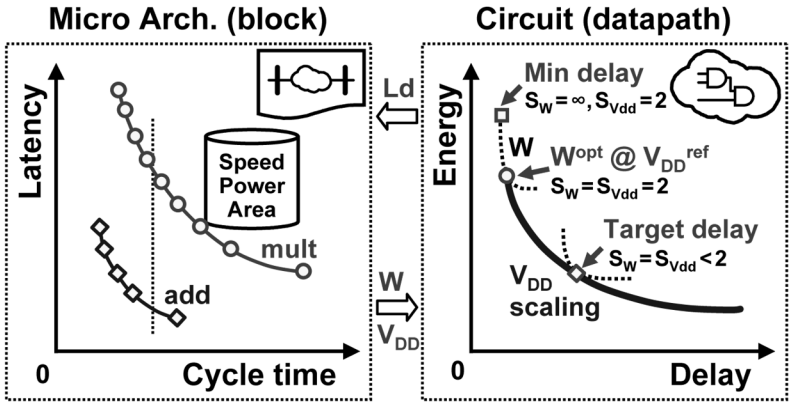


Fig. 6. Block characterization methodology. (a) Block library is augmented with information for speed, power, and area, in addition to existing wordlength and latency parameters. (b) Latency versus cycle time for a block is characterized over a range of supply voltages. Optimal logic depth (L_d), supply (V_{DD}) and gate size (W) for a block is determined.

multiply, shift, mux, and so on with notion of hardware parameters such as finite wordlength and latency. This is the level of abstraction we work with to efficiently optimize the architecture.

D. FPGA-Assisted ASIC Design Flow

Design cycle includes algorithm modeling, design optimizations, and final ASIC verification, all carried out using Simulink environment. Once the bit-true, cycle-accurate behavior is established, the Simulink description of the block interconnects is used to generate the hardware description for mapping the design onto a FPGA array. The same description is also used as an entry into the ASIC tool flow.

The ASIC flow shown in Fig. 7 takes block-level connectivity from Simulink and generates HDL for chip synthesis using an in-house tool [16]. The tool builds scripts to synthesize basic primitives and allows constraints to be passed to synthesis to improve compilation results. The tool also does first synthesis to map the complete design to the gate level, and HDL simulation to verify functional equivalence between the two hardware descriptions, using test vectors from Simulink. The synthesized netlist is then optimized using a set of custom scripts for register retiming and logic optimization, before going to the final stage of physical layout synthesis. With technology-dependent characterization of basic blocks for speed, power, and area, the architecture can be also optimized in Simulink, without the need for excessive iterations through the design flow.

IV. Algorithm–Architecture–Circuit Co-Design

Design environment presented in Section III is used to evaluate various architectural techniques in the energy-area-performance space. Prior work evaluated the impact of circuit-level decisions such as gate sizing and circuit topology on various FIR filter architectures [17], based on gate size characterization in a standard VLSI synthesis environment. The unified methodology presented in this paper encompasses more degrees of freedom in the design and is scalable to complex designs.

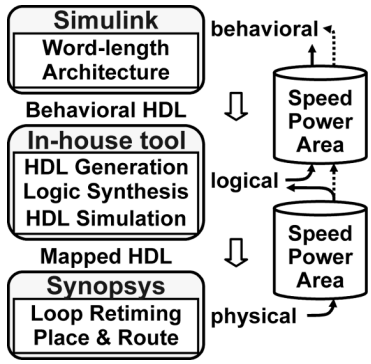


Fig. 7. ASIC design flow. In-house tool generates HDL for chip synthesis based on the FPGA Simulink design entry. The ASIC flow maintains functional equivalency with the FPGA flow to allow FPGA-based ASIC test. The architecture is optimized in Simulink based on architectural feedback (speed, power, area) from synthesis.

A. Wordlength Optimization

The first step in the design optimization is wordlength reduction, which is done using FFC, a custom floating-to-fixed point conversion tool [18]. The objective is to minimize hardware utilization subject to user specified mean-square quantization error.

The use of FFC is illustrated in Fig. 8. An FFC spec-marker defines the mean-square error (MSE) specification for desired nodes in the system, typically output, while the hardware cost estimation block calculates FPGA hardware utilization in terms of FPGA slices, flip-flops, and look-up tables. After the MSE constraints have been formulated, the FFC tool runs a number of simulations to calculate wordlength sensitivities and applies perturbation theory [19] to find the optimal result. Integer bits are determined based on input data dependent range detection. The optimization is performed hierarchically, to relax the memory requirements and reduce the simulation time. The tool can also append some guard bits in order to

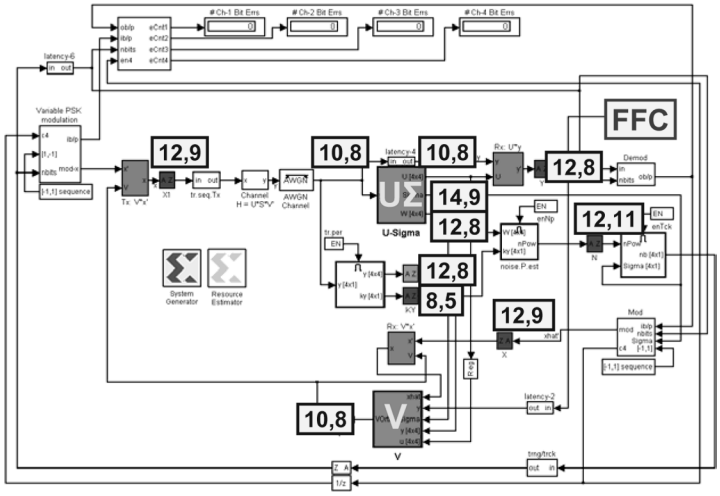


Fig. 8. Simulink model of a MIMO transceiver used to evaluate the SVD algorithm. Optimal top-level wordlengths are indicated, the numbers are (total, fractional) bits.

avoid long simulation times. A detailed description of the tool with examples and a user tutorial can be found in [20].

B. Architectural Transformations

Architectural transformations, such as pipelining, parallelism, time-multiplexing and folding are applied successively to a bit-true, cycle-accurate Simulink model to minimize power and area under throughput/latency constraints. Impact of individual transformations on the overall chip characteristics highly depends on the circuit-level performance.

1) Parallelism and Pipelining: The concepts of parallelism and pipelining (Fig. 9) are well-known in architecture design [21], [22]. Parallelism distributes computation over several parallel branches, while pipelining inserts pipeline registers between logic blocks. When combined with adjustment in supply voltage, both techniques reduce the energy by relaxing the delay of the datapath logic (blocks A and B in Fig. 9). The area-energy-delay plane in Fig. 9 shows energy-delay tradeoff in the logic blocks, and the area of corresponding architecture. Starting from the reference architecture, parallelism and pipelining relax the delay constraint to reduce energy at the expense of increased area (shaded blocks). Alternatively, with constant clock frequencies, parallelism and pipelining trade off the increase in area for higher throughputs.

2) Time-Multiplexing: Time-multiplexing is another commonly used architectural technique. The concept is illustrated in Fig. 10 on an example of a datapath, where block A is some logic function that evaluates multiple incoming streams of data. Time-multiplexing reduces the area through resource sharing, but needs a higher clock rate and thus higher supply voltage to maintain the throughput, as shown in the area-energy-delay plane in Fig. 10. Alternatively, time-multiplexing can lower the throughput by maintaining the clock rate.

3) Data-Stream Interleaving: Data-stream interleaving is a major technique for area reduction in multidimensional signal processing. Recursive operation is the underlying principle in the LMS-based tracking of eigen-modes, so simple case is examined in Fig. 11 to illustrate the concept. Top diagram illustrates block-level representation of the recursive operation on

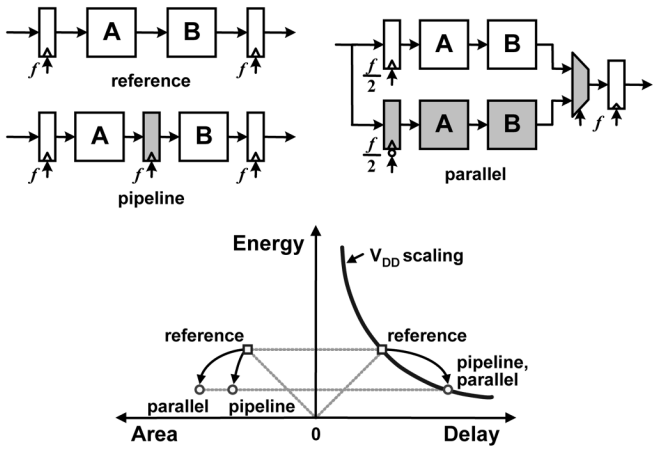


Fig. 9. Parallelism and pipelining increase delay of the underlying logic blocks and reduce energy through voltage scaling. Area of the resulting implementation increases.

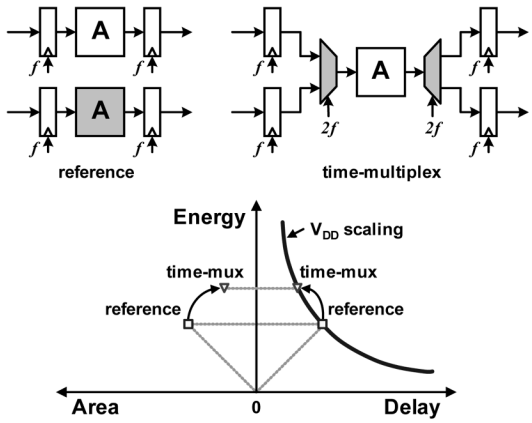


Fig. 10. Time multiplexing reduces delay of the underlying logic blocks and increases energy through increase in supply voltage. Area of the resulting implementation decreases.

the right. The output sample $z(k)$ is a sum of a current input and a delayed and scaled version of a previous output. Clock frequency f_{Clk} corresponds to the sample time. Simple model

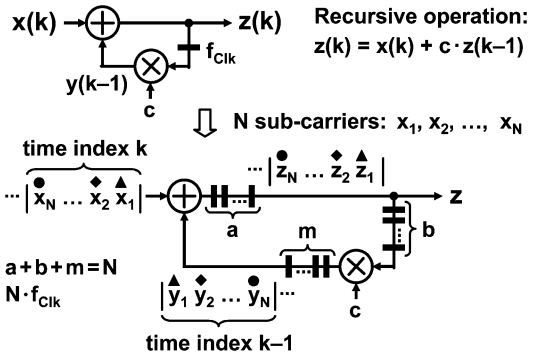


Fig. 11. Data-stream interleaving. Independent streams of data are up-sampled and time-interleaved to maximize resource sharing for reduced area.

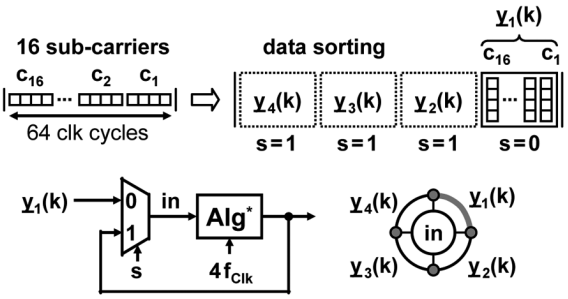


Fig. 12. Folding allows resource sharing in serially ordered signal processing. *Alg* block performs a recursive operation and needs deeper pipelining to support folding. The example illustrates folding of the $U\Sigma$ algorithm from Fig. 4.

shown in the top diagram assumes zero latency of the add and multiply blocks. The model is refined by adding appropriate latency (a, m) at the output of the add and multiply blocks, respectively, as shown in the bottom diagram. This creates an opportunity to interleave multiple streams of data and reduce area compared to the case of direct-mapped parallel realization. Data-stream interleaving is directly applicable to multiple carriers corresponding to narrowband sub-channels.

If the number of sub-carriers N exceeds the latency required by arithmetic blocks, balancing registers are inserted. Integers a, m, and b represent latency of the adder, multiplier, and the number of balancing pipeline registers, respectively. To maintain the overall throughput, N incoming streams of data are up-sampled by N and time-interleaved. Data-stream interleaving is applicable to parallel execution of independent data streams.

4) Folding: For time-serial ordering of data streams, folding as shown in Fig. 12 is used [23]. The *Alg* block performs some recursive operation such as that illustrated in Fig. 11. The multiplexer can take the output of the *Alg* block or select incoming data stream $\underline{\boldsymbol{y}}_1$. After processing $\underline{\boldsymbol{y}}_1$, the output of *Alg* is folded over in time, back to its input, to compute $\underline{\boldsymbol{y}}_2$, $\underline{\boldsymbol{y}}_3$, and $\underline{\boldsymbol{y}}_4$ as shown in the life-chart. As an example, Fig. 12 illustrates folding of four serially ordered *Alg* blocks, each processing 16 sub-carriers. Each of the carriers is a vector of real and imaginary data, sorted in time and space to occupy 16 consecutive clock cycles and free up the slots needed to store the data from sub-sequent *Alg* blocks.

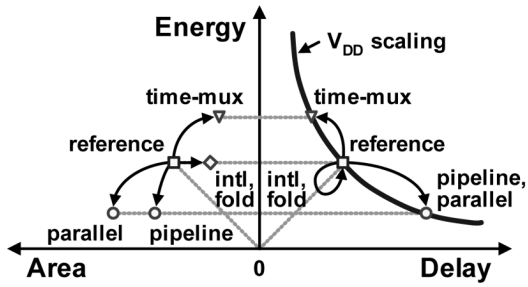


Fig. 13. Energy-delay characterization of datapath logic is a compact characterization for comparing multiple architectural realizations. Area of the resulting implementation is also considered.

Both interleaving and folding introduce pipeline registers to store the internal state. To maintain the overall throughput, pipelining is combined with up-sampling. This transformation maps back to approximately the same point in the energy-delay space, as indicated in Fig. 13. Due to shared pipeline logic, area of the resulting implementation is reduced roughly by the degree of interleaving or folding (less the area of the additional pipeline registers).

Energy-delay characterization of datapath logic provides simple framework to evaluate multiple architectural realizations, as summarized in Fig. 13. This plot is the basis for architectural selection under fixed throughput constraint. Simple procedure follows: the goal is to move towards desired point on the optimal energy-delay curve, while minimizing the area. "Reference" point in the energy-delay space, for example, is a good tradeoff point since outside of this region the design becomes too costly in terms of delay or energy.

5) Loop Retiming: Adding pipeline registers needed for interleaving and folding raises the question of their optimal distribution around the loops. Loop retiming is a technique of distributing pipeline registers around recursive loops [24]. Retiming on a mapped gate-level design involves moving the registers across combinational logic gates in order to minimize the critical path. To initiate retiming procedure, the first step is to assign latency to functional blocks such that all internal combinational logic blocks have the same position in the energy-delay space after retiming. This guarantees top-level optimality.

The approach to latency assignment is illustrated in Fig. 14 on the example of iterative division described by (1):

$$x_d(k+1) = x_d(k) \cdot (2 - Z \cdot x_d(k)), \quad x_d(k) \to \frac{1}{Z}, \quad k \to \infty. \tag{1}$$

This is a simple example with two nested loops, but the concepts are general. The analysis starts from a data-flow graph (DFG) representation of a function shown in Fig. 14, where a, m, and u are the latencies of pre-characterized adder, multiplier, and multiplexer blocks, respectively. This extends the retiming algorithm from [25] that assumes equal latency in add and multiply blocks, as applied to FIR filters. Starting from the DFG

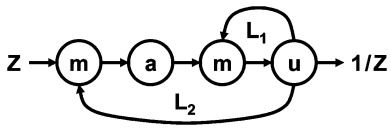


Fig. 14. Loop retiming involves latency assignment to processing blocks. Example of data-flow graph model of iterative division based on Newton–Rhapson method. (m, a, u indicate latency).

representation, the latency assignment is carried out as follows. For each loop, constraints are formulated as described in (2):

$$\begin{aligned} \text{Loop } L_1: &\quad m+u+b_1 = N \\ \text{Loop } L_2: &\quad 2m+a+u+b_2 = N \\ \text{I/O latency } (\text{div}^{(1)}): &\quad 2m+a+u \end{aligned} \tag{2}$$

where b_1 and b_2 are the number of balancing registers needed to satisfy the loop latency N. The first two equations are loop constraints for the divider block; the third equation is parameterized I/O latency of the divider. The I/O latency is needed for hierarchical expansion. After formulating loop constraints, the next step is to solve for the latency parameters (m, a, u). Loop constraints L_1 and L_2 appear under-constrained since there are more variables than equations. This is resolved by the fact that the cycle time is common for all blocks. The cycle time determines latency parameters (m, a, u) in various functional blocks, based on block characterization in the latency versus cycle time space [26]. Finally, the balancing latency b_1 and b_2 is determined for each of the loops in order to meet the loop latency constraint N. This modularity decomposes a difficult global loop retiming problem to a number of very simple feed-forward cases since the registers do not need to cross block boundaries.

The latency assignment strategy above can be easily expanded to multiple layers of hierarchy. Blocks at each hierarchical level are characterized with internal loop constraints as well as latency from their primary inputs to their primary outputs. For example, I/O latency in (3) indicates the latency of a divider block at level 1 of hierarchy, $\text{div}^{(1)}$. This information is used to derive I/O latencies and loop constraints at the next level of hierarchy. At the top-level, loop constraints from all the underlying levels are thus considered.

6) Delayed Iteration: Delayed iteration is another technique to consider in the process of improving power and area of the architecture. The idea is to relax timing constraints of the pipelined logic by adding an extra cycle. Ideally, all loops should have the same loop latency and no balancing registers needed, which implies the same criticality of the datapaths from the performance perspective. When majority of loops have similar latency and only a few loops are critical (have long latency), then forcing the same latency constraint on all the loops would require a large number of additional balancing registers in the non-critical loops, creating a large timing slack and resulting in excess power consumption. A delayed iteration improves the design by tolerating delayed data samples in the critical loops. This effectively extends latency in previously performance critical loops and results in a better balance of pipeline logic delay. If this is algorithmically possible, requirements for power and area can be much improved.

V. Power/Area Optimal SVD Architecture

A directly-mapped fully-parallel SVD architecture exceeds the area of the baseband signal processing in WLAN applications (typically around 20 mm^2 in commercial chips) by an order of magnitude. At the same time, the clock speed required to implement the algorithm in such a parallel architecture is significantly lower than what is easily achievable in current technology. By using a series of architectural transformations the area is brought within the limits. Concurrent architectural and circuit optimization techniques are used to minimize the energy per operation, starting from a direct-mapped parallel architecture at a nominal supply. This process is repeated until all the techniques are balanced, [27].

A. Design Specifications

Main design specification is the throughput constraint for the algorithm, which is derived according to the 4×4 16-carrier system specification outlined in Section III. Since 1 MHz wide sub-channels require 1MS/s to process the data, the requirement is thus to process 16 parallel streams of data at a 1 MHz rate.

1) Starting Point: Direct-mapped parallel architecture is used as a starting point in architectural exploration, because this point is well defined. Feasible architectures are then derived from the throughput constraint for the algorithm, power and area budget. A direct-mapped parallel architecture would need 1 MHz baseline clock if unit operation of the $U\Sigma$ algorithm (LMS-based update) can be realized with one cycle of latency.

2) Architectural Optimization: Architectural optimization is based on datapath characterization as shown in Fig. 13 and presented in Section IV-B. The first step is to check the feasibility of the starting architecture. One recursion of the $U\Sigma$ algorithm requires 6 real multiply, 11 add, and 2 mux operations. This is feasible within 1 μs in a 90 nm technology, even at a reduced V_{DD}, however with exceedingly large area. To minimize the area, some degree of interleaving and folding is needed, resulting in a variety of architectural choices.

B. Architecture Optimized for 0.4 V

The simplest way to minimize area without impacting energy-delay characteristics of pipeline logic, to a first order, is to employ interleaving and folding. The most natural scenario for the 4×4 $U\Sigma$ algorithm shown in Fig. 4, operating with 16 sub-carriers, is to interleave the 16 parallel sub-carriers and fold over 4 antennas. Clock specification for the resulting architecture then becomes 64 MHz (1 MHz $\times$ 16 sub-carriers $\times$ 4 antennas). This speed can be reached at scaled supply of 0.4 V in a 90 nm technology and corresponds to the energy-delay sensitivity of 0.8. This is a good tradeoff between energy and delay, corresponding to 0.8% change in energy for 1% change in delay. Interleaving and folding reduce the area by a factor of 36 compared to the direct-mapped parallel architecture. The area of the logic blocks is shared by the sub-carriers and also over antennas, leading to area reduction. The design is further optimized as follows.

Step 1: Wordlength optimization Starting from all 16-bit realization of the algorithm, we apply hierarchical wordlength optimization for a 30% reduction in energy and area. Area reduction is estimated from the resource estimation tool within FFC, which estimates hardware area based on pre-characterized library blocks. To a first order, this directly translates to energy reduction due to reduced gate capacitance. The next step is logic synthesis that incorporates gate sizing and supply voltage optimizations.

Step 2: Gate size and V_{DD} optimization. In order to maximize energy efficiency, supply voltage scaling and gate sizing are needed. The voltage can be scaled down to 0.4 V, without compromising static VTC characteristic of the logic gates, [27]. Since the standard-cell library was characterized at 1 V, the timing specifications were translated to 1 V according to transistor-level simulations. Designing for 64 MHz at 0.4 V in the SS corner translates to 512 MHz equivalent timing constraint for logic synthesis under the worst case model (0.9 V, 125 °C).

Due to limitation of synthesis tool, we balance the tradeoffs with respect to gate sizing (W) and supply voltage (V_{DD}) sequentially. From prior work [11], we know that sizing is the most effective at small incremental delays compared to the minimum delay. To approximate that, the design is first synthesized with a 20% slack, followed by incremental compilation to utilize benefits of sizing. The gate sizing step results in a 40% reduction in energy and a 20% reduction in area of the standard-cell implementation, based on synthesis estimates. Gate sizing transformation is also shown graphically in Fig. 15. With the supply voltage scaled down to 0.4 V, the sensitivities are balanced at 0.8, resulting in the design optimized for target speed of 64 MHz. V_{DD} scaling provides a 7× reduction in energy per operation. At the optimal V_{DD} and W, energy-delay curves of sizing and V_{DD} are tangent as illustrated in Fig. 15, corresponding to equal sensitivity [26]. As part of synthesis optimization on a mapped netlist, loop retiming is performed with $N = 64$, corresponding to 16 interleaved sub-carriers and folding over four antennas. This is the final design carried through physical synthesis. Impact of individual techniques on chip energy and area is summarized in Table I. Compared to the 16-bit direct-mapped parallel realization with gates optimized for speed at nominal voltage, the total area reduction of the final design is 64× and the total energy reduction is 16×. Section VI presents experimental results.

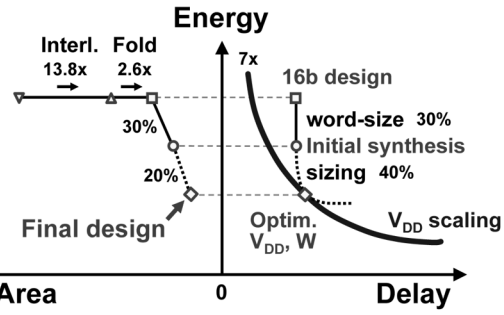


Fig. 15. SVD algorithm optimization in the area-energy-delay space. The plot shows the impact of interleaving, folding, word-size, gate sizing, and supply voltage optimizations on energy/op and area. Compared to 16-bit design with direct-mapped parallel realization using gates optimized for speed at nominal V_{DD}, the final design achieves 64× reduction in area and 16× reduction in energy/op (12-bit add equivalent).

TABLE I
SUMMARY OF MAIN OPTIMIZATION TECHNIQUES

Opt technique	Area reduction	Energy reduction
Wordlength opt	30%	30%
Gate sizing (W)	20%	40%
V_{DD} scaling	N/A	7×
Interleaving/folding	13.8× / 2.6×	−2%

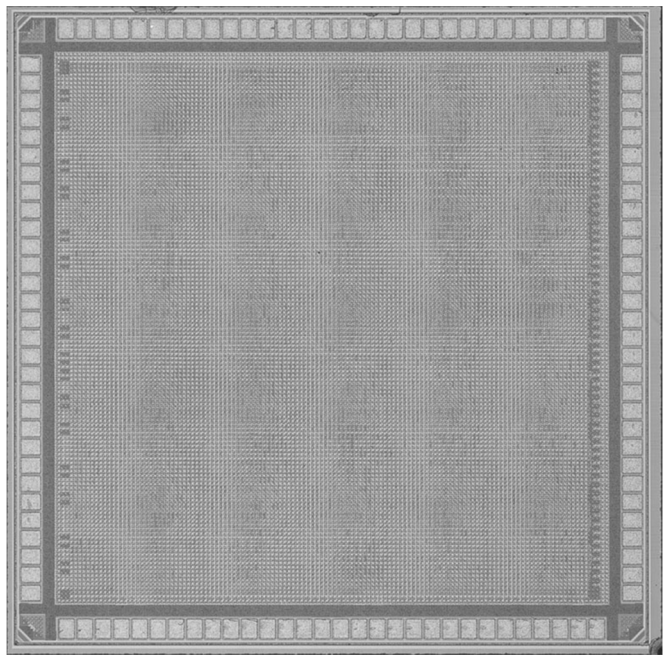


Fig. 16. Die photo of 4×4 SVD chip. The core is optimized for 0.4 V achieving 100 MHz operation. The chip is also functional at 255 mV with a 10 MHz clock.

VI. EXPERIMENTAL VERIFICATION

Fig. 16 is a die photo of the first ASIC realization of an adaptive 4×4 singular value decomposition. The chip is fully optimized in a standard-V_T 90 nm CMOS process from STMicroelectronics. Chip features are summarized in Table II. The core area is 3.5 mm^2 with 88% density of standard-cell placement in the final design, after clock tree optimization and hold-time fixing buffer insertion. The total chip area with I/O pads is 5.1 mm^2. Separate supply voltages are used for the chip core and I/O pads. The core V_{DD} is tunable in a 0.2 V–1 V range and the I/O pads run at a 1 V supply. The standard cross-coupled PMOS level converting circuit [28] is used. A dual-well approach is used in the layout to isolate the core from I/O supply and maintain the standard-cell height. Level converting cells are placed at the core boundary, close to the output pads. Level converters are used only at the output, while core inputs are driven with a full-swing signal. Chip outputs are delivered to the printed circuit board (PCB) through digital pads which operate at a standard 1 V supply. An extra analog pad, connected to the input of a level converter, is used to allow testing of the level converter.

Simulink environment is used in experimental verification through real-time hardware co-simulation. This is the final

TABLE II
CHIP FEATURES

Technology	90nm CMOS
Core area	1.9×1.9 mm
Die area	2.3×2.3 mm
Pad count	120
IO/core V_{DD}	1V / 0.4V
Cell count	420,304
Frequency	100 MHz
P (act/leak)	30mW / 4mW
Efficiency	2.1GOPS/mW

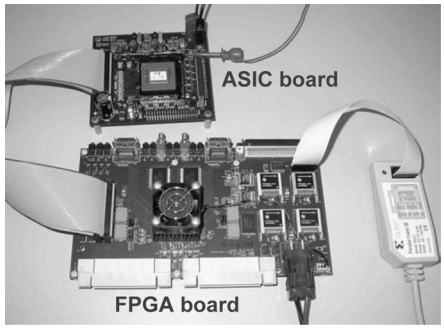


Fig. 17. The use of silicon infrastructure for hardware based ASIC verification. FPGA stimulates the ASIC over general purpose I/Os and compares outputs in real-time.

touch-point between the FPGA and ASIC flows, which also demonstrates full functionality of our design infrastructure.

A. FPGA-Based ASIC Verification

An efficient ASIC verification is performed with the use of an FPGA emulation board, as shown in Fig. 17. The algorithm is programmed onto the FPGA, which stimulates the ASIC over general purpose I/Os using test vectors exported from Simulink. The FPGA feeds the data into the ASIC and samples outputs for real-time comparison. Outputs of the comparison are stored in block RAMs on the FPGA, which can be read out through the serial port. Signals that control the read and write ports of the block RAMs as well as other controls, such as reset and clock enable, are set by the user. User can bring in external clock or use internally generated clock from the FPGA board.

Communication between the FPGA and the ASIC board is done through general purpose I/Os operating at 3.3 V, with single-ended signaling, as required by the FPGA. Level conversion from 3.3 V (FPGA) to 1 V (ASIC) is realized on the ASIC board with a 50 Ω termination resistive dividers, while the up-conversion is realized with integrated comparators. The FPGA chip used in the experiment (Xilinx Virtex-II Pro) has a number of dedicated 18-bit multipliers, a PowerPC processor, a memory, and a regular FPGA fabric [29]. Dedicated multipliers are used throughout the design in order to save the FPGA fabric for other functional blocks. With this approach, the entire design fits in one chip quite easily (13,400 slices, which is less than 50% utilization).

B. Measured Functionality

The result of the adaptive 4×4 eigen-mode decomposition is shown in Fig. 18. The plot shows tracking of eigenvalues

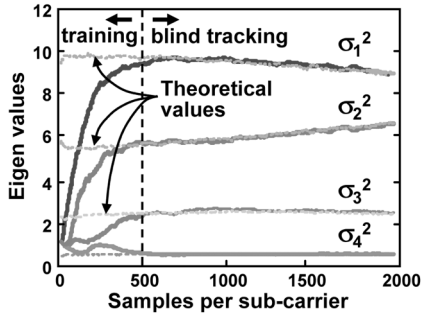


Fig. 18. Tracking of eigenvalues over time for one sub-carrier (gains of spatial channels are square root of the eigenvalues). Shown are measured and theoretical values. After training, the chip switches into blind-tracking mode.

TABLE III
AREA AND POWER OF FUNCTIONAL BLOCKS

100MHz	Power (mW)	Area (mm^2)	GOPS
UΣ LMS	20	2.31	42.6
Deflation	14	1.19	27.4

over time, for one sub-carrier. Gains of spatial sub-channels are square root of the eigenvalues displayed in Fig. 18. After the reset, the chip is trained with a stream of identity matrices and then it switches to blind tracking mode. Shown are measured and theoretical values to also illustrate tracking performance of the algorithm. In the blind tracking mode, PSK modulated data is sent over the antennas with constellations varying according to the estimated SNR. On average, 10 bits per symbol are transmitted. Although the algorithm is constrained with constant amplitude modulation, we are still able to achieve 250 Mb/s over 16 sub-carriers using adaptive PSK modulation. Due to 100 MHz operation, this is equivalent to a 25 MHz bandwidth. The achieved data-rate is thus near the upper limit of the 802.11n standard [2], which specifies theoretical 540 Mb/s using a 40 MHz bandwidth.

C. Performance and Power

The core is optimized for 0.4 V, achieving a 100 MHz operation at that voltage, consuming 34 mW of power in the full activity mode with random input data. The 100 MHz operation is measured over 9 die samples with minimum required supply voltages ranging from 385 mV to 425 mV. This die set showed a $2\times$ variation in leakage power at equivalent speed. The chip is also functional at 255 mV, running with a 10 MHz clock.

The chip is fabricated in a standard-V_T process that offers good balance between the leakage and active components of power. Since V_T is reduced, a lower V_{DD} is required for the same speed, which results in reduced overall power consumption. In active mode, the leakage and clocking power are 12% and 30% of the total power, respectively. Area and power of functional blocks are given in Table III. The total power is measured while the power breakdown is based on synthesis estimates.

D. Energy and Area Efficiency

Highest power (or energy) efficiency per computation is achieved with techniques that allow voltage scaling and down-sizing of datapath logic. The key to evaluating power

TABLE IV
SUMMARY OF 4 × 4 MIMO DECODER ASICS

Year [reference]	2005 [40]	2005 [41], [42]	2004 [43], [44]	2003 [44], [45]	2006 This work
Modulation	16-QAM	QPSK	16-QAM	QPSK	var. PSK
Detection	depth-free sphere	ML-APP	K-best sphere	V-BLAST	SVD
Bandwidth	14 MHz*	5 MHz	4 MHz*	26 MHz*	25 MHz
Technology	0.25µm	0.18µm	0.35µm	0.35µm	90nm
Clock frequency	71 MHz	123 MHz	100 MHz	80 MHz	100 MHz
Core V_{DD}	2.5V	1.8V	2.8V	2.7V	0.4V
Gate count	50k + preP.	685k	91k + preP.	190k	980k
Throughput	169 Mbps	28.8 Mbps	52 Mbps	160 Mbps	250 Mbps
Power	473 mW	257 mW	626 mW	608 mW	34 mW
Spectral Effcy (bps/Hz)	12.1	5.8	13.0	6.2	10.0

*bandwidth estimated from throughput and modulation

optimality in general is the efficiency metric [30], [31] given by (3).

$$\text{Energy efficiency} = \frac{op}{pJ} = \frac{\frac{op}{s}}{\frac{pJ}{s}} = \frac{GOPS}{mW} = \text{Power efficiency},$$
$$\text{Energy/Power efficiency} \propto CV_{DD}^2 \tag{3}$$

where op represents some average operation, such as add or multiply. Power/energy efficiency is therefore the amount of energy required for some average operation. This means that the efficiency can be improved by scaling down the supply voltage and gate size of the underlying circuits, but this has to be done simultaneously with architectural selection to maintain the performance, and also minimize area/cost of the implementation.

In this study, 12-bit equivalent addition is used as the atomic operation. Block characterization shows that an equivalent multiply is about 6 times more complex than 12-bit add in terms of area and power. The total computational complexity of the algorithm implemented is equivalent to 700 12-bit add equivalent operations. At 100 MHz, this amounts to 70 GOPS. With 34 mW of power, the resulting energy/power efficiency is thus 2.1 GOPS/mW.

Presenting the energy efficiency alone could be misleading, since it can be improved by parallelism which increases area. More complete information about the cost of a design is obtained by looking into the efficiency of silicon area utilization. In this paper, area efficiency is defined as the amount of silicon area required to perform an operation per unit time. For highest area efficiency, the goal is to maximize the utilization of datapath logic. Interleaving and folding streams of data allows datapath sharing, resulting in the overall chip core area of just 3.5 mm^2. The area of pipeline registers is about 40% of the total chip area. With 70 GOPS of computational throughput, the chip achieves 20 GOPS/mm^2 area efficiency (13.7 GOPS/mm^2 including I/O pads).

A comparison of the the key features of implementation presented in this paper with other published 4×4 MIMO chips includes modulation, spectral efficiencies, throughput and power, and is summarized in Table IV. For most of these designs, detailed power consumption analysis has not been disclosed; therefore to asses our design methodology, we compare the energy and area efficiency of the SVD chips to published baseband and media processors [32]–[39]. To make the comparison fair, the data rate and power are normalized to a 1 V 90 nm technology, and 12-bit wordlengths.

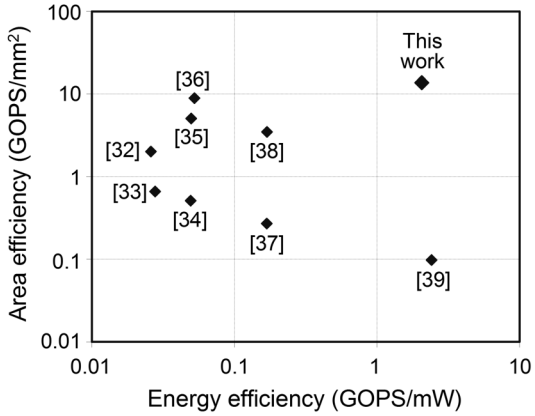


Fig. 19. Area efficiency versus energy efficiency normalized to a 1 V 90 nm process. Representative baseband and media ISSCC chips are compared against this work.

For example, hearing aid DSP processor [39] dissipates 660 μW of power with a 2.56 MHz clock at 1.05 V, executes 50 Million 22-bit operations, with the resulting energy efficiency of 75 MOPS/mW in a 0.25 μm technology. When scaled to 90 nm and accounting only for switching energy its efficiency is 2.4 GOPS/mW. If a 12% leakage overhead were included, the resulting efficiency would be 2.1 GOPS/mW, which matches our result. This stems from the fact that a voltage-scaled 1.05 V design in 0.25 μm is equivalent to a 0.42 V in a 1 V 90 nm CMOS. Comparison results are illustrated in Fig. 19. For example, the chip presented in [39] achieves similar energy efficiency, but has substantially lower area efficiency, while the chip in [36] reaches similar area efficiency but has lower energy efficiency.

VII. CONCLUSION

This work demonstrates hierarchical energy- and area-efficient mapping of an algorithm into an integrated circuit. Simulink provides the level of abstraction needed for architectural optimizations. The architectural selection is based on energy-delay-area characterization of datapath logic that

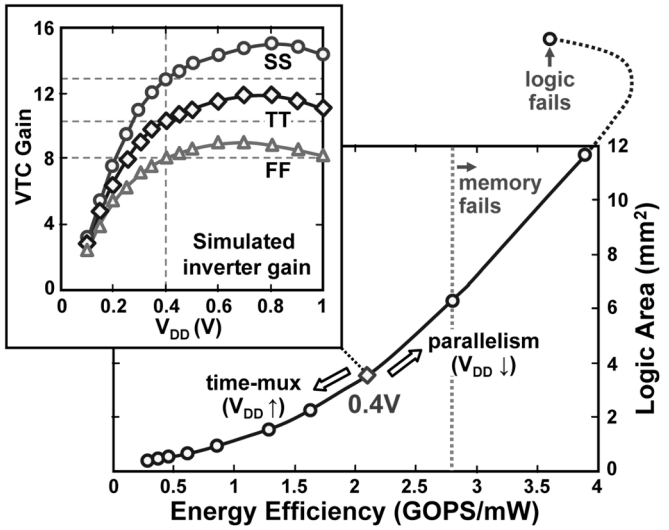


Fig. 20. Tradeoff between energy efficiency and logic area. Final architecture is optimized for 0.4 V, which has good static gain for an inverter (shown for process corners). Other architectural realizations are possible with more parallelism or time-multiplexing.

allows for comparison of multiple architectural realizations. The choice of architecture is thus highly influenced by power and performance of the underlying technology. To demonstrate the methodology, a baseband MIMO SVD signal processing algorithm over multiple carriers has been implemented in a 90 nm CMOS technology, achieving 2.1 GOPS/mW and 20 GOPS/mm^2 (13.7 GOPS/mm^2 including I/O pads).

By changing the area or performance constraints, or the optimization goal, other energy and area efficient architectures can be derived. For example, these can be obtained by varying the level of time-multiplexing or parallelism. Fig. 20 plots the trade-off between energy efficiency and gate area in various architectures. The dot indicates the design optimized for a 0.4 V operation. Starting from this point, energy efficiency can be traded for smaller area by increasing the level of time-multiplexing, which comes with an increase in supply voltage and, hence, reduced energy efficiency. Going in the direction of increased parallelism, the energy efficiency can be improved by further scaling down the supply voltage. This result is largely theoretical, since technology variability limits the minimum practical supply voltage.

ACKNOWLEDGMENT

The authors thank V. Stojanović and M. Horowitz for initial work in sensitivity-based optimization, A. Poon for discussions about the SVD algorithm, C. Shi for maintaining the FFC tool, B. Richards for automating the ASIC tool flow, and P. Droz, C. Chang, and H. Chen for help with chip testing infrastructure. The authors acknowledge Xilinx for FPGA hardware support.

REFERENCES

[1] J. Thomson *et al.*, "An integrated 802.11a baseband and MAC processor," in *IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers*, Feb. 2002, pp. 126–127.

[2] IEEE 802.11 Wireless Local Area Networks—The Working Group for WLAN Standards. [Online]. Available: http://grouper.ieee.org/groups/802/11/

[3] D. Tse and P. Viswanath, *Fundamentals of Wireless Communication*. Cambridge, U.K.: Cambridge Univ. Press, 2005.

[4] G. Foschini, "Layered space-time architecture for wireless communication in a fading environment when using multi-element antennas," *Bell Labs Tech. J.*, pp. 41–59, 1996.

[5] G. Foschini, G. Golden, R. Valenzuela, and P. Wolniansky, "Simplified processing for high spectral efficiency wireless communication employing multi-element arrays," *IEEE J. Sel. Areas Commun.*, vol. 17, no. 11, pp. 1841–1852, Nov. 1999.

[6] D. Rabideau, "Fast, rank adaptive subspace tracking and applications," *IEEE Trans. Signal Process.*, vol. 44, no. 9, pp. 2229–2244, Sep. 1996.

[7] J. Laurila, K. Kopsa, R. Schurhuber, and E. Bonek, "Semi-blind separation and detection of co-channel signals," in *Proc. Int. Conf. Communications*, Jun. 1999, vol. 1, pp. 17–22.

[8] A. Poon, D. Tse, and R. W. Brodersen, "An adaptive multiple-antenna transceiver for slowly flat-fading channels," *IEEE Trans. Commun.*, vol. 51, no. 11, pp. 1820–1827, Nov. 2003.

[9] C. V. Ramamoorthy, J. R. Goodman, and K. H. Kim, "Some properties of iterative square-rooting methods using high-speed multiplication," *IEEE Trans. Comput.*, vol. C-21, no. 8, pp. 837–847, 1972.

[10] D. Marković, B. Nikolić, and R. W. Brodersen, "Power and area efficient VLSI architectures for communication signal processing," in *Proc. Int. Conf. Communications*, Jun. 2006.

[11] D. Marković, V. Stojanović, B. Nikolić, M. A. Horowitz, and R. W. Brodersen, "Methods for true energy-performance optimization," *IEEE J. Solid-State Circuits*, vol. 39, pp. 1282–1293, Aug. 2004.

[12] V. Zyuban *et al.*, "Integrated analysis of power and performance for pipelined microprocessors," *IEEE Trans. Comput.*, vol. 53, no. 8, pp. 1004–1016, Aug. 2004.

[13] V. Srinivasan *et al.*, "Optimizing pipelines for power and performance," in *Proc. IEEE/ACM Int. Symp. Microarchitecture*, Nov. 2002, pp. 333–344.

[14] SystemC. [Online]. Available: http://systemc.org

[15] W. R. Davis *et al.*, "A design environment for high throughput, low power dedicated signal processing systems," *IEEE J. Solid-State Circuits*, vol. 37, no. 3, pp. 420–431, Mar. 2002.

[16] C. Chang, K. Kuusilinna, B. Richards, A. Chen, N. Chan, and R. W. Brodersen, "Rapid design and analysis of communication systems using the BEE hardware emulation environment," in *Proc. IEEE Rapid System Prototyping Workshop*, Jun. 2003.

[17] T. Gemmeke, M. Gansen, H. J. Stockmanns, and T. G. Noll, "Design optimization of low-power high-performance DSP building blocks," *IEEE J. Solid-State Circuits*, vol. 39, no. 7, pp. 1131–1139, Jul. 2004.

[18] C. Shi and R. W. Brodersen, "Automated fixed-point data-type optimization tool for signal processing and communication systems," in *Proc. IEEE Design Automation Conf.*, Jun. 2004, pp. 478–483.

[19] C. Shi and R. W. Brodersen, "A perturbation theory on statistical quantization effects in fixed-point DSP with non-stationary inputs," in *Proc. Int. Symp. Circuits and Systems (ISCAS 2004)*, May 2004, pp. 373–376.

[20] C. Shi, "Floating-point to fixed-point conversion," Ph.D. dissertation, Univ. California, Berkeley, 2004.

[21] A. P. Chandrakasan, S. Sheng, and R. W. Brodersen, "Low-power CMOS digital design," *IEEE J. Solid-State Circuits*, vol. 27, no. 4, pp. 473–484, Apr. 1992.

[22] J. Rabaey, A. Chandrakasan, and B. Nikolić, *Digital Integrated Circuits: A Design Perspective*, 2nd ed. Upper Saddle River, NJ: Prentice-Hall, 2003.

[23] K. K. Parhi, *VLSI Digital Signal Processing Systems*. New York: Wiley, 1999.

[24] V. Srinivasan and R. Vemuri, "A retiming based relaxation heuristic for resource-constrained loop pipelining," in *Proc. 11th Int. Conf. VLSI Design: VLSI for Signal Processing*, Jan. 1998, vol. 2, pp. 435–441.

[25] Y. Yi, R. Woods, L. K. Ting, and C. F. N. Cowna, "High sampling rate retimed DLMS filter implementation in Virtex-II FPGA," in *IEEE Workshop on Signal Processing Systems*, Oct. 2002, pp. 139–145.

[26] D. Marković, "A power/area optimal approach to VLSI signal processing," Ph.D. dissertation, Univ. California, Berkeley, 2006.

[27] D. Marković, R. W. Brodersen, and B. Nikolić, "A 70 GOPS, 34 mW multi-carrier MIMO chip in 3.5 mm^2," in *Proc. Int. Symp. VLSI Circuits*, Jun. 2006, pp. 196–197.

[28] K. Usami *et al.*, "Automated low-power technique exploiting multiple supply voltages applied to a media processor," *IEEE J. Solid-State Circuits*, vol. 33, no. 3, pp. 463–472, Mar. 1998.

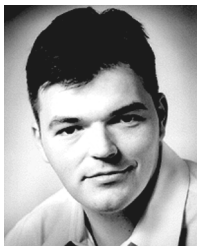
[29] BEE2: Berkeley Emulation Engine 2. [Online]. Available: http://bwrc.eecs.berkeley.edu/Research/BEE/BEE2/index.htm

[30] R. Brodersen, "Technology, architecture, and applications," presented at the IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers, Special Topic Evening Session: Low Voltage Design for Portable Systems, Feb. 2002.
[31] T. A. C. M. Claasen, "High speed: Not the only way to exploit the intrinsic computational power of silicon," in *IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers*, Feb. 1999, pp. 22–25.
[32] S. Santhanam *et al.*, "A low-cost 300 MHz RISC CPU with attached media processor," in *IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers*, Feb. 1998, pp. 298–299.
[33] J. Williams *et al.*, "A 3.2 GOPS multiprocessor DSP for communication applications," in *IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers*, Feb. 2000, pp. 70–71.
[34] T. Ikenaga and T. Ogura, "A fully-parallel 1Mb CAM LSI for real-time pixel-parallel image processing," in *IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers*, Feb. 1999, pp. 264–265.
[35] M. Wosnitza, M. Cavadini, M. Thaler, and G. Troster, "A high precision 1024-point FFT processor for 2-D convolution," in *IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers*, Feb. 1998, pp. 118–119.
[36] F. Arakawa *et al.*, "An embedded processor core for consumer appliances with 2.8 GFLOPS and 36 M polygons/s FPU," in *IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers*, Feb. 2004, pp. 334–335.
[37] M. Strik *et al.*, "Heterogeneous multi-processor for the management of real-time video and graphics streams," in *IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers*, Feb. 2000, pp. 244–245.
[38] H. Igura *et al.*, "An 800 MOPS 110 mW 1.5 V parallel DSP for mobile multimedia processing," in *IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers*, Feb. 1998, pp. 292–293.
[39] P. Mosch *et al.*, "A 720 μW 50 MOPs 1 V DSP for a hearing aid chip set," in *IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers*, Feb. 2000, pp. 238–239.
[40] A. Burg *et al.*, "VLSI implementation of MIMO detection using the sphere decoding algorithm," *IEEE J. Solid-State Circuits*, vol. 40, no. 7, pp. 1566–1577, Jul. 2005.
[41] D. Garrett, G. Woodward, L. Davis, G. Knagge, and C. Nicol, "A 28.8 Mb/s 4×4 MIMO 3G high speed downlink packet access receiver with normalized least mean square equalization," in *IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers*, Feb. 2004, pp. 420–421.
[42] D. Garrett, G. Woodward, L. Davis, and C. Nicol, "A 28.8 Mb/s 4×4 MIMO 3G CDMA receiver for frequency selective channels," *IEEE J. Solid-State Circuits*, vol. 40, no. 1, pp. 320–330, Jan. 2005.
[43] Z. Guo and P. Nilsson, "A VLSI architecture for the schnorr-euchner decoder for MIMO systems," in *Proc. IEEE CAS Symp. Engineering Technologies: Frontiers of Mobile and Wireless Communications*, Jun. 2004, pp. 65–68.
[44] Z. Guo, "MIMO decoding, algorithm and implementation," Ph.D. dissertation, Lund Univ., Lund, Sweden, 2005.
[45] Z. Guo and P. Nilsson, "A VLSI implementation of MIMO detection for future wireless communications," in *Proc. IEEE Symp. Personal, Indoor, and Mobile Radio Commun.*, Sep. 2003, pp. 2852–2856.

Dejan Marković (S'96–M'06) received the Dipl.Ing. degree from the University of Belgrade, Yugoslavia, in 1998, and the M.S. and Ph.D. degrees from the University of California, Berkeley, in 2000 and 2006, respectively, all in electrical engineering.

In 2006, he joined the faculty of the Electrical Engineering Department at the University of California, Los Angeles, as an Assistant Professor. His research is focused on rapid prototyping and optimization of power limited digital systems, with emphasis on the next generation wireless communication devices.

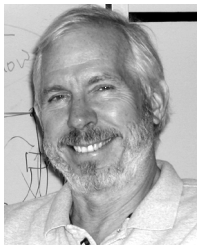
Dr. Markovic was awarded the CalVIEW Fellow Award in 2001 and 2002 for excellence in teaching and mentoring of industry engineers through the UC Berkeley distance learning program. In 2004, he was a co-recipient of the Best Paper Award at the IEEE International Symposium on Quality Electronic Design.



Borivoje Nikolić (S'93–M'99–SM'06) 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, where he is now an Associate Professor. His research activities include high-speed and low-power digital integrated circuits and VLSI implementation of communications and signal processing algorithms. He is coauthor of the book *Digital Integrated Circuits: A Design Perspective*, 2nd ed. (Prentice-Hall, 2003).

Dr. Nikolic received the IBM Faculty Partnership Award in 2005, NSF CAREER award in 2003, 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. For work with his students and colleagues he received the Best Paper Award at the ACM/IEEE International Symposium of Low-Power Electronics in 2005, and the 2004 Jack Kilby Award for the Outstanding Student Paper at the IEEE International Solid-State Circuits Conference.



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 and Co-Scientific Director of the Berkeley Wireless Research Center. 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. In 2001 he was awarded the Lewis Winner Award for outstanding paper at the IEEE International Solid-State Circuits Conference and in 2003 was given an award for being one of the top ten contributors over the 50 years of that conference.