

# The Cost of Flexibility in Systems on a Chip Design for Signal Processing Applications

Ning Zhang  
Atheros Communications, Inc.  
and  
Robert W. Brodersen  
Berkeley Wireless Research Center  
University of California, Berkeley

## Abstract

Providing flexibility into a system on a chip design is sometimes required and generally always desirable. However the cost of providing this flexibility in terms of energy consumption and silicon area is not well understood. This cost can range over many orders of magnitude depending on the architecture and implementation strategy. To quantify this cost, efficiency metrics are introduced for energy (MOPS/mW) and area (MOPS/mm<sup>2</sup>) and are used to compare a variety of designs and architectures for signal processing applications. It is found that the critical architectural parameters are the amount of flexibility, the granularity of the architecture in providing this flexibility and the amount of parallelism. A range of architectural solutions which tradeoff these parameters are presented and applied to example applications.

## I. Introduction

The tradeoff of various types of architectures to implement digital signal processing (DSP) algorithms has been a subject of investigation since the initial development of the theory [1]. Recently, however the application of these algorithms to systems that require low cost and the lowest possible energy consumption has placed a new emphasis on defining the most appropriate solutions. For example, advanced communication algorithms which exploit frequency and spatial diversities to combat wireless channel impairments and cancel multi-user interference for higher spectral efficiency (data rate / bandwidth) have extremely high computational complexity (10's-100's of billions of operations/second), and thus require the highest possible level of optimization.

In order to compare various architectures for these complex applications it is necessary to define design metrics which capture the most critical characteristics relating to energy and area efficiency as a function of the flexibility of the implementation. Flexibility in implementing various applications after the hardware has been fabricated is a desirable feature, and as will be shown it is the most critical criteria in determining the energy and area efficiency of the design. However, it is found that there is a range of multiple orders of magnitude differences in these efficiencies depending on the architecture used to provide various levels of flexibility. Therefore, it is important to understand the cost of flexibility and choose an architecture which provides the required amount at the highest possible efficiency.

Consequently, the flexibility consideration becomes a new dimension in the algorithm/architecture co-design space. Often the approach to flexibility has been to provide an unlimited amount through software programmability on a Von Neumann architecture. This approach was based on hardware technology assumptions which assumed hardware was expensive and the power consumption was not critical so that time multiplexing was employed to provide maximum sharing of the hardware resources. The situation now for highly integrated “system-on-a chip” implementations is fundamentally different: hardware is cheap with potentially 1000’s of multipliers and ALU’s on a chip and the energy consumption is a critical design constraint in many portable applications. Even in the case of applications that have an unlimited energy source, we are now beginning to move into an era of power constrained performance for the highest performance processors since heat removal requires the processor to operate at lower clock rates than dictated by the logic delays.

The importance of architecture design is further underscored by the ever and faster increasing algorithm complexity, and purely relying on technology scaling many DSP architectures will fall short of computational capability for more advanced algorithms demanded by future systems. An efficient and flexible implementation of high-performance digital signal processing algorithms therefore relies on architecture optimization. Unfortunately, the lack of a systematic design approach and consistent metrics currently prevents the exploration of various realizations over a broad range of

architectural options. The focus of this paper is to investigate the issues of flexibility and architecture design together with system and algorithm design and technology to get a better understanding of the key trade-offs and the costs of important architecture parameters and thus more insight in digital signal processing architecture optimization for high-performance and energy-sensitive portable applications.

Section II introduces the design metrics used for architecture evaluation and comparison with an analysis of the relationship between key architecture parameters and design metrics. Section III uses an example to compare two extreme architectures: software programmable and dedicated hardware implementation, which result in orders of magnitude difference in design metrics. The architectural factors that contribute to the difference are also identified and quantified. Section IV focus on intermediate architectures and introduces an architecture approach to provide function-specific flexibility with high efficiency. This approach is demonstrated through a design example and the result is compared to other architectures.

## **II. Architectural Design Space and Design Metrics**

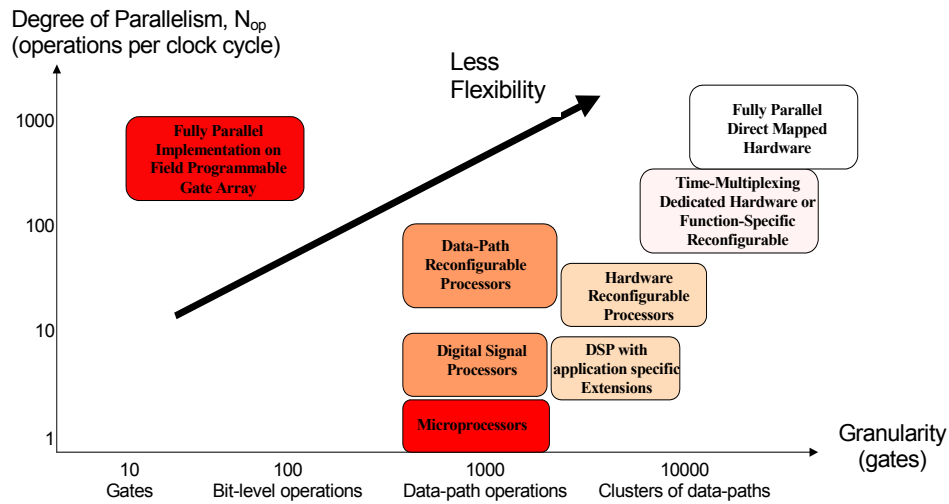
### ***A. Architecture Space***

The architectures being considered range from completely flexible software based processors including both general purpose as well as those optimized for digital signal processing (DSP's), to inflexible designs using hardware dedicated to a single application. In addition architectures will be investigated which lie between these extremes.

Flexibility is achieved in a given architecture either through reconfiguration of the interconnect between computational blocks (e.g. FPGA or reconfigurable datapaths) or by designing computational blocks that has logic that can be programmed by control signals to perform different functions (e.g. microprocessor datapath). The granularity is the size of these computational blocks and can range from up to single or multiple datapaths as shown in Figure 1. In general, the larger the granularity, the less the flexibility an architecture can provide since the interconnection within the processing unit is pre-defined at fabrication time.

Digital signal processing systems typically have throughput requirements to meet hard real-time requirements. Throughput will be defined as the product of the average number of operations per clock cycle,  $N_{op}$ , and the clock frequency,  $f_{clk}$ . The key architectural feature to provide throughput is the degree of parallelism, which is measured by  $N_{op}$ . The more parallel hardware in an architecture, the more operations can be executed in a clock cycle, which implies a lower level of time-multiplexing and thus lower clock frequency necessary to meet a given throughput requirement.

Therefore there are two key architectural parameters: the granularity of data-path unit and the degree of parallelism. Figure 1 depicts qualitatively the relative positions of different architectures in this two-dimensional space. Due to the energy and area overheads associated with small granularity and time multiplexing architectures, this large design space provides a wide range of trade-offs between flexibility, energy efficiency and area efficiency.



**Figure 1: The architectural space covering granularity of data-path unit and the degree of parallelism**

## ***B. Design Metrics***

Design metrics are defined to be able to compare different architectures. Energy efficiency and cost are two of the most stringent requirements of portable systems. The basic parameters of importance are performance, power (or energy) and area (or cost). All of these parameters can be traded off, so to reduce the degrees of freedom we will

compare the power and area required for various architectural solutions for a given level of throughput.

Throughput will be characterized, as the number of operations per second as MOPS (Millions of operations per second). For signal processing systems, to make fair comparison across different architecture families, an operation is often defined as algorithmically interesting computation. For a specific application where the operation mix is known, a basic operation is further defined as a 16-bit addition to normalize all other operations (shift, multiplication, data access, etc.).

While this approach works for signal processing computation, it is not as appropriate for applications implemented on general-purpose microprocessors. For comparison with these architectures we will define operation as being equivalent to an instruction, in spite of the fact that a number of instructions may be required to implement an algorithmically interesting operation. This will provide a conservative over estimate of the performance achievable from general-purpose processors, and thus will just further substantiate the observations about relative architectural efficiency that will be made in later sections.

### **Energy Efficiency Metric**

The streaming nature of signal processing applications and the associated hard real-time constraints usually makes it possible to define a basic block of repetitive computation. This makes it possible to define an average number of operations per clock cycle,  $N_{op}$  with the average energy required to implement these operations being given by  $E_{op}$ . A useful energy efficiency metric is therefore the average number of operations per unit energy,  $\epsilon_E$ , which can be defined as the ratio of  $N_{op}$  to  $E_{op}$ :

$$\epsilon_E = \text{Number of operations/Energy required} = N_{op} / E_{op} \quad (1)$$

The architectural exploration problem is to determine the approach which has the highest energy efficiency, while meeting the real-time constraints of completing the signal processing kernel.

For the situation in which the processor is connected to an unlimited energy source there is a related problem in that the performance may be limited by the difficulty and cost of dissipating the generated heat. In fact this has become such an issue that the highest performance general-purpose processors are now operating in a power limited mode, in which the maximum clock rates are limited by heat dissipation rather than logic speed. The relevant metric for this case is the power required to sustain a given level of throughput or the throughput to power ratio. Using the above variables the average throughput or number of operations per second is given by  $N_{\text{ops}}/T_{\text{clk}}$  with the power required being given by  $E_{\text{ops}}/T_{\text{clk}}$ . The throughput to power ratio is then found to be the same as the energy efficiency metric,  $\epsilon_E$ , defined in Eq. 1,

$$\epsilon_E = \text{Throughput/Power} = (N_{\text{op}}/T_{\text{clk}}) / (E_{\text{op}}/T_{\text{clk}}) = N_{\text{op}} / E_{\text{op}} \quad (2)$$

If the throughput is expressed in MOPS and power in milliWatts, the efficiency,  $\epsilon_E$ , will have units of MOPS/mW and it is these units that we will use when presenting the efficiency comparisons.

### **Area Efficiency Metric**

While the  $\epsilon_E$  metric is relevant for both energy and power limited operation, it does not incorporate the other major constraint, which is the cost of the design as expressed by the area of the silicon required to sustain a given level of throughput. The relevant metric for this consideration is therefore the throughput to area ratio given by,

$$\epsilon_A = (N_{\text{ops}}/T_{\text{clk}}) / A_{\text{op}} \quad , \quad (3)$$

where  $A_{\text{op}}$  is the integrated circuit area required for the logic and memory to support the computation of  $N_{\text{op}}$  operations. If the units for area are chosen to be square millimeters and throughput again is expressed in MOPS,  $\epsilon_A$  has units of MOPS/mm<sup>2</sup>, which measures computation per unit area. It is a well-known low-power design technique [2] to trade-

off area efficiency for energy efficiency, thus these two metrics need to be considered together.

### Metric Comparisons

In fixed function architectures, the operation count is straightforward, which is not the case in comparisons with processors that are flexible. In this case depending on the benchmark different possible throughputs can be achieved. When making comparisons of architectures for different applications we will use the highest possible throughput numbers that can be achieved in a given architecture. In sections III and IV we will compare various architectures for the same application.

An estimate can be made of the maximum achievable energy and area efficiencies for the basic 16 bit add operation for a given technology and clock rate if we assume that we can fill an entire integrated circuit with 16 bit adders (our basic operation) and achieve a throughput corresponding to every adder operating in parallel. In Table I, the energy and area values are given along with the associated calculated energy and area efficiencies for .25 micron technology for two different supply voltages. The minimum clock period is assumed to be taken somewhat arbitrarily as approximately 2 times the delay through an adder, which is consistent with the example circuits presented below. As can be seen from equations (2) and (3) the area efficiency scales directly with the clock rate, while the energy efficiency is independent of it if the voltage is not changed.

Supply Voltage	Adder Area	Adder Energy	Logic Delay	$\epsilon_E$ (MOPS/mW) Basic Operation	$\epsilon_A$ (MOPS/mm <sup>2</sup> ) Basic Operation
1.0 Volt ( $f_{clk} = 35$ MHz)	.006 mm <sup>2</sup>	.53 pJ	14 nS	1900	5800
2.5 Volts ( $f_{clk} = 120$ MHz)	.006 mm <sup>2</sup>	3.1 pJ	4.3 nS	300	20000

**Table I: The energy and area efficiency of the basic operation (a 16 bit add) in .25 micron technology**

For a given application, the basic operation rate can be obtained by profiling the algorithm and it can be used together with the energy and area efficiencies of a basic

operation to determine the lower bound of average power consumption of an algorithm. This lower bound of power and area for fixed throughput, or higher bound of efficiencies, not only provides an implementation feasibility measure to algorithm developers, but also provides a comparison base to evaluate any implementation.

### Comparison of Architectures assuming Maximum Throughput

Figure 2 shows the efficiency metrics for a number of chips chosen from the International Solid State Circuits Conference from 1998-2002 under the criteria that they were in a technology that ranged from .18 - .25 micron and that all the information was available to do a first order technology scaling and to calculate the energy and area efficiencies. Though this is relatively small sample of circuits it is believed that the trends and relative relationships are accurate representations of the various architectures being compared because of the remarkable consistency of the results. Table II gives a summary of all the circuits that were used in the comparison.

CHIP #	YEAR	PAPER #	DESCRIPTION	CHIP #	YEAR	PAPER #	DESCRIPTION
1	1997	10.3	$\mu$ P - S/390	11	1998	18.1	DSP -Graphics
2	2000	5.2	$\mu$ P – PPC (SOI)	12	1998	18.2	DSP - Multimedia
3	1999	5.2	$\mu$ P - G5	13	2000	14.6	DSP – Multimedia
4	2000	5.6	$\mu$ P - G6	15	2002	22.1	DSP -MPEG Decoder
5	2000	5.1	$\mu$ P - Alpha	14	1998	18.3	DSP – Multimedia
6	1998	15.4	$\mu$ P - P6	16	2001	21.2	Encryption Processor
7	1998	18.4	$\mu$ P - Alpha	17	2000	14.5	Hearing Aid Processor
8	1999	5.6	$\mu$ P – PPC	18	2000	4.7	FIR for Disk Read Head
9	1998	18.6	$\mu$ P - StrongArm	19	1998	2.1	MPEG Encoder
10	2000	4.2	DSP – Comm	20	2002	7.2	802.11a Baseband

**Table II: Description of chips used in the analysis from the International Solid State Circuits Conference**



In the table and in the figures the designs were sorted according to their energy efficiency and very surprisingly this sorting also resulted in their being grouped into the three basic architectural categories. Chips 1-9 were general purpose microprocessors, with chips 10-15 optimized for DSP but still software programmable and chips 16-20 being dedicated signal processing designs with only a very limited amount of flexibility in comparison to the full flexibility of the microprocessors and DSP's.

As seen in Figure 2, the energy efficiency varies by 1-2 orders of magnitude between each group with an overall range of 4 orders of magnitude between the most flexible solutions and the most dedicated. It is not surprising that the efficiency decreases as the flexibility is increased, but it is the enormous amount of this cost that should be noted. The area efficiencies have a similar range, with the exception of a few designs that had low clock rates set by the application for which they were designed. As mentioned previously the energy efficiency is independent of clock rate whereas the area efficiency is directly proportional to it. It should be re-emphasized that the strategy for counting operations was to use the maximum possible throughput for the software processor solutions, while the dedicated designs used the actual useful operations required for the application. In a later section a comparison will be made for the same application which allows a more consistent comparison of operation counts.

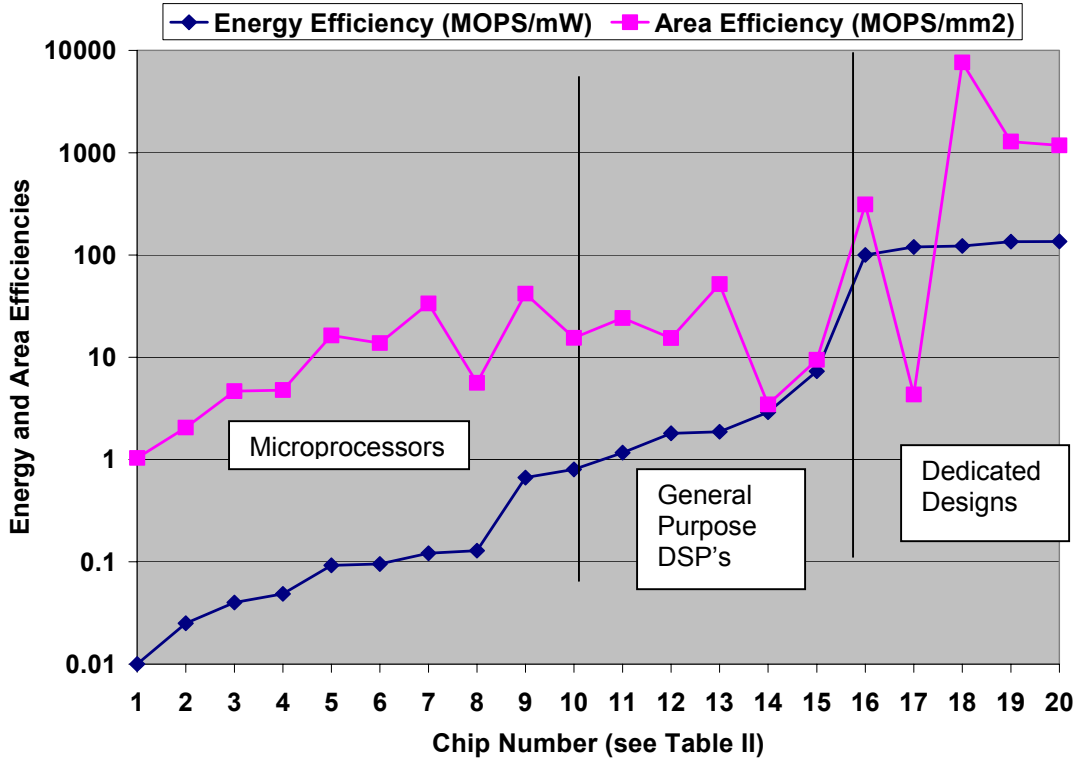


Figure 2: Energy and area efficiency of different architectures

Fi

Due to the lack of reference designs, FPGA's are not included in Figure 2. Studies have shown that FPGA's are about 2 orders magnitude less energy efficient and about 3 orders of magnitude less area efficient than dedicated designs. An example will be given in Section IV. Although FPGA designs have a high degree of parallelism as the dedicated hardware, the inefficiency comes from the fine granularity of the functional units. The basic functional unit in a FPGA is a bit-processing element, or Configurable Logic Block (CLB). The area and energy overhead of the interconnect network due to the fine granularity is substantial. For example, one study [12] showed 65% of the total energy in a Xilinx XC4003A FPGA is due to the wires, and another 21% and 9% are taken by clocks and I/O. The CLB's are responsible for only 5% of the total energy consumption.

### C. Relationship Between Architecture and Design Metrics

To attempt to understand the enormous energy efficiency gaps between the three architecture families, we will break down the energy efficiency metric into three components. To do this we start with the average power,  $P_{ave}$ , as:

$$P_{ave} = A_{chip} * C_{sw} * V_{dd}^2 * f_{clk} \quad (4)$$

where  $A_{chip}$  is the total area of the chip,  $V_{dd}$  is the supply voltage,  $f_{clk}$  is the clock rate and  $C_{sw}$  is the average switched capacitance per unit area that includes both the transition activity as well as the capacitance being switched averaged over the entire chip. It can be found since all the other variables in Eq. 4 are available from the data. Substituting  $P_{ave}$  into Eq. 2 we find,

$$\epsilon_E = \text{Throughput/Power} = (N_{ops}/T_{clk}) / P_{ave} = 1/(A_{op} * C_{sw} * V_{dd}^2) \quad (5)$$

where  $A_{op}$  is the average area of each operation per cycle found from the total area of the chip divided by the number of operations per clock cycle, i.e.  $A_{op} = A_{chip}/N_{op}$ .

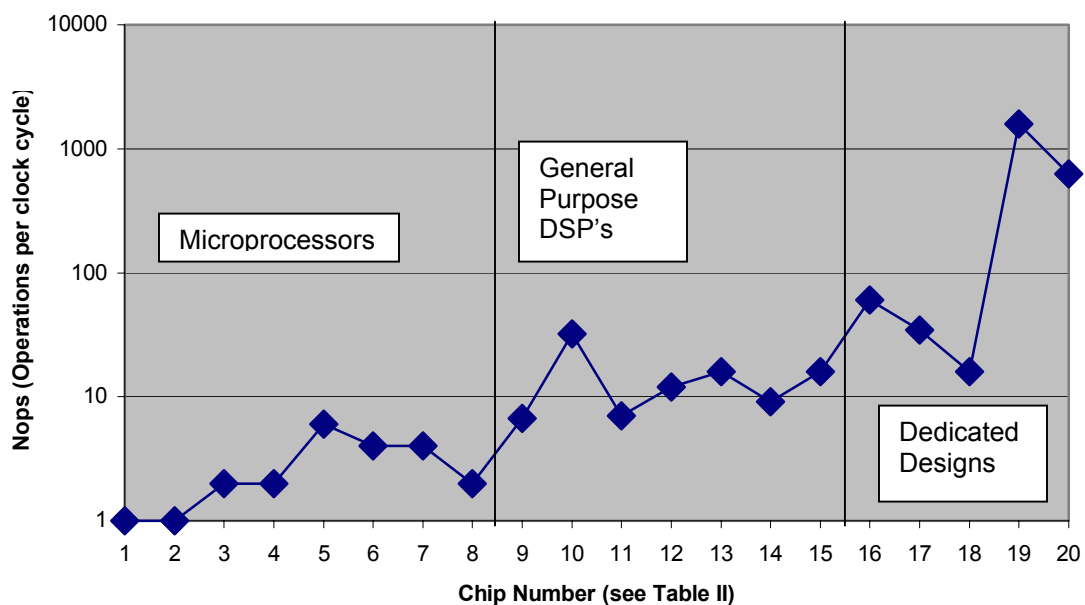
		MICROPROCESSOR	DSP	DEDICATED
$F_{clk}$	Range Average	450-1000 MHz 550 MHz	50-300 MHz 170 MHz	2.5-275 MHz 90 MHz
$C_{sw}$	Range Average	40-100 pF/mm <sup>2</sup> 70 pF/mm <sup>2</sup>	10-60 pF/mm <sup>2</sup> 30 pF/mm <sup>2</sup>	10-45 pF/mm <sup>2</sup> 20 pF/mm <sup>2</sup>
$N_{op}$	Range Average	1-6 op/clock 2 op/clock	7-32 op/clock 14 op/clock	460 op/clock 16-1580 op/clock

**Table III: The average and ranges of design parameters for the 3 architectural groups**

As seen in Table III the value of the switched capacitance doesn't vary widely across the design examples. However, the fact that it is larger for the software based processors (the microprocessor and DSP columns) and not substantially lower than the dedicated designs is somewhat counter-intuitive since these processors contain a significant amount of circuitry (particularly in the memories) that is not switching on every clock cycle. It is the opposite case in the dedicated designs, which typically have a very large percentage of their logic switching, since the designs are optimized to fully utilize as much hardware as possible.

The primary reason for the high  $C_{sw}$  in the processors is the elevated clock rates (average of 550 MHz) compared to what is required for the more parallel dedicated chips (90 MHz average). The high clock rates require more capacitance in the clocking networks as well

as in the high density custom designed logic and memory which is necessary to achieve low values of delay. The high clock rates come from the relatively low level of parallelism in the software processors which then necessitates high levels of time multiplexing (e.g. higher clock rates) to achieve the necessary throughput. This is clearly demonstrated by the range of values for the parallelism parameter,  $N_{op}$ , the number of operations per clock cycle, as seen in Figure 3. It is seen that the level of parallelism in dedicated designs ranges up to 3 orders of magnitude higher than that achieved by some software processors.



**Figure 3: The amount of parallelism as determined from the number of clock cycles per operation**

The fundamental reason that the microprocessors have a low level of parallelism is the difficulty of exploiting a parallel architecture when the specification is inherently sequential (e.g. in C). As seen in Figure 3 the DSP software processors do achieve more parallelism, but at a considerable cost in increased programming complexity. In fact because of the difficulty of compiling for the parallel DSP architectures from a high level sequential description, the programming of the critical computation kernels is typically manually performed at the assembly level.

The supply voltage also doesn't vary widely across all of the designs, ranging from 1.5 V to 2.5 V, with no significant difference for any of the architecture groups. It is therefore the variation in  $A_{op}$ , the area per operation which is plotted in Figure 4, that contains the explanation for the wide variation in efficiencies. It can be seen from this figure that it ranges from 100's of  $mm^2$  for the microprocessors to less than  $.1 mm^2$  for some of the dedicated designs. It should be noted that while  $.1 mm^2$  seems quite small, it still is more than 10 times the size of a 16 bit add (see Table I). The primary reason for the large values of  $A_{op}$  is that the relatively small computational blocks and datapaths are surrounded by the large control and memory circuits that are used to provide full flexibility and high degree of time-multiplexing.

To see the problem we look at the two designs which have a single time-multiplexed data path (chips1 & 2), so that  $N_{op} = 1$  as seen in Figure 3. To achieve the average throughput of the dedicated chips with this lack of parallelism, the datapath on these chips must be time multiplexed approximately 460 times in the period of one clock of the dedicated design (see Table III). Also, since full flexibility is required the datapath needs to be fully reconfigured 460 times by control circuitry. The breakdown of the factors due to this time multiplexing, which result in such an enormous level of overhead in  $A_{op}$ , will be given in Section III.

The range in  $A_{op}$  directly relates to  $E_{op}$  or the energy required per operation. Since as noted earlier the capacitance being switched,  $C_{sw}$ , per unit area is larger for the high clock rate architectures, the energy per operation  $C_{sw} * A_{op} * V_{dd}^2$  will have an even larger range as seen in Figure 2.

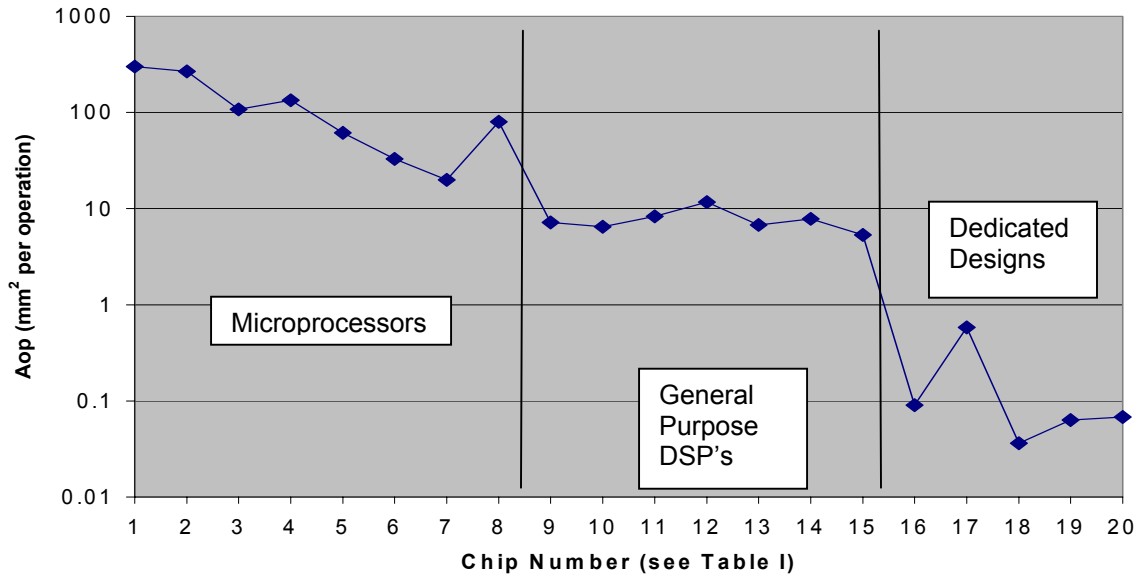


Figure 4: The silicon area per operation

### III. Software Implementation vs. Hardware Implementation for the Same Application

The previous section compared a variety of architectures under the condition of maximum possible throughput. This section will use a single application to give further insight into the reasons for the wide variation in efficiency.

Four multi-user detection schemes were evaluated in [3], that can be used in a wideband synchronous DS-CDMA system. The blind adaptive MMSE multi-user detector, which was one of those algorithms analyzed, will be used in this section since it gives the best trade-off between system performance and computational complexity. This algorithm requires a word-length of 12-bits, has 13000 MOPS (an operation is defined as a basic operation of 12-bit add), and at a clock rate of 25 MHz has a lower bound of power consumption of 5 mW.

#### 1). Digital Signal Processor

Data from one of the lowest-power DSP's reported in a .25  $\mu\text{m}$  technology [4] was chosen for analyzing the efficiencies of a software implementation. It is a 1 V, 63 MHz, 16-bit fixed-point DSP designed specifically for energy-efficient operation. It has a single cycle multiply accumulator and an enhanced Harvard architecture. Its on-chip memory includes a 6K x 16b SRAM and 48K x 16b ROM.

C code describing the algorithms was compiled using performance optimizations because it was found that the shortest sequence typically consumes the lowest energy. This guideline is further justified by the results of instruction level analysis [5, 6], where it was found that the variation of energy consumption for any instruction is relatively small. The explanation for the observation lies in the fact that there is a large underlying energy dissipation common to the execution of any instruction, independent of the functionality of the instruction. This is the energy associated with clock, instruction fetching and control, which means the overhead associated with instruction level control and centralized memory storage dominates the energy consumption. Performance and power consumption results are based on benchmarking compiled assembly code. Further investigation of the software implementation showed by optimizing the assembly code through better register usage and memory allocation, utilization of special parallel instructions, etc. an improvement of 35% in performance and energy-efficiency was achieved.

To execute the design example, the arithmetic instructions only take about 30% of the total execution time and power, memory accesses take about 30%, and data movement and control take the remainder 40%. A hardware component breakdown shows that 26% of total power goes to clocking, 18% to instruction fetching and control, 38% to memory, and 18% to arithmetic execution.

## **2). DSP Extension**

Improvement on software-based implementation can be made through supporting application-specific instructions and adding accelerating co-processors. For the design example, it was observed that the algorithm operates on complex values, one way to improve efficiency is to modify the data-path to directly operate and store complex numbers. Also, this algorithm does not need all the memory provided on the DSP, two 256 x 32 bit SRAM banks for data memory and a 256 x 16 bit SRAM for instruction memory are sufficient (this memory structure already doubles the actual requirements of this algorithm).

Potential energy saving comes from reduction of instruction fetching and decoding, reduction of memory access count and large memory overhead, and optimization of arithmetic execution units. The main idea behind this approach is that due to the large power overhead associated with each instruction, packing instructions or adding more complicated execution units reduces the relative overhead. For example, for a complex multiplication

$$Y_{\text{real}} = A_{\text{real}} * B_{\text{real}} - A_{\text{imag}} * B_{\text{imag}}; Y_{\text{imag}} = A_{\text{real}} * B_{\text{imag}} + A_{\text{imag}} * B_{\text{real}};$$

this architecture only needs to read A and B from memory once instead of twice as in the original DSP architecture.

This architecture increases the granularity of data-path unit to reduce the energy overhead and cuts unnecessary memory to reduce the area overhead. As the result, this architecture is less flexible since it is optimized only for computations with complex inputs and applications with limited memory requirements.

### **3). Direct-Mapped Dedicated Hardware**

A direct-mapped implementation maps each function in the data flow graph of the algorithm into a dedicated hardware block. Figure 5 depicts the block diagram of the example algorithm and the corresponding layout view of its dedicated implementation. The architecture is directly mapped from the algorithm with the maximum possible level of parallelism which allowed 1 V operation at clock rate of 25 MHz. The fixed point word lengths were optimized throughout the architecture with 12-bit word length being the most common.



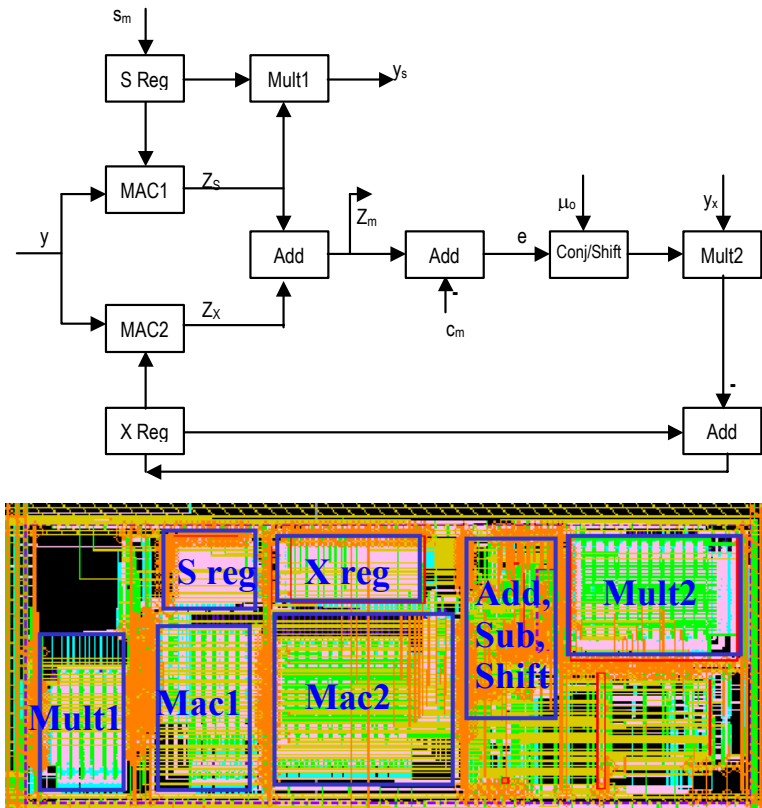


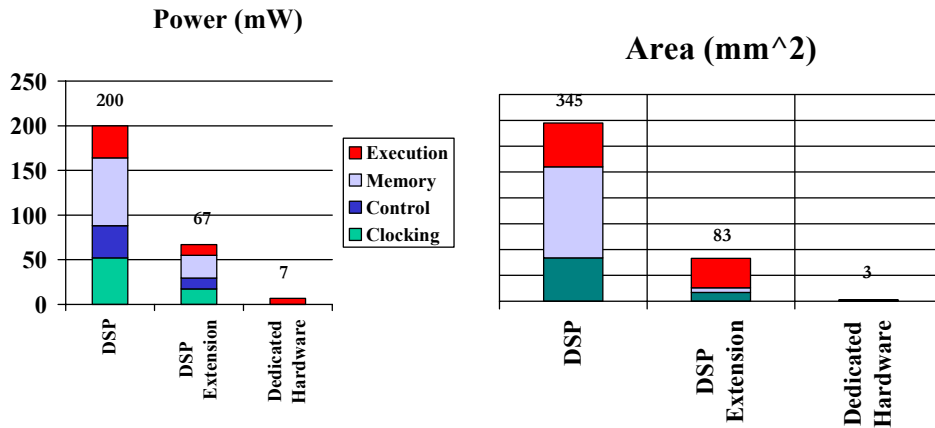
Figure 5: Direct mapped implementation of a multi-user detection algorithm

#### 4). Architecture Comparison and Analysis

The comparisons of architectural implementations are shown in Table IV and the breakdowns are depicted in Figure 6. For the DSP implementations, since one processor cannot meet the throughput requirement, parallel processors are assumed and the overhead of splitting the algorithm into a number of processors is neglected for the upper efficiency bound of the software solutions. Notice the DSP extension can achieve considerable savings in both power consumption and area by increasing the data-path granularity and reducing memory size. The dedicated hardware achieves the highest efficiency as expected, with the ratios of 30-50 times the DSP processor and 10 times the processor with the application specific extensions.

	Low-Power DSP		DSP Extension	Dedicated Hardware	Upper Bound (12-bit Basic Operation, $f_{clk}=25$ MHz)
	Compiled	Optimized			
Energy Efficiency (MOPS/mW)	43	65	190	1900	2500
Normalized Energy Efficiency	0.02	0.03	0.1	1	1.3
Area Efficiency (MOPS/mm <sup>2</sup> )	25	38	160	4300	5600
Normalized Area Efficiency	0.01	0.01	0.04	1	1.3

**Table IV: Architectural implementation comparisons of multi-user detection**



**Figure 6: Power consumption and area breakdown of different architectural implementations of multi-user detection**

The optimized DSP implementation requires at least 15 parallel processors to meet the throughput requirement. This shows that adding parallel processing elements without an architecture matched to the target computation is not sufficient. The parallel processor approach only avoids the energy overhead associated with a higher clock rate, without fundamentally changing the structure of small data-path granularity required to provide flexibility and resultant high degree of time-multiplexing to achieve throughput. As seen in Figure 2 DSP's have efficiencies 1-2 orders of magnitude below that of dedicated hardware, this difference comes from the following components:

a). Execution units: a factor of 6 is attributed to the overhead hardware required to build a fully programmable ALU and register bank in addition to the basic adder, shifter, and registers, which performs the required computation [7]. The dedicated hardware can also use the shortest possible word-length, such as a 12-bit data-path instead of 16-bit in this design example, and it can preserve data correlation. These application-specific low-power design techniques save another factor of 2 in energy consumption [8].

b). Instruction fetching and control: a factor of 5 in the energy overhead is due to hardware support for instruction fetches, branches, loads, stores, and other instruction-specific functionalities [7]. In the design example, control circuits consume about the same energy as execution units.

c). Memory access: it is a significant source of energy consumption in a DSP due to its highly time-multiplexed architecture, which requires a large amount of memory accesses per useful computation. In this design example, 40% of total energy consumption is from memory. Also, in order to provide flexibility, the memory is usually over-designed to accommodate different algorithms, which adds extra overhead since energy per memory access increases with memory size. It also causes the area overhead of large on-chip memories of a typical processor. In the example DSP, the CPU only occupies about 25% of the area. Consequently, memory accounts for another factor of 5-10 overhead in energy consumption for a DSP. For example, the energy efficiency of a FFT processor was improved by 16 times through memory structure optimization [9].

d). Other factors: In this design example, 30% of the energy consumption of the DSP is from a fast clocking scheme required by the time-multiplexing. Due to its large area, global busses and interconnects also contribute to the overall overhead.

## **IV. Intermediate Architectures**

One of the main problems with dedicated hardware design preventing it being more commonly used is the lack of flexibility. This section presents our method to attack the problem. Flexibility is a desirable feature in many portable systems. For example, one characteristic of wireless systems is diverse and evolving application requirements (e.g., data rate and bit error rate) and changing parameters of the available communication link (e.g., bandwidth and channel parameters), which directly impact the choices of system

design parameters. Consequently it is desirable for wireless receivers to adapt their operation rather than only be targeted for a fixed or worst-case scenario. These receivers will have to be flexible enough to accommodate various operation conditions and their computational requirements while simultaneously achieving the lowest possible power consumption required for portable devices.

As shown in Figure 1, flexibility in an implementation can be achieved at different granularities. An FPGA provides reconfigurability at bit level and typically uses a mesh network for the interconnect structure. For data-path intensive applications this approach often leads to an energy-inefficient solution since it implicitly carries an extensive energy overhead and its reconfiguration granularity is not matched to data-path, which is typically composed of bus-wide arithmetic modules, such as adders, shifters, registers.

On the other hand, DSP and data-path processors match the programming granularity of data-path intensive signal processing applications. DSP achieves programmability through a set of instructions that dynamically modify the behavior of otherwise statically connected modules. It relies on the shared busses for the transfer of control and data and time-shares a few data-path units (ALU's and multipliers).

A reconfigurable data-path processor is a more parallel architecture with arrays of data-path units, memories and word-wide interconnect busses. In order to map different algorithms, the data-path units need to be configurable for any operation and the interconnect network needs to support full connectivity. So again the overhead associated with configuration control and interconnect is considerable and moreover, since the mix of different components is fixed, it cannot optimally fit all algorithms. For example, the pre-chosen memory allocation might become the bottleneck for mapping certain algorithms.

By increasing the reconfiguration granularity to the function level and only providing the flexibility necessary to satisfy the system requirements, the associated reconfiguration overhead can be minimized – resulting in “function-specific reconfiguration”. This approach is based on the observation that signal processing applications typically have a

few dominating computational kernels with high regularity, which allows the algorithm to be decomposed to patterns of computation, memory access and interconnection. A pattern can be directly mapped to a function-specific component, which is a combination of dedicated data-path units and specially partitioned and accessed memory blocks connected by dedicated links. Control and reconfigurable interconnect overhead is reduced since they are not needed within the components. Basically, this approach starts with most efficient dedicated hardware and adds in flexibility support when necessary.

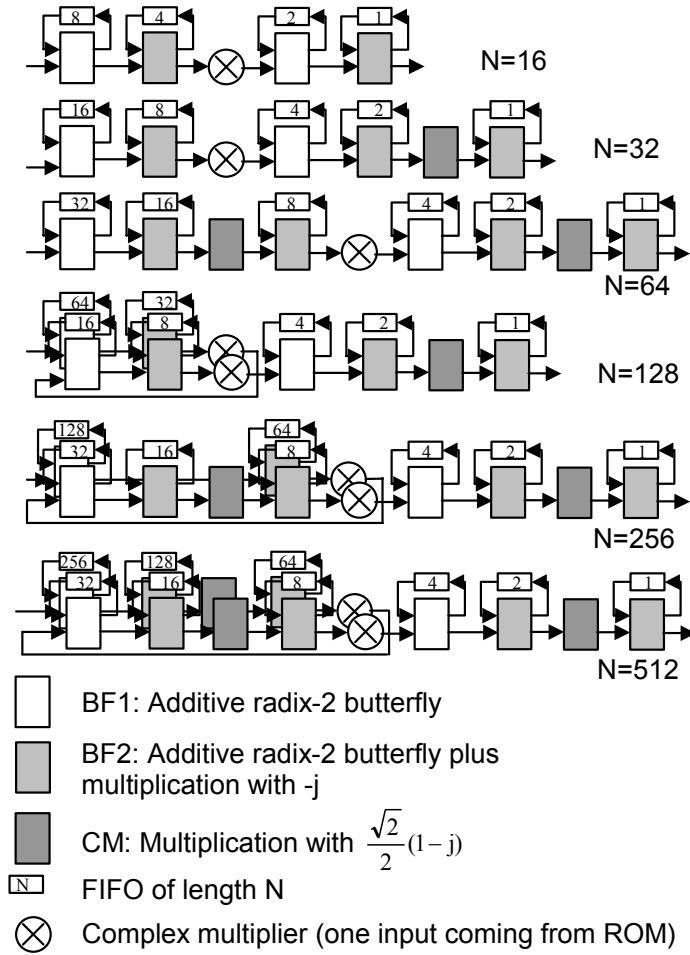
### ***Example: FFT***

A comparison was made of 5 different programmable or reconfigurable architectures, ranging from software programmable DSP's to function-specific reconfigurable hardware, to see the range of costs of flexibility. The results are based on FFT implementations on those architectures.

Xilinx's high-performance and high-density Virtex-E family was chosen as an example of FPGA architecture. The Xilinx CORE Generator system was used to generate optimized cores. Performance benchmarks were based on post-layout timing analysis. Two commonly used fixed-point DSP's from Texas Instruments were chosen for this comparison: the TMS320C6000 platform is an example of high-performance VLIW DSP, and the TMS320C5000 platform is optimized for power efficiency. Performance was evaluated based on optimized assembly programs and benchmarks published by the vendor. Chameleon Systems CS2000 family was chosen as a reconfigurable data-path processor. The performance and energy consumption results were based on Chameleon Systems preliminary measurements and estimates. In addition, the upper bound of energy efficiency for a given technology is used as a comparison base. In order to make fair and meaningful architectural comparisons, the results were scaled to a common technology (.18  $\mu\text{m}$ ). The scaling factors for delay and energy consumption are taken from circuit simulations.

A pipelined FFT architecture based on a single-path delay feedback structure is used for the function-specific reconfigurable implementation [10]. Flexibility in the length of the FFT, from 16 to 512 points, is achieved by changing the length of the shift registers and

simple muxing of the I/O blocks as shown in Figure 7. With a clock frequency at the input sample rate the entire range of FFT's can be accommodated by either directly mapping to hardware and disabling unneeded blocks for the shorter length FFT's or by folding the processing stages and time-sharing the hardware for the longer (but lower symbol rate) cases.



**Figure 7: Pipelined FFT schemes for N=16, ... 512**

The design has an area of 1.03 mm<sup>2</sup> in a .25 μm technology. Table V lists the maximum input sample rate supported for the entire range with a 1V supply voltage. (High performance wireless systems, such as IEEE 802.11a, have a basic symbol rate of about 20 MHz.) The word length is up to 12 bits, if less precision is needed, the lower side bits can be disabled to avoid wasteful switching activity.

	N=512	N=256	N=128	N=64	N=32	N=16
Max. input rate	25 MHz	25 MHz	25 MHz	50 MHz	50 MHz	50 MHz
Power at max. speed	5.1 mW	4.6 mW	4.0 mW	5.5 mW	4.9 mW	4.0 mW

**Table V: Implementation parameters of function-specific reconfigurable FFT**

The results of the architecture comparison are shown in Figure 8 and 9, which demonstrate the tremendous range of efficiencies for various architectural strategies. In Figure 8 the energy efficiency shows that the most efficient low-power DSP is more than 2 orders of magnitude less efficient than the upper bound, with the FPGA, reconfigurable data-path architecture and high-performance VLIW even more inefficient. On the other hand, the approach that uses function-specific reconfigurability is within a factor of 2-3 of the upper bound. Figure 9 presents the area efficiency. Again the reconfigurable data-path, the various programmable processors and the FPGA are more than two orders of magnitude less efficient than the function-specific reconfigurable approach. If a metric is used which combines both area and energy efficiencies, there is more than 4 orders of magnitude difference between the most efficient (function-specific reconfigurability) and the other common digital signal processing solutions.

The function-specific reconfigurable hardware achieves highest efficiency since it has the least amount of time-multiplexing and largest data-path granularity among all the designs. This example suggests that there is a considerable opportunity for adding sufficient flexibility to application-specific hardware while achieving the energy dissipation close to the bound provided by the technology. Several general observations can be made from the comparisons. For example, if we compare the data-path reconfigurable processor to high-performance VLIW DSP in Figure 8, the data-path processor is more parallel or less time-multiplexed with about the same granularity and thus it achieves higher energy efficiency. Also, Figure 9 shows the least area efficient architecture is FPGA due to its large overhead with fine granularity of computation. Similar results are obtained for other designs, such as a Viterbi decoder [10] and an encryption processor [11].

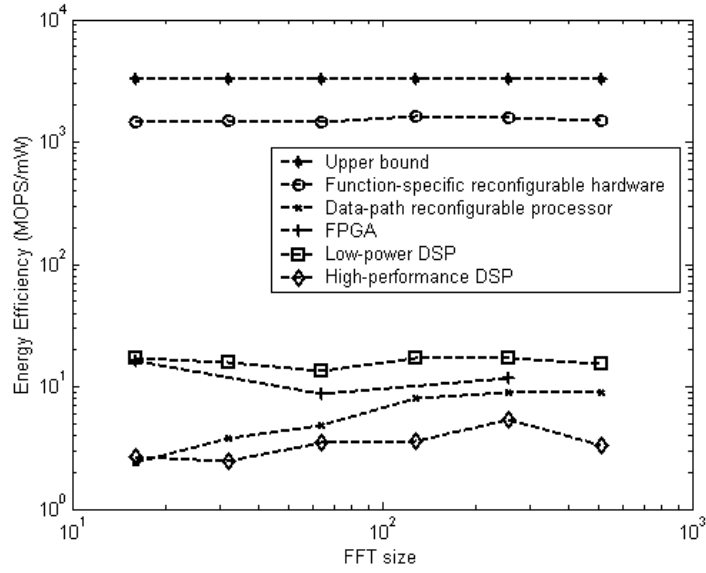


Figure 8: Energy efficiency comparison of FFT implementations

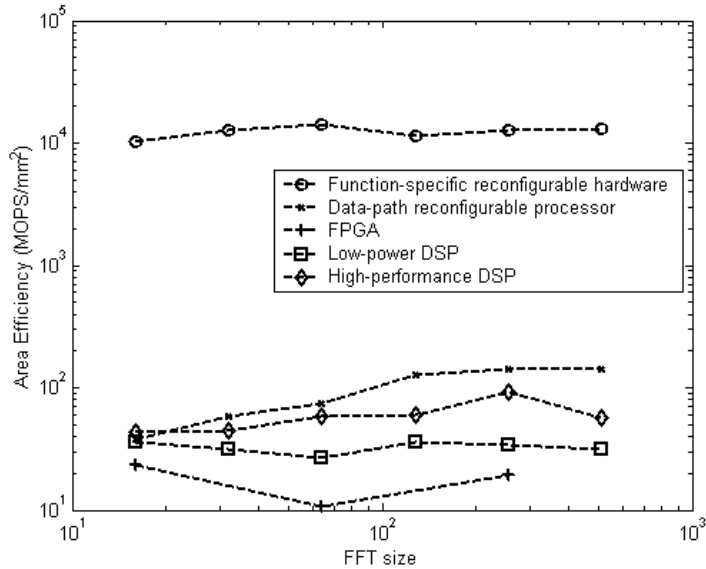


Figure 9: Area efficiency comparison of FFT implementations

## VI. Conclusion

Digital signal processing systems typically have hard real-time (throughput) constraints with flexibility as a required or desirable feature. The cost of this flexibility has been investigated and through the introduction of energy and area metrics the impact of unlimited flexibility has been quantified. It is seen that there is a range of up to 3 orders of magnitude in the energy and area inefficiency when full flexibility is compared to fully dedicated architectures. The architectural factors that contribute to the difference have



been also identified and analyzed with the level of granularity and the amount of parallelism being the critical parameters. Architectures that have an intermediate level of flexibility are also shown and demonstrate that it is possible to achieve nearly the efficiencies of dedicated solutions once the requirement of unlimited flexibility is removed.

## Acknowledgements

This work was sponsored by DARPA and the SIA under the MARCO focus centers program as well as the sponsors of the Berkeley Wireless Research Center.

## Reference

- [1] B. Gold and C. M. Rader, *Digital Processing of Signals*, 1969.
- [2] A.P. Chandrakasan and R.W. Brodersen, "Minimizing power consumption in digital CMOS circuit," *Proceedings of the IEEE*, vol.83, no.4, p.498-523, April 1995.
- [3] N. Zhang, A. Poon, D. Tse, R. W. Brodersen, and S. Verdu, "Trade-offs of Performance and Single Chip Implementation of Indoor Wireless Multi-Access Receivers," *Proc. IEEE Wireless Communications and Networking Conference*, September 1999.
- [4] W. Lee, et al., "A 1-V Programmable DSP for Wireless Communications," *IEEE Journal of Solid-State Circuits*, November, p. 1766-1777, 1997.
- [5] V. Tiwari, S. Malik, A. Wolfe, and M. Lee, "Instruction Level Power Analysis and Optimization of Software," *Journal of VLSI Signal Processing*, p. 1-18, 1996.
- [6] C. Turner, "Calculation of TMS320LC54x Power Dissipation," *Technical Application Report SPRA164*, Texas Instruments, 1997.
- [7] T. D. Burd, "Energy-Efficient Processor System Design," *Ph.D. Dissertation*, U. C. Berkley, 2001.
- [8] N. Zhang, "Implementation Issues in the Design of a CDMA Baseband Receiver," *Master thesis*, University of California, Berkeley, 1998.
- [9] B. M. Baas, "A low-power, high-performance, 1024-point FFT processor," *IEEE Journal of Solid-State Circuits*, March 1999.
- [10] N. Zhang and R. W. Brodersen, "Architectural Evaluation of Flexible Digital Signal Processing for Wireless Receivers," *Proc. Asilomar Conference on Signals, Systems and Computers*, Pacific Grove, CA, October 2000.
- [11] J. Goodman and A. Chandrakasan, "An Energy Efgcient IEEE 1363-based Reconfigurable Public-Key Cryptography Processor, 2001 ISSCC Technical Digest, pp. 330-331.
- [12] E. Kusse, "Analysis and Circuit Design for Low Power Programmable Logic Modules," *Masters Thesis*, University of California, Berkeley, 1997.