Comparing Power Consumption of an SMT and a CMP DSP for Mobile Phone Workloads

Stefanos Kaxiras
Agere Systems
600 Mountain Ave
Murray Hill, NJ 07974
kaxiras@agere.com

Girija Narlikar
Bell Labs, Lucent
600 Mountain Ave
Murray Hill, NJ 07974
narlikar@lucent.com

Alan D. Berenbaum
Agere Systems
600 Mountain Ave
Murray Hill, NJ 07974
adb@agere.com

Zhigang Hu
Engineering Quad
Princeton University
Princeton, NJ 08544
hzg@ee.princeton.edu

Abstract

In the DSP world, many media workloads have to perform a specific amount of work in a specific period of time. This observation led us to examine Simultaneous Multithreading (SMT) and Chip Multiprocessing (CMP) for a VLIW DSP architecture (specifically the Star*Core SC140), in conjunction with Frequency/Voltage scaling to decrease dynamic power consumption in next-generation wireless handsets. We study the resulting performance and power characteristics of the two approaches using simulation, compiled code, and realistic workloads that respect real-time constraints. We find that a multithreaded DSP can utilize the available functional units much more efficiently, performing as well as a non-multithreaded DSP but with substantial power savings. Power consumption can also be lowered by using a chip-multiprocessor (CMP) operating at low frequency. We compare the power consumption of an SMT DSP with a CMP DSP under different architectural assumptions; we find that the SMT DSP uses up to 40% less power than the CMP DSP in our target environment.

1 Introduction

New developments and new standards in communications, such as third generation wireless technology, call for a significant increase in DSP performance while at the same time the mobile device market demands ever lower power consumption. Developers are also moving from hand-written code to compiled code, as standards become more complex and the cost and time-to-market advantages of compilation outweigh the performance disadvantages. We are investigating simultaneous multithreaded VLIW DSPs as a promising direction to satisfy these diverse demands.

Using multithreading to improve the performance of a *workload* rather than the performance of a single application ameliorates the drawback of compiled code. Multithreading is particularly applicable to realistic cell phone workloads, which consist of multiple multimedia applications running simultaneously on the handset.

Multithreading was originally proposed as a way to increase throughput for a general-purpose workload by hiding long latencies [5]. Other studies [3][13] proposed similar mechanisms in order to allow multiple instruction streams to exploit data parallel architectures. More recently Tullsen, Eggers and Emer proposed simultaneous multithreading (SMT) to increase utilization of out-of-order superscalar processors [2][4]. What makes SMT appealing in that context is that the same hardware mechanisms that support out-of-order execution can be used to handle multiple simultaneous threads [4]. Recently announced commercial implementations of SMT include the Alpha 21464 [28] and the XStream network processor [27].

In the DSP arena, VLIW [6] rather than out-of-order superscalar architectures have prevailed for simplicity and chip-area reasons. Leading DSP architectures such as the TI 320C6x [7] or the Star*Core SC140 [8] leverage VLIW technology to provide multiple operations per cycle. In this paper we propose an SMT VLIW architecture using the Star*Core SC140 DSP as a starting point. We provide replicated thread state (e.g., multiple register files) but we share a single set of function units among all threads. In each cycle we select multiple (variable length) instruction packets from ready threads—as many as we can accommodate—and assign them to the function units.

By multithreading the workload we increase parallelism (Instructions per Cycle, or IPC), and can therefore decrease clock frequency and still do the same amount of work in the same time. Decreasing frequency also allows us to decrease the supply voltage (voltage scaling). Both lower frequency and lower voltage contribute to a significant reduction in power consumption.

Adding support for multithreading increases the chip area, leading to a higher load capacitance (which, in turn, increases dynamic power consumption). However, we show that this increase in power is offset by the power savings obtained from scaling the voltage and frequency. In this paper we study the

CASES'01, November 16-17, 2001, Atlanta, Georgia, USA.

power-performance trade-offs for two manufacturing technologies: a slower, $.25\mu$ technology and a faster, $.16\mu$ technology; the two technologies differ in their range of feasible operating frequencies and corresponding minimum V_{dd} values (data presented in Section 4).

We simulate multithreaded DSP architectures running workloads consisting a mix of speech encoders/decoders (GSM EFR), channel encoders/decoders (Trellis modulated channel encoding) and video encoders/decoders (MPEG-2). Our workloads approximate real-world processing in cell phones by respecting real time constraints. Our results show:

- A multithreaded DSP (with just the function units of the base DSP) can easily run a small number of compiled threads without significantly degrading their performance. By adding more load/store units—which are the bottleneck in our workloads—performance improves further.
- Despite the increased complexity and utilization of a multithreaded DSP, we can use it to reduce power consumption. We show how we can exploit the high IPC of the multithreaded architecture to reduce clock frequency and voltage and thus reduce power (and conserve energy) when the required performance is bounded. Power consumption can be reduced by a factor of 4 over a single-threaded DSP or by up to 40% over a chip-multiprocessor (CMP) DSP also running at low frequency and low voltage.

Structure of this paper. In Section 2 we describe the base architecture and the multithreaded DSP architecture. In Section 3 we discuss our evaluation methodology. Section 4 describes chip area and power consumption estimation. Section 5 presents the power comparison results, and Section 6 concludes this paper.

2 Multithreaded VLIW DSPs

To increase ILP without complexity, DSPs have turned to very long instruction word architectures (VLIW)[7][8]. VLIW directs data flow to all the parallel data units simultaneously as instructed by the compiler, in lieu of more complex issue logic attempting to uncover parallelism at runtime [6][24][20]. However, because the long instructions specify many simple operations, instead of a few complex or compound functions, VLIW DSPs make less efficient use of code memory. The increased memory traffic due to larger code footprint, plus the wide internal busses and function units running in parallel, mean that VLIW DSPs consume more power to perform the same work as more traditional DSPs.

2.1 Base architecture

The base architecture we have chosen for this study is the Star*Core architecture developed by Motorola and Lucent Technologies [8]. The Star*Core SC140 is a variable-length VLIW architecture containing one to six atomic operations which execute in parallel. Scheduling and instruction assembly are performed by the compiler.

Figure 5(a) shows a block diagram of the base architecture. The architecture contains four data ALUs (DALUs or MACs), which can perform single-cycle multiply-accumulates, two address AGUs (Address Generation Units), a bit-manipulation unit (BMU) and four bitfield units (BFU). There are 32 registers, 16 data (DREG file) and 16 address registers (AREG file). The data registers are internally extended to 40 bits wide to provide the required fixed-point precision required by international standards. The data register file is duplicated in two banks for fast access. Data registers can be read and written by four different DALUs within a single cycle. In addition to the above function units, there is program sequencer logic (PSEQ), and debugging logic (OTHER).

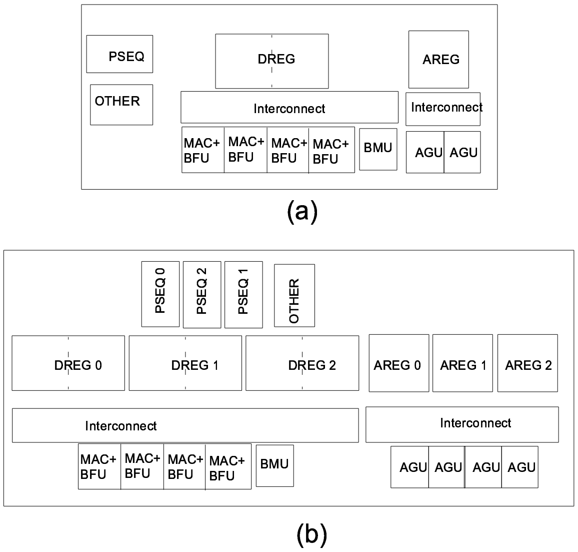

FIGURE 1. Block diagrams (not to scale) of (a) our base architecture, similar to Star*Core SC140, and (b) the SMT version, shown here with 3 hardware contexts. BMU is the bit mask unit; BFU represents the four bitfield units; AREG is the address register file; DREG is the data register file.

2.2 SMT architecture

For application workloads that consist of several independent threads, the throughput of a DSP can be increased by replicating the core DSP engine on a single die yielding a chip-multiprocessor (CMP) structure [10][22]. However, the utilization (the fraction of time spent in useful, non-idle computation) of function units is likely to be low, since not all threads will be able to issue maximum-width operations simultaneously and so on any cycle at least some of the function units will be idle.

An alternative to the CMP structure is Simultaneous Multithreading (SMT) [2][4][25]. Independent instruction issue units, including the register arrays that comprise a thread's state, simultaneously issue commands to a single shared set of function units. For the same number of register sets, SMT architectures require fewer execution units than the aggregate number used in CMP designs, and can potentially make more efficient use of resources that are available. For example, recent work studied the application of SMT to network processors [23].

Our model SMT architecture resembles the base architecture in that it retains the same function units (both in number and type). Instruction issue logic (PSEQ) and processor context (DREG and AREG register files and related state) are replicated appropriately to accommodate a fixed number of threads. We can increase the number of data ALU's, address ALU's and bit manipulation units arbitrarily but a single thread cannot use more than six units (the architectural length limit) at any given cycle. For both CMP and SMT alternatives we used the same compiler.

The largest additional cost in implementing the SMT version of the base architecture is routing data to and from the register arrays to the function units and—in accordance to the SMT work [4]—we allowed an additional pipeline stage over the base model to account for wire and setup delays. In the additional pipeline stage, we decide which threads will issue instructions to the ALUs. Multiple threads can issue their instructions as long as there are no resource conflicts.

As is common in DSP architectures we assume a memory system consisting solely of on-chip memory. Memory latency is 1 cycle and we model contention in the memory system through the address ALUs (AGUs). A larger number of AGUs translates to higher memory bandwidth. Memory banking can be used to provide higher bandwidth. This is also the case when we examine the CMP variant with shared on-chip memory.

3 Methodology

In this section we describe the DSP benchmarks used in our experiments. We then present the experimental setup used to compile and run the benchmarks.

3.1 Benchmarks

Our choice of benchmarks is intended to model next generation wireless handsets that support multimedia (voice and video) applications. Real-time constraints play an important role in this domain, especially for voice-based applications. It becomes critical to ensure that their performance is not affected by other, non-real-time applications that are running on the same DSP. Therefore, instead of evaluating the performance of each multimedia benchmark in isolation, we have designed a set of benchmarks that run simultaneously on the DSPs while obeying real-time constraints. We measure the power and performance for runs of one second duration. The following benchmarks will run simultaneously on a video-enabled mobile phone:

- **Speech encoder and decoder.** We have used bit-exact C code for a GSM standard speech coder, the Enhanced Full Rate (EFR) coder [11]. The standard requires a bit rate of 12.2Kbits/sec, with 244-bit frames transmitted every 20ms (such a bit rate is appropriate for wireless handsets). We therefore run the encoder and decoder on one frame of data every 20 ms, ensuring that it completes within the 20ms real-time deadline.
- **Channel encoder and decoder.** As shown in Figure 2 the speech encoder and decoder are connected to a channel encoder and channel decoder, respectively. The channel encoder/decoder use Viterbi's algorithm [16] for data coding and trellis-coded modulation [15]. To match the real-time requirements of the speech coder, we run both the channel encoder and decoder once every 20 milliseconds, for one frame's worth of input.

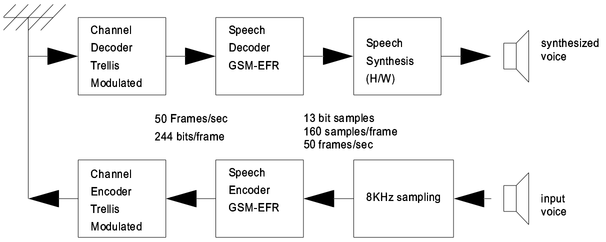

FIGURE 2. Speech processing in a cellular phone.

- **Video encoder and decoder.** We use an implementation of the MPEG-2 encoder and decoder provided by the MPEG Software Simulation Group [17]. The MPEG encoder is the most computationally intensive of all our applications; we use image sizes of 32 x 32 for the slower (.25μ) handset technology, and 64 x 64 for the faster (.16μ) technology. These are the largest frame sizes that can be run on the single-threaded base SC140 core at the highest operating frequencies for the two technologies. The MPEG encoder encodes 2 frames per second and the MPEG decoder decodes a 64 x 64 image at 3 frames per second. We approximate a parallel encoder with 4 independent MPEG encoder threads; each thread processes one quarter of the original image size (16x16 for the slower technology and 32x32 for the faster technology).

All the above six codes are adapted from sample implementations of industry standards, and were not modified to aid optimization by the compiler. For the single-threaded experiments, we run 6 threads: an MPEG encoder and decoder, a speech encoder and decoder, and a channel encoder and decoder. In the experiments with the SMT and CMP cores, we run 9 threads (three additional MPEG encoders); each of the 4 MPEG encoder threads processes a quarter of the original image. In all the experiments, the threads are run for a one second duration. The speech and channel encoders and decoders repeat for one frame of data every 20ms. The MPEG threads run once from start to completion.

3.2 Simulation environment

The experiments were carried out using a cycle-accurate, instruction-level simulator [18]. The simulator emulates the SC140 DSP core, including exception processing activity. We have extended the original simulator by adding support for multiple threads, each with a separate address space and register file. We use the Enterprise C/C++ compiler designed for the commercial SC100 family of DSP cores [14]. All the binaries used in our experiments were generated from pure C. Table 1 shows the IPC for each benchmark run (simulated) individually on the base SC140 architecture. Although the SC140 has a peak IPC of 6, it is clear that the compiler is unable to extract much parallelism. As the compiler evolves, it will likely be able to discover more parallelism. Improvements in parallelism are unlikely to affect the results significantly until, and if, the compiler can reach substantially greater levels of parallelism than it does now.

	IPC	Cycles for 1 sec
GSM EFR encoder (50 frames/sec)	1.42	20137650
GSM EFR decoder (50 frames/sec)	1.50	1815500
Trellis decoder (50 frames/sec)	1.16	1306100
Trellis encoder (50 frames/sec)	0.74	2018600
MPEG-2 encoder 16x16 frame (2 frames/sec)	0.58	16921130
MPEG-2 encoder 32x32 frame (2 frames/sec)	0.59	67013987
MPEG-2 encoder 64x64 frame (2 frames/sec)	0.61	268016411
MPEG-2 decoder 64x64 frame (3 frames/sec)	1.02	645008

TABLE 1: IPC and cycles for 4 threads (1 frame)

In our SMT experiments we found that the AGUs are a major bottleneck when running multiple threads. Thus, we examine an SMT design where we have increased the AGUs from two to four. No individual thread can use all four AGUs simultaneously since all threads have been compiled for only two AGUs available. However, the benefits in the multithreaded architecture are considerable, as shown in Section 5. Since AGUs are tied to memory ports in the SC140 architecture, we assume four memory ports in the enhanced SMT. This however does not lead to unfair comparisons with the CMP since there too we assume multiple memory ports to the same on-chip shared memory. Specifically, we assume that the CMP runs a parallelized version of the MPEG encoder which requires shared memory. Since each core in the CMP has by default two memory ports, we easily exceed the memory ports of the enhanced SMT when we use more than two cores. The on-chip memory can be divided into multiple pipelined banks to provide the necessary bandwidth in either design.

4 Power Computations

Dynamic power consumption is the chief source of power consumption in CMOS processors. Dynamic power is defined by the formula:

$$P = V^2_{dd} \times F \times A \times C$$

where F is the operating frequency, V_{dd} is the supply voltage and C is load capacitance of the circuit. The term A is the *activity factor* as used by Brooks, Tiwari, and Martonosi [26] to represent the average switching activity of transistors per clock cycle; it takes a value between zero (no switching) and one (switching every cycle). We derive the average power consumption over the one second period for an architecture (single- or multi-threaded) operating at frequency F using the following steps:

1. Given F, we scale V_{dd} from Figure 3 which shows the range of frequencies and the corresponding minimum V_{dd} for the two manufacturing technologies (0.25μ and 0.16μ IC technologies).

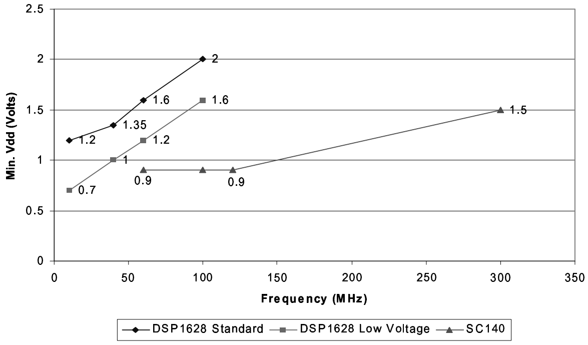

FIGURE 3. Minimum V_{dd} vs. frequency for .25μ DSP1628 (data adapted from [12]) and .16μ SC140 processes. V_{dd} does not decrease below 0.9Volts in the .16μ process in frequencies below 120MHz.

2. Since we do not have the absolute values of the load capacitances (C), we instead compute the power consumption of the SMT and CMP architectures *relative* to the power consumption of the base architecture, based on area estimates. For the multithreaded case, we use the load capacitances given 5 threads (hardware contexts). We assume conservatively that the increase in the load capacitance of a subblock will be proportional to the increase in area [29]. Area estimates are presented in Section 4.1.
3. We use our cycle-accurate simulator to determine the activity factors for the different subblocks on the DSP core similarly to the Wattch simulator [26]. For example, if over the 1-second run, the simulator finds that on average 1 DALU (out of 4) is busy per cycle, the DALU utilization A_{dalu} is computed to be 1/4. The AGU utilization A_{agu} is derived in a similar

manner. The AGU and DALU utilizations as a function of clock frequency are given in Figure 4. The activity factor of the address/data registers and their interconnect is the same as that of the AGUs/DALUs.

4. The average power consumption is:

$$P = V_{dd}^2 \times F \times \sum_{subblockb} C_b \times [A_b + (1 - A_b) \times Idle]$$

where C_b is the load capacitance of the subblock b. *Idle*, the idle ratio of the chip, is the fraction of each subblock (in terms of area) that always incurs transistor switching, even when the subblock has no other activity. The global clock distribution network is a subblock assumed to have a constant activity factor of 1. A_b is the activity factor for subblock b.

4.1 Chip area computation

As a starting point we use subblock areas of a synthesized SC140 core (subblock areas for the .16μ technology are listed in Table 2). Data and address register files are replicated on the SMT but since no single thread can issue more than 6 instructions per cycle, each individual register file has the same number of ports—hence same size—as in the base architecture. The increased program sequencer logic (PSEQ) and interconnect among the registers and functional units also contributes to the additional area in the SMT. Support circuitry and other function units (e.g., BMU) are not replicated in the multithreaded architecture.

Unit	area (mm^2)	area (%)
MAC x4	1.04	10%
BFU x 4	0.80	8%
DREG x2	2.09	21%
BMU	2.35	24%
AGU x2	.50	5%
AREG	1.04	10%
PSEQ	.80	8%
OTHER	1.33	14%
TOTAL	9.95	100%

TABLE 2: Absolute and relative areas of function units and register files for the synthesized SC140 core using Lucent's COM2 SCDS process (.16μ technology).

Similarly to previous work [13], we use the process-independent parameter λ to estimate the size of the registers and their interconnects. λ is equal to one half of the minimum feature size; for the .25μ technology, λ = .125μm and for the .16μ technology, λ = .08μm. Based on the areas for each block of the base architecture (Table 2) and assuming a wire pitch of 6λ, we compute the lengths and areas in the base and SMT

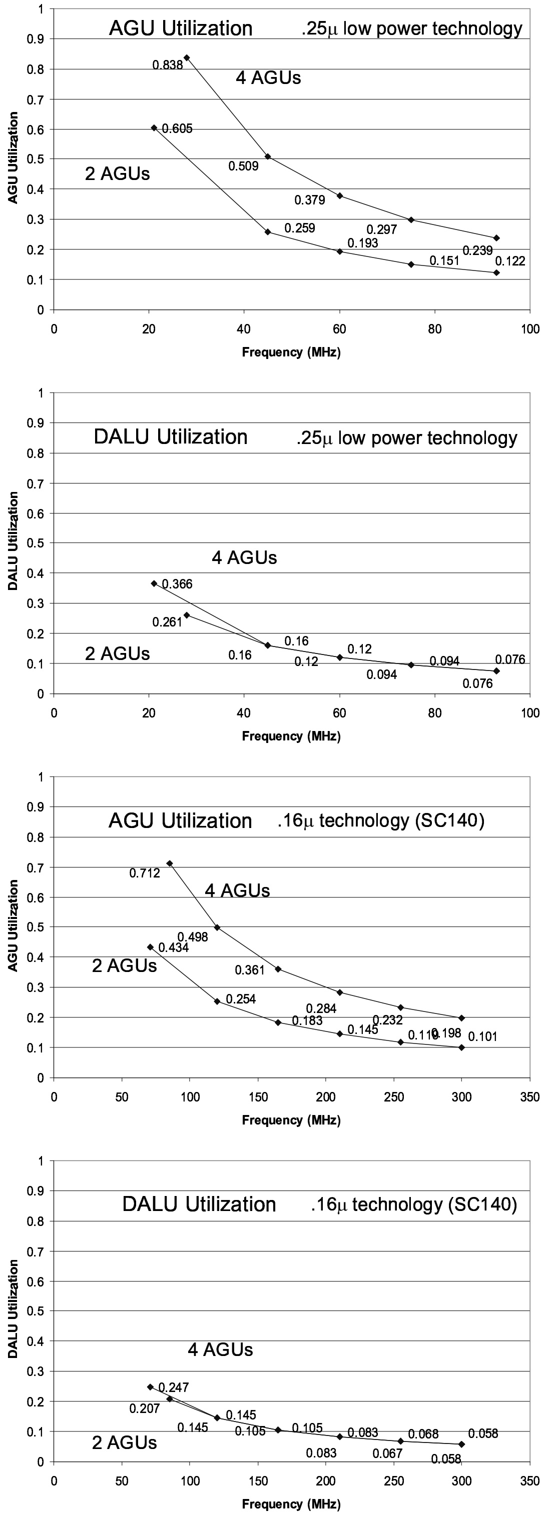

FIGURE 4. AGU and DALU utilization as a function of frequency for the multithreaded DSP running a fixed workload for one second.

cores in terms of λ. The dimensions of the data and address registers, the MACs (DALUs), the AGUs, and their bus-based interconnect for the .16μ technology are shown in Figure 5. We conservatively assume that the area for the program sequencer logic (PSEQ) increases linearly with the number of hardware contexts. In Figure 5, we illustrate an SMT with only 3 (instead of the 5 we use in our evaluations) hardware thread contexts for clarity. Since λ is process-independent, we also apply the same λ-based area increments to the .25μ technology.

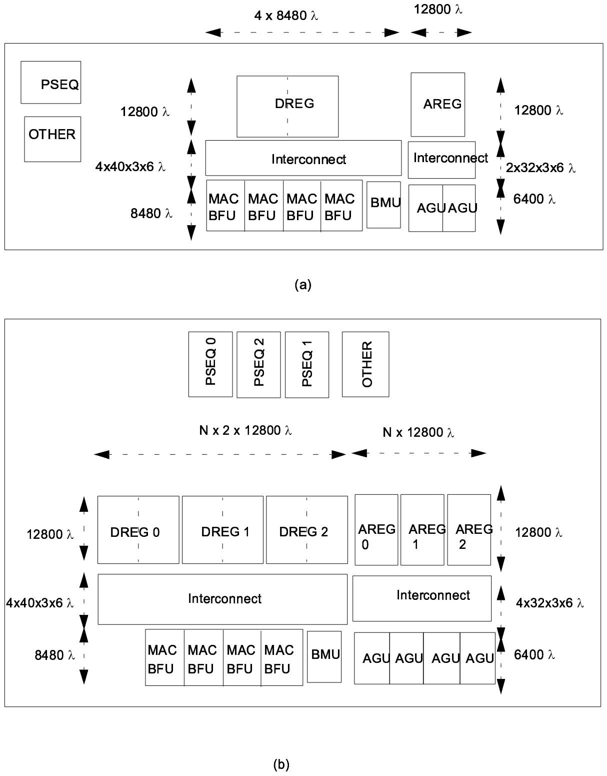

FIGURE 5. Subblock dimensions for (a) our base architecture and (b) an SMT with 3 hardware contexts.

The SMT version of the SC140 has 5 pairs of DREGs, one for each hardware context; each pair is 2*12800 λ wide. Therefore, the length of the DALU-DREG interconnect now becomes 5*2*12800 λ, and its area becomes 2.36mm^2. Therefore the additional routing (interconnect) overhead for the DREGs in the SMT is 2.36-0.625 = 1.735mm^2 = 17.4% of the base SC140 chip area (see Figure 5). In addition to the interconnect, the total area for the DREGs is scaled up by a factor of 5 for the SMT.

The area for the AGUs, the address registers, and the interconnect between them is derived in a similar manner. When the number of AGUs is increased to 4 in the SMT version, the height of the bus-based interconnect also doubles (the number of buses goes from 2 to 4). The number of read/write ports (and hence the area) for each address register remains the same, since each hardware context can only access two AGUs in any cycle. Thus, the areas of the address registers and the AGUs are scaled up linearly with number of threads.

Table 3 shows estimated areas for the multithreaded DSPs with up to five threads. All percentages are with respect to the chip area of the base architecture. The chip area for a CMP increases linearly with the number of cores, while the corresponding area overhead for the SMT is significantly lower.

2 AGUS

	Threads				
Block	**Base**	**2**	**3**	**4**	**5**
4 MACs+ BFUs	18%	18%	18%	18%	18%
2 AGUs	5%	5%	5%	5%	5%
1 BMU	24%	24%	24%	24%	24%
Data register files	21%	42%	63%	84%	105%
Address register files	10%	20%	30%	40%	50%
PSEQ	8%	16%	24%	32%	40%
Other	14%	14%	14%	14%	14%
Routing over-head	0%	7%	14%	21%	28%
Total	100%	144%	188%	232%	276%
CMP Area	100%	200%	300%	400%	500%

4 AGUS

	Threads				
Block	**Base**	**2**	**3**	**4**	**5**
4 MACs+ BFUs	18%	18%	18%	18%	18%
2 AGUs	5%	10%	10%	10%	10%
1 BMU	24%	24%	24%	24%	24%
Data register files	21%	42%	63%	84%	105%
Address register files	10%	40%	60%	80%	100%
PSEQ	8%	16%	24%	32%	40%
Other	14%	14%	14%	14%	14%
Routing over-head	0%	11%	21%	30%	40%
Total	100%	175%	234%	292%	351%
CMP Area	100%	200%	300%	400%	500%

TABLE 3: Chip area estimates for the multithreaded architecture (all estimates conservative). All percentages correspond to chip area of the base architecture. CMP areas are for configurations with 1 to 5 processors.

5 Experimental Results

In all the experiments, we simulate one full second of processing for the base architecture, the CMP with five replicated cores, and the SMT architecture with five hardware contexts.

The speech and channel threads involve a fixed amount of computation and their spacing every 20ms dilutes the IPC as a function of clock frequency: the higher the frequency, the lower the IPC. Figure 6 shows the IPC as a function of clock frequency for the two technologies and for two and four AGUs. As frequency increases the multithreaded DSP is able to finish the large MPEG encoder threads early, leaving the speech and channel threads running every 20ms.

For all our SMT experiments we selected the following mapping, which allows the benchmarks to meet their deadlines at the lowest possible frequency (see Figure 7); this was also the mapping of threads to processors in the CMP experiments.

- .25μ technology: one of the five hardware contexts executes the speech encoder, while the other four hardware contexts each execute an MPEG encoder thread along with one of the other four remaining applications (speech decoder, channel encoder, channel decoder, MPEG decoder).
- .16μ technology: the four MPEG encoders run on separate hardware contexts, while the remaining five applications (speech codecs, channel codecs, and MPEG decoder) all execute on the fifth hardware context.

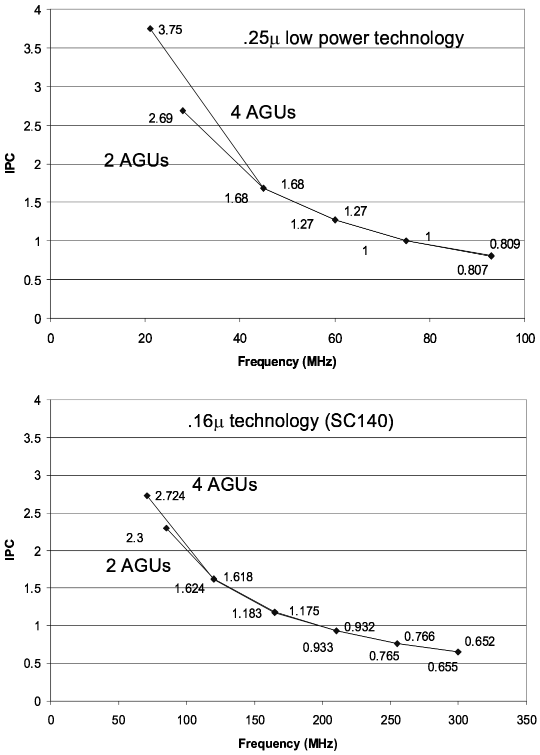

FIGURE 6. Average IPC of a fixed cell phone workload running for one second as a function of clock frequency for a multithreaded DSP with 5 hardware contexts

5.1 Comparison of power consumption

In the SMT case, for the .25μ technology, the lowest feasible operating frequency was 28 MHz (using 2 AGUs). Lowering the frequency further did not allow the workload to meet the 20ms real-time deadline. Increasing the number of AGUs to 4 increases performance, and therefore allows the frequency to be further lowered to 21 MHz. Similarly, for the .16μ technology, the lowest feasible operating frequency was determined to be 85 MHz using 2 AGUs and 71 MHz using 4 AGUs. In this case, the MPEG encoder was the limiting factor in reducing frequency. Similarly, the lowest possible frequencies for the CMP architecture are 20 MHz for the .25μ technology, and 67 MHz for the .16μ technology. For the .16μ technology voltage scaling stops at 120MHz and we cannot go below 0.9Volts V_{dd} for lower frequencies (Figure 3).

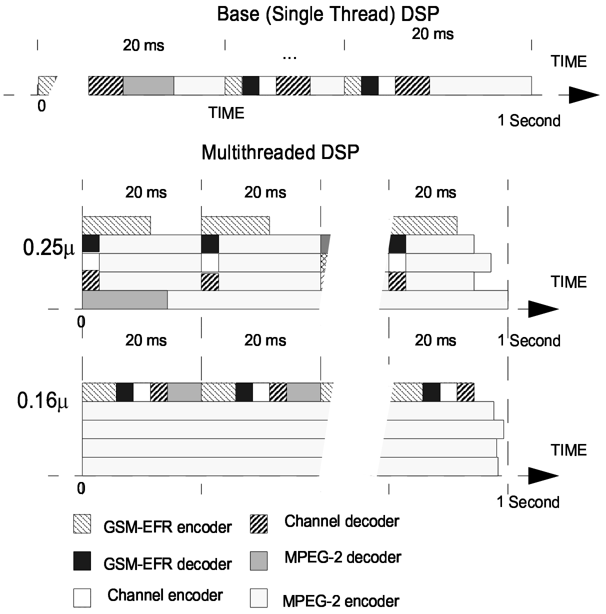

FIGURE 7. Schedule for the cell-phone workload. The GSM and Channel codecs run once every 20ms and have to finish within this time. The MPEG-2 codecs consume the rest of the cycles within the one second we simulate. In the base architecture (top) threads context-switch with zero overhead. In the multithreaded architecture threads run in five hardware contexts.

We ran the multithreaded experiments at different frequencies ranging from the lowest to the highest feasible operating frequencies. For each frequency, we computed the power consumptions of the CMP and SMT DSPs relative to the base architecture. Based on previous work [26] we assume that the global clock distribution network requires either 0% or 10% of the chip area. Similarly, the idle ratio (fraction of each subblock that is always switching) is assumed to be either 0% or 10%. The resulting power consumptions for the 4 (idle ratio, global clock) combinations are shown in Figures 8, 9, 10, and 11. In all four cases, substantial savings in power over the base architecture are possible at low frequencies using both the SMT and CMP versions of the DSP. As clock frequency is increased, however, they begin consuming more power because the effect of increased frequency and V_{dd} outweighs the effect of lower utilization. Further, the load capacitances of the SMT and DSP chips are higher than the base DSP, and therefore as frequency is increased, they eventually begin to consume more power than the base DSP. For example, the break-even points for the SMT DSP are at 60MHz and 195MHz for the .25μ and .16μ technologies respectively,

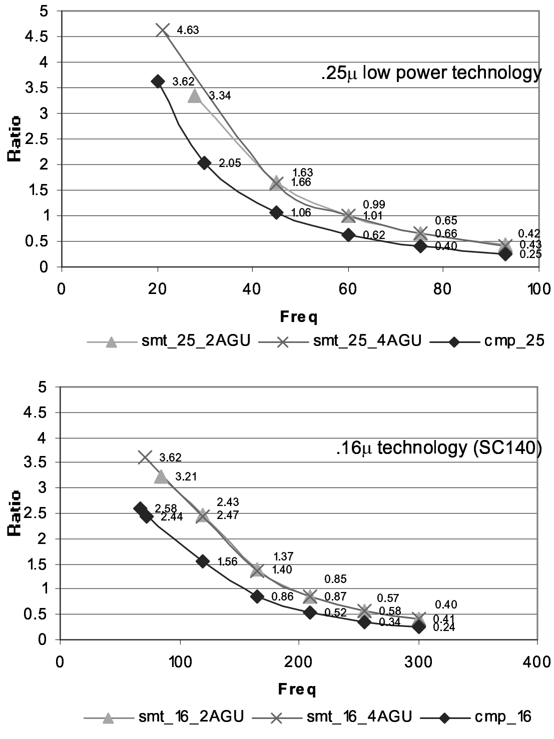

FIGURE 8. Power ratio [Base/(SMT or CMP)] executing the same workload in one second. Ratios greater that one indicate that the SMT or CMP DSP is more power-efficient than the base architecture. We vary the frequency of the SMT and CMP DSPs down to the lowest possible frequency that can safely accommodate the workload. Frequency cannot be further decreased without breaking real-time constraints. Here idle ratio = 10% and global clock = 10%.

when the idle ratio is 10% and the global clock is 10%.

For the SMT architecture, we show results for both 2 and 4 AGUs. Increasing the number of AGUs allows a lower operating frequency, and this benefit outweighs the increased load capacitance of the AGUs and address registers. As seen in Figures 8-11, this benefit is cancelled out at higher frequencies.

For both technologies, the lowest feasible operating frequency for the CMP system is lower than that of the SMT system; this is because the most computationally intensive benchmarks run on separate processors and do not contend for resources. However, as seen from the results, the benefit of a lower frequency is outweighed by the increased load capacitance of the CMP system, except when no power is consumed in idle mode (idle ratio = 0% and global clock = 0%). In that case, the CMP outperforms the SMT in the power comparison, since it operates at a lower frequency and V_{dd} setting. Also, the power consumption increases slightly at the lowest frequencies for the .16μ technology (Figure 11). This is because V_{dd} cannot be lowered beyond 0.9V for frequencies below 120MHz, and unlike in Figures 8-10, no power is consumed during idle cycles at higher frequencies. In a more realistic setting when the idle ratio = 10% and global clock = 10%, the SMT system consumes 28% (40%) less power than the CMP system for the .25μ (.16μ) technology.

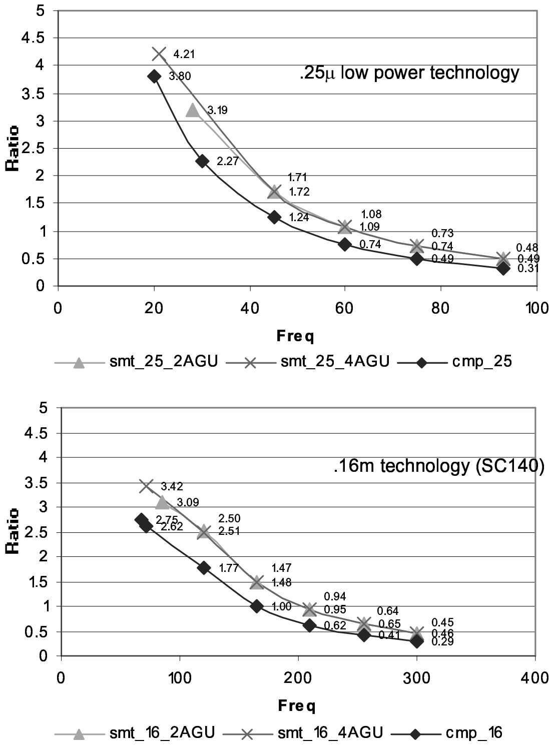

FIGURE 9. Power ratio [Base/(SMT or CMP)] executing the same workload in one second. Here idle ratio = 10% and global clock = 0%.

6 Conclusions

Signal processing and real-time applications often require a different figure of merit than minimizing the number of cycles required to complete a task. Minimization of power may be more important, as long as application-specific real-time requirements are met. We conducted experiments that represent real-world multimedia workloads for mobile handsets, where power consumption is a major concern.

The experiments show that using SMT it is possible to save power (by up to a factor of 4.6) in comparison with a uniprocessor, single-threaded architecture running at a higher clock rate. The SMT processor can meet the application-specific real time deadlines at a lower clock rate and supply voltage by running tasks in parallel as separate threads and making more efficient use of the functional units. Compared to a CMP that can also run multiple tasks in parallel at low frequency and low voltage, the SMT typically retains some advantage in power consumption.

We used a commercial DSP architecture as the base of our study, and did not modify the compiler or other software tools. The results are therefore conservative in that it is possible to

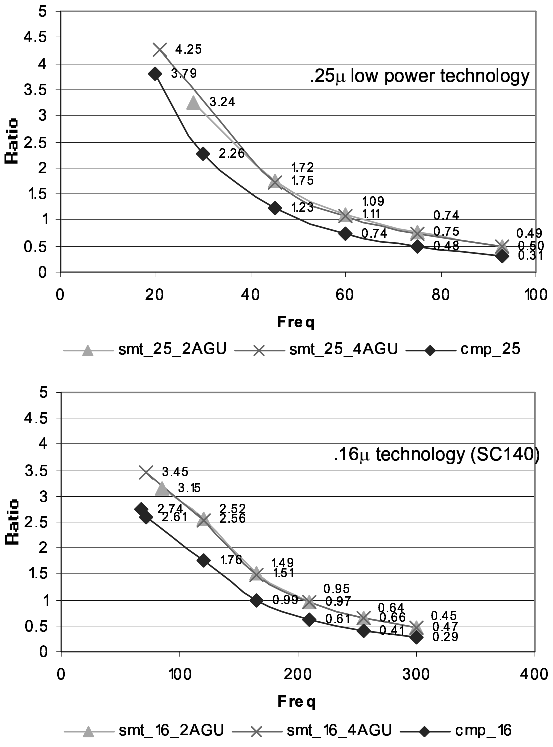

FIGURE 10. Power ratio [Base/(SMT or CMP)] executing the same workload in one second. Here idle ratio = 0% and global clock =10%.

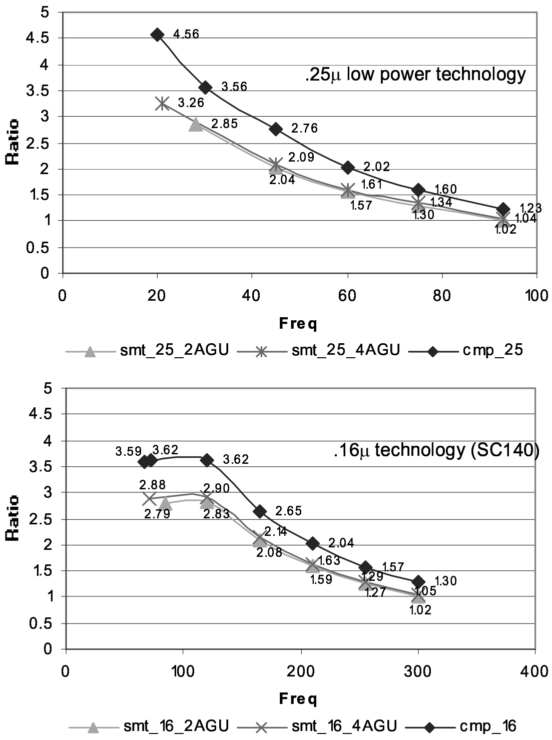

FIGURE 11. Power ratio [Base/(SMT or CMP)] executing the same workload in one second. Here idle ratio = 0% and global clock = 0%

optimize the programs to exploit the SMT configuration and extend the efficiency advantages of SMT over CMP organizations. On the other hand, our study is constrained by the compiler we used and the workloads we chose. Our compiled codes do not exhibit high IPC so a multithreaded architecture can easily accommodate multiple instances. However, we believe that compiled code is becoming increasingly important in the development cycle for DSP applications and an architecture that reduces the impact of lower compiled performance is likely to find acceptance. In many applications, the power and cost benefits of the SMT approach could make it a more attractive alternative to a simpler CMP design.

Acknowledgments

We would like to thank Nevin Heintze, Rae Mcllelan, Tor Jeremiassen, and Cliff Young for their comments on drafts of this paper. We would also like to thank Paul D'Arcy and Isik Kizilyalli who provided data for this parer.

7 References

[1] Ojanpera and R. Prasad, Wideband CDMA for Third Generation Mobile Communications, Artech House, Oct. 1998.

[2] Dean Tullsen, Susan Eggers, and Henry Levy "Simultaneous Multithreading: Maximizing On-Chip Parallelism," In *Proceedings of the 22rd Annual International Symposium on Computer Architecture*, Santa Margherita Ligure, Italy, June 1995, pages 392-403.

[3] Andrew Wolfe and John P. Shen, "A variable instruction stream extension to the VLIW", Pages 2 - 14, In *Proceedings of Fourth International Conference on Architectural Support for Programming Languages and Operating Systems,* 1991.

[4] Dean Tullsen, Susan Eggers, Joel Emer, Henry Levy "Exploiting Choice: Instruction Fetch and Issue on an Implementable Simultaneous Multithreading Processor," In *Proceedings of the 23rd Annual International Symposium on Computer Architecture,* Philadelphia, May 1996.

[5] B. Smith, "Architecture and Applications of the HEP multiprocessor computer System," *In SPIE Real Time Signal Processing IV*, pp 241-248, 1981.

[6] J. Fisher, "VLIW architectures: an inevitable standard for the future?" *Journal of Supercomputer, Vol. 7, No 2*, pp 29-36. Mar 1990.

[7] Texas Instruments. TMS320C6211 Fixed Point Digital Signal Processor—Product Review, August 1998. SPRS073.

[8] “Star*Core Launches First Architecture,” *Microprocessor Report 12:14*, 10/26/98.

[9] “Lucent rolls out its first Star*Core-based DSP, promises to double Internet chip capacity”, *Semiconductor Business News*, 6/12/00.

[10] Basem A. Nayfeh and Kunle Olukotum, “Exploring the Design Space for a Shared-Cache Multiprocessor,” In *Proceedings of the 23rd Annual International Symposium on Computer Architecture,* Chicago, April 1994.

[11] European Telecommunications Standards Institute. “Digital cellular telecommunications system: Enhanced Full Rate (EFR) speech transcoding (GSM 06.60),” March 1997.

[12] I.C Kizilyalli et al. “A WSi/WSiN/Poly:Si Gate CMOS Technology for 1.0V DSPs,” In *Proceedings of the First International Symposium on ULSI Process integration, The Electrochemical Society Proceedings Vol. 99-18*. pp 347-352.

[13] Stephen W. Keckler and William J. Dally, “Processor coupling: integrating compile time and runtime scheduling for parallelism”, In *Proceedings of the 19th annual International Symposium on Computer Architecture,* Queensland Australia, 1992.

[14] Star*Core, “SC100 C/C++ Compiler User’s Manual”, available at http://www.starcore-dsp.com/.

[15] E. Biglieri, D. Divsalar, P. McLane, and M. K. Simon, “Introduction to Trellis-Coded Modulation with Applications”, Macmillan, New York, 1991.

[16] A. J. Viterbi, “Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm”, *IEEE Trans. Information Theory, vol. IT-13*, 260-269, April 1967.

[17] MPEG Software Simulation Group (MSSG), http://www.mpeg.org/MPEG/MSSG/

[18] Tor E. Jeremiassen, “A DSP with Caches - A Study of the GSM-EFR Codec on the TI C6211,” *IEEE International Conference on Computer Design*, pages 138-145, October 1999.

[19] MicroDesign Resources, “Embedded Processor Watch, #78”, December 14, 1999. http://www.mpronline.com

[20] Keith Diefendorff, “Transmeta Unveils Crusoe: Supersecret Startup Attacks Mobile Market with VLIW, Code Morphing,” *Microprocessor Report*, 1/24/00.

[21] “Philips Hopes to Displace DSPs with VLIW,” *Microprocessor Report 8:16*, 12/5/94.

[22] L. Gwennap, “Network processors race toward 10-gigabit goal,” *Electrical Engineering Times, Issue 1118*, 6/19/00.

[23] P. Crowley, M. Fiuczynski, J.-L. Baer and B. Bershad, “Characterizing Processor Architectures for Programmable Network Interfaces.” *Proceedings of the 2000 International Conference on Supercomputing*, Santa Fe, NM, May 2000.

[24] “IA-64 and Merced--What and Why,” *Microprocessor Report*, 12/30/96

[25] Michael Slater, Linley Gwennap, Keith Diefendorff, Peter Glaskowsky “Alpha 21464 Targets 1.7 GHz in 2003,” *Microprocessor Watch Issue #25*, MicroDesign Resources, November 18, 1999.

[26] David Brooks, Vivek Tiwari, Margaret Martonosi, “Wattch: A Framework for Architectural-Level Power Analysis and Optimizations,” In *Proceedings of the 27th Annual International Symposium on Computer Architecture,* Vancouver Canada, June 2000.

[27] Peter Glaskowsky, "Networking Gets XStream: Startup Debuts Simultaneous Multithreading in Network Processor", *Microprocessor Report,* 11/13/00, pg. 1-2.

[28] Joel S. Emer, “Simultaneous Multithreading: Multiplying Alpha Performance”, *Microprocessor Forum*, October 1999.

[29] Tohru Ishihara and Hiroto Yasuura, “Experimental analysis of Power Estimation Models of CMOS VLSI Circuits”, *IEICE Trans. Fundamentals*, Vol. E80-A, No. 3, pp.480-486, March 1997.