SYNTHESIS OF POWER EFFICIENT SYSTEMS-ON-SILICON

Darko Kirovski[*], *Chunho Lee*[*], *Miodrag Potkonjak*[*], *and William Mangione-Smith*[o]
[*]Computer Science Department and [o]Electrical Engineering Department
University of California, Los Angeles, CA 90095-1596, USA

ABSTRACT

We developed a new modular synthesis approach for design of low-power core-based data-intensive application-specific systems on silicon. The power optimization is conducted in three steps: minimization of instruction cache misses, placement of frequently executed sequential basic blocks of code in consecutive Gray code addressed memory locations, and processor and cache application-driven selection for low-power. In order to bridge the gap between the profiling and modeling tools from the two traditionally disjoint synthesis domains (architecture and CAD), we developed a new synthesis and evaluation platform. The platform integrates the existing modeling, profiling, and simulation tools with the developed system-level synthesis tools. The effectiveness of the approach is demonstrated on a variety of modern industrial-strength multimedia and communication applications.

1. INTRODUCTION

The ever increasing popularity of communications applications, such as wireless telephony, Internet browsing, has established them as a target for developers of consumer electronics. Systems for these markets demand high design flexibility and large volume data management. The fact that battery life has not followed the progress pace of the semiconductor technology, has established low-power design as premier design goal for many systems. Similarly, semiconductor technologies have allowed high-level integration of processors and memory structures on a single die. Thus, low-power systems-on-silicon (SOS) consisting of a programmable processor and memory hierarchy attracted a great deal of attention of all silicon vendors. Since significant portion of the SOS power consumption comes from the processor and cache cores, we developed a new synthesis technique which focuses on selecting a processor and instruction and data cache configuration for minimized power consumption on a set of target applications.

The distribution of power dissipation for an application-specific SOS depends on the actual application. An extensive on-chip power distribution analysis has been conducted for a number of modern general purpose processors. Gonzalez and Horowitz [Gon96] show that 25%-40% of the total energy dissipation of a typical general purpose microprocessor is consumed by on-chip caches. Burd and Brodersen [Bur96] show that out of 1.2W of total Infopad processor system power dissipation, 120mW and 600mW are due to microprocessor (ARM60) and 512Kb SRAM cache. The Toshiba design team [Nag95] showed that the cache, the datapath, power and control power are about 50 - 100mW, 100mW - 150mW and 50mW for 3.3V 50MHz operation of their 150MIPS/W RISC.

We developed a new synthesis approach for low-power core-based data-intensive application specific SOS. The approach optimizes the power of the integrated system by applying a number of power and performance optimization strategies. The power minimization is addressed at several synthesis and compilation phases, including placement of frequently executed sequential basic blocks of program in consecutive Gray code addressed memory locations, minimization of instruction cache misses, processor, I- and D- cache size and organization selection.

The compilation strategy for energy reduction relies on a global least-constraining most-constrained probabilistic heuristic used to minimize the number of I-cache conflicts and the switching activity in the I-cache decoding logic by relocating code. Improved cache performance enables scaling down the supply voltage. This change results in larger delays which are tolerated according to the target application's timing constraints. Cache conflicts are reduced by relocating basic blocks in the executable in such a way that frequently sequentially executed blocks map to different cache lines. The heuristic for performance optimization groups basic blocks into sets, where all blocks in the set map the same cache line. The actual assignment of sets of blocks to discrete cache lines is done by the power-optimization heuristic. This heuristic shuffles the sets according to the profiling information of the program execution, so that frequently sequentially executed basic blocks are stored into memory locations with consecutive Gray-code addresses. Thus, the switching activity of the cache decoder is minimized for a particular program execution.

As an outer synthesis loop, the resource allocation algorithm performs a bounded search for a power efficient system configuration. The result consists of a microprocessor core selection, and I- and D-cache specification (size, associativity and line size). The bounded search is guided by evaluating the design trade-offs for the selection of each of the configuration parameters. Sharp lower and upper bounds significantly reduce the number of cache configurations considered for power and performance estimation. This reduction is important because accurate cache performance and power estimation is done using trace-driven simulations.

In order to bridge the gap between the profiling and modeling tools from the two traditionally independent synthesis domains (architecture and CAD), we have developed a new synthesis and evaluation platform. The platform integrates the existing modeling (CACTi [Wil94]), profiling (SHADE [Cme94]) and simulation (DINEROIII [Hil88]) tools with the developed system-level synthesis tools. The effectiveness of the approach is demonstrated on the MediaBench benchmark suite [Lee97].

2. PREVIOUS WORK

The related work can be traced along the following three lines: application-specific system synthesis, processor and memory hierarchy design and energy-efficiency evaluation, and cache line coloring techniques. The market-driven need for application specific system cost reduction has spurred the development of system level synthesis techniques [Wol94, Pot95]. As embedded applications have become more sophisticated and commercially relevant, hardware-software codesign and techniques for system level syn-

thesis have also become increasingly important [Gup93, Gaj94]. The importance of low-power design has resulted in adequate treatment of these optimization issues in system-level and high-level synthesis [Meh96, Lee96].

The increased interest in embedded system design with reusable components has encouraged the development of high level architecture evaluation models. The Microprocessor Report presents a monthly summary of the power dissipation and performance of processor cores [Mic97]. Low-power processor design has been discussed in [Bur96, Gon96]. I- and D-caches, as the highest level of the memory hierarchy, have been thoroughly studied [Hil88, Jou93]. Cache models for latency and power have been developed and power minimization strategies have been evaluated for a number of design parameters [Wil94, Wad92, Ko95, Eva95, Su95].

Compiler directed techniques for minimizing number of cache misses in such studies has received significant attention in the research community. Bershad et al. have proposed dynamic address remapping for avoiding conflict misses in large direct-mapped instruction caches [Ber94]. Static code repositioning by using cache line coloring at the procedure or basic block level has been an alternative approach proposed and evaluated in [Hwu89, Tor96].

3. PRELIMINARIES

Several factors combine to influence system performance: instruction and data cache miss rates and penalty, processor performance, and system clock speed. Power dissipation of the system is estimated using processor power dissipation per instruction and number of executed instructions per task, supply voltage, energy required for cache read, write, and off-chip data access as well as the profiling information about the number of cache and off-chip accesses. The approach that we use to realistically address the factors used to estimate power consumption leverages on existing cache and processor models.

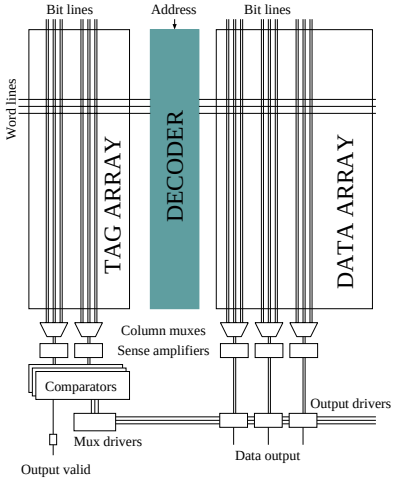

Figure 1: Standard cache implementation.

Typical cache design is illustrated in Figure 1. The influence of invidual cache components on its latency is quantified in [Wad92, Wil94]. There exists also a number of reports on how energy consumption is distributed among parts of the cache structure. Evans and Franzon [Eva95] show that global decode, precharging, and address latches are each responsible for approximately one third of the power consumption. Ko and Balsara [Ko94] conclude that the bit array, address decode, control generator, and sense amplifiers consume 40, 18, 25, and 17% respectively of the total power dissipation. We used CACTi [Wil94] as a cache delay estimation tool with respect to the main cache design choices: size, associativity, and line size. The energy model was adopted from [Eva95, Su94] and scaled with respect to the industrial implementations.

Minimal cycle time (ns) for direct-mapped caches with variable line sizes					
Cache size	8B	16B	64B	128B	512B
512B	6.06857	5.98835	-	-	-
1KB	6.44459	6.1325	6.23345	-	-
2KB	6.88323	6.51749	6.34239	6.5073	-
4KB	7.66557	7.02079	6.48923	6.56152	-
8KB	8.3447	7.80652	6.99162	6.90572	9.39627
16KB	9.30408	8.58294	7.62046	7.54262	9.79919
32KB	1.04131e-08	9.44989	8.591	8.69023	10.04495

Table 1: A sample of the cache latency model.

Power consumption (nJ) estimation for various direct-mapped caches with variable line sizes (5V)						
	No optimizations			Block buffering, sub-banking and Gray code addressing		
Cache size	8B	16B	32B	8B	16B	32B
512B	0.3301	0.3777	0.4683	0.2913	0.3219	0.4012
1KB	0.3561	0.3943	0.4626	0.2953	0.2993	0.3452
2KB	0.4223	0.4442	0.4894	0.3171	0.2706	0.2710
4KB	0.6513	0.6662	0.6942	0.4557	0.3336	0.2729
8KB	1.146	1.15.6	1.175	0.7686	0.5044	0.3474
16KB	2.158	2.164	2.174	1.412	0.8693	0.5298
32KB	4.198	4.202	4.209	2.702	1.608	0.9223

Table 2: A sample of the cache power consumption model.

Caches typically found in current embedded systems range from 128B to 32KB. Although larger caches correspond to higher hit rates, their power consumption is proportionally higher, resulting in a design trade-off. Since higher cache associativity results in significantly higher access time, we consider only direct-mapped caches. We experimented 2-way set associative caches, but they did not dominate in a single case. Cache line size was variable in our experimentations. Its variation corresponded to the following trade-off: larger line size results in higher cache miss penalty delay and higher power consumption by the sense amplifiers and column decoders; smaller line size results in large cache decoder power consumption. Extreme values result in significantly increased access time. We estimated the cache miss penalty based on the operating frequency of the system and external bus width and clock for each system investigated. Write-back was adopted in oppose to write-through, since it is proven to provide superior performance and especially power savings in uniprocessor systems [Jou93] though at increased hardware cost. Each of the processors considered is constrained by the Flynn's limit, and is able to issue at most a single instruction per clock period. Thus, caches were designed to have a single access port. A sample of the cache model data is given in Tables 1 and 2. Cache access delay and power consumption model were computed for a number of organizations and sizes, assuming feature size of 0.5 μm and typical six transistors per CMOS SRAM cell. The nominal energy consumption per single off-chip memory access, $98nJ$, was adopted from [Fro97].

Data on microprocessor cores was extracted from The Microprocessor Report [MPR97]. A sample of the collected data is presented in Table 3. The table presents embedded microprocessor core operating frequency, MIPS performance, technology, area, and nominal power consumption. The last two rows of the table show two integrated microprocessor products with on-chip caches and their system performance data.

Voltage scaling is used as a powerful power consumption minimization strategy in the developed synthesis strategy. The supply

voltage V_{dd} vs. delay T_D model used in the experimentations was adopted from [Cha92] and is quantified by the following formula: $T_D = K \cdot \frac{V_{dd}}{(V_{dd}-V_t)^2}$, where K is constant.

Microprocessor core	Clock MHz	MIPS	Feature μm	Area mm^2	Power mW
StrongARM	233	266	0.35	4.3	300 (1.65V)
ARM, 7	40	36	0.6	5.9	200 (5V)
ARM, 7 Low-Power	27	24	0.6	3.8	45 (3.3V)
LSI Logic, TR4101	81	30	0.35	2	81 (3.3V)
LSI Logic, CW4001	60	53	0.5	3.5	120 (3.3V)
LSI Logic, CW4011	80	120	0.5	7	280 (3.3V)
DSP Group, Oak	80	80	0.6	8.4	190 (5V)
NEC, R4100	40	40	0.35	5.4	120 (3.3V)
Toshiba, R3900	50	50	0.6	15	400 (3.3V)
Motorola, 68000	33	16	0.5	4.4	35 (3.3V)
PowerPC, 403	33	41	0.5	7.5	40 (3.3V)
ARM7 / 8KB	40	72	0.6	34	424 (5V)
StrongARM / 32KB	200	230	0.35	50	900 (1.65V)

Table 3: A sample of the processor performance vs. area model.

4. THE SYSTEM-LEVEL SYNTHESIS APPROACH

In this section we describe the function of each module in the synthesis system and how modules are combined into a modular optimizing and synthesis system. The core engine of the synthesis flow is the global least-constraining most-constrained heuristic for basic block relocation. Basic blocks are repositioned statically in such a way that frequently sequentially executed basic blocks are mapped onto different cache lines. The mappings are shuffled so that the switching activity in the I-cache decoder is minimized for a particular execution of the target application. The temporal correlation of branch outcomes is used to improve cache performance by putting additional constraints on the basic block mapping. Code repositioning is accomplished with negligible increase in the static program size. The modification required to the program involves basic block motion, branch target updating, and branch insertion. The application-driven search for a low-power core and cache system requires usage of trace-driven cache simulation for each promising point considered in the design space. We attack these problems by scanning the space using search algorithms with conservative sharp lower and upper bounds and by providing powerful performance and power estimation techniques.

The developed synthesis tools synthesize a programmable application specific SOS which satisfies the requirements of multiple applications. This system requirement represents a realistic design scenario for most modern non-preemptive multitask application specific SOS (speech compression, RF signal processing, etc.). The synthesis technique considers each non-dominated processor core and competitive cache configuration, and selects the hardware which requires minimal power consumption and satisfies the performance requirements of all target applications.

System performance is evaluated using a platform which integrates simulation, modeling, and profiling tools as shown in Figure 2. SHADE is a tracing tool which allows users to define custom trace analyzers and collect rich information on runtime events. The executable binary program is dynamically translated into host machine code. The tool provides a stream of data to the translated code which is directly executed to simulate and trace the original application code. A custom analyzer is linked to SHADE to control and analyze the generated trace information. The analyzer sources relevant trace information from SHADE and builds a control flow graph (CFG) corresponding to the dynamically executed code. Details about the tracing process and collected data are elaborated in [Kir97]. Once the CFG is obtained, an algorithm is employed to reposition application basic blocks in such a way that instruction cache misses and cache decoder switching activity are minimized. Our experimentation uses a basic block relocation look up table to simulate the relocation of basic blocks in main memory. Stream of memory references is fed to a program that remaps the basic blocks using the relocation look-up table from the original into the optimized address space. The remapped trace of addresses, along with all unmodified data memory references, is sent to DINERO for cache simulation.

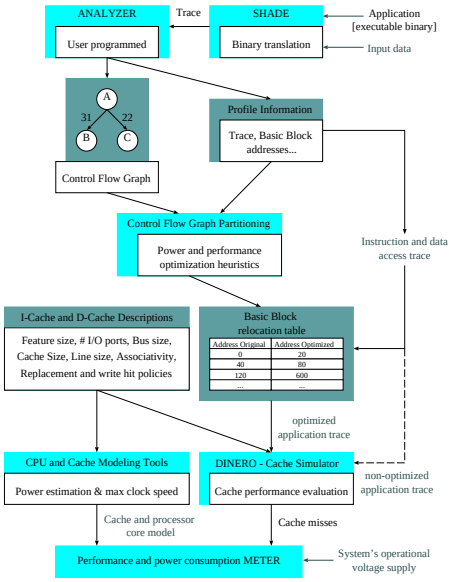

Figure 2: Synthesis and performance evaluation platform.

The final system performance was computed using the following formula:

$$CyclesPerInstruction = \frac{SystemClockFrequency}{MIPS} + (I - CacheMissRatio + D - CacheMissRatio) \cdot CacheMissPenalty$$

$X - CacheMissRatio$ was computed during the trace driven simulation of the cache subsystem. $CacheMissPenalty$, $MIPS$, and $SystemClockFrequency$ are system parameters introduced in Section 3. The system power consumption was computed as:

$$SysPowCons = ((TC - CPUIC) \cdot CPUEPC + CA \cdot EPCA + MA \cdot EPMA) \cdot Scale(VoltageSupply)$$

where TC is the total number of cycles required to execute the application, $CPUIC$ is the number of cycles while the CPU is idle waiting for data to be fetched from main memory to cache (blocking caches assumed due lower hardware cost), $CPUEPC$ is the average consumed energy per cycle for a particular processor core, CA and $EPCA$ are the number of cache accesses and energy consumed per access respectively, and MA and $EPMA$ are the number of main memory accesses (reads due cache misses and write-back updates) and energy consumed per memory access respectively. Function $Scale(VoltageSupply)$ returns the scaling factor for $VoltageSupply$ with respect to the nominal supply voltage (3.3V). The power savings due decrease of transitions in the cache decoder are encountered by scaling down 30% of the total cache power consumption with the ratio of the number of transitions due address change in the relocated and traditional basic block schedule.

5. SYNTHESIS OPTIMIZATION ALGORITHMS

The problems encountered in synthesis of a low power SOS and competitive optimization algorithms are described in the following subsections. First, the algorithm for basic block relocation is discussed. Then, based on the obtained code repositioning table and improved cache performance we assemble a power-efficient system configuration (core and cache structure) which satisfies performance requirements of a set of target applications.

5.1. Application to Cache Mapping for Low-Power

The core of the proposed application driven system level synthesis technique involves basic block repositioning based upon profile information. Block repositioning is undertaken in two phases. In the first phase the relocation aims for application execution on fixed hardware resources with minimal number of cache misses. The second phase relocates the part of code assigned to a particular cache line to a set of addresses which map to the same cache entry in a way that basic blocks executed frequently sequentially are stored at consecutive Gray code locations. In other words, sets of basic blocks mapped to the same cache line are reassigned to, not necessarilly, different cache entries such that the number of transitions in the instruction cache decoder is minimized.

The first phase in the relocation strategy is described in detail in [Kir97]. The collected run-time information on the execution of the program is used to generate and weight a control flow graph (CFG). Then, the CFG is transformed so that spatially and temporally highly probable execution paths are identified. Once the CFG is transformed, basic blocks are selected for mapping to particular addresses relative to the starting address of the program. The goal of the mapping function is to minimize the number of cache misses for the application, i.e. to map basic blocks that are likely to be executed sequentially into memory locations which result in mappings to different cache lines. Since the problem of finding the optimal basic block mapping is computationally intractable [Gar79], we opted to employ the heuristic presented in [Kir97].

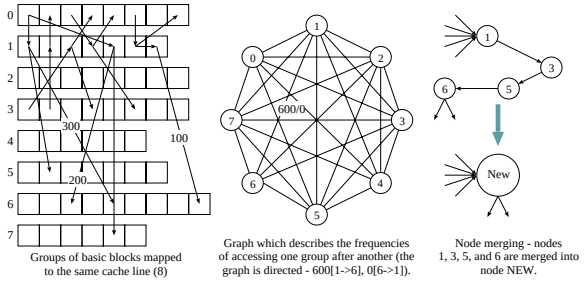

Figure 3: Modeling the problem of encoding each basic block group which maps to the same cache line.

After groups of basic blocks are identified, in the second phase of the optimization strategy we assign these groups to particular addresses. As shown in Figure 3 the problem is modeled using a graph representation. Groups of basic blocks which map to the same cache lines are represented as nodes in the graph. The weights of all edges from the CFG that connect basic blocks from two groups are summed. The sum is the weight of the directed edge connecting two vertices that represent the appropriate two groups of basic blocks. Obviously the number of nodes corresponds to the number of entries in the cache structure. The goal of the optimization algorithm is to assign codes to nodes in such a way that the sum products of the Hamming distance and directed weights between two nodes is minimal. The problem of embedding a general weighted graph, to minimize the weighted lengths, in an n-dimensional cube is NP-hard [Afr85]. Algorithms which address this optimization problem exist [Che97].

Repeat $\frac{Vertices}{SubsetCardinality}$ times
Randomly select a subset S of $SubsetCardinality$ vertices
Compute its objective function as:

$$ObjFunc(S) = \frac{\sum_{V_i \in S} {}^{E_{V_i, V_{i+1}}}}{\sum_{V \in S, W \notin S} {}^{(E_{V,W} \cdot Hamming(V,W))}}$$

where $Hamming(V, W)$ returns Hamming distance between V and W.
If W has not been assigned an address, $Hamming(V, W)$ returns one.
Repeat K times
Select two random nodes V (in) and W (outside) the current subset.
Replace V with W. Compute ObjFunc for the current subset.
Select the ordered subset S^+ with the greatest ObjFunc and assign $SubsetCardinality$ successive Gray-code addresses to vertices in S^+ in the order they are selected.
Merge all nodes in each subset into one node. The incoming edges into the starting and outgoing edges from the ending vertex in the parent subset the subset remain existing in the new graph G^*.
If $Cardinality(G^*) < M \rightarrow SolveExact(G^*)$ **else** RecursiveCall

Figure 4: Pseudo code for the encoding assignment procedure.

We have developed a novel probabilistic least-constraining most-constrained heuristic for minimum-switching state encoding. The heuristic is explained in detail using the pseudo-code presented in Figure 4. The algorithm iteratively randomly selects an ordered subset of nodes with the highest value for an objective function that guides the heuristic. The objective function (see Figure 4) takes into account the constraints of each considered subset of nodes. It forces early selection of a subset of vertices with the highest sum of edge weights (most-constrained) and smallest sum of edges connecting two nodes, one in, and the other outside the subset (least-constraining). The edges are summed corresponding to their order in the subset. Subset cardinality and the number of iterations in the randomized search for each objective-efficient subset are parameters of the heuristic. Once the subset is selected, it is assigned a set of consecutive Gray-code addresses and removed from the list of vertices considered in the subset selection inner shell. The vertices are not removed from the graph. The assigned encodings are used to add another component to the objective function described above. The objective function of an ordered subset is now scaled with a factor which quantifies how well the new subset fits the existing encoding assignments (see pseudo-code in Figure 4).

When all subsets are selected, the graph is transformed so that all nodes in one subset are merged into a single vertex (see Figure 3). This vertex inherits only the incoming edges of the initial and outgoing edges of the ending vertex of the parent subset. Although the graph has reduced size, the goal of the optimization strategy is still the same. Since the number of nodes in the graph is reduced to $\frac{Vertices}{SubsetCardinality}$ nodes, we either recursively apply the described procedure in order to further reduce the cardinality of the problem, or perform an exact algorithm.

5.2. Low-Power Resource Allocation

In this phase of the synthesis approach, a search is conducted for an energy-efficient system configuration and its voltage supply value which satisfy the performance requirements of the set of target applications. The search algorithm is described using the pseudo-code shown in Figure 6. Figure 5 demonstrates how competitive hardware configurations are traced for the optimal solution. The search algorithm evaluates number of cache systems

and uses a technique to reduce the search space of competitive processor-cache systems from a continuous to a discrete domain.

Since performance and power evaluation of a single processor, I- and D-cache configuration require trace-driven simulation, the goal of our search technique is to reduce the number of evaluated cache systems using sharp lower and upper bounds for cache system performance and power estimations. A particular cache system is evaluated using trace-driven simulation only once. The data retrieved from such simulation can be used for overall system power consumption estimation for different embedded processor cores with minor additional computational expenses.

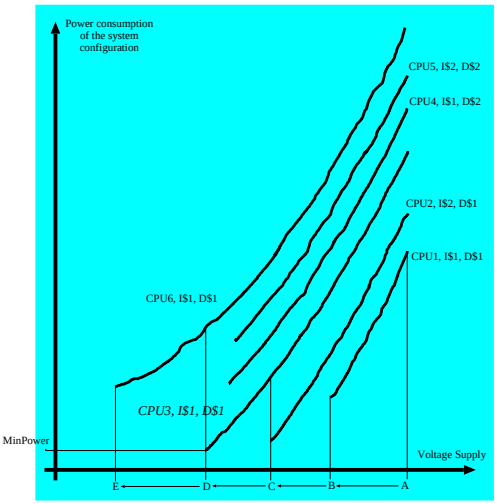

Figure 5: Search for the power optimal hardware configuration.

Firstly, the algorithm excludes from further consideration processors dominated by other processor cores. One processor type dominates another if it consumes less power at higher frequency and results in higher MIPS performance at the same nominal power supply. The competitive processors are then sorted in ascending order with respect to their power consumption per instruction and frequency ratio. Microprocessors which seem to be more power-efficient are, therefore, given priority in the search process. This step provides later on sharper bounds for search termination. Finally, at the nominal supply voltage, power consumption is estimated for the most power efficient processor combined with all cache configurations which satisfy the performance requirements of all target applications (Figure 5; point A).

The search for the most efficient cache configuration is bounded with conservative sharp lower and upper bounds. The sharp upper bound on cache increase for fixed voltage supply and processor core is determined by measuring the number of conflict misses and comparing the energy required to fetch the data from off-chip memory due measured conflict misses and the power that would have been consumed for twice as large cache and same number of cache accesses (number of cache conflicts assumed to be zero). In the former case we terminate further increase of the particular cache structure. Similarly, the lower bound is defined at the point when the energy required to fetch the data from off-chip memory due conflict cache misses for twice smaller cache with zero-energy consumed per cache access, is larger than the energy required for both fetching data from cache and off-chip memory in the case of the larger cache structure. The smallest cache system which meets the performance requirements of the set of target applications guarantees power consumption optimality for fixed processor and fixed supply voltage (Figure 5; point A; configuration $CPU1, I\$1, D\$1$).

After evaluating all competitive hardware configurations, we obtain the cache-processor configuration that represents the power-optimal solution at the nominal voltage supply (in our experimentations 3.3V). The voltage supply can now be reduced until a point where the current best configuration does meet the applications' timing constraints. The key step in the search mechanism is translating the search domain for the voltage supply from continuous into discrete scope. As shown in Figure 5 the curves that represent power consumption behavior for a single system configuration are congruent with respect to the voltage supply change. This means that a configuration which gives the best energy savings for a specific voltage continuously has the best results in the domain of its definition, i.e. until able to satisfy the application constraints. According to this fact, we compare hardware configurations only at the discrete points of discontinuity of curves which quantify configurations' power analysis. In Figure 5, the point of discontinuity of the configuration selected at point A is point B. Point B represents the first solution (configuration and voltage) candidate.

In order to search for a configuration which results in lower power consumption, at point B, we estimate the power characteristics of all application feasible configurations (which do not have the processor core from the first candidate) and select the one with the best performance ($CPU2, I\$2, D\$1$). However, since we already have the current best solution (from previous processor core evaluations, $CPU1, I\$1, D\$1$) the bounds are now even sharper. Namely, we set the final upper and lower bound on the cache structure size and organization as follows:

- **Upper bound.** Given: Feasible cache structure (I- and D-cache), and processor core A. We terminate further increase of the cache structure when the increase of the power consumption due larger cache (although cache conflicts total zero) would be larger than the energy consumed by the current best solution.
- **Lower bound.** Given: Feasible cache structure (I- and D-cache), and processor core A. We abort further decrease of the cache structure if the amount of energy required to bring the data due additional cache misses from off-chip memory (although the smaller cache power consumption totals zero) is larger than the energy consumed by the current best solution.

$VoltageSupply = 3.3V$
Sort processor cores in a list L according to $\min(\frac{EnergyPerInstruction}{Frequency})$
Delete the dominated processor cores from L
Repeat
  For I-cache = 512B..32KB and CacheLineSize = 8B..512B
    For D-cache = 512B..32KB and CacheLineSize = 8B..512B
      Check upper, lower bounds; if exceeded **break**;
      If (I- and D-cache configuration has never been evaluated)
        Evaluate power consumption of the cache structure
      According to the existing cache system analysis evaluate the
      power consumption of the entire system (with the processor core)
      Memorize Configuration C if power consumption is minimal
  Scale down the $VoltageSupply$ until C does not satisfy the constraints.
  Delete the processor core $PC \in C$ from the list of cores L
until L is not empty.

Figure 6: Pseudo code for the resource allocation procedure.

The algorithm iterates the process of scaling down the voltage supply of the current best configuration until it can satisfy the deadlines. It consecutively records the best configuration and the appropriate voltage with different processor core. These iterations are repeated until the last application-feasible processor core's configuration is evaluated for power. This configuration has the ability to satisfy the application constraints at the lowest voltage supply (point E; configuration $CPU6, I\$1, D\$1$). However, this property does not guarantee the best overall power sav-

ings. In Figure 5, the most power-efficient configuration found is $CPU3, I\$1, D\$1$ at point D.

6. EXPERIMENTAL RESULTS

The effectiveness of our synthesis approach is demonstrated on a set of applications and input data from the MediaBench suite lee97], and a set of hard real-time constraints. The results are shown in Table 4. The upper part of the table presents the configurations which were found to be the most power-effective for a particular benchmark. For each application and its timing constraint, the winning configuration is described by specifying its processor type, the associated I- and D-cache structures (cache sizes followed by cache line size). These properties of the system are shown in the third column. The next column quantifies the power dissipation of the system for execution of one iteration of the application. The last column points to the percentage improvement with respect to the average configuration among all the configurations minimal at the discrete points of system evaluation (see Figure 5; points A, B, C, D, and E). The energy efficacy for an average configuration is presented in the lower part of Table 4.

		Our synthesis approach		
Appl.	TC	CPU:I-Cache:D-Cache	Energy	%
JPEGenc	1.8s	M68000:1KB/64B:2KB/64B	8.93e+06	8
JPEGdec	0.08s	LSITR4101:1KB/128B:4KB/64B	3.13e+06	31
EPIC	1.0s	LSITR4101:512B/64B:4KB/64B	3.39e+07	32
MipMap	0.85s	LSITR4101:512B/32B:4KB/32B	4.92e+07	243
OSDemo	0.35s	LSICW4001:1KB/64B:2KB/32B	7.35e+06	11
TexGen	1.0s	LSITR4101:1KB/64B:16KB/32B	1.22e+08	464
GSMenc	3.5s	M68000:1KB/128B:2KB/64B	1.01e+08	21
UnEPIC	0.33s	LSICW4001:1KB/64B:4KB/64b	5.88e+06	14
GSMdec	0.3s	PowerPC403:512B/64B:1KB/128B	1.00e+10	35
		Average minimal configuration		
JPEGenc	1.8s	ARM7LP:512B/64B:1KB/64B	9.66e+06	
JPEGdec	0.08s	LSI TR4101:512B/32B:4KB/64B	4.10e+06	
EPIC	1.0s	LSI TR4101:512B/32B:4KB/64B	4.45e+07	
MipMap	0.85s	ARM7LP:512B/64B:2KB/128B	1.11e+8	
OSDemo	0.35s	LSI TR4101:512B/64B:1KB/64B	8.16e+06	
TexGen	1.0s	PowerPC 403:1KB/32B:1KB/128B	5.70e+08	
GSMenc	3.5s	LSI TR4101:1KB/64B:1KB/64B	1.22e+08	
UnEPIC	0.33s	LSI TR4101:512B/64B:512B/32B	6.68e+06	
GSMdec	0.3s	M68000:512B/32B:512B/128B	1.35e+10	

Table 4: Experimental results: comparison of our synthesis approach to a greedy strategy.

The approach results in quantitatively diverse improvements depending on the actual applications and selected deadline. We opted to test the approach for strict as well as relaxed deadlines. Note that the highest improvements in the power dissipation occurred in cases where the number of cache misses was significantly reduced using the basic block relocation compilation technique (this heuristic has been experimented in [Kir97]). On the other hand the heuristic used to minimize the transitions in the cache decoder resulted in average 35% less estimated power dissipation in the I-cache decoder with negligible standard deviation. The percentage of power dissipation of the overall system due the I-cache accesses ranged from 21% to 44%. For a single application and fixed timing constraint, the resulting power consumptions for various configurations appeared to cluster at particular values in the range of one to three orders of magnitude. Therefore, the power consumption results have varying improvements from 8% to 464% depending on the number and range of configuration clusters.

7. CONCLUSION

We developed algorithms for system-level power minimization of instruction cache misses, instruction and data cache size and organization selection, placement of frequently executed sequential basic blocks of code in consecutive Gray code addressed memory locations, and power-efficient processor and cache matching with the applications. The compilation performance and power optimization algorithms are engine for synthesis of core-based low power system-on-silicon. The new synthesis platform integrates the existing modeling, profiling, and simulation tools with the developed system-level synthesis tools. The effectiveness of the approach is demonstrated on a variety of modern industrial-strength multimedia and communication applications.

8. REFERENCES

[Afr85] F. Afrati, et al. The complexity of Cubical Graphs, Information and Control, Vol. 66, pp. 53-60, 1985.

[Ber94] B.N. Bershad, et al. Avoiding conflict misses dynamically in large direct-mapped caches, 6th ASPLOS, Vol.29, No.11, pp. 158-70, 1994.

[Bur96] T.D. Burd, R.W. Brodersen. Processor design for portable systems, Journal of VLSI Signal Processing, Vol.13, No.2-3, pp. 203-221, 1996.

[Cha92] A.P. Chandrakasan, et. al. Low-power CMOS digital design, IEEE Journal of Solid-State Circuits, Vol.27, No.4, pp.473-484, 1992.

[Che97] De-S. Chen, M. Sarrafzadeh. Cube-embedding based state encoding for low power design, ASP-DAC, pp. 613-618, 1997.

[Cme94] B. Cmelik, D. Keppel. Shade: a fast instruction-set simulator for execution profiling, SIGMETRICS, Vol.22, No.1, pp. 128-37, 1994.

[Eva95] R.J. Evans, P.D. Franzon. Energy consumption modeling and optimization for SRAMs, IEEE Journal of Solid-State Circuits, Vol.30, No.5, pp. 571-579, 1995.

[Fly96] M.J. Flynn. Computer Architecture: Pipelined and Parallel Processor Design, Jones and Bartlett, Boston, MA, 1996.

[Fro97] R. Fromm, et al. The energy efficiency of IRAM architectures, 24th ISCA, pp. 327-337, 1997.

[Fur96] K. Furumochi, et al. A 500 MHz 288 kb CMOS SRAM macro for on-chip cache, IEEE Intl. Solid-State Circuits Conference, pp. 156-157, 1996.

[Gaj94] D.D. Gajski, et al. A System-Design Methodology: Executable Specification Refinement, Euro-DAC, pp. 458-463, 1994.

[Gar79] M.R. Garey, D.S. Johnson. Computers and intractability: a guide to the theory of NP-completeness, W. H. Freeman, San Francisco, CA, 1979.

[Gon96] R. Gonzalez, M. Horowitz. Energy dissipation in general purpose microprocessors, Journal of Solid-State Circuits, Vol.31, No.9, pp. 1277-1284, 1996.

[Gup93] R.K. Gupta and G. De Micheli. Hardware-software cosynthesis for digital systems, Design and Test of Computers, Vol.10, No.7, pp. 29-41, 1993.

[Hil88] M.D. Hill. A case for direct-mapped cache, Computer, pp. 25-40, 1988.

[Hwu89] W.W Hwu, P.P Chang. Achieving high instruction cache performance with an optimizing compiler, 16th ISCA, pp. 242-251, 1989.

[Jou93] N.P. Jouppi. Cache write policies and performance, 20th ISCA, pp. 191-201, 1993.

[Kir97] D. Kirovski, et al. Application-driven synthesis of core-based systems, to appear, ICCAD, pp. 104-107, 1997.

[Ko95] U. Ko, P.T. Balsara. Characterization and design of a low-power, high-performance cache architecture, VLSI Technology, Systems, and Appl., pp. 235-238, 1995.

[Lee97] C. Lee, et al. MediaBench: A Tool for Evaluating and Synthesizing Multimedia and Communications Systems, to appear, Micro-30, 1997.

[Lee96] H.-D. Lee, et al. A novel high level synthesis algorithm for low power ASIC design, Journal of Microelectronic Systems Integration, Vol.4, No.4, pp. 219-232, 1996.

[Meh96] R. Mehra, L.M. Guerra, J.M. Rabaey. Low-power architectural synthesis and the impact of exploiting locality, Journal of VLSI Signal Processing, Vol.13, No.2-3, pp.239-258, 1996.

[Mic97] Microprocessor Report, all isues, 1997.

[Nag95] M. Nagamatsu, et al. A 150 MIPS/W CMOS RISC processor for PDA applications, Intl. Solid-State Circuits Conference, pp. 114-115, 1995.

[Pan96] P. Panda, N. Dutt, A. Nicolau. Memory organization for improved data cache performance in embedded processors, 9th International Symposium on System Synthesis, pp. 90-95, 1996.

[Pot95] M. Potkonjak, W.H. Wolf. Cost Optimization in ASIC implementation of Periodic Hard-Real Time Systems using Behavioral Synthesis Techniques, ICCAD, pp. 446-451, 1995.

[Su95] C.-L. Su, A.M. Despain. Cache design trade-offs for power and performance optimization, Intl. Symposium on Low Power Design, pp. 63-68, 1995.

[Tor96] C. Xia, J. Torrellas. Instruction prefetching of systems codes with layout optimized for reduced cache misses, 23rd ISCA, pp. 271-82, 1996.

[Wad92] T. Wada, et al. An analytical access time model for on-chip cache memories, IEEE Journal of Solid-State Circuits, Vol.27, No.8, pp.1147-1156, 1992.

[Wil94] S.J.E. Wilton, N.P. Jouppi. CACTI: an enhanced cache access and cycle time model, IEEE Journal of Solid-State Circuits, Vol.31, No.5, pp. 677-688, 1996.

[Wol94] W.H. Wolf. Hardware-Software Co-Design of Embedded Systems, Proc. of the IEEE, Vol.82, No.7, pp. 967-989, 1994.