SYNTHESIS OF DSP SOFT REAL-TIME MULTIPROCESSOR SYSTEMS-ON-SILICON

Darko Kirovski and Miodrag Potkonjak

Computer Science Department, University of California, Los Angeles, CA 90095-1596, USA

ABSTRACT

The recent convergence of applications (Internet and embedded applications) and technology (reuse and very high integration level) trends resulted in a strong need for design of soft real-time DSP systems on silicon. We developed a new hierarchical modular approach for synthesis of area efficient soft real-time DSP systems on silicon. This synthesis strategy employs a number of optimization intensive scheduling, performance monitoring, and allocation steps. The backbone of the optimization approach is a novel on-line scheduling algorithm which uses meta-algorithmic techniques for on-the-fly heuristic selection and parameter tuning. Resource allocation refers to a predetermined lower-bound system performance, to perform a branch-and-bound resource allocation search for an area-efficient multiprocessor configuration where each processor has local instruction and data cache.

In order to bridge the gap between the profiling, modeling, and synthesis tools of the two traditionally independent synthesis domains (architecture and CAD), we develop a new synthesis and evaluation platform which integrates the existing modeling, profiling, and simulation tools with the new developed system-level synthesis tools. The effectiveness of the approach is demonstrated on the industrial strength MediaBench benchmark suite.

1. INTRODUCTION

The recent convergence of DSP, multimedia, and communication applications has emerged intensive signal processing and high volume data management application-specific systems. Due to the ever-changing communication and signal processing standards, most of these systems are preferably programmable. The immense growth of various Internet applications, such as video/audio format conversions and streaming, Internet telephony, video/audio data retrieval, etc. has created the high demand for embedded soft real-time systems. The trend of increasing application processing volume and available task-level parallelism has made multiprocessor DSP systems a must [Olu97]. For example, most ISDN, ADPCM, and 830020 cards address the issue of remote LAN access, computer integrated telephony, RF signal processing, video conferencing, bandwidth management, disaster recovery, etc.

To address the semiconductor technology trends and the need for programmable DSP multiprocessor platforms, we developed a new hierarchical modular approach for synthesis of area-minimal DSP server systems-on-silicon (SOS). A typical multiprocessor core-based system consists of a number of processor systems where each system comprises a programmable core, instruction and data cache cores, and a number of hardware DSP accelerators and control blocks as depicted in Figure 1. The role of controlling the DSP system is dedicated to one of the processing elements (master), while the other processors are executing tasks in the slave mode. Numerous application-specific systems' manuals and documents outline that most of the SOS area is dedicated to the processor cores and associated caches [Mic97]. Therefore, the synthesis strategy employs a branch-and-bound search for an SOS with minimal chip area dedicated to the programmable and cache cores. The resulting configuration is required to satisfy a predetermined set of system processing throughput requirements with respect to an expected input datastream. In order to improve the number of serviced soft real-time requests, we developed a novel heuristic for task on-line scheduling which employs meta-algorithms for on-the-fly parameter adjustment.

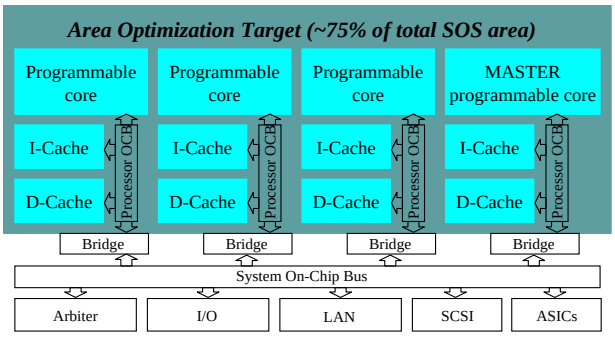

Figure 1: A typical multiprocessor SOS.

In order to bridge the gap between the profiling and modeling tools from the two traditionally independent synthesis domains (architecture and CAD), we developed a new synthesis and evaluation platform. We used SHADE [Cme94] as an instruction-set, and DINEROIII [Hill89] as a cache simulator for trace-driven system simulation to accurately evaluate performance of cache subsystems for the MediaBench suite [Lee97]. The Stanford on-line Cache Design Tool (CDT) was used to estimate cache access latency and area based on properties such as cache total size, line size, semiconductor feature size, replacement policy, and associativity [Fly96]. Data extrapolated from the available literature for commercial programmable cores was used as a simplified model to approximate processor core performance with respect to its area. The effectiveness of the approach is demonstrated on the industrial strength MediaBench benchmark suite.

2. PREVIOUS WORK

The related work can be traced through areas of application-specific system optimization, processor and memory hierarchy design and evaluation, and soft real-time systems.

SOS is becoming an important focus area for both research and commercial developers [Bol94]. Shortened design cycles, due to market pressure, have encouraged usage of predesigned processor cores. Market pressure to reduce system cost for consumer products has spurred the development of system level synthesis techniques [Wol94]. As embedded applications have become more sophisticated and commercially relevant, hardware-software codesign and techniques for system-level synthesis have also become

increasingly important [Gup93, Gaj94].

The increased interest in embedded system design with reusable core components has encouraged the development of high-level ASIC architecture evaluation models. For example, The Microprocessor Report presents a monthly summary of the area and performance of numerous commercial processors [Mic97]. Instruction and data caches, as the highest level of memory hierarchy, have been thoroughly studied [Hill89, Jou93].

The conceptual advantages and disadvantages of hard, soft, and hybrid real-time systems have been described in [Rze94]. The development of scheduling techniques for soft real-time systems has been reported in [Kao93, Kao94]. The performance implications of using different static processor allocation strategies in serving soft real-time requests has been studied in [Car94]. Implementation issues of various scheduling heuristics at the operating system level have been discussed in [Ade94].

3. PRELIMINARY DISCUSSION

In this section we describe the hardware performance models for caches and processor cores. Three factors combine to influence the execution performance of a processing element: cache miss rate, CPU performance, and CPU clock speed. The approach that we use here is to leverage existing models to estimate the area and performance of both cache and processor cores. This approach allows the synthesis approach to be rapidly updated and applied to new environments with new technology.

The CDT is used to evaluate the impact of cache design choices on area and access time. Caches typically found in current embedded systems range from 128B to 32KB. Since higher associativity results in significantly higher access time, here we consider only direct mapped, and 2-way set associative caches. We use cache line size fixed to 32 bytes. This decision attempts to eliminate the well known cache penalty tradeoff problem. Large cache lines generally result in increased latency when fetching from main memory, while short cache lines increase access latency due to greater control hardware. We estimate the cache miss penalty based on the operating frequency of the system, external bus width, and clock for each system investigated. This penalty ranges between 4 and 20 system clock cycles. Write-back was adopted as oppose to write-through, since it is proven to provide superior performance [Jou93] though at increased hardware cost. Each of the processors considered is constrained by the Flynn limit, and thus is able to issue at most a single instruction per clock period. Thus, the caches were designed to have a single access port. A sample overview of the estimated cache model is presented in Table 1. Cache access latency and area is computed for a number of organizations and sizes, all with implementation feature size fixed at 0.5 μm and typical six transistors per CMOS SRAM cell.

Associativity	Area Latency	Cache size 1KB	2KB	4KB	8KB	96KB
Direct Mapped	mm^2	2.11	3.81	7.20	13.96	161.7
	ns	3.79	3.97	4.16	4.51	6.79
2-way	mm^2	2.38	4.02	7.29	13.80	156.1
	ns	5.67	5.75	5.94	6.23	8.54
4-way	mm^2	3.04	4.65	7.85	14.25	154.0
	ns	6.04	6.24	6.34	6.58	8.83

Table 1: A sample of the cache area and latency data.

Microprocessor performance and area information was culled from datasheets as well as from the CPU Center Info web site [Cpu]. A sample of the collected data is presented in Table 2. The area dedicated to the cache subsystem in a programmable embedded processor can be observed from the last two rows in Table 2, where data for two commercial chips is presented.

Microprocessor	Clock MHz	MIPS	Feature μm	Area mm^2
StrongARM	233	266	0.35	4.3
ARM, 7	40	36	0.6	5.9
ARM, 9 TDMI	150	165	0.35	4
LSI Logic, TR4101	80	30	0.35	2
LSI Logic, CW4001	60	53	0.5	3.5
LSI Logic, CW4011	80	120	0.5	7
DSP Group, Oak	80	80	0.6	8.4
NEC, R4100	40	40	0.35	5.4
Toshiba, R3900	50	50	0.6	15
Motorola, M-core	50	48	0.35	2.2
ARM 710 (ARM7 / 8KB)	40	72	0.6	34
SA-110 (StrongARM / 32KB)	200	230	0.35	50

Table 2: A sample of the processor performance vs. area data.

The task model that we adopt is described using the following assumptions. First, there exists a limited number of task occurrences in the system. Upon arrival $a(T)$, each task T is assigned a real-time constraint, deadline $d(T)$. The available time for execution of the task, $d(T)-a(T)$, is equal or greater than the time $r(T)$ required to execute the task on the fastest processing element (PE). The execution slack for each task $s(T)$ equals $d(T)-a(T)-r(T)$. Each task type is statically partitioned into sequentially dependent subtasks where each subtask can be executed on a different PE. A task is serviced if all its subtasks are serviced. System inability to service a task is acceptable. Task arrival and system load are unpredictable. Due to high context-switching overhead, subtask execution on a PE is non-preemptive. The goal is to minimize the number of missed deadlines in the system. Such soft real-time system is typical for many communication and financial transaction applications [Ade94].

4. THE NEW SYSTEM LEVEL SYNTHESIS APPROACH

Figure 2 illustrates the synthesis framework. In this section we describe the role of each optimization and simulation module and how modules are combined into a tightly integrated synthesis system. The branch-and-bound search for the area-minimal multiprocessor configuration iteratively generates different CURRENT architectures. The performance of each PE in CURRENT is evaluated using the simulation platform which consists of processor and cache models, SHADE and DINEROIII simulators, and MediaBench executable and data inputs. The performance of each PE in CURRENT is fed to the core system performance optimization process: the on-line soft task scheduling algorithm. The developed algorithm, having the system architecture, individual performance of each PE, benchmark input data stream, and required soft task service rate, evaluates whether the CURRENT application satisfies the soft real-time system requirements. The report on CURRENT's performance estimation is returned to the resource allocation algorithm which augments this information in the pessimistic bounds of the search engine. The conveyed report comprises information on the unsuccessful soft task services by each PE.

System performance is evaluated using a platform which integrates simulation, modeling, and profiling tools. SHADE is a tracing tool which allows users to define custom trace analyzers and thus collect rich information on runtime events. SHADE dynamically translates the executable binary program into host machine code. The tool also provides a stream of data to the translated code which is directly executed to simulate and trace the original application code. We use SHADE to trace memory references while

executing code from the MediaBench suite. The stream of references is piped to DINERO for cache simulation for each PE.

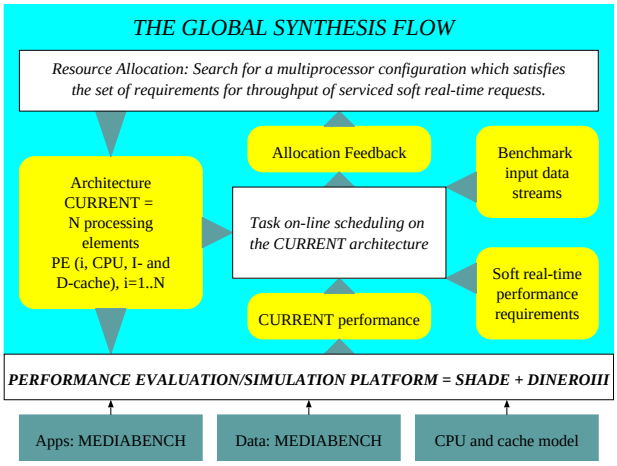

Figure 2: The hierarchical modular synthesis approach.

The final PE performance is computed using the following formula:

$$CyclesPerInstruction = \frac{SystemClockFrequency}{MIPS} + CacheMissRatio \cdot CacheMissPenalty$$

where $CacheMissRatio$ was computed during the trace driven simulation of the cache subsystem. $CacheMissPenalty$, $MIPS$, and $SystemClockFrequency$ are system parameters introduced in Section 3.

5. SYNTHESIS OPTIMIZATION ALGORITHMS

The problems encountered in synthesis of a multiprocessor DSP single chip system and competitive optimization algorithms are described in detail in the following subsections. First, the problem of on-line servicing of soft real-time tasks is recognized, its complexity is established, and an efficient heuristic is proposed. Second, the resource allocation algorithm which performs a branch-and-bound search for an area-minimal configuration that satisfies the system service throughput requirements is elucidated.

5.1. On-Line Soft Real-Time Task Scheduling

In order to improve the service rate of a multiprocessor system for a given, expected, input datastream, we developed a novel on-line soft real-time task scheduling algorithm. The algorithm targets on-line scheduling of the task model described in Section 3. The optimization problem can be formulated using the standard Garey-Johnson format [Gar79]:

Problem: On-Line Soft Real-Time Task Scheduling.

Instance: Given a set of k heterogeneous processing elements $PE_i, i = 1..k$, a set of n tasks $T_j, j = 1..n$, where each task T_j contains n_j subtasks $S_p^j, p = 1..n_j$ which have execution times $r(S_p^j, PE_i)$ when executed on PE_i. Input stream of tasks $I(T_m, a(T_m), d(T_m))$ defined with arrival $a(T_m)$ and deadline $d(T_m)$ times for each task T_m.

Question: Is there a dynamic schedule such that all subtasks of all tasks in the input stream are serviced by $PE_i, i = 1..k$?

The On-Line Soft Real-Time Task Scheduling problem is NP-complete, since there is one-to-one mapping between the its special case and the Multiprocessor Scheduling problem ([Gar79]), where each task contains one subtask, arrival times of all tasks are known at all times, and the set of PEs is homogeneous. We have developed a heuristic solution for dynamic subtask scheduling on a heterogeneous set of PEs, which employs weighted sum of several heuristics to produce a performance superior, champion heuristic. The existing heuristic approaches to this problem select for execution the pending subtask of all pending tasks which has the smallest [Kao93]:

- Ultimate Deadline. The deadline of each subtask is equal to the deadline of the parent task.
- Effective Deadline (ED). The deadline of a subtask is equal to the difference of the parent task's deadline and expected remaining execution time of the task.
- Equal Slack (ES). The remaining slack is divided equally among the remaining subtasks.
- Equal Flexibility (EF). The remaining slack is divided among the remaining subtasks proportionally to their execution times.

For (;;)
  If PE_i is idle
    For each subtask S_T to execute of each pending task T
      Compute $OF(T, S_T) = \alpha ED(S_T) + \beta ES(S_T) + \gamma EF(S_T) + \delta TE(T) + \epsilon SE(T)$
      where the execution time $r(S_T)$ used in the heuristics corresponds to PE_i
    Schedule the task with the smallest $OF(T, S_T)$

Figure 3: Pseudo code for on-line soft real-time task scheduling.

The implementation and effectiveness of these approaches are described in [Kao93]. We employ two new decision-making heuristic functions:

- Time to Execute (TE). The remaining task execution time.
- Subtasks to Execute (SE). The remaining task's subtasks.

In Section 6, we prove that the influence of these heuristics significantly improves the system servicing throughput. The dynamic scheduling heuristic is described using the pseudo-code in Figure 3. Note that all parameters of the algorithm objective function are statistically determined and validated at run-time using the metaalgorithmic parameter evaluation strategy introduced in [Kir97].

5.2. Resource Allocation

The outer shell of the synthesis approach is a branch-and-bound search loop which generates a number of multiprocessor configurations and selects an area-minimal configuration. The configurations should be capable of satisfying the soft real-time service throughput requirement with respect to the benchmark input data stream. The resource allocation algorithm is described in detail using the pseudo-code in Figure 4.

The outer loop of the branch-and-bound search iteratively increases the number of processing elements, PES. For each PES value, a set of all microprocessor configurations which can potentially satisfy the soft real-time service (SRTS) requirements is generated. A microprocessor configuration can potentially satisfy the SRTS requirements when each microprocessor is assigned an infinite cache structure, is assumed not to be idle at any time, is assumed to execute only subtasks of, later on, completed tasks, and does not take more area than the current best solution with the smallest available I- and D-caches. For each microprocessor configuration, an exhaustive search is performed for the smallest multiprocessor-cache structure which satisfies the SRTS requirements. While searching through the space of all cache structure variations, individual I-caches are increased exponentially. Since the function that describes a cache miss ratio monotonically decreases with the increase of the cache size, a binary search on the exponential scale of the corresponding D-cache size is employed. The search is bounded at any time with lower and upper

pessimistic bounds which determine dominated solutions. Dominated solutions are found according to the following rules:

- For a set of programmable cores A, the best cache subsystem so far recorded for A of Q bytes, and fixed I-cache of P bytes, we do not evaluate D-caches larger than $Q - P$ bytes.
- When the best core-cache configuration totals $R\ mm^2$ and a set of cores of area X, we terminate the search whenever the sum of the I-caches and D-caches exceeds $R - X\ mm^2$.

PES = 2
While
  For each potentially satisfiable processor configuration
    For each potentially satisfiable cache configuration
      where for each $I\_cache :: size = 128B..maxB, sets = 1..2$
      do Binary Search for D_cache of minimal size such that the
      configuration $(core, I\_cache, D\_cache)$ satisfies SRTS requirements.
  PES = PES + 1
until the area dedicated only to the potentially satisfiable processor configuration is smaller than the area of the current best solution.

Return the overall configuration of minimal total area.

At any point during the algorithm, search is terminated along paths which are dominated by any other already pessimistically evaluated solution.

Figure 4: Pseudo code for the resource allocation procedure.

6. EXPERIMENTAL RESULTS

We present experimental results for both the on-line soft real-time scheduling heuristic and our resource allocation algorithm. To emulate realistic traffic soft real-time system environment we used traffic traces from public networks [Duk] and multimedia applications from the MediaBench benchmark suite [Lee97] partitioned into logic subtasks. In Table 5 a comparison of a random (R), round robin (RR), ED, EF, and our on-line soft real-time scheduling heuristic is shown. Columns 2-7 show the percent of serviced tasks with respect to a particular task arrival rate. Note that the meta-algorithmic approach resulted in consistently better scheduling with respect to all other approaches. In column eight, the averaged improvements are presented when meta-algorithmics scheduling was used with respect to the other scheduling approaches.

Algorithm	Task arrival rate						Impr.
	1	1.5	2	2.5	3	3.5	
R	0.95	0.66	0.31	0.16	0.12	0.10	
RR	0.92	0.60	0.33	0.22	0.16	0.13	
ED	0.89	0.68	0.48	0.37	0.29	0.24	
EF	0.90	0.71	0.55	0.46	0.40	0.35	
Meta	1.00	0.83	0.64	0.53	0.45	0.40	
Algorithm	Task arrival rate						
	4	5	6	7	8	10	
R	0.07	0.03	0.01	0.00	0.00	0.00	76%
RR	0.10	0.07	0.10	0.06	0.04	0.04	68%
ED	0.21	0.16	0.13	0.11	0.09	0.07	49%
EF	0.32	0.27	0.23	0.21	0.19	0.16	29%
Meta	0.36	0.31	0.27	0.24	0.21	0.18	-

Figure 5: Comparison of five soft real-time scheduling heuristics.

Note that for a given on-line soft real-time scheduling, the resource allocation algorithm results in optimal solution. Therefore, we present a sample configuration which satisfies the soft real-time requirements for traffic traces at [Duk] and all applications from the MediaBench suite where each task was given deadline twice as long as the execution time of that task on the slowest core. Table 3 shows the processor core configuration with cache strucutres for each processor and the allocated total area.

7. CONCLUSION

The increasing popularity of Internet and communications applications has resulted in high demand for DSP soft real-time server systems. We developed a methodology for synthesis of multiprocessor soft real-time DSP systems. The approach relies on an efficient meta-algorithmic based heuristic for on-line scheduling of tasks with soft timing constraints. The developed resource allocation algorithm employs lower and upper pessimistic bounds to reduce the solution search space. The efficacy of the approach was tested on a set of real-life traffic traces and industry strength multimedia benchmark applications.

Processor Core	I-Cache	D-Cache
$3\cdot$ {LSI Logic CW4001}	each with 1KB	each with 1KB
$+\ 2\cdot$ {LSI Logic TR4101}	both with 512KB	both with 1KB
+ Motorola M-core	512B	1KB
+ StrongARM	1KB	2KB
Total system area		52.0 mm^2

Table 3: Area-minimal system configuration for a given input data stream and set of applications application.

8. REFERENCES

[Ade94] B. Adelberg, H. Garcia-Molina, B. Kao. Emulating soft real-time scheduling using traditional operating system schedulers. Real-Time Systems Symposium, pp. 292-8, 1994.

[Bol94] I. Bolsens, K. Rompaey, H. De Man. User requirements for designing complex systems on silicon. VLSI Signal Processing VII, pp. 63-72, 1994.

[Car94] B.M. Carlson, L.W. Dowdy. Static processor allocation in a soft real-time multiprocessor environment. IEEE Trans. on Parallel and Distributed Systems, Vol.5, No.3, pp. 316-20, 1994.

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

[Cpu] http://infopad.eecs.berkeley.edu/CIC/

[Duk] http://www.ncstate.net/nts/research/traffic/

[Fly96] M.J. Flynn. Computer Architecture: Pipelined and Parallel Processor Design. Jones and Bartlett, 1996.
(http://umunhum.Stanford.EDU/tools/cachetools.html)

[Gaj94] D.D. Gajski, et al. A System-Design Methodology: Executable Specification Refinement. Euro-DAC '94, 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, 1979.

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

[Hill89] M.D. Hill, A.J. Smith. Evaluating Associativity in CPU Caches. IEEE Trans. on Computers, pp.1612-1630, 1989.

[Jou93] N.P. Jouppi. Cache write policies and performance. 20th ISCA, Vol.21, No.2, pp. 191-201, 1993.

[Kao93] H. Kao, H. Garcia-Molina. Deadline assignment in a distributed soft real-time system. International Conference on Distributed Computing Systems, pp. 428-37, 1993.

[Kao94] B. Kao, H. Garcia-Molina. Subtask deadline assignment for complex distributed soft real-time tasks. International Conference on Distributed Computing Systems, pp. 172-81, 1994.

[Kir97] D. Kirovski, M. Potkonjak. System level synthesis of low-power hard real time systems. 34th DAC, pp.697-703, 1997.

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

[Mic97] The Microprocessor Report, all isues, 1997.

[Olu97] K. Olukotun, et al. The case for a single-chip multiprocessor. SIGPLAN Notices, Vol.31, No.9, pp.2-11.

[Rze94] H. Rzehak. Hard, soft, real-time systems and their use. Real Time Computing, pp. 604-5, 1994.

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