Heuristic Techniques for Synthesis of Hard Real-Time DSP Application Specific Systems

Miodrag Potkonjak
Dept. of Computer Science
University of California
Los Angeles, CA 90095

Wayne Wolf
Dept. of Electrical Engineering
Princeton University
Princeton, NJ 08540

ABSTRACT

We introduce an approach for design and optimization of ASIC implementations which realize multiple computational tasks under hard real-time constraints. The approach designs a multitask ASIC by combining techniques from hard real-time scheduling and behavioral synthesis. The key component of the methodology is successive multi-resolution synthesis technique. The technique starts from an incompletely specified preliminary solution and uses interchangeably operating systems and behavioral synthesis tools to derive increasingly more detailed and complete design solution. The effectiveness of the optimization algorithms is demonstrated on several multiple task designs.

1. INTRODUCTION

We have developed behavioral synthesis algorithms for the creation of **multi-task application-specific systems**. By using information provided by **hard-real time scheduling** methodologies and **VLSI DSP behavioral synthesis** tools, we connect the synthesis process to operating systems methodologies and enable **efficient sharing of hardware by several tasks**.

We target hardware design problem for systems of processes with deadlines is specified as a set of periodic **tasks**. Each task is defined using control-data flow graph and the set of timing constraints. For each task three timing constraints are imposed: **period interval**, the **start time**, and the **finish time**. In our implementation, we assumed that the end of sampling interval period of each task is its finish time.

The synthesis goal is to partition the set of tasks in an arbitrary number of subsets so that each subset can be implemented on one dedicated multifunctional ASIC. The partitioning is conducted in a such a way that the area of the ASIC is minimized. We developed a search strategy which partitions the tasks into groups. Each partition (subset of tasks) is implemented by a single datapath/controller machine; tasks are executed one at a time on the datapath and preemption is not allowed in order to avoid time-consuming context switching overhead. The search strategy is guided by a simple and fast estimation procedure which predicts the required hardware and corresponding time resources for each task.

2. PRELIMINARIES

Our task-level computational model assumes that for each task the sampling rate is specified, that the deadline for each task is the end of its sampling period, a single thread of control, and no-preemption is allowed.

We use different techniques as synthesis progresses to obtain progressively more accurate (and expensive) estimates. At the task level, rate-monotonic scheduling is used for estimation purposes. Rate-monotonic scheduling (RMS) theory addresses the problem of ensuring that independent periodic tasks are scheduled without violation of the associated timing constraints. The real-time scheduling basis for our work is found in the following two theorems [Liu73, Sha90].

Feasibility Theorem [Liu73]: A set of n independent periodic tasks scheduled by the rate monotonic algorithm will always meet its deadlines, for all task phasings, if

$$\frac{C_1}{T_1} + \ldots + \frac{C_n}{T_n} \leq n\left(2^{1/n} - 1\right) = U(n)$$

where C_i and T_i are the execution time and period of task τ_i respectively.

The feasibility theorem guarantees a sufficient condition for any distribution of start times that can arise when the rate monotonic scheduling policy is applied.

Critical Zone Theorem, revised version [Leh89]: A set of independent periodic tasks scheduled by the rate monotonic algorithm will always meet its deadlines, for all tasks phasings, if and only if

$$\forall \quad i, 1 \leq i \leq n,$$

$$\min_{(k,l)\,\varepsilon R_i} \sum_{j=1}^{i} C_j \frac{1}{lT_k} \left\lceil \frac{lT_k}{T} \right\rceil \leq 1$$

0-7803-3192-3/96 $5.00©1996 IEEE

where C_j and T_j are the execution time and period of task τ_j respectively and

$$R_i = \{ (k, l) \mid 1 \le k \le i, l = 1,\ldots,\lfloor T_i/T_k \rfloor \}.$$

For area and time estimation for individual tasks, we again use two techniques: fast and elaborated. Both techniques use the Hyper set of estimation tools [Rab91, Cha95]. The fast technique predicts the number of instances of hardware primitives at RT-level using min-bound technique. The accurate technique uses the Hyper-LP statistical model to estimate the area of implementation.

3. MULTIRESOLUTION REFINEMENT

The synthesis problem for hard real-time multitasks DSP high level synthesis of can be defined as follows:

Given: A set of M tasks described by their CDFGs and their hard real-time timing constraints (task period, start and finish time). *Goal:* Partition the set of tasks in N subsets so that for each subset can be implemented on one dedicated chip assuming a single thread of control and no preemption The cost (sum of areas of all chips) must be minimized.

Several of the problems which must be solved during synthesis are NP-hard. To handle the complexity of this problem, we developed a new multiresolution synthesis strategy. The method interchangeably refines an initial solution, by employing increasingly accurate and increasingly slower lower-bound estimations at the task-level and for individual tasks. As soon as one of estimation indicates that the proposed solution is infeasible, the search strategy backtracks to the previous best feasible solution. A new solution is accepted only after its area is obtained using the Hyper high level synthesis system and at the task level nonpreemptive schedule is generated.

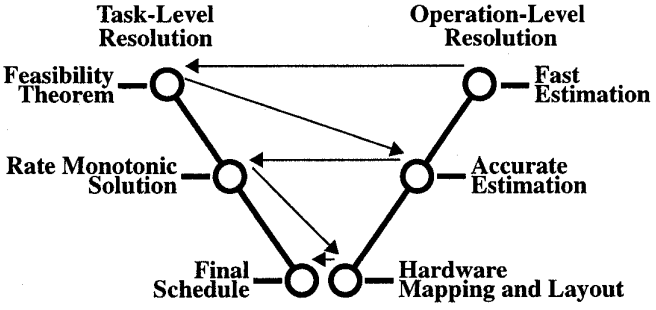

FIGURE 1. V-Chart of the new multiresolution synthesis approach.

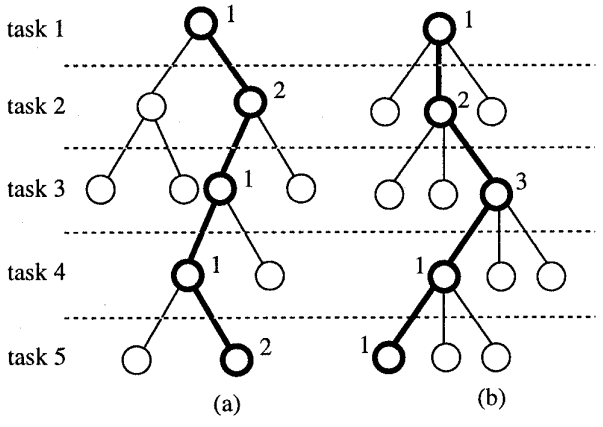

FIGURE 2. Branch and Bound-based Optimal Synthesis Approach : (a) two processor and (b) three processor synthesis procedure

4. OPTIMAL BRANCH-AND-BOUND ALGORITHM

The first synthesis algorithm we present is an optimal branch-and-bound method. Figure 2 illustrates the approach. At each step of the algorithm decision two which group to assign a particular task is made. To each task the task number is assigned. In order to speed-up pruning, the tasks with larger single task areas are assigned smaller numbers. The algorithm first tries to clusters all tasks in two groups, then in three, and so on until it is proven that all solutions which use the larger number of groups are inferior to the current best solution. Each of these attempt we term an epoch. Figure 2a and b shows epochs h which correspond to clustering into 2 and 3 groups respectively. Each node has k children, where k is the number of groups considered in particular epoch. The children are numbered from the left to the right.

The complete tree encodes using paths from the root to any of leaves all possible solutions for a given number of groups in the following way. A particular task (node) is assigned to the group numbered by the number which connects it to its parent. The root is, without loss of generality, assigned to group 1. For example, the bold path in Figure 2a, indicates that tasks 1, 3, and 4 are in group 1, and tasks 2 and 5 in group 2. We compare all paths using the depth-first search. During the enumeration all paths which corresponds to solutions which either violate timing constraints or are inferior to the current best solution are terminated. The size of solution and feasibility are checked using the strategy provided by the V-chart.

In order to speed-up the B&B approach, the initial solution is generated using the heuristic algorithm presented in the next section. Even then, small instances of the problem require very long run times (several hours on a modern workstation).

5. HEURISTIC ALGORITHM

Our second synthesis algorithm is heuristic. Figure 3 shows the design flow for the heuristic approach, which combines successively increasingly accurate high level synthesis estimations with increasingly accurate real-time scheduling tests. The final result is schedule for all tasks and multipurpose ASIC for tasks when preemption is not allowed. The search starts with an initial solution where each tasks is assigned to a separate IC. The search iteratively matches tasks as long as the reduction in sum of areas of all IC is achieved at each step. The criteria for guiding the search engine which tasks from which ICs should be merged into a new IC are based on the following four criteria:

1. **Similar bit widths for operations.** Hardware is wasted if two processes operate on data of different bit widths. Also, longer bit width implies longer cycle time. If neither process has multiplies, the similarity is quantified using the ratio of the number of bits in the process; otherwise, the similarity measure is the square of the ratio, so that the both increases in the hardware size and the clock cycle time are taken into account.
2. **Similar types of functional units required.** The similarity is quantified as the sum of ratios of the estimated number of execution units for all types of resources.
3. **The sources and sinks of data transfers.** Similarly, it is important to match tasks which have similar communication patterns. The Hyper estimation tools are used to obtain this information and the measure is quantified in the same way as for functional units.
4. **A similar number of registers.** The required number of registers is estimated using the Hyper estimation tools which calculate the maximum cutset of the dataflow graph [Rab91], as measured by the ratio of the estimated number of registers in the initial designs.

Note that the preference measures based on different observations may be contradictory. In our previous attempt [Pot95] we used a rank-based function of a similar set of observation to guide the search. However, both intuition and experimentation indicates that it is sometimes important to consider all four measure simultaneously in a balanced way. We accomplish this goal by forming a weighted average of the four ratios calculated using the observation presented in this section.

6. EXPERIMENTAL RESULTS

Table 1 gives descriptions of 16 tasks used to construct five different task sets, including the number of operations, the word length, and the initial area when each task is implemented on a separate chip. We used these tasks to construct the task sets used for the experiments. Table 2 presents the solutions produced by our rank-order based high level synthesis algorithm. Both the average and the median area reductions are by factors slightly larger than two, clearly indicating advantage of combining several tasks on one ASIC implementation. The comparison with the solutions produced by the optimal branch and bound method indicates that for 8 task examples the optimal solution is generated, The run time for all examples is less than 10 minutes on SUN Sparcstation-5, including the complete synthesis of the final solutions.

7. CONCLUSION

We introduced an approach for synthesis of multi-task ASICs. We integrated techniques from hard real-time operating systems and high-level VLSI DSP synthesis. On several examples, both heuristic and optimal techniques achieved a significant area reduction in comparison with the DSP synthesis traditional methods which do not explore task-level hardware sharing.

Acknowledgments

Wolf was supported by the National Science Foundation under grant MIP-9424410.

REFERENCES

[Cha95] A.P. Chandrakasan, et. al."Optimizing Power Using Transformations", Vol. 14, No. 1, pp. 13-32, *IEEE Transactions on CAD*, 1995.

[Leh89] J.P. Lehoczky, L. Sha, Y. Ding, "The Rate Monotonic Scheduling Algorithms - Exact Characterization and Average Case Behavior" *IEEE Real-Time System Symp.*, pp. 181-191, 1986.

[Liu73] C.L. Liu, J.W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard Real Time Environment", *Journal of ACM*, Vol. 20, No. 1, pp. 46-61, 1973.

[Pot95] M. Potkonjak, W.H. Wolf, "Cost Optimization in ASIC implementation of Periodic Hard-Real Time Systems using Behavioral Synthesis Techniques", IEEE Conference on Computer-Aided Design, pp. 446-451, 1995.

[Rab91] J. Rabaey, et al."Fast Prototyping of Datapath-Intensive Architectures", *IEEE Design and Test of Computers*, Vol. 8, No. 2, pp. 40-51, 1991.

[Sha90] L. Sha, J.B. Goodenough: "Real-Time Scheduling Theory and Ada", *IEEE Computer*, Vol. 23, No. 4, pp. 53-62, 1990.

Task #	Task Description	#operations	# bits	Initial area (mm^2)
1.	Controller1	48	8	10.47
2.	Controller2	108	20	38.88
3.	Controller3	97	16	27.28
4.	Controller4	67	16	23.63
5.	Wavelet filter	31	12	15.10
6.	Filter 1	32	10	13.65
7.	Filter 2	38	11	17.82
8.	Filter 3	42	14	18.24
9.	Filter 4	30	13	16.37
10.	8X8 DCT	46	24	27.06
11.	DAC	354	16	26.62
12.	modem 1	227	20	31.78
13.	modem 2	200	20	35.52
14.	Controller 5	324	32	66.82
15.	Formatter	464	32	73.26
16.	Echo-Cancel-ler	212	32	64.57

Table 1. Individual Characteristics of Tasks considered during experimentations

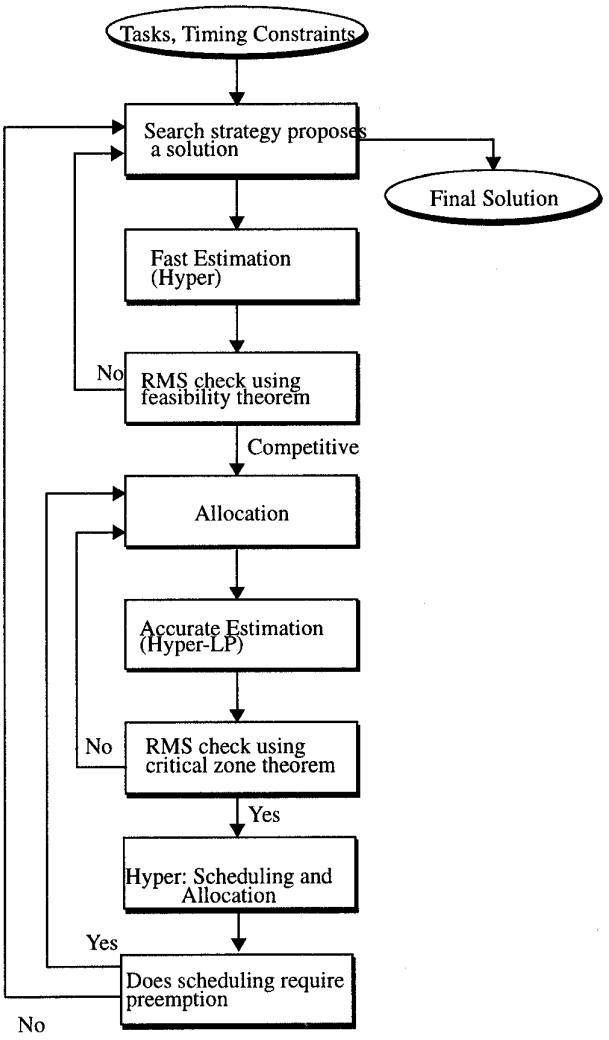

FIGURE 1. The design flow of heuristic synthesis algorithms for design of hard-real time multitasks

set of tasks	Tasks in the set	Final solution	Final area (mm^2)	Improvement initial/final
set 1	{1,2,3,4,5,6,7,8}	{1,5,6} {2,3,4,7,8}	18.92+63.27	**2.01** (165.07/ 82.09)
set 2	{5,6,7,8,9,10,11,12,13,14, 15,16}	{5,6,7,8,9,11} {10,11,12,13} {14, 15,16}	52.77+54.09 + 98.80	**1.98** (406.81/ 205.66)
set 3	{9,10,11,12,13,14,15,16, 1,2,3,4}	{1,3,4} {2,11,12,13} {10,14,15, 16}	32.28+45.92 + 99.22	**2.49** (442.26/ 177.42)
set 4	{13,14,15,16,1,2,3,4,5,6, 7,8}	{1,5,6} {2,3,4,7,8} {13} {14,15,16}	18.92+63.27 +35.52+ 98.80	**1.87** (405.24/216.51)
set 5	{1,2,3,4,5,6,7,8,9,10,11, 12,13,14,15,16}	{1,3,4,5,6,7,8,9,10} {2,11,12,13}{14,15, 16}	63.36+49.98 + 98.80	**2.39** (507.07/220.79)

Table 2: Experimental result - Area of implementation for ASIC hard-real time designs.