Integrated Resource Assignment and Scheduling of Task Graphs Using Finite Domain Constraints

Krzysztof Kuchcinski

Dept. of Computer and Information Science, Linköping University
Linköping, Sweden
kku@ida.liu.se

Abstract

This paper presents an approach to modeling of task graphs using finite domain constraints. The synthesis of such models into an architecture consisting of microprocessors, ASICs and communication devices, is then defined as an optimization problem and it is solved using constraint solving techniques. The presented approach offers an elegant and powerful modeling technique for different architecture features as well as heterogeneous constraints. The extensive experimental results prove the feasibility of this approach.

1. Introduction

The specification of embedded systems is usually provided in a form of communicating tasks. The goal of the synthesis is to find an assignment of subtasks to processors and ASICs, and communication subtasks to communication devices (e.g., buses or links) as well as their related static schedule. The schedule, in this case, is defined as an assignment of starting times to every subtask and communication activity. Different heuristics and Mixed Integer Linear Programing (MILP) formulations have been proposed for this problem. Constraint Logic Programing (CLP) formulation has been originally suggested by the author [6, 7].

This paper further examines the use finite domain constraints to model and synthesize heterogeneous embedded systems. Constraint solving techniques combined with optimization methods are then proposed to provide a solution to the synthesis problem.

2. Finite Domain Constraints System Model

A system is defined by an acyclic task data-flow graph with nodes denoting subtasks and arcs data precedence relations (communications) between them. When the last subtasks finish their execution the first subtasks are started again. The target architecture consists of computational components (CPU's and ASIC's) connected by buses. Each subtask has a specified execution time on a selected computational component.

The subtask of a task data-flow graph is modeled as a 3-tuple of *finite domain variables*:

$$T=(\tau, \delta, \rho) \tag{1}$$

where τ denotes the start time of the subtask, δ its duration and ρ the resource number which is assigned for its execution. The data flow precedence relation, represented as an arc in the task data-flow graph, is expressed as an *inequality constraint*. For example, if the subtask T_i precedes subtask T_j, the following constraint is imposed:

$$\tau_i+\delta_i \le \tau_j \tag{2}$$

To synthesize the previously specified task data-flow graph, it is necessary to introduce additional constraints on resources sharing. These constraints forbid simultaneous use of shared resources, such as processors and buses. In the prototype system we makes use of a rectangle interpretation of a subtask. The rectangle based resource constraint assures that the 2-dimensional rectangles $R_i=[\tau_i, \rho_i, \delta_i, 1]$ and $R_j=[\tau_j, \rho_j, \delta_j, 1]$ representing the subtasks T_i and T_j do not overlap. It is defined by the `diffn/1` constraint. The graphical representation of this constraint is depicted in Fig. 1.

A subtask execution time depends on a selected resource, such as microprocessors or ASICs. The mapping which binds a given execution time to a given resource is defined by the `element/3` constraint.

Pipelining a task data-flow graph can be modeled by introducing *n* copies of existing rectangles, starting at positions *k*, *2·k*, *3·k*, etc. This prevents to place subtasks in forbidden locations, which are to be used by subsequent pipeline computations.

3. System Synthesis

System synthesis is defined, in our approach, as a pro-

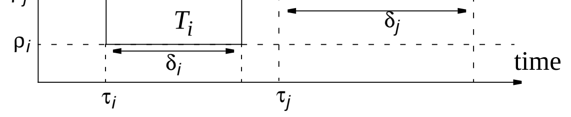

Figure 1. A graphical representation of the rectangle constraint.

Supported by the Wallenberg Foundation project "Information Technology for Autonomous Aircraft"

cess of finding an assignment to all domain variables which satisfies all constraints and minimizes a given cost function defined as a domain variable. The cost function can be defined, for example, as a maximum value among all $\tau_i+\delta_i$. Minimizing this domain variable yields the fastest implementation satisfying all constraints.

The optimal solutions can be obtained using branch-and-bound algorithm (B&B). CLP systems offer built-in minimization and enumeration (labeling) procedures. Additional labeling strategies, such as *domain splitting* strategy [8] and *domain intervals*, have also been implemented and tested. The advantage of CLP is possibility to use heuristic search algorithms. In this paper, we have examined two meta-heuristics, *limited discrepancy search* (LDS) [5] and *credit search* [1], which can be used together with B&B algorithm and selected labeling strategy.

4. Experimental Results

For experiments we have used random task graphs [4] and an example from [2]. The experiments have been carried out using a prototype implementation of the synthesis system implemented in CHIP 5 [3] on the Pentium 200MHz computer. If not explicitly indicated, the runtimes presented in this paper are the total optimization execution times for finding a solution and proving that it is the optimal one.

We have used 125 random task graphs divided into 5 groups of 25 task graphs. Each group has at least 20, 40, 75, 130 and 200 computational and communication subtasks. In each group, there are 15 task graphs with uniform distribution and 10 with exponential distribution of the subtask execution time. The implementation architecture consist of a number of processors interconnected by busses.

Table 1 presents results obtained with optimal methods with the execution time-out of 10 minutes. In the Table 2, we present results obtained with heuristic optimization algorithms (LDS, credit search and a simple heuristic) and compare them with the best obtained results presented in the summary of the Table 1.

The last example is the video coding algorithm H.261 derived from [2]. The task graph contains 12 subtasks and 14 interconnections between them. We have made three experiments using non-pipelined and pipelined implementations.

Labeling method		tasks/processors/buses				
		20/3/1	40/4/2	75/4/2	130/6/3	200/6/3
Input order	solutions/optimal	25/25	25/22	25/12	24/8	25/11
	worse than LB	-	17.74%	2.80%	3.44%	2.80%
	Runtime (s)	8.91	21.14	15.11	17.25	37.78
Domain split	solutions/optimal	25/25	23/21	25/12	23/7	24/11
	worse than LB	-	4.38%	2.86%	3.54%	2.12%
	Runtime (s)	7.79	5.50	49.89	14.93	31.20
Domain intervals	solutions/optimal	25/25	25/19	25/6	25/4	25/4
	worse than LB	-	4.83%	4.03%	6.51%	2.79%
	Runtime (s)	0.57	17.43	1.78	26.95	9.13
Summary	solutions/optimal	25/25	25/24	25/15	25/9	25/13
	worse than LB	-	4.00%	2.96%	3.16%	1.84%
	Runtime (s)	0.38	11.23	39.78	11.19	23.4

Table 1: Synthesis results for random task graphs using algorithms providing optimal solutions.

Heuristic		tasks/processors/buses				
		20/3/1	40/4/2	75/4/2	130/6/3	200/6/3
LDS	Δ from best	6.36%	3.09%	-0.60%	-1.33%	0.18%
	Runtime (s)	0.49	2.03	8.10	50.65	142.77
Credit (L/2)	Δ from best	6.49%	2.74%	-0.80%	-2.66%	1.23%
	Runtime (s)	0.63	1.29	6.92	37.23	204.62
Credit (sqrt(L))	Δ from best	6.59%	3.23%	1.27%	2.31%	1.94%
	Runtime (s)	0.57	1.13	3.64	12.62	43.06
Simple heuristic	Δ from best	11.18%	12.59%	6.70%	7.75%	4.65%
	Runtime (s)	0.05	0.13	0.42	1.31	2.90

Table 2: Synthesis results for random task graphs using heuristic optimization algorithms.

5. Conclusions

We have presented a new approach to modeling and synthesis of heterogeneous embedded systems using finite domain constraints and constraints solving techniques. The experimental results indicate that the presented method can be used for synthesis of heterogeneous systems offering very good performance.

References

[1] N. Beldiceanu, E. Bourreau, H. Simonis and P. Chan, Partial search strategy in CHIP, *Presented at 2nd Metaheuristic International Conference MIC97*, Sophia Antipolis, France, 21-24 July 1997.

[2] A. Bender, Design an Optimal Loosely Coupled Heterogeneous Multiprocessor System, In *Proc. of the European Design and Test Conference*, March 11-14, 1996, Paris, France.

[3] *CHIP, System Do*cumentation, COSYTEC, 1996.

[4] P. Eles, K. Kuchcinski, Z. Peng, A. Doboli and P. Pop, Scheduling of Conditional Process Graphs for the Synthesis of Embedded Systems, *Proc. Design, Automation and Test in Europe Conference*, Feb. 23-26, 1998, Paris, France.

[5] W. D. Harvey and M. L. Ginsberg, Limited Discrepancy Search, *Proc. IJCAI 1995*.

[6] K. Kuchcinski, Embedded System Synthesis by Timing Constraints Solving, In *Proc. of the 10th Int. Symposium on System Synthesis*, Sep. 17-19, 1997, Antwerp, Belgium.

[7] K. Kuchcinski, An Approach to High-Level Synthesis Using Constraint Logic Programming, *Proc. 24th Euromicro Conference, Workshop on Digital System Design*, Västerås, Sweden, August 25-27, 1998.

[8] K. Mariot and P. J. Stuckey, *Programming with Constraints: An Introduction*, The MIT Press 1998.

Design	Performance (time units)	Stage latency (time units)	Runtime (s)	B&B nodes
non-pipeline, 2 buses	2963	—	0.3	26
3 stage pipeline, 1 bus	3996	2351	19.34	27
3 stage pipeline, 2 buses	3373	1154	12.07	261
3 stage pipeline, 3 buses	3329	1110	5.91	261

Table 3: Synthesis results for H.261 example.