ISSN:0975-9646

A novel algorithm for co-synthesis of wireless client-server systems using preference vectors

Mohammad Mehdi Hassani , Vahid Jalali

Islamic Asad University Aiatollah Amoli Branch,Amol,Iran

***Abstract*—Embedded systems are used all over our society. Current estimates indicate that over 90 percent of worldwide computers are embedded systems [1]. As the complexity of system design increases, use of pre-designed components, provides an effective way to reduce the complexity of synthesized hardware. Hardware-Software co-synthesis is the process of partitioning an embedded system specification into hardware and software modules in order to meet performance, power consumption and cost goals. While the design problem of systems that contain processors and ASIC chips is not new, computer aided synthesis of such heterogeneous or mixed systems poses challenging problems because of the differences in model and rate of computation by application-specific hardware and processor software. One of the areas that are investigated recently is the simultaneous co-synthesis of client and server processing elements in real time embedded client server systems. In this paper we propose an improvement on COWLS algorithm to take into account preference and peak power consumption information.**

***Keywords*—embedded systems, low power consumption, wireless systems, client server systems.**

I. INTRODUCTION

Most digital systems (e.g. real-time embedded systems) used for dedicated applications consist of general-purpose processors, memory and application-specific ICs (ASIC). In addition to being application-specific, such systems are also designed to respect constraints related to meet relative timing deadlines of their actions, hence these are referred to as *real-time embedded systems*.

There is a significant body of available work on hardware-software co-design. The main body of hardware-software co-design consists of four problems:

- Architecture selection: determine the communication architecture including memory structure and interconnection network structure
- Component selection: determine the processing elements to be used. For hardware modules, an implementation of each task should be selected.
- Partitioning and scheduling: partition the input tasks into processing elements and perform static scheduling to determine the execution times of tasks.
- Performance evaluation: evaluate the quality of solution and check whether design constraints are met. Hardware-software co-synthesis is the automated synthesis of hardware-software embedded systems. Work in hardware-software co-design focuses on providing a designer with tools and guidelines which ease the exploration of available implementation options.

Two distinct approaches have been used for distributed system co-synthesis: optimal and heuristic. Hardware-software co-synthesis involves various steps such as allocation, scheduling and performance estimation. Both allocation and scheduling are known to be NP-complete. Therefore, optimal co-synthesis is computationally a very hard problem. In the optimal domain, the two approaches are mixed integer linear programming

(MILP) and exhaustive. MILP solution has the following limitations:

- it is restricted to one task graph
- it does not handle preemptive scheduling
- it requires determination of the interconnection topology upfront
- because of complexity, it is suitable only for small task graphs

Exhaustive enumeration of all possible solutions is also impractical for large task graphs.

Heuristic co-synthesis methods cannot guarantee the optimality of the answer. There are two distinct approaches in the heuristic co-synthesis domain: iterative and constructive. Iterative procedure considers only one type of communication link and does not allow mapping of each successive instance of a periodic task to different processing elements (PE). Constructive co-synthesis procedure does not support communication topologies such as bus, LAN etc., and it uses a pessimistic performance estimation technique. It is also not suitable for multi-rate embedded systems. Power consumption is not been optimized in any of these co-synthesis techniques.

In short words, the hardware-software co-synthesis problem is intractable. Presently, only non-exhaustive optimization algorithms are capable of solving large problem instances of distributed, embedded systems in a reasonable amount of time. Researchers have tackled variants of the co-synthesis problem with iterative improvement algorithms, constructive algorithms, simulated annealing algorithms, evolutionary algorithms and a rapid sub-optimal timing constraints solver.

II. COWLS ALGORITHM

Unlike most of distributed, heterogeneous embedded systems co-synthesis algorithms, COWLS algorithm is based on simultaneous synthesis of client and server. However, independent synthesis of client and server will produce similar results as the previous work.

The COWLS algorithm models three main types of resources: processing elements, communication resources and memory. Processing elements are considered to be general purpose or special- purpose processors which are all capable of executing available tasks.

There exist two types of processing elements: client PEs and server PEs. There are multiple characteristics associated with each PE type. Characteristics considered in the COWLS algorithm consist of price, idle power consumption. Considering the problem formulation discussed above and given the client-server system requirements in the form of a set of task graphs, as well as the attributes of the PEs, memory, and communication resources available, COWLS attempts to synthesize client-server systems which meet the requirements with minimum price, client power consumption and soft deadline. In the case of hard deadline violation the solution is no more valid, however when a soft deadline violation occurs, the system will be still valid and the aim will be to minimize the response time.

Architecture's expenses depend on the manner in which resources are used in its edifice. Therefore, by attempting to meet real-time constraints, one ensures that high speed PEs, which are tailored to the tasks required, are used for tasks which lie along critical paths in the task graphs. By attempting to minimize price, one ensures that PEs, which are capable of carrying out the required tasks with minimal price, are used. By attempting to minimize client power consumption, one minimizes the number of power intensive tasks run on power-hungry PEs located on the client. Of course, some of these goals conflict with each other. For this reason, a single run of COWLS generates multiple solutions which explore the trade-offs among different costs.

After the co-synthesis is carried out for the client and servers together, the cost of the design is estimated by multiplying the cost of client by number of clients and cost of the server by the number of available servers. If the only important cost is the client cost, the number of servers is set to zero in the evaluation process i.e. the server cost is multiplied by zero.

Independent synthesis of client and server is similar to co-synthesis problem for distributed, heterogeneous embedded system with two types of processing elements used for presenting client and server.

III. OPTIMIZATION

Optimizations for different costs of a system can be achieved by finding a single solution and then running constructive or iterative improvements on it. There exists many ways to find an optimal solution among different proposed solutions. One famous way is to use a cost function that brings into account relative importance of all factors to evaluate relative optimality of a solution. Another solution is to rank each solution based on its costs and choose the best solution after ranking. In the COLWS algorithm we use the ranking method to find the optimal solution. We need a hierarchy in finding the optimal solution.

IV. ENHANCEMENTS ON COWLS

In the COWLS algorithm, we have two types of processing elements, server PEs and clients PEs. There exists a unique table associated with each PE type indicating performance of each task on that PE type as well showing that if it is possible to execute this task on this type of processing element. Formulation notation and exploration is discussed in the model formulation section above. A factor that can influence and improve the quality of solutions and amount of CPU time required for the co-synthesis action, is to bring into account the preference of the designer in binding procedure. The preference of the designer may be because of previous experimental results e.g. it may be proven in the previous experiments that a particular task is better to be done on the client than on the server.

P: Preference vector

$P = \{p(i,j) \mid 0<i< Nt, 0<j<NPE\}$

p(i,j) : Preference to do the task i on the processing element j Another factor that may help improve the algorithm and make it more practical is to add the peak power consumption information in the co-synthesis process in order to gain improvements on packaging cost. The peak power information can be formulated to use as below;

PPV: Peak power vector

$PPV = \{pp(i,j) \mid 0<i< Nt, 0<j<NPE\}$

pp(i,j): Peak power value of the task ti when running on the processing element type j The third improvement is to bring into account is the definition of exclusion vector for each task, which specifies whether certain tasks can co-exist on the same processing element.

Each of the matters discussed above are considered in the following solution and added to the COWLS algorithm with slight divergence.

V. EXPERIMENTAL RESULTS

The proposed methodology is implemented using C++. The C++ program used for this simple synthesis tool is implemented by me. It is consist of a core code that is used to represent the task graph and system requirements using the questions asked from the user. The next step will be to complete the user interface in order to graphically display the characteristics and requirements of a system.

Different co-synthesis algorithms can be added to the core code as daemons and can be run by selecting the co-synthesis algorithm. The daemon will run on the specified model, ask questions needed to complete the synthesis of the model and produce results based on the co-synthesis algorithm and model characteristics. The different improvements proposed for the COWLS algorithm are implemented independently. First, the preference problem is implemented. It is shown that the processing time needed to complete the synthesis action acquired by time function in C++ shows enormous amount of decrease when preference is available. It is because of the elimination of some of states required processing. The decrease in time was proportional to the portion of preferred tasks on processing elements.

The peak power information is also used to provide packaging costs in the proposed method without imposing a bing burden on algorithm. The exclusion vector is not yet implemented in experiment. But it will also decrease the CP U time needed to complete the synthesis action.

The experiment is run on a simple task graphs proposed by me. But if one requires results with applications on real life, can use TGFF which is a kind of tool used for creation of different problem instances. It is not guaranteed that all problem instances used generated by TGFF are solvable. For a number of examples, multiple non-dominated solutions may be produced by each design run, as in hand written experiments.

VI. CONCLUSION

There are several algorithms for hardware-software co-synthesis of distributed, heterogeneous embedded systems. Among these algorithms the only one that simultaneously synthesizes the client and server requirements is the COWLS algorithm. COWLS automatically synthesizes embedded client-server architectures. It uses a multi objective evolutionary algorithm to simultaneously produce multiple solutions which trade off different costs. It optimizes cost, average client power consumption, and soft deadline violations under hard real-time constraints and constrained client-server communication bandwidth. The experimental results show that COWLS frequently makes design decisions which are similar to those which would be intuitive to a human designer. However, it occasionally makes counter-intuitive decisions which are preserved if they assist in the evolution of non-dominated solutions. The COWLS algorithm is frail in it cannot bring into account the preference of the designer that can reduce the synthesis time and effort. Another factor that may help COWLS improve is to consider peak power consumption as well. The third improvement is to bring into account is the definition of exclusion vector for each task, which specifies whether certain tasks can co-exist on the same processing element. This feature may help the algorithm better decide about the tasks co-existing on the same PE.

The proposed algorithm is implemented using C++ and the results show colossal decrease in the CPU time used for synthesis as well as better results for packaging. Another problem that I think is interesting to be continued is to consider only one wireless communication media s it is for common cellular networks and try to solve the problem just with the tasks assigned to each PE.

REFERENCES

[1] Jecob levman, Gul Khan, Javad Alirezaie, “Hardware-Software Co-synthesis of bus architecture embedded devices,” VLSI systems, IEEE transactions on , 2004

[2] Dave, B.P.; Lakshminarayana, G.; Jha, N.K.; “COSYN: Hardware-software cosynthesis of heterogeneous distributed embedded systems, “ *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on,* Vol 7, Issue 1, pp.92 – 104, march 1999

[3] Giovanni De Micheli, Rajesh K. Gupta, “Hardware-Software Co-Synthesis for Digital Systems, ”

[4] El-Kharashi, M.W., El-Malaki, M.H., Hammad, S., Salem, A. and Wahdan, A.; “Towards automating hardware/software co-design,” System-on-Chip for Real-Time Applications, 2004.Proceedings. 4th IEEE International Workshop on, pp. 189 – 92, July 2004

[5] Dick, R.P.; Jha, N.K.; ”COWLS: hardware-software co-synthesis of wireless lowpower distributed embedded client-server systems,” *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on,* Vol 23, Issue 1, pp 2 – 16, Jan. 2004

[6] Jha, N.K.; Dick, R.P.; “COWLS: hardware-software co-synthesis of distributed wireless low-power embedded client-server systems,” *VLSI Design, 2000; Thirteenth International Conference*, pp. 114 – 120, Jan 2000

[7] Dave, B.P.; Lakshminarayana, G.; Jha, N.K.; “Cosyn: Hardware-software Cosynthesis Of Embedded Systems,” *Design Automation Conference, 1997. Proceedings of the 34th* pp. 703 -8, June 1997