Performance-Complexity Analysis in Hardware-Software Codesign for Real-Time Systems

Victor V. Toporkov

Department of Computer Engineering
Moscow Power Engineering Institute (Technical University)
14, Krasnokazarmennaya str., Moscow, 111250, Russia

Abstract

The paper presents an approach for performance and complexity analysis of hardware/software implementations for real-time systems on every stage of the partitioning. There are two main features of the approach. The first one is the rapid performance-complexity estimations for software based on the set of introduced stochastic characteristics. The second one is the systematic exploration of the codesign space, that enables to determine the partition process direction. These properties allow the renovation of the code-segment candidate list for hardware implementation during the partitioning with internal representation transformations, and the significant design space reduction.

1. Introduction

A successful solution of the constrained hardware-software partitioning problem depends on adequate estimates of performance characteristics and the implementation cost (the complexity) of appropriate HW/SW system parts on all stages of the partitioning. To reduce the HW/SW codesign space and to control the partitioning process one could use an appropriate cost function counting performance-complexity requirements.

Both the HW-oriented [1, 2] and the SW-oriented [3, 4] approaches allow fine-grain automatic partitioning. Among the related work, authors in [5] investigate the partitioning problem from a cospecification.

Despite the similarity of the results for different initial conditions [4] the efficiency of HW/SW partitioning in these approaches depends on the initial solution in the codesign space, and the cost function must be automatically adapted.

A clustering approach [6, 7] with using closeness criteria [8] to control the partitioning process turns to account the design space properties. However the user decides on clustering and partitions the operations. In addition, the highly nonmonotonic design space makes difficulties in introducing the metric (the distance function) [7]. In [9], an approach is described which uses a relaxed cost function that enables the partition algorithm to focus on satisfying performance and to handle the HW minimization. The parameterized architecture model [10] is proposed which allows to consider the number of buses, memory ports, and connection styles affecting machine parallelism.

Fine-grain partitioning in HW- and SW-oriented approaches has such serious side effects as communication time overheads [2, 4].The flexible paradigm for the problem of communication between HW and SW subsystems via communication units (controllers) is proposed in [11, 12]. It is difficult to predict communication side effects precisely without a global dataflow analysis and under the fixed set of code-segment candidates for moving to HW [4].

2. Main goals and features

The main objectives of performance-complexity analysis are to estimate marginal satisfiability for performance requirements on every stage of HW/SW partitioning and to determine the partition process direction in the HW/SW codesign space for the cost function minimization. There are several distinctive features in the proposed approach.

- First, starting from the system specification as a C program (as in the software-oriented approach [3, 4]) it allows to extract the Pareto optimal set of system alternatives in the HW size - system performance codesign space, to estimate extremely different

implementations as HW [1, 2] or SW [3, 4], and to choose an optimal HW/SW one.

- Second, profiling the C program and using the special graph for an internal representation - a metaoperator net (M-net) [13], this approach enables to estimate the software complexity on the object code level and even on the assembly language level with the rapid performance estimation system. It is important because using the assembly code based on the details of the processor selection let us reduce redundancy introduced by different compilers in SW timing estimation, and the estimation is fast due to the special C program profiler realization.
- Third, using generalized performance-complexity estimates and the codesign space properties (the Pareto subsets) it is possible to control the partitioning process as in [6, 7], but, in constrast with above works, this approach enables fine-grain automatic partitioning, and the communication overhead minimization.

Experimental results discussed in Section 6 are promising and prove the relative insensibility of the proposed approach to the initial solution.

3. Performance-complexity analysis overview

This section addresses an inner loop of performance-complexity analysis. After HW/SW partitioning, assembly (for SW) and VHDL (for HW) code generation, and high-level synthesis, the stage of global run time analysis is necessary (an outer analysis loop). The major steps of the inner loop analysis are the following.

1) *Preliminary profiling.* The GSSS system [13] was used as a platform for the performance-complexity investigation in HW/SW partitioning.

We use the two-stage investigation of the SW complexity: on the level of C functions, basic blocks and statements, and on the assembly code level by building the SW execution trace. This trace can be built with using trace interruptions (as an example, the interruption 01 in BIOS for IBM PC) and the frequency counters method. This method consists of short operations and instructions automatic clustering, gathering statistics, and using special tables for the calculation of execute instruction times for different processors.

In the presence of nondeterministic operations in the system specification (data-dependent operations, loops and waiting for external events [2]) we use stochastic estimates for the SW complexity (the number of processor cycles) and the maximal CPU cycle time.

2) *SW run modelling.* Those SW (code) segments are selected for HW moving, where timing constraints are violated. For multiprocessor systems the partition task is complicated by global scheduling and allocation. The selected code segments are belonging to the critical path, and the partition task is solved for these segments. For those code segments which are not critical, D.R. Fulkerson task is solved. That is an optimal delay distribution for a cost function minimization under minimum/maximum timing constraints.

3) *The SW-segment candidate list renovation.* After the HW evaluation of the selected code segment the internal representation transformations may be possible (as an example, concurrent operations in SW and HW). In consequence of these transformations the renovation of timing constraints is probable, and the candidate list may be reduced significantly. In the multiprocessor case only those transformations are possible which do not violate minimum/maximum timing constraints for noncritical path SW-segments. Above properties define the proposed approach as an adaptive one.

After the Pareto optimal variant extraction and the systematic HW/SW codesign space exploration the constrained partition optimization is realized.

4. Performance-complexity estimates in HW/SW partitioning

4.1. Processing model

In this section, the software running model for a general-purpose processor is discussed. The main goal is to use it in performance-complexity analysis.

A central processor unit (CPU) model for dedicated real-time embedded systems captures the following function units: a general-purpose L-bit processor (it may be a single chip one or a bit-slice one); an internal random-access memory (RAM); the direct memory access (DMA) logic to avoid K/L processor interruptions, where K is a length of an input/output words, and $K \geq L$. The average value of K-bit processing time is defined with counting an average number $\overline{a}_p$ of operations for L-bit processing with a basic set of processor operations, an average number $\Delta\overline{a}_p$ of operations for L-bit preprocessing and postprocessing (the internal RAM accesses and respective operations), an average value $\overline{\tau}_p$ of the CPU cycle time, and the coefficient p determined by the processing manner:

$$\overline{T}_p(K) = \left[(K/L)*\overline{a}_p + p*(K/L-1)*\Delta\overline{a}_p\right]*\overline{\tau}_p . \quad (1)$$

Figure 1 (see the next page) shows a purely serial (a) and an internal pipelined (b) processing, when $K > L$. For cases (a) and (b) correspondent values of p are $p_s = 4$ and $p_p = 2$.

We are given an input data block consisting of d bits, the time constraint $T_p(d)$ for d-bit processing by using software $S_p(d)$, which requires not more than $|S_p(d)|$ processor cycles under a given value of L.

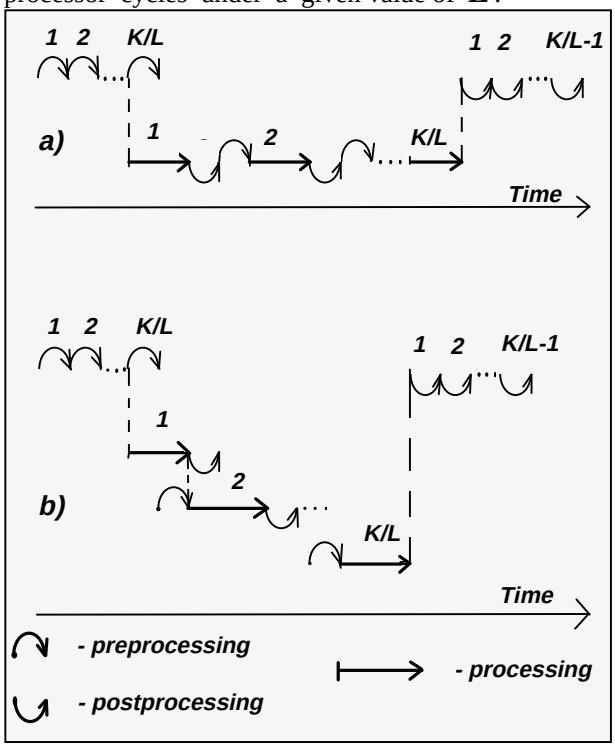

Figure 1: A purely serial (a) and an internal pipelined (b) processing models

The value of K is determined by a target architecture and system parameters during HW/SW partitioning. On the initial stage of the SW complexity estimation (section 4.3) it is supposed $K = L$. The estimation of $\bar{a}_p$ is defined as $\bar{a}_p = \left(|S_p(d)| / d\right) * L$, and the value of $\Delta\bar{a}_p$ depends on a basic set of processor operations.

The maximal acceptable value of the CPU cycle time is

$$\hat{\tau}_p = T_p(d) / \left[|S_p(d)| + p * \frac{d}{K} * \left(\frac{K}{L} - 1 \right) * \Delta\hat{a}_p \right], \quad (2)$$

where $\Delta\hat{a}_p$ is the maximum of Δa_p.

If we suppose in (1), that $1 \le \Delta a_p \le \bar{a}_p$,then we can obtain from (2) coarse-grain estimates for examples (a) and (b) in Figure 1:

a) $\hat{\tau}_{ps} = T_p(d) * K / \left[(5K/L - 4) * |S_p(d)| * L \right]$,

b) $\hat{\tau}_{pp} = T_p(d) * K / \left[(3K/L - 2) * |S_p(d)| * L \right]$.

It is obvious, that $\hat{a}_{ps} = \hat{a}_{pp}$, if $K = L$.

In all cases for all feasible values of $|S_p(d)|$ in HW/SW partitioning (under system parameters variations including K, L, p) the real value of τ_p must be constrained as $\check{\tau}_p \le \tau_p \le \hat{a}_p$, where $\check{\tau}_p$ is the minimal CPU cycle time which can be obtained by the well-known methods.

Upper indices "-" and "+" in the following notations designate the minimum and the maximum values.

So, global timing constraints for a single processor system are satisfiable, if $T_p^+(d) \le T^*$ (T^* is an upper bound of the execution time) and $\hat{a}_p^- > \check{\tau}_p^+$ in accordance with (2) for $T_p^-(d)$, $|S_p^+(d)|$.

Further, it is supposed that software $S_p(d)$ is devided into n program threads (code segments) $S_{pi}(d)$, $i \in 1..n$, so, that $S_p(d) = \bigcup_{i=1}^{n} S_{pi}(d)$, $\bigcap_{i=1}^{n} S_{pi}(d) = \varnothing$, and, as a consequence,

$$|S_p(d)| = \sum_{i=1}^{n} |S_{pi}(d)| \quad . \quad (3)$$

Each of code segments with

$$\hat{\tau}_{pi}^- = T_{pi}^-(K_i) / \left[|S_{pi}^+(d)| * \frac{K_i}{d} + p * \left(\frac{K_i}{L} - 1 \right) * \Delta\hat{a}_{pi} \right], \quad (4)$$

such, that $\hat{a}_{pi}^- \le \check{\tau}_p^+$, is the candidate for the HW implementation. In (4) K_i is the value of K for the SW segment with the index i .

After partitioning $\hat{a}_p$ is defined as $\hat{\tau}_p = \min_t \{\hat{\tau}_{pt}^-\}$ where $t \le n$, t is the number of the code segments in the HW/SW system.

4.2. Multiprocessing model

For multiprocessor systems with m processors the expression (3) is modified as

$$|S_{pm}(d)| = \sum_{l=1}^{C} |S_{pl}(d)| \quad , \quad (5)$$

$S_{pl}(d) \subseteq S_{Cm}(d), l \in 1..C$, where S_{Cm} is the critical path set of C segments, $C \le n$, $n \ge m$.

If $T_{pC}(d)$ is the execution time of the critical path segments, then global timing constraints for a m-processor system are satisfiable, when $(\forall i \in 1..n)$

$(\exists j \in 1...m)\ \left[\left(T^{+}_{pC}(d) \le T^{*}\right)\&\left(\hat{\tau}^{-}_{pij} > \tilde{\tau}^{+}_{pj}\right)\right]$, that is respective scheduling and allocation are found such, that the code segment i might be executed on the processor j.

The selection of code segments of the critical path as candidates for the HW implementation is defined in accordance with the violation $\Phi^{-}_{pij} \le \tau^{+}_{pj}$, where Φ^{-}_{pij} is estimated similarly to (4).

Expressions in (3), (5) are additive measures. So, we can use them in high-level transformations of the internal representation (as an example, in unrolling), in scheduling based on the critical path method and D.R. Fulkerson model, and in partitioning by M-net approach [13].

Thereby, on every HW/SW partitioning stage we can estimate $\left|S_p(d)\right|$ or $\left|S_{pm}(d)\right|$ in accordance with (3), (5). We call these parameters as SW complexity.

4.3. The SW performance-complexity stochastic analysis

After r software $S_p(d)$ runs (profiling and the program execution) with complexities $\left|S^1_p(d)\right|,...,\left|S^r_p(d)\right|$ we can obtain the estimators of the mathematical expectation

$$\left|\tilde{S}_p(d)\right| = \sum_{k=1}^{r}\left|S^k_p(d)\right|/r, \qquad \%_p = \sum_{k=1}^{r}\Phi^k_p/r,$$

and the unbiassed estimators of the dispersion

$$\tilde{D}_s = \sum_{k=1}^{r}\left[\left|S^k_p(d)\right| - \left|\tilde{S}_p(d)\right|\right]^2/(r-1),$$

$$\tilde{D}_\tau = \sum_{k=1}^{r}\left(\hat{\tau}^k_p - \tilde{\tau}_p\right)^2/(r-1).$$

For every SW run Φ^k_p is defined in accordance with (2) for the given processor type and the processing coefficient p.

Then the classical estimators for the confidence interval are

$$I_\beta = \left(\left|\tilde{S}_p(d)\right| - t_\beta\sqrt{\tilde{D}_s/r}; \left|\tilde{S}_p(d)\right| + t_\beta\sqrt{\tilde{D}_s/r}\right),$$

$$I_\gamma = \left(\tilde{\tau}_p - t_\gamma\sqrt{\tilde{D}_\tau/r}; \tilde{\tau}_p + t_\gamma\sqrt{\tilde{D}_\tau/r}\right),$$

where β, γ are the values of the confidence probability; t_β, t_γ- the roots of the equations $2\Phi(t_\beta) - 1 = \beta$, $2\Phi(t_\gamma) - 1 = \gamma$ with the Laplace function $\Phi(\beta)$, $\Phi(\gamma)$.

Therefore, we must operate with the statistic values of $\left|\tilde{S}_p(d)\right|$ and $\%_p$, so that $\tau^{+}_p \le \tau_p \le \Phi^{-}_p$, where $\Phi^{-}_p = \%_p - t_\gamma\sqrt{\tilde{D}_\tau/r}$, τ^{+}_p is the maximum of minimal CPU cycle times for different operation types.

4.4. Communication overhead cost estimation

Communication overhead minimization is a challenge in fine-grain HW/SW partitioning. The main goal of the communication cost estimation is to obtain an upper bound of the number of variables to be communicated if the SW segment (a M-net node) or the segment set (several nodes) are moved to hardware and the number of processor cycles for variable transfering. A global data flow analysis is computation-time-intensive, but an analysis only of adjacent blocks (nodes) may be coarse-grain for real embedded systems [4].

Effective relaxation algorithms for M-net marking based on the Least Common Multiple (LCM)-method were developed. They enables a rapid global data flow analysis.

For implementation details, see [13]. The basic idea of the LCM-method is the following: each node i (a SW segment) has a set of input and a set of output variables. The variable sizes $\{in(i)\}$ and $\{out(i)\}$ may be different, but the number of variables for every input and every output of the node i is the same. Each node obtains the input variables from its predecessors with variable sizes $\{out(pr)\}$ and transmits the output variables to its successors with variable sizes $\{in(suc)\}$.

5. The formal definition of the constrained partition optimization problem

As mentioned in section 4.1, the CPU model captures not only a processor. So, we shall not neglect CPU hardware.

The experimental results presented in the Section 6 base on the target architectures consisting of the following functional units: CPU with the HW size H_P; formatting conversion units (FCU) with the HW size H_F; communication units (CU) with the HW size H_C (that is buses, multiplexers etc.); memory (it is distinct from RAM in CPU) with the HW size H_M .

For every functional unit the HW size is estimated as following

$$H = \sum_{I=1}^{A} \sum_{J=1}^{B} H_{IJ} * \alpha_{IJ} \quad ,$$

where A is the number of VLSI families; B is the number of circuit types in the family C_I ; H_{IJ} is the number of circuits in the family C_I; α_{IJ} is the transition coefficient for the stated metrics.

Generally, for the multiprocessor architecture the cost function CF is defined as follows

$$CF = \left(\alpha_P * H_P + \alpha_F * H_F + \alpha_C * H_C + \alpha_M * H_M\right) * \\ * \left(1 - \sum_{l=1}^{C} \left|\tilde{S}_{pl}^{-}(d)\right| * \hat{\tau}_{pj}^{-} / T\right) \quad , \qquad (6)$$

where: $\alpha_P, \alpha_F, \alpha_C, \alpha_M$ are the weights of the respective functional units. They are chosen during HW/SW codesign space exploration (section 6.2); in m-processor architectures $\alpha_p * H_p$ includes the total HW size; $\left|\tilde{S}_{pl}^{-}(d)\right|$ is the SW complexity of the segment l for the CPU $_j$ with the maximal CPU $_j$ cycle time $\$_{pj}$; T is the total time for realizing of algorithms $S_{pl}(d)$ in HW/SW implementations.

The task is defined as: for the given system software specification $S_p(d)$ to minimize the function CF under the timing constraint $T \le T^*$.

6. Experimental results

6.1 .Target architectures

As an examples three target architectures of controllers for a solid-state emulator of the floppy-disk were used in experiments with the GSSS system [13]. The first architecture A_1 realizes data ***formatting***, ***processing*** and ***transfering*** without intermediate ***buffering*** ($H_M = 0$). The second one A_2 realizes ***processing*** before writing to and after reading from the memory (***buffering***). In architectures A_1, A_2 a single processor is used. In the third architecture A_3 a distinct processor is used for ***transfering*** with a standard communication protocol.

Am 2900 processor family was used in all examples for CPU building. The SW complexity varied from 10^3 to 10^5 processor cycles with the minimal CPU cycle time 200 ns. The maximal data block was 512 byte with the hypergeometric distribution of data arrivals from 10μs to 230 μs and the confidence probability 0,95. The maximal delay coefficient for CPU with DMA logic was not more than 1,04.

6.2. The HW/SW codesign space exploration

The codesign space exploration is the first stage of performance-complexity estimation in HW/SW partitioning.

During this stage the Pareto optimal sets of system alternatives in hardware size (H) - system performance (T) space are extracted. The next step is variant clustering in accordance to the timing constraint T^*.

Figure 2 shows the HW/SW codesign space for target architectures A_1 , A_2 ,A_3 after clustering with different values of T^*.

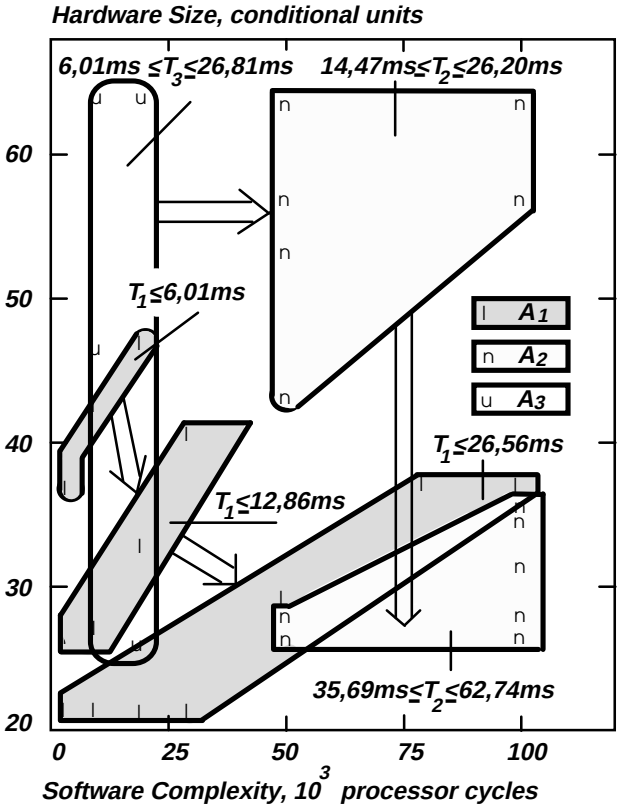

Figure 2: The HW/SW codesign space

One could account that for every feasible value of the SW complexity there are one or more (the Pareto optimal set) variants of the designed system in H-T space.

6.3. The determination of the partitioning direction

This is the second stage of performance-complexity estimation. In experiments we supposed the weights $\alpha_P = \alpha_F = \alpha_C = \alpha_M = 1$ for the explicit extraction of the SW complexity and performance variation during HW/SW partitioning. As Figure 2 shows, under the fixed time constraint T_1, if the SW complexity increases, HW size must be increased for preserving time constraint satisfiability.

Figure 3 (see the next page) shows the HW size portion of different units in the total HW dependence upon the SW run time portion in the total execution time T (that is $\sum_{l=1}^{C} \left| \tilde{S}_{pl}^{-}(d) \right| * \hat{\tau}_{pj}^{-} / T$ in (6)).

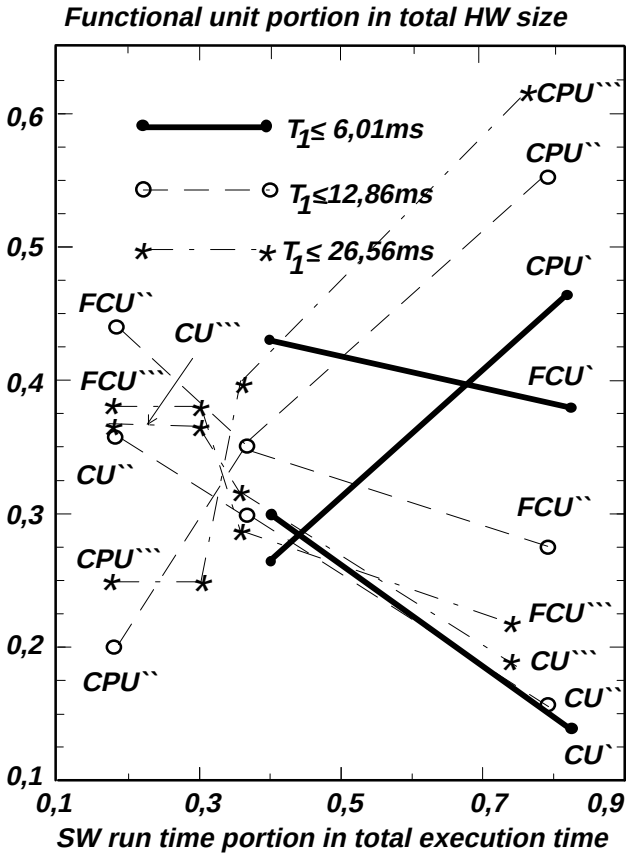

Figure 3: Functional unit portion vs. software run time portion

Coefficients $\alpha_P, \alpha_F, \alpha_C, \alpha_M$ must be adapted during this stage for the CF minimization in (6).

As any acceptable partitioning the supposed approach minimizes the HW-SW communication (the CU portion decreases under software run time portion increasing).

7. Conclusions and future work

The major result of this work is the following. The method of performance-complexity analysis in HW/SW patitioning for real-time systems under timing constraints is suggested.

The distinct features of the method are (a) the rapid performance-complexity estimation for SW based on the set of introduced stochactic characteristics and the SW experimental investigation; (b) the exploration of HW/SW codesign space by the Pareto optimal sets of system variants extraction, that enables to define the partition process direction for the cost function minimization. These features define the adaptive HW/SW partitioning.

The proposed approach will be extended by the RISC processors inclusion (Intel i860, Motorola M88000, Sun SPARC) and the DLX RISC core using for the processing model generalization.

Now the GSSS system is integrated with Vantage Optium™ , version 5.100 containing Styx for adequate performance analysis of total execution time accounting real HW delays.

References

[1] R.K.Gupta and G.De Micheli, "System-Level Synthesis Using Re-programmable Components," *Proc. Third European Conf. Design Automation,* IEEE CS Press,1992, pp.2-7.

[2] R.K. Gupta and G.De Micheli, "Hardware-Software Cosynthesis for Digital Systems," *IEEE Design & Test of Computers*, Sept. 1993, pp.29-41.

[3] R.Ernst and J.Henkel, "Hardware-Software Codesign of Embedded Controllers Based on Hardware Extraction," *Handouts From Int'l Workshop on Hardware-Software Codesign*, Estes Park, Oct. 1992.

[4] R.Ernst, J.Henkel, and T.Benner,"Hardware-Software Cosynthesis for Microcontrollers," *IEEE Design & Test of Computers*, Dec. 1993, pp.64-75.

[5] N.Woo, W.Wolf, and A.Dunlop,"Compilation of a Single Specification into Hardware and Software," *Handouts from Int'l Workshop on Hardware-Software Codesign*, Estes Park, Oct.1992.

[6] E.Barros and W.Rosenstiel, "A Method for Hadware/Software Partitioning," *Proc. Compeuro*, IEEE CS Press, 1992.

[7] E.Barros, W.Rosenstiel, and X.Xiong "A Method for Partitioning UNITY Language in Hardware and Software," *Proc. EURO-DAC'94 with EURO-VHDL'94*, IEEE CS Press, 1994, pp.220-225.

[8] E.D.Lagnese and D.E.Thomas, "Architectural Partitioning for System-Level Design," *Proc. 26th Design Automation Conf.*, IEEE CS Press, 1989, pp.62-67.

[9] F.Vahid, J.Gong, and D.D.Gajski, ”A Binary-Constraint Search Algorithm for Minimizing Hardware during Hardware/Software Partitioning," *Proc. EURO-DAC’94 with EURO-VHDL’94*, IEEE CS Press, 1994, pp.214-219.

[10] J.Gong, D.D.Gajski, and A.Nicolau,“A Performance Evaluator for Parameterized ASIC Architectures,"*Proc. EURO-DAC’94 with EURO-VHDL’94*, IEEE CS Press, 1994, pp.66-71.

[11] K.O'Brien, T.Ben Ismail, and A.A.Jerraya,"A Flexible Communication Modelling Paradigm for System-Level Synthesis," *Int'l Workshop on Hardware-Software Codesign*, Cambridge,Mass, USA, Oct. 1993.

[12] A.A.Jerraya and K.O'Brien, "SOLAR: An Intermediate Format for System-Level Modelling and Synthesis," *in "Computer Aided Software/Hardware Engineering,"* J.Rozenblit and K.Buchenrieder (eds), IEEE Press, 1994.

[13] V.V.Toporkov,“Hardware-Software Cosynthesis by Metaoperator Net," *Proc. VHDL-Forum for CAD in Europe, Fall’94 Meeting at EURO-DAC'94 with EURO-VHDL'94*, pp.17-27.