Introducing Flexible Quantity Contracts into Distributed SoC and Embedded System Design Processes*

Judita Kruse, Clive Thomsen, Rolf Ernst
Institute of Computer and
Communication Network Engineering
kruse|thomsen|ernst@ida.ing.tu-bs.de

Thomas Volling, Thomas Spengler
Institute for Economics and
Business Administration
t.volling|t.spengler@tu-bs.de

Technical University of Braunschweig, D-38106 Braunschweig/Germany

Abstract

Increasing design complexity eventually leads to a design process that is distributed over several companies. This is already found in the automotive industry but SoC design appears to move in the same direction. Design processes for complex systems are iterative, but iteration hardly reaches beyond company borders. Iterations require availability of preliminary design data and estimations, but due to cost and liability issues suppliers often hesitate to provide such preliminary data. Moreover, companies are rarely able to judge the accuracy and precision of externally estimated data. So, the systems integrator experiences increased design risk. Particular mechanisms are needed to ensure, that the integrated system will meet the overall requirements even if part of the early estimations are wrong or imprecise. Based on work in supply chain management, we propose an inter-company design process that is based on formal techniques from real-time systems engineering and so called flexible quantity contracts. In this process, formal techniques control design risk and flexible contracts regulate cooperation and cost distribution. The process effectively delays the design freeze point beyond the contract conclusion to enable design iterations. We explain the process and give an example.

1. Introduction

Efficient design processes and new strategies are needed to meet the challenges of the increasing complexity of current and future embedded system and SoC designs. Industries like automotive engineering and space applications have established sophisticated supplier-integrator chains. We expect that the semiconductor SoC design will follow the same path in SoC development. The success of IP providers such as ARM and their frameworks for inter-company exchange and interoperability of documents and tools [3] are clear indicators

Such supply chains turn out to be the first step towards a distributed design process. Today we already find distributed design processes in automotive engineering and space applications, where software plays an important role. Distributed design processes are spread over several companies and cover the whole design flow from subcomponents to the integration of the overall system.

Increasing SoC complexity, multiple layers of hardware dependent software, application programmer interfaces, and third party software libraries give raise to further design process distribution over a heterogeneous set of players that follow divergent business goals. They must be coordinated via engineering and business mechanisms. It is about time for systematic approaches to that issue that put these mechanisms into context.

Especially in complex product development implementation and integration are done in the late phases of classical design flows (e.g. specified for ESA projects by ECSS standard 'System Engineering' [8]). Hence accurate assertions about design data and objective requirements are late as well. But contracting based on design specifications containing requirements is done at the beginning of a new design. Therefore integrator and suppliers estimate their needs conservatively leading to increased cost and suboptimal products.

One key problem in distributed design processes is to reach true collaborative inter-company developments, while keeping corporate know-how proprietary at the same time.

The approach in this paper was motivated by the SpeAC project in which a large space systems company, EADS Astrium, develops an inter-company design flow with two suppliers and applies it to a concrete satellite hardware/software subsystem development as a demonstrator. As a prerequisite, design data are shared between the companies using restricted access mechanisms to protect IP and management internals.

*This work is supported by a grant from EADS and from the EU in the SpeAC project (MEDEA+ A508).

One of the side effects of this inter-company design flow is the availability of formalized design data using standards such as SystemC[5] for hardware description or SysML[6] for overall system modeling including thermal, power, and mechanical design aspects. It has been shown in the project that the necessary data for the process proposed in the following paper can be formulated and communicated in SysML.

After the discussion of related work in chapter 2, we introduce flexible quantity contracts and their application to distributed SoC and embedded system design processes in chapter 3. We will illustrate our approach with an example given in chapter 4 and conclude in 5.

2. Related work

The realization of a continuous SoC and embedded system design flow is of great concern and is accelerated by the corporate as well as by the science community. Consortia like the VSI Allicane [7] and SPIRIT [3] are founded to enable the development of SoCs with a special focus on configurable predesigned IP-blocks. They specify a catalog of standards, listing essential design data for different groups of components.

Engineering science discusses different approaches to enable an efficient SoC design process like platform or component based strategies [9, 15, 16, 30]. Formal real-time analysis techniques are closely connected to these approaches and address heterogeneous systems especially [24, 21]. To enable advanced real-time analysis techniques particular system models are introduced in [23, 20]. In our paper real-time requirements are used as an example of non-functional system properties. This class of systems is well suited for illustrating problems emerging from the interference of different components.

Requirement management and tracing is an important issue in design processes of complex products. The commercial tool DOORS [2] is widely used in the system industry to achieve this, while the eurostep AP233 demonstrator [1] deals with the task of mapping requirements to system components. Booth are used in the mentioned development project. The interaction of competing requirements is not addressed in this paper. For an overview of current approaches refer to [13].

Contractual agreements have been studied in a variety of academic disciplines, notably economics, engineering, law, and operations management. In contrast to various qualitative contributions (e.g. [11, 14, 25]), quantitative treatment can be found in operations management, particularly in work analyzing supply-chains. Here, settings of at least two independent actors, interlinked by flows of information, physical goods, and financial funds are analyzed [10]. Common premises include asymmetric access to information as well as significant difficulties to monitor other actors' performance (hidden action) [27, 28].

In this context a central phenomenon causing inefficiencies is titled double marginalization and arises in supply-chains where decisions of individuals are driven by their subjective perception of the supply-chain incentive structure. This does not necessarily correspond to the optimal decision basis as compared to a central decision maker [12]. A favorable contractual agreement in this context is one, that at the same time promotes efficiency, Pareto optimality, and incentive compatibility, hereby incorporating individual rationality [29]. A comprehensive review of contract analysis in the field of supply-chain management is provided by [27]. First analysis of contractual issues in software business has been proposed by [22, 29].

For situations of uncertain demand the managerial flexibility to change a chosen course of action is usually associated with better performance [18]. The objective of flexible contracting in this sense is to increase the flexibility of certain entities. Accordingly uncertainty and thus risk is being shared with respect to the contractual agreement in place. Of particular interest for the research presented are contracts with inbuilt flexibility in terms of quantity, so called flexible quantity contracts [26]. Hereby contractual clauses exist, that define conditions under which the exact amount to be purchased may diverge from an initial estimate. Instances of flexible quantity contracts include total minimum and buy-back contracts. However, in order to adopt flexible contracting schemes to the development of microelectronic devices, specific aspects of interdependent design parameters need to be incorporated. To our knowledge yet no contribution exists in this field of analysis.

3. Introducing flexible quantity contracts

3.1. Fixed price versus flexible quantity contracts

Figure 1a outlines a classical distributed design process based on fixed contracts. In case of SoC or embedded system design the *system* S is concretized by a hardware/software system under development.

Initially the integrator performs system partitioning and optimization. Subsequently she formulates her needs regarding the *components* s_i as *requirements* r_i and releases final component specifications at a certain point in time t^* called *design freeze*. Requirements concern functional and non-functional design data and can originate from the systems environment, from technical needs, or from customer requests. Requirements can be classified according to *critical* and *non-critical requirements*, depending on whether the system must or should achieve the associated demands. In fixed contracting scenarios integrators will pass on critical system requirements to components as critical requirements again, to avoid the risk that a critical overall system requirement is missed (see figure 1b).

In a next step the suppliers prepare offers that correspond to the requirements predicting their design data and cost with special emphasis on critical requirements that must be estimated very conservatively. The offer includes these predictions that are passed to the integrator as higher level component *assertions* a_i. With *contracting* at t_0 the suppliers

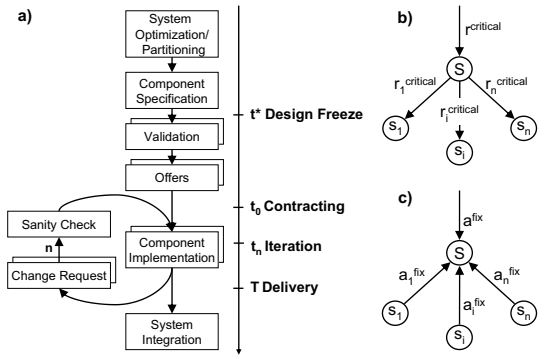

Figure 1. Flow with fixed price contracts

ultimately give *fixed assertions* that their implementations will meet the requirements (figure 1c). By fixing the price p_i of a component s_i, the specific price associated with its assertion a_i is determined by

$$p_i = p(s_i) = p(a_i)$$

In case the integrator refines her requirements during contract negotiations, design freeze and contracting coincide at $t^* = t_0$. Regarding a simple single requirement system consisting of n components and assuming that each component s_i is described exactly by one assertion a_i, the price P is given at t_0 by

$$P = \sum_{i=1}^{n} p(s_i) = \sum_{i=1}^{n} p(a_i)$$

In a subsequent implementation phase, the suppliers take the risk of not complying with assertions, while the integrator's risk is that of changing requirements due to unexpected contingencies, or even worse, of a faulty integration. In case of the necessity to change requirements, usually formal change requests have to be filed.

With growing complexity a series of drawbacks of fixed price contracts can be identified. As subsequent changes of requirements are not explicitly part of the contract (as opposed to the procedure how changes are dealt with, i.e. change management) they consequently are neither foreseeable in terms of cost nor of performance and development time. In this situation rational behavior dictates for the integrator to build in more safety in the system architecture and thus in request for proposals than initial analysis suggests. In addition suppliers benefit from tight requirements, as these represent extra business from their point of view, but at the same time due to an increased exposure to development risk raise their safety margin. The inevitable consequence is the pursuit of over-dimensioned designs.

In fixed contracts potential design trade-offs at system level are limited to the pre-contracting phase ($t < t_0$), resulting in an early design freeze and consequently hampering reactions promoted by better information available. Also trade-off cost information will be private to the individual subjects and thus only, if at all, available to the other actors after renegotiation. Accordingly when conducting design trade-offs, decisions are restricted to local information. Results therefore solely reflect a fraction of the design project's decision basis. This situation corresponds to the mentioned double marginalization phenomenon in supply-chain management. Similar contracting schemes should therefore be applicable.

In the problem setting described, flexible agreements offer an opportunity to both accelerate the decision process and increase its efficiency. The basic idea of flexible contracting in distributed decision making is to on the one hand defer design freeze by extending the period for possible design refinements and on the other to increase the efficiency of decentralized decision making. A basic approach of setting up efficient decentralized process control, as proposed in economics, is to impose the cost structure of the total value-chain (i.e. the design-consortium) on the decision-making entity (i.e. each individual actor), thereby internalizing external effects [19]. One commonly accepted approach in supply-chain management lies in designing more complex contracts by means of contractual baseline and incentive schemes applied [27].

As a prerequisite of adopting a contracting scheme of such kind to design processes, initially suitable measures substituting quantities of physical goods in supply-chain applications need to be identified. In the following we will introduce structured estimation data as a capable measure for flexible contracting.

3.2. Flexible contracts in design processes

To enable flexible contracts in SoC and embedded system design, we propose a *structured estimation format* for assertions and introduce *set-critical* requirements. Assertions in the structured estimation format consist of three values: a conservative estimation a^{gua}, a target a^{tar}, which denotes the baseline, and a best case a^{top}, replacing physical quantities.

The term *set-criticality* will be used for requirements which are critical in conjunction with other requirements, but are not critical by themselfs. This statement can be held even for requirements derived from critical requirements, provided that several combinations of interdependent components with different implementations exist. Obviously any kind of optimization benefits from this fact. Doing so a critical system requirement can be attenuated to set-critical component requirements.

A distributed design process based on flexible contracts is illustrated by figure 2a. The integrator starts with system partitioning and component specification. But in contrast to fixed price contracts she will pass a critical system requirement as set-critical requirements to derived components (figure 2b). The suppliers submit offers with assertions in the structured estimation format $a_i = (a_i^{gua}, a_i^{tar}, a_i^{top})$ instead of delivering only a fixed assertion (figure 2c), along with a corresponding set of prices $p_i = (p_i^{gua}, p_i^{tar}, p_i^{top})$.

Using the flexible contracting scheme comprehensively illustrated in figure 3 the integrator will determine the slope

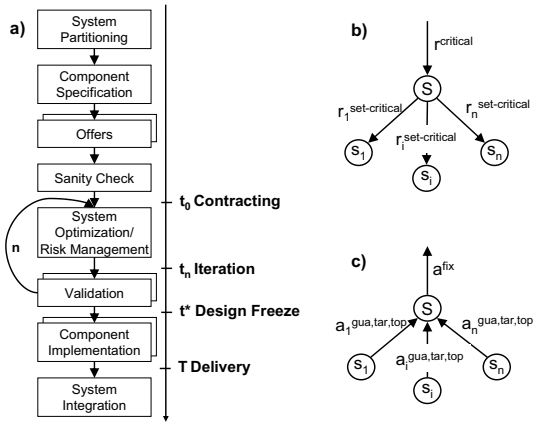

Figure 2. Flow with flexible contracts

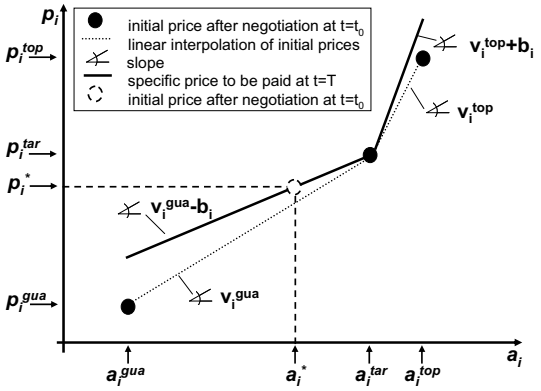

Figure 3. Flexible pricing scheme

parameters v_i^{gua} and v_i^{top} corresponding to linear interpolations. In order to compensate the supplier for the increased exposure to development risk, an additional premium b_i will be paid for deviations to the initial target. From the integrator's point of view, b_i is used as an incentive for the suppliers to direct the development of the system in a suitable way. Applying sensitivity analysis she can identify which components have the largest influence on the critical property and can assign them a higher b_i.

Regarding a single requirement and assuming that each component s_i is described exactly by one assertion a_i, the specific price p_i associated with a_i is determined by

$$p_i(a_i) = \begin{cases} p_i^{tar} + (v_i^{gua} - b_i) \cdot (a_i - a_i^{tar}) \; for \; a_i < a_i^{tar} \\ p_i^{tar} + (v_i^{top} + b_i) \cdot (a_i - a_i^{tar}) \; for \; a_i \geq a_i^{tar} \end{cases}$$

Based on the sets of assertions given by the suppliers, the integrator executes a sanity check, to assure that the critical system requirement will not be violated, according to integrator defined rules. Examples for sanity check metrics are:

The critical system requirement r with a safety margin of $m\%$ will be achieved (with $x\%$ of $a_i^{top} - a_i^{gua}$) if...

one-sided: ...each final component implementation will change the target assertion a_i^{tar} at most by $x\%$.

conservative: ...one-sided, but at most y final component implementations will not improve the guaranteed assertion a_i^{gua}.

optimistic: ...one-sided, but at least y final component implementations will reach the top assertion a_i^{top}.

balanced: ...y of n final component implementations will improve the target assertion a_i^{tar} at least by $x\%$, while $(n-y)$ final component implementations will fail the target assertion a_i^{tar} at most by $x\%$.

weighted: ...the final component implementations will change the target assertion a_i^{tar} by an individual coefficient of $x\%$, e .g. depending on assessments based on former experiences with the particular supplier.

If the sanity check was passed successfully, contracting is done at $t = t_0$ with the offered assertions and their corresponding flexible pricing schemes.

During the subsequent prototyping phase, the suppliers deliver n iteratively refined estimations at $t = t_i$ for the final implementation, but subjected to the boundaries given by a^{gua} and a^{top}, and before a certain point in time t^*. At design freeze the final specification will be determined, and the so far set-critical requirements become critical ones with $r_i = a_i^*$ for a price of $P_i^* = \sum_{i=1}^{n} p(a_i^*)$.

With flexible contracts an efficient strategy is proposed to establish system wide risk management and optimization within the refinement loop of distributed SoC and embedded system design processes. Change requests can be omitted for the most part, because possible variations of implementations have been anticipated by deferring design freeze beyond t_0, the point contracting is done.

4. Example

To test the applicability of our approach, we calculated different scenarios for the example system shown in figure 4. An integrator assigned to design the system partitioned it choosing the following architecture: three functions F_1, F_2, F_3 should be implemented in software, running on a customized CPU. Due to hard deadlines with periodic activations given for the three functions a real-time scheduler is needed. The critical system requirement is $r^{critical}$ ='CPU-utilization $\leq 100\%'$ and can be verified performing schedulability analysis provided, e.g. by SymTA/S[4].

The software shall be subcontracted to a software supplier, and the implementation of the CPU to a hardware supplier respectively. The suppliers include the estimations of the worst case execution times for the software and the clock rate for the CPU along with prices (in mu, i.e. 'money units') as assertions in their offers. Example data are shown in table 1, with the calculated slopes v_i^{gua} and v_i^{tar}. The integrator chooses the scheduling policy a_{sched} ='simple static priority preemptive'[17].

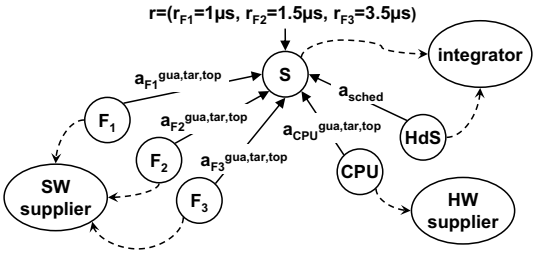

Figure 4. Model of the example system

Table 1. Example data

	a_i^{gua}	p_i^{gua}	a_i^{tar}	p_i^{tar}	a_i^{top}	p_i^{top}	v_i^{gua}	v_i^{top}
F_1	30	10	20	50	10	150	-4	-10
F_2	60	10	40	60	20	180	-2.5	-6
F_3	150	10	100	60	80	120	-1	-3
CPU	50	100	100	200	200	600	2	4

$a_{F1,F2,F3}[cycles], a_{CPU}[MHz], p_i[mu], v_i[\frac{mu}{[a_i]}]$

Depending on the risk she is willing to take, the integrator will pick an appropriate sanity check at $t = t_0$. As the results in table 2 show, the selection is crucial. With an assumed safety margin of 5% the critical requirement 'CPU-utilization' must not exceed 95%. In our example the integrator chooses the balanced check, which passed successful. To find reasonable premiums b_i, a sensitivity analysis tool, as offered by SymTA/S is valuably. Results a_i^{sens} of the sensitivity analysis and the chosen b_i are shown in table 3.

After contracting but before design freeze the suppliers start prototyping and update their estimations frequently for a predefined number of n iterations. As design progresses, the integrator will monitor expected system performance, based on estimations with constantly increasing accuracy. For illustration table 4 shows data for three hypothetical iterations at $t_0 \leq t_1, t_2, t_3 \leq t^*$.

When validation stops, the integrator freezes the design. In the example the critical system requirement was met and a fairly good CPU-utilization was reached. Therefor the integrator will tighten the component requirements r_i from set-critical to critical ones according to the assertions at t_3.

Table 2. Sanity checks

check	$x[\%]$	y	pass	info
one-sided	0	–	y	75.24% CPU-util.
one-sided	-5	–	y	85.4% CPU-util.
one-sided	-10	–	n	105.77% CPU-util.
conservative	0	1	n	failed for CPU
optimistic	-10	1	y	-
balanced	10	2	y	$\frac{n!}{y!\cdot(n-y)!}$ analysis steps
weighted	[1]	–	y	85.4% CPU-util.

[1] $x(F1, F2, F3, CPU)[\%] = (-20, +10, -50, 0)$

Table 3. Sensitivity analysis based on a_i^{tar}

	a_i^{tar}	a_i^{sens}	Δa_i	$b_i[\frac{mu}{[a_i]}]$
$F_1[cycles]$	20	39	19	-3
$F_2[cycles]$	40	69	29	-2
$F_3[cycles]$	100	169	69	-0.5
$CPU[MHz]$	100	80	20	1

Table 4. Risk management

t	a_{F1}	a_{F2}	a_{F3}	a_{CPU}	CPU-util.$[\%]$
t_1	21	55	120	100	91.95
t_2	28	42	135	100	94.57
t_3	21	37	150	100	88.52

$a_{F1,F2,F3}[cycles], a_{CPU}[MHz]$

The associated prices can be found in table 5 with calculated prices for a fixed price scenario. The fixed price values are based on the assumption, that the integrator would have called for bids on critical requirements equal to the target implementation, which leaves a significant safety margin regarding CPU-utilization. The comparison shows that a fair value-chain was obtained with a win-win situation for all actors. Supplier of F_1 receives slightly less, but avoids change requests due to a missed critical requirement. Supplier of F_2 is rewarded for her effort, while supplier of F_3 earns significantly less, though is prevented from potential contractual penalty. From the integrator's perspective the development of the system is less expensive without an oversized dimensioning.

Employing flexible contracts with estimations the integrator gains an extended period for design trade-offs. Consequently cost reductions can be achieved from the integrators perspective. Considering the suppliers, an overall loss in business can be identified in some cases. However incentive compatibility is created, because an extra premium is paid, due to the compensation scheme. Moreover broader requirements are likely to reduce the suppliers internal costs. Consequently applying the flexible contracting scheme an enhanced design process is facilitated, complying with Whang's criteria for optimal contracting [29].

5. Conclusion

Inter-company design processes gain importance. Contractual constraints and IP protection issues entail an overly

Table 5. Pricing

	F_1	F_2	F_3	CPU	$\sum_{i=1}^{n} p_i(a_i)$	CPU-util
p_i^*	49	84	35	200	368	88.52
p_i^{fix}	50	60	60	200	370	75.24

$a_{F1,F2,F3}[cycles], a_{CPU}[MHz], p_i[mu]$

conservative design style. Motivated by a satellite component design project, a design process based on flexible quantity contracting and formal analysis was proposed that reduces overdesign but allows to control design risk. Redesign cycles can be reduced, because critical component requirements can be traded off against each other. A simple example shows the incentives for the different players in the design process and demonstrates fairness.

6. Acknowledgement

We would like to thank Roland Müller and others from Astrium, Hubert Stich from EuroTelematik, and Peter Ganal from Tecnotron for their valuable input and feedback from design practice.

References

[1] AP233. http://ap233.eurostep.com/.

[2] DOORS. http://www.telelogic.com/products/doorsers/doors/.

[3] SPIRIT Consortium. http://www.spiritconsortium.com/.

[4] SymTA/S. http://www.symta.org/.

[5] SystemC. http://www.systemc.org/.

[6] Systems Modeling Language. http://www.sysml.org/.

[7] Virtual Socket Interface Alliance. http://www.vsi.org/.

[8] *ECSS-E-10A Standard, System engineering*. European Cooperation for Space Standardization, Apr. 1996.

[9] H. Chang, L. Cooke, M. Hunt, G. Martin, A. McNelly, and L. Todd. *Surviving the SOC Revolution*. Kluwer Academic Publishers, 1999.

[10] M. Christopher. *Logistics and supply chain management*. Financial Times/Prentice Hall, London, 1998.

[11] R. Cooter and T. Ulen. *Law and economics*. Pearson Addison-Wesley, Boston, 4. ed edition, 2004.

[12] C. J. Corbett and C. S. Tang. Designing supply contracts. In S. Tayur, R. Ganeshan, and M. Magazine, editors, *Quantitative models for supply chain management*, pages 269–298. Kluwer Academic Publishers, Boston, 1999.

[13] A. Dardenne, A. van Lamsweerde, and S. Fickas. Goal-directed requirements acquisition. In *Selected Papers of the 6th Int. Workshop on Software Specification and Design*, pages 3–50. Elsevier Science Publishers B. V., 1993.

[14] S. Kawasaki and J. McMillan. The design of contracts: evidence from Japanese subcontracting. *Journal of the Japanese and international economies*, 1(3):327–349, 1987.

[15] K. Keutzer, S. Malik, A. R. Newton, J. M. Rabaey, and A. Sangiovanni-Vincentelli. System-level design: Orthogonalization of concerns and platform-based design. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 19(12):1523–1543, 2000.

[16] H. Kopetz. Component-based design of large distributed real-time systems. In *14th IFAC Workshop on Distributed Computer Control Systems (DCCS'97)*, pages 171–177, Seoul, Korea, 1997.

[17] C. L. Liu and J. W. Layland. Scheduling algorithm for multiprogramming in a hard-real-time environment. *Journal of the ACM*, 20, 1973.

[18] A. MacCormack, R. Verganti, and M. Iansiti. Developing products on 'internet time'. *Management science*, 47(1):133–150, 2001.

[19] G. F. Mathewson and A. Winter. An Economic Theory of Vertical Restraints . *RAND Journal of Economics*, 15(1):27–38, 1984.

[20] J. C. Palencia, J. J. G. Garcia, and M. G. Harbour. Best-case analysis for improving the worst-case schedulability test for distributed hard real-time systems. In *Proc. 10th Euromicro Workshop on Real-Time Systems*, page 35, Berlin, Germany, June 1998.

[21] P. Pop, P. Eles, and Z. Peng. Bus access optimization for distributed embedded systems based on schedulability analysis. In *Proc. Design, Automation and Test in Europe (DATE'00)*, Paris, France, 2000.

[22] W. B. Richmond and A. Seidmann. Software development outsourcing contract. *Journal of management information systems*, 10(1):57–72, 1993.

[23] K. Richter, M. Jersak, and R. Ernst. A formal approach to MpSoC performance verification. *IEEE Computer*, 36(4), Apr. 2003.

[24] S. K. S. Chakraborty and L. Thiele. A general framework for analysing system properties in platform-based embedded system designs. In *Proc. Design, Automation and Test in Europe (DATE'03)*, Munich, Germany, Mar. 2003.

[25] L. Telser. A theory of self-enforcing agreements. *Journal of Business*, 53:27–44, 1980.

[26] A. A. Tsay. The Quantity Flexibility Contract and Supplier-Customer Incentives. *Management science*, 45(10):1339–1358, 1999.

[27] A. A. Tsay, S. Nahmias, and N. Agrawal. Modeling Supply Chain Contracts - A Review. In S. Tayur, R. Ganeshan, and M. Magazine, editors, *Quantitative models for supply chain management*, pages 299–336. Kluwer, Boston, 1999.

[28] E. T. Wang, T. Barron, and A. Seidmann. Contracting Structures for Custom Software Development: The Impacts of Informational Rents and Uncertainty. *Management science*, 43(12):1726–1744, 1997.

[29] S. Whang. Contracting for Software Development. *Management science*, 38(3):307–324, 1992.

[30] T. Zhang, L. Benini, and G. D. Micheli. Component selection and matching for IP-based design. In *Proc. Design, Automation and Test in Europe (DATE'01)*, pages 190–195, Munich, Germany, Mar. 2001.