Towards Flexible Systems Engineering by Using Flexible Quantity Contracts*

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

With increasing design complexity flexible systems engineering becomes a challenging issue in SoC and embedded system design. Furthermore the heterogeneity of different components of a single system leads to a trend to distribute the development process over several companies. Today we already find such distributed design processes in automotive engineering and space applications, where software plays an important role. We expect a similar development in SoC design.

We propose an approach that introduces flexible quantity contracts into distributed SoC design processes. Flexible quantity contracts are well known in operations management, particularly in research on supply chains. Applied to SoC design, these contracts will lead to an extension of the period design trade-offs can be conducted to realize a significantly better overall outcome. To enable flexible quantity contracts we propose a structured data format for estimated design data. By using structured estimation design data, combined with flexible quantity contracts we expect improved design productivity in inter-company design processes.

Related Topics
Development processes and their improvements

Keywords
SoC Design, Estimation Data, Flexible Quantity Contracts, Incentive Schemes

1. Introduction

Nowadays embedded systems play an increasingly important role in the design of airplanes, road and rail transportation systems. But the development processes of embedded system and SoCs are getting more and more dominated by the increasing complexity of the design.

Due to the fundamental demands on safety aspects in the automotive engineering and space industries, embedded systems are subject to hard real time constraints. Current approaches allow the verification of heterogeneous embedded hardware-software systems by using real time analysis. These technics rely on design data which describe the properties of the system under development.

However, accurate design data is not available until near completion of implementation. Therefore, system analysis for specification and contracting in the early phases of the design process can only be done based on estimation data.

Design processes are iterative, but iteration cycles hardly reach beyond company borders. Iterations require availability of preliminary design data, but due to liability clauses in legal contracts suppliers often hesitate to provide preliminary design data. In addition, to minimize individual risks, estimated values are always conservative and add up in an oversized design.

A recent study revealed that among the top ten problems encountered in collaborative design projects three were directly related to contractual issues. These were lengthy discussions on contract price elements, complex communication interfaces, and hidden specifications [29]. What seems to be needed to cope with the increased complexity of future embedded system design, is an approach that dynamically directs development effort toward critical development tasks. In particular, analyzing collaborative settings the question arises, how to organize such development processes.

When subcontracting is done today, the integrator formulates her needs concerning functional and non-functional design data as requirements, while suppliers determine assertions concerning the values guaranteed to be reached. The integrator takes the risk that requirements change during the development process, while the risk of a supplier is to break assertions made. In the case of the necessity to

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

change fixed design data, change requests have to be filed.

We propose a structured data format consisting of a set of estimations denominating a guaranteed, a target and a top value, to describe the properties of system components. Such a set of values gives the system integrator a higher flexibility for design specification and mapping. But greater freedom allowed in the specification, requires a mechanism to direct the development of the components towards a stable overall system. We will therefore introduce a differentiated contracting scheme to provide an efficient coordination.

After the discussion of related work in chapter 2, we describe our system model in 3 and extend flexible quantity contracts to distributed SoC and embedded system design processes in chapter 4. We will illustrate our approach with an example given in chapter 5 and conclude in 6.

2. Related work

2.1 SoC and embedded system design

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 [12, 17, 18, 31]. Formal real-time analysis techniques are closely connected to these approaches especially addressing heterogeneous systems [24, 22]. To enable advanced real-time analysis techniques particular system models are introduced in [23, 21]. 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. For an overview of current approaches refer to [14]. The commercial tool DOORS [2] is widely used in the system industry to achieve requirement management, while the eurostep AP233 demonstrator [1] deals with the task of mapping requirements to system components.

There are some approaches of defining an appropriate language for system design. SystemC [5] emerged from the idea of integrating hardware (VHDL) and software description (C++) to enable rapid simulation, whereas SysML [6] is influenced by the UML and system engineering community.

Embedded system and SoC design is increasingly interrelated with the system integrators industries. As an example, the space industry started research on possible SoC solutions a few years ago [16] and has developed meanwhile a series of SPARC compliant cores called LEON. The next generation LEON3 is currently announced [9]. Accordingly, the alignment of the SoC design flow with the established distributed overall system design flow is of high concern.

2.2 Introducing the supply chain management perspective

The collaborative product development process will be defined as the set of inter-linked tasks which is executed by independent actors and aims at the transformation of a business opportunity into a product for sale [26]. What results is a network of organizations that is closely coupled by up- and downstream linkages. This network is called supply chain, the corresponding task of coordination and integration supply chain management [13].

As work is being distributed at the same time decision making is. We will therefore refer to the collaborating actors as decision making units [25]. Accordingly several aspects of decentralized control arise with coordination of those decision making units being one central issue. Without coordination decisions are restricted to locally available information, which represent an isolated view on the supply chain and are consequently likely to cause inefficiencies up- and downstream. By using the appropriate contract to regulate the parameters of the interaction it is possible to set up efficient decentralized control as the cost structure of the total value chain can be imposed on each decision making unit involved in the process [28]. Thereby contracts may contribute to less complex interactions as well as congruent goals, as the individual objectives are aligned towards a global one. In order to do so, behavioral aspects like locally perceived incentives and risk need to be taken into consideration in a manner, that all participants are left with their own best interests [27]. A favorable contractual agreement is consequently one, that at the same time promotes efficiency, pareto optimality and incentive compatibility, while incorporating individual rationality [30].

The analysis and design of contracts makes up a central research objective of supply chain management literature. For recent reviews refer to [11] and [28]. Particularly contracts or more precisely contract parameters i.e. incentive schemes are looked for, that direct individual actions towards a globally desired outcome. Incentive schemes in this understanding are defined as the relation between the transfer paid to a contractor and the contractors performance [20].

Incentive contracts have a long tradition for example in the avionics industry and in public procurement [8]. However, lacking further information usually tangible measures such as schedule and total costs are being employed for rewarding contractors. Albeit clearly correlated to an efficient design, quality measures are due to their rather intangible nature not taken into account in most practical applications. Especially, to the authors knowledge no integrated approach exists, that simultaneously analyzes component design and contracting issues.

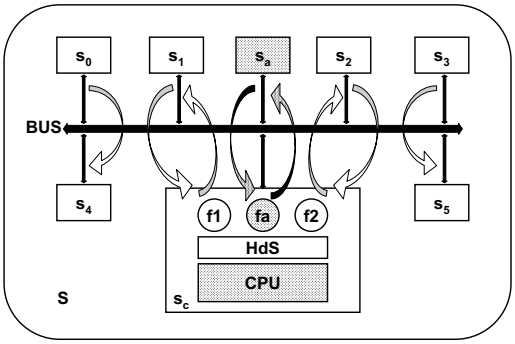

Figure 1. Example System

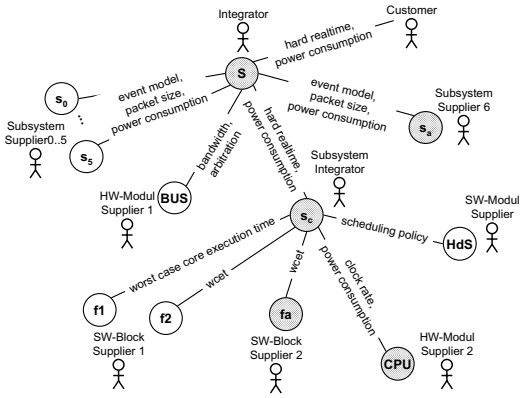

Figure 2. System model with actors

3. System model

In our approach a hierarchical system model is used, which reflects the system structure as well as the supply chain. A system is usually partitioned into *subsystems*, *modules and blocks*, collectively referred to as *components* s_i. We distinguish between components by granularity. A *block* is the smallest entity considered and often directly linked to a particular implementation like a single VHDL or C source. A *module* denominates a more complex component, for example a generic core. Modules should either be hard- or software. Components covering a hardware/software system or a 'pure' but very complex hardware or software system are called *subsystems*. A subsystem could contain other subsystems, modules and blocks, while modules can only be composed of blocks.

Figure 1 shows an example system with several subcomponents connected to a bus. Communication is indicated by the curved arrows. s_0 to s_5 and s_a are black-boxed to subsystems, whereas s_c is broken down into three software blocks f_1, f_2 and f_a, the hardware module CPU and a software module HdS, which in general stands for hardware dependent software and refers to the scheduler in the example. The appropriate system model along with the actors of the supply chain is shown in figure 2.

Inherently, each component has a large number of functional and non-functional properties, e.g. timing, power dissipation, thermal stress resistance or even the color of a LED. In distributed system development all these properties and their characteristics need to be defined by contracts. A couple of non-functional properties of the example system is annotated in figure 2.

Requirements r characterize the properties a component must achieve. They can originate from the environment of the system, from technical needs or from customer requests. Derived requirements on a subcomponent are obtained from higher level requirements.

Assertions a are properties components guarantee to other components. The higher level component can simulate or analyze the behavior of the subcomponents based on the assertions given and can assert a specific behavior to the next higher level on its part.

In fixed price scenarios requirements are firmly specified by contracts and for this reason all assertions are fixed. Penalty clauses assure that important properties are reflected by critical requirements. This determination of critical requirements and assertions is a time-consuming task for suppliers as for integrators in early specification phases. Due to conservative estimations of either actor the overall system is likely to end up in an over-estimated design. In distributed development environments with fixed contracts this fact cannot be neglected, since the over-estimation adds up with every other supplier.

To enable flexible contracts in distributed embedded system design, we proposed a *structured estimation format* for assertions and introduced *set-critical* requirements in [19]. The term *set-criticality* is used for requirements which are critical in conjunction with other requirements, but are not critical by themselfs. Assertions in the structured estimation format consist of three values: a conservative estimation, a target, which denotes the expected, and a best case:

$$a = (a^{gua}, a^{tar}, a^{top})$$

In many cases more than one requirement concern a single physical parameter. If all associated requirements force the parameter in the same direction, they strengthen each other. However, often they are competing.

A typical example regards a CPU which should be faster for performance requirements, but at the same time low power dissipation is needed. Due to

$$P = C_{gates} \cdot f \cdot V_{CC}^2$$

fulfilling a timing requirement r^T will always decrease the assertable low power requirement r^P for the very same component (i.e. same technology, same core voltage).

Hence, we extend the structured estimation format for assertions regarding competing requirements as follows. All competing requirements of a component are prioritized. The assertion for a lower priority requirement is estimated as a function of the higher priority one. Let r^T be prior to

r^P. The related assertions for a component i will be given by

$$a_i^T = (a_i^{T,gua}, a_i^{T,tar}, a_i^{T,top})$$
$$a_i^P = (f^{P,gua}(a_i^T), f^{P,tar}(a_i^T), f^{P,top}(a_i^T))$$

4. Contract structure and design

The objective of the following is to provide a framework for the design of contractual agreements with respect to the setting introduced above. For our analysis we assume that specific cost estimates p^{gua}, p^{tar}, and p^{top} are available for a three point interval of a property defined by a^{gua}, a^{tar}, and a^{top}.

From an economic perspective a universal representation of a component development contract is given by

$$P_s = P_s^{fix} + \sum_{J=1}^{m} p^J(a^J)$$

where P_s refers to the transfer to be paid to a vendor contracted to develop a component s. P_s^{fix} denotes a fixed price part while $p^J(a^J)$ are m specific price parts as a function of asserted properties a^J for s. More specifically the latter term is comprised of two summands, a fixed baseline $p^{J,base}$ and $p^{J,inc}$ denoting the incentive scheme applied.

$$p^J(a^J) = p^{J,base}(\alpha^J) + p^{J,inc}(a^J)$$

For applicability reasons we determine $p^{J,base}(\alpha^J)$ such as $\alpha^J = 100$ sets $p^J(a^{J,gua})$ equal to $p^{J,gua}$, whereas $\alpha^J = 0$ results in $p^J(a^{J,tar})$ equaling $p^{J,tar}$ no matter what incentive parameters are chosen.

$$p^{J,base}(\alpha^J) = -\frac{\alpha^J}{100} \cdot b^{J,gua} \cdot (a^{J,tar} - a^{J,gua})$$

Finally the incentive scheme is given by

$$p^{J,inc}(a^J) = \begin{cases} p^{J,tar} + (v^{J,gua} + b^{J,gua}) \cdot (a^J - a^{J,tar}) \\ \qquad\qquad for a^J < a^{J,tar} \\ p^{J,tar} + (v^{J,top} + b^{J,top}) \cdot (a^J - a^{J,tar}) \\ \qquad\qquad for \;\; a^J \geq a^{J,tar} \end{cases}$$

The parameters $v^{J,gua}$ and $v^{J,top}$ can be interpreted as the slope of the linear interpolation between $a^{J,gua}$, $a^{J,tar}$, and $a^{J,top}$ and their associated prices. Compensative payments granted to the supplier are represented by $b^{J,gua}$ and $b^{J,top}$. Penalties for assertions worse than the target value can be adjusted using α^J

We will further distinguish three characteristic instances comprehensively illustrated in figure 3 for a 'the-more-the-better' scenario: *fixed price, quantity flexible,* and *incentive* contracts. We thereby restrict our analysis to piecewise linear contracts as they stand for most real-world applications [20]. The resulting prices are shown as a function of the assertions made at design freeze. As an additional reference the linear interpolations of the initial cost estimates are given by the dotted lines. These may serve as a simple approximation of the price to be paid, if selecting a fixed price contract.

Fixed price contracts are due to their transparent structure widely used across industries. In this case the contract is given by two parameters for each requirement: a fixed assertion a^{tar} and a corresponding price p^{tar}. With $\alpha = 0$ deviations to the specified value are either not permitted or not rewarded (see figure 3a). The contractor is residual claimant for any cost savings but also carries the entire risk of cost overruns. Thus what influences the contractors decision making is his local i.e. private cost information. The cost structures of other players are not taken into account.

In order to improve overall supply chain performance several approaches like *flexible quantity contracts* [10] or backup agreements [15] have been discussed. An example of a flexible quantity contract applied to a distributed embedded system design is given in [19]. Here, analogue to the fixed price buyer and supplier agree on a design target and a target price. Additionally a bonus is defined, if the buyer requires more or less than the target. The allowed deviations are restricted to an interval narrowed down by an upper boundary a^{top} and a lower boundary a^{gua} (figure 3b). The intention of the bonus is to compensate the supplier for an increased exposure to design risk, as she is obliged to accept reduced overall business, if stipulated by the integrator. Accordingly changes to the initial agreement are not excluded, but are associated with higher specific costs for the buyer. Still, as they result in net savings, reduced requirements might be desirable for the integrator if backed by improved system analysis.

Incentive contracts finally represent the most complex contract instance. In addition to the flexible quantity case a fixed baseline bonus p^{base} is granted. They are particularly well suited for settings where uncertainty prevails in terms of the achievement of design goals represented by a^{tar}. To motivate better performance it is rewarded with a higher transfer. Specifically the parameters α, b^{gua}, and b^{top} are used to define the power of the incentive scheme (figure 3c).

In assessing the fitness of a contractual scheme for a certain component property essentially three decisions can be employed. The criticality of a property is affected by the environment, internal system effects or customer demands. A sensitivity analysis can be used to determine this measure. For properties with associated requirements, which can be attenuated to set-critical ones, incentive or flexible quantity schemes are a good choice.

The trade-off relevance arises for set-critical attributes. A high trade-off relevance indicates, that a better performance facilitates broader requirements with respect to other properties. Global cost savings could therefore be obtained. In this case incentive contracts motivating better perfor-

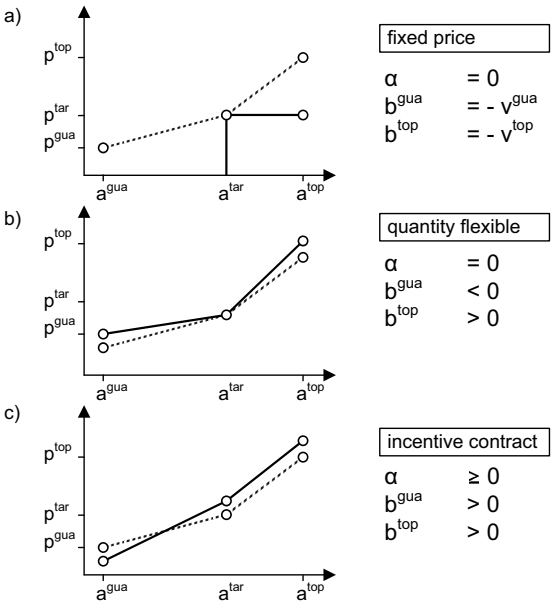

Figure 3. Pricing schemes

mance should be used.

The third evidence is the scalability of a property. Scalability refers to the technical feasibility of changes to a particular parameter merely restricted to given boundaries. Examples for scalable attributes are the number of I/O-interfaces or the size of the memory. Obviously flexible quantity contracts are not applicable, if the attribute at hand is not scalable. Consequently, fixed price contracts are chosen for critical properties which cannot be attenuated to set-critical ones. For scalable attributes on the contrary flexible quantity contracts induce a higher flexibility without significant drawbacks and should therefore be pursued.

The actual transfer paid to a contractor is finally determined by applying the general contract representation introduced above. The challenge of contract design lies in determining well balanced contracts. That is, as better performance is rewarded in the incentive contract case, flexible quantity clauses are needed to adopt to the changed system configuration without encountering massive cost overruns.

5. Example

To illustrate our approach we take the example system from figure 1. An already existing system consisting of s_0 to s_5 and s_c and the bus shall be extended by the additional functionality of the pair of components s_a and f_a. Optionally, the hitherto used CPU can be displaced by an alternative with the same instruction set.

The properties of the initial system are listed in table 1. The 8bit system bus operates at 33MHz with a load of 40.24%. s_0 to s_5 consume 10mW each. s_c is the relevant component for system power consumption and consumes

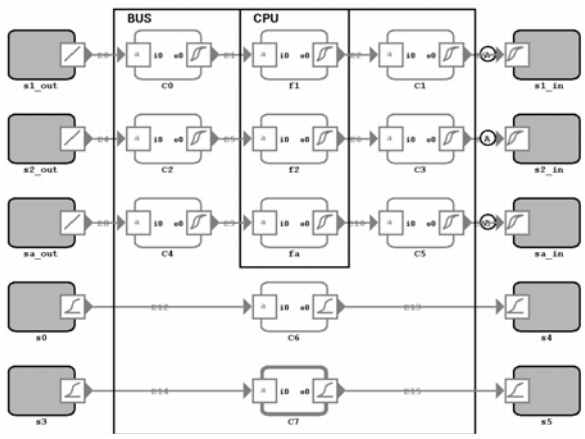

Figure 4. SymTA/S model for revisioned system

255mW at 50MHz and has a load of 80.00%. The subsystems communication behavior with end-to-end deadlines and packet sizes can be found in table 1, also. Overhead is not regarded, neither for the CPU nor for the bus.

The additional component s_a will produce a packet of 8 byte periodically. This packet shall be send over the bus to an additional task f_a running on the CPU of s_c. The resulting packet of 8byte will be send back to s_a over the bus again. The end-to-end deadline of the path from s_a to s_c and back to s_a shall equal the period of s_a.

The development of s_a will be subcontracted to a hardware-module supplier, while the implementation of f_a will be subcontracted to a software supplier. A bid for alternative CPUs will be solicited from another hardware supplier.

Hard real-time behavior and a maximal power consumption of 320mW are critical overall system requirements. Power consumption and CPU speed are competing requirements concerning the physical parameter clock-frequency. For this reason we priorize timing prior to power.

The real-time analysis tool SymTA/S[4] is used for timing analysis of the overall system bus as well as for the local analysis of the subsystem s_c. An equivalent SymTA/S model of the example system is shown in figure 4. To analyze the bus SymTA/S needs event stream models for s_0 to s_5, s_a and s_c, descriptions of the communication tasks c_0 to c_7, the bus-speed, and the arbitration. For the scheduling analysis of the CPU, the scheduling policy, the clock-rate, and worst case execution times for f_1, f_2 and f_a are needed.

Every component of the initial system provides its initially analyzed timing properties as fixed assertions to the system. s_a, f_a and the CPU are set-critical concerning the timing requirement and provide assertions in the structured estimation format. Since s_a and f_a are not implemented yet, the worst case execution time for f_a and the event models for s_a have to be estimated. The hardware supplier has reliable informations on possible clock-rates

Table 1. Initial system properties

component	timing	power
S	busload 40.24%	system power consumption 315mW
BUS	33MHz,8bit,priority based arbitration, overhead not regarded	power consumption not regarded
s_0	produces 64byte with period $P = 100\mu$s, jitter $J = 0.02\mu$s	subsystem power consumption 10mW
s_1	produces 4byte with period $P = 1\mu$s, jitter $J = 0$ and consumes 4byte with period $P = 1\mu$s, jitter $J = 1\mu$s, requires end-to-end deadline of 1μs	subsystem power consumption 10mW
s_2	produces 4byte with period $P = 2\mu$s, jitter $J = 0$ and consumes 4byte with period $P = 2\mu$s, jitter $J = 2\mu$s, requires end-to-end deadline of 2μs	subsystem power consumption 10mW
s_3	produces 32byte with period $P = 50\mu$s, jitter $J = 0.02\mu$	subsystem power consumption 10mW
s_4	consumes 64byte sporadically, with period $P = 100\mu$s, jitter $J = 50\mu$s	subsystem power consumption 10mW
s_5	consumes 32byte sporadically, with period $P = 50\mu$s, jitter $J = 25\mu$s	subsystem power consumption 10mW
s_c	cpuload 80%	CPU power consumption 255mW
f_1	$20cycles$	no back-annotation used for software
f_2	$40cycles$	no back-annotation used for software
HdS	static priority preemptive, overhead not regarded	no back-annotation used for software
CPU	50MHz	idle:75mW, busy:300mW

Table 2. Pricing schemes applied

component	α_i	b_i^{gua}	b_i^{top}	pricing scheme
$r^T_{s_a}$	50	0.5	0.5	incentive
$r^P_{s_a}$	0	-	-	fixed
$r^T_{f_a}$	50	-1	-1	incentive
r^T_{CPU}	0	-0.5	0.5	flexible quantity
$r^{P_{idle}}_{CPU}$	25	-0.2	-0.2	incentive
$r^{P_{busy}}_{CPU}$	25	-1	-1	incentive

for the CPU. Therefore, the integrator chooses an incentive pricing scheme for s_a and f_a, whereas flexible quantity is used for the CPU. The timing assertions and corresponding prices of the additional components are shown in table 3, along with assumed incentive parameters. For the example the fixed price part of a component development contract is regarded as zero. Table 2 summerizes the applied pricing schemes.

For the power consumption of S a simplified calculation is used:

$$P = \sum P_i(s_i)$$

Table 3. Timing assertions

component	$a_i^{T,gua}$	$p_i^{T,gua}$	$a_i^{T,tar}$	$p_i^{T,tar}$	$a_i^{T,top}$	$p_i^{T,top}$
$s_a[\mu s]$	4.7	10	5.5	60	6.6	120
f_a[cycles]	150	80	100	120	80	300
CPU[MHz]	50	50	100	100	200	300

The power consumption of s_c is estimated by the following formula:

$$P = P_{idle} \cdot (1 - Utilization_{busy}) + P_{busy} \cdot Utilization_{busy}$$

For power requirements we assume, that s_a asserts a fixed power consumption of 10mW. We do not regard the power consumption of the bus. Therfore $r_s^{P,critical}$ can be passed to s_c as $r_s^{P,critical} = 250mW$. Since the timing requirement r^T is prior to the power requirement r^P, the power assertions are provided in dependence of the timing requirement. See table 4 for the details.

An initial system analysis based on the target values for the additional components and the so far used CPU reveals that such a system does not have the required real-time capability (table 5.1). Analysis number two indicates that the

Table 4. Power assertions for CPU

component	$a_{CPU}^{P,gua}$	$a_{CPU}^{P,tar}$	$a_{CPU}^{P,top}$
$f^{P_{idle}}(a^{T,gua})$[mW]	75	60	40
$f^{P_{busy}}(a^{T,gua})$[mW]	300	200	170
$f^{P_{idle}}(a^{T,tar})$[mW]	100	75	50
$f^{P_{busy}}(a^{T,tar})$[mW]	400	240	230
$f^{P_{idle}}(a^{T,top})$[mW]	120	100	65
$f^{P_{busy}}(a^{T,top})$[mW]	550	400	330
$p_{CPU}^{P_{idle}}$[mu]	25	50	150
$p_{CPU}^{P_{busy}}$[mu]	25	50	150

chosen target set-up is appropriate: both, the timing as well as the power requirement is met (table 6.2). Setting all components to the guaranteed value, except the CPU running at the target point, will result in a system violating both requirements (table 5.3). The negative overall system price results from the incentive parameters used, which enable penalties within incentive pricing contracts. However, as the timing properties of the CPU was contracted with flexible quantity, the integrator can employ the CPU with the top clock-rate to obtain a stable overall system (table 5.4). Due to the low transfers for the components not improving the guaranteed value, the overall system price is similar to a typical target system. The other corner case is calculated using the analysis set number five (table 5.5). s_a and f_a reach their top points and the system price exceeds the target system price more than twice. As analysis number six shows, using a CPU at guaranteed clock-rate instead is of no option, because the CPU would violate its timing requirement. Furthermore, the system price could even increase, if the hardware supplier has enhanced the power consumption properties of the chip (table 6.6). The last analysis example indicates a possible solution, assuming that the hardware supplier offers not only CPUs with the corner point frequencies.

6. Conclusion

Flexible Systems Engineering becomes a more and more important issue in embedded system design processes. A system model was described to capture requirements and assertions of a complex system in a distributed design environment. The concept of structured estimation data was extended to capture competing requirements as well. An incentive mechanism was added to the flexible contracting scheme and a decision matrix was given.

Our further work is on research of how dependencies in SoCs can be determined and traced. In addition we study, how the structured estimation data model can be extended by assessment methods.

7. Acknowledgment

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] Federal Aquisition Regulations - Volume 1, 2001. General Services Administration and Department of Defense and National Aeronautics and Space Administration.
[9] Leon3 processor core, Oct. 2004. Gaisler Research AB.
[10] R. Anupindi and Y. Bassok. Supply Contracts with Quantity Commitments and Stochastic Demand. In S. Tayur, editor, *Quantitative models for supply chain management*, pages 198–232. Kluwer, Boston, 1999.
[11] G. P. Cachon. Supply chain coordination with contracts. In A. G. d. Kok and S. C. Graves, editors, *Supply chain management*, pages 229–339. Elsevier, Amsterdam, 2003.
[12] H. Chang, L. Cooke, M. Hunt, G. Martin, A. McNelly, and L. Todd. *Surviving the SOC Revolution.* Kluwer Academic Publishers, 1999.
[13] M. Christopher. *Logistics and supply chain management*. Financial Times/Prentice Hall, London, 1998.
[14] 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.
[15] G. D. Eppen and A. V. Iyer. Backup Agreements in Fashion Buying. *Management science*, 43(11):1469–1484, 1997.
[16] S. Habinc. Design space applications using synthesisable cores. In *Proc. 2nd Military & Aerospace Applications of Programmable Devices & Technologies Int. Conf.(MAPLD'99)*, Maryland, USA, Sept. 1999.
[17] 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.
[18] 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.
[19] J. Kruse, T. Volling, C. Thomsen, R. Ernst, and T. Spengler. Introducing flexible quantity contracts into distributed soc and embedded system design processes. In *Proc. Design, Automation and Test in Europe (DATE'05)*, Munich, Germany, 2005.
[20] J.-J. Laffont and J. Tirole. *A theory of incentives in procurement and regulation*. MIT Press, Cambridge, 5 edition, 2002.

Table 5. Timing analysis results

no.	set-up	$a^T_{f_a}$	$p(a^T_{f_a})$	$a^T_{s_a}$	$p(a^T_{s_a})$	a^T_{CPU}	$p(a^T_{CPU})$	$a^T_{S_c}$	critical requirement	a^T_S	$\sum p_i(a^T_i)$
1	a^{gua}_{CPU}, s^{tar}_a, f^{tar}_a	100	95	5.5	80	50	75	116.36	CPU violated	49.06	250
2	a^{tar}_{CPU}, s^{tar}_a, f^{tar}_a	100	95	5.5	80	100	100	58.18	ok	49.06	275
3	a^{tar}_{CPU}, s^{gua}_a, f^{gua}_a	150	5	4.7	-100	100	100	71.91	deadline s_a violated	50.56	5
4	a^{top}_{CPU}, s^{gua}_a, f^{gua}_a	150	5	4.7	-100	200	350	35.96	ok	50.56	255
5	a^{tar}_{CPU}, s^{top}_a, f^{top}_a	80	295	6.0	250	100	100	53.33	ok	48.32	645
6	a^{gua}_{CPU}, s^{top}_a, f^{top}_a	80	295	6.0	250	50	75	106.67	CPU violation	48.32	620
7	a^{75MHz}_{CPU}, s^{top}_a, f^{top}_a	80	295	6.0	250	75	87.5	71.11	ok	48.32	632.5

$a^T_{f_a}[cycles]$, $a^T_{s_a}[\mu s]$, $a^T_{CPU}[MHz]$, $a^T_{S_c}[\%]$ =CPU-Util., $a^T_S[\%]$=BUS-Util., $p^T_i[mu]$

Table 6. Power analysis results

no.	set-up	$a^{P_{idle}}_{CPU}$	$p(a^{P_{idle}}_{CPU})$	$a^{P_{busy}}_{CPU}$	$p(a^{P_{busy}}_{CPU})$	$a^{P_{sum}}_{CPU}$	$p(a^{P_{sum}}_{CPU})$	$a^P_{s_i}$	$p(a^P_{s_a})$	a^P_S	$\sum p_i(a^P_i)$	$\sum p_i(a^J_i)$
1	$a^{P,tar}_{CPU}(a^{T,gua}_{CPU})$	60	49.25	200	25	232.73	74.25	70	10	302.73	84.25	334.25
2	$a^{P,tar}_{CPU}(a^{T,tar}_{CPU})$	75	48.75	240	10	171	58.75	70	10	241	68.75	343.75
3	$a^{P,gua}_{CPU}(a^{T,tar}_{CPU})$	100	18.75	400	-175	315.74	-156.25	70	10	385.74	-146.25	-141.25
4	$a^{P,tar}_{CPU}(a^{T,top}_{CPU})$	100	49	400	12.5	207.87	61.5	70	10	277.87	71.5	326.5
5	$a^{P,top}_{CPU}(a^{T,tar}_{CPU})$	50	153.75	230	120	146	273.75	70	10	216	283.75	928.75
6	$a^{P,top}_{CPU}(a^{T,gua}_{CPU})$	40	153.25	170	155	181.33	308.25	70	10	378.25	318.33	938.25
7	$a^{P,gua}_{CPU}(a^{T,75MHz}_{CPU})$	40.5	21.25	153	-85	120.5	-63.5	70	10	190.5	-53.5	579

$a^P[mW]$, $p^P_i[mu]$

[21] 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.

[22] 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.

[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] C. Schneeweiss and K. Zimmer. Hierarchical coordination mechanisms within the supply chain. *European Journal of Operations Research*, 153(3):687–703, 2004.

[26] S. A. Shane and K. T. Ulrich. Technological Innovation, Product Development, and Entrepreneurship in Management Science. *Management science*, 50(2):133–144, 2004.

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

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

[29] F. van Echtelt, J. F. Wynstra, A. van Weele, and G. Duysters. Critical processes for managing supplier involvement in new product development. 2004.

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

[31] 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.