14

Resource-Oriented Load Distribution in a Framework Environment *

Jürgen Schubert **Arno Kunzmann**

Forschungszentrum Informatik, FZI
Systementwurf in der Mikroelektronik (Prof. W. Rosenstiel)
Haid-und-Neu-Straße 10-14, D-76131 Karlsruhe, Germany
E-mail: schubert/kunzmann@fzi.de

Abstract

In this paper the framework-based load distribution system FLODIS is presented. In order to reduce the overall design time, FLODIS provides a user-transparent distribution of design activities over a heterogeneous network of workstations. The distribution algorithm is mainly controlled by estimations about the required activity-specific resources that can be derived by user-defined data, recorded execution profiles and methodology information of the framework. Experimental data with several design flows and different design activities show that by the proposed distribution system the overall execution time can be distinctly reduced. Compared with conventional distribution systems an average reduction up to over 60% can be observed.

1 Introduction

The development of complex systems requires a highly efficient cooperation of design teams to reduce the overall design time. A key objective is the support of concurrent engineering. This necessarily implies that a simultaneous and coordinated execution of a large number of different design tools is required. Depending on the selected parameters and the input/output data, each tool can perform several activities. Given a network of several workstations, the execution time for each activity is heavily influenced by the selected workstation. In order to reduce both the individual and the overall response time, several load distribution systems have been developed. Since the management of complex design tasks is typically embedded in a specific design framework, the efficiency of load distribution systems can be significantly improved by using framework knowledge.

*This work is supported by the ESPRIT project JESSI-COMMON-Frame (7364).

F. J. Rammig et al. (eds.), *Electronic Design Automation Frameworks*

In this paper, the concept and implementation of the load distribution system FLODIS (Framework based LOad DIstribution System) will be discussed. Experimental results show that compared with conventional load distribution approaches, FLODIS distinctly reduces the overall response time in average up to over 60 %.

2 The FLODIS Approach: Motivation and Concepts

One general problem of most load distribution systems is the missing knowledge about the upcoming load. Hereto, static approaches use statistics, for instance, fixed job arrival rates. The distribution is performed without measuring the load of the available workstations, assuming an average external load [TT85]. In contrast, dynamic distribution algorithms consider the current system load, in order to select the workstations with least load.

Embedded in framework environments, several approaches were reported: The MCC CAD framework selects an appropriate workstation based on the (static) Round-Robin algorithm [ARF90]. FLOW is a representative approach for dynamic load distribution [Kas92]. PAPYRUS is based on the operating system SPRITE and, therefore, also process migration is supported [CK93]. In the ULYSSES II framework, resources are dynamically allocated [Bus94]. Here, the best suited workstation is selected by calculating the processing power and work load. A common feature of all these distribution strategies is that they are only based on the number of activities. But obviously, the necessary resources (e.g., CPU-time, I/O-time, main memory) are essential to compute efficient allocations. Therefore, FLODIS also takes resources into account, but since exact values are only available in very special cases, estimations about the required resources will be used. In a framework environment precise predictions about activity specific resources can be gathered by different sources:

a) The tool integrator often knows which complexity dimension (CD) value (e.g., number of gates, transistors, primary inputs) are important for computing the activity specific resources. Additionally, detailed resource requirements, e.g., how the number of gates influences the memory and CPU-time consumption may be known. For several activities like simulation and test pattern generation, an average complexity of the basic algorithm is available and the corresponding calculation formula can be used for load predictions.

b) With every activity execution, the used resources are recorded, always related to the CD values. In FLODIS a linear regression is used to forecast CPU-time, I/O-time and memory requirements [DS66].

c) Derived from the methodology management, the information of repeatedly executed activities within a subproject can be used. In this case the CD values are expected to be similar because the same specification is used for the part which has to be designed. The mean value of the previously used resources is sufficiently accurate.

d) For executions within the same design flows, where predefined activity sequences have to be followed, the CD value of the last execution is chosen, in general providing an even more accurate value. This is a typical case because a design process often is iterative which includes the re-execution of activities with slightly modified input until the specification is fulfilled.

To establish the required communication between framework and load distribution system, the framework sends activities to FLODIS together with the information about the identification of the design flow and the subproject from which the activity is started. In FLODIS the resource estimation algorithm in Figure 1 is used for characterization with respect to the relative CPU and I/O usage of the activities to distinguish between CPU and I/O intensive jobs.

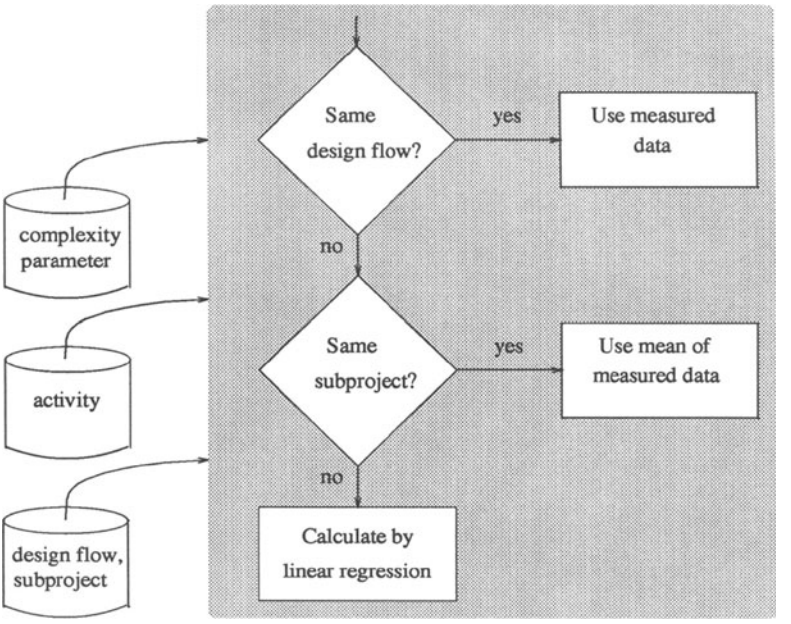

Figure 1: Activity resource estimation algorithm

Since an execution of jobs with the same characteristics on the same workstation results in relatively high increases of the response times, in FLODIS an allocation strategy is implemented that prefers the combination of tools with different characteristics. Given a workstation specific maximum CPU and I/O load, activities can be allocated as long as this limit is not exceeded. To take also workstations with different computing power into account, transfer factors for adapting CPU and I/O requirements are calculated. In order

to optimize the use of powerful workstations, the distribution algorithm is based on a list of available workstations, descending ordered according to their computing power. Always starting at the top of this list, activities are allocated if the available resources fulfil the activity requirements. In this case, the workstation resources are decreased according to the resource predictions for the allocated activity, otherwise the next workstation is scanned.

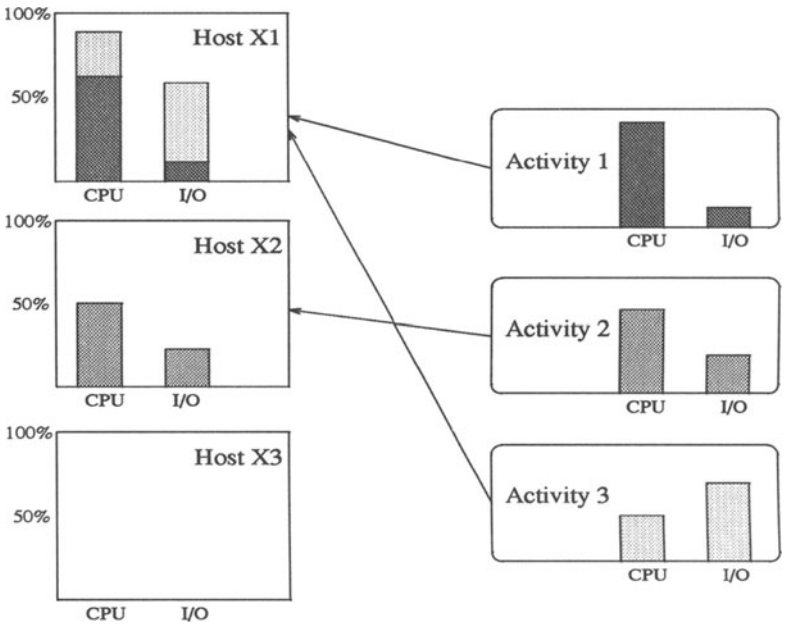

Figure 2: Activity allocation algorithm

Figure 2 presents an example for the allocation of three activities that have to be executed. The available three hosts are ordered according to their computing power. Both activities, activity 1 and activity 2, are CPU intensive jobs. The allocation algorithm selects the more powerful host (Host X1) for execution of activity 1 and the available resources of this workstation are decreased by the estimated activity requirements. The second activity needs more CPU resources than host X1 can provide, therefore, the less powerful host (Host X2) is selected. The third activity needs a lot of I/O resources. The allocation algorithm tries to allocate the activity at the most powerful host X1. As both, the CPU requirements and the I/O requirements, do not exceed the predefined limit, host X1 is selected. This guarantees on one hand that the activities don't have to wait for resources which are used by another activity, and on the other hand, the activities are executed on the most powerful workstation which is available. The minimum execution time of the

activities is provided in spite of the fact that the host X3 is not used.

The presented algorithm does not take the real load of the workstations into account. It is assumed that in a framework design environment all the activities on the workstations are controlled by the framework. Of course, this assumption does not hold, if the designer is allowed to start a process which is not known by the framework. Therefore, FLODIS offers the possibility to refuse an execution if an overload of a workstation has to be avoided. FLODIS checks the selected host before an activity is sent for execution. If the host does not provide the required resources because non framework processes are running, the activity is sent back to the allocation algorithm for a re-selection of another host. Additionally, the specific host is marked as 'overloaded'. The execution of activities is not delayed because only the selected host has to be checked and not the workstation environment. Of course, a delay occurs in case of an overloaded host, but this is assumed to be an exception.

3 Experimental Results

The reported results are computed by a prototype implementation of the FLODIS system based on the JESSI-COMMON-Framework (JCF) [LJ92]. At present FLODIS is implemented as a domain neutral tool that is loosely coupled with the JCF framework, i.e. all necessary data for the load distribution are exchanged via the JCF Tool Management System. The experimental analyses are based on three different types of design flows. As indicated in Figure 3, sub-flows can be executed in parallel. The degree of parallelism characterizes the three basic flow types, where (n,m) is set to (2,1), (3,3) and (6,6). Obviously, in the last case 24 activities have to be executed.

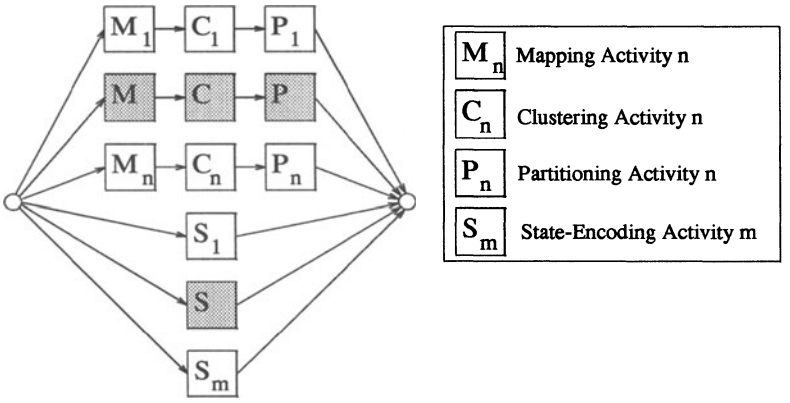

Figure 3: Principle structure of the evaluated design flows

All evaluated flows are typically used for the design of field-programmable gate-arrays (FPGAs), and consist of four basic activities for mapping, clustering, partitioning and state encoding. To receive practice-oriented measurements, all flows were executed with up to 12 different control parameters and input data, e.g. circuit sizes. For the subsequently given response times, a network of six SUN workstations with different computing power is assumed (1xSS20, 1xSS10, 2xSS2, 2xSS1). In Table 1 and Table 2, the results for two conventional approaches (static: Round Robin and dynamic: LB [KS94]) and FLODIS are listed, based on three different settings of n and m.

	n=2,m=1			n=3, m=3					
	Parameter Set			Parameter Set					
	1	2	3	1	2	3	4	5	6
Round Robin	243	88	126	333	171	462	87	212	589
Dynamic (LB)	243	106	103	261	135	114	127	200	302
FLODIS	98	49	72	211	74	97	65	88	164
Savings RR [%]	59,7	44,3	42,9	36,6	56,7	79,0	25,3	58,5	72,2
Savings LB [%]	59,7	53,8	30,1	19,2	45,2	14,9	48,8	56,0	45,7

Table 1: Response times for two different flow types with 3 and 6 parameter sets [sec.]

	n=6,m=6											
	Parameter Set											
	1	2	3	4	5	6	7	8	9	10	11	12
Round Robin	539	548	202	67	130	69	132	159	340	100	221	290
Dynamic (LB)	226	310	154	53	133	64	119	152	317	118	173	249
FLODIS	176	256	114	49	123	50	121	180	260	81	128	186
Savings RR [%]	67,3	53,3	43,6	26,9	5,4	27,5	8,3	-13,2	23,5	19,0	42,1	35,9
Savings LB [%]	22,1	17,4	26,0	7,5	7,5	21,9	-1,7	-18,4	18,0	31,4	26,0	25,3

Table 2: Response times for the third flow type with 12 different parameter sets [sec.]

The given response times show the impact of the selected parameters. Besides these times values, for each parameter a direct comparison between the Round-Robin, LB and FLODIS approach is given by the last two rows. With only a very few exceptions, the ranking between these three systems is obvious: the dynamic LB has slight advantages over the static Round Robin approach, whilst the proposed new method requires the lowest response time in average. This result is also stated by Table 3, where the individual response times are summed up. Compared with Round-Robin, the relative response time savings of FLODIS are between 38 % and 62 %, for LB the achieved reductions range between 16 % and 51 %.

	n=2 m=1	n=3 m=3	n=6 m=6
Round Robin	459	1857	2802
Dynamic (LB)	454	1142	2074
FLODIS	219	701	1728
Savings RR [%]	52,3	62,3	38,3
Savings LB [%]	51,8	38,6	16,7

Table 3: Comparison of the overall response times [sec.]

The most significant savings can be achieved if a small number of activities is executed in the workstation net (n=2/m=1). This is the typical case in design environments. However, FLODIS also provides better results if the number of activities is higher than the available number of workstations.

4 Conclusions and Future Work

Short time-to-market is one main objective in the development of new products. Especially the design of complex systems where extensive CPU usage can be foreseen, an optimized usage of the available workstations can provide distinct response time savings. Within a framework environment FLODIS takes advantage of the available information about activities, project hierarchy and predefined design flows. Controlled by the current system load and the design activity profiles that allow the prediction of the expected CPU and I/O behaviour, an optimized allocation of activities to workstations can be performed. In contrast to conventional static load distribution systems, FLODIS allocates systematically several activities to one workstation as long as a load limit is not exceeded. As underlined by the experimental results the proposed distribution strategy shows distinct savings of the necessary overall response time up to 62 %. This reduction does not only contribute to a reduced design time, but also the designer will benefit of the distribution system by reduced response times.

The estimation algorithm of FLODIS provides all information about the activities that is necessary to control the allocation algorithm. Until now, the allocation uses relative values (percentages) to indicate the resource requirements. The results show significant reduced response times of the activities. However, other allocation strategies using the available information may result in the same or more reduction of the overall execution time. An interesting approach is the evaluation of a more simplified estimation algorithm which classifies the activities. The allocation algorithm will avoid to select the same workstation for the same classes. On one hand, this algorithm needs less precise information about the activities, but on the other hand the savings of the distribution algorithm might be in the range of the originally proposed algorithm.

Acknowledgments

We would like to thank Prof. W. Rosenstiel and Prof. F. Wagner for the fruitful discussions and their encouragement of this research which enabled this publication.

References

[ARF90] W. Allen, D. Rosenthal, and K. Fiduk. Distributed Methodology Management for Design-in-the-Large. In *ICCAD*, pages 346–349, Santa Clara, 1990.

[Bus94] M. Bushnell et al. Distributed Computing, Automatic Design and Error Recovery in the ULYSSES II Framework. In *EDAC*, Paris, March 1994.

[CK93] T. Chiueh and R. Katz. A Distributed Transaction Facility for Design Task Management. In *EDAC*, Paris , France, 1993.

[DS66] N. Draper and H. Smith. *Applied Regression Analysis.* John Wiley & Sons, New York, 1966.

[ELZ86] D. Eager, E. Lazowska, and J. Zahorjan. Adaptive load sharing in homogeneous distributed systems. *IEEE Transactions on Software Engineering*, 12(5):662–675, May 1986.

[Kas92] Y. Kashai. Flow - A Concurrent Methodology Manager. In *EDAC*, Brussels , Belgium, March 1992.

[Kir94] S. Kirschke. Aufbau eines modularen, planenden Lastverteilungssystems für den Einsatz in Frameworks. Master's thesis, Universität Karlsruhe, January 1994.

[KS94] D. Kassabian and T. Soyata. *LB - The Load Balancer.* Dept. of Electrical Engineering, University of Rochester, April 1994.

[LJ92] D. Liebisch and A. Jain. JESSI-COMMON-Framework Design Management - The means to Configuration and Execution of the Design Process. In *EuroDAC*, Hamburg, 1992.

[NXG85] L. Ni, C. Xu, and T. Gendreau. A distributed drafting algorithm for load balancing. *IEEE transactions on software engineering*, 11(10):1153–1161, October 1985.

[TT85] A. Tantawi and D. Towsley. Optimal Static Load Balancing in Distributed Computer Systems. *ACM Journal*, 32(2):445–465, April 1985.

[Wei93] C. Weiler. Lastverteilung von CAD-Werkzeugen unter Einbeziehung von Framework-Informationen. Master's thesis, Forschungszentrum Informatik, November 1993.