

A formal Approach for the Optimization of Heterogeneous Multiprocessors for Complex Image Processing Schemes

M. Schwiegershausen

P. Pirsch

University of Hannover
Laboratorium für Informationstechnologie
Schneiderberg 32, 30167 Hannover, Germany

Abstract

This paper presents a formal approach using MILP and its implementation as CAD tool for the optimization of heterogeneous multiprocessor systems. These heterogeneous systems, consisting of application-specific as well as of programmable processors, are highly suitable for performing complex schemes of image processing algorithms under real time constraints. By means of a formal optimization approach, it becomes possible to derive efficient, real time hardware, being optimal in terms of area expense and throughput rate.

1 Introduction

Image processing and coding schemes like high definition TV (HDTV), 3D-image analysis, image compression and coding (JPEG, MPEG) or scene interpretation often have to be processed under real-time constraints. This requires extremely high computational and throughput rates, which can only be achieved by massive application of parallel processing and pipelining as provided by multiprocessor systems. These image processing schemes can be splitted into several subtasks with different requirements leading to different architectures appropriate for the individual tasks.

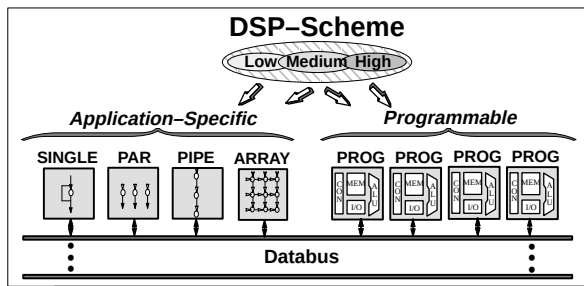


Fig. 1: Heterogeneous Multiprocessor

This results in a heterogeneous multiprocessor system for the implementation of the composite scheme, see Fig. 1. These heterogeneous systems may be composed of different, application-specific processing units like array processors for implementing low and medium level tasks with high computational demands and programmable processing elements on the other hand to provide the flexibility required by medium and high level tasks. Nevertheless, the large multitude of suitable processors for each single algorithm makes it

impossible for the human designer to choose manually the best combination of application-specific and programmable processors for a given image processing scheme. Consequently, a formal approach for the optimization of these heterogeneous systems is mandatory.

This paper is organized as follows: Section 2 reviews related research and previous work. Section 3 introduces our formal approach for the optimization as part of a whole design trajectory for heterogeneous systems. Section 4 illustrates the optimization approach using a video coding scheme as application example. Section 5 gives some hints on the implementation as CAD tool. Finally, concluding remarks are provided in Section 6.

2 Related Research

There has been extensive research on mapping algorithms onto multiprocessor systems. Mainly, in the field of high-level synthesis, methodologies for mapping one single algorithm onto a dedicated datapath [1] or onto an array of processing elements like systolic array processors [2] were tackled. On the other hand, synthesis scripts like *CATHEDRAL II, 2nd* [3]–[4] were developed in order to derive VLIW architectures consisting of synchronous DSP units with dedicated datapaths connected via a bus, addressing medium throughput applications. However, image processing applications demand for architectures providing extremely high computational and throughput rates. In [5] an approach for domain-specific multiprocessor systems is presented but in opposite to our work it aims at calculating feasible schedules for a given application-specific architecture without considering the area expense as well as the throughput rate.

Therefore, these approaches have to be extended concerning the derivation of heterogeneous systems, being optimal in terms of area expense and throughput rate for the execution of composite schemes of image processing algorithms. So, we present a new methodology for the optimization of these heterogeneous systems. It is part of a whole design trajectory for systems with application-specific and programmable processors.

3 Design Trajectory

The derivation of an optimal, heterogeneous multiprocessor system for a composite scheme of image processing algorithms can be divided into three steps, the

decomposition of the whole scheme into single tasks or algorithms, the *assignment* of all possible processors for each single task, and finally the *optimization* of the overall system, see Fig. 2.

Fig. 2: Design Trajectory

3.1 Decomposition

3.2 Assignment

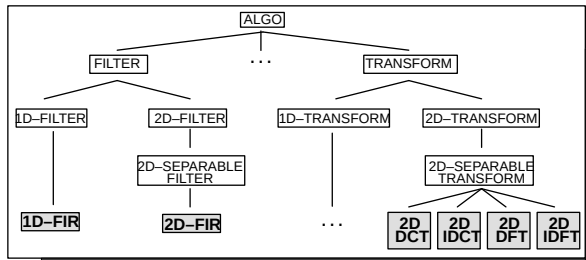


Fig. 3: Hierarchical Tree of Algorithms

1-dimensional and 2-dimensional transform / filter algorithms is necessary. Fig. 3 shows the representation of the image processing algorithms as hierarchical tree, whereas each leaf of the tree corresponds to a single image processing algorithm. Each class of algorithms $\mathcal{ALGO}(m_i)$ has an associated set of parameters θ_i like block-size (M_B, N_B) for transform algorithms, an additional kernel-size (M_K, N_K) for filter algorithms, or search/reference block-size $((M_S, N_S), (M_R, N_R))$ for block-matching algorithms, etc. Based on these algorithm-specific parameters θ_i , our aim is to explore the permissible design space, by deriving a set of parametrizable datapaths $\mathcal{DP}_j(\theta_i)$ for each single image processing algorithm.

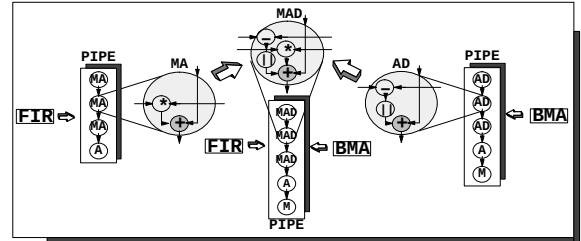


Fig. 4: Initial Assignment and Resource Sharing

$$f_{assign} : \mathcal{ALGO}(m_i) \xrightarrow{\text{type}} \mathcal{DP}_j(\theta_i, \text{type})$$

$$\mathcal{DP}_j \stackrel{\text{def}}{=} \{\mathbf{type}, \tau_{i,j}, A_j, \{BB_1, \dots, BB_L\}, \dots\}$$

and on the processor type. The attributes as well as their meanings are sketched below:

- type : processor type, i.e. one of **SINGLE**, **PAR**, **PIPE**, **ARRAY**.
- $\tau_{i,j}$: execution time for $\text{ALGO}(m_i)$ on the processor p_j with datapath $\mathcal{DP}_j$
- A_j : area expense of datapath $\mathcal{DP}_j$
- BB_l : required building blocks for datapath $\mathcal{DP}_j$, e.g. **MUL**, **ADD**, etc.

Tab. 1 shows these attributes for a filtering algorithm (**FIR**), with $\theta_{FIR} = \{(8, 8), (3, 3)\}$.

TYPE	τ [cycles]	A [trans.]	BB					
			ADD	SUB	MUL	ABS	MIN	REG
SINGLE	577	2432	1		1			1
PAR	194	7856	4		3			4
PIPE	196	7856	4		3			4
ARRAY	72	24756	12		9			21

Table 1: Attributes for FIR datapaths

Tab. 2 the corresponding attributes for a motion estimation algorithm (**BMA**), with $\theta_{BMA} = \{(8, 8), (3, 3)\}$, for each processor type.

TYPE	τ [cycles]	A [trans.]	BB					
			ADD	SUB	MUL	ABS	MIN	REG
SINGLE	326	2102	1	1		1	1	3
PAR	110	5222	4	3		3	1	6
PIPE	112	5222	4	3		3	1	6
ARRAY	46	15030	12	9		9	1	23

Table 2: Attributes for BMA datapaths

The corresponding datapaths $\mathcal{DP}_{FIR}$, $\mathcal{DP}_{BMA}$, assuming a processor of type **PIPE**, are sketched in Fig. 4. As a matter of fact, these processors are dedicated to exactly *one* algorithm. But, due to the fact, that many image processing algorithms often consist of similar operations, there is a large intersection of building blocks of the processor's datapaths. Thus, it is reasonable to merge the processor's datapaths to a combined datapath suitable for several algorithms, in order to derive additional processors with higher utilization. For example, the two datapaths $\mathcal{DP}_{FIR}$, $\mathcal{DP}_{BMA}$ of Fig. 4 can be favourably merged to a third datapath $\mathcal{DP}_{FIR/BMA}$ with several *shared* adders. Consequently, the whole assignment is performed in two consecutive steps by

1. Determining for each single algorithm or task all possible processors according to the processor library, leading to an initial assignment and
2. Applying a resource sharing algorithm by merging the processor's datapaths on a PE- or BB- level.

However, if it is really advantageous to select the combined processor of type **PIPE** or the two single processors, also of type **PIPE**, with respect to the efficiency of the overall system can not be answered on the fly. On the contrary, this leads to a combinatorial optimization problem, as described next in the third step of the design trajectory:

3.3 Optimization

Finally, it is necessary to select from the large amount of feasible assignments derived so far, the one which leads to an overall multiprocessor system with high computational and throughput rate, meeting the real-time constraint, as well as an area expense as low as possible. So, we are concerned with the problem to determine for each task the processor type, the temporal order of the task execution and of the data transfers, taking into consideration the precedence between the tasks, the availability of input / output data and the non overlapping usage of processors and buses in order to derive a valid multiprocessor scheduling and binding. This problem can be formulated as an optimization problem and efficiently solved by means of mixed integer linear programming, as follows:

3.3.1 MILP-Formulation

A general MILP consists of a cost function z , a set of variables $x_j \in \vec{x}$, and a system of linear constraints $\sum a_{ij} \cdot x_j = b_i$, called restrictions. It can be written in standard form as:

$$\begin{aligned} z &= \vec{c} \cdot \vec{x} \rightarrow \min \\ \text{subject to} \\ A \cdot \vec{x} &= \vec{b} \end{aligned}$$

Although the idea to tackle scheduling, allocation, and binding by (mixed integer) linear programming, MILP, is not essentially new, the presented approach is extended in comparison to previous works [6] by two main aspects. First, a multitude of different application-specific processors can be included by introduction of architecture dependent parameters which result from the assignment step. Second, periodic applications are treated taking into consideration requirements of real-time image processing. Here, the MILP models the problem of selecting the best combination of application-specific and programmable processors for a given image processing scheme. Thus, in the proceeding, the used variables, the restrictions as well as the cost function are presented, which are necessary to describe this special scheduling-, binding-, and allocation-problem by means of a MILP. Concerning the variables $\vec{x}$, it is reasonable to distinguish between

- binary-valued decision variables
 $\vec{x}_B = [\dots, x_{i,j}, \dots]$
- integer-valued variables
 $\vec{x}_I = [\dots, Y, B, \dots]$
- real-valued timing/area variables
 $\vec{x}_R = [\dots, s_i, e_i, \dots]$

For example, if task m_i is executed by processor p_j then the corresponding binary decision variable $x_{i,j} = 1$ and zero otherwise. The number of required buses or different processors is given by the integer-valued variables B, Y . Equivalently, the time at which the execution of task m_i starts/ends, is denoted by the real-valued variable s_i, e_i respectively. This leads to the definition of $\vec{x} \stackrel{\text{def}}{=} [\vec{x}_B | \vec{x}_I | \vec{x}_R]^T$ as the concatenation of the binary-, integer-, and real-valued variables. Concerning the restrictions $A \cdot \vec{x} = \vec{b}$, a distinction between three main groups is reasonably, namely:

- task-processor restrictions
 - *exclusive task-processor assignment*
 $\sum_j x_{i,j} = 1$
 - *task execution start*
 $s_k \geq ce_{i,k} - \sum_j \beta_{k,j}^I \cdot x_{i,j}$
 - *task execution end*
 $e_i = s_i + \sum_j \tau_{i,j} \cdot x_{i,j}$
 - *etc.*
- transfer-bus restrictions
 - *exclusive transfer-bus assignment*
 $\sum_l \eta_{i,k}^l = \gamma_{i,k}^R$
 - *transfer execution start*
 $cs_{i,k} \geq s_i + \sum_j \beta_{i,j}^O \cdot x_{i,j}$
 - *transfer execution end*
 $ce_{i,k} = cs_{i,k} + \gamma_{i,k}^R \cdot \frac{D_{i,k}}{B_R} + \gamma_{i,k}^L \cdot \frac{D_{i,k}}{B_L}$
 - *etc.*
- general restrictions
 - *number of processors*
 $Y = \sum_j \xi_j$
 - *number of buses*
 $B = \sum_l \eta_l$
 - *area expense*
 $A = \sum_j A_j \cdot \xi_j$
 - *etc.*

For example, the *exclusive task-processor assignment* guarantees, that exactly one processor p_j is selected for each task m_i . Furthermore, the *transfer execution end* time can be expressed as a linear constraint by means of the number of data items $D_{i,k}$ to be transmitted from task m_i to task m_k , the transfer rate B_R of the bus, and whether it is a remote ($\gamma_{i,k}^R = 1$) or a local ($\gamma_{i,k}^L = 1$) transfer. The *area expense* A , taken from the group of general restrictions can be expressed by the sum of areas A_j for all selected processors ($\xi_j = 1$), etc.

Finally, after the derivation of the variables $\vec{x}$ and the linear constraints $\sum_j a_{i,j} \cdot x_j = b_i$, we are concerned with the problem to choose an appropriate cost function z for the MILP. Since we aim at an optimal, fully static and overlapped multiprocessor scheduling with low area expense and high throughput rate, the computation time (latency) T , the computation period P in case of a continuous, periodic execution of the whole scheme of algorithms as well as number Y and area A of processors, as well as number B of buses is taken into consideration. This leads to:

$$z = c_T \cdot T + c_P \cdot P + c_A \cdot A + c_Y \cdot Y + c_B \cdot B$$

The individual weights $c_T, c_P, \dots, c_B$ provide the opportunity to consider different goals of the designer. This leads to a priority-based optimization technique. For example, if the main goal is to derive an overall multiprocessor system with minimum area expense A and additionally an execution time T as small as possible the priority order is $A \succ T \succ \dots \succ B$. The corresponding, individual weights according to the designer's intention have to be $c_A \gg c_T \gg \dots \gg c_B$.

For two adjacent weights c_i, c_{i+1} , the ratio $r = \frac{c_i}{c_{i+1}}$ has to be large enough ($r \gg 1$), in order to achieve the optimum according to the priority order. Furthermore, it is necessary to transform all the parameters $V \in \{A, T, P, Y, B\}$ of the objective function according to

$$V' = \frac{V - V_{min}}{V_{max} - V_{min}} \quad 0 \leq V' \leq 1$$

to an identical and normalized range, i.e. $[0..1]$. These normalized parameters V' can be derived by calculating the minimum as well as the maximum possible values.

4 Experimental Results

In order to validate the formal optimization approach, using mixed integer linear programming, a typical complex image processing application was chosen: Low bit rate coding according to the hybrid videocodec scheme based on CCITT recommendation H.261 [7]. It is used for data reduction necessary to transmit video data on a line with $p \times 64$ kbit/s ($1 \leq p \leq 30$), e.g. video telephone, video conferencing or even multimedia.

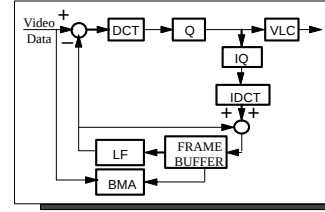


Fig. 5: Video encoder algorithm according to H.261

The application consists of several computational extensive regular, low level tasks like motion estimation by block matching algorithm (BMA), discrete cosine transform and its inverse (DCT, IDCT), and FIR filtering (LF) as well as data dependent, irregular medium level tasks like quantization and its inverse (Q, IQ) and variable / runlength coding (VLC, RLC) combined to coding (COD), see Fig. 5.

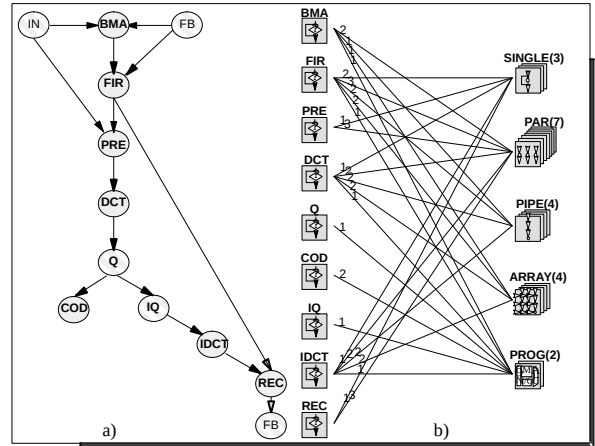


Fig. 6: Task Graph and Possible Assignments

Fig. 6a shows the corresponding task graph derived from the video coding algorithm H.261. In addition to

the tasks mentioned above, there are two tasks not yet described, namely prediction (PRE) and reconstruction (REC). They denote the subtraction/addition of the reference-block of the current video frame and of the best matching block with respect to a search area of the previous video frame. Fig. 6b shows the possible assignments of the $M = 12$ tasks to $N = 20$ processors, assuming four different processor types (SEQ, PAR, PIPE, ARRAY) for the low-level tasks and two programmable processors (PROG) for the low- and medium-level tasks. By application of resource sharing, 12 out of 18 application specific processors consist of a multifunctional and merged datapath. Each of these merged datapaths can be used to calculate up to three different low-level tasks. The numbers in parenthesis at the processor types in Fig. 6b indicate, how many different processors of each type could be derived, whereas a number at an assignment edge indicates how many different processors of same type each task can be mapped to. In order to take the data transfer of video data between the tasks and the video memory also into consideration, two other tasks are supplied, namely IN, denoting the buffer of the current video frame and BUF the buffer of the previous video frame. The corresponding MILP model under these assumptions leads to 117 variables and 208 restrictions which could be solved by *branch-and-bound* within a few cpu-minutes. For this purpose, the LP-Solver of the NAG¹ Fortran-Library was adapted to these needs. According to the designers goal or priority order different solutions for an efficient heterogeneous multiprocessor system can be derived. The performance and area expense of four solutions derived by solving the MILP under different assumptions concerning the designer's priority are shown in Table 3.

Priority		$A \succ T, P \succ Y$	$P \succ A \succ T \succ Y$	$P \succ T \succ A \succ Y$	$Y \succ A \succ T, P$
Area Expense	A [#trans.]	172902	726930	605820	184190
Computation Time	T [#cycles]	7100	4514	4994	6134
Computation Period	P [#cycles]	6324	1073	1073	5581
Processors	Y [#proc.]	3	7	8	3

Table 3: Performance and area expense

The value of the most important parameter concerning the priority order is sketched grey in each column. As can be seen, the minimum value with respect to the designer's main goal could always be achieved. If for example the designer's main goal is to derive an overall system with minimum computation period P , and additionally small area expense A and small latency T , i.e. $P \succ A \succ T \succ Y$, the result is shown in the second column of Table 3. For this goal the optimal assignment derived by solving the MILP is shown in Fig. 7a. The corresponding architecture consists of $Y = 8$ processors, one of type **SINGLE** and **PIPE**, and each two of type **PROG**, **PAR**, and **ARRAY**, leading to an overall system with an area expense of $A = 605820$ transistors. The values V of all system parameters like computation period P , computation time T , area expense A , as well as the number Y of different processors are sketched in normalized form in Fig. 7b. In this case the priority order

was $P \succ A \succ T \succ Y$. Therefore, the normalized computation period $P' = \frac{P - P_{min}}{P_{max} - P_{min}}$ is near to zero ($P \rightarrow P_{min}$) according to the highest priority of the cost function.

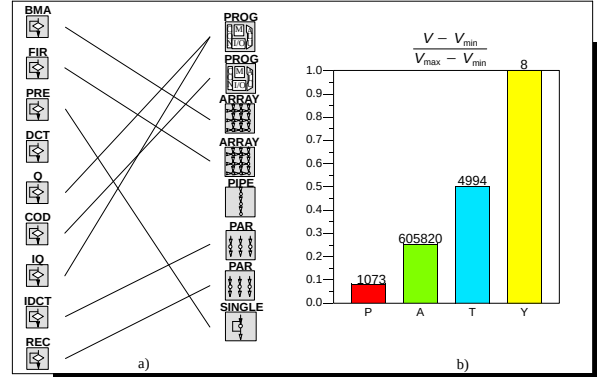


Fig. 7: Optimal Assignment, Normalized System Parameters

The overlapped and periodic schedule for the multiprocessor system is shown in Fig. 8. The computations of the different macro blocks (16*16 pixels) are sketched in different colors. It can be seen, that the computations of three macro blocks overlap. In case a processor is idle, white areas are shown.

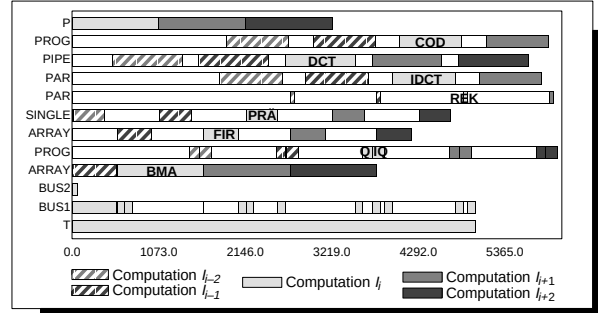


Fig. 8: Periodic and overlapped schedule

As can be seen from Fig. 8 the achievable throughput rate or computation period P is fixed by the number of clock cycles necessary to compute the task **BMA** on the processor of type **ARRAY**. Since there are no idle times between the execution of two consecutive macro blocks on this processor, the achievable computation period is given by the execution time $\tau_{BMA} = 1073$ cycles of the blockmatching algorithm. In contrast to this, the latency or computation time T necessary to perform the hybrid coding once on a whole macro block including all tasks takes $T = 4994$ cycles!

The four solutions with different priorities like minimum area expense or minimum computation period according to the four columns of Table 3 are sketched in the area- and time-plane, see Fig. 9. The possible design spaces according to the supported processor library and the real-time constraints of the (video) application are marked by the dashed rectangles, showing the possible solutions concerning area expense A and number of processors Y , Fig. 9a, as well as con-

¹The Numerical Algorithms Group Limited

cerning computation period P and latency T , Fig. 9b.

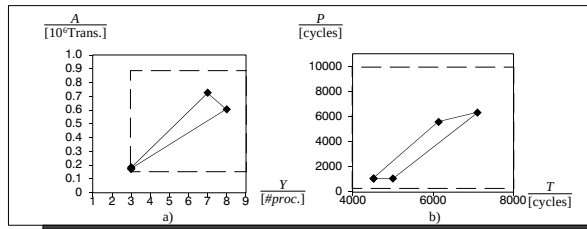


Fig. 9: Design spaces concerning A , Y and P , T

For example, the upper edge of the P , T plane reflects the real-time constraint ($P \leq 10101$). Clearly, any feasible solution must reside inside the dashed design regions. Thus, depending on the designer's main goal either area or time optimal solutions can be derived.

5 Implementation

The presented design trajectory including the optimization approach using MILP is part of a system-level synthesis aid for the derivation of heterogeneous multiprocessors. It is currently under development as a prototype CAD implementation in COMMON LISP/CLOS² on Sparc Stations.

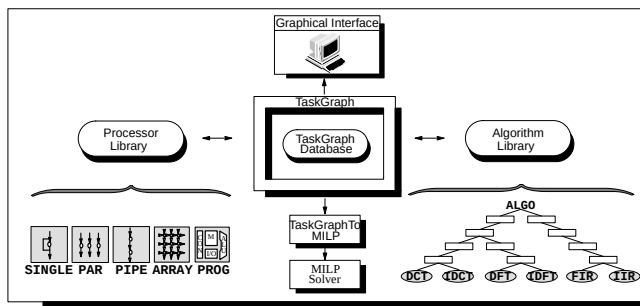


Fig. 10: Overview of the CAD tool

An overview of the CAD system is given in Fig. 10 with a specialized task graph as central database, whereas the database is connected to the algorithm library as well as to the processor library. Based on the processor library, all feasible assignments are derived, and transferred to the taskgraph. Finally, the transformation to a MILP formulation is performed, and the (MI)LP solver is invoked. The results of the optimization can be visualized using a graphical user interface (GUI), based on CLUE/CLX³, see Fig. 11. The GUI provides diagrams showing the expense with respect to the normalized system parameters, gantt charts showing the multiprocessor scheduling, and graphs showing the design space as well as the optimal assignment.

6 Conclusion

In this paper, a design trajectory including a formal approach for the optimization of heterogeneous multiprocessor systems is presented. By the use of mixed

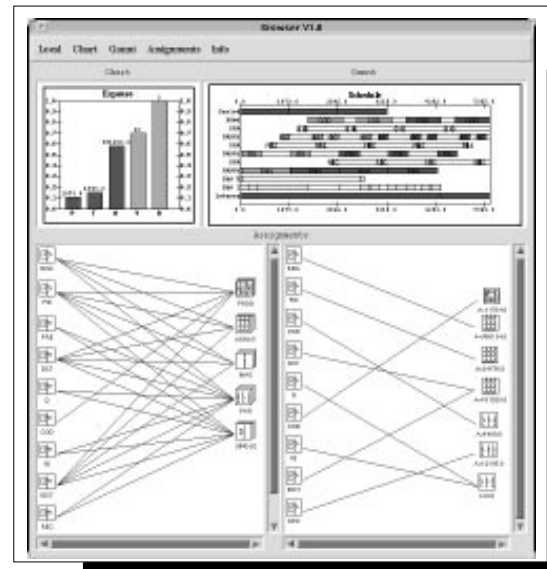


Fig. 11: Graphical User Interface

integer linear programming as mathematical framework of the optimization, it becomes possible to derive the best combination of application-specific and programmable processors for heterogeneous systems, being optimal in terms of a designer's priority or main goal like minimum area expense or maximum throughput.

Acknowledgements

This work is supported by the Deutsche Forschungsgemeinschaft, contract number Pi-169/5.

References

- [1] W. Rosenstiel, H. Krämer, "Scheduling and Assignment in High Level Synthesis", in *High Level VLSI Synthesis*, R. Composano, W. Wolf, editors, Kluwer Academic Publishers, pp. 355-382, 1991.
- [2] S.Y. Kung, S.N. Jean, "A VLSI Array Compiler System (VACS) for Array Design", R.W. Brodersen, H.S. Moscovitz, editors, *VLSI Signal Processing III*, Chapter 45, pp. 495 - 508, IEEE Press, New York, 1988.
- [3] J. Rabbaey, H. de Man, J. Vanhoof, G. Goosens, F. Cathoor, "CATHEDRAL II: A Synthesis System for Multiprocessor DSP Systems", in *Silicon Compilation*, Addison-Wesley, pp. 311-360, 1988.
- [4] J. Vanhoof, I. Bolsens, G. Goosens, H. de Man, K. Rompaey, "High level synthesis for real-time digital signal processing", Kluwer Academic Press, 1993.
- [5] A. Bachmann, M. Schöbinger, L. Thiele, "Synthesis for Domain-Specific Multiprocessor Systems including Memory Design", L.D.J. Eggermont, P. Dewilde, E. Depette, J. van Meerbergen, editors, *VLSI Signal Processing VI*, Part 3, pp. 417-425, IEEE Press, New York, 1993.
- [6] S. Prakash, A.C. Parker, "Synthesis of Application Specific Heterogeneous Multiprocessor Systems", *Journal of Parallel and Distributed Computing*, no. 16, pp. 338 - 351, December 1992.
- [7] CCITT Study Group XV: Recommendation H.261, "Video Codec for Audiovisual Services at p x 64 kbit/s, Report R37, Geneva, July 1990.

²CLOS Specification, ANSI, X3J13

³Common Lisp User Interface Environment/X