Localization of Data Transfer in Processor Arrays

Dirk Fimmel and Renate Merker

Department of Electrical Engineering
Dresden University of Technology

Abstract. In this paper we present an approach to localize the data transfer in processor arrays. Our aim is to select channels between processors of the processor array performing the data transfers. Channels can be varying with respect to the bandwidth and to the communication delay and can be bidirectional. Our objective is to minimize the implementation cost of the channels while satisfying the data dependencies.
The presented approach also applies to the problem of localizing data dependencies for a given interconnection topology. The formulation of our method as an integer linear program allows its use for automatic parallelization.

1 Introduction

Processor arrays (PA) are well suited to implement time-consuming algorithms of signal processing with real-time requirements. Technological progress allows the implementation of even complex processor arrays in silicon as well as in FPGAs. To explore the degrees of freedom in the design of processor arrays automatic tools are required.
Processor arrays are characterized by a significant number of processors which communicate via interconnections in a small neighborhood. Data transfer caused by the original algorithm has to be organized using local interconnections between processors. This paper covers the design of a cost-minimal interconnection network and the organization of the data transfers using this interconnections. A solution of the problem of organizing the data transfers for a given interconnection network is also presented.
The design of processor arrays is well studied (e.g. [2, 7, 8, 10, 13]) and became more realistic by inclusion of resource constraints [3, 5, 12]. But up to now, only some work has been done in the organization of data transfer. Fortes and Moldovan [6] as well as Lee and Kedem [9] discuss the need of a decomposition of global interconnections into a set of local interconnections without consideration of access conflicts to channels. Chou and Kung [1] present an approach to organize the communications in a partitioned processor array, but do not give a solution for the decomposition problem.
In this paper, we present an approach to localize the data transfer in processor arrays. Channels with different bandwidth and latency can be selected to implement the interconnections between processors. The data transfers which are

P. Amestoy et al. (Eds.): Euro-Par'99, LNCS 1685, pp. 401–408, 1999.

given as displacement vectors are decomposed into a set of channels. The decomposition of displacement vectors as well as the order of using the channels is determined by an optimization problem. The objective of the optimization problem is to minimize the cost associated with an implementation of the channels in silicon.

The paper is organized as follows. Basics of the design of processor arrays are given in section 2. In section 3 a model of channels between processors is introduced. The communication problem which describes the organization of the data transfers is discussed in section 4. In section 5 a linear programming approach for the solution of the communication problem is presented. An example is given in section 6 to explain our approach followed by concluding remarks in section 7.

2 Design of Parallel Processor Arrays

In our underlying design system of processor arrays we consider the class of *regular iterative algorithms* (RIA) [10]. A RIA is a set of equations S_i of the following form:

$$S_i: \quad y_i[\mathbf{i}] = F_i(\cdots, y_j[f_{ij}^k(\mathbf{i})], \cdots), \quad \mathbf{i} \in \mathcal{I}, \quad 1 \leq i,j \leq m, 1 \leq k \leq m_{ij}, \quad (1)$$

where $\mathbf{i} \in \mathbb{Z}^n$ is an index vector, $f_{ij}^k(\mathbf{i}) = \mathbf{i} - \mathbf{d}_{ij}^k$ are index functions, the constant vectors $\mathbf{d}_{ij}^k \in \mathbb{Z}^n$ are called dependence vectors, y_i are indexed variables and F_i are arbitrary operations. The equations are defined in an index space $\mathcal{I}$ being a polytope $\mathcal{I} = \{\mathbf{i} \mid \mathbf{H}\mathbf{i} \geq \mathbf{h}_0\}$, $\mathbf{H} \in \mathbb{Q}^{n_h \times n}, \mathbf{h}_0 \in \mathbb{Q}^{n_h}$. We suppose that RIAs have a *single assignment form* (each instance of a variable y_i is defined only once in the algorithm) and that there exists a partial order of the instances of the equations that satisfies the data dependencies.

Next we introduce a graph representation to describe the data dependencies of the RIA. The equations of the RIA build the m nodes $S_i \in \mathcal{S}$ of the reduced dependence graph $\langle \mathcal{S}, \mathcal{E} \rangle$. The directed edges $(S_i, S_j) \in \mathcal{E}$ are the data dependencies weighted by the dependence vectors $\mathbf{d}_{ij}^k$. The weight of an edge $e \in \mathcal{E}$ is called $\mathbf{d}(e)$, the source of this edge $\sigma(e)$ and the sink $\delta(e)$.

The main task of the design of processor arrays is to determine the time and the processor when and where each instance of the equations of the RIA has to be evaluated. In order to keep the regularity of the algorithm in the resulting processor array only uniform affine mappings [10] are applied to the RIA.

A uniform affine *allocation function* assigns an evaluation processor to each instance of the equations and has the following form:

$$\pi_i : \mathbb{Z}^n \rightarrow \mathbb{Z}^{n-1}: \quad \pi_i(\mathbf{i}) = \mathbf{S}\mathbf{i} + \mathbf{p}_i, \quad 1 \leq i \leq m, \quad (2)$$

where $\mathbf{p}_i \in \mathbb{Z}^{n-1}$ and $\mathbf{S} \in \mathbb{Z}^{n-1 \times n}$ is of full row rank and assumed to be e-unimodular [11] leading to a dense processor array (see [4] for further details). Since $\mathbf{S}$ is of full row rank, the vector $\mathbf{u} \in \mathbb{Z}^n$ which is coprime and satisfies $\mathbf{S}\mathbf{u} = \mathbf{0}$ and $\mathbf{u} \neq \mathbf{0}$ is uniquely defined (except to the sign) and called *projection vector*. The importance of the projection vector is due to the fact that those and only those index points of an index space lying on a line spanned by the projection vector $\mathbf{u}$ are mapped onto the same processor.

Using the allocation function $\pi_i(\mathbf{i})$ dependence vectors $\mathbf{d}(e)$ are mapped onto *displacement vectors* $\mathbf{v}(e)$ by: $\mathbf{v}(e) = \mathbf{S}\mathbf{d}(e) + \mathbf{p}_{\delta(e)} - \mathbf{p}_{\sigma(e)}$.
A uniform affine *scheduling function* assigns an evaluation time to each instance of the equations and has the following form:

$$\tau_i : \mathbb{Z}^n \to \mathbb{Z} : \quad \tau_i(\mathbf{i}) = \boldsymbol{\tau}^T \mathbf{i} + t_i, \quad 1 \leq i \leq m, \tag{3}$$

where $\boldsymbol{\tau} \in \mathbb{Z}^n, t_i \in \mathbb{Z}$.
The following *causality constraint* ensures the satisfaction of data dependencies:

$$\boldsymbol{\tau}^T \mathbf{d}(e) + t_{\delta(e)} - t_{\sigma(e)} \geq d_{\sigma(e)}, \quad \forall e \in \mathcal{E}, \tag{4}$$

where d_i is the time needed to compute operation F_i.
Due to the regularity of the index space and the uniform affine scheduling function, the processor executes the operations associated with that index points consecutively if $\boldsymbol{\tau}^T \mathbf{u} \neq 0$ with a constant time distance $\lambda = |\boldsymbol{\tau}^T \mathbf{u}|$ which is called *iteration interval*.

3 Channels and Decomposition of Displacement Vectors

A set $\mathcal{W} = \{\mathbf{w}_1, \cdots, \mathbf{w}_{|\mathcal{W}|}\}$ of channels $\mathbf{w}_i \in \mathbb{Z}^{n-1}$ between processors is supposed to be given. The subset $\mathcal{W}^* \subseteq \mathcal{W}$ of channels can be used bidirectional, the subset $\mathcal{W} \backslash \mathcal{W}^*$ of onedirectional channels is denoted $\mathcal{W}^+$. Each channel $\mathbf{w}_i \in \mathcal{W}$ is weighted by implementation cost c_i and by a communication time l_i. The communication time l_i can depend on data dependencies, i.e. $l_i = l_i(e)$, which allows to consider both channels and data with different bandwidth. The value $l_i(e)$ is assumed to already contain a multiple use of a channel $\mathbf{w}_i$ if the bandwidth of value $y_{\sigma(e)}$ exceeds the bandwidth of that channel. In our model, the communication on a channel is restricted to one value per time even if the bandwidth of a channel would allow some values of smaller bandwidth per time.
In order to organize the data transfer the displacement vectors $\mathbf{v}(e)$ have to be decomposed into a linear combination of channels $\mathbf{w}_i \in \mathcal{W}$:

$$\mathbf{v}(e) = \sum_{i=1}^{|\mathcal{W}|} b_i(e) \mathbf{w}_i, \quad \forall e \in \mathcal{E}, \tag{5}$$

where $b_i(e) \in \mathbb{Z}$ and $\forall \mathbf{w}_i \in \mathcal{W}^+ . b_i(e) \geq 0$.
Since the summation in (5) is commutative we allow the use of the channels in an arbitrary order.

4 Communication Problem

The communication problem consists in minimizing the implementation cost for the channels subject to a conflict free organization of the data transfer. Due to the regularity of the processor array it is sufficient to consider one iteration interval of one processor to describe the communication problem.

Following the notation of the previous sections the implementation cost for the channels are measured by $C = \sum_{i=1}^{|\mathcal{W}|} n_i c_i$, where n_i is the number of instances of channel $\mathbf{w}_i \in \mathcal{W}$ leading off from one processor, and c_i is the implementation cost of channel $\mathbf{w}_i$.
The data transfer must be causal which includes the following points for a data dependence $e \in \mathcal{E}$:

- Data transfer can start at time $t^{start}(e) = t_{\sigma(e)} + d_{\sigma(e)}$,
- Data transfer must be finished at time $t^{end}(e) = \boldsymbol{\tau}^T \mathbf{d}(e) + t_{\delta(e)}$,
- When a data transfer e uses several channels successively, the use of that channels must not be overlapping. Suppose that $t_i^j(e)$ and $t_k^l(e)$ are the starting time of the j-th and l-th communication on the channels $\mathbf{w}_i$ and $\mathbf{w}_k$ respectively. Then either $t_i^j(e) \geq t_k^l(e) + l_k(e)$ or $t_k^l(e) \geq t_i^j(e) + l_i(e)$ has to be satisfied depending on which communication is starting first.

The causality of an example data transfer $e \in \mathcal{E}$ is depicted in figure 1. Either $t^{start}(e) \leq t_1^1(e) \leq t_1^1(e) + l_1(e) \leq t_2^1(e) \leq t_2^1(e) + l_2(e) \leq t^{end}(e)$ or $t^{start}(e) \leq t_2^1(e) \leq t_2^1(e) + l_2(e) \leq t_1^1(e) \leq t_1^1(e) + l_1(e) \leq t^{end}(e)$ has to be satisfied.

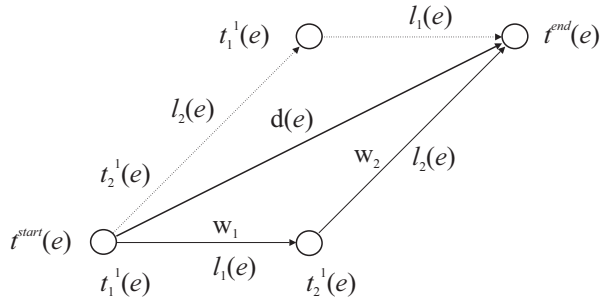

Fig. 1. *Causality of data transfer*

As already mentioned above it is sufficient to consider one iteration interval to ensure a conflict free use of the channels, since the data transfer on each channel is repeated periodically with the period λ. W.l.o.g. we consider the interval $[0, \lambda)$. To this end we decompose the starting time of the j-th communication on channel $\mathbf{w}_i$ used for the data transfer $e \in \mathcal{E}$ into $t_i^j(e) = p_i^j(e)\lambda + o_i^j(e)$, where $0 \leq o_i^j(e) < \lambda$ and $p_i^j(e), o_i^j(e) \in \mathbb{Z}$. A conflict free use of channel $\mathbf{w}_i$ by data transfers $e_1, e_2 \in \mathcal{E}$ is satisfied if:

$$\left.\begin{aligned} o_i^j(e_1) - o_i^k(e_2) &\geq l_i(e_2), \\ \lambda - o_i^j(e_1) + o_i^k(e_2) &\geq l_i(e_1), \end{aligned}\right\} \text{ if } o_i^j(e_1) > o_i^k(e_2),$$
$$\left.\begin{aligned} o_i^k(e_2) - o_i^j(e_1) &\geq l_i(e_1), \\ \lambda - o_i^k(e_2) + o_i^j(e_1) &\geq l_i(e_2), \end{aligned}\right\} \text{ if } o_i^j(e_1) \leq o_i^k(e_2), \tag{6}$$

for all $e_1, e_2 \in \mathcal{E}$, $1 \leq j \leq |b_i(e_1)|$, $1 \leq k \leq |b_i(e_2)|$.
The case $o_i^j(e_1) \leq o_i^k(e_2)$ is depicted in Fig. 2. We have $p_i^j(e_1) = 0$ and $p_i^k(e_2) = 1$ since $t_i^j(e_1) = o_i^j(e_1)$ and $t_i^k(e_2) = \lambda + o_i^k(e_2)$.

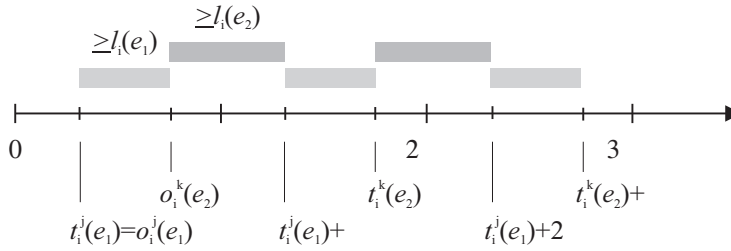

Fig. 2. *Conflict free use of channels*

The grey bars show the delay time that must exceed the communication time l_i of channel $\mathbf{w}_i$. Two remarks have to be added to the conflict free use of channels: 1) The implementation of several instances of channel $\mathbf{w}_i \in \mathcal{W}$ requires the solution of (6) only if e_1 and e_2 use the same instance of $\mathbf{w}_i$. 2) Particularly in the case of bit-serial channels $\mathbf{w}_i$ communication time $l_i(e)$ may exceed the iteration interval λ, i.e. $l_i(e) \geq \lambda$. Then $\lfloor l_i(e)/\lambda \rfloor$ instances of $\mathbf{w}_i$ are permanently used by communication $e \in \mathcal{E}$ and $l_i(e)$ has to be replaced by $(l_i(e) \mod \lambda)$ in (6). Both remarks are satisfied in the integer linear program formulation presented in the next section.

Next, we give a constraint for the solvability of the communication problem. The minimal communication time $\Delta t(\mathbf{v}(e))$ needed for the data transfer $e \in \mathcal{E}$ under assumption of unlimited instances of channels $\mathbf{w}_i \in \mathcal{W}$ is:

$$\Delta t(\mathbf{v}(e)) = \min_{\mathbf{v}=\sum_{i=1}^{|\mathcal{W}|} b_i(e)\mathbf{w}_i} \sum_{i=1}^{|\mathcal{W}|} |b_i(e)| l_i(e), \quad b_i(e) \in \mathbb{Z}, \quad \forall \mathbf{w}_i \in \mathcal{W}^+ . b_i(e) \geq 0. \quad (7)$$

Note that a fast computation of the value of $\Delta t(\mathbf{v}(e))$ is possible if matrix $\mathbf{W} = (\mathbf{w}_1, \cdots \mathbf{w}_{|\mathcal{W}|})$ is e-unimodular which leads to an integer polyhedron $\{b_i(e) \mid \mathbf{v} = \sum_{i=1}^{|\mathcal{W}|} b_i(e)\mathbf{w}_i\}$ [11].
In order to get a solution of the communication problem the causality constraint (4) has to be replaced by:

$$\boldsymbol{\tau}^T \mathbf{d}(e) + t_{\delta(e)} - t_{\sigma(e)} \geq d_{\sigma(e)} + \Delta t(\mathbf{v}(e)), \quad \forall e \in \mathcal{E}. \quad (8)$$

A value needed for the program formulation in the next section is the maximal number of channels $\mathbf{w}_i = (w_{i1}, \cdots, w_{in-1})^T \in \mathcal{W}$ that can be used for data transfer $e \in \mathcal{E}$. We call this value $b_i^{max}(e)$. If we restrict the data transfer to have no steps backwards in each component of $v_j(e)$, $1 \leq j \leq n-1$, of the displacement vector $\mathbf{v}(e)$ then $b_i^{max}(e)$ can be computed by:

$$b_i^{max}(e) = \max\{\max_{sign(w_{ij})v_j(e) \geq b_i w_{ij}, 1 \leq j \leq n-1} b_i, 0\}, \quad 1 \leq i \leq |\mathcal{W}|, \forall e \in \mathcal{E}. \quad (9)$$

For $\mathbf{w}_i \in \mathcal{W}^*$ it is easy to prove that $b_i^{max}(e) = 0$ either for $\mathbf{w}_i$ or for $-\mathbf{w}_i$.

$$\textbf{minimize: } \sum_{i=1}^{|\mathcal{W}|} n_i c_i, \qquad n_i \in \mathbb{Z}, \qquad \#1$$

subject to:

$$\mathbf{v}(e) = \sum_{i=1}^{|\mathcal{W}|} \sum_{j=1}^{b_i^{max}(e)} \beta_i^j(e)\mathbf{w}_i, \qquad \beta_i^j(e) \in \{0,1\}, \forall e \in \mathcal{E}, \qquad \#2$$

$$\sum_{k=1}^{N_i} \alpha_{ik}^j(e) = 1, \qquad \alpha_{ik}^j(e) \in \{0,1\}, 1 \le i \le |\mathcal{W}|, 1 \le j \le b_i^{max}(e), \forall e \in \mathcal{E}, \qquad \#3$$

$$n_i \ge \sum_{k=1}^{N_i} k\alpha_{ik}^j(e) - (1-\beta_i^j(e))N_i + \sum_{e \in \mathcal{E}} \sum_{j=1}^{b_i^{max}} \beta_i^j(e)\lfloor l_i(e)/\lambda \rfloor, \qquad \#4$$
$$1 \le j \le b_i^{max}(e), \quad \forall e \in \mathcal{E}_{(l_i(e) \bmod \lambda) \ne 0}, 1 \le i \le |\mathcal{W}|,$$

$$t^{start}(e) \le p_i^j(e)\lambda + o_i^j(e), \qquad \#5$$
$$t^{end}(e) \ge p_i^j(e)\lambda + o_i^j(e) + l_i(e),$$
$$p_i^j(e) \in \mathbb{Z}, 1 \le j \le b_i^{max}(e), 1 \le i \le |\mathcal{W}|, \forall e \in \mathcal{E},$$

$$p_i^j(e)\lambda + o_i^j(e) - p_k^l(e)\lambda - o_k^l(e) \ge l_k(e) - (\gamma_{ik}^{jl}(e) + (1-\beta_i^j(e)) + (1-\beta_k^l(e)))C_k(e), \qquad \#6$$
$$p_k^l(e)\lambda + o_k^l(e) - p_i^j(e)\lambda - o_i^j(e) \ge l_i(e) - ((1-\gamma_{ik}^{jl}(e)) + (1-\beta_i^j(e)) + (1-\beta_k^l(e)))C_i(e),$$
$$\gamma_{ik}^{jl}(e) \in \{0,1\}, 1 \le j \le b_1^{max}(e), 1(+j \text{ if } i=k) \le l \le b_k^{max}(e), 1 \le i \le k \le |\mathcal{W}|, \forall e \in \mathcal{E},$$

$$o_i^j(e_1) - o_i^k(e_2) \ge l_i(e_2) - (\delta_i^{jk}(e_1,e_2) + (2-\alpha_{il}^j(e_1) - \alpha_{il}^k(e_2)) + (2-\beta_i^j(e_1) - \beta_i^k(e_2)))(2\lambda + l_i(e_2)), \qquad \#7$$
$$\lambda - o_i^j(e_1) + o_i^k(e_2) \ge l_i(e_1) - (\delta_i^{jk}(e_1,e_2) + (2-\alpha_{il}^j(e_1) - \alpha_{il}^k(e_2)) + (2-\beta_i^j(e_1) - \beta_i^k(e_2)))l_i(e_1),$$
$$o_i^k(e_2) - o_i^j(e_1) \ge l_i(e_1) - ((1-\delta_i^{jk}(e_1,e_2)) + (2-\alpha_{il}^j(e_1) - \alpha_{il}^k(e_2)) + (2-\beta_i^j(e_1) - \beta_i^k(e_2)))(2\lambda + l_i(e_1)),$$
$$\lambda - o_i^k(e_2) + o_i^j(e_1) \ge l_i(e_2) - ((1-\delta_i^{jk}(e_1,e_2)) + (2-\alpha_{il}^j(e_1) - \alpha_{il}^k(e_2)) + (2-\beta_i^j(e_1) - \beta_i^k(e_2)))l_i(e_2),$$
$$1 \le l \le N_i, 1 \le j \le b_i^{max}(e_1), 1(+j \text{ if } e_1 = e_2) \le k \le b_i^{max}(e_2), \forall e_1, e_2 \in \mathcal{E}, 1 \le i \le |\mathcal{W}|.$$

Table 1. Integer linear program of the communication problem

5 Solution of the Communication Problem

In this section we present an integer linear program formulation of the communication problem. The entire program is listed in table 1. The constraints in Table 1 are explained in the following:

#2 (Decomposition of displacement vectors): Variable $b_i(e)$ is substituted by $b_i(e) = \sum_{j=1}^{b_i^{max}(e)} \beta_i^j(e)$, where $\beta_i^j(e)$ determines whether channel $\mathbf{w}_i \in \mathcal{W}$ is used for data transfer $e \in \mathcal{E}$ or not. Channel $\mathbf{w}_i$ has to be replaced by $-\mathbf{w}_i$ if $\mathbf{w}_i \in \mathcal{W}^*$.

#3 (Assignment to instances of channels) : Instance $\sum_{k=1}^{N_i} k\alpha_{ik}^j(e)$ of channel $\mathbf{w}_i$ is used for communication $e \in \mathcal{E}$, N_i is the maximal number of instances of channel $\mathbf{w}_i$ leading away from one processors.

#4 (Number of instances of channels): The first part ensures that instance $\sum_{k=1}^{N_i} k\alpha_{ik}^j(e)$ of channel $\mathbf{w}_i$ has to be considered only if $\beta_i^j(e) = 1$. The second part includes additional instances of channel $\mathbf{w}_i$ if $l_i(e) \ge \lambda$ as discussed in the previous section.

#5, #6 (Causality of data transfer): The binary variables γ_{ik}^{jl} are used to decide whether the j-th communication on channel $\mathbf{w}_i$ starts before or after the l-th communication on channel $\mathbf{w}_k$. Constant $C_i(e)$ can be determined by $C_i(e) = t^{end}(e) - t^{start}(e) + l_i(e)$.

#7 (Conflict free use of channels): The binary variables $\delta_i^{jk}(e_1,e_2)$ decide the if part of constraint (6). As mentioned in the previous section $l_i(e)$ has to be replaced by $(l_i(e) \mod \lambda)$ if $l_i(e) \ge \lambda$.

The integer program consists of $|\mathcal{W}| + 2\sum_{e\in\mathcal{E}}\sum_{i=1}^{|\mathcal{W}|} b_i^{max}(e)$ integer and $\sum_{e\in\mathcal{E}}\sum_{i=1}^{|\mathcal{W}|}(1 + N_i)b_i^{max}(e) + \sum_{e\in\mathcal{E}}(\sum_{i=1}^{|\mathcal{W}|} b_i^{max}(e)(\sum_{i=1}^{|\mathcal{W}|} b_i^{max}(e) - 1))/2 + \sum_{i=1}^{|\mathcal{W}|}(\sum_{e\in\mathcal{E}} b_i^{max}(e)(\sum_{e\in\mathcal{E}} b_i^{max}(e) - 1))/2$ binary variables. The number of constraints is $3\sum_{e\in\mathcal{E}}\sum_{i=1}^{|\mathcal{W}|} b_i^{max}(e) + \sum_{e\in\mathcal{E}}(\sum_{i=1}^{|\mathcal{W}|} b_i^{max}(e)(\sum_{i=1}^{|\mathcal{W}|} b_i^{max}(e) - 1)) + |\mathcal{E}|(n-1) + 2\sum_{i=1}^{|\mathcal{W}|}\gamma_i(\sum_{e\in\mathcal{E}} b_i^{max}(e)(\sum_{e\in\mathcal{E}} b_i^{max}(e) - 1))$.

6 Experimental Results

Because of lack of space we present only one example. We assume that we have three data transfers and four available channels. The related data are summarized in Table 2. The iteration interval is given with $\lambda = 8$.

	e_1	e_2	e_3
$\mathbf{v}(e)$	$(1,0)^T$	$(3,2)^T$	$(4,1)^T$
$t^{start}(e)$	2	4	7
$t^{end}(e)$	18	26	28

	$\mathbf{w}_1$	$\mathbf{w}_2$	$\mathbf{w}_3$	$\mathbf{w}_4$
	$(1,0)^T$	$(0,1)^T$	$(1,1)^T$	$(2,1)^T$
$l_i(e)$	4	4	4	16
c_i	8	8	8	2

Table 2. Data transfers and available channels

Application of the integer program leads to the result that minimal cost for the implementation of channels arise using two instances of channel $\mathbf{w}_1$ and one instance of channels $\mathbf{w}_3$ and $\mathbf{w}_4$. The starting time of each use of a channel is shown in Table 3 and depicted as a bar chart in Figure 3.

$t_1^1(e_1)$	$t_3^1(e_2)$	$t_4^1(e_2)$	$t_1^1(e_3)$	$t_1^2(e_3)$	$t_1^3(e_3)$	$t_3^1(e_3)$
11	4	10	7	11	15	24

Table 3. Starting time of communications

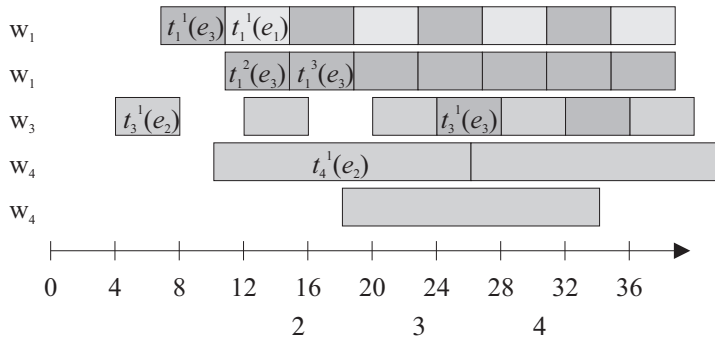

Fig. 3. Bar chart of communications

The linear program has 36 integer and 148 binary variables and consists of 588 constraints if we set $N_i = 3$, $1 \le i \le 4$. The solution takes about 28 seconds on a SUN-SPARC station.

7 Conclusion and Further Research

The presented approach is suitable to implement the data transfer in parallel processor arrays using local communications. The objective of the approach is the minimization of implementation cost for the channels. Only small changes to the integer program are required to match a given interconnection topology. To this end, (#1) in Table 1 can either be replaced by **minimize**: $\sum_{i=1}^{|\mathcal{W}|} n_i$ or left empty, whereas the constraint $n_i \leq N_i$, $1 \leq i \leq |\mathcal{W}|$, has to be added.
The iteration interval λ as well as $t^{start}(e)$ and $t^{end}(e)$ depend on the scheduling function $\tau(\mathbf{i})$. Hence, a more consequent approach consists in solving the communication problem and determining the scheduling function in one problem. This can be done by adopting techniques presented in [5]. In principle, this allows to determine a scheduling function, the functionality of processors and the channels between processors by solving one optimization problem. Unfortunately, the arising integer program tends to long solution time even for small problems.
The presented method also applies to the problem of limited communication support in partitioned processor arrays and extends the approach in [1].

References

[1] W.H. Chou and S.Y. Kung. Scheduling partitioned algorithms on processor arrays with limited communication support. In *Proc. IEEE Int. Conf. on Application Specific Systems, Architectures and Processors'93*, pages 53–64, Venice, 1993.
[2] A. Darte and Y. Robert. Constructive methods for scheduling uniform loop nests. *IEEE Trans. on Parallel and Distributed Systems*, 5(8):814–822, 1994.
[3] M. Dion, T. Risset, and Y. Robert. Resource constraint scheduling of partitioned algorithms on processor arrays. *Integration, the VLSI Journal*, 20:139–159, 1996.
[4] D. Fimmel and R. Merker. Determination of the processor functionality in the design of processor arrays. In *Proc. IEEE Int. Conf. on Application Specific Systems, Architectures and Processors'97*, pages 199–208, Zürich, 1997.
[5] D. Fimmel and R. Merker. Design of processor arrays for real-time applications. In *Proc. Int. Conf. Euro-Par '98*, pages 1018–1028, Southampton, 1998. Lecture Notes in Computer Science, Springer.
[6] J.A.B. Fortes and D.I. Moldovan. Parallelism detection and transformation techniques useful for vlsi algorithms. *Journal of Parallel and Distributed Computing*, 2:277–301, 1985.
[7] R.M. Karp, R.E. Miller, and S. Winograd. The organization of computations for uniform recurrence equations. *Journal of the ACM*, 14:563–590, 1967.
[8] S.Y. Kung. *VLSI Array Processors*. Prentice Hall, Englewood Cliffs, 1987.
[9] P.Z. Lee and Z.M. Kedem. Mapping nested loop algorithms into multidimensional systolic arrays. *IEEE Trans. on Parallel and Distributed Systems*, 1:64–76, 1990.
[10] S.K. Rao. *Regular Iterative Algorithms and their Implementations on Processor Arrays*. PhD thesis, Stanford University, 1985.
[11] A. Schrijver. *Theory of Integer and Linear Programming*. John Wiley & Sons, New York, 1986.
[12] L. Thiele. Resource constraint scheduling of uniform algorithms. *Int. Journal on VLSI and Signal Processing*, 10:295–310, 1995.
[13] Y. Wong and J.M. Delosme. Optimal systolic implementation of n-dimensional recurrences. In *Proc. ICCD*, pages 618–621, 1985.