Efficient Realization of Data Dependencies in Algorithm Partitioning Under Resource Constraints

Sebastian Siegel and Renate Merker

Dresden University of Technology, Institute of Circuits and Systems,
01062 Dresden, Germany
`<siegel, merker>@iee1.et.tu-dresden.de`

Abstract. Mapping algorithms to parallel architectures efficiently is very important for a cost-effective design of many modern technical products. In this paper, we present a solution to the problem of efficiently realizing uniform data dependencies on processor arrays. In contrary to existing approaches, we formulate an optimization problem to consider the cost of both: channels and registers. Further, a solution to the optimization problem assigns which channels shall be implemented and it specifies the control for the realization of the uniform data dependencies. We illustrate our method on the edge detection algorithm.

1 Introduction

Many modern technical products need to cope with fast digital signal, image and video processing under real-time requirements. Massively parallel data processing on processor arrays (PAs) is known to accelerate compute-intensive algorithms. The semiconductor industry presents more and more solutions for implementations of PAs in portable and other embedded systems. These solutions range from ASICs, reconfigurable systems in FPGAs, arrays of CPU-cores to platforms such as DRP from NEC [1] and picoArray [2].

PAs are mainly characterized by their processing elements (PEs). Some PEs are connected to the periphery, e. g. via a memory hierarchy (Fig. 1 (b)). The PEs are characterized by functional units and local memory. This paper focuses on the communication within the PA which is realized by regular local interconnections between the PEs.

To exploit the processing performance of PAs, we apply a new design flow which consists of mainly **two steps: 1)** partitioning the algorithm in order to match the PA parameters such as shape of PA, number of PEs, communication to a memory hierarchy [3] and **2)** realizing the data dependencies of the algorithm on the PA. This paper deals with the second step.

The first step can be summarized as follows: We consider compute-intensive algorithms e. g. described as systems of uniform recurrence equations (SUREs) [4]. They consist of many elementary computational tasks, the so-called *iterations*, that are aligned in an iteration space. Using our parameterized partitioning method [5], we map algorithms to PAs, i. e. each iteration to a PE (allocation) and determine the corresponding time of execution (schedule). The optimal use of the data path of each PE for the operations of the algorithm is addressed in [6–9].

This paper considers the second step: the realization of the data dependencies of an algorithm on the PA. The data dependencies cause data transfer within the PA. This data transfer is realized using *channels* between PEs and *registers* within PEs. Other existing works consider only one or the other. The optimal use of channels for the realization of data dependencies is considered in [10] for one-dimensional PAs and in [11] embedded in the traditional design flow which applies linear space-time mappings. The optimal use of registers is regarded in [12, 13] for a single processor machine or in [14] for the design of PAs based on the traditional design flow.

We present an approach which addresses the optimal use of both: channels and registers for the realization of the uniform data dependencies of the algorithms. It is important to consider channels and registers in one model because channels with a delay (e. g. with a pipeline structure) can reduce the usage of registers.

Given several channels and their implementation cost, we formulate and solve the *communication problem* by integer linear programming (ILP). A solution to the communication problem specifies which channels shall be implemented and the control of the data dependencies, i. e. when a channel or a register is used to realize a data dependency. This solution also includes the specification of an inner schedule for all computational tasks (given by the statements of the SURE) within an iteration. Our approach can be extended by existing methods concerning the optimal use of the data path.

To select the channels with minimum cost, our method can be applied in several ways. E. g. in reconfigurable computing where different algorithms shall be implemented on the same PA, we can determine which channels would best realize the communication within the PA for each algorithm. This information combined with the reconfiguration cost can lead to an efficient solution for the communication. In [15] we consider savings in the communication cost by avoiding redundancy. There it is necessary to know the channel selection (binding of data dependencies to channels) a priori. We apply the method presented in this paper to determine an efficient channel selection.

This paper is organized as follows. We describe the notation of algorithms as SUREs and we summarize partitioning in Sect. 2. Section 3 is the main contribution of this paper. In this section we derive a method to formulate and solve the communication problem. We give a solution to the communication problem for an example application in Sect. 4. Finally we draw some conclusions in Sect. 5.

2 Algorithm Coding and Partitioning

To demonstrate our methods for the communication problem, we consider systems of uniform recurrence equations which are defined as follows:

Definition 1 (System of Uniform Recurrence Equations (SURE)). *A system of uniform recurrence equations consists of a set of J statements $(1 \leq j \leq J)$ of the form*

$$S_j : \quad y_j\left[\mathbf{i}\right] := f_j\left(\ldots, y_i\left[\mathbf{i} - \widetilde{\mathbf{d}}^{r}_{j,i}\right], \ldots\right),\ \forall \mathbf{i} \in \mathcal{I}_j,\ i, j \in \mathbb{N}$$

where $\mathcal{I}_j$ denotes the iteration space of statement S_j, $\mathcal{I}_j$ is a polyhedral subset of a $\mathbb{Z}$-module and f_j denotes a single-valued function.

A variable y_i that is computed by statement S_i is a dependent variable if it is input to some (other) statement S_j. Vector $\widetilde{\mathbf{d}}^{r}_{j,i}$ denotes the corresponding uniform data dependency. Upper index "r" is used only if more that one data dependency exists between statements S_j and S_i.

With the embedding given by $\mathcal{I} = \mathrm{conv}(\bigcup_j \mathcal{I}_j) \subset \mathbb{Z}^n$ we determine the iteration space $\mathcal{I}$ of the SURE. In Algorithm 1 we show the edge detection algorithm (EDA) in the notation of a SURE which we use throughout this paper as an example.

Algorithm 1: Edge detection algorithm (EDA)

```
$S_1 :$     $q\left[\begin{smallmatrix}x\\y\end{smallmatrix}\right] = 2 \cdot p_i\left[\begin{smallmatrix}x\\y\end{smallmatrix}\right],$     $\left(\begin{smallmatrix}x\\y\end{smallmatrix}\right) \in \mathcal{I}_1 = \{\left(\begin{smallmatrix}x\\y\end{smallmatrix}\right) \in \mathbb{Z}^2 \mid \begin{smallmatrix}0\leq x\leq N-1\\0\leq y\leq M-1\end{smallmatrix}\}$
$S_2 :$     $h_1\left[\begin{smallmatrix}x\\y\end{smallmatrix}\right] = p_i\left[\begin{smallmatrix}x\\y-1\end{smallmatrix}\right] + p_i\left[\begin{smallmatrix}x\\y+1\end{smallmatrix}\right],$     $\left(\begin{smallmatrix}x\\y\end{smallmatrix}\right) \in \mathcal{I}_2 = \{\left(\begin{smallmatrix}x\\y\end{smallmatrix}\right) \in \mathbb{Z}^2 \mid \begin{smallmatrix}0\leq x\leq N-1\\1\leq y\leq M-2\end{smallmatrix}\}$
$S_3 :$     $h_2\left[\begin{smallmatrix}x\\y\end{smallmatrix}\right] = h_1\left[\begin{smallmatrix}x\\y\end{smallmatrix}\right] + q\left[\begin{smallmatrix}x\\y\end{smallmatrix}\right],$     $\left(\begin{smallmatrix}x\\y\end{smallmatrix}\right) \in \mathcal{I}_3 = \mathcal{I}_2$
$S_4 :$     $v_1\left[\begin{smallmatrix}x\\y\end{smallmatrix}\right] = p_i\left[\begin{smallmatrix}x-1\\y\end{smallmatrix}\right] + p_i\left[\begin{smallmatrix}x+1\\y\end{smallmatrix}\right],$     $\left(\begin{smallmatrix}x\\y\end{smallmatrix}\right) \in \mathcal{I}_4 = \{\left(\begin{smallmatrix}x\\y\end{smallmatrix}\right) \in \mathbb{Z}^2 \mid \begin{smallmatrix}1\leq x\leq N-2\\0\leq y\leq M-1\end{smallmatrix}\}$
$S_5 :$     $v_2\left[\begin{smallmatrix}x\\y\end{smallmatrix}\right] = v_1\left[\begin{smallmatrix}x\\y\end{smallmatrix}\right] + q\left[\begin{smallmatrix}x\\y\end{smallmatrix}\right],$     $\left(\begin{smallmatrix}x\\y\end{smallmatrix}\right) \in \mathcal{I}_5 = \mathcal{I}_4$
$S_6 :$     $h_3\left[\begin{smallmatrix}x\\y\end{smallmatrix}\right] = h_2\left[\begin{smallmatrix}x-2\\y-1\end{smallmatrix}\right] - h_2\left[\begin{smallmatrix}x\\y-1\end{smallmatrix}\right],$     $\left(\begin{smallmatrix}x\\y\end{smallmatrix}\right) \in \mathcal{I}_6 = \{\left(\begin{smallmatrix}x\\y\end{smallmatrix}\right) \in \mathbb{Z}^2 \mid \begin{smallmatrix}2\leq x\leq N-1\\2\leq y\leq M-1\end{smallmatrix}\}$
$S_7 :$     $h_4\left[\begin{smallmatrix}x\\y\end{smallmatrix}\right] = |h_3\left[\begin{smallmatrix}x\\y\end{smallmatrix}\right]|,$     $\left(\begin{smallmatrix}x\\y\end{smallmatrix}\right) \in \mathcal{I}_7 = \mathcal{I}_6$
$S_8 :$     $v_3\left[\begin{smallmatrix}x\\y\end{smallmatrix}\right] = v_2\left[\begin{smallmatrix}x-1\\y-2\end{smallmatrix}\right] - v_2\left[\begin{smallmatrix}x-1\\y\end{smallmatrix}\right],$     $\left(\begin{smallmatrix}x\\y\end{smallmatrix}\right) \in \mathcal{I}_8 = \mathcal{I}_6$
$S_9 :$     $v_4\left[\begin{smallmatrix}x\\y\end{smallmatrix}\right] = |v_3\left[\begin{smallmatrix}x\\y\end{smallmatrix}\right]|,$     $\left(\begin{smallmatrix}x\\y\end{smallmatrix}\right) \in \mathcal{I}_9 = \mathcal{I}_6$
$S_{10} :$  $s\left[\begin{smallmatrix}x\\y\end{smallmatrix}\right] = h_4\left[\begin{smallmatrix}x\\y\end{smallmatrix}\right] + v_4\left[\begin{smallmatrix}x\\y\end{smallmatrix}\right],$     $\left(\begin{smallmatrix}x\\y\end{smallmatrix}\right) \in \mathcal{I}_{10} = \mathcal{I}_6$
$S_{11} :$  $p_o\left[\begin{smallmatrix}x-1\\y-1\end{smallmatrix}\right] = \min(255, s\left[\begin{smallmatrix}x\\y\end{smallmatrix}\right]),$     $\left(\begin{smallmatrix}x\\y\end{smallmatrix}\right) \in \mathcal{I}_{11} = \mathcal{I}_6$
```

Next we briefly describe locally parallel, globally sequential (LPGS) partitioning [5] and we use it as a parameterized method to directly map an algorithm to a PA. LPGS-partitioning separates an iteration $\mathbf{i} \in \mathcal{I}$ into $\widehat{\boldsymbol{\kappa}}$ (denoting a partition) and $\boldsymbol{\kappa}$ (representing the position within a partition) by the tiling step as follows [5]:

$$\mathbf{i} = \boldsymbol{\Theta}\widehat{\boldsymbol{\kappa}} + \boldsymbol{\kappa}, \qquad 0 \leq \kappa_k < \vartheta_k,\ 1 \leq k \leq n,\ \widehat{\boldsymbol{\kappa}} \in \widehat{\mathcal{K}} \subset \mathbb{Z}^n,\ \boldsymbol{\kappa} \in \mathcal{K} \subset \mathbb{N}^n \qquad (1)$$

where $\boldsymbol{\Theta} = \mathrm{diag}(\vartheta_1 \cdots \vartheta_n) \in \mathbb{N}^{n \times n}$ is a square matrix whose diagonal elements represent the size of the partitions in each of the n directions of the iteration space $\mathcal{I}$. The size of the partitions corresponds to the size of the PA. Only two elements of $\mathbf{\Theta}$ may be greater than one to obtain a two-dimensional PA. The PA consists of $\prod_{i=1}^{n} \vartheta_i$ PEs.

We extend the scheduling function from [5] to determine the time of execution for each statement S_j of an iteration given by $\widehat{\boldsymbol{\kappa}}$ and $\boldsymbol{\kappa}$ as follows:

$$t_j(\widehat{\boldsymbol{\kappa}}, \boldsymbol{\kappa}) = \lambda \boldsymbol{\tau} \widehat{\boldsymbol{\kappa}} + \boldsymbol{\tau}^{\text{offs}} \boldsymbol{\kappa} + b_j \quad \text{with} \quad \boldsymbol{\tau}, \boldsymbol{\tau}^{\text{offs}} \in \mathbb{Z}^{1 \times n}, \;\; b_j \in \mathcal{L} \;\; . \tag{2}$$

The first term in (2) determines the starting time for each partition. The second term allows to shift the schedule within a partition according to $\boldsymbol{\tau}^{\text{offs}}$ to avoid data dependency conflicts or to change the time behavior of the I/O of the PA. With b_j we determine the starting time for each statement S_j within the iteration interval λ which is defined as follows:

Definition 2 (Iteration Interval λ). *The iteration interval λ denotes the number of time steps between the beginning of two successive iterations. The set $\mathcal{L} = \{0, 1, 2, \ldots, \lambda - 1\}$ consists of these time steps.*

Note that the parameters b_j will be specified by a solution to the communication problem.

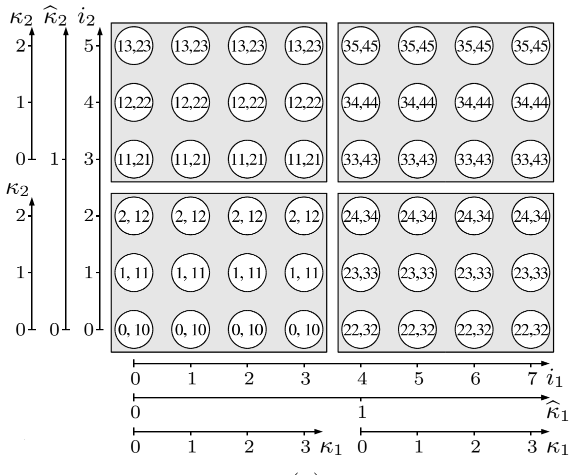
In Fig. 1 (a) we show LPGS-Partitioning for the EDA. Each circle denotes an iteration. The numbers within each circle describe the beginning and the end of the corresponding iteration interval. In Fig. 1 (b) we illustrate the obtained PA with a memory hierarchy. Memory L_0 denotes local registers whose cost will be determined in the communication problem. Some boundary PEs are connected to the memory hierarchy whose size depends on the tile size and the size of the image (M and N). We refer to [3] where we describe how to determine this memory hierarchy. The local interconnections are not depicted in Fig. 1 (b). They will be determined by solving the communication problem.

We introduce the set $\widetilde{\mathcal{D}}$ which comprises all uniform data dependencies with $\forall \widetilde{\mathbf{d}} \in \widetilde{\mathcal{D}}$: $\widetilde{d}_k < \vartheta_k$ where $1 \leq k \leq n$. For a data dependency $\widetilde{\mathbf{d}} \in \widetilde{\mathcal{D}}$ there exists at least one iteration within a partition serving as the source of the data dependency $\widetilde{\mathbf{d}}$ and there exists at least one iteration in the same partition serving as the corresponding drain.

Let $\vartheta_{k_1} > 1$ and $\vartheta_{k_2} > 1$ represent the two dimensions of the partitions. Then, only the elements $\widetilde{d}_{k_1}$ and $\widetilde{d}_{k_2}$ of any vector $\widetilde{\mathbf{d}} \in \widetilde{\mathcal{D}}$ can be greater than zero. To make things easier (especially for $\dim(\mathcal{I}) > 2$), we introduce the set $\mathcal{D} \in \mathbb{Z}^2$ as follows: Each vector $\mathbf{d} \in \mathcal{D}$ corresponds to one and only one vector $\widetilde{\mathbf{d}} \in \widetilde{\mathcal{D}}$ with $d_1 = \widetilde{d}_{k_1}$ and $d_2 = \widetilde{d}_{k_2}$. Hence there exists a bijective mapping between the sets $\widetilde{\mathcal{D}}$ and $\mathcal{D}$. Note that one-dimensional partitions can be regarded as a special case where w. l. o. g. we set $\vartheta_{k_2} = 1$.

For the example of the EDA we consider partitions with $\vartheta_1, \vartheta_2 \geq 3$. Therefore, all the 13 data dependencies which can be extracted from Algorithm 1 belong to the set $\mathcal{D}_{\text{EDA}}$:

$$\begin{aligned}\mathcal{D}_{\text{EDA}} = \big\{ &\mathbf{d}_{3,1} = \tbinom{0}{0}, \; \mathbf{d}_{3,2} = \tbinom{0}{0}, \; \mathbf{d}_{5,1} = \tbinom{0}{0}, \; \mathbf{d}_{5,4} = \tbinom{0}{0}, \; \mathbf{d}^1_{6,3} = \tbinom{2}{1}, \mathbf{d}^2_{6,3} = \tbinom{0}{1}, \mathbf{d}_{7,6} = \tbinom{0}{0}, \\ &\mathbf{d}^1_{8,5} = \tbinom{1}{2}, \; \mathbf{d}^2_{8,5} = \tbinom{1}{0}, \; \mathbf{d}_{9,8} = \tbinom{0}{0}, \mathbf{d}_{10,7} = \tbinom{0}{0}, \; \mathbf{d}_{10,9} = \tbinom{0}{0}, \mathbf{d}_{11,10} = \tbinom{0}{0} \big\} \;\; .\end{aligned}$$



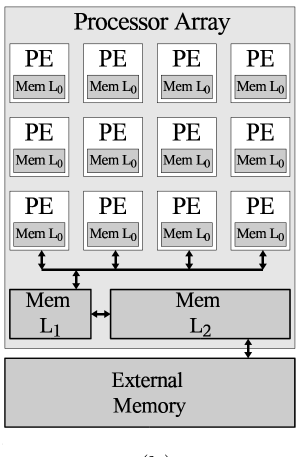


Fig. 1. (a) LPGS-Partitioning of the EDA with an iteration space of size $M \times N = 8 \times 6$ according to $\mathbf{\Theta} = \mathrm{diag}(4\ 3)$ and $t_j(\widehat{\boldsymbol{\kappa}}, \boldsymbol{\kappa}) = 11 \cdot (2\ 1) \cdot \widehat{\boldsymbol{\kappa}} + (0\ 1) \cdot \boldsymbol{\kappa} + b_j$, (b) Corresponding PA with a 2-level memory hierarchy

3 Communication Problem

The uniform data dependencies given by the set $\mathcal{D}$ need to be realized by a conflict free organization of the data transfer they cause. The communication problem consists in minimizing the implementation cost in terms of channels and registers for these data dependencies. In order to determine this cost, we introduce a model which allows a description of the communication.

3.1 Modelling the Communication

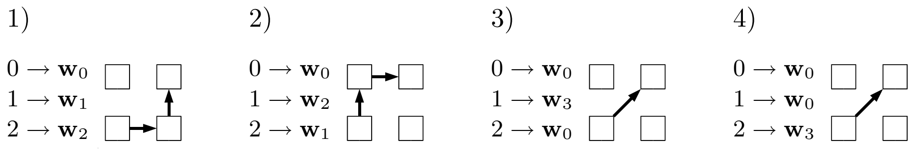
A set $\mathcal{W} = \{\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_{|\mathcal{W}|}\}$ of channels $\mathbf{w}_i \in \mathbb{Z}^2$ between PEs is supposed to be given. The elements of $\mathcal{W}^{\mathrm{bi}} \subseteq \mathcal{W}$ denote the channels that may be used bidirectional. To each element $\mathbf{w}_k$ of $\mathcal{W}$ corresponds a delay $l_k^{\mathbf{w}} \in \mathbb{N}$ which represents the time it takes to transfer an instance of a variable on that channel. Each channel with $l_k^{\mathbf{w}} = 0$ denotes a broadcast. Channels with $l_k^{\mathbf{w}} > 0$ represent a pipeline structure which may only be used onedirectional. Hence the delay of channel $\mathbf{w}_k$ may only be non-zero if $\mathbf{w}_k \in \mathcal{W} \setminus \mathcal{W}^{\mathrm{bi}}$.

The realization of data dependency $\mathbf{d}_{j,i}$ consists of two parts. First, it realizes the transfer of an instance of variable y_i from its source to the relative position given by $\mathbf{d}_{j,i}$. Second, the realization is responsible for the storage of that instance until it is input to a data path which executes statement S_j.

We model the transfer of each instance of a dependent variable y_i by a *sequence of moves* which describes the path from its source to its drain. A sequence of moves m where $0 \leq m \leq M_{j,i}$ is characterized by a mapping

$\{0, 1, 2, \ldots, M_{j,i}\} \longrightarrow \mathcal{W} \cup \{\mathbf{w}_0\}$ which determines the order in which the channels are used for a realization of data dependency $\mathbf{d}_{j,i}$. With $M_{j,i}$ we denote the maximum number of moves it may take to realize a data dependency $\mathbf{d}_{j,i}$. We determine each $M_{j,i}$ a priori according to [16]. The element $\mathbf{w}_0$ represents "no transfer". With $m = 0$ we describe an initial "move" which will always be mapped to $\mathbf{w}_0$.

Example 1. Suppose $\mathcal{W} = \{\mathbf{w}_1 = \left(\begin{smallmatrix}1\\0\end{smallmatrix}\right), \mathbf{w}_2 = \left(\begin{smallmatrix}0\\1\end{smallmatrix}\right), \mathbf{w}_3 = \left(\begin{smallmatrix}1\\1\end{smallmatrix}\right)\}$, $\mathbf{d} = \left(\begin{smallmatrix}1\\1\end{smallmatrix}\right)$ and $M = 2$. The following four different mappings would realize data dependency $\mathbf{d}$:


1)
$0 \rightarrow \mathbf{w}_0$
$1 \rightarrow \mathbf{w}_1$
$2 \rightarrow \mathbf{w}_2$
2)
$0 \rightarrow \mathbf{w}_0$
$1 \rightarrow \mathbf{w}_2$
$2 \rightarrow \mathbf{w}_1$
3)
$0 \rightarrow \mathbf{w}_0$
$1 \rightarrow \mathbf{w}_3$
$2 \rightarrow \mathbf{w}_0$
4)
$0 \rightarrow \mathbf{w}_0$
$1 \rightarrow \mathbf{w}_0$
$2 \rightarrow \mathbf{w}_3$

To determine this mapping is one task of the communication problem. We add binary variables $\beta_{j,i,k,m}$ to the communication problem which parameterize this mapping as follows:

$$\beta_{j,i,k,m} = \begin{cases} 1 \text{ if channel } \mathbf{w}_k \text{ is used at move } m \text{ to realize data dep. } \mathbf{d}_{j,i} \\ 0 \text{ otherwise} \end{cases} . \quad (3)$$

In order to assure that each move is mapped to one and only one channel, the variables $\beta_{j,i,k,m}$ are subject to:

$$\forall \mathbf{d}_{j,i} \in \mathcal{D} : \textstyle\sum_{k=0}^{|\mathcal{W}|} \beta_{j,i,k,m} = 1, \quad 0 \leq m \leq M_{j,i} \quad \text{where} \quad \beta_{j,i,0,0} = 1 . \quad (4)$$

To avoid mappings which represent similar paths as given in Example 1 by realizations 3) and 4), we force mappings to channel $\mathbf{w}_0$ to be placed as far to the end as possible in the sequence of moves for $m \geq 1$. Therefore we add the following constraints:

$$\forall \mathbf{d}_{j,i} \in \mathcal{D} : \beta_{j,i,0,m+1} \geq \beta_{j,i,0,m}, \quad 1 \leq m \leq M_{j,i} - 1 . \quad (5)$$

To distinguish between the direction in which a bidirectional channel $\mathbf{w}_k \in \mathcal{W}^{\text{bi}}$ is used we add binary variables $\beta^{\text{bi}}_{j,i,k}$ to the communication problem which parameterize the direction of a potential use of channel $\mathbf{w}_k$ by data dependency $\mathbf{d}_{j,i}$ as follows:

$$\beta^{\text{bi}}_{j,i,k} = \begin{cases} 0 \text{ if channel } \mathbf{w}_k \text{ would be used in the direction given by } \mathbf{w}_k \\ 1 \text{ if channel } \mathbf{w}_k \text{ would be used in the direction given by } -\mathbf{w}_k \end{cases} \quad (6)$$

where $\mathbf{w}_k \in \mathcal{W}^{\text{bi}}$. The path of a dependent variable from its source to its drain is fixed once the sequence of moves is determined. To ensure that this path leads to the correct final destination, we add the following constraints:

$$\forall \mathbf{d}_{j,i} \in \mathcal{D} : \mathbf{p}^{\text{rel}}_{j,i,M_{j,i}} = \mathbf{d}_{j,i} \quad (7)$$

where $\mathbf{p}^{\text{rel}}_{j,i,m}$ describes the relative position where an instance of variable y_i is located within the PA (relative to its source) after the m^{th} move of the realization of data dependency $\mathbf{d}_{j,i}$. We determine $\mathbf{p}^{\text{rel}}_{j,i,m}$ as follows:

$$\mathbf{p}^{\text{rel}}_{j,i,m} = \sum_{m'=1}^{m} \Big(\sum_{\{k \,|\, \mathbf{w}_k \in \mathcal{W}^{\text{bi}}\}} \beta_{j,i,k,m'} (1 - 2\,\beta^{\text{bi}}_{j,i,k})\, \mathbf{w}_k + \sum_{\{k \,|\, \mathbf{w}_k \in \mathcal{W} \setminus \mathcal{W}^{\text{bi}}\}} \beta_{j,i,k,m'} \mathbf{w}_k \Big) \ . \quad (8)$$

In the following, we will regard the time behavior of the realization of the data dependencies. The causality of all data dependencies of the set $\mathcal{D}$ is ensured by the following constraints:

$$\forall \mathbf{d}_{j,i} \in \mathcal{D} : t^{\text{d}}_{j,i} = \underbrace{b_j - (b_i + l_i)}_{①} + \underbrace{\boldsymbol{\tau}^{\text{offs}} \cdot \mathbf{d}_{j,i}}_{②} - \underbrace{\textstyle\sum_{m=1}^{M_{j,i}} \sum_{k=1}^{|\mathcal{W}|} \beta_{j,i,k,m} \cdot l^{\mathbf{w}}_k}_{③} \geq 0 \ . \quad (9)$$

The delay $t^{\text{d}}_{j,i}$ equals the number of time steps for which an instance of the dependent variable y_i needs to be stored until it is used by statement S_j. Of course, this amount of time may not be negative (causality constraint). In (9), term ① determines the time between the availability of an instance of the dependent variable y_i and its use (disregarding the scheduling offset). With l_i we describe the number of time steps after which the result of statement S_i is available at the output register of the data path. We assume that the input of the data path can be connected directly to the end of any local channel or to any local register. Term ② takes the scheduling offset into account as it represents the relative time difference between the iteration serving as the source and the iteration serving as the drain of data dependency $\mathbf{d}_{j,i}$. And term ③ denotes the time it takes to transfer an instance of the dependent variable y_i from the source to the drain.

If $t^{\text{d}}_{j,i} > 0$, then it is necessary to store an instance of the dependent variable y_i along its path for $t^{\text{d}}_{j,i}$ time steps. We introduce variables $t^{\text{r}}_{j,i,m} \in \mathbb{N}$ in the communication problem to parameterize the storage of an instance of a dependent variable y_i along its path from the source to the drain of data dependency $\mathbf{d}_{j,i}$. The value of variable $t^{\text{r}}_{j,i,m}$ gives the number of time steps for which an instance of variable y_i is stored in a local register **after** the m^{th} move of the realization of data dependency $\mathbf{d}_{j,i}$. The variables $t^{\text{r}}_{j,i,m}$ are subject to:

$$\forall \mathbf{d}_{j,i} \in \mathcal{D} \,:\, t^{\text{d}}_{j,i} = \textstyle\sum_{m=0}^{M_{j,i}} t^{\text{r}}_{j,i,m} \ . \quad (10)$$

Equation (10) ascertains that an instance of the dependent variable y_i is stored for as many time steps as given by $t^{\text{d}}_{j,i}$ along its path. Each variable $t^{\text{r}}_{j,i,m}$ with $m = 0$ denotes the time of storage for dependent variable y_i at the source of data dependency $\mathbf{d}_{j,i}$, i. e. before y_i is transported anywhere. This explains why we always map move $m = 0$ to channel $\mathbf{w}_0$. Note that our model above is also valid for data dependencies with $\mathbf{d}_{j,i} = \mathbf{0}$ where we use $M_{j,i} = 0$.

3.2 The Objective Function

The values of variables $\beta_{j,i,k,m}$, $\beta^{\text{bi}}_{j,i,k}$, and $t^{\text{r}}_{j,i,m}$ fully describe the realization of all data dependencies on the PA. In the following, we will determine the objective function of the communication problem. Hence we need to derive the cost for channels and registers. This cost can only be determined indirectly from variables $\beta_{j,i,k,m}$, $\beta^{\mathrm{bi}}_{j,i,k}$, and $t^{\mathrm{r}}_{j,i,m}$. Therefore we introduce further variables in the communication problem through which the cost can be described.

With $t^{\mathrm{in}}_{j,i,m}$ we describe the time at the PE at the relative position $\mathbf{p}^{\mathrm{rel}}_{j,i,m}$ at which an instance of variable y_i arrives as the m^{th} move of the realization of data dependency $\mathbf{d}_{j,i}$. And with $t^{\mathrm{out}}_{j,i,m}$ we describe the time at which that instance of variable y_i leaves the PE to perform move $m+1$ of the realization of data dependency $\mathbf{d}_{j,i}$. Variables $t^{\mathrm{in}}_{j,i,m}$ and $t^{\mathrm{out}}_{j,i,m}$ are determined as follows:

$$t^{\mathrm{in}}_{j,i,m} = \underbrace{b_i + l_i}_{①} + \underbrace{\textstyle\sum_{m'=1}^{m} \sum_{k=1}^{|\mathcal{W}|} \beta_{j,i,k,m'} \cdot l^{\mathbf{w}}_k}_{②} + \underbrace{\textstyle\sum_{m'=0}^{m-1} t^{\mathrm{r}}_{j,i,m'}}_{③} - \underbrace{\boldsymbol{\tau}^{\mathrm{offs}} \cdot \mathbf{p}^{\mathrm{rel}}_{j,i,m}}_{④} \ , \tag{11}$$

$$t^{\mathrm{out}}_{j,i,m} = t^{\mathrm{in}}_{j,i,m} + t^{\mathrm{r}}_{j,i,m} \ . \tag{12}$$

In (11), term ① determines the time when the source of a data dependency $\mathbf{d}_{j,i}$ is available. Term ② represents the delay caused by the transfer until the m^{th} move. Term ③ gives the delay caused by the storage along its path. And term ④ accounts for the time difference between the source and the position after the m^{th} move of the realization of the data dependency according to the scheduling offset.

Note that for $m = 0$, the time $t^{\mathrm{in}}_{j,i,0}$ denotes the time when an instance of variable y_i is fetched from the data path at the relative position $\mathbf{p}^{\mathrm{rel}}_{j,i,0} = \mathbf{0}$. After the last move ($m = M_{j,i}$), an instance of variable y_i arrives at the drain of data dependency $\mathbf{d}_{j,i}$. And time $t^{\mathrm{out}}_{j,i,M_{j,i}}$ denotes the time when that instance serves as an input to the data path at the relative position $\mathbf{p}^{\mathrm{rel}}_{j,i,M_{j,i}} = \mathbf{d}_{j,i}$ for the computation of statement S_j.

For $\boldsymbol{\tau}^{\mathrm{offs}} = \mathbf{0}$ in (2), it would be sufficient to consider one iteration interval λ of one PE to describe the communication problem [11]. For the general case where the scheduling offset may also be non-zero, we have to use an extended approach to solve the communication problem.

As a consequence we determine a priori for each data dependency $\mathbf{d}_{j,i} \in \mathcal{D}$ a set of time steps $\mathcal{L}'_{j,i}$. The set $\mathcal{L}'_{j,i}$ is defined in a similar way to the set $\mathcal{L}$ (see Def. 2) with the only difference that it takes the scheduling offset into account so that it may consist of some different time steps. We refer to [16] for a detailed derivation of how to determine the set $\mathcal{L}'_{j,i}$.

We introduce binary variables which account for the use of channels and registers caused by the realization of data dependency $\mathbf{d}_{j,i} \in \mathcal{D}$. Binary variable $\gamma_{j,i,k,m,l'}$ denotes whether a channel $\mathbf{w}_k$ is used at time step $l' \in \mathcal{L}'_{j,i}$ at the m^{th} move of the realization of that data dependency. And binary variable $\delta_{j,i,m,l'}$ denotes whether a register is used at time $l' \in \mathcal{L}'_{j,i}$ **after** the m^{th} move of the realization of that data dependency. The binary variables $\gamma_{j,i,k,m,l'}$ and $\delta_{j,i,m,l'}$ are determined as follows:

$$\gamma_{j,i,k,m,l'} = \left\{ \begin{array}{l} 1 \text{ if } l' = t^{\mathrm{out}}_{j,i,m-1} + \beta^{\mathrm{bi}}_{j,i,k} \boldsymbol{\tau}^{\mathrm{offs}} \mathbf{w}_k \ \wedge \ \beta_{j,i,k,m} = 1 \\ 0 \text{ otherwise} \end{array} \right\} , \ l' \in \mathcal{L}'_{j,i}, \ m > 0 \ , \tag{13}$$

$$\delta_{j,i,m,l'} = \left\{ \begin{array}{l} 1 \text{ if } t^{\text{in}}_{j,i,m} \leq l' \wedge l' < t^{\text{out}}_{j,i,m} \\ 0 \text{ otherwise} \end{array} \right\} \quad \text{with} \quad l' \in \mathcal{L}'_{j,i} \ . \tag{14}$$

Note that for a channel with a pipeline structure, the binary variable $\gamma_{j,i,k,m,l'}$ takes the value of one only for the first time step during which the channel is used for the realization of data dependency $\mathbf{d}_{j,i}$. Hence, another communication can begin to use the same channel at the next time step.

The number of channels $\mathbf{w}_k$ used at time $l \in \mathcal{L}$ caused by the realization of data dependency $\mathbf{d}_{j,i}$ is given by variable $c^{\mathbf{w}}_{j,i,k,l}$. The corresponding number of registers that is used at time l is given by $c^{\text{r}}_{j,i,l}$. We determine variables $c^{\mathbf{w}}_{j,i,k,l}$ and $c^{\text{r}}_{j,i,l}$ as follows:

$$c^{\mathbf{w}}_{j,i,k,l} = \sum_{\{l' \in \mathcal{L}'_{j,i} \mid l' \bmod \lambda = l\}} \quad \sum_{m=1}^{M_{j,i}} \gamma_{j,i,k,m,l'} \quad \text{with} \quad l \in \mathcal{L} \ , \tag{15}$$

$$c^{\text{r}}_{j,i,l} = \sum_{\{l' \in \mathcal{L}'_{j,i} \mid l' \bmod \lambda = l\}} \quad \sum_{m=0}^{M_{j,i}} \delta_{j,i,m,l'} \quad \text{with} \quad l \in \mathcal{L} \ . \tag{16}$$

In (15) and (16), the first sum considers all time steps of the set $\mathcal{L}'_{j,i}$ that will be mapped to time $l \in \mathcal{L}$ by modulo arithmetics. And the second sum adds over the moves that realize data dependency $\mathbf{d}_{j,i}$.

Next we determine the total cost for the realization of all data dependencies of the set $\mathcal{D}$. Variable $c^{\mathbf{w}}_k$ denotes the maximum number of channels $\mathbf{w}_k$ that is used at an arbitrary PE. And variable c^{r} denotes the maximum number of registers used at an arbitrary PE. The values of variables $c^{\mathbf{w}}_k$ and c^{r} are determined as follows:

$$c^{\mathbf{w}}_k = \max_{l \in \mathcal{L}} \sum_{\mathbf{d}_{j,i} \in \mathcal{D}} c^{\mathbf{w}}_{j,i,k,l} \qquad \text{and} \qquad c^{\text{r}} = \max_{l \in \mathcal{L}} \sum_{\mathbf{d}_{j,i} \in \mathcal{D}} c^{\text{r}}_{j,i,l} \ . \tag{17}$$

Finally, the objective function of the communication problem is determined as follows:

$$\min \left(\eta^{\text{r}} \cdot c^{\text{r}} + \sum_{k=1}^{|\mathcal{W}|} \eta^{\mathbf{w}}_k \cdot c^{\mathbf{w}}_k \right) \tag{18}$$

where η^{r} denotes the cost for a register and $\eta^{\mathbf{w}}_k$ denotes the cost for channel $\mathbf{w}_k$.

4 Experimental Results

For the EDA (Algorithm 1) we discuss two different target architectures (PA1 and PA2) as given in Table 1. The cost for a register is $\eta^{\text{r}} = 1$ for PA1 and $\eta^{\text{r}} = 1.5$ for PA2. In both cases, we assume one data path with a latency of one for solving each statement S_j within each PE. Hence, we search for a solution to the communication problem with $\lambda = 11$. The scheduling offset is $\boldsymbol{\tau}^{\text{offs}} = \mathbf{0}$.

In Fig. 2 we illustrate the optimal solution to the communication problem. The starting time of each use of a channel and/or register within the iteration interval is shown. The gray background in the use of channel $\mathbf{w}_5$ denotes a use in the negative direction. The inner schedule is given by the succession of the statements S_j.

ILPs were generated to solve the communication problem. Both ILPs consist of 3838 constraints, 2001 binary and 220 integer variables. The ILPs were solved

Table 1. Available channels $\mathbf{w}_k$, their latency $l_k^{\mathbf{w}}$ and their cost $\eta_k^{\mathbf{w}}$

(PA1)

k	1	2	3	4	5
$\mathbf{w}_k$	$(1\ 0)^T$	$(0\ 1)^T$	$(1\ 1)^T$	$(1\ {-1})^T$	$\pm(1\ 0)^T$
$l_k^{\mathbf{w}}$	3	3	1	2	0
$\eta_k^{\mathbf{w}}$	1.5	1.5	1	2.5	2

(PA2)

k	1	2	3	4	5
$\mathbf{w}_k$	$(1\ 0)^T$	$(0\ 1)^T$	$(1\ 1)^T$	$(1\ {-1})^T$	$\pm(1\ 0)^T$
$l_k^{\mathbf{w}}$	2	2	1	2	0
$\eta_k^{\mathbf{w}}$	1	1	1.5	3	1.5

using ILOG CPLEX v. 9.1 on an Athlon 64 Processor 3800+. It took 40 sec. and 72 sec. for PA1 and PA2 respectively to find an optimal solution of each ILP (including the verification that the solution is optimal).

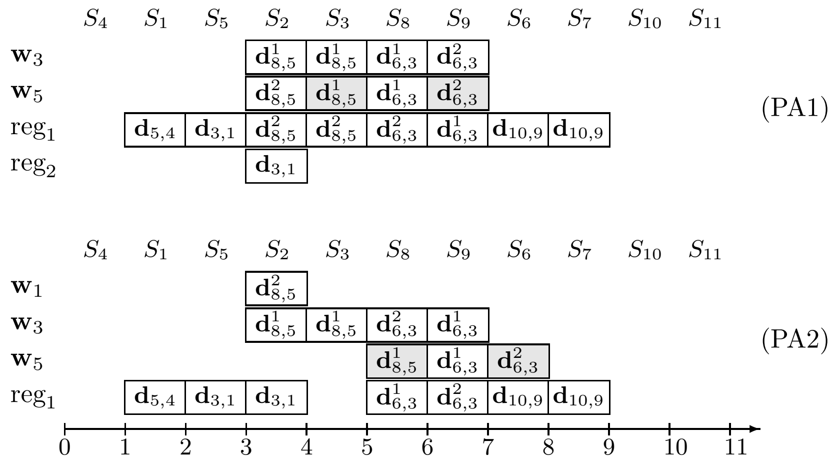


Fig. 2. Bar chart denoting the usage of channels and registers for the communication caused by all data dependencies of the set $\mathcal{D}_{\text{EDA}}$ within an iteration interval $\lambda = 11$.

5 Conclusions

In this paper we have formulated and solved the communication problem to realize uniform data dependencies within a PA using ILP. Our method takes the cost of channels and registers of the target architecture into account. A solution to the communication problem determines a selection of channels and the control of the communication caused by the uniform data dependencies. Further, it specifies the inner schedule for all computational tasks within an iteration.

Our method can also be used to find suitable realizations of the uniform data dependencies of algorithms on a given PA with fixed local interconnections. Future work includes an extension of our approach to non-uniform data dependencies (e. g. the realization of input/output) or to multi-level partitioning (e. g. co-partitioning).

References

1. Motomura, M.: A Dynamically Reconfigurable Processor Architecture. In: Microprocessor Forum. (2002)
2. Duller, A., Towner, D., Panesar, G., Gray, A., Robbins, W.: picoArray Technology: The Tool's Story. In: Design, Automation and Test in Europe (DATE'05). Volume 3. (2005) 106–111
3. Siegel, S., Merker, R.: Optimized Data-Reuse in Processor Arrays. In: Proc. IEEE Int. Conf. on Application-Specific Systems, Architectures, and Processors (ASAP 2004). (2004) 315–325
4. Karp, R.M., Miller, R.E., Winograd, S.: The organisation of computations for uniform recurrence equations. JACM **14**(3) (1967) 563–590
5. Siegel, S., Merker, R.: Algorithm Partitioning including Optimized Data-Reuse for Processor Arrays. In: Proc. IEEE International Conference on Parallel Computing in Electrical Engineering (PARELEC 2004). (2004) 85–90
6. Derrien, S., Rajopadhye, S., Sur-Kolay, S.: Combining Instruction and Loop Level Parallelism for Array Synthesis on FPGAs. In: International Symposium on System Synthesis. (2001)
7. Feautrier, P.: Fine-grain Scheduling under Resource Constraints. In: Proc. 7th Int. Workshop on Languages and Compilers for Parallel Computing (LCPC '94). Volume 892 of LNCS., Springer (1994) 1–15
8. Thiele, L.: Resource constraint scheduling of uniform algorithms. Journal of VLSI Signal Processing **10** (1995) 295–310
9. Teich, J., Thiele, L.: A New Approach to Solving Resource-Constrained Scheduling Problems based on a Flow-Model. Technical Report 17, TIK, Swiss Federal Institute of Technology (ETH) Zürich (1996)
10. Dion, M., Risset, T., Robert, Y.: Resource-constrained scheduling of partitioned algorithms on processor arrays. Integration, the VLSI Journal **20** (1996) 139–159
11. Fimmel, D., Merker, R.: Localization of Data Transfer in Processor Arrays. In: Parallel Processing: 5th International Euro-Par Conference (Euro-Par '99). Volume 1685 of LNCS., Springer (1999) 401–408
12. Eisenbeis, C., Sawaya, A.: Optimal loop parallelization under register constraints. Technical Report 2781, INRIA (1996)
13. Müller, J., Fimmel, D., Merker, R.: Optimal Loop Scheduling with Register Constraints Using Flow Graphs. In: Proc. of the 7th Int. Symposium on Parallel Architectures, Algorithms, and Networks (I-SPAN), Hong Kong, China (2004)
14. Fimmel, D.: Optimaler Entwurf paralleler Rechenfelder unter Verwendung ganzzahliger linearer Optimierung. PhD thesis, Dresden University of Technology, Institute of Circuits and Systems (2002)
15. Siegel, S., Merker, R.: Minimum Cost for Channels and Registers in Processor Arrays by Avoiding Redundancy. In: IEEE 17th Int. Conf. on Application-Specific Systems, Architectures, and Processors (ASAP 2006). (2006)
16. Siegel, S., Merker, R.: Optimal Realization of Uniform Data Dependencies in Algorithm Partitioning Under Resource Constraints. Technical report, Dresden University of Technology, Institute of Circuits and Systems (2006, http://www.iee.et.tu-dresden.de/~siegel/paper/SM06a.pdf)