

Output Serialization for FPGA-based and Coarse-grained Processor Arrays

Frank Hannig and Jürgen Teich,
Department of Computer Science 12,
Hardware-Software-Co-Design,
University of Erlangen-Nuremberg, Germany.
{hannig, teich}@cs.fau.de

Abstract

This paper deals with the mapping of loop programs onto processor arrays either implemented in an FPGA or available as (reconfigurable) coarse-grained processor architectures. Usually the proportion of processing elements to I/O-interfaces is much higher whereby problems of data transportation and synchronization are arising. In this realm, we propose a systematic approach in order to feed-out data. Here, (a) an efficient routing strategy is presented and (b) a novel retiming strategy is given in order to ensure collision free output serialization.

Keywords:

Processor Arrays, Hardware Mapping, Serialization.

1 Introduction

Today massively parallel architectures called processor arrays are of great interest, since progressive integration densities and modern nanotechnology allow implementations of hundreds of 32-bit microprocessors and more in one SoC. Moreover, with the advent of reconfigurable architectures, processor arrays have become as flexible as the design of software. Such arrays can solve efficiently a large number of problems in digital signal, image, and video processing, or numerical linear algebra.

Due to technology constraints usually there is an imbalance of the massive internal parallelism compared to the number of available I/O-ports. Therefore, several processors may have to share the same output-port.

In this context, our new contributions are,

1. An efficient routing and serialization strategy is presented.
2. A novel retiming methodology is given to allow for time division multiplexing of output data.

The rest of the paper is organized as follows. In Section 2, we give an overview of related work and briefly

describe our mapping methodology. In Section 3, we accurately formulate the problem we are dealing with in this paper, which is afterwards systematically solved in Section 4. Finally, we give some concluding remarks in Section 5.

2 Related Work and Background

In this section we give a brief overview of our existing mapping methodology PARO [13] when generating synthesizable descriptions of massively parallel processor arrays [2] and targeting coarse-grained reconfigurable processor arrays [9], respectively, from a class of nested loop algorithms. The main steps during the design trajectory are briefly described in the following.

Starting point is a given nested loop program in *single assignment code* (SAC), where the whole parallelism is explicitly expressed. SAC is closely related to a set of recurrence equations, a formalism introduced by Karp, Miller, and Winograd [11]. This formalism has been used in many languages and advanced over the years about affine dependencies or piecewise definitions. E.g., *Systems of Affine Recurrence Equations* (SARE) which are used in the Alpha language [18], the class of *Affine Indexed Algorithms* (AIA) [5], the class of *Piecewise Linear Algorithms* (PLA) [16], and recently, an algorithm class with runtime dependent conditionals, *Dynamic Piecewise Linear Algorithms* (DPLA) [10]. Mapping methodologies based on these formalisms can be classified to the area of loop parallelization in the polytope model [7]. Here, key components are linear transformations and schedules in order to derive preferably homogeneous processor arrays with local and regular communication structures and a high degree of pipelining and parallelism.

Also, linear transformations are used as *space-time mappings* in order to assign a processor p (space) and a sequencing index t (time) to instances (index space) of the given loop nest. Here, partitioning techniques

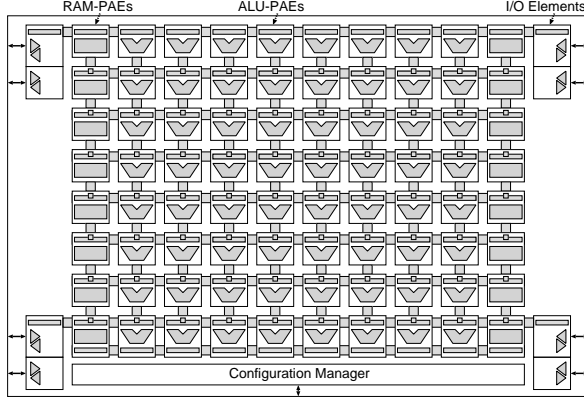


Figure 1: Structure of the PACT XPP64-A [1] reconfigurable 8×8 processor array. On two sides of the array each eight RAM blocks and in the four corners of the array independent I/O interfaces can be used as communication facilities.

are of utmost importance in order to cover the index space of computation using congruent hyperplanes, hyperquaders, or parallelepipeds called *tiles*. For processor arrays, it is carried out in order to match a loop nest implementation to resource constraints in terms of available number of processing elements (PEs), local memory, and communication bandwidth. Well known partitioning techniques are multiprojection, LSGP (local sequential global parallel, often also referred as clustering or blocking), LPGS (local parallel global sequential, also referred as tiling), and hierarchical combination of them such as co-partitioning in [5].

Definition 2.1 (*Block Pipelining Period*). *The block pipelining period of an allocated and scheduled algorithm is the time interval between the initiations of two successive problem instances and is denoted by β .*

With this background we can formulate our problem.

3 Problem Definition

In order to achieve high performance in combination with low monetary and energy cost, processor array architectures are preferable homogeneous and instead of having global memory access for every processor local communication structures to neighboring processor elements are used. Usually two-dimensional processor arrays exist in an FPGA or as a coarse-grained reconfigurable processor array such as the D-Fabrix [6], the DRP from NEC [12], the PACT XPP [1], Bresca from Silicon Hive (Philips) [15], or QuickSilver Technology's Adaptive Computing Machine (ACM) [14], have less communication ports to their periphery than

processing elements. An example of such an architecture is depicted in Fig. 1. In practice often two problems arise:

1. It can happen that also computation results from inner processor elements are needed externally.
2. The number of global output-ports is less than the amount of output data over all processors.

Therefore, we have to deal with the following questions: How can output data from inner processor elements be efficiently transported to output-ports? How and where should the output data of several processor elements sharing one output-port be merged/serialized?

4 Output Serialization

Let the 2-dimensional rectangular processor array be described by the following bounded \mathbb{Z} -polyhedron,

$$\mathcal{A} = \left\{ \begin{pmatrix} x \\ y \end{pmatrix} \in \mathbb{Z}^2 \mid 1 \leq x \leq W \wedge 1 \leq y \leq H \right\}$$

where $H \in \mathbb{Z}$ is the processor array's number of rows and $W \in \mathbb{Z}$ is the processor array's number of columns, respectively. Furthermore, we assume that all I/O-ports are located at the edges of polytope \mathcal{A} and that the interconnection topology of the array is a grid. Finally, after an appropriate space-time mapping has been performed consider an arbitrary convex area \mathcal{P} of processor elements,

$$\mathcal{P} = \left\{ \begin{pmatrix} x \\ y \end{pmatrix} \in \mathbb{Z}^2 \mid A \cdot \begin{pmatrix} x \\ y \end{pmatrix} \geq b \right\}$$

$A \in \mathbb{Z}^{m \times 2}, b \in \mathbb{Z}^m$

where each PE $p[x, y] \in \mathcal{P}$ emits one global output signal $o[x, y]$ and all these signals have to share one output-port $O[X, Y]$, where $(X, Y)^T \in \mathbb{Z}^2$ denotes an edge element of the array. Also given from the space-time mapping is an output-sequence of data. This output pattern is repeated with the duration of the block pipelining period β (s. Definition 2.1).

Definition 4.1 (*Output Profile*) *Let a block pipelining period β be given. Then, for a processor's output $o[x, y]$ the output profile is a mapping from time $t \mapsto f \in \{0, 1\}$, for all $t \in [0, \beta - 1] \subset \mathbb{Z}$. The profile is written as a vector, $\Delta_o[x, y] = (f_0, f_1, \dots, f_{\beta-1})$.*

Such we can assign to each output $o[x, y]$ a vector $\Delta[x, y]$ of length β with boolean elements denoting if $\Delta_o[x, y, t]$, i.e., a processor's output at position $(x, y)^T$ emits at relative time step t a value or not. Due to the usage of linear scheduling functions, often the considered output regions are homogeneous, i.e., they have all the same output profile rotated only by an offset [3].

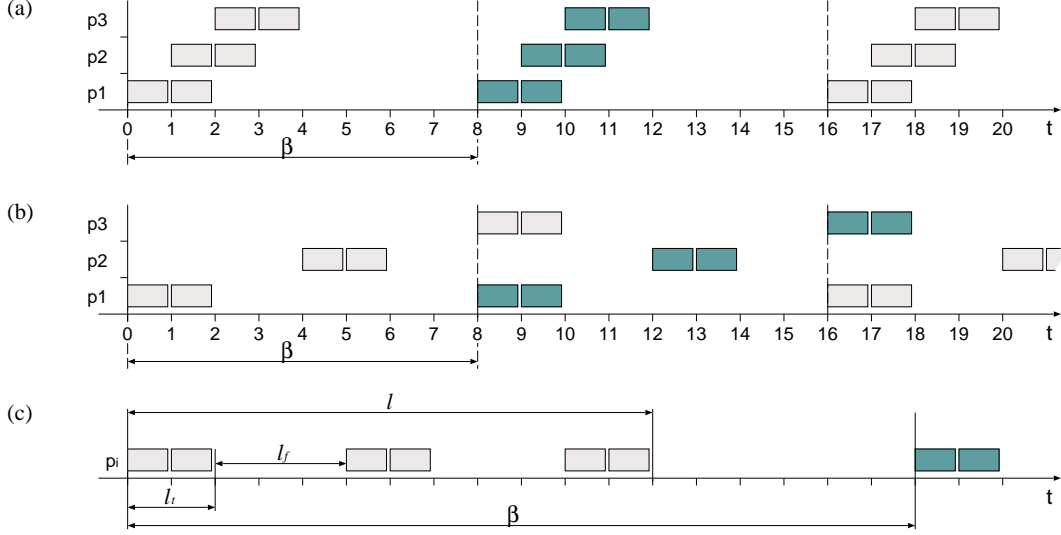


Figure 2: In (a) and (b) different output profiles for three processors p_1 , p_2 , and p_3 is shown. In (c), labeling of characteristics in periodic output profiles.

Example 4.1 Consider three processors p_1 , p_2 , p_3 , and a block pipelining period of $\beta = 8$. Furthermore the output profiles looks as follows,

$$\begin{aligned} p_1 : \quad \Delta_1 &= (1, 1, 0, 0, 0, 0, 0, 0) \\ p_2 : \quad \Delta_2 &= (0, 1, 1, 0, 0, 0, 0, 0) \\ p_3 : \quad \Delta_3 &= (0, 0, 1, 1, 0, 0, 0, 0) \end{aligned}$$

We write for instance $\Delta_{p_2}(t=0) = 0$ and $\Delta_{p_2}(1) = 1$. A visualization of these output profiles is shown in Fig. 2 (a). Another profile is shown in Fig. 2 (b), where can be seen that output data from different instances may overlap.

In the next section we introduce a necessary condition to determine whether $n = |\mathcal{P}|$ outputs with given output profiles can share one global output-port, i.e., if a bandwidth bound is satisfied or not. Afterwards, the routing and timing correct merging of output signals is investigated.

4.1 Bandwidth Bounds

In general the output data of $n = |\mathcal{P}|$ outputs $o[x, y, t]$ can be serialized if the following condition holds,

$$\sum_{\forall p[x, y] \in \mathcal{P}} \sum_{t=0}^{\beta-1} \Delta_o[x, y, t] \leq \beta \quad (1)$$

Or in case of homogeneity, Eq. (1) can be simplified to,

$$n \sum_{t=0}^{\beta-1} \Delta_o[x, y, t] \leq \beta, \quad \exists p[x, y] \in \mathcal{P} \quad (2)$$

This condition can be seen as a bandwidth constraint, since the number of output data of one area \mathcal{P} can not be greater than the block pipelining period. For instance in Example 4.1 the output behavior is homogeneous such we can write $3 \cdot 2 = 6 \leq \beta = 8$, thus the bandwidth constraint is satisfied. The $n : 1$ communication can be formally described by adding an equation of following type to the considered algorithm,

$$O[X, Y, t] = \text{merge}_{\forall p[x, y] \in \mathcal{P}} o[x, y, t] \quad (3)$$

Then all output variables o of processors $p \in \mathcal{P}$ shall be merged (timely multiplexed) at global output-port O in a way that they are timely conflict-free. Thus, the merging in Eq. (3) can be seen as OR relation,

$$O[X, Y, t] = \text{OR}_{\forall p[x, y] \in \mathcal{P}} o[x, y, t] \quad (4)$$

4.2 Routing and Merging of Output Data

In the following we present a methodology to determine a resource-optimal routing of PEs' outputs $o[x, y]$ to one output-port $O[X, Y]$. Since, we have to deal with a $n : 1$ communication relation, bottlenecks due to high communication near the output-ports are foreseen. Therefore, our main concept is to serialize already locally two signals, propagate the result to the next PE, serialize the signal with its output and so on. For the case of $1 : n$ communication, such linear serialization concepts also often called localization are well studied [17] whereas only few work for the $n : 1$ communication case exists, e.g., [4], [8], or our work in [9] which is limited to one-dimensional domains \mathcal{P} .

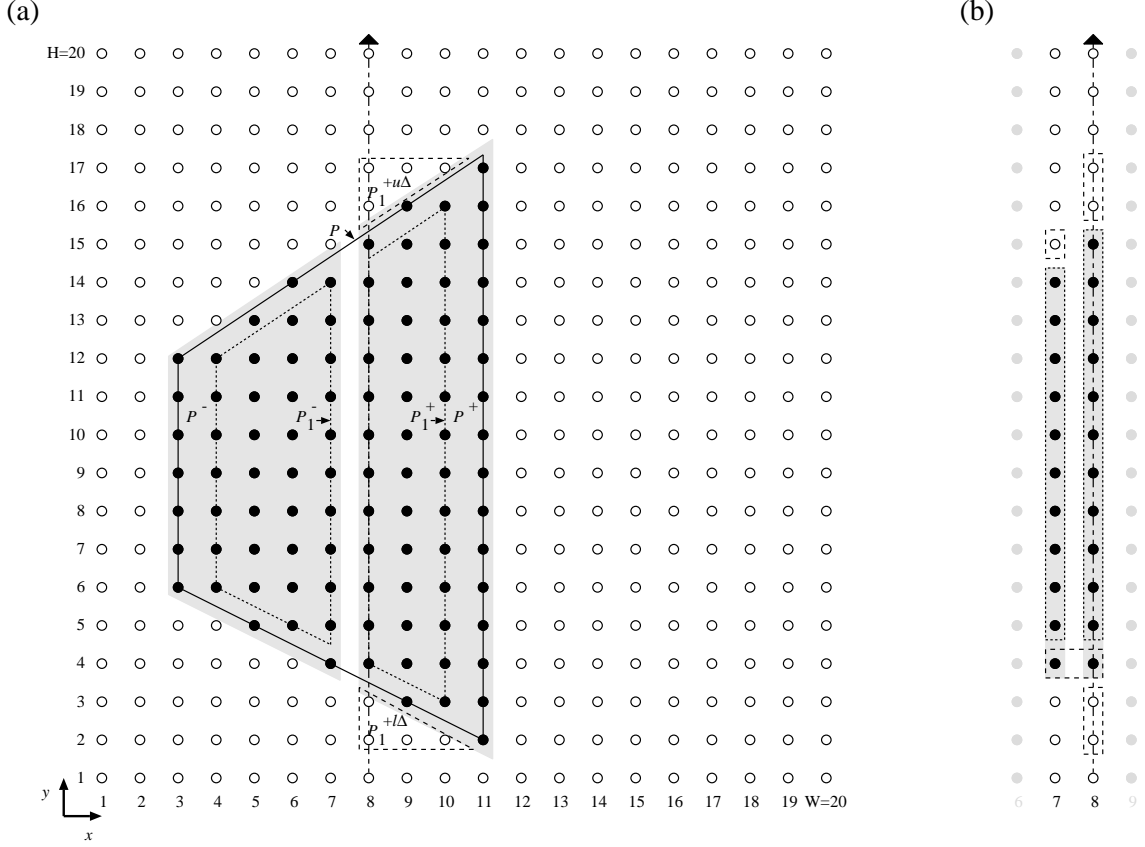


Figure 3: Sketch of a 20×20 processor array. The decomposition of \mathcal{P} into several spaces for the firstly horizontal propagation is shown in (a). In (b), the decomposition result of the second propagation in vertical direction is depicted.

Our procedure proposed in this paper can be formalized as follows, due to the assumed grid structure of the array four possible propagation directions are possible, viz, horizontally $\pm d_h = (1, 0)^T$ and vertically $\pm d_v = (0, 1)^T$. This assumption is no restriction, since our following method can also be extended by diagonal communication links (e.g., in direction $(1, 1)^T$) or for tree-dimensional technologies, but this would be beyond the scope of this paper and not helpful for the sake of clarity. Let $\mathcal{R}(\mathcal{P})$ be the rectangular hull of \mathcal{P} ,

$$\mathcal{R}(\mathcal{P}) = \left\{ \begin{pmatrix} x \\ y \end{pmatrix} \in \mathbb{Z}^2 \mid \begin{array}{l} X_{min} \leq x \leq X_{max} \\ Y_{min} \leq y \leq Y_{max} \end{array} \right\}$$

The considered processor area \mathcal{P} is divided by the location $(X, Y)^T$ of the output-port into a positive space \mathcal{P}^+ and a negative space \mathcal{P}^- . \mathcal{H}^+ denotes the points on the cut in \mathcal{P}^+ (i.e., one horizontal or vertical line of processor elements) and \mathcal{H}^- denotes the neighboring points

on the cut in \mathcal{P}^- .

$$\begin{aligned} \mathcal{P}^+ &= \left\{ \begin{pmatrix} x \\ y \end{pmatrix} \in \mathbb{Z}^2 \mid A_{\mathcal{P}^+} \cdot \begin{pmatrix} x \\ y \end{pmatrix} \geq b_{\mathcal{P}^+} \right\} \\ &= \begin{cases} \mathcal{P} \cap \left\{ \begin{pmatrix} x \\ y \end{pmatrix} \in \mathbb{Z}^2 \mid x \geq X \right\} & \text{if } (Y = 1 \vee Y = H) \\ \mathcal{P} \cap \left\{ \begin{pmatrix} x \\ y \end{pmatrix} \in \mathbb{Z}^2 \mid y \geq Y \right\} & \text{else} \end{cases} \\ \mathcal{P}^- &= \left\{ \begin{pmatrix} x \\ y \end{pmatrix} \in \mathbb{Z}^2 \mid A_{\mathcal{P}^-} \cdot \begin{pmatrix} x \\ y \end{pmatrix} \geq b_{\mathcal{P}^-} \right\} \\ &= \begin{cases} \mathcal{P} \cap \left\{ \begin{pmatrix} x \\ y \end{pmatrix} \in \mathbb{Z}^2 \mid x < X \right\} & \text{if } (Y = 1 \vee Y = H) \\ \mathcal{P} \cap \left\{ \begin{pmatrix} x \\ y \end{pmatrix} \in \mathbb{Z}^2 \mid y < Y \right\} & \text{else} \end{cases} \\ \mathcal{H}^+ &= \begin{cases} \mathcal{P} \cap \left\{ \begin{pmatrix} x \\ y \end{pmatrix} \in \mathbb{Z}^2 \mid x = X \right\} & \text{if } (Y = 1 \vee Y = H) \\ \mathcal{P} \cap \left\{ \begin{pmatrix} x \\ y \end{pmatrix} \in \mathbb{Z}^2 \mid y = Y \right\} & \text{else} \end{cases} \\ \mathcal{H}^- &= \begin{cases} \mathcal{P} \cap \left\{ \begin{pmatrix} x \\ y \end{pmatrix} \in \mathbb{Z}^2 \mid x = X - 1 \right\} & \text{if } (Y = 1 \vee Y = H) \\ \mathcal{P} \cap \left\{ \begin{pmatrix} x \\ y \end{pmatrix} \in \mathbb{Z}^2 \mid y = Y - 1 \right\} & \text{else} \end{cases} \end{aligned}$$

The rectangular hull of \mathcal{P}^+ and \mathcal{P}^- is given by $\mathcal{R}(\mathcal{P}^+)$ and $\mathcal{R}(\mathcal{P}^-)$, respectively,

$$\begin{aligned} \mathcal{R}(\mathcal{P}^+) &= \left\{ \begin{pmatrix} x \\ y \end{pmatrix} \in \mathbb{Z}^2 \mid \begin{array}{l} X_{min}^+ \leq x \leq X_{max}^+ \\ Y_{min}^+ \leq y \leq Y_{max}^+ \end{array} \right\} \\ \mathcal{R}(\mathcal{P}^-) &= \left\{ \begin{pmatrix} x \\ y \end{pmatrix} \in \mathbb{Z}^2 \mid \begin{array}{l} X_{min}^- \leq x \leq X_{max}^- \\ Y_{min}^- \leq y \leq Y_{max}^- \end{array} \right\} \end{aligned}$$

Then, the serialized output routing can be described as follows,

1. Propagation in first direction:

($d_1 \in \{(1, 0)^T, (0, 1)^T\}$)

$$o_1[I] = \begin{cases} o[I] & \forall I \in \mathcal{P}^+ \setminus \mathcal{P}_1^+ \\ \text{OR}(o[I], o_1[I+d_1]) & \forall I \in \mathcal{P}_1^+ \\ o_1[I+d_1] & \forall I \in \mathcal{P}_1^{+u\Delta} \cup \mathcal{P}_1^{+l\Delta} \\ o[I] & \forall I \in \mathcal{P}^- \setminus \mathcal{P}_1^- \\ \text{OR}(o[I], o_1[I-d_1]) & \forall I \in \mathcal{P}_1^- \\ o_1[I-d_1] & \forall I \in \mathcal{P}_1^{-u\Delta} \cup \mathcal{P}_1^{-l\Delta} \end{cases}$$

$$\mathcal{P}_1^+ = \mathcal{P}^+ \cap \left\{ I \in \mathbb{Z}^2 \mid A_{\mathcal{P}^+} \cdot (I+d_1) \geq b_{\mathcal{P}^+} \right\}$$

$$\mathcal{P}_1^- = \mathcal{P}^- \cap \left\{ I \in \mathbb{Z}^2 \mid A_{\mathcal{P}^-} \cdot (I-d_1) \geq b_{\mathcal{P}^-} \right\}$$

$$\mathcal{P}_1^{+u\Delta} = (\mathcal{R}(\mathcal{P}^+) \setminus \mathcal{P}^+) \cap \left\{ d_2^T \cdot I > h_{max}^+ \right\}$$

$$\mathcal{P}_1^{+l\Delta} = (\mathcal{R}(\mathcal{P}^+) \setminus \mathcal{P}^+) \cap \left\{ d_2^T \cdot I < h_{min}^+ \right\}$$

$$\mathcal{P}_1^{-u\Delta} = (\mathcal{R}(\mathcal{P}^-) \setminus \mathcal{P}^-) \cap \left\{ d_2^T \cdot I > h_{max}^- \right\}$$

$$\mathcal{P}_1^{-l\Delta} = (\mathcal{R}(\mathcal{P}^-) \setminus \mathcal{P}^-) \cap \left\{ d_2^T \cdot I < h_{min}^- \right\}$$

$$h_{max}^+ = \max \left\{ d_2^T \cdot I \mid I \in \mathcal{H}^+ \right\}$$

$$h_{min}^+ = \min \left\{ d_2^T \cdot I \mid I \in \mathcal{H}^+ \right\}$$

$$h_{max}^- = \max \left\{ d_2^T \cdot I \mid I \in \mathcal{H}^- \right\}$$

$$h_{min}^- = \min \left\{ d_2^T \cdot I \mid I \in \mathcal{H}^- \right\}$$

$$h_{max}^\pm = \min \{ h_{max}^+, h_{max}^- \}$$

$$h_{min}^\pm = \max \{ h_{min}^+, h_{min}^- \}$$

2. Propagation in second direction:

($d_2 \in \{(1, 0)^T, (0, 1)^T\} \setminus \{d_1\}$)

Similar to the propagation in the first direction a propagation in the second orthogonal direction is performed. This can be done by the insertion of instances of a second variable o_2 . Due to the sake of brevity we omit this procedure.

3. The data of the two spaces \mathcal{P}^+ and \mathcal{P}^- are merged at exact one fixed position $o_3[x, y]$.

4. Finally a path from $o_3[x, y]$ to $O[X, Y]$ is determined.

4.2.1 Example

Let us consider a 20×20 processor array \mathcal{A} and the following processor elements' area \mathcal{P} ,

$$\mathcal{P} = \left\{ \begin{pmatrix} x \\ y \end{pmatrix} \in \mathbb{Z}^2 \mid \begin{pmatrix} 2 & -3 \\ 1 & 0 \\ -1 & 0 \\ 1 & 2 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix} \geq \begin{pmatrix} -30 \\ 8 \\ -11 \\ 15 \end{pmatrix} \right\}$$

Let the output signal $o[x, y]$ of each processor element in \mathcal{P} be a global output signal which has to use the common I/O-port $O[X=8, Y=H=20]$. In Fig. 3 the array \mathcal{A} is depicted, the polytope \mathcal{P} is marked by black points, and the output $O[8, 20]$ is denoted as a small triangle. First of all, by the location of the I/O-port, \mathcal{P} is divided into \mathcal{P}^+ and \mathcal{P}^- by adding inequality $x \geq 8$ and $x < 8$ to \mathcal{P} , respectively¹. After removing redundant constraints,

¹Since $x \in \mathbb{Z}$, $x < 8$ can be rewritten to $-x + 7 \geq 0$.

\mathcal{P}^+ , \mathcal{P}^- , and \mathcal{H} are given by,

$$\mathcal{P}^+ = \left\{ \begin{pmatrix} x \\ y \end{pmatrix} \in \mathbb{Z}^2 \mid \begin{pmatrix} 2 & -3 \\ 1 & 0 \\ -1 & 0 \\ 1 & 2 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix} \geq \begin{pmatrix} -30 \\ 8 \\ -11 \\ 15 \end{pmatrix} \right\}$$

$$\mathcal{P}^- = \left\{ \begin{pmatrix} x \\ y \end{pmatrix} \in \mathbb{Z}^2 \mid \begin{pmatrix} 2 & -3 \\ 1 & 0 \\ -1 & 0 \\ 1 & 2 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix} \geq \begin{pmatrix} -30 \\ 3 \\ -7 \\ 15 \end{pmatrix} \right\}$$

$$\mathcal{H}^+ = \left\{ \begin{pmatrix} x \\ y \end{pmatrix} \in \mathbb{Z}^2 \mid x = 8 \wedge 4 \leq y \leq 15 \right\}$$

$$\mathcal{H}^- = \left\{ \begin{pmatrix} x \\ y \end{pmatrix} \in \mathbb{Z}^2 \mid x = 7 \wedge 4 \leq y \leq 14 \right\}$$

The rectangular hulls $\mathcal{R}(\mathcal{P}^+)$ and $\mathcal{R}(\mathcal{P}^-)$ are given by,

$$\mathcal{R}(\mathcal{P}^+) = \left\{ \begin{pmatrix} x \\ y \end{pmatrix} \in \mathbb{Z}^2 \mid 8 \leq x \leq 11 \wedge 2 \leq y \leq 17 \right\}$$

$$\mathcal{R}(\mathcal{P}^-) = \left\{ \begin{pmatrix} x \\ y \end{pmatrix} \in \mathbb{Z}^2 \mid 3 \leq x \leq 7 \wedge 4 \leq y \leq 14 \right\}$$

As first propagation direction $d_1 = (1, 0)^T$ is chosen and as a result the second direction is $d_2 = (0, 1)^T$. From this the lower routing equations follow,

$$o_1[I] = \begin{cases} o[I] & \forall I \in \mathcal{P}^+ \setminus \mathcal{P}_1^+ \\ \text{OR} \left(o[I], o_1 \left[I + \begin{pmatrix} 1 \\ 0 \end{pmatrix} \right] \right) & \forall I \in \mathcal{P}_1^+ \\ o_1 \left[I + \begin{pmatrix} 1 \\ 0 \end{pmatrix} \right] & \forall I \in \mathcal{P}_1^{+u\Delta} \cup \mathcal{P}_1^{+l\Delta} \\ o[I] & \forall I \in \mathcal{P}^- \setminus \mathcal{P}_1^- \\ \text{OR} \left(o[I], o_1 \left[I - \begin{pmatrix} 1 \\ 0 \end{pmatrix} \right] \right) & \forall I \in \mathcal{P}_1^- \\ o_1 \left[I - \begin{pmatrix} 1 \\ 0 \end{pmatrix} \right] & \forall I \in \mathcal{P}_1^{-u\Delta} \cup \mathcal{P}_1^{-l\Delta} \end{cases}$$

$$o_2[I] = \begin{cases} \vdots \end{cases}$$

$$o_3[I] = \text{OR} \left(o_2[I], o_2 \left[I - \begin{pmatrix} 1 \\ 0 \end{pmatrix} \right] \right) \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 8 \\ 17 \end{pmatrix}$$

Defined on the following domains:

$$\mathcal{P}_1^+ = \left\{ \begin{pmatrix} x \\ y \end{pmatrix} \in \mathbb{Z}^2 \mid \begin{pmatrix} 2 & -3 \\ 1 & 0 \\ -1 & 0 \\ 1 & 2 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix} \geq \begin{pmatrix} -30 \\ 8 \\ -10 \\ 15 \end{pmatrix} \right\}$$

$$\mathcal{P}_1^- = \left\{ \begin{pmatrix} x \\ y \end{pmatrix} \in \mathbb{Z}^2 \mid \begin{pmatrix} 2 & -3 \\ 1 & 0 \\ -1 & 0 \\ 1 & 2 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix} \geq \begin{pmatrix} -28 \\ 4 \\ -7 \\ 16 \end{pmatrix} \right\}$$

$$\mathcal{P}_1^{+u\Delta} = \left\{ \begin{pmatrix} x \\ y \end{pmatrix} \in \mathbb{Z}^2 \mid \begin{pmatrix} -2 & 3 \\ 1 & 0 \\ 0 & -1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix} \geq \begin{pmatrix} 30 \\ 8 \\ -17 \end{pmatrix} \right\}$$

$$\mathcal{P}_1^{+l\Delta} = \left\{ \begin{pmatrix} x \\ y \end{pmatrix} \in \mathbb{Z}^2 \mid \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ -1 & -2 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix} \geq \begin{pmatrix} 8 \\ 2 \\ -14 \end{pmatrix} \right\}$$

$$\mathcal{P}_1^{-u\Delta} = \mathcal{P}_1^{-l\Delta} = \emptyset$$

The complete serialized routing is visualized in Fig. 4.

4.2.2 Optimization Remarks

Depending on the hardware architecture and the number of already occupied merging and interconnection resources it has to be investigated if either first the horizontal or the vertical propagation yields in better results or vice versa.

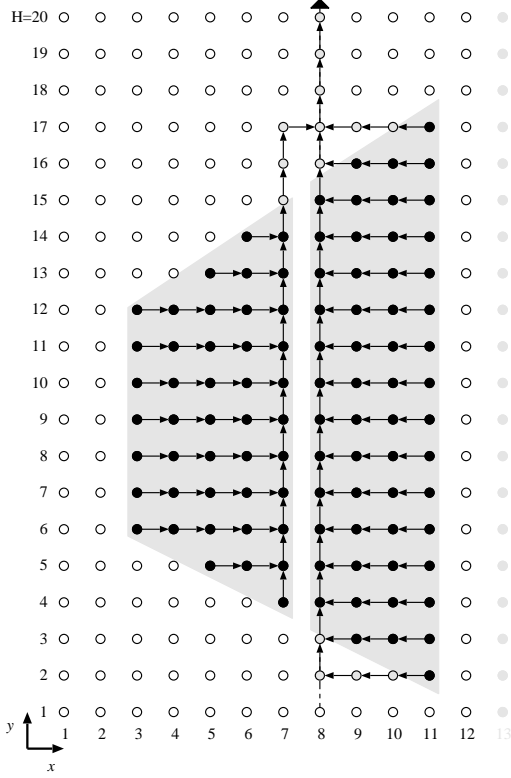


Figure 4: Serialized $n : 1$ output routing.

4.3 Retiming of Output Data

After finding a localized routing as described in Section 4.2 it remains to ensure that at no time and no position two values collide. In this matter the bandwidth constraint in Eq. (3) is only a sufficient condition but does not avoid data collision. To model the timing behavior, outputs $o[x, y]$ are annotated by the time t , i.e., $o[x, y, t]$. Critical for collisions is the merging in the OR operations. Without loss of generality we consider in the following only one regular serialization path denoted by $p[I_1] \rightarrow p[I_2] \rightarrow \dots \rightarrow p[I_k] \rightarrow \dots \rightarrow p[I_N]$.

Theorem 4.1 (*Collision Free*) An equation $o_3[I_k, t] = \text{OR}(o_2[I_k, t], o_1[I_{k-1}, (t - \delta) \div \beta])$ is collision free if the maximum over all elements in $\Delta_s[I_k, t]$ is less or equal than one,

$$\max_{d_i \in \Delta_s[I_k, t]} \{d_i\} \leq 1 \quad (5)$$

Where $\Delta_s[I_k, t]$ is the recursively summation of output profiles,

$$\Delta_s[I_i, t] = \begin{cases} \Delta[I_i, t] & \text{if } i = 1 \\ \Delta[I_i, t] + \Delta_s[I_{i-1}, (t - \delta) \div \beta] & \text{else} \end{cases}$$

δ denotes the number of time steps for propagating a value from one processor two its neighbor.

OutputRetiming

IN: number of outputs N , output profile Δ ,
block pipelining period β , scheduling offset δ
OUT: array $f[1..N]$

```

1 BEGIN
2    $(l, l_t, l_f) \leftarrow \text{computeL}(\Delta)$ 
3    $I \leftarrow \lfloor \frac{l_f}{l_t} \rfloor + 1$ 
4    $i \leftarrow 1$ 
5    $p \leftarrow 0$ 
6   WHILE  $(i \leq N)$  DO
7     FOR  $j = 0, \dots, I - 1$  DO
8       IF  $(i \leq N)$  THEN
9          $f[i] \leftarrow (j \cdot (l_t + \delta) + p) \div \beta$ 
10         $f[i] \leftarrow f[i] - \delta \cdot (i - 1)$ 
11      ENDIF
12       $i \leftarrow i + 1$ 
13    ENDFOR
14     $p \leftarrow p + l + I \cdot (\delta + l_t) - l_t$ 
15  ENDWHILE
16 END
```

Figure 5: Retiming Algorithm (' \div ' denotes a modulo division).

Proof. A proof by induction can be easily constructed by first checking the base case when $i = 1$ and successive induction. \square

In the following we present an efficient retiming methods for the already earlier discussed case of homogeneity where all $p[I_i]$ have the same output profile Δ merely rotated by a constant offset. Furthermore, the output profile should have only a periodic pattern such as described by the regular expression:

$$l_t \mid ((l_t l_f)^* l_t) \quad (6)$$

Let l be the length of the expression in Eq. (6), for an intuitive description see Fig. 2 (c). Then, if the condition in Eq. (7) holds, algorithm OutputRetiming in Fig. 5 computes a collision free retiming.

$$\begin{aligned}
a_{dense} + a_{remain} &\leq \beta & (7) \\
a_{dense} &= \lfloor N/I \rfloor \cdot (l + (I - 1) \cdot l_t) \\
a_{remain} &= \begin{cases} 0 & \text{if } N \div I = 0 \\ l + (N \div I) \cdot l_t & \text{else} \end{cases}
\end{aligned}$$

In Eq. (7), I denotes how many instances of an output profile can be interleaved $I = \lfloor l_f/l_t \rfloor + 1$. For instance, consider Fig. 2 (c) where $l_f = 3$ and $l_t = 2$, thus $I = \lfloor 3/2 \rfloor + 1 = 2$, i.e., two output patterns can be interleaved whereas a third one can start earliest $l + l_t$ time steps later.

Algorithm OutputRetiming gets as parameter an output profile Δ and a number N denoting the N outputs in the serialization path $p[I_1] \rightarrow \dots \rightarrow p[I_N]$. The scheduling offset δ denotes that $p[I_{i+1}]$ starts with the same output profile as $p[I_i]$ but δ time steps later. Fourth and last parameter is the block pipelining period β . As result of the algorithm an array $f[1..N]$ is returned, where each element $f[i]$ denotes the delay of output $o[I_i]$ in order to ensure a collision free serialization.

Example 4.2 Let $\Delta = (1, 0, 0, 1, 0, 0, 0, 0, 0, 0)$, $N = 4$, $\delta = 1$, and $\beta = 10$ be given. This means that output $o[I_1]$ emits at time step zero data. Due to the scheduling offset $\delta = 1$, $o[I_2]$ emits at time step zero no data but at time step one, at the same time the value sent one step earlier by $o[I_1]$ arrives at position I_2 and causes a collision. Thus we have to investigate if set of outputs is serializable and if so if an appropriate re-timing could be found. The bandwidth constraints in Eq. (2) is satisfied since the number over all '1' in Δ multiplied by N is equal to 8. Next we evaluate Eq. (7) which leads to $10 \leq \beta = 10$. Subsequent application of the proposed retiming algorithm leads to the following result, $f[1] = 0$, $f[2] = 1$, $f[3] = 2$, and $f[4] = 6$, e.g., outputs from $o[I_2]$ have to be delayed by one time step.

5 Conclusions and Outlook

Due to the enormous mismatch of computational power and I/O bandwidth sophisticated partitioning and scheduling techniques are important. We have presented an efficient transformation in order to serialize output data. Additionally, we have shown how a frequently existent class of output patterns could be retimed in order to avoid collisions. If such a retiming is not possible anymore but the bandwidth constraints still hold, FIFOs and more advanced control structures can be used to ensure a right timing behavior, but usually also for a higher price. In the future we would like to investigate the tighter coupling of our partitioning and scheduling techniques together with the methodologies described in this paper.

References

- [1] V. Baumgarte, G. Ehlers, F. May, A. Nüchel, M. Vorbach, and M. Weinhardt. PACT XPP – A Self-Reconfigurable Data Processing Architecture. *The Journal of Supercomputing*, 26(2):167–184, 2003.
- [2] M. Bednara and J. Teich. Synthesis of FPGA Implementations from Loop Algorithms. In *First International Conference on Engineering of Reconfigurable Systems and Algorithms (ERSA '01)*, pages 1–7, Las Vegas, NV, June 2001.
- [3] M. Bednara and J. Teich. Interface Synthesis for FPGA Based VLSI Processor Arrays. In *Proc. of the International Conference on Engineering of Reconfigurable Systems and Algorithms (ERSA 02)*, Las Vegas, Nevada, U.S.A., June 2002.
- [4] U. Eckhardt. *Algorithmus-Architektur-Codesign für den Entwurf digitaler Systeme mit eingebettetem Prozessorraster und Speicherhierarchie*. PhD thesis, Technische Universität Dresden, June 2001.
- [5] U. Eckhardt and R. Merker. Hierarchical Algorithm Partitioning at System Level for an Improved Utilization of Memory Structures. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 18(1):14–24, 1999.
- [6] Elixent Ltd. www.elixent.com
- [7] P. Feautrier. Automatic Parallelization in the Polytope Model. Technical Report 8, Laboratoire PRiSM, Université des Versailles St-Quentin en Yvelines, 45, avenue des États-Unis, F-78035 Versailles Cedex, June 1996.
- [8] G. Gupta, S. Rajopadhye, and P. Quinton. Scheduling Reductions on Realistic Machines. In *Proceedings 14th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA 2002)*, pages 117–126, Winnipeg, Manitoba, Canada, Aug. 2002. ACM Press.
- [9] F. Hannig, H. Dutta, and J. Teich. Mapping of Regular Nested Loop Programs to Coarse-grained Reconfigurable Arrays – Constraints and Methodology. In *Proceedings of the 18th International Parallel and Distributed Processing Symposium (IPDPS 2004)*, Santa Fe, NM, U.S.A., Apr. 2004. IEEE Computer Society.
- [10] F. Hannig and J. Teich. Resource Constrained and Speculative Scheduling of an Algorithm Class with Run-Time Dependent Conditionals. In *Proceedings of the 15th IEEE International Conference on Application-specific Systems, Architectures, and Processors (ASAP 2004)*, pages 17–27, Galveston, TX, U.S.A., Sept. 2004.
- [11] R. Karp, R. Miller, and S. Winograd. The Organization of Computations for Uniform Recurrence Equations. *Journal of the Association for Computing Machinery*, 14(3):563–590, 1967.
- [12] M. Motomura. A Dynamically Reconfigurable Processor Architecture. In *Microprocessor Forum*, CA, 2002.
- [13] PARO Design System Project. www12.informatik.uni-erlangen.de/research/paro
- [14] QuickSilver Technology. www.qstech.com
- [15] Silicon Hive. www.siliconhive.com
- [16] L. Thiele. *Computer Systems and Software Engineering: State-of-the-Art*, chapter 4, Compiler Techniques for Massive Parallel Architectures, pages 101–151. Kluwer Academic Publishers, Boston, U.S.A., 1992.
- [17] L. Thiele and V. Roychowdhury. Systematic design of local processor arrays for numerical algorithms. In *Algorithms and Parallel VLSI Architectures*, volume A: Tutorials, pages 329–339, Amsterdam, 1991. Elsevier.
- [18] D. Wilde and O. Sié. Regular Array Synthesis using Alpha. In *Int. Conf. on Application Specific Array Processors, San Francisco, California*, pages 200–211, Aug. 1994.