In the proceedings of the ICASSP 96, Atlanta, GA, May 7-10, 1996.

AN EXTENSION OF MULTIDIMENSIONAL SYNCHRONOUS DATAFLOW TO HANDLE ARBITRARY SAMPLING LATTICES

Praveen K. Murthy, Edward A. Lee

Dept. of EECS, University of California, Berkeley CA 94720

{murthy,eal}@eecs.berkeley.edu

Abstract[1]

Multidimensional Synchronous Dataflow (MDSDF) [5][2] is a model of computation that has been proposed and implemented for specifying multidimensional multirate signal processing systems such as image and video processing algorithms. The model is an extension of synchronous dataflow (SDF) [4] and has all of the desirable properties of the SDF model such as static schedulability, exposure of data and functional parallelism, and a visually pleasing syntax well suited for block diagram signal processing environments such as Ptolemy [1] and Khoros [3]. However, the MDSDF model as specified in [5] is limited to modeling multidimensional systems sampled on the rectangular lattice. Since some multidimensional systems of practical interest use non-rectangular sampling lattices and non-rectangular multirate operators like hexagonal decimators, models that are capable of representing and simulating such systems are of interest. This paper describes an extension of the MDSDF model that allows signals on arbitrary sampling lattices to be represented, and that allows the use of non-rectangular downsamplers and upsamplers.

1. Introduction

Dataflow has proven to be a useful programming model for use in software environments for simulating and prototyping signal processing systems. This is because such systems are increasingly being based on graphical, block-diagram programming systems[3][1], and a dataflow graph forms a very natural abstraction for a program specified as a block diagram. Most signal processing algorithms have a predictable flow of control; hence, it has been found that a particular model of dataflow, synchronous dataflow (SDF), is especially well suited for modeling unirate and multirate signal processing systems of arbitrary complexity. Chief amongst the many useful properties that this model has is the possibility of constructing schedules at compile time; this leads to very efficient implementations since overhead due to run-time decision making is completely avoided. Moreover, compile time schedulability implies that schedules can be optimized for any of several criteria including program and data memory usage [7], and high throughput multiprocessor schedules taking interprocessor communication into account [8]. The second possibility, namely that of constructing good *multiprocessor* implementations, is due to the ability of SDF to expose not just the functional parallelism but also *data* parallelism in the algorithm.

The standard dataflow models, including SDF, suffer from the limitation that their streams (sequences of tokens passed along the arcs) are unidimensional. Although a multidimensional object such as an array can be embedded in a one dimensional stream, it may be awkward to do so. In particular, compile- time information about data parallelism and flow of control may not be very clear [2]. Multidimensional Synchronous Dataflow (MDSDF) was proposed recently for specifying multidimensional systems [5]; this model is an extension of SDF and preserves all of the nice properties of SDF such as compile-time schedulability. Moreover, it is also capable of exposing data parallelism in multidimensional systems to a much greater extent than would be possible with multidimensional objects embedded in one dimensional streams in SDF. However, the model reported in [5] was limited to expressing systems sampled on the standard rectangular lattice. Since there is a benefit sometimes in using non-separable lattices for sampling multidimensional signals [9], and for using non-separable multirate operations such as hexagonal decimation or quincunx upsampling, it is of interest to have dataflow models that can also express these systems. In this paper, we present an extension of the MDSDF model to handle such systems, without sacrificing either compile-time schedulability or exposition of data-parallelism. In fact, extending the model without sacrificing these properties is the main challenge since it was conjectured in [5] that a dynamic dataflow model might have to be extended to multiple dimensions in order to model non-rectangular index spaces. While it is not clear to what extent the generalization presented in this paper can model arbitrary non-rectangular index spaces, it can certainly model the interesting subclass of applications (that require non-rectangular index spaces) arising out of the use of non-rectangular sampling lattices and multirate operators.

2. Non Rectangular Sampling

Consider the sequence of samples generated by

$$x(n_1, n_2) = x_a(a_{11}n_1 + a_{12}n_2, a_{21}n_1 + a_{22}n_2)$$

1. This research is part of the Ptolemy project, which is supported by the Advanced Research Projects Agency and the U.S. Air Force (under the RASSP program, contract F33615-93-C-1317), the Semiconductor Research Corporation (SRC) (project 95-DC-324-016), the National Science Foundation (MIP-9201605), the State of California MICRO program, and the following companies: Bell Northern Research, Cadence, Dolby, Hitachi, Lucky-Goldstar, Mentor Graphics, Mitsubishi, Motorola, NEC, Philips, and Rockwell.

where $x_a(t_1, t_2)$ is a continuous time signal. Notice that the sample locations retained are given by the equation

$$\hat{t} = \begin{bmatrix} t_1 \\ t_2 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} n_1 \\ n_2 \end{bmatrix} = V\hat{n}$$

The matrix V is called the **sampling matrix**. That is, the sample locations are vectors $\hat{t}$ that are linear combinations of the columns of the sampling matrix V. The set of all sample points $\hat{t} = V\hat{n}$, $\hat{n} \in \aleph$, where $\aleph$ space of all integer vectors (of the appropriate dimension), is called the **lattice** generated by V, and is denoted $LAT(V)$. The set of points $V\hat{x}$ where $\hat{x} = [x_1, x_2]^T$, with $0 \le x_1, x_2 < 1$, is called the **fundamental parallelepiped** of V and is denoted $FPD(V)$ [9]. The set of integer points within $FPD(V)$ is denoted by the set $N(V)$. If V is an integer matrix, then the number of elements (called the **integer volume**) in $N(V)$ is given by $|N(V)| = |det(V)|$. Suppose that $\hat{n}$ is a point on $LAT(V)$. Then there exists an integer vector $\hat{k}$ such that $\hat{n} = V\hat{k}$. The points $\hat{k}$ are called the **renumbered points** of $LAT(V)$.

2.1 Multidimensional Decimator and Expander

The two basic multirate operators for multidimensional systems are the decimator and expander. For an MD signal $x(\hat{n})$ on $LAT(V_I)$, the M-fold decimated version is given by $y(\hat{n}) = x(\hat{n}), \hat{n} \in LAT(V_I M)$ where M is an $m \times m$ non-singular integer matrix, called the **decimation matrix**. Note that $LAT(V_I) \supseteq LAT(V_I M)$. The **decimation ratio** for a decimator with decimation matrix M is defined to be the number of points thrown away for every point kept from the input and is given by $|N(M)| = |det(M)|$.

The "expanded" output $y(\hat{n})$ of an input signal $x(n)$ is given by:

$$y(n) = \begin{pmatrix} x(n) & n \in LAT(V_I) \\ 0 & \text{otherwise} \end{pmatrix} \forall n \in LAT(V_I L^{-1})$$

where V_I is the input lattice to the expander. Note that $LAT(V_I) \subseteq LAT(V_I L^{-1})$. The expansion ratio, defined as the number of points added to the output lattice for each point in the input lattice, is given by $|det(L)|$.

3. Multidimensional Synchronous Dataflow

A model suited for multidimensional systems is an extension of SDF called MDSDF where the arcs become m-dimensional arrays instead of FIFO queues. Data along only one of the m dimensions is allowed to be an infinite stream; this is because if the stream were infinite in more than one dimension, the computation might depend on the schedule for the system, leading to non-determinacy. The produced/consumed numbers for each node on each arc are now m-dimensional tuples. A set of **balance equations** (which dictates that the number of firings of nodes in the graph must result in the total number of samples produced on an arc to equal the total number consumed) can be written for each of the dimensions to get a **repetitions matrix** where each column (of length m) of the matrix represents the repetitions of the node along the different directions. Figure 1 makes these notions clearer. It represents a 2 dimensional MDSDF graph where node A produces an 2x1 array of samples on each firing, and node B consumes an 1x3 array of samples on each firing. The horizontal dimension is taken to be the direction along which the stream is infinite. The second diagram shows the underlying data space. The data space can be thought of as a two dimensional array that is infinite in the horizontal direction and of size 2 in the vertical direction. The first column is the data produced by A on its first invocation. The balance equations are given by $r_{A,1} \times 2 = r_{B,1} \times 1$, $r_{A,2} \times 1 = r_{B,2} \times 3$. This can be solved to yield $(r_{A,1}, r_{A,2}) = (1, 3)$ and $(r_{B,1}, r_{B,2}) = (2, 1)$. This means that A fires 3 times in the horizontal direction (produces 3 columns of data) and once in the vertical direction, and B fires once in the horizontal direction and twice in the vertical direction. The total number of samples exchanged on the arc is an array of 2x3 samples.

4. Semantics of the Generalized Model

MDSDF handles only rectangular data-spaces whereas a system that handles arbitrary lattices must be able to deal with non-rectangular data-spaces. In building such a model, several questions arise:

- One objective is to retain the producer/consumer model between actors for these streams; how do we determine the number of samples produced and consumed when the lattice is non-rectangular?
- How do the various actors know which lattice they are consuming and producing samples on?
- How do scanning orders and "next sample consumed" get determined?
- How do we model non-rectangular decimators and interpolators?
- Can we do compile time scheduling by solving "balance equations" as is done in MDSDF?

Consider the system depicted in figure 2, where a source actor produces an array of 6x6 samples each time it fires ((6,6) in MDSDF parlance). This actor is connected to the decimator with a non-diagonal decimation matrix. The circled samples indicate the samples that fall on the decimators output lattice; these are retained by the decimator. In order to represent these samples on the decimators output, we will think of the buffers on the arcs as containing the renumbered equivalent of the samples on a lattice. For a decimator, if we renumber the samples at the output according to $LAT(V_I M)$, then the samples get written to a parallelogram shaped array rather than a rectangular array. To see what this parallelogram is, we introduce the concept of a "support matrix" that describes precisely the region of the rectangular lattice where samples have been produced. Figure 2 illustrates

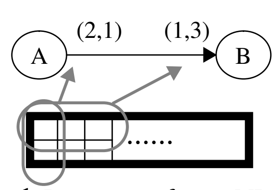

Figure 1. Data space for an MDSDF arc

Figure 3. Generalized expander and decimator with arbitrary input lattices and support matrices.

this for a decimation matrix, where the retained samples have been renumbered according to $LAT(M)$ and plotted on the right. The labels on the samples show the mapping. The renumbered samples can be viewed as the set of integer points lying inside the parallelogram that is shown in the figure. In other words, the **support** of the renumbered samples can be described as $FPD(Q)$ where

$$Q = \begin{bmatrix} 3 & 1.5 \\ 3 & 1.5 \end{bmatrix}$$

We will call Q the **support matrix** for the samples on the output arc. In the same way, we can describe the support of the samples on the input arc to the decimator as $FPD(P)$ where

$$P = \begin{bmatrix} 6 & 0 \\ 0 & 6 \end{bmatrix}$$

It turns out that $Q = M^{-1}P$.

Definition 1: The **containability condition:** let X be a set of integer points in $\Re^m$. We say that X satisfies the *containability condition* if there exists an $m \times m$ rational-valued matrix W such that $N(W) = X$.

Definition 2: We will assume that any source actor in the system produces data in the following manner. A source S will produce a set of samples ζ on each firing such that each sample in ζ will lie on the lattice $LAT(V_S)$. Hence, the set $\bar{\zeta} = \{V_S^{-1}\hat{n} \colon \hat{n} \in \zeta\}$ is a set of integer points, consisting of the points of ζ renumbered by $LAT(V_S)$. We assume that the set $\bar{\zeta}$ satisfies the containability condition.

Given a decimator with decimation matrix M as shown in figure 3, we make the following definitions and statements.: Denoting the input arc as e and the output arc as f, V_e, V_f are the bases for the input and output lattice respectively. W_e, W_f are the support matrices for the input and output arcs respectively. With this notation, we can state the following theorem:

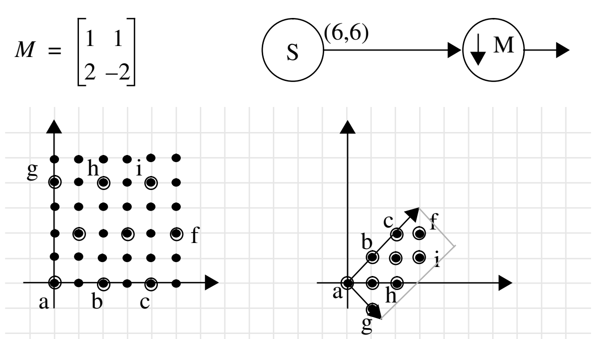

Figure 2. Output samples from the decimator renumbered to illustrate concept of support matrix.

Theorem 1: [6]The input/output relationships are given by:

Decimator: $V_f = V_e M$, $W_f = M^{-1}W_e$
Expander: $V_f = V_e L^{-1}$, $W_f = LW_e$.

Definition 3: Let X be a set of points in $\Re^2$, and x, y be two positive integers such that $xy = |X|$. X is said to be organized as a **generalized** (x, y) **rectangle** of points, or just a generalized (x, y) rectangle, by associating a **rectangularizing** function with X that maps the points of X to an integer (x, y) rectangle.

In summary, our generalized model for expressing non-rectangular systems has the following semantics:

- Each arc has associated with it a support matrix and a lattice generating matrix.
- An expander with expansion matrix L consumes (1,1) and produces the set of samples in $FPD(L)$ that is ordered as a generalized (L_1, L_2) rectangle of data where L_1, L_2 are positive integers such that $L_1 L_2 = |det(L)|$.
- A decimator with decimation matrix M consumes a rectangle (M_1, M_2) of data where this rectangle is interpreted according to the way it has been ordered (by the use of some rectangularizing function) by the actor feeding the decimator. It produces (1,1) on average.
- On any arc, the global ordering of the samples on that arc is established by the actor feeding the arc. The actor consuming the samples follows this ordering.
- A set of balance equations are written down using the various factorizations. Additional constraints for arcs that feed a decimator are also written down. These are solved to yield the repetitions matrix for the network. A scheduler can then construct a static schedule by firing firable nodes in the graph until each node has been fired the requisite number of times as given by the repetitions matrix.

For example, consider the system below, where a decimator follows an expander (figure 4(a))

We start by specifying the lattice and support matrix for the arc SA. Let $V_{SA} = I$, where I is the identity matrix, and $W_{SA} = 3I$. So the source produces (3,3) in MDSDF parlance. For the system above, we can compute the lattice and support matrices for all other arcs given these. The expander consumes the samples on arc SA in some natural order; for example, scanning by rows. The expander produces $FPD(L)$ samples on each

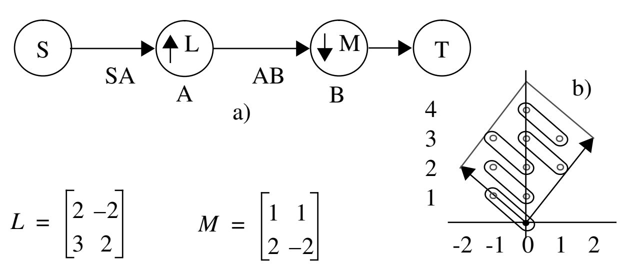

Figure 4. An example to illustrate balance equations and the need for some additional constraints. a) The system. b) Ordering of data into a 5x2 rectangle inside FPD(L).

firing; these samples are organized as a generalized (L_1, L_2) rectangle.

Suppose we choose the factorization 5x2 for $|det(L)|$. One way to map the samples into an integer $(5, 2)$ rectangle is as shown by the groupings in figure 4(b). Notice that the horizontal direction for $FPD(L)$ is the direction of the vector $[2, 3]^T$ and the vertical direction is the direction of the vector $[-2, 2]^T$. A global ordering on the samples can be deduced from these directions. For the decimator, a factorization of $|det(M)|$ is chosen, and a "rectangle" of those samples, where the "rectangle" is deduced from the global ordering imposed above is consumed per firing (figure 5).

Now we can write down a set of "balance" equations using the "rectangles" that we have defined. Denote the repetitions of a node X in the "horizontal" direction by $r_{X,1}$ and the "vertical" direction as $r_{X,2}$. We have

$$3r_{S,1} = 1r_{A,1},\ 5r_{A,1} = 2r_{B,1},\ r_{B,1} = r_{T,1}$$
$$3r_{S,2} = 1r_{A,2},\ 2r_{A,2} = 2r_{B,2},\ r_{B,2} = r_{T,2} \qquad \textbf{(EQ 1)}$$

where we have assumed that the sink actor T consumes (1,1) for simplicity. We have also made the assumption that the decimator produces exactly (1,1) every time it fires. This assumption is usually invalid as is shown in figure 5. Hence, we need to augment the balance equations with an additional constraint: $|N(W_{BT})| = |N(W_{AB})|/|M|$. We can symbolically derive expressions for W_{BT}, W_{AB} using $r_{S,1}, r_{S,2}$ [6]:

$$W_{AB} = \begin{bmatrix} 6r_{S,1} & -6r_{S,2} \\ 9r_{S,1} & 6r_{S,2} \end{bmatrix}, \text{ and } W_{BT} = \frac{1}{4}\begin{bmatrix} 21r_{S,1} & -6r_{S,2} \\ 3r_{S,1} & -18r_{S,2} \end{bmatrix}$$

The solution to 1 gives $r_{S,1} = 2, r_{S,2} = 1$. With these values, we get

$$W_{BT} = \begin{bmatrix} 21/2 & -3/2 \\ 3/2 & -9/2 \end{bmatrix}.$$

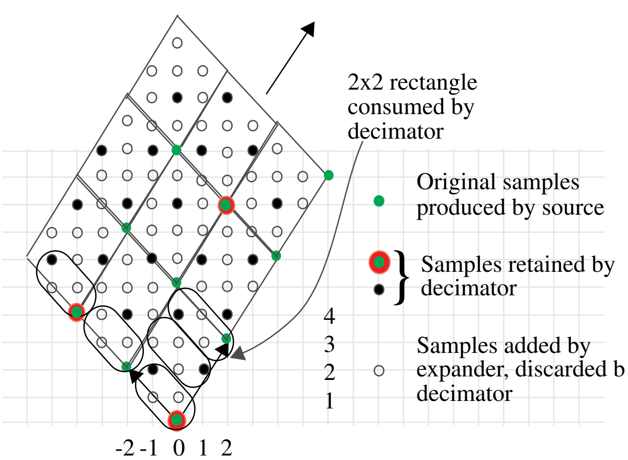

Figure 5. Some of the data produced by the source in one iteration of the periodic schedule determined by the balance equations in equation 1.

Since this matrix is not integer-valued, it appears that its integer volume has to be computed by brute force; it turns out to be 47. Hence, $|N(W_{BT})| = |N(W_{AB})|/|M|$ is not satisfied. One way to satisfy it is to force W_{BT} to be an integer matrix.; the smallest values that do it are $r_{S,1} = 4, r_{S,2} = 2$. It is shown in [6] that it is always possible to solve these *augmented* balance equations.

5. Future Work

The different choices of factorizations leads to different sets of balance equations, and these in turn will lead to different schedules. One optimization problem that arises is to choose these factorizations in such a way that the schedules are as small as possible. Some of these issues are discussed in more detail in [6], although general solutions have not been obtained yet.

Parallel scheduling issues have not been dealt with yet. In order to effectively exploit data parallelism, techniques developed in the systolic arrays community [10], and the loop parallelization community [11] might prove to be useful.

6. References

[1] Buck, S. Ha, E. A. Lee, D. G. Messerschmitt, "Ptolemy: a Framework for Simulating and Prototyping Heterogeneous Systems", International Journal of Computer Simulation, Jan., 1994.

[2] M. C. Chen, "Developing a Multidimensional Synchronous Dataflow Domain in Ptolemy", MS Report, UC-Berkeley, UCB/ERL Memo No. M94/16, June 1994

[3] K. Konstantinides, J. R. Rasure, "The Khoros software development environment for image and signal processing", IEEE Transactions on Image Processing, May 1994.

[4] E. A. Lee, D. G Messerschmitt, "Static Scheduling of Synchronous Dataflow Programs for Digital Signal Processing," IEEE Trans. on Computers, Jan. 1987

[5] E. A. Lee, "Multidimensional Streams Rooted in Dataflow", Proceedings of the IFIP Working Conference on Architectures and Compilation Techniques for Fine and Medium Grain Parallelism, Jan. 20-22, 1993, Orlando, FL

[6] P. K. Murthy, E. A. Lee, "A Generalization of Multidimensional Synchronous Dataflow to Arbitrary Sampling Lattices", Technical report, UCB/ERL M95/59, Electronics Research Laboratory, UC Berkeley, Ca 94720, Mar. 1995

[7] P. K. Murthy, S. S. Bhattacharyya, E. A. Lee, "Combined Code and Data Minimization for Synchronous Dataflow Programs", ERL Memo No. UCB/ERL M94/93, Electronics Research Lab, UC Berkeley, CA 94720, Nov. 1994

[8] G. C. Sih and E. A. Lee, "Declustering: A New Multiprocessor Scheduling Technique," IEEE Trans. on Parallel and Distributed Systems, June 1993.

[9] P. P. Vaidyanathan, *Multirate Systems and Filter Banks*, Prentice Hall, 1993

[10] S. Y. Kung, *VLSI Array Processors*, Prentice Hall, 1988

[11] H.Zima and B.Chapman, *Supercompilers for Parallel and Vector Computers*, ACM Press, 1990.