[19] 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, and submitted to the Journal on Formal Methods in System Design, Nov. 1994

[20] G. L. Nemhauser, L. A. Woolsey, *Integer and combinatorial optimization*, Wiley 1988

[21] T. A. Ramstad, “Digital Methods for Conversion between Arbitrary Sampling Frequencies”, IEEE Transaction on Acoustics, Speech, and Signal Processing, vol. ASSP-32, pp.571-591, June 1984

[22] P. P. Vaidyanathan, “Fundamentals of Multidimensional Multirate Digital Signal Processing,” *Sadhana*, vol. 15, pp. 157-176, Nov. 1990

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

[24] E. Viscito, J. P. Allebach, “The analysis and Design of Multidimensional FIR Perfect Reconstruction Filterbanks for Arbitrary Sampling Lattices”, IEEE Transactions on Circuits and Systems, vol. CAS-38, pp. 29-41, Jan 1991

8 References

[1] R. Ansari, S. H. Lee, "Two Dimensional Nonrectangular Interpolation, Decimation Filterbanks", Proc. of the ICASSP, New York 1988

[2] R. H. Bamberger, "The Directional Filterbank: a Multirate Filterbank for the Directional Decomposition of Images", Ph.D. Thesis, Georgia Institute of Technology, Nov. 1990

[3] G. Bilsen, M. Engels, R. Lauwereins, J. Peperstraete, "Static Scheduling of Multi-rate Cyclo-static DSP applications," IEEE workshop on VLSI Signal Processing, San Diego October 1994

[4] S. S. Bhattacharyya, P. K. Murthy, E. A. Lee, "APGAN and RPMC: Complementary Heuristics for Translating DSP Block Diagrams into Efficient Software Implementations," UCB/ERL Tech. Memo. M95/3, and submitted to the IEEE Transactions on Signal Processing, January 10, 1995

[5] F. Bosveld, R. L. Lagendijk, J. Biemond, "Compatible Spatio-Temporal Subband Encoding of HDTV", Signal Processing, vol. 28, (no. 3):271-289, Sept. 1992

[6] Buck, S. Ha, E. A. Lee, D. G. Messerschmitt, "Ptolemy: a Framework for Simulating and Prototyping Heterogeneous Systems", *International Journal of Computer Simulation*, special issue on "Simulation Software Development," January, 1994.

[7] M. C. Chen, "Developing a Multidimensional Synchronous Dataflow Domain in Ptolemy", MS Report, UC Berkeley, June 1994

[8] E. Dubois, "The Sampling and Reconstruction of Time-varying Imagery with Applications in Video Systems", Proceedings of the IEEE, vol. 73, pp. 502-522, April 1985

[9] D. E. Dudgeon, R. M. Mersereau, *Multidimensional Digital Signal Processing*, Prentice Hall, 1984

[10] B. L Evans, "Knowledge-Based Environment for the Design and Analysis of Multidimensional Multirate Signal Processing Algorithms", Ph.D. Thesis, Georgia Institute of Technology, September 1993

[11] R. Hopkins, "Progress on HDTV Broadcasting Standards in the United States", Signal Processing: *Image Communication*, vol. 5, 355-378, December 1993

[12] G. Karlsson, M. Vetterli, "Theory of Two Dimensional Multirate Filter Banks", IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. ASSP-38, pp. 925-937, June 1990

[13] K. Konstantinides, J. R. Rasure, "The Khoros software development environment for image and signal processing", IEEE Transactions on Image Processing, May 1994, vol.3, (no.3):243-52

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

[15] 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

[16] R. Manduchi, G. M. Cortelazzo, and G. A. Mian, "Multistage Sampling Structure Conversion of Video Signals", IEEE Transactions on Circuits and Systems for Video Technology, Vol. 3, No. 5, October 1993

[17] R. M. Mersereau, T. C. Speake, "The Processing of periodically sampled Multidimensional Signals", IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. ASSP-31, pp. 188-194, Feb 1983

[18] P. K. Murthy, S. S. Bhattacharyya, E. A. Lee, "Minimizing Memory Requirements for Chain Structured Synchronous Dataflow Graphs", Proceedings of the ICASSP, Adelaide, Australia, April 1994

repetitions of a node is also generalized to depend on the lattice and support matrix on outgoing and incoming arcs; that is, the direction can be different for an input arc from an output arc.

There are many open problems and issues that have not been tackled yet. Some of these are listed below:

•The issue of buffering efficiency and buffer implementations has not been addressed. It would be desirable to have systematic ways of determining schedules that minimize for code size and buffer-usage, as was done in the SDF case for well-ordered graphs [18] and general acyclic graphs [19][4]. The techniques in [18][19] can be easily applied to the rectangular MDSDF case since rectangular MDSDF can be thought of as several independent SDF graphs (one for each dimension); with non-rectangular systems, it is less obvious how to extend the techniques of [18][19]. The buffers would probably have to be implemented directly as linear arrays but indexed appropriately (at least for simulation) since techniques like using matrix and submatrix data-structures [7] may not be feasible for non-rectangular support matrices.

•An extension of lemma 6 to arbitrary support regions would be desirable. This would give us a way of choosing factorizations that do not have the cyclostatic behavior that an arbitrary factorization generally does. This would also ensure that solving balance equations is sufficient and constraints of the type in equation 20 do not have to be solved. However, this appears to be difficult since there does not seem to be a clean, analytical way of expressing the ordering of samples as done in table 1. Since the understanding of how points on $LAT(M)$, where M is the decimation matrix, map to the rectangle under this ordering is needed before a claim of the sort made in lemma 6, this extension would be non-trivial.

•If methods of choosing factorizations remain ad-hoc, then an efficient way is required to determine the number of integer points in the fundamental parallelepiped of an arbitrary matrix. Hopefully, an analytic expression for this number exists, or else an efficient algorithm would be desirable. Counting the number of points in a brute-force manner quickly becomes inefficient if the support matrix has large entries.

•More complicated examples. The examples presented in this report have been those of simple, chain-structured graphs. Concrete examples of acyclic graphs that represent non-rectangular systems with a lot of inherent functional parallelism include directional decomposition filterbanks as described in [2], and hierarchical video coding applications [5].

•Examination of higher dimensions. The examples in this report have all been for 2-D MDSDF; generalizations to higher dimensions may be trickier. A concrete example of a 3 dimensional digital signal is a fully scanned TV signal (where the horizontal direction is scanned also).

•There are many other ways of doing the sampling-structure conversion [16]. If more decimation stages are used, more of the constraints of the type in equation 20 have to be solved. Some of these ways of doing the conversion might be better in computational terms than other ways in that the repetitions of the various actors in between are lower for some ways than others. It would be interesting to explore systematic ways of evaluating the various ways.

$$\begin{aligned} r_{S,1} &= 3, r_{S,2} = 1 \\ r_{A,1} &= 6, r_{A,2} = 8 \\ r_{B,1} &= 12, r_{B,2} = 16 \\ r_{C,1} &= 4, r_{C,2} = 16 \end{aligned} \quad \textbf{(EQ 19)}$$

Again, we can calculate the various support matrices symbolically in order to verify whether

$$|N(W_{CT})| = |N(W_{BC})| / |det(M)| \quad \textbf{(EQ 20)}$$

We have $|N(W_{BC})| = det\left(\begin{bmatrix} 2r_{S,1} & 0 \\ 0 & 32r_{S,2} \end{bmatrix}\right) = 64r_{S,1}r_{S,2}$, and

$$W_{CT} = M^{-1}W_{BC} = \begin{bmatrix} r_{S,1} & -(16/3)\,r_{S,2} \\ 0 & (32/3)\,r_{S,2} \end{bmatrix}$$

With the values obtained from the balance equations, equation 20 turns out to be not satisfied because $|N(W_{CT})| = 33$! In actuality, since there are 625 lines vertically, a more accurate model for the source is that it produces samples in (2,625), again according to the rectangle defined above. However, the resulting matrix

$$W_{CT} = \begin{bmatrix} r_{S,1} & -(1250/3)\,r_{S,2} \\ 0 & (2500/3)\,r_{S,2} \end{bmatrix}$$

presents a rather nasty challenge for determining the number of integer points inside its FPD (since it is not an integer matrix with the solution to the balance equations). However, in the model where the source produces (2,8), we can force W_{CT} to be an integer matrix by making $r_{S,2}$ a multiple of 3. If the source is in fact trying to produce 625 lines vertically, then the smallest multiple of 3 that is greater than or equal to 625 is given by 209; we can make this the vertical blocking factor. As in the previous example, we can try to find better models for the source, one that will allow it to cover exactly 625 lines vertically for example. Of-course, this is assuming that the source actor can be coded in a flexible way that allows it to produce as much data or as little as desired.

7 Conclusions and open problems

We have described an extension of MDSDF to handle arbitrary sampling lattices. The key extensions have been to associate two parameters for each arc in the graph: the sampling lattice for the data on that arc, and a “support matrix” that describes the region of the space where current data has been produced. Equivalently, these two parameters can be specified for source actors, and can be determined for the other arcs by tracing the operations that each node in the graph performs. Since decimation and expansion are the only two actors (of signal processing interest) that change lattices, it should be straightforward to determine the lattice and support matrix for every arc in the graph given the information at the source (assuming that there are no cycles in the graph). Given these two parameters, we have shown how we can compute a global ordering for the data on the arcs, and how generalized “rectangles” can be defined. Using these, we can derive a set of decoupled balance equations in an analogous manner to the rectangular case. However, this is not enough; for decimators, some other constraints must also be satisfied. The directions of the

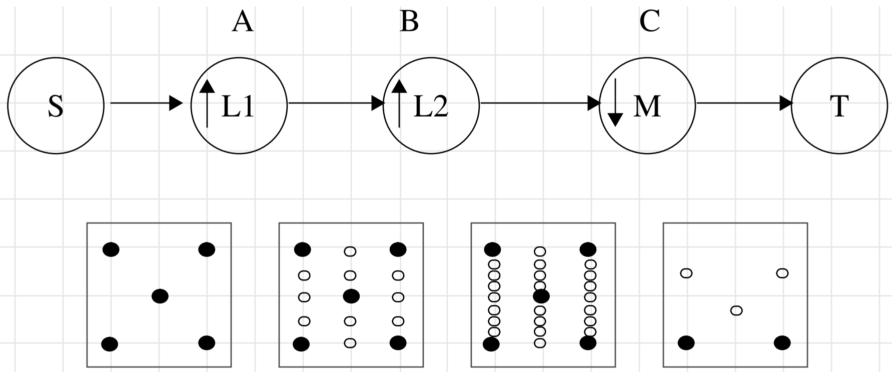


Fig 23. System for doing multistage sampling structure conversion from 4/3 aspect ratio to 16/9 aspect ratio for a 2:1 interlaced TV signal.

$$L_1 = V_{AB}^{-1} V_{SA} = \begin{bmatrix} 1/T_f & 0 \\ 0 & 2/d_y \end{bmatrix} \begin{bmatrix} 2T_f & T_f \\ 0 & d_y \end{bmatrix} = \begin{bmatrix} 2 & 1 \\ 0 & 2 \end{bmatrix}, \; L_2 = V_{BC}^{-1} V_{AB} = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}, \text{ and}$$

$M = V_{BC}^{-1} V_{CT} = \begin{bmatrix} 2 & 1 \\ 0 & 3 \end{bmatrix}$. Note that these matrices do not depend on T_f and d_y.

Suppose

$$|det(L_1)| = 4 = 2 \times 2, \; |det(L_2)| = 2 = 1 \times 2, \text{ and } |det(M)| = 6 = 3 \times 2$$

Let us assume that S produces samples in $LAT(V_{SA}) \cap N\left(\begin{bmatrix} 2 & 0 \\ 0 & 8 \end{bmatrix}\right)$. The support matrix for this can be computed as

$$V_{SA}^{-1} \begin{bmatrix} 2 & 0 \\ 0 & 8 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 1 & -1 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} 2 & 0 \\ 0 & 8 \end{bmatrix} = \begin{bmatrix} 1 & -4 \\ 0 & 8 \end{bmatrix}$$

Suppose further that the samples that the source produces are rectangularized in the following obvious way: sample at (0,0) is (0,0); sample at (1,1) is (1,0); sample at (0,2) is (0,1); sample at (1,3) is (1,1) etc. So S produces (2,8) where the rectangle is understood to be as above. We can write down the following balance equations:

$$\begin{aligned} 2r_{S,1} &= 1r_{A,1} \\ 8r_{S,2} &= 1r_{A,2} \\ 2r_{A,1} &= 1r_{B,1} \\ 2r_{A,2} &= 1r_{B,2} \\ 1r_{B,1} &= 3r_{C,1} \\ 2r_{B,2} &= 2r_{C,2} \\ r_{C,1} &= r_{T,1} \\ r_{C,2} &= r_{T,2} \end{aligned} \qquad \textbf{(EQ 18)}$$

These solve to

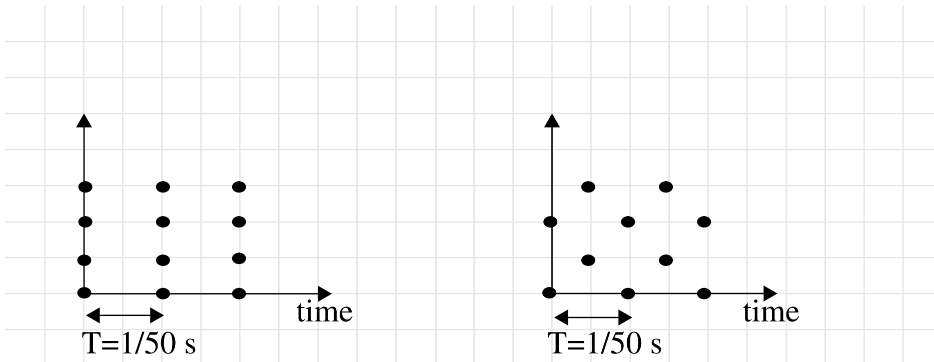


Fig 21. Progressive scanning and 2:1 interlaced scanning in the vertico-temporal plane

pling rate and sampling structure conversion in stages for multidimensional systems. The two aspect ratios and the two lattices are shown in figure 22.

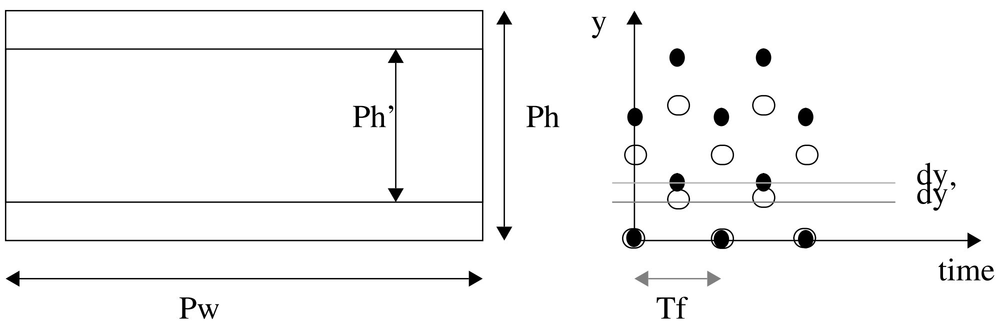


Fig 22. Picture sizes and lattices for the two aspect ratios 4/3 and 16/9

The relationship between the height and width for the two aspect ratios is given by $P_h = (3/4)P_w$ and $P_h' = (9/16)P_w$. Since the number of lines N is the same in both cases, the interline distances d_y' and d_y, corresponding to the 16/9 and 4/3 aspect ratios respectively, are related as $d_y' = P_h'/N = (3/4)\,(P_h/N) = (3/4)d_y$. Thus the bases for the lattices for the two aspect ratios are given by

$$A = \begin{bmatrix} 2T_f & T_f \\ 0 & d_y \end{bmatrix} \quad \text{and} \quad A' = \begin{bmatrix} 2T_f & T_f \\ 0 & (3/4)d_y \end{bmatrix} \qquad \textbf{(EQ 17)}$$

for the 4/3 and 16/9 aspect ratios respectively. By the CCIR System M standard 625/2:1/50, $T_f = 1/50\ s$ and $N = 625$. One way to do the conversion between the two lattices above is as shown below in figure 23 [16]. Below each arc, the desired lattice on that arc is shown. For simplicity, the filters that are needed between the upsampling and downsampling stages are omitted. From the lattices shown, we can compute the required values for L_1, L_2, M:

$$V_{SA} = A = \begin{bmatrix} 2T_f & T_f \\ 0 & d_y \end{bmatrix},\ V_{AB} = \begin{bmatrix} T_f & 0 \\ 0 & d_y/2 \end{bmatrix},\ V_{BC} = \begin{bmatrix} T_f & 0 \\ 0 & d_y/4 \end{bmatrix},\ \text{and}\ V_{CT} = A' = \begin{bmatrix} 2T_f & T_f \\ 0 & 3d_y/4 \end{bmatrix}$$

Since using realistic values for T_f, d_y will make all the calculations rather ugly, we will just use $T_f = 1, d_y = 1$ without any loss in generality.

By theorem 1,

•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. Unfortunately, there does not seem to be any way of making the decimators output any more concrete.

•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.

6 Multistage sampling structure conversion example

In this section, we illustrate another example; this is a practical example of a system that does sampling structure conversion for video signals. We will show how the semantics developed above can be used to specify and determine a schedule for the system. This example is drawn from [16].

6.1 Video Signals

A video signal can be thought of as a three-dimensional signal where two of the dimensions correspond to the height(vertical) and width(horizontal) of the image while the third dimension is time. The current practice is to sample the signal in two of the dimensions and keep the third dimension continuous. The vertical and temporal directions are sampled (this processed is called scanning) while the horizontal direction is not. Hence, as a discrete-time signal, a video signal is two-dimensional, with samples occupying the *vertico-temporal* plane. Note that an actual video signal is one-dimensional since the resulting lines (from the scanning) are abutted to form a one-dimensional signal [8]. There are currently two types of vertico-temporal lattices in use: the progressively scanned signal, which corresponds to a rectangular lattice, and the 2:1 interlaced signal, which corresponds to a “quincunx” lattice. There are trade-offs between using these two lattices in various types of distortions and interline flicker but in general, 2:1 interlaced scanning is preferable to sequential scanning for a given scanning density [8][1]. The two lattices are shown in figure 21.

6.2 Sampling structure conversion

An application of considerable interest in current television practice is the format conversion from 4/3 aspect ratio to 16/9 aspect ratio for 2:1 interlaced TV signals. It is well known in one-dimensional signal processing theory that sample rate conversion can be done efficiently in many stages [21]. Similarly, it is more efficient to do both sam-

1. The Advisory Committee on Advanced Television Service, a committee formed by the FCC in 1987 to assist the FCC in establishing an HDTV standard for the United States, tested 4 all-digital HDTV proposals (DigiCipher, DSC-HDTV, AD-HDTV, CCDC) and found that systems using interlaced scanning (DigiCipher and AD-HDTV) had the best quality overall.[11].

5.2 Delays in the generalized model

Delays can be interpreted as translations of the buffer of produced values along the vectors of the support matrix (in the renumbered data space) or along the vectors in the basis for the sampling lattice (in the lattice data space). Figure 20 illustrates a delay of (1,2) on a non-rectangular lattice.

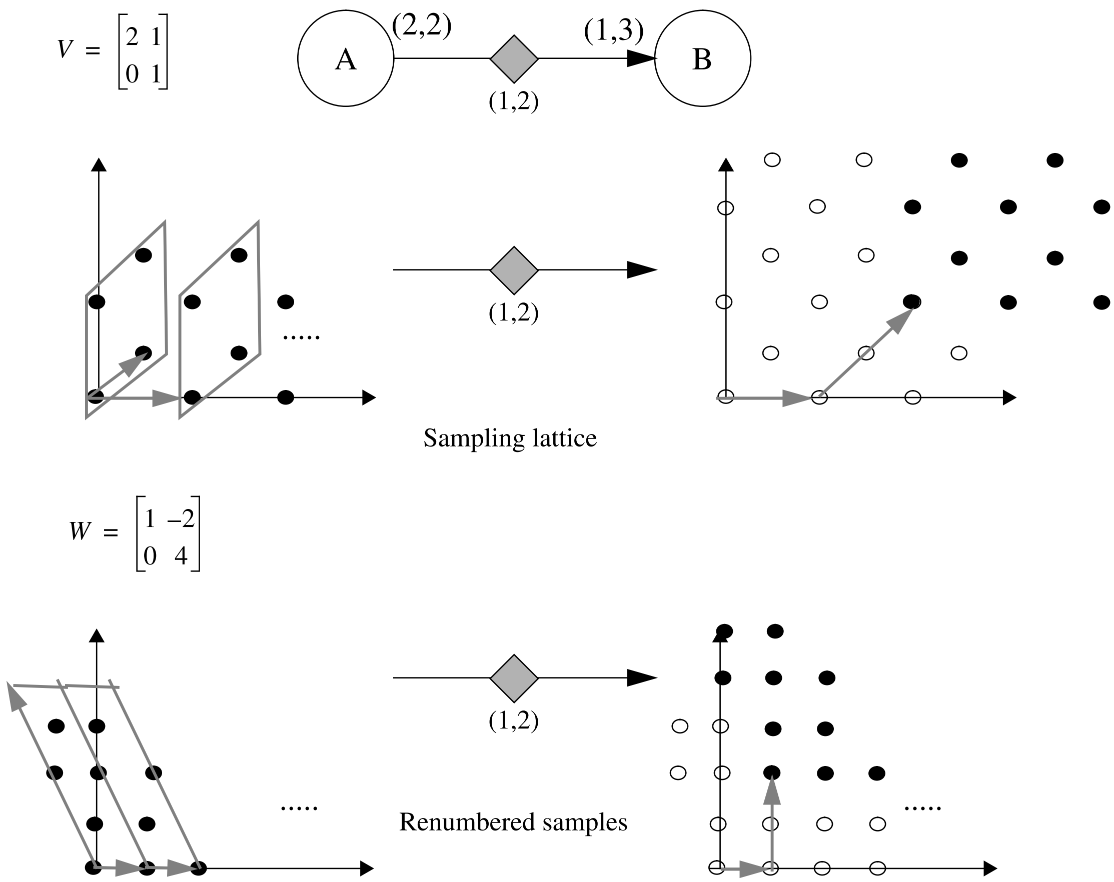


Fig 20. Delays on non-rectangular lattices

5.3 Summary of generalized model

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

•Sources produce data in accordance with definition 4. The support matrix and lattice-generating matrix on the sources output arcs are specified by the source. The source produces a generalized (S_1, S_2) rectangle of data on each firing.

•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 $L_1 L_2 = |det(L)|$.

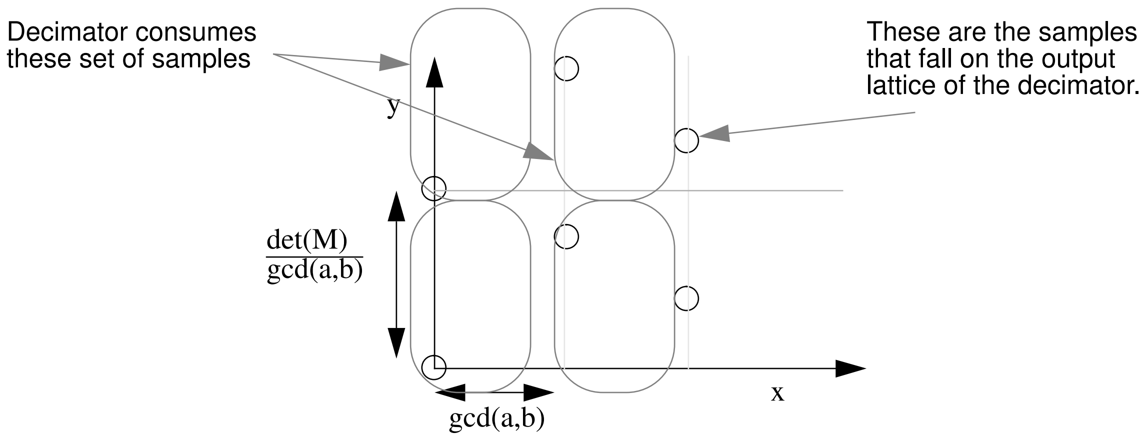


Fig 19. Figure to illustrate proof of lemma 6.

$$k_1' = \frac{dx_0 - b(y_0 + j)}{ad - bc} \text{ and } k_2' = \frac{a(y_0 + j) - cx_0}{ad - bc}$$

also integers. Rearranging the above expressions, we get

$$k_1' = k_1 - \frac{bj}{ad - bc} \text{ and } k_2' = k_2 + \frac{aj}{ad - bc}$$

Clearly, $j = |det(M)|/gcd(a, b)$ makes both k_1' and k_2' integers. It is also the smallest: let $m = ad - bc$. Then, we have that $bj/m = i_1$ for some integer i_1 and $aj/m = i_2$ for some integer i_2. Hence, $i_1 a = i_2 b$ giving us the claimed value of j as the smallest such value.

A symmetric argument shows that if y is fixed, then the values of x that solve equation 16 differ by $|det(M)|/gcd(c, d)$.

Without loss in generality, consider the rectangle of dimensionality given by the first of the factorings in the statement of the lemma. When this rectangle is placed at the origin, only the sample at the origin falls inside it and is output by the decimator (see figure 19). Invocation (0,1) of the decimator would consume the rectangle whose lower left corner is at $y = |det(M)|/gcd(a, b)$, $x = 0$; this also contains only one sample that is output by the decimator (namely, the lower left corner sample). Clearly, as the rectangle is moved up along the y-axis by steps of $|det(M)|/gcd(a, b)$, the sample kept is always the one on the lower left corner. Now consider moving the rectangle to the right in steps of $gcd(a, b)$. None of these rectangles can contain two samples which do not have the same x coordinate. This is because the x coordinate in the solution equation 16 has to be a multiple of $gcd(a, b)$. These rectangles cannot contain two samples which have the same x coordinate either because, as was shown, when x is fixed, the y values differ by $|det(M)|/gcd(a, b)$. Hence, all of the rectangles that the decimator consumes will contain exactly one sample that falls on the output lattice, allowing the decimator to be non-cyclostatic. **QED.**

For the example matrix in the figure 18, we can see that the two factorizations that work from the above equation are $A_1 = 1, B_1 = 4$ and $A_2 = 2, B_2 = 2$; this is what we see from figure 18 (b),(c).

for any matrix, there are always two factorizations of the determinant such that the decimator produces (1,1) for all invocations. Also, this example illustrates two other points: it is only *sufficient* that the decimator produce 1 sample on each invocation for the additional constraints on decimator outputs to be satisfied by the balance equation solution. It is only *sufficient* that the support matrix on the decimators output be integer valued for the additional constraints to be satisfied. Indeed, we have

$$W_{SM} = \begin{bmatrix} 6r_{S,1} & 0 \\ 0 & 6r_{S,2} \end{bmatrix},\ W_{MO} = \begin{bmatrix} 3r_{S,1} & 1.5r_{S,2} \\ 3r_{S,1} & -1.5r_{S,2} \end{bmatrix},$$

where W_{MO} is the support matrix on the decimators output. For the case where $M1 = 4, M2 = 1$, we have $r_{S,1} = 2, r_{S,2} = 1$, making W_{MO} non-integer valued. However, we do have that $|N(W_{MO})| = |N(W_{SM})|/|det(M)|$, despite the fact that W_{MO} is non-integer valued and the decimator is cyclo-static.

Lemma 6: If

$$M = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

is any non-singular, integer 2x2 matrix, then there are at most two factorizations (and at least one) of $|det(M)|$, $A_1B_1 = |det(M)|$ and $A_2B_2 = |det(M)|$ such that if $M_1 = A_1, M_2 = B_1$ or $M_1 = A_2, M_2 = B_2$ in figure 18, then the decimator produces (1,1) for all invocations. Moreover,

$$A_1 = gcd(a,b),\ B_1 = \frac{|det(M)|}{gcd(a,b)},\ \text{and}\ A_2 = \frac{|det(M)|}{gcd(c,d)},\ B_2 = gcd(c,d)$$

Remark: Note that $gcd(a,0) = a$; hence, if M is diagonal, the two factorizations are the same and there is only one unique factorization. This implies that for rectangular decimation, there is only one way to set the MDSDF parameters, which is comforting.

Proof: Firstly, note that since $det(M) = ad - bc$, $gcd(a,b)$ divides $det(M)$ and $gcd(c,d)$ divides $det(M)$. The decimator keeps samples with coordinates given by

$$M\begin{bmatrix} k_1 \\ k_2 \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} \qquad \textbf{(EQ 16)}$$

or $ak_1 + bk_2 = x$ and $ck_1 + dk_2 = y$; hence, x and y have to be multiples of $gcd(a,b)$ and $gcd(c,d)$ respectively. Suppose $y = 0$. Then, with a little algebra, it can be seen that the smallest positive, non-zero value of x that solves equation 16 (meaning that k_1, k_2 are integers) is given by $x = |det(M)|/gcd(c,d)$. Similarly, if $x = 0$, the smallest non-zero, positive value of y is given by $y = |det(M)|/gcd(a,b)$. Hence, any rectangle consumed by the decimator cannot have a vertical dimension of greater than $gcd(c,d)$ or a horizontal dimension of greater than $gcd(a,b)$ since if it did, then at least two samples that are kept by the decimator will fall inside the rectangle based at the origin. So it remains to show that the two factorizations given in the statement of the lemma do in fact result in one sample being kept on *all* invocations. Fix some value for x that is a multiple of $gcd(a,b)$; call it x_0. Let y_0 be the smallest positive integer solution to equation 16. Then, the next positive integer y that solves equation 16 is given by $y_0 + |det(M)|/gcd(a,b)$. To see this, note that $k_1 = (dx_0 - by_0)/(ad - bc)$ and $k_2 = (ay_0 - cx_0)/(ad - bc)$ are both integers. We want to determine the smallest positive constant j such that makes

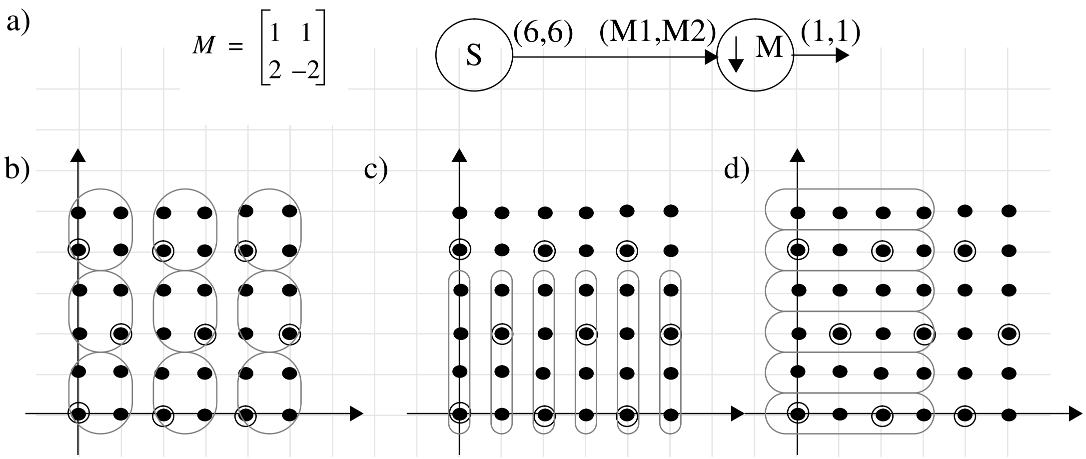


Fig 18. An example to illustrate that two factorizations always exist that result in non-cyclostatic behavior with the decimator. a) The system. b) M1=2, M2=2. c) M1=1, M2=4. d) M1=4, M2=1

$$W_{BT} = M^{-1}W_{AB} = \frac{1}{4}\begin{bmatrix} 2 & 1 \\ 2 & -1 \end{bmatrix}\begin{bmatrix} 6r_{S,1} & -8r_{S,2} \\ 9r_{S,1} & 8r_{S,2} \end{bmatrix} = \frac{1}{4}\begin{bmatrix} 21r_{S,1} & -8r_{S,2} \\ 3r_{S,1} & -24r_{S,2} \end{bmatrix} = \begin{bmatrix} 21/4 & -2 \\ 3/4 & -6 \end{bmatrix}$$

and we can determine that $|N(W_{BT})| = 30$ as required (i.e., $3 \times 4 \times 10/4 = 30$). Hence, we get away with having to produce only one extra row rather than three, assuming that the source can only produce 3 meaningful rows of data (and any number of columns).

5.1 Eliminating cyclostatic behavior

The fact that the decimator does not behave in a cyclostatic manner in figure 17 raises the question of whether factorizations that result in non-cyclostatic behavior in the decimator can always be found. The following example and lemma give an answer to this question for the special case of a decimator whose input is a rectangular lattice.

Example 2: Consider the system in figure 18 where a 2-D decimator is connected to a source actor that produces an array of (6,6) samples on each firing. The black dots represent the samples produced by the source and the circled black dots show the samples that the decimator should retain; these are the samples that lie on $LAT(M)$ intersected with the samples produced by the source. Since $|det(M)| = 4$, there are three possible ways to choose M_1, M_2. With $M_1 = M_2 = 2$ we see that the 6x6 array can be tiled with 2x2 arrays such that 1 sample is produced in each 2x2 array (figure 18 (b)). However, since some of the 2x2 blocks retain the sample on the left-bottom corner and some the sample on the right-bottom corner, a time-varying phase would have to be used to determine which sample should be output on a given invocation. There are two other ways to choose M_1, M_2 so that their product is 4: $M_1 = 1, M_2 = 4$ and $M_1 = 4, M_2 = 1$. Figures 18 (b),(c) illustrate the tiling with these choices. For both these choices, the repetitions of the source is different than (1,1); hence the tiling isn't complete but can be completed if the source produces more data as specified by the solution to the balance equations ((2,1) and (1,2) respectively). In figure 18 (c) also, we see that for every (1,4) consumed, (1,1) is produced, although again, a time-varying phase will be required. However, in figure 18(d), we see that on some invocations, *no* samples are produced (that is, (0,0) samples are produced) while in some invocations, 2 samples are produced. This raises the question of whether there is always a factorization that ensures that the decimator produces (1,1) for all invocations. The following lemma ensures that

$$\begin{aligned} 3r_{S,1} &= 1r_{A,1} \\ y_S 1 &= 1r_{A,2} \\ 5r_{A,1} &= 1r_{B,1} \\ 2r_{A,2} &= 4r_{B,2} \\ r_{B,1} &= r_{T,1} \\ r_{B,2} &= r_{T,2} \end{aligned} \qquad \textbf{(EQ 14)}$$

The solution to this is given by

$$\begin{aligned} r_{S,1} &= 1, y_S = 2k \qquad k = 1, 2, \ldots \\ r_{A,1} &= 3, r_{A,2} = 2k \\ r_{B,1} &= 15, r_{B,2} = k \\ r_{T,1} &= 15, r_{T,2} = 3 \end{aligned} \qquad \textbf{(EQ 15)}$$

and we see that $k = 2$ satisfies our constraint. Recalculating the other quantities,

$$W_{AB} = LW_{SA} = \begin{bmatrix} 2 & -2 \\ 3 & 2 \end{bmatrix} \begin{bmatrix} 3r_{S,1} & 0 \\ 0 & 4r_{S,2} \end{bmatrix} = \begin{bmatrix} 6r_{S,1} & -8r_{S,2} \\ 9r_{S,1} & 8r_{S,2} \end{bmatrix} \text{ and}$$

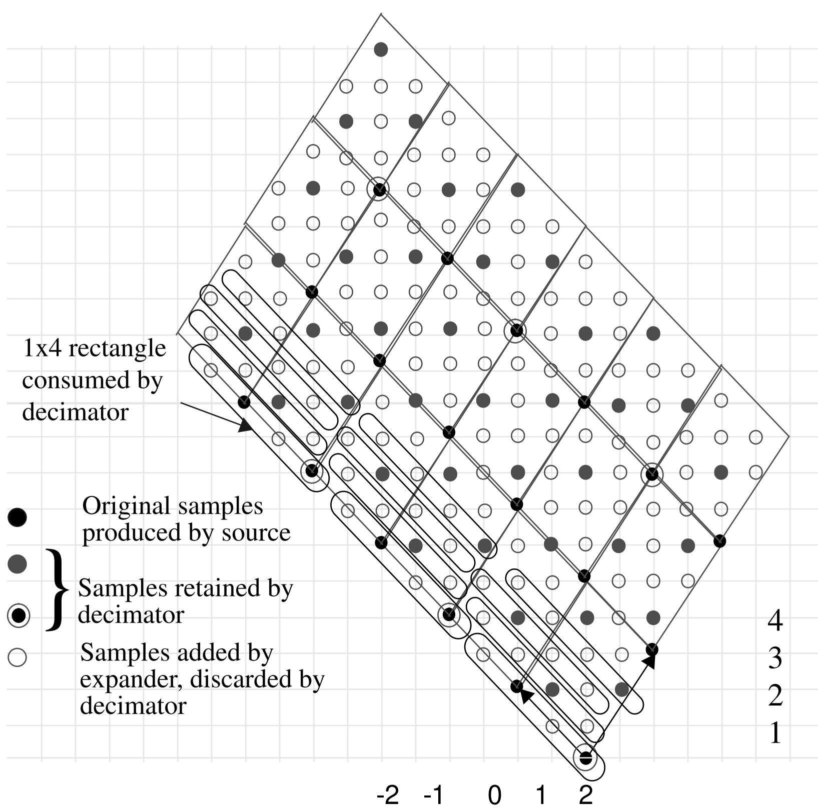


Fig 17. Total amount of data produced by the source in one iteration of the periodic schedule determined by the balance equations in equation 13. The samples that are kept by the decimator are the lightly shaded samples.

an iteration corresponds to the processing of an image at the next sampling instant. Only the repetitions of 3 along the time dimension makes physical sense. Hence, there is potentially room for great inefficiency if the user of the system has not made sure that the rates in the graph match up appropriately so that we do not actually end up generating images of size 1024x1024 when the actual image size is 512x512. In rectangular MDSDF, it might be reasonable to assume that the user is capable of setting the MDSDF parameters such that they do not result in absurd repetitions being generated in the non-stream directions since this can usually be done by inspection. However for non-rectangular systems, we would like to have more formal techniques for keeping the repetitions matrix in check since it is much less obvious how to do this by inspection than in the rectangular case. The number of variables are also greater for non-rectangular systems since different factorizations for the decimation or expansion matrices give different solutions for the balance equations.

To explore the different factoring choices, suppose we use 1x4 for the decimator instead of 2x2. The balance equations become

$$\begin{aligned} 3r_{S,1} &= 1r_{A,1} \\ 3r_{S,2} &= 1r_{A,2} \\ 5r_{A,1} &= 1r_{B,1} \\ 2r_{A,2} &= 4r_{B,2} \\ r_{B,1} &= r_{T,1} \\ r_{B,2} &= r_{T,2} \end{aligned} \qquad \textbf{(EQ 12)}$$

The solution to these is given by

$$\begin{aligned} r_{S,1} &= 1, r_{S,2} = 2 \\ r_{A,1} &= 3, r_{A,2} = 6 \\ r_{B,1} &= 15, r_{B,2} = 3 \\ r_{T,1} &= 15, r_{T,2} = 3 \end{aligned} \qquad \textbf{(EQ 13)}$$

From equation 9, W_{BT} is given by

$$W_{BT} = \begin{bmatrix} 21/4 & -3 \\ 3/4 & -9 \end{bmatrix}$$

Again using graph paper, it can be determined that $|N(W_{BT})| = 45$ as required. So in this case, we do not need to increase the blocking factor to make W_{BT} an integer matrix, and this is because the decimator is producing 1 token on every firing as shown in figure 17.

However, if the stream in the above direction were in the horizontal direction (from the point of view of the source), then the solution given by the balance equations (eq. 13) may not be satisfactory for reasons already mentioned. For example, the source may be forced to produce only zeros for invocation (0,1). One way to incorporate such constraints into the balance equations computation is to specify the repetitions vector instead of the number produced or consumed. That is, for the source, we specify that $r_{S,2} = 1$ but leave the number it produces in the vertical direction unspecified. The balance equations will give us a set of acceptable solutions involving the number produced vertically; we can then pick the smallest such number that is greater than or equal to three. Denoting the number produced vertically by y_S, our balance equations become

actor, like the decimation factor. In our model for the decimator, the number of phases is not just a function of the decimation matrix; it is also a function of the sampling lattice on the input to the decimator (which in turn depends on the actor that is feeding the arc), and the factorization choice that is made by the scheduler. Secondly, in CSDF, SDF actors are represented as cyclostatic by decomposing their input/output behavior over one invocation. For example, in figure 11(c) we saw that the CSDF decimator was behaving exactly like the SDF decimator except that the CSDF decimator does not need all M data inputs to be present before it fires. In our case, the cyclostatic behavior of the decimator is arising *across* invocations rather than within an invocation. It is as if the CSDF decimator with decimation factor 4 were to consume {4,4,4,4,4,4} and produce {2,0,1,1,0,2} instead of consuming {1,1,1,1} and producing {1,0,0,0}.

One way to avoid dealing with constraints of the type in equation 10 would be to choose a factorization of $|det(M)|$ that ensured that the decimator produced one sample on each invocation. For example, if we were to choose the factorization 1x4 for the example above, the solution to the balance equations would automatically satisfy equation 10. As we show later, we can find factorizations where the decimator produces one sample on every invocation in certain situations but generalizing this result appears to be a difficult open problem since there does not seem to be an analytical way of writing down the re-numbering transformation that was shown in table 1. Another way to avoid the constraint would be to figure out how many distinct phases (in a cyclostatic sense) the decimator has for any given factorization. From this, a set of cyclostatic balance equations could be written down whose solution would also automatically satisfy the constraint of equation 10. The technique we have given above is equally valid, with the only drawback being an inability to calculate efficiently the quantity $|N(W)|$ when W is not integer valued. If an efficient way can be found to do this calculation, then our method might be much more efficient than trying to determine the number of phases the decimator has and producing a set of “cyclostatic” balance equations. For example 1, with the decimator consuming a generalized (2,2) rectangle, the number of horizontal phases turns out to be 10 as determined by looking at the number of samples produced by the decimator for each consumption. This sequence of the number of samples produced for each (2,2) consumed turns out to be {2,1,1,2,1,0,1,1,0,1} in the “horizontal” direction (which is the direction given by $[2\ \ 3]^T$ in the renumbered samples data-space on the input). For the next “row” of samples (that is by going up to $[2\ \ -2]^T$ and then scanning in the $[2\ \ 3]^T$ direction), this sequence turns out to be {0,1,1,0,1,2,1,1,2,1} which is seen to be the sequence for the first row shifted by 5. The third row has the same sequence as the first row; hence these are the only two distinct sequences.

Implications of the above example for streams

Some remarks must be made about balance equations and streams. In SDF, there is only one dimension, and the stream is in that direction. Hence, whenever the repetitions of a node is greater than unity, then the data processed by that node corresponds to data along the stream. In MDSDF, only one of the directions is the stream. Hence, if the repetitions of a node, especially a source node, is greater than unity for the non-stream directions, the physical meaning of invocations in those directions becomes unclear. For example, consider a 3-dimensional MDSDF model for representing a progressively scanned video system. Of these 3 dimensions, 2 of the dimensions correspond to the height and width of the image, and the third dimension is time. Hence, a source actor that produces the video signal might produce something like (512,512,1) meaning 1 512x512 image per invocation. If the balance equations dictated that this source should fire (2,2,3) times, for example, then it is not clear what the 2 repetitions each in the height and width directions signify since they certainly do not result in data from the next iteration being processed, where

such that increasing the repetitions of each node in each dimension by the corresponding factor in J will result in the support matrices being integer valued for all arcs in the network.

Proof: By lemma 5, a term in an entry in the j^{th} column of the support matrix on any arc is always a product of a rational number and repetitions variable $r_{S,j}$ of source S. We force this term to be integer valued by dictating that each repetitions variable $r_{S,j}$ be the lcm of the values needed to force each entry in the j^{th} column to be an integer. Such a value can be computed for each support matrix in the network. The lcm of all these values and the balance equations solution for the source would then give a repetitions vector for the source that makes all of the support matrices in the network integer valued and solves the balance equations. **QED**

The rectangular case

Here we show that the constraint of the type in equation 10 is always satisfied by the solution to the balance equations when all of the lattices and matrices are diagonal. Since we are only interested in these additional constraints for arcs between an expander and decimator, consider the system in figure 14. The balance equations for arc AB are

$$\begin{aligned} L_1 r_{A,1} &= M_1 r_{B,1} \\ L_2 r_{A,2} &= M_2 r_{B,2} \end{aligned} \qquad \textbf{(EQ 11)}$$

The support matrix for arc AB is given by

$$W_{AB} = \begin{bmatrix} L_1 r_{A,1} & 0 \\ 0 & L_2 r_{A,2} \end{bmatrix}$$

since the input lattice and support matrix on the expanders input are both diagonal. The support matrix for arc BT is given by

$$W_{BT} = M^{-1} W_{AB} = \begin{bmatrix} M_1^{-1} & 0 \\ 0 & M_2^{-1} \end{bmatrix} \begin{bmatrix} L_1 r_{A,1} & 0 \\ 0 & L_2 r_{A,2} \end{bmatrix} = \begin{bmatrix} M_1^{-1} L_1 r_{A,1} & 0 \\ 0 & M_2^{-1} L_2 r_{A,2} \end{bmatrix}$$

We have $|N(W_{AB})| = L_1 L_2 r_{A,1} r_{A,2}$. A solution to the balance equations in equation 11 implies that the matrix W_{BT} is an integer matrix; hence, $|N(W_{BT})| = |det(W_{BT})| = M_1^{-1} L_1 r_{A,1} M_2^{-1} L_2 r_{A,2}$ and we see that equation 10 is satisfied since $|det(M)| = M_1 M_2$. So we see that the rectangular MDSDF case is a special case of the more general set of constraints needed for non-rectangular systems.

The fact that the decimator produces a varying number of samples per invocation might suggest that it falls nicely into the class of cyclostatic actors. However, there are a couple of differences. In the CSDF model of [3], the number of cyclostatic phases are assumed to be known before hand, and is only a function of the parameters of the

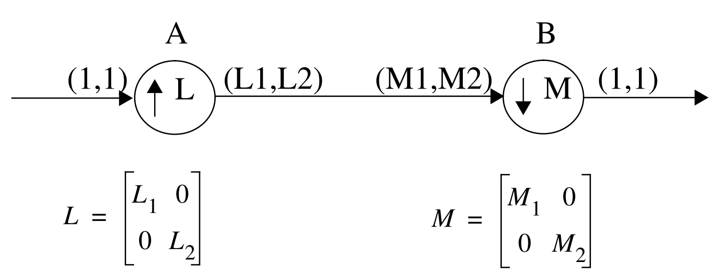


Fig 16. For a rectangular system, the constraint of equation 10 is always met.

Hence, our first requirement is that $r_{S,1}r_{S,2} = 2k \qquad k = 0, 1, 2, \ldots$. The balance equations gave us $r_{S,1} = 2, r_{S,2} = 1$; this satisfies the requirement. With these values, we get

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

Since this matrix is not integer-valued, lemma 3 cannot be invoked to calculate the number of integer points in $FPD(W_{BT})$. For non-integer matrices, the only way to compute $|N(W_{BT})|$ appears to be by brute force: by drawing this out on graph paper, it can be determined that there are 47 points inside. Hence, equation 10 is not satisfied! One way to satisfy equation 10 is to force W_{BT} to be an integer matrix. This implies that $r_{S,1} = 4k, k = 1, 2, \ldots$ and $r_{S,2} = 2k, k = 1, 2, \ldots$. The smallest values that make W_{BT} integer valued are $r_{S,1} = 4, r_{S,2} = 2$. From this, the repetitions of the other nodes are also multiplied by 2, thus increasing the *blocking factor* to 2, where the definition of the blocking factor is as in the MDSDF case. Note that the solution to the balance equations by themselves are not "wrong"; it is just that for non-rectangular systems equation 10 gives a new constraint that must also be satisfied.

We formalize the ideas developed in the example above in the following.

Lemma 4: The support matrices in the network can each be written down as functions of the repetitions variables of one particular source actor in the network.

Proof: Immediate from the fact that all of the repetitions variables are related to each other via the balance equations.

Lemma 5: In a multidimensional system, the j^{th} column of the support matrix on any arc can be expressed as a matrix that has entries of the form $a_{ij}r_{S,j}$, where $r_{S,j}$ is the repetitions variable in the j^{th} dimension of some particular source actor S in the network, and a_{ij} are rationals.

Proof: Without loss of generality, assume that there are 2 dimensions. Let the support matrix on the output arc of source S for one firing be given by $W_S = \begin{bmatrix} p & q \\ r & s \end{bmatrix}$. For $r_{S,1}, r_{S,2}$ firings in the "horizontal" and "vertical" directions (these are the directions of the columns of W_S), the support matrix becomes $W'_S = \begin{bmatrix} p & q \\ r & s \end{bmatrix}\begin{bmatrix} r_{S,1} & 0 \\ 0 & r_{S,2} \end{bmatrix} = \begin{bmatrix} pr_{S,1} & qr_{S,2} \\ rr_{S,1} & sr_{S,2} \end{bmatrix}$ (in multiple dimensions, the right multiplicand would be a diagonal matrix with $r_{S,j}$ in j^{th} row).

Now consider an arbitrary arc (u, v) in the graph. Since the graph is connected, there is at least one undirected path P from source S to node u. Since the only actors that change the sampling lattice (and thus the support matrix) are the decimator and expander, all of the transformations that occur to the support matrix W_S along P are left multiplications by some rational valued matrix. Hence, the support matrix on arc e, W_e, can be expressed as $W_e = AW_S$, where A is some rational valued matrix. The claim of the lemma follows from this.

Theorem 2: In an acyclic network of actors, where the only actors that are allowed to change the sampling lattice are the decimator and expander in the manner given by theorem 1, and where all source actors produce data according to definition 4, whenever the balance equations for the network have a solution, there exists a blocking factor vector J

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

Recall that the samples that the decimator produces are the integer points in $FPD(W_{BT})$. Hence, we want to know if

$$|N(W_{BT})| = |N(W_{AB})|/|M| \quad \textbf{(EQ 10)}$$

is satisfied by our solution to the balance equations. Now, by lemma 3, the size of the set $N(A)$ for an integer matrix A is given by $|det(A)|$. Since W_{AB} is an integer matrix for any value of $r_{S,1}, r_{s,2}$ (these variables are integers of-course), we have

$$|N(W_{AB})| = |det(W_{AB})| = 90r_{S,1}r_{S,2}$$

The right hand side of equation 10 becomes

$$(90r_{S,1}r_{S,2})/4 = (45r_{S,1}r_{S,2})/2$$

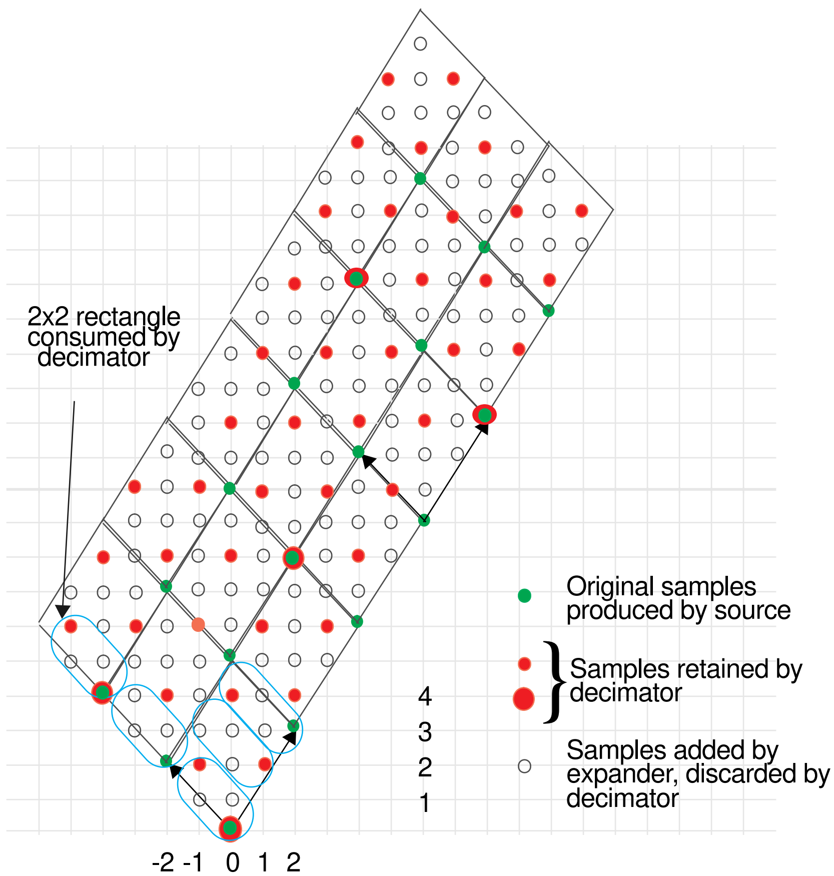


Fig 15. Total amount of data produced by the source in one iteration of the periodic schedule determined by the balance equations in equation 8.

We have already mentioned the manner in which the source produces data. We add that the subsequent firings of the source are always along the directions established by the vectors in the support matrix on the source's output arc.

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 $r_{X,2}$. These directions are dependent on the geometries that have been defined on the various arcs. Thus, for example, the directions are different on the input arc to the expander from the directions on the output arc. We have

$$\begin{aligned} 3r_{S,1} &= 1r_{A,1} \\ 3r_{S,2} &= 1r_{A,2} \\ 5r_{A,1} &= 2r_{B,1} \\ 2r_{A,2} &= 2r_{B,2} \\ r_{B,1} &= r_{T,1} \\ r_{B,2} &= r_{T,2} \end{aligned} \qquad \textbf{(EQ 7)}$$

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 but the calculations done below are still valid as will be discussed later. These equations can be solved to yield

$$\begin{aligned} r_{S,1} &= 2, r_{S,2} = 1 \\ r_{A,1} &= 6, r_{A,2} = 3 \\ r_{B,1} &= 15, r_{B,2} = 3 \\ r_{T,1} &= 15, r_{T,2} = 3 \end{aligned} \qquad \textbf{(EQ 8)}$$

Figure 15 shows the data space on arc AB with this solution to the balance equations. It might be reasonable to wonder whether, the total number of samples output by the decimator is equal to the total number of samples it consumes divided by the decimation factor if each actor is invoked the number of times specified by equation 8. After all, this is always true when we have purely rectangular lattices and rectangular expanders and decimators (this will be proved below). And of-course, this is also true in SDF. Note that this not an issue for the expander since it consumes (1,1) and produces exactly $|det(L)|$ for each one it consumes, unlike the decimator which might sometimes produce no samples at all since there might not be any that fall on its lattice amongst the ones it has consumed on some particular invocation.

In order to compute the number of samples output by the decimator, we have to compute the support matrices for the various arcs assuming that the source is invoked (2,1) times (so that we have the total number of samples being exchanged in one schedule period). We will do this symbolically using $r_{S,1}, r_{S,2}$ and plug in the values later. We get

$$W_{SA} = \begin{bmatrix} 3 & 0 \\ 0 & 3 \end{bmatrix} \begin{bmatrix} r_{S,1} & 0 \\ 0 & r_{S,2} \end{bmatrix} = \begin{bmatrix} 3r_{S,1} & 0 \\ 0 & 3r_{S,2} \end{bmatrix}$$

$$W_{AB} = LW_{SA} = \begin{bmatrix} 2 & -2 \\ 3 & 2 \end{bmatrix} \begin{bmatrix} 3r_{S,1} & 0 \\ 0 & 3r_{S,2} \end{bmatrix} = \begin{bmatrix} 6r_{S,1} & -6r_{S,2} \\ 9r_{S,1} & 6r_{S,2} \end{bmatrix} \qquad \textbf{(EQ 9)}$$

We start by specifying the lattice and support matrix for the arc SA. Let $V_{SA} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ and $W_{SA} = \begin{bmatrix} 3 & 0 \\ 0 & 3 \end{bmatrix}$. 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. We will need to specify the scanning order for each arc as well that tells the node the order in which samples should be consumed. Assume for the moment that the expander will consume the samples on arc SA in some natural order; for example, scanning by rows. Now we need to specify what the expander produces on each "firing". The natural way to specify this is that the expander produces $FPD(L)$ samples on each firing. What should be the scanning order for the production of these samples? Again, the natural order is one specified by the support matrix on arc AB (this is explained in detail later).

In order to get a set of "balance equations", as in the usual MDSDF case, we also have to order the samples in $FPD(L)$ in such a way that we can say that the expander produces (L_1, L_2) samples; this is understood to be the set $FPD(L)$ of points organized as a generalized (L_1, L_2) rectangle. Suppose we choose the factorization 5x2 for $|det(L)|$. Consider figure 14(b) where the samples in $FPD(L)$ are shown. One way to map the samples into an integer $(5, 2)$ rectangle is as shown by the groupings. Notice that the horizontal direction for $FPD(L)$ is the direction of the vector $\begin{bmatrix} 2 & 3 \end{bmatrix}^T$ and the vertical direction is the direction of the vector $\begin{bmatrix} -2 & 2 \end{bmatrix}^T$; the grouping reflects this nicely. We can number the samples as follows:

Table 1. Ordering the samples produced by the expander

Original sample	(0,0)	(0,1)	(0,2)	(1,2)	(1,3)	(-1,1)	(-1,2)	(-1,3)	(0,3)	(0,4)
Renumbered sample	(0,0)	(1,0)	(2,0)	(3,0)	(4,0)	(0,1)	(1,1)	(2,1)	(3,1)	(4,1)

Hence, $FPD(L)$ is a generalized $(5, 2)$ rectangle if we associate the function given in the table above with it as the rectangularizing function. Note that the above re-numbering has nothing to do with the numbering of the samples according to the lattice. Given a factoring of the determinant of L, the function given above can be computed easily; for example, by ordering the samples according to their euclidean distance from the two vectors that correspond to the horizontal and vertical directions. The scanning order for the expander across invocations is determined by the numbering of the input sample on the output lattice. For example, the sample at (1,0) that the source produces maps to location (2,3) in the re-numbered lattice at the expanders output. Hence, consuming samples in the [1 0] direction on arc SA results in 5x2 samples (i.e, $FPD(L)$ samples but ordered according to the table) being produced along the vector [2 3] on the output. Similarly, the sample (0,1) produced by the source corresponds to (-2,2) on the output. A global ordering on the samples is imposed by renumbering the sample at (2,3) as (5,0) since the first $FPD(L)$ of samples produced ended with sample (4,1). With this global ordering, it becomes clear what the semantics for the decimator should be. Again, choose a factorization of $|det(M)|$, and consume a "rectangle" of those samples, where the "rectangle" is deduced from the global ordering imposed above. For example, if we choose 2x2 as the factorization, then the (0,0) invocation of the decimator consumes the (original) samples at (0,0), (-1,1), (0,1), and (-1,2). The (0,2)th invocation of the decimator would consume the (original) samples at (1,3), (0,4), (2,3) and (1,4). The decimator would have to determine which of these samples falls on its lattice; this can be done easily.

5 A generalization of MDSDF for non-rectangular systems

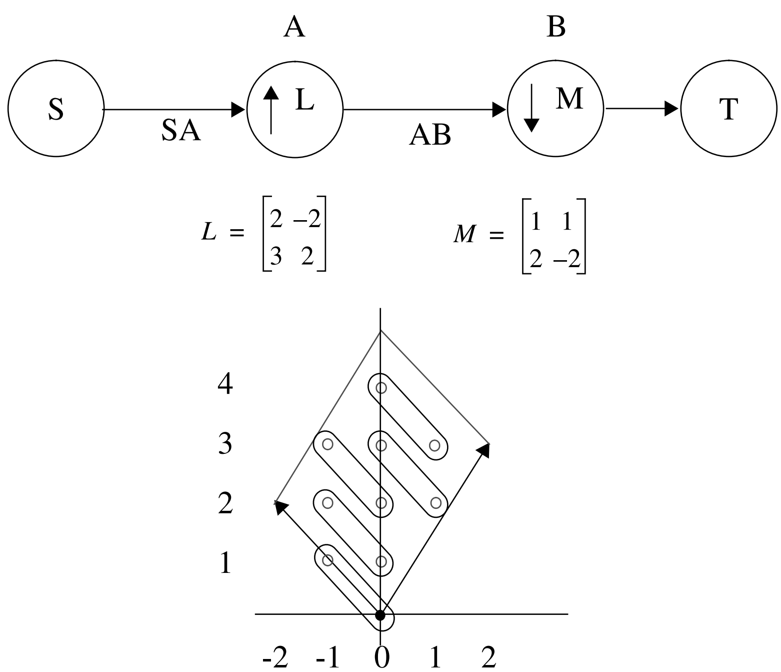
In the following, we will develop the semantics of a model that can express these non-rectangular systems by going through a detailed example. In general, our model for the production and consumption of tokens will be the following: an expander produces $FPD(L)$ samples on each firing where L is the upsampling matrix. The decimator consumes a "rectangle" of samples where the "rectangle" has to be suitably defined by looking at the actor that produces the tokens that the decimator consumes. While the asymmetry of the above definition, namely that $FPD(M)$ does not come into play (M is the decimation matrix) but $FPD(L)$ does, is a bit disconcerting, the asymmetry is to be expected since in one case samples are being produced while in the other case, samples are being consumed.

Definition 5: An **integer** (a, b) **rectangle** is defined to be the set of integer points in $[0, a) \times [0, b)$, where a, b are arbitrary real numbers.

Definition 6: Let X be a set of points in $\Re^2$, and x, y 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.

Example 1: Consider the system below, where a decimator follows an expander (figure 14(a))

a)



b)

Fig 14. 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).

Definition 4: 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 $\bar{\zeta} = \{V_S^{-1}\hat{n}: \hat{n} \in \zeta\}$ 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 13, we make the following definitions and statements.: Denoting the input arc to the decimator as e and the output arc as f, V_e, V_f are the bases for the input

Fig 13. Generalized expander and decimator with arbitrary input lattices and support matrices.

and output lattice respectively. W_e, W_f are the support matrices for the input and output arcs respectively, in the sense that samples, numbered according to the respective lattices, are the integer points of fundamental parallelepipeds of the respective support matrices. Similarly, we can also define these quantities for the expander depicted in figure 13. With this notation, we can state the following theorem:

Theorem 1: The relationships between the input and output lattices, and the input and output support matrices for the decimator and expander depicted in figure 13 are:

Decimator $\quad V_f = V_e M, \qquad W_f = M^{-1} W_e.$

Expander $\quad V_f = V_e L^{-1}, \qquad W_f = L W_e.$

Proof: The relationships between the input and output lattices follow from the definition of the expander and decimator. Consider a point n on the decimator's input lattice. There exists an integer vector k such that $n = V_e k$. If $M^{-1}k$ is an integer vector, then this point will be kept by the decimator since it will fall on the output lattice; i.e, $n = V_e M k'$ where $k' = M^{-1}k$. This point n is renumbered as $k' = M^{-1}V_e^{-1}n = M^{-1}k$ by the output lattice. Since k was the renumbered point corresponding to n on the input lattice, and hence in $N(W_e)$, every point k in $N(W_e)$ that is kept by the decimator is mapped to $M^{-1}k$ by the output lattice. Now, $k \in N(W_e) \Rightarrow \exists z \in [0, 1)^2$ s.t. $k = W_e z$. So $M^{-1}k \in N\left(M^{-1}W_e\right)$ because $M^{-1}k = M^{-1}W_e z$. Conversely, let j be any point in $N\left(M^{-1}W_e\right)$. Then, $\exists z \in [0, 1)^2$ s.t. $j = M^{-1}W_e z$. Since $W_e z = Mj$, we have that $Mj \in N(W_e)$. Also, the corresponding point to this on the input lattice is $V_e M j$ implying that the point is retained by the decimator. Hence, $W_f = M^{-1}W_e$. The derivation for the expander is identical, only with different expressions. **QED**

Corollary 1: In an acyclic network of actors, where the only actors that are allowed to change the sampling lattice are the decimator and expander in the manner given by theorem 1, and where all source actors produce data according to definition 4, the set of samples on every arc, renumbered according to the sampling lattice on that arc, satisfies the containability condition.

Proof: Immediate from theorem.

Notation: We will frequently use terms like "rectangular systems", or "rectangular MDSDF"; these always mean systems where all sampling lattices are rectangular. "Rectangular MDSDF" will mean the MDSDF model of computation from [15].

Consider the system depicted in figure 12, 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 need to introduce the concept of a "support matrix" that describes precisely the region of the rectangular lattice where samples have been produced. Figure 12 illustrates this for a decimation matrix,

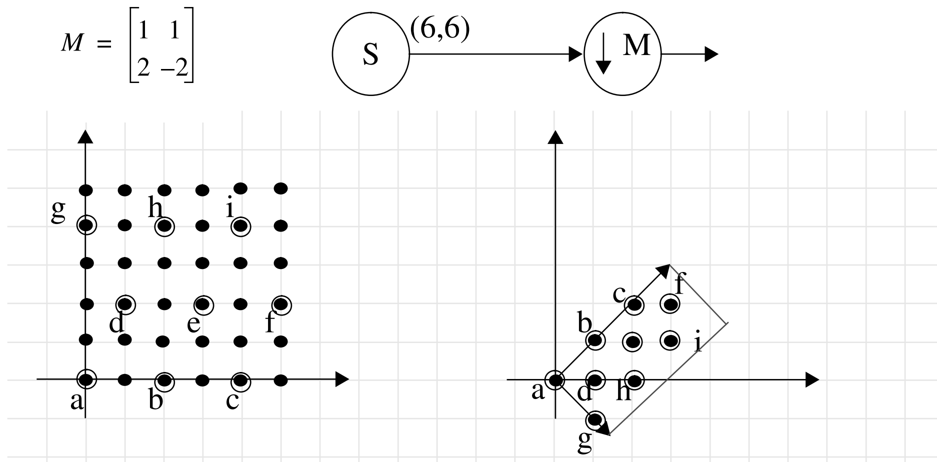


Fig 12. Output samples from the decimator renumbered to illustrate concept of support matrix.

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 3: 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 $N(W) = X$.

In SDF, the actor that represents a decimator consumes N tokens and produces 1 token on each firing, as illustrated in figure 11(a). Similarly, an expander actor consumes 1 token and produces L tokens on each firing. These actors have an additional parameter for the *phase* of upsampling or downsampling. For the downsampler, if phase = p, with $0 \le p < N$, then the output sample is the p^{th} input sample from the N samples consumed. For the upsampler, if phase = p, with $0 \le p < L$, then the input sample will be the p^{th} sample among the L output samples. These phases are usually fixed at compile time and do not change from invocation to invocation, although there is no reason why the phase cannot be time-varying (since a time-varying phase does not destroy the SDF semantics of the actor). However, since there do not seem to be any interesting signal processing uses of having a decimator (or upsampler) that has a time-varying phase, the phases are usually fixed.

The CSDF decimator and expander have M and L phases respectively. The decimator consumes one token in each phase but produces one token only on the p^{th} decimation-phase, and zero tokens in the other phases. The behavior of the upsampler is similar. Figure 11(c) shows these actors with a decimation-phase and expansion phase of 0.

The MDSDF decimator consumes a tuple $(M_1, \ldots, M_m)$ and produces $(1, \ldots, 1)$ while the MDSDF expander consumes $(1, \ldots, 1)$ and produces $(L_1, \ldots, L_m)$ (figure 11(b)). An m-tuple of phases can be specified for

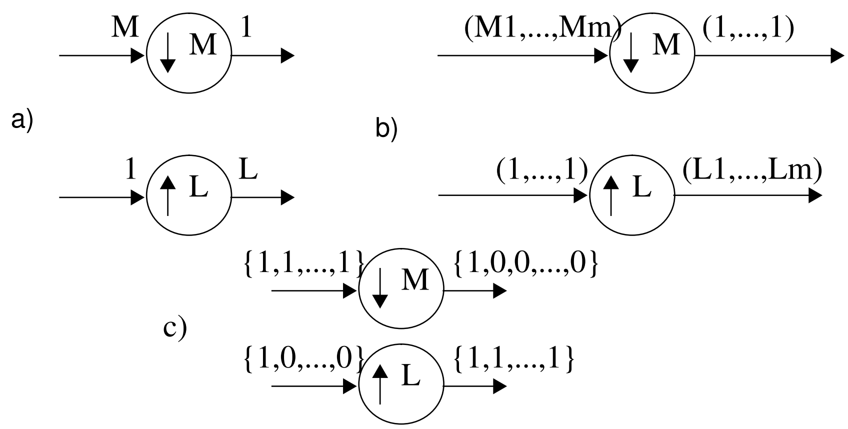


Fig 11. a) SDF decimator and expander. b) MDSDF decimator and expander. c) CSDF decimator and expander

these MDSDF actors as well; the phases specify how the input samples map to the output as in the SDF case.

Since the MDSDF decimator consumes a rectangle of samples for every sample it produces, the decimation matrix for an MDSDF decimator is always diagonal if the decimation phase is fixed and not time-varying: $M = diag(M_1, \ldots, M_m)$. Similarly, the expander matrix is also diagonal if the phase is fixed.

These observations show that MDSDF is not capable of allowing a) non-rectangular sampling lattices because the arcs are always rectangular arrays, and b) non-rectangular decimators and expanders because the produced/consumed semantics necessarily restrict us to consume and produce rectangles of data.

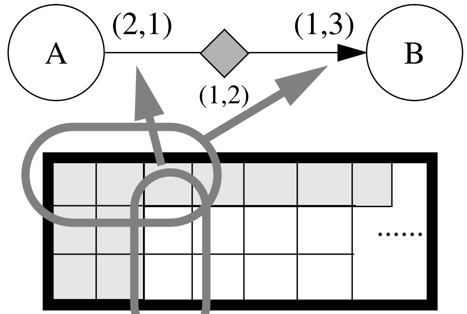


Fig 9. Delays in MDSDF

3.3 Cyclostatic Synchronous Dataflow (CSDF)

This is a generalization of SDF that has been introduced recently in [3]. In this model, a node is not restricted to consume or produce the same number of tokens on every invocation, but is allowed to vary in a restricted way. The number of tokens that the node produces or consumes is drawn from a finite set and is periodic with some finite period. Figure 10 shows an example of a CSDF graph. The notation means actor A has three phases: it pro-

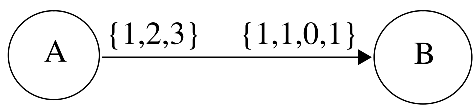


Fig 10. A CSDF graph

duces 1, 2, and 3 tokens in these phases. Similarly, actor B has four phases, and it consumes 1, 1, 0, 1 tokens in each of these phases. These phases always occur sequentially. Hence a valid schedule for the above graph would be one complete phase sequence for A, meaning three invocations, and two complete phase sequences for B meaning 8 invocations. For example, the 2nd and 4th invocations of B would consume the two tokens produced by the second invocation of A. One of the advantages of the CSDF model is that data buffering requirements are less for certain kinds of multirate graphs, although this might occur at the expense of code-size. However, the idea that a node need not consume or produce the same number of tokens all the time is an attractive one, and as we shall see, might be required in an MDSDF model capable of expressing non-rectangular systems.

4 MDSDF decimators and expanders

In this report, the only multirate actors we concentrate on are decimators and expanders in order to specify semantics for a generalized MDSDF model that can describe non-rectangular systems. The reason for concentrating only on these two actors is both for simplicity, and the fact that most practical signal processing systems that use non-rectangular lattices, like sampling structure converters and multirate filterbanks, are composed of decimators and expanders, in addition to some other non-multirate actors like FIR filters. Also, these two actors are the only two signal-processing actors that change sampling lattices.

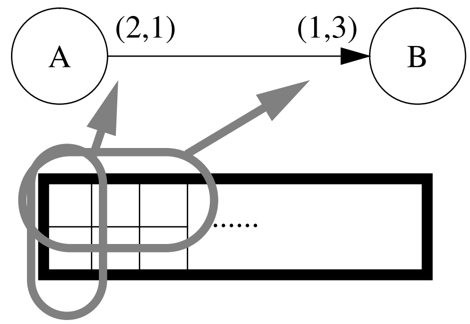


Fig 8. Data space for an MDSDF arc

3.2 Multidimensional Synchronous Dataflow (MDSDF)

A model more 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 analogous to equation 6 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 8 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 as follows:

$$r_{A,1} \times 2 = r_{B,1} \times 1$$
$$r_{A,2} \times 1 = r_{B,2} \times 3$$

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. The data produced in the next period will be thought of as data produced on columns 4 through 6. Notice that the actual buffer on the arc is of size 2x3. Usually, the horizontal dimension will represent time.

3.2.1 Delays

In SDF, a delay is an initial token or sample on the arc. In MDSDF, delays are also multidimensional tuples and represent initial *rows* and *columns*. Figure 9 shows a delay of (1,2) on an arc. This means that the production of data is offset by 1 row and 2 columns as shown in the figure. Note that the consumption of data is not offset; it proceeds as usual. Thus, in each firing cycle, the (0,0)th firing of B will always consume a vector of *initial* values in figure 9. There is also no change in the way the balance equations are written or solved.

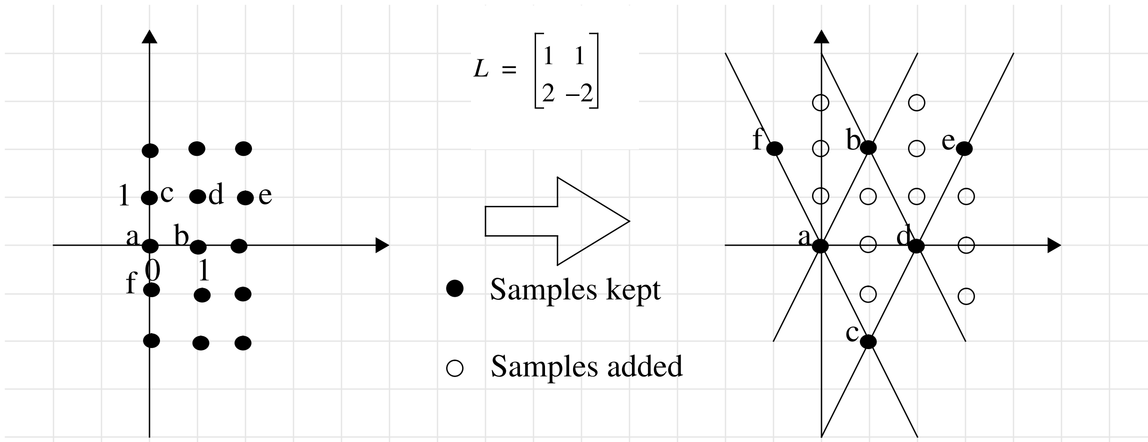


Fig 7. Renumbered samples from the expanders output

3 Models of computations

3.1 Synchronous Dataflow (SDF)

In the SDF model of computation, the algorithm or program is represented by a directed multigraph $G = (V, E)$ where the nodes represent computations and arcs represent precedence constraints and communication channels. Each node produces and consumes a fixed number of tokens, and these numbers are known at compile time. A **schedule** for an SDF graph is a sequence of actor invocations that return the buffers to their initial state. The state of a buffer is simply the number of tokens it contains. Supposing that a node u produces O_u^{uv} samples onto an arc (u, v), and node v consumes I_v^{uv}, we can write down a set of **balance equations** that specify that the total number of samples produced on the arc should equal the number of samples consumed in any schedule. Defining r_u and r_v to be the number of times u and v are invoked in any schedule, we get that

$$r_u O_u^{uv} = r_v I_v^{uv} \tag{EQ 6}$$

We get $|E|$ such equations (one for each arc). The smallest integer solution to these equations comprises the **repetitions vector** which specifies the number of times each node should be invoked in any periodic schedule. In [14], an efficient algorithm is given for solving the balance equations.

In the SDF model, arcs are FIFO queues of bounded length; these bounds can be easily computed once a schedule has been constructed. Since FIFO queues are inherently one-dimensional, the SDF model is well suited for specifying uni-dimensional signal processing algorithms where the data is always a one-dimensional stream. Ofcourse, higher dimensional systems can also be modelled if the data is suitably packaged as a one dimensional stream. For example, a video signal can be considered a one-dimensional stream if each particle in the stream is a two dimensional array representing an image at a sampling instant. However, modelling a video system this way does not expose all of the data parallelism that might be present. See [7] for more examples where using the SDF model for specifying multidimensional algorithms becomes awkward.

$$y(n) = \begin{pmatrix} x(n) & n = k'T_I \\ 0 & \text{otherwise} \end{pmatrix} \forall n = kT_I L^{-1}$$

In the multidimensional case, the definition is similar, but in terms of the $m \times m$ expander matrix L, which is again non-singular and has integer entries. The "expanded" output $y(\hat{n})$ is 0 unless $\hat{n}$ belongs to the lattice generated by V_I:

$$y(n) = \begin{pmatrix} x(n) & n \in LAT(V_I) \\ 0 & \text{otherwise} \end{pmatrix} \forall n \in LAT\left(V_I L^{-1}\right) \qquad \textbf{(EQ 5)}$$

where V_I is the input lattice to the expander. Note that $LAT(V_I) \subseteq LAT\left(V_I L^{-1}\right)$. The expansion ratio, defined as the number of points added to the output lattice for each point in the input lattice, is given by $|det(L)|$. This is because a point n on $LAT\left(V_I L^{-1}\right)$ that is also on $LAT(V_I)$ can be written as $n = V_I \hat{k} = V_I L^{-1} \tilde{k} \Rightarrow \tilde{k} = L\hat{k}$. So the points of y that are the original points of x are renumbered as $L\hat{k}$ by the output lattice. Every other integer point k in $FPD(L)$ is clearly a renumbering of the point $V_I L^{-1} k$ on the output lattice. Hence, $N(L)$ points are added by the expander for every point in the input; this implies that the expansion ratio is $|N(L)| = |det(L)|$. Figure 6 shows two examples of expansion. In the example on the left, the output lattice is also rectangular and is generated by

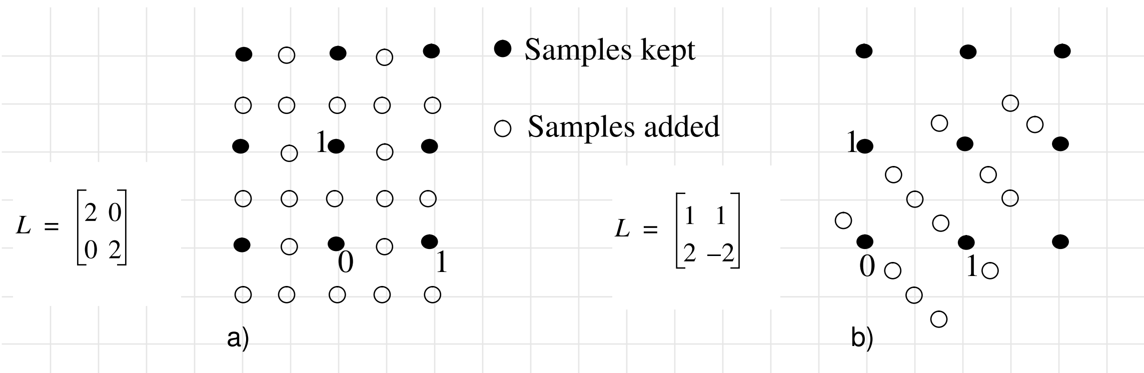


Fig 6. a) Rectangular expansion. b) Non-rectangular expansion

$$\begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix}.$$

The example on the right shows non-rectangular expansion, where the lattice is generated by

$$L^{-1} = \begin{bmatrix} 0.5 & 0.25 \\ 0.5 & -0.25 \end{bmatrix}$$

An equivalent way to view the above diagrams is to plot the renumbered samples. Notice that the samples from the input will now lie on $LAT(L)$ (figure 7). Some of the points have been labeled with letters to show where they would map to on the output signal.

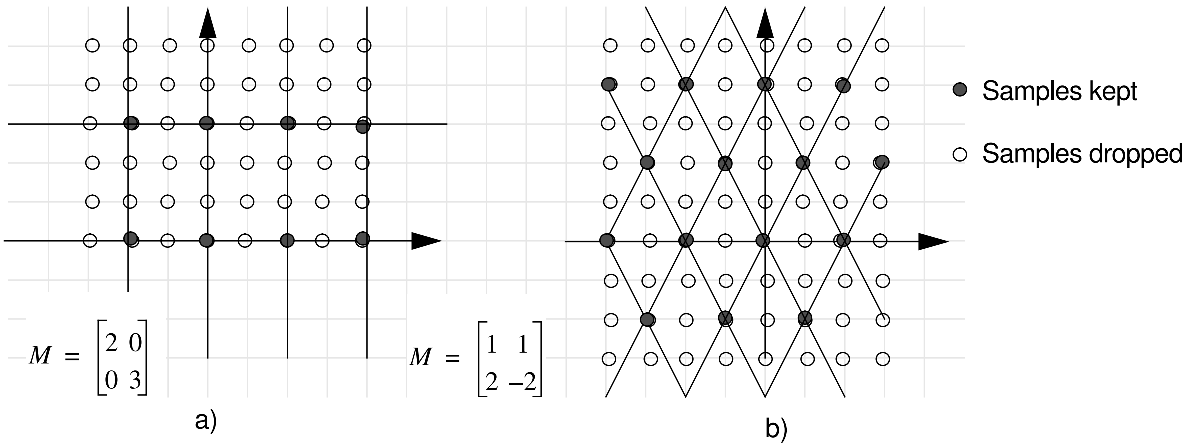


Fig 4. a) Rectangular decimation. b) Hexagonal decimation

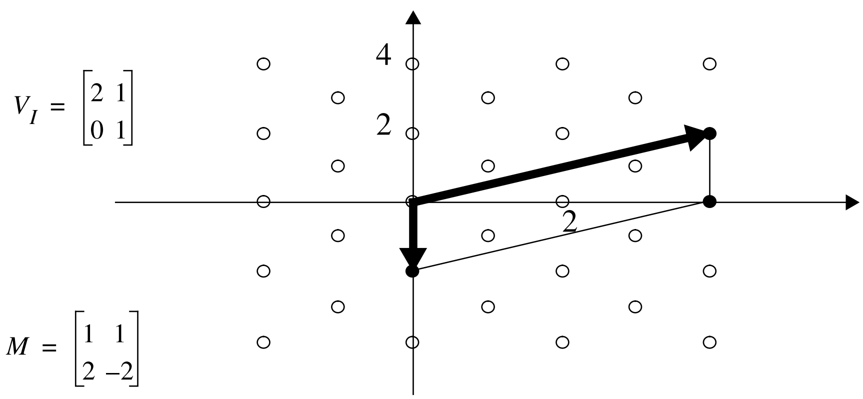


Fig 5. Decimation on a non-rectangular lattice

1*(2,0)+2*(1,1)=(4,2). Of-course, the points that are retained can also be determined by computing the lattice generated by $V_I M$. 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. A point n on the output lattice is of the form $n = V_I(Mk)$. Since this point is also on the input lattice $LAT(V_I)$, it is renumbered as Mk by the input lattice. Hence, only the points that fall on $LAT(M)$ when renumbered by $LAT(V_I)$ are kept by the output. Clearly, any integer point k in $FPD(M)$ is a renumbering of the point $V_I k$ on the input lattice. Hence, for every such point on $LAT(M)$, $N(M)$ points are discarded, implying that the decimation ratio is given by $|N(M)| = |det(M)|$. The decimation ratio for the example on the left in figure 4 is 6 and is 4 for the example on the right.

Multidimensional expanders

In the 1-D case, the L-fold expander is defined to be a device with the following input-output relationship:

$$y(n) = \begin{cases} x(n/L) & n = \text{multiple of L} \\ 0 & \text{otherwise} \end{cases}$$

Equivalently, if the input signal is sampled with period T_I, then we can write

Proof: See [20]; it also gives a polynomial time algorithm for finding C.

Lemma 3: Let V be an integer matrix. The number of elements in $N(V)$ is given by

$$|N(V)| = |det(V)| \qquad \textbf{(EQ 4)}$$

Proof: Consider an upper triangular matrix M. In two dimensions, $M = \begin{bmatrix} e & f \\ 0 & g \end{bmatrix}$, and this corresponds to a parallelogram with one of its sides on the x-axis. Since the side that is on the x axis has integer length e, the number of integer points on the x axis that lie inside $FPD(M)$ is just e. It is easy to see that there are f such rows of integer points in $FPD(M)$, and each of these corresponds to the vector $\begin{bmatrix} e & 0 \end{bmatrix}^T$ being shifted by an integer vector. Hence the number of integer points in each row is also e, making the total number of points in $FPD(M)$ equal to $ef = |det(M)|$. This argument can be extended to higher dimensions in the same way; the number of points will be the product of all the entries in M along the diagonal. Hence the lemma is true for integer, upper triangular matrices.

By lemma 2, we can always find an integer unimodular matrix C such that CV is upper triangular for any integer matrix V. Let x be an integer point in $N(CV)$. The point $C^{-1}x$ is an integer point and is in $N(V)$. Hence, for every integer point x in $N(CV)$, there is a unique integer point $C^{-1}x$ in $N(V)$. For every integer point x in $N(V)$, there is a unique integer point Cx in $N(CV)$. Hence, the size of $N(V)$ equals the size of $N(CV)$. Since CV is upper triangular, $|N(CV)| = |det(CV)| = |det(V)|$, where the last equality follows from the unimodularity of C. **QED**

2.1.4 Multirate multidimensional operators

Multidimensional decimators

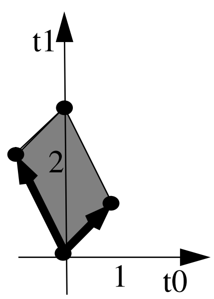
The two basic multirate operators for multidimensional systems are the decimator and expander. Given a 1-D discrete-time signal $x(n)$, the M-fold decimated version of the signal is defined as $y(n) = x(Mn)$, where M is a positive integer. Equivalently, if the input signal has sample period T_I, then the decimated version is given by $y(n) = x(n), n = T_I Mk$. In an analogous way, 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***. Figure 4 shows two examples of decimation. The example on the left is for a diagonal matrix M; this is called rectangular decimation because $FPD(M)$ is a rectangle rather than a parallelepiped. In general, a rectangular decimator is one for which the decimation matrix is diagonal. The example on the right is for a non-diagonal M and is loosely termed "hexagonal" decimation.

In the above example, the input signal to the decimator was on the rectangular lattice. Note that in order to use the method of drawing the column vectors of M to determine the samples retained, the columns of M have to be interpreted according to the renumbering by the input lattice. Figure 5 shows an example where the input signal is on a non-rectangular lattice. Notice that the point (2,0) is actually renumbered as (1,0) by this lattice, and the point (1,1) is renumbered (0,1). Hence, the point (1,2)=1*(1,0)+2*(0,1), and the first column of M is actually the point at

depends on the sampling matrix, and hence the sampling geometry, and thus, by a judicious choice of sampling geometry, we can make sampling density as low as possible. An example of this is the sampling of an m-dimensional signal that is bandlimited to a hyperspherical region in the frequency plane. The savings in sampling density by using a particular non-diagonal sampling matrix ("hexagonal" sampling) over a diagonal sampling matrix (rectangular sampling) for sampling this hyperspherically bandlimited signal is a factor of 1.15 for $m = 2$. The savings factor goes up to a factor of 2 for 4 dimensional signals, and a factor of 16 to 8 dimensional signals [9]. Hence, there is a practical reason to favor non-rectangular sampling schemes over rectangular ones because of the potential savings in storage requirements and processing rates due to the lower sampling density that can result [17]. Some applications that favor non-rectangular sampling schemes include 2:1 interlaced TV scanning [8], hierarchical video coding applications [5], and filterbanks for doing directional decomposition [2]. Filter design techniques for non-rectangular lattices have also been maturing as evidenced by many publications in the area [12][24][1][2][10][22][23].

2.1.3 The fundamental parallelepiped

Given a sampling matrix V, suppose that the column vectors are sketched from the origin, as shown in figure 3 for the example from figure 2. The completed parallelogram resulting from these vectors is called the funda-



$$V = \begin{bmatrix} 1 & -1 \\ 1 & 2 \end{bmatrix}$$

Fig 3. The fundamental parallelepiped for a matrix V

mental parallelepiped of V, denoted $FPD(V)$. The points which fall inside $FPD(V)$ can be represented as the set $V\hat{x}$ where $\hat{x} = \begin{bmatrix} x_1 & x_2 \end{bmatrix}^T$, with $0 \le x_1, x_2 < 1$. The entire lattice $LAT(V)$ can be thought of as a tiling of the plane by copies of $FPD(V)$ shifted so that there is no overlap with other tiles, with the points of the lattice always falling on the corners of these tiles.

From geometry it is well known that the volume of $FPD(V)$ is given by $|det(V)|$. Since only one integer sample point falls inside $FPD(V)$, namely the origin, we can see why the sampling density is given by the inverse of the volume of $FPD(V)$.

Definition 2: Denote the set of integer points within $FPD(V)$ as the set $N(V)$. That is, $N(V)$ is the set of integer vectors of the form $V\hat{x}, \hat{x} \in [0, 1)^m$. The following lemma characterizes the number of integer points that fall inside $FPD(V)$, or the size of the set $N(V)$. But before that, we state another lemma that is required for the proof of the lemma characterizing $|N(V)|$.

Lemma 2: For any integer matrix A, there exists an integer, unimodular matrix C such that CA is upper triangular.

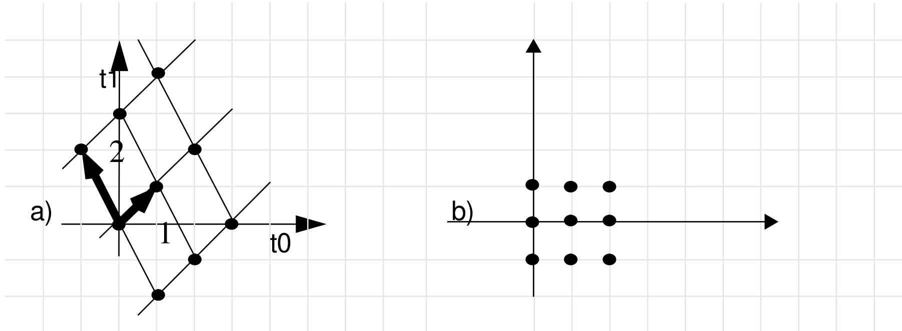


Fig 2. Sampling on a non-rectangular lattice. a) The samples on the lattice. b) The renumbered samples of the lattice.

The set of all sample points $\hat{t} = V\hat{n}$, $\hat{n} \in \aleph$, that is, the set of vectors $\sum_{k=1}^{m} n_k \hat{v}_k$, is the set of all integer linear combinations of the columns $\hat{v}_1, \ldots, \hat{v}_m$ of the sampling matrix V. This set is called the ***lattice*** generated by V, and is denoted $LAT(V)$. The matrix V is the ***basis*** that generates the lattice $LAT(V)$. The basis for a lattice is not unique; for example,

$$LAT\left(\begin{bmatrix} 1 & -1 \\ 1 & 2 \end{bmatrix}\right) = LAT\left(\begin{bmatrix} 0 & -1 \\ 3 & 2 \end{bmatrix}\right)$$

The set of matrices that generate the same lattice can be characterized as follows.

Definition 1: A unimodular integer matrix E is a matrix with integer entries such that $det(E) = \pm 1$. Note that the inverse of E is also a unimodular integer matrix.

Lemma 1: Let V be a an $m \times m$ real non-singular matrix generating the lattice $LAT(V)$. Then, for any integer unimodular matrix E, $LAT(V) = LAT(VE)$. If $\widehat{V}$ is any other basis for $LAT(V)$, then there exists an integer unimodular matrix E such that $\widehat{V} = VE$.

Proof: This proof can be found in [23].

2.1.1 Numbering on a lattice

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 $LAT(V)$. Figure 2(b) shows the renumbered samples for the samples on $LAT(V)$ shown in figure 2(a).

2.1.2 Sampling density

The determinant of the sampling matrix plays a role in the sampling of an MD signal. The *sampling density* ρ, defined as the number of sample points per unit volume (or area in the 2-dimensional case), is given by $\rho = 1/|det(V)|$. The sampling density is an important concept that plays a role in the multidimensional sampling theorem. Intuitively, for an MD signal that is bandlimited in some fashion, we expect the sampling density to be above some minimum if we expect to retain all of the information in the signal. However, the sampling density

mentally more complicated than sampling a unidimensional signal because of the many different ways the sampling geometry can be chosen. The straightforward extension of unidimensional sampling would result in the sequence $x(n_1, n_2) = x_a(n_1T_1, n_2T_2)$ where we have considered the $m = 2$ case for simplicity. Thus all values of t_1, t_2 that are integer multiples of the sampling periods T_1, T_2 are retained by the sampler. Figure 1 shows the samples that are retained for the case where $T_1 = 2, T_2 = 1$. As can be seen, the samples are arranged in a rectangular pattern, and thus this sampling scheme is known as **rectangular sampling**. A more general sampling scheme is to consider the sequence generated by

$$x(n_1, n_2) = x_a(a_{11}n_1 + a_{12}n_2, a_{21}n_1 + a_{22}n_2) \quad \textbf{(EQ 1)}$$

Notice that the sample locations retained are given by the equation

$$\hat{t} = \begin{bmatrix} t_1 \\ t_2 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} n_1 \\ n_2 \end{bmatrix} = V\hat{n} \quad \textbf{(EQ 2)}$$

The matrix V is called the sampling matrix. Every sample location $\hat{t}$ is of the form

$$\hat{t} = n_1\hat{v_1} + n_2\hat{v_2} \quad \textbf{(EQ 3)}$$

where

$$\hat{v_1} = \begin{bmatrix} a_{11} \\ a_{21} \end{bmatrix}, \hat{v_2} = \begin{bmatrix} a_{12} \\ a_{22} \end{bmatrix}.$$

That is, the sample locations are vectors $\hat{t}$ that are linear combinations of the columns of the sampling matrix V. Given the sampling matrix, the sample locations can be obtained graphically by first drawing the two vectors $\hat{v_1}$, $\hat{v_2}$ from the origin. Then draw two sets of equispaced parallel lines such that the two vectors form two sides of a parallelogram generated by these lines. The sample points are then located at the intersections of these lines. Figure 2(a) shows an example.

Note that the sampling matrix need not be an integer matrix but must be real and non-singular. Using the above terminology, rectangular sampling can be represented by a diagonal sampling matrix:

$$V = \begin{bmatrix} T_1 & 0 \\ 0 & T_2 \end{bmatrix}$$

In fact, rectangular sampling is defined to be a sampling scheme for which the sampling matrix is diagonal.

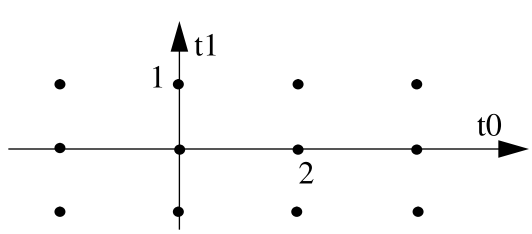


Fig 1. An illustration of rectangular sampling

Technical Memorandum, UCB/ERL M95/59, Electronics Research Laboratory, University of California at Berkeley, Berkeley Ca 94720

August 24, 1995

A Generalization of Multidimensional Synchronous Dataflow to Arbitrary Sampling Lattices

Department of Electrical Engineering and Computer Science

University of California

Berkeley, California 94720

Praveen K. Murthy
Edward A. Lee
March, 20 1995

1 Abstract[1]

Multidimensional Synchronous Dataflow (MDSDF) [15][7] is a model of computation that has been proposed for specifying multidimensional multirate signal processing systems such as image and video processing algorithms. The model is an extension of synchronous dataflow (SDF) [14] and has all of the desirable properties of the SDF model such as static schedulability, exposition of data and functional parallelism, and a visually pleasing syntax well suited for block diagram signal processing environments such as Ptolemy [6] and Khoros [13]. However, the MDSDF model as specified in [15] is limited to modelling multidimensional systems sampled on the standard rectangular lattice. Since many multidimensional signals of practical interest are sampled on non-rectangular lattices, for example, 2:1 interlaced video signals, and many multidimensional multirate systems use non-rectangular multirate operators like hexagonal decimators, it is of interest to have models that are capable of representing and simulating such systems. This report 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.

2 Introduction

2.1 Multidimensional signal processing fundamentals

A multidimensional signal of dimension m, $x_a(t_1, \ldots, t_m)$, is a function of m real variables $t_1, \ldots, t_m$. This signal can be sampled to generate a discrete time signal. However sampling a multidimensional signal is funda-

1. A portion of this research was undertaken as 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), Semiconductor Research Corporation (project 94-DC-008), National Science Foundation (MIP-9201605), Office of Naval Technology (via Naval Research Laboratories), the State of California MICRO program, and the following companies: Bell Northern Research, Dolby, Hitachi, Mentor Graphics, Mitsubishi, NEC, Pacific Bell, Phillips, Rockwell, Sony, and Synopsys.