In the first one, we have shown how to define a model of data representation as a multiprojection. This representation is convenient in terms of reusability and development of functions. More precisely we have seen that a function can be reused independently of the number of dimensions by using M-operators to transpose the data representation.

The second idea is to use higher-order function with a graphical language. We have used the higher-order function concept with one-dimensional functions to build representations of multidimensional signal processing. Higher-order functions allow us to use a hierarchical representation that is a convenient feature in designing complex simulations. It also presents an expressive description of the functions applied in a style that is close to the hardware implementation. Although not explored in this paper, it also exposes the natural parallelism in the algorithm.

These methods have been combined successfully to simulate a complex radar processing algorithm.

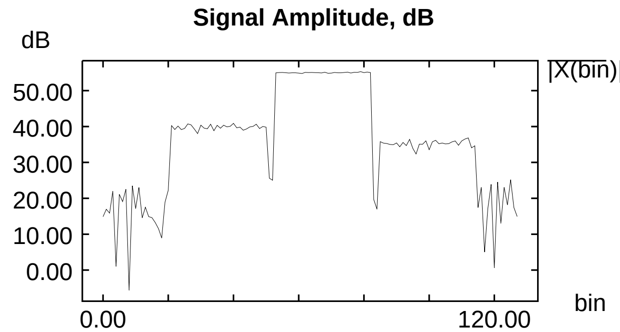
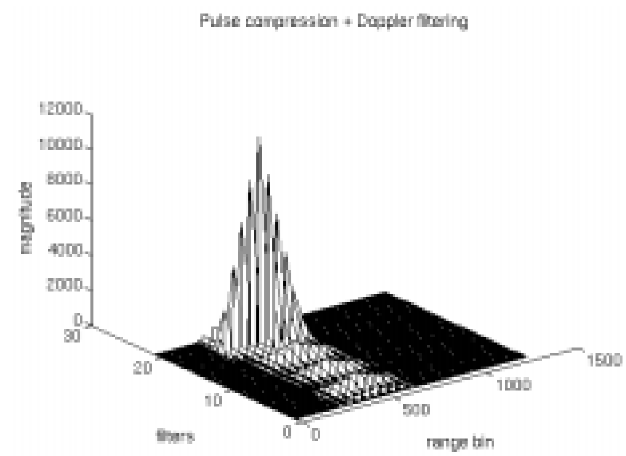
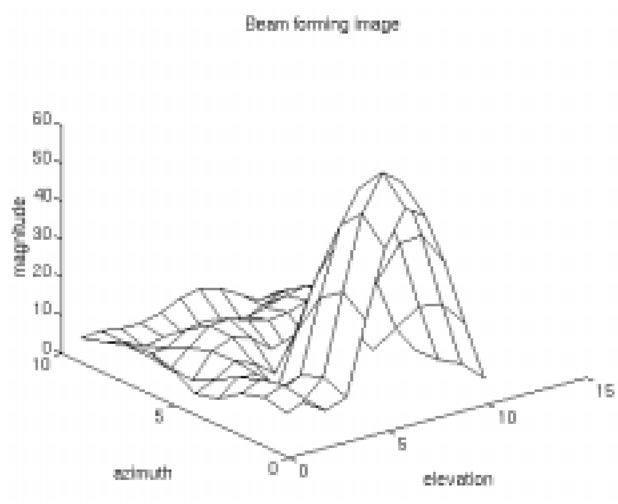


Figure 7. Generation of three targets by a HOF.



applied on one target.



target (Figure 8)

8. ACKNOWLEDGEMENTS

The authors gratefully acknowledge the entire Ptolemy team for building the infrastructure within which we could carry out these experiments.

9. REFERENCES

[1] J. T. Buck, S. Ha, E. A. Lee, D. G. Messerschmitt, "Multirate Signal Processing in Ptolemy", *Proceedings of ICASSP*, Toronto, April 1991.

[2] J. 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," Vol. 4, pp. 155-182, April, 1994.

[3] G. Desodt, D. Muller, D. Puzenat, "OSCAR, a simulation environment dedicated to the design and performances assessment of RADAR system," *International Conference on Radar*, p. 414,419, Paris, 1994.

[4] F. H. M Fransen, F. Balasa, M. F. X. B. van Swaaij, F. V. M. Cathoor, and H. J. De Man "Modeling Multidimensional Data and Control Flow", *IEEE Trans. on VLSI Systems*, 1(3), pp. 319-327, September, 1993.

[5] D. L. Johnson, London Mathematical Society, *Presentations of Groups*, Cambridge University press, 1990.

[6] E. A.Lee, "Dataflow process networks", Memo No. UCB/ERL M94/53, ERL, UC-Berkeley, Ca 94720

[7] E. A. Lee and D. G. Messerschmitt, "Synchronous Dataflow", *Proceedings of the IEEE*, September, 1987.

[8] E. A. Lee, "Representing and Exploiting Data Parallelism Using Multidimensional Dataflow Diagrams," *Proc. of IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, Minneapolis, MN, April, 1993.

.

5. RADAR FUNCTIONS

A radar function is usually applied to a single dimension, like filtering applied in the range bin dimension, beamforming applied to the channels, and doppler filtering in the pulse dimension. In fact, the dimensions are orthogonal in terms of radar treatment.

Since the model of simulation is based on dataflow process networks, the functions are applied to a set of tokens extracted from the main data stream. Thus, the principle in using higher-order functions is to define a generic function that operates only on the modified dimensions. Before we apply the higher-order functions, we must often use an M-operator to present the correct dimension at the input to the higher-order functions. The purpose of the M-operator is to reorganize the data stream in a way so that the inner dimensions are the ones manipulated by the higher-order functions. Thus the other dimensions, no matter how many, are transparent.

Therefore, to define a radar function we see it as an elementary operation applied on generic elements. This approach presents a simple and concise way to define a function in a manner equivalent to functional languages using higher-order functions.

6. RADAR SIMULATION

This simplified example contains more than eight levels of hierarchy and is presented top-down. This simulation models a parameterizable radar system, where the geometry of the antenna network, the chirp pulse shape, the filters used, and other parameters can be easily varied.

The first two levels (which contain two other hidden levels) illustrate how the reception on an elementary or generic receiver can be defined using the hierarchical language. The second level uses a higher-order function to allow us to simply generate signals reflected by several targets with added thermal noise. This higher-order function generates different instances for the distance, the magnitude, the angle, and the speed of the target. (figure 7.)

On the third level, the first leftmost higher-order function is used to generate the signal reception for the target on different sensors. This higher-order function creates different instances for position and angles defining the antenna network. Each sensor can be viewed as a process that runs concurrently. Usually the natural concurrent structure of a higher-order function appears simply in the visual representation. The next higher-order function realizes pulse compression on different channels. Then there is a windowing function. An initial M-operator appears in order to present the pulse dimension to the input of the doppler filtering function (figure 8).

The fifth level creates one beamformer. The last level with a higher-order function allows us to generate as many beams as we desire (figure 9).

7. CONCLUSIONS

In this paper, we have developed a methodology for building visual dataflow representations of signal processing applications that operate on multidimensional data. Two key innovations are described.

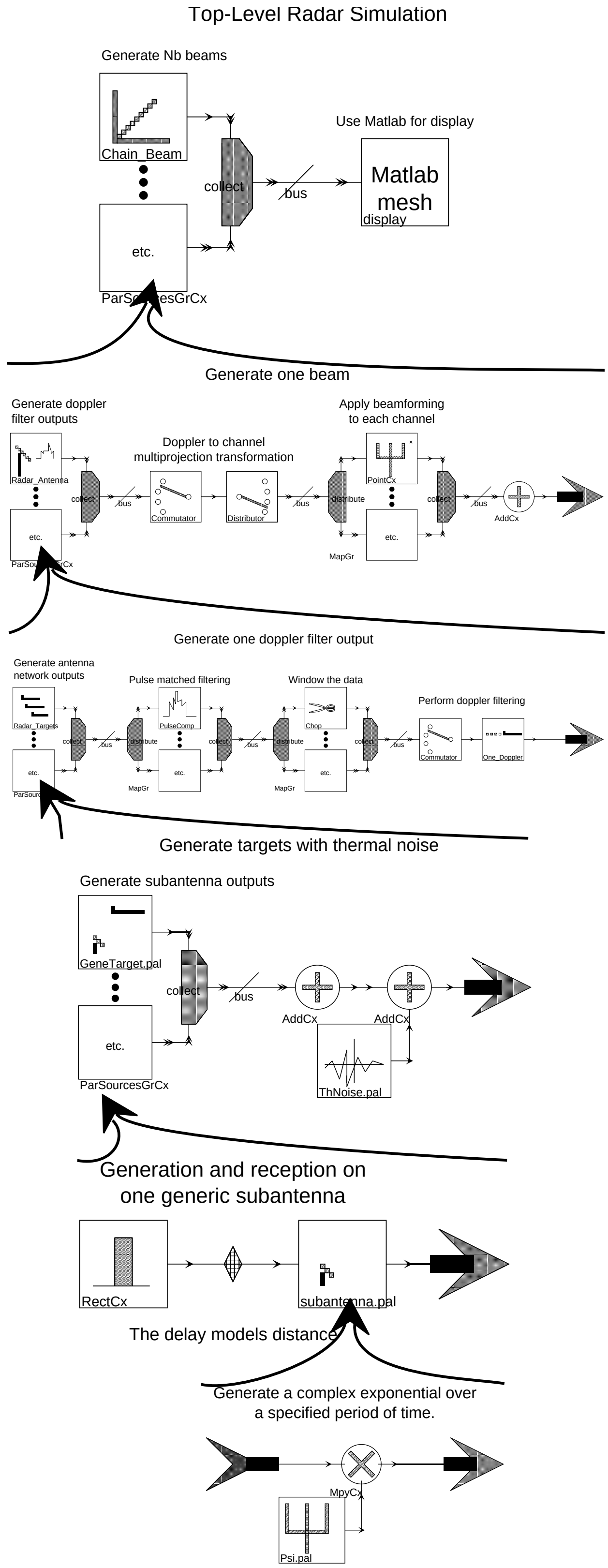


Figure 6. Hierarchical structure of the radar simulation with top-down representation of the functions.

3. MULTIPROJECTION OPERATORS

From an n-dimensional hypercube of data, we can generate $n!$ multiprojections or permutations of the n dimensions. There are many ways to define transformations from one multiprojection to another. We use two basic operators available in the Ptolemy synchronous dataflow domain, the *distributor* and the *commutator.* These are simple to realize and are close in spirit to a direct hardware approach. Of course such transformations can also be defined analytically in terms of piece-wise linear functions [4]. A distributor begins by consuming α elements from its input and transfering them to the first output; in the second step it takes the same α number of elements from its input and transfers them to the second output; it continues until the γ^{th} output is used, and then it begins again on with the first output. (Figure 2).

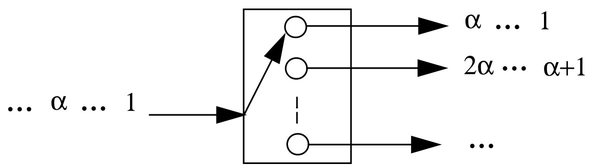


Figure 2. Distributor definition. $Dist^{\alpha}_{\gamma}$

The inverse function of the distributor is a commutator with γ inputs. We define an *M-operator* to be a composition of a distributor and a commutator. We realize a mapping ϕ from the set of M-operators to the group of permutations. This is a many-to-one mapping, where some M-operators have no image in the group of permutations of the hypercube dimensions.

Below, we give the image of the inverse of ϕ restricted to the subset of transpositions.

$$[id] \quad Com^{\alpha}_{\beta} \bullet Dist^{\alpha}_{\beta}(a_{ijk}) = a_{ijk}$$

$$[\sigma_L] \quad Com^{I}_{J\times K} \bullet Dist^{1}_{J\times K}(a_{ijk}) = a_{ijk}$$

$$[\sigma_L] \quad Com^{1}_{I} \bullet Dist^{J\times K}_{I}(a_{ijk}) = a_{jki}$$

$$[\sigma_R] \quad Com^{1}_{I\times J} \bullet Dist^{K}_{I\times J}(a_{ijk}) = a_{kij}$$

$$[\sigma_R] \quad Com^{I\times J}_{K} \bullet Dist^{1}_{K}(a_{ijk}) = a_{kij}$$

$$[\sigma_{12}] \quad Com^{K}_{I} \bullet Dist^{J\times K}_{I}(a_{ijk}) = a_{jik}$$

$$[\sigma_{12}] \quad Com^{I\times K}_{J} \bullet Dist^{K}_{J}(a_{ijk}) = a_{jik}$$

$$[\sigma_{23}] \quad Com^{J}_{K} \bullet Dist^{1}_{K}(a_{ijk}) = a_{ikj}$$

$$[\sigma_{23}] \quad Com^{1}_{J} \bullet Dist^{K}_{J}(a_{ijk}) = a_{ikj}$$

$$[\sigma_{13}] \quad \sigma_R \bullet \sigma_{12}(a_{ijk}) = a_{kji}$$

$$[\sigma_{13}] \quad \sigma_L \bullet \sigma_{23}(a_{ijk}) = a_{kji}$$

We have the same type of expressions for 4-dimensional hypercubes; these can be nearly deduced directly from the previous formulae. A recurrent algorithm can be defined to generalize these relations to n-dimensional hypercubes; the principle is to merge two invariant adjacent dimensions and then apply a $(n\text{-}1)$ dimensional M-operator formula. Knowing that the transposition generates the group of permutations [5], we can determine any multiprojection in terms of M-operators.

4. HIGHER-ORDER FUNCTIONS AND VISUAL PROGRAMMING LANGUAGES

A general form of most of the functions used to simulate radar processing is $Y=F(X,D)$, where X and Y represent the main stream of data. More precisely, X is the input signal, while Y is the output that is the result of applying the function F using the auxiliary parameters or data D. For example, in a filtering function, D is the filter to apply on the data X, while in pulse compression (matched filtering), D is the original chirp pulse shape to convolve with X.

The higher-order function concept for visual programming languages is discussed in [6]. Higher-order functions are functions that have other functions as arguments or return functions. In languages supporting higher-order functions, a function is a first-class data structure. This concept has been implemented in Ptolemy [6].

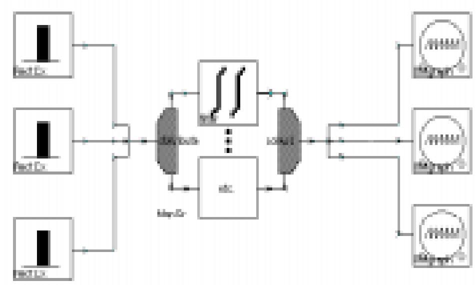


Figure 4. higher order function use with filtering function.

In figure 4 we see a higher-order function used in a filtering operation. Three signals are generated without using a higher-order function at the left. Then in the middle, a filter block is applied to each of the three signals. The structure in the middle of figure 4 represents a higher-order function that maps a specified function (given graphically by the upper block) to the input streams. The number of instances of the filter block is determined by its context rather than being specified directly graphically. At the right, three display blocks plot the signals. Figure 5 shows an equivalent schematic built entirely without higher-order functions.

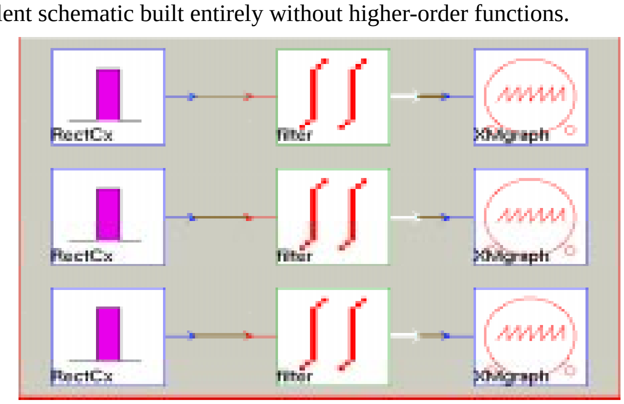


Figure 5. Equivalent schematic for HOF.

Usually, each of the three subgraphs in figure 5 has different values for its parameters. For instance, each schematic can have a specific filter to apply to its channel, taking into account different defect corrections. The higher-order function in figure 4 must managing the mapping of parameter values to the instances of the filter.

— Proceedings of ICASSP-95, Detroit, May, 1995 —

MODELING RADAR SYSTEMS USING HIERARCHICAL DATAFLOW

Karim P. Khiar and Edward. A. Lee
Thomson-CSF and University of California, Berkeley
{khiar, eal}@eecs.berkeley.edu

ABSTRACT

The synchronous dataflow model is used in a variety of visual programming environments to describe and design digital signal processing systems [7]. In this paper, we present two main improvements over existing methodologies. Both are concerned with convenient manipulations of multidimensional data. The first describes a systematic method for transposing multidimensional data structures embedded within a one dimensional stream. This enables the use of scalar stream operators for processing multidimensional data. The second shows how higher-order functions combined with visual hierarchy can be used to build intuitive, maintainable, and scalable applications that operate on multidimensional data. These techniques are combined to design a beamforming radar simulation using the Ptolemy simulation environment [2].

1. INTRODUCTION

In the signal processing industry, simulation is used extensively in developing complex systems, including radar processing systems [3]. The development of new algorithms induces a constant improvement of simulation models in terms of precision and complexity.

In this paper we develop a methodology to define a radar simulation by taking advantage of visual higher-order functions. Radar simulations, like most real time signal processing tasks, require processing significant quantities of data. These data usually represent a collection of different signals. The organization of the data implies some constraints on the way the data is processed by the different radar functions [4]. Thus, in section 2, we define a suitable data representation and a transformation called a multiprojection in order to manipulate multidimensional data within a one-dimensional stream. In section 3, we define a set of operators for generating all possible multiprojections. These multiprojections bring an independence to the way the data is processed from the ordering of the data in the stream. Section 4 briefly defines the higher-order function concept. This feature is only a part of the unique functionality available in a visual programming language. In section 5, we show how to use these higher-order functions to model radar functions. The last section presents an application to a radar simulation using the Ptolemy simulation environment.

2. MULTIDIMENSIONAL DATA AND MULTIPROJECTION

A collection of signals can be represented as a multidimensional data structure ordered by dimension.

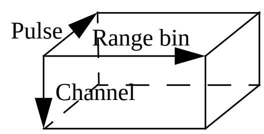


Figure 1.a

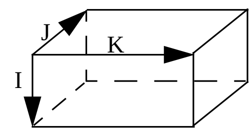


Figure 1.b

In the case of radar processing, the dimensions have a precise meaning (Figure 1a). Usually we have more than three dimensions. Typically the dimensions are indexed by the range bin, channel number, pulse number, target, doppler filter, beam number, azimuth, elevation, etc. To simplify and to generalize, we consider the dimensions abstractly (Figure 1b).

One representation of such multidimensional data that would be suiltable for visual programming uses multidimensional streams [8]. That method, however, is not yet proven, and has not been fully implemented in any software environment. We therefore adopt a more conservative approach here that relies only on one-dimensional streams.

A multiprojection is a representation of this *n*-dimensional hypercube of data in a one-dimensional stream of data. We take the notation a_{ijk} to represent one particular multiprojection that could be obtained by writing nested loops and printing the data, where *k* is the inner loop index and *i* the outer loop index.

When each element of this hypercube is seen as a data *token*, then this representation is suitable for embedding in a one-dimensional stream. There exist more direct *n*-dimensional high-level descriptions of data for modeling data flow [3][4][8], but we demonstrate in the next section that the one-dimensional stream of data is not fundamentally a restrictive representation. We do this by showing that a pair of simple operators can accomplish any arbitrary transposition of the data.

1. This research was sponsored by the THOMSON CSF Detection Systems Group, and was carried out 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, Cadence, Dolby, Hitachi, Mentor Graphics, Mitsubishi, NEC, Pacific Bell, Philips, Rockwell, Sony, and Synopsys.