PARAMETER ESTIMATION FRAMEWORK FOR FUSING SPATIO-TEMPORAL DATA IN WATERSHED ANALYSIS AND ITS POTENTIAL FOR RC-BASED HARDWARE ACCELERATION

By

KARTHIK NAGARAJAN

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

2009

To my mom and dad

ACKNOWLEDGMENTS

This work would not have been possible if not for the continued support, belief and guidance of many people. First, I want to thank my parents Smt. Vanitha Nagarajan and Shri. G. Nagarajan for their support, prayers, and the wonderful learning platform they have provided since early childhood. I am also thankful to my brother and sister-in-law for having provided moral support and the occasional fun that is ever so required in a graduate student's life. I also must thank my wonderful wife Vidhya for giving her love and support. She has been patient, understanding, and sacrificed her standard of living and more importantly her time with me while I remained in school.

I wish to thank my advisors, Prof. K. Clint Slatton and Prof. Alan D. George for their continued guidance and encouraging words. Without their input and financial support, this work would not have been made possible. Though tough at times, they have gradually molded me into a better researcher. I also wish to thank Dr. Wendy Graham for helping me understand certain hydrology-related problems and also for the time and effort she invested in refining the research contributions. I am also thankful to my friends at the Adaptive Signal Processing Laboratory and the Center for High-performance Reconfigurable Computing for their inputs and support.

This work was partially supported by the National Science Foundation (Grant No. EAR-0609968) and I/UCRC Program of the National Science Foundation (Grant No. EEC-0642422).

TABLE OF CONTENTS

page

ACKNOWLEDGMENTS 4

LIST OF TABLES 7

LIST OF FIGURES 8

ABSTRACT....10

CHAPTER

1 INTRODUCTION 12

1-1 Unified Data Fusion Framework 12
1-2 Rapid Computation on Hardware Platforms 14
1-3 Contributions and Outline 17

2 BACKGROUND 24

2-1 Data Fusion Methodologies in Hydrology 24
2-2 Study Site and Data Description 27
2-2-1 Observed Data....28
2-2-2 Derived Data 29
2-3 Hardware Acceleration on FPGAs 31
2-4 Case-study Algorithms 34
2-4-1 Probability Density Function Estimation....34
2-4-2 K-Means Clustering 37
2-4-3 Correlation 38
2-4-4 Algorithm Similarity....38

3 SPATIO-TEMPORAL BAYESIAN NETWORK 42

3-1 Bayesian Network Fundamentals 44
3-1-1 Network Learning....46
3-1-2 Network Topology....47
3-1-3 Network Inference 48
3-1-4 Performance Metrics....49
3-2 Measurement of Incremental Information....50
3-2-1 Feature Ranking 50
3-2-2 Validating Information-Gain Analysis 52
3-3 Model Predictions at Training Stations 52
3-4 Model Predictions at Intermediate Stations....55

4 HARDWARE ACCELERATION ON FPGAS 68

4-1 Pattern-based Algorithm Decomposition 68
4-1-1 Pattern Description 69
4-1-1-1 Pipeline pattern 69
4-1-1-2 Datapath replication pattern 70
4-1-1-3 Loop fusion pattern 71
4-1-2 Quantifying Design Patterns 72
4-2 Scalable and Portable Architecture for PDF Estimation 73
4-2-1 Performance Prediction 74
4-2-2 1-D PDF Design 77
4-2-2-1 Single-core design 78
4-2-2-2 Dual-core design 79
4-2-3 2-D PDF Design 79
4-2-4 Experimental Results 80
4-2-4-1 Speedup, power, and resource utilization 81
4-2-4-2 Data verification 82
4-2-4-3 Scaling to multiple cores 82
4-2-4-4 Portability considerations 83
4-2-4-5 Scaling to multi-FPGA systems 84
4-3 Composite Patterns for Design Reuse 85
4-3-1 K-Means Clustering 85
4-3-2 Correlation 88

5 MULTISCALE ESTIMATION FRAMEWORK 98

5-1 Test Area 100
5-2 Multiscale Estimation Framework 100
5-3 Mixture of Experts (MOE) 103
5-4 Results and Analyses 104

6 CONCLUSIONS AND FUTURE WORK 112

6-1 Message Passing 113
6-1 Streamflow Estimation via Message Passing 114

LIST OF REFERENCES 120

BIOGRAPHICAL SKETCH 127

LIST OF TABLES

Table		page
1-1	Elapsed time in estimating high dimensional PDFs using Parzen windows.	20
2-1	Set of available features	39
3-1	Comparison of computational complexity between traditional Bayesian method, RDBE and STBN	57
3-2	Feature ranking based on conditional entropy (CH)	57
3-3	Performance metrics for cases 1-4 validating information-gain analysis	57
3-4	Performance metrics for networks N1 (Q_{net}) and N2 (WAM)	58
3-5	Performance metrics for N2 under 80% Training Data	58
3-6	Performance metrics for N2 under 50% Training Data	58
4-1	List of primitive patterns and their description	89
4-2	Parameters of primitive patterns	89
4-3	RAT input parameters for analysis of 1-D and 2-D PDF estimation algorithms	90
4-4	Speedup, power, and utilization for single-core designs	90
4-5	Performance factors for single-core designs	90
4-6	Speedup and utilization for dual-core designs	91
4-7	Speedup and utilization for single-core designs across platforms	91
4-8	Speedup for single-core 2-D PDF design across multiple FPGAs	91
4-9	Speedup and utilization for PDF estimation, K-means clustering, and Correlation (single-core designs)	91
5-1	MSE obtained over the simulated image	107
5-2	Incremental variance across scales and empirical $F(s)$ values	107
5-3	Average model weights obtained using MM-MKS for real data	108
6-1	Performance metrics at 2321500 after message passing	118

LIST OF FIGURES

Figure		page
1-1	Fusing data measured across domains	21
1-2	Research concept diagram	22
1-3	Research roadmap.	23
1-4	Availability of measurements at multiple scales	23
2-1	A map of the Santa Fe River Watershed with its streams and rivers	40
2-2	Illustration of case-study algorithms	41
2-3	Dataflow and computational structure in PDF estimation, K-means clustering, and Correlation	41
3-1	Basic node structures in BNs	58
3-2	Sample network representation of a BN for embedding spatio-temporal data	59
3-3	Bayesian networks for streamflow estimation given spatio-temporal data sources	59
3-4	Streamflow estimates at station 2321000	60
3-5	Streamflow estimates at station 2321500	61
3-6	Streamflow estimates at station 2322500	62
3-7	Model validation at 2321000	63
3-8	Model validation at 2321500	64
3-9	Model validation at 2322500	65
3-10	Streamflow estimation at intermediate points	66
3-11	Streamflow and MAP estimates along the Santa Fe river on October 4th, 2006.	66
3-12	Observed versus predicted streamflow at USGS station 2321975	66
3-13	Model validation at 2321975	67
4-1	Pattern-based analysis of 1-D PDF estimation algorithm	92
4-2	Development stages in system design	92

4-3 State transition diagram for a single-core 1-D PDF design 93
4-4 Design details of 1-D PDF 93
4-5 Host-centric execution flows 94
4-6 Architecture design for 2-D PDF estimation algorithm 94
4-7 State transition diagram for a single-core 2-D PDF design 94
4-8 1-D PDF estimates on GPP and FPGA 95
4-9 2-D PDF estimates on GPP and FPGA 95
4-10 Multi-FPGA system configuration 95
4-11 Architecture for K-means clustering 96
4-12 Decomposition and design of correlation 97
5-1 Forest hydrologic cycle processes 108
5-2 Elevation image created by gridding last stop elevation points using an inverse-distance weighting of study area 109
5-3 Mixture of Gaussians fitted over the histogram plot of fractal dimension 109
5-4 Fusion of Kalman filter bank estimates using Mixture-of-experts network 109
5-5 Fine-scale simulated image with two terrain roughness 110
5-6 Weights obtained for smooth and rough filters at scales m=7 and 6 respectively 110
5-7 Filter weights under large and small P_0 111
5-8 Weights assigned to Model 1, 2, and 3 at various scales 111
6-1 Illustration of message passing 118
6-2 Message passing framework for improving streamflow estimates 118
6-3 Prediction at USGS 2321500 after message passing 119
6-4 Composite pattern for tree-based algorithms 119
6-5 Composite pattern for mesh-based algorithms 119

Abstract of Dissertation Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Doctor of Philosophy

PARAMETER ESTIMATION FRAMEWORK FOR FUSING SPATIO-TEMPORAL DATA IN WATERSHED ANALYSIS AND ITS POTENTIAL FOR RC-BASED HARDWARE ACCELERATION

By

Karthik Nagarajan

August 2009

Chair: K. Clint Slatton
Cochair: Alan D. George
Major: Electrical and Computer Engineering

Knowledge of the flow rates in rivers and major streams is critical for understanding and effectively managing watersheds because it bears on such things as predicting flood vulnerability and fluvial erosion of the landscape, as well as mitigating the damaging effects of floods and waterborne pollution. Spatially dense networks of *in situ* streamflow measurements are generally nonexistent because of the high cost of installing, calibrating, and maintaining the necessary equipment. A framework is therefore needed to predict streamflow from other available data. The relevant data are often of different modalities, however, such as images (e.g., digital topographic maps), time-series (e.g., an *in situ* streamflow guage), and spatio-temporal (e.g., daily radar-based precipitation estimates), and the estimation of parameters that exhibit complex spatio-temporal dependencies from such multi-modal measurements poses a significant challenge. Most data fusion algorithms explored to date work with data belonging to a single mode or "family" possessing one-to-one sample mappings, such as images or time series data, and cannot be directly applied here. In this work, a simple yet efficient probabilistic framework based on Bayesian networks (BNs) is developed that provides a mechanism to incorporate

spatial, temporal and spatio-temporal features. Graph topologies are generated based on physical models characterizing the underlying random process.

Solving high-dimensional estimation problems over large data sets leads to computational demands that are often impractical to run even on high-end General Purpose Processors. This is true for the framework proposed in this work as well. FPGA-based reconfigurable computing has been successfully used to accelerate computationally intensive problems in a wide variety of scientific domains to achieve speedup over traditional software implementations. However, this potential benefit is quite often not fully realized because creating efficient FPGA designs is generally carried out in a laborious, case-specific manner requiring a great amount of redundant time and effort. Algorithm decomposition, performance prediction, and design reuse for parallel implementation all need to be performed in an efficient and structured manner in order to develop successful FPGA designs in a productive way. To address these challenges, an approach for pattern-based decomposition of algorithms for FPGA design and development is proposed.

The major contributions of this work include (1) development of a scalable approach to fuse spatial, temporal and spatio-temporal watershed data via a BN to estimate streamflows; (2) prediction of uncertainties in streamflow estimates as a function of location (space), condition (time) and the particular mix of sensors; and (3) accelerating algorithm execution on FPGAs using pattern-based decomposition with significant increases in productivity via design reusability.

CHAPTER 1
INTRODUCTION

1-1 Unified Data Fusion Framework

Traditional data fusion is the process of putting together information in the form of measurements (or features) collected from a heterogeneous set of sensors into a single composite "picture" of the environment. The composite picture, for example, might deal with parameter estimation or a classification problem wherein the results depend on the set of features. Such data fusion problems arise in a variety of contexts, most notably in remote sensing and geophysical applications. For example, flow of water through stream networks and rivers directly impacts flooding and transport of sediments and pollutants (e.g., nutrients) in watershed systems. Hence knowledge of streamflow rates is critical for management of water quality, in-stream flows and mitigation of flooding and drought events.

Unfortunately, spatially dense networks of *in situ* streamflow meters are not generally available and would be prohibitively expensive to deploy and maintain. Thus, an estimation framework is needed that utilizes the available data to predict streamflow through a river network. Spatial data, such as surface topography, landcover, and soil type; temporal data, such as time series of measured streamflow and groundwater levels; and spatio-temporal data, such as rainfall, all bear on streamflow. Some of these quantities can be obtained from remote sensing imagery; however, utilizing such disparate data types using traditional data fusion methods is problematic.

In general, estimation of parameters that exhibit complex spatial and temporal dependencies on multi-modal measurements (features) poses a significant challenge. This is because most data fusion algorithms explored to date work with data samples belonging to a single mode or "family" of data, such as image data at a particular point in time [1] or time series

data at a particular point in space. Certain algorithms do offer the possibility to fuse measurements of different types (e.g., spatial data with point measurements) [2]. In doing so, they still only capture the variations of the parameter being estimated spatially or temporally. In other words, most methods developed for fusing image data do not integrate point-source and distributed time series data efficiently.

Application-specific deterministic models (e.g., physically-based hydrologic models) have been used to predict streamflow in watersheds, but often with significant uncertainty because numerous assumptions often have to be made for many unmeasured input and parameter values.

Purely statistical methods (e.g., regression analysis) have been used extensively as an alternative to physically-based models in an effort to account for unmeasured input parameters and/or parameter uncertainty. However, such approaches often deal only with second-order moments (variance) and also impose calibration requirements for their extrapolation to new locations or conditions.

Probabilistic methods (e.g., traditional Bayesian inference) that utilize the joint (multivariate) probability density function (PDF) of all the features and parameters rather than merely a finite subset of its statistical moments can be used to incorporate (fuse) different data sets in order to estimate streamflow. This approach, however, suffers from a super-linear increase in computational complexity as the number of data sets to be fused grows.

This demonstrates the need for a unified estimation framework that provides the computational platform to fuse spatio-temporal, spatial and/or temporal data (Figure 1-1), in conjunction with limited training data/measurements. When measurements for some desired input parameters are unavailable the framework should accommodate hidden or proxy variables through the use of physically-based relationships. In this way, the “information” contained in

our physical understanding of the system can leverage the information in the training data, while uncertainty and variability in the parameters can be accommodated to relate parameters whose precise deterministic relationships are not well known.

1-2 Rapid Computation on Hardware Platforms

Machine-learning is a general term that has been used to refer to a large and diverse set of methods for extracting patterns from data. In general, it encompasses portions of statistical and adaptive signal processing, probabilistic decision theory such as Bayes' Rule, and meta-heuristic strategies such as the genetic algorithm. Machine-learning algorithms have become widely used for problems such as detection, estimation, and classification involving diverse data types, including time series, image, and video (spatio-temporal) data [3]. Raw image and video data do not explicitly contain high-level information and thus generally require an abstraction process to extract useful information from the data. This process is often accomplished via a sequence of data segmentation algorithms [3, 4]. Clustering algorithms are the most common type of algorithms used for segmenting data points into groups and employ various measures of similarity. Probabilistic algorithms can then be employed to represent the segmented data as PDFs rather than single data points (e.g., cluster centers). Such data representations are essential in many applications involving decision-making based on probabilistic reasoning [5, 6, 7]. Alternatively, as pointed out in Section 1-1, certain estimation algorithms use statistical methods to model relationships between datasets as a function of the separation between them (dissimilarity). These methods are useful for applications such as data content retrieval and indexing [8], data segmentation, and speech recognition systems [9, 10] that benefit from quantitative measures of the statistical relationships between features.

The streamflow estimation framework proposed in this work is driven by machine-learning algorithms primarily based on a data-driven approach wherein the computational burden is

heavily impacted by the volume of data being processed. Three such principal algorithms are the Parzen window-based non-parametric estimation of PDFs, K-means clustering and correlation. While applying the estimation framework over small watersheds (e.g., the 3585 km^2 Santa Fe River watershed) would result in moderate execution times, extending the framework to larger watersheds (e.g., the 28542 km^2 Suwannee River basin) would lead to unreasonable computational times (Figure 1-2). For example, Table 1-1 illustrates the computational burden incurred in terms of time elapsed (where optimized C code is executed on a 3.2 GHz single-core Xeon processor) in computing PDFs of increasing dimension (number of features considered) using the Parzen window technique with Gaussian kernels. The total number of data samples was set to 204800 (N) and the support size of the PDF on each dimension was set to 256 (n). The high computational burden arises because most problems involve large volumes of data and require complex computations to be performed in a high-dimensional space. To mitigate this problem, application researchers have investigated ways to approximate the solution by either solving the problem under a reduced dimensional space or providing alternative methods like the Fast Gauss Transform (FGT) [11]. The FGT addresses the dimensionality issue to a certain extent by reducing the number of computations in each dimension. Irrespective of the approximation method used, the computation involved still grows exponentially with increasing dimensionality.

Since the three algorithms mentioned (Parzen estimation of PDFs, K-means clustering and correlation) form an integral part of the estimation framework (and other numerous machine-learning applications as well), their fast and efficient execution is extremely desirable. Further, according to Amdahl's law [12], significant improvements to the overall execution time of an application are obtained only when significant portions of it are improved. In other words,

overall efficiency gains are most likely to occur when the principal (or most often used) portions of the machine-learning tasks are made more efficient. FPGA-based reconfigurable computing (RC) has been successfully used to accelerate computationally intensive problems in a wide variety of scientific domains to achieve speedup over traditional software implementations.

Developing FPGA designs, however, is often time-consuming and not every algorithm is amenable to hardware acceleration. Preliminary analyses should be performed to predict the algorithm's amenability to a hardware paradigm before undertaking a lengthy development process. In contrast to the general computing domain, there is a relatively modest variety of FPGA-based hardware platforms from which a designer can choose. These platforms primarily vary in the type of FPGA they house and the manner in which data is communicated between the FPGA and host processor; variations on each platform are also available based on the constraints on the particular application design and performance. For example, decisions regarding bit precision, design architecture, memory utilization, and storage depend on the FPGA's resource availability and the size of the hardware design. The presence of dedicated arithmetic blocks also dictates the implementation options for certain mathematical instructions. Thus, it is best to determine the expected performance attainable at an early stage when migrating an algorithm to an FPGA platform by employing simple performance prediction tools.

Another barrier to successfully using FPGA-based RC technology for algorithm acceleration is the prohibitive time and effort that is required to develop the necessary FPGA core designs in a custom, case-specific manner. A more desirable approach would be to recognize and exploit common patterns in terms of computational structure and data flow across a variety of algorithms and prove that those design patterns are indeed suitable for use in other FPGA implementations. The primary goal is hence to develop a concise set of design patterns

that can be formally used to represent the decomposition of an algorithm for parallel implementation. If the hardware design of an algorithm can be represented at a high level graphically via composite patterns (mixture of computation and communication patterns), then other algorithms having a similar high-level representation can reuse the existing design with only minimal customization. Such a high-level description is also desirable to aid inexperienced FPGA designers (e.g., domain scientists) so they can easily comprehend and reuse existing hardware designs while developing new applications. Pattern-based descriptions of designs could be used to effectively disseminate and share hardware cores as well. Such an approach can lead to a significant increase in productivity via design reusability. Several important machine-learning algorithms, such as PDF estimation using Parzen windows, K-means clustering, correlation, and information-theoretic measures, have very similar underlying algorithmic and dataflow structures and thus could potentially benefit from such an approach. In fact, the FGT algorithm can be decomposed for a parallel implementation in a similar fashion to that of the Parzen window technique and could potentially benefit from the architecture designed for the latter. In general, the similarity in dataflow in many machine-learning algorithms is due to the fact that application data (input to the learning system) is compared with several representative values (parameters of the learning system) to perform inference. In addition to being structurally similar, most of these algorithms are often applied to very large volumes of data and impose difficult constraints on memory.

1-3 Contributions and Outline

In this dissertation, *"a unified and efficient framework to fuse spatio-temporal, spatial and/or temporal data for streamflow prediction is presented and key algorithms in the framework are accelerated on FPGAs in a productive manner via design reuse using pattern-based algorithm decomposition."* The topic is timely because the hydrology community is

currently seeking principled approaches to determine how best to instrument large watersheds in the United States with limited resources. The National Science Foundation (NSF) has termed such instrumented watersheds "hydrologic observatories" as an analogy to large astronomical observatories in which NSF invests funds so that many different investigators can use the same resource. While the exact schedule for the creation of these hydrologic observatories remains uncertain, NSF has made a strong commitment to this vision by funding several pilot projects [13]. The work herein is designed to advance our knowledge of how such an observatory might be characterized algorithmically.

The major contributions of this work include (1) development of a scalable approach to fuse spatial, temporal and spatio-temporal data via a Bayesian Network (BN) to estimate streamflow in a watershed system; (2) prediction of uncertainties in streamflow estimates as a function of location (space), condition (time) and the particular mix of sensors; and (3) accelerating algorithm execution on FPGAs using pattern-based decomposition with significant increases in productivity via design reusability.

Chapter 1 gave a basic overview of the problem of fusing data collected from disparate sources across various domains, some of the existing methodologies, their characteristics, and scope for improvements. The issue of computational burden faced while handling large datasets and the choice of FPGAs as potential solutions were pointed out. The major contributions in the work and the application case studies that would be used for purposes of illustrating the proposed concepts were also presented (Figure 1-3).

Chapter 2 discusses background on existing methodologies for solving data fusion problems in hydrology and the amenability of signal processing algorithms for implementation on FPGAs. Discussions on relevant performance prediction methods and design patterns for RC

used in this work are briefly presented. Case-study algorithms (from the estimation framework) chosen for hardware acceleration are detailed as well. A brief description of the study site (the Santa Fe River watershed system) and the features used to predict streamflow are also provided.

Chapter 3 describes the Spatio-Temporal Bayesian Network (STBN) method that offers a unified platform to incorporate measurements observed across various domains. The network is then applied to the problem of estimating streamflow in a watershed system taking into account the complex non-linear dependencies of flow with a variety of other measurements made across space and time. When sufficient training data are available for the full dynamic range of the random variables in each dimension, optimal graph topologies for BNs can be obtained using optimization methods, such as simulated annealing [14]. However, in many geophysical applications, such as ours, coincident measurements are only available for relatively limited time periods or spatial locations, thus we must exploit our physical knowledge of the hydrologic processes to determine reasonable BN topologies. An information-theoretic methodology based on conditional entropy is then employed to help quantify the impact of adding nodes in the graphical model in terms of information gained. While this approach does not strictly guarantee that the optimal BN topology is found, it does allow us to measure the benefit of one topology over another. The framework also offers the flexibility of embedding knowledge from hydrologic models calibrated for the study area by introducing them as additional nodes in the network, thereby improving prediction accuracy. Posterior probabilities of streamflow estimates and the associated entropy values provide valuable information on the quality of predictions and offer directions for future watershed instrumentation.

Chapter 4 describes the pattern-based decomposition, analysis, and implementation of a scalable and portable architecture for multi-dimensional, non-parametric PDF estimation using

Gaussian kernels on FPGAs. The algorithm's amenability to a hardware paradigm and expected speedups are predicted before implementation. Multi-core architectures are developed to further improve performance by scaling the design. Portability of the hardware design across multiple FPGA platforms is also analyzed. After implementing the PDF algorithm, the value of pattern-based decomposition to support reuse is demonstrated by rapid development of the K-means and correlation algorithms.

Chapter 5 presents a multiscale estimation framework that would prove useful in data fusion problems where measurements (image data in particular) are available at a wide range of scales. In such cases, there is a need to convert all the features to a common spatial resolution (through a downsample/upsample operation). Most often, this is a scale at which the spatio-temporal features are measured (Figure 1-4). Although this work does not extend the streamflow estimation framework to incorporate multiscale data, a proof-of-concept study for multiscale analysis is provided and the actual integration is left as future work.

Chapter 6 summarizes the work through conclusions and lists ideas for future work.

Table 1-1. Elapsed time in estimating high dimensional PDFs using Parzen windows.

Dimension	Time elapsed	Increase factor
1	0.58 seconds	×1
2	158.75 seconds	×275
3	~12.00 hours (estimated)	×75000
4	~139.00 hours (estimated)	$\sim\times 2\times 10^7$

"Estimated" indicates the expected time elapsed based on knowledge of algorithm complexity. The total number of data samples was set to 204800 (N) and the support size of the PDF on each dimension was set to 256 (n)

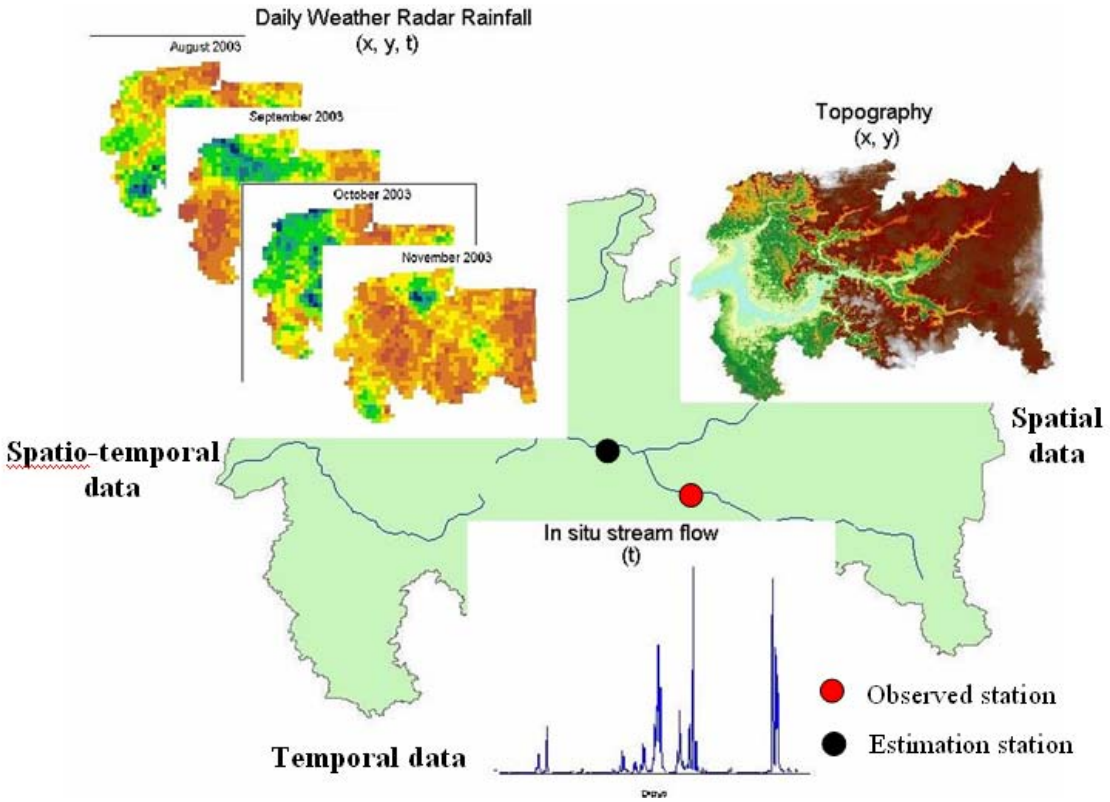


Figure 1-1. Fusing data measured across domains.

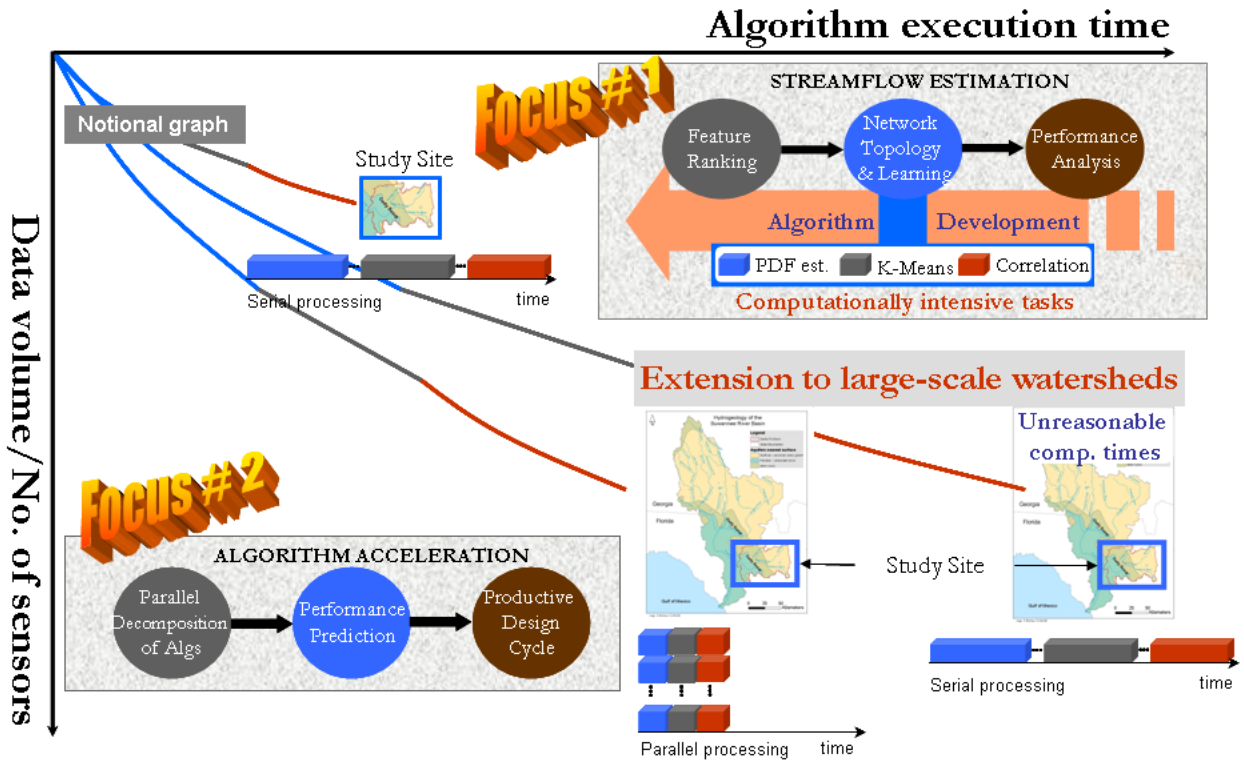


Figure 1-2. Research concept diagram.

Note: The key algorithms used in the estimation framework are PDF estimation, K-means clustering and correlation (color-coded as blue, gray, and red). While applying the framework to the SFW (upper left) results in manageable computational times, extending the framework to the Suwannee Basin (lower right) leads to unreasonable times due to high data volume. Accelerating algorithm execution via parallel processing is a viable solution to reduce overall computational time.

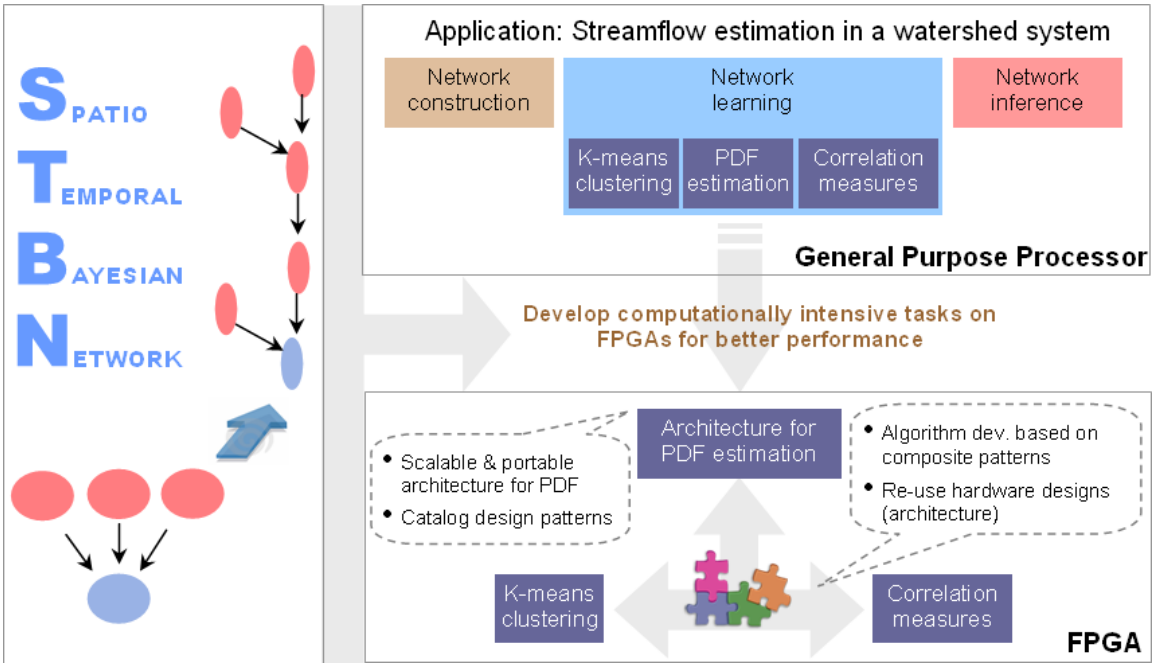


Figure 1-3. Research roadmap.

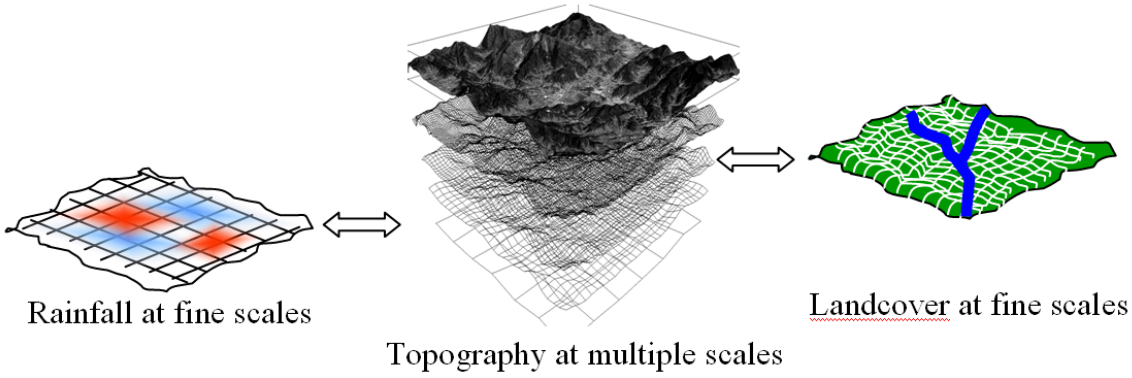


Figure 1-4. Availability of measurements at multiple scales.

CHAPTER 2
BACKGROUND

2-1 Data Fusion Methodologies in Hydrology

Flow rates in rivers and major streams are governed by a complex set of physical parameters. The subset of those parameters that are measured often comprise a wide range of scales and uncertainties, making fusion of the data more difficult. In addition, some data are strictly temporal (e.g., time series of measured flow at a particular location), others are spatial (e.g., digital maps of soil type and topography), while still others are spatio-temporal (e.g., estimates of distributed daily rainfall from weather radars). To date, most of the work on streamflow estimation has been handled by using deterministic models of hydrologic processes that attempt to parameterize the functional relationships between quantities, such as rainfall, surface topography, antecedent soil moisture, and stream discharge [15, 16]. These physically-based approaches, though appealing, are inherently limited because it is typically impossible to specify all the necessary boundary conditions at all times and locations with reasonable accuracy. Thus, such approaches often necessitate unreasonable assumptions of uniformity on various hydrologic parameters over the watershed.

Statistical methods, such as regression, have been extensively used in conjunction with physically-based models in an effort to account for parameter uncertainty. However, such approaches often deal only with second-order moments (variance) and also impose calibration requirements for their extrapolation to new locations or conditions. In [17, 18], the authors identify and propose strategies for improved site calibration using statistical methods. In [19], the authors support the claim that many existing physical models assisted by statistical approaches to parameter estimation can potentially be too site-specific or globally-common. Multi-resolution data fusion techniques have been applied to assimilate information from

multiple images [20]. Many investigators have developed specialized methods for fusing low- and high-resolution multispectral and hyperspectral images of the terrain [21, 22]. Most methods developed for fusing image data do not integrate point-source and distributed time-series data efficiently, yet many hydrologic modeling problems require both image and time series data. The ensemble Kalman Filter, first applied to ocean and atmospheric modeling in the mid 1990s, has been more recently applied to problems in hydrology and soil science [23, 24] to accommodate such mixed-mode data. These methods, however, can be suboptimal when the data and underlying processes are not Gaussian distributed, which is often the case.

Attempts have been made to fuse data from multiple sources using Bayesian approaches to accommodate non-Gaussian distributions [25, 26, 27]. A hierarchical approach based on Bayesian inference was proposed in [19] to estimate parameters of a physical model developed for predicting oxygen demand in estuaries. While the work assumed parametric models for fitting the underlying data, the target variable was estimated based on non-parametric formulations. In theory, non-parametric Bayesian inference yields optimal estimates (minimum probability of error) because it utilizes the joint (multivariate) PDF rather than merely a finite subset of its statistical moments. However, estimating high dimensional PDFs is computationally very difficult due to the well known phenomenon known as the "curse of dimensionality" [28]. Information theory has been employed in many areas to effectively handle large datasets. Authors in [29] address the "curse of dimensionality" issue in probabilistic methods by selecting a subset of features based on an information-theoretic approach. However, this eliminates information that might help to characterize the parameter being estimated. Another drawback to the method in [29] is that, in its current formulation, the estimation problem is solved either strictly in the temporal domain or spatial domain.

Formulating Bayesian inference problems on graphical models, such as Bayesian networks (BNs), is known to ameliorate the computational complexity issue. Existing BN approaches applied to hydrology problems are based on simple datasets [30] applied at different spatial locations [31] and do not traditionally capture the impact of spatial nonstationarities on response variables. Further, they have been primarily limited to addressing qualitative issues like landscape quality (for pleasure and leisure) and ecological value (species and biodiversity) [32], compliance violations at water treatment plants [33], and impact of pollutant transfer [34]. The authors in [27] present a Bayesian spatio-temporal dynamic model for determining ocean surface winds by fusing multi-resolution datasets in a hierarchical manner. Incorporating a complex spatio-temporal covariance structure in the model captures the spatial nonstationarity in ocean wind fields.

However, as indicated earlier, estimation of hydrologic parameters that exhibit complex (non-linear) spatial and temporal dependencies on multi-modal measurements poses a significant challenge. Unlike most data fusion applications explored to date where the data samples all come from a single "family" of data, the measurements from which the conditional PDFs (likelihoods) must be learned include spatial, temporal, and spatio-temporal measurements that are often sparse. Thus new strategies for designing the graph structure that incorporate knowledge of the physical processes are required.

In this work, a non-parametric BN methodology is proposed that incorporates both remotely-sensed spatial and *in situ* temporal data (features) extracted from a watershed to predict streamflow with quantified uncertainty. BNs are appropriate to model the intuitive understanding of physical hydrologic processes as simple causal relations quantified by probability densities. They also provide capabilities to incorporate physical-model (i.e., models

calibrated for the watershed under study) knowledge by adding them as nodes in the network topology. This can potentially lead to improved performance. Even though the method maintains manageable computational complexity while still allowing many features to be considered, it is desirable to quantify the usefulness of features towards estimation. In this work, information-theoretic concepts are employed to rank features (added as nodes to the BN) based on the information they contribute. The proposed methodology is readily scalable and generalizable to other monitored watersheds because the primary prediction parameters (likelihoods in this case) are learned from the data itself. Posterior probabilities of streamflow estimates and entropy values provide valuable information in determining sites for future instrumentation.

2-2 Study Site and Data Description

The study site for this work is the Santa Fe River watershed (SFW) in North Central Florida, USA (Figure 2-1). It shares many climatic, hydrologic, geologic, and land use characteristics of the larger Suwannee River basin of which it is a part but at a smaller scale (i.e., 3700 km^2). It is thus an ideal test bed for examining scientific and engineering questions pertinent to the larger Suwannee basin. The SFW exhibits large variations in the relationships between rainfall and streamflow as a function of location due to changes in draining (contributing) area, soil type, geology, and vegetation ranging from farm land to forests. The SFW also experiences dramatic variations in the amount and frequency of precipitation, exhibiting dry conditions from late winter to early spring, convective and tropical storms during the summer and early fall with high rainfall rates, and frontal systems in the late fall through late winter with low rainfall rates. The region commonly experiences extreme events (e.g., flooding from tropical storms and droughts associated with the North Atlantic multi-decadal oscillation).

The most distinguishing feature of the SFW, however, is the fact that the river traverses a major hydrogeologic divide – the Cody Escarpment – dividing the watershed into two major regions. The Cody Escarpment is the erosional edge of the confining unit, which includes clay-rich Hawthorn Group rocks. On the upstream side of this boundary, the Hawthorn clay acts as a confining layer retarding the movement of surface water into the underlying Floridan Aquifer. Thus, a dense network of streams and rivers formed in this region transport surface water to the outlet of the watershed. On the downstream side of the boundary, the clay layer is absent, resulting in karst terrain ground water-discharges to the surface through numerous springs. As the Santa Fe River crosses the boundary, it exhibits significant transitions in the manner and magnitude of surface and ground water interactions, with important implications for the fate and transport of water and pollutants [35, 36]. The SFW therefore serves as a good test site on which to develop hydrologic estimators because these varied characteristics cause it to exhibit relationships that can be found at many other watersheds. However, the complexity also implies that many of the relationships between rainfall and streamflow are non-Gaussian and non-stationary in space and time.

2-2-1 Observed Data

Topography data from the USGS National Elevation Dataset (NED) [37] was used to determine the region of the watershed area that drains rainfall and runoff to the particular location in the stream where streamflow must be predicted, referred to as the "contributing area" for that location. Streamflow measurements were obtained from the USGS NWIS database for locations indicated in Figure 2-1. Ground water levels (GWL) from monitoring wells (Figure 2-1) were provided by the Suwannee River Water Management District (SRWMD) at sparse locations. These values were then linearly interpolated spatially to obtain ground water levels ($G_{x,y}$) at all points across the watershed. Rainfall estimates ($R_{x,y}$) are obtained at moderate spatial

resolution (260m) and high temporal resolution (daily) from the National Weather Service NexRAD database for February 2001 through December 2006. Wetland classes and coverage (land cover) were provided by the National Wetland Inventory (NWI) and underlying soil characteristics (soil drainage) was obtained from the Soil Survey Geographic (SSURGO) Database at 260m resolution.

2-2-2 Derived Data

The Watershed Assessment Model (WAM) is a watershed scale model (Soil and Water Engineering Technology, Inc. 2008) that predicts stream flow as well as the transport and transformation of particulate and soluble phosphorus, particulate and soluble nitrogen, total suspended solids and biochemical oxygen demand in the stream system. The conceptual hydrologic model underlying WAM includes rainfall, evapotranspiration, overland flow, groundwater flow and river flow. The model takes Geographic Information System (GIS) layers for landuse, soils, topography, streamflow, basin boundaries, and meteorological data such as rainfall as inputs. It uses a GIS raster or grid cell representation of a watershed. Based on soils, topography and landuse in each grid cell, the GLEAMS field-scale model [41] is run to generate overland and groundwater flow (and associated water quality constituents) produced from each grid cell on a daily basis. The model delivers the daily overland and groundwater flows and constituents to the nearest down gradient stream based on predetermined flow paths and empirical flow velocities and hydrographs that are different for surface water and groundwater, but are assumed constant over the modeled domain in both space and time. There is no cell-to-cell interaction as the water and its constituents are routed to the stream.

Complex physically-based models, such as WAM, require very specific input data and are usually employed for long-term scenario analyses. When applied to watersheds where large amounts of input data are available for calibration, the WAM model can perform well for

ranking of alternative future landuse and water management practices, but typically is not accurate enough for streamflow forecasting. However, a framework that utilizes information from physical models, whenever available, is desirable. In this work, we analyze the utility of WAM for improving real-time streamflow networks by incorporating it as a node in the Bayesian network. In particular, streamflow predictions from the WAM model configured for the SFW by Srivastava et al [41] streamflow predictions were evaluated for incorporation into the Bayesian Network. WAM predictions of streamflow concentrations for nutrients, sediments and BOD were not utilized in this study.

Predictions from a calibrated hydrologic model are not always available and therefore we also analyzed the utility of using a simpler estimate of surface runoff using the SCS Curve number method as a feature in the estimation framework. This is more computationally efficient than considering land cover, soil drainage, and antecedent moisture as individual features in the estimation framework. The Soil Conservation Service's (SCS) Runoff Curve Number (Equation 2-1) method [38] was employed to generate daily runoff contributions ($Q_{x,y}$) across the contributing area at point (x,y) where I_a=0.2×s, s=(1000/$C_{x,y}$)-10.

$$Q_{x,y} = \begin{cases} \dfrac{(R_{x,y} - I_a)^2}{R_{x,y} - I_a + S} & if \;\; R_{x,y} > I_a \\ 0 & otherwise \end{cases} \tag{2-1}$$

$C_{x,y}$ and $R_{x,y}$ are the curve number and current day's rainfall at location (x.y) in the watershed, respectively. Wetland classes, land cover and underlying soil characteristics (soil drainage) were employed to estimate $C_{x,y}$. Since curve number is known to vary with time due to fluctuations in antecedent soil moisture, commonly used adjustments for curve number under dry and wet antecedent moisture conditions (which can be found in the hydrologic literature, such as [39, 40]) were used. Antecedent moisture condition for a particular day was determined by

accumulating rainfall from the preceding 5 days (R_A). Further, flow at the outlet station is affected by recent runoffs (through rainfall events) proximal to that point and also by earlier events at points farther upstream. Time dependencies between rainfall and streamflow were inferred from second-order statistics and are used to determine lags ($l_{x,y}$) experienced by a surface flow component originating at point (x,y) to reach the watershed outlet. Peak temporal cross-correlation times (i.e., lag at which cross-correlation is maximum) between every rainfall pixel within the contributing area ($R_{x,y}$) and flow at the outlet were computed as the lag ($l_{x,y}$) corresponding to that pixel. Runoffs ($Q_{x,y}$) over different portions of the basin are grouped by their corresponding time lags ($l_{x,y}$) to generate net runoff (Q_{net}) for the contributing area (Equation 2-2).

$$Q_{net}(t) = \sum_{l_{x,y}} Q_{x,y}(t - l_{x,y}) \tag{2-2}$$

Table 2-1 summarizes the potential inputs for the Bayesian models utilized in this work. The observed features are generally widely available for watersheds in the United States and in Europe and are known to strongly impact streamflow. The derived features are also relatively easy to estimate for most watersheds in the United State and Europe. All data were obtained through the University of Florida Water Institute [41].

2-3 Hardware Acceleration on FPGAs

The three algorithms (non-parametric PDF estimation, K-means clustering, and correlation) targeted for hardware acceleration, in addition to being an integral part of the framework proposed in this work, form an important core in many applications. The Parzen window-based non-parametric PDF estimation algorithm is used for many applications in pattern recognition, such as fingerprint recognition, Bayesian classification, feature extraction [7], bioinformatics [5], networking [42], stock market prediction [43], and image processing [6]. The

authors in [9] employ cross-modal correlations for clustering features in an image-audio dataset. Clustering and feature extraction methods are applied for video content retrieval and analysis in [44]. A novel video segmentation technique was presented in [4] based on correlating audio and video data. A simplified probability-based methodology using histograms was proposed in [8] for video analysis. Hence, a large number of applications can benefit from fast computations of PDF estimation, K-means clustering and correlation, especially in time-critical missions.

FPGAs lie between general-purpose processors and ASICs on the spectrum of processing elements in that they are highly flexible (like processors) and also have potential for high performance (hardware acceleration). Commonly used deterministic benchmarking cores, such as Fast Fourier Transforms (FFTs) [45], convolution, Lower-Upper triangular matrix Decomposition (LUD) [46], and Basic Linear Algebra Subprograms (BLAS), have been effectively implemented on FPGAs by virtue of their inherent multi-level functional and data parallelism. In addition to these benchmarking cores, significant performance gains have been achieved with FPGAs by selecting, developing, and testing diverse applications from various fields in signal and image processing. Previous works have also discussed the feasibility of targeting some structurally similar algorithms for FPGAs. In [47], the authors use a Gaussian kernel estimator as one of their case studies in presenting a MATLAB-to-VHDL application mapper. It is a subset of the algorithm considered here, in that the streamflow estimation framework employs Gaussian kernels to estimate PDFs and is hence more computationally intensive. In [48], the authors present simpler algorithmic variants of the K-means algorithm by considering Manhattan and Max distances instead of Euclidean distance for an efficient hardware implementation. Significant acceleration of the algorithm with acceptable accuracy was achieved in clustering a sample dataset. This work is different from [48] in that Euclidean

distance is used for implementing K-means (which is the preferred method in most machine-learning applications) and, moreover, the primary motivation is to accelerate a set of machine-learning algorithms used in different stages of estimation by reusing hardware designs leading to shorter design times. Previous work in parallel processing for large data volumes is also relevant to this work. An object-recognition application was developed in [49] by traversing huge volumes of data in a parallel fashion. The paper showcased how a data-parallel programming model can be used to solve a general set of data-intensive algorithms. In many of these examples, the applications could achieve significant performance gains if there was a way to rapidly compute multi-dimensional PDFs. PDF estimation enables the computation of metrics such as noise, error rates, and uncertainties that are very important for researchers in the application domain [5,6,7,50]. The authors in [51] and [52] had developed several signal-processing applications on FPGAs in order to obtain speedup and discussed tradeoffs in solution accuracy and resource utilization. However, these papers only predict performance factors in a relatively limited setting. Various performance prediction tools have been proposed [53, 54] that base their techniques on parameterizing the algorithm and target FPGA platform. Algorithms are decomposed and analyzed to determine their total size (in terms of resources) and computational density. Computational platforms are primarily characterized by their memory structure, capacity, and interconnect bandwidth. In particular, the RC Amenability Test (RAT) presented in [53] is a simple methodology that suggests a step-by-step procedure to predict the performance, in terms of speedup, of a specific design for an application on a specific platform before coding begins. This pre-implementation test also helps the designer to understand the strengths and weaknesses of a particular algorithm towards parallel implementation. Communication and computation parameters in RAT are quantified based upon algorithm and

FPGA platform characteristics. An estimate of performance in terms of execution time in hardware is then derived from the analytical models embedded in RAT. Comparing the hardware prediction against a known execution time for a software baseline leads to the overall speedup estimation. In this work, we use RAT to predict performance of an algorithm on multiple FPGA platforms. Building a good reconfigurable design requires skill and effort, and is often accompanied by a steep learning curve for designers without prior FPGA design experience. In [55], the authors emphasize the importance of identifying and cataloging an exhaustive list of design patterns to solve recurring challenges in RC and increase productivity. In this work, the utility of design patterns is validated by identifying and classifying relevant primitive patterns. The patterns are then quantified and applied for algorithm decomposition and performance prediction using RAT.

2-4 Case-study Algorithms

Knowledge of the algorithm's data flow and computational complexity is essential in order to make strategic decisions during FPGA design development. An overview of the algorithms used as case studies in this work is presented and commonalities in their structures are investigated in this section.

2-4-1 Probability Density Function Estimation

The common parametric forms of PDFs (e.g., Gaussian, Binomial, Rayleigh distributions) represent mathematical idealizations and, as such, are often not well matched to densities encountered in practice. In particular, most of these classical parametric densities are unimodal (having a single local maximum), whereas many practical problems involve multimodal densities. Furthermore, because of correlations among the data features, it is unusual for high-dimensional density functions to be accurately represented as a simple product of one-dimensional densities, the latter of which are significantly easier to compute. Thus, an estimate

of the joint (multi-dimensional), non-parametric PDF (i.e., a PDF that assumes no particular functional form) is often required. The computational complexity of the Parzen window-based PDF estimation algorithm is $O(Nn^d)$, where N is the total number of data points, n is the number of discrete points at which the PDF along a dimension is estimated (i.e., bins) and d is the number of dimensions. Mathematically, the probability that point i falls in a d-dimensional space is given by Equation 2-3 where h is the bin size, which acts as a tuning parameter for resolution of the PDF estimate, the set $(x^j,\ldots y^j)$ represents the d subsets of bin centers at which the PDF is estimated, and the set $(x_i,\ldots,y_i)$ represents the i^{th} input data point in a d-dimensional space, where i ranges from 1 to N.

$$p(i) = \frac{1}{n_1 \ldots n_d} \sum_{j_1=1}^{n_1} \ldots \sum_{j_d=1}^{n_d} \varphi(x_i, x^{j_1}, h) \ldots \varphi(y_i, y^{j_d}, h) \tag{2-3}$$

The kernel function φ could be as simple as a histogram function, a more general rectangular kernel, or the widely favored Gaussian kernel. The first two cases fall under the class of naïve estimators. In the first case (histogram function), the data range is divided into a set of successive and non-overlapping intervals (bins), and the frequencies of occurrence in the bins are plotted against the discretized data range. The rectangular kernel case is similar except that overlapping intervals are permitted. In either case, the bin size should be chosen such that a sufficient number of observations fall into each bin. The resulting PDF estimate depends on the bin size as well as the discretized range of the dataset and is discontinuous at the bin boundaries.

Although naïve estimators yield discontinuous results, the construction can be easily generalized to achieve continuous PDF estimates by employing different kernel functions. The smooth reproduction of the underlying Gaussian process in Figure 2-2A clearly shows the advantages of Gaussian kernels in this aspect and is thus the motivation for its use here. The

mathematical formula for a Gaussian kernel and the resulting expression for the 1-D Parzen window PDF estimate are defined in Equations 2-4 and 2-5 respectively where *h* plays the role of the standard deviation.

$$\varphi(x_i, x^j, h) = \frac{1}{\sqrt{2\pi}h} e^{-\frac{(x_i - x^j)^2}{2h^2}} \tag{2-4}$$

$$p(x^j) = \frac{1}{\sqrt{2\pi}hn} \sum_{j=1}^{n} e^{-\frac{(x_i - x^j)^2}{2h^2}} \tag{2-5}$$

Computing complex exponentials is challenging in hardware because it requires significant hardware resources. To make the algorithm more suitable for hardware, a truncated second-order Taylor series expansion replaces the exponential function for the development of the core in hardware (Equation 2-6).

$$\phi(x_i, x^j) = \begin{cases} \frac{1}{\sqrt{2\pi}h}\left(1 - \frac{(x_i - x^j)^2}{2h^2}\right), & \text{for } \left(1 - \frac{(x_i - x^j)^2}{2h^2}\right) \geq 0 \\ 0, & \text{otherwise} \end{cases} \tag{2-6}$$

The degree to which the quadratic (Equation 2-6) approximates the true Gaussian kernel (Equation 2-4) decreases for data points that yield large values of $|x_i - x^j|$. That condition is avoided by giving zero weight to points that cause the argument $1-(x_i-x^j)^2/2h^2$ to be less than zero, which occurs when $|x_i - x^j|$ is greater than $\sqrt{2}h$. The plot in Figure 2-2A for the Gaussian kernel employs the approximation given by Equation 2-6. Additional management of approximation error is achieved by pre-scaling the kernel to unity variance prior to computations on the FPGA. The datasets x_i and x^j are scaled by $\sqrt{2}\,h$ before being transferred from the processor to the FPGA, which reduces the factor $1-(x_i-x^j)^2/2h^2$ to $1-(x_i-x^j)^2$, thus making the computational error due to the use of the Taylor series representation on the FPGA independent of the particular choice of *h*.

The dynamic range of the dataset can further be reduced by considering only the higher-order bits during computation. This method is a simple task to implement since FPGAs are extremely efficient at performing bit-level manipulations.

2-4-2 K-Means Clustering

In data analysis problems, given a training set of points, unsupervised learning algorithms are often applied to extract structure from them. Typical examples of unsupervised learning tasks include the problem of image and text segmentation [3]. A simple way to represent data is to specify similarity between pair of objects. If two objects share the same structure, it should be possible to reproduce the data from the same prototype. This idea underlies clustering methods that form a rich subclass of unsupervised algorithms.

The K-means algorithm is a popular unsupervised clustering algorithm that partitions N data points into m clusters by finding m mean vectors $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_m$ representing the centroids of the clusters (Figure 2-2B shows an illustration of data points represented by two features partitioned into 4 clusters). It is an iterative algorithm that tries to minimize the total intra-cluster variance or the squared error function (Equation 2-7) where there are m clusters $\boldsymbol{S}_j$, $j = 1, 2, \ldots, m$ and $\boldsymbol{\mu}_i$ is the centroid of all the points x_i in $\boldsymbol{S}_j$.

$$V = \sum_{j=1}^{m} \sum_{x_i \in S_j} \left(x_i - \mu_j \right)^2 \tag{2-7}$$

To begin with, the data points are partitioned into m initial sets, either at random or using some heuristic approach. The centroid of each set is then computed and a new partition is constructed by associating each data point with the closest centroid. The centroids are then recalculated for the new clusters. This process is repeated until data points no longer switch

clusters or alternatively centroids remain unchanged. The computational complexity of the algorithm is O(*NdmT*) where *d* is the number of features representing the data point and *T* is the number of iterations.

2-4-3 Correlation

Correlation computes a measure of similarity between two datasets (e.g., audio signals) as they are shifted by one another across time. Correlation is often used to measure the delay between two signals having the same shape with one delayed relative to the other. Autocorrelation is the correlation of a signal with itself while cross-correlation is the correlation between two different signals. The correlation result reaches a maximum at the time when the two signals match best. For example, if the two signals are identical, this maximum is reached at $t = 0$ (i.e., no delay). Autocorrelation is useful for finding repeating patterns in a signal, such as determining the presence of a periodic signal that has been buried under noise, or identifying the fundamental frequency of a signal that does not actually contain the fundamental frequency component but implies it with many harmonic frequencies. Mathematically, the correlation between two discrete signals x_i and y_i with n samples each, where τ is the delay and m is the maximum delay is given in Equation 2-8.

$$R(\tau) = \sum_{i=1}^{n} x_i y_{i+\tau};\ 0 \le \tau \le m;\ m < n \tag{2-8}$$

2-4-4 Algorithm Similarity

Although the algorithms overviewed in the previous sections address different problems, involve different computations, and appear at different stages during data analysis, the underlying data flow within each algorithm has significant similarity. The data flow in these algorithms follows an exhaustive data permutation pattern wherein the interaction between every sample in two input vectors I_1 and I_2 affects the value of every sample in an output vector O. In

PDF estimation, it is between the data points (I_1) and the points at which the PDF is estimated (I_2). In K-means clustering it is the Euclidean distance between the data points (I_1) and the cluster centers (I_2), and in correlation, it is the multiplication of two signals (I_1 and I_2) shifted over different time delays. Figure 2-3 illustrates the dataflow and computations involved in each of the three algorithms in a common framework.

Table 2-1. Set of available features

Feature	Notation	Resolution	Type
Streamflow at neighboring station	TNN	Daily	Temporal
Ground water level	GWL	Daily	Temporal
Ground water level	$G_{x,y}$	260m	Spatial
Rainfall	R	Daily	Temporal
Rainfall	$R_{x,y}$	260m	Spatial
Runoff	Q_{net}	Daily	Temporal
Runoff	$Q_{x,y}$	260m	Spatial
WAM estimates	WAM	Daily	Temporal

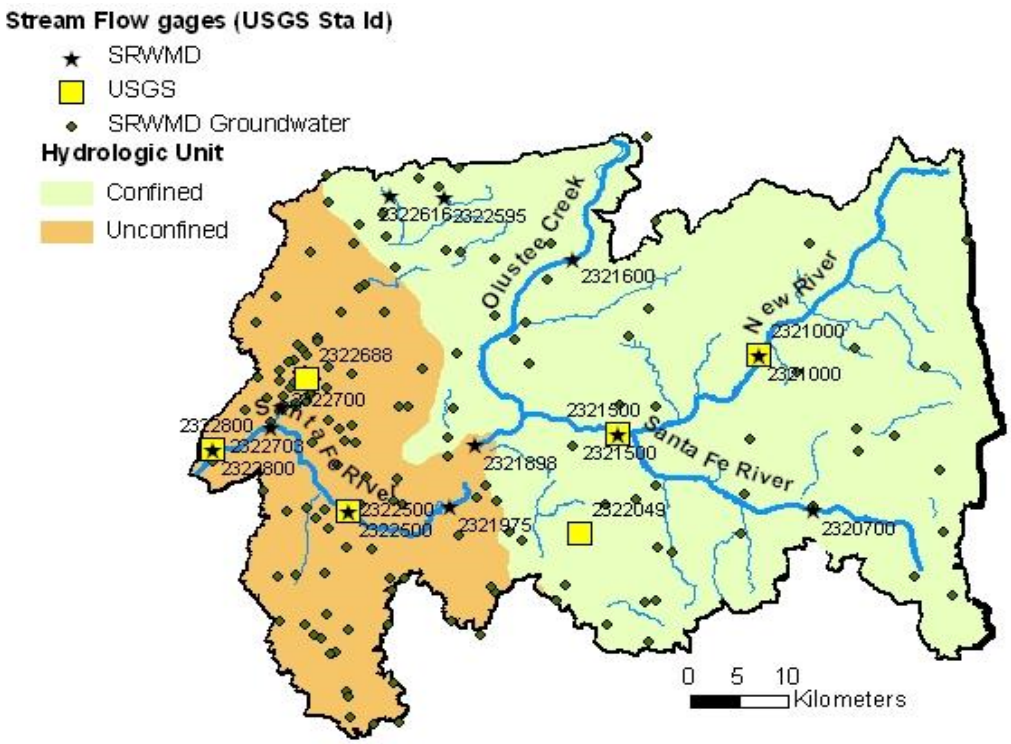


Figure 2-1. A map of the Santa Fe River Watershed with its streams and rivers [41].

Note: Two distinct hydrogeologic regimes are present: the confined eastern region and the unconfined western region. Locations of streamflow measuring stations are indicated.

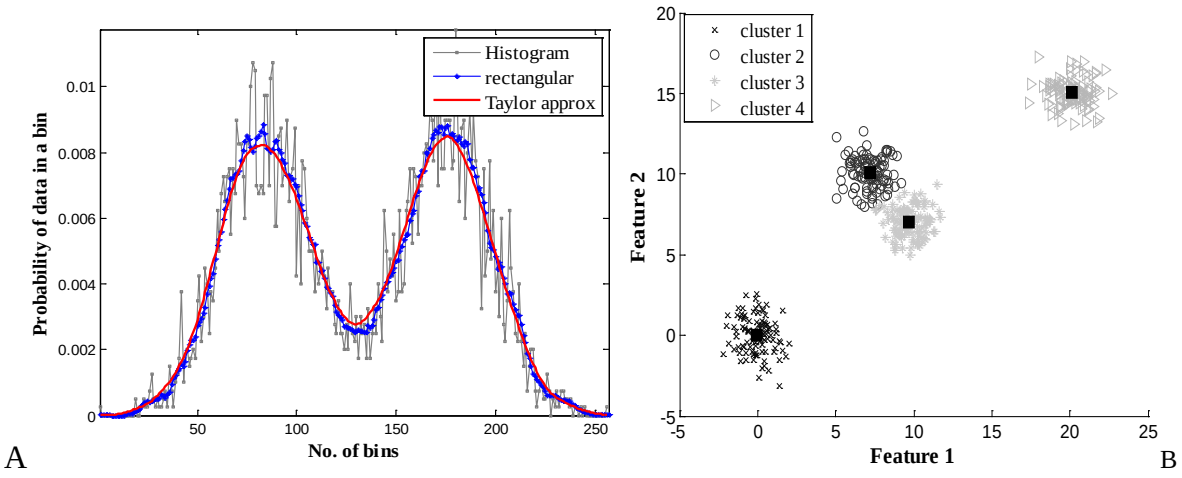


Figure 2-2. Illustration of case-study algorithms. A) 1-D PDF estimation using three different kernels. B) K-means clustering.

Note: For the Gaussian kernel, the Taylor series approximation is used. In general, PDFs can be highly non-Gaussian (e.g., bimodal) as shown here. Parzen estimates using rectangular, and in particular Gaussian kernels, generally provide estimates of the true underlying PDF that are far superior to the histogram function. In the case of data clustering, it is desired to partition complex data sets in groups that share similar feature values.

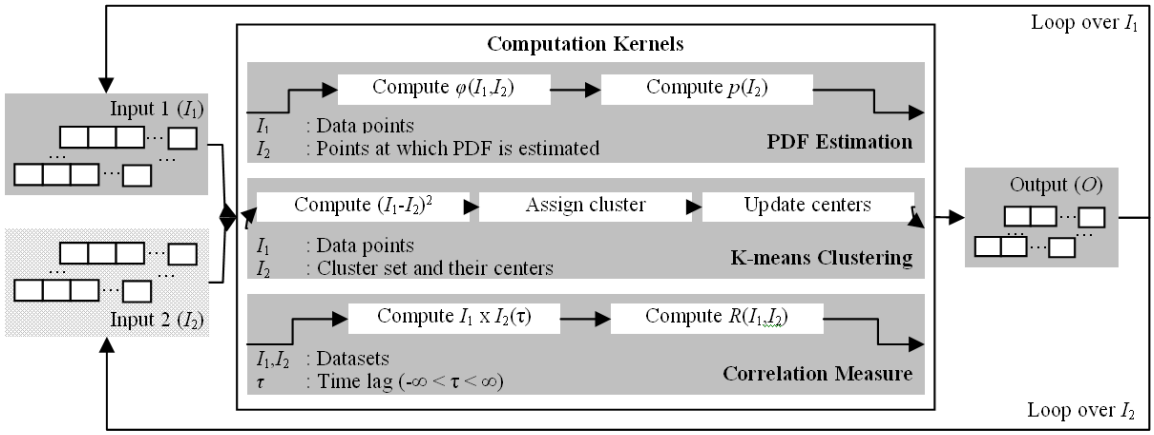


Figure 2-3. Dataflow and computational structure in PDF estimation, K-means clustering, and Correlation.

CHAPTER 3
SPATIO-TEMPORAL BAYESIAN NETWORK

Probabilistic methods that utilize the joint (multivariate) PDF of all the features and parameters rather than merely a finite subset of its statistical moments can be used to incorporate (fuse) different data sets in order to estimate streamflow. Bayesian estimation incorporates probabilities based on multiple features that influence flow. The many-to-one mapping of a $d\times1$ feature vector x to streamflow constitutes a probabilistic data fusion problem, where d denotes the number of data types used as features (Table 2-1). A discrete formulation of Bayes' rule is used in Equation 3-1 for which streamflow is segmented into fifteen levels or "classes" c_i, i=1,…,15. The ranges of stream flow magnitudes that correspond to these classes are independently determined using k-means clustering for the three long-term flow monitoring stations (2321000, 2321500, 2322500) in the SFW. Fifteen levels (k=15) were determined to give a reasonable trade off between precision of the estimated flow and maintaining a sufficient number of samples in each class for reliable estimation of the likelihoods via Parzen windowing. The likelihoods $p(x|c_i)$ of the training flow data are thus estimated in feature space for each flow class. Prior probabilities $p(c_i)$ for the classes are computed based on class memberships obtained from k-means clustering. The *a posteriori* probabilities $p(c_i|x)$ are then computed based on these PDFs for each test data point and the flow level is labeled according to the class with the maximum *a posteriori* (MAP) probability.

$$p(c_i\,|x) = \frac{p(x|c_i)\,p(c_i)}{p(x)},\ p(x) = \sum_{i=1}^{k} p(x|c_i) \tag{3-1}$$

In theory, this direct application of Bayes' rule is optimal with respect to minimizing the probability of error of misclassification. Since it is not dependent on physically-based equations, this method does not require as much *a priori* knowledge of the particular hydrologic processes

that dominate in a given watershed as do physical models such as WAM. In that sense, it is more easily applied to different watersheds where different combinations of input data may be available. Of course, the accuracy depends on the particular data types available. In general, the more features available, the better. The critical limitation with this approach arises when there are many features, forcing the estimation of the joint likelihoods into a high-dimensional space. Computational complexity and memory requirements increase super-linearly with dimension [56] making it often impractical to implement a direct Bayesian classification for joint PDFs of greater than three dimensions over large areas and long time periods. While it is possible to decompose Equation 3-1 from a *d*-dimensional problem into *d* 1-dimensional problems to ameliorate the memory demands, this so-called naïve Bayesian implementation is far from optimal because it ignores all statistical relationships among the features. Some data-driven methods (the Reduced-Dimension Bayesian Estimate – RDBE) mitigate the increase in computational complexity by using a reduced feature data set, thus tolerating greater error [29]. BNs offer a formal method for decomposing high-dimensional likelihoods into lower-dimensional factors by invoking the well known conditional probability theorem [57, 58], thereby reducing computational complexity. A comparison of the computational complexity involved in a traditional Bayesian classification, RDBE-based and BN-based methods is given in Table 3-1. It is also desirable to use a reduced set containing only the most informative features. A method determining the information gained by incorporating a new feature in the network is hence needed (Section 3.3). Such a methodology would also offer a platform to analyze the potential benefit of including physical model estimates as nodes in the BN. Further, well-informed tradeoffs can be made regarding computational complexity and accuracy.

Hidden Markov models (HMMs) are also a frequently used tool for modeling time series data that belongs to a subset of the more general BNs. It extends the concept of discrete markov models in that the observation is a probabilistic function of the state (i.e., streamflow). The resulting model is a doubly embedded stochastic process with an underlying stochastic process that is not directly observable; it is *hidden*. The *hidden* states are assumed to be discrete and can only be observed through another set of stochastic processes that produce the sequence of observations. The advantage of the BN framework over HMMs is that it allows for an arbitrary set of *hidden* variables with arbitrary independence assumptions. If the conditional independence assumption results in a sparse network, this may result in an exponential decrease in the number of parameters required to represent a probability distribution. Often there is a corresponding decrease in the computational load as well. Hence BNs offer better statistical and computational efficiency.

3-1 Bayesian Network Fundamentals

A Bayesian network is a Directed Acyclic Graphical (DAG) model where nodes represent random variables (RVs) and arcs between nodes specify the conditional dependencies between RVs. If there is an arc from node *A* to another node *B*, *A* is called a parent of *B* and *B* is a child of *A*. The set of parent nodes of a node X_i is denoted by parents(X_i). A directed acyclic graph is a BN relative to a set of variables if the joint distribution of the node values can be written as the product of the local distributions of each node and its parents [58] (Equation 3-2).

$$P(X_1, X_2, ..., X_N) = \prod_{i=1}^{n} P(X_i | parents(X_i)) \tag{3-2}$$

If node X_i has no parents, its local probability distribution is said to be unconditional, otherwise it is conditional. If the value of a node is observed, then the node is said to be an evidence node. The graph encodes independencies between variables. Conditional independence

is represented by the graphical property of *d*-separation: If two sets of nodes *X* and *Y* are *d*-separated in the graph by a third set *Z*, then the corresponding variable sets *X* and *Y* are independent given the variables in *Z*. The minimal set of nodes which *d*-separates node *X* from all other nodes is given by *X*'s Markov blanket. *d*-separation is defined as follows. A path *p* is said to be *d*-separated (or blocked) by a set of nodes *Z* if and only if,

- Path contains a chain $p \rightarrow m \rightarrow j$ or a fork $i \leftarrow m \rightarrow j$ such that the middle node *m* is in *Z*.
- Path contains an inverted fork (or collider) $i \rightarrow m \leftarrow j$ such that the middle node *m* is not in *Z* and no descendant of *m* is in *Z*.

A set *Z* is said to *d*-separate *x* from *y* in a directed acyclic graph *G* if all paths from *x* to *y* in *G* are *d*-separated by *Z*. The '*d*' in *d*-separation stands for 'directional', since the behavior of a three node link on a path depends on the direction of the arrows in the link. This property offers a formal method for decomposing high-dimensional likelihoods into lower-dimensional factors by invoking the well known conditional probability theorem and the effects of causality. The fundamental connections in a BN that would be used in this work and commonly found in literature are illustrated in Figure 3-1. In most of the applications where BNs are employed, the nodes are either qualitative representations of some random phenomenon/occurrence or a quantitative measure of a RV observed in a particular domain (time or space but not both). A BN framework that offers the capability to embed both spatial and temporal RVs in proposed in this work as illustrated in Figure 3-2.

- The set of immediate neighbor nodes of any node *A* should be observed in the same domain as that of *A*.
- Domain transformation is obtained through *aggregate* functions.
    - TYPE I. Simple statistical function (e.g., sum, minimum/maximum, mean) values.
    - TYPE II. Physical model-based mappings depending on the application.
    - TYPE III. Probability distribution-based transformations.

In the sections to follow, the proposed BN framework is applied to the problem of estimating streamflow in the Santa Fe watershed system. The application, in addition to serving as a proof-of-concept, would also help bring out some of the advantages in choosing a BN formulation.

3-1-1 Network Learning

Probabilistic methods can readily be applied to different monitored watersheds because they rely on learning the parameter PDFs from the available data in a straightforward manner using methods such as Parzen windowing [59]. As with any learning approach, the accuracy of the PDF estimates depends on how well the particular data sets sample the probability space. In general, time series data used for training should have daily time steps and be at least a few years in duration to capture annual variations in streamflow. Only three of the flow monitoring stations in the SFW (2321000, 2321500, 2322500) had continuous records of measured streamflow over the time period that the NexRAD data was available. Thus, these three stations were used to provide training data. An ideal scenario to estimate PDFs using parzen windows is the availability of infinite number of data samples (i.e., $n \rightarrow \infty$) and setting h to a significantly small value (i.e., $h_n \rightarrow 0$) in Equation 2-3. However, the application under study is characterized by historically short datasets requiring better strategies in the selection of a suitable h. For fixed n, small values of h imply higher variance leading to a noisy appearance of the resulting PDF estimate. On the other hand, large values of h represent an excessively smoothed version of the true underlying PDF. While there are approaches [60] that discuss optimal ways of selecting kernel sizes, they are either primarily targeted towards the representation of individual PDFs or are computationally very demanding [60]. The objective in this work is to be able to represent every possible sample of the feature vector x, within the range of the training set, with a streamflow class realized by at least one of the training stations (i.e., for all $i \epsilon [1,15]$ and

$j \rightarrow$ [2321000,2321500,2322500], there exists a $p(x|c_i,s_j) \neq 0$). To implement this, h is initialized to some small value and incremented gradually until the objective is satisfied. In general, it was observed that smaller kernel sizes are better able to capture short term rainfall and flow events and thus tend to work well for stations in the upstream confined regions of the SFW. Slightly larger kernels perform better for stations in the unconfined region where flow depends on longer transit times for rain falling over a larger contributing area to impact flow at the station.

3-1-2 Network Topology

Structure of the networks used for estimating streamflow is determined based on the physical knowledge of the underlying processes. The impact of nodes forming the BN is quantitatively analyzed later in Section 3-4. Runoff is a spatio-temporal process impacted by the joint interaction of spatial rainfall, land cover, and soil moisture (i.e., percentage of rainfall intercepted at a point (x,y) contributing to surface runoff is affected by the land cover and level of antecedent soil moisture at that point). Inclusion of SCS estimated runoff and WAM streamflow predictions as nodes in the BN is redundant since both attempt to capture the impact of spatial rainfall, landuse, and soil moisture on streamflow. However, estimates from a calibrated hydrologic model are not always available and so we analyze the performance of two BNs: one that excludes WAM (N1) and another that includes WAM (N2) as shown in Figure 3-3. In the absence of land cover and soil drainage measurements that characterize curve number ($C_{x,y}$), runoff values in network N1 cannot be directly computed using the SCS method. To address this issue, SCS estimated runoff (from training period) is conditioned and inferred from antecedent rainfall and spatial groundwater levels as these are known to primarily impact runoff and also more readily available. Note that groundwater level was used as a surrogate for soil moisture since it is more commonly measured. In N1, $\mathbf{Q_{x,y}}$ at a particular point on the terrain is conditioned on antecedent rainfall $\mathbf{R_A}$ (in this work, rainfall is accumulated over the preceding 5

days) and ground water level $\mathbf{G_{x,y}}$ at that point (handles spatial nonstationarity). Note that groundwater level was used as a surrogate for soil moisture since it is more commonly measured and thus more readily available. $\mathbf{Q_{net}}$ at time *t* is the sum of all runoffs $\mathbf{Q_{x,y}}$ over the basin as discussed in Section 2-2. Streamflow **F** at a current station is then conditioned on $\mathbf{Q_{net}}$ (or **WAM** in case of N2), streamflow at the station nearest to it (**TNN**), and ground water level at a well closest to the station (**GWL**). Conditional probability tables (CPTs) for all parent- and child-node connections in the networks are then computed in a straightforward manner. Computing these CPTs from the training data can be viewed as a learning step.

3-1-3 Network Inference

Given spatial rainfall data, spatial and temporal ground water level (or WAM estimates), and flow at a nearby station, the most probable $\mathbf{Q_{x,y}}$, $\mathbf{Q_{net}}$, and **F** values are inferred from the graph. The expected value of estimated flow on a particular day *d* is given in Equation 3-3 where $p(c_i)$ is the posterior for the i^{th} class, ω_i is the mean flow value for the i^{th} class, and $i \epsilon [1,15]$.

$$E(f) = \sum_{i=1}^{k} p(c_i)\omega_i \tag{3-3}$$

The classes and their corresponding ω_i values were determined by segmenting the training data using *k*-means clustering. The error in model prediction (e_{pred}) can be defined as the deviation of the maximum *a posteriori* flow from the expected flow (Equation 3-4). The actual error (e_{actual}) is defined as the deviation of observed flow from the expected flow (Equation 3-5). Reliability of network performance can be inferred by comparing actual errors and expected errors (suggested by the model).

$$e_{pred} = \sum_{i=1}^{k} p(c_i)\omega_i - p_{MAP}\omega_{MAP} = \sum_{i=1}^{k-1} p(c_i)\omega_i \quad \forall i \neq MAP \tag{3-4}$$

$$e_{actual} = \sum_{i=1}^{k} p(c_i)\omega_i - f_{obs} \tag{3-5}$$

3-1-4 Performance Metrics

In addition to probabilistic performance measures like error entropy (*H*) and maximum *a posteriori* probability, the widely used Nash-Sutcliffe coefficient (*NSC*) is employed as a "good ness-of-fit" measure to analyze performance [61] (Equation 3-6).

$$NSC = 1 - \frac{\sum_{i=1}^{N}(O_i - P_i)^2}{\sum_{i=1}^{N}(O_i - O_{mean})^2}; \qquad H = -\sum_{j} p(e_j)\log p(e_j) \tag{3-6}$$

In Equation 3-3, O_i is observed streamflow, P_i is predicted streamflow, O_{mean} is the mean of observed streamflow, and N is the total number of data points. Since *NSC* utilizes some form of deviation from the mean, it can be expected to provide reasonable indications of accuracy primarily when the underlying PDF is uni-modal and not heavily skewed. Therefore, additional metrics based on the signed error are also examined. The signed error is defined as predicted minus observed streamflow values, so that a positive error implies overestimation. The mean (bias) and standard deviation of the error are computed. Finally, the error PDF is estimated via the normalized Parzen windowing procedure described in Section 2-4-1, thus enabling the calculation of error entropy. A Gaussian kernel with size h=1 and N=128 was used in all cases to estimate the error PDFs.

3-2 Measurement of Incremental Information

When estimating streamflow at a location with no *in situ* data, it is important to know which features will contribute the most information to limit the dimensionality and complexity of the problem. One way to pick the most informative features is a system of feature ranking based on measures such as mutual information (MI) or conditional entropy (CH) amongst the flow classes. The feature combinations that provide the most separation among all the classes are considered best. The MI gives a quantitative way of measuring this separation. Given a feature vector x and class c, the best subset of x is the set that gives the largest MI between the feature PDFs and class-membership based on the training data. Another way to approach this problem is to minimize CH of streamflow classes with respect to the feature vector. The Shannon MI, expressed as $I(x;c)=H(x)-H(c|x)$, reveals a dependence on conditional entropy. When relatively few training samples are available for one or more classes, however, the estimate of $H(c)$ can become biased. Using the CH of streamflow class c given feature vector x avoids this dependence on the sole entropy of c in MI methods.

3-2-1 Feature Ranking

In the proposed approach, the feature nodes are ranked using CH and the performance of networks are compared using the suggested features. The hydrologic makeup of the study site is heterogeneous, making it difficult to find a single BN to accurately predict stream discharge at all locations in the watershed. Three USGS flow measurement stations in the watershed are used as test sites. Stations 2321000 and 2321500 are in the confined area, while station 2322500 is in the unconfined area. The feature ranking algorithm is applied separately to each of the three stations that aid in obtaining an optimal yet simple structure at each location. Temporal features in Table 2-1 are considered for this analysis (rainfall R was not included as its effect was embedded in WAM and Q_{net}).

First, the information metric (i.e., CH) is used to determine the single most informative feature at each of the test stations (rankings shown in "First Feature" rows of Table 3-2 with the best ranked feature in column "Rank 1"). The remaining features, each paired with the best feature from the previous step ("Second Feature" in Table 3-2), are ranked in the next step. This process is continued until either all features have been ranked or the required number of features has been ranked. Each time a feature is added to the network, the information metrics are computed for the entire network. The decrease in values along the columns in Table 3-2 indicates features with additional information are being added to the network, reducing CH.

The features under column "Rank 1" in Table 3-2 are those chosen as the best three features at each training station. Since streamflow was divided into 15 classes, the highest CH value of class separability is 2.7 (i.e., $log_e 15$). At stations 2321000 and 2321500, most of the information is contained in feature TNN. This is because these stations are located in the upstream confined area of the watershed where rainfall has an immediate and short-lived effect on discharge. In such cases where groundwater does not contribute significantly to streamflow, upstream flow is a very good indicator of flow at a downstream station. Adding groundwater and WAM contribute only a 1.4% reduction in CH, while TNN eliminates 41.6% of the uncertainty at 2321000. At 2321500, TNN and GWL eliminate 33.8% uncertainty and adding WAM eliminates only 0.5% more. Low CH values and smaller incremental information gains at these two stations indicate that a complex BN is probably not necessary.

Station 2322500 lies in the unconfined region, where groundwater is the dominant contributor to streamflow. Here, flow at station 2321500 was used as the TNN feature which lies in the confined region. Due to this relationship, TNN is not chosen as the best feature.

Streamflow in the unconfined region is affected by a number of variables making it hard for one feature to adequately characterize it (net reduction of 30.2% in CH values with 3 features).

3-2-2 Validating Information-Gain Analysis

In order to validate the results of information-gain analysis, performance of networks constructed from combinations of features were studied. The study would also warrant the use of CH as a metric for information analysis in the strategic construction of simpler networks leading to reduced but acceptable performance. The following set of scenarios was considered.

- CASE 1. Single-node network containing the rank 1 feature.
- CASE 2. Single-node network containing a feature that was not ranked best.
- CASE 3. Two-node network that contains the rank 1 feature.
- CASE 4. Two-node network containing features not ranked best.

Table 3-3 summarizes the results obtained for the case-study scenarios at each of the three training stations for the testing period. Low CH values and smaller incremental information gains at stations 2321000 and 2321500 indicate that a complex network is probably not necessary in confined regions for achieving satisfactory performance. Addition of TNN to WAM at 2322500 leads to significant improvements in performance supportive of the large reduction in CH values in Table 3-2. Performance metrics for Case 4 networks decreased dramatically at all three stations validating the efficacy of the proposed information-gain analysis.

3-3 Model Predictions at Training Stations

It is impractical to assume the availability of a calibrated hydrologic model for all watersheds, and is hence important to analyze the performance of the two networks (Figure 3-3) over the same watershed to infer if they can be used interchangeably. To gauge network performance and validate the findings of information-gain analysis, the performance metrics mentioned in Section 3-2 are employed to ascertain estimation accuracy.

The probabilistic models were trained using data obtained from 2001 – 2004 and tested during the period 2005 – 2006 in order to ascertain its predictive potential. Table 3-4 shows the performance of the network incorporating Q_{net} (N1) and the network incorporating WAM (N2) in terms of the metrics presented in Section 3-2-4. To avoid misjudgments, *NSC*, entropy (*H*), and mean absolute error (*e*) metrics should always be analyzed in tandem while evaluating the performance of a model. For example, a model that consistently over-estimates streamflow could still have a small entropy value due to a peaked uni-modal error distribution. Also, if the error distribution is not uni-modal, analyzing a second-order moment-based metric like *NSC* without considering entropy values can lead to misinterpretation of model behavior.

Taking these into consideration, it can be inferred from Table 3-4 that N1 and N2 perform equally well in the confined regions, achieving high *NSC* and low mean absolute error and entropy values. In general, the performance degrades in the unconfined region with lower *NSC* and larger mean and entropy values with N2 outperforming N1 primarily due to the inclusion of WAM estimates. The estimated and observed streamflow values over the test period for the three stations are shown in Figures 3-4, 3-5, and 3-6 for N1 and N2.

Maximum *a posteriori* probability (MAP) values were used to analyze performance. Since the distributions were non-Gaussian, investigating error as a multiple of standard deviation would not be meaningful. The MAP values obtained over time is very similar for the two networks, although higher values are achieved throughout the estimation in case of N2 (supports the fact that WAM ranked higher than runoff in Section 3-2 in terms of information contribution). The following analyses apply to the performance of both networks. MAP values are moderately high at stations 2321000 and 2321500 during heavy rain/drought events. Since flow in confined regions is predominantly characterized by rainfall and surface runoff, inclusion

of these nodes in the network leads to good performance. However, lower MAP values are achieved during intermittent rain events. This suggests that a component that efficiently captures complex short-term sub-surface behaviors is required for improving the reliability of flow estimates during such periods. In the unconfined region, sub-surface interactions contribute to a persistent baseflow. Since baseflow is neither captured by WAM (due to simplified ground water representation) nor nearest neighbor flow (i.e., flow at 2321500 located in the confined region), MAP values are low during baseflow-dominant periods. Relatively higher MAP values are achieved while predicting flow during rain events when baseflow is augmented by surface runoff.

The performance drop for N2 was also analyzed under reduced percentage of training data (i.e., a percentage of days starting from 2001 instead of all days from 2001-2004) and was found to be largely negligible (Tables 3-5 and 3-6), particularly in the confined region. The reduction in entropy values at station 2321000 is counter-intuitive as this is accompanied by a reduction in *NSC* values and an increase in mean absolute error. The steeper performance drop in the unconfined region is naturally understandable because of the use of fewer pertinent features in the network. This suppor ts the notion that when more relevant features are used in the prediction, a shorter temporal record of data can still give satisfactory results. This study is important since many watersheds have limited historical data, in some cases because *in situ* sensors were deployed only recently and in others because existing sensors were not maintained continuously.

Time series of actual and predicted errors are plotted for all three training stations over the test period in Figures 3-7, 3-8, and 3-9 along with the PDFs of the error time series. It can be

observed that the error predictions are significantly close to the actual errors, particularly at the confined regions, indicating reliability of model performance.

3-4 Model Predictions at Intermediate Stations

One of the primary motivations for developing data fusion estimation methodologies for watersheds is to eventually be able to estimate streamflow, and subsequently transport rates for sediment and pollution, in large river basins that can only be sparsely instrumented with *in situ* sensors. Viewed another way, it is desirable for the estimators to be spatially explicit such that they can be applied to any arbitrary location along the reach of a stream or river. It is also highly desired that, in a sparsely distributed sensor problem, all available information be exploited for achieving best performance. To estimate flow and uncertainty at any intermediate test point along the river, information from all available training stations are fused (Figure 3-10). Similarities in feature space are calculated between training stations and intermediate test points (given WAM_k, TNN_k,GWL_k) which are then used to weigh individual flow values estimated from each training station. The flow and posterior probabilities obtained on October 4th, 2006 at points all along the river is shown in Figure 3-11. Locations where estimates are accompanied by low posterior probabilities (i.e., high uncertainty values) can be considered as future instrumentation sites. The model indirectly refers to heterogeneity in the feature vector (i.e., the feature vector is not effectively captured in the probability space describing the BN) at these locations. The formulae used to estimate expected flow and maximum *a posteriori* probabilities at intermediate points along the river are given in Equation 3-7.

The error analysis, explained in Section 3-2-3 is applied to validate performance of the models at intermediate points. Availability of sparse streamflow measurements imposes considerable constraints in performing such an analysis at every intermediate point along the river. The methodology was hence validated at USGS station 2321975 (located between stations

2321500 and 2322500: Figure 2-1) that had measured streamflow for the test period. The observed and predicted streamflow values at 2321975 are plotted in Figure 3-12. The estimates were predicted with an *NSC* value of -0.29, mean absolute error of 530ft^3/sec, and an entropy value of 2.96nats. The actual and predicted errors along with their PDFs, shown in Figure 3-13, suggest reliable model performance.

$$
\begin{aligned}
p(\mathit{flow}_k \in \omega_j \mid x_k) &= \sum_i p(\omega_j \mid x_k, s_i)\, p(s_i \mid x_k) \\
&= \sum_i \frac{p(x_k \mid s_i)\, p(s_i)}{\sum_l p(x_k \mid s_l)\, p(s_l)} \times \max_j \frac{p(\omega_j \mid s_i)\, p(x_k \mid \omega_j, s_i)}{\sum_j p(\omega_j \mid s_i)\, p(x_k \mid \omega_j, s_i)} \\
E(\mathit{flow}_k \mid x_k) &= \sum_j \omega_j\, p(\mathit{flow}_k \in \omega_j \mid x_k) \\
&= \sum_j \omega_j \sum_i p(\omega_j \mid x_k, s_i)\, p(s_i \mid x_k) \\
&= \sum_i p(s_i \mid x_k) \sum_j \omega_j\, p(\omega_j \mid x_k, s_i) \\
&= \sum_i \frac{p(x_k \mid s_i)\, p(s_i)}{\sum_l p(x_k \mid s_l)\, p(s_l)} \sum_j \omega_j \frac{p(\omega_j \mid s_i)\, p(x_k \mid \omega_j, s_i)}{\sum_j p(\omega_j \mid s_i)\, p(x_k \mid \omega_j, s_i)}
\end{aligned}
\qquad (3\text{-}7)
$$

ω_j = Mean streamflow in class j; $j \in [1,15]$; x_k = feature vector at point k

$s_i, s_l \equiv [2321000, 2321500, 2322500]$; $p(s_{i,l}) = 1/3 \forall i, l$

Table 3-1. Comparison of computational complexity between traditional Bayesian method, RDBE and STBN

Method	Operation	Complexity	Description
Bayes'	PDF estimation	$O(Nn^d)$	$d \rightarrow$ No. of reduced dimensions or features $N \rightarrow$ No. of training samples $n \rightarrow$ No. ofdiscrete levels along a dimension
RDBE	PDF estimation	$O(Nn^t)$	$t \rightarrow$ No. of reduced dimensions or features $N \rightarrow$ No. of training samples $n \rightarrow$ No. of discrete levels along a dimension
	Entropy estimation	$O(mNn^t)$	$t \rightarrow$ No. of dimensions in reduced set $N \rightarrow$ No. of training samples $n \rightarrow$ No. of discrete levels along a dimension $m \rightarrow$ No. of class pairs
BN	PDF estimation	$\text{Max}_i\ O(Nn^i)$	$i \rightarrow$ No. of incoming arcs to a node in the tree $N \rightarrow$ No. of training samples $n \rightarrow$ No. of discrete levels along a dimension

Table 3-2. Feature ranking based on conditional entropy (CH)

Stations	Rank	Rank 1	Rank 2	Rank 3	Rank 4
2321000	First feature	TNN :1.5768	WAM:1.8315	GWL:1.8764	Q_{net}:1.9576
	Second feature	GWL :1.5480	WAM:1.5587	Q_{net} :1.5759	
	Third feature	WAM:1.5397	Q_{net} :1.5477		
2321500	First feature	TNN :1.8980	WAM:1.9033	GWL:1.9243	Q_{net}:2.0952
	Second feature	GWL :1.7872	WAM:1.8443	Q_{net} :1.8881	
	Third feature	WAM:1.7735	Q_{net} :1.7848		
2322500	First feature	WAM:2.0097	GWL :2.0718	TNN :2.1713	Q_{net}:2.4283
	Second feature	GWL :1.9934	TNN :1.9935	Q_{net} :2.0074	
	Third feature	WAM:1.9318	Q_{net} :1.9844		

The set of available features reduces by one with each column, represented by '-' entries

Table 3-3. Performance metrics for cases 1-4 validating information-gain analysis

Station	Case no.	Feature	*NSC*	*E*	*H*
2321000	1	TNN	0.6434	94	2.24
	2	WAM	0.3312	88	1.98
	3	TNN, GWL	0.6434	92	2.29
	4	GWL, Q_{net}	–4.4210	523	2.14
2321500	1	TNN	0.7727	215	3.01
	2	WAM	0.4330	206	2.12
	3	TNN, GWL	0.7717	202	2.41
	4	GWL, Q_{net}	–1.8580	926	2.58
2322500	1	WAM	0.0440	327	2.44
	2	TNN	0.0240	345	2.93
	3	WAM, TNN	0.2581	272	2.47
	4	GWL, Q_{net}	–1.9340	840	3.67

Table 3-4. Performance metrics for networks N1 (Q_{net}) and N2 (WAM)

Metric	Network	2321000	2321500	2322500
NSC	N1	0.6865	0.7664	0.0371
	N2	0.6486	0.7183	0.2586
Mean abs. error (cu.ft/sec)	N1	83.0000	207.0000	317.0000
	N2	64.0000	174.0000	272.0000
Std.dev of error (cu.ft/sec)	N1	103.0000	182.0000	435.0000
	N2	125.0000	248.0000	387.0000
Entropy of error PDF (nats)	N1	2.0400	2.4100	2.7900
	N2	2.2000	2.1500	2.5300

NSC: Nash Sutcliffe coefficient ranges from -∞ to 1 (good)

Table 3-5. Performance metrics for N2 under 80% Training Data

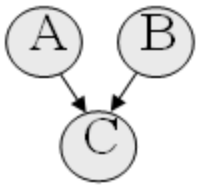
Metric	2321000	2321500	2322500
NSC	0.5704	0.7068	0.1655
Mean abs. error (cu.ft/sec)	68.0000	171.0000	385.0000
Std.dev of error (cu.ft/sec)	139.0000	257.0000	320.0000
Entropy of error PDF (nats)	1.2700	2.3700	2.9200

NSC: Nash Sutcliffe coefficient ranges from -∞ to 1 (good)

Table 3-6. Performance metrics for N2 under 50% Training Data

Metric	2321000	2321500	2322500
NSC	0.5482	0.4939	–1.0850
Mean abs. error (cu.ft/sec)	60.0000	183.0000	676.0000
Std.dev of error (cu.ft/sec)	147.0000	363.0000	414.0000
Entropy of error PDF (nats)	1.0800	2.4800	3.4600

NSC: Nash Sutcliffe coefficient ranges from -∞ to 1 (good)



$$P(A,B,C) = P(A)P(B)P(C|A,B)$$

A

$$P(A,B,C) = P(A)P(B|A)P(C|B)$$

B

Figure 3-1. Basic node structures in BNs. A) Convergent connection. B) Serial connection.

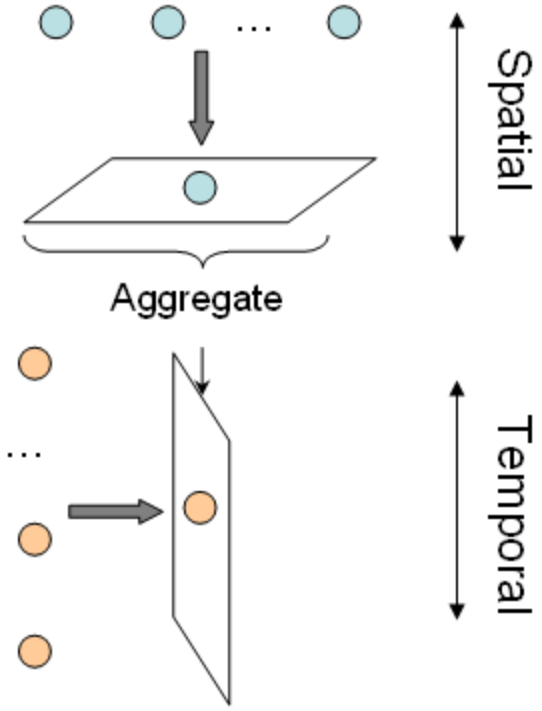


Figure 3-2. Sample network representation of a BN for embedding spatio-temporal data.

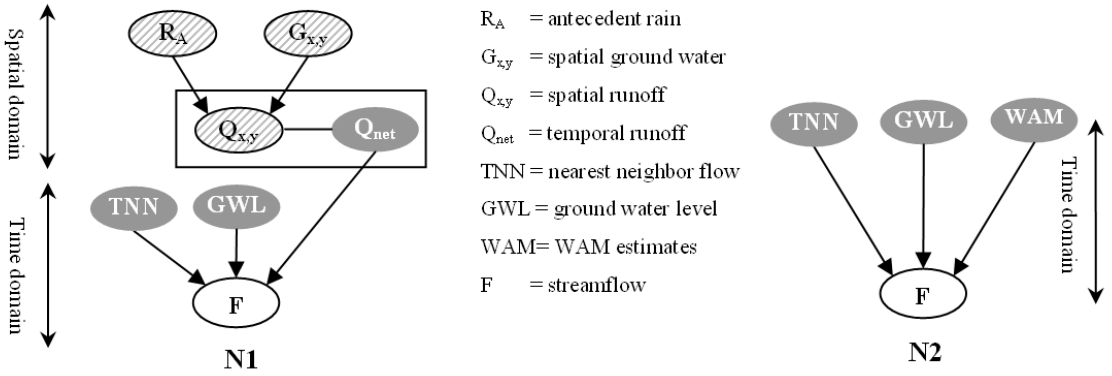
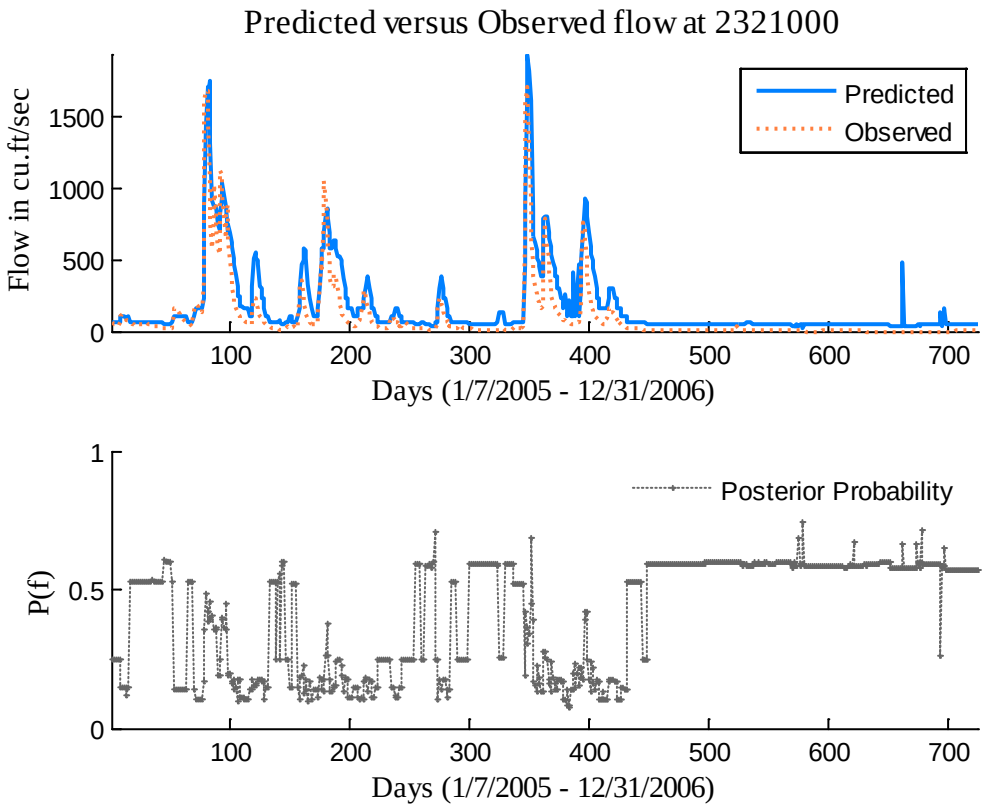
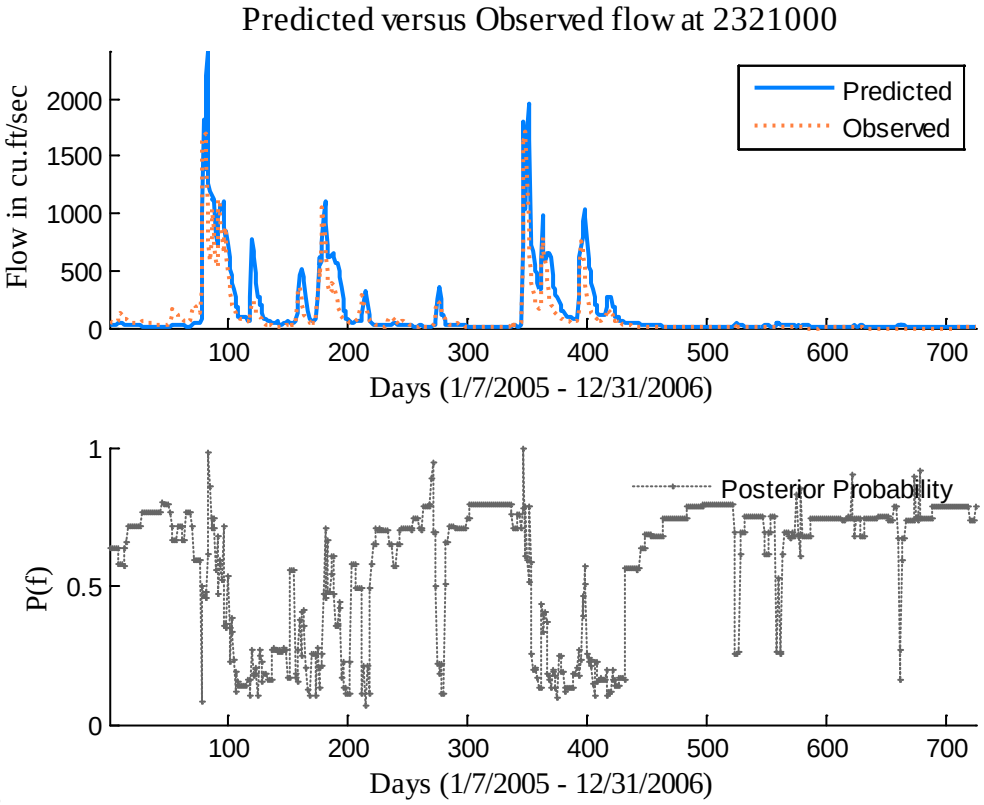


Figure 3-3. Bayesian networks for streamflow estimation given spatio-temporal data sources.

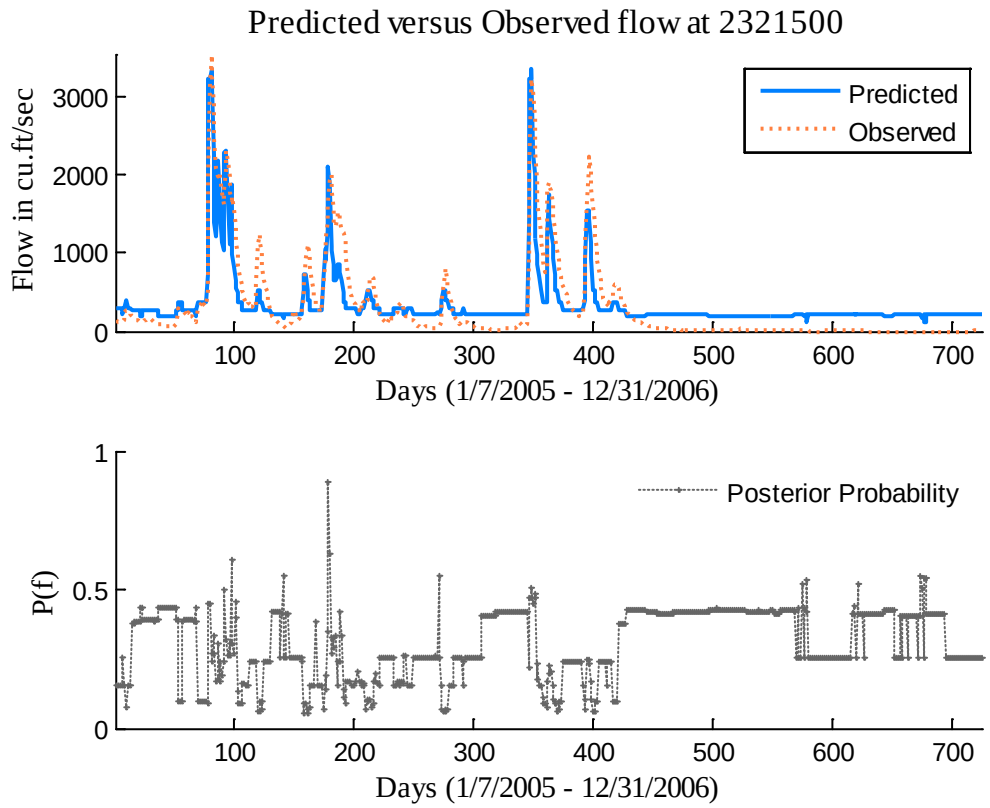


A

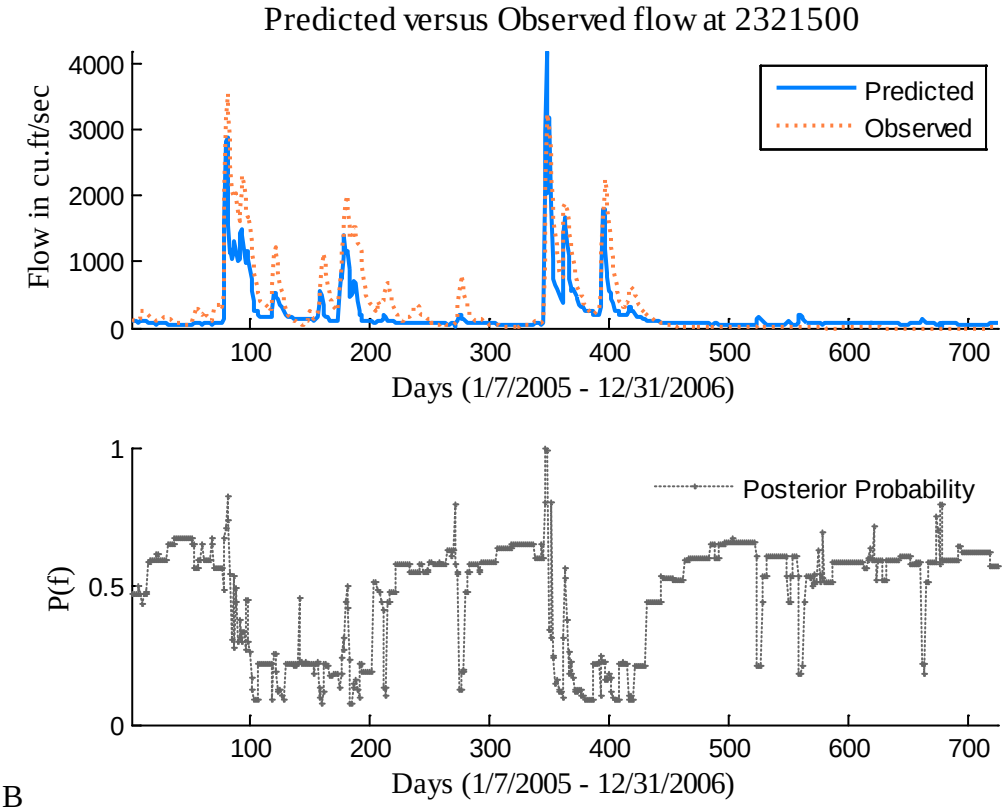


B

Figure 3-4. Streamflow estimates at station 2321000. A) N1 estimates. B) N2 estimates.

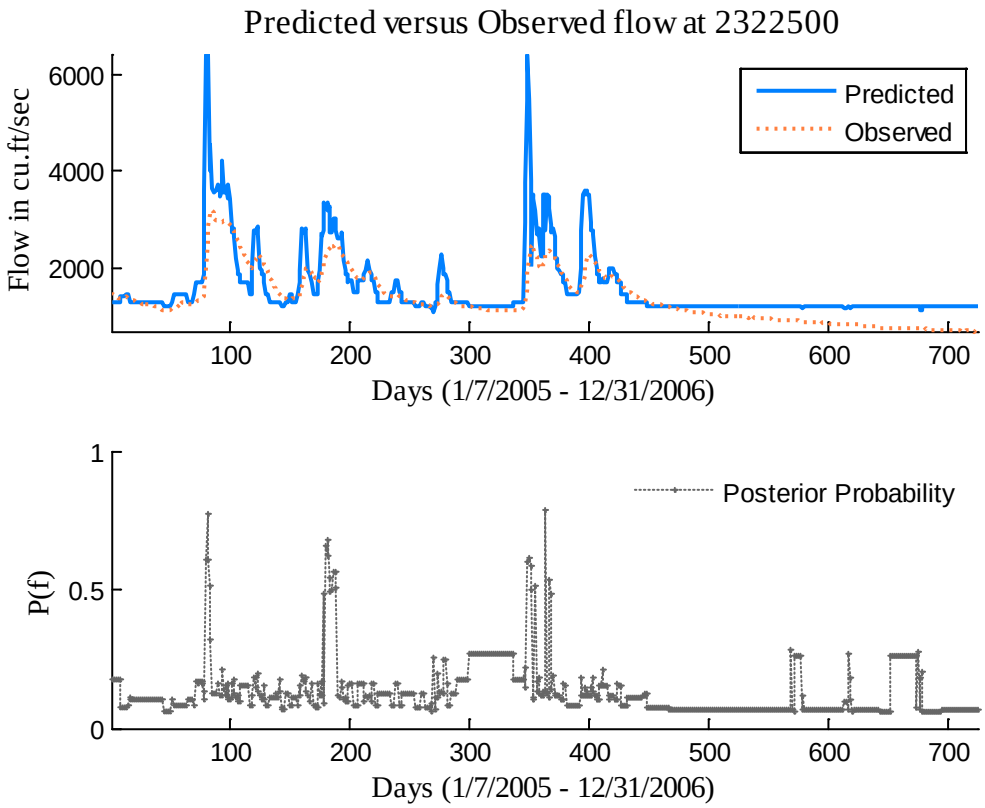


A

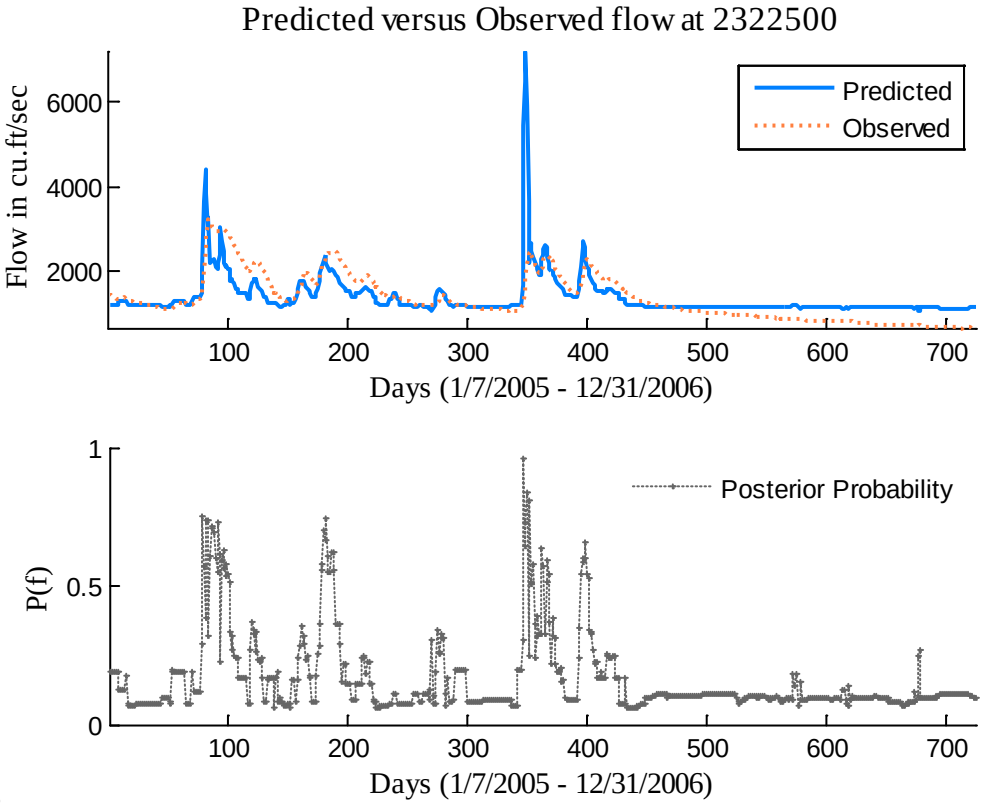


B

Figure 3-5. Streamflow estimates at station 2321500. A) N1 estimates. B) N2 estimates.

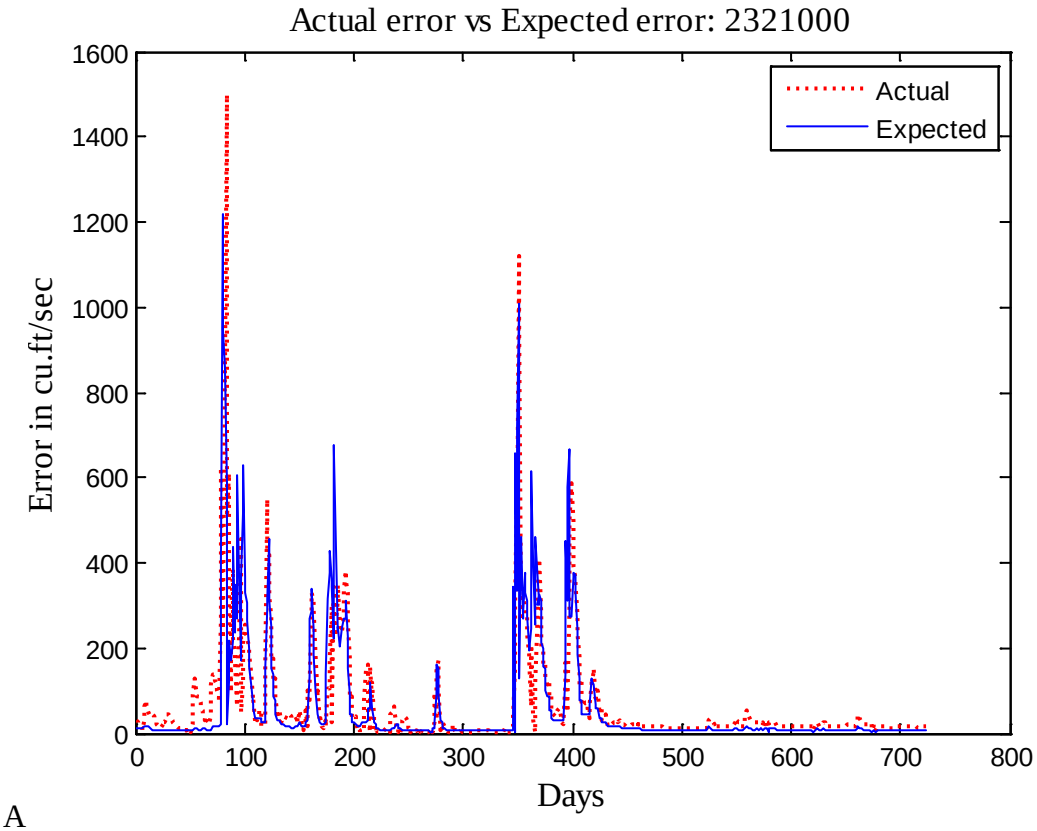


A

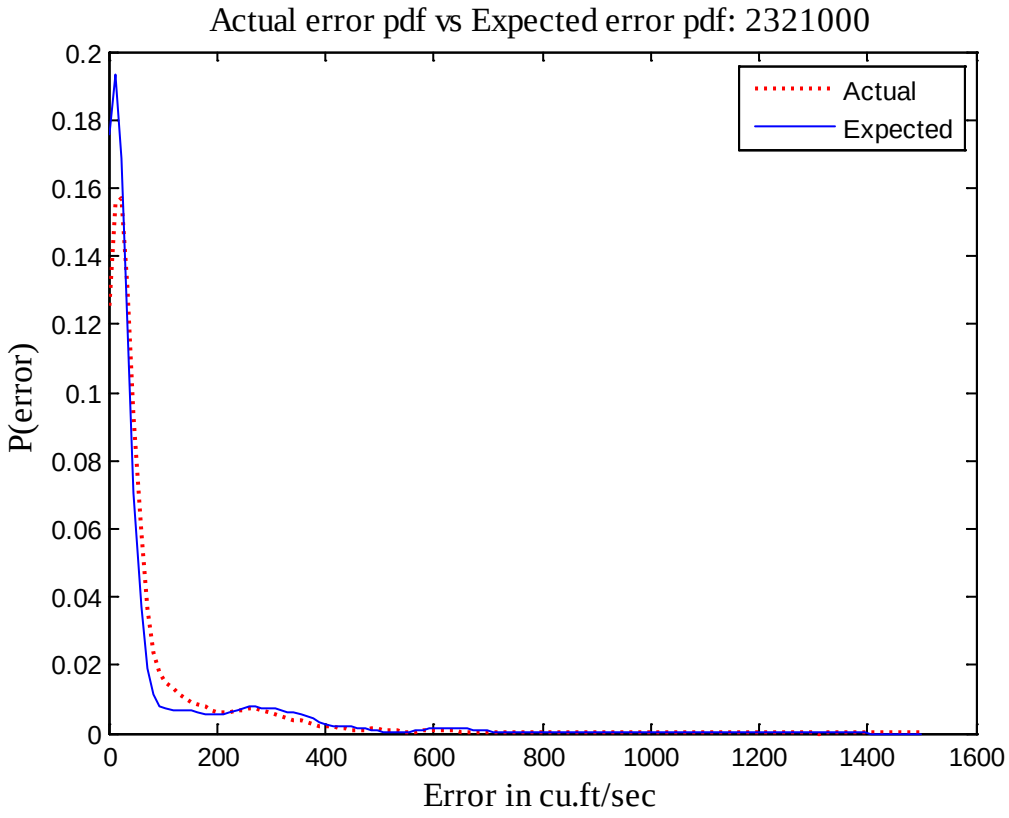


B

Figure 3-6. Streamflow estimates at station 2322500. A) N1 estimates. B) N2 estimates.

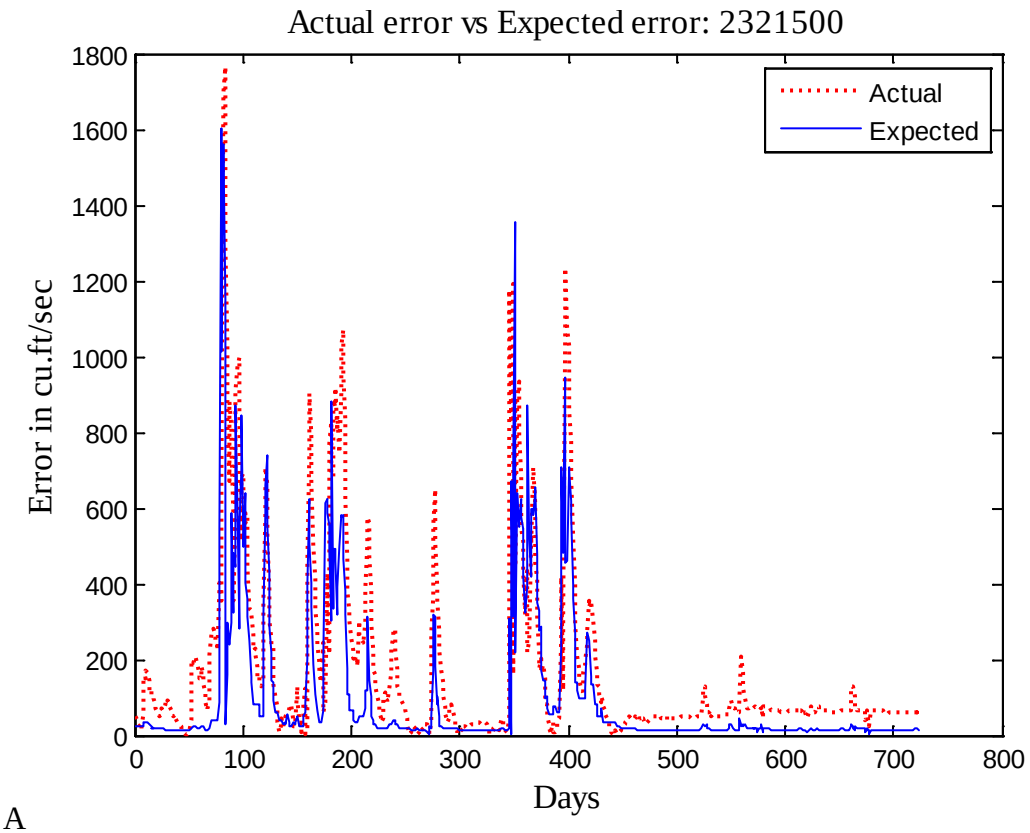


A

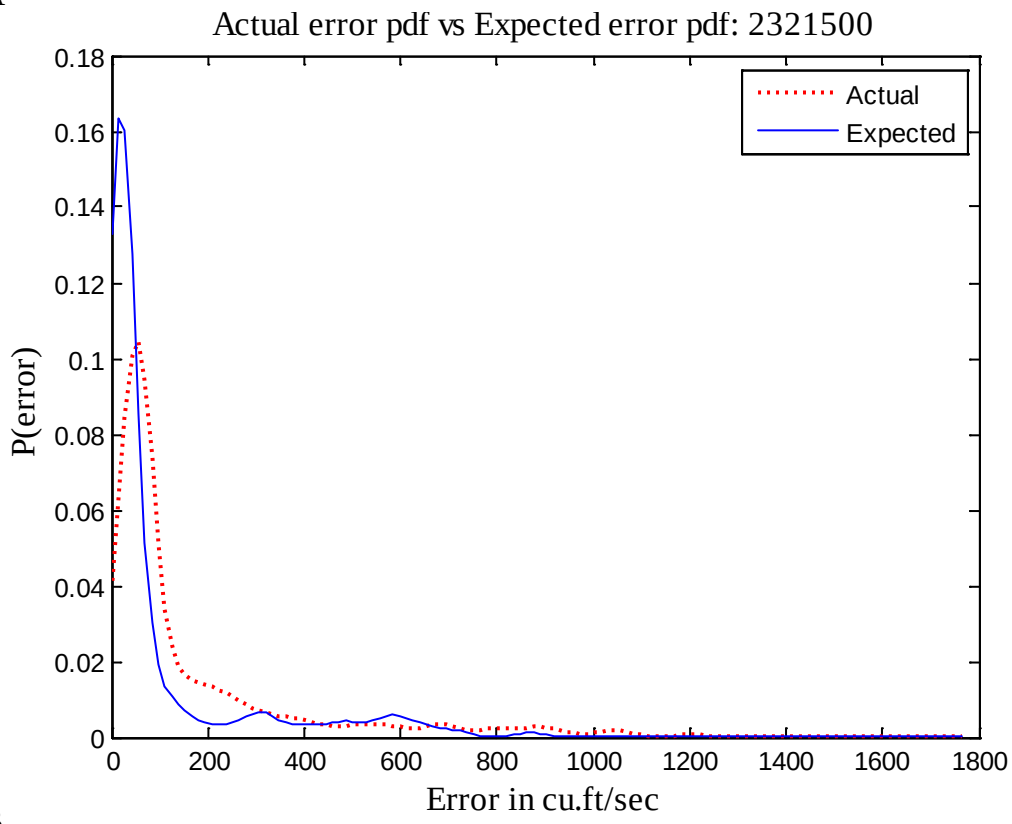


B

Figure 3-7. Model validation at 2321000. A) Estimation errors. B) PDF of errors.

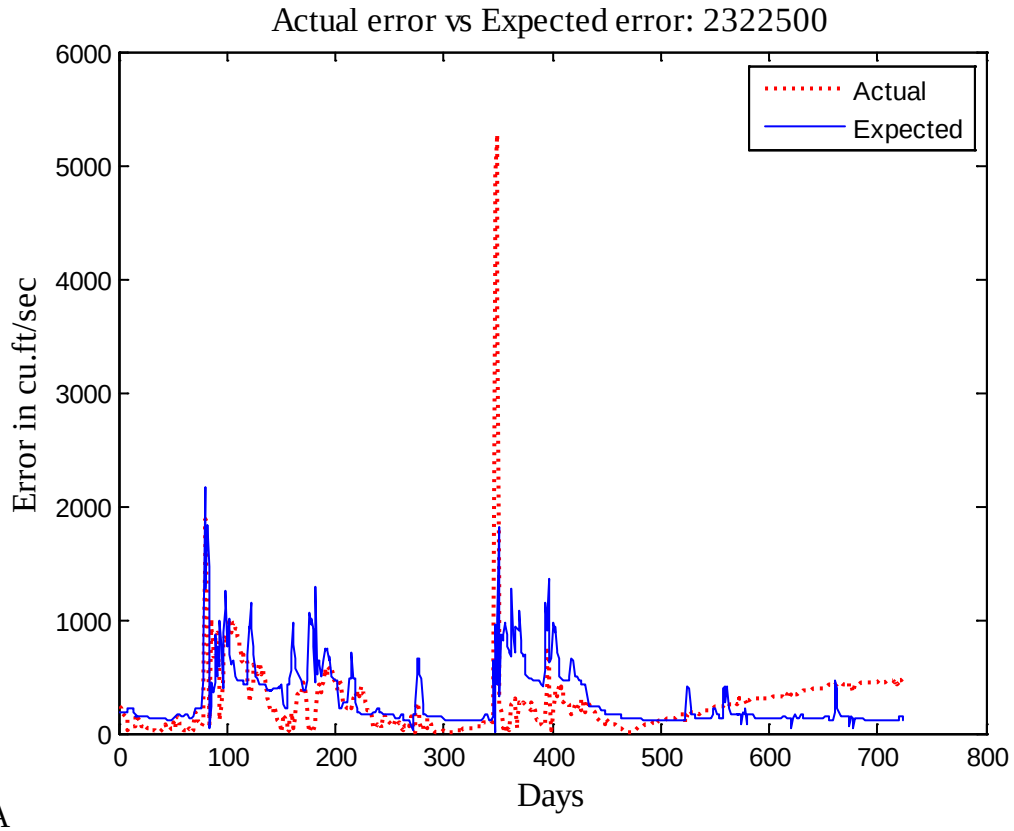


A

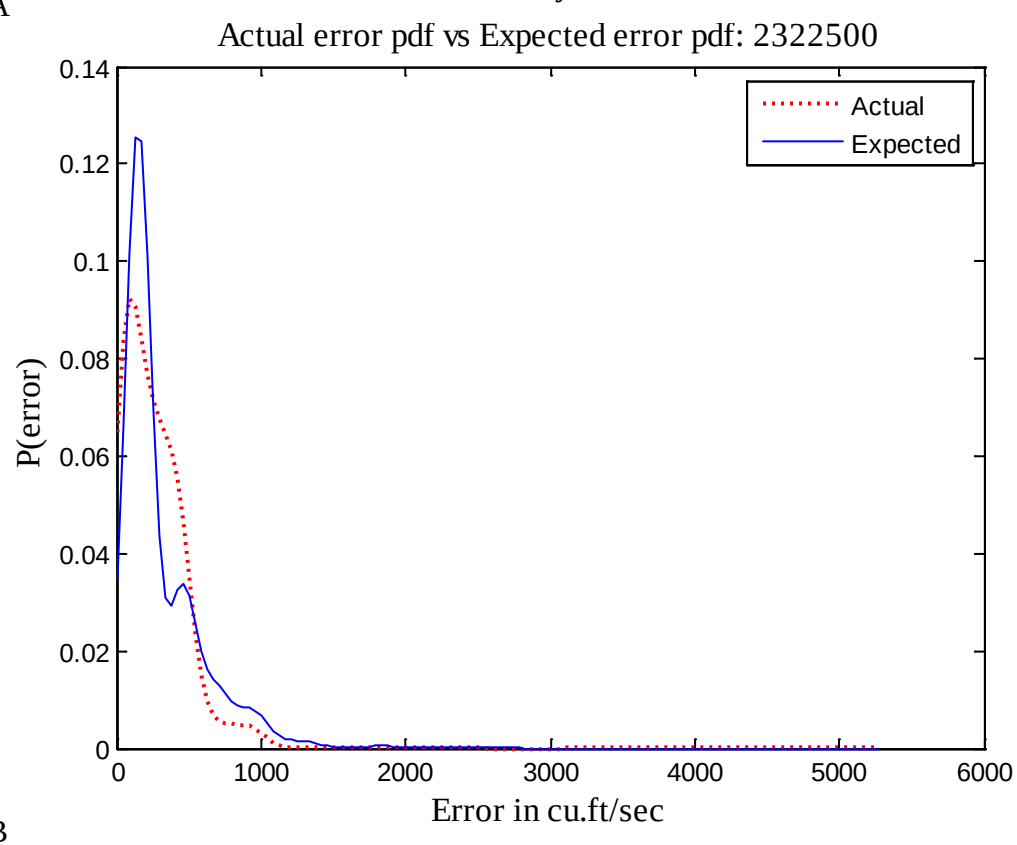


B

Figure 3-8. Model validation at 2321500. A) Estimation errors. B) PDF of errors.



A



B

Figure 3-9. Model validation at 2322500. A) Estimation errors. B) PDF of errors.

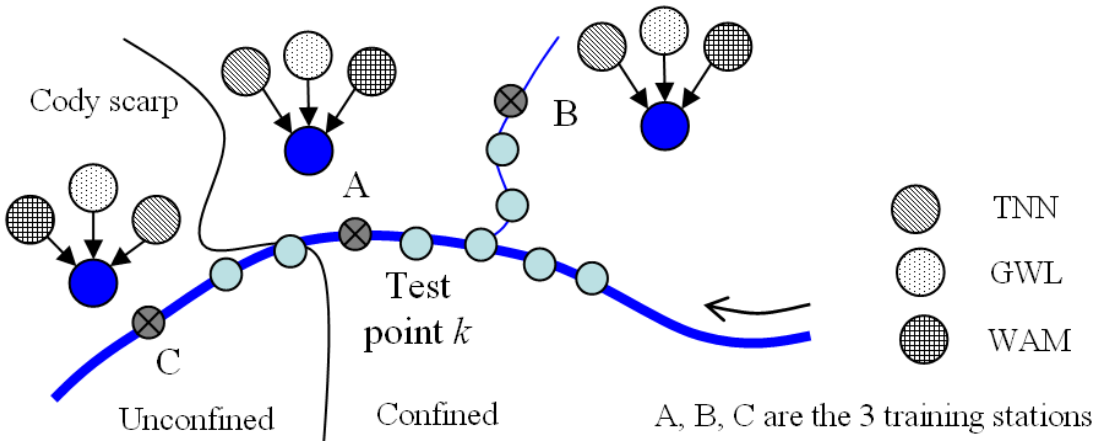


Figure 3-10. Streamflow estimation at intermediate points.

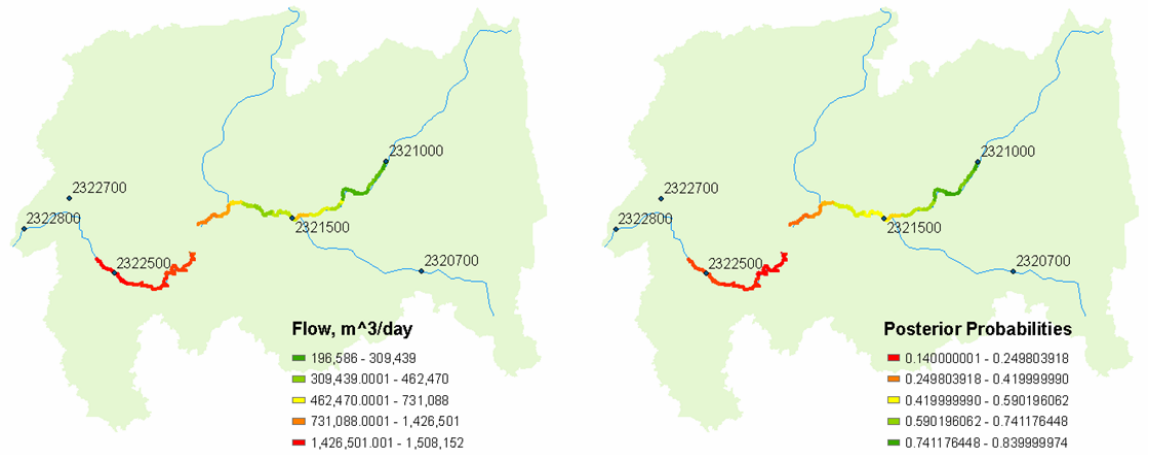


Figure 3-11. Streamflow and MAP estimates along the Santa Fe river on October 4th, 2006.

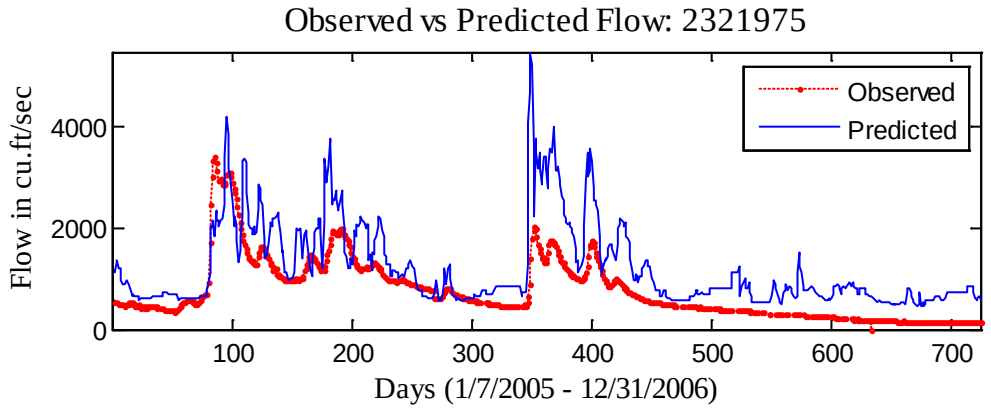
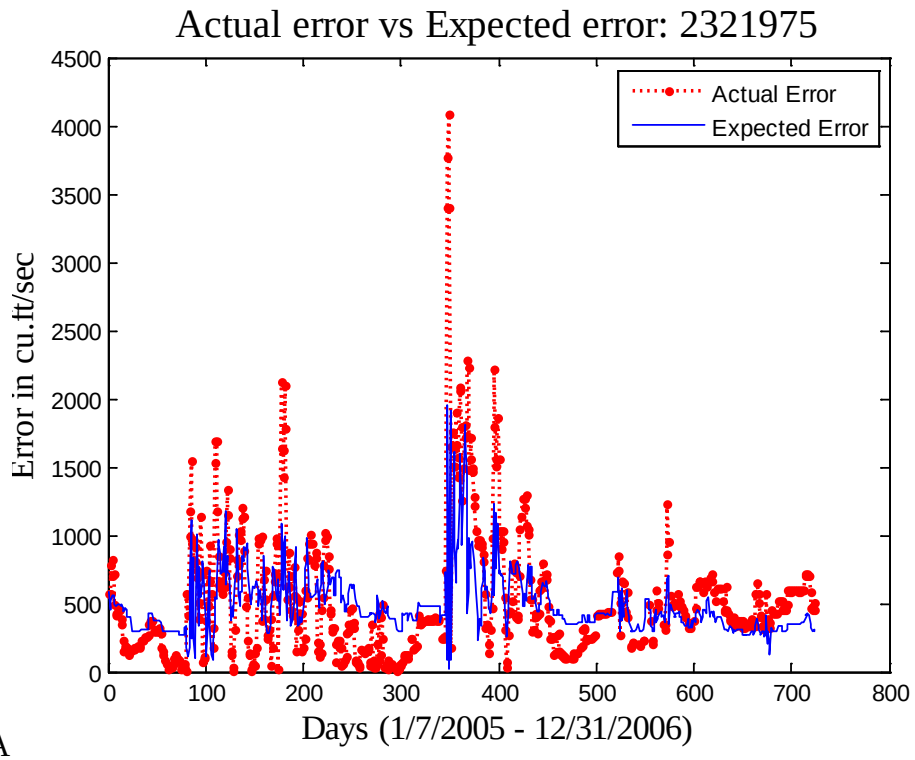
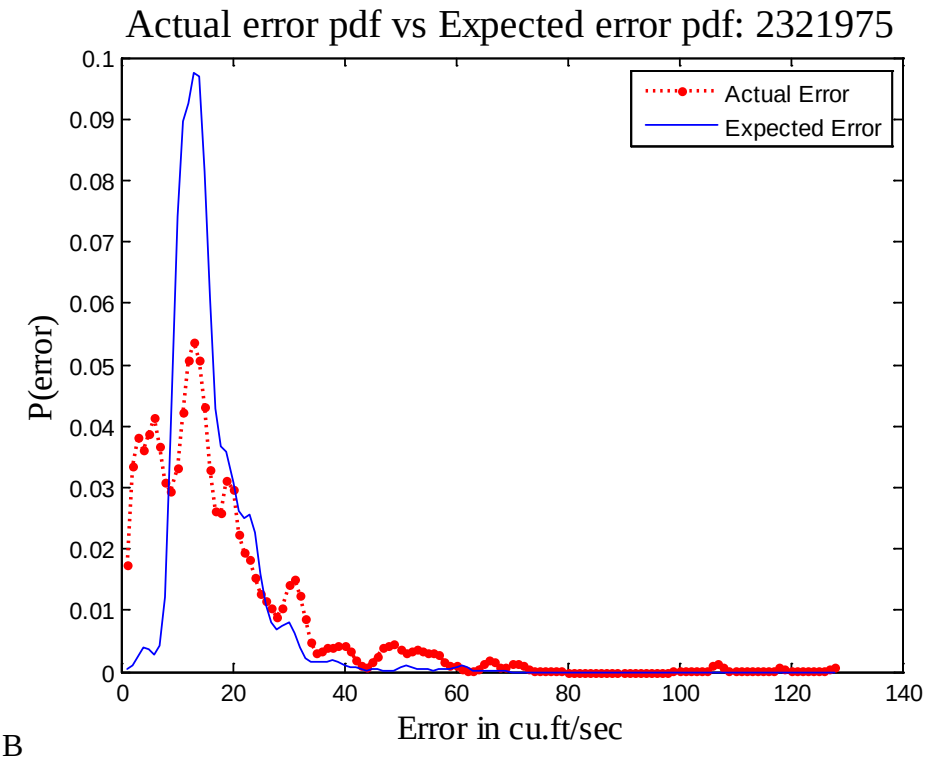


Figure 3-12. Observed versus predicted streamflow at USGS station 2321975.



A



B

Figure 3-13. Model validation at 2321975. A) Estimation errors. B) PDF of errors.

CHAPTER 4
HARDWARE ACCELERATION ON FPGAS

4-1 Pattern-based Algorithm Decomposition

An emerging method for accelerating algorithm execution involves the use of RC based on FPGA technology. Designers who have achieved success in attaining orders of magnitude increase in performance (e.g., speed and power) through hardware acceleration of algorithms are in a select group and are typically highly skilled at exploiting FPGA-based systems and related tools (e.g., hardware description languages (HDLs), logic design, and performance prediction tools). To significantly increase the productivity of FPGA design and development in general, we must improve the productivity of designers who are not FPGA experts. High-level language tools, popularly known as application mappers (e.g., Impulse C, AccelDSP, Carte C, C2H), translate high-level language code to HDL versions, thereby improving productivity. Unfortunately, performance in such scenarios is highly dependent on the mapper's efficiency in extracting parallelism from the algorithm structure. The mapper also dictates algorithm decomposition for a parallel implementation, restricting the designer’s freedom to explore different decomposition strategies.

Another way of improving design productivity is to reuse and leverage existing hardware designs. One of the primary motivations behind this work is to represent FPGA designs at a higher level of abstraction, thus making them more readable and therefore reusable. Design patterns [62] were introduced in the software engineering domain as common solutions to recurring problems and have successfully formed the basis for creating and exploring designs, reusing solutions for similar software applications. More recently, there have been efforts to apply pattern-based design to FPGA-based reconfigurable computing [63, 64]. In [55], the authors identified and cataloged an extensive list of design patterns that can be used to solve a

broad range of problems spanning diverse fields. By decomposing algorithms in terms of simple design patterns, a developer is able to understand the dataflow and structure behind the parallel implementation without delving into the details of a hardware circuit. In this work, a list of primitive design patterns have been identified and categorized in such a manner that is most appropriate for the design of machine-learning algorithms used in many estimation problems and for efficiently exploring the designs using performance prediction tools like RAT [53]. Two of the most important categorizations of patterns and their examples are shown in Table 4-1: computation patterns and communication patterns. While computation patterns deal with extracting and implementing parallelism in the algorithm, communication patterns provide different means of regulating data flow to keep the computation patterns busy. Note that the patterns listed in Table 4-1 are primitive patterns. A variety of composite patterns can then be constructed from these primitive patterns to solve potentially any complex problem. For example, a wavefront pattern [55] can be constructed using a datapath replication pattern with each parallel datapath comprising of pipeline patterns of different lengths.

4-1-1 Pattern Description

The pipeline and datapath replication patterns are described in this section using a standard format suggested in [55]. These two primitive patterns are highly important in that any composite pattern for the purpose of expressing and implementing parallelism can be constructed from a combination of pipeline and datapath replication patterns.

4-1-1-1 Pipeline pattern

- INTENT. Used for implementing programming structures wherein successive/intermediate instructions overlap in execution. The instruction throughput is increased by executing a number of instructions per unit of time resulting in a lesser number of net clock cycles spent during processing.
- MOTIVATION. The primary motivation is to achieve faster processing speeds. When a program runs on a processor, the basic assumption is that each instruction in the program is

executed before the execution of a subsequent instruction is begun. However, in many cases the instruction throughput could be increased if the program allows the possibility to execute more instructions per unit of time. Each instruction is computed and linked into a 'chain' so each instruction's output is an input to a subsequent instruction.

- APPLICABILITY. This pattern is intended for program structures where different instructions in the program execution could operate over different data concurrently.

- PARTICIPANTS. A communication pattern regulates dataflow; partial/intermediate results may need to be accumulated or buffered for subsequent usage.

- COLLABORATIONS. A communication pattern regulates dataflow and keeps data fed to the pipeline.

- CONSEQUENCES.

    - When all instructions in the program execution are not independent, the pipeline control logic (i.e., the controller) must insert a stall or wasted clock cycle into the pipeline until the dependency is resolved.

    - While pipelining can theoretically increase performance over a non-pipelined core by a factor of the number of stages (assuming the clock frequency also scales with the number of stages), in reality, design limitation will not always lead to ideal execution.

    - The instruction latency in a non-pipelined design is slightly lower than in a pipelined equivalent. This is due to the fact that extra flip flops must be added to the data path of a pipelined design.

- KNOWN USES. Any algorithm/program that has independent instructions in its structure (e.g., PDF estimation, Correlation).

4-1-1-2 Datapath replication pattern

- INTENT. Used for exploiting computational parallelism observed in sequential programming structures such as computational loops.

- MOTIVATION. The primary motivation is to achieve faster processing speeds. Parallelism observed in computations can be efficiently exploited using custom logic. Computational structures such as loops iterate over the same set of processing instructions multiple times. In case of zero data dependencies between iterations (or limited data dependency), the processing instructions can be replicated in hardware to perform multiple iterations of the loop in parallel.

- APPLICABILITY. This pattern is intended for exploiting parallelism in computational structures such as loops. An important consideration for implementing the pattern is data dependencies between loop iterations. Depending on the individual case, the

implementation may need intermediate buffers or communication links between parallel implementations of computational kernels.

- PARTICIPANTS. Apart from the replicated kernels, a communication pattern regulates dataflow with possible requirements of additional buffers.

- COLLABORATIONS. Communication patterns regulating dataflow and the total number of iterations required on the kernel instances implemented in hardware.

- CONSEQUENCES.

    - Since the pattern implements an area-time tradeoff, higher processing speeds achieved via parallel instances of the kernel come at the cost of increased implementation footprint in hardware.

    - Additional overhead in terms of parallel data communication and control logic are present. Depending on the FPGA platform and implementation, the problem may be communication-bound, limiting the number of parallel kernels that can be fed with data in parallel.

- KNOWN USES. PDF estimation, Molecular dynamics, K-Means clustering, Sorting algorithms.

4-1-1-3 Loop fusion pattern

- INTENT. Used to alleviate potential pipeline stalls for nested loops in hardware designs particularly for automated pipelining HLL compilers. It also encompasses the combining of separate loops into a single loop for potential optimization.

- MOTIVATION. The primary motivation is to remove pipeline stalls from computational blocks. Nested loops are a common feature of computations but can create pipeline stalls in between iterations of the inner-most loop. Automated compilers can quickly create optimized designs for inner-most loop computations. Consequently, fusing loops will allow the automated compiler to pipeline the entire computational block.

- APPLICABILITY. This pattern is primarily intended for nested loops. It also applies to combining separate loops to reduce the overall number of interactions, potentially at the expense of a larger memory footprint. Reducing overall cycle count (through alleviating pipeline stalls) is the goal of loop fusion.

- PARTICIPANTS. A nested loop is fused into a single loop. Alternatively, multiple independent loops are combined into a single loop. Additional logic is potentially added to the loop to retain n-dimensional indexing in a flat, single-dimensional loop.

- COLLABORATIONS. A controller manages the dataflow to keep feeding data in parallel to the implemented kernels. It also manages the total number of iterations required on the number of kernel instances implemented in hardware.

- CONSEQUENCES.
    - Additional logic for indexing is required to support nested loop fusion. Naïve implementations of loop fusion can require computationally expensive operations such as division and modulo arithmetic. Other implementations require counters whose values persist between loops. Without extra compiler support, such implementations may not be optimal.
    - Independent loop fusion may not have substantial benefit if loop overheads can be overlapped with computation.
- KNOWN USES. Molecular Dynamics, N-body problem.

4-1-2 Quantifying Design Patterns

Since design patterns provide a formalized way to represent algorithm decomposition and the associated communication for parallel implementation, they can be quantified and used in analytical models like RAT to predict the algorithm's amenability to hardware acceleration (to be described in Section 4-2-1), before undertaking a lengthy (and possibly fruitless) development process. The impact of each design pattern can be analyzed in terms of throughput and/or overhead contribution to design performance. In this work, *throughput* of a design pattern is defined as the maximum number of operations that it can perform per clock cycle. *Latency* is defined as the number of cycles spent in data transfers before any useful computation is performed. For example, a pipeline pattern has a *throughput* equal to the number of pipeline stages and contributes an equal cycle count on *latency* (i.e., clock cycles spent in filling up the pipeline stages). The *throughput*, *latency*, and graphical notation of patterns relevant to this work are given in Table 4-2. Having decomposed the algorithm as a composite of primitive patterns, the net *throughput* and *latency* can be inferred from the structure and behavior of the constructed composite pattern.

4-2 Scalable and Portable Architecture for PDF Estimation

Having investigated and analyzed the commonalities shared by the three case-study algorithms in this work (Section 2-4-4) we suggest that an efficient FPGA design developed for one of the algorithms, say PDF estimation, can be effectively reused for computing the others. For this reason, an in-depth explanation of the design and performance evaluation of a suitable architecture for only the PDF estimation algorithm will be given in the following sub-sections. Then, in Section 4-3, we will discuss how the architecture developed for PDF estimation can be effectively adapted by reusing most of its design for developing other similar algorithms.

The general architecture of the 1-D PDF estimation algorithm based on pattern-based decomposition is shown in Figure 4-1. A computational kernel updates the PDF value at a particular point x (i.e., a bin) based on the value of a particular data sample x_i (where $x_i \epsilon I_1$ and $x \epsilon I_2$). The Parzen window technique is an embarrassingly parallel algorithm where the PDF values at multiple points can be evaluated at the same time by replicating the computational kernel (datapath replication pattern). The kernels in the parallel datapath are *seeded* with different values in I_2 via a scatter communication pattern. Thus, data samples can be broadcasted across kernels that can then process data independently in parallel. Each kernel can also benefit from a pipelined implementation while performing all the operations (Equation 2-6) required on every data sample (pipeline pattern). The PDF values are computed in $\varphi(I_1,I_2)$, accumulated in $p(I_2)$ and stored in output memory O. Load balancing and reduction of communication and synchronization requirements are necessary to ensure that hardware outperforms its sequential software counterpart. In estimating multi-dimensional PDFs, computation of the kernel functions φ in each dimension are independent of the others and hence performed in a parallel fashion within each pipeline. Internal registering for each bin keeps a running total of all processed data; these cumulative totals comprise the final estimation of the PDF function.

The development stages in the system design are highlighted in Figure 4-2 and explained briefly in the following bullets for a 1-D PDF design. The same can be extended for higher-dimensional PDF estimation with suitable modifications made to the computational kernel.

- PERFORMANCE PREDICTION. Based on a preliminary design of the algorithm (in the form of design patterns shown in Figure 4-1A) and the basic resources available for a selected FPGA-based computing platform (in the form of parameter values to quantify the design patterns), the attainable speedups are predicted using RAT. Numerical analysis is conducted and a suitable fixed-point implementation is chosen because probability values lie between 0 and 1, negating the need for the dynamic range features of a floating-point format. Details of RAT's performance prediction are provided in Section 4-2-1.

- KERNEL AND CORE DESIGN. The basic task of a kernel in the 1-D case is the computation of the kernel function $\varphi = 1-(x_i-x)^2$. A core contains a number of kernels (k) with each kernel in the design performing the aforementioned computation for a different x (i.e., different *seed*). The computation increases in complexity while estimating higher-dimensional PDFs (Section 4-2-3). The parameter k is chosen based upon the preliminary resource analyses performed earlier. Details concerning the kernel and core designs are given in Section 4-2-2.

- TEST BENCH AND SIMULATION. Memory instantiations for x and x_i are made, test bench files are generated, and functional simulations are performed to validate the design.

- OVERALL SYSTEM DESIGN, VERIFICATION, AND VISUALIZATION. Integration of the core with the host processor (middleware design) is developed followed by verification of the computed PDF.

4-2-1 Performance Prediction

Although FPGAs have much to offer in terms of flexibility and performance, they are not amenable for all algorithms. RAT [53] is a simple methodology developed for predicting the performance of a specific algorithm design on a specific platform. By parameterizing a particular design strategy and a platform selection into RAT, the developer can analyze and predict likely speedups attainable. For RAT, *speedup* is defined as the ratio of execution time on a relevant general-purpose processor (t_{GPP}) to the execution time on an FPGA (t_{RC}).

$$speedup = \frac{t_{GPP}}{t_{RC}} \tag{4-1}$$

For ease of predicting speedup, the RAT analytic models take the form of a worksheet, reproduced in Table 4-3, and feature two important steps. The first step deals with estimating the communication burden (t_{comm}) involved in transferring data in and out of the FPGA. The entries in the worksheet related to this step are the communication parameters ($throughput_{ideal}$, α_{write}, α_{read}) and the dataset parameters. The second step involves the estimation of time spent in performing computation (t_{comp}) over the data transferred ($N_{elements}$) on the FPGA. In the original formulation of RAT, parameters $N_{ops/element}$, f_{clock}, $throughput_{proc}$ and $N_{elements}$ determine computation time. The total time spent on the FPGA (t_{RC}) to execute the entire algorithm is calculated (see [23] for details) as given in Equation 4-2.

$$
\begin{aligned}
t_{RC} &= N_{iter}(t_{comm} + t_{comp}) \\
t_{comm} &= t_{read} + t_{write} \\
t_{read/write} &= \frac{N_{elements} \times N_{bytes/element}}{\alpha_{read/write} \times throughput_{ideal}} \\
t_{comp} &= \frac{N_{elements} \times N_{ops/element}}{f_{clock} \times throughput_{proc}}
\end{aligned}
\tag{4-2}
$$

The memory available on the FPGA or the board hosting the FPGA is often much smaller than what is required for storing all of the application data. N_{iter} denotes the number of iterations of communication and computation required to process all available data. The authors in [53] suggest that the parameter $throughput_{proc}$ be approximately chosen based upon the number of data samples that can be processed in parallel. This is not trivial to estimate in many cases and might lead to suboptimal predictions if carelessly chosen. A pattern-based design embeds the parallelism (*throughput*) and the accompanying overhead (*latency*) during algorithm decomposition and would help better parameterize RAT while estimating t_{comp}. Latency effects

in pipelines and buffers are easily extracted from the patterns as against manual interpretation. For example, in the PDF design (Figure 4-1), t_{comp} is computed as given in Equation 4-3.

$$
\begin{aligned}
t_{comp} &= \left(Latency_{net} + \frac{N_{elements} \times N_{ops/element}}{throughput} \right) \times \frac{1}{f_{clock}} \\
Latency_{net} &= L_s + L_p + L_g \\
throughput &= T_p \times T_d
\end{aligned}
\tag{4-3}
$$

The worksheet in Table 4-3 shows the input parameters for the RAT analysis for the 1-D and 2-D PDF estimation algorithms. The communication parameters model a Nallatech H101-PCIXM card containing a Xilinx V4LX100 user FPGA connected to a Xeon host processor over a 133MHz PCI-X bus. Although the entire application involves 204,800 data samples, each iteration of the 1-D PDF estimation on the FPGA will involve only a portion of that data (512 data samples, or 1/400 of the total set) because of memory constraints on the FPGA. A corresponding input buffer size of 512 is chosen for entry in the worksheet. Since the computation is performed in a two-dimensional space for a 2-D PDF, twice the number of data samples (in blocks of 512 words for each dimension) is sent to the FPGA. The number of output elements in the RAT table corresponds to the number of outputs for every call to the FPGA. In the 1-D case, the estimated PDF values at 256 points are returned from the FPGA after all calls to the FPGA are complete (i.e., after N_{iter} calls). There is sufficient block RAM on the FPGA to hold the results of the 1-D PDF algorithm. Since the PDF output elements need not be sent for each call, the communication time is accounted for by distributing the total number of output elements over N_{iter} calls. From Table 4-3, it is understood that the FPGA is called 400 times resulting in an average value of <1 output element per call. This is rounded to the next largest integer (1 in this case). In the 2-D case, the PDF values are computed over 256 × 256 (i.e.,

65536 in the worksheet) points and are sent back to the host for every call made to the FPGA due to insufficient block RAM to store results between calls.

The computational density of the algorithm is defined in terms of $N_{ops/element}$. Each data sample in the 1-D PDF case requires 3 operations at each of the 256 points at which the PDF is estimated, resulting in 768 operations. In the 2-D PDF algorithm, each data sample requires 6 operations at each of the 256 × 256 points leading to 393216 operations. The computational throughputs for the 1-D and 2-D designs are estimated using FPGA clock frequencies of 150MHz and 100MHz, respectively. The *throughput* of the design is determined based on the constructed composite pattern (Figure 4-1A) – 8 parallel datapaths, each comprising of a 3-stage pipeline, leads to a net *throughput* of 24 ops/cycle. L_s and L_g equal 256 (i.e., size of I_2 used to seed the parallel datapaths) and L_p equals 3 leading to a net *latency* of 515 cycles. For the 2-D PDF design, the latency L_s and L_g increase to 65536 (256 × 256 seeds) while the number of replicated datapaths equals 16 (i.e., 8 for each dimension). The software baseline (t_{GPP}) for computing speedup values was measured from an optimized C program (optimization flag was set to O3) executed on a 3.2GHz single-core Xeon processor using single-precision floating point. The truncated Taylor series expansion was used in place of the exponential in the C program as well. RAT predictions are then estimated to determine an attainable execution time and later compared against experimentally measured software results to compute speedup. The predicted speedup for the 1-D and 2-D PDF algorithms was 11.5 and 9.0, respectively.

4-2-2 1-D PDF Design

A design of a 1-D PDF relies heavily on the availability of dedicated arithmetic units and memory blocks. Not only should the available resources be efficiently used but also the design should scale well with application complexity. Taking these points into account, a multi-core design with a key design parameter *k* (the number of kernels or parallel datapaths in a core) is

proposed. The support size of the PDF is 256 along one dimension (I_2) and the number of data samples processed for every call to the FPGA is 512 (I_1). The multi-core aspect of the design is developed with consideration toward future extensions to multi-FPGA systems. Larger FPGAs would be able to house more kernels and hence process more data in parallel leading to faster computations. In the following sections, single- and dual-core designs are described, along with a discussion of *scalability* with respect to the number of cores.

4-2-2-1 Single-core design

The FPGA receives data (I_1 and I_2) over an interconnect (e.g., PCI-X, PCI-Express, RapidIO) and the core accesses data from the FPGA memory. The basic blocks in the design are pictorially represented in Figure 4-4A and the host-centric execution flow is illustrated in Figure 4-5A. A set of data samples (I_1) is first loaded onto the FPGA that is then signaled to start the computation. As the FPGA processes data, the host processor polls for a "process complete" signal from the FPGA. The FPGA sends the "process complete" signal once the computation is finished and the host sends in the next set of data samples (I_1) until all data are processed. Data and control flows to the parallel kernels housed in the core are regulated by a finite-state machine explained as follows (Figure 4-3). The set of *seeds x* scattered to the *k* parallel kernels in the core are represented by I_2**,** the data samples x_i by I_1, and the PDF values by p in the state machine description.

- IDLE. Remain in idle state until activation signal GO=0; transition to scatter state if GO=1.
- SCATTER. Distribute *k* values of I_2, $p(I_2)$ to the parallel kernels; transition to compute state.
- COMPUTE. Broadcast data samples I_1 sequentially, compute $\varphi(I_1,I_2)$, and update $p(I_2)$; transition to scatter state to load subsequent *k* values of I_2, else move to gather state.
- GATHER. Update p in FPGA memory; transition to done state.
- DONE. Send "process complete" signal done=1 to host; transition to idle state when GO=0.

4-2-2-2 Dual-core design

The architectural details and the execution flow of the dual-core design are illustrated in Figures 4-4B and 4-5B, respectively. The state machine for a single-core design and the corresponding core are replicated with changes made to the data flow in the software control program at the host end. Since multiple cores share the same interconnect, arbitration for bus access is performed while loading data onto the on-chip memory of the FPGA and the subsequent polling for the "process complete" signal. The FPGA systems used in this work have a host processor for explicitly managing data movement in and out of the FPGA. Data is written to the first core and, while the first core processes the data, the next set of data samples is loaded to the second core to process. The host then polls the first core for the "process complete" signal as the second core processes its data. There could be a situation where the host cannot poll one of the cores while it is loading data to the other core. The core that is not being polled would sit idle during this contention period.

4-2-3 2-D PDF Design

In the 2-D case, the PDF is evaluated at a matrix of points or bins (n_1 rows of x values × n_2 columns of y values). The number of computations grows from $(N - n)^2 + c$ to $(N - n_1)^2 + (N - n_2)^2 + c$ and the basic kernel computation expands to $1-((x_i-x)^2 + (y_i-y)^2)$ where x_i, y_i represent one data sample. Despite the added complexity of the 2-D PDF algorithm, the increased quantity of parallelizable operations makes this algorithm still amenable to the RC paradigm, assuming sufficient quantities of hardware resources are available. To perform the 2-D PDF estimation, the basic architecture designed for the 1-D case is extended with modifications made primarily to the kernel design (Figure 4-6) and the control flow. Due to the limited amount of FPGA memory, the PDF matrix is computed one column at a time (i.e., for all x and one y, represented

by $p(I_{2x},y)$). This partial PDF matrix is returned to the host before a subsequent call to the FPGA is made. The kernels in the 2-D PDF design are initialized with two sets of seeds (x and y).

The finite-state machine regulating the data and control flow is illustrated in Figure 4-7 and explained as follows. The set of seed values x and y scattered to the parallel kernels (k in number) in the core are represented by I_{2x} and I_{2y}, the data samples x_i and y_i by I_{1x} and I_{1y}, and the PDF column values by $p(I_{2x},y)$ in the state machine description.

- IDLE. Remain in idle state until activation signal GO=0; transition to scatter state if GO=1.
- SCATTER. Distribute y, k values of I_{2x}, and $p(I_{2x},y)$ to the parallel kernels; transition to compute state.
- COMPUTE. Broadcast data samples I_{1x} and I_{1y} concurrently and update $p(I_{2x},y)$; transition to scatter state to load subsequent k values of I_{2x}, else move to gather state.
- GATHER. Update $p(I_{2x},y)$ (i.e., column of PDF matrix) in FPGA memory; transition to done state.
- DONE. Set the "process complete" signal done=1; when GO=0 transition to scatter state if CNT(I_{2y})≤256 (point to next element in y), else move to idle state.

4-2-4 Experimental Results

In this section, experimental results with the PDF FPGA designs are presented and analyzed. The experimental platform used throughout these experiments contains the Nallatech H101 board with V4LX100 FPGA. This Xilinx chip has dedicated DSP48 arithmetic blocks capable of performing rapid 18-bit multiplies and multiply-accumulates compared to logic-based MAC cores. The board communicates with the host processor over a PCI-X interconnect. A 32-bit wide communication channel is used of which 9 bits were allocated for the fixed-point fractional segment. Function-level simulation was performed in ActiveHDL from Aldec, and H101 implementation was rendered using DIMEtalk from Nallatech.

4-2-4-1 Speedup, power, and resource utilization

One of the primary objectives in this work is to obtain speedups in execution by exploiting parallelism at the hardware level when compared to the sequential version execution on a high-end CPU. The single-core 1-D and 2-D PDF designs operated at FPGA clock speeds of 150MHz and 100MHz, respectively. The number of kernels, k, in the core was set to 8. The basic criterion was to develop an efficient design where the resources (e.g., DSP48s, block RAMs, slices) are uniformly consumed. It should be noted that, since the algorithm is embarrassingly parallel, improved performance can be achieved by scaling the design, that is processing more data in parallel by increasing k until on-chip resources are exhausted.

Actual speedup, power and resource utilization factors for the 1-D and 2-D PDF algorithms are presented in Table 4-4. Power was computed using the XPower analysis tool provided by Xilinx under a 12% activity rate. It can be seen from the comparisons made in Table 4-5 that the actual speedups obtained are reasonably close to the predicted speedups using RAT (originally shown in Table 4-3). The deviation between the values is primarily due to inaccuracies in the estimate of t_{comm} values. Although the channel efficiency values (α_{write}, α_{read}) chosen for the RAT analysis are a reasonable assumption for large data transfers, they tend to be lower for smaller data transfers as in this scenario. The t_{comp} predictions are relatively close to the experimental values validating the advantage of employing design patterns for algorithm decomposition and performance prediction. The number of DSP48s scales linearly with the number of kernels k in the core. In the 2-D PDF scenario, each kernel uses twice the number of DSP48s as was consumed in the 1-D case because the computation is conducted along two dimensions. Although the total amount of FPGA memory consumed by the PDF core is moderate, a significant percentage of it is consumed by DIMEtalk's interface to buffer and transfer data to and from the host and FPGA. In addition to low net power consumption, two-

thirds of the total consumption was affiliated to static power (~1020mW) in both the 1-D and 2-D PDF designs. This suggests that scaling the design would potentially result in small power increments while still maintaining low total power.

4-2-4-2 Data verification

Even though significant speedup was shown in the previous section, speedup is meaningless if the results are not accurate. In this section, we verify the accuracy of the data provided by the FPGA designs. Data samples from a bi-modal mixture of Gaussian distribution and from a two-dimensional uni-modal Gaussian distribution were generated in MATLAB to verify the functionality of the 1-D and 2-D PDF designs. PDF values were computed by generating 800KB (N=204,800) of data samples (n=256 for 1-D PDF; n_1=256 and n_2=256 for 2-D PDF). The computed PDF values were read and error in the solution was computed as the difference in FPGA and GPP estimates. Maximum error E of 3.8% and 0.57% were obtained for the 1-D and 2-D algorithms respectively (Equation 4-4), which are reasonable for most real-time applications that involve decision-making based on probabilistic reasoning. The errors in the estimates are due to the Taylor series truncation of the exponential function rather than the fixed-point effects (the exponential function was not truncated in the GPP estimates for this study). The resulting PDFs shown in Figures 4-8 and 4-9 were plotted in MATLAB for verification.

$$E = \frac{\max\left(\left| p(x)_{RC} - p(x)_{GPP} \right|\right)}{p(x)_{GPP}} \tag{4-4}$$

4-2-4-3 Scaling to multiple cores

Since the PDF algorithm is embarrassingly parallel, significant speedup can be achieved by scaling the design to multiple cores. In this work, dual-core architectures were developed and evaluated to study inherent characteristics for scalability on single- or multiple-device systems. The dual-core results for the 1-D and 2-D cases are summarized and compared to the single-core

implementation in Table 4-6. Design frequencies of 150MHz and 70MHz were obtained for the 1-D and 2-D algorithm designs, respectively. The reduction in frequency was primarily because of increased routing delays attributed to a larger design. Consumption of a greater percentage of DSP48 slices is also bound to increase routing delays as the DSP48 slices are located at specific areas across the IC. An important point to note here is that speedup does not double in a dual-core design (as compared to the single-core design) due to interconnect contention as discussed in Section 4-2-2-2 and a reduced frequency in the 2-D PDF case. These parameters would potentially differ from one platform to another depending upon the system interconnect bandwidth and its architecture. Also, the dual-core 1-D PDF design offered a higher increase factor in speedup when compared to the dual-core 2-D PDF design because time spent on data communication could be effectively hidden under time spent on computation in the 1-D PDF case. This outcome was not achievable in the 2-D PDF designs due to longer communication times (columns of the PDF matrix are sent back to the host after every call to the FPGA) and more importantly, a reduced frequency of operation.

4-2-4-4 Portability considerations

In addition to having scalability impacts, application characteristics also contribute greatly to platform selection and performance. In this work, optimum choice of the number of kernels per core, *k*, should be made based upon the interconnect bandwidth and FPGA resources available. An important byproduct of the RAT methodology, not mentioned earlier, is the computation and communication utilization factors (Equation 4-5).

$$util_{comm} = \frac{t_{comm}}{t_{comm} + t_{comp}}; \quad util_{comp} = \frac{t_{comp}}{t_{comm} + t_{comp}} \tag{4-5}$$

These metrics indicate the nature of a particular design in terms of whether it is communication-bound orcomputation-bound. If there are a variety of platforms to choose from, the designer could make use of these metrics to modify architecture selection for achieving optimum results. The primary resource-limiting factor in increasing the value of *k* in this algorithm design is the number of dedicated multiplier blocks in the FPGA. While platforms with FPGAs having more multipliers (e.g., the Cray XD1 with Virtex-2 Pro FPGAs) and faster theoretical interconnects (e.g., RapidArray in XD1) might intuitively suggest improvements in speedup by performing more computations in the PDF algorithm in parallel, these could be negated by poorread and write efficiencies (α_{read}, α_{write}) during data communication. This case was particularly true with the Cray XD1 system. Due to extremely low read speeds on CPU-initiated transfers from FPGA-host memory (~4MB/s for small data transfers), RAT predicted a smaller speedup number (Table 4-7) in migrating the 2-D PDF algorithm to the XD1 system even though it housed a theoretically faster interconnect. In comparison to the Nallatech platform, the utilization factors in Table 4-7 illustrate the fact that in the XD1 more time is spent in data communication as compared to algorithm computation for the 2-D PDF design. On the contrary, since the computed PDF values are sent to the host after all FPGA iterations are complete, fewer read operations are required in the 1-D PDF design resulting in the XD1 offering more than double the speedup achieved on the Nallatech platform.

4-2-4-5 Scaling to multi-FPGA systems

With emerging multi-FPGA systems, it is also desirable to analyze the potential for scaling the PDF designs across multiple FPGAs and potentially achieve higher speedup. The 2-D PDF single-core design was replicated across a cluster of Nallatech cards connected over infiniband (Figure 4-10). Unified Parallel C (UPC) was used to develop the system-level code wherein the entire application data is distributed equally (i.e., shared) among nodes in the cluster. The

additional communication times involved in the distribution and collection of data along with the attained speedup for different cluster sizes (number of Nallatech cards) is shown in Table 4-8. As the number of FPGAs increase, the number of iterations required to process all application data decrease. Speedup eventually drops with increasing number of nodes as time spent on gathering PDF results from nodes increases.

4-3 Composite Patterns for Design Reuse

As discussed earlier, it is often the case that most of the estimation frameworks (inclusive of the BN framework presented in this work) employ different algorithms at different stages of data analysis/processing. To be highly productive in creating efficient hardware designs for each of those algorithms, it is essential that we reuse existing hardware designs. One of the goals of design patterns is to provide the framework to realize this concept. In this section, the commonalities between the algorithms described in Section 2-4-4 and the similarities in the composite patterns describing their decompositions are exploited by reusing the structural and communication architecture developed for PDF estimation. Also, state transition diagrams (Figures 4-3 and 4-7) primarily deal with managing data flow and activate the computations to execute at appropriate times. This task is particularly cumbersome to design and debug, and any means of reducing this burden would be helpful to designers. Due to similarities in algorithm decomposition, the state transition diagram developed for PDF estimation can be effectively reused for the other two algorithms.

4-3-1 K-Means Clustering

K-means clustering can be decomposed in a similar fashion to that of PDF estimation. Computation of the Euclidean distance between every data sample and the *k* cluster centers can be done in parallel. The minimum distance indicating the cluster closest to the data sample can then be obtained hierarchically by comparing the Euclidean distances pair-wise in a parallel

fashion (Figure 4-11). The basic update rule in K-means clustering for the cluster to which the new sample belongs can be defined as $\Delta center = \eta\,(sample_{new} - center_{old})$, where η is the learning rate [65]. In this work, the cluster centers are updated according to Equation 4-6, as and when a new data sample is assigned to a particular cluster.

$$center_{new} = \frac{center_{old} + sample_{new}}{2} \tag{4-6}$$

This approximate update rule (where $\eta = 1/2$) is often used for on-line clustering and is applicable to real-time scenarios in which fast reaction to changing statistics is often desirable. The update method reduces the computational requirements to addition and right shift operations and eliminates the more resource-consuming division operation. While this offers an efficient hardware implementation, it can lead to slower convergence for broadly distributed clusters due to its relatively high sensitivity to each new sample. In such cases, the traditional approach proposed by MacQueen [65] can be used where $\eta = 1/n_j$ (here, n_j is the number of data points in class S_j). This choice for η achieves a more gradual updating of cluster centers as the number of data samples presented to algorithm grows. The design developed in this work can be extended to employ MacQueen's approach with minimal resource usage by the inclusion of a multiplier and look-up table (for storing η values) to implement the update rule. All the computations involved in the algorithm can further be implemented as a pipeline. Analyzing the composite pattern for decomposing K-means in Figure 4-11, it can be inferred that the architecture developed for PDF estimation can be effectively reused for K-means clustering with minimal modifications as explained in the following section. The cluster centers (I_2) are used to *seed* the parallel datapaths in Figure 4-1A (for the PDF estimation algorithm) and the data samples (I_1) are fed into the pipelines sequentially to determine their cluster association. The number of parallel datapaths and the length of I_2 are both increased from 8 (in the architecture of PDF

estimation) to 64, corresponding to the number of clusters. Upon convergence, the updated cluster centers are read back by the host. Figure 4-11 shows the architecture for performing K-means clustering on a 1-D dataset (N=102,400 and number of iterations for convergence of the algorithm T=100) over 64 clusters. In the first pipeline stage, the Euclidean distance between the cluster centers and a data sample is computed. In the second stage, the cluster closest to the sample is inferred and in the third stage, the inferred cluster center is updated according to Equation 4-6. It can be inferred that only the pipeline computations are modified in Figure 4-1B to implement the clustering algorithm while retaining most of the communication fabric. More importantly, the connection to the middleware design, which is platform-specific and often a time-consuming process, remains unchanged. For the 1-D case (i.e., d=1) the architecture reduces the computational complexity of the algorithm from $O(NmT)$ to $O(NT)$, where m is the number of clusters. This reduction in complexity is attainable as long as there are enough multiplier resources (m multipliers) on the FPGA. The experimental speedup and utilization for K-means clustering algorithm on the Nallatech platform is shown in Table 4-9. The speedup obtained is considerably low for the extent of parallelism extracted in the algorithm (i.e., all operations required on a data sample are performed in one cycle). The speedup is primarily affected by the overhead time spent in transferring data from the CPU to the FPGA over a relatively small-bandwidth low-efficient interconnect. While only 400 iterations (N_{iter} in the RAT worksheet) of FPGA communication and computation were required for PDF estimation, a total of 100 × 200 iterations are required for K-means clustering (200 for processing all available data and 100 for algorithm convergence) leading to a considerable increase in total RC time ($t_{RC} = N_{iter} \times (t_{comm} + t_{comp})$). The $util_{comm}$ factor for K-means clustering is slightly higher in comparison to that obtained for 1-D PDF estimation indicating a longer time spent in data communication.

4-3-2 Correlation

The architecture developed for PDF estimation is also a good fit for computing correlation on an FPGA. The computations involved in calculating the cross-correlation value at different lags τ can be decomposed and mapped in a similar way as that of PDF estimation for a parallel implementation (Figures 4-12B and 4-12C).

The similarity in dataflow between the two algorithms cannot easily be deciphered until we analyze a decomposition strategy for Correlation via design patterns. Consider two vectors $X=[x_1,x_2,x_3,\ldots,x_{512}]$ and $Y=[y_1,y_2,y_3,\ldots,y_{512}]$ whose cross-correlation needs to be computed. A number of product terms involved in the computation can be performed in parallel. In particular, the x_i values (I_2) *seed* the parallel datapaths in Figure 4-1A while the y_i values (I_1) are fed sequentially in a pipeline flow. Due to resource limitations (multipliers) on the FPGA, correlation values are computed over 64 parallel datapaths (i.e., in blocks of 64 values of x_i) and accumulated until all x_i values are accounted for. While accumulating values every cycle, the write address of output memory O is incremented by one to match the raster-like pattern observed in computing correlation at various lags (Figure 4-12A). Since the underlying architecture is scalable and portable (Sections 4-2-4-3 and 4-2-4-4), the designs can be extended to solve larger scale problems (i.e., clustering data points into more clusters/computing correlation over longer datasets) and across various platforms having b igger FPGAs with minimal efforts. The experimental speedup and utilization for the correlation algorithm on the Nallatech platform is shown in Table 4-9.

Table 4-1. List of primitive patterns and their description

Pattern name	Description
Computation	
Pipeline	Increases throughput by overlapping execution of ind.computations
Datapath replication	Exploits parallelism by duplicating computational kernels
Loop fusion	Alleviates potential pipeline stalls in nested loops by fusing them
Tree	Provides computational structure for implementing tree-based algs
Mesh	Provides templates for implementing image-based or 2-D problems
Look-up tables	Reduces run-time comp. of complex operators by simpler lookup ops
Communication	
Scatter	Distributes data among many computational kernels
Gather	Collects data (results) from many computational kernels
Broadcast	Replicates data to many computational kernels
Round robin	Scatters equally-sized data blocks in sequential order among all kernels
Memory resolution	Resolves memory contention by inserting pipeline stages & buffers

Table 4-2. Parameters of primitive patterns

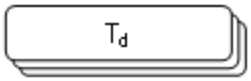
Pattern	*Throughput* (ops/cycle)	*Latency* (cycles)	Notation
Pipeline	T_p = no. of pipeline stages	L_p = no. of pipeline stages	
Datapath replication	T_d = no. of replications	N/A	
Broadcast	N/A	N/A	
Scatter	N/A	L_s = buffer length	
Gather	N/A	L_g = buffer length	

Table 4-3. RAT input parameters for analysis of 1-D and 2-D PDF estimation algorithms

Dataset parameter	Units	1-D PDF	2-D PDF
$N_{elements}$, input	Elements	512.00	1024.00
$N_{elements}$, output	Elements	1.00	65536.00
$N_{bytes/element}$	Bytes/element	4.00	4.00
Communication parameters			
$throughput_{ideal}$	MB/s	1000.00	1000.00
α_{write}	$0<\alpha<1$	0.37	0.37
α_{read}	$0<\alpha<1$	0.16	0.16
t_{comm}	Sec	6.0E-6	1.6E-3
Computation parameters			
$N_{ops/element}$	ops/element	768.00	393216.00
T_d	ops/cycle	8.00	16.00
$Latency_{net}$	Cycles	515.00	131072.00
T_p	ops/cycle	3.00	3.00
f_{clock}	MHz	150.00	100.00
t_{comp}	Sec	1.2E-4	4.3E-2
t_{GPP}	Sec	0.58	158.80
N_{iter}	Iterations	400.00	400.00
Predicted speedup		11.50	9.00

Table 4-4. Speedup, power, and utilization for single-core designs

Description	1-D PDF	2-D PDF
DSP48s (%)	8.0	16.0
BRAMs (%)	12.0	15.0
Slices (%)	11.0	13.0
Power (mW)	1552.0	1595.0
Actual speedup	7.8	7.1

Table 4-5. Performance factors for single-core designs

Description	Predicted	Predicted	Actual	Actual
Algorithm	1-D PDF	2-D PDF	1-D PDF	2-D PDF
t_{comm} (sec)	6.0E-6	1.6E-3	2.5E-5	1.1E-2
t_{comp} (sec)	1.2E-4	4.3E-2	1.4E-4	4.4E-2
t_{RC} (sec)	5.1E-2	1.7E+1	7.4E-2	2.2E+1
Actual speedup	11.5	9.0	7.8	7.1

Table 4-6. Speedup and utilization for dual-core designs

Description	Single-core	Single-core	Dual-core	Dual-core
Algorithm	1-D PDF	2-D PDF	1-D PDF	2-D PDF
DSP48s (%)	8	16.0	16.0	33.0
BRAMs (%)	12.0	15.0	15.0	21.0
Slices (%)	11.0	13.0	16.0	22.0
Actual speedup	7.8	7.1	13.4	8.3

Table 4-7. Speedup and utilization for single-core designs across platforms

Description	Nallatech	Nallatech	Cray XD1	Cray XD1
Algorithm	1-D PDF	2-D PDF	1-D PDF	2-D PDF
$util_{comm.}$	0.14	0.20	0.03	0.38
$util_{comp}$	0.86	0.80	0.97	0.62
RAT predicted speedup	11.50	9.00	13.00	3.70
Actual speedup	7.80	7.10	13.40	8.30

Table 4-8. Speedup for single-core 2-D PDF design across multiple FPGAs

No. of nodes	2	4	6	8	10
$t_{scatter}$ (sec)	0.176	0.177	0.183	0.174	0.175
t_{gather} (sec)	0.950	2.800	4.667	6.521	8.383
N_{iter}	200.000	100.000	67.000	50.000	40.000
t_{RC} (sec)	16.140	10.400	9.750	10.340	11.420
Speedup	9.900	15.300	16.300	15.400	13.900

Table 4-9. Speedup and utilization for PDF estimation, K-means clustering, and Correlation (single-core designs)

Description	1-D PDF	2-D PDF	K-means	Correlation
DSP48s (%)	8.00	16.00	66.00	66.00
BRAMs (%)	12.00	15.00	9.00	11.00
Slices (%)	11.00	13.00	9.00	11.00
$util_{comm.}$	0.14	0.20	0.18	0.21
RAT predicted speedup	11.50	9.00	4.30	9.60
Actual speedup	7.80	7.10	3.20	8.10

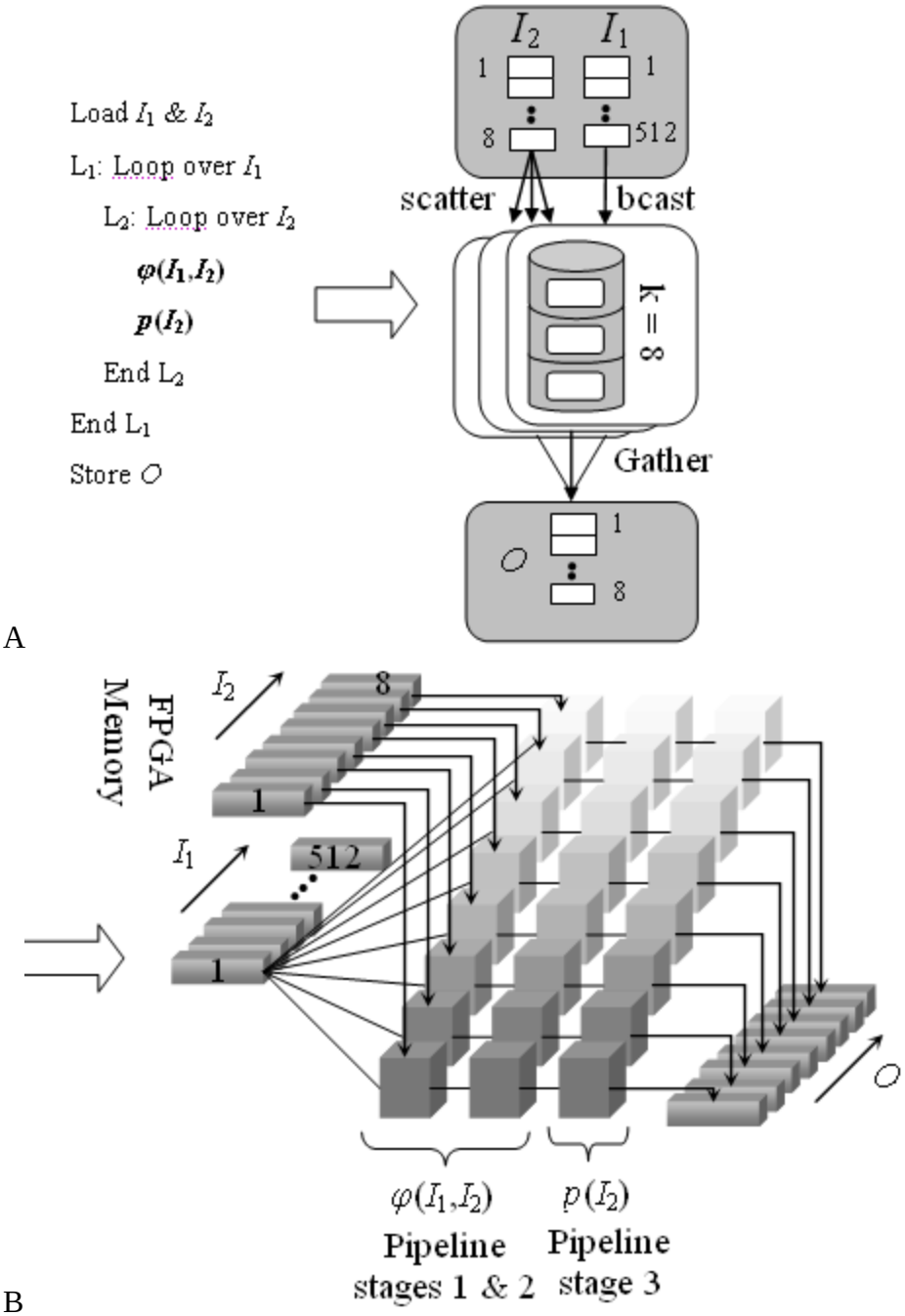


Figure 4-1. Pattern-based analysis of 1-D PDF estimation algorithm. A) Decomposition. B) Design.

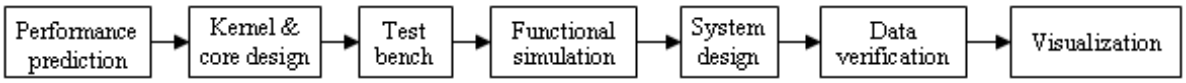


Figure 4-2. Development stages in system design.

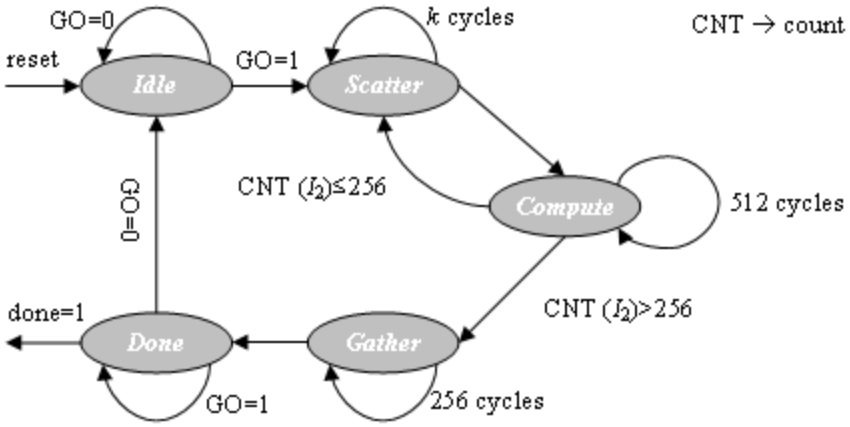


Figure 4-3. State transition diagram for a single-core 1-D PDF design.

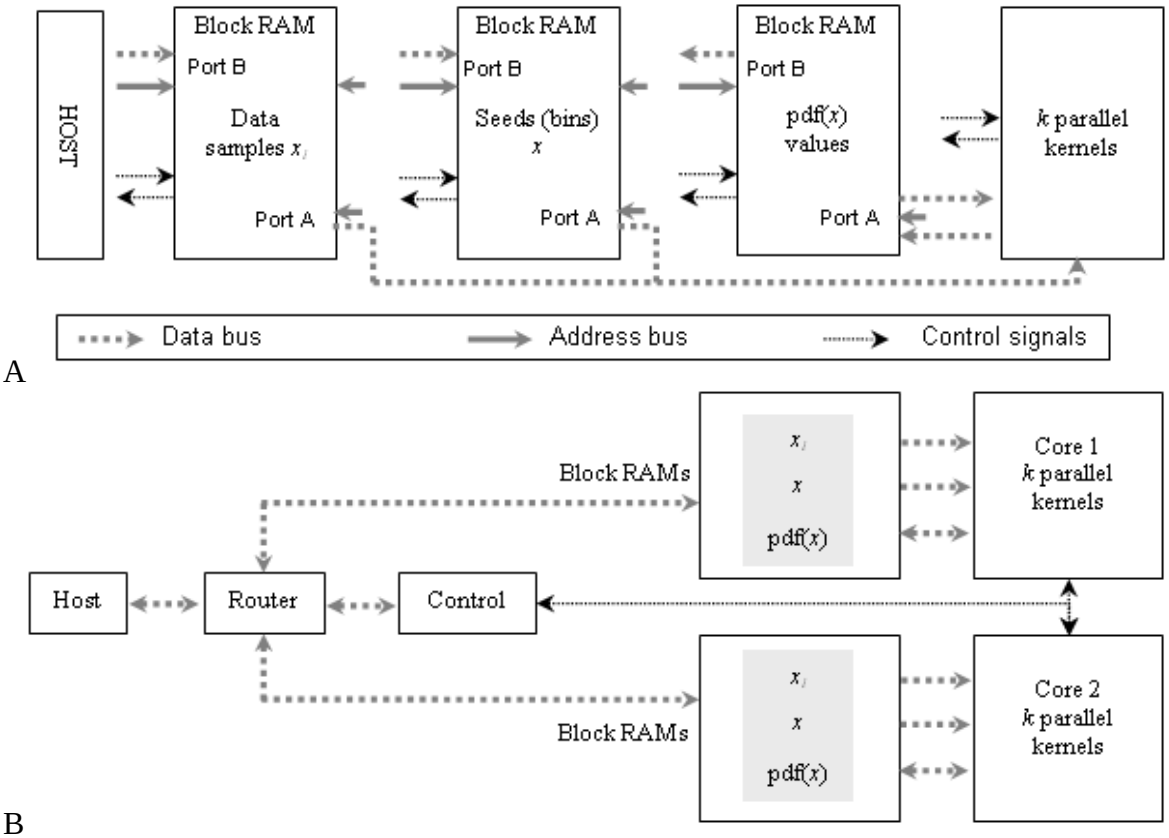


Figure 4-4. Design details of 1-D PDF. A) Single-core design. B) Dual-core design.

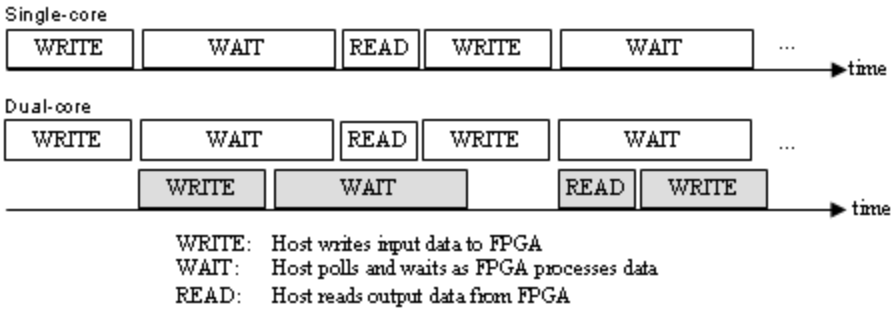


Figure 4-5. Host-centric execution flows. A) Single-core design. B) Dual-core design.

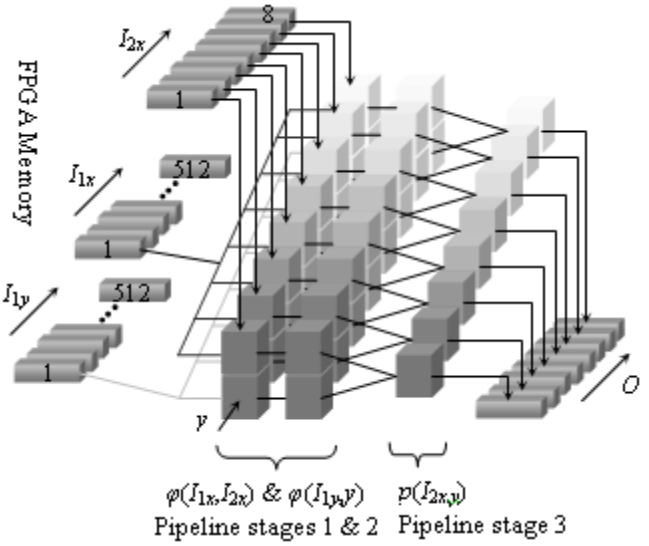


Figure 4-6. Architecture design for 2-D PDF estimation algorithm.

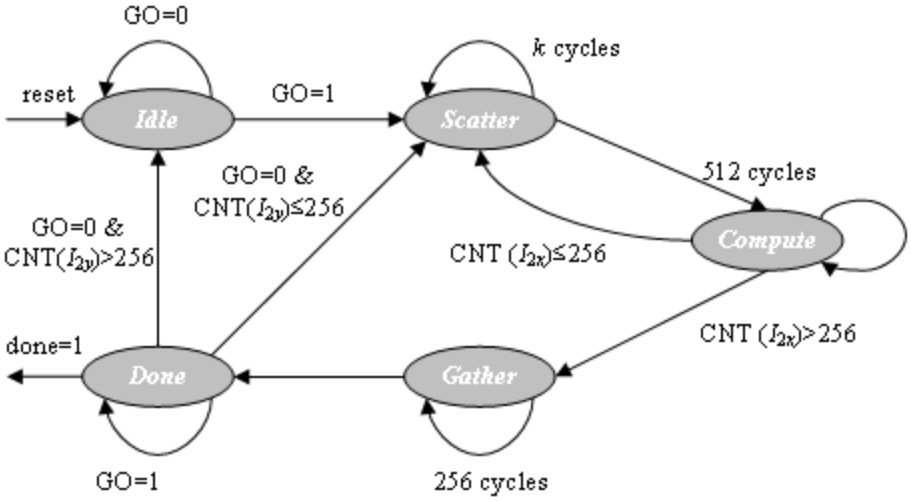


Figure 4-7. State transition diagram for a single-core 2-D PDF design.

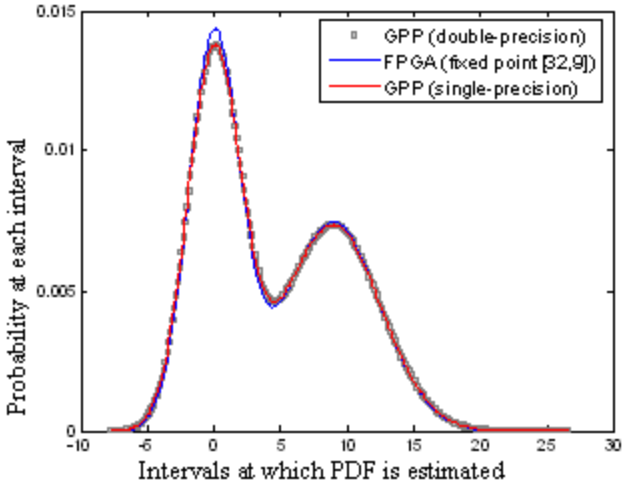


Figure 4-8. 1-D PDF estimates on GPP and FPGA.

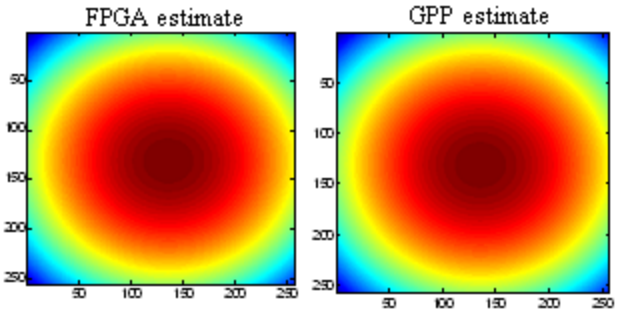


Figure 4-9. 2-D PDF estimates on GPP and FPGA.

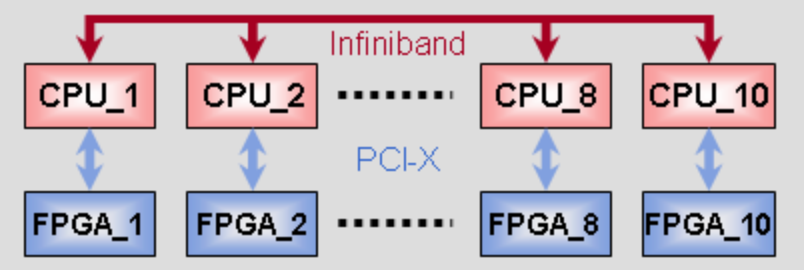


Figure 4-10. Multi-FPGA system configuration.

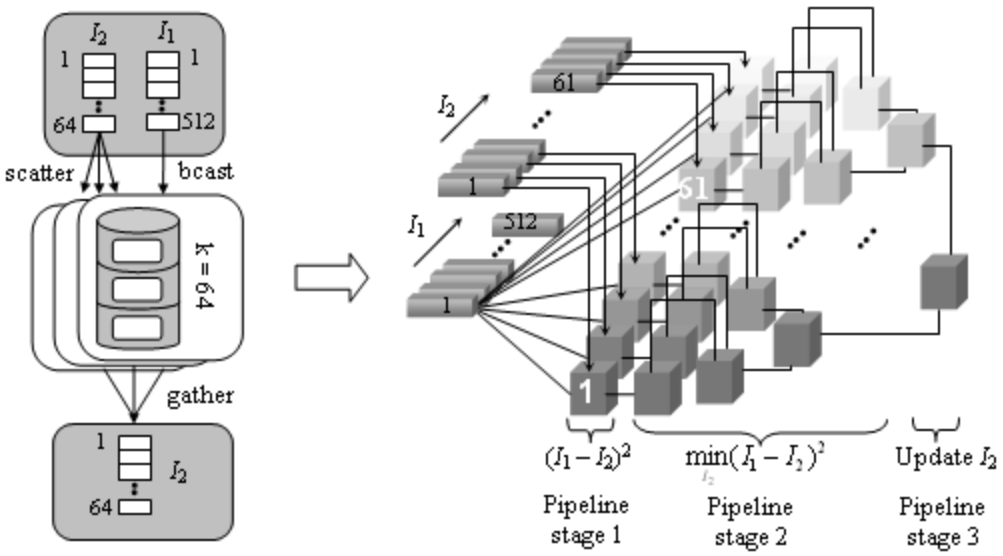


Figure 4-11. Architecture for K-means clustering.

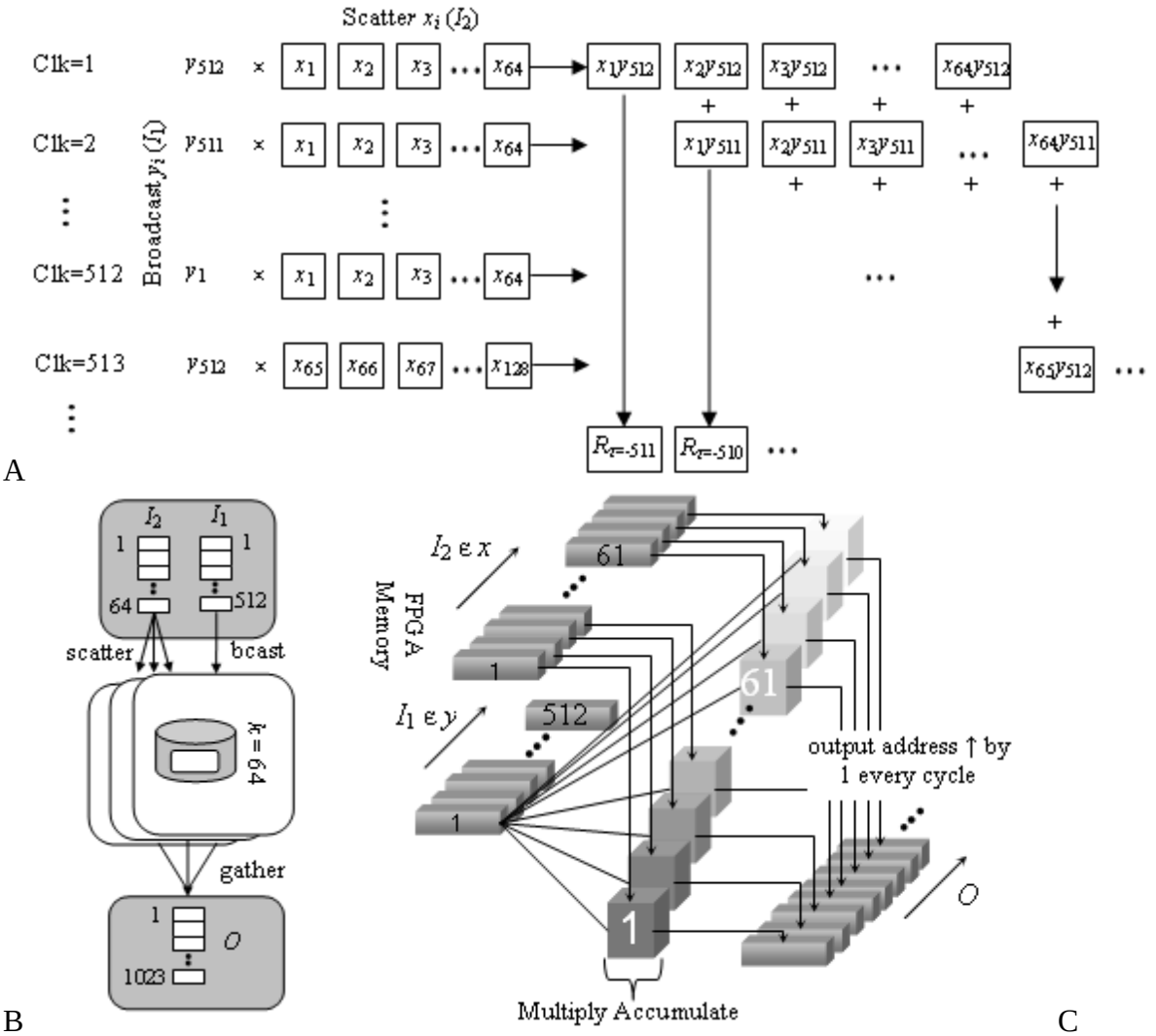


Figure 4-12. Decomposition and design of correlation. A) Computation structure. B) Pattern-based decomposition. C) Design of correlation.

CHAPTER 5
MULTISCALE ESTIMATION FRAMEWORK

While not strictly part of the BN or RC-accelerated work discussed so far, we take a brief look at the problem of observations at multiple resolutions. The problem of streamflow estimation is further exacerbated when measurements (image data in particular) are made at a wide range of scales. In such cases, one would want to get all the features to a common spatial resolution (through a downsample/upsample operation). Most often, this is a scale at which the spatio-temporal features are measured (e.g., the NexRad rainfall estimates). Surface morphology (elevation, slope, and curvature) and land cover (vegetation type and density) strongly influence the distribution of precipitated water among the major mechanisms of infiltration, increased soil moisture, evaporation, and runoff (Figure 5-1). Precipitation, topography and surface morphology measurements could be embedded into a state-space model for estimating various hydrologic boundary conditions across scales. By understanding multiscale variations, fine-scale features can be estimated from sparse multi-source measurements observed at multiple scales. This could prove beneficial for the integrated study of streamflow from small sub-basins (catchments) at urban scales all the way up to the larger watersheds in which the small catchments reside. Erosion rates and information regarding flooding/land slide risk are affected by shapes of stream banks and hill slopes. These features are often small in spatial scale (meter to sub-meter) and so require fine-scale measurements over large areas. Yet, when estimating the long-term flow behavior at the outlet of an entire watershed, the larger scale topography that defines contributing area is more important. As a proof-of-concept, estimation of topography at multiple scales is chosen as a case study in this work.

The elevations of many natural surfaces, including the topography of the Earth, have long been recognized to approximate self-affine fractals [66, 67]. A statistically self-affine fractal

surface can be thought of as a fractal surface that is not isotropic. In a two-dimensional *xy*-space, a self-affine fractal $f(x,y)$ is statistically similar to $r^H f(rx,ry)$, where r is a scaling factor and H is the Hausdorff measure. Such surfaces can be modeled as fractional Brownian motions (fBm) since their power spectra exhibit power law dependence on the wave number of spatial frequency [68, 69, 1, 70]. Multiscale Kalman Smoothers (MKS) with embedded fractional Brownian motion stochastic models have been used previously to estimate topography [1] from multi-resolution topographic data. However, the standard MKS algorithm with a single embedded stochastic model is not sufficient to incorporate spatial variations in the elevation statistics; for example, a rough undulating terrain yields an elevation surface with a shorter correlation length than a flat smooth terrain and hence a different power spectrum as well. In [68], the author emphasizes the need for different terrain descriptors for different terrain types and an efficient multiscale representation. Although the $1/f$ assumption tends to be a reasonable approximation over large areas (>10 km^2), it is often a poor approximation over smaller areas (<10 km^2), especially if vegetation is present and modulates the topographic measurements.

The inclusion of multiple models in the MKS estimation framework has been employed in this work and is found to give better estimates of topography over non-stationary terrain than the standard MKS algorithm [71]. A mixture-of-experts (MOE) network developed in [72] is employed to weigh the individual model outputs based on the Kalman filter residuals (known as the innovations). Fractal theory concepts are employed for estimating the parameters of the models by finding the fractal dimension of the image over different regions. Fractal-based measures have been previously used in classification and segmentation algorithms [69, 70, 73, 74]. In particular, the relation between fractal dimension (related to Hurst exponent) and roughness has been verified in [70, 75] through psychophysical studies. Authors in [69, 76]

propose the use of multiple hurst parameters for representing topographic like datasets deviating from the *1/f* property. This work not only acknowledges the need for multiple models to effectively represent topography but also incorporates such a model in an MKS framework that further handles uncertainties in the observations and hence aids in better segmentation of terrain at multiple scales. Although the estimates of topography are improved, the primary motivation for employing a multiple-model MKS (MM-MKS) is for segmentation of terrain images based solely on spatial variability of the elevation surface. This is important when no other information (e.g., multispectral) is available. The importance of multiscale terrain analysis in remote sensing applications is discussed in [77].

5-1 Test Area

The study area analyzed for this work is a mix of coniferous and deciduous forest in North-central Florida, USA. The forest is part of the Hogtown Greenway within the city of Gainesville, so urban development (roads and houses) appears along the forest edge, as shown in Figure 5-2.

The study site was imaged with an ALSM sensor owned by the University of Florida (UF) from an above ground altitude of 600m. The last stop returns represent penetration on targets that have sparse structure at the scale of a few tens of centimeters and are used for this analysis. The 3D point cloud obtained is gridded into different spatial resolutions to produce images of varying resolutions.

5-2 Multiscale Estimation Framework

The multiscale models of focus in this work were proposed in [1] and provide a scale recursive framework for estimating topographies at multiple resolutions. These models are defined on index sets and organized as multi-level quadtrees. Multiscale estimation is initiated with a fine-to-coarse sweep up the quadtree that is analogous to Kalman filtering with an added merge step. The fine-to-coarse sweep up the quadtree is followed by a coarse-to-fine sweep

down the quadtree that corresponds to Kalman smoothing. This algorithm is referred to as MKS. The linear coarse-to-fine model in [1] is given in Equation 5-1 where s is the node index on the tree, and B is the backshift operator in scale such that Bs is one scale coarser than s.

$$\begin{aligned} x(s) &= A(s)x(Bs) + B(s)w(s) \\ y(s) &= H(s)x(s) + v(s) \end{aligned} \tag{5-1}$$

Here, $x(s)$ is the state variable (elevations), and $y(s)$ represents the ALSM observations. Random process $w(s)$ is assumed to be a Gaussian white noise process with identity variance, and the measurement error $v(s)$ is a Gaussian white noise process with variance $R(s)$. $A(s)$ is the coarse-to-fine state transition operator, $B(s)$ is the coarse-to-fine process noise standard deviation and $H(s)$ is the measurement-state relation. The scale is represented by the level in the quadtree, and is denoted by m. The support of the image at level m is $2^m \times 2^m$.

The standard Kalman formulation provides optimal estimates (in the mean squared sense) when there is perfect *a priori* knowledge of the state and measurement model parameters $A(s)$, $B(s)$, and $R(s)$. The state process is assumed to follow a *1/f* model. Using this model, the power spectrum of the state variable $x(s)$ can be represented by the multiscale model in Equation 5-1 by using $A(s)=1$ and $B(s)=B_0 2^{(1-\mu)m/2}$ [1]. The MKS approach provides reasonable estimates for the evolution of the state process in scale. However, the resulting coarse-to-fine process noise variance $B^2(s)$ is constant at each scale; hence, it cannot accommodate non-stationarities in the imagery at a particular scale. The fine-to-coarse process noise variance $Q(s)$ used in the Kalman filter is a function of $B^2(s)$ and is therefore also spatially uniform. An image that contains more than one type of topography is an example of a non-stationary 2*D* process. Therefore, using a data model that is variable in scale but uniform in space will lead to suboptimal estimates in general.

In this work, fractal dimension (*D*) was used to discriminate different classes of terrain based on topographic variation. Spectral matching methods have been widely employed to determine the parameters for a single model but do not give accurate results when applied to multiple models. Fractals, however, provide a good description of natural objects exhibiting statistical self-similarity. In this work, the fractal dimension was calculated using a 9m × 9m window by the Triangular Prism Surface Area (TPSA) method [78]. In order to determine the natural portioning of the ALSM height data, the point cloud data were gridded into an image array with a spatial resolution of 2m × 2m. Small fractal dimension implies smooth areas as can be seen over the road and roof tops. A large fractal dimension characterizes rough areas and is seen over forest regions. The number of fractal models and their respective fractal dimension are calculated by employing the Expectation-Maximization (EM) algorithm on the histogram of the fractal dimension image. The mean value of the Gaussians would then give the fractal dimension of the underlying terrain.

The algorithm, when applied to the data resulted in three different Gaussians (clearly following the tri-modal distribution of *D*). The histogram plot (Figure 5-3) of a dataset generated from the resulting Mixture of Gaussians is very similar to that of the fractal dimension plot. The fractal dimension *D* is related to the Hurst exponent *H* through $H=3-D$ for two-dimensional data. *H*, in turn can be related to the slope parameter μ of the power spectrum density through $\mu=2H+1$ [79]. By virtue of using the Multiscale structure and fBm process, the incremental variances between scales are expressed as given in Equation 5-2 where *W* represents the windowed data used in the sample variance calculation.

$$Var(W)=\sigma^2\left[1+\left(\frac{1}{2}\right)^{2H}+\left(\frac{1}{2}\right)^{4H}+\ldots+\left(\frac{1}{2}\right)^{16H}\right] \tag{5-2}$$

The total variance of the models at the 512 × 512 support level was set equal to the sample variance computed for different regions in the ALSM elevation data. The variance scale factor σ is solved from Equation 5-2. This can then be used to find $B(s)$ at different scales from $\sigma[1,(1/2)^H,(1/2)^{2H},\ldots,(1/2)^{7H}]$ and the value of B_0 can be interpolated by fitting a linear line in log space.

5-3 Mixture of Experts (MOE)

The MOE used in this work is a single-layer network that consists of multiple experts and a gating network that arbitrates among the expert estimators. The MOE assigns weights to the experts in an unsupervised fashion as they compete for the desired response. In this work, each expert system is a multiscale Kalman filter embedded with its own $1/f$ model for a particular class of topography (parameter vector α_i) as shown in Figure 5-4. The most likely model among these filters for a particular input is assigned a higher weight g_i by the gating network. The input to such a network is the observation data set and the desired outputs are the elevation estimates of the topography. The weights can be interpreted as *a priori* probabilities for the corresponding experts for the current input. The softmax transformation (Equation 5-3) forces the condition on the weights to sum to one.

$$g_i = e^{u_i}\left[\sum_{j=0}^{N} e^{u_j}\right]^{-1} \tag{5-3}$$

Here, $u_i=z^T a_i$ and a_i is the weight matrix of the i^{th} filter in the modular network. The filter residuals (innovations) r_k are used to calculate the Gaussian conditional probability f of the measured observations at time k, given the filter realization and the measurements up to time k-1 (Equation 5-4) where $W_k = H_k P_k^- H_k^T + R_k$ and $r_k = z_k - H_k x_k^-$ [1].

$$f(z_k \mid \alpha_i, z_{k-1}) = \left(2\pi \mid W_k \mid\right)^{-0.5} e^{-0.5 r_k^T W_k^{-1} r_k} \tag{5-4}$$

The probability distribution of the filter bank is given as the weighted sum of the individual conditional distributions of the filters (Equation 5-5).

$$f(z_k) = \sum_{i=1}^{N} f(z_k \mid \alpha_i) g_i \tag{5-5}$$

Equation 5-5 can be considered as a likelihood function, which upon maximization would give the best possible estimates. The *a posteriori* probabilities h_i are defined in Equation 5-6.

$$h_i = f(z_k \mid \alpha_i) g_i \left[\sum_{j=1}^{N} f(z_k \mid \alpha_j) g_j \right]^{-1} \tag{5-6}$$

The weight matrix a_i is updated by maximizing Equation 5-6 using a gradient ascent procedure, yielding an update $a_i = a_i + \eta(h_i - g_i)z_k$, where η is a learning rate parameter.

In the case of multiscale image fusion, the filter recursion steps represent different scales and therefore cannot exceed the base-2 logarithm of the finest-scale image support. In traditional data fusion applications, observations are present generally only at two or three scales, which limit the number of iterations that the gating network has to estimate weights. The limited number of iterations requires an adaptive estimation algorithm that can react quickly. The MOE is able to react quickly due to the geometrically decreasing impact of previous measurements. The effect is further alleviated by gridding the ALSM point data to different resolutions hence providing observation sets at more scales.

5-4 Results and Analyses

Simulated data was first used to illustrate the performance benefits of employing multiple models in the MKS framework. Fractals are often used for simulating $1/f$ processes and natural terrain. Figure 5-5 shows one such simulated fractal image embedded with two different terrains. The mean squared error (MSE) obtained from a multiple model – MKS embedded with two models is lower than that obtained by either of the two models alone (Table 5-1). Figure 5-6

shows the weights assigned to each of the two models. The elevation values over the smooth terrain are significantly small and lead to very low residuals in both filter estimates. The rough filter hence still receives moderately high weights over the smooth terrain.

In this work, real ALSM data were gridded to populate the quadtree at scales m={10,9,8,7,6,5,4,3,2,1}. Interestingly, it was observed that real data does not follow a 1/f process as much as the simulated fractal terrain. Incremental variances and empirical $F(s)$ values calculated from the dataset across scales illustrate this fact (Table 5-2). Canopy penetration and presence of tall vegetation in the dataset causes the data collected over large areas (>100s of km^2) to deviate from the 1/f property. Embedded 1/f models are no longer well matched to the state process and hence there is scope for further refinements to the proposed MM-MKS approach. Though the measurement noise covariance $R(s)$, by definition, embeds only the uncertainty in the measuring device, the interception of laser light by the vegetation can be thought of as a source of measurement noise and is embedded into the model parameter $R(s)$. The covariance value is calculated on a per-pixel basis by computing the variance of the points that fall under each grid cell forming the pixel. In this way, the estimators are still modeled under a 1/f assumption with the deviations taken care of by a spatially varying $R(s)$. Ideally, MKS filters should embed non-1/f models to account for the deviation from 1/f behavior, but following the above procedure gives acceptable solutions. Proposing a non-1/f modeling approach requires more elaborate testing and is considered for future research work.

The state transition operator $F(s)$ in the upward sweep of the MKS, defined as the ratio of the state covariance at two consecutive scales (with $A(s)$=1) [1], regulates the estimation of the prior value of the state at each scale based on the embedded model. The variations in the prior

estimates of the filters are hence significantly affected by its value and indirectly by the initialization of the root node covariance P_0.

$$F(s) = \frac{P_s(\gamma s)}{P_s(s)} \quad where \; P_s(s) = A(s)P_s(\gamma s)A^T(s) + B^2(s) \tag{5-7}$$

For large values of P_0, the effects of stochastic details $B(s)$ embedded in the models are diminished and the filters give similar estimates and are assigned similar weights at almost all the scales. Limited recursions in scale imply that the weights have little chance of convergence. For small values of P_0, the estimates of the smooth filter deviate significantly from the observation (due to smaller Kalman gain K) and the rough filter is preferred by the MOE at finer scales (Figure 5-7). Knowledge of the incremental variances across scales gives an intuitive way of initializing the root node state covariance. By virtue of the multiscale structure, the extrapolated sample variance at scale 1 gives the expected root node state covariance ($P_0=B_0^2$).

The Kalman model parameters were specified before the multiple-model MKS estimator was implemented. The fractional Brownian stochastic model dictates that $A(s)=1$. The value for the mapping matrix $H(s)$ is unity in grid cells containing ALSM data and zero in empty grid cells. For quadtree levels containing observations, the measurement noise variance $R(s)$ was specified empirically for each scale. Corresponding to the tri-modal nature of the fractal dimension values, a three-model filter bank was chosen. One model represents low variance (smooth) terrain, another represents high variance terrain and land cover, and the third is tuned to track the intermediate terrain. Parts of the forest encompassing sudden changes from tall trees to bare ground (forest gaps) exemplified the highly varying region (variance=52.56). Road surfaces represented the other extreme (variance=3.5). Forest edges represent the third category (variance=25). These variances are then used to find the value of σ (Equation 5-2) using the corresponding Hurst exponents calculated using segmentation for road, forest, and edges. The

stochastic detail $B(s)$ terms were determined as detailed in Section 5-2. The estimation algorithm starts with the upward sweep wherein the estimates calculated by the individual filters are employed to compute the weights using MOE. The filter and smoothers then continue to run independently of the weighing network. The weights are then used to fuse the Kalman filter estimates in the Kalman smoothing downward sweep. The presence of observations at multiple scales provides several recursion steps for the model weights to adapt. Average weights assigned to each model for different terrain and land cover regimes are listed in Table 5-3. Figure 5-8 shows the weights assigned to each of the three models. Notice that Model 1 receives high weights over the smooth road and relatively low weights over the forest and edges with higher fractal dimension. The converse is true for Model 2. Model 3 reacts to forest edges and the less rough areas. The relative weighting remains consistent across scales. All *a priori* (scale 10) model weights were 0.33.

Table 5-1. MSE obtained over the simulated image

Estimates	MSE (sq.m)	%MM-MKS improvement
MM-MKS	7.1999E-4	N/A
Rough filter	7.6675E-4	6.1
Smooth filter	8.3828E-4	14.1

Table 5-2. Incremental variance across scales and empirical $F(s)$ values

Variance between scales	Variance	Empirical *F(s)*
11-10	9.5621	0.7661
10-9	8.2091	0.8133
9-8	4.2015	0.8825
8-7	4.8402	0.8466
7-6	4.9543	0.8146
6-5	5.0405	0.7684

Table 5-3. Average model weights obtained using MM-MKS for real data

Description	Scale 9	Scale 8	Scale 7	Scale 6
Road				
Model 1	0.4064	0.5619	0.5791	0.6055
Model 2	0.2423	0.1660	0.1578	0.1210
Model 3	0.3513	0.2721	0.2631	0.2735
Highest fractal dimension forest				
Model 1	0.0977	0.0517	0.0165	0.0045
Model 2	0.4555	0.5304	0.8123	0.9135
Model 3	0.4468	0.4099	0.1712	0.0820
Intermediate fractal dimension forest				
Model 1	0.3430	0.3559	0.2649	0.1658
Model 2	0.2891	0.2580	0.3046	0.3175
Model 3	0.3679	0.3861	0.4306	0.5167

Model 1, 2, and 3 represent smooth, rough, and intermediate terrains respectively

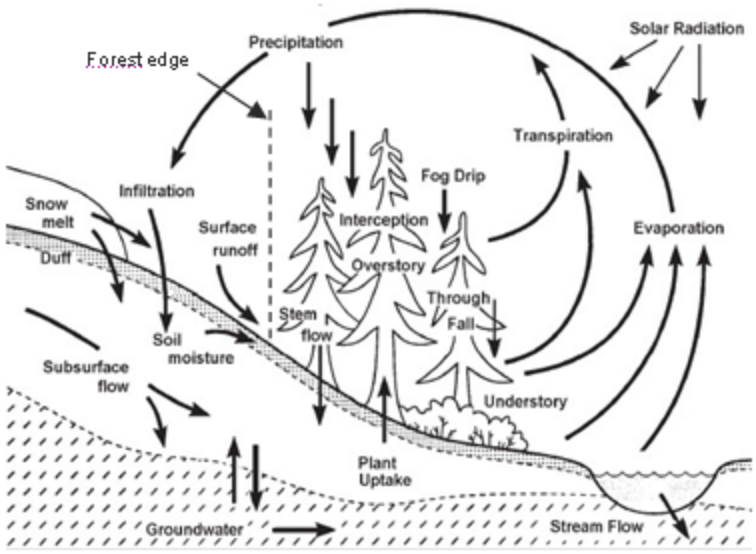


Figure 5-1. Forest hydrologic cycle processes [Pike 2003].

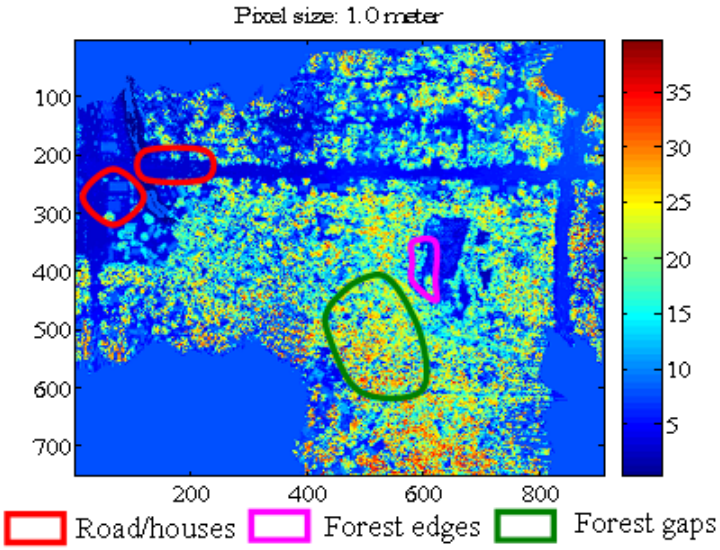


Figure 5-2. Elevation image created by gridding last stop elevation points using an inverse-distance weighting of study area.

Note: Axis and colorbar values in meters.

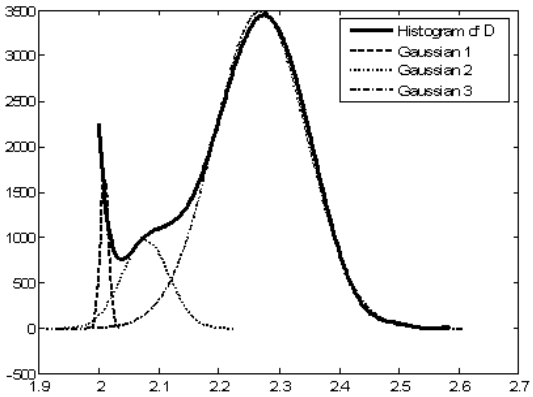


Figure 5-3. Mixture of Gaussians fitted over the histogram plot of fractal dimension.

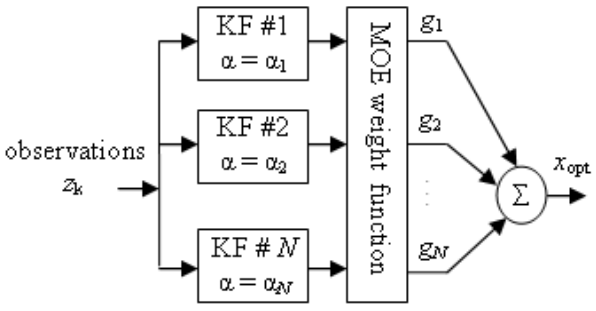


Figure 5-4. Fusion of Kalman filter bank estimates using Mixture-of-experts network.

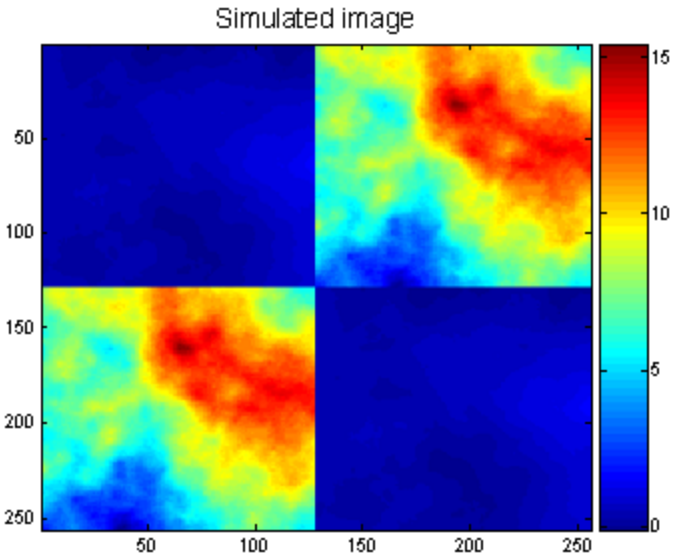


Figure 5-5. Fine-scale simulated image with two terrain roughness.

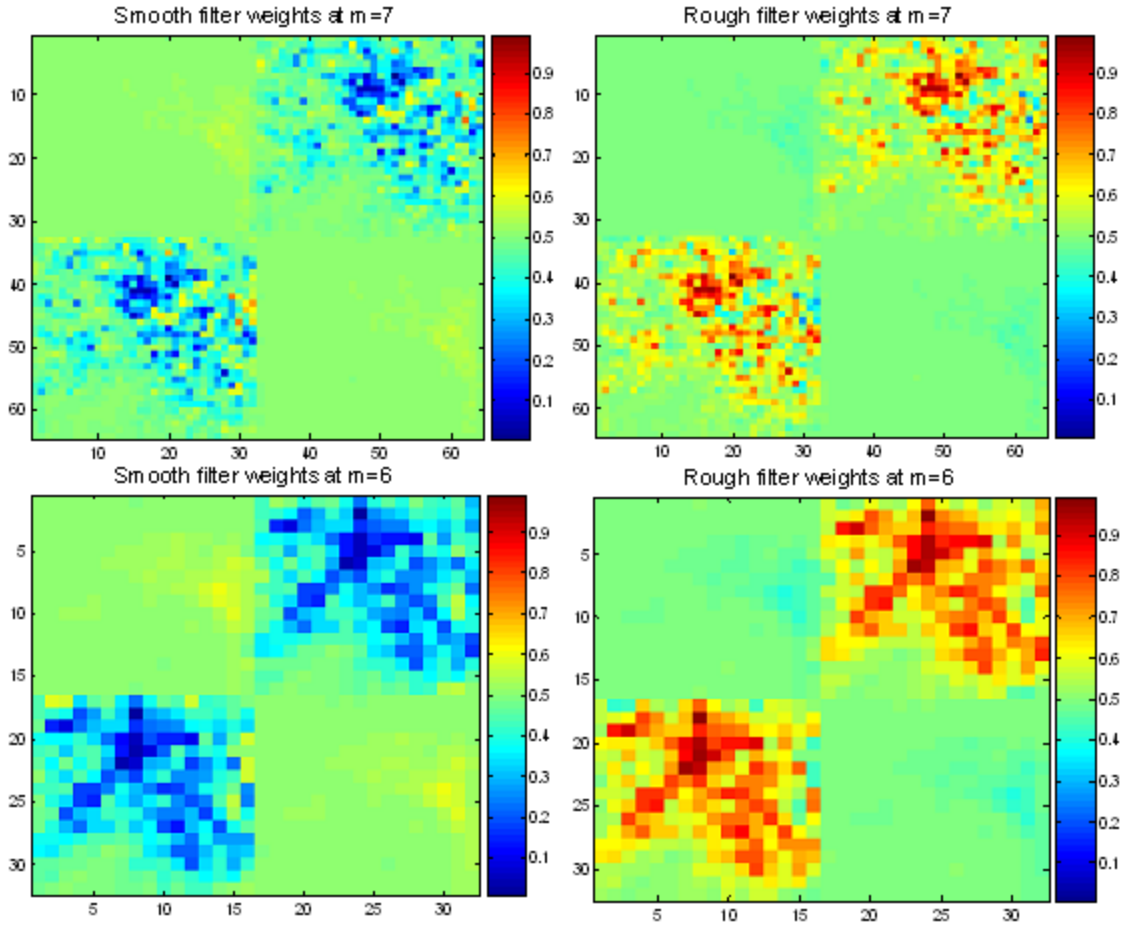


Figure 5-6. Weights obtained for smooth and rough filters at scales m=7 and 6 respectively.

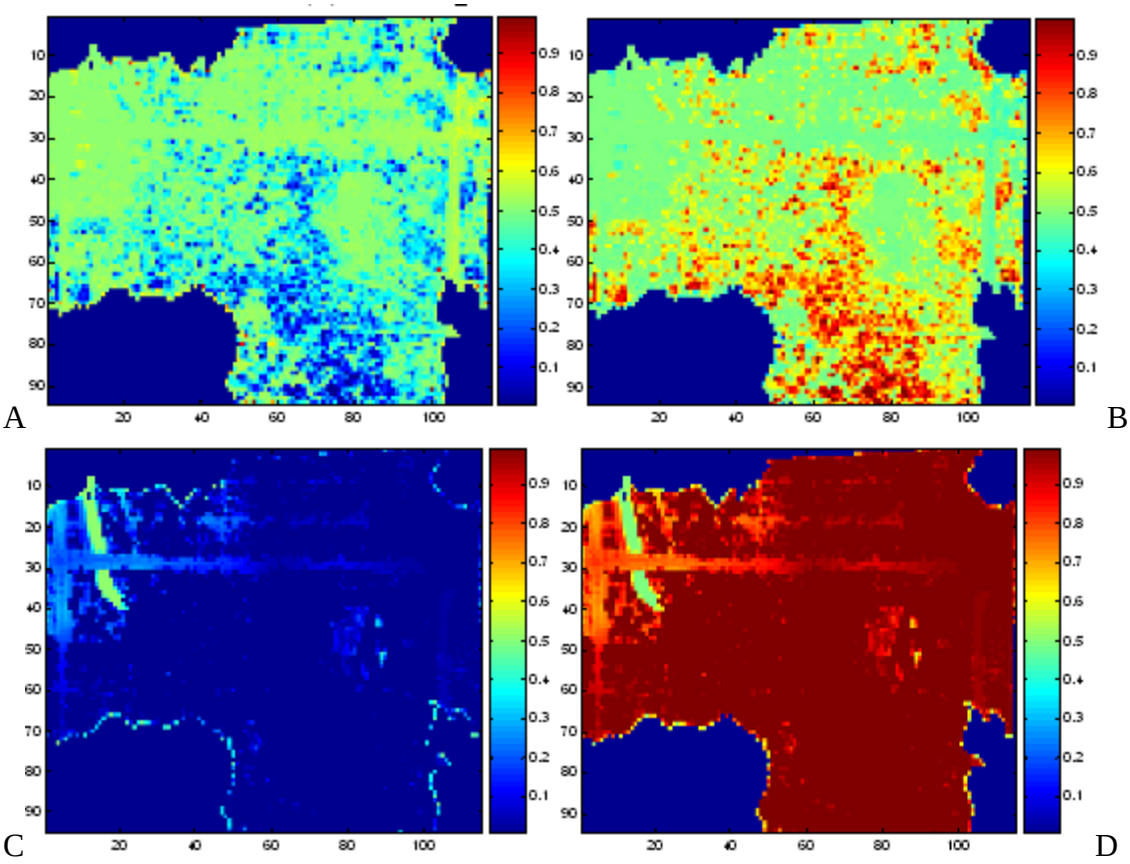


Figure 5-7. Filter weights under large and small P_0. A) Smooth filter under large P_0. B) Rough filter under large P_0. C) Smooth filter under small P_0. D) Rough filter under small P_0.

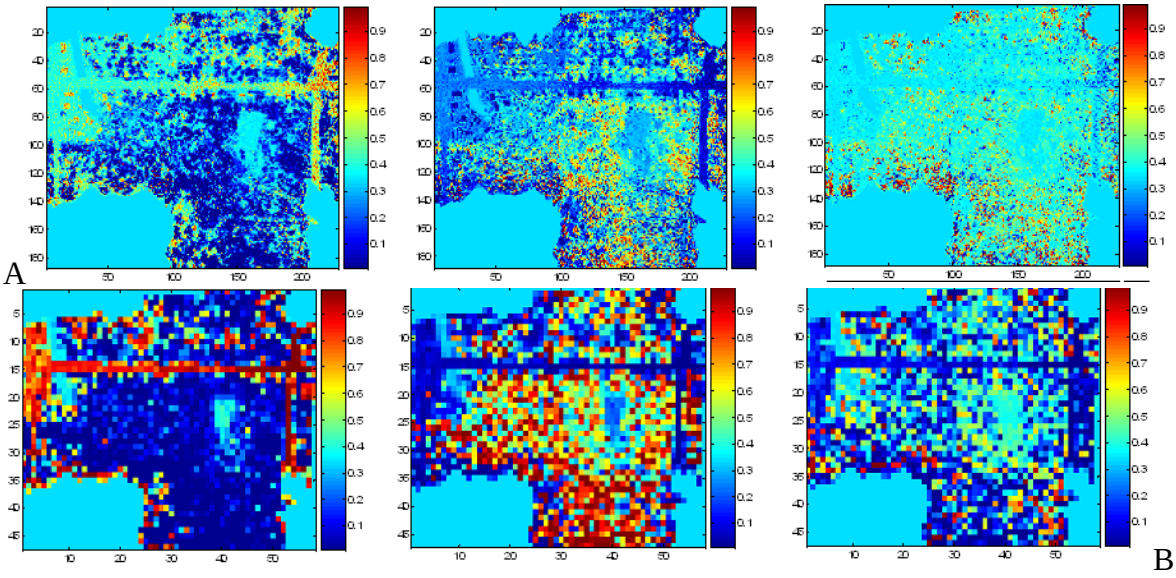


Figure 5-8. Weights assigned to Model 1, 2, and 3 at various scales. A) Scale 9. B) Scale 7.

CHAPTER 6
CONCLUSIONS AND FUTURE WORK

The primary motivation for this work was to develop a practical framework for estimating streamflow in a complex system (i.e., a watershed) monitored by an ad hoc collection of measurement types. A highly scalable BN framework was proposed that allows the fusion of many disparate forms of data that are typically used to characterize hydrologic processes, including remotely sensed images from multiple sensors, GIS layers, point-source time series, and spatio-temporal data, such as meteorological data acquired from multiple ground-based sensors. An information-theoretic methodology, based on conditional entropy was also presented to help quantify the impact of adding nodes to the network in terms of information gain. Posterior probabilities of streamflow estimates and the associated entropy values provided valuable information in quantifying network performance and also offered directions for future watershed instrumentation. The proposed methodology can also quantitatively suggest the most informative features for estimation at any location in the watershed for which there exists feature data and potentially reduce the number of features needed to achieve a specific performance goal. While the network achieves improved performance by exploiting calibrated physical model estimates like WAM, it also offers the possibility to account for inherent non-stationarities in space and time by incorporating both spatial and temporal input features in the absence of WAM and achieve comparable performances. The framework is spatially explicit such that it can be applied to any arbitrary location along the reach of a stream or river.

The networks produced accurate predictions in the confined region (stations 2321000 and 2321500). Sub-surface interactions in the unconfined region (station 2322500) necessitate a more complex network to sufficiently capture streamflow variability. The most important improvement to be made in the near future will be to further expand the network graph to allow

additional features to be considered. In particular, data related to sub-surface interactions and evapotranspiration (e.g., daily temperatures) are expected to significantly improve estimator performance. A graphical model constructed by representing spatial stations as a chain of nodes along the river would potentially offer further performance improvements by fusing and propagating the impact of new evidence (streamflow measurements) and beliefs efficiently via message passing.

6-1 Message Passing

Message passing deals with fusing and propagating the impact of new evidence and beliefs through BNs. It does so via a self-activated propagation mechanism wherein the network is considered as an array of simple and autonomous processors which communicate locally via the links provided by the network [80]. The impact of any new evidence is viewed as a perturbation that propagates through the network via message passing between neighboring variables.

Consider the network $X \rightarrow Y$. If evidence $e=\{Y=y\}$ is observed, then from Bayes' rule, the belief distribution of X is given by Equation 6-1 where $\alpha=P(e)^{-1}$, $P(x)$ is the prior probability of X, and $\lambda(x)$ is the likelihood vector $\lambda(x)=P(e|x)=P(Y=y|x)=M_{y|x}$.

$$BEL(x) = P(x \mid e) = \alpha P(x)\lambda(x) \tag{6-1}$$

As it can be inferred, the parameter $\lambda(x)$ can be computed at node Y (from the knowledge of the probability density function $P(Y=y|X=x)$) and transmitted to X, enabling X to compute its belief $BEL(x)$. To understand the propagation scheme, consider the network $X \rightarrow Y \rightarrow Z$ with the evidence $e=\{Z=z\}$. The likelihood vector $\lambda(x)$ can no longer be directly obtained from the matrix $M_{y|x}$, however, but must reflect the matrix $M_{z|y}$ as well (Equation 6-2).

$$\lambda(x) = P(e|x) = \sum_y P(e|y,x)P(y|x) = \sum_y P(e|y)P(y|x) = M_{y|x}\lambda(y) \tag{6-2}$$

Generalizing the chain of networks, every node can calculate the correct current value of its λ vector if it learns the correct λ vector of its successor. Since the chain ends with an observed variable whose value is determined externally, the λ vector of all variables can be determined recursively. If the chain ends with an unobserved variable Z, $\lambda(z)$ is set to one and all belief distributions will coincide with the prior distribution. Assuming that each node constantly inspects the λ of its child and updates its own λ accordingly, it is guaranteed that every variable along the chain will obtain its correct λ, properly reflecting any changes that might have occurred in e.

A dual-parameter (λ and π) communication/message passing is required when new evidence emerges from both a descendent of a node and its ancestor (i.e., evidence at both the head and tail of the chain). Consider the network in Figure 6-1A where e^+ and e^- represent evidences available at nodes T and Z (shaded) – nodes in X's parent and child subsets. In this case, the two evidences are handled by two separate vectors (λ and π) and propagated along the chain independent of the other (Equation 6-3) where $\lambda(x)$ remains as defined before (Equation 6-2) and $\pi(x)= \pi(u)M_{x|u}$. Belief propagation using the bi-directed message passing is represented in Figure 6-1B.

$$\begin{aligned} \pi(x) &= P(x|e^+) \\ \lambda(x) &= P(e^-|x) \\ BEL(x) &= \alpha\lambda(x)\pi(x) \end{aligned} \tag{6-3}$$

6-1 Streamflow Estimation via Message Passing

The test points along the river (T_1, T_2, etc…) at which we estimate streamflow form the chain network. In message passing, *probability vectors* (λ and π) are transmitted between nodes. If nodes in the network represent flow, the process would involve passing around flow probabilities. This would imply discretising flow that has a relatively large dynamic range.

Instead, we formulate the chain wherein nodes represent the difference in flow between consecutive test points (nodes N_1, N_2, etc… in Figure 6-2). The differences in flow estimates have a relatively smaller dynamic range and potentially lead to fine-grained results (reduces blockiness). Preliminary results and performance improvements in using message passing are illustrated in Figure 6-3 and Table 6-1 respectively. The following bullets describe the procedure adopted to update streamflow estimates using message passing.

- Twenty four test points along the river and the 3 USGS stations (2321000, 2321500, 2322500) form the chain network.
- Streamflow estimated using the BN model are employed for computing difference nodes (N_1, N_2, etc…) and used as priors to estimate the conditional densities relating adjacent nodes in the chain ($N_1 \rightarrow N_2 \rightarrow \ldots$), namely $M_{y|x}$.
    - The three node BN model comprising of WAM, true nearest neighbor, and ground water level was used.
- Measured flow at 2321000 and 2322500 for the test period (Feb. 2005 – Dec. 2006) was used to compute values for N_1 and N_n (tail node) and used as evidence.
- In the current formulation, each N_i node has 64 levels (i.e., the $P_{N1}(D)$ has 64 bins). If evidence on a particular day is d, $P(D=d)=1$, that is in the vector $P(D)$, only one entry is 1, the rest of the 63 levels are zero (i.e., $P(D)$ is a delta vector). Accordingly π at the head node and λ at the tail node are delta vectors.
- Message passing is utilized to propagate evidence available at N_1 and N_n. After estimating difference flow at intermediate nodes, validation is performed at 2321500.
- Measured flow at 2321500 is compared to three-model BN-estimated flow and belief-propagated flow.

Solving estimation problems over large data sets in high dimensional spaces leads to computational demands that are often impractical to run on General Purpose Processors. FPGAs were proposed as suitable candidates for fast and efficient execution of computationally intensive algorithms. Key algorithms that would benefit from hardware acceleration were identified in the BN framework. In proposing FPGA-based reconfigurable computing as a suitable technology for hardware acceleration, various challenges faced while designing and developing algorithms

on FPGAs were analyzed. Algorithm decomposition, performance prediction, and design reuse for parallel implementation all need to be performed in an efficient and structured manner for developing successful FPGA designs in a productive manner. To address these challenges, an approach for pattern-based decomposition of algorithms for FPGA design and development was proposed.

Significant performance improvements in terms of speedup were obtained by using FPGA-accelerated implementation of the Parzen window-based PDF estimation algorithm decomposed using primitive RC design patterns. Further improvements can be obtained by scaling the design on FPGA device families that have larger number of dedicated DSP units and a high-bandwidth interconnect. Dual-core architectures were developed and evaluated on a single-device system by exploiting the scalability in the algorithm. Key design parameters were identified for tuning the architecture to suit different platforms as well. Precision effects were investigated and data verification along with error statistics suggested a sufficient fixed-point configuration for the algorithm. The benefit of composite patterns for design reuse was also validated. With the successful design of a scalable and portable architecture for the PDF algorithm, rapid hardware development for the K-means clustering and correlation algorithms was possible due to the similarity of their underlying design patterns. Further, the work also showcased the benefit of quantifying design patterns in efficiently exploiting a performance prediction tool, RAT, to predict an algorithm's amenability to a hardware platform before undertaking a lengthy development process.

The architecture developed in this work was designed with consideration toward future extensions to multi-FPGA systems. Further research is warranted in understanding the potential improvements that can be achieved and bottlenecks that might have to be addressed while

migrating designs to multi-FPGA systems. Directions for future work also include investigating and developing design patterns for solving other problem sets that have underlying similarity in their algorithms. Case-study algorithms for such an analysis can be sampled from popular toolboxes in various fields and develop composite patterns for classes of algorithms within them. For example, Figures 6-4 and 6-5 illustrate patterns for tree-based (e.g., neural networks, many-to-one mapping problems, multiscale algorithms, Bayesian networks) and mesh-based algorithms (e.g., image erosion/dilation, edge detectors, color conversion, spatial filters) parameterized by tree-depth and mesh-size respectively. This would be a critical step in enabling FPGA-based RC for solving a general class of computationally intensive problems. This work has shown that one such general architecture exists for a class of machine-learning algorithms.

In summary, the primary contributions of this work are listed below.

- Proposed a scalable framework that allows efficient fusion of multi-modal (domain-based diversity) datasets.
- Developed a Spatio-temporal Bayesian Network topology for estimating streamflow along a river by fusing spatio-temporal datasets.
- Predicted and validated streamflow estimates with uncertainties as a function of location (space), condition (time) and particular mix of sensors.
- Developed a scalable and portable architecture for rapid estimating of multi-dimensional PDFs on multiple platforms.
- Identified and catalogued primitive design patterns for representing key algorithm decompositions in the proposed BN framework for parallel implementations.
- Quantified design patterns to efficiently exploit and parameterize analytic models in a performance prediction tool.
- Effected design reuse aided by pattern-based algorithm decomposition and achieved speedups in the development of K-means and Correlation.

A few future directions to the current work are provided below.

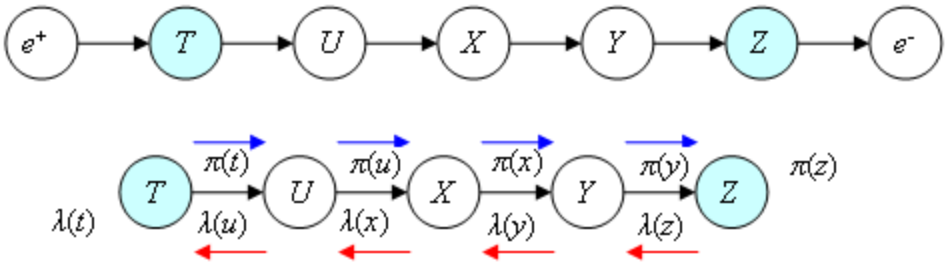
- Integrating MKS and the proposed BN estimator. This could be accomplished by converting MKS to a BN formulation on a multiscale tree (e.g., quadtree).

- Implementing a Dynamic Bayesian Network (DBN) to model streamflow over days (or any time resolution) as a sequence of variables thereby extracting knowledge of past streamflow in estimating current or future streamflow.
- Applying the proposed data fusion framework to non-hydrologic applications such as forestry, bio-medical engineering and other remote sensing applications.
- Exploiting design patterns for generating code templates for FPGA design and development.

Table 6-1. Performance metrics at 2321500 after message passing

Metric	3-node BN (N2)	Message passing
NSC	0.7183	0.7346
Mean abs. error (cu.ft/sec)	174.0000	58.0000
Std.dev of error (cu.ft/sec)	248.0000	288.0000
Entropy of error PDF (nats)	2.1500	2.3400

A



B

Figure 6-1. Illustration of message passing. A) Evidence originating at both ends of chain. B) Dual-parameter message passing.

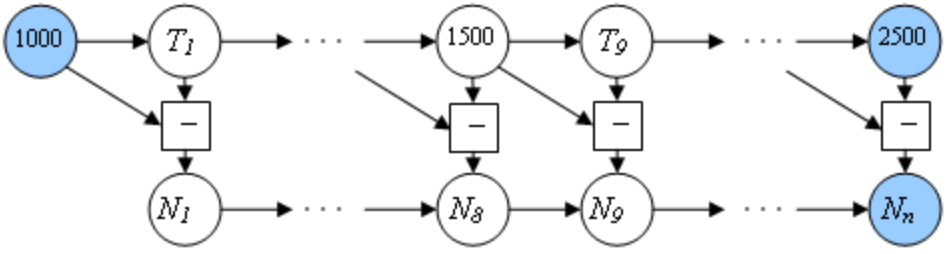


Figure 6-2. Message passing framework for improving streamflow estimates.

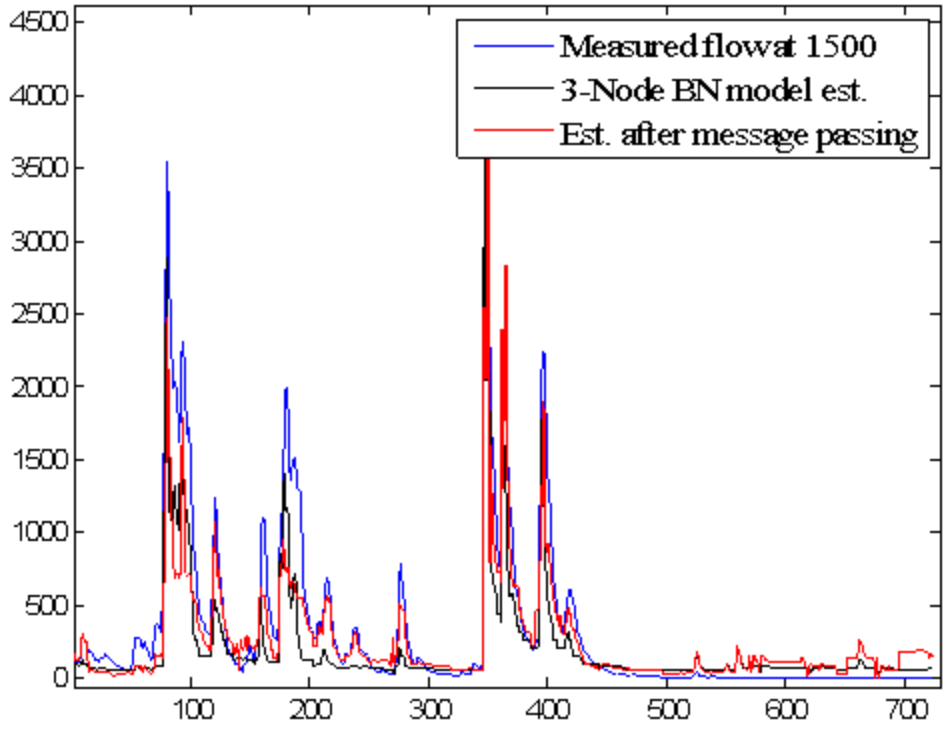


Figure 6-3. Prediction at USGS 2321500 after message passing.

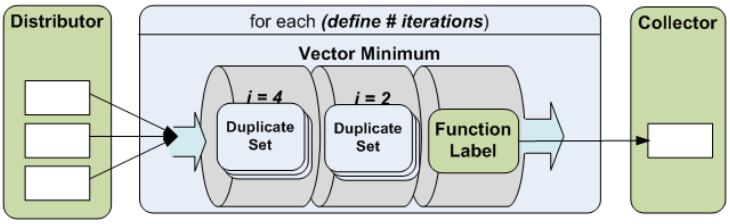


Figure 6-4. Composite pattern for tree-based algorithms.

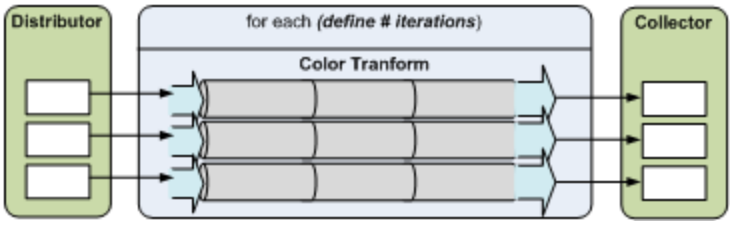


Figure 6-5. Composite pattern for mesh-based algorithms.

LIST OF REFERENCES

1. P. W. Fieguth, W. C. Karl, A. S. Willsky, and C. Wunsch, "Multiresolution Optimal Interpolation and Statistical Analysis of TOPEX/POSEIDON Satellite Altimetry," IEEE Transactions on Geoscience and Remote Sensing, vol. 33(2), 1995, pp. 280–292.

2. M. M. Daniel, and A. S. Willsky, "A Multiresolution Methodology for Signal-level Fusion and Data Assimilation with Applications to Remote Sensing," Proceedings of the IEEE, vol. 85(1), 1997, pp. 164-180.

3. F. Camastra and A. Vinciarelli, "Machine Learning for Audio, Image and Video Analysis – Theory and Applications," Springer-Verlag London Limited, London, UK, 2008.

4. N. Dimitrova, H. Zhang, B. Shahraray, I. Sezan, T. Huang, and A. Zakhor, "Applications of Video-content Analysis and Retrieval," Journal of Multimedia, vol. 9(3), 2002, pp. 42-55.

5. F. Lillo, S. Basile, and R. N. Mantegna, "Comparative Genomics Study of Inverted Repeats in Bacteria," Bioinformatics, vol. 18(7), 2002, pp. 971-979.

6. F. Pitie, A.C. Kokaram, and R. Dahyot, "N-Dimensional Probability Density Function Transfer and its Application to Color Transfer," Proceedings of the 10th International Conference on Computer Vision, Beijing, China, 17th October 2005 – 21st October 2005, pp. 1434 – 1439.

7. S. M. Kay, A. H. Nuttall, and P. M. Baggenstoss, "Multidimensional Probability Density Function Approximations for Detection, Classification, and Model Order Selection," IEEE Transactions on Signal Processing, vol. 49(10), 2003, pp. 2240 – 2252.

8. Y. Chang, W. Zeng, I. Kamel, and R. Alonso, "Integrated Image and Speech Analysis for Content-Based Video Indexing," Proceedings of 3rd International Conference on Multimedia Computing and Systems, Hiroshima, Japan, 17th June1996 – 23rd June 1996, pp. 306-313.

9. H. Zhang, Y. Zhuang, and F. Wu, "Cross-modal Correlation Learning for Clustering on Image-Audio Dataset," Proceedings of the 15th International Conference on Multimedia, Augsburg, Germany, 2007, pp. 273-276.

10. R. Goecke, J. B. Millar, A. Zelinsky, and J. Robert-Ribes, "Analysis of Audio-Video Correlation in Vowels in Australian English," International Conference on Audio-Visual Speech Processing, Aalborg, Denmark, 2001, pp. 115-120.

11. L. Greengard, and J. Strain, "The Fast Gauss Transform", SIAM Journal on Scientific and Statistical Computing, vol. 12(1), 1991, pp. 79–94.

12. D.E. Culler and J.P. Singh, "Parallel Computer Architecture: A Hardware/Software Approach," Morgan Kaufmann Publishers Incorporation, San Francisco, California, 1999.

13. WATERS NETWORK, WATERS Test Bed Sites, 2005, http://www.watersnet.org/wtbs/index.html, 22nd April 2009.

14. D. E. Holmes, L. C. Jain, "Innovations in Bayesian Networks: Theory and Applications", Springer, New York, New York, 2009.

15. J. Zhang, and B. Whalen, "Estimated Phosphorus Load Reductions Under Various Water Management Alternatives," American Society of Agricultural and Biological Engineers Annual International Meeting, St.Joseph, Michigan, July 2005.

16. A. B. Bottcher, B. M. Jacobson, and J. G. Hiscock, "Characterizing Flow and Nutrient Loads for TMDL Development in Florida using WAM," Proceedings of the Conference on Total Maximum Daily Load (TMDL) Environmental Regulations-II, Albuquerque, New Mexico, 8th November 2003 – 12th November 2003, pp. 27-34.

17. M. Yadav, T. Wagener, and H. Gupta, "Regionalization of Constraints on Expected Watershed Response Behavior for Improved Predictions in Ungauged Basins," Advances in Water Resources, vol. 30(8), 2007, pp. 1756–1774.

18. D. P. Boyle, H. V. Gupta, and S. Sorooshian, "Toward Improved Calibration of Hydrologic Models: Combining the Strengths of Manual and Automatic Methods," Water Resources Research, vol. 36(12), 2000, pp. 3663–3674.

19. M. E. Borsuk, D. Higdon, C. A. Stow, and K. H. Rechhow, "A Bayesian Hierarchical Model to Predict Benthic Oxygen Demand from Organic Matter Loading in Estuaries and Coastal Zones," Ecological Modeling, vol. 143(3), 2001, pp. 165–181.

20. K. C. Slatton, S. Cheung, and H. Jhee, "Reduced-Complexity Fusion of Multiscale Topography and Bathymetry Data over the Florida Coast," IEEE Geoscience and Remote Sensing Letters, vol. 2(4), October 2005, pp. 389–393.

21. L. Alparone, S. Baronti, A. Garzelli, and F. Nencini, "A Global Quality Measurement of Pan-sharpened Multispectral Imagery," IEEE Geoscience and Remote Sensing Letters, vol. 1(4), October 2004, pp. 313–317.

22. V. Tsagaris, V. Anastassopoulos, and G. A. Lampropoulos, "Fusion of Hyperspectral Data Using Segmented PCT for Color Representation and Classification," IEEE Transactions on Geoscience and Remote Sensing, vol. 43(10), October 2005, pp. 2365–2375.

23. W. T. Crow, "Correcting Land Surface Model Predictions for the Impact of Temporally Sparse Rainfall Rate Measurements Using an Ensemble Kalman Filter and Surface Brightness Temperature Observations," Journal of Hydrometeorology, vol. 4(5), April 2003, pp. 960–973.

24. J. W. Jones, W. D. Graham, D. Wallach, M. Bostick, and J. Koo, "Estimating Soil Carbon Levels using an Ensemble Kalman Filter," Transactions of the American Society of Agricultural Engineers, vol. 47(1), 2004, pp 331–339.

25. L. Bruzzone, D. F. Prieto, and S. B. Serpico, "A Neural-Statistical Approach to Multitemporal and Multisource Remote-Sensing Image Classification," IEEE Transactions on Geoscience and Remote Sensing, vol. 37(3), May 1999, pp. 1350–1359.

26. M. Datcu, F. Melgani, A. Piardi, and S. B. Serpico, "Multisource Data Classification with Dependence Trees," IEEE Transactions on Geoscience and Remote Sensing, vol. 40(3), March 2002, pp. 609–617.

27. C. K. Wikle, R. F. Milliff, D. Nychka, and L. M. Berliner, "Spatiotemporal Hierarchical Bayesian Modeling: Tropical Ocean Surface Winds," Journal of the American Statistical Association, vol. 96(454), June 2001, pp. 382–397.

28. M. Koeppen, "The Curse of Dimensionality," 5th Online World Conference on Soft Computing in Industrial Applications (WSC5), held on the internet, 4th September 2000 – 18th September 2000.

29. C. Krekeler, K. C. Slatton, and M. Cohen, "Multi-variate Bayesian Classification of Soil Drainage Using Feature-level Fusion of Topographic and Hydrologic Data," Proceedings of IEEE International Geoscience and Remote Sensing Symposium, Denver, Colorado, 31st July 2006 – 4th August 2006, pp. 2522–2525.

30. N. Mount, and T. Scott, "A Discrete Bayesian Network to Invesigate Suspended Sediment Concentrations in an Alpine Proglacial Zone," Journal of Hydrological Processes, vol. 22(18), February 2008, pp. 3772–3784.

31. M. Molina, R. Fuentetaja, and L. Garrote, "Symbolic and Quantitative Approaches to Reasoning with Uncertainty", SpringerLink, Heidelberg, Berlin, 2005, pp. 88–99.

32. S. Lanini, "Water Management Impact Assessment Using a Bayesian Network Model", International Conference on Hydroinformatics, Nice, France, 4th September 2006 – 8th September 2006, pp. 105–112.

33. W. A. Pike, "Modeling Drinking Water Quality Violations with Bayesian Networks," Journal of the American Water Resources Association, vol. 40(6), 2004, pp. 1563–1578.

34. M. Park, and M. K. Stenstrom, "Using Satellite Imagery for Stormwater Pollution Management with Bayesian Networks," Waters Research, vol. 40(18), 2006, pp. 3429–3438.

35. B. G. Katz, R. S. DeHan, J. J. Hirten, and J.S Catches, "Interactions Between Ground Water and Surface Water in the Suwannee River Basin, Florida", Journal of the American Water Resources Association, vol. 33(6), 1997, pp. 1237–1254.

36. C. A. Crandall, B. G. Katz, and J. J. Hirten, "Hydrochemical Evidence for Mixing of River Water and Groundwater During High-flow Conditions, Lower Suwannee River Basin, Florida, USA," Hydrogeology Journal, vol. 7(5), October 1999, pp. 454–467.

37. US Geological Society, National Elevation Dataset, 2008, http://seamless.usgs.gov/, 22nd April 2009.

38. National Engineering Handbook, Section 4 - Hydrology, Washington, D.C. , USDA-SCS, 1985.

39. W. Viessman, G. L. Lewis, and J. W. Knapp, "Introduction to Hydrology," Harper-Collins, 3rd edition New York , NY, 1989.

40. R. H. McCuen, "Hydrologic Analysis and Design," Prentice Hall, 3rd edition, Upper Saddle River, NJ, 2005.

41. Santa Fe River Watershed Test bed, a NSF WATERS Project, 2007, http://suwanneeho.ifas.ufl.edu/SantaFe.htm, 22nd April 2009.

42. M. Llyas, "General Probability Density Function of Packet Service Times for Computer Networks," Electronics Letters, vol. 23(1), 1987, pp. 31–32.

43. R. R. Bliss and N. Panigirtzoglou, "Testing the Stability of Implied Probability Density Functions," Journal of Banking and Finance, vol. 26(2), 2002, pp. 381–422.

44. H. Jiang, T. Lin, and H. Zhang, "Video Segmentation with the Assistance of Audio Content Analysis," International Conference on Multimedia and Expo, New York, NY, 2000, pp. 1507-1510.

45. K.S. Hemmert and K.D. Underwood, "An Analysis of the Double-precision Floating-point FFT on FPGAs," IEEE Symposium on Field-Programmable Custom Computing Machines, Washinton, DC, April 2005, pp. 171–180.

46. G. Govindu, S. Choi, V. Prasanna, V. Daga, S. Gangadharpalli, and V. Sridhar, "A High-performance and Energy-efficient Architecture for Floating-point-Based LU Decomposition on FPGAs," IEEE Symposium on Parallel and Distributed Processing, Santa Fe, New Mexico, April 2004, pp. 149

47. J. S. Kim, P. Mangalagiri, K. Irick, N. Vijaykrishnan, M. Kandemir, L. Deng, K. Sobti, C. Chakrabarti, N. Pitsianis, and X. Sun, "TANOR: A Tool for Accelerating N-body Simulations on Reconfigurable Platform," Proceedings of the 17th International Conference on Field Programmable Logic and Applications, Amsterdam, August 2007, pp. 68 – 73.

48. M. Leeser, J. Theiler, M. Estlick, and J. J. Szymanski, "Design Tradeoffs in a Hardware Implementation of the K-means Clustering Algorithm," Proceedings of Sensor Array and Multichannel Signal Processing Workshop, Cambridge, Massachusetts, 16th March 2000 – 17th March 2000, pp.520-524.

49. I. Frohlich, A. Gabriel, D. Kirschner, J. Lehert, E. Lins, M. Petri, T. Perez-Cavalcanti, J. Ritman, D. Schafer, A. Toia, M. Traxler, and W.Kuehn, "Pattern Recognition in the HADES - Spectrometer: An Application of FPGA Technology in Nuclear and Particle Physics," Proceedings of the International Conference on Field-Programmable Technology (FPT), Singapore, December 2002 , pp. 443 – 444.

50. S. Neo, H. Goh, W. Y. Ng, J. Ong, and W. Pang, "Real-time Online Multimedia Content Processing: Mobile Video Optical Character Recognition and Speech Synthesizer for the Visual Impaired," Proceedings of the International Convention on Rehabilitation Engineering and Assistive Technology, Singapore, 2007, pp. 201-206.

51. H. Schmit and D. Thomas, "Hidden Markov Modeling and Fuzzy Controllers in FPGAs," Proceedings of Symposium on FPGAs for Custom Computing Machines, Napa, CA, April 1995, pp. 214 – 221.

52. T. VanCourt and M. Herbordt, "Three Dimensional Template Correlation: Object Recognition in 3D Voxel Data," Proceedings of Computer Architecture for Machine Perception, Washinton, DC, 2005, pp. 153-158.

53. B. Holland, K. Nagarajan, C. Conger, A. Jacobs, and A. D. George, "RAT: A Methodology for Predicting Performance in Application Design Migration to FPGAs," Proceedings of High-Performance Reconfigurable Computing Technologies and Applications Workshop, Reno, Nevada, 11th November 2007.

54. C. P. Steffen, "Parametrization of Algorithms and FPGA Accelerators to Predict Performance," Proceedings of Reconfigurable System Summer Institute (RSSI), Urbana, Illinois, 2007, pp. 17-20.

55. A. DeHon, J. Adams, M. DeLorimier, N. Kapre, Y. Matsuda, H. Naeimi, M. Vanier, and M. Wrighton, "Design Patterns for Reconfigurable Computing", IEEE Symposium on Field Programmable Custom Computing Machines, Napa Valley, California, 2004, pp. 13-23.

56. J. W. Fisher and J. C. Principe, "A Methodology for Information Theoretic Feature Extraction," Neural Networks Proceedings on IEEE World Congress on Computational Intelligence, vol. 3(1), 4th May 1998 – 9th May 1998, pp. 1712–1716.

57. J. Pearl, "Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference," Morgan Kaufmann, San Francisco, CA, 1988.

58. F. Jensen, "Bayesian Networks and Decision Graphs," Springer-Verlag, New York, New York, 2001.

59. R. O. Duda, P. E. Hart, and D. G. Stork, "Pattern Classification," John Wiley & Sons Inc., 2nd edition, New York, New York, 2001.

60. D. Erdogmus, Y. N. Rao, and J. C. Principe, "Nonlinear Independent Component Analysis by Homomorphic Transformation of the Mixtures," Proceedings of the IEEE International Joint Conference on Neural Networks, vol. 1, 25th July 2004 – 29th July 2004, pp. 47–52.

61. D. R. Legates and G. J. McCabe, "Evaluating the Use of "Goodness-of-fit" Measures in Hydrologic and Hydroclimatic Model Validation," Water Resources Research, vol.35(1), 1999, pp. 233–242.

62. E. Gamma, R. Johnson, H. Helm, J. M. Vlissides, and G. Booch, "Design Patterns: Elements of Reusable Object-Oriented Software", Addison-Wesley Professional, 1994, pp. 416.

63. J. Anvik, S. MacDonald, D. Szafron, J. Schaeffer, S. Bromling and K. Tan, "Generating Parallel Programs from the Wavefront Design Pattern", Proceedings of the International Workshop on High-Level Parallel Programming Models and Supportive Environments, Fort Lauderdale, FL, 2002, pp. 104-111.

64. K. T. Gribbon, D. G. Bailey, and C. T. Johnston, "Design Patterns for Image Processing Algorithm Development on FPGAs," IEEE TENCON, 21st November 2005 – 24th November 2005, pp. 1-6.

65. M. Y. Mashor, "Improving the Performance of K-means Clustering Algorithm to Position the Centres of RBF Networks," International Journal of the Computer, the Internet and Management, vol. 6(2), 1998, pp. 1–19.

66. D. L. Turcotte, "Fractals and Chaos in Geology and Geophysics," Cambridge University Press, 2nd edition, New York, New York, 1997.

67. H. O. Peitgen and D. Saupe, "The Science of Fractal Images," Springer, New York, New York, 1988.

68. Z. Li, Q. Zhu and C. Gold, "Digital Terrain Modeling: Principles and Methodology," CRC Publishers, Boca Raton, Florida, 2004.

69. L. M. Kaplan, "Extended Fractal Analysis for Texture Classification and Segmentation," IEEE Transactions On Image Processing, vol. 8(11), November 1999, pp. 1572 – 1585.

70. A. P. Pentland, "Fractal-based Description of Natural Scenes," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 6, November 1984, pp. 661–674.

71. V. Aggarwal, K. Nagarajan, and K. C. Slatton, "Multiple-Model Multiscale Data Fusion Regulated by a Mixture-of-Experts Network," Proceedings of IEEE International Geoscience and Remote Sensing Symposium, vol. 1, September 2004, pp. 364 – 367.

72. W. S. Chaer, R. H. Bishop, and J. Ghosh, "A Mixture of Experts Framework for Adaptive Kalman Filtering," IEEE Transactions on Systems, Man and Cybernetics, vol. 27(3), June 1997, pp. 452–464.

73. C. V. Stewart, B. Moghaddam, K. J. Hintz, and L. M. Novak, "Fractional Brownian Motion Models for Synthetic Aperture Radar Imagery Scene Segmentation," Proceedings of IEEE, vol. 81, October 1993, pp. 1511–1522.

74. J. M. Keller, S. Chen, and R. M. Crownover, "Texture Description and Segmentation Through Fractal Geometry," Computer Vision, Graphics, and Image Processing, vol. 45(2), 1989, pp. 150–166.

75. T. Kumar, P. Zhou, and D. A. Glaser, "Comparison of Human Performance with Algorithms for Estimating Fractal Dimension of Fractional Brownian Statistics," Journal of the Optical Society of America, vol. 10(6), June 1993, pp. 1136–1146.

76. L. M. Kaplan and C. J. Kuo, "Texture Roughness Analysis and Synthesis via Extended Self-similar (ESS) Model," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 17(11), November 1995, pp. 1043–1056.

77. Y. Deng, "New Trends in Digital Terrain Analysis: Landform Definition, Representation, and Classification," Progress in Physical Geography, vol. 31(4), 2007, pp. 405-419.

78. K. C. Clarke, "Computation of the Fractal Dimension of Topographic Surfaces Using the Triangular Prism Surface Area Method," Computers and Geoscience, vol. 12(5), May 1986, pp. 713 – 722.

79. M.Schroeder, "Fractals,Chaos,Power Laws," W. H. Freeman, Gordonsville, Virginia, 8th print, 1992.

80. J. Pearl, "Probabilistic reasoning in intelligent systems: Network of plausible inference," Morgan Kaufman, San Mateo, California, 1988.

BIOGRAPHICAL SKETCH

Karthik Nagarajan received his Ph.D. and M.S. degrees in electrical engineering from the University of Florida in 2005 and 2009 respectively while simultaneously working for the Adaptive Signal Processing Laboratory (ASPL) and the NSF Center for High-performance Reconfigurable Computing (CHREC). He received the B.E. degree in electronics and communication engineering from the University of Madras in 2003. His research interests include pattern recognition, information theory, graphical models, and multiscale estimation concepts. He is also an active researcher in the field of high performance and reconfigurable computing with focus in FPGA design and development, performance analysis, and productivity studies.