Hindawi Publishing Corporation
International Journal of Reconfigurable Computing
Volume 2010, Article ID 492560, 15 pages
doi:10.1155/2010/492560

Research Article

RapidRadio: A Domain-Specific Productivity Enhancing Framework

Jorge A. Surís,[1] Adolfo Recio,[2] and Peter Athanas[2]

[1] *Oak Ridge National Laboratory, Oak Ridge, TN 37831, USA*
[2] *Bradley Department of Electrical and Computer Engineering, Virginia Polytechnic and State University, Blacksburg, VA 24067, USA*

Correspondence should be addressed to Jorge A. Surís, jasuris@gmail.com

Received 7 March 2010; Revised 5 July 2010; Accepted 30 November 2010

Academic Editor: Viktor K. Prasanna

The RapidRadio framework for signal classification and receiver deployment is discussed. The framework is a productivity enhancing tool that reduces the required knowledge-base for implementing a receiver on an FPGA-based SDR platform. The ultimate objective of this framework is to identify unknown signals and to build FPGA-based receivers capable of receiving them. The architecture of the receiver deployed by the framework and its implementation are discussed. The framework's capacity to classify a signal and deploy a functional receiver is validated with over-the-air experiments.

1. Introduction

With the ever increasing processing power of general purpose processors and FPGAs, radio platforms are no longer tied to using static hardware. Using more flexible hardware platforms allows the radio to adapt to its environment, modifying the modulation scheme, symbol rate, or other link properties to maximize the efficiency of the communications link. This added flexibility then results in a volatile communications environment in which a participant may not know in advanced the properties of the link. This is often the case in signal intelligence applications where the user is attempting to become a silent third-party participant in a private communication. In this case, the user must first determine the link properties (modulation type, carrier frequency, symbol rate, etc.) and then create a receiver with the desired properties. Given the complexity of both of these tasks, it is desirable to automate the process.

Determining the properties of the physical communications link is a difficult problem and has been the focus of a large number of research efforts [1]. Although great strides have been made in this area, large assumptions about the environment or the signal are made that limit the applicability of the proposed techniques. Additionally implementation of a receiver for the newly classified signal is rarely addressed. When addressed, mostly software-based solutions are offered. The flexibility provided by FPGAs makes them good candidates for the implementation of the receivers, but the complexity associated with the design and development of FPGA-based solutions has relegated them to be second-class citizens. Instead of exploiting the full potential of the FPGA, radio designers implement only what is absolutely necessary on the FPGA. This strategy has worked so far because signal processing is performed on high-end processors on a PC. For embedded systems, however, the power requirements of these processors are unacceptable.

In this paper, the RapidRadio framework for signal classification and receiver deployment is discussed. RapidRadio is a domain-specific productivity enhancement tool that aims to reduce the knowledge base required for the development of radio receivers. Many productivity enhancing tools provide a design environment where the user can specify a series of high-level blocks that compose the system [2, 3]. Although these tools abstract away part of the design process, the user must still understand the block's interfaces and how they are interconnected. By narrowing the scope of the tool suite to a specific task, such as receiver deployment, RapidRadio can abstract most of the system architecture away. The user

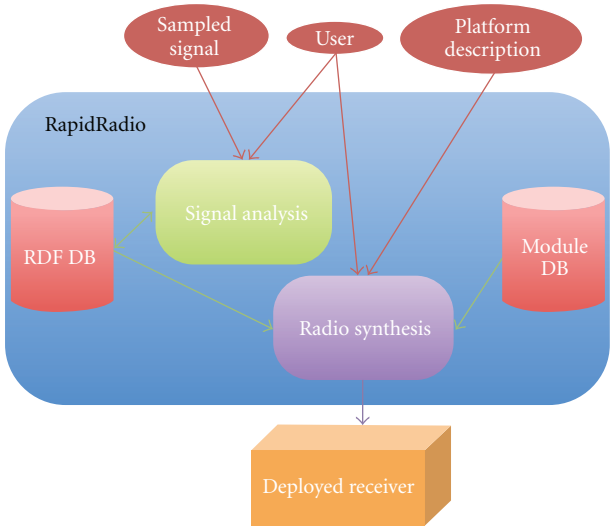


Figure 1: RapidRadio system architecture.

need only be concerned with signal analysis activities, and the framework will automatically identify the parameters of the physical link layer and build a receiver for it.

2. Rapid Radio Framework

RapidRadio divides the process of radio creation into two phases: the analysis phase and radio synthesis phase. The analysis phase guides the user through the process of classifying an unknown signal and determining its modulation scheme and parameters, cast in the domain of a signal intelligence analyst. Various transforms are provided to the user and the results of these transforms are presented. The term "transform" here is used in the abstract sense and refers to the process of applying various operations on continuous signal sets to accentuate certain features. Using high-level transforms shield the user from the underlying operations. The user, for example, may need to determine if a signal is synchronized or not but does not need to know how the synchronization is being performed. In addition to presenting the results of the transforms to the user, an expert system analyzes the results and suggests a possible course of action to the user. The output of the analysis phase is an abstract representation of the architecture of the receiver called the Radio Description File (RDF). The receiver synthesis phase transforms the platform independent RDF into a platform-specific description of the receiver. Figure 1 shows the high-level architecture of the RapidRadio framework and how these two phases interact.

Signal detection and classification is an inherently difficult problem. Before any classification is attempted, the signal desired must be identified. Most classification systems depend on some artificial intelligence structures [4, 5] to make decisions. The problem with this approach is that the artificial intelligence structures need to be trained a priori. This makes it hard to expand the system to classify new signal types. The RapidRadio framework uses the "human in the loop" approach to address this problem. "human in the loop" implies that the user is prompted to validate the decisions made by the framework. All the data used to make a decision is presented to the user. If the user disagrees with the decision made by the framework, then the framework's decision can be overridden. The RapidRadio framework utilizes the "plug-in" concept to coarsely describe various modulation and synchronization strategies that it can use in its search process. For example, constellations tested are specified in the form of an XML file. Adding a new constellation for consideration is easily accomplished by creating an XML file for the constellation. The interaction between the user and the framework is shown in Figure 1. The signal analysis part of the framework is discussed in Section 3 followed by a discussion of the receiver deployment phase in Section 4. Results from simulations and over-the-air experiments are presented in Section 6.

3. Signal Analysis

Signal analysis is the process through which the modulation scheme used for a signal transmission is identified. A large body of work has concentrated on Automatic Modulation Classification (AMC), but typically many assumptions are made about the signal that simplify the classification process [6, 7]. In many cases, the system has a priori knowledge about the signal being classified [8, 9]. In such cases, making assumptions such as perfect synchronization is reasonable. When working with the emergency bands, for example, it may be acceptable to assume that the receiver has a priori knowledge of the signals because the number of users is limited to the police, fire-fighters, and rescue personnel.

The RapidRadio framework is designed to perform blind signal classification of single-carrier, linearly modulated signals. As mentioned above, many contemporary signal classification systems assume perfect synchronization, which is difficult to obtain if the modulation scheme is unknown. RapidRadio follows a different path to signal classification that allows it to avoid such assumptions. Figure 2 shows a high-level description of the signal analysis process. First, the spectrum is sampled and displayed. The user then selects the area of interest in the spectrum. The selected signal is brought down to baseband and a spectral fitting is done to obtain the basic modulation parameters. For each hypothesized constellation, synchronization is attempted. Lastly, a set of metrics are obtained to determine the correctness of each hypothesis based on the synchronized signal. The user can evaluate the metrics and determine which hypothesis is correct. A Matlab front-end, with a CLIPS [10] back-end, is used to allow the user to interface with the RapidRadio framework. The steps involved in the analysis phase are discussed in more detail in subsequent sections.

3.1. Signal Selection and Isolation Interface. Upon initialization of the framework, the user is presented with a graphical user interface that allows him or her to obtain a sample of the spectrum, isolate a small portion of the spectrum, and

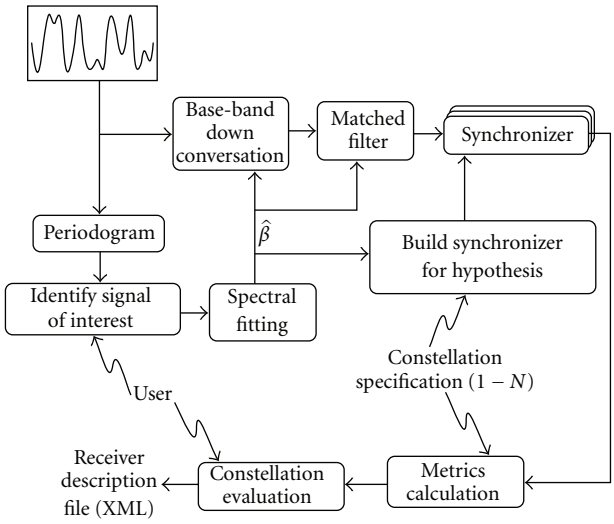


Figure 2: High-level flow of the analysis phase.

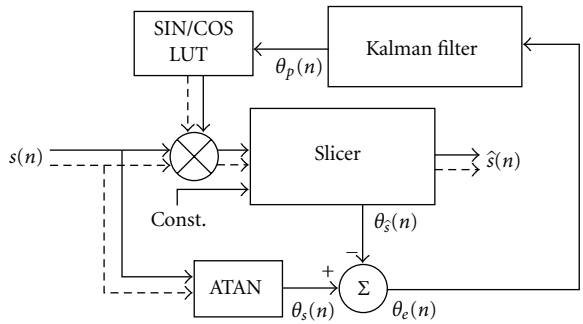


Figure 3: RapidRadio derotator architecture.

estimate the modulation parameters. Additionally, the CLIPS engine is initialized. CLIPS is a rule-based expert systems that makes inferences based on a set on known "facts" and rules. Facts represent what the expert system knows or has observed. The rules are used to interpret to indicate how facts should be interpreted. Initializing the framework eliminates transient facts developed during a previous run from affecting future runs. A set of default facts stored in CLIPS, such as the location of the constellation description files, are used to initialize the framework. The framework then loads the constellations into memory and adds them to the CLIPS fact set.

The operating environment is then added to the CLIPS fact-set. Using predefined rules, CLIPS defines the probability of each constellation based on the in-use environment. The RapidRadio framework includes three predefined operating environments.

(i) *Unknown*: there is no knowledge about the environment. All constellations are assumed to be equiprobable.

(ii) *Urban*: a high-noise environment is assumed. Constellations with less than four bits per symbol are assumed to have a higher probability than those with four or more bits.

(iii) *Microwave*: a low-noise environment is assumed. Constellations with four or more bits per symbol are assumed to have a higher probability than those with three or less bits.

These predefined environments were defined for testing and validation purposes only and may not accurately represent reality. All results shown in this paper were obtained using the Unknown environment. Environments can be easily added or altered by modifying the CLIPS rule set. This permits the user to taylor the environment to his or her operational conditions by reflecting a priori probabilities of certain constellations. New rules are automatically used, because the rule-sets are always reloaded during initialization.

3.2. Spectrum Sampling. Obtaining a sample of the spectrum is also done at the end of system initialization. The platform configuration used to sample the spectrum and the sampling frequency are obtained from the CLIPS fact-set. The Matlab front-end establishes a TCP/IP link with a daemon running on the target embedded platform. Through this link, the front-end configures the radio platform and extracts a sample of the signal. The signal is then converted to the frequency domain to allow examination of the spectrum.

The framework's GUI presents the periodogram, using the Bartlett method [11], of the captured signal. The user can then select a portion of the spectrum to analyze, by zooming into the area of the periodogram containing the signal. The area selected is used to obtain a crude estimate of the carrier frequency ($\hat{f}_c$).

3.3. Modulation Parameter Estimation. Throughout the analysis phase, only two assumptions about the signal are made. The first is that the signal is linearly modulated. Although this is a limiting factor, it is not an unreasonable assumption because linear modulations have wide-spread use. The second assumption made is that a root of raised cosine filter is used at the transmitter to limit bandwidth. This is a very common filter because it limits the bandwidth of the signal and reduces intersymbol interference. Because the ideal shape of the signal is known, an estimate of the carrier frequency ($\hat{f}_c$), symbol rate ($\hat{f}_s$), the roll-off factor ($\hat{\alpha}$), and signal-to-noise ratio (SNR) can be obtained by fitting the spectral shape of the incoming signal to the model. Finding the best fit is then a nonlinear Least Square optimization problem. This process is discussed in detail in [12]. The initial estimates for the unknowns are obtained from the user's specification of the signal to analyze.

3.4. Symbol Timing Recovery. Symbol timing recovery circuits attempt to determine the optimal moment in which to sample the incoming signal to extract the transmitted symbol. Many symbol timing recovery circuits, with varying levels of flexibility, have been proposed in the literature.

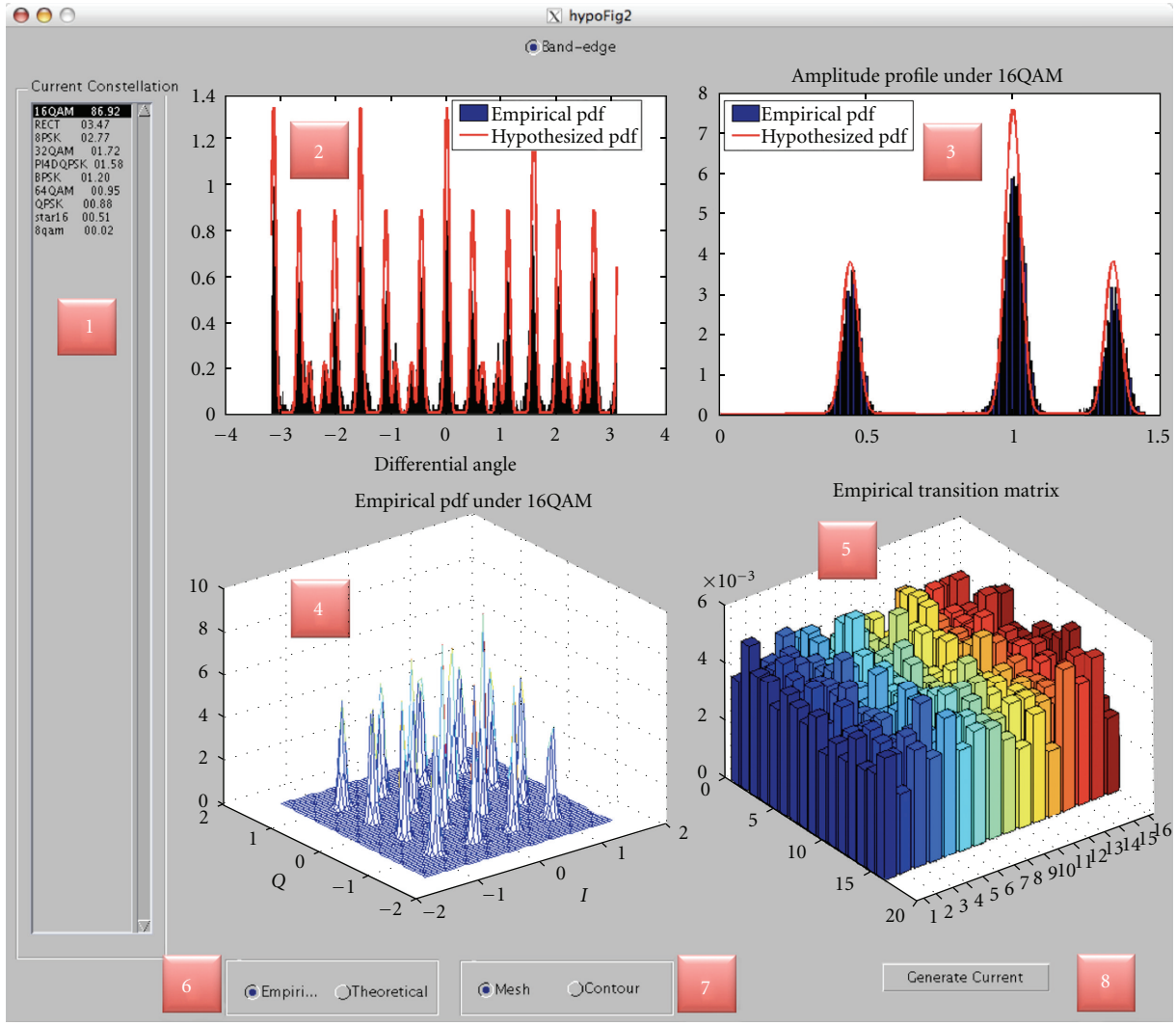


Figure 4: RapidRadio hypothesis evaluation interface showing the deployment of an inferred 8QAM receiver.

In [13], an early/late synchronization algorithm is presented, capable of operating a various symbol rates, but a training sequence is required. An efficient phase-independent synchronizer is presented in [14], but only phase modulated signals are supported.

To provide maximum flexibility, the chosen timing recovery architecture should work with little to no change for all linear modulation schemes. The architecture must also be rotation agnostic because carrier recovery is done after symbol synchronization. For RapidRadio, a modified passband timing recovery synchronizer with an oversampling rate of four is used [15, 16]. This architecture was chosen because it extracts the timing information from a spectral component in the signal, making it independent of the actual modulation scheme being used. A spectral line generator is used to extract the symbol timing from the received signal. The spectral generator produces a sinusoidal signal with a frequency equal to the actual symbol frequency. A local, smoother copy of the signal is produced using a phased-locked loop (PLL). Using the accumulator of the PLL, the correct timing instant is calculated. A Lagrange interpolator is used to obtain the value of the signal at the correct instant.

3.5. Carrier Recovery. The output of the timing recovery circuit is a stream of symbols. Each symbol is represented by the value of two signals; the in-phase (I) and the quadrature (Q) signals. They are represented in a two-dimensional plane where the x axis represents the magnitude of the in-phase signal and the y axis represents the magnitude of the quadrature signal. In the presence of carrier frequency error, symbols rotate on the I/Q plane and need to be derotated. An efficient derotation architecture is presented for 16QAM in [17, 18]. The RapidRadio architecture expands on this work to increase the derotator's flexibility and tolerance to frequency errors. The derotator architecture can be seen in Figure 3.

The incoming symbol ($s(n)$) is derotated using the predicted phase error ($\theta_p(n)$). A constellation-specific slicer is used to determine the closest constellation symbol ($\hat{s}(n)$). The slicer also produces the phase of $\hat{s}(n)$ ($\theta_{\hat{s}}(n)$). The

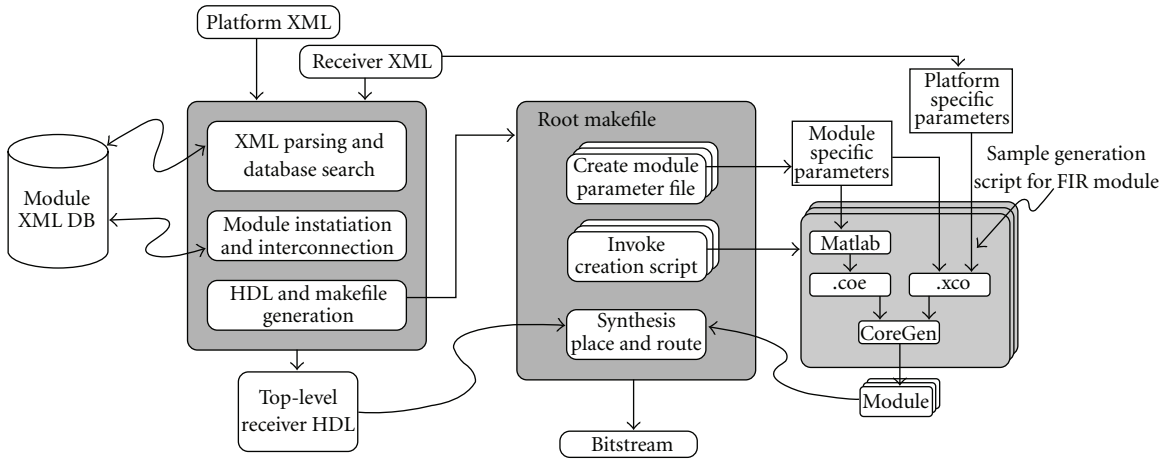


Figure 5: Radio synthesis flow.

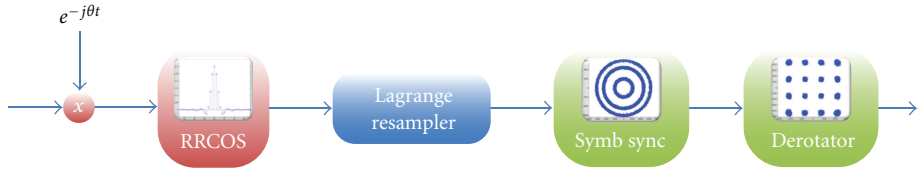


Figure 6: RapidRadio receiver architecture.

measured phase error ($\theta_e(n)$) is the difference between the phase of the $s(n)$ ($\theta_s(n)$) and $\theta_{\hat{s}}(n)$. The traditional PLL is replaced with a Kalman filter that tracks both the phase error and the frequency (f_d).

3.6. Hypothesis Fitness Evaluation. For each hypothesized constellation, the output of both the symbol synchronizer and the derotator is evaluated to determine the correctness of the demodulation. The evaluation is based on four metrics: the phase profile (pp) which measures the change in phase between two consecutive symbols, the amplitude profile (ap) which measures the magnitude of the received symbols, the symbol distribution (sd) which measures how the symbols are distributed in the I/Q plane, and the final metric is the symbol transition matrix (tm) which measures how often a transition between two specific symbols is observed.

All four metrics are calculated by comparing the theoretical probability distribution function (PDF) of the data with the empirical PDF. The empirical PDF is obtained by grouping the received data points in a histogram. Bin sizes for the histogram are calculated using Scott's formula for optimal bin size [19]. The count for each bin is then divided by the total number of data points obtained. The theoretical PDFs are obtained from models adjusted for the level of noise observed. The two PDFs are compared using the Hellinger distance (H) [20, 21]. The Hellinger distance is a measure of how similar two PDFs are. It takes values in the range of $(0, 1)$, where a zero indicates the PDFs are identical and a value of one indicates they are completely different.

The hypothesis evaluation GUI, shown in Figure 4, presents both the theoretical and empirical PDFs for the metrics in graphical form for evaluation by the user. The panel on the left marked with a one displays all the hypothesized constellations along with their scores. The hypotheses are ordered based on their likelihood, with the most likely constellation on the top. By selecting a constellation in this panel, the user can view the data used by the framework to formulate its decision. Navigating through the different hypotheses allows the user to make an informed decision on whether the framework chose the correct constellation or not. Areas marked two through five show the data used for the phase profile, the amplitude profile, the symbol distribution and the transition matrix, respectively. Both the theoretical and the empirical values are shown. Some datasets need to be presented in 3 dimensions making it impossible to present the theoretical and empirical values in the same chart, so buttons are provided to allow the user to switch between the two (Areas six and seven). Lastly a button to indicate to the framework to generate the radio for the provided constellation is shown in area eight.

3.7. Phase Profile. The phase profile examines the change in phase (θ) between consecutive received symbols. The theoretical PDF is obtained using the valid transitions for the hypothesized constellation. This does not account for errors due to signal noise however. Given that an estimate of the noise is obtained by the parameter estimator (see Section 3.3), the theoretical PDF can then be adjusted to reflect the phase variance due to the variance in the symbols.

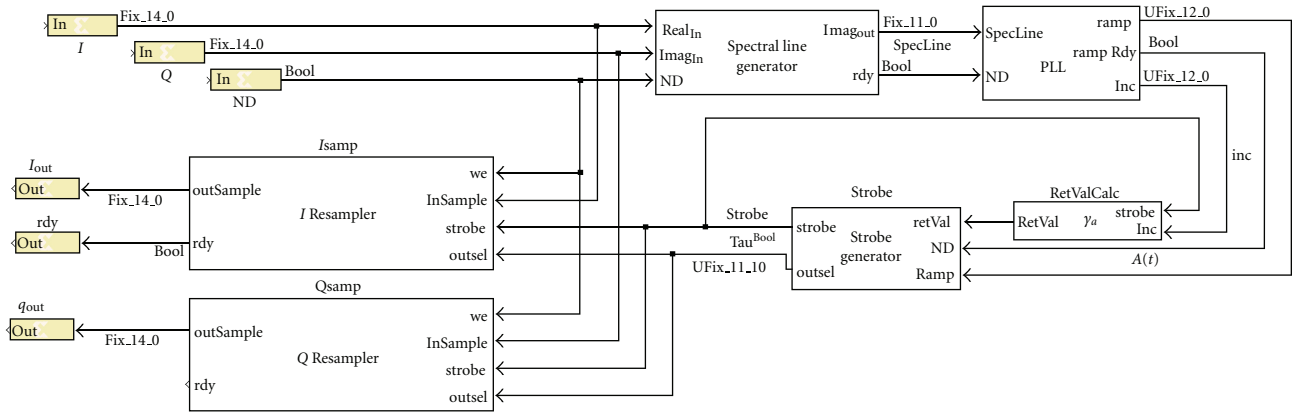


Figure 7: Symbol synchronizer implementation.

The PDF of the phase difference between two vectors perturbed by Gaussian noise is given as [22]

$$f(\psi) = \frac{1}{2\pi}\int_0^{\pi/2} e^{-\rho(1-\cos(\psi)\cos(\theta))}\left(1+\rho+\rho\cos(\psi)\cos(\theta)\right)\cos(\theta)\mathrm{d}\theta, \tag{1}$$

where $\rho = 1/\sigma^2$, σ^2 is the variance of the Gaussian noise, and $\psi = \theta(t) - \theta(t-1)$ with zero mean. Let ζ equal the expected phase difference for a symbol transition and ψ_n the phase difference with mean n; then

$$f(\psi_\zeta) = f(\psi_0 - \zeta). \tag{2}$$

The theoretical PDF is then

$$f_{\mathrm{pp}|c}(x) = \sum_{i=0}^{N-1} P(\theta = \zeta_i) f(\psi_0 - \zeta_i), \tag{3}$$

where N is the total number of expected phase changes for a given constellation c.

This metric is obtained prior to derotation because the derotator modifies the symbols phase according to the hypothesized constellation. The rotation, however, can be expressed as a constant phase error (θ_e) and (3) can be restated as

$$f_{\mathrm{pp}|c}(x) = \sum_{i=0}^{N-1} p(\theta = \zeta_i) f(\psi_0 - \zeta_i - \theta_e). \tag{4}$$

The value of θ_e that results in the lowest Hellinger distance is used to calculate the value of the phase profile metric. The Hellinger distance for the phase profiles is then expressed as

$$H_{\mathrm{pp}|c} = H\left(h_{\mathrm{pp}}(x), f_{\mathrm{pp}|c}(x)\right), \tag{5}$$

where $h_{pp}(x)$ is the the empirical histogram.

3.8. Amplitude Profile. The Amplitude profile examines the distribution of the magnitudes for the received symbols and compares it to the theoretical values. As with the phase profile, the measured magnitudes are grouped up into bins to form a histogram (h_{ap}). The theoretical PDF is obtained from the constellation description and is assumed to have a Rice distribution:

$$f(x,\nu) = \frac{x}{\sigma^2}\exp\left(\frac{x^2+\nu^2}{2\sigma^2}\right) I_0\left(\frac{x\nu}{\sigma^2}\right), \tag{6}$$

where ν is the amplitude, σ^2 is the symbol variance, and I_0 is a Bessel function with order 0. The theoretical PDF is then given as

$$f_{\mathrm{ap}|c}(c) = \sum_{i=0}^{N-1} p(\nu = \nu_i) f(x,\nu), \tag{7}$$

where ν_i is the ith amplitude and N is the number of possible amplitudes for constellation c. The amplitude profile metric is the Hellinger distance between the actual distribution and the theoretical distribution and can be expressed as

$$H_{\mathrm{ap}|c} = H\left(h_{\mathrm{ap}}(x), f_{\mathrm{ap}|c}(x)\right). \tag{8}$$

3.9. Symbol Variance and Distribution. Symbol distribution examines the number of received symbols for each constellation point. Symbol variance is a measure of the distance between the received symbol and the theoretical location of the symbol. These two metrics are combined to form a two-dimensional PDF of the received symbols. Assuming that in-phase and quadrature components of the symbol are independent random variables with Gaussian noise, the joint PDF can be expressed as

$$f_j(i,q) = \frac{1}{2\pi\sigma^2}\exp\left(\frac{\left(i-I_j\right)^2+\left(q-Q_j\right)^2}{\sigma^2}\right), \tag{9}$$

where I_j and Q_j are the expected in-phase and quadrature values for the jth symbol and σ^2 is variance of the noise. Assuming that the all symbols in a constellation are equally probable, the two-dimensional theoretical PDF is then

$$f_{\mathrm{sd}|c}(i,q) = \frac{1}{N}\sum_{i=0}^{N-1} f_i(i,q), \tag{10}$$

where N is the total number of symbols in constellation c.

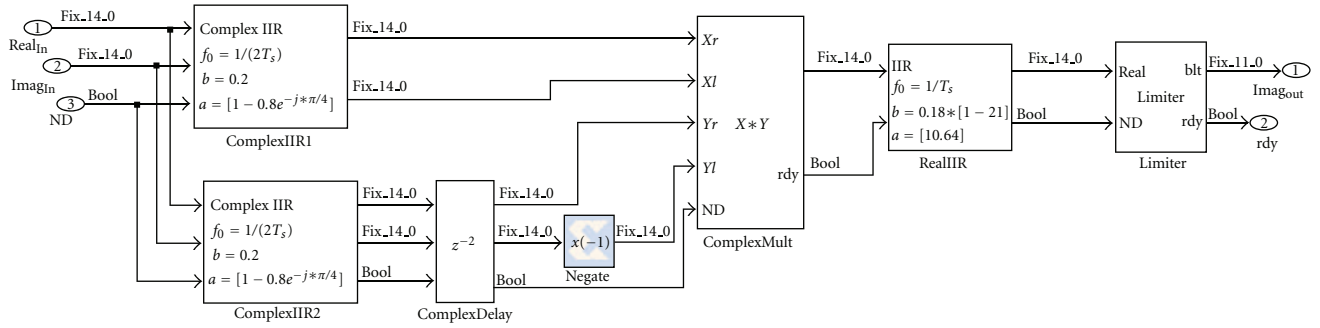


FIGURE 8: Spectral Line generator Diagram.

The empirical PDF is obtained using a two-dimensional histogram (h_{sd}) to group the received symbols into bins. The Hellinger distance can be used to measure the difference between the empirical and theoretical PDFs and is expressed as

$$H_{\text{sd}|c} = H\left(h_{\text{sd}|c}(x,y), f_{\text{ap}|c}(x,y)\right). \tag{11}$$

3.10. Symbol Transitions. This metric compares the distribution of symbol transitions to the theoretical values. An $M \times M$ matrix is populated with all the transitions observed, where the rows of the matrix represent the ith and the columns represent the ith+1 symbol. The matrix is then divided by the total number of observed transitions to obtain the empirical PDF. The theoretical PDF is obtained from the constellation's description file. For each symbol in the constellation, the description file has a list or range of valid symbols a given symbol can make a transition to. As with other metrics, the Hellinger distance is used

$$H_{\text{tm}|c} = H\left(h_{\text{tm}|c}(x,y), f_{\text{tm}|c}(x,y)\right), \tag{12}$$

where $h_{\text{tm}|c}(x,y)$ is the empirical transition matrix and $f_{\text{tm}|c}(x,y)$ is the theoretical transition matrix.

3.11. Combining the Metrics. All of the metrics discussed above provide information on the fitness of a hypothesis, but cannot on their own identify the correct hypothesis. A mechanism for combining the metrics in a manner that results in a single value that can be used to rank the hypothesized constellations is needed. Additionally, the mechanism must allow for easily modifying the probability of occurrence for any constellation. Artificial intelligence blocks such as neural networks were considered as a method for combining the metrics, but because they have to be trained, they lack the flexibility desired for the RapidRadio framework. A Bayesian network on the other hand does not require *a priori* training and has a natural mechanism for integrating the probability of occurrence of constellations.

Scoring is done in two stages. First the *a priori* probabilities of occurrence of all the constellations are pushed down to the processing node. At the same time the probability of each metric given a hypothesis is passed to the processing node. The new probability of each constellation is then calculated according to (13) below:

$$P(h_i \mid \text{pp}, \text{ap}, \text{sd}, \text{tm}) = \frac{P(\text{pp}, \text{ap}, \text{sd}, \text{tm} \mid h_i) P(h_i)}{\sum_{i=0}^{N-1} \left[P(\text{pp}, \text{ap}, \text{sd}, \text{tm} \mid h_i) P(h_i)\right]}. \tag{13}$$

Including the *a priori* probability of the hypothesized constellations in (13) ensures that constellations that are known to more likely have a higher probability of being chosen. pp, ap, sd, and tm are conditionally independent which allows the dividend of (13) to be defined as

$$\begin{aligned} &P(\text{pp}, \text{ap}, \text{sd}, \text{tm} \mid h_i) \\ &\quad = P(\text{pp} \mid h_i) P(\text{ap} \mid h_i) P(\text{sd} \mid h_i) P(\text{tm} \mid h_i). \end{aligned} \tag{14}$$

The probabilities for each metric are approximated as follows:

$$\begin{aligned} P(\text{pp} \mid h) &= 1 - H_{\text{pp}|h}, \\ P(\text{ap} \mid h) &= 1 - H_{\text{ap}|h}, \\ P(\text{sd} \mid h) &= 1 - H_{\text{sd}|h}, \\ P(\text{tm} \mid h) &= 1 - H_{\text{tm}|h}. \end{aligned} \tag{15}$$

This approximation assigns higher probabilities to hypothesis with empirical PDFs that closely resemble the theoretical PDFs. Using the Hellinger distance for all metrics ensures that they are equally weighed. The result of the Bayesian network is a set of probabilities that indicate the likelihood of all constellations. Situational awareness and knowledge from past experiences can be inserted into the model by modifying the *a priori* probabilities of each constellation. From (13), it can be seen that the joint probability of the metrics given the hypothesized constellation is normalized by the sum of the joint probabilities of the metrics for all hypothesized constellations. This normalization allows the addition of new constellations with little to no change to the classification algorithm.

4. Radio Deployment Phase

The radio deployment phase creates an FPGA-based receiver for the classified signal that produces a stream of symbols. Using a TCP/IP link, the framework starts the build on a server that hosts the vendor tools. When the build process has

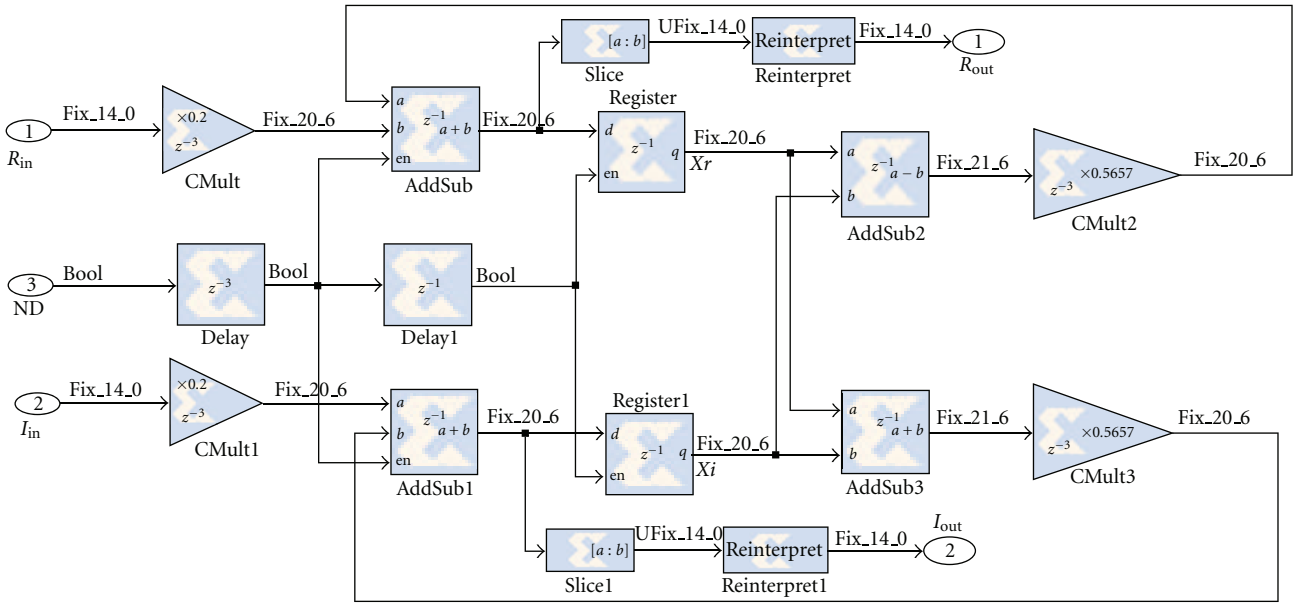


Figure 9: Complex IIR.

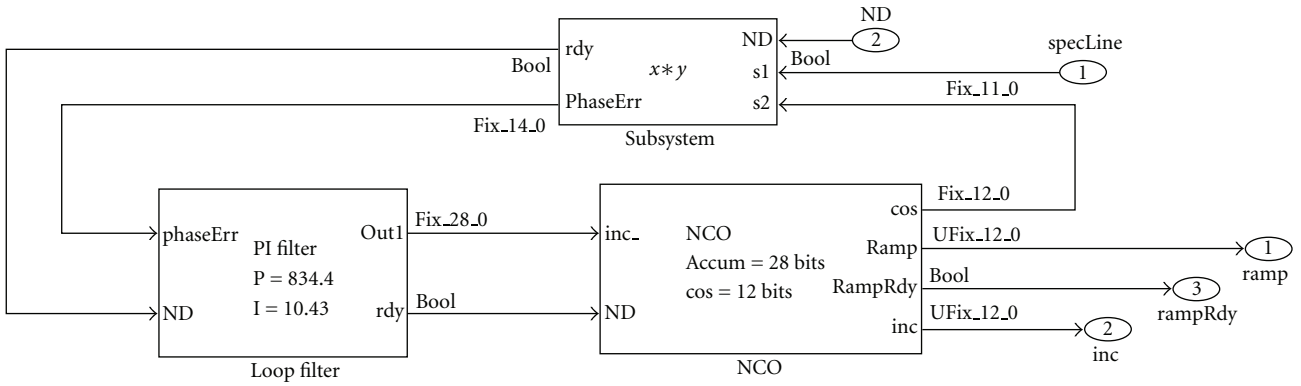


Figure 10: PLL Implementation.

completed, the new configuration bitstream is loaded into the target platform. A set of 81,000 noncontiguous symbols is then extracted from the platform and displayed on the user interface. A high-level description of the radio synthesis phase can be seen in Figure 5. The synthesis tool developed is written in C++ and builds on-top of other tools such as Makefiles, Xerces-C, Matlab, and Xilinx's Core Generator. The following sections discuss the major pieces of the radio synthesis phase.

4.1. Platform Description File. A goal of the RapidRadio framework is to reduce the amount of FPGA knowledge necessary to create a system. There are many parameters, however that are platform unique, such as ADC initialization, intercomponent communications, and output pin usage to name a few. To hide this information from the user, the framework could be made platform specific, building all the knowledge about a given platform into the system. This approach however is not very attractive because it would make the framework hard to modify to target a different platform.

The RapidRadio framework solves this problem by assuming that a top level design was previously created. This top level design serves as a host to the receiver and takes care of all the initialization and communication infrastructure. The framework therefore produces a *receiver* module which accepts an input data stream and produces an output data stream. This approach requires a one-time setup when a new platform is chosen to serve as a target, but the cost of designing and testing the top level design is relatively small when compared to the cost of building an entire new design every time.

4.2. Module Selection and Generation. The module database shown in Figure 5 does not contain the modules themselves, but module description files written in XML. The file can represent prebuilt modules or a set of rules dictating how to create a module. Definitions of the interfaces for the module

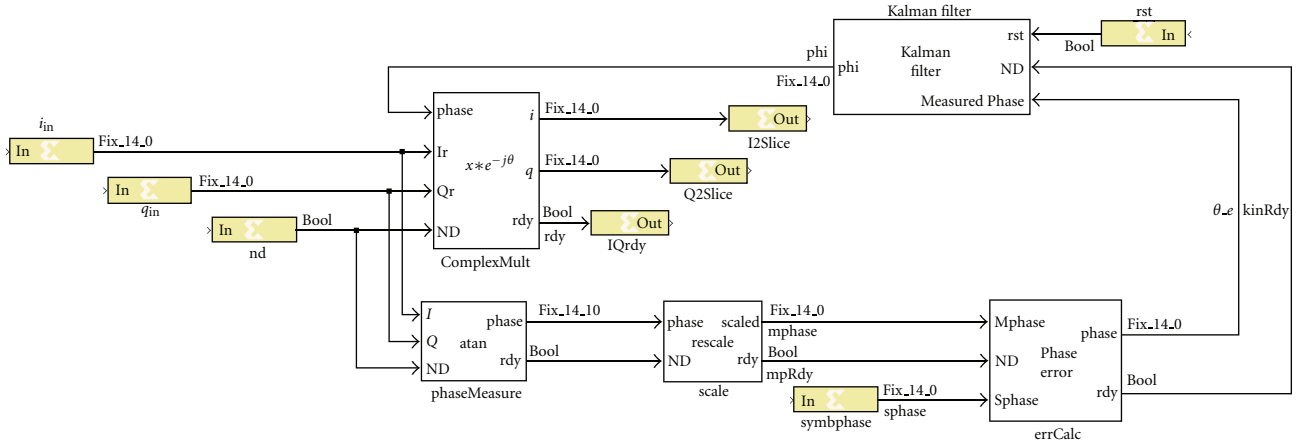


Figure 11: Derotator FPGA implementation.

and a list of all the ports associated with each interface are also contained. If the implementation is not device specific or the module has not yet been implemented, the description file will indicate that it can match any device. Otherwise the device type information is listed. Lastly all module parameters are listed in the file. The module description file uses some of the tags previously defined for the RDF and the platform description file.

When a module is encountered in the RDF, a database is searched for a matching module. Matches are based on the modules type/subtype, parameters, and the target device. Parameters that are hard-coded into the modules implementation are shown as having a specific value. If the database module does not have the same value as that requested in the RDF, then it is not considered a match. If the database module has a value of "configurable" for the parameter, then this parameter is ignored during the match process because the module can be configured to use the value provided in the RDF.

5. Receiver Architecture and Implementation

Receivers deployed by the RapidRadio framework share the general architecture shown in Figure 6. The signals sampling rate at each block is indicated by the blocks color. A complex mixer is used to bring the signal down to baseband. Decimating root-raised cosine filters are used for matched filtering and sampling rate reductions. A resampler circuit is used to recondition the signal, reducing the sampling rate to four times the symbol rate. Synchronization is then performed to produce a stream of symbols. The following sections describe in detail the FPGA implementation of these blocks. All blocks were designed for 14-bit interfaces. The bit-width of internal registers was determined at design time in such a manner as to prevent overflow with full-strength signals.

5.1. Downconverter. Using an automatically generated multiplier block and a 14-bit NCO, the signal is brought down to baseband. The multiplier is created using Xilinx's Core Generator. The output width of the multiplier (w_o) is inherited from the platform description file. The width of the multiplicands can be different because one is the output of the NCO which has a fixed output width of 14-bits and the other is the input data which has a width equal to the input width (w_{in}). The output width of the multiplier is then

$$w_o = 14 + w_{\text{in}}. \tag{16}$$

The effective output of the multiplier is $w_o - 1$ because both inputs are signed. To avoid bit growth, the output of the multiplier is truncated, to keep it the same size as the input. The MSB of the multiplier block is

$$\text{MSB} = w_o - 2, \tag{17}$$

and the LSB is

$$\text{LSB} = \text{MSB} - w_{\text{in}} + 1. \tag{18}$$

5.2. Matched Filtering. Matched filters are also created using the Core Generator. The roll-off factor is obtained from the RDF. The filter order is dependent on the number of samples per symbol at the filter's input and on the roll-off factor. The roll-off factor is used to determine how many symbols the filter should contain to minimize ISI. The length of the filter in symbols (l_s) is defined as

$$f_l = 4 * \left\lceil \sqrt{\frac{2}{\alpha}} \right\rceil. \tag{19}$$

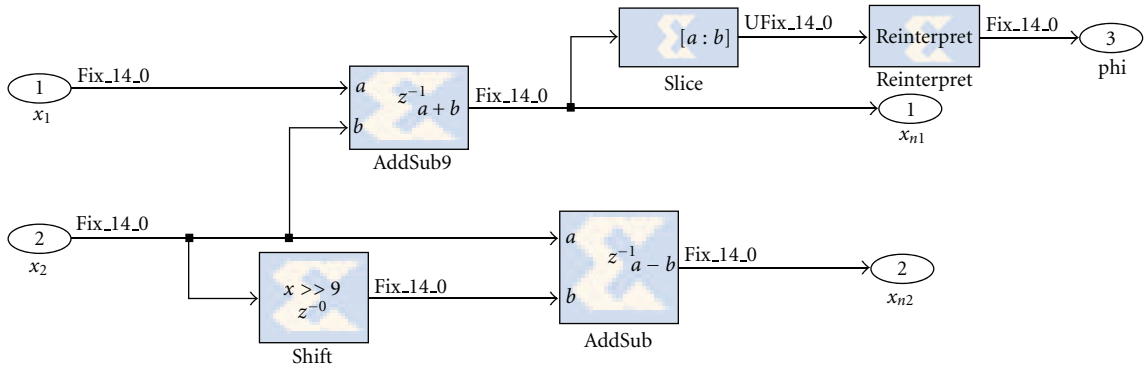
The number of samples per symbol at the input is:

$$s_s = \left\lfloor \frac{F_s}{f_s} \right\rfloor. \tag{20}$$

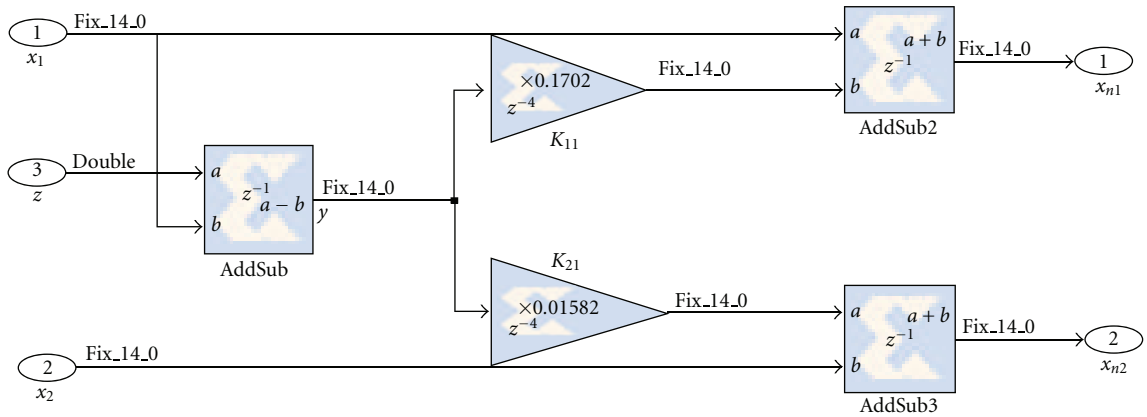
The filter order is then defined as

$$\max(64, s_s \cdot f_l). \tag{21}$$

The filter's decimation factor (η) is a function of the filter's input sampling rate (F_s) and the minimum required



(a) Prediction stage



(b) Update stage

Figure 12: Kalman filter FPGA implementation.

input sampling rate of the resampler (F_{sr}). η is calculated by the CLIPS back-end. Filter coefficients are obtained from Matlab using the sampling rate, symbol rate, and roll-off factor. Coefficients are then scaled and stored into a ".coe" file.

A filter of order N has $N + 1$ multiplications and N additions. Assuming that the filter coefficients are the same size as the input data, the width of the output (w_o) is approximately

$$w_o = 2w_{\text{in}} + \left\lceil \log_2\left(\frac{N+1}{\eta}\right)\right\rceil, \tag{22}$$

where w_{in} is the input width. As with the downconverter, the output of the filter is truncated to avoid bit-growth. Selecting which bits of the output will be used however is not an obvious choice. Selecting the w_{in} MSB will guarantee that the output never overflows, but may result in underflow. To obtain an estimate of which bits to use, the scaled coefficients are used to process the sampled data obtained in the spectrum sampling phase (see Section 3.2). The magnitude of the filtered signal is then used to determine the MSB:

$$\text{MSB} = \left\lceil \log_2(\max(x_s))\right\rceil, \tag{23}$$

where x_s is the filtered signal. This methodology reduces the possibility of overflow at the output of the filter, as long as the signal level is similar to that encountered during the analysis phase.

5.3. Resampler. The RapidRadio framework assumes that it does not control the sampling rate of the input data stream. The synchronization circuit however, requires an input sampling rate of $4f_s$. To obtain the sampling frequency required by the synchronizer, a resampler circuit is used. The resampler uses 16-bit accumulator and a Lagrange interpolator to produce a copy of the input signal at a lower sampling rate. Aliasing is avoided because the matched filters eliminated higher frequency components.

The accumulator is incremented by a fixed value κ for every input sample received. A new output sample is desired when the value of the accumulator is zero. The increment is calculated as follows:

$$\kappa = \frac{2^{16} f_{\text{out}}}{f_{\text{in}}}, \tag{24}$$

where f_{in} is the input sampling rate and f_{out} is the output sampling rate. When the accumulator wraps, the interpolator uses the last three samples received and the value of the

accumulator ($A(t_n)$) to create the new sample. The interpolation is then performed over the sample set $\{t_{n-2}, t_{n-1}, t_n\}$. Assigning time indexes -1, 0, and 1, respectively, to the samples gives the interpolation polynomials in (25).

$$\begin{aligned}\ell_0 &= \frac{x(t_n) + x(t_{n-2})}{2} - x(t_{n-1}),\\ \ell_1 &= \frac{x(t_n) - x(t_{n-2})}{2},\\ \ell_2 &= x(t_{n-1}),\\ \widetilde{x}(\tau) &= \tau^2\ell_0 + \tau\ell_1 + \ell_2,\end{aligned} \tag{25}$$

where $x(t)$ is the value of the original signal at time t and $\widetilde{x}$ is the resampled signal. For the resampler, the desired sampling instant (τ) is the distance between the sample at time t_{n-1} and the maximum value of the accumulator (A_{mx}) normalized to κ. τ is then expressed as

$$\tau = \frac{A_{\text{mx}} - A(t_{n-1})}{\kappa}. \tag{26}$$

5.4. Timing Recovery Implementation. The implemented architecture of the symbol timing recovery module is shown in Figure 7. The spectral line generator and one of the IIR filters are shown in Figures 8 and 9, respectively. Internally the IIR filter uses six bits for decimal representation in addition to 14 bits for integer representation. By only truncating to 14 bits at the output of the filter, the truncation error is reduced.

Notice that only the imaginary part of the complex multiplication is required for the spectral generation circuit. This permits the optimization of the multiplication to only require two multiplications and one addition. Given that the inputs to the multiplier are four 14 bit numbers, the output could require up to 29 bits of precision. Matlab simulations, however, showed that only 25 bits are required when the input signal to the synchronizer is fully scaled to $\pm 2^{13}$. To maintain a signal of 14 bits, the output of the multiplier is rescaled by shifting it right 11 bits. The output of the third filter is then passed through a limiter circuit that saturates at ± 725 to eliminate any amplitude modulation.

In the PLL, shown in Figure 10, the phase difference between the signal generated by the spectral generator and a cosine generated by a local NCO is calculated. A multiplier is used to measure the phase error. A PI filter is used for the PLL's loop filter. Due to the small size of the loop filters coefficients, they cannot be properly represented using a reasonable number of bits. Obtaining representable values requires that the normalization factor (μ), that converts the phase error into an NCO accumulator offset, be moved into the filter. An additional factor of 2^3 is used to upscale the filter values because multiplying by μ does not result in sufficiently large values. Truncating the three least significant bits of the output of the filter provides the increment correction to the NCO. The NCO's increment and accumulator are then used to calculate the perfect timing instant (γ_a), the timing error (τ), and the strobe signal. Table 1 shows some of the values used for the timing recovery module.

Table 1: Symbol synchronizer implementation values.

Variable	Value
Loop filter proportional constant (α)	2.44×10^{-6}
Loop filter integrating constant (β)	3.02×10^{-8}
Phase error size	14 bits
NCO input size	12 bits
NCO accumulator size	28 bits
NCO ramp output size	12 bits
μ	$2^{28}/2\pi$
τ size	11 bits

Table 2: Derotator implementation parameters.

Variable	Value
θ_p size	14 bits
sin/cos size	12 bits
ATAN processing elements	9
Phase error scaling	$(2^{13} - 1)/\pi$
Kalman filter register size	14 bits
Kalman gain $K_{1,1}$	0.1702
Kalman gain $K_{2,1}$	0.01582

Table 3: Modulation parameters obtained from RapidRadio framework for test over-the-air signals. The first row shows the nominal values used by the transmitter. The second row shows the values used for receiver deployment.

Modulation	E_sN_0 (db)	Value type	α	f_c (Hz)	f_s (Hz)
32QAM	9.18	Nominal	0.65	2000000	100000
		Deployed	0.54	1999314	100010
16QAM	7.36	Nominal	0.65	2000000	250000
		Deployed	0.64	1999130	250039
8QAM	8.50	Nominal	0.75	2000000	350000
		Refined	0.74	1999806	350046
QPSK	12.56	Nominal	0.65	2000000	250000
		Deployed	0.68	1999211	250044

5.5. Derotator Implementation. The derotator is implemented in two blocks. The first is comprised of the static portion of the architecture that is not dependent upon the constellation. This block, shown in Figure 11, was created in System Generator. The complex mixer derotates the input signal using a predicted phase error and a sin/cos lookup table. The phase of the input symbol is calculated using Xilinx's CORDIC atan core. The output of the atan block, which is in the range of $[-\pi, \pi]$ is normalized to a power of 2 in order to facilitate further operations, such as wrapping. The phase output of the slicer is then used to calculate the phase error of the input symbol. The measured phase error is in the equivalent range of $[-2\pi, 2\pi]$ because it is calculated using a subtraction. The error is re-normalized to $[-\pi, \pi]$ by adding 2π if the value is less than $-\pi$ or adding -2π if the value is greater than π. Table 2 shows the implementation parameters for the derotator.

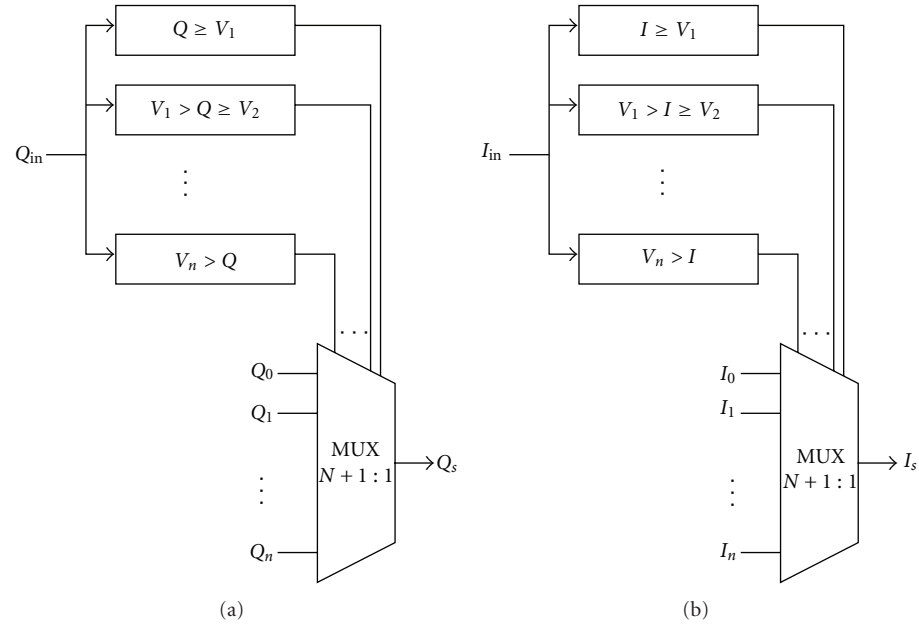


Figure 13: Grid-based slicer architecture.

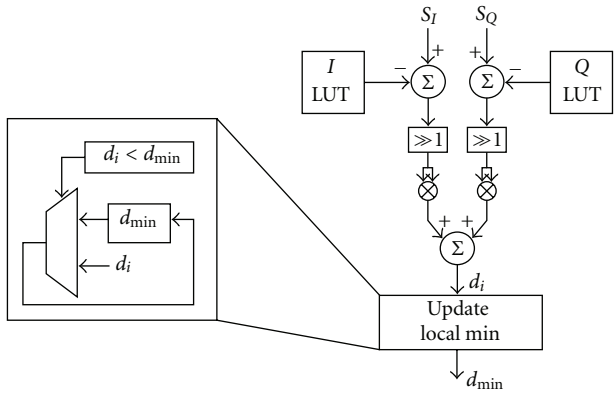


Figure 14: Distance block.

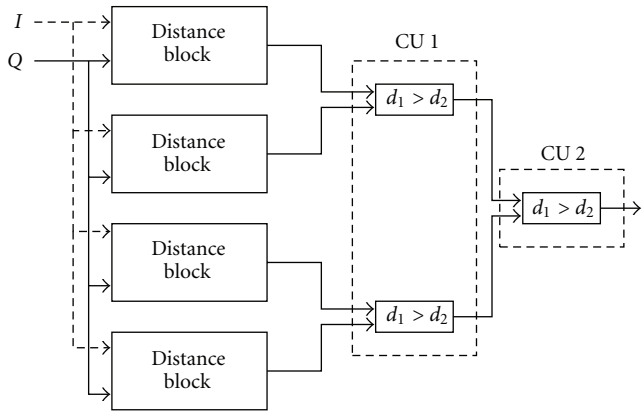


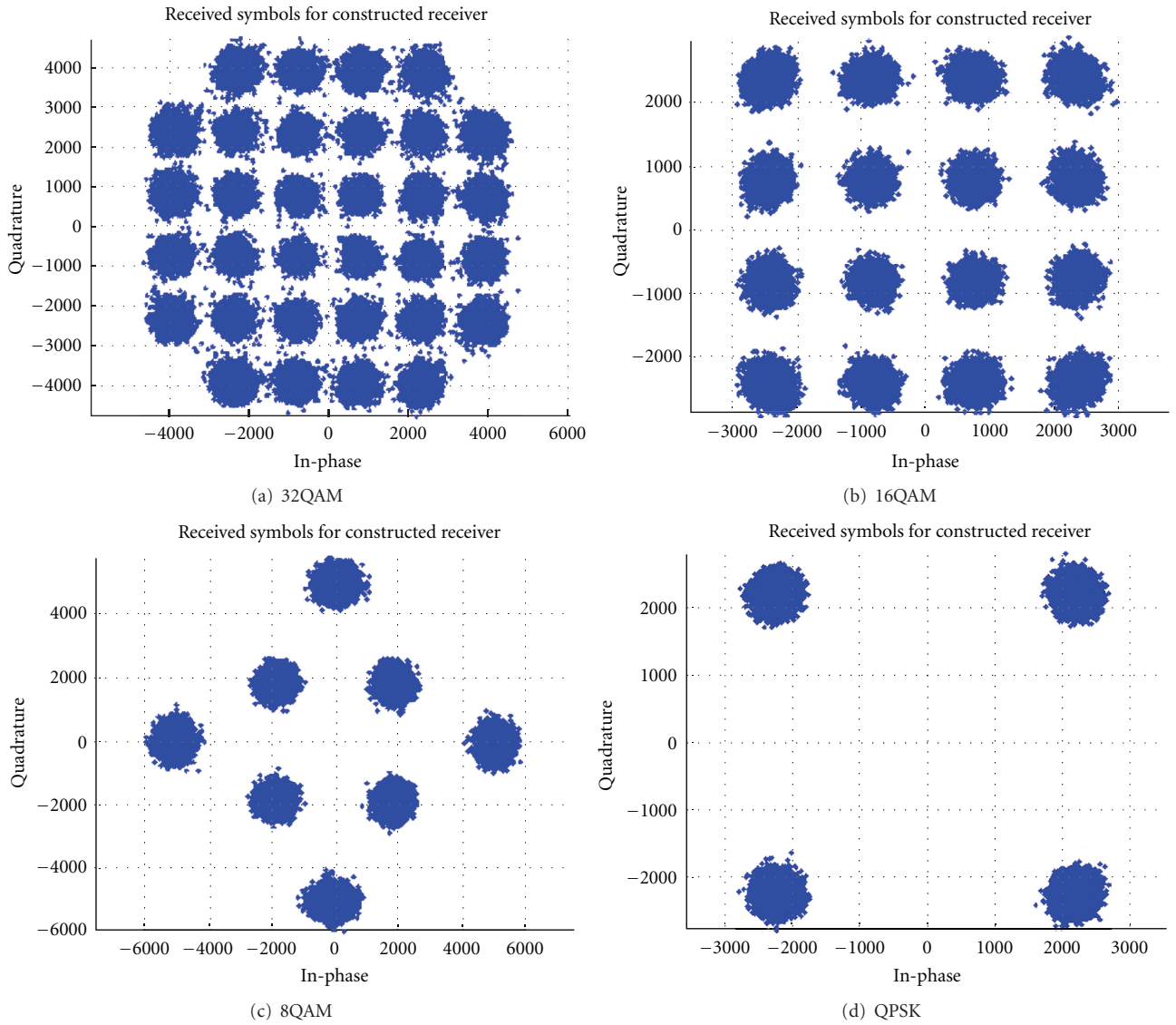
Figure 15: Distance-based slicer architecture.

As mentioned in Section 3.5, the Kalman gains are static and precalculated at design time. Their values are shown in Table 2. Figure 12 shows the FPGA implementation of both the prediction and the update stage of the Kalman filter.

Because the slicer block is constellation unique, it is generated at implementation time using a custom C++ HDL generator. A constellation name and the average signal power are provided as input. A Matlab engine is opened by the generator and the constellation description is loaded. All constellations are stored in XML files and contain a list of the constellation points. The magnitude of the constellation points has been normalized to obtain an average power of one. A Matlab script is then used to determine the slicer architecture to be used. Constellation points are scaled by the average power to ensure proper slicing.

The grid-based architecture is shown in Figure 13. Values V_1 through V_n are the decision boundaries. The input values are compared with the boundaries and the correct symbol is chosen using two $n + 1 : 1$ muxes, where n is the number of boundaries. This architecture is fairly efficient as it only requires $2n$ comparators and four lookup tables. All comparisons run in parallel, which provides a slicing latency of one clock cycle.

A distance-based slicer works for any constellation but is computationally expensive. For every point in the constellation, two subtractions, two multiplications, and one addition must be done. Additionally, $\log_2(M)$ comparisons are needed to determine the shortest distance, where M is the number of points in the constellation. Ideally every constellation point would be processed in parallel, but this would not take advantage of the pipelining built into the multiplier blocks. The resource requirements would also be excessive. For 64QAM, for example, 128 multipliers would



(a) 32QAM (b) 16QAM

(c) 8QAM (d) QPSK

Figure 16: Sample symbols extracted from the deployed receivers. Clustering of symbols indicates that synchronization was properly achieved.

Table 4: Resource utilization for deployed receivers on a Xilinx Virtex 4 XC4VLX60 FPGA, using a sampling rate of 8 MHz and an operating clock rate of 192 MHz.

			RRCOS filter		
Constellation	Order	MSB	DSP48	RAMB16	Slices
32QAM	608	31	23 (35%)	27 (16%)	7242 (27%)
16QAM	248	30	25 (39%)	31 (19%)	4348 (16%)
8QAM	176	29	25 (39%)	21 (13%)	5189 (19%)
QPSK	256	30	27 (42%)	33 (20%)	4403 (16%)

be required. A simple solution is to multiplex access to the multipliers.

The RapidRadio distance-based slicer uses a distance block for every four constellation points. Each block has two lookup tables containing the I and the Q values for four symbols. Calculating the distance between the input and each of the symbols in the table is accomplished using the circuit in Figure 14. Shifting the output of the subtractor right one bit keeps the signal 14 bits wide. The sum of the squares is then calculated (d_i). Each distance is compared to the previous minimum $d_{\min}$. If it is lower, then the minimum distance is updated to the current value. This process is

shown on the left side of Figure 14. The output of the distance block is the smallest distance and the location and phase of the corresponding symbol (the latter is not shown in Figure 14). Multipliers have a pipeline depth of four, and the addition and subtraction circuits have a latency of one cycle. With the the comparison and control logic, the distance block has a total latency of 10 cycles.

Slicing constellations with more that four symbols is accomplished by using more than one distance unit. For a given constellation, a total of $U = M/4$ distance blocks are required. A set of cascading comparison units is used to merge the output of the distance blocks to obtain the constellation point. Each comparison unit divides the L inputs into $L/2$ pairs. For each pair, a relational operator is used to eliminate the largest distance. A total of $\log_2(U)$ comparison units are necessary to obtain the results because each comparison unit reduces the number of candidate symbols by a factor of two. Given that each comparison units has a latency of one clock cycle, the distance-based slicer then has a total latency of $10+\log_2(U)$. Figure 15 shows the overall architecture of the slicer.

6. Results

This section discusses the performance of the RapidRadio framework prototype. Three signals were generated and transmitted over the air at 2.05 GHz. The framework was used to classify the signal, determine the modulation parameters, and synthesize the radio. The modulation schemes used for the tests were QPSK, 16QAM, and 32QAM. Both the transmitter and the receiver were implemented in the Harris SDR SIP package which contains four Xilinx Virtex 4 XC4VLX60 FPGAs. Separate systems and clock sources were used for the transmitter and receiver to ensure that synchronization is not achieved simply because the same clock is used for both systems.

For each signal, the constellation chosen by the framework was validated and the receiver deployment phase was started. The synthesis phase was executed on linux machine, using the standard vendor tools. A TCP/IP daemon running on the ARM processor in the development board then loads the newly created receiver bitstream onto the FPGA connected to the receiver RF chain. Figure 16 shows the extracted symbols for the test constellations. It can be observed that for all constellations, synchronization was properly recovered because the extracted symbols are clustered appropriately.

Table 3 shows the modulation parameters for all test signals. The first row shows the nominal values used by the transmitter. The second row contains the values obtained by the framework, which are used to deploy the receiver. It can be observed that the parameters obtained by the framework are very similar to the nominal values.

Table 4 shows the resource utilization for the deployed systems as well as the instantiation parameters for the matched filters. As expected the signals with the lowest data rate require high-order filters because of the high number of samples per symbol at the input sampling rate. The highest system utilization is observed by the constellations that require a distance-based slicer (32QAM and 8QAM).

7. Conclusion

In this paper, the RapidRadio framework for signal classification and receiver deployment was discussed. The framework guides the user through the process of determining the modulation type of an unknown signal and building an FPGA-based receiver capable or demodulating the signal. Reducing the scope of the framework narrows down the possible implementations and allows the hiding of implementation details.

Unknown signals were classified using the four metrics: the phase profile, the amplitude profile, the symbol distribution, and the symbol transition matrix. Using a Bayesian network, the metrics were combined to assign each hypothesis a probability. Using a Bayesian network provides added flexibility to the framework by ensuring that it can automatically adjust itself to new constellations.

The frameworks functionality was verified by capturing off-the-air signals. All signals were properly classified and functional receivers were deployed on a Virtex-4 FPGA.

Acknowledgment

The authors would like to thank the Harris Corporation, Government Communications Division for supporting this research.

References

[1] O. A. Dobre, A. Abdi, Y. Bar-Ness, and W. Su, "Survey of automatic modulation classification techniques: classical approaches and new trends," *IET Communications*, vol. 1, no. 2, pp. 137–156, 2007.

[2] Xilinx Inc., *System Generator for DSP: Getting Started Guide*, Xilinx Inc., 2007.

[3] G. J. Minden, J. B. Evans, L. Searl et al., "KUAR: a flexible software-defined radio development platform," in *Proceedings of the 2nd IEEE International Symposium on New Frontiers in Dynamic Spectrum Access Networks (DySPAN '07)*, pp. 428–439, Dublin, Ireland, April 2007.

[4] K. Kim, I. A. Akbar, K. K. Bae, J.-S. Um, C. M. Spooner, and J. H. Reed, "Cyclostationary approaches to signal detection and classification incognitive radio," in *Proceedings of the 2nd IEEE Symposium on New Frontiers in Dynamic Spectrum Access Networks (DySPAN '07)*, pp. 212–215, Dublin, Ireland, April 2007.

[5] A. Fehske, J. Gaeddert, and J. H. Reed, "A new approach to signal classification using spectral correlation and neural networks," *Proceedings of the 1st IEEE International Symposium on New Frontiers in Dynamic Spectrum Access Networks (DySPAN '05)*, pp. 144–150, November 2005.

[6] J. Y. J. Chen and R. Morelos-Zaragoza, "Design and implementation ofan algorithm for modulation identification of analog and digital signals," in *Proceedings of the Software Defined Radio Forum Technical Conference (SDR '06)*, SDR Forum, Orlando, Fla, USA, 2006.

[7] L. Liu and J. Xu, "A novel modulation classification method based on high order cumulants," in *Proceedings of the International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM '06)*, pp. 1–5, September 2006.

[8] B. Le, F. A. G. Rodriguez, Q. Chen et al., "A public safety cognitive radio node," in *Proceedings of the Software Defined Radio Forum Technical Conference (SDR '07)*, SDR Forum, Denver, Colo, USA, November 2007.

[9] B. Le, T. W. Rondeau, D. Maldonado, D. Scaperoth, and C. W. Bostian, "Signal recognition for cogntive radios," in *Proceedings of the Software Defined Radio Forum Technical Conference (SDR '06)*, SDR Forum, Orlando, Fla, USA, 2006.

[10] "*CLIPS Reference Manual: Volume I Basic Programming Guide*," December 2007.

[11] P. Stoic and R. Moses, *Spectral Analysis of Signals*, chapter 2, Prentice Hall, New York, NY, USA, 2005.

[12] A. Recio, J. Surís, and P. Athanas, "Blind signal parameter estimation for the rapid radio framework," in *Proceedings of the Military Communications Conference (MILCOM '09)*, IEEE, October 2009.

[13] Y. Tachwali, W. J. Barnes, and H. Refai, "Configurable symbol synchronizers for software-defined radio applications," *Journal of Network and Computer Applications*, vol. 32, no. 3, pp. 607–615, 2009.

[14] P. Zicari, E. Sciagura, S. Perri, and P. Corsonello, "A programmable carrier phase independent symbol timing recovery circuit for QPSK/OQPSK signals," *Microprocessors and Microsystems*, vol. 32, no. 8, pp. 437–446, 2008.

[15] D. N. Godard, "Passband timing recovery in an all-digital modem receiver," *IEEE Transactions on Communications*, vol. 26, no. 5, pp. 517–523, 1978.

[16] M. Simon, "Noncoherent symbol synchronization techniques," in *Proceedings of the Military Communications Conference (MILCOM '05)*, vol. 5, pp. 3321–3327, IEEE, October 2005.

[17] C. Dick, F. Harris, and M. Rice, "FPGA implementation of carrier synchronization for QAM receivers," *Journal of VLSI Signal Processing*, vol. 36, no. 1, pp. 57–71, 2004.

[18] C. Dick and F. Harris, "FPGA QAM demodulator design," in *Proceedings of the 12th International Conference of Field-Programmable Logic and Applications: Reconfigurable Computing Is Going Mainstream (FPL '02)*, pp. 279–287, 2002.

[19] D. W. Scott, "On optimal and data-based histograms," *Biometrika*, vol. 66, no. 3, pp. 605–610, 1979.

[20] X. Huo and D. Donoho, "Simple and robust modulation classification method via counting," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP '98)*, vol. 6, pp. 3289–3292, May 1998.

[21] K. Umebayashi, J. Lehtomäki, and K. Ruotsalainen, "Analysis of minimum hellinger distance identification for digital phase modulation," in *Proceedings of the IEEE International Conference on Communications (ICC '06)*, pp. 2952–2956, July 2006.

[22] R. F. Pawula, S. O. Rice, and J. H. Roberts, "Distribution of the phase angle between two vectors perturbed by gaussian noise," *IEEE Transactions on Communications*, vol. 30, no. 8, pp. 1828–1841, 1982.