RAPID RADIO: HUMAN-ASSISTED SIGNAL CLASSIFICATION AND RECEIVER IMPLEMENTATION

Jorge A. Surís (Virginia Tech, Blacksburg, VA, USA; jasuris@vt.edu); Adolfo Recio (Virginia Tech, Blacksburg, VA, USA; recio@vt.edu); and Peter Athanas (Virginia Tech, Blacksburg, VA, USA; athanas@vt.edu)

ABSTRACT

An analysis-based framework for the rapid development of a radio receiver for signals with unknown parameters is proposed, exploiting the reconfiguration capabilities of FPGAs. The framework guides a non-expert user through the process of signal classification and FPGA-based receiver implementation. System efficiency is traded off with implementation time in order to allow fast radio creation. A set of high-level transformations are applied to the unknown signal based on different hypothesis about the modulation scheme. The results of the transformations are presented to the user, who can steer the process of analysis. The parameters of the radio are then mapped by means of an Implementation Engine to modules implemented in a general purpose FPGA-based receiver.

1. INTRODUCTION

There are many circumstances when a signal intelligence expert is faced with the need to quickly extract data from an otherwise unknown signal, but lacks the knowledge of how the signal was modulated. The traditional approach to this would be to go through an *analysis phase* to construct a hypothesis of the signal's modulation parameters, an then enter a *design phase* to create the required custom radio receiver. This process is a slow progression of steps from signal analysis, receiver modeling, receiver design, to prototyping. This process mimics the traditional approach towards radio receiver design where a fairly efficient radio is ultimately realized. Efficiency is usually measured in terms of hardware resources used, such as gates and multipliers, or power consumption. With the progression of technology, many of these metrics are outdated or even meaningless when instant gratification is needed. For example, minimizing the number of multipliers may be moot if the target platform contains multiple FPGAs with thousands of otherwise unused multipliers is available. If the speed of deployment is paramount, and the prototyping platform has an abundance of resources, then the design process can be altered in a way to produce an operational yet possibly suboptimal receiver in a short time.

In this paper we present an analysis-based approach to receiver implementation called the Rapid Radio framework. This framework is a human-guided framework that provides the user with the required tool set for classifying an unknown signal and creating an FPGA-based implementation of the receiver. The analysis of a signal is broken down into several steps where high-level transforms are applied to the signal. At each step, variations of a given transform are applied to the signal, the results of which are shown to the user via a graphical user interface (GUI) and a series of metrics. With the provided information, the user can choose which of the transforms is correct. By choosing the correct transform, the user is in essence defining the receiver architecture. The results of the chosen transform are then used as the input to other transforms. For example once timing recovery has been accomplished, a series of different timing, phase and frequency algorithms are applied to the signal and the resulting constellations are shown to the user. The user can then choose which constellation looks correct.

After the necessary transforms have been applied and the symbols have been identified, a Radio Description File (RDF) is created detailing the modules required for the receiver based on the transforms applied to the signal. An all digital, FPGA-based implementation of the receiver is then provided to the user.

2. RELATED WORK

A great amount of effort has been devoted to the problem of automatic modulation classification (AMC). In [1], the author presents a comprehensive overview of the state of the art in the field, and presents a broad classification of AMC algorithms as likelihood based (LB) and feature based (FB). Cyclostationary techniques to classification are presented in [2]. Approaches directed towards the implementation of actual radios based in multiple features are presented in [3] and [4]. In particular, they explore the use of the quasi-baseband signal obtained after down-conversion, which is readily available in a combined radio implementation/modulation classification approach. The Rapid Radio framework builds on top of the tools and techniques developed from the AMC field to allow a user to analyze an unknown signal and create an FPGA based implementation.

3. PROPOSED FRAMEWORK

The Rapid Radio framework is an analysis tool that directly leads to the creation of a rapid prototype of a receiver. Similar to other analysis tools, it is designed to aid the user in the classification of an unknown radio signal. The unique aspect of this framework is that the radio architecture description is generated by the tool as a result of a guided analysis process. This process abstracts away much of the knowledge required by the process of radio design and implementation.

Due to the complex nature of signal classification a "human in the loop" approach is used in the Rapid Radio framework. Instead of attempting a full automatic classification of the signal, the framework presents the results of the intermediate tests to the user, allowing the user to steer the identification and radio implementation processes. The framework is to be initially tested over the phase/amplitude digital modulation family of signals and will have a Matlab front end.

Currently the framework consists of the following six major transformation types:

- Signal identification and isolation
- Modulation parameter estimation
- Symbol timing recovery
- Constellation identification
- Symbol mapping
- FPGA-based receiver implementation

These transform will allow the user to determine the modulation type used for the signal and produce a receiver implementation.

3.1. Signal identification and isolation

The frequency band of interest must be selected by the user in order to eliminate any other signals present in the spectrum that may interfere with the signal classification process. Figure 1 shows the GUI used for signal isolation. A periodogram of the captured spectrum is shown to the user. Using the scrollbars at the bottom left, the user selects the area on which further processing will be done.

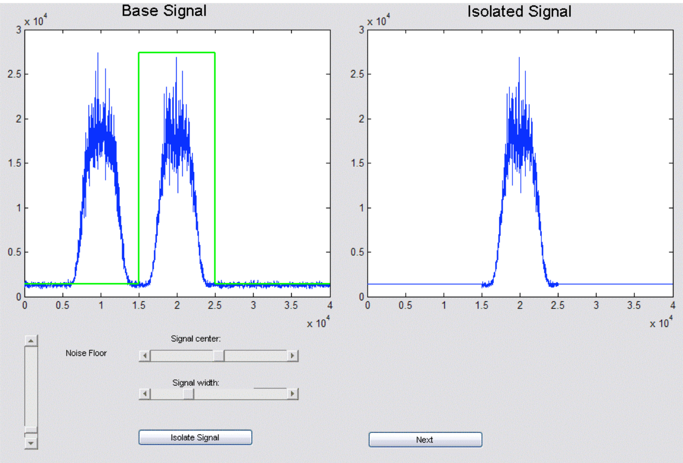


Figure 1. Signal isolation GUI

3.2. Modulation parameter estimation

A modulation parameter estimator in the frequency domain is achieved by fitting an analytic description of the spectral shape of the signal. Based on user input of likely modulation formats, several spectral fittings are applied to the isolated signal power spectrum. The fitting is then presented to the user. For linear amplitude/phase modulations, if the transmitted pulse has a square root raised cosine pulse shape, the power spectrum will have a raised cosine spectral shape, which has a non-linear dependence on the parameters f_c, f_{sym}, and α. Use of the unsmoothed periodogram is required to avoid bias in the estimate of the roll-off factor α. A non-linear optimization method is to be applied to find the least square error fitting of the spectral shape. The dogleg method [5] allows for a computationally inexpensive solution to the problem. An example display of a fitting under the square root raised cosine hypothesis is presented in Figure 2.

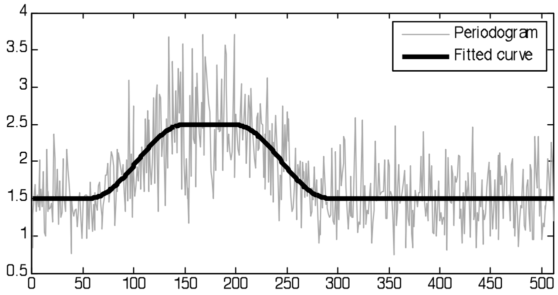


Figure 2. Spectral fitting of RRCOS PSK signal

Using the estimated modulation parameters, the signal's bandwidth can be estimated and an isolation band-pass filter is synthesized and applied to the signal. The estimate of the desired signal's center frequency is used to down-convert it to baseband. The signal is re-sampled to four times the estimated symbol rate, as a preconditioning step in order to apply the timing recovery algorithm. Lastly a matched baseband filter is created based on the estimated parameters and applied to the baseband signal.

3.3. Symbol timing recovery

In order to synchronize a signal with a high degree of uncertainty around its nominal symbol rate, a high carrier frequency offset robust symbol synchronizer was implemented. A modified Gardner [6] timing error detector which utilizes four samples per symbol is proposed for obtaining a normalized timing error measurement. Using a larger number of samples per symbol results in a synchronizer with a higher symbol rate error tolerance. The increased tolerance compensates for errors in the estimated values. The Gardner symbol synchronizer was also chosen because it can be eas-

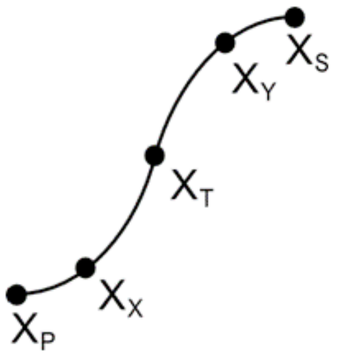


Figure 3. Sampling for timing error detector

ily modified to work with more complex modulation schemes. Because the modifications to the algorithm preserve the rotation invariance of the Gardner detector, symbol timing recovery is possible in presence of carrier frequency offset. Carrier synchronization is to be performed afterwards jointly with the modulation classification task.

Figure 3 presents a simplified diagram of samples at a symbol boundary. X_P represents the strobe sample at the previous symbol. The strobe sample is the sample containing the information for the symbol. X_T represents the timing measurement sample. X_S represents the next strobe sample. X_X and X_Y are auxiliary samples required for normalization. The samples are represented by 2×1 vectors denoting the sample's coordinates in the I-Q plane.

If we assume that X_X, X_T, and X_Y lie on a straight line, then the normalized timing error measurement is given by Equation 1:

$$\boldsymbol{\tau} = \frac{1}{2} \frac{\langle \boldsymbol{X_T}, \boldsymbol{d} \rangle}{\|\boldsymbol{d}\|^2}$$

Equation 1. Normalized timing error

Where $d = X_Y - X_X$. This timing error measurement is normalized to symbol interval units. Therefore, a value of $\tau > 1/8$ indicates that a *fill* operation is required (make X_S=X_Y, and advance three samples); a value $\tau < 1/8$ indicates that a *skip* is required (ignore strobe and advance one sample). Otherwise, coarse timing is accepted and timing is *shifted* by four samples to the next symbol. In all cases, a polyphase filter adjusted by the value of τ is required to interpolate the signal at the optimum sampling instant.

The timing error measurement can only be applied when a transition between different symbols occurs and is prone to self-noise which is aggravated by the construction of the receiver's matched filter based on inexact estimates. Therefore, a tracking algorithm must be included in order to follow the symbol timing. A Kalman filter based on a frequency and phase state model of the sampling offset relative to the symbol timing is used to predict the symbol timing when a measurement is unavailable. The model is updated when actual measurements are performed. A clear advantage of the Kalman filter approach is that the value of the state variable for the frequency offset can be used to refine the initial symbol rate estimate. Furthermore, the Kalman filter shows rapid convergence over a wide range of frequency offsets, which is a requirement when only a coarse estimate of the symbol rate is available.

The results of these transformations are shown to the user in order to confirm convergence of the estimators (i.e.: assessment of an eye diagram is a useful tool to test sampling time correctness). Constellation points prone to carrier frequency error (rotating) are presented to the next stage.

3.4. Constellation identification

The problems of de-rotation and constellation test are interdependent, as no hypothesis on the constellation can be tested without de-rotation; and vice versa: no de-rotation is possible if the constellation is unknown [3, 7]. A joint method based in successive de-rotation, evaluation of the likelihood of the hypothesized constellations given the quasi-baseband sequence, and evaluation of the likelihood of the quasi-baseband sequence given the hypothesized constellations is proposed. A general diagram of the process is shown in Figure 4.

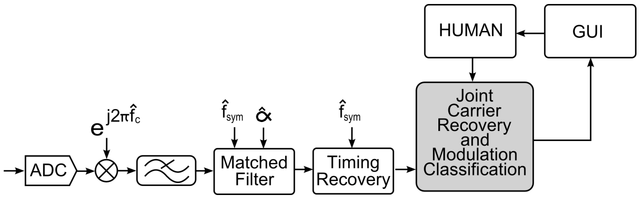


Figure 4. Constellation identification block

PLLs are currently used to recover the carrier information; however, as the initial parameters are gross estimates, potential hangup may occur within this approach. A feedforward approach is proposed, with double application of slicers. A slicer is applied before phase correction, in order to obtain the phase shift slope inside the decision regions. A non-linear Kalman filter with frequency and phase states is applied to the wrapped phase in order to obtain an unwrapped phase vector. The phase boundaries are determined by the description of the constellation under test. The conjugate phase vector is then multiplied with the rotated symbol sequence in order to achieve de-rotation. Figure 5 illustrates this approach. As in the symbol recovery case, the state vector can be used to refine the initial carrier frequency estimate.

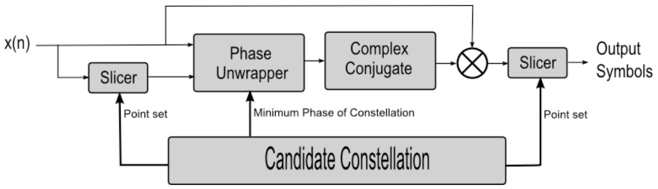


Figure 5. Symbol de-rotation

This process is to be performed for every hypothesized constellation. In order to perform a likelihood test for a given constellation, power normalization is required before the de-rotation and hypothesis testing blocks. The likelihood of the constellation can be measured in two ways:

Constellation error: Measure the power of the difference of the signals obtained before and after the output slicer. In the case of the true constellation, this measurement corresponds to the additive noise power. The drawback of this measurement is that it does not allow for differentiation of nested constellations.

Symbol distribution: For a given constellation, a relative occurrence of the symbols included in the constellation set is expected at the output slicer. A metric is developed based in the consideration of the underlying Bernoulli process that is mapped to the constellation points. Differential and Trellis coded modulations are to be tested according to the symbol transition statistics observed at this stage. This measurement fully differentiates nested constellations, and therefore complements the constellation error metric.

An advantage of the presented scheme is that the architecture of the constellation identification block is modulation independent, and only requires the input of the constellation description, consisting on the set of points, the absolute probability of each point and the transition probabilities among the constellation points.

3.5 Symbol mapping

A library with candidate bit groups to symbol mappings is provided to the user in order to identify symbols. Provision is made for passing the results for further processing, as channel decoding and descrambling.

3.6. FPGA-Based Receiver Implementation

In addition to aiding the user to classify an unknown signal, the Rapid Radio framework contains an Implementation Engine, which guides the user through the process of generating an FPGA-based implementation of the radio receiver. The Implementation Engine receives as input a Radio Description File (RDF) and a top level Hardware Description File (HDF). The implementation engine's output is a receiver module definition in HDL. Any object files, such as an ngc, required to instantiate the module in the target platform will also be provided. Because a large part of the implementation process is automated, the user is shielded from the intricacies of HDL design. This reduces the required knowledge base for the user, thereby increasing the possible user-base.

3.6.1. Radio Description File

The Radio Description File contains the list of all modules required for the receiver and how they are connected. The RDF is constructed from the list of transforms applied to the signal at analysis time. Because of its structured format and the large number of parsing libraries available, eXtensible Markup Language (XML) will be used for the RDF. A new XML schema will be created to define the syntax of the RDF. Module definitions in the RDF consist of the module name (which must be unique), a module type and subtype, a high-level definition of the module's interface, and any type-specific parameters. Because the module's ports are implementation-dependant, they are not known at analysis-time, therefore only a high-level definition of the modules interface is provided.

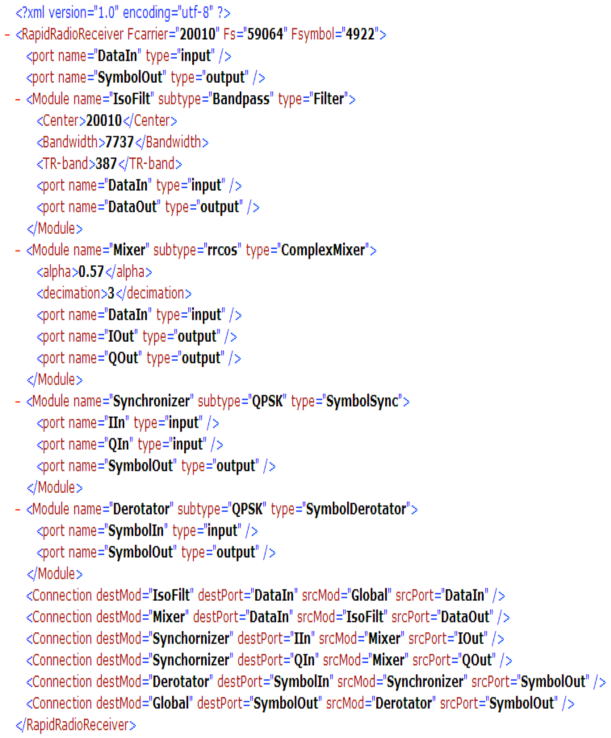
A modules parameters define attributes of the module which are specific to its type or subtype. A band-pass FIR filter, for example, will have its center frequency, bandwidth and transition band width as parameters, whereas a complex mixer would have the mixing frequency. Some parameters which affect more than one module, such as the sampling frequency or the symbol rate are included as global parameters.

As previously mentioned, a list of connections between modules is included in the RDF. These specify how the module interfaces interconnect. A connection element include the names of the source and destination modules and the source and destination interfaces. A module name of Global is used to indicate that the interface belongs to the receiver module.

Storing the results of signal classification in the RDF separates signal classification process from the receiver implementation processes. This separation allows for the quick re-implementation of receivers for previously identified signals. Figure 6 shows a sample RDF.

3.6.2. Hardware Description File

The Hardware Description File consists of platform specific information required for the implementation of the receiver, such as the system clock frequency, the location of a top level HDL file and the FPGA vendor and model. The top level HDL must include any logic necessary to obtain the input data, such as ADC initialization and administration, as well as what to do with the data produced by the receiver. This allows the generation of the receiver to be platform independent.



```
  <?xml version="1.0" encoding="utf-8" ?>
- <RapidRadioReceiver Fcarrier="20010" Fs="59064" Fsymbol="4922">
    <port name="DataIn" type="input" />
    <port name="SymbolOut" type="output" />
  - <Module name="IsoFilt" subtype="Bandpass" type="Filter">
      <Center>20010</Center>
      <Bandwidth>7737</Bandwidth>
      <TR-band>387</TR-band>
      <port name="DataIn" type="input" />
      <port name="DataOut" type="output" />
    </Module>
  - <Module name="Mixer" subtype="rrcos" type="ComplexMixer">
      <alpha>0.57</alpha>
      <decimation>3</decimation>
      <port name="DataIn" type="input" />
      <port name="IOut" type="output" />
      <port name="QOut" type="output" />
    </Module>
  - <Module name="Synchronizer" subtype="QPSK" type="SymbolSync">
      <port name="IIn" type="input" />
      <port name="QIn" type="input" />
      <port name="SymbolOut" type="output" />
    </Module>
  - <Module name="Derotator" subtype="QPSK" type="SymbolDerotator">
      <port name="SymbolIn" type="input" />
      <port name="SymbolOut" type="output" />
    </Module>
    <Connection destMod="IsoFilt" destPort="DataIn" srcMod="Global" srcPort="DataIn" />
    <Connection destMod="Mixer" destPort="DataIn" srcMod="IsoFilt" srcPort="DataOut" />
    <Connection destMod="Synchornizer" destPort="IIn" srcMod="Mixer" srcPort="IOut" />
    <Connection destMod="Synchornizer" destPort="QIn" srcMod="Mixer" srcPort="QOut" />
    <Connection destMod="Derotator" destPort="SymbolIn" srcMod="Synchronizer" srcPort="SymbolOut" />
    <Connection destMod="Global" destPort="SymbolOut" srcMod="Derotator" srcPort="SymbolOut" />
  </RapidRadioReceiver>
```

Figure 6. Sample Radio Description File

3.6.2. Module Generation and System Assembly

Hardware modules required for the generation of the radio will be either originate from a library of implementation independent modules or they will be automatically generated. The library will contain all modules necessary to support the available transform and their variations or instructions on how to automatically generate them. A module description file, attached to each module describes the specifics of the modules implementation. The file also indicates the modules interface type and how the interface maps to its ports. A given interface can be mapped to one or more ports of varying directions. The input interface for Coregen generated filters, for example, is mapped to three ports: the dataIn bus (input), the NewData signal (input) and the ReadyForData signal (output).

Implementation dependent modules, such as FIR filters used during down-conversion will be generated at implementation time using vendor tools such as Xilinx's Coregen. All generated modules will be then added to the library for use in future implementations. When a suitable implementation has been found for all required modules, a receiver module is assembled by the implementation engine.

Because all transforms are mapped to self-contained modules, system assembly mainly consists of module instantiations and communication synthesis. Instantiating modules is a trivial task involving basic string processing.

Communication synthesis on the other hand is not as trivial. To enable the automatic creation of connections between two interfaces programmatically a set of interconnection rules must be defined. These rules define how to create connections between the ports of two interfaces. For any given interface type a rule must exist defining how to connect it to other interfaces types. The interface definitions and the interconnections rules will be stored in an XML file based on the technologies presented in [7] and [8].

Because of the varying styles of communications used by interfaces, interconnecting two interfaces may require more than just creating wires between ports. For example, let's consider connecting a block-based interface with a stream-based interface. Upon receiving the ready for data signal a block-based interface will send *N* data words to the receiver, where *N* is a known constant. In a stream-based interface the sender will only send one data word per ready for data signal. To create a connection between these two interfaces, a bridge module with a FIFO and some control logic needs to be inserted. Any glue logic needed, such as the FIFO mentioned above will be integrated into the interconnection rules.

The FPGA configuration bitstream is the created from the top-level system provided in the HDF and the radio module and objects provided by the implementation engine. A vendor specific makefile is used to invoke vendor tools. Figure 7 shows the hardware generation process.

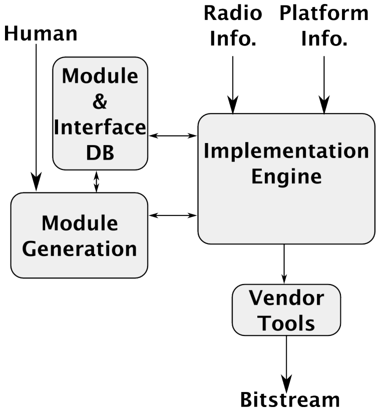


Figure 7. Hardware generation flow

4. CONCLUSION

A Rapid Radio framework based on a guided process for radio signal analysis was presented. The framework allows the user to apply transforms to the signal and is presented with the results. In accepting or rejecting the transform, the user is defining the architecture of the radio. After all transforms have been applied, a Radio Description File is created detailing the architecture of the receiver.

Information extracted about the unknown signal is used with the Hardware Description File to build an FPGA-based radio receiver. A communication synthesis scheme is used to connect modules of different interfaces. The design of both the radio and hardware description files must be such that they avoid dependence on specific vendors.

Hardware efficiency is traded off with implementation time in order to allow a fast radio creation over a general purpose hardware platform. This permits the use of resource intensive blocks, such as the Kalman filters and the 4-sample Gardner synchronizer, which provide wider operating ranges.

Providing high-level transforms to aid in classifying a signal eliminates the need for detailed communications knowledge by the user. The combination of pre-built modules, automatic module generation and communication synthesis reduces the burden of HDL development and in-depth knowledge of FPGAs. Support for other modulation families can be added by implementing new transforms.

5. REFERENCES

[1] O. A. Dobre, A. Abdi, Y. Bar-Ness, and W. Su, "Survey of automatic modulation classification techniques: classical approaches and new trends," *Communications, IET,* vol. 1, pp. 137-156, 2007.

[2] 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 in Cognitive Radio," in *New Frontiers in Dynamic Spectrum Access Networks, 2007. DySPAN 2007. 2nd IEEE International Symposium on*, 2007, pp. 212-215.

[3] B. Le, T. W. Rondeau, D. Maldonado, D. Scaperoth, and C. W. Bostian, "Signal Recognition for Cognitive Radios," in *Software Defined Radio Forum Technical Conference* Orlando, FL, 2006.

[4] B. Le, T. W. Rondeau, and C. W. Bostian, "General Radio Interface Between Cognitive Algorithms and Reconfigurable Radio Platforms," in *Software Defined Radio Forum Technical Conference* Denver, CO, 2007.

[5] J. Nocedal and S. J. Wright, *Numerical optimization*, 2nd ed. New York: Springer, 2006.

[6] F. Gardner, "A BPSK/QPSK Timing-Error Detector for Sampled Receivers," *Communications, IEEE Transactions on [legacy, pre - 1988],* vol. 34, pp. 423-429, 1986.

[7] V. Berman, "Standards: The P1685 IP-XACT IP Metadata Standard," *Design & Test of Computers, IEEE,* vol. 23, pp. 316-317, 2006.

[8] R. Bergamaschi, W. R. Lee, D. Richardson, S. Bhattacharya, M. Muhlada, R. Wagner, A. Weiner, and F. White, "Coral-automating the design of systems-on-chip using cores," in *Custom Integrated Circuits Conference, 2000. CICC. Proceedings of the IEEE 2000*, 2000, pp. 109-112.