Matlab as a Development Environment for FPGA Design

Tejas M. Bhatt
Nokia Research Center
6000, Connection Drive
Irving, TX - 75039
+1-972-894-4733
tejas.bhatt@nokia.com

Dennis McCain
Nokia Research Center
6000, Connection Drive
Irving, TX - 75039
+1-972-894-5917
dennis.mccain@nokia.com

ABSTRACT

In this paper we discuss an efficient design flow from Matlab® to FPGA. Employing Matlab for algorithm research and as system level language allows efficient transition from algorithm development to implementation. We show that integrating Matlab with HDL design tools such as HDL designer® and Precision-C®, an efficient design flow, suitable for rapid prototyping, can be obtained. The design flow accelerates process of algorithm development and simplifies test-bench formulation and verification process. The overall development time thus can be significantly reduced. We elaborate on the advantages and disadvantages of the design flow. It will be shown that Matlab based design flow generates functional specifications that are useful for RTL development.

Categories and Subject Descriptors

[**Special Sessions**]: *System Level Design and Verification, MATLAB TM – The Other Emerging System-Design Language*

General Terms: Design, Verification.

Keywords: System Design Flows, Rapid Prototyping.

1. INTRODUCTION

The quest of higher data rates has lead to significantly complex physical layer (PHY). In order to push more bits/sec/Hz over a hostile wireless environment; a variety of schemes such as MIMO, advanced signal processing, advanced forward error correcting (FEC) schemes have been proposed. Many standards, such as WCDMA(HSDPA), CDMA2000(1xEV-DO), Wi-MAX, have already included these schemes. A numerous computationally complex algorithms have been proposed to improve the performance of a communications receiver. Although many signal-processing algorithms have been proposed, not all are suitable for the real time implementation and many have prohibitively high implementation complexity. Further, if implemented for mobile devices, these algorithms must be efficient in resource utilization, area usage and power consumption.

Rapidly evolving wireless standards have emphasized the need of the rapid prototyping. It is necessary to evaluate real-time implementation complexity, efficient architectures and required hardware resources. A proof-of-concept of such algorithms or system is useful to demonstrate and validate the gains in the real world. Further, rapid prototypes are helpful in assessing various implementation strategies and identifying implementation bottlenecks. Moreover, an implementation specifications can be obtained from the functional specification and thus aiding the process of technology transfer and product development. Since, variety of promising transceiver options and algorithms are possible, a flexible, simple and fast design is essential for the proof-of-concept.

Field Programmable Gate Arrays (FPGAs) provide reconfigurability and parallel processing capability, making them ideal for prototyping of computationally intensive PHY algorithms. FPGAs allow to experiment and evaluate the best implementation architecture. Due to their re-programmability, the rigorous verification and testing procedures required at various stages of hardware development can be relaxed. Further, a reasonably good estimate of the required hardware resources can be obtained by from the number of FPGA gates in design. Thus, different algorithms and various implementation strategies can be measured in the form of hardware costs.

One key task for the hardware implementation, be it for product or for proof-of-concept, is to develop a fixed-point architecture for the different algorithms. In order to limit the hardware resources, most communications algorithms are implemented with fixed word-length accuracy. The performance of the algorithms must be analyzed in the fixed-point environment. Moreover, many algorithms are not implementable in their original form and require modifications. The performance of the modified algorithm must be tested and calibrated against the original. During the initial phase of technology, the process of candidate algorithm selection and evaluation may require several iterations. Thus, a common platform, which seamlessly integrates the algorithm development and implementation process, is required. In other words, a compact design flow is required that encompasses the algorithm research, technology white-papers, algorithm specifications and hardware implementation.

Matlab has become the preferred language of computing for the researchers. In recent years, Matlab has evolved from research tool to a system developers' language. In this paper, we will analyze Matlab as a development environment for the FPGA implementation. A simple design flow encompassing Matlab as a system level language is described in section II. The design flow utilizes Mentor Graphics' Precision-C® and HDL-Designer® for the purpose of rapid prototyping. In section III, we will show the application of the design flow on High-Speed Downlink Packet Access (HSDPA) system. We will show that the proposed design flow cuts the time needed to translate the functional specification in

DAC 2005, June 13–17, 2005, Anaheim, California, USA.

to implementation specifications. The advantages of the proposed design flows are discussed. The bottlenecks in the design flows are also identified.

2. MATLAB BASED DESIGN FLOW

In recent years Matlab has successfully bridged the gap between the system developer and researcher. With both, system developer and researcher, working on the same platform, the required time for translating a research paper or technical specification into the functional specifications can be significantly reduced. Previously, fixed-point analysis within Matlab platform was manual and required pain-staking efforts. This typically involved converting generic Matlab code to a flat m-code. Different variables must be checked and verified for the required word-length. Alternatively, the m-code can be converted to a generic C/C++ fixed-point code. The C-code can then be integrated in Matlab simulation platform and verified. However, various tools are now available to support the fixed-point functionalities [1, 2]. Although manual, translation of an m-code from floating-point to fixed-point data-types, requires simple modifications. One advantage is that different implementations have the same interface making it easier for both developer and researcher. Any modifications in the algorithms can be easily reflected in the fixed-point architecture evaluation. The other advantage is that the resultant fixed-point Matlab-code or C-code code can be considered as an implementation specification for the hardware or Digital Signal Processor (DSP) implementation.

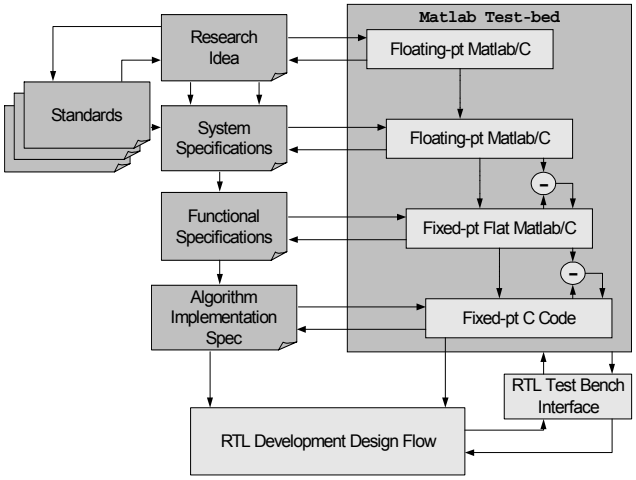

Figure 1. Matlab As a System/Algorithm Evaluation Tool

Figure 1 relates different stages of algorithm development to a Matlab-based design flow. Matlab can be effectively utilized as a test-bed to evaluate the algorithm or system performance, at the different stages of the algorithm development. C-codes can be integrated in Matlab environment using MEX utilities and in figure 1 it is considered as an integral part of Matlab test-bed.

Regardless the target platform, many design flows require a generic fixed-point C/C++ realization of the algorithm as a staring point for hardware implementation of baseband communications algorithms. Here, we consider a design flow suitable for rapid prototyping that targets the FPGAs. The most common method of creating hardware design for an FPGA is by using conventional hardware description language (HDL) such as VHDL or Verilog. Typical HDL design method involves hand-written HDL or graphical design tools such as Hardware Design System (HDS) from Cadence and HDL Designer from Mentor Graphics. Graphical design tools capture schematics and automate the HDL generation and are in general, easier to use. However, the schematic layout is manual and requires pre-conceived hardware architecture. For complex algorithms, comparison of various architectures and implementation strategies, evaluation of trade-offs between area and timing become difficult and time consuming. Recently, tools such as Precision-C from Mentor Graphics have tried to address this problem using a C/C++ level scheduling. The tool incorporates most of the design styles of generic C/C++ codes and generates the HDL. The tool is very efficient for complex algorithm implementation that requires multiple functional units. It is rather straightforward to incorporate system level and instruction level pipelines [3]. Further, it requires only a small alteration to generic C/C++ code. However, in case of complex algorithm, the code must be appropriately segmented before it can be used within Precision C design flow. But, it is still more convenient to operate at C-level abstraction. Further, since it is a generic C/C++ code, it can be simulated with Matlab test-bed, thus verifying the code. Hence, a tighter link between Matlab and HDL implementation is obtained.

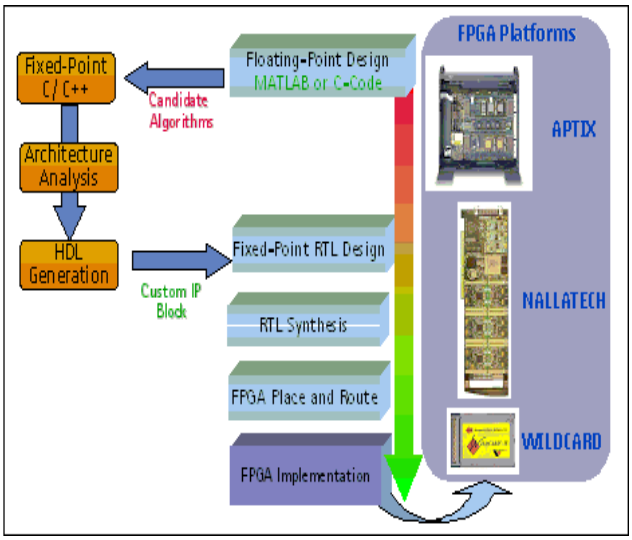

Figure 2. Matlab-Based RTL Design Flow for FPGA

Figure 2 depicts RTL design flow that uses Matlab at the top level. The proposed design flow integrates Matlab tightly at the different stages of RTL development and hardware implementation. Matlab is used as a test-bench for fixed-point generic C/C++ models. Moreover, Matlab generated test-vectors are also used for verification of VHDL implementation. VHDL implementation can be manual or an automated procedure based on tools such as, HDL Designer or Precision-C.

3. CASE STUDY: HSDPA SYSTEM

3.1 HSDPA System Overview

We implemented the Matlab-based design methodology for HSDPA system prototyping. HSDPA is an evolutionary mode of WCDMA system that offers high-speed data access over the 3G networks [4]. HSDPA enhances link performance and system capacity through

features such as link-adaptation and fast H-ARQ. Figure 3 represents the block diagram of the high-speed shared data channel (HS-DSCH) that carries packet data. The channel is time-multiplexed and code-shared between different users. It should be noted that synchronization block in the receiver implements time acquisition, carrier frequency acquisition and clock tracking mechanism. We have omitted control flow details for the ease of understanding.

3.2 HSDPA Algorithm Development

The system implementation in Matlab evolved with the standards. Various algorithms were implemented in Matlab or C. The C-code were integrated in simulation chain using C-mex utility. Further, various algorithms were evaluated (e.g. Rake, equalizer, channel estimator) with different parameters. Next, the algorithms were converted to flat m-code and then to fixed-point code. The fixed-point translation was manual. The time required for the fixed-point translation and performance evaluation depends greatly on the complexity of algorithms, operations that are involved as well as the personal skill. For algorithms such as synchronization or channel estimation, which typically involve Multiply-Accumulate (MAC) operations, the fixed-point translation is rather straightforward. On the other hand, chip equalizer is a computationally intensive algorithm and typically involves matrix inversion. It requires thorough fixed-point analysis and the time spent in the fixed-point conversion can be significantly high.

The resultant fixed-point code is used as a template for the hardware implementation. Graphical design tools such as HDL Designer provides ease in capturing the large design. Individual modules can be hand-coded in the VHDL. However, as shown in [1], some blocks (e.g. equalizer) are considerably complex. For example, equalizer block consists of equalizer tap-solver and filtering, where the tap-solver involves matrix inversion or FFT [5].

As suggested in [3], Precision-C can be employed to implement complex algorithms that need rapid scheduling. Precision C allows flexibility in architecture evaluation and the design trade-offs at C-level abstraction. It can significantly reduce the time for RTL development and makes overall design flow adaptable. However, it requires conversion of fixed-point m-code to a fixed-point C-code.

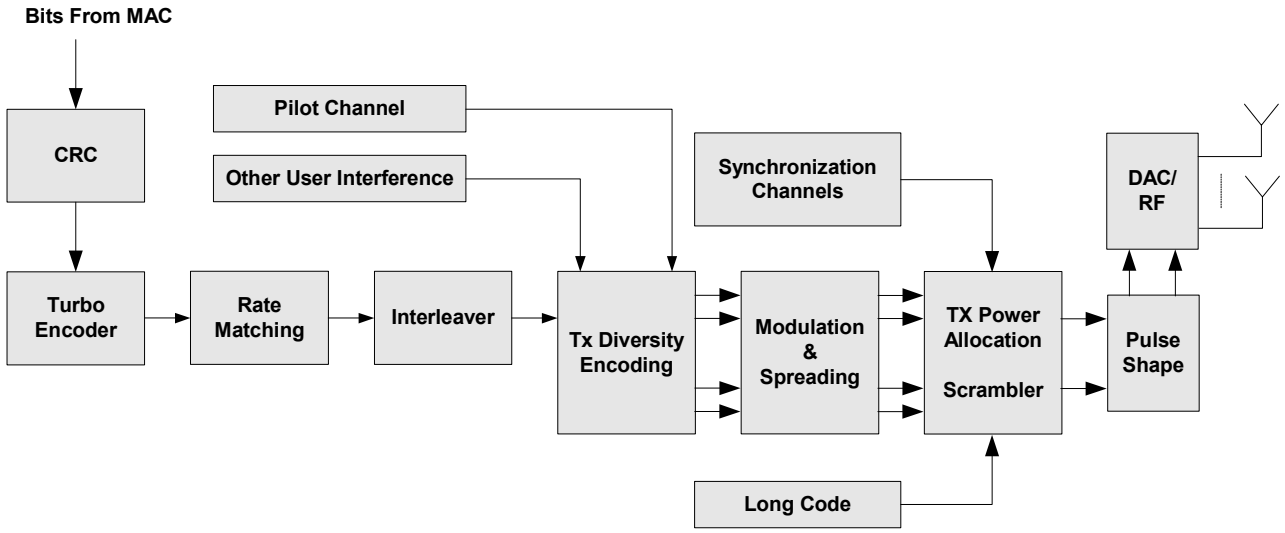

Figure 3a – HSDPA Transmission Chain

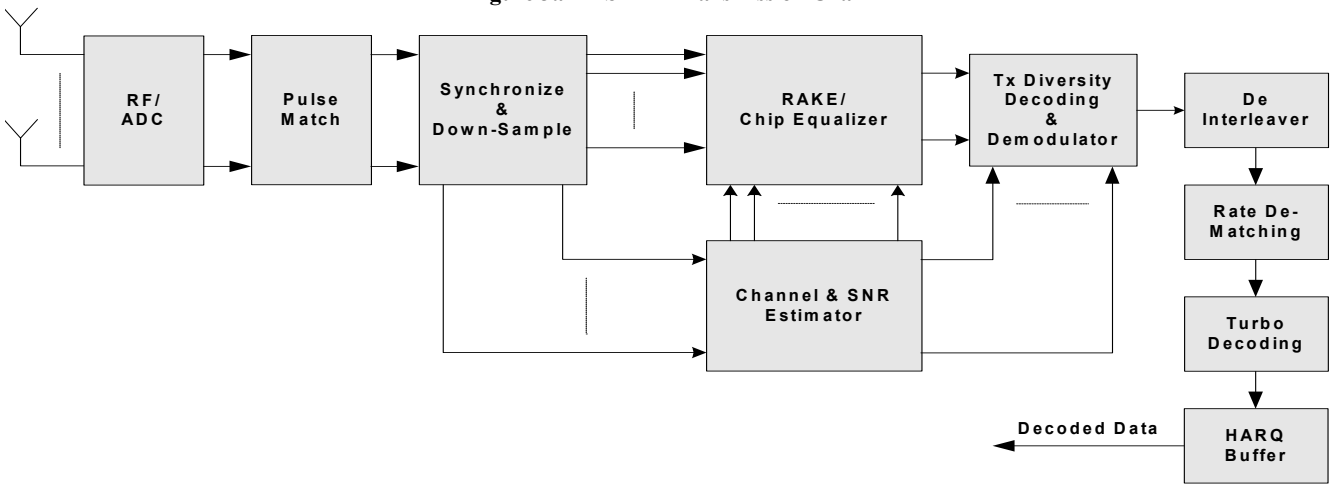

Figure 3b – HSDPA Receiver Chain

One of the most important steps in RTL development is functional testing and verification. The RTL design must be verified with the Matlab design using the test-vectors generated by Matlab test-bed. We used a data file interface to transfer the test-vectors and output data from and to Matlab test-bed. The RTL design is simulated using ModelSim®. This scheme is rather simple and detached from main design flow. It is laborious, especially when a lot of different test cases are to be simulated. Some design tools, such as System-C provides capability to perform software/hardware co-simulation. This can be a great asset in reducing the overall development time. Recently, it has become possible to co-simulate RTL design with Matlab test-bed, utilizing Matlab to ModelSim link [6].

3.3 Advantages and Disadvantages

Since, Matlab is a preferred tool for researchers, integrating Matlab in the RTL design flow allowed faster transition from algorithm research to implementation. The overall design time from initial Matlab floating-point implementation to hardware implementation can be reduced significantly. Integrating Precision-C in the Matlab-based design flow, the hardware development process is accelerated. The design flow is particularly advantageous for proof-of-concept applications, because it bridges the gap between the algorithm research and algorithm development. Further, by providing a C-level abstraction in architecture evaluation, a tighter link can be obtained between original Matlab algorithm and the implemented RTL design.

One major bottleneck in the design flow is transition from floating-point Matlab to fixed-point Matlab. In order to obtain fixed-point Matlab-code, sometimes the vector operations must be avoided. Further, in absent of fixed-point data-types additional coding is required to avoid overflow and guarantee fixed-point accuracy. This often results in a reduced simulation speed. The other critical task in the proposed design flow is to translate the fixed-point m-code to a fixed-point C-code.

4. CONCLUSION

An efficient design flow centered on Matlab is illustrated for the rapid prototyping. The proposed design flow is well suited for FPGA implementation and reduces the time for algorithm development. Integrating Matlab and Precision-C in the same design flow makes it flexible and yields quick transition to VHDL.

5. ACKNOWLEDGMENTS

We would like to thank our colleague Yuanbin Guo for his valued comments on the RTL design flows.

6. REFERENCES

[1] http://www.mathworks.com/products/fixed/

[2] http://www.catalyticinc.com/deltafx.html

[3] Guo, Y., Xu, G., McCain, D and Cavallaro, J. R. Rapid *Scheduling of Efficient FPGA Architectures for Next-Generation HSDPA Wireless System Using Precision C Synthesizer*. IEEE International Workshop on Rapid Systems Prototyping, pp. 179-185, San Diego, CA, (June 2003).

[4] 3GPP, Technical Specification, 25.211, v5.5.0, 2003-09.

[5] Zhang, J. Bhatt, T. and Mandyam, G., *Efficient Linear Equalization for High Data Rate Downlink CDMA Signaling*. 37th IEEE Asilomar Conference on Signals, Systems, and Computers, pp. 141 - 145, vol. 1, Monterey, CA, (Nov. 2003).

[6] http://www.mathworks.com/products/modelsim/