EFFICIENT FPGA-BASED RADIO DESIGN USING MATLAB ENTRY IN HDL CODER AND STANDARDS-BASED HARDWARE

Richard E. Cagley, Scott A. McNally, and Jason Hannon *

Toyon Research Corporation
Goleta, CA 93117
Email: rcagley@toyon.com

ABSTRACT

The primary contribution of this paper is to present a set of basic wireless transceiver signal processing algorithms and compare their implementation performance using hand-coded Verilog, AccelDSP, and MATLAB entry in HDL Coder. One of the major criticisms of AccelDSP was the efficiency of generated Hardware Description Language (HDL) code. With the introduction of Mathworks MATLAB entry in HDL Coder we seek to compare its performance among established, well-understood algorithms. In the paper we also discuss several practical issues that MATLAB entry in HDL Coder addresses that have been lacking in previous tools. These features ensure that the tool generates HDL that synthesizes efficiently, is readable, and usable in real-world hardware designs. This paper also seeks to illustrate and promote high-level synthesis (HLS) as a development methodology for both algorithm developers as well as hardware engineers. With the ability to automatically generate floating, fixed-point, and HDL testbenches, the use of MATLAB entry in HDL Coder can dramatically reduce development time and provide significantly more robust testbenches, thus catching and fixing bugs much earlier in the development process. Seeking to increase the accessibility of radio development for academic institutions and small design groups, we present a FPGA Mezzanine Card (FMC) low-cost radio. When combined with commercial-off-the-shelf (COTS) FPGA boards with FMC connectors and developing MATLAB entry in HDL Coder wireless transceiver cores, a small team can quickly showcase advanced wireless communication algorithms.

1. INTRODUCTION

Field Programmable Gate Arrays (FPGAs) offer the highest computational performance of any embedded processing unit while still allowing for reconfiguration. For this reason they are the embedded resource of choice for a wide range of high data throughput applications. In particular, they play very important roles in the operation of wireless digital communication systems. Typical Software Defined Radio (SDR) implementations include filtering and frequency translation. Hardware Description Language (HDL) entry tools handle these types of applications quite reasonably. Examples include classical Verilog and VHDL programming as well as block-based flows, such as System Generator, DSP Builder, and Simulink HDL Coder. However, more complex portions of a radio waveform are traditionally performed in digital signal processors (DSP) and general-purpose processors (GPPs). There is a strong motivation in moving more computation from these forms of sequential processing into FPGAs.

For more than a decade there has been interest in behavioral language entry and translation to HDL for FPGAs. This has culminated in a wide array of electronic design automation (EDA) tools that can translate high level languages, such as C/C++ and their variants, to synthesizable HDL. Largely driven by the EDA industry rather than algorithm and system-level developers, these tools are expensive, difficult to use, and generally require expert knowledge. The desire to employ high level synthesis (HLS) for FPGA language entry has existed since the invention of HDL language entry. Indeed, the field of EDA is littered with failed attempts to create an HLS tool that provides several key features.

- Syntactically-rich language
- High-quality HDL code generation
- Reasonable cost
- Straightforward to use

These oftentimes conflicting needs are epitomized by one tool in particular, AccelDSP. AccelDSP was a MATLAB to HDL HLS product by AccelChip and then Xilinx. AccelDSP was originally termed the MATCH compiler and had its genesis in the DARPA Adaptive Computing Systems program [1]. With its introduction over a decade ago, AccelDSP offered an elegant solution for translating behav-

*The authors would like to thank Dr. Bruce Fette. This work was supported by DARPA contract D11PC20101. The views expressed are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

ioral MATLAB to HDL. By having top-level language entry being MATLAB, AccelDSP offered a state-of-the-art, industry-standard simulation environment along with a contextually-rich language for function entry. While a useful tool, AccelDSP suffered from the fact that while algorithm designers are comfortable with MATLAB, hardware engineers are typically more comfortable with classical HDL and more trusting of hand-written code.

The demise of AccelDSP presents a major challenge as the tools that are currently available for FPGA language entry are poorly understood and of limited utility to communication system engineers. Furthermore, those well versed in FPGA language entry often have limited mathematical skills and simulation experience with communication system algorithms. These challenges are being addressed with the release of HDL Coder [2]. By being able to directly translate MATLAB code to HDL, there is no longer a division line between algorithm development and hardware engineering, as illustrated in Figure 1.

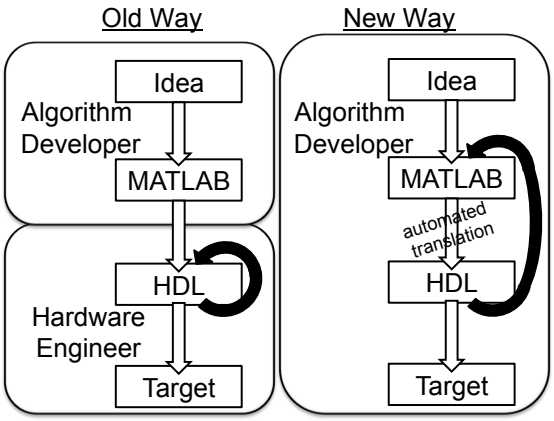

Fig. 1. The use of MATLAB HLS removes the strong seperation between algorithm developers and hardware engineers.

2. DEVELOPMENT OVERVIEW

MATLAB is an excellent HLS language for implementation of advanced signal processing algorithms in FPGAs. This is largely due to the fact that MATLAB is the de-facto standard for initial implementation of advanced signal processing algorithms. One of the main reasons for its popularity is its unparalleled testbench capabilities. This means that very complex stimulus can be generated and processed for fixed-point and RTL simulation. Finally, when algorithm designers have more control over the destiny of their ideas, FPGA designs can be created more efficiently and more of them can get done.

In Figure 2 we illustrate a flow used for IP core development and integration. Generally, our approach has been

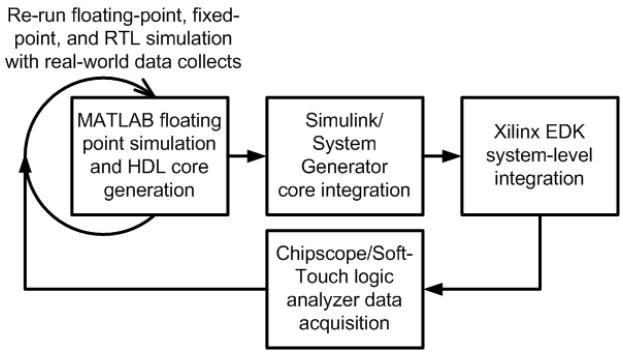

Fig. 2. Development flow using a mixture of core generation and integration tools.

that MATLAB entry in HDL Coder is an excellent tool for IP core generation. With its graphical interface, Simulink is well-suited to quickly connect cores and visualize the overall system flow. Furthermore, with their availability of in-circuit debugging via such tools as SignalTap and Chipscope, Simulink packages such as Xilinx System Generator and Altera DSPBuilder make logic analyzer insertion very straightforward. Additionally, these Simulink-based tools provide bus interface support for FPGA-based hard and soft processors, which dramatically reduces hardware-software integration times. The algorithm developer can also leverage commonly used IP cores, such as filters, that take advantage of hardware features specific to the target device. As a final integration step, we employ system-level integration tools, such as Xilinx Embedded Development Kit (EDK) or Altera Nios II. With an overall focus on test and simulation, the overall flow supports feedback of real-world collected data back into the simulation model.

3. RECEIVER PROCESSING CORES

In this section we first present the signal model and then the means for estimating start-of-packet for synchronization as well as the means to determine time and frequency offset. The packet structure for the waveform is shown in Figure 3. As can be see from the figure, we have the ability to accommodate arbitrary packet lengths and there is also a CRC for packet validity verification.

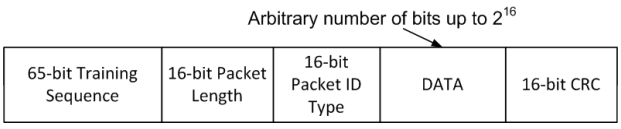

Fig. 3. Packet structure.

3.1. Signal Model

The receive signal $r_c(t) \in \mathbf{C}$ is given in continuous-time by

$$r_c(t) = \sum_{m=0}^{\infty} h(m)p(t-\tau-mT)e^{i\delta\omega t}s(m) + v(t), \quad (1)$$

where $h \in \mathbf{C}$ is the channel from the Tx antenna, T is the symbol period, ω is the frequency offset in Hz, and τ is the corresponding delay [3]. The term $v(t)$ represents additive white Gaussian noise with variance σ^2. The symbols $s(m)$ represent a training sequence in the estimation phase. The pulses $p(t)$ are raised-cosine with minimal excess bandwidth to maximize orthogonality. The total transmitted bandpass energy is E_s per information symbol.

This receive signal can be converted to a discrete-time signal by sampling at an oversampling rate with period T_{os}. Assuming N_D samples per symbol period and sampling period n, the resulting symbol index l is related as $N_D l \equiv n$. The resulting discrete time received signal is

$$r_c(n) = \sum_{m=0}^{\infty} h(m)q(n-\eta-mN_D)e^{i\delta\Omega n}s(m)+v(n) \quad (2)$$

where $\eta \equiv \tau/T_{os}$, $\delta\Omega \equiv \delta\omega T_{os}$, and $q(n) = p(nT_{os})$ is the discrete-time sampled pulse shape of $p(t)$.

3.2. Correlator for Packet Synchronization

As we consider a packet-based system, a parameter that must be estimated is the start of the packet. While a simple correlator is an appealing technique, the presence of a potentially large carrier offset precludes its use for all but the shortest training sequences. Striving for a simple hardware implementation, our approach centers around the homodyne detector.

In forming the detector, consider a new set of symbols formed by the conjugate pairs of received samples. The individual terms are given by

$$c(n) = r(n)r^*(n-N_D). \quad (3)$$

We note that if sampled at the symbol rate, and aligned with the peak of the pulse shape $q(0)$, the resulting received signal can be simplified to

$$r(n) = e^{i\delta\Omega n}s(n-\eta) + v(n). \quad (4)$$

If proper timing alignment is achieved, the conjugate pair in (3) can be simplified to

$$c(n) = e^{i\delta\Omega N_D}d(n) + \bar{v}(n) \quad (5)$$

where $d(n) = s(\lfloor (n-\eta)/N_D \rfloor)s^*(\lfloor (n-\eta-1)/N_D \rfloor)$ are the conjugate pairs of received symbols and $\bar{v}(n) = v(n)v^*(n-N_D)$ is the product of two independent circular Gaussians. We can collect a set of these homodyne symbols corresponding to one less than the length of the training sequence $\mathbf{c}(n) = [c(n), c(n-1), \ldots, c(n-N_{ts}+2)]'$ with N_{ts} being the training sequence length. We see that if the symbols $d(n)$ were known then a correlation would result and associated carrier offset formed. For this purpose we create a set of conjugate training symbol pairs $\mathbf{u} = [u_{N_{ts}-1}, \ldots, u_2, u_1]^T$ where $u_i \equiv \tilde{s}_{i+1}\tilde{s}_i^*$ and $\{\tilde{s}_i\}$ are a known set of training symbols. Next, a correlation can be formed

$$z(n) = \mathbf{u}^H\mathbf{c}(n). \quad (6)$$

We note there are two sample times of interest. The first is when the correlation symbols and the training period of the transmitted packet are not aligned. In this case, the conjugate sample and symbol pairs will have low correlation and $z(n)$ will result in a low value. However, if the proper offset is encountered the following signal will result

$$z(\eta) = (N_{ts}-1)E_s e^{i\delta\Omega N_D} + \sum_{m=1}^{N_{ts}-1} u(m)v(m)v^*(m-N_D) \quad (7)$$

with E_s being the training symbol energy and η the maximum correlation offset. We see

$$\bar{z}(\eta) \equiv E[z(\eta)] = (N_{ts}-1)E_s e^{i\delta\Omega N_D} \quad (8)$$

providing an unbiased estimate of the carrier offset. In addition, we can observe that

$$\sigma_{z(\eta)} \equiv E[(z(\eta)-\bar{z}(\eta))^2] = (N_s-1)\sigma^4 \quad (9)$$

and the resulting signal to noise ratio for the estimate of the carrier offset is

$$\mathrm{SNR} = \frac{(N_{ts}-1)E_b^2}{(\sigma^2)^2} \quad (10)$$

which we note scales linearly with increasing numbers of training symbols.

3.3. Early-Late Gate Timing Recovery

We consider timing recovery via output power maximization [4]. The estimate for η is recursively estimated using the equation

$$\eta(n+1) = \eta(n) + \mu_t r_c(n)\left[r_c(n+\eta(n)+1) - r_c(n+\eta(n)-1)\right] \quad (11)$$

where μ_t is the step size, nominally set to 1.0 in our applications. From a MATLAB core implementation perspective, the core is interesting in that it provides a rate translation.

Table 1. Resource use of wireless receiver cores for Xilinx Spartan-6LX FPGA.

	Slice Registers	Slice LUTs	BRAM	DSP48Es	Performance (MHz)	Rate
HDL Coder - Correlator	584	2144	0	4	54.2	1
AccelDSP - Correlator	476	1170	0	3	24.9	1
Hand coding - Correlator	342	1091	0	4	29.1	1
HDL Coder - Timing Recovery	562	602	0	4	38.9	1
AccelDSP - Timing Recovery	383	848	0	3	32.6	2
Hand coding - Timing Recovery	372	453	0	4	43.0	2
HDL Coder - Costas Loop	16	896	0	7	25.7	1
AccelDSP - Costas Loop	98	624	0	11	17.4	5
Hand coding - Costas Loop	219	1416	0	5	56.6	3

That is, the input rate is at the sampling rate of any front-end filtering in the application. The output rate is the symbol rate of the amplification. In our instantiations of the core we allow the core to operate at the symbol rate, passing a vector of samples equal to the oversampling rate into the core.

3.4. Costas Loop Carrier-Offset Estimation

There are a variety of methods for estimating carrier frequency offset range from the Fast-Fourier Transform (FFT) to decision-directed phase tracking [4]. Here we consider a hard-limited Costas-loop based implementation [5]. The carrier compensated output signal is given by

$$z(n) = r_c(n)e^{j2\pi\phi(n)} \tag{12}$$

where

$$\phi(n+1) = \phi(n) + \mu_f e(n) \tag{13}$$

and

$$e(n) = z_r \text{sign}(z_i) - z_i \text{sign}(z_r) \tag{14}$$

with $z_r = \text{real}(z(n))$ and $z_i = \text{imag}(z(n))$.

The core is interesting from an HDL generation perspective for several reasons. Along with the basic mathematical operations, there is the additional aspect of those being complex numbers. Both AccelDSP and MATLAB entry in HDL Coder handle this well, whereas introduction of complex numbers in HDL adds significant complexity. Additionally, we note from a design choice perspective there are different methods by which mathematical functions can be implemented. From an HLS perspective the algorithm developer can quickly select from different translation options, e.g., CORDIC, lookup-tables, or other methods depending on the mathematical function. In our case, we chose to use linear lookup tables (LUTs).

4. PERFORMANCE OF MATLAB ENTRY IN HDL CODER

One of the major challenges of HLS adoption has been the complaint by hardware engineers that HLS does not produce sufficiently readable or efficient code. In order to determine whether this is actually the case for the new MATLAB entry in HDL Coder tool we have identified a set of basic signal processing cores for wireless transmission and compared implementation performance using three different approaches. The first is MATLAB entry in HDL Coder. The second is the classic MATLAB to HDL translation offered by AccelDSP. Finally, as a baseline we have hand-coded each core.

These results are presented in Table 1. As we see from the table, MATLAB entry in HDL Coder offers significant gains in implementation efficiency over AccelDSP and compares well to hand coding. What is not shown in the table and that we speak to qualitatively is the readability of MATLAB HDL Coder generated HDL. In this regard, in addition to general readability, a novel feature is the inlining of MATLAB code and comments directly into the HDL as comments. We also note that with synchronous clock-enabled based or oversampling interfaces it is straightforward to integrated the generated HDL code into a broader design.

5. FMC-BASED RF FRONT-END

Over the last decade there has been tremendous interest in SDR technology [6, 7]. This has included novel work on RF filtering, mixing, and general circuit design in order to promote flexibility at the analog front end for such functions as frequency translation. Work has also focused on baseband signal processing to promote adaptation and minimization of computational complexity for support of different waveforms on the same platform [8]. Finally, tremendous advances in CMOS technologies and particularly FPGAs allow developers to design and deploy novel waveforms in timeframes and at cost levels that are dramatically less than a decade ago. However, for all its promise, there are relatively few examples of SDRs that have been demonstrated by research groups and fielded by commercial or government organizations.

Certain research groups, such as Minden et. al, have attempted to rectify this problem by designing their own RF front ends and using commodity single-board PC computers [9]. However, with a focus on research and design the group is not well suited to manufacture and distribute the design such that it can be readily procured by other academic, government, and industry organizations. Finally, companies such as Spectrum Digital, Lyrtech, CommBlock, and Texas Instruments have designed and are making available SDR platforms. However, these systems are overly complex and costly for small to moderate final system platforms. An example of a commercially available product is the Small Form Factor SDR Development Platform available from Texas Instruments. While offering a highly integrated development solution, its high cost of $10k per unit make it inaccessible for budget-conscious research groups. Additionally, a lack of standard expansion and modularity do not make it well suited to transition to final applications.

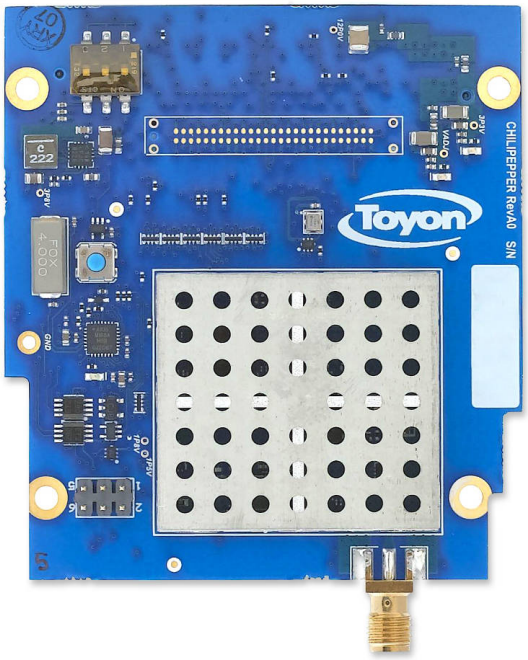

Fig. 4. FMC based RF front-end.

Shown in Figure 4, Toyon has developed a complete RF front-end solution based on the VITA 57 FPGA Mezzanine Card (FMC) standard [10, 11]. With its integrated onboard microcontroller the RF front-end can be configured to operate in several built-in modes for ease of use with first designs or can be fully customized by more advanced users. In its most basic form, the user only need supply parallel digital signaling to the digital-to-analog converter (DAC) and read in parallel data from the analog-to-digital converter (ADC). Time duplex transmit/receive switching is accomplished with a simple GPIO pin from the FPGA.

The board has many features making is well-suited to prototyping

- Centered on a highly integrated Lime Microsystems radio frequency integrated circuit (RF-IC) with onboard microcontroller for configuration [12]
- Multi-band operation with support for full-duplex frequency-division duplex (FDD), half-duplex time-division duplex (TDD), and broadband reception
- Compatible with any FPGA development board that hosts an LPC/HPC FMC connector
- Standard SMA antenna connector

Toyon will also release MATLAB entry in HDL Coder-based IP cores that can serve as a basis for researchers' and developers' own designs or as examples. Demonstration and tutorials designs are also being developed.

6. CONCLUSION

With the increasing logic density of FPGAs there is the potential to implement sophisticated, state-of-the-art wireless communication algorithms. While such algorithms have traditionally required many years of research and development with large hardware design teams, the release of MATLAB entry in HDL Coder holds the promise to dramatically reduce development time and group size. The hope is that there can be an increasing number of radio designs that showcase a wider breadth of innovation than is currently the case. HLS should enable small businesses and universities to implement practical, real-world designs, and larger groups to do so more quickly.

In this paper we have sought to address a major challenge for the use of HLS tools in that there are relatively few data points that compare the performance of HLS across different HLS tools and to hand coding. As we have seen in the paper, MATLAB entry in HDL Coder offers significant performance gains over AccelDSP and compares well to hand coding. This is in addition to its general utility in terms of such time-saving structural development features as automated testbench generation and rapid algorithm re-design with feedback from synthesis.

In addition to the tool itself, we have presented an RF front-end that can work with any FPGA development or production board with an FMC connector. The introduction of the board addresses a need in the wireless communications development community that there is a lack of commodity hardware tools for prototyping. As illustrated by the popularity of the Universal Software Radio Platform (USRP) there is strong interest in the ability to demonstrate over-the-air (OTA) solutions [13]. However, while the USRP is focused on processor based solutions, our RF front-end can lead to practical, cost-effective, and low-power products.

While we feel the presented flow represents an efficient, state-of-the-art methodology for translating algorithms to FPGA hardware, there are many areas left for improvement. As algoirthm developers often develop in floating point MATLAB there needs to be better techniques and visualization for fixed-point exploration. In particular, the ability to identify quantization loss as it flows through a multi-variable, multi-core system can assist the HDL Coder developer in how best to assign integer and fractional precision. Similarly, visualizaiton and feedback of portions of the code that lead to highest resource use and result in timing bottlenecks are very important. Another major area in need of development is in the area of system integration and code partitioning. With MathWorks C-code generation for embedded targets there is the potential to partition code among processor and FPGA fabricate in a seamless fashion. This is particularly the case as we look towards Altera/Xilinx standardization of their onboard processor interfaces to AMBA.

7. REFERENCES

[1] P. Banerjee, "An overview of a compiler for mapping MATLAB programs onto FPGAs," in *Proc. Design Automation Conference*, 2003, pp. 477 – 482.

[2] MathWorks, "HDL Coder," http://www.mathworks.com/products/hdl-coder/.

[3] J. Proakis, *Digital Communications*. New York, NY: McGraw-Hill, 1995.

[4] C. R. Johnson Jr. and W. A. Sethares, *Telecommunications Breakdown*. New Jersey: Prentice Hall, 2004.

[5] M. Simon, "Tracking performance of costas loops with hard-limited in-phase channel," *IEEE Transactions on Communications*, vol. 26, pp. 420 – 432, 1978.

[6] R. Bagheri, A. Mirzaei, M. E. Heidari, S. Chehrazi, M. Lee, M. Mikhemar, W. K. Tang, and A. A. Abidi, "Software-defined radio receiver: dream to reality," *IEEE Communications Magazine*, vol. 44, pp. 111–118, Aug. 2006.

[7] D. Efstathiou, L. Fridman, and Z. Zvonar, "Recent developments in enabling technologies for software defined radio," *IEEE Communications Magazine*, vol. 37, pp. 112–117, Aug. 1999.

[8] J. Melby, "JTRS and the evolution toward software-defined radio," in *Proc. IEEE Military Communications Conference*, vol. 2, Oct. 2002, pp. 1286–1290.

[9] G. J. Minden et. al, "Cognitive radios for dynamic spectrum access - an agile radio for wireless innovation," *IEEE Communications Magazine*, pp. 113–121, May 2007.

[10] VITA, "FPGA Mezzanine Card," http://www.vita.com/fmc.html.

[11] Toyon Research Corp., "Chilipepper - FMC Radio Board," http://www.toyon.com/chilipepper.asp.

[12] Lime Microsystems, "LMS6002D configurable broadband transceiver IC," http://www.limemicro.com/lms6002d.php.

[13] Ettus Research Inc., "Universal Software Radio Peripheral," http://www.ettus.com.