High-Level Hardware-Software Co-design of an 802.11a Transceiver System using Zynq SoC

Benjamin Drozdenko, Matthew Zimmermann, Tuan Dao, Miriam Leeser, and Kaushik Chowdhury
School of Electrical and Computer Engineering, Northeastern University, Boston, MA 02115
Email: bdrozdenko@coe.neu.edu, skulryk@gmail.com, dao.tua@husky.neu.edu, mel@coe.neu.edu, krc@ece.neu.edu

***Abstract*—Modern-day wireless communications standards are constantly evolving to meet the needs of an increasing number of devices. To adapt to these trends, software-defined radio has garnered more interest. In order to adapt to evolving standards while maintaining strict timing constraints, heterogeneous computing has been explored. In this demonstration, we take a new approach to the design of an 802.11a transceiver system on a heterogeneous system, the Zynq SoC. We take high-level Simulink models and develop several variants to enact different boundaries between components targeted for hardware and software. We then auto-generate C code from the software components and HDL code from the hardware components and use this to build both a CPU executable and an FPGA bitstream. We validate, profile, and analyze the models using metrics such as maximum step time per frame and FPGA resource utilization. Our results demonstrate how to select a co-design configuration for optimal operation of the 802.11a wireless standard.**

I. Introduction

In recent years, the field of wireless technology has seen a surge in the diversity of devices, protocols, and applications. Wireless devices are more prevalent and used for a variety of purposes. This increase in use has led to more needs, including modified protocols, higher data rates, and lower energy consumption. In the past, these devices were produced by ASIC fabrication. In recent years, the premise of a software-defined radio (SDR) has opened up the field to allow for reconfiguration of a transceiver to adapt to evolving protocols. However, with many computing architectures and languages available, no parties have yet identified how to optimally configure and map any protocol to a set of processing elements on a heterogeneous system. Specifically, researchers have not yet to fully decided on which components are best suited for processor software (SW) or reconfigurable hardware (HW).

We explore this problem using commercially available tools including Simulink and Vivado. We use IEEE 802.11a transmitter (Tx) and receiver (Rx) Simulink models to ensure correctness against Annex G of the 802.11a specification[1]. We modify the models to use complex, fixed-point data types and adjust the settings in the Simulink model to better target execution on HW or SW. Then, we generate HDL code and IP core blocks for the components targeted for execution in HW. We also generate C code to be compiled into an executable that runs on the ARM processor. Finally, we gather timing information for each frame and FPGA usage statistics for the different HW/SW co-designs, leading us to conclude which designs are optimally suited for the needs of wireless applications.

II. Hardware Features and Software Tools

Our setup requires 4 power outlets, space for a standard 6 foot table, and an hour setup time. The HW consists of 2 radio frequency (RF) front ends, the Analog Devices (ADI) AD-RFCOMMS3-EBZ (FMComms3) board, 2 Xilinx Zynq ZC706 evaluation boards, and 2 host laptops. The FMComms3 features the wideband wireless AD9361 transceiver chip and attaches to the FPGA Mezzanine Card (FMC) slot on Xilinx FPGA and SoC boards. The AD9361 transceiver supports up to 2 transmit (Tx) and 2 receive (Rx) channels at bands from 70 MHz to 6.0 GHz[2]. Each ZC706 board has 3 connections to host PC: UART, JTAG, and Ethernet. Start/stop signals are sent between PS and host by Ethernet. The ZC706 features the Xilinx Zynq Z7045 SoC, which has an ARM-based processing system (PS) and Kintex-7 FPGA-based programmable logic (PL). The 802.11a transceiver system contains both a Tx path, from PS to PL to FMComms3, and a Rx path, from FMComms3 to PL to PS, shown in Fig. 1.

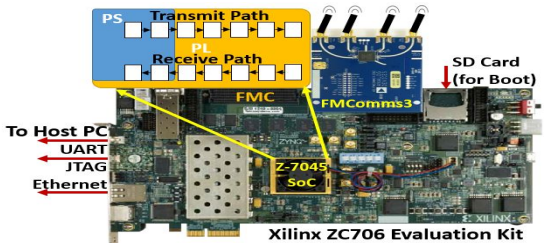

Fig. 1: 802.11a Transceiver HW using Zynq SoC & FMComms3

In addition to HW, specialized SW is needed to effectively work with the SDR systems and perform signal processing tasks such as modulation, encoding, and filtering. We use Simulink to create and simulate synchronous dataflow models. Additional HW support packages allow us to interface with RF front ends. We use MathWorks tools on the host to describe the high-level design of the transceiver system and implement it on the Zynq, as illustrated in Fig. 2.

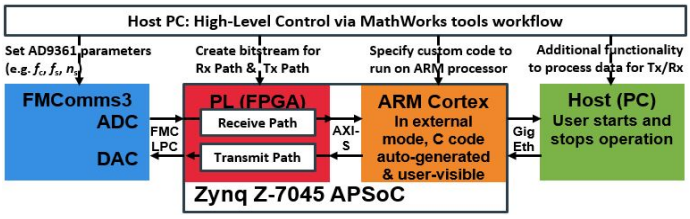

Fig. 2: High-Level SW Workflow for Zynq APSoC & FMComms3

978-1-4673-9955-5/16/$31.00 ©2016 IEEE

Each Simulink model captures all the information about the Zynq transceiver system. Using Zynq-Based Radio blocks, this model sets parameters related to the FMComms3, including center frequency, sampling frequency, and number of samples per frame[3]. The model clearly distinguishes the subsystem targeted for execution on the PL from the other model components which are targeted to run on PS. Using HDL Coder support for Zynq, we generate HDL code for the PL block. Two IP Core blocks encapsulate the Design Under Test (DUT) for Tx and Rx. A Vivado diagram is generated to combine the Rx DUT, Tx DUT, and all the AXI interface components shown in Fig. 3. Xilinx Vivado is invoked to synthesize, implement, and make a bitstream for the PL[4].

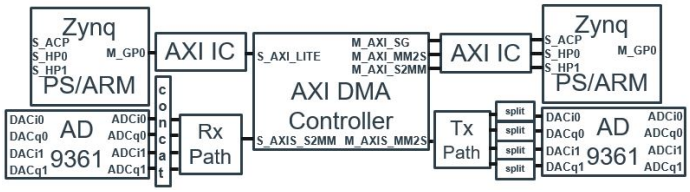

Fig. 3: AXI Interface between Zynq APSoC & AD9361

Next, we generate an executable for the PS. Using Embedded Coder support for Zynq, we generate C code for all blocks on the PS[5]. Xilinx SDK is invoked to package and compile the executable for the PS[6]. Another Simulink model is generated to run in External mode, where pressing the play button sends a signal via Ethernet to launch the executable.

III. Transceiver HW-SW Co-design

To meet real-time constraints, an 802.11a transceiver must complete its PHY-layer transactions before the end of each sampling period or else suffer unacceptable data losses. These transactions include scrambling, convolutional encoding, block interleaving, BPSK modulation, symbol-to-subcarrier mapping, OFDM modulation, and cyclic prefix attachment. We explore HW-SW co-design by creating a number of Tx and Rx model variants that implement an varying number of functional blocks in the Zynq PS and PL. The transmitter model dataflow is shown in Fig. 4.

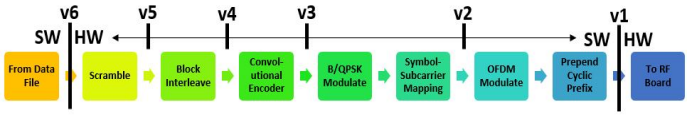

Fig. 4: 802.11a Transmitter Chain HW/SW Co-design Variants

For the transmitter, the initial model (v1) is a SW-only design that implements all functional blocks on the PS. As the variant number increases, the design incorporates an additional component or two on PL. The final model (v6) is a HW-only design that performs all data processing on PL. A similar approach is taken for the Rx model, which contains an additional component for preamble detection and thus has 7 variants, as shown in Fig. 5. Since operations on the PL fabric can operate many times faster than on the ARM, we model PS blocks to process one frame at a time, and have the PL process one sample at a time.

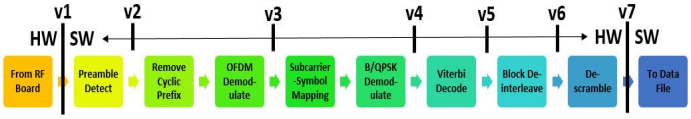

Fig. 5: 802.11a Receiver Chain HW/SW Co-design Variants

TABLE I: Results: Execution Time

Version	PL Lat (μs)	Max PS Step Time (μs)	Lookup Tables (#/% Util)	Registers (#/% Util)
v1 (Preamble)	1.64	480	5040/2.31%	6334/1.45%
v2 (IFFT&CP)	5.74	55	10183/4.66%	12549/2.87%
v3 (BPSK&Map)	7.02	55	10213/4.67%	12467/2.85%
v4 (Interleave)	7.98	55	10324/4.72%	12469/2.85%
v5 (ConvEncode)	8.04	55	10377/4.75%	12478/2.85%
v6 (PL-Only)	8.06	36-55	10633/4.86%	12657/2.90%

IV. Experimental Results

The execution timing results for the 802.11a transmitter are shown in table I. The maximum PS step time decreases as more components are moved onto the PL. Moving the IFFT to PL results in the largest drop in step time. To meet the 802.11a specifications, we would need a 4 μs frame time. Thus, we need to perform further computational optimizations in order to reduce the max PS step time. The resource utilization results show increasing lookup table (LUT) and register usage as more components are put onto PL. However, in all model versions, even the PL-only variant, the FPGA is at less than 5% utilization, meaning that it retains many LUTs and registers for use by the receiver and higher OSI layers.

V. Conclusion

We have introduced and explored a method for exploring HW-SW co-designs for 802.11a wireless transmission and reception systems. We have shown that for direct-feedthrough algorithms, execution entirely in HW (v6) results in faster execution speed, but adds the risk of overwhelming FPGA resources. For future work, we plan to perform tests with live, online radio transmissions to measure bit error rate (BER) for the different co-designs. This demo can be used as a basis for future work in MIMO and higher layers (e.g. MAC). Once this design has been fully explored, the system can be adapted to more modern wireless standards such as 802.11ac for beamforming, 802.11af for UHF band reuse, and 4G LTE.

References

[1] *IEEE Std 802.11 a-1999*, IEEE 802.11 Working Group Std., 1999.

[2] Analog Devices, Inc. (2015) Integrated transceivers, transmitters, and receivers. [Online]. Available: http://www.analog.com/en/products/rf-microwave/integrated-transceivers-transmitters-receivers.html

[3] MathWorks, Inc. (2016) Communications system toolbox support package for xilinx zynq-based radio. [Online]. Available: http://www.mathworks.com/help/supportpkg/xilinxzynqbasedradio

[4] Xilinx, Inc. (2016) Vivado design suite - hlx editions. [Online]. Available: http://www.xilinx.com/products/design-tools/vivado.html

[5] MathWorks, Inc. (2016) Embedded coder support package for xilinx zynq-7000 platform. [Online]. Available: http://www.mathworks.com/help/supportpkg/xilinxzynq7000ec

[6] Xilinx, Inc. (2016) Xilinx software development kit (sdk). [Online]. Available: http://www.xilinx.com/products/design-tools/embedded-software/sdk.html