SDR Control Interface
A FPGA based infrastructure for control of VPX Software Defined Radio systems

Stefanie Castillo, Armando Astarloa, Jesús Lázaro
University of the Basque Country
Bilbao, Spain

Sergio Salas
System-on-Chip *engineering*
Zitek Bilbao - ETSI
48013 Bilbao, Spain

Isaac Ballesteros
iQUBE Research
Ra Fonte das Abelleiras, s/n
36310, Vigo, Spain

***Abstract*—We present a flexible, scalable, re-usable open-source based low-overhead BIST and control SoPC hardware infrastructure: the *SDR control interface*. Our approach targets to enable centralized VPX switch-fabric systems with a hardware infrastructure to control and test the RF Transceiver ICs in a SDR system. We include a detailed description of the design, shows results of an FPGA-based implementation of the system and finally experimental validation of the system is carried-out by successfully configuring and controlling the Texas Instruments RF Transceiver SDR ICs.** *

I. Introduction

Flexible, scalable, cost-effective and low time-to-market solutions for systems are required to endure the rapid evolution of technologies in today's industry [1], [2], [3]. Time-to-market is essential in commercial and industrial products, therefore developments in these areas must be highly flexible, scalable and also re-usable in other products [3]. Furthermore, any development on these areas should embrace standardized communication interfaces to be easily interconnected to other systems; as the industry is heading towards a standardized interconnection of system-of-systems [4].

Since its inception in the radio industry [5], Software Defined Radio (SDR) technology promised to solve the high level of dependency on dedicated hardware infrastructure, of classic Radio Frequency (RF) modems. Usually a RF modem consist of three main blocks, that is: the Baseband Integrated Circuit (IC), the Radio Frequency IC (a.k.a. RF Transceiver) and a RF Front-End. The overall RF modem can be subdivided in four main stages, that is: the analog front-end (AFE), the digital front-end (DFE), the inner modem and the outer modem [6].

Within the RF Transceiver there is a digital part which is the DFE and there is also an analog part which comprises in between others: the Analog-to-Digital Converter (ADC), the Digital-to-Analog Converter (DAC), modulator/demodulator and PLL/LO generator. The SDR approach implies full re-configurability or re-programmability of all blocks in the RF modem. In particular we will refer to the reconfigurability/re-programmability of the ICs of the RF Transceiver to support multi-band, multi-mode wireless radio architectures. Therefore SDR systems require an infrastructure to reconfigure, test and control the components of the RF Transceiver ICs. This is where we focus our design.

High-speed switch fabric topologies are having more and more presence on industrialized systems. VPX is the latest VME backplane industry standard, with support for switched fabrics [7]. VME bus systems have been used in all segments of science, industrial, and military applications [8]. An industrialized SDR system is typically implemented in a backplane architecture and VME is the backplane standard most frequently used by vendors [9].

With this background in mind, we propose a simple low over-head Built-In Self-Test (BIST) and control SoPC infrastructure to easily enable VPX SDR systems with a hardware infrastructure to control the RF Transceiver SDR ICs. Our approach embraces the previously mentioned industry requirements by using a standard industrial backplane infrastructure (the VPX standard) standardized open-source on-chip bus architecture (wishbone), open-source Intellectual Property (IP) cores [10], and standard communication protocols (Inter-Integrated Circuit (I^2C), Universal Asynchronous Receiver/Transmitter (UART)) to maximize the re-use of existing intellectual property and maximize compatibility to other systems. Finally, our approach addresses specifically VPX centralized switch-fabric systems, which are highly used nowadays on high speed industry solutions. Our design can be also easily extended to fit the topology and necessities of other systems.

The reminder of this paper is structured as follows: in Section II a review of related work on the subject is carried-out; in Section III the SDR control interface architecture is covered; in Section IV the architecture, protocol and the two versions of the main component of the system: the BIST Slave IP core (BS IP Core), are be presented; Section V exposes the experimental results of the BIST-control interface integrated in a VPX SDR system and finally Section VI concludes this paper.

*Preprint version, submitted to IEEE.

II. Related work

A good overview of the evolution of RF transceiver control architectures in industry is done in [11]. In the early days of RF modems the RF transceiver control system had a digital serial interface, and the data signals had usually a differential analog I/Q interface. Later on, a digital interface (DigRF) between the baseband IC and the transceiver IC was introduced. First, transceivers had separate digital interfaces for control and data [12] and later on with the inception of the UMTS standard, a shared interface for both control and data [13] was introduced. The latest DigRF standard [14] addresses the increased bandwidth handled by HSPA+ and LTE architectures. Even though a standard has been established in industry it is still common to have SDR projects with serial based interfaces, like in the early days of RF transceivers. For instance in this fully reconfigurable SDR transceiver [15], the control system is a scalable Network-on-Chip (NoC) that controls each circuit based on the input it gets from a serial interface.

In other cases extra hardware has to be included in order to support the control interface for the SDR system. For example in the the SDR platform KUAR [16]. This platform includes both the baseband and the RF transceiver blocks. In this project the authors developed a similar system to ours which is based on an interconnection of three printed circuit board cards: a power board, a control processor board, and a RF transceiver board. The baseband processing part is comprised of both an Field Programmable Gate Array (FPGA) and a General Purpose Processor (GPP). The approach the authors use for configuration and control of the RF transceiver ICs of their SDR system, was to use a Freescale 8-bit micro-controller unit to interface the digital processing section with the programmable components on the RF transceiver module. In this case an extra micro-controller was needed to enable configuration and control of the RF transceiver board ICs, whereas with our approach, this extra micro-controller is eliminated, as the control system could be included within the FPGA used by the authors for digital processing applications.

In general, the design of the control system and it's design specifications is not well described or it is highly tied to the specific SDR system. We find this limiting when building a control interface for a new SDR system. For instance, in WARP [17] the authors present an open-access programmable wireless platform. The platform consist of custom hardware, platform support packages and an open-access repository. This project matches the idea of an open-access platform, but all hardware and packages are highly tied to the architecture of the platform, making it unfeasible to be integrated with other architectural designs in any other industrial or commercial systems. In one of the latest publications regarding control interfaces, SORA [18], the authors present a SDR platform using both FPGAs and GPP in the system. The system is subdivided in two main blocks, what they call the *soft radio stack* and a *radio front-end*. These two main blocks are interconnected by a memory and a radio control board (RCB). The radio front-end comprises what is the analog part of the RF Transceiver and the RF front-end. The RCB provides a low-latency control path for software of the GPP to control the radio front-end hardware, which includes the RF Transceiver ICs. Inside the RCB there is a component called the *RF Controller* which is the one who controls and communicates with the radio front-end. The approach matches with ours in that the control sub-system it is embedded within the FPGA, no need for extra hardware. Nevertheless the authors don't provide details on the implementation of this control infrastructure, which makes it unviable to reproduce.

Lots of approaches have been adopted to configure the RF Transceiver ICs on SDR platforms Within this evolution of control interfaces for RF transceivers, our approach matches the early days implementation of a control interface for RF transceivers as a serial interface. Our main contribution is to give detailed implementation of the control interface, highly used in literature but we have found lack of detail description. Our main requirements were for the design to be as small as possible to minimize area, utilize only open-source blocks to enable for others to reproduce the design and to avoid including new hardware to build the control interface for the SDR system. Even though our system was designed for a VPX system, the design can be easily adapted to fit other types of systems with minimal effort as we shall describe along this paper.

III. SDR Control Interface

The *SDR control interface* is a System-on-Programmable-Chip (SoPC) design with a topology of a centralized switched fabric of interconnected boards. Figure 1 shows the topology of the system. The master board is the *switch board*, which implements the switching functions and initiates all test and control operations. The slave *payload boards* carry the target SDR ICs for test and control operations. The boards interconnect through a serial bus over the VPX interface. The communication of the system is established through the *System Management* lines, *SR[0..3]* of connector *P0*, defined by the VPX standard.

Figure 2 shows a close-up look of a single interconnection between the switch board and a *payload board*. In this diagram one *switch board* is connected to one single *payload board* through a serial VPX interface. The *SDR Control Interface* requires a master of the system on the *switch-board* and a special slave IP-core we developed on the *payload-board*. This developed slave IP-core is the *BS IP Core*. The master of the system can be a GPP or any other device with a master serial interface.

In the test application, the ICs to test and control are the Texas Instruments (TI) SDR ICs [19], [20], [21], which conform the building blocks of a RF transceiver. On this system, the *switch board* integrates a GPP as the *System Management* bus master and a VPX interface, in between others components related to the host system's application. The *payload board* has all the circuits to test and control, a payload FPGA which carries the developed BS IP Core and a VPX interface. The SDR control interface enables the system's

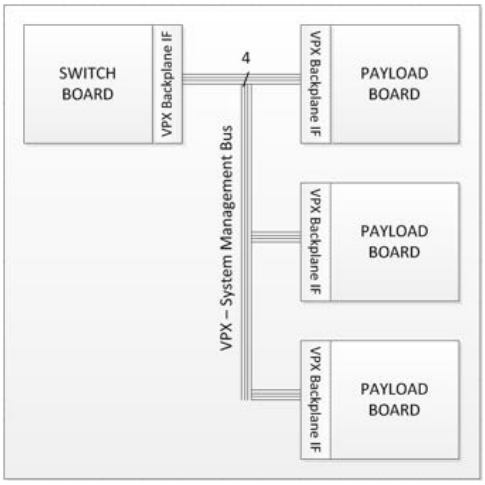

Figure 1. *SDR control interface* system topology

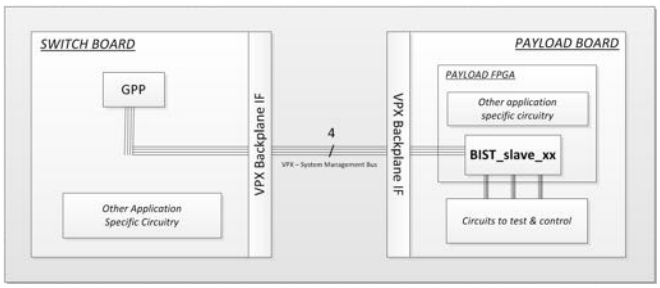

Figure 2. Interconnection between the *switch board* and a *payload board* in *SDR Control Interface*.

master on the *switch-board*, means to test and control ICs on the *payload-boards* through a command based serial protocol.

The system is intended to be integrated on industrial systems and therefore has been implemented over the latest industrialized backplane standard with support for switch-fabrics, that is: the VPX bus. There are two types of cards adopted by the VPX standard, the IEEE-1101 3U card which has 3 connectors $(P0 - P2)$ and the 6U card with 7 connectors $(P0-P6)$. In both cards, connector $P0$ has four reserved lines for *System Management* functions, these are $SM[3:0]$ [22]. In our design each board has a VPX backplane interface through which the control and test signals are routed on the *System Management* lines of the VPX $P0$ connector. The architecture of the system as a VPX centralized switched fabric is highly related to our specific needs, as this SDR system was meant to be integrated on an industrial platform. Nevertheless the same concept can be applied to other architectures, as the system main architecture has a classical *master-slave* topology.

The system requires a master on the *switch-board*, e.g. a GPP, and a special slave IP-core we developed on the *payload-board*. This IP-core is the *BS IP Core*, a synthesizable FPGA core which is a key component within the system, which we will shortly describe in detail. The BS IP Core was embedded in each of the *payload boards* of the system.

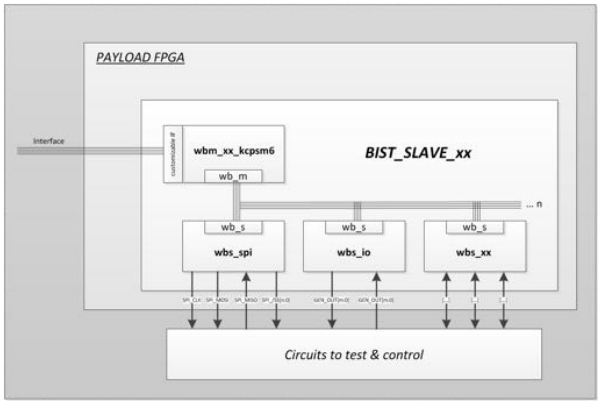

Figure 3. BIST Slave IP Core generic architecture

IV. BIST Slave IP Core

The BS IP Core and has a Wishbone Bus (WB) core-based design. Figure 2 shows the BS IP Core core as *BIST_SLAVE_XX*. The generic architecture of the BS IP Core is shown in Figure 3. To show the flexibility of the architecture, two versions of the BS IP Core were developed based on a generic template.

The core has two interfaces. One serial interface to receive commands from the VPX master of the system, and the other interface faces the circuits to test and control. The serial interface of the core, regardless of which protocol it implements, allows access to the internal WB of the core through a specified series of commands that are processed by the master of the internal WB of the core: *wbm_XX_kcpsm6* module. The serial interface protocol is customizable. In this work two cores were implemented, each implementing a different serial interface protocol.

A. Architecture

The architecture of the core is a one-master WB architecture. The master of the internal WB faces the interface to the master of the VPX fabric on the switch board. The WB slaves of the core face the interface to the circuits to test and control. In our implementation, only two WB slaves were needed, as only two external communication protocols were needed to configure the targeted TIs SDR ICs [19], [20], [21]. The two supported protocols of the interface facing the circuits to test and control are a Serial Peripheral Interface (SPI) interface and a parallel interface. More WB compliant slaves may be added to the design, enabling easy scalability of the system to support new protocols.

The design of this master core required an FPGA-synthesizable soft-processor, with small gate count and high reliability. After reviewing the available options [23], the selected soft-core was the *Pico Blaze* [24] soft-core processor. This is a small, 8-bit, cost-effective soft-core processor optimized for Xilinx's FPGAs.

The *wbm_XX_kcpsm6* master core integrates the *kcpsm6* soft-micro-processor, a WB interface to the slave cores, a communication interface to the VPX master and a VHDL

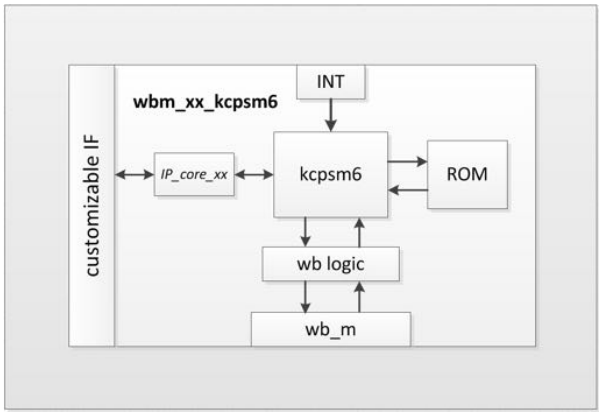

Figure 4. *wbm_XX_kcpsm6* generic core block diagram

description of a *kcpsm6* assembler firmware as a ROM, as shown in Figure 4.

This is an IP core designed in HDL, targeting FPGAs. It comprises building-blocks in both *VHDL* and *Verilog* languages. The *wbm_XX_kcpsm6* is the master of the internal WB of the core and it is the most important component of the design, as it is where all processing operations take place.

The two WB compliant slaves in the BS IP Core are:

- The SPI WB core: This core is based on and active project [25] of the *Open cores* web site [10]. Figure 3 shows this core as *wbs_spi*.
- The IO WB slave core:This core was an internal development of this project. It is a simple parallel wishbone slave core. Figure 3 shows this core as *wbs_io*.

One of our main requirements was to re-use existing IP as much as possible to minimize time-to-market and minimize non-recurring engineering costs. The main core developed in this project, BS IP Core is entirely build upon existing IP and therefore shows how the system enables re-usability of existing IP, minimizing product development metrics.

B. Commands

The BS IP Core interprets the received commands described on Table I and acts over the WB slaves accordingly. The firmware stored in the ROM, describes:

- The decoding of commands
- The hardware actions that shall be executed for each decoded command

The commands directly translate to WB access cycles in order to address, read and write the WB slaves of the internal architecture of the core. The commands allow access to a 16-bit address and 32-bit data WB. These make use of the ASCII character-encoding scheme.

The core decodes seven commands; 6 for *writing* and 1 for *reading*. Each *write* command consist of one ASCII encoded letter followed by two hexa-decimal ASCII encoded digits. The *read* command consists of only one ASCII letter with no following digits. The supported commands and protocol can be easily modified or enhanced by modifying the program executed by the *kcpsm6* soft-micro-processor.

All *write* commands are of the form:

$$C\dagger\dagger$$

Where C represents a letter in the ASCII character-encoding scheme,

$$C \in \{A(\$41), S(\$63), L(\$4c), H(\$48), V(\$56), Z(\$5a)\}$$

and each $\dagger$ represents a Hexadecimal (HEX) number in the ASCII character-encoding scheme,

$$\dagger \in \{0(\$30) - 9(\$39)\}, \{a(\$61) - f(\$66)\}$$

The only *read* command is of the form:

$$J$$

Where J represents a letter in ASCII character-encoding scheme.

$$J \in \{R(\$52)\}$$

The reason to use this ASCII based protocol is for backward compatibility with others of our systems with the same control and configuration requirements. These commands can be optimized to maximize the word length of the data bus of the control system in accordance to the target system's needs. This adjustment would require a new firmware, but the concept and hardware stays the same.

C. Versions

To show the flexibility of the system to easily adapt to different serial communication interfaces, two versions of the master core where designed: *wbm_uart_kcpsm6* and *wbm_i2c_kcpsm6*. Each version implements the specified serial interface to face the VPX master of the entire system. Figure 4 shows the generic structure of the core. Each version has the same generic structure and only differ by the serial protocol each implements over its communication interface. If we refer to Figure 4 the only diference between the two versions is the block identified by *IP_core_XX*, each implementing the desired interface.

Each version is explained as follows.

1) UART version: The BS IP Core has a UART interface. The *wbm_uart_kcpsm6* core implements a hardware UART interface via two macros developed and distributed by Xilinx, the *uart_tx6* and the *uart_rx6* [26].

2) I^2C version: The BS IP Core has an I^2C interface. The *wbm_i2c_kcpsm6* core implements a hardware I^2C interface though a modified and enhanced IP core acquired from the *Open cores* web site [10]. The original core is the *I^2CSlave* core [27].

Table I
BIST SLAVE IP CORE COMMANDS

Type	Command	Description
Address Write	Syy	sets wb_ADDRESS[15..8] = $yy
	Axx	sets wb_ADDRESS[7..0] = $xx
Data Write	Zii	sets wb_DATA[31..24] = $ii
	Vzz	sets wb_DATA[23..16] = $zz
	Hjj	sets wb_DATA[15..8] = $jj
	Lzz	sets wb_DATA[7..0] = $zz and initiates WB write cycle
Data Read	R	initiates WB read cycle of wb_DATA[31..0] in wb_ADDRESS[15..0]

V. EXPERIMENTAL RESULTS

In order to validate the developed system's compliance to the initial requirements and validate it's functionality, an implementation of the system and a proper experimental environment was assembled. The main objective was for the designed system to enable a centralized VPX switched-fabric host system with BIST and control operations over it's ICs. In our target test application, the ICs to test and control are the TI SDR ICs. These conform the IC of a RF transmitter:

- a DAC [19],
- an IQ modulator [20]
- a clock generator [21]

Before heading out to a physical experimentation environment, all cores were first validated though simulation on the ISim [28] simulation tool with appropriate test benches. Only when the proper response of the cores was validated though simulation, a physical experimentation environment was assembled.

An experimental environment was setup in the lab to validate the systems functionality. Figure 5 shows the experimental setup block diagram. In the block diagram, the master test device is a GPP and it emulates the master in the *switch-board* of our system. The FPGA board of the block diagram is the Xilinx *ML605 board* [29] and holds an implementation of the BS IP Core, emulating the *payload-FPGA* in the *payload-board*. Finally the *TSW3003* [30] shown in the diagram holds the targeted SDR ICs. The interconnected blocks in the experimental setup were:

- a GPP with a master serial interface
- the Xilinx development board for Virtex-6 *ML605* which carries the BS IP Core,
- the TI's SDR transmit board *TSW3003* which carries the ICs to test and control,
- the Agilent Logic Analysis System [31] with integrated pattern generator to inject digital I/Q symbols into the SDRs system
- the Agilent CSA Spectrum analyzer [32] to verify the correct frequency response of the system, hence validating the correct configuration of the system.

The objective of the experimental setup was to validate the functionality of the SDR control interface in combination with the ICs of the TI *TSW3003* evaluation board and a GPP. This evaluation board is a high performance RF transmit board which has all the circuits our *payload-board* includes. The system was validated with the use of the Xilinx *ML605 board* a GPP, the TIs RF transmit board, a patter generator and measurement instrumentation to validate the correct RF response of the system. The validation also included the verification of the system's compliance to the implemented protocols by frame analyzer. The pattern generator was used to insert digital I/Q symbols to the DAC. The system successfully tested, controlled and configured the SDR ICs on the TIs board, showing the correct RF response according to the inserted I/Q symbols.

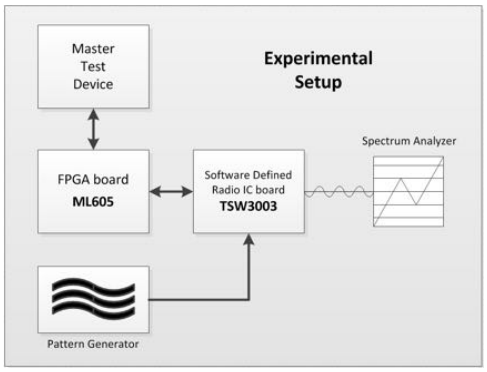

Figure 5. Experimental setup block diagram

The FPGA used in the experimental setup was the Xilinx Inc. Virtex-6 [33]. The implementation results of the BS IP Core on the Xilinx Virtex-6 XC6VLX240T-1FFG1156 FPGA on the ML605 board are shown in Table II. The UART version of the core uses less resources than the I^2C version, as the first protocol is simpler than the second one. As it can be depicted from these implementation results, the resources utilization is very low; these validate the core's architecture and our requirement to minimize size of the development. This validates the core's architecture, as it implements a BIST-control infrastructure with minimal additional overhead.

VI. CONCLUSION

In this paper we have presented a flexible, scalable and reusable SoPC hardware infrastructure to configure and control SDR ICs. In SDR platforms, this control infrastructure is always need, as reconfigurability is the heart of a SDR

Table II
IMPLEMENTATION RESULTS OF THE BS IP CORE ON A XILINX VIRTEX-6 XC6VLX240T-1FFG1156 FPGA

Core	Slice Registers	Slice LUTs	BRAMS
BIST SLAVE UART	483 (0.16%)	689 (0.20%)	1 (0.12%)
BIST SLAVE I2C	594 (0.46%)	826 (0.55%)	1 (0.12%)

system. To address this need in SDR systems, the *SDR control interface* implements a low-overhead *BIST-control system* by means of a SoPC solution.

Along the evolution of control interfaces, our design matches the serial digital interface implementation. This type of interface is highly used in current developments. Within this evolution, we have identified the lack of detailed description of this type of control system in literature, which we aim to engage with this disclosure of an entirely reproducible open-source based control system.

Our approach targets industrialized VPX switched fabric systems in need of a hardware infrastructure to test and control the RF Transceiver ICs of a SDR system. This approach can be easily adapted to other systems as the main topology of our system is a classic *master-slave* topology.

Our design allows to accelerate time-to market of SDR systems in need for BIST and control tasks by embracing this simple, flexible, scalable, re-usable, open-source based hardware infrastructure. With this paper we encourage the community to re-use the design of this simple, open-source based control and configuration system to enable their SDR systems with control and BIST capabilities.

We wish to further extend this work by optimizing the design to minimize power consumption. As future research we intend also to evaluate the throughput of the control system to support real-time control operations in radio systems.

REFERENCES

[1] A. Ghosal, P. Giusto, P. Sinha, and H. Zeng, "Metrics for quantifying and evaluating ability of electronic control system architectures to accommodate changes," 2011.

[2] A. Allan, D. Edenfeld, W. Joyner, A. Kahng, M. Rodgers, and Y. Zorian, "International technology roadmap for semiconductors," *IEEE Comput*, 2002.

[3] F. Luo, W. Williams, R. Rao, R. Narasimha, and M. Montpetit, "Trends in signal processing applications and industry technology [In the spotlight]," *Signal Processing Magazine, IEEE*, vol. 29, no. 1, pp. 184 –174, Jan. 2012.

[4] H. Kopetz, *Real-time systems: design principles for distributed embedded applications*. Springer-Verlag New York Inc, 2011, vol. 25.

[5] I. Mitola, J., "Software radios: Survey, critical evaluation and future directions," *Aerospace and Electronic Systems Magazine, IEEE*, vol. 8, no. 4, pp. 25 –36, Apr. 1993.

[6] R. Fasthuber, "An energy-efficient architecture template for wireless communication systems," Ph.D. dissertation, Katholieke Universiteit Leuven, Belgium, Sep. 2012.

[7] R. Cooper and M. Littlefield, "OpenVPX: architectures for High-Performance embedded computing," in *Thirteenth Annual Workshop on High Performance Embedded Computing (HPEC2009).—Massachusetts Institute of Technology, Lincoln Laboratory*, 2009, pp. 22–24.

[8] V.I and Vinogradov, "Advanced high-performance computer system architectures," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 571, no. 1–2, pp. 429 – 432, 2007, proceedings of the 1st International Conference on Molecular Imaging Technology 2006.

[9] E. Grayver, "Standardization efforts for software-defined radio," in *Aerospace Conference, 2010 IEEE*, Mar. 2010, pp. 1 –8.

[10] Open Cores. (2011) Free open source ip cores and chip design. [Online]. Available: http:/www.opencores.org,

[11] S. Brandstatter, B. Neurauter, and M. Huemer, "A novel architectural approach for control architectures in RF transceivers," in *SOC Conference (SOCC), 2010 IEEE International*, Sep. 2010, pp. 407 –412.

[12] MIPI Std., "DigRF v1.12 Baseband/RF digital interface specification," Feb. 2004.

[13] ——, "DigRF v3.09 dual-mode 2.5G/3G baseband IC - radio frequency IC interface standard," Nov. 2006.

[14] ——, "DigRF v4 v1.10," Jan. 2012.

[15] J. Craninckx, M. Liu, D. Hauspie, V. Giannini, T. Kim, J. Lee, M. Libois, D. Debaillie, C. Soens, M. lngels, A. Baschirotto, J. Van Driessche, L. Van der Perre, and P. Vanbekbergen, "A fully reconfigurable software-defined radio transceiver in 0.13um CMOS," in *Solid-State Circuits Conference, 2007. ISSCC 2007. Digest of Technical Papers. IEEE International*, Feb. 2007, pp. 346 –607.

[16] G. Minden, J. Evans, L. Searl, D. DePardo, V. Petty, R. Rajbanshi, T. Newman, Q. Chen, F. Weidling, J. Guffey, D. Datla, B. Barker, M. Peck, B. Cordill, A. Wyglinski, and A. Agah, "KUAR: a flexible software-defined radio development platform," in *2nd IEEE International Symposium on New Frontiers in Dynamic Spectrum Access Networks, 2007. DySPAN 2007*, Apr. 2007, pp. 428 –439.

[17] K. Amiri, Y. Sun, P. Murphy, C. Hunter, J. Cavallaro, and A. Sabharwal, "WARP, a unified wireless network testbed for education and research," in *IEEE International Conference on Microelectronic Systems Education, 2007. MSE '07*, Jun. 2007, pp. 53 –54.

[18] K. Tan, H. Liu, J. Zhang, Y. Zhang, J. Fang, and G. M. Voelker, "Sora: high-performance software radio using general-purpose multi-core processors," *Commun. ACM*, vol. 54, no. 1, p. 99107, Jan. 2011. [Online]. Available: http://doi.acm.org/10.1145/1866739.1866760

[19] Texas Instruments Incorporated, "Dac5687 datasheet (rev.e)," SLWS164E, September 2006.

[20] ——, "Trf3761 datasheet (rev.j)," TRF3761j, August 2008.

[21] ——, "Cdcm7005 datasheet (rev.d)," SCAS793D, August 2009.

[22] *American National Standard for VPX Baseline Standard (rev. 1.2)*, VMEbus International Trade Association Std. ANSI/VITA 46.0-2007, October 2007.

[23] J. Tong, I. Anderson, and M. Khalid, "Soft-Core processors for embedded systems," in *Microelectronics, 2006. ICM '06. International Conference on*, Dec. 2006, pp. 170 –173.

[24] K. Chapman, *PicoBlaze for Spartan-6 and Virtex-6 (KCPSM6)*, Application note, Xilinx, Inc., March 2011.

[25] S. Srot, "Spi master core project," Open Core project, March 2004.

[26] K. Chapman, *Ultra-Compact UART Macros for Spartan-6 and Virtex-6*, Xilinx, Inc, May 2011.

[27] S. Fielding, "i2cslave specification (rev.1.1)," Open Core project, December 2008.

[28] Xilinx, Inc., "Isim user guide," UG660, September 2009.

[29] ——, "Ml605 hardware user guide (v1.6)," UG534, July 2011.

[30] Texas Instruments Incorporated, "Tsw3003 demonstration kit, user's guide (rev.d)," SLWU029D, August 2007.

[31] Agilent Technologies, Inc., "Agilent 1680 and 1690 series logic analyzers," 5988-2675E, October 2003.

[32] ——, "Agilent csa spectrum analyzer n1996 datasheet," 5989-3678E, September 2010.

[33] Xilinx, Inc., "Virtex-6 family overview (v2.3)," DS150, March 2011.