Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics
San Antonio, TX, USA - October 2009

Software Defined Radio for Broadband OFDM Protocols

Brian Kelley, Senior Member of the IEEE
Dept. of Electrical and Computer Engineering
University of Texas at San Antonio
San Antonio, TX, USA
Brian.Kelley@utsa.edu

***Abstract*—We describe new concepts in broadband software define radio architectures optimized for very high bandwidth wireless communication protocols. This is important since a trend in wireless communication transceivers is the adoption of increasingly sophisticated radio link algorithms which maximize utility functions consisting of network capacity, data rate reliability, and throughput. Many prior efforts in software defined radio suffer from either a presumption that sufficiently high clock rates exist to employ traditional single instruction multiple data (SIMD) multi-processor architectures or that renaissance programmers are available to convert thousand page wireless protocol specifications into fine-grain data flow graphs at the operation level. The challenge for new generations of software defined radio is to maintain flexibility whiles simultaneously supporting computationally efficient broadband communication algorithms and ease of programming. Our system architecture is viable for wide varieties of communication protocols and physical data channels. We focus principally on OFDM transceiver styles due to their bandwidth scalability and their popularity in many next generation wireless air interfaces. Our SDR system jointly minimizes power, maximize algorithm flexibility, and to enable rapid software re-programming. We feel that these concepts are critical for the adoption of software defined radio in 21st century broadband wireless networks.**

Keywords— communications, 4G, OFDM, multicarrier modulation, SDR, software defined radio, broadband, flexibility.

I. Introduction to Introduction To Broadband Software Defined Radio

The concept of software defined radio (SDR) is traceable to the early efforts of Joseph Mitola who defined (see [1]-[7]) software defined radio as "a set of Digital Signal Processing primitives (and) a meta-level system for combing the primitives into a communication system function and a set of target processors on which the software radio is hosted for real-time communications." More recently, groups such as the Software Radio Forum have modified the original definition (see [8],[22]) to include "concepts, technologies and standards for wireless communications systems and devices, to support the needs of all user domains including consumer, commercial, public safety, and military markets, and stakeholders such as regulatory authorities….and a collection of interfaces and standards."

Generally, no single definition exists for SDR. We submit that one shortcoming of prior research in this area is the non-demarcation of narrow-band SDR from broadband SDR. We propose that broadband SDR be placed under a different category definition. In light of this, we proffer the following definition of broadband SDR: "an extensible, flexible, spectrally efficient radio system which, across the categories of broadband wireless protocols, enables early and late binding capability versus performance tradeoffs." In this definition, early binding implies system design that integrates a priori knowledge of the protocols and algorithms. Late binding implies adjusting the design parameters after the hardware has been manufactured. Typical design tradeoffs include latency and receiver sensitivity versus computational efficiency and power efficiency. Programmable software algorithms enable the SDR radio flexibility. However, as will be discussed in the next section, programmability is a flexibility component under our definition of flexibility. Three key principles in our new definition of broadband SDR are new concepts in flexibility, extensibility, and tradeoffs of capability.

In section II, we introduce new flexibility concepts in broadband SDR. In Section III, we present the notion of SDR extensibility. Finally in Section IV, we describe new system architectures for SDR supporting 3GPP LTE.

II. Broadband SDR Concepts

A. Definitions in Broadband SDR Flexibility

An inexorable trend in the advancement of wireless networks is the progression of network capacity up the efficiency curves outline by Claude Shannon in [9]. Due to the fact that modern networks are evaluated in terms of spectral efficiency relative to theoretical peak capacity, communication radio engineers will continue to increases communication requirements at a rate which is faster than computational improvement in hardware systems. This leads to the infamous Moore's law gap shown in Figure 1. Therefore, the notion that SDRs can program their algorithms in software misses an important concept: irrespective of the wireless protocol, communication engineers can always pursue communication algorithms unrealizable to the target SDR.

978-1-4244-2794-9/09/$25.00 ©2009 IEEE

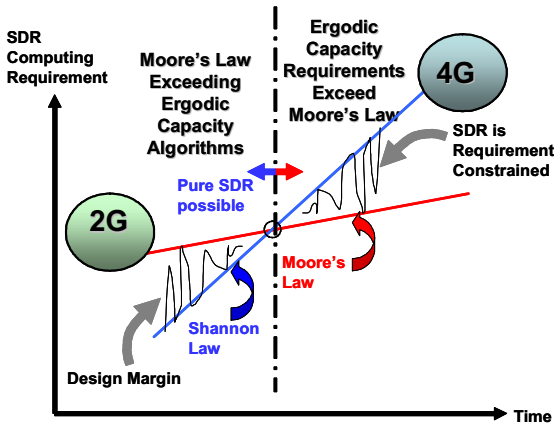

Figure 1: Communication algorithm requirements inevitably outstrip hardware capability.

B. SDR Flexbility Equation

Two key attribute areas that we have defined in regards to broadband SDR are flexibility and extensibility. Traditionally, flexibility is embodied in the ability of the radio system to take on highly variable styles of algorithms and RF configuration. We often conceptualize this as the ability of the hardware to permute. However, permutation is only one aspect of flexibility. Often overlooked are design-margin flexibility and modification-time flexibility. We therefore define flexibility as

$$\text{flexibilty} = \min\left[\begin{array}{l} k_0 \times \text{permutations}, \\ k_1 \times \text{design\_margin}, \dfrac{k_2}{\text{modification\_time}} \end{array}\right] \quad (1)$$

The constants k_0, k_1, and k_2 in (1) normalize the relative importance of each constituent flexibility component.

Transceiver design-margin involves those aspects of the radio system that effectively destroy the performance of the wireless application or hinder operation in a manner consistent with the required quality of service (QOS). Typical system attributes are power-consumption or cost. In our framework, systems exceeding the target battery capacity, for instance, are not flexible (i.e. low design margin). Power, though important, is often an elastic parameter in wireless systems. A mandated set of inelastic parameters are typically co-channel interference performance, packet error rate (PER) in AWGN noise, minimum sensitivity in Eb/No, and PER operation in specific urban and rural channel environments [10]. Failure of wireless systems to meet mandated requirements can invalidate wireless system. We consider them inflexible based upon Equation (1) design margin component.

SDRs are typically associated with programmable systems. We note that an SDR based upon an instruction set of size N constrained by a program memory size M is capable of exponential degrees of permutation (i.e. $O(N^M)$) of the instructions. However, in our notation of flexibility, system on the precipice of design margin constraints, *in spite of significant permutation*, still rank low in the flexibility component.

A third often overlooked component of flexibility is modification time. Here, we make no distinction as to the style of programming (i.e. FPGA level or micro-processor software level or C++ network level). The only key attribute that we consider is how rapidly the modification occurs. This is highly correlated with the complexity of the programming paradigm. Our flexibility model ranks SDR designs relying on low level machine instructions programming paradigms low in modification flexibility. Conversely, on the other end of the spectrum, the sole fact that the modification is possible through high level software does not necessary translate to swift modification time under our rule. Microsoft Vista, for instance, contains 50 million lines of code!

We note that the only inevitable, universally sought requirement for SDR is the desire to change the system capability. If we consider SDR as a firmware system, we can generally categorize firmware as a seamless integration of software and hardware components. A key aspect to minimizing SDR modification time (thereby increasing flexibility) is the decoupling of hardware improvement from software improvements. This can be accomplished through virtual interface definitions (see [2],[11],[22]). Closely related to this is the key tradeoff in the choice of algorithm and the degree to which the algorithms are suitably migrated to either software or firmware hardware. Suitably developed virtual interfaces abstract the computed protocols and algorithms from the underlying hardware or software state machines systems they are computed on. Therefore, virtual interfaces allow upgrades to hardware to occur with little to no changes in the software, thereby reduce modification time. This is a key in our flexibility framework. In broadband SDR, we can anticipate a queue of in-network wireless algorithms waiting to be downloaded over the air when hardware upgrades are detected in wireless nodes. The major concept is that the hardware upgrades are a continual process. Virtual interfaces minimize modification time of software upon upgrading hardware.

The converse constraint is that new SDR software should not affect existing hardware. In this sense, newly loaded software can query the hardware capability profile upon loading and can reconfigure its algorithms to enable the highest capacity communication protocols supported by the hardware configuration.

III. HARDWARE EXTENSIBILITY

Another key attribute of broadband SDRs is the hardware extensibility. By extensibility of the hardware, we imply how well it copes with both defined (early binding) attributes the time of hardware inception and undefined or (late binding) attributes at the time of hardware manufacturing. Extensibility can be thought of as "future-proofing." We note that the challenge here is to enable three styles of firmware

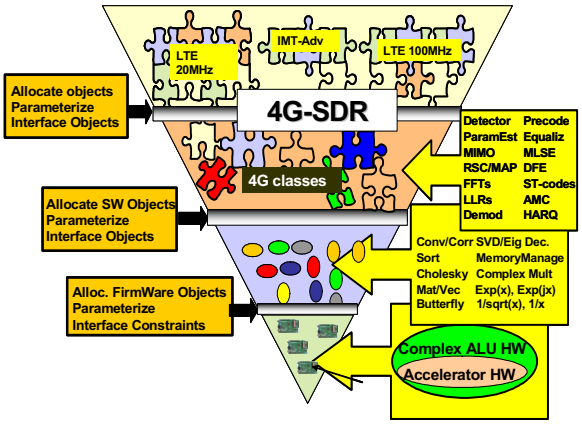

Figure 2: Hierarchical Model of SDR System

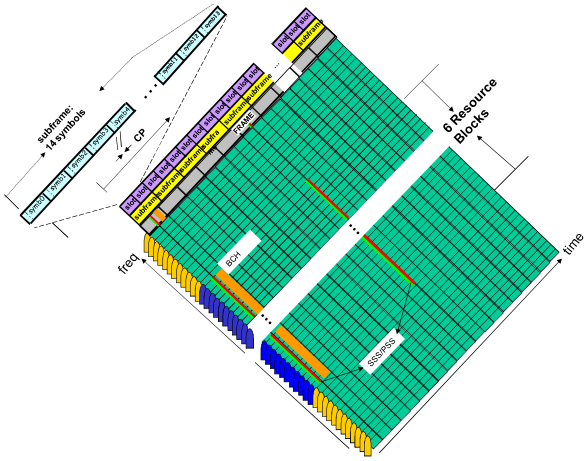

Figure 3: Down link shared channel model for 3GPP LTE

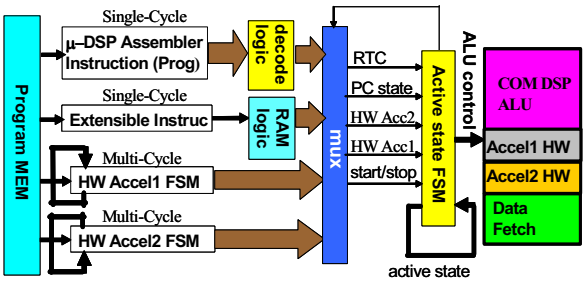

Figure 4: Extensible processor nodes for 4G SDR.

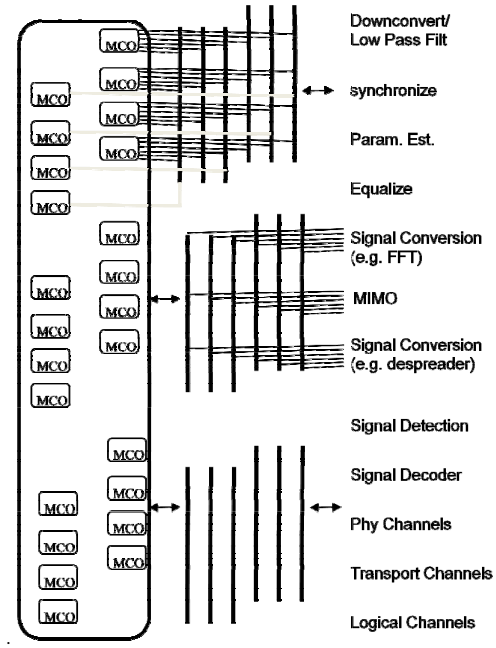

Figure 5: Receiver MCO mapping to SDR algorithms.

extensibility. The first style assigns spare computational capability into the broadband SDR system for unforeseen applications. This is typically accomplished by integrating lightly loaded microprocessors and DSPs into the system definition. The spare computation is reserved for upgrades. This style of extensibility can be real time if the new software flow descriptions can be dynamically loaded.

A second style of hardware extensibility involves the notion of external, multi-cycle finite state machine systems. We model such external peripherals and multi-cycle instructions. This style of hardware extensibility involves the addition of external hardware systems that access parallel and serial I/O pins to increase the efficiency and scope of the on chip (single or multi-cycle) SDR instruction set. It is, in a sense, an incremental modification.

Another unique extensibility style involves the ability of the fundamental SDR system to be hardware redefined so as to achieve new architectural paradigms. Examples of this would be converting a real HW ALU to a complex HW ALU, modifying single cycler per clock operation to super-scalar, or transforming single ALU systems to parallel ALU systems. This style of hardware reprogramming should be evaluated as a type of extensibility if the modification time, though non-real time, is reasonably low. We consider modern RTL hardware description language, as a method of reprogramming hardware. Note that several application specific processor compilers exist which convert architectural descriptions to a programmable machine with a correct by construction compiler [12]-[15]. Such C-compilers are generated involved only when application specific hardware is re-programmed. If

mapped to FPGAs, the modification time is on the order of that associated with traditional software.

IV. System Architecture Description of a SDR Supporting 3GPP-LTE

Our SDR system architecture description is illustrated in Figure 2. We focused on broadband wireless protocols such as 3GPP-LTE [16]-[20] as a target application. Figure 3 illustrates the physical layer downlink shared channel (PDSCH) for 3GPP-LTE. The downlink physical channel of 3GPP-LTE is a shared resource upon which multiple users simultaneously receive downlink transmission. The downlink utilizes QPSK, 16QAM and 64QAM multi-carrier orthogonal frequency division multiplexing with multiple access (MC-OFDM-A). The 3GPP-LTE protocol supports both frequency division duplex (FDD) and time division duplex (TDD) signaling. In FDD mode, 15 KHz spaced sub-carriers are denoted as resource elements (REs). The normal cyclic prefix is 4.76 μsec.

Analysis of control and signaling overhead also lead the 3GPP-Radio Access Network (RAN) group to define a smallest resolvable time-frequency data structure, known as a resource block (RB). For a downlink transmission using a normal cyclic prefix, a resource block consists of 2-Dim grid of 12 REs in frequency versus a time grid of 14 OFDM symbols. Each OFDM symbol waveform is approximately 71.4 μsec in duration. The actual duration depends on the size of the cycle prefix chosen. A subframe represents a 2 OFDM slots. A frame is 10 msec, and corresponds to 10 sub-frames or 20 slots. This is illustrated in Figure 3.

A. Micro Computer Object Architecture

Our broadband SDR solution involves defining an ALU firmware systems with hardware acceleration complexes modeled as programmable, extensible, and invokeable micro-level communication objects (MCOs). This implies that we have an object oriented class description of the MCO hardware unit. Sometimes we interact with the physical hardware unit and sometimes with its firmware model which can be instantiated as an object. In Figure 2, MCOs are shown at the bottom of the pyramid. A more detailed illustration of a single MCO is shown Figure 4 and Figure 6. At some level, as shown at the bottom level of Figure 2, all computation is ultimately enjoined by MCOs. In addition to hardware accelerated multi-cycle instructions, each MCO supports extensible instruction set design. As shown in Figure 5, the MCOs, are allocated on an as needed bases to meet the computational requirements. If the computing margin is low, excess MCOs are added to the hardware complex to insure an extensibility of computational requirements. Allocation of MCOs is distributed according to the dataflow profile, which is protocol specific. Each MCO contains collections of software primitive classes with objects instantiated and constructed prior to run time and dynamically parameterizable during run time. Virtual interfaces are created so as to emulate algorithmic parameters. This first level interface exists between the MCO level and the software primitive level so as to abstract the hardware configuration from the software algorithms. The second level of Figure 2 contains algorithmic class primitives such as butterfly, correlate, sort, convolution, ComplexMAC, and ComplexDIV instructions.

The MCO is a programmable application specific DSP engine generated through standards means such as [12],[14]. All MCOs contain a core homogenous set of complex arithmetic instructions, some of which are single cycle and some of which are multi-cycle. Furthermore, MCOs also contain a differentiated set of hardware accelerated instructions unique to that class of MCO. For example, demodulator MCOs and decoder MCOs both contain complex multiply-accumulate arithmetic logic units (CMAC ALUs). However decoder MCOs also contain specialize multi-cycle MAP instructions to accelerate Turbo decoding. Such decoding can be performed with the homogeneous set of instructions on the demodulator MCO. However Turbo decoding is both shortened in latency and more power efficient when the specialized MAP instruction found only on decoding MCOs are utilized for message passing decoding.

A key aspect of MCOs is that the assemblers and C-compilers are generated through correct by construction methodologies. The program can be changed quickly using high level language. Also, multi-cycle run-to-completion hardware accelerated FSM module instructions can be inserted into the instruction set, and external I/O ports are devoted to very-wide register files that contain late binding extensible instruction parameterizable after design completion. Specialized MCO instruction architectures components can be seen in Figure 4 and Figure 6. Multi-cycle instructions pass control of the MCO system state over to the finite state machine (FSM) in the control unit corresponding to the multi-cycle instruction. These modules can accelerate computation by using larger control words via an FSM. In addition they access unique complex arithmetic hardware in the ALU which remains dormant in normal single cycle instructions.

A second style of instruction is the extensible (external) instruction. This is a late-binding instruction. In the design of systems, we can defined the system parameters in a coordinated manner during the design phase (early binding) and or after the design has been completed (late binding). We note that late binding systems add significant levels of extensibility, but contain less architectural integration. Extensible instructions control and definition occurs from combinations of off-chip finite state machine systems and on-chip programmable register files. The register files contain very wide decoding bits that enable fine-grain control of the hardware resources of the system. The key for these instructions are they form a type of extensible instruction set that can be developed after the system design and development is complete. The price paid for this is the allocation of additional I/O pads, large memory width, and careful understanding of the register-level architecture.

MCOs are fixed point, but are enabled with quasi-floating point capability by means of extension scale registers that allow power of 2 shifting of data in ALU registers. This enables numerically sensitive operations to be performed with of nearly floating point precision, while fixed point operations can be performed for operations with low sensitivity to quantization noise associated with fixed point register affects.

B. Hierarchical System Model

The third level of Figure 2 communicates with the second level through virtual interfaces which abstracts level-3 firmware constraints from algorithmic parameters. Virtual interfaces are important to allow new hardware and firmware complexes to be installed without affecting system level software (i.e. design time modification flexibility). Examples of some of the algorithms integrated at the 3rd level of hierarchy as instantiated objects are detectors, precoders, MIMO space time decoders, FFTs, multi-carrier equalizers, and MAP decoders.

Finally, at level 4 in Figure 2, we incorporate communication layer virtual interfaces and communication object models representing the communication standard integrated into the software defined radio. Note that the SDR at the top level of Figure 2 could represent a 20MHz version of 3GPP-LTE using spaced time coding and 16QAM or 100MHz version of 3GPP-LTE Advanced [21] using spatial multiplexing and 64 QAM.

One challenge with broadband SDRs mentioned is the fact that we are typically margin constrained. This is due to the fact that the requirements for efficient, near-ergodic capacity algorithms (e.g. Turbo decoding, iterative equalization, Sphere-decoding, MLSE) [21] can virtually always be defined for superior sensitivity performance in a way that outstrips the computational limits of the SDR. To mitigate this, some MCOs are design for improved computational efficiency through hardware accelerated instructions, extensible instruction sets, and specialized multi-cycle instructions. These capabilities are indicated in the design architecture of the MCO control and ALU in Figure 4 and MCO architecture in Figure 6. The instruction set control-words of the extensible instruct sets are designed to be much larger than typical program words. The extensible instruction word-size is on the same order as the logic decoder generated control. This is also true of the hardware accelerated control logic. We rationalize the long control words of these units by noting that they are pre-designed to have maximum control over the entire set of internal registers, MUXs, memories, and ALU operators. For instance, extensible instructions are loaded into a RAM registers file prior to run time based upon the specifics of the new instructions that we desire to incorporate. We note that the addition of extensible instructions occurs *after design of the MCO*. It is therefore a late binding mechanism.

All instructions, whether multi-cycle or extensible, are integrated into the normal program instruction flow associated with program memory. Thus, a program instruction referring

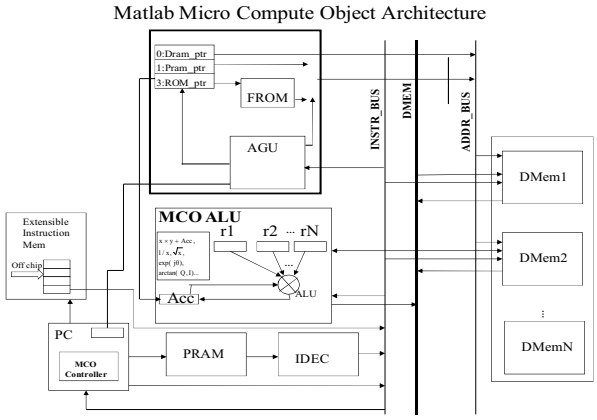

Figure 6: MCO Matlab Architecture

to an extensible instruction initiates a write of new logical control information from the extensible instruction memory that has been pre-stored in the extensible instruction register file. Multi-cycle, extensible programs could be formed by storing multiple single cycle extensible instructions references in particular sub-sections of extensible MCO program.

HW Accelerated multi-cycle instructions are integrated as a single instruction in main program memory. However, a reference to a multi-cycle instruction initiates a state change from the system level finite state machine (FSM) to the multi-cycle FSM machine. Note that multi-cycle instructions are all run to completion, but must be designed into the MCO at *design time*. In part, acceleration of the hardware occurs for extensible and multi-cycle instructions due to the finer control granularity capable over the entire set of register transfer level (RTL) resources available in the MCO. The ALU is optimized for a general set of mathematical computations (e.g. 1/sqrt(x), exp(jx), exp(x), atan(y,x)) and a differentiated set of instructions.

OBJECT ORIENTED SOFWTARE RADIO SIMULATION PRIMITIVES MODELS IN MATLAB

We have initiated development of object oriented primitive models of the MCO in Matlab using M-file classes. In normal SDR control operation, the MCO performs single cycle operations. In accelerated instruction mode, the MCO controller changes from its normal system state to the k'th accelerator state. This enables the k'th accelerator's control stream to control the entire hardware regime of the MCO hardware and allows the accelerator to access specialized ALU logic that increases the effective latency in clock cycles of the accelerator operation. The overall Matlab model in Figure 6 consists of an ALU, controller, program memory, multiple (N) data memories, instruction decoder, address generator unit, and the extensible instruction memory. In Matlab each MCO hardware module is defined via a class description. In turn, we create classes of high level primitive operations and protocol classes, each of which are allocate multiple MCOs as

illustrated in Figure 5. The protocol objects consisting of MCO elements are then instantiated and declared during the constructor call in Matlab. Our virtual processor model in Matlab can be easily integrated into our radio link simulation models along with SDR processor based computation of wireless communication algorithms or hardware abstracted functional versions. The merging of algorithms, hardware, and software in Matlab enable the creation of the virtual SDR system sample.

Each MCO contains a wide-width register file of memory in which extensible instruction capability can be externally loaded prior to run time or dynamically during run time. As shown in Figure 6, when the MCO controller recognizes the specialized extensible instruction the decode bits are fetched from the extensible instruction memory, which has been preloaded with the correct sequence of decode bits.

CONCLUSION

In this paper, we have described novel approaches to SDR definition by defining new metrics of SDR flexibility that lead to unique system level architectures. Our system level framework is optimized around a hybrid software-hardware entity which we define as a micro computer object (MCO). We note that our system is designed for both flexibility and extensibility. The bane of the optimization problems for broadband SDR is that we are always resource limited. Therefore, extensibility concepts, virtual abstraction, and acceleration hardware are important architectural features. We also discuss new simulation models in Matlab that lead toward the concept of the virtual system sample. In this framework, we completely merge hardware, software, algorithms, into one radio link system level objected oriented model.

A. Authors and Affiliations

Dr. Brian Kelley is with the University of Texas at San Antonio's (UTSA's) Department of Electrical and Computer Engineering (ECE). Dr. Kelley received his BSEE from Cornell University's College of EE and his MSEE/PhDEE from Georgia Tech, where he was an ONR and Georgia Tech Presidential Fellow. Dr. Kelley spent 10 years with Motorola's Advanced Cellular Research group. In 2007, Dr. Kelley joined the faculty of the University of Texas at San Antonio as an Assistant Professor in Communications. At UT-San Antonio, he has taught numerous graduate courses in communications and founded the Wireless Advanced Next Generation Laboratory (WANLab). Dr. Kelley's current research interests include Software Define Radio, 3GPP-LTE Cellular, Communications, and ad-hoc networks. He is a Senior Member of the IEEE, Chair of the San Antonio IEEE Communication and Signal Processing Chapter, and Associate Editor of Computers & Electrical Engineering, Elsevier, 2008. Has numerous publications, is a holder of 13 US patents or patents pending, and is a member of Tau Beta Pi and Eta Kappa Nu.

REFERENCES

[1] Joseph Mitola, "Software Radios Survey, Critical Evaluation, and Future Directions Technologies, "Software Radio Techn. Selected Read, IEEE Press, 1999.

[2] Brian Kelley, "Jointly Optimized Software Radio for Low Power 4G Systems," Invited Talk, Proceedings of the IEEE Asilomar Conference on Systems, Signals, and Computers, 2007.

[3] Alan C. Tribble, "The Software Defined Radio: Fact and Fiction," IEEE Radio and Wireless Symposium, 2008.

[4] John Boyd, Jennifer Schlenzig, "Directional Networks for Above 2GHz SDR," 2005 IEEE Milcom.

[5] Sangwon Seo, Trevor Mudge, Yuming Zhu, Chaitali Chakrabarti, "Design and Analysis of LDPC Decoders for Software Defined Radio," 2007 IEEE Workshop on Signal Processing.

[6] Y. Lin, H. Lee, M. Woh, Y Harel, S Mahke, C. Chakrabarti, K Flautne., "Soda: A low-power architecture for software radio," Proceedings of the 33rd Annual International Symposium on Comp. Arch. (ISCA), 2006.

[7] Chittabrata Ghosh and Dharma P. Agravwal, "Channel Assignment with Route Discovery (CARD) Using Cognitive Radio in Multi-channel Multi-radioWireless Mesh Networks", 2006 IEEE Conference on SDR.

[8] SDR Forum, www.sdrforum.org.

[9] Robert McLiece, The Theory of Information and Coding, Cambridge University Press; Subsequent edition, 1984.

[10] 3GPP Organization Website, www.3gpp.org

[11] Seungheon Hyeon, June Kim, Seungwon Choi, "Topics in Radio Communications: Evolution and Standardization of the Smart Anenna System for Software Defined Radio," IEEE Communication Magazine, September 2008

[12] LisaTek, http://www.coware.com

[13] http://www.ni.com/

[14] http://www.xilinx.com/products/

[15] N Cheung, J. Henkel, S. Parameswaran, "Rapid Configuration and Instruction Selection for an ASIP: Case Study", 2003 Design Automation and Test in Europe Conference and Exhibition, pages 802-807.

[16] Amitava Ghash, Rapeepat Ratasuk, Brian Classon, Vijay Nangia, Robert Love, Dale Schwent, David Wilson, "Uplink Control Channel Design for 3GPP LTE", 18th Annual IEEE International Symposium on Personal, Indoor and Mobile Radio Comm., September 2007.

[17] 3GPP TS 36.201: 3GPP TSG-RAN EUTRA LTE Physical Layer General Description, R8 WG1, September 2007.3GPP LTE

[18] 3GPP TS 36.211: 3GPP v8.0 TSG-RAN EUTRA Physical Channels and Modulation (Release 8), 2007.

[19] 3GPP TS 36.211: 3GPP TS 36.211 v1.2.0, 3GPP TSPG RAN Physical Channels and Modulation (Release 8).

[20] 3GPP TS 36.212: 3GPP TSG-RAN EUTRA LTE Multiplexing and Channel Coding, (Release 8), 2007.

[21] Brian Kelley, "3GPP Long Term Evolution Aspects and Migration to 4G Cellular Systems," 2009 IEEE Radio and Wireless Workshop on 3GPP LTE, San Diego, CA.

[22] Brian Kelley, "Software Defined Radio for Advance Gigabit Cellular Systems," to appear in DSP Handbook/Wireless, Networking, Radar, Sensor Array Processing, and Nonlinear Signal Processing, Chapter 22, 2009.