ENERGY-EFFICIENT SOFTWARE-DEFINED RADIO SOLUTIONS FOR MIMO-BASED BROADBAND COMMUNICATION

Bruno Bougard, Andre Bourdoux, Frederik Naessens, Miguel Glassee, Veerle Derudder, Liesbet Van der Perre

IMEC, Leuven, Belgium

{bougardb, bourdoux, naessens, glasseem, derudder, vdperre}@imec.be

Abstract

Multi-antenna transmission over multi-input, multi-output (MIMO) channels are considered in almost all recent broadband wireless communication standards. Besides, the fast-pacing diversity and evolution of those standards, next to the deep submicron integration cost explosion, urges multi-mode reconfigurable solutions. Software Defined Radio (SDR) is envisioned to enable low-cost, high-volume multi-mode baseband modems both for base-station and user terminals. Yet, supporting high-throughput MIMO standard with limited energy budget as in user terminals is a challenge for SDR architectures. With Space Division Multiplexing (SDM) for instance, N being the number of antennas, the computation load is multiplied by $>N^2$.

Capitalizing on a low complexity SDM-OFDM functional architecture, a heterogeneous multi-processor SoC platform with DSP cores delivering 50 to 250MOPS/mW and an integrated software development flow, we demonstrate the SDR implementation of 100Mbps+ SDM-OFDM with 3.6 nJ/bit energy efficiency (383mW average power).

1. INTRODUCTION

Because they enable significant throughput, capacity and/or robustness increase, multi-antenna transmission techniques are getting omnipresent in broadband wireless communication standards such as those ruling Wireless Local Area Networks (IEEE 802.11n), Wireless Broadband Access Networks (IEEE 802.16) and 3.5G/4G cellular networks (HSDPA, 3GPP-LTE).

Besides, the combination of the continuously growing variety of wireless standards and the increasing cost related to integrated circuit design and handset integration make implementation of wireless standards on reconfigurable radio platforms the only viable option in the near future [1]. An effective solution for reconfigurable baseband processing is known as software defined radio (SDR). The radio digital processing is there implemented as software components on a highly programmable platform.

The SDR concept, which has been so far restricted to the base-station segment, is slowly making its entry into handsets. SDR platforms are being proposed sustaining up to 5-10Mbps transmission with around 1W power consumption. However, current SDR solutions are still far from satisfying the computational requirements of 100Mbps+ multi-antenna based standards for which even direct VLSI implementation is still challenging [2].

To support those standards, broadband signaling and multi-antenna processing are jointly applied. In IEEE 802.11n specifically, space-division multiplexing (SDM) is combined with Orthogonal Frequency Division Multiplexing (OFDM). If we let N be the number of antennas both at transmit and receive side (2 to 4), the link throughput, when applying SDM, is optimally multiplied by N while the complexity is increased by a factor between N and N^2 – depending on the multi-stream detection algorithm – compared to single antenna OFDM transceivers. Processing load is hence multiplied by a factor N^2 to N^3.

Tackling the SDR implementation of such multi-antenna OFDM standards under stringent power constraint requires carefully selecting and optimizing the algorithmic options and the processing architecture parameters. The SDR platform must provide the required computing power at optimized energy efficiency. Besides, special attention must be paid to the software refinement flow, from functional modeling down to final platform mapping, so that the rough computing power offered by the SDR platform is efficiently utilized while development time is minimized.

In this paper, we first present low complexity algorithmic solutions for SDM-OFDM targeting the IEEE 802.11n standard, which has been identified to be the most demanding (section 2). Algorithms are selected and optimized to facilitate an SDR implementation. Next, based on the functional analysis of the required algorithms, a SDR platform architecture is proposed (section 3). The latter is designed to sustain the execution, next to IEEE 802.11n, of IEEE 802.16e and foresees provisions for future deployment of 3GPP-LTE. The refinement of the functionality and its mapping to the platform is then developed (section 4). Finally, the power consumption of the proposed platform when processing the targeted SDM-OFDM operation is discussed (section 4). Conclusions are drawn in section 5.

2. SDR-AWARE SDM-OFDM FUNCTIONALITY

Although the SDR solution proposed is targeted for multiple modes of operation (IEEE802.11n, IEEE802.16e and 3GPP-LTE), we focus the discussion on the IEEE 802.11n mode which has been identified to be the most demanding [1,3]. IEEE 802.11n recommends multi-antenna processing based on SDM, which is combined with OFDM on a carrier-per-carrier way. Although the standard foresees up to 4x4 SDM in up to 40 MHz channel (channel bonding), we focus, as a first step, on 2x2 transmission in a 20MHz channel. Each stream is encoded according to IEEE 802.11a, yielding a throughput of 108Mbps. The baseband platform is designed to operate with two direct-conversion analog transceivers, which means that 2 streams at 40 Msample/s must be processed for transmit or receive (2X over-sampling). The processing carried out on those streams can be broken down into 7 threads: automatic gain control, coarse, time-domain signal acquisition, fine time and frequency domain acquisition, channel estimation, tracking, multi-stream detection (post-filtering) and per-stream OFDM processing. Those threads are now detailed and the associated computation load evaluated. A “greenfield” preamble is considered [4].

Automatic Gain control

During AGC, the DC offset is first compensated for. Then, the analog front-end gain is set (via the front-end control interface) to optimize the ADC range. This is done during the first 3/10 of the

short training field (STF), which takes 2.4 µs. The load is <50OPs per STF, requiring <20MOPS.

Coarse acquisition

Error! Reference source not found. shows the block diagram of the auto-correlator used for coarse timing and frequency acquisition. This auto-correlation is performed on the first part (B-sequence) of the long training field (LTF). A double auto-correlator is used [5], which reduces the allowed maximum frequency offset but increases significantly the robustness of the acquisition. With a delay of 32 baseband samples, the acquisition range is ±1/(2x32*50ns)= ±312.5 kHz, which corresponds to ±60 ppm at 5.3 GHz.

The (time-domain) location of the peak magnitude at the output of the auto-correlator provides the estimate of the coarse timing while the slope of the auto-correlator output can be linked to the carrier frequency offset by:

$$\Delta f = \frac{2}{\pi} angle\left(\sum_{k=1}^{q} y^*(k-16)y(k) \right) \quad (1)$$

where *y(k)* are the incoming samples and *q* stands for the time index at which the auto-correlator output reaches its maximum. This auto-correlation requires *q* complex multiplications and q complex additions (*q* is at most equal to 16 times 7 = 112). In case the double correlator of Figure 1 is used, an additional (*q*-16) complex multiplications and (*q*-16) complex additions is required. An additional arctan operation is also needed to extract the frequency value. A *1°* accurate look-up table (LUT) is used to rotate the incoming samples prior to the FFT operation. For real time operations, the coarse acquisition must be executed during maximum 4µs (remainder of the STF minus reaction time of the front-end). This yields a computing load of around 500MOPS.

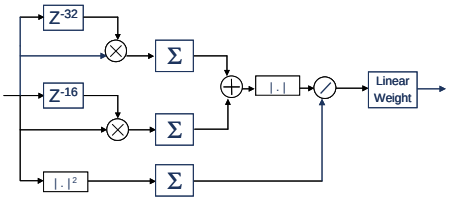

Figure 1 – Auto-correlation with double correlator for coarse synchronization

Fine acquisition

Fine acquisition of the carrier frequency offset is achieved by performing an autocorrelation on the LTF while fine acquisition of the timing offset is achieved by cross-correlating the received sequence with the ideal (known) LTF sequence as illustrated in Figure 2. The complexity of the fine carrier frequency estimation is similar to that of the coarse estimation.

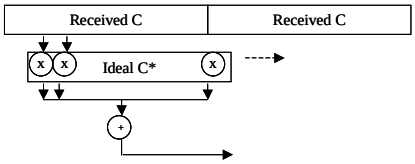

Figure 2 – Cross-correlation for fine timing synchronization

The cross-correlation is less sensitive than the autocorrelation to AWGN noise (since the reference is free of noise) but more sensitive to multipath (since the reference is not filtered by the channel). This makes peak detection delicate. Therefore, we use an adaptive threshold to detect the cross-correlator highest peak and use an additional shift of 4 samples in the backward direction to avoid any risk of ISI due to incorrect timing estimate. The cross correlation is a very costly operation since, for each incoming sample, N complex multiplications and additions must be carried out (N=64). The length (W) of the search window is therefore critical and can be selected to provide scalability. W is typically set to 11, yielding 704OPS.

Channel estimation

Our algorithm is based on the maximum likelihood estimator. The latter is based on preambles that use different subsets of sub-carriers on the two transmit antennas (note that this differs from the greenfield preamble definition in 802.11n [4] although this can easily be adapted to). Functionally, a smoothing and an interpolation are performed. A least-squares solution (maximum likelihood in the AWGN case) is searched with the time-domain constraint that the impulse response of an indoor wireless channel has a limited duration of typically a few 10s or 100s of nanoseconds. We use a constraint length (V) of 12 baseband samples, i.e., 12x50 = 600ns. This leaves some margin for timing inaccuracy. This constrained least square solution of the channel estimation is as follows:

$$\hat{h} = \begin{bmatrix} \hat{h}_T \\ \hat{h}_S \end{bmatrix} = \begin{bmatrix} DFT_T^V \\ DFT_S^V \end{bmatrix} \cdot DFT_T^{V\perp} \cdot h_T \quad (2)$$

where $\hat{h}$ is the channel estimated over all sub-carriers (index T stands for the transmitted sub-carriers and index S for the non-transmitted sub-carriers i.e. the sub-carriers used by the other antenna and the zero-carriers; superscript $^\perp$ stands for the pseudo-inverse), h_T is the frequency domain symbol of channel measurements on the transmitted sub-carriers and the DFT are blocks of the DFT matrices. More specifically, DFT_T^V is a subset of the DFT matrix given by the intersection of V first columns with the rows indexed by the indices in T.

The above equation involves two matrix multiplications. The size of the matrices are 52xV, Vx26 and 26x1, hence the total number of multiplications is Vx78. For V=12, this leads to 12x78=936 complex multiplications. In a 2x2 MIMO system, 4 such channel estimations need to be computed. Note that besides h_T, the other matrices in the above equation are constant and, hence, can be computed and optimized at design time. The same algorithm and hardware can also be used for single-antenna transmission and other channel estimation for instance for maximum ratio combining.

Tracking

An important choice was to set the tracking block before the space-time processing block in the receiver. In this case, the reference signal is obtained by multiplying the known pilots by the channel matrix. The main advantage of this solution is that the same tracking loop can be used with any MIMO scheme, even non-linear ones. This is a future-proof choice and a flexibility enabler (for different SDM receiver algorithms). This MIMO tracking loop (Figure 3) estimates a single value for the residual carrier frequency offset and the common rotation due to phase noise for all antennas, which is valid if a common local oscillator is used for both signal paths in the transmitter and in

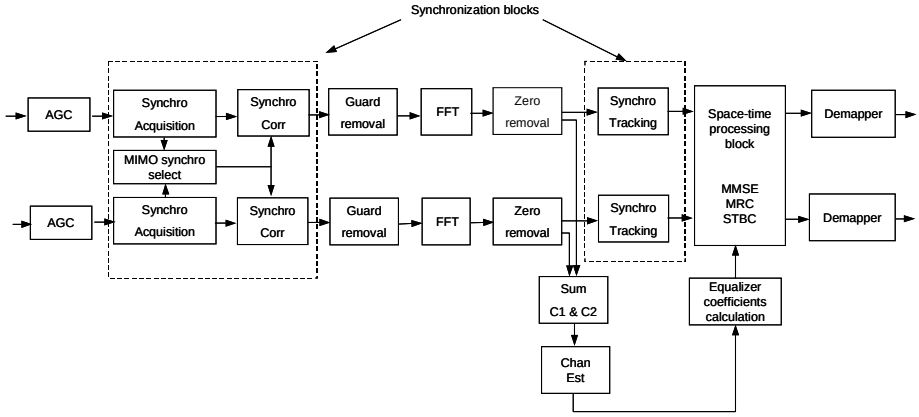

Figure 3 – Complete MIMO-RX Architecture

the receiver. The phase offset estimation itself accounts for 7 complex adds and 1 arctan (LUT). The residual CFO compensation rotates 96 symbols (LUT), hence totally ~100OPs.

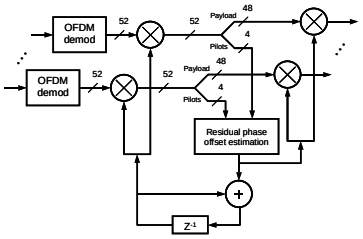

Figure 4 – Tracking loop

Multi-stream detection

The theoretical per- carrier system model is as follows:

$$\mathbf{y} = \mathbf{F} \cdot (\mathbf{H} \cdot \mathbf{x} + \mathbf{n}) \qquad (3)$$

where **y** is the signal vector after applying the linear receiver filter **F:**

$$\mathbf{F}_{\mathrm{MMSE}} = \left(\mathbf{H}^{\mathrm{H}}\mathbf{H} + \sigma^2 \mathbf{I}_{2x2}\right)^{-1}\mathbf{H}^{\mathrm{H}} \qquad \textbf{(4)}$$

3. LOW POWER HIGH PERFORMANCE SDR PLATFORM

The functional analysis carried out in section 2 shows that the computational requirements vary significantly between the different threads (20MOPS to 2.5GOPS). Besides, one may notice that the duty cycle of those threads are also very different. AGC is always active, even in idle mode. Coarse acquisition is active when the AGC loop detects power. Fine acquisition and channel estimation are only active during preamble receiving after a burst has been detected. Tracking, multi-stream detection and OFDM stream processing are active during the receiving of the packet data. Moreover, each thread may also have different requirements in terms of flexibility. This heterogeneity of duty/flexibility/load requirements fits well a heterogeneous Multi-Processor System-on-Chip (MPSOC) implementation approach [6-9]. Such MPSOC are generally articulated around a CPU to which slave processing entities (PE) are appended. PEs are typically designed keeping in mind the most important characteristics of wireless physical layer processing: *high data level parallelism* (DLP) and *data flow dominance*.

The platform considered in this work pushes forward the heterogeneity to better match the requirements of the functional threads identified above. As depicted on **Figure 5**, besides a ARM9 CPU and its peripherals, three types of PE are implemented with different programmability/peak computing power/energy-efficiency tradeoffs:

Baseband processing engine – Very long instruction word (VLIW) instruction set processors with SIMD (Single Instruction – Multiple Data) functional units are mostly considered to exploit the data level parallelism with limited instruction fetching overhead [3,9,10]. Besides, data flow dominance is often exploited in coarse grain reconfigurable arrays (CGA) [8,11]. We have formerly proposed a hybrid CGA-SIMD processor made of 16 densely interconnected 64-bit 4-way SIMD units with shared and distributed register-files [12]. The CGA is associated with a 4-bank data scratchpad (L1). It can be programmed from C based on the DRESC compiler framework [13]. A limited number of units can be operated in VLIW mode, accepting arbitrary C-compiled code (glue code) fetched through a 32K 128-bit wide instruction cache. When in array mode, C-compiled DSP kernels are executed while keeping configuration into local buffers that are configured through direct memory access (DMA). An AHB slave interface is provided for configuration and data exchange. Two such baseband processors are implemented in the platform. Each can sustain a theoretical computing load of 50GOPS and consumes 90mW in VLIW mode and 360mW in kernel mode, including memories and interfaces.

FEC accelerators – Forward error correction typically requires 10x more computing power than inner modem processing [3]. Moreover, it mostly relies on a limited set of well-known algorithms (Viterbi, RS, turbo, belief-propagation). Therefore, configurable application specific VLSI architectures are usually considered. Our platform embeds two FEC accelerators with configurable convolution encoder, Viterbi decoder, (de)scrambler and CRC calculation/check support.

Digital Front-End (DFE) – the computing power of the CGA baseband engine coupled to its high level of programmability makes it a good choice for fine acquisition, channel estimation, tracking, multi-stream detection and OFDM processing threads. However, for AGC and coarse acquisition, which have a high duty cycle, a lower computational load and lower flexibility requirements, higher energy efficiency cores are needed. Therefore, three digital front-end tiles are implemented. Each is associated with a signal path from a multi-antenna analog front-end. A single tile comprises a transmit section that buffer, over-sample and forward the I/Q samples to the ADC, and a receive section where packet

detection is implemented. The receiver path is supervised by a C-programmable *tile controller* (delivering 40MOPS @1,1mW) where AGC, DC offset compensation and power detection are implemented. When signal power is detected, the samples are down-sampled (specifically designed filters) and buffered. Besides, an assembly-programmable *detection engine* (delivering theoretically 5GOPS @20mW, [14]) is activated where coarse acquisition is implemented. A DFE tile generates an interrupt upon signal acquisition, waking up the platform. The latter then schedules the remaining threads using the baseband engines and the FEC accelerators.

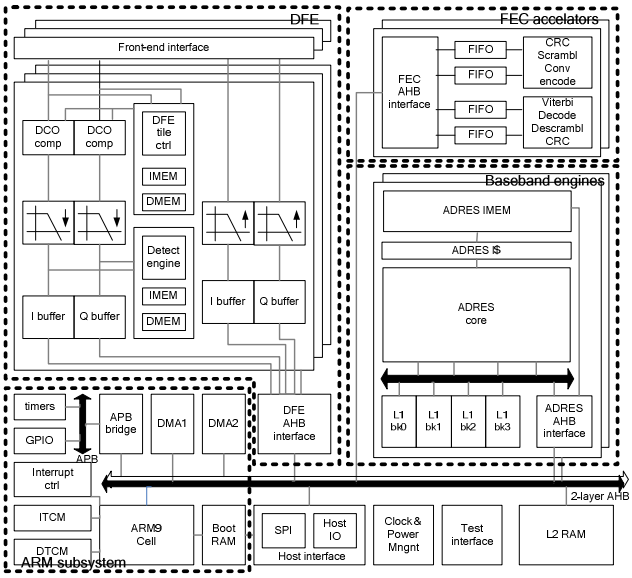

Figure 5 - SDR platform block diagram

4. SDR SOFTWARE REFINEMENT AND MAPPING

The functional and platform architectures being defined, the final step consists of assigning, mapping and scheduling the execution of the functional threads to and on the platform processing entities. Such task often reveals error prone and subject to efficiency loss. Therefore a strongly integrated successive refinement and back-annotation flow is needed (Figure **6**). One starts with an end-to-end executable specification of the functional architecture, written in Matlab language. The latter is used as reference for back-annotation.

A first refinement step, still in Matlab, consists of restructuring and pruning the functional model so that execution code and test-bench are clearly separated. Next, the execution code is translated to C. This is automated with Catalytic™ [15]. The resulting floating-point C code is then refined to fixed-point, which is supported by Catalytic™ through comprehensive statistical analysis and specific Matlab constructs. The tool automatically quantizes the variables that depend on already quantized ones. The resulting code is then optimized towards memory access reduction (data transfer and storage exploration, [16]).

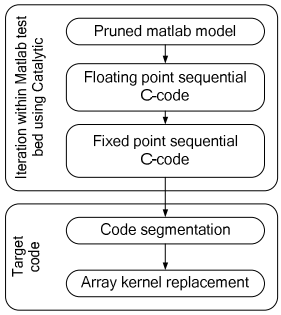

Figure 6 : Software flow from Matlab towards baseband processor optimized code

The first part of the flow yields sequential code that still has to be assigned, scheduled and mapped on the platform cores. The AGC thread is assigned to the *DFE tile controller*. Coarse acquisition is mapped on the *DFE detection engine*. The 5 remaining threads are mapped on the baseband engine. Those are first mapped targeting the VLIW mode. Next, their DSP kernels – namely, FFT, IFFT, cross-correlator bank, equalization and demapping – are optimized and mapped with DRESC for execution on the CGA.

To sustain the required processing load (up to 2.5GOPS) and provisioning sufficient margin in mapping efficiency due to the cost of high-level language programmability (directly from Matlab in our case), two baseband engines are used in parallel. The multi-processor execution of the 2x2 SDM-OFDM receiver is scheduled as depicted on **Figure 7**. A data payload of 228 symbols with 64QAM sub-carrier modulation is assumed, corresponding to a maximum length packet at a physical rate of 108Mbps.

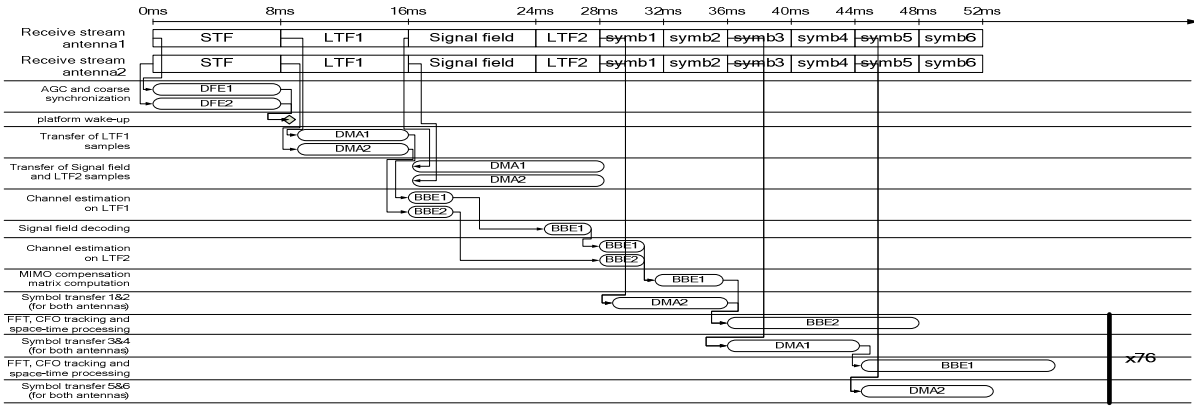

Figure 7 : Schedule of the SDM-OFDM receive threads execution onto multi processor platform

The schedule, transfers to and processing in the outer modem are not considered. During the STF, the AGC and coarse acquisition are performed, after which the rest of the platform is enabled via interrupt. The LTF is transferred to the baseband engines (BBE), through separate DMA transfers, allowing simultaneous channel estimation on both antenna streams. Meanwhile the signal field and second LTF are transferred too. The first BBE is decoding the signal field, after which the second LTF is used to perform an additional channel estimation. The channel estimates performed on BBE2 are transferred to BBE1 to determine the MIMO compensation matrix, during this task the first two data symbols are transferred to BBE2, which can start processing (FFT, CFO tracking and space-time processing) once the transfer is done. The DMA transfers and BBE processing can now occur in a pipelined manner, until the end of the packet (76 blocks of 3 symbols per processor).

The duty cycle of each processing entity during the reception of the burst can be analyzed from **Figure 7**. They are listed in Table 1 next to the core steady power consumption in active and standby mode obtained from design results. The baseband engines are observed executing 30% of the active time in VLIW mode and 70% in CGA mode. Assuming perfect power management (no significant start up energy and leakage control in sleep mode), the average power consumption of the platform during the reception of the frame can be estimated to 383mW, which correspond to 3.6nJ/bit.

TABLE 1 PES DUTY CYCLE AND POWER CONSUMPTION

PE	Duty	Power
DFE1	0.51%	21.1mW
DFE2	0.51%	21.1mW
ARM subsystem	99.49%	42mW
BBE1 VLIW	17.68%	90 mW
BBE1 CGA	41.20%	360 mW
BBE2 VLIW	17.85%	90 mW
BBE2 CGA	41.67%	260 mW
Average		**383 mW**

5. CONCLUSIONS

The SDR implementation of 100Mbps+ 2x2 SDM-OFDM has been presented. Capitalizing on a low complexity functional architecture, a heterogeneous MPSOC platform currently designed in 90nm CMOS and an integrated software development flow, real time operation with an average power consumption of only 383mW is achieved, corresponding to an energy efficiency of 3.6nJ/bit. This let us conclude that the considered platform is a good alternative to recent dedicated hardware solutions as in [2].

REFERENCES

[1] L. Van der Perre, B. Bougard, e. al., "Architectures and Circuits for Software defined Radios: Scaling and Scalability for Low Cost and Low Energy," *International Solid State Circuits Conference (ISSCC)*. San Francisco, CA, 2007.

[2] P. Petrus, Q. Sun, S. Ng, e. al., "An Integrated Draft 802.11n Compliant MIMO Baseband and MAC Processor," *International Solid State Circuit Conference*, IEEE, Ed. San Francisco, 2007.

[3] K. Van Berkel, H. F., P. Meuwissen, K. Moeren, M. Weiss, "Vector Processing as an Enabler for Software Defined Radio in Handsets for 3G+WLAN Onwards," *SDR Forum Technical Conference*, 2004, pp. 125-130.

[4] P. n. D1.10, "Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: Enhancements for Higher Throughput," January 2007 2007.

[5] A. Fort and W. Eberle, "Synchronization and AGC proposal for IEEE 802.11 a burst OFDM systems," Vol. 3, pp. 1335-1338 vol.13332003.

[6] B. Bougard, D. Novo, F. Naessens, L. Hollevoet, T. Schuster, M. Glassee, A. Dejonghe, L. Van der Perre, "A scalable programmable baseband platform for energy-efficient reactive software-defined radio," *Cognitive Radio Conference and related topics*, IEEE, Ed. Mykonos, Greece, 2006.

[7] G. Desoli and E. Filippi, "An outlook on the evolution of mobile terminals: from monolithic to modular multiradio, multiapplication platforms," *Circuits and Systems Magazine, IEEE*, Vol. 6, pp. 17-29, 2006.

[8] A. Lodi, A. Cappelli, M. Bocchi, C. Mucci, M. Innocenti, C. De Bartolomeis, L. Ciccarelli, R. Giansante, A. Deledda, F. Campi, M. Toma, R. Guerrieri, "XiSystem: a XiRisc-based SoC with reconfigurable IO module," *Solid-State Circuits, IEEE Journal of*, Vol. 41, pp. 85-96, 2006.

[9] L. Yuan, L. Hyunseok, M. Woh, Y. Harel, S. Mahlke, T. Mudge, C. Chakrabarti, K. Flautner, "SODA: A Low-power Architecture For Software Radio," pp. 89-1012006.

[10] J. Glossner, D. Iancu, L. Jin, E. Hokenek, M. Moudgill, "A software-defined communications baseband design," *Communications Magazine, IEEE*, Vol. 41, pp. 120-128, 2003.

[11] H. Singh, L. Ming-Hau, L. Guangming, F. J. Kurdahi, N. Bagherzadeh, E. M. Chaves Filho, "MorphoSys: an integrated reconfigurable system for data-parallel and computation-intensive applications," *Computers, IEEE Transactions on*, Vol. 49, pp. 465-481, 2000.

[12] D. B. Novo, B. Bougard, P. Raghavan, T. Schuster, H.-S. Kim, H. Yang, L. Van der Perre, "Energy-Performance Exploration of a CGA-based SDR Processor," *SDR Forum Technical Conference*. Orlando, FL., 2006.

[13] B. Mei, S. Vernalde, D. Verkest, H. De Man, R. Lauwereins, "Exploiting loop-level parallelism on coarse-grained reconfigurable architectures using modulo scheduling," *Computers and Digital Techniques, IEE Proceedings-*, Vol. 150, pp. 255-261, 2003.

[14] T. Schuster, B. Bougard, P. Raghavan, R. Prieswasser, L. Van der Perre, F. Catthoor, "Design of a Low Power Pre-Synchronization ASIP for Multimode SDR Terminals," *SAMOS Symposium*. Samos, Greece, 2007.

[15] http://www.catalyticinc.com/.

[16] K. Danckart, F. Catthoor, e. al., *Data Access and Storage Management for Embedded Programmable Processors*, Springer ed: Springer, 2002.