Algorithm-Architecture Co-Design for Energy Efficient Software Defined Radio Baseband

Min Li[†‡], David Novo[†‡], Bruno Bougard[†], Claude Desset[†], Antoine Dejonghe[†],
Liesbet Van Der Perre[†], Francky Catthoor[†‡]
[†] Nomadic Embedded System Division, IMEC, Leuven, Belgium
[‡] ESAT, K.U.Leuven, Leuven, Belgium
Email: {limin, novo, bougardb, desset, dejonghe, vdperre, catthoor}@imec.be

***Abstract*—The diversity and evolution of wireless communication standards are fast pacing. This requires a wide variety of baseband implementations within a short time-to-market. Besides, the deep sub-micron technology significantly increases design complexity and associated cost. These yield an increasing need for reconfigurable/programmable baseband solutions. Implementing all baseband functionalities on programmable architectures, as foreseen in the tier-2 SDR, will become a must. However, the energy efficiency of SDR baseband platforms is unavoidably worse than ASICs, this brings a challenging gap to bridge. The gap is broadening with the exploding baseband complexity. We advocate a system level approach to bridge the gap with a holistic view. First of all, we explicitly introduce architecture and compiler friendliness from the very beginning of the design flow, this enables highly efficient mapping on the targeted architecture. Furthermore, we fully leverage the advantages (programmability) of SDR platforms to compensate its disadvantages (energy efficiency). Highly flexible baseband implementations are developed to exploit the abundant dynamics in the environment and the user requirement to reduce energy consumption.**

I. Introduction

Nowadays, mobile devices are integrating an increasing variety of wireless communication standards, and each standard demands a multitude of different modes. Such a tremendous diversity, combined with the increasing development cost of deep sub-micron silicon, requires highly flexible baseband implementations. The Tier-2 SDR (Software Defined Radio) paradigm, where the entire baseband runs on programmable architectures, is attractive to obtain the desired flexibility.

Parallel ISP (Instruction Set Processor) based flexible baseband platforms have attracted extensive interest in recent years. However, such implementations typically come with a lower efficiency than traditional ASICs (Application Specific Integrated Circuits) baseband implementations. This efficiency gap remains to be bridged in order to make SDR more pervasive. Moreover, as the improvement of communication performances comes at the expense of significantly increased complexity, the gap is becoming even more challenging in the emerging high rate communication standards. Most SDR researches focus on reducing the gap by means of more efficient architectures. Several ASIPs (Application Specific Instruction Processors) and multi-processor architectures have been proposed with power consumption compatible with mobile devices. However, these are still far from being able to tackle the latest MIMO-OFDM (Multiple Input Multiple Output - Orthogonal Frequency Division Multiplexing) systems. Clearly, further disruptive innovations are still desired. Specifically, besides the extensive effort on processor level, more research is demanded at system level. Particularly, architecture oriented optimizations of the baseband itself remain to be exploited to achieve further efficiency improvement.

Our contribution is a system level design methodology for the implementation of advanced baseband processing on parallel ISP. The methodology explicitly takes into account the characteristics of parallel ISP. In most existing works, the implemented baseband is very similar to those in traditional baseband ASICs. Hence, due to the lower energy efficiency (GOPS/mW) of parallel ISP, SDR baseband implementations usually consume 2x to 10x more energy than equivalent ASICs. On the contrary, in our work the baseband explicitly targets parallel ISP. We design and optimize the baseband processing based on the thorough analysis of the constraints and opportunities of the new underlying architectures.

First of all, the baseband is designed to be compatible with the constraints imposed by parallel ISP and associated compilers. This enables maximum utilizations of parallel resources in the architecture. More importantly, we leverage the advantages of programmability as much as possible to compensate the disadvantages of ISP. Specifically, because ISP takes the programmability as a key design criterion, the multiplexing of data path and memory is much easier than ASIC. Hence, with ISP, the implementation cost of structure complexity at (macro-block level) is mostly increased code size overhead. The flexibility is an obvious advantage that we should exploit. We enable highly agile baseband implementations that can adapt to the dynamics in wireless communications to reduce energy consumption.

The remaining part of this paper consists of the following sections: section II presents the aspects of architecture/compiler friendliness and case studies; section III introduces how to enable agile baseband and case studies; section IV concludes the paper.

II. Introducing Architecture/Compiler Friendliness

ASIC implementations customize architectures to algorithms. Contrarily, in SDR, the parallel ISP is often given

with many specific constraints. In addition, the associated compiler imposes many constraints as well. For instance, SIMD (Single Instruction Multiple Data), vector and VLIW (Very Long Instruction Word) architectures all have very specific requirements on the data-flow structures. However, many important emerging baseband processing algorithms are inherently incompatible. If these problems are not eliminated at high level, low level compiler transformations hardly help. This results in inefficient resource utilizations on parallel ISP and hence low energy efficiency. We propose to introduce architecture/compiler friendliness from the very beginning of the design flow. Specifically, we combine mathematic/algorithms transformations and compiler transformations to derive data flow and control flow structures, which can easily bring efficient mappings on architectures. Note that in our work the aforementioned transformations may be I/O approximate.

We have demonstrated this on many key baseband algorithm blocks in MIMO-OFDM systems. For instance, a partial FFT based OFDMA modualtor/demodulator on ILP architectures, a near-ML MIMO detector on ILP/DLP architectures, a soft MIMO detector on ILP/DLP architectures, a sparse-matrix multiplication based OFDM channel estimator implementation on ILP architectures. In the following we will briefly introduce three of these case studies.

A. Near-ML MIMO Detector on ILP/DLP Architecture

When applying MIMO transmissions, the remarkable throughput improvement comes at the cost of significantly increased receiver complexity. With SDM (Spatial Division Multiplexing) transmissions, the major complexity increment is in the MIMO detector. Among existing MIMO detectors, the ML (Maximum-Likelihood) and near-ML detectors are superior to traditional linear detectors. In recent years, the algorithmic optimizations and implementations of ML/nearl-ML detector have attracted lots of interest. Almost all of the implementations are delivered in ASIC or FPGA (Field Programmable Gate Array), but not on programmable architectures.

Our contribution is to co-optimize the algorithm and implementation for scalable near-ML MIMO detector on *parallel programmable baseband architectures*, such as the DSPs (Digital Signal Processors) with VLIW, SIMD or vector processing features.

Unfortunately, none of the existing near-ML detectors fit ILP and DLP architectures well. Sphere decoders and most of its variants [1] are essentially sequential and non-deterministic, so that the the parallelization is difficult. On the other hand, although K-Best, QRD-M and their variants have been realized in several ASIC-style implementations, they have a fundamental problems when mapping on ILP and DLP architectures: The spanning-sorting-deleting process incurs irregular dataflow, non-deterministic control flow, extensive shuffling and extensive memory-rearrangement. These characteristics will result in very low resource-utilizations on ILP and DLP architectures. If these problems are not eliminated at high-level, low-level compiler optimizations can hardly solve them.

In order to bridge the algorithm/architecture gap, we bring-in explicit architecture-friendliness from the very beginning of the design flow. In early steps, high-level algorithmic transformations make the dataflow structure fit architectures very well. We enable abundant *vector parallelism* in our proposed SSFE (Selective Spanning with Fast Enumeration) detector; memory rearrangement, shuffling and non-deterministic dynamism are all excluded. Comparing to the ASIC-minded K-Best, the SSFE not only significantly reduces the algorithmic complexity, but also results in a completely regular and deterministic dataflow structure. Hence, it can be easily parallelized and efficiently mapped onto not only ILP but also DLP (SIMD, vector) architectures. Furthermore, we apply comprehensive pre-compiler transformations with the help of application-level information. Control-flow transformations, loop transformations, strength reductions, algebraic simplifications, common-expression reductions, variable-expansions are performed to optimize not only computation-operations but also address-generations and memory-accesses. Details of this work can be found in [2].

B. Partial FFT on ILP Architecture

The FFT (Fast Fourier Transform) is undoubtedly one of the most important and fundamental techniques in the signal processing world. Among the countless hardware and software implementations, the vast majority optimize their results for the case when all input and output bins are needed. However, for many real life applications this is *not* the case. For instance, the multi-resolution partial spectrum analysis; the modulation and demodulation in the emerging OFDMA (Orthogonal Frequency Division Multiplexing/Multiple Access) systems; FFT based interpolations; computation of a high-resolution eigenvalue spectrum in array signal processing. These partial input/output cases are crucial for future wireless communication systems [3], such as IEEE802.11x, IEEE802.16 and 3GPP LTE.

As we do not want to pay the price of a full FFT for only part of the input/output bins, the redundant data flow in the aforementioned cases can be pruned *without* changing I/O behaviors. This is called a *PFFT* (Partial FFT). In many applications the percentage of required input/output bins is very small. For instance, in the 3GPP LTE (an important candidate for air interfaces beyond 3.5G), if the OFMDA symbol size is 1024 (corresponding to 10MHz bandwidth mode) and 12 users equally share the available 600 sub-carriers, only 50 of the 1024 FFT output bins (4.88%) are required for each active user [4]. Hence, the computations, memory accesses and shuffling operations saved by the PFFT are very attractive in practical situations.

Although research on the PFFT can be traced back for more than 30 years [5] and the theoretical aspects of the PFFT have been thoroughly studied, there were rarely PFFT implementation breakthroughs. We find very few papers about implementations of PFFT in real life systems. The most important obstacle is that the full regularity in a FFT is destroyed in a PFFT. Hence, efficient implementations on

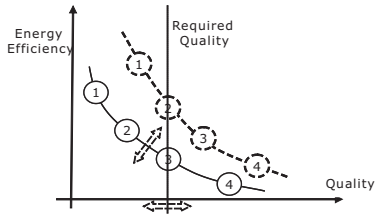


Fig. 1. The dynamics in the user requirement and the environment

parallel architectures become very difficult. In addition, a size-N PFFT has 2^N possibilities regarding the input or output patterns. A case-by-case optimization approach is not feasible. Clearly, delivering both highly optimized flexibility and efficiency in the same implementation is very difficult. This challenge is imposed not only on hardware implementations but also software implementations on parallel programmable platforms, such as ILP architectures.

We proposed a *generic* method to efficiently map the irregular data flow of an arbitrary PFFT onto ILP architectures with SWP (SoftWare Pipelining). Our work exploits the available opportunities while minimizing the problems that the constraints introduce in FFT algorithms, ILP architectures and their interactions. We apply optimization techniques covering both algorithmic aspects and low level implementation schemes. Most importantly, at algorithmic level, we select an appropriate PFFT data flow variant to enable aggressive optimizations in subsequent implementation steps. Then, we apply a divide and conquer strategy, partitioning the PFFT into three phases to enable *heterogeneous* implementations. For each of the phases, control flow structures, SWP schemes, address generation schemes and memory operations are specialized. For a specific PFFT instance, the data flow information is encoded in a compact table that is used for control operations, so that the desired flexibility is delivered. Our proposal achieves significant reductions in terms of cycle count, number of active instructions and memory activities. To the best of our knowledge, this is the first reported work about the generic software pipelined PFFT on ILP architectures. More details of the work can be found in [7]

III. Introducing Agile Baseband with Dynamic Scalability

The user requirement in wireless communications is inherently varying, when propagating the property to baseband, the required quality of baseband processing is varying as well. Since a programmable platform offers easy and flexible multiplexing of data-path and memory, we propose to implement the baseband in a flexible way, so that the baseband can tradeoff the energy efficiency and the quality of processing. In this way, the baseband implementation can dynamically switches at run time and work with maximized energy efficiency. This is illustrated in Fig.1, where the required quality is marked with a vertical line. We show a simple example with option 1-4 on a Pareto-optimal curve: the option 1 offers the highest energy efficiency but the lowest quality; the option 4 offers the highest quality but the lowest energy efficiency. When the required quality shifts, we can track the change and switch to another energy optimal option. Such a scheme requires a flexible trade off between the energy efficiency and the quality of processing, which is define as the *scalability*. Implementing a scalable baseband on programmable SDR platforms is much easier than on an ASIC.

Besides, the environment is varying as well. This influences the systems and offers further opportunities. Specifically, the environment dynamics shifts the tradeoff curve of scalable baseband implementations. In Fig.1, this is illustrated with a double-side arrow on the curve. A possible shift is shown with dashed lines. In this situation, if the quality requirement remains the same, we can switch from option 3 to option 2, which is more energy efficient. For example, a channel estimator tracks varying wireless channels. if a mobile is moving with a vehicle at a high speed, an energy hungry equalizer (e.g., option 3) is required to track the intensive channel variations. If the mobile gets out from the vehicle and moves at a pedestrian speed, the tradeoff curve will shifts as Fig.1. A lower power equalizer implementation (option 2) is already enough. In order to optimize the energy efficiency in such way, a run-time controller is required to monitor the environment dynamics and switch the mode of scalable implementations.

From the above discussions, we can clearly see that the following design steps are required:

- Enabling scalability. This is to enable baseband implementations that allow a flexible tradeoff between the energy efficiency and the quality of processing.
- Designing a run time controller, which makes decision on which mode to use for the scalable baseband implementations enabled in the first step.

We have applied the design to most key components of the inner modem in both traditional CDMA based systems, emerging MIMO-OFDM and MIMO-OFDMA systems, including channel estimator/equalizer, channelization filters, OFDMA modualtor/demodulator, advanced MIMO decoders and so on. For the sake of limited space, we will illustrate 3 representative ones in this paper.

A. OFDMA Modulator

First we start from a simple example: an agile OFDMA and MIMO-OFDMA modulator implementation. It is very important for emerging WiMAX and 3GPP LTE systems. In MIMO-OFDMA transmitters, the modulator becomes one of the most energy-consuming parts. Despite that IFFT/FFT have long been considered as one of the most efficient signal processing primitives, The modulator implementation is still challenging. In this case-study, the scalability is enabled with down-sampled processing. We exploit that fact that the required modulation quality is dynamically changing.

1) Enabling Scalability: As scheme #1 in Fig.2, the original OFDMA modulator is just a size-N IFFT with only M of N input bins being non-zeros. Note that N is large (up to 2048) and M is small, e.g., M=25. We transform scheme #1

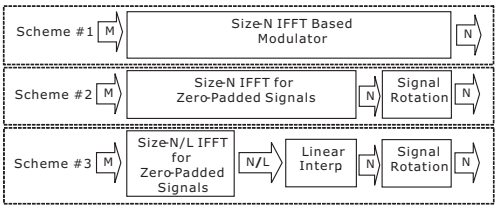


Fig. 2. Transformations for the agile OFDMA modulator implementation

TABLE I
RUN-TIME DECISION AND ENERGY EFFICIENCY IMPROVEMENT

Modulation (m)	BPSK	QPSK	16QAM	16QAM	64QAM	64QAM
Coding Rate (c)	1/2	1/2	3/4	1/2	2/3	3/4
% of carriers (p)	8%	8%	8%	8%	8%	8%
RCE Margin (d)	5 dB	5 dB	15 dB	10 dB	5 dB	5 dB
L	16	16	8	8	8	4
Energy Efficiency	12×	12×	7×	7×	7×	2×

to #2, consisting of a zero-padded IFFT (with nonzero data starting from the first input bin) and a signal rotator. With this transformation, the signal before the rotator becomes highly correlated. Hence, we can further apply an I/O-approximate transformation and derive scheme #3, which use a size-N/L IFFT and an efficient N/L to N piece-wise linear Cartesian interpolator to approximate the size-N zero-padded IFFT in scheme #2. The interpolation factor L brings the scalability. Larger L brings lower energy consumption. Importantly, the blocks in scheme #3 can all be efficiently mapped on parallel programmable architectures.

2) Designing Controller: The modulation accuracy is considered as the quality metric. Let m denote the modulation, c denote the coding, p denote the amount of allocated subcarriers. Then (m, c, p) specify the input signal and L is the knob of the scalability. Importantly, when (m, c, p) and L is given, the modulation accuracy is statistically stable and analyzable. Hence, based on the model, we propose a feed forward scheme with a controller based on loop-up table. The decision policy is $\Psi : (m, c, p, d) \longmapsto L$, where d is an indication of the required modulation accuracy. With input characterized by (m, c, p) (this is completely observable), the controller selects the right L that satisfies the required accuracy d. The overhead of this controller is just a table lookup operation.

3) Implementation Results: Targeting reproducible results, the work has been implemented on the TI TMS320C6000. We evaluate the implementation for WiMAX, which defines the RCE (Relative Constellation Error) to evaluate the modulation accuracy. We need to guarantee that the modulation accuracy has a safe margin to the minimal requirements specified in WiMAX standards. The margin is denoted by d. With L=$\{2,4,8,16\}$, 2× to 12× energy efficiency improvements can be achieved with this work. Some examples are given in Table I. We can see substantial improvements even with large RCE margins. More details about the work can be found in [8].

B. Near-ML MIMO Decoder

Near-ML (Maximum Likelihood) MIMO decoders significantly outperform traditional linear decoders. However, their implementation is challenging on programmable architectures. We present an agile near-ML decoder specifically designed for programmable architectures. With reproducible results on TI TMS320C6416 it delivers 80-103 Mbps throughput (4×4,64QAM). To the best of our knowledge, this is the first programmable implementation having throughput compatible with emerging standards (3GPP LTE, WiMAX, 802.11n). The key enabler is that our work exploits the dynamics in both the environment and the user requirement.

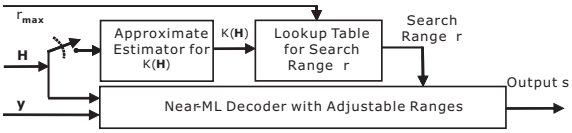


Fig. 3. Framework of agile MIMO decoder implementations

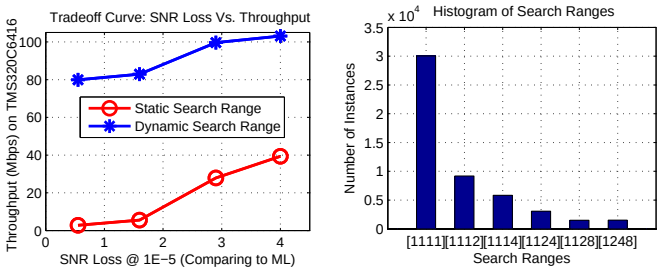
1) Enabling Scalability: The ML MIMO decoding is defined as $\hat{s} = \arg\min_{s \in \Omega^{Nt}}, \|\mathbf{y} - \mathbf{H}\mathbf{s}\|^2$, where $\mathbf{H}$ is the channel matrix, $\mathbf{y}$ is the received vector signal, Ω^{Nt} is contains all possibilities of $\mathbf{s}$. Most near-ML decoders are based on a tree-search in a smartly selected subset of Ω^{Nt}. Importantly, the size of the sub-set can be adjusted. Adjusting the search range enables the desired scalability.

2) Designing Controller: The dynamic environment shifts the tradeoff curve (refer to Fig.1). For near-ML MIMO decoders, we show that the numerical properties of $\mathbf{H}$ have a significant impact on the curve. If $\mathbf{H}$ is well conditioned with small condition number $\kappa(\mathbf{H})$, a small search range already brings a good quality. On the contrary, with large $\kappa(\mathbf{H})$, a large search range is necessary for a good quality. Hence, we consider the condition number $\kappa(\mathbf{H})$ as the observed variable.

We define the quality metric as the SNR (Signal to Noise Ratio) loss comparing to the optimal ML decoder @ BER (Bit Error Rate) 10^{-5}. In this case, we found that the relation among the observed variable ($\kappa(\mathbf{H})$), the controlling variable (search range), and resulted quality metric (SNR loss) is also statistically stable and analyzable. Hence, we propose an I/O model based feed forward controller to adjust search range according to $\kappa(\mathbf{H})$ and the required quality. This is illustrated in Fig.3. However, in this case $\kappa(\mathbf{H})$ is not readily observable, an efficient and effective estimator is required. We apply an approximate iterative estimator with only 2 iterations. In addition, we applies a down-sampled processing, as shown in Fig.3 with a switch for $\mathbf{H}$. The estimator is activated only at the beginning of a burst processing. In this way, the overhead of the estimator becomes negligible (below 1%).

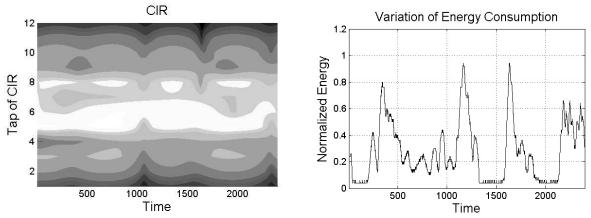
A efficient decision policy for search range r: $\Psi : \kappa(\mathbf{H}) \longmapsto r$ is proposed. The controller selects a search range between the minimal one r_{min} and the maximally allowed one r_{max}. r_{max} is a given parameter as an indication of required quality. To minimize the overhead, we apply a lookup table for $\Psi : \kappa(\mathbf{H}) \longmapsto r$. The table is calibrated at design time to ensure that the proposed decoder with dynamically adjusted search ranges is only 0.1 dB better than a static decoder.

3) Implementation Results: We apply the above generic design to the SSFE (Selective Spanning with Fast Enumer-



(a) (b)

Fig. 4. Results. (a) Comparison of throughput-quality tradeoffs; (b) Histogram of search ranges for practical channel models



(a) (b)

Fig. 6. Channel dynamism and energy consumption of the equalizer. (a) Average speed 50km/h; (b) Average speed 120km/h. The equalizer consumes much energy only when channel is varying intensively

ation) near-ML decoder, which is the first near-ML decoder specifically optimized for parallel programmable architectures [9][2]. It has been implemented on TI TMS320C6416 for reproducible results. In Fig.4 (a), the X-axis is the SNR loss as introduced before. The Y-axis is the average throughput. The results is for 1/2 Turbo-coded 4×4 $64QAM$ transmissions over 3GPP suburban macro channels. We can see that, with close SNR losses, the agile SSFE decoder significantly boosts the throughput compared to the static SSFE. The improvement factor ranges from 2.6× to 28.6×. This translates into remarkable reductions of energy consumption. The histogram of search ranges in Fig.4 (b) gives more insights, where strings $\{[1111], \ldots, [1248]\}$ indicate search ranges, number of spanned nodes of each antennas (4 in total) is described by the corresponding digit in the strings. $[1111]$ is r_{min} and $[1248]$ is r_{max}. The histogram shows that the agile SSFE decoder mostly searches over small ranges (such as $[1111]$ and $[1112]$). Moreover, we can observe that the agile implementation can trade off throughput and quality. If a low quality is tolerable, we can switch r_{max} to a smaller one for higher energy efficiency.

C. Adaptive Filter Implementation

1) Overview: An adaptive filter is a filter that self adjusts its transfer function (or coefficients). It has a wide spectrum of important applications in baseband processing. For example: equalization/channel estimation, signal prediction, interference cancelation. We introduce a generic framework for energy efficient agile adaptive filter implementations, and validate it with adaptive equalization for HSDPA. Our work exploits the dynamics of variations in the process tracked by an adaptive filter. For instance, the wireless channel is not always varying fast. With a pedestrian channel, we can switch to a lower power mode while still delivering the required quality. In this case study, we design a feed back controller without an explicit estimator for the environment.

2) Enabling Scalability: The technique for enabling scalability is illustrated in Fig.5. In the shaded area, two algorithms are multiplexed. In addition, we can also set the alg.#2 just as a direct link that does *no* updating. By adjusting how often alg.#1 is selected, we can enable the scalability. Importantly, in this way we implement only one algorithm (alg.#1) but the scalability is still enabled. The speed of variations in the environment shifts the curve at at run time. For example, an adaptive equalizer tracks a varying wireless channel. If the mobile is moving with a vehicle at 120km/h, we need to select alg.#1 frequent. If the mobile gets out from the vehicle and moves at a pedestrian speed, a longer updating interval is enough to deliver the required quality.

3) Designing Controller: In order to exploit the aforementioned opportunities, the controller needs to track the dynamics in variations. For such statistical information, an explicit estimator incurs significant complexity. In addition, it does not necessarily converge in a practical system with high order complex stochastic behaviors (such as the wireless channel).

With the feed back control, we propose a generic and efficient technique without any explicit estimator for the dynamics. As shown in Fig.5, besides the feed back loop for coefficient updating (shaded area), an extra feed back loop is built to control the tracking error, which is the error between the desired output (training signal) and the actual output of the filter. An efficient PID controller is applied to keep the tracking error around a predefined set-point. It adjusts how often the alg.#1 is selected for updating filter coefficients. Since the PID controller requires only a few multiplications and additions, the overhead is negligible compared to the adaptive filter itself. In this controller, the controlling variable is the updating interval, the controlled variable is the tracking error, and the observed variable is the tracking error as well.

4) Implementation Results: The above design is very attractive for channel equalizer, because wireless channels have complex stochastic behaviors. We have demonstrated this for HSDPA chip level equalizer, which is one of the most energy consuming parts of HSDPA receivers. This has been implemented on TI TMS320C6713. First, we study the dynamic behavior of the work. Fig.6 (a) shows how the CIR (Channel Input Response) varies over time in a 3GPP channel. Fig.6 (b) plots the normalized energy consumption over time. We can observe that the energy consumption fluctuates intensively. Comparing the X-axis (time) of Fig.6 (a) and (b), we can see that this work consumes much energy only when CIR varies rapidly. Second, we study the potentials of energy reductions. In Fig.7 we plot results for 3GPP channel case 5. Each curve consists of points with different tracking errors. Uncoded BER (Y-axis) is considered as the quality metric. A

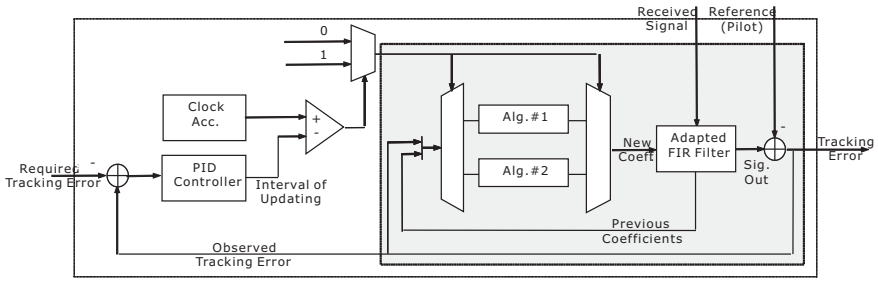


Fig. 5. A generic framework for agile adaptive filter implementations

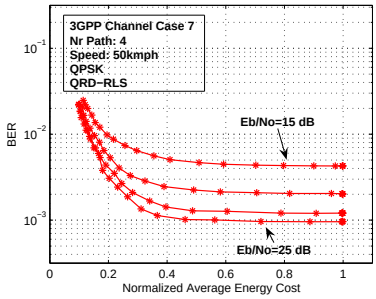


Fig. 7. Quality/Energy tradeoff

conventional static equalizer is characterized by the rightmost points on these curves. We can observe that, with negligible BER degradations, the design can reduce more than 60% energy comparing to the conventional static equalizer. More details of the work can be found in [10][11].

IV. Conclusion

Although extensive research efforts have been investigated in energy efficient SDR baseband, most work focus on processors/platforms. System level approaches are less mature. In this paper, we introduced the algorithm/architecture co-design for SDR baseband implementations. Such an approach ensures that algorithm and architecture match with each other and the energy efficiency is significantly optimized.

References

[1] A. Burg, M. Borgmann, M. Wenk, M. Zellweger, W. Fichtner, and H. Bolcskei. Vlsi implementation of mimo detection using the sphere decoding algorithm. *Solid-State Circuits, IEEE Journal of*, 40(7):1566–1577, July 2005.

[2] Min Li, Bruno Bougard, Weiyu Xu, David Novo, Liesbet Van Der Perre, and Francky Catthoor. The optimization of near-ml mimo detector for sdr baseband on parallel programmable architectures. *The IEEE/ACM Design Automation and Test in Europe (DATE) 2008*, pages 1–6, Nov. 2006.

[3] Zhong Hu and Honghui Wan. A novel generic fast fourier transform pruning technique and complexity analysis. *Signal Processing, IEEE Transactions on [see also Acoustics, Speech, and Signal Processing, IEEE Transactions on]*, 53(1):274–282, Jan. 2005.

[4] 3gpp lte tr 25.814: Physical layer aspects for e-utra. *3GPP LTE specifications*, 2007.

[5] J. Markel. Fft pruning. *Audio and Electroacoustics, IEEE Transactions on*, 19(4):305–311, Dec 1971.

[6] Vicki H. Allan, Reese B. Jones, Randall M. Lee, and Stephen J. Allan. Software pipelining. *ACM Comput. Surv.*, 27(3):367–432, 1995.

[7] Min Li, David Novo, Bruno Bougard, Liesbet Van Der Perre, and Francky Catthoor. Generic multi-phase software-pipelined partial-fft on instruction-level-parallel architectures and sdr baseband applications. *The IEEE/ACM Design Automation and Test in Europe (DATE) 2008*, pages 1–6, Nov. 2006.

[8] Min Li, B. Bougard, E. Lopez-Estraviz, A. Bourdoux, L. Van Der Perre, and F. Catthoor. The quality-energy scalable ofdma modulation for low power transmitter and vliw processor based implementation. *Global Telecommunications Conference, 2007. GLOBECOM '07. IEEE*, pages 2894–2898, 26-30 Nov. 2007.

[9] Min Li, Bruno Bougard, Weiyu Xu, David Novo, Liesbet Van Der Perre, and Francky Catthoor. Selective spanning with fast enumeration: A near-ml mimo detector designed for parallel programmable architectures. *The IEEE International Conference on Communications (ICC) 2008*, pages 1–6, Nov. 2006.

[10] Min Li, Bruno Bougard, Francois Horlin, Marc Engels, Liesbet Van Der Perre, and Francky Catthoor. Spc04-1: Quality-energy scalable chip level equalization for hsdpa. *Global Telecommunications Conference, 2006. GLOBECOM '06. IEEE*, pages 1–6, Nov. 2006.

[11] M. Li, B. Bougard, and F. Catthoor. Exploit multiple-domain sparseness for hsdpa chip level equalization in sdr: Algorithm and dsp implementation. *Signal Processing Systems Design and Implementation, 2006. SIPS '06. IEEE Workshop on*, pages 16–21, Oct. 2006.