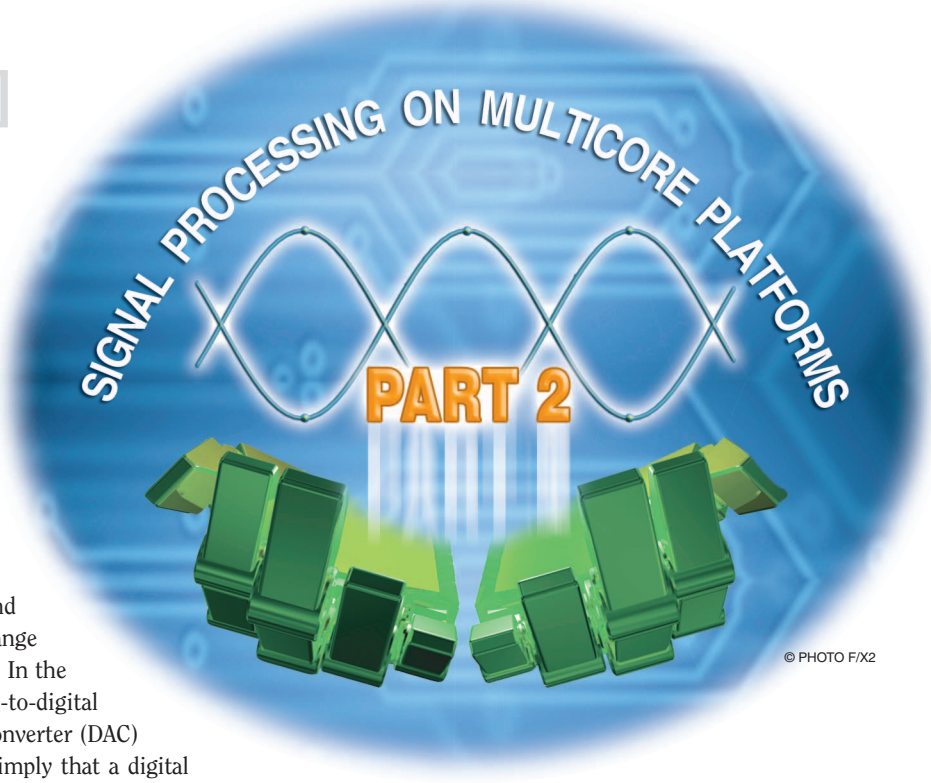
Martin Palkovic, Praveen Raghavan, Min Li,
Antoine Dejonghe, Liesbet Van der Perre, and Francky Catthoor

Future Software-Defined Radio Platforms and Mapping Flows

An overview of their multicore architectures



A software-defined radio (SDR) system is a radio communication system in which physical layer components are implemented on a programmable or reconfigurable platform. The modulation and demodulation is performed in software and thus the radio is able to support a broad range of frequencies and functions concurrently. In the ideal SDR transceiver scheme, an analog-to-digital converter (ADC) and a digital-to-analog converter (DAC) are attached to the antenna. This would imply that a digital signal processor (DSP) is connected to the ADC and the DAC, directly performing signal processing for the streams of data from/to antenna [1]. Today, the ideal SDR transceiver scheme is still not feasible and thus some processing has to happen in the reconfigurable analog front end [2].

INTRODUCTION

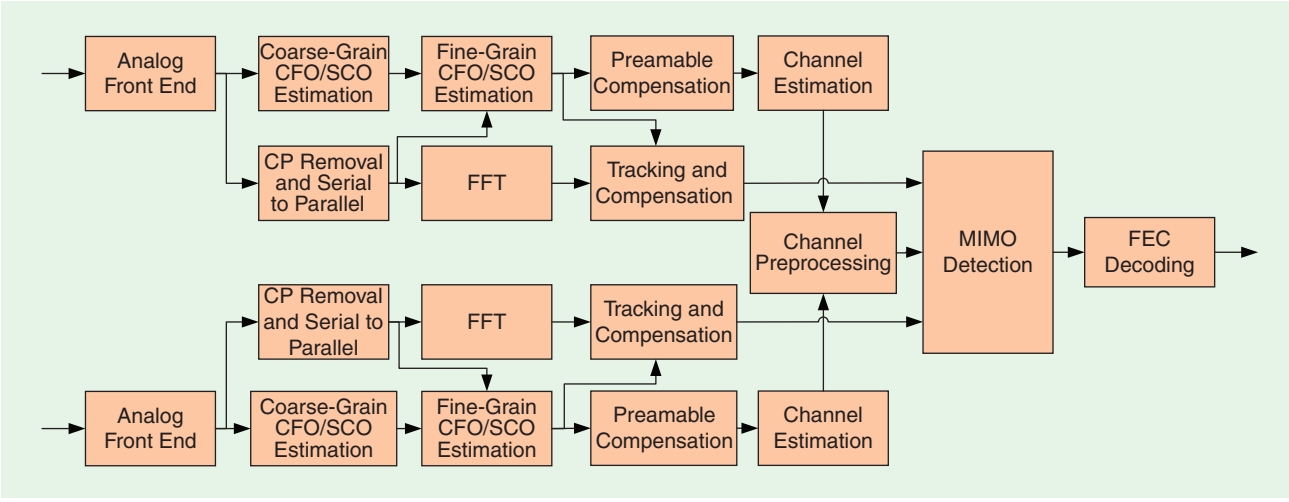
In the past, SDR was mainly attractive for the military and wireless infrastructure segments. Recently, the SDR paradigm has also entered into the consumer electronics segment. This is driven by three main factors. First, the soaring chip development cost and respin rate in deep submicro era has driven chip vendors to share the development costs across product lines. Second, extremely diversified market demand for different wireless standards has triggered the need for an ideal mobile terminal to support these standards (from cellular to broadcasting) within a tight cost budget. Third, the fast evolution of wireless standards has caused a shorter time-to-market, which makes a programmable SDR solution attractive.

SDR: THE NEED FOR MULTICORES AND WHAT IT BRINGS

The physical layer of a typical radio transceiver consists of an inner modem and an outer modem. The inner modem contains

Digital Object Identifier 10.1109/MSP.2009.935386

1053-5888/10/$26.00©2010IEEE

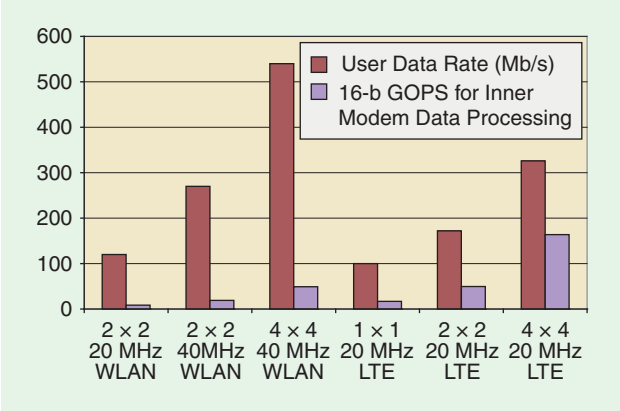


[FIG1] **Block diagram of typical WLAN RX functionality.**

transmission and environment parameters estimation (including various acquisition and tracking tasks) and data detection (e.g., demapping and multiantenna detection), while the outer modem mostly performs the decoding/coding of the received/transmitted stream of data (see Figure 1). On top of these two compute-intensive functionalities, multiple access control (MAC) functionalities is also needed to ensure appropriate timing of the operations and appropriate acknowledgment schemes. The various functionalities that are present in a radio transceiver need different types of signal processing tasks and have different duty cycles. Furthermore the core computation of these different functionalities also vary substantially. For example, the inner modem requires mostly a fairly irregular computation with a large variation across the different standards, whereas the outer modem computation is a much more regular computation that needs limited flexibility. Each of these blocks also exhibit different types of parallelism (data level, instruction level, and task level). To reach maximal energy efficiency it is more efficient to tune one (or more) cores to a given functionality rather than to make a very flexible core. Various research works such as [3]–[5] have pointed towards optimally adapting the processor to make an application-specific instruction-set processor (ASIP). This gives almost an order of magnitude difference in the energy efficiency compared to a DSP or a reduced instruction set computer (RISC). Given the above reasoning, it is quite evident that for a power efficient platform there exists a need for a heterogeneous multicore solution.

Furthermore within a single standard [e.g., wireless local area network (WLAN)] multiple modes exist. Each mode may require different signal processing tasks (e.g., different multiantenna transmission schemes) and also different computational load. Figure 2 shows the performance requirement for some of the different modes of WLAN and long-term evolution (LTE) standards. The performance requirement is based on the number of 16-b RISC giga-operations per second (GOPS) required for the inner-modem for peak-payload processing only. Performing all the computation on the same core running at a higher speed would be a very energy-inefficient solution. This confirms the need for a multicore solution for SDR. Furthermore, the computation have to be performed on ASIP-like architectures instead of a DSP. A back-of-the-envelope computation for WLAN multiple input multiple output (MIMO) 2×2 40 MHz that requires approximately 25 GOPS shows that an application-specific integrated circuit (ASIC) that has an energy per operation of 5–10 pJ/op [3] gives a power of 125 mW, a DSP that has a power efficiency of 125–250 pJ/op gives a power of 3 W, whereas an SDR ASIP that has a power efficiency of 15–30 pJ/op would give 375 mW. The power efficiency of 15–30 pJ/op is based on Interuniversity Microelectronics Centre's (IMEC's) SDR solution [6]. Because the heat dissipation in a handheld device should be kept under 3 W [7] and RF parts and user interface consumes approximately 1.5 W [8], we see that using 3 W DSP is not feasible. This further confirms the need for a more specialized and heterogeneous ASIP solution rather than a general purpose solution.

An evolving step in the SDR community is the need for supporting multiple standards on the platform in parallel. This is



[FIG2] **Performance requirements for inner modem peak data/payload processing for WLAN and LTE.**

one of the essential features that would be needed for evolution into a truly cognitive radio solution. A multicore solution is therefore essential to support such a case.

SDR: THE NEED FOR MAPPING TOOLS

Multicore solutions make the mapping of the application(s) more difficult than ever before. The sequential aspect of the program that was kept before from the initial specification to final implementation has to change to parallel during application mapping. The demand for mapping tools that make this process easier rises drastically.

Wireless standards have hard real-time constraints on top of demanding throughput requirements, which makes the parallelization task even more complex. Figure 2 shows the throughput requirement for WLAN and LTE as well as an estimate of the computational requirement for the different modes. The computational load for only the inner modem computation ranges from a few 16-b GOPS to 150 GOPS on the most demanding case. Note that these are estimates of the performance under various assumptions of algorithmic choice, channel conditions, etc. To provide such a scalability, a need exists for tools to enable and explore the different parallelization schemes on the multiprocessor system.

This problem is further augmented by the possibility of exploiting parallelization at different levels. Platforms today exploit parallelization across the cores, across the threads within one core, across different functional units (FUs) within one core and within one FU (in a data parallel way). Exploring this large mapping space manually is not feasible. Thus, a selective tool support at different levels is crucial.

An important aspect for the designer is not only to pick right tools to help him/her with the parallelization but also to think about the whole mapping flow and combination and interoperability of the tools to achieve wished global optimality. The ordering of parallelization exploration at different levels is also crucial as we will see later in the text.

MULTICORE SDR ARCHITECTURES: A COMBINATION OF HETEROGENEOUS AND HOMOGENEOUS MULTICORE APPROACHES

Radio transceiver ASICs for one standard consist of accelerators for the different functional blocks in the transmitter/receiver chain. These ASICs give the extreme end of the spectrum with little or no flexibility. Next-generation radios have been evolving into more programmable and more configurable solutions. The increased amount of standards to support and the dynamism in each of these standards have pushed baseband radio implementations towards a software centric end. This has lead to an evolution where the baseband radio is implemented on flexible and programmable processors (SDR platforms) instead of pure ASIC-based solution. An overview of the freedom and the evolution of the flexibility is shown in [9].

Future SDR platforms will require multi-Gb/s connectivity, concurrency support, and spectrum sensing capabilities. This is not feasible without multicore SDRs. Different multicore approaches exist in reconfigurable radio architectures. Mostly, the combination of heterogeneous and homogeneous multicore approach is a viable option. At the top level, the platform resembles a heterogeneous system, with a specialized digital front end (DFE), inner modem, and outer modem (forward-error correction) part. Each part potentially consists then of homogeneous multicore subsystem. Such a system should be able to run multiple future standards up and beyond to 1 Gb/s, allow sharing of hardware resources among several standards and support runtime (RT) mechanisms at hardware and software level as well as supporting spectrum sensing capabilities.

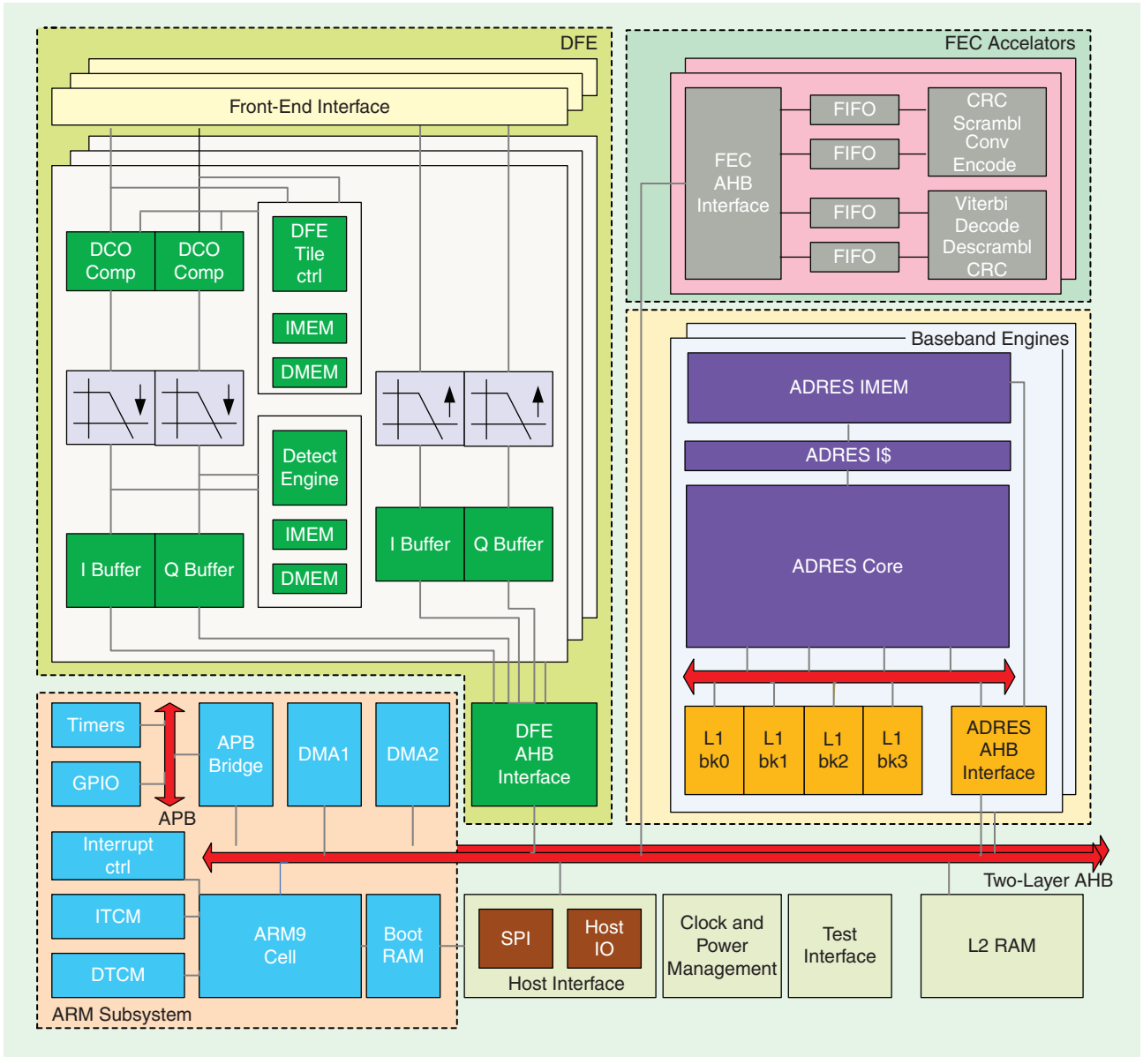
Given the large amount of software present inside these standards, to meet the high-performance and low-power requirements, there is a need for efficient exploitation of the different types of parallelism present. Broadly the parallelism can be broken down into three types: instruction-level parallelism (ILP), data-level parallelism (DLP), and task-level parallelism (TLP). ILP is when multiple instructions are executed in the processor in parallel, DLP is when multiple data elements undergo the same processing in parallel, and TLP is when multiple threads or tasks run in parallel on the processor or platform. Note that TLP can be exploited inside a single processor (intracore) or among processors (intercore).

Different SDR platforms exploit different types of parallelism in a better or worse way. Next, we give an overview of state-of-the-art multicore SDR platforms and highlight their important features. In the section "Comparative Study of Different Solutions," we highlight the features of the different platforms and summarize the pros and cons of the listed SDR platforms.

IMEC'S BEAR PLATFORM

IMEC's baseband engine for adaptive radio (BEAR) platform is a multicore heterogeneous platform consisting of six cores and two accelerators (see Figure 3). The six processors include three ASIPs for coarse time synchronization (DFE), one ARM processor for control (ARM subsystem), and two architecture for dynamically reconfigurable embedded systems (ADRES) processors (baseband engines) for baseband inner-modem processing. The coarse time synchronization ASIP is a low-power very-long instruction word (VLIW) with two scalar and three vector issue slots. The ADRES processor is a coarse grain reconfigurable array (CGRA) processor that is highly flexible and energy efficient. More information on the ADRES processor template can be found in [10]. The platform also contains accelerators for Viterbi decoding [forward error correction (FEC) accelerators]. The ARM processor is capable of performing control on the platform as well as to perform the MAC processing on the data stream. All the different cores are connected via an advanced microcontroller bus architecture (AMBA) for communication.

The BEAR platform offers a good mix of homogeneous and heterogeneous intercore TLP. Both the ADRES as well as the DFE processors have been designed to provide the appropriate mix of DLP and ILP for their corresponding tasks. Since the outer modem processing requires low flexibility



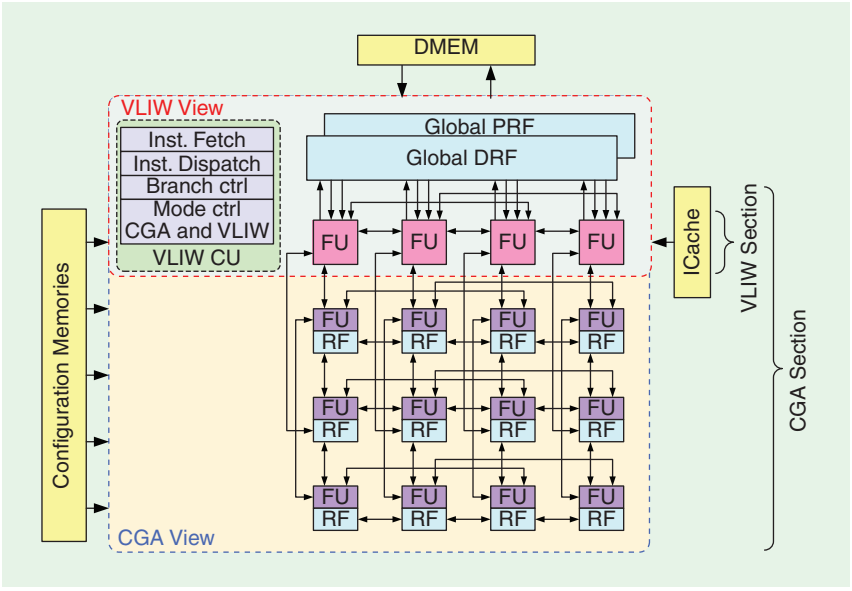
[FIG3] **IMEC's BEAR SDR platform.**

and high computation, it has been implemented as an ASIC accelerator. The inner modem, which requires high flexibility across different standards and inside one standard, has been implemented as a programmable processor. More detailed measurement results of the platform can be found in [6].

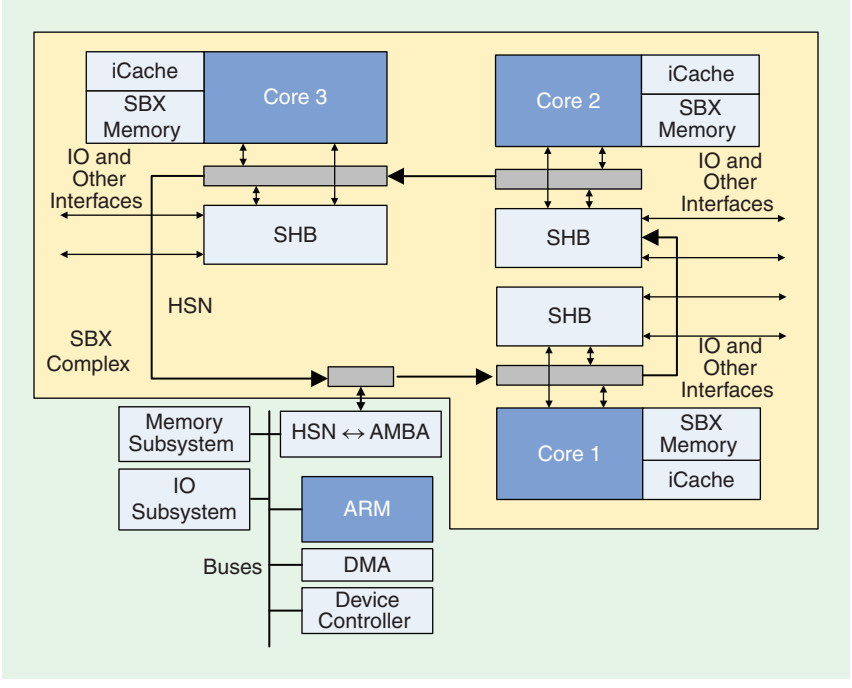
For the baseband processing, two ADRES processors are used. One such ADRES processor is shown in Figure 4. Each of the different FUs in the ADRES processor supports single instruction multiple data (SIMD) operations. Given that each ADRES processor offers ILP and DLP, and there are two such processors (TLP can be also used), a good parallelization strategy is a must to obtain an efficient mapping. Because the wireless application domain offers all the three different types of parallelism, various tradeoffs are possible on the type of parallelization strategy chosen, and each choice would have a different cost impact. These tradeoffs become even more varied when each standard to be mapped has various different modes, each of which have different computation and communication requirements.

SANDBRIDGE/SB3500

Sandbridge's SB3500 [11], [12] is the latest SDR platform generation from Sandbridge. The block diagram of the platform is depicted in Figure 5. This platform consists of four cores connected on an AMBA bus. Similar to other SDRs the control and the management of the platform is performed on the ARM processor. The remaining three cores on the platform are custom cores from Sandbridge called Sandblaster. Thus, the heterogeneity on this platform is very limited, differentiating only between the control (ARM) and the data processing (Sandblaster). All the inner- and outer-modem processing is done on the three Sandblaster cores.



[FIG4] **IMEC's ADRES processor in the BEAR platform.**



[FIG5] **Sandbridge SB3500 platform architecture.**

Each of the three Sandblaster cores has support for SIMD instructions and thus it can exploit the DLP available in the application. Because the platform consists of three data processing cores, inter-TLP among the different tasks in the application can be also exploited on the platform. Each Sandblaster core also offers a fine-grain intra-TLP inside a single core. This intracore parallelism is also referred to as "token triggered threading" (T^3), which is a form of simultaneous multithreading (SMT). Support for SMT allows the core to switch between different threads and their contexts quickly. However, the Sandblaster core has only limited ILP where only four instructions can be executed in parallel.

INFINEON MUSIC

Infineon's MuSIC-1 platform [9] is a heterogeneous multicore platform that consists of various accelerators along with four programmable cores. Each of these four programmable cores provides DLP and is used for the inner modem PHY processing with the help of filter accelerators. The turbo/Viterbi accelerators are used for performing the outer modem PHY processing. The block diagram of the platform is depicted in Figure 6.

The multicore nature of the MuSIC-1 platform supports intercore TLP, which allows the mapping of different tasks on different cores. Similar to Sandbridge, the ILP inside a single core is limited.

ST-ERICSSON EXTREME VECTOR PROCESSOR PLATFORM

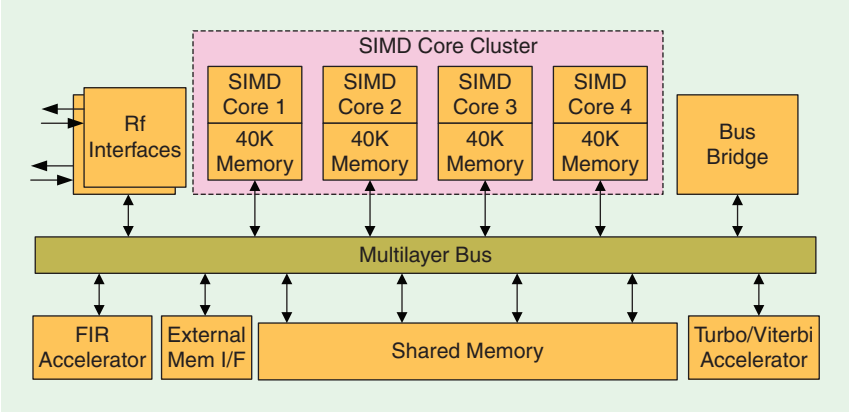
The extreme vector processor (EVP) [13] consists of 16-wide SIMD processor with five issue slots. Three of the five slots operate on vector data and two operate on scalar data. This processor exploits both data- and instruction-level parallelism in the application. However, not much public information is available on the complete platform architecture and how many cores would be needed to support a wireless standard.

ARM/UNIVERSITY OF MICHIGAN'S ARDBEG PLATFORM

ARM/University of Michigan's Ardbeg platform [14] consists of three processor cores. Two cores are allocated for baseband processing and one core for control. The platform also consists of a turbo coprocessor for outer-modem processing (see Figure 7). The platform enables TLP to be exploitable between the four functional blocks (control processor, two baseband cores, and a turbo accelerator). Each of the baseband cores is 512-b wide and is capable of performing 64-way, 32-way, and 16-way SIMD on 8-b, 16-b, and 32-b data, respectively. However, the baseband core does not allow a large amount of ILP inside the core. The baseband processor is also used to perform certain outer-modem functionality such as Viterbi decoding.

OTHER SOLUTIONS

Other platforms include Silicon Hive's CSP series [15]. These processors allow a large mix of ILP and DLP that can be exploited in the processor. However, there is not enough public information so it is not clear how these processors would fit on a platform, what parts of the PHY would map on the processor, and where the MAC processing would be mapped. Picochip's PC205 [16] solution also offers a mix of ILP, DLP, and TLP on the platform. However, this platform consists of a large number of hardware accelerators that pushes the platform towards a less flexible solution. Ceva's Ceva-XC platform [17] also consists of a mix of DLP and ILP available on the platform. The vector cores used on the platform consist of special instructions to accelerate the outer-modem processing.

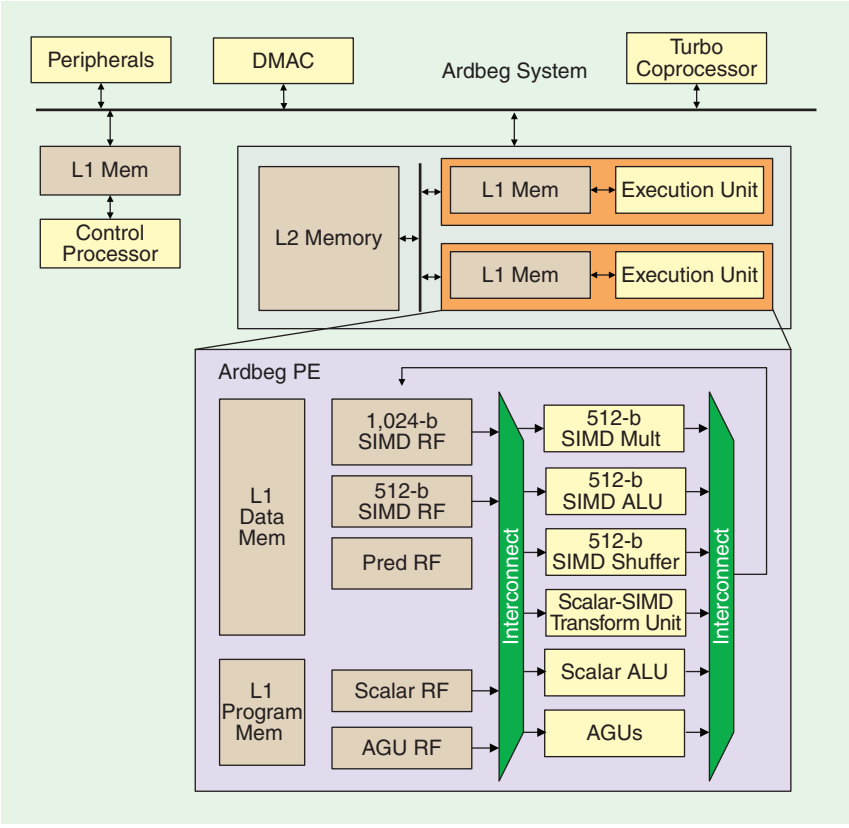


[FIG6] **Infineon MuSIC-1 platform architecture.**

COMPARATIVE STUDY OF DIFFERENT SOLUTIONS

Table 1 gives a comparative study of the parallelism offered by the different platforms. Each platform offers a different mix of parallelism to the programmer. It is interesting to note that all these platforms offer a high- to medium-data level parallelism. This is largely because of the fact that DLP is an energy-efficient way to exploit parallelism and most standards offer data-level parallelism. However, the other types of parallelisms are quite varied across the different platforms. To reach the required performance, each platform exploits different types of parallelism on top of the DLP.

In terms of area and power, it is very difficult to perform a good comparison with the information available in the public domain. Based on public information, it is not clear what the precise set of functionality is running on the platform, what the duty cycle of the processing is, what the area includes, what mode it is measured under, and what level of power/area estimation is used. Furthermore, the objective function that is even harder to compare is flexibility, platforms may have a specialized instruction set or accelerators that may heavily limit flexibility to port another standard on it.



[FIG7] **ARM/University of Michigan's Ardbeg architecture.**

MAPPING FLOWS FOR MULTICORE RADIO ARCHITECTURES

As shown in previous sections, different state-of-the-art SDR platforms exhibit different granularity and type of parallelism.

[TABLE 1] COMPARISON OF PARALLELISM OFFERED BY DIFFERENT PLATFORMS (L: LOW, M: MEDIUM, H: HIGH).

PLATFORM	DLP	ILP	INTER TLP	INTRA TLP
CEVA-XC [17]	M	M	L	L
IMEC BEAR [6]	M	H	M	L
UMICH/ARM [14]	H	L	L	L
ST-NXP [13]	H	L	L	L
INFINEON'S MUSIC-1 [9]	M	M	L	M
SANDBRIDGE'S SB3500 [11], [12]	M	L	M	H

Programming such radio architectures requires having an appropriate mapping flow supported by the tools that can explore TLP, DLP, and ILP. Mapping process starts from algorithmic specification of a radio standard that is tuned to a optimized code utilizing DLP and ILP and keeping task distribution (TLP) in mind.

SYSTEMATIC GLOBAL FLOW PRINCIPLES FOR EXPLOITING PARALLELIZATION

As mentioned before, different types of parallelism can be exploited in the application: TLP, DLP, and ILP. These types of parallelism have different granularity and impose different constraints on the remaining part of the mapping flow. Because of its middle granularity, ILP is the most restrictive type of parallelism and it should be applied as the last step in the parallelization mapping flow. Applying ILP too early can seriously restrict some TLP options. For example, when applying the transformations to achieve a good ILP in a certain part of the application, it might not be possible to split this part to two separate tasks any more. Reversed ordering, i.e., applying ILP after TLP, is less restrictive. DLP has the fine granularity and thus it can be applied very locally. Still, when exploiting ILP, DLP should be already explored. This can be motivated by the fact that DLP can always be broken down into ILP and not vice versa. However, DLP should be exploited in the individual tasks, because different tasks can utilize different DLP strategies. Thus, DLP exploration should be placed between TLP and ILP exploration. TLP itself has two subclasses, interprocessor TLP, which is TLP across several processor cores, and intraprocessor TLP, which is TLP across several threads within one core. To perform interprocessor TLP first and then intraprocessor TLP is natural order as it provide the lowest constraints on the available search space freedom. The tasks operating on different cores should have minimal communication, whereas the tasks running on different threads within one core can afford more communication overhead.

When exploiting TLP, functional and/or data split can be applied. The functional split assigns different functionality to different tasks. The data split assigns different iteration ranges of the same functionality to different threads. Also, a combination of both is possible. A specific combination is task pipelining, when different functionality in different iteration ranges is executed in parallel. Data split in TLP might limit the DLP, however, the freedom for DLP is not so limited as it would be in reversed ordering. This also confirms the need to apply TLP before DLP.

WORKLOAD ESTIMATION ISSUES IN THE MAPPING FLOW

In the section "Systematic Global Flow Principles for Exploiting Parallelization," the flow requires an estimate of the workload for load balancing in TLP and estimation of communication overhead. This requires exploration of DLP and ILP first to obtain the timing information. There are two possibilities that can solve this issue. The first one is to utilize high-level estimators during decision on the TLP parallelization strategy. Those estimators provide the designer with the upper and lower bounds of DLP and ILP. The second possibility is to rely on an experienced designer that can perform those estimations "in his head." We can observe this in most practical mapping flows. Even when we start with DLP and ILP parallelism that can be exploited within one processing core, we also implicitly explore TLP at the beginning of the flow. This type of practical mapping flow was also confirmed by the Multicore Association, which encompasses many important industrial players [18]. If the designer is not experienced enough and performs wrong implicit TLP exploration, he/she will enter in the global loop where he/she has to return to the DLP or ILP exploration when no satisfactory TLP solution has been found. Those loops are, of course, too costly and are very rarely entered by an experienced designer. However, it is still possible. The high-level estimators can eliminate these errors and thus are a crucial component for a more automated future mapping flows, especially when RT managers have to make these decisions (see the section "Dynamic Scalability of Baseband Signal Processing: Impact on Mapping").

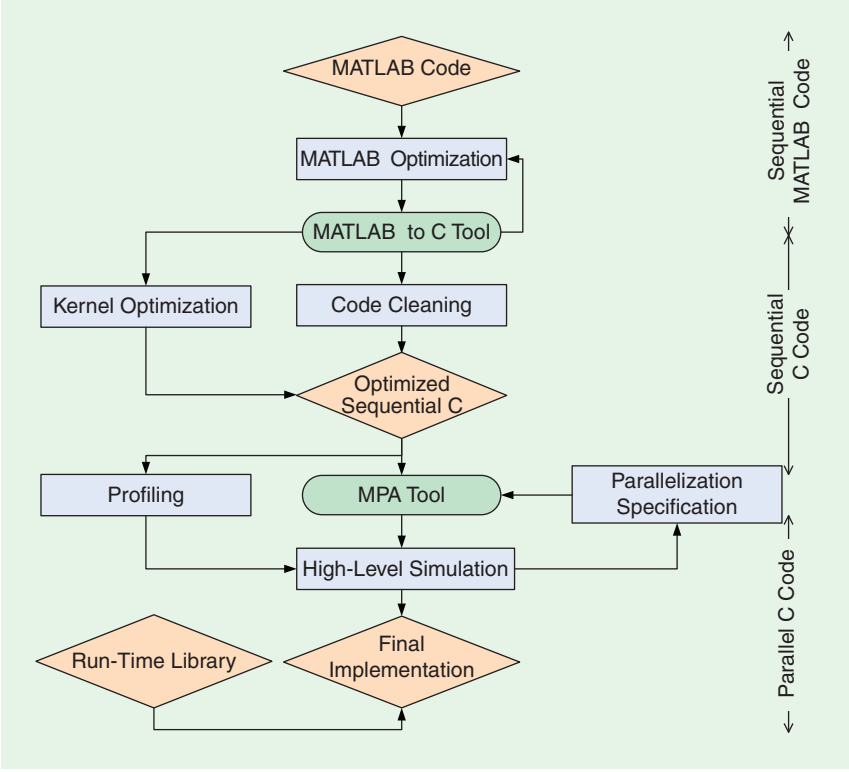
BEAR MAPPING FLOW

Every mapping flow starts with initial algorithmic specification and ends with final implementation targeting the best performance, energy and/or area on a given platform. During the mapping flow, different transformations are applied that expose certain properties of the application. In the BEAR mapping flow, we have mainly focused on exposing parallelization at different granularity levels, that allows us utilizing the platform resources efficiently. These transformations are orthogonal with transformations targeting other issues such as optimizing the memory hierarchy system [19], [20].

The BEAR mapping flow in Figure 8 starts with initial MATLAB algorithmic specification. Then, high-level MATLAB transformations are applied. Those optimizations allow efficient C code generation later and also include global data-flow and loop transformations [19] , [20]. The code is quantized and MATLAB to C conversion tool [21] is used to generate the C code. The C code is split into kernels such as FFT, tracking, channel compensation, demodulation, and skeleton code that is calling these kernels. The skeleton code is cleaned, i.e., the constructs not supported by the parallelization flow are rewritten (such as dynamic allocation). The kernels are optimized in separate path. First, special intrinsic instructions of the target ADRES processor [10] are used that allow SIMD operations. After exploiting DLP, ILP is exploited by (low-level) loop transformations such as loop unrolling, loop coalescing, if conversion, code hoisting etc. Note the difference with the loop transformations at MATLAB level that are applied more globally, not only within one loop nest. Precompilation of the kernels using our DRESC compiler [10] is ending the kernel optimization process.

After kernel optimization, precompiled kernels can be combined with the cleaned skeleton code resulting in efficient sequential implementation that is profiled. To exploit TLP across the kernels, we utilize our MPA in-house tool [22]. MPA takes as input the sequential C code and a parallelization

specification, based on which the parallel code is generated. The synchronization is automatically inserted in the parallel code when needed to obey the original dependencies. The parallelization specification distributes each iteration instance of the different kernel among the different threads and/or processors (intra- and inter-TLP). When knowing the durations of the kernels in each iteration via the profiling (see Figure 8), the performance of the parallelized code can be rapidly evaluated by our high-level simulator. The parallelization is specified by parallelization specification file written by the designer. As previously mentioned, the experienced designer will have the possible parallelization specifications "in his head" even before starting the DLP and ILP exploration. After selecting the best parallelization strategy, the parallelized code is combined with the RT library to bridge the MPA tool high-level application programming inteface (API) and the platform low-level API to achieve the final implementation. On the BEAR platform, the threads and communication between them is controlled by the ARM processor.



[FIG8] **BEAR mapping flow.**

Our mapping flow allowed exploration of different parallelization possibilities such as an antenna- and symbol-based split for WLAN MIMO 2×2 40 MHz. It also allowed achievement of the real-time throughput behavior [23] and exploration of the maximum feasible parallelism for increased number of processing cores for the same application. We also experienced the global loops in the flow that are mentioned in the previous section. When we first optimized the kernels for DLP and ILP with focusing on per-symbol TLP split [23], it was not feasible with the same DLP and ILP solution to perform the antenna TLP split. The mapping flow can be used as in multiprocessor context (inter-TLP) so in multithreading context (intra-TLP).

SANDBRIDGE MAPPING FLOW

In the case of the Sandbridge architecture described in the section "Sandbridge/SB3500," not many details are available on a full methodology. Based on the mapping strategies described in [24] and [25], it is clear that the intercore thread-level parallelism is first exploited, followed by intracore thread-level parallelism. The process of deciding on the intracore threads is fine-grained compared to other platforms as the Sandbridge processor exploits the T^3 technology. The T^3 technique allows quick context switches among multiple threads on the same core to up to eight threads on a single core. After deciding the inter- and intrathreads, the data level and the instruction-level parallelism is finally exploited. The decision on the parallelization strategy and the parallelization itself is left to the developer.

TEXAS INSTRUMENTS' ALGORITHM ARCHITECTURE MATCHING METHODOLOGY

The algorithm architecture matching (AAM) methodology developed by Texas Instruments maps an algorithm that is described as a graph to a physical architecture given a set of constraints [26]. The architecture is described as the architecture graph in which vertices represent operators (DSP cores) and the edges represent communication. The AAM methodology takes the two graphs and set of constraints and it performs placement and scheduling of the algorithm graph nodes over architecture graph nodes. This resembles IMEC's dynamically reconfigurable embedded system compiler (DRESC) modulo scheduling approach [10], but on a much coarser level. The architectural nodes in AAM are complete DSP cores where as in the DRESC, the architectural nodes are FUs in the CGRA part of the processor.

The AAM methodology is employed using the parallel real-time embedded executives scheduling method (PREESM) tool. Even when the tool can generate code, the input of the tool has to be described as a synchronous data flow (SDF) graph, which is increasing the manual effort during the mapping process. On the other side, once the SDF graph is present, the exploration for different architectures is straightforward. Of course, appropriate architecture graphs have to be present.

INFINEON MUSIC MAPPING FLOW

For the MuSIC platform described in the section "Infineon MuSIC," the mapping flow starts from a functional C

description of the complete standard. This is further refined by choosing the right parallelization strategy in both the thread level and data level [27], [28]. The DLP parallelization (SIMD) is then exploited using extensions to the C language as described in [29]. As mentioned in the section "Workload Estimation Issues in the Mapping Flow," the SIMD estimations are performed in the head of the designer before performing the thread-level parallelization. The DLP parallelization (using SIMD instructions) itself is performed manually, which is often common in most design flows. Furthermore, the MuSIC platform also has a lightweight real-time operating system (RTOS) called ILTOS to enable multithreaded programming. The API of ILTOS provides basic functions for thread administration, memory management, synchronization etc. Communication API is similar to the API provided by IMEC's MPA tool.

THE FAST EVOLUTION OF WIRELESS STANDARDS HAS CAUSED A SHORTER TIME-TO-MARKET, WHICH MAKES A PROGRAMMABLE SDR SOLUTION ATTRACTIVE.

SPEX—A PROGRAMMING LANGUAGE FOR SDR

SPEX [30], from the University of Michigan, is an object-oriented programming language based on C++ semantic targeting the SDR platforms. Three additional keywords are added to the language kernel, stream, and synchronous to distinguish the sequential C kernels in the application, concurrent data streaming, and discrete real-time computations. This provides a clear interface among the DSP algorithm description in kernel SPEX, parallel execution in stream SPEX, and system integration in synchronous SPEX. The different SPEXs are compiled by different compilers in different phases of the mapping; kernel SPEX with the SIMD and VLIW compiler, stream SPEX with the data flow compiler, and synchronous SPEX with the real-time compiler. The SPEX flow is shown in Figure 9.

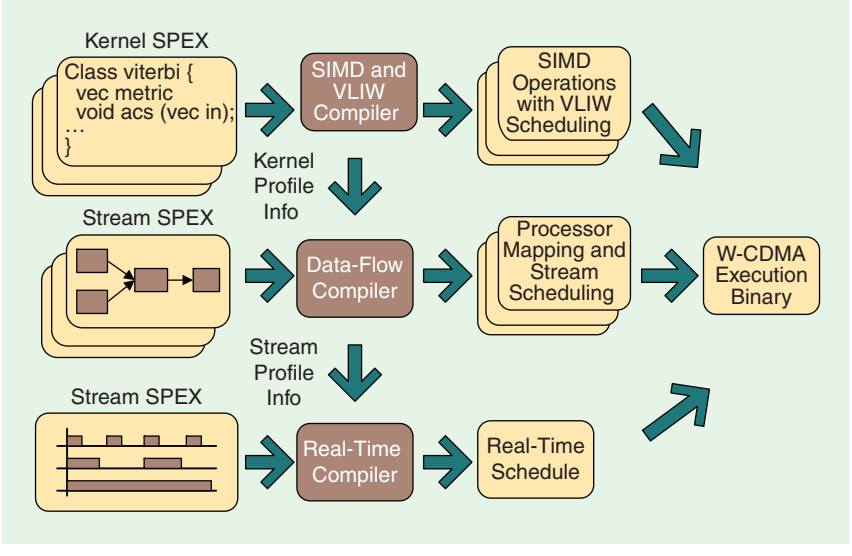
We consider this approach similar to our BEAR baseband mapping flow approach (see the section "BEAR Mapping Flow"). The kernel approach is similar to IMEC's approach where kernels are optimized and precompiled by the DRESC. The stream SPEX can be compared to the mixture of the parallelization phase in BEAR flow (using MPA) and the first part of system integration phase. Synchronous SPEX can be considered the last phase of the IMEC system integration step.

OTHER MAPPING FLOWS

In a task-transaction level (TTL) methodology [31] developed at University of Twente and Philips Research, an application is modeled as a task graph, where a task is an entity that performs computations. The implementations are encapsulated in TTL shells that are exposed to the platform. The TTL approach raises the level of the abstraction, and it tries to close the gap between application models used for specification and the optimized implementation of the application by its combination. However, it seems that the placement and scheduling process it manual to the large extend compared e.g., to the AAM approach described in the section "Texas Instruments' Algorithm Architecture Matching Methodology."

COMPARISON OF DIFFERENT MAPPING FLOWS

In previous sections, we first looked at the SDR mapping methodology from the meta-level perspective and then provided instantiations developed by different groups for different SDR platforms. The common drawback we see for most of the methodologies is either the need for graph description of the application (e.g., AAM) or just providing programming models for the SDR mapping (e.g., TTL) without any tool support. Both approaches can result in tedious and error-prone mapping. Also, most methodologies focus only on mapping for a particular homogeneous system (e.g., SPEX is targeting multicore SIMD DSP processor platform only) and we miss heterogeneity of given solutions. Currently this is the main challenging task we foresee in the SDR mapping. Some attempts have been made recently to cover this gap, i.e., extend the OpenMP for heterogeneous multicore systems [32]. In this context, we see the BEAR mapping flow as one of more complete solutions. Even though some parts of the flow are fully manual, the critical parts of the mapping are automated and tool support is present.



[FIG9] SPEX design flow [30].

FUTURE RESEARCH DIRECTIONS: MOVING TOWARD DYNAMIC MULTICORE COGNITIVE RADIO PLATFORMS

Multicore SDR and the associated mapping techniques enable applications that were not feasible in the single-core SDR era. In the following sections, we will discuss the trends that are enabled by multicore SDR. First, we will focus on concurrent data streams/connections and their impact on RT management. Then we will highlight dynamic scalability of wireless systems and its impact on mapping. Finally, we will tackle new complexity challenges for fourth generation (4G) and beyond, multiuser and network MIMO, and spectrum sensing.

CONCURRENT MULTIPLE CONNECTIONS/STREAMS

MOTIVATION

In the past, the management of concurrent data streams or connections mostly happened for SDR base stations. However, this functionality of management of concurrent streams is moving to the mobile terminals as well. For instance, a mobile terminal may need to stream encoded music data from a WLAN connection, and the decoded music needs to be sent to a wireless earphone with a Bluetooth link. The scenario of voice over Internet protocol calls with a wireless earphone is very similar. In many other cases, although multiple data streams are not explicitly visible for the user, the baseband processing still needs to tackle multiple connections with different streams of raw data. For instance, when the user is browsing the Internet with the WLAN connection of his or her mobile, in the background the mobile may be continuously ranging and synchronizing with multiple base stations for cellular links such as LTE or LTE-advanced. In the long term, tackling concurrent connections and streams will become even more important. In future wireless mesh and ad-hoc network, terminals may need to relay multiple data streams in the background, whereas the foreground data streams still have to be guaranteed. Clearly, tackling multiple connections and data streams simultaneously will be essential for future SDR devices.

IMPACT ON RT MANAGEMENT

Multiple streams and standards operation on the SDR platform in parallel will have also an impact on the RT management of the platform. The platform control should be more distributed compared to today's centralized control. The platform and the platform control should support coexistence of multiple standards and the hand-over between the standards. We foresee the flexibility supporting this during the RT in the next three planes.

- The multistandard plane supporting concurrent run of multiple standards in parallel ensures that each standard can run on separate processor core and/or as separate threads within a core, if the core can support intraprocessor TLP. Note that a standard can be distributed across several threads and/or cores.
- The horizontal plane ensures the flexibility and implementation scalability of a standard (within a given mode). For example, there are several possible implementations of WLAN MIMO 2×2 40 MHz, resulting in energy-time-area tradeoff, where one implementation can be energy efficient but not so performing as other, energy-hungry implementation. The implementations can differ also in code size that is reflected in the area axis of the tradeoff.
- The vertical plane ensures mode scalability of a standard, i.e., that we can switch among different modes of the given standard. A typical example is the single input single output/MIMO mode support where a decision is made based on the incoming signal field (in the case of WLAN).

Those planes are important also in a hand-over scenario, where the situation could be as follows. One standard that is running on the platform scales down (horizontal plane) to free space for the second standard. For a certain period of time, the two standards run parallel and the hand-over will happen. Then the first standard is stopped and the second standard is scaled to all available resources. Thus, the hand-over issue is naturally connected to concurrent connections and RT support for the future platforms should be adapted to this.

DYNAMIC SCALABILITY OF BASEBAND SIGNAL PROCESSING: IMPACT ON MAPPING

In wireless systems, both the environment and the user requirement contain abundant dynamics. First, the environment is inherently time varying, e.g., channel conditions, interferences, and spectrum utilization. In addition, the user requirement also contains lots of dynamics, e.g., data rate, tolerable error, tolerable jitter, and tolerable latency. Baseband with dynamic scalability can adjust processing complexity according to the above dynamics [33]. For instance, the search range of nonlinear MIMO detectors, the modulation accuracy of orthogonal frequency division multiple access modulator, and the tracking strength of channel estimators can be adjusted based on different metrics or requirements. These adaptations will lead to heavily reduced computations and memory accesses, which eventually translates into substantially reduced average energy consumption.

Although such dynamic scalability can improve energy efficiency, it imposes many challenges on multicore mapping. Importantly, the work load of tasks are not completely deterministic anymore, it will depend on channel conditions and input data. This brings complex situations when handling real-life baseband signal processing on a multicore platform. Worst-case mapping would suffer severely from efficiency, therefore a mix of design-time and RT decisions would be needed to reach an efficient solution.

NEW COMPLEXITY CHALLENGES

BASEBAND COMPLEXITY OF 4G AND BEYOND

International Mobile Telecommunications-Advanced (IMT-Advanced), has already initialized massive effort for the evolution beyond the 3rd Generation Partnership Project (3GPP), LTE, and IEEE802.16e. The planned improvement

spreads over system architecture, radio resource management, MAC scheme, and air interface. Regarding the air interface, it is clear that very large bandwidth (e.g., 100 MHz) and larger MIMO systems (e.g., 4×4 or even higher) [34] will be implemented. This increased bandwidth directly translates to a complexity challenge that requires multiple-core baseband platform and efficient mapping therefore becomes more crucial.

CLEARLY, FUTURE MULTICORE SDR DEVICES WILL BE ESSENTIAL FOR TACKLING MULTIPLE CONNECTIONS AND DATA STREAMS SIMULTANEOUSLY.

To enable this, there is a need for further improvement in the platform architecture as well as in the algorithms to obtain the best combination of architecture and algorithm. Better use of the parallelism is definitely one of the key challenges to be addressed. Importantly, future algorithmic engineering will have to take into account architecture characteristics from the very beginning of the mapping flow, to ensure that TLP, DLP, and ILP can be effectively exploited in later mapping steps.

MULTIUSER MIMO AND NETWORK MIMO

Multiuser MIMO and network MIMO have been considered as a promising candidate technology for highly spectrum efficient wireless systems. Whereas traditional MIMO leaped from the traditional multipath avoidance and compensation paradigm to the multipath exploitation paradigm, multiuser MIMO and network MIMO jump one step further, shaping interference and exploiting interference. However, such a new paradigm brings higher spectrum efficiency at the cost of significantly increased computation complexity. For multiuser MIMO, the condition or capacity of many user-specific MIMO channels have to be analyzed, steered, or compensated. Complex signal processing algorithms, such as eigenvalue spread analysis, will become necessary. For network MIMO, joint processing and detection will increase the dimension of signal processing, the increment of complexity would be orders of magnitudes.

SPECTRUM SENSING

One of the key enablers of next-generation cognitive radio systems would be "spectrum sensing" [1], [35]. This would imply that each of the different users would be able to sense the spectrum usage in the air and use the spectrum that would be the most optimal in a more dynamic way. This would enable each user to use the spectrum in a more effective way. Various standardization efforts like IEEE 802.22 [36] are currently in progress to standardize cognitive radio and sensing requirements. This may not only happen in the digital TV bands but spectrum sensing would also help a better coexistence scenarios between various standards like 802.11 and 802.15.4. Such spectrum sensing techniques would also bring higher levels of dynamism and complexity to the mobile terminal.

CONCLUSIONS

In this article, we provided an overview of multicore architectures for future SDR platforms and their mapping flows. First, we motivated the need for scalable and reconfigurable heterogeneous multicore SDR platforms driven by technology constraints, user demands, and business aspects. We also highlighted the urgent need for appropriate mapping flows when mapping on those platforms. Then we gave an overview of the most popular multicore SDR platforms on the market with a short comparative study among them. In the mapping flow section, we started a discussion on general mapping flow going to different instances of the mapping flows. Finally, we discussed the new aspects and applications the multicore SDR era with proper mapping flow will bring.

AUTHORS

Martin Palkovic (palkovic@imec.be) received his M.Sc. degree in electrical engineering (with highest distinction) from the Slovak University of Technology, Bratislava, Slovakia, in 2001, and his M.Sc. degree in economics from the University of Economics, Bratislava, Slovakia, in 2000. He joined IMEC in 2001, where he was a researcher in the design technology group from 2001 to 2008 and has been a senior researcher in the wireless group since 2009. From 2002 to 2007, he worked towards his Ph.D. degree in the Department of Electrical Engineering at Technische Universiteit Eindhoven, The Netherlands. His research interests include high-level optimizations in data dominated multimedia applications and wireless systems, platform architectures for low power, and mapping techniques for multicore SDR.

Praveen Raghavan (ragha@imec.be) received his bachelor's degree from the National Institute of Technology, Trichy, India, in 2002. He received his master's degree in electrical engineering from Arizona State University in 2004, and his Ph.D. degree from K.U. Leuven in electrical engineering in 2009. He is currently a wireless systems researcher in the wireless group at IMEC. His research interests include low-power design, system design, low-power architectures, and SDR.

Min Li (limin@imec.be) received his B.E. degree (with the highest honor) in 2001 from Zhejiang University, Hangzhou, China. From 2001 to 2004, he was a postgraduate student with Zhejiang University. In 2003, he was an intern with Lucent Bell Labs Research China, working on network processors. From September 2003 to September 2004, he interned with Microsoft Research Asia, working on low-power mobile computing. From 2004 to 2009, he was a Ph.D. researcher with IMEC and a Ph.D. student with the ESAT group at K.U. Leuven. He received the 2007 IEEE SIPS Best Paper Award. He is currently a researcher with IMEC. His research interests include low-power signal processing and implementation.

Antoine Dejonghe (dejonghe@imec.be) received the electrical engineering degree and the Ph.D. degree from the Université Catholique de Louvain (UCL), Louvain-la-Neuve, Belgium, in 2000 and 2004, respectively. From 2000 to 2004, he pursued a Ph.D. degree with the Communications and Remote Sensing

Laboratory as a research fellow of the Belgian National Fund for Scientific Research. Since December 2004, he has been with the wireless group of IMEC, Leuven, where he is currently a program manager for cognitive reconfigurable baseband activities. He is the author and coauthor of over 75 scientific publications.

Liesbet Van der Perre (vdperre@imec.be) received the M.Sc. degree in electrical engineering from K.U. Leuven, Belgium, in 1992. The research for her thesis was completed at the Ecole Nationale Superieure de Telecommunications in Paris, France. She graduated with a Ph.D. degree in electrical engineering from K.U. Leuven in 1997. After joining IMEC in 1997, her focus was on ASIC architectures for OFDM, turbo coding, and SDR radio. Currently, she is a program director for the cognitive reconfigurable radios and mm-wave communications programs at IMEC. She is a part-time professor at K.U. Leuven, Belgium, and she is an author and coauthor of over 200 scientific publications.

Francky Catthoor (catthoor@imec.be) received the engineering and a Ph.D. degrees in electrical engineering from Katholieke Universiteit Leuven, Belgium, in 1982 and 1987, respectively. Between 1987 and 2000, he headed several research domains in the area of high-level and system-synthesis techniques and architectural methodologies, including related application and deep submicron technology aspects, all at IMEC, Heverlee, Belgium. He is an IMEC fellow. He is a part-time full professor in the Electrical Engineering Department of K.U. Leuven. In 1986, he received the Marconi International Fellowship Council's Young Scientist Award. He has been an associate editor for several IEEE and ACM journals, including *IEEE Transactions on VLSI Signal Processing, IEEE Transactions on Multimedia,* and *ACM TODAES*. He was the program chair of several conferences including ISSS'97 and SIPS'01. He is an IEEE Fellow.

REFERENCES

[1] J. Mitola, "Cognitive radio architecture evolution," *Proc. IEEE*, vol. 97, no. 4, pp. 626–641, Apr. 2009.

[2] V. Giannini, J. Craninckx, S. D'Amico, and A. Baschirotto, "Flexible baseband analog circuits for software-defined radio front-ends," *IEEE Solid-State Circuits*, vol. 42, no. 7, pp. 1501–1512, July 2007.

[3] W. J. Dally, J. Balfour, D. B. Shaffer, J. Chen , R. C. Harting, V. Parikh, J. Park and D. Sheffield, "Energy efficient computing," *IEEE Comput. Mag.*, vol. 41, no. 7, pp. 27–32, July 2008.

[4] M. Gries, K. Ketuzer, H. Meyr and G. Martin, *Building ASIPS: The Mescal Methodology*. New York: Springer, 2005.

[5] P. Ienne and R. Leupers, *Customizable Embedded Processors: Design Technologies and Applications*. San Francisco, CA: Morgan Kaufman, 2006.

[6] V. Derudder, B. Bougard, A. Couvreur, A. Dewilde, S. Dupont, L. Folens, L. Hollevoet, F. Naessens, D. Novo, P. Raghavan, T. Schuster, K. Stinkens, J.-W. Weijers, and L. Van der Perre, "A 200 Mbps+ 2.14nJ/b digital baseband multi processor system-on-chip for SDRs," in *Proc. VLSI Circuits Symp.*, June 2009, pp. 292–293.

[7] Y. Neuvo, "Cellular phones as embedded systems," in *Proc. IEEE Int. Solid-State Circuits Conf.*, Feb. 2004, vol. 1, pp. 32–37.

[8] O. Silven and K. Jyrkka, "Observations on power-efficiency trends in mobile communication devices," *EURASIP J. Embedded Syst.*, vol. 2007, no. 1, pp. 17–27, 2007.

[9] U. Ramacher, "Software-defined radio prospects for multistandard mobile phones," *IEEE Comput. Mag.*, vol. 40, no. 10, pp. 62–69, Oct. 2007.

[10] B. Mei, S. Vernalde, D. Verkest, H. De Man, and R. Lauwereins, "ADRES: An architecture with tightly coupled VLIW processor and coarse-grained reconfigurable matrix," in *Proc. IEEE Conf. Field-Programmable Logic and Applications (FPL)*, Lisbon, Portugal, Sept. 2003, pp. 61–70.

[11] Sandbridge Technologies. (2009). *The Sandblaster Architecture* [Online]. Available: http://www.sandbridgetech.com

[12] M. Moudgill, J. Glossner, S. Agrawal, and G. Nacer, "The sandblaster 2.0 architecture and SB3500 implementation," in *Proc. Software Defined Radio Technical Forum*, Oct. 2008 [Online]. Available: http://www.sdrforum.org/sdr08_papers/2.5/2.5-3.pdf

[13] K. van Berkel, F. Heinle, P. P. E. Meuwissen, K. Moerman, and M. Weiss, "Vector processing as an enabler for software-defined radio in handheld devices," *EURASIP J. Appl. Signal Process.*, vol. 2005, no. 16, pp. 2613–2625, 2005.

[14] M. Woh, Y. Lin, S. Seo, S. Mahlke, T. Mudge, C. Chakrabarti, R. Bruce, D. Kershaw, A. Reid, M. Wilder, and K. Flautner, "From soda to scotch: The evolution of a wireless baseband processor," in *Proc. 2008 41st IEEE/ACM Int. Symp. Microarchitecture (MICRO-41)*, Nov. 2008, pp. 152–163.

[15] Silicon Hive. SiliconHive HiveFlex CSP (2009). [Online]. Available: http://www.siliconhive.com

[16] Picochip. Picochip PC205 product brief (2009). [Online]. Available: http://www.picochip.com

[17] CEVA Inc. CEVA DSP core X1641 product note (2009). [Online]. Available: http://www.parthus.com

[18] *Multicore Association. Multicore-Programming Practices Group* (2009). [Online]. Available: http://www.multicore-association.org/workgroup/mpp.php

[19] F. Catthoor, K. Danckaert, C. Kulkarni, E. Brockmeyer, P. G. Kjeldsberg, T. Van Achteren, and T. Omnes, *Data Access and Storage Management for Embedded Programmable Processors*. Norwell, MA: Kluwer, 2002.

[20] P. R. Panda, N. Dutt, and A. Nicolau, *Memory Issues in Embedded Systems-on-Chip: Optimizations and Exploration*. Norwell, MA: Kluwer, 1998.

[21] Agility. Agility MATLAB to C (2009) [Online]. Available: http://www.agilityds.com/products/matlab_based_products/default.aspx

[22] J.-Y. Mignolet, R. Baert, T. J. Ashby, P. Avasare, H.-O. Jang, J. C. Son, "MPA: Parallelizing an application onto a multicore platform made easy," *IEEE Micro*, vol. 29, no. 3, pp. 31–39, May/June, 2009 [Online]. Available: http://www.computer.org/portal/web/csdl/doi/10.1109/MM.2009.46

[23] M. Palkovic, A. Folens, H. Cappelle, M. Glassee, and L.Van der Perre, "Optimization and parallelization of 40 mhz MIMO SDM-OFDM baseband processing to achieve real-time throughput behaviour," in *Proc. ICT-MobileSummit 2009 Conf.*, Santander, Spain, June 2009.

[24] D. Iancu, H. Ye, J. Glossner, A. Iancua, and J. Takala, "Software only implementation of DVB-H," in *Proc. SPIE Multimedia on Mobile Devices*, vol. 6821. San Jose, CA, Jan. 2008, p. 68210F.

[25] D. Iancu, H. Ye, M. Senthilvelana, V. Kotlyar, J. Glossner, S. Agrawal, S. Jinturkar, A. Iancua, G. Nacer, S. Stanleya, M. Sima, and J. Takala, "Hand held analog television over WiMAX executed in SW," in *Proc. SPIE Multimedia on Mobile Devices*, vol. 6507. San Jose, CA,2007, p. 650709.

[26] M. Pelcat, S. Aridhi, and J. F. Nezan, "Optimization of automatically generated multi-core code for the LTE RACH-PD algorithm," in *Proc. Design and Architectures for Signal and Image Processing Conf. (DASIP)*, Bruxelles, Belgium, Nov. 2008, pp. 69–76 [Online]. Available http://www-labsticc.univ-ubs.fr/~gogniat/DASIP2008/proceedingDASIP2008.pdf

[27] Y. Jiang, W. Xu, and C. Grassmann, "Implementing a DVB-T/H receiver on a software-defined radio platform," *Int. J. Digital Multimedia Broadcast*, vol. 2009.

[28] C. Grassmann, M. Richter, and M. Sauermann, "Mapping the physical layer of radio standards to multiprocessor architectures," in *Proc. DATE*, Apr. 2007, pp. 1412–1417.

[29] Crescent Bay Software. Data Parallel C Extensions(DPCE) [Online]. Available: http://www.crescentbaysoftware.com/dpce/index.html

[30] Y. Lin, R. Mullenix, M. Woh, S. Mahlke, T. Mudge, A. Reid, and K. Flautner, "Spex: A programming language for software defined radio," 2008 [Online]. Available: http://data.memberclicks.com/site/sdf/SDR06-2.3-3.pdf

[31] Q. Zhang, A. B. J. Kokkeler, and G. J. M. Smit, "Cognitive radio design on an mpsoc reconfigurable platform," in *Proc. IEEE 2nd Int. Conf. Cognitive Radio Oriented Wireless Networks and Communications (CrownCom 2007), IEEE Communications Society*, Aug. 2007, pp. 187–191.

[32] L. Huang and B. Chapman, "Programming heterogeneous systems based on openMP," in *Proc. Int. Conf. Parallel Computing (ParCo'09)*, ENS, Lyon, France, Sept. 2009, submitted for publication.

[33] M. Li, B. Bougard, L. Van der Perre, and F. Catthoor, "Energy-aware algorithm and implementation of SDR oriented HSDPA chip-level equalizer," *J. Signal Process. Syst.*, vol. 56, no. 2–3, pp. 327–340, Sept. 2009.

[34] S. Parkvall, E. Dahlman, A. Furuskar, Y. Jading, M. Olsson, S. Wanstedt, and K. Zangi, "LTE-advanced evolving LTE towards IMT-advanced," *Proc. VTC*, pp. 1–5, Sept. 2008.

[35] P. F. Marshall, "Extending the reach of cognitive radio," *Proc. IEEE*, vol. 97, no. 4, pp. 612–625, Apr. 2009.

[36] *IEEE 802.22 Work Group on WRANs (Wireless Regional Area Networks)*, IEEE 802.22, Feb. 2009.

[SP]