34 Demonstrators and testbeds

Andreas Burg and Markus Rupp

34.1. Introduction

Since the establishment of the first point-to-point radio communication by Guglielmo Marconi in 1895, the complexity of communication systems has grown dramatically in the continuing search for higher system capacity, higher data rates, and bandwidth efficiency. Research in information theory and digital signal processing has been the foundation for the development of systems that are able to handle millions of users all over the globe.

However, the enormous proliferation of systems such as GSM or IS-95 has only become possible through the additional significant advances in integrated circuits technology. The ability to integrate complex algorithms and systems in a small, low-power device at a reasonable price has been the key factor for the success of mobile communication.

While modern ASIC technology allows for the integration of systems, the cost factor is largely dependent on the development effort and time to market. As systems become larger and more complex, the majority of the development effort is spent on its verification and performance analysis [1, 2]. With the necessity to introduce standards as the basis for enormous future development efforts, important decisions need to be made very early in the design cycle of a new system. At this point the actual implementation of a product is often not an immediate issue or might even not yet be feasible at reasonable cost. Erroneous decisions at this point can have significant impact on the success of a future system. An example of this is the introduction of HIPERLAN/1, where after many years of successful standardization, only one product came on the market, see http://www.proxim.com. At the time of standardization, it was believed that an equalizer would solve all observed transmission problems. Although equalizer algorithms are well explored in literature and showed very promising behavior in simulations, once the equalizer was to be implemented, the realization became very difficult. Particular solutions had to be investigated [3, 4, 5] in order to meet the stringent requirements, which finally turned out to be far too expensive to become a successful product.

A *quick-and-dirty* implementation of a proposed system or at least of its critical components (sometimes referred to as "rapid" prototyping) is often a means to detect potential problems early and thus de-risk a future product development.

34.1.1. Overview of the chapter

After some motivation and a brief classification of prototypes in general, this chapter first gives an overview of some of the recently presented MIMO prototypes. Subsequently hardware issues that need to be considered for a successful implementation are considered for SISO and later in particular for MIMO systems. The last section concentrates on rapid prototyping design methodology and tools.

34.1.2. Implementation types

Three types of systems can be identified, which from the management perspective serve a different purpose and from an engineering perspective are based on very different design and implementation requirements. It therefore makes sense to distinguish between the three experimental platforms based on the following definitions which are derived from their use in the literature [6] and from the dictionary.

(1) A *prototype* is the initial realization of a research idea or a standard, either as a reference, a proof of concept, or as a vehicle for future developments and improvements. As opposed to a simulation, it is not an imitative representation of the device. Instead it has significant similarities. In industry, a migration into a product is often intended.

(2) A *demonstrator* mainly serves as a sales vehicle and to show technology to customers. In general it will implement a new idea, concept, or standard that has already been established and has been finalized to some degree. Requirements on scalability are therefore less important than its functionality and often the required design time.

(3) A *testbed* on the other hand is generally used for research. It is a vehicle for further developments or for verification of algorithms or ideas under real-world or real time conditions. This results in the requirement for scalability, modularity, and extendibility.

A second criterion that allows a categorization of an implemented system is the real-time aspect. Most systems that have been implemented operate based on one of the following two paradigms.

(1) *Offline processing* schemes have the least stringent real-time requirements and are in general the easiest to implement. They have therefore been frequently used in a number of testbeds [7]. These schemes are often also called record and playback. They are based on offline preparation of a radio frame that is then sent in real time through the channel. A simple device records the received signal and all further processing is done offline without any timing constraints. The most stringent limitation of such a system is its inability to support feedback loops between

transmitter and receiver. Also, no (or only little) information will be gained about the actual implementation complexity of an algorithm.

(2) *Real-time* processing on the other hand generates the transmitted signal on the fly. The receiver processing takes place at essentially the same rate at which data is transmitted. This allows for a continuous operation of the system, which is its main characteristic and difference with respect to the offline-processing approach. However, at the same time it should be noted that real-time operation does not necessarily require the system to operate at its final target frequency or bandwidth. This is, for example, of interest if an appropriate scaling of all transmitter- receiver-, and channel-parameters is possible (e.g., when operating through a channel emulator).

34.1.3. Justification of a prototyping effort

It has been widely recognized that prototype implementations and field trials are an essential part of the verification of new system concepts, standards, and products. Consequently, extensive prototyping efforts have been carried out for today's single-input single-output (SISO) systems, such as GSM/GPRS, WLAN, HIPERLAN, and UMTS/HSDPA.

While the effort for a MIMO prototype is significantly higher than for a SISO system, its justification is also much more obvious and is founded on five critical issues.

(1) The availability of multiple antennas at the transmitter and at the receiver allows for a variety of new algorithms. Each of them has particular advantages and disadvantages and the optimum choice depends strongly on the actual channel conditions and the overall system context. Due to poor knowledge of the behavior of the MIMO channel, simulations often do not reflect them accurately and real-time/real-world experiments as well as field trials are necessary.

(2) In addition to the more realistic channel conditions and operating environment, a prototype also has the advantage over simulations of providing a significantly more exhaustive data set. The full or near real-time operation allows to test a system with a dramatically higher number of scenarios that can never be fully covered in slow computer simulations, especially in the high-SNR (low-BER) regime.

(3) It is also noticed that computer simulations often make numerous assumptions and rely on mathematical models which may turn out to be overly optimistic. Examples are the assumption of perfect channel- or even SNR knowledge at the receiver.

(4) From an implementation point of view, prototyping efforts have the advantage of pointing out complexity issues early in the design cycle of a new product or even already during standardization. Critical implementation issues are discussed from the beginning and delays due to unrealistic specifications (as in the rollout of UMTS) are avoided.

(5) Demonstrators are also important from a marketing perspective, as they show the ability of the company or product division to realize the latest technology.

34.1.4. Prototyping and SDR

Numerous synergies are recognized between rapid prototyping of communication systems and what is generally referred to as software-defined radio (SDR), see http://www.sdrforum.org/index.html. Both efforts aim at the implementation of a radio system on a generic reusable platform. A significant portion of the available literature in both fields is therefore of interest. A difference between the two subjects can be seen in the level at which the development is conducted and in the constraints under which the efforts are made.

For rapid prototyping (RP), design time of a new system is most critical and often more important than its cost, form-factor, or power consumption. A major focus is therefore on the efficiency of the tools that are being used in the development process. The hardware is built on standard components, flexible *at design time*.

In an SDR project, an efficient and flexible hardware platform capable of being reprogrammed or even reorganized (reconfigured) instantly *after design time* is the key to success. While of course suitable tools need to be available that allow to use the platforms capabilities, they are less crucial than the hardware itself and the supporting real-time operating system (RTOS). Consequently, most SDR projects are concerned with the development of flexible processing devices in order to meet the stringent constraints of mobile communication. Once such HW devices are available, they are interesting candidates for the hardware of a testbed. On the other hand, tools from RP-efforts may eventually be considered as valuable support tools for an actual SDR platform, which itself might have been developed and tested initially in an RP-environment.

From this short discussion it can be concluded that RP and SDR efforts complement each other quite well and will profit from each other.

34.2. Existing systems

In 1998 Bell-Labs reported the first narrowband MIMO prototype using the V-BLAST architecture. The prototype was built to obtain initial measurements to verify the predicted gains [8] of MIMO transmission. The system supported 8 transmit and 12 receive antennas; signal processing was performed offline on a frame-by-frame basis on a Pentek platform using TI-C40-DSPs [9].

Between 2000 and 2002, the first prototypes appeared applying MIMO concepts to broadband systems. In particular Lucent had shown the application of MIMO as an extension to the CDMA-based UMTS-FDD downlink. As opposed to the original V-BLAST prototype the new design used a combination of DSPs and FPGAs and was able to operate in real time [10]. IOSpan Wireless was the first company to develop and demonstrate MIMO-OFDM system prototypes as a basis for products that were then tested in field trials. Some of the results from this work are summarized in [11].

Table 34.1. List of selected MIMO prototypes.

	Tx/Rx	Mode	Platform	F_c/BW	Year	Ref.
Lucent	8/12	Offline	Pentek C-40 DSP	1.9 GHz 30 KHz/QAM	1998	[8]
Virginia Tech.	2/2	Real time	TI-EVM Boards	2.05 GHz 750 KHz	2001	[14]
IOSpan	3/2	Real time	ASIC FPGA/DSP	2.5 GHz 2 MHz	2001/02	[11]
Lucent/ETH	4/4	Real time	Custom FPGA/DSP	2 GHz 5 MHz/CDMA	2002	[10]
Lucent	4/4	Real time	ASIC FPGA/DSP	2 GHz/5 MHz 5 MHz/CDMA	2003	[13]
Brigham	16/16	Offline	PC	2.4 GHz	2003	[15]
Rice Univ.	2/2	Real time	Nallatech FPGA	2.5 GHz 20 MHz	2003	[16]
HHI	4/5	Real time	DSP/FPGA	5.2 GHz/12.5 MHz	2003	[17]
TU-Eindhoven	3/3	Offline	PC	5 GHz/20 MHz 20 MHz/OFDM	2003	[18]
ETH	4/4	Real time	Hunt Eng. FPGA	2 GHz 20 MHz/OFDM	2004	[19]
TU Wien	4/4	Offline	Sundance DSP/FPGA	2.45 GHz 6 MHz	2004	[20]
IMEC	2/2	Real time	FPGA ASIC	5.2 GHz 20 MHz	2004	[12]

Since then a number of universities have developed their own testbeds and experimental platforms to perform MIMO measurements and real-time experiments. Many of these systems (in particular those considering wideband systems) are based on offline processing of sampled data and use commercially available prototyping platforms. These again are often intended to perform channel measurements and to verify the performance of algorithms without considering implementation issues. Consequently, the usage of feedback is generally not possible.

However, some implementations are also capable and specifically designed for real-time operation, either to demonstrate adaptive modulation and other algorithms involving feedback or to also consider implementation complexity. An example for the former is the demonstrators reported by the Heinrich Hertz Institute (HHI) or for the latter the MIMO-OFDM demonstrator, developed at the Swiss Federal Institute of Technology (ETH). Another testbed [12], developed at IMEC, implements real-time operation of a 2×2 OFDM system with up- and downlink to demonstrate and evaluate MIMO processing at the transmitter.

Prototyping has also been used in conjunction with ASIC design [13] to demonstrate and test their functionality. Thereby the ASIC contains a critical part of the entire system and is supported by other ASICs, FPGAs, DSPs, and microprocessors.

An overview of some existing MIMO platforms is given in Table 34.1. Note that the distinction between *real-time* and *offline* processing in the table only refers

to how the platform has been used in recent publications, while some of the described offline hardware platforms also do support real-time signal processing. Also note that work in this field is also conducted at numerous other universities, research institutes, and companies. An extensive list of prototypes and products, which use smart antennas on only one side of the link to extend the capacity of existing systems is given in [21].

34.3. Hardware platform

34.3.1. Processing architectures and partitioning

Today, three main processing options are available for a prototyping hardware setup in wireless communications: FPGAs, high-performance DSPs, and microprocessors. Each of them is most suitable for a particular kind of operation and typically takes a particular role in a wireless communication system.

FPGAs are most suitable for high-rate regular datapath-dominated operations (e.g., despreading) that can be carried out in fixed-point arithmetic with relatively short word lengths. They also allow the efficient implementation of simple but real-time critical control functions that can often be realized efficiently in finite state machines with a limited number of states. FPGAs are found in almost all modern prototype designs. The amount of functionality that is realized by them generally grows rapidly as the real-time aspect becomes increasingly important. In a prototype design for a future ASIC implementation, FPGAs have the advantage that the code can often be mapped directly onto an ASIC process with only minor modification, which greatly eases the verification and reduces the coding time. While different FPGA vendors exist, the market is clearly dominated by the Xilinx-Virtex(II) series and the Altera Apex and Stratix series (see http://www.xilinx.com and http://www.altera.com), due to their very high capacity. Devices with integrated processors are also available. Both vendors focus on wireless communications for which the devices are well suited in prototyping. From the examples reported in the literature it appears that Xilinx devices are used more frequently in research prototypes.

DSPs are also mostly suited for regular arithmetic operations but impose less constraints on the regularity of an algorithm and thus also support control-dominated functions. Some DSPs are also able to efficiently carry out floating-point operations, which makes them a particularly appealing choice for numerically critical algorithms. From the rapid prototyping point of view, they are easier to program as compared to FPGAs. Their ability to directly use code from high-level simulations (typically ANSI C or even C++) makes them attractive wherever they provide sufficient performance.

Microprocessors (usually in a host PC) are most frequently used in offline processing prototypes, where no real-time constraints apply for the actual processing of the data. In this configuration, only little knowledge is gained about the actual complexity of an algorithm, but this is often also not of interest. The large variety of available tools greatly simplifies in general the implementation. In a real-time

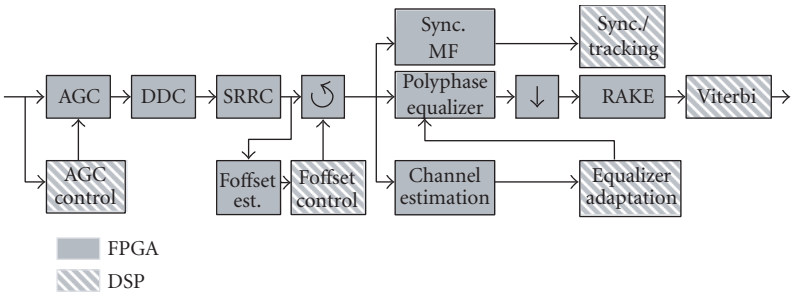

Figure 34.1. Partitioning example of a UMTS-FDD receiver.

prototype, microprocessors usually handle the higher layers of the protocol stack and interfaces to the outside world.

34.3.2. Partitioning

The previously discussed properties of the processing components dictate the partitioning of the system to a large extent. As an example, a block diagram of the physical layer of a UMTS-FDD downlink is given in Figure 34.1. It can be seen that almost all of the straightforward sample- and chip-rate processing is done in the FPGAs, covering a significant part of the overall complexity [22]. The DSP handles low-rate control functions and those parts of the design that are under development and require constant modifications. Important criteria for the partitioning are also the real-time and bandwidth constraints of the communication between the functional blocks.

34.3.3. A basic setup

Building a dedicated and highly optimized hardware platform from scratch for a particular system appears initially often to be a cost effective and powerful solution. However, the effort for the development of the hardware itself and especially of suitable middleware (device drivers and interfaces) and the debugging of the system is enormous and reliable schedules are hard to make. This is particularly true for systems that combine different processing architectures such as DSPs, microprocessors, and FPGAs. Such risky development may jeopardize the entire project. As a consequence, commercially available or existing proprietary processing or even dedicated prototyping platforms are mostly used. These are available from a variety of vendors offering configurations from data acquisition modules with virtually no processing capability over dedicated FPGA or DSP modules to flexible systems that can be equipped with different processing elements, according to the individual requirements. Widely used are products from Sundance, Pentek, Nallatech, Spectrum Signal Processing, Hunt Engineering, and many others. Some companies offer complete design environments, consisting of hardware and

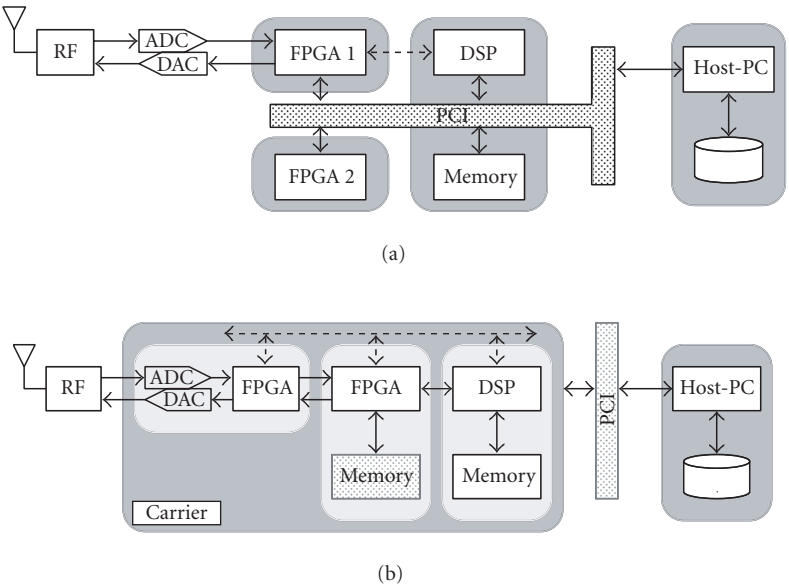

FIGURE 34.2. Typical architectures of commercial prototyping platforms: (a) common bus structure and (b) point-to-point interconnect of a carrier/module-based system.

software either for system development (such as Lyrtech) or for so-called presilicon prototyping of ASICs (such as Aptix). However at the same time, custom-designed platforms are often still of interest in the industry for customer demonstrations and field trials.

Whether a proprietary system is to be designed or a commercially available prototyping platform is to be used, a few basic points should be considered carefully.

System topology and connectivity. For the topology of a system two basic approaches are generally considered (see Figure 34.2). The first integrates all components of the system as independent modules, connected through a standard bus or point-to-point links [23, 24]. Especially the bus-based architecture allows to combine products from various vendors and provides a maximum flexibility. It also allows seamless access from a DSP or microprocessor to the remaining hardware which appears as part of its address space. This greatly facilitates the implementation of interfaces and improves the operability of the hardware for debugging or performance analysis. The common bus architecture typically resembles an architecture as it would most likely be used in a future system-on-chip (SoC) integration. However it also quickly becomes a performance bottleneck that ultimately limits the extension of such systems.

The second alternative is a modular (*carrier-based*) system which is often realized on the basis of a carrier board extendable by different (DSP, FPGA) modules. This type represents the majority of the commercially available prototyping

platforms. Such systems often use a more restrictive proprietary module interconnect scheme (such as the Hunt Engineering HEART system, the Sundance SHB-bus, or the Pentek VIM interconnect) on the basis of point-to-point links or ring structures supported by FIFO-interfaces. While this often provides a significantly increased bandwidth between the components, it reduces the flexibility and limits the use of boards from other vendors. Moreover, it often requires tedious protocols to allow a mapping of hardware components into a DSP or host processors address space. Consequently, the observability of the hardware states is reduced and significant interface changes are required when a successfully tested prototype is to be integrated as an SoC. A combination of a bus architecture with high-speed point-to-point connections based on a standard bus is therefore most desirable.

Integration of custom components. The ability to integrate custom boards or components into a system is a great advantage. On one hand it allows to supplement an existing system with custom ASICs implementing functions exceeding the capabilities of the prototyping system. On the other hand it allows the verification and debugging of custom boards or circuits in the system for which it was originally developed.

Standalone operation. While for a pure R&D platform a standalone operation is not of utmost importance, its marketing value in a technology demonstrator setup should not be underestimated. Depending on the field of application it is therefore important to consider the ability of a system to function and startup autonomously, for example, from a configuration in a flash memory or (E)EPROM.

Real-time capability. As a system needs to be partitioned into multiple DSP devices, real-time capability becomes an important aspect. The resulting requirements are essentially low-latency links between the components or boards with predictable deterministic delays. For systems which are partially based on DSPs and/or processors, interrupt capabilities are also of significant importance. A typical example is guaranteeing real-time constraints in time critical control functions such as in the IEEE 802.11 medium access control layer.

Memory. In particular when real-time processing is not required or a system can tolerate some latency, block processing has proven to be significantly more efficient than stream processing. However, on the other hand this gain often comes along with large memory requirements that may exceed the already significant amounts of storage on today's FPGAs. It is therefore advantageous to ensure that sufficient additional memory is or can be made available on a system.

Clocks and synchronization. In particular in a carrier-based system with multiple modules, clock distribution and synchronization becomes an important issue. If a common reference clock cannot be distributed between modules, resynchronization measures have to be taken to effectively design a system that works as an ensemble of locally synchronous, but globally asynchronous components. This can often be tedious and error-prone and adds additional hardware and latency, especially in systems that use a stream based processing (e.g., CDMA).

Baseband versus digital IF. In a SISO system, convincing arguments can be made for both digital baseband and digital-IF architectures. A digital-IF architecture, however, is known to avoid a number of problems such as carrier leakage and IQ-imbalance and is therefore often preferred. An appealing choice seems to be an approach, where the low-IF mixer of a superheterodyne receiver is substituted by bandpass sampling using an appropriate IF. Similarly at the transmitter, an image of the digital IF would be used directly as the low IF. The disadvantage of such a topology is its sensitivity to jitter, which requires high-quality clock sources for the converters. The resulting requirements often exceed the specifications of the DLLs that are usually contained in FPGAs. A thorough frequency planning is also required that leaves almost no room for scaling of the bandwidth of the signal if the system cannot immediately operate at its final speed.

34.3.4. Additional requirements for MIMO systems

The previous section discussed some basic requirements for the hardware of a generic prototyping platform. However, the systems considered so far were not particularly assuming multiple transmit and receive antennas; their constraints would equally apply to single-antenna systems. For the realization of a multiple-antenna system, some additional points need to be considered.

Baseband versus digital IF. In addition to the general arguments that were already presented for the SISO case, the reduced number of converters needed for a digital-IF realization becomes another strong argument in a MIMO scenario. The reduced cost, size, and number of connections needed usually compensate clearly for the higher sampling rate requirements which, for reasonable system bandwidth, still lie in the range of numerous available devices.

Clocks and synchronization. As the number of antennas grows, it becomes often necessary to distribute DA/AD conversion to multiple boards. As many MIMO systems rely on the in principle reasonable assumption of fully synchronized antennas, a timing or frequency offset can often not be tolerated. These constraints apply equally to the transmit and receive chains, but generally not between them. Having all transmit chains on one board and all receive chains on the other should therefore be strongly preferred over a solution where A/D and D/A converters are combined and the antennas are distributed over different boards.

Bandwidth and storage. A further important point is the significant increase in internal communication bandwidth in MIMO systems. While data rates between data acquisition and processing modules grow only linearly with the number of transmit/receive antennas, most existing platforms have been designed with SISO systems in mind. In a MIMO configuration, many of them quickly approach their performance limits even with moderate number of antennas (such as in a 4×4 system). In offline processing systems more high-speed storage needs to be provided close to the converters. However for some parts of a MIMO system, in particular for the channel state information (CSI), storage requirements grow quadratic

Table 34.2. Rate and storage requirements for future MIMO systems.

	$N_T \times N_R$	Baseband data rate	Raw bitrate	Samples for 1 frame	Words for 1 set of CSI
HSDPA	1×1	15.36 Msps	4.8 Mbps	150 K	64
	4×4	61.44 Msps	19.2 Mbps	600 K	1024
802.11a	1×1	20.00 Msps	54 Mbps	107 K	64
	4×4	80.00 Msps	216 Mbps	427 K	1024

with the number of antennas. Two examples are given in Table 34.2. They consider a MIMO extension of HSDPA and 802.11a with four antennas on both sides of the link. The former assumes an oversampling factor of four while the latter applies no oversampling. For the channel, 64 taps are considered in both cases. In an offline processing setup, for example, the on-chip memory of a large FPGA would be sufficient to record an entire SISO frame, while the corresponding MIMO configuration would require additional external memory. Similarly with some effort, the PCI bandwidth would allow a direct transfer of the baseband samples to the PC for a SISO system, while in the MIMO setting the PCI bandwidth would not be sufficient.

Module connectivity. A typical partitioning, especially in an open MIMO system, often allows for multiple data acquisition modules each of which supports only one or two channels. Many of these modules (as they are available) combine ADCs (DACs) with some limited processing capability through an FPGA. This is desirable to allow, for example, digital up-/down-conversion (DUC/DDC), automatic gain control (AGC), or subsampling algorithms to be placed as close to the converters as possible to avoid unnecessary data traffic to the main processing units. However, in MIMO designs some of these functions need to be coordinated among the different modules. It is therefore important that in addition to the high-speed data links they also provide a means for communicating with each other and to exchange control information. Direct coordination is typically only required within the receive and transmit chains, but not between them. Combining all transmit chains on one board and all receive chains on another should therefore again be preferred over a mix of both, distributed over two or more boards.

Processing requirements. Often MIMO systems are derived directly from similar SISO systems. Therefore, the actual MIMO processing is typically separated from the remaining part of the transmitter/receiver. The complexity of the latter generally scales linearly or quadratically (for channel estimation and equalization related parts) with the number of antennas. However, the complexity of the MIMO decoding itself (typically matrix inversions or ML-detection) can quickly become the dominant part. A quantification of the processing requirements, however, is certainly difficult to make as it strongly depends on the implemented system and algorithms. Therefore, it is even more important in MIMO systems to ensure an open system architecture that can be extended with new processing devices, which

can be connected to each other with sufficient bandwidth. It is also noted that FPGA technology evolves quickly and it should always be ensured that an upgrade to new devices remains possible without fundamental architecture changes.

34.3.5. System evolution: from SISO to MIMO

As mentioned before, most MIMO systems are based on an underlying SISO modulation scheme which by itself (e.g., in wideband systems) can be quite challenging to implement. It therefore appears most useful to develop a MIMO prototype on the basis of a suitable successfully tested SISO link implementation. Good example are CDMA or OFDM-based systems, where a significant amount of the processing is merely a replication of a corresponding SISO link from an algorithmic point of view [25]. This is generally exploited in DSP-based prototypes. However, for FPGA-based system components, simple replication is often not possible due to area constraints and/or because it is simply not economic. In this case interleaving of the multiple data streams is an interesting option. While this leads to a higher clock rate and still requires more storage elements in a design, it avoids replication of complex functional units (e.g., multipliers) and thereby leads to more efficient designs. Some FPGAs even provide dedicated hardware inside their configurable logic blocks (CLBs) for interleaved processing of multiple streams (e.g., Xilinx Virtex FPGAs). In this context it should be noted that interleaving is even recommended by FPGA vendors as a means to improve the utilization of their devices and to increase throughput through the resulting pipelining. This is founded on the observation that sampling rates even in wideband systems are often far below the capabilities of today's FPGA technology. An example is the straightforward realization of transversal SRRC filters in a four times oversampled UMTS system, which will run at a clock of less than 16 MHz, while 64 MHz are easily achievable on most commercially available FPGAs. It is noted that interleaved processing of multiple independent streams allows even pipelining of recursive algorithms, without causing stalls due to data dependency problems (e.g., multiple FFTs/IFFTs).

34.3.6. Channel emulation

While in most cases the eventual goal is the demonstration of a system under real-world and real-time conditions, this might sometimes not be feasible or is not even required. In particular when a system is designed for high mobility and large cell sizes, practical experiments are major cost factors that are often overlooked or underestimated at the beginning of a project. Only a few of them will be briefly mentioned here to undermine the argument for the necessity of channel emulation.

(i) Operating licenses for a particular spectrum need to be obtained, as power restrictions in the ISM bands do not allow for experiments with larger cells.

(ii) Power amplifiers with their undesired nonlinear behavior and other expensive RF equipment are needed.

(iii) Prototyping systems are usually not suitable for mobility and mechanical problems are most likely to occur.

Moreover, real-world scenarios are inherently uncontrollable and unpredictable and therefore particularly unsuitable for the initial analysis of a new algorithm. Instead, artificial but repeatable conditions are necessary to obtain an intuitive understanding for a prototype's behavior and to be able to compare it to other alternatives and simulations. From a functional verification point of view, debugging also calls for the ability to reproduce a problem and its history as accurately as possible to be able to analyze its origins. Therefore in addition to the control over channel conditions, it might be desirable to have full control over the noise in the system. This is only possible if all analog components are eliminated. Consequently, channel emulation is an essential development method that cannot be avoided and fully digital-IF or baseband interfaces are definitely a helpful feature. SISO systems have been available for quite some time. However, emulators with a sufficient number of channels for the emulation of a MIMO system have only emerged recently (see http://www.elektrobit.com and www.spirentcom.com) and are still extremely expensive. The main reason is the high complexity. To put this into perspective, the number of sample-spaced taps that can be emulated in a state-of-the-art FPGA (Xilinx XC2V6000-6) is illustrated in the following, assuming a bandwidth of 20 MHz. In a 1×1 system, the 144 integrated real-valued multipliers suffice to support 144 taps, whereby each multiplier is timeshared to perform a complex multiplication. In a 2×2 system only 36 taps can be emulated. In a 4×4 system the limit comes down to only 9 taps.

34.4. Design flow and tools

While many different HW platforms for prototyping exist, only little attention has been paid in the past to design methods and utilities. On the one hand expensive electronic design automation (EDA) tools for chip design were used, on the other hand particular DSP chips and FPGAs always come with development tools, some even being excellent. As long as cost and complexity of a design is not an issue, such tools can offer excellent design environments. However, in wireless MIMO prototypes, the designer has to realize very high complexities of the involved algorithms and thus a design methodology supporting a team effort as well as different hardware targets becomes more and more important. This section will discuss requirements of such a design methodology and give an overview of existing flows and available tools.

34.4.1. Requirements

Rapid prototyping of wireless MIMO systems faces similar problems as today's chip design in wireless: *high complexity algorithms* are to be implemented in a *short time* by a *design team*. While high complexity can easily be solved by parallel systems, in particular in prototyping where size and power consumption is not so

much of interest, short implementation time, changing specifications, and team effort, clearly require some organization.

In a classical chip design cycle [26], several design teams are working more or less interconnected one after the other using different design environments and languages, always those that suit their needs best. Since such languages and environments are incompatible, manual work is required to translate intermediate design results from one team to the next. Naturally, such a procedure is inconsistent, unnecessarily slow, and error-prone.

In [26] five concepts to improve design efficiency in a rapid prototyping design flow have been specified:

(1) *one* design environment,
(2) *one* automatic documentation by specification,
(3) *one* forward-backward compatible code revision tool,
(4) *one* code to be worked on by refinement steps,
(5) *one* team, to improve communication.

This paradigm is called the *five-ones approach*.

(1) It is shown that the design effort appears as a feed-forward system, handing the design over from one part of the design team to the next and a feedback loop concerning communication about the common design goals, and progress achieved. Such feedback slows down the design process. Since communication is required, the feedback loops cannot be broken. Also the required skill sets are not present in all parts of the teams. However, the response time can be changed dramatically, once all design teams share **one-design environment**. This observation on its own is not new and many tools in the EDA community exist (Simulink, COSSAP/CoCentric System Studio, and SPW to name the most widespread). However, since they have been developed to support specifically chip design (and to some extent algorithmic design) the architectural level and its exploration, as well as testing and system integration on specific hardware platforms (so-called platform-based design) are not supported. Also, due to specific language constraints, many researchers refuse to use such systems, since they believe their productivity is dramatically reduced by them. On top of that, high license costs prevent many companies from using them throughout the entire design chain.

(2) A second aspect is the missing documentation of the research part of the design team. While such people focus on meaningful results of their simulations, the following design teams are mostly interested in functional specifications. Graphical systems like COSSAP/CoCentric System Studio, SPW, and Simulink offer the possibility to define functional blocks with clearly defined input and output ports and corresponding data rates. Using such a graphical system **induces a documentation** while specifying the functional blocks. Specification allows detection of flaws at an early stage and, much more importantly, it can be used to supply additional information into functional descriptions of algorithms. The graphical description has further advantages: it avoids global variables. Global variables can lead to the undesired effect that information from the transmitter and/or channel is known at the receiver and some (undesired) cheating can be the result. It is, for example, quite common in the literature to present MMSE receivers having

perfect knowledge of the signal-to-noise ratio (SNR), while this value in reality needs to be estimated. The problem of estimating such a value is typically underestimated. In addition to the graphical specification, COSSAP, for example, allows writing so-called *Generic-C*, an ANSI C program enriched by essentially a header specifying the names, types, and rates of the input and output variables, a feature well preserved in the so-called PRIM models of the new version CoCentric System Studio.

(3) A third aspect of the slowdown in the product flow is the required permanent recoding. Although the research part of the design team defines a code for simulation, the system design team is not able to reuse the code, mainly due to its poor documentation and coding style used. Based on the anticipated hardware platform, other languages (assembler, VHDL) have to be used at a certain level of refinement, requiring time-consuming hand recoding. Such foregoing is error prone and requires a solution based on automatic recoding tools. See, for example, [27] where it is claimed that errors found late in the design process cost up to 100 times more than those found early. While graphical tools do not provide an immediate solution to the tedious recoding process, they can support it by allowing multiple, but different descriptions for each block. This alleviates the transformation and allows doing it piece by piece. Simulink, COSSAP/CoCentric System Studio (see http://www.synopsys.com), and SPW (see http://www.cadence.com) allow for such code versions, but only on a block level, that is, if a modification impacts several blocks at the same time, the system does not work out the required consistency.

Revision control tools (such as CVS, see http://www.gnu.org/software/cvs/, or ClearCase, see http://www.rational.com/products/clearcase/index.jsp) can help here. At certain times in the design flow, the entire design becomes frozen and can be reinstantiated at a later time allowing to track a bug that shows up at a certain (refinement) stage in the design flow, but was not noticed before. Further aspects of revision level tools are personal responsibility: the blocks can be assigned to specific people in the team and cannot be altered by others. Using such blocks, it is guaranteed that everybody in the team is working in the same, rather than in a personal environment. In order to guarantee backward compatibility, it is important to stick with one code and the same language for as long as possible.

(4) Furthermore, graphical systems allow an easy method of **code refinement** by cosimulation. The code can be refined from one revision level to the next and by instantiating the two versions at the same time, their output can be compared while they are fed by the same input. An important step in code refinement is the switch from float to fix-point code. The recent *SystemC* initiative [28], and also see http://www.systemc.org supports this step by extending ANSI C with fix-point data types. A|RT-Library from Adelante Technologies (now owned by ARM) offered such C++ Library extension for many years. The underlying idea is that by providing more and more details in the code at a very high level, the automatic tools modify the code iteratively into the required (meta-) descriptions until the final product is defined in every (technical) detail. These final descriptions express the code in a desired form, that is, assembler code to program a DSP chip, VHDL,

or VERILOG code to program an FPGA or synthesize the required masks for an ASIC. However, since the automatic tools map the code from a high level to each of the lower levels, the refinement of the code is only performed on the high level description, that is, the C-code. By iteratively rewriting the original C-code used in simulation to suit the needs of a specific hardware platform, the code remains backward compatible at any state and thus allows all teams to share the code and investigate problems. Specifically, there is no need to switch design environments or test benches when transferring from one team to another. While commercial tools rarely support hardware-in-the-loop, this becomes an important feature to check proper functionality and should be a requirement for future prototyping platforms. In [26], a so-called real-sync method is reported which allows to map DSP algorithms automatically from Simulink to TI C6X DSPs and run them there while the rest of the environment still runs under Simulink.

(5) One last aspect when analyzing the slow development process is the team size. Poor communication is a drawback of rather large teams. Fortunately, the required amount of people in a prototype team is much smaller and it is possible to keep all team members as **one team** supporting full information to everybody. This is clearly a particularity in rapid prototyping that cannot easily be realized in a large product design team.

34.4.2. Existing tools

DSP providers, in particular TI (see http://www.ti.com) offer a rich design environment that can be customized to specific DSP evaluation boards, including also real-time operating systems. Due to the excellent C-compiler and optimizer, signal processing procedures can be specified in a high-level language and mapped automatically onto a DSP. Many prototyping platform providers take advantage of this environment and customize their platforms accordingly. Mathworks (see http://www.mathworks.com) offers Simulink, a graphical interface with a rich library of toolboxes and fixed-point design tools, easing the design of communication algorithms. The recent introduction of cosimulation links between Matlab/Simulink and Mentor Graphics' popular ModelSim VHDL simulator helps to close the gap between algorithm and VHDL simulations. Most interesting is also the real-time workshop supporting automatic mapping of Simulink designs into C code for TIs C6x and other processors. However, so far the efficiency of the coding is quite limited. Harsh conditions on real time as they are common in wireless designs are not supported. FPGA providers like Xilinx and Altera enriched Simulink with their own libraries, called Xilinx System Generator and Altera's DSP Builder, respectively. They allow to simulate predefined DSP functions under Simulink and map the designs directly onto the corresponding FPGA chips. These tools lead to quite efficient FPGA designs, provided the DSP functions are available in the libraries. Especially, data-flow-oriented designs can be performed quickly, utilizing such tools.

34.5. Summary

The number of recently published papers on MIMO prototypes and testbeds by companies and universities is a clear indication that the effort is justified from a research as well as from a business point of view. It is therefore expected that the number of reported MIMO prototypes will significantly increase in the next years.

While on the hardware side the basic requirements for a successful realization are quite similar to SISO systems, more attention needs to be paid to proper synchronization of the system components, the increased communication bandwidth between them, and the higher processing and storage requirements. The variety of complex algorithms makes an open extendable system even more important than in the SISO case.

From the design methodology and implementation side, the more complex designs call for a well-organized design and prototyping methodology. Clearly defined interfaces between the design stages are most important. A common design environment using one language for high-level simulations and implementation based on successive refinements of the code is highly desirable.

Acknowledgments

Andreas Burg would like to thank his colleagues in the IIS and CTL at ETH whose work and experiences contributed significantly to the content of this chapter.

This work was funded by the Christian Doppler Pilot Laboratory for Design Methodology of Signal Processing Algorithms

Abbreviations

ADCs	Analog-to-digital converters
AGC	Automatic gain control
ASIC	Application-specific integrated circuit
CLBs	Configurable logic blocks
CSI	Channel state information
DACs	Digital-to-analog converters
DDC	Digital down-conversion
DLLs	Delay-locked loops
DSPs	Digital signal processors
DUC	Digital up-conversion
EDA	Electronic design automation
FPGAs	Field programmable gate arrays
GPRS	General packet radio service
GSM	Global system for mobile communication
HSDPA	High-speed downlink packet access
IF	Intermediate frequency
MIMO	Multiple-input multiple-output
PLLs	Phase-locked loops
RP	Rapid prototyping

RTOS	Real-time operating system
SDR	Software-defined radio
SISO	Single-input single-output
SoC	System-on-chip
UMTS	Universal mobile telecommunication system
WLAN	Wireless local area network

Bibliography

[1] B. Bailey, "The waking of the sleeping giant-verification," http://www.mentor.com/consulting/techpapers/mentorpaper_8226.pdf, April 2002.

[2] A. Hoffmann, T. Kogel, and H. Meyr, "A framework for fast hardware-software co-simulation," in *Proc. IEEE Conference and Exhibition Design, Automation and Test in Europe (DATE '01)*, pp. 760–765, Munich, Germany, March 2001.

[3] Y. Sun, A. R. Nix, D. R. Bull, et al., "Design of a novel delayed LMS decision feedback equaliser for HIPERLAN/1 FPGA implementation," in *Proc. 49th IEEE Vehicular Technology Conference (VTC '99)*, vol. 1, pp. 300–304, Houston, Tex, USA, May 1999.

[4] J. S. Wang, P. L. Lin, W. H. Sheen, D. Sheng, and Y. M. Huang, "A compact adaptive equalizer IC for HIPERLAN system," in *Proc. IEEE International Symposium on Circuits and Systems (ISCAS '00)*, vol. 2, pp. 265–268, Geneva, Switzerland, May 2000.

[5] Y. Baltaci, I. Kaya, and A. Nix, "Implementation of a HIPERLAN/1 compatible CMF-DFE equaliser," in *Proc. 51st IEEE Vehicular Technology Conference (VTC '00)*, vol. 3, pp. 1884–1888, Tokyo, Japan, May 2000.

[6] F. Kordon and J. Henkel, "An overview of rapid system prototyping today," *Kluwer Journal on Design Automation for Embedded Systems*, vol. 8, no. 4, pp. 275–282, 2003.

[7] T. Kaiser, A. Wilzeck, M. Berentsen, and M. Rupp, "Prototyping for MIMO systems: an overview," in *Proc. 12th European Signal Processing Conference (EUSIPCO '04)*, Vienna, Austria, September 2004.

[8] P. W. Wolniansky, G. J. Foschini, G. D. Golden, and R. A. Valenzuela, "V-BLAST: an architecture for realizing very high data rates over the rich-scattering wireless channel," in *Proc. International Symposium on Signals, Systems, and Electronics (ISSSE '98)*, pp. 295–300, Pisa, Italy, October 1998.

[9] D. Samardzija, C. Papadias, and R. Valenzuela, "Experimental evaluation of unsupervised channel deconvolution for wireless multiple-transmitter/multiple-receiver systems," *Electronics Letters*, vol. 38, no. 20, pp. 1214–1216, 2002.

[10] A. Adjoudani, E. Beck, A. Burg, et al., "Prototype experience for MIMO BLAST over third-generation wireless system," *IEEE J. Select. Areas Commun.*, vol. 21, no. 3, pp. 440–451, 2003.

[11] D. Gesbert, L. Haumonte, H. Bolcskei, R. Krishnamoorthy, and A. J. Paulraj, "Technologies and performance for non-line-of-sight broadband wireless access networks," *IEEE Commun. Mag.*, vol. 40, no. 4, pp. 86–95, 2002.

[12] M. Wouters, A. Bourdoux, S. Derore, S. Janssens, and V. Derudder, "An approach for real time prototyping of MIMO-OFDM systems," in *Proc. 12th European Signal Processing Conference (EUSIPCO '04)*, Vienna, Austria, September 2004.

[13] D. Garrett, G. Woodward, L. Davis, G. Knagge, and C. Nicol, "A 28.8 Mb/s 4×4 MIMO 3G high-speed downlink packet access receiver with normalized least mean square equalization," in *Proc. IEEE International Solid-State Circuits Conference (ISSCC '04)*, vol. 1, pp. 420–536, San Francisco, Calif, USA, February 2004.

[14] R. Gozali, R. Mostafa, R. C. Palat, et al., "Virginia tech space-time advanced radio (VT-STAR)," in *Proc. IEEE Radio and Wireless Conference (RAWCON '01)*, pp. 227–231, Waltham, Mass, USA, August 2001.

[15] J. W. Wallace, M. A. Jensen, A. L. Swindlehurst, and B. D. Jeffs, "Experimental characterization of the MIMO wireless channel: data acquisition and analysis," *IEEE Transactions on Wireless Communications*, vol. 2, no. 2, pp. 335–343, 2003.

[16] P. Murphy, F. Lou, A. Sabharwal, and P. Frantz, "An FPGA based rapid prototyping platform for MIMO systems," in *Proc. 37th IEEE Asilomar Conference on Signals, Systems, and Computers*, pp. 900–904, Pacific Grove, Calif, USA, November 2003.

[17] http://www.hhi.fraunhofer.de/english/mimo_at_globecom2003.pdf.

[18] A. van Zelst and T. C. W. Schenk, "Implementation of a MIMO OFDM-based wireless LAN system," *IEEE Trans. Signal Processing*, vol. 52, no. 2, pp. 483–494, 2004.

[19] H. Bölcskei, S. Häne, D. Perels, et al., "Implementation aspects of a real-time multi-terminal MIMO-OFDM testbed," in *IEEE Radio and Wireless Conference (RAWCON '04)*, Boston, Mass, USA, August 2004, http://www.nari.ee.ethz.ch/commth/pubs/p/RAWCON2004.

[20] E. Aschbacher, S. Caban, C. Mehlfuehrer, G. Mayer, and M. Rupp, "Design of a Flexible and Scalable 4×4 MIMO Testbed," in *Proc. 11th Digital Signal Processing Workshop 3rd Signal Processing Education Workshop*, pp. 178–181, Taos Ski Valley, NM, USA, August 2004.

[21] A. O. Boukalov and S. Haggman, "System aspects of smart-antenna technology in cellular wireless communications-an overview," *IEEE Transactions on Microwave Theory and Techniques*, vol. 48, no. 6, pp. 919–929, 2000.

[22] A. A. Kountouris, C. Moy, L. Rambaud, and P. Le Corre, "A reconfigurable radio case study: a software based multi-standard transceiver for UMTS, GSM, EDGE and Bluetooth," in *Proc. 54th IEEE Vehicular Technology Conference (VTC '01)*, vol. 2, pp. 1196–1200, Atlantic City, NJ, USA, October 2001.

[23] P. Murphy, J. P. Frantz, E. Welsh, R. Hardy, T. Mohsenin, and J. Cavallaro, "VALID: custom ASIC verification and FPGA education platform," in *Proc. IEEE International Conference on Microelectronic Systems Education (MSE '03)*, pp. 64–65, Anaheim, Calif, USA, June 2003.

[24] M. Vasilko, L. Machacek, M. Matej, P. Stepien, and S. Holloway, "A rapid prototyping methodology and platform for seamless communication systems," in *Proc. 12th IEEE International Workshop on Rapid System Prototyping*, pp. 70–76, Monterey, Calif, USA, June 2001.

[25] A. Burg, E. Beck, M. Rupp, D. Perels, N. Felber, and W. Fichtner, "FPGA implementation of a MIMO receiver front-end for UMTS," in *Proc. International Zurich Seminar on Broadband Communication (IZS '02)*, pp. 8.1–8.6, Zurich, Switzerland, February 2002.

[26] M. Rupp, A. Burg, and E. Beck, "Rapid prototyping for wireless designs: the five-ones approach," *Signal Processing*, vol. 83, no. 7, pp. 1427–1444, 2003.

[27] R. S. Janka, *Specification and Design Methodology for Real-Time Embedded Systems*, Kluwer Academic Publishers, Norwell, Mass, USA, 2002.

[28] W. Mueller, J. Ruf, D. Hoffmann, J. Gerlach, T. Kropf, and W. Rosenstiehl, "The simulation semantics of SystemC," in *Proc. Design, Automation and Test in Europe (DATE '01)*, pp. 64–70, Munich, Germany, March 2001.

Andreas Burg: Integrated Systems Laboratory, Swiss Federal Institute of Technology Zurich, Gloriastrasse 35, 8092 Zurich, Switzerland

Email: apburg@iis.ee.ethz.ch

Markus Rupp: Institute of Communications and Radio-Frequency Engineering, Vienna University of Technology, Gusshausstrasse 25/389, 1040 Wien, Austria

Email: mrupp@nt.tuwien.ac.at