DESCRIBING MIMO DESIGNS FOR RAPID PROTOTYPING IN THE BEE ENVIRONMENT

Kimmo Kuusilinna[1], Chen Chang, Hans-Martin Bluethgen, Danijela Cabric, Brian Richards, and Robert W. Brodersen

[1]Institute of Digital and Computer Systems, Tampere University of Technology
Korkeakoulunkatu 1, FIN-33720 Tampere, Finland (Europe)
phone: +358 3 3115 11, fax: +358 3 3115 3095, email: kimmo.kuusilinna@cs.tut.fi

ABSTRACT

Designing VLSI circuits for efficient stream processing implementations requires the designer to simultaneously consider both algorithmic and hardware architectural trade-offs. Rapid prototyping for such applications implies not only short design time but also satisfying certain design constraints, such as timing and silicon area. Direct-mapping is a method for reasoning about these issues and describing them unambiguously. The BEE (Berkeley Emulation Engine) with its rapid prototyping flow provides the platform for emulating a class of these systems in real-time.

1. INTRODUCTION

The Berkeley Emulation Engine (BEE) [4] is a hardware emulator that comprises twenty five-hundred-thousand-gate FPGAs and it is designed to facilitate the real-time prototyping of digital signal processing in communication systems. Real-time operation is important to allow the use of the same radio front-ends that are utilized in the final system. In addition, real-time operation is good at revealing possible timing problems in the design.

The BEE hardware architecture provides a homogeneous two-layer mesh interconnection structure, which facilitates the design of fast tightly coupled DSP implementations. The first layer is utilized for local connections and the second layer for board-level connections. System clock rates can reach 50 MHz during the emulation, and local connections can be even faster, enabling real-time behaviour for many highly parallel implementations.

A single BEE board has 2400 I/O pins for interfacing with external components, such as radio front-ends, or other BEE units. Depending on the utilized signalling standard (LVTTL or LVDS) and the off-board connection speed, this results in a theoretical I/O transmission capacity of 100-200 Gb/s. Particularly, the nature of the Multiple-Input Multiple-Output (MIMO) systems is such that large I/O bandwidths are typically required.

2. MIMO SYSTEMS AND PROTOTYPING

An exemplary application of BEE is the simulation and prototyping of MIMO communication systems. As described in the following section, this application benefits strongly from the computational resources and the high-bandwidth input/output capabilities the BEE environment provides.

2.1 Introduction to MIMO Systems

There has been a rising interest in multiple antennas for wireless communications systems after Foschini showed [2] that by exploiting the spatial characteristics of a wireless channel a communication structure with multiple transmit and receive antennas can increase its transmission capacity over the single antenna approach. Earlier systems used multiple antennas either on the transmitter (Tx) or receiver (Rx) side, and deployed beam forming and diversity techniques to obtain power gain and more reliable wireless links. However, having multiple antennas on both Tx and Rx ends creates a MIMO system in which parallel data streams can be transmitted in the same frequency band through spatial multiplexing [11]. Compared to a single antenna system (SISO) that operates in bandwidth B and has Shannon's capacity $C = B \cdot \log_2(1+SNR)$, MIMO transmission under the same bandwidth constraints can increase the capacity proportionally to the number of antennas (the minimum of the number of Tx and the number of Rx antennas).

In order to develop intuition for the capacity of an NxN MIMO system and how that capacity can be achieved, we will consider the following commonly used representation of a linear MIMO system:

$$y = Hx + z$$

where H is the NxN channel matrix, x is an Nx1 vector of transmitted data on N Tx antennas, y is an Nx1 vector representing the received signal on N Rx antennas, and z is an Nx1 vector of additive white Gaussian noise, whose power σ^2 determines the SNR in the channel. A singular-value decomposition (SVD) [3] of matrix H is:

$$H = U \cdot diag(d_1, d_2, ..., d_N) \cdot V^{*T}$$

where U and V are orthogonal matrices, that is $UU^{*T}=VV^{*T}=I$. If input x and output y are multiplied by V and U^{*T}, respectively, then the channel is decoupled into N parallel streams, that is N independent SISO channels as depicted in Fig. 1. Then, the MIMO channel capacity can be expressed as:

$$C = B\sum_{i=1}^{N} \log_2(1 + d_i^2 \frac{P}{N\sigma^2})$$

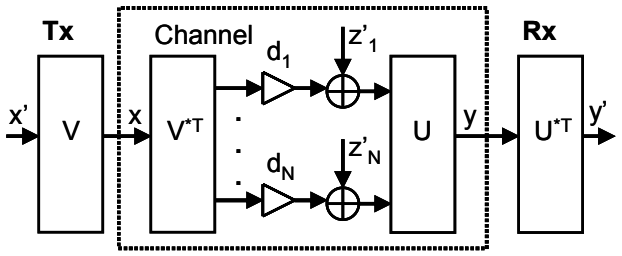

Figure 1: Block diagram of the SVD decoupling concept.

In practice, achieved data rates can be lower than predicted by the linearly increasing capacity growth due to many reasons that are subject of ongoing research. Important questions include the nature of the MIMO wireless channel model, the antenna array configurations, the channel estimation algorithms, and the implementation loss in the analog front-end and finite precision digital processing. To accurately model and experiment with these issues, a real-time wireless MIMO prototype is necessary.

Decoding a MIMO channel requires the estimation of an NxN matrix and solving of a system of N linear equations. There are numerous algebraic methods to solve this system, for example SVD, LU, Cholesky, and QR decomposition [3]. The first developed prototype of a MIMO system was V-BLAST, based on decision feedback detectors, which used successive cancellation based on QR decomposition of the channel matrix. However, this algorithm is sub-optimal because the channel is only known at the Rx side. For optimal performance, both Tx and Rx need to have knowledge about the channel. This can be achieved through SVD-based algorithms [10]. Nevertheless, the improved performance is a result of much larger computational complexity. Table 1 compares the number of operations required by SVD and QR algorithms in a 4x4 MIMO system.

Table 1: Computation complexity of MIMO algorithms.

	Addition/Subtraction	Multiplication	Division
SVD	344	455	47
QR	79	79	22

Channel estimation can be done through adaptive least square estimation techniques (LMS or RLS) [7] using a pilot sequence or even through blind algorithms. In addition, multiple antenna system design may include the shutoff of weak spatial channels that reduces the complexity and the size of the system, or adaptive modulation techniques, which both require some form of feedback from the receiver back to the transmitter.

Theoretical and simulation studies characterized multiple antenna algorithms under various channel models but less attention has been paid to implementation issues including architectural and circuit optimization, feasibility for integration on a silicon chip, and testing in real indoor wireless environment. Implementing the algorithm that provides best theoretical channel capacity might not always be feasible due to the computational complexity under real-time operation constraints. Our approach for implementation is to explore architectures of various multiple antenna algorithms, identify common building blocks, and choose solutions that are the most suitable for evaluation and test using the fast BEE prototyping platform.

2.2 Real-time Prototyping of MIMO Systems

Some properties of MIMO systems like the channel capacity can be derived by theoretical approaches. Other properties like the stability of adaptive algorithms under time-varying channel conditions or the effect of finite word length implementations need to be found out by running simulations. One of the important quality measures for communication systems is the bit error rate (BER). The number of cycles required for the simulation of an entire BER curve can easily be in the order of 10^{10}. Taking into account the number of operations per cycle typical for MIMO systems, large computational resources are required to run these simulations in reasonable time. With the BEE prototyping system, it is possible to map computation intensive parts of a MIMO system onto parallel functional units. For example, a matrix-vector multiplication can be mapped directly onto the appropriate number of multipliers and adders. By this direct-mapping approach in combination with the large number of FPGAs, the BEE system allows for simulating even complex MIMO systems in real-time.

By connecting radio frequency (RF) front-ends to BEE, it is possible to evaluate MIMO systems under real channel conditions. BEE provides the necessary I/O hardware in terms of the number of ports and the data bandwidth. The sample rate for a system that uses 20 MHz bandwidth has to be at least 40 MHz with I and Q channel having a resolution of 16 bit each. A 4x4-antenna system requires 256 I/O signals and a total bandwidth of 640 MB/s for either transmit or receive direction. A 16x16-antenna system requires 1024 I/O signals and 2560 MB/s in either transmit or receive direction.

2.3 MIMO System Setup

Fig. 2 shows a block diagram of an NxN-antenna MIMO system demonstrator that will be implemented using the BEE platform. This system employs the narrow-band channels of an OFDM modulation scheme and a multiple-antenna algorithm based on singular-value decomposition

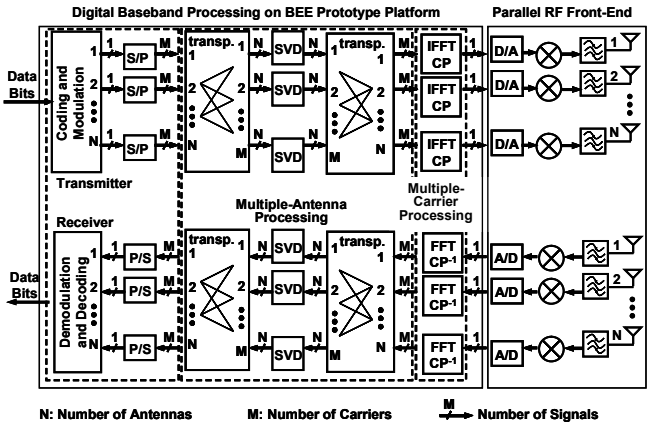

Figure 2: Block diagram of an NxN-antenna MIMO system with a parallel RF front-end.

[6]. The digital baseband processing of the transceiver, performed on BEE, connects to a scaleable parallel RF front-end with up to 16 transmit and receive antennas. First experiments with narrow-band transmission systems and RF front-ends have shown the feasibility of this approach [5]. The next step of setting up the parallel RF front-end is currently in preparation.

3. DESCRIBING DESIGNS FOR BEE

There are two basic emulation configurations for our MIMO system as depicted in Fig. 3. The first option is to emulate the whole transmitter-channel-receiver chain on a single BEE unit. This approach has the advantage that the conditions are very controllable and experiments can be conducted with different channel models. Disadvantages include limited decoupling of the transmitter and the receiver, not exercising the real radio front-ends, and possibly too idealized channel models. These issues can be addressed by using two BEE units as shown in Fig. 3b. Naturally, larger systems could be built using multiple BEE units; four units have been built so far.

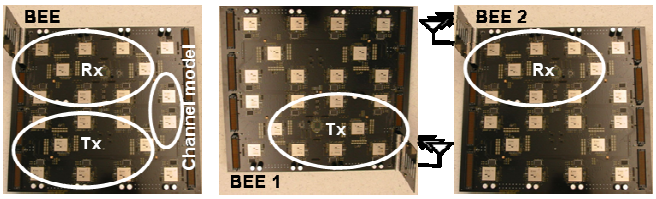

Figure 3: The basic conceptual emulation configurations.

3.1 Direct-mapping

Direct-mapping is a design method that simultaneously describes the functionality and the architecture utilizing blocks of sub-designs with well-known behaviour and hardware implementation.

Matlab Simulink [8] and Xilinx System Generator (XSG) [9] are used for design entry. The XSG provides a set of blocks that can be simulated in Simulink and implemented on a FPGA. Corresponding to the nature of Simulink simulations, the blocks are a good match for datapath components. Since the bocks include components for basic Boolean logic, control can also be implemented. However, this is somewhat cumbersome and, therefore, Matlab StateFlow finite state machines are the preferred way to embed control into the design. As seen from Fig. 2, the MIMO system to be prototyped is very dataflow oriented which suits the direct-mapping paradigm.

XSG design blocks are cycle and bit-accurate representations, but the internal hardware architecture may vary in different designs due to low-level architectural optimizations. Utilizing direct-mapping, the implementation form typically follows the function, which tends to result in highly parallel architectures. Parallelism allows the designs to run on slower clock speeds, which suits the 50 MHz platform constraint. The relatively low frequency clock allows us to emphasize low-power design practices and relatively straightforward ASIC conversion without compromising the necessary throughput.

Direct-mapped designs are easy and therefore fast to synthesize because the designer already made many of the architectural decisions during the design entry. Incorporated with the parameterization property of XSG blocks, this allows rapid experimentation with word-length effects of fixed-point computations. As mentioned earlier, this is important for validating the stability of the algorithm under realistic run-time conditions.

3.2 Coarse Partitioning and Routing

The MIMO implementation is much larger than the capacity of a single FPGA. Therefore, the implementation must be partitioned on multiple FPGAs. To parallel the component-level direct-mapping, the partitioning is the responsibility of the designer and it is achieved by grouping blocks into hierarchical sub-designs. The two primary concerns in this process are the amount of logic in a sub-design, which should normally be bounded by the logic and I/O capacity of the FPGA.

In the basic case, there are 48 signal lines between neighbouring FPGAs on the BEE board. This FPGA-to-FPGA routing and FPGA I/O assignment is done automatically by a dedicated program. Additional routing capacity can be achieved by time multiplexing signals or routing them through other FPGAs.

3.3 Extending the Blockset

The convenience of the direct-mapping method is quite dependent on the suitability and completeness of the underlying component library (blockset). If the desired component does not exist in the library, either a new block must be created or its functionality achieved by combining components that are more primitive. These are also the primary methods for increasing the expressiveness of the original blockset.

Sub-designs consisting of original blocks are convenient since it is easy to examine their construction and, if necessary, change the design to suit the new requirements. However, sometimes the sub-designs are so large or contain special structures that designing a new dedicated component is easier. An example of such a situation is a large adder tree. The System Generator environment includes a black box component that can be used to refer to user-defined designs in VHDL. Synopsys' Module Compiler is an effective way to describe these large datapath components and the concepts behind building a custom library are documented in more detail in [1].

3.4 Estimation

Due to the explicit architecture in direct-mapped designs, an experienced designer can usually estimate the design area just by adding up the estimated areas of all the utilized components. If the design blocks are on sufficiently high-level, this is not an unreasonable task. Furthermore, the designer can visually estimate the delays associated with combina-

tional sections and use this information to balance the logic depth between registers.

The same property allows us to estimate the design area automatically using a separate program. Simulink-level estimations are very important because they provide fast feedback on the feasibility of a design. The estimations are based on the pre-characterization of each block used in the libraries. This should be done exhaustively over all reasonable values of the block parameters. Similarly, the delay through each block and the energy required for the computation can be tabulated. However, this method has its limitations. It is not unusual for the Simulink-level estimation to be 30% off. However, system-level estimations can be run in a matter of minutes instead of hours or days that are typical for implementations of large designs. Although small performance differences in estimations at this level may not be relevant or even accurate, larger differences are useful for determining the relative merits of designs. Alternatively, the XSG provides a native method for area estimation.

3.5 Testing Designs at Run-time

A number of options are available in run-time design testing to trade-off measurement convenience, test repeatability, the maturity of the components in the test setup, and other similar issues. The source of the inputs to the design is the primary distinguishing factor. If the setup is very mature and, for example, the radio front-ends are available, the tests can be performed with real-world I/O. If repeatable test vectors are needed, they can be fed from mass storage devices, or a test bench can be compiled for the emulation along with the design under test.

As depicted in Fig. 3, the capacity of the emulator allows multiple conventionally sized chips to be run in the same emulation, for example, the interoperability of a transmitter, a receiver, and a channel model can be tested. This is very useful for initial debugging, although it is not a very realistic operation environment, since a common clock that can be used to reduce timing-related problems can be routed to all parts of the design. Furthermore, modelling the channel allows the exploration of conditions that might be physically difficult to set up. Similarly, performance and functionality verification can be implemented on the emulator.

Viewing the signal behaviour inside the FPGAs is possible with Xilinx ChipScope [9]. ChipScope embeds additional logic into the designs, which mimics the functionality of a logic analyzer. In addition, it is possible to include extra logic to the design to monitor interesting behaviour or to store intermediate or final results. These methods facilitate data collection over long periods of time, for example to validate signal to noise ratios or bit error rates.

4. CONCLUSIONS

For many digital streaming applications, particularly those that interface with radio front-ends, implementations that are power and area efficient with real-time constraints are needed. Meeting all these criteria with synthesis from purely functional descriptions may not be possible and the algorithm and architecture trade-offs are closely intertwined. Direct-mapping with parameterizable components provides a way to describe both computation and architecture simultaneously. As an additional benefit, synthesis from these descriptions is fast and provides predictable results.

For rapid prototyping, enhancing the designer productivity is of primary interest. This paper concentrates on four issues from the different stages of prototyping and details their implementation on the BEE environment. Estimation is crucial in early design stages to tie the algorithmic decisions to hardware trade-offs. Design re-use is advocated by using an extensible set of direct-mapped components. Real-time prototyping is facilitated by the fixed wire arrangement in the BEE emulator and the corresponding partitioning and routing methods. Finally, the validation of designs is supported by the rapid prototyping itself and the debugging methodology.

5. ACKNOWLEDGEMENTS

This work was funded by Academy of Finland, DARPA and MARCO under C2S2, the U.S. Army Research Office, and the Berkeley Wireless Research Center supporting companies. In addition, we would like to acknowledge donations from Xilinx and The Mathworks Inc.

REFERENCES

[1] W. R. Davis, et al., “A Design Environment for High-Throughput, Low-Power Dedicated Signal Processing Systems,” IEEE J. Solid-State Circuits, vol. 37, pp. 420-431, Mar. 2002.

[2] G. J. Foschini, “Layered space-time architecture for wireless communications in fading environment using multiple-elements antennas,” Bell Labs Technical Journal, vol. 1, pp. 41-59, Autumn 1996.

[3] G. H. Golub and C. F. V. Loan, Matrix Computation, 2nd edition, John Hopkins University Press, MD, 1993.

[4] K. Kuusilinna, et al., “Designing BEE: a Hardware Emulation Engine for Signal Processing in Low-Power Wireless Applications,” EURASIP J. on Applied Signal Processing, vol. 2003, pp. 502-513, May 2003.

[5] K. Kuusilinna, et al., Real-Time System-on-a-Chip Emulation – Emulation Driven System Design with Direct Mapped Virtual Components, In Winning the SoC Revolution - Experiences in Real Design, Ed: G. Martin and H. Chang, Kluwer, 2003.

[6] A. S. Y. Poon, D. N. C. Tse, and R. W. Brodersen, “An adaptive multiple-antenna transceiver for slowly flat-fading channels,” IEEE Trans. on Communications, vol. 51, pp. 1820–1827, Nov. 2003.

[7] J. G. Proakis, et al., Algorithms for Statistical Signal Processing, Prentice Hall, 2001.

[8] www.mathworks.com.

[9] www.xilinx.com.

[10] N. Zhang, Algorithm/Architecture Co-Design for Wireless Communications Systems, Ph. D. Thesis, UC Berkeley, 2001.

[11] L. Zheng and D. N. C. Tse, “Diversity and Multiplexing: A fundamental tradeoff in multiple antenna channels,” IEEE Trans. on Information Theory, vol. 49, pp. 1073-1096, May 2003.