Efficiency Enablers of Lightweight SDR for MIMO Baseband Processing

Abstract:

The flexibility and programmability of an application-specific instruction-set processor (ASIP) come at the expense of reduced area and energy efficiency compared to application-specific integrated circuit (ASIC) solutions. Nevertheless, ASIPs are desirable for versatile application domains like wireless communications and software defined radio (SDR). Typically, ASIP designers reduce the ASIC-ASIP efficiency gap by increasingly complex architectures with decreasing flexibility and usability. This paper takes the opposite approach and presents concepts for a highly efficient, lightweight SDR ASIP. Efficiency enablers include simple but effective measures like a carefully chosen instruction set, optimized data access techniques for efficient utilization of functional units, and the use of flexible floating-point arithmetic with runtime-adaptive numerical precision. We present a conceptual processor core to show the impact of these measures and discuss its potential as well as limitations compared to tailored ASIC solutions. For demonstration, we choose the field of linear multiple-input multiple-output (MIMO) detection. We present synthesis results for several design versions in 90 nm CMOS technology and the corresponding energy benchmarks. Also, we show post-layout results for a selected design to demonstrate the feasibility of our concept. The proposed architecture of this paper analysis the logic size, area and power consumption using Xilinx 14.2.

Enhancement of the project:

Existing System:

An efficient ASIP needs a suitable instruction set, versatile enough to support a multitude of use cases but also application-specific enough to boost the processor’s efficiency into the range of comparable ASICs. The vectorial nature of multiple-input multiple-output (MIMO) baseband
processing motivates a single instruction multiple data (SIMD) instruction set with native support for complex-valued arithmetic. To support, for example, multiple antenna configurations, the instruction set has to handle a set of matrix and vector dimensions efficiently. This calls for tailored permutation units to map the desired functionality onto the existing data path. Also, to ensure high utilization of the available functional units, a specialized bypassing unit that can retrieve computational results from different points within the pipelined arithmetic logic unit (ALU) is needed. The limited dynamic range of fixed-point number formats requires additional effort for numerical stabilization (e.g., by scaling or matrix factorization), which can be avoided by the use of floating-point arithmetic. Despite the increased energy consumption per operation, the higher dynamic range enables the use of algorithms with reduced runtime, which puts this drawback into perspective.

MIMO baseband processing algorithms show diverse requirements in numerical precision depending on the use case (e.g., antenna setup). Moreover, some of these algorithms can be decomposed into distinct sections with different precision requirements. This inspired the concept of numerically aware processing (NAP), which adapts the numerical precision of the data path at runtime on a bit-granular level to reduce switching activity and hence energy consumption. The idea of NAP is related to the concept of approximate computing (AC) which assumes that a small degradation of processing accuracy is tolerable e.g., due to perceptual limitations of humans with regard to multimedia content. Our research shows that the same concept applies to MIMO baseband processing.

Disadvantages:

- High area coverage
- High power consumption

Proposed System:

The napCore is a fully programmable SIMD processor core designed for vector arithmetic.

Pipeline Overview
Fig. 1 shows the pipeline structure of the SIMD core. An instruction word is requested from the program memory (PMEM) in the pre-fetch stage (PFE) and received one cycle later in the fetch stage (FE). It is then interpreted in the decode stage (DC), which configures all further stages. Operands are loaded and preprocessed by the PrepOp-DC unit, which also performs operand bypassing to resolve data hazards. The following four arithmetic stages (EX1, EX2, RED1, RED2) are designed to match the processing scheme of standard vector arithmetic operations, which is a composition of multiplications and subsequent additions.

Operand Acquisition

For programmable architectures with inherent parallelism like SIMD or very long instruction word (VLIW) processors, the potential for data-level parallelism is defined by the parallelism of the data path, given there is an efficient operand acquisition mechanism. Even for regular vector arithmetic operations, this is a challenging task. Consider the previously described SIMD architecture with a scalar and a vector register file. Depending on the instruction, very different data access patterns have to be realized, which leads to the complex operand acquisition architecture depicted in Fig. 2 for the first operand. Widths of the data path are given as multiples of complex-valued scalars.
Permutation Network

Fig. 3 shows the schematics of the two permutation networks in front of the multipliers in EX1. Apart from straight pass-through, the networks support patterns especially for $2 \times 2$ vector arithmetic operations like matrix inversion, determinant calculation, or matrix-matrix multiplication. Since the first vector typically holds the left-hand value of a multiplication and $2 \times 2$ matrices are stored row-wise in the vector registers, the left and right pair of multiplexers are wired to select one of the two matrix rows via hilo1 and hilo2.
In our processor core, we place a masking unit as in Fig. 4 at the end of operand loading in PrepOp-DC as well as after every arithmetic component within the 4-stage ALU. The bitmask can be adapted at runtime by a configuration instruction in the program code.

Configurable Reduction Stages

To support a versatile instruction set, e.g., for efficient processing of vectorial data of different dimensions, the reduction stages RED1 and RED2 are designed to fit the requirements of a wide range of vector arithmetic operations. The maximum number of required complex adders in RED1 corresponds to SIMD parallelism degree P, which is needed, if a multiply-accumulate operation with P-dimensional vector operands is executed. Note that for an inner product, an adder tree of depth ld(P) is sufficient, which requires P/2 adders in RED1 (if P is a power of 2). In RED2, P/4 adders are sufficient for an inner product, but for our use case of P = 4, we chose to place an additional adder, which is used for some specialized instructions for √P × √P vector arithmetic (the dimension for which one square matrix fits into one vector register). Fig. 5 shows parts of the reduction stages RED1 and RED2 for P = 4.
Fig. 5. Reduction stages RED1 and RED2.

**Advantages:**

- Reduce the area and power

**Software implementation:**

- Modelsim
- Xilinx ISE