Adaptive Beamforming using QR in FPGA

Richard Walke

Real-Time Systems Lab, QinetiQ Ltd
Malvern Technology Centre, WR14 3PS, UK
rlwalke@QinetiQ.com

Abstract

Adaptive beamforming plays an important role in sensor array systems in countering interference outside of the direction of interest. However, calculation of the adaptive weights generally requires a large number of operations that rapidly grows with the number of antennas. Consequently, a large number of programmable processors is commonly required to calculate the weights, which in some systems may present excessive weight, volume and power requirements.

Field programmable gate arrays (FPGAs) offer a credible alternative to re-programmable technology for implementing digital signal processing. Current devices provide the equivalent of 6 million programmable gates, and dedicated fixed-point multipliers. These support performances in the range of 50 GOPS for the fixed-point filtering and beamforming functions required by radar and communications systems. Furthermore, devices are emerging that contain embedded processors and high-speed serial interfaces. This functionality allows interfaces to be constructed that simplify integration of FPGAs with RISC processors for handling of back-end processing and control.

We present an implementation of an adaptive beamformer in FPGA technology. In particular, we describe a software programmable digital receiver, and the implementation in an FPGA of a scalable processor for adaptive weight calculation using QR decomposition. This operation solves the least-means squares problem at the heart of adaptive beamforming, and is a useful building block from which to construct a range of real-time implementations of algorithms offering Space-Time-Adaptive-Processing (STAP), wideband, and beamforming on moving platforms.

QR Processor in an FPGA

The QR processor employs a novel mapping of the QR algorithm to a parallel array of processors that allows processor implementations to be optimised to specific functions. Furthermore, the final architecture is a linear array, with local interconnect, which can be scaled up or down to increase the amount of processing for a particular problem size. The processor also employs floating-point arithmetic. This is uncommon in FPGAs, and we justify its use by adopting an algorithm that reduces the number of operations by exploiting the high dynamic range of floating-point. The wordlength of the mantissa is also minimised to a point sufficient only to meet the requirements of the application. In this way, we obtain fast and efficient implementations with good numerical performance.

The library of floating-point operators is parameterised, and has been implemented as parameterised relationally placed macros for Xilinx FPGAs. This ensures predictable timing and dense layout. We use standard VHDL and synthesis tools to obtain these components for particular mantissa wordlengths.

Using this library, a QR processor with sustained computation rate of **20GFLOPS** is obtained on a Xilinx XC2V6000-4 with 14-bit mantissa wordlength. Power consumption of the order of 15W is obtained, which represents almost an order of magnitude reduction over a PowerPC^{TM} implementation with equivalent performance.

Adaptive beamformer in hardware

The QR processor has been used to implement an adaptive beamformer, which includes a software programmable digital receiver, beamformer and back-substitution processor. The latter completes the weight calculation process. The QR processor and digital receiver have been implemented on an FPGA and the remaining functions on a PowerPC.

A channel-based communication mechanism has also been employed to provide communication between the PowerPC and FPGA. An application-programming interface (API) has been implemented to allow a range of transfers including streaming data, commands and remote memory access. This allows transfers over a packet-switched fabric or PCI bus. API functions on the PowerPC are complemented by a set of cores on the FPGA, that provide break-out of the physical communication fabric into a number of channels. This enables parts of the system on PowerPC to communicate with multiple cores on FPGA in a similar fashion to communications between two conventional processors.

Heterogeneous system design

The creation of a range of parameterised cores and a communications API provides us with the infrastructure to rapidly create a heterogeneous implementation combining both FPGAs and PowerPCs. However, for greater productivity an environment is required to model, partition and automatically generate the implementation from a library of cores. GEDAE^{TM} is a well-established graphical modelling and auto-code generation environment that can target parallel arrays of conventional processors. It supports a data-flow model of computation that is well matched to sensor array signal processing problems, and maps well onto FPGAs. As such, it presents a good starting-point for a heterogeneous design environment.

In this presentation we show how GEDAE has been used to develop the heterogeneous adaptive beamformer. It is also used in the hardware demonstration to generate stimulus and graphically present the results. In effect, the complete system is truly heterogeneous running on the Pentium host, PowerPC and FPGA.

Acknowledgements

PowerPC is a trademark of IBM Corporation..
GEDAE is a trademark of Blue Horizon Development Software.

This work has been sponsored by the UK Ministry of Defence Corporate Research Programme and has been undertaken in collaboration with BAE SYSTEMS ATC, Gt. Baddow, UK. Contributions have been made by John McAllister of the Queen's Uinversity of Belfast.