Implementation of Digital Front End Processing Algorithms with Portability across Multiple Processing Platforms

John Holland, Northrop Grumman Corporation, John.Holland@ngc.com
Jeremy W. Horner, Northrop Grumman Corporation, Jeremy.Horner@ngc.com
Randy Kuning, Northrop Grumman Corporation, Randy.Kuning@ngc.com
David B. Oeffinger, Northrop Grumman Corporation, David.Oeffinger@ngc.com

Abstract

Digital front end processing algorithms are commonly used in many systems such as radar, electronic warfare, or communication systems. Algorithms can include filtering, tuning, channelizing, digital beamforming and are used across different user systems – space, airborne, shipboard, and land-based. Because of the wide variety of uses, real-time digital front end processing algorithms are embedded on many different hardware platforms. This paper describes a process for implementing these algorithms on different hardware platforms – ASIC, FPGA, or software on both custom and COTS hardware – as a means of providing computing technologies for different or challenging form factors.

Hardware Architecture Development

Figure 1 shows an architecture development process that implements a generic algorithm using a given technology.

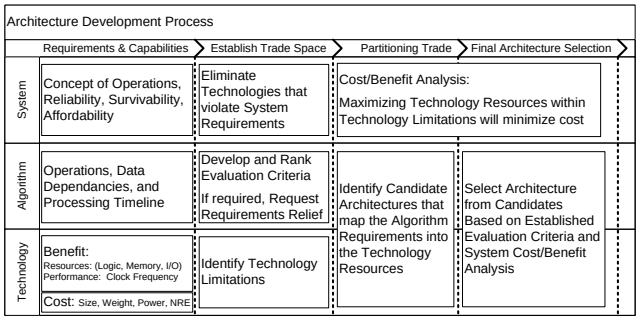

Figure 1: Architecture development process

Requirements & Capabilities – Phase 1 determines the system and algorithm *requirements* and evaluates the *capabilities* of candidate hardware. The process begins with an algorithm's generic architecture and tailors it to meet the system requirements, concept of operations, environmental conditions, and cost constraints. The tailored algorithm is then analyzed to determine its data dependencies and its performance, memory, and I/O requirements. Finally, a cost/benefit analysis is performed in order to understand the relative strengths and weaknesses of each technology.

Establish Trade Space – In Phase 2, critical system requirements (e.g. survivability) immediately eliminate some technologies from the *trade space*. Those remaining are evaluated according to algorithm-specific criteria to assess limitations on the implementation of the algorithm. For example, a given target technology may cause regions of the architecture to become I/O-bound, memory-bound, operation-bound, power-constrained, or size-constrained.

Partitioning Trade – In Phase 3, candidate architectures *partition* the tailored algorithm into the technologies taking into account the strengths and weaknesses of the technology and the algorithm's requirements.

Final Architectural Selection – In Phase 4, the final phase of the process, candidate architectures are analyzed using the established, system-specific evaluation criteria to select the *final architecture* and *implementation technology*.

This process can be used with any algorithm. This paper will examine its use in implementing a digital beamforming algorithm using various target technologies.

Digital Beamforming

Digital beamforming (Figure 2) forms one or more beams from sub-apertures of an antenna. In the figure the incoming wavefront is received by each of the channels of the antenna, down-converted, and A/D sampled. Digital beamfoming uses time delays and/or frequency-specific phase shifts to steer the input channels so that each senses the same frequency and phase of the wavefront.

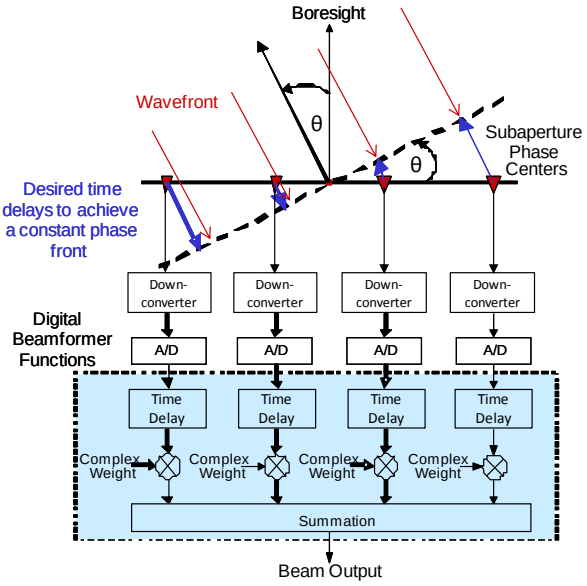

Figure 2: Notional digital beamforming architecture.

Key parameters for digital beamforming architectures are: input channels and sample rates, output beams and data bandwidth, as well as cost, size, weight, power, and environmental requirements.

Hardware Implementation Options

Front end processing algorithms can be written in hardware languages (VHDL, Verilog, etc.) or software languages (such as C) and implemented in a variety of hardware such as:

ASIC – Application Specific Integrated Circuits, custom devices designed and fabricated for a specific application

FPGA – Field Programmable Gate Arrays, reprogrammable commercial-off-the-shelf (COTS) devices

Multi-core processors – processors with multiple software-programmable processing cores and an interconnect fabric with interfaces to memory and other devices

GPU – Graphic Processing Units

Custom boards – fully custom printed circuit board (PCB) designs

COTS FPGA boards – PCBs with standard FPGA devices and standard external interfaces

This paper describes a number of digital beamforming systems. Each system uses a hardware platform comprised of the above components and offers a unique set of capabilities and limitations (shown in Table 1).

	Custom: ASIC, Board	Custom: FPGA, Board	COTS: FPGA Board	COTS: Processor Board
I/O Flexibility	Highest	High	Medium	Low
Capability	Highest	Medium	Medium	Low
Processing Efficiency	Highest	Medium	Medium	Lowest
Programmability	None	Medium	Medium	High
Complexity	High	Medium	Medium	Low
NRE Cost	Highest	Medium	Low	Lowest
Power	Lowest	High	High	Medium

Table 1: Hardware Platform Capabilities and Limitations

The following examples show how the development process enables portability in the implementation of digital front end processing algorithms across a diverse set of hardware platforms and system requirements.

Example – FPGA and Custom Board

This platform was selected to achieve a balance between capability, flexibility, and cost.

- *System Requirements:* Process 18 channels of sub-banded data to form 4 output beams.
- *Algorithm Requirements:* 138 Giga-Operations/sec
- *Establish the Trade Space:* System environmental and cost requirements restrict the trade space to radiation-tolerant FPGAs.
- *Algorithm Data-Dependencies:* None exists between subbands, but a significant data-dependency between output beams.
- *Candidate Architectures:* The optimal architecture minimizes the number of devices used to process the output beams for a given subband and increases the number of subbands processed in a given device until one of the resources of that is consumed.

The architecture in Figure 3 shows the result of the partitioning trade. It minimizes the number of FPGAs required by maximizing the resource utilization of each FPGA within the I/O constraints of the FPGA technology. All output beam processing for a given subband is performed in the same FPGA – giving priority to the data dependency between beams.

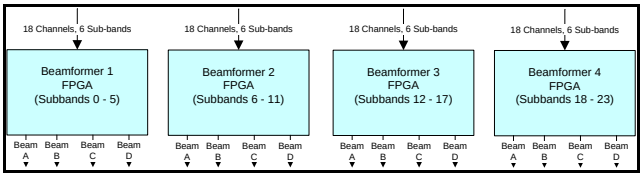

Figure 3: Block diagram of FPGA and custom board.

Full Custom – ASICs and Custom Board

This form factor was selected to minimize power. Figure 4 (on the left) shows a full custom implementation of a digital beamformer for space applications. This cascadable beamformer board receives four input channels with sample rates of 480 MHz and produces 4 beams in each of the 24 subbands.

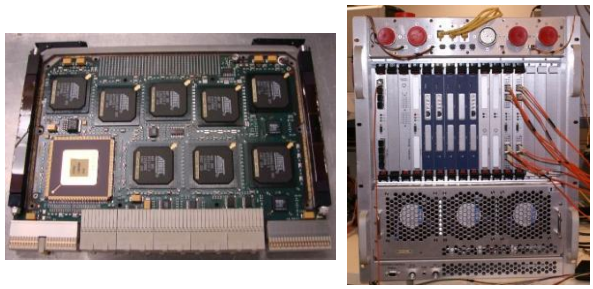

Figure 4: Full custom digital beamformer with ASICs and COTS modular processor beamformer.

COTS Modular Processor Architecture

Figure 4 (on the right) shows an example of digital beamforming implemented in an open system architecture using highly modular programmable COTS hardware. The processor receives 80 A/D input channels and forms 30 independent beams. This scalable architecture consists of five modules, each containing four FPGAs. High-speed fiber optic links between modules create a well-connected regular, modular, scalable architecture.

COTS & Advanced Processor Architectures

The same techniques for front-end processing algorithm implementations can be applied to COTS processor elements and architectures – legacy CPUs, multi-core processors, GPUs, or advanced processor architectures. The algorithm can be divided between these processing elements and other hardware such as FPGAs or ASICs. The algorithms are portable – allowing an open trade space at the beginning of the algorithm development process.

Conclusion

Front end processing algorithm must be portable across a variety of hardware platforms. The platforms include full custom hardware that maximizes performance/power ratios, reprogrammable devices that provide real-time hardware execution of front end processing algorithms, or software solutions on advanced processors. A design process that adapts algorithms to different requirements allows portability of digital processing algorithms across systems.