Design of Path Optimization Algorithm Using COTS Field Programmable Gate Array Hardware and Software Platforms

Neil Harold, John Ostapovich
Nallatech
E-mail: n.harold@nallatech.com
E-mail: j.ostapovich@nallatech.com

Rick Pancoast
Lockheed Martin MS2
E-mail: rick.pancoast@lmco.com

Jon Russo
Lockheed Martin ATL
E-mail: jrusso@atl.lmco.com

Introduction

It seems obvious to state that High Performance Embedded Computing (HPEC) technology has changed radically in the last 10 years, with significant upgrades in capability offering superior performance. One of the enablers of this is the commoditization of computer components helping to reduce the overall cost of HPEC technologies.

On closer analysis, however, one could argue that in fact very little has changed, in the sense that the challenges faced 10 years ago are the very same as those confronting developers today. Even more significantly, the techniques used to overcome these obstacles really haven't changed either, in particular, the processing technologies that underpin embedded computing applications. As they start to reach performance limits imposed by the laws of physics, developers are faced with critical decisions over which direction to take to deliver future HPEC systems.

Revolution in Programmable Logic

Programmable logic has undergone a dramatic change over the last 10 years, with the advent of advanced multi-million gate devices, leading to the introduction of hugely powerful families of Field Programmable Gate Arrays (FPGAs). The increasing capability of these devices has seen them evolve from playing a supplementary role, often as "glue" logic in a system, to performing vital processing tasks that had previously been the domain of ASICs.

While the widespread use of FPGAs for embedded signal processing has taken longer to gather momentum, the insatiable appetite of military HPEC applications has meant that the DoD has shown more interest than most in using FPGAs for this purpose. The use of FPGAs has tended to be limited, however, to the role of a "hardware accelerator" supporting a more traditional processing architecture.

Vendors, developers and users are beginning to realize the potential of FPGAs for a greater share of embedded computing tasks, but even in this high-interest climate, questions remain over just how much extra performance can be gained when using FPGAs and, perhaps more importantly, at what cost?

FPGA Technology

The fixed architecture of general purpose processors (GPPs) and DSPs has constrained their main improvements in capability over in the last 10 years to those gains attained through increasing clock frequencies. The flexible architecture of FPGAs, on the other hand, have been able to take full advantage of increasing transistor count to offer greater volumes of programmable logic combined with fixed function pieces of silicon such as embedded processors, multi-gigabit transceivers and embedded digital signal processing blocks.

The technical advances achieved in the last few years in particular have made FPGAs more powerful at the device level, easier to integrate at the board level and more applicable to signal processing applications at the system level. Unfortunately, these advances have not necessarily made developing applications for FPGAs any easier. Despite significant industry investment in design tools and compilers, the use of FPGAs remains mainly limited to user base with long-standing expertise in developing solutions using Hardware Description Languages (HDLs) such as VHDL and Verilog.

High-Level Languages for FPGAs

The last 5 years has seen unprecedented progress in the area of high-level tools for FPGAs, with varying degrees of success. There are a number of tools available offering a variety of approaches to solving this problem, mainly focused on offering translating high-level languages such as C/C++ and Matlab to a "middleware" level such as VHDL that can be mapped to the FPGA.

Many parameters influence the effectiveness of these tools, in particular the precise syntax of the high-level language used (they rarely operate on fully standard languages) and their ability to implement parallelism. Most of the available tools attack parallelism at the micro-level – they break down specific functions in the code that can be parallelized. Some tools look at the code at a macro-level, but this generally requires further variation from the standard language, i.e. more effort on the part of the programmer.

Nallatech's DIME-C tool operates on a micro-level, translating discrete functions written in ANSI-C code into VHDL that can then be put through the standard synthesis and implementation process to generate a bitstream for the FPGA. It expresses the C-code as a data-flow and analyses that dataflow to determine which variables are related to each other in order to understand where inter-dependencies lie. This information is then used to implement parallel pipelines where appropriate.

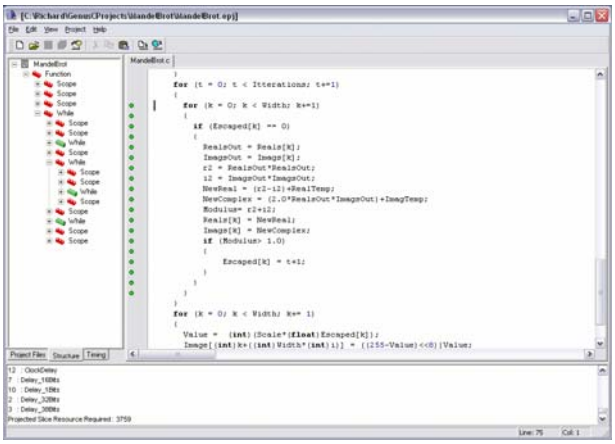

Figure 1: Screenshot from DIME-C

Users of DIME-C require a degree of knowledge about the FPGA architecture, thus, the main benefit of the tool is in improving productivity of existing design teams, rather than introducing vast numbers of new users to FPGAs.

Reducing FPGA Development Times

Many experienced FPGA users would attest to the fact that building reliable communications structures in multi-FPGA systems is actually a more time-consuming task than creation of the algorithms themselves, with as much as 80% of development time being spent on this element of the design. Despite this potential for productivity improvement, development of communication structures within devices and systems is an area of FPGA tool development that has gone largely ignored by the community. The goal of Nallatech's DIMEtalk is to address this issue by making the development and targeting of applications to COTS FPGA hardware more efficient, straightforward and reliable.

Application and Productivity Performance

Graph-based path optimization is applicable to several domains, including craft mission planning and VLSI design. We consider the problem of shortest path finding in a partially connected graph whose nodes represent quantized positions in N-dimensional space, and whose edges represent costs of traveling between node pairs. The DIME-C implementation of the shortest path solver was based on a dynamic programming approach which may also be applied to maximum likelihood algorithms such as Viterbi decoding. Figure 2 shows the simple interface to the dynamic programming engine done using DIME-C.

Persistent cost data is loaded and distributed into the route engine using an initialization call. Subsequent transfers may incrementally modify persistent data, or may query the engine for a route table.

Lockheed Martin has utilized standard DIMEtalk APIs, integrated within a discrete event simulator known as CSIM, providing a hardware-in-the-loop capability to assist in reconfiguring and controlling the FPGA and to provide federated access to the dynamic programming engine across remotely located participants.

The five-million "equivalent gate" design is comprised of 27 1K X 32-bit dual port memories, and coordinates over 100 operators in a highly irregular parallel pipeline. The first pass design outperforms modern conventional processors by 10-to-1 in actual time, and 100-to-1 in cycle count, running at 100 MHz on a Xilinx Virtex-II 2V4000.

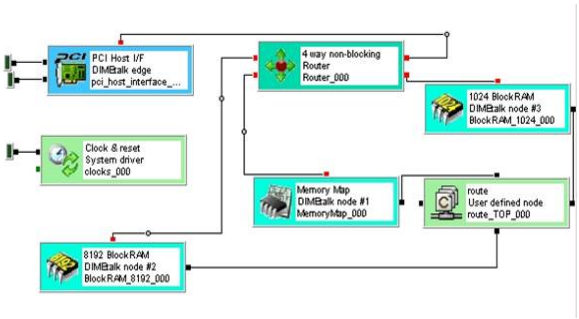

Figure 2: DIMEtalk Design

Related designs with higher levels of unrolling have achieved close to 100-fold improvements in performance, without any custom optimization.

The full design took approximately one person-month to conceive, implement, and test, compared with an estimated time of 1 year perform the same tasks using VHDL.

The Future

It appears that the laws of physics are finally catching up with Moore's Law, which will affect the future performance gains of GPP–based HPEC systems. This will lead to greater interest than ever before in doing "more with less" – that is, considering new technologies and architectures that can deliver increased performance while adhering to the power and thermal restrictions of the targeted platform and environment. At this juncture, it is clear that new technologies and alternative processing approaches will be vital to the technological progression and increased effectiveness of DoD systems.

FPGA technology currently offers a great deal in this regard as a new paradigm in processing architectures. The most significant obstacle to their mainstream acceptance is the creation of a mature design-flow that enables non-experts to target FPGA platforms. The work presented here does not resolve that issue, but it makes a significant contribution to those developers already using FPGAs and begins to bridge the gap between the traditional FPGA design flow and the utopian goal of "C-to-gates".

Wider-spread adoption of FPGAs could have a massive impact on the future capabilities of DoD architectures. In addition to offering superior performance, the lower power characteristics of these devices have the promise to deliver on the desire of the DoD to move more processing into the sensor payload. Some of the advantages include a reduction in the burden on communications networks, thus freeing up vital bandwidth that will assist initiatives such as the Global-Information-Grid and truly enable a fully network-centric battlefield.