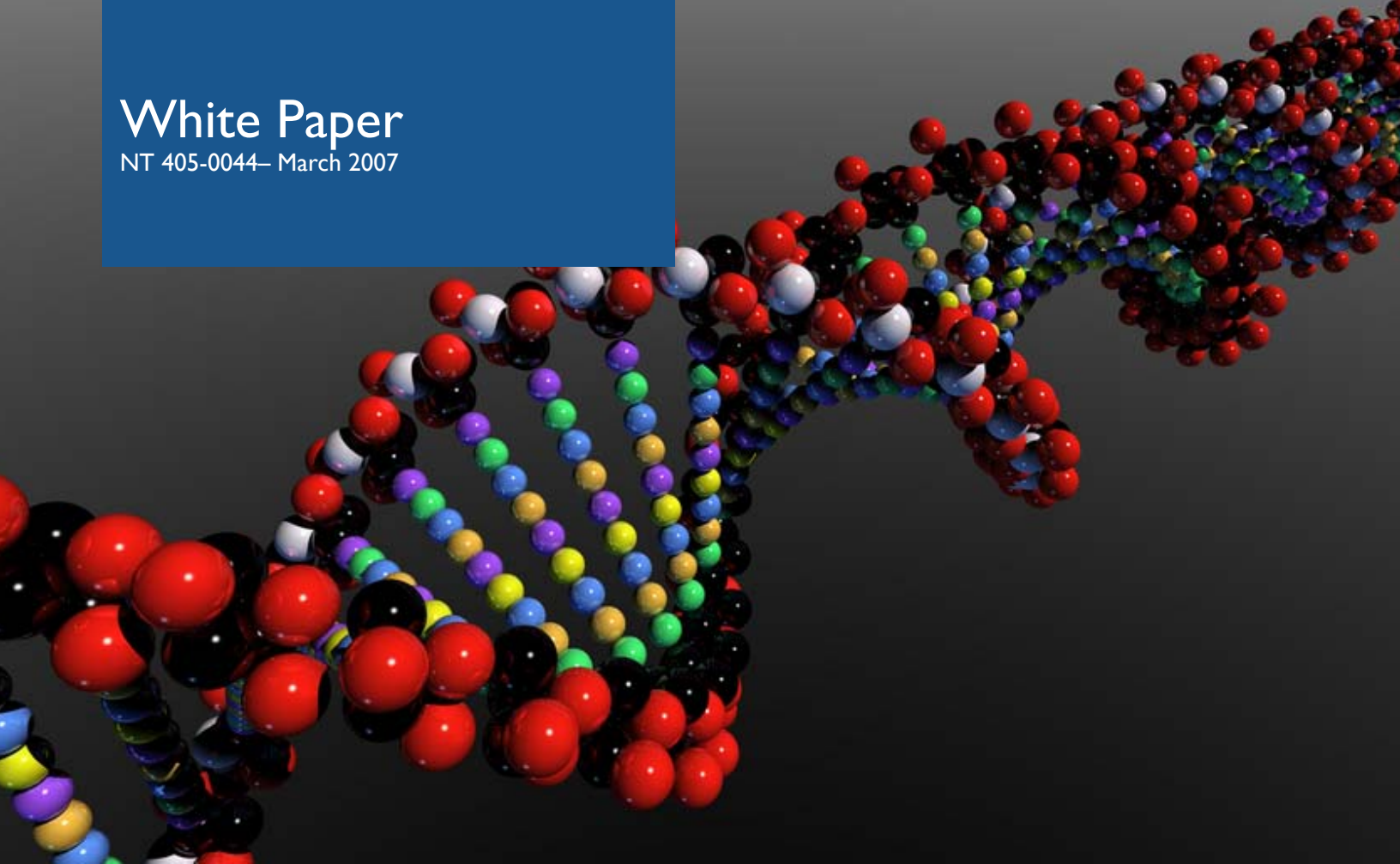


White Paper

NT 405-0044– March 2007



Developing Applications for Nallatech HPC Platforms



The inherently parallel nature of the FPGA offers application developers the potential for performance improvements that far outstrip those possible with conventional microprocessors. This white paper looks to explain how application developers with no experience in FPGA computing can develop algorithms using Nallatech's HPC line of products and the development environment provided. It will detail the features of Nallatech's High-Level Compiler, DIME-C, before giving an overview of the development process of Nallatech HPC products.

Robin Bruce
EngD Research Engineer

High-Level Compilers

There are now a number of high-level compiler tools available to develop HPC algorithms on FPGAs. These tools allow designers to focus on the structure of their algorithm, and abstract away the production of HDL code that describes circuit behavior. High level compiler tools allow users to express their algorithms in a manner more closely resembling a software development. Three different high-level compilers can be used with the Nallatech HPC line of products, DIME-C, Impulse C and Mitrion-C. DIME-C is Nallatech's own High-Level FPGA compiler and it compiles a syntax that closely follows the ANSI C standard. Details on the Impulse C and Mitrion-C compilers can be obtained from the websites www.impulsec.com and www.mitrionics.com respectively.

All high-level FPGA compilers require users to structure their algorithm in a manner that will result in optimal hardware structures. Designers must prepare the algorithm for a two-level compilation process. Firstly, the high-level compiler will automatically create a behavioral description of the algorithm, specifying in exact detail how logical operations will occur in relation to clock cycles. The description will exist in some form of HDL. This behavioral description will then be passed to a second compilation process where abstract logical operations are assigned to FPGA logic resources, routing and pin mappings. This two-step compilation process is usually fully automatic from the perspective of the application developer. In order to obtain the best performance, designers must pipeline and parallelize as much of their algorithm as possible. The limiting constraints are the nature of the algorithm, the resources available on the FPGA and the capabilities of tool and designer combined. FPGA Pipelines have no theoretical depth limit and a single design may have multiple parallel pipelines, with depths of hundreds or even thousands of cycles. This, combined with other parallelization techniques such as logic replication and concurrent scheduling of independent operations, gives FPGAs their vast performance potential. The approach taken to generate parallel and pipelined structures is a distinguishing factor between FPGA HLLs. Some languages require users to explicitly tag sections of code to pipeline or schedule in parallel. Others have the philosophy that the tool should attempt to parallelize and pipeline code wherever it sees the opportunity. Some compilers have certain pragmas or compiler directives that allow users to specify large-scale logic duplication without adding any non-ANSI-compliant grammar to their syntax.

DIME-C

Nallatech has developed the DIME-C tool both as a tool for application developers that use Nallatech reconfigurable computing platforms, as well as for use by Nallatech researchers and engineers working on industrial and academic projects. Research and development of DIME-C is continuous. The feature set expands with time, and the conversion process that turns algorithms into HDL is improved to obtain the best performance possible for the hardware targeted.

DIME-C uses a syntax that is a subset of ANSI C. There are a number of justifications for the choice of the DIME-C syntax.

Lack of Pointer Support

Only the elements of ANSI C grammar that are most easily representable in hardware have been kept. Pointers are not supported in the DIME-C syntax. Pointers can be used for storing addresses, addresses of addresses of data, etc. In the hardware everything is implemented and will have a physical instantiation. Without pointers code will directly show parallel use of resources within the code. If pointers were used it could be more difficult for the compiler to track the parallel use of hardware resources.

Omissions of Redundant Grammar

Most of the other omissions in the grammar are due to the origins of DIME-C. As the tool developed, ANSI C grammar was added to increase the functionality of the tool. As a consequence, certain types of switch statements or loops do not exist, if the same functionality can be obtained by another means. Recently added to DIME-C have been structures. These can be very useful for certain applications, though it is not possible to create arrays of structures.

No Additional Keywords

Within DIME-C it is possible to handle FIFO channels or exchanges with FPGA memories by using the ANSI-C function calls. The grammar therefore remains standard ANSI C. The development philosophy in DIME-C is to increase functionality without breaking out of the confines set by the ANSI C grammar. DIME-C may one day encompass the full ANSI C grammar, but it should never go beyond it.

Developing an Application for High Performance

Developing an application using a high-level compiler such as DIME-C involves carrying out a number of steps, first to obtain a functional version of the application, and then to adapt the application to obtain the highest possible performance. The following steps apply first and foremost to DIME-C, but can be generalized to apply to most high-level compilers.

Step 1: Conversion from Source Code to ANSI C

If the application originates as software routines in a language other than ANSI C, then it must first be converted. Some grammar conversions are therefore necessary. This is quite an easy step, though it naturally depends on the nature of the source language. Some automated translation tools exist for converting languages, e.g. for converting FORTRAN code into C.

Step 2: Creating a Software Simulation

The subset of ANSI C used in DIME-C can be executed on a microprocessor at all stages of the application development process. This has the advantage of allowing users to simulate their algorithm “at-speed” rather than through a slow hardware simulation. As the algorithm is changed to gather the highest performance, it is important to verify in a software simulation that the functionality does not change.

Step 3: Adapting the Code for Performance

To speed up a code it is usually necessary to pipeline it. To pipeline code, one must attempt to express the algorithm in for loops that do not show recursion. Other modifications are often necessary. For example, it is important to reshape the algorithm to have memory access patterns that best suit the target architecture. It is also possible to leverage large scale spatial parallelism with the simple use of pragmas.

Step 4: Synthesizing and Executing the Algorithm in Hardware

This is the final step before a functional hardware version verification of the design. It should be stressed that perhaps the most important design aspect of implementing an algorithm on FPGAs is verifying that functionality is not changing as the code is optimized for performance

Nallatech HPC Development Flow

Figure 1 shows the development flow used in conjunction with Nallatech HPC products such as the H100 series of products. The key to successful implementation of HPC applications on clusters with FPGA compute acceleration is ease of application development. Support for three C-to-FPGA compiler tools is offered with the H100 products, enabling users to select the tool best suited to their application. Industry-leading compilers from Mitrionics and Impulse are complemented by Nallatech’s own DIME-C compiler. The development flow enables software developers to effectively harness the benefits of FPGA compute technology with little or no understanding of the underlying technical complexities. The Nallatech DIME-C compiler aids the software engineer in the process of creating a parallel and pipelined implementation of their application. This is achieved by providing developers with graphical feedback of the code elements that have been parallelized and pipelined. Additional reporting points allow the developer to identify possible coding techniques that would improve the implementation. The other supported compilers provide similar capabilities. Having implemented the design using the compiler tools, the developer then simply compiles the application code using Nallatech’s Application Builder, generating a runtime executable that can be ran on the targeted platform. This works without user intervention, however expert users can achieve even higher performance levels by accessing the advanced features of the application integration tool. The code implementation is designed to work in conjunction with a host processor. Nallatech provides a standard Software API for Windows and Linux that allows developers to easily integrate the accelerated code on the FPGA with the rest of the computing environment.

White Paper

NT 405-0044– March 2007

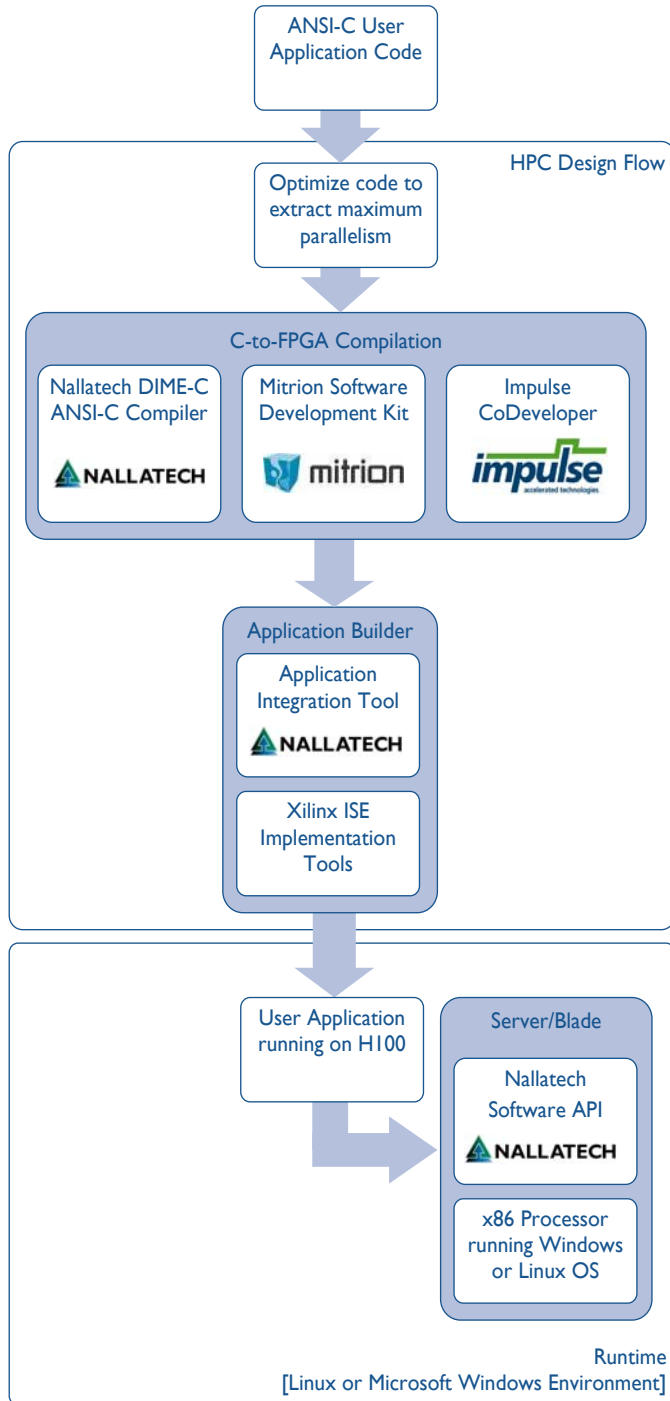


Figure 1 – Nallatech HPC Development Flow

Conclusions

It has been shown how the Nallatech HPC platforms and their development flow can be used to simplify the process of developing an application for FPGAs. Users of Nallatech HPC platforms have the choice of three high-level compilers: DIME-C, Impulse C and Mitrion-C. These allow users to develop applications whilst being shielded from the complexities of electronic design underlying these compilers.

To learn more, visit www.nallatech.com/hpc

Document: NT 405-0044 March 2007.

Intellectual Property: The contents of this document are the copyright of Nallatech Limited – © Nallatech Limited 2007. All rights reserved. The product name, Nallatech, the Nallatech logo, "The High Performance FPGA Solutions Company", DIMETalk, DIMEScript, DIME-II and FUSE are all trade marks of Nallatech Limited. All names, images and logos identifying Nallatech Limited or third parties and their products and services are subject to copyright, design rights and trade marks of Nallatech Limited and/or third parties. Nothing contained in these terms shall be construed as conferring by implication, estoppel or otherwise any license or right to use any trademark, patent, design right or copyright of Nallatech Limited, or any other third party. Microsoft and Windows are either registered trade marks or trade marks of Microsoft Corporation in the United States and/or other countries.

Disclaimer: This document is for general information purposes only and is not tailored for any specific situations or circumstances. Although Nallatech Limited believes the contents to be true and accurate as at the date of writing, Nallatech Limited makes no assurances or warranty regarding the accuracy, currency or applicability of any contents in relation to specific situations and particular circumstances. As such, the content should not be relied upon and readers should not act on this information without further consultation with Nallatech Limited. Nallatech Limited accepts no responsibility for loss which may arise as a result of relying on the information in this document alone.