On Coding Techniques for Targeting
FPGAs via OpenCL

Nuno PAULINO\textsuperscript{a,1}, Luís REIS\textsuperscript{a}, João M.P. CARDOSO\textsuperscript{a}

\textsuperscript{a}INESC TEC and Faculty of Engineering of the University of Porto, Porto, Portugal

Emails: \{nmcp, luis.cubal, jmpc\}@fe.up.pt

Abstract. Software developers have always found it difficult to adopt Field-
Programmable Gate Arrays (FPGAs) as computing platforms. Recent advances in
HLS tools aim to ease the mapping of computations to FPGAs by abstracting
the hardware design effort via a standard OpenCL interface and execution model. How-
ever, OpenCL is a low-level programming language and requires that developers
master the target architecture in order to achieve efficient results. Thus, efforts ad-
dressing the generation of OpenCL from high-level languages are of paramount
importance to increase design productivity and to help software developers.

Existing approaches bridge this by translating MATLAB/Octave code into
C, or similar languages, in order to improve performance by efficiently compiling for the
target hardware. One example is the MATISSE source-to-source compiler, which
translates MATLAB code into standard-compliant C and/or OpenCL code.

In this paper, we analyse the viability of combining both flows so that sections
of MATLAB code can be translated to specialized hardware with a small amount
of effort, and test a few code optimizations and their effect on performance. We
present preliminary results relative to execution times, and resource and power con-
sumption, for two OpenCL kernels generated by MATISSE, and manual optimiza-
tions of each kernel based on different coding techniques.

Keywords. FPGA, OpenCL, High-Level Synthesis, MATLAB, source-to-source
compilers

1. Introduction

In the scope of parallel programming and parallel-oriented architectures, the most com-
monly used devices are either GPUs or multi-core processors. The appeal of these de-
vices is the high-level abstraction from the hardware itself by mature flows and program-
ning models, e.g. OpenCL [Khr16], for GPUs, and OpenMP for multi-core CPUs.

In contrast, FPGAs allow for fine-grain configuration of their circuitry in order to
implement specialized hardware. Thus, beyond being prototyping devices, FPGAs can
be used as application-specific accelerators in embedded or high-performance comput-
sing systems. Unlike GPU architectures, which are oriented for SIMT (single-instruction,