Chapter 1

High Level Synthesis

A recurring theme in the literature is the relative difficulty of implementing an algorithm on an FPGA compared to conventional implementation techniques on CPU's and GPU's. Both development time and place-and-route take considerably more time compared to programming/compiling for more traditional architectures. [5]–[7]. High level synthesis tools enable a designer to implement an algorithm in a high level language and to have it compiled and synthesized into hardware. These tools enable faster prototyping and implementation.[8] The latest generation of these tools uses C or variants of this language to enable programmers without a background in HDL design to benefit from the advantages of FPGA accelerators without facing the steep learning curve of learning a HDL such as VHDL or Verilog. For existing HDL designers HLS tools these tools present a reduction in the number of lines of code that are needed to describe the design.[9] These tools enable to shift the focus from low-level implementation details to the development and improvement of the algorithm in a rapid prototyping fashion.[10] HLS tools have a long history dating back to the 1970's but only recently have these tools matured enough to become adopted by industry. These tools present an interesting evolution and a possible paradigm shift in hardware design and prototyping.[11] Two such tools will be used for this master thesis: the Riverside Optimising Compiler for Configurable Computing (ROCCC) and Xilinx Vivado.

1.0.1 Riverside Optimising Compiler for Configurable Computing

ROCCC is a C-to-vhdl compiler which focusses on FPGA based code acceleration. It implements a subset of the C language on which it performs loop analysis techniques to provide increasing throughput with less usage of area. [12] The generated VHDL is independent from FPGA platforms and supports code reuse

through the use of modules. ROCCC uses the streaming paradigm, in which data is represented by streams, a data format similar to the way arrays are stored in memory. These streams pass through a set of operations called kernels. This way of representing data makes it possible to express parallelism and is relatively easy mapped to the FPGA hardware. This paradigm removes the need for area-coslty soft-core processors.[13]
This streaming paradigm is also what enables the platform independence of the ROCCC hardware. As long as the data is delivered to the system in the form of a stream it can be used. Another important feature of ROCCC are the so-called smart buffers. These attempt to utilize the data-locality of certain applications to increase the performance. This is done by utilizing intelligent data reuse to minimize the number of off-chip memory acceses.

1.0.2 Xilinx Vivado High Level Synthesis Tool

Vivado High-Level Synthesis is part of the Xilinx Vivado design suite and is the product of the acquisition of AutoESL and the re-branding of their AutoPilot High-Level Synthesis tool. Vivado represents the next evolution of Xilinx tools fitting in their vision of an ”all programmable world”. It allows C, C++ and SystemC code to be synthesized into VHDL or Verilog code. Functional simulation can be done in C, which is a great improvement over the typical VHDL or Verilog simulation. The Vivado tool is based on the Eclipse platform and incorporates the C Development tool (CDT).[15] Due to the recent release of Vivado there is not much academic information to be found about this tool.