New Algorithms, Architectures And Applications For Reconfigurable Computing

11.3 The Byte Code Compiler (p.137- 138) The Byte Code Compiler is a very important feature of the VHBC approach, because it provides the means to compile working hardware designs, coded as a VHDL description, into a portable and efficient VHBC representation, thus removing the need for redesigning working hardware projects. The tool flow within the VHDL compiler can basically be divided into three main stages, the hardware synthesis, the net list to byte code conversion and the byte code optimization and scheduling. In the first stage the VHDL description is compiled into a net list of standard components and standard logic optimization is performed upon it, resulting in an optimized net list. The design of the compiler chain can be streamlined through the use of off-the-shelf hardware synthesis tools. Current implementations of the VHDL compiler make e.g. use of the FPGAExpress tool from Synopsis. These tools produce the anticipated code using a fairly standardized component library, as in the case of FPGA Express the SimPrim library from Xilinx. The resulting output of the first stage is converted to structural VHDL and passed on to the second stage. Most standard industry VHDL compilers with a support for FPGAs design readily provide the functionality needed for this step and can therefore be applied. In the second stage the components of the net list are substituted by VHBC fragments to form aVHBCinstruction stream. Before, however, the components are mapped to a VHBC representation, the net list is analyzed and optimized for VHBC. The optimization is necessary because commercial compilers targeting FPGAs usually output designs which contain large amounts of buffers to enhance signal integrity otherwise impaired by the routing of the signals. Furthermore, compilers show a tendency towards employing logic representations based on NAND or NOR gates, which are more ef.cient when cast into silicon. However, the resulting logic structure is more complex, revealing higher levels of logic. The code fragments used for substituting the logic components are based on predefined, general implementations of the latter in VHBC and are adjusted according to the data flow found in the structural description from the first phase, thus registers are allocated and the instructions are sequenced according to the data dependencies inherent. In the third stage the byte code sequence is optimized and scheduled

into blocks of independent instructions. First of all the data flow graph of the entire design is constructed, which is possible due to the lack of control flow instructions such as jumps. The code fragments introduced in the second stage are very general, so the resulting code gives a lot of room to code optimization techniques. One such technique is dead code elimination, which removes unnecessary instructions. The code is further optimized by applying predefined code substitution rules along the data paths, such as XOR extraction or doublenegation removal, to reduce the number of instructions and compact the code. The thus optimized code is scheduled using a list based scheduling scheme [14]. The objective of the scheduling is to group the instructions into code blocks such that the number of code blocks is minimal and the number of instructions per code block is evenly distributed among all code blocks. Furthermore, the time of data not being used, i.e. the number of clock cycles between the calculation of a datum and its use in another operation should be minimal. The scheduled code is then converted to the VHBC image format and the compiler flow concludes. EAN/ISBN : 9781402031281 Publisher(s): Springer Netherlands, Springer US Discussed keywords: Algorithmen, Rechnerarchitektur Format: ePub/PDF Author(s): Lysaght, Patrick - Rosenstiel, Wolfgang