Chapter 3: ASIP Design Methodology

3.1 Steps in ASIP Design

The methodology and the various steps used in the ASIP design flow are shown in figure 3.1 and are described below:

3.1.1 Application Analysis

In our case, the application is Klatt’s ‘C’ code for parametric speech synthesis [6]. The purpose of application analysis is to analyze the application for assessing the frequencies of different arithmetic operations, functions and data types used that can guide the design of micro-architecture and machine instruction set. In order to meet the needs of real-time operation, the delay specifications for each main operation are also worked out. The application analysis has been performed manually. Details of the application analysis are given in chapter 4.

3.1.2 Datapath (execution unit) micro-architecture design

Datapath (execution unit) micro-architecture design involves exploration of top-level hardware architectural diagram of the execution unit including various functional blocks in the datapath (execution unit), registers, memories, buses and I/O ports with an over-all view of realizing real-time execution of the application. All the functional blocks are described using behavioral-level VHDL blocks and delay constraints on the blocks are speculated in view of the over-all real-time execution requirement of the application. They serve as specifications for the implementation of functional blocks as synthesizable RT-level VHDL code in later steps. Details of the datapath (execution unit) micro-architecture design are given in chapter 5.
Figure 3.1: ASIP Design Flow Followed
3.1.3 Instruction Set Design

Based on the results of application analysis, different functions/operations required to be performed are identified to define their corresponding instructions. Frequently occurring computational patterns in the application (involving a fixed sequence of operations and data transfers) are encapsulated in a single instruction to speed-up their execution (through provision of high performance hardware functional units in the datapath (execution unit) micro-architecture design step).

In view of the number of different types of functional blocks in the datapath unit (execution unit), the number and types of operands required by them, the number of registers, the number, types and sizes of memories, control requirements of input/outputs and the number of flags in the datapath, the format(s) and field sizes pertaining to the instruction set are arrived at. These in turn will determine the choice of an over-all bit-width for the instructions. Details of the instruction set design are given in chapter 5.

3.1.4 Instruction Set Implementation

After deciding the machine instruction set, the next step is to implement the instruction set. The hardware implementation of the machine instruction set over a given micro-architecture of the datapath (execution unit) results in the design of the control part of the ASIP. However, in order to test the functional correctness of hardware implementation of the instruction set one needs a reference to check against. In our case this reference is provided by a software implementation of the machine instruction set – resulting into a machine instruction set simulator.

In our work we have used ‘C’ language to implement the machine instruction set simulator and it is described in detail in chapter 8.

For hardware implementation of the machine instruction set (over the given datapath micro-architecture), we have used the flowchart method [32].
3.1.5 Design and Implementation of Control Part

We have followed the strategy of micro-coding to design the control part of the ASIP. The technique of hardware flowcharting has been used to develop the control sequences required to implement the functionality of each instruction using the datapath (execution unit) micro-architecture already defined.

A horizontal micro-coding strategy has been used to define the number and bit-widths of different fields in the micro-instruction. The control part is functionally verified by simulating its RT-level VHDL model together with the RT-level model of the datapath (execution unit) for every instruction in the instruction set. Timing of the control part is also verified against the delay specifications provided for it. Details of the design and implementation of control part are given in chapter 6.

3.1.6 Exploration of Hardware Algorithms and their FPGA Implementations for Datapath (execution unit) Functional Units

Design space explorations are carried out for the implementation of each functional unit in the datapath (execution unit) to achieve its functionality within the delay constraint stipulated. Since many functional units perform sufficiently complex computational tasks, one needs to explore the various hardware algorithms and their different implementations for each function to select the best suited hardware algorithm and its particular hardware implementation that meets the delay constraints specified for the function using the smallest number of gates required for the implementation of the function. This step ultimately produces synthesizable RT-level VHDL code for each functional block in the datapath (execution unit) whose synthesized version after place and route on the target FPGA has successfully met the delay constraints specified for the function. Details of the exploration of hardware algorithms and their FPGA implementations for datapath (execution unit) functional units and the testing approach used for the functional units/blocks are given in chapter 7.
3.1.7 Integration of ASIP and its FPGA Implementation

A top-level module is created for the ASIP in VHDL in which all the blocks are instantiated at the Register-Transfer-Level (RTL). After integrating the whole design at RT-level, it is translated into gate-level netlist and optimized for the target FPGA. Synthesis and implementation have been carried out using Xilinx’s Project Navigator, which is the primary user interface for ISE. After the hardware synthesis step, post-synthesis simulation (both functional and timing) is done at the gate-level to check functionality and timing. Finally the file with .bit extension generated after the configuration step is used to program the target Xilinx FPGA device (Spartan-IIE XC2S3000E). Details of the Integration of ASIP and its FPGA implementation are given in chapter 9.