Abstract The User Guided Synthesis approach targets the generation of coprocessor under timing and resource constraints. Unlike other approaches that discover the architecture through a specific interpretation of the source code, this approach requires that the user guides the synthesis by specifying a draft of its data-path architecture. By providing this information, the user can get nearly the expected design in one shot instead of obtaining an acceptable design after an iterative process. Of course, providing a data-path draft limits its use to circuit designers.

The approach requires three inputs: The first input is the description of the algorithm to be hardwired. It is purely functional, and does not contain any statement or pragma indicating cycle boundaries. The second input is a draft of the data-path on which the algorithm is to be executed. The third one is the target frequency of the generated hardware.

The synthesis method is organized in two main tasks. The first task, called Coarse Grain Scheduling, targets the generation of a fully functional data-path. Using the functional input and the draft of the data-path (DDP), that basically is a directed graph whose nodes are functional or memorization operators and whose arcs indicate the authorized data-flow among the nodes, this task generates two outputs:

- The first one is a RT level synthesizable description of the final coprocessor datapath, by mapping the instructions of the functional description on the DDP.
- The second one is a coarse grain finite state machine in which each operator takes a constant amount of time to execute. It describes the flow of control without knowledge of the exact timing of the operators, but exhibits the parallelism among the instruction flow.

The data-path is synthesized, placed and routed with back-end tools. After that, the timings such as propagation, set-up and hold-times, are extracted and the second task, called Fine Grain Scheduling, takes place. It basically performs the retiming of the Coarse Grain finite state machine taking into account the target frequency and the fine timings of the data-path implementation.
Compared to the classical High Level Synthesis approaches, the User Guided Synthesis induces new algorithmic problems. For Coarse Grain Scheduling, it consists of finding whether an arithmetic and logic expression of the algorithmic input can be mapped on the given draft data-path or not, and when several mappings are found, to choose the one that maximizes the parallelism and minimizes the added resources. So the Coarse Grain Scheduling can be seen as a classical compiler, the differences being firstly that the target instruction set is not hardwired in the compiler but described fully or partially in the draft data-path, and secondly that a small amount of hardware can be added by the tools to optimize speed.

For Fine Grain Scheduling, it consists of reorganizing the finite state machine to ensure that the data-path commands are synchronized with the execution delays of the operators they control. The fine grain scheduling also poses interesting algorithmic problems, both in optimization and in scheduling.

**Keywords:** Behavioral synthesis, FSM retiming, Design space exploration, Scheduling, Resource binding, Compilation

### 10.1 Introduction

#### 10.1.1 Enhanced Y Chart

The Y chart representation proposed by Gajski [10] can not accurately represent the recent synthesis and high level synthesis tools. So we have enhanced it as shown Fig. 10.1. In the enhanced Y chart, a control flow level is inserted between the system and data flow levels. It corresponds to the coprocessor synthesis and allows to distinguish co-design from high level synthesis. In the structural view, a coprocessor is usually a data-path controlled by a FSM. There are two possible types of description in the behavioral view. The synchronized description is more or less a Register Transfer Language where cycle boundaries are explicitly set by the language. The non-synchronized description is based on imperative languages such as C, PASCAL, and in this type of description, the cycle boundaries are not given. As shown in the Fig. 10.1, High Level Synthesis consists of making a structural description of a circuit from a non-synchronized description of a coprocessor. A usual approach [6, 14, 24] is to generate the synchronized description of the coprocessor (plain arrow in Fig. 10.1) and to submit it to a CAD frameworks having an efficient RTL synthesis tool.

#### 10.1.2 UGH Overview

The multiple arrows noted 1.a, 1.b and 1.c on the Fig. 10.2a describe the User Guided High level synthesis tool (UGH). It starts from a non-synchronized description of an algorithm and generates a structural description of the coprocessor (arrow