

Introduction

This book presents a new methodology for high level and logic design of complicated digital systems. This methodology is based on Algorithmic State Machine (ASM) transformations (composition, minimization, extraction, etc.), special algorithms for Data Path and Control Unit design and a very fast optimizing synthesis of FSMs as well as combinational circuits with hardly any constraints on their size, i.e., the number of inputs, outputs and states. Design tools supporting this methodology allow us to implement, check and estimate many possible design versions very fast, to find an optimized decision of a design problem and to simplify the verification problem for digital systems.

Let us suppose that the task is to design a digital system. Problem orientation regarding this system is nonessential – it can be a processor, a robot, a controller, etc. If the system is rather complicated, it is possible to pick out some subbehaviors (modes) in its behavior. For a processor it can be an instruction or a set of instructions that can be described together; for a mobile robot – its different modes (cruise, follow, avoid, escape etc.). We also suppose that any digital system is usually regarded as a composition of a *Control unit* and an *Operational unit (Data path)*. In a processor, for example, a data path contains such regular blocks as memory, registers, ALU, counters, coders, encoders, multiplexers, demultiplexers, etc. A control unit produces a sequence of control signals that force an implementation of microoperations in a data path. Let us discuss the main stages of the suggested design process in more details.

Stage 1 Design ASMs G_1, \dots, G_M for each separate mode. We can present these ASMs in VHDL or use the special tool *ASM Creator* from the EDA tool *Abelite* supported by the described design methodology. In *Abelite*, it is very simple to draw an ASM and compile it in VHDL and/or several other representations. It is really important that an ASM may contain any number of *generalized operators*. Each of such operators is an ASM itself and it will be automatically inserted in the combined ASM at the fourth stage. Moreover, there are no restrictions on the number of such generalized operators in an ASM and on the number of included levels – each of such operators can contain any number of generalized operators itself (see Sections 4.1.1 and 7.2.1).

Stage 2 Combine separate ASMs into one combined functional ASM. After constructing separate ASMs we combine them into one combined functional ASM still containing generalized operators (see Sections 6.4 and 7.3.1). I would like to underline that at this stage each microoperation is presented at the functional level. Really, we do not have the real architecture for our project, we only know some units of our future Data path. Thus, microoperations at this level are similar to assignments of variables in some programming language and are not connected with any specific Data path. During ASM combining we minimize the number of operator vertices in the combined ASM. If several ASMs contain the same operator vertex, there will be only one such operator vertex in the combined ASM.

Stage 3 Minimize the combined functional ASM. At this stage, the number of conditional vertices in the combined ASM is minimized (Sections 6.2 and 7.3.1). Such minimization allows us to reduce dramatically the number of vertices in the ASM (sometimes for two or three times) and to reduce the complexity of logic circuits at the stage of logic design.

Stage 4 Include generalized operators (see Sections 7.2.1). At this stage, generalized operators constructed at the first stage are included into the minimized ASM

constructed at the previous stage (see Sections 7.3.1). It is the last stage of the functional ASM design.

Stage 5 Data path synthesis. First, we construct a Connection graph from the functional ASM design on Stage 4. Such a graph contains a list of sources and targets for each component of an operational unit and some metrics that will be used in the optimization of the Data path (see Sections 7.3.2, 7.3.3). Next we construct an optimized List of parallel microoperations to increase the speed of the design system (see Sections 7.3.2 and 7.3.3). Then we design the Graph of incompatibility (see Section 7.3.3) from the Connection graph and the List of parallel microoperations. On the final step of Data path synthesis we construct Muxes (buses) by coloring the Graph of incompatibility and the List of direct connections from the Connection graph.

Stage 6 Control unit design. Using the functional ASM (stage 4) and the Muxes and the List of direct connections (stage 5) we immediately construct the structural ASM. This ASM describes the behavior of the Control unit corresponding to the Data Path (see Section 7.4.1). On the last step of this stage we construct the Finite state machine (Sections 4.2 and 7.4) and its multilevel logic circuit (Section 5.5).

Stage 7 VHDL code design. The Data path constructed according to our design methodology does not contain any “cloud” (irregular) circuits. It makes it possible to simplify considerably VHDL or Verilog code for the Data path using the structure style of VHDL to combine VHDL or Verilog codes of units (Section 8.3.1). Moreover, the presented formal methods of system design allow us to formalize the design of test bench for the Data path and to reduce it considerably (Section 8.3.2). VHDL code for the Control unit is constructed automatically (Section 8.4). VHDL code for the top level of the system is the result of combining VHDL codes of the Data path and the Control unit (Section 8.5).

Many special methods and algorithms presented in this book, such as

- Synthesis of logic circuits with NOR-NAND gates (Chapter 1);
- Introduction of Combined model of an automaton (Chapter 2);
- Synthesis of FSMs Mealy, Moore and their Combined model from ASM (Chapter 4);
- FSM decomposition (Chapter 4);
- Synthesis of multilevel and multioutput circuits for FSMs (Chapter 5);
- Minimization of conditional vertices in ASMs (Chapter 6);
- Combining of ASMs with generalized operators (Chapter 6 and Chapter 7);
- Design methodology for high level synthesis (Chapter 7)

were developed by the author during his long-term work in the design of digital systems and EDA tools and teaching various courses in different colleges and universities. Some methods described in the book are published for the first time. These methods were implemented in the EDA tool Abelite for logic and system design of very complicated digital circuits and systems. Most of the examples in this book were constructed using this tool.