CAD System SYNTHESIS1 for FSMs Synthesis

CAD system Synthesis1 for logic synthesis of very complex finite state machines (FSM) with hardly any constraints on their size, that is, the number of inputs, outputs and states.
Why is it so important to synthesize Finite State Machines (FSM)? There are at least two serious reasons:
1. FSM, or a control unit, is unique in each project. To design a data path you can use very large library units from the previous projects. It is not possible for FSM as you cannot use former projects for new FSM's design.
2. In the latest few years, a share of FSMs in each project was considerably increased. There are also some reasons for this tendency:
   (a) implementation of a software as a hardware;
   (b) complex specialized processors with very complex embedded functions;
   (c) parallel processors with a complex control.

SYNTHESIS1 contains the following subsystems:
1. Algorithmic State Machine Transformer
Algorithmic State Machine (ASM) in SYNTHESIS1 is presented in VHDL or as two-connected list (a text file also). The following procedures are implemented in ASM Transformer:
(a) ***Show ASM***. This procedure displays a picture of ASM at the screen.
(b) ***Print ASM***. This procedure prints a picture of ASM into post-script file.
(c) ***Check ASM***. This procedure verifies the correctness of ASM.
(d) ***ASM minimization***. This procedure minimizes the number of vertices in ASM. As a rule at the next steps, the number of states in FSM and a complexity of FSM logic circuit depend on the number of vertices in the corresponding ASM.
(e) ***ASMs combining***. This procedure combines separate ASMs into one ASM and minimizes the number of vertices in this ASM.
(f) ***ASM decomposition***. This procedure decomposes a given ASM into the set of component ASMs with given constraints in the number of logical conditions (input variables), microoperations (output variables) and vertices.
2. Finite State Machine Synthesizer
The following procedures are implemented in Finite State Machine (FSM) Synthesizer:
(a) ***Constructing of FSM transition table***. ASM description in VHDL or ASM as two connected list are the inputs to this procedure which constructs 16 various types of FSMs tables - Mealy and Moore, transition and structure, direct and reverse etc. The output of this procedure is FSM in VHDL or a text file with FSM table. In SYNTHESIS1, there exist the procedures for transformation of FSM into special FSM representation in CAD systems Synopsys and Compass.
(b) ***State assignment***. FSM in VHDL or a text file with FSM table list are the inputs to this procedure. A text file with FSM table, containing states codes and input memory functions, is the output of this procedure. A designer can also enter his own state codes from the keyboard.
(c) ***FSM decomposition***. This procedure decomposes a given FSM into the set of connected component FSMs with given constraints to the complexity of component FSMs (a number of inputs, outputs, states and rows in their transition tables).
3. Logic Circuit Synthesizer
(a) ***Logic circuit synthesis with gates***. FSM in VHDL or a text file with FSM table are the inputs into this subsystem of Logic Circuit Synthesizer, multilevel logic circuit in VHDL or a netlist as a text file are its output. A designer can enter constraints to the number of inputs in gates and/or to the number of levels (the length of a maximum path) in the logic circuit. The special procedure finds the level of the circuit (if the circuit was constructed without constraints to the number of levels) and the distances of each gate from the input and to the output. Another procedure simulates a logic circuit - it checks that the logic circuit really

implements FSM and, if this procedure finds a lack of correspondence, it localizes the place of incompatibility.

(b) ***Synthesis of logic circuits with Look-Up-Tables (LUTs) for FPGA technology***. The design problem for FSM logic circuit with LUT(k) is formulated as follows: it is necessary to decompose a logic circuit into such subcircuits (cells, or LUTs) that each cell implements one function with not more than k inputs or two functions with not more than $k-1$ inputs, but in the last case, the common number of inputs into cell should also not exceed k inputs. FSM in VHDL or a text file with FSM table are the inputs to this subsystem, its output is a text file with a netlist containing LUTs. SYNTHESIS1 allows to construct circuits with k=4-8. As in the synthesis of logic circuits with gates, the special procedure simulates a logic circuit - it checks that the logic circuit really implements FSM and, if this procedure finds a lack of correspondence, it localizes the place of incompatibility.

(c) ***Synthesis of logic circuits with PLAs.*** FSM in VHDL or a text file with FSM table are the inputs into this subsystem of Logic Circuit Synthesizer, a designer should also enter the maximal numbers of inputs (s), outputs (t) and rows (q) in Programmable Logic Array (PLA) used. The output of this subsystem is the text file with programming tables for PLAs. The techniques used in this subsystem are based at the circuit factorization into factor-circuits and such that each factor-circuits has no more than s inputs, t outputs and q terms. As in the another subsystems of Logic Circuit Synthesizer, the special procedure checks that the logic circuit really implements FSM and, if this procedure finds a lack of correspondence, it localizes the place of incompatibility.

(d) ***Synthesis of logic circuits with PLAMs.*** FSM in VHDL or a text file with FSM table are also the inputs into this subsystem of Logic Circuit Synthesizer, designer should also enter the maximal numbers of inputs (s), outputs (t), rows (q) and flip-flops (r) in Programmable Logic Array with Memory (PLAM) used. The output is the text file with programming tables for PLAMs. This subsystem uses the procedure "FSM decomposition" with given constraints s, t, q, r to the complexity of component FSMs.

(e) ***Matrix realization of FSM.*** FSM in VHDL or a text file with FSM table are the inputs into this subsystem, its output contains programming tables for each matrix in the multimatrix logic circuit. The techniques, used here, minimize the chip area. They are base on the circuit representation as a composition of four or six matrices, input variables replacement and special methods for state assignment and microinstructions coding. At the same time, this subsystem constructs a classical two-matrices logic circuit. Our numerous experiments showed that a multimatrix realization reduces chip area by 30-40%.

SYNTHESIS1 is implemented on IBM PC. On Pentium 200Mhz even for synthesis of logic circuits for very complex FSMs it needs not more than one second.

We will demonstrate SYNTHESIS1 and will give the results of experiments with this system and with Synopsys, Xilinx and Altera