ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA ELEKTROTECHNICKÁ

CZECH TECHNICAL UNIVERSITY IN PRAGUE
FACULTY OF ELECTRICAL ENGINEERING

Ing. Hana Kubátová, CSc.

Kódování vnitřních stavů synchronního sekvenčního obvodu

Synchronous Sequential Circuit Internal States Coding

Summary

This paper describes the well-known subject – the problem of internal states coding for synchronous sequential circuits (described by a formal FSM model). But our aim to search for the "optimal" encoding of the internal states of a FSM, is especially leading up to VLSI regular structures – FPGA or CPLD.

Our encoding methods are neither based on the minimum number of internal states, nor on the minimum number of used flip-flops in their hardware implementation, because these constraints are not suitable for regular structures, FPGAs or CPLDs. Here different types of evaluations and appropriate decompositions with respect to other quantitative properties must be performed. New heuristic algorithms for optimal internal states coding with respect to space, time, power or reliability demands have to be searched. Therefore this paper deals with the possibility of the description and decomposition of the finite state machine (FSM) according its quantitative properties. The aim is to obtain better placement of a designed FSM to the concrete type and size of the FPGA.

Several methods of encoding of the internal states are compared with respect to the space (the number of CLB blocks) and time (maximal working frequency) characteristics. It evaluates the FSM benchmarks and seeks for such qualitative properties to choose the best method for encoding before performing all EDA tool algorithms since these processes could be time consuming. The new method for encoding the internal FSM states based on the combination of "one-hot" and "minimum-lengths" methods, called FEL-code is presented.

All results are obtained on the base of many experiments and verified by several EDA tools performed by author and her PhD, master and bachelor students.

Souhrn

Příspěvek popisuje známý problém – kódování vnitřních stavů synchronního sekvenčního obvodu (popsaného formálním modelem– konečným automatem). Hlavním cílem při hledání „optimálního“ kódování vnitřních stavů bylo zohlednění cílové implementační struktury - regulární VLSI struktury konkrétního hradlového pole FPGA nebo CPLD.

Naše kódovací metoda není založena na minimálním počtu vnitřních stavů ani na minimálním počtu nutných klopných obvodů, protože tato omezení nejsou pro použité struktury vhodná. Je nutné provést různá ohodnocení a vhodná rozdělení (dekompozice) návrhu s ohledem na kvantitativní parametry závislé na konkrétním navrhovaném obvodu. Výzkum autorky hledá nové heuristické algoritmy pro kódování vnitřních stavů s ohledem na minimální plochu, maximální pracovní kmitočet, spotřebu i spolehlivost. Proto tento příspěvek popisuje různé možnosti popisu i částečné dekompozice sekvenčního obvodu (konečného automatu) podle jeho kvantitativních vlastností.

Několik základních metod pro kódování vnitřních stavů je srovnáno podle zabraného prostoru (počtu CLB bloků v FPGA) a časových parametrů (maximálního pracovního kmitočtu). Zkušební úlohy (benchmarky) pro sekvenční obvody jsou ohodnoceny podle svých kvantitativních vlastností, aby bylo možné vybrat nejlepší metodu kódování jejich vnitřních stavů bez nutnosti zkoušet různé metody pomocí profesionálních návrhových prostředků, protože tento proces může být časově velmi náročný. Je prezentována zcela nová metoda kódování vnitřních stavů nazvaná FEL-code, která vhodným heuristickým způsobem kombinuje kódování „1 z n“ a „binární“.

Všechny dosažené a prezentované výsledky jsou získané z mnoha experimentů a jsou ověřeny autorkou a jejími studenty v rámci doktorských, diplomových, bakalářských a semestrálních projektů.

Klíčová slova

Konečný automat, jazyk pro popis hardwaru (HDL), graf přechodů, metoda kódování, dekompozice, zkušební úloha, hradlové pole

Key words

Finite State Machine (FSM), Hardware Description Language (HDL, VHDL), State Transition Graph (STG), encoding method, decomposition, benchmark, FPGA

Content

1 Introduction 6
2 Motivation and Design Tools 9
3 Theory and methods 10
3.1 Encoding methods 11
3.2 The „FEL-code“ method 13
4 Experiments 16
4.1 Information about the FSM benchmarks 16
4.2 The way of VHDL description 17
4.3 Special testing FSMs 20
4.4 FSM internal state coding experiments 20
5 Conclusions 23
References 27

1 Introduction

Today's hardware implementation base with high integration densities is getting even more complicated and structural. The Moore's law says that the number of transistors on a chip doubles annually [29]. This means not only quantitative, but also qualitative changes in the digital design process. The optimization methods are very important due to the possible miniaturization of chips used in the space or medicine. Regular structures like FPGAs and CPLDs integrated in a system-on-chip (SoC) design need new design methodology for hardware-software codesign, new testing methods, and design methodologies with respect to easy testing (built-in self-test – BIST), methods that enable dynamic reconfiguration of the part of the designed circuit to save space or to increase reliability. Formal specification and verification at all levels of this structural design must be ensured. Reconfigurable hardware is useful in the design of dependable computing systems for applications such as nuclear reactor control, fly-by-wire systems, communication platforms, and remote applications-space stations and satellites, for example. (Dependability comprises reliability, availability, testability, maintainability and fault-tolerance.), [7], [27], [28].

Design automation for digital circuits can be divided into three steps: high-level synthesis, logic design, and physical design. High level synthesis transforms the behavior level specification into a register-transfer level structure that implements the behavior [10]. During logic design, the description of a circuit is converted into a set of technology-specific gates and interconnects. Each of the gates realizes a particular function. The objective of logic design is to minimize the active (gate) area, critical path delay, and to assure testability. The physical design step involves placing and interconnecting the gates. The goal is to fit the design into as compact area as possible. The design process is divided into these separate steps to make the process manageable. However, this approach can result in the loss of optimality. In today's aggressive designs it is important to take advantage of all existing opportunities. For example, there are many arbitrary decisions made during the logic design which create hardship during the physical design. In a Field Programmable Gate Array (FPGA), where the routing resources are limited and the role of wiring is more crucial than in the mask-programmable technologies, it is important to explore all degrees of freedom which exist in the specification, [30].

When implementing digital circuits with FPGAs, designers are always concerned with limited and inflexible resources. Additionally, fulfillment of timing, space, dependability and power consumption constraints is always an issue. This text describes a series of algorithms and possible improvements

developed to assist a designer fitting a circuit into an FPGA chip with respect to the additional pre-conditions. Some of these improvements and algorithms traditionally used in design process became disadvantageous for FPGAs. In the traditional circuit design flow, the logic design step is completely separated from physical design step. There are many arbitrary decisions made during logic design which create difficulties in the physical design. This is particularly true in FPGA designs, where the routing resources are limited. It is very important to explore the freedom between the interaction of the logic and physical design steps. Fig. 1 describes such schedule from informal system requirements to its physical implementation (left part), but with the necessary feedback including testing and verification at all levels of the design process.

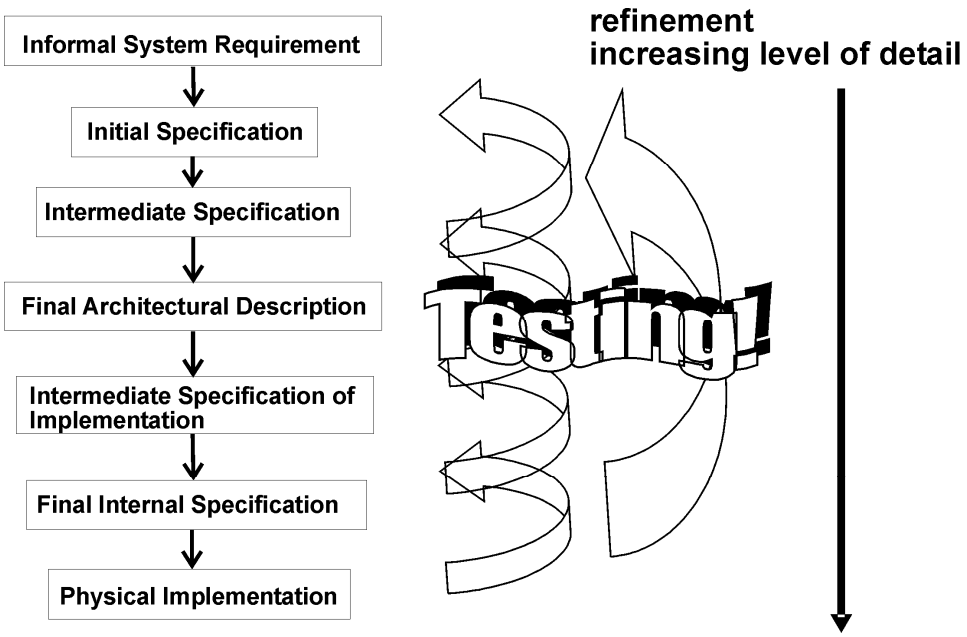


Fig. 1 Block diagram for the design methodology and the design refinement

An FPGA consists of an array of identical logic blocks, segmented wires, connection blocks and switch boxes. In a Look-Up Table (LUT) architecture, each logic block can implement any k-input Boolean function. To interconnect these logic blocks, we need to program switches inside the connection blocks and switch boxes. Only a finite number of switches is available, so the routing resources are limited and inflexible. In order to fit a circuit into an FPGA with a given size, the realization must not exceed the number of available resources. As a result, it is almost always beneficial to implement a circuit with fewer rather than more gates. With limited routing resources on FPGA chips, one of the essential problems in design is to route all signal networks. However, without

actually attempting to route a chip, a designer is not able to say whether a design can be fully routed. It is often possible to transform an unroutable into a routable design by slightly altering its logic structure. The ramifications of these incremental changes in the logic structure, when most of a circuit is placed and routed, are relatively easy to predict, [7].

The traditional design flow for FPGAs consists of four steps. In the first step, logic expression of a circuit is minimized, [3]. Then, a technology mapper maps the Boolean network into a minimum number of k-input LUTs [6], [10]. Placement and routing then follow, see Figs. 2, 3. During the physical design process, the logic information of the circuit is no longer used. Only the topology, typically represented as a graph, is retained.

This paper presents the FSM internal states coding with respect to the final implementation in some FPGA structure.

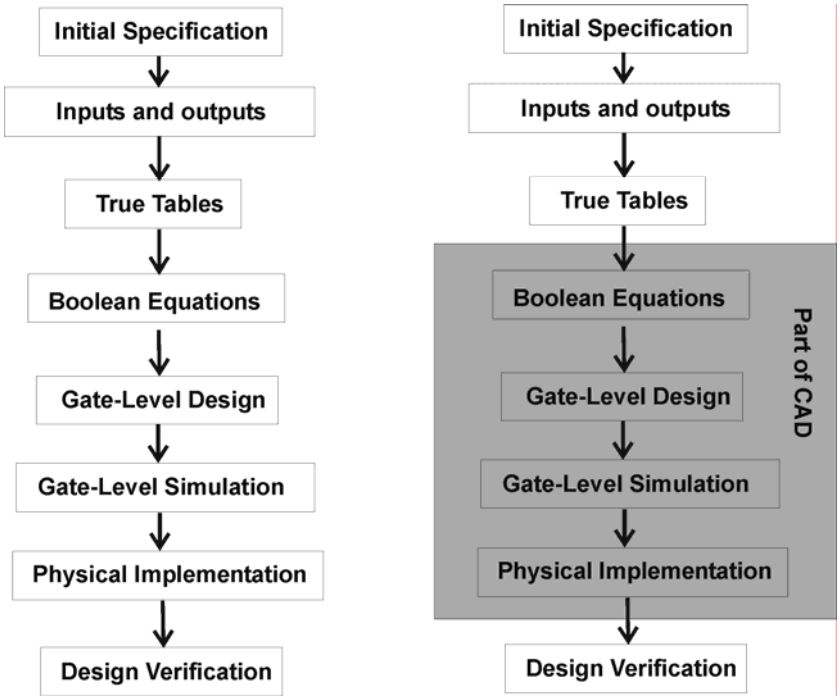


Fig. 2 A traditional digital design flow a) and its simplification by the FPGA design tool b)

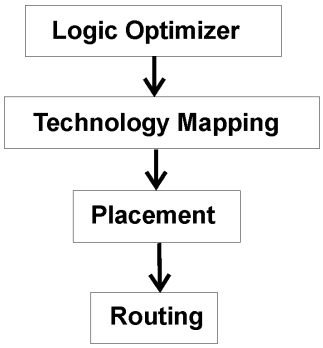


Fig. 3 A traditional FPGA design flow – the gray part from Fig. 2.

2 Motivation and Design Tools

This research was initiated by the problem "how to implement large sequential circuits into relatively small FPGAs". There are two ways how to do it – to optimize the circuitry (that means appropriate mapping to the predefined regular structure mainly by the optimal selection of internal coding of states), or to decompose the circuit, [5] to implement it in two or more FPGAs. The results were partially published in [18] – [24].

Most research reports and other materials devoted to searching for the "optimal" encoding of the internal states of a FSM are based on the minimum number of internal states, and sometimes also on the minimum number of used flip-flops in their hardware implementation [25]. The only method how to get really optimal results is to test all possibilities. But sometimes "wasting" the internal states or flip-flops is a better solution due to the speed of the designed circuit. Most such "optimal" encoding methods are not suitable for regular structures, e.g. different types of FPGAs or CPLDs, because different types of decompositions or designs with respect to other quantitative properties must be performed, other cost functions have to be exploited in appropriate heuristic algorithms for optimal internal states coding with respect to space, time, power or reliability demands.

Several types of sequential circuit benchmarks and their internal state coding were compared to search for the relation between the type of the circuit (the number of the internal states, inputs, outputs, cycles, branching) and the encoding method with respect to their implementation in a XILINX FPGA.

EDA tools XILINX FOUNDATION v2.1i, XILINX ISE 5.2i and ALTERA Quartus II version 2.1 SP1 were used during our experiments. We have used the benchmarks from the Internet in KISS2 format [31], some encoding algorithms from JEDI program and system SIS 1.2 [33], [34]. First of

all we classified the FSM benchmarks to know their quantitative characteristics: the number of internal states, inputs, outputs, transitions (i.e. the number of arcs in the state transition graph - STG), the maximum number of input arcs, the maximum number of output arcs to and from STG nodes, etc. We compared eight encoding methods: "one-hot", "minimum-lengths" (binary), "Johnson" and "Gray" implemented in the XILINX FOUNDATION system. "Fan-in" and "Fan-out" oriented algorithms, the algorithm "FAN" connecting Fan-in and Fan-out ones and the "two-hots" methods were implemented. The second group of our experiments studied different FSM decompositions. Our original method called a "FEL-code" is based on these experimental results and is presented in the section 3.2. The final results (the number of CLB blocks and maximum frequency) were obtained for a concrete FPGA implementation (Spartan XCS05-PC84).

3 Theory and methods

The designed circuits can be described by several ways [1], [4], [8], [10]: by means of the next-state and output Boolean equations, state and output tables or by state diagrams. Each of them completely specifies any sequential circuit. All of these descriptions are based on the model of the finite-state machine (FSM).

Definition 1. *A finite-state machine (FSM) is a quintuple FSM = <Q, X, Y, δ, λ> where*

- *Q is a finite set of states (internal states),*
- *X is a finite set of inputs,*
- *Y is a finite set of outputs,*
- *δ is a next-state function δ: Q x X → Q,*
- *λ is an output function λ: Q x X → Y.*

The FSM model assumes that the system time is divided into uniform intervals and that the transitions from one state to another occur only at the beginning of each time interval. Therefore, the next-state function δ defines what the state of the FSM will be in the next time interval given the state and input values in the present interval. The output function determines the output values in the present state. There are two different types of FSM according the type of output function. A Moore FSM (which is sometimes called state-based) has each output symbol assigned to each state, its output function is a mapping $Q \rightarrow Y$. A Mealy FSM (input-based) has the output function λ defined as mapping Q x $X \rightarrow Y$. It means that the output can earlier react to the input signal

than a Moore FSM. A Mealy FSM can be transformed to a Moore FSM and vice versa but their implementation by FPGA can lead to the totally different quantitative characteristics.

The final result depends on many aspects with respect not only to the functionality, but also to the quantitative characteristics (delay, maximal working frequency, space, power consumption, etc.), Moore or Mealy types of FSM, the state minimization, the state encoding, implementation bases, a FSM description style. Some of them are contra-productive, e.g. the “one-hot” encoding method is always better with respect to the power consumption, but it is not always good with respect to the number of used memory elements.

3.1 Encoding methods

A “one-hot” method uses the same number of bits as the number of internal states. This great number of internal variables is its main disadvantage. The states that have the same next state for a given input, should be given adjacent assignments ("Fan-out oriented"). The states that are the next states of the same state, should be given adjacent assignments ("Fan-in oriented"). The states which have the same output for a given input should be given adjacent assignments, which will help to cover the 1's in the output Karnaugh-maps; "output oriented" method.

Very popular and frequently used method is the “minimum-length” code (also called “binary”) that uses the minimum number of internal variables, and the Gray code with the same characteristics and adjacent codes assigned to consequent states. The method “two-hots” uses such a combination of two “1” to distinguish all states.

Tab. 1. Examples of codes for 6 internal states

	Binary	Gray	Johnson	one-hot	two-hots
St1	000	000	000	000001	0011
St2	001	001	001	000010	0101
St3	010	011	011	000100	1001
St4	011	010	111	001000	0110
St5	100	110	110	010000	1010
St6	101	111	100	100000	1100

Initial results based on these eight methods and benchmark characteristics were presented in [18] - [24]. We have found out that the most successful methods are the “minimum-length” and “one-hot” ones. The “minimum-length” encoding is better than the “one-hot” encoding for small FSMs and for FSMs that fulfill the following condition: a STG (State Transition Graph) describing

the FSM should be complete or nearly complete. If the ratio of the average output degree of the node to the number of states is greater than 0.7, it is better to use the "minimum-length" code. On the contrary, the "one-hot" encoding is better when this ratio is low. Let's define this qualitative property of the FSM as a characteristic parameter AN:

Definition 2. *A FSM characteristic parameter AN - an average node branching - is defined by the equation*

$$AN = AverageOutEdges/(NumberOfStates - 1)$$

where AverageOutEdges is an average output number of edges for all nodes and NumberOfStates is the number of states for a FSM.

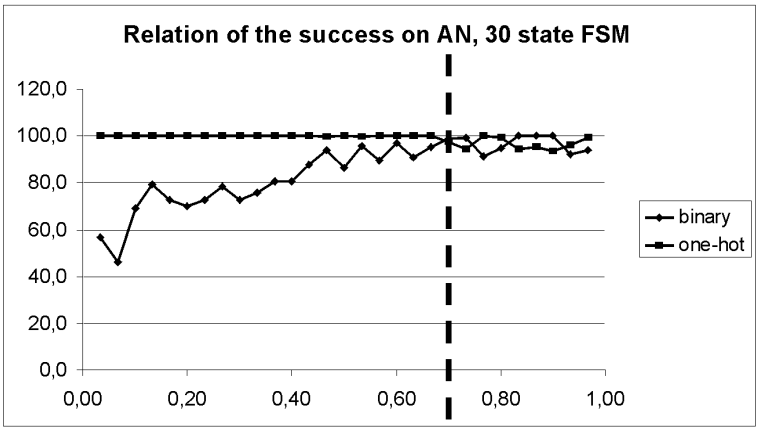


Fig. 4. The comparison of „minimum-length" and "one-hot" enencoding with respect to AN

The value AN = 0.7 was experimentally verified on benchmarks and on our specially generated test FSMs - Moore type FSMs with the determined numbers of internal states and the determined numbers of the transitions from the internal states. Our FSM has the STG with a strictly defined number of edges from all states. This number of output edges must be the same for each internal state. The resulting format is the KISS2 format – e.g. 4.kiss testing FSM has the STG with four edges from each internal state (node). The next state connections were generated randomly to overcome the XILINX FOUNDATION optimization for the counter design. The relationship between "one-hot" and „minimum-length" encoding methods is illustrated in Fig. 4. The comparison was made for FSMs with 30 internal states and different branching. The number of transitions from all states is expressed by AN (axis X). The axis Y expresses the percentage success of the methods with respect to the space (the minimum number of CLBs

for all encoding methods and every test FSM is 100%). It can be seen that for lower branching (smaller AN) the "one-hot" code is always better until AN = 0.7.

3.2 The „FEL-code" method

Our method combines both the „one-hot" and „minimum-length" encoding methods. It is based on the partially FSM internal state decomposition such as several level, cascade, general, where the number of levels depends on the FSM properties, [12], [13], [14], [15] and[16]. The global number of states of the decomposed FSM is equal to the original number of FSM internal states. The internal states are divided into groups (sets of internal states) with many connections between states (for these "strongly connected" states the „minimum-length" encoding is better to use), see Fig. 5. The internal state code is composed of the „minimum-length" part (a serial number of the state in its set in binary notation) and the one-hot part (a serial number of a set in one-hot notation). The number of „minimum-length" part bits is equal to b, where 2^b is greater or equal to the maximum number of the states in sets and the number of one-hot part bits corresponds to the number of sets.

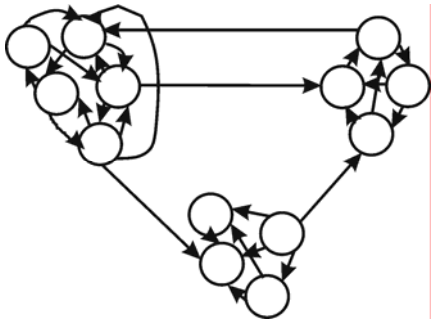


Fig. 5. FEL-code basis

The global algorithm could be described as follows:

1. Place all FSM internal states Qi to the set $S0$.
2. Select the state Qi (from $S0$) with the greatest number of transitions to other disjoint states from $S0$. Take away Qi from $S0$, Qi becomes the first member of the new set $S_{group.}$
3. Construct the set of neighboring internal states of all members of S_{group} – $S_{neighbor}$. Compute the score expressing the placement suitability for a state Qj into S_{group}, for all states from $S_{neighbor}$. Add the state with the highest score to S_{group} .

   The score is a sum of:

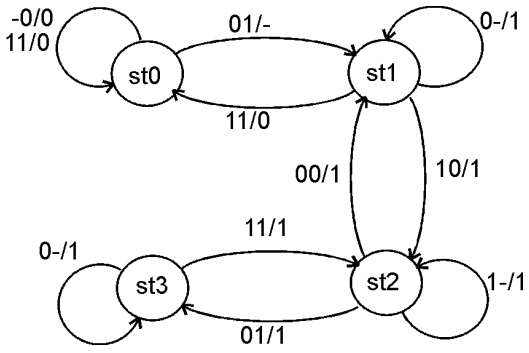
a. the number of transitions from Qj to all states from Sgroup multiplied by the constant 10;
b. the number of such states from Sgroup for which exists the transition from Qj into those ones multiplied by the constant 20;
c. the number of transitions from Qj to all neighboring internal states from Sgroup (i.e. to all states from Sneighbor) multiplied by the constant 3;
d. the number of such states from Sneighbor for which exists the transition from Qj to those ones multiplied by the constant 6;
e. the number of transitions from all internal states from Sgroup to Qj multiplied by the constant 10;
f. the number of such states from Sgroup the transition exists from those ones into Qj multiplied by the constant 20;
g. the number of transitions from all neighboring states of Sgroup (placed in Sneighbor) to Qj multiplied by the constant 3;
h. the number of the neighboring states in Sneighbor for which exists the transition from those ones to Qj multiplied by the constant 6;

4. Compute *AN* (1) characteristics for S_{group}.

   When this value is grater than the "border value" (the input parameter of this algorithm, in our experiments is usually 0.7) the state *Qj* becomes the real member of S_{group}. Now continue by step 3. When the ratio is less than the "border value", state *Qj* is discarded from the S_{group} and this set is closed. Now continue by step 2.

5. If all internal states are placed into sets S_i and at the same time S_0 is empty, construct the internal states code:

   It is connected both from the binary part (a serial number of the state in its set in binary notation) and the one-hot part (a serial number of a set in one-hot notation). The number of binary part bits is equal to *b,* where 2^b is greater or equal to the maximum number of states in sets. The number of one-hot part bits is equal to the number of sets S_i.




```
.i 2
.o 1
.p 11
.s 4
-0 st0 st0 0
11 st0 st0 0
01 st0 st1 -
0- st1 st1 1
11 st1 st0 0
10 st1 st2 1
1- st2 st2 1
00 st2 st1 1
01 st2 st3 1
0- st3 st3 1
11 st3 st2 1
```

Fig. 6. STG and the kiss2 format of the lion benchmark

Example (*lion* benchmark (Web-MCNC Benchmarks) border ratio 0.7, Fig. 6):

1. Place all FSM internal states Q_i to the set S_0

$$S_0 = \{st0, st1, st2, st3\}$$

2. For all S_0 elements compute the number of transitions to next disjoint states from S_0:

$$(st0\ldots1, st1\ldots2, st2\ldots2, st3\ldots1).$$

Choose the state with the highest value and construct the new set S_1:

$$S_0 = \{st0, st2, st3\},\ S_1 = \{st1\}$$

3. Construct the set of neighboring internal states of all members of $S_1 - S_{neighbor}$:

$$S_0 = \{st0, st2, st3\},\ S_1 = \{st1\},\ S_{neighbor} = \{st0, st2\}$$

Compute the score for all states from $S_{neighbor}$:

$$st0_{score} = 1.10+1.20+2.3+1.6+1.10+1.20+2.3+1.6 = 84$$

$$st2_{score} = 1.10+1.20+1.3+1.6+1.10+1.20+1.3+1.6 = 78$$

Choose the state with the highest score and add it to S_1:

$$S_0 = \{st2, st3\}, S_1 = \{st0, st1\}, S_{neighbor} = \{st2\}$$

4. Compute *AN* (1) for the elements from S_1:

$$AN = 1.0.$$

AN is grater then 0.7, therefore the state *Qj* becomes a real member of S_1. Now continue by step 3.

3. Try to add the state *st2* into S_1 and compute the *AN*. Because $AN = 0.66$, state *st2* is discarded from the S_1 and this set is closed. Now continue by step 2.

At the end all internal states are placed into 2 groups:

$$S_1 = \{st0, st1\}, S_2 = \{st2, st3\}$$

Now internal state code is connected from the one bit binary part and the two bits one-hot parts:

st0 … 0/01 st1 … 1/01 st2 … 0/10 st3 … 1/10

4 Experiments

4.1 Information about the FSM benchmarks

The conversion program between KISS2 format and VHDL was necessary to build (because the used benchmarks were in KISS2 format, see Fig. 6, where the *lion* benchmark is described by STG and KISS2, too). We have implemented the converter *K2V_DOS* (in *C++* by compiler *GCC* for *DOS OS*). The *K2V_DOS* program allows an acquisition of information about the FSM like e.g.: the number of states (#State), number of inputs (#In), number of outputs (#Out), number of transitions - arcs (#Tran), node degree, etc., some of them are in Tab. 2.

Tab. 2. Our benchmark set and the most important benchmark properties

Name	#State	#In	#Out	#Tran	AN
Bbara	10	4	2	60	0,300
Bbara2	128	4	2	852	0,020
Bbsse	16	7	7	56	0,146
Bbtas	6	2	2	24	0,267
Beecount	7	3	4	28	0,381
Cse	16	7	7	91	0,163
dk14	7	3	5	56	0,572
dk15	4	3	5	32	0,750
dk16	27	2	3	108	0,141

dk17	8	2	3	32	0,375
dk27	7	1	2	14	0,310
dk512	15	1	3	30	0,143
Donfile	24	2	1	96	0,130
ex1	20	9	19	138	0,150
ex2	19	2	2	72	0,164
ex3	10	2	2	36	0,311
ex4	14	6	9	21	0,088
ex6	8	5	8	34	0,429
ex7	10	2	2	36	0,267
Keyb	19	7	2	170	0,132
Lion	4	2	1	11	0,500
lion9	9	2	1	25	0,222
Mark1	15	5	16	36	0,167
Mc	4	3	5	10	0,333
modulo12	12	1	1	24	0,091
opus	10	5	6	30	0,267
planet	48	7	19	115	0,031
pma	24	8	8	73	0,087
s1	20	8	6	107	0,179
s1488	48	8	19	251	0,051

s1494	48	8	19	250	0,051
s1a	20	8	6	107	0,179
s208	18	11	2	153	0,111
s27	6	4	1	34	0,633
s298	218	3	6	1096	0,023
s386	13	7	7	64	0,205
s420	18	19	2	137	0,111
s510	47	19	7	77	0,024
s8	5	4	1	20	0,400
s820	25	18	19	232	0,142
s832	25	18	19	245	0,142
sand	32	11	9	184	0,060
scf	121	27	56	286	0,019
shiftreg	8	1	1	16	0,250
sse	16	7	7	56	0,146
styr	30	9	10	166	0,084
tav	4	4	4	49	0,333
tma	20	7	6	44	0,100
train11	11	2	1	25	0,127
train4	4	2	1	14	0,333

4.2 The way of VHDL description

The FSM can be described by different ways in VHDL (with different results), [11]:

- **one** big **process** sensitive to the clock signal and to the input signals (one *case* statement is used in this process - it selects active state and in each branch of the *case* there are *if* statements, which define next states and outputs - this is the same method, like the *XILINX FOUNDATION* uses for conversion between STG and *VHDL,* Fig. 7);
- **three processes** (*next-state-proc* for implementation of the next-state function, *state-dff-proc* for asynchronous reset and using D flip-flops and *output-proc* for the FSM output function realization).

Our *K2V_DOS* program can generate both methods. The ratio of using one VHDL process and three VHDL processes for all coding methods and all benchmarks in % is in Tab. 3. The remarkable result is that the three processes are about 1.5 better then one process in all cases.

Tab. 3. #CLB for one process / #CLB for three processes in %

	Binary	Gray	Johnson	one-hot	fanin	fanout	Fan
Average	156,9	155,5	153,7	157,1	115,5	164,5	149,6

```
process ( clk )
begin
  if clk'event and clk = '1' then
  case state is
    when state_definition =>
      if transition_condition then

        -- next_state_setting;
           state <= next_state;
        -- output_setting;
           output <= "0000";

      elsif

      {next_conditions}

      endif;

    {next_states}

    when others => null;
  end case;
end process;
```

Fig. 7. VHDL description of the FSM by one process

To overcome the *XILINX FOUNDATION* optimization for the "one-hot" coding method [32], we have used **direct code assignment** instead of the **enumerating type**. The difference can be seen in Tab. 4 and Fig. 8. Due to the better results (less number of CLB blocks) for enumerating type for all methods next we will always use this one.

Tab. 4 #CLB for direct code assignment / #CLB for enumerating type in %

	binary	Gray	Johnson	one-hot	two-hots	fanin	fanout	fan	FEL 0,7
Avarage	120,9	126,9	153,5	197,2	169,9	124,9	126,0	126,4	180,5

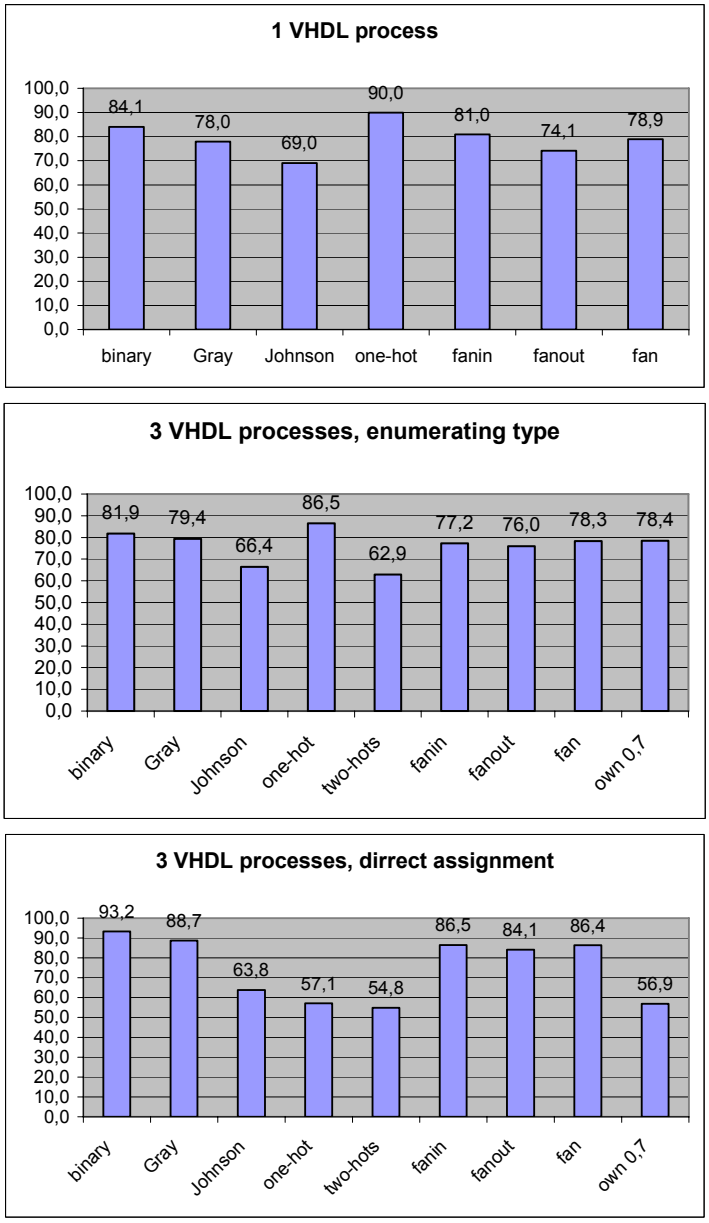


Fig. 8. $IU_{\#CLB}$ for different styles of descriptions

4.3 Special testing FSMs

The K2V_DOS program system can generate our special testing FSM (for more precise setting of the "border ratio" AN). We have generated the Moore type FSM with the determined number of internal states and mainly the determined number of the transitions from the internal states. Our FSM has the STG with the strictly defined and the same number of transitions from all states. The resulting format is the KISS2 format – e.g. 4.kiss testing FSM has the STG with four edges from every internal state (node). Both the first and also the next state connections are generated randomly to overcome the XILINX FOUNDATION optimization for the counter design.

The choice of the AN=0.7 (axis x) can be seen of the Fig. 9a) and b). There the relationship between "one-hot" and binary coding methods with respect to the number of internal states and the number of transitions from all states is represented. It can be seen that for greater branching (the greater AN) the binary code is better. The border is 0.7. The Y axis expresses the percentage success of the methods (the minimal number of CLB from all methods for every benchmark is 100%).

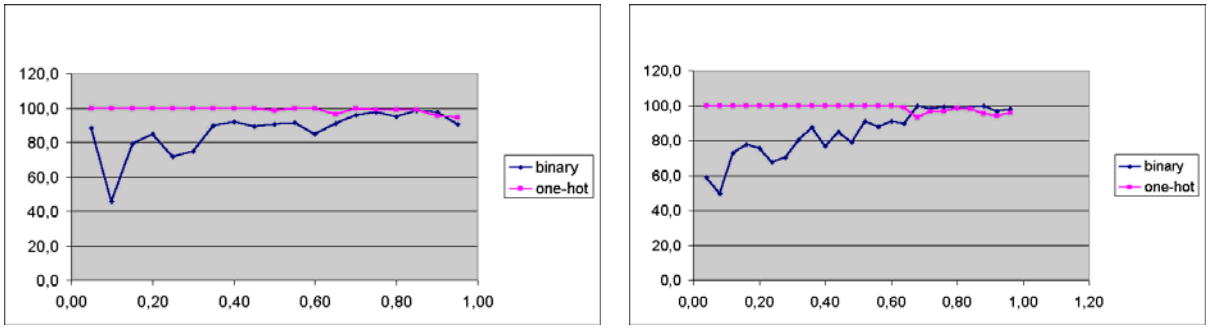


Fig. 9. a) 20 state FSM and b) 25 states FSM

4.4 FSM internal state coding experiments

Our experiments needed the special software tools - programs for generating different internal states codes, special testing circuits with predefined properties, special batches for easier performing of them by the EDA system with predefined choices from menu and statistical processing of the results. The results were compared and processed from the different points of view, under the different criteria:

- number of CLB blocks - # CLB
- maximal working frequency – max(*f*)

- IU - **i**ndex of s**u**ccess with respect to the space (# CLB) and speed (max(*f*)):

$$IU_{\#CLB} = \frac{\min(\#CLB)}{\#CLB} * 100\%$$

$$IU_{f} = \frac{f}{\max(f)} * 100\%$$

- number of the best solutions *BS*- the minimal number of CLB blocks (or the maximal working frequency) for the best coding methods for every benchmark is 100% and the resulted value presented in graphs is average of this percentage values, Figs. 10 - 13.

The K2V_DOS program can generate different FSM internal state coding by methods minimum-length (binary), Gray, Johnson, "one-hot", "two-hots", Fan-in, Fan-out and FAN and "FEL-code". To overcome the impact of the way of VHDL descriptions we have presented the results for one VHDL process, for three processes with direct assignment and enumerating type in the relative values, see Fig. 9.

The comparison for three processes and enumerating type (number of the best solutions BS for every method) is in Figs. 10 and 11. From all graphs can be seen that the most important and mainly the most successful are binary or "one-hot" methods. Only when we have used direct code assignment FANIN methods can give relative good results (but because the FOUNDATION outputs optimization is not used). The most successful methods are "one-hot" (53%) and binary (13%), see Fig 10. According these remarks we have proposed, presented and implemented our own method "FEL-code" (Sec. 3.2). The comparison for three VHDL processes, enumerating type and eight coding methods is in Fig. 11, where the best is "one-hot" (43%), second is binary (13%) and third "FEL-code" (10%).

The *K2V_DOS* program system can generate our special testing FSMs for more precise setting of the characteristics *AN*, Definition 2. The *K2V_DOS* program can generate different FSM internal state encoding by "minimum-lengths", "Gray", "Johnson", "one-hot", "two-hots", "Fan-in", "Fan-out", "FAN" and "FEL-code" methods.

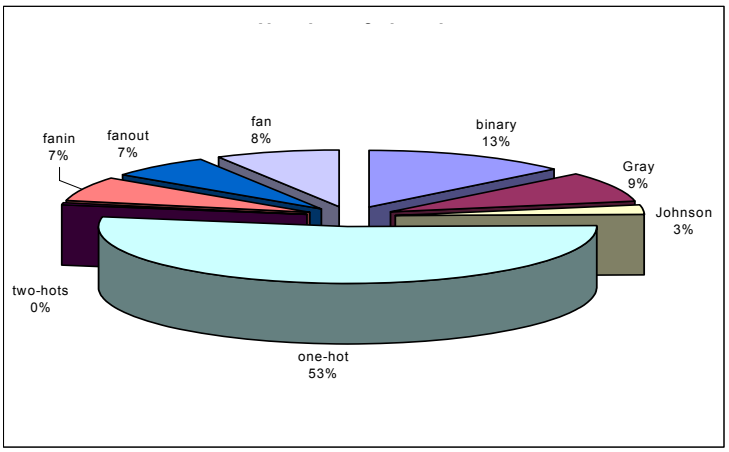


Fig. 10. *BS* for three VHDL processes, enumerating type

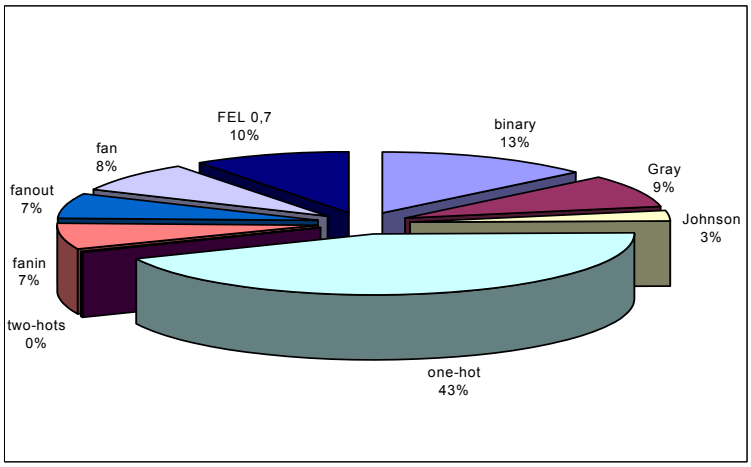


Fig. 11. BS for three VHDL processes, enumerating type with FEL-code

All results have been compared under the different criteria and with respect to benchmark properties. Mostly the most important benchmark property is its size, i.e. the number of its inputs, outputs and internals states. The relative number of the best solutions for the FSMs with few internal states is in Fig. 12 and for many internal states are in Fig. 13. From those Figures it is clear, that the minimum-lengths (here binary) is better to use for relatively little circuits and one-hot for greater ones. The method FEL-code is more successful for FSMs with more internal states.

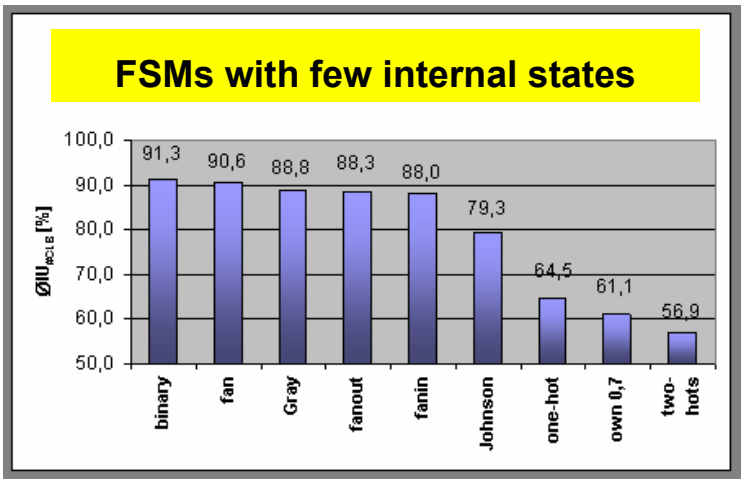


Fig. 12. **BS for three VHDL processes, enumerating type with FEL-code (AN = 0,7) for benchmarks with few internal states**

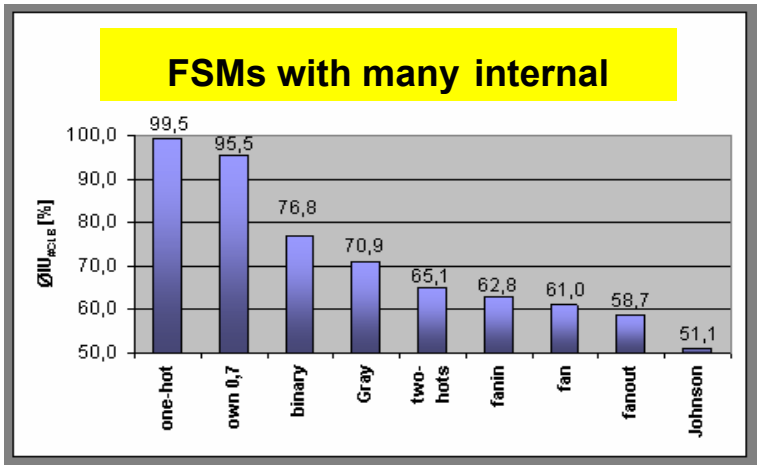


Fig. 13. **BS for three VHDL processes, enumerating type with FEL-code (AN = 0,7) for benchmarks with many internal states**

5 Conclusions

A new method for coding internal FSM states was described. This method is called FEL-code (according to the **F**aculty of **EL**ectrical Engineering), and it gives relatively good results with respect to the FPGA implementation. The results are verified by experimental implementations in real commercial products – FPGAs from Xilinx and Atmel.

We have performed about 2000 experiments with different types of encoding and decomposition methods for 50 benchmarks. The comparison of nine encoding methods is shown in Fig. 14. There is condensed information

about the average "success" – IU - for all encoding methods. The minimum number of CLB blocks for a benchmark is divided by the number of CLB blocks for a particular encoding method. Similarly, the working frequency for a particular encoding method is divided by the maximum frequency for a benchmark. These values were computed for all benchmarks for maximum working frequency (dark columns) and number of CLB blocks (#CLB, white columns) and expressed as percentage "success" for each encoding method.

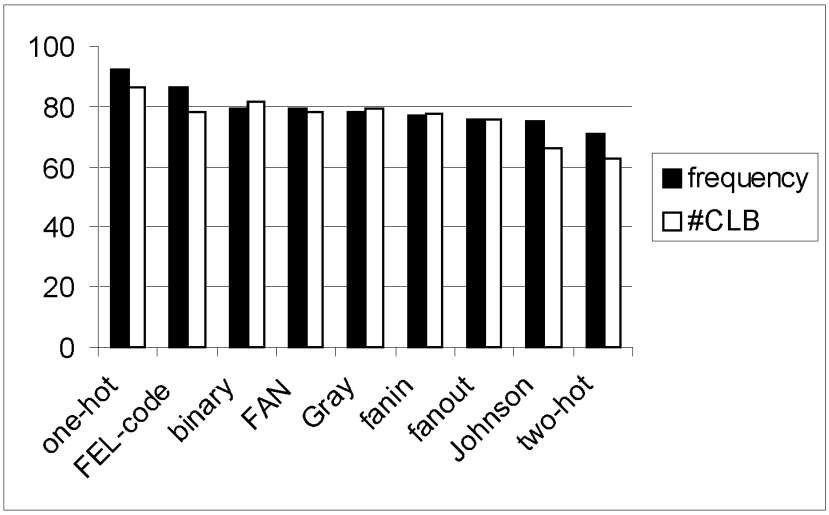


Fig. 14. The final comparison of all used encoding methods with respect to IU

We can arrive at the following conclusions based on our encoding and decomposition experiments:

- the „minimum-length" encoding method gives the best results for FSM with few internal states (5) and for FSM with AN > 0.7 (the state transition graph with many cycles)
- "one-hot" encoding method is better for other cases and mostly generates the faster circuits (but the XILINX FOUNDATION uses optimization methods for "one-hot" encoding)
- "FEL-code" method is universal as combines the advantages of both "one-hot" and „minimum-length" methods. This method is heuristic, the parameters (AN and the score evaluations) have been experimentally verified
- other tested encoding methods give worse results in most cases and have no practical signification
- for such FSM implementation, where the majority of the CLB blocks are used (e.g. 90%) the "one-hot" methods gives better results mainly with respect to the maximum working frequency due to easier wiring

- a different strategy for searching for the partitions – the best FSM partition is not that one with the minimum number of internal states but that one with the minimum sets of input and output symbols – could be used for FPGA implementation.

These conclusions were based on experiments on XILINX FOUNDATION performed and published 2 years ago in [19] – [24]. The experimental results performed on the recent XILINX ISE EDA tool have not be sufficiently compared with those presented above since many qualitative changes were incorporated into this tool – new types of the final platforms and new design algorithms were incorporated into the EDA system. According our recent results we can conclude that the most successful methods still remains "one-hot" and „minimum-lengths". The average improvement (for all benchmarks but only for working frequency) is presented in Tab. 5. It can be stated that the encoding methods offered by the professional EDA system are better then the outside methods and the AN equals to 0.7 is a rightful value.

Future work needs deeper knowledge about the design processes performed by professional EDA design tools, experiments with different types of internal states coding methods closer to the real regular structure which the designed circuits is implemented into, and decompositions with respect to these structures [17]. Also the digital circuit must be designed with respect to its testing, external or mostly internal (BIST structures), with respect not only to the space and working frequency but also to its dependability and power consumption [26]. All these aspects must be taken account and accomplished to the real and future design processes, which must enable formal verifications at all levels of design process.

Tab. 5. Frequency improvement

Encoding method	Average maximum frequency [MHz]		Frequency increase [%]
	Spartan	Spartan-II	
binary	76,6	145,0	47,2
one hot	80,5	167,1	51,8
FEL-c 0,5	71,3	108,7	34,4
FEL-c 0,7	75,6	128,1	41,0

Classical methodology for digital circuit design is well-developed today. Both the implementation and the verification processes are well-documented, [2], [9], [10]. Nevertheless, the basic minimization methods improvements are actual as well, mainly with respect not only to the gate equivalents but to the

other parameters which must be fulfilled especially when the regular structures are used as the final implementation bases. Our results and conclusions will lead to the shorter design time and the design tools improvements. The work on these experiments goes on and has to continue. Most important seems to be the interconnection of all levels of the digital design process with respect to the different design constraints depending on the final implementation platforms.

References

[1] Ashar, P., Devadas, F., Newton, A.R.: „Sequential Logic Synthesis." Kluwer Academic Publishers, Boston/Dordrecht/London, 1992, 225 pp.

[2] Bergeron, J.: „Writing Testbenches: Functional Verification of HDL Models." Kluwer Academic Publishers, Boston/Dordrecht/London, 2000, 354 pp.

[3] Brayton, R. K. et al.: „Logic minimization algorithms for VLSI synthesis." Kluwer Academic Publishers, Boston, MA 1984, 192 pp.

[4] Cassandras, C. G., Lafortune, S.: „Introduction to Discrete Event Systems." Kluwer Academic Publishers, 1999, 822 pp.

[5] Chojnacki, A., Jozwiak, L.: „An Effective and Efficient Method for Functional Decomposition of Boolean Functions based on Information Relationship Measures." Proceedings 3rd DDECS 2000 Workshop, Smolenice, Slovakia, pp. 242 – 249.

[6] Drechsler, R., Becker, B.: „Binary Decision Diagrams, Theory and Implementation." Kluwer Academic Publishers, Boston, 1998, 200 pp.

[7] Feske, K., Mulka, S., Koegst, M. Elst, G.: „Technology-Driven FSM Partitioning for Synthesis of Large Sequential Circuits Targeting Lookup-Table Based FPGAs." Proceedings 7th Workshop Field-Programable Logic and Applications (FPL '97) London, UK, LNCS 1304, 1997, pp. 235 – 244.

[8] Gajski, D. D.: „Principles of Digital Design." Prentice-Hall International, Inc., USA, 1997, 447 pp.

[9] Hachtel, G. D., Somenzi, F.: „Logic synthesis and verification algorithms." Kluwer Academic Publishers, Boston, MA 1996, 564 pp.

[10] Hassoun, S., Sasao, T.: „Logic Synthesis and Verification." Kluwer Academic Publishers, Boston, MA, 2002, 454 pp.

[11] „IEEE Standard for VHDL Register Transfer Level (RTL) Synthesis." IEEE Computer Society, 1076.6 - 1999 edition.

[12] Jozwiak, L.: „Simultaneous decomposition of sequential machines." Microprocessing and Microprogramming 30, 1990, pp. 305 – 312.

[13] Józwiak, L., Kolsteren, J. C.: „An Efficient Method for Sequential General Decomposition of Sequential Machines." Microprocessing and Microprogramming 32, 1991, pp. 657 – 664.

[14] Jozwiak, L.: „An Efficient Heuristic Method for State Assignment of Large Sequential Machines." Journal of Circuits, Systems and Computers, Vol. 2, No. 1, 1991, pp. 1 – 26.

[15] Jozwiak, L.: „General decomposition and its use in digital circuit synthesis." VLSI Design, 3 (3), 1995, pp. 225 – 248.

[16] Jozwiak, L., Chojnacki, A.: „Effective and efficient FPGA synthesis through general functional decomposition." Journal of Systems Architecture, Vol. 49, Issue 4-6, September 2003, pp. 247 – 265.

[17] Józwiak, L., Slusarczyk, A.: „General decomposition of incompletely specified sequential machines with multi-state behavior realization." Journal of Systems Architecture, Vol. 50, No 1, January 2004, pp. 445-492.

[18] Kubátová, H.: „Finite State Machine Implementation in FPGAs." Design of embedded control systeme. Editors - Adamski, Karatkevich and Wegrzyn, Kluwer Academic Publisher (Springer Verlag), 2005, pp. 177 – 188.

[19] Kubátová, H.: „How to obtain better implementation of the FSM in FPGA." Proceedings of the 5th IEEE Workshop DDECS 2002, Brno, Czech Republic 2002, pp. 332 -335.

[20] Kubátová H., Bečvář M.: „FEL-Code: FSM Internal State Enncoding Method." In: 5th International Workshop on Boolean Problems. Freiberg: Technische Universität Bergakademie, 2002, pp. 109-114.

[21] Kubátová, H.: „Implementation of the FSM into FPGA." Proceedings of the International Workshop on Discrete-Event System Design DESDes '01, Oficyna Wydawnicza Politechnika Zielona Góra, Przytok, Poland 2001, pp. 141 -146.

[22] Kubátová H.: „Optimal Implementation of FSM in FPGA." Proceedings of the Computer Science Education Workshop 2001. Košice: Department of Computers and Informatics of FEI, Technical University Košice, 2001, pp. 107-110.

[23] Kubátová H.: „Hardware Implementation and Simulation of Petri Nets.“ Proceedings of the Computer Science Education Workshop 2001, Košice, Department of Computers and Informatics of FEI, Technical University Košice, pp. 103-106.

[24] Kubátová, H., Hrdý, T., Prokeš, M.: „Problems with the Encoding of the FSM with the Relation to its Implementation by FPGA.“ ECI2000, Electronic Computers and Informatics, International Scientific Conference, Herl'any 2000, pp. 183 – 188.

[25] McCluskey, E.J.: „Minimization of Boolean functions“ The Bell System Technical Journal, 35, No. 5, Nov. 1956, pp. 1417-1444

[26] Mitra, S., McCluskey, E. J.: „Diversity Techniques for Concurrent Error Detection.” Center for Reliable Computing, Dept. of Electrical Engineering and Computer Science Stanford University, Technical Report 00-7, 2000.

[27] Mitra, S., Huang, W., Saxena, N.R., Yu, S.Y., McCluskey, E.J.: „Reconfigurable Architecture for Autonomous Self-Repair.” IEEE Design and Test, May/June 2004, Vol. 21, No. 3, pp. 228-240.

[28] Pradhan, D. K.: „Fault-tolerant computer system design.“ Prentice Hall ptr, New Jersey, 1995, 550 pp.

[29] Ross, P.E.: „5 Commandments.“ IEEE Spektrum, December 2003, Vol. 40, No.12, pp. 30 – 35.

[30] Shih-Chieh Chang: „Layout Driven Logic Synthesis and Optimization Techniques for FPGA.“ PhD thesis, University of California, Santa Barbara, 1994, 119 pp.

Used web pages:

[31] Web-MCNC Benchmarks: ftp://ftp.mcnc.org/pub/benchmark/Benchmark.dirs/LGSynth93/LGSynth93.tar, testcases/pla/ or MCNC - ftp://ic.eecs.berkeley.edu

[32] Web-The Product Data Sheets: Xilinx, Inc. http://www.xilinx.com/partinfo.

[33] Web-SIS (1996): Logic Optimisation and Synthesis: http://eent3.sbu.ac.uk/units/ade/sis/sis00.htm

[34] Web-SIS (2002): Bengtsson, T., Kumar, S.: „Logic optimization using SIS.” Version 2002: http://hem.hj.se/~beto/courses/logic_optimization/Lab1_manual.pdf

Ing. Hana Kubátová, CSc.

Po maturitě v roce 1975 na gymnáziu Arabská v Praze 6 (se změřením na programování) studovala obor technická kybernetika na Českém vysokém učení technickém v Praze, fakultě elektrotechnické (v diferencované formě), které ukončila s vyznamenáním v roce 1980.

V letech 1980 – 81 byla na katedře počítačů FEL ČVUT v Praze na stáži a poté v interním aspirantském studiu. V roce 1983 nastoupil do pracovního poměru na FEL ČVUT jako odborná asistentka na katedře počítačů do skupiny zabývající se počítačovým hardwarem, kde působí dodnes.

V roce 1986 obhájila kandidátskou disertační práci a získala titul CSc. v oboru Elektronické počítače (kandidátská práce: „Modifikované Petriho sítě“

Své pedagogické působení na FEL ČVUT zahájila již jako stážistka v roce 1981, kdy vedla cvičení předmětu *Logické systémy* (tehdy na oborech Výpočetní technika i Kybernetika) a pomáhala zavádět laboratorní výuku do těchto předmětů. Jako odborná asistentka vedla cvičení z předmětů Diagnostika a spolehlivost a Konstrukce počítačů.

Po návratu z mateřské dovolené se podílela na cvičeních z téměř všech hardwarových předmětu (Aritmetika, Architektura počítačů, Problémy a algoritmy, Realizace počítačových systémů, aj.) i ze základního softwarového předmětu (Výpočetní technika a programování pro všechny 1. ročníky FEL).

Od roku 1996 přednáší předmět Úvod do počítačových systémů (v základním studiu). Od roku 2002 převzala předmět Diagnostika a spolehlivost (v magisterském studiu) po prof. Hlavičkovi včetně jeho rozšířené formy pro doktorandské studium, kde nyní participuje s prof. Novákem.

Ve strukturovaném studiu zavedla nový předmět Logické obvody s moderní organizací cvičení – praktický návrh pro hradlová pole s využitím profesionálního návrhového systému (XILINX ISE) a do značné míry inovovaný předmět Úvod do počítačových systémů pro obor Sdělovací technika. Vedu a vedla jsem i konzultace a soustředění pro kombinované studium v několika předmětech.

Během svého působení na katedře vedla a vede řadu diplomových a bakalářských prací a semestrálních projektů.

Pravidelně se zúčastňuje každoročně pořádaného setkání zástupců kateder počítačů z českých a slovenských universit (Computer Science Education Workshop - CSEW), v roce 2000 toto setkání organizovala a připravovala sborník.

Od počátku svého působení na katedře počítačů se aktivně zúčastňuje vědeckých konferencí a podílí se na jejich organizaci. Zúčastnila se projektu mezinárodní spolupráce s Varšavskou polytechnikou (v 80. letech na téma číslicový návrh, modelování a diagnostika) a několika vzájemných setkání, z nichž některé organizovala. Byla zakládající členkou organizačního a programového výboru mezinárodní konference Mikrosystém, která se konala každoročně v letech 1983 – 1989, vždy jednou na Slovensku a jednou v Česku. Byla členkou organizačního výboru konference EUROMICRO 1996, pořádané v Praze a předsedkyní organizačního výboru konference Dependable Computing EDCC-3 pořádané v Praze v roce 1999. V současné době jsem členem programového a organizačního výboru konference DSD – EUROMICRO (2004, 2005) a byla zvolena do rady direktorů vědecké společnosti EUROMICRO jako člen korespondent a jediný zástupce za ČR. V současnosti připravuje IEEE Workshop DDECS (jako předsedkyně organizačního výboru), který se bude konat v Masarykově koleji v roce 2006.

V roce 2002 schválila vědecká rada ČVUT FEL jmenování Ing. Kubátové do funkce školitele pro studijní obor doktorského studia *26-xx-x Informatika a výpočetní technika*. V současné době je Ing. Kouba školitelem čtyř interních a jednoho externího doktoranda. Od roku 2004 je Ing. Kubátová členkou ORO pro stejný obor a je rovněž zvána jako členka komise pro obhajobu doktorských disertačních prací na pracoviště školící doktorandy v oborech týkající se číslicového návrhu, zejména na TU Liberec, VUT Brno a TU Košice.

Hlavním odborným zájmem Ing. Kubátové je problematika číslicového návrhu, a to jak formálního založeného na použití formálních modelů (Petriho sítích, spolehlivostních modelů), tak konkrétních realizací hlavně v programovatelných strukturách použitelných ve výuce (FPGA), a to včetně testování. Již v 80. letech v rámci VHČ se podílela na návrhu a oživení přístroje pro simulaci a testování leteckých signálů pro sběrnici C-ARINC 429 (pro VZLÚ Letňany) a na psaní testů pro číslicové desky vyráběné v ZPA Čakovice.

Byla navrhovatelkou a spoluřešitelkou několika projektů GAČR, FRVŠ a Interní grantové agentury ČVUT.

Již v 80. letech publikovala na konferencích (Mikrosystém 1983, 1984, MOP – Moderní Programování 1986, Digital Airborne Systeme 1988), v časopise Automatizace (1983,26/6 a 1984, 27/3), v rámci kurzu ČSVTS (Methods of Specification and Evaluation of Microprocessor Systems Design, 1986) na témata týkající se formálního návrhu, simulace, Petriho sítí a návrhu číslicových systémů.

V letech 1988 – 1991 přerušila svoji profesní kariéru z důvodů mateřské dovolené. Následné starosti o děti a rodinu, syn (1978) a dcera (1988), např. nutnost pracovat pouze na částečný zaměstnanecký úvazek a faktická změna

poměrů nejen na škole, ale i v odborné praxi, měly za následek nutnost nového nastartování odborné a profesní kariéry. Proto se aktivity Ing. Kubátové začaly více rozvíjet až v roce 1995, kdy se opět mohla aktivně zúčastňovat konferencí (MOSIS, ASIS, ECI, DESDes, DDECS, DSD) a podílet se na jejich organizaci.

Jako autor/spoluautor publikovala v posledních deseti letech své výsledky v jedné kapitole v mezinárodní monografii, ve 25 příspěvcích na mezinárodních a 15 příspěvcích na tuzemských konferencích. Její nejvýznamnější publikace jsou:

[A1] Kubátová, H.: *Finite State Machine Implementation in FPGAs.* In: Design of Embedded Control System. Editors - M. Adamski, A. Karatkevich, M. Wegrzyn, Kluwer Academic Publisher (Springer Verlag), May 2005, ISBN 0-387-23630-9.

[A2] Fišer P., Kubátová H.: *Boolean Minimizer FC-Min: Coverage Finding Process*. In: EUROMICRO Symposium on Digital System Design. Piscataway: IEEE, 2004, p. 152-159. ISBN 0-7695-2203-3.

[A3] Dobiáš R., Kubátová H.: *FPGA Based Design of Raiway's Interlocking Equipment.* In EUROMICRO Symposium on Digital System Design. Piscataway: IEEE, 2004, p. 467-473. ISBN 0-7695-2203-3.

[A4] Kubátová H.: *Direct Implementation of Petri Net Based model in FPGA.* In: Proceedings of the International Workshop on Discrete-Event System Design - DESDes'04. Zielona Gora: University of Zielona Gora, 2004, p. 31-36. ISBN 83-89712-15-6.

[A5] Kubátová H., Jelinek R.: *Digital Testing and Reliability Education (DTRE) Computer Tool.* In: The 9th IEEE International Conference on Electronics, Circuits and Systems. Piscataway: IEEE, 2002, s. 1227-1230. ISBN 0-7803-7596-3, CD-ROM: ISBN 0-7803-7597-1.