SINMEF - A Decomposition Based Synthesis Tool for Large FSMs

Carlos Humberto Llanos Quintero* and Marius Strum**
(*)Departamento de Computação, Universidade de Brasilia
(**) LME, Laboratório de Microeletrônica, Universidade de São Paulo
llanos@guarany.unb.br, strum@lme.usp.br

Abstract

This paper describes the SINMEF environment, composed of the DECMEF and the SIS [9] systems, used to synthesize large finite state machines (FSMs). The DECMEF system consists of a set of tools to decompose a FSM into a set of cooperating sub-FSMs. An efficient cost function is used to guide the decomposition process. The decomposed FSMs are state encoded and further optimized and technology mapped using tools from the SIS system. Results obtained for FSMs with more than 1000 states showed an improvement of as much as 60.42% in critical path and 14.79% in area. Preliminary results show that the recursive use of the decomposition system extends its application to FSMs with several thousands of states.

Keywords: *FSM, decomposition, non-deterministic transitions, redundant transitions, clustering technique.*

1 Introduction

Large sequential circuits, modeled as FSMs containing hundreds of states, are commonly mapped onto a pre-defined ROM based (micro-programmed) architecture due to: 1) the inability of FSM synthesis tools to obtain efficient solutions for large problems [8] and, 2) the inability of FSM synthesis tools to obtain solutions at all. FSM decomposition techniques permit the designer to split the original FSM into a set of smaller interacting submachines (sub-FSMs). This strategy is desirable for several reasons: 1) the set of smaller machines leads to an improved circuit performance due to critical path reduction and better floorplanning; 2) state assignment tools work better for small FSMs than for large ones [13]; 3) decomposition techniques allow logic synthesis systems to find solutions for large FSMs otherwise unsolved [1].

This paper presents the SINMEF FSM synthesis environment which allows users to synthesize FSMs containing up to a few thousand of states, initially represented by state transition tables (KISS format). SINMEF is composed of 2 parts. The first consists of a set of tools (DECMEF) that decomposes the original FSM into two (or more) sub-FSMs, each represented by a partition of the original set of states. The resulting KISS description of each sub-FSM is further optimized and technology mapped using the SIS system [9].

Several FSM decomposition methods have been proposed in [2],[4],[11] and [12]. However they only present results for medium size FSMs (maximum of 121 states in [1]). Our DECMEF system allows to obtain decompositions for FSMs with more than 1000 states. It consists of two parts. The **GERPAR** tool generates two or more state partitions (π_1, π_2, ... π_n) using a state clustering heuristics. The GERTAB tool removes the redundant transitions and encodes the auxiliary input states required to eliminate the non-deterministic transitions resulting from the decomposition process.

2 The GERPAR Tool

The GERPAR tool performs a state clustering technique in order to determine two or more partitions required to decompose a FSM. This technique **groups** the original states into sets of clusters (blocks), also called **super-states** [3]. The product among these partitions is constrained to be the *zero partition* [1].

The procedure starts from a pair of "zero partitions". The N states containing the largest number of fan-out transitions are chosen to become the "seeds" of each cluster in each partition (N is user defined). All other states are then moved into these clusters obeying the *zero partition* constraint and satisfying a cost function. We call this procedure **decomposition by clustering**. The cost function used to guide the movement tries to optimize two components:

- Minimize the number of non-deterministic transitions in order to reduce the communication among the sub-FSMs.
- Maximize the number of redundant transitions [6], [7] in order to reduce the area of the decomposed FSM.

As these can be conflicting objectives, two **priority parameters** (OPT1 and OPT2 respectively) allow the user to search for an optimal solution. We found that each parameter may lead to better results for different problem.

The whole procedure to obtain two partitions consists of the following steps:

1) Build a pair of zero partitions containing all the original states. An alternative is to start from a pair of partitions resulting from a factorization technique [7].
2) Sort the states according to the number of their fanout transitions.
3) Choose the N states with largest fanouts as the "seeds" of the super-states.
4) Select the unused state with largest fan-out.
5) Find the target super-state in the first partition that optimizes the cost function.
6) Move the selected state to the target super-state.
7) Define the set of possible target super-states in the second partition, checking the zero-product property.
8) If the result of step 7 is not empty, find the target super-state in the second partition that optimizes the cost function.
9) Repeat steps 3 through 7 until no more states can be moved.

States that present an empty solution for step 7 are left out of the super-states. This procedure produces two state partitions of the original FSM. A recursive use of this procedure may lead to more than two partitions.

3 The GERTAB Tool

At this point of the synthesis process, each sub-FSM, described by one of the state partitions, may contain redundant and non-deterministic transitions that resulted from the partitioning procedure [7]. The GERTAB tool removes the redundant transitions and solves all the indeterminations generating the transition tables of each sub-FSM. The architecture of the resulting decomposed FSM (figure 4.1) shows that the communication between the sub-FSMs is resolved using all the super-states of one sub-FSM as **auxiliary present states** (APS) of the other sub-FSM.

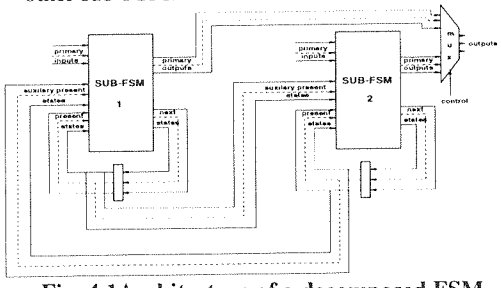

Fig. 4.1Architecture of a decomposed FSM

The advantage of this procedure in comparison with the elimination of non-deterministic transitions by extra auxiliary output states [7] is the reduction of the number of output functions that will be generated by the logic optimization tool that follows.

The input overhead in sub-FSM2 due to the auxiliary present states (APS) is given by: $n_2 = log_2$ (*number of super-states of sub-FSM1*) and vice versa. The encoding of these auxiliary states is solved by initially assigning arbitrary codes to the APS of sub-FSM1 which are interpreted as (extra) primary inputs of this sub-FSM. Now the state encoding tools from SIS, NOVA [10] or JEDI [5], may be used to encode the states of sub-FSM1 and, next, the states of sub-FSM2. The latter codes are used to replace the initial (arbitrary) codes. The whole procedure consists of the following steps:

1) Read the KISS format of the original FSM.
2) Read the partitions file.
3) Eliminate the redundant transitions in sub-FSM1 and sub-FSM2 [7].
4) Determine the non-deterministic transitions in sub-FSM1 and sub-FSM2.
5) Derive the auxiliary present states for each sub-FSM.
6) Arbitrarily encode all auxiliary present states of sub-FSM1.
7) Run a state assignment tool (NOVA or JEDI) for sub-FSM1.
8) Run the same state assignment tool for sub-FSM2 (the auxiliary present states of sub-FSM2 were obtained in step 7).
9) Replace the random codes assigned to the auxiliary present states (step 6) with the codes found in step 8.
10) Generate the KISS description for each sub-FSM.

The decomposed FSM outputs are obtained either by **output splitting**, which means assigning half of the outputs to each sub-FSM or by **output multiplexing** which means activating each sub-FSM for half of the outputs (figure 4.1) [6],[7].

4 Experimental results

The decomposed FSMs resulting from DECMEF were optimized and technology mapped by the SIS system. The state count of the FSM examples shown in Table 1 ranged between 32 and 1098. The *blanca, monica* and *aida* examples were generated using a random FSM generator [7]. All the other examples are known benchmarks.

Table 2 shows the SIS synthesis results for the **non-decomposed** FSMs. The *lib2.genlib* library was used for these analysis.

Table 1: FSM characteristics

name	states	in-puts	out-puts	transi-tions
sand	32	11	9	184
planet	48	7	19	115
scf	121	27	56	166
controller	353	2	4	1400
blanca++	600	2	3	1203
mónica++	700	5	4	1050
aida++	1000	4	2	1500
master-read	1098	13	7	4989

Table 2: Synthesis results for the non-decomposed FSM

name	number of gates	critical path (**)
sand	473	38.81
planet	540	50.68
scf	797	80.58
controller	1547	128.35
blanca++	*	*
mónica++	*	*
aida++	*	*
master-read	*	*

(*) *SIS was not able to synthesize.*
(**) *Relative time units.*

Table 3 shows the synthesis results for the same examples using the GERPAR tool with priority parameter OPT1.

Table 3: Synthesis results for the decomposed FSM

name	sub-FSM	states	gates	critical path
sand	1	6	245	25.06
	2	6	158	17.93
planet	1	19	367	31.00
	2	13	156	17.07
scf	1	47	450	38.28
	2	42	437	37.09
contro-ller	1	123	747	50.79
	2	124	747	48.42
blanca	1	250	3564	214.80
	2	250	3360	195.52
mónica	1	300	3841	254.12
	2	300	3898	290.85
aida	1	200	4936	292.72
	2	200	5221	301.97
master	1	350	12602	446.02
	2	350	6954	279.55

Besides a performance improvement for the smaller examples, SINMEF also found solutions for the larger ones.

Table 4 shows the area and the critical path gain obtained for the 4 decomposed FSMs. The *sand* benchmark showed the largest area gain (14.79%). The *controller* example showed the largest critical path gain (60.42%).

The recursive use of the DECMEF system on FSMs containing up to 3000 states showed that the resulting sub-FSMs are small enough to be optimized by the SIS system. For example a FSM with 3140 states, 20 inputs, 12 outputs and 14152 transitions was successfully decomposed into 8 sub-FSMs with 100 states each.

Table 4: Area and critical path gain for the decomposed FSMs

name	area (%)	critical path (%)
sand	14.79	35.42
planet	3.14	38.83
scf	-11.29	52.49
controller	3.42	60.42

5 Conclusions

This paper presented a powerful environment, called SINMEF, to synthesize large FSMs, based on the DECMEF FSM decomposition system and the SIS logic synthesis system. The GERPAR tool partitions the original states into a set of super-states, each representing one sub-FSM of the decomposed FSM. The GERTAB tool removes the redundant transitions and encodes the auxiliary present states required to remove the non-deterministic transitions resulting from the decomposition procedure. A parameterized cost function allows the user to guide the whole procedure towards an optimal solution. Finally the DECMEF system generates the KISS description for each sub-FSM.

The SINMEF environment was tested for several known benchmarks as well as for some large examples (more than 100 states). The results showed a performance improvement in area of 14.79% and of 60.42% in critical path for the decomposed FSMs in respect to the original ones.

6 References

[1] Ashar, S. Devadas, A. R. Newton. "Sequential logic Synthesis". Kluwer Academic Publishers, 1992.

[2] Devadas, A. R. Newton. "Decomposition and Factorization of Sequential Finite State Machines". IEEE Trans. Computer-Aided

Design, pp. 1206-1217, November 1989.
[3] Hartmanins, R. E Stearn. "Algebraic Structure Theory of Sequential Machines". Englewood Cliffs, Nj: Prentice Hall, 1966.
[4] Kukula, S. Devadas. "Finite State Machine Decomposition By Transition Pairing". Proc. of ICCAD, pp. 114-417, November, 1991.
[5] Lin, R. Newton. "Synthesis of Multiple Level logic from Symbolic High-Level Description Languages". Proc. of VLSI89, Munich, Germany, pp. 187-196, August 1989.
[6] H. Llanos Quintero, M. Strum. "DECMEF - An Interactive FSM Synthesis System". Proc. of X SBMicro, Canela, Brasil, pp. 59-68, August 1995.
[7] H. Llanos Quintero , M. Strum. "GERPAR and GERTAB2, Two FSM Decomposition Optimization Tools". Proc. of XI SBMICRO, Aguas de Lindoia, pp. 28-33, August, 1996.
[8] Saucier, M. Crastes de Paulet, P. Sicard. "ASYL: A Rule-Based System for Controllers Systhesis". IEEE Trans. on Computer-Aided Design, pp. 1088-1097, November 1987.
[9] Sagiovanni-Vicentelli et all. "SIS: A System for Sequential Circuits Synthesis". Electronics Research Laboratory, memorandum No. UCB/ERL M92/41. University of California, Berkeley, May 1992.
[10]Villa, A. Sangiovanni-Vicentelli. "NOVA: State Assignment of Finite Machines for Optimal Two-Level Logic Implementations". Proc. of 26th ACM/IEEE DAC, pp. 905-924, 1989.
[11]Yang, R. Owens, M. Irwin. "Multi-way FSM Decomposition Based on Interconnect Complexity". Proc. of EuroDac, Hamburg, Germay, pp. 390-395, September, 1993.
[12]Hasan, M. J. Ciesielski. "FSM Decomposition and Functional Verification of FSMs". VLSI Design: An International Journal of Custom-Chip Design, Simulation and Testing, Vol. 3, No 3-4, pp. 249-265, 1995
[13]Z. Kohavi "Switching and Finite Automata Theory". Computer Science Series, second edition 1978.