A New Approach for Factorizing FSM's

C. Rama Mohan
Cadence Design Systems(India) Pvt Ltd
SDF A-1/B-8, NEPZ, Noida-201305, India

P.P. Chakrabarti
Dept of Computer Science and Engg
I.I.T, Kharagpur -721302, India

Abstract

Exact Factors as defined in [2], if present in an FSM can result in most effective way of factorization. However, it has been found that most of the FSM's are not exact factorizable. In this paper, we have suggested a method of making FSM's exact factorizable by minor changes in the next state space while maintaining the functionality of the FSM. We have also developed a new combined state assignment algorithm for state encoding of Factored and Factoring FSM's. Experimental results on MCNC benchmark examples, after running MISII on the Original FSM, Factored FSM and Factoring FSM have shown a reduction of 40% in the worst case signal delay through the circuit in a multilevel implementation. The total number of literals, on an average is the same after factorization as that obtained by running MISII on the original FSM. For two-level implementation, our method has been able to factorize Benchmark FSM's with a 14% average increase in overall areas, while the areas of combinational components of Factored and Factoring FSM's have been found to be significantly less than the area of the combinational component of the original FSM.

1 Introduction

Decomposition of a large FSM into smaller interacting submachines often leads to improvement in the performance of the circuit. Pranav Ashar et.al. [1] have suggested the use of the sum total of the number of product terms in the one-hot coded and logic minimized submachines, as a cost function for optimum two-way decomposition. Effective Factorization of an FSM also results in smaller interacting submachines. The factoring submachine has a state corresponding to each state in the factor, and the factored submachine has a state corresponding to each factor occurrence. The factored and factoring submachines interact as shown in figure 1. The realization of such interacting FSM's, will be cost-effective, if the flow of state information between the factored and factoring submachines is minimal. Exact factoring as defined in [2], leads to maximally reducing the number of states and transition edges in the original machine, and thus results in a cost-effective decomposition. Often exact factors do not exist in a given machine. In such cases, inexact factors are found. However, this does not yield as good a decomposition as the one with exact factors. Thus it is worthwhile to consider the possibility of making an FSM, exact factorizable. In this work, we have attempted to extract maximal number of exact factors out of an FSM, irrespective of its original factorizability, by suitable modifications in the STT of the FSM and addition of extra hardware to retain the functionality. We have also evolved a method of combined state assignment for the resultant Factored and Factoring FSM's, so as to reduce the area overhead.

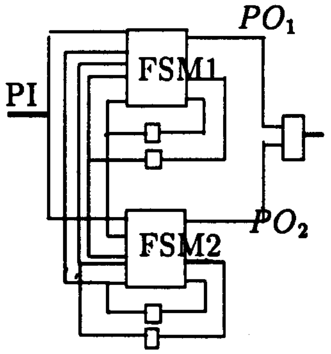

Figure 1: Interacting FSM's

A **factor** is sets of states and all fanout edges from these sets of states in the given machine. Each set of states is called an **Occurence** of the factor. A transition edge in the occurence of a factor is an **Internal edge** if it fans into and fans out of states within its factor. A state in a factor occurence is an **Internal state**, if all edges from the states fan into states within its occurence alone. An **Exit state** in the occurence of a factor, is a state that has no fanout edges within its occurence. A **E-reachable factor** is a factor where, in each occurence, at least one state exists from which all other states in that occurence can be reached. Given two occurences of a factor o_1 and o_2, a **state correspondence pair** (s_1, s_2) is a pair in which $s_1 \in o_1$, and $s_2 \in o_2$, and neither s_1 nor s_2, are present in any other correspondence pair. A factor is **Exact** if, (1) State correspondence pairs for all states in o_1 and o_2 can be found (2) For each internal edge with identical input labels, such that $e_1 \in o_1$ and $e_2 \in o_2$, $(e_1 \rightarrow fanin, e_2 \rightarrow fanin)$ and $(e_1 \rightarrow fanout, e_2 \rightarrow fanout)$ are state correspondence pairs, (3) The factor has only internal and exit states. Consider the STT of an FSM as given in figure 2.

In the corresponding STG for figure 1, the I/O of edge $< st1, st4 >$ matches with the I/O of edge $< st5, st6 >$. Similarly, I/O of edge $< st4, st2 >$ matches with the I/O of edge $< st6, st3 >$. Now,

©1994 ACM 0-89791-690-5/94/0011/0698 $3.50

0	st0	st7	0	1	st4	st6	0
1	st0	st4	0	0	st5	st6	1
0	st1	st4	1	1	st5	st6	1
1	st2	st5	0	0	st6	st3	0
0	st3	st1	1	1	st6	st5	0
0	st4	st2	0	0	st7	st3	1
1	st7	st7	1				

Figure 2: STT of an FSM

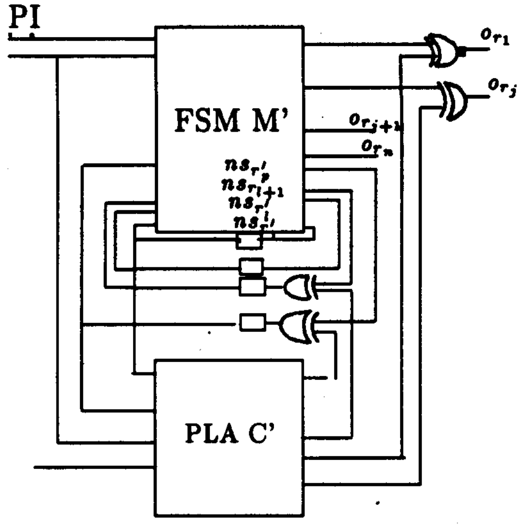

Figure 3: MAR model of implementation

let us explore the possibility of making $\{st1, st4, st2\}$ and $\{st5, st6, st3\}$ as two occurences of of an exact factor. Since states st2 and st3 have no fanout edges to states within their factor occurences, the possibiltiy of $\{st1, st5\}$ and $\{st4, st6\}$ being internal states and st2, st3 being exit states in their respective factor occurences can be considered. All fanout edges from st1,st4 have corresponding fanout edges from st5, st6 with identical I/O labels except the following pair.

e_1:	1	st6	st5	0
e_2:	1	st4	st6	0

If the next state in e_2 would have been st1, instead of st6, {st1, st4, st2} and {st5,st6,st3} would become two occurences of an exact factor. In that case, in the factored FSM fsm1, a state s'_1 corresponds to the factor occurence {st1, st4, st2} and a state s'_2 corresponds to the occurence {st5, st6, st3}. Thus we obtain two FSM's, of four states each, from the given FSM, by extracting exact factors of size 3. By changing edge e_2 to e'_2, the functionality of original FSM is obviously changed. To ensure that the functionality remains the same, we have used the model shown in figure 3, for the implementation of the FSM. As shown in the figure 3, the next state lines $ns_{r'_1}$ to $ns_{r'_i}$ are fedback through the flip-flops to FSM M', while $ns_{r'_{i+1}}$ to $ns_{r'_p}$ are fed to the inputs of exclusive-OR gates, whose outputs are given to flip-flops. While the FSM M' is obtained by changing the next states and outputs of some of the rows in the original FSM M, so that M' is factorizable, the function of the restoring PLA C' is to maintain the functionality of the original circuit, by generating the next state codes at the input of M' and the outputs o_{r_1} to o_{r_n} same as intended in the transitions of FSM M. In the above example, for changing edge e_2 to e_2' , st6 has to be changed to st1 in the next state space. Suppose the codes for states st6 and st1 differ in d places. The restoring PLA C' in that case, has a row, which generates '1' at d output lines, in positions r_1' to r_d' for input '1' and present state st4. These are the d positions where the codes of states st6 and st1 differ. At these positions in the next state lines, the exclusive-OR output is the complement of ns'_{r_1} to ns'_{r_d} , generated at the output of M'. Thus the functionality of M is maintained by this circuit.

2 Extraction of exact factors

Due to constraints of space, we will give a brief summary of the method followed by us for extracting exact factors, and state the results. The subproblems associated with the extraction of exact factors out of an FSM, by modifying the output space in the STT of the FSM, are,
1. To modify the next state and output of a given FSM, such that maximal number of exact factors can be extracted.
2. To extract exact factors from the given FSM, by the addition of minimal size restoring PLA.
3. To perform combined state assignment for the Factored and Factoring FSM's together.
The problem of achieving maximum factorization out of an FSM, by the addition of minimal overhead combinational circuit, is a trade-off between the three problems stated above.
Exact factors as defined in [2] lead to most cost-effective decomposition of FSM after factorization. Often exact factors does not exist in a given machine, in which case, inexact factors are found that does not yield as good a decomposition as the one with exact factors. Thus it is worthwhile to consider the possibility of making an FSM, exact factorizable by using Modify-Restore technique. The problems associated with the above mentioned technique of extracting exact factors out of an FSM, by modifying the output space in the STT of the FSM, are,
1. To modify the next state and output of a given FSM, such that maximal number of exact factors can be extracted.
2. To extract exact factors from the given FSM, by the addition of minimal size restoring PLA.
The problem of achieving maximum factorization out of an FSM, by the addition of minimal overhead combinational circuit, is a trade-off between the two problems stated above. Let us simplify the task, by extracting only E-reachable exact factors, with *two* occurrences. For this purpose, we have to find two

paths, in the STG of the given FSM, such that the states in the two paths, form correspondence pairs, and the transitions are labelled with identical input/output. Such paths are called ***isomorphic paths.*** In our method, first we find the longest pair of all isomorphic paths starting from the pair of states (X_{m_i}, Y_{m_i}), in the STG of a given FSM. The maximum length pair T_{max} is made as an exact factor, by suitable modifications in the next states and outputs of some transitions in the STT. Each such modification will introduce a row in the restoring PLA. The overall problem is to extract those isomorphic paths, whose corresponding exact factors will lead to Factored and Factoring FSM's(fsml and fsm2), with minimal number of states and edges, and minimal size restoring PLA.

Note that the number of columns in the PLA of the FSM will depend on the state encoding length for fsml and fsm2, which in turn depends on the constraints obtained by the symbolic minimization of these FSM's. The restoring PLA, necessary to preserve the functionality of the original FSM, in which the next states and outputs of some transitions are modified, is also generated. The output state space for those rows in the restoring PLA, which are present to maintain functionality of the original FSM, due to output state change, is determined only after state assignment is done, as the positions where state codes for two different states does not match, is determined by state encoding. Once the FSM M' and the restoring PLA C' are generated, exact factors are extracted from M', Factored and Factoring FSM's are generated and individually minimized.

3 State Assignment for Factorization model

Performing state assignment for factored and factoring FSM's , using techniques like KISS and NOVA , does not yield overall minimal area implementation, as increasing the encoding length of one FSM increases the area of the other FSM. Hence a combined state assignment strategy that leads to overall area minimization has to be developed for area-effective state assignment of Factored and Factoring FSM's. In our new algorithm for state assignment of interacting FSM's, a suitable cost is evaluated for each constraint in M_1 and M_2, after symbolic merging. The constraint that yields minimum value of the cost function, is chosen and the corresponding incompatibility graph is constructed. In the next step, the cost is reevaluated, for the unselected constraints. New constraints are selected in the subsequent steps, such that the cost gets minimized in a greedy manner.

Let n_{r_i} represent the number of transistors which are connected to row r_i and n_{o_j} represent the number of transistors which are connected to column o_j. Since the time constant for the output signal to rise in response to a signal at the input of the device depends on the impedance of the device, the row and column in the PLA corresponding to the maximum value of $(n_{r_i} + n_{o_j})$ slow down the signal propagation most. Thus $\mathrm{MAX}(n_{r_i} + n_{o_j})$ is a reasonably good measure of the performance of the FSM with PLA as a combinational component. While using MAR model for factorization, this measure is calculated for the combinational components of Factored and Factoring FSM's, and for Restoring PLA. The largest of these three values, is finally going to govern the duration of the system clock. The values of this measure are given in table 2.

4 Experimental Results

The algorithm has been written for extraction of exact factors with two occurences. The implementation is done in C-Language. The following steps have been performed-
1) Symbolic Two-level minimization has been performed on the STT of the given FSM, 2) The best exact factor has been extracted , the corresponding Factored FSM, Factoring FSM and Restoring PLA have been generated using our method. The Factored and Factoring FSM's are optimized using symbolic Two-level minimizer, 3) DIET [4], an input encoding algorithm, has been run on the symbolically minimized Original FSM and the combined state assignment algorithm mentioned earlier, has been run on Factored and Factoring FSM's to generate the final realization. The areas and performance measures(as described in previous section) of Factored FSM(fsml), Factoring FSM(fsm2) and the Restoring PLA(C), are calculated, 4) Individual state assignment has also been done for fsml and fsm2, and the areas and their performance measures are evaluated for comparision against the combined state assignment strategy.
Table 1 gives the number of product terms and edges in the Original FSM(M), Factored FSM(fsml) and Factoring FSM(fsm2), and the number of product terms in the restoring PLA(C), as obtained from the experiments conducted.
In table1, F denotes the exact factor, st = number of

	M		*fsm1*		*fsm2*		*F*	*C*
BM	p_1	*st*	*ed*	*st*	*ed*	*st*	n	r_{pla}
bbara	44	10	21	6	11	4	3	15
bbsse	23	16	20	14	9	3	2	9
bbtas	22	6	15	4	10	3	2	7
train11	11	11	9	9	5	3	2	2
dk27	10	8	5	3	5	4	3	4
dk512	23	15	16	9	13	5	4	9
cse	38	16	36	14	10	3	2	7
dk16	80	27	65	23	23	4	3	17
ex2	38	19	30	9	15	7	6	20
ex3	21	10	7	4	9	5	4	15
ex4	18	14	12	10	7	4	3	2
ex5	21	9	14	7	9	3	2	4
shiftreg	16	8	9	4	6	4	3	7
sand	163	32	151	18	23	9	8	2
lion9	11	9	12	7	7	3	2	6

Table 1: States and edges

states, ed = number of edges in STG, and n = number of 2 factor occurences. In table 2,
A= Area of Original FSM, after state assignment using DIET, A_1= Area of fsml+fsm2+Restoring PLA, by running DIET individually on the FSM's, and A_2=

	D_1	D_1	D_3				
BM	*M*			*P*	*A*	A_1	A_2
train11	15	12	12	12	286	402	379
bbara	35	19	13	8	1100	1347	1230
bbsse	23	22	22	20	966	1696	1420
bbtas	13	9	10	4	330	470	516
ex3	19	14	12	6	504	698	545
dk17	21	14	18	10	462	814	722
ex4	13	10	10	8	558	812	728
ex5	16	12	15	10	441	643	577
ex2	30	20	17	16	1140	1795	1386
dk27	11	8	7	4	160	239	205
shiftreg	11	7	6	4	192	238	212
sand	113	95	94	34	8965	12028	9355
dk16	51	43	40	10	2720	2849	2799

Table 2: Slowest signal delay and PLA areas

Area of fsm1+fsm2+Restoring PLA, by running our combined state assignment algorithm.
Area of the EX-OR gates is not considered in area calculations of tables 1 and 2.
D_1=Slowest signal delay measure(SSDM) in Original FSM, D_2=Maximum of SSDM's for fsm1 and fsm2, after individual state assignment, and D_3=Maximum of SSDM's for fsm1 and fsm2, after combined state assignment.
P indicates SSDM in the restoring PLA. As evident from the results, the Factored and Factoring FSM's have fewer number of states and edges than the original FSM after symbolic minimization. The reduction is more significant in cases where bigger factors could be extracted. **Except sand, in all other examples, no exact factor could be obtained from the Original FSM, without modification of output space.** After formation of Factored and Factoring FSM's, our state Assignment algorithm has been executed on fsm1 and fsm2 , to estimate the areas . Some further reduction in the areas of Restoring PLA can be obtained by running a two-level minimizer on it. Results from table 2, show that the combined approach for state assignment of interacting FSM's, yields less area implementations, for most of the cases.

To see the impact of Factorization using MAR model on multilevel implementation of Factored and Factoring FSM's, MISII[3] has been run on the orignal(M), Factored(M_1) and Factoring FSM's(M_2). The reduced FSM's M, M_1 and M_2 have been mapped to **nand-nor** library and the Delays have been found. The number of literals and delays computed by MISII have been listed in table 3. The delays and number of literals in table 3 are **inclusive of the EX-OR gates** in the feedback path. In table 3, L is the number of literals and D the delay, and T = Total number of literals in $M_1 + M_2$ + Restoring PLA.
It can be seen from table 3 that the total number of literals, on an average, remained same after factorization using MAR model followed by combined state assignment .

5 Conclusion

In this paper, we have presented a method of exact factoring a given FSM, by manipulation of the output space of the State transition table, and implementing it with a model, that retains the functionality of the original FSM. We have also developed a combined state assignment algorithm for interacting FSM's, so that the combinational overhead required for enhancing Factorizability is low. Our measure for performance does not include interconnection delay, as for a PLA, interconnection delay is proportional to the area. From the tables 1, 2 and 3 it can be found that for FSM's which are well factorizable using MAR model, the total area as well as performance have shown improvement, while area and performance have degraded for FSM's which are less Factorizable. Since the primary motivation behind factorization, is reduction of the delay through the critical path, it is worthwhile to reduce the number of states and edges by decomposing the given FSM into factored and factoring submachines, at the cost of some increase in the overall area of the decomposed circuit.

BM	*L*		*D*			
	T	M	M_1	M_2	RC	M
bbsse	198	154	6	3.8	0.6	13.1
cse	345	296	8.3	4.9	2.9	17.5
dk16	430	453	6.8	3.8	0.1	1.5
sla	291	404	6.4	0.1	7.1	8.6
ex2	141	213	4.7	0.3	0.0	9.5
sand	570	629	7	18.2	0.1	23
ex3	87	109	0.5	3.9	1.2	9.1
ex4	133	107	5	6.7	0.0	5.7
ex5	118	114	2.7	4.0	0.2	10.8
train11	109	92	4.3	0.4	0.2	10.8
bbara	137	128	3.8	1.1	6.4	6.4
dk17	168	125	6.1	5.8	0.0	9.1
bbtas	58	27	0.5	2.1	0.1	2.7
dk27	51	40	2.8	6.1	0.0	4.3
ex7	142	103	4.0	0.2	2.2	3.8
modulo12	44	30	0.2	0.8	0.2	3.9

Table 3: MISII results on fsm1 and fsm2 versus M

References

[1] Pranav Ashar, Srinivas Devadas, and A.Richard Newton. Optimum and heuristic algorithms for an approach to finite state machine decomposition. *IEEE Trans. on CAD*, 10(3), March 1991.

[2] Srinivas Devadas and A.R. Newton. Decomposition and factorization of sequential finite state machines. *IEEE.Trans. on CAD*, 8(11), November 1989.

[3] R.K.Brayton, R.Rudell, A.Sangiovanni Vincentelli, and A.R.Wang. MIS:a multiple level logic optimization system. *IEEE Trans.on CAD*, 6(6):1062, November 1987.

[4] Saeyang Yang and Maciej J.Ciesielski. Optimum and suboptimum algorithms for input encoding and its relationship to logic minimization. *IEEE Trans.on CAD*, 10(1), January 1991.