Sādhanā, Vol. 14, Part 3, December 1989, pp. 213–218. © Printed in India.

Maximum compatible classes from compatibility matrices

NRIPENDRA N BISWAS

Department of Electrical Communication Engineering, Indian Institute of Science, Bangalore 560012, India

MS received 24 January 1990; revised 26 February 1990

Abstract. A fast algorithm for the computation of maximum compatible classes (MCC) among the internal states of an incompletely specified sequential machine is presented in this paper. All the maximum compatible classes are determined by processing compatibility matrices of progressingly diminishing order, whose total number does not exceed $(p+m)$, where p is the largest cardinality among these classes, and m is the number of such classes. Consequently the algorithm is specially suitable for the state minimization of very large sequential machines as encountered in VLSI circuits and systems.

Keywords. Compatibility; maximum compatible classes; incompletely specified sequential machines.

1. Introduction

Most algorithms for the state minimization of incompletely specified sequential machines (ISSM) require at some stage the set of maximum compatible classes (MCC). The extensively used method of determining the MCC for a given sequential machine (Kohavi 1978; Hill & Peterson 1981; McCluskey 1986) consists of two steps. First, the compatible pairs (CP) of the machine are found by the implication chart of Paull & Unger (1959). Second, the maximum compatible classes are computed by an exhaustive comparison among the compatible pairs. Since both the steps depend on exhaustive comparison, the complexity of the first step has the order $O(n^2)$, where n is the number of states of the ISSM, and that of the second step has the order $O(m^2)$, where m is the number of CP. Thus the overall complexity of the algorithm has the order $O(n^4)$. Hence this method is suitable only for small and medium size machines not exceeding, say, 15 states. However, with the advent of very large scale integration (VLSI), many hardware and software systems are now being designed as a collection of finite state sequential machines (DeMicheli & Sangiovanni-Vincentelli 1983). Such machines are invariably incompletely specified and have very large number of internal states often exceeding 15 or 20. In such a situation the existing algorithms are very uneconomical, both time and computation-wise. In this paper a new algorithm is presented for the fast computation of maximum compatible classes from the compatibility matrices with far less computation and time. The algorithm is also programable and therefore can be very conveniently developed into a computer-aided design (CAD) package for use in the design of VLSI circuits and systems.

2. The algorithm

In describing the algorithm, the frequently used terms, such as, *incompletely specified sequential machine* (ISSM), *compatibility*, *compatible pair* (CP) and *maximum compatible class* (MCC) will be assumed to have their usual definitions as given in books on switching theory and logical design (Kohavi 1978; Hill & Peterson 1981). We shall develop the algorithm by working out an example. Like other algorithms here also the compatibility relations between all pairs of states are determined by the implication chart. This compatibility relation is then depicted in the form of an $n \times n$ matrix where n is the number of states of the ISSM. The rows and columns of the matrix are the states of the machine. A 1(0) is written at the intersection of a row and column when the corresponding pairs of states is compatible (incompatible). Table 1 shows the compatibility matrix CM_1 of a 9-state ISSM showing the compatibility relations between all pairs as obtained from the implication chart. Note that the compatibility matrix CM_1 is a symmetric matrix with an all-1 leading diagonal. This is so because each state is compatible with itself. In table 1 two more columns are added to CM_1. The two additional columns give the weight (α) and candidate maximum compatible class (CMCC) of each row, which are defined below.

DEFINITION 1

The number of ones in a row will be called its *weight*.

It will be designated by α.

Again each row of the other additional column gives the collection of all states which are compatible with the state heading the row. Thus in table 1 each of the states among ABCDHI is compatible with state A, and each of the states BCDEH is compatible with the state E.

DEFINITION 2

The collection of states which are compatible with a state will be called a *candidate maximum compatible class* (CMCC). The state with which each of the states of a CMCC is compatible will be called the *generating state* of the CMCC.

Thus every state of the machine, that is, every row of the CM_1 generates a CMCC. It is also obvious that each MCC is either a CMCC itself or a subset of a CMCC. The algorithm finds all the maximum compatible classes of the ISSM from these candidate classes with the help of the following theorems and corollaries.

Table 1. Compatibility matrix (CM_1).

	A	B	C	D	E	F	G	H	I	α	CMCC
A	1	1	1	1	0	0	0	1	1	6	ABCDHI
B	1	1	1	1	1	0	1	1	1	8	ABCDEGHI
C	1	1	1	0	1	0	1	1	0	6	ABCEGH
D	1	1	0	1	1	0	0	1	1	6	ABDEHI
E	0	1	1	1	1	0	0	1	0	5	BCDEH
*F	0	0	0	0	0	1	1	0	0	2	FG
G	0	1	1	0	0	1	1	0	0	4	BCFG
H	1	1	1	1	1	0	0	1	1	7	ABCDEHI
I	1	1	0	1	0	0	0	1	1	5	ABDHI

Theorem 1. *A candidate maximum compatible class is a maximum compatible class if and only if it is one of the least cardinality* CMCC *among all the* CMCCs *and all its constituent states are pairwise compatible.*

Proof. If (Sufficiency): If all the constituent states of a CMCC are pairwise compatible then the CMCC is an MCC if it is not contained in a larger MCC. Now, since the CMCC has the least cardinality, the generating state of the CMCC has all its compatible states included in the CMCC and therefore there can be no MCC with greater cardinality which may contain the generating state within it. Therefore the CMCC cannot be contained in another larger MCC. Hence the CMCC is an MCC.

ONLY IF (Necessity): It is obvious that the two conditions which are sufficient are also necessary to make a CMCC an MCC. QED

Note that all the elements of the compatibility matrix of an MCC are ones.

COROLLARY. 1.1

A CMCC *of cardinality* 1 *or* 2 *is an* MCC.

The proof of this corollary is obvious.

Theorem 2. *A* CMCC *with the least cardinality which is itself not an* MCC *will generate more than one* MCC *with the generating state as one of its members. Each* MCC *produces an all-*1 *matrix, and its columns (or rows) form a subset of the* CMCC *which is not contained in a larger subset.*

Proof. The CMCC with the least cardinality fails to become an MCC only when one (or more) pairs in it is (are) not compatible. Therefore, each of the subsets which produces an all-1 matrix has all its states pairwise compatible. Since the subset is not contained in a larger subset, it is an MCC. QED

COROLLARY. 2.1

If there are two and only two zeros in the subcompatibility matrix of a CMCC, *then the* CMCC *produces two maximum compatible classes.*

Proof. If the element a_{ij} of a subcompatibility matrix (SCM) is a 0, then the element a_{ji} will also be a 0, since the SCM is a symmetric matrix. Now, this pair of zeros can be eliminated from the SCM in two ways, by deleting either the row and column i or the row and column j. Hence, the SCM will produce two all-1 submatrices and, therefore, TWO MCCS. QED

If the SCM of a CMCC has more than two zeros, then find all the MCC by following the generalized procedure as applied to a CM.

Theorem 3. *After all maximum compatible classes contained in a* CMCC *have been determined, the generating state of the* CMCC *can be deleted from the rest of the candidate maximum compatible classes.*

Proof. A CMCC has *all* the states which are compatible with its generating state.

Table 2. Compatibility matrix (CM_2)

	A	B	C	D	E	G	H	I	α	CMCC
A	1	1	1	1	0	0	1	1	6	
B	1	1	1	1	1	1	1	1	8	
C	1	1	1	0	1	1	1	0	6	
D	1	1	0	1	1	0	1	1	6	
E	0	1	1	1	1	0	1	0	5	
*G	0	1	1	0	0	1	0	0	3	BCG
H	1	1	1	1	1	0	1	1	7	
I	1	1	0	1	0	0	1	1	5	

Therefore, *all* maximum compatible classes having the generating state in it are determined from the CMCC, and there cannot be any other MCC which may have the generating state in it. Hence the theorem. QED

To apply theorem 1 to CM_1 of table 1, first the CMCC of the least cardinality is chosen. In this case it is the CMCC FG generated by the state F of the ISSM and it has a cardinality of 2. By corollary 1.1 FG can be listed as an MCC without any further processing. Once FG is selected as an MCC, the row and column of state F are deleted from CM_1 as a direct consequence of theorem 3.

This is how CM_2 (table 2) is derived from CM_1.

Note that while CM_1 was a matrix of order 9, CM_2 becomes a matrix of a reduced order, namely, 8. Again applying theorem 1 to CM_2 the CMCC BCG generated by the state G has the least cardinality. In order to check if the CMCC is an MCC, a submatrix with B, C and G as rows and columns is derived from CM_2. This is called the *subcompatibility matrix* SCM_2 (BCG) and is shown in table 3. Obviously all the states will be pairwise compatible, if the submatrix is an all-1 matrix. It can be seen that the submatrix SCM_2(BCG) is an all-1 matrix, as all the row weights are 3. Hence, by theorem 1, BCG is an MCC. After BCG is selected as an MCC, state G is deleted from the rows and columns of CM_2, and CM_3 as shown in table 4 is obtained.

Table 3. Subcompatibility matrix SCM_2 (BCG).

	B	C	G	α
B	1	1	1	3
C	1	1	1	3
G	1	1	1	3

Table 4. Compatibility matrix (CM_3).

	A	B	C	D	E	H	I	α	CMCC
A	1	1	1	1	0	1	1	6	
B	1	1	1	1	1	1	1	7	
*C	1	1	1	0	1	1	0	5	ABCEH
D	1	1	0	1	1	1	1	6	
*E	0	1	1	1	1	1	0	5	BCDEH
H	1	1	1	1	1	1	1	7	
*I	1	1	0	1	0	1	1	5	ABDHI

Table 5. Subcompatibility matrix $(SCM)_3$ (ABCEH).

	A	B	C	E	H	α
A	1	1	1	0	1	4
B	1	1	1	1	1	5
C	1	1	1	1	1	5
E	0	1	1	1	1	4
H	1	1	1	1	1	5

ABCH
BCEH

In this table three candidate maximum compatible classes generated by the states C, E and I have the least cardinality of 5. The corresponding candidate classes are now processed one after another. First the submatrix SCM_3 (ABCEH) as shown in table 5 is derived. The row weights of the SCM are not 5 in all rows. It has two rows A and E having weights 4. It can be easily seen that the submatrix of the SCM, SCM_3 (ABCH) will be an all-1 matrix, as the row and column E having the 0 will be absent. Similarly the SCM_3(BCEH) will also be an all-1 matrix as the row and column A having the 0 will be absent. Hence, by Corollary 2.1 ABCH and BCEH are maximum compatible classes. Note that before deriving the SCM_3(BCDEH) the state C is deleted from the CMCC as the CMCC of the generating state C have been processed (theorem 3). Hence, SCM_3(BDEH) (table 6) is derived from CM_3. This is an all-1 matrix (as all the row weights are 4). Therefore BDEH is an MCC. After this the submatrix SCM_3(ABDHI) is derived. This produces the MCC ABDHI. This completes the processing of all the least cardinality candidate maximum compatible classes of CM_3. Before deriving CM_4 from CM_3, the number of rows of CM_4 should be calculated. Here it is 4. Therefore, if CM_4 produces any MCC, its cardinality must be less than those

Table 6. Subcompatibility matrix (SCM_3) (BDEH).

	B	D	E	H	α
B	1	1	1	1	4
D	1	1	1	1	4
E	1	1	1	1	4
H	1	1	1	1	4

BDEH

Table 7. Subcompatibility matrix (SCM_3) (ABDHI).

	A	B	D	H	I	α
A	1	1	1	1	1	5
B	1	1	1	1	1	5
D	1	1	1	1	1	5
H	1	1	1	1	1	5
I	1	1	1	1	1	5

ABDHI

computed from CM_3. But all such maximum compatible classes must have already been determined. Hence, there is no need to derive or process CM_4. Procedure terminates here. All the MCCs of the machine are FG, BCG, ABCH, BCEH, BDEH and ABDHI. It should be evident now that before the program derives a CM_{k+1} from CM_k, it must check if the number of remaining rows of CM_k is less than the cardinality of the MCC found in CM_k. If yes, then the program produces a "stop derivation" signal which ends the algorithm.

The algorithm can now be summarized as follows.

Begin

DERIVE_COMPATIBILITY_MATRIX CM_1 of the given machine
FIND_MAXIMUM COMPATIBLE CLASSES of cardinality 1 and 2 from CM_1
DERIVE_NEXT_MATRIX and
COMPUTE_MAXIMUM COMPATIBLE CLASSES by deriving submatrices and checking for all-1 matrices
Repeat this procedure
Until STOP_DERIVATION signal is received

End

3. Conclusion

It is evident from the above description of the algorithm that to determine all the maximum compatible classes, the number of compatibility matrices which need to be derived from the previous matrices starting from CM_1 will not exceed the largest cardinality among the maximum compatible classes, and the number of all-1 submatrices to be derived and processed will not exceed the total number of such classes. The processing of matrices also does not involve expensive matrix operations such as multiplication or inversion. For these reasons the algorithm becomes much faster than existing ones, specially for large and very large incompletely specified sequential machines. To compute all 30 MCC of a 22-state machine the compatibility matrix algorithm took only 0·755 s of CPU time in a DEC 1090 computer, compared to 5·329 s of CPU time (Chakraborty 1987) in the same computer by the existing algorithm as presented in Kohavi (1978), Hill & Peterson (1981), and McCluskey (1986).

References

Chakraborty A 1987 *On state minimization of very large incompletely specified sequencial machines*, BE project report, Department of Electrical Communication Engineering, Indian Institute of Science, Bangalore

DeMicheli G, Sangiovanni-Vincentelli A 1983 Computer aided synthesis of PLA based finite state machines, *Proc. Int. Conf. on Computer-Aided Design* (Washington, DC: IEEE Comput. Soc. Press) pp. 154–156

Hill F J, Peterson G R 1981 *Introduction to switching theory and logical design* 3rd ed (New York: Wiley)

Kohavi Z 1978 *Switching and finite automata theory*, 2nd edn (New York: McGraw-Hill)

McCluskey E J 1986 *Logic design principles* (Engelwood Cliffs, NJ: Prentice-Hall)

Paull M C, Unger S H 1959 *IRE Trans. Electron. Comput.* EC-8: 356–357