map a complete Moore 3-tree onto a chain or ring and thus obtain diameter $2\lceil \log_2((N+2)/3)\rceil$. This algorithm yields augmented chains or rings that are outerplanar (Fig. 4(b)) in $O(N)$ time.

The algorithm of Section IV can obviously be applied to any Hamiltonian graph. While the problem of finding Hamiltonian paths in graphs is intractable in general [4], it is trivial for most nearest-neighbor arrays. Many array processors have nearest-neighbor interconnections. Examples include the Illiac-IV, the Finite Element Machine [7], and PACS [5]. The $n \times n$ nearest-neighbor array lends itself to the efficient solution of many interesting problems [8], [10], but has the disadvantage of an $O(N)$ diameter which results in poor execution of global operations such as finding maximum. We can use our algorithm to augment such arrays to obtain networks with all the advantages of nearest-neighbor arrays as well as those of tree machines. It is interesting to note that only one additional layer of interconnecting wires is required for this purpose.

Finally, we showed in Section V how the powerful perfect-shuffle interconnection can be superimposed on a two-dimensional nearest-neighbor array. This gives us an interconnection pattern with all the advantages of nearest-neighbor arrays as well as those of the perfect shuffle. The results in this correspondence show how an N node array can be augmented so that it can perform a shuffle exchange of size $N/2$ in constant time. Subsequent research [2] has shown how the same augmented array can be used to perform a shuffle exchange of size N in constant time.

ACKNOWLEDGMENT

The authors wish to thank Prof. K. E. Durrani and Dr. R. G. Voigt for their encouragement of this research and the referees for their detailed comments. A discussion with Dr. A. Aggarwal was also very useful.

REFERENCES

[1] S. H. Bokhari, "On the mapping problem," *IEEE Trans. Comput.*, vol. C-30, pp. 207–214, Mar. 1981.
[2] ——, "Shuffle exchanges on augmented meshes," in *Proc. 1st Int. Symp. Supercomput. Syst.*, Dec. 1985, pp. 613–617.
[3] N. Deo, *Graph Theory with Applications to Engineering and Computer Science*. Englewood Cliffs, NJ: Prentice Hall, 1974.
[4] M. R. Garey and D. S. Johnson, *Computers and Intractability*. San Francisco, CA: Freeman, 1979.
[5] T. Hoshino, T. Kawai, T. Shirikawa, J. Higashino, A. Yamaoka, H. Ito, T. Sato, and K. Sawada, "PACS: A parallel microcomputer systems for scientific calculations," *ACM Trans. Comput. Syst.*, vol. 1, pp. 195–221, Aug. 1983.
[6] D. E. Knuth, *The Art of Computer Programming, Vol. 1, Fundamental Algorithms*. Reading, MA: Addison Wesley, 1973.
[7] H. J. Jordan, "A special-purpose architecture for finite element analysis," in *Proc. 1978 Conf. Parallel Processing*, Aug. 1978, pp. 263–266.
[8] A. Rosenfeld, "Parallel image processing using cellular arrays," *Computer*, vol. 16, pp. 291–295, Jan. 1983.
[9] H. S. Stone, "Parallel processing using the perfect shuffle," *IEEE Trans. Comput.*, vol. C-20, pp. 153–161, Feb. 1971.
[10] ——, "Parallel computers," in *Introduction to Computer Architecture*, H. S. Stone, Ed. Palo Alto, CA: Science Research Associates, 1980.

Prime Implicants, Minimum Covers, and the Complexity of Logic Simplification

C. MCMULLEN, AND J. SHEARER

***Abstract*—We show that any Boolean function f which can be expressed in a sum-of-products form using m product terms can contain as many as $2^m - 1$ implicants but no more.**

Manuscript received October 25, 1984; revised March 16, 1985.

C. McMullen was with the Department of Mathematics, Harvard University, Cambridge, MA 02138. He is now with the Department of Mathematics, Massachusetts Institute of Technology, Cambridge, MA 02139.

J. Shearer is with IBM T. J. Watson Research Center, Yorktown Heights, NY 10598.

IEEE Log Number 8609407.

***Index Terms*—Boolean functions, complexity, logic simplification, minumum covers, prime implicants.**

Consider the problem of expressing a Boolean function f as a sum-of-products, using as few product terms as possible. Equivalently, we seek to express the onset of f as the union of as few cubes as possible. One approach to the solution of this problem is to generate all maximal cubes contained in the onset of f, and then to extract from these a minimum covering. This is the classical Quine–McClusky algorithm (see, for example, [3]).

To determine the complexity of this algorithm, we need to know how many maximal cubes (prime implicants) can arise in a problem of a given size. Let us measure the size of a problem by the minimum number of product terms in an expression for f, and denote this by $m(f)$. Assuming input to the algorithm is a sum-of-products expression for f, this is the smallest the input can be. Let $p(f)$ denote the number of prime implicants of f; these must be generated during the first step of the Quine–McClusky algorithm. Our purpose here is to exhibit a sharp bound for $p(f)$ in terms of $m(f)$. The examples showing this bound is sharp are then worst cases for exact logic simplification.

Theorem: For all Boolean functions f

$$p(f) \le 2^{m(f)} - 1.$$

This bound is sharp; for each m there exists a function f such that $m = m(f)$ and $p(f) = 2^m - 1$.

To complete the Quine–McClusky algorithm, one must extract a minimum cover from the set of prime implicants. The general minimum covering problem is NP-hard; if we assume an exact solution requires exponential time in the size of the set from which the cover is to be extracted, we obtain a worst case complexity of at least exp (exp (cm)) for the entire procedure. Here m is the number of terms in the input to the algorithm, and c is a positive constant. Of course the minimum covering algorithm might in principle perform much better; indeed, it is not known that a polynomial-time minimum-covering algorithm does not exist. Even if it does, the worst case complexity of the Quine–McClusky algorithm is exponential in m.

Proof: We begin by describing a sequence of examples which show the bound is sharp. The examples will be functions of independent Boolean variables x_1, x_2, etc.

Set $f_1(x_1) = x_1$. Then $m(f_1) = p(f_1) = 1$. Define inductively

$$f_{m+1}(x_1, x_2, \cdots, x_{2m+1}) = x_{2m} \wedge f_m(x_1, x_2, \cdots, x_{2m-1}) \vee \bar{x}_{2m} \wedge x_{2m+1}.$$

Certainly $m(f_{m+1}) \le m(f_m) + 1$, since a cover for f_{m+1} may be obtained by ANDing every cube in a cover for f_m with x_{2m} and then adjoining the single additional cube $\bar{x}_{2m} \wedge x_{2m+1}$. Also, observing that the cubes

$$x_{2m} \wedge p, \quad x_{2m+1} \wedge p, \quad \text{and} \quad \bar{x}_{2m} \wedge x_{2m+1}$$

are prime implicants of f_{m+1}, for any p which is a prime of f_m, we can assert that $p(f_{m+1}) \ge 2p(f_m) + 1$. Applying these inequalities inductively, we obtain $m(f_m) \le m$ and $p(f_m) \ge 2^m - 1$. It is a consequence of the remainder of the theorem that in fact equality holds.

We now turn to the proof of the inequality stated in the theorem. Let $\mathcal{F}$ be a set of cubes of minimum cardinality whose union is f; that is, assume $|\mathcal{F}| = m(f)$. Let p be any prime implicant of f, and let $\mathcal{G} \subset \mathcal{F}$ be an irredundant cover for p; i.e., $\mathcal{G}$ is a set of cubes whose union contains p and from which no cube can be removed while retaining this property. We claim that $\mathcal{G}$ uniquely determines p. Granting this, it is clear that the number of prime implicants of f is no larger than the number of subsets of $\mathcal{F}$; moreover, the set $\mathcal{G}$ is never empty, and hence $p(f) \le 2^{m(f)} - 1$ as claimed.

To complete the proof of the theorem, we will show that the prime p can be recovered from its covering $\mathcal{G}$.

Let $x_1, x_2, \cdots$ be the Boolean variables in terms of which the cubes

of $\mathcal{G}$ are expressed. Since p is prime, it is a product of some subset of these variables and their negations. Suppose that among the cubes of $\mathcal{G}$, some of the products involve x_i and some involve $\bar{x}_i$. Then, since every cube of $\mathcal{G}$ must meet c, neither x_i nor $\bar{x}_i$ can occur in the cube c.

On the other hand, suppose x_i occurs in only one polarity among the cubes of $\mathcal{G}$; say, for example, that some cubes of $\mathcal{G}$ involve x_i but no cube involves $\bar{x}_i$. We claim x_i must appear in the product expression for c. If not, it is easy to check that the cubes in $\mathcal{G}$ involving x_i are redundant, contrary to assumption. Indeed, if p does not involve x_i, then the product $p \wedge \bar{x}_i$ is a nontrivial subcube of c, and must be covered by the cubes in $\mathcal{G}$ not involving x_i; since these cubes in $\mathcal{G}$ do not involve $\bar{x}_i$ either, they also cover $p \wedge x_i$. But $p = p \wedge (x_i \vee \bar{x}_i)$, so the remaining cubes of $\mathcal{G}$ are redundant as claimed.

Thus by examining $\mathcal{G}$ we can determine, for each i, whether x_i or $\bar{x}_i$ occurs in the product expression for c; hence p is uniquely determined by its irredundant covering $\mathcal{G}$, completing the proof of the theorem.

The Referees and the Editor have suggested [1], [2], [4], and [5] are also relevant.

References

[1] Igarashi, "An improved lower bound on the maximum number of prime implicants," *Trans. IECE,* Japan, vol. E-62, pp. 389–394, June 1979.

[2] F. Mileto and G. Pulzolv, "Average quantities appearing in Boolean function minimization," *IEEE Trans. Electron. Comput.*, vol. EC-13, pp. 87–92, 1969.

[3] S. Muroga, *Logic Design and Switching Theory*. New York: Wiley, 1979.

[4] C. R. Papochristou, "Characteristic measures of switching functions," *Inform. Sci.*, vol. 13, pp. 51–75, 1977.

[5] S. Y. Yablonskii, "The problem of bounding the length of reduced disjunctive normal forms," *Problemi Kibernetki,* vol. 7, pp. 229–230, 1962; translation, *Problems of Cybernetics.* London: Pergamon.

An Implementation of Mixed-Radix Conversion for Residue Number Applications

N. B. CHAKRABORTI, JOHN S. SOUNDARARAJAN, AND A. L. N. REDDY

***Abstract*—A method of residue number system (RNS) conversion to mixed-radix (MR) representation is presented. This method is found to be cost-effective and efficient, particularly for moduli size 4/5 bits. A comparison of conversion times and hardware necessary for RNS conversion to MR digits based on different methods is also presented.**

***Index Terms*—Mixed-radix representation, parallel lookup, residue number system, serial lookup.**

I. Introduction

The residue number system (RNS) is being increasingly used to implement high-speed, high-throughput digital computing and signal processing because of the parallel nature of its arithmetic [1]–[4]. However, RNS is found to be inferior to weighted number systems since sign determination, magnitude comparison, and overflow detection are slow. Such operations can be carried out if the mixed-radix (MR) digits of a number are known. Several residue to MR conversion algorithms have been reported. A classical conversion algorithm may be found in Szabo and Tanaka [5]. A converter for a three moduli set has been described in [6] and a fully parallel algorithm has been reported in [7]. An RNS digital-to-analog conversion based on Chinese remainder theorem (CRT) has been described in [3]. A cost-effective MR converter implementation using commercially available chips for serial or parallel table lookup with multiple residue addressing is presented here.

Manuscript received: December 4, 1984, revised: June 13, 1985.

The authors are with the Department of Electronics and Electrical Communication Engineering, Indian Institute of Technology, Kharagpur 721 302, (W.B.), India.

IEEE Log Number 8609408.

II. MR Conversion Using Serial Lookup

In RNS using relatively prime moduli $R_1, R_2, \cdots, R_n$ the dynamic range is $M = \Pi R_i$. An integer X, $0 < X < M$, has the residue representation $X \equiv (r_1, r_2, \cdots, r_n)$ where $r_i = \langle X \rangle R_i = X$ modulo R_i. In MR representation, the integer X is written as $X = a_1 + a_2R_1 + \cdots + a_nR_1R_2 \cdots R_{n-1}$ where $a_1, a_2, \cdots, a_n$ are the MR digits. It is seen that MR system is a weighted number system and a_n is the most significant digit. We assume that the ROM and moduli sizes are such that two moduli may be used to address a ROM table lookup. Considering a four moduli system, the number X in serial lookup is represented as $X = X_1 + X_2$ where $X_1 = a_1 + a_2R_1 = (r_1, r_2, r_3(0), r_4(0))$ and $X_2 = qR_1R_2 = a_3R_1R_2 + a_4R_1R_2R_3 = (0, 0, \langle r_3 - r_3(0)\rangle R_3, \langle r_4 - r_4(0)\rangle R_4)$. $r_1, r_2, r_3(0), r_4(0)$ are the residues of X_1 in the subfield $[0, R_1R_2 - 1]$. Thus, the process of conversion consists of finding a_2, $r_3(0)$, and $r_4(0)$, and using $r_3' = \langle r_3 - r_3(0)\rangle R_3$ and $r_4' = \langle r_4 - r_4(0)\rangle R_4$ to find a_3 and a_4 from direct mapping. The architecture for this algorithm is shown in Fig. 1. Memory chip $T1$ gives a_2 and chip $T2$ maps $r_3(0)$ and $r_4(0)$ as a direct function of r_1 and r_2; r_3' and r_4' may be used to find a_3 and a_4 through chip $T3$. It is possible to carry out the subtractions $\langle r_3 - r_3(0)\rangle R_3$ and $\langle r_4 - r_4(0)\rangle R_4$ by normal adders. The size of ROM's gets doubled for each residue addressing in such a case. Furthermore, all locations addressed by a number r_3 should have the same mapping as that addressed by $(r_3 - R_3)$ where $(r_3 - R_3)$ can be represented in two's complement form. Similarly for r_4. Considerable simplification results if $\langle R_1R_2\rangle R_3 = 1$ in which case, $a_3 = r_3'$ and chip $T3$ need provide a_4 only. Standard high-speed memory sizes are restricted to 4K words. This restricts the use of the above ROM table lookup method to 6-bit (5-bit, if normal adders are used) moduli.

The following example is presented to illustrate the method. Let $R = (3, 5, 7, 11)$, $M = 1155$, and $X = 143$. Therefore, $X = (2, 3, 3, 0)$ in residue number representation.

	2	3	3	0	
Table lookup		2	1	8	Chip $T1$ outputs $a_2 = 2$
r_3', r_4'			2	3	
Table lookup gives			2	1	

Therefore, MR digits are (2, 2, 2, 1). Here we can verify $r_3' = a_3$ as $\langle 3 \times 5\rangle 7 = 1$. Scaling can also be efficiently performed by the proposed method as r_3' and r_4' can be used to map into the autoscaled variable, as described in [8]. The proposed method gives an efficient way of truncating X to obtain $\bar{X} = (0, 0, r_3', r_4')$ where $\bar{X} = (X - a_2R_1 - a_1)$, with a conversion time equal to one table lookup plus one addition time. It may be noted that X_2 may be computed using MR digits of r_3' and r_4' (see Section III). Alternatively, one may use CRT to find $q = \langle x_3R_3 + x_4R_4\rangle R_3R_4$ where $x_3 = \langle r_3/R_1R_2R_4\rangle R_3$ and $x_4 = \langle r_4/R_1R_2R_3\rangle R_4$; this is useful when the number q in the range $[0, R_3R_4]$ is specifically desired.

The above method can be extended to larger number of moduli. The general procedure for n moduli (for n even) is outlined below.