ALGORITHMS FOR IMPLEMENTING MULTI-VARIABLE KARNAUGH MAP ON CUBE-CONNECTED AND MESH-CONNECTED COMPUTERS

Yau-Tak Luk and Mi Lu
Electrical Engineering Department
Texas A&M University
College Station, TX 77843

Jiming Yin
Electrical Engineering Department
University of Houston
Houston, TX 77004

ABSTRACT

The application of the Karnaugh map in digital circuit design is limited by the visual inspection ability and becomes intrivial as the number of variables increases beyond five. This paper introduces a computer aided method for multi-variable Karnaugh mapping which can be implemented on Mesh-Connected Computers or Cube-Connected Computers. In addition to making the above problem solvable, we make effort to exploit the inherent parallelism and to improve the implementing speed.

I. INTRODUCTION

The Karnaugh map, for short K-map, is a powerful tool in the digital logic design which can simplify Boolean functions by recognizing patterns in graphical representation of data [1-3]. As the number of variables increases beyond five, however, the visual inspection becomes impracticable due to the human ability. This paper introduces a computer aided method for multi-variable Karnaugh mapping. Parallel algorithms are presented for implementing on Mesh-Connected Computers (MCC) and Cube-Connected Computers (CCC) which have been widely used to solve various problems [4-12].

The design of a map for the simplification of, say, eight variable function is shown in Fig. 1. Each minterm number of a cell can be naturally mapped to a PE index on MCC's or CCC's and thus the minterms can be distributed. Any two cells in the map differing only in one literal will contain "adjacent" minterms and the variable in which they differ can be eliminated. For an MCC or CCC consists of N PE's, our algorithms can perform the K-map simplification for n variables where $n = \log N$. The algorithm includes n iterations. In an iteration, the minterms stored in the corresponding PE's of the paired blocks can be grouped, and the groupings in different pairs of blocks can be performed in parallel. The bookkeeping for the variable eliminations are maintained. Two kinds of implicant, the essential prime implicant and the non-essential prime implicants, can be recognized, and redundant groupings can be avoided. In addition, the "don't cares" can be included to form a maximum sized group so that the prime implicant contains the fewest literals.

II. ALGORITHM FOR MESH-CONNECTED COMPUTERS

2.1 PRELIMINARIES

$N$ PE's are utilized to perform the K-map simplification for $n$ variables where $\log N = n$. Each PE is represented by a binary index $(a_{n-1} \cdots a_0)$, which corresponding to a cell on the K-map. For example, the PE with index 1001 stores the input variable combination $ABC$. The PE's on the mesh are indexed in shuffled row-major order as shown in Fig. 2 of a $4 \times 4$ MCC. The shuffled row major is obtained from the standard row-major order by shuffling the binary representation of the PE indices.

Our algorithm includes $n$ iterations where $n$ is the number of the input variables. The $i$th iteration is performed on a $2^{i+1}$-blocks [9,10] which consists of $2^{i+1}$ PE's. The indices of the PE's in a $2^i$-block agree on the $n-i$ most significant bits (MSB's).

PE$(j)$ and PE$(k)$ are said to be adjacent in level $i$ if their indices differ in $b_i$ only. The minterms stored in such pair of PE's will differ only in one literal, and hence are adjacent. The adjacent PE may be paired and the adjacent minterms can be identified. The variables in which they differ will be eliminated and the prime implicant will be formed by "covering" the adjacent minterms. The degree of adjacency of a minterm is defined as the number of other minterms differing from the minterm by one change of variable. If a group of minterms with degree of adjacency $k$ combined with other minterms each having degree of adjacency $k$ or more to form a larger group of $2^{k+1}$ minterms, then an

† This research is partially supported by the National Science Foundation under Grant No. MIP-8809328.
essential prime implicant is generated from the group.
An essential prime implicant covers all the minterms in the group. For the example in Fig. 3, group 3 is an essential prime implicant from group 1, but a non-essential implicant from group 2.

The PE with \( b_i = 0, 0 \leq i < n \), in its index is referred as the receiver and the one with \( b_i = 1 \) is referred as transmitter of iteration \( i \). The iterations alternate between a horizontal and a vertical step, beginning with a horizontal step. In iteration \( i \), data communication occurs between a pair of \( 2^i \)-block submeshes. All the PE's will route the data to the \( i \)th-level adjacent neighbors. Concurrent data movements are thus made possible (see Fig. 4). Thus, group 1 in Fig. 3 is generated in the first iteration, group 2, 4 and 5 are generated in the second iteration, and group 3 is generated in the third iteration.

Associated with each PE are the following records.

\[ PE \ (a_{n-1} \cdots a_1a_0) \]

\( S(mz) \), a two bits indicator. \( mz = 10 \) when the maintained minterm function is a “1”, \( mz = 01 \) when it is a “don’t care”. \( mz = 00 \) otherwise.

Record \( V(v_{n-1} \cdots v_1v_0, r) \), indicating the presence of the \( i \) variables in the prime implicant, and the responsibility for reporting the prime implicant. Denote \( (v_{n-1} \cdots v_1v_0) \) as \( V \) pattern. Each PE will stack up all the \( V \) records resulted from every iteration. Only the PE's with \( S(mz) = 10 \) or 01 keep the \( V \) records which is initially \( (1 \cdots 11, 1) \). Other PE’s have null \( V \) records. If variable \( v_i \) is eliminated in a prime implicant, \( v_i = 0 \) is set in the record for this prime implicant. If the associated \( r = 1 \), the PE should report the corresponding prime implicant in the final stage; otherwise, no report is necessary. As an example, if \( PE(1100) \) has \( V(v_3v_2v_1v_0, r = (1010, 1)) \), it should report the prime implicant \( A_3A_1 \). This is because that the minterm stored in \( PE(1100) \) is \( A_3A_2A_1A_0, v_2 = v_0 = 0 \), and hence the variables \( A_2 \) and \( A_1 \) are eliminated.

**2.2 MCC ALGORITHM**

Performing the Boolean function simplification on K-map, one would attempt to (a) discover the largest prime implicant (largest in terms of the number of minterms), (b) identify the irredundant prime implicant and (c) utilize the “don’t cares” to eliminate more literals. These three issues are realized as follows.

(a) Consider two patterns \( V(v_{n-1} \cdots v_1v_0) \) and \( V'(v_{n-1} \cdots v_1v_0), 0 \leq i \leq n-1 \). If \( v'_i = 0 \ \forall i \in \eta \), such that \( \eta = \{ i \mid v_i = 0, 0 \leq i \leq n-1 \} \), we say \( V \|_0 \subseteq V' \|_0 \). For the example in Figure 3, group 4 has \( V_4 \) pattern (1100) and group 5 has \( V_5 \) pattern (1101), therefore, \( V_5 \|_0 \subseteq V_4 \|_0 \). In addition, group 2 has \( V_2 \) pattern (1100) and group 4 has \( V_4 \) pattern (1100), thus \( V_2 \|_0 = V_4 \|_0 \). Note that a new group can be generated from group \( j \) and \( k \) only if \( V_j \|_0 \subseteq V_k \|_0 \) or \( V_k \|_0 \subseteq V_j \|_0 \).

(i) If a new group is formed in iteration \( i \) when \( V_i \|_0 = V_j \|_0 \), i.e., an essential prime implicant is formed, we update the records for the minterms in group \( j \) and \( k \) by setting \( b_i \) to 0. (ii) If \( V_j \|_0 \subseteq V_k \|_0 \), all the minterms in group \( j \) are covered by the new group and we update the records for them by setting \( b_i \) to 0. In the meantime, part of the minterms in group \( k \) are covered and the new records for them should be as same as those for the minterms in group \( j \). (iii) If \( V_k \|_0 \subseteq V_j \|_0 \), the situation will be vice versa. In the above three cases, if \( r = 0 \) in the records for both group \( j \) and group \( k \), then \( r = 0 \) in the updated or new generated cords. Otherwise, set \( r = 1 \) in the updated or new generated record maintained by the PE which is a receiver in iteration \( i \), and set \( r = 0 \) in those maintained by the PE which is a transmitter. (b) After all the prime implicats are generated, a procedure will be conducted for checking the redundant groups. The main approach is that each PE computes the number of the groups, say \( C \), by which the minterm it contains is covered. PE’s then perform the random access write (RAW) with conflict [5] to write the \( C \) value to the reporters of each group. A reporter only accepts the minimum of the \( C \) values written to it. If it is greater than 1, then the group should not be reported since all the members with \( C \geq 2 \). Recall that only the group containing at least one minterm which can not be covered by any other group can be considered as a prime implicant. To find the reporter of a group, each PE can examine the corresponding \( V(v_{n-1} \cdots v_1v_0) \) pattern for the group. Performing a bitwise AND upon the PE index and the \( V \) pattern will result in the PE index of the reporter. For example, if \( PE(1011) \) has the \( V \) pattern equal to 0101, then the reporter will be the PE with index 0001 (see Fig. 5). Note that a PE may contain \( 2^i \) records in the worst case. All the groups generated in the \( i \)th iteration contain the minterms stored on a \( 2^i+1 \)-block submesh, and the RAW can be performed on different \( 2^i+1 \)-block submeshes simultaneously.

(c) Operations of RAW with conflict are also needed to write the \( S(mz) \) value to the reporters of the groups. A reporter only accepts the maximum of the \( S(mz) \) values written to it. If it is less than \( (10)_2 \), then the reporter should not report this group since all the minterms are “don’t cares” indicating by \( S(mz) = 01 \).

The elimination algorithm can be described as follows.
2.3 TIME COMPLEXITY OF MCC ALGORITHM

The time performance of the MCC algorithm can be analyzed as follows. A pair of PE's adjacent in level \( i \) are \( 2^{i/2} \) steps far away from each other. Thus, the communication time required between them in Step 1 is \( 2 \cdot 2^{i/2} \), considering the data shifts in two opposite directions. A PE may have \( 2^i \) records to route in iteration \( i \). The data routing can be pipe-lined, and the time needed can be counted as \( 2(2^{i/2} + 2^i) = O(2^{i/2}) \). Step 2 needs \( O(2^i) \) time in iteration \( i \), since the records in the stack can be considered as "sorted" in some order and the time needed to match the patterns is linear to the number of the patterns. Step 3 needs \( O(2^n) \) time in total.

In Step 4, the destination of RAW performed by a PE is within a \( 2^{i+1} \)-block submesh in iteration \( i \). \( O(2^{i/2}) \) time units are sufficient to accomplish a RAW of one data item [5], and \( O(2^{i/2} + 2^{i+1}) \) for all the data. If \( k \) is the maximal number of prime implicants which need to be reported after the elimination, then Step 5 requires \( O(k + 2^n) \) time, where \( 2^n \) is the diameter of the mesh and thus the time needed to shift data through the mesh.

In a word, the time needed to perform the elimination algorithm for \( n \) variables on an MCC is

\[
O \left( \sum_{i=0}^{n-1} 2^i + \sum_{i=0}^{n-1} 2^i 2^n + \sum_{j=0}^{n-1} 2^{i/2} j + k + 2^{n/2} \right)
\]

\( = O(2^{3/2} n) \).

III. ALGORITHM FOR CUBE-CONNECTED COMPUTERS

3.1 CCC IMPLEMENTATION

The distribution of the minterms to the PE's on a Cube-Connected Computer is similar to that on an MCC. To perform \( n \) variables Boolean function minimization, \( N \) PE's are required, where \( \log N = n \). In the Cube-Connected Computer consists of \( N \) PE's, each PE is directly connected to \( n \) neighbors. The neighboring PE's differ in exactly one bit position. In other word, the PE with binary index \( (a_{n-1} \cdots a_1 a_0) \) is connected to \( PE(a_{n-1} \cdots a_1 \cdots a_0) \), for \( 0 \leq i \leq n - 1 \) (see an example of CCC in Fig. 6 for \( n = 3 \)).

The records maintained in PE's on a Cube-Connected Computer are as same as those on the MCC. Similar algorithm as described in the previous section can be implemented on a Cube-Connected Computer. \( n \) iterations are needed for our CCC algorithm. In it-
The presented work may provide extensive field for further research and may show new directions in Computer Aided Design (CAD).

REFERENCES


Figure 2

Figure 3

Figure 4

Figure 5

Figure 6