1. REPORT NUMBER
   ARO 20090.20-EL

2. GOVT ACCESSION NO.
   N/A

3. RECIPIENT'S CATALOG NUMBER
   N/A

4. TITLE and Subtitle
   DETECTION OF BI SYMMETRIC FUNCTIONS

5. TYPE OF REPORT & PERIOD COVERED
   Internal Technical

6. PERFORMING ORG. REPORT NUMBER
   N/A

7. AUTHOR(s)
   B. Kalyanasundaram & R.M. Owens

8. CONTRACT/GRANT NUMBER(s)
   DAAG29-83-K-0126

9. PERFORMING ORGANIZATION NAME AND ADDRESS
   Department of Computer Science
   The Pennsylvania State University
   University Park, PA 16802

10. PROGRAM WORK UNIT NUMBERS
    N/A

11. CONTROLLING OFFICE NAME AND ADDRESS
    U.S. Army Research Office
    Post Office Box 12211
    Research Triangle Park, NC 27709

12. REPORT DATE
    March 1986

13. NUMBER OF PAGES
    6

14. MONITORING AGENCY NAME & ADDRESS
    N/A

15. SECURITY CLASS. (of this report)
    Unclassified

16. DISTRIBUTION STATEMENT (of this Report)
    Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the Abstract entered in Block 20)
    N/A

18. SUPPLEMENTARY NOTES
    The views, opinions, and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy, or decision, unless so designated by other documentation.

19. KEY WORDS
    Logic minimization, bi symmetric functions

20. ABSTRACT
    A detection algorithm for a new class of functions (defined in the paper) has been presented. This algorithm not only detects such functions but also provides the decomposition to facilitate automatic layout and helps in finding a faster circuit and an efficient layout. If an example (two inputs) is given to verify that the function is not symmetric then this detection algorithm asks a polynomial (on number of variables) number of queries about the function and comes up with a circuit if one exists.
Detection of BI Symmetric Functions

Balasubramanian Kalyanasundaram and Robert Owens

Department of Computer Science
The Pennsylvania State University
University Park, PA 16802

ABSTRACT

Abstract: A detection algorithm for a new class of function (defined in the paper) has been presented. This algorithm not only detects such functions but also provides the decomposition to facilitate automatic layout and helps in finding a faster circuit and an efficient layout. If an example (two inputs) is given to verify that the function is not symmetric then this detection algorithm asks polynomial (on number of variables) number of queries about the function and comes up with a circuit if one exists.
Detection of Bi Symmetric Functions

Balasubramanian Kalyanasundaram

and

Robert Michael Owens

Department of Computer Science,
Pennsylvania State University.

Abstract: A detection algorithm for a new class of function (defined in the paper) has been presented. This algorithm not only detects such functions but also provides the decomposition to facilitate automatic layout and helps in finding a faster circuit and an efficient layout. If an example (two inputs) is given to verify that the function is not symmetric then this detection algorithm asks polynomial (on number of variables) number of queries about the function and comes up with a circuit if one exists.

Introduction:

In VLSI, it is well known that a faster circuit and an efficient layout are strongly dependent on the nature of the function to be implemented. It has been found that a reasonably faster circuit with an efficient layout can be achieved if the function is symmetric. Unfortunately many functions occurring in a problem may not fall into the category of symmetric function. So it is natural to consider the layout complexity of somewhat symmetric function. In this report we have introduced a notion of bi symmetric function. As far as faster circuit and efficient layout are concerned this class of function has almost all the nice properties of a symmetric function. An algorithm has been developed to check whether a given function is bi symmetric or not. With some help this algorithm asks polynomially (on number of variables) many queries only.

Definition 1: A function \( F : (0,1)^n \rightarrow (0,1) \) is bi symmetric if there exists these three functions \( F', G_1 \) and \( G_2 \) such that

\[
F(x_1, x_2, \ldots, x_n) = F'(G_1(x_{f_1}, x_{j_1}, \ldots, x_{l_1}), G_2(x_{f_2}, x_{j_2}, \ldots, x_{l_2})),
\]

where the set \( \{x_1, x_2, \ldots, x_n\} = \{x_{f_1}, x_{j_1}, \ldots, x_{l_1}\} \) and the functions \( G_1 \) and \( G_2 \) are symmetric. Figure 1 clearly explains the definition in the circuit format.
Since the functions $G_1$ and $G_2$ are symmetric, they can be realised by a reasonably fast circuit and an efficient layout. Maximum number of output bits for both $G_1$ and $G_2$ is of the order of $\log(n)$. Hence the function $F'$ is comparatively simple to implement.

Given a function $F$, known to be not symmetric, we would like to know whether it is bi symmetric or not. An algorithm has been developed which asks queries on the given function and outputs the three functions $F', G_1$ and $G_2$ such that latter two are symmetric (fig. 2). They satisfy the property that $F = F'$ if and only if $F$ is bi symmetric.

**Definition 2:** $2^n$ combinations of a bit input can be grouped into $(n+1)$ different categories called $\text{Level}(0), \text{Level}(1), \ldots, \text{Level}(n)$ such that all strings with exactly $k$ ones are in $\text{Level}(k)$ and not in any other.

**Lemma 1:** If $F$ is not symmetric then there exists a $k, x$ and $y$ such that $x, y \in \text{Level}(k)$ and $F(x) \neq F(y)$.

**Proof (by contradiction):** If there exists no $k$ such that $x, y \in \text{Level}(k)$ and $F(x) \neq F(y)$ then for all $k$ and for all pairs $x, y \in \text{Level}(k)$ $F(x) = F(y)$. So $F$ is symmetric (a contradiction).□
Lemma 2: If $F$ is not symmetric then there exists a $k, x$ and $y$ such that $x, y \in \text{Level}(k)$ and $F(x) \not= F(y)$ and hamming distance between $x$ and $y$ is two (i.e., positions of a zero and one are swapped).

Proof (by construction): From Lemma 1, we know that there exists a $k, x$ and $y$ such that $x, y \in \text{Level}(k)$ and $F(x) \not= F(y)$. Since $x, y \in \text{Level}(k)$ the number of zero positions in $x$ that are one’s in $y$ is equal to the number of one positions in $x$ that are zero’s in $y$. Let that number be $l$ ($\leq k$). Now one can march from $x$ towards $y$, each time correcting one pair. It can be represented by the following sequence:

$$x (=x_0), x_1, \ldots, x_{l-1}, y (=x_l)$$

where $x_i$ is obtained by swapping the positions of a zero and one from $x_{i-1}$ and it is closer to $y$ (hamming distance measure). Note that hamming distance between $x_i$ and $x_{i+1}$ is two. Since $F(x) \not= F(y)$ there must be an $i$ such that $F(x_i) \not= F(x_{i+1})$. □

Fact 1: If $F$ is a bi symmetric and there exists an $x, y \in \text{Level}(k)$ such that hamming distance between $x$ and $y$ is two and $F(x) \not= F(y)$, then those bits they differ from one another must be in different $G_i$ ($i=1,2$).

Proof (by contradiction): Without loss of generality, assume that both bits belong to $G_1$. Since $G_1$ is symmetric, swapping two bits should not result in different output (a contradiction). □

Lemma 3: Let $F$ be a bi symmetric function and there exists an $x, y \in \text{Level}(k)$ such that hamming distance between $x$ and $y$ is two and $F(x) \not= F(y)$. Without loss of generality assume that $b_i$ and $b_j$ are the two bits differing (causing the hamming distance) in $x$ and $y$ and the former is an input to $G_1$ and latter to $G_2$. Let $b_i$ th bit in $x$ be one. The following statements are true.

1). All zero’s positions in $x$, that change output when swapped with the $b_i$ th, belong to $G_2$ and all other zero’s positions belong to $G_1$.
2). All one’s positions in $x$, that change output when swapped with the $b_j$ th, belong to $G_1$ and all other one’s positions belong to $G_2$.

Proof: Since $G_s$ are symmetric functions we can just concentrate on number of ones in the input. We observe that if we lose one 1 from $G_1$ and gain one 1 in $G_2$ then the result change from the original (original input being $x$). So the same should happen if we swap a one ($b_i$) to any one of the zeros in $G_2$. So all zeros in $x$ that are input to $G_2$ must change the result when swapped with $b_i$. One can make a similar argument to second statement also. □
Algorithm:
This algorithm consists of three stages. First the input is partitioned into two parts where one is input to $G_1$ and the other to $G_2$. In the second stage of the algorithm, the truth table of the three functions $F', G_1$ and $G_2$ are obtained. The third stage just verifies whether the resultant composition is $F$ or not.

Stage 1 (partitioning the given input into two parts):

1. Assume that the given function is bi symmetric.
2. Find a string $z$ such that there exists a $y$ with the property $x,y \in \text{Level}(k)$, hamming distance between $x$ and $y$ is two and $F(x) \neq F(y)$. Now apply the procedure described in Lemma 3 to find the partition.

For ease of presentation we are assuming that the number of inputs to $G_1$ and $G_2$ are equal. Let it be $m$. Since $G$s are symmetric functions, we can pick representative for each $\text{Level}(0)$, $\text{Level}(1)$, ..., $\text{Level}(m)$. Now form a boolean matrix $M ((m+1) \times (m+1))$ which is the result of $F$ for various pairs of these $(m+1)$, $(m+1)$ representative inputs for each $G_1$ and $G_2$. If $F$ is a bi symmetric function, this matrix can be rewritten by permuting some rows and columns such that we get monochromatic rectangles with the nice pattern shown in fig. 3. Note that the value written inside the rectangles in fig. 3 is arbitrary.

Figure 3.
At least there exists a trivial partition such that each monochromatic rectangle has only one element.

**Stage 2: (Finding the three functions)**

1. Pick those representatives and form the matrix $M$.
2. Collect all identical rows and bunch them together (kind of sorting).
3. Now collect all identical columns and bunch them together (kind of sorting).

Step two and three creates monochromatic rectangles. Let the rows be divided into $k_1$ parts and the columns $k_2$ parts. Let the row side be $G_1$ and column side be $G_2$. Now the number of output bits for $G_1$ is $\lfloor \log(k_1) \rfloor$. One can easily write coding for each of the $k_1$ parts using $\lfloor \log(k_1) \rfloor$.

![Figure 4.](image-url)
bits. So the $G_1$ function is nothing but the coder. Similarly one can also find $G_2$. Now the truth table for $F'$ is derived by replacing each part in the row by its code and each part in the column by its code and the monochromatic rectangles are replaced by the single value it has. Figure 4 illustrates the algorithm clearly.

Complexity Analysis and Conclusion:
The first part of the second step in the stage 1 of the algorithm is the help needed to run the algorithm in polynomial number of queries. Otherwise the number of queries (complexity) of stage 1 is order $n$. The second stage takes only order $m^2$ steps where $m \leq n$. The third stage of the algorithm is to verify whether $F = F'$. This will be a thorough verification and it will take queries as many as the size of the truth table of $F$. If we believe that the given function is bi symmetric the verification is not needed.

It should be interesting to know if there exists an algorithm for somewhat symmetric functions with three or more $G$'s.
END
11 - 87
ΠΤΙΚ