Verification of Graphical Approach to Realizing Symmetric Boolean Functions with Threshold Logic Elements

Deepshikha
M.Sc., Embedded Systems
School of Computer Science and Engineering
Nanyang Technological University
Singapore

Swarna Kamakshi Jayaraman
M.Sc., Embedded Systems
School of Computer Science and Engineering
Nanyang Technological University
Singapore

***Abstract*—Implementation of the algorithm "A Graphical Interpretation of Realization of Symmetric Boolean Functions with Threshold Logic Elements" authored by C. L. Sheng (1964) and validation results of the claims put forth in the same journal are discussed.**

Keywords*—component; formatting; style; styling; insert (key words)

I. Introduction

As predicted by Gordon Moore, the number of transistors in an integrated circuit has been ever increasing and physicists and engineers have continuously cited that CMOS is approaching a quantum mechanical and physical limit, owing to larger power requirements and increased thermal dissipation with increasing number of transistors. Theoretically, erasing of a single bit of information dissipates heat equivalent to KBTln2, as observed by Von Neumann (1949) and Ralph Landauer (1961). Here, K_B is the Boltzmann constant and T is the room temperature in Kelvin. So, this pushes us to rethink the process of logic synthesis and look for other alternatives which could be as good as CMOS or better, without the physical limitations of the same. Threshold Logic Elements have been recognized as an alternative to traditional logic design. C.L. Sheng in his paper explains a graphical approach to realization of symmetric functions with threshold logic elements. This paper aims at explaining the implementation and validation of this algorithm.

II. DEFINITIONS

A. *Threshold Logic Elements*

A threshold gate is a logic gate which has 'n' input variables x_i, where (i = 1, 2, …n), where each input can take either 0 or 1. Each of the inputs has an associated weight w_i , (i = 1, 2, …n) and a threshold T, such that the output of the gate is 1 when

$$\sum_{i=1}^{n} x_i w_i \geq T \geq 0$$

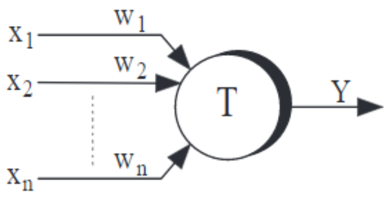


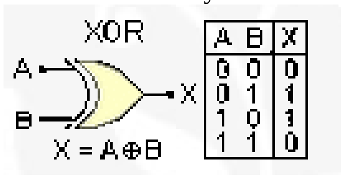
Fig. 1. Threshold logic element

B. *Majority Gates*

Majority gates are a class of threshold gates with each input having unity weight. Here, the threshold is half the number of inputs.

C. *Symmetric Boolean Functions*

A function of n variables f(x1, x2,…xn) is said to be symmetric if and only if it is invariant under any permutation of variables. Example: In an EX-OR Gate, output is 1 for odd number of 1s in the input and 0 for even number of 1s. It does not depend on the order in which they occur.



A	B	X
0	0	0
0	1	1
1	0	1
1	1	0

Fig. 2. Truth table of EX-OR Gate

A symmetric function of n variables is represented by the notation $S_{a1,\ a2,\ \ldots\ ak}(x_1,\ x_2,\ \ldots x_n)$ where S denotes the property of symmetry, a_1, $a_2 \ldots a_k$ denote the number of 1s required in the input for the function to be true and x_1, $x_2, \ldots x_n$ denote all input variables in the function. Hence, EX-OR function can be represented as S_1(a, b) wherein a and b are the input variables and 1 should occur once in the input.

III. Algorithm

A. *Input Specifications*

Input file takes three inputs: Number of input variables, number of constants for which the symmetric Boolean function is one and the set of constants.

B. *Output Specification*

Output is a network of Threshold Logic elements with each element having a threshold and each input to the threshold logic element having an associated weight.

C. *Objective*

The objective of the algorithm is to minimize the network size, i.e. the number of threshold logic elements to the theoretical minimum value.

D. Assumptions

For symmetric Boolean functions to be realized using threshold logic elements, all elements should have the same weight. Because the simplest integral weight is unity, weight of each input variable is assumed to be unity. Hence, the effective weight of j variables $w_j = j*1 = j$.

E. Method 1

Step 1: Parse the input file to get the input data: number of input variables, number of constants for which symmetric Boolean function is 1 (n), and the set of constants, a[i], (i = 1, 2, …n).
Step 2: Identify the discontinuities in the set of constants a[i] and determine the depth of the discontinuities, d.
Step 3: The number of threshold elements required to realize this logic is the (number of depths+1) in the set of constants required to make the output 1.
Step 4: Threshold of the last element is given by the smallest constant value among the set of constants. i.e. t[last_element] = a [0]

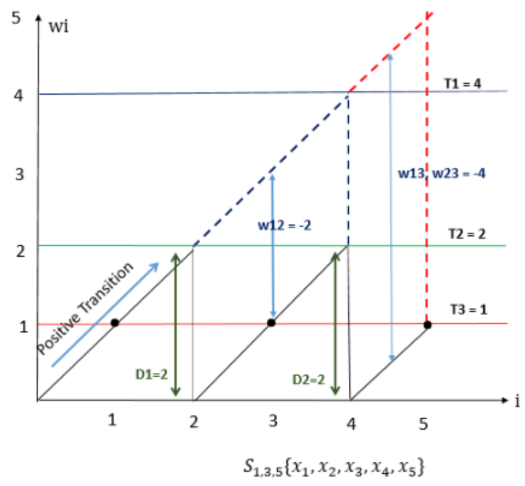


Fig.3. Example Problem $S_{1,2,3}\{x_1,x_2,x_3,x_4,x_5\}$constructed using method 1

Step 5: For other elements, threshold is computed as the t[k] = a[i-1] + 1, where k = 0 to number of threshold elements – 1.
Step 6: Weight of each element w_i = sum of the previous depths.

Step 7: Verify the results, i.e. the threshold and the weight values.

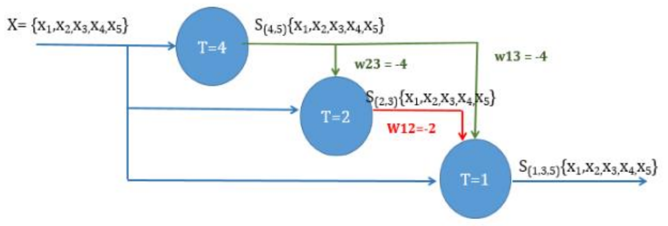


Fig.4. Threshold Logic network for function $S_{1,2,3}\{x_1,x_2,x_3,x_4,x_5\}$ using method 1

On careful analysis of the algorithm, it was identified that the outputs of each element are processed only in the final stage. So, all other weights have not been calculated because they are redundant.

E. Method 2

Step 1: Parse the input file to get input data: number of input variables, number of constants for which the symmetric Boolean function is 1 (n) and set of constants, a[i], (i = 1, 2, …n).
Step 2: If number of 1s in input (n) > number of input variables, an error message should be printed and the algorithm shall not be processed further.
Example: Consider the case of 7 inputs. If the user feeds the number of outputs to make the function true as 12, it is not a possible use-case, as there can never be 12 1's in an input sequence of only 1 variables.
Step 3: Identify the discontinuities in the set of constants a[i] to determine the number of depths.
Step 4: In the graph, for each positive transition, identify the points (x, y_upper) and (x, y_lower).

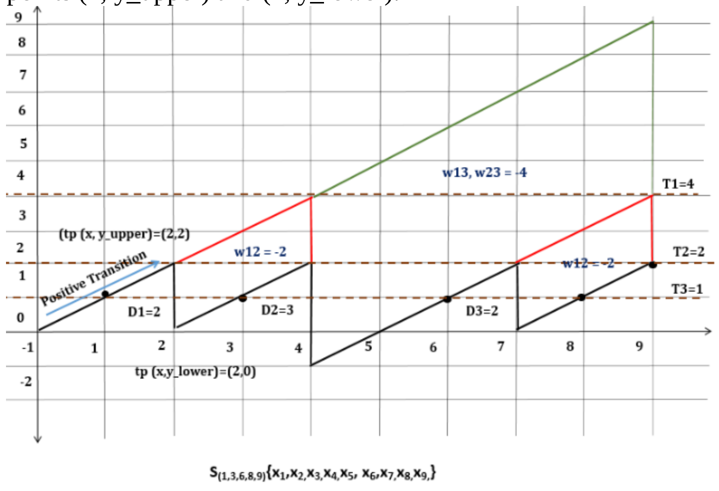


Fig.5. $S_{1,3,6,8,9}\{x_1,x_2,x_3,x_4,x_5,x_6,x_7,x_8,x_9\}$constructed using method 2

Step 5: Theoretically, the minimum number of threshold elements required to realize a symmetric function with its weight line having P positive transitions is given by ceil $(\log_2 P) + 1$.
Step 6: When last point above threshold is not equal to the total number of elements, there shall be an additional depth

considered between the last point and the point representing the total number of elements. This point is assumed to be below threshold.

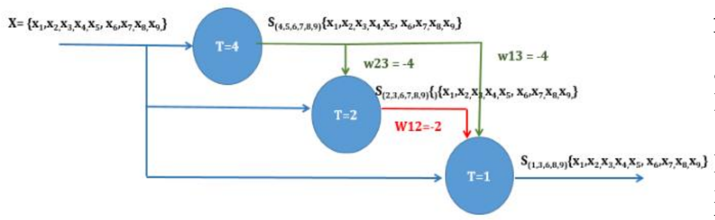
Step 7: The threshold of the last element is taken as the first point above threshold. i.e. the smallest of the set of constants from the input file (a[0]).

Step 8: Threshold of other elements is given by the transition point's y_upper coordinate value.

Step 9: From the depth matrix, the depth value having the maximum frequency is identified. If there is no repetition of depth values, the first depth is assumed as depth with maximum frequency d_{max_freq}.

Step 10: Combine the depths = d_{max_freq} to the next transition points to form a rhombus. After combining, update the new transition point values (x, y_upper), (x, y_lower).

Step 11: Weight at each rhombus, or combined depth = difference between the current folding point and y_lower of the transition point. These relations have been derived based on the translation of the logic based on graph into an algebraic function.


X= {x1,x2,x3,x4,x5, x6,x7,x8,x9}
T=4
S(4,5,6,7,8,9){x1,x2,x3,x4,x5, x6,x7,x8,x9}
w23 = -4
w13 = -4
T=2
S(2,3,6,7,8,9)(){x1,x2,x3,x4,x5, x6,x7,x8,x9}
W12=-2
T=1
S(1,3,6,8,9){x1,x2,x3,x4,x5, x6,x7,x8,x9}

F. *Examples*

Number of Inputs: 9
Set of constants: {1, 3, 6, 8, 9}

Parameters	Algorithm 1	Algorithm 2
Number of threshold elements	4	3
Threshold values	1, 2, 4, 7	1, 2, 4

Weights	a12 = -2
	{a23, a13} = -5
	{a14, a24, a34} = -7
		a12 = -2
		a23 = -5
		a13 = -5

Correctness for all cases	Yes	No. Works with limitations. Cannot be generalized.
Optimal?	Not optimal. Demands as many gates as the number of depths	Achieves the theoretical optimum

IV. RESULTS AND OBSERVATIONS

On validation of the graphical algorithm specified in the journal by C. L. Sheng, it has been identified that Method 1 of realizing symmetric Boolean functions is functionally correct. However, the number of elements required to construct the threshold network is identified to be equal to the number of depths or discontinuities amongst the set of constants. This is not an optimized solution in case of large input sequence with several depths.

Method 2 can achieve a theoretical minimum of ($\log_2 P$) +1 threshold logic elements for a given symmetric boolean function. It has been successfully tested for 15-input, 39-input and 47-input EX-OR gates, the results of which have been manually verified and found to be correct. It is also possible to achieve correct results for more values, if the number of transition points is even.

Comparison of Optimality:
A 49 input EX-OR gate realized using Method 1 used 24 threshold logic elements (= number of depths). In case of Method 2, this was achieved with just 6 threshold logic elements.

Failure Case:
When odd number of transition points are encountered, this algorithm is found to fail and produce incorrect results. Example: 21 input EX-OR.

Method 2, however optimal, isn't a standard solution for any input data. The algorithm is not mathematically rigorous and hence has to be tweaked on a case to case basis to achieve the right answer. Such lack of generalization restricts its automation and hence limits the use of this algorithm for logic synthesis.

V. CONCLUSION

The graphical heuristic to realization of symmetric Boolean functions using threshold logic elements is not a solution to future logic synthesis due to the difficulty in generalization of this method for all cases of inputs. It prompts us to look for alternate approaches to solving this problem. Integer Linear Programming (ILP) maybe exploited to solve the problem of finding weights and thresholds for threshold logic elements. However, optimality of the results is still in question and should be studied carefully. In conclusion, any solution to future logic synthesis shall be general and true for any set of input data.

REFERENCES

[1] C. L. Sheng, A Graphical Interpretation of Realization of Symmetric Boolean Functions with Threshold Logic Elements", vol.EC-14, Issue 1, IEEE Transactions on Electronic Computers , February, 1965