Complexity Estimation and Network Synthesis based on Functional Smoothness and Entropy Measures

M. S. Bhat and H.S. Jamadagni

Abstract-- **We propose a novel method to estimate the complexity of multiple-valued logic functions based on functional smoothness and information theoretic measures. Further, we show that such complexity measures can be used to (a) estimate the area of the circuit implementation and (b) reduce the search space of potential solutions in evolvable network synthesis.**

Index Terms—**Circuit complexity, entropy, multiple-valued logic, network synthesis.**

I. Introduction

Multiple valued logic (MVL) circuit designs have been receiving considerable attention over the last two decades because of their relative advantages of small circuit size, lower power dissipation and higher circuit speed compared to their binary counter parts [1]-[3]. Optimization of area, speed and power dissipation are still being the main constraints to be satisfied in IC designs, an accurate estimation of these parameters would help the designer to perform a design space exploration by trading-off one constraint criterion with another to optimally meet the design objectives. Entropic properties of the functions are related to the complexity of circuits that realize these functions and serve as an estimate for area and power dissipation of the circuit [4] and help in the evolution of the desired network [5]. Since the EDA tools used for synthesizing electronic circuits use libraries optimized for speed. area or power or a combination of these, if the complexity of the function to be realized can be computed in terms of the complexities of library components, it is possible to synthesize the given function optimally using appropriate library components or transform an existing implementation to realize a new functionality.

II. Information Theoretic Measures

Information theoretic approaches have been employed to define ***computational work*** of a Boolean transformation [6]-[7] and to develop power and area estimation techniques [4]. Information, measures have also been used in evolutionary network design [5],[8]. In all these cases, only functional entropy is taken into account and functional smoothness property has not been considered.

A. *Entropy as Information Measure*

Let f and g be n-variable, m-valued logic functions. Non-empty finite sets of all possible values of f and g be denoted by A and B. Let $A=\{a_0,a_1,........a_{m-1}\}$ with probabilities $p_0,.......p_{m-1}$ be a complete set of events of functionf. The entropy $H(f)$ of the function f is defined by,

$$H(f)=-\sum_{i=0}^{m-1}p(f=a_i)\log p(f=a_i) \qquad (1)$$

where $p(f=a_i)$ is the probability of f taking a value a_i in the partition and logarithm is of base m. The *joint entropy* $H(f,g)$ of f and g is defined by

$$H(f,g)=-\sum_{i=0}^{m-1}\sum_{j=0}^{m-1}p(f=a_i,g=b_j)\log p(f=a_i,g=b_j) \qquad (2)$$

The conditional ***entropy*** $H(f\,|\,g)$ off given g be

$$H(f\,|\,g)=H(f,g)-H(g) \qquad (3)$$

***Example 1**. For the MAX function of table 1, the input and output entropy are as given below.*

Input Entropy: $H(X)=-16.\tfrac{1}{16}.\log_4\tfrac{1}{16}=2bits$

Output Entropy:

$$H(f)=-\begin{bmatrix}\tfrac{1}{16}.\log_4\tfrac{1}{16}+\tfrac{3}{16}.\log_4\tfrac{3}{16}\\+\tfrac{5}{16}.\log_4\tfrac{5}{16}+\tfrac{7}{16}.\log_4\tfrac{7}{16}\end{bmatrix}=0.87bit$$

B. *Functional Smoothness*

The smoothness of an MVL function is measured from the number of transition-it ***makes*** when one of the variables is cycled through 0 to m-1 keeping all other variables fixed. There will be 2-transition vectors, each of length m, for a 2-variable, m-valued function, ***3m*** vectors for a 3 variable function and so on. The entries in the vector correspond to the number of output transitions of the function along the

M. S. Bhat is with Centre for electronics Design and Technology, Indian institute of Science, Bangalore - 560012, India (telephone: 080-23600810. e-mail: msbhat@cedt.iisc.ernet.in

H. S. Jamadagni is with Centre for electronics Design and Technology, Indian institute of Science, Bangalore - 560012. India (telephone: 080-23600810, e-mail: hsjam@cedt.iisc.ernet.in

corresponding row/column, with a maximum value of *m-l*. We define a term ***transition density*** (*T_D*) for a function, which is the ratio of average number of transitions to the maximum number (*m-l*) of transitions. *T_D* is a bounded, normalized value in the range [0,1]. A function is said to be **smooth** if *T_D* of that function is low.

TABLE 1 2-VARIABLE, 4-VALUED MAX FUNCTION

x1 / x2	0	1	2	3	T1	Avg.
0	0	1	2	3	3	1.5
1	1	1	2	3	2	
2	2	2	2	3	1	
3	3	3	3	3	0	
T2	3	2	1	0	T_D	
Avg.	1.5				0.5	

Example 2 *For the 2-variable, 4-valued MAX function shown in table 1, the transition vectors T1 and T2 corresponding to variables x1 and x2 are [3210] and [3210] respectively, Vector T1 corresponds to the transitions along the rows of the truth table, with 3 transitions in the first row, 2 in the second row and so on. Similarly, T2 corresponds to the transitions along the columns of the truth table. T_D for this case is 0.5*

III. ESTIMATION OF COMPLEXITY

Entropy gives an idea of the randomness present in the behavior of the function and. in turn, the complexity of generating/realizing functions [6]-[7]. This notion has been used for estimating area (literal ***count)*** and power dissipation in Boolean functions [4],[9]. However, taking entropy alone in complexity computation will result in large inaccuracies. We show that by taking transition density also into consideration, we get a better estimate of the complexity. Let us consider two different scenarios of implementing MVL circuits and examine their implementation complexities.

Case 1: Functions are expressed in minimum sum-of-products (SOP) form and are realized using window literals and three stage self-restoring logic architecture of [10]. In this case, cost of the function is measured as the number of literals present in the expression.

Consider the following functions f_1 and f_2

$$\textit{Function}\ f_1 = {}^2x_1^3\ {}^0x_2^1 + 2\ {}^0x_1^1\ {}^0x_2^0\ \mathrm{t}$$
$$3({}^1x_1^1\ {}^1x_2^2 + {}^2x_1^3\ {}^3x_2^3)$$

cost (f_1) = 8
Entropy (f_1) = ***0.95***

$$T_D(f_1) = \frac{\textit{Avg.No.of}\ \textbf{Tansitions}}{\textit{Max.No.of Transitions}}$$
$$= \left(\frac{1.5+1.75}{2}\right) * \frac{1}{3} = 0.542$$

TABLE 2 FUNCTION f_1

x1 / x2	0	1	2	3	T1	Avg
0	2	2	1	1	1	1.5
1	0	3	1	1	2	
2	0	3	0	0	2	
3	0	0	3	3	1	
T2	1	2	2	2	$T_D(f_1)$	
					0.542	

$$f_2 = {}^2x_1^2\ {}^0x_2^0 + {}^3x_1^3\ {}^1x_2^1 + {}^0x_1^1\ {}^2x_2^2$$
$$+2\,({}^0x_1^0\ {}^0x_2^0 + {}^1x_1^1\ {}^1x_2^1)$$
$$+\ 3\,({}^3x_1^3\ {}^0x_2^0 + {}^2x_1^2\ {}^1x_2^1 + {}^1x_1^1\ {}^3x_2^3 + {}^3x_1^3\ {}^3x_2^3)$$

Cost (f_2) = 18
Entropy (f_2) = 0.95

$$T_D(f_2) = \frac{2.625}{3} - 0.875$$

TABLE 3 FUNCTION f_2

x1 / x2	0	1	2	3	T1	Avg
1	0	2	3	1	3	2.5
2	1	1	0	0	1	
3	0	3	0	3	3	
T2	3	3	2	3	$T_D(f_2)$	
Avg.	2.75				0.875	

If we assume that the input MVL blocks of f_1 and f_2 have similar literal generating circuits and the output MVL blocks are identical, then the binary logic will have different complexities to account for different ***costs***. Similar to the Boolean case [4], considering cost as a function of entropy, we get,

$$C_{f1}\ \text{a}\ H(f_1) \text{ and } C_{f2}\ \text{a}\ H(f_2) \qquad (4)$$

where C_f is the cost of function f.

However, since the entropies of both the functions are identical, functional complexity (and hence cost) as a measure of entropy alone will give identical values, which clearly is not the case, Now if we take transition density also into account in computing cost, we can write,

$$\bar{C}_f\ \boldsymbol{a}\ n.T_D(f).H(f) \qquad (5)$$

where ***n*** is the number of inputs. This gives us,

$$\bar{C}_{f1}\ \boldsymbol{a}\ 2\,(0.542)(0.95) \doteq 1.03$$

$$\bar{C}_{f2} \alpha\ 2(0.875)(0.95) = 1.66$$

This shows that f_2 has a higher complexity than f_1, which is in conformity with the cost when measured as the number of literals in sum-of-product form,

Case 2: Functions are expressed in tabular form and their cost is measured in terms of universal literals as in [11]. We used the cost table given in [12] for the four-valued universal literals in estimating the cost of the given function. Example 3 justifies the use of transition density in computing the function complexity.

Example 3 Consider functions f and g given in table 4 and 5

TABLE 4 FUNCTION *f*

x2 \ x1	0	1	2	3	T1
0	2	2	3	3	1
1	2	1	0	3	3
2	0	1	1	1	1
3	0	2	2	2	1
T2	1	2	3	2	$T_D(f)$ 0.58
H(f)	0.97				

Using universal literals, logic function f can be expressed as [12].

$$f(x1,x2) = <0133>(x1)<3000>(x2)$$
$$+<2103>(x1)<3300>(x2)$$
$$+<0233>(x1)<0012>(x2)$$

Entropy, H(f) = 0.97
*Cost(f) = (5+8+14+9+4+4+5*3)* = 52

TABLE 5. FUNCTION *g*

x2 \ x1	0	1	2	3	T1
0	1	[illegible]	[illegible]	[illegible]	3
1	1	[illegible]	[illegible]	[illegible]	0
2	1	2	0	2	3
3	0	1	3	0	3
T2	1	3	3	3	$T_D(g)$ 0.792
H(g)	0.89				

$$g(x1,x2) = <0130>(x1)<0103>(x2)$$
$$+<1202\ 3\ (x1)<2120\ 3\ (x2)$$

Entropy, H(g) = 0.89
*Cost(g) = (14+15+16+17+5*3)* = 77

If we consider only entropy for computing the complexity of functions, then function *g* would be less complex than function *f* since $H(g) < H(f)$. But that is not the case since *cost(g) > cost(f)*. Considering transition density along with entropy, we get,

$$C_f\ \alpha\ n.T_D(f).H(f) = 2.(0.58)(0.97) = 1.125$$

and

$$C_g\ \alpha\ n.T_D(g).H(g) = 2.(0.792)(0.89) = 1.4\ .$$

This new complexity measure of *f* and *g* is in conformity with the cost indicated above, which was computed as per [12].

The four functions, f_1, f_2, *f* and *g*, are implemented using current-mode self-restoring logic architecture of [10] and the number of transistors used are listed in table 6. We see that the transistor count increases with complexity. The results from the above two cases indicate that, by considering transition density along with functional entropy gives better estimates of implementation complexity of MVL functions.

TABLE 6 TRANSISTOR COUNTS FOR THE EXAMPLE FUNCTIONS USED IN THIS PAPER

Function name	*Entropy*	*T_D*	*Complexity*	*Number of transistors used*
f1	0.95	0.542	1.03	73
f2	0.95	0.875	1.66	149
f	0.97	0.58	1.125	111
g	0.89	0.792	1.3	127

IV. FUNCTIONAL SMOOTHNESS IN NETWORK SYNTHESIS

In the case of evolutionary network synthesis [10], in order to implement a desired function *f*, an iterative strategy based on information theoretic measures is used to arrive at a set of possible solutions. We consider a part of the problem in the synthesis procedure, wherein if a solution exists, then the function *f* can be implemented using the evolved network and a constant function.

In an *m*-valued, *n*-variable MVL logic, there exist m^{m^n} functions. From this, the solution set for implementing *f* is iteratively generated by considering entropy of the function [10]. Let us denote the possible set of solutions at *i*-th iteration as $G_i = [g_i^1, g_i^2, ...]$. A function $g_i^k \in G_i$ if,

$$H(g_i^k) \geq H(f) \qquad (6)$$

It is assumed that if a network *Net** implements the function $g \in G_i$, then it is possible to transform Net* into a network with desired output *f* through a regular procedure [8]. Any function $g \in G_i$ is a solution if $H(f\,|\,g) = 0$. Under this condition, there exists a logic function φ such that $f = \varphi(g)$. Although, the search space for a solution is reduced from m^{m^n} to G_i using (6), the actual number of possible solutions is $|G_i|$, which could still be very large. Fig. 1 shows the relation between the different sets of functions. Let us now consider the transition vectors in refining the solution space off. The new solution set K_i for *f* is given by $K_i = [k_i^1, k_i^2, ...]$. Let $k = k_i^l \in K_i$ be a function in the solution set, with transition vectors $T_k^1, T_k^2, T_k^3, ...$ The function *k* qualifies to be a solution off only if the corresponding transition vectors of k and f are either identical or symmetric and $H(k) \geq H(f)$. The condition imposed by the transition vectors eliminate a large number of functions from G_l which otherwise would have been considered as possible solutions, With this new constraint, we have

$$|K_l| < |G_l| \; and \; K_l \subset G_l \qquad (7)$$

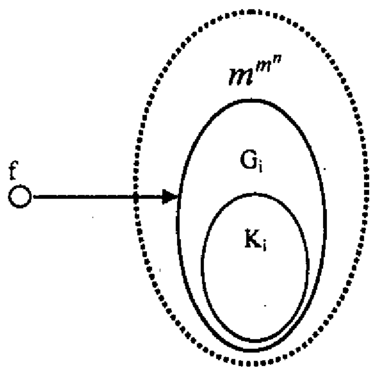

Fig. 1 Sets of Functions

Example 4 *Consider the ternary 2-variable functions f, g_1, g_2 and g_3 shown in table 7 (a) – (d), where f is the target function and g_1, g_2 and g_3 are the functions for which network implementations are available. The problem is to find, among the networks for g_1, g_2 and g_3, which one can be transformed to realize f?*

Fifth column and fifth row in each table gives the transition vectors of the respective functions corresponding to the two variables. Entropies of the four functions are shown at the bottom of each table. $H(g_1)$ is less than $H(f)$ and hence can not be a candidate for implementing f (as per (6)). $H(g_2)$ and $H(g_3)$ ore equal to $H(f)$ and hence lie in the solution space G_l.

Both g_2 and g_3 can be solutions since $H(f|g_2) = H(f|g_3) = 0$. However, f we consider the transition vectors, f and g_2 have identical vectors, where as. g_3 has different vectors. Hence, a simple permutation of the g2 output would give function f and the corresponding permutation being $\varphi = [120]$ and $f = \varphi(g_2)$. Since the transition vectors of g3 are different from that of f, g_3 cannot be a candidate solution. Thus,

$$g_2, g_3 \in G_l \; and \; g_2 \in K_l, \; but \; g_3 \notin K_l$$

The search space for a potential solution can thus be reduced from G_l to K_l by considering the transition vectors. The resultant network is shown in figure 2.

One immediate observation is that when the transformation φ is a simple permutation, then the transition vectors of the two functions are either identical or symmetric. However, the converse is not true. A permutation is a bijective mapping from f to g. Therefore, when transition vectors are identical or symmetric, there might exist a mapping from g to f. This property results in the reduction of solution space when compared to the solution space obtained by considering functional entropy alone. We show in example 5, the meaning of symmetry in transition vectors and the synthesis off given g.

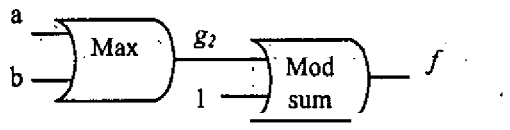

Fig. 2 Synthesized Network for f

Example 5 Consider tsum and tprd operators used in MVL circuit synthesis. The operators for 2-variable, 3-valued case are defined as follows and are shown in table 8(a) and (b) below.

$$tsum(x1,x2) = min(x1 + x2, m\text{-}1)$$
$$tprd(x1,x2) = max(x1 + x2 - (m\text{-}1), 0)$$

The transition vectors for the two functions are,

tsum :[210], [210] and tprd : [012], [012].

The transition vectors of the two functions are symmetric in the sense that one set of vectors can be obtained from the other by inversions of the entries of the first vector as $\bar{x} = (m-1) - x$. Using this, the function tprd can be obtained from tsum by the transformation,

$$tprd(x_1, x_2) = \overline{tsum(\bar{x}_1, \bar{x}_2)}, \; where \; \bar{x} = (m-1) - x$$

This is shown in figure 3.

This shows that we can find a mapping between two functions even when their respective transition vectors are not identical, However, this is not true for every case because, there are functions with same entropies and identical or symmetric transition vectors, but yet they cannot be mapped from one to other. On the other hand, by considering transition vectors, the search space for mapping can be reduced.

TABLE 7(a)

f	0	1	2	T1
0	1	2	0	2
1	2	2	0	1
2	0	0	0	0
T2	2	1	0	0.5
H(f) = 0.85				

2	2	2	2	
T2			0	0.33
H(g1) = 0.62				

T2	2	1	0	0.5
H(g2) = 0.85				

(d)

g_3	0	1	2	T1
0	2	1	0	2
1	1	1	0	1
2	1	1	0	1
T2	1	0	0	0.42
H (g3)= 0.85				

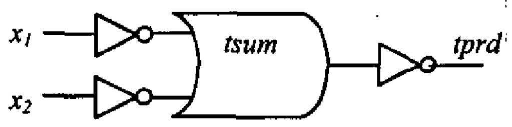

Fig. 3 Synthesis of *tprd* from *tsum*

Whenever the transition vectors of a function g are either identical or symmetric with respect to a target function f, then function g falls in the set K_i of potential solutions for f. By considering transition vectors in evolutionary network synthesis, it is possible to reduce the search space for potential solutions,

TABLE 8(a) FUNCTION *TSUM*

T2	*2*	*1*	*0*	*0.5*
H(tsum) = 0.77				

TABLE 8(b) FUNCTION *TPRD*

tprd	*0*	*1*	*2*	*T1*
0	*0*	*0*	*0*	*0*
1	*0*	*0*	*1*	*1*
2	*0*	*1*	*2*	*2*
T2	*0*	*1*	*2*	*0.5*
H(tprd) = 0.77				

V. CONCLUSIONS

In this paper, we show the effectiveness of using functional smoothness with information theoretic measures in estimating the functional complexity. We investigated the utility of transition density in estimating the cost of circuit realization in *two* different architecture styles. The results have shown that the transition density indeed improves the cost estimates. We have also shown that using transition vectors, the search space for potential solutions in evolutionary network synthesis can be reduced.

VI. REFERENCES

[1] S.L.Hurst, " Multiple-valued logic - Its status and future", *IEEE trans. Computers*, vol. *C-33*, pp. 1160-1179, Dec.1984.

[2] K.C.Smith, 'Multiple-valued logic - A tutorial and appreciation," IEEE *Computer*, vol.21, pp.17-27, Apr.1988

[3] S. Kawahito, et al., "A 32x32 bit Multiplier using Multiple-valued MOS Current-made circuits", IEEE *J. Solid State Circuits*, vol. *23*, no. 1, pp.124-132, Feb.1988

t41 Hwang C. H. and Wu A. C. H., *"An Entropy* Measure for Power Estimation of Boolean Functions", Proc. *of Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 101-106, *1997.*

[5] Luba T., Moraga C., Yanushkevich S., Opoka M. and Shmerko V., ."Evolutionary Multi-Level Network Synthesis in given Design Style", *Proc.* of *30th Intl. Conf. On Multiple-Yalued* Logic. pp. 253-258, 2000.

[6] Hellerman L., "A Measure of Computational Work", *IEEE Trans.* on *Computers*, vol. C-21, pp. 439-446, May 1972.

[7] Cook R. W. and Flynn M. J., "Logical Network Cost and Entropy", *IEEE Trans.* on *Computers*, vol. C-22(9), pp. 823-826, September 1973.

[8] Cheushev V., Yanushkevich S., Shmerko V., Moraga C.. and . Kolodziejczyk J., "Information Theory Methods for Flexible Network Synthesis", *Proc. of 31st International Symposium on Multiple-Valued Logic*, pp. 201-206,2001.

[9] Nemani M. and Najm F. N., 'Towards a high level power estimation capability", *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 15, No. 6, June 1997.

[10] Teng H. Y. and Bolton R. J., "A self-restored current-mode CMOS multiple-valued logic design. architecture," *Proceedings of* the *7th IEEE Pacific* Rim *Conference on Communications, Computers and Signal Processing*, pp. 436-439, Aug. 1999.

[11] Lie K. and Vranesic Z. G., "Towards the realization of 4-valued CMOS circuits", Proc. of 22nd International Symposium on Multiple-Valued Logic, pp.104-110, 1992.

[12] Hozumi T., Kakusho O. and Hata Y.. "The output permutation far the multiple-valued logic minimization with universal literals", Proc. of 29th International Symposium on Multiple-Valued Logic, pp.105-109.1999.