AN ENTROPY MEASURE FOR THE COMPLEXITY OF MULTI-OUTPUT BOOLEAN FUNCTIONS

Kwang-Ting Cheng *and* Vishwani D. Agrawal

AT&T Bell Laboratories,
Murray Hill, NJ 07974

ABSTRACT - **The complexity of a Boolean function can be expressed in terms of computational work. We present experimental data in support of the *entropy* definition of computational work based upon the input-output description of a Boolean function. Our data show a linear relationship between the computational work and the average number of literals in a multi-level implementation. The investigation includes single-output and multi-output function with and without don't care states. The experiments, conducted on a large number of randomly generated functions, showed that the effect of don't cares is to reduce the computational work. For several finite state machine benchmarks, the computational work gave a good estimate of the size of the circuit. Finally, circuit delay is shown to have a non-linear relationship to the computational work.**

1. INTRODUCTION

In information theory, the statistical behavior of a digital signal is characterized by its entropy. A digital circuit transforms its input signal into the output signal. In general, the entropy (or information) may not be preserved in such a transformation. In a thermodynamical sense, the transformation performed by a circuit is analogous to computational work. A computing element or Boolean gate possesses some computing power. To perform a given amount of work over some given time we need certain number of gates. Of course, tradeoffs between power and time may be possible. In this paper, our motivation is to examine the entropy measures in view of the current logic synthesis methodology. We believe, the statistical formulation may have applications to the problems of high-level synthesis, Boolean function specification, state assignment, logic minimization, timing optimization, and testability.

In 1972, Hellerman [1] proposed a definition of computational work based on entropy. About the same time, Cook and Flynn [2] also made a similar proposal showing that the entropy formulation did, in fact, explain the cost behavior of Boolean functions that was reported in an earlier paper by Kellerman [3]. Among these, Hellerman's paper gives a more complete analysis. Although the effect of don't care states on the complexity of a Boolean function was observed in the experimental work of Kellerman, a specific relation for computational work involving don't cares appeared only in a paper by Pippenger [4]. Most of the reported work deals with single output functions.

Among applications, the use of information theory for generating tests was proposed by Agrawal [5]. The state assignment problem was discussed by Lala [6] and others [7].

In this paper, we generalize the entropy formulation to multi-output functions. The relationship between entropy and the average amount of logic required for implementing a combinational network is investigated through a series of experiments. For single output functions, the relationship between the delay of the network and the entropy is also studied.

2. ENTROPY OF A MULTIPLE-OUTPUT FUNCTION

Fully Specified Functions. For a multiple-output, fully specified Boolean function f with n input and m output signals, there are 2^n possible input vectors and 2^m output vectors. For each output vector O_i, the probability that $f = O_i$ is:

$$Pr(f = O_i) = P_i = \frac{n_{O_i}}{2^n} \tag{1}$$

where n_{O_i} is the number of times O_i appears in the truth table of f. The entropy H of f is a function of $\boldsymbol{P} = \{P_1, P_2, \ldots, P_{2^m}\}$ and is defined as:

$$H(\boldsymbol{P}) = \sum_{i=1}^{2^m} P_i \cdot \log_2 \frac{1}{P_i} \tag{2}$$

The value of $H(\boldsymbol{P})$ is always between 0 and m. When $P_1 = P_2 = \ldots\ldots = P_{2^m} = 2^{-m}$, $H(\boldsymbol{P}) = m$. When $P_i = 1$ and $P_j = 0$ for all $j \neq i$, $H(\boldsymbol{P}) = 0$.

The entropy of a single output function is given by

$$H(P_1) = P_1 \cdot \log_2 \frac{1}{P_1} + (1 - P_1) \cdot \log_2 \frac{1}{1 - P_1} \tag{3}$$

where P_1 is the fraction of 1's in the output column of the truth table.

Partially Specified Functions. Suppose a partially specified function f_d has n_d don't cares in its truth table. The probability that $f_d = O_i$ is then modified as:

$$Pr(f_d = O_i) = P_i = \frac{n_{O_i}}{2^n - n_d} \tag{4}$$

The formula for entropy of f_d will be the same as that for a fully specified function, i.e., Eqn. (2).

3. COMPUTATIONAL WORK AND CIRCUIT AREA

Computational work in an n-input combinational function is defined as [1]:

$$\textit{Computational Work} = 2^n \cdot H(\boldsymbol{P})$$

It is presumed that the circuit area, which is proportional to the number of gates or logic devices, will be proportional to the computational work. In the following, we will use the literal count, denoted by L, in a multi-level minimized form as a measure of the circuit area.

Single Output Functions. The average amount of logic required for implementing an n-input Boolean function can be expressed as [4]:

$$L = (1 - d) \cdot K(n) \cdot H(P_1) \tag{5}$$

where d is the fraction of don't cares and $K(n)$ is a function of the number of inputs and is independent of P_1. We provide experimental results using the multi-level logic synthesis system MIS [8] to verify this formula. We use the number of literals

in the multi-level implementation as a measure of the amount of required logic.

Fully specified function: For a given P_1 (probability of $f = 1$), we randomly generated 100 functions. The truth table description of these functions was minimized by the MIS program for a multi-level implementation. Figure 1 shows the relationship between the number of literals and P_1 for 6-input single-output functions. The points on the three solid curves are the maximum, average and minimum number of literals each obtained from 100 functions. The dotted curve is the entropy function normalized to fit the measured average literal count.

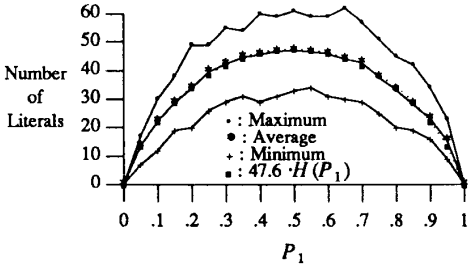

Fig. 1: 6-input, single-output functions.

Figure 2 shows the average literal count versus entropy data for 6, 7, 8 and 9-input functions. Each point represents an average over 200 randomly generated functions. A near-linear relationship between L and $H(P_1)$ is evident. Since these functions are fully specified, $d = 0$, and according to Eq. (5), the slope should be $K(n)$. The measured slope for each n is given in Table 1. Also, we notice that $K(n)/K(n-1)$ is close to 2. This confirms the previous observations [9]:

$$K(n) \approx k \cdot 2^n \qquad (6)$$

for some constant k.

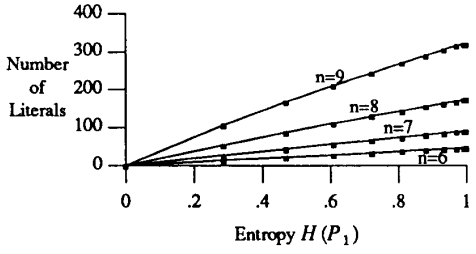

Fig. 2: fully specified n-input functions.

Table 1 - Values of $K(n)$		
n	$K(n)$	$K(n)/K(n-1)$
6	47.59	-
7	93.05	1.96
8	179.05	1.92
9	338.05	1.89
10	624.61	1.85

Partially specified function: Don't cares in a combinational network specification are known to reduce the amount of logic. Figure 3 gives the results for 6-input single-output functions. The five sets of data correspond to the functions with 0%, 20%, 40%, 60% and 80% don't cares in the truth table. Each point is an average of 100 random functions. Figure 4 gives similar data for 7-input functions. The average ratio of the literal count (L) to entropy (H) for various fractions (d) of don't cares is listed in Table 2. The closeness of the normalized value of (L/H) to $1-d$ provides a confirmation of Eq. (5).

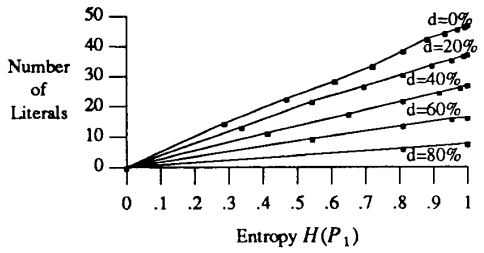

Fig. 3: 6-input single-output functions with don't cares.

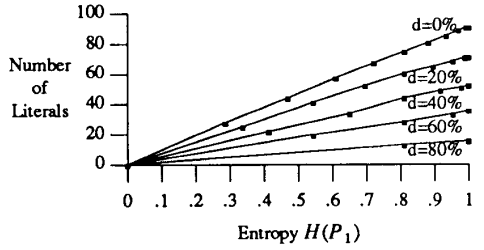

Fig. 4: 7-input single-output functions with don't cares.

Table 2 - Ratio of literal count to entropy (L/H)				
Don't cares	n=6		n=7	
100×d %	L/H	Normalized L/H	L/H	Normalized L/H
0%	47.6	1.00	93.1	1.00
20%	38.1	0.80	73.6	0.79
40%	27.0	0.57	53.0	0.57
60%	16.8	0.35	35.5	0.38
80%	7.7	0.16	15.8	0.17

Multioutput Functions. We again consider two cases. In the first case, we assume that the functions are fully specified and in the second case, the outputs are specified only for a subset of the input vectors.

Fully specified function: Figure 5 shows the relationship between the literal count and the entropy as obtained from 1,200 randomly generated 7-input 2-output fully specified functions. The entropy, $H(P)$, for the functions was computed from Eq. (2). The functions were minimized by MIS using a standard script. Figure 6 shows similar data obtained from 1,500 randomly generated 7-input 3-output functions. Figures 7 and 8 summarize the data obtained from randomly generated 8-input, 4-output functions and 9-input, 7-output functions, respectively.

An interesting inference from these results is that, irrespective of the number of inputs and outputs, the average amount of hardware (literal count) is always given by the computational work formula. For multi-output functions, from Eqs. (5) and (6), we get

$$L(n,d,P) = (1-d) \cdot k \cdot 2^n \cdot H(P) \qquad (7)$$

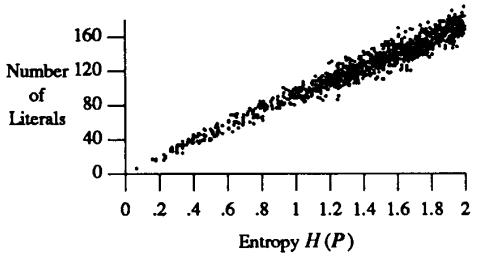

Fig. 5: 7-input 2-output random functions (1200 samples).

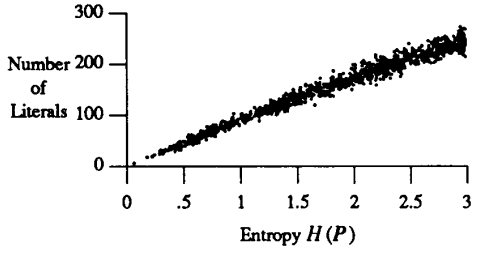

Fig. 6: 7-input 3-output random functions (1500 samples).

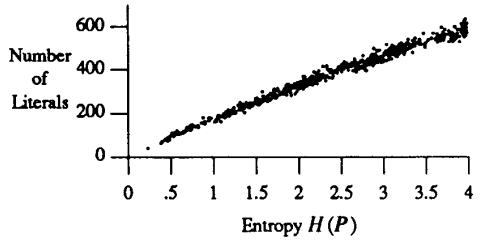

Fig. 7: 8-input 4-output random functions (750 samples).

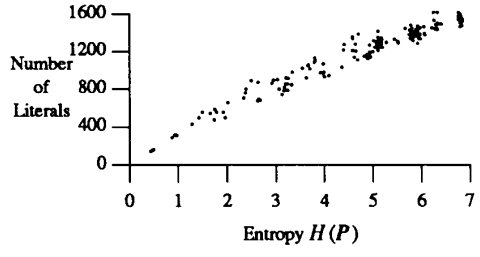

Fig. 8: 9-input 7-output random functions (100 samples).

where n is the number of inputs and the entropy $H(P)$ is given by Eq. (2). The fraction d of don't cares is 0 in the fully specified case. The constant k depends on the units of L (literals, gates, etc.). As an example, consider $H(P)$=1.8 for which Fig. 5 gives an approximate value L=160. This is also the value of L in Fig. 6 for $H(P)$=1.8. Both cases correspond to 7-input functions. For 8-input functions, Fig. 7 gives approximately 320 literals. Similarly, for 9-input functions, we observe about 640 literals in Fig. 8.

A closer examination of these results shows that the number of literals to entropy relation slightly deviates from linearity. That is, the L/H ratio decreases slightly when the entropy H increases. The average L/H ratio for different ranges of H is listed in Table 3. Since the average L/H ratio on the range of H between 0 and 1 (third column) is very close to the values of $K(n)$ for single output function shown in Table 1, we conclude that $K(n)$ is independent of the number of outputs and is simply a function of the number of inputs. The reduced slope for higher entropy may be due to greater sharing between the multiple output functions.

Table 3 - Average L/H Ratio for Multioutput Functions

Func.	Average L/H				
	Overall	$0 \le H \le 1$	$1 < H \le 2$	$2 < H \le 3$	$H > 3$
7-in, 2-out	94.10	96.5	93.116	-	-
7-in, 3-out	87.2	93.4	89.4	84.3	-
8-in, 4-out	161.8	186.4	169.8	161.0	152.2
9-in, 7-out	243.7	342.6	320.2	254.4	241.2

Partially specified function: Figure 9 illustrates the effect of don't cares on 7-input 3-output functions. Five sets of data are shown in this figure. These correspond to fully specified functions (shown by · in Fig. 9), and the functions having 20% (■), 40% (Δ), 60% (□) and 80% (+) don't cares. Each set consists of 650 random samples. All functions were minimized by MIS. The average L/H ratios are listed in Table 4. These results justify that the L/H ratio is proportional to $1-d$ where d is the fraction of don't cares (Eq. (7)).

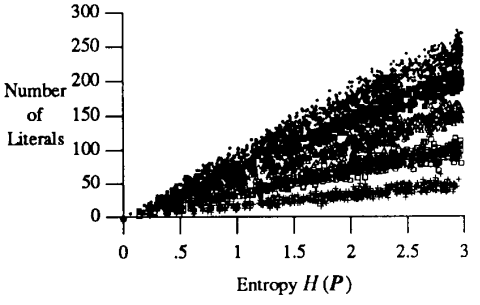

Fig. 9: 7-input 3-output functions with don't cares d = 0 (·), 0.2 (■), 0.4 (Δ), 0.6 (□) and 0.8 (+).

Table 4 - L/H Ratio for functions with don't cares

Don't cares(%) $100 \cdot d$	Average L/H	Normalized L/H
0%	87.2	1.00
20%	72.6	0.83
40%	54.6	0.62
60%	36.4	0.41
80%	17.2	0.20

Computational Work in Finite State Machine Benchmark Circuits. A finite state machine is usually specified by its state

table. Different state assignments will result in different combinational functions. However, the entropy of these functions is independent of the assigned state codes. Thus, the entropy of a state machine can be computed directly from the state table. We used twelve benchmark finite state machines [10] in our experiment (Table 5). Don't cares in the combinational functions vary from 0% to 91.6%. The literal count $L(n,d,P)$ was computed from Eq. (7) using $K(n)$ and $H(P)$ given in Table 5. Entropy $H(P)$ was computed from Eq. (2) and $K(n)$ was interpolated from Tables 1 and 3. Each finite state machine was implemented using a random state assignment and the state assignment obtained from Mustang [11]. The number of literals in these implemenations are listed in columns labeled *Random* and *Mustang*, respectively. In most cases, the literal count $L(n,d,P)$, obtained from computational work, gives a good approximation.

Table 5 - Computational Work for Benchmark FSM

FSM Name	in-puts/ out-puts	Don't care (%) 100× d	$K(n)$	Entropy $H(P)$	Literal count $L(n,d,P)$/ **Random/** Mustang
dk15	5/7	0.0	23	3.97	92/96/92
dk16	7/8	15.6	80	6.03	400/383/307
dk17	5/6	0.0	23	3.87	89/83/84
dk27	4/6	12.5	12	3.18	33/29/29
dk512	5/7	6.3	23	4.34	93/82/84
bbtas	5/5	25.0	23	2.99	51/35/36
beecount	6/7	20.3	47	2.35	88/56/50
red	7/6	70.3	84	2.39	60/60/65
modulo12	5/5	25.0	23	3.59	62/42/43
malati	9/9	91.6	338	4.25	120/193/181
s8	7/4	84.4	84	2.26	30/31/27
lion9	6/5	60.9	47	3.37	62/49/22

4. TIMING

Figure 10 shows the relationship between the delay of the implemented single-output Boolean functions and entropy. The delay is estimated using the unit-fanout delay model in MIS, i.e., each gate is assumed to have one unit of delay and each fanout adds 0.2 unit. Every point in Fig. 10 represents an average of 30 random samples. The observed linear relationship between the square of delay (D^2) and entropy (H) can be expressed as follows:

$$D^2 = R(n) \cdot H$$

Table 6 summarized the slopes of the straight lines fitted to the data of Fig. 10. Since $R(n)/R(n-1)$ is approximately 2, we assume that $R(n)$ is proportional to 2^n. Thus

$$D = k_D \cdot 2^{\frac{n}{2}} \cdot H^{\frac{1}{2}} \qquad (8)$$

where k_D is a constant of proportionality.

5. CONCLUSION

We have presented a statistical measure of the complexity of Boolean functions. We measure computational work based on the information theoretic entropy of the output signals. Computational work is shown to have a direct relation to the hardware needed to implement the function. Circuit delay is also related to the computational work. The computational work is reduced in direct proportion of don't cares in the specification of a function. Partially specified functions are known to require less hardware compared to the fully-specified functions with the same number of input variables. Thus, it is possible to estimate the area and delay of a circuit from its functional description. Statistical measures have not been used in the modern logic synthesis. We believe, there are relevant applications.

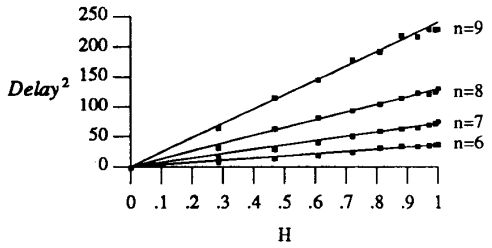

Fig. 10: Delay and entropy of single output functions.

Table 6 - Ratio of $Delay^2/H$

n	$R(n)=Delay^2/H$	$R(n)/R(n-1)$	k_D
6	35.5	-	1.11
7	72.0	2.02	1.13
8	130.2	1.81	1.02
9	242.5	1.86	0.95

References

1. L. Hellerman, "A Measure of Computational Work," *IEEE Trans. Computers*, vol. C-21, pp. 439-446, May 1972.
2. R. W. Cook and M. J. Flynn, "Logical Network Cost and Entropy," *IEEE Trans. Computers*, vol. C-22, pp. 823-826, September 1973.
3. E. Kellerman, "A Formula for Logical Network Cost," *IEEE Trans. Computers*, vol. C-17, pp. 881-884, September 1968.
4. N. Pippenger, "Information Theory and the Complexity of Boolean Functions ," *Math. Syst. Theory*, vol. 10, pp. 129-167, 1977.
5. V. D. Agrawal, "An Information Theoretic Approach to Digital Fault Testing," *IEEE Trans. Computers*, vol. C-30, pp. 582-587, August 1981.
6. P. K. Lala, "An Algorithm for the State Assignment of Asynchronous Sequential Circuits," *Electronics Letters*, vol. 14, pp. 199-201, 1978.
7. C. R. Edwards and E. Eris, "State Assignment and Entropy," *Electronics Letters*, vol. 14, pp. 390-391, 1978.
8. R. K. Brayton et al, "MIS: A Multiple Level Logic Optimization System," *IEEE Trans. CAD*, vol. CAD-6, pp. 1062-1081, November 1987.
9. K. Mase, "Comments on *A Measure of Computational Work* and *Logical Network Cost and Entropy*," *IEEE Trans. Computers*, vol. C-27, pp. 94-95, January 1978.
10. R. Lisanke, "Finite state machine benchmark set," *Preliminary benchmark collection*, September 1987.
11. S. Devadas et al, "MUSTANG: State Assignment of Finite State Machines Targeting Multi-Level Logic Implementations," *IEEE Trans. CAD*, vol. CAD-7, pp. 1290-1300, December 1988.