

The optimal sequence compression

(Preliminary unfinished version)

A. E. Andreev

LSI Logic Corporation, AE-187
1551 McCarthy Blvd.
CA 95305 Milpitas, USA
andreev@lsil.com

12-25-2005

Abstract

This paper presents the optimal compression for sequences with undefined values.

Let we have $(N - m)$ undefined and m defined positions in the boolean sequence \vec{V} of length N . The sequence code length can't be less than m in general case, otherwise at least two sequences will have the same code.

We present the coding algorithm which generates codes of almost m length, i.e. almost equal to the lower bound.

The paper presents the decoding circuit too. The circuit has low complexity which depends from the inverse density of defined values $D(\vec{V}) = \frac{N}{m}$.

The decoding circuit includes RAM and random logic. It performs sequential decoding. The total RAM size is proportional to the

$$\log \left(D(\vec{V}) \right) ,$$

the number of random logic cells is proportional to

$$\log \log \left(D(\vec{V}) \right) * \left(\log \log \log \left(D(\vec{V}) \right) \right)^2 .$$

So the decoding circuit will be small enough even for the very low density sequences. The decoder complexity doesn't depend of the sequence length at all.

1 Introduction

The problem consideration follows to the well known technical problem – compression of test vectors for large digital designs. The undefined points of sequence are called "don't care" values in this terminology.

The random test vectors cover usually the main portion of the circuit failures. The not big random vectors amount usually cover more then 95% of failures. We need usually much more additional vectors to cover the remaining several percents of the failures, which not covered by random vectors.

The new random vectors don't increase coverage after some point (usually 95-97%). So we have to create individual vectors for individual failures or for small groups of failures from the remaining.

These vectors save their test properties when we are changing some of the vector positions. Such positions are called "don't care". We can put any value (0 or 1) instead of "don't care" on each of those potions.

There are many papers which deal with the subject of coding and decoding test vectors with "don't care" values. The results observation can be found for example in [21].

There are at least several groups of methods:

- Periodically-loaded schemes with a fixed number of free variables per test cube: [1] , [2], [3].
- Periodically-loaded schemes with a variable number of free-variables per test cube: [4] ,[5], [6] , [7] , [8].
- Continuous-flow schemes with a fixed number of free variables per test cube: [18],[9], [11], [12], [13], [14], [15], [16].
- Continuous-flow schemes with a variable number of free variables per test cube: [20], [17] , [18] , [19]

All of them some how use LFSR seeding for encoding and decoding. The second common characteristic: it is unknown how close those methods to the optimal solution.

It is easy that code length of general sequence with m "care" positions have to be at least m , because the number of such sequences more then 2^m . So this is natural lower bound for the code length.

We introduce new coding and decoding architecture for sequences with don't care values. The code length in our case is close to m , i.e. we produce almost optimal codes for those sequences.

More then 15 years ago the development of small mathematical theory has been completed: the asymptotical complexity of partial boolean functions – [22], [23], [24], [25], [26], [27], [28] . The author of this paper has proved the last result in this theory.

The partial boolean function is function of boolean arguments with boolean values defined on some subset of boolean cube. Any sequence with don't care values can be represented as partial boolean function. This is the function defined on binary representations of indexes correspondent to "care" values. The function value on some index is the correspondent sequence member.

The circuit compute partial boolean function if produces the same values on the function definition area. It can produce any value on the rest of boolean cube.

The main result of mentioned theory: the natural code length of the circuit for computing partial boolean function defined on m points is close to m . Our idea roughly is to use circuit code of correspondent partial function for encoding sequences with "don't care" values.

2 Formalization and main result

Let \vec{V} is the vector from $\{0, 1, *\}^N$, where S^n is the set of all n -dimensional vectors with the components from the set S . Let $\mathcal{M}(\vec{V})$ be the set of all indexes correspond to the "care" values and $\mu(\vec{V})$ is the number of such values. We consider indexes as vectors from $\{0, 1\}^n$, where $n = \lceil \log(N) \rceil$, the $\log(x)$ is $\log_2(x)$ by default.

Let $F_{\vec{V}}(x_1, x_2, \dots, x_n)$ be the partial boolean function computes the value $\vec{V}[i]$ from vector representation of index i . It has "don't care" value if $i \geq N$. By $\mu(F)$ we denote the number of values from $\{0, 1\}$, i.e. $\mu(\vec{V}) = \mu(F_{\vec{V}})$.

The code length for vector \vec{V} is tightly connected with the complexity of the function $F_{\vec{V}}$. The standard code of the computing $F_{\vec{V}}$ minimal circuit can be used as compression of the vector \vec{V} . This is the main idea of this paper. The only reasons we don't do exactly this is the complexity of finding minimal circuit and complexity of decoding device.

Our approach is to create code which close to the minimal circuit code in average, but has simple algorithms for encoding and decoding.

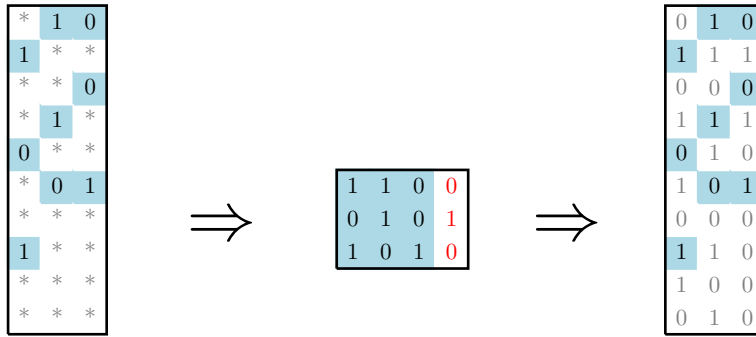


Figure 1: Coding and Decoding

The general coding and decoding diagram is captured on the Figure 1. The coding (compression) algorithm is working in software realization. Its result is the tester program. The run time of this software isn't critical because the compression have to be done just once. By the way our compression algorithm is simple so this run time is small versus the test vectors creation run time.

The code (tester program) is the binary vector \vec{C} . It sectioned on words of $(k+2)$ width for some positive integer k . The word length is represented at the program start by the code

$$\underbrace{00 \dots 0}_k 11 .$$

The each word $\mathbf{w}[i]$ of the program rest sectioned on the 2-bit command $\mathbf{com}[i]$ and k -bit data $\mathbf{data}[i]$. The 3 types of command are defined:

- 00 – pause, the data is the transmission pause duration in this case
- 01 – transmission, the data have to be sequentially transmitted in this case
- 11 – stop, the tester have stop program execution at that point

We suppose that interface between tester and decoder includes 2 signals

- **code** – sequential vector code
- **cen** – (code enable) the decoder accept the **code** only if **cen** == 1

The tester sends **code** to the decoder in mentioned above bursts with pauses, because the output vector of decoder has more bits versus the compressed code. We don't consider here the standard **clock** and **reset** signals and leave the reset sequence outside of our scope.

Let the informative part of tester program is the sequence of $(k+2)$ -length words

$$(\mathbf{com}[i], \mathbf{data}[i]), \quad i = 1, 2, \dots, s,$$

then the transmission sequence $\mathbf{code}[j]$, $\mathbf{cen}[j]$ and transmission duration t can be computed by the simple program on Figure 2

The Figure 3 captures the global decoder architecture and interface between tester and decoder.

The decoder output includes 2 signal

- **vect** – sequential vector
- **ven** – (vector enable) the decoder output is void if $VEN == 0$.

The decoder output is almost continuous. I.e. there is some initialization period when **ven** == 0 but after some point **ven** became 1 and decoder continuously transmits test vector to the design. The **ven** signal goes low when transmission has finished.

Let \vec{C} is the tester program. Let ts and te are the first and last cycles t respectively when $\mathbf{ven}[t] = 1$. We denote by $\mathbf{D}(\vec{C})$ the decoder **D** output vector, i.e.

$$\mathbf{D}(\vec{C}) = (\mathbf{vect}[ts], \mathbf{vect}[ts+1], \dots, \mathbf{vect}[te]) .$$

Let

$$X^* = \bigcup_{N=1}^{\infty} X^N .$$

```

void tester(int k, int* com, int* data, int s,
           int* code, int* cen, int& t)
{
    int i, j;    t = 0;
    for(i = 0; i < s; i++){
        switch(com[i]){
            case 0: for(j=0; j<data[i]; j++){
                    code[t]=0; cen[t]=0; t++;
                } break;
            case 1: for(j=0; j<k; j++){
                    code[t]=(data[i]>>j)%2; cen[t]=1; t++;
                } break;
            case 3: return; break;
        }
    }
}

```

Figure 2: The tester program

The compression system \mathbf{S} is the pair $\mathbf{S} = (\mathbf{F}^{\mathbf{S}}, \mathbf{D}^{\mathbf{S}})$ where

- $\mathbf{F}^{\mathbf{S}}$ is the compression function $\mathbf{F}^{\mathbf{S}} : \{0, 1, *\}^* \longrightarrow \{0, 1\}^*$
- $\mathbf{D}^{\mathbf{S}}$ is decoding circuit

such that for any positive integer N and for any $\vec{V} \in \{0, 1, *\}^N$ the vector $\vec{W} = \mathbf{D}^{\mathbf{S}}(\mathbf{F}^{\mathbf{S}}(\vec{V}))$ is belong to the set $\{0, 1\}^N$ and

$$\vec{V}[i] \in \{0, 1\} \implies \vec{W}[i] = \vec{V}[i], \quad i = 0, 1, \dots, N-1.$$

The $\mathbf{F}^{\mathbf{S}}(\vec{V})$ is the code of vector \vec{V} .

Let $\mathbf{L}_{code}(\mathbf{S}, N, m)$ is the maximal code length for vectors from $\{0, 1, *\}^N$ with m components from $\{0, 1\}$ and $(N - m)$ components equal to $*$ ("don't care").

We define for the circuit \mathbf{D} the following measures

- $\mathbf{L}_m(\mathbf{D})$ is the total number of memory bits
- $\mathbf{l}_m(\mathbf{D})$ is the number of memories
- $\mathbf{L}_c(\mathbf{D})$ is the number of cells

This paper introduces the parametrical family of decoders and correspondent compression algorithms.

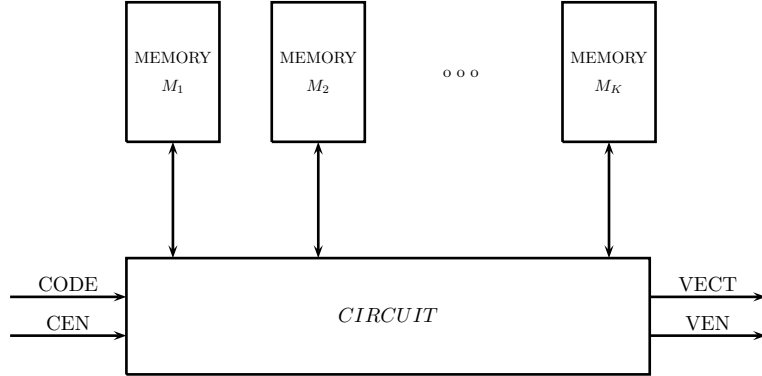


Figure 3: The decoder architecture

Theorem 2.1. *There exists the sequence of compression systems $\mathbf{S}_\alpha, \alpha \geq \alpha_0$, such that*

$$\begin{aligned}
 \mathbf{L}_{code}(\mathbf{S}_\alpha, N, m) &\leq m + O(2^{-\alpha} * N) + O(\alpha) \\
 \mathbf{L}_c(\mathbf{D}^{\mathbf{S}_\alpha}) &\leq O((\log \alpha)(\log \log \alpha)^2) \\
 \mathbf{l}_m(\mathbf{D}^{\mathbf{S}_\alpha}) &\leq O(\log \log \alpha) \\
 \mathbf{L}_m(\mathbf{D}^{\mathbf{S}_\alpha}) &\leq O(\alpha)
 \end{aligned}$$

3 Partial Boolean Functions Decomposition

3.1 History

Let $\mathbf{L}(\vec{V})$ is the minimal possible complexity of computing $F_{\vec{V}}$ circuit. We consider circuits from all gates with two or one input. We suppose that complexity of circuit is circuit size. Let introduce the Shannon function

$$\mathbf{L}(N, m) = \max_{\vec{V} \in \{0,1,*\}^N : \mu(\vec{V}) \leq m} L(\vec{V}) .$$

C.Shannon (1949) [22] has put the complexity problem and has obtained the first results in this direction. He has proved:

$$\mathbf{L}(N, N) = \Theta\left(\frac{N}{\log N}\right) .$$

O.Lupanov (1956) [23] has obtained the asymptotic behavior of Shannon function:

$$\mathbf{L}(N, N) \sim \frac{N}{\log N} .$$

E.Nechiporuk (1965) [24] has obtained first results about worst case complexity for partial boolean function:

$$\mathbf{L}(N, m) \sim \frac{m}{\log m} .$$

He has prove this asymptotic in the case where $m = N^{1-o(1)}$. L.Sholomov (1969) [25] has proved this asymptotic behavior for more general case.

N.Pipenger (1977) [26] has considered the classes of partial functions with fixed part of units. He has put Shannon function $\mathbf{L}(N, m, \alpha)$ – worst case complexity of partial functions with size of domain m and with α -part of units.

He has proved asymptotic behavior of this function

$$\mathbf{L}(N, m, \alpha) \sim (-\alpha \log \alpha - (1 - \alpha) \log(1 - \alpha)) \frac{m}{\log m}$$

in the case where $m = \Omega(N)$, $\min(\alpha, 1 - \alpha) = \Omega(1)$.

A.Andreev (1989) [28] has proved the best result for function $\mathbf{L}(N, m)$. He has proved

$$\mathbf{L}(N, m) \sim \frac{m}{\log m} + O(\log N)$$

(no restrictions for domain size). We will use A.Andreev basic technic from this paper and [27] too.

3.2 Decomposition Architecture

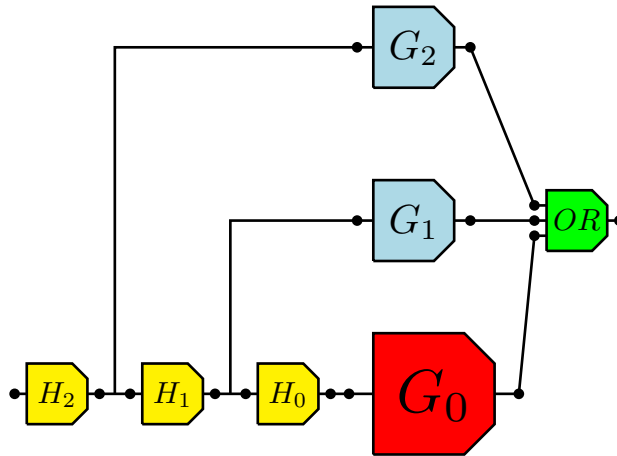


Figure 4: The partial function canonical representation.

Let $F(\vec{\mathbf{x}})$, $\vec{\mathbf{x}} = (x^1, x^2, \dots, x^n)$, is partial boolean function. Let $n \geq n_2 \geq n_1 \geq n_0$ and

$$\begin{aligned}\vec{\mathbf{x}}_2 &= (x_2^1, x_2^2, \dots, x_2^{n_2}) , \\ \vec{\mathbf{x}}_1 &= (x_1^1, x_1^2, \dots, x_1^{n_1}) , \\ \vec{\mathbf{x}}_0 &= (x_0^1, x_0^2, \dots, x_0^{n_0}) .\end{aligned}$$

Let H_2 , H_1 and H_0 are boolean operators

$$\begin{aligned}H_2 &: \{0, 1\}^n \longrightarrow \{0, 1\}^{n_2} , \\ H_1 &: \{0, 1\}^{n_2} \longrightarrow \{0, 1\}^{n_1} , \\ H_0 &: \{0, 1\}^{n_1} \longrightarrow \{0, 1\}^{n_0} .\end{aligned}$$

The function $F(\vec{\mathbf{x}})$ can be represented as

$$\begin{aligned}\vec{\mathbf{x}}_2 &= H_2(\vec{\mathbf{x}}) ; \quad \vec{\mathbf{x}}_1 = H_1(\vec{\mathbf{x}}_2) ; \quad \vec{\mathbf{x}}_0 = H_0(\vec{\mathbf{x}}_1) ; \\ F(\vec{\mathbf{x}}) &= G_0(\vec{\mathbf{x}}_0) \vee G_1(\vec{\mathbf{x}}_1) \vee G_2(\vec{\mathbf{x}}_2)\end{aligned}$$

To be able define G_0 , G_1 and G_2 we have to introduce intermediate functions

$$\begin{aligned}F_2 &: \{0, 1\}^{n_2} \longrightarrow \{0, 1, *\} , \\ F_1 &: \{0, 1\}^{n_1} \longrightarrow \{0, 1, *\} .\end{aligned}$$

Let $U^{-1}(\mathbf{b})$ is the set of all \mathbf{a} such that $U(\mathbf{a}) = \mathbf{b}$.

The function F determines function F_2 based on operator H_2 by

$$F_2(\mathbf{b}) = \begin{cases} 0 & \text{if } H_2^{-1}(\mathbf{b}) \cap F^{-1}(0) \neq \emptyset \\ 1 & \text{if } H_2^{-1}(\mathbf{b}) \cap F^{-1}(1) \neq \emptyset \\ * & \text{otherwise} \end{cases}$$

We assume that

$$F(\mathbf{a}_1) = 0, F(\mathbf{a}_2) = 0 \implies H_2(\mathbf{a}_1) \neq H_2(\mathbf{a}_2).$$

The function F_2 determines functions F_1 and G_2 based on operator H_1 by following way.

$$F_1(\mathbf{b}) = \begin{cases} 0 & \text{if } H_1^{-1}(\mathbf{b}) \cap F_2^{-1}(0) \neq \emptyset \\ 1 & \text{if } H_1^{-1}(\mathbf{b}) \cap F_2^{-1}(0) = \emptyset; \quad H_1^{-1}(\mathbf{b}) \cap F_2^{-1}(1) \neq \emptyset \\ * & \text{otherwise} \end{cases}$$

$$G_2(\mathbf{a}) = \begin{cases} 1 & \text{if } F_2(\mathbf{a}) = 1; \quad H_2^{-1}(H_1(\mathbf{a})) \cap F_2^{-1}(0) \neq \emptyset \\ 0 & \text{otherwise} \end{cases}$$

The function F_1 determines functions F_0 and G_1 based on operator H_0 by similar way. We let $G_0 = F_0$.

References

- [1] Knemann, B., LFSR-Coded Test Patterns for Scan Designs, Proc. of European Test Conf., pp. 237-242, 1991.
- [2] Hellebrand, S., J. Rajski, S. Tarnick, S. Venkataraman and B. Courtois, Built-In Test for Circuits with Scan Based on Reseeding of Multiple-Polynomial Linear Feedback Shift Registers, IEEE Trans. on Computers, Vol. 44, No. 2, pp. 223-233, Feb. 1995.
- [3] Krishna, C.V., A. Jas, and N.A. Touba, "Test Vector Encoding Using Partial LFSR Reseeding", Proc. of IEEE International Test Conference, pp. 885-893, 2001.
- [4] Zacharia, N., J. Rajski, and J. Tyszer, Decompression of Test Data Using Variable-Length Seed LFSRs, Proc. of VLSI Test Symposium, pp. 426-433, 1995.
- [5] Zacharia, N., J. Rajski, J. Tyszer, and J. Waicukauski Two Dimensional Test Data Decompressor for Multiple Scan Designs, Proc. of International Test Conf., pp. 186-194, 1996.
- [6] Rajski, J., J. Tyszer, and N. Zacharia, Test Data Decompression for Multiple Scan Designs with Boundary Scan, IEEE Trans. on Comp., Vol. 47, No. 11, pp. 1188-1200, Nov. 1998.
- [7] Hellebrand, S., B. Reeb, S. Tarnick, and H.-J. Wunderlich, Pattern Generation for a Deterministic BIST Scheme, Proc. of International Conference on Computer-Aided Design (ICCAD), pp. 88-94, 1995.
- [8] Krishna, C.V., and N.A. Touba, "Reducing Test Data Volume Using LFSR Reseeding with Seed Compression", Proc. of IEEE International Test Conference, pp. 321-330, 2001.
- [9] Jas, A., B. Pouya, and N.A. Touba, Virtual Scan Chains: A Means for Reducing Scan Length in Cores, Proc. of VLSI Test Symposium, pp. 73-78, 2000.
- [10] Bayraktaroglu, I., and A. Ogailoglu, Test Volume and Application Time Reduction Through Scan Chain Concealment, Proc. of Design Automation Conference, pp. 151-155, 2001.
- [11] Bayraktaroglu, I., and A. Ogailoglu, Decompression Hardware Determination for Test Volume and Time Reduction through Unified Test Pattern Compaction and Compression, Proc. of VLSI Test Symposium, pp. 113-118, 2003.
- [12] Mitra, S., and K.S. Kim, XMAX: X-tolerant architectures for Maximal Test Compression, Proc. of International Conference on Computer Design, pp. 326-330, 2003.

- [13] Rajski, J., et al., Embedded Deterministic Test for Low Cost Manufacturing Test, Proc. of Int. Test Conf., pp. 301-310, 2002.
- [14] Mrugalski, G., J. Rajski, and J. Tyszer, High Speed Ring Generators and Compactors of Test Data, Proc. of VLSI Test Symposium, pp. 57-62, 2003.
- [15] Hamzaoglu, I., and J.H. Patel, Reducing Test Application Time for Full Scan Embedded Cores, Proc. of Int. Symposium on Fault Tolerant Computing, pp. 260-267, 1999.
- [16] Hsu, F.F., K. M. Butler, J. H. Patel, "A Case Study on the Implementation of the Illinois Scan Architecture," Proc. of International Test Conference, pp. 538-547, 2001.
- [17] Volkerink, E.H., and S. Mitra, Efficient Seed Utilization for Reseeding Based Compression, Proc. of VLSI Test Symposium, pp. 232-237, 2003.
- [18] Bayraktaroglu, I., and A. Ogailoglu, Test Volume and Application Time Reduction Through Scan Chain Concealment, Proc. of Design Automation Conference, pp. 151-155, 2001.
- [19] Rao, W., I. Bayraktaroglu, and A. Orailoglu, Test Application Time and Volume Compression through Seed Overlapping, Proc. of Design Automation Conference, pp. 732-737, 2003.
- [20] Knemann, B., A SmartBIST Variant with Guaranteed Encoding Proc. of Asian Test Symposium, pp. 325-330, 2001.
- [21] C.V. Krishna and Nur A. Touba 3-Stage Variable Length Continuous-Flow Scan Vector Decompression Scheme. Proceedings of the 22nd IEEE VLSI Test Symposium (VTS 2004)
- [22] Shannon, C.E. (1949), The synthesis of two-terminal switching circuits, Bell. Syst. Tech. J. 28, pp.59-98.
- [23] Lupanov, O.B. (1956) About gating and contact-gating circuits, Dokl. Akad. Nauk SSSR 111, pp.1171-11744. (In Russian), English translation in Soviet Math. Docl.
- [24] Nechiporuk, E.I. (1965), About the complexity of gating circuits for the partial boolean matrix, Dokl. Akad. Nauk SSSR 163, pp.40-42. (In Russian). English translation in Soviet Math. Docl.
- [25] Sholomov, L.A. (1969) About realization of partial boolean functions by circuits from functional elements, Problemy Kibernet. 21, pp.215-226. (in Russian). English Translation in Systems Theory Res. v.21, 1971.
- [26] Pippenger, N. (1977) Information theory and the complexity of Boolean functions, Math. Systems Theory 10, pp.129-167.

- [27] Andreev A.E. The universal principle of self-correction. *Mat. Sbornik* 127(169), N 6, p.147-172, 1985.(In Russian) English transl. in *Math. USSR Sbornik*, vol.55, 1986, N 1, pp.145-169.
- [28] Andreev, A.E. (1989), On the complexity of the realization of partial Boolean functions by circuits of functional elements, *Diskret. mat.* 1, pp.36-45. (In Russian). English translation in *Discrete Mathematics and Applications* 1, pp.251-262.