ICS 31.200
CCS L71

团 体 标 准

T/CESA 1267.1—2023

电子设计自动化 电路仿真
第 1 部分：波形数据编码规则

Electronic design automation— Circuit simulation—

Part 1 : Coding rules of waveform data

2023-07- 27 发布　　2023-08- 01 实施

中国电子工业标准化技术协会　发布

目　　次

前　言

本文件按照GB/T 1.1-2020《标准化工作导则 第1部分：标准化文件的结构和起草规则》的规定起草。

请注意本文件的某些内容可能涉及专利。本文件的发布机构不承担识别专利的责任。

本文件是T/CESA 1267《电子设计自动化电路仿真》的第1部分。

T/CESA 1267 已经发布了以下部分：

——第1部分：波形数据编码规则。

本文件由中国电子技术标准化研究院提出。

本文件由中国电子技术标准化研究院、中国电子工业标准化技术协会归口。

本文件起草单位：中国电子技术标准化研究院、清华大学、南京集成电路设计服务产业创新中心有限公司、北京华大九天软件有限公司、上海国微思尔芯技术股份有限公司、国微集团（深圳）有限公司、深圳市海思半导体有限公司、北京超逸达科技有限公司、芯华章科技股份有限公司、无锡亚科鸿禹电子有限公司。

本文件主要起草人：刘舒宁、菅端端、喻文健、陈刚、郭继旺、刘晓明、林铠鹏、张岩、肖靖帆、胡超、傅强、闫宇暾。

引　言

仿真波形数据的编码方法，旨在设计并实现一种高效的波形压缩算法，并由此确立国内的波形数据存储格式标准。主要内容为对于电路仿真输出的各种波形数据，进行压缩后，存入文件中，形成一种特定的波形格式，对这种格式的文件进行解压缩，读取出指定的信号波形数据。

本文件是指导电子设计自动化中的电路仿真工具波形数据的编码与解码的基础标准，旨在确保数据编码与解码符合电路仿真工具应用需求，拟由2个部分组成。

——第1部分：波形数据编码规则。目的在于确立电子设计自动化中的电路仿真工具波形数据的编码与解码的技术要求。

——第2部分：波形文件接口要求。目的在于规定数字和模拟波形的读写的接口要求，使得兼容波形访问的接口便于业界的波形交换。

电子设计自动化 电路仿真 第1部分：波形数据编码规则

1 范围

本文件规定了集成电路仿真波形数据的技术要求，包括信号类型、数据类型、压缩率要求、精度要求和模块设计要求，描述模拟和数字波形的压缩和解压算法、预测算法和存储格式等技术要求。

本文件适用于电子设计自动化中的电路仿真工具波形数据的编码与解码。

2 规范性引用文件

本文件没有规范性引用文件。

3 术语和定义

下列术语和定义适用于本文件。

3.1

压缩模块 compression module

将仿真产生的波形数据进行编码和压缩，并存储为特定格式的波形数据的模块。

3.2

解压模块 decompression module

将压缩模块产生的特定格式波形数据进行解码和解压缩，并恢复为通用格式的波形数据的模块。

3.3

事件数据类型 event data type

用于对测试电路里的信号进行监测的变量的数据类型。

注：满足触发条件就触发事件并引发相应的处理。

3.4

总线数据类型 bus data type

描述一组信号组成的接口界面的数据类型。

3.5

VCD 格式 VCD format

记录信号变化的格式文件，是通用的文件格式。

4 缩略语

下列缩略语适用于本文件。

EDA 电子设计自动化（Electronic design automation）

LZSS 詹姆斯·斯托尔与托马斯·西曼斯基提出的压缩算法（Lempel-Ziv-Storer-Szymanski）

5 技术要求

5.1 信号类型

支持的信号类型应包括模拟信号和数字信号。

信号显示应包含横轴和纵轴。

横轴的类型应为时间、频率或任意的扫描变量，如温度等。

纵轴的类型应为电学物理类型或数学统计类型，如电压、电流、电容、功率等。

5.2 数据类型

模拟信号和数字信号的横轴均应支持单精度或双精度浮点型数据，默认为双精度，且当横轴类型为时间时，还应支持unsigned long long整型数据。

模拟信号的纵轴数据类型应为浮点型，支持单精度或双精度，默认为双精度。当横轴类型为频率时，模拟信号的纵轴数据应为复数，其实部和虚部均为浮点型，即支持单精度或双精度，默认为双精度。

数字信号的纵轴数据类型为整数，应支持0，1，x，z这四种基础类型，宜支持事件（event）数据类型，总线（bus）数据类型 和SystemVerilog接口(Interface) 数据类型。

5.3 压缩率

压缩率应符合下列要求：

a） 除特殊要求外，压缩率应在 5 倍以上；

b） 模拟信号的压缩率宜不低于普通无压缩二进制格式的压缩率；

c） 数字信号的压缩率宜不低于 VCD 格式的压缩率。

5.4 精度误差

精度误差符合下列要求：

a） 模拟信号和数字信号的横轴均不应产生精度误差；

b） 纵轴精度误差：

1） 模拟信号可精度误差：

- 相对误差应小于千分之一；
- 电压信号的最大绝对误差宜不大于 1μV；
- 电流信号的最大绝对误差宜不大于 1nA；
- 其它属性应按具体电路设计要求进行自定义设置。

2） 数字信号不应有精度误差。

5.5 模块设计

5.5.1 压缩模块

压缩模块应符合下列要求：

a） 压缩模块的功能应将仿真产生的波形数据进行编码和压缩，并存储为特定格式的波形文件；
b） 压缩模块应支持根据信号个数自适应改变默认的缓存(buffer)大小，用户可自定义设置 buffer 大小；
c） 应支持在仿真期间进行 flush 操作，将 buffer 中的数据立刻写入到磁盘中；
d） 应支持基于时间点(time)的方式或基于信号（signal）的方式进行写入操作：，
1） 基于 time 的方式，每次将一个时间点的多个信号的数据同时写入；
2） 基于 signal 的方式，每次将一个信号的所有时间点的数据同时写入。
e） 应支持压缩文件分割操作，可设置文件大小的限制值，当超过这个值时，后面的数据按顺序写入到下一个文件中；
f） 分割后的多个文件，名称应有相同的前缀，并以数字的递增表示数据的顺序；
g） 应支持信号的物理属性名称的缩写或简称、单位、绝对误差等信息的自定义设置，没有指定时采用默认值；
h） 应支持多线程，仿真计算、数据压缩、输出文件三个过程能并行处理。

注：“仿真计算过程”指仿真过程中对电路方程的建立和求解过程。

5.5.2 解压模块

解压模块应符合下列要求：
a） 应读取压缩后的波形文件，对文件中的部分或全部信号的波形数据进行解压缩；
b） 解压缩后的波形数据与压缩前的波形数据相比应符合设定的精度误差要求；
c） 应支持增量式读取数据，即从任意时刻开始读取数据，可读取指定仿真开始和结束时间内的数据，应在指定的结束时刻停止读取。
d） 信号在开始时刻或结束时刻没有仿真的时间点，应进行插值并返回对应的值。
e） 应支持多线程，读取文件、数据解压缩、数据处理三个过程并行进行。

注：“数据处理过程”包括用图形界面工具（如波形显示器）将解压缩后的波形进行绘制渲染，进行数学计算和测量等操作。

6 信号压缩算法

6.1 模拟波形的压缩算法

6.1.1 模拟波形的压缩方法

瞬态电路仿真中的波形可用向量值函数表示，见公式（1）：

$$v(t)=\left[v_1(t)\quad v_2(t)\quad \cdots\quad v_{N_s}(t)\right]^T \qquad (1)$$

式中：
t ——仿真时间；
N_s ——仿真信号的数量；
$v_i(t)$ ——第 i 个信号在时间 t 的值。

波形不能被连续存储，仅能存储N_t个离散时间点$t=[t_1,t_2,...,t_{Nt}]$，以及它们对应的信号值$V=[v(t_1),v(t_1),...,v(t_{Nt})]$，波形可用矩阵形式表示，见公式（2）：

$$W=\left[\frac{t}{V}\right]=\begin{bmatrix} t_1 & t_2 & \cdots & t_{N_t} \\ v_{11} & v_{12} & \cdots & v_{1N_t} \\ v_{21} & v_{22} & \cdots & v_{2N_t} \\ \vdots & \vdots & \ddots & \vdots \\ v_{N_s1} & v_{N_s2} & \cdots & v_{N_sN_t} \end{bmatrix} \quad \text{（2）}$$

式中：

W ——波形矩阵；

t ——仿真时间；

V ——离散时间点对应的信号值；

N_s ——存储的仿真信号数量；

N_t ——存储的离散时间点数量；

采用默认的双精度格式存储矩阵 W 的所有元素时，原始波形需要占用 $64\times(N_s+1)\times N_t$ 比特的空间。

波形压缩可与仿真同时处理，压缩算法应快速处理从仿真器中输出的流数据时，以免延缓仿真算法的运行。

压缩波形可被存储为硬盘上的一个文件，压缩率可按公式（3）计算：

$$R_{comp}=\frac{S_{raw}}{S_{file}} \quad \text{（3）}$$

式中：

R_{comp} ——压缩率；

S_{raw} ——原始波形流数据大小；

S_{file} ——压缩文件大小。

解压算法能从压缩文件中恢复波形信号。仅需部分信号时，解压算法能恢复W中特定的行，解压算法运行时间应明显短于直接从原始数据中读取波形信号的时间。

6.1.2 模拟波形信号的误差和类型

波形的压缩和解压缩可能引入误差，任何的信号值和其对应的恢复信号值，均需考虑绝对误差和相对误差，见公式（4）和公式（5）：

$$\varepsilon=|v-\hat{v}| \quad \text{（4）}$$

$$\eta=\frac{\varepsilon}{|v|}=\left|\frac{v-\hat{v}}{v}\right| \quad \text{（5）}$$

式中：

ε ——绝对误差；

η ——相对误差；

v ——任意信号值；

$\hat{v}$ ——对应的恢复信号值。

不允许时间误差，允许一定范围内的信号值误差。在压缩之前可分别设置每个信号的绝对误差限ε_{abs}以及相对误差限ε_{rel}。

对每一个恢复的信号值，压缩和解压缩算法应确保$\varepsilon\leq\max(|v|,|\hat{v}|)\cdot\varepsilon_{rel}+\varepsilon_{abs}$。在瞬态电路仿真电压信号时，一种通用的误差限为$\varepsilon_{abs}=10^{-6}$以及$\varepsilon_{rel}=10^{-4}$。

实际压缩过程中，允许规定范围内误差，即$\varepsilon\leq\varepsilon_{abs}$或$\eta\leq\varepsilon_{rel}$。按照此要求，对于较大的信号值$v$，相对误差更符合要求，而对于较小的$v$绝对误差更符合要求，见公式（6）：

$$\begin{cases} \varepsilon \le \varepsilon_{abs}, 当|v| \le \tau 且 \eta \le \varepsilon_{rel} \\ \eta \le \varepsilon_{rel}, 当|v| > \tau 且 \varepsilon \le \varepsilon_{abs} \\ \tau = \dfrac{\varepsilon_{abs}}{\varepsilon_{rel}} \end{cases} \quad \cdots\cdots (6)$$

式中：

ε_{abs} ——绝对误差限；

ε_{rel} ——相对误差限；

τ ——阈值。

对于满足$|v| \le \tau$的信号值（称为小信号值），压缩格式应优先考虑满足$\varepsilon \le \varepsilon_{abs}$。对于满足$|v| \ge \tau$的信号值（称为大信号值），压缩格式应优先考虑满足$\eta \le \varepsilon_{rel}$。

针对小信号值和大信号值提供不同的压缩格式，满足相对误差或绝对误差的要求同时减少存储空间。

6.1.3 模拟波形信号压缩格式

模拟波形信号的大信号值压缩格式见图1：

1 bit	1 bit	2 bits	e bits	m bits
0	S	Δ_E	E	M

说明：

1——e 和m 分别表示存储值E 和M 的位数；

2——Δ_E为当前时间点E 值与上一时间点E'值的差，Δ_E的取值如下式（7）：

$$\Delta_E = \begin{cases} 0, 当 E = E' - 1 \\ 1, 当 E = E' \\ 2, 当 E = E' + 1 \\ 3, 其他 \end{cases} \quad \cdots\cdots (7)$$

3——当且仅当Δ_E=3时，整数E 才被显式储存。

图 1 模拟大信号值压缩格式

基于模拟大信号值的压缩格式，恢复后的模拟波形信号值应按照式（8）计算得出：

$$\hat{v} = (-1)^S \cdot \left(1 + M \cdot 2^{-m}\right) \cdot 2^E \cdot \tau \quad \cdots\cdots (8)$$

式中：

$\hat{v}$ ——恢复后的信号值。

模拟大信号情况下，可储存模拟波形信号绝对值的最大值计算值按式（9）：

$$|\hat{v}|_{\max} = \left(2 - 2^{-m}\right) \cdot 2^E \cdot \tau \quad \cdots\cdots (9)$$

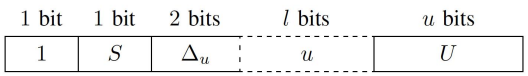
式中：

$|\hat{v}|_{\max}$ ——可储存模拟波形信号的最大值。

为了满足误差要求，参数的设置应符合式（10）：

$$\begin{cases} m \ge -\log_2 \varepsilon_{rel} - 1 \\ 2^e - b > \log_2 \left(\varepsilon_{abs} / \varepsilon_{rel}\right) \end{cases} \quad \cdots\cdots (10)$$

模拟小信号值储存相邻两个时间点信号值的差，压缩格式应符合图2显示的压缩格式。



说明：

1——l 和 u 分别表示存储值 u 和 U 的位数；

2——Δ_u 为当前时间点 u 值与上一时间点 u' 值的差，Δ_u 的取值如下式（11）：

$$\begin{cases} \Delta_u = 0,\ u = u' - 1,\ 当 \left|\log_2 U\right| < u' \\ \Delta_u = 1,\ u = u',\ 当 \left|\log_2 U\right| = u' \\ \Delta_u = 2,\ u = u' + 3,\ 当 u' < \left|\log_2 U\right| \le u' + 3 \\ \Delta_u = 3,\ u = \left|\log_2 U\right|,\ 当 \left|\log_2 U\right| > u' + 3 \end{cases} \quad \text{（11）}$$

3——当且仅当 $\Delta_u = 3$ 时，整数 u 才被显式储存。

图 2 模拟小信号值压缩格式

基于模拟小信号值的压缩格式，恢复后的模拟波形信号值应满足式（12）计算值进行恢复：

$$\hat{v} = (-1)^S \cdot U \cdot c \quad \text{（12）}$$

式中：

c ——压缩参数，且为满足误差要求，应满足 $c < 2 \cdot \varepsilon_{abs}$。

模拟小信号情况下，可储存模拟波形信号绝对值的最大值按式（13）：

$$|\hat{v}|_{\max} = \left(2^{2^l} - 1\right) \cdot c \quad \text{（13）}$$

式中：

$|\hat{v}|_{\max}$ ——可储存模拟波形信号的最大值，且应大于等于τ。

6.1.4 模拟波形的完整存储数据格式

模拟波形的完整存储格式应包含如下信息：

a) 波形信息，由以下部分组成：

1) N_t：64 位无符号整数，表示时间点数目；

2) N_{type}：32 位无符号整数，表示信号类型数目；

3) N_s：32 位无符号整数，表示信号数目；

4) T_0：双精度浮点数，表示起始时间；

5) T_N：双精度浮点数，表示结束时间；

6) N_b：32 位无符号整数，表示每个数据块最多可能包含的时间点数。

b) 字符串类型的数据应按如下格式储存：

1) L：32 位无符号整数，表示字符串字节长度；

2) L 个字节：表示字符串内容。

c) 类型信息包含了信号纵轴类型的信息，应由 N_{type} 个类型子块组成，每个类型子块应依次由以下部分组成：

1) 类型描述：字符串；

2) 单位：字符串；

3) ε_{abs}：双精度浮点数，表示最大容许绝对误差；

4) ε_{rel}：双精度浮点数，表示最大容许相对误差；

5) $|v|_{\max}$：双精度浮点数，表示最大信号绝对值；

6) τ,e,m,c,l：均为双精度浮点数，储存压缩参数。

d) 信号信息应由 N_s 个信号子块组成，每个信号子块应依次由以下部分组成：

1) 信号名：字符串；

2) 类型：32 位无符号整数，表示信号类型的序号。

波形压缩应在瞬态仿真过程中执行，所产生的信号值应为数据流，波形矩阵W从左往右动态增加。应将压缩后的信号值按块存储，控制压缩的峰值内存。每个块对应一个时间段，n个相邻的时间点构成波形矩阵W的n个相邻列。模拟波形数据块的完整存储格式见图3，它由一个块头（n、$\hat{t}$和p）和N_s个压缩信号值的子块组成。块中不同信号宜相互独立，方便于解压部分信号。

块头			压缩的信号值		
n	$\hat{t}$	p	信号1	…	信号 N_s

说明：

1——n为32位无符号整数，代表相邻的时间点数目，可通过设定峰值内存确定；

2——$\hat{t}$ 为存储在双精度浮点数中的n个时间点：

3——向量p存储该块中N_s个信号子块的开始位置，p为32位无符号数。

图 3 模拟波形的完整存储格式

在波形压缩期间可使用通用的无损压缩算法（例如Deflate）对数据块执行二次压缩，压缩算法可结合LZSS压缩算法和Huffman编码，进一步减少数据占用的空间，具体方法见附录A。

6.1.5 模拟波形信号压缩的预测算法及存储格式

模拟小信号值和大信号值的可变长度压缩格式，已经一定程度利用了相邻时间点信号值之间的相关性，信号预测方法可进一步利用这种相关性。

预测算法找出大致位于一条直线上的连续信号值，见图4。若已知直线的斜率，则无需在其上存储大多数信号值，可使用直线方程式进行计算。以下提出两种组合方案，将预测与压缩格式相结合。

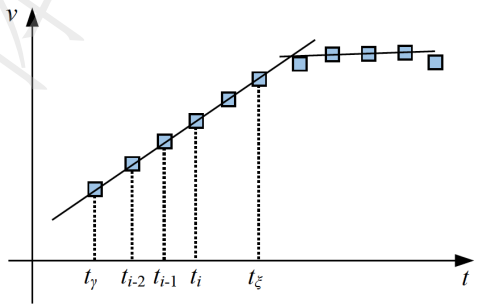


图 4 模拟波形预测算法示意图

第一种方案检测信号值构成的所有近似直线段，并使用直线段的信息表示这些信号值。每个信号的压缩值都在子块中，可在实际压缩之前分析每个信号子块以找到直线段。假设v_r（对于时间t_r）是直线段的可能起点，首先计算直线的斜率：

$$L=\frac{v_{r+1}-v_r}{t_{r+1}-t_r} \quad \text{（14）}$$

式中：

L——直线的斜率。

对于随后的时间点，可用 L 预测信号值：

$$\hat{v}_i = v_r + (t_i - t_r) \cdot L \quad \text{（15）}$$

式中：

$\hat{v}_i$——时间点 t_i 的预测波形。

若满足准确性需求，则此预测成功。假设预测在时间t_{r+2}至t_ε都成功，且在时间$t_{\varepsilon+1}$预测失败，那么这段信号值可用一个元组（r,ε,v_r,L）表示。如果此子序列的长度ε-r+1超过阈值Γ_p，我们称之为可预测的子序列（P序列），可代替原始压缩格式进行更大的压缩。特别地，当L=0时，表示P序列中的所有值几乎相同，L可在元组中省略并且子序列称为同值子序列（S序列）。

假设子块中n个时间点的信号值为：v_1,v_2,...,v_n,从r=1开始执行上述过程。在使用贪心策略快速检测值序列后，获得了一些P序列和S序列。对于不在P序列或S序列中的信号值，将以小信号值或大信号值压缩格式进行编码。

对于获得的P序列和S序列，在每个信号子块的前面添加了一个预测子块，该子块存储了表示P序列和S序列的元组。在对应信号子块中，P序列或S序列中的值被省略，图5展示了带有信号预测的压缩格式。该预测子块包括n_p个P序列和n_s个S序列，对于每个元组，r和ε被存储为16位无符号整数（假定n<65536），而 v_r 和 L 被存储为双精度浮点数。每个P序列占用160bit内存，每个S序列占用96bit。若正确选择Γ_p，则当有许多信号值构成直线段时，此方案可节省大量存储空间。

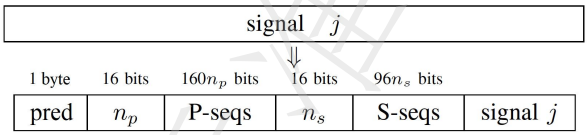


图 5 带有信号预测的压缩格式

在第二种方案中，不依赖长直线段，而基于前两个时间点的恢复值来预测当前时间点的信号值：

$$\hat{v}_i = \hat{v}_{i-1} + \frac{(t_i - t_{i-1})(\hat{v}_{i-1} - \hat{v}_{i-2})}{t_{i-1} - t_{i-2}} \quad \text{（14）}$$

若预测成功，可用预测值表示恢复值。否则，$\hat{v}_i$将以小信号值或大信号值压缩格式编码。

为将预测集成到压缩算法，添加N_s个预测子块，这些子块将预测信息存储到每个数据块中。第j个预测子块应指示对于信号j的n个值预测是否成功，仅对于成功预测的值，省略其在信号子块中的储存。如果预测成功率较低，可能导致更大的存储空间。为避免此缺点，应监控所有信号的预测成功率。若监视的成功率高于阈值，则信号进行预测。否则，信号不进行预测处理，并且省略预测子块。

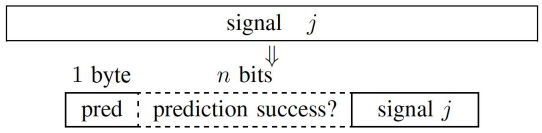


图 6 预测子块添加过程

图6展示了预测子块是如何被添加到信号子块之前的。预测子块的第一个标志字节pred指示了预测的方案：0代表不预测（即方案二低成功率），1代表方案一，2代表方案二（高成功率）。

与第一种组合方案相比，第二种方案为每个信号值计算了斜率，包含更多的计算，通过预测成功的判断，两种方案均确保每个恢复的信号值将满足预设的精度需求。且新加入的预测子块也可用无损压缩进行进一步压缩。

6.2 数字波形的压缩算法

6.2.1 数字波形文件格式

逻辑仿真的对象是由逻辑门和功能块等元件组成的逻辑电路，逻辑仿真的结果主要就是波形文件，波形文件随着仿真时钟的前进不断添加新的仿真结果，故数字波形就是以信号事件为驱动，当在某一个仿真时间节点发生了信号值变化，则对应信号的数字波形发生相应的变化。图7给出了一个与门的数字仿真电路图，输入是*x1*和*x2*信号，输出是*y*信号。仿真时间表示为*t*=[0,1,2,3,4,5]，信号*x1*的变化表示为*x1*=[1,1,1,1,0,0]，信号*x2*的变化表示为*x2*=[0,1,0,0,1,1]，*y*的变化表示为*y*=[0,1,0,0,0,0]，1表示高电平，0表示低电平。

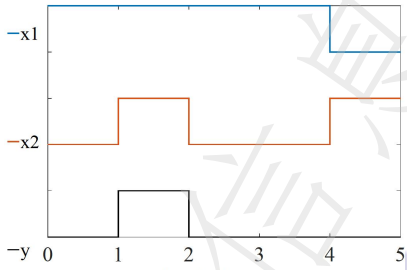


图 7 逻辑仿真结果波形图

每个信号仅在某几个仿真时间点会发生变化，故此可得到如下的信号值变化数据：

——#0，1 x1，0 x2，0 y

——#1，1 x2，1 y

——#2，0 x2，0 y

——#4，0 x1，1 x2

只需要记录有信号值发生变化的仿真时间点*t*=[0,1,2,4]以及对应发生变化后的信号值即可，这样记录信号值变化的数据比记录完整的波形数据节省空间，这也是工业中许多的逻辑仿真器采用的数字波形数据的输出方式。基于ASCII码字符的VCD文件格式是一种在工业中被广泛使用的数字波形表示格式，该数字波形表示格式也完全满足以上信号值变化的规则，可将VCD文件当作初始的波形文件。

6.2.2 数字波形类型与表示方法

定义*Transition(TR)*表示一个信号的变化情况，即*TR* ={*v*,*id* }，其中*v*表示信号值的变化，*id* 表示对应的信号名，信号名通常由几位大小为33-126的ASCII码字符组成表示。

定义*VTR*代表*Transition*的集合，则可将*VTR*表示为*VTR*=[TR_0,TR_1,…]，*Transition Block(TRB)*为一组信号变化情况，可表示为*TRB*=[*t*,*VTR*]，其中*t*表示该组信号发生变化的仿真时间点。*VTRB*表示完整的信号值变化部分，即*VTRB*=[TRB_0,TRB_1,…]。

定义*Transiton Signal Block(TRSB)*表示*Transition Block*里信号发生变化的信号名的集合，可表示为*TRSB*=[id_0,id_1,…]。

在数字信号值里，通常可划分信号类型为SCALAR类型、VECTOR类型，信号值由高电平1、低电平0、未定态x和高阻态z组成。对于SCALAR类型信号值，可表示为1p，1表示信号值，p表示信号名，而对于VECTOR类型信号值，可表示为1x0z1 p，1x0z1表示信号值，该信号值位宽为5，p表示信号名。

6.2.3 数字波形信号压缩格式

针对数字波形信号的压缩和解压缩最关键的是对信号值变化部分的处理，将*TRB*作为仿真器输出基本数据块，先将信号发生变化的仿真时间点*t*依次存储到一个时间数据流(*time_stream*)里，再对该数据流

进行压缩存储。针对*VTR*的处理，可先对信号名*id*进行压缩处理，初始化一个对照表(*TRSB_accessid*)。设置对照表为空，然后开始遍历*TRSB*，去对照表里查询。如果在对照表里没有查询到则插入*TRSB*到表中，并返回一个访问索引值(*accessid*)，写入*accessid*到访问索引值数据流(*accessid_stream*)中，如果在表中查询到了则返回对应的*accessid*并写入到*acceessid_stream*中。

数字波形信号压缩格式应支持随机访问，支持不是从0时间点开始顺序访问的格式，可跳转到某个时间点开始检索数据。

数字波形信号压缩格式应支持快照模式，支持差分和绝对值两种情况，以适应信号的插入、修改等操作。

数字波形信号压缩方案应先进行编码，将低电平编码成二进制的00，高电平编码成01，未定态编码成10以及高阻态编码成11。可观察到数字信号值里最频繁出现的是高电平和低电平，编码后信号值的二进制表示中的奇数位往往都是0，针对该特点可将信号值编码的数据提取32bit，放置奇数位在高16bit，放置偶数位在低16bit，再借助变长编码进行压缩。变长编码后的数据还可进一步借助如zip类压缩算法进行二次压缩。使用以上基础编码压缩方法，有两种有效的压缩方案。

方案1：基于历史信号值的预测压缩方法。核心需要一个历史数据表来存储信号值的历史数据。先考虑对SCALAR类型信号的预测方法，该类型信号的位宽始终为1，如果在当前仿真时间点的信号值为0，则下一次仿真时间点的同一个信号值则会大概率变为1。如果遇到高阻态z和未定态x，则预测下一个时间点的同一个信号值会变成0。对于VECTOR类型信号的压缩方法，该类型信号长度大于1，推测下一个时间点的信号值即为最低有效位进行翻转后的历史信号值，其中最低有效位指最高的非0位及随后的低位，例如：0b0010111的最低有效位为0b10111。将预测值写入到预测数据流(*predict_stream*)中，并将当前时间点原始信号值写入原始数据流(*original_stream*)中，同时更新表，将读取处理完毕后得到的*predict_stream*和*original_stream*进行异或(XOR)运算后得到最后的数据流(*xor_stream*)，这样得到的*xor_stream*里会有很多数据位等于0。

方案2：基于混合历史信号和信号集的预测处理方法。每一个*TRSB*经过第一个方案后得到了*xor_stream*，可将之记录在一个信号集数据表中，根据表中信号集历史数据*data_stream*与当前数据*xor_stream*再进行异或运算得到*xor_xor_stream*，再将*xor_xor_stream*和*xor_stream*里数据位等于0较多的一个作为结果写入到输出数据流，并同时写入一个标记位到标记数据流(*token_stream*)，最后更新表，将*xor_stream*写入到表中。

根据方案1，对仿真器输出的*TRB*中的信号值进行相应的压缩得到*output_stream*，可得到如图8所示的压缩数据块存储格式。

N	*output_stream*	*time_stream*	*accessid_stream*

说明：

1—— *N* 表示仿真时间点个数以及访问索引值的个数，即*time_stream*中*t*的个数和*accessid_stream*中*accessid*的个数。为了区别不同的数据流，需要在数据流之前存储该数据流的大小，例如在*output_stream*之前需要存储*output_stream*的大小。

图 8 方案 1 的压缩数据块存储格式

对于*time_stream*可采用时间点之间作差的方法，只需要存储第一个时间点的数值以及后续时间点之间的差值，由于仿真时间点之间差值较小，会使得存储的数值较小，从而能采用变长压缩方法进行高效的压缩，最后再对之使用二次压缩方法，如字典转换算法、上下文数据转换进行压缩。在通常情况下对照表本身较小，因此*accessid*的数值不会很大，故可用变长压缩后再二次压缩的方法进行压缩。对于方案 1， *output_stream* 存储的是 *predict_stream* 和 *original_stream* 异或运算的结果 *xor_stream*， 再对 *output_stream*运用变长编码后再二次压缩即可。

根据方案2的压缩方法，对仿真器输出的*TRB*中的信号值进行相应的压缩得到*outpu_stream*，需要还存储的有标记数据流*token_stream*，因此得到如图9所示的存储格式。

N	*output_stream*	*time_stream*	*accessid_stream*	*token_stream*

说明：

1——见图8说明1。

图 9 方案 2 的压缩数据块存储格式

对于*token_stream*，无法保证提取一段32bit或者64bit的数据，其高位都为0，直接采用字典转换算法、上下文数据转换压缩存储即可。此处的*output_stream*存储的是*xor_steam*或者*xor_xor_stream*，以*token_stream*存储的标记作为区分，压缩存储方式同上。

在实际情况下，逻辑仿真器输出的数字波形数据要更加复杂，如VCD文件不仅包含了波形信号值变化数据，还包含了波形信号名、信号类型、信号位宽、层级等仿真过程中十分重要的头部信息。对这部分信息主要是读取识别数字信号波形里存在的少量REAL类型的数据，由于该类数据以实数表示，因此无法使用编码压缩方案，将实数类型的信号名收集到一个容器里，最后将这类数据进行额外存储即可。且这部分信号值在VCD文件当中占较小的比例，不会占用很多存储。此外还需要识别在头部信息里信号名和对应的信号位宽，将其作为一个映射表存储起来，最后等头部信息全部读取识别完毕，将读取的全部头部信息存储在一个字符串里进行表示，再直接对该字符串运用第三方压缩方法，如字典转换算法、上下文数据转换压缩存储。由于头部信息占VCD文件较小的一部分，可运用bzip2进行压缩得到对应数字波形文件的压缩文件存储格式如图10所示。此外，在波形文件存储格式中可增加Version字段，双字节，分主、次版本号信息，放在头部信息的开始之处。

h_size	*header info*
write_times	*pos*
$CVTRB_0$	
$CVTRB_1$	
$CVTRB_2$	
...	
TRSB_accesid	

图 10 数字波形压缩文件的存储格式

压缩文件的第一部分是经过二次压缩后的头部信息及其大小*h_size*，第二部分是两个整形数值*write_times*和*pos*，第三部分就是多个压缩数据块，第四部分就是对照表*TRSB_accessid*。为了方便解压缩算法的运行，*write_times*指代压缩数据块部分包含了几块如图8或者图9格式的压缩数据，*pos*是文件指针的地址，指向*TRSB_accessid*。*write_times*在文件中的偏移量可通过头部信息的大小获得。

6.2.4 数字波形压缩解压算法

6.2.4.1 适用于逻辑仿真器的高效数字波形压缩算法

数字波形压缩过程的输入为从逻辑仿真器输出信号值变化的数据$D=[TRB_i,...,TRB_j]$，D满足一定的条件以防止占用过多的内存开销，例如D中的*TRB*的个数不超过某一阈值或D中*TR*的总数不超过某一阈值，其输出为数字波形文件对应的压缩文件F。压缩过程的描述如算法1所示。

算法1：适用于逻辑仿真器的高效数字波形压缩算法

步骤 1：初始化 *output_stream*，*predict_stream*，*original_stream*，*xor_stream*，*accessid_stream*，*time_stream*，*write_times*；

步骤 2：*While simulation is not end do*；

步骤 3：For *k in i,i+1,…,j* do；

步骤 4：从信号值变化数据的 TRB_k 中得到时间 *t* 以及 T_k，添加时间 *t* 到 *time_stream* 当中，清空临时变量 *TRSB*；

步骤 5：*For l in 0,1,…,size(T_k) do*；

步骤 6：从 T_k 中得到 TR_l；

步骤 7：从 TR_l 中得到信号名 *id* 并添加到 *TRSB* 当中去；

步骤 8：*if type(id) == REAL then*；

步骤 9：把 TR_l 中对应的信号值变化 *v* 进行读取后额外存储起来为 *vreal*；

步骤 10：*else*；

步骤 11：把 TR_l 中对应的信号值变化 *v* 进行编码后写入到 *original_stream*，查询 *history_table*，如果查询到了记录过该信号名的信号值，则取出历史的信号值，并将其进行预测后写入到 *predict_stream* 当中，同时更新 *history_table*，如果没有查询到，则编码后的信号值也写入 *predict_stream*，再更新 *history_table*；

步骤 12：通过对 *predict_stream* 和 *origina_stream* 进行异或运算并写入结果到 *xor_stream*；

步骤 12：*end if*；

步骤 13：*end for*；

步骤 14：在对照表 *TRSB_accessid* 里查询对应的 *TRSB*，如果查询到了，则记录对应的 *accessid* 为索引值，写入到 *accessid_stream* 中，如果没有查询到，则插入 *TRSB* 到对照表 *TRSB_accessid* 当中，同时生成一个 *accessid* 并将之写入到 *accessid_stream*；

步骤 15：*end for*；

步骤 16：*if size(xor_stream)+size(output_stream)<size then*；

步骤 17：将 *xor_stream* 写入到 *output_stream*；

步骤 18：*else*；

步骤 19：先对 *time_stream*，*accessid_stream* 进行变长编码后，再使用二次压缩方法，如字典转换算法、上下文数据转换分别对 *output_stream*，*time_stream*，*accessid_stream* 进行二次压缩；

步骤 20：拼接压缩后的 *output_stream*、*time_stream*、*accessid_stream* 形成压缩数据块并写入到压缩文件 *F* 中，*write_times*++；

步骤 21：清空后重新写入当前对应的数据到 *output_stream*，*accessid_stream*，*time_stream*；

步骤 22：*end if*；

步骤 23：清空 *predict_stream*，*original_stream*，*xor_stream*；

步骤 24：*end while*；

步骤 25：将对照表 *TRSB_accessid* 经过变长压缩后再通过二次压缩后写入到压缩文件 *F* 中，再将 *vreal* 也写入到压缩文件 *F* 中，移动文件指针，写入 *write_times* 和 *pos*。

注1：算法1的前提是压缩文件已经写入头部信息、*write_times*以及*pos*，算法1的输入是逻辑仿真器的输出数据，算法1的第2步是当仿真没有结束，不断接收来自仿真器的输出数据。

注2：算法1的第3步则是遍历数据里的*TRB*，而第5步则是遍历单个*TRB*里的信号值变化*TR*。

注3：算法1的第7、14步就是本文件第6.2.2节信号名的压缩方法，而第11、12步就是本标准第6.2.2节中信号值的压缩方案1，*history_table*表示历史数据表。

注4：算法1的第16~22步主要是将数据压缩并拼接成数据块再写入到压缩文件中去，其中第16步的参数*Size*是预设的*output_stream*大小。

注5：算法1的第25步是将对照表*TRSB_accessid*、REAL类型的信号值大小*vreal*、整形数值*write_times*和*pos*写入到压缩文件对应位置。

注6：针对6.2.2节第二种信号值压缩方案以及图5所对应的压缩数据块存储格式，可在第14步和第15步之间插入第2.2小节压缩方案2，即将第12步得到的*xor_stream*和数据表中对应*accessid*的*data_stream*进行异或运算得到*xor_xor_stream*，最后得到*bit*=1较少的结果记为*final_stream*写入到*output_stream*中，同时更新标识符到*token_stream*，可再改写算法1第16~23步，将*xor_stream*替换成*final_stream*，并将第19~21步数据块的组成加上*token_stream*即可。

6.2.4.2 适用于逻辑仿真器的高效数字波形解压缩算法

解压缩过程的输入为执行算法1得到的压缩文件F，输出为解压得到波形信号值变化的数据$D=[TRB_i,...,TRB_j]$。解压过程要注意对头部信息的处理以及对照表的解压缩，具体描述如算法2所示。

算法2：适用于逻辑仿真器的高效数字波形压缩算法

步骤 1：对头部信息进行读取解压缩处理获得信号长度信息、信号类型，然后再读取了 *write_time* 和 *pos* 后，移动文件指针从压缩文件 *F* 最后面读取解压得到对照表信息 *accessid_TRSB* 以及 REAL 数据信息 *vreal*，再移动文件指针回压缩数据块开始位置；

步骤 2：*While write_times≠0 do*；

步骤 3：读取压缩数据块并使用对应的变长解压缩以及二次解压缩，如字典转换算法、上下文数据转换算法解压得到 *N*，*output_stream*，*time_stream*，*accessid_stream*；

步骤 4：*For k in 0,1..., N do*；

步骤 5：从 *time_stream* 里得到时间 *t*，并添加时间 *t* 到 TRB_k 当中.从 *accessid_stream* 里得到访问索引值 *accessid*；

步骤 6：根据 *accessid* 从 *accessid_TRSB* 里得到 *TRSB*，根据 *TRSB* 中数据总长度从 *output_stream* 提取数据到 *final_stream* 中；

步骤 7：*For l in 0,1..., sie(TRSB) do*；

步骤 8：从 $TRSB_l$ 里获得信号名 *id*；

步骤 9：*if type(id)=REAL then*；

步骤 10：从 *vreal* 中得到对应的数据，得到信号值 *v*，再结合信号名 *id* 恢复成信号变化 TR_l，添加 TR_l 到 TRB_k 当中；

步骤 11：*else*；

步骤 12：根据读取头部信息获得的信号长度信息，从 *final_stream* 里取出对应长度的位数据，存储为 *xor_data*；

步骤 13：再查询 *history_table*，如果查询到了记录过该信号名的信号值，则取出历史的信号值 *pre_val*，并将其与 *xor_data* 进行对应预测解码后得到信号值 *v*，同时更新 *history_table*，如果没有查询到，则解码后的信号值为 *v*，更新 *history_table*。由信号值 *v*，再结合信号名 *id* 恢复成信号变化 TR_i，添加 TR_i 到 TRB_k 当中；

步骤 14：*end if*；

步骤 15：*end for*；

步骤 16：形成 TRB_k；

步骤 17：*end for*；

步骤 18：形成数据流 $D=[TRB_i,...,TRB_j]$；

步骤 19：*write_times--*；

步骤 20：*end while*。

注1：算法2的输入是压缩文件F，按照压缩文件的存储格式对头部信息先进行读取并解压缩获得信号长度信息、信号类型信息，然后读取*write_times*和*pos*后，根据*pos*移动文件指针从而读取对照表信息，最后再读取压缩数据块信息。故算法2的第3步是读取压缩数据块，第4、7步是遍历得到对应的TRB_k的循环，第13步则是对应算法1的第11步的预测解压缩方法，即如果是SCALAR类型信号，则应按照表1解压。

表 1 基于信号值历史的 SCALAR 类型信号解压方法

pre_val	*xor_data*	*v*
0	0	1
0	1	0
0	x	z
0	z	x
Others	m	m

注2：如果*pre_val*=0，*xor_data*=1，则表示预测失败，实际信号值应为0，其他情况可依次分析得到，而最后*pre_val*=Others指的是当上一个时间点信号值为1或x或z时，下一个信号值都应预测为0，任何实际值m和0异或运算结果都是m. VECTOR类型信号可按照异或运算类型可得，最后算法2的输出是解压得到数据$D=[TRB_i,...,TRB_j]$。

注3：算法2是对应第一种压缩方案的解压缩算法，而对应第二种压缩方案的解压缩算法只需要对算法2的第3步压缩数据块的解码进行修改并在第6步和第7步之间插入第二种解压方案。

注4：对于第3步压缩数据块的解码，只需要按照图5的格式进行解码，并得到额外的*token_stream*即可。

注5：在第6步之后，根据*token_stream*得到当前TRB_k对应的标识符，判断将第6步得到的*final_stream*与数据表中对应*accessid*的*data_stream*进行异或运算的结果作为后续第12步的*final_stream*，或将第6步得到得*final_stream*直接作为后续第12步的*final_stream*。

注6：假设波形数据包含n个*TRB*，平均每个*TRB*包含m个*TR*，则上述数字波形压缩与解压缩的参考算法的时间复杂度为O(mn)，空间复杂度为O(m)。

附 录 A

（资料性）

三阶段流水线压缩方法说明

A.1 模拟波形三阶段流水线压缩方法

模拟波形压缩是逐个数据块进行的，每个数据块的处理可简单视为三级流水线，如图A.1所示。第一阶段中数据块根据压缩算法进行编码，第二阶段中使用Deflate算法进一步压缩，第三个阶段中将压缩块写入磁盘文件。在并行计算中，两个相邻阶段之间的关系可描述为典型的生产者和消费者模型，压缩算法可通过三个线程并行化。与电路仿真相比，波形压缩所消耗的时间要短很多，来自仿真器的信号值的生成要比执行压缩流水线中的每个阶段慢很多，并行压缩方法几乎不会增加整个模拟时间。

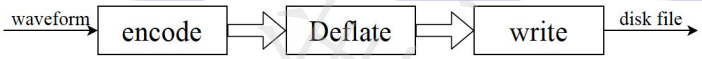


图 A.1 模拟波形三阶段流水线压缩方法示意图

类似地，解压缩过程也可采用如图A.2所示的并行流水线方式，显著减少解压缩的时间。

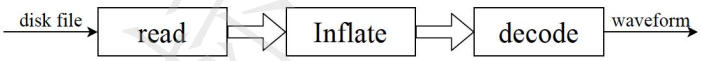


图 A.2 模拟波形三阶段流水线解压缩方法示意图

A.2 数字波形三阶段流水线压缩方法

根据本标准提出的数字波形的高效压缩存储格式、压缩文件格式以及适用于仿真器的数字波形压缩算法、解压算法，可将整个压缩过程按照三阶段流水线的模式连起来，见图A.3。

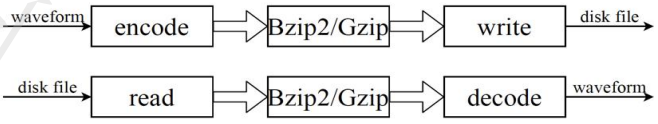


图 A.3 数字波形三阶段流水线压缩解压方法示意图

通过使用并行计算，可进一步减少算法1和算法2的运行时间。波形压缩的主要过程是逐个数据块进行的，每个数据块的处理可简单地视为三阶段流水线。在第一阶段中，使用算法1里3～15步的压缩方法形成相应的数据流，然后在第二阶段中使用二次压缩算法，如字典转换算法、上下文数据转换压缩数据块，最后将压缩数据块写入磁盘文件. 在并行计算中，两个相邻阶段之间的关系可描述为典型的生产者-消费者模型，因此算法1可很容易地用三个线程并行化。与电路仿真相比波形压缩所消耗的时间要少得多，因此来自仿真器的信号值变化数据的生成比执行压缩流水线中的每个阶段要慢得多，这意味着并行压缩方法几乎不会增加整个仿真的时间。同样可设计出类似的并行解压程序，利用它也可减少解压缩的时间。

参 考 文 献

[1] GB/T 9178 集成电路术语