|  |  |
| --- | --- |
| **Сущность и её архитектура** | **Testbench сущности** |
| library IEEE;  use IEEE.STD\_LOGIC\_1164.all;  entity and2 is  port(  x : in STD\_LOGIC\_VECTOR(0 to 1);  y : out STD\_LOGIC  );  end;  architecture and2\_arch of and2 is  begin  y <= x(0) and x(1);  end and2\_arch; | library IEEE;  use IEEE.STD\_LOGIC\_1164.all;  entity and2\_tb is  end and2\_tb;  architecture TB\_ARCHITECTURE of and2\_tb is  component and2  port(  x : in STD\_LOGIC\_VECTOR(0 to 1);  y : out STD\_LOGIC  );  end component;  signal x : STD\_LOGIC\_VECTOR(0 to 1);  signal y : STD\_LOGIC;  begin  UUT: and2 port map (  x(0) => x(0),  x(1) => x(1),  y => y  );  x0\_process: process  begin  x(0) <= '0';  wait for 1 ns;  x(0) <= '1';  wait for 1 ns;  end process;  x1\_process: process  begin  x(1) <= '0';  wait for 2 ns;  x(1) <= '1';  wait for 2 ns;  end process;  end TB\_ARCHITECTURE; |

**ОЧЕНЬ СИЛЬНО ХРОМАЕТ ТЕРМИНОЛОГИЯ НА РУССКОМ, НАЙТИ ОПОРНЫЙ ИСТОЧНИК!!!!!**

Разбор VCD-waveform, который генерируется GHDL

К сожалению, не существует стандартного формата файлов wave, поддерживающего все базовые типы VHDL, поэтому используется или формат VCD, заимствованный из стандарта Verilog-HDL и частично поддерживающий базовые типы (описано далее), или проприетарные форматы, такие как GHW (GHDL Waveform) от GHDL, ASDB (A Simulation DataBase) от ACTIVE-HDL, SAIF (Switching Activity Interchange format) Xilinx WebPACK ISE, WLF (Wave Log Format) от ModelSim. [www.aldec.com/resources/manuals/Active-HDL/cmd00101.htm](https://www.aldec.com/resources/manuals/Active-HDL/cmd00101.htm)

**Далее будет рассмотрен именно формат VCD, описание которого взято из "IEEE Standard for Verilog Hardware Description Language", IEEE Std 1364-2001, стр. 324-349.**

VCD (value change dump) – Файл дампа изменений значений, ASCII-файл в человекочитаемом и машиночитаемом формте, содержит информацию об изменениях значений выбранных переменных в проекте, хранящуюся в задачах системы дампа изменений значений, а также дополнительные метаданные. Существует два типа файлов VCD:

а) Стандартный (four state VCD): представление изменений переменных в 0, 1, x ((неопределенное состояние) и z (выключено).

b) Расширенный (Extended VCD): представление изменений переменных во всех состояниях, как и в обычном VCD, а также информации о силе сигнала (0-highz, 1-small, 2-medium, 3-weak, 4-large, 5-pull, 6-strong, 7-supply) и некоторые дополнительные данные.

Далее речь пойдёт про стандартный формат, поскольку он более распространён и используется в GHDL.

Поскольку изначально VCD был разработан для Verilog-HDL, он может поддерживать дамп значений лишь сигналов, основанных на базовых типах:

* из пакета std.standard:

– bit

– bit\_vector

* из пакета ieee.std\_logic\_1164:

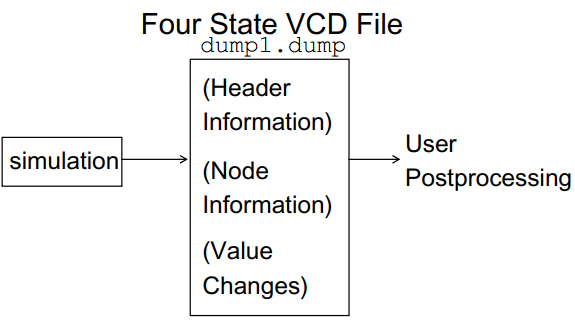
– std\_ulogic

– std\_logic (является подтипом std\_ulogic)

– std\_ulogic\_vector

– std\_logic\_vector

* любой целочисленный тип



Файл VCD начинается с информации в заголовке (может быть опущена), содержащей дату, номер версии симулятора, используемого для моделирования, и используемая временная шкала.

|  |  |
| --- | --- |
| **$date**  Wed May 29 22:06:15 2024  **$end** | Метаданные о времени создания VCD. |
| **$version**  GHDL v0  **$end** | Метаданные об инструменте, создавшем VCD и его версии. |
| **$timescale**  1 fs  **$end** | Метаданные о шкале времени.  Шкала времени включает в себя шаг времени (1, 10 или 100), за которым следует единица измерения (s, ms, us, ns, ps, fs).  В случае пропуска отображение в специализированных программах непредсказуемо (в GTKWave автоматически будет выбрано 1ms, а в vs.drom будет оборажаться NaN).  GHDL всегда проставляет шаг времени 1fs. |

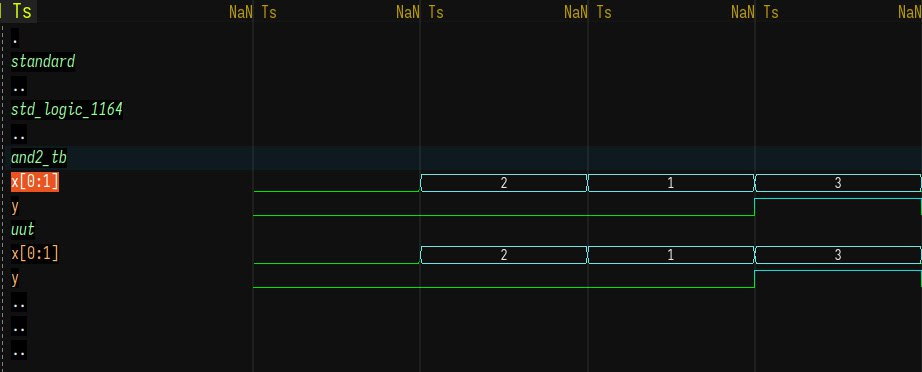


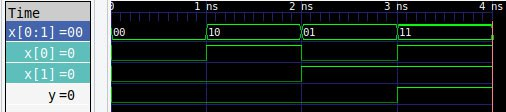
Рисунок с NaN (vc.drom отображает битовые вектора в виде десятичного числа)

Далее файл содержит определения области действия и типа сбрасываемых переменных

|  |  |
| --- | --- |
| **$scope** module standard **$end**  **$upscope $end**  **$scope** module std\_logic\_1164 **$end**  **$upscope $end**  **$scope** **module** and2\_tb **$end**  **$var** reg 2 ! x[0:1] **$end**  **$var** reg 1 " y **$end**  **$scope** **module** uut **$end**  **$var** reg 2 # x[0:1] **$end**  **$var** reg 1 $ y **$end**  **$upscope $end**  **$upscope $end** | Переменные определяются иерархически с указанием области видимости.    **Переменные объявляются по формату $var** vary\_type size identifier\_code reference **$end** |

var\_type ::= event | integer | parameter | real | reg | supply0 | supply1 | time | tri | triand | trior | trireg | tri0 | tri1 | wand | wire | wor

существует множество типов, в 99% случаев используется reg



Корректная waveform

|  |  |
| --- | --- |
| $enddefinitions $end | раздел $enddefinitions отмечает конец информации и определений заголовка |

От конца заголовка до конца файла идет раздел данных. Этот раздел содержит строки двух типов: строки времени моделирования и строки изменения значений.

|  |  |
| --- | --- |
| #0  b00 !  0"  b00 #  0$  #1000000  b10 !  b10 #  #2000000  b01 !  b01 #  #3000000  b11 !  1"  b11 #  1$  #4000000  b00 !  0"  b00 #  0$ | Временные линии моделирования начинаются с # и значения времени (unsigned number)  Строки изменения значения содержат значение, которое принимает переменная, за которым следует идентификационный код переменной, присвоенный в заголовке. Эти строки необходимы только при каждом изменении значения, о котором идет речь.  Для однобитовых значений значение в строке изменения значения состоит из 0, 1, x или z, за которыми следует идентификационный код, присвоенный этому значению в заголовке. Для многоразрядных значений перед всеми битами стоит буква b. Если заданы не все биты, то значение расширяется влево беззнаковым способом. |

<https://github.com/tcr/node-vcd> - js парсер vcd

Синтакс VCD four state (можно вставить в приложение к НИР)

value\_change\_dump\_definitions ::=

{ declaration\_command }{ simulation\_command }

declaration\_command ::=

declaration\_keyword [ command\_text ] **$end**

simulation\_command ::=

simulation\_keyword { value\_change } **$end**

| **$comment** [ comment\_text ] **$end**

| simulation\_time

| value\_change

declaration\_keyword ::= $comment

| **$date**

| **$enddefinitions**

| **$scope**

| **$timescale**

| **$upscope**

| **$var**

| **$version**

simulation\_keyword ::=

**$dumpall**

| **$dumpoff**

| **$dumpon**

| **$dumpvars**

simulation\_time ::=

# decimal\_number

value\_change ::=

scalar\_value\_change

| vector\_value\_change

scalar\_value\_change ::=

value identifier\_code

value ::=

**0 | 1 | x | X | z | Z**

vector\_value\_change ::=

**b** binary\_number identifier\_code

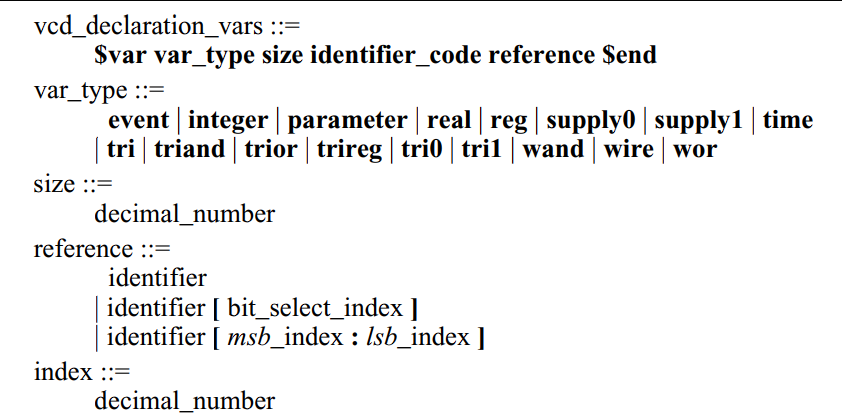
| **B** binary\_number identifier\_code

| **r** real\_number identifier\_code

| **R** real\_number identifier\_code

identifier\_code ::=

{ ASCII character }



<https://cyberleninka.ru/article/n/formirovanie-opisaniy-komponentov-dlya-vnutrikristalnoy-otladki-tsifrovyh-ustroystv-i-vstraivaemyh-mikroprotsessornyh-sistem-na-3>

<https://cyberleninka.ru/article/n/kompleks-programmnyh-sredstv-dlya-predvaritelnoy-otsenki-kachestva-testov-tsifrovyh-shem/viewer>