

---

# Cache Debugger使用手册

---

Cache小组

2015年1月9日

目录

# 1 引言

## 1.1 编写目的

编写本使用说明的目的是充分叙述本工具所能实现的功能及其运行环境，以便使用者了解本工具的使用范围和使用方法，并为软件的维护和更新提供必要的信息。

# 2 概述

## 2.1 用途

Cache Debugger是一个基于串口的VHDL通用调试工具。使用本工具可以在PC端发送指令到硬件系统，具有启动、断点、继续运行、单步调试、查看信号等多种功能。

## 2.2 系统配置

带有串口的目标设备

支持VHDL语言和该目标设备的集成软件开发环境

Unix或类Unix操作系统，安装有串口设备驱动，支持C++11的编译器

# 3 硬件部署

以CaCPU为例叙述如何将本工具应用于被测试模块。

首先在被测试模块的实体加入两个输出信号：被查看信号和断点条件。

```
entity cacpu is
    port(
        -- clock
        clk : in std_logic;
        -- reset
        e : in std_logic;
        +      -- debug tool output
        +      debug_out_datas : out std_logic_vector(4095 downto 0);
        +      -- debug tool breakpoint condition
        +      monitor : out std_logic_vector(63 downto 0);
        -- other ports
        baseram_addr: out std_logic_vector(19 downto 0);
        baseram_data: inout std_logic_vector(31 downto 0);
        ...
    );
```

```
end cacpu;
```

在结构体的并发处理语句中定义被查看信号和断点条件。被查看信号的定义可使用工具自动生成，默认8对齐，未使用的用0填充。断点条件未使用的部分必须用0填充。

在本例中被查看信号共4096位，断点条件64位。被查看的信号state类型为std\_logic\_vector，共3位，用0填充5位以实现8对齐。被查看的信号clk\_cpu类型为std\_logic，占1位，用0填充7位。断点条件低32位为CPU当前PC，接着3位是state信号，剩余位用0填充。

```
architecture bhv of cacpu is
+signal out_datas : std_logic_vector(4095 downto 0);
...
begin
...
+  debug_out_datas <= out_datas;
+  -- output signals
+  out_datas(2 downto 0) <= state(2 downto 0);
+  out_datas(7 downto 3) <= (others => '0');
+  ...
+  out_datas(32) <= cpu_clk;
+  out_datas(39 downto 33) <= (others => '0');
+  ...
+  out_datas(495 downto 464) <= general_values(31 downto 0);
+  out_datas(527 downto 496) <= general_values(63 downto 32);
+  out_datas(4095 downto 528) <= (others => '0');
+  -- define breakpoint
+  monitor(63 downto 35) <= (others => '0');
+  monitor(34 downto 32) <= state;
+  monitor(31 downto 0) <= this_PC;
...
end bhv;
```

在新的顶层模块中原件例化被调试的模块，并将输出端口对应起来。

```
entity cadb is
  port(
    -- clock, use by cadb, can be shared
    clk: in std_logic;
    -- reset, used by cadb, can be shared
    e: in std_logic;
+  -- ports of module debugged
+  baseram_addr: out std_logic_vector(19 downto 0);
```

```

+     baseram_data: inout std_logic_vector(31 downto 0);
+     ...
    -- serial port, used by cadb, cannot be shared
    serialport_txd: out  STD_LOGIC;
    serialport_rxd: in   STD_LOGIC;
);
end cadb;
architecture bhv of cadb is
+-- module debugged
+component cacpu is
+    port(
+        clk: in std_logic;
+        e: in std_logic;
+        debug_out_datas: out std_logic_vector(4095 downto 0);
+        monitor: out std_logic_vector(63 downto 0);
+        -- other ports
+        baseram_addr: out std_logic_vector(19 downto 0);
+        baseram_data: inout std_logic_vector(31 downto 0);
+        ...
+    );
+end component;

    signal send_data: std_logic_vector(4095 downto 0);
    signal m_monitor: std_logic_vector(63 downto 0);
    signal bp_clk: std_logic;
    begin
+        u1: cacpu port map(clk=>bp_clk, e=>e,
+            debug_out_datas=>send_data,
+            monitor => m_monitor,
+            baseram_addr=>baseram_addr, baseram_data=>baseram_data,
+            ...
+        );
    end bhv;

```

应将被调试模块的原时钟连接到bp\_debug模块的输入时钟，将bp\_debug模块的输出时钟连接到被调试模块的时钟。  
为cadb分配串口管脚。

## 4 软件部署

首先编写配置文件，配置文件共三列。

第一列是软件端查看变量值时的标签，接受不包含空白字符的字符串。

第二列是VHDL源代码中实际被查看信号名称，接收不包含空白字符的字符串，在被调试模块中应有同名信号，且该信号不能为输出端口。

第三列是被查看信号的范围。对于std\_logic\_vector应标明范围，对于std\_logic第三列空。

运行软件端程序cadb gen <目标文件>，信号定义部分VHDL代码将自动生成并保存在目标文件中。

每次更新配置文件后都应该重新生成VHDL代码并将其更新到硬件中。

运行cadb <串口设备路径>，调试工具将启动。

## 5 使用方法

软件端程序启动后应该出现如图??所示界面，此时输入命令help可查看帮助。

图 1: 软件客户端界面

```
Welcome to Cache Debugger(Version 1.0).
Commands end with enter.
Type 'help' for help. Type Ctrl + D or 'quit' to quit.
Copyright (c) 2014 Jamis Hoo, Terran Lee, Hao Sun.
Distributed under the MIT LICENSE.
Loading configuration...
Load configuration successful.
cadb> help
run                : start running.
break <address>    : set breakpoint at <address>, globbing('_') is supported.
continue           : continue running.
step [n]           : run n clocks, n is default 1.
print[b|x] <label> : print value of <label> in binary or hexadecimal(default).
display[b|x] <label> : automatically print value of <label> each time program stops.
undisplay <label>  : remove item <label> from auto-display list.
help               : print usage information of Cache Debugger.
quit               : quit.
cadb> █
```

所有支持的命令：

1. run: 开始运行。
2. break <value>: 设置断点，支持十六进制（0x）和二进制（0b）。
3. continue: 继续运行。
4. step [n]: 前进n个时钟，n默认为1，支持二进制、八进制、十进制、十六进制。
5. print[b|x] <label>: 以二进制（b）或十六进制打印变量。默认为十六进制。

6. `display[b|x] <label>`: 自动打印变量。
7. `undisplay <label>`: 取消所有该变量的自动打印。
8. `help`: 显示帮助信息。
9. `quit`: 退出。

输入`run`相当于被调试模块得到一个`reset`信号后得到连续的时钟信号，被调试模块应该在此条件下能正确运行。

输入`run`、`step`、`continue`命令后程序会阻塞，直到到达断点条件，用户才可继续操作，此时可使用`print`命令查看信号值。

断点设置有高级用法，参见技术文档。

## 6 维护

如果在使用中遇到问题请仔细阅读使用手册和技术文档，确保使用方法正确，满足设计约定。如果问题仍然不能解决，请联系作者，联系方式请咨询本使用手册提供者。