Реализация возможности пошаговой отладки при отладке тестовых сценариев на модели СБИС СнК

А.В. Андрианов

ЗАО НТЦ «Модуль», Москва, andrianov@module.ru

Аннотация - При разработке и отладке программных тестов на модели СБИС средства пошаговой и интерактивной отладки в большинстве недоступны, а программные средства отладки, такие как форматированный вывод, требуют существенных ресурсов. Коммерческие программно-аппаратные комплексы для отладки по интерфейсу JTAG в большинстве своем не поддерживают работу на модели СБИС. В статье описана практика применения свободной реализации JTAG отладчика OpenOCD для реализации функции пошаговой отладки на модели СБИС.

Ключевые слова - gdb, jtag, simulation, rtl, система на кристалле (СнК)

1. Введение

При разработке тестовых программ для модели СБИС СнК использование стандартных отладочных средств затруднено или невозможно. Использование программных средств, таких как форматированный вывод, автоматическое инструментирование кода компилятором для трассировки, профилирования, сборки покрытия, а также использование программного сервера отладки gdb, связаны с существенным замедлением процесса моделирования. Существующие решения для пошаговой отладки кода при работе без операционной системы требуют наличия специальной дорогостоящей аппаратуры, а использовать их совместно с моделью СБИС невозможно, так как этот сценарий изначально не предусмотрен производителем.

В данной статье рассматривается реализация функции пошаговой отладки на модели СБИС при помощи программного JTAG сервера OpenOCD совместно с отладчиком GNU Debugger (gdb) и особенности подключения данного сервера к верификационному окружению СБИС.

2. Программные средства

В этой части статьи приведена общая информация об открытых программных компонентах, применение которых будет детально рассматриваться в дальнейшем: OpenOCD[1] и GDB[2].

OpenOCD – проект реализации протокола внутрисхемной отладки JTAG с открытым исходным кодом. OpenOCD поддерживает работу с большим набором процессорных ядер таких архитектур как ARM, MIPS, avr32 и других. Этот программный пакет позволяет использовать широкий набор недорогих интерфейсных адаптеров для подключения к целевой аппаратуре.

OpenOCD чаще всего применяется в режиме сервера, к которому могут подключаться клиенты, например отладчик gdb или telnet для доступа к встроенной в OpenOCD интерактивной tcl среде.

GNU Debugger (GDB) – Универсальный отладчик с открытым исходным кодом, с интерфейсом командной строки, который поддерживает отладку программ на различных языках программирования, включая Си, C++, Free Pascal. Данный отладчик поддерживает удаленную отладку по сети, подключаясь к серверу, реализующему протокол gdb. В роли этого сервера может выступать, в том числе и OpenOCD. Многие популярные интегрированные среды разработки и отладки, такие как eclipse и atom позволяют использовать gdb для предоставления функции интерактивной отладки ПО.

3. Существующие способы отладки программного обеспечения на модели СБИС

В данной части приводится краткий обзор существующих способов отладки тестовых сценариев на модели СБИС, их преимущества и недостатки.

1. *Использование встроенных средств просмотра временных диаграмм и трассировки цифровой схемы (напр. simvision, schematic tracer).*

Главным преимуществом данного способа отладки, является то, что он доступен начиная с самых ранних этапов разработки СБИС. Данный метод отладки позволяет находить аппаратные ошибки. Однако, он мало приспособлен для отладки программного обеспечения, так как не представляет удобных инструментов для просмотра состояния регистров процессора и не позволяет быстро просматривать и

редактировать значения переменных в памяти и регистров процессора.

2. *Отладка с использованием форматированного вывода*

Данный способ отладки требует наличия канала обмена данными между программой, исполняемой на модели СБИС и верификационным окружением, поэтому становится доступен на более поздних этапах, когда уже обеспечена минимальная работоспособность процессорного ядра и подключены соответствующие компоненты верификационного окружения.

Таким каналом может быть как реальный интерфейс, например, блок универсального асинхронного последовательного приемо-передатчика (UART), так и фрагмент внутренней памяти СБИС, который используется одновременно как верификационным окружением СБИС через иерархические ссылки, так и программой, работающей на модели СБИС (разделяемая память).

Применение форматированного вывода приводит к существенному замедлению процесса моделирования на каждое выводимое сообщение. Точные значения варьируются в зависимости от выбранного канала обмена данными между окружением СБИС и тестовой программой, используемым пакетом ПО для моделирования и производительностью сервера.

Для примера, форматирование строки в памяти на тестовой СБИС, содержащей процессорное ядро с архитектурой ARM, работающее на частоте 800Mhz заняло 44688 нс при моделировании, или 62 секунды реального времени. Для теста использовалась реализация функции sprintf (вариант nano) из библиотеки языка C newlib. (замерялось время исполнения вызова *‘sprintf(buf, “Hello world %d %d %d\n”, i,j,k)*’).

3. *Использование гибридного верификационного окружения*

Гибридное верификационное окружение [4] предполагает перенос программы, генерирующей тестовое воздействие, из модели СБИС на хост-компьютер. Такой подход позволяет отлаживать тестовую программу, как если бы это было обычное приложение для ПК. Для обмена данными между моделью СБИС и программой генерации тестового воздействия используется TCP/IP соединение или unix socket. Схематично, возможная реализация такого окружения представлена на Рис. 1. Для обмена данными используется разделяемая память. На моделируемом процессоре исполняется программа-заглушка, выполняющая обращения к регистрам устройств на основе данных в разделяемой памяти.

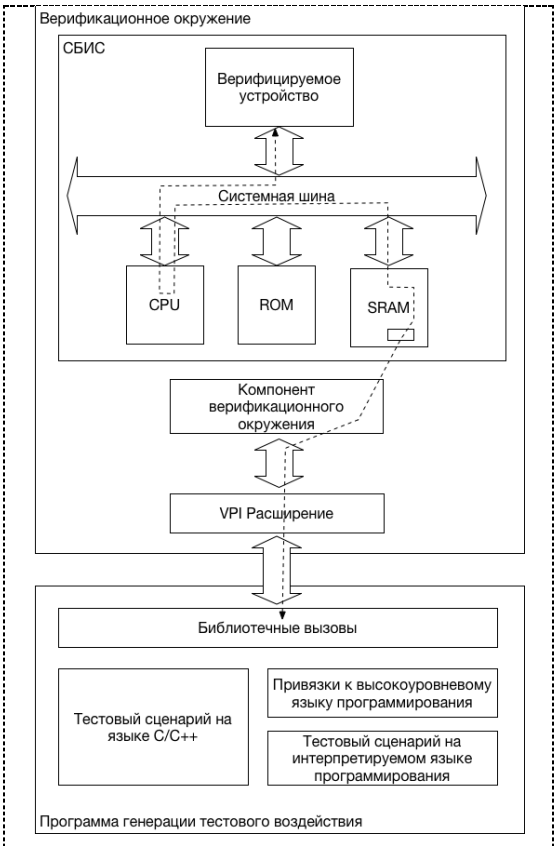

Рисунок 1. Схема гибридного верификационного окружения

К преимуществам гибридного окружения стоит отнести высокую скорость исполнения тестового сценария и доступность всех функций для пошаговой отладки. К недостаткам стоит отнести повышенные требования к качеству кода для сохранения переносимости, и требованию наличию слоя абстракции для доступа к регистрам, памяти и прерываниям.

4. *Использование OpenOCD сервера совместно с отладчиком GDB для отладки программного обеспечения на модели СБИС*

Помимо собственно обеспечения возможности отладки программного обеспечения, этот способ позволяет проверить работоспособность интерфейса JTAG с реальным программным обеспечением.

Для взаимодействия между OpenOCD сервером и верификационным окружением используется TCP/IP соединение, использующее протокол remote bitbang.

Схематично, такое верификационное окружение приведено на Рис. 2.

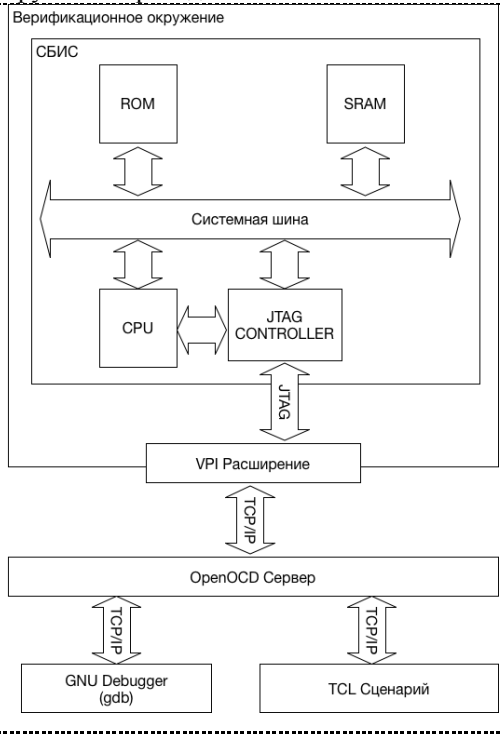

Рисунок 2. Схема подключения OpenOCD в верификационное окружение СБИС

5. *Использование программного gdb сервера*

Как и форматированный вывод, программный gdb сервер требует наличия канала передачи данных. При работе на реальной аппаратуре чаще всего используется блок приемопередатчика UART. Помимо канала передачи данных требуется также VPI расширение для пакета моделирования, которое будет транслировать данные от gdb-сервера, запущенного на модели СБИС в TCP/IP соединение или UNIX socket, к которому может подключиться отладчик gdb.
Такой подход позволяет отлаживать программное обеспечение на модели СБИС даже если поддержка данной архитектуры отсутствует в OpenOCD и применение интерфейса JTAG не представляется возможным.
Недостатком такого способа отладки является меньшая производительность, т.к. функции предоставляемые аппаратурой JTAG приходится эмулировать программно. Также загрузка/чтение больших объемов данных в/из памяти СБИС занимает продолжительное время.
Схематично, такое верификационное окружение представлено на Рис. 3.

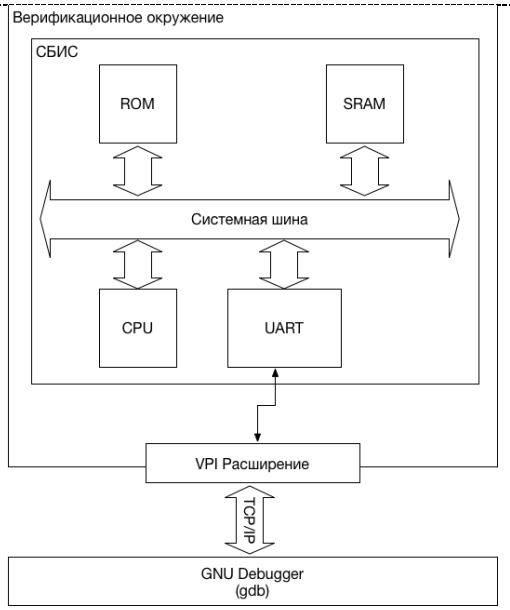

Рисунок 3. Верификационное окружение с подключением gdb

3. Подключение OpenOCD с виртуальным адаптером remote bitbang к модели СБИС

OpenOCD поддерживает широкий набор недорогих JTAG адаптеров, включая адаптеры на основе интерфейсных микросхем FT2232/FT2232H, buspirate и т.п. для работы с аппаратурой. В том числе он поддерживает работу с JTAG адаптером поверх TCP/IP соединения, используя “виртуальный адаптер” под названием “remote bitbang”..

Протокол “remote bitbang” является простым текстовым протоколом, по которому передается состояние линий JTAG интерфейса[11] и может быть задействован для моделирования.

Этот адаптер проще всего задействовать для подключения к верификационному окружению модели СБИС через интерфейс VPI (Verilog Procedural Interface). Необходимость использования VPI обусловлена отсутствием встроенных средств для обработки TCP/IP соединений в стандартной бибилиотеке функций языка Verilog/SystemVerilog. В таком сценарии VPI расширение получает состояние линий JTAG от OpenOCD посредством TCP/IP соединения или используя unix socket и передает их верификационному окружению. Пример кода тестового окружения СБИС, описывающего получение линий JTAG от VPI компонента приведен на Листинге 1.

```
int adap;
reg DBG_TDI_r;
reg DBG_TMS_r;
reg DBG_TCK_r;
reg DBG_TRSTN_r;
reg RSTN_r;
wire DBG_TDO;

initial begin
   adap = $tag_adaptor_create();
end

always @ ( posedge REFCLK ) begin
  if (adap != -1) begin
   $jtag_adaptor_io(adap, DBG_TDI_r, DBG_TDO, DBG_TMS_r, DBG_TCK_r, DBG_TRSTN_r, RSTN_r);
  end
end
```

Листинг 1. Пример подключения VPI компонента в testbench

Особенности такой реализации:

1. VPI функция jtag_adaptor_io не должна быть блокирующей, так как это вызовет остановку процесса моделирования до получения следующего пакета от OpenOCD, а так же может вызвать неработоспособность графического интерфейса simvision.
2. Адаптер “remote bitbang” не поддерживает управление частотой обмена данными по JTAG интерфейсу, максимальная частота сигнала JTAG_TCK задается в верификационном окружении.
3. Так как моделирование и OpenOCD запускаются независимо, то момент времени, когда на линии JTAG будет сгенерировано воздействие, может различаться от запуска к запуску.

Для настройки OpenOCD сервера потребуются два конфигурационных tcl файла. Файл, описывающий конфигурацию адаптера представлен на Листинге 2.

```
interface remote_bitbang
remote_bitbang_host 127.0.0.1
remote_bitbang_port 4000
jtag_ntrst_delay 10000
```

Листинг 2. Пример настройки openocd для использования протокола remote bitbang

Первый конфигурационный файл, описывающий конфигурацию целевой микросхемы представлен на Листинге 3.

```
source [find target/swj-dp.tcl]
swj_newdap $_CHIPNAME cpu -irlen 8 -ircapture 0x1 -irmask 0xf -expected-id $_CPUTAPID
set _TARGETNAME $_CHIPNAME.cpu
target create $_TARGETNAME cortex_a -endian $_ENDIAN -chain-position $_TARGETNAME
```

Листинг 3. Пример минимальной настройки OpenOcd для использования протокола remote bitbang (для процессорного ARM Cortex-A5)

5. Поддержка со стороны средств разработки и отладки

Eclipse[6].

Интегрированная среда разработки Eclipse поддерживает удаленную отладку при помощи gdb и OpenOCD используя расширение “openocd debugging plugin”[5], доступное в репозитории расширений.
Перед началом отладки необходимо убедиться, что запущен процесс моделирования СБИС.

Atom[7]

Atom – бесплатный текстовой редактор с открытым кодом. Интерактивная отладка доступна при помощи расширений atom-swd-debugger и atom-gdb-debugger. В отличие от eclipse, эти расширения при настройке требуют запуска пользователем не только процесса моделирования СБИС, но и OpenOCD сервера.

Литература

1. http://openocd.org/ (Дата обращения 01.04.2018)
2. https://www.gnu.org/software/gdb/ (Дата обращения 01.04.2018)
3. https://gnu-mcu-eclipse.github.io/debug/openocd/ (Дата обращения 01.04.2018)
4. “Методика гибридной верификации СБИС “Система-на-Кристалле”, Андрианов А.В., Шагурин И.И. "Датчики и системы" , 2018г., №2, с. 14-18.
5. https://gnu-mcu-eclipse.github.io/debug/openocd (Дата обращения 01.04.2018)
6. https://www.eclipse.org/ (Дата обращения 01.04.2018)
7. https://atom.io/ (Дата обращения 01.04.2018)
8. Debugging with GDB: The GNU Source-Level Debugger. Free Software Foundation, 9th edition - Stallman, Pesch, et al. - 2002
9. GDB Internal Manual: A guide to the internals of the GNU debugger. Free Software Foundation - Gilmore, Shebs, et al.
10. Abstract Open On-Chip Debugger. Hubert Högl , Dominic Rath , Fachhochschule Augsburg , Fakultät Für Informatik
11. https://github.com/ntfreak/openocd/blob/master/doc/manual/jtag/drivers/remote_bitbang.txt (Дата обращения 01.04.2018)