montavista™

Debugging linux kernel with JTAG

Anna Dushistova, Alexandre Rusev

MontaVista Software Inc.

Содержание

- Инструменты отладки на уровне ядря в Linux
- Что такое JTAG?
- Принципы работы.
- Типичные примеры использования JTAG
- Обзор программных и аппаратных инструментов JTAG
- Примеры использования Eclipse + CDT Hardware Debug Plugin
- Вопросы и ответы

Инструменты отладки в на уровне ядра linux

1) Kernel debugger, kdb

- Реализован в виде патча к ядру;
- Позволяет производить отладку на той же машине
- Позволяет просматривать/модифицоровать память и регистры, устанавливать точки останова, произовдить трассировку стека

2) Kernel GNU debugger, kgdb

- включен в ядро с версия 2.6.26
- требует наличия машины разработки и целевой машины, соединенных по RS-232
- работает совместно с gdb. (gbd на машине разработки, kgdb на целевой)

3) JTAG- based debuggers.

Цели разработки JTAG

- Тестирование связей между интегральными схемами, после того, как они были смонтированы на печатной плате или другой основе;
- Наблюдение за работой компонент без вмешательства в их нормальную работу, или непосредственное управление одним или более компонентом;
- Обеспечение стандартизованного доступа к произвольным средствам самотестирования, встраиваемым в БИС;

История развития JTAG

- Joint Test Action Group (JTAG) связан с со стандартом IEEE 1149.1 который называется *Standard Test Access Port and Boundary-Scan Architecture*
- Предложен в 1990 как механизм для тестирования цифровых схем
- В 1994, добавлено описания boundary scan description language (BSDL).
- JTAG в настоящее время используется для тестирования компонентов интегральных схем, а также для отладки встраиваемых приложений

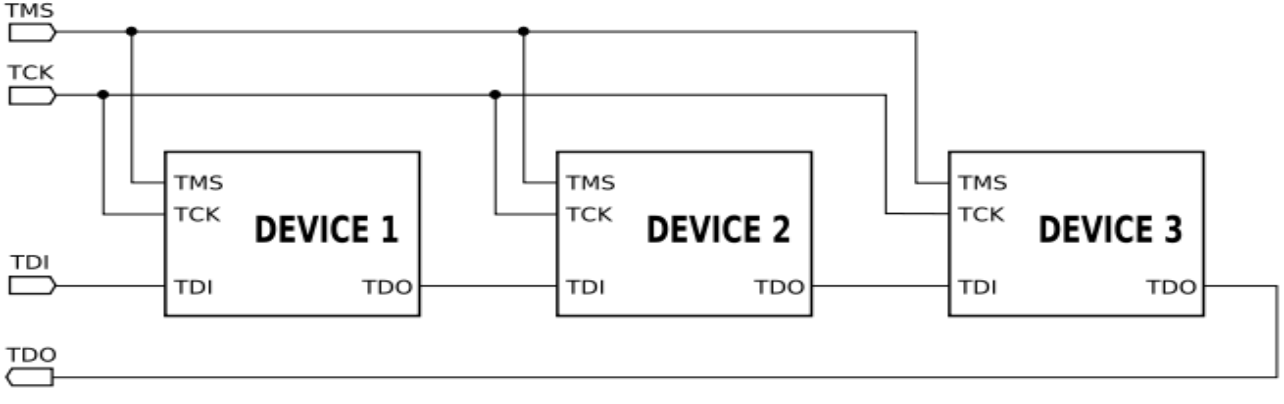
Граничное сканирование

Метод проектирования интегральных контролепригодных схем, в соответствии с которым в каждый чип вводятся сдвигающие регистры, состоящие из ячеек по одной на каждый внешний вывод микросхемы. Благодаря ячейкам, можно при проверке межсоединений печатных плат или подложек многокристальных СБИС отключать внутрикристальные цепи, а при проверке внутренних схем подключать сканирующие регистры или генераторы тестовых наборов и схемы компрессии результатов.

Для целей проектирования схем с граничным сканированием разработаны специальный стандарт IEEE 1149.1 и языки BSDL и HSDL (Boundary and Hierarchical Scan Description Languages), являющиеся подмножеством VHDL

Электрические характеристики JTAG

- JTAG это специализированный четырех/пяти проводной интерфейс
- Определены следующие линии:
  - TDI (Test Data In)
  - TDO (Test Data Out)
  - TCK (Test Clock)
  - TMS (Test Mode Select)
  - TRST (Test Reset) - optional



Пример схемы с поддержкой JTAG

- В зависимости от состояния автомата TAP в канал может быть включен либо регистр данных либо регистр команды.
- Регистр команды в JTAG контроллере всегда один.
- Регистров данных в JTAG контроллере может быть сколько угодно. Какой именно выбран для подключения определяется загруженной командой.
- Стандарт JTAG требует наличия в контроллере одноразрядного регистра данных, называемого BYPASS.

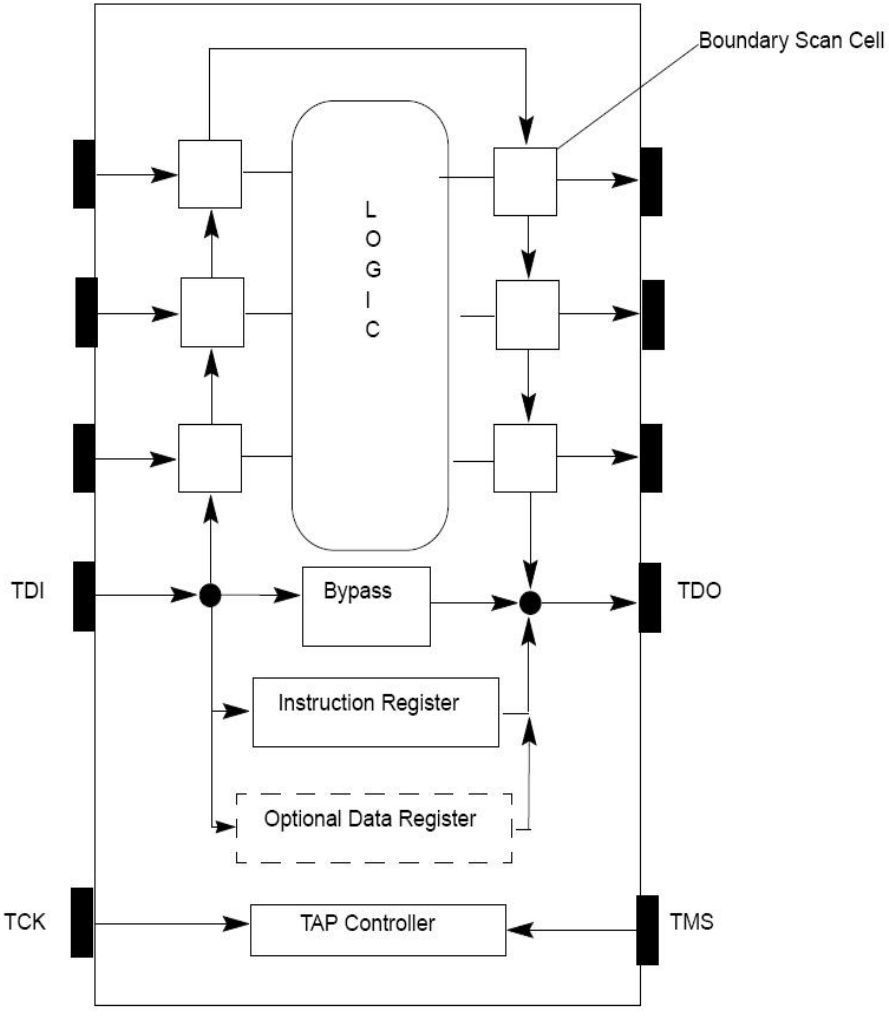
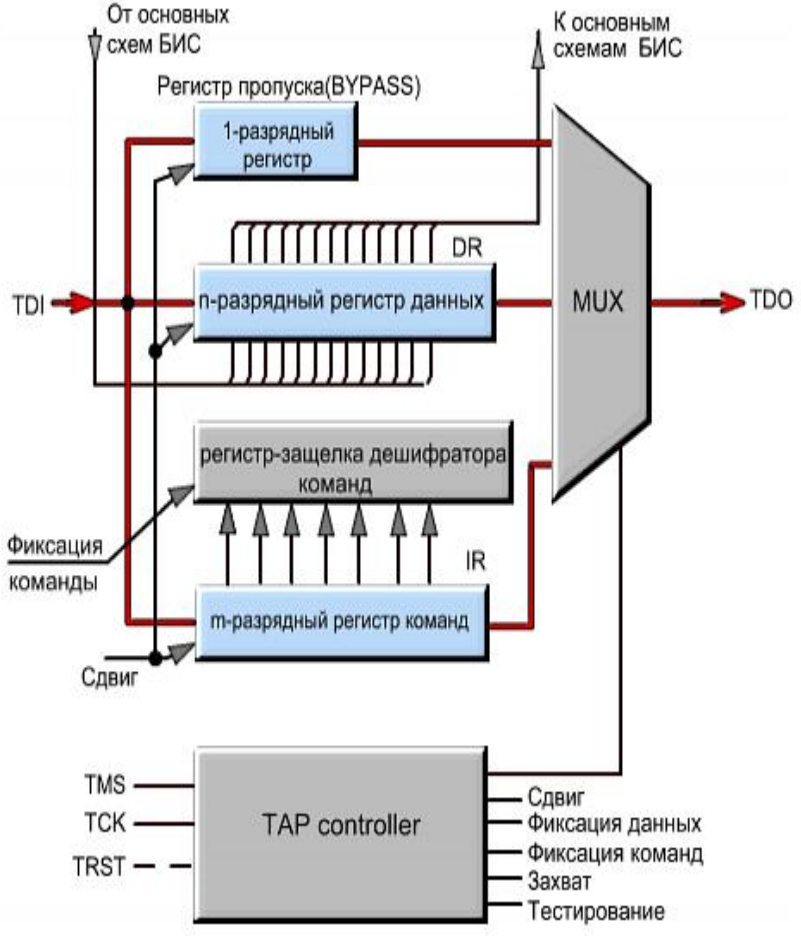


Схема управления JTAG интерфейсом

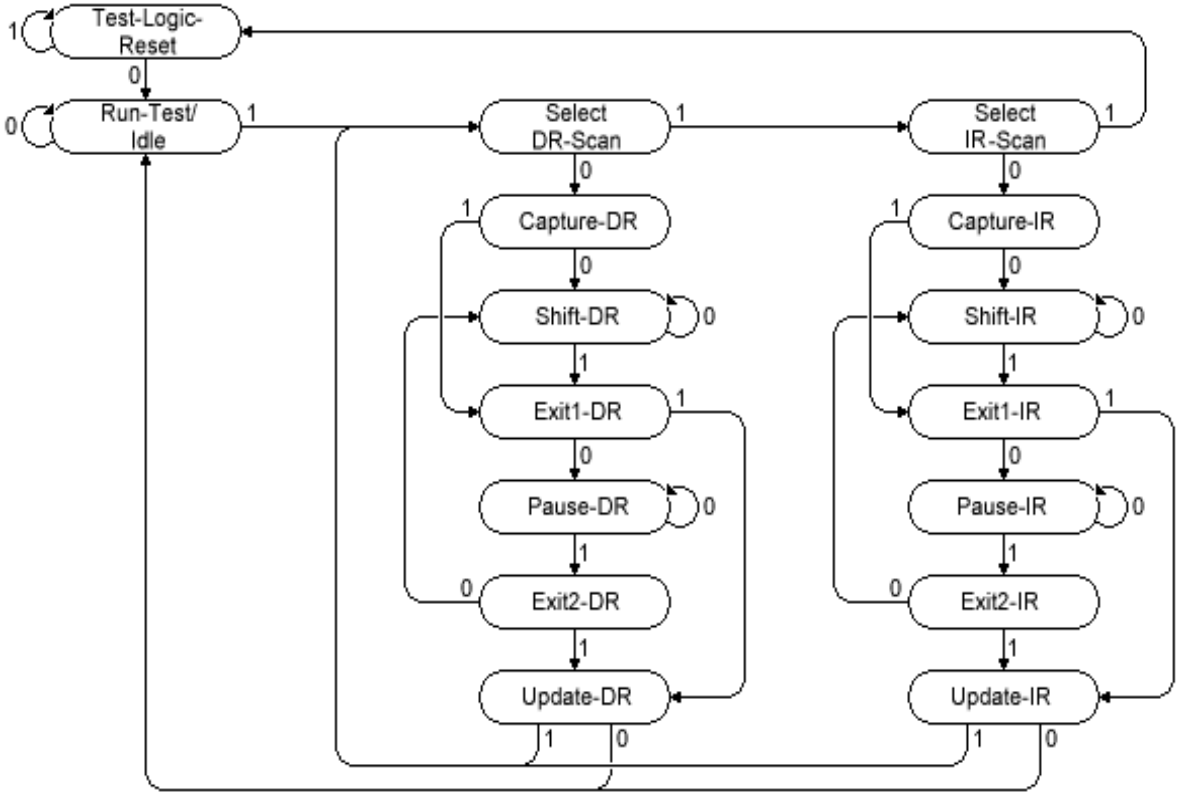
В состав схемы входят:

- Три сдвиговых регистра (регистр команд (IR), регистр пропуска (Bypass) и регистр данных (DR);
- Выходной мультиплексор (MUX);
- Контроллер управления (TAP Controller).
- Основным регистром является регистр данных, он служит источником и приемником данных при выполнении в JTAG цепочках любых команд. С точки зрения устройства управления, регистр данных является одним из трех сдвигающих регистров, включаемых между контактом для подачи входной информации (контакт TDI) и контактом для получения выходной информации (контакт TDO).



Конечный автомат для контролера TAP

Все чипы поддерживающие JTAG включают специальный TAP (test access port) контроллер, конечный автомат которого имеет 16 состояний



Команды JTAG

- Обязательные команды:
  - **BYPASS** 111...1 "all ones" command,

    регистр команд TAP-а заполняется единицами
  - **EXTEST** 000 ..0 "all zeros" command

    подключает "boundary scan" регистр между TDI и TDO
  - **SAMPLE** 000...1 "one in last bit" command

    присоединяется 1-bit "bypass" регистр между TDI и TDO

    TAP функционирует как прозрачный 1-битный регистр сдвига

Команды JTAG (продолжение)

- Стандартные опциональные команды:
  - **INTEST**

  перевести IC в режим internal boundary-test и выбрать boundary-scan регистр, который будет подключен между TDI and TDO

  - **CLAMP**

  устанавливает выходы IC в логические состояния определяемые содержимым boundary-scan регистров и выбирает bypass register для присоединения между TDI и TDO

  - **HIGHZ**

  устанавливает все выходы (включая two-state и three-state types) IC в состояние high-impedance и выбирает bypass register для соединения между TDI и TDO

  - **USERCODE**

  позволяет функциональный режим и выбирает регистр устройства для соединения между TDI и TDO

Команды JTAG (советы по использованию)

- BYPASS можно использовать для подсчета числа устройств в цепочке JTAG!
  - Если каждое JTAG IC задерживает TDI-TDO цепочку на один такт, мы можем послать некоторые данные и проверить какова будет задержка их появления на выходе. Это даст нам число IC в цепочке.

Boundary scan description language

* Boundary Scan Description Language (BSDL) является подмножеством VHDL и используется для спецификации того, как JTAG (IEEE 1149.1) реализован в конкретном устройстве

* Для того, чтобы устройство было JTAG compliant, для него должен быть соответствующий BSDL файл.

* Многие из имеющихся на рынке инструментальные средства IEEE Std 1149.1 используют BSDL как входной формат. Эти инструменты обладают различными возможности для работы с продуктами, поддерживающими IEEE Std 1149.1:

board interconnect automatic test-pattern generation (ATPG)

automatic test equipment (ATE).

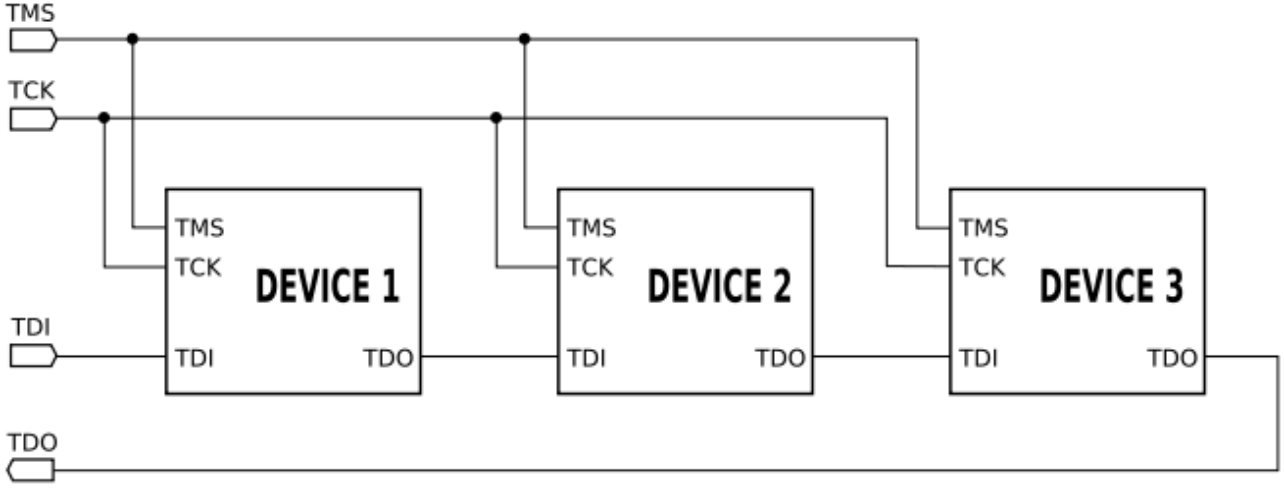
Формат BSDL файла

BSDL файл содержит следующие элементы:

- Entity Description
- Generic Parameter
- Port Description
- Use Statements
- Pin Mapping(s)
- Scan Port Identification
- Instruction Register Description
- Register Access Description
- Boundary Register Description

Daisy Chain

Несколько чипов на печатной плате могут быть связаны TDO предыдущей на TDI последующего, образуя “daisy chain".



JTAG позволяет нам выбрать с каким из чипов в “daisy chain” мы будем работать.

Daisy Chain (пример конфигурационного файла для BDI)

Нам необходимо указать для BDI2000/3000 сколько устройств (чипов) присоединено до нашего CPU и после, а также общую длину регистров команд этих групп устройств.

```
SCANPRED    2 24        ;Two JTAG devices connected before CPU: 8
                  ;(SYSTEM ACE) + 16 (XCF32P):
SCANSUCC    1 14        ;One only JTAG device is  connected after CPU: 6
                   ;(FPGA) + 8 (XC95144xl)
```

Типичные сценарии использование JTAG

- Тестирование внешних соединений чипа
- Программирование Flash
- Пошаговая отладка (на уровне инструкций)
- Ручное тестирование IO портов процессора

Тест внешних соединений (Chip external connection test) – традиционный подход

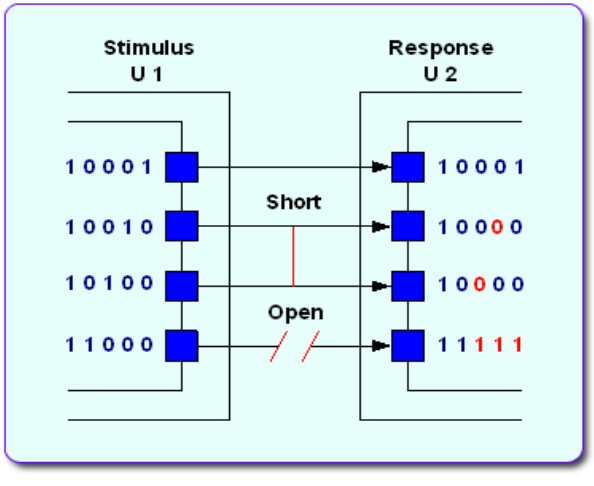
- **Традиционный подход**: используются логические анализаторы для отладки взаимодействия между процессором и внешними чипами
- **Проблемы**:
  - Дорогое оборудование ($10,000 и более)
  - Плата должна быть оборудования тестовыми точками
  - Для проведения тестирования программисты нуждаются в помощи со стороны инженеров электронщиков

Тест внешних соединений (Chip external connection test) с использованием JTAG

- Предположим, что в схеме две ошибки:
  - замыкание между линиями 2 и 3, а также
  - разрыв линии 4
- Предположим, что замыкание проявляется как wired-AND и разрыв как постоянное наличие логической 1.

Для выявления и изоляции обоих дефектов тестирующий сдвигает битовую маску приведенную на рис.2 в U1 регистре boundary-scan и подает эту маску на входы U2. Входные значения U2 boundary-scan регистра сдвигаются на выход и сравниваются с ожидаемыми результатами.

В этом случае результаты (помеченные красным) на линиях 2 ,3 и 4 будут отличатся от ожидаемых. Таким образом тестирующий определит наличие ошибок в линиях 2, 3, and 4.



Тест внешних соединений (Chip external connection test) с использованием JTAG(continued)

www.opencollector.org

- Предлагает большое количество программных проектов нацеленных на решение задач связанных с тестированием внешних соединений

Программирование Flash

- Алгоритм программирования Flash управляет выходами нашего чипа используя JTAG EXTTEST режим
- Логическое состояние на контактах устанавливается в 0,1 или high impedance в соответствии с диаграммой сигналов указанной в дата шите на FLASH chip.

Программирование Flash (пример кода из OpenOCD)

```
static void
pxa250_bus_write( bus_t *bus, uint32_t adr, uint32_t data )
{
	part_t *p = PART;
	chain_t *chain = CHAIN;

	if (adr >= 0x04000000)
		return;

	part_set_signal( p, nCS[0], 1, 0 );
	part_set_signal( p, DQM[0], 1, 0 );
	part_set_signal( p, DQM[1], 1, 0 );
	part_set_signal( p, DQM[2], 1, 0 );
```

Программирование Flash (пример кода из OpenOCD)

```
    part_set_signal( p, DQM[3], 1, 0 );
    part_set_signal( p, RDnWR, 1, 0 );
    part_set_signal( p, nWE, 1, 1 );
    part_set_signal( p, nOE, 1, 1 );
    part_set_signal( p, nSDCAS, 1, 0 );
    setup_address( bus, adr );
    setup_data( bus, adr, data );
    chain_shift_data_registers( chain, 0 );
    part_set_signal( p, nWE, 1, 0 );
    chain_shift_data_registers( chain, 0 );
    part_set_signal( p, nWE, 1, 1 );
    chain_shift_data_registers( chain, 0 );
}
```

Пошаговая отладка

- Обычно является внутренней возможностью коммерческих JTAG инструментов

- OpenOCD

(http://openhardware.net/Embedded_ARM/OpenOCD_JTAG/) реализация алгоритмов JTAG

Пошаговая отладка (продолжение)

Две возможные реализации отладочного программного обеспечения JTAG

- **Первый случай:** программа взаимодействует с JTAG устройством в терминах состояний конечного автомата TAP контроллера; битовые векторы сдвигаются в и из регистров TAP контроллера.
  - Физическое соединение с TAP контроллером осуществляется в программном обеспечении в режиме bit-banging.
  - Оптимизированный доступ к TAP контроллеру кабель ускоряет JTAG операции на физическо

Пример:

Olimex-USB-OCD + OpenOCD for ARM

Пошаговая отладка (продолжение)

- **Второй вариант:**
  - Программа взаимодействует с JTAG устройством в терминах команд отладчика, таких как "next instruction", "step-in", "step-over", "show registers"

Пример:

Abatron BDI 2000

Пример оптимизированного доступа к TAP контроллеру (JJTAG API)

- https://jjtag.dev.java.net/source/browse/jjtag/
- Использование логического представления JTAG на этом уровне позволяет разработчику JTAG устройства принимать решение использовать ли bit-banging или HW-optimized доступ к JTAG FSM на физическом уровне прозрачно для приложения.
- API разработано таким образом, что представляется как реализация "HW-optimized access" в то время как внутренняя реализация может быть любой.
- Код испльзующий APIs данного типа может работать на разных платформах (ОС); необходим только драйвер для низкоуровневого взаимодейтствия
- В простейшем случае это может быть bit-banging с кабелем параллельного порта. Пользователи PC с Windows , Linux, или “классические” EDA пользователи (обычно использующие Solaris) могут использовать его одинаково просто!

Следующий код иллюстрирует использование API:

Пример кода

```
public class ARMICE {
  // Instruction register size
  public static final int IR_SIZE = 4;

  // Instruction register values
  public static final String
    EXTEST = "0000", // External test
    SCAN_N = "0010", // Select scan chain
    SAMPLE_PRELOAD = "0011",
    RESTART = "0100", // Restart core
    CLAMP = "0101", // Clamp pins
    HIGHZ = "0111", // HiZ pins
    CLAMPZ = "1001", // Clamp, HiZ pins
    INTEST = "1100", // Internal test
    IDCODE = "1110", // Read ID code
    BYPASS = "1111"; // Bypass core
```

Пример кода (продолжение)

```
// Scan chains
  public static final String
    SCAN0 = "0000", // Macrocell scan test
    SCAN1 = "0001", // Debug
    SCAN2 = "0010", // EmbeddedICE-RT registers
    SCAN3 = "0011"; // External boundary-scan

  /** JTAG controller */
  JTAGController controller;

  /** Constructor - we provide an
    implementation-independent controller
  */
  public ARMICE(JTAGController c) {
    controller = c;
  }
```

Пример кода (продолжение)

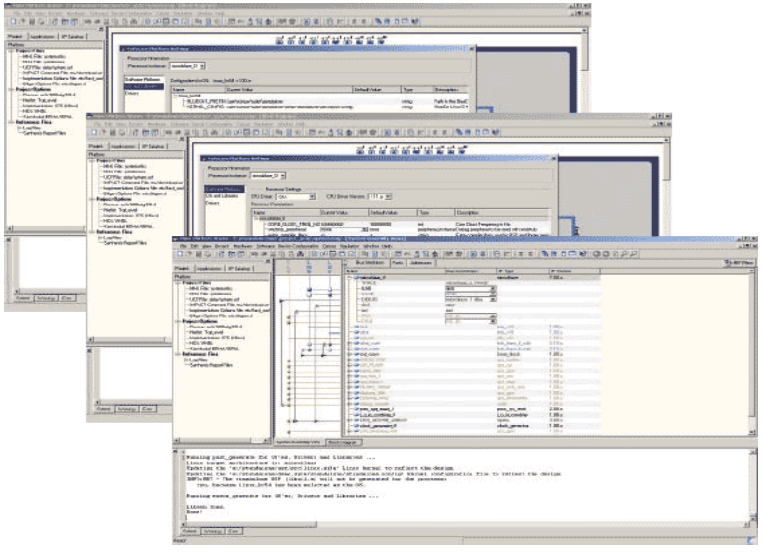
```
/* Some helper functions */
  String shift(byte ir, String dr) {
    controller.jump(ir);
    return controller.tdi(dr);
  }
  String shiftIR(String b) {
    return shift(SHIFT_IR, b);
  }
  String shiftDR(String b) {
    return shift(SHIFT_DR, b);
  }
  void selectScan(String s) {
    shiftIR(SCAN_N);
    shiftDR(s);
    shiftIR(INTEST);
    controller.jump(RUN_TEST_IDLE);
  }
```

Пример кода (продолжение)

```
String idcode() {
    controller.jump(SHIFT_IR);
    controller.tdi(IDCODE);
    controller.jump(SHIFT_DR);
    return controller.tdi(new Bits(0, 32)).toString();
  }
 public static void main(String[] args) throws Exception {
    ARMICE ai = new ARMICE(new JTAGWiggler(0x378));
    // reset the controller, set idle signal values
    ai.controller.initialize();
    // soft-reset the TAP, just in case
    ai.controller.reset();
    String c = ai.idcode(),
      ch = Integer.toHexString((int)Long.parseLong(c, 2));
    System.out.println("IDCODE: " + c + ", " + ch);
    System.out.println( "Controller is in state: " + description[ai.controller.state]);
  }
}
```

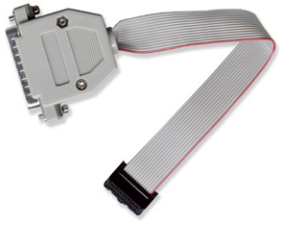
Chip-specific usages

- В связи с распространением заказной логики основанной на FPGA и ASIC , возрастает роль инструментов JTAG для отладки программного кода работающего с такой логикой.
- Разработчики конечных приложений стремяться использовать JTAG TAP controllers не только в части проекта связанной с основным процессором но и в связанной с заказной логикой. Xilinx предоставляет заготовки позволяющие встраивать регистры TAP контроллера в заказные микросхемы и свызывать их вdaisy chain.



Коммерческое оборудование

- Есть много различных вариантов различающихся по функциональности и стоимости.



Заключение

- JTAG очень прост на уровне аппаратного обеспечения так как он управляется простым конечным автоматом (“trivial FSM”)
- Ключевым местом JTAG являются высокоуровневые алгоритмы скрытые в коммерческих и опен-соурс инструментах, которые обеспечивают JTAG инструментам способность работать с FLASHes, пошаговой отладкой процессоров, работу с регистрами контроллеров, и другие механизмы, необходимые для запуска плат (board initial startup) (настройка SDRAM таймингов, GPIO мультиплексирование, начальная консоль UART, и т.д.)

Обзор opensource JTAG инструментов

- **Open Collector** http://opencollector.org/ база данных содержащая информацию о большом колличестве JTAG инструментов:
- **JTAG-O-MAT** (http://jtagomat.sourceforge.net) простой и гибко конфигурируемый инструмент коммадной строки для Win32 и Linux, в основном используется для первоначальной загрузки программного обеспечения на неинициализированные платы через JTAG интерфейс. В отличии от аналогичных проектов, сконцентрирован на автоматическмо выполнении JTAG последовательностей. Код специально упрощен для обеспечения переносимости и модифицируемости.
- **jtag-util 0.02** (http://recycle.lbl.gov/~ldoolitt/jtag.html) первая попытка разработки программного слоя для доступа с хоста (for host-side access) к JTAG Test Access Ports. Большинство разработчиков разрабатывают свои дублирующие друг друга решения, особеннос когда пытаются добиться переносимости между JTAG адаптерами, реализуют отладчики, или реализуют автодетект устройств в JTAG цепочке. Попытка комьюнити разработать унифицированный компонет для выполнения этих задач на основе общего API для mid-level доступа к JTAG устройствам. Это позволи разработчикам высокоуровневых приложений не изобретать колесо работая с низкоуровневыми деталямт такими как JTAG pins, etc.

Обзор opensource JTAG инструментов (продолжение)

- **JTAG Tools** (http://openwince.sourceforge.net/jtag/) программный пакет который позволяет работать с JTAG-совместимыми (IEEE 1149.1) аппаратными устройствами через JTAG адаптер. Пакет имеет отрытую модульную архитектуру и позволяет разрабатывать дополнительные модули расширения(как board testers, flash memory programmers, и т.д.).
- **EBS 0.1** (http://ebsp.sourceforge.net/) Назначение Experimental Boundary Scan project (EBSp) предоставить полностью открытый и гибкое программное решение для поддержки коммерчески доступных JTAG/IEEE 1149-1 boundary scan master (BSM) контроллеров. BSM контроллеры используются для обеспечения JTAG test bus control возможности на различных аппаратных платформах. EBSp предназначен для обеспечения адекватной программной поддержки, которая облегчит использования этих устройств под Linux на x86 архитектуре. В настоящее время, EBSp обеспечивает Linux драйвер устройства для Texas Instruments SN74ACT8990 BSM. Кроме того доступны высокоуровневые программные средства такие как аппаратно независимый парсер для Serial Vector Format (SVF) с plug-in модулей и графическим интерфейсом пользователя.

Opensource JTAG Tools Overview (continued)

- **ianjtag 1.2** (http://www.inaccessnetworks.com/projects/ianjtag/) набор инструментов и примеры кода для использования JTAG интерфесainterface для выполнения аппаратных тестов и программирования Flash Memory Devices подключенных к процессорной шине. Наиболее применим во встраиваемых системах для выполнения первоначального тестирования и инициализации загрузчиков прототипов систем. ianjtag tools выполняются на host системе system (e.g., PC с Linux) и взаимодействует с целевой системой (e.g., the embedded system's CPU board) через простой 5-line аппаратный интерфейс. В текущей реализации используется параллельный порт целевой системы в качестве аппаратного интерфейса, кроме того возможны другие аппаратные реализации.
- **MITOUJTAG 0.0.2** (http://www.tokudenkairo.co.jp/jtag/) предназначена для полной программной поддержки JTAG для Linux. Позволяет выполнять boundary scan любой IC имеющей BSDL файл и package shape description файл. (Для некоторых стандартных решений такиех как DIP , PLCC , QFP and BGA, есть стандартные package files) Даже если у Вас нет дорогого асцилографа и логического анализатора вы можете увидеть любые состояния интегральной схемы. Может быть подключена через TCP/IP.

OpenOCD project

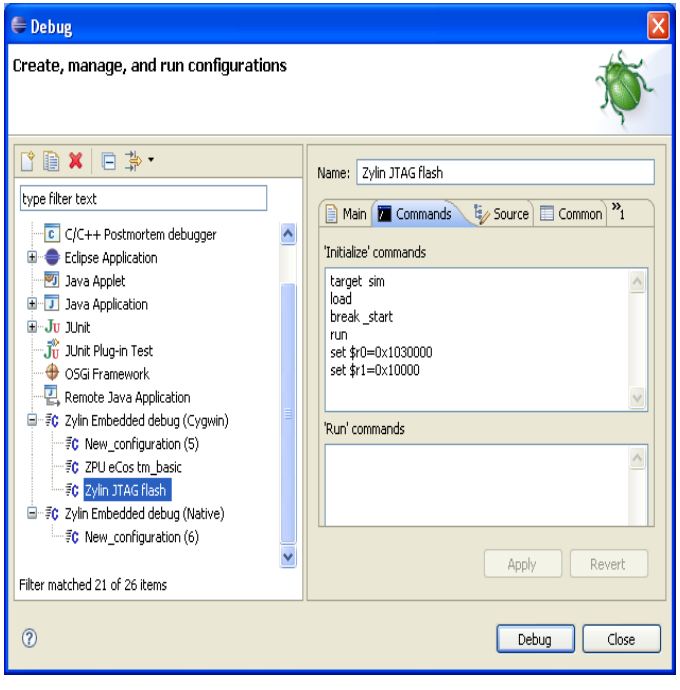
- The Open On-Chip Debugger (openocd) предназначене для отладки, in-system programming и boundary-scan тестирования для встраиваемых устройств.
- Openocd в настоящее время поддерживает Wiggler (clones), FTDI FT2232-based JTAG interfaces, the Amontec JTAG Accelerator, and the Gateworks GW1602. It allows ARM7 (ARM7TDMI and ARM720t), ARM9 (ARM920t, ARM922t, ARM926ej-s, ARM966e-s), XScale (PXA25x, IXP42x) and Cortex-M3 (Luminary Stellaris LM3 and ST STM32) based cores to be debugged.
- Запись Flash поддерживается для внешних CFI compatible flashes (Intel and AMD/Spansion command set) и некоторых внутренних flashes (LPC2000, AT91SAM7, STR7x, STR9x, LM3 and STM32x). Включая предварительную поддержку для LPC3180's NAND flash контроллера.

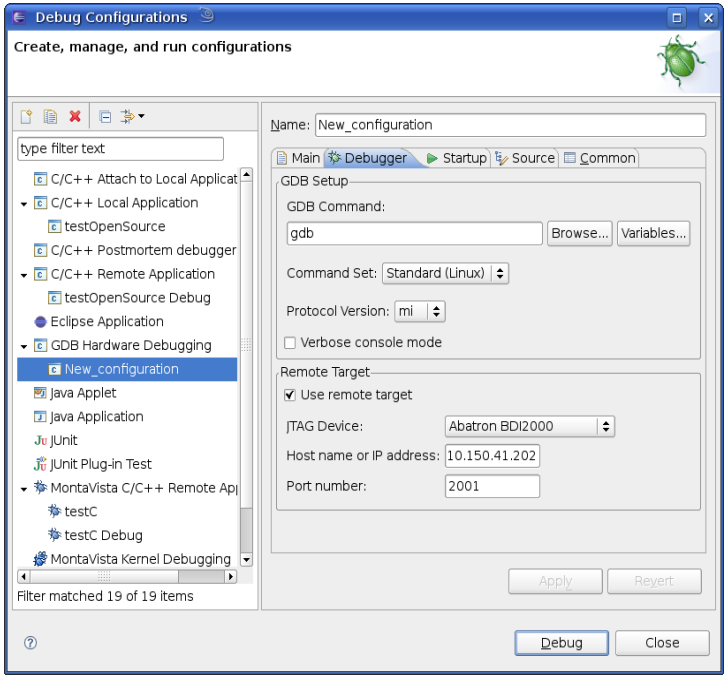
OpenOCD project (continued)

- OpenOCD выполняется как демон, ожидает соединения от клиентов (Telnet or GDB).
- Читает конфигурацию из файла openocd.cfg расположенного в текущей дирректории или из дирректории указанной при помощи опции '-f <configfile>'.

Opensource Eclipse Front-ends

- Zylin Embedded CDT http://www.zylin.com/embeddedcdt.html
- CDT Hardware Debugging plugin http://www.eclipse.org/cdt/

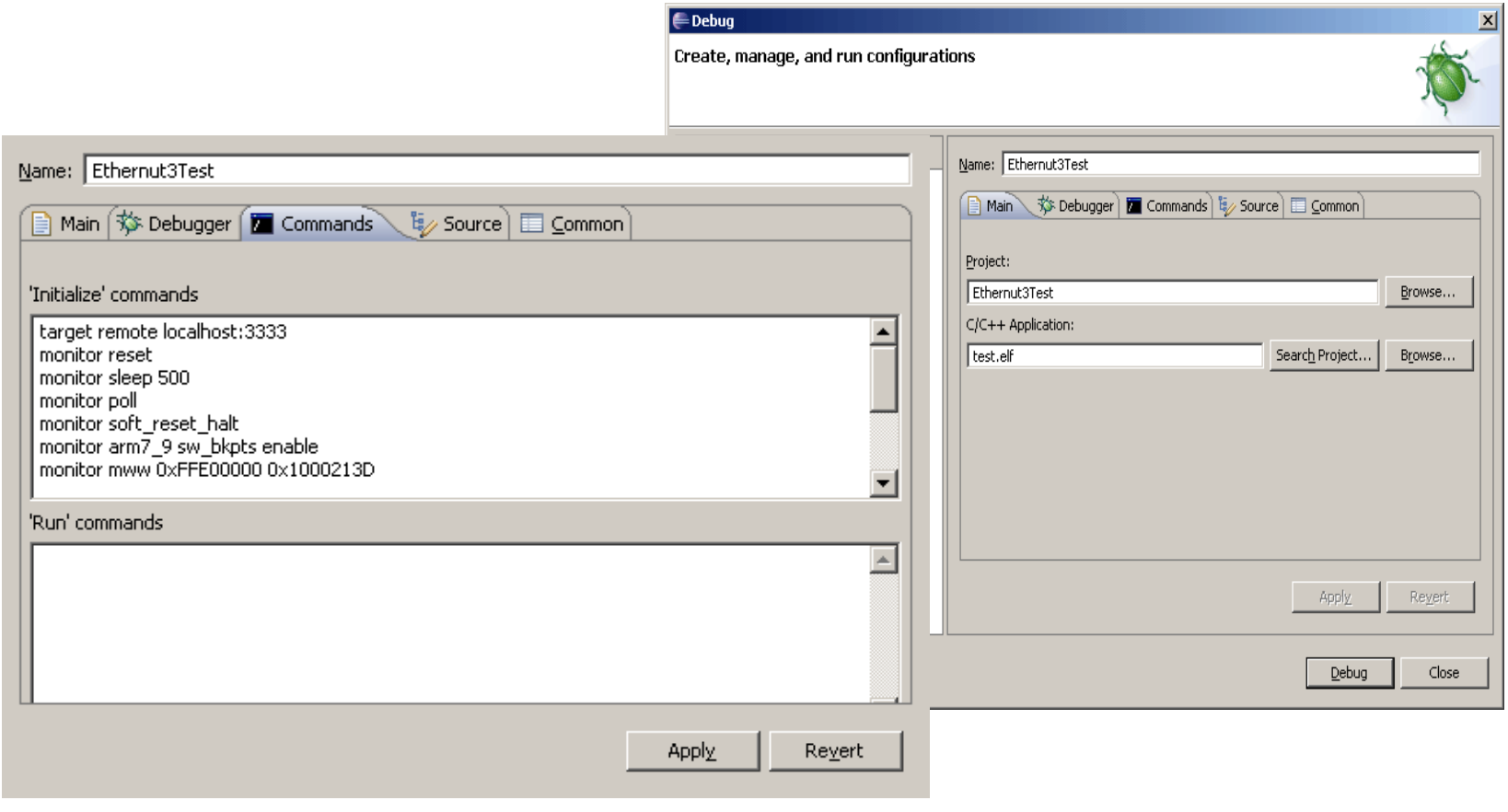




Zylin Embedded CDT

- Appeared first, works with the official Eclipse CDT 4.x or higher
- Actually modifies CDT
- Supports most of JTAG/hardware debuggers: you can issue "monitor XXX" commands to send commands to the JTAG/hardware debuggers in the GDB startup/run scripts to perform such functions as flash programming and resetting the target.
- No fancy flash programming GUIs
- Can be used with any CPU that GDB supports

Zylin Embedded CDT (continued)



CDT Hardware Debugging Plugin

- Created to avoid the need for forking the CDT
- Closely replicates what Zylin has done
- Included in CDT 5.0

Initialization Commands

☑ Reset and Delay (seconds): 3

☑ Halt

```
monitor debug_level 2
monitor mt_internal_rc
load
compare sections
```

Name: testC (1)

Main | Debugger | Startup | Source | Common

GDB Setup

GDB Command:

gdb | Browse... | Variables...

Command Set: Standard (Linux)

Protocol Version: mi

☐ Verbose console mode

Remote Target

☑ Use remote target

JTAG Device: Generic

Host name or IP address: localhost

Port number: 3333

Apply | Revert

- Нет одного opensource или коммерческого решения для всех вариантов использования l JTAG и всех JTAG устройств.

Поставщики Vendors Jtag переходников/устройств обычно предлагают какие либо программные инструменты и GUI для своих продуктов за отдельную плату.

- Некоторые такие как Abatron (BDIx000 devices vendor) предоставляют терминальный пользовательский интерфейс и поддержку для GDB совместимых инструментов в результате можно использовать такие оболочки как Eclipse, OpenOCD, DDD с их устройствами.
- Разработчики могут также купить или собрать дешевый LPT JTAG кабель, но это решение страдает из за очень низкой производительности.
- Для отладик программ с использование JTAG, opensource предлагает множество решений совместимых с коммерческими JTAG кабелями/устройствами.

Обзор коммерческих интегрированных JTAG решений

- Linux Scope (Ultimate Solutions)

  http://www.ultsol.com/mfgs_comp_linuxscope.htm
- PowerView (Lauterbach)

  http://www.lauterbach.com/frames.html?powerview.html
- WindRiver Workbench (WindRiver)

  http://www.windriver.com/products/OCD/workbench_OCD/
- TimeStorm IDE (TimeSys)

  https://timesys.com/products/tools/timestorm
- Arriba Embedded Edition (Viosoft)

  http://www.viosoft.com/index.php?option=content&task=section&id=3&Itemid=30

Вопросы?

Благодарности

Мы благодарим

- Dina Kommar (MontaVista)

  for testing Olimex Arm-USB-OCD with openOCD and Eclipse CDT Hardware Debugging plug-in

- Laurette Wharton (MontaVista)

  for language and style of this presentation