JTAG,OpenOCD,BDM and GDB.

Dheeraj Chidambaranathan (1205016081)
Imtiyaz Hussain (1204032877)

Agenda

- JTAG and its working.
- OpenOCD and its working.
- Interaction between JTAG and OpenOCD.
- GDB and its interaction with OpenOCD.
- What is BDM and how it works.
- Proposed project.

JTAG.

- JTAG= Joint Test Action Group.

- IEEE 1149.1 standard titled Standard Test Access Port and Boundary Scan Architecture.

- Started in 1990 as a digital test mechanism; became a standard in 1994.

- Primarily used for accessing blocks of ICs but is also used as a mechanism for debugging embedded systems.

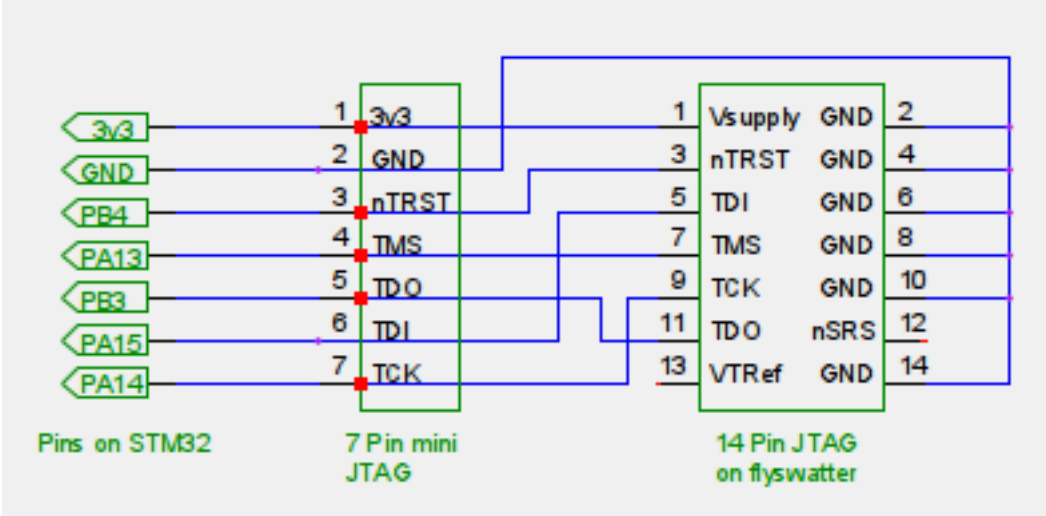
- Uses a controller called TAP(Test Access Port).

Why use JTAG?

- Biggest problem with ICE is the increasing pin count of ICs.
- Each pin is required to be wired to the ICE and as frequency increased, each wire behaved like an antenna!
- BGA processors didn't have a center point to be accessed because of soldering!
- JTAG using Boundary scan to scan numerous devices in a chain.
- Allows multiple devices as it's a simple serial protocol.
- Max Speed of JTAG is 100Mhz… hence a simple ribbon cable is sufficient to connect.
- Can connect to many ports namely, USB, Parallel Port, Serial Port, Ethernet, etc…

Characteristics of JTAG.

- 10/14/16/20 pin Interface.
- Main pins are:
  - TDI (Test Data In)
  - TDO (Test Data Out)
  - TCK (Test Clock)
  - TMS (Test Mode Select)
  - TRST (Test Reset) – optional.



Boundary Scan.

- JTAG provides means to test interconnects between integrated circuits on a board without physical test probes.

- Adds a boundary-scan cell that includes a multiplexer and latches to each pin on device.

- Captured data is serially shifted out into cells.

- Hence large test points are not needed for complex circuits.

Basic IC architecture of IEEE 1149.1.

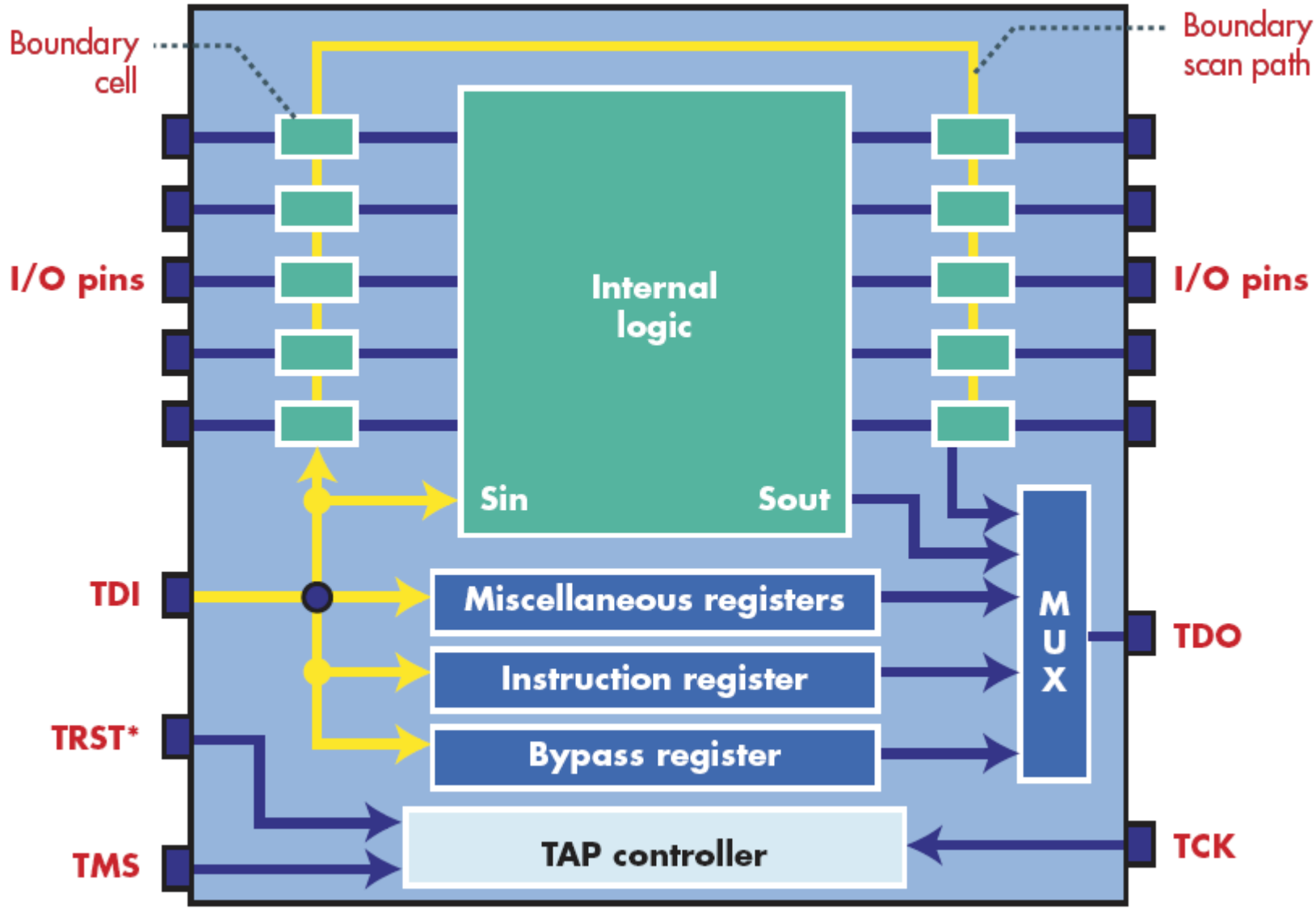


Figure 1

JTAG Pins and description.

- TCK- Synchronizes the internal state machine operations.
- TMS – Sampled at rising edge of TCK to determine next state.
- TDI- Data shifted into device's test or Programming logic.
- TDO- Data shifted out of the device's test or programming logic.Sampled at falling edge of TCK.
- TRST- can reset the TAP controller's state machine.

OpenOCD

- Open On-Chip Debugger
- Created by Dominic Rath at University of Applied Sciences Augsburg.
- Provides debugging, in-system programming and boundary-scan testing.
- Uses a debug adapter- in our case JTAG.

OpenOCD

- Supports-
  - •Dongles.
  - •GDB Debug.
  - •Flash Programming.

- User interaction is realized through -
  - a telnet CLI
  - a gdb remote protocol server
  - RPC connection to interface with OpenOCD's Jim Tcl engine.

Invoking OpenOCD

- Process configuration commands provided on the CLI.

```
$ openocd -f interface/ADAPTER.cfg -f board/MYBOARD.cfg\
```

- Verifies the JTAG scan chain. On success, OpenOCD starts running as daemon.

- Waits for connection from clients - Telnet, GDB and others.

- Alternatively, Commands can be used to terminate the configuration stage early, perform work (such as updating some flash memory), and then shut it down without acting as a daemon.

Interface Configuration

- Tells OpenOCD what type of debug adapter is used.
  - interface *name* - Use interface driver name to connect to the target
- Others -
  - interface_list - List debug adapter drivers
  - interface transports *transport_name* - Specify the transports supported.
  - adapter_name - Returns debug adapter driver name

Configuration Files.

```
#
# TinCanTools Flyswatter
#
# http://www.tincantools.com/product.php?productid=16134
#

interface ft2232
ft2232_device_desc "Flyswatter"
ft2232_layout "flyswatter"
ft2232_vid_pid 0x0403 0x6010
```

```
source [find target/samsung_s3c2410.cfg]

$_TARGETNAME configure -event reset-init {
        # Reset Script for the TinCanTools S3C2410 Based Hammer Module
        # http://www.tincantools.com
        #
        # Setup primary clocks and initialize the SDRAM
        mww 0x53000000 0x00000000
        mww 0x4a000008 0xffffffff
        mww 0x4a00000c 0x000007ff
        mww 0x4c000000 0x00ffffff
        mww 0x4c000014 0x00000003
        mww 0x4c000004 0x000a1031
        mww 0x48000000 0x11111122
        mww 0x48000004 0x00000700
        mww 0x48000008 0x00000700
        mww 0x4800000c 0x00000700
        mww 0x48000010 0x00000700
        mww 0x48000014 0x00000700
        mww 0x48000018 0x00000700
        mww 0x4800001c 0x00018005
        mww 0x48000020 0x00018005
        mww 0x48000024 0x009c0459
        mww 0x48000028 0x000000b2
        mww 0x4800002c 0x00000030
        mww 0x48000030 0x00000030
        flash probe 0
}


#flash configuration
#flash bank <name> <driver> <base> <size> <chip_width> <bus_width> <target> [driver_options ...]
set _FLASHNAME $_CHIPNAME.flash
flash bank $_FLASHNAME cfi 0x00000000 0x1000000 2 2 $_TARGETNAME
```

Transport Configuration

Depending on the version of OpenOCD and the debug adapter being used, several transports are available to communicate with debug targets.

- JTAG Transport
- SWD Transport - Serial Wire Debug
- SPI Transport - Serial Peripheral Interface

Daemon Stages

- Configuration Stage
  - Enters when OpenOCD server starts up.
  - Only *Configuration commands* are recognized.
  - Basic setup - TAPs, flash banks and interface.

- Run Stage
  - Verifies the JTAG scan chains.
  - Number of commands become available - init, mww, probe, jtag_init

OpenOCD commands

- Cannot enter commands directly to OpenOCD
- $ telnet localhost 4444
- Commands -
  - > reset - Resets the target board.
  - > halt - Sends halt request to the board.
  - > resume - Resume from a halt.
  - > reg - Lists target board's registers.
  - > reg [*entry*][*value*] - Sets the value of a register.
- For full list of commands, enter help in the telnet window.

GDB basics.

- GDB- GNU DeBugger.
- Written by Richard Stallman in 1986.
- GDB 7.5 latest version.

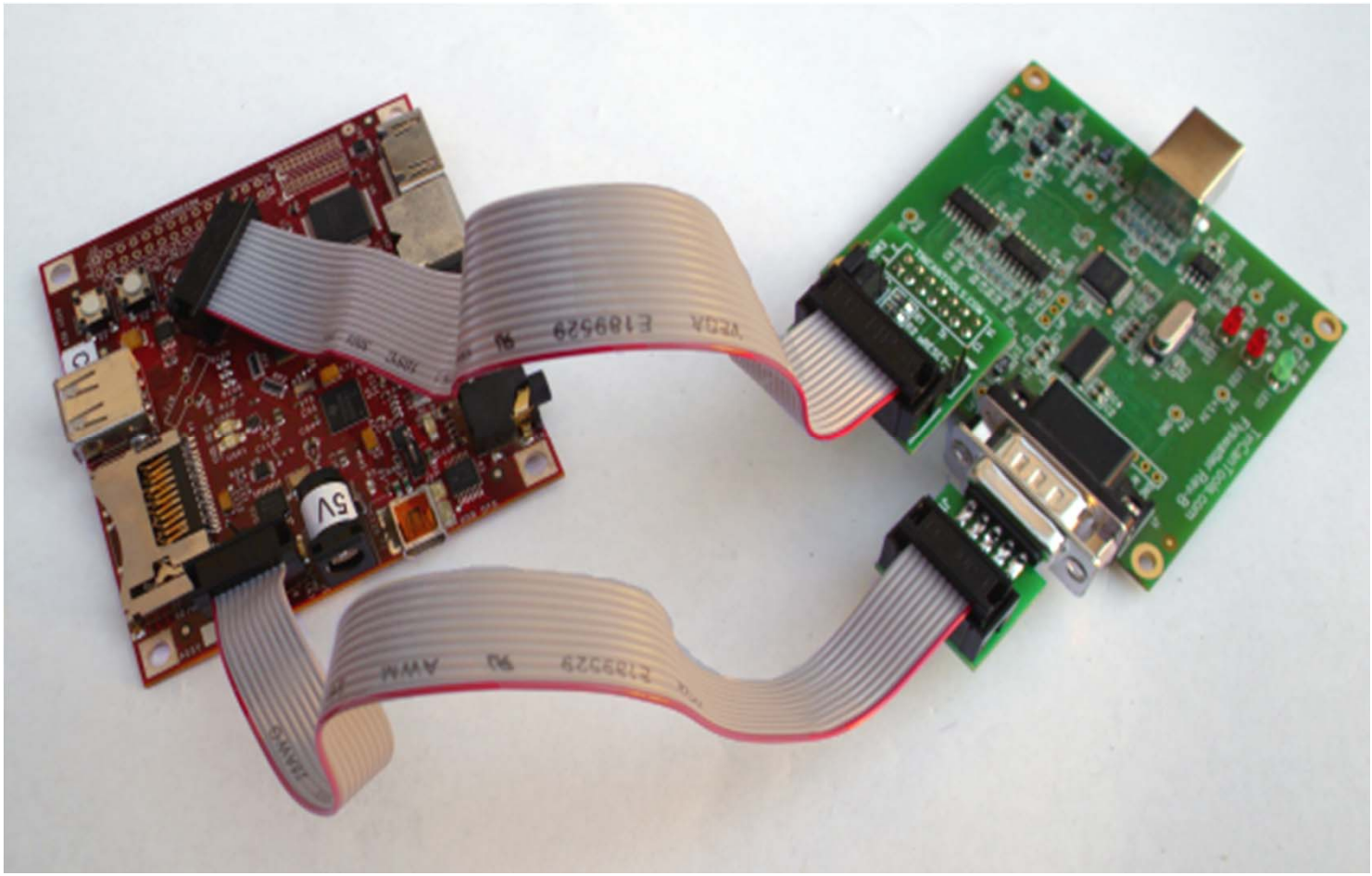
Important Commands:

- r: Run the program till next breakpoint.
- b: Assign breakpoint.
- n: Step to next line.
- bt: Backtrace to previous breakpoints.
- c: Continue.
- display *var*: Display the variable at each breakpoint.
- print *var*: Print the value of the Variable at that line.
- info b: Show existing breakpoints.

GDB Support in OpenOCD

- CodeSourcery ARM Toolchain provides GDB build to support ARM based embedded devices.
- Starting GDB
  - arm-non-eabi-gdb
- Connecting to OpenOCD
  - (gdb) target remote localhost:3333
  - (gdb) monitor reset init
  - (gdb) monitor halt
- *monitor* - tells GDB to send command to OpenOCD

Flyswatter and Beagleboard



BDM

- Background Debug Mode.
- 1 wire interface.
- Electronic interface.
- Provides in-circuit debugging functionality in microcontrollers.
- Requires single wire and specialized electronics in the system being debugged.
- Appears in many Freescale semiconductor products.

BDM continued

- Signals are initiated by host processor to communicate data to and from the target. Host *negates* the transmission line and then either -
  - Asserts the line sooner, to output a 1.
  - Asserts the later, to output a -.
  - Tri-states its output, allowing the target to drive the line.

- BDM commands - READ_BYTE, WRITE_BYTE, GO and more.

References

- OpenOCD Official Guide
- http://openocd.sourceforge.net/ - Nice Video
- http://en.wikipedia.org/wiki/Debugger#Hardware_support_for_debugging
- http://en.wikipedia.org/wiki/In-System_Programming
- http://eli.thegreenplace.net/2003/10/30/hardware-debugging-is-hard/
- http://www.tincantools.com/wiki/Flyswatter_How_To#Installing_OpenOCD
- http://www.tincantools.com/wiki/GDB_Debugger
- http://en.wikipedia.org/wiki/Background_Debug_Mode_interface
- http://www.tincantools.com/wiki/Running_OpenOCD_on_Linux_with_the_Beagleboard
- http://hackaday.com/2012/09/27/beginners-look-at-on-chip-debugging/
- http://elinux.org/BeagleBoardOpenOCD
- http://www.tincantools.com/wiki/Running_OpenOCD_on_Linux
- http://elinux.org/Debugging_The_Linux_Kernel_Using_Gdb
- http://www.thegeekstuff.com/2010/03/debug-c-program-using-gdb/