

# Blackfin debugging via gdb and JTAG

An elegant and robust solution to debug kernel software and hardware on Blackfin powered embedded systems

# Overview

1. The Blackfin series from Analog Devices Inc: A DSP/uC hybrid SoC
2. The JTAG standard
3. The ICEbear JTAG adapter, Gnu debugger and gdbproxy
4. Debugging features

# The JTAG interface

- Four pin serial protocol:
  - TCK: Clock
  - TMS: Mode Selection
  - TDI: Data input
  - TDO: Data output
- Optional: TRST reset pin
- Well established and standardized as IEEE 1149.1

# The Blackfin architecture

- Family of Hybrid DSPs, running from 400-600 MHz
- Excellent DMA engine and efficient cache
- Low power, various peripherals
- Well supported ports of uClinux, RTEMS, eCos

Typical applications:

- Mid power audio/video solutions
- Realtime controllers, communication

# Embedded complexity

Usage scenarios:

- Mikrocontroller/DSP: no operating system, standalone interrupt driven control software, little RAM
- System on chip: Operating system, SDRAM

Different levels of debugging:

- Kernel space, close to hardware
- User space, abstracted

# Potential Failures

- Production errors: weak balls/pins
  - Hardware misconfigured
  - Peripheral errors
  - Programming errors
- ▶ Debug as close to the hardware as possible

# Debugging via JTAG

Use:

- Non-intrusive Kernel and driver debugging:  
"Keep the probe (almost) invisible"
- Built in debugging/performance measuring features of the Blackfin core
- Hardware debugging/pin probing via Boundary Scan techniques

**In Circuit Emulation (ICE):** We are emulating the CPU behaviour via JTAG from outside

# GNU tools

## **Pro:**

- Free and OpenSource: "If you need to, improve the software yourself" (and share).
- Large and skilled community
- Powerful and often more robust than vendor tools

## **Con:**

- More difficult learning curve

# Classic remote debugging

- Front end: GDB (on user PC)
- Back end: gdbserver (running on embedded target) or serial debug interface

The gdbserver needs a OS to operate

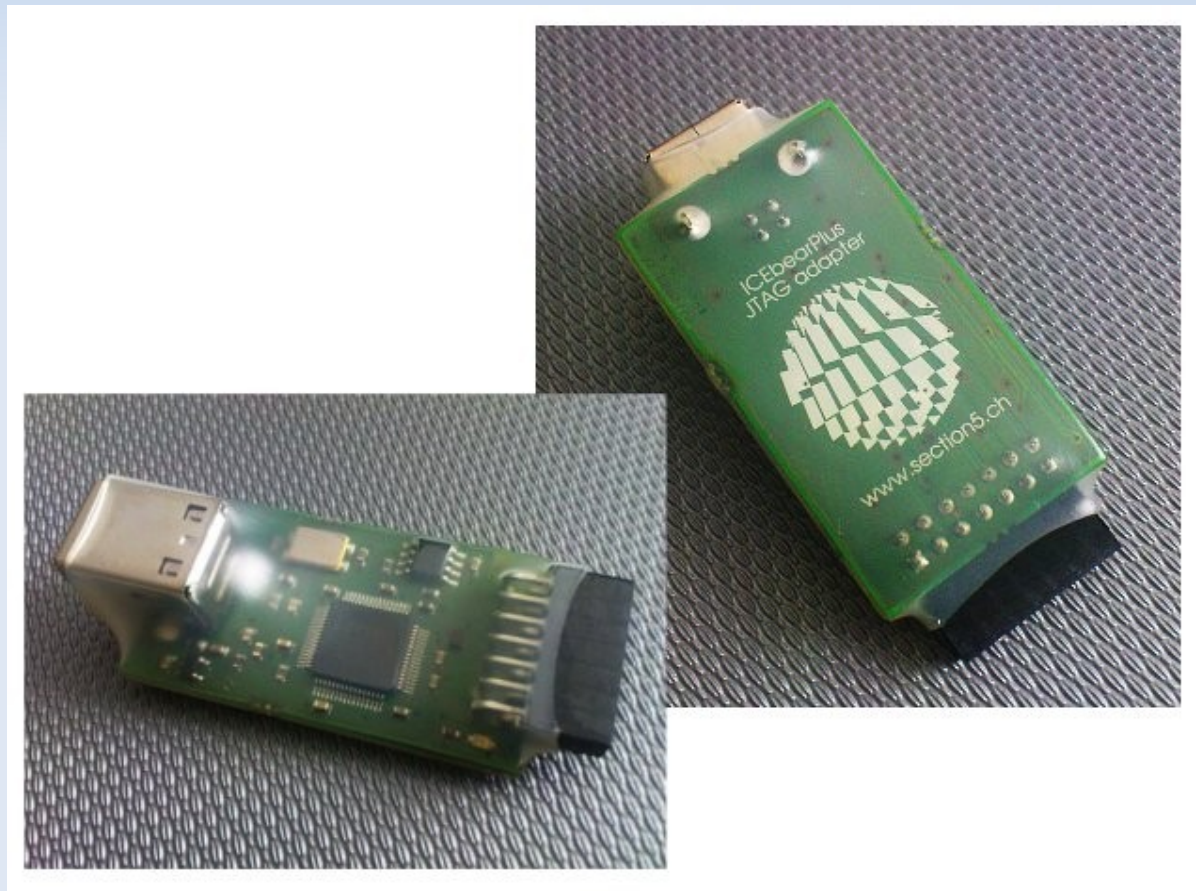
▶ Kernel debugging only possible via *intrusive* back end

# JTAG remote debugging

- Uses GDB as front end on user PC
  - Runs gdbproxy tool as debug agent on same PC (or another)
  - Gdbproxy drives JTAG library which drives the JTAG interface of the target
- ▶ non-intrusive debugging without OS or special debug port

# Real world

ICEbear JTAG adapter:



# Usage

- Plug in ICEbear to target header (normally 14 pin, ADI standardized) and USB port on PC
- Program flash using **BFloader**
- Debugging using **gdbproxy** and **GDB**

Features:

- USB 2.0 High speed
- Up to 30 Mhz JTAG clock
- Software for both Windows and Linux

# Debugging with Insight/GDB

The screenshot displays the Insight/GDB interface with several windows open:

- Terminal Window (strubi@samson: /home/strubi):** Shows GDB proxy logs, including messages like "Setting clock wait cycle to 1", "Detected 1 core(s)", "gdbproxy: waiting on TCP port 2000", "gdbproxy: connected", "Detected CPU: BF-533, rev: 3", "gdbproxy: debugger detached", "gdbproxy: will wait for a new connection", and "Got RX IRQ from [0]".
- Source Window (shell.c - Source Window):** Displays the source code for the `gettoken` function. Line 165, `return T_BAR;`, is highlighted in green. The code includes a `switch` statement for `state` with cases for `S_NEUTRAL`, `';`, `'&`, `'!`, `'<`, `'\n'`, and `T_NL`.
- Memory Window:** Shows a list of memory addresses and their values. A watch window is overlaid on this, displaying the following variables:
  - `*$IPEND = (unsigned long) 0x8001`
  - `seqstat = (int32_t) 0x2000`
  - `$p0 = (void *) 0xffb00d2c`
  - `$r0 = (int32_t) 124`
- Watch Window:** A dialog box titled "Watch" with an "Add Watch" button.
- Disassembly Window:** Shows assembly instructions for the `gettoken` function. Line 212, `r0 = 0x3 (x);`, is highlighted in green.
- Program Status:** The status bar at the bottom indicates "Program is running." with the current instruction address `ffa0070c` and instruction number `165`.

# Insight/GDB

- GDB under the hood
- Graphical interface extension with Tcl/Tk
- Source code and assembly debugging
- Register and variable display
- Powerful scripting language

# Debug facilities

- Breakpoints (Software/Hardware)
  - Single stepping
  - Program download
  - Automated debugging and system testing using scripts
- ▶ Almost all functionality you would be used to from debugging a local program. And more...

# BFloder

”The Blackfin Flash Loader”

- Programming of flash images onto the target
- Command line tool for batch or script driven programming
- Graphical interface with sector map display for interactive and bulk programming

Customized variants for multiple production lanes (parallel programming) exist

# Tool comparison

	<b>ICEbear toolchain</b>	<b>VisualDSP++</b>
<b>Target user</b>	Expert, Kernel developer	Beginner, evaluation
<b>Work Speed</b>	Medium (ICEbear classic) High (ICEbearPlus)	Slow to high, depending on ICE adapter
<b>Support</b>	Debugger, Flash- Programming GNU-Compiler must be installed separately	Full IDE Emulator not included No uClinux-Support
<b>Preferred OS</b>	Linux, Windows	Windows
<b>Costs</b>	~230 USD	Check Analog Devices website

# ICEbear variants

## ICEbearPlus

- Full debugging support and flash programming
- Support for (inhomogenous) multiprocessor environments
- Developer support

## ICEbear light

- Flash programming only via Bfloader ( of serial and parallel flashes)
- Supports manual production programming
- No developer support

# Links

- Blackfin uClinux resources:  
<http://blackfin.uclinux.org>
- ICEbear home page:  
<http://www.section5.ch/icebear>
- Official ADI website  
<http://www.analog.com>