gdbproxy: An open source GDB stub for Blackfin

Jie Zhang
jie.zhang@analog.com

HelloGCC '09, Beijing
November 7, 2009

The Original rproxy

- Developed by Aleksey of Quality Quorum around 1999 – 2000.
- Original released under GPL, but later changed to BSD License.
- Not maintained now.

The Original gdbproxy

- Developed by Steve Underwood for TI MSP430.
- Based on rproxy.
- It was used to embed a proprietary library from TI.
- Only generic code is in CVS, with a skeleton target.
- Not actively maintained now.
- fetproxy: an open source replacement.

The Original gdbproxy for Blackfin

- Developed by Martin Strubel of section5.
- Use a proprietary library for JTAG operation: libbfemu.
- Support IGLOO parallel port JTAG cable and ICEbear USB JTAG cable.

My Motivation

- Free as in Freedom.
- Our customers need a JTAG debugging solution: Fast, Cheap, Linux.
- VisualDSP++: Extremely expensive. Only available on Windows.
- IGLOO parallel port JTAG cable: Too slow. No parallel port at all.
- ICEbear USB JTAG cable: Too expensive (US$ 299). Not open source.
- I want to learn something about JTAG.
- We CAN do it.

An Overview of JTAG (1)

- Joint Test Action Group. IEEE 1149.1-2001.
- Formed in 1985 to develop a method to test populated circuit boards after manufacture.
- Now used for: Debugging. Storing Firmware. Boundary Scan Testing.

An Overview of JTAG (2)

JTAG pins:

- TDI (Test Data In)
- TDO (Test Data Out)
- TCK (Test Clock)
- TMS (Test Mode Select)
- TRST (Test Reset) optional
- Vendor pins

An Overview of JTAG (3)

- Instruction register
- Instructions: BYPASS, SAMPLE/PRELOAD, IDCODE (optional).
- Data registers: Boundary Scan Register, Bypass Register, Device Identification Register.
- Driven by a state machine.

The JTAG Chain

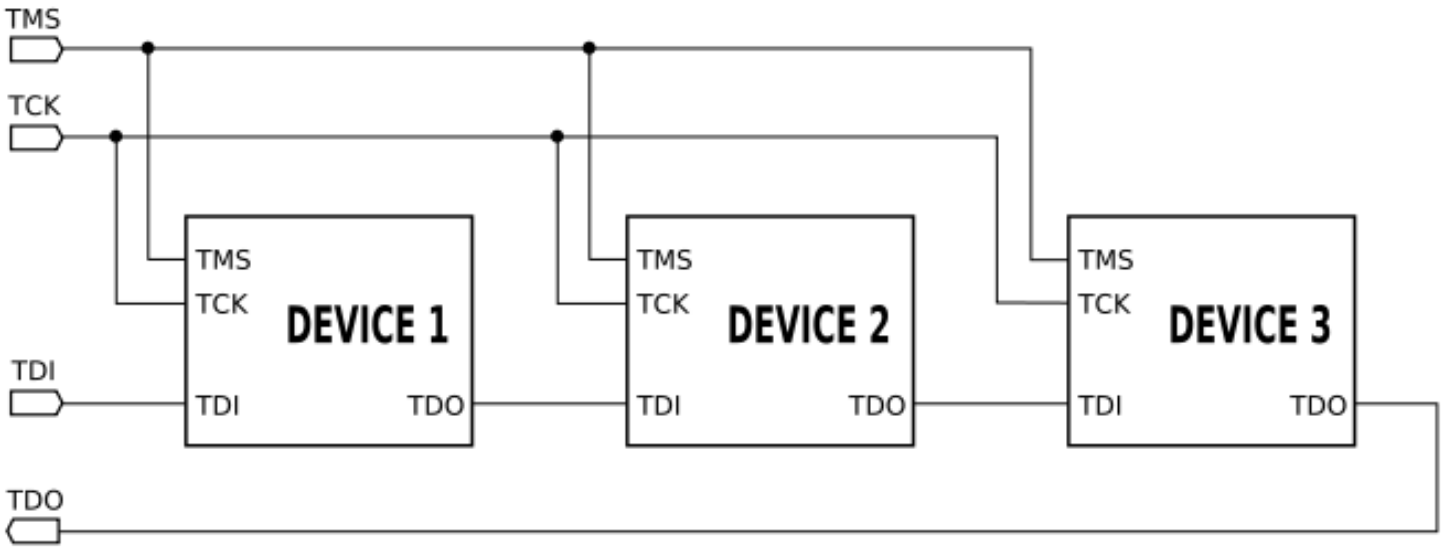


Figure: JTAG Chain[1]

[1] http://en.wikipedia.org/wiki/File:Jtag_chain.svg

Blackfin Scan Chains

Scan Chain Name	**Size (in Bits)**	**Contents**
DBGSTAT_SCAN	16	DBGSTAT bits 15:0
DBGCTL_SCAN	16	DBGCTL
EMUIR_SCAN	32/48/64	EMUIR
EMUDAT_SCAN	32/40/48	EMUDAT
EMUPC_SCAN	32	EMUPC
BOUNDARY_SCAN	N	Chip Boundary scan

Blackfin Watchpoint Unit (1)

Watchpoint Unit registers, accessible in Supervisor and Emulator modes:

- The Watchpoint Status register
- 6 Instruction Watchpoint Address registers (can be grouped)
- 6 Instruction Watchpoint Address Count registers
- The Instruction Watchpoint Address Control register
- 2 Data Watchpoint Address registers (can be grouped)
- 2 Data Watchpoint Address Count registers
- The Data Watchpoint Address Control register

Blackfin Watchpoint Unit (2)

Two operations implement instruction watchpoints:

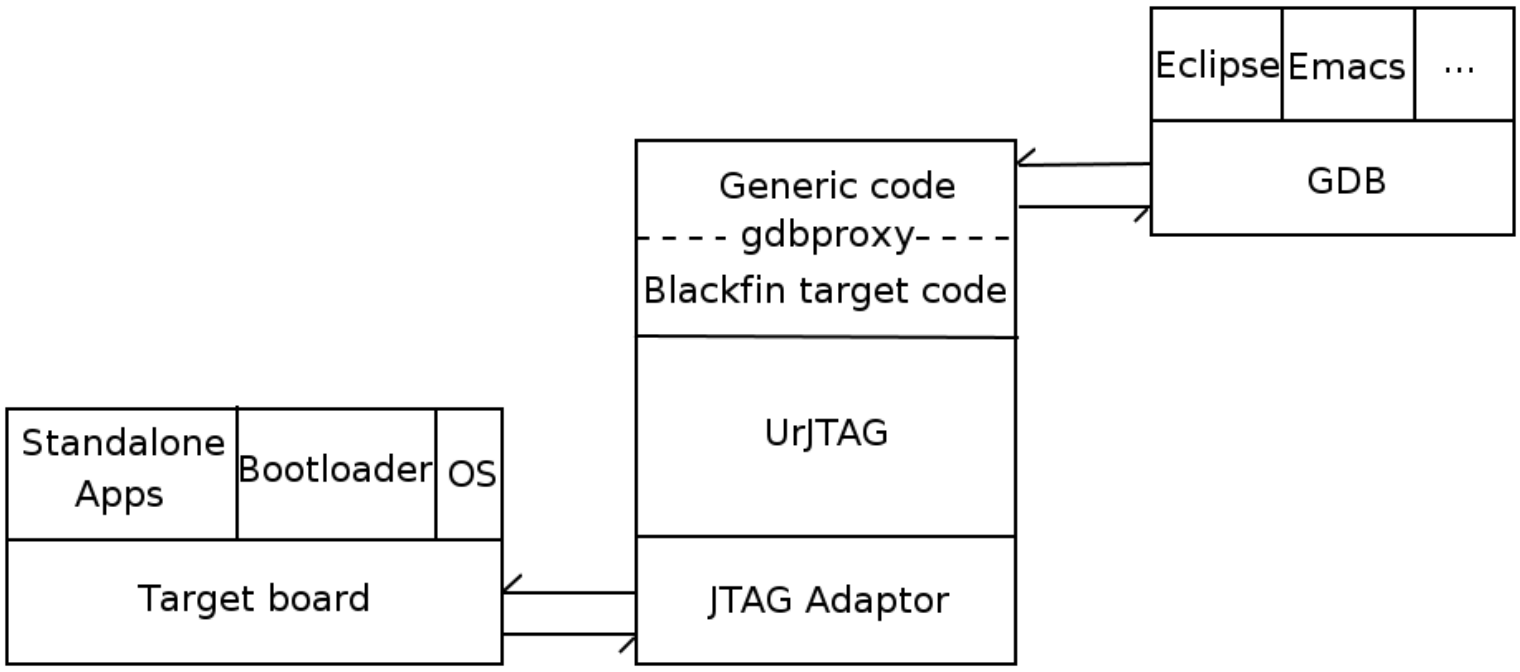
- The values in the six Instruction Watchpoint Address registers, are compared to the address on the instruction bus.
- Corresponding count values in the Instruction Watchpoint Address Count registers, are decremented on each match.

Blackfin Watchpoint Unit (3)

Two operations implement data watchpoints:

- The values in the two Data Watchpoint Address registers, are compared to the address on the data buses.
- Corresponding count values in the Data Watchpoint Address Count registers, are decremented on each match.

The Big Picture



gdbproxy: Generic Code

- Provide stubs for target part of GDB Remote Serial Protocol.
- Parse incoming packets and construct outgoing packets.
- Call corresponding target functions.

gdbproxy: Blackfin Target

- Provide implementation for each stub.
- Use UrJTAG for low level JTAG operations.
- Written from scratch.
- Released under GPLv2.

How It Works

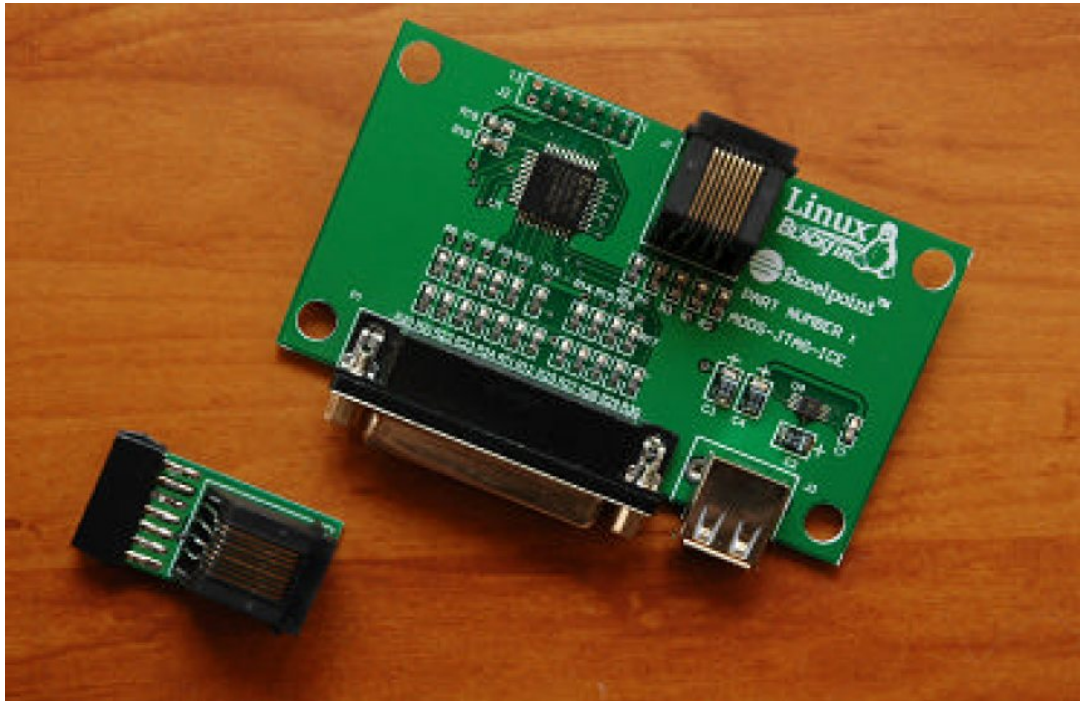
Take single register reading for an example:

- GDB sends a read register packet, `p n`, to gdbproxy.
- gdbproxy receives this packet, parses, calls into `handle_read_single_register_command`.
- `handle_read_single_register_command` calls into `bfin_read_single_register`.
- An instruction `EMUDAT = Rn;` is generated and scanned into EMUIR register.
- Run the JTAG state machine to a specific state so the instruction in EMUIR register is executed.
- EMUDAT_SCAN is selected to shift out the bits in EMUDAT register.
- gdbproxy constructs a reply packet and sends it back to GDB.

Features

- Free software.
- Supported host OS: Linux, Mac OS X, Windows.
- Actively maintained.
- Support multi-core processor: Dual-core BF561.
- Support multi-processor (partially): Multiple Blackfin processors. Blackfin processor chained with other chips, like FPGA.
- Potential to support many JTAG adaptors. All JTAG adaptors supported by UrJTAG can be used.
- Stable. Well tested.

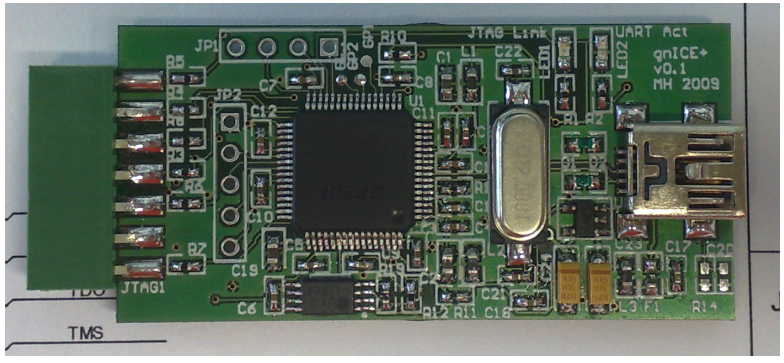
IGLOO Parallel Port Cable



gnICE USB JTAG Cable

- Based on FT2232D/L.
- Support full speed (12Mb/s) of USB 2.0.
- Cheap (about €60).
- Open source hardware.

gnICE+ USB JTAG Cable



- Based on FT2232H.
- Support high speed (480Mb/s) of USB 2.0.
- Also cheap (no price now).
- Open source hardware.

Future Development

- Make libftdi work with libusb-1.0. Linux: Done. Windows: Use libusb-0.1.
- Support non-stop mode for multi-core debugging.
- Fully support multi-processor debugging.

Resources

Source Code of Blackfin gdbproxy

http://blackfin.uclinux.org/gf/project/toolchain/scmsvn/trunk/gdbproxy

Document of Blackfin gdbproxy

http://docs.blackfin.uclinux.org/doku.php?id=toolchain:debug:gdbproxy

Another very interesting JTAG application

http://docs.blackfin.uclinux.org/doku.php?id=bfin:jtag:comm

Thank you for your attention.

Any questions?