Raspberry Pi debugging with JTAG

Arseny Kurnikov
Aalto University

December 13, 2013
Outline

- JTAG
- JTAG on RPi
- Linux kernel debugging
Joint Test Action Group is a standard for a generic transport interface for integrated circuits.

- Boundary scan - read the outputs of the pins
- Debugging - if a board has debug ports (TAP - test access port)
JTAG signals and registers

Registers
- Data Register - read/write the data
- Instruction Register - data register selector

Signals
- Test Data In (TDI) - write data
- Test Data Out (TDO) - read data
- Test Clock (TDI) - synchronization
- Test Mode Select (TMS) - state diagram transitions

Other optional signals for a specific circuit. For example TRST, System Reset.
JTAG transition diagram
Adapter that implements JTAG for a particular board.

- Throughput
- Voltage range
- Supported host software
- Price
Raspberry Pi

- ARM processor, 700 MHz
- 256 Mb - 512 Mb of RAM
- HDMI output
- Runs Linux

\(^1\) Image courtesy of Switched On Tech Design (www.sotechdesign.com.au)
General Purpose Input Output.

- UART (Universal Asynchronous Receiver/Transmitter) - serial port
- SPI (Serial Peripheral Interface)
- \( I^2C \) (Inter Integrated Circuit)
- PWM (Pulse Width Modulator)
- ARM JTAG
GPIO function selection

- Alternative functions on the same pins.
- Selection registers

<table>
<thead>
<tr>
<th>Register</th>
<th>Pin</th>
<th>Configuration</th>
<th>JTAG signal</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPFSSEL0</td>
<td>GPIO4</td>
<td>ALT5</td>
<td>TDI</td>
</tr>
<tr>
<td>GPFSSEL2</td>
<td>GPIO22</td>
<td>ALT4</td>
<td>TRST</td>
</tr>
<tr>
<td>GPFSSEL2</td>
<td>GPIO24</td>
<td>ALT4</td>
<td>TDO</td>
</tr>
<tr>
<td>GPFSSEL2</td>
<td>GPIO25</td>
<td>ALT4</td>
<td>TCK</td>
</tr>
<tr>
<td>GPFSSEL2</td>
<td>GPIO27</td>
<td>ALT4</td>
<td>TMS</td>
</tr>
</tbody>
</table>
• GPU boots from ROM
• The execution is passed to kernel.img on SD card
Configure JTAG on RPi

The snapshot is from the JTAG enabling program \(^1\).

\[
\begin{align*}
ra &= \text{GET32(GPFSEL0)}; \\
ra &= \sim(7<<12); \quad // \text{gpio4} \\
ra &= 2<<12; \quad // \text{gpio4 alt5 ARM_TDI} \\
\text{PUT32}(\text{GPFSEL0},ra); \\
\end{align*}
\]

\(^1\)David Welch, https://github.com/dwelch67
Principal setup

Host
OpenOCD

USB, Ethernet

JTAG Adapter

4 JTAG pins

Target
ARM processor

Arseny Kurnikov  Aalto University  Raspberry Pi debugging with JTAG
Open On-Chip Debugger.

- JTAG and SWD (Serial Wire Debug)
- Boundary scan and debug
- GDB protocol for various processors
interface ft2232
ft2232_device_desc
   "Olimex OpenOCD JTAG ARM-USB-TINY-H"
ft2232_layout olimex-jtag
ft2232_vid_pid 0x15ba 0x002a

FT2232 - USB driver, Future Technology Devices International
OpenOCD configuration for RPi

# Broadcom 2835 on Raspberry Pi

telnet_port 4444
gdb_port 5555
adapter_khz 1000

set _CHIPNAME raspi
reset_config none
set _CPU_TAPID 0x07b7617F

jtag newtap $_CHIPNAME arm -irlen 5
    -expected-id $_CPU_TAPID

set _TARGETNAME $_CHIPNAME.arm
target create $_TARGETNAME arm11
    -chain-position $_TARGETNAME
Open On-Chip Debugger 0.5.0 (2011-08-11-06:56)
Licensed under GNU GPL v2
For bug reports, read
   http://openocd.berlios.de/doc/doxygen/bugs.html
Info : only one transport option; autoselect ’jtag’
1000 kHz
none separate
raspi.arm
Info : max TCK change to: 30000 kHz
Info : clock speed 1000 kHz
Info : JTAG tap: raspi.arm tap/device found:
   0x07b7617f (mfg: 0x0bf, part: 0x7b76, ver: 0x0)
Info : found ARM1176
Info : raspi.arm: hardware has 6 breakpoints, 2 watchpoints
Info : accepting ’telnet’ connection from 4444
target state: halted
<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>adapter_khz [khz]</td>
<td>display/change maximum JTAG speed</td>
</tr>
<tr>
<td>adapter_name</td>
<td>the name of the currently selected adapter</td>
</tr>
<tr>
<td>arm core_state</td>
<td>display/change ARM core state</td>
</tr>
<tr>
<td>arm disassemble</td>
<td>disassemble instructions</td>
</tr>
<tr>
<td>arm reg</td>
<td>display ARM core registers</td>
</tr>
<tr>
<td>exit</td>
<td>exit telnet session</td>
</tr>
<tr>
<td>halt</td>
<td>request target to halt</td>
</tr>
<tr>
<td>jtag</td>
<td>perform JTAG TAP actions</td>
</tr>
<tr>
<td>load_image file address</td>
<td>loads a file into the target memory</td>
</tr>
<tr>
<td>mww</td>
<td>write memory word</td>
</tr>
<tr>
<td>mdw</td>
<td>display memory words</td>
</tr>
<tr>
<td>step</td>
<td>step one instruction</td>
</tr>
</tbody>
</table>
- build it for ARM architecture
- create a kernel.img by the provided tool
- will put the kernel to 0x8000 and continue execution from there
- usually a compressed kernel image
JTAG enabled before kernel

- the tool is modified
- JTAG enabling code is put at 0x8000
- the kernel image is put afterwards
- requires manual resume from the kernel address to continue booting
Drivers

- kernel drivers might override JTAG pins
- SPI and I²C
- disabled by default
- do not interleave with JTAG pins

From spi-bcm2708.c

```c
/* SPI is on GPIO 7..11 */
for (pin = 7; pin <= 11; pin++) {
    INP_GPIO(pin); /* set mode to GPIO input */
    SET_GPIO_ALT(pin, 0); /* set mode to ALT 0 */
}
```
Kernel debugging

- load symbols
- specify remote target
- debug like a normal program
- breakpoints
- examine memory
- stepping
Conclusions

- **JTAG** – debugging instrument for integrated circuits
- enable JTAG on RPi through GPIO
- debug bare metal programs
- enable JTAG before Linux kernel booting
Questions?