Custom Hardware Modelling for FPGAs and Embedded Linux Platforms with QEMU

John Williams, PetaLogix
<john.williams@petalogix.com>

Edgar Iglesias, Axis
<edgar.iglesias@gmail.com>

Embedded Linux Conference 2010 San Francisco
Who are we?

- John Williams
  - FPGA/Linux developer
  - PetaLogix founder/CEO
- Edgar Iglesias
  - Maintainer of CRIS and MicroBlaze QEMU ports
  - Software Engineer at Axis
- Why the double-act?
  - ELC 2009
  - No commercial r/ship between PetaLogix/Axis
Agenda

- FPGA-based SOC/Linux 101 (John)
- QEMU 101 (Edgar)
- MicroBlaze/QEMU (John)
- Cosimulation Case-study and technology (Edgar)
- Looking Forward (John)
FPGA-based SoC design

• An FPGA is a blank digital logic canvas
• The configuration bitstream gives the FPGA its “personality”
  • on every power cycle...
• CPUs and peripherals are just digital logic
FPGA-based SoC design

- Build the system you need
- Standard IP
  - CPU, buses, memory controllers
  - I/O
- Custom IP
  - Coprocessors
  - I/O processors
Linux and FPGAs

• Every platform is different
• Perpetual state of board-bringup?
  • No!
• Generate system description from CAD tools
  • CPU, memory, devices, …
• Configure kernel to match this description
  • Flat device trees
/dts-v1/;
/
{
    DDR2_SDRAM: memory@90000000 {
        device_type = "memory";
        reg = < 0x90000000 0x10000000 >;
    };
    cpus {
        #cpus = <0x1>;
        microblaze_0: cpu@0 {
            ...
        };
    };
    mb_plb: plb@0 {
        compatible = "xlnx,plb-v46-1.03.a", "xlnx,plb-v46-1.00.a", "simple-bus";
        FLASH: flash@a0000000 {
            ...
        };
        IIC_EEPROM: i2c@81600000 {
            compatible = "xlnx,xps-iic-2.00.a";
            interrupt-parent = <&xps_intc_0>;
            interrupts = < 6 2 >;
            reg = < 0x81600000 0x10000 >;
        };
    }
}
Agenda

- FPGA-based SOC/Linux 101 (John)
- QEMU 101 (Edgar)
- MicroBlaze/QEMU (John)
- Cosimulation Case-study and technology (Edgar)
- Looking Forward (John)
“QEMU is a generic and open source machine emulator and virtualizer.”
QEMU 101

• System emulation
  • Emulates a complete machine.
  • Cross run unmodified OS/Firmware.
  • Can also emulate boot-roms including different bootstrap methods.

• Linux-user emulation
  • Emulates the target processor.
  • Cross run linux programs.
  • Syscalls run natively on the host (through an argument translator).
Debugging & Profiling
CRIS/MicroBlaze

- Builtin GDB stub
- Execution traces
- L1 Cache model
- Processor pipeline model
- Interrupt latency tracker
- Kcachegrind compatible statistics
- Coverage
- Track peripheral programming inefficiencies and errors
### KcacheGrind

#### Default Idle

<table>
<thead>
<tr>
<th>Event Type</th>
<th>Incl.</th>
<th>Self</th>
<th>Short</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instruction Fetch</td>
<td>-</td>
<td>-</td>
<td></td>
<td>Ir</td>
</tr>
<tr>
<td>L1 Instr. Fetch Miss</td>
<td>-</td>
<td>-</td>
<td></td>
<td>L1mr</td>
</tr>
<tr>
<td>Data Read Access</td>
<td>5.97</td>
<td>5.97</td>
<td></td>
<td>Dr</td>
</tr>
<tr>
<td>L1 Data Read Miss</td>
<td>0.00</td>
<td>0.00</td>
<td></td>
<td>D1mr</td>
</tr>
<tr>
<td>Data Write Access</td>
<td>0.00</td>
<td>0.00</td>
<td></td>
<td>Dw</td>
</tr>
<tr>
<td>L1 Data Write Miss</td>
<td>0.00</td>
<td>0.00</td>
<td></td>
<td>D1mw</td>
</tr>
<tr>
<td>Insns</td>
<td>2.57</td>
<td>2.57</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Insns_masked</td>
<td></td>
<td></td>
<td></td>
<td>insns</td>
</tr>
</tbody>
</table>

| Cycles               | 1.08  | 1.08 |       |                  |

| Blocks               | -     | -    |       |                  |
| CPI %                | 0.03  | 0.03 |       | CPI%             |
| L1 Miss Sum          | 0.00  | 0.00 |       | L1m = L1mr + D1mr + D1mw |

#### Flat Profile

<table>
<thead>
<tr>
<th>Self</th>
<th>Function</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>60.66</td>
<td>(unknown)</td>
<td>(unknown)</td>
</tr>
<tr>
<td>18.54</td>
<td>memset</td>
<td>(unknown)</td>
</tr>
<tr>
<td>1.43</td>
<td>strcmp</td>
<td>(unknown)</td>
</tr>
<tr>
<td>1.08</td>
<td>default_idle</td>
<td>(unknown)</td>
</tr>
<tr>
<td>0.95</td>
<td>calibrate_delay</td>
<td>(unknown)</td>
</tr>
<tr>
<td>0.86</td>
<td>sysfs_link_sibling</td>
<td>(unknown)</td>
</tr>
<tr>
<td>0.85</td>
<td>sysfs_find_direct</td>
<td>(unknown)</td>
</tr>
<tr>
<td>0.47</td>
<td>get_page_from_freelist</td>
<td>(unknown)</td>
</tr>
<tr>
<td>0.37</td>
<td>memmap_int_zone</td>
<td>(unknown)</td>
</tr>
<tr>
<td>0.25</td>
<td>map_page</td>
<td>(unknown)</td>
</tr>
<tr>
<td>0.22</td>
<td>__free_pages_ck</td>
<td>(unknown)</td>
</tr>
<tr>
<td>0.20</td>
<td>find_next_zero_bit</td>
<td>(unknown)</td>
</tr>
<tr>
<td>0.20</td>
<td>__rmqueue_smallest</td>
<td>(unknown)</td>
</tr>
<tr>
<td>0.18</td>
<td>free_pages_bulk</td>
<td>(unknown)</td>
</tr>
<tr>
<td>0.15</td>
<td>cache_alloc_refill</td>
<td>(unknown)</td>
</tr>
<tr>
<td>0.15</td>
<td>get_pageblock_flags_group</td>
<td>(unknown)</td>
</tr>
<tr>
<td>0.15</td>
<td>simple_strerror</td>
<td>(unknown)</td>
</tr>
<tr>
<td>0.14</td>
<td>vsprintf</td>
<td>(unknown)</td>
</tr>
<tr>
<td>0.12</td>
<td>update_time</td>
<td>(unknown)</td>
</tr>
<tr>
<td>0.11</td>
<td>__free</td>
<td>(unknown)</td>
</tr>
<tr>
<td>0.11</td>
<td>mapin_ram</td>
<td>(unknown)</td>
</tr>
<tr>
<td>0.11</td>
<td>radix_tree_insert</td>
<td>(unknown)</td>
</tr>
<tr>
<td>0.10</td>
<td>free_hot_cold_page</td>
<td>(unknown)</td>
</tr>
<tr>
<td>0.10</td>
<td>__alloc_pages_internal</td>
<td>(unknown)</td>
</tr>
</tbody>
</table>
Agenda

- FPGA-based SOC/Linux 101 (John)
- QEMU 101 (Edgar)
- MicroBlaze/QEMU (John)
- Cosimulation Case-study and technology (Edgar)
- Looking Forward (John)
FPGA-based SoC simulation

- Gate/register level
  - Super accurate
  - Super slow
- Previous attempts at MicroBlaze simulators/emulators from Xilinx
  - ISS
  - VirtualPlatform
QEMU and MicroBlaze

- ELC 2009
  - John talked about Linux and FPGAs
  - Edgar talked about QEMU
  - It made so much sense!
- A month later
  - Edgar was booting MicroBlaze Linux kernels in QEMU
Device Trees (reprise)

- Historically, QEMU machine descriptions are static

“Why don't we just create the QEMU machine model from the same device tree that drives the kernel?”
Fleshing out the details

- MicroBlaze is deeply customisable
  - ALU (mul/shift/div)
  - FPU, MMU, caches
  - Exceptions
- Models for common Xilinx IP
  - Uartlite / uart16550
  - Ethernet lite, temac
This is so awesome, now what?

- Kernel debugging
  - QEMU gives instruction-by-instruction trace/register output
    - How did we end up *there*?
    - Why did this register get trashed?
- Bug triangulation
  - Compare behaviour in QEMU vs HW
  - QEMU modelling fault?
  - HW bug?
Cool and useful?

- Kernel profiling
  - Kcache grind output for the kernel
- MicroBlaze pipeline model still very crude
  - Relative ordering and magnitude of main offenders
QEMU - Other uses

- Kernel Stress Testing
  - Linux Test Project
- Platform testing
  - PetaLinux supports multiple HW platforms on multiple FPGA boards
  - Quickly and automatically test the main features of our BSPs
- Complements HW testing
Accelerating embedded development

- QEMU bundled with PetaLinux
  - Friendly wrappers for virtual network setup etc
- PetaLogix funding cleanup of useful features
  - Parallel (CFI) flash device model
- Can simulate full system boot sequence
  - From flash (including u-boot)
  - From network
Agenda

- FPGA-based SOC/Linux 101 (John)
- QEMU 101 (Edgar)
- MicroBlaze/QEMU (John)

- Cosimulation Case-study and technology (Edgar)
- Looking Forward (John)
Cosimulation Case Study

- AXIS Next video SoC
- Develop new peripheral/Co-Processor to accelerate and offload computationally intensive operations
- Develop software drivers early
ARTPEC Co-Processor

- Profile system (SW & HW) with QEMU
- Modified the design
- Iterated a few times until satisfied
ARTPEC Co-Processor

- Protyped HW on a tiny FPGA system
- Ported the initial C test-bench to run on the FPGA
- FPGA output onto the serial port and compare results (/bin/diff).
Co-Processor RTL

- Implemented the entire System Verilog VMM test bench against the C reference models
- Optimized RTL was later coded based on the low-level C model and the verilog FPGA prototype
Co-processor SW

- QEMU model logs and traps on programming errors.
- Randomized error conditions to test error paths
- Immediate and late interrupts to test both ends of potential race conditions
- Developing against QEMU saved a lot of time avoiding long vcs runs
Co-processor SW

• QEMU-rbus, a remote bus connection
• Implemented a VMM inspired test framework working through QEMU-rbus
• Made it possible to test unmodified SW libraries
• Complementing the verilog simulators to get more coverage
QEMU/r-bus architecture

Guest Software

QEMU  QEMU-rbus

Randomized test driver  Coverage
Randomized test cases  Refmodel
Xactor
BFM
QEMU-rbus
QEMU r-bus

- Thin remote bus layer
- Sub channels used to implement DMA, interrupts or any arbitrary device communication
Agenda

- FPGA-based SOC/Linux 101 (John)
- QEMU 101 (Edgar)
- MicroBlaze/QEMU (John)
- Cosimulation Case-study and technology (Edgar)

- Looking Forward (John)
Are we there yet?

- The whole point of FPGA-based SoC is custom...
  - Architectures
  - devices/IP
- New device models written in C
  - Requires a rebuild of QEMU
  - We'd like something a bit more user-friendly
Extending QEMU

• R-bus
  • Generic remote bus protocol for connecting to models outside the main QEMU binary
• What about things we find in the device tree without a built-in model?
  • compatible="<my-device-v1.00.a>"
  • currently ignored
Extending QEMU

• Wouldn't this be better?

```c
if(h=dlopen("my-device-v1.00.a.so",))
{
    /* insert magic here */
}
```
Even more magic

- C models from VHDL/Verilog?
  - GHDL, Verilator
- r-bus
  - General purpose QEMU co-simulation framework
  - Hardware in the loop
Q & A

Thank you