Overview

• Leverage partial reconfiguration to build PYNQ overlays on demand.
• “Software Library” approach to hardware features:
  • Collection of hardware modules loaded on demand.
  • Support graceful Linux driver load/unload.
• Allow average PYNQ users to leverage FPGA reconfigurability.
• Create custom hardware configurations without hardware expertise.

Design Flow

• Custom IP to add PR functionality to your PYNQ overlay
  • AXI memory-mapped connection
  • Integrated with the PYNQ framework
  • Python drivers loaded automatically
  • Kernel drivers also supported
• Publicly available at https://github.com/byuccl/pynq_prio_pip.git
*current work not fully released.

Benefits

On-Demand Scalability
• Library of modules are given to the user
  • 100s of modules is feasible
• Library can include different configurations of an IP
• User chooses at run-time which hardware controllers will be present in the system.

Simplified Hardware Design
• Each module is a stand-alone Vivado project
• Faster compile (5-10 min vs 1 hour)
• Static design remains locked; no redoing timing closure
• Gradual introduction to HW design for educational use

Conclusions

• Proposed framework of PR-based bitstream assembly provides several advantages while maintaining a simple HW/SW design flow.
• With transitions to larger devices (PYNQ on UltraScale+) the need for a scalable, modular approach will only increase.

The Original PYNQ I/O System

• “Kitchen sink” approach with many HW modules.
• Fixed configuration; any changes require a full recompile.

Our PR “On-Demand” Design

• 6 PR regions.
• Any region can do any protocol.
• New modules added as needed.
• 20% reduction in area.

Device Tree Overlays

• Device tree fragments can be added to the tree at run-time; essential for supporting a PR framework.

User Experience

• We provide a library of classes that abstract away the PR and device tree modifications from the user.
• User can make a single call to reconfigure a region.
• The driver for the old module is gracefully removed and the new driver is automatically loaded.

Device Tree Fragment

```c
prf: pr0040a3a0000 { 
    compatible = "apio10-410-1616"; 
    reg = <0x10a0000 0xe00000>; 
    interrupts = <0x0 0x0000>; 
};
prf: pr0040a32a0000 { 
    compatible = "apio10-410-1616"; 
    reg = <0x12a0000 0xe00000>; 
    interrupts = <0x0 0x0000>; 
};
```

Sample User Application Code

```c
... 
prof: pr0040a1a0000 { 
    compatible = "apio10-140-1016"; 
    reg = <0x10a0000 0xe00000>; 
    interrupts = <0x0 0x0000>; 
    }; 
```

Vivado Project for Timer PR Module

Device Tree Fragment Sample User Application Code

Conclusions

• Proposed framework of PR-based bitstream assembly provides several advantages while maintaining a simple HW/SW design flow.
• With transitions to larger devices (PYNQ on UltraScale+) the need for a scalable, modular approach will only increase.