Microkernel Dedicated for Dynamic Partial Reconfiguration on ARM-FPGA Platform

XIA Tian, Jean-Christophe Prévotet and Fabienne Nouvel
INSA, IETR, UMR 6164, F-35708 RENNES

EWiLi’14, Lisboa, Portugal
2014-11-13
Motivation and Object
Overview of Proposed Platform
Microkernel Architecture
Case study and Analysis
Motivation and Object

DPR: Dynamic Partial Reconfiguration

- Reduced hardware resource utilization
- Reduced reconfiguration latency
- Improved design efficiency

Supporting technologies

- Bare-metal application
- Existing OS extension (Linux): RAMPSoCVM
- Embedded OS: CAP-OS, ReconOS
Object: Microkernel-supported ARM-FPGA platform

Advantages:

- Higher security level: completely isolated environment
- Smaller trust computing base (TCB) than traditional OS
- Mixed criticals: hard/soft/non real-time applications
Proposed Platform

Zynq-7000 All programmable SoC

- Dual core ARM Cortex-A9 (Single processor currently)
- 512MB DDR
- AXI bus based PS/PL Interface
- Processor Config Access Port (PCAP)

PS (Processing System)
PL (Programmable Logic)
Proposed Platform

Proposed Architecture

- **PCAP**
- **DMA**
- **HW task Manager**
- **Microkernel**
- **ARM**
- **DDR**

- **PS**
  - **Interconnect**
  - **AXI_HP**
  - **AXI_GP**
  - **DMA**

- **PL**
  - **PRR Controller**
  - **PCAP**

- **Partial Reconfiguration Regions (PRR)**
  - **HW Task1**
  - **HW Task2**
  - **HW Task3**

- **Hardware tasks**
- **Communication Interface**
- **PRR controller**
Proposed Architecture

- **DDR**
- **T1, T2, T3, T4**
- **HW task data**
- **Guest**
- **HW task Manager**
- **Microkernel**
- **ARM**
- **Interconnect**
- **PS**
- **PL**
- **AXI_HP**
- **AXI_GP**
- **PRR Controller**
- **DMA**
- **PCAP**
- **Partial Reconfiguration Regions (PRR)**

- Hardware tasks
- Communication Interface
- PRR controller
Proposed Platform

Hardware tasks

- **Container:** PRR (Partial Reconfig Region)
- **Stored by:** DDR Memory
- **Download by:** DMA transfer through PCAP
- **Reconfig. overhead is linearly correlated to PRR size,** which means **predictable latency** of hw task switch.

### DDR

<table>
<thead>
<tr>
<th>HW task</th>
<th>Func</th>
<th>Bitstream file</th>
</tr>
</thead>
<tbody>
<tr>
<td>HW task1</td>
<td>FFT-512</td>
<td>Hwt1.bit</td>
</tr>
<tr>
<td>HW task2</td>
<td>FFT-1024</td>
<td>Hwt2.bit</td>
</tr>
<tr>
<td>HW task3</td>
<td>GSM</td>
<td>Hwt3.bit</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Proposed Platform

Proposed Architecture

- **Hardware tasks**
- **Communication Interface**
- **PRR controller**
**Proposed Platform**

### Communication Interface

<table>
<thead>
<tr>
<th></th>
<th>AXI_GP</th>
<th>AXI_HP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>Mode</td>
<td>Master</td>
<td>Slave</td>
</tr>
<tr>
<td>Access</td>
<td>Unified Addr space</td>
<td>DMA</td>
</tr>
<tr>
<td>Throughput</td>
<td>600MB/s</td>
<td>1200MB/s</td>
</tr>
</tbody>
</table>

- **AXI_GP**: Control HW task behaviors
- **AXI_HP**: High speed data exchange
Proposed Architecture

- Hardware tasks
- Communication Interface
- PRR controller & synchronization
Proposed Platform

PRR Controller Structure

- Cooperate with HW task manager
- Configure HW task parameters:
  - DMA address, data size
  - Working mode
- Generate HW task Synchronization IRQ
- Monitor HW switch

PRR Controller

<table>
<thead>
<tr>
<th>Status</th>
<th>Parameters</th>
<th>IRQ Status</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HW Task1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PRR1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HW Task2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PRR2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HW Task3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PRR3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Proposed Platform

PRR Controller Monitor

- Cooperate with HW task service to **Monitor HW switch**.
- Guarantee the HW task security, avoiding **invalid data output** and undesired task state.
Microkernel Architecture

- **Small TCB Size** (Kernel: 8 KLOC; User environment: 3 KLOC)
- **De-privileged C Library** to handle privileged operations: Cache, page table
- **Bootloader** for guest OS/Applications (User Server)
- **Separate virtual address spaces** for kernel and guests
- **Specific Priority-based** round-robin scheduling
Microkernel Architecture

Isolated virtual address spaces

Mini-NOVA Privilege Level (PL)

- PL0: Host/User/FPGA
- PL1: Host/User
- PL2: Host
- Mix: PL1/PL2
- Unmapped

0xFFFF1000
0xFFFF0000
0xFE000000
0xE0000000
0xCA000000
0xC0000000
0xFFF00000
0x00000000

- User space
- Kernel
- Except Vector
- Peripherals
- FPGA Space
- Bitstreams
- HW task data
- User space

Mini-NOVA Microkernel
HW task Manager
Guest 1
HW Task
Priority-based scheduling

**Scheduling Principle:**
- HW task requires tighter time constrain (hard real-time)
- Quick response for the HW task management should be guaranteed

**Functions:**
- **HW Manager Enqueue():** Add the HW task manager into the run queue and preempt lower prio
- **HW Manager Dequeue():** Remove the EC of the HW task manager from the run queue
- **reschedule():** Update the schedule and dispatch the highest priority EC
HW Task Manager

- HW tasks’ switch/config is isolated from other guests, should be done by the HW task manager.

- This mechanism is to ensure the security of the FPGA fabric.

- Process flow:
  1) Guest’s System call (hw task id, args)
  2) Check HW task ready/args
  3) Switch/Configure hw task
  4) Return to guest

.Syscall_HW_Manager
(HW task id, arg01, arg02, arg03)
Case study and analysis

Case Description

Proposed scenario:

Computing system within a mobile wireless terminal, which is capable of dynamically change its configuration in order to obtain the best level of performances according to the channel conditions.

![Diagram showing the proposed scenario](image-url)
Case study and analysis

Implementation

- Modulation and IFFT execute in pipeline, the reconfiguration will cause a suspension.
- To minimize the significant time overhead we proposed a multiple-path structure.

**t1**: Syscall HW Manager;
**t2**: PCAP Start,
HW Manager dequeue;
**t3**: PCAP Done;
**t4**: Data frame over.

Pipeline suspension

- FFT-512 (PRR1)
- FFT-8192 (PRR2)
### Results & Analysis

<table>
<thead>
<tr>
<th>Task name</th>
<th>Type</th>
<th>Execution Time(ms)</th>
<th>Reconfig.Time(ms)</th>
<th>Resource Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>ChannelSensor</td>
<td>SW</td>
<td>3</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>HW Manager</td>
<td>SW</td>
<td>0.0096</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>Guest Switch</td>
<td>SW</td>
<td>0.00232</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>QAM (4/16/64)</td>
<td>HW</td>
<td>0.09-0.03 (1 frame)</td>
<td>0.231</td>
<td>2%</td>
</tr>
<tr>
<td>IFFT (256-8192)</td>
<td>HW</td>
<td>0.006-0.168 (1 frame)</td>
<td>2.72</td>
<td>13%</td>
</tr>
</tbody>
</table>

- Frame size: 18,800 bits
- FPGA Freq: 100MHz
- ARM Freq: 660MHz

**Reconfig Overhead:**

Pipeline suspension: **0.168 ms** (8096 points FFT).
Conclusions

• An ARM based microkernel is built on Zynq-7000 architecture.

• Propose hardware task manager and PRR/PRR Controller to support DPR.

• Apply separate memory space and multiple access privileges to improve the system security, especially for FPGA access.

• Use priority-based round-robin scheduling to guarantee run-time hw task management.

• Perspectives:
  - Further virtualization with Linux and other RTOS;
  - Performance evaluation with standard benchmarks.
Thank you for your attention!

XIA Tian
INSA, IETR, France
EWiLi’14, Lisboa, Portugal
2014-11-13
Proposed Platform

Proposed Architecture

- Hardware tasks
- Communication
- PRR controller

- PRR: Partial Reconfiguration Region
Priority-based scheduling

Guest 1 (Prio 1)
Guest 2 (Prio 1)
Guest 3 (Prio 1)
HW task Manager (Prio 2)

EC 1  EC 2  EC 3  EC HW

Execution Context (EC):
- CPU/FPU register state
- Coprocessor state
- Page Table Address (TTBR)
- Scheduling Priority

Mini-NOVA Microkernel
HW Task Manager

- HW tasks’ switch/config is isolated from other guests, should be done by the HW task manager.
- This mechanism is to ensure the security of the FPGA fabric.

Process flow:
1) Guest’s System call (hw task id, args)
2) Check HW task ready/args
3) Switch/Configure hw task
4) Return to guest

Syscall_HW_Manager
(HW task id, IRQ_en, arg01, arg02, arg03)
**Microkernel Architecture**

---

**Initialization**

1. Start → Init → Dequeue (wait)

**Processing Loop**

- Dequeue (suspend)
- IRQ_PRR_Ready
- Reschedule

---

**Arguments:**

(HW task ID, args)

**Args Legal?**

- Y → HW task exist?
- N → Dequeue (error)

**HW task exist?**

- Y → Transfer args
- N → Ready for Reconfig?

**Dequeue (success)**

**Transfer args**

**PCAP transfer**

**Dequeue (error)**

**Dequeue (suspend)**

**Reschedule**

---

**Syscall_HW_manager(ID, args)**

**Guest 1** → **HW task Manager**

**HW Task1** → **HW Task2**

---

**Syscall_HW_Manager (HW task id, arg01, arg02, arg03)**