Towards Real-Time Operating Systems for Heterogeneous Reconfigurable Platforms

Marco Pagani, Mauro Marinoni, Alessandro Biondi, Alessio Balsini and Giorgio Buttazzo

Scuola Superiore Sant’Anna, Pisa, Italy
HETEROGENEOUS PLATFORMS

• Emerging in the **embedded** domain

![Diagram of heterogeneous platforms with various components and frequencies]
WHAT IS A FPGA?

- **A field-programmable gate array (FPGA) is an integrated circuit** designed to be configured (by a designer) **after** manufacturing.

- FPGAs contain an array of **programmable logic blocks**, and a hierarchy of reconfigurable interconnects that allow to “**wire together**” the blocks.

> from ni.com
WHY FPGAs?

Top 5 benefits (according to National Instruments)

- Cost
- Time-to-market
- Reliability
- Long-term maintenance

Performance

Ad-hoc hardware acceleration of specific functionalities with a consistent speed-up
DYNAMIC PARTIAL RECONFIGURATION

- Modern FPGA offers **dynamic partial reconfiguration (DPR)** capabilities.
- DPR allows **reconfiguring** a portion of the FPGA at **runtime**, while the rest of the device continues to operate.
DYNAMIC PARTIAL RECONFIGURATION

• DPR opens a new dimension in the resource management problems for such platforms.
• Likewise multitasking, DPR allows virtualizing the FPGA area by “interleaving” (at runtime) the configuration of multiple functionalities.

Analogy with multitasking

<table>
<thead>
<tr>
<th>CPU</th>
<th>FPGA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Context switch</td>
<td>DPR</td>
</tr>
<tr>
<td>CPU registers</td>
<td>FPGA config memory</td>
</tr>
<tr>
<td>Tasks</td>
<td>Hardware accelerators</td>
</tr>
<tr>
<td>SW</td>
<td>programmable logic</td>
</tr>
</tbody>
</table>
THE PAYBACK

- DPR does not come for free!

- Reconfiguration times are \( \sim 3 \) orders of magnitude higher than context switches in today’s processors.

- Determines further complications in the resource management problems.
What happened to reconfiguration times in the last 16 years?
Theoretical Throughput (MB/s)

Year


Virtex II  Virtex II-pro  Virtex 4  Virtex 5  Virtex 6  Stratix V  Zynq Ultrascale+  Ultrascale

THE TREND
THE TREND

Very promising trend!

Year

Theoretical Throughput (MB/s)


Zynq Ultrascale+
EXPLOITING DPR FOR REAL-TIME APPLICATIONS

A proposal for a system architecture
SYSTEM ARCHITECTURE

- System-on-chip (SoC) that includes:
  - One processor;
  - One DPR-enabled FPGA fabric;
- DRAM shared memory.

![System Architecture Diagram]
COMPUTATIONAL ACTIVITIES

• Software Tasks (SW-Tasks)
  • Periodic (or sporadic) real-time tasks running on the processor;

• Hardware Tasks (HW-Tasks)
  • Functionalities implemented in programmable logic and executed on the FPGA.

while(true) {
  ...
  <prepare input data>
  EXECUTE_HW_TASK(my_hw_task);
  <retrieve output data>
  <...>
  suspend_until(next_activation);
}
SLOTTED APPROACH

- FPGA area is statically **partitioned** in slots.
- HW-Tasks are programmed onto slots.
**RECONFIGURATION INTERFACE**

- **DPR**-enabled FPGAs dispose of a **FPGA reconfiguration interface** (**FRI**) (e.g., PCAP, ICAP on Xilinx platforms).
- In most **real-world** platforms, the **FRI**
  - can reconfigure a slot **without affecting** HW-tasks that execute on the other slots;
  - is an **external device** to the processor (e.g., like a DMA);
  - can program **at most one** slot at a time.

---

**Unique resource → Contention**
A SYSTEM PROTOTYPE
REFERENCE PLATFORM

Xilinx Zynq-7000 SoC
- 2x ARM Cortex A9
- Xilinx series-7 FPGA
- AMBA Interconnect
HW-TASK INTERFACE

• Each **bitstream** has been equipped with a common interface:

  • **AXI Interface** – to **access memory** (via AMBA Interconnect)
  
  • **Interrupt signal** – to **notify HW-task completion**
HW-task management implemented as a user-level library in FreeRTOS.

**FIFO semaphores** have been used to regulate the contention of slots and FRI
- Counting semaphore for the slots;
- Mutex to protect the FRI.

**Cache coherency** for input/output data of HW-tasks is *explicitly* handled by the library
- The AXI interfaces in the programmable logic are not subject to cache coherency
HW-TASK DESCRIPTOR

- One bitstream per each slot
  - bitstream relocation is not supported by the Xilinx tools
  - Example of size: 4 slots → 4 x 338 KB
  - Bitstreams are preloaded in RAM at the system startup

- Two callbacks
  - Start and completion of the HW-task

- Input and output parameters
  - Memory pointers (or data)
EXPERIMENTAL STUDY

Is the use of DPR viable for real-time applications?
EXPERIMENTAL SETUP

Xilinx Zybo Board with Zynq-7010

Saleae Logic Analyzer
CASE STUDY

• Four computational activities:
  • **Sobel** image filter; 800x600 @ 24-bit
  • **Sharp** image filter;
  • **Blur** image filter; 640x480 @ 24-bit
  • **Matrix** multiplier. 64x64 elements

• Both **HW-task** and **SW-task** versions have been implemented
  • **Xilinx Vivado HLS** synthesis tool for HW-task
  • **C** language for SW-tasks
QUESTIONS

1. **Speed-up evaluation**
   How much is it possible to speed-up the execution with an implementation in **programmable logic**?

2. **Reconfiguration times profiling**
   What is the actual **throughput** for reconfiguring a portion of the FPGA?

3. **Response-time evaluation**
   Besides **reconfiguration times** and the additional **contention**, can DPR provide **benefits** for real-time applications?
SPEED-UP EVALUATION

- **CPU**: Cortex A9 @ 650Mhz
- **FPGA**: Artix-7 @ 100Mhz

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Mult [ms]</th>
<th>Sobel [ms]</th>
<th>Sharp [ms]</th>
<th>Blur [ms]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Observed HW execution times</td>
<td>0.785</td>
<td>12.710</td>
<td>24.631</td>
<td>24.628</td>
</tr>
<tr>
<td>Observed SW execution times</td>
<td>1.980</td>
<td>115.518</td>
<td>304.975</td>
<td>374.785</td>
</tr>
<tr>
<td>Speedup Average</td>
<td>2.523</td>
<td>9.089</td>
<td>12.381</td>
<td>15.217</td>
</tr>
<tr>
<td>Speedup Minimum</td>
<td><strong>2.515</strong></td>
<td><strong>9.087</strong></td>
<td><strong>12.380</strong></td>
<td><strong>15.216</strong></td>
</tr>
</tbody>
</table>

**Up to 15x**
RECONFIGURATION TIMES

• Time needed to reconfigure a region of ~4K logic cells, 25% of the total area

• **Memory-intensive** SW activity: data transfer between two 32MB buffers (> L2 cache size)

![Histograms showing reconfiguration times](chart.png)

- Without memory interference: < 3 ms
- ~110 MB/s


RESPONSE TIMES

- The case study is **not feasible**
  - with a **pure SW** implementation (CPU overloaded);
  - with **any combination** of SW and statically configured HW tasks.

With DPR seems **feasible**!

<table>
<thead>
<tr>
<th>SW-task</th>
<th>Mult</th>
<th>Sobel</th>
<th>Sharp</th>
<th>Blur</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Period [ms]</strong></td>
<td>30</td>
<td>50</td>
<td>80</td>
<td>100</td>
</tr>
<tr>
<td>Cache flush [ms]</td>
<td>0.030</td>
<td>1.123</td>
<td>1.754</td>
<td>1.754</td>
</tr>
<tr>
<td>Cache invalidate [ms]</td>
<td>0.017</td>
<td>1.240</td>
<td>1.939</td>
<td>1.939</td>
</tr>
<tr>
<td>Observed Average [ms]</td>
<td>3.829</td>
<td>17.603</td>
<td>31.416</td>
<td>35.624</td>
</tr>
<tr>
<td>Response time</td>
<td>Longest [ms]</td>
<td>24.017</td>
<td>20.418</td>
<td>33.086</td>
</tr>
</tbody>
</table>
THE RESULTS ARE ENCOURAGING!
CONCLUSIONS

• Experimental study aimed at evaluating the use of DPR for implementing a timesharing mechanism to virtualize the FPGA area.

• We proposed a possible system architecture and developed a prototype as a user-level library on FreeRTOS.

• Reconfiguration times in today’s platforms are not prohibitive (and are likely to decrease in future)

• DPR can improve the performance of real-time application upon static FPGA management
Two major **bottlenecks** have been identified

- The **FRI** is an **unique** resource
  - More FRIs would help in reducing the contention

- The **reconfiguration process** creates **additional contention** on the **bus**
  (bitstreams are stored in the main memory)
  - Dedicated memory for bitstreams would improve performance and predictability
CHALLENGES
a lot...

• Scheduling algorithms for HW-tasks
• Real-time Analysis
  • Task scheduling, Interconnect,…
• Investigation on partitioning approaches for the FPGA
• Improved inter-task communication mechanisms
• Design and implementation of RTOS support
Thank you!

Alessandro Biondi
alessandro.biondi@sssup.it