A Framework for Supporting Real-Time Applications on Dynamically Reconfigurable FPGAs

Alessio Balsini

Scuola Superiore Sant’Anna, Pisa, Italy
Breaking In
Embedded Systems Nowadays

More and more **heterogeneous**

![Diagram showing different components and frequencies](image-url)
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 allows to “wire together” the blocks.
Why FPGAs?

Top 5 benefits *(according to National Instruments)*

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

Performance

Ad-hoc **hardware acceleration** of specific functionality results in a consistent **speed-up**
Dynamic Partial Reconfiguration

Modern FPGAs offer dynamic partial reconfiguration (DPR)

DPR allows to reconfigure 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><strong>CPU</strong></th>
<th><strong>FPGA</strong></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>Bitstream</td>
</tr>
</tbody>
</table>
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?
Reconfiguration Times Trend

Theoretical Throughput (MB/s)

- 100
- 300
- 500
- 700
- 900

Year

- 2000
- 2002
- 2004
- 2006
- 2008
- 2010
- 2012
- 2014
- 2016

Virtex II
Virtex II Pro
Virtex 4
Virtex 5
Virtex 6
Virtex 7
Virtex UltraScale
Virtex UltraScale+

Alessio Balsini – IWES 2016
Reconfiguration Times Trend

Theoretical Throughput (MB/s)

Year


Virtex II
Virtex II Pro
Virtex 4
Virtex 5
Virtex 6
Virtex 7
Virtex UltraScale
Virtex UltraScale+

Very **promising** trend!
FRED

Framework for supporting real-time applications on FPGAs with DPR feature
More Advanced Architecture

System-on-chip (SoC) that includes:

- Processor(s)
- DPR-enabled FPGA fabric

DRAM shared memory

![Diagram of More Advanced Architecture](image-url)
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

```c
SW-task (s) {
  /* ... */
  <prepare input data for i-th HW-task>
  EXECUTE_HW_TASK (i);
  <retrieve output data from i-th HW-task>
  /* ... */
  <prepare input data for j-th HW-task>
  EXECUTE_HW_TASK (j);
  <retrieve output data from j-th HW-task>
  /* ... */
  SUSPEND_UNTIL (next_activation);
}
```
FPGA Partitioning: Slotted Approach

Reconfigurable FPGA area is statically partitioned in slots (chosen at design time)

HW-tasks are programmed onto those slots
Each **bitstream** has been equipped with a **common interface**:

- **AXI Interface** to access memory (via AMBA Interconnect)
- **Interrupt signal** to notify HW-task completion

![Diagram](image)
HW-Task Descriptor

One bitstream for each slot accessible by the HW-task

- Bitstream relocation is not supported by the architecture
  \[\text{Size Example: 4 slots } \rightarrow 4 \times 338 \, KB\]

- 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)
Reconfiguration Interface

DPR-enabled FPGAs embed 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 executing on other slots
- Is an external device to the processor (e.g., like a DMA device)
- Can program a single slot at a time

Single resource  →  Contention
Simple Schedule Example

- 2 SW-tasks
- 2 HW-tasks
- 2 Slots

Contention on FRI

SW-task activation
- SW-task execution
- HW-task execution

Acceleration request
- FRI waiting
- FPGA reconfiguration

SW₁
SW₂
HW₁
HW₂
Proposes a **scheduling infrastructure** for the management (and predictability) of **HW-tasks** requests from **SW-tasks**
Practical Validation and Profiling
Reference Platform

Xilinx Zynq-7000 SoC

- 2x ARM Cortex A9
- Xilinx series-7 FPGA
- AMBA Interconnect
**Software Support**

**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
Experimental Setup

Saleae Logic Analyzer

Xilinx Zybo Board with Zynq-7010
Questions

Speed-up evaluation

- How much is it possible to speed-up the execution with an implementation in programmable logic?

Reconfiguration times profiling

- What is the actual throughput for reconfiguring a portion of the FPGA?

Response-times evaluation

- Besides reconfiguration times and the additional contention, can DPR provide benefits for real-time applications?
Case Study

Four computational activities:

- **Sobel** image filter
- **Sharp** image filter
- **Blur** image filter
- **Matrix** multiplier

Both **HW-task** and **SW-task** are implemented in **C** language

- **Xilinx Vivado HLS** synthesis tool generates the HW-task bitstreams
### Speed-up Evaluation

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

<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><strong>Observed HW execution times</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Average [ms]</td>
<td>0.785</td>
<td>12.710</td>
<td>24.631</td>
<td>24.628</td>
</tr>
<tr>
<td>Longest [ms]</td>
<td>0.785</td>
<td>12.712</td>
<td>24.633</td>
<td>24.629</td>
</tr>
<tr>
<td><strong>Observed SW execution times</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Average [ms]</td>
<td>1.980</td>
<td>115.518</td>
<td>304.975</td>
<td>374.785</td>
</tr>
<tr>
<td>Longest [ms]</td>
<td>2.017</td>
<td>115.521</td>
<td>304.994</td>
<td>374.811</td>
</tr>
<tr>
<td><strong>Speedup</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Average</td>
<td>2.523</td>
<td>9.089</td>
<td>12.381</td>
<td>15.217</td>
</tr>
<tr>
<td>Minimum</td>
<td>2.515</td>
<td>9.087</td>
<td>12.380</td>
<td>15.216</td>
</tr>
</tbody>
</table>

**Up to 15x**
Reconfiguration Times Profiling

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)

Without memory interference

With memory interference

< 3 ms

~110 MB/s
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>Period [ms]</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 Longest [ms]</td>
<td>24.017</td>
<td>20.418</td>
<td>33.086</td>
<td>43.160</td>
</tr>
</tbody>
</table>
Questions

Speed-up evaluation

- How much is it possible to speed-up the execution with an implementation in programmable logic?

Reconfiguration times profiling

- What is the actual throughput for reconfiguring a portion of the FPGA?

Response-times evaluation

- Besides reconfiguration times and the additional contention, can DPR provide benefits for real-time applications?
Experimental Results
Schedulability Analysis

Response-time analysis experiments on different scenarios:

- **Static**: infinite FPGA area, slots statically assigned to HW-tasks
- **FRED-P**: framework, with preemptive FRI
- **FRED-NP**: framework, with non-preemptive FRI
- **Software**: purely software approach

**FPGA:**

- Performance tuned according to the Zybo board measurements

**Synthetic workload:**

- Random periods chosen from **buckets**
- Utilization factors randomly chosen under uniform distribution
Schedulability Analysis: Architecture 1
Schedulability Ratio

**Speed-up:** 1x \( (\text{pessimistic}) \)

\( U^H: 0.1 \) - kind of utilization factor determining hardware load of a FPGA slot

- **Purely software**
- **Ideally infinite area**
Schedulability Ratio

Speed-up: $1x$ (pessimistic)

$U: 0.1$

![Graph showing Schedulability Ratio with labels for Purely software and Ideally infinite area]
Schedulability Analysis: Architecture 2

CPU

SW-task
HW-task

+  +

HW-task
HW-task

Partition
Slot  Slot

Partition
Slot  Slot

Alessio Balsini – IWES 2016
Schedulability Ratio

Speed-up: \(3x\) - \(U: 0.1\) - \(UH: 0.1\)

- Purely software
- Ideally infinite area
Challenges / Work in Progress

FRI is **unique** and **non preemptive**

- **Multiple FRI** modules improve the performance on reconfiguration contention
- Implement a **preemptive FRI** interface

Reconfiguration and data passing generate **bus contentions**

- **Local memories** for each slot would reduce the bus load
- Improve **inter-task communication** mechanisms

Optimize the reconfigurable area in **slots/partitions**?

Integrate **new scheduling algorithms** (with analysis) for HW-tasks

System **simulator** (average performance, power, …)
Conclusions

Introduced **FPGAs** and **DPR** feature for implementing a timesharing mechanism to virtualize the FPGA area

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

Presented **FRED**,

- Proposed a **model** for **SW-** and **HW-tasks**.
- **Managed** of hardware **acceleration requests** on DPR-enabled systems.
- Developed of a **prototype** and performed **measurements** on a real platform (Zybo board) as a user-level library on FreeRTOS
- Performed **response-time analysis** experiments of our framework on different use cases
Conclusions

Introduced **FPGAs** and **DPR** feature for implementing a timesharing mechanism to virtualize the FPGA area

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

**FRED** made feasible systems that were not!

- Proposed a model for SW- and HW-tasks.
- Managed of hardware acceleration requests on DPR-enabled systems.
- Developed of a **prototype** and performed **measurements** on a real platform (Zybo board) as a user-level library on FreeRTOS.
- Performed **response-time analysis** experiments of our framework on different use cases.
Thank you!

Alessio Balsini
a.balsini@sssup.it