Prototyping Pipelined Applications on a Heterogeneous FPGA Multiprocessor Virtual Platform

Antonino Tumeo, Marco Branca, Lorenzo Camerini, Marco Ceriani, Gianluca Palermo, Fabrizio Ferrandi, Donatella Sciuto

Politecnico di Milano

Matteo Monchiero,
HP Labs

ASPDAC '09 – Yokohama – January 21st 2009
Outline

- Introduction
- Related work
- Architecture
- Software layer
- Case studies
- Experimental results
- Conclusions
Motivations

- FPGA based Multiprocessor Systems-on-Chip (MPSoCs) are an appreciable platform for prototyping and final implementation of embedded systems.
- Pipelining is an appealing programming paradigm for embedded multiprocessor systems.
  - Streaming applications: audio/video compressors, data packet coding/decoding.
- How to ease and validate the development of pipelined applications onto heterogeneous platforms?
Paper objectives

- Introducing a multiprocessor platform on FPGA
  - Integrating commercial-off-the-shelf (COTS) and ad hoc components
    - Heterogeneous
    - Interrupt based only synchronization and sequencing
- Describing a software layer that runs on top of this platform
  - Sufficiently generic to be exploited by future automated exploration flows
- Proposing several case studies of applications with unbalanced pipeline stages that run on this platform
Related work

- Architectures and methods for pipelined applications
  - Stanford Imagine, MIT StreamIT+RAW
- Pipelined applications on ASIP based platforms
  - Shee et al.
- FPGA multiprocessor system prototypes
  - RAMP initiative
  - Homogeneous, shared memory architectures
    - Xilinx: Tumeo et al., Clark et al., James-Roxby et al.
    - Altera: Gai et al., Hung et al.
    - LEON 3
  - Streaming
    - Ravindran et al., Karanam et al., Bonnot et al.
Software layer

- A thin operating system kernel to schedule pipelined applications
- The scheduler runs on the master PPC
- Tasks on the slave processors are sequenced through interrupts
- Task communication and data movements are performed through a simple DMA engine
- The DMA is managed by the master processor, with a software transfer list
Micro-kernel setup

- Micro-kernel parameters setting
  - The developer determines which elements and features of the virtual platform are used by the target application

- Data structures definition
  - The developer determines input and output data structures for each active slave

- Data flow setting
  - The developer defines the actual data flow of the application
Pipeline management
Micro-kernel primitives

- **VP_send**
  - Moves data blocks from the storage locations in external memory to buffers of any of the slaves

- **VP_receive**
  - Moves data blocks from local buffers of any slaves to the storage locations in external memory

- **VP_receive_and_send**
  - Transfers results from a slave to another slave, either directly or checkpointing the data in external memory

- **Pipeline**
  - Enables pipelining, keeping trace of the status of each worker and synchronizing the data flow through interrupts
Case studies: procedure

- Application profiling
  - Recognizing the critical points of the applications
- Identification of the pipeline stages
  - Partitioning of the application in single tasks/kernels to be assigned to the stages of the pipeline
- Implementation of the pipeline stages
  - Rewriting the application, taking into consideration data dependences, to obtain the desired data flow
- Experiments and validation
  - Evaluation of the modeling of the pipelining
ADPCM - CRC

- Communication of CRC key through the point-to-point connection
- Phases run on independent blocks
- As the size of the input stream increases, the performance increases
Frequency modulation

- Not balanced phases (e.g. equalization is by far the slowest stage)
- Data transfers from the external memory have not a regular pattern
Stages are independent on blocks of 8x8 pixels

First value in each block for entropy coding is calculated from the previous block

We partitioned on blocks of 16x16x3 (768) bytes
## Experimental results

<table>
<thead>
<tr>
<th>Compute Element</th>
<th>ADPCM-CRC</th>
<th>FM</th>
<th>JPEG</th>
</tr>
</thead>
<tbody>
<tr>
<td>MB0</td>
<td>1.3947</td>
<td>0.1929</td>
<td>0.2716</td>
</tr>
<tr>
<td>MB1</td>
<td>0.0524</td>
<td>0.1715</td>
<td>0.2617</td>
</tr>
<tr>
<td>MB2</td>
<td>0.0712</td>
<td>3.2498</td>
<td>0.2629</td>
</tr>
<tr>
<td>MB3</td>
<td>0.0810</td>
<td>-</td>
<td>0.2210</td>
</tr>
<tr>
<td>PPC1</td>
<td>-</td>
<td>0.0005</td>
<td>0.0192</td>
</tr>
<tr>
<td>TOTAL [MB/s]</td>
<td>0.052</td>
<td>0.0005</td>
<td>0.0188</td>
</tr>
</tbody>
</table>
Our solution is a *tradeoff* between:
- The use of a software simulator of the target hardware
  - Cheap and configurable, but slow
- The use of the real target hardware, modified for the testing
  - Fast, but expensive and not flexible

Can be used to validate mapping and partitioning algorithms on the higher layers of a design toolchain

Thanks to reconfigurable logic, the processing elements can be customized and tailored until reaching the required performance
Conclusions

- Presented a heterogeneous multiprocessor FPGA virtual platform for prototyping pipelined applications
- Converted some multimedia applications to the pipelined programming model (ADPCM-CRC, FM, JPEG)
  - Validated on our platform
  - ADPCM-CRC and JPEG have been successfully pipelined, FM has a bottleneck in one of the stages
  - Starting from the initial evaluation, further steps to optimize the applications can be taken
- It is the first step towards a more comprehensive framework for fast prototyping
  - For testing new ideas in the hardware and the software
Thank you for your attention!

Questions?

tumeo@elet.polimi.it

http://trac.elet.polimi.it/cerbero