A software platform to support dynamically reconfigurable Systems-on-Chip under the GNU/Linux operating system

26th July 2005

Alberto Donato
donato@elet.polimi.it

Relatore: Prof. Fabrizio Ferrandi
Correlatore: Ing. Marco Domenico Santambrogio
Index

- Aim of the work

- The reconfigurable platform:
  - Hardware architecture
  - Partial dynamic reconfiguration flows

- Software architecture:
  - The ICAP kernel module
  - The IP-Core Manager

- Tests and results

- Future work
Aim of the Work

FPGAs reconfiguration capabilities allow creation of Systems-on-Chip whose hardware components can be modified, added and removed at runtime.

Provide software support for **dynamic partial reconfiguration** on Systems-on-Chip running the **LINUX operating system**.

Issues:

- Partial reconfiguration process management from the OS
- Addition and removal of hardware reconfigurable components
- Automatic loading and unloading of specific drivers for the IP-Cores upon components configuration/deconfiguration
- Easier programming interface for specific drivers
Hardware Architecture

Architecture derived from Caronte:

- General purpose processor
- Memory (BRAM, DDR)
- ICAP
- Reconfigurable components
Hardware Architecture (2)

Components of the hardware architecture:

- PPC405 processor
- SRAM and SDRAM memory
- Flash memory
- Xilinx ICAP IP-Core

I/O peripherals:

- Xilinx UARTLite interface
- Ethernet controller
- GPIO LEDs
- 7-Segments display
Partial Dynamic Reconfiguration Flows

Reconfiguration of areas of the FPGA fabric without interfering with the rest of the system.

Two different approaches: small-bits and module-based reconfiguration.

Small-bits manipulation:
- change single reconfigurable elements, such as slices
- useful to modify the configuration of hardware components of the system

Module-based reconfiguration:
- addition or removal of system components
- changes in the resources available to the system
Software Side: Previous Work

Self-reconfigurable platform:
- PowerPc/MicroBlaze processor
- ICAP component
- Stand-alone application using the XPART API

Supports small-bits reconfiguration.

Egret architecture:
- Modular hardware/software architecture
- Based on the LINUX OS
- Dynamic configuration of hardware modules and loading of drivers
Software Architecture

Composed of two main elements:

- Driver to support partial reconfiguration
- Manager for the IP-Cores devices

The software architecture provides:

- Access to ICAP component from userspace
- Interface between IP-Cores low-level drivers and kernel
- Access to reconfigurable devices from userspace processes

<table>
<thead>
<tr>
<th>Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux Kernel</td>
</tr>
<tr>
<td>ICAP Driver</td>
</tr>
<tr>
<td>IP-Core Manager</td>
</tr>
<tr>
<td>Low-level Device Drivers</td>
</tr>
<tr>
<td>ICAP</td>
</tr>
<tr>
<td>Reconfigurable Devices</td>
</tr>
</tbody>
</table>
The ICAP Kernel Module

Implements a device driver, adds kernel support for the Xilinx ICAP component.

- Access from userspace via standard *device node* mechanism (i.e. `/dev/icap`)
- Masks hardware details
- Reconfiguration data provided in the form of *partial bitstream files*
Reconfiguration process using the ICAP kernel module:

1. partial bitstream is copied into ICAP module buffer from a userspace process
2. reconfiguration ioctl call is performed from userspace
3. the kernel module sends partial configuration data to the ICAP component

The hardware ICAP component is accessed through the memory mapping mechanism.
The IP-Core Manager

A LINUX kernel module which implements a unified infrastructure for the management of the IP-Cores.

- IP-Cores Plug-and-Play
- Runtime loading of specific IP-Cores drivers
- Management of operations common to all drivers
- Access to reconfigurable components from userspace
- Standardize and simplify writing of specific drivers
The IP-Core Manager acts as a *layer* between the operating system kernel and the low-level device drivers.

The low-level drivers contain:

- system calls implementation
- devices initialization and shutdown functions

The drivers also contain a *stub*:

- provides standard kernel module interface
- provides module initialization and shutdown functions
Registration process of a new IP-Core with the IPCM:

1. interrupt is received from the HW-IPCM
2. read device data (base address, device id, address range)
3. load low-level driver if not already loaded
4. the low-level driver initialization function registers driver data structures
5. data structures for the new device is initialized
## Tests and Results

### Comparison between original *Caronte* architecture and the one supporting *LINUX*

<table>
<thead>
<tr>
<th>Resource</th>
<th>Original Caronte</th>
<th>Caronte LINUX</th>
<th>Total available</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slice Flip Flops</td>
<td>1843</td>
<td>18%</td>
<td>2369</td>
</tr>
<tr>
<td>4-input LUTs</td>
<td>1727</td>
<td>17%</td>
<td>2173</td>
</tr>
<tr>
<td>Occupied Slices</td>
<td>1818</td>
<td>36%</td>
<td>2262</td>
</tr>
<tr>
<td>Bonded IOBs</td>
<td>107</td>
<td>27%</td>
<td>168</td>
</tr>
<tr>
<td>Block RAM</td>
<td>32</td>
<td>72%</td>
<td>32</td>
</tr>
<tr>
<td>DCMS</td>
<td>2</td>
<td>50%</td>
<td>2</td>
</tr>
</tbody>
</table>

### Resources usage in different hardware architectures supporting LINUX

<table>
<thead>
<tr>
<th>Resource</th>
<th>Base Arch.</th>
<th>No Ethernet</th>
<th>No Flash</th>
<th>Total available</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slice Flip Flops</td>
<td>5079</td>
<td>51%</td>
<td>2369</td>
<td>24%</td>
</tr>
<tr>
<td>4-input LUTs</td>
<td>5883</td>
<td>59%</td>
<td>2173</td>
<td>22%</td>
</tr>
<tr>
<td>Occupied Slices</td>
<td>4926</td>
<td>99%</td>
<td>2262</td>
<td>45%</td>
</tr>
<tr>
<td>Bonded IOBs</td>
<td>205</td>
<td>51%</td>
<td>168</td>
<td>42%</td>
</tr>
<tr>
<td>Tbufs</td>
<td>64</td>
<td>2%</td>
<td>64</td>
<td>2%</td>
</tr>
<tr>
<td>Block RAM</td>
<td>36</td>
<td>81%</td>
<td>32</td>
<td>72%</td>
</tr>
<tr>
<td>GCLKs</td>
<td>6</td>
<td>37%</td>
<td>4</td>
<td>25%</td>
</tr>
<tr>
<td>DCMS</td>
<td>2</td>
<td>50%</td>
<td>2</td>
<td>50%</td>
</tr>
</tbody>
</table>
Small-bits manipulation partial reconfiguration tests

<table>
<thead>
<tr>
<th>Label</th>
<th>Reconfig. frames</th>
<th>Bitstream size (Byte)</th>
<th>Configuration time (msec)</th>
<th>Throughput (MByte/sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC 0</td>
<td>33</td>
<td>24179</td>
<td>15.509</td>
<td>1.487</td>
</tr>
<tr>
<td>RC 1</td>
<td>46</td>
<td>40171</td>
<td>25.674</td>
<td>1.492</td>
</tr>
<tr>
<td>RC 2</td>
<td>50</td>
<td>44907</td>
<td>28.628</td>
<td>1.496</td>
</tr>
<tr>
<td>RC 3</td>
<td>60</td>
<td>51347</td>
<td>32.700</td>
<td>1.497</td>
</tr>
<tr>
<td>RC 4</td>
<td>68</td>
<td>53619</td>
<td>34.203</td>
<td>1.495</td>
</tr>
<tr>
<td>RC 5</td>
<td>88</td>
<td>67099</td>
<td>42.814</td>
<td>1.495</td>
</tr>
<tr>
<td>RC 6</td>
<td>104</td>
<td>60567</td>
<td>38.598</td>
<td>1.459</td>
</tr>
<tr>
<td>RC 7</td>
<td>132</td>
<td>100595</td>
<td>64.140</td>
<td>1.496</td>
</tr>
<tr>
<td>RC 8</td>
<td>148</td>
<td>94063</td>
<td>60.045</td>
<td>1.494</td>
</tr>
<tr>
<td>RC 9</td>
<td>182</td>
<td>119319</td>
<td>76.049</td>
<td>1.496</td>
</tr>
</tbody>
</table>

Reconfiguration times have been measured on the *Avnet Virtex-II Pro Development Board*:

- with 66 Mhz processor: throughput ~1.5 Mbyte/s
- with 100 Mhz processor: throughput > 3.0 Mbyte/s
Future Work

Possible extensions of the described software architecture:

- extension of the ICAP driver to allow creation of reconfiguration command from reconfiguration data (without pre-synthesized bitstreams)
- runtime calculation of difference bitstreams
- implementation of the HW-IPC
- extension of the ICAP component to support DMA transfers