FPGAs for flexible interfacing and distributed computational tasks

AO4RTC 2023
Roberto Biasi
Outline

• Very brief company presentation
• FPGA-based flexible sensors and actuators interface platform
• FPGA for distributed computational tasks
• FPGA: lessons learned, pros and cons
Main Business Units

- **ENGINEERING**
  Real-time control systems for telescopes and adaptive optics

- **PROFESSIONAL TIMING**
  Professional timing systems

- **TRAINING & SPORT**
  Athletic performance evaluation systems

- **MEDICAL REHAB**
  Gait analysis for medical rehabilitation and injury prevention
Where we are, what we do

• New premises located in the industrial area of Bolzano-south, South Tyrol
  • 6,000 m² (including MPD)
  • Electronics labs
  • Mechanical workshop
  • Thermal and EMC tests
  • Optical test areas
  • Large clean integration room: 400 m², 20 t overhead crane, large climatic test pit

• The internal capabilities cover the entire process of electronic systems design and manufacturing
  • Hardware design (digital, analog)
  • Firmware (FPGA, microcontrollers)
  • Software
  • Control system design and multiphysics simulation
  • Prototyping
  • Integration of complex mechatronics systems
  • Testing
  • Production
Microgate Engineering

- Microgate (with ADS International, INAF and Politecnico di Milano) has developed the large, contactless, VCM-driven adaptive mirror technology over the past > 25 years
  - Deployed on MMT, LBT, Magellan, VLT
  - In construction: Subaru, ESO-ELT, GMT
Microgate Engineering

- Other fields:
  - Motion control systems
  - Metrology
  - Optical communication: ALASCA
  - AO RTCs: Keck, Padova AO Lab RTC (ANU, SUT)
FPGA background at Microgate

DM control: Typical hard-real-time application

- Very fast control loop cycle, \(~ 80\) kHz
- Very few \(\mu\)s latency
- Large number of on-board devices (up to 36 ch, ADCs, DACs)
- Strong parallelism

Ideal playground for FPGAs

- ‘Modern’ approach, DSP not supported any more
- System on chip, one device to cope with all needs
- Lower power consumption
- Less space

All FPGA-based – on-chip DSP blocks
µXLink, PCIe FPGA board by Microgate

Main motivations to develop a general-purpose PCIe FPGA board

• The **GreenFlash** H2020 EU founded project, led by D.Gratadour, aiming to compare RTC technologies for the ELTs, gave us the opportunity to develop a new cutting-edge FPGA based interface board (**µXLink**)  

• Developing our own FPGA board instead of using a COTS allows us to be **vendor-independent** in terms of drivers and software  

• Since we have all **knowledge in our hands** (hardware and firmware), we can better guarantee **long term support** of our electronics, including proper obsolescence management.

• Optimal selection of **interfaces** during the design phase, so to allow connecting of a large **variety of sensors and mirrors**  

• Optimal usage of all available hardware resources to route the real-time data path so to achieve **minimum latency** and **maximum time determinism**.
Board facts:

- SoC FPGA with ARM-A9 dual core + 1855 FP DSP blocks
- Board size compliant with PCIe standard full height and >= ¾ length
- # Layers: 18 (9 signal, 9 power-ground)
- # Components: 1752
- # Tracks: 7155 (300 LVDS pairs)
- # Vias: 10207
µXLink applications: Interface Module

- Interfaces to a large **variety of sensor and actuators** (mirrors)
- **Flexibility** in adding/changing interfaces by exchanging simple passive front-end boards
- **Rugged system** that can be in telescope enclosure environment, close to the devices
- Flexible and accurate **synchronization** of devices operation
- **Low power consumption**
- **Hardware abstraction** – one single interface and communication protocol to/from the computational units
IM example: Keck RTC architecture
Flexible FPGA-based interface module

- GPUDirect and CPU DMA transfer
  - Direct data transfer from μXLink to the **GPU** (NVIDIA) via **GPUDirect** without interacting with Host CPU
  - Direct data transfer from μXLink to RAM memory over PCIe for CPU based reconstructor without Host CPU interaction

- SW initialization
  - Host software configures the μXLink PCIe engine to perform directly CPU/GPU RAM memory data transfers
  - The computational software acts only on data available in memory. In this way the computational software is totally **abstracted** from the **hardware interfaces**.
Real-time pipeline transfer processed in HW
Interface module SW ecosystem

- ARM Dual-Core CPU in the ARRIA 10 FPGA (SoC)
- Full software stack in our hands
  - Allows full software support to customer
- Optimized Compact Linux OS
  - based on Yocto
  - we provide dedicated drivers
- SSH interface to µXLink board
- Direct command interface to different cameras
- High level software for:
  - Configuration
  - Housekeeping
  - Maintenance
Interface Module and Computational Engine

Real performance facts: ALASCA
- No computation in FPGA
- CPU based CE

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0 (Round trip)</td>
<td>15.0</td>
<td>17.6</td>
<td>21.5</td>
</tr>
<tr>
<td>100x100</td>
<td>18.0</td>
<td>20.6</td>
<td>24.0</td>
</tr>
<tr>
<td>200x200</td>
<td>28.3</td>
<td>30.9</td>
<td>42.8</td>
</tr>
<tr>
<td>400x400</td>
<td>60.5</td>
<td>63.3</td>
<td>68.3</td>
</tr>
<tr>
<td>600x600</td>
<td>112.5</td>
<td>114.5</td>
<td>135.5</td>
</tr>
<tr>
<td>700x700</td>
<td>156.5</td>
<td>159.4</td>
<td>173.0</td>
</tr>
</tbody>
</table>

ALASCA specific case:
- MVM (224x224)
  - $T_{\text{min}} = 31.0 \, \mu s$
  - $T_{\text{mean}} = 33.7 \, \mu s$
  - $T_{\text{max}} = 36.5 \, \mu s$

- Computation on single core: Intel(R) Xeon(R) CPU E5-2667 v4 @ 3.20GHz
- input/output: uint16
- matrix/computation: double

David Jenkins on CaNaPy RTC @ 14:50
**Interface Module and Computational Engine**

Real performance facts: Keck RTC

- No computation in FPGA
- GPU based CE, implementing the COSMIC ecosystem

<table>
<thead>
<tr>
<th>Sensor</th>
<th>Frequency</th>
<th>Roundtrip Time</th>
<th>Camera Readout time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>min</td>
<td>mean</td>
</tr>
<tr>
<td>CCD39</td>
<td>41.35</td>
<td>41.36</td>
<td>41.36</td>
</tr>
<tr>
<td></td>
<td>1031.30</td>
<td>1032.08</td>
<td>1033.33</td>
</tr>
<tr>
<td></td>
<td>2402.23</td>
<td>2405.54</td>
<td>2411.90</td>
</tr>
<tr>
<td>OCAM</td>
<td>49.99</td>
<td>49.99</td>
<td>50.00</td>
</tr>
<tr>
<td></td>
<td>998.40</td>
<td>999.91</td>
<td>1001.26</td>
</tr>
<tr>
<td></td>
<td>1994.43</td>
<td>1999.82</td>
<td>2005.88</td>
</tr>
</tbody>
</table>

a) Image calibration
b) Centroid computation
c) MVM (352x608)
d) DM Control Filter

**OCAM 2kHz (240x240 pixel)**

**CCD39 2.4kHz (80x80 pixel)**

D. Gratadour, J. Bernard, N. Doucet on COSMIC
Computation with FPGA

• Besides the fast (~ 80 kHz) local control, our DMs require some global computational task:
  • Transformation of modal commands from the RTC into zonal commands for the actuators
  • Modal clipping, to avoid force saturation of any actuator:
    • Computation of the force pattern applied by actuators to achieve the position (shape) setpoint
    • Seeking for the largest number of commanded modes (usually all of them…) that can be applied to the mirror without any force saturation
    • Return of the ‘applied setpoint’ to the RTC

• Very tight timing constraints
• Substantially it’s a large MVM, could be implemented directly in the RTC, but execution at DM level is preferred (safety involved, clear interface, avoid additional load to the RTC, …)
Computation with FPGA: ESO ELT-M4 case

- 5316 actuators
- Fast (~80 kHz) and quite complex local control loop realized by 180 FPGA-based control bricks with FP DSP blocks
- Very efficient data interconnection
- Why not implementing the global computation (clipping) in the same FPGAs?
  - Efficient HW exploitation
  - Power efficient
  - Ultra low latency
Computation with FPGA: ESO ELT-M4 case

- Main task: execute **two** FP32 MVMs,  \([6480 \times 5352] \times \{5352\}\) each

- Computation **parallelized** over the 180 FPGAs available on the DM control **bricks**  \([36 \times 5352] \times \{5352 \times 1\}\) each

- The computation is **pipelined** with the modal command data distribution to all **bricks** (3.125 Gbit/s proprietary redundant link), no additional latency introduced by the data transfer

- Data distribution and **arbiter** functions performed by a single **µXLink board**

- **85 µs computational time**, 125 µs to return the applied setpoint to the RTC (ESO req: 150 µs), jitter in sub-µs level
General considerations 1/2

• Pros and cons of in-house approach (no COTS)
  + Master all **hardware and low-level firmware** (drivers)
  + Continuity in **support to customer**
  + Increase **internal know how**, easy migration to next generation, obsolescence management
  - Limited resources do not allow to follow consistently the technological updates. Really so negative? Stability, consolidation of solutions, lifetime of typical AO projects...

• Proven by several examples in the field
General considerations 2/2

• Lessons learned from FPGA in real applications
  • Development time is a real issue; stable once done, but reaching final deployment is not easy and can be very time consuming
  • FPGA use shall be justified by the context – and this is often the case!
    • Need of flexible hardware interface, on- and off-board
    • Very hard-real-time constraints, can’t do by SW
    • Size, power
  • Development tools improvement not as fast as HW growth
  • Can high level programming assure a bright future to FPGAs for computational tasks – HW accelerators?
    • Our experience: still not optimal in HW utilization and performance
    • Not solving the compile-fitting time issue
    • But... strong investments by the main players + research activities (e.g. RisingSTAR project, aiming to a single SW description on heterogeneous platforms CPU/FPGA/GPU)
Wenn eine Idee nicht zuerst absurd erscheint, taugt sie nichts.

A. Einstein

Thank you for your attention!