Enabling Seamless Execution on Hybrid CPU/FPGA Systems: Challenges & Directions

Meena Belwal, Madhura Purnaprajna, Sudarshan TSB
Department of Computer Science and Engineering, Amrita School of Engineering, Bangalore Campus, India
Applications defined for conventional processors cannot be used directly on FPGAs.
Hybrid (CPU+FPGA) Computing Bottleneck

Diversified Programming Languages
(Verilog, OpenMP, OpenAcc, CUDA, VHDL)

Ideally, dynamic automatic code partitioning and application mapping can simplify the process of introducing FPGAs in legacy CPU-only systems.
Contents

1. Design Approaches
2. Programming Models and Hardware Compilation
3. OS and Run-Time Systems
4. Overhead Minimization
5. Future Research Directions
### Design Approaches

<table>
<thead>
<tr>
<th>FPGA-only model</th>
<th>Processor-only model</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="FPGA-only model diagram" /></td>
<td><img src="image2" alt="Processor-only model diagram" /></td>
</tr>
<tr>
<td>Processor + FPGA (offload) model</td>
<td>Processor + FPGA (Unified OS) model</td>
</tr>
<tr>
<td><img src="image3" alt="Processor + FPGA (offload) model diagram" /></td>
<td><img src="image4" alt="Processor + FPGA (Unified OS) model diagram" /></td>
</tr>
</tbody>
</table>

- **Processor-only model**
  - Application
  - OS
  - CPU

- **FPGA-only model**
  - Application
  - OS
  - FPGA

- **Processor + FPGA (offload) model**
  - Application
  - OS
  - CPU
  - FPGA

- **Processor + FPGA (Unified OS) model**
  - Application
  - OS
  - CPU
  - FPGA
<table>
<thead>
<tr>
<th>Design Approaches</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>FPGA-only model</strong></td>
</tr>
<tr>
<td><img src="image1" alt="FPGA-only model" /></td>
</tr>
<tr>
<td><strong>Processor + FPGA (offload) model</strong></td>
</tr>
<tr>
<td><img src="image3" alt="Processor + FPGA (offload) model" /></td>
</tr>
</tbody>
</table>
To tap the full potential of CPU/FPGA hybrid systems a computational model with modern operating services is needed that hides platform specific CPU/FPGA distinction from the programmers.

- CPU, FPGA at par
- Static techniques are application specific that leads to non-portability, and limits the designer’s productivity.

Programming Model | OS and Run-time | Overhead Minimization
Programming Models and Hardware Compilation

Front End Programming

C-based
- Handel-C [1]
- CHiMPS [2]
- LegUp [3]
- SOpenCL [4]

Unified
- OpenCL [5]

- Static Code Compilation
- Source-to-Source Translation
OS and Run-Time Systems

- Hthreads [6]
- ReconOS [7]
- Rainbow OS [8]

- Uniform APIs
- Application profiling
- Multitasking
- Dynamic Frequency Scaling

- Overhead and jitter
- Static configuration of HW threads
- Application specific
Overhead Minimization

Soft processors implemented on FPGA fabric

- Ease FPGA usability (Programmed using HLL)

<table>
<thead>
<tr>
<th>Overlays</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPU Overlay [9]</td>
<td>Floating and vector operations</td>
</tr>
<tr>
<td>VDR Overlay [10]</td>
<td>DFG (Data Flow Graphs) mapped to FPGA</td>
</tr>
<tr>
<td>Other Overlays</td>
<td>IF[11], FSM[12]</td>
</tr>
</tbody>
</table>
## Overhead Minimization

### Acceleration of synthesis and place and route process

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Megablocks[13]</td>
<td>RPU generated statically</td>
</tr>
<tr>
<td>Virtual FPGA[15]</td>
<td>Lean synthesis and P &amp; R algorithms</td>
</tr>
<tr>
<td>HMFlow[16]</td>
<td>Hard macros</td>
</tr>
<tr>
<td>BPR[17]</td>
<td>Functional reuse</td>
</tr>
<tr>
<td>qFlow[18]</td>
<td>Split design (invariant &amp; evolving logic set)</td>
</tr>
</tbody>
</table>
Future Research Directions

Alternatives to Overlays

- Modifying FPGA fabric to map soft processors [19]
- Easing code portability on FPGAs by making the reconfigurable fabric more processor-centric

Accelerating Hardware Compilation

Drawing Parallels from CPU/GPU Hybrid Systems
Future Research Directions

Alternatives to Overlays

- Lazy Man’s logic synthesis (to perform logic synthesis offline) [20]
- The optimal structure for a Boolean function is retrieved from a precomputed library of structures.
- The library is created by an exhaustive synthesis of a large set of benchmarks

Accelerating Hardware Compilation

Drawing Parallels from CPU/GPU Hybrid Systems
Future Research Directions

Alternatives to Overlays

- SnuCL
- FluidCL
- Static Schemes: OpenMP, CUDA, Unified Memory
- APU from AMD

Accelerating Hardware Compilation

Drawing Parallels from CPU/GPU Hybrid Systems
Thank You