OpenMP Device Offloading to FPGA Accelerators
Lukas Sommer, Jens Korinth, Andreas Koch
Embedded Systems and Applications Group / Computer Systems Group
Technische Universität Darmstadt

1. Motivation & Goals

Motivation
- FPGAs increasingly used for implementation of accelerators in HPC systems (e.g., Microsoft Azure)
- Programming heterogeneous systems is non-trivial
- Desirable: Programming with a single, portable code base

Goals
- Implement toolflow to automatically map target regions to FPGA accelerators
- Extend LLVM OpenMP Runtime [2,3] to manage data transfers and FPGA execution

2. OpenMP Device Offloading

```c
#pragma omp target
map(to:x[0:SIZE])
map(tofrom:y[0:SIZE])
{
#pragma omp parallel for[...]
for(i=0; i<SIZE; i++)
{y[i] = a*x[i]+y[i];}
}
```

3. ThreadPoolComposer

<table>
<thead>
<tr>
<th>PE</th>
<th>PE</th>
<th>Infrastructure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host Interface</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Memory Interface</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Custom Clang toolchain compiles binary and extracts kernel code from target region
- TPC-specific binary as entry point for hardware execution, launches accelerator
- Fully automated flow from extracted kernel code to full FPGA-bitstream with host- and memory connectivity using TPC facilities

4. Compilation Flow

- Single-core execution on the Virtex 7 at 250 MHz leaves room for improvement compared to quad-core execution on the x86-CPU at 4.0 GHz (6.7x/3.4x) → Distribute computation across multiple kernels to close the gap
- Including Vivado HLS pipelining pragma results in 2x speedup
- Offloading overhead mainly dependent on size of data transferred to device memory and back.

5. Execution Flow

1. Execution starts on the host
2. Data is transferred to/from accelerator memory before/after PE execution
3. TPC-specific binary is loaded, transfers arguments and launches PE execution
4. Use of two-layered TPC API ensures portability

6. Evaluation

- Fully functional implementation of OpenMP offloading to FPGAs
- Program FPGA-based heterogeneous systems with a single, portable code base
- Future Work: Make use of coarse-grain parallelism by distributing computations across multiple kernel instances (e.g., `target teams distribute`)

7. Conclusion & Outlook

- Program FPGA-based heterogeneous systems with a single, portable code base
- Future Work: Make use of coarse-grain parallelism by distributing computations across multiple kernel instances (e.g., `target teams distribute`)

8. Contact & References

Contact me:
sommer@esa.tu-darmstadt.de
Get open-source release of ThreadPoolComposer:
https://goo.gl/qf6u38

