A Coarse-Grain FPGA Overlay for Executing Data Flow Graphs

Davor Capalija and Tarek Abdelrahman
University of Toronto

CARL 2012
FPGAs as Accelerators

- Emergence of FPGAs as accelerators
  - Gaining popularity in heterogeneous HPC
  - Raw computational power rivals CPUs and GPUs
- FPGA strengths
  - Massive parallelism (1 million LUTs)
  - Customization of computation and interconnect
  - Reconfiguration
- Exponential increase in FPGA resource count will continue to 2030 (FPGA 2012 workshop)
FPGA “Programmability Wall”

• FPGA programming is much harder than software systems programming

• FPGA programming
  • HDL such as Verilog
  • High-level synthesis from C-like languages

• Knowledge of low-level FPGA fabric and tools
  • Timing (Fmax), resources (LUTs), place and route

• FPGA synthesis takes hours, even days

• The iterative nature of this process makes it even harder
Our Goal

• Hide FPGA details (timing, LUTs)
  • Raise the level of abstraction to data flow graphs
• Radically reduce compile time
  • Remove synthesis from the tool chain
• Keep FPGA strengths
  • Massive parallelism
  • Customization of computation and interconnect
  • Reconfiguration
Our Approach: VDR Overlay

• **Virtual** **Dynamically** **Reconfigurable** Overlay

• Pre-synthesized FPGA circuit designed to execute data flow graphs

• It can be **configured** at **run-time** to implement any data flow graph
VDR Overlay Architecture

- Coarse-grain (wide) Functional units

VDR Overlay

FPGA

VDR Block

ADD

MUL
VDR Overlay Architecture

- Coarse-grain (wide) Functional units Switches
- Configurable at run-time

VDR Overlay
VDR Switch
VDR Block
ADD
MUL
FPGA
DFG-to-VDR mapping

Data Flow Graph (DFG)
DFG-to-VDR mapping

Data Flow Graph (DFG)
VDR Overlay Synchronization

- VDR Block
- VDR Switch
- VDR Block

- MUL
- ADD
- MUL
- SHIFT

Data  Valid  Ready
VDR Overlay – Benefits

• Direct DFG execution
  • Bridges the gap between SW and HW

• Rapid compile time (DFG-to-VDR)
  • Seconds

• Rapid reconfiguration time (DFG-to-VDR)
  • Several microseconds

• Can be targeted by a static compiler or a dynamic run-time system
VDR Overlay – Benefits

• Virtualization
  • Can be implemented on any commodity FPGA

• DFG portability
  • Across vendors, device families and sizes

• DFG relocation within VDR
  • Easier dynamic and partial reconfiguration
VDR Overlay – Research Issues

• VDR Block architecture
  • Generic FUs, custom FUs
  • Homogenous, heterogeneous

• Interconnect topology
  • e.g. 2-NN, 4-NN, 8-NN

• DFG-to-VDR mapping algorithm

• High performance

• Low FPGA resource overhead

• Scalability – large overlays
Feasibility Study
- Dynamic Soft Processor Acceleration

```
ADD      R7,R6,R1
ADD      R9,R7,R10
BEQZ     end
ADD      R1,R3,R7
MULT     R11,R12,R13
ADD      R8,R1,R2
SUB      R9,R6,#6
SLT      R8,R4,R7
BEQZ     R8,end
ADD      R7,R6,R1
ADD      R9,R7,R10
```

Commodity FPGA

Soft
Processor

VDR Overlay
**Feasibility Study**  
- Dynamic Soft Processor Acceleration

```
ADD R7,R6,R1
ADD R9,R7,R10
BEQZ end
ADD R1,R3,R7
MULT R11,R12,R13
ADD R8,R1,R2
SUB R9,R8,#8
SLT R8,R4,R7
BEQZ R8,end
ADD R7,R6,R1
ADD R9,R7,R10
```

- Monitor execution to detect a hot section of code
Feasibility Study
- Dynamic Soft Processor Acceleration

- Monitor execution to detect a hot section of code
- Create a DFG and map it to the VDR

Commodity FPGA
Feasibility Study
- Dynamic Soft Processor Acceleration

- Monitor execution to detect a hot section of code
- Create a DFG and map it to the VDR
- Configure the VDR
- Execute the DFG on the VDR
Traces and Data Flow Graphs

Trace = BB1, BB3, BB4

Straightline section of code

ld r2, 0(r1)
mul r3, r3, r2
mul r4, r4, r2
mul r5, r5, r2
beq r2, r0, exit_1
add r6, r3, r4
add r7, r4, r5
add r8, r6, r7
addi r9, r9, 4
st r8, 0(r9)
beq r9, r15, exit_2
Acceleration Model

Instruction-level parallelism

Soft processor

VDR Overlay
Acceleration Model

Instruction-level parallelism

Pipeline parallelism

VDR Overlay

Soft processor
Acceleration Model

DFG-level parallelism (replication)

Soft processor

VDR Overlay
Feasibility Study

- A system around the Nios II soft processor
  - DE3 board with Altera Stratix III
- RGB2YIQ benchmark from the EEMBC suite
  - A single trace (single DFG)
- A VDR overlay
  - Application-tailored FUs and interconnect topology
Feasibility Study

• Measured
  • Overlay overhead
  • Benchmark speedup

• Compared
  • Nios II
  • VDR overlay
  • Hardwired overlay

• Hardwired overlay
  • Same units as the overlay
  • Not configurable - switches removed
# VDR Overlay Overhead

<table>
<thead>
<tr>
<th>Version name</th>
<th>Fmax</th>
<th>ALUTs</th>
<th>ALUTs relative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nios II</td>
<td>290</td>
<td>1483</td>
<td>1</td>
</tr>
<tr>
<td>VDR</td>
<td>172</td>
<td>4753</td>
<td>3.2</td>
</tr>
<tr>
<td>Hardwired</td>
<td>286</td>
<td>2978</td>
<td>2</td>
</tr>
</tbody>
</table>

- VDR overlay resource breakdown
  - 60% units and 40% switches
## Benchmark Acceleration

<table>
<thead>
<tr>
<th>Version name</th>
<th>DFG replicas</th>
<th>Ports/Mem (total ports)</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nios II</td>
<td>--</td>
<td>--</td>
<td>1</td>
</tr>
<tr>
<td>VDR-1</td>
<td>1</td>
<td>1 (1)</td>
<td>3.31</td>
</tr>
<tr>
<td>VDR-2</td>
<td>1</td>
<td>2 (2)</td>
<td>4.85</td>
</tr>
<tr>
<td>VDR-2 x2</td>
<td>2</td>
<td>2 (4)</td>
<td>8.82</td>
</tr>
<tr>
<td>HDW-1</td>
<td>1</td>
<td>1 (1)</td>
<td>5.87</td>
</tr>
<tr>
<td>HDW-2</td>
<td>1</td>
<td>2 (2)</td>
<td>8.87</td>
</tr>
</tbody>
</table>
# Overlay Execution Breakdown

- For a 640x480 frame: ~11 M cycles

<table>
<thead>
<tr>
<th>Execution Component</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trace detection</td>
<td>0.06</td>
</tr>
<tr>
<td>Trace-to-VDR mapping</td>
<td>9.19</td>
</tr>
<tr>
<td>Overlay reconfiguration</td>
<td>0.01</td>
</tr>
<tr>
<td>Register initialization</td>
<td>0.01</td>
</tr>
<tr>
<td>Overlay execution</td>
<td>90.73</td>
</tr>
</tbody>
</table>
Current Work

• Regular generic overlay
  • Generic integer FUs
  • Topology - 4-NN with pass-through links

• High-throughput overlay
  • Scalability - from 2x2 up to 32x32
  • High Fmax – 400+ MHz

• Flexible mapping algorithm
  • Ability to map any trace/DFG
  • Place and route algorithm
Next Steps

• Memory interfaces
  • Streaming, FIFOs, caches, scratch-pad memories

• Application customized FUs
  • Integer, fixed-point, floating-point
  • Specialized FUs

• Application customized topology
  • e.g. 4-NN with hops, 6-NN
Thank You