Reconfigurable Cell Array as an Enabler for Future Processing

Chenxin Zhang
Department of Electrical and Information Technology
Lund University, Sweden

Outline
- Motivation and related work
- Coarse-grained reconfigurable cell array
  - Processing cell
  - Memory cell
  - Network router cell
  - System reconfiguration
- Software development
  - Application mapping
  - Design exploration
- Case studies
  - Multi-standard OFDM coarse time synchronization

Reconfigurable computing
- Updates on the data path in addition to the control flow.
- Combined flexibility with high performance at a feasible hardware cost.
- Software-centric programming approach.
- Coarse-grained granularity – trade-off between efficiency, flexibility, and programmability.
- Dynamic reconfigurability.

Motivation
- Hardware sharing
  - Accelerators: poor hardware reusability
  - Reconfigurable architecture
    + Multi-task
    + Multi-standard
    + Multi-algorithm
    - Control overhead, e.g. area, power.
Related work

- ALU clusters: MathStar FPOA, RICA...
  - Instruction level, data level parallelism
  - SIMD or VLIW

- Processor array: RAW, WPPA, REMARC...
  - Instruction level, data level, and task level parallelism
  - MIMD

- Hybrid structure: ADRES, PACT XPP...
  - Instruction level, data level, and task level parallelism
  - SIMD or VLIW and MIMD
  - Combined complexity?

System infrastructure

- An array of resource cells.
- Heterogeneous cell array:
  - Processing cell
  - Memory cell
  - Accelerator
    (e.g. no configuration)
- Hierarchical cell array.

Resource cell

- Dedicated local interconnections:
  - High data throughput
- Hierarchical global routing network:
  - Flexible global data transmission
  - External data access
  - Global cell (re)configuration
- Data driven synchronization
- Single-Cycle-Per-Hop latency
- AMBA 4 AXI4-stream protocol
- GALS network data transmission

Processing cell

- Processing core
  - ALU, DSP, SIMD, VLIW, CORDIC...
  - Implicit load-store operations in all instructions.
  - Run-time control and conditional reconfiguration.
  - In-cell NoC supervision and reconfiguration.
- Processing shell
  - Network adapter
Example 1: Generic signal processing cell

- 4 pipeline stages.
- Hybrid Load-Store & Memory-Memory architecture.
- Compact program size (memory references).
- With external memory cells:
  - Complex addressing modes, e.g. memory indirect, auto-increment.
  - Flexible usage: program/data memory, processor stack, (cache).
- Single-cycle delayed branch.
- Zero-delay conditional inner loop control.

Example 2: Dataflow processing cell (I)

- SIMD/VLIW-like operation:
  - 2/4-way 16/8-bit independent data processing
  - Multi-level data processing (implicit prolog & epilog processing)
- Dual-operand instruction set:
  - Dual-OpCode & Dual-Operand: e.g. ADDSUB R[d1], R[d2], R[s1], R[s2]
  - Vector operation option: e.g. complex number arithmetic
- Dynamic data path reconfiguration
- Conditional instruction executions

Dataflow processing cell: ALU

- Addition/Subtraction:
  - 16-bit real
  - 8-bit complex
- Multiplication (4 LSBs):
  - 8-bit real
  - 4-bit complex
Dataflow processing cell:
Dynamic data path reconfiguration

Dataflow processing cell:
Run-time data arrangement (I)

Complex number multiplication vs. Real number multiplication
- \( MUL \ R3, \ R1, \ R2 \); \( R3 = R1 \times R2 \) where \( \{ab\} \) is stored in \( R1 \) and \( \{cd\} \) is stored in \( R2 \).

Memory cell (I)
- Cell structure:
  - One or more memory banks
  - Memory descriptors (DSC):
    - Max. 4 DSCs (index)
    - 3 x 32 bits (package)
- Operation controller
- Operation mode:
  - FIFO (Coefficient ROM)
  - RAM (ROM)
- Run-time concatenation of multiple memory cells in FIFO mode.
Memory cell (II)
Memory descriptor

• Blocking/Non-blocking execution of memory descriptors.

• Micro-block function enables memory access with finer wordlength.

<table>
<thead>
<tr>
<th>Package</th>
<th>Field</th>
<th>Bits</th>
<th>Length</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>6-1</td>
<td>6</td>
<td>1</td>
<td>100 TX destination ID, only used when output through GSO</td>
</tr>
<tr>
<td>block_size</td>
<td>3-1</td>
<td>3</td>
<td>1</td>
<td>Size of the micro-block under manipulation. This option is used to determine the number of data manipulations in each memory read/write operation. Options of block size are 1, 2, 4, 8, 16, and 32.</td>
</tr>
<tr>
<td>blk_mode</td>
<td>8-27</td>
<td>5</td>
<td>1</td>
<td>Distance to the next micro-block (incremental step size). It means continuous reading.</td>
</tr>
<tr>
<td>blk_wptr</td>
<td>28-33</td>
<td>5</td>
<td>1</td>
<td>Current micro-block read pointer.</td>
</tr>
<tr>
<td>blk_wptr</td>
<td>34-39</td>
<td>5</td>
<td>1</td>
<td>Current micro-block write pointer.</td>
</tr>
<tr>
<td>blk_mode</td>
<td>40-47</td>
<td>8</td>
<td>1</td>
<td>Micro-block data mask. It enables data buffering in the corresponding data position.</td>
</tr>
<tr>
<td>blk_band</td>
<td>48-55</td>
<td>8</td>
<td>1</td>
<td>Sign extend the masked data before sending out the ID part.</td>
</tr>
</tbody>
</table>

Memory cell (III)
Micro-block function

<table>
<thead>
<tr>
<th>31</th>
<th>27</th>
<th>23</th>
<th>19</th>
<th>16</th>
<th>11</th>
<th>7</th>
<th>3</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sign</td>
<td>Inphase</td>
<td>Sign</td>
<td>Quadrature</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>12 bits -&gt; 4 bits</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>PCI -&gt; MC0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>c</td>
<td>Shift by 0 &amp; mask</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>d</td>
<td>Shift by 4 &amp; mask</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Address X</td>
<td>40(I)</td>
<td>40(Q)</td>
<td>30(Q)</td>
<td>30(I)</td>
<td>21(I)</td>
<td>21(Q)</td>
<td>10(I)</td>
<td>10(Q)</td>
</tr>
</tbody>
</table>

Network router cell (I)
Cell structure:

• Decision unit
• Routing structure:
  • Parallel network
  • MUX-DEMUX switch
• Output packet queue (FIFOs)
Network router cell (II)  
Decision unit

- Static routing table
- Managing data transactions:
  - Check in
  - Packet arbitration (MUX-DEMUX switch)
    - Fixed
    - Round-robin
  - Data broadcast
  - Configure routing path

Action list with logged transactions

Action list with candidate transactions

(Parallel network)

Action list with candidate transaction

(MUX-DEMUX switch)

Static & Dynamic configuration (I)

- Master
- Conf. Ctrl
- MPMC
- Stream Ctrl
- Memory

Static & Dynamic configuration (II)

Mapping flexibility

- FIR filter
  - Processing cell: MAC
  - Memory cell: Input data FIFO, coefficient ROM

- Time-multiplexed structure for area driven application.
- Unfolding (parallelize) to improve processing throughput.
- High-precision computations.
Software development

- Automated application mapping.
  - CAL dataflow language
  - Parallelism exploration
  - Within the HiPEC project together with CS dept.

- Design exploration with SCENIC framework.
  - SystemC-based virtual platform
  - Explore system behavior and evaluate performance

Application mapping

Application mapping from CAL

SCENIC – Design Exploration (I)
Create and Configure
SCENIC – Design Exploration (II)
Control – Map Applications

Design specification (XML)
Application mapping (XML)
Cell configuration (ASM)

SCENIC – Design Exploration (III)
Control and Observe

Case study:
Multi-standard OFDM synchronization

- Multiple wireless radio standards
- Concurrent data stream processing
- Coarse Time Synchronization
- Carrier Frequency Offset (CFO) estimation

PC / Processed samples: 1246 samples
PC / Executed instructions: 100996 cc
PC / Stall cycles: 4 cc
Network / Packets sent: 1264
Network / Utilization: 1.2 %
SMC / out stream: 1288 words
SMC / in stream: 1246 words
Speedup over CPU: 7 times

TCP/IP

Correlation FIFO
(Size = M)
Moving sum FIFO
(Size = L)

CFO
Peak detection

Department of Electrical and Information Technology, Lund University
Implementation results (I)

- 65 nm low-power regular VT CMOS:
  - Area: 0.479 mm²
  - Clock frequency: 534 MHz

- Adaptive word length scheduling.
- Adoption of different algorithms, e.g. Novel sign-bit OFDM acquisition.

<table>
<thead>
<tr>
<th>Concurrency</th>
<th>Standard</th>
<th>Quantization accuracy</th>
<th>Memory utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single-Stream</td>
<td>802.11n</td>
<td>4 bits</td>
<td>84.48%</td>
</tr>
<tr>
<td></td>
<td>LTE</td>
<td>4 bits</td>
<td>65.18%</td>
</tr>
<tr>
<td></td>
<td>DVB-H 2K</td>
<td>4 bits</td>
<td>85.71%</td>
</tr>
<tr>
<td></td>
<td>DVB-H 4K</td>
<td>2 bits</td>
<td>85.71%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Signs bit</td>
<td>85.71%</td>
</tr>
</tbody>
</table>

| Dual-Stream | 802.11n & 802.11n | 4 or 2 bits | 16.96% | 4 or 2 bits | 45.09% |
|            | 802.11n & LTE    | 2 bits      | 65.63% | 2 bits      | 73.21% |
|            | DVB-H 2K        | 2 bits      | 93.75% | 2 bits      | 93.75% |

Implementation results (II)

- Compare with ASIC solution, ~4 times more area cost.
- Case study does not explore the potential usage.
  - Simple algorithms
  - No task-level hardware sharing
- Currently looking at channel estimation & MIMO detection for LTE-A.
  - Task-level hardware sharing
  - Cooperation with the DARE project

Take a step forward…

- Hardware track @ EIT:
  - ETIN01 - Digital IC-project & Verification (HT2 ~ VT2)
  - Master thesis, e.g. video, baseband processing.

- Software track @ CS:
  - EDA180 - Compiler Construction (VT1, VT2)
    [http://cs.lth.se/eda180](http://cs.lth.se/eda180)
  - EDA230 - Optimising Compilers (HT1)
    [http://cs.lth.se/eda230](http://cs.lth.se/eda230)