Reconfigurable Cell Array for Real-time Multi-task Applications

Chenxin Zhang
October 11, 2010

Outline

- Introduction
- Motivation and related work
- Framework design and development
  - Processing cell
  - Memory cell
  - Network router cell
- Case studies
  - Reconfigurable radix-2^2 FFT processor
  - Multi-standard OFDM coarse time synchronization

Reconfigurable computing

- Reconfigurable computing is the ability to make substantial changes to the data path itself in addition to the control flow.
- The goal with reconfigurable computing is to combine a high degree of flexibility with high performance at a feasible hardware cost.
- Reconfigurable computing provides a more software-centric programming approach, which enhances productivity by simplifying system integration and verification.
- Granularity refers to the size of the basic reconfigurable elements. A coarse-grained architecture should be a trade-off between efficiency, flexibility, and programmability.
- We focus on designing coarse-grained run-time reconfigurable architectures.

Motivation

- Standard processor (GPP, DSP…)
  - Flexible, short design time
  - Lack of computational capacity
- Specialized hardware (ASIC)
  - Real-time performance, small size, low power
  - Less flexible, manufacturing defects
  - High non-recurring engineering (NRE) cost
- Fine-grained reconfigurable architecture (FPGA)
  - High calculation capacity, flexible
  - Routing overhead, high power consumption
  - Hardware oriented design approach
- Coarse-grained reconfigurable architecture (CGRA)
  - High calculation capacity, flexible
  - Relevantly fast development (software oriented), tolerance to manufacturing defects
  - Sacrificed area & energy efficiency in comparison to ASIC solution
  - Sacrificed mapping flexibility in comparison to FPGA solution
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: PACT XPP...
  - Instruction level, data level, and task level parallelism
  - SIMD or VLIW and MIMD
  - Combined complexity?

Coarse-grained reconfigurable architecture

- Decoupled processing and memory cells.
- Dedicated local connections – high data throughput.
- Hierarchical global routing network – communication flexibility.
- Heterogeneous cell array.

Communication fabric

- 2D-mesh local communication network.
- Tree-structured global communication network.
- Synchronous network data transmission.
- Single-Cycle-Per-Hop communication latency.
- Data driven synchronization (blocking operation) and flow control for all data transfers to simplify algorithm mapping.

Processing cell (I)

- Examples: Simple ALU, DSP, SIMD/VLIW processor, CORDIC...
- Generic processing cell:
  - Customized RISCs with 4 pipeline stages.
  - Memory accesses are reached by direct I/O port register addressing.
  - Hybrid model of Load-Store and Memory-Memory architecture.
  - Implicit load-store operations in all instructions.
  - Compact program size due to the reduced instruction count for memory references.
Processing cell (II)

- Complex addressing modes, e.g., memory indirect, auto-increment.
- Run-time control and conditional reconfiguration.
- In-cell NoC supervision and reconfiguration.
- Conditional instruction executions.
- Single-cycle delayed branch.
- Zero-delay conditional inner loop control.

\[ P_3 = f(P_1, P_2) \]

<table>
<thead>
<tr>
<th>Item</th>
<th>Option</th>
<th>Configurability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instruction download</td>
<td>Full, partial</td>
<td>Run-time</td>
</tr>
<tr>
<td>Program counter value update</td>
<td>Instruction address</td>
<td></td>
</tr>
<tr>
<td>Operation control and debug</td>
<td>Start, stop, reset, single step</td>
<td></td>
</tr>
<tr>
<td>Running status tracing</td>
<td>---</td>
<td></td>
</tr>
</tbody>
</table>

Memory cell

- Contains one or more memory banks.
- Memory descriptors contain configurations.
- Memory descriptors are run-time reconfigurable.
- A memory bank is configured to act either as FIFO or RAM (ROM).
- Blocking/Non-blocking execution of memory descriptors.
- Micro-block function enables memory access with finer wordlength.
- Run-time concatenation of multiple memory cells.

<table>
<thead>
<tr>
<th>Interaction between PCs and MCs</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCs are shared as global resources among network cells.</td>
</tr>
<tr>
<td>Decoupled data processing and memory addressing.</td>
</tr>
<tr>
<td>- In-line memory addressing in PC.</td>
</tr>
<tr>
<td>- Autonomous memory (FIFO) addressing in MC.</td>
</tr>
<tr>
<td>- Memory indirect addressing with multiple MCs.</td>
</tr>
<tr>
<td>MC is used as communication buffers between PCs.</td>
</tr>
<tr>
<td>- Burst data transmission.</td>
</tr>
<tr>
<td>- Data reordering.</td>
</tr>
<tr>
<td>- Data duplication.</td>
</tr>
</tbody>
</table>

Memory descriptor

- FIFO & sequential ROM

<table>
<thead>
<tr>
<th>Field</th>
<th>Bits</th>
<th>Length</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dtype</td>
<td>63-62</td>
<td>2</td>
<td>Operation mode selection: FIFO mode = 0; Sequential ROM mode = 2.</td>
</tr>
<tr>
<td>rd ck</td>
<td>61</td>
<td>1</td>
<td>FIFO/Sequential ROM reading status, 1: read possible. FIF0 mode = 0.</td>
</tr>
<tr>
<td>wr_ek</td>
<td>60</td>
<td>1</td>
<td>FIFO writing status, 1: write possible. This field is reserved in sequential ROM mode operation.</td>
</tr>
<tr>
<td>sec</td>
<td>59-56</td>
<td>4</td>
<td>Data source port ID, 0-7: LIO; 15: GIO. This field is reserved in sequential ROM mode operation.</td>
</tr>
<tr>
<td>dst</td>
<td>55-52</td>
<td>4</td>
<td>Data destination port ID, 0-7: LIO; 15: GIO.</td>
</tr>
<tr>
<td>id</td>
<td>51-42</td>
<td>10</td>
<td>GIO TX destination ID, only used when output through GIO.</td>
</tr>
<tr>
<td>base</td>
<td>41-32</td>
<td>10</td>
<td>Starting address in memory array.</td>
</tr>
<tr>
<td>high</td>
<td>31-22</td>
<td>10</td>
<td>Ending address in memory array.</td>
</tr>
<tr>
<td>rptr</td>
<td>21-12</td>
<td>10</td>
<td>Current FIFO reading pointer.</td>
</tr>
<tr>
<td>wptr</td>
<td>11-2</td>
<td>10</td>
<td>Current FIFO Sequential ROM writing pointer. This field is reserved in sequential ROM mode operation.</td>
</tr>
<tr>
<td>io_bank reset</td>
<td>1</td>
<td>1</td>
<td>Active high reset for memory I/O bank, hardware releases reset automatically.</td>
</tr>
<tr>
<td>---</td>
<td>0</td>
<td>1</td>
<td>Reserved.</td>
</tr>
</tbody>
</table>
Router cell

- Consists of a decision unit, a routing structure and an output packet queue (FIFOs).
- Static routing table
- Routing structure: parallel network or MUX-DEMUX switch.
- Fixed or Round-robin packet arbitration.
- Data broadcast.

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.
- Allocate more resources for high-precision computations.

Case study – Radix-2² FFT

- Radix-2² structure

- Basic radix-2² FFT building block in cell array

Case study – 2048-point pipeline FFT

Concatenated memory cells operate as one larger SDF buffer to reduce high storage capacity requirement in each single unit.
Program instruction size summary for the processor cells

<table>
<thead>
<tr>
<th>Processor cell</th>
<th>PC (0, 0)</th>
<th>PC (1, 1)</th>
<th>PC (2, 1)</th>
<th>PC (3, 0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Usage</td>
<td>BTI-I</td>
<td>Trivial mau.</td>
<td>BTI-II</td>
<td>Coef. gen.</td>
</tr>
<tr>
<td>Instruction amount</td>
<td>53</td>
<td>34</td>
<td>44</td>
<td>20</td>
</tr>
<tr>
<td>Instruction code size [bytes]</td>
<td>212</td>
<td>136</td>
<td>176</td>
<td>80</td>
</tr>
<tr>
<td>Reconfiguration inst. amount</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>Reconfiguration code size [bytes]</td>
<td>16</td>
<td>12</td>
<td>16</td>
<td>12</td>
</tr>
</tbody>
</table>

- Program routines are designed to emphasize the functional flexibilities.
- Changing transform length requires at most 4 instructions for each PC.

FFT benchmark comparison

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Area* [mm²]</th>
<th>f_{max} [MHz]</th>
<th>FFT size [point]</th>
<th>Execution time [cc]</th>
<th>Code size [byte]</th>
<th>Reconfiguration code size [byte]</th>
</tr>
</thead>
<tbody>
<tr>
<td>CGRA</td>
<td>1.8675 (est.)</td>
<td>325</td>
<td>256 1024</td>
<td>4,290 20,212</td>
<td>604</td>
<td>56</td>
</tr>
<tr>
<td>Texas TMS-320VC5502</td>
<td>-</td>
<td>300</td>
<td>256 1024</td>
<td>5,389 25,921</td>
<td>462</td>
<td>462 (code reload)</td>
</tr>
<tr>
<td>ARM926EJ-S</td>
<td>2.78</td>
<td>276</td>
<td>256 1024</td>
<td>13,194 66,196</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

* 0.13 μm technology

- 8 times outperform the conventional DSP solution in reconfiguration code size.
- The execution clock cycles are reduced by at least 3 times compared to an ARM implementation, and ~20% of the clock reduction is gained over the DSP solution.
- Chip size is ~30% smaller than the ARM926E processor.
Case study – Multi-standard OFDM time synchronization

- Multi-standard support
  - WLAN 802.11a/n
  - LTE
  - DVB-H (2K mode)
- Concurrent data processing for any combination of the three supported standards, e.g. WLAN/WLAN, LTE/DVB-H.
- Scenario-aware adaptive resource allocation to distribute computational power and memory space among concurrent data streams.
- This work has focus on:
  - Coarse time synchronization for OFDM-based systems.
    - To find the start of each OFDM symbol.
    - Rough estimation of the carrier frequency offset (CFO).
  - Support for two concurrent data streams.
- Tape-out in June in Infineon 65nm

Design constraints

- Clock frequency: ~ 300MHz
- Max. input data sampling rate: 40 MHz
  => 8 clock cycles for data processing in each PC @ 320MHz
- Concurrent support for 2 data streams
  => 4 clock cycles in each PC for each stream @ 320MHz

OFDM time synchronization

- Correlation
  - Logic shift
  - Multiplication
  - Sub-Accumulate
  - Memory addr.
- Peak detection
  - ABS
  - Conditional ops
- CFO
  - CORDIC

Multi-standard OFDM time synchronization (2)

- Processing cells
  - PC0 @ (0,0):
    - Input data scaling
    - Correlation
    - Data sample indexing
  - PC1 @ (1, 1):
    - Peak detection
  - PC0 & PC1:
    - CFO estimation using iterative CORDIC
- Memory cells
  - Input sample delay FIFO
  - Delay FIFO for moving sum in correlation
  - Communication buffers between two PCs
Processing cell – Special features

- SIMD/VLIW-like operation with run-time data path reconfiguration.
- 2/4-way 16/8-bit independent data processing.
- Three-stage data processing: Pre-, Target-, and Post-processing.

![Diagram of processing cell](image)

Run-time data rearrangement

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

Run-time resource management (I)

- Scenario 1: Single data stream

Run-time resource management (II)

- Scenario 1: Single data stream
  - Scenario 2: Concurrent multi-standard data streams
    - Computation resources: time-multiplexed.
    - Data memories: sacrificed data precision.
Run-time resource management (III)

- Scenario 1: Single data stream
  - Computation resources: time-multiplexed.
  - Data memories: sacrificed data precision.

- Scenario 2: Concurrent multi-standard data streams

Configuration generator

Synthesis results

- Timing:
  - Typical case: 534 MHz
  - Worst case: 262 MHz
- Area: 0.479 mm²
- ~3x larger than function identical ASIC solution
- High system flexibility:
  - Different algorithms
  - Future standards
  - Multi-task

Take a step forward...

- Hardware track @ EIT:
  - ETIN01 - Digital IC-project & Verification (HT2 ~ VT2) [http://www.eit.lth.se/index.php?id=409&L=1]
  - Master thesis, e.g. video, baseband processing.

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