Software Development Environment for Reconfigurable Communications Architecture

Vladimir Ivanov
Radio Communications Lab/Corporate Technology Group

Contributor: Vicki Tsai
Radio Communications Lab/Corporate Technology Group

Reconfigurable Computing Tutorial
International Symposium on System-on-Chip Conference
Tampere, Finland
Intel Corporation
18 November 2003

Outline

• RCA Review
  • What are the specific architectural features which impact software development tools?
  • Programming flow
    • How do specific architectural features impact software development process?
• Software Development Environment
  • Goals and Challenges
  • Process specifics
  • System-level issues
• Development Environment Concept
  • Algorithm and compiler point of view
RCA Review

What are the specific architectural features which impact software development tools?

- Scalable mesh interconnect of heterogeneous processing elements (PEs)
- Interconnect with Nearest Neighbour Mesh
- Clock frequency dependent on load and process
RCA Review

Ubiquitous wireless communication across multiple protocols

A scalable mesh interconnect of heterogeneous processing elements (PEs):
- Configurable basebands for multiple (concurrent) PHY/MAC operation
- Power and size conserving when compared to “multiple” dedicated cores or “traditional” SDR (S/W defined radio) approaches
- Tools for simple programming and portability to different arrays of elements

Programming Flow

How do specific architectural features impact the software development process?
1. Divide the protocol into modes

Preamble Detect:

Diversity Selection:

Steady-State Data:

Each mode refers to a different, non-overlapping period in time

2. Partitioning

Specify functions for each mode

Note: This description is function based and not hardware based.
3. Communication
Establish communication structure among functions

Preamble Detect:
- AFE1 (ant. 1)
- AGC
- Dec. Filter
- Preamble Det.

Diversity Selection:
- AFE1 (ant. 1)
- AGC
- Dec. Filter
- SNR Calc.

AFE1 (ant. 2)
- AGC
- Dec. Filter
- SNR Calc.

Steady-State Data:
- AFE1 (ant. 1)
- Dec. Filter
- AFC
- Pred IQ Imb. Cor.

AFE1 (ant. 2)
- Dec. Filter
- AFC
- Pred IQ Imb. Cor.

AFE1 (ant. 1)
- Dec. Filter
- AFC
- Pred IQ Imb. Cor.

AFE1 (ant. 2)
- Dec. Filter
- AFC
- Pred IQ Imb. Cor.

4. Aggregation
Determine onto which PE types the functions could be mapped
5. Check if resources available for the current hardware layout

Preamble Detect:
- AFE1 (ant. 1) → AGC → Dec. Filter → Preamble Det.

Diversity Selection:
- AFE1 (ant. 1) → AGC → Dec. Filter → SNR Calc.
- AFE1 (ant. 2) → AGC → Dec. Filter → SNR Calc.

Steady State Data:

Resource Usage (%):
- PE typeA
- PE typeB
- PE typeC
- PE typeD
- PE typeE

HW topology

6. Mapping
Place functions onto specific PEs

Preamble Detect:
- AFE1 (ant. 1) → AGC → Dec. Filter → Preamble Det.

Diversity Selection:
- AFE1 (ant. 1) → AGC → Dec. Filter → SNR Calc.
- AFE1 (ant. 2) → AGC → Dec. Filter → SNR Calc.

Steady State Data:
- Host IQ → Descram. → Demux → I/Q Demux → Adap. IQ Inb. Corr. → FEC → 64-Pt FFT
7. Generate “code” for this mapping

- Preamble Detect:
  - AFE1 (ant. 1)
  - AGC
  - Dec. Filter
  - Preamble Det.

- Diversity Selection:
  - AFE1 (ant. 1)
  - AGC
  - Dec. Filter
  - SNR Calc.

- Steady State Data:
  - AFE1 (ant. 2)
  - AGC
  - Dec. Filter
  - SNR Calc.
  - Diversity Sel.
  - AFE1 (ant. 1)
  - Steady-State Data:
    - Dec. Filter
    - AFC
    - Guard Int. Removal
    - FEQ
    - 64-Pt FFT
    - QAM Demap
    - Viterbi
    - Descram.

8. Check if desired performance has been reached

- If desired performance has been met, output the binary images.
- Otherwise, use the results to adjust the mapping and go to Step 2 or 4 or 6.
**Programming Flow Summary**

1. Divide the protocol into modes
2. Specify functions for each mode
3. Establish communication structure among functions
4. Determine onto what PE types the functions could be mapped
5. Check if we have the resources in the hardware
6. Place functions onto specific PEs
7. Generate “code” for this mapping
   - If code cannot be generated because the PE cannot fit the assigned functions, try a different mapping
8. Check if desired performance has been reached
   - If not, try a different mapping
   - Otherwise, output the generated code from Step 6

**Software Development Environment**

- Goals and Challenges
- Process specifics
- System-level issues
Communications Technology Lab

Tools Goals

- Primary goal is to assure development of effective code for RCA
  - Developed code should effectively use all RCA capabilities
  - Implemented protocols should meet users requirements
- Abstract code development from hardware
  - If the number of total PEs change or the number of PEs of a certain type change, the algorithm does not need to be altered
- Give reasonable programming abstraction level for the programmer
- Provide effective environment for development, debugging and testing of software

Tools Challenges

- Reasonable balance for abstracting software development from hardware
- Classical challenges for parallel architecture
  - Decomposition of program into parallel processes
  - Effective mapping of processes to PEs
  - Effective communication among processes
  - Synchronization among processes
- Protocol concurrency implies dynamic RCA resource distribution among protocols
- Heterogeneity of PEs mesh
- Variety of Processing Elements (PEs)
  - PEs may not be processor-based
  - Methods to program PEs differ greatly
- Guaranteed protocol performance
- Effective data visualization from multiple PEs
- High performance simulation of RCA
**Input Example**

```
myFn(int16 in0[]),
    int16 out0[]){
    int16 i;
    for (i = 0; i < IN1LEN; i++) {
        x = in0[i] * in0[i];
        send_output(0, x);
    }
}
```

**System Simulator**

- Cycle accurate simulation
- High performance
- Allow to evaluate latency and computational overhead
- Possibility to connect two instances of the System Simulator to each other
- Provide debugging facilities
System Simulator

- SysSim contains Simulator Core (SC) and Individual Simulators (IS)
- Two abstraction layers for IS representation
  - High level object
  - Scheduled Object
- Object design principle: If being in state S1 and got an input signal in than after delay D change the state to S2 and produce an output signal Out

Simulation Performance

- Comparing SystemC core and SysSim core
- SC METHOD process was used for SystemC
- Simulated object is N instances of D flip-flop objects
- Simulation on Intel 2.4 GHz Pentium 4
- 4x4 Mesh (~1000 objects), 400 MHz
- 1 sec simulation takes ~100 hours for SystemC Core and ~13 hours for SySim Core
Development Environment Concept

Algorithm and Compiler point of view

Tools Development Concepts

- Naive Phase:
  - Manual program partitioning
  - Manual code optimization
  - Independent compiler tools
  - Static hardware and software

- Mature Phase:
  - Automatic program partitioning
  - Automatic code optimization
  - Common compiler tools
  - Static hardware and software

- Advanced Phase:
  - Macro architecture description tools
  - Automatic generation of micro architecture description
  - Automatic software tools generation
  - Protocol partitioning for joint hardware-software optimization
Tools Development Naive Phase

- Enhanced Traditional Model
  - Networking (communication architecture)
  - Mapping (distributable compilation)
  - Traditional tool-suite for RCA
    - Complete development tool-suite
    - Integration of tools for sequential programming
  - Solution constraints
    - Aided mapping (user-defined mapping of process to PE)

Enhanced Traditional Model

- C source code for PE
- Assembly code for FMCA
- Specialized code for VMCA

- C Compiler
- Assembly
- C source code for PE
- Linker
- Executable module
- RCA Simulator
- Debugger

- RCA Linker
- Loadable image
- RCA Simulator

- Make directives
  - Description Translator
  - Tiny Mapper

- Link directives

Communications Technology Lab
Tools Development Mature Phase

- True distributable compilation
  - Automated mapping
- Global optimization
  - Intermodule optimization
  - Optimization on heterogeneous environment
- Enhanced development tools
  - C Compiler with high-level IR generation
  - High-level IR Linker
  - Retargetable Code Generator

Distributable Compilation Architecture

Diagram showing the flow of code compilation and optimization processes, including C source code, Assembly code, Specialized code, IR Linker, Mapper, and object modules.
Tools Development Advanced Phase

- Distributable compilation
- Retargetable development tools
  - Retargetable C Compiler (tunable CG and optimization)
  - Retargetable Assembler (target architecture templates)
  - Retargetable Simulator (for RCA configurations)
- Comprehensive Target Descriptive Language
- Target Tools Generator
- HDL code generation
- Joint hardware and software optimization

Co-Design Architecture

Software Design

- Source Code for RCA
- High Level IR
- Target Representation

Target Tools
- C Compiler
- CGI
- Assemblers
- RCA Simulator
- Debugger

RCA Hardware Design

- Comprehensive Target Description
- Tools Generator
- HDL Output
- HDL (VHDL)

communications technology lab
**Summary**

- RCA programming process characteristics
  - Parallel running processes with message exchange
  - Procedure level parallelism
  - “Partitioning-communication-aggregation-mapping” based optimization cycle
- RCA software development env contains standard set of tools for
  - Algorithm and source code development
  - Source code translation and linking
- RCA software development environment contains specific set of tools for the optimization cycle
- 3 phases of software tools development
  - Main goal of the naïve and mature phases is to assure (manually or automatically) program code effectiveness
  - Main goal of advanced phase is to assure joint hardware-software effectiveness of PHY/MAC algorithms implementation

**Acknowledgements**

Ernest Tsui, Vladimir Pudovkin, Vladimir Pavlov, Sergey Mironov, Veronica Mikheeva, Tony Chun, Michael K. Chen