Streams-C

Sc2 C-to-FPGA Compiler

Maya Gokhale, Janette Frigo, Christine Ahrens, Marc Popkin-Paine
Los Alamos National Laboratory

Janice M. Stone
Stone Ergonaut

Los Alamos National Lab
Overview

- **Language**
  - C subset augmented with parallel communicating processes
  - FIFO-based streams to communicate data between processes
  - Signals and Parameters for coordination and flow control
  - Process located on hardware (FPGA) or software (Linux PC)

- **Compiler**
  - Based on Stanford University Intermediate Format (SUIF) library
  - Targets Linux PC based AMS Firebird board
  - Easily re-targetable: board architecture described in a file
  - Generates Register-Transfer-Level VHDL
  - Source Code available at http://rcc.lanl.gov

- **Applications**
  - Signal and image processing
  - Fixed point, use external memory and Block RAM
Sc2 Processes

- Process body (the code it contains) is described in a process function
  - ///PROCESS_FUN directive describes a process function
  - Process function header describes streams, signals, and parameters that the process function uses

- Each process is an independent unit
  - ///PROCESS directive describes the process

- Processes execute concurrently
  - Sc_initiate intrinsic is used to start a process
  - Any software process may initiate another software process or hardware process

- Arrays of processes can be defined
Example: Process Function directives

Two process functions with input and output streams

// PROCESS_FUN host1_run
// OUT_STREAM sc_uint32 output_stream

// PROCESS_FUN_BODY

. . .

// PROCESS_FUN_END

// PROCESS_FUN controller_run
// IN_STREAM sc_uint32 input_stream
// OUT_STREAM sc_uint32 output_stream

// PROCESS_FUN_BODY

. . .

// PROCESS_FUN_END
Example: Process and Connect Directives

/// PROCESS controller PROCESS_FUN controller_run
   TYPE HP ON PE0

/// PROCESS host1 PROCESS_FUN host1_run

/// CONNECT host1.output_stream controller.input_stream

Connections can also be described graphically, and /// directives are generated
Streams and Signals

- Streams transmit data between processes
- Streams can be defined between software, hardware-software, and hardware processes
- Stream intrinsic functions are defined to:
  - Read
  - Check for end of stream
  - Write
- Hardware streams are implemented as hardware FIFOs with user-defined FIFO depth in the Streams-C hardware library
- Software streams are managed by the thread-based Streams-C software runtime library

- Signals are used to synchronize processes and coordinate phases of processing
- Signal intrinsic functions are defined to:
  - Post a signal, along with a single word of data
  - Wait for a signal and receive a single word of data
- Hardware and software signal implementation is similar to streams
- Parameters provide a mechanism for giving each newly initiated process a word of unique data.
Sc2 Code Example: Polyphase Filter

while( !sc_stream_eos(input_stream) ) {
    #pragma SC pipeline
    for (i=0 ; i < 4; i++) {
        #pragma SC unroll 4
        s[i] = sc_bit_extract(data, 0, 8);
        in1[i] = C1[i] * s[i];
        in2[i] = C2[i] * s[i];
        in3[i] = C3[i] * s[i];
        in4[i] = C4[i] * s[i];
        if (evenp) {
            y1[i] = e1[i] + (sc_uint8) sc_bit_extract(in1[i],12,8);
            e1[i] = e2[i] + (sc_uint8) sc_bit_extract(in2[i],12,8);
            e2[i] = e3[i] + (sc_uint8) sc_bit_extract(in3[i],12,8);
            e3[i] = sc_bit_extract(in4[i],12,8);
            sc_bit_insert(data_o, 0, 8, y1[i]);  }
        else {
            y2[i] = o1[i] + (sc_uint8) sc_bit_extract(in4[i],12,8);
            o1[i] = o2[i] + (sc_uint8) sc_bit_extract(in3[i],12,8);
            o2[i] = o3[i] + (sc_uint8) sc_bit_extract(in2[i],12,8);
            o3[i] = sc_bit_extract(in1[i],12,8);
            sc_bit_insert(data_o, 0, 8, y2[i]);  } /* end for loop */
    } /* end while loop */
    sc_stream_write(output_stream, data_o); /* filter output */
    data = sc_stream_read(input_stream);
    evenp = !evenp;  }

Loop is pipelined
Loop is unrolled
Access to memory arrays automatically scheduled
Compiler Structure

Functional Simulator

Software Library

Process Functions

Analysis & Scheduling

Sequence info + Datapath Operations

VHDL Generator

Hardware Synthesis Compiler

Hardware Library

Synthesis

Place and Route

RTL for Processing Element

Configuration Bit Stream

Host Processes

Software processes

Hardware processes
Synthesis Compiler Features

- Uses the SUIF 1.3 library (suif.stanford.edu)
- Uses Tim Callahan’s inline pass to inline function calls
- Optimizations include
  - SUIF optimizations such as constant folding, common sub-expression elimination, dead code elimination
  - Loop pipelining of innermost loops
  - Loop unrolling (directive)
- Compiler schedules sequential code, performs fine-grained parallelization
- Compiler reads board architecture from a file
  - Easily retargetable
- Compiler source is available at rcc.lanl.gov
Board Definition File

Memory Type EXTERNAL64
Data size 64 bits
Read/Write Port
  OUT    MAR width 32 bits
  BUFFER MDR width 64 bits
  OUT    R_EN width 1 bit
  OUT    W_EN width 1 bit
Identify MAR_name MAR
Identify MDR_name MDR
Identify Read_enable_name R_EN
Identify Write_enable_name W_EN
Load     latency 7  MAR, MDR, MDR, MDR, MDR, MDR, MDR, MDR
Store    latency 1  (MAR, MDR)
Memcopy  latency 8  MAR, MDR, MDR, MDR, MDR, MDR, MDR,(MAR,MDR)

Architecture Firebird
Board Virtex2000
Processor PE0
  4 EXTERNAL64 memory mem_size 1000000  memory-number 0
    controller Mem641
    generics ( schedule = priority, LADbase=0x1000, LADinc=0x200,
        mem_component=EXTERNAL)
Applications

- Poly phase filter bank of four
  - Ppf_a: 32-bit stream input data, external memory for coefficients
  - Ppf_ab: 32-bit stream input data, block ram for coefficients
  - Ppf1: 64-bit external memory input data, registers for coefficients
  - Ppf: 32-bit stream input data, registers for coefficients

- K-Means Clustering
  - Unsupervised clustering of multi-spectral data
  - 32-bit stream input data, block ram for centers

- Fast Folding
  - Modified butterfly FFT

- Performance evaluation in progress – automatically generated hardware ppf1 faster than GHz Pentium

- Applications source code available on web site
Summary

- Streams-C compiler - synthesizes hardware circuits that run on reconfigurable hardware from parallel C programs
- C-to-hardware tool with parallel programming model and efficient hardware libraries
- Functional Simulator runs on host workstation
- 5 - 10x faster development time over hand-coded
- Performance comparable to GHz Pentium
- Open Source – we welcome collaborations