Synchronous Dataflow Processing

Claus Traulsen and Reinhard von Hanxleden

Christian-Albrechts Universität zu Kiel
Echtzeitsysteme / Eingebettete Systeme

March 2010
Outline

Motivation

Kiel Lustre Processor
   Architecture
   Compilation
   Experimental Results

Conclusion
Reactive Systems

- Continuously react to inputs
- Reaction time is determined by environment
- Safety critical
- Examples: airbag, flight control, ...

Key Observation:
Reactive control flow is not matched by common processors:
- Concurrency: handle multiple inputs simultaneously
- Preemption: deterministic suspend and abort
- Worst case more important than average case
Synchronous Languages

Synchrony Hypothesis:
Reactions do not take time
⇒ discrete ticks.

Features:
- Two forms of logical concurrency
  - Explicit concurrency
  - Preemption
- Deterministic behavior
- Compilation to Software / Hardware / Mix
- Best known:
  - Esterel / SyncCharts: imperative
  - Lustre / Scade: dataflow
Reactive Processors

- Special processors to support reactive control flow
- Inspired by synchronous languages

Benefits

- Deterministic behavior
  Simplify WCRT analysis
- Better resource usage
  - Program size
  - Power consumption
  - Resource per high-level operation
- Dependability
  Close link between high-level-language ↔ assembler
  ⇒ correct compiler, traceability
Related Work

Real-time Processors:
- PReT: Precision Timed machine (Edwards, Lee)
- JOP: Java Optimized Processor (Schöberl)

Reactive Processors for Esterel:
- EMPEROR: (Roop, et al.)
- KEP: Kiel Esterel Processor (Li)
- StarPRO (Roop, et al.)

Synchronous Language Extensions:
- SC (von Hanxleden)
- PRET-C: (Girault, Roop, et al.)

Compilation of Synchronous Languages:
- Automata (Berry, Halbwachs)
- Static Scheduling (Halbwachs, Edwards, Potop-Butucaru)
- Netlists (Berry, Halbwachs)
KLP: Basic Ideas

Processor to execute arbitrary Lustre / Scade programs

- Direct mapping of equations to hardware
- Explore low level parallelism
- Support mixed Data- / Control-flow
Architecture Overview

Register

- Value
- Previous value
- PC
- (Priority)
- (Clock)
- Done flag
Instruction Set Architecture

INPUT C
OUTPUT O
SETPC O L_O
SETCLK O C
PRIO O 1
OUTPUT S
SETPC S L_S
PRIO S 2
DONE

L_O: IADD O pre(O) 1
DONE L_O

L_S: ADD S O pre(S)
DONE L_S
Execution Model

KLP execution

Running

/ preVal=val, done=false

/ execute ready reg / update flags

Tick

all done
Parallel Execution

Model can be naturally extended to parallel execution

1. Dynamic scheduling
   - Each registers computes ready status
   - Based on next instruction and Done flags
   - Program is acyclic $\Rightarrow$ no deadlock

2. Priority based scheduling
   - Assign priorities at compile time
   - $PRIO(R1) > PRIO(R2)$ computation of $R2$ depends on $R1$
   - Might miss potential parallelism
   - Simpler hardware
Resource Usage: Delay
Compilation

Lustre Model

Extended code (ec)

Clocked Equations (CEQ)
\[ v = \text{current}((i \rightarrow e) \text{ when } c) \]

Scade Model

KLP Assembler
Compiling Automata

Problem

- Non-local dependencies
- Dynamic reconfiguration of registers

Solution

- One control thread
- Logic:
  1. Check strong abort
  2. Execute content
  3. Check weak abort
- Implemented by priorities
- Parallel execution of content

```
LOCAL SM
SETPC SM L_SM_S1
PRIO SM 0
...
L_SM_S1: JT A L_S12S2
  PRIO SM 2
  PRIO SM 0
  DONE SM
L_S12S2: SETPC C L_C_S2
  PRIO C 2
  JMP L_SM_S2
L_SM_S2: ...
```
Experimental Results

Implementation

- Processor described in Esterel: Compile to software simulation (C) or hardware (VHDL)
- Compiler and evaluation bench part of KIELER

Validation

- Performance evaluation: compare to C code running on MicroBlaze
Worst Case Reaction Time

![Graph showing worst case reaction time for different components: abro, counter, elevator_lus, watch, parallel. The x-axis represents different components, and the y-axis represents time in microseconds (us). The graph compares KLP 1, KLP 4, v4, and reluc.](image-url)
Code Size

![Bar chart showing code size in KBytes for different processes and versions. The x-axis represents different processes: abro, counter, elevator_lus, elevator_scade, watch, parallel. The y-axis represents code size in KBytes, ranging from 0 to 6. The chart includes data for KLP, v4, and reluc versions.]
Conclusion

- Reactive Processors:
  - Special instruction set for concurrency + preemption
  - Simplify WCRT analysis
- Dataflow-processor for Lustre / Scade
  - Direct support for dataflow equation
  - Support clocks
  - Parallel execution:
    - dynamic scheduling or priorities
  - Combined data- / control flow

Further information
www.informatik.uni-kiel.de/rtsys/krep

Thank you for your attention
Questions?