Reactive Processing

KReP, KEP, and Esterel

Malte Tiedje Claus Traulsen

Real-Time Systems and Embedded Systems
Department of Computer Science
Christian-Albrechts-University Kiel

SLA++P 2008

Outline

KEP and KReP

Two reactive processors from Kiel:

KEP	- ISA based on Esterel - Full support for concurrency and preemption - Multi-threaded - Implemented in VHDL and Esterel v7 - Available at `www.informatik.uni-kiel.de/rtsys/kep`

KEP and KReP

Two reactive processors from Kiel:

KEP	▶ ISA based on Esterel ▶ Full support for concurrency and preemption ▶ Multi-threaded ▶ Implemented in VHDL and Esterel v7 ▶ Available at `www.informatik.uni-kiel.de/rtsys/kep`
KReP	▶ Reactive processing without Esterel ▶ More dataflow oriented (Lustre, Scade) ▶ Multicore ▶ Implemented in Esterel v7 (prototype)

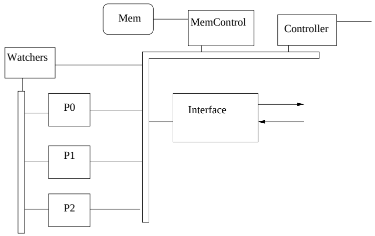
KEP and KReP

Two reactive processors from Kiel:

KEP	▶ ISA based on Esterel ▶ Full support for concurrency and preemption ▶ Multi-threaded ▶ Implemented in VHDL and Esterel v7 ▶ Available at `www.informatik.uni-kiel.de/rtsys/kep`
KReP	▶ Reactive processing without Esterel ▶ More dataflow oriented (Lustre, Scade) ▶ Multicore ▶ Implemented in Esterel v7 (prototype) ▶ **Work in Progress**

Processor: Basic Ideas

- Small Cores with simple ISA
- Special `sync` instruction
- Parallel or pipelined execution
- Fast and/or precise timing
- Special instructions for Automata

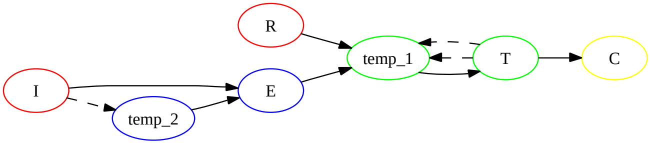


Compiler: Basic Ideas

Takes Lustre programs as inputs

1. Create dependency graph
2. Order Equations
3. add pre to preserve synchrony
4. Assign cores to variables

```
node COUNTER(R:bool; I:bool)
  returns (C:int);
T : int;
E : int;
let
E = I and not pre(I);
T = 0 -> if R
         then 0
         else if E
              then pre(T)+1
              else pre(T);
C = T;
tel
```



Demo

There exists at least one Lustre program, which can be compiled and executed on the KReP, giving the correct behavior.

Open Questions / To do

- Handle automata
- Benefit from Lustre clocks
- Timing Constraints
- Performance Evaluation
- Tune processor and compiler
- Correctness

  ⋮

Description of KEP and KReP

Both are descriped in Esterel v7

- Better maintainability
- Gain more experience with Esterel

Description of KEP and KReP

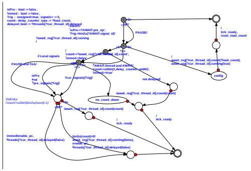
Both are descriped in Esterel v7

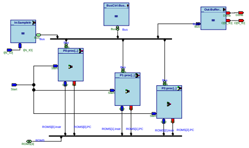
- Better maintainability
- Gain more experience with Esterel

Its also a nice Benchmark

KEP
- 6 Safe State Machines
- 76 Esterel Modules
- 4759 lines Esterel Code

KReP
- 3 Diagrams
- 26 Esterel Modules
- 1361 lines Esterel Code





Experience with Esterel v7

Mainly positive:

- Easy to use
- Nice formal tools
- SW + HW generation

Experience with Esterel v7

Mainly positive:

- Easy to use
- Nice formal tools
- SW + HW generation

Some minor problems:

- Weird compilation error (internal error in . . .)
- May not use complex expressions
- Code efficiency highly influenced by programming style
- State coverage not exhaustive

Experience with Esterel v7

Mainly positive:

- Easy to use
- Nice formal tools
- SW + HW generation

Some minor problems:

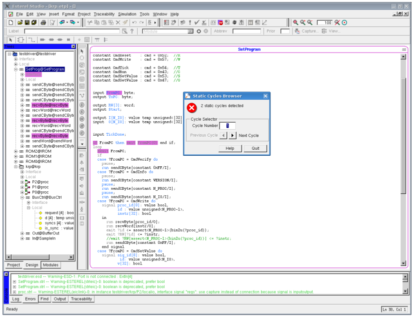
- Weird compilation error (internal error in . . .)
- May not use complex expressions
- Code efficiency highly influenced by programming style
- State coverage not exhaustive

Two major ones:

- Cyclic programs
- Interfacing the real world

Cyclic Programs

- What is a cycle?
  Depends on the compiler.
- Hard to find (not local)
- Tool should show minimal cycle
- Main obstacles while programming



```
constant CmdReset    : cmd = 0x52;  //R
constant CmdWrite    : cmd = 0x57;  //W

constant CmdTick     : cmd = 0x54;  //T
constant CmdRun      : cmd = 0x43;  //C
constant CmdSetValue : cmd = 0x53;  //S
constant CmdGetValue : cmd = 0x47;  //G


input FromPC: byte;
output ToPC: byte;

output RW[3]: word;
output Start;

output I[N_IO]: value temp unsigned<[32]
input  O[N_IO]: value temp unsigned<[32]


input TickDone;

if FromPC then emit FromPC(0) end if;
loop
  await FromPC;
  if
  case ?FromPC = CmdVerify do
    pause;
    run sendCByte[constant 0xFF/I];
  case ?FromPC = CmdInfo do
    pause;
    run sendCByte[constant VERSION/I];
    pause;
    run sendCByte[constant N_PROC/I];
    pause;
    run sendCByte[constant N_IO/I];
  case ?FromPC = CmdWrite do
    signal proc_id[8]: value bool,
           id : value unsigned<N_PROC-1>,
           instr[32]: bool
    in
      run recvByte[proc_id/O];
      run recvWord[instr/O];
      emit ?id <= assert<N_PROC-1>(bin2u(?proc_id));
      emit ?RW[?id] <= ?instr;
      //emit ?RW[assert<N_PROC-1>(bin2u(?proc_id))] <= ?instr;
      run sendCByte[constant 0xFF/I];
    end signal
  case ?FromPC = CmdSetValue do
    signal sig_id[8]: value bool,
            id: value unsigned<N_IO>,
           v[32]: bool
```

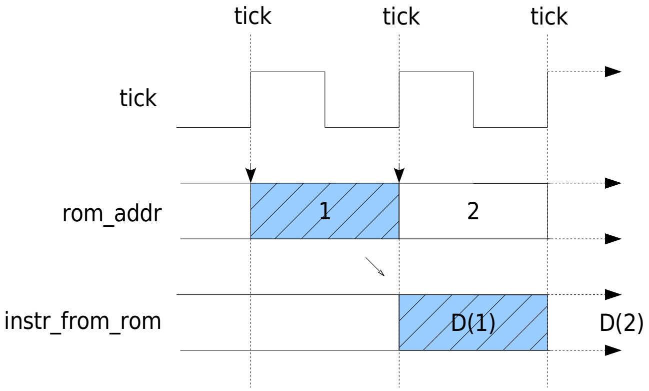
Static Cycles Browser

2 static cycles detected

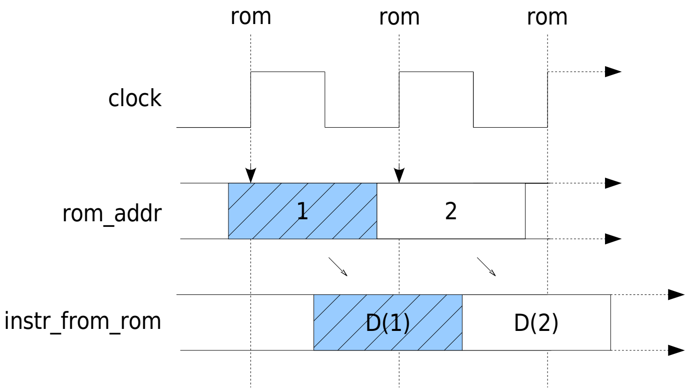
Cycle Selector
Cycle Number: 1
Previous Cycle | Next Cycle
Help | Quit

```
testdriver.esd -- Warning-ESD-1: Port is not connected : ExtIn[4]
SetProgram.strl -- Warning-ESTEREL(strleic)-0: boolean is deprecated, prefer bool
SetProgram.strl -- Warning-ESTEREL(strleic)-0: boolean is deprecated, prefer bool
proc.strl -- Warning-ESTEREL(eiclink)-0: in instance testdriver/krp/P2/localio, interface signal "regs": use capture instead of connection because signal is inputoutput.
```

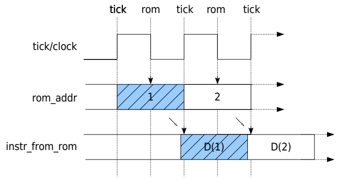
Interfaces: Synchronous view



Interfaces: Real world



Interfaces: Solution



- Add pause statements
- Request acknowledgement
- Multiclock

Conclusion

KReP
- Dataflow reactive processor seams feasible
- Still a lot to do
- Benefits?

Conclusion

KReP	▶ Dataflow reactive processor seams feasible
	▶ Still a lot to do
	▶ Benefits?
Esterel	▶ Easy to get prototype HW
	▶ Hard to get efficient implementation
	▶ Need some HW knowledge
	▶ Interfaces and Cycles main problems