HW/SW Co-Design for a Reactive Processor

Sascha Gädtke, Xin Li, Marian Boldt, Reinhard von Hanxleden

Real-Time and Embedded Systems Group
Department of Computer Science
Faculty of Engineering
Christian-Albrechts-University Kiel, Germany

Student Poster Session, LCTES’06
Ottawa, June 14-16, 2006
Classical approach: Run an Esterel program on a reactive processor like the *Kiel Esterel Processor (KEP)*

**Drawback:** Complex signal expressions have to be sequentialized into a series of KEP assembler instructions

**Our Approach:** Accelerate reactive processing via external logic block

---

**Figure:** EXAMPLE Esterel program and resulting KEP assembler code

```plaintext
module EXAMPLE:
input A, B, C;
output O, P;

present (A and C) or (B and C) then
  emit O
end;
present A or B then
  emit P
end
end module

1 % module EXAMPLE
2 INPUT A
3 INPUT B
4 INPUT C
5 OUTPUT O
6 OUTPUT P
7 PRESENT A,A0
8 PRESENT C,A0
9 GOTO A1
10 A0: PRESENT B,A2
11 PRESENT C,A2
12 A1: EMIT O
13 A2: PRESENT A,A3
14 GOTO A4
15 A3: PRESENT B,A5
16 A4: EMIT P
17 A5: HALT
```
**Approach**

**First Step: Source Code Transformation**

Original Esterel program partitioned into three modules
Approach

First Step: Source Code Transformation

Original Esterel program partitioned into three modules

- **SW module**
  - Copy of the original program
  - Complex signal expressions are replaced with auxiliary signals

```
module EXAMPLE_SW:

output O;
output P;
input A_and_C_or_B_and_C;
input A_or_B;

present A_and_C_or_B_and_C then
  emit O
end present;
present A_or_B then
  emit P
end present

end module
```
Approach

First Step: Source Code Transformation

Original Esterel program partitioned into three modules

- **SW module**
  - Copy of the original program
  - Complex signal expressions are replaced with auxiliary signals

- **HW module**
  - Computes complex signal expressions in parallel threads
  - At any time a signal expression is TRUE, the auxiliary signal is emitted

```plaintext
module EXAMPLE_SW:
  output O;
  output P;
  input A_and_C_or_B_and_C;
  input A_or_B;

  present A_and_C_or_B_and_C then
    emit O
  end present;
  present A_or_B then
    emit P
  end present
end module
```

```plaintext
module EXAMPLE_HW:
  input A, C, B;
  output A_and_C_or_B_and_C, A_or_B;

  every immediate [A_or_B and C] do
    emit A_and_C_or_B_and_C
  end every
  ||
  every immediate [B or A] do
    emit A_or_B
  end every
end module
```
Approach

First Step: Source Code Transformation

Original Esterel program partitioned into three modules

- **SW module**
  - Copy of the original program
  - Complex signal expressions are replaced with auxiliary signals

- **HW module**
  - Computes complex signal expressions in parallel threads
  - At any time a signal expression is TRUE, the auxiliary signal is emitted

- **Main module runs SW- and HW module in parallel**

```plaintext
module EXAMPLE_SW:
    output O;
    output P;
    input A and C or B and C;
    input A or B;

    present A and C or B and C then
    emit O
end present;

present A or B then
emit P
end present

end module
```

```plaintext
module EXAMPLE_HW:
    input A, C, B;
    output A and C or B and C, A or B;

    every immediate [A or B and C] do
    emit A and C or B and C
end every
||
 every immediate [B or A] do
emit A or B
end every

end module
```

```plaintext
module EXAMPLE:
    input A, B, C;
    output O, P;

    signal A and C or B and C,
    A or B in

run EXAMPLE_SW
|
run EXAMPLE_HW
end signal

end module
```
Approach

Second Step: HW/SW Synthesis

- SW module is compiled to KEP assembler code and then executed on the KEP
- HW module is transformed into a VHDL description of a combinational logic
- KEP has to provide an interface to the logic block

Figure: Schematic of the KEP with external logic block
Experimental Results

Test conditions
- Benchmark: TCINT from the EstBench suite

Results
- KEP code already faster than the fastest MicroBlaze code generated by the Esterel V7 compiler
- Maximum execution time of the co-design has reduced to less than the half of the pure SW solution

<table>
<thead>
<tr>
<th></th>
<th>Original program (SW)</th>
<th>Partitioned program (SW)</th>
<th>HW+SW</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MicroBlaze</td>
<td>KEP</td>
<td></td>
</tr>
<tr>
<td></td>
<td>V5  V7  CEC</td>
<td>V5  V7  CEC</td>
<td>KEP</td>
</tr>
<tr>
<td>Memory (in bytes)</td>
<td>14860  11376  15340</td>
<td>3527  17308  11416  17460</td>
<td>4471  1894</td>
</tr>
<tr>
<td>Average Clock cycles</td>
<td>3488  1797  2121</td>
<td>261  4248  1826  2971</td>
<td>720  204</td>
</tr>
<tr>
<td>Maximum Clock cycles</td>
<td>3580  1878  2350</td>
<td>729  4336  1907  3311</td>
<td>981  345</td>
</tr>
<tr>
<td>Empty input</td>
<td>3476  1807  2101</td>
<td>237  4267  1838  2956</td>
<td>771  204</td>
</tr>
</tbody>
</table>

**Table:** Experimental results for the TCINT benchmark.