

HW/SW Co-Design for a Reactive Processor

Motivation

- One approach to deal with Esterel programs is to run them on a reactive processor like the *Kiel Esterel Processor (KEP)* [1]
- **Drawback:** In KEP assembler code the computation of complex signal expressions has to be sequentialized into a series of instructions:
 - For example the “present (A and C) or (B and C) ...” statement in the EXAMPLE code is translated to the five KEP assembler instructions in lines 7-11.
- We present an approach to accelerate reactive processing via an external logic block. To provide easier validation, the transformation is done in two steps with an optional intermediate logic minimization step:
 - In the first step partitioning into SW and HW parts is done at the Esterel level
 - In the second step SW and HW synthesis is done

Approach

First step: Source Code Transformation

- Transformation of the Esterel source code into equivalent Esterel program consisting of three modules: A concurrent *software* and *hardware* module and a main module running them in parallel
- SW-Module
 - Copy of the original program
 - Complex signal expressions are replaced with auxiliary signals
- HW-Module (*logic block*)
 - Computes complex signal expressions in parallel threads
 - Every time a signal expression is TRUE, the auxiliary signal is emitted

Second step: HW/SW Synthesis

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

Experiments

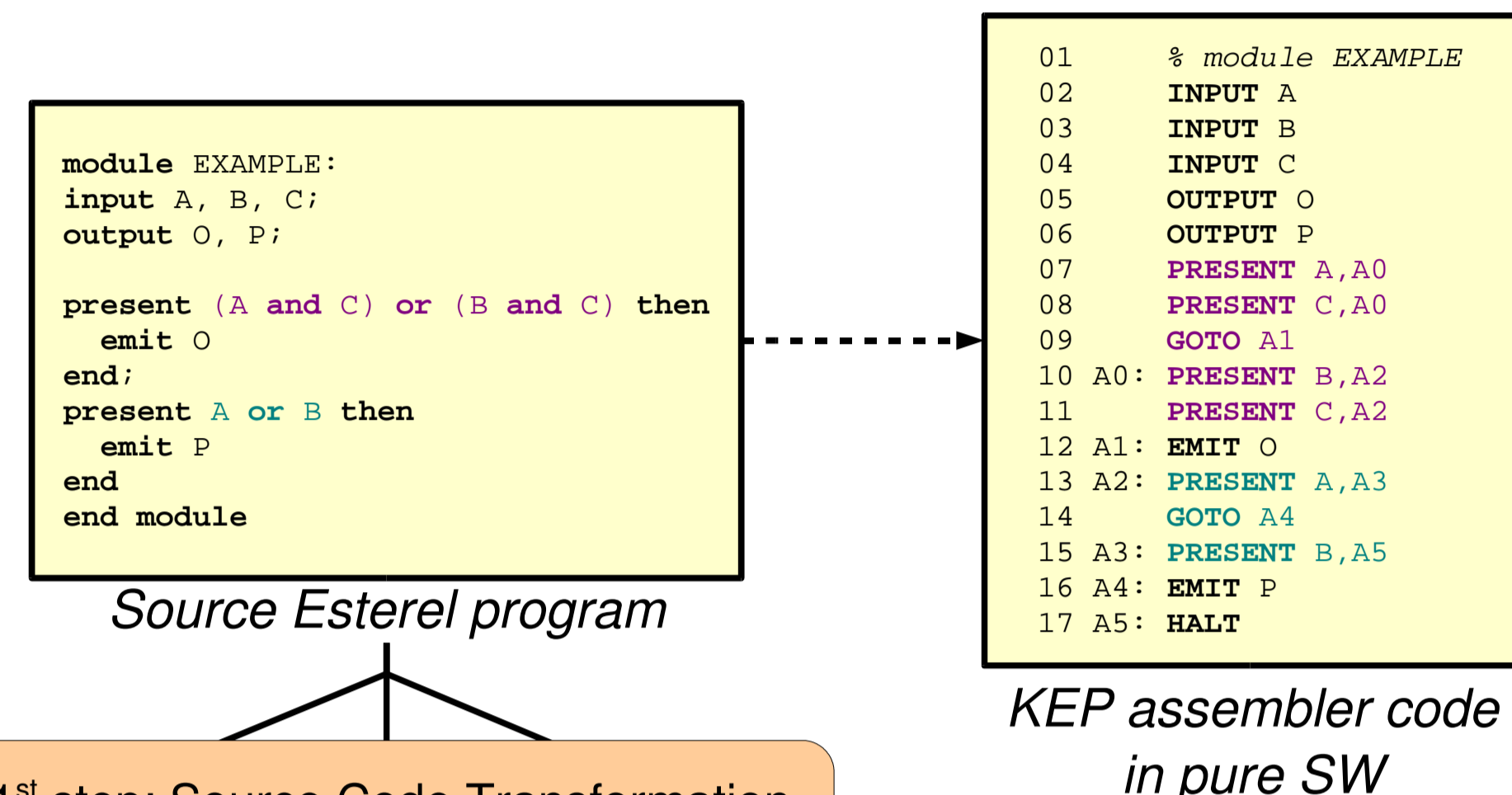
Test conditions

- Benchmark: TCINT from the EstBench suite [2]
- Comparison of code size and speed of the original code, the partitioned program (pure SW implementation) and the co-design
- Microblaze code generated by the Esterel V5 and V7 compilers and the Columbia Esterel Compiler (CEC) [3]

Results

- KEP code is always more compact and faster than the fastest Microblaze code
- The maximum execution time of the co-design has reduced to less than the half of the pure SW solution

Classical SW Synthesis



```

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
    
```

SW-Module

```

module EXAMPLE_MAIN:
input A;
input B;
input C;
output O;
output P;

signal A_and_C_or_B_and_C,
A_or_B in
run EXAMPLE_SW
||
run EXAMPLE_HW
end signal
end module
    
```

Main-Module

```

module EXAMPLE_HW:
input A;
input C;
input B;
output A_and_C_or_B_and_C;
output 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
    
```

HW-Module

2nd step: HW/SW Synthesis

```

01 % module EXAMPLE_SW
02 INPUT A_and_C_or_B_and_C
03 INPUT A_or_B
04 OUTPUT O
05 OUTPUT P
06 PRESENT A_and_C_or_B_and_C,A0
07 EMIT O
08 A0: PRESENT A_or_B,A1
09 EMIT P
10 A1: HALT
    
```

KEP assembler code of the SW-Module

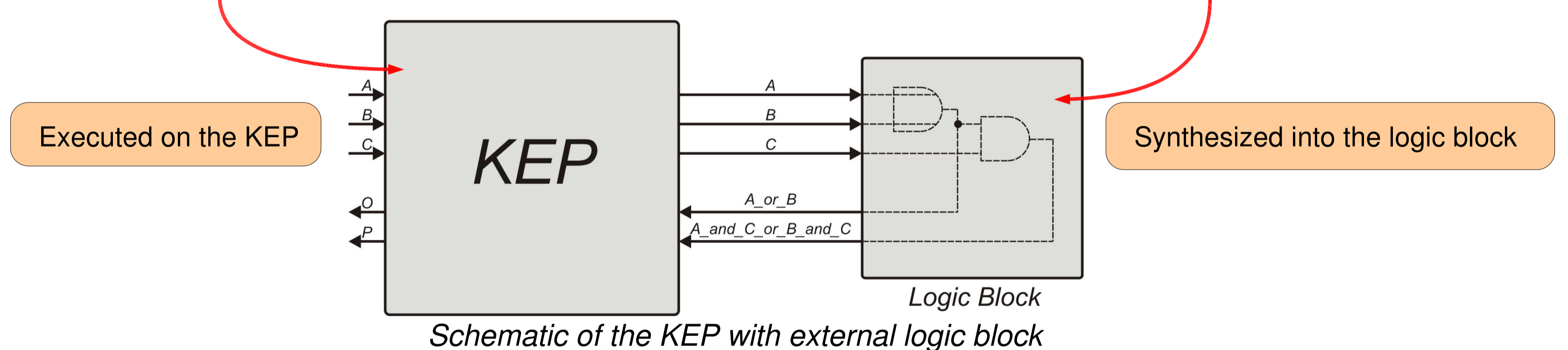
```

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity EXAMPLE_HW is
port(A: in std_logic;
C: in std_logic;
B: in std_logic;
A_and_C_or_B_and_C: out std_logic;
A_or_B: out std_logic);
end EXAMPLE_HW;

architecture EXAMPLE_HW_BEH of EXAMPLE_HW is
signal A_or_B_int: std_logic;
begin
A_and_C_or_B_and_C <= A_or_B_int and C;
A_or_B_int <= B or A;
A_or_B <= A_or_B_int;
end EXAMPLE_HW_BEH;
    
```

VHDL description of the logic block



Schematic of the KEP with external logic block

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

Experimental results for the TCINT benchmark

[1] Xin Li and Reinhard von Hanxleden. Mapping Esterel onto a Multi-Threaded Embedded Processor. In *Proceedings of the 12th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS'06)*, San Jose, CA, October 21-25, 2006
 [2] EstBench Esterel Benchmark Suite. <http://www1.cs.columbia.edu/~sedwards/software/estbench-1.0.tar.gz>. [3] S. Edwards. CEC: The Columbia Esterel Compiler. <http://www1.cs.columbia.edu/~sedwards/cec/>.

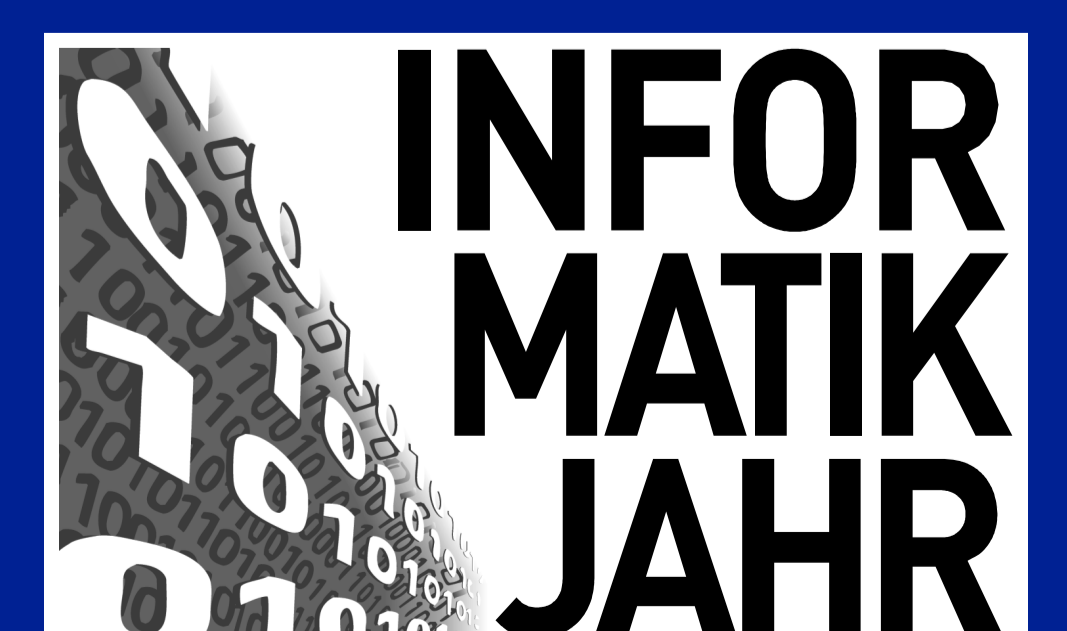
Contact Person:

Prof. Dr. Reinhard von Hanxleden
 Real-Time and Embedded Systems
 Christian-Albrechts-Universität zu Kiel
 Olshausenstr. 40, 24098 Kiel
 Phone: +49 (0)431 880-7281
 Fax: +49 (0)431 880-7615
 rvh@informatik.uni-kiel.de

Contact Person:

Sascha Gädtke
 Real-Time and Embedded Systems
 Christian-Albrechts-Universität zu Kiel
 Olshausenstr. 40, 24098 Kiel

sga@informatik.uni-kiel.de



Wissenschaftsjahr 2006