KEP

The Kiel Esterel Processor is a reactive processor.

Its ISA is based on Esterel v5 and has direct support for preemption and concurrency.

There are two implementations of it.

Testcases

In order to generate esi/eso testcases for an Esterel Program, you can run the Makefile esieso_gen.makefile from kep/Misc/scripts/. It performs codecoverage on the Esterel source program and than simulate the outputs. Check, if you have the esi/eso tools (Esterel) in your path. If the Esterel Studio Version changes, you may gaet the mesg. "Obsolete configuration(s) detected. Performing automatic update."

KEP in VHDL

This is the original implementation by Xin Li.

KEP in Esterel

The KEP was reimplemented in Esterel v7 using Esterel Studio. There are some limitations:

- No valued signals
- No traps

SoftKEP

An advantage of the Esterel implementation is, that it can be compiled to C code, for easier testing. The SoftKep has been tested under Linux and Windows with Cygwin.

Installation for Windows

```
# Install cygwin with the packages make, perl and gcc.
# Install the  null-modem emulator  [http://com0com.sourceforge.net/ com0com]
# Us com0com to emulate a connection from COM1 to COM2: '''install PortName=COM1 PortName=Com2'''
# In the kep directory, goto KEP/Esterel/development/kep-e and adjust the paths in the makefile.
# Create a softkep with '''make soft.small.kep'''
```

This requires the installation of Esterel Studio. The COM port the softkep uses is hardcoded in softkep_data.c, you might want to adjust this setting.

Start the KepEvalBench from Verifier/development/KepEvalBenchSoft.

HardKep

The following steps are necessary to get the KEP running on an FPGA board.

You either have to install gmake, or simply link make to gmake.

Ensure that your COM1 port is set to 115200baud, 8 Databits, no parity and no flow control.

Compute parameters for RS232 connection

There are two parameters that effect the sample rate of the RS232 connection:

- the frequency of the base clock
- the uart_clk_div constant in data_types/data.strl/constants

You can use the script *calc_clk* to determine the correct value. It takes the rate of the base clock (in kHz) as ean input parameter. The script returns all possible values for the dcm, oversampling and the clock_div in the KEP. You can choose any of the given values, but you should try to keep an oversampling rate of 16, and the frequency for the KEP should not be to high. Set the constant uart_clk_div in the kep-e project to the computed value value of kep_clock_div. You will use the value for dcm_clock_div for later to generate a digitial clock manager.

Good but slow values for the virtex2P board are, uart_clk_div=2, clock divied in dcm 9, when using the 32MHz clock.

Translate Esterel to vhdl

goto KEP/Esterel/development/kep-e and execute

> *make hard.small.kep or make hard_modular.small.kep.*

This requires Esterel-Studio. You may choose other sizes, but note that anything larger than small may take a long time to synthesize, and may not fit on our boards .

Three files are generated in the hardkep subdirectory:

> **esterel_numeric_std.vhd *hardkep_data_pkg_template.vhd *hardkep.vhd*

Xilinx ISE Project

Generate a new Xilix ISE project with the appropriate board parameters. Copy or link the generated vhdl files into the project.

Instruction ROM

Create a new source. Choose IP (Coregen & Architecture Wizard) and name it ROM. Choose Memories->RAMs->Single Port Block Memory. The width is determined by the size of the opcode (40 Bits). Choose the depth for the ROM according to the generated version of the KEP, the value is printed by the makefile.

Trace Buffer (optional)

The Trace Buffer stores the execution trace of every instance, it is not mandatory for the execution of programs. Create a new source. Choose IP (Coregen & Architecture Wizard) and name it TRACE. Choose Memories->RAMs->Single Port Block Memory. The width is determined by the depth of Instruction ROM: log2(depth of ROM). Choose the depth for the TRACE according to the generated version of the KEP, the value is printed by the makefile.

Clock

Create a new source. Choose IP (Coregen & Architecture Wizard) and name it dcm. Choose FPGA Featurs->Clocking-> Virtex II -> Single DCM.

Set the Input Frequency to one available on the board and the divide value according to the uart_clk_div constant. Activate the CLKDV output.

Connect Schematics

Create a schematic symbol for the hardkep, the clock and the ROM.

Create a new schematic source and add the symbols for the hardkep, the clock and the ROM. Do _not_ name it hardkep, otherwise Xilinx ISE will crash. Connect the following ports:

- input:clk -> dcm:CLKIN_IN
- dcm:CLKDV_OUT -> hardkep:clk
- dcm:CLKDV_OUT -> ROM:clk
- dcm:CLKDV_OUT -> TRACE:clk
- input:rst-> hardkep:rst
- input:RX -> hardkep: RX
- ROM:dout -> hardkep: instr_from_rom
- TRACE:dout -> hardkep: pc_val_from_buffer
- hardkep:rom_addr_data -> ROM addr
- hardkep:instr_to_rom -> ROM:din
- hardkep:wea_instr -> ROM:we
- hardkep:trace_buffer_addr -> TRACE addr
- hardkep:pc_val_to_buffer -> TRACE:din
- hardkep:wea_trace -> TRACE:we
- hardkep:TX -> output:TX
- hardkep:no_rx_error -> output:rx_error
- hardkep:tx_busy -> output:tx_busy
- hardkep:no_tx_error -> output:tx_error
- tick_warn -> output:tick_warn

User Constraint File

Connect the global I/O to the corresponding interfaces of the board. You can find an example user constraint file in the hardkep subdirectory.

Upload bit-file

Now generate the programming file and upload it to the Board. After this, youd have to reset the KEP once, but setting the rst input. (switch 0 or the Virtex 2+ board).

Test

Connenct the FPGA to the PC via the serial port. You can test the KEP with a Hyperterminal by sending V to the KEP. It should reply by sending 012345...