RIVYERA

reconfigure versatilely your efficient raw architecture

PingPong

Version 2.0.1.0

SciEngines GmbH
Fraunhoferstrasse 13
24118 Kiel
Germany

www.sciengines.com

Revision: 1.91.00 January 24, 2013

Table of Contents

1 Introduction

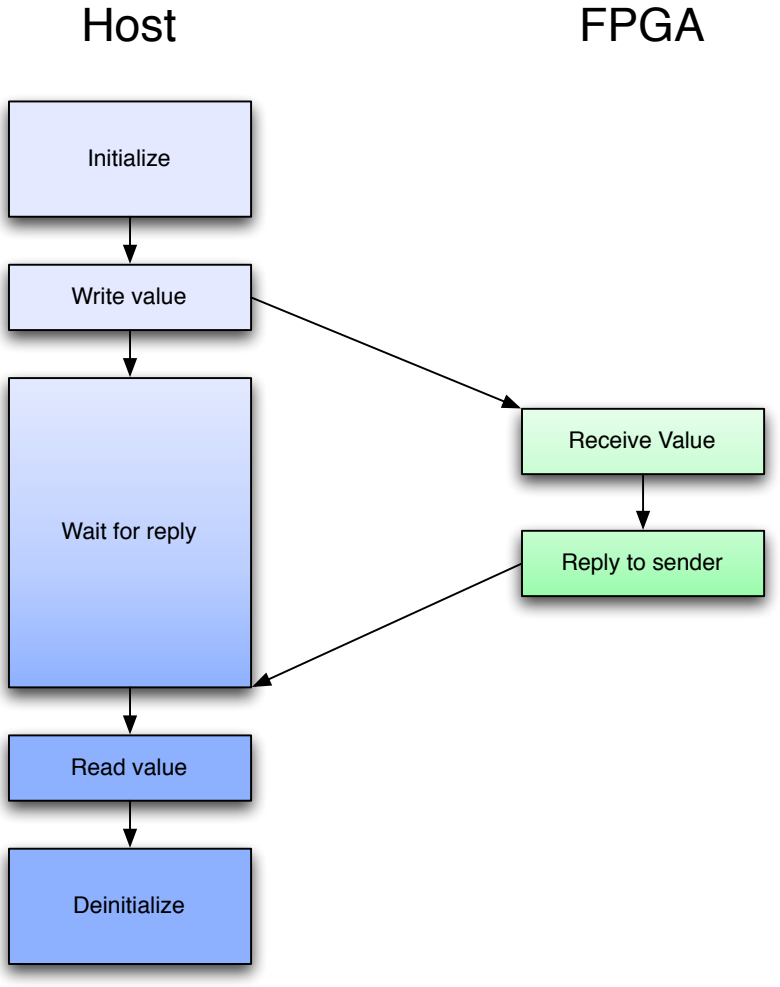
SciEngines RIVYERA is a high performance reconfigurable computing platform. It is capable of massive performance gains compared to standard architectures. However, it does not follow the *von Neumann* architecture and therefore it requires a different style of programming, which is illustrated in this sample.
PingPong is a very simple communication example to get in first touch with SciEngines RIVYERA platform. It follows the basic functionality of the commonly known *ping*. In general, *ping* is executed on one system, triggers a communication partner and waits for its reply. Common implementations additonally measure the time, that it takes the system to complete the action. In this example, things are kept as easy as possible, which is why an additonal timer is not added.
To stress the asynchronity of the system, which is a really important factor in algorithmic design, the autonomous write feature of the Machine API is used. Therefore, the FPGA directly replies to sent words, without waiting for a read request.

2 RIVYERA Implementation

This example shows how to realize simple communication with an FPGA. As noted in the introduction, the example uses the FPGA's autonomous write feature, which causes the host not to actively request information but only waits for an answer of any FPGA. This mechanism for reading is also called *passive read*.

The PingPong example is implemented as follows: After being programmed, an arbitrary number is written to any FPGA on any slots. Whenever an FPGA receives a number (signaled by `api_i_empty_in` low), it reacts by sending exactly the same number back to the sender.

Now after sending, the host waits for incoming data using the API function `se_waitForData()`. `se_waitForData()` waits for data from any possible source to be sent to the host and then writes the source address to the provided `SE_ADDR` structure. Using `se_waitForData()` is not really necessary in this example because we always know the address for incoming data: It is the same as the target address for the `se_write()` call. Once `se_waitForData()` recognizes incoming data (this is when `se_waitForData()` returns successfully), it reads the data from the API internal buffer and writes it to the user provided buffer.



3 FPGA VHDL Sources

3.1 pingpong_main.vhd

```
1  ---
2  -- Project: PingPong
3  -- File:    pingpong_main.vhd
4  -- Date:    Thu Nov 22 16:03:58 CET 2012
5  -- Author:  sciengines
6  --
7  -- Description:
8  -- Copyright (c) 2012-2013, SciEngines GmbH
9  -- All rights reserved.
10 --
11 -- Redistribution and use in binary forms, with or without modification,
12 -- are permitted provided that the following conditions are met
13 --
14 --   * Redistributions of source code is not permitted otherwise
15 --     redistributions of source code must retain the above copyright
16 --     notice, this list of conditions and the following disclaimer.
17 --   * Redistributions in binary form must reproduce the above copyright
18 --     notice, this list of conditions and the following disclaimer in
19 --     the documentation and/or other materials provided with the
20 --     distribution.
21 --   * Neither the name of the SciEngines GmbH nor the names of its
22 --     contributors may be used to endorse or promote products derived
23 --     from this software without specific prior written permission.
24 --
25 -- THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
26 -- "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
27 -- LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
28 -- A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
29 -- OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
30 -- SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
31 -- LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
32 -- DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
33 -- THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
34 -- (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
35 -- OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
36 ---
37 library ieee;
38 use work.sciengines_api_types.all;
39 use ieee.STD_LOGIC_1164.all;
40
41 -- Uncomment the following library declaration if using
42 -- arithmetic functions with Signed or Unsigned values
43 --use IEEE.NUMERIC_STD.ALL;
44
45 -- Uncomment the following library declaration if instantiating
46 -- any Xilinx primitives in this code.
47 --library UNISIM;
48 --use UNISIM.VComponents.all;
49
50 entity pingpong_main is
51    generic (
52       NUM_LEDS               : integer
53    );
54    port (
55       ------------------------------
56       --------- API PORTS ---------
57       ------------------------------
58       -- USER PORTS
59       api_clk_in             : in    std_logic;
60       api_rst_in             : in    std_logic;
61       api_led_out            : out   seBusFlag_type(NUM_LEDS-1 downto 0)
             := (others => '0');
62       -- ADDRES PORTS
63       api_self_contr_in      : in    seFlag_type;
64       api_next_contr_in      : in    seSlotAddr_type;
65       api_prev_contr_in      : in    seSlotAddr_type;
66       api_self_slot_in       : in    seSlotAddr_type;
67       api_self_fpga_in       : in    seFpgaAddr_type;
68       -- OUTPUT REGISTER PORTS
69       api_o_clk_out          : out   std_logic;
70       api_o_rfd_in           : in    seFlag_type;
71       api_o_tgt_slot_out     : out   seSlotAddr_type := (others => '0');
72       api_o_tgt_fpga_out     : out   seFpgaAddr_type := (others => '0');
73       api_o_tgt_reg_out      : out   seRegAddr_type  := (others => '0');
74       api_o_tgt_cmd_out      : out   seCmd_type      := CMD_WR;
75       api_o_src_reg_out      : out   seRegAddr_type  := (others => '0');
76       api_o_src_cmd_out      : out   seCmd_type      := CMD_WR;
77       api_o_data_out         : out   seData_type     := (others => '0');
78       api_o_wr_en_out        : out   seFlag_type     := '0';
79       -- INPUT REGISTER PORTS
80       api_i_clk_out          : out   std_logic;
81       api_i_src_slot_in      : in    seSlotAddr_type;
82       api_i_src_fpga_in      : in    seFpgaAddr_type;
83       api_i_src_reg_in       : in    seRegAddr_type;
84       api_i_src_cmd_in       : in    seCmd_type;
85       api_i_tgt_reg_in       : in    seRegAddr_type;
86       api_i_tgt_cmd_in       : in    seCmd_type;
87       api_i_data_in          : in    seData_type;
```

```
      api_i_empty_in         : in    seFlag_type;
      api_i_am_empty_in      : in    seFlag_type;
      api_i_rd_en_out        : out   seFlag_type     := '0'
   );
end entity pingpong_main;

architecture pingpong_main_behave of pingpong_main is

   -- Define some local signals because
   -- we can not read from the out-ports.
   signal api_i_rd_en    : seFlag_type       := '0';
   signal api_o_wr_en    : seFlag_type       := '0';

begin

   -- When defining an own clock domain to
   -- run the user design with a different
   -- clock, the following two lines should
   -- probably be altered
   api_i_clk_out <= api_clk_in;
   api_o_clk_out <= api_clk_in;

   -- route the internal signals to the out-ports
   api_i_rd_en_out <= api_i_rd_en;
   api_o_wr_en_out <= api_o_wr_en;

   -- A Spartan 3 FPGA has one LED and a Spartan 6 FPGA
   -- has two LEDs for debugging purposes.
   -- Set these LEDs disabled here. Comment following
   -- line and use this signal anywhere, you want to!
   api_led_out <= (others => '0');

   ping_pong : process
   begin
      wait until rising_edge(api_clk_in);

      -- Do not read anything by default.
      api_i_rd_en <= '0';

      -- Do not write anything by default.
      api_o_wr_en <= '0';

      -- Set the answer's target slot-,
      -- fpga- and register-addresses.
      api_o_tgt_slot_out <= api_i_src_slot_in;
      api_o_tgt_fpga_out <= api_i_src_fpga_in;
      api_o_tgt_reg_out  <= api_i_src_reg_in;
      api_o_tgt_cmd_out  <= api_i_src_cmd_in;
      api_o_src_reg_out  <= api_i_tgt_reg_in;
      api_o_src_cmd_out  <= api_i_tgt_cmd_in;

      -- We always pingpong the incoming words.
      -- In others words: We take the incoming
      -- word and copy it to the output register.
      api_o_data_out <= api_i_data_in;

      if (api_rst_in = '1') then
         -- While user_rst_out is high, the api
         -- is not ready for use.
      else
         if (api_i_empty_in = '0'
               and api_i_rd_en = '0'
               and api_o_rfd_in = '1') then
            -- There is a new word
            -- and this word has not been read last clock cycle
            -- and the API is ready for data (rfd) o be written.

            if (api_i_tgt_cmd_in = CMD_WR) then
               -- Only for target command CMD_WR
               -- pingpong the incoming word.

               -- Acknowledge incoming data.
               api_i_rd_en <= '1';

               -- Tell the API there is a
               -- new word to be written.
               api_o_wr_en <= '1';
            else
               -- In this example, we do not handle different
               -- target commands than CMD_WR, so we just
               -- discard the incoming word.
               api_i_rd_en <= '1';
            end if;
         end if;
      end if;
   end process ping_pong;


end architecture pingpong_main_behave;
```

4 Host C Sources

4.1 pingpong.c

```
1  /*
2   * Project: PingPong
3   * File:    pingpong.c
4   * Date:    Thu Nov 22 16:25:08 CET 2012
5   * Author:  dsi
6   *
7   * Description:
8   * Insert your project desciption here...
9   */
10 #include <stdio.h>
11 #include <stdlib.h>
12 #include <unistd.h>
13 #include <string.h>
14 #include <getopt.h>
15 #include <SeHostAPI.h>
16
17 #define  API_ERROR_CHECK(x)   {SE_STATUS rc = x; if(rc != SeApiSuccess) {
       printf("ERROR! SciEngines API returned %d (= %s)! \n", rc,
       se_status2str(rc)); exit(EXIT_FAILURE);} else { printf("SUCCESS!\n");
        }}
18
19 extern const char *__progname;
20 static const char *PROGRAM_NAME = "pingpong";
21 static const char *PROGRAM_VERSION = "1.91.00";
22 static const char *COPYRIGHT_TXT = "Copyright (c) 2013, SciEngines GmbH";
23
24 const char* BIT_FILE_S3 = "../../../fpga/vhdl/xc3s5000-4fg676/pingpong_top
       .bit";
25 const char* BIT_FILE_S6 = "../../../fpga/vhdl/xc6slx150-3fgg676/
       pingpong_top.bit";
26
27 static void printUsage() {
28    printf("Usage: %s [-options] [BIT_FILE]\n", __progname);
29    printf("   \n");
30    printf("where options include:\n");
31    printf("   -h --help           print this help and exit\n");
32    printf("   -v --version        print product version and exit\n");
33 }
34
35 static void printVersion() {
36    printf("%s version %s\n", PROGRAM_NAME, PROGRAM_VERSION);
37    printf("%s\n", COPYRIGHT_TXT);
38 }
39
40 static int run_program(se_machine_t machine, const char* bitFile) {
41    SE_STATUS   retval = EXIT_SUCCESS;
42    se_slot_t   cslot;
43    se_fpga_t   cfpga;
44    SE_SLOTINFO pSLOTInfo;
45    SE_FPGAINFO pFPGAInfo;
46    SE_ADDR     addr;          /* address structure   */
47    se_slot_t   slotCount;     /* number of slots     */
48    se_fpga_t   fpgaCount;     /* number of fpgas     */
49    __uint64_t  receive;       /* received data word */
50    __uint64_t  send;          /* send data word      */
51
52    printf("Running example for SciEngines Rivyera Host-API v %d.%d.%d (%s)
           ...\n\n",
53           SE_API_VERSION_MAJOR, SE_API_VERSION_MINOR, SE_API_VERSION_SP,
54           SE_API_VERSION_REVISION);
55
56    if (machine >= se_getMachineCount()) {
57       printf("Error: No such index: %d\n", machine);
58       exit(EXIT_FAILURE);
59    }
60
61    printf("Allocating device %d: ", machine);
62    API_ERROR_CHECK(se_allocMachine(machine, NULL))
63
64    /* determine how many slots our machine has */
65    API_ERROR_CHECK(se_getSlotCount(machine, &slotCount))
66    printf("%d card(s) found.\n", slotCount);
67
68    /* set address to all slots, all FPGAs. */
69    addr.slot  = SE_ADDR_SLOT_ALL;
70    addr.fpga  = SE_ADDR_FPGA_ALL;
71    addr.reg   = 0;
72    addr.contr = 0;
73    printf("Programming fpgas: ");
74    API_ERROR_CHECK(se_program(machine, &addr, bitFile, 1000))
75
76    /* now we ping each fpga... */
77    send = 123;
78    for (cslot = 0; cslot < slotCount; cslot++) {
79       addr.slot = cslot;
80       se_getSlotInfo(machine, cslot, &pSLOTInfo);
81       fpgaCount = pSLOTInfo.fpgaCount;
82       for (cfpga = 0; cfpga < fpgaCount; cfpga++) {
83          addr.fpga = cfpga;
```

```
84          API_ERROR_CHECK(se_getFPGAInfo(machine, &addr, &pFPGAInfo))
85          addr.slot = cslot;
86          addr.fpga = cfpga;
87          printf("Writing value %ld to [machine %d, contr %d, slot %d, fpga
                %d, reg %d]: ", send, machine, addr.contr, addr.slot, addr.
                fpga, addr.reg);
88          API_ERROR_CHECK(se_write(machine, &addr, &send, 1, NULL, 1000))
89
90          printf("Wait for data: ");
91          API_ERROR_CHECK(se_waitForData(machine, 0, &addr, NULL, 1000))
92
93          printf("Passively reading 32 words from [machine %d, contr %d,
                slot %d, fpga %d, reg %d]: ", machine, addr.contr, addr.slot
                , addr.fpga, addr.reg);
94          API_ERROR_CHECK(se_read(machine, &addr, &receive, 1,
                SeReadPassive,  NULL, 1000))
95
96          printf("Read value %ld.\n", receive);
97
98          if (addr.slot != cslot) {
99             fprintf(stderr, "addr.slot != cslot\n");
100            retval = EXIT_FAILURE;
101         } else if (addr.fpga != cfpga) {
102            fprintf(stderr, "addr.fpga != cfpga\n");
103            retval = EXIT_FAILURE;
104         } else if (send != receive) {
105            fprintf(stderr, "send != receive\n");
106            retval = EXIT_FAILURE;
107         }
108         send++;
109      }
110   }
111
112   printf("Deprogramming fpgas: ");
113   addr.slot  = SE_ADDR_SLOT_ALL;
114   addr.fpga  = SE_ADDR_FPGA_ALL;
115   API_ERROR_CHECK(se_deprogram(machine, &addr))
116
117   printf("Freeing device %d: ", machine);
118   API_ERROR_CHECK(se_freeMachine(machine))
119
120   exit(retval);
121}
122
123int main(int argc, char* const argv[] ) {
124   /* getopt_long stores the option index here. */
125   int option_index = 0;
126   char c;
127   const char *bit_file = BIT_FILE_S6;
128
129   static struct option long_options[] = {
130         {"help",          no_argument,        NULL, 'h'},
131         {"version",       no_argument,        NULL, 'v'},
132         {"timestamp",     no_argument,        NULL, 0},
133         {0, 0, 0, 0}
134   };
135
136   while ((c = getopt_long(argc, argv, "hv", long_options, &option_index))
          != -1) {
137      switch (c) {
138      case 'h':
139         printUsage();
140         exit(EXIT_SUCCESS);
141         break;
142      case 'v':
143         printVersion();
144         exit(EXIT_SUCCESS);
145         break;
146      case 0:
147         if (strcmp("timestamp", long_options[option_index].name) == 0) {
148            printf("%s %s\n", __DATE__, __TIME__);
149            exit(EXIT_SUCCESS);
150         }
151         break;
152      default:
153         exit(EXIT_FAILURE);
154         break;
155      }
156   }
157
158   if (optind + 1 < argc) {
159      fprintf(stderr, "Unexpected argument: \"%s\"\n", argv[optind + 1]);
160      exit(EXIT_FAILURE);
161   } else {
162      if (optind < argc) {
163         bit_file = argv[optind];
164      }
165      exit(run_program(0, bit_file));
166   }
167
168   return 0;
169}
```

Thank you for choosing an original SciEngines product.

Imprint
Responsible for content:

Firm SciEngines GmbH

Street Fraunhoferstr 13

ZIP D-24118

City Kiel

Country Germany

Phone +49 431 5302 482

Email info@sciengines.com

WWW http://www.sciengines.com

CEO Gerd Pfeiffer

Commercial Register Amtsgericht Kiel

Commercial Register No. HR B 9565 KI

VAT- Identification Number DE 814955925