RIVYERA

reconfigure versatilely your efficient raw architecture

PingPong

Version 2.0.1.0

SciEngines GmbH
Am Kiel-Kanal 2
24106 Kiel
Germany

www.sciengines.com

Revision: 1.92.02 December 10, 2015

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 the 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 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

The PingPong example took the well known network tool *ping* as prototype. Similar to the ping network tool, this example's FPGA design takes an incoming data packet and writes its payload to its source address. The communication flow is illustrated in figure 1. This example shows how to realize simple communication between host and FPGA. As noted in the introduction, the example uses the FPGA's autonomous write feature, where the host does not request a data packet, actively. Instead the host simply uses the SciEngines API's `se_read()` function with mode `SeReadPassive`. Using this *passive read* mechanism, `se_read()` blocks until either the given timeout is over or the specified number of 64 bit words are read.
Please refer to the *ActiveRead* example to learn about the *active read* mechanism.

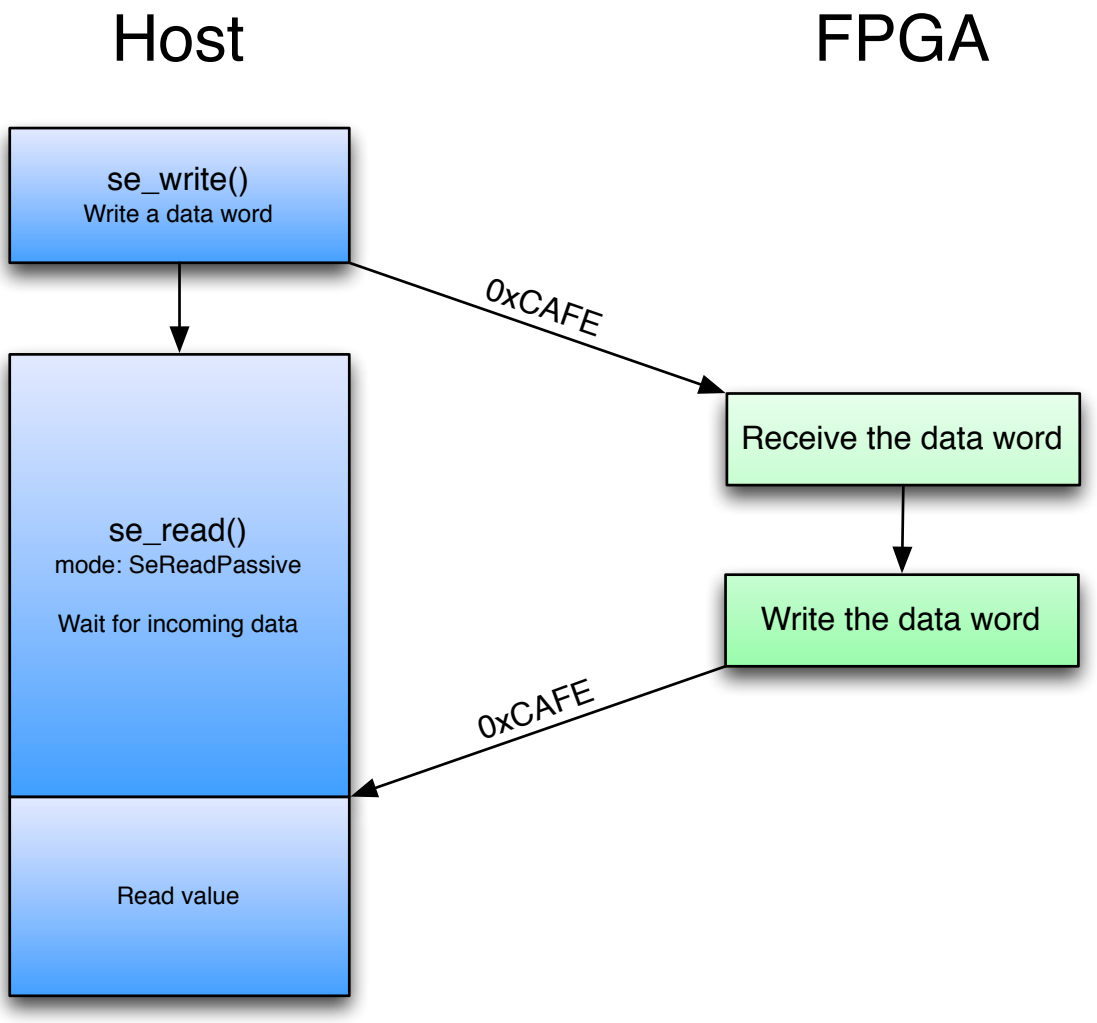

Figure 1: Communication flow.

2.1 FPGA Design

The FPGA design's key features are reading an incoming data packet and reply its payload value to the data packet's source address (see pingpong_main.vhd in chapter 3.1).

While there is no incoming data packet, `api_i_empty_in` remains '1' and the if statement in line 130 is false. That means that there is no data packet sent.
Lets assume there is a single incoming data packet with payload value `0xCAFE`, sent from the host via the first controller (controller address `0`) to target register address 5. Further lets assume the first controller is the first card (slot address `0`).

In the first clock cycle this will happen:

The if statement in line 130 is true (`api_i_empty_in` equals `'0'` and indicates an incoming data packet). Also, `api_i_tgt_cmd_in` is equal to `CMD_WR` indicating a regular data packet. Thus the if statement in line 135 is true. The source slot address stored in `api_i_src_slot_in` has value `0` since the host sent the data packet via controller `0` which is in slot `0`. The source FPGA address `api_i_src_fpga_in` is equal to the SciEngines API's special host constant `SE_ADDR_FPGA_HOST` since the data packet's source is the host. The source register address `api_i_src_reg_in` is set to 0.[1] Also, the source command `api_i_src_cmd_in` is equal to `CMD_WR`.[2] The target command `api_i_tgt_cmd_in` is also equal to `CMD_WR` since it is a regular data packet. `api_i_tgt_reg_in` is equal to `5` because we assumed the target register to be 5. Finally the `api_i_data_in` holds the 64 bit payload value `0xCAFE`. By default the incoming payload value is stored to the outgoing data packet port in line 124. In line 146 the incoming data packet is acknowledged by setting `api_i_rd_en` to 1 and in line 142 the value stored to `api_o_data_out` is written to the target defined by setting the `api_o_tgt_`-ports.

In the second clock cycle:

The incoming data packet is now being acknowledged. The if statement in line 130 is false due to the `api_i_rd_en` signal being `'1'`. So the default assignment in line 107 resets the `api_i_rd_en` to `'0'` again. In the next clock cycle the process is ready to handle another data packet.

2.2 Host Design

The host design's key features are (see pingpong.c in chapter 4.1 and PingPong.java in chapter 5.1):

1. One payload 0xCAFE is sent to FPGA 0 in slot 5, register address 5.
2. Read one data word from FPGA 0 in slot 5, register 5 using mode `SeReadPassive`.

The communication that occurs when executing this example is represented in detail in figure 2.

```
# writing word 0xCAFE to FPGA 0 in slot 0, register 5
src [s0 fH r0 cWR]    >>> tgt [s0 f0 r5 cWR]: 0x000000000000cafe
# passively reading 1 word from FPGA 0 in slot 0, register 5
tgt [s0 fH r0 cWR] <<<    src [s0 f0 r5 cWR]: 0x000000000000cafe
```

Figure 2: This is a communication log for the example execution. Within the squared brackets the *s* stands for *slot*, *f* for *fpga*, *r* for *register*, *c* for *command*, *WR* for *CMD_WR*, *RD* for *CMD_RD* and *h* for *host*.

[1] If the host is a data packet's source, `api_i_src_reg_in` is always equal to `0`.
[2] If the host is a data packet's source, `api_i_src_cmd_in` is always equal to `CMD_WR`.

Note that in this very simple example the FPGA and Slot addresses are hardcoded. The SciEngines-host-API functions `se_getSlotCount()` and `se_getFPGACount()` might be useful for setting addresses dynamically.

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:  Daniel Siebert (SciEngines GmbH)
6  | --
7  | -- Description:
8  | -- This is the PingPong example project.
9  | --
10 | -- Copyright (c) 2012-2015, SciEngines GmbH
11 | -- All rights reserved.
12 | --
13 | -- Redistribution in source or binary forms, with or without modification,
14 | -- is not permitted without specific prior written permission.
15 | --
16 | -- THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
17 | -- "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
18 | -- LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
19 | -- A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
20 | -- OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
21 | -- SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
22 | -- LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
23 | -- DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
24 | -- THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
25 | -- (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
26 | -- OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
27 | ---
28 |library ieee;
29 |use work.sciengines_api_types.all;
30 |use ieee.STD_LOGIC_1164.all;
31 |
32 |entity pingpong_main is
33 |   generic (
34 |      NUM_LEDS               : integer
35 |   );
36 |   port (
37 |      ------------------------------
38 |      --------- API PORTS ---------
39 |      ------------------------------
40 |      -- USER PORTS
41 |      api_clk_in             : in    std_logic;
42 |      api_rst_in             : in    std_logic;
43 |      api_led_out            : out   seBusFlag_type(NUM_LEDS-1 downto 0) := (others => '0');
44 |      api_hw_rev_in          : in    seHwRev_type;
45 |      -- ADDRESS PORTS
46 |      api_self_contr_in      : in    seFlag_type;
47 |      api_next_contr_in      : in    seSlotAddr_type;
48 |      api_prev_contr_in      : in    seSlotAddr_type;
49 |      api_self_slot_in       : in    seSlotAddr_type;
50 |      api_self_fpga_in       : in    seFpgaAddr_type;
51 |      -- OUTPUT REGISTER PORTS
52 |      api_o_clk_out          : out   std_logic;
53 |      api_o_rfd_in           : in    seFlag_type;
54 |      api_o_tgt_slot_out     : out   seSlotAddr_type := (others => '0');
55 |      api_o_tgt_fpga_out     : out   seFpgaAddr_type := (others => '0');
56 |      api_o_tgt_reg_out      : out   seRegAddr_type  := (others => '0');
57 |      api_o_tgt_cmd_out      : out   seCmd_type      := CMD_WR;
58 |      api_o_src_reg_out      : out   seRegAddr_type  := (others => '0');
59 |      api_o_src_cmd_out      : out   seCmd_type      := CMD_WR;
60 |      api_o_data_out         : out   seData_type     := (others => '0');
61 |      api_o_wr_en_out        : out   seFlag_type     := '0';
62 |      -- INPUT REGISTER PORTS
63 |      api_i_clk_out          : out   std_logic;
64 |      api_i_src_slot_in      : in    seSlotAddr_type;
65 |      api_i_src_fpga_in      : in    seFpgaAddr_type;
66 |      api_i_src_reg_in       : in    seRegAddr_type;
67 |      api_i_src_cmd_in       : in    seCmd_type;
68 |      api_i_tgt_reg_in       : in    seRegAddr_type;
69 |      api_i_tgt_cmd_in       : in    seCmd_type;
70 |      api_i_data_in          : in    seData_type;
71 |      api_i_empty_in         : in    seFlag_type;
72 |      api_i_am_empty_in      : in    seFlag_type;
73 |      api_i_rd_en_out        : out   seFlag_type     := '0'
74 |   );
75 |end entity pingpong_main;
76 |
77 |architecture pingpong_main_behave of pingpong_main is
78 |
79 |   -- Define some local signals because
80 |   -- we can not read from the out-ports.
```

```
81  |    signal api_i_rd_en    : seFlag_type       := '0';
82  |    signal api_o_wr_en    : seFlag_type       := '0';
83  |
84  |begin
85  |
86  |    -- When defining an own clock domain to
87  |    -- run the user design with a different
88  |    -- clock, the following two lines should
89  |    -- probably be altered
90  |    api_i_clk_out <= api_clk_in;
91  |    api_o_clk_out <= api_clk_in;
92  |
93  |    -- route the internal signals to the out-ports
94  |    api_i_rd_en_out <= api_i_rd_en;
95  |    api_o_wr_en_out <= api_o_wr_en;
96  |
97  |    -- A Spartan 6 FPGA has two LEDs for debugging purposes.
98  |    -- Set these LEDs disabled here. Comment following
99  |    -- line and use this signal anywhere you want to!
100 |    api_led_out <= (others => '0');
101 |
102 |    ping_pong : process
103 |    begin
104 |       wait until rising_edge(api_clk_in);
105 |
106 |       -- Do not read anything by default.
107 |       api_i_rd_en <= '0';
108 |
109 |       -- Do not write anything by default.
110 |       api_o_wr_en <= '0';
111 |
112 |       -- Set the answer's target slot-,
113 |       -- fpga- and register-addresses.
114 |       api_o_tgt_slot_out <= api_i_src_slot_in;
115 |       api_o_tgt_fpga_out <= api_i_src_fpga_in;
116 |       api_o_tgt_reg_out  <= api_i_src_reg_in;
117 |       api_o_tgt_cmd_out  <= api_i_src_cmd_in;
118 |       api_o_src_reg_out  <= api_i_tgt_reg_in;
119 |       api_o_src_cmd_out  <= api_i_tgt_cmd_in;
120 |
121 |       -- We always pingpong the incoming payload.
122 |       -- In others words: We take the incoming data packet and
123 |       -- copy its payload to the API's outgoing data packet port.
124 |       api_o_data_out <= api_i_data_in;
125 |
126 |       if (api_rst_in = '1') then
127 |          -- While user_rst_in is high, the api
128 |          -- is not ready for use.
129 |       else
130 |          if (api_i_empty_in = '0'
131 |                and api_i_rd_en = '0') then
132 |             -- There is a new incoming data packet
133 |             -- and this data packet has not been read last clock cycle
134 |
135 |             if (api_i_tgt_cmd_in = CMD_WR) then
136 |                -- Only for target command CMD_WR
137 |                -- pingpong the incoming payload.
138 |
139 |                -- if the API is ready for data (rfd),
140 |                -- tell the API there is a
141 |                -- new data packet to be written
142 |                api_o_wr_en <= api_o_rfd_in;
143 |
144 |                -- Acknowledge the incoming data packet
145 |                -- if writing to the API is allowed (rfd)
146 |                api_i_rd_en <= api_o_rfd_in;
147 |             else
148 |                -- In this example, we do not handle different
149 |                -- target commands than CMD_WR, so we just
150 |                -- discard the incoming data packet.
151 |                api_i_rd_en <= '1';
152 |             end if;
153 |          end if;
154 |       end if;
155 |    end process ping_pong;
156 |
157 |
158 |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:  Daniel Siebert (SciEngines GmbH)
6   *
7   * Description:
8   * This is the PingPong example project.
9   *
10  * Copyright (c) 2012-2015, SciEngines GmbH
11  * All rights reserved.
12  *
13  * Redistribution in source or binary forms, with or without modification,
14  * is not permitted without specific prior written permission.
15  *
16  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
17  * "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
18  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
19  * A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
20  * OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
21  * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
22  * LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
23  * DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
24  * THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
25  * (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
26  * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
27  */
28 #include <stdio.h>
29 #include <stdlib.h>
30 #include <unistd.h>
31 #include <string.h>
32 #include <getopt.h>
33 #include <SeHostAPI.h>
34
35 #define  API_ERROR_CHECK(x)   {SE_STATUS rc = x; if(rc != SeApiSuccess) { printf("ERROR! SciEngines
       API returned %u (= %s)! \n", rc, se_status2str(rc)); exit(EXIT_FAILURE);} else { printf("SUCCESS
       !\n"); }}
36
37 extern const char *__progname;
38 static const char *PROGRAM_NAME = "pingpong";
39 static const char *PROGRAM_VERSION = "1.92.02";
40 static const char *COPYRIGHT_TXT = "Copyright (c) 2012-2015, SciEngines GmbH";
41
42 const char* BIT_FILE = "../../fpga/vhdl/xc6slx150-3fgg676/pingpong_top.bit";
43 const char* SIM_FILE = "../../fpga/vhdl/xc6slx150-3fgg676/pingpong.sim";
44
45 static void printUsage() {
46    printf("Usage: %s [-options] [BIT_FILE]\n", __progname);
47    printf("   \n");
48    printf("where options include:\n");
49    printf("   -h --help          print this help and exit\n");
50    printf("   -v --version       print product version and exit\n");
51    printf("   -m --machine       run your program on a specific machine\n");
52 }
53
54 static void printVersion() {
55    printf("%s version %s\n", PROGRAM_NAME, PROGRAM_VERSION);
56    printf("%s\n", COPYRIGHT_TXT);
57 }
58
59 static int run_program(se_machine_t machine, const char* bit_file) {
60    SE_STATUS         retval = EXIT_SUCCESS;
61    SE_ADDR           addr;         /* address structure  */
62    __uint64_t        receive;      /* received data word */
63    __uint64_t        send;         /* send data word     */
64    SE_CONTROLLERINFO controllerInfo;
65
66    printf("Running PingPong with SciEngines RIVYERA Host-API v %02u.%02u.%02u (%s)...\n\n",
67          SE_API_VERSION_MAJOR, SE_API_VERSION_MINOR, SE_API_VERSION_SP,
68          SE_API_VERSION_REVISION);
69
70    if (machine >= se_getMachineCount()) {
71       printf("Error: No such index: %u\n", machine);
72       return EXIT_FAILURE;
73    }
74
75    printf("Allocating machine %u: ", machine);
76    API_ERROR_CHECK(se_allocMachine(machine, NULL))
77
78    /* set address to all slots, all FPGAs. */
```

```
    addr.contr = 0;
    addr.slot  = SE_ADDR_SLOT_ALL;
    addr.fpga  = SE_ADDR_FPGA_ALL;
    addr.reg   = 0;

    printf("Getting␣controller␣information␣for␣machine␣%u:␣", machine);
    API_ERROR_CHECK(se_getControllerInfo(machine, addr.contr, &controllerInfo));
    printf("Programming␣fpgas:␣");
    if (strcmp(controllerInfo.driver_name, "isim") == 0) {
       API_ERROR_CHECK(se_program(machine, &addr, SIM_FILE, 5000))
    } else if (bit_file) {
       API_ERROR_CHECK(se_program(machine, &addr, bit_file, 5000))
    } else {
       API_ERROR_CHECK(se_program(machine, &addr, BIT_FILE, 5000))
    }

    send = 0xcafe;
    addr.slot = 0;
    addr.fpga = 0;
    addr.reg  = 5;
    printf("Writing␣value␣0x%016lx␣to␣[machine␣%u,␣contr␣%u,␣slot␣%u,␣fpga␣%u,␣reg␣%u]:␣",
          send, machine, addr.contr, addr.slot, addr.fpga, addr.reg);
    API_ERROR_CHECK(se_write(machine, &addr, &send, 1, NULL, 5000))

    printf("Passively␣reading␣1␣word␣from␣[machine␣%u,␣contr␣%u,␣slot␣%u,␣fpga␣%u,␣reg␣%u]:␣",
          machine, addr.contr, addr.slot, addr.fpga, addr.reg);
    API_ERROR_CHECK(se_read(machine, &addr, &receive, 1, SeReadPassive,  NULL, 5000))
    printf("Read␣value␣is␣0x%016lx.\n", receive);

    printf("Deprogramming␣fpgas:␣");
    addr.slot  = SE_ADDR_SLOT_ALL;
    addr.fpga  = SE_ADDR_FPGA_ALL;
    API_ERROR_CHECK(se_deprogram(machine, &addr))

    printf("Freeing␣machine␣%u:␣", machine);
    API_ERROR_CHECK(se_freeMachine(machine))

    return retval;
}

int main(int argc, char* const argv[] ) {
    /* getopt_long stores the option index here. */
    int            option_index = 0;
    char           c;
    const char     *bit_file = NULL;
    se_machine_t   machine = 0;

    static struct option long_options[] = {
       {"help",         no_argument,        NULL, 'h'},
       {"version",      no_argument,        NULL, 'v'},
       {"machine",      required_argument,  NULL, 'm'},
       {"timestamp",    no_argument,        NULL, 1},
       {0, 0, 0, 0}
    };

    while ((c = getopt_long(argc, argv, "hvm:", long_options, &option_index)) != -1) {
       switch (c) {
       case 'h':
          printUsage();
          exit(EXIT_SUCCESS);
          break;
       case 'v':
          printVersion();
          exit(EXIT_SUCCESS);
          break;
       case 'm':
          if (sscanf(optarg, "%u", &machine) != 1){
             fprintf(stderr, "Could␣not␣read␣argument␣%s\n", optarg);
          }
          else{
          }
          break;
       case 1:
          printf("%s␣%s\n", __DATE__, __TIME__);
          exit(EXIT_SUCCESS);
          break;
       default:
          exit(EXIT_FAILURE);
          break;
       }
    }

    if (optind + 1 < argc) {
       fprintf(stderr, "Unexpected␣argument:␣\"%s\"\n", argv[optind + 1]);
       exit(EXIT_FAILURE);
    } else {
```

```
165 |       if (optind < argc) {
166 |          bit_file = argv[optind];
167 |       }
168 |       exit(run_program(machine, bit_file));
169 |    }
170 |
171 |    return 0;
172 | }
```

5 Host Java Sources

5.1 PingPong.java

```
  1 | import com.sciengines.rivyera.api.types.*;
  2 | import com.sciengines.rivyera.api.types.exceptions.*;
  3 | import static com.sciengines.rivyera.api.SciEngines_API.*;
  4 | import static com.sciengines.rivyera.api.SciEngines_API_Const.*;
  5 | import java.nio.ByteBuffer;
  6 | import java.nio.ByteOrder;
  7 |
  8 | /**
  9 |  * Project: PingPong
 10 |  * Date:    Thu Nov 22 16:25:08 CET 2012
 11 |  * @author  Daniel Siebert (SciEngines GmbH)
 12 |  *
 13 |  * Description:
 14 |  * This is the PingPong example project.
 15 |  *
 16 |  * Copyright (c) 2012-2015, SciEngines GmbH
 17 |  * All rights reserved.
 18 |  *
 19 |  * Redistribution in source or binary forms, with or without modification,
 20 |  * is not permitted without specific prior written permission.
 21 |  *
 22 |  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS
 23 |  * IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED
 24 |  * TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
 25 |  * PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER
 26 |  * OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
 27 |  * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
 28 |  * PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
 29 |  * PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
 30 |  * LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
 31 |  * NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
 32 |  * SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 33 |  */
 34 | public class PingPong {
 35 |
 36 |    private static final String PROGRAM_NAME = "PingPong";
 37 |    private static final String PROGRAM_VERSION = "1.92.02";
 38 |    private static final String COPYRIGHT_TXT = "Copyright␣(c)␣2012-2015,␣SciEngines␣GmbH";
 39 |
 40 |    private static final String BIT_FILE = "../../../fpga/vhdl/xc6slx150-3fgg676/pingpong_top.bit";
 41 |    private static final String SIM_FILE = "../../../fpga/vhdl/xc6slx150-3fgg676/pingpong.sim";
 42 |
 43 |    private static void printUsage() {
 44 |       System.out.println("Usage:␣java␣" + PROGRAM_NAME + "␣[-options]␣[BIT_FILE]");
 45 |       System.out.println("␣␣␣");
 46 |       System.out.println("where␣options␣include:");
 47 |       System.out.println("␣␣␣-h␣--help␣␣␣␣␣␣␣␣␣␣print␣this␣help␣and␣exit");
 48 |       System.out.println("␣␣␣-v␣--version␣␣␣␣␣␣␣␣print␣product␣version␣and␣exit");
 49 |       System.out.println("␣␣␣-m␣--machine␣␣␣␣␣␣␣␣run␣your␣program␣on␣a␣specific␣machine");
 50 |    }
 51 |
 52 |    static void printVersion() {
 53 |       System.out.println(PROGRAM_NAME + "␣version␣" + PROGRAM_VERSION);
 54 |       System.out.println(COPYRIGHT_TXT);
 55 |    }
 56 |
 57 |    public static int run_program(int machine, String bitFile) {
 58 |       int              retval = 0;
 59 |       SeOptions        options;
 60 |       SeAddress        addr;            /* address structure  */
 61 |       long             receive;         /* received data word */
 62 |       long             send;            /* send data word     */
 63 |       ByteBuffer       payload;
 64 |       SeControllerInfo controllerInfo;
 65 |
 66 |       options = new SeOptions(
 67 |               SeOptions.SeWriteBehavior.se_write_async,
```

```
            SeOptions.SeRoutingMethod.se_routing_normal);
    payload = ByteBuffer.allocateDirect(8)
            .order(ByteOrder.nativeOrder());
    System.out.printf("Running␣PingPong␣with␣SciEngines␣RIVYERA␣Host-API␣v␣%02d.%02d.%02d␣(%s)...\n
        \n",
            SE_API_VERSION_MAJOR, SE_API_VERSION_MINOR, SE_API_VERSION_SP, SE_API_VERSION_REVISION)
                ;

    if (machine >= se_getMachineCount()) {
       System.out.println("no␣such␣index:␣" + machine);
       System.exit(1);
    }

    try {
       System.out.print("Allocating␣machine␣" + machine + ":␣");
       se_allocMachine(machine, options);
       System.out.println("SUCCESS!");

       addr = new SeAddress(0, SE_ADDR_SLOT_ALL, SE_ADDR_FPGA_ALL, 0);
       System.out.print("Getting␣controller␣information␣for␣machine␣" + machine + ":␣");
       controllerInfo = se_getControllerInfo(machine, addr.contr);
       System.out.println("SUCCESS!");

       System.out.print("Programming␣fpgas:␣");
       if ("isim".equals(controllerInfo.getDriverName())) {
          se_program(machine, addr, SIM_FILE, 5000);
       } else if (bitFile != null) {
          se_program(machine, addr, bitFile, 5000);
       } else {
          se_program(machine, addr, BIT_FILE, 5000);
       }
       System.out.println("SUCCESS!");

       send = 0xcafe;
       addr.slot = 0;
       addr.fpga = 0;
       addr.reg  = 5;
       System.out.printf("Writing␣value␣0x%016x␣to␣[machine␣%d,␣contr␣%d,␣slot␣%d,␣fpga␣%d,␣reg␣%d
           ]:␣",
            send, machine, addr.contr, addr.slot, addr.fpga, addr.reg);
       payload.putLong(0, send);
       se_write(machine, addr, payload, 1, 5000);
       System.out.println("Success!");

       System.out.printf("Passively␣reading␣1␣word␣from␣[machine␣%d,␣contr␣%d,␣slot␣%d,␣fpga␣%d,␣
           reg␣%d]:␣",
            machine, addr.contr, addr.slot, addr.fpga, addr.reg);
       se_read(machine, addr, payload, 1, SE_READ_PASSIVE,
               1000);
       System.out.println("Success!");
       receive = payload.getLong(0);
       System.out.printf("Read␣value␣is␣0x%016x.\n", receive);

       System.out.print("Deprogramming␣fpgas:␣");
       addr.slot = SE_ADDR_SLOT_ALL;
       addr.fpga = SE_ADDR_FPGA_ALL;
       se_deprogram(machine, addr);
       System.out.println("SUCCESS!");

       System.out.print("Freeing␣machine␣" + machine + ":␣");
       se_freeMachine(machine);
       System.out.println("SUCCESS!");
    } catch(SeApiException e) {
       System.err.println("Failed:␣" + e.getMessage());
    }

    return retval;
 }

 public static void main(String[] args) {
    String   bitFile = null;
    int      machine = 0;

    for (int optind = 0; optind < args.length; optind++) {
       if (args[optind].equals("-h") || args[optind].equals("--help")) {
          printUsage();
          System.exit(0);
       } else if (args[optind].equals("-v") || args[optind].equals("--version")) {
          printVersion();
          System.exit(0);
       } else if (args[optind].equals("-m") || args[optind].equals("--machine")) {
          optind++;
          if (optind < args.length) {
             machine = Integer.parseInt(args[optind]);
          } else {
               System.err.println("Missing␣argument␣for␣option␣" + args[optind - 1]);
               System.exit(1);
```

```
151 |             }                                                                                                 |
152 |          } else if (bitFile == null) {                                                                        |
153 |             bitFile = args[optind];                                                                           |
154 |          } else {                                                                                             |
155 |             System.err.println("Unexpected␣argument␣\"" + args[optind] + "\"");                              |
156 |             System.exit(1);                                                                                   |
157 |          }                                                                                                    |
158 |       }                                                                                                       |
159 |       System.exit(run_program(machine, bitFile));                                                            |
160 |    }                                                                                                          |
161 |                                                                                                               |
162 | }                                                                                                             |
```

Thank you for choosing an original SciEngines product.

Imprint
Responsible for content:

Firm SciEngines GmbH

Street Am Kiel-Kanal 2

ZIP D-24106

City Kiel

Country Germany

Phone +49 431 9086200 0

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