RIVYERA

reconfigure versatilely your efficient raw architecture

ActiveRead

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.

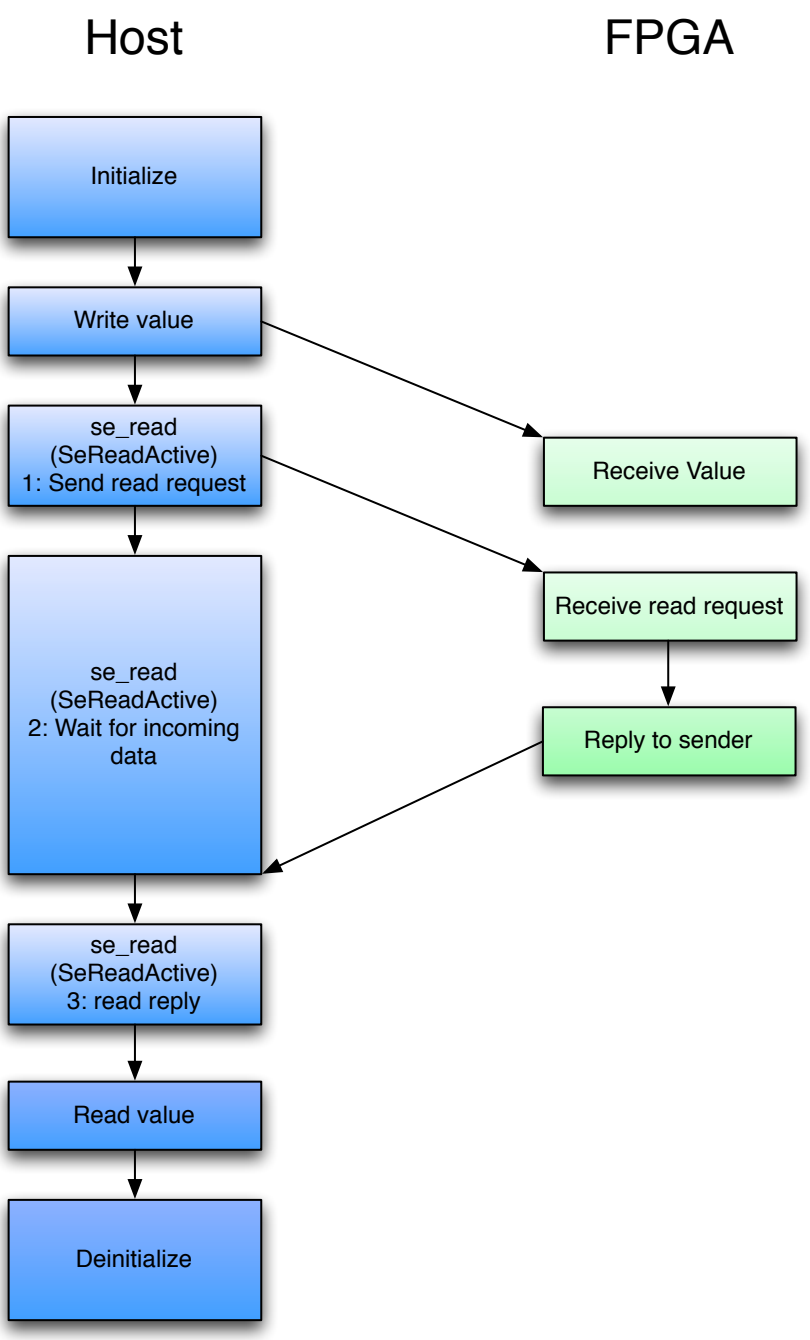
ActiveRead is a very simple communication example to get in touch with the active read mechanism which is a special mode for the `se_read()` function. The active read is used for requesting a number of words to an FPGA, waiting for the reply and reading it. In this example, the FPGA may be asked to write a arbitrary number of words to the requester. These words initially start with 0 and are incremented word by word. It is also possible to change the starting number by writing this starting number to an FPGA. It might be a good idea to have a look at the PingPong example first, before trying to understand this example.

2 RIVYERA Implementation

In contrast to the PingPong example, which uses the passive read mechanism, this example uses the active read.
The ActiveRead example is implemented as follows: After being programmed, an arbitrary number is written to the first FPGA on the first card. Note that in this very simple example the FPGA and Card indices are hard-coded, but you could of course determine certain valid FPGA and Card indices. Whenever an FPGA receives a number at register 0 (signaled by `api_i_empty_in` low and `api_i_tgt_reg_in = 0`), which has been sent using `se_write()` (signaled by `api_i_tgt_cmd_in = CMD_WR`), it sets this number as new starting number for the next active read. Afterwards the host issues an active read by using `se_read()` with mode `SeReadActive`. This active read is internally split up into three parts:

1. Writing a read request to an FPGA.
2. Waiting for data from the FPGA.
3. Reading the answer and return.

For an incoming read request to an FPGA (signaled by `api_i_tgt_cmd_in = CMD_RD`), the incoming data word refers to the number of requested words (`api_i_data_in`), the FPGA then writes this number of words to the requester.


Host
FPGA
Initialize
Write value
se_read
(SeReadActive)
1: Send read request
se_read
(SeReadActive)
2: Wait for incoming
data
se_read
(SeReadActive)
3: read reply
Read value
Deinitialize
Receive Value
Receive read request
Reply to sender

3 FPGA VHDL Sources

3.1 activeread_main.vhd

```
1  ---
2  -- Project: ActiveRead
3  -- File:    activeread_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 use ieee.std_logic_unsigned.all;
41 use ieee.std_logic_arith.all;
42
43 entity activeread_main is
44    generic (
45       NUM_LEDS               : integer
46    );
47    port (
48       ------------------------------
49       --------- API PORTS ---------
50       ------------------------------
51       -- USER PORTS
52       api_clk_in             : in    std_logic;
53       api_rst_in             : in    std_logic;
54       api_led_out            : out   seBusFlag_type(NUM_LEDS-1 downto 0)
             := (others => '0');
55       -- ADDRES PORTS
56       api_self_contr_in      : in    seFlag_type;
57       api_next_contr_in      : in    seSlotAddr_type;
58       api_prev_contr_in      : in    seSlotAddr_type;
59       api_self_slot_in       : in    seSlotAddr_type;
60       api_self_fpga_in       : in    seFpgaAddr_type;
61       -- OUTPUT REGISTER PORTS
62       api_o_clk_out          : out   std_logic;
63       api_o_rfd_in           : in    seFlag_type;
64       api_o_tgt_slot_out     : out   seSlotAddr_type := (others => '0');
65       api_o_tgt_fpga_out     : out   seFpgaAddr_type := (others => '0');
66       api_o_tgt_reg_out      : out   seRegAddr_type  := (others => '0');
67       api_o_tgt_cmd_out      : out   seCmd_type      := CMD_WR;
68       api_o_src_reg_out      : out   seRegAddr_type  := (others => '0');
69       api_o_src_cmd_out      : out   seCmd_type      := CMD_WR;
70       api_o_data_out         : out   seData_type     := (others => '0');
71       api_o_wr_en_out        : out   seFlag_type     := '0';
72       -- INPUT REGISTER PORTS
73       api_i_clk_out          : out   std_logic;
74       api_i_src_slot_in      : in    seSlotAddr_type;
75       api_i_src_fpga_in      : in    seFpgaAddr_type;
76       api_i_src_reg_in       : in    seRegAddr_type;
77       api_i_src_cmd_in       : in    seCmd_type;
78       api_i_tgt_reg_in       : in    seRegAddr_type;
79       api_i_tgt_cmd_in       : in    seCmd_type;
80       api_i_data_in          : in    seData_type;
81       api_i_empty_in         : in    seFlag_type;
82       api_i_am_empty_in      : in    seFlag_type;
83       api_i_rd_en_out        : out   seFlag_type     := '0'
84    );
85 end entity activeread_main;
86
87 architecture activeread_main_behave of activeread_main is
```

```vhdl

   -- 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';
   signal ar_current_word  : seData_type       := (others => '0');
   signal ar_counter       : seData_type       := (others => '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');

   activeread_process : 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;

      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
               -- this is an incoming word which has
               -- been written using se_write
               case api_i_tgt_reg_in is
                  when "000000" =>
                     -- this means, we set an incoming word as start
                     ar_current_word      <= api_i_data_in;
                     api_i_rd_en          <= '1';
                  when others =>
                     -- only handle register 0,
                     -- discard incoming word
                     api_i_rd_en          <= '1';
               end case;
            elsif (api_i_tgt_cmd_in = CMD_RD) then
               -- this is an incoming word that has been
               -- written using se_read with mode SeReadActive
               -- or mode SeReadRequest
               case api_i_tgt_reg_in is
                  when "000000" =>
                     if (ar_counter = api_i_data_in) then
                        ar_counter        <= (others => '0');
                        api_i_rd_en       <= '1';
                     else
                        ar_current_word   <= ar_current_word + 1;
                        api_o_data_out    <= ar_current_word;
                        api_o_wr_en       <= '1';
                        ar_counter        <= ar_counter + 1;
                     end if;
                  when others =>
                     -- only handle register 0,
                     -- discard incoming word
                     api_i_rd_en          <= '1';
               end case;
            else
               -- This should never happen, but if it does,
               -- we just, discard the incoming word.
               api_i_rd_en <= '1';
```

```
            end if;
         end if;
      end if;
   end process activeread_process;


end architecture activeread_main_behave;
```

4 Host C Sources

4.1 activeread.c

```
/*
 * Project: ActiveRead
 * File:    activeread.c
 * Date:    Thu Nov 22 16:25:08 CET 2012
 * Author:  dsi
 *
 * Description:
 * Insert your project desciption here...
 */
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <getopt.h>
#include <SeHostAPI.h>

#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");
      }}

extern const char *__progname;
static const char *PROGRAM_NAME = "activeread";
static const char *PROGRAM_VERSION = "1.91.00";
static const char *COPYRIGHT_TXT = "Copyright (c) 2013, SciEngines GmbH";

const char* BIT_FILE_S3 = "../../../fpga/vhdl/xc3s5000-4fg676/example_top.
     bit";
const char* BIT_FILE_S6 = "../../../fpga/vhdl/xc6slx150-3fgg676/
     example_top.bit";

static void printUsage() {
   printf("Usage: %s [-options] [BIT_FILE]\n", __progname);
   printf("   \n");
   printf("where options include:\n");
   printf("   -h --help           print this help and exit\n");
   printf("   -v --version        print product version and exit\n");
}

static void printVersion() {
   printf("%s version %s\n", PROGRAM_NAME, PROGRAM_VERSION);
   printf("%s\n", COPYRIGHT_TXT);
}

static int run_program(se_machine_t machine, const char* bitFile) {
   SE_STATUS   retval = EXIT_SUCCESS;
   SE_ADDR     addr;         /* address structure   */
   se_slot_t   slotCount;    /* number of slots     */
   __uint64_t  pReceive[32]; /* received data words */
   __uint64_t  send;         /* send data word      */
   size_t      tc;
   size_t      i;

   printf("Running example for SciEngines Rivyera Host-API v %d.%d.%d (%s)
        ...\n\n",
         SE_API_VERSION_MAJOR, SE_API_VERSION_MINOR, SE_API_VERSION_SP,
         SE_API_VERSION_REVISION);

   if (machine >= se_getMachineCount()) {
      printf("Error: No such index: %d\n", machine);
      exit(EXIT_FAILURE);
   }

   printf("Allocating device %d: ", machine);
   API_ERROR_CHECK(se_allocMachine(machine, NULL))

   /* determine how many slots our machine has */
   API_ERROR_CHECK(se_getSlotCount(machine, &slotCount))
   printf("%d card(s) found.\n", slotCount);

   /* set address to all slots, all FPGAs. */
   addr.slot  = SE_ADDR_SLOT_ALL;
   addr.fpga  = SE_ADDR_FPGA_ALL;
   addr.reg   = 0;
   addr.contr = 0;
   printf("Programming fpgas: ");
   API_ERROR_CHECK(se_program(machine, &addr, bitFile, 1000))
```

```
72 |
73 |    send = 80;
74 |
75 |    addr.slot = 0;
76 |    addr.fpga = 0;
77 |    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);
78 |    API_ERROR_CHECK(se_write(machine, &addr, &send, 1, NULL, 1000))
79 |
80 |    printf("Actively␣reading␣32␣words␣from␣[machine␣%d,␣contr␣%d,␣slot␣%d,␣
   |         fpga␣%d,␣reg␣%d]:␣", machine, addr.contr, addr.slot, addr.fpga,
   |         addr.reg);
81 |    API_ERROR_CHECK(se_read(machine, &addr, pReceive, 32, SeReadActive,  &
   |         tc, 1000))
82 |
83 |    for (i = 0; i < tc; i++) {
84 |       printf("Received␣value␣at␣index␣%-2ld:␣%ld.\n", i, pReceive[i]);
85 |    }
86 |
87 |    printf("Deprogramming␣fpgas:␣");
88 |    addr.slot  = SE_ADDR_SLOT_ALL;
89 |    addr.fpga  = SE_ADDR_FPGA_ALL;
90 |    API_ERROR_CHECK(se_deprogram(machine, &addr))
91 |
92 |    printf("Freeing␣device␣%d:␣", machine);
93 |    API_ERROR_CHECK(se_freeMachine(machine))
94 |
95 |    exit(retval);
96 |}
97 |
98 |int main(int argc, char* const argv[] ) {
99 |    /* getopt_long stores the option index here. */
100|    int option_index = 0;
101|    char c;
102|    const char *bit_file = BIT_FILE_S6;
103|
104|    static struct option long_options[] = {
105|          {"help",          no_argument,        NULL, 'h'},
106|          {"version",       no_argument,        NULL, 'v'},
107|          {"timestamp",     no_argument,        NULL, 0},
108|          {0, 0, 0, 0}
109|    };
110|
111|    while ((c = getopt_long(argc, argv, "hv", long_options, &option_index))
   |          != -1) {
112|       switch (c) {
113|       case 'h':
114|          printUsage();
115|          exit(EXIT_SUCCESS);
116|          break;
117|       case 'v':
118|          printVersion();
119|          exit(EXIT_SUCCESS);
120|          break;
121|       case 0:
122|          if (strcmp("timestamp", long_options[option_index].name) == 0) {
123|             printf("%s␣%s\n", __DATE__, __TIME__);
124|             exit(EXIT_SUCCESS);
125|          }
126|          break;
127|       default:
128|          exit(EXIT_FAILURE);
129|          break;
130|       }
131|    }
132|
133|    if (optind + 1 < argc) {
134|       fprintf(stderr, "Unexpected␣argument:␣\"%s\"\n", argv[optind + 1]);
135|       exit(EXIT_FAILURE);
136|    } else {
137|       if (optind < argc) {
138|          bit_file = argv[optind];
139|       }
140|       exit(run_program(0, bit_file));
141|    }
142|
143|    return 0;
144|}
```

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