Partial reconfiguration and fault simulation with Altera Cyclone V FPGA

Students project at the chair of computer architecture university freiburg

Markus Weiß

February 12, 2016

Contents

1 HW/SW for this project

The Hardware and Software for both partial projects do not differ, so here is a short summary of the used hard- and software.

1.1 Hardware

The hardware used in this project were a "Terasic DE1-SoC Development Kit". This development kit consists of a FPGA, named Altera Cyclone V FPGA (5SCEMA5F31C6N), a dual-core Cortex A9 processor, also named Hard-Processor-System (HPS),memories and interfaces (USB,JTAG,i2c). The communication interfaces are a onboard USB-Blaster II, USB 2.0 ports, UART, ethernet port and extended module for gpio pins.
The fpga is the heart of the development kit and the mainly used component of the project. Here are some facts about the fpga Cyclone V:

- 85000 logic cells
- 4450 kb embedded memory
- access to an external 64 MB static random access memory (SRAM)
- low energy consumption
- standard clock frequency is 50 MHz

The Cyclone V is configured with a JTAG interface, where a bitstream is loaded to the board. This bitstream is lost once the energy is off. Also available is the option to program an in-system programmable NOR flash memory, which safes the bitstream while power is taken off. Another way to program the fpga is to use the HPS. This is done while the boot progress with "U-Boot" or the "preloader" but it also works if the operating system is fully booted.
The **preloader** initiates the clock, multiplexing of pins, configuration of the main memory and load the U-Boot. The configuration of the main memory of the HPS and the AXI Bridges is important, because these have to grant the fpga access to this.
The **universal boot loader** initiates and tests hardware components and also download and execute application software. This bootloader is required to initiate the boot process of the unix-kernel.
The **Avalon-Memory-Mapped-Interface** enables efficient read- and write operations, interrupts, clocks, resets and control progresses. This Avalon-MM-Interface enables adress based read- and write procedures in a master/slave connection. The master put a request to the slave, which read or write special adresses of the memory. This interface is required to get access from the FPGA to the main memory of the HPS.
To work with an external C/C++ programm an operation system for the hps is needed. The chosen os is a linux system.

1.2 Software

The software used in this project were Quartus II Version 14.1/15.0 and QSYS on Windows 7/8.1.
Quartus II is needed to generate VHDL- or Verilog programs which can be used by a FPGA. The software can be configured to reach different goals, e.g. time complexity, speed, optimization of compilation time, energy consumption etc.
QSYS is described as a tool for system integration to safe time, therefore cost, of developer by generating automatic connectionlogic of different components. The developed VHDL code from Quartus II software can be imported to QSYS to connect different components, e.g. on the DE-SoC1. QSYS has to initiate, configure and connect the HPS to the FPGA if they have to communication between each other.

2 Partial reconfiguration

2.1 Motivation

The idea of reconfigurating a fpga while the rest of it is still running is highly desirable. With this feature it is easier to change designs or improve the the functionality without stopping its working progress. Some applications need to change some logic while another part is not allowed to stop, e.g. the flow of data. Therefore this function is desirable in the field of communication systems and high-performance computing platforms. A benefit of this idea is that the number of devices can be scaled down, therefore the power consumption and of course the cost. The tasks another fpga has handled before can now be handled by the same fpga, only another part of the logic cells take this.

Thanks to this partial reconfiguration the fpga can handle a new input while another part still executes its commands. This reconfiguration is initialised by an external processor on the development kit and programs the desired locig cells of the fpga while the other part remains in its process.

The intiation is done by an internal host which runs a C/C++ programm. This C/C++ programm is stored in the RAM (read access memory) of the development kit board.

2.2 Implementation

The idea of the project were to implement an internal host, which initialises the reconfiguration with a c/c++ program. To do so, the internal host has to communicate with the processor and fpga - where is the internal host stored?- The communication between HPS and FPGA, the main memory and in- and output interfaces is an important point. There are different options for the communication between HPS and FPGA.

1. Advanced extensible interface bus bridges (AXI)
    a) Lightweight HPS-to-FPGA bridge
    b) HPS-to-FPGA bridge

c) FPGA-to-HPS bridge

2. FPGA-to-SDRAM

FPGA access the memory and load the bitstream safed in the sdram?? **which option is used in our project? which one should be used?**

The FPGA development process is done by Quartus II and QSYS. Quartus II creates the FPGA components in VHDL, while QSYS connects different system components which communicate and access each other. This information are required to generate the "preloader". This "preloader" coordinates the boot process. The next step of booting is the "universal boot loader" which configures the fpga and load the "linux-kernel". The Avalon-MM-Interface is required to grant the fpga access to the main memory of the HPS. The Avalon-MM-Interface is a read-/write interface which uses master/slave connections.

2.3 Problems

No permission to use partial reconfiguration feature in Quartus Software. Get a new license which allows us to use the feature but our Cyclone V FPGA is not supporting this feature. A more special FPGA is required which is not affordable. After working a few weeks on the field of partial reconfiguration a big problem occurs. Altera do not offer a permission to this feature on this hardware. Partial reconfiguration is only available for a new, more expensive board.

The status quo at the time of cancellation this task

- literature survey
- top level design
- static region → ticker (still runing while reconfiguration)
- pr region → two different LED pattern; persona A, persona B
- wrapper design

2.4 Discussion

partial reconfiguration could be a good thing, but the opinions differ a lot. some say it is only the work for univesity in resarch fields but not in industry. pro vs. contra arguments to be stated here.

2.5 Appendix - VHDL Code

2.5.1 PR top level

```
library IEEE;
use IEEE.std_logic_1164.all;

entity PR_test_top is
    port(

            --Button to start PR
            PR1_button          : in STD_LOGIC;


            --DIP switch to select PR_bitstream
            dir_switch_1        : in STD_LOGIC;

            --DIP switch to change blinking mode
            dir_switch_2        : in STD_LOGIC;


            system_clock        : in STD_LOGIC;
            PR_reset_button : in STD_LOGIC;

            PR_done_led         : out STD_LOGIC;
            PR_error_led        : out STD_LOGIC;
            LED                 : out STD_LOGIC_VECTOR (3 downto 0);
            -- 7 segment display
            disp_hex0           : out STD_LOGIC_VECTOR (6 downto 0);
            disp_hex1           : out STD_LOGIC_VECTOR (6 downto 0);
            disp_hex2           : out STD_LOGIC_VECTOR (6 downto 0);
            disp_hex3           : out STD_LOGIC_VECTOR (6 downto 0);
            disp_hex4           : out STD_LOGIC_VECTOR (6 downto 0);
            disp_hex5           : out STD_LOGIC_VECTOR (6 downto 0)
        );

end PR_test_top;

architecture behv of PR_test_top is
    component freeze_region
         port( clk          :in STD_LOGIC;
            dir             :in STD_LOGIC;
            freeze          :in STD_LOGIC;
            leds                :out STD_LOGIC_VECTOR (3 downto 0)
```

```
        );
    end component freeze_region;

    component pr_user_host is
    port(   areset          : in STD_LOGIC;
            clk                     : in STD_LOGIC;
            channel             : in STD_LOGIC;
            allow_pr_start : in STD_LOGIC;

             -- Outputs
            testing_done        : out STD_LOGIC;
            pr_freeze           : out STD_LOGIC;
            error_flag_pr   : out STD_LOGIC;
            really_done         : out STD_LOGIC);
    end component pr_user_host;

    component ticker_disp is
         port( clk                  : in STD_LOGIC;
               disp_hex0            : out STD_LOGIC_VECTOR (6 downto 0);
               disp_hex1            : out STD_LOGIC_VECTOR (6 downto 0);
               disp_hex2            : out STD_LOGIC_VECTOR (6 downto 0);
               disp_hex3            : out STD_LOGIC_VECTOR (6 downto 0);
               disp_hex4            : out STD_LOGIC_VECTOR (6 downto 0);
               disp_hex5            : out STD_LOGIC_VECTOR (6 downto 0)
         );
    end component ticker_disp;

    -- ****************** Missing ************************
    --  component for LED 3-0
    -- clk                  : in STD_LOGIC;
    --  LED                 : out STD_LOGIC_VECTOR (3 downto 0));


--signal max_count:integer  := 300000000;       -- 08.06.2015 Ticker
--signal count:integer      :=0;                    -- 08.06.2015 Ticker
signal pr_freeze_reg        : STD_LOGIC;
signal pr_freeze                : STD_LOGIC;
signal channel              : STD_LOGIC;
signal done_w                   : STD_LOGIC;
signal really_done          : STD_LOGIC;
signal error_flag_pr_w      : STD_LOGIC;

begin
channel <= dir_switch_1;
```

```
freeze_region_inst: freeze_region
 port map (
   clk      => system_clock,
   freeze   => pr_freeze_reg,
   dir      => dir_switch_2,
   leds     => LED(3 downto 0)

    );

pr_user_host_inst : pr_user_host
port map (
      areset          => PR_reset_button,
      clk             => system_clock,
      testing_done    => done_w,
      pr_freeze       => pr_freeze,
      error_flag_pr   => error_flag_pr_w,
      allow_pr_start  => PR1_button,
      channel         => channel,
      really_done     => really_done);

ticker_inst: ticker_disp
 port map (
   clk          => system_clock,
    disp_hex0    => disp_hex0,
    disp_hex1    => disp_hex1,
    disp_hex2    => disp_hex2,
    disp_hex3    => disp_hex3,
    disp_hex4    => disp_hex4,
    disp_hex5    => disp_hex5
   );

 process(system_clock) begin
    if rising_edge(system_clock) then
        PR_done_led          <= not done_w;
        pr_freeze_reg     <= pr_freeze;
        PR_error_led         <= not error_flag_pr_w;
    end if;
end process;


end behv;
```

3 Fault simulation

3.1 Idea of fault simulation

The idea of fault simulation on the fpga is, to detect stuck at faults very fast.

3.2 Implementation

approach, vorgehensweise -> adaption of the previous work. was man alles beachten muss, was man alles aendern muss... pfade aendern, treiber installieren (-> nicht trivial !!), tcl, batch, c programm, vhdl, qsys, quartus programm verstehen, was was macht. design flow, work flow, compilation. what to do... what changed...

3.3 Problems

Behindert alles

3.4 Discussion

what can be changed, what is the point of work, what are the next steps. welche probleme koennen wie geloest werden?