IFAC
Publications

DEVELOPMENT OF A DEBUG MODULE FOR A FPGA-BASED MICROCONTROLLER

Rainer Bermbach, Martin Kupfer

University of Applied Sciences Braunschweig/Wolfenbuettel, Germany

Abstract: Description of the development of a debug system for a PIC™-compatible SoC microcontroller implemented in a FPGA. The debug module comprises a hardware interface in VHDL, which communicates with the respective microcontroller components and a software application to present the relevant information to the user and to allow for convenient control of microcontroller functions. Communication between front-end and back-end is accomplished via the JTAG port. The debug module allows easy non-interfering debugging of software for the SoC microcontroller. *Copyright © 2006 IFAC*

Keywords: debugging, microprocessor, hardware, SoC, VHDL, JTAG port.

1. INTRODUCTION

Today, convenient debugging of embedded system software is more or less state of the art. Powerful in-circuit emulators or on-chip hardware assisted debuggers help engineers to develop the programs for all those embedded systems based on standard microcontrollers and microprocessors. Typically, difficulties arise when working with IPs of processors and controllers for building a system-on-chip (SoC). Usually, one has to utilise old-fashioned software debuggers which require at least some system resources by themselves. Interference of the debugger with the target device hinders real-time debugging and often makes full system tests impossible.

At the Computer Engineering Lab of the University of Applied Sciences Braunschweig/Wolfenbuettel a PIC™-compatible microcontroller called VHDL-PIC had been developed and optimised in former projects, see (Bermbach, 2003; Cramm, 2003; Bermbach, 2004; Andreas, 2004). The VHDL-based controller, fully compatible to the cores of the PICmicro™ mid-range MCU family, features typical peripherals such as ports, timer, UART, etc., at a frequency of up to 100 MHz, i.e. with processing power of up to 25 MIPS.

When using the microcontroller in FPGA implementations all the above mentioned difficulties arose. Debugging of system software turned out to be troublesome and tedious. In addition, all software changes had to be run through the software development system. Then the code needed to be transformed into VHDL ROM initialisation code and finally, all the steps of the hardware implementation process had to be performed again. For each error correction all of the above mentioned steps needed to be repeated which was not very effective. To bypass all those problems the idea of an on-chip hardware assisted non-interfering debugger with direct code download was born, see (Kupfer, 2005). The debug module is comprised of a hardware interface (back-end) written in VHDL, which communicates with the respective microcontroller components and a Microsoft Windows™ based software application (front-end) to present the relevant information to the user and to allow for convenient control of microcontroller functions. Communication between front-end and back-end is accomplished via the JTAG port which is available on the FPGA. The debug module allows easy non-interfering real-time debugging of the SoC microcontroller.

The development and implementation of that debugging system is presented in this text. In the

following section a system overview will be given, whereas section 3 describes the hardware interface. Section 4 discusses the implemented debugging functions and section 5 the structure of the back-end/front-end communication. Section 6 gives some detail on the front-end and its implementation and passes over to the conclusion.

2. SYSTEM OVERVIEW

The configuration (see fig. 1) for developing SoC hardware and software with the available VHDL-PIC microcontroller is comprised of a PC running Windows XP™ and a FPGA development board from Digilent Inc. which carries a 200k gate Spartan 3 FPGA from Xilinx. On the PC the free integrated development environment (IDE) MPLAB™ v7.01 from Microchip is used to input source code and assemble it into Intel Hex encoded executable form. Additionally, MPLAB™ may be used to do first software simulations running the built-in simulator.

By means of the graphical user interface (GUI) the user selects the program code to be run on the VHDL-PIC and sends it via the debugging software to the FPGA where it is loaded into internal Block RAM memory configured as ROM. Communication occurs through a parallel port driver serving the JTAG port (IEEE 1149.1) on the FPGA development board. The communication interface inside the FPGA, the so-called TAP controller, handles all read, write and communication requests between the hardware interface and the debugging software.

In addition, VHDL development takes place on the PC utilising Active VHDL™ (Aldec) whereas software from Xilinx (ISE™ 6.3i) synthesises, fits, routes and downloads any hardware code to the Spartan 3 FPGA on the Digilent Spartan-3 Starter Kit Board.

3. HARDWARE

This section describes the underlying architecture and structure of the microcontroller as well as the requirements and the implementation's approach to make the processor "debuggable".

3.1 PIC Microcontroller Architecture

The PICmicro™ controller is a simple but powerful 8-bit processor with a Harvard architecture, a 14-bit instruction word, a hardware stack, a work register which may be involved in nearly every instruction and an internal RAM space of up to 512 bytes organised in four banks, see (Microchip, 1997). The register file (general purpose RAM, GPR) is located in that address space. All special function registers (SFR) of the CPU and peripherals are mapped into this space as well. The controller has single cycle instructions; only jumps, conditional jumps, calls and returns require two cycles. It works with a 2-stage

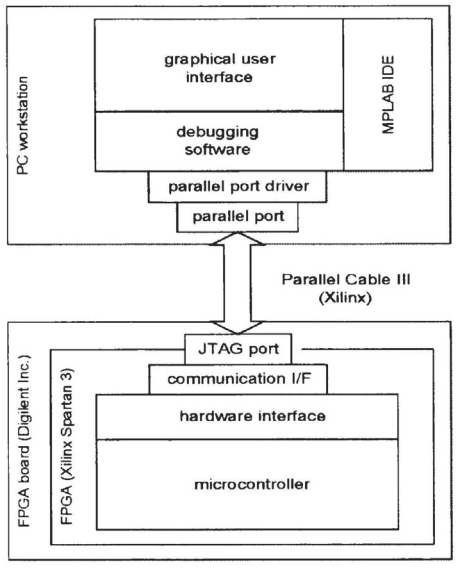

Fig. 1. Block diagram of the development system

pipeline: One instruction is read while the previous is decoded and executed. The machine cycle is divided into four steps called Q1 to Q4. In Q1 the instruction is decoded, in Q2 operands are read, in Q3 the operation is executed and in Q4 results are written back.

3.2 VHDL-PIC Structure

The VHDL-PIC core is fully compatible with the PICmicro™ controller. It not only executes the same instructions as its standard IC counterpart, it was also designed to meet the publicly available timings of the PICmicro™. So the machine cycle stages, interrupt cycles, etc. are fully equivalent.

For an efficient implementation, the instruction ROM uses the Block RAM feature of the Spartan 3 FPGA. Up to two dedicated RAM blocks can be utilised as program ROM giving a maximum code size of 4 Kbytes (2K x 16 bit). The GPR register file is also implemented as Block RAM. The hardware stack may be built from Block RAM or as dedicated flip-flops which is defined in a special library, MyPICPack. The library also determines the size and memory locations of the GPR to adapt the VHDL-PIC to various real PICmicro™ types. In addition, a lot of other configuration data may be manipulated by setting/resetting corresponding constants in the library.

3.3 Debugging Hardware

The functions of the debugger can be divided into hardware and software functions. The means to directly start, stop, read and write, etc. have to be implemented in the hardware interface to the processor core.