UNIVERSITY OF SOUTHAMPTON
Faculty of Engineering, Science and Mathematics

A group design project report submitted for the award of
Master of Engineering

Supervisor: Dr Peter R. Wilson
Examiner:

ELEC6027 VLSI Group Design Project
Team I
Southampton Superchip Automated Tester

by
Alexander Hornung
Xiaolong Li
Pavlos Progias
Romel Torres
Bo Zhou
May 15, 2012

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF ENGINEERING, SCIENCE AND MATHEMATICS

A group design project report submitted for the award of Master of Engineering

by
Alexander Hornung
Xiaolong Li
Pavlos Progias
Romel Torres
Bo Zhou

Contents

List of Figures

List of Tables

Listings

List of Symbols

w The weight vector

Acknowledgements

Thanks to no one.

To ...

Chapter 1

Introduction

1.1 Aim and Background

Nano (get started)
mention: D2, Superchip, stuff

1.2 DE2 Board Overview

Nano (get started)

1.3 System Overview

Pavlos (get started)
see slides

Chapter 2

Hardware

2.1 Hardware Overview

The hardware part consists of two main blocks: NIOS System on Programmable Controller and the Tester.

The *NIOS SoPC* coordinates the whole system, writing the data from SD Card into the SRAM for the Tester to read the data from it; retrieving the result that the Tester writes

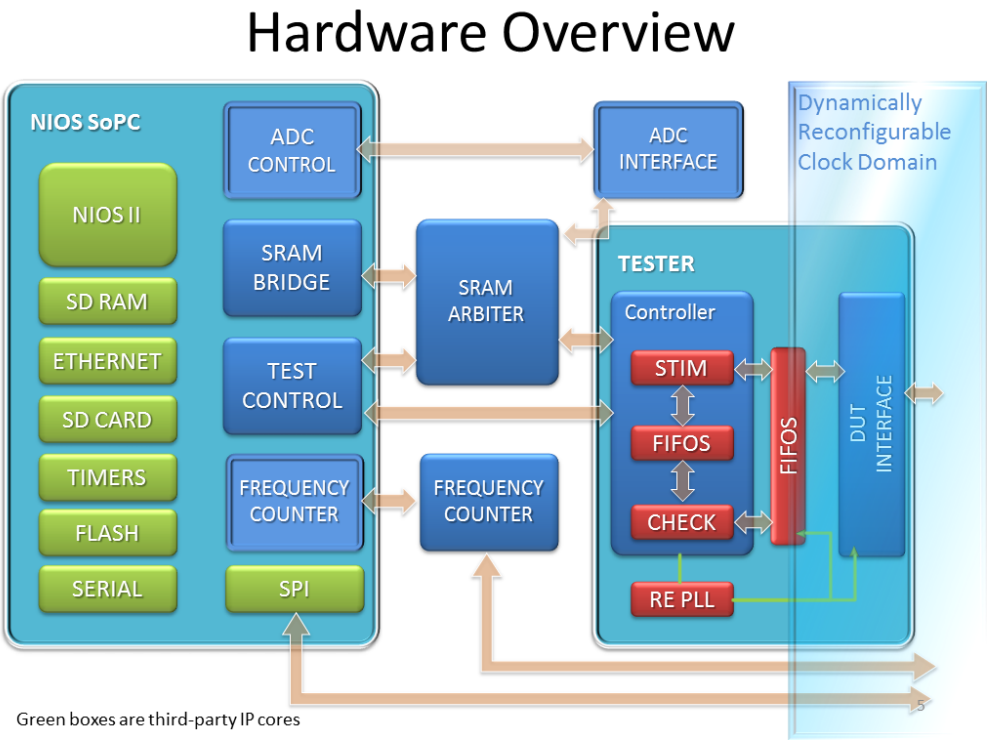

FIGURE 2.1: Hardware Overview.

back to the SRAM and sending the data to the backend for archiving and reviewing for the users. Its tasks include:

- coordinate the data from multiple sources (SRAM, SD Card, Flash, Serial connection, SPI bus, Ethernet, etc.)
- control the testing process (which is executed by the Tester, frequency counter and ADC)
- send results to the backend
- user interface

The *SRAM arbiter* is the interface between the SoPC, tester and the SRAM by Avalon Memory Mapped Interface (Avalon-MM). It handles the SRAM read and write requests from SoPC and tester.

The *Tester* contains several HDL blocks which are responsible for the testing process.

- read the command, request and test vector from SRAM
- generate the test input signals for every pin of the DUT
- monitor the output pins of the DUT and write the result into the memory when the it is valid
- offer a dynamically reconfigurable clock for the DUT and its relative interface

The purpose of the *frequency counter* and the *ADC* is that there is a designated pin on the chip which is the output of a ring oscillator. It is preferable that not only the frequency of this clock output is tested, but that the output waveform is also monitored.

2.2 PCB (interface board)

For the chip tester, two PCBs are needed in this project. One which is called $B2$ has been designed for the virtual designs to be loaded in a slave FPGA. The other is called $B1$ that is developed as an interface for those real chips are about to be tested. Both PCBs are two layers board and designed by using Allegro 16.3 PCB designer tools. The dimensions of $B1$ and $B2$ are $98.93mm * 68.5mm$ and $81.28mm * 68.33mm$ and the minimum wire size is $8mm$.

2.2.1 B1 (DUT testing board)

2.2.1.1 DUT

The superchip under test has the following specific features:

- 6 separate design sites
- 24 digital input pins $[A0 \dots A23]$, shared between all design sites;
- 4 digital output pins $[Q0 \dots Q23]$, shared between all design sites;
- 16 separate VDD pins, one dedicated to each design site;
- 1 global GND pin for all design sites and infrastructure circuitry;
- 1 global VDD pin to power the site buffers and I/O pad ring;
- 68-pin JLCC package (with 2 unused pins).

The chip is interfaced with the PCB by a 68WAY PLCC socket. The power is supplied by the $3.3V$ VDD from the DE2 board. The GND is also connected to the DE2 GND.

The shared I/Os are connected to the powered design site, and disconnected from the other design sites. Hence a controllable power switch is designed with a 4-16 decoder and an integrated power switch array. The $Q1$ is the output of a ring oscillator. The frequency can be digitally tested by the general I/O. However, to examine the analog properties of this output, an analog part including a buffer, 3^{rd} order Butterworth filter and ADC with their own power supply is also designed.

2.2.1.2 Digital Part

Decoder Although there are 16 design sites to be selected, the data can be shrunk down to 4 bits to save space of the test vector and metadata, since only one site is selected at a time. The decoder we used is $74HC4514$, a $4 - to - 16$ line decoder with latch. The input and output are active high.

Switch Array The switch array consists of four $TPS2095$ quad power-distribution switches. One could simply use a CMOS to realize a switch. However, with thermal sense, current limit and charge pump, the $TPS2095$ switches are more reliable, stable, smooth (minimum switching current surges) and relatively compact in scale. The operation of a switch is simple: when EN pin is asserted, the OUT pin offers the power with the same voltage in the IN pin. Otherwise the OUT pin is disconnected from the input power. In one $TPS2095$ chip there are 4 such switches, the EN pins of which are all active high.

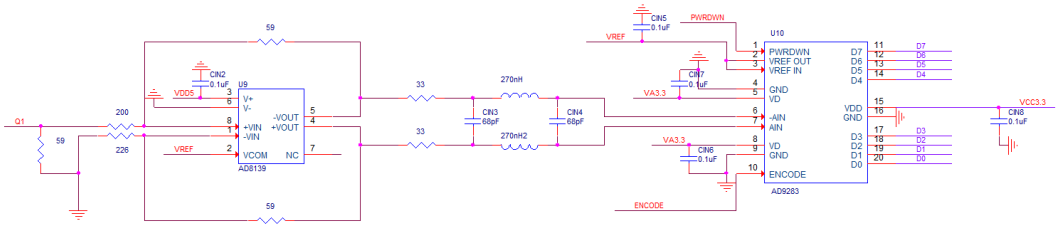

FIGURE 2.2: Analog Part Overview.

2.2.1.3 Analog Part

Buffer An $AD8139$ differential ADC driver is used as the front buffer of the ADC. It is an ultralow noise, high performance differential amplifier with rail to rail output from Analog Devices. The designed gain of the buffer is 0.295, and the differential gain is approximately 0.14.

ADC The $8-bit$, $100MSPS$ ADC is used to convert the analog signal. The encode clock is the clock from the DE2 board ($100MHz$ by default). A power-down function select is also connected to the FPGA for shutting down the ADC when its not in use. The ADC signal wires should be placed far away from the other wires for minimum interference; and they should also be placed parallel to each other to have similar length for better signal integrity. However, since the pin density is too high of the HSMC connector we used, it is inevitable that the ADC wires are closed to the other wires. The ADC is placed on the other side of the PCB board for a better result.

Butterworth Filter To minimize the high frequency noise, a third-order low pass Butterworth filter is designed. With three poles, the attenuation is $-60^{dB}/_{decade}$ on signals higher than the cut-off frequency. The filter topology used here is balanced ladder topology. Automatic design tool Elsie is used for designing the parameters of the RCI circuit. The cut-off frequency is $40MHz$.

Separate Power Supplies The buffer and ADC are separately powered from the other parts of the board. The $AD8139$s power ranges from $5V$ to $12V$, and the ADC ranges from $2.7V$ to $3.6V$. Two low dropout regulators (LDP), $ADP3335$ and $ADP3333$ are used to offer $5V$ and $3.3V$ voltages respectively. The supply voltage of these two LDP ranges from $2.6V$ to $12V$. Their load currents are up to $500mA$ and $300mA$ respectively.

2.2.2 B2 (Virtual design board)

2.2.2.1 Slave FPGA

Slave FPGA is the core chip of the virtual design board. It can be used for programing designs when they are going to be tested. An Altera Cyclone III FPGA will be implemented as the slave FPGA.

Cyclone III ($EP3C25E144$) belongs to Cyclone III device family. With densities ranging from about $5,000$ to $200,000$ logic elements (LEs) and $0.5Mb$ to $8Mb$ of memory for less than $^1/_4V$ of static power consumption, Cyclone III devices makes it easier to meet the power budget (Altera Corporation, 2011).

The reason for choosing Cyclone III is their lowest power, high functionality with the lowest cost. 144 pins are enough for this design. I/O pins on the Cyclone III are grouped together into I/O banks, and each bank has a separate power bus. There are eight I/O banks, as shown in Figure 1 (Altera Corporation, 2011). All single-ended I/O standards are supported in all banks except HSTL-12 Class II which is only supported in column banks. The same case can be found in all differential I/O standards.

Each I/O banks of Cyclone III has a VREF bus to accommodate voltage-referenced I/O standards. Connect the VREF pin of each group to the appropriate voltage. In this design three voltage levels, which $VCCIO = 3.3V$, $VCCINT = 1.2V$ and $VCCA = 2.5V$, are necessary. Proper bypassing and decoupling technique for the power pins is very important for reliable design operation. In this case, additional decoupling capacitance is needed. The VCCINT, VCCIO and ground pins should add as many as $0.2\mu F$ power-supply decoupling capacitors as possible. So $0.01\mu F$ capacitor for VCCIO and $0.1\mu F$ for VCCINT seems to be appropriate. Note that all the capacitors should be place close enough to the power pins.

In regard of the Cyclone III configuration, the Cyclone II device uses SRAM cells to store configuration data. $EP3C25E144$ can only be configured in Fast Active serial (AS) mode.

Configuration data is loaded into Cyclone III at each DCLK cycle. As soon as the device receives all data, the device releases the open-drain CONF_DONE pin, which is pulled high by an external 10- pull-up resistor. The CONF_DONE pin transits low-to-high to indicate that configuration is complete and initialization of the device can begin. The CONF_DONE pin must have a 10- pull-up resistor for initialization.

Pulling the nCONFIG pin low can begin reconfiguration and this pin must be low for at least $500ns$. The Cyclone III device is reset when nCONFIG is pulled low. Meanwhile, the device will pulls nSTATUS and CONF_DONE low and I/O pins are tri-stated. When nCONFIG returns to high and nSTATUS is released, reconfiguration begins.

The clock source for initialization is either a $10MHz$ (typical) internal oscillator or an optional CLK pin. In this situation, an external oscillator (A $TXC-50MHz$ oscillator is going to be introduced later) will be implemented as the clock source. The required clock for initialization in Cyclone III is $3,185$ and the maximum CLK frequency is $133MHz$.

The configuration mode is selected by driving the MSEL pins either low or high. The MSEL pins can be powered by VCCIO and GND. For AS mode, MSEL [1] should be pulled up by connecting to VCCIO. MSEL [0] and MSEL [2] are pulled down by connecting to GND. The MSEL pins have 9 internal pull-down resistors that are always active.

The maximum active master frequency for Cyclone III is $0MHz$ typically and device only work with serial configuration devices that support up to $40MHz$.

In AS mode, Cyclone III reads the configuration data providing by serial configuration (A Spansion SPI Flash is going to be introduced later) via a serial interface. The serial configuration device controls the configuration interface.

There are four pins on the serial configuration devices:

- Serial clock input (DCLK)
- serial data output (DATA)
- As data output (ASDI)
- Active-low chip select (nCS)

Connect these four pins to Cyclone III device pins, as shown in Figure 2.

A 25Ω series resistor must connect a between serial configuration device and the Cyclone III device at the near end of the serial configuration device for DATA [0] when configure the Cyclone III device in the AS mode. The 25Ω resistor works to minimize the driver impedance mismatch with the board trace and reduce the overshoot seen at the Cyclone III device input pin DATA [0].

The maximum trace length between Cyclone III device and the serial configuration device, in another words, the CLK, DATA [0], NCSO and ADSO pins, must less than $10in$. In the B1 PCB designing, only $8mm$ is used.

Cyclone III device uses a $40MHz$ internal oscillator to generate DCLK to controls the entire configuration cycle and provide timing for the serial interface.

By driving the nCSO output pin low, which connect to nCS pin of the configuration device, the Cyclone III device enables the configuration device. DCLK and DATA [1] pins are used to send operation commands and read address signals to the serial configuration device. The configuration device sends data on DATA pin which connects to

the DATA [0] pin of the Cyclone III device. After all the configuration bits are received, cyclone III releases the open-drain CONF_DONE pin with a 10Ω pull-up resistor. The CONF_DONE pin must have an external 10Ω resistor for the device to initialize.

2.2.2.2 Serial Configuration Device

Cyclone III FPGAs are programmable logic devices used for basic logic functions, chip-to-chip connectivity, signal processing, and embedded processing. They can be programmed and configured by a microprocessor, JTAG port, or directly bay a serial PROM or flash. Spansion SPI (Serial Peripheral Interface) flash $S25FL064K$ can configure the FPGA easily at power-up (Spansion, 2011).

The three stages of the configuration cycle are power-on reset, configuration, and initialization. When the FPGA enters power-on rest (POR), it drives the nSTATUS signal low to indicate it is busy, drives the CONF_DONE signal low to indicate the configuration has not been completed, and tri-states all I/O pins. All pins will be released after POR.

The DCLK generated by the FPGA device control the configuration data transferring. The CONF_DONE pin will be released with pulling high by an external pull-up resistor after all configuration data is transferred to the FPGA. The FPGA enters user mode after internal initialization.

The SPI is a simple four-pin synchronous interface protocol which enables a master device and one or more slave devices to intercommunicate. Four signal wires are:

- Master Out Slave In (MOSI) signal generated by the master (data to slave)
- Master In Slave Out (MISO) signal generated by the slave (data to master)
- Serial Clock (SCK) signal generated by the master to synchronize data transfers
- Slave Select (SS) signal generated by master to select individual slave devices (also known as Chip Select (CS) or Chip Enable (CE))

Figure 3 displays a simple block diagram of the connection between FPGA and SPI flash, as well as the HSMC header and JTAG programming the SPI flash from a host PC.

Figure 4 displays the details of the connection between SPI flash and FPGA. According to the introduction of Cyclone III, we can acquire the whole routing of FPGA, SPI flash and the HSMC header (A Samtec ASP header will be introduced later).

The TMS, TDI, TDO and TCK pins of Cyclone III device are used to operating in the IEEE Std. 1149.1 BST. The TDO pin is powered by VCCIO (3.3V). TDI and TMS are powered by VCCA (2.5V).

The header for JTAG and the header for SPI Flash In-system is use the same HSMC header with 172 pins which is quite enough for these two headers.

2.2.2.3 External Oscillator

As mentioned in the Cyclone III part, an external oscillator is needed as the clock source to provide certain frequency for the FPGA. A $50MHz$ oscillator of TXC will be implemented. The typical clock frequency for Cyclone III device is $30HMz$ and the maximum is $40MHz$. The TXC DEL04 oscillator is a sealed clock crystal oscillator unit with high precision characteristic covering up to wide frequency range ($1MHz$ to $170MHz$), which is appropriate for the FPGA. The supply voltage range is $1.8V \sim 5V$ (TXC).

2.2.2.4 Voltage Regulator

The $B2$ board needs at least three different voltage levels, as mentioned before, $VCCIO = 3.3V$, $VCCINT = 1.2V$ and $VCCA = 2.5V$. The external power source is provided by the main FPGA DE2-115 board at $3.3V$. Hence two voltage regulators are used to transfer $3.3V$ into $1.2V$ and $2.5V$.

1.2V Voltage Regulator $LD1117$ The $LD111712$ is a low drop voltage regulator able to provide up to $800mA$ of output current, available in adjustable version (). The device is supplied in DPAK surface mount package optimize the thermal characteristics even offering a relevant space saving effect. A very common 10 minimum capacitor is needed for stability (ST Microelectronics, 2012).

Figure 5 is the application circuit for $1.2V$ output.

2.5V Voltage Regulator $TPS78225$ The $TPS78228$ is a low dropout linear voltage regulator designed by TI. The enable pin (EN) is compatible with standard CMOS logic while the low drop output is stable with any capacitor greater than 1. The device requires minimal board space for miniaturized packaging and potentially small output capacitor (Texas Instruments, 2008).

The enable pin is active high and is compatible with standard and low-voltage CMOS levels. Therefore if the shutdown capacitor is not necessary, enable pin can connect to the IN pin as shown on Figure 6.

2.2.2.5 HSMC Header

The Altera High Speed Mezzanine Card (HSMC) specification defines the electrical and mechanical properties of the HSMC adapter inter face for FPGA-based motherboards. The HSMC connector is based on the Samtec $0.5mm$ pitch, surface-mount QTH/QSH family of connectors (Altera Corporation, 2009). Two versions can are used in FPGA board. $ASP-122953-01$ Socket for the host boards and $ASP-122952-01$ Header for Mezzanine Cards (slave boards).

Figure 7 is the diagram for HSMC header. The clock-data-recovery differential signals in Bank 1 are the highest frequency signals. Signals between the HSMC connector and the host board FPGA device are intended to be D/C coupled. The JTAG, a system management bus (SMBus), and clock signals are also dedicated in Bank 1. In banks 2 and Bank 3, there are main CMOS/LVDS interface signals, including LVDS/COMS clocks, as well as both $12V$ and $3.3V$ power pins.

The host board is any board with an FPGA connected to one or more HSMC interface. In this project it is $DE2-115$ developed board. The interconnect I/O pins available on the HSMC connector can have all possible I/O standard and logic features that can be supported by the host FPGA since FPGAs are configurable devices. However basically they are limited by the wire types on the board.

The HSMC connectors provided the interface between host and slave boards. The "header" part ($ASP-122952-01$) on slave board plugs into the "socket" part on the host board. The host board provides $+12V$ DC and $+3.3V$ DC power to the slave board via the HSMC connector. In addition to power and clock signals, the host board provides access to JTAG, high speed serial I/O, and single-ended or differential I/O via the HSMC connector.

The HSMC connector has a total of 172 pins, including 121 signal pins (120 signal pins + 1 PSNTn pin), 39 power pins, and 12 ground pins. The ground pins are much larger than the power pins and are located between the two rows of signal and power pins. Figure 7 is the modules for HSMC connectors.

The $ASP-122952-01$ header provides 160 total pins and 12 ground plane connection pins down the center. Bank 1 has 40 pins with every third pin removed. Bank 2 and 3 have 60 pins each as no pins are removed. Host boards provide transceivers to Bank 1 which is not used in this project. Single-ended signals are provided to Bank 2 and 3. Typically, the single-ended signals are capable of differential signalling such as LVDS.

The JTAG signals are intended to connect to dedicated JTAG pins on the host FPGA and be part of the JTAG chain. The JTAG signals TCK, TMS and TDI are intended to be output from host board while JTAG TDO should be the input to host board as Figure 4 before.

Tables 1 and 2 in the appendix show the pin-outs for the HSMC header on B1 and B2 boards, respectively.

2.3 PCB (virtual design board)

Nano (and Joey)

mention: reason behind choice of components, etc (e.g. 100 MS/s ADC, because we use a 100MHz main clock on FPGA, also enough for sampling 10 MHz signals, etc.; Altera FPGA because uni uses Altera mainly)

mention: filter design (3rd order passive Butterworth - I can provide you with the transfer function and so on; filter was "designed" using Elsie (a software you can download) and tested/characterised in Multisim)

2.4 HSMC board-to-board interface

Nano (and Joey)

mention: signals used on HSMC connector for each of the boards

don't go into details as in, JTAG_TMS on pin 1, JTAG_TDO on pin 2, etc; rather: we have a JTAG interface, a clock, 24 pins for this, 24 pins for that, SPI interface, ...

2.5 Chip Tester (Verilog Module)

Joey (and a bit Alex)
mention: reconfigurable clock domain, synchronizing DC FIFOs
- test controller
- mem_if
- stim
- check
- dut_if (and its pipeline)
- reconfigurable PLL

2.6 SRAM Arbiter (sram_arb_sync.v)

Pavlos

Chapter 3

SoPC

3.1 Overview

Alex
- Altera-provided Peripherals
- Nios II core
- Avalon-MM bus (at least the signals we use)

3.2 SRAM_bridge

Alex

3.3 test_runner

Alex

3.4 Frequency Counter

The frequency counter is good.

3.5 ADC

Pavlos

Chapter 4

Embedded Software

4.1 Bootloader and Operating System

Alex
- u-boot
- uClinux (w Linux 3.3)

4.2 Drivers

Alex
- fcounter
- test_runner (and arbiter)
- adc
- Character LCD driver
- sram_access (not really driver, simply memory mapped)

4.3 confrd

Alex
- parser, lcd, api access, hardware access, ...

4.4 SPI Flash Configuration

Romel

4.5 Configuration File Format

Alex (get started)

Chapter 5

Backend Software

5.1 Overview

In order to store and show the results sent from the **FPGA** in a database a web server was developed; this server provides the user an interface where they can see and manage the results, upload designs and configuration files to the **chip tester** and administrate the database where the results have been stored. Basically the web server "serves" the content to the user through the Internet to a web page, where the users (administrators or students) can check the results of their designs or manage the results stored in the database. The users load their designs into the web server and then whenever the FPGA sends a "download configuration files request" to the server, it will send the files to the FPGA (if available).

Nowadays almost every programming language provides libraries to develop web servers; for this application we required a robust, easy to use, programming language. Java, Python, PhP, Ruby, Perl are the most common languages used to develop servers. We decided to use Ruby with its**web framework** Sinatra. Ruby is a general-purpose object-oriented programming language easy to learn and powerful, Sinatra is a Domain Specific Language (DSL) for quickly creating web applications in ruby.

UNICORN INSTEAD OF WEBBRICK, WHY?

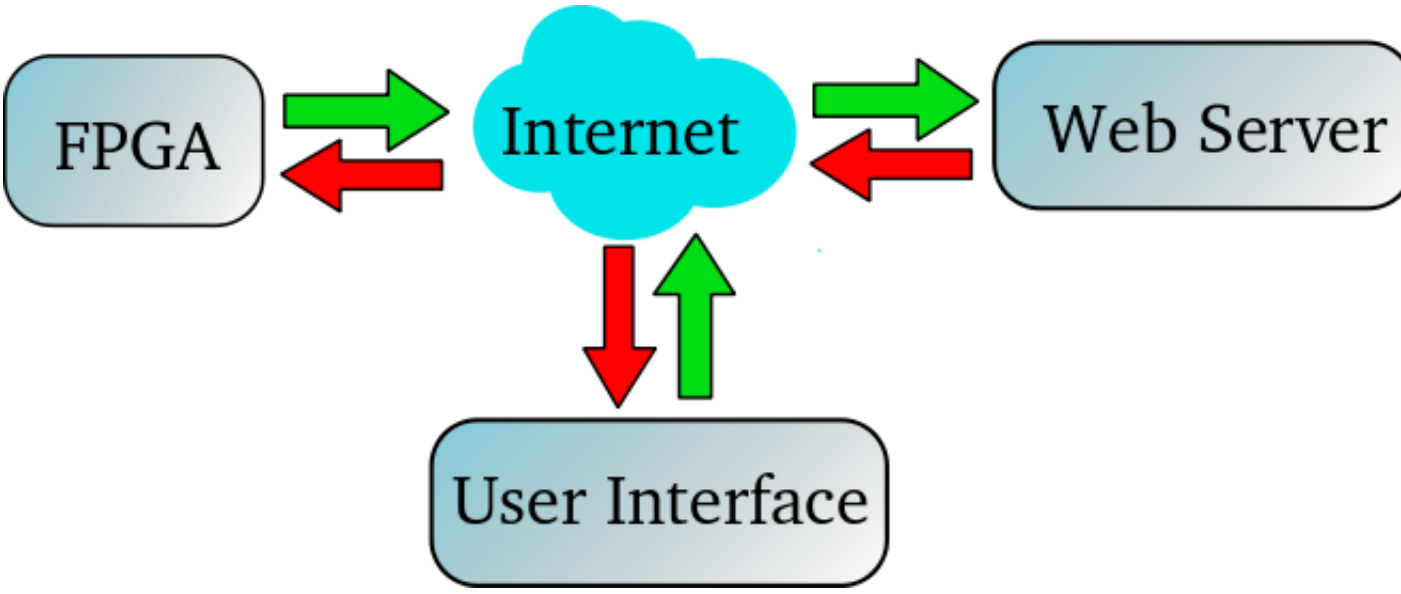

FIGURE 5.1: Interaction between elements in the system

5.2 API

5.2.1 Hypertext Transfer Protocol

The way the FPGA communicates to the server is through the Hypertext Transfer Protocol (HTTP). In the client-server model HTTP works as a request-response protocol. The FPGA and the user web browser act as clients, while the application running on the computer hosting the web site is the server. The FPGA and web browser submit HTTP request messages to the server. The server stores content, provides resources such as configuration files, sends emails, manages the database on behalf of the clients. HTTP defines 9 methods indicating the action to be performed on a resource: HEAD, GET, POST, PUT, DELETE, TRACE, OPTION, CONNECT and PATCH. To develop this application we have used only three of this methods **GET** which retrieves information from the server, **POST** which sends data to the server as part of the request and **DELETE** which deletes resources. The figure 5.2 shows an example of the get method, both in the server and in the browser.

In order to communicate from the FPGA to the server a data exchange language is needed; the most popular data exchange language is "XML", nevertheless, adding the xml parsing libraries in the μClinux would slow down the parsing data process due to the complexity of the xml libraries. JSON, is a lightweight data exchange library supported in C (for the FPGA) and ruby (for the Server) therefore it was perfect for our application.

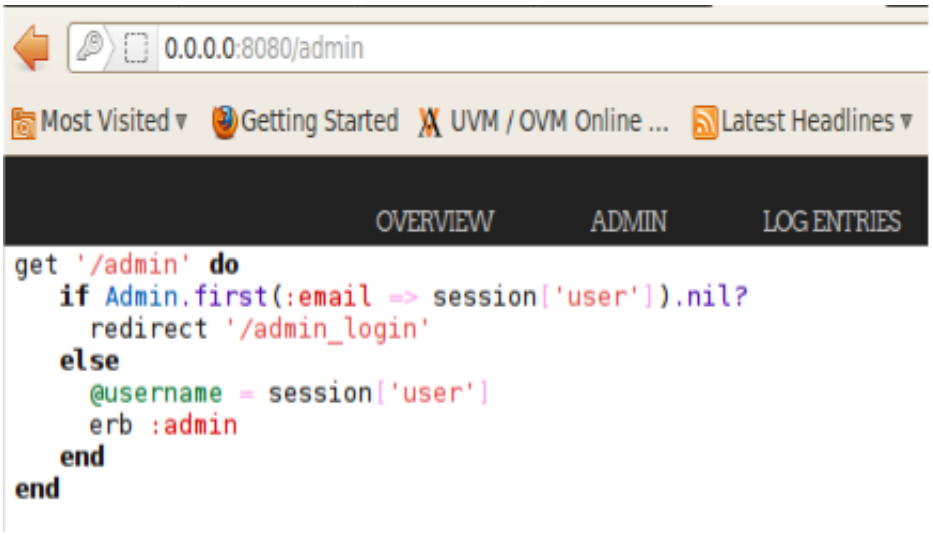

FIGURE 5.2: Example of the HTTP method get in the browser and in the server

5.3 "Server" (Sinatra stuff, database, etc)

```
+server
   +public
      +css
      +img
   +uploads
   +utils
   +views
-config.ru
-config.yml
-database.rb
-email.rb
-Gemfile
-init.rb
-Rakefile
-s.css
-server.rb
-s.scss
-unicorn.conf.rb
```

FIGURE 5.3: Folder structure of the server

Inside the folder called **server** are all the files regarding to our back-end application. In the folder **public** are the stylesheets for the web page styling, and the images displayed in the page, in **uploads** are all the files uploaded in the server by the users, in **utils** are some utility scripts written in ruby such as secure password generation, **views** contains all the views in the page that the different users can have access to.

5.3.1 Database

Datamapper is an ORM (Object-Relational Mapping) that maps objects written in ruby to any supported database, thus giving freedom to the administrators of the system to

Email	Permission	Password_hash
String	Integer	BCryptHash

TABLE 5.1: Admin table in the database

Id	Email	Team	File_name	Is_valid	Created_at	Updated_at
Serial	String	Integer	String	Boolean	Date	Date

TABLE 5.2: File Upload table in the database

use any database system (such as MySQL, SQLite, Postgres,. . .) to choose the database the administrator simply has to change the configuration parameters in the file **config.yml** (found in the server folder) as shown in the figure 5.4 where the administrator has set up the database system with MySql.

```
database:
    type: mysql
    user: root
    password: chip1234
    database_name: ChipTester
    address: localhost
```

FIGURE 5.4: Database system selection using the config.yml file

In the table Admin 5.1 is stored the administrators of the database. The email field is the email used t log in, the Permission field is the level of authority that each administrator has. It can be either 0 or 1 (where 0 means that it can add more administrators to the system) and the password_hash field is a hash related to the user password (more in the server security section)

The initial administrator is configured in the archive config.yml as shown in the figure 5.5. For security reasons the password of the initial administrator must be encrypted using the script **pass.rb** located in server/utils. Whenever the server is run by the first time, or the database has been created again this user will be mapped into the database, granting always access to the administrator.

```
#Master email address and password
admin:
    username: prw@soton.ac.uk
    password: $2a$10$KVIbrolv6wwL0CoHYt.L2.EQab4kGVTWdmRgz2YRV4SEe56PQErNG
    permission: 0
#Maximum permission in the databse
```

FIGURE 5.5: Initial Master Admin selection using the config.yml file

The File Upload table 5.2 in the database stores the configuration files that has been uploaded to the server by the students.

The Log entry table stores all the logs sent from the FPGA to the server, this table is displayed to the user in the web page.

Id	Type	Message	File	Line	Created_at	Updated_at
Serial	Integer	String	String	Integer	Date	Date

Table 5.3: Log Entry table in the database

Id	Team	Academic Year	Created_at	Updated_at	Virtual
Serial	Integer	String	Date	Date	Boolean

Table 5.4: Result table in the database

Id	Created_at	Updated_at	Triggers	File_name	Clock_Frequency	Design_name
Id	Date	Date	String	String	String	String

Table 5.5: Design Result table in the database

Trigger_timeout	Has_run	Fail	Created_at	Updated_at
Boolean	Boolean	Boolean	Date	Date

Table 5.6: Test Vector Result table in the database

Id	Frequency	Created_at	Updated_at
Serial	Float	Date	Date

Table 5.7: Frequency Measurement table in the database

The result table has

Id	Type	Input_vector	Expected_result	Actual_result	Cycle_count
Serial	Integer	String	String	String	Integer

5.3.2 Security

BCRYPT

5.3.3 Views

5.4 E-Mails

Romel

5.5 Remote Reconfiguration

Romel

5.6 User Interface

Pavlos

Chapter 6

Some other yet-unnamed section

6.1 Detailed resource usage on FPGA

stuff.

6.2 Rough bill of materials (or more like, cost of each board)

more stuff.

Pin Number	Pin Name
39	ENCODE
48	A0
50	A1
54	A2
56	A3
60	A4
62	A5
66	A6
68	A7
72	A8
74	A9
78	A10
80	A11
96	A12
98	A13
102	A14
104	A15
108	A16
110	A17
114	A18
116	A19
120	A20
122	A21
126	A22
128	A23
132	Q0
134	Q1
138	Q2
140	Q3
144	Q4
146	Q5
150	Q6
152	Q7
156	Q8
158	Q9
157	Q10
155	Q11
151	D0
149	D1
145	D2
143	D3
139	D4
137	D5
133	D6
131	D7
127	Q12
125	Q13
121	Q14
119	Q15
115	Q16
113	Q17
109	Q18
107	Q19
103	Q20
101	Q21

Pin Number	Pin Name	Pin Number	Pin Name
35	TCK	36	TMS
37	TDO	38	TDI
39	Ded_Clock	41	CONF_DONE
42	nSTATUS	43	nCONFIG
44	nCE	48	IO1
50	IO2	54	IO3
56	IO4	60	IO5
62	IO6	66	IO7
68	IO8	72	IO9
74	IO10	78	IO11
80	IO12	84	MOSI
86	MISO	90	nCS
92	SCK	96	IO13
98	IO14	102	IO15
104	IO16	108	IO17
110	IO18	114	IO19
116	IO20	120	IO21
122	IO22	126	IO23
128	IO24	132	IO25
134	IO26	138	IO27
140	IO28	144	IO29
146	IO30	150	IO31
152	IO32	156	IO33
158	IO34	157	IO35
155	IO36	151	IO37
149	IO38	145	IO39
143	IO40	139	IO41
137	IO42	133	IO43
131	IO44	127	IO45
125	IO46	121	IO47
119	IO48	115	IO49
113	IO50	109	IO51
107	IO52	103	IO53
101	IO54	97	IO55
95	IO56	91	IO57
89	IO58	85	IO59
83	IO60	79	IO61
77	IO62	73	IO63
45	VCCIO	51	VCCIO
57	VCCIO	63	VCCIO
69	VCCIO	75	VCCIO
81	VCCIO	87	VCCIO
93	VCCIO	99	VCCIO
105	VCCIO	111	VCCIO
117	VCCIO	123	VCCIO
129	VCCIO	135	VCCIO
141	VCCIO	147	VCCIO
153	VCCIO	159	VCCIO
161	GND	162	GND
163	GND	164	GND
165	GND	166	GND
167	GND	168	GND
169	GND	170	GND
171	GND	172	GND

TABLE 2: Pin-outs for the HSMC header on B2 board

Bibliography

Altera Corporation. High Speed Mezzanine Card specification, June 2009. Revision 17.

Altera Corporation. *Cyclone III Device Handbook*, volume 1. December 2011.

Spansion. Connecting Spansion SPI Serial Flash to Configure Altera FPGAs, 16 November 2011. Revision 4.

ST Microelectronics. LD1117xx datasheet, Adjustable and fixed low drop positive voltage regulator, 13 February 2012. Datasheet.

Texas Instruments. TPS782 150mA, Ultra-low Quiescent Current, Low-Dropout Linear Regulator Datasheet, September 2008.

TXC. Oscillators 7X5mm SMD CMOS CXO 7W Series Datasheet.