DESIGN AND DEVELOPMENT OF JTAG USING RASPBERRY PI

Pilavendiran S.A     Keshavarajan J.L     Dr.S.Vijayakumar
Department of ECE     Department of ECE     Department of ECE
Paavai Engineering College     Paavai Engineering College     Paavai Engineering College
Namakkal, Tamilnadu, India     Namakkal, Tamilnadu, India     Namakkal, Tamilnadu, India

Abstract— A motherboard contain a different microprocessor, microcontroller and power consumer. This are all working under a different BIOS program the individuals work with unique programming languages, so it is very complex to program each microprocessor, microcontroller, FPGA and etc., present in a motherboard and cost to write the individual programs into concern chips is very high and the time to write BIOS program into each chips is long in existing system. Apart from long duration and high cost the debugging process takes place in individual chips thus to overcome all those drawbacks present in existing system. We use our project as a single board computer. This single board computer is combination of raspberry pi along with JTAG through LINUX operating system. The main goal of our project is to write BIOS program in various different chips present in motherboard through JTAG simultaneously and the programming cost is very less compared with current existing method and the total execution time is comparatively less than existing system.

Keywords— Raspberry Pi, JTAG (Joint Test Action Group), GUI, Actel FPGA, STM32 Microcontroller

I. INTRODUCTION

Nowadays many industries manufacture millions of motherboards, in each target motherboard there are many types of microprocessor, microcontroller, FPGA are present in it and they have a separate BIOS (binary input and output) programs. BIOS means getting ready the chip to load or to execute a firmware without error. The chips are manufactured and loaded with this is why we go for raspberry pi along with JTAG for their fast computation time and error check using boundary scan technique. BIOS programming software for each microprocessor, microcontroller and FPGA are unique and we need to buy those programming separately with the corresponding company thus the cost is very high, so we go for raspberry pi with JTAG interface to reduce the cost of buying programming software by storing the BIOS programming software in the internal memory of the raspberry pi. The JTAG cable is used to form a bridge between the raspberry pi and the target motherboard which consist of several microprocessor, microcontroller and FPGA. The transfer time by using JTAG debugger is less while compared to other programmers. The auto error detection and correction is achieved using JTAG cable simultaneously debugging is achieved by it. Raspberry pi performs faster and it act as single board computer, so raspberry pi is used to replace normal computers. Thus the industries save lots of money by replacing computers with raspberry pi board.

II. PROPOSED ALGORITHM

A. Raspberry pi –

The Raspberry Pi is used around the world for it’s portability and efficiency. The Raspberry Pi is used as single board computer to perform all the operations at a faster rate. The Raspberry Pi contains 2 GPIO pins and those pins allow JTAG to connect with Raspberry Pi. The Raspberry Pi is having inbuilt RAM(Random Access Memory) and ROM(Read Only Memory).

The Raspberry Pi stored the BIOS programs in it’s internal storage to access at any time. The Raspberry Pi performs every operations computer does for professional work purpose. The Raspberry Pi execute operations at a faster rate while compared to normal computers. The Raspberry Pi is a serial of small single-board computer.

Fig. 1. Raspberry Pi board
Raspberry Pi is having 5V, 3A, full power delivery to USB devices. It is having a CPU of 1.5GHz, 64/32-bit quad-

75
core ARM cortex-A72. It has SoC of BCM2837 and Central Processing Unit of Quad Cortex A53 @1.2GHz.

Table 1: Raspberry Pi GPIO configuration for JTAG

<table>
<thead>
<tr>
<th>Register</th>
<th>Pin</th>
<th>Configuration</th>
<th>JTAG signal</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPFSEL0</td>
<td>GPIO4</td>
<td>ALT5</td>
<td>TDI</td>
</tr>
<tr>
<td>GPFSEL2</td>
<td>GPIO22</td>
<td>ALT4</td>
<td>TRST</td>
</tr>
<tr>
<td>GPFSEL2</td>
<td>GPIO24</td>
<td>ALT4</td>
<td>TDO</td>
</tr>
<tr>
<td>GPFSEL2</td>
<td>GPIO25</td>
<td>ALT4</td>
<td>TCK</td>
</tr>
<tr>
<td>GPFSEL2</td>
<td>GPIO27</td>
<td>ALT4</td>
<td>TMS</td>
</tr>
</tbody>
</table>

Raspberry Pi is having instruction set of ARMv8-A. It has GPU of 400 MHz Video Core IV and the RAM (random access memory) of 1GB SD RAM. It has Micro-SD slot present in it with Ethernet 10/100. The wireless is containing 802.11n/Bluetooth 4.0 and Video output of HDMI/Composite. Audio output of HDMI/Headphone.

B. JTAG

JTAG is abbreviated as Joint Test Action Group. It’s an industry standard for verifying designs and testing printed circuit boards after manufacture. It’s a serial communication interface for low overhead access without requiring direct external access to the system address and data buses. It’s able to access to all chips on a target circuit mother board.

JTAG is having four primary pins they are given as follows

- TDI
- TDO
- TMS
- TCK
- TRST

TDI is abbreviated as Test data input
TDO is abbreviated as Test data output
TMS is abbreviated as Test mode select
TCK is abbreviated as Test clock
TRST is abbreviated as Test reset

Fig. 2. GPIO-Raspberry Pi

C. GUI

The GUI stands for graphical interface may be a interface that allows user to interact with electronic devices through graphical icons and audio. A GUI uses a mixture of technologies and devices to supply a platform that users can interact with, for tasks of gathering and producing information.

In our project GUI is employed to make a decision that which program must install during which chip. Thus programming of multiple chips is formed possible without error thanks to presence of GUI (graphical user interface). There’ll be no trouble in wrongly programming multiple chip because GUI decides program to be installed in each chip set present during a target motherboard.

GUI is used to create the graphical user interface to execute the BIOS program into chip. The execution is done in LINUX operating system present within the Raspberry Pi. Through JTAG the BIOS program is transferred into chip and the desire of choosing BIOS program to write in chip is done in GUI.
D. Linux kernel

This section describes the way to enable JTAG on RPi for debugging Linux kernel. Linux may be a default choice as a general purpose OS for RPi [4]. Kernel debugging isn’t a trivial task, but doing it remotely simplifies the approach tons. By default, JTAG isn’t enabled within the kernel, so special steps are needed before the kernel are often debugged over JTAG.

The kernel compilation for RPi follows similar steps as for x86 processors, for instance. Like bare metal programs, the compiler should be configured to create programs for ARM processors.

The kernel expects to be loaded at the address 0x8000. But usually the kernel image is placed on the booting partition during a compressed form as (zImage) alongside decompress or. A decompress or is ignorant to where it’s loaded to, and it's smart enough to uncompressed the kernel and put it to the right memory address. This is often important because when building a picture for the RPi which will enable JTAG before booting the kernel, the particular zImage are going to be shifted in memory to introduce the JTAG enabling code ahead of it.

The RPi community provides the script to make the image kernel.img which will be put to the cardboard from a newly compiled Linux kernel. The script puts a little bootloader at the address 0x0, the kernel arguments at the address 0x100 and zImage at 0x8000. The bootloader and therefore the arguments are provided as text files with hexadecimal strings most likely converted from binary files.

Following an equivalent idea, it's possible to convert the bare metal JTAG enabling program into a document with the specified hexadecimal strings and put it at 0x8000, in order that JTAG gets enabled first. During this case, zImage is appended then code.

In order to be ready to debug the kernel booting process from the very beginning, the JTAG enabling software can enter an infinite loop, in order that the kernel booting should be started manually: by opening an OpenOCD session, halting the execution, and jumping to the address, where zImage ended up being loaded to.

So, the kernel are often debugged over JTAG from the very beginning of the booting process. But during its execution some kernel drivers might override the GPIO pins used for JTAG for his or her purposes. All GPIO JTAG pins have some alternative configurations for other functions. Possible overriding candidates are I2 C (Inter-Integrated Circuit) and SPI (Serial Peripheral Interface) drivers. Neither I2 C, nor SPI signal pins overlap with those of JTAG. Quite that, both I2 C and SPI drivers are disabled within the kernel by default.

E. OpenOCD

The OpenOCD is an open-source tool for debugging embedded systems. It runs on the host machine as a server, allowing connections over Telnet or from GDB (The GNU Project Debugger), so as to debug the target systems, connected to the host machine through a JTAG adapter.

Since there are many variant of the adapters, also as many JTAG implementations, OpenOCD should be configured to figure with a specific adapter and therefore the board.

III. Experiment and Result

The Raspberry Pi contains the BIOS programs for each microprocessor, microcontroller, FPGA present in a target motherboard. By using GUI (graphical user interface) the BIOS programs are instructed to install into concern chips it can be either microprocessor, microcontroller, FPGA. Thus JTAG transfer the data into respective chip sets present in the target motherboard simultaneously.

Therefore, the debugging is achieved through JTAG debugger and programming of multiple chips is made possible in a less amount of time, minimum power consumption and
low cost through JTAG using Raspberry Pi in Linux operating system.

IV. ANALYSIS TABULATION

<table>
<thead>
<tr>
<th>Device name</th>
<th>SuperPro 610P</th>
</tr>
</thead>
<tbody>
<tr>
<td>Device supported</td>
<td>EPROM, Paged EPROM, Parallel and serial EEPROM, BPROM, NVRAM, SPLD, CPLD, EPLD, Firmware HUB, Microcontroller, MCU</td>
</tr>
<tr>
<td>PC interface</td>
<td>USB 2.0</td>
</tr>
<tr>
<td>Power supply</td>
<td>AC adapter: Input AC 100V-240V Output: 12V/1.5A</td>
</tr>
<tr>
<td>Speed</td>
<td>14 Seconds</td>
</tr>
<tr>
<td>PIN</td>
<td>48-pin</td>
</tr>
<tr>
<td>Programming Testing features</td>
<td>TTL/CMOS logic ICs and memories</td>
</tr>
<tr>
<td>Compatible</td>
<td>Windows XP/Vista/7/8/10 (32/64 bit)</td>
</tr>
<tr>
<td>Cost</td>
<td>35,289 Rupees</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Connector Type</th>
<th>JTAG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Device supported</td>
<td>All embedded systems</td>
</tr>
<tr>
<td>PC interface</td>
<td>Bluetooth 5.0, USB Type C</td>
</tr>
<tr>
<td>Operating temperature</td>
<td>0-50 degrees C ambient</td>
</tr>
<tr>
<td>Speed</td>
<td>10x faster than existing system</td>
</tr>
<tr>
<td>PIN</td>
<td>40-GPIO pin</td>
</tr>
<tr>
<td>RAM</td>
<td>4 GB</td>
</tr>
<tr>
<td>Processor</td>
<td>Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz</td>
</tr>
<tr>
<td>Compatible</td>
<td>Linux</td>
</tr>
<tr>
<td>Cost</td>
<td>5000 Rupees</td>
</tr>
</tbody>
</table>

V. CONCLUSION

The debugging and programming of several microprocessors, microcontrollers and FPGA present in target motherboard is processed simultaneously with less consumption of time. The cost of programming is reduced several times compared to existing method due to pre-stored program present in internal storage of raspberry pi and auto error correction is achieved through JTAG debugger.

VI. REFERENCE


