CALIFORNIA STATE UNIVERSITY, NORTHRIDGE

COLLEGE OF ENGINEERING AND COMPUTER SCIENCE

Zybo-Z7-10 Hardware Accelerator Platform

Sage Vigil

April 14th 2022

Final Project ECE 520

Submitted to: Professor. Saba Janamian

This report examines creating a hardware accelerator, built on the ZYBO Z7-10 board. It contains theoretical information along with step-by-step instructions allowing the reader to follow along and create an IP. It is assumed that the reader has a general understanding of C programming language, Github, Windows operating system, Linux commands, FPGA, PetaLinux and its associated parameters.

This report is part of the authors' academic project for ECE 520, for the SPRING 2022 semester assigned by Professor. Janamian. It is to be known that this report is built upon the works of the following authors: Sage Vigil

TABLE OF CONTENTS

SECTION I – INTRODUCTION

This project's goal is to create a hardware accelerator platform for the Zybo-Z7-10 board. A hardware accelerator is using specifically built hardware to perform a highly specific task.The benefits of using a hardware accelerator over a software is that a hardware is generally faster at performing said task. Some more benefits include reduced power consumption,lower latency, increased parallelism and bandwidth. Software based accelerators have the advantage of rapid development,lower cost and ease of future development.

Some common applications of hardware acceleration include BIT BLIT which is related to acceleration tasks in GPUs. Spam control applications in web servers more specifically prevention of (DoS) attacks. Below is an example of an applied hardware accelerator used in a medical application:

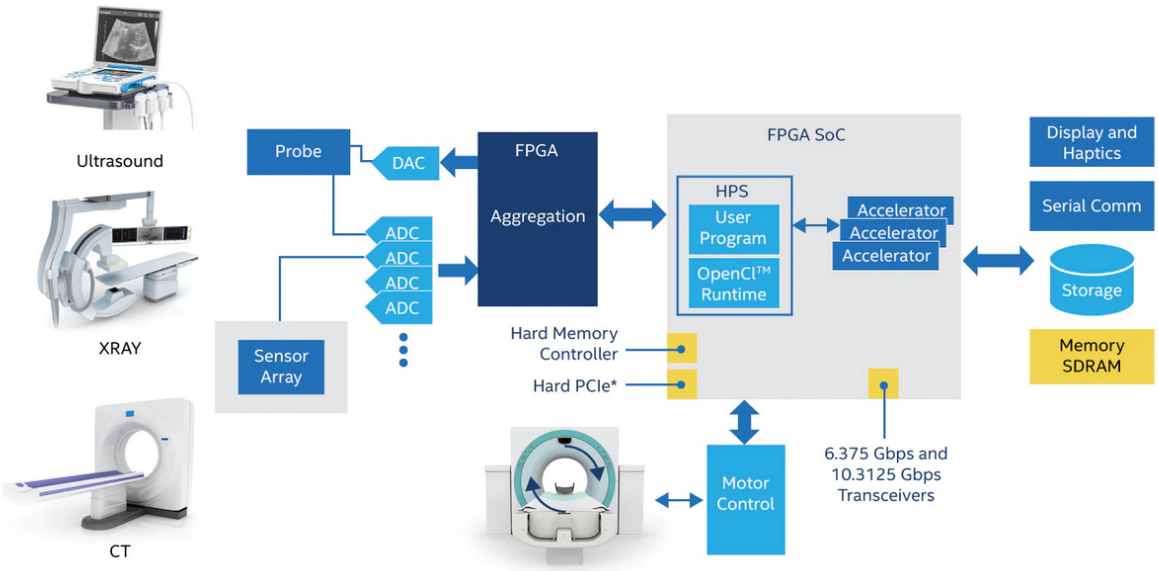


Figure 1.0: Hardware Acceleration Application

[1]"FPGAhardwareacceleration-intel®FPGA," *Intel*.[Online].Available:https://www.intel.in/content/www/in/en/healthcare-it/products/programmable/applications/hardware-acceleration.html. [Accessed: 27-Apr-2022].

The above image shows several hardware intensive machines that require a high degree of processing. Therefore these devices offload the host processor to the FPGA. Resulting in an increase in computational capabilities and improved power.

- **SECTION II – SYSTEM DESIGN**
  - A Part 1: Hardware

This subsection (Part 1 Hardware) will highlight the process of using Vivado to create the initial hardware components and ultimately generate a usable XSA file.

Start by confirming that both a recent copy of Vitis and PetaLinux are installed in the hosts operation system. Once confirmed create both a folder in your workspace and within that created folder directories (hardware software platform). Lastly create a folder (linux-files) within the software sub folder. See Following commands:

```
$ cd  ~/workspace
$ mkdir zybo_z7_10_base_2021_2
$ cd zybo_z7_10_base_2021_2
$ mkdir hardware software platform
$ cd software/
$ mkdir linux-files
$ cd linux-files/
$ mkdir boot
```

Startup Vivado and create a new project and choose the desired board.Now start by creating the custom block design. Add the Zynq IP, run block automation. Once completed add a clocking wizard IP, and add five clocks in the output clocks window. Further customization to the clock is needed. Reset type must be “**ACTIVE LOW**”. See figure:

Figure 1.1: Clocking Wizard

Now add five “**Processor System Reset IP’s**”. Connect all of the following IP’s as shown in the following figure:

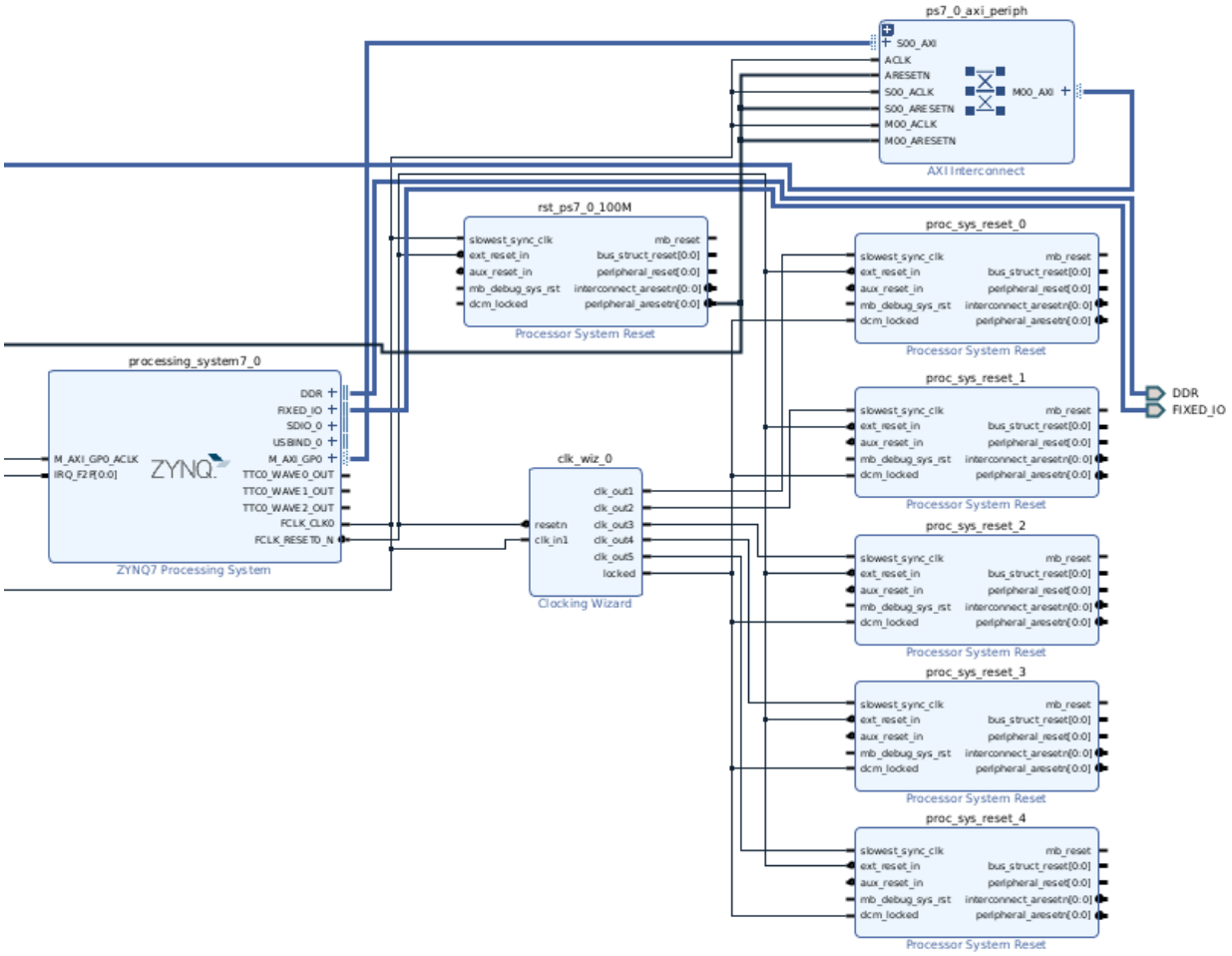


Figure 1.2: Hardware Acceleration Diagram

Note: You must enable IRQ_F2P interrupts on the ZYNQ7 Processing System

Moving along we can now add an “**AXI interrupt controller**”, and change the output connection of the controller to single.Next we can navigate to the “**Window Platform Setup**” tab in order to configure the following ports and SP tags: “**M_AXI_GP1 and S_AXI_HP0 to S_AXI_HP3**”. Also in the clock tab change the clock rates as follows:

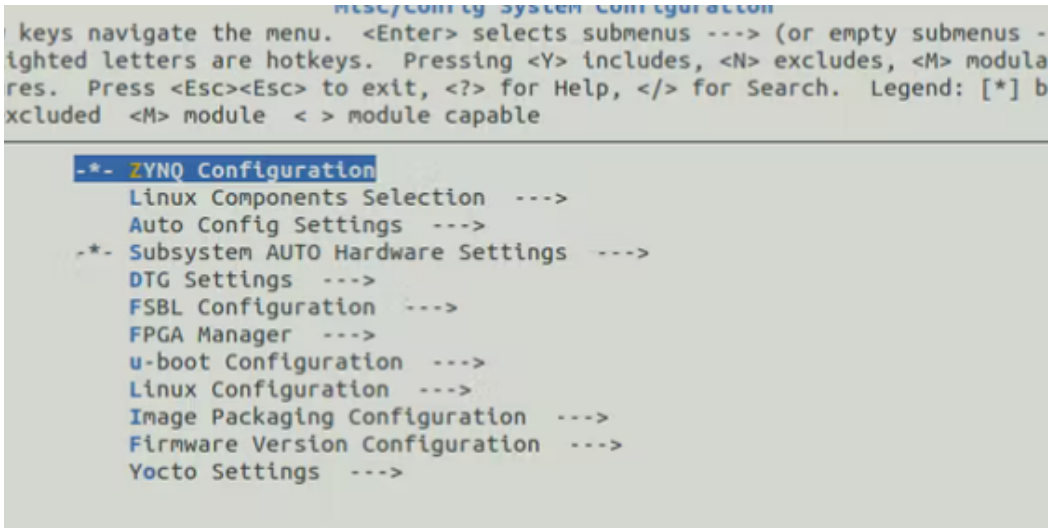
Figure 1.3: Platform Clock Settings

Lastly set “**SELECTED_SIM_MODEL**” to tlm, this allows vitis hardware emulation to be used. Once complete Generate the output products and generate the HDL wrapper, Then the Bitstream file.

- B Part 2: Software

We will now transition to the software side of the project by utilizing PetaLinux. Some background information on PetaLinux is important. Petalinux allows for the creation of embedded systems for specific hardware applications. This project in particular will configuring the kernel, and rootfs.

Start by running the PetaLinux tool by sourcing the `setting64.sh.` Create a new project by using the command: `petalinux-create -t project --template zynq -n <PROJECT NAME > -petalinux` Then move into the newly created directory and run the command: `petalinux-config--get-hw-description=../../hardware/<PROJECT NAME>-vivado` Once ran the “**misc/config system configuration”** widow should appear as shown:



Next we will configure the Kernel. The kernel can be thought of as the main driver of the OS. It is loaded right after the bootloader. It handles such tasks as disk management,memory, etc. Most importantly it is the communication between the user and the hardware components. To configure the kernel run the command: `petalinux-config -c kernel` a window will appear, accept the default values and save. Now there are several files that need to be modified. First open the file located at using the command: `gedit ./project-spec/meta-user/conf/user-rootfsconfig` this will open up the rootfsconfig folder which handles the rootfs configuration. Add the following lines: CONFIG_xrt

```
CONFIG_xrt-dev
CONFIG_zocl
CONFIG_opencl-clhpp-dev
CONFIG_opencl-headers-dev
CONFIG_packagegroup-petalinux-opencv
```

Lastly we need to add the boot.src file that can be found “ https://github.com/highlevelsynthesis/zyboz720-platform/raw/main/boot.scr?_ga=2.144240309.106171838.1650860675-1901173077.1647199174 “

- C Part 3 Platform

We will now explore the process of creating the platform folder within the Vitis workspace. Start by creating a new platform project, and locating the generated XSA file and selecting the linux OS and correct processor. Next we need to modify the paths of several default Domain options. See figure:

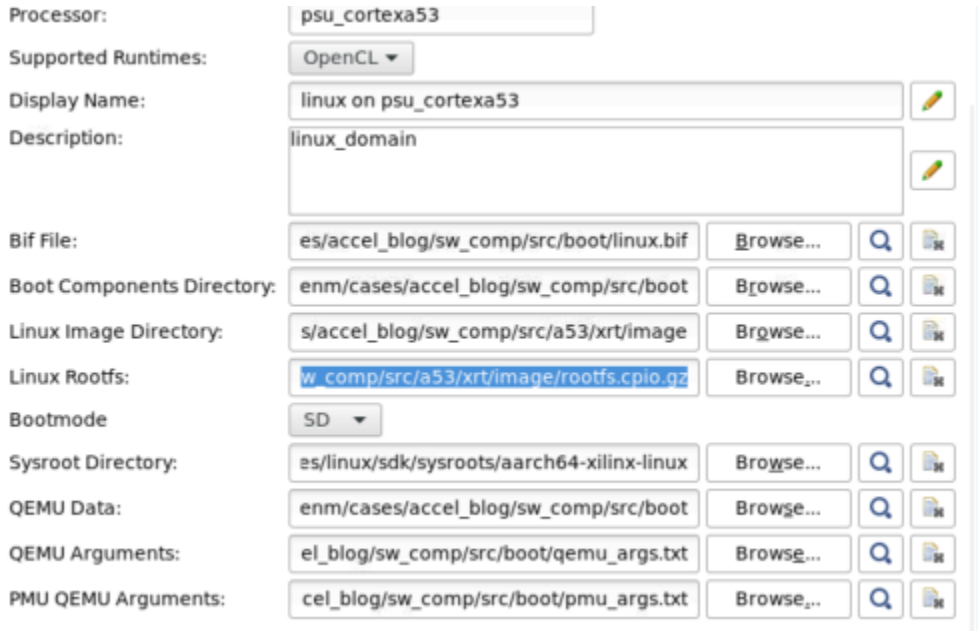


Figure 1.5: Linux Domain Folder Path Configuration

Once formatted, build the project and proceed to creating another new application project. Select the newly created platform and select **vector addition** as a template.

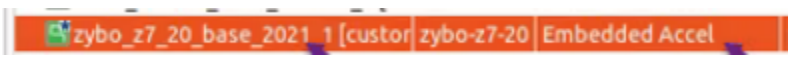


Figure 1.6: Application Platform

At this point you can decide to run this project on hardware or an emulator. I decided to run the application on the physical hardware. To do this first source an SD card, and I recommend using GParted to format your SD card.

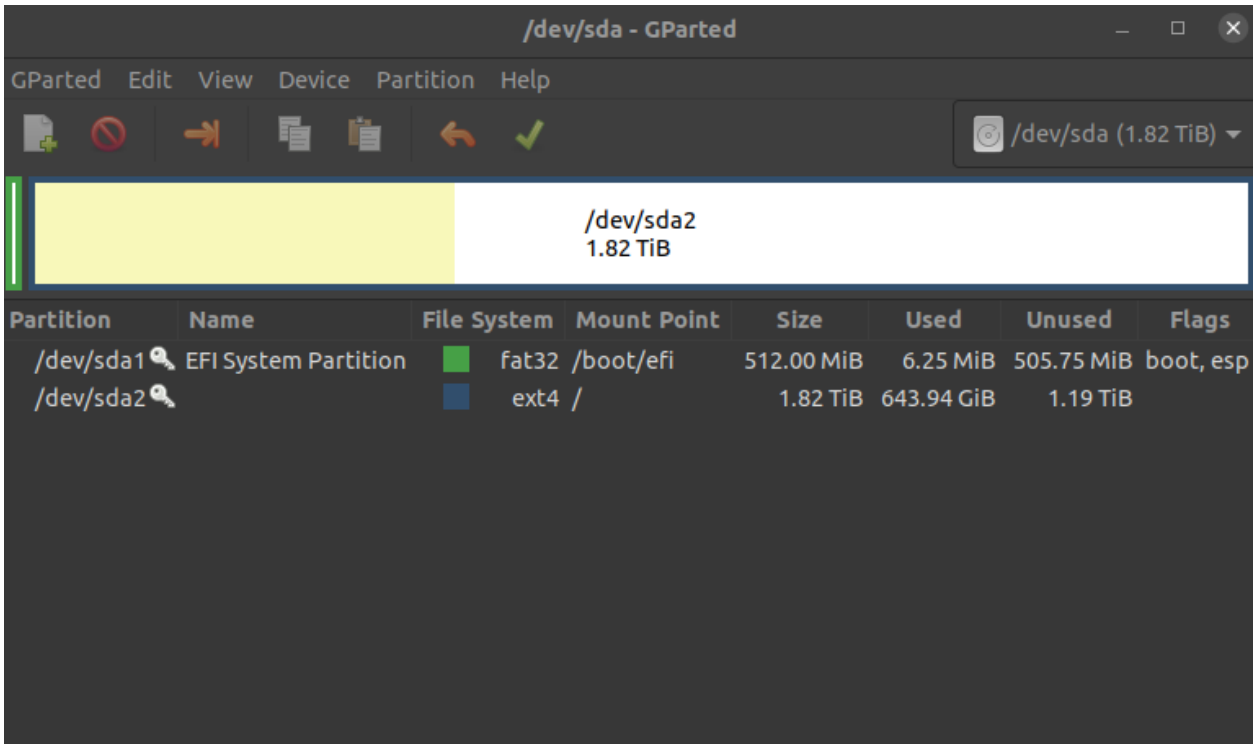


Figure 1.7: GParted SD Formatting

To use GParted select in the upper right corner your SD card, then select **partition > Resize > Format .** Format the card to FAT32 file format. Once formatted you must copy the following program files to the card: `$ cp binary_container_1.xclbin BOOT.BIN boot.scr image.ub vector_addition /media/numvar/BOOT/` Once completed you can run the application. I recommend using GTKTerm for the serial communicator. Once you have serial connected your board to the computer run the application by using the command:
`$ ./vector_addition binary_container_1.xclbin`

Figure 1.8: PetaLinux Test Pass

SECTION III -- CONCLUSION

This simple demonstration showed how easy it is to create a simple hardware accelerator application within vivado. Although a basic vector addition program was used to showcase the potential. There are far more advanced applications that acceleration can be applied to. Another possible application could be a Support Vector Machine: https://projects.digilentinc.com/mohammad-hosseinabady2/support-vector-machine-on-zynq-zynqmpsoc-zybo-ultra96v2-5ba6d3

SECTION V – ACKNOWLEDGEMENT

This paper and the research behind it would not have been possible without the exceptional support of my Professor, Janamian. His enthusiasm, knowledge and determination to teach have been an inspiration and kept my work on track.

SECTION VI – REFERENCES

1. sleibso, “Introduction to the Zynq Triple Timer Counter Part Four: Adam Taylor’s MicroZed Chronicles Part 20,” *Xilinx Customer Community*. [Online]. Available: https://support.xilinx.com/s/article/416243?language=en_US. [Accessed: 30-Apr-2022].
2. S. Janamian, “Interrupts,” *Google*. [Online]. Available: https://docs.google.com/presentation/d/1pjsGZdrQyGiuueZElqCHzTEBV59y_yQQWZxfCCE7j8M/edit#slide=id.g11dd3c3fcb9_0_38. [Accessed: 30-Apr-2022].
3. F. K. | www.fpgakey.com, “Zynq-7000 series three-Way Timer (TTC) explained,” *FPGAKey*. [Online]. Available: https://www.fpgakey.com/technology/details/zynq-7000-series-three-way-timer-ttc-explained. [Accessed: 30-Apr-2022].
4. A. Taylor, *Xilinx Customer Community*. [Online]. Available: https://support.xilinx.com/s/article/413105?language=en_US. [Accessed: 30-Apr-2022].
5. A. Taylor, *Xilinx Customer Community*. [Online]. Available: https://support.xilinx.com/s/article/413105?language=en_US. [Accessed: 30-Apr-2022].

Notes

- Several screenshots/snippets for the basic concepts (such as, adding IP blocks), in SECTION IV:Implementation (tutorial) have been obtained through Mr. Saba Janamian's lecture recordings since the report was made after the design was implemented and run. However, all images containing boards, wires, results, codes, video demonstration and other major parts of the tutorial where we had reached completion and could revisit that part for the report, are the author's own property.