Spring 2019, Real-time Embedded Systems June 2019

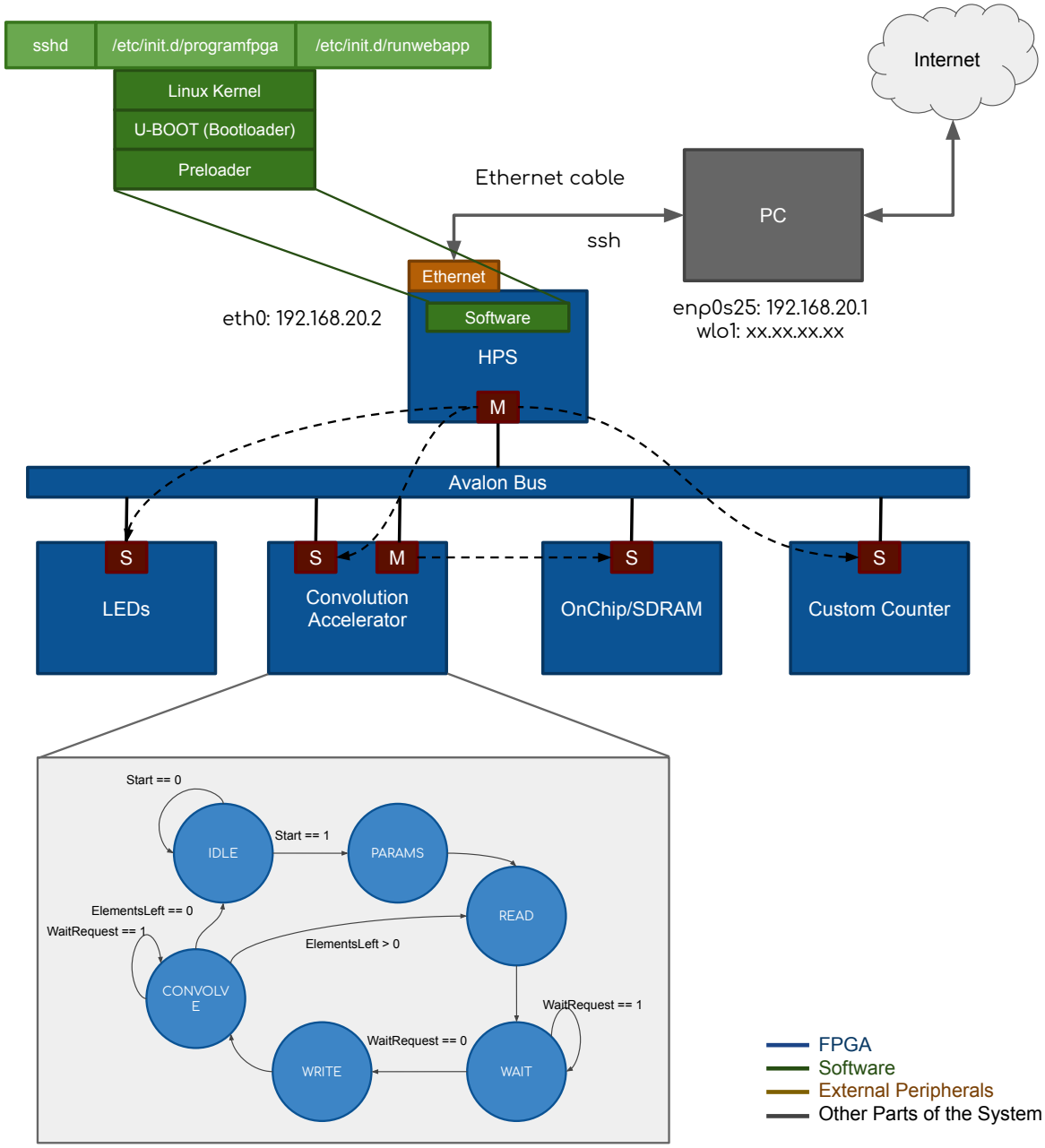
2D Convolution Acceleration

Pinar Ezgi Çöl & Darko Lukić

Professor: Beuchat René

Contents

1 Summary


sshd
/etc/init.d/programfpga
/etc/init.d/runwebapp
Linux Kernel
U-BOOT (Bootloader)
Preloader
Internet
Ethernet cable
ssh
PC
Ethernet
Software
HPS
eth0: 192.168.20.2
enp0s25: 192.168.20.1
wlo1: xx.xx.xx.xx
M
Avalon Bus
S
LEDs
S
M
Convolution Accelerator
S
OnChip/SDRAM
S
Custom Counter
Start == 0
IDLE
Start == 1
PARAMS
READ
ElementsLeft == 0
WaitRequest == 1
ElementsLeft > 0
CONVOLV E
WaitRequest == 1
WaitRequest == 0
WRITE
WAIT
FPGA
Software
External Peripherals
Other Parts of the System

2 Introduction

In this project, we aimed to develop a hardware and software system which performs the convolution operation on the images for edge detection. The system is implemented in both C language and with a hardware accelerator. In addition, a custom counter is a part of the system in order to measure the performance.

Linux is configured to run on the HPS processor (with DDR3 memory) and main goals of the Linux are:

- Execute the C implemented convolution in the user space.
- Communicate to the convolution accelerator.
- Run a web server with a simple user interface.
- Uploading Raw Binary File (.rbf) to the FPGA.
- Provide a comfortable development environment.

In the remaining parts of the report, we explain about the configurations made to operate system fully. Then, system architecture is demonstrated. In the next sections, results of the both hardware and software applications are visualized and listed. Finally, we conclude the report.

3 Architecture

3.1 System Design

System design is demonstrated in Fig. 1. In the system, Convolution Accelerator has both master and slave behaviour. It behaves as slave to the HPS and master to the SDRAM/OnChip. For debugging purposes, the HPS controls the LEDs. Counter is used for performance measurements of the system.

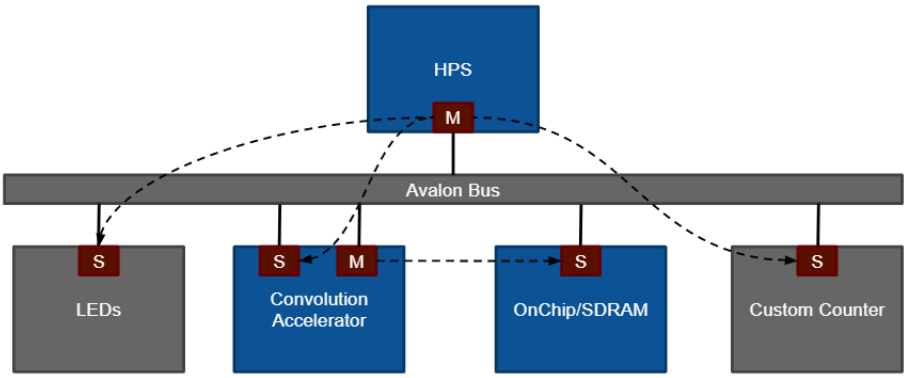


Figure 1: System Design

In addition, the board is connected to the external network using Ethernet port. Benefits of having the Ethernet connection are:

- Access to the Internet - easy way to update and install packages,
- Easy debugging using SSH,
- Mounting a filesystem from the board to a PC - easy access to the code.
- Opportunity to create web server - easy access for the end user.

The implemented design from Fig. 1 is available in the following picture (Fig. 2).

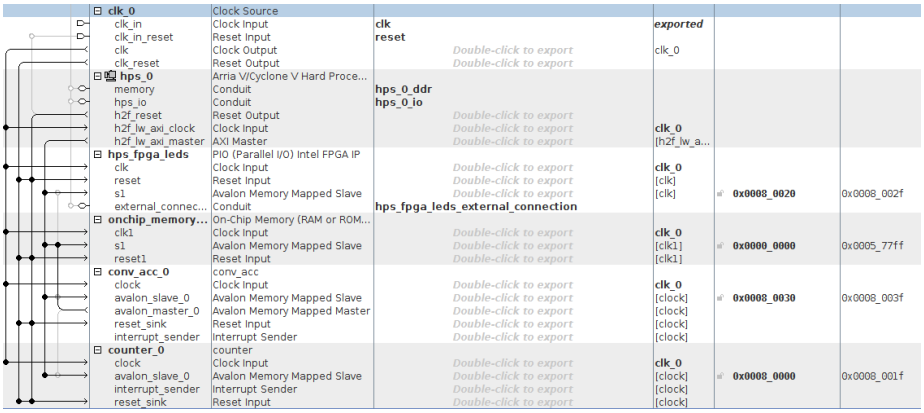


Figure 2: System Design in Qsys

As shown in Fig. 2, OnChip memory is used for storing the image.

3.2 Convolution Accelerator

In this part, we explain how Convolution Accelerator module is integrated with the whole system and our design choices over the convolution operation. We chose to use Roberts Filter as the convolution filter which is demonstrated in Fig 3. First, Linux part writes into the memory (with C language) by taking a window in size 2x2 same as Roberts Filter. This window corresponds to 4 bytes(1 byte for each number). The window is written into memory in a linear way 4. The reason why we have chosen this specific design is that accelerator is reading 4 bytes at time. After the convolution operation is done in the accelerator, it is written back into the memory and again it is recovered back into the its original size in C language.

Reg. Addr.	Name	R/W	Size	Function
0	RegAddStart	W/R	32	Register Address Start
1	RegLgt	W/R	16	Length of the array in which operation is performed
2	Start	W/R	1	Starts the convolution
2	Finish	R	1	Reads if the convolution is finished
2	iIRQEn	R	1	Reads if the interrupt is enabled
3	iIRQEn	W	1	Enables the interrupt
3	ClearFinish	W	1	Resets the finish signal

Table 1: Register Map of the Convolution Accelerator

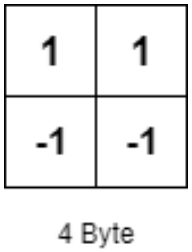


Figure 3: Roberts Filter

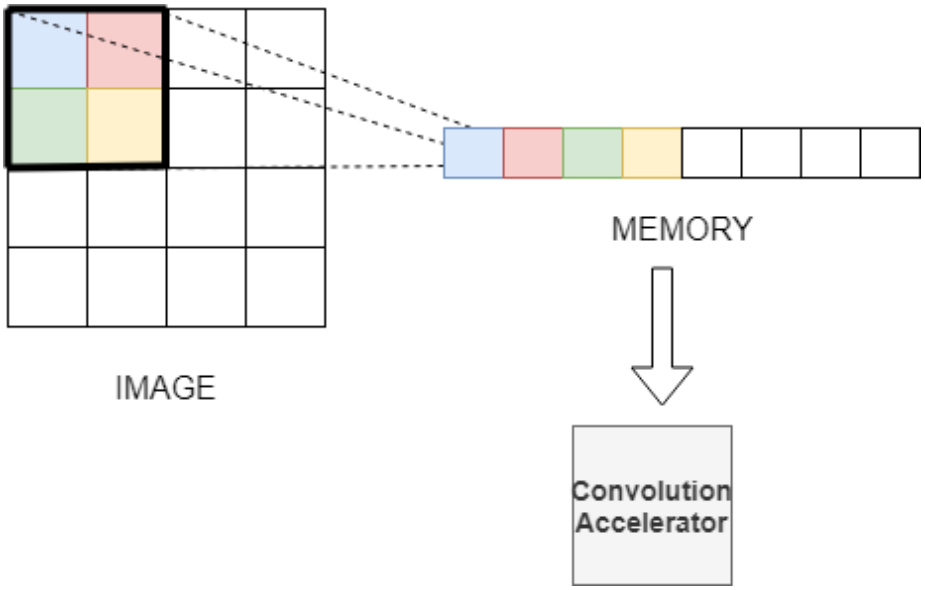


Figure 4: Memory Design for Convolution Accelerator

3.2.1 Finite State Machine

Finite state machine (FSM) implement in the convolution accelerator is shown in the Fig. 5.

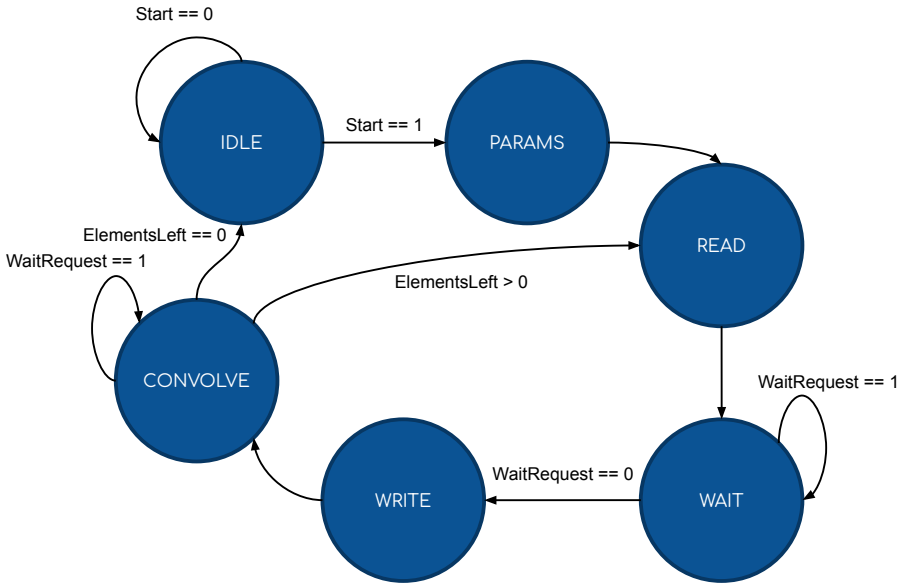


Figure 5: Convolution Accelerator FSM

By default the accelerator is state **IDLE**. After start signal received the accelerator starts initialization by loading number of elements and an address of the first element (state **PARAMS**). Unconditionally, the state is changed to state **READ** in which the accelerator initializes reading process and changes state to **WAIT**. It stays in state **WAIT** until the require data is received and the state is then changed to **WRITE** in which that accelerator initializes write process. Finally, next state is **CONVOLUTION** in which convolution and write to the memory is performed. Depending if the convolution accelerator reached the end of the buffer size it changes the state to **IDLE** or **READ**.

3.2.2 Communication with Peripherals

Our C application is running in the user space of Linux. This forbids it to have a direct access to memory addresses reserved for FPGA peripherals. Since we need access to the convolution accelerator and OnChip memory we were able to use two approaches, to create a kernel module or to use */dev/mem* device. Using */dev/mem* requires a root access from user space, but we still chose the method as it is more simple approach.

Functions *mmap()* and *munmap()* from *sys/mman.h* allow an easy interaction with */dev/mem*. Using those functions it was easy to map the required address of OnChip memory and the convolution accelerator to the user space.

3.3 Web Application

After the C application is implemented web server is built to facilitate the usage of the C application.

Figure 6: Screenshot of the web application

Web application handles image uploads. After an user uploads the image, the image is stored on SDCARD and a system call to the C application is made. The C application accepts input filename, C convolved filename and FPGA convolved filename, and it returns details about the convolution. All results are loaded (image convolved in C and FPGA, and details about the convolution) and shown back to the user.

In order to access the web application and allowing the application to be visible to the Internet configuration described in Section 6.1 is applied (public IP address is not necessary, tunnelling to the localhost can used, e.g. ngrok).

3.3.1 Linux

Linux boots from a SDCARD, but first the preloader boots U-BOOT (which is chosen as Bootloader) from the SDCARD. U-BOOT starts Linux kernel on which Ubuntu is based. SystemV loads many services automatically on the boot, but the most important for the context are shown in Fig. 7 in blue.

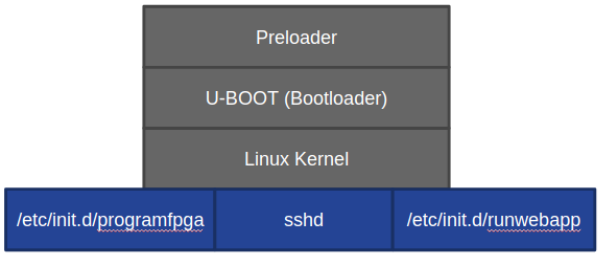


Figure 7: Boot sequence of the Linux

Purpose of the services is the following:

- **/etc/init.d/programfpga**: Loads the Raw Binary File to the FPGA (see Section 6.2)
- **sshd**: Starts SSH server.
- **/etc/init.d/runwebapp** : Start the web application (described in Section 3.3)

4 Results

In this sections, the results of the system is demonstrated. In Fig. 8b, image is convoluted in C language, whereas Fig. 8c is convoluted with the Convolution Hardware component. There are minor differences between two outputs because of normalization of the images before and after the convolution operators.

(a) (b) (c)

Figure 8: (a) is the original image, (b) is convoluted in C and (c) is convoluted in accelerator

In the following table (Table 2) performance comparison between FPGA and C implementation are given.

Image Size	C implementation	FPGA implementation
128x128	472,654 clock cycles (9.5ms)	29,033 clock cycles (2.5ms)
256x256	1,913,501 clock cycles (38ms)	520,186 clock cycles (10.5ms)

Table 2: Performance comparison between FPGA and C implementation

According to the measurements implementation in FPGA is about 3.7 times faster. Note that FPGA performance is measured after the image is preprocessed (the process explained in Section 3.2).

5 Conclusion

In this project we have acquired a knowledge of running and configuring Linux to run on the development board. Also, we have learned how communicate to peripherals from the Linux user space using */dev/mem*.

Convolution is implemented C and VHDL and the performances are measured. The performances are measured for different image sizes and the speed improvement is about 3.7 times. Therefore, acceleration is achieved. In addition, web server is created to enable easy access to the end user.

However, on one hand, we should also consider that HPS has advantage in terms of frequency (more cycles per second) compared to accelerator. Also, accelerator reads from OnChip memory while the HPS reads from DDR3 which is faster. On the other hand, the time measurement for accelerator doesn't include prepossessing and C application is compiled with a default optimization level (*-O0*).

Additional improvement of the accelerator can be made by adding IRQ which would raise an interrupt when the processing is done. Also, the accelerator can be extended to accept arbitrary size of convolution kernel.

Overall, this acceleration can be particularly useful in cases where multiple image filters are applied sequentially, e.g. Gabor filter banks. In the cases with multiple image filters prepossessing time is negligible compared to total processing time.

References

[1] Sahand Kashani-Akhavan. *Tutorial for using the DE1-SoC/DE0-Nano-SoC boards for bare-metal and linux programming: sahandKashani/SoC-FPGA-Design-Guide*. original-date: 2016-04-16T17:16:36Z. May 2019. URL: `https://github.com/sahandKashani/SoC-FPGA-Design-Guide` (visited on 06/09/2019).

[2] "Using Linux on the DE1-SoC". en. In: (2015), p. 23.

6 Appendix

6.1 Network Configuration

We wanted to have a network connection between our PC and the development board, as well as Internet access from the board. However, since we didn't have a switch, we connected directly PC and the board using ordinary Ethernet cable. The following configuration was required.

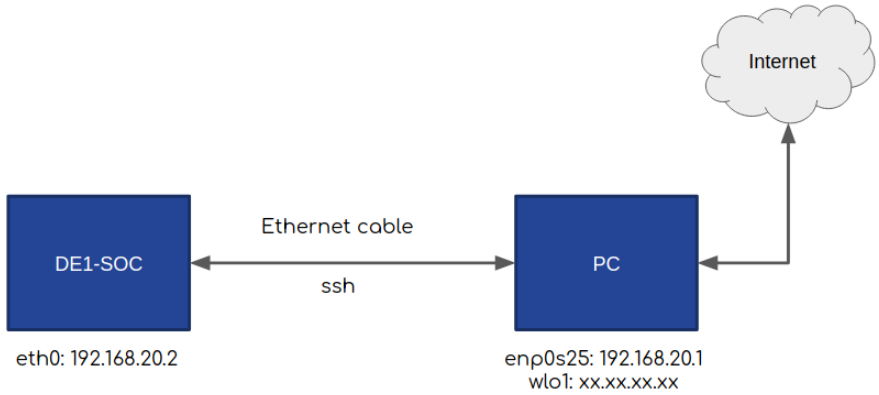


Figure 9: Description of the network

6.1.1 PC Configuration

Both devices have to have static IP addresses since DHCP server is not available, therefore we configure a static IP address on our PC.

```
$ ip ad add 192.168.20.1/24 dev enp0s25
```

Now, we have to "pass" packages from interface with Internet connection (e.g. wlo1) to Ethnernet interface (e.g. enp0s25).

```
$ sysctl -w net.ipv4.ip_forward=1
$ iptables -t nat -A POSTROUTING -o wlo1 -j MASQUERADE
$ iptables -A FORWARD -i wlo1 -o enp0s25  -m state --state ESTABLISHED,RELATED -j ACCEPT
$ iptables -A FORWARD -i enp0s25 -o wlo1 -j ACCEPT
```

After FPGA configuration is done we were able to connect to the development board over SSH:

```
$ ssh root@192.168.20.2
```

as well as to mount desired folder:

```
$ sshfs root@192.168.20.2:/home/root fpga/
```

6.1.2 FPGA Board Configuration

Again, we want to configure static IP address. Since Linux on the board doesn't have ip-address tool we can achieve the similar result by adding configuration to */etc/network/interfaces*:

```
auto eth0
iface eth0 inet static
address 192.168.20.2
netmask 255.255.255.0
gateway 192.168.20.1
```

But the Linux on the board is not configured to use any DNS server, so we have to edit that as well. DNS servers can be added to */etc/resolv.conf*:

```
nameserver 8.8.4.4
nameserver 8.8.8.8
```

Restart the FPGA board and then we can install required tools.

```
$ apt-get update
$ apt-get install vim openssh-server
```

After that we could start ssh server and enable it for later usage (with SystemV, systemd not available):

```
$ service ssh start
$ service ssh enable
```

We also may want to change root password:

```
$ passwd root
```

6.1.3 Additional Configuration

```
echo "Europe/Belgrade" > "/etc/timezone"
dpkg-reconfigure -f noninteractive tzdata
ntpdate time.nist.gov
```

6.2 Uploading Raw Binary File

Shell script which uploads Raw Binary File (.rbf) to the FPGA board.

```
#!/bin/sh
echo 0 > /sys/class/fpga-bridge/hps2fpga/enable
echo 0 > /sys/class/fpga-bridge/fpga2hps/enable
echo 0 > /sys/class/fpga-bridge/lwhps2fpga/enable
dd if=/home/root/DE1_SoC_Computer.rbf of=/dev/fpga0 bs=1M
echo 1 > /sys/class/fpga-bridge/hps2fpga/enable
echo 1 > /sys/class/fpga-bridge/fpga2hps/enable
echo 1 > /sys/class/fpga-bridge/lwhps2fpga/enable
```

6.3 Source Code

`https://gitlab.com/lukicdarkoo/image-accelerator`