SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

LRI1 – Multimedijske arhitekture i sustavi

Lucija Abramac

Ana Ćubelić

Pejo Gusak

Marko Jurin

Ivan Trubić

Jurica Vidak

Zagreb, siječanj 2020.

Sadržaj

1. Uvod

U sklopu kolegija „Multimedijske arhitekture i sustavi“ i „Laboratorij računalnog inženjerstva“ potrebno je ostvariti sustav za snimanje, obradu, prijenos i prikaz slika koristeći PYNQ-Z1 razvojnu pločicu i OV7670 kamera modul.

Sklopovlje sustava kreirano je pomoću alata „Xilinx Vivado“ za izradu sklopovlja, a glavna aplikacija napravljena je pomoću alata za razvoj programske podrške „Xilinx SDK“.

2. Opis sustava

2.1 Oprema za rad

Sklopovlje se sastoji od 2 glavna dijela, a to su PYNQ-Z1 razvojna pločica i OV7670 kamera modul.

Razvojna pločica PYNQ-Z1 zasnovana je na Xilinx Zynq porodici programirljivih sklopova koja programerima omogućava korištenje svih mogućnosti programabilnih SoC-ova bez potrebe za dizajniranjem programabilnih logičkih sklopova. Oni se uvoze kao biblioteke sklopovlja i programiraju kroz API-je.

Kamera modul OV7670 niskonaponski je CMOS video senzor koji pruža funkcionalnost VGA kamera i procesora slika. Omogućava 8-bitni potpuni okvir, uzorak ili prozor slike u širokom rasponu formata i razne obrade slike (npr. ekspozicija, balans bijele boje), a kontrolira se preko serijske upravljačke sabirnice (SCCB). Modul radi do 30 slika u sekundi (*fps*) u VGA formatu (640x480 piksela) te korisnik određuje kvalitetu slike, oblikovanje i daljnji prijenos izlaznih podataka.

2.2 Algoritam

Algoritam sustava sastoji se od sedam koraka:

1) PYNQ-Z1 preko I2C sučelja inicijalizira OV7670 kamera modul, lwIP server te GPIO

2) na računalu se šalje paket preko TCP-a na IP adresu 192.168.1.10:7

(lwIP server na PYNQ-Z1)

3) PYNQ-Z1 prima paket te pokreće dohvaćanje okvira s OV7670 kamera modula preko GPIO ulaza

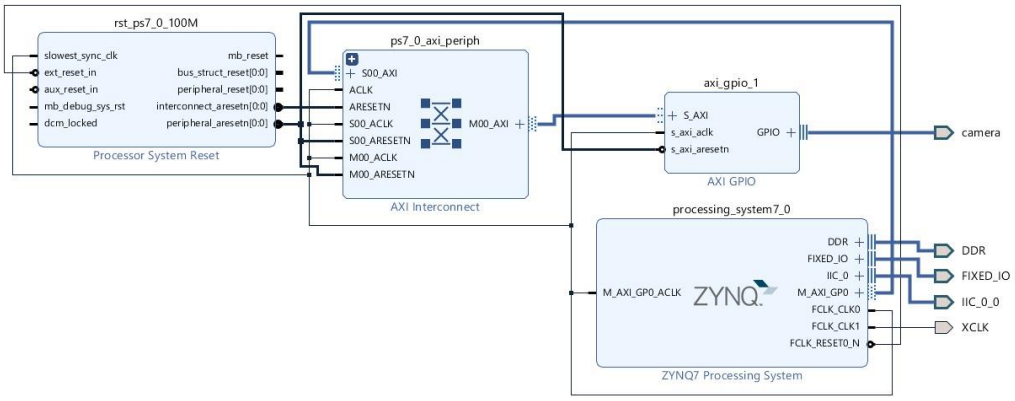
5) slika se sprema na SD karticu

6) nakon kompresije, JPEG enkodiranja i kriptiranja podaci se šalju TCP protokolom nazad klijentu (računalu)

7) računalo prikazuje dobivenu sliku

2.3 Dizajn

Koristeći alat Vivado 2019.1 i ugrađeni IP Integrator dizajnirano je ponašanje PYNQ-Z1 razvojne pločice, odnosno dizajn bloka (eng. *block design*).



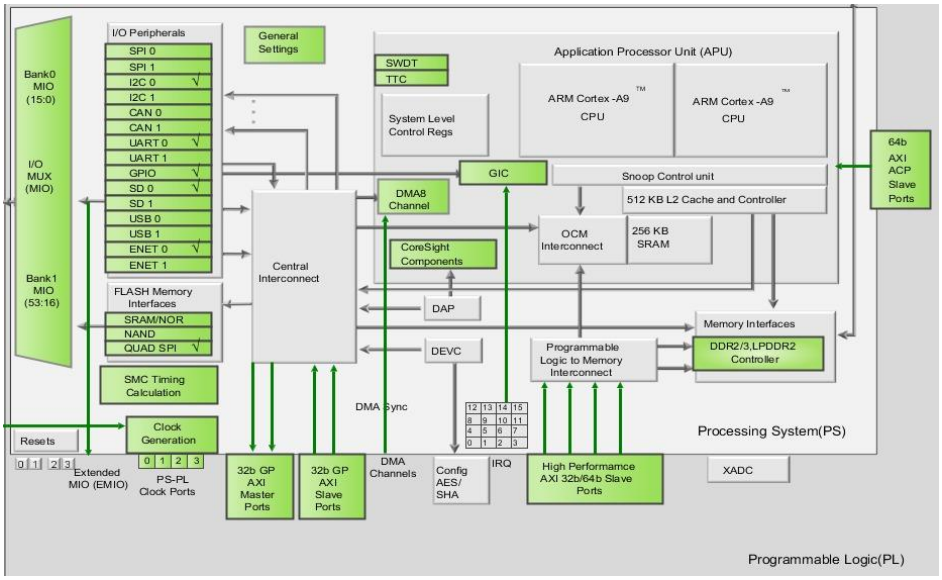
Slika 1. Dizajn bloka

Dizajn bloka sastoji se od više IP-a:

- ZYNQ procesor
- Processor System Reset
- AXI Interconnect
- AXI GPIO

Prilikom konfiguriranja ZYNQ IP-a određena su i dva *clocka*: jedan od 100 MHz za pokretanje internih sklopova, a drugi od 12 MHz koji je spojen na XCLK priključak OV7670 kamera modula.

Također su mu određena svojstva I2C 0, UART 0, GPIO, SD 0, ENET 0.



Slika 2. Konfiguracija ZYNQ processing system IP-a

Nakon dizajniranja i konfiguriranja, koristeći Vivado alat, automatski je generiran HDL omotač u VHDL jeziku, a zatim je pokrenuta simulacija ponašajnog dijela RTL-a i dizajn je sintetiziran.

Nakon sinteze dizajna, u elaboriranom dizajnu (*RTL analysis*) konfigurirani su ulazi i izlazi dizajna, odnosno postavljena su ograničenja. Datoteka *„constraints.xml“* korištena u ovom koraku, nalazi se na kraju dokumenta kao prilog.

Nakon postavljanja ograničenja, dizajn je pripremljen za implementaciju i generiranje *bitstreama*.

2.4 Glavni program

Prije početka implementiranja glavnog programa, za što se koristi Xilinx SDK, dizajn sklopovlja i *bitstream* izvode se u odgovarajući radni prostor.

U sklopu ovog rada u Xilinx SDK napravljena su dva projekta.

Program prvog projekta uspješno inicijalizira lwIP server i ostvaruje početnu komunikaciju s klijentskom stranom, odnosno računalom.

Program drugog projekta preko I2C protokola inicijalizira kamera modul i odgovarajuće GPIO priključke za primanje piksela slike. Nakon što program primi piksele ispisuje ih na serijsku vezu, što je pokazatelj da je uspješno dohvatio nešto od kamera modula.

Sljedeći korak bi bio spremanje tih podataka na SD karticu, no pri pokušaju pisanja, sam Xilinx SDK stvarao je poteškoće. Naime paket *"xilffs_v4_1"* stvarao je probleme prilikom dodavanja u projekt te je njegova prisutnost izazivala nepopravljive greške koje su ometale ispravan rad sustava.

2.5 Desktop aplikacija

Klijentska (desktop) strana projekta napisana je u Python programskom jeziku. Ona ostvaruje TCP vezu sa razvojnom pločicom na portu 7 te čeka od korisnika zahtjev za slikom.

Slika 3. Pokretanje desktop aplikacije

Slika 4. Čekanje na zahtjev od korisnika

Pritiskom na tipku „Enter“ šalje se zahtjev te dohvaća trenutno sliku. Kad se slika u cijelosti prihvati program je sprema u direktorij sa jedinstvenim imenom. Nakon što je slika pohranjena, otvara se u standardnom pregledniku te se ponovno čeka na zahtjev korisnika.

Slika 5. Dohvaćanje, prikaz slike i čekanje sljedećeg zahtjeva

3. Zaključak

Sustav koji se sastoji od jednostavnih komponenti, naizgled je bio lako izvediv, međutim alati za rad jako su nespretni što je ometalo, a na kraju i onemogućilo razvoj. Krajnje točke projekta, inicijalizacija sklopovlja i snimanje slike, kao i dohvaćanje slike preko TCP veze i prikaz iste, uspješno su ostvareni. Ono što nedostaje jest spremanje dohvaćene slike na SD karticu i slanje TCP vezom.

4. Prilog

constraints.xml:

```
set_property PACKAGE_PIN Y17 [get_ports {camera_tri_i[0]}]
set_property PACKAGE_PIN W14 [get_ports {camera_tri_i[1]}]
set_property PACKAGE_PIN W19 [get_ports {camera_tri_i[2]}]
set_property PACKAGE_PIN V16 [get_ports {camera_tri_i[3]}]
set_property PACKAGE_PIN Y16 [get_ports {camera_tri_i[4]}]
set_property PACKAGE_PIN Y14 [get_ports {camera_tri_i[5]}]
set_property PACKAGE_PIN W18 [get_ports {camera_tri_i[6]}]
set_property PACKAGE_PIN W16 [get_ports {camera_tri_i[7]}]
set_property PACKAGE_PIN T11 [get_ports {camera_tri_i[8]}]
set_property PACKAGE_PIN U19 [get_ports {camera_tri_i[9]}]
set_property PACKAGE_PIN V12 [get_ports {camera_tri_i[10]}]
set_property IOSTANDARD LVCMOS33 [get_ports {camera_tri_i[10]}]
set_property IOSTANDARD LVCMOS33 [get_ports {camera_tri_i[9]}]
set_property IOSTANDARD LVCMOS33 [get_ports {camera_tri_i[8]}]
set_property IOSTANDARD LVCMOS33 [get_ports {camera_tri_i[7]}]
set_property IOSTANDARD LVCMOS33 [get_ports {camera_tri_i[6]}]
set_property IOSTANDARD LVCMOS33 [get_ports {camera_tri_i[5]}]
set_property IOSTANDARD LVCMOS33 [get_ports {camera_tri_i[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {camera_tri_i[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {camera_tri_i[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {camera_tri_i[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {camera_tri_i[0]}]

set_property PACKAGE_PIN T10 [get_ports IIC_0_0_scl_io]
set_property PACKAGE_PIN Y18 [get_ports IIC_0_0_sda_io]
set_property IOSTANDARD LVCMOS33 [get_ports IIC_0_0_scl_io]
set_property IOSTANDARD LVCMOS33 [get_ports IIC_0_0_sda_io]
set_property PULLUP true [get_ports IIC_0_0_scl_io]
set_property PULLUP true [get_ports IIC_0_0_sda_io]
set_property PACKAGE_PIN Y19 [get_ports XCLK]
set_property IOSTANDARD LVCMOS33 [get_ports XCLK]
set_property SLEW FAST [get_ports XCLK]
set_property OFFCHIP_TERM NONE [get_ports XCLK]
```

5. Literatura

[1] prezentacije iz kolegija „Multimedijske arhitekture i sustavi“

[2] materijali dostupni na stranicama kolegija (Xilinx tutorial, OV7670 opis)

[3] https://embeddedprogrammer.blogspot.com/2012/07/hacking-ov7670-camera-module-sccb-cheat.html?fbclid=IwAR1nzCTVyjCWoGq3Qd2w0KXB8VInYEtpBU6WrE6m1Vs5pA__u1Mog1YAGmA

[4] tutoriali dostupni u Vivado alatu