TECH+FPGA Guide

- ## Vivado setup

Prvi korak jeste skidanje zip fajla sa git repozitorijuma, adresa je https://github.com/filip-stefanovic-bitsolver/hakaton_2023_eee.

Fajlovi za ucitavanje projekta u Vivadu se nalaze u hakaton_vivado/ direktorijumu. Potrebno je izvrsiti sledece korake kako biste ucitali projekat:

1. Otvoriti Vivado 2021.1
2. Odabrati putanju Tools-> Run Tcl Script
3. Navigirati do hakaton_vivado direktorijuma
4. Izabrati hakaton.tcl I pokrenuti izvrsavanje.

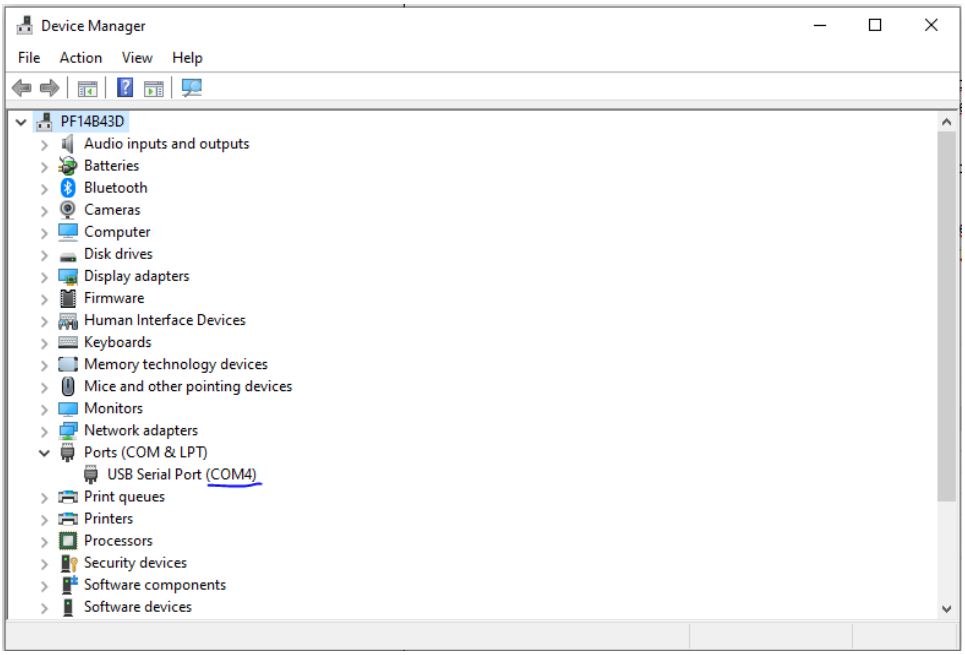
Projekat ce biti kreiran unutar direktorijuma u kome se nalazi hakaton.tcl skripta. Projekat ce biti imenovan hakaton_eee i sadrzi 2 DMA kontrolera, example_design i 4 Logic Analyzer-a. Prvi DMA koristi 8bitnu AXI Stream magistralu za slanje slike ka dizajnu i prijem obradjene slike sa dizajna. Drugi DMA samo vrsi prijem obradjene slike, ali sa 16bitne AXI Stream magistrale. Modul example_design je primer dizajna koji prihvata piksele sa AXI Stream magistrale i zatim ih shift-uje nalevo ili nadesno. Takodje ima integrisan AXI-APB bridge kojim se moze pristupati memorijski-mapiranim registrima (ukoliko ima potrebe za time). Logic Analyzer-i se mogu koristiti za posmatranje AXI magistrala koje se koriste u komunikaciji sa dizajnom.

Vasu implementaciju zadatka treba da integrisete u ovaj projekat.

- ## Jupiter notebook setup

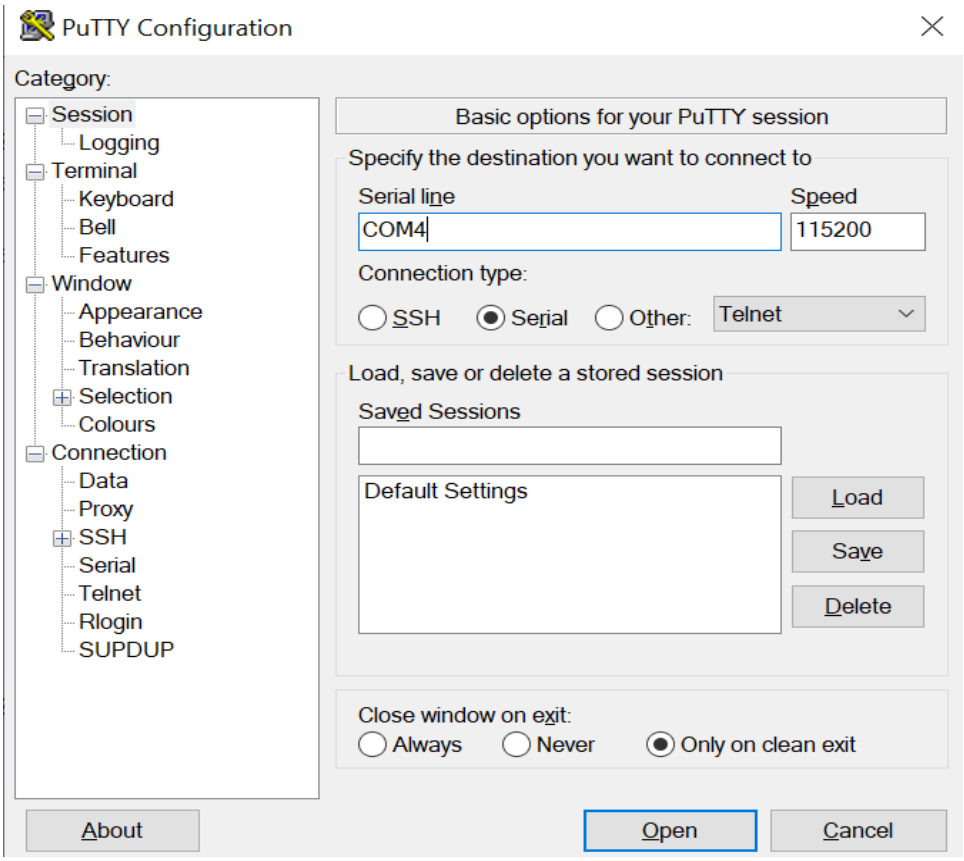
Dizajn je moguce koristiti pomocu python interaktivne konzole - jupyter notebook-a. Nakon boot-ovanja OS-a ce jupyter notebook biti dostupan na IP adresi, kojoj treba pristupiti preko vaseg browsera. Da bi ste znali koju IP adresu koristite za pristup treba da ispratite sledece korake.

1. Kada programirate plocicu, treba da pogledate na koji port je plocica povezana. Ovo mozete da vidite u Device Manager-u -> Ports (COMX).



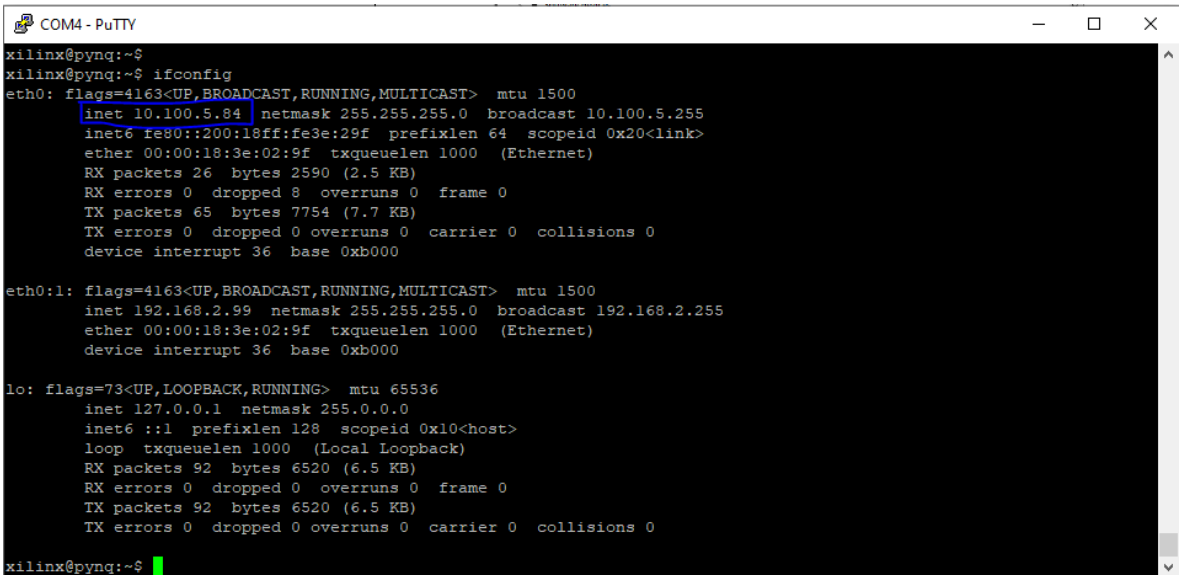
Slika 1. Provera na koji port je plocica povezana

2. Nakon toga treba da pokrenete Putty aplikaciju I da je podesite kao na slici.



Slika 2. Konfiguracija Putty za serijski pristup

3. Kada Vam se otvori terminal treba da ukucate komandu *ifconfig* I IP adresu koja vam je prikazana kucate u browser. Bitno je da posle IP adrese navedete I port 9090, koji je svima isti (Primer. 10.100.2.200:9090).

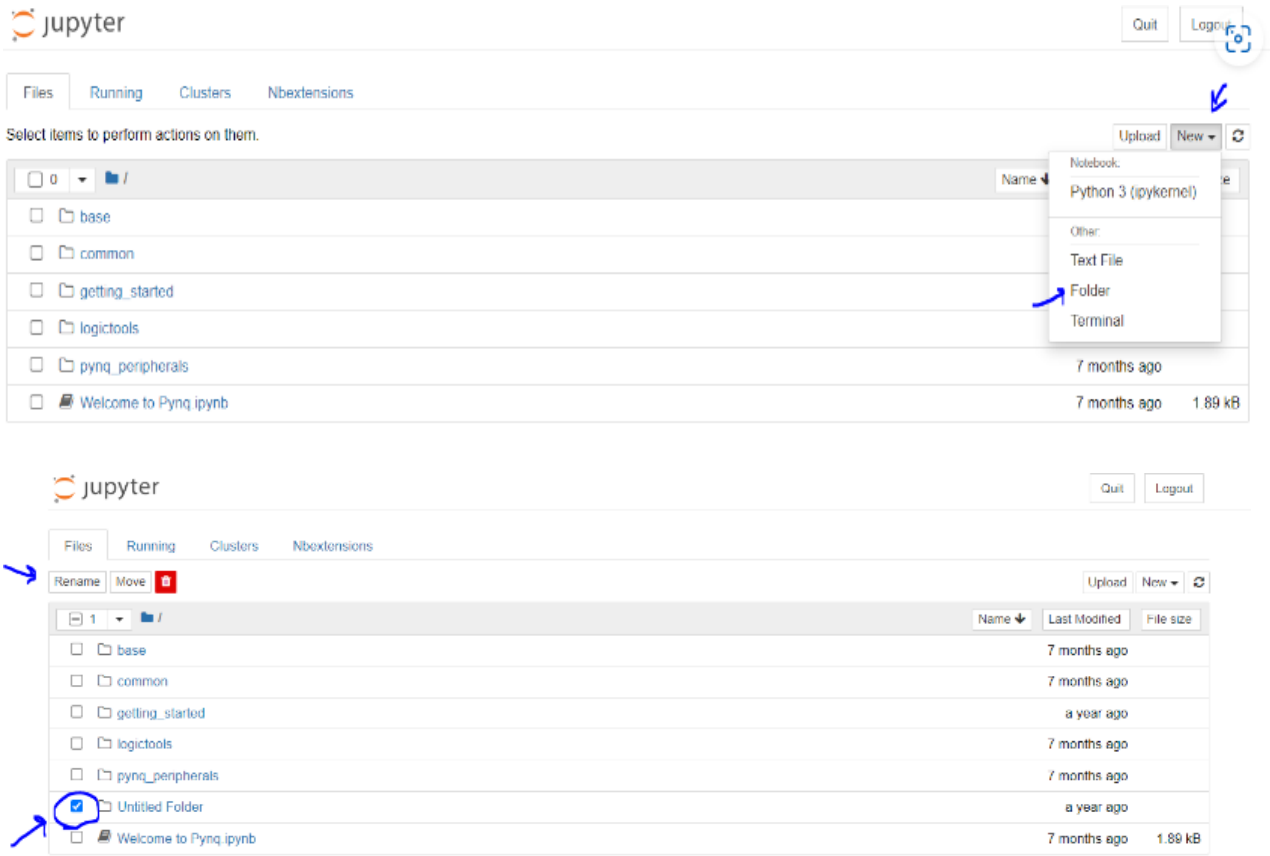


Slika 3. IP adresa pristupa browser-u

Fajlovi za testiranje projekta se nalaze u direktorijumu jupiter_demo. Lozinka za pristup plocici je xilinx. Nakon sto se ulogujete cete videti ovakvu strukturu direktorijuma:

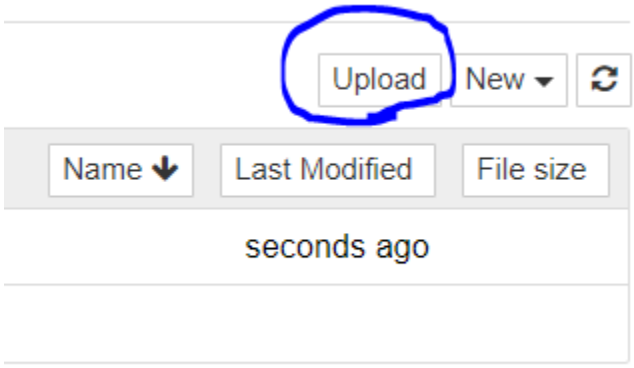
Slika 4. Prvobitni izgled direktorijuma

Potrebno je kreirati direktorijum /hakaton.



Slika5. I 6. Kreiranje direktorijuma I promena imena

Unutar njega upload-ovati sve fajlove iz jupiter_demo direktorijuma.



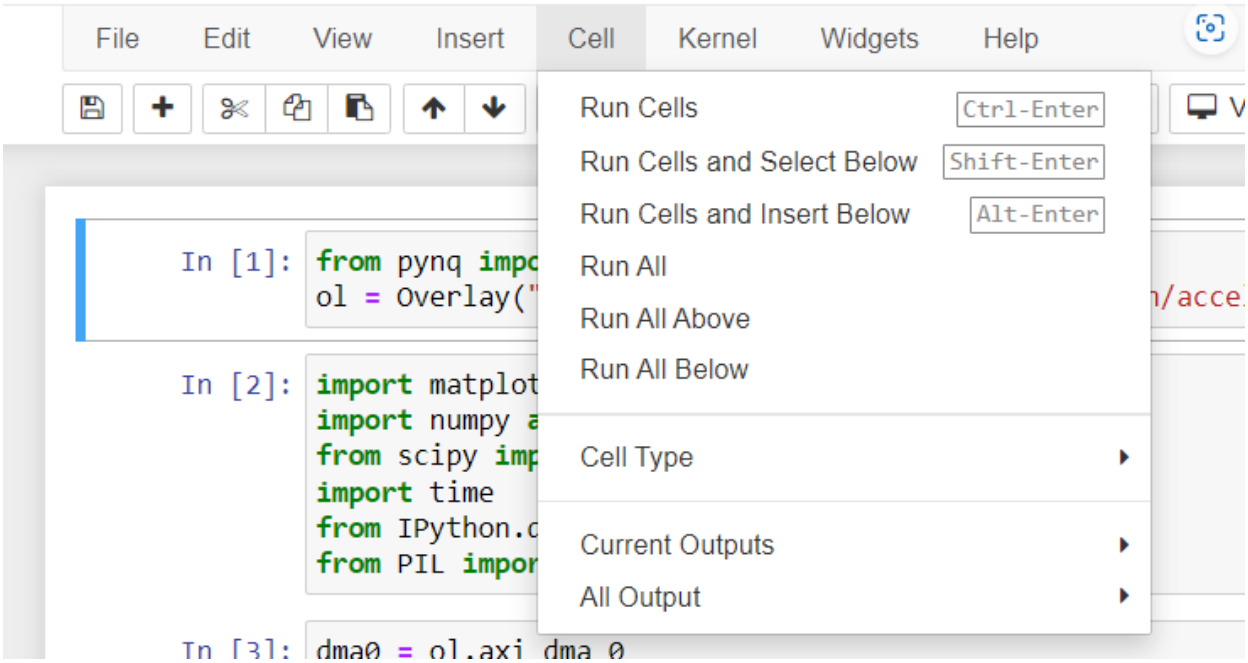
Slika 7. Upload file

Direktorijum /hakaton treba da izgleda ovako:

Slika 8. Izgled direktorijuma hakaton

Nakon toga otvorite hakaton_demo.ipynb, gde se nalazi primer python koda kojim se ucitava slika, salje koriscenjem DMA kontrolera (dma0.sendchannel.transfer(input_buffer)), i zatim se obradjena slika se prima koriscenjem 2 DMA kontrolera (dma0.recvchannel.transfer(output_buffer1) dma1.recvchannel.transfer(output_buffer2)). Takodje ima primere upisa u memorijski-mapirane registre, kao i koriscenje tajmera za merenje performansi.

Mozete izvrsiti sve instrukcije pritiskom na Cell:



Slika 9. Pokretanje Celija u notebook-u

Mozete menjati demo notebook po svojoj potrebi. U sledecem poglavlju je ukratko opisana HW-SW veza.

- ## Ucitavanje novog dizajna na plocicu

Nakon svake izmene u dizajnu je neophodno zameniti accelerator_top.bit i accelerator_top.hwh fajlove, koje ste upload-ovali prilikom setup-a Jupiter notebook-a. Tako ce najnovija verzija dizajna biti koriscena prilikom pokretanja hakaton_demo.ipynb notebook-a.

Potrebno je izvrsiti sintezu, implementaciju i ispis bitstream-a.

Ekstenzija .bit oznacava bitstream fajl - on se automatski generise nakon gore navedenih koraka, na putanji hakaton_eee\hakaton_eee.runs\impl_1/accelerator_top_wrapper.bit. Ekstenzija .hwh oznacava hardware handoff fajl, neophodan je da bi SW znao kako da komunicira sa dizajnom, a generise se u hakaton_eee\hakaton_eee.gen\sources_1\bd\accelerator_top\hw_handoff/accelerator_top.hwh, takodje prilikom izvrsavanja koraka opisanih iznad.

Ovi fajlovi se ucitavaju u prvom koraku hakaton_demo.ipynb. Neophodno je da imaju isto ime, tako da ih treba preimenovati pre pokretanja notebook-a. Ime koje notebook koristi je definisano ovom linijom:

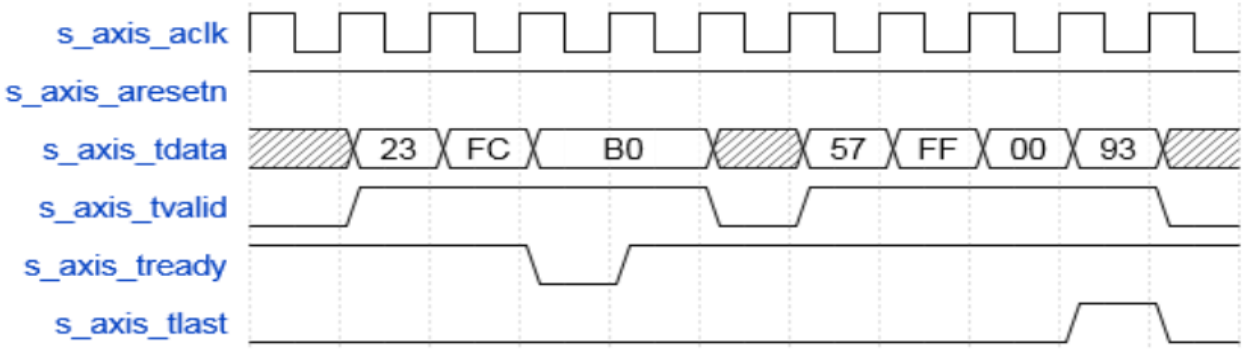
ol = Overlay("/home/xilinx/jupyter_notebooks/hakaton/accelerator_top.bit"), a na osnovu toga se ocekuje i accelerator_top.hwh. Imena mogu biti proizvoljna.

- AXI Stream (AXIS) protokol

Na slici je prikaz svih signala koji cine AXI-Stream interfejs, u slučaju AXI master komponente zajedno sa kratkim objašnjenjem svakog od signala. U slučaju AXI slejv komponente AXI-Stream interfejs bi se sastojao iz istih signala, samo bi smer svakog od njih bio obrnut.

Naziv signala	Opis signala
m_axis_aclk	Sinhronizacioni signal AXI interfejsa.
m_axis_aresetn	Reset signal AXI interfejsa.
m_axis_tdata	Magistrala podataka.
m_axis_tvalid	Indikacija da je tekući podatak validan.
m_axis_tready	Indikacija da je AXI slejv spreman da primi sledeći podatak.
m_axis_tlast	Indikacija da je tekući podatak ujedno i poslednji podatak koji je potrebno preneti.

Na narednoj slici su prikazani talasni oblici signala AXI-Stream interfejsa AXI-Stream mastera prilikom transakcije prenosa bloka podataka. Proces prenosa započinje kada AXI-Stream master postavi prvi podatak na m_axis_tdata magistralu i podigne m_axis_tvalid signal na jedinicu. Postavljeni podatak ostaje na m_axis_tdata magistrali sve dok AXI-Stream slejv modul ne signalizira AXI-Stream masteru da je spreman za prihvat sledećeg podatka, postavljajući m_axis_tready signal na jedinicu, , što se lepo može videti na slici. Kada AXI-Stream master uoči da je AXI-Stream slejv spreman za prijem sledećeg podatka, on u sledećem taktu na m_axis_tdata magistralu postavlja sledeći podatak. Ukoliko iz nekog razloga AXI-Stream master nije u stanju da postavi sledeći podatak na m_axis_tdata magistralu, on ovu činjenicu signalizira AXI-Stream slejvu spuštanjem m_axis_tvalid podatka. Opisani postupak prenosa podataka nastavlja se dok se ne dodje do poslednjeg podatka koji je potrebno preneti. Kada AXI-Stream master postavi poslednji podatak na m_axis_tdata magistralu i označi ga kao validan, postavljajući m_axis_tvalid signal na jedinicu, istovremeno postavlja i m_axis_tlast signal na jedinicu, signalizirajući da je reč o poslednjem podatku iz tekućeg paketa. Nakon što AXI-Stream slejv preuzme i ovaj podatak, postavljajući m_axis_tready signal na jedinicu, tekuća transakcija prenosa bloka podataka je završena.



Slika 10. Talasni oblici signala AXI Stream protokola

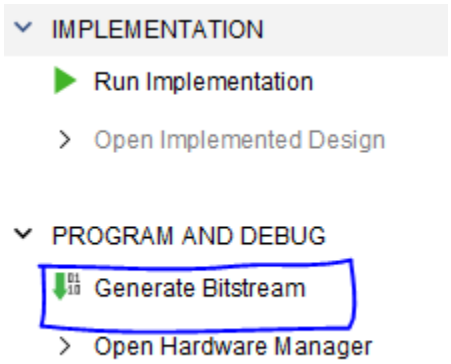
- ## Uputstvo kako koristiti Logic Analyzer-e

Pored mogucnosti simulacionog testiranja funkcionalnosti vaseg sistema (eng. testbench), postoji mogucnost testiranja u realnom vremenu pomocu Logic Analyzer-a. U projektu imate 4 Logic Analyzer-a koji su objasnjeni u narednoj tabeli.

Broj Logic Analyzer-a	Naziv	Sta proverava
1.	ILA_0	Proverava slanje podataka od prvog DMA do vaseg dizajna
2.	ILA_1	Proverava slanje podataka od vaseg dizajna do prvog DMA (8bit podaci)
3.	ILA_2	Proverava slanje podataka od vaseg dizajna do drugog DMA (16bit podaci)
4.	ILA_3	Proverava slanje podataka na AXI-APB bridge (ukoliko ima potrebe za tim)

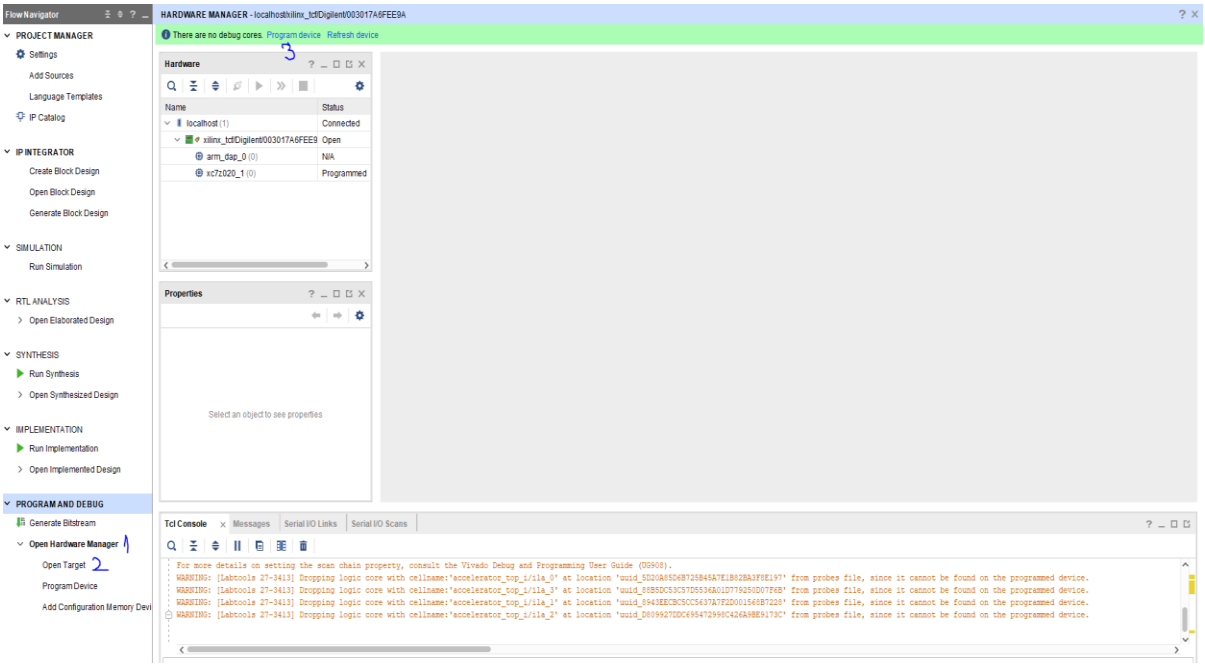
Koriscenje Logic Analyzer-a se radi u trenutku kada imate izgenerisan bitstream I programiranu plocicu.

Bitstream generisete pokretanjem opcije Generate Bitstream.



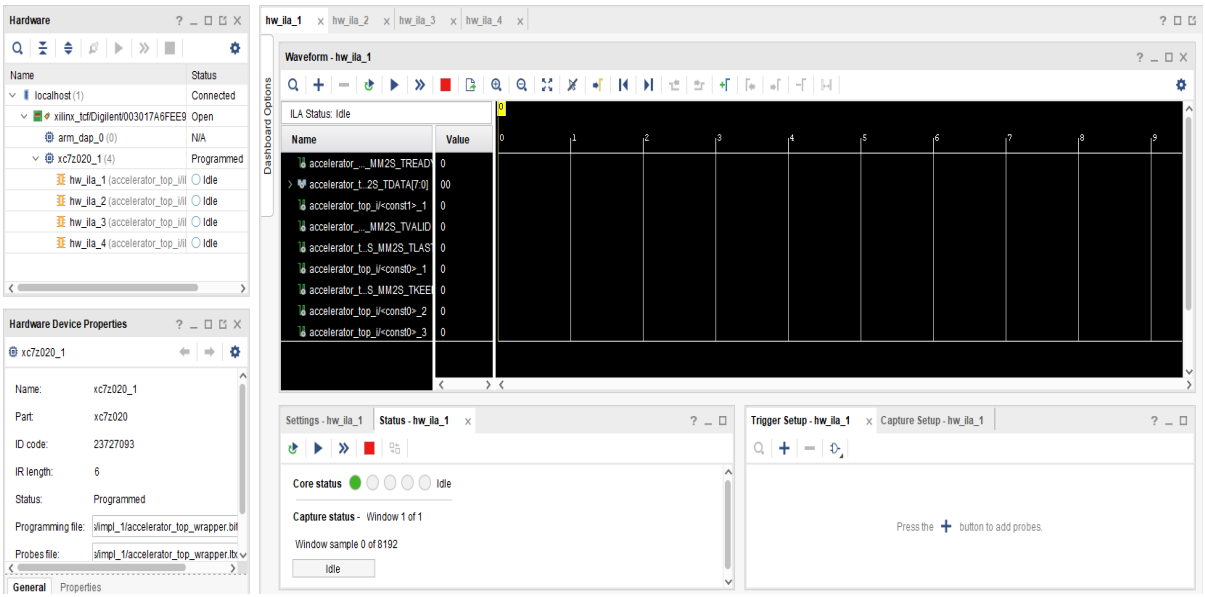
Slika 11. Generisanje Bitstream-a

Nakon sto se uspesno generise bitstream sledeci korak jeste pokretanje Open Hardware Manager -> Open target -> Auto Connect -> Program device. Ovim ce se vasa implementacija programirati (“spustiti”) na plocicu.



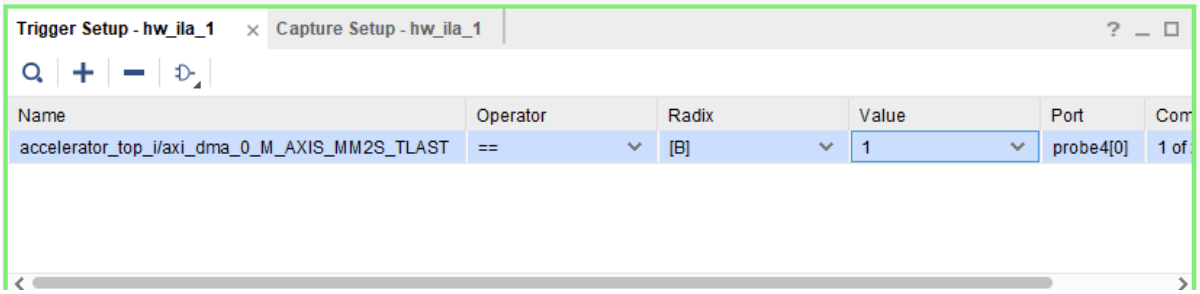
Slika 12. Programiranje plocice

Nakon sto programirate plocicu, treba da vam se pojavi displej za svaki od 4 Logic Analyzer-a.



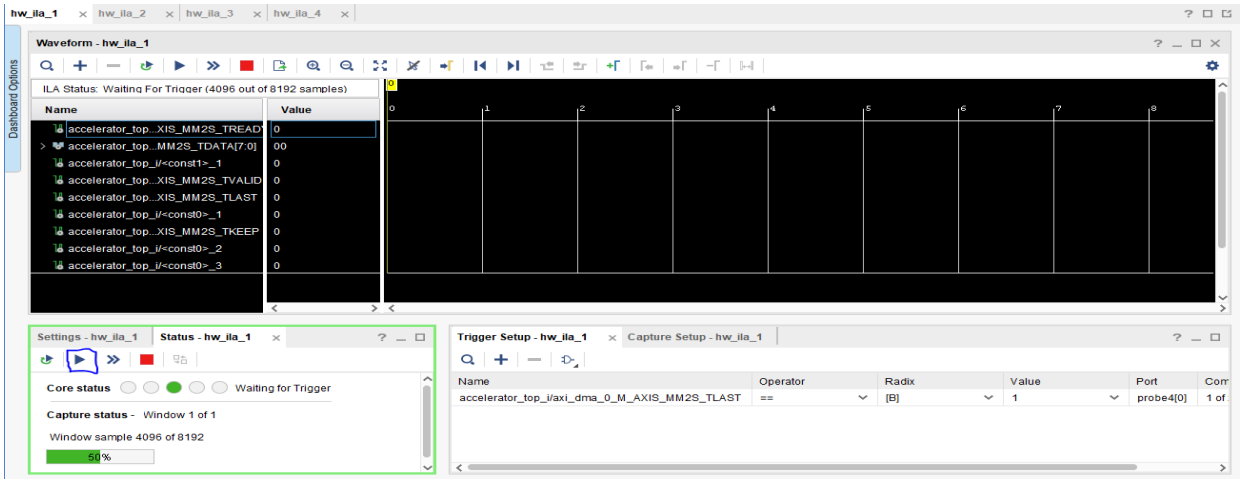
Slika 13. Programirana plocica

Naredni korak jeste da u kartici Trigger Setup dodate uslove koje hocete da pratite. Primer hocete da vidite kada se signal *TLAST* promeno na ‘1’. Kako se to setuje mozete da vidite na narednoj slici.



Slika 14. Triger setup

Da vidite kada se ovaj signal trigerovao, morate da pokrenete dve stvari u datom redosledu. Prvo pokrenete rad ILA-e u Vivadu, a nakon toga pokrecete u Jupiter nootbook-u celiju (eng. Cell) za transakciju podataka izmedju DMA I vaseg dizajna. Ukoliko je sve ispravno treba da se vratite na Vivado I pogledate da li je Logic Analyzer uhatio ovu promenu. Ova tri koraka su prikazana u naredne tri slike.



Slika 15. Pokretanje Logic Analyzer-a

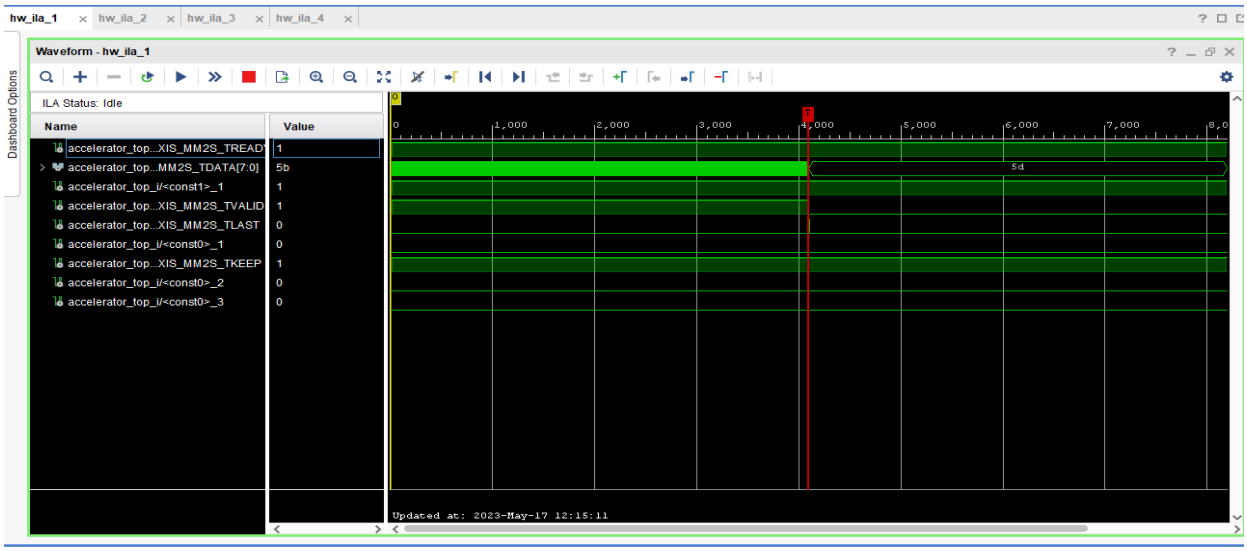
```
In [ ]: t1 = time.perf_counter()

        dma0.sendchannel.transfer(input_buffer)
        dma0.recvchannel.transfer(output_buffer1)
        dma1.recvchannel.transfer(output_buffer2)

        dma0.sendchannel.wait()
        dma0.recvchannel.wait()
        dma1.recvchannel.wait()

        t2 = time.perf_counter()
```

Slika 16. Pokretanje Celije za transfer podataka



Slika 17. Prikaz kako je Logic Analizer uhvatio situaciju kada se signal *TLAST* trigerovao