

Metod export-ovanja simuliranog signala i njegove vizualizacije

9. Januar, 2023

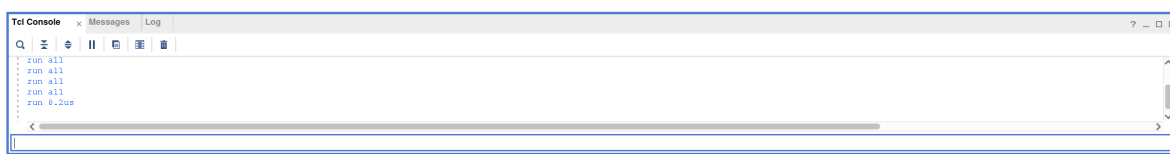
1 Motivacija

Ovaj postupak značajno ubrzava debug-ovanje projekta ovog predmeta jer zaobilazi potrebu za implementaciju, sintezu i uploadovanje na fpga. Poglavlje o exportovanju važi generalno za sve signale i primene, dok je python skipta pisana specifično za obradu izlaza median filtera.

2 Exportovanje simuliranog signala (Vivado)

Vivado pruža mogućnost simuliranja integrisanog kola, što daje uvid u vremenske oblike signala unutar istog, odlično kada posmatramo izlaz sabirača ili bilo kog kola za koje je korektnost signala lako proveriti prostim gledanjem vremenskog oblika. Ovaj metod postaje nepraktičan kada želimo da verifikujemo signal koji menja stanje par stotina puta i ručno poredjenje sa očekivanim rezultatom postane mukotrpno. Za ove slučajeve unutar Vivado-a implementirane su funkcije koje vremenski oblik signala zapisuju u fajl koji korisnik dalje obrađuje po volji.

Ovo se može positići pokretanjem sledećih komandi u tcl konzoli:



Unutar tcl konzole pokrenuti `open_vcd` komandu uz prateći argument koji odgovara imenu fajla u koji upisujemo. Primer komande bi bio:

```
open_vcd xsim_dump.vcd
```

Gde je `xsim_dump.vcd` ime fajla u koji se signal upisuje, i koji se nalazi u direktorijumu: "direktorijum_projekta/ime_projekta.sim/sim_1/behav/xsim/xsim_dump.vcd"

Zatim je potrebno pokrenuti `log_vcd` komandu uz prateći argument koji odgovara imenu signala. Primer komande:

```
log_vcd /tb/DUT/median
```

Gde je `"/tb/DUT/median"`, ime signala, `"tb"` adresira top modul u simulacionoj hierarhiji, `"DUT"` adresira uredjaj unutar testbench-a, a `"median"` adresira signal unutar tog DUT uredjaja, u ovom slučaju to je izlaz median filtra.

Nakon toga potrebno je pokrenuti simulaciju na neko vreme odredjeno argumentom.

```
run 8us
```

i zatvoriti fajl:

```
close_vcd
```

3 Obrada fajla

Fajl u koji smo upisali signal izgledaće ovako:

```
1 $date
2   Wed Jan  4 19:12:48 2023
3 $end
4 $version
5   2020.2
6 $end
7 $timescale
8   1ps
9 $end
10 $scope module main_tb $end
11 $scope module DUT $end
12 $var wire 8 ! median_out [7:0] $end
13 $upscope $end
14 $upscope $end
15 $enddefinitions $end
16 #0
17 $dumpvars
18 bx !
19 $end
20 #64000
21 b0 !
22 #857832000
23 b10001010 !
24 #858168000
25 b10000110 !
26 #858280000
27 b10000101 !
28 #858392000
29 b1111111 !
30 #858504000
31 b1111001 !
32 #858616000
33 b1110100 !
34 #858728000
35 b1101011 !
36 #858840000
37 b1011100 !
```

Gde su prvih 15 linija posvećeno generalnim informacijama, nakon kojih se naizmenično ispisuje vreme u piko sekundama sa # prefiksom, zatim se u sledećem redu nalazi vrednost na koju se signal postavio u trenutku iz prošle linije, sa prefiksom koji odgovara tipu podatka (u ovom slučaju b jer je binaran broj).

Ovaj fajl je dalje moguće analizirati po volji, ali za svrhe ovog projekta možete koristiti sledeću skriptu. Koristeći skriptu `signal_to_img.py` signal iz fajla se prikazuje u obliku slike, na isti način kao što bi pokretanjem na ploči, ali preskakajući dugotrajan proces sinteze i uploadovanja.

Skripta se pokreće iz komandne linije operativnog sistema:

```
python3 signal_to_img.py xsim_dump.vcd
```

Skripta je modifikovana verzija `main.py` skripte koja je data kao materijal za projekat.

Skripta, nalik originalu, crta 3 slike, prvu na osnovu `xsim_dump.vcd` fajla (koja bi odgovarala izlazu UART-a), drugu koja je original (`lenaCorrupted.bmp`) i treću koja je željeni rezultat. Više o radu skripte piše u njenim komentarima. Pre pokretanja skripte prvih nekoliko irelevantnih linija je potrebno obrisati (u ovom slučaju do 22 linije).