Univerza v Ljubljani
Fakulteta za elektrotehniko

Tadej Živic – 64160353

Poročilo projekta – žični model kocke v 3D prostoru

Digitalna integrirana vezja in sistemi

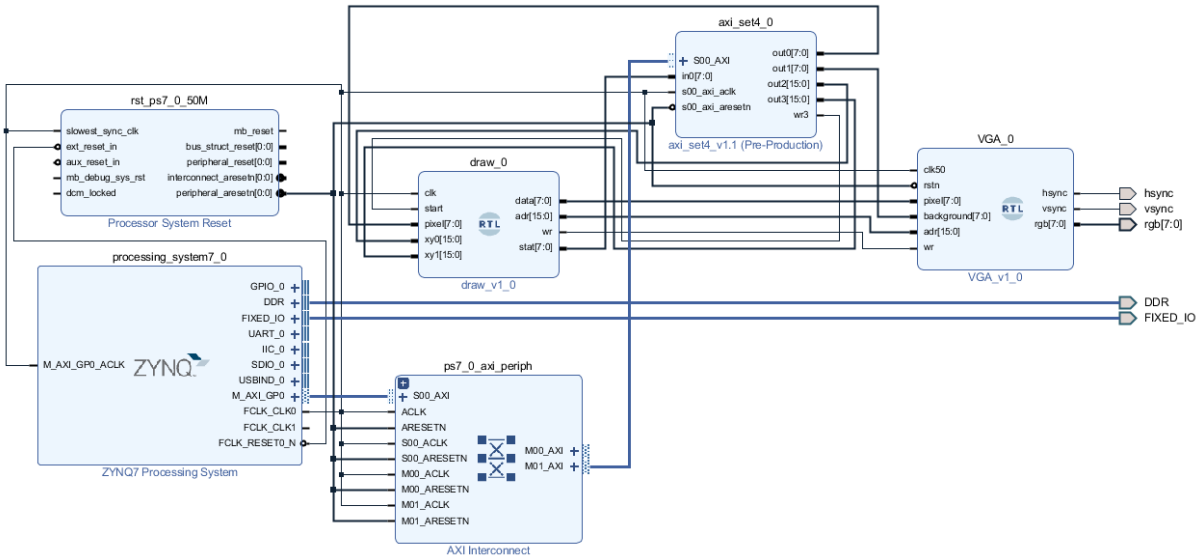
Ljubljana, 2022

1 Naloga

V VHDL-u sem izdelal strojno pospeševanje za izris črte. Izhajal sem iz projekta, ki sem ga delal na vajah, kjer sem izdelal strojno pospeševanje izrisa kvadrata. Pred tem smo risali črte s procesorjem, kjer smo vsak piksel preračunali na procesorju in FPGAju poslali koordinate, kjer naj izriše piksel.

Za pohitritev izvajanja risanja, sem v strojni opremi, v našem primeru FPGA-ju izvedli izris črte tako, da v našem spominu zapolni mesta med dvema koordinatama s pomočjo Bresenhamovega algoritma.

2 Bločni diagram



Slika 1: Blokovna shema sistema(vaja5_block_design.tcl)

3 Cilji

Za cilj sem si zastavil izris žičnega modela kocke, ki se ji prostovoljno nastavlja rotacijo in velikost v 3D prostoru.

4 Delovanje

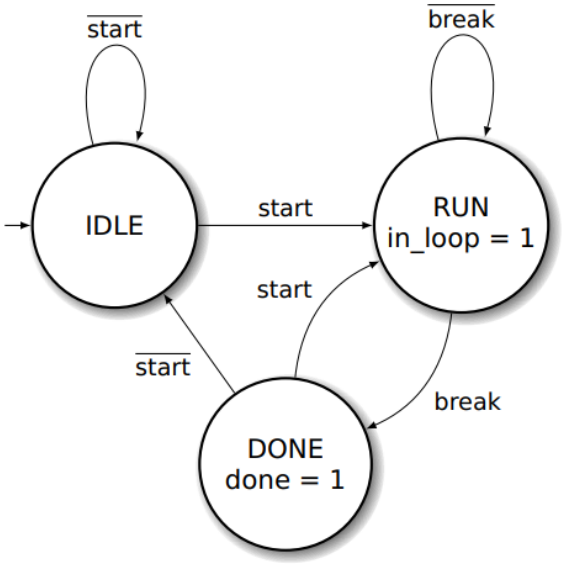
Delovanje programa sem razdelil na dva dela, in sicer na izris črte, ki ga bom izvajal na strojni opremi in izris kocke, ki se bo izvedel na procesorju. Taka delitev nam omogoča hiter izris črt in preprosto programiranje v jeziku C.

4.1 Izris črte

Kot že omenjeno sem izhajal iz primera izrisa kvadrata, kjer sem ga zamenjal z Bresenhamovim algoritmom za risanje črt. Na naslovu XPAR_AXI_SET4_0_S00_AXI_BASEADDR+0 določimo barvo črte, z obliko RRRGGGBB, kjer vsaka črka predstavlja bit barve. Na naslovu XPAR_AXI_SET4_0_S00_AXI_BASEADDR+4 nastavljamo barvo ozadja. Vse to sem zapakiral v funkcije setLineColor() in setBackgroundColor(). Prva koordinata točke daljice, se nahaja v registru na naslovu XPAR_AXI_SET4_0_S00_AXI_BASEADDR + 8, in sicer višji bajt predstavlja x koordinato in nižji bajt y. Isto velja tudi za drugo točko daljice, ki se nahaja v registru na naslovu XPAR_AXI_SET4_0_S00_AXI_BASEADDR + 12.

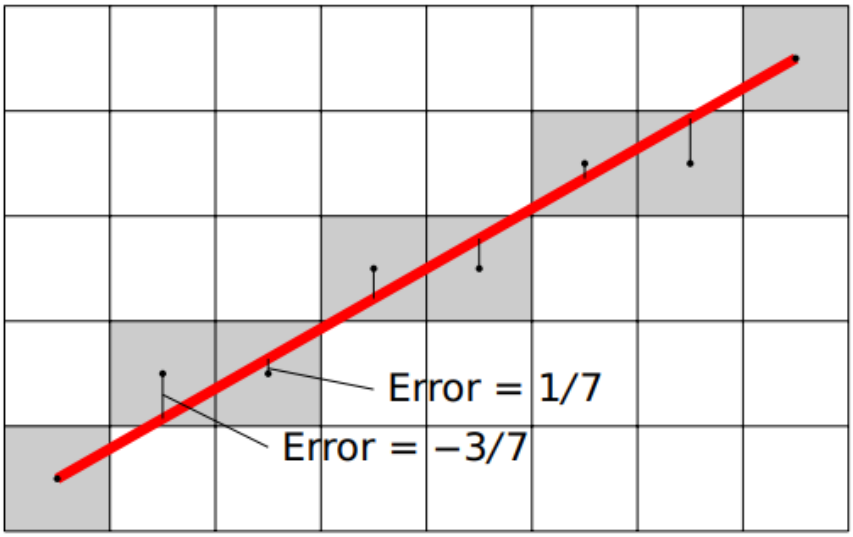
4.1.1 Delovanje algoritma

Algoritem deluje v avtomatu stanj, kjer nastopajo 3 stanja, in sicer IDLE, kjer čaka na ukaz risanja, RUN, kjer algoritem deluje in DONE, kjer nam algoritem naznanja, da je izris črte kočan. Stanja in prehode med njimi nam določa vhodni signal *start*, ki avtomat stanj postavi v stanje RUN oz. interni signal break, ki postavi avtomat stanj v DONE. Diagram stanj in prehode si lahko ogledamo na sliki 2.



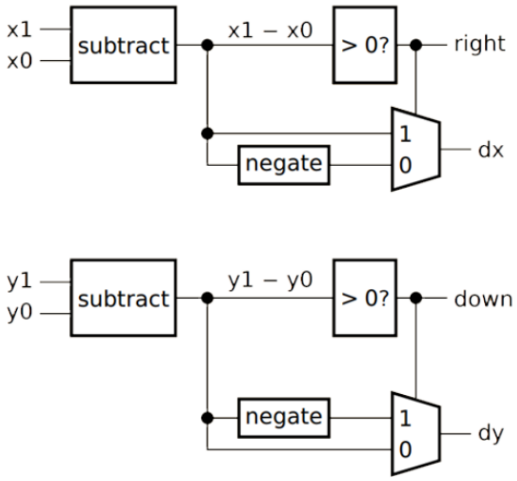
Slika 2: Blokovni diagram avtomata stanj

Algoritem deluje tako, da med dvema točkama aproksimira črto, za vsako x koordinato izračuna y koordinato in izbere piksel, ki je najbližje željeni y koordinati, kot lahko vidimo na sliki 3.



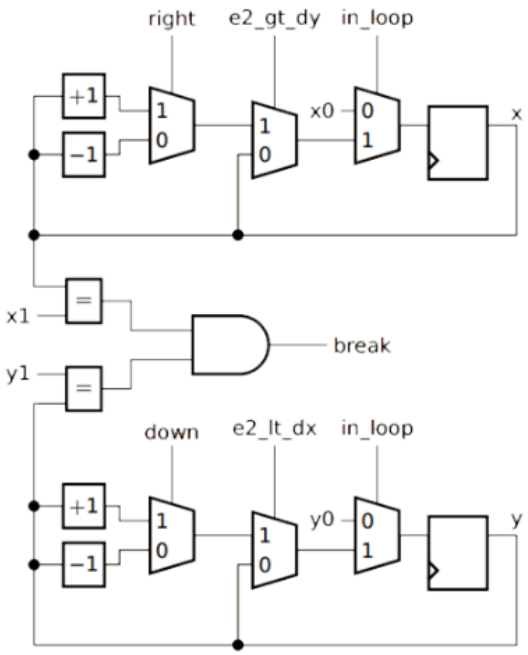
Slika 3: Izris daljice med dvema točkama, kjer algoritem izbere najbližji piksel izračunani koordinati.

Algoritem prvo določi smer risanja črte, se pravi ali črto rišemo gor, dol, levo ali desno, nato pa še razliko med koordinatama daljice, kot vidimo na sliki 4.



Slika 4: Blokovni diagram vezja za izračun smeri risanja in razlike med koordinatama.

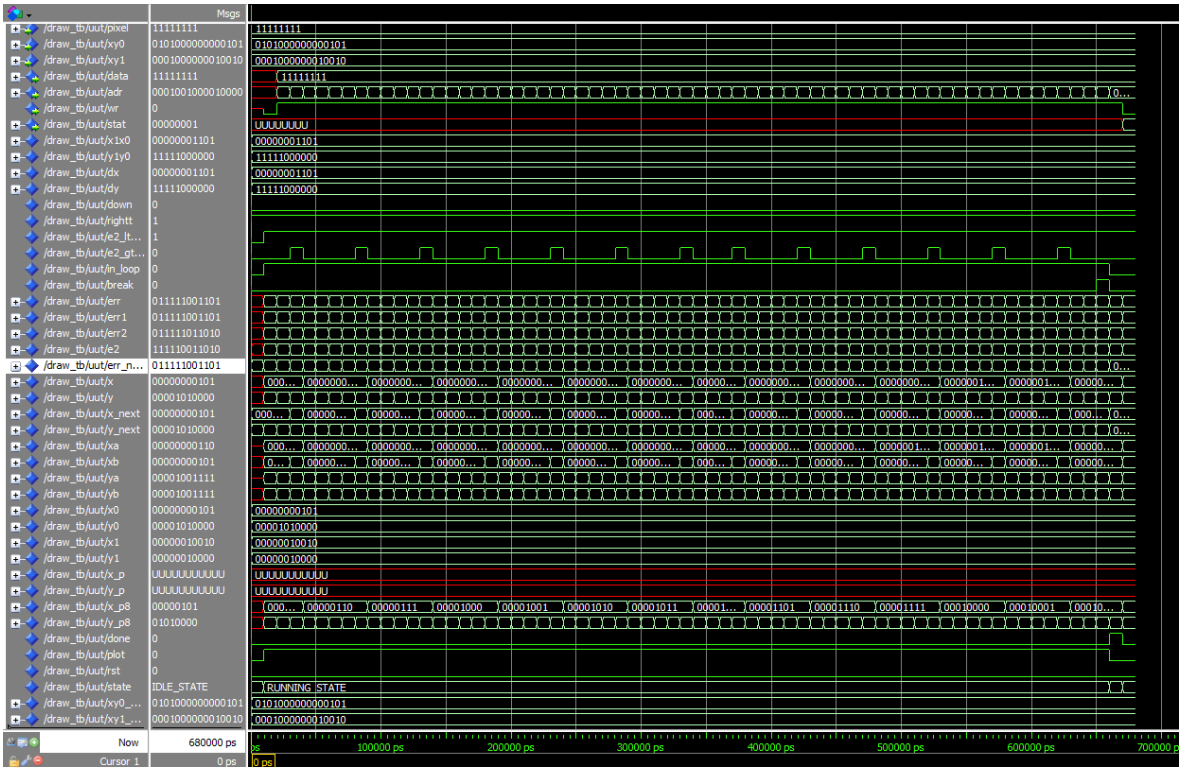
Z izrisom začne v koordinatah prve točke in glede na smer risanja prišteva ali odšteva piksel po piksel obema koordinatama in glede na napako med piksli, določi pravega. To lahko vidimo na shemi na sliki 5.



Slika 5: Blokovni diagram vezja za izračun piksla izrisa, glede na napako. Izhoda vezja sta signala x in y.

4.1.2 Simulacija

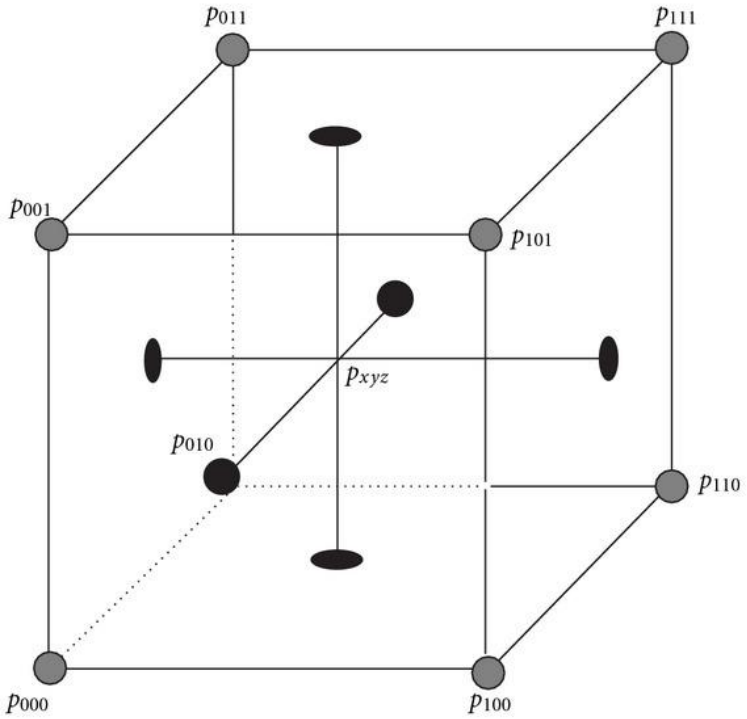
V programu ModelSim sem izvedel simulacijo izrisa. Na simulaciji vidimo vhodne signale *x0*, *y0*, *x1* in *y1*, ki določajo koordinate točk daljice. Prav tako je videti signal *in_loop*, ki naznanjuje da se algoritem izvaja in na koncu se signal *done* postavi na '1'.



Slika 6: Simulacija algoritma

4.2 Kocka

V C programu sem izdelal strukturo kocke, ki zajema 8 točk, med njimi pa izrišem črte. Koordinatni sistem sem postavil v notranjost kocke, predvsem zaradi lažjih translacij in rotacij.



Slika 7: Točke, ki definirajo kocko in koordinatni sistem v kocki.

4.2.1 Definicija kocke

Kocki sem točke postavil tako, da ime točke v binarnem zapisu direktno predstavlja tudi njeno koordinato. Se pravi točka P_{000} predstavlja točko z koordinatami x, y, z = 0, 0, 0 po translaciji in pri ničelni rotaciji itd.

4.2.2 Rotacija

Ker sem si postavil koordinatni sistem v kocko, lahko rotacijo izvajam preprosto tako, da za vsako točko izračunam nove koordinate z uporabo rotacijskih matrik.

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\ \theta_x & -\sin\theta_x \\ 0 & sin\ \theta_x & cos\ \theta_x \end{bmatrix}$$

$$R_y = \begin{bmatrix} cos\ \theta_y & 0 & \sin\ \theta_y \\ 0 & 1 & 0 \\ -\sin\ \theta_y & 0 & cos\ \theta_y \end{bmatrix}$$

$$R_z = \begin{bmatrix} \cos\ \theta_z & -\sin\ \theta_z & 0 \\ \sin\ \theta_z & cos\ \theta_z & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Pri tem moram paziti, da se koordinate točke spremenijo po izvedeni rotaciji in ne vmes, saj lahko to vpliva na uspešnost rotacije. Za izračun kotnih funkcij, sem uporabil vgrajeno knjižnico <math.h>, ki sem jo moral omogočiti v razvojnem okolju.

4.2.3 Izris kocke

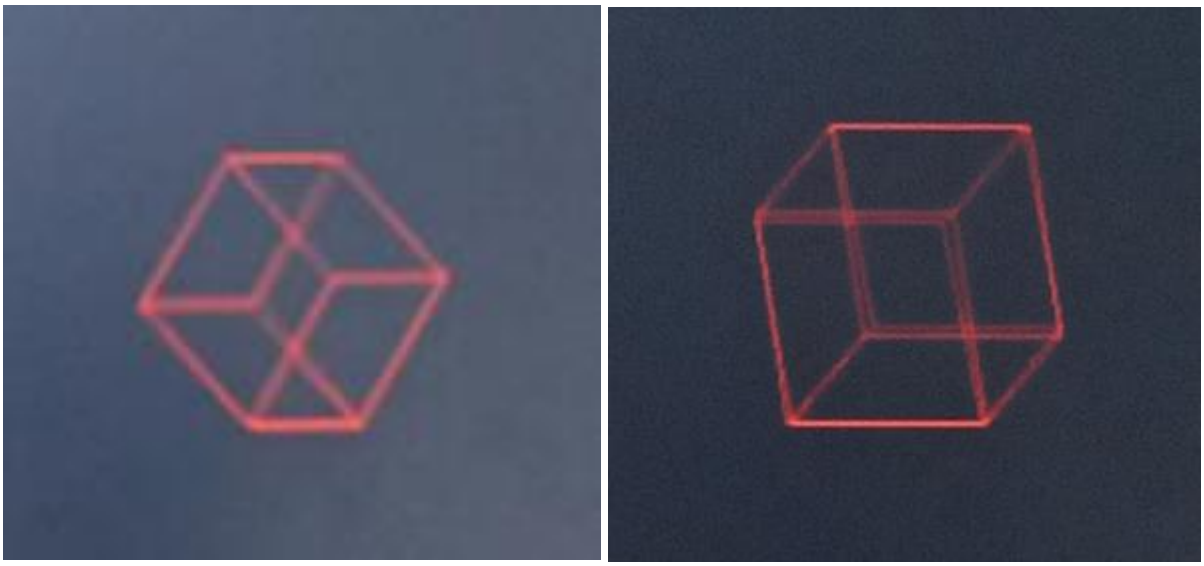
Na začetku programa nastavim barvo črt in ozadja. Pred risanjem vsako točko v kocki transliram za vektor pozicije kocke in poljubno rotiram z uporabo funkcije rotateCube(). Nato pa točke povežem z mojo funkcijo, ki izriše črto med točkama. Ker imamo ekran, ki podpira dvodimenzionalen izris, ne moramo izrisati tridimenzionalne kocke, kar pomeni, da jo moramo pretvoriti v dvodimenzionalen prostor. To sem izvedel z ortogonalno projekcijo, ki v bistvu zanemari *z* koordinato:

$$P(v_x, v_y, v_z) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} = \begin{bmatrix} v_x \\ v_y \\ 0 \end{bmatrix}$$

Pri izrisu črt moram biti pozoren, saj ne smem povezati vseh točk, ampak le tista oglišča, ki so sosednja. Za primer, točka P_{000} se poveže s točkami $P_{001,}$ $P_{010\ in}$ P_{100}, ne sme se pa povezati s točko P_{111}.

5 Preizkus:

Prvo definiram kocko z določeno velikostjo in jo izrišem. Ko izrišem vse črte, nekaj časa počakam, nato pa isto kocko zrišem še enkrat, vendar v tem primeru s črtami črne barve. Efektivno mi to trenutno kocko izbriše in tako pripravi prazno platno za izris nove. Preden izrišem novo, pa ji rahlo spremenim velikost in rotacijo za vse smeri. Ta postopek ponavljam v neskončnost. Izriše pa se rotirajoča se kocka, ki spominja na nekakšen ohranjevalnik zaslona, kot lahko vidimo na sliki 8.



Slika 8: Izrisana kocka, pri drugi sliki lahko vidimo dvojne črte, ki so posledica načina izrisa, vendar so prostim očem nevidne.

6 Rezultat

Nad rezultatom sem zelo zadovoljen, vendar še vedno vidim veliko izboljšav. Prva je, da procesor ne ve, kdaj je strojna oprema izrisala črto. Trenutno po ukazu za izris črte, vsakič počakam fiksno dolžino časa, kar mi definira zgornjo mejo kompleksnosti programa.

Še ena funkcija, ki bi jo rad imel je definicija debeline črte, ki jo FPGA izriše, saj trenutno je debelina fiksna in sicer 1 piksel, kar se pri temnejših barvah slabše vidi.