Modul B: Integrirana vezja

Zagovor laboratorijskih vaj

Pong

Bruno Čeferin, 64170210

1 Opis naloge

Pri laboratorijskih vajah modula B (načrtovanje digitalnih elektronskih sistemov) sem se odločil, da bom za projekt – VGA igro – izdelal kopijo igre Pong.

Kot osnova je uporabljen grafični sistem izdelan pri laboratorijskih vajah. Sistem sem nadgradil ter mu dodal novo funkcionalnost. Gradniki sistema so prikazani na sliki 1.

2 Gradniki in nadgradnje

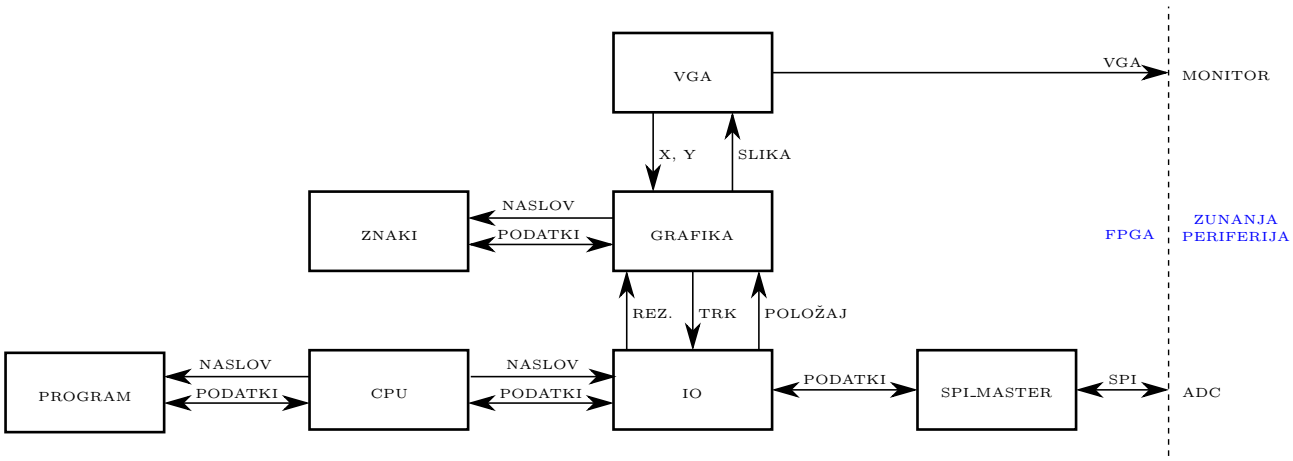
Osnova sistema je mikrokrmilnik. Tega sestavljajo centralna procesna enota (CPU), programski pomnilnik ter vhodno-izhodna enota (IO). CPU in pomnilnik sta enaka kot pri vajah. Nadgrajena je bila le IO enota. Dodal sem ji povezave z grafično enoto potrebne za določanje položaja loparjev in žoge. Poleg tega sem za branje potenciometrov za upravljanje loparjev dogradil preprost avtomat stanj, ki preko vmesniškega modula za zaporedno periferneo vodilo (SPI) krmili in zajema podatke z analogno-digitalnega pretvornika (ADC).

Za komunikacijo z ADC na razvojnem vezju sem dodal krmilni modul za komunikacijo SPI. Implementacijo tega sem poiskal na spletni strani OpenCores [1], kjer je pod odprtokodnimi licencami poleg procesorskih jeder na voljo tudi veliko različnih perifernih modulov. Izbral sem krmilnik SPI [2], ki sem ga lahko konfiguriral za pravilno delovanje v kombinaciji z ADC. Ker modul potrebujem le za branje dveh kanalov ADC, sem za krmiljenje uporabil prej omenjen avtomat stanj.

Za potrebe igre sem v grafičnem sistemu spremenil risanje "sprite"-ov tako, da omogoča risanje dveh loparjev ter žoge. Za detekcijo trka med loparjem in žogo (odboj žoge) sem uporabil obstoječ mehanizem za detekcijo trka. Poleg tega sem dodal še izpis rezultata na zaslon. Rezultat je za vsakega igralca na ekranu prikazan kot trimestna heksadecimalna številka. Tako sem se izognil potrebi po pretvorbi 12-bitne vrednosti v posamezne decimalne števke. Znaki so shranjeni v bralnem pomnilniku (ROM). Za dekodiranje števk v znake sem priredil naslavljanje znakov tako, da sem na naslovno vodilo pripeljal 4-bitno vrednost za izbor znaka ter 3-bitni x in y poziciji. Tako zaradi omejenih x in y vrednosti (znaki so širine 5 in višine 7 pikslov) ROM ni popolnoma izkoriščen, vendar je dekodiranje znakov močno poenostavljeno.

3 Delovanje igre

Delovanje igre upravlja program v mikrokrmilniku. Preko IO enote program bere stanje uporabnikovih vhodov (potenciometri za lopar, tipka za servis žoge). Napetostne vrednosti s potenciometrov program omeji na dovoljene položaje loparjev ter jih preko IO enote poda grafičnemu modulu. Tipko za servis žoge program bere le ob začetku igre ali dobljeni točki. Takrat v zanki čaka na pritisk ter ob pritisku sproži premikanje žoge.



Slika 1: Blokovna shema s pomembnejšimi signali

Slika 2: Izgled igre

Za enakomerno premikanje žoge je potrebna časovna referenca. Program za to uporablja vertikalni sinhronizacijski signal, na katerega čaka v zanki. Ko signal sprosti zanko, program preračuna nove koordinate žoge, preveri morebiten trk ali izgubo žoge ter zapiše novo hitrost žoge, če je potrebno. Koordinate žoge se računajo enostavno tako, da program vsak časovni korak komponentama položaja žoge prišteje komponenti hitrosti. Predznak posamezne komponente hitosti se spremeni ob odboju. Za odboje od zgornje in spodnje stene igrišča se spremeni predznak y komponente hitrosti. Za odboje od loparjev se predznak x komponente hitrosti zamenja. y komponenta hitrosti se takrat izračuna glede na razdaljo med sredino loparja in sredino žoge. Tako odboj bližje robu loparja odbije žogo pod bolj navpičnim kotom. Za bolj zanimivo igro se hitrost žoge po x poveča pri vsakem odboju od loparja.

Vse koordinate igre so 12-bitne vrednosti, vendar za opis položajev na igrišču zadostuje 10 bitov. Zato so vrednosti položajev ob zapisu v grafični modul pomaknjene v desno za dva bita. To format zapisa položajev spremeni v zapis s fiksnim necelim delom. S tem je poenostavljeno počasno premikanje žoge in omogočeno več kotov hitrosti žoge.

4 Povzetek rezultatov sinteze

Celoten sistem po sintezi zasede 757 vpoglednih tabel (LUT), 363 logičnih registrov ter 3839 bitov pomnilnika. Od tega porabi `sld_hub` [3] (modul za urejanje pomnilnika med delovanjem, JTAG vmesnik in podobno) 119 LUT-ov in 86 registrov. Pomnilnik uporabljata samo grafični ROM za znake in programski pomnilnik. V uporabniškem delu sistema porabi največ LUT-ov modul grafike (263), največ logičnih registrov pa CPU (56).

Literatura

[1] OpenCores. `https://opencores.org/`.

[2] Jonny Doin. SPI Master/Slave Interface. `https://opencores.org/projects/spi_master_slave`.

[3] Why do I see the design entity sld_hub:sld_hub_inst in my design? `https://www.intel.com/content/www/us/en/programmable/support/support-resources/knowledge-base/solutions/rd09012005_325.html`.