UNIVERZITET U BEOGRADU
ELEKTROTEHNIČKI FAKULTET

HARDVERSKA IMPLEMENTACIJA IGRE ŠAH

Diplomski rad

Mentor:
Doc Dr. Jelena Popović Božović

Student:
Miloš Repić 0289/2017

Beograd, Septembar 2021.

SADRŽAJ

1. UVOD

U ovom diplomskom radu biće predstavljeni problemi i rešenja hardverske realizacije popularne igre „šah“. Hardver će biti realizovan sintezom koda napisanog u VHDL-u na FPGA čipu CycloneV proizvođača Altera (Intel), na pločici DE1-SoC, korišćenjem Intelovog softvera Quartus Prime 18.1 Light Edition i ModelSim – Intel FPGA Starter Edition 10.5b

FPGA su integrisane programabilne komponente koje se koriste za implementaciju ili za funkcionalno testiranje prototipa digitalnih sistema sa visokim stepenom integracije, kompleksnom logikom, strogim *Real-time* zahtevima i specifičnom namenom.

Sistem opisan u ovom dokumentu koristi tastaturu sa PS/2 interfejsom za unošenje pokreta figura i monitorom sa VGA ulazom za prikazivanje šahovkse table i ostalih informacijama o trenutnoj partiji. U samom čipu hardverski je implementirana logika šaha, koja dozvoljava samo poteze koji su u skladu sa pravilima igre. Na četiri sedmosegmentna LED displeja prikazuje se koji karakteri sa tastature su unešeni za trenutni potez. Pritiskom tastera na tastaturi se resetuje unos, a tasterom na pločici se resetuje ceo sistem. Sistem na čipu je dizajniran tako da radi na učestanosti takta od 65 MHz.

Biće pokazano da je ovakav sistem tipičan primer sinhrone sekvencijalne mašine stanja Mealy tipa, koja na svaki takt menja stanje u zavisnosti od ulaza i trenutnog stanja, gde je stanje sistema upravo trenutno stanje šahovske table u datom momentu.

U drugom poglavlju biće opisani blok dijagram sistema, uloge svakog od blokova i njihove međusobne zavisnosti. Biće opisane uloge signala koji povezuju blokove i formati podataka koje blokovi šalju jedni drugima. Takođe, biće objašnjeni oblici ulaznih signala i očekivani oblici izlaznih signala. Biće prikazane mašine stanja za pojedine delove sistema i biće objašnjen princip rada svakog od blokova kroz analizu VHDL koda.

U trećem poglavlju biće prikazani rezultati simulacija iz ModelSim simulatora nakon analize i sinteze VHDL koda. Rezultati će biti upoređeni sa očekivanim rezultatima i biće prikazani vremenski oblici važnih signala u pojedinim modulima.

U četvrtom poglavlju će biti opisan realizovani hardver, biće procenjeno logičko zauzeće čipa i biće diskutovana maksimalna učestanost rada. Biće prikazan izgled sistema iz perspektive korisnika i biće objašnjena primena algoritama za efikasne hardverske realizacije operacije celobrojnog deljenja i ostatka pri celobrojnom deljenju. Biće predložene izmene VHDL koda za slučaj fabrikacije ovakvog uređaja i biće spomenuta neka od ograničenja projektovanog sistema. Takođe će biti objašnjen rad sa ROM memorijom za skladištenje informacija za iscrtavanje na ekranu, ukoliko se u budućnosti ovakva promena realizuje.

2. OPIS SISTEMA

2.1. Blok dijagram

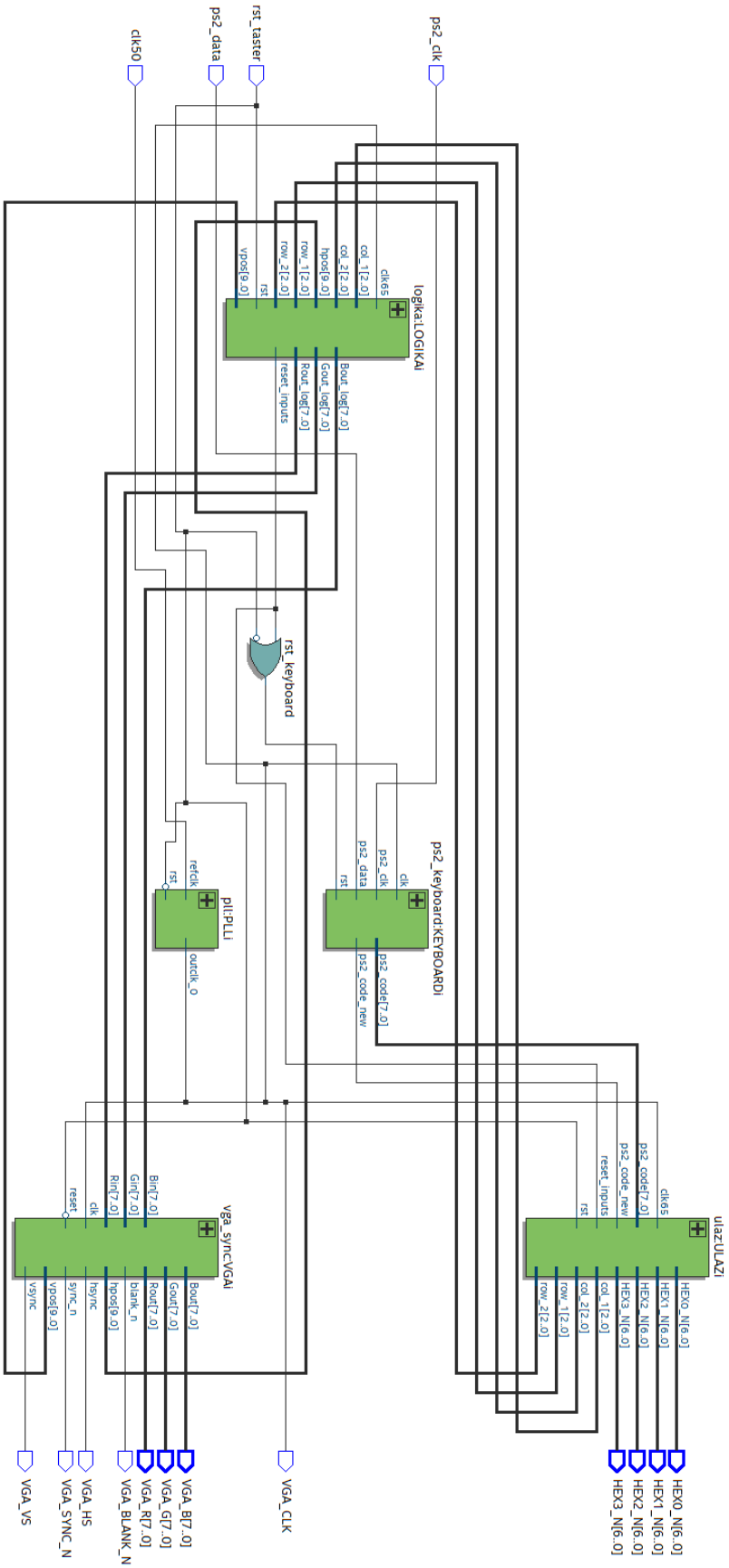
Svaki od ovih blokova implementiran je u posebnom .vhd fajlu, a zajedno su povezani u *Top-Level Entity* pod nazivom sah.vhd. Fajlovi koji se nalaze u projektu, pored navedenih su ulaz.vhd, vga_sync.vhd, seven_seg_letters.vhd, seven_seg_digits.vhd, ps2_keyboard.vhd, pkg.vhd, logika.vhd, debounce.vhd i pll.vhd. U fajlu pkg.vhd definisanom kao paket konstanti nalaze se sve konstante za konfiguraciju sistema i sva potrebna memorija za skladištenje slika. Sa slike 2.1.1. mogu se uočiti 5 osnovnih blokova sistema:

- PLL
- ULAZ
- PS2_KEYBOARD
- LOGIKA
- VGA_SYNC

Fajl logika.vhd odgovara bloku LOGIKA, fajl pll odgovara bloku PLL, fajlovi seven_seg_letters.vhd, seven_seg_digits.vhd i ulaz.vhd zajedno konfigurišu blok ULAZ, fajlovi ps2_keyboard.vhd i debounce.vhd odogvaraju bloku PS2_KEYBOARD, a fajl vga_sync.vhd odogvara bloku VGA_SYNC.

Fajlovi ps2_keyboard.vhd, debounce.vhd i vga_sync.vhd preuzeti su sa linkova koji se mogu naći u literaturi na kraju dokumenta[4] su izmenjeni za potrebe ovog projekta.

Signali *ps2_clk* i *ps2_data* su signali tipa *std_logic* koji dolaze iz tastature po protokolu PS/2 o kome će biti više reči u narednim potpoglavljima. Taktni signal koji obezbeđuje ploča DE1-SoC je učestanosti 50 MHz i on je obeležen na slici 2.1.1 signalom *clk50* takođe tipa *std_logic*. Signal *rst_taster* je signal „*hard*“ reseta, tojest, reseta koji dovodi ceo sistem u poznato stanje, i njime se resetuju svi blokovi prikazani na slici 2.1.1. Ovaj signal je aktivan sa logičkom nulom, koja se dovodi pritiskom tastera 4 na ploči DE1-SoC. Signali *HEX0_N, HEX1_N, HEX2_N* i *HEX3_N* su sedmobitni signali *std_logic_vector* tipa kojima se pale ili gase segmenti 4 sedmosegmentna LED displeja. Signali *VGA_CLK, VGA_HS, VGA_VS, VGA_BLANK_N, VGA_SYNC_N* (*std_logic), VGA_R, VGA_G* i *VGA_B* (*std_logic_vector*) su signali za upravljanje monitorom preko VGA protokola.



Слика 2.1.1. Blok dijagram sistema

Svi ostali signali obeleženi na slici 2.1.1 su signali kojima se međusobno povezuju svaki od navedenih blokova. Signal *reset_n* tipa *std_logic* je signal reseta aktivan na visokom logičkom nivou sa kojim rade blokovi PLL i VGA_SYNC. Signal *clk65* je taktni signal tipa *std_logic*. On se dobija iz PLL-a podešenog tako da stvara signal takta učestanosti 65 MHz za koji je ceo sistem projektovan. U potencijalnoj proizvodnji ovog čipa je PLL blok nepotreban pošto je moguće dovesti signal takta od 65 MHz direktno na sistem. Uloga preostalih signala biće jasnija iz detaljnijih opisa svakog od blokova u sledećim potpoglavljima.

2.2. PLL, takt i reset

Svaki taktovan blok unutar sistema projektovan je da radi na 65 MHz. Ova učestanost je izabrana tako da monitor prikazuje sliku rezolucije 1024x768 piksela i učestanosti osvežavanja ekrana (*refresh rate*) 60 Hz. Sistem može da radi i na nižim učestanostima taktnog signala i drugačijim rezolucijama i brzinama, ali je potrebno u pkg.vhd fajlu podesiti konstante koje definišu način rada VGA_SYNC bloka i proveriti da konstanta koja definiše *debounce* period tastature odgovara traženoj. Takođe je potrebno obezbediti da učestanost takta nije preniska za ispravan rad komponente ps2_keyboard.vhd. Informacije o konstantama koje treba podesiti nalaze se na linku iz literature[5]. Potrebno je podesiti PLL tako da radi na traženoj učestanosti ukoliko se vrši izmena učestanosti takta na FPGA ploči. „*Timing*“ analizom iz softverskog alata softvera Quartus Prime 18.1 Light Edition navodi se da sistem ne može garantovano raditi ispravno na učestanostima taktnog signala većoj od 78.48 MHz.

U ovom sistemu postoje više različitih reseta, od kojih su neki „soft“ reseti, a neki su „hard“ reseti. „Hard“ reset u potpunosti vraća čitav sistem u početno stanje: resetuje kolo za generisanje signala takta, briše unešene podatke iz tastature i gasi sedmosegmentne LED displeje, prekida iscrtavanje piksela na VGA displeju, i vraća stanje igre u početno, što bi značilo, šahovksa tabla se vraća u početni položaj, sledeći igrač je igrač sa belim figurama i vremena za svakog igrača su 10:00 (ili drugačije ukoliko se promeni konstanta MAXIMUM_MINUTES iz pkg.vhd). Ovaj tip reset signala dovodi se preko tastera na FPGA ploči, i koristi se u slučaju nepredviđenih grešaka u ponašanju sistema ili želje igrača za novom partijom šaha. „Soft“ reset dešava se kada jedan od igrača izgubi igru. U slučaju ovakvog reseta, šahovska tabla vraća se u početni položaj, igra beli igrač i vremena igrača se postavljaju na 10:00, Podaci primljeni iz tastature se brišu i očekuju se novi podaci, gase se svi sedmosegmentni displeji, ali se ne resetuju blokovi za kontrolu takta i iscrtavanja slike na ekranu.

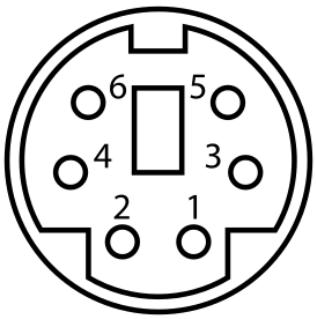
2.3. PS/2 prijemnik i obrada ulaznih podataka

2.3.1. PS/2 protokol za komunikaciju

PS/2 komunikacioni protokol[4] je bidirekcioni (u slučaju tastature je unidirekcioni) sinhroni serijski standard za slanje poruka između računara i tastature ili miša. PS/2 port ima takozvani muški ili ženski priključak za predajni uređaj i prijemni uređaj, respektivno. Sa slike 2.3.1 vidi se da PS/2 port ima 6 pinova:

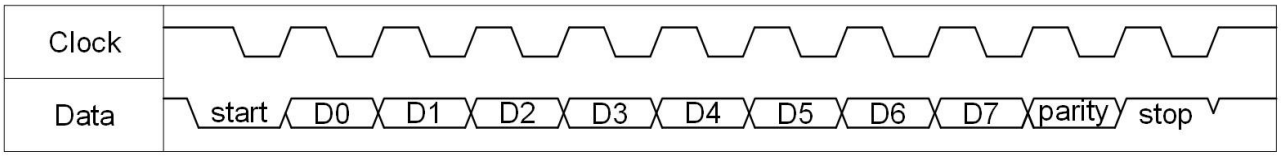
- pin 1: podaci iz tastature (ps2_data)
- pin 2: nije spojeno
- pin 3: uzemljenje (GND)
- pin 4: napajanje (VDD)
- pin 5: taktni signal (ps2_clk)

- pin 6: nije spojeno



Slika 2.3.1 Izgled PS/2 porta i raspored pinova

Signali *Data* i *Clock* sa slike 2.3.2 prikazuju princip rada PS/2 protokola. Visok naponski nivo je nekativan logički nivo. Silaznom ivicom *Data* signala započinje se komunikacija između prijemnika i predajnika. Prvom silaznom ivicom *Clock* signala proverava se da li je ispravan *start* bit, tojest, da li je signač *Data* na niskom logičkom nivou. Sledećih 8 silaznih ivica signala takta odabiraju bajt podatka koji se šalje sa tastature. Prvi poslat bit je najniži, dok je poslednji najviši. Nakon najvišeg bita, *Clock* odabira *Data* signal i proverava bit parnosti zbog provere ispravnosti slanja podataka. Bit parnosti u kombinaciji sa bitima D0 do D7 treba da daje neparan broj logičkih jedinica. Poslednji silazni takt u jednom ciklusu slanja podatka odabira *stop* bit, koji je uvek na logičkoj jedinici ako je slanje ispravno. Tipična vrednosti periode takta signala *Clock* je između 60μs i 100μs.



Slika 2.3.2 Protokol slanja podataka sa PS/2 tastature

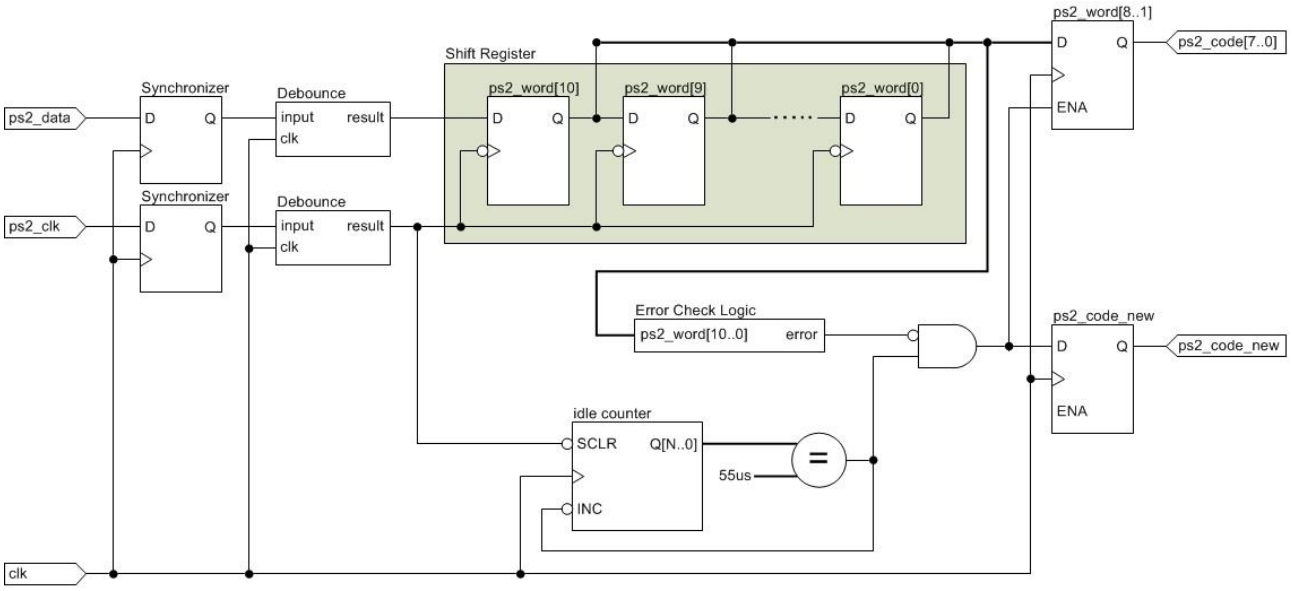
Pritisak tastera na tastaturi šalje jedan bajt informacije o pritisnutom tasteru u vidu koda karaktera broja ili slova, a puštanje tastera šalje 2 bajta informacije, gde je prvi bajt uvek F0h, a drugi bajt je isti kao kod karaktera za slovo ili broj (za ovaj sistem važni su nam samo brojevi od 1 do 8 i slova od A do H, uključujući i slovo R).

2.3.2. PS/2 prijemnik

Zadatak PS/2 prijemnika je da prepozna koji karakter je pritisnut ili pušten i da o tome obavesti blok ULAZ. Prijemnik pretvara serijski podatak u paralelni podatak, smešta ga u signal *ps2_code* tipa *std_logic_vector* (*7 downto 0*) i generiše signal *ps2_code_new* tipa *std_logic* koji obaveštava blok ULAZ da je stigao novi podatak.

Najpre je potrebno sinhronizovati sistemski takt sa taktom iz tastature. Ovime se rešava čest problem u sistemima sa više taktnih signala pod nazivom „Crossing Clock Domains“, kada može doći do pogrešne propagacije signala usled nezodovljavanja vremena držanja i uspostavljanja flip-flopova u tom delu hardvera i dovođenja delova sistema u metastabilno stanje. Takvom sinhronizovanom taktu potrebno je neutralizovati odskok, ili u engleskoj literaturi poznatiji kao „debouncing“. Zatim se na svaku silaznu ivicu takta tastature pomera shift registar i upisuje novi bit pomoću dva flip flopa koji odabiraju vrednosti takta tastature u dva različita trenutka, razmaknuta za jednu periodu sistemskig takta, a zatim se upoređuju te dve uzastopne vrednosti. Kombinaciona

logika za proveru validnosti primljenog podatka proverava bite parnosti, start bit i stop bit, i ukoliko su validni, postavlja bit greške u prenosu error na logičku nulu. Ukoliko ne postoji greška, postavljaju se signali *ps2_code* i *ps2_code_new*. Deo hardvera koji opisuje rad prijemnika PS/2 tastature opisan je u fajlu ps2_keyboard.vhd, a principska šema hardvera prijemnika prikazana je na šemi slike 2.3.2.



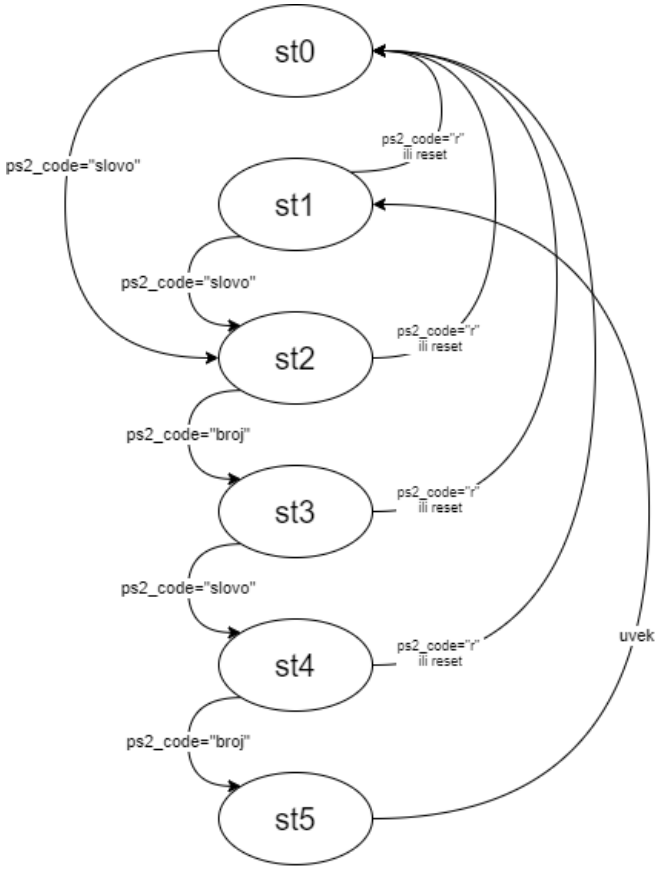
Slika 2.3.2 Principska šema hardvera prijemnika PS/2 tastature

2.3.3. Neutralizacija odskoka

Signali *Data* i *Clock* koje tastatura šalje nisu uvek stabilni i mogu se pojavljivati lažne promene signala. Ovaj problem se rešava tajmerom koji proverava da li signali drže svoje vrednosti nakon svake promene signala. Deo hardvera koji neutrališe odskoke opisan je u fajlu debounce.vhd i može se videti na slici 2.3.2, gde nosi naziv *Debounce*. Uloga bloka *idle counter* je da prepozna kada je završen prenos. To radi tako što računa da li visok naponski nivo sinhronizovanog signala takta iz tastature *ps2_clk* traje duže nego poluperioda tog takta.

2.3.4. Obrada podataka

Za obradu ulaznih podataka zaslužan je blok ULAZ, čiji je hardver opisan u fajlu ulaz.vhd. Na osnovu podataka iz PS/2 prijemnika, mašina stanja generiše izlaze. Ti izlazi su informacije o tome koja figura se pomera sa kog polja na koje polje (*row_1*,*row_2*,*col_1*,*col_2*) i informacije o tome koji segmenti sedmosegmentnih displeja se uključuju u kom trenutku. Mašina stanja je opisana dijagramom sa slike 2.3.3. Pritiskom slova A-H se prelazi iz stanja st0 u stanje st2, iz stanja st1 u stanje st2, ili iz stanja st3 u stanje st4, a pritiskom broja 1-8 se prelazi iz stanja st2 u stanje st3 ili iz stanja st4 u stanje st5. Stanje st5 traje tačno 1 taktni ciklus i u njemu se generiše izlaz koji pomera figure. Svako slovo ili broj za izlaze se koduju sa 3 bita. “000” odgovara gornjem redu ili levoj koloni (red 8 ili kolona A), a “111” odgovara donjem redu ili desnoj koloni (red 1 ili kolona H). Na ovaj način, 12 bita koduje skakanje sa bilo kog polja na bilo koje polje. Pritisak slova “R” na tastaturi, pobeda nekog od igrača (reset_inputs=’1’) ili sistemski reset vraćaju mašinu stanja u početno stanje i čeka se novi unos pokreta figure.



Slika 2.3.3 Mašina stanja bloka ULAZ

Puštanjem tastera na tastaturi šalju se 2 bajta informacije, F0h i kod puštenog tastera. Zbog ovoga može doći do pogrešnog kretanja kroz stanja. Na primer, ako bi korisnik pritisnuo taster „A“, a zatim pritisnuo taster „2“, zatim pustio taster „A“, a nakon toga pustio taster „2“, mašina stanja bi ušla u stanje st5 sa komandom „sa A2 na A2“, što ne bi bio željeni unos. Ovo ne dovodi do katastrofalnih posledica, jer ovakva komanda neće pomeriti ni jednu figuru na šahovskoj tabli, ali zato će morati da se unese novi potez. Zbog ovoga se preporučuje korisnicima da nakon što pritisnu taster, otpuste isti taster pre nego što pritisnu sledeći. Svaka greška u kucanju može se ispraviti pritiskom slova „R“ na tastaturi.

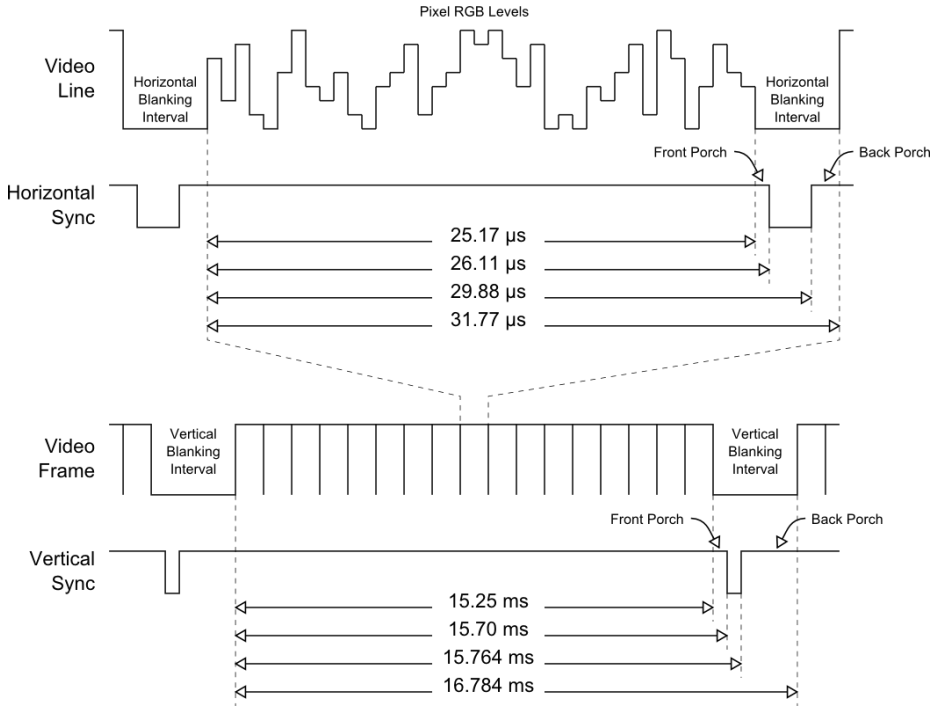
2.3.5. Sedmosegmentni displeji

Svaki od 4 sedmosegmentna displeja je kontrolisan od strane posebnih komponenti u fajlovima seven_seg_digits.vhd ili seven_seg_letters.vhd. seven_seg_letters.vhd ispisuje slova od A-H na sedmosegmentnim displejima HEX3_N i HEX1_N, a seven_seg_digits.vhd ispisuje brojeve od 1-8 na sedmosegmentnim displejima HEX2_N i HEX0_N. Ove komponente u suštini dekoduju trobitne informacije sa ulaza u sedmobitnu informaciju na izlazu za kontrolu LED dioda preko *look-up* tabela. Ove komponente imaju ulaze za dozvolu dekodovanja, sa kojom se mogu ugasiti LED displeji. Ovi ulazi se kontrolišu tako da se svakim pritiskom novog tastera sa tastature uključuje sledeći displej koji pokazuje sledeći pritisnut taster.

2.4. Komunikacija sa VGA priključkom monitora

2.4.1. VGA standard

VGA[5] je standardni interfejs za kontrolu analognih monitora. Glavni signali za upravljanje monitorom su analogni signali za R, B i G kanale piksela koji se trenutno iscrtava i hsync, vsync za poziciju piksela koji se trenutno iscrtava. Na DE1-SoC pločici, na kojoj je hardver projektovan, nalazi se ADV7123[6], DAC koji pretvara digitalne vrednosti R, G i B u analogne, prilagođene za monitor. Periodični diskretni signali vsync i hsync služe za vertikalnu i horizontalnu sinhronizaciju. Sa svakom periodom hsync signala iscrtava se novi red piksela, a sa svakom periodom vsync signala iscrtana je nova slika na ekranu. Na slici 2.4.1 vide se vremenki oblici signala hsync i vsync za neku nenavedenu rezoluciju ekrana i učestanost.



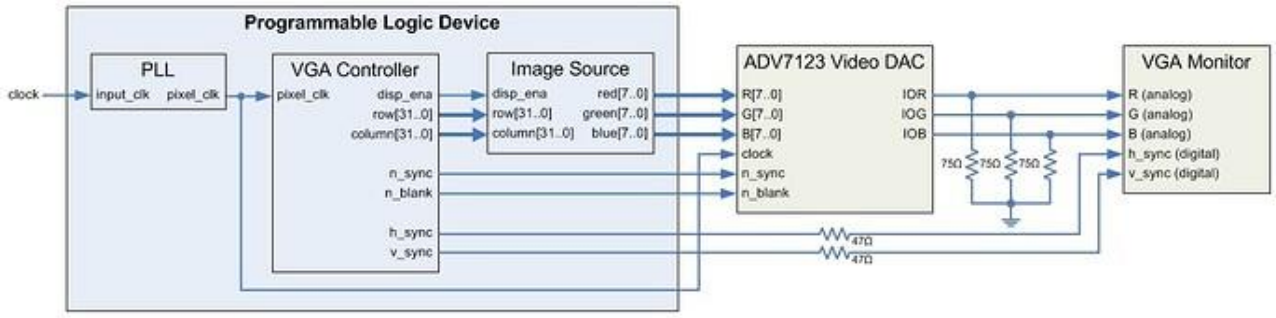
Slika 2.4.1 Vremenski oblici hsync i vsync signala[7]

U starijim displejima (CRT monitori) su se nalazili „*deflection coil*“, naelektrisane ploče koje su usmeravale snop svetlosti u piksel koji se iscrtava. Izvan „Blanking“ intervala za horizontalnu ili vertikalnu sinhronizaciju, snop svetlosti bi se kontinualno pomerao za po piksel u desno ili za red piksela nadole, iscrtavajući sledeći piksel na ekranu. Tokom „Blanking“ intervala, vertikalni ili horizontalni „*deflection coil*“ su pokazivali na piksele koji se ne nalaze na ekranu ili su se vraćali ulevo („*Horisontal Blanking Period*“) ili nagore („*Vertical Blanking Period*“). Ovaj standard za signale su nasledile sledeće generacije monitora sa novijim tehnologijama.

2.4.2. VGA_SYNC blok

Hardver bloka VGA_SYNC opisan u fajlu vga_sync.vhd ima ulogu kontrolera VGA displeja. On usaglašava izlaze bloka logika sa VGA monitorom. Nizovi brojača definišu signale hpos, vpos, hsync i vsync na osnovu širine i dužine displeja izraženih u pikselima i konstanti definisanih za vrednosti dužine vremena „*Front Porch“* i „*Back Porch“* za vertikalno i horizontalno pomeranje za traženu rezoluciju ekrana i broj piksela u sekundi. Signali hsync i vsync dovode se na pinove VGA konektora, a signali hpos i vpos tipa *integer* šalju se na blok LOGIKA.

Na osnovu njih blok LOGIKA pomoću interne kombinacione logike dovodi vrednosti na Rout_log, Gout_log i Bout_log, koji se kasnije sprovode do DA konvertora. Na slici 2.4.2 dat je primer hardvera za kontrolu VGA displeja. Suštinska razlika šeme sa slike 2.4.2 od šeme hardvera sa realizovanog sistema je ta da u realizovanom sistemu hardvera šaha VGA kontroler zadaje boje DA konvertoru na osnovu informacija iz bloka LOGIKA, dok u sistemu prikazanom na šemi 2.4.2 blok koji se bavi logikom dodele crvene, zelene i plave boje (*Image Souce*) dodeljuje boje DA konvertoru na osnovu horizontalne i vertikalne pozicije iz VGA kontrolera.



Slika 2.4.2 Principska šema rada VGA kontrolera

2.5. Logika šaha, iscrtavanja i merenja proteklog vremena

2.5.1. Logika šaha

Cilj igre, projektovane na ovaj način, je pojesti protivnikovog kralja pre isteka vremena. Zato je obaveza igrača da vode računa o tome da li je kralj pod šahom, patom, ili šah matom, kao što bi radili i da igraju preko prave šahovske table. Logika ovog sistema ne vodi računa o ovim situacijama, ali ne dozvoljava protivniku da pomera figure na način koji nije u skladu sa pravilima pomeranja figura u šahu. Takođe, kada pijun dođe do drugog kraja šahovske table, uvek postaje kraljica, gde bi se u pravom šahu biralo između 4 različite figure: kraljice, lovca, topa ili konja. Ovo je opravdano pošto je u retkim situacijima poželjnija promocija u konja nego u kraljicu, a nikada u lovca ili topa nego u kraljicu.

Pijuni se mogu pomerati za jedno polje unapred ako je to polje slobodno ili dva polja unapred u svom prvom pomeranju, ali ne mogu preskočiti figuru. Pijuni mogu pojesti figuru koja se nalazi za polje dijagonalno u oba smera, ili mogu pojesti drugog pijuna „u prolazu“ („*en passant*“) ako je to dozvoljeno. Pojesti pijuna u prolazu je dozvoljeno samo nakon poteza u kome je protivnikov pijun preskočio polje. Pijun uvek postaje kraljica kada dođe do kraja table. Konj se uvek pomera tako da je zbir horizontalnog i vertikalnog pomeraja jednak 3, ali potez nije vertikalan ili horizontalan (kreće se u ćiriličnog obliku slova „Г“). Top se kreće horizontalno ili vertikalno u odnosu na polje na kome se nalazi, a lovac se kreće po dijagonali u odnosu na polje na kome se nalazi. Kraljica može da se kreće ili kao top ili kao lovac u bilo kom potezu. Kraljica, top i lovac ne mogu preskakati preko figura. Kralj se kreće za po jedno polje u bilo kom pravcu. Kada se između nepomerenog kralja i nepomerenog topa ne nalaze figure, može se uraditi rokada kao potez. Rokada podrazumeva da se kralj pomeri za dva polja prema topu, a top prelazi na suprotnu stranu kralja, polje pored njega. Ovo je jedini potez u kome se pomeraju dve figure. Prazno polje se ne može pomerati.

O dozvoljenim ili nedozvoljenim pokretima figura vodi računa proces PIECE_MOVEMENT u fajlu logika.vhd. Najvažniji signal u bloku LOGIC je *board*, veličine 8x8, dubine 5 bita, i on predstavlja šahovsku tablu sa figurama. Najviši bit govori da li je figura crna ili

bela (‘0’ za prazno polje). Najniža 3 bita koduju jednu od 6 različitih figura ili prazno polje. Četvrti bit po težini označava da je figura specijalna na neki način. Ovaj bit je jednak logičkoj jedinici za pijuna ako pijun može biti pojeden „*en passaant*“, logičkoj nuli za kralja ako sme da uradi rokadu, ili logičkoj nuli za topa ako sme da uradi rokadu. Taj bit je uvek jednak logičkoj nuli za sve ostale figure. Najniža 3 bita svakog polja koduju figure na sledeći način :

- “000” - prazno polje
- “001” - pijun
- “010” - lovac
- “011” - konj
- “100” - top
- “101” - kraljica
- “110” - kralj

Igrač koji je na potezu bira koja će se figura pomeriti tako što preko tastature unese obeležje polja sa koga se figura pomera u formatu slovo-broj, a zatim obeležje polja na koje se figura pomera, takođe u formatu slovo-broj. Blok ULAZ prevodi te informacije u 3 trobitne vrednosti za kolonu sa koje se pomera, red sa koga se pomera, kolonu na koju se pomera, red na koji se pomera. Neispravnim unosom se šahovska tabla ne menja, a mašina stanja bloka ULAZ ostaje u istom stanju. Potez se dešava samo na uzlaznu ivicu signala takta. Pošto se prazno polje ne može pomeriti, a svaki potez ostavlja prazno polje na mestu sa koga se figura pomerila, nije potrebno signalizirati da se potez izvršava i nije potrebno menjati ulaze za pokrete u tom slučaju.

Signal *white_turn_black_turn* je na početku igre jednak logičkoj jedinici, i označava da li je na potezu beli igrač. Ovaj signal menja svoju logičku vrednost svaki put kada se odigra validan potez. Svaki reset vraća igru u početno stanje, i time, vraća ovaj signal na logičku jedinicu. U gornjem levom uglu ekrana se na osnovu signala *white_turn_black_turn* ispisuje koji od igrača je na potezu („WHITE“ ili „BLACK“).

Kretanje lovca, topa ili kraljice obavlja se pomoću variabli *distance* tipa *integer* i *ok* tipa *std_logic_vector (1 to 7).* Variabla *distance* ima vrednost broja polja za koliko kraljica, lovac ili top treba da se pomere. Variabla *ok* dozvoljava ili ne dozvoljava da se lovac, kraljica ili top kreću na tražen način. Levih *distance-1* bita variable *ok* se setuju samo ako se između topa, lovca ili kraljice i polja na koje skaču nalaze prazna polja. Svi ostali biti varijable *ok* su uvek setovani. Ako su svi biti variable *ok* setovani, figura može da se pomeri pošto se između nje i polja na koje skače ne nalazi ni jedna figura.

Na svaku uzlaznu ivicu signala takta, kombinacionom logikom se proverava da li se na šahovskoj tabli ne nalazi beli ili crni kralj. Nedostatak nekog od kralja znači da je jedan od igrača izgubio, i ovaj događaj dovodi do „*soft*“ reseta sistema, tojest, nove partije .

Ako je igrač pomerio pijuna za dva polja unapred setuje se bit koji označava pijuna kao „prekakajućeg pijuna“. Sledeći put kada isti igrač igra potez, na prvu ivicu signala takta se ovaj bit resetuje, jer pijun više nije „preskakajući“. Ovo izvršava kombinaciona logika koja proverava da li se na nekom od polja nalazi „preskakajući pijun“ koji pripada igraču koji je na potezu. Na sledeću uzlaznu ivicu takta partija se nastavlja i čeka se pokret korisnika. Ukoliko nema „preskakajućeg pijuna“, ovaj korak se preskače.

2.5.2. Logika iscrtavanja na displeju

U paketu konstanti pkg.vhd nalaze se sve zajedničke konstante kojima pristupaju ostali .vhd fajlovi. Iako se na ekranu iscrtavaju 4 različite nijanse sive, moguće je sačuvati informaciju za svaki piksel u jednom bitu. Zato su informacije smeštene u matrice dubine jednog bita, a svaka matrica može imati posebno značenje (polje figure, cifra za ispis vremena...). Informacija o tome da li se iscrtava beli, svetlo sivi, tamno sivi ili crni piksel na ekranu zavisi od informacije o tome da li je figura crna ili bela, i da li je polje tamno sivo ili svetlo sivo. Da bi se uštedelo na memoriji, rezolucija svake od matrica sa informacijama ne odgovara rezoluciji te matrice kada se iscrtava na ekranu, sem pri iscrtavanju figura sa poljem. Figure sa poljem zauzimaju matricu u memoriji 64x64 bita, a veličina slike svakog polja sa figurom na ekranu je takođe 64x64 piksela. Matrice u memoriji su dobijene digitalnom obradom slike iz literature[8] u programskom jeziku Python.

Šahovksa tabla će zauzeti centralnih 512x512 piksela ekrana. Najpre je potrebno izračunati relativan *hpos* i *vpos* u odnosu na gornji levi piksel šahovske table tako da iscrtavanje šahovske table počne na centru ekrana. Zatim se koriste biti 8-6 horizontalne i verkalne pozicije relativne u odnosu na horizontalnu i vertikalnu poziciju gornjeg levog piksela za informaciju o figuri koja se trenutno iscrtava. Najnižim bitom zbira horizontalne i vertikalne pozicije figure koja se iscrtava na šahovskoj tabli dobija se informacija da li je polje crno ili belo. U paketu pkg.vhd se nalaze slike svih mogućih figura. Polja su širine 64x64 piksela, i u memoriji su smeštene u matricu 64x64 dubine 1 bit. Zbog toga se za horizontalne i vertikalne koordinate informacije u memoriji koriste biti 5-0 horizontalne i verkalne pozicije relativne u odnosu na horizontalnu i vertikalnu poziciju gornjeg levog piksela šahovske table. U memorijskoj lokaciji date figure sa poljem je bit 1 tamo gde trebaiscrtati figuru, a bit je 0 tako gde treba iscrtati pozadinu.

U pkg.vhd nalaze se i matrice informacija koje iscrtavanju koji je igrač trenutno na potezu. U zavisnosti od signala *white_turn_black_turn*, pristupa se ili konstanti „*WhiteTurn*“ ili konstanti „*BlackTurn*“. Nije potreban pomeraj za hpos i vpos jer se ova informacija slika u gornjem levom ćošku ekrana, ali se horizontalna i vertikalna pozicija dele celobrojno sa 2 funkcijom „ASR“ da bi se dobio indeks u memoriji prethodno pomenutih konstanti. Razlog za to je što je matrica u memoriji veličine 18x48, a veličina dela ekrana u kojem se iscrtava je 36x96.

Za crtanje trenutnog vremena u gornjem i donjem desnom ćošku ekrana, koristi se isti princip, računa se horizontalni i vertikalni pomeraj tako da se data cifra crta počevši od tačno određenog piksela, a zatim se iz memorije izvlači informacija na osnovu pomerene vertikalne i horizontalne pozicije piksela za iscrtavanje, podeljene celobrojno sa 4 funkcijom „ASR2“. Izgled slike cifara se nalazi u memoriji kao niz matrica pod nazivom „*Digits_0to9*“. Svaki indeks niza matrica u „*Digits_0to9*“ odgovara tačno toj cifri koja se crta. Identičnim principom crtaju se oznake polja (A-H,1-8) smeštene u memoriji pod nazivom „Red“ i „Kolona“.

2.5.3. Logika merenja proteklog vremena

Svakim taktom se unutar procesa TIME_MENAGMENT iz fajla logika.vhd dekrementira jedan od dva brojača taktova nadole koji broje od 65000000 (konstanta freq iz pkg.vhd) do 0 (po jedan brojač sa svakog igrača) u zavisnosti od vrednosti signala *white_turn_black_turn*. Dolaskom ovog brojača do 0, prošla je sekunda i tada se dekrementira broja sekundi za tog igrača koji počinje od 0, a dolaskom broja sekundi do 0, dekrementira se broj minuta tog igrača koji počinje od MAXIMUM_MINUTES (konstanta iz pkg.vhd). Dekrementiranje sekundi kada je vrednost nula vraća broj sekundi na 59, a dekrementiranje brojača taktova kada mu je vrednost nula, vraća vrednost tog brojača na 65000000. Kada za nekog od igrača svi brojači dođu do nule, igra se

resetuje. Bilo koji reset postavlja brojač taktova na freq, brojač sekundi na 0 i brojač minuta na MAXIMUM_MINUTES.

Iz vrednosti sekundi i minuta za svakog od igrača izvlače se informacije o ciframa na ekranu kojima se prati trenutno preostalo vreme. Na mestu jedinica sekundi je cifra koja se dobija kao ostatak deljenja sa 10 broja sekundi, na mestu desetica sekundi je cifra koja se dobija celobrojnim deljenjem sa 10 broja sekundi, na mestu jedinica minuta je cifra koja se dobija kao ostatak deljenja sa 10 broja minuta, a na mestu desetica minuta je cifra koja se dobija celobrojnim deljenjem sa 10 broja minuta. Celobrojno deljenje sa 10 realizovano je funkcijom DIV10 koja se nalazi u fajlu logika.vhd, a ostatak deljenja sa 10 realizovan je funkcijom REM10 koja se takođe nalazi u fajlu logika.vhd.

Po startovanju nove igre, čeka se da igrač sa belim figurama odigra potez. Do tada ne teče vreme ni za jednog ni za drugog igrača. Ovo je postignuto poređenjem trenutne pozicije šahovske table sa početnom pozicijom šahovske table. Drugim rečima, ako je signal *board* jednak konstanti *pocetni_polozaj*, ne dozvoljava se dekrementiranje brojača. Ovo bi moglo da dovede do situacije u kojoj vreme ne teče igračima nakon 4 poteza. Jedini slučaj da se ovako nešto desi je ukoliko igrači odigraju poteze G1F3, B8C6, F3G1, C6B8. Sada se vreme vraća na 10:00 za oba igrača, iako je igra počela i nije se završila. Opravdanje za ovako nešto je da ne postoji dobro opravdanje za igrače da odigraju poteze G1F3, B8C6, F3G1, C6B8 na početku igre, s obzirom da ne vodi pobedi ni jednog ni drugog igrača, a potrebna bi bila dodatna logika koja bi vodila računa o tome da je odigrani potez prvi u igri ili ne.

3. REZULTATI SIMULACIJA

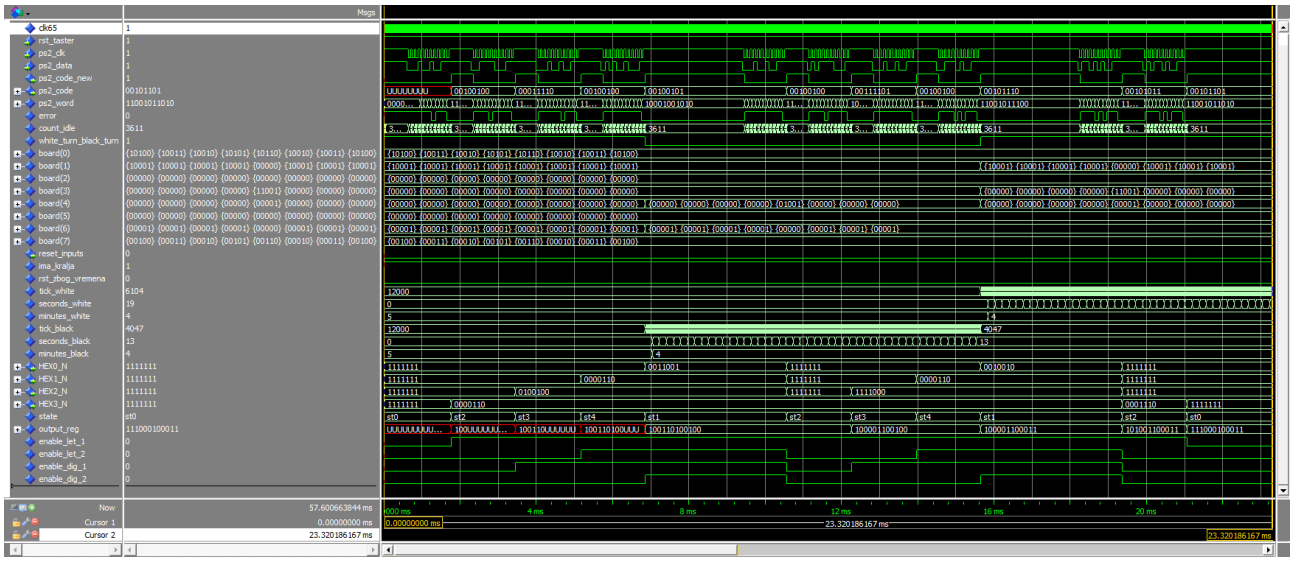
3.1. Promena signala nakon legalnog poteza

U ovom potpoglavlju prikazaće se ispravnost funkcionalnosti igre, pokazaće se na nekoliko primera da se figure kreću kao što se očekuje, a takođe će se prikazati da sedmosegmentni displeji prikazuju ispravne podatke. Za izvođenje simulacija korišćen je softverski alat ModelSim – Intel FPGA, verzija Starter Edition 10.5b. Potrebno je napomenuti da vreme u simulaciji teče brže radi demonstracije reseta zbog isteka vremena. Ovo je urađeno promenom konstante freq iz pkg.vhd sa 65000000 na neku manju vrednost koja odgovara simulaciji.

Na slici 3.1.1 vide se signali nakon poteza E2E4, E7E5, FR. Beli pijun nakon poteza E2E4 preskače polje i postaje „preskakajući pijun“ tako što mu se setuje bit 3 (board(4),peta kolona,10 ms, vrednost “01001”). Nakon poteza E7E5 Taj pijun više nije „preskakajući“ (vrednost “00001”), ali crni pomereni pijun jeste (board(3), peta kolona, 17ms, vrednost “11001”).

Vidi se da se nakon svakog pritiska tastera na tastaturi uključuje sledeći sedmosegmentni displej (signali HEX_N) i da se prelazi iz jednog stanja u sledeće. Nakon stanja st4 se postavljaju ulazi za pomeranje figura row_1, col_1, row_2 i col_2, a signal *board* dobija novu vrednost i *white_move_black_move* menja svoju vrednost.

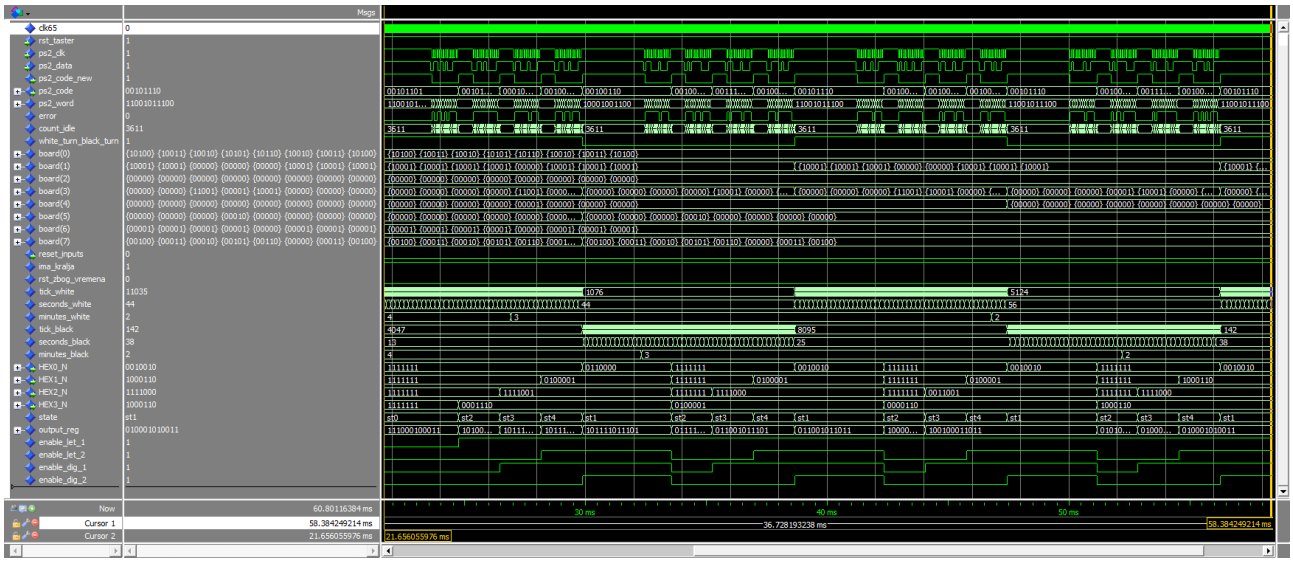
Nakon pritiska tastera F, a zatim pritiska tastera R, mašina stanja prelazi u početno stanje st0, gase se sve diode sedmosegmentnih displeja pomoću signala dozvole prikaza na sedmosegmentnim displejima *enable_dig1*, *enable_dig2*, *enable_let1*, i *enable_let2* i čeka se novi unos.



Slika 3.1.1 Pomeranje pijuna i pritisak „R“ tastera

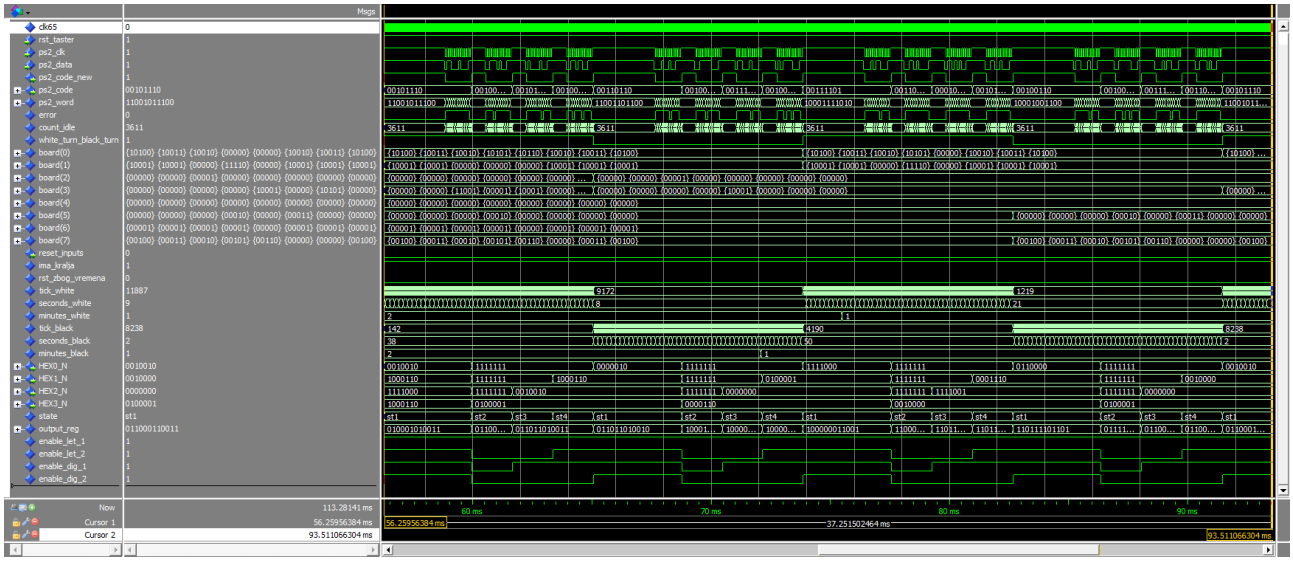
Sledeća 4 poteza su F1D3, D7D5, D4D5, C7C5 i stanje šahovske table nakon ta 4 poteza vidi se na slici 3.1.2 sa leve strane, pored dijagrama. U prvom potezu ispravno je pomeren beli lovac sa F1 na D3. Zatim se pomera crni pijun sa D7 na D5. Zatim se pomera beli pijun sa D7 na

D5, a zatim crni pijun sa C7 na C5. Ovo znači da pijun sa D5 može da pojede pijuna na C5 „en passant“.



Slika 3.1.2 Sledeća 4 poteza

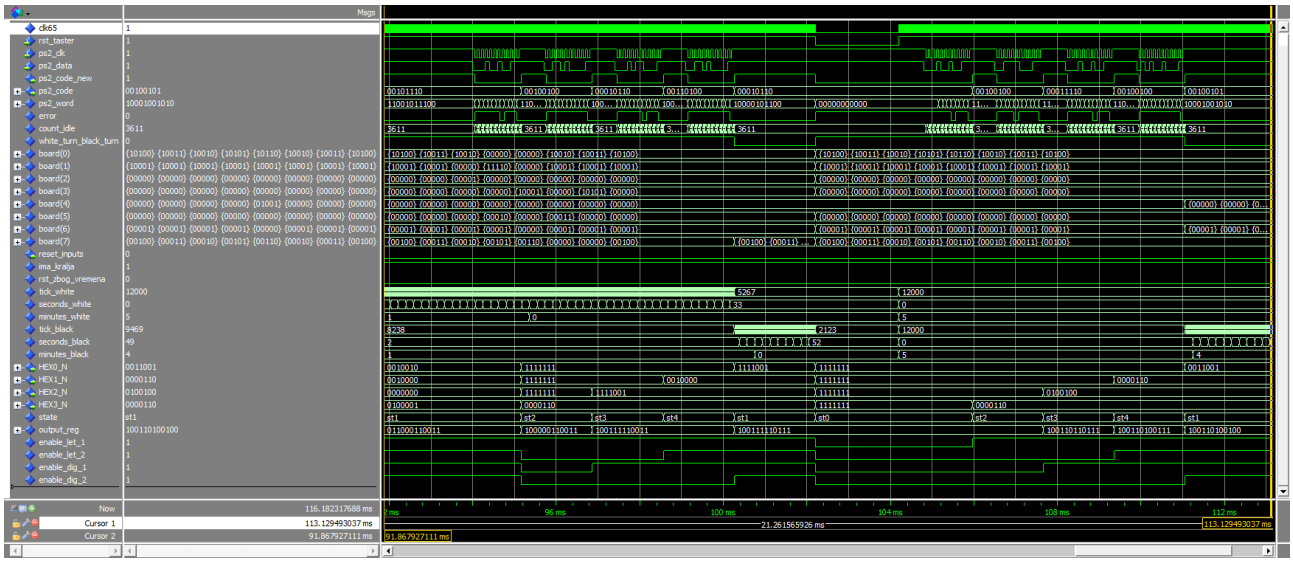
Sledeća 4 poteza su D5C6, E8D7, G1F3, D8G5, i stanje šahovske table može se videti na slici 3.1.3, levo od vremenskih dijagrama. U prvom potezu crni pijun sa polja C5 je pojeden „en passant“. Zatim se crni kralj pomera sa E8 na D7. Ovo inače ne bi bilo dozvoljeno zbog belog pijuna na C6, ali je ovde prikazano zbog demonstracije logike kretanja figura. Takođe, bit 3 kralja se setuje, i više ovaj kralj ne može uraditi rokadu. Zatim se beli konj pomera sa G1 na F3. Potom crna kraljica prelazi sa D8 na G5. Beli kralj je na potezu i pojavio se uslov za rokadu.



Slika 3.1.3 Sledeća 4 poteza

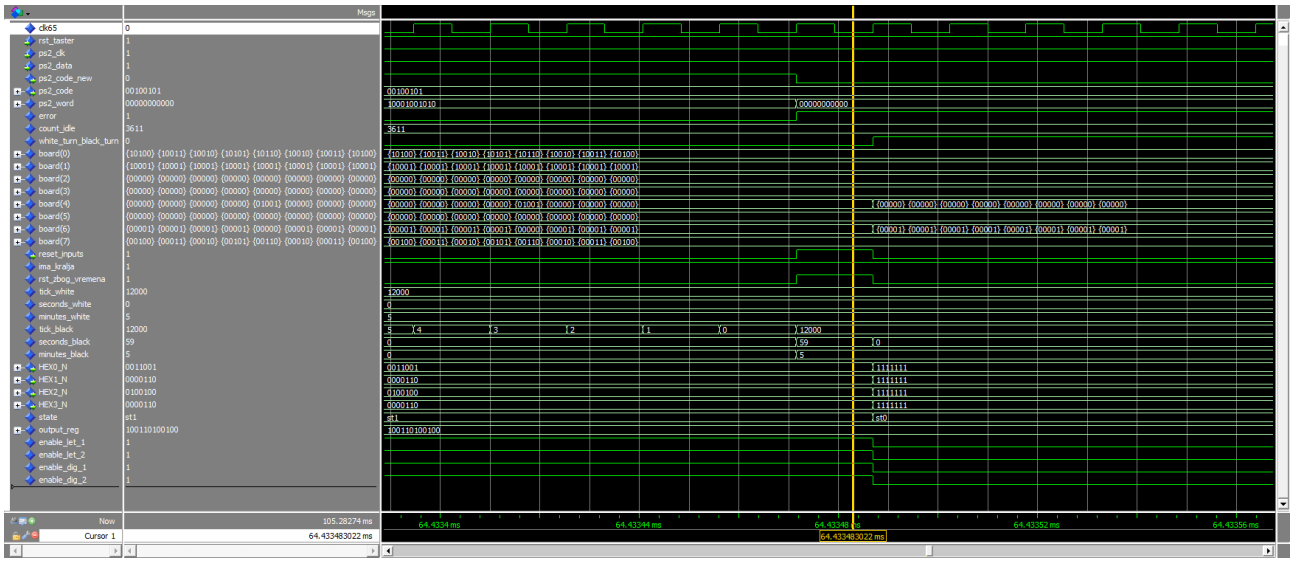
U sledećem potezu može se videti sa slike 3.1.4 da beli igrač vrši rokadu potezom E1G1. Sa leve strane slike može se videti stanje šahovske table nakon ovog poteza. U poslednjem redu signala *board* vide se pozicija belog kralja i belog topa nakon rokade. I belom kralju i belom topu se setuje bit 3 koji znači da su pomereni i da ne mogu ponovo uraditi rokadu. Tokom signala reseta sa tastera, šahovska tabla se vraća u početni položaj, igra beli igrač i vreme je zamrznuto na 10:00

za oba igrača i čeka se da igrač sa belim figurama odigra potez. Nakon reseta se igra potez E2E4 da bi se demonstriralo da sistem radi ispravno nakon reseta.



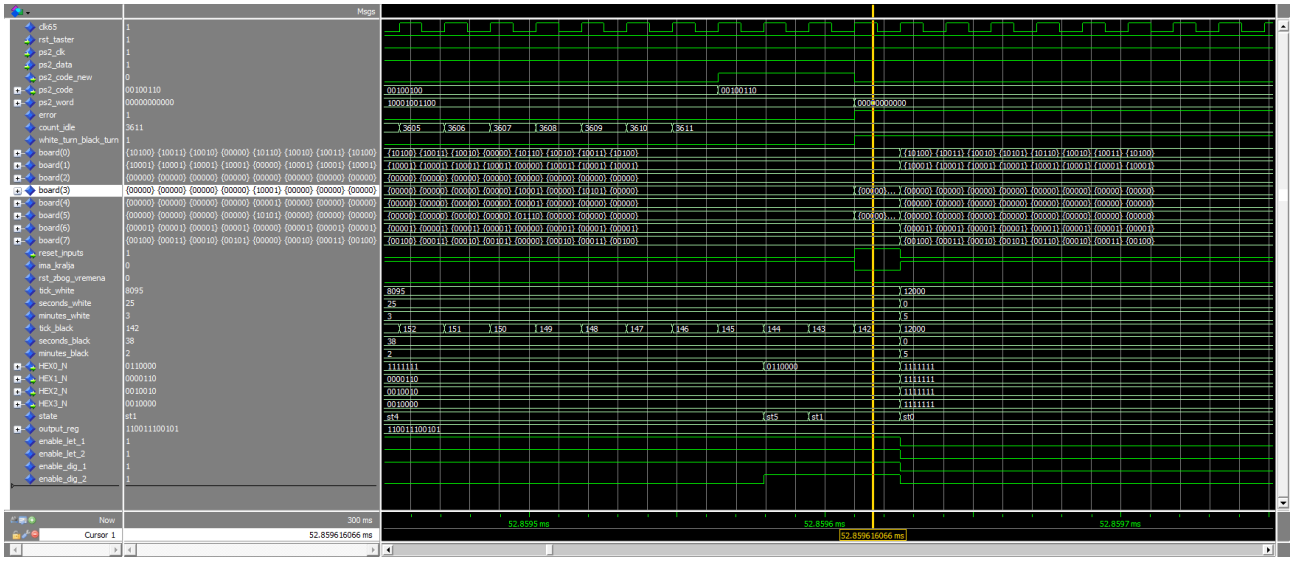
Slika 3.1.4 Rokada i reset

Sa slike 3.1.5 vidi se šta se dešava u slučaju da istekne vreme nekom od igrača. Signal *rst_zbog_vremena* postaje logička jedinica, šahovska tabla se vraća u početni položaj, vremena se vraćaju na maksimalna, igra beli igrač i čeka se na njegov potez.



Slika 3.1.5 Isteklo je vreme belom igraču

Sa slike 3.1.6 vidi se šta se dešava u slučaju da je pojeden neki od kraljeva. U ovom slučaju, crna kraljica sa G5 jede belog kralja sa E3. Deaktivira se signal *ima_kralja* i nakon što se kraljica pomeri, kralj je pojeden. Tada kombinaciona logika detektuje da fali jedan od kraljeva i na sledeću ivicu signala takta igra se resetuje. Figure se vraćaju na početni položaj, vreme se vraća na početno i igra beli igrač.

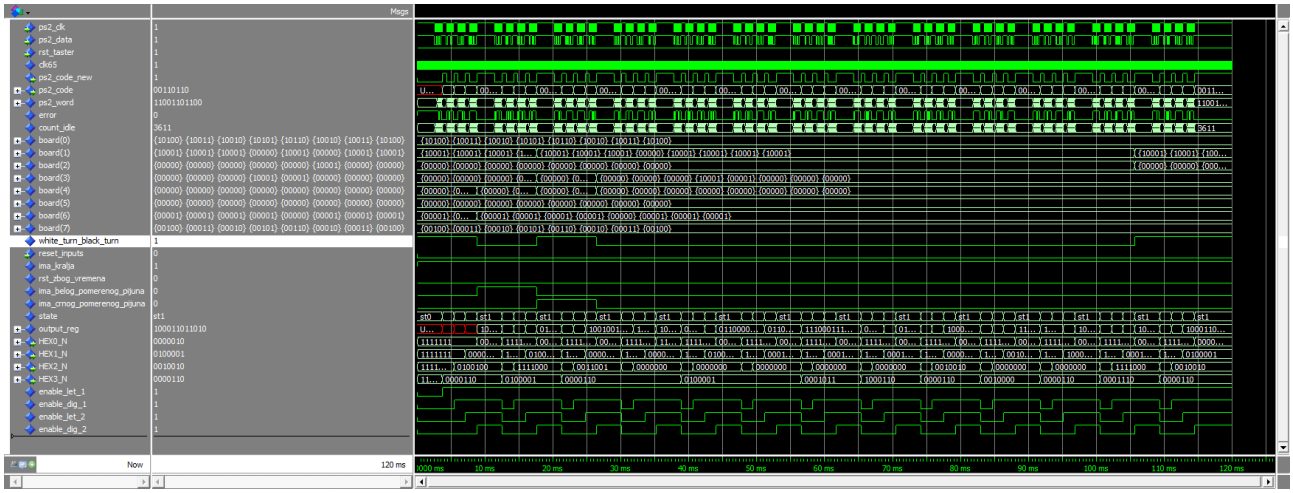


Slika 3.1.6 Pojeden je beli kralj

3.2. Ispitivanje nelegalnih poteza

Na slici 3.2.1 vidi se skup od 8 različitih nelegalnih poteza. Potezi odigrani od početka igre su redom: E2E4, D7D5, E4E5, E8E7, D8D4, D8H4, H8H4, C8A6, E5E6, G8G5, E8C8, F7F6, E5D6. Posmatraće se potezi od momenta prikazanog žutim kursorom, nakon poteza E4E5. U tom momentu, crni pijun je na polju D4, a beli pijun je na polju E4.

Potezom E8E7 bi se kralj pomerio sa E8 na svog pijuna, što nije dozvoljeno. Potezom D8D4 i D8H4 demonstrira se da kraljica ne može da preskače preko figure. Potezima C8A6 se pokazuje da lovac ne može da preskače figure, a potezom H8H4 se demonstrira da top ne može da preskače figure. Potez E5E6 ne može da se odigra zato što je crni igrač na potezu, a G8G5 ne može da se odigra zato što se konj ne kreće na taj način. F7F6 je dozvoljen potez, ali E5D6 nije jer pijun može da se pojede u prolazu samo potez nakon što je protivnik preskočio polje sa pijunom. Lako se može videti da potezi nisu dozvoljeni činjenicom da se signali *white_turn_black_turn* i *board* menjaju samo nakon dozvoljenog poteza.

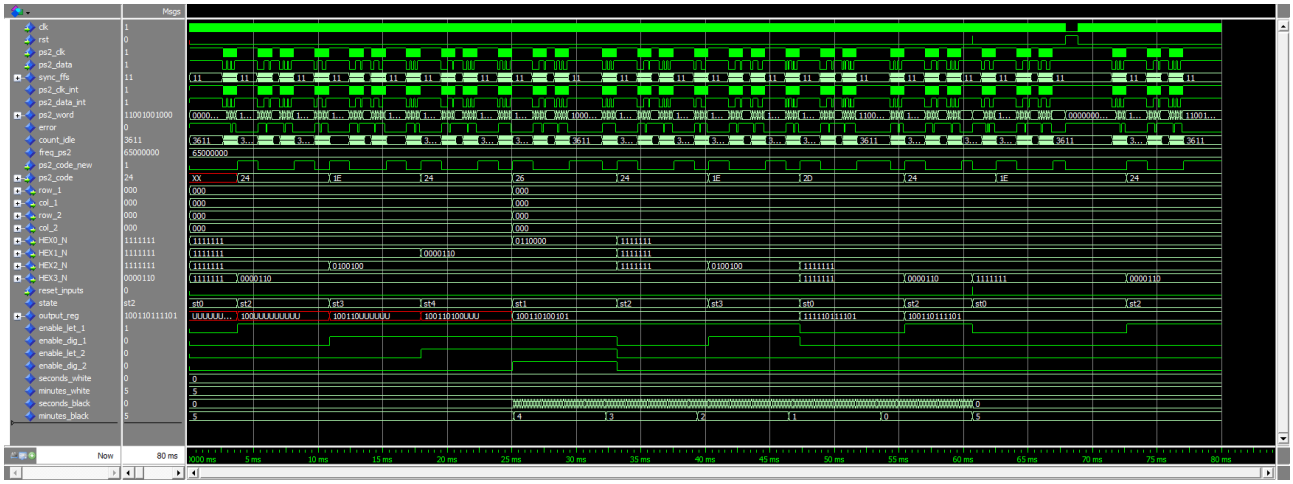


Slika 3.2.1 Skup nelegalnih poteza

3.3. Tastatura i obrada ulaznih podataka

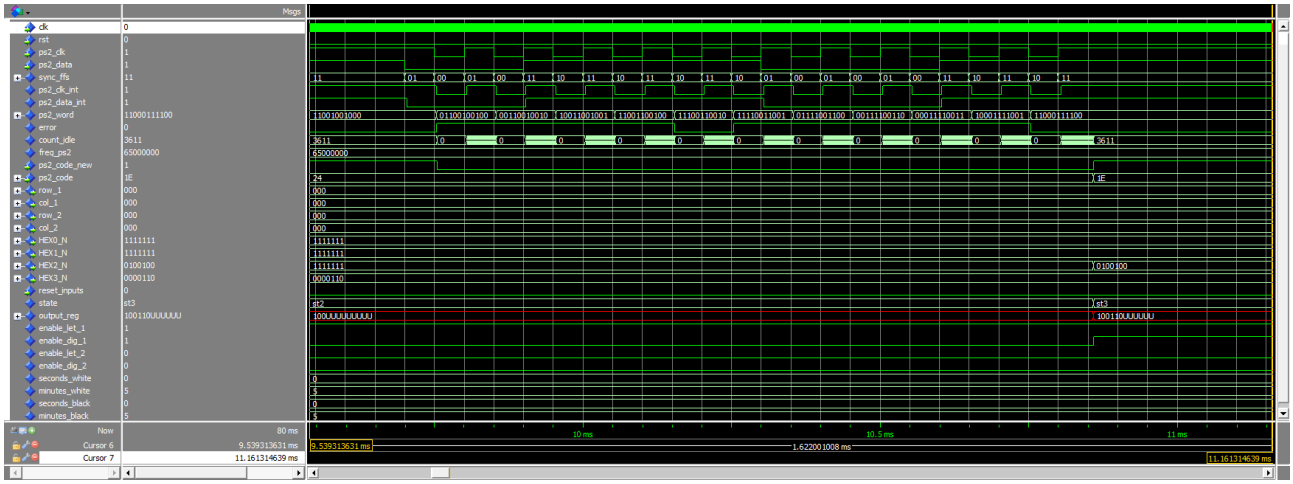
U ovom potpoglavlju prikazaće se važni signali iz blokova PS2_KEYBOARD i ULAZ i simuliraće se neke od mogućih situacija. Trajanje jedne sekunde u simulaciji je promenjeno u odnosu na trajanje sekunde u krajnjoj realizaciji radi prikaza promene signala usled reseta pri isteku vremena nekom od igrača.

Na slici 3.3.1 vidi se rezultat simulacije u kojoj su prikazani signali nakon uspešnog unošenja 4 karaktera za pomeranje figure (25 ms), unos 2 karaktera, a zatim pritisak tastera „R“ sa tastature za ponovni unos (47.2 ms), reset usled isteka vremena (60.6 ms) i stanje nakon sistemskog reseta (68.9 ms). Radi verodostojnosti, u simulaciji se simulira i puštanje tastera pre pritiska sledećeg, u kom slučaju tastatura šalje kod F0, a zatim kod tastera koji je pušten kako bi se prikazalo ispravno kretanje kroz stanja.



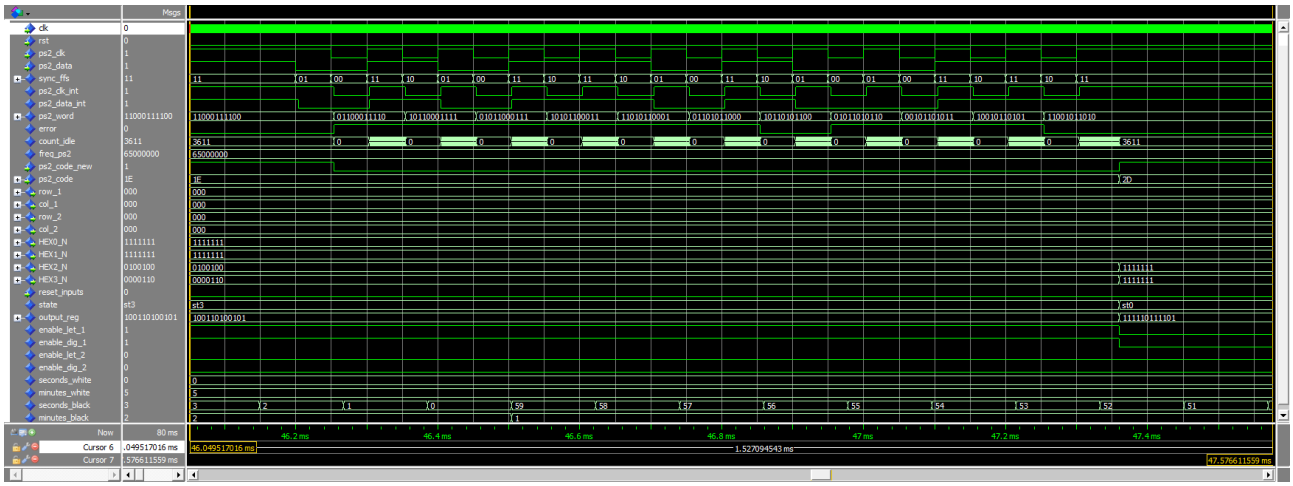
Slika 3.3.1 Niz pritisnutih tastera tastarure i reset

Na slici 3.3.2 vidi se promena signala nakon unosa karaktera „2“ sa tastature. Tastatura po pritisku tastera spušta naponsku vrednost *ps2_data* signala, a zatim kreće oscilacija *ps2_clk* signala sa periodom od 100 μs. Neutralizacijom odskoka ova dva signala dobijaju se signali *ps2_data_int i ps2_clk_int,* koji će formirati konačni izlazni podatak. Na svaku silaznu ivicu signala *ps2_clk_int* se odabira signal *ps2_data_int* i upisuje su *shift* registar označen sa *ps2_word*. Nakon poslate cele poruke, po dobrojavanju brojača *count_idle* do 3611, ako su *stop*, *parity* i *start* biti ispravni, postavlja se *ps2_code _new* signal koji označava da je nova poruka uspešno primljena, a u izlazni signal *ps2_code* se upisuje novoprimljena 8-bitna poruka. U ovom slučaju, mašina stanja bloka ULAZ prelazi iz stanja st2 u stanje st3, i sada svetle diode leva dva sedmosegmentna displeja. Signali *row_1*, *row_2*, *col_1* i *col_2* kojima se pomeraju figure dobijaju vrednosti samo u stanju st5 tokom jedne periode sistemskog takta kada su unešena 4 tastera sa tastature u određenom redosledu.



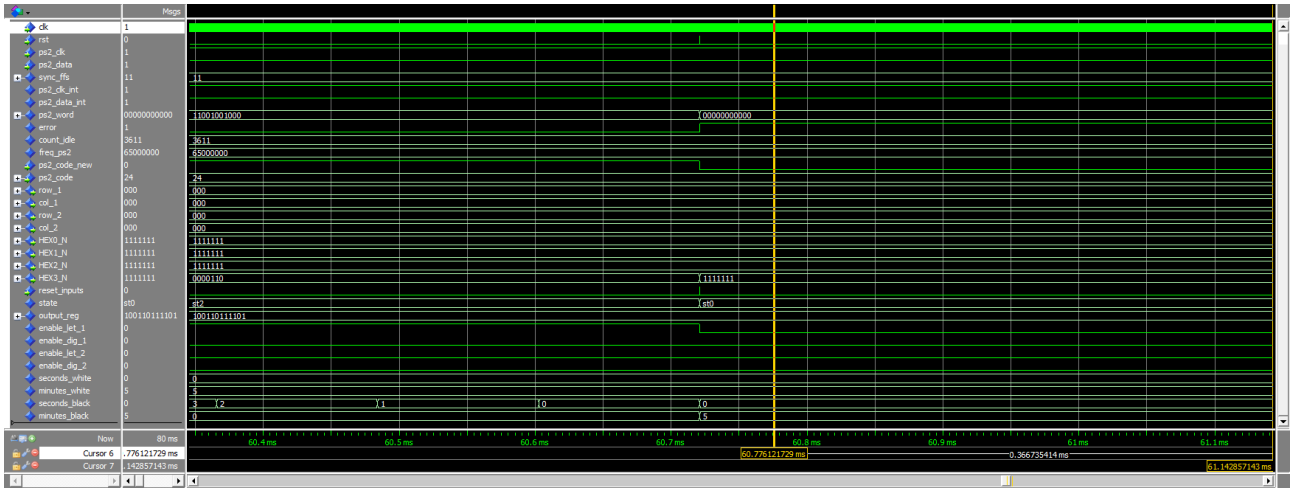
Slika 3.3.2 Primer Unosa karaktera sa tastature

Na slici 3.3.3 vidi se promena signala kao posledica pritiska tastera „R“ sa tastature, čiji je heksadecimalni kod 1Eh. Sistem se vraća u stanje st0, kao što je i očekivano, potez koji se zahteva je A8A8, što neće dovesti do promene šahovske table, svi sedmosegmentni displeji su isključeni, igra isti igrač, a vreme za potez nastavlja da teče tom igraču. Sistem je sada u stanju u kome očekuje jedan od karaktera izmeću A i H.



Slika 3.3.3 Stanje nakon pritiska tastera „R“ sa tastature

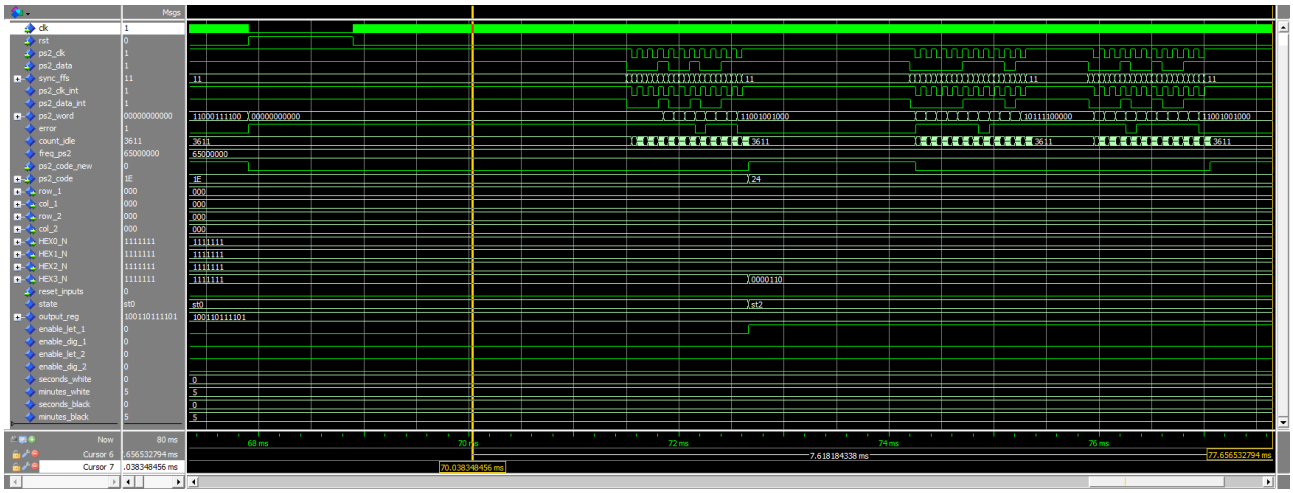
Na slici 3.3.4 vidi se stanje dela sistema za obradu ulaznih signala nakon isteka vremena jednom od igrača. Istek vremena generiše signal *reset_inputs*, kojim se tastatura i blok ULAZ dovode u početno stanje. Signali *ps2_code_new* i *ps2_word* dobijaju vrednosti ‘0’ i “00000000000”, respektivno, i time se ne može desiti da se po resetu sistema učita nepoželjno slovo za pokret bele figure na početku partije. Takođe, svi sedmosegmentni displeji se gase jer mašina stanja ulazi u početno stanje st0. Pošto se signal *reset_inputs* generiše kada istekne vreme jednom od igrača ili ukoliko je neki od kraljeva pojeden (oba signala generišu se sinhorono sa signalom takta), ista promena signala sa slike 3.3.3 se dešava i kada je jedan od kraljeva pojeden.



Slika 3.3.4 Stanje nakon pobede jednog og igrača

Slično kao i sa resetom usled gubitka partije, svi pomenuti signali dobijaju nove vrednosti po sistemskom resetu. Pošto signal sistemskog reseta dolazi asinhrono u odnosu na sistemski takt, a takođe, signal sistemskog reseta *rst_taster* gasi pll koji dovodi signal takta od 65 MHz do svakog bloka sistema, potrebno je asinhrono sa signalom takta resetovati ovaj deo sistema kako se blok ULAZ ne bi našao u neželjenom stanju prilikom reseta.

Sa slike 3.3.5 vidi se stanje signala blokova ULAZ i PS2_KEYBOARD nakon sistemskog reseta (signal rst aktivan sa logičkom jedinicom ekvivalentan signalu *reset_n* sa glavne blok šeme sistema). Signali *ps2_code_new* i *ps2_word* dobijaju vrednosti ‘0’ i “00000000000”, respektivno, i time se ne može desiti da se po resetu sistema učita nepoželjno slovo za pokret bele figure na početku partije. Takođe, svi sedmosegmentni displeji se gase jer mašina stanja ulazi u početno stanje st0. Posle 71. milisekunde vidi se unos tastera „E“ sa tastature (heksadecimalni kod 24h), promena mašine stanja u stanje st2, a zatim, održavanje istog stanja nakon puštanja tastera „E“ sa tastature.



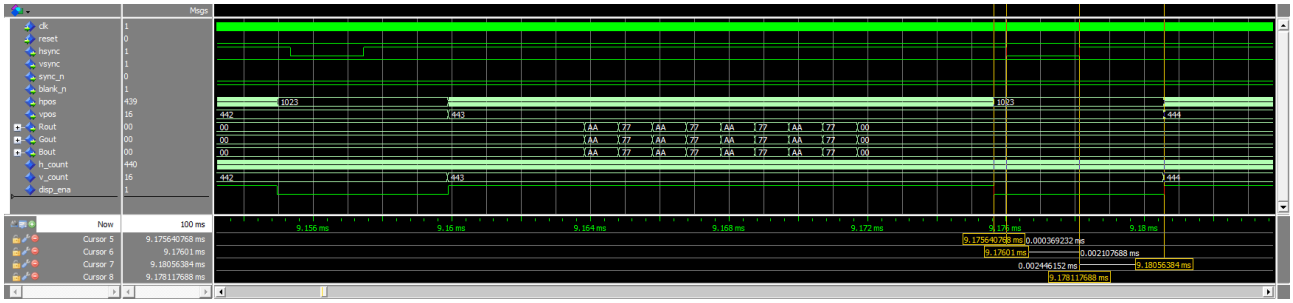
Slika 3.3.5 Stanje nakon sistemskog reseta

3.4. Izlazni signali sistema

U ovom potpoglavlju će se prikazati signali koji kontrolišu VGA displej i prikazaće se vrednosti boja koje se iscrtavaju na displeju u određenim opsezima vremena. Videće se kako logika sistema iscrtava redove piksela, i kako prelazi u sledeću kolonu. Videće se, takođe, ponašanje izlaza usled reseta.

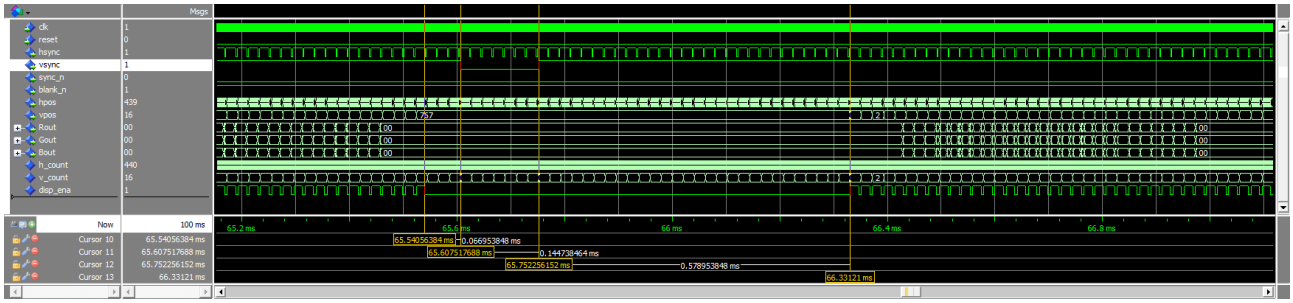
Sa slike 3.4.1 vidi se iscrtavanje jednog reda piksela. Red piksela koji je odabran za prikaz je u redu šahovske table u kom nema figura, i red piksela koji se iscrtava ne prikazuje brojeve redova šahovske table sa leve strane od šahovske table. Pošto su vrednosti boja za polje šahovske table AAh i 77h, u 8 alternativnih iscrtavanja iscrtavaju se prazna polja šahovske table.

Sa 4 žuta kursora (Slika 3.4.1) ograničena su 3 karakteristična vremena u kojima se ne iscrtavaju pikseli, već se signali postavljaju za iscrtavanje sledećeg reda. Između prva dva žuta kursora, prikazano je „*Front Porch"* horizontalno vreme, u kome horizontalni „*deflection coil*" pokazuje na piksele desno i izvan ekrana. Između sledeća dva žuta kursora se prikazuje vreme horizontalne sinhronizacije u kome se horizontalni „*deflection coil*" vraća na početnu levu poziciju, levo od ekrana. U trećem periodu između sledeća dva žuta kursora se prikazuje „*Back Porch"* vreme koje je potrebno da horizontalni „*deflection coil*" pokazuje na prvi piksel ekrana sa leve strane.



Slika 3.4.1 Iscrtavanje jednog reda piksela

Sa slike 3.4.2 Vide se karakteristična vremena tokom prelaska sa iscrtavanja jedne slike ekrana na sledeću sliku ekrana, nakon što je iscrtan poslednji red piksela. Vreme između prva dva žuta kursora karakteriše vertikalni „*Front Porch"*, tokom kog vertikalni „*deflection coil*" pokazuje na piksele koji bi se nalazili ispod ekrana. Između sledeća dva žuta kursora je vreme vertikalne sinhronizacije za koje se vertikalni „*deflection coil*" vraća na poziciju iznad ekrana. Zatim je potrebno „*Front Porch"* vremena prikazano između sledeća dva žuta kursora da bi vertikalni „*deflection coil*" pokazivao na gornji red piksela na ekranu.

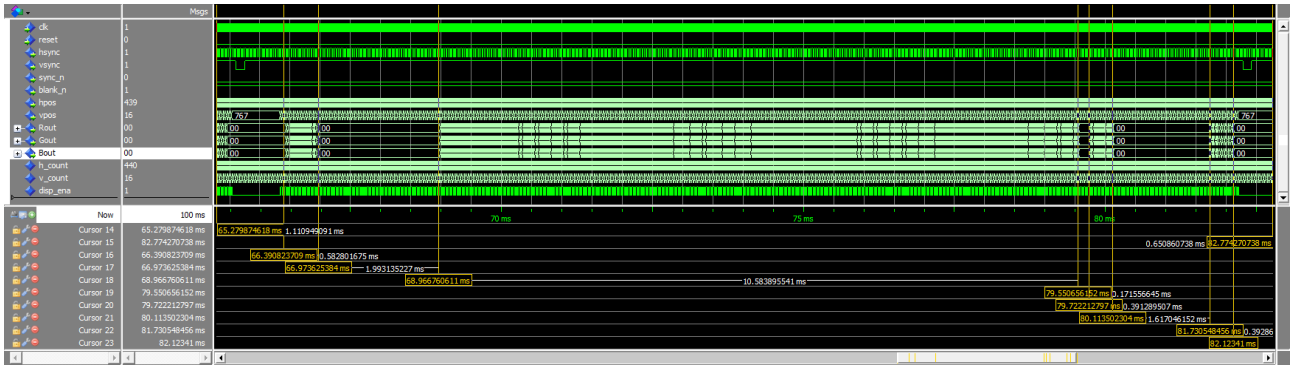


Slika 3.4.2 Signali prilikom iscrtavanja kolone

Sa slike 3.4.3 okvirno se može videti kako se iscrtava jedan frejm slike u vremenu. Korisno je primetiti vremenske periode u kojima se menjaju vrednosti Rout, Gout i Bout, izlazni

signali iz sistema na čipu, koji ulaze u DA konvertor i šalju taj signal monitoru za iscrtavanje. Promene ovih signala govore u kojim vremenskim trenucima nisu svi horizontalni pikseli crni, jer inače bi im vrednost bila 00h.

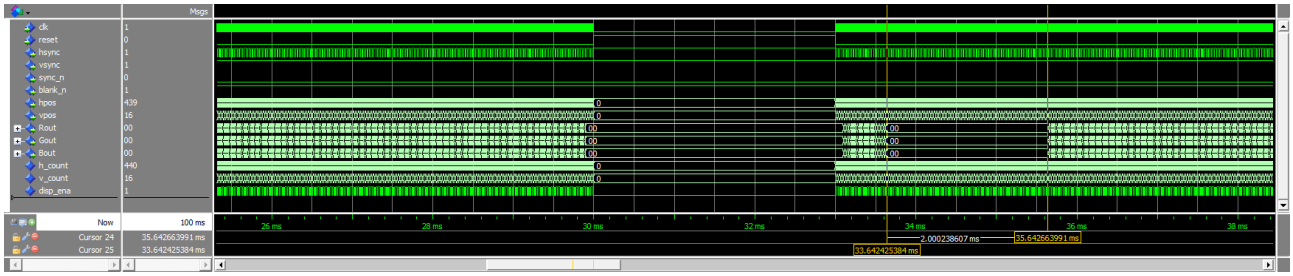
Između prvog para žutih kursora Rout, Gout i Bout se menjaju jer se iscrtavaju redovi piksela u kojima se na ekranu nalaze informacije o tome koji igrač trenutno igra i vreme preostalo crnom igraču. U sledećem periodu između žutih kursora iscrtava se šahovska tabla i brojevi koji obeležavaju redove šahovske table, a između sledećeg para žutih kursora, iscrtavaju se slova koja obeležavaju kolone šahovske table. Između poslednjeg para žutih kursora iscrtava se vreme preostalo belom igraču.



Slika 3.4.3 Iscrtavanje jednog frejma

Sa slike 3.4.4 vidi se ponašanje sistema tokom i nakon reseta. Tokom reseta, koji se u VGA_SYNC blok dovodi sa aktivnom logičkom jedinicom, nestaje takt VGA_SYNC bloka jer se resetuje pll blok koji ga generiše. Signali za horizontalnu i vertikalnu sinhronizaciju se postavljaju na neaktivnu vrednost, logičku jedinicu, i iscrtava se crna boja na ekranu. Svi brojači iz vga_sync.vhd se postavljaju na početne vrednosti i ne dozvoljava se iscrtavanje niskim nivoom signala *disp_ena*.

Nakon reseta počinje iscrtavanje novog frejma od početka, i u periodu između dva žuta kursora vidi se iscrtavanje redova crnih piksela između vertikalnog reda piksela za oznaku igrača koji igra i šahovske table, što nagoveštava da je iscrtavanje ispravno.



Slika 3.4.4 Stanje nakon reseta

4. HARDVER

4.1. Zauzeće čipa

Resource	Usage	%
Logic utilization (ALMs needed / total ALMs on device)	4,524 / 32,070	14 %
ALMs needed [=A-B+C]	4,524	
[A] ALMs used in final placement [=a+b+c+d]	4,696 / 32,070	15 %
[a] ALMs used for LUT logic and registers	245	
[b] ALMs used for LUT logic	4,431	
[c] ALMs used for registers	20	
[d] ALMs used for memory (up to half of total ALMs)	0	
[B] Estimate of ALMs recoverable by dense packing	189 / 32,070	< 1 %
[C] Estimate of ALMs unavailable [=a+b+c+d]	17 / 32,070	< 1 %
[a] Due to location constrained logic	0	
[b] Due to LAB-wide signal conflicts	0	
[c] Due to LAB input limits	17	
[d] Due to virtual I/Os	0	
Difficulty packing design	Low	
Total LABs: partially or completely used	602 / 3,207	19 %
-- Logic LABs	602	
-- Memory LABs (up to half of total LABs)	0	
Combinational ALUT usage for logic	6,243	
-- 7 input functions	235	
-- 6 input functions	2,998	
-- 5 input functions	1,150	
-- 4 input functions	967	
-- <=3 input functions	893	
Combinational ALUT usage for route-throughs	4	
Dedicated logic registers	754	
-- By type:		
-- Primary logic registers	529 / 64,140	< 1 %
-- Secondary logic registers	225 / 64,140	< 1 %
-- By function:		
-- Design implementation registers	529	
-- Routing optimization registers	225	

Resource	Usage	%
Virtual pins	0	
I/O pins	61 / 457	13 %
-- Clock pins	3 / 8	38 %
-- Dedicated input pins	0 / 21	0 %
Hard processor system peripheral utilization		
-- Boot from FPGA	0 / 1 (0 %)	
-- Clock resets	0 / 1 (0 %)	
-- Cross trigger	0 / 1 (0 %)	
-- S2F AXI	0 / 1 (0 %)	
-- F2S AXI	0 / 1 (0 %)	
-- AXI Lightweight	0 / 1 (0 %)	
-- SDRAM	0 / 1 (0 %)	
-- Interrupts	0 / 1 (0 %)	
-- JTAG	0 / 1 (0 %)	
-- Loan I/O	0 / 1 (0 %)	
-- MPU event standby	0 / 1 (0 %)	
-- MPU general purpose	0 / 1 (0 %)	
-- STM event	0 / 1 (0 %)	
-- TPIU trace	0 / 1 (0 %)	
-- DMA	0 / 1 (0 %)	
-- CAN	0 / 2 (0 %)	
-- EMAC	0 / 2 (0 %)	
-- I2C	0 / 4 (0 %)	
-- NAND Flash	0 / 1 (0 %)	
-- QSPI	0 / 1 (0 %)	
-- SDMMC	0 / 1 (0 %)	
-- SPI Master	0 / 2 (0 %)	
-- SPI Slave	0 / 2 (0 %)	
-- UART	0 / 2 (0 %)	
-- USB	0 / 2 (0 %)	
M10K blocks	0 / 397	0 %
Total MLAB memory bits	0	
Total block memory bits	0 / 4,065,280	0 %
Total block memory implementation bits	0 / 4,065,280	0 %
Total DSP Blocks	0 / 87	0 %

Resource	Usage	%
Fractional PLLs	1 / 6	17 %
Global signals	2	
-- Global clocks	2 / 16	13 %
-- Quadrant clocks	0 / 66	0 %
-- Horizontal periphery clocks	0 / 18	0 %
SERDES Transmitters	0 / 100	0 %
SERDES Receivers	0 / 100	0 %
JTAGs	0 / 1	0 %
ASMI blocks	0 / 1	0 %
CRC blocks	0 / 1	0 %
Remote update blocks	0 / 1	0 %
Oscillator blocks	0 / 1	0 %
Impedance control blocks	0 / 4	0 %
Hard Memory Controllers	0 / 2	0 %
Average interconnect usage (total/H/V)	4.9% / 4.9% / 4.6%	
Peak interconnect usage (total/H/V)	50.5% / 51.5% / 47.6%	
Maximum fan-out	914	
Highest non-global fan-out	914	
Total fan-out	34351	
Average fan-out	4.82	

Tabela 4.1.1 Zauzeće čipa nakon kompilacije VHDL koda

4.2. Memorija

Iz analize zauzeća FPGA čipa, može se videti da su sve informacije o slikama za iscrtavanje smeštene u Look-up tabele u vidu ALM-ova. U slušaju fabrikacije ovakvog čipa, razumnije bi bilo smestiti sve podatke u ROM memoriju zbog smanjenja cene prozvodnje. Pošto je memorijska reč većine memorija jedan bajt, 8 podataka o iscrtavanju piksela mogu se smestiti u jednu memorijsku lokaciju. Pri dizajnu ovakvog sistema, dodao bi se jedan prihvatni registar koji će svakih 8 ciklusa sistemskog takta učitati jednu reč iz memorije za iscrtavanje reda od 8 piksela na ekranu. Zato je za učitavanje iz memorije potrebna ROM memorija koja može da radi na učestanosti od 8.125 MHz.

Najveći deo memorije zauzimaju slike figura. Svaka figura zauzima 64x64 bita u memoriji, tojest, 512 bajtova. Postoji 6 različitih figura, i zato, figure zajedno zauzimaju 3072 bajtova ROM memorije. Prazno polje se ne smešta u memoriju jer ne nosi korisne informacije. Konstante „Red“ i „Kolona“ zauzimaju po 128 bajtova u memoriji. Konstante „WhiteTurn“ i „BlackTurn“ zauzimaju po 18x48 bita u memoriji, tojest, po 108 bajtova ROM memorije. Ako se svaki red informacija Konstante „Digits_0to9“ smesti u bajt memorije, konstanta „Digits_0to9“ će zauzeti 70 bajtova memorije, a konstanta „Dvotacka“ bi u tom slučaju zauzela 7 Bajtova memorije. Ovo znači da bi ukupno zauzeće u memoriji za skladištenje slika koje će se iscrtavati na ekranu biti 3621 bajtova, što bi dalje značilo da sva memorija može stati u ROM memoriju kapaciteta 4 kB (12 bita za adrese).

4.3. Celobrojna deljenja

U ovom sistemu potrebna su celobrojna deljenja sa 2, 4 i 10, a potreban je i ostatak deljenja sa 10. Celobrojna deljenja sa 2 ili 4 omogućavaju smanjeno zauzeće memorije za neke od slika u ROM memoriji, a deljenje sa 10 i ostatak deljenja sa 10 dozvoljavaju izvlačenje informacije o cifri broja koja na ekranu prikazuje preostalo vreme nekog od igrača. Da se ovakav hardver ne bi sintetizovao u množačima ili DSP blokovima unutar FPGA čipa, celobrojna deljenja sa 2 ili sa 4 realizuju se preko funkcija ASR i ASR2 koje vrše logičko pomeranje binarnog broja udesno za jedno ili dva mesta, čime se izbegava komplikovaniji hardver za deljenje. Deljenje sa 10 i ostatak deljenja sa 10 realizuje se preko funkcija DIV10 ili REM10, koje preko *look-up* tabela koje mogu imati ulaze između 0 i 59, generišu izlaze između 0 i 5 (za cifre desetica) ili između 0 i 9 (za cifre jedinica).

4.4. Ograničenja

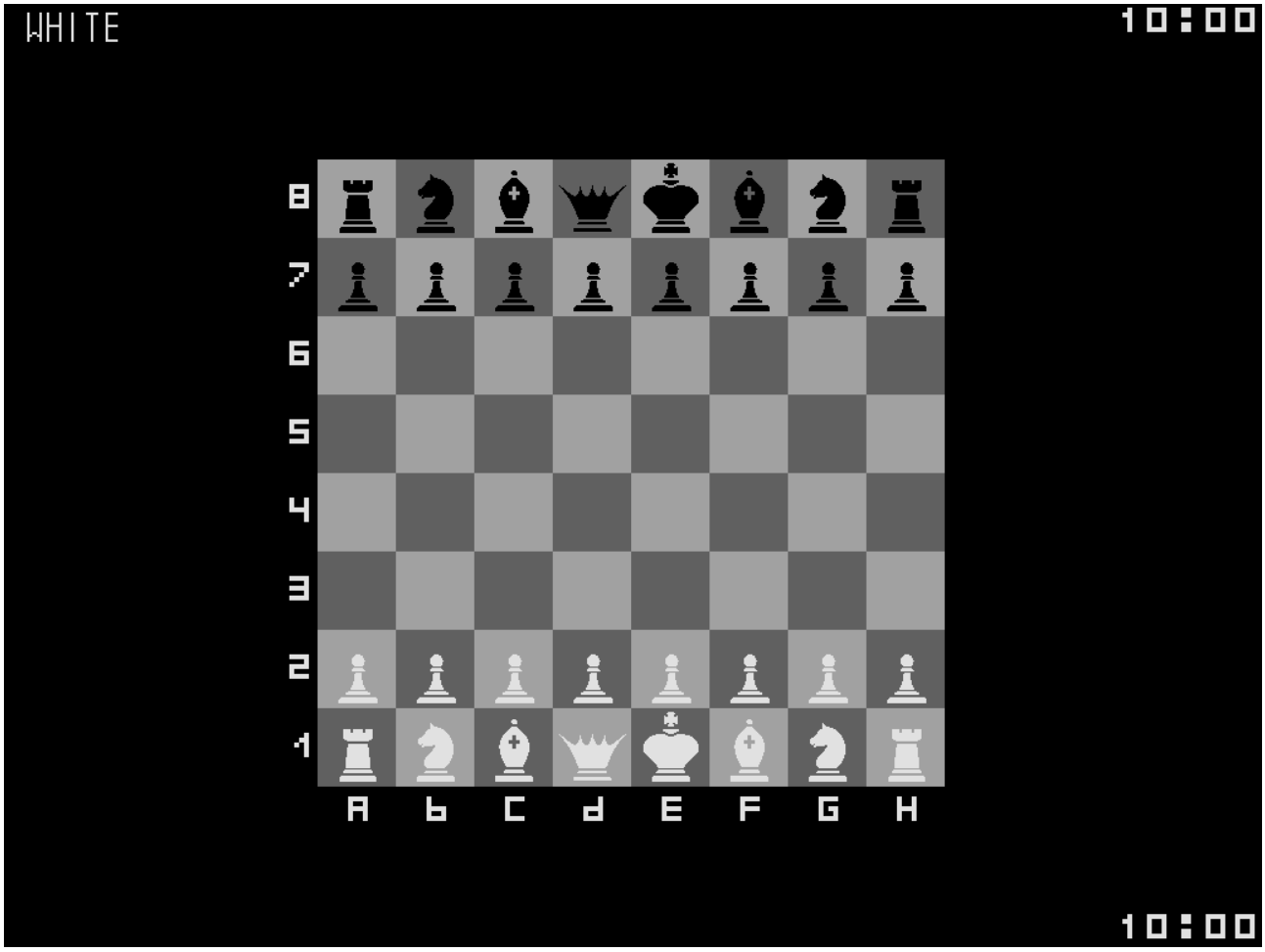
Sistem je projektovan tako da radi na maksimalnoj učestanosti od 78.48 MHz. Ova učestanost ograničava rezoluciju ekrana i broj piksela u sekundi koji može da se prikazuje. Ideja za prikaz na ekranima veće rezolucije bi bila da se smanji *Refresh Rate*, a poveća rezolucija ekrana, pošto brzina iscrtavanja figura na ekranu nije kritična jer je slika ekrana u glavnom statična, osim kada se pomera neka figura.

Na nekim displejima sa manjom rezolucijom ne bi mogla da se prikazuje slika na ekranu na ispravan način, zato što iscrtavanje piksela zavisi od vertikalne i horizontlane pozicije koja se iscrtava. Zato bi sistem morao ponovo da se projektuje za rezoluciju šahovske table 256x256 piksela.

Pošto je preporučeno vreme otklanjanja odskoka signala sa PS/2 tastature 5 μs, teoretski, perioda takta mora biti manja od toga, i zato je (teorijska) najniža učestanost signala sistemskog takta na kojoj bi sistem mogao da radi 200 kHz.

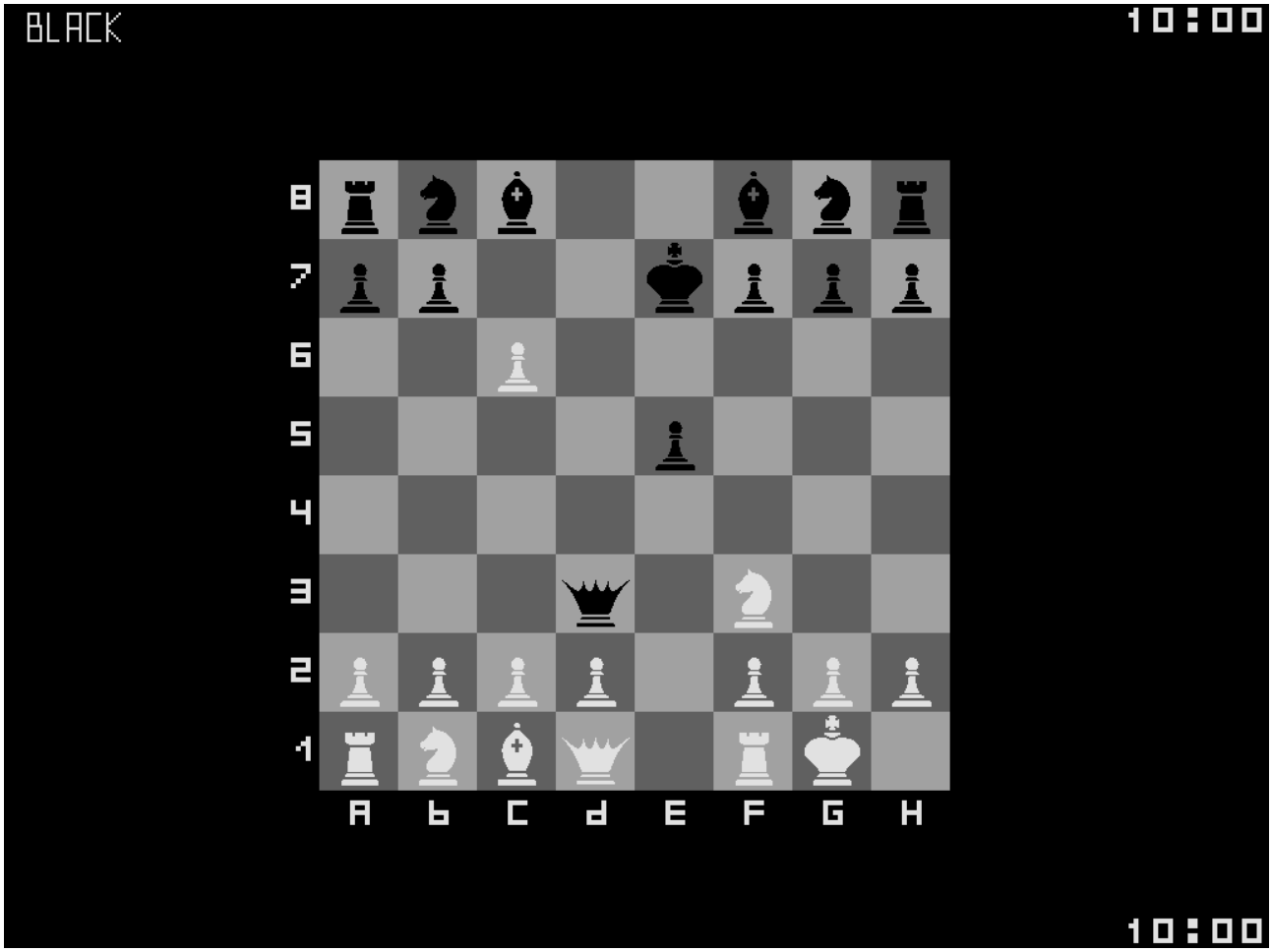
4.5. Konačni izgled igre

Za iscrtavanje slike 4.5.1 korišćen je VGA simulator sa veb-sajta koji se nalazi na linku u literaturi[9]. Iz *Test bench* fajla se putem procesa u .txt fajl upisuju vrednosti trenutnog vremena, hpos, vpos, R, G i B svake uzlazne ivice signala sistemskog takta. VGA simulator na osnovu .txt fajla generiše frejmove na ekranu.



Slika 4.5.1 Izgled ekrana na početku igre

Na slici 4.5.2 vidi se izgled ekrana nakon odigranih poteza E2E4, E7E5, F1D3, D7D5, E4D5, C7C5, D5C6, E8E7, G1F3, D8D3, E1G1. Sledeći igrač je crni. Vreme je još uvek na 10:00 zato što su svi potezi odigrani za manje od 200 ms u RTL simulaciji.



Slika 4.5.2 Izgled ekrana nakon 11 poteza

4.6. Potencijalna poboljšanja

- Na većim ekranima, uz manje promene VHDL koda, šahovska tabla bi se mogla prikazivati na 1024x1024 centralnih piksela ekrana radi veće preglednosti.
- Mogla bi se dodati opcija bilo kog od igrača da se preda pritiskom tastera „P“ na tastaturi. Takođe bi se na ekranu onda ispisivao broj pobeda za jednog ili drugog igrača. Sistemski reset bi vratio brojeve pobeda na 0.
- Bilo bi poprilično teško napraviti hardverski sistem šaha tako da može da prepozna šah mat ili pat situaciju, ali moguće je promeniti postojeći kod tako da ako je neki od kraljeva pod šahom, sledećim potezom se kralj mora izbaviti iz šaha. Ovo bi se ispitivalo kombinacionom logikom koja bi proveravala da li se u dijagonali kralja nalazi protivnikova kraljica ili lovac, da li se na dijagonalnom polju nalazi protivnikov pijun, da li se nalazi protivnikov konj koji ga napada... i ukoliko se nalazi, dozvolio bi se samo potez u kome kralj više nije pod šahom. Pošto više nije moguće pojesti kralja, igra bi se završila predajom igrača koji ne može da odigra validan potez. Na ovaj način, mogu se izbegnuti neke nemoguće situacije na šahovskoj tabli, kao na primer da je jedan kralj pored drugog ili da neki od igrača ignoriše činjenicu da je pod šahom i igra nevalidan potez...

- Mogla bi se dati korisnicima veća kontrola oko toga kakvu igru žele da igraju pomoću prekidača na FPGA pločici. Na primer, lako bi moglo da se podesi da li će šahovska tabla izgledati sivo, ili u nekim nijansama braon, ili možda nekim drugim, dodeljivanjem vrednosti promenljivoj *color* u zavisnosti od prekidača. Takođe bi bilo lako promeniti VHDL kod tako da se prekidačima podešava koliko dugo igra može da traje nakon početka nove partije.
- VGA i PS/2 standard su zastareli standardi koji se koriste sve manje. Poboljšanje bi bio prelazak na USB tastaturu i HDMI ulaz za monitor. Značajna promena za lagodnost igranja bi bila promena interfejsa sa PS/2 tastature na miš.
- Promenom načina rada blokova PS2_KEYBOARD i ULAZ mogu se primati poslednja dva bajta iz tastature, i time se dekodovati da je taster pritisnut ili pušten. Na ovaj način neće dolaziti do pogrešnog unošenja informacija usled puštanja tastera tastature. Ovo bi se postiglo još jednom mašinom stanja unutar bloka ULAZ.

5. ZAKLJUČAK

Hardver sistema je realizovan sintezom koda napisanog u VHDL-u na FPGA čipu CycloneV proizvođača Altera (Intel), na pločici DE1-SoC, korišćenjem Intelovog softvera Quartus Prime 18.1 Light Edition i ModelSim – Intel FPGA Starter Edition 10.5b.

Projekat „Hardverska implementacija igre šah“ zadovoljava funkcionalne i hardverske specifikacije. Na VGA displeju se svake sekunde prikazuje 60 slika, na rezoluciji 1024x768 piksela. Korisniku je prikazana šahovska tabla, vreme svakog og igrača i informacija o tome koji je od igrača na potezu. Preko tastature se unose pokreti figure, koji mogu da se vide na sedmosegmentnim displejima, sa opcijom ponovnog unosa. Tasterom na pločici se sistem dovodi u početno stanje.

„Tajming“ analizom iz softverskih alata potvrđeno je da je učestanost signala takta od 65 MHz u granicama koje zadovoljavaju sva vremena postavljanja i držanja svakog od flop-flopova. Neke od glavnih informacija o zauzeću čipa su: 4526/32070 logičkog iskorišćenja (ALM), 755 registara i 61/457 pinova.

Razumnom količinom testiranja logike hardvera za pomeranje figura je ustanovljeno da se sve figure kreću samo na način koji odgovara funkcionalnim specifikacijama. Svaki od signala se unutar simulacija ponaša predviđeno za različite moguće kombinacije ulaza.

U četvrtom poglavlju navedene su neke od ideja za buduće poboljšanje sistema i navedeni su neki od aspekata za moguću fabrikaciju čipa. Najznačajnije ideje su: rad sa ROM memorijom, potencijalna promena ulaznih interfejsa, mogućnost predaje igrača i prepoznavanje situacije „šah“ od strane hardvera.

LITERATURA

[1] Predavanja i vežbe iz predmeta „Uvod u projektovanje VLSI sitema“, Elektrotehnički fakultet, Univerzitet Beogradu

[2] https://www.ics.uci.edu/~jmoorkan/vhdlref/Synario%20VHDL%20Manual.pdf

[3] http://tnt.etf.bg.ac.rs/~oe4upv/materijali/projekti/DE1-SoC_User_manual_reve.pdf

[4] https://forum.digikey.com/t/ps-2-keyboard-interface-vhdl/12614

[5] https://forum.digikey.com/t/vga-controller-vhdl/12794

[6] http://tnt.etf.bg.ac.rs/~oe4upv/materijali/projekti/DE1-SoC-schematic.pdf

[7] https://xess.com/blog/vga-the-rest-of-the-story/

[8] https://www.alamy.com/black-and-white-magnetic-chess-pieces-silhouettes-image367837888.html

[9] https://www.ericeastwood.com/blog/8/vga-simulator-getting-started

СПИСАК СКРАЋЕНИЦА

ALM	*Adaptive Logic Module*
ALUT	*Adaptive look-up table*
DAC	*Digital-to-analog Converter*
FPGA	*Field Programmable Gate Array*
JTAG	*Joint Test Action Group*
LAB	*Logic Array Block*
LED	*Light Emiting Diode*
PLL	*Phase locked loop*
R, G, B	*Red, Green, Blue*
RTL	*Register Transfer Level*