SVEUČILIŠTE U SPLITU

FAKULTET ELEKTROTEHNIKE, STROJARSTVA I BRODOGRADNJE

PROJEKTIRANJE DIGITALNIH SUSTAVA

PROJEKT

ŠAH

Gabriel Ivančić

Split, 2025.

SADRŽAJ

1. UVOD

Zadatak projekta je realizacija igre šah na Xilinx Spartan-3E pločici u Verilogu. Verilog je jedan od najraširenijih hardverskih opisnih jezika (HDL) koji omogućava modeliranje, simulaciju i sintezu digitalnih sustava. Njegova prednosti je u tome što omogućava opis digitalnih sklopova na visokoj razini apstrakcije, ali i dovoljno detaljno da se mogu sintetizirati u stvarne elektroničke komponente na FPGA pločici. U praksi se koristi za dizajn procesora, memorija, komunikacijskih sučelja i različitih digitalnih logika, a u ovom projektu je poslužio za realizaciju složenije igre i grafičkog prikaza na VGA izlazu.

Za potrebe projekta korištene su tipke na pločici (lijevo, desno, gore, dolje i srednja) kojima se ostvaruje komunikacija s igrom, dok se prikaz šahovske ploče i figura ostvaruje putem VGA izlaza na vanjskom monitoru. Na taj način implementacija igre ne zahtijeva dodatne vanjske uređaje osim pločice i monitora. Cilj projekta bio je povezati teorijsko znanje o digitalnim sustavima i Verilogu sa praktičnom implementacijom na FPGA pločici.

U nastavku izvještaja bit će objašnjena logika igre i način implementacije osnovnih pravila šaha, zatim glavni modul u kojem se nalazi kod same igre, crtanja i VGA prijenosa, kao i ostali moduli potrebni za realizaciju. Na kraju su prikazani rezultati, mogućnosti igrice te prijedlozi za daljnja poboljšanja.

2. ŠAH

U ovom projektu izrađena je verzija igre šaha koja se izvodi na FPGA pločici, a prikazuje na vanjskom monitoru putem VGA izlaza. Naglasak je na preglednom vizualnom prikazu šahovske ploče i figura, intuitivnom upravljanju tipkama te jasnoj izmjeni poteza između dva igrača.

Igra se odvija između dva igrača bijeli i crni (u našem slučaju označen plavom bojom) koji naizmjenično odigravaju poteze. Na početku se prikazuje šahovnica sa početnim rasporedom figura. Potez se sastoji od dva koraka: odabir figure koju igrač želi pomaknuti i odlaganje figure na željeno mjesto. Ako je ciljano polje slobodno figura se premješta na to polje, a ako je ciljano polje zauzeto protivničkom figurom izvodi se uzimanje tj. protivnička figura se uklanja sa ploče, a nova figura zauzima njeno mjesto. Nakon što smo odigrali željeni potez, automatski se prebacuje red na drugog igrača što je vizualno prikazano različitom bojom pokazivača (highlightera), odnosno kada je bijeli na potezu pokazivač je žute boje, a kada je crni (plavi) na potezu pokazivač je crvene boje.

Na slici 1 prikazana je pločica koju smo koristili. Tipke označene različitim bojama služe za sljedeće: desno (žuta), lijevo (plava), gore (ljubičasta), dolje (zelena), uzimanje i ostavljanje figura (crvena), reset šahovnice i figura (crna).

Slika 1.Pločica

3. IMPLEMENTACIJA U VERILOGU

Izradu projekta smo započeli crtanjem šahovnice. Kod na sljedećoj slici radi sljedeće, u svakom piksel-taktu (posedge clk) provjeravamo gdje se trenutačni piksel nalazi (h_counter, v_counter) i prema tome palimo RGB kanale. Na taj način crtamo: debele bijele vanjske okvire oko ploče, okomite bijele linije između stupaca i vodoravne bijele linije između redaka. U svim ostalim pikselima pozadina je crna. Unutarnja ploča je od x=120 do 520 i od y=40 do 440, odnosno točno 400x400 px što znači da su 8x8 polja veličine 50 piksela. Kada je piksel na vanjskom okviru ili okomitoj ili vodoravnoj liniji postavljamo crvena=1, zelena=1, plava=1 i time dobijemo bijele linije, dok u suprotnom sve boje postavljamo na 0 da dobijemo crnu pozadinu.

```
//  CRTANJE ploče
always @(posedge CLK) begin
  // okviri/linije
  if (((h_counter >= 110 && h_counter <= 120) && (v_counter >= 40 && v_counter <= 440)) ||
      ((h_counter >= 520 && h_counter <= 530) && (v_counter >= 40 && v_counter <= 440)) ||
      ((v_counter >= 30 && v_counter <= 40)  && (h_counter >= 110 && h_counter <= 530)) ||
      ((v_counter >= 440 && v_counter <= 450) && (h_counter >= 110 && h_counter <= 530))) begin
    zelena <= 1; plava <= 1; crvena <= 1;
  end
  else if (((h_counter >= 169 && h_counter <= 170) && (v_counter >= 40 && v_counter <= 440)) ||
           ((h_counter >= 219 && h_counter <= 220) && (v_counter >= 40 && v_counter <= 440)) ||
           ((h_counter >= 269 && h_counter <= 270) && (v_counter >= 40 && v_counter <= 440)) ||
           ((h_counter >= 319 && h_counter <= 320) && (v_counter >= 40 && v_counter <= 440)) ||
           ((h_counter >= 369 && h_counter <= 370) && (v_counter >= 40 && v_counter <= 440)) ||
           ((h_counter >= 419 && h_counter <= 420) && (v_counter >= 40 && v_counter <= 440)) ||
           ((h_counter >= 469 && h_counter <= 470) && (v_counter >= 40 && v_counter <= 440))) begin
    zelena <= 1; plava <= 1; crvena <= 1;
  end
  else if (((h_counter >= 120 && h_counter <= 520) && (v_counter >= 89  && v_counter <= 90 )) ||
           ((h_counter >= 120 && h_counter <= 520) && (v_counter >= 139 && v_counter <= 140)) ||
           ((h_counter >= 120 && h_counter <= 520) && (v_counter >= 189 && v_counter <= 190)) ||
           ((h_counter >= 120 && h_counter <= 520) && (v_counter >= 239 && v_counter <= 240)) ||
           ((h_counter >= 120 && h_counter <= 520) && (v_counter >= 289 && v_counter <= 290)) ||
           ((h_counter >= 120 && h_counter <= 520) && (v_counter >= 339 && v_counter <= 340)) ||
           ((h_counter >= 120 && h_counter <= 520) && (v_counter >= 389 && v_counter <= 390))) begin
    zelena <= 1; plava <= 1; crvena <= 1;
  end
  else begin
    crvena <= 0; zelena <= 0; plava <= 0;
  end
```

Slika 2. Crtanje ploče

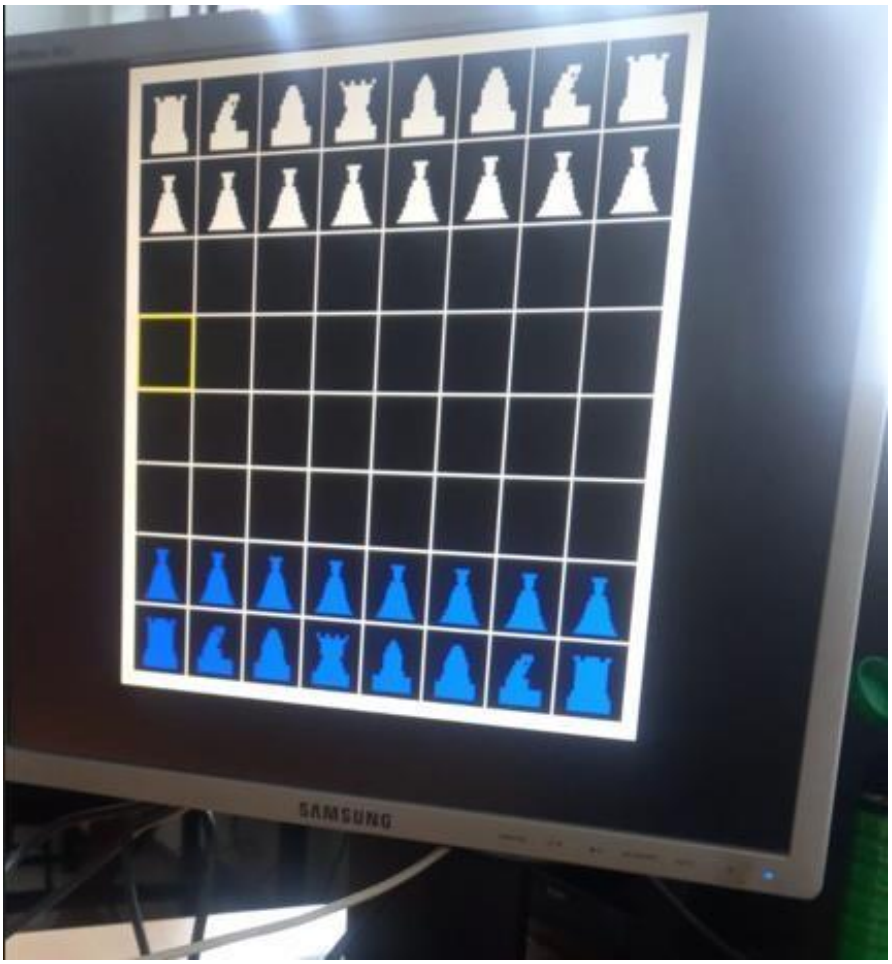
Nakon što smo nacrtali ploču sljedeći korak je bio crtanje figura i dodjela jedinstvenih ID. Na sljedećim slikama je prikazano crtanje samo bijelih pješaka, ali je logika za crtanje svih ostalih figura ista. Za crtanje figura koristimo pristup „siluete po pikselu“. Funkcija pawn_pixel(local_x, local_y) vraća 1 kada je trenutni piksel unutar jedne od zadanih pravokutnih traka koje čine obris pješaka, a inače je 0. Na taj način bez skupnih aritmetičkih operacija dobivamo čistu siluetu pješaka u lokalnim koordinatama polja 50x50 px. Početni raspored se postavlja u reset_board_ids(): bijele figure dolje (ID 0-15), crne gore (ID 16-31) Da bismo uvijek znali koja je figura gdje, svakoj smo dodijelili jedinstveni ID iz raspona od 0 do 31. Za taj ID imamo dva podatka:

- kind[id]- tip figure (od 1 do 6 bijele, 7 do 12 crne, npr. 6-12 su pješaci)
- pos[id]- pozicija na ploči kao indeks polja 0 do 63 (linearno 8*row+col), posebna vrijednost NONE (7h'7f) znači da je figura skinuta sa ploče. Ovo ćemo kasnije koristit za pomicanje figura.

```
// Crtanje figura
function pawn_pixel;   input [9:0] x,y; begin
  pawn_pixel = 1'b0;
  if (y>=10 && y<=12 && x>=20 && x<=30) pawn_pixel = 1'b1;
  else if (y>=13 && y<=18 && x>=22 && x<=28) pawn_pixel = 1'b1;
  else if (y>=19 && y<=20 && x>=23 && x<=27) pawn_pixel = 1'b1;
  else if (y>=21 && y<=24 && x>=20 && x<=30) pawn_pixel = 1'b1;
  else if (y>=25 && y<=28 && x>=18 && x<=32) pawn_pixel = 1'b1;
  else if (y>=29 && y<=32 && x>=17 && x<=33) pawn_pixel = 1'b1;
  else if (y>=33 && y<=36 && x>=16 && x<=34) pawn_pixel = 1'b1;
  else if (y>=37 && y<=40 && x>=14 && x<=36) pawn_pixel = 1'b1;
  else if (y>=41 && y<=43 && x>=12 && x<=38) pawn_pixel = 1'b1;
  else if (y>=44 && y<=45 && x>=10 && x<=40) pawn_pixel = 1'b1;
end endfunction

// Postavljanje indentiteta figurama
function piece_on; input [3:0] code; input [9:0] x,y; begin
  case (code)
    4'd1, 4'd7:  piece_on = king_pixel(x,y);
    4'd2, 4'd8:  piece_on = queen_pixel(x,y);
    4'd3, 4'd9:  piece_on = rook_pixel(x,y);
    4'd4, 4'd10: piece_on = bishop_pixel(x,y);
    4'd5, 4'd11: piece_on = knight_pixel(x,y);
    4'd6, 4'd12: piece_on = pawn_pixel(x,y);
    default:     piece_on = 1'b0;
  endcase
end endfunction

task reset_board_ids;
integer p;
begin
  // BIJELI DOLJE
  pos[0]  <= 56; kind[0]  <= 4'd3;   // rook
  pos[1]  <= 57; kind[1]  <= 4'd5;   // knight
  pos[2]  <= 58; kind[2]  <= 4'd4;   // bishop
  pos[3]  <= 59; kind[3]  <= 4'd2;   // queen
  pos[4]  <= 60; kind[4]  <= 4'd1;   // king
  pos[5]  <= 61; kind[5]  <= 4'd4;   // bishop
  pos[6]  <= 62; kind[6]  <= 4'd5;   // knight
  pos[7]  <= 63; kind[7]  <= 4'd3;   // rook
  // bijeli pijuni (ID 8..15)
  for (p=0; p<8; p=p+1) begin
    pos[8+p]  <= 6'd48 + p[5:0];
    kind[8+p] <= 4'd6;
  end
```

Slika 3. Crtanje figura i dodjela ID

Na sljedećoj slici je prikazan konačni rezultat.



Slika 4. Konačni izgled

Sljedeći korak je bilo crtanje pokazivača (okvir žute boje na slici 4) kojim ćemo šetati po poljima i koji nam je služio za realizaciju logike uzimanja i ostavljanja figura. Pokazivač (highlighter) po ploči opisujemo koordinatama pokazivac_x i pokazivac_y u rasponu 0-7 koje označavaju stupac i red šahovnice. Iz tih koordinata izvodimo granice pravokutnika sktivnog polja. Ploča je usidrena u točki (120,40), svako polje ima 50x50 px, pa pravokutnik pokazivača uvijek „upadne“ točno na odabrano polje. Kretanje pokazivača je tempirano brojačem slow_cnt u domeni clk25. Kada brojač dosegne zadanu vrijednost (≈ 100 ms) generira se impuls tick koji dopušta točno jedan pomak pokazivača ovisno o tome koju smo tipku pritisnuli tj. gdje želimo ići sa njim. Time se dobiva ugodan ritam kretanja bez preskakanja više polja na jedan pritisak tipke. Uz to se koriste registri btnL_q, btnR_q, btnU_q, btnD_q za pamćenje prethodnog stanja tipki što smanjuje mogućnost nenamjernog dvostrukog pomaka unutar istog ticka. Ovakva organizacija omogućuje da je pokazivač uvijek poravnat sa mrežom ploče, pomiče se u pravilnim sporim koracima i ostaje unutar granica 8x8 polja.

```
// logika za Highlighter
 wire [9:0] kocka_x1 = 120 + pokazivac_x * 50;
 wire [9:0] kocka_x2 = kocka_x1 + 49;
 wire [9:0] kocka_y1 =  40 + pokazivac_y * 50;
 wire [9:0] kocka_y2 = kocka_y1 + 49;


// Logika samog pokazivaca , uzimanje ostavljanje figure te reset
      always @(posedge clk25) begin
  // usporavanje highlightera
  if (slow_cnt == 22'd2_499_999) slow_cnt <= 0;
  else                           slow_cnt <= slow_cnt + 1;



  if (reset) begin
    pokazivac_x <= 0; pokazivac_y <= 0;
    btnL_q <= 0; btnR_q <= 0; btnU_q <= 0; btnD_q <= 0;
    holding <= 1'b0; sel_id <= 6'h3F;
    reset_board_ids();
              turn_white <= 1'b1;
  end else begin
    // kretanje pokazivaca na klik
    if (tick) begin
      if (BTN_L && (!btnL_q || BTN_L) && pokazivac_x > 0) pokazivac_x <= pokazivac_x - 1;
      else if (BTN_R && (!btnR_q || BTN_R) && pokazivac_x < 7) pokazivac_x <= pokazivac_x + 1;
      else if (BTN_SR && (!btnD_q || BTN_SR) && pokazivac_y < 7) pokazivac_y <= pokazivac_y + 1; // dolje
      else if (ROT_BT && (!btnU_q || ROT_BT) && pokazivac_y > 0) pokazivac_y <= pokazivac_y - 1; // gore
      btnL_q <= BTN_L; btnR_q <= BTN_R; btnU_q <= ROT_BT; btnD_q <= BTN_SR;
    end
```

Slika 5. Pokazivač

Da bi omogućili igru dva igrača napravili smo to da se boja pokazivača mijenja u ovisnosti koji je igrač na potezu, pa je tako pokazivač žute boje ako je bijeli na potezu, a crvene boje ako je crni na potezu. Također napravljeno je ograničenje da kada je bijeli na potezu odnosno kada je pokazivač žute boje smijemo igrati samo sa bijelim figurama (crne ne možemo uzimati) i isto tako vrijedi kada je crni na potezu da možemo igrati samo sa crnim figurama.

U nastavku ćemo opisati cijelu logiku igre uzimanje figure i ostavljanje figure na željeno mjesto.

1. Uzimanje figure
    - na klik dohvaćamo ID figure na aktivnom polju tmp_id=find_piece_at_idx(cur_idx).
    - ako je polje prazno (tmp_id==6'h3F) nema akcije tj. uzimanja figure
    - ako figura postoji provjerava se boja i koji je igrač na potezu
      -bijeli na potezu: kind[tmp_id] ≤ 6
      -crni na potezu: kind[tmp_id] ≥ 7
    - ako je boja ispravna za aktivnog igrača spremamo je u selektiranu sel_id<=tmp_id, ulazimo u holding: holding<=1'b1 i maknemo je sa ploče pos[tmp_id]<=NONE. Ovo je ključno jer kada uzmemo figuru ona se briše sa tog polja sa kojeg smo je uzeli tako da sigurno nećemo na dva mjesta imati istu figuru.

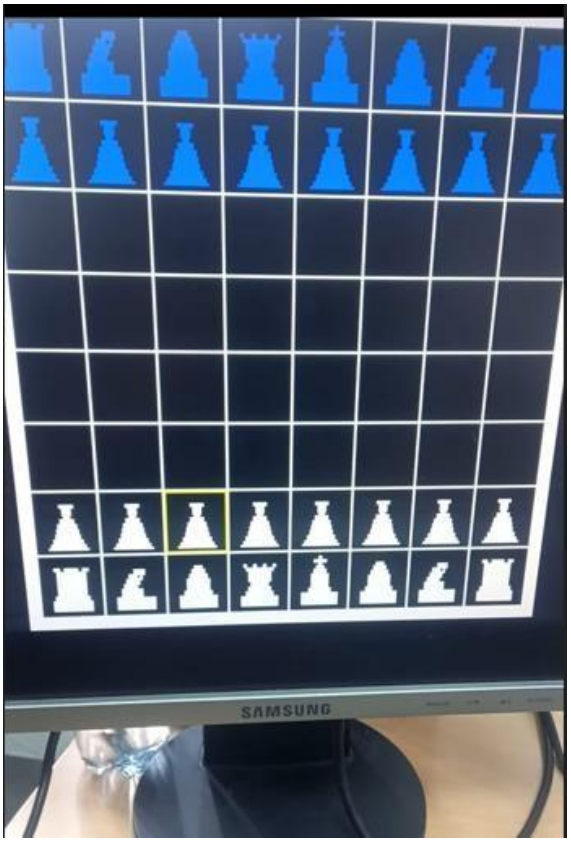
2. Odlaganje figure
    - na klik ponovno gledamo što je na polju na koje želimo igrati tmp_id=find_piece_at_idx(cur_idx)
    - prazno polje (tmp_id==6'h3F):
      -postavimo našu figuru na cilj: pos[sel_id]<={1'b0, cur_idx}
      -izađemo iz holding-a: holding<=1'b0, sel_id<=6'h3F
      -promjena poteza: turn_white<= ~turn_white
    - zauzeto polje:
      -provjera je li tamo protivnik: (kind[sel_id] ≤ 6) != (kind[tmp_id] ≤ 6)
      -ako je suparnik capture: pos[tmp_id]<=NONE, zatim odloži našu figuru na cilj, izađi iz holding-a i promijeni potez
      -ako je naša boja na cilju ostajemo u holding-u i tražimo drugo polje

Na sljedećoj slici je prikazana logika.

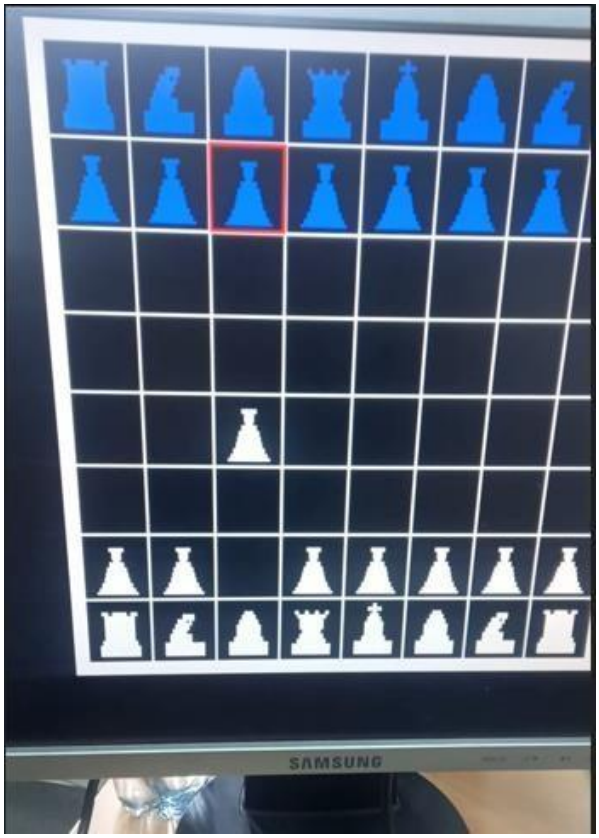
```
//PICK & PLACE po smjenama (bijeli - crni)
// Pravila:
// - kad je turn_white=1, smiješ uzeti/dirati SAMO bijelu figuru (kind<=6)
// - kad je turn_white=0, smiješ uzeti/dirati SAMO crnu/plavu figuru (kind>=7)
// - spuštanje: prazno polje OK; suparnik OK (capture); vlastita boja NIJE OK (ostaješ u "holding")
if (TAKE_p) begin
  if (!holding) begin
    // pokušaj UZETI figuru na trenutnom polju
    tmp_id = find_piece_at_idx(cur_idx);
    if (tmp_id != 6'h3F) begin
      // provjeri boju figure i smjenu
      if ( (turn_white && (kind[tmp_id] <= 4'd6)) ||
           (!turn_white && (kind[tmp_id] >= 4'd7)) ) begin
        // smiješ uzeti
        sel_id  <= tmp_id;
        holding <= 1'b1;
        pos[tmp_id] <= NONE;           // makni s ploče dok je "u ruci"
      end
      // inače: nije tvoja boja › ignoriraj klik
    end
    // ako je prazno polje › ignoriraj klik
  end else begin
    // držimo figuru: pokušaj SPUSTITI
    tmp_id = find_piece_at_idx(cur_idx);  // tko stoji na ciljnom polju?

    if (tmp_id == 6'h3F) begin
      // prazno polje › spusti
      pos[sel_id] <= {1'b0, cur_idx};
      holding     <= 1'b0;
      sel_id      <= 6'h3F;
      turn_white  <= ~turn_white;        // potez gotov › promijeni smjenu
    end else begin
      // ciljno polje zauzeto
      // provjeri boju figure na ciljnom polju u odnosu na našu (kind[sel_id])
      if ( (kind[sel_id] <= 4'd6) != (kind[tmp_id] <= 4'd6) ) begin
        // suparnik › CAPTURE
        pos[tmp_id]   <= NONE;
        pos[sel_id]   <= {1'b0, cur_idx};
        holding       <= 1'b0;
        sel_id        <= 6'h3F;
        turn_white    <= ~turn_white;    // potez gotov › smjena
      end
```

Slika 6. Logika uzimanja i ostavljanja figura

Konačno na slikama 7 i 8 vidimo rezultate. Dakle, na slici 7 pokazivač je žute boje što znači da je red bijelog za igrati, a na slici 8 vidimo da je bijeli odigrao, pomjerio se za dva mjesta naprijed, i pokazivač je promijenio boju u crvenu što znači da je sada crni na potezu.



Slika 7. Bijeli na potezu



Slika 8. Crni na potezu

Za kraj smo napravili prikaz vremena za odigravanje poteza. Koristimo dvije vodoravne linije koje se spuštaju niz ekran: žuta za bijelog (lijevo od ploče) i crvena za crnog (desno od ploče). Svaka linija nacrta se na visini vrha ploče kao jedan red piksela. Tako korisnik stalno vidi tko je na potezu i koliko je potrošenog vremena vizualno akumulirao. Pokretanje je vezano uz potez igrača. Žuta linija ne kreće odmah nego tek kada bijeli prvi put uzme figuru. U svakom trenutku ide samo linija aktivnog igrača (run_white_line/run_black_line). Kretanje je tempirano djeliteljem koji generira line_tick: na svaki line_tick aktivnoj liniji povećamo line_y_* za 1 px (dok ne dosegne dno ploče na 399). Brzina je određena preko LINE_TICKS. Pri svakoj promjeni poteza liniju onoga koji je upravo odigrao podignemo prema vrhu za 5 px (parametar BOOST_PIXELS), čime se vizualno naglašava uredno zaključivanje poteza i dobiva jasniji ritam igre. Kada bilo koja linija dosegne dno ploče (line_y_*=399) igra završava i više se ne prihvaćaju potezi. Ovakav dizajn je lako podesiv LINE_TICKS određuje tempo spuštanja, a BOOST_PIXELS iznos podizanja nakon odigranog poteza.

```
// --- Kretanje linija (spušta se aktivna; resetira se na promjenu poteza) ---
always @(posedge clk25 or posedge reset) begin
  if (reset) begin
    line_y_white       <= 0;
    line_y_black       <= 0;
    run_white_line     <= 1'b0;
    run_black_line     <= 1'b0;
    first_line_started <= 1'b0;
    turn_white_q       <= 1'b1;
  end else begin

  if (sw3_pulse) begin
  line_y_white <= 9'd0;
  line_y_black <= 9'd0;
  end


  // 1) detekcija promjene smjene (rub turn_white)
if (turn_white != turn_white_q) begin
  // upravo je odigrao onaj koji je bio na potezu (turn_white_q)
  if (turn_white_q) begin
    // bijeli je odigrao: umjesto full reseta, digni žutu liniju za +5 s (saturiraj na 0)
    if (line_y_white > BOOST_PIXELS) line_y_white <= line_y_white - BOOST_PIXELS;
    else                             line_y_white <= 9'd0;

    run_white_line <= 1'b0;   // bijeli staje
    run_black_line <= 1'b1;   // crni kreće
  end else begin
    // crni je odigrao: digni crvenu liniju za +5 s (saturiraj na 0)
    if (line_y_black > BOOST_PIXELS) line_y_black <= line_y_black - BOOST_PIXELS;
    else                             line_y_black <= 9'd0;

    run_black_line <= 1'b0;   // crni staje
    run_white_line <= 1'b1;   // bijeli kreće
  end
  turn_white_q <= turn_white;
end


    // 2) prvi start: žuta krene tek kad bijeli PRVI put uzme figuru
    if (!first_line_started && TAKE_p && turn_white && !holding) begin
      run_white_line     <= 1'b1;
      first_line_started <= 1'b0; // ostaje 0? › želimo da se više ne gleda ovaj uvjet
      first_line_started <= 1'b1; // (dvije linije su ok; efekt je "postavi na 1")
    end

    // 3) pomak po ticku (clamp do dna ploče)
    if (line_tick) begin
      if (run_white_line && line_y_white < 9'd399) line_y_white <= line_y_white + 1;
      if (run_black_line && line_y_black < 9'd399) line_y_black <= line_y_black + 1;
    end
  end
end
```

Slika 9. Vrijeme za odigrati

Da bismo sve figure i vrijeme za odigravanje vratili u početno stanje za reset koristimo tipku SW3.

```
      // reset ploče
if (SW3) begin
  reset_board_ids();
  holding   <= 1'b0;
  sel_id    <= 6'h3F;
  turn_white <= 1'b1;


end

// --- Kretanje linija (spušta se aktivna; resetira se na promjenu poteza) ---
always @(posedge clk25 or posedge reset) begin
  if (reset) begin
    line_y_white       <= 0;
    line_y_black       <= 0;
    run_white_line     <= 1'b0;
    run_black_line     <= 1'b0;
    first_line_started <= 1'b0;
    turn_white_q       <= 1'b1;
  end else begin

  if (sw3_pulse) begin
  line_y_white <= 9'd0;
  line_y_black <= 9'd0;
  end
```

Slika 10. Reset ploče i vremena za igranje

UCF

Naš kod iz Veriloga povezujemo sa pinovima na pločici preko UCF datoteke.

```
NET "CLK" LOC = "C9" | IOSTANDARD = LVCMOS33 ;

NET "VGA_R" LOC = "H14" | IOSTANDARD = LVTTL | DRIVE = 8 | SLEW = FAST ;
NET "VGA_G" LOC = "H15" | IOSTANDARD = LVTTL | DRIVE = 8 | SLEW = FAST ;
NET "VGA_B" LOC = "G15" | IOSTANDARD = LVTTL | DRIVE = 8 | SLEW = FAST ;
NET "VGA_H" LOC = "F15" | IOSTANDARD = LVTTL | DRIVE = 8 | SLEW = FAST ;
NET "VGA_V" LOC = "F14" | IOSTANDARD = LVTTL | DRIVE = 8 | SLEW = FAST ;

NET "BTN_R" LOC = "H13" | IOSTANDARD = LVTTL | PULLDOWN ;
NET "BTN_RST" LOC = "V4" | IOSTANDARD = LVTTL | PULLDOWN ;
NET "BTN_L" LOC = "D18" | IOSTANDARD = LVTTL | PULLDOWN ;
NET "BTN_SR" LOC = "K17" | IOSTANDARD = LVTTL | PULLDOWN ;
NET "ROT_BT" LOC = "V16" | IOSTANDARD = LVTTL | PULLDOWN ;
NET "SW3" LOC = "N17" | IOSTANDARD = LVTTL | PULLDOWN ;
```

4. PROBLEMI U RADU I OPCIJE POBOLJŠANJA

Pri izradi smo nailazili na određene probleme, jedan od njih nam je bio da je pokazivač „bježao“ u rubove ploče, odnosno kada bi pritisnuli tipku za ići desno on bi „preletio“ sva polja i došao u zadnje pa kada bi pritisnuli za ići dolje isto bi napravio. Taj problem smo riješili tako što smo uveli usporavanje koraka (~100 ms tick) i pamćenje prethodnog stanja tipki, pa se pokazivač pomiče točno za jedno polje. Drugi problem nam je bio oko uzimanja i ostavljanja figura- u početku smo mislili da ih moramo povezati sa poljima, a ispravno rješenje je bilo razdvojiti stanje od prikaza: svaka figura ima ID, kind, pos; kad je figura uzeta postavimo pos=NONE, pa se crtanje samo prilagodi stanju, a pri ostavljanju figure ažuriramo pos ciljnog polja.

Ideja za napredak projekta bila bi dodati oznake polja uz rub (A-H,1-8), isticanje legalnih poteza za odabranu figuru, implementirati cjelovita šahovska pravila: provjeru legalnosti kretanja po tipu figure, blokadu kroz figure, šah/mat, rokadu, en passant i promociju pješaka.

5. ZAKLJUČAK

U ovom projektu uspješno je realizirana igra šaha na FPGA pločici Xilinx Spartan-3E. Ostvaren je stabilan VGA prikaz šahovnice i figura (640x480 @ 60 Hz), intuitivno upravljanje tipkama (kretanje pokazivača, odabir figura i ostavljanje figura), jasna izmjena poteza između dva igrača te vizualna indikacija vremena putem žute i crvene linije. Logika uzmi/ostavi provedena je razdvajanjem stanja od prikaza (svaka figura ima vlastiti ID, tip i poziciju) čime se crtanje automatski prilagođava igri bez dvostrukih prikaza.

Cilj projekta je bio demonstrirati temeljnu funkcionalnost šaha na FPGA uz vlastiti grafički prikaz. Sustav je pregleda, stabilan za demonstraciju i modularno organiziran što olakšava daljnji razvoj. Kao prirodna nadogradnja predviđeno je dodavanje oznaka polja i potpune provjere pravila šaha. Time bi se postojeća osnova nadogradila u funkcionalno u korisnički potpuno iskustvo igranja.