

Univerzitet u Tuzli
Fakultet elektrotehnike
Usmjerenje: Automatika i robotika

Conway's Game of Life implementacija na FPGA i uporedba sa implementacijom u C na Raspberry Pi-u

Autor: Ajdin Nakčević
Predmet: Projektovanje sistema na čipu
Nastavnik: vanr.prof.dr.Lejla Banjanović-Mehmedović
Asistent: mr.sc. Ivan Bosankić
Akademska 2018 / 2019 godina

Tuzla, Februar 2019. godine

Sadržaj

Sadržaj.....	2
Conway's Game of Life.....	3
Pravila.....	3
Porijeklo.....	3
Primjeri uzoraka.....	5
VGA.....	8
Kako radi VGA.....	8
Horizontal Pixel Timings.....	9
Vertical Line Timings.....	9
VGA_Param.h.....	9
VGA sinhronizacija.....	10
VGA Audio PLL.....	13
C IMPLEMENTACIJA.....	23
Kod.....	23
Usporedba: Raspberry Pi vs. Thinkpad t420.....	25
Raspberry Pi 3 model B.....	25
Lenovo Thnikpad t420.....	26
MATLAB implementacija.....	27
FPGA Verilog implementacija.....	29
Pocetno stanje.....	29
Populacija.....	32
Clock.....	35
MODUL GLAVNE FUNKCIJE.....	36
GAME OF LIFE FUNKCIJA / MODUL.....	39
Vrijeme potrebno za generisanje 1000 generacija na FPGA.....	46
Altera DE2 Cyclone II.....	47
PINOUT.....	48
Cijena.....	49
Reference.....	51

Conway's Game of Life

The Game of Life , također poznat jednostavno kao život , je celularni automat je osmislio britanski matematičar John Horton Conway u 1970.

Igra je igra sa nula igrača , što znači da je njena evolucija određena njegovim početnim stanjem, što ne zahtijeva daljnje unošenje. U interakciji sa igrom života stvara se početna konfiguracija i posmatra kako se razvija, ili, za napredne igrače, stvarajući obrasce sa određenim svojstvima.

Pravila

Univerzum igre života je beskonačna, dvodimenzionalna ortogonalna mreža kvadratnih ćelija , od kojih je svaka u jednom od dva moguća stanja, živa ili mrtva , (ili naseljena i nenaseljena). Svaka ćelija stupa u interakciju sa svojih osam susjeda , koje su ćelije koje su horizontalno, vertikalno ili dijagonalno susjedne. Na svakom koraku se pojavljuju sljedeći prijelazi:

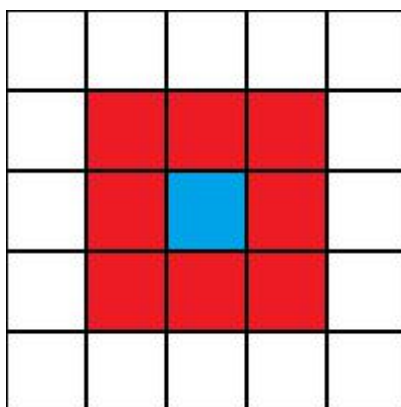
Svaka živa ćelija sa manje od dva živa susjeda umire, kao da je nedovoljna.

Svaka živa ćelija sa dva ili tri živa susjeda živi od sljedeće generacije.

Svaka živa ćelija s više od tri živa susjeda umire, kao da je prenapučena.

Svaka mrtva ćelija sa tačno tri žive komšije postaje živa ćelija, kao da se reprodukuje.

Početni uzorak predstavlja sjeme sistema. Prva generacija je stvorena primjenom gore navedenih pravila istovremeno na svaku ćeliju u sjemenu; rođenja i smrti se javljaju istovremeno, a diskretni trenutak u kojem se to dešava ponekad se naziva krpelj . Svaka generacija je čista funkcija prethodne generacije. Pravila se i dalje ponavljaju kako bi se stvorile nove generacije.



Slika 1. Centralna ćelija sa svojih 8 susjeda

Porijeklo

Krajem 1940. godine John von Neumann je definisao život kao stvaranje (kao biće ili organizam) koji se može reproducirati i simulirati Turingovu mašinu . Von Neumann

je razmišljao o inženjerskom rješenju koje bi koristilo elektromagnetske komponente koje su slučajno plutale u tekućini ili plinu. Ovo se pokazalo da nije realno s tehnologijom koja je tada bila na raspolaganju. Tako je, genijalno, Stanisław Ulam izumio ćelijske automate, koje su imale za cilj da simuliraju teoretske elektromagnetske konstrukcije von Neumanna. Ulam je diskutovao koristeći kompjutere da simulira svoje ćelijske automate u dvodimenzionalnoj rešetki u nekoliko radova. Paralelno, Von Neumann je pokušao da izgradi Ulamov ćelijski automat. Iako uspješan, bio je zaokupljen drugim projektima i ostavio neke detalje nedovršenim. Njegova konstrukcija je bila komplikovana jer je pokušala da simulira sopstveni inženjering. Vremenom su jednostavniji životni objekti bili obezbeđeni od strane drugih istraživača i objavljivani u radovima i knjigama.

Motiviran pitanjima u matematičkoj logici i djelimično od strane Ulama, simulacije igara, John Conway je 1968. godine počeo raditi eksperimente sa različitim pravilima 2D staničnog automata. [3] Konvejov početni cilj bio je da definiira zanimljiv i nepredvidiv ćelijski automat. Prema tome, on je želio da neke konfiguracije traju dugo vremena prije umiranja, druge konfiguracije zauvijek ne dozvoljavaju cikluse, itd. To je bio značajan izazov i otvoren problem godinama prije nego što su stručnjaci za ćelijske automate uspjeli dokazati da, Konvejova igra života priznala je konfiguraciju koja je bila živa u smislu zadovoljenja dva generalna zahtjeva Von Neumanna. Dok su definicije pre Conwayevog životabili su orijentisani na dokaz, Konvejova konstrukcija imala je za cilj jednostavnost bez a priori dokazivanja da je automat živ.

Conway je pažljivo odabrao svoja pravila, nakon značajnih eksperimenata, kako bi ispunio ove kriterije:

Ne bi trebalo biti eksplozivnog rasta.

Treba postojati mali početni obrazac sa haotičnim, nepredvidivim ishodima.

Trebalo bi postojati potencijal za von Neumann univerzalne konstruktore .

Pravila bi trebala biti što jednostavnija, uz poštivanje gore navedenih ograničenja. [4] Igra je napravio svoj prvi javni nastup u pitanju Oktobar 1970 Scientific American , u Martin Gardner " 's Matematička Igre koloni". Teoretski, Conwayev život ima moć univerzalne Turingove mašine : sve što se može izračunati algoritamski može se izračunati u okviru Life .Gardner je napisao: "Zbog analogija života s porastom, padom i izmjenama društva živih organizama, on pripada rastućoj klasi onoga što se naziva" simulacijske igre "(igre koje sliče stvarnim). životni procesi). " [8]

Od njegovog objavljivanja, Konvejova igra života privukla je veliko interesovanje, zbog iznenađujućih načina na koje se obrasci mogu razvijati. Život je primjer nastanka i samoorganizacije . Znanstvenici iz različitih oblasti, kao što su informatika , fizika , biologija , biokemija , ekonomija , matematika , filozofija i generativne nauke , koristili su način na koji kompleksni obrasci mogu proizaći iz implementacije jednostavnih pravila igre. [potreban citat]Igra može poslužiti i kao didaktička analogija , korištena za prenošenje donekle protu-intuitivne ideje da se dizajn i organizacija mogu spontano pojaviti u odsustvu dizajnera. Na primer, kognitivni naučnik Daniel Dennett je koristio analogiju Conwayevog "univerzuma" da bi ilustrovao moguću evoluciju kompleksnih filozofskih konstrukata, kao što su svest i slobodna volja , iz relativno jednostavnog skupa determinističkih fizičkih zakona, koji bi mogli da upravljaju našim univerzumom .

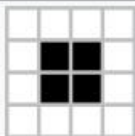
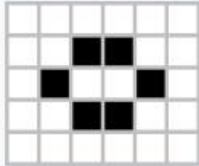
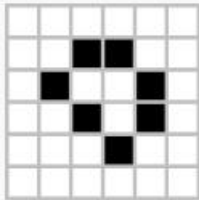
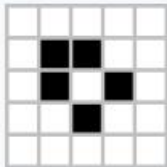
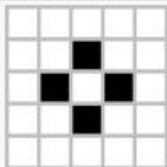
Popularnost Conwayeve igre života pomogla je njenom pojavljivanju na vrijeme za novu generaciju jeftinog pristupa kompjuteru koji je pušten na tržište. Igra se može satima izvoditi na ovim mašinama, koje bi inače ostale neiskorišćene noću. U tom smislu, on je nagovijestio kasniju popularnost računalno generisanih fraktala. Za mnoge, Život je jednostavno bio izazov za programiranje: zabavan način za korištenje inače izgubljenih procesorskih ciklusa. Za neke, međutim, životimao je više filozofskih konotacija. Razvio je kult nakon sedamdesetih i kasnije; sadašnji razvoj je otišao toliko daleko da je stvorio teoretske emulacije kompjuterskih sistema unutar granica Life ploče.

Primjeri uzoraka

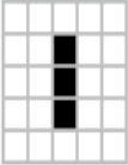
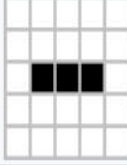
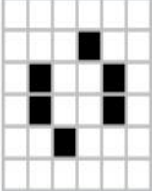
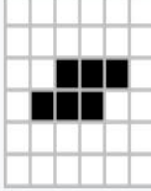
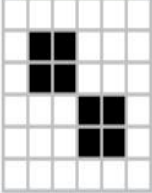
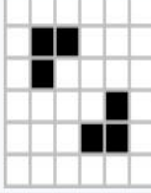
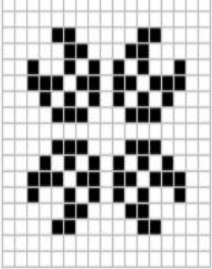
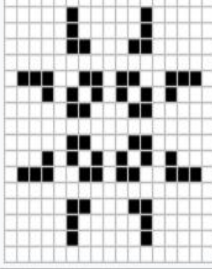
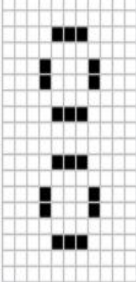
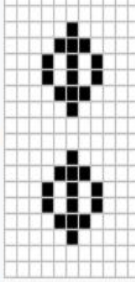
Mnogo različitih tipova obrazaca se pojavljuje u igri života, koja su klasificirana prema njihovom ponašanju. Uobičajeni tipovi uzoraka su: mrtve prirode, koje se ne mijenjaju iz jedne generacije u drugu; oscilatori koji se nakon konačnog broja generacija vraćaju u svoje početno stanje; i svemirskih brodova, koji se prenose preko mreže.

Najraniji zanimljivi obrasci u igri života otkriveni su bez upotrebe kompjutera. Najjednostavnije mrtve prirode i oscilatori otkriveni su dok su pratili sudbine raznih malih početnih konfiguracija koristeći graf papir, crne table i fizičke ploče za igru, kao što su one korištene u Go. Tokom ovog ranog istraživanja, Conway je otkrio da se R-pentomino nije uspio stabilizirati u malom broju generacija. U stvari, potrebno je 1103 generacije da se stabilizuje, do kada ima populaciju od 116 i generisalo je šest jedrilica koje su pobile; [14] ovo su bili prvi svemirski brodovi ikad otkriveni. [15]

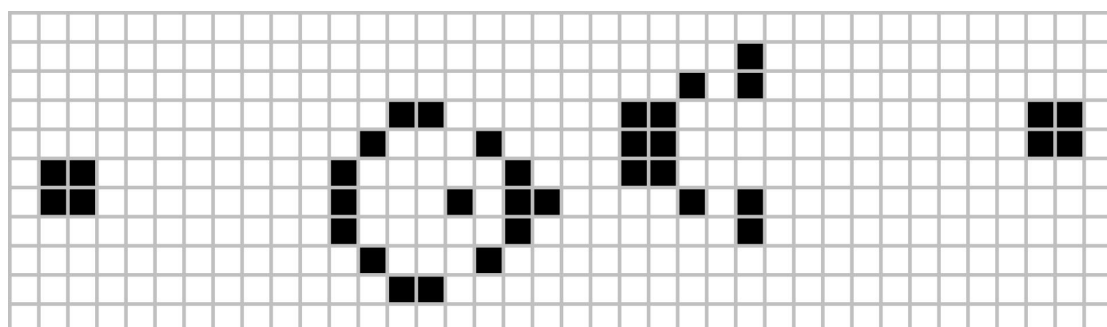
Često se pojavljuju primjeri (po tome što se često pojavljuju iz slučajne početne konfiguracije ćelija) tri gore spomenuta tipa uzoraka, a žive stanice prikazane u crnim i mrtvim ćelijama u bijelom. Period se odnosi na broj ćelija koje uzorak mora proći prije povratka u početnu konfiguraciju.

Still lifes	
Block	
Beehive	
Loaf	
Boat	
Tub	

Slika 2. Primjer živih bica koja se ne mijenja tokom vremena

Oscillators		Oscillators	
Blinker (period 2)		Blinker (period 2)	
Toad (period 2)		Toad (period 2)	
Beacon (period 2)		Beacon (period 2)	
Pulsar (period 3)		Pulsar (period 3)	
Pentadecathlon (period 15)		Pentadecathlon (period 15)	

Slika 3. Primjer živih bica koja se periodično mijenjaju tokom vremena



Slika 4. Primjer Glider gun - do beskonacnosti kreiraira Glidere

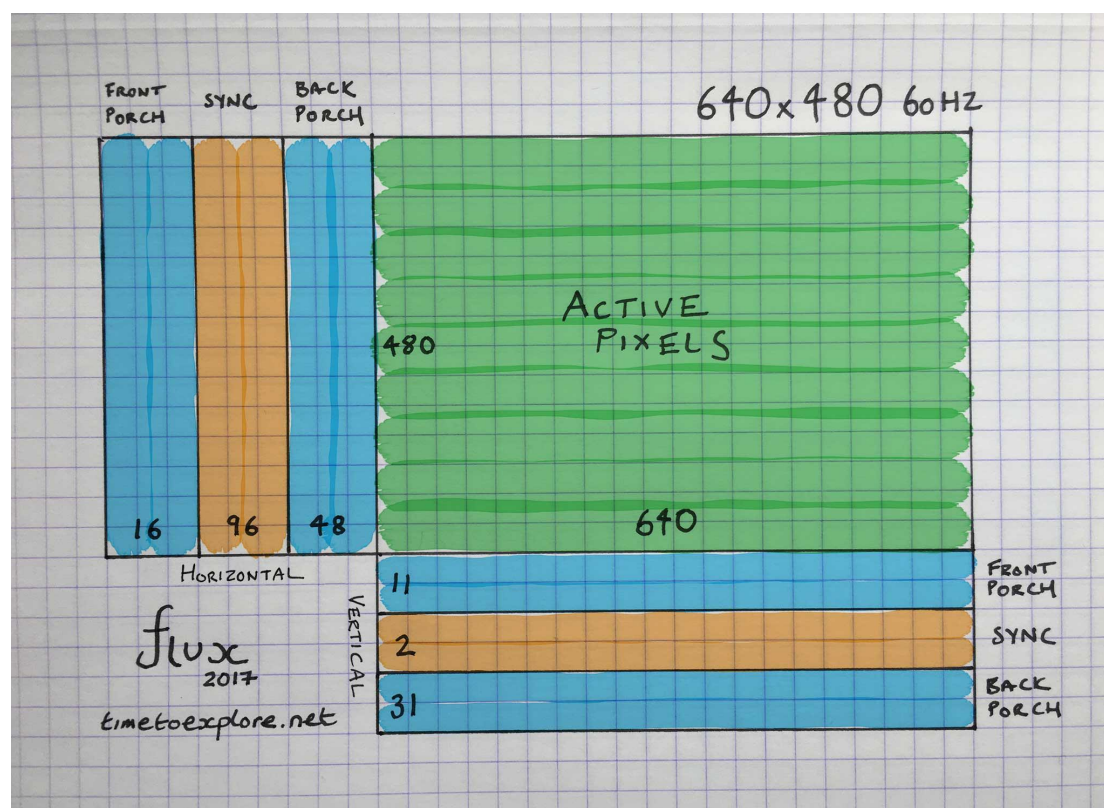
VGA

Kako radi VGA

VGA je analogni video standard koji koristi 15-pinski D-sub konektor. Ne zahteva visoke taktove ili složeno kodiranje, pa je odličan početak za učenje o FPGA grafici. VGA ima pet glavnih pinova signala: jedan za svaki od crvenih, zelenih i plavih i dva za sinhronizaciju. Horizontalna sinhronizacija označava liniju. Vertikalna sinhronizacija označava ekran, takođe poznat kao okvir.

VGA signali imaju dvije faze: crtanje piksela i interval praznog hoda. Sinhronizacijski signali se javljaju unutar intervala praznog hoda; odvojeno od crteža piksela ispred verande i zadnjeg trijema. Kada je razvijen VGA, monitori su bili bazirani na katodnim cevima (CRT): interval prekida daje vreme za stabilizaciju nivoa napona i za povratak elektronskog topa na početak linije ili ekrana.

U ovom članku kreirat ćemo klasični 640x480 60 Hz VGA zaslon. Potreban pikselni sat je 25 MHz, što je dobar okrugli dio od 100 MHz sata naše ploče. Ključ za dobijanje valjanog VGA signala je dobivanje pravih vremena. Radi jednostavnosti, uradićemo vremena u pikselima i linijama. Svaki piksel je kvačica pikselnog sata od 25 MHz (40 ns). Linija je kompletan skup horizontalnih piksela.



Slika 5. VGA sinhronizacija

Horizontal Pixel Timings

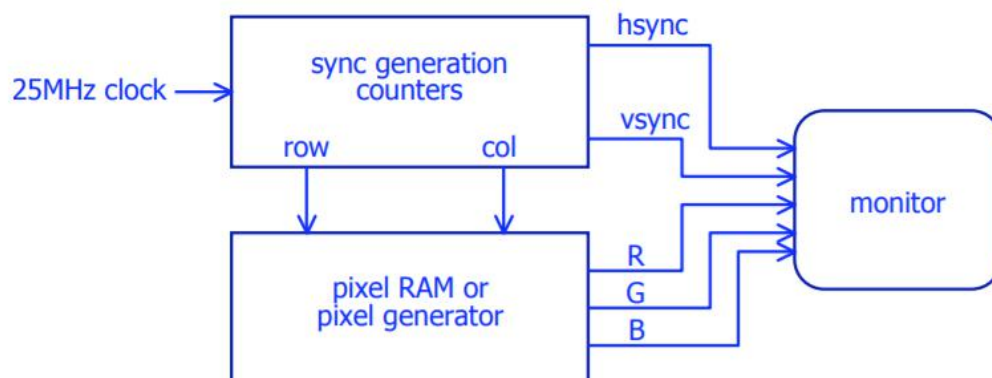
- Front Porch: 16
- Sync Pulse: 96
- Back Porch: 48
- Active Video: 640
- Total pixels: 800

Vertical Line Timings

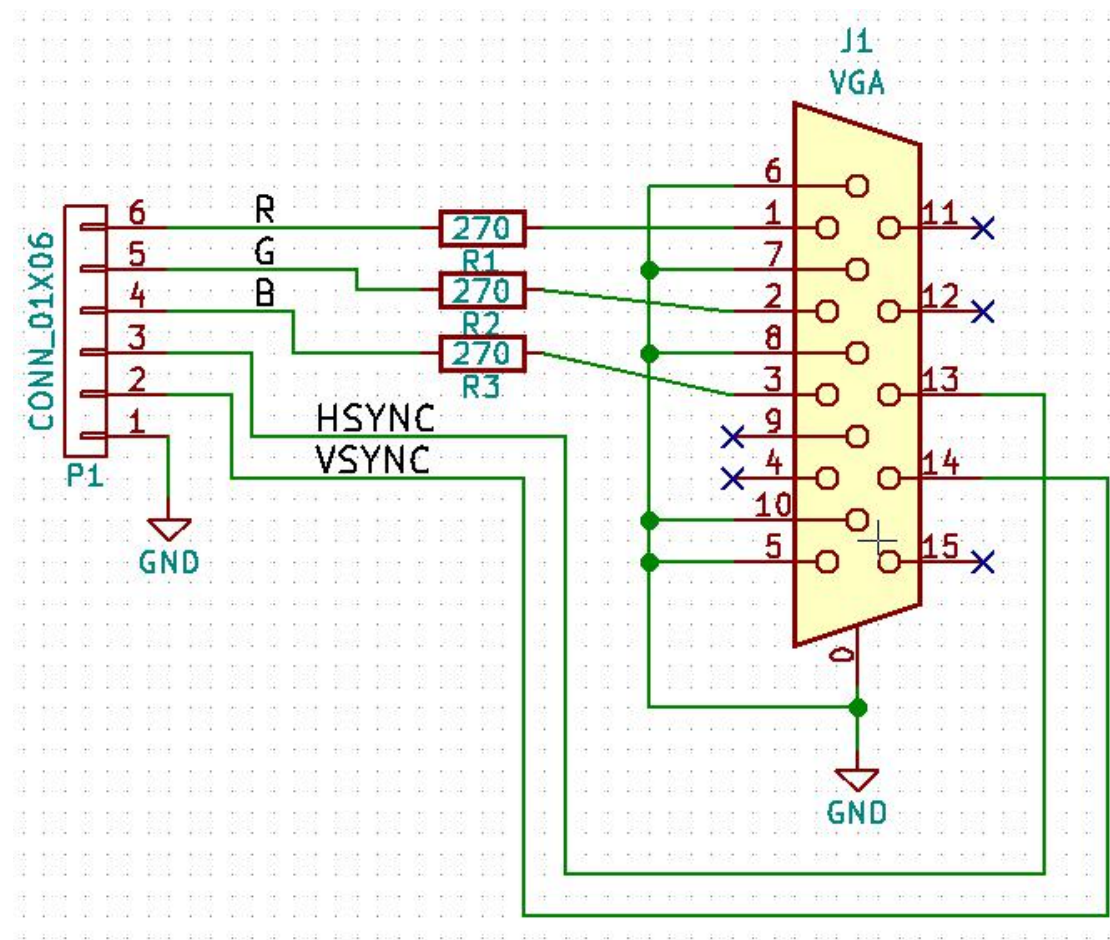
- Active Video: 480
- Front Porch: 10
- Sync Pulse: 2
- Back Porch: 33
- Total lines: 525

VGA_Param.h

```
// Horizontal Parameter ( Pixel )
parameter H_SYNC_CYC = 96;
parameter H_SYNC_BACK = 45+3;
parameter H_SYNC_ACT = 640; // 646
parameter H_SYNC_FRONT = 13+3;
parameter H_SYNC_TOTAL = 800;
// Vertical Parameter ( Line )
parameter V_SYNC_CYC = 2;
parameter V_SYNC_BACK = 30+2;
parameter V_SYNC_ACT = 480; // 484
parameter V_SYNC_FRONT = 9+2;
parameter V_SYNC_TOTAL = 525;
// Start Offset
parameter X_START = H_SYNC_CYC+H_SYNC_BACK+4;
parameter Y_START = V_SYNC_CYC+V_SYNC_BACK;
```



Slika 6. RGB v/h sync izlazi iz FPGA I VGA ulazi u monitor



Slika 7. VGA pinout

VGA sinhronizacija

```
module vga_sync(
    input iCLK, // 25 MHz clock
    input iRST_N,
    input [9:0] iRed,
    input [9:0] iGreen,
    input [9:0] iBlue,
    // pixel coordinates
    output [9:0] px,
    output [9:0] py,
    // VGA Side
    output [9:0] VGA_R,
    output [9:0] VGA_G,
    output [9:0] VGA_B,
    output reg VGA_H_SYNC,
    output reg VGA_V_SYNC,
    output VGA_SYNC,
```

```

        output VGA_BLANK
    );

    assign  VGA_BLANK    =  VGA_H_SYNC & VGA_V_SYNC;
    assign  VGA_SYNC    =  1'b0;

    reg [9:0] h_count, v_count;
    assign px = h_count;
    assign py = v_count;

    // Horizontal sync

    /* Generate Horizontal and Vertical Timing Signals for Video
    Signal
    * h_count counts pixels (640 + extra time for sync signals)
    *
    *
    *-----horiz_sync-----
    * h_count      0              640      659      755
    799
    */
    parameter H_SYNC_TOTAL = 800;
    parameter H_PIXELS    = 640;
    parameter H_SYNC_START = 659;
    parameter H_SYNC_WIDTH = 96;

    always@(posedge iCLK or negedge iRST_N)
    begin
        if(!iRST_N)
            begin
                h_count <= 10'h000;
                VGA_H_SYNC <= 1'b0;
            end
        else
            begin
                // H_Sync Counter
                if (h_count < H_SYNC_TOTAL-1) h_count <= h_count + 1'b1;
                else h_count <= 10'h0000;

                if (h_count >= H_SYNC_START &&
                    h_count < H_SYNC_START+H_SYNC_WIDTH) VGA_H_SYNC = 1'b0;
                else VGA_H_SYNC <= 1'b1;
            end
        end
    end
    /*
    *
    *-----vertical_sync-----
    *
    *-----
    * v_count      0              480
    493-494      524
    */

```

```
*/
parameter V_SYNC_TOTAL = 525;
parameter V_PIXELS      = 480;
parameter V_SYNC_START = 493;
parameter V_SYNC_WIDTH  = 2;
parameter H_START = 699;

always @(posedge iCLK or negedge iRST_N)
begin
    if (!iRST_N)
    begin
        v_count <= 10'h0000;
        VGA_V_SYNC <= 1'b0;
    end
    else if (h_count == H_START)
    begin
        // V_Sync Counter
        if (v_count < V_SYNC_TOTAL-1) v_count <= v_count + 1'b1;
        else v_count <= 10'h0000;

        if (v_count >= V_SYNC_START &&
            v_count < V_SYNC_START+V_SYNC_WIDTH) VGA_V_SYNC =
1'b0;
        else VGA_V_SYNC <= 1'b1;
    end
end

// Put all video signals through DFFs to eliminate any delays
that cause a blurry image

wire video_h_on = (h_count<H_PIXELS);
wire video_v_on = (v_count<V_PIXELS);
wire video_on = video_h_on & video_v_on;

assign VGA_R = (video_on? iRed: 10'h000);
assign VGA_G = (video_on? iGreen: 10'h000);
assign VGA_B = (video_on? iBlue: 10'h000);

/*
always @(posedge iCLK or negedge iRST_N)
    if (!iRST_N)
    begin
        VGA_R <= 10'h000;
        VGA_G <= 10'h000;
        VGA_B <= 10'h000;
    end
    else
    begin
        if (video_on)
        begin
```

```
        VGA_R[9:0] <= iRed[9:0];
        VGA_G[9:0] <= iGreen[9:0];
        VGA_B[9:0] <= iBlue[9:0];
    end
    else
    begin
        VGA_R <= 10'h000;
        VGA_G <= 10'h000;
        VGA_B <= 10'h000;
    end
end
*/

endmodule
```

VGA Audio PLL

- Alterin wizard generisan fajl - potreban za adekvatno funkcionisanje vga sinhronizacije

```
// megafunction wizard: %ALTPLL%
// GENERATION: STANDARD
// VERSION: WM1.0
// MODULE: altp1l

//
=====
===
// File Name: VGA_Audio_PLL.v
// Megafunction Name(s):
//     altp1l
//
// Simulation Library Files(s):
//     altera_mf
//
=====
===
//
*****
***
// THIS IS A WIZARD-GENERATED FILE. DO NOT EDIT THIS FILE!
//
// 7.2 Build 175 11/20/2007 SP 1 SJ Web Edition
//
*****
***
```

```
//Copyright (C) 1991-2007 Altera Corporation
//Your use of Altera Corporation's design tools, logic
functions
//and other software and tools, and its AMPP partner logic
//functions, and any output files from any of the foregoing
//(including device programming or simulation files), and any
//associated documentation or information are expressly
subject
//to the terms and conditions of the Altera Program License
//Subscription Agreement, Altera MegaCore Function License
//Agreement, or other applicable license agreement,
including,
//without limitation, that your use is for the sole purpose
of
//programming logic devices manufactured by Altera and sold
by
//Altera or its authorized distributors. Please refer to the
//applicable agreement for further details.
```

```
// synopsys translate_off
`timescale 1 ps / 1 ps
// synopsys translate_on
module VGA_Audio_PLL (
    areset,
    inclk0,
    c0,
    c1,
    c2);

    input  areset;
    input  inclk0;
    output c0;
    output c1;
    output c2;

    wire [5:0] sub_wire0;
    wire [0:0] sub_wire6 = 1'h0;
    wire [2:2] sub_wire3 = sub_wire0[2:2];
    wire [1:1] sub_wire2 = sub_wire0[1:1];
    wire [0:0] sub_wire1 = sub_wire0[0:0];
    wire c0 = sub_wire1;
    wire c1 = sub_wire2;
    wire c2 = sub_wire3;
    wire sub_wire4 = inclk0;
    wire [1:0] sub_wire5 = {sub_wire6, sub_wire4};

    altpll altpll_component (
        .inclk (sub_wire5),
        .areset (areset),
```

```

        .clk (sub_wire0),
        .activeclock (),
        .clkbad (),
        .clkena ({6{1'b1}}),
        .clkloss (),
        .clkswitch (1'b0),
        .configupdate (1'b0),
        .enable0 (),
        .enable1 (),
        .extclk (),
        .extclkena ({4{1'b1}}),
        .fbin (1'b1),
        .fbmimicbidir (),
        .fbout (),
        .locked (),
        .pfdena (1'b1),
        .phasecounterselect ({4{1'b1}}),
        .phasedone (),
        .phasestep (1'b1),
        .phaseupdown (1'b1),
        .pllana (1'b1),
        .scanaclr (1'b0),
        .scanclk (1'b0),
        .scanclkena (1'b1),
        .scandata (1'b0),
        .scandataout (),
        .scandone (),
        .scanread (1'b0),
        .scanwrite (1'b0),
        .sclkout0 (),
        .sclkout1 (),
        .vcooverrange (),
        .vcounderrange ());
defparam
    altpll_component.clk0_divide_by = 15,
    altpll_component.clk0_duty_cycle = 50,
    altpll_component.clk0_multiply_by = 14,
    altpll_component.clk0_phase_shift = "0",
    altpll_component.clk1_divide_by = 3,
    altpll_component.clk1_duty_cycle = 50,
    altpll_component.clk1_multiply_by = 2,
    altpll_component.clk1_phase_shift = "0",
    altpll_component.clk2_divide_by = 15,
    altpll_component.clk2_duty_cycle = 50,
    altpll_component.clk2_multiply_by = 14,
    altpll_component.clk2_phase_shift = "-9921",
    altpll_component.compensate_clock = "CLK0",
    altpll_component.inclk0_input_frequency = 37037,
    altpll_component.intended_device_family = "Cyclone
II",
    altpll_component.lpm_type = "altpll",

```

```
    altp11_component.operation_mode = "NORMAL",
    altp11_component.port_activeclock = "PORT_UNUSED",
    altp11_component.port_areset = "PORT_USED",
    altp11_component.port_clkbad0 = "PORT_UNUSED",
    altp11_component.port_clkbad1 = "PORT_UNUSED",
    altp11_component.port_clkloss = "PORT_UNUSED",
    altp11_component.port_clkswitch = "PORT_UNUSED",
    altp11_component.port_configupdate = "PORT_UNUSED",
    altp11_component.port_fbin = "PORT_UNUSED",
    altp11_component.port_inclk0 = "PORT_USED",
    altp11_component.port_inclk1 = "PORT_UNUSED",
    altp11_component.port_locked = "PORT_UNUSED",
    altp11_component.port_pfdena = "PORT_UNUSED",
    altp11_component.port_phasecounterselect =
"PORT_UNUSED",
    altp11_component.port_phasedone = "PORT_UNUSED",
    altp11_component.port_phasestep = "PORT_UNUSED",
    altp11_component.port_phaseupdown = "PORT_UNUSED",
    altp11_component.port_pllena = "PORT_UNUSED",
    altp11_component.port_scanaclr = "PORT_UNUSED",
    altp11_component.port_scanclk = "PORT_UNUSED",
    altp11_component.port_scanclkena = "PORT_UNUSED",
    altp11_component.port_scandata = "PORT_UNUSED",
    altp11_component.port_scandataout = "PORT_UNUSED",
    altp11_component.port_scandone = "PORT_UNUSED",
    altp11_component.port_scanread = "PORT_UNUSED",
    altp11_component.port_scanwrite = "PORT_UNUSED",
    altp11_component.port_clk0 = "PORT_USED",
    altp11_component.port_clk1 = "PORT_USED",
    altp11_component.port_clk2 = "PORT_USED",
    altp11_component.port_clk3 = "PORT_UNUSED",
    altp11_component.port_clk4 = "PORT_UNUSED",
    altp11_component.port_clk5 = "PORT_UNUSED",
    altp11_component.port_clkena0 = "PORT_UNUSED",
    altp11_component.port_clkena1 = "PORT_UNUSED",
    altp11_component.port_clkena2 = "PORT_UNUSED",
    altp11_component.port_clkena3 = "PORT_UNUSED",
    altp11_component.port_clkena4 = "PORT_UNUSED",
    altp11_component.port_clkena5 = "PORT_UNUSED",
    altp11_component.port_extclk0 = "PORT_UNUSED",
    altp11_component.port_extclk1 = "PORT_UNUSED",
    altp11_component.port_extclk2 = "PORT_UNUSED",
    altp11_component.port_extclk3 = "PORT_UNUSED";

endmodule

//
=====
===
// CNX file retrieval info
```



```
//  
=====
```



```
===  
// Retrieval info: PRIVATE: ACTIVECLK_CHECK STRING "0"  
// Retrieval info: PRIVATE: BANDWIDTH STRING "1.000"  
// Retrieval info: PRIVATE: BANDWIDTH_FEATURE_ENABLED STRING  
"0"  
// Retrieval info: PRIVATE: BANDWIDTH_FREQ_UNIT STRING "MHz"  
// Retrieval info: PRIVATE: BANDWIDTH_PRESET STRING "Low"  
// Retrieval info: PRIVATE: BANDWIDTH_USE_AUTO STRING "1"  
// Retrieval info: PRIVATE: BANDWIDTH_USE_CUSTOM STRING "0"  
// Retrieval info: PRIVATE: BANDWIDTH_USE_PRESET STRING "0"  
// Retrieval info: PRIVATE: CLKBAD_SWITCHOVER_CHECK STRING  
"0"  
// Retrieval info: PRIVATE: CLKLOSS_CHECK STRING "0"  
// Retrieval info: PRIVATE: CLKSWITCH_CHECK STRING "1"  
// Retrieval info: PRIVATE: CNX_NO_COMPENSATE_RADIO STRING  
"0"  
// Retrieval info: PRIVATE: CREATE_CLKBAD_CHECK STRING "0"  
// Retrieval info: PRIVATE: CREATE_INCLK1_CHECK STRING "0"  
// Retrieval info: PRIVATE: CUR_DEDICATED_CLK STRING "c0"  
// Retrieval info: PRIVATE: CUR_FBIN_CLK STRING "e0"  
// Retrieval info: PRIVATE: DEVICE_SPEED_GRADE STRING "Any"  
// Retrieval info: PRIVATE: DIV_FACTOR0 NUMERIC "1"  
// Retrieval info: PRIVATE: DIV_FACTOR1 NUMERIC "6"  
// Retrieval info: PRIVATE: DIV_FACTOR2 NUMERIC "1"  
// Retrieval info: PRIVATE: DUTY_CYCLE0 STRING "50.00000000"  
// Retrieval info: PRIVATE: DUTY_CYCLE1 STRING "50.00000000"  
// Retrieval info: PRIVATE: DUTY_CYCLE2 STRING "50.00000000"  
// Retrieval info: PRIVATE: EXPLICIT_SWITCHOVER_COUNTER  
STRING "0"  
// Retrieval info: PRIVATE: EXT_FEEDBACK_RADIO STRING "0"  
// Retrieval info: PRIVATE: GLOCKED_COUNTER_EDIT_CHANGED  
STRING "1"  
// Retrieval info: PRIVATE: GLOCKED_FEATURE_ENABLED STRING  
"1"  
// Retrieval info: PRIVATE: GLOCKED_MODE_CHECK STRING "0"  
// Retrieval info: PRIVATE: GLOCK_COUNTER_EDIT NUMERIC  
"1048575"  
// Retrieval info: PRIVATE: HAS_MANUAL_SWITCHOVER STRING "1"  
// Retrieval info: PRIVATE: INCLK0_FREQ_EDIT STRING "27.000"  
// Retrieval info: PRIVATE: INCLK0_FREQ_UNIT_COMBO STRING  
"MHz"  
// Retrieval info: PRIVATE: INCLK1_FREQ_EDIT STRING "27.000"  
// Retrieval info: PRIVATE: INCLK1_FREQ_EDIT_CHANGED STRING  
"1"  
// Retrieval info: PRIVATE: INCLK1_FREQ_UNIT_CHANGED STRING  
"1"  
// Retrieval info: PRIVATE: INCLK1_FREQ_UNIT_COMBO STRING  
"MHz"
```

```
// Retrieval info: PRIVATE: INTENDED_DEVICE_FAMILY STRING  
"Cyclone II"  
// Retrieval info: PRIVATE: INT_FEEDBACK__MODE_RADIO STRING  
"1"  
// Retrieval info: PRIVATE: LOCKED_OUTPUT_CHECK STRING "0"  
// Retrieval info: PRIVATE: LONG_SCAN_RADIO STRING "1"  
// Retrieval info: PRIVATE: LVDS_MODE_DATA_RATE STRING "Not  
Available"  
// Retrieval info: PRIVATE: LVDS_MODE_DATA_RATE_DIRTY  
NUMERIC "0"  
// Retrieval info: PRIVATE: LVDS_PHASE_SHIFT_UNIT0 STRING  
"deg"  
// Retrieval info: PRIVATE: LVDS_PHASE_SHIFT_UNIT1 STRING  
"deg"  
// Retrieval info: PRIVATE: LVDS_PHASE_SHIFT_UNIT2 STRING  
"ps"  
// Retrieval info: PRIVATE: MIRROR_CLK0 STRING "0"  
// Retrieval info: PRIVATE: MIRROR_CLK1 STRING "0"  
// Retrieval info: PRIVATE: MIRROR_CLK2 STRING "0"  
// Retrieval info: PRIVATE: MULT_FACTOR0 NUMERIC "1"  
// Retrieval info: PRIVATE: MULT_FACTOR1 NUMERIC "5"  
// Retrieval info: PRIVATE: MULT_FACTOR2 NUMERIC "1"  
// Retrieval info: PRIVATE: NORMAL_MODE_RADIO STRING "1"  
// Retrieval info: PRIVATE: OUTPUT_FREQ0 STRING "25.20000000"  
// Retrieval info: PRIVATE: OUTPUT_FREQ1 STRING "18.00000000"  
// Retrieval info: PRIVATE: OUTPUT_FREQ2 STRING "25.20000000"  
// Retrieval info: PRIVATE: OUTPUT_FREQ_MODE0 STRING "1"  
// Retrieval info: PRIVATE: OUTPUT_FREQ_MODE1 STRING "1"  
// Retrieval info: PRIVATE: OUTPUT_FREQ_MODE2 STRING "1"  
// Retrieval info: PRIVATE: OUTPUT_FREQ_UNIT0 STRING "MHz"  
// Retrieval info: PRIVATE: OUTPUT_FREQ_UNIT1 STRING "MHz"  
// Retrieval info: PRIVATE: OUTPUT_FREQ_UNIT2 STRING "MHz"  
// Retrieval info: PRIVATE: PHASE_RECONFIG_FEATURE_ENABLED  
STRING "0"  
// Retrieval info: PRIVATE: PHASE_RECONFIG_INPUTS_CHECK  
STRING "0"  
// Retrieval info: PRIVATE: PHASE_SHIFT0 STRING "0.00000000"  
// Retrieval info: PRIVATE: PHASE_SHIFT1 STRING "0.00000000"  
// Retrieval info: PRIVATE: PHASE_SHIFT2 STRING  
"-90.00000000"  
// Retrieval info: PRIVATE: PHASE_SHIFT_STEP_ENABLED_CHECK  
STRING "0"  
// Retrieval info: PRIVATE: PHASE_SHIFT_UNIT0 STRING "deg"  
// Retrieval info: PRIVATE: PHASE_SHIFT_UNIT1 STRING "deg"  
// Retrieval info: PRIVATE: PHASE_SHIFT_UNIT2 STRING "deg"  
// Retrieval info: PRIVATE: PLL_ADVANCED_PARAM_CHECK STRING  
"0"  
// Retrieval info: PRIVATE: PLL_ARESET_CHECK STRING "1"  
// Retrieval info: PRIVATE: PLL_AUTOPLL_CHECK NUMERIC "1"  
// Retrieval info: PRIVATE: PLL_ENA_CHECK STRING "0"  
// Retrieval info: PRIVATE: PLL_ENHPLL_CHECK NUMERIC "0"
```

```
// Retrieval info: PRIVATE: PLL_FASTPLL_CHECK NUMERIC "0"
// Retrieval info: PRIVATE: PLL_FBMIMIC_CHECK STRING "0"
// Retrieval info: PRIVATE: PLL_LVDS_PLL_CHECK NUMERIC "0"
// Retrieval info: PRIVATE: PLL_PFDENA_CHECK STRING "0"
// Retrieval info: PRIVATE: PLL_TARGET_HARCOPY_CHECK NUMERIC
"0"
// Retrieval info: PRIVATE: PRIMARY_CLK_COMBO STRING "inclk0"
// Retrieval info: PRIVATE: RECONFIG_FILE STRING
"VGA_Audio_PLL.mif"
// Retrieval info: PRIVATE: SACN_INPUTS_CHECK STRING "0"
// Retrieval info: PRIVATE: SCAN_FEATURE_ENABLED STRING "0"
// Retrieval info: PRIVATE: SELF_RESET_LOCK_LOSS STRING "0"
// Retrieval info: PRIVATE: SHORT_SCAN_RADIO STRING "0"
// Retrieval info: PRIVATE: SPREAD_FEATURE_ENABLED STRING "0"
// Retrieval info: PRIVATE: SPREAD_FREQ STRING "50.000"
// Retrieval info: PRIVATE: SPREAD_FREQ_UNIT STRING "KHz"
// Retrieval info: PRIVATE: SPREAD_PERCENT STRING "0.500"
// Retrieval info: PRIVATE: SPREAD_USE STRING "0"
// Retrieval info: PRIVATE: SRC_SYNCH_COMP_RADIO STRING "0"
// Retrieval info: PRIVATE: STICKY_CLK0 STRING "1"
// Retrieval info: PRIVATE: STICKY_CLK1 STRING "1"
// Retrieval info: PRIVATE: STICKY_CLK2 STRING "1"
// Retrieval info: PRIVATE: SWITCHOVER_COUNT_EDIT NUMERIC "1"
// Retrieval info: PRIVATE: SWITCHOVER_FEATURE_ENABLED
STRING "1"
// Retrieval info: PRIVATE: SYNTH_WRAPPER_GEN_POSTFIX STRING
"0"
// Retrieval info: PRIVATE: USE_CLK0 STRING "1"
// Retrieval info: PRIVATE: USE_CLK1 STRING "1"
// Retrieval info: PRIVATE: USE_CLK2 STRING "1"
// Retrieval info: PRIVATE: USE_CLKENA0 STRING "0"
// Retrieval info: PRIVATE: USE_CLKENA1 STRING "0"
// Retrieval info: PRIVATE: USE_CLKENA2 STRING "0"
// Retrieval info: PRIVATE: USE_MIL_SPEED_GRADE NUMERIC "0"
// Retrieval info: PRIVATE: ZERO_DELAY_RADIO STRING "0"
// Retrieval info: LIBRARY: altera_mf
altera_mf.altera_mf_components.all
// Retrieval info: CONSTANT: CLK0_DIVIDE_BY NUMERIC "15"
// Retrieval info: CONSTANT: CLK0_DUTY_CYCLE NUMERIC "50"
// Retrieval info: CONSTANT: CLK0_MULTIPLY_BY NUMERIC "14"
// Retrieval info: CONSTANT: CLK0_PHASE_SHIFT STRING "0"
// Retrieval info: CONSTANT: CLK1_DIVIDE_BY NUMERIC "3"
// Retrieval info: CONSTANT: CLK1_DUTY_CYCLE NUMERIC "50"
// Retrieval info: CONSTANT: CLK1_MULTIPLY_BY NUMERIC "2"
// Retrieval info: CONSTANT: CLK1_PHASE_SHIFT STRING "0"
// Retrieval info: CONSTANT: CLK2_DIVIDE_BY NUMERIC "15"
// Retrieval info: CONSTANT: CLK2_DUTY_CYCLE NUMERIC "50"
// Retrieval info: CONSTANT: CLK2_MULTIPLY_BY NUMERIC "14"
// Retrieval info: CONSTANT: CLK2_PHASE_SHIFT STRING "-9921"
// Retrieval info: CONSTANT: COMPENSATE_CLOCK STRING "CLK0"
```

```
// Retrieval info: CONSTANT: INCLK0_INPUT_FREQUENCY NUMERIC
"37037"
// Retrieval info: CONSTANT: INTENDED_DEVICE_FAMILY STRING
"Cyclone II"
// Retrieval info: CONSTANT: LPM_TYPE STRING "altpll"
// Retrieval info: CONSTANT: OPERATION_MODE STRING "NORMAL"
// Retrieval info: CONSTANT: PORT_ACTIVECLOCK STRING
"PORT_UNUSED"
// Retrieval info: CONSTANT: PORT_ARESET STRING "PORT_USED"
// Retrieval info: CONSTANT: PORT_CLKBAD0 STRING
"PORT_UNUSED"
// Retrieval info: CONSTANT: PORT_CLKBAD1 STRING
"PORT_UNUSED"
// Retrieval info: CONSTANT: PORT_CLKLOSS STRING
"PORT_UNUSED"
// Retrieval info: CONSTANT: PORT_CLKSWITCH STRING
"PORT_UNUSED"
// Retrieval info: CONSTANT: PORT_CONFIGUPDATE STRING
"PORT_UNUSED"
// Retrieval info: CONSTANT: PORT_FBIN STRING "PORT_UNUSED"
// Retrieval info: CONSTANT: PORT_INCLK0 STRING "PORT_USED"
// Retrieval info: CONSTANT: PORT_INCLK1 STRING "PORT_UNUSED"
// Retrieval info: CONSTANT: PORT_LOCKED STRING "PORT_UNUSED"
// Retrieval info: CONSTANT: PORT_PFDENA STRING "PORT_UNUSED"
// Retrieval info: CONSTANT: PORT_PHASECOUNTERSELECT STRING
"PORT_UNUSED"
// Retrieval info: CONSTANT: PORT_PHASEDONE STRING
"PORT_UNUSED"
// Retrieval info: CONSTANT: PORT_PHASESTEP STRING
"PORT_UNUSED"
// Retrieval info: CONSTANT: PORT_PHASEUPDOWN STRING
"PORT_UNUSED"
// Retrieval info: CONSTANT: PORT_PLENA STRING "PORT_UNUSED"
// Retrieval info: CONSTANT: PORT_SCANACLK STRING
"PORT_UNUSED"
// Retrieval info: CONSTANT: PORT_SCANCLK STRING
"PORT_UNUSED"
// Retrieval info: CONSTANT: PORT_SCANCLKENA STRING
"PORT_UNUSED"
// Retrieval info: CONSTANT: PORT_SCANDATA STRING
"PORT_UNUSED"
// Retrieval info: CONSTANT: PORT_SCANDATAOUT STRING
"PORT_UNUSED"
// Retrieval info: CONSTANT: PORT_SCANDONE STRING
"PORT_UNUSED"
// Retrieval info: CONSTANT: PORT_SCANREAD STRING
"PORT_UNUSED"
// Retrieval info: CONSTANT: PORT_SCANWRITE STRING
"PORT_UNUSED"
// Retrieval info: CONSTANT: PORT_clk0 STRING "PORT_USED"
// Retrieval info: CONSTANT: PORT_clk1 STRING "PORT_USED"
```

```
// Retrieval info: CONSTANT: PORT_clk2 STRING "PORT_USED"
// Retrieval info: CONSTANT: PORT_clk3 STRING "PORT_UNUSED"
// Retrieval info: CONSTANT: PORT_clk4 STRING "PORT_UNUSED"
// Retrieval info: CONSTANT: PORT_clk5 STRING "PORT_UNUSED"
// Retrieval info: CONSTANT: PORT_clkena0 STRING
"PORT_UNUSED"
// Retrieval info: CONSTANT: PORT_clkena1 STRING
"PORT_UNUSED"
// Retrieval info: CONSTANT: PORT_clkena2 STRING
"PORT_UNUSED"
// Retrieval info: CONSTANT: PORT_clkena3 STRING
"PORT_UNUSED"
// Retrieval info: CONSTANT: PORT_clkena4 STRING
"PORT_UNUSED"
// Retrieval info: CONSTANT: PORT_clkena5 STRING
"PORT_UNUSED"
// Retrieval info: CONSTANT: PORT_extclk0 STRING
"PORT_UNUSED"
// Retrieval info: CONSTANT: PORT_extclk1 STRING
"PORT_UNUSED"
// Retrieval info: CONSTANT: PORT_extclk2 STRING
"PORT_UNUSED"
// Retrieval info: CONSTANT: PORT_extclk3 STRING
"PORT_UNUSED"
// Retrieval info: USED_PORT: @clk 0 0 6 0 OUTPUT_CLK_EXT VCC
"@clk[5..0]"
// Retrieval info: USED_PORT: @extclk 0 0 4 0 OUTPUT_CLK_EXT
VCC "@extclk[3..0]"
// Retrieval info: USED_PORT: areset 0 0 0 0 INPUT GND "areset"
// Retrieval info: USED_PORT: c0 0 0 0 0 OUTPUT_CLK_EXT VCC
"c0"
// Retrieval info: USED_PORT: c1 0 0 0 0 OUTPUT_CLK_EXT VCC
"c1"
// Retrieval info: USED_PORT: c2 0 0 0 0 OUTPUT_CLK_EXT VCC
"c2"
// Retrieval info: USED_PORT: inclk0 0 0 0 0 INPUT_CLK_EXT
GND "inclk0"
// Retrieval info: CONNECT: @inclk 0 0 1 0 inclk0 0 0 0 0
// Retrieval info: CONNECT: c0 0 0 0 0 @clk 0 0 1 0
// Retrieval info: CONNECT: c1 0 0 0 0 @clk 0 0 1 1
// Retrieval info: CONNECT: c2 0 0 0 0 @clk 0 0 1 2
// Retrieval info: CONNECT: @inclk 0 0 1 1 GND 0 0 0 0
// Retrieval info: CONNECT: @areset 0 0 0 0 areset 0 0 0 0
// Retrieval info: GEN_FILE: TYPE_NORMAL VGA_Audio_PLL.v TRUE
FALSE
// Retrieval info: GEN_FILE: TYPE_NORMAL VGA_Audio_PLL.inc
FALSE FALSE
// Retrieval info: GEN_FILE: TYPE_NORMAL VGA_Audio_PLL.cmp
FALSE FALSE
// Retrieval info: GEN_FILE: TYPE_NORMAL VGA_Audio_PLL.bsf
FALSE FALSE
```

```
//      Retrieval      info:      GEN_FILE:      TYPE_NORMAL
VGA_Audio_PLL_inst.v FALSE FALSE
// Retrieval info: GEN_FILE: TYPE_NORMAL VGA_Audio_PLL_bb.v
FALSE FALSE
//      Retrieval      info:      GEN_FILE:      TYPE_NORMAL
VGA_Audio_PLL_waveforms.html TRUE FALSE
//      Retrieval      info:      GEN_FILE:      TYPE_NORMAL
VGA_Audio_PLL_wave*.jpg FALSE FALSE
// Retrieval info: GEN_FILE: TYPE_NORMAL VGA_Audio_PLL.ppf
TRUE FALSE
// Retrieval info: LIB_FILE: altera_mf
```



Slika 8. testne fotografije iscertavanja kvadratica razlicitih boja na VGA

C IMPLEMENTACIJA

Kod

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <time.h>

#define for_x for (int x = 0; x < w; x++)
#define for_y for (int y = 0; y < h; y++)
#define for_xy for_x for_y
void show(void *u, int w, int h)
{
    int (*univ)[w] = u;
    printf("\033[H");
    for_y {
        for_x printf(univ[y][x] ? "\033[07m \033[m" : " ");
        printf("\033[E");
    }
    fflush(stdout);
}

void evolve(void *u, int w, int h)
{
    unsigned (*univ)[w] = u;
    unsigned new[h][w];

    for_y for_x {
        int n = 0;
        for (int y1 = y - 1; y1 <= y + 1; y1++)
            for (int x1 = x - 1; x1 <= x + 1; x1++)
                if (univ[(y1 + h) % h][(x1 + w) % w])
                    n++;

        if (univ[y][x]) n--;
        new[y][x] = (n == 3 || (n == 2 && univ[y][x]));
    }
    for_y for_x univ[y][x] = new[y][x];
}

void game(int w, int h)
{
    unsigned univ[h][w];
    for_xy univ[y][x] = rand() < RAND_MAX / 10 ? 1 : 0;
    for(int i = 0; i < 10000; i++)
```

```
    {
        show(univ, w, h);
        evolve(univ, w, h);
    }
}

int main(int c, char **v)
{
    time_t begin = clock();

    int w = 0, h = 0;
    if (c > 1) w = atoi(v[1]);
    if (c > 2) h = atoi(v[2]);
    if (w <= 0) w = 64;
    if (h <= 0) h = 48;
    game(w, h);

    time_t end = clock();
    double time_spent = (double)(end - begin) /
CLOCKS_PER_CYCLE * 2;

    printf("\nPotrebno vrijeme da se izvrsi 10.000 generacija
je : %f sekundi\n", time_spent);
}
```


Usporedba: Raspberry Pi vs. Thinkpad t420

Raspberry Pi 3 model B



Slika 8. Raspberry pi 3 model b - uređaj koji je korišten

Raspberry Pi 3 Specifications

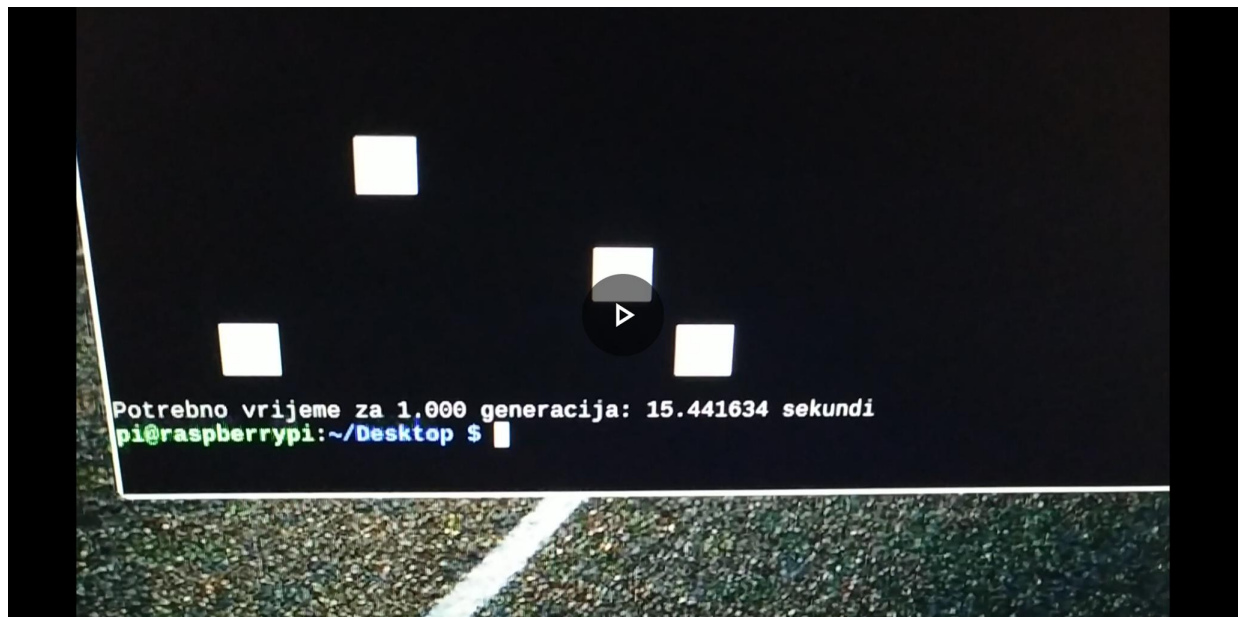
SoC: Broadcom BCM2837

CPU: 4× ARM Cortex-A53, 1.2GHz

GPU: Broadcom VideoCore IV

RAM: 1GB LPDDR2 (900 MHz)

Slika 9. Specifikacije R Pi

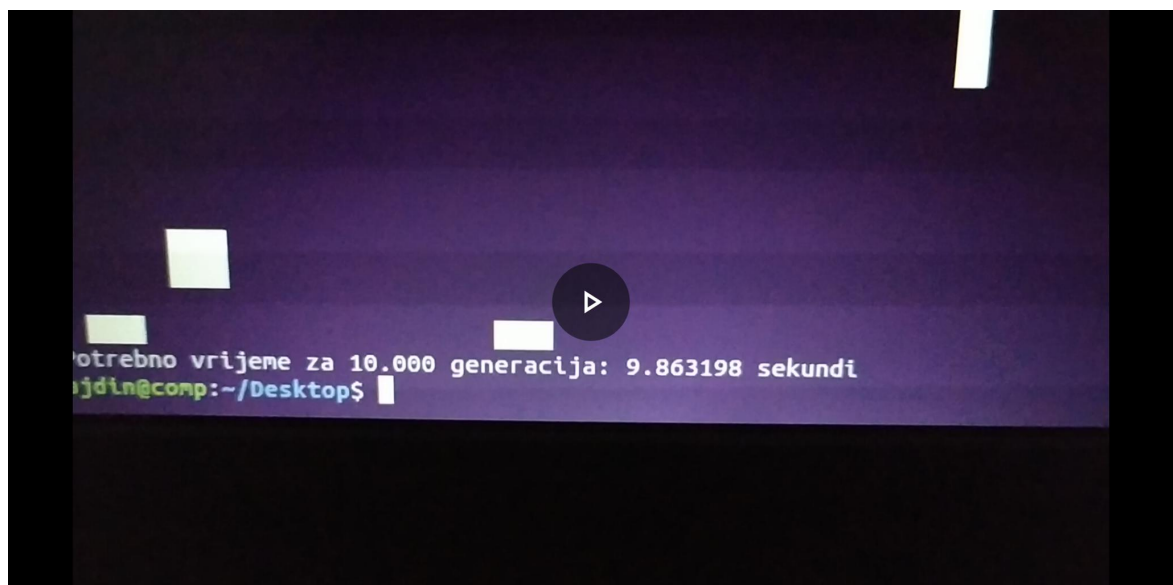


Slika 11. trajanje generisanja 1.000 generacija na Raspberry Pi - 15.44 sek.

Lenovo Thnikpad t420

Intel® Core™ i5 processor i5-2410M with dual-core, dual thread
DDR3 memory controller (up to 1333MHz), Intel Turbo Boost, Hyper-Threading
technology; 3MB cache

RAM: 8GB

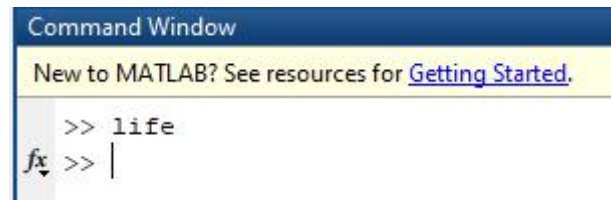


Slika 12. trajanje generisanja 10.000 generacija na LT t420 - 9.83 sek.

MATLAB implementacija

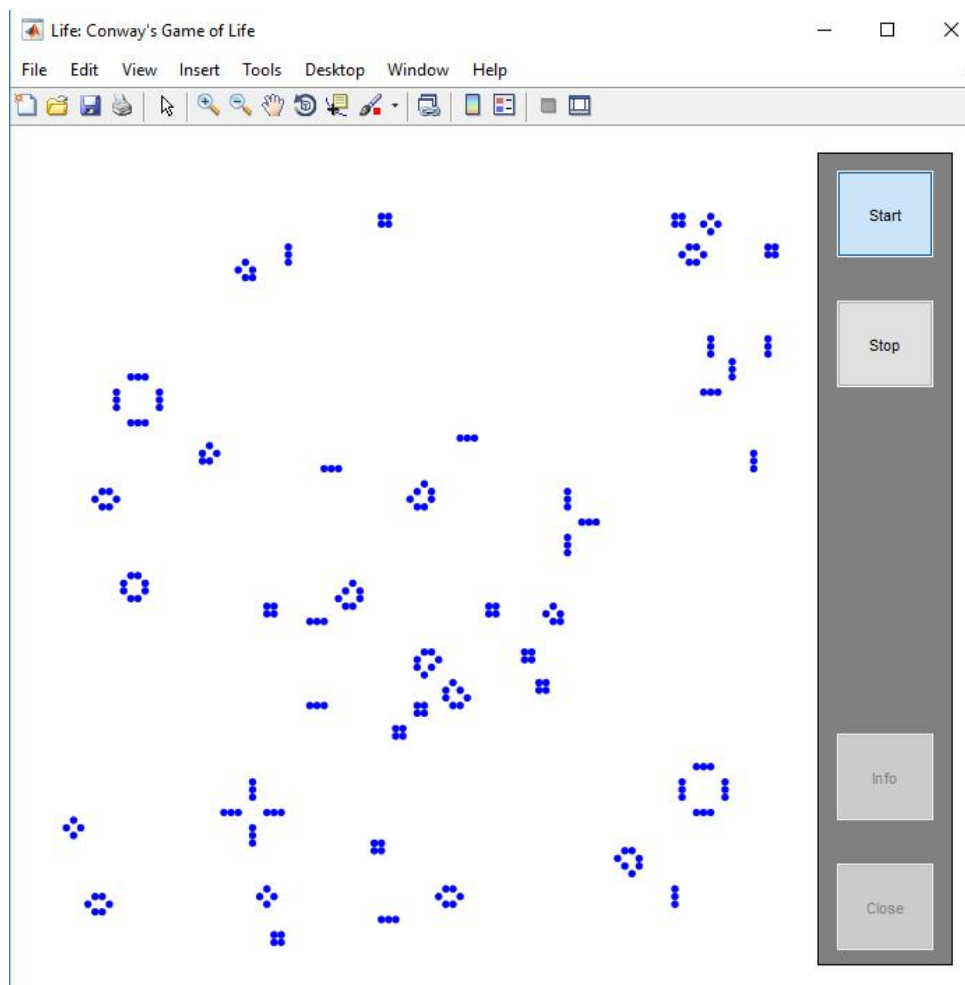
MATLAB ima igradjeni Game of Life GUI. Potrebno je samo upisati u komandni prostor:

Life



Slika 13. poziv matlab funkcije u komnadnom prostoru

Te ce se pokrenuti:



Slika 14. Game of Life implementacija u Matlabu

Da biste vidjeli izvorni kod ukucajte:

```
open(fullfile(matlabroot, 'toolbox', 'matlab', 'demos',  
'life.m'))
```

Primjer koda jednostostavni primjer Game of Life
N generacija
U kvadratu stranice N

```
function GoL(S, N) %  
    colormap copper; whitebg('black');  
    G= round(rand(S));  
    A = [S 1:S-1]; B = [2:S 1];  
    for k=1:N  
        Sum  
        G(A,:)+G(B,:)+G(:,B)+G(:,A)+G(A,B)+G(A,A)+G(B,B)+G(B,A);  
        G = double((G & (Sum == 2)) | (Sum == 3));  
        surf(G); view([0 90]); pause(0.001)  
    end  
end
```

FPGA Verilog implementacija

Pocetno stanje

Postavljanje "igralista"
Tj. Pocetnog stanja igre
64*48 kvadratica

[illegible]

[illegible]

[illegible]

```
        cells_preset[64*46:64*46+63]                                     <=
64'b0000000000000000000000000000000000000000000000000000000000000000
000000000000;
        cells_preset[64*47:64*47+63]                                     <=
64'b0000000000000000000000000000000000000000000000000000000000000000
000000000000;
End
```

Populacija

Odredjivanje susjeda I proračun populacija za svaki kvadratic

```
if (i == 0)
    begin
        // Top-left square
        population <= cells_preset[64*48 - 1] +
                                cells_preset[64*48 - 64] +
                                cells_preset[64*48 - 64 + 1] +
                                cells_preset[i      + 64 - 1] +
                                cells_preset[i      + 1] +
                                cells_preset[64      + 64 - 1] +
                                cells_preset[64] +
                                cells_preset[64      + 1];
    end

    else if (i == 63)
    begin
        // Top-right square
        population <= cells_preset[64*48 - 1 - 1] +
                                cells_preset[64*48 - 1] +
                                cells_preset[64*48 - 64] +
                                cells_preset[i      - 1] +
                                cells_preset[0] +
                                cells_preset[i      + 64 - 1] +
                                cells_preset[i      + 64] +
                                cells_preset[i      + 1];
    end

    else if (i == 64*48 - 64)
    begin
        // Bottom-left square
        population <= cells_preset[i - 1] +
                                cells_preset[i - 64] +
                                cells_preset[i - 64 + 1] +
                                cells_preset[i + 64 - 1] +
                                cells_preset[i + 1] +
                                cells_preset[0 + 64 - 1] +
                                cells_preset[0] +
                                cells_preset[0 + 1];
    end
end
```



```

        else if (i == 64*48 - 1)
        begin
            // Bottom-right square
            population <= cells_preset[i - 64 - 1] +
                                cells_preset[i - 64] +
                                cells_preset[i - 64 - 64 + 1] +
                                cells_preset[i - 1] +
                                cells_preset[i - 64 + 1] +
                                cells_preset[0 + 64 - 1 - 1] +
                                cells_preset[0 + 64 - 1] +
                                cells_preset[0];
        end

        else if (i < 63)
        begin
            // Top row
            population <= cells_preset[64*48 - 64 + i - 1] +
                                cells_preset[64*48 - 64 + i] +
                                cells_preset[64*48 - 64 + i + 1] +
                                cells_preset[i - 1] +
                                cells_preset[i + 1] +
                                cells_preset[i + 64 - 1] +
                                cells_preset[i + 64] +
                                cells_preset[i + 64 + 1];
        end

        else if (i > 64*48 - 64)
        begin
            // Bottom row
            population <= cells_preset[i - 64 - 1] +
                                cells_preset[i - 64] +
                                cells_preset[i - 64 + 1] +
                                cells_preset[i - 1] +
                                cells_preset[i + 1] +
                                cells_preset[0 + i - 1] +
                                cells_preset[0 + i] +
                                cells_preset[0 + i + 1];
        end

        else if (i % 64 == 0)
        begin
            // Leftmost column
            population <= cells_preset[i - 1] +
                                cells_preset[i - 64] +
                                cells_preset[i - 64 + 1] +
                                cells_preset[i + 64 - 1] +
                                cells_preset[i + 1] +
                                cells_preset[i + 64 + 64 - 1] +
                                cells_preset[i + 64] +
                                cells_preset[i + 64 + 1];
        end

        else if ((i + 1) % 64 == 0)
        begin
            // Rightmost column
            population <= cells_preset[i - 64 - 1] +
                                cells_preset[i - 64] +

```

```

                                cells_preset[i - 64 - 64 + 1] +
                                cells_preset[i - 1] +
                                cells_preset[i - 64 + 1] +
                                cells_preset[i + 64 - 1] +
                                cells_preset[i + 64] +
                                cells_preset[i + 1];
end

    else
    begin
    // Middle squares
    population <= cells_preset[i - 64 - 1] +
                                cells_preset[i - 64] +
                                cells_preset[i - 64 + 1] +
                                cells_preset[i - 1] +
                                cells_preset[i + 1] +
                                cells_preset[i + 64 - 1] +
                                cells_preset[i + 64] +
                                cells_preset[i + 64 + 1];
end
end
```

Clock

Generisanje clock-a koji ce usporiti proracun - iscrtavanje kako bi se lakse uocile promjene

```
module clock(Clk, Clk_50);  
    input Clk_50;  
    output Clk;  
    reg [31:0] counter;  
  
    always @(posedge Clk_50)  
    begin  
        if (counter == 32'd1600)  
            counter <= 0;  
        else  
            counter <= counter + 32'd1;  
        end  
        assign Clk = (counter == 32'd1600);  
Endmodule
```

MODUL GLAVNE FUNKCIJE

```
module vgalab1(
// Clock Input
    input CLOCK_50, // 50 MHz
    input CLOCK_27, // 27 MHz
// Push Button
    input [3:0] KEY, // Pushbutton[3:0]
// DPDT Switch
    input [17:0] SW, // Toggle Switch[17:0]
// 7-SEG Display
    output [6:0] HEX0,HEX1,HEX2,HEX3,HEX4,HEX5,HEX6,HEX7,
// Seven Segment Digits
// LED
    output [8:0] LEDG, //LED Green[8:0]
    output [17:0] LEDR, // LED Red[17:0]
// GPIO
    inout [35:0] GPIO_0,GPIO_1, // GPIO Connections
// TV Decoder
//TD_DATA, // TV Decoder Data bus 8 bits
//TD_HS, // TV Decoder H_SYNC
//TD_VS, // TV Decoder V_SYNC
    output TD_RESET, // TV Decoder Reset
// VGA
    output VGA_CLK, // VGA Clock
    output VGA_HS, // VGA H_SYNC
    output VGA_VS, // VGA V_SYNC
    output VGA_BLANK, // VGA BLANK
    output VGA_SYNC, // VGA SYNC
    output [9:0] VGA_R, // VGA Red[9:0]
    output [9:0] VGA_G, // VGA Green[9:0]
    output [9:0] VGA_B, // VGA Blue[9:0]
);

// All inout port turn to tri-state
assign GPIO_0 = 36'hzzzzzzzzzz;
assign GPIO_1 = 36'hzzzzzzzzzz;

wire RST;
assign RST = KEY[0];

// reset delay gives some time for peripherals to initialize
wire DLY_RST;
Reset_Delay r0( .iCLK(CLOCK_50),.oRESET(DLY_RST) );

// Send switches to red leds
assign LEDR = SW;
```

```
// Turn off green leds
assign LEDG = 8'h00;

wire [6:0] blank = 7'b111_1111;

// blank unused 7-segment digits
assign HEX0 = blank;
assign HEX1 = blank;
assign HEX2 = blank;
assign HEX3 = blank;
assign HEX4 = blank;
assign HEX5 = blank;
assign HEX6 = blank;
assign HEX7 = blank;

wire      VGA_CTRL_CLK;
wire      AUD_CTRL_CLK;
wire [9:0] mVGA_R;
wire [9:0] mVGA_G;
wire [9:0] mVGA_B;
wire [9:0] mCoord_X;
wire [9:0] mCoord_Y;

assign    TD_RESET = 1'b1; // Enable 27 MHz

VGA_Audio_PLL p1 (
    .areset(~DLY_RST),
    .inclk0(CLOCK_27),
    .c0(VGA_CTRL_CLK),
    .c1(AUD_CTRL_CLK),
    .c2(VGA_CLK)
);

wire [9:0] r, g, b;

////////////////////////////////////
// game of life module
game_of_life c1(mCoord_X, mCoord_Y, r, g, b, CLOCK_50);

wire [9:0] gray = (mCoord_X<80 || mCoord_X>560? 10'h000:
    (mCoord_Y/15)<<5 | (mCoord_X-80)/15);

wire s = SW[0];
assign mVGA_R = (s? gray: r);
assign mVGA_G = (s? gray: g);
assign mVGA_B = (s? gray: b);

vga_sync u1(
```

```
.iCLK(VGA_CTRL_CLK),  
.iRST_N(DLY_RST&KEY[0]),  
.iRed(mVGA_R),  
.iGreen(mVGA_G),  
.iBlue(mVGA_B),  
// pixel coordinates  
.px(mCoord_X),  
.py(mCoord_Y),  
// VGA Side  
.VGA_R(VGA_R),  
.VGA_G(VGA_G),  
.VGA_B(VGA_B),  
.VGA_H_SYNC(VGA_HS),  
.VGA_V_SYNC(VGA_VS),  
.VGA_SYNC(VGA_SYNC),  
.VGA_BLANK(VGA_BLANK)  
);
```

```
endmodule
```

[illegible]

40

41

[illegible]

```
        else if (i == 64*48 - 64)
        begin
        // Bottom-left square
        population <= cells_preset[i - 1] +
                                cells_preset[i - 64] +
                                cells_preset[i - 64 + 1] +
                                cells_preset[i + 64 - 1] +
                                cells_preset[i + 1] +
                                cells_preset[0 + 64 - 1] +
                                cells_preset[0] +
                                cells_preset[0 + 1];
        end

        else if (i == 64*48 - 1)
        begin
        // Bottom-right square
        population <= cells_preset[i - 64 - 1] +
                                cells_preset[i - 64] +
                                cells_preset[i - 64 - 64 + 1]
+
                                cells_preset[i - 1] +
                                cells_preset[i - 64 + 1] +
                                cells_preset[0 + 64 - 1 - 1]
+
                                cells_preset[0 + 64 - 1] +
                                cells_preset[0];
        end

        else if (i < 63)
        begin
        // Top row
        population <= cells_preset[64*48 - 64 + i - 1] +
                                cells_preset[64*48 - 64 + i]
+
                                cells_preset[64*48 - 64 + i
+ 1] +
                                cells_preset[i - 1] +
                                cells_preset[i + 1] +
                                cells_preset[i + 64 - 1] +
                                cells_preset[i + 64] +
                                cells_preset[i + 64 + 1];
        end

        else if (i > 64*48 - 64)
        begin
        // Bottom row
        population <= cells_preset[i - 64 - 1] +
                                cells_preset[i - 64] +
                                cells_preset[i - 64 + 1] +
                                cells_preset[i - 1] +
                                cells_preset[i + 1] +
                                cells_preset[0 + i - 1] +
                                cells_preset[0 + i] +
                                cells_preset[0 + i + 1];
        end

        else if (i % 64 == 0)
        begin
        // Leftmost column
        population <= cells_preset[i - 1] +
```

```

                                cells_preset[i - 64] +
                                cells_preset[i - 64 + 1] +
                                cells_preset[i + 64 - 1] +
                                cells_preset[i + 1] +
                                cells_preset[i + 64 + 64 - 1]
+
                                cells_preset[i + 64] +
                                cells_preset[i + 64 + 1];

                                end

                                else if ((i + 1) % 64 == 0)
                                begin
                                // Rightmost column
                                population <= cells_preset[i - 64 - 1] +
                                cells_preset[i - 64] +
                                cells_preset[i - 64 - 64 + 1]
+
                                cells_preset[i - 1] +
                                cells_preset[i - 64 + 1] +
                                cells_preset[i + 64 - 1] +
                                cells_preset[i + 64] +
                                cells_preset[i + 1];

                                end

                                else
                                begin
                                // Middle squares
                                population <= cells_preset[i - 64 - 1] +
                                cells_preset[i - 64] +
                                cells_preset[i - 64 + 1] +
                                cells_preset[i - 1] +
                                cells_preset[i + 1] +
                                cells_preset[i + 64 - 1] +
                                cells_preset[i + 64] +
                                cells_preset[i + 64 + 1];

                                end

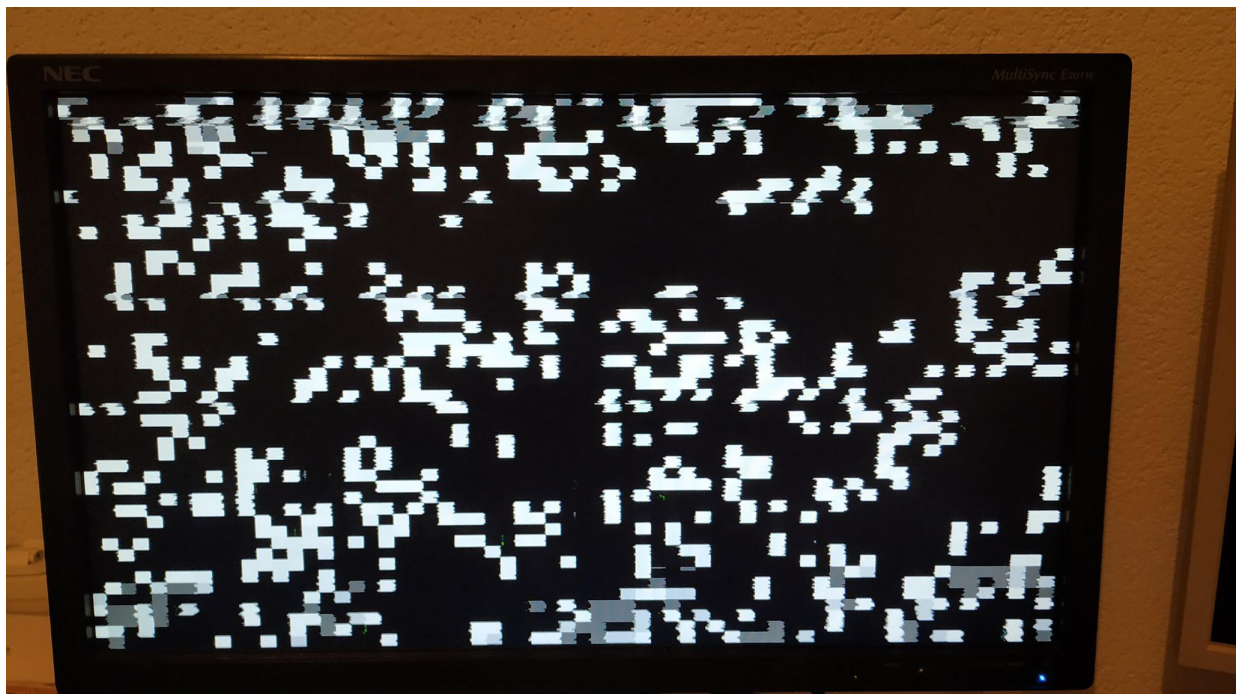
                                cells[i] <= ((cells_preset[i] & (population == 2)) |
                                (population == 3));

                                if(i==64*48)
                                begin
                                    i <= 0;
                                    cells_preset = cells;
                                end
                                else
                                    i <= i + 1;
                                end
                                end

                                always @(x or y)
                                begin
                                    if (cells[(y / 10) * 64 + (x / 10)])
                                        RGB = 30'b11111111111111111111111111111111;
                                    else
                                        RGB = 30'b00000000000000000000000000000000;
                                    end
                                end

```

```
    assign r = RGB[29:20];  
    assign g = RGB[19:10];  
    assign b = RGB[9:0];  
  
endmodule  
  
module clock(Clk, Clk_50);  
    input Clk_50;  
    output Clk;  
    reg [31:0] counter;  
  
    always @(posedge Clk_50)  
    begin  
        if (counter == 32'd1600)  
            counter <= 0;  
        else  
            counter <= counter + 32'd1;  
        end  
    assign Clk = (counter == 32'd1600);  
Endmodule
```



Slika 15. Implementacija Game of Life na FPGA u Verillgu

Vrijeme potrebno za generisanje 1000 generacija na FPGA

Proracun vremena za realizaciju 1.000 generacija na FPGA:

$$64 \cdot 48 = 3.072$$

Potrebno clock ciklusa za iscrtavanje 1 slike

Dakle

Za iscrtavanje 1.000 slika potrebno:

$$3.072 \cdot 1000 \text{ ciklusa}$$

Sto za integrirani clock od 50MHz predstavlja 16.276 ti dio sekunde

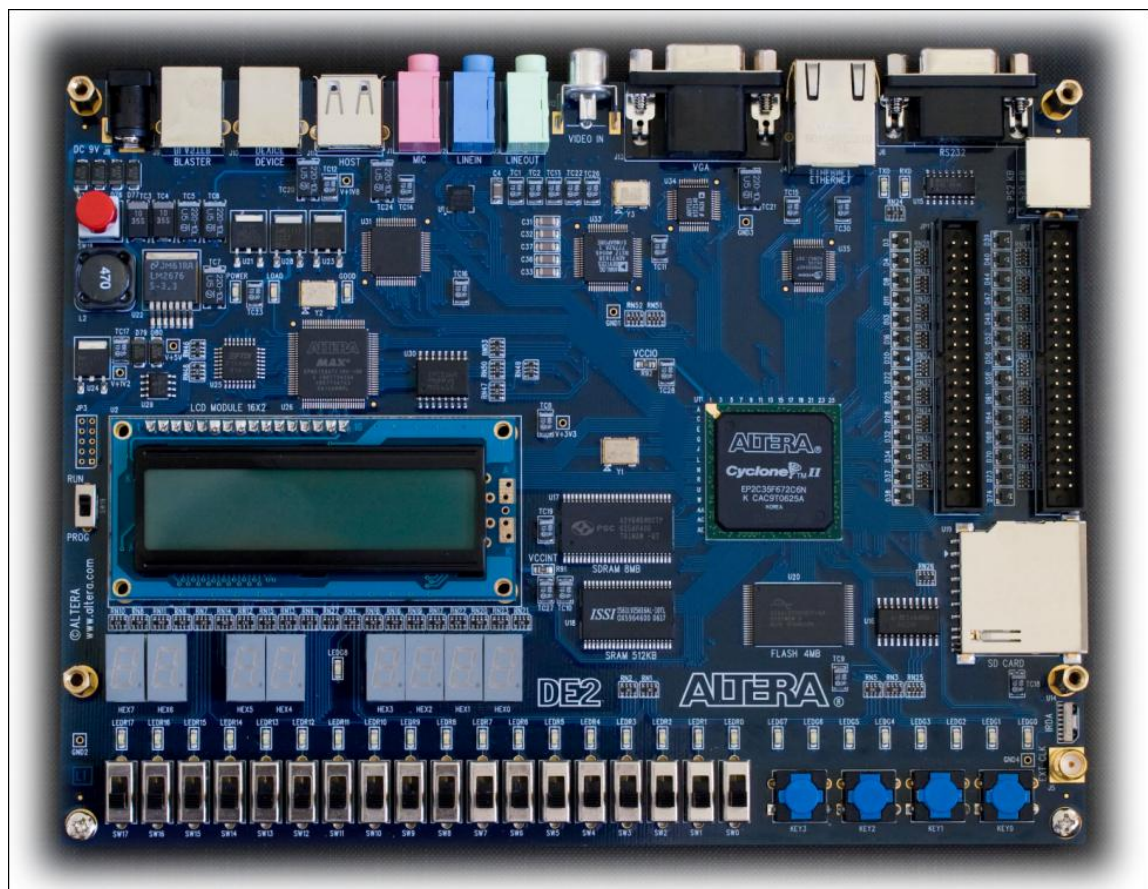
Medjutim, za iscrtavanje na ekranu smo ograniceni integrisanim clockom od 25MHz, te bi kranji proracun bio:

1/8.138 dio sekunde za generisanje 1000 generacija

Tj.

0.123 sekundi za generisanje 1000 generacija

Altera DE2 Cyclone II



Slika 16. koristeni development board: DE2 Cyclone II

Total logic elements	13,347 / 33,216 (40 %)
Total combinational functions	12,195 / 33,216 (37 %)
Dedicated logic registers	6,250 / 33,216 (19 %)
Total registers	6250
Total pins	215 / 475 (45 %)
Total virtual pins	0
Total memory bits	0 / 483,840 (0 %)
Embedded Multiplier 9-bit elements	0 / 70 (0 %)
Total PLLs	1 / 4 (25 %)

Slika 17. specifikacije boarda kao I potrebni resursi za koristeni program

PINOUT

Signal Name	FPGA Pin No.
SW[0]	PIN_N25
SW[1]	PIN_N26
SW[2]	PIN_P25
SW[3]	PIN_AE14
SW[4]	PIN_AF14
SW[5]	PIN_AD13
SW[6]	PIN_AC13
SW[7]	PIN_C13
SW[8]	PIN_B13
SW[9]	PIN_A13
SW[10]	PIN_N1
SW[11]	PIN_P1
SW[12]	PIN_P2
SW[13]	PIN_T7
SW[14]	PIN_U3
SW[15]	PIN_U4
SW[16]	PIN_V1
SW[17]	PIN_V2

Slika 18. pinout switcheva

Signal Name	FPGA Pin No.	Description
CLOCK_27	PIN_D13	On Board 27 MHz
CLOCK_50	PIN_N2	On Board 50 MHz
EXT_CLOCK	PIN_P26	External Clock

Slika 19. pinout clock-ova

Signal Name	FPGA Pin No.
VGA_R[0]	PIN_C8
VGA_R[1]	PIN_F10
VGA_R[2]	PIN_G10
VGA_R[3]	PIN_D9
VGA_R[4]	PIN_C9
VGA_R[5]	PIN_A8
VGA_R[6]	PIN_H11
VGA_R[7]	PIN_H12
VGA_R[8]	PIN_F11
VGA_R[9]	PIN_E10
VGA_G[0]	PIN_B9
VGA_G[1]	PIN_A9
VGA_G[2]	PIN_C10

VGA_G[3]	PIN_D10
VGA_G[4]	PIN_B10
VGA_G[5]	PIN_A10
VGA_G[6]	PIN_G11
VGA_G[7]	PIN_D11
VGA_G[8]	PIN_E12
VGA_G[9]	PIN_D12
VGA_B[0]	PIN_J13
VGA_B[1]	PIN_J14
VGA_B[2]	PIN_F12
VGA_B[3]	PIN_G12
VGA_B[4]	PIN_J10
VGA_B[5]	PIN_J11
VGA_B[6]	PIN_C11
VGA_B[7]	PIN_B11
VGA_B[8]	PIN_C12
VGA_B[9]	PIN_B12
VGA_CLK	PIN_B8
VGA_BLANK	PIN_D6
VGA_HS	PIN_A7
VGA_VS	PIN_D8
VGA_SYNC	PIN_B7


Slika 20. pinout za VGA

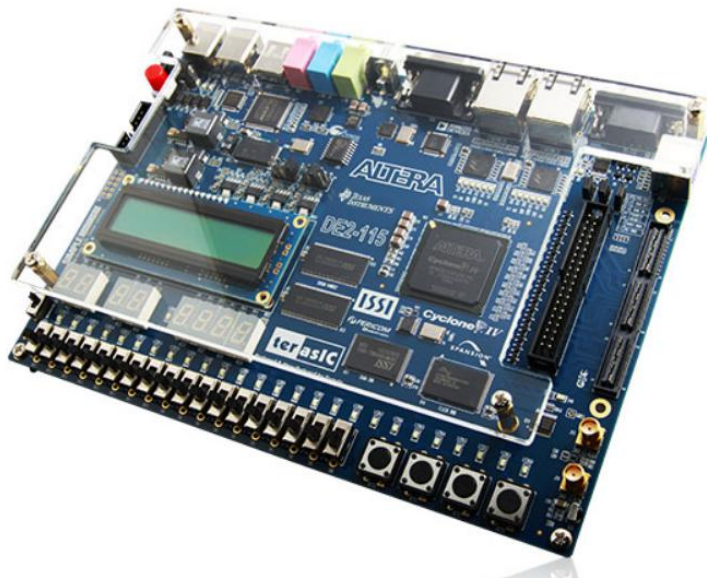
Cijena

Ovaj board trenutno nije u prodaji, uzet je iz uporabe prije nekoliko godina
Altera ne podrzava vise
Quartus 13.0 zadnji koji podrzava

Trenutno ekvivalent:
Altera DE2-115 Cyclone II

Altera DE2-115 Development and Education Board

 Svida mi se sljedeći broj osoba kaže da im se ovo sviđa: 22. Budite prvi među svojim prijateljima.



(Currency: USD)

Price: \$595

Academic: \$309

[Buy it now](#)

Slika 21. DE2-115 cyclone IV I cijena

Reference

- [1] https://en.wikipedia.org/wiki/Cellular_automaton
- [2] https://en.wikipedia.org/wiki/Conway%27s_Game_of_Life
- [3] <https://www.youtube.com/playlist?list=PL2E0D05BEC0140F13>
- [4] <http://people.ece.cornell.edu/land/courses/ece5760/FinalProjects/>
- [5] http://www.terasic.com.tw/attachment/archive/30/DE2_Pin_Table.pdf
- [6] <https://timetoexplore.net/blog/artty-fpga-vga-verilog-01>
- [7] <https://ktln2.org/2018/01/23/implementing-vga-in-verilog/>