SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

ELEKTROTEHNIČKI FAKULTET OSIJEK

Sveučilišni studij

DIZAJN VIDEO UPRAVLJAČKOG SKLOPA POMOĆU FPGA

Završni rad

Sven Pothorski

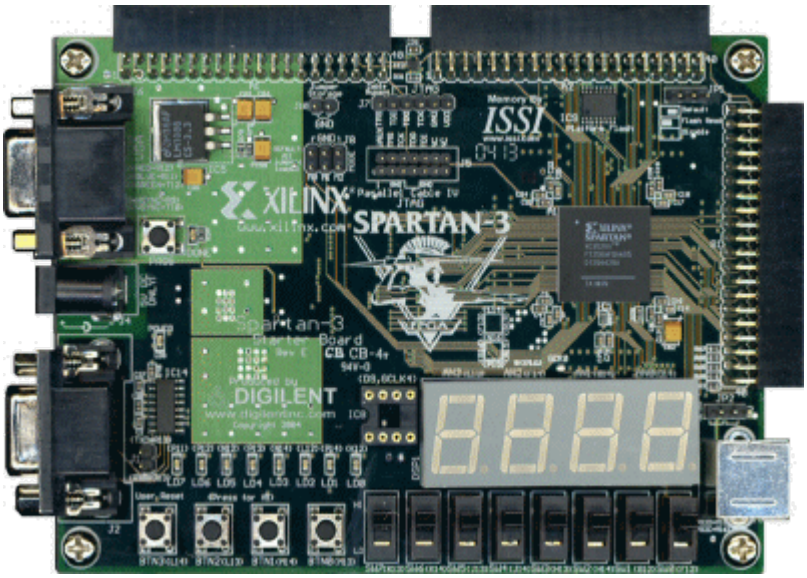
Osijek, 2010.

SADRŽAJ

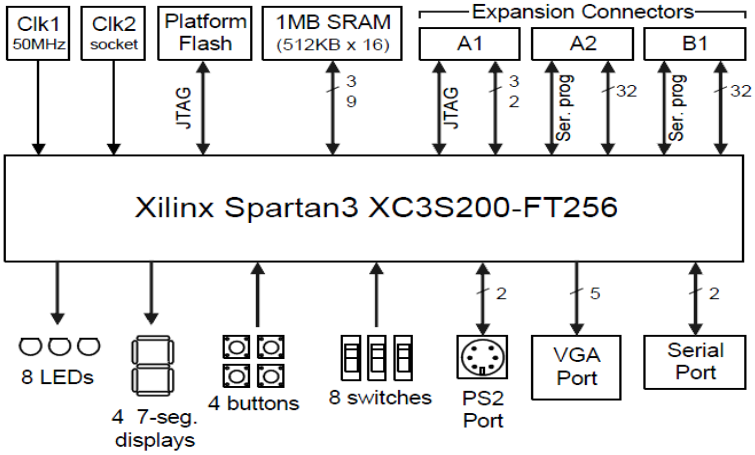
1. UVOD

Cilj ovog završnog rada jest izraditi VGA kontroler s kojim se odabire rezolucija i frekvencija osvježavanja monitora. S pomoću prekidača na razvojnoj pločici Xilinx Spartan 3 odabire se željeni način rada. Ovisno o modu rada definirana je rezolucija i frekvencija monitora. Na monitor je potrebno prikazati testni uzorak s kojim se pokazuje da je kontroler ispravno dizajniran.

Za rad se koristi maketa Xilinx Spartan 3 Starter XC3S200FT256 FPGA koji sadrži 200000 vrata, 4320 logičkih ćelija, 12 memorijskih RAM blokova od 18 Kbita, 12 hardverskih množitelja 18x18



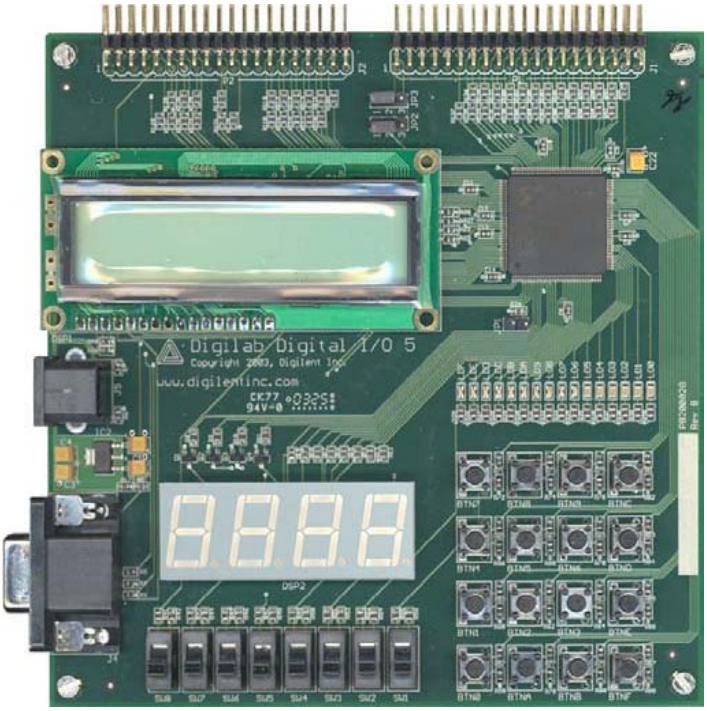
Slika 1.1 Xilinx Spartan 3 Starter razvojna pločica



Slika 1.2 Xilinx Spartan 3 Starter blok dijagram

Programski kod je pisan u VHDL-u pomoću programa Xilinx WebISE. Programiranje i debbugiranje se vrši preko JTAG paralelnog kabla kojim se spaja računalo sa Spartan 3 razvojnom pločicom.

Xilinx Spartan 3 Starter se koristi za razna istraživanja i razvoj. Pomoću ove pločice možemo napraviti bilo kakav sustav s ulazima i izlazima, tj. možemo spojiti na njega tipkovnicu i miša, CRT ili LCD monitor. Pločica je proširiva, pa preko dodatnih konektora možemo na nju prispojiti i neki drugi dio, kao npr. Digital I/O 5 koji sadrži VGA, PS/2 port i 2x16 LCD, ili nešto što smo možda sami kreirali ili poboljšali već postojeće.



Slika 1.3 Digilab Digital I/O 5 modul

2. FPGA I VHDL

FPGA (Field-programmable gate array) je poluvodički uređaj koji se sastoji od programabilnih logičkih komponenti koji su nazvani „logički blokovi“ i programabilnih spojeva. Logički blokovi se mogu programirati za izvođenje funkcija osnovnih logičkih sklopova kao što su I i XILI, te složenijih kombinacijskih sklopova kao što su dekoderi ili matematičke funkcije. U većini FPGA, logički blokovi također sadrže memorijske elemente, koji mogu biti jednostavni bistabili ili mnogo složeniji blokovi memorije.
Hijerarhijski programabilni spojevi da se logički blokovi spajaju kako ih želi spojiti sistemski dizajner, nešto kao jedno-čipni razvojni sustav. Logički blokovi i spojeve mogu programirati korisnici ili dizajneri, FPGA-ovi se izrađuju za implementaciju bilo koje logičke funkcije – u daljnjem tekstu „field-programmable“.

FPGA-ovi su uobičajeno sporiji nego specifični primjenski integrirani krugovi –ASIC, ne mogu obraditi kompleksni dizajn, konzumiraju više snage (za bilo koji poluvodički proces). Njihove prednosti su kraće vrijeme isporuke na tržište, mogućnost ispravaka pogrešaka u dizajnu, i niži inženjerski troškovi. Dobavljači mogu prodavati jeftinije manje fleksibilne verzije njihovih FPGA-ova čiji dizajn se ne može mijenjati nakon programiranja. Dizajn je izražen na običnom FPGA-u i onda premješten u statičnu verziju koja naliči ASIC-u.

„Complex Programmable Logic Device“ (CPLD-ovi) su zamjena za jednostavnije dizajne. Oni također ostaju programirani bez obzira na to da li su spojeni na izvor snage.

Kako bismo konfigurirali („programirali“) FPGA ili CPLD možemo odlučiti na koji način ćemo raditi sa dijagramom logičkog sklopa ili izvornim kodom korištenjem jezika za opis hardvera (HDL – Hardware Description Language). HDL oblik je jednostavniji za rad sa velikim strukturama jer je moguće definirati svaki dio numerički, a ne moramo ga crtati rukom. S druge strane, shematski prikaz može pružiti bliskije specifikacije onoga što očekujemo.

Prijelaz iz sheme/HDL izvornih datoteka u stvarnu konfiguraciju: Izvorne datoteke se dostavljaju softverskom paketu od proizvođača FPGA/CPLD i kroz nekoliko različitih koraka nastaje datoteka. Ova datoteka se prenosi na FPGA/CPLD putem serijskog sučelja (JTAG) ili vanjskih memorijskih uređaja kao što su EEPROM-ovi.

2.1. Povijest FPGA

FPGA vuče svoje korijenje iz CPLD-ova (CPLD – Complex Programmable Logic Devices) ranih 1980ih. Suosnivač Xilinx-a, Ross FreeMan je otkrio FPGA 1984. Godine. CPLD-ovi i FPGA-ovi sadrže relativno veliki broj programabilnih logičkih elemenata. Gustoće CPLD logičkih sklopova se kreću od nekoliko do stotinu tisuća logičkih sklopova, dok je kod FPGA to tipično od desetak tisuća do nekoliko milijuna.

Glavna razlika između CPLD-a i FPGA-a je njihova arhitektura. CPLD ima nešto ograničenu strukturu koja se sastoji od jednog ili više programabilnih logičkih sklopova „suma umnožaka“ koji pune relativno mali broj taktnih registara. Ovo ima za posljedicu nižu fleksibilnost, sa prednošću predvidljivijeg vremena kašnjenja i višeg „logic-to-interconnect“ omjera. S druge strane u FPGA arhitekture dominiraju spojevi. Ovo ih čini mnogo fleksibilnijima (u pogledu raspona dizajna koji su praktični za implementaciju u njima), ali su također kompleksniji za dizajn.

Druga važna razlika između CPLD-a i FPGA-a je u postojanje viših ugrađenih funkcija (zbrajala i množitelji) i više ugrađene memorije kod FPGA-a.

Neki FPGA-ovi imaju mogućnost djelomične rekonfiguracije što omogućuje preprogramiranje jednog dijela dok drugi dio radi.

2.2. Moderni razvoj FPGA

Trenutni trendovi razvoja su korištenje krupnozrnog arhitekturnog pristupa korak dalje kombiniranjem logičkih blokova i spojeva tradicionalnih FPGA-ova s ugroženim mikroprocesorima i popratnim periferijama radi formiranja potpunog „sustava na programabilnom čipu“. Primjeri tih hibridnih tehnologija mogu se naći u Xilinx Virtex-II PRO i u Virtex-4 koji sadrže jedan ili više PowerPC procesora ugrađenih unutar FPGA logike. Atmel FPSLIC je drugi takav uređaj koji koristi AVR procesor u kombinaciji sa Atmelovom programabilnom logičkom arhitekturom.

Alternativni pristup korištenju ovih hard-macro procesora je korištenje „mekih“ procesorskih jezgri implementiranih unutar FPGA logike razjašnjene kasnije u „Soft processors“.
Kao što je prije spomenuto mnogi moderni FPGA-ovi imaju mogućnost reprogramiranja tijekom izvođenja i ovo vodi do ideje rekonfigurabilne obrade ili rekonfigurabilnih sustava – CPU koji rekonfigurira sam sebe kako bi ugodio trenutnom zadatku. Mitrion Virtual Processor od Mitronica je primjer rekonfigurabilnih mekih procesora koji su implementirani u FPGA-u. Ipak on ne podržava dinamičku rekonfiguraciju pri izvođenju, ali ipak se prilagođava specifičnom programu.
Usput nove ne-FPGA arhitekture se pojavljuju. Softverski-konfigurabilni procesori kao što su Strech S5000 se prilagođavaju hibridnom pristupu pružanjem niza procesorskih jezgri i programabilnih jezgri sličnih FPGA-u na istom čipu.

2.3. Primjene FPGA

Primjene FPGA-a uključuju obradu digitalnog signala, softverski definiran radio, sustavi svemirskih letova i obrambeni sustavi, ASIC prototipiranje, medicinsko slikovna dijagnostika, računalni vid, prepoznavanje glasa, kriptografija, bioinformatika, emulacija računalnog hardvera i rastući raspon drugih područja. FPGA je započeo kao konkurencija CPLD-u i konkurirao je u sličnom području, onoj „glue logic“ za PCB-ove. Kako su njegova veličina, mogućnosti, i brzina rasli, počeo je zauzimati sve veće i veće funkcije do stanja kada se predstavlja kao cijeli sustav na čipu (SOC).
FPGA posebno nalazi primjenu u području ili algoritmima gdje se može koristiti masivni paralelizam koji pruža njegova arhitektura. Jedno od tih područja je razbijanje kodova u napadima grubom snagom (brute-force) ili kriptografskim algoritmima.
FPGA-ovi se povećano koriste u konvencionalnim primjenama visokih performansi gdje se koriste kerneli za računanje kao što su kod FFT (Fast Fourier Transformation) ili Konvolucije gdje se umjesto mikroprocesora koriste FPGA-ovi. Korištenje FPGA-a za zadatke obrade je znano i kao rekonfigurabilna obrada.
Svojstveni paralelizam logičkih resursa na FPGA pruža značajnu moć obrade čak i ispod takta od 500 MHz. Na primjer, trenutne generacije FPGA mogu implementirati oko 100 jedinica jednostruke preciznosti sa tekućim zarezom, gdje svaka može izračunati rezultat svaki signal takta. Fleksibilnost FPGA dozvoljava i više performanse odbacivanjem preciznosti i ranga

brojevnog formata za dobivanje i većeg broja paralelnih aritmetičkih jedinica. Ovo dovodi do novog tipa obrade zvanog rekonfigurabila obrada, gdje se vremenski intenzivne zadaće prebacuju sa softvera na FPGA.

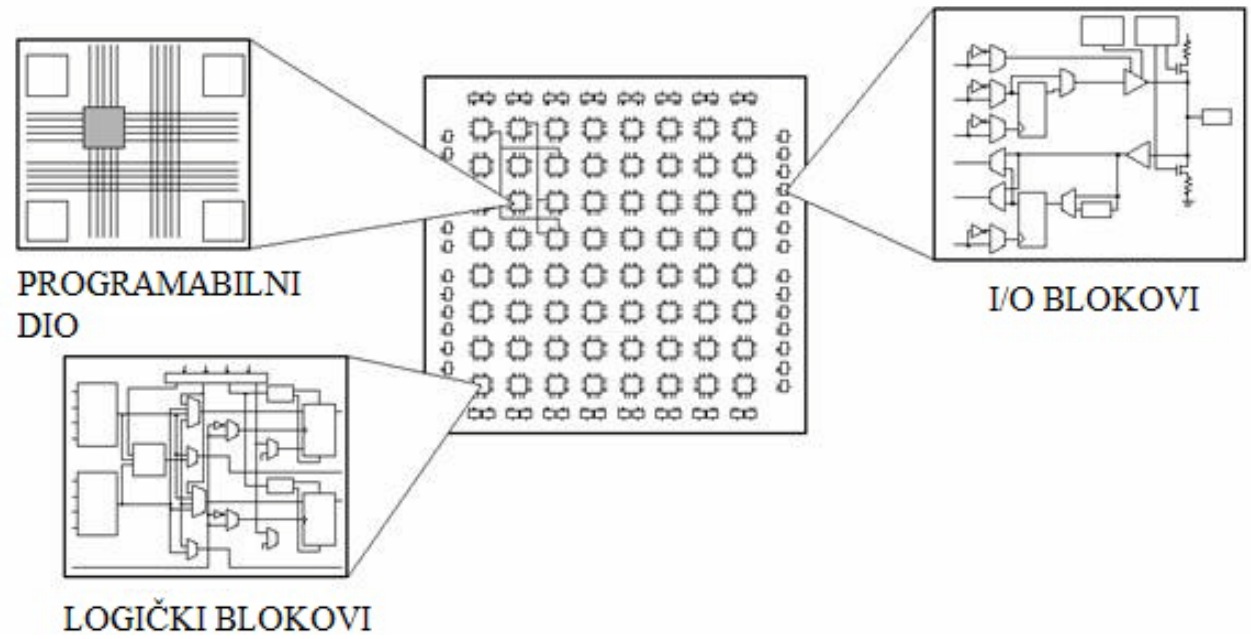
Prihvaćanje FPGA-a u obradi visokih performansi je trenutno ograničeno kompleksnošću FPGA dizajna u odnosu na konvencionalni softver i ekstremno visokog perioda čekanja obrade trenutnih alata za dizajn, gdje je čekanje od 4-8 sati potrebno za male promjene u izvornom kod

2.4. Arhitektura FPGA

Tipična osnovna arhitektura se sastoji od polja konfigurabilnih logičkih blokova i kanala za usmjeravanje. Višestruka I/O podloga može stati u visinu jednog reda ili širini jednog stupca u polju. Općenito, svi kanali usmjeravanja imaju istu širinu (broj žica).

Jedna primjena kruga mora se smjestiti u FPGA na adekvatni resurs.

Klasični FPGA logički blok se sastoji od 4-ulazne „lookup“ tabele (LUT) i bistabila. Nedavno proizvođači su se počeli prebacivati na 6-ulazne LUT-ove u svojim komponentama visokih performansi, tražeći rast performansi.



Slika 2.1. - Arhitektura FPGA

Ima samo jedan izlaz koji se može smjestiti u LUT preko registra ili bez registra. Logički blok ima četiri ulaza za LUT i ulaz za takt. Kako su signali takta (i često drugi signali sa visokim

stupnjem grananja fanouta) uobičajeno usmjereni preko namjenskih usmjerivačkih mreža posebne namjene u komercijalnim FPGA-ovima, oni i drugi signali su posebno upravljani.
Svakom ulazu se može pristupiti sa druge strane logičkog bloka, dok se izlazu može prispojiti preusmjeravanjem žica na kanal desno ili na kanal ispod logičkog bloka.
Svaki pin izlaza logičkog bloka se može spojiti svakom žičnom segmentu u susjednom kanalu.
Slično se U/I podložak može spojiti bilo kojem žičnom segmentu u susjednom kanalu. Na primjer jedan U/I podložak na vrhu čipa se može spojiti na bilo koji od W žica (gdje W je širina kanala) u horizontalnom kanalu odmah ispod.
Općenito FPGA prespajanje je ne segmentirano. Svaki žični segment ima raspon od samo jednog logičkog bloka prije nego što se zatvori u preklopniku. Uključivanjem nekog od programabilnih prekidača unutar preklopnika mogu se stvoriti duži putevi. Zbog bržeg prespajanja neke FPGA arhitekture koriste duge prespojne linije koje obuhvaćaju više logičkih blokova.
Neovisno da li se horizontalni ili vertikalni kanali presijecaju, postoji preklopnik. U ovoj arhitekturi kada žica uđe u preklopnik postoje tri programabilna prekidača koja joj dopuštaju spajanje sa tri druge žice u susjednom kanalnom segmentu. Ovaj model ili topologija preklopnika koja se koristi u arhitekturi je planarna ili bazirana na području preklopnika. U ovoj preklopničkoj topologiji žica u kolosijeku jedan spaja se jedino sa žicama koje su susjedne tom segmentu, žice u kolosijeku 2 spajaju se samo sa drugim žicama susjednim kolosijeku 2 itd.

Moderne obitelji FPGA koriste mogućnosti objašnjene gore kako bi uključile viši nivo funkcionalnosti ubačenog u silicij. Imajući ove česte funkcije ugrađene u silicij smanjuju potrebno područje i daju tim funkcijama veću brzinu nasuprot onima izgrađenim od osnovnih blokova. Primjeri uključuju množitelje, generičke DSP blokove, ugrađene procesore i brze U/I logike, te ugrađene memorije.
FPGA-ovi su često korišteni u sustavima kvalitete uključujući pre-silikonsku provjeru, post-silikonsku provjeru i razvoj firmvera. Ovo pruža tvrtkama koje razvijaju čipove da provjere svoj dizajn prije nego što je čip proizveden u tvornici, skraćujući vrijeme pristupa tržištu.

2.5. FPGA dizajn i programiranje

Kako bi definirali ponašanje FPGA korisnik pruža dizajn u shemi ili opis pomoću jezika za opis hardvera (HDL). Česti HDL-ovi su VHDL i Verilog. Korištenjem alata za automatizaciju

elektroničkog dizajna generira se mreža lista tehnološkog mapiranja. Mreža lista se onda može smjestiti u aktualnu FPGA arhitekturu korištenjem „place-and-route procesa“, najčešće korištenjem odgovarajućeg „place-and-route“ softvera kojeg isporučuje FPGA tvrtka. Korisnik provjerava mapiranje, smještanje i rutiranje rezultata pomoću vremenske analize, simulacije i drugih verifikacijskih metodologija. Jednom kada je završen proces dizajna i provjere generira se binarna datoteka (također korištenjem vlastitog FPGA tvrtkinog softvera) za konfiguriranje FPGA-a.

Kako bi se smanjila kompleksnost dizajniranja HDL-a koji se smatra gotovo ekvivalentnim asemblerskom jezik, postoje naznake podizanja nivoa apstrakcije u dizajnu. Tvrtke kao što su „Cadence“, „Synopsys“ i „Celoxica“ predstavljaju „SystemC“ kao način kombiniranja jezika visokog nivoa sa konkurentnim modelima za dobivanje bržeg razvoja FPGA nego što je to moguće sa tradicionalnim HDL-ovima. Pristupi stvoreni na standardnom C-u ili C++-u (sa bibliotekama ili drugim ekstenzijama koje pružaju paralelno programiranje) mogu se naći u „Catapult C“ alatima od „Mentor Graphicsa“, i u „Impulse C“ alatima od „Impulse Accelerated Technologies“. „Annapolis Micr Systems, Inc“. „CoreFire Design suite“ i „National Instruments LabVIEW FPGA“ pružaju grafički pristup toku podataka za pristup dizajnu više razine. Jezici kao što su „SystemVerilog“, „SystemVHDL“ i „Handel-C“ (od Celoxica) pokušavaju postići isti cilj, ali ciljaju postići postojećim hardverskim inženjerima veću produktivnost nasuprot pristup postavljanja FPGA-a pristupačnijim postojećim softverskim inženjerima.

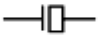
Kako bi pojednostavili dizajn kompleksnih sustava u FPGA-ovima postoje biblioteke kompleksnih predefinirani funkcija i sklopova koji su testirani i optimizirani kako bi ubrzali proces dizajna. Ovi predefinirani sklopovi su često zvani IP jezgre, i moguće ih je nabaviti od isporučitelja FPGA i neovisnih dobavljača (rijetko besplatnih, tipično licencirani s vlasničkim uvjetima). Ostali predefinirani sklopovi su dobavljivi od razvojnih zajednica kao što su „OpenCores“ (tipično besplatni, i objavljeni pod GPL, BSD ili sličnim licencama), i drugih izvora.

Tipični tok razvoja, razvijatelj FPGA aplikacija simulira dizajn u više stadija tokom razvojnog procesa. Inicijalno RTL opis u VHDL ili Verilogu se simulira stvaranjem test bencheva simuliranje sustava i promatranjem rezultata. Nakon što je „engine“ sintetizirao dizajn u netlistu, netlista se pretvara na nivou opisa logičkih sklopova gdje se simulacija ponavlja kako bi se potvrdila da je sinteza završila bez pogrešaka. Napokon dizajn je stavljen u FPGA iz odakle se

mogu dodati propagacijska kašnjenja i možemo nanovo simulirati ponovno sa tim vrijednostima vraćenim u bilješku netliste.

2.6. Osnovni tipovi FPGA procesne tehnologije

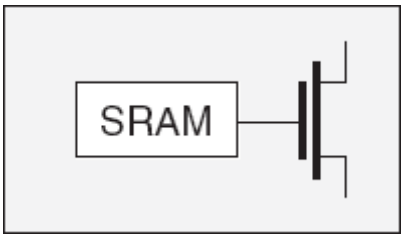
3. SRAM – bazirano na tehnologiji statičke memorije. U sistemu programabilne i reprogramabilne. Zahtjeva vanjski uređaj za učitavanje. CMOS
4. Antifuse – jedanput programabilni. CMOS
5. EPROM – „Erasable Programmable Read-Only Memory“ tehnologija. Uobičajeno jedanput programabilno u proizvodnji zbog plastičnog pakiranja. Uređaji sa prozorom mogu biti se izbrisati korištenjem ultraljubičastog (UV) zračenja. CMOS
6. EEPROM – „Electrically Erasable Programmable Read-Only Memory“ tehnologija. Može se izbrisati, i u plastičnom pakiranju. Nekad, ali ne uvijek EEPROM uređaji mogu se programirati u sustavu. CMOS.
7. Flash – „Flash“ - izbrisiva EPROM tehnologija. Može se izbrisati i u plastičnom pakiranju. Neki ali ne svi „flash“ uređaji mogu se programirati u sustavu. Uobičajeno, „flash“ ćelije su manje nego ekvivalentne EEPROM ćelije i jeftinije su za proizvesti. CMOS
8. Fuse – Jedanput programabirljive. Bipolarne.

Tehnologija	Simbol	Primjena
Fusible-link		SPLD
Antifuse		FPGA
EPROM		SPLD i CPLD
E^2PROM/ FLASH		SPLD i CPLD (neki FPGA-ovi)
SRAM		FPGA (neki CPLD-ovi)

Slika 2.2. - Pregled tehnologija izrade programabilnih logičkih sklopova i njihovih simbola.

2.6.1. SRAM tehnologija

SRAM tehnologija koristi male SRAM ćelije za svaki programabilni element. Kada se vrijednost učita u SRAM ćeliju, ostaje nepromijenjena sve dok se ponovnim programiranjem ne promijeni ili dok se ne prekine napajanje sustava.
SRAM ćelija se sastoji od višetranzistorskog SRAM memorijskog elementa čiji izlaz upravlja dodatnim upravljačkim tranzistorom. Ovisno o sadržaju pohranjenom u memorijskom elementu (logička „0“ ili logička „1“), upravljački tranzistor će biti uključen ili isključen.



Slika 2.3. - Izgled SRAM ćelije u programabilnim logičkim sklopovima

Nedostaci SRAM tehnologije:

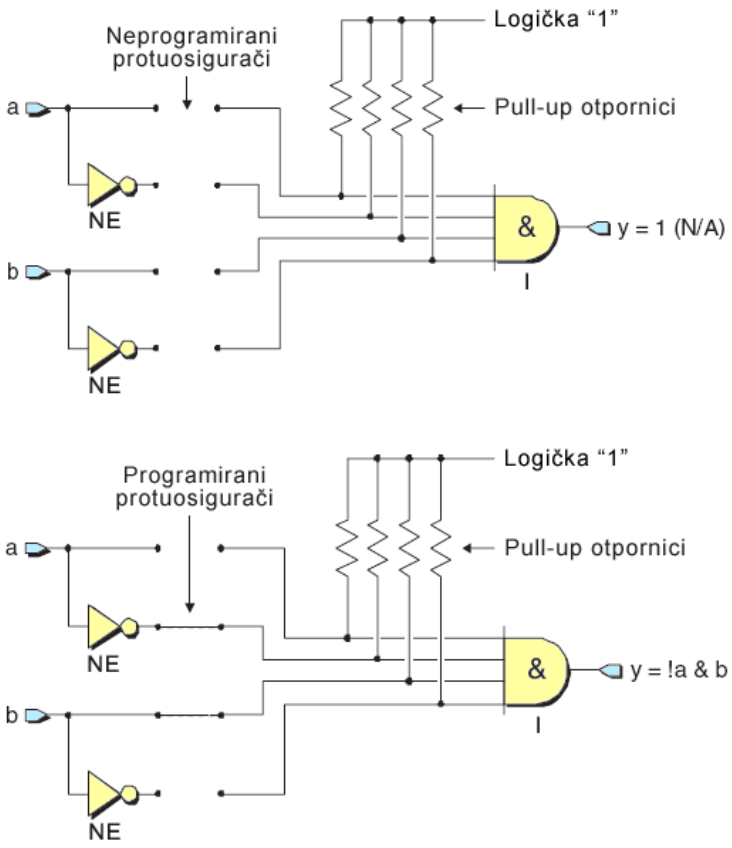
1. Zauzeće velikog prostora (ćelije se sastoje od 4-6 tranzistora)
2. Gubitak podatak u slučaju prekida napajanja
3. Osjetljivost na male naponske promjene
4. Velika kašnjenja zbog rutiranja

Prednosti SRAM tehnologije:

1. Za proizvodnju FPGA-ova mogu se koristiti standardni proizvodni procesi
2. FPGA-vi se mogu reprogramirati neograničen broj puta.

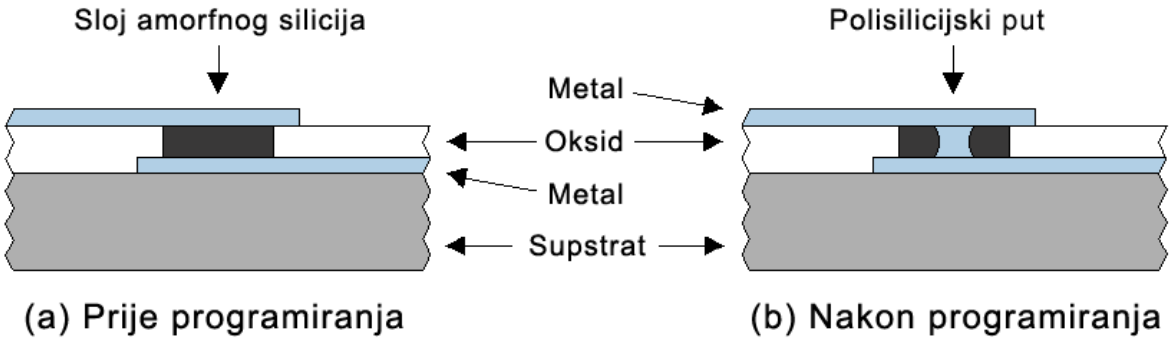
2.6.2. Anti-fuse tehnologija

Koristi mikroskopske strukture koje, za razliku od osigurača, ne tvore vezu. U neprogramiranom stanju „anti-fuse“ element ima vrlo veliki otpor koji se može predstaviti otvorenim krugom, odnosno prekidom veze. Prilikom programiranja, dovođenjem impulsa visokog napona i struje na ulaze sklopa, vrši se stvaranje veze, tj. spajanje krajeva „anti-fuse“ elemenata.



Slika 2.4. Prikaz anti-fuse tehnologije u stanju neprogramiranosti i programiranosti.

„Anti-fuse“ element je sloj amorfnog (nekristaliziranog) silicija koji spaja dva metalna sloja. U neprogramiranom stanju, silicij se ponaša kao izolator sa vrlo visokim otporom (reda veličine 1 milijun ohma). Postupkom programiranja vrši se stvaranja veze, tj. puta pretvorbom silicija u vodljivi polisilicij.



Slika 2.5. Prikaz izgleda sklopa izrađenog „anti-fuse“ tehnologijom prije i poslije programiranja

Prednosti „anti-fuse“ tehnologije:

1. Neosjetljivost
2. Mala kašnjenja koja rezultiraju bržim radom

Nedostaci „anti-fuse“ tehnologije:

1. Zahtijevaju složeni proizvodni proces
2. FPGA-ovi zahtijevaju poseban uređaj za programiranje
3. FPGA-ovi se mogu programirati samo jednom

2.7. Proizvođači i njihove posebnosti

Postoje dva glavna proizvođača FPGA-ova „opće namjene“ i brojni drugi igrađi koji se razlikuju u nuđenju unikatnih mogućnosti.

1. Xilinx i Altera su trenutno vodeći na FPGA tržištu. Xilinx pruža besplatni Windows i Linux softver za dizajn, dok Altera pruža besplatne Windows alate, i Solaris i Linux alate putem pretplate
2. Lattice Semiconductor pruža oba SRAM i postojane, flash bazirane FPGA-ove.
3. Actel pruža antifuse i reprogramabilne flash bazirane FPGA-ove te također pruža mješanje signalne flash bazirane FPGA-ove
4. Atmel pruža sitno-zrne rekonfigurabilne uređaje, kao što su Xilinx XC62xx. Njihov glavni fokus je pružanje Atmel AVR mikrokontrolera sa FPGA jezgrama na istom komadu silicija.
5. QuickLogic ima antifuse (jednom programabirljive) proizvode čvrsto fokusirane na ručno upravljane primjene.
6. Achronix Semiconductor ima vrlo brze FPGA-ove u razvoju, koji se fokusiraju na brzinama blizu 2 GHz
7. MathStar pruža uređaje slične FPGA uređajima zvane FPOA (Field Programmable Object Array

2.8. VHSIC Hardware Description Language (VHDL)

VHDL (VHSIC hardware description language) je često korišten kao početni dizajnerski jezik za "Field-programmable gate array" (FPGA) i "Application-specific integrated circuits" (ASIC) u automatizaciji elektroničkog dizajna digitalnih krugova.

2.9. Povijest VHDL-a

VHDL je izvorno razvijen na zahtjev Američkog ministarstva obrane kako bi se dokumentiralo ponašanje ASIC-a koji su dobavljači uključivali u opremi. Takoreći VHDL je razvijen kao alternativa velikim, kompleksnim priručnicima koji su bili materija specifičnih pojedinosti implementacije.

Ideja da se može simulirati dokumentacija je bila tako zanimljiva da su razvijeni logički simulatori koji su mogli čitati VHDL datoteke. Slijedeći korak je razvoj alata za logičku sintezu koji čitaju VHDL, i daju definiciju fizičke implementacije sklopova. Moderni alati za sintezu mogu izdvojiti RAM, brojače i aritmetičke blokove iz koda, i implementirati ih na načina kako to korisnik odredi. Time isti VHDL kod možemo sintetizirati različito za jeftiniju cijenu, veću efikasnost potrošnje, veću brzinu ili neke druge zahtjeve.

VHDL uzima oba koncepta iz Ada programskog jezika (npr, dio za označavanje dijelova indeksa jednodimenzionalnih polja) i sintaksu. VHDL ima konstruktore koji barataju ugrađenim paralelizmom u hardverskom dizajnu, ali ti konstruktori (procesi) se razlikuju po sintaksi od paralelnih konstruktora u Adi (zadaće). Kao Ada, VHDL je čvrsto oblikovan i nije osjetljiv na velika i mala slova. Postoje mnoge mogućnosti VHDL-a koje ne postoje u Adi, poput proširenog seta Booleovih operatora uključujući NI i NILI, kako bi predstavio direktne operacije koje su česte kod hardvera. VHDL također dopušta indeksiranje polja u svakom smjeru (rastućem ili padajućem) zato što se obje konvencije koriste u hardveru, gdje Ada (kao i većina programskih jezika) pruža samo rastuće indeksiranje. Razlog sličnosti između ova dva jezika je zato što je Ministarstvo Obrane zahtijevalo što više moguće sintakse bazirane na Adi, kako bi izbjegli ponovno uvođenje koncepata koji su već bili testirani kod razvoja Ade.

Inicijalna verzija VHDL-a, dizajnirana za IEEE standarde 1076-1987, je uključivala široki raspon tipova podataka, uključujući numeričke („integer“ i „real“), logičke (bit i „boolean“, „character“ i „time“, plus polja bitova zvana „bit_vector“ i znakova zvana „string“).

Problem koji ova izvedba nije riješila, je „multi-valued logika“ gdje spadaju snaga vođenja signala (bez, slabi ili jaki) i nepoznate vrijednosti. Zahtijevani standard IEEE 1164, koji je definirao 9 tipova logičkih vrijednosti: skalarna „std_ulogic“ i njena vektorska verzija „std_ulogic_vector“.
Drugi izvedba IEEE 1076 sintakse 1993 je učinila sintaksu dosljednijom, dopustila više fleksibilnosti kod naziva, dodala znakovne tipove ISO-8859-1 printabilnih znakova, dodala EXNILI operator.
Manje promjene u standardu (2000 i 2002) dodale su ideju zaštićenih tipova (slično konceptu klasa u C++) i uklonile neka ograničenja iz pravila mapiranja portova.
Kao dodatak IEEE standardu 1164, nekoliko podstandarda su predstavljeni kako bi povećali funkcionalnost jezika. IEEE standard 1076.2 dodao je bolje upravljanje realnim i kompleksnim tipovima podataka. IEEE standard 1076. 3 predstavlja „signed“ (s predznakom) i „unsigned“ (bez predznaka) tipove kako bi olakšao aritmetičke operacije na vektorima. IEEE standard 1076.1 (poznat i kao VHDL-AMS) pruža analogne i dizajn signala s miješanim krugovima.
Neki drugi standardi pružaju šire korištenje VHDL-a, najznačajniji je VITAL (VHDL Inicijativa prema ASIC bibliotekama) i dodaci za mikrovalni dizajn krugova.
U Lipnju 2006., VHDL Tehnički komitet Accellere (delegiran od IEEE za rad na slijedećoj nadogradnji standarda) je potvrdio takozvani Nacrt 3.0 VHDL-2006. Kako bi održao potpunu kompatibilnost sa starijim verzijama, ova verzija brojna proširenja koja pružaju pisanje i održavanje VHDL koda jednostavnijim. Glavne promjene sadržavaju uključene podstandarde (1164, 1076.2, 1076.3) u glavni standard 1076, prošireni set operatora, fleksibilniju sintaksu "case" i "generate" izraza, uključivanje VHPI (sučelja za C/C++ jezike) i podset PSL (Property Specification Language). Ove promjene trebaju poboljšati kvalitetu sintetizibilnosti VHDL koda, učiniti testiranje fleksibilnijim i omogućiti šire korištenje VHDL u opisivanju na nivou sustava.

2.10. Rasprave

VHDL je jezik opće namjene, iako zahtjeva simulator na kojemu se vrti kod. On može čitati i zapisivati datoteke na „host“ računalu, pa se VHDL program može napisati tako da generira drugi VHDL program koji se uključuje u dizajn koji razvijamo. Zbog te njegove prirode opće namjene, moguće je koristiti VHDL za pisanje „testbencheva“ koji provjeravaju funkcionalnost dizajna korištenjem datoteka na „host“ računalu za definiranje stimulacije, interakcije s

korisnikom, i uspoređivanje rezultata sa očekivanim. Ovo je slično mogućnostima Verilog jezika. VHDL je čvrsto opisan jezik, a njega neki smatraju superiornijim od Veriloga. Superiornost jednog jezika naspram drugog je predmet intenzivne rasprave između razvijatelja dugo vremena. Oba jezika pružaju relativno jednostavna razvoj neiskusnim korisnicima, ali koji ne možemo sintetizirati u uređaj, ili je prevelik da bi bio praktičan. Jedna tipična zamka u oba jezika je slučajno stvaranje nevidljivih latcheva, umjesto D-bistabila kao elemenata za spremanje. Glavna prednost VHDL kada se koristi za dizajn sustava je da pruža opisivanje ponašanja traženog sustava i provjeru (simulaciju) prije nego li ga alati za sintezu pretvore u pravi hardver (sklopovi i spojevi). Druga prednost je da VHDL pruža opis istodobnih sustava (mnogo dijelova, svaki sa svojim podponašanjem, rade zajedno u isto vrijeme). VHDL je jezik s tokom podataka, za razliku od proceduralnih programskih jezika kao što su BASIC, C i asemblerski kod, koji se svi izvode slijedno, jednom instrukcijom u vremenu.
Krajnja točka je kada se VHDL model prevede u "sklopove i spojeve" koji su mapirani u programabilni logički uređaj kao što je CPLD ili FPGA. Tada je on hardver koji se konfigurira, a ne izvodi kao kod procesora.

2.11. Početak

Kao i kod bilo kojeg hardverskog ili softverskog jezika, da bi postao uspješan VHDL zahtjeva učenje i prakticiranje. Iako je pozadinsko poznavanje kod računalnih programskih jezika (kao što je C) poželjno, nije nužno. Danas postoje besplatni VHDL simulatori, ali su ograničeni u funkcionalnosti kada ih usporedimo sa komercijalnim simulatorima, koji su i više nego dostatni za nezavisno učenje. Ako korisnik želi naučiti RTL kodiranje, npr. dizajn hardverskih sklopova u VHDL-u (naspram jednostavnog dokumentiranja i simulacije ponašanja), onda je potreban i paket za sintezu/dizajn kod učenja. Kao i kod VHDL simulatora, postoje i besplatni FPGA sustavi za sintezu, i više su nego dovoljni za nezavisno učenje. Povratna informacija od alata za sintezu daje korisniku osjećaj relativnu efikasnost različitih stilova kodiranja. Shematski/gate prikaz pruža korisniku sintetizirani dizajn kao dijagram netliste. Mnogi FPGA dizajn paketi također pružaju alternativne metode ulaznog dizajna, kao što su blok dijagrami (sheme) i alate za dijagram stanja. Oni pružaju korisnu početnu šablonu za koridranjem određenih tipova ponovljivih struktura, ili kompleksnih prijelaznih dijagrama. Napokon, uključeni tutorijali i

primjeri su vrijedna pomoć. Gotovo svi FPGA tijekovi dizajna i simulacije podržavaju oba Verilog i VHDL, pružajući korisniku učenje jednog ili oba jezika.

2.12. Primjeri kodova

VHDL dizajn sadrži minimalno jedan entitet koji opisuje sučelje i arhitekturu koja sadrži aktualnu implementaciju. Kao dodatak većina dizajna uključuje module sa bibliotekama. Neki dizajni također sadrže više arhitektura i konfiguracija.

Jednostavni I sklop u VHDL sliči na ovo:

```
-- (Ovo je VHDL komentar)
-- import std_logic from the IEEE library
library IEEE;
use IEEE.std_logic_1164.all;

-- ovo je entitet
entity ANDGATE is
  port (
       IN1 : in std_logic;
       IN2 : in std_logic;
       OUT1: out std_logic);
end ANDGATE;

architecture RTL of ANDGATE is
begin
 OUT1 <= IN1 and IN2;
end RTL;
```

Iako prethodni primjer može izgledati preopširan HDL početnicima, uzmimo u obzir da su mnogi dijelovi opcionalni ili se pišu samo jednom. Općenito jednostavne funkcije kao ove su dio većeg „behavioral“ model, umjesto postojanja različitih modula za nešto tako jednostavno. Kao dodatak, korištenje elemenata kao što su „std_logic“ tip mogu na prvi pogled izgledati

presloženo. Možemo jednostavno koristiti ugrađeni bit tip i izbjeći uvođenje biblioteka za početak. Ipak, korištenje ovih logika sa 9 vrijednosti (U,X,0,1,Z,W,H,l,-) umjesto jednostavnih bitova pružaju moćnu alat za simulaciju i ispravljanje pogrešaka dizajneru koji ne postoje trenutno u nijednom drugom HDL-u. U primjerima koji slijede, vidjeti ćemo da VHDL kod možemo pisati u vrlo kompaktnom obliku. Ipak, iskusni dizajneri obično izbjegavaju ove jednostavne oblike i koriste opširne stilove kodiranja zbog čitljivosti i održavljivosti. Druga prednost opširnijeg stila kodiranja je u manjoj količini resursa koja se koristi kod programiranja u programabilne logičke uređaje kao što su CPLD-ovi.

2.12.1. Sintetizibilni konstruktori i VHDL šablona

Originalno sintezeri, koji odgovaraju kompajlerima u HDL svijetu, koriste set šablona za identifikaciju čestih hardverskih konstruktora u HDL kodu (podsjetnik VHDL je jezik za opis hardvera, ne programski jezik). Ove šablone se mogu još uvijek koristiti, ipak HDL alati su sofisticiraniji sada. Zbog njihovog uobičajenog jedan na jedan mapiranja u dobro poznate digitalne sklopove, ove šablone su ono što uobičajeno inače iskusni hardverski dizajner koristi kod ulaska u HDL svijet. Oni su također korisni za one koji su potpuno novi u digitalnom dizajnu.
Neke digitalne komponente imaju više šablona, kao ovaj multiplekser u primjer koji slijedi:

2.12.2. MUX šablona

Multiplekser, ili "MUX" kao što se često zove, je jednostavni konstruktor vrlo čest u hardverskom dizajnu. Primjer ispod prikazuje jednostavni 2-1 MUX, sa ulazima A i B, i izborom S, te izlazom X:

```
-- template 1:
X <= A when S = '1' else B;

-- template 2:
with S select X <= A when '1' else B;
```

```
-- template 3:
process(A,B,S)
begin
  case S is
    when '1'    => X <= A;
    when others => X <= B;
  end case;
end process;

-- template 4:
process(A,B,S)
begin
  if S = '1' then
    X <= A;
  else
    X <= B;
  end if;
end process;

-- template 5 - 4:1 MUX, gdje S je 2-bit std_logic_vector :
process(A,B,C,D,S)
begin
  case S is
    when "00"   => X <= A;
    when "01"   => X <= B;
    when "10"   => X <= C;
    when others => X <= D; -- or when "11"
  end case;
end process;
```

Ove tri zadnje šablone koriste ono što se u VHDL-u zove sekvencijalni kod. Sekvencijalni dio se uvijek stavlja unutar procesa i ima blago različitu sintaksu koja nalikuje tradicionalnim programskim jezicima.

2.12.3. Latch šablona

Transparentni „latch“ je osnovna jednobitna memorija koja se nadograđuje kada je „enable“ signal u prijelazu iz niskog u visoko logičko stanje:

```
-- latch template 1:
Q <= D when Enable = '1' else Q;

-- latch template 2:
process(D,Enable)
begin
  if Enable = '1' then
    Q <= D;
  end if;
end process;
```

SR-bistabil koristi set i reset signale umjesto toga:

```
-- SR-bistabil template 1:
Q <= '1' when S = '1' else
  '0' when R = '1' else Q;

-- SR-bistabil template 2:
process(S,R)
begin
  if S = '1' then
    Q <= '1';
  elsif R = '1' then
    Q <= '0';
  end if;
end process;
```

Šablona 2 ima implicitno "else Q <= Q;" koje možemo eksplicitno dodati ako želimo.

```
-- Ovo je RS-bistabil (tj. reset dominira)
process(S,R)
begin
  if R = '1' then
    Q <= '0';
  elsif S = '1' then
    Q <= '1';
  end if;
end process;
```

2.12.4. D-tip bistabila

D-tip bistabil uzrokuje dolazeći signal na rastućem ili padajućem bridu signala takta. D-tip bistabila je osnova svih sinkronih logika.

```
-- najjednostavniji template D-tipa bistabila (ne preporučljiv)
Q <= D when rising_edge(CLK);

-- preporučljiv template D-tipa bistabila:
process(CLK)
begin
  -- koristimo falling_edge(CLK) kako bismo uzorkovali na padajući brid
  if rising_edge(CLK) then
    Q <= D;
  end if;
end process;

-- alternativni template D-tip bistabila:
process
begin
  wait until rising_edge(CLK);
```

```
  Q <= D;
end process;
```

Neki bistabili također imaju i „Enable“ signal i asinkroni ili sinkrone „Set“ i „Reset“ signale:

```
-- template za asinkroni reset sa Enable signalom:
process(CLK, RESET)
begin
  if RESET = '1' then   -- ili '0' ako je RESET aktivan u 0...
    Q <= '0';
  elsif rising_edge(CLK) then
    if Enable = '1' then  -- ili '0' ako je Enable aktivan u 0...
      Q <= D;
    end if;
  end if;
end process;

-- template za sinkroni reset sa clock enable:
process(CLK)
begin
  if rising_edge(CLK) then
    if RESET = '1' then
      Q <= '0';
    elsif Enable = '1' then  -- ili '0' ako je Enable aktivan u 0...
      Q <= D;
    end if;
  end if;
end process;
```

Česta početnička pogreška je postavljanje set ili reset ulaza i nekorištenje. Npr. slijedeći komadići koda nisu ekvivalentni, prvi je jednostavni D-tip bistabila, dok je slijedeći D-tip bistabila sa povratnim MUX-om.

```
-- jednostavni D-tip bistabila
process(CLK)
begin
  if rising_edge(CLK) then
    Q <= D;
  end if;
end process;

-- BAD VHDL: ovaj je kod ne čini bistabil bez reseta!!
process(CLK, RESET)
begin
  if RESET = '1' then
       -- ne čini ništa. Q nije postavljen...
  elsif rising_edge(CLK) then
    Q <= D;
  end if;
end process;
```

2.12.5. Primjer brojača

Slijedeći primjer je brojač prema gore sa asinkronim resetom, paralelnim punjenjem i podesivom širinom. Prikazuje korištenje „unsigned“ tipa i VHDL generika. Generici su vrlo slični argumentima ili šablonama u drugim tradicionalnim programskim jezicima kao što je C ili C++.

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;    -- za unsigned tip podatka

entity counter_example is
generic ( WIDTH : integer := 32);
port (
  CLK, RESET, LOAD : in std_logic;
```

```
  DATA : in  unsigned(WIDTH-1 downto 0);
  Q    : out unsigned(WIDTH-1 downto 0));
end entity counter_example;

architecture counter_example_a of counter_example is
signal cnt : unsigned(WIDTH-1 downto 0);
begin
  process(RESET, CLK)
  begin
    if RESET = '1' then
      cnt <= (others => '0');
    elsif rising_edge(CLK) then
      if LOAD = '1' then
        cnt <= DATA;
      else
        cnt <= cnt + 1;
      end if;
    end if;
  end process;

  Q <= cnt;

end architecture counter_example_a;
```

Tip „std_logic_vector“ možemo koristiti kao i „unsigned“ tip. Kompleksniji brojači mogu sadržavati „if/then/else“ izraze unutar „rising_edge(CLK)“, „elsif“ za dodavanje druge funkcije, kao što su omogućavanje brojanja, zaustavljanje i vraćanje na neku drugu vrijednost, generiranje izlaznih signala za brojač, ... Treba obratiti pažnju kada se slažu i ugnježđuju takve kontrole ako se koriste zajedno, kako bi proizvele željene prioritete i minimizirale broj logičkih nivoa.

2.12.6. Fibonaccijev niz

Slijedeći primjer je složeniji:

```
-- Fib.vhd
-- Generator niza Fibonnacijevih brojeva

library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;

entity Fibonacci is
port (
        Reset : in std_logic;
        Clock : in std_logic;
        Number : out unsigned(31 downto 0)
);
end entity Fibonacci;

architecture Rcingham of Fibonacci is

signal Previous : natural;
signal Current : natural;
signal Next_Fib : natural;

begin
        Adder:
        Next_Fib <= Current + Previous;

        Registers:
        process (Clock, Reset) is
        begin
                if Reset = '1' then
```

```
                Previous <= 1;
                Current <= 1;
        elsif rising_edge(Clock) then
                Previous <= Current;
                Current <= Next_Fib;
        end if;
    end process Registers;

    Number <= to_unsigned(Previous, 32);

end architecture Rcingham;
```

Kada se simulira uspješno generira Fibonaccijev niz, sve dok Next_Fib ne pređe raspon natural tipa podataka. Kada se sintetizira sa alatima FPGA dobavljača, jedan "Adder" modul se implementira. Inače trebamo zamijeniti izraz sa instancom komponente koja implementira logičku funkciju.

2.12.7. Samo simulacijski konstruktori

Širok podskup VHDL kodova ne možemo pretočiti u hardver. Ovaj podskup znamo i kao ne-sintetizibilan ili samo simulacijski podskup VHDL-a i možemo ga koristiti samo za prototipiranje, simulaciju i debugiranje. Na primjer, slijedeći kod generira takt frekvencije 50MHz. On može poslužiti na primjer za vođenje ulaza takta u dizajnu tijekom simulacije. On je ipak samo simulacijski konstruktor i ne može se implementirati u hardver.

```
process
begin
  CLK <= '1'; wait for 10 ns;
  CLK <= '0'; wait for 10 ns;
end process;
```

Samo simulacijski konstruktori se mogu koristiti za izgradnju kompleksnih valnih oblika u kratkom vremenu. Takvi valni oblici mogu se koristiti za testiranje vektora kompleksnih dizajna ili za prototipiranje nekih sintetizibilnih logika koje ćemo implementirati u budućnosti.

```
process
begin
  wait until START = '1'; -- pričekaj dok START nije 1

  for i in 1 to 10 loop -- onda pričekaj nekoliko perioda trajanja takta...
    wait until rising_edge(CLK);
  end loop;

  for i in 1 to 10 loop   -- upiši brojeve 1 do 10 u DATA svaki ciklus
    DATA <= to_unsigned(i, 8);
    wait until rising_edge(CLK);
  end loop;

  -- pričekaj dok se output ne promjeni
  wait on RESULT;

  -- sada povisi ACK za signala trajanja takta
  ACK <= '1';
  wait until rising_edge(CLK);
  ACK <= '0';

  -- i tako dalje...
end process;
```

3. ANALIZA RAZVOJNE PLOČICE I KOMPONENATA

Zadatak ovog završnog rada jest preko razvojne pločice Xilinx Spartan 3 isporgramirati kod u VHDL-u koji će odabirom prekidača s pločice na ekranu spojenom na VGA port na pločici prikazivati određeni uzorak boja (R G B) u prije određenoj i isprogramiranoj rezoluciji i vremenu osvježavanja monitora (reslolution @ refresh rate). Signali R G B su jednobitni i koriste se kao bi pokazali samo linije na ekranu za svaki od odabranih načina rada.

Za programiranje pločice se koristi Xilinx-ov program WebISE verzije 11.1 koji je besplatan, te koristi svoje potprograme kao što su PlanAhead v.11 za dodjeljivanje određenih funkcija iz programa na određene pinove na pločici i iMPACT v.11 za programiranje napravljenog koda u VHDL-u i dodijeljenih pinova na razvojnu pločicu Spartan 3.

Programiranje i debbugiranje se vrši preko JTAG paralelnog kabla kojim se spaja računalo sa Spartan 3 razvojnom pločicom.

Pločica ima i integrirani takt „mclk“ od 50 MHz, koji koristimo i pomoću DCM-a (Digital Clock Manager) takt prilagođujemo taktu „pixel clocka“, tj. takta piksela. Pixel clock za svaku rezoluciju sa određenim vremenom osvježavanja ima drugačiju vrijednost u rasponu od 25 MHz do 100 MHz.

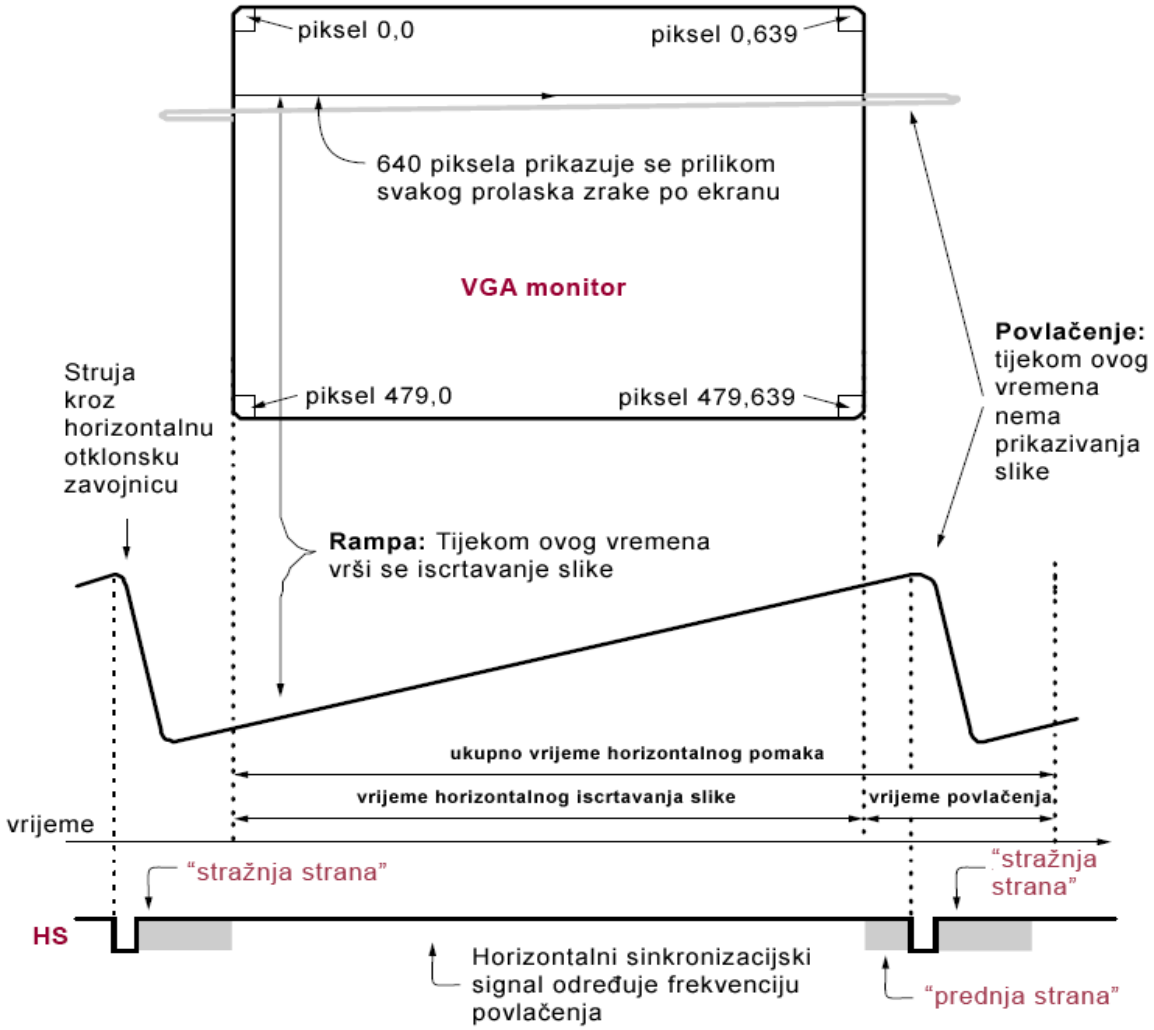
Odabir rezolucija i vremena osvježavanja monitora program povlači iz „lookup“ tablice (LUT) koja se sprema u memoriju na pločici Spartan 3.

Da bismo sve to učinili, potrebno je i poznavanje načela rada monitora, u ovom slučaju CRT, svojstva VGA porta, svojstva čipa s pločice XS3C200 FT256A. Uz sve to potrebna nam je i shema spajanja te prikaz isprogramiranog sklopa pomoću grafičkih blokova.

3.1. CRT monitori

CRT monitori rade na principu amplitudno moduliranih pokretnih elektronskih zraka koje na ekranu presvučenom fosforom iscrtavaju sliku. LCD monitori koriste polje ćelija od tekućeg kristala. Promjenom napona na ćeliji mijenja se permitivnost svjetlosti koja prolazi kroz ćeliju (piksel). LCD monitori koriste isti oblik signala za prikaz slike kao i CRT monitori. Kod CRT monitora, valni oblici struje, prolazeći kroz otklonske zavojnice, proizvode magnetska polja koja

otklanjaju elektronske zrake. Elektronska zraka prolazi po ekranu horizontalno, s lijeva na desno, te vertikalno, od gore prema dolje. Na slici 3.1. prikazan je princip rada.

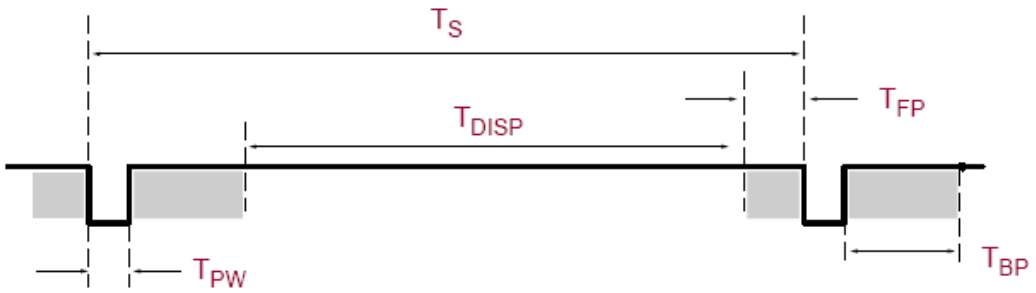


Slika 3.1. Princip rada CRT monitora

Veličina zrake, brzina prolaska zrake po ekranu te brzina modulacije elektronske zrake određuje rezoluciju prikaza. Razvojna pločica sa Spartan 3 FPGA-om koristi tri bita po pikselu za prikaz jedne od osam mogućih boja. Podaci koji se prikazuju na ekranu dolaze nalaze se u video memoriji. VGA kontroler uzima podatke iz međuspremnika video podataka istodobno sa kretanjem zraka po ekranu. Kontroler prikazuje podatke o određenom pikselu točno u trenutku prolaska elektronske zrake preko tog piksela. VGA kontroler generira horizontalne (HS) i vertikalne (VS) sinkronizacijske signale i usklađuje isporuku video podataka pri svakom taktu piksela. Takt piksela određuje vrijeme koje potrebno za prikaz jednog piksela. VS signal određuje frekvenciju osvježavanja prikaza, odnosno frekvenciju pri kojoj se svi podaci ponovno

iscrtavaju na ekranu. Minimalna frekvencija osvježavanja je funkcija fosfora i intenziteta elektronske zrake ekrana. Frekvencija osvježavanja se obično kreće od 60 Hz do 120 Hz.

U tablici 3.1. nalaze se parametri sinkronizacijskog signala za VGA prikaz rezolucije 640x480 pri taktu piksela od 25 MHz i frekvenciji osvježavanja 60 Hz ± 1. Sinkronizacijski signal prikazan je na slici 3.2.



Slika 3.2. Sinkronizacijski signal

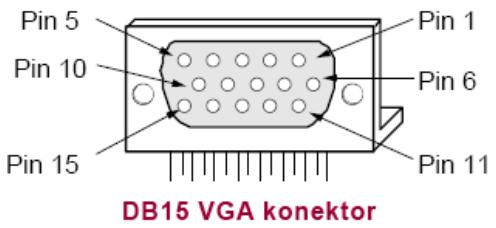
Simbol	Parametar	Horizontalna sinkronizacija			Vertikalna sinkronizacija	
		Vrijeme	Taktovi	Linije	Vrijeme	Linije
T_S	Vrij. sinkr. impulsa	16.7 ms	416800	521	32 µs	800
T_{DISP}	Vrijeme prikaza	15.36 ms	384000	480	25.6 µs	640
T_{PW}	Širina impulsa	64 µs	1600	2	3.48 µs	96
T_{FP}	Prednji prag	320 µs	8000	10	640 µs	16
T_{BP}	Zadnji prag	928 µs	23200	29	1.92 µs	48

Tablica 3.1. Parametri sinkronizacijskog signala za 640x480 VGA prikaz

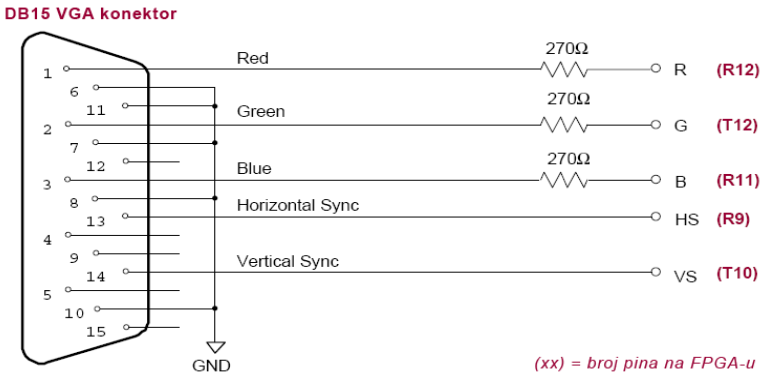
Takt piksela vrši pokretanje brojača koji upravlja horizontalnim sinkronizacijskim signalom. Dekodirane vrijednosti brojača generiraju HS signal. Ovaj brojač omogućava praćenje trenutne lokacije piksela u određenom redu. Odvojeni brojač vrši praćenje vertikalnog sinkronizacijskog signala. Ovaj brojač se uvećava za jedan sa svakim HS impulsom. Dekodirane vrijednosti brojača generiraju VS signal. Vertikalno sinkronizacijski brojač omogućava praćenje trenutnog reda. Horizontalno i vertikalno sinkronizacijski brojač neprestano formiraju adresu u međuspremniku video prikaza.

3.2. VGA port

Na razvojnoj pločici Spartan-3 nalazi se VGA port (konektor DB15) koji je prikazan na slici 3.3. Port se može preko standardnog monitorskog VGA kabela povezati sa većinom CRT ili LCD monitora. FPGA upravlja sa pet VGA signala kako je prikazano na slici 3.4. Signali su RGB komponente, crvena (R), zelena (G) i plava (B), te signali za horizontalnu (HS) i vertikalnu (VS) sinkronizaciju.



Slika 3.3. Standardni DB 15 VGA konektor



Slika 3.4. Konekcije između pinova DB15 konektora i pinova FPGA čipa

FPGA pinovi koji su povezani sa VGA portom prikazani su u tablici 3.2.

Signal	FPGA pin
Crvena (R)	R12
Zelena (G)	T12
Plava (B)	R11
Horizontalna sinkronizacija (HS)	R9
Vertikalna sinkronizacija (VS)	T10

Tablica 3.2. Konekcije između pinova DB15 konektora i pinova Spartan-3 FPGA čipa

Na svakoj liniji za boju nalazi se otpornik od 270Ω. Ovaj otpornik u kombinaciji sa terminirajućim 75 ohmskim otpornikom na VGA kabelu osigurava da se RGB signali zadrže u

rasponu od 0V do 0.7V. Signali HS i VS su TTL kompatibilni. RGB signali mogu generirati boju rezolucije 3-bita, tj. osam mogućih boja koje su navedene u tablici 3.3.

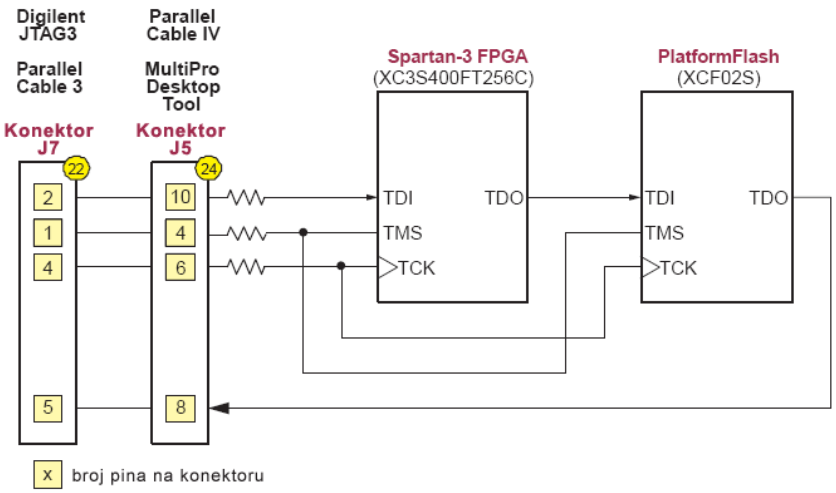
Crvena (R)	Zelena(G)	Plava(B)	Komb. boja
0	0	0	Crna
0	0	1	Plava
0	1	0	Zelena
0	1	1	Cijan
1	0	0	Crvena
1	0	1	Ljubičasta
1	1	0	Žuta
1	1	1	Bijela

Tablica 3.3. 3-bitni kodovi boja

Oblik VGA signala definiran je VESA (Video Electronics Standards Association) standardom.

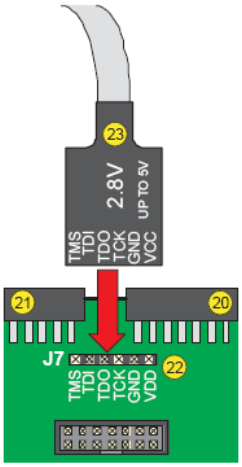
3.3 JTAG port

Razvojna pločica Spartan-3 posjeduje JTAG sučelje za programiranje i debuggiranje. FPGA Spartan-3 i flash PROM dio su JTAG lanca kao što je prikazano na slici 3.5. Na pločici se nalaze dva JTAG porta (konektora) za različite tipove kabela. Uz razvojnu pločicu isporučen je Digilent JTAG3 paralelni/JTAG kabel koji se priključuje na konektor J7 na razvojnoj pločici (slika 3.5). Drugi kraj kabela priključuje se na paralelni port osobnog računala. Digilent JTAG3 kabel kompatibilan je sa softverskim alatom Xilinx iMPACT.



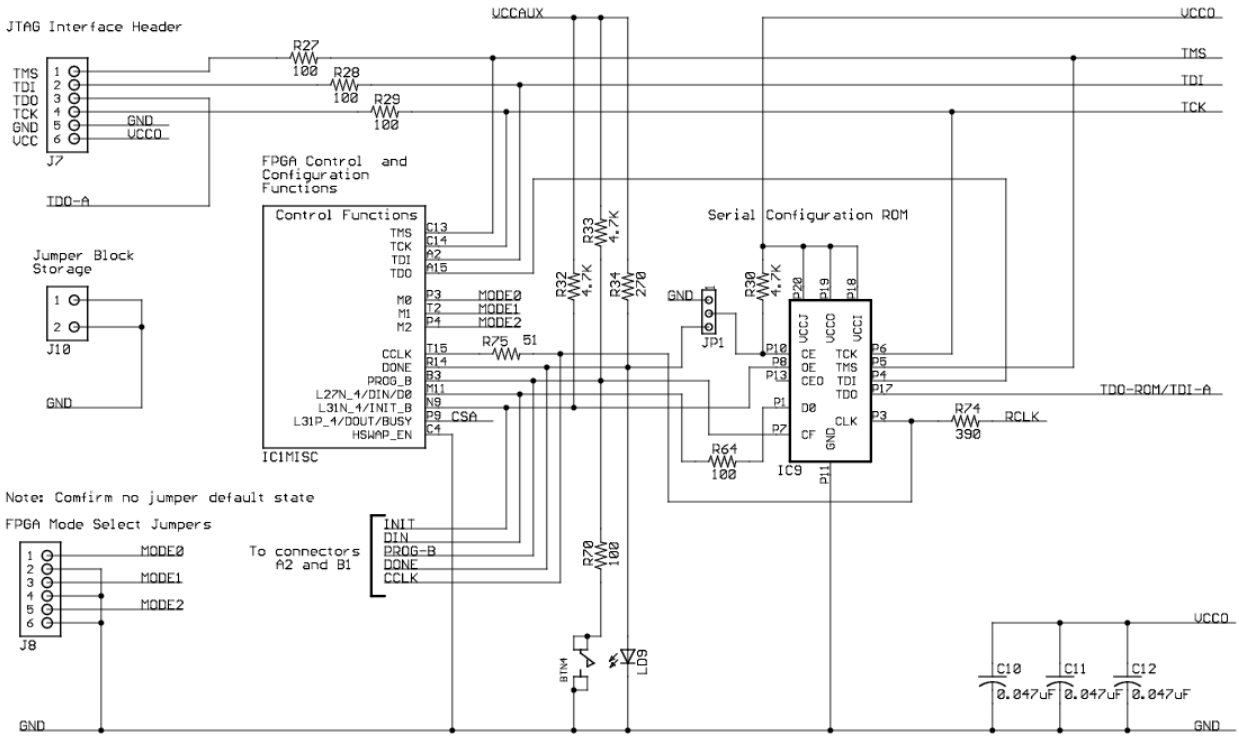
Slika 3.5. JTAG lanac na razvojnoj pločici Spartan-3

Na slici 3.6. prikazana je ispravan položaj konektora JTAG3 kabela u odnosu na konektor J7 na razvojnoj pločici.



Slika 3.6. Ispravan položaj JTAG3 kabela u odnosu na J7 konektor

Detaljna električna shema JTAG sučelja nalazi se na slici 3.7.

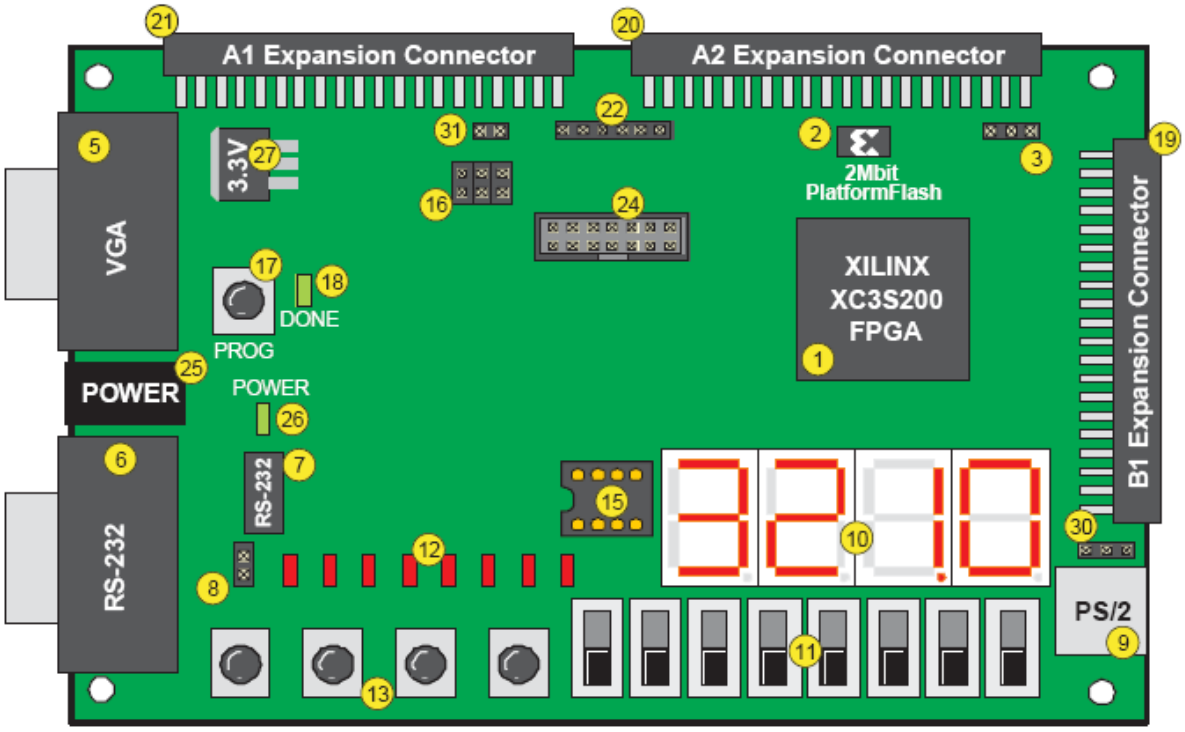


Slika 3.7. Električna shema JTAG sučelja

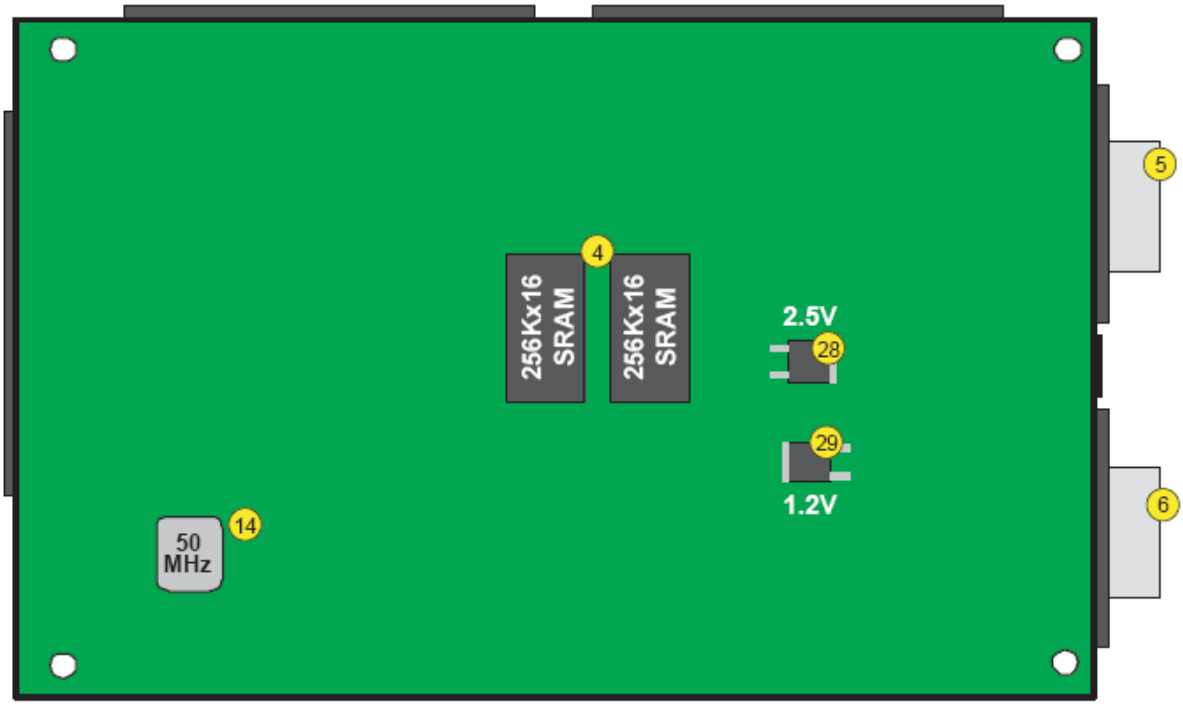
Uz JTAG konektor J7, na razvojnoj pločici se nalazi i JTAG konektor J5 (slika 3.7.) namijenjen za 14-pinske flat kabele.

3.4. Specifikacije ostalih dijelova na razvojnoj pločici

1. Xilinx XC3S200FT256 FPGA
2. Xilinx XCF02S programabilni konfiguracijski flash PROM, 2 Mbita
3. Kratkospojnik koji omogućava FPGA-u čitanje podataka sa PROM-a ili drugih resursa
4. Brzi asinkroni SRAM, 1 MB (dva SRAM čipa kapaciteta 256Kx16, brzine 10ns, na donjoj strani pločice)
5. VGA port, 3 bita, 8 boja
6. RS-232 serijski port, 9-pinski
7. RS-232 naponski pretvornik
8. drugi RS-232 transmit and receive kanal
9. PS/2 port
10. 7-segmentni LED pokazivač, 4 znaka
11. 8 prekidača
12. 8 LED dioda
13. 4 tipkala
14. Kristalni oscilator od 50 MHz
15. Podnožje za dodatni generator takta
16. Kratkospojnici za podešavanje načina rada FPGA
17. Tipkalo za rekonfiguriranje FPGA
18. LED dioda koja pokazuje kada je FPGA uspješno konfiguriran
19. 40-pinski B1 konektor za povezivanje sa perifernim karticama
20. 40-pinski A2 konektor za povezivanje sa perifernim karticama
21. 40-pinski A1 konektor za povezivanje sa perifernim karticama
22. JTAG download/debug port
23. JTAG kabel za povezivanje sa paralelnim portom računala, služi za programiranje i debuggiranje FPGA
24. JTAG download/debug port kompatibilan sa Xilinxovim Paralel Cable IV i MultiPRO Desktop Tool
25. Ulaz za napajanje dovedeno sa AC adaptera
26. LED pokazivač koji signalizira rad sustava
27. Naponski regulator 3.3V
28. Naponski regulator 2.5V
29. Naponski regulator 1.2V



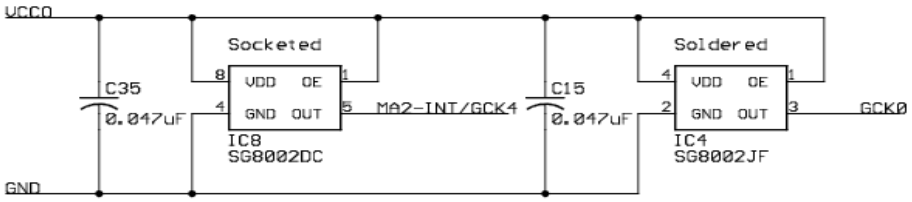
Slika 3.8. Xilinx Spartan 3 Starter – prednja strana



Slika 3.9. Xilinx Spartan 3 Starter – stražnja strana

3.5. DCM (Digital Clock Manager)

Pomoću DCM-ova (Digital Clock Manager) ili Generatora Takta, Spartan-3 FPGA čipa moguće je iz osnovne frekvencije signala takta dobiti druge frekvencije. Na razvojnoj pločici nalazi se 50 MHz oscilator Epson SG-8002JF (slika 3.9, element 14) i 8-pinsko podnožje za dodatni oscilator (slika 3.8, element 15). Detaljna električna shema prikazana je na slici 3.10.

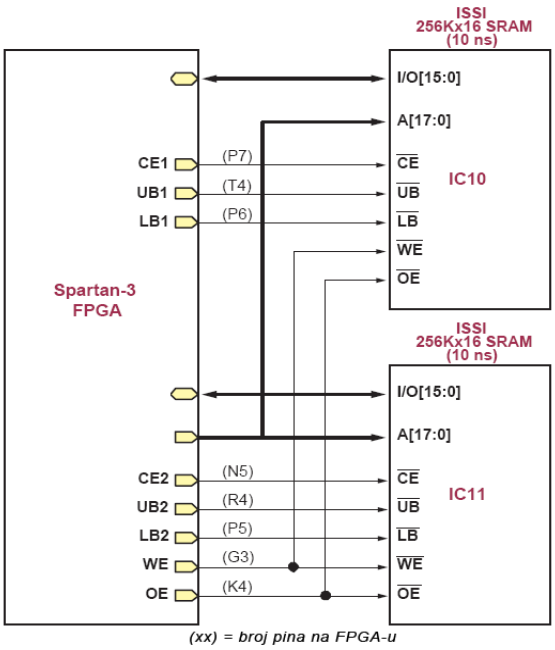


Slika 3.10. Električna shema generatora takta

DCM može implementirati kao digitalni sintetizer frekvencije, digitalni zakretač faze, digitalni pokrivač spektra, kao takt kašnjenja zaključane petlje,…

3.6. SRAM

Razvojna pločica posjeduje 1 MB brze asinkrone SRAM memorije (dva 256Kx16 SRAM čipa). Shema konekcije između SRAM čipova i FPGA čipa nalazi se na slici 3.11.



Slika 3.11. Shema konekcije između FPGA i SRAM-a

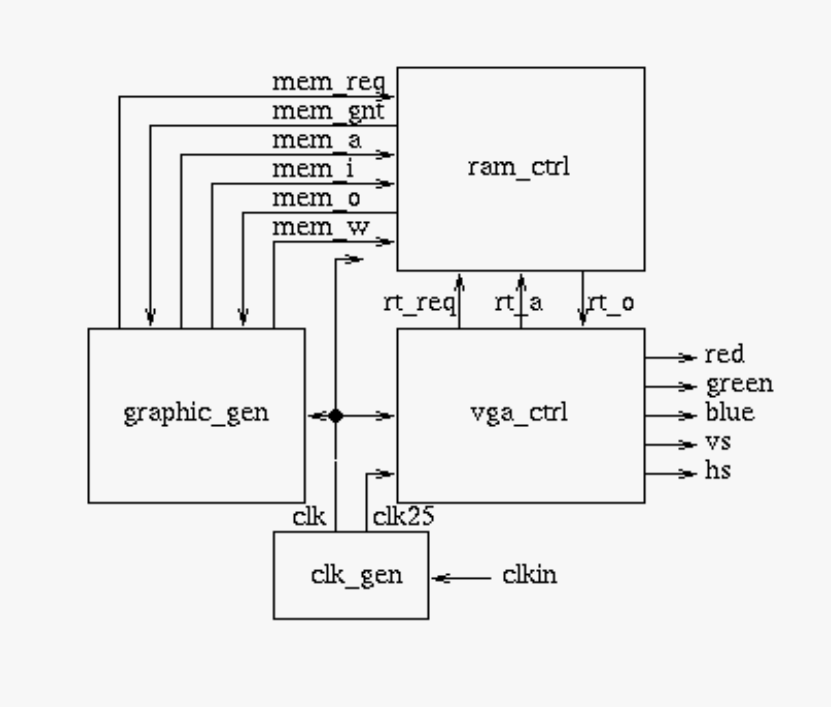
SRAM polje sačinjava jedna 256Kx32 SRAM memorija ili dvije neovisne 256Kx16 SRAM memorije. WE (write-enable), OE (output-enable) i adresne linije (A[17:0]) su zajedničke za oba SRAM čipa. Svaki čip ima vlastitu CE upravljačku liniju (chip-select), te zasebne UB i LB linije (byte-enable) za izbor višeg ili nižeg bajta u 16-bitnoj podatkovnoj riječi.

4. RAZVOJ I OBJAŠNJENJE ZADATKA

Kod pisan za ovaj zadatak se sastoji od više segmenata. „Main“ obilježava glavni dio programa koji sadrži svoje segmente kao što su „vga_controller“, „lookup“, „Clk_GEN“, te „counter“ i „mux“. Svaki do tih dijelova ima svoju ulogu u programu, tj. kontrolira jedan dio sklopova na razvojnoj pločici. Izrada dizajna vrši se pisanjem VHDL koda koji opisuje ponašanje (funkciju) elektroničkog sklopa

4.1. „main“

U arhitekturi „main“-a se opisuje način na koji bi se sklop trebao ponašati, funkcionirati, te se definiraju sve komponente programa koje su dalje raščlanjene, te signali koji se koriste u sklopu. U kodnom dijelu „begin“ nastavljamo sa rasporedom i priključivanjem određenih pinova na ostale pinove koje smo zadali u programu, za to nam služi naredba „port map“. Isto tako se dodjeljuju pinovi za svaki signal u svakom segmentu „main“-a.



Slika 4.1. Grafički prikaz koda u „main“-u

4.2. „vga_controller“

Segment vezan za prikaz slike na monitoru se naziva „vga_controller“ i sastoji se od svojih ulaznih i izlaznih signala. Za ulaz koristimo „mclk“ što bi označavalo glavni takt koji se treba generirati pomoću DCM-ova i za svaku rezoluciju izabirati iz lookup tablice koja je pohranjena kao drugi segment u programu. Iz lookup tablice se šalju i ostali podaci, ali o tome će kasnije biti riječi u potpoglavlju 4.3.
Osim ulaza, dakako, postoje i izlazi, koji su:
„hs“ - predstavlja horizontalni sinkronizacijski puls
„vs“ - predstavlja vertikalni sinkronizacijski puls
„red“ – predstavlja crvenu komponentu signala
„grn“ – predstavlja zelenu komponentu signala
„blu“ – predstavlja plavu komponentu signala
„hpixels“ – ukupan broj horizontalnih točaka
„vlines“ – ukupan broj vertikalnih točaka
„hbp“ – horizontalna otklonska zavojnica
„hfp“ – horizontalna frekvencija povlačenja
„vbp“ – vertikalni otklonska zavojnica
„vfp“ – vertikalni frekvencija povlačenja
„hpol“ – horizontalnu polarizaciju
„vpol“ – vertikalnu polarizaciju

Ovaj modul kreira tri linije R G i B na ekranu koristeći vertikalnu frekvenciju osvježavanja (refresh rate) po želji. On se dobiva dijeljenjem sistemskog takta po potrebi i takav takt (clock) koristi za takt točke (pixel clock).
Signali koji se koriste su:
signal hc, vc - horizontalni i vertikalni brojači (counter-i)
signal vidon - govori kad se na ekranu počinje prikazivati signal
signal vsenable - enable za vertikalni brojač (counter)

Tijek programa se odvija tako da kad brojač dosegne kraj brojanja točaka (pixel count), resetira se. Horizontalni sinkronizacijski puls se povećava ako je u stanju „enabled“ te ako je dosegnut broj linija. Isto tako se ponaša i vertikalni sinkronizacijski puls, ako je dosegnut određen broj

linija, te ako je zadovoljen uvjet stanja „enabled“. Uključuju se crveni (red), zeleni (grn) i plavi (blu) piksel na specifičnom horizontalnom broju. Enable na kraju dozvoljava video izlaz kada je on u granicama dopuštenih točaka, tj. Piksela.

4.3. „lookup“ tablice

Lookup tablice se koriste za povlačenje veće količine podataka, za iste komponente signala koje se koriste u programu. U ovom slučaju one sadržavaju podatke za rezolucije i frekvencije osvježavanja monitora, te polarizacije u određenom stanu ulaznih prekidača SW1, SW2, SW3 i SW4.

Signali koji se „uzimaju“ iz lookup tablica su: hpixels, vlines, bhp, hfp, vbp, vfp, hpol i vpol.

Uz rezolucije naveden je i takt piksela, koji se dobiva preko DCM-ova, i različit je za svaku rezoluciju i frekvenciju osvježavanja monitora. Dolje je prikazana tablica 4.1. u kojoj su iskazani svi parametri koje koristi program za računanje i kreiranje slike na monitoru. Podaci su zapisani u dekadskom obliku radi lakšeg čitanja u tablici 4.1. dok su u tablici 4.2. zapisani u binarnom obliku, kakve program i koristi.

#	SWT	Hpixels	Vlines	Hbp	Hfp	Vbp	Vfp	HP	VP	res@ref.rate	Pixel_clk
1	0000	800	525	144	784	31	511	-	-	640x480@60Hz	25 MHz
2	0001	832	520	168	808	31	511	-	-	640x480@73Hz	31.5 MHz
3	0010	840	500	184	824	19	499	-	-	640x480@75Hz	31.5 MHz
4	0011	832	509	136	776	28	508	-	-	640x480@85Hz	36 MHz
5	0100	848	509	168	808	28	508	-	-	640x480@100Hz	43.16MHz
6	0101	1024	625	200	1000	24	624	+	+	800x600@56Hz	36 MHz
7	0110	1056	628	216	1016	27	627	+	+	800x600@60Hz	40 MHz
8	0111	1040	666	184	984	29	629	+	+	800x600@72Hz	50 MHz
9	1000	1056	625	240	1040	24	624	+	+	800x600@75Hz	49.5 MHz
10	1001	1048	631	216	1016	30	630	+	+	800x600@85Hz	56.25MHz
11	1010	1072	636	224	1024	35	635	-	+	800x600@100Hz	68.18MHz
12	1011	1264	817	232	1256	49	817	+	+	1024x768@43Hz	44.9 MHz
13	1100	1344	806	276	1320	35	803	-	-	1024x768@60Hz	65 MHz
14	1101	1328	806	280	1304	35	803	-	-	1024x768@70Hz	75 MHz
15	1110	1312	800	272	1296	31	799	+	+	1024x768@75Hz	78.8 MHz
16	1111	1376	808	304	1328	39	807	+	+	1024x768@85Hz	94.5 MHz

Tablica 4.1. – Podaci u dekadskom zapisu

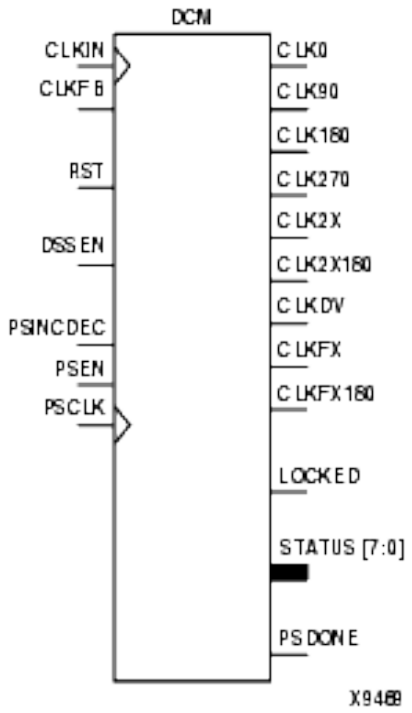
#	Hpixels	Vlines	Hbp	Hfp	Vbp	Vfp
1	00001010000	01000001101	00010010000	01100010000	00000011111	00111111111
2	01101000000	01000001000	00010101000	01100101000	00000011111	00111111111
3	01001001000	00111110100	00010111000	01100111000	00000010011	00111110011
4	01101000000	00111111101	00010001000	01100001000	00000011100	01111111001
5	01101010000	00111111101	00010101000	01100101000	00000011100	00111111100
6	10000000000	01001110001	00011001000	01111101000	00000011000	01001110000
7	10000100000	01001110100	00011011000	01111111000	00000011011	01001110011
8	10000100000	01001110001	00011110000	01000001000	00000011000	01001110000
9	10000100000	01001110001	00011110000	10000010000	00000011000	01001110000
10	10000011000	01001110111	00011011000	01111111000	00000011110	01001110110
11	10000110000	01001111100	00011100000	01000000000	00000100011	01001111011
12	10011110000	01100110001	00011101000	10011101000	00000110001	01100110001
13	10101000000	01100100110	00100010100	10100101000	00000100011	01100100011
14	10100100000	00001010000	00100010000	10100010000	00000011111	01100011111
15	10100100000	01100100000	00100010000	10100010000	00000011111	01100011111
16	10101100000	01100101000	00100110000	10100110000	00000100111	01100100111

Tablica 4.2. – Podaci u binarnom zapisu

4.4. „Clk_GEN“ generator takta

Generator takta služi za, kao i što sam naziv kaže, generiranje taktova preko 4 DCM-a koja imamo na razvojnoj pločici. On ima svojih par mana koji znatno utječu na programski kod i otežavaju pomalo programiranje. Naime, može se dijeliti i množiti s najviše 32, svaki DCM može imati recimo najmanji zajednički višekratnik koji podijeljen s nekim određenim brojem daje, vrlo često približni, jer je teško pogoditi točni takt piksela, te ovo ponekad predstavlja problem jer se ne mogu odabrati onda baš sve rezolucije sa frekvencijama osvježavanja monitora koje postoje, nego samo neke, koje imaju slične, približne ili djeljive taktove koji se mogu izvući iz ta 4 DCM-a. U ovom slučaju DCM1 ima najmanji zajednički višekratnik 94.5, što ujedno odgovara već jednom taktu piksela, te kad ga podijelimo s 3 dobivamo 4 nove rezolucije, tj. 4 nova takta piksela. DCM2 ima najmanji zajednički višekratnik 325 koji se obavezno mora dijeliti, te već pri dijeljenju s 2 imamo jednu rezoluciju, pri dijeljenju sa 3 imamo 3 rezolucije, sa4 jednu i sa 5 opet jednu novu i različitu rezoluciju od svih do sada. DCM3 ima najmanji zajednički višekratnik 1000 koji dijeljenjem daje veći spektar frekvencija. Tako dijeljen sa 8, 12, 20, 22, 23, i 29 daje po jednu različitu frekvenciju. DCM4 služi kao pomoćni, tj. na njemu ćemo

generirati neke rezolucije koje se nisu uspjele pokriti sa ova prethodna 3 DCM-a, tako da imamo više različitih načina rada monitora. On zapravo može pokrivati i sam jednu ili dvije rezolucije.

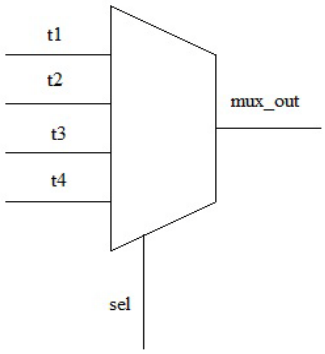


Slika 4.2. DCM (digitalni menadžer takta)

4.4.1. „counter“

Kao segment generatora takta, brojač služi da bi za svaku rezoluciju brojao do određenog broja s kojim se dijeli određeni DCM, te na taj način izvukao određeni takt piksela. Nakon što je brojač došao do određene vrijednosti on se resetira i nastavlja brojati za sljedeću rezoluciju.

4.4.2. „MUX“ multiplekser



Slika 4.3. multiplekser sa ulazima i izlazima

Multiplekser, isto kao komponenta generatora takta služi da bi ukazao brojaču na koji od četiri DCM-a se šalje određeni signal, tj. za koliko brojač mora brojati u svakome DCM-u dok se ne ispoštuju uvjeti za sve zadane rezolucije koje ovise o tome DCM-u.

ZAKLJUČAK

U ovom završnom radu napravljen je program koji kontrolira rezoluciju i frekvenciju osvježavanja monitora pomoću prekidača koji predstavljaju ulaze, a nalaze se na razvojnoj pločici. Razvojna pločica podržava samo neke rezolucije zbog integriranog takta koji je 50MHz i DCM-ova koji su malo neprecizni i ograničeni pa se ne može dobiti dovoljno precizan „pixel clock“, što rezultira ograničavanjem na neke rezolucije. Slika koja se prikazuje na ekranu se maksimalno sastoji od 3 bita - R G B, jer mikrokontroler na pločici ne podržava više bitova za veći spektar boja. To se može riješiti tako da se na pomoćne konektore za povezivanje s perifernim jedinicama dodaju drugi moduli sa vlastitim mikrokontrolerima koji podržavaju i 24-bitne boje, te rade na većim takovima.

Na Spartan 3 razvojnu pločicu mogu se kasnije dodati u ulazne jedinice poput tipkovnice i miša, dodati memorijski kontroler u kojem će se pohraniti neka grafika ili tekst ili nešto slično što će koristiti ovo grafičko sučelje za prikaz slike na monitoru.

LITERATURA

[1] V.A. Pedroni, Circuit design with VHDL, Massachusetts Institute of Technology, 2004.

[2] C. Maxfield, The Design Warrior's Guide to FPGAs 2nd edition, Elsevier Science, 2004.

[3] Peter J. Ashenden, Jim Lewis, VHDL 2008: Just the New Stuff (Systems on Silicon), MK, 2008.

[4] Pong P. Chu, FPGA prototyping by VHDL examples, Cleveland State University, 2008.

[5] www.tinyvga.com/vgatiming

[6] http://www.xilinx.com

[7] https://www.digilentinc.com

SAŽETAK

Razvojna pločica Xilinx Spartan 3 Starter ima puno mogućnosti koje se mogu iskoristiti. Jedna od njih je bila i dizajn video upravljačkog sučelja pomoću FPGA. Pod tim se smatra da pomoću razvojne pločice Spartan 3 i programa pisanog u VHDL-u načinimo video upravljačko sučelje koje bi na monitoru priključenom na razvojnu pločice prikazivalo određenu sliku. Slika se može prikazati na raznim rezolucijama sa raznim frekvencijama osvježavanja, koje se odabiru pomoću prekidača koji se nalaze na razvojnoj pločici. Ona se može nadopuniti pomoćnim modulom koji prikazuje spektar ili sliku sa 24-bitnim bojama, za razliku od same pločice koja je u mogućnosti prikazati samo 3-bitni signal, R G B.

Ključne riječi:

- DCM - Digitalni menadžer takta
- VHSIC - hardverski jezik za opis programskog elementa
- FPGA - programabilno polje sa nizom vrata
- Brojač
- Generator takta
- Multiplekser
- VGA upravljač
- VGA sučelje

ABSTRACT

Xilinx Spartan 3 Starter development board has many feauters taht can be used. One of them is design of video controller with FPGA. Within it we know that using development board Spartan 3 and program coded in VHDL we make video controller interface that will show an image on the display connected to a development board. The image can be shown on different resolutions on different refresh rates, which are selected on the switches that are palced on development board. It can be upgraded with other modules that can be connected directly to the board and together they can show 24-bit depth color and images, as opposed the Spartan 3 board, which can show only 3-bit color.

Keywords:

- FPGA – field-programmable gate array
- VHDL - VHSIC hardware description language
- DCM – Digital Clock Manager
- Counter
- Clock generator
- Multiplexer
- VGA controller
- VGA port

ŽIVOTOPIS

Osobni podaci:

Ime i prezime:	Sven Pothorski
Datum i mjesto rođenja:	27. studenog 1987., Osijek
Adresa:	Zagorska 9, 31000 Osijek, Hrvatska
Tel.:	+385 91 7825414
E-mail:	sven.pothorski@gmail.com

Obrazovanje:

2002. - 2006.	Elektrotehnička i prometna škola Osijek, Smjer: elektrotehničar
2006. - 2009.	Elektrotehnički Fakultet Osijek, redovan student elektrotehnike, 3. godina Smjer: komunikacije

Dodatna edukacija:

2006. - 2008. CARNet EduPoint:

- Obrada audio/video zapisa
- Obrada slika pomoću Gimp-a / Photoshopa
- Izrada animacija pomoću Flash-a
- Izrada i objavljivanje web stranica pomoću FrontPagea
- Osnove CSS-a
- Osnove HTML-a
- Kolaboracija i komunikacija putem Interneta
- Osnove računalne sigurnosti na Internetu

Znanja i vještine:

- Aktivno poznavanje engleskog jezika u govoru i pisanju
- Pasivno poznavanje njemačkog jezika u govoru i pisanju
- Korištenje računala (hardware/software)

- Korištenje Windows i Linux OS-a, te MS Office 2010 i Adobe CS5 paketa, AutoCAD 2011, C++, VHDL, MatLab 2010, Eagle

Ostalo:

Hobi:	elektronika, biciklizam
Vozačka dozvola:	B kategorija
Bračno stanje:	neoženjen
Motivacija:	daljnje usavršavanje i edukacija

Radno iskustvo:

Rujan 2009. –	Studenski servis Osijek
	HT d.d. - snimanje DTK mreže

Sven Pothorski

_____