Fakultet elektrotehnike i računarstva
Zavod za elektroniku, mikroelektroniku,
računalne i inteligentne sustave

DIGITALNA LOGIKA

UPUTA ZA DRUGU LABORATORIJSKU VJEŽBU

dr.sc. Marko Čupić

Zagreb, 2013.

1. Zadatak

Zadatak druge laboratorijske vježbe jest modeliranje složenijih digitalnih sklopova strukturnim modeliranjem kroz više razina. Ovo znači da ćemo neki relativno složeni sklop najprije razložiti (dekomponirati) na nekoliko jednostavnijih sklopova. Potom ćemo svaki od tih sklopova neovisno jedan o drugome razložiti u još jednostavnije sklopove, i tako ponavljati proceduru dok ne dođemo do jednostavnih sklopova koje je jednostavno modelirati osnovnim logičkim sklopovima.

Ovaj pristup koristili bismo, primjerice, prilikom modeliranja današnjeg računala. Tako bismo najprije uočili da se današnje računalo sastoji od matične ploče, memorije, niza kartica spojenih u odgovarajuće utore, tvrdih diskova, optičkih jedinica i sl. Zatim bismo krenuli u modeliranje svakog od tih dijelova zasebno. Primjerice, na samoj matičnoj ploči uočili bismo niz upravljačkih sklopova (engl. *chipset*), procesor itd. Pogledamo li sada sam procesor, opet ćemo uočiti čitav niz podsustava o kojima ćete više čuti na drugim predmetima. U konačnici, stigli bismo do osnovnih logičkih sklopova.

Zadatak ove vježbe jest modelirati jedan složeniji sustav čija je namjena prijenos poruka. Sličan primjer i njegovo rješenje preporuča se kao pripremu pogledati u [2], zadatak 3.25 stranica 101 (zanemarite VHDL opise sklopova).

Prije dolaska na termin laboratorijske vježbe potrebno je:

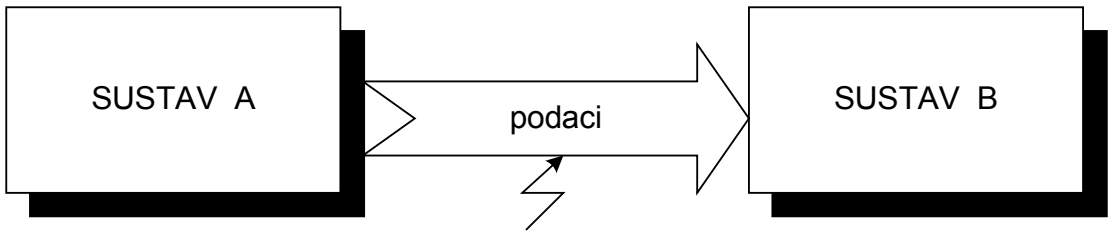
- pročitati ovu uputu,
- odgovoriti na sva postavljena pitanja (u za to predviđen prostor),
- riješiti sve postavljene zadatke (izrada tablice istinitosti, očitavanje algebarskog zapisa i slično),
- nacrtati sve sheme u za to predviđene okvire,
- popuniti tablicu koja se nalazi na kraju upute,
- u sustavu VHDLLab2 modelirati sklopove koji su navedeni na predzadnjoj stranici upute te
- provjeriti ispravnost modeliranih sklopova izradom ispitnih sklopova.

Napominjemo još jednom: u sustavu VHDLLab2 nije potrebno napraviti modele svih sklopova koje je potrebno razviti u okviru ove upute: što je točno potrebno modelirati u sustavu VHDLLab2 zadano je na kraju upute.

U termin laboratorijske vježbe sa sobom obavezno ponesite ovu popunjenu uputu.

2. Sustav za prijenos podataka

Sustav A razmjenjuje podatke sa sustavom B putem komunikacijskog kanala na kojem je moguća pojava smetnji (slika 1). Pri pojavi smetnje, neki od bitova u podatcima koji se šalju promijenit će svoju vrijednost: logička jedinica postat će logička nula i obrnuto. Sustav A i sustav B razmjenjuju poruke širine 4-bita. Kako bi povećali otpornost na pogreške, umjesto slanja čistih poruka sustav A na poruku će primijeniti Hammingov kôd nakon čega će sustavu B poslati tako zaštićenu kodnu riječ.



Slika 1. Sustavi čiji će se dijelovi projektirati

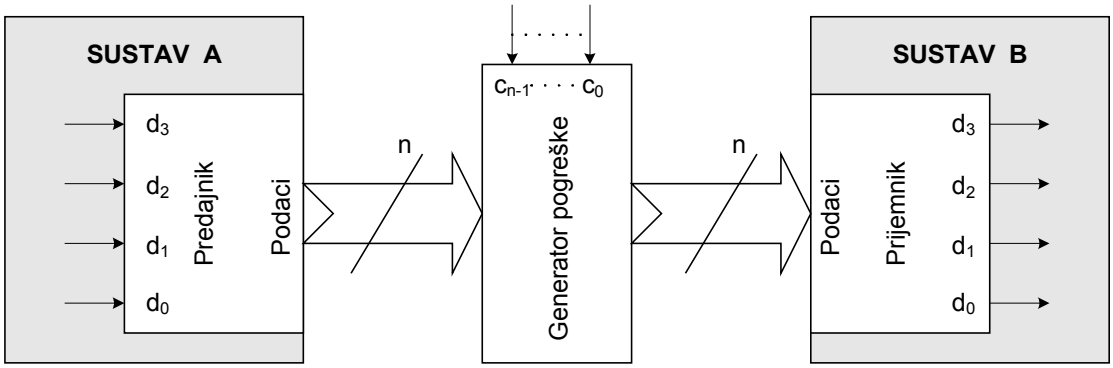
Cjelokupni sustav koji se sastoji sustava A, sustava B te komunikacijskog kanala modelirat ćemo strukturno. Da bismo to mogli, sustav A, sustav B te sam prijenos podataka potrebno je dekomponirati u jednostavnije sklopove.

Od čega se sastoji sustav A? Sam sustav sigurno ima niz digitalnih sklopova koji generiraju i obrađuju poruke; no ono što nas ovom prilikom zanima jest sljedeće – sustav A ima podsustav koji preuzima 4-bitnu poruku, primjenjuje zaštitno kodiranje i na izlazu generira odgovarajuću Hammingovu kodnu riječ. Taj ćemo sklop (ili podsustav) nazvati 'Predajnik'.

Od čega se sastoji sustav B? I taj sustav obavlja određenu funkciju koja nam u ovom trenutku nije važna; važno je da taj sustav sigurno ima podsustav koji s komunikacijskog kanala prima poruku (koja može biti i oštećena), na odgovarajući način primjenjuje zaštitno kodiranje kako bi očitao što je zapravo predajnik poslao, i taj podatak prosljeđuje dalje sklopovima sustava B. Taj ćemo sklop (ili podsustav) u sustavu B nazvati 'Prijemnik'.

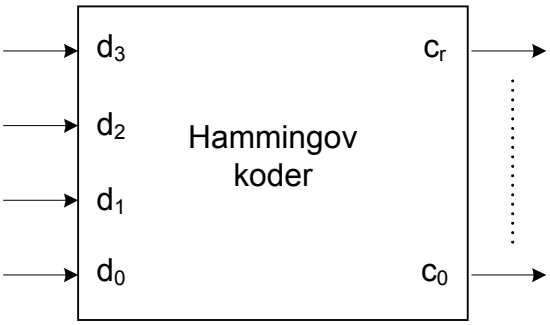
Još na odgovarajući način moramo modelirati sam komunikacijski kanal. Kako zapravo funkcionira komunikacijski kanal? Kroz komunikacijski kanal šalje se *n*-bitni podatak. Idealno, ako nema djelovanja pogrešaka, komunikacijski kanal možemo zamisliti kao *n* paralelnih vodiča – za svaki bit po jedan vodič. Međutim, naš komunikacijski kanal mora omogućiti modeliranje pogrešaka, i to na bilo kojem od *n* bitova (ili na više njih istovremeno). Stoga ćemo komunikacijski kanal modelirati kao digitalni sklop koji ima *n* podatkovnih ulaza, *n* podatkovnih izlaza, te *n* upravljačkih ulaza (za svaki podatkovni ulaz imat će jedan upravljački ulaz koji će odrediti hoće li se na odgovarajući izlaz propustiti sam podatak ili njegov komplement, čime možemo modelirati djeluje li pogreška na taj bit ili ne).

Ovako razloženi sustav za prijenos podataka prikazan je na slici 2.



Slika 2. Sustavi čiji će se dijelovi projektirati

Strukturno modeliranje sklopova iterativan je proces. Do sada smo čitav sustav dekomponirali u tri dijela: *predajnik*, *generator pogreške* te *prijemnik*. Predajnik i prijemnik još su uvijek prilično složeni sklopovi. Pokušajmo sada strukturno modelirati svaki od tih sklopova zasebno. Ako se prisjetimo Hammingovog kodiranja, prilikom izračuna zaštićene kodne riječi kao i prilikom ispravljanja pogrešaka radimo jednu zajedničku operaciju: na temelju podatkovnih bitova (bilo originalnih, bilo očitanih s komunikacijskog kanala) računamo zaštitne bitove. Stoga ćemo ovu operaciju izdvojiti u zaseban sklop: '*Hammingov koder*'. Taj će sklop na ulazu dobiti 4 podatkovna bita a na izlazu generirati *r+1* zaštitni bit (i samo zaštitne bitove; ovaj sklop ne generira cjelokupnu Hammingovu kodnu riječ). *U ovoj konkretnoj vježbi, koliko zaštitnih bitova treba generirati Hammingov koder, odnosno koliki je r+1*? *Na temelju čega se to određuje?* Sučelje ovog sklopa prikazano je na slici 3.

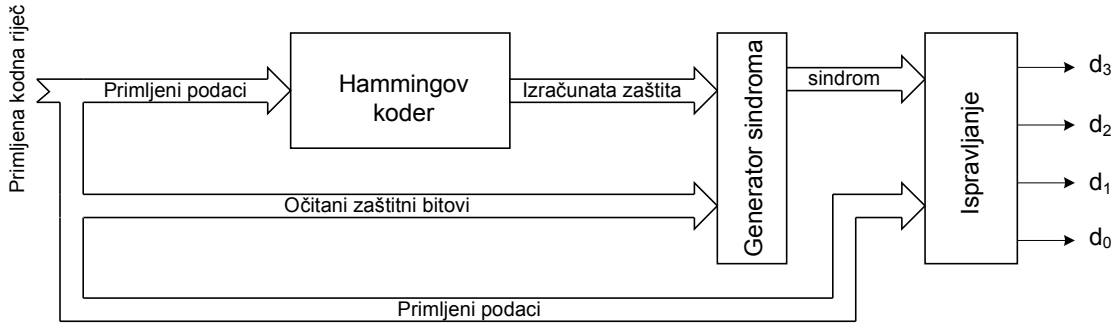


Slika 3. Hammingov koder

Vaš je prvi zadatak modelirati komponentu '*Hammingov koder*' strukturno, uporabom osnovnih logičkih sklopova. Shemu nacrtajte u odgovarajući okvir u nastavku ove pripreme. Na shemi se moraju vidjeti ulazi d_0 do d_3, izlazi c_0 do c_r te način na koji se na temelju ulaza generiraju izlazi (drugim riječima, shema koja prikazuje kako se osnovnim logičkim sklopovima, u koje ubrajamo i isključivo-ili, na temelju ulaza generiraju izlazi). Osnovni logički sklopovi mogu imati i više od dva ulaza.

Potom strukturno modelirajte komponentu '*Predajnik*' koristeći komponentu '*Hammingov koder*'. Shemu također nacrtajte u za to predviđeni okvir u nastavku ove pripreme. U okviru u kojem ćete nacrtati strukturni model *Predajnika*, komponentu '*Hammingov koder*' koristit ćete kao crnu kutiju: nacrtat ćete simbol na kojem moraju biti naznačeni ulazi d_0 do d_3 te izlazi c_0 do c_r (građu te komponente u ovom modelu ne prikazujete). Umjesto toga, shema treba prikazati kako se temeljem ulaza d_0 do d_3 te potrebnih komponenata generiraju izlazi koji čine cjelokupnu Hammingovu kodnu riječ.

Za razliku od komponente '*Predajnik*' koja je relativno jednostavna, komponenta '*Prijemnik*' je nešto složenija. Jedna moguća dekompozicija predajnika prikazana je na slici 4.



Slika 4. Struktura prijemnika

'*Prijemnik*' je dekomponiran na sljedeći način: na sklop '*Hammingov koder*' dovode se primljeni podatkovni bitovi. '*Hammingov koder*' za njih računa zaštitne bitove. Izračunati zaštitni bitovi zajedno s očitanim zaštitnim bitovima vode se na sklop '*Generator sindroma*' čiji je zadatak na temelju izračunatih i očitanih zaštitnih bitova izračunati sindrom. Uočite: ulazi ovog sklopa su samo očitani i primljeni zaštitni bitovi dok izlaz čine bitovi sindroma. *Kako se na temelju očitanih i primljenih zaštitnih bitova računa sindrom? Koliko ulaza i koliko izlaza ima ovaj sklop?* Potom se na sklop '*Ispravljanje*' dovode primljeni **podatkovni bitovi** te izračunati sindrom. Na temelju tih podataka sklop na svom izlazu generira ispravljene **podatkovne bitove**. Uočite: ovaj sklop ne dobiva cjelokupnu primljenu riječ već samo primljene podatkovne bitove te bitove izračunatog sindroma.

Prilikom realizacije sklopa '*Ispravljanje*' razmislite o sljedećem. S obzirom da su u Hammingovoj kodnoj riječi poslana četiri podatkovna bita (označimo ih d_0, d_1, d_2 i d_3), taj sklop mora također imati četiri izlaza (označimo ih x_0, x_1, x_2 i x_3) pri čemu će izlaz x_i odgovarati primljenom podatkovnom bitu d_i ako je taj ispravan odnosno njegovom ispravku ako je taj neispravan.

Razmislite i odgovorite na sljedeće pitanje: je li izlaz x_0 Booleova funkcija od svih primljenih podatkovnih bitova i svih bitova sindroma, ili ovisi samo o nekim od tih bitova? Odgovor na to pitanje kao i isto pitanje za preostale izlaze napišite u nastavku na predviđenu crtu. Pomozite si tako da razmislite o sljedećem: kako znamo je li primljeni podatkovni bit d_0 ispravan ili nije, odnosno o čemu ovisi što ćemo generirati na izlazu x_0?

Izlaz x_0 je funkcija od _____ *(broj)* varijabli; one su: _____ *(imena)*.
Izlaz x_1 je funkcija od _____ *(broj)* varijabli; one su: _____ *(imena)*.
Izlaz x_2 je funkcija od _____ *(broj)* varijabli; one su: _____ *(imena)*.
Izlaz x_3 je funkcija od _____ *(broj)* varijabli; one su: _____ *(imena)*.

Sada kada ste sigurni u odgovor, napišite na sljedećoj stranici tablicu istinitosti za izlaz x_0. Tablica ne bi smjela imati više od 16 redaka!

Tablica istinitosti za izlaz x_0:

Očitajte algebarski zapis te funkcije u obliku minimalne sume produkata (K-tablicu prikažite u nastavku). Potom algebarski svedite taj zapis na zapis oblika $x_0 = d_0$ *Ex-ILI ostatak* (i to prikažite u nastavku). Pogledajte dobro što ste dobili kao *ostatak*: možete li odgovoriti na pitanje zašto ste baš to dobili? Razmislite što je rezultat izračuna d_0 *Ex-ILI 0* a što d_0 *Ex-ILI 1*. Povežite Vaš odgovor s činjenicom da upravo *ostatak* predstavlja tu konstantu.

Temeljem prethodnog razmatranja u nastavku nacrtajte sklop koji se temelji na sklopu Isključivo-ILI a koji generira izlaz x_0.

(tražena slika)

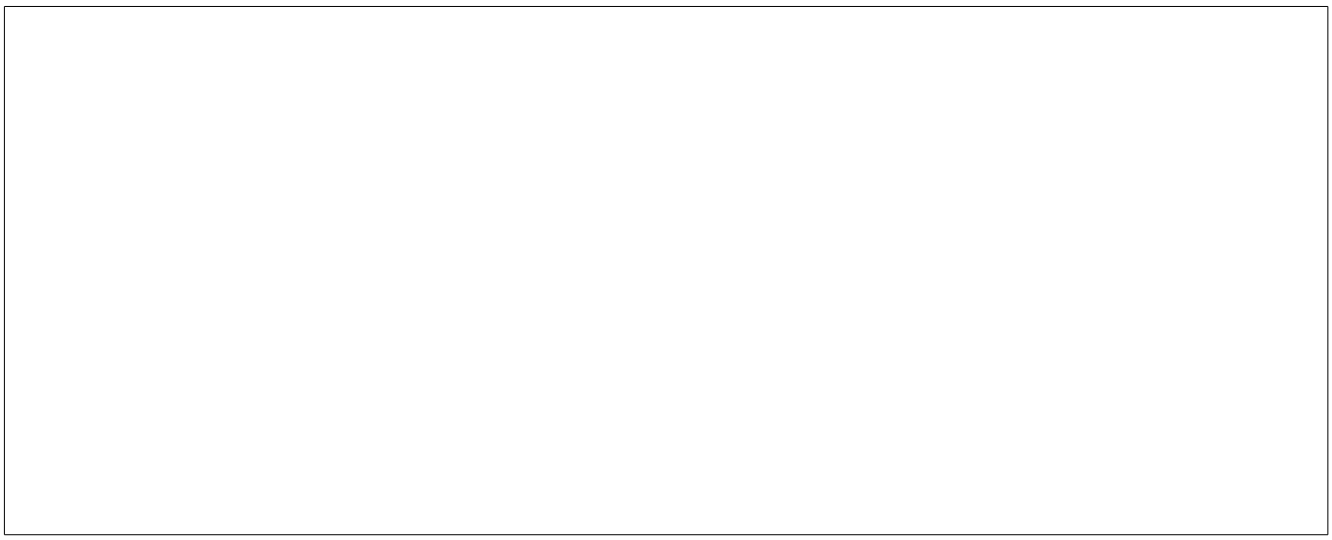
Jedan od mnogih naziva koji se koristi za sklop oblika d_0 *Ex-ILI ostatak* je i *upravljivi invertor*. Možete li objasniti od kuda potiče taj naziv?

(objašnjenje)

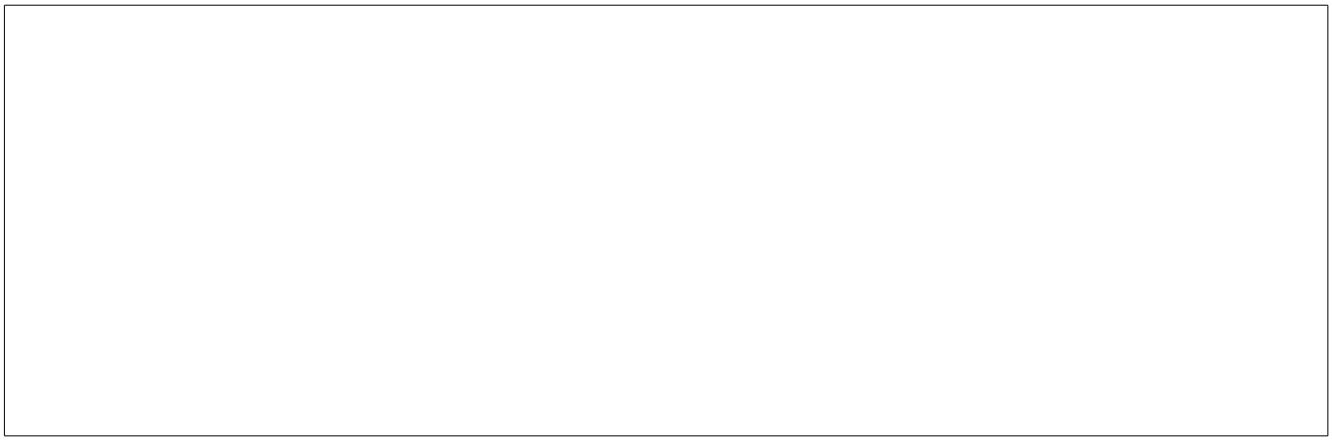
Sada se možemo vratiti na strukturno modeliranje dijelova sklopa '*Prijemnik*'. Napravite sljedeće:

- strukturno modelirajte sklop '*Generator sindroma*' uporabom osnovnih logičkih sklopova (u njih ubrajamo i isključivo-ili),
- strukturno modelirajte sklop '*Ispravljanje*' uporabom osnovnih logičkih sklopova (u njih ubrajamo i isključivo-ili) te
- strukturno modelirajte sklop '*Prijemnik*' uporabom sklopova '*Hammingov koder*', '*Generator sindroma*' te '*Ispravljanje*'.

Kako biste dovršili modeliranje do kraja, sklop '*Generator pogreške*' koji je zapravo model komunikacijskog kanala također modelirajte strukturno uporabom osnovnih logičkih sklopova. Pri tome sklop *isključivo-ili* možete smatrati osnovnim logičkim sklopom.



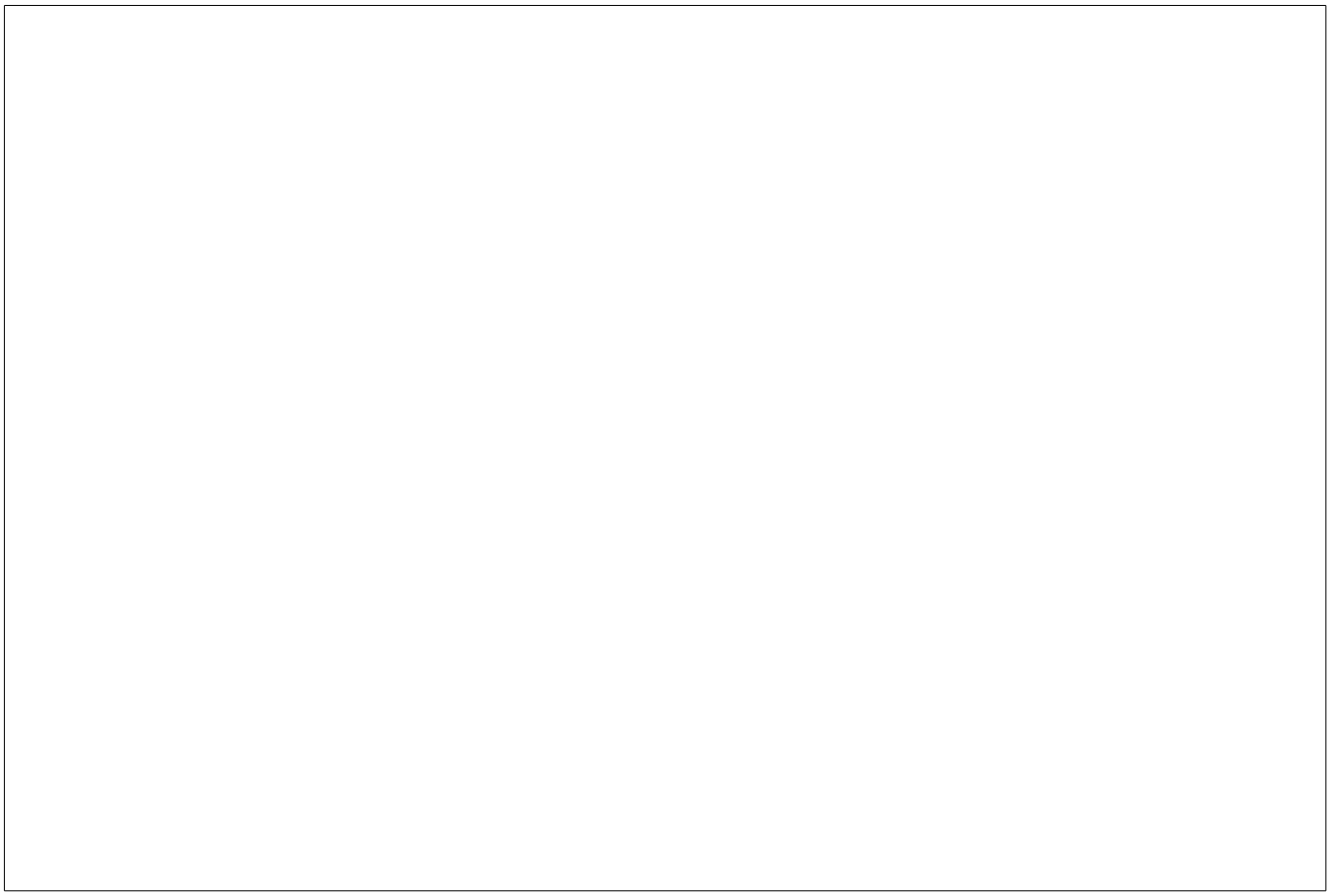
Shema komponente *Hammingov koder*



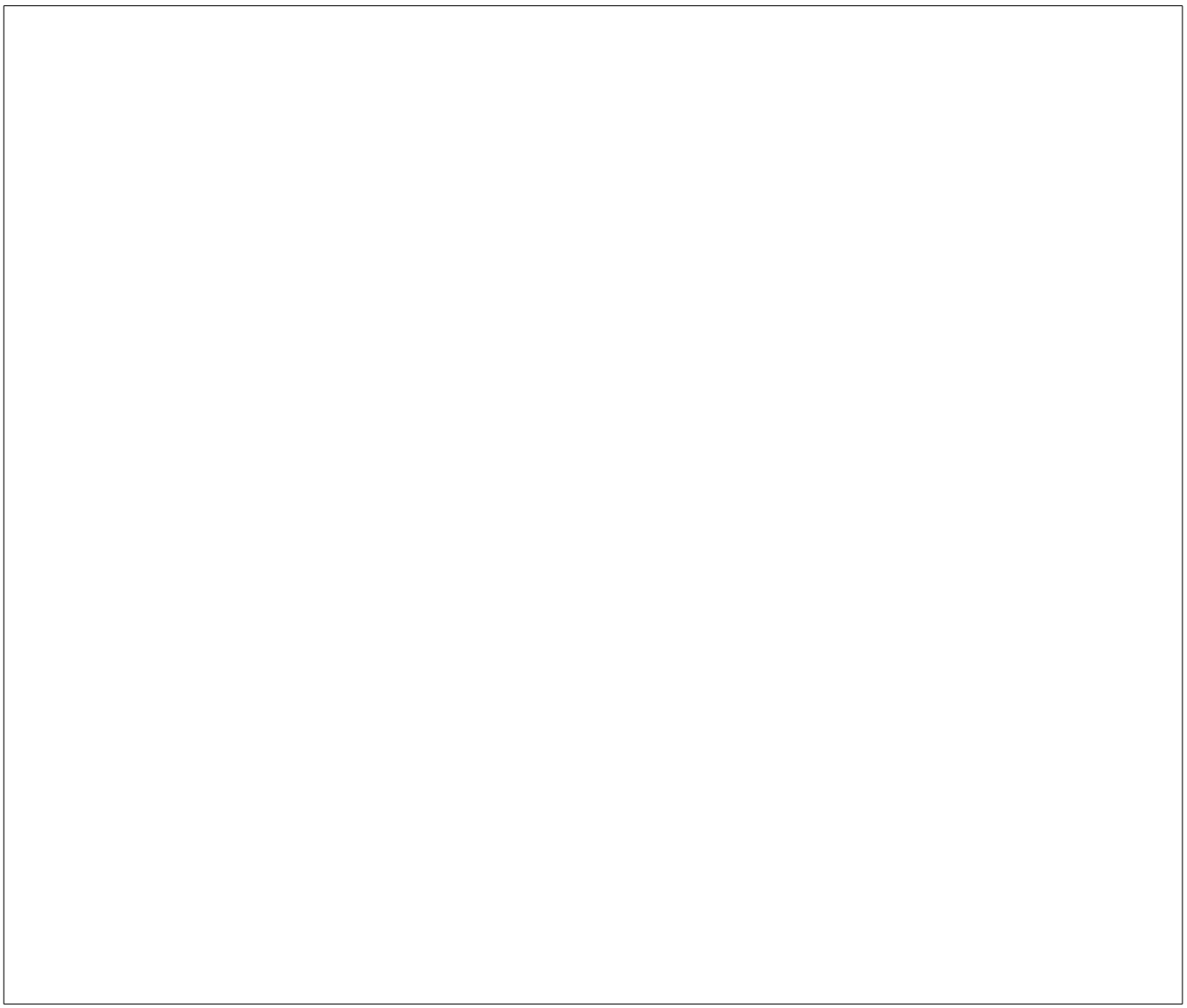
Shema komponente *Predajnik*

Shema komponente *Generator sindroma*

Shema komponente *Generator pogreške*



Shema komponente *Ispravljanje*



Shema komponente *Prijemnik*

Prilikom ispitivanja rada sustava na upravljačke ulaze *Generatora pogreške* tipično bismo dovodili riječi *skoro-jednojediničnog* kôda. Skoro-jednojedinični *n*-bitni kôd je kôd čije su riječi širine *n* bitova i koje imaju vrijednost `1` postavljenu na najviše jednom mjestu. Tako, primjerice, 4-bitnom skoro-jednojediničnom kôdu pripadaju kodne riječi `0000`, `1000`, `0100`, `0010`, `0001`; kodne riječi `1100`, `0101`, `1111` i sl. ne pripadaju ovom kôdu jer je kod njih više od jednog bita postavljeno na vrijednost `1`. Razlog za uporabu ovakvog koda pri ispitivanju trebao bi biti jasan uzmemo li u obzir zaštitnu moć Hammingovog kôda odnosno broj pogrešaka koje taj kôd može ispraviti.

U sustavu VHDLLab2 nije potrebno realizirati čitav sustav. Umjesto toga, potrebno je u potpunosti modelirati sklop *Prijemnik* (i sve sklopove koji su za to potrebni) te ispitati njegov rad. Da bi to bilo moguće, prije dolaska na termin vježbe pripremite:

a) dva podatka koja ćete dovesti na ulaz prijemnika u kojima nema pogreške,
b) dva podatka u kojima znate da je nastupila pogreška na nekom podatkovnom bitu,
c) dva podatka u kojima znate da je nastupila pogreška na nekom zaštitnom bitu te
d) dva podatka u kojima znate da su nastupile dvije pogreške.

Upišite te podatke u sljedeću tablicu i zaokružite bitove na kojima se dogodila pogreška (u predzadnjem stupcu tablice). Potom napravite ispitni sklop (engl. *testbench*) koji svakih 100 ns na ulaz dovodi jedan od podataka iz te tablice. Odsimulirajte rad sustava i provjerite slažu li se dobiveni rezultati simulacije s onima koje ste izračunali.

	4-bitni podatak koji šaljemo	***Zaštićena kôdna riječ***	***Riječ koju prima prijemnik s komunikacijskog kanala***	***Podatak koji bi prijemnik trebao dati na izlazu***
a.1				
a.2				
b.1				
b.2				
c.1				
c.2				
d.1				
d.2				

Prilikom rada u sustavu VHDLLab2, pridržavajte se sljedećih pravila.

1. Vježbu napravite u novom projektu (nazovite ga `lab2`).
2. Sve komponente koje radite u okviru ove vježbe *moraju* biti u istom projektu; u suprotnom jednostavnije komponente nećete moći koristiti pri modeliranju složenijih komponenata jer sustav "vidi" samo komponente koje su napisane u istom projektu.
3. Najprije napravite sklop *HammingovKoder*. Kad ste gotovi s crtanjem sheme, obavezno snimite shemu. Pokušajte sklop prevesti (akcija *Compile*). Ako sustav javlja pogrešku – ispravite shemu, snimite pa ponovno prevedite. Ne idite dalje dok ne uspijete uspješno prevesti sklop. Ako je shemu moguće prevesti bez pogrešaka, za sklop napravite ispitni sklop i ispitajte njegov rad za barem četiri različite pobude (namjestite ih u 0., 100., 200. i 300. nanosekundi). Pokrenite simulaciju i provjerite je li rezultat u skladu s očekivanim. Ako nije, korigirajte shemu i ponovite postupak. Ne idite dalje tako dugo dok simulacija rada sklopa nije u skladu s očekivanjima.
4. Postupak opisan prethodnom koraku primijenite na sve ostale komponente. Sklopove koje ste već napraviti u novim shemama bit će Vam dostupni u popisu komponenti s desne strane uređivača (na mjestu gdje inače birate osnovne logičke sklopove).

Literatura:

[1] Peruško, Glavinić: *Digitalni sustavi*. Školska knjiga, 2005.

[2] Čupić, *Digitalna elektronika i Digitalna logika. Zbirka riješenih zadataka*. Kigen, 2006.