

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

SVE SCHEME OVDJE I NA  
VHDLlab-u NISU IDENTIČNE  
(NEKE SU „NAOPAČKE“), BIT  
JE DA SU PRAVI ULAZI I IZLAZI  
SPOJENI TAKO DA NEMA  
RAZLIKE U RJEŠENJU

## **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 uređaje, 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 podacima 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.

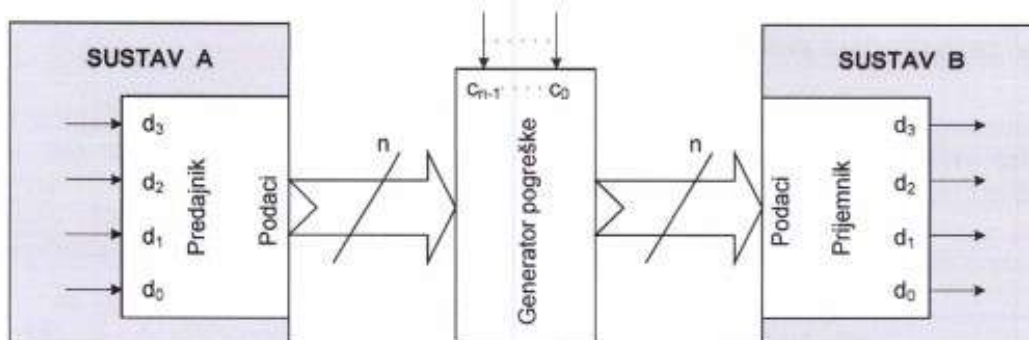
Od čega se sastoji sustav A? Sam sustav sigurno ima niz digitalnih sklopova koji generiraju i obrađuju poruke; no ono što nas ovom prilikom zanima jest sljedeće – sustav A ima podsustav koji preuzima 4-bitnu poruku, primjenjuje zaštitno kodiranje i na izlazu generira odgovarajuću Hammingovu kodnu riječ. 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	<u>4</u>	(broj) varijabli; one su:	<u><math>d_0, s_0, s_1, s_2</math></u>	(imena).
Izlaz $x_1$ je funkcija od	<u>4</u>	(broj) varijabli; one su:	<u><math>d_1, s_0, s_1, s_2</math></u>	(imena).
Izlaz $x_2$ je funkcija od	<u>4</u>	(broj) varijabli; one su:	<u><math>d_2, s_0, s_1, s_2</math></u>	(imena).
Izlaz $x_3$ je funkcija od	<u>4</u>	(broj) varijabli; one su:	<u><math>d_3, s_0, s_1, s_2</math></u>	(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  $x_0$ :

$d_0$	$s_0$	$s_1$	$s_2$	$x_0$	(indeksi dobiveni od Sindroma)
0	0	0	0	0	0 - (nema greške)
0	0	0	1	0	1
0	0	1	0	0	2
0	0	1	1	1	3
0	1	0	0	0	4
0	1	0	1	0	5
0	1	1	0	0	6
0	1	1	1	0	7
1	0	0	0	1	0 - (nema greške)
1	0	0	1	1	1
1	0	1	0	1	2
1	0	1	1	0	3
1	1	0	0	1	4
1	1	0	1	1	5
1	1	1	0	1	6
1	1	1	1	1	7

nepotrebno

bitovi	$c_0$	$c_1$	$d_0$	$c_2$	$d_1$	$d_2$	$d_3$
indeksi	1	2	3	4	5	6	7

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.

	$d_0$	$d_1$	$d_2$	$d_3$	$s_0 s_1$
00	0	0	0	0	
10	0	0	0	1	
11	1	1	1	0	
01	1	1	1	1	

$\bar{s}_0 \bar{s}_1 \bar{s}_2 \bar{d}_0$

$\bar{s}_1 \bar{d}_0$   $s_0 d_0$   $\bar{s}_2 d_0$

$$\begin{aligned}
 x_0 &= s_0 d_0 + \bar{s}_1 d_0 + \bar{s}_2 d_0 + \bar{s}_0 s_1 s_2 \bar{d}_0 \\
 &= d_0 (s_0 + \bar{s}_1 + \bar{s}_2) + \bar{d}_0 \bar{s}_0 s_1 s_2 \\
 &= d_0 (s_0 + \bar{s}_1 \bar{s}_2) + \bar{d}_0 (\bar{s}_0 s_1 s_2) \\
 &= d_0 (\bar{s}_0 \bar{s}_1 \bar{s}_2) + \bar{d}_0 (\bar{s}_0 s_1 s_2) \\
 &= d_0 \oplus (\bar{s}_0 s_1 s_2)
 \end{aligned}$$

- kao ostatak u izrazu smo dobili  $\bar{s}_0 s_1 s_2$  što odgovara logičkoj vrijednosti true za broj 3 ( $011$ ,  $s_0=0$ ,  $s_1=1$ ,  $s_2=1$ )

•  $d_0 \oplus 0 = d_0$

$d_0$	0	$f = d_0 \oplus 0$
0	0	0
1	0	1

A	B	A ⊕ B
0	0	0
0	1	1
1	0	1
1	1	0

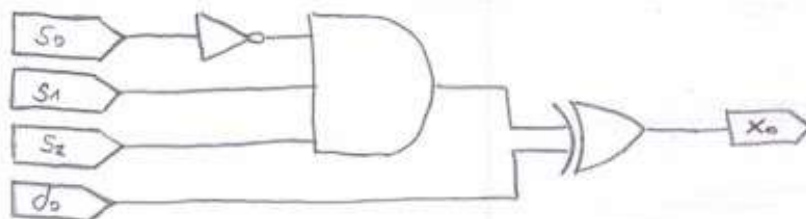
$d_0 \oplus 1 = \bar{d}_0$

$d_0$	1	$f = d_0 \oplus 1$
0	1	1
1	1	0

- to znači da kada je ostatak (sindrom) istinit za član, on postaje suprotna vrijednost tj. postaje točna (ispravljena) vrijednost u odnosu na pristigli

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?

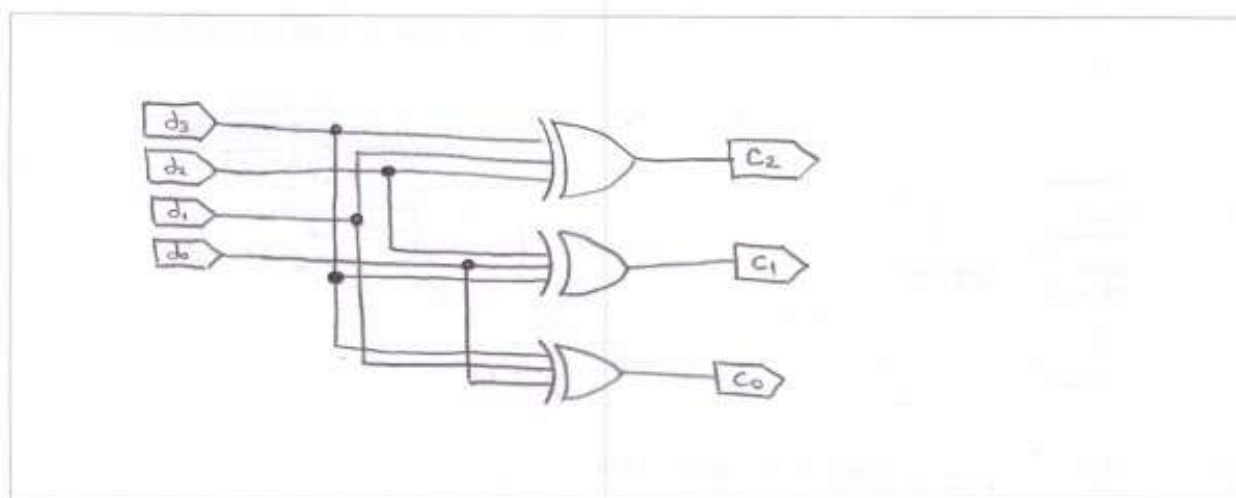
(objašnjenje)

Potiče od toga da kada se kao ostatak dovede istinita logička vrijednost  $d_0$  postaje  $\bar{d}_0$ .  
To omogućava da možemo pomoću XOR sklopova upravljati kada želimo invertirati vrijednosti.

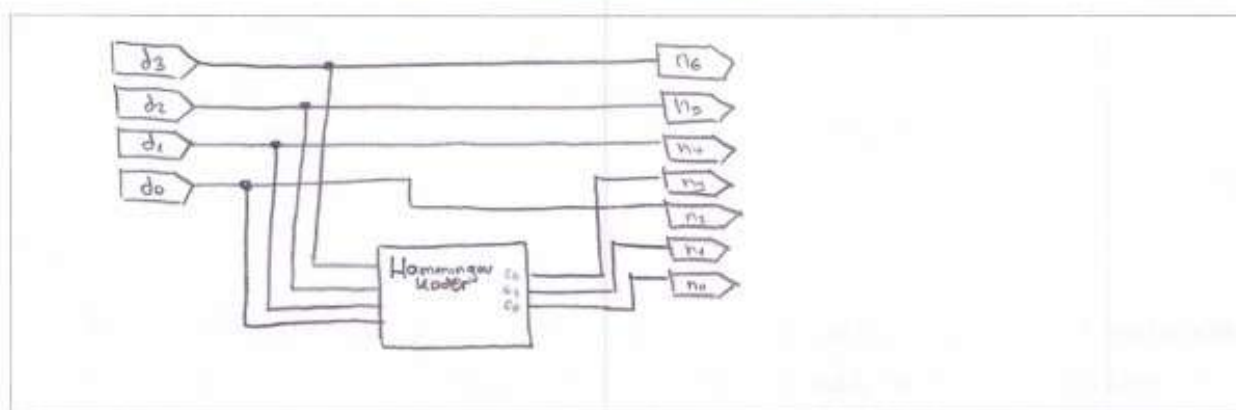
Sada se možemo vratiti na strukturno modeliranje dijelova sklopa 'Prijemnik'. Napravite sljedeće:

- 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'.

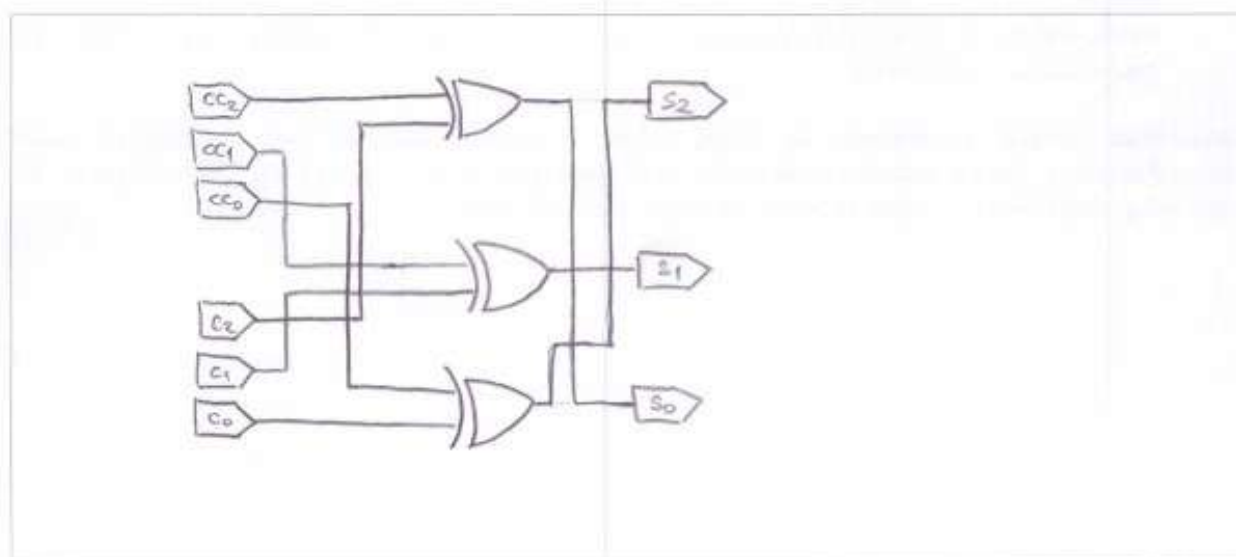
Kako biste dovršili modeliranje do kraja, sklop 'Generator pogreške' koji je zapravo model komunikacijskog kanala također modelirajte strukturno uporabom osnovnih logičkih sklopova. Pri tome sklop *isključivo-ili* možete smatrati osnovnim logičkim sklopom.



Shema komponente *Hammingov koder*

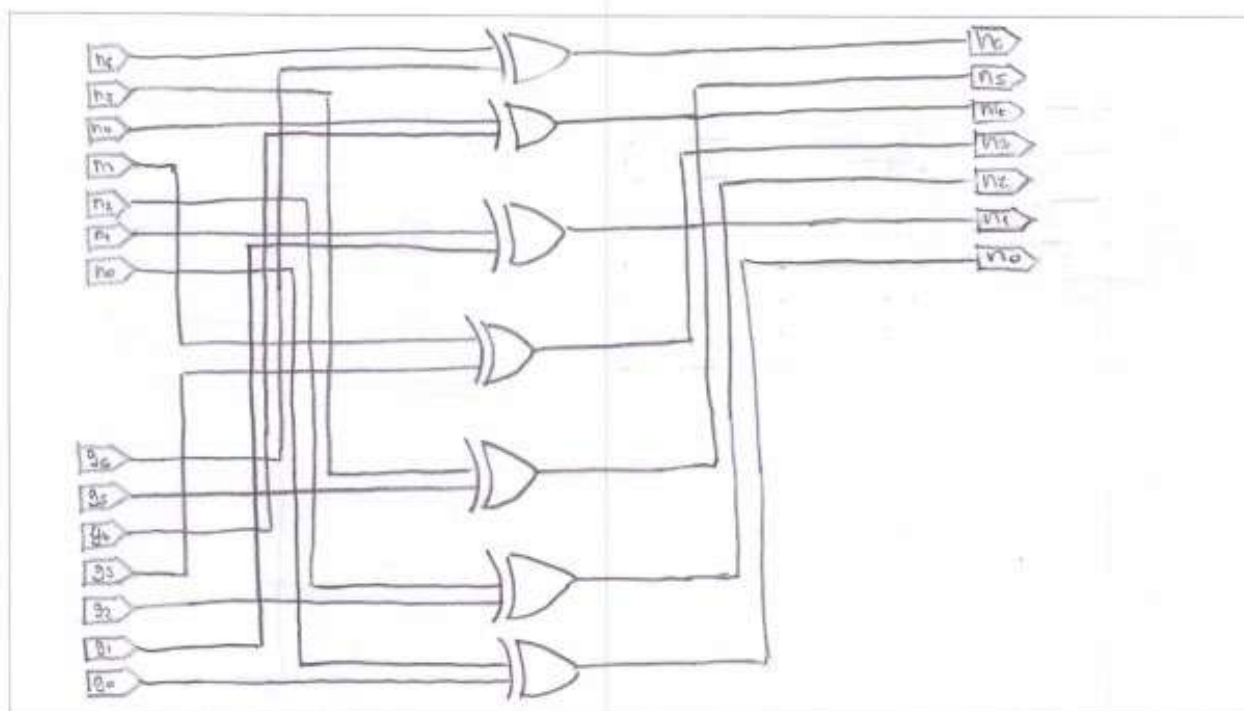


Shema komponente *Predajnik*

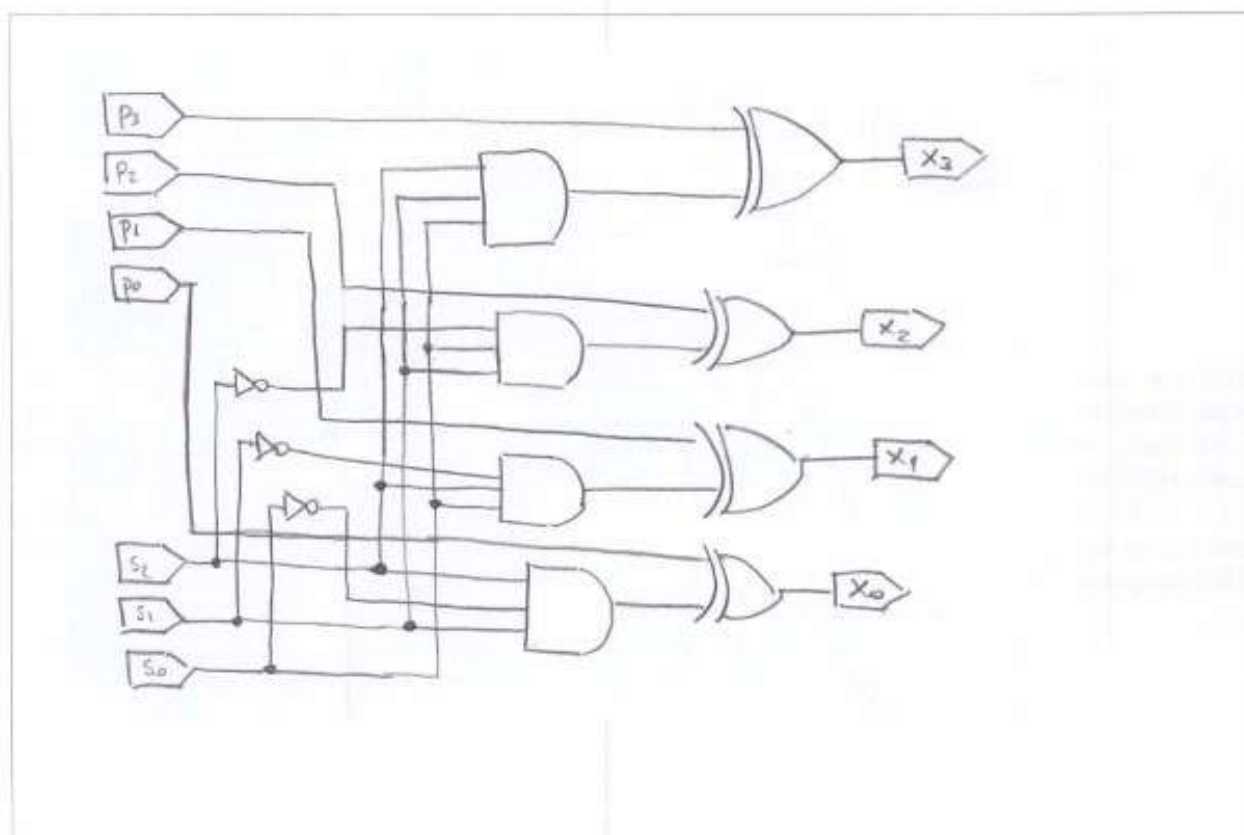


Shema komponente *Generator sindroma*

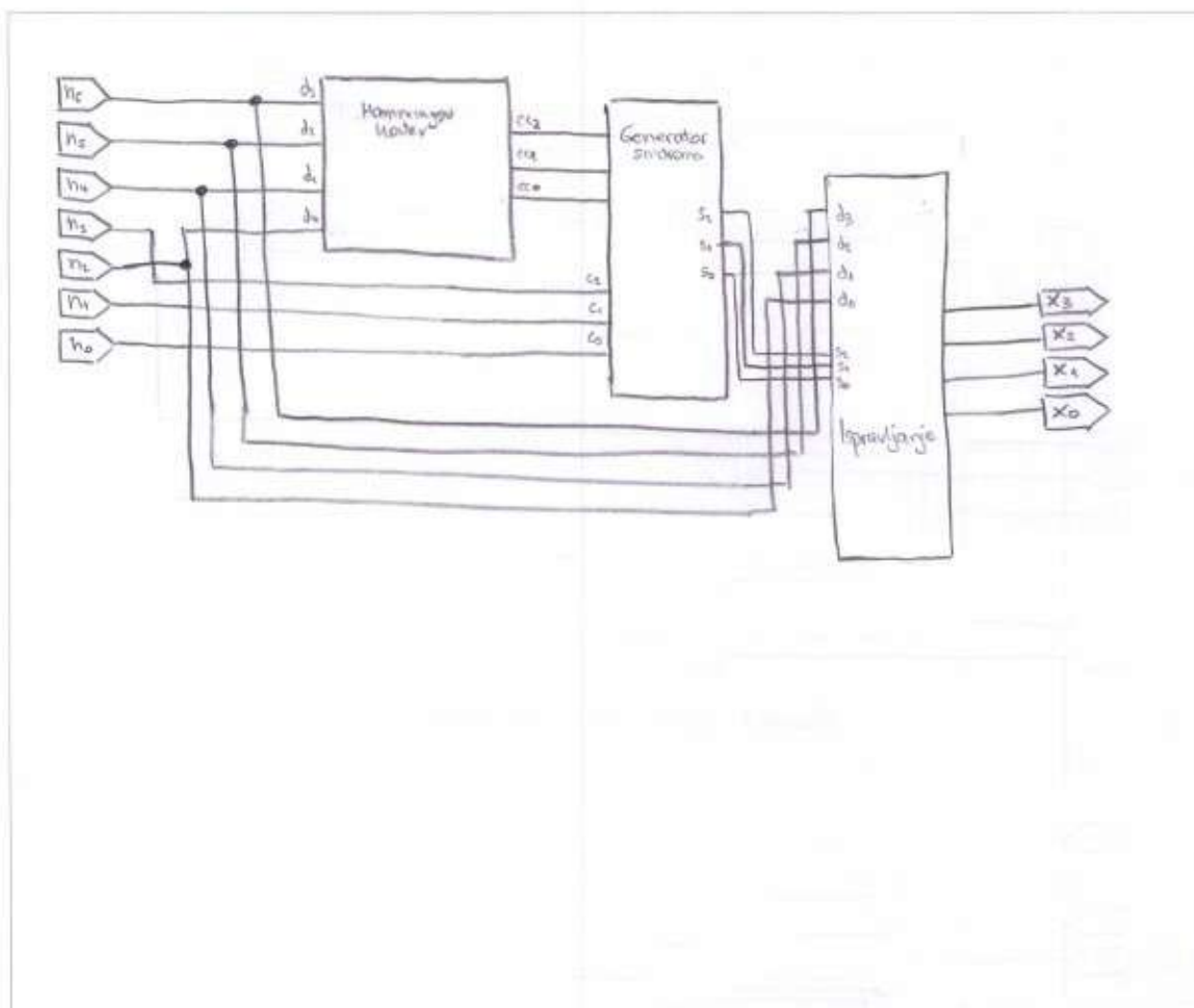




Shema komponente *Generator pogreške*



Shema komponente *Ispravljanje*



Shema komponente *Prijemnik*

Prilikom ispitivanja rada sustava na upravljačke ulaze *Generatora pogreške* tipično bismo dovodili riječi *skoro-jednoliničnog* kôda. Skoro-jednolinični  $n$ -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-jednoliničnom 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 kôda 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:

- dva podatka koja ćete dovesti na ulaz prijemnika u kojima nema pogreške,
- dva podatka u kojima znate da je nastupila pogreška na nekom podatkovnom bitu,
- dva podatka u kojima znate da je nastupila pogreška na nekom zaštitnom bitu te
- 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 <small><math>A_0 A_1 A_2 A_3 P_0 P_1 P_2 P_3 A_4 A_5 A_6 A_7</math></small>	Podatak koji bi prijemnik trebao dati na izlazu <small><math>X_0 X_1 X_2 X_3</math></small>
0-100ns	a.1	0000	00000000	00000000	0000 ✓
100-200ns	a.2	1010	10110110	10110110	1010 ✓
200-300ns	b.1	0101	01001011	01010101	0101 ✓
300-400ns	b.2	1111	11111111	11111111	1111 ✓
400-500ns	c.1	0000	00000000	00000000	0000 ✓
500-600ns	c.2	1010	10110110	10110110	1010 ✓
600-700ns	d.1	0101	01001011	01010101	1100 ✓
700-800ns	d.2	1111	11111111	11111100	1100 ✓

Prilikom rada u sustavu VHDLLab2, pridržavajte se sljedećih pravila.

- Vježbu napravite u novom projektu (nazovite ga *lab2*).
- 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.
- Najprije napravite sklop *HammingovKoder*. Kad ste gotovi s crtanjem sheme, obavezno snimate shemu. Pokušajte sklop prevesti (akcija *Compile*). Ako sustav javlja pogrešku – ispravite shemu, snimate 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 očekivanjem. Ako nije, korigirajte shemu i ponovite postupak. Ne idite dalje tako dugo dok simulacija rada sklopa nije u skladu s očekivanjima.
- 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).



## Hammingov kod

$$2^r \geq k+r+1$$

$$n = k+r$$

ukupno b.    zaštitni b.  
↑  
informacijski b.

$$k = 4$$

$$r = ?$$

$$n = ?$$

$$\begin{aligned} 2^r &\geq k+r+1 \\ 2^r - r - 1 &\geq 4 \\ 2^3 - 3 - 1 &\geq 4 \\ 4 &\geq 4 \quad \checkmark \end{aligned}$$

$$r = 3$$

$$n = 4+3$$

$$n = 7$$

nepotrebno

Dokaz: (ovo ne vrijedi)

$$\begin{array}{ccccccc} c_0 & c_1 & d_0 & c_2 & d_1 & d_2 & d_3 \\ \hline & & & & & & \\ & & & & & & \\ & & & & & & \\ & & & & & & \\ & & & & & & \\ & & & & & & \\ & & & & & & \\ & & & & & & \end{array}$$

$n = 7$   
 $r = 3$

- Trebamo 3 zaštitna bita u ovom slučaju.
- Odredujemo pomoću formule  $2^r - r - 1 \geq k$ .
- Na temelju početnih (cc) i primljenih (c) bitova se dobiva sindrom tako da idemo bit po bit i vršimo XOR operaciju ( $cc_i \oplus c_i$ ) i na kraju dobivenim bitovima obrnemo redoslijed.
- Imamo 6 ulaza ( $c_0 - c_3$  i  $cc_0 - cc_3$ ) i 3 izlaza ( $s_0 - s_2$ ) jer su izlazi rezultati operacije XOR svakih dva ulaza s obrnutim indeksom ( $s_0 = c_2 \oplus cc_2$ ,  $s_1 = c_1 \oplus cc_1$ ,  $s_2 = c_0 \oplus cc_0$ ).
- $x_0$  je bodovna funkcija od primljenog podatka do i svih brojeva sindroma jer nam sindrom govori koji je bit kriv a primljeni podatak mijenjamo u skladu sindrom predstavlja njegovu poziciju u Hammingovom kodu, u suprotnom ga samo propuštamo dalje.

nepotrebno

slučaj zatačke i mijenjanje primjera za  
ulaz i ispravljanje

$s_0$	$s_1$	$s_2$	
0	0	0	/ svi su ispravni
0	0	1	$c_0$
0	1	0	$c_1$
0	1	1	$d_0$
1	0	0	$c_2$
1	0	1	$d_1$
1	1	0	$d_2$
1	1	1	$d_3$

- Koristimo skoro-jednolični kod pri korištenju generatora pogreške jer Hammingov kod ima sposobnost ispravljanja samo jednog bita bez da se promijeni pravilna vrijednost.

$$x_0 = d_0 \oplus (\overline{s_0} \overline{s_1} s_2)$$

$$x_1 = d_1 \oplus (\overline{s_0} s_1 \overline{s_2})$$

$$x_2 = d_2 \oplus (s_0 s_1 \overline{s_2})$$

$$x_3 = d_3 \oplus (s_0 s_1 s_2)$$

# Testovi

## Hammingov koder

Kraćica označava da je na simulaciji ocekivano rješenje.

	$d_0 d_1 d_2 d_3$	$c_0 c_1 c_2$	
0ns	0 0 0 0	0 0 0	✓
100ns	1 1 1 1	1 1 1	✓
200ns	0 0 0 1	1 1 1	✓
300ns	1 0 0 0	1 1 0	✓
400ns			

## Generator Sindroma

	$c_0 c_1 c_2 c_0 c_1 c_2$	$s_0 s_1 s_2$	
0ns	0 0 0 0 0 0	0 0 0	✓
100ns	0 0 0 0 0 1	1 0 0	✓
200ns	0 0 0 1 1 1	1 1 1	✓
300ns	1 1 1 0 0 0	1 1 1	✓
400ns			

## Ispriavljanje

	$p_0 p_1 p_2 p_3$	$s_0 s_1 s_2$	$x_0 x_1 x_2 x_3$	
0ns	0 0 0 0	0 0 0	0 0 0 0	✓
100ns	0 0 0 0	0 0 1	0 0 0 0	✓
200ns	0 0 0 0	0 1 1	1 0 0 0	✓
300ns	0 0 0 0	1 1 1	0 0 0 1	✓
400ns				

- slučaj bez greške
- greška na prvom bitu (kontrolni bit)
- prvi poduhvatni ispravljen
- četvrti poduhvatni ispravljen

## Prijamnik

- testovi na zadnjem listu uputa

Postupak za zadnja dva:

d.1)

$$\text{dob. } \begin{array}{r} d_0 \quad d_1 \quad d_2 \quad d_3 \quad x_0 \rightarrow 0 \\ 0 \quad 1 \quad 1 \quad 1 \quad 1 \quad 0 \quad 1 \\ \underline{1 \quad 0 \quad 0} \end{array}$$

$$\begin{array}{r} 0 \quad 1 \quad 1 \\ \oplus 1 \quad 0 \quad 0 \\ \hline 1 \quad 1 \quad 1 \end{array} = 7_{(10)}$$

$$X = 1100$$

d.2)

$$\text{dob. } \begin{array}{r} 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 0 \quad 0 \\ \underline{0 \quad 1 \quad 1} \end{array}$$

$$0 \quad 1 \quad 1 \oplus 1 \quad 1 \quad 1 = 1 \quad 0 \quad 0_{(10)} = 4_{(10)}$$

$$X = 1100$$