Zaštitno kodiranje signala

Laboratorijska vježba 19

SP

FCS – Niz za provjeru polinoma $x^nM(x)/G(x)$ (dioba modulo-2)

Student:

prezime	Ime	mat. broj
	Laboratorij	Datum

Sadržaj:

19. 19. FCS – NIZ ZA PROVJERU ISPRAVNOGA PRIJEMA DUGOGA BINARNOGA POLINOMA $X^N M(X)/G(X)$ (DIOBA MODULO-2)Koristiti:

1. "Upute za rad simulacijskim alatom Logisim 2.7.1." (Moodle)
2. "Simulacijski primjeri" (Moodle)

Priprema za vježbu

Iz skripte "*Zaštitno kodiranje signala*" pročitati poglavlja:

Sazdano od:

Napomena: Cjelovit opis vježbe nalazi se na: MOODLE

1. Vidi: Calculation Of CRC.doc(C++)
2. Vidi: Calculation Of CRC_.doc
3. Vidi: CRC-32 algoritam.doc
4. Vidi: Cyclic Codes.doc
5. Vidi: crc-Ross-OCR.doc
6. Vidi: Error Detection -- Cyclic Redundancy Check.doc
7. Vidi: Fast CRC32.doc
8. Vidi: Opis algoritma CRC-32.doc
9. Vidi: Painless Guide to CRC Error Detection Algorithm 3E.doc
10. Vidi: GSW_Error_Detection.pdf (VIP)
11. Vidi: Convolutional Coding tutorial (o tome kako to učiniti)

Popis *.circ datoteka: Ostatak_010.circ(VIT), Generator_FCS.circ; Koder_i_dekoder.circ; FCS.circ; dekoder.circ; CRC-12.circ; FCS generator (111001).circ;

Sadržaj:

19.1. 19.1 Uvod

Vidjeli smo da ciklički pomak u polinomskoj kodnoj riječi i kodiranje polinomske poruke podrazumijeva diobu jednoga polinoma drugim. Takva operacija lako se ostvaruje *krugom za dijeljenje* (posmični registar s povratnom vezom). Zadana su dva polinoma $M(x)$ i $G(x)$, gdje je

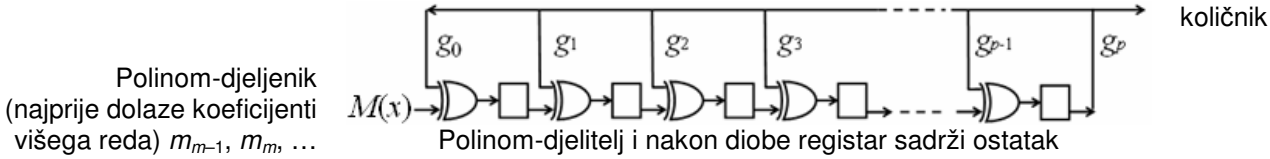
$$M(x) = m_0 + m_1x + m_2x^2 + \ldots + m_mx^m$$

i

$$G(x) = g_0 + g_1x + g_2x^2 + \ldots + g_px^p$$

tako da uz $m \geq p$, krug za dijeljenje na slici 19.1 postupno dijeli polinom $M(x)$ polinomom $G(x)$, čime se određuju izrazi *količnika* i *ostatka*:

$$\frac{M(x)}{G(x)} = Q(x) + \frac{P(x)}{G(x)}$$



Slika 19.1: Opći oblik kruga za diobu polinoma polinomom.

Registar najprije treba dovesti u stanje "sve 0". Prvih p posmika unose članove polinoma *najvišega stupnja* (potencije) od $M(x)$ prema nižim potencijama. Nakon p-toga posmika, na izlazu se dobije *količnik* čija potencija varijable prvoga člana (sasvim desno) ima *najviši* stupanj. Za svaki koeficijent *količnika* q_i, polinom $q_iG(x)$ mora se oduzeti od *djeljenika*. Povratne veze na slici 19.1 omogućuju ovo oduzimanje.

Razlika između p lijevih članova preostalih u *djelitelju* (LFSR) i povratnih članova (veza) $q_iG(x)$ oblikuju se pri svakome posmiku u krugu i pojavljuju se kao sadržaji registra. Pri svakome posmiku registra, člana s varijablom najviše potencije (razlika) posmiče se za jedno mjesto i izlazi iz registra. Sljedeći koeficijent uz varijablu najviše potencije koji je preostao u vektoru **v** na ulazu u posmični registar posmiče se u njega. Nakon m+1 ukupnih posmika u registar, *količnik* se serijski prikazuje na izlazu, a *ostatak* ostaje u registru.

1. Primjer 1 Krug za dijeljenje

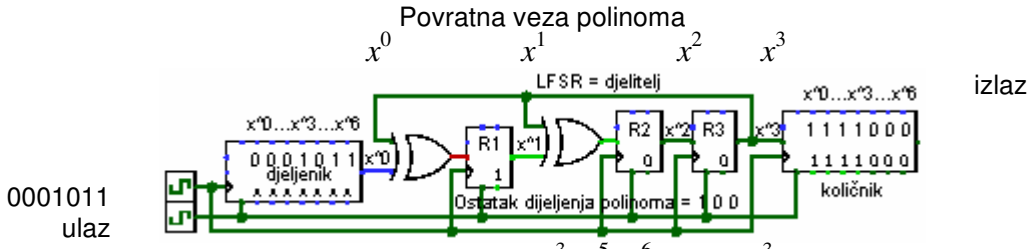
Koristimo krug za dijeljenje polinoma polinomom oblika što ga prikazuje slika 19.1. Dijeli se $M(x) = x^3 + x^5 + x^6$ (**v** = [0001011]) s $G(x) = 1 + x + x^3$ = (**g** = [1101]). Nađimo iznose *količnika* i *ostatka*, a zatim ručnom diobom polinoma usporedimo krug diobe.

Rješenje

Krug za dijeljenje treba izvesti sljedeće operacije:

$$\frac{x^3 + x^5 + x^6}{1 + x + x^3} = q(x) + \frac{r(x)}{1 + x + x^3}$$

Potreban LFSR, prema općemu obliku na slici 19.1, prikazuje slika 19.2.

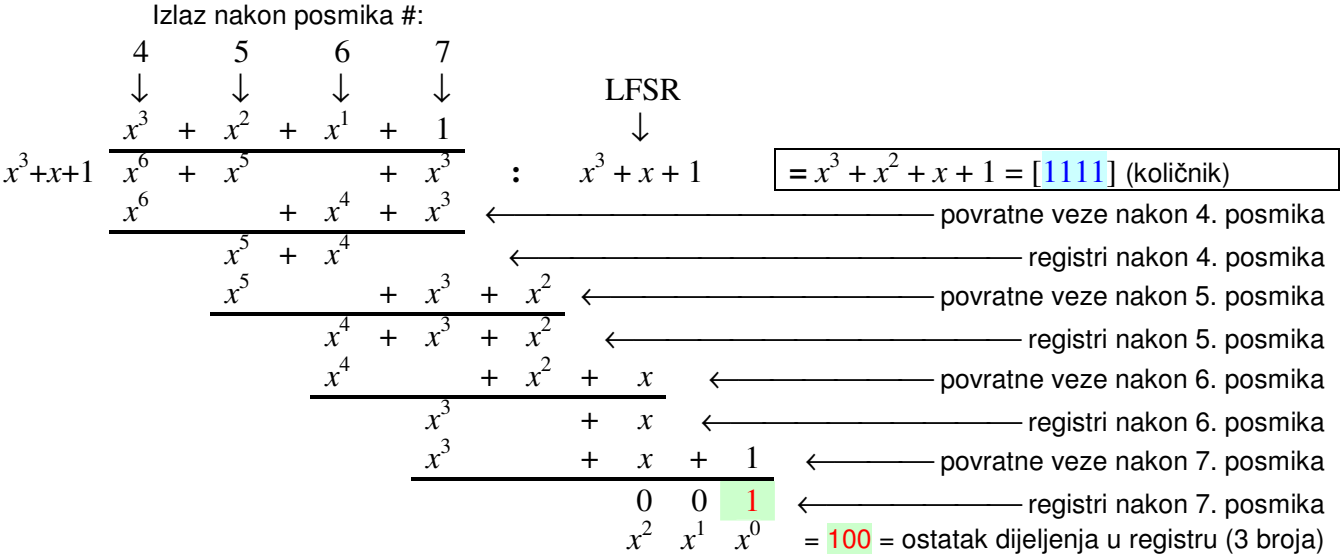


Slika 2 Krug za dijeljenje polinoma $(x^3+x^5+x^6)/(1+x+x^3)$ *iz primjera 1.*

Početni sadržaji registra su "sve 0", stanja registara nakon posmikâ su:

Ulazni spremnik (djeljenik)							Br.	Sadržaji registra							Povratna veza polinoma	Izlaz
x^0	x^1	x^2	x^3	x^4	x^5	x^6		x^0	R1=(1+x^3)	x^1	R2=x+x^3	x^2	R3=R2	x^3		
0	0	0	1	0	1	1	0		0		0		0		–	–
	0	0	0	1	0	1	1		1		0		0		0	–
		0	0	0	1	0	2		1		1		0		0	–
			0	0	0	1	3		0		1		1		1	–
				0	0	0	4		0		1		1		1	**1**
					0	0	5		1		1		1		1	**1**
						0	6		1		0		1		1	**1**
						–	7		**1**		**0**		**0**		1	**1**

Nakon sedmoga posmika, koeficijenti količnika $\{q_i\}$ predstavljeni su serijski na izlazu, a vidljivo je da su [1111], odnosno polinom *količnika* je $Q(x) = 1 + x + x^2 + x^3$. Koeficijenti što su ostali kao sadržaji registra $[r_i]$ su 100, odnosno ostatak diobe polinoma je $R(x) = 1{\cdot}x^0 + 0{\cdot}x^1 + 0{\cdot}x^2$ = [100]. Krug za izračun $M(x)/G(x)$ može se prikazati kao:



19.1.1. 19.1.2. NIZ ZA PROVJERU OKVIRA

Ideja je da se niz bitova FCS (*Frame Check Sequence*) smjesti na kraj dugoga niza podataka (okvira podataka), tako da se cio okvir (*frame*) koji može imati nekoliko tisuća (ili čak nekoliko milijuna) bitova uključujući i ove dodatne bitove, razmatra kao jedan jako dug binaran broj. Taj broj djeljiv je sigurno i pomno odabranim brojem[1] koji se zove *generator uzorka* ili *polinom-generator*. Prijemnik (dekoder) iste strukture, također dijeli primljen niz bitova onako kako dolaze, a ako je konačan ostatak nula, pretpostavlja se velikom vjerojatnošću da se okvir ispravno primio.

Napomena: Ovdje se koristi aritmetika modulo-2. To samo znači da se zanemaruju prijenos ili pozajmljivanje bitova. Dakle sva zbrajanja i oduzimanja, samo su operacija isključivo ILI, koju je sklopovski veoma lako napraviti (EXOR vrata i LFSR). Imajte na umu da ako se nešto dijeli binarnim brojem dugim (n+1) bitova, onda se dobije *ostatak* od n bitova (onoliko koliko LFSR ima stanja).

19.2. 19.2. Ciklički kodovi

Ciklički kodovi su *posebna* vrsta *linearnih blok kodova* koji su popularni, jer su vrlo učinkoviti za otkrivanje i ispravak pogrešaka, a njihove kodere i dekodere lako je primijeniti u sklopovima.

Definicija:

Ciklički kod je linearan (n, k) blok-kod sa svojstvom da svaki *ciklički pomak kodne riječi rezultira drugom kodnom riječi.*

Ciklički pomaci bilo kojih binarnih znamenaka konačne riječ $[b_{n-1}, b_{n-2}, \ldots, b_0]$ generiraju ispis niza i pomicanje bitova (lijevo ili desno) za željeni broj pomaka, na takav način da bilo koji bit što napusti

[1] Posebno opisati kako se odabiru polinom generatori!

riječ na jednome kraju, ponovno se doda na početak te riječi na drugome kraju. Drugim riječima, pomičemo bitove prenoseći ih s jednoga kraja na drugi.

Na primjer, generator-matrica $\mathbf{G} = \left[\begin{array}{c|cc} 1 & 1 & 0 \\ 1 & 0 & 1 \end{array}\right]$ za (3, 2) linearan blok-kod, proizvodi kodne riječi: [000], [110], [101] odnosno [011] koje odgovaraju podatkovnim riječima [00], [10], [01] odnosno [11]. Odaberite bilo koju kodnu riječ, npr. [110]. Njezini ciklički pomaci [011] i [101] su ispravne kodne riječi.

19.3. 19.3. Kodovi provjere cikličke zalihosti

Vrlo popularan kod za *otkrivanje pogrešaka* što se primjenjuje u mnogim shemama za prijenos podataka, je kod nazvan "*ciklička provjera zalihosti*" CRC (*cyclic redundancy check*). Imajte na umu da je to u osnovi vrsta linearnoga blok-koda.[2]

19.3.1. 19.3.1. PRIKAZ BINARNIH RIJEČI POLINOMOM

Da bi razumjeli kako radi CRC koder, najprije definirajmo pojam *polinoma* pridruženoga binarnome nizu. Obzirom na neku opću varijablu x, polinom je niz potencija od x u kombinaciji s binarnim koeficijentima (brojevi 0 i 1).

Za niz bitova $[b_{k-1}, b_{k-2}, \ldots, b_1, b_0]$ pripadajući polinom je

$$b_{k-1}x^{k-1} + b_{k-2}x^{k-2} + \ldots + b_1x^1 + b_0x^0$$

Imajte na umu da ovaj polinom *reda* k–1 za niz bitova (riječ) ima *duljinu* od k *bitova*.

Na primjer, niz poruke od $k = 10$ bitova [1010100101], prikazuje se kao polinom

$$M(x) = x^9 + x^7 + x^5 + x^2 + 1 = [\overset{9\,8\,7\,6\,5\,4\,3\,2\,1\,0}{1010100101}].$$

Neka je ukupna duljina kodne riječi $n + k = N$ i neka je sukladna poruci $M(x)$. Pretpostavimo da se k bitova podataka poruke kodira u ukupno N bitova računajući *pridruživanje* niza od $n = N - k$ bitova $[r_{n-1}, r_{n-2}, \ldots, r_1, r_0]$ kraju poruke (desno), slično tvorbi sustavnih blok-kodova (dodavanjem paritetnih bitova desno od informacijskih). Neka je $R(x)$ polinom što predstavlja ove dodane bitove:

$$M(x) = [b_{k-1}, b_{k-2}, \ldots, b_1, b_0, r_{n-1}, r_{n-2}, \ldots, r_1, r_0],$$

kojemu odgovara polinomski prikaz $T(x) = x^nM(x) + R(x)$ (pomak bitova u $M(x)$ za n mjesta u lijevo te dodavanje isto toliko "0" desno. Nakon toga, pomaknut vektor i bitovi *ostatka* $R(x)$ zbrajaju se modulo-2 (≅ EXOR zbroju). Ako su:

- ukupna dužina "kodirane" riječi ($x^nM(x)$) $N = 13$ bitova,
- dužina informacijskoga vektora $k = 10$ bitova i
- dužina niza koji će se pridružiti informacijskome vektoru nakon diobe je

$$n = N - k = 13 - 10 = 3 \text{ [bita]}$$

$M(x) = 1010100101$	$x^nM(x) =$	+	1010100101**000**	+
$x^nM(x) = 1010100101$**000**	$R(x) =$ **?**		**???**	

potrebno je izračunati $R(x)$. Njegovu dužinu (*ostatak*: $n = 3$) određuje broja stanja LFSR, a povratne veze LFSR definiraju polinom-*djelitelj* (u ovome primjeru on je $x^2 + x + 1$, dakle imamo:

Izvorni podaci poruke = $M(x)$ (dužine k bitova)	FCS = $R(x)$ (dužine n bitova)
Okvir spreman za slanje: $T(x) = x^nM(x) + R(x)$ (ukupna dužina = $k + n$ bitova)	

Slika 3: Pridruživanje FCS od n bitova podatkovnome okviru

[2] CRC (*cyclic redundancy check*) … kružna (ciklička) provjera zalihosti ili *provjera cikličke zalihosti*. (Rječnik komunikacijske tehnologije); CRC (*cyclic redundancy code*) … *ciklički zalihostan kod*, itd. (Digitalni prijenos informacija NUANCE.doc), (Digital Communication 3.Ed)

Slijedi, da izvorni bitovi poruke sada zauzimaju mjesta najznačajnijih bitova u kodnoj riječi (mjesta sasvim lijevo). Svaki bit poruke pomiče se *lijevo* za *n* bitova da bi se napravilo mjesta za pridružiti *n* FCS bitova.

Na primjer, ako gornjemu nizu poruke od 10_{10} bitova pridružimo FCS niz $R(x)$ = [111] od k = 3 bita, nastat će kodni niz od 13_{10} bitova čije polinomski prikaz je:

$$x^n M(x) + R(x) = x^3(x^9 + x^7 + x^5 + x^2 + 1) + (x^2 + x + 1) = x^{12} + x^{10} + x^8 + x^5 + x^3 + x^2 + x + 1.$$

19.3.2. 19.3.2. POSTUPAK KODIRANJA ZA IZRAČUN OSTATKA PRI DIOBI POLINOMA

Napomena: Djelitelj ≡ struktura LFSR

Da bi definirao cjelovit zaštićen kod, iz (u lijevo pomaknutih) informacijskih bitova, moramo izračunati polinom $R(x)$ (*ostatak*). To su bitove koji će se pridružiti informacijskome polinomu s desne strane, a usko su funkcijski povezani skupom bitova poruke $M(x)$ (*djeljenik*). Oni predstavljaju ostatak diobe informacijskoga polinoma i strukture LFSR (*djelitelj*). Pridruženi bitovi $R(x)$, zovu se *niz za provjeru* FCS (*Frame Check Sequence*) skupa/okvira (*frame*) od k informacijskih bitova podataka.

- CRC kod $R(x)$ (n pridruženih bitova) odnosno niz n bitova za provjeru okvira FCS (*Frame Check Sequence*), definira se pomoću posebnoga polinom-generatora $G(x)$ stupnja n s "ne-nultim" koeficijentima najvišega i najnižega reda.

Na primjer: Za dodatnih n = 12 bitova, $G(x)$ (≅ LFSR polinom-generator) jednak je npr. $x^{12} + x^{11} + x^3 + x^2 + x + 1$ = [1100000001111]; ili za dodatnih n = 6 bitova, $G(x)$ (≅ LFSR polinom-generator) jednak je npr. $x^5 + x^4 + x^2 + 1$ = [110101]. Prva i zadnja znamenka vektora, MORAJU biti 1, jer su to ulaz odnosno izlaz LFSR. Vidi primjer u nastavku.

- Za linearne blok kodove, polinom-generator igra ulogu generator-matrice.
- Za dobiti FCS bitove, ili ekvivalentno $R(x)$, podijelit ćemo $x^n M(x)$ s $G(x)$ (modulo-2 dioba), a ostatak diobe je $R(x)$.

Imajte na umu da pri diobi polinoma modulo-2 binarnim koeficijentima:

(i) moramo *oduzimati* koeficijente istih potencija (kao i u bilo kojoj dugoj diobi), ako se dijele polinomi;

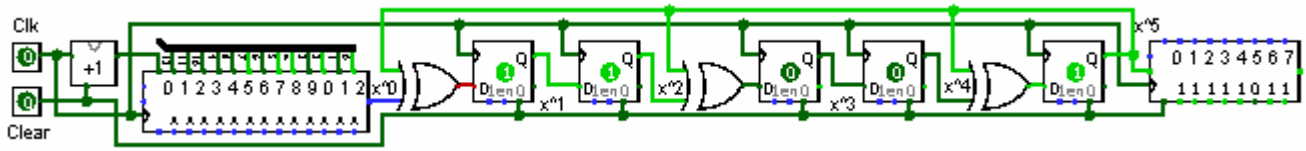
(ii) binarno *oduzimanje* koeficijenata modulo-2 jednako je binarnome *zbrajanju* koeficijenata modulo-2.

Prisjetimo se da je u aritmetici modulu-2 imamo

$$0 - 0 = 0 = 0 + 0$$
$$1 - 0 = 1 = 1 + 0$$
$$0 - 1 = 1 = 0 + 1$$
$$1 - 1 = 0 = 1 + 1$$

2. Primjer: Dioba proširenoga informacijskoga polinoma

Dioba proširenoga informacijskoga polinoma $x^n M(x)$=[1010110100000] polinomom $G(x)$=[110101], (a to je struktura LFSR). Napraviti sklop: LFSR: [1010110100000] : [110101] i potvrditi ostatak $R(x)$ = [10011].



Slika 19.2: Sklop za dijeljenje [1010110100000] : [110101] = [11011111]

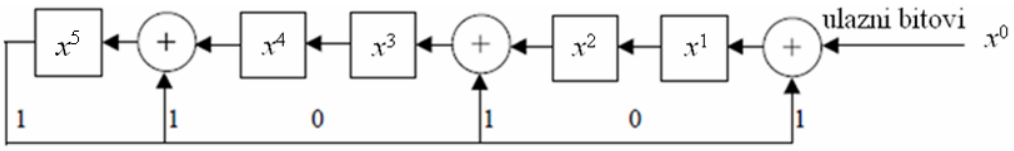
													x^7	6	5	4	3	2	1	x^0	
													1	1	0	1	1	1	1	1	← količnik
$G(x)$ = djelitelj (= LFSR)							$x^n M(x)$ = djeljenik														
potencije	5	4	3	2	1	0		12	11	10	0	8	7	6	5	4	3	2	1	0	
LFSR =	1	1	0	1	0	1		1	0	1	0	1	1	0	1	0	0	0	0	0	← djeljenik
							+	1	1	0	1	0	1	↓							← djelitelj
									1	1	1	1	0	0							← količnik
							+	1	1	0	1	0	1		↓						← djelitelj
	Ako je potencija djeljenika								0	1	0	0	1	1							←količnik
	manja od potencije djelitelja →						+	0	0	0	0	0	0		↓						←djelitelj
										1	0	0	1	1	0						← količnik
									+	1	1	0	1	0	1	↓					← djelitelj
											1	0	0	1	1	0					← količnik
								+	1	1	0	1	0	1	↓						← djelitelj
											1	0	0	1	1	0					← količnik
									+	1	1	0	1	0	1	↓					← djelitelj
												1	0	0	1	1	0				← količnik
										+	1	1	0	1	0	1	↓				← djelitelj
													1	0	0	1	1	0			← količnik
											+	1	1	0	1	0	1	↓			← djelitelj
														1	0	0	1	1	0		← količnik
													+	1	1	0	1	0	1	↓	← djelitelj
															1	0	0	1	1	0	← količnik
														+	1	1	0	1	0	1	← djelitelj
								koeficijenti polinoma →								1	0	0	1	1	← ostatak
								polinom →								x^4	$+x^3$	$+x^2$	$+x^1$	$+x^0$	

Slika 19.3: Primjer izračuna FCS $\Rightarrow R(x)=1+x+x^4=\overset{01234}{[11001]}=x^4+x+1=\overset{43210}{[10011]}$

19.3.2.1. 19.3.2.1. Primjer:

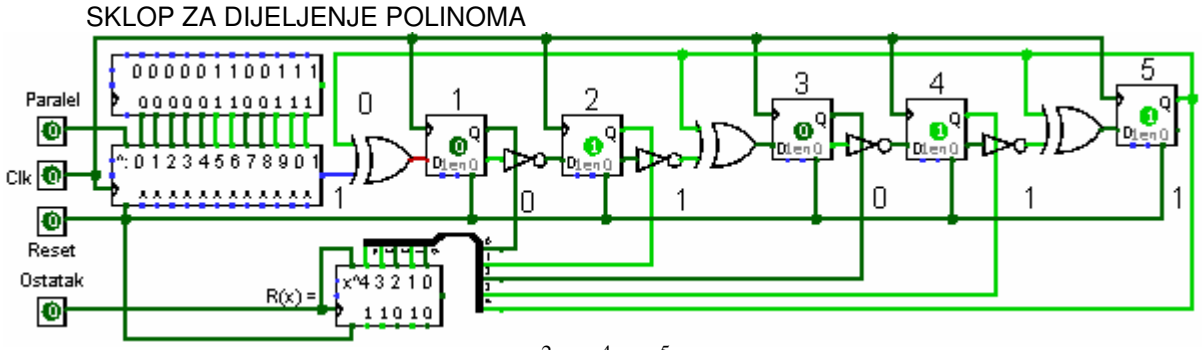
x^5	4	3	2	1	x^0		x^{12}	11	10	9	8	7	6	5	4	3	2	1	x^0		x^6	5	4	3	2	1	x^0
1	1	0	1	0	1		1	0	1	0	1	1	0	1	0	0	0	0	0	=	1	0	1	1	1	1	1
						+	1	1	0	1	0	1																
								1	1	1	1	0	0							1								
						+	1	1	0	1	0	1																
								0	1	0	0	1	1							0								
						+	0	0	0	0	0	0																
									1	0	0	1	1	0						1								
								+	1	1	0	1	0	1														
										1	0	0	1	1	0					1								
									+	1	1	0	1	0	1													
											1	0	0	1	1	0				1								
										+	1	1	0	1	0	1												
												1	0	0	1	1	0			1								
											+	1	1	0	1	0	1											
													1	0	0	1	1	0		1								
												+	1	1	0	1	0	1										
														1	0	0	1	1	0	1								
													+	1	1	0	1	0	1									
															1	0	0	1	1									
															4	3	2	1	x^0									

3. *Primjer: LFSR za stvaranje FCS*



$$G(x)=x^5+x^4+x^2+1=1+x^2+x^4+x^5=[101011]$$

Slika 19.4: LFSR za stvaranje FCS (zrcalna slika slike 4 s lijevim posmikom)



$$G(x) = 1 + x^2 + x^4 + x^5 = [101011]$$

Slika 19.5: LFSR za stvaranje FCS (zrcalna slika slike 19.3 s desnim posmikom). Ostatak je $R(x) = {}^{4\,3\,2\,1\,0}11010$

Napomena: Polinom $G(x)$ ima 1 član više u odnosu na broj stanja pripadnoga LFSR

4. *Primjer: Poruka (djelitelj) [11100110] (8 bitova):*

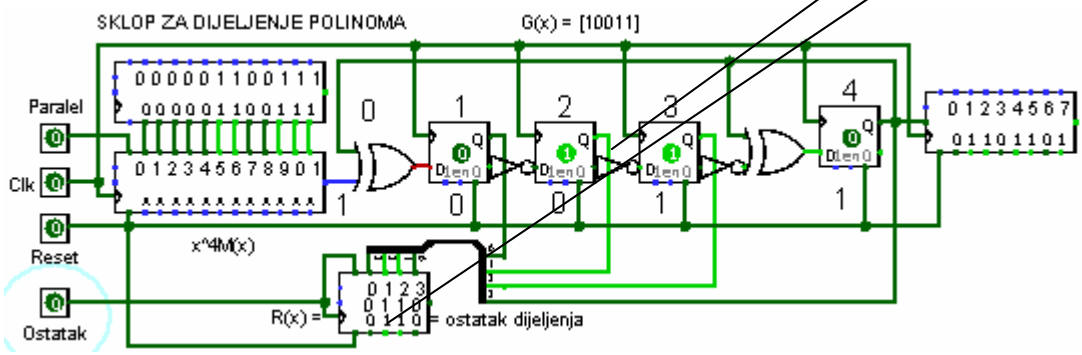
$$M(x) = x^7 + x^6 + x^5 + x^2 + x = [11100110]$$

Neka je *niz za provjeru okvira* FCS (*Frame Check Sequence*) dugačak $n = N–k = 4$ i neka je zadan polinom-*djelitelj* $G(x) = x^4 + x^3 + 1$ (ovo je struktura LFSR). (Napomena: Dužina FCS jednaka je broju stanja LFSR.)

Podijelimo: $x^n M(x) = x^4 M(x) = x^4 \cdot [11100110] = [\,111001100000\,]$ s $G(x) = [11001]$[3]:

$$\frac{x^{11}+x^{10}+x^9+x^6+x^5}{x^4+x^3+1} = x^7 + \frac{x^9+x^7+x^6+x^5}{x^4+x^3+1} = x^7 + x^5 + \frac{x^8+x^7+x^6}{x^4+x^3+1} = x^7 + x^5 + x^4 + \frac{x^6+x^4}{x^4+x^3+1} =$$

$$= x^7 + x^5 + x^4 + x^2 + \frac{x^5+x^4+x^2}{x^4+x^3+1} = x^7 + x^5 + x^4 + x^2 + x = \frac{x^2+x}{x^4+x^3+1}$$

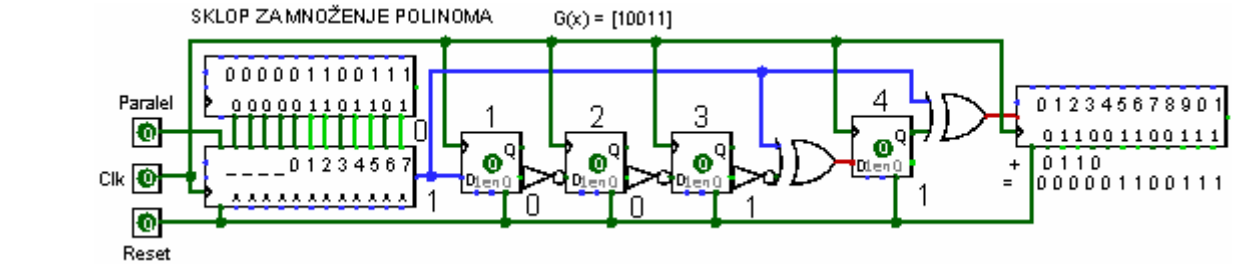


Slika 19.6: Dijeljenje: [111001100000] : [11001] = [10110110] + [0110]

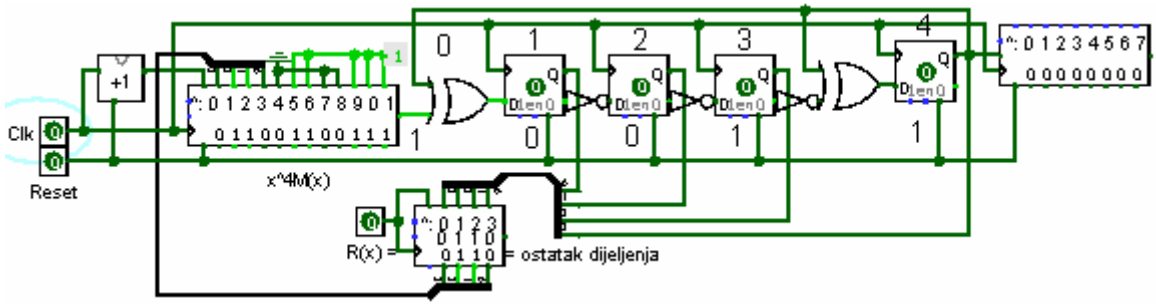
Ostatak je $R(x) = x^2 + x$, dakle FCS jednak je $[\,{}^{3\,2\,1\,0}0110\,]$, jer znamo da je $n = 4$ (broj stanja LFSR). Imajte na umu da je *ostatak* uvijek polinom maksimalnoga reda $n–1$ (3 u ovom primjeru, dakle manji je od najviše potencije djelitelja tj. jednak je dužini LFSR). Također imajte na umu da je $x^n M(x)$ polinom koji odgovara nizu bitova poruke gdje se kraju niza pridružuje *n* nula) (u ovome primjeru, [11100110**0000**] (dakle, 4 nule, tj. upravo onoliko "0" koliko LFSR-djelitelj ima stanja). Dioba polinoma može se smatrati *dioba ovoga binarnoga niza* (bitovi poruke + *n* dodanih nula) nizom koji odgovara generator-polinomu, a ovdje je to $[\,{}^{0\,1\,2\,3\,4}10011\,]$ (struktura LFSR).

[3] Množenjem s x^4 množe se svi članovi polinoma čiji su koeficijenti jednaki 1, a to znači da se polinomu pridružuje desno onoliko nula kolika je potencija od x (u ovome primjeru ona je 4).

Provjera prethodnoga dijeljenja:



Slika 19.7: Provjera diobe na predajnoj strani: [01101101] · [10011] = [011001100111]



Slika 19.8: Provjera diobom na prijemnoj strani: $R(x) + M(x)$ = [011001100111] ÷ [10011] = [000001100111] = FCS/LFSR = cjelobrojan rezultat, bez ostatka tj. $R(x) = 0$

19.3.3. 19.3.3. DEKODIRANJE KAO DIOBA POLINOMA

Nakon prijema, prenijete kodne riječi koje nemaju *pogreške* i ako je polinom primljene kodne riječi $T(x) = x^n M(x) + R(x)$, djeljiv s $G(x)$, dobije se *ostatak* $R(x) + R(x)$ = redak "sve 0" (imajte na umu da je to zbroj modulo-2), jer $x^n M(x)/G(x)$ ima *ostatak* $R(x)$ (definicija konstrukcije kodne riječi), a $R(x)/G(x)$ ima ostatak $R(x)$, jer je $R(x)$ polinom nižega reda od $G(x)$.

Ako se dobije kodna riječ, a *ostatak nije* "sve 0", polinom se dijeli generator-polinom $G(x)$ pa imamo naznaku da se pri prijenosu dogodila *pogreška* te smo primili riječ koja nije ispravna kodna riječ (ne pripada skupu važećih kodnih riječi).

Ako se pojavila pogreška, neka je *uzorak pogreške* polinom $E(x)$[4] (niz duljine N ima 1 na mjestu na kojemu se dogodila pogreška bita - komplement). Onda je primljen polinom $T(x) + E(x)$. Ostatak što ga dobijemo dijeleći primljen polinom $T(x) + E(x)$ s $G(x)$, jednostavno je ostatak dobiven diobom $E(x)$ s $G(x)$. Ako se diobom $E(x)/G(x)$ dobije ostatak, može se pokazati da pomoću FCS od n bitova, možemo oblikovati *polinom-generator*, tako da je moguće otkriti:

(a) Sve jednostruke pogreške.

$G(x)$ sadrži najmanje dva izraza, x^n i 1. Za jednostruku pogrešku, $E(x)$ je u obliku x^i pa se $G(x)$ ne može se podijeliti s $E(x)$ bez ostatka.

(b) Sve dvostruke pogreške

Za dvije pogreške imamo $E(x) = x^i + x^j$ za $N–1 \geq i > j \geq 0$. Stoga je $E(x) = x^j(x^{i–j} + 1)$. Dakle $G(x)$ ne može podijeliti $x^p + 1$, za bilo p manji ili jednak $N–1$. Dobar polinom $G(x)$ nižega stupnja i svojstva da $G(x)$ ne dijeli x^p+1 sve do vrlo velikoga p.

(c) *Bilo koji neparan broj pogrešaka* [sve dok $G(x)$ ima faktor $(1+x)$, kao što je slučaj za popularne CRC polinome koji se koriste u praksi].

Na primjer, neka je $E(x) = x^5+x^2+1$. Dioba ove vrste polinoma neparnoga broja koeficijenata bilo kojim $G(x)$ koji ima $(x+1)$ kao faktor, nije moguća. Pretpostavimo da se može naći $(x+1)P(x)$ koji se može podijeliti takvim $E(x)$. To bi značilo da je $E(x) = (x+1)P(x)Q(x)$, gdje je $Q(x)$ rezultat diobe (bez ostatka). Sada stavimo x=1 u ovaj rezultat. Lijeva strana ima točno neparan broj

[4] Kako se dobije *uzorak pogreške* polinoma $E(x)$?

jedinica, tako da je zbroj modulo-2 jednak 1. Na desnoj strani, (x+1) uvijek će dati 0 što je proturječno.

(d) Bilo koja *praskovita pogreška* duljine ≤ n i većina većih praskova pogrešaka.

Za n bitova u FCS, $G(x)$ je polinom stupnja n. Za praskovite pogreške trajanja $P \leq n$, $E(x)$ je u obliku $x^m + \ldots + x^{m-p+1} = x^{m-p+1}(x^{p-1} + \ldots + 1)$. Ali polinom $G(x)$ stupnja n, ne može se podijeliti bez ostatka.

Ova svojstva čine CRC kodove vrlo korisnima za otkrivanje pogrešaka u prijenosu dugih okvira binarnih znamenaka. Najpoznatiji CRC kodovi za komunikacijske aplikacije koriste CRC-16, CRC-32 i CRC-CCITT generator-polinome. Koristan posmični registar koji primjenjuje diobu polinoma, može se jednostavno primijeniti.

19.4. 19.4. Opis algoritma CRC-32

19.4.1. 19.4.1. UVOD

CRC-32 je skraćenica od *cyclic redundancy code*, dok broj 32 označava da je izlaz iz *hash* algoritma veličine 32 bita. Algoritam CRC-32 rijetko se koristi u računalskoj sigurnosti zbog male veličine izračunatoga sažetka. Međutim CRC-32 naišao je na veliku primjenu u otkrivanju pogrešaka pri prijenosu podataka gdje pošiljatelj računa sažetak poruke i pridružuje ga poruci te ih zajedno šalje primatelju. Primatelj također računa sažetak i uspoređuje ga primljenim sažetkom te se na temelju podudarnosti (ili nepodudarnosti) sažetaka utvrđuje ispravnost ili neispravnost dobivene poruke.

Prednost ovoga algoritma nad klasičnim načinima otkrivanja pogrešaka (paritet, Hammingov kod, …) je u tome što ti algoritmi u nekim slučajevima ne mogu otkriti pogrešku (npr. ako pri provjeri pariteta, dva bita promijene vrijednost), dok CRC-32 otkriva pogrešku, jer i za male promjene poruka, sažeci se znatno razlikuju.

CRC-32 također se koristi kao alat za arhiviranje podataka, te kod medija za pohranu podataka (npr. CD, DVD). Tu se također CRC-32 koristi za otkrivanje pogrešaka među podacima. CRC-32 nije *hash* algoritam, koji ima neku veliku važnost u računalnoj sigurnosti. On je važan u primjenama i ondje gdje je potreban pouzdan i brz alat za otkrivanje pogreške.

19.4.2. 19.4.2. OSNOVNA IDEJA ALGORITMA CRC-32

Osnovna ideja koja se krije iza CRC algoritma je pretvoriti poruku u vrlo velik binaran broj što ga se dijeli definiranim brojem koji se naziva *ključ* (*key*). Dijeljenjem tih dvaju brojeva, dobivaju se *kvocijent* i *ostatak* $R(x)$. Ostatak dijeljenja predstavlja sažetak (*hash*) poruke i on nam je važan.

Za CRC algoritam, binaran broj koji se dobije diobom poruke i definiranoga ključa, ne predstavlja se kao cio broj već kao polinom binarnih koeficijenata. Prije opisa rada algoritma opisat će se binarni polinomi tj. polinomi s binarnim koeficijentima i aritmetika nad binarnim poljem (aritmetika mod-2).

19.4.3. 19.4.3. ARITMETIKA NAD BINARNIM POLJEM

Binarno polje BF (*binary field*), ima dva elementa i možemo ga prikazati kao:

$$BF = \{0,1\}$$

Nad tim poljem provodimo aritmetiku mod-2, što znači da radimo operacije kao i inače samo što kao rezultat uzimamo cjelobrojan ostatak dijeljenja brojem 2 koji može biti 0 ili 1, a pošto su to elementi binarnoga polja, rezultat operacija pripada binarnome polju.

Tablice operacija aritmetike mod-2:

Zbrajanje:	Oduzimanje:	Množenje:	Dijeljenje:
0 + 0 = 0	0 – 0 = 0	0 * 0 = 0	0/1 = 0
0 + 1 = 1 + 0 = 1	1 – 0 = 1, 0 – 1 = 1	0 * 1 = 0	1/1 = 0
1 + 1 = 0 (2 mod-2 = 0)	1 – 1 = 0	1 * 1 = 1	

Iz ovoga se vidi da su operacije *zbrajanja* i *oduzimanja* jednake te se mogu zamijeniti binarnom operacijom $\oplus$ (XOR - izričito ILI), a operacija množenja može se zamijeniti binarnom operacijom $\otimes$ (& - I, AND).

19.4.4. 19.4.4. POLINOMI S BINARNIM KOEFICIJENTIMA ILI BINARNI POLINOMI

Binarni polinomi imaju koeficijente iz binarnoga polja BF. Primjer binarnoga polinoma:

$$1 \cdot x^4 + 0 \cdot x^3 + 0 \cdot x^2 + 1 \cdot x^1 + 1 \cdot x^0$$

Ovaj polinom može se prikazati kao binaran broj:

$$10011$$

Da je ovaj prikaz dobar može nam poslužiti primjer zbrajanja i množenja dvaju binarnih polinoma:

$$(1x^2 + 1x^1 + 0x^0) + (0x^2 + 1x^1 + 1x^0) = 1x^2 + 0x^1 + 1x^0$$

$$(1x^2 + 1x^1 + 0x^0) * (0x^2 + 1x^1 + 1x^0) = 0x^2 + 1x^1 + 0x^0$$

Prikažimo sada te polinome kao binarne brojeve te napravimo nad njima operaciju XOR ($\oplus$) kao zamjenu za zbrajanje i operaciju & ($\otimes$, I, AND) kao zamjenu za množenje:

$$110 \oplus 011 = 101$$

$$110 \otimes 011 = 010$$

Vidimo da su koeficijenti u oba načina prikaza jednaki te da se binaran polinom može prikazati kao binaran broj. Polinomi se inače prikazuju bez koeficijenata 1 i 0 tako da u zapisu polinoma imamo samo one potencije varijable x koje imaju koeficijent 1, dok se članovi s koeficijentom 0 ne prikazuju. Takav zapis prikazao bi se na sljedeći način:

$$x^8 + x^5 + x^4 + x^2 + x^0$$

Taj zapis jednak je:

$$\begin{matrix} 8\,7\,6\,5\,4\,3\,2\,1\,0 \\ 1\,0\,0\,1\,1\,0\,1\,0\,1 \end{matrix}$$

Iz ovih razmatranja slijedi da se svaki binaran podatak može prikazati kao polinom s binarnim koeficijentima odnosno binaran polinom.

19.5. 19.5. Algoritam diobe

Algoritam CRC-32 uzima podatak iz kojega računa sažetak te ga pretvara u binaran polinom $M(x)$ te dijeli taj polinom definiranim binarnim polinom zvanim ključ $G(x)$. Ostatak dijeljenja tih polinoma $M(x)/G(x)$ daje kvocijent $Q(x)$ i ostatak $R(x)$. Ostatak toga dijeljenja sažetak je $H(x)$ dobiven algoritmom CRC-32. Može se pisati:

$$M(x)/G(x) = Q(x) + R(x)$$

$$R(x) = H(x)$$

Pošto CRC-32 ima 32-bitni sažetak $H(x)$ iz toga slijedi da ostatak $R(x)$ mora biti 32-bitni broj. Ako želimo imati ostatak stupnja d tada djelitelj mora biti stupnja d+1. Sažetak za CRC-32 ima stupanj 31 i iz toga slijedi da ključ $G(x)$ mora biti stupnja 32. Ključ koji se najčešće koristi za CRC-32 u heksadekadskome zapisu možemo prikazati kao:

$$G(x) = 0\text{x}4\text{C}11\text{DB}7 \cong \begin{matrix} \overset{32}{1} & \overset{31.....28}{0\,0\,0\,0} & \overset{27.....24}{0\,1\,0\,0} & \overset{23.....20}{1\,1\,0\,0} & \overset{19.....16}{0\,0\,0\,1} & \overset{15.....12}{0\,0\,0\,1} & \overset{11.....8}{1\,1\,0\,1} & \overset{7.....4}{1\,0\,1\,1} & \overset{3.....0}{0\,1\,1\,1} \\ 1 & 0 & 4 & C & 1 & 1 & D & B & 7 \end{matrix}$$

Kao binaran polinom, ključ ima sljedeći oblik:

$$G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x^1 + x^0 \text{ (= LFSR struktura)}$$

Dakle CRC-32 zahtijeva dijeljenje binarnoga polinoma poruke $M(x)$ i ključa $G(x)$. Ova dva polinoma ne možemo izravno podijeliti, jer u registre 32-bitnih procesora (još je velik broj računala ovakve konfiguracije) stane najviše 32 bita dok poruka može biti duljine nekoliko milijuna bitova. Čak ni ključ ne stane u registar, jer je i on duljine 33 bita. Vidi se da trebamo nekakav algoritam koji će podijeliti

ova dva broja. Pri diobi, nije nam važan *kvocijent* diobe već samo *ostatak* koji čini sažetak (*hash*) pa se kvocijent neće ni uzimati u obzir.

19.5.1. 19.5.1. OPIS RADA ALGORITMA DIOBE

Koeficijent binarnoga polinoma $M(x)$ (poruka) sasvim lijevo, koeficijent je s najvišom potencijom. ~~Ako je taj koeficijent nula idemo na sljedeći koeficijent tj. pomaknemo se u desno.~~ Ako je taj koeficijent 1, uzmemo sljedećih n bitova gdje je n duljina djelitelja (ili ključa) te ih oduzmemo prema aritmetici modulo-2. Oduzimanje je XOR operacija zbrajanja u aritmetici modulo-2. Sada se opet pomaknemo za jedno mjesto u desno te provodimo isti postupak do kraja poruke.

Za prikaz primjera rada algoritma, podijelit ćemo dva binarna polinoma:

$$M(x) = x^5 + x^2 + x^1$$

$$G(x) = x^2 + x^0$$

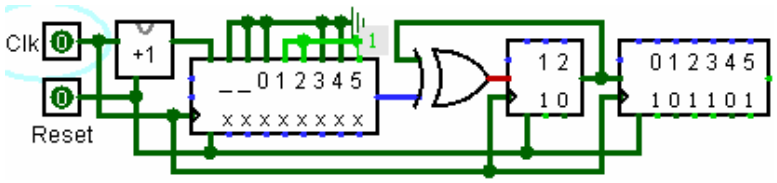
Ta dva polinoma su u binarnom zapisu:

$$M = [100110]^{5\,4\,3\,2\,1\,0}$$

$$G = [101]^{2\,1\,0}$$

Priskrba ostatka diobe ova dva binarna polinoma je kako slijedi:

x^7	x^6	x^5	x^4	x^3	x^2	x^1	x^0		x^2	x^1	x^0		x^5	x^4	x^3	x^2	x^1	x^0
1	0	0	1	1	0	0	0	÷	1	0	1	=	1	0	1	1	0	1
1	0	1																
0	0	1	1	1														
		1	0	1														
		0	1	0	0													
			1	0	1													
			0	0	1	0	0											
					1	0	1											
					0	0	1											



Slika 19.9: Sklop za dijeljenje $M(x)/G(x)$

U ovome slučaju, ostatak je $R(x) = 01^{1\,0}$. Iz ovoga primjera vidimo da se bitovi polinoma ključa sasvim lijevo i sasvim desno ne moraju zapisivati, jer su oni uvijek jednaki 1 (ulaz i izlaz u LFSR). Isto vrijedi za polinom ključa polinoma $G(x)$ algoritma CRC-32 Tako se ni krajnji bitovi polinoma ključa $G(x)$ sasvim lijevo i sasvim desno algoritma CRC-32 ne zapisuju te pamtimo samo 31 bit, a ne 33 koja niti ne stanu u CPU registar.

Sada ćemo opet prikazati prethodnu diobu, ali tako da će se podebljati bitovi polinoma $M(x)$ koji su u registru tijekom dobivanja ostatka i podcrtat će se oni bitovi polinoma $G(x)$ koji su također u registru.

```
00010011000/101
001|||||||
 010|||||||
  100|||||
  101|||||
  ---|||||
  001|||||
   011||||
    111|||
    101|||
    ---|||
    010|||
```

```
100||
101||
---||
001||
 010|
 100
 101
 ---
 001
```

Ovaj način dobivanja sažetka poruke dosta je spor, jer se obrađuje bit po bit. Iz toga razloga napravljena je optimizirana verzija CRC-32 algoritma. Ta verzija ne radi bitovima već bajtovima. Za ovu verziju prethodno moramo imati izračunatu tablicu za sve kombinacije broja od 8 bitova tj. tablicu s 256 elemenata. Nakon što imamo tu tablicu možemo izračunati sažetak.

Sažetak računamo tako da uzmemo posljednji bajt (najmanje značajan bajt) iz registra te napravimo XOR operaciju između njega i sljedećega ulaznoga bajta poruke. Tu vrijednost koristimo kao indeks elementa u tablici iz koje očitamo spremljenu vrijednost operacije. Napravimo XOR operaciju između te vrijednosti i prijašnje vrijednosti registra pomaknute za jedan bajt (8 bitova) u desno.

Algoritam za izračun CRC-32 pomoću tablice je:

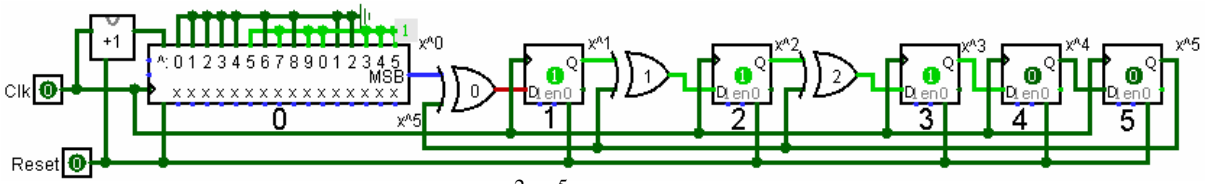
```
crc32 (ulaz[],duljina_poruke){
hash = 0
for(i=0;i<duljina_poruke;i++){
izlazni_byte=hash&0xff
hash=crc32_tablica[ulaz[i]XOR izlazni_byte]XOR(hash>>8)
}
return hash;
}
```

19.6. 19.6. Primjena diobe za izračun CRC

CRC kod generira se postupkom diobe polinoma. To se može provesti pomoću digitalnoga logičkoga sklopa koji se sastoji od isključivo ILI (XOR) vrata i posmičnoga registra. Vrata XOR predstavljaju zbrajalo modulo-2 s dva ulaza od jednoga bita. Primjenjuje se matematika 0 + 0 = 0, 1 + 0 = 1, 0 + 1 = 1 i 1 + 1 = 0. U praksi, svaki ulaz i izlaz ima jednu od dvije moguće naponske razine u strujnome krugu. Posmični registar je niz uređaja za pohranu jednoga bita (D-bistabil) i svaki ima priključke ulaza i izlaza. Posebnim taktom, bit pohranjen u svakome registru, pomakne se i zamjenjuje novim bitom s ulaza.

19.6.1. 19.6.1. FCS GENERATOR

Razmotrimo dijeljenje binarne poruke $[b_{k-1}, b_{k-2}, \ldots, b_0] = M(x) = [11100110101]$ duljine $k = 11$ uz $n = 5$ uz pridruženih 5 nula [00000] (koliko LFSR ima stanja), djeliteljem $(x^5 + x^2 + x + \mathbf{1}) = [\overset{MSB}{1}00111\overset{LSB}{1}]$ = $[\overset{MSB}{1}00111\overset{LSB}{]}$ kako bi pronašli *ostatak* (FCS). Ovdje su položaji najvažnijega bita **sasvim lijevo**, a zove se MSB (*most significant bit*). Sljedeći digitalni krug ostvaruje takvu diobu, a radi kao *FCS generator.* U ovome LFSR krugu na ulazu, MSB bit je sasvim desno.



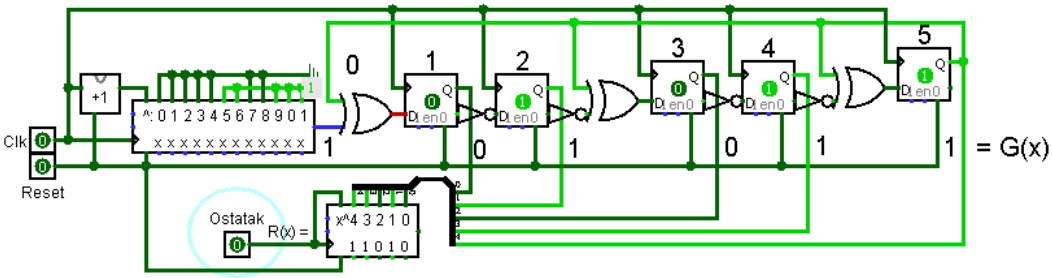
Slika 19.10: FCS generator $[G(x) = 1+x+x^2+x^5]$ *(Pazi! Redoslijed je obrnut u odnosu na prethodan opis i u ulaznome registru, MSB se nalazi sasvim desno!) Ulazna poruka najprije se zbraja izlazom iz LFSR (povratna veza), a u ostalim XOR vratima rezultat zbroja u povratnim vezama, zbraja se i posmiče prema izlazu.*

Srednja 4 bita djelitelja 100111 = $(x^5 + x^2 + x + 1) = \overset{MSB}{[}100111\overset{LSB}{]}$ ili $\overset{LSB}{[}111001\overset{MSB}{]}$ zbrajaju se (XOR) između ulaznoga registra informacijskoga LFSR (izlazni bit MSB sasvim desno) i posmičnoga registra LFSR s $n = 5$ stupnjeva. Prvi i zadnji bitovi djelitelja uvijek su jednaki x^0 = "1". Svaki stupanj

posmičnoga registra (ukupno n = 5 stupnjeva) plus jedna XOR vrata ispred 1. stupnja, zbraja "1" (x^0) djelitelja. Napomenimo da je redoslijed ovih stupnjeva u LFSR, od LSB do MSB tj. [$R(x) = 1+x+x^2+x^5$] (Galois).

Jedinice i nule prikazane na slici, jednostavno naznačuju da se odgovarajući djelitelj primjenjuje u posebno raspoređenim sklopovima XOR vrata.

Početni sadržaji posmičnoga registra su stanja "sve 0". Nakon što svih k bitova poruke $M(x) = [1100111]^{0\,1\,2\,3\,4\,5\,6}$ i dodanih "0" (onoliko koliki je broj memorijskih mjesta LFSR = 5) prođe kroz XOR vrata sasvim desno, sadržaji stupnjeva posmičnoga registra su željenih n = 5 bitova ostatka $[11010]^{MSB\quad LSB}$. Jednom, kada su smješteni u stupnjevima posmičnoga registra, ovi bitovi mogu se iščitati paralelnim prijenosom u poseban registar (vidi sliku 19.11).



Slika 19.11: Sklop za dijeljenje polinoma i izračun ostatka diobe R(x) = FCS

19.7. 19.7. Provjera FCS generatora

Pretpostavimo da niz poruke od jedne "1", sedam "0" i 5 dodatnih "0" ([0000000000001]^5), ulazi u gornji sustav pa provjerite je li na kraju, u posmičnome registru niz ostatka jednak $[11010]^{4\,3\,2\,1\,0}$ = FCS.

posmik	ulazni vektor	povratna veza	sadržaji posmičnoga registra (s lijeva u desno)					
			0	0	0	0	0	
1.	1	0	1	0	0	0	0	
2.	0	0	0	1	0	0	0	
3.	0	0	0	0	1	0	0	
4.	0	0	0	0	0	1	0	
5.	0	1	0	0	0	0	1	
6.	0	0	1	1	1	0	0	
7.	0	0	0	1	1	1	0	
8.	0	1	0	0	1	1	1	
9.	0	1	1	1	1	1	1	
10.	0	1	1	0	0	1	1	
11.	0	1	1	0	1	0	1	
12.	0	0	1	0	1	1	0	
13.	0	1	**0**	**1**	**0**	**1**	**1**	= FCS
			R0	R1	R2	R3	R4	
			Ostatak (od MSB do LSB) = $x^4 + x^3 + x$					

Provjerite radi li se o ispravnome ostatku koristeći produženu diobu (*long division*)[6] polinoma [0000000000001] s [100111].

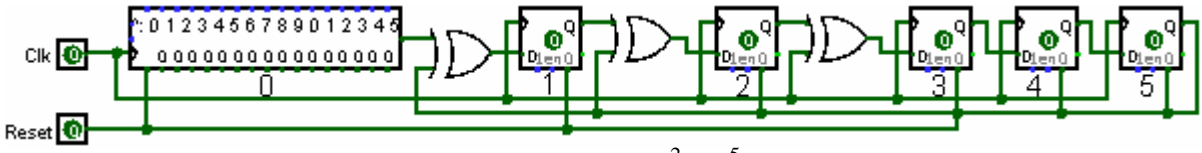
19.8. 19.8. Prijemnik-dekoder

Prijemnik (*dekoder*) isti je kao i koder. Jedina razlika je što se kroz lijevi posmični registar, sada propušta niz bitova primljen preko kanala (prvi u LFSR ulazi MSB). Na kraju informacijskoga vektora niz sadrži FCS (koji se pridružio kao ostatak diobe izračunat na *predajnoj* strani). Dakle, polinom-

[5] "Jedinični" odziv?

[6] *long division* … produžena dioba (odnosi se na dodavanje onoliko "0" polinomu poruke koliko LFSR za dijeljenje ima stanja)

djeljenik nalazi se u ulaznome spremniku (lijevi LFSR), a član najviše potencije, smješten je sasvim desno u tome istome spremniku kao i u koderu (slika 19.12).



$$G(x) = 1 + x + x^2 + x^5$$

Slika 19.12: Bitovi primljene poruke uključujući i FCS - najprije ulazi MSB. Izlaz iz LFSR istovremeno se zbraja u svim XOR vratima

Sada, primljeni (ulazni) bitovi, nakon prolaza kroz kanal, posmiču se s lijeva. Opet je prvi MSB, a stanja registra "sve 0". Nakon bitova poruke, posmiču se i FCS bitovi. Ako nema pogreške, stanja posmičnih registara sadrže niz "sve 0". Iz posmičnoga registra, ostatak se može očitati paralelno, usporediti s vektorom "sve 0" i ako je rezultat usporedbe istinit, informacijski vektor primljen je ispravan.

Imajte na umu da, iako ovaj prijemnik primjenjuje diobu polinoma, također se može koristiti i kao koder. U stvari, shema (slika 19.12) standardna je shema za diobu polinoma i proizvodnju rezultata. Međutim, prva shema (slika 19.10) učinkovitija je kao koder, jer se prateće 0 ne unose izričito u registar da bi se napravila dioba koja proizvodi FCS. Činjenica da se bitovi poruke posmiču u registar na desnoj strani vodi brigu o tome. Dakle, pri stvaranju ostatka, shema na slici 19.10, učinkovitija je od sheme na slici 19.11 (odnosno 19.12), jer izravno dijeli i prateće nule.

Napokon napomenimo da su ove jednostavne digitalne strukture praktične primjene za proizvodnju FCS i za provjeru primljene poruke, a moguće su zbog *cikličke* (kružne) strukture CRC. Svaki FCS predstavlja jedinstven "otisak" polinoma $M(x)$, s ciljem provjere ispravnosti prijenosa preko kanala poruke koja se sastoji od tisuća ili čak milijuna bitova.