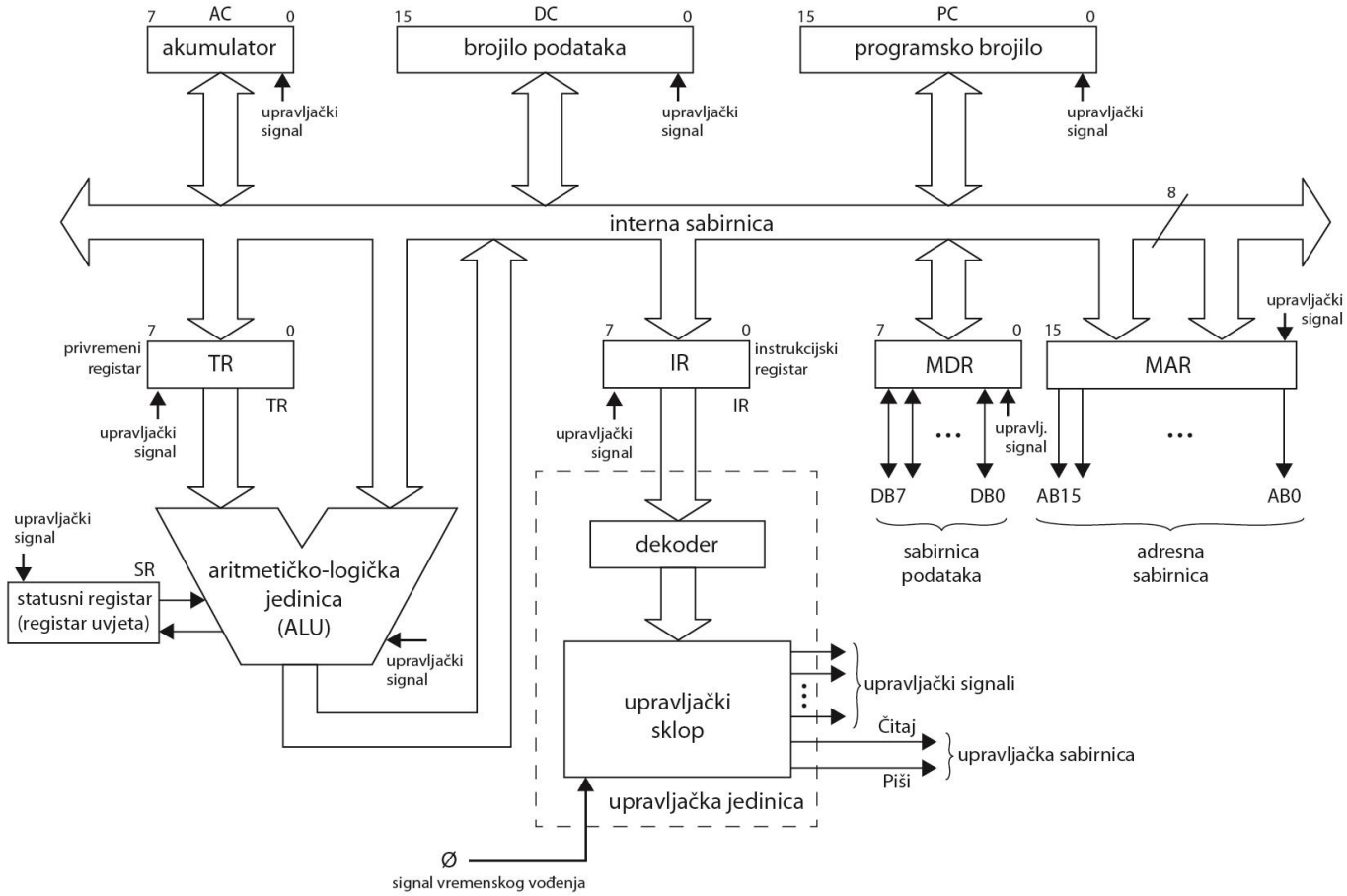
Građa računala

Elementi ALU

Elementi standardne arhitekture procesora

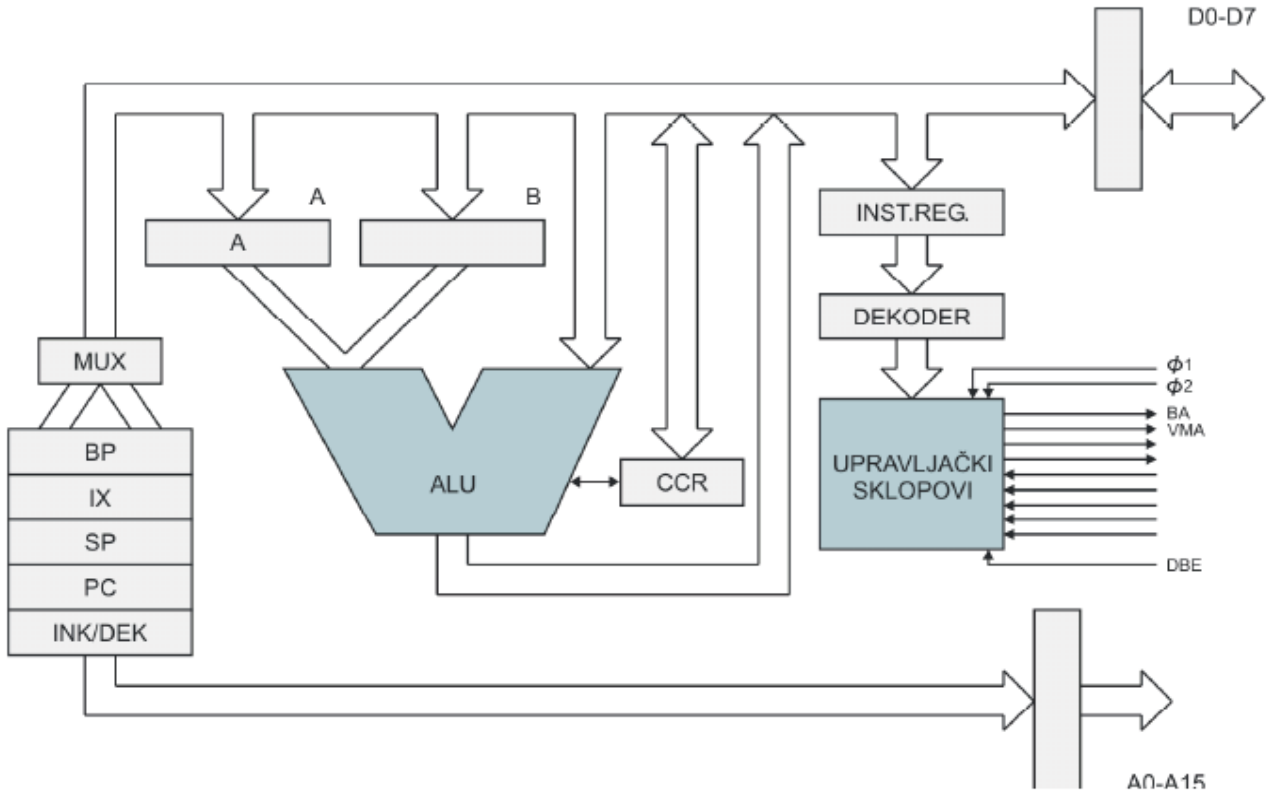
- Upravljačka jedinica
- Aritmetičko-logička jedinica (ALU)
- Jedan ili više akumulatora
- Registri opće namjene
- Adresni registri
- Interne sabirnice

Pojednostavljeni model procesora



MC 6800

Primjer: MC 6800



Sklopovi procesora

- Mikroprocesor ima:
  - sklopove za rukovanje podacima i
  - upravljačke sklopove

Sklopovi za rukovanje podacima

- aritmetičko-logička jedinica
- akumulatori
- registri opće namjene
- registar uvjeta (status registar, PSW, CCR)
- adresni registri
- segmentni registri

Uloga ALU

- Uobičajene operacije koje obavlja ALU:
  - Aritmetičke operacije (npr. zbrajanje i oduzimanje)
  - Logičke operacije (npr. AND i OR)
  - Premještanje podataka (npr. Load i Store)
- ALU svoje ime duguje osnovnim operacijama koje izvršava
- ALU ostvarujemo korištenjem kombinacije različitih vrsta krugova/sklopova

Komponente dizajna ALU

- ALU komponente koje rade aritmetičke operacije (zbrajanje, oduzimanje, dijeljenje, množenje) su dizajnirane *oko* sklopova koji su dizajnirani za takve operacije, tj.koji kvalitetno odrađuju takve poslove/algoritme
- ALU komponente koje odrađuju poslove kao što su FP operacije, ili decimalne operacije su obično puno kompleksnijeg dizajna i za njih smo kroz povijest razvoja često puta koristili naziv *koprocesori*
- Koprocesori rade “u harmoniji” sa glavnim procesorom

Specifikacije dizajna ALU

- Osnovni kriterij za odabir specifikacije dizajna ALU je ISA (Instruction Set Architecture), pošto ALU mora moći odraditi sve poslove koji su navedeni u ISA
- Izvršavanje instrukcije u procesoru ostvarujemo kroz manipulaciju I premještanje podataka koji su pridruženi instrukciji ili instrukcijama
- Premještanje podataka se radi kroz datapath, tj.tok podataka
- Navigaciju podacima u toku podataka se odrađuje kroz korištenje funkcije učitavanja (LOAD)

Von Neumann-ovo računalo

- Program koji se pokreće pohranjuje se u binarnom formatu u memorijskoj jedinici – kako bi rezultati izvršavanja mogli biti korišteni za neki “efekt” u programu – tzv. “Stored Program Concept”
- Ovaj concept kaže da su instrukcije pohranjene u memoriji skupa sa podacima u formatu koji je čitljiv CPU-u, a računalo može manipulirati sa tim podacima pošto instrukcije i podaci ovise i o drugim parametrima (npr. kontrolnim stanjima)
- Instrukcije se izvode serijski (sekvencijalno) prateći kontrolni tok programa kako je zapisan u memoriji

Zašto nam je to važno?

- Zato što tako rade sva računala
- Zato što je to *defaultni* način izvršavanja programa
- Kad ne bismo imali ovakav concept, sve bi se instrukcije morale pokretati ručno – nepraktično, beskorisno i nemoguće za korištenje

Arhitektura von Neumannovog računala

- Upravo kako bismo bili u mogućnosti ostvariti ovakvu metodu izvršavanja na von Neumannovom konceptu računala, trebamo uobičajene gradivne jedinice von Neumannovog računala:
  - ALU
  - Memory
  - Input I output
  - Kontrolna jedinica
- Isto tako, ALU mora imati registar koji se obično zove akumulator
- Kontrolna jedinica mora imati registar/brojač koji se zove PC (*Program Counter*, programsko brojilo) – prati koja je slijedeća instrukcija za izvršavanje u toku programa
- Ovi registri su obično ostvareni kao memorijske komponente u CPU-u koje pomažu ispravnom izvršavanju toka programa

Kako radi von Neumannovo računalo?

- Izvršava ili *emulira* Fetch-Decode-Execude sekvencu za izvršavanje programa
  - *Fetch* – dobavlja instrukciju iz memorije sa adrese koja je zapisana u PC, inkrementira se PC tako da se namjesti na adresu iduće instrukcije
  - *Decode* – dekodiranje instrukcije korištenjem kontrolne jedinice
  - *Execute* – izvršava se pgraom korištenjem podatkovnog toka, kontrolne jedinice i ALU
    - Dobavljaju se podaci iz memorije za izvršavanje instrukcije
    - Rezultati koje treba osvježiti u memoriji se osvježavaju na kraju izvršavanja instrukcije
- Nakon ovog ciklusa, kreće se od početka

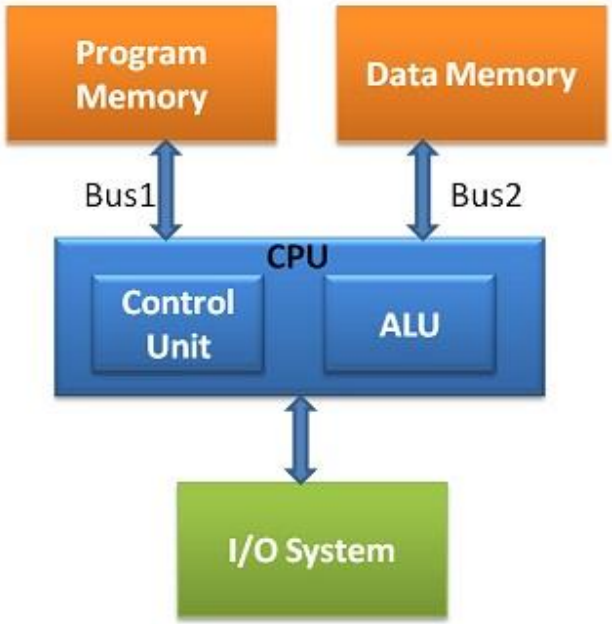
Problemi dizajna von Neumannovog računala

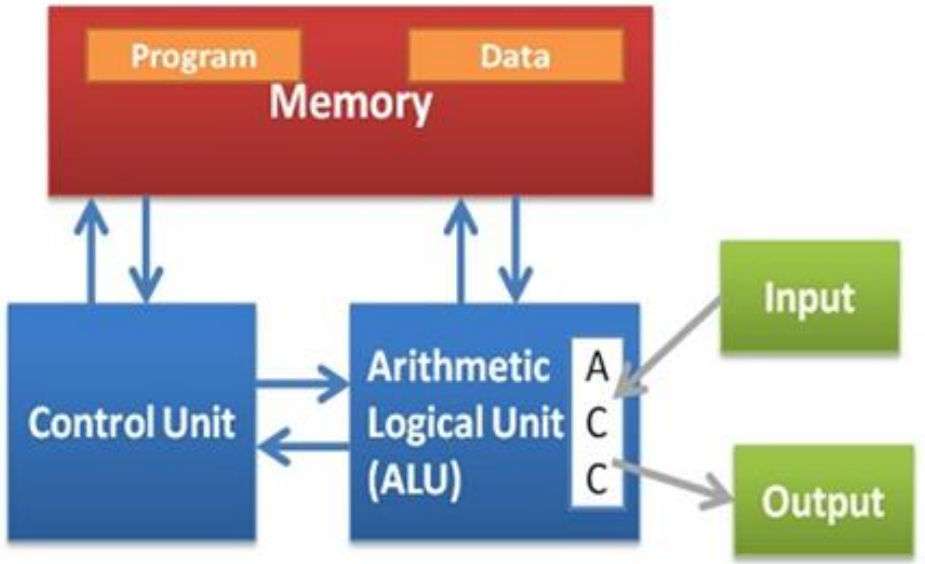
- Dizajnirano da procesira jednu po jednu instrukciju (nema paralelizma) – CPU ne može u isto vrijeme raditi i *Fetch* i *Execute*
- Poseban dio tog problema je činjenica da i instrukcije i podaci koriste isti tok podataka – problem memorije zbog natjecanja
- Problem je I činjenica da je memorijski ciklus dulji od CPU ciklusa – CPU čeka puno vremena da budu “in sync” – dodatno usporavanje
- Sve ovo skupa agregatno zovemo “usko grlo” von Neumannovog računala

Potencijalna rješenja problema

- Rješenje je (i opet) - bolji dizajn (ko-dizajn)
  - Memorija i CPU kao odvojena sučelja, jedno samo za instrukcije, drugo za podatke
  - Dizajn CPUa sa *cache* memorijom značajno povećava iskoristivost puta između CPU-a i glavne memorije, što povoljno utječe na smanjenje vremena besmislenog čekanja koje CPU mora izvoditi
  - Korištenje boljih programskih jezika, zbirki funkcija (*library*), funkcija I algoritama, kao i paradigm programiranja kako bismo smanjili bemisleno razbacivanje podataka amo-tamo u normalnom toku izvršavanja programa
- Prevedeno, sigurno nećemo za fantastično efikasno programiranje modernih aplikacija koje kvalitetno koriste resurse “ispod” (CPU, memorija, …) iskoristiti programske jezike tipa FORTRAN
- Opće IT kulture radi, jedna od arhitektura koja ide u smjeru ovih rješenja je i tzv. *Harvard arhitektura* – posebni sklopovi za pohranu, posebni putevi za instrukcije i podatke

von Neumannovo računalo vs Harvard računalo, slikovito





Broj bitova u riječi u različitim procesorima

- Intel 4004 (1971. godine) - 4 bita
- prva generacija (kraj 1971.)- 8-bitni
- Intel 8086 (1978.) - 16 bita
- Treća generacija - 32-bitna
- Četvrta generacija - 64-bitna
- Grafički procesori i procesori za posebne namjene (VLIW, multimedijski) -128 bitova i više

Prikaz znakova

- Alfanumerički znakovi (A – Z, a – z, 0 – 9, simboli *, -, +, !, ?, @) predočeni su binarnim uzorcima
- **ASCII** (*American Standard Code for Information Interchange) obično se koristi za kodiranje* alfanumeričkih znakova
- Svaki je alfanumerički znak predočen 7-bitnim kodom
- ukupno **128** znakova, od čega je **96** znakova “normalnih”, koji su ispisni, (engl. *printing characters), a **32** znaka su neispisni znakovi za upravljačke funkcije*

ASCII kod

		0 000	1 001	2 010	3 011	4 100	5 101	6 110	7 111
0	0000	NULL	DCL	SP	0	@	P	‘	p
1	0001	SOH	DC1	!	1	A	Q	a	q
2	0010	STX	DC2	"	2	B	R	b	r
3	0011	ETX	DC3	#	3	C	S	c	s
4	0100	EOT	DC4	$	4	D	T	d	t
5	0101	ENQ	NAK	%	5	E	U	e	u
6	0110	ACK	SYN	&	6	F	V	f	v
7	0111	BEL	ETB	’	7	G	W	g	w
8	1000	BS	CAN	(	8	H	X	h	x
9	1001	HT	EM	)	9	I	Y	i	y
A	1010	LF	SUB	*	:	J	Z	j	z
B	1011	VT	ESC	+	;	K	[	k	}
C	1100	FF	FS	,	<	L	\	l	|
D	1101	CR	GS	-	=	M	]	m	}
E	1110	SO	RS	.	>	N	^	n	~
F	1111	SI	US	/	?	O	_	o	DEL

ASCII kod

Unicode

- Problem sa ASCII kodom - različite zemlje različite kodne stranice
- Unicode koristi jedinstven broj za svaki znak, bez obzira na platformu, na program, na jezik.
- Njime je opisan SVAKI znak iz SVIH poznatih jezika
- Koristi 16 bitova,
- Moguće kodirati čak $2^{16} = 65536$ različitih znakova

Prikaz cijelih brojeva

- Većina današnjih računala koristi pozicijski brojevni sustav s bazom 2 (**binarni brojevni sustav**)
- Postoje izvedbe posebnih računala koja za prikaz brojeva koriste simboličke prikaze
- Opseg *cijelih brojeva (engl. integer) proteže se na područje:*
  - *cijelih negativnih brojeva*
  - *nule* i
  - područje cijelih pozitivnih brojeva

Prikaz cijelih brojeva

- U računalu se koriste različiti načini za prikaz cijelih brojeva:
  - predznak-apsolutna vrijednost (engl. *signed and magnitude);*
  - jedinični ili nepotpuni komplement (engl. *one's complement);*
  - potpuni ili dvojni komplement (engl. *two's complement).*

Prikaz cijelih brojeva

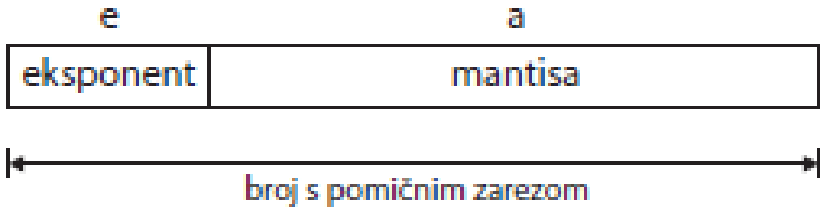
- Prikaz brojeva u notaciji potpunim ili dvojnim komplementom - *najčešće je korišteni način* prikaza cijelih brojeva
- Svojstva brojeva predočenih u dvojnom komplementu:
  - dvojni komplement je zaista “pravi” komplement jer vrijedi da je +X + (-X) = 0
  - postoji samo jedna nula
  - pozitivan broj - najznačajniji bit 0, negativni broj 1
  - opseg prikaza brojeva je od -2^n-1 do $+2^n-1-1$
  - dvojni komplement dvojnog komplementa od X je X

Prikaz brojeva s pomičnim zarezom (floating-point number)

- Upotrebljavaju se u računanju na području znanosti
- Vrijednosti brojeva kreću u vrlo širokom rasponu: od vrlo malih vrijednosti (npr. 2^{-120}) do vrlo velikih vrijednosti (npr. 2^{+120})
- Viši programski jezici dopuštaju definiranje varijabli tipa ***real*** i one mogu imati vrijednosti između dva slijedna cijela broja
- U računalu su ti brojevi predočeni u obliku brojeva s pomičnim (binarnim) zarezom

Prikaz brojeva s pomičnim zarezom (floating-point number)

- Broj se u tom načinu prikaza sastoji od dva dijela
  - eksponenta;
  - mantise;
- Možemo ga predočiti u obliku:



  - a - *mantisa*
  - r -*izabrana* baza brojevnog sustava
  - e -*eksponent*

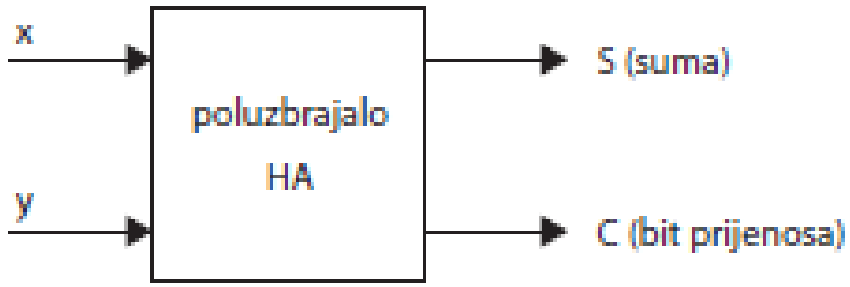
Prikaz brojeva s pomičnim zarezom (floating-point number)

- Broj s pomičnim zarezom u računalu se pohranjuje slijedom bitova koji definiraju dva polja:
  - Polje eksponenta *e i*
  - Polje mantise (argumenta)
- Vrijednost baze ***r*** eksplicitno se ne pohranjuje u računalu

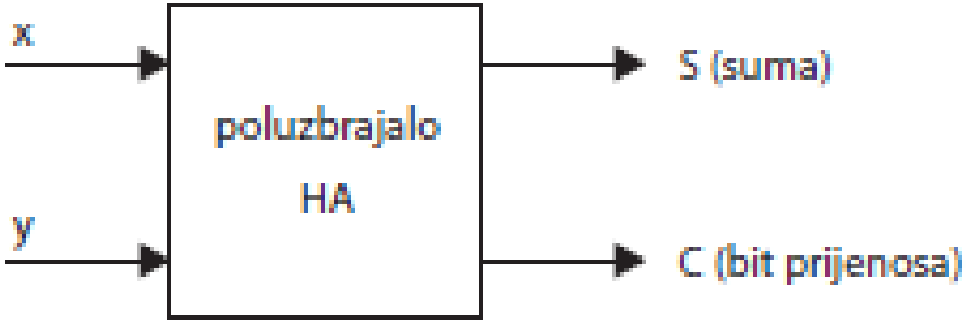
Zbrajanje dvaju binarnih brojeva

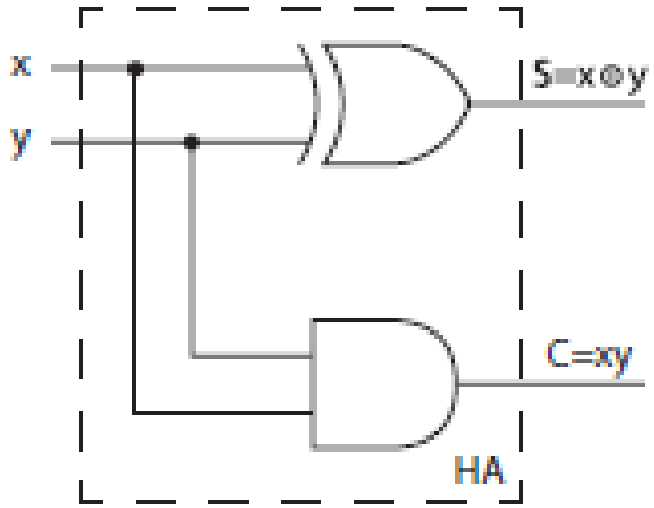
- Sklop za zbrajanje dvaju binarnih znamenaka - kombinacijski sklop
- naziva se ***poluzbrajalo*** HA (engl. *half adder)*
- *Na temelju pravila zbrajanja možemo napisati tablicu istinitosti* (ili tablicu kombinacija) te na temelju nje realizirati sklop

Ulaz x	Ulaz y	Izlaz S (suma)	Bit prijenosa C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



Poluzbrajalo HA



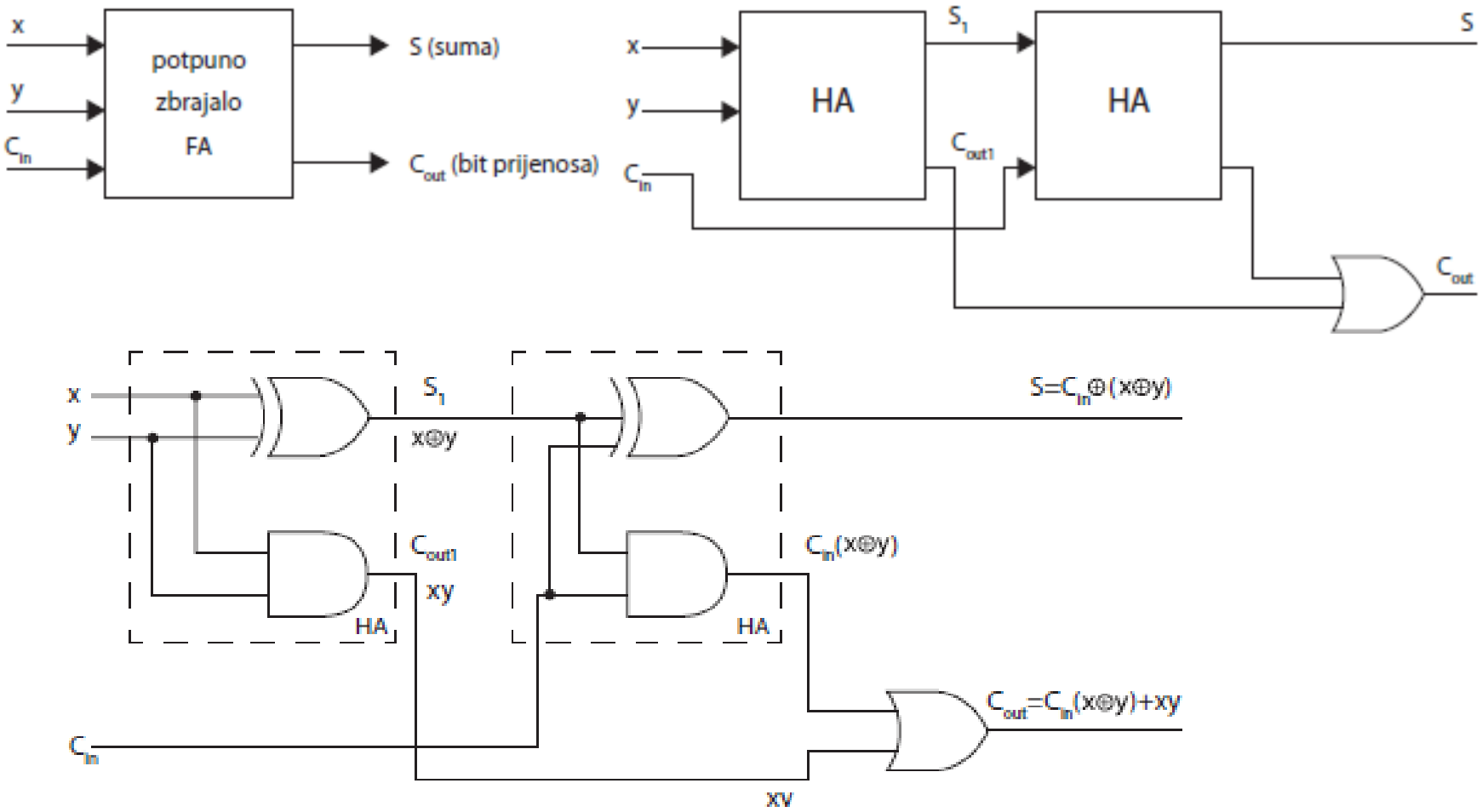


c)

Tablica istinitosti za potpuno zbrajalo

Ulaz x	Ulaz y	Ulaz C_{in}	Izlaz S (suma)	Izlaz C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Potpuno zbrajalo FA (engl. full adder)

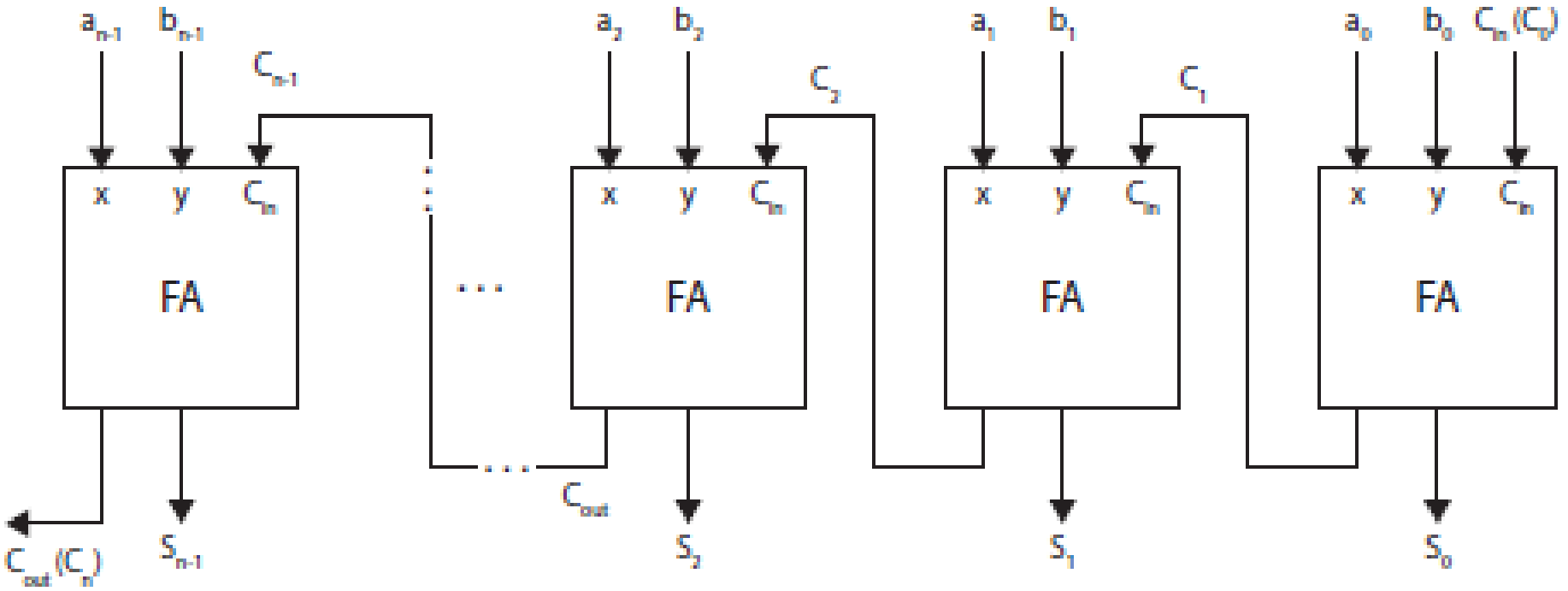


Potpuno zbrajalo

- dvorazinski kombinacijski sklop koji računa sumu triju ulaznih bitova
  - **x** i **y** – bitovi operanda (Ai , Bi)
  - **Ci** – bit prijenosa iz prethodnog stupnja
- dva izlaza: **Fi** (jednobitni rezultat) i **Ci+1**(bit prijenosa u sljedeći stupanj)

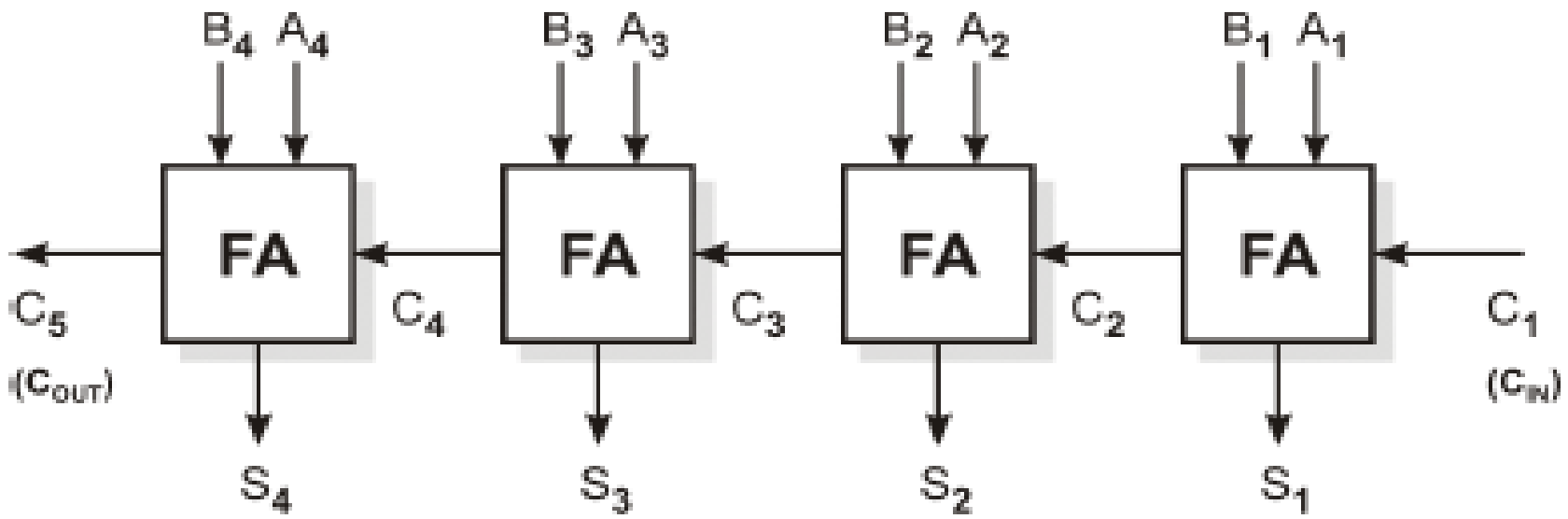
Paralelno zbrajalo

- zbraja *n-bitnu riječ A i n-bitnu riječ B u jednom koraku*



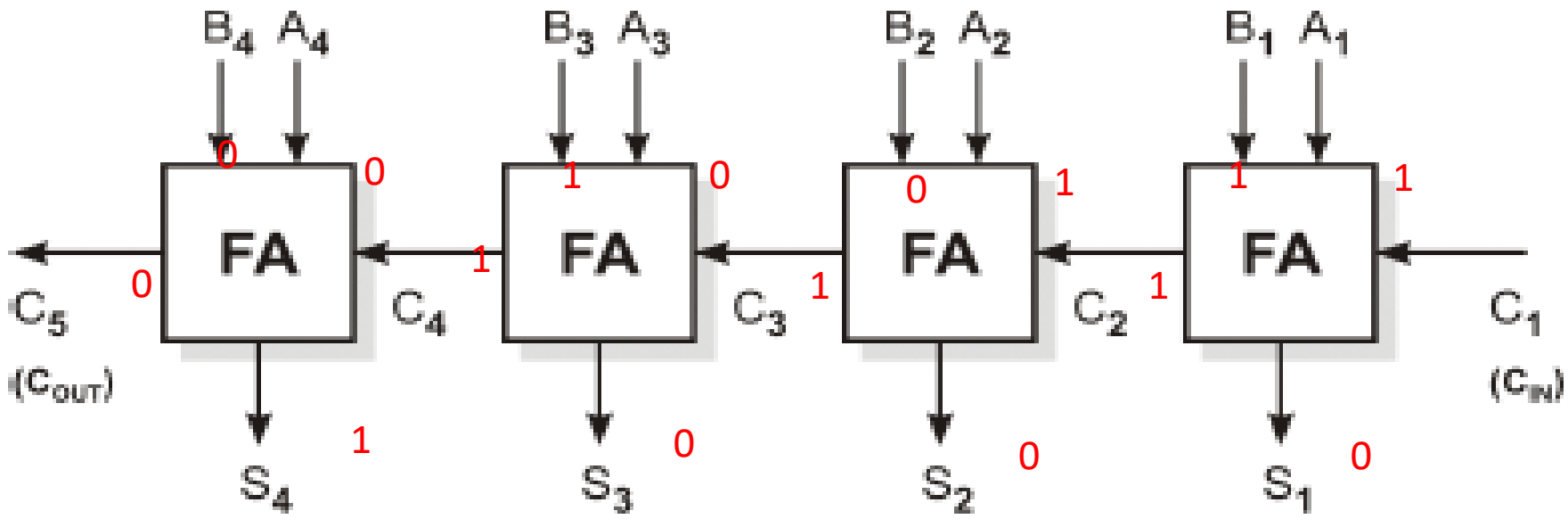
Paralelno zbrajalo

- izraz “paralelno zbrajalo” jer se podrazumijeva da će se *n zbrajanja* izvesti gotovo istodobno

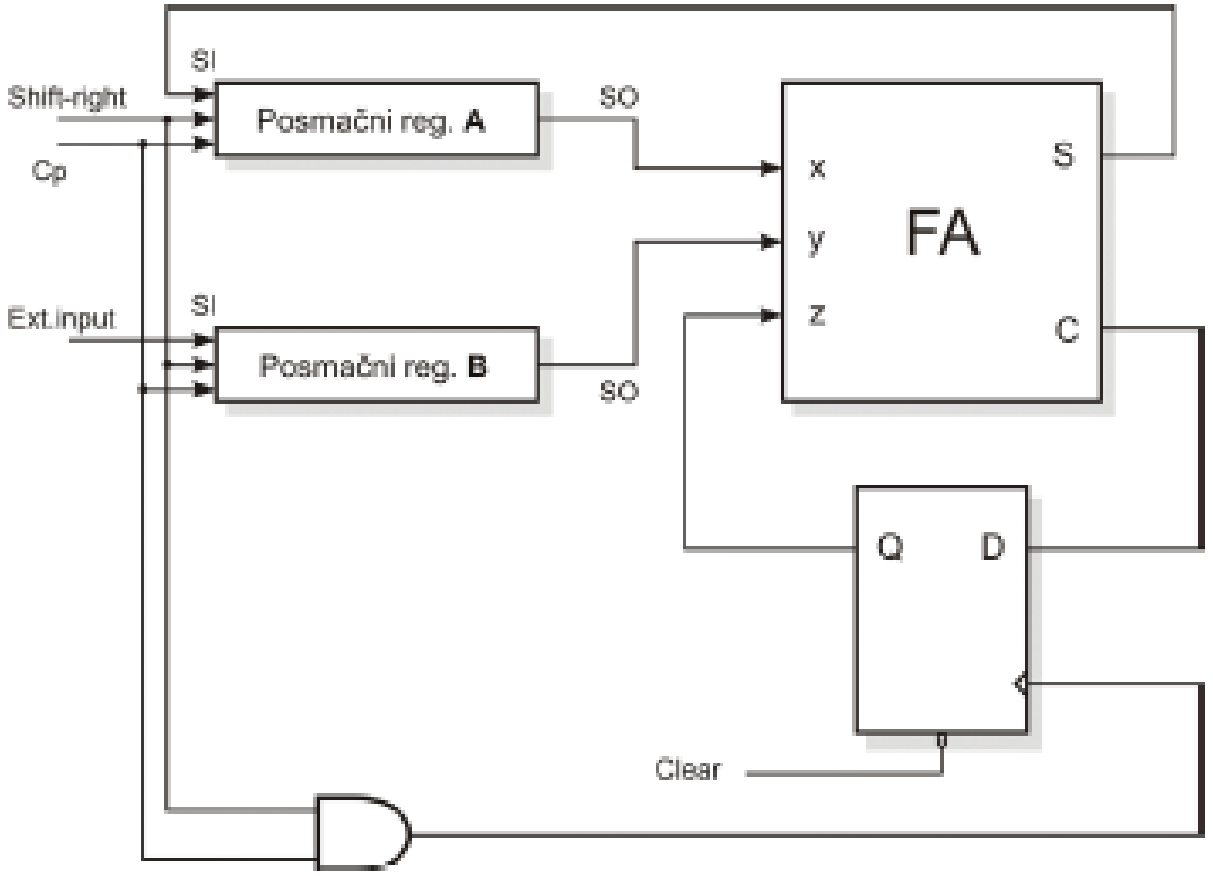


Paralelno zbrajalo

- izraz “paralelno zbrajalo” jer se podrazumijeva da će se *n zbrajanja* izvesti gotovo istodobno - primjer: 5+3   (0101 + 0011)



Serijsko zbrajalo



Serijsko zbrajalo

- U posmačnim registrima A i B pohranjeni su *n-bitni operandi*
- *Bitovi se posmakom udesno prosljeđuju jednom stupnju potpunog zbrajala*
- Rezultat zbrajanja svake bitovne pozicije pohranjuje se (i posmiče) u posmačnom registru A
- D bistabil pamti bit prijenosa iz prethodne operacije zbrajanja bitova

Oduzimanje dvaju binarnih brojeva

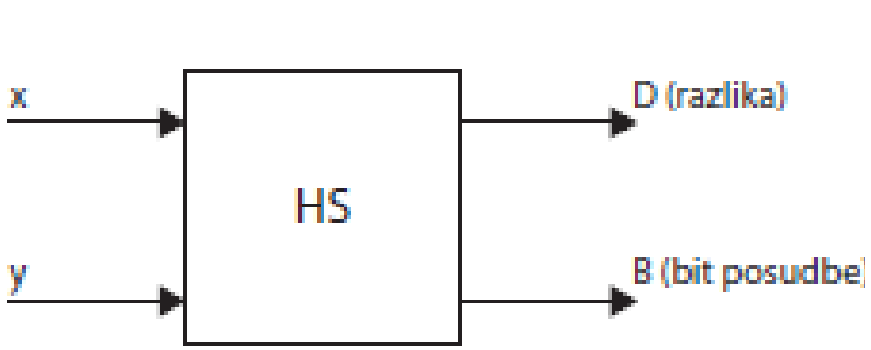
- Označimo s D razliku, odnosno diferenciju (rezultat oduzimanja)
- B posudbu (B od engl. *borrow)*
- *Na temelju osnovnog pravila za oduzimanje* možemo napisati tablicu istinitosti

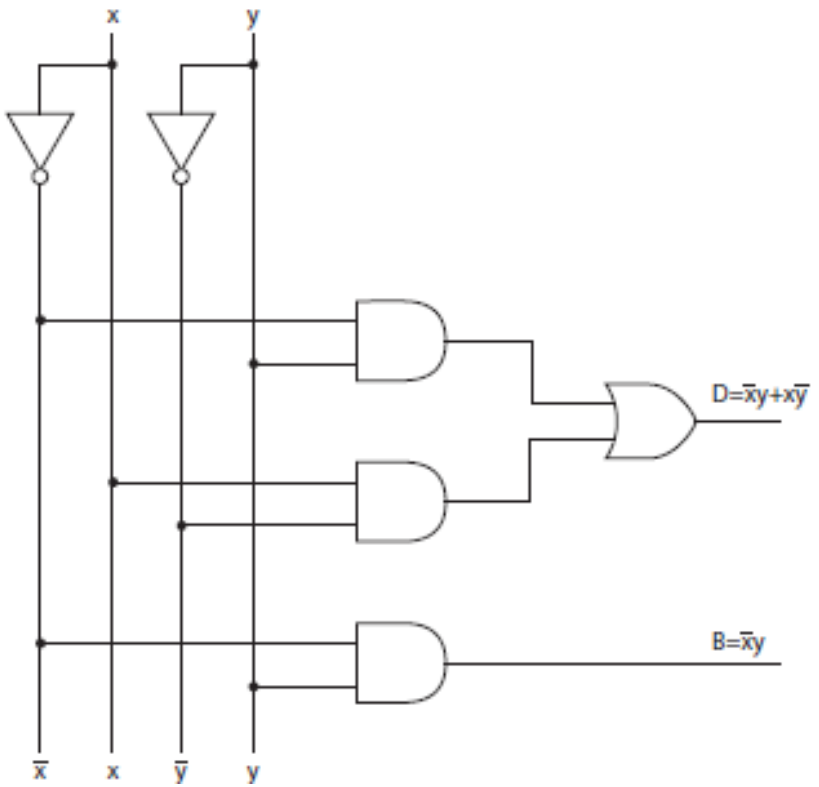
Ulaz x	Ulaz y	Izlaz D (razlika)	Bit posudbe B
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Oduzimanje dvaju binarnih brojeva

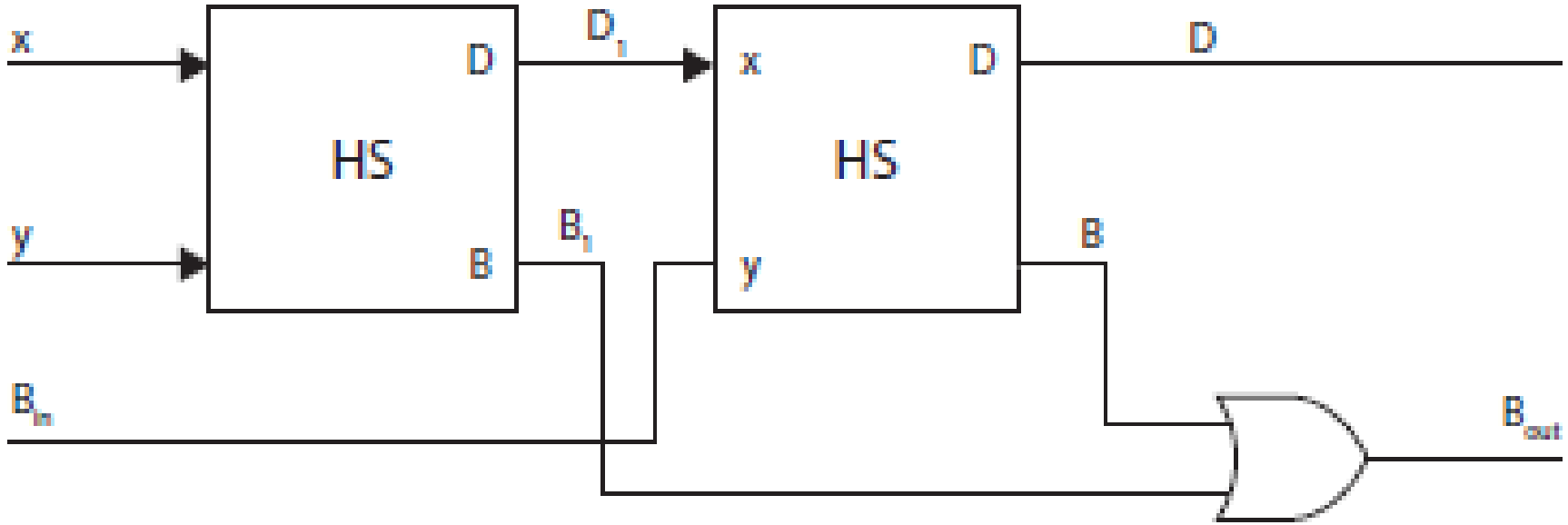
- Sklop koji podržava logičke funkcije za operaciju oduzimanja naziva se *poluoduzimalo HS (engl. half-subtractor)*
- Slično kao u slučaju potpunog zbrajala, uporabom dvaju poluoduzimala može se dobiti *potpuno oduzimalo FS (engl. full-subtractor)*

Poluoduzimalo



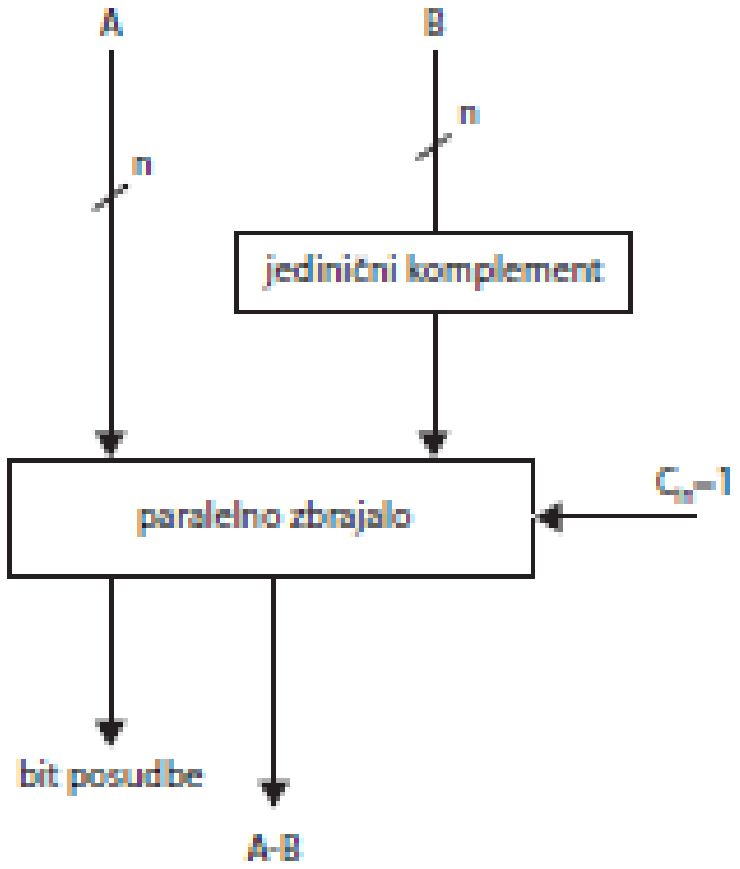


Potpuno oduzimalo sastavljeno od dva poluoduzimala



Oduzimanje pomoću komplementa binarnog broja

- Oduzimanje se može izvesti operacijom zbrajanja dvojnog ili potpunog komplementa umanjitelja (suptrahenda)
- Dvojni komplement operanda B dobiva se tako da se njegovom jediničnom komplementu pribroji Cin = 1

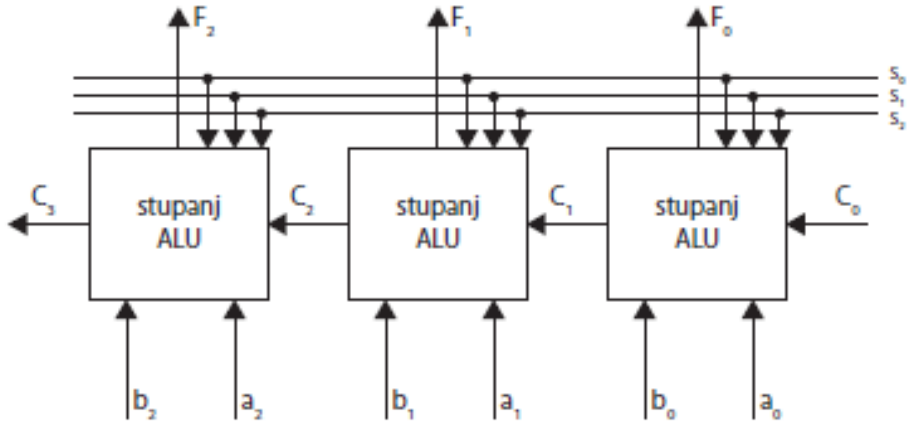


Oblikovanje jednostavne ALU

- Aritmetičko-logička jedinica (ALU) je višefunkcijski digitalni sklop
- Izvodi aritmetičke i logičke operacije
- Vrlo često se upotrebljava i za računanje efektivne memorijske adrese izvorišta operanada ili odredišta rezultata

Jednostavna ALU

- Sklop koji ima **pravilnu strukturu**

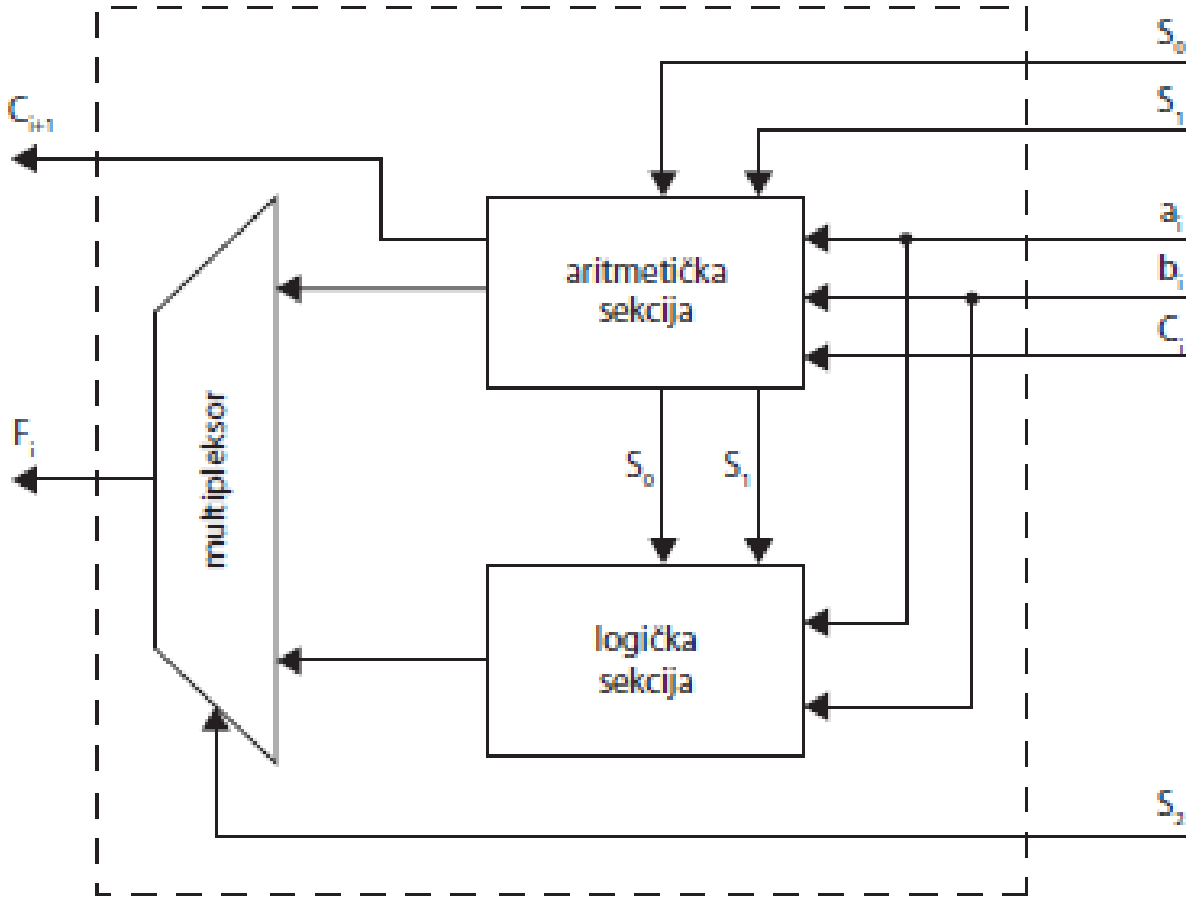


- Sastoji se od identičnih stupnjeva povezanih u kaskadu
- Svaki je stupanj odgovoran za aritmetičke ili logičke operacije **na jednom** bitu operanda
- Jednostavna ALU koja se koristi *n-bitnim (cjelobrojnim) operandima* sastoji se od *n takvih stupnjeva*
- *Svaki se od njih u funkcijskom smislu može prikazati* kao da je sastavljen od:
  - aritmetičke
  - logičke sekcije

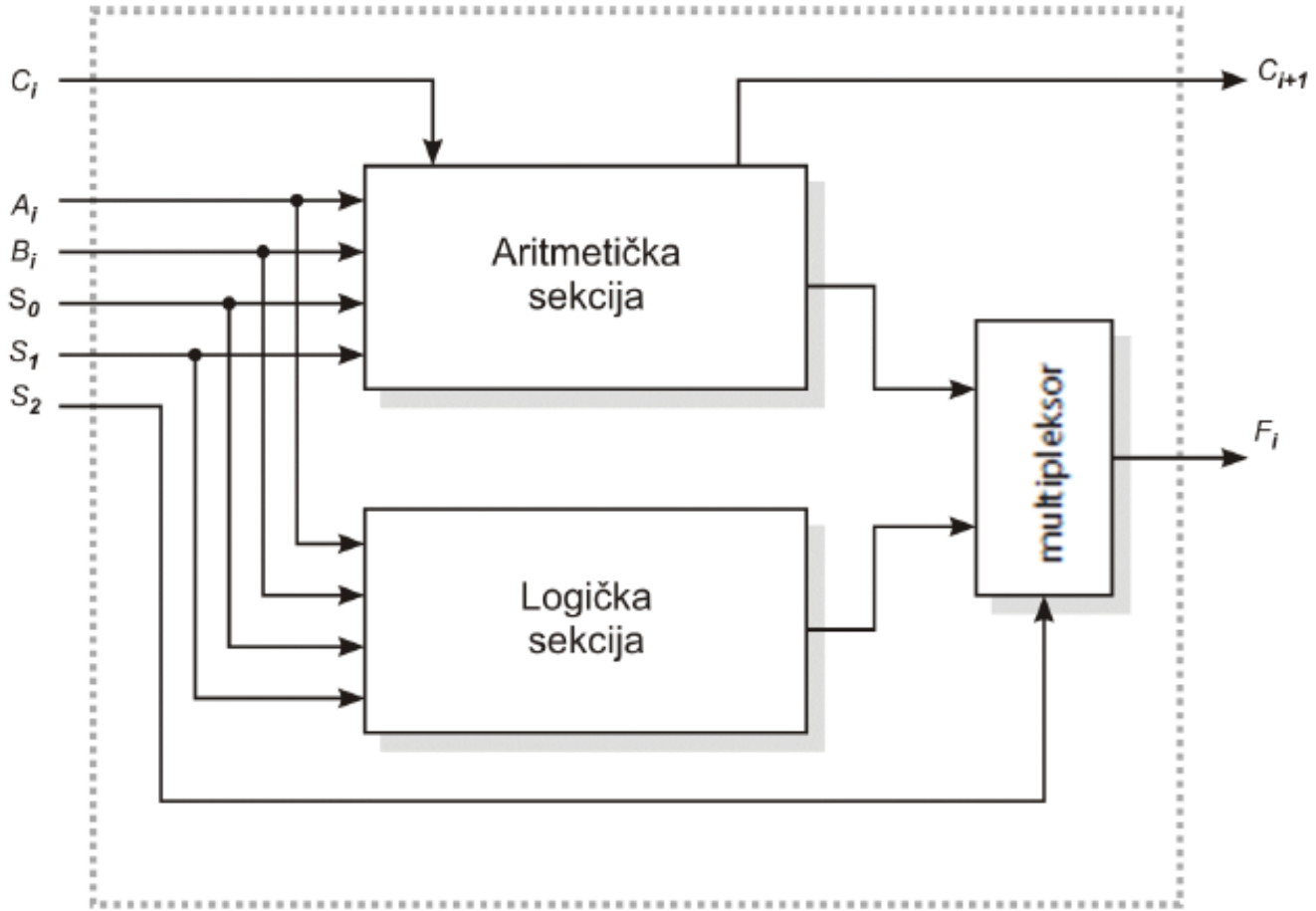
ALU – funkcionalni prikaz

- svaki stupanj može se u funkcionalnom smislu prikazati kao kombinacija:
  - sklopova aritmetičke sekcije
  - sklopova logičke sekcije

i-ti stupanj ALU



i-ti stupanj ALU



Ulazi i izlazi ALU

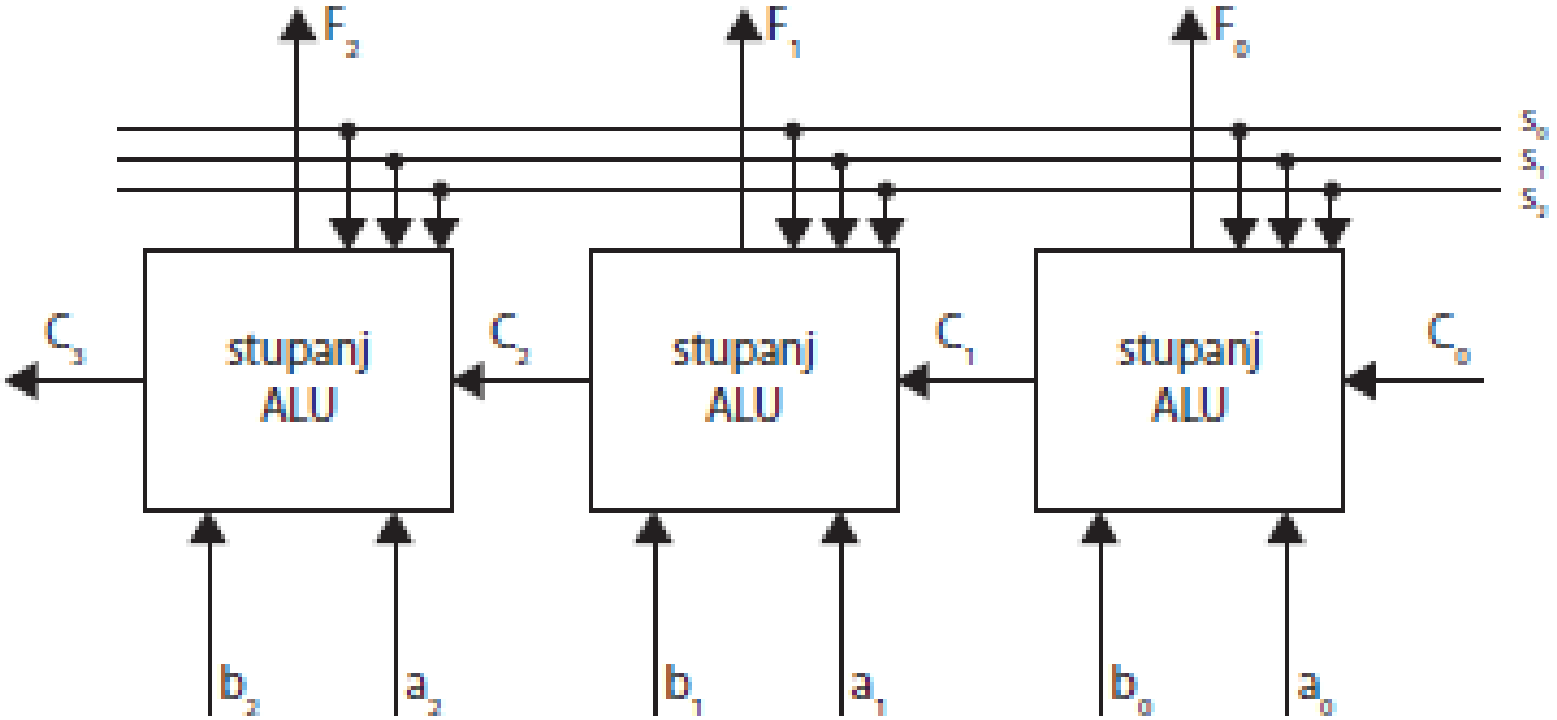
- a_i i b_i su ulazi za operande
- C_i je bit prijenosa iz prethodnog stupnja,
- F_i je jednobitni rezultat aritmetičke ili logičke operacije,
- C_{i+1} je bit prijenosa u sljedeći stupanj
- Upravljački ulazi S_o, S_1 i S_2 upotrebljavaju se za izbor različitih aritmetičkih i logičkih operacija

ARITMETIČKA SEKCIJA

- Osnovna komponenta aritmetičke sekcije jednog stupnja je ***potpuno zbrajalo***
- Povezivanjem n stupnjeva dobiva se *paralelno zbrajalo* koje zbraja dvije riječi duljine n bita
- Oduzimanje se izvodi zbrajanjem potpunog komplementa operanda
- Da bi se moglo izvoditi više operacija potrebno je dodati sklop za priređivanje operanda B

Povezivanje stupnjeva u kaskadu

- ostvaruje se pomoću linija bita prijenosa Ci i Ci+1



Oblikovanje jednostavne ALU

- Podjela stupnja aritmetičko-logičke jedinice na aritmetičku i logičku sekciju samo je funkcionalna
- Pri oblikovanju jednostavne ALU koristi se sljedeći pristup:
  - prvo se oblikuje sklopovlje aritmetičke sekcije neovisno o logičkoj sekciji
  - zatim se određuju logičke operacije koje se mogu izvesti sklopovima iz aritmetičke sekcije;
  - izvode se preinake na sklopovima da bi se mogle izvoditi sve željene logičke operacije

Aritmetička sekcija ALU

- Osnovna građevna sastavnica jednog stupnja ALU je **potpuno zbrajalo**
- Povezivanjem *n stupnjeva ALU jedinice u kaskadu dobiva se* ***paralelno zbrajalo***
- *Istodobno* zbraja dva operanda duljine *n bitova*
- Željeli bismo da jednostavna ALU ovisno o kombinaciji na njezinim ulazima izvodi osam operacija

Oblikovanje jednostavne ALU

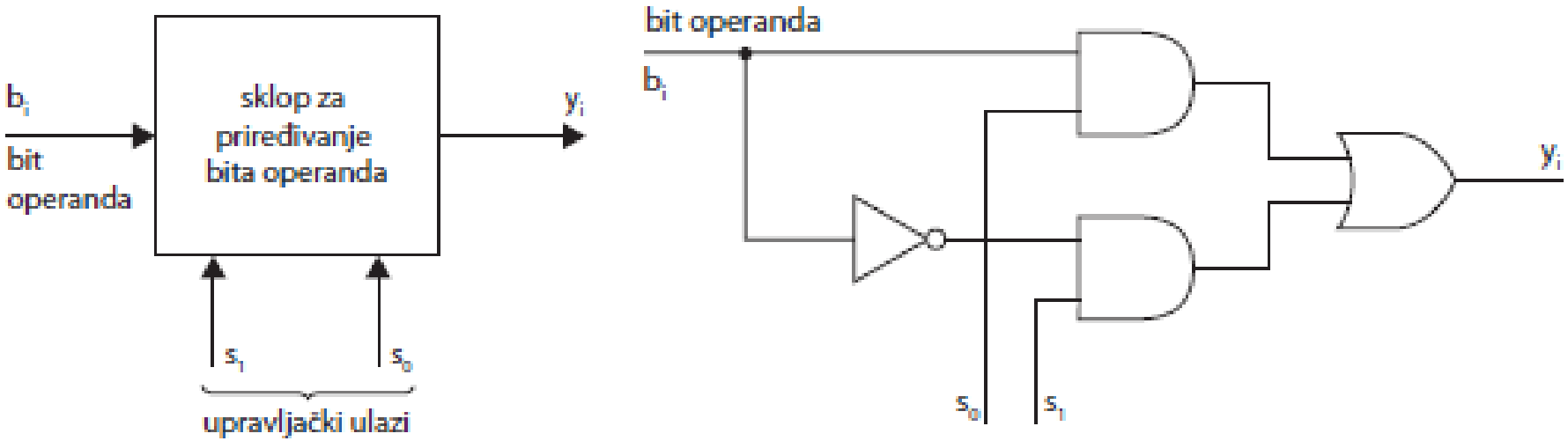
Ulazi			Izlaz F
X	Y	C_{in}	
X	Y	0	Zbrajanje: $F = X + Y$
X	Y	1	Zbrajanje s bitom prijenosa: $F = X + Y + 1$
X	Jedinični komplement: $\overline{Y}$	0	$F = X + \overline{Y}$
X	$\overline{Y}$	1	Oduzimanje: $F = X + \overline{Y} + 1$
X	0	0	Prijenos: $F = X$
X	0	1	Inkrementiranje: $F = X + 1$
X	Sve jedinice: 111...111	0	Dekrementiranje: $F = X - 1$
X	Sve jedinice: 111...111	1	Prijenos $F = X$

Oblikovanje jednostavne ALU

- prilagodbom operanda na ulazu Y i izborom vrijednosti bita prijenosa u najniži stupanj ALU Cin, ostvarili smo:
  - Zbrajanja, oduzimanje
  - Inkrementiranje operanda X,
  - Dekrementiranje operanda X,
  - Zbrajanje operanda X i jediničnog komplementa operanda Y
  - prijenos, odnosno prosljeđivanje operanda X s ulaza na izlaz

Sklop za priređivanje bita operanda y

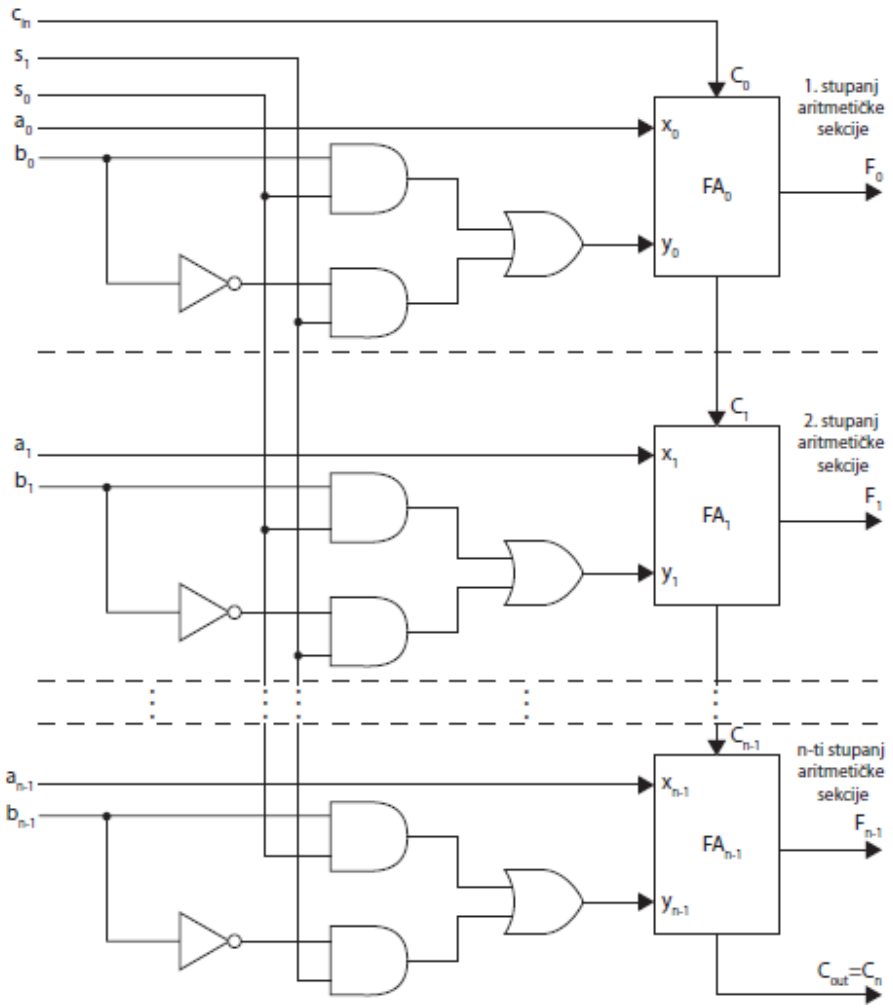
- Da bi se paralelnim zbrajalom, sastavljenim od *n potpunih zbrajala, moglo izvoditi navedenih osam* operacija, potrebni su dodatni logički sklopovi koji na ulazu Y **priređuju** operand



Priređivanje bita operanda na ulazu Y

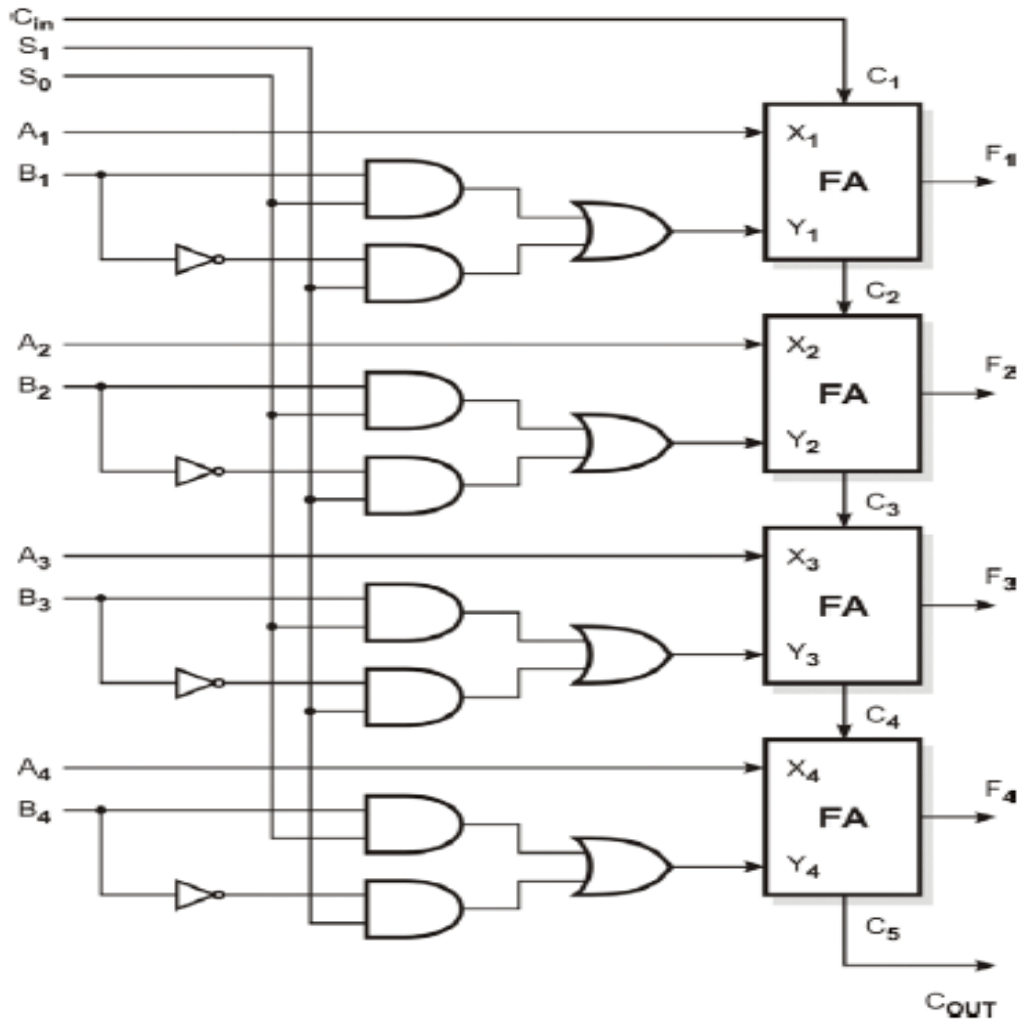
S_1	S_0	Ulaz: bit operanda	Izlaz iz sklopa za priređivanje y_i
0	0	b_i	0
0	1	b_i	b_i
1	0	b_i	$\overline{b_i}$
1	1	b_i	1

Aritmetička sekcija ALU



Kaskadiranje ALU

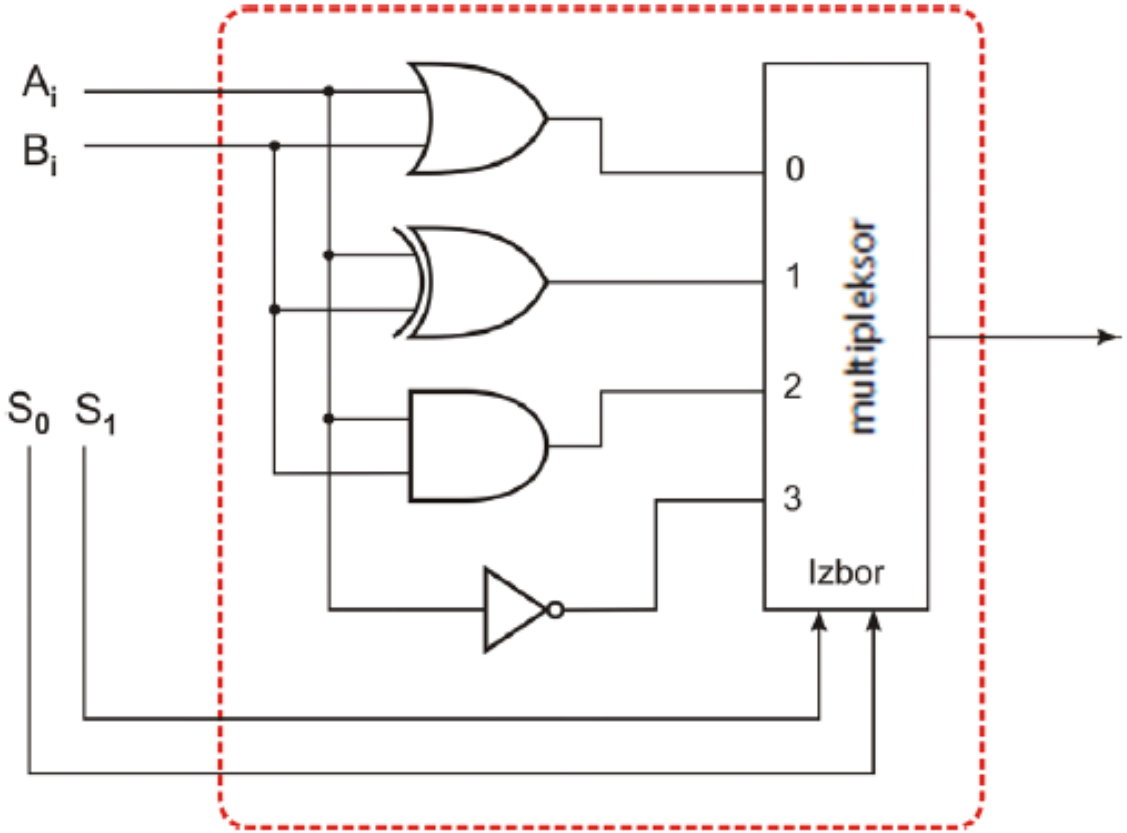
- Aritmetičke sekcije ALU povezane u kaskadu
- Zadnji stupanj ima izlaz Cout koji se upisuje u status registar



Logička sekcija

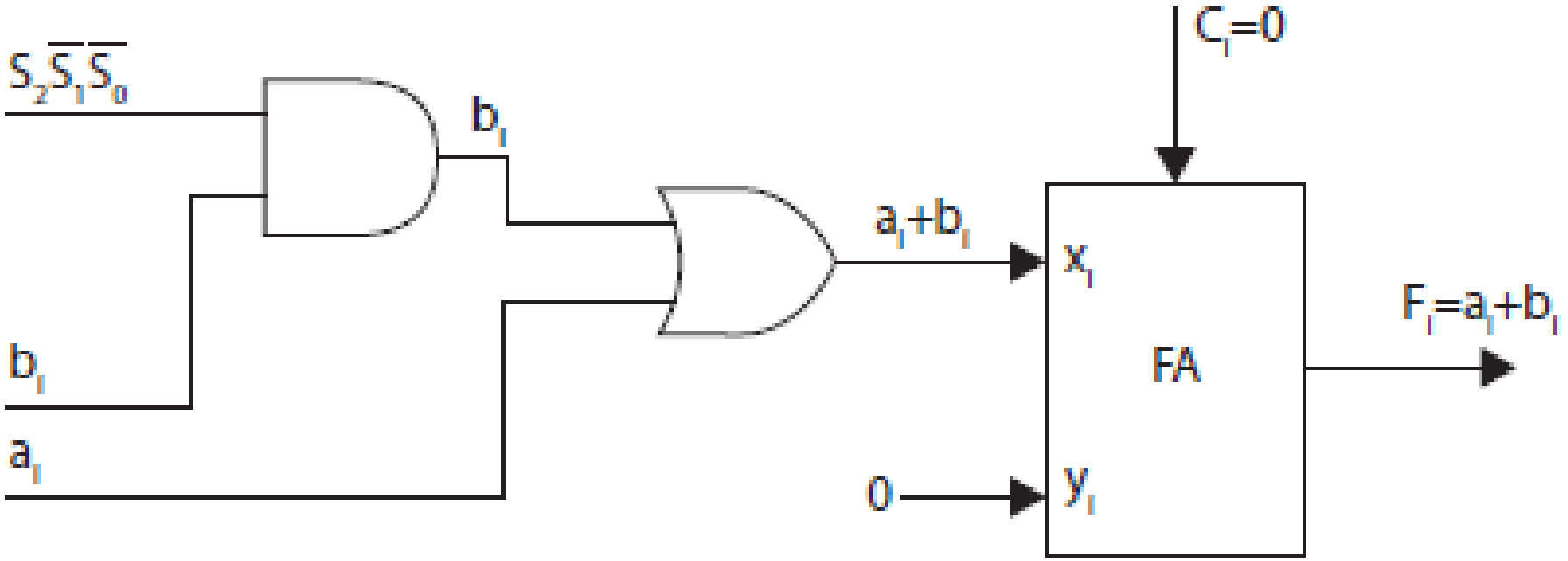
- Logičke operacije rukuju bitovima operanada izdvojeno i svaki se bit operanda promatra kao logička, binarna varijabla
- Logička sekcija treba podržati četiri osnovne logičke operacije I, NE, ISKLJUČIVO ILI i ILI
- Aritmetička i logička sekcija pojedinog stupnja stapaju se u jednu sekciju
- Upravljački ulaz S2 upotrebljava se za određivanje logičke ili aritmetičke operacije

Logička sekcija



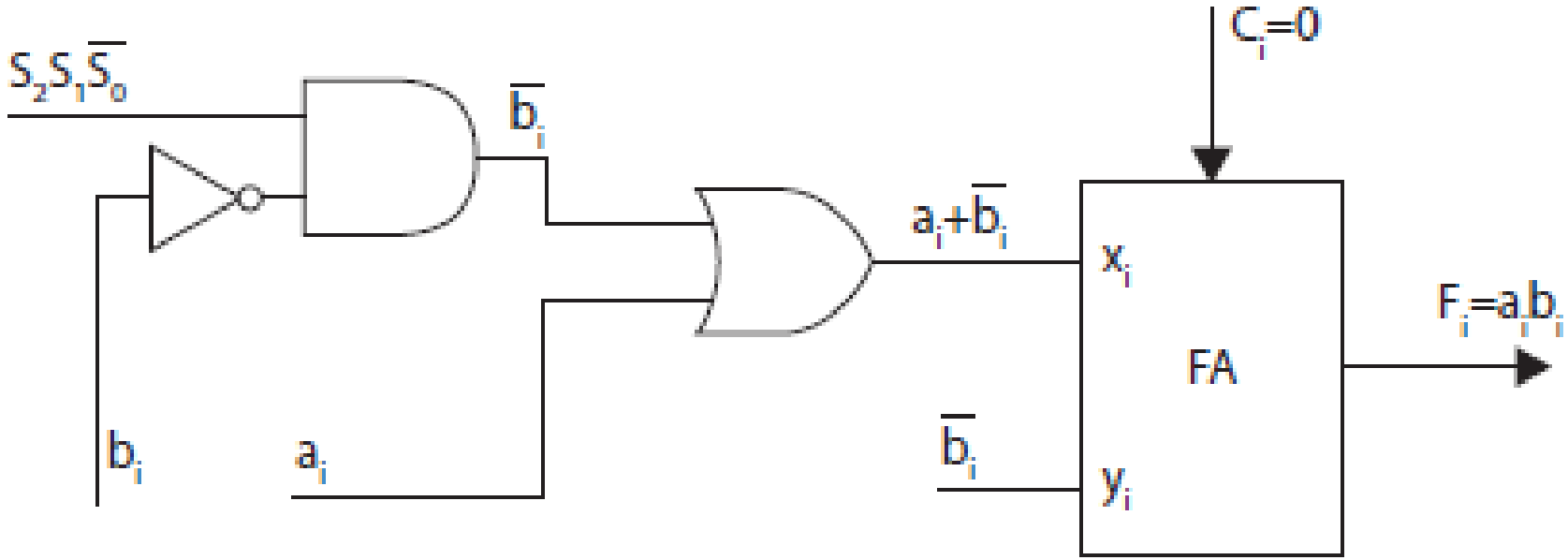
Logička operacija ILI

- Sklopovska preinaka potrebna da bi se realizirala logička operacija ILI



Logička operacija I

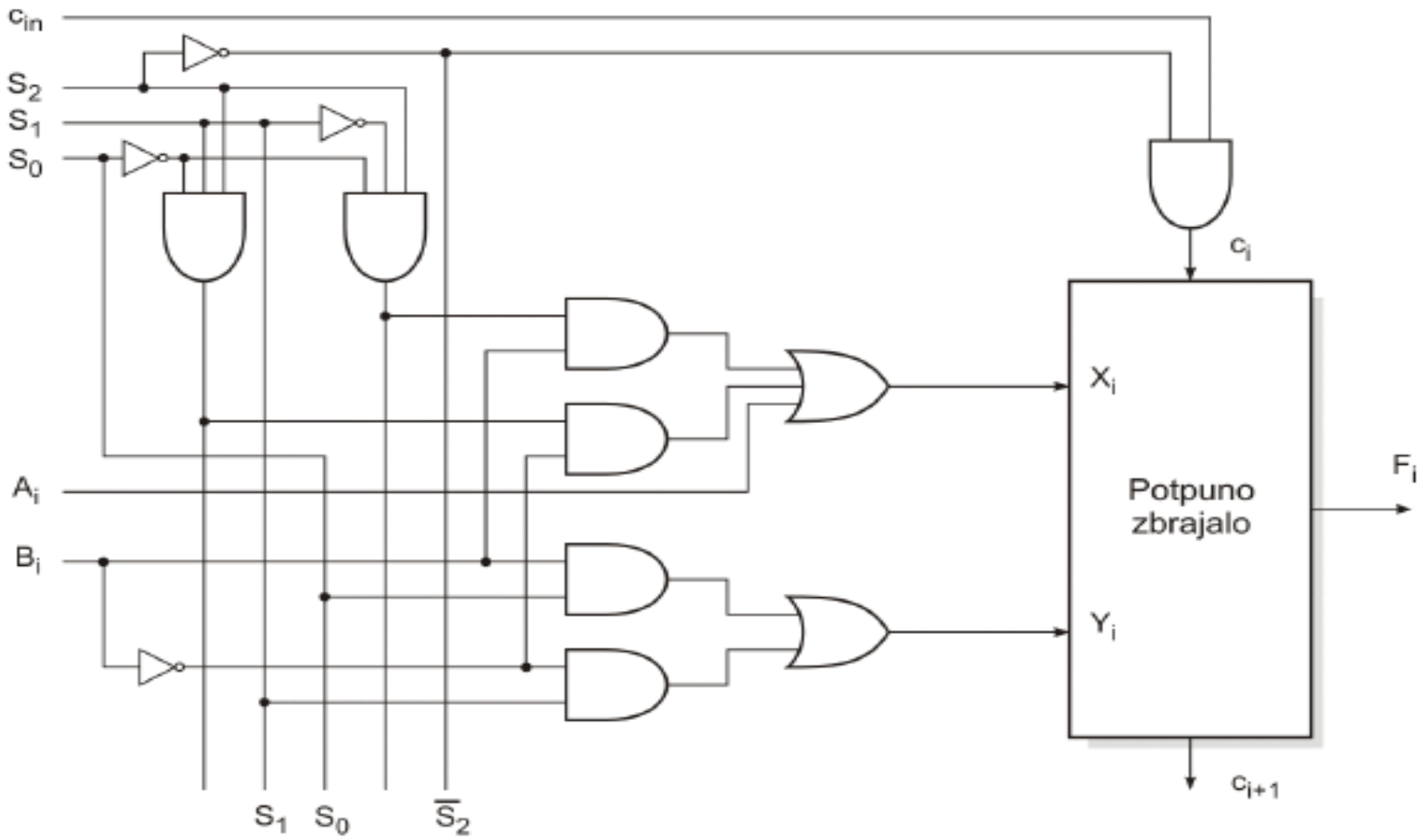
- Preinaka i-tog stupnja jednostavne ALU da bi se dobila logička operacija I



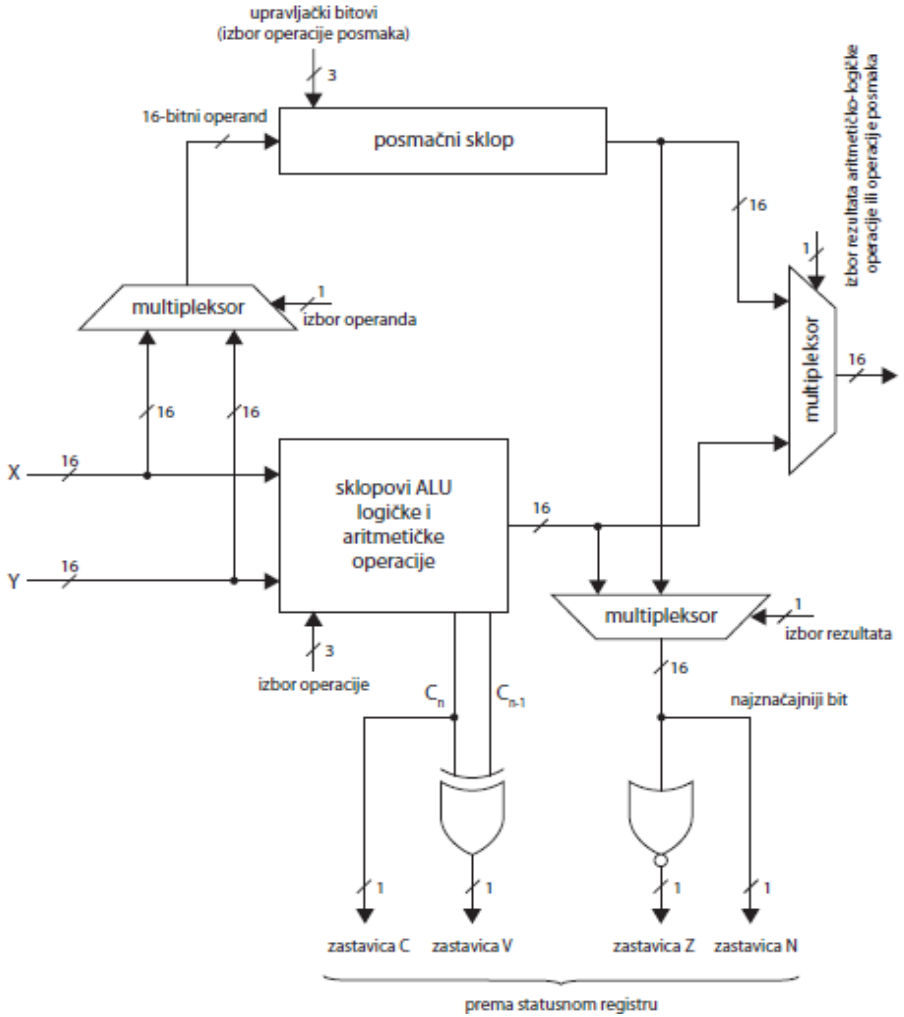
Stvarna izvedba ALU

- podjela stupnja ALU na aritmetičku i logičku sekciju samo je funkcionalna.
- prilikom oblikovanja ALU obično se upotrebljava sljedeći pristup:
  - najprije se oblikuje aritmetička sekcija
  - određuju se logičke operacije koje se mogu izvesti aritmetičkom sekcijom
  - modificiraju se aritmetički sklopovi da bi se mogle izvesti i logičke operacije

Končna izvedba ALU



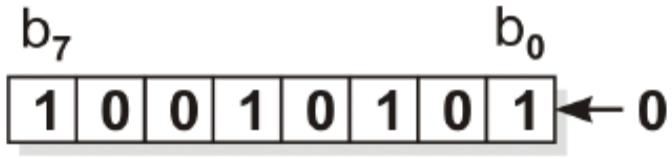
Jednostavna višefunkcijska ALU

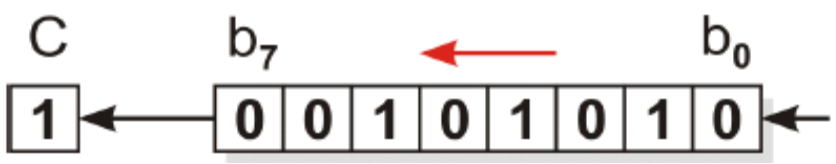


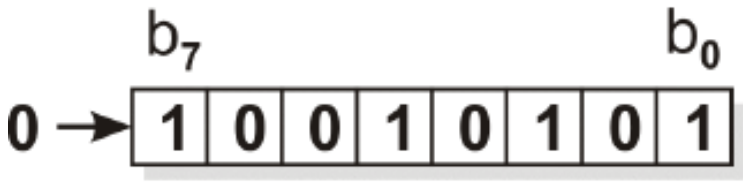
Sklop za posmak

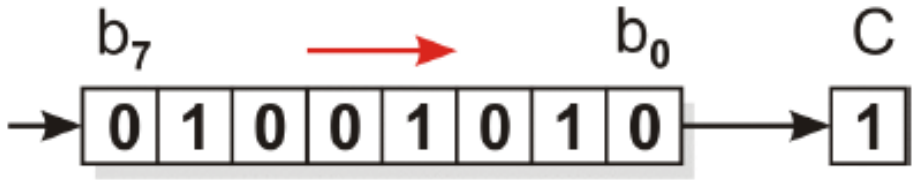
- Osnovna je sastavna komponenta ALU
- Smješta se na izlazu ALU i povezuje je sa sabirnicom
- Prenosi rezultat aritmetičke ili logičke operacije na sabirnicu:
  - izravno bez posmaka
  - s posmakom ulijevo ili udesno
- Sklopovi za posmak mogu se izvesti kao dvosmjerni posmačni registri ili kao kombinacijski sklopovi

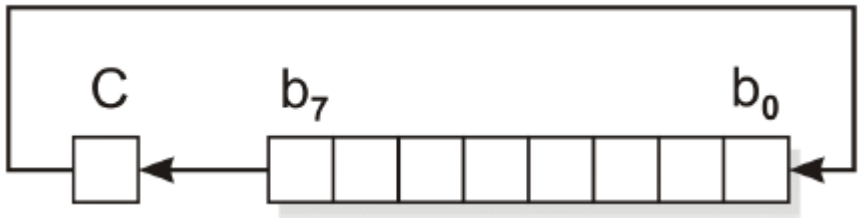
Sklopovi za posmak



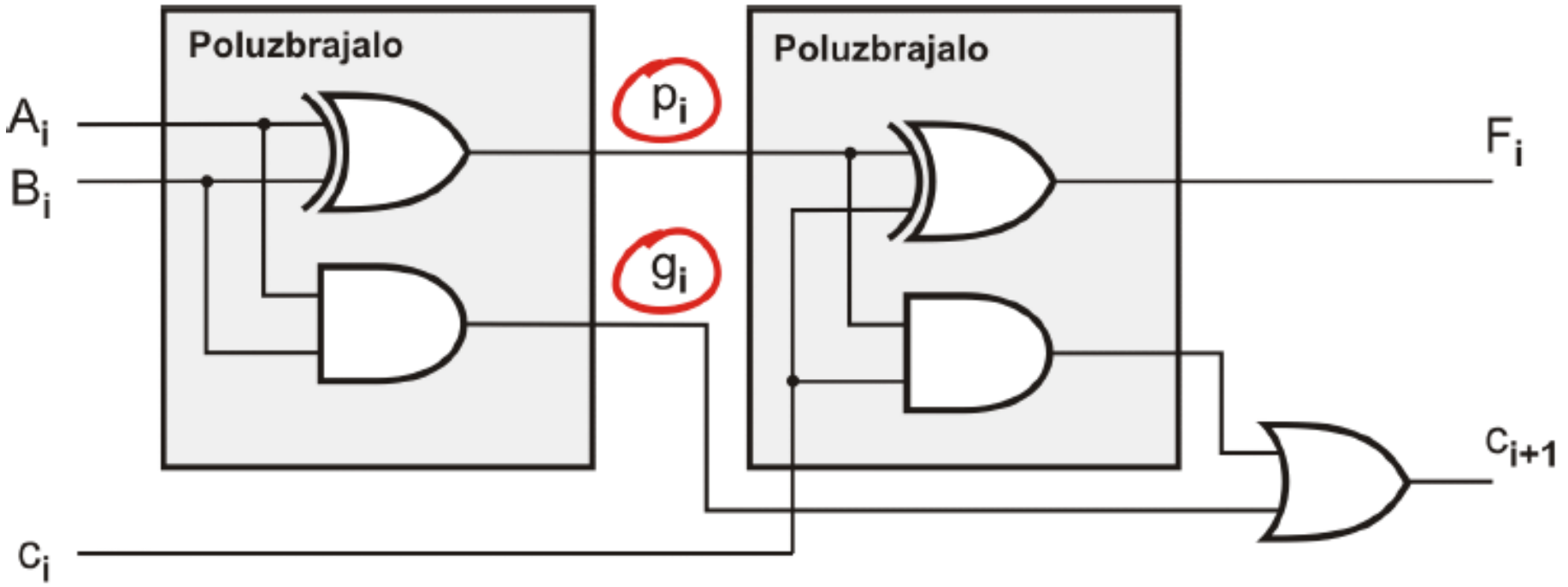








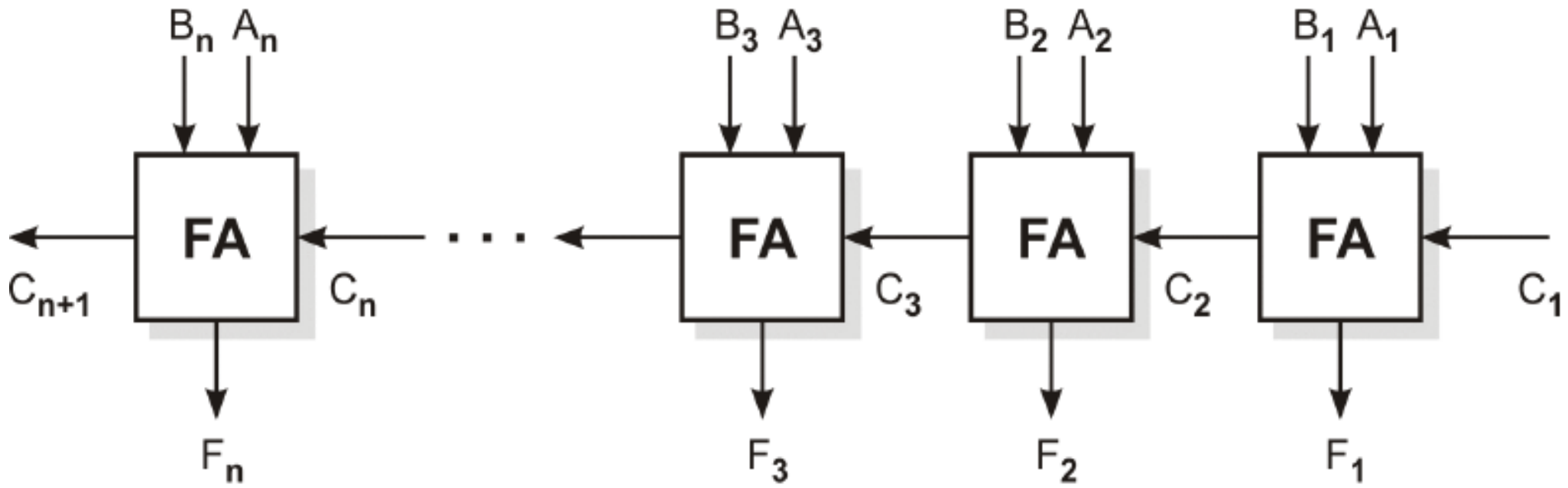
Sklop za predviđanje bita prijenosa



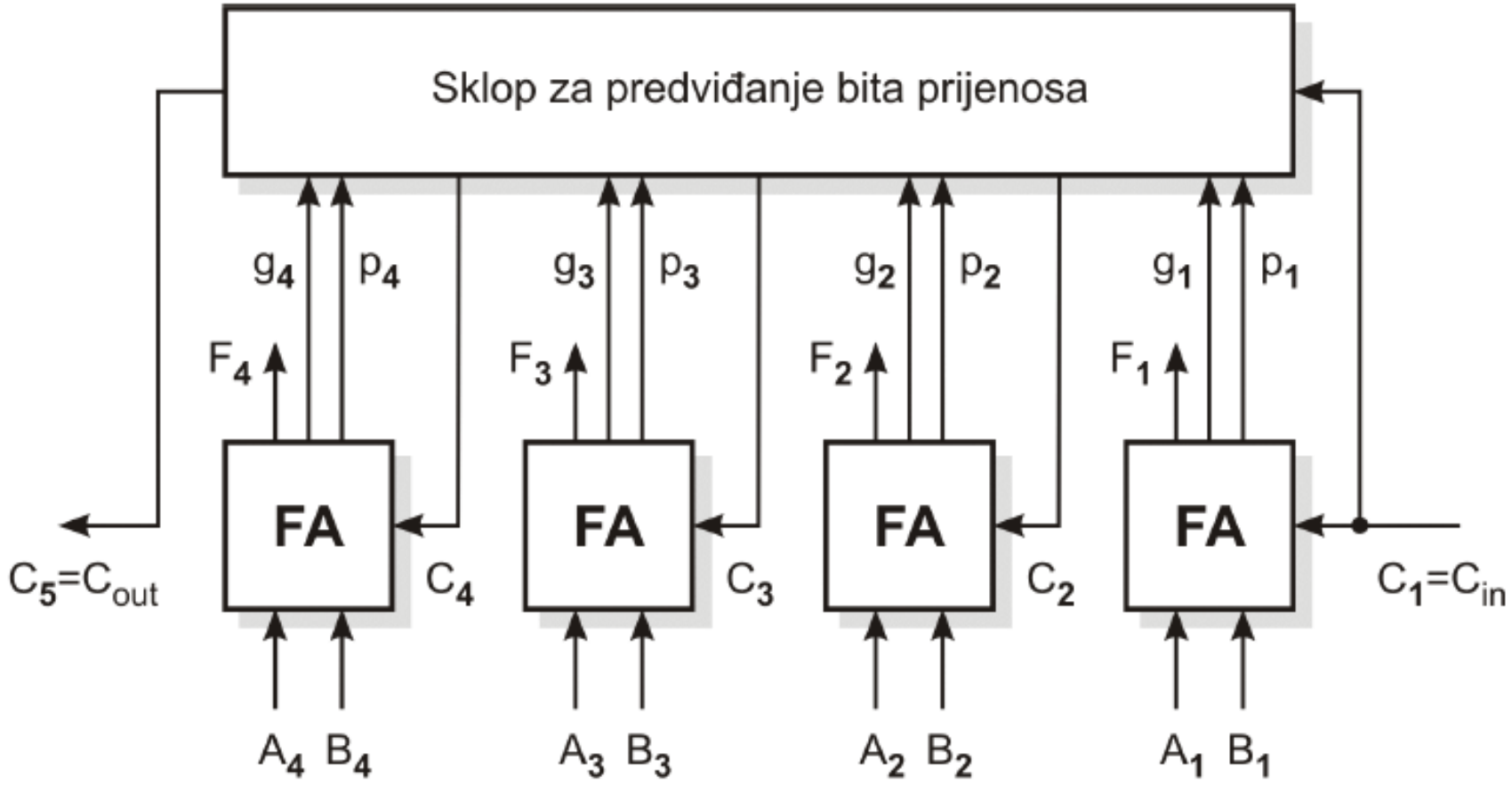
$g_i = A_i B_i$ signal generiranja bita prijenosa

$p_i = A_i \oplus B_i$ signal širenja (propagacije) bita prijenosa

Kašnjenje paralelnog n-bitnog zbrajala



Izvedba brzog zbrajala



Opći algoritam za množenje

				a_3	a_2	a_1	a_0
							x
				b_3	b_2	b_1	b_0
Parcijalni produkt 0				a_3b_0	a_2b_0	a_1b_0	a_0b_0
Parcijalni produkt 1			a_3b_1	a_2b_1	a_1b_1	a_0b_1	
Parcijalni produkt 2		a_3b_2	a_2b_2	a_1b_2	a_0b_2		
Parcijalni produkt 3	a_3b_3	a_2b_3	a_1b_3	a_0b_3			

Množenje

23_{10} x
19_{10}

---

2 0 7

2 3

---

4 3 7

1 0 1 1 1 x
1 0 0 1 1

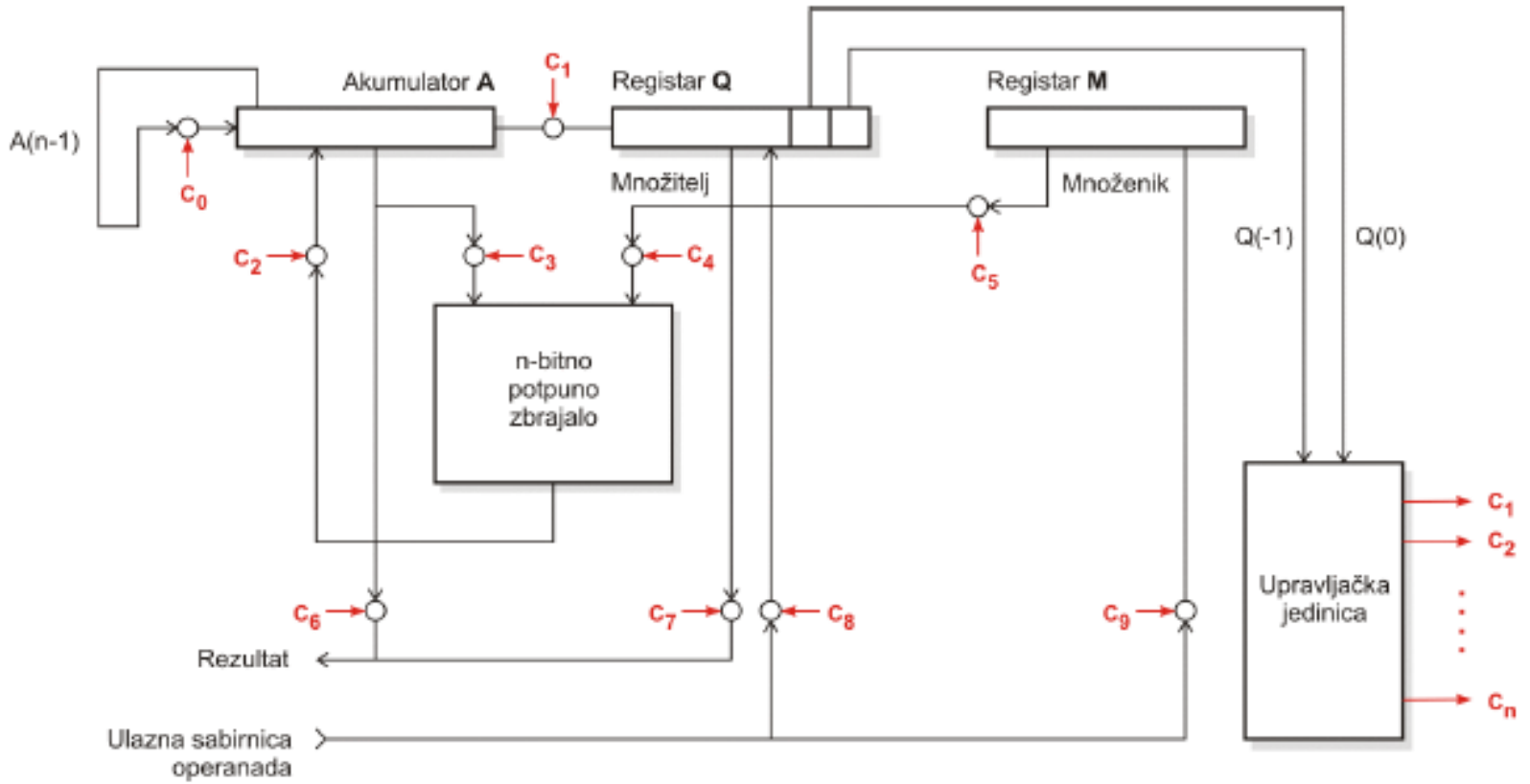
---

1 0 1 1 1
1 0 1 1 1
0 0 0 0 0
0 0 0 0 0
1 0 1 1 1

---

1 1 0 1 1 0 1 0 1

Blok shema množila



Hvala na pažnji!