Strujni krugovi u računalu

Miličić, Ivan

Sveučilište J. J. Strossmayera u Osijeku
Odjel za matematiku

Ivan Miličić

Strujni krugovi u računalu

Diplomski rad

Osijek, 2019.

Sveučilište J. J. Strossmayera u Osijeku
Odjel za matematiku

Ivan Miličić

Strujni krugovi u računalu

Diplomski rad

Mentor: izv. prof. dr. sc. Domagoj Matijević
Komentor: mag. math. Luka Borozan

Osijek, 2019.

Sadržaj

Uvod

U ovom radu predstavljena je osnovna podjela računalne logike na kombinacijsku i sekvencijalnu te je sve potkrijepljeno primjerima strujnih krugova u računalu. Koristit će se dva međusobno ekvivalentna načina prikaza strujnih krugova: grafički (u smislu grafa, gdje su logički sklopovi vrhovi, a veze među njima bridovi) i pomoću jezika HDL (eng. Hardware Description Language). Jednostavniju verziju ovog jezika, koji će biti korišten u ovom radu, razvili su profesori Noam Nisan i Shimon Shocken u sklopu kursa Nand2Tetris zajedno s alatima za pokretanje programskog kôda dostupnim na njihovoj web stranici [2].

Prvo poglavlje služi kao uvod: u njemu su dane definicije osnovnih pojmova i tvrdnje koje će kasnije biti korištene u ovom radu.

U drugom poglavlju uvodi se najprije osnovni element kombinacijske logike, logički sklop, pomoću kojeg se konstruiraju strujni krugovi kombinacijske logike. U ovom poglavlju se također uspostavlja veza strujnih krugova s Booleovim izrazima te se sve poopćuje Boolevim funkcijama. Poglavlje završava opisom implementacije i rada aritmetičko-logičke jedinice, dijela računala odgovornog za aritmetičke i logičke operacije.

U trećem poglavlju opisana je sekvencijalna logika koja omogućuje spremanje podataka u računalo. Njezin osnovni element je bistabil koji može spremiti jedan bit podatka. U ovom poglavlju je opisana i podjela sekvencijalne logike na sinkronu i asinkronu te su predstavljene prednosti i nedostaci jedne i druge. Na kraju poglavlja predstavljen je pregled RAM-a i ROM-a, osnovnih komponenti računala odgovornih za spremanje podataka.

Prva dva poglavlja te dio trećeg se većim dijelom oslanja na prvih sedam poglavlja knjige *System Architecture as an Ordinary Engineering Discipline* ([1]). U trećem poglavlju je najviše korištena knjiga *Sequential Logic: Analysis and Synthesis* ([3]) i drugo poglavlje knjige *Digital Logic and Microprocessor Design* ([5]).

Sav napisan HDL kôd dostupan je na GitHubu [4].

Poglavlje 1

Osnovni pojmovi

Uvedimo najprije neke oznake i pojmove. Interval cijelih brojeva između i i j, u oznaci $[i : j]$, je skup $\{i, i+1, i+2, \ldots, j-1, j\}$. Formalna definicija je

$$[i : i] = \{i\}$$
$$[i : j+1] = [i : j] \cup \{j+1\}.$$

Za uređenu n-torku $a = (a_1, a_2, \ldots, a_n)$ elemenata iz nekog skupa A koristit će se oznaka $a[1 : n]$ i zvat će se vektor. Specijalno, umjesto $a[i]$ ponekad će se pisati samo a_i. Formalno, a je funkcija $a\colon [1 : n] \to A$. Zbog toga, skup $\{a_1, a_2, \ldots, a_n\}$ ćemo označavati s $a([1 : n])$. Ponekad, najčešće kada vektori budu predstavljali binarne brojeve, koristit ćemo i oznaku $a[n-1 : 0]$.

Niz je funkcija $a\colon \mathbb{N} \to A$ čiji ćemo n-ti element označavati s a^n. Nadalje, vektor nizova označavat ćemo isto kao gore: $a[1 : n]$. Ako želimo od svakog niza u vektoru $a[1 : n]$ uzeti r-tu komponentu, to ćemo označiti s $a^r[1 : n]$ ili samo a^r. Neformalno, to je $a^r = a^r[1 : n] = (a_1^r, a_2^r, \ldots, a_n^r)$.

Konkatenaciju vektora $a = a[1 : n]$ i $b = b[1 : m]$ označavamo s $a \circ b$ ili ab i definiramo kao novi vektor $c = c[1 : n+m]$ pri čemu je

$$c_i = \begin{cases} a_i, & i \leq n \\ b_{i-n}, & i \geq n+1 \end{cases}.$$

Također, za vektor a i $n \in \mathbb{N}$ definiramo a^n kao vektor dobiven konkatenacijom a sa samim sobom n-puta, tj.

$$a^1 = a$$
$$a^{n+1} = a \circ a^n.$$

Funkciju $f\colon A^n \to B^m$ uvijek ćemo promatrati kao vektorsku funkciju. U tom smislu, f je zapravo vektor funkcija $f[1 : m]$ pa i-tu funkciju označavamo s $f[i]$ ili f_i. Nadalje, za $a \in A^n$ je $f(a) \in B^m$ vektor, a njegova i-ta komponenta je upravo $f_i(a)$.

Definicija 1. *Za cijele brojeve $a, b \in \mathbb{Z}$ i prirodan broj $k \in \mathbb{N}$ kažemo da su a i b kongruentni modulo k ukoliko k dijeli razliku brojeva a i b, tj.*

$$a \equiv b \pmod{k} \Leftrightarrow (\exists z \in \mathbb{Z})(a - b = z \cdot k).$$

Definicija 2. *Za $a, b \in \mathbb{Z}$ i $k \in \mathbb{N}$ definiramo cjelobrojni ostatak pri dijeljenju a s k, u oznaci $a \bmod k$, kao broj b takav da je*

$$a \equiv b \pmod{k} \wedge b \in [0 : k-1].$$

Definicija 3. *Za $a, b \in \mathbb{Z}$ i paran broj $k = 2k'$ za $k' \in \mathbb{N}$ definiramo $a \operatorname{tmod} k$ kao broj b takav da je*

$$a \equiv b \pmod{k} \wedge b \in \left[-\frac{k}{2} : \frac{k}{2} - 1\right].$$

Primjer 1. *Primjerice, za $k = 4$ će rezultat operacije* tmod *biti iz skupa $[-2 : 1] = \{-2, -1, 0, 1\}$:*

$$7 \bmod 4 = 3$$
$$7 \operatorname{tmod} 4 = -1.$$

Operacija tmod *koristit će se kasnije kod dvostrukog komplementa.*

Lema 1. *Neka je $k \in \mathbb{Z}$ i $x \equiv y \pmod{k}$.*

1. *Ako je $x \in [0 : k-1]$, onda je $x = y \bmod k$.*
2. *Neka je dodatno k paran broj. Ako je $x \in [-k/2 : k/2 - 1]$, onda je $x = y \operatorname{tmod} k$.*

1.1 Binarni brojevi

Budući da se bavimo binarnim znamenkama, koristit ćemo oznaku $\mathbb{B}$ za skup $\{0, 1\}$.

Definicija 4. *Neka je $a = a[n-1 : 0] \in \mathbb{B}^n$ bitovni vektor. Interpretaciju vektora a kao binarni broj označavamo s $\langle a \rangle$ i definiramo kao*

$$\langle a \rangle = \sum_{i=0}^{n-1} a_i \cdot 2^i.$$

Vektor a zovemo binarni zapis prirodnog broja $\langle a \rangle$.

Primjer 2. *Primjerice, za $100, 111 \in \mathbb{B}^3$ je*

$$\langle 100 \rangle = 4$$
$$\langle 111 \rangle = 7.$$

Napomena 1. *Uočimo da za $1^n \in \mathbb{B}^n$ vrijedi*

$$\langle 1^n \rangle = \sum_{i=0}^{n-1} 2^i = 2^n - 1.$$

Dobiveni broj odgovara najvećem binarnom broju koji se može zapisati s n bitova.

Definicija 5. *Neka je $n \in \mathbb{N}$. Definiramo $B_n \subset \mathbb{N}$ kao podskup skupa svih prirodnih brojeva čiji se binarni zapis sastoji od n bitova:*

$$B_n = \{\langle a \rangle : a \in \mathbb{B}^n\}.$$

Napomena 2. *Lako se pokaže da je* $B_n = [0 : 2^n - 1]$.

Definicija 6. *Za cijeli broj* $x \in B_n$ *definiramo binarni zapis broja* x *duljine* n*, u oznaci* $bin_n(x)$ *ili* x_n*, kao binarni vektor* $a \in \mathbb{B}^n$ *takav da je* $\langle a \rangle = x$.

U nastavku je dano nekoliko lema koje će se kasnije koristiti u dokazima.

Lema 2. *Neka je* $a \in \mathbb{B}^n$ *i* $m \leq n$*. Tada vrijedi*

$$\langle a[n-1:0] \rangle = \langle a[n-1:m] \rangle \cdot 2^m + \langle a[m-1:0] \rangle$$

Dokaz.

$$\begin{aligned}
\langle a[n-1:0] \rangle &= \sum_{i=m}^{n-1} a_i \cdot 2^i + \sum_{i=0}^{m-1} a_i \cdot 2^i \\
&= \sum_{j=0}^{n-1-m} a_{m+j} \cdot 2^{m+j} + \langle a[m-1:0] \rangle \\
&= 2^m \cdot \sum_{j=0}^{n-1-m} a_{m+j} \cdot 2^j + \langle a[m-1:0] \rangle \\
&= 2^m \cdot \langle a[n-1:m] \rangle + \langle a[m-1:0] \rangle
\end{aligned}$$

□

Lema 3. *Za* $a \in \mathbb{B}^n$ *i* $m \leq n$ *vrijedi*

$$\langle a[m-1:0] \rangle = \langle a[n-1:0] \rangle \bmod 2^m.$$

Dokaz. Iz prethodne leme očito je

$$\langle a[n-1:0] \rangle \equiv \langle a[m-1:0] \rangle \pmod{2^m}.$$

Kako je $\langle a[m-1:0] \rangle \in B_m$, prema Lemi 1 slijedi tvrdnja. □

1.2 Dvostruki komplement

Definicija 7. *Neka je* $a = a[n-1:0] \in \mathbb{B}^n$ *bitovni vektor. Interpretaciju vektora* a *kao dvostruki komplement označavamo s* $[a]$ *i definiramo kao*

$$[a] = -a_{n-1} \cdot 2^{n-1} + \langle a[n-2:0] \rangle.$$

Vektor a *zovemo zapis dvostrukog komplementa cijelog broja* $[a]$.

Primjer 3. *Primjerice, za* $1010, 0100 \in \mathbb{B}^4$ *je*

$$\begin{aligned}
[1010] &= -1 \cdot 2^3 + \langle 010 \rangle = -8 + 2 = -6 \\
[0100] &= -0 \cdot 2^3 + \langle 100 \rangle = 0 + 4 = 4
\end{aligned}$$

Definicija 8. *Neka je $n \in \mathbb{N}$. Definiramo T_n kao skup svih prirodnih brojeva čiji se zapis dvostrukog komplementa sastoji od n bitova:*

$$T_n = \{[a] \colon a \in \mathbb{B}^n\}.$$

Napomena 3. *Lako se pokaže da je $T_n = [-2^{n-1} : 2^{n-1} - 1]$.*

Definicija 9. *Za cijeli broj $x \in T_n$ definiramo zapis dvostrukog komplementa duljine n broja x, u oznaci $twoc_n(x)$, kao binarni broj $a \in \mathbb{B}^n$ takav da je $[a] = x$.*

Lema 4. *Neka je $a \in \mathbb{B}^n$. Tada vrijedi*

$$\begin{aligned}
[0a] &= \langle a \rangle \\
[a] &\equiv \langle a \rangle \pmod{2^n} \\
[a] < 0 &\Leftrightarrow a_{n-1} = 1 \\
[a_{n-1}a] &= [a] \\
-[a] &= [\bar{a}] + 1.
\end{aligned}$$

Dokaz. Prva tvrdnja se jednostavno pokaže:

$$[0a] = 0 \cdot 2^n + \langle a[n-1:0] \rangle = \langle a \rangle.$$

Druga tvrdnja slijedi iz

$$\begin{aligned}
[a] - \langle a \rangle &= -a_{n-1} \cdot 2^{n-1} + \langle a[n-2:0] \rangle - (a_{n-1} \cdot 2^{n-1} + \langle a[n-2:0] \rangle) \\
&= -a_{n-1} \cdot 2^n.
\end{aligned}$$

Treću tvrdnju dokazujemo u dva smjera. Pretpostavimo da je $[a] < 0$. Tada je

$$\langle a[n-2:0] \rangle < a_{n-1} \cdot 2^{n-1}.$$

Kada bi a_{n-1} bio jednak 0, tada bismo dobili kontradikciju jer je $\langle a[n-2:0] \rangle \geq 0$. Stoga, $a_{n-1} = 1$. Obratno, ako je $a_{n-1} = 1$, onda imamo

$$\begin{aligned}
[a] &= -2^{n-1} + \langle a[n-2:0] \rangle \\
&\leq -2^{n-1} + 2^{n-1} - 1 && \text{Napomena 2} \\
&= -1
\end{aligned}$$

pa je $[a] < 0$. Četvrta tvrdnja pokazuje se na sljedeći način:

$$\begin{aligned}
[a_{n-1}a] &= -a_{n-1} \cdot 2^n + \langle a[n-1:0] \rangle \\
&= -a_{n-1} \cdot 2^n + a_{n-1} \cdot 2^{n-1} + \langle a[n-2:0] \rangle \\
&= -a_{n-1} \cdot 2^{n-1} + \langle a[n-2:0] \rangle \\
&= [a].
\end{aligned}$$

U petoj tvrdnji koristit ćemo $\bar{x} = 1 - x$ za $x \in \mathbb{B}$.

$$\begin{aligned}
[\bar{a}] &= -\bar{a}_{n-1} \cdot 2^{n-1} + \sum_{i=0}^{n-2} \bar{a}_i \cdot 2^i \\
&= -(1 - a_{n-1}) \cdot 2^{n-1} + \sum_{i=0}^{n-2} (1 - a_i) \cdot 2^i \\
&= -2^{n-1} + \sum_{i=0}^{n-2} 2^i + a_{n-1} \cdot 2^{n-1} - \sum_{i=0}^{n-2} a_i \cdot 2^i \\
&= -2^{n-1} + (2^{n-1} - 1) - \left(-a_{n-1} \cdot 2^{n-1} + \sum_{i=0}^{n-2} a_i \cdot 2^i \right) \\
&= -1 - [a].
\end{aligned}$$

□

1.3 Booleovi izrazi

Za Booleov izraz e koji ovisi o varijablama $x = x[1 : n]$ pišemo $e(x)$. Varijable x_i mogu poprimiti vrijednosti iz $\mathbb{B}$. Vrijednost Booleovog izraza e za vektor $a \in \mathbb{B}^n$ označavamo s $e(a) \in \mathbb{B}$ i dobiva se supstitucijom varijabli $x_i = a_i$. Nadalje, za dva Booleova izraza e i f kažemo da su ekvivalentni, u oznaci $e \equiv f$, ukoliko se njihove tablice istinitosti podudaraju, tj.

$$e \equiv f \Leftrightarrow (\forall a \in \mathbb{B}^n)(e(a) = f(a)).$$

Lema 5. *Za dane Booleove izraze $e(x)$ i $e'(x)$ vrijedi*

$$e \equiv e' \Leftrightarrow (\forall a \in \mathbb{B}^n)(e = 1 \Leftrightarrow e' = 1).$$

Neka je e Booleov izraz i $a \in \mathbb{B}$. Definiramo

$$e^a = \begin{cases} e, & a = 1 \\ \bar{e}, & a = 0 \end{cases}.$$

Lako se vidi da je $e^a \equiv e \Leftrightarrow a$. U sljedećoj lemi dane su tri Booleove jednadžbe i njihova rješenja.

Lema 6. *Neka su $e(x)$ i $e_i(x)$, $i \in [1 : n]$ Booleovi izrazi i $a \in \mathbb{B}$. Tada vrijedi*

1. $e^a = 1 \Leftrightarrow e = a$,
2. $(\bigwedge_{i=1}^n e_i) = 1 \Leftrightarrow (\forall i \in [1 : n])(e_i = 1)$,
3. $(\bigvee_{i=1}^n e_i) = 1 \Leftrightarrow (\exists i \in [1 : n])(e_i = 1)$

Poglavlje 2

Kombinacijska logika

2.1 Strujni krug

U nastavku ćemo definirati kombinacijsku logiku. Pojam kombinacijske logike odnosi se na logičke strujne krugove kojima izlazne vrijednosti ovise isključivo o trenutnim ulaznim vrijednostima. Osnovni element takvog strujnog kruga je logički sklop.

Definiramo skup $\mathcal{B}_{n,m}$ kao skup svih funkcija $f\colon \mathbb{B}^n \to \mathbb{B}^m$. Specijalno, $\mathcal{B}_{n,1} = \mathcal{B}_n$. Definirajmo i projekciju $\pi_i \in \mathcal{B}_n$ s $\pi_i(x) = x_i$.

Definicija 10. *Neka je $n \in \mathbb{N}$ i $C = \mathbb{B} \cup \pi([1:n])$. Pojam logičkog sklopa definiramo induktivno:*

1. *uređeni parovi $(\neg, \mathbf{p}^{(1)})$, $(\wedge, \mathbf{p}^{(2)})$, $(\vee, \mathbf{p}^{(3)})$ i $(\veebar, \mathbf{p}^{(4)})$ su logički sklopovi, gdje je $\mathbf{p}^{(1)} \in C$ i $\mathbf{p}^{(2)}, \mathbf{p}^{(3)}, \mathbf{p}^{(4)} \in C^2$,*
2. *ako su $g_1 = (\omega_1, \mathbf{p}^{(1)})$ i $g_2 = (\omega_2, \mathbf{p}^{(2)})$ logički sklopovi, tada su i $(\neg\omega_1, g_1)$, $(\omega_1 \circ \omega_2, (g_1, g_2))$, za $\circ \in \{\wedge, \vee, \veebar\}$, također logički sklopovi,*
3. *svaki logički sklop može se dobiti primjenom prethodna dva pravila.*

Vektor $\mathbf{p}$ iz prethodne definicije zovemo vektor prethodnika logičkog sklopa g.

Primjer 4. *Neka je $g_1 = (\neg, \pi_1)$ i $g_2 = (\neg, \pi_2)$, $\pi_1, \pi_2 \in \mathcal{B}_2$. Očito su g_1 i g_2 logički sklopovi. Tada je i $g = (\wedge, (g_1, g_2))$ također logički sklop.*

Sada možemo definirati strujni krug.

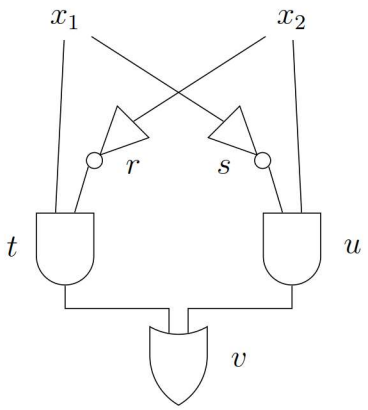
Definicija 11. *Logički strujni krug C je uređena trojka $(H, \mathbf{x}, \mathbf{y})$ pri čemu je*

- $\mathbf{x} = \mathbf{x}[1:n] \in \mathbb{B}^n$ *vektor ulaznih podataka,*
- $H = g([1:b])$, $b \in \mathbb{N}$, *skup logičkih sklopova pri čemu je logički sklop $g_i = (\omega_i, \mathbf{p}^{(i)})$, $\omega_i \in \mathcal{B}_{n(i)}$, gdje je $\mathbf{p}^{(i)} \in (\mathbb{B} \cup \pi([1:n]) \cup g([1:i-1]))^{n(i)}$,*
- $\mathbf{y} = \mathbf{y}[1:m] \in (\mathbb{B} \cup \pi([1:n]) \cup H)^m$ *vektor izlaznih podataka.*

Napomena 4.

- *Skup $\mathbb{B} \cup \pi([1 : n]) \cup H$ iz prethodne definicije označavat ćemo sa $Sig(C)$ i zvati ga signali strujnog kruga C.*
- *Budući da radimo sa podskupovima skupova Booleovih funkcija $\mathcal{B}_1$ i $\mathcal{B}_2$, vektor $p^{(i)}$ će uvijek biti jednočlan ili uređeni par.*

Primjer 5. *Na Slici 2.1 dan je logički strujni krug. Formalno ga definirajmo.*



Slika 2.1: Primjer logičkog strujnog kruga.

Označimo taj logički strujni krug s L. Tada je $L = (H, \mathbf{x}, \mathbf{y})$ gdje je

- $\mathbf{x} = (x_1, x_2)$,
- $H = \{r, s, t, u, v\}$,

$$\begin{aligned} r &= (\neg, \pi_2) & s &= (\neg, \pi_1) \\ t &= (\wedge, (r, \pi_1)) & u &= (\wedge, (\pi_2, s)) \\ v &= (\vee, (u, t)), \end{aligned}$$

- $\mathbf{y} = v$.

Uočavamo da je vrlo jednostavno odrediti kako će strujni krug raditi, tj. kako ćemo odrediti izlaznu vrijednost strujnog kruga za dane ulaze. Ipak, slijedi definicija koja to formalno opisuje.

Definicija 12. *Neka je C strujni krug i $s \in Sig(C)$. Signalu s pridružujemo Booleov izraz $s(\mathbf{x})$ definiran na sljedeći način:*

1. *ako je $s \in \mathbb{B}$ konstanta:*

$$s(\mathbf{x}) = \begin{cases} 0, & s = 0 \\ 1, & s = 1 \end{cases},$$

2. *ako je $s = \pi_i$ projekcija:*

$$s(\mathbf{x}) = x_i,$$

3. *ako je* $s = g_i = (\omega_i, \mathbf{p}^{(i)})$ *logički sklop:*

$$s(\mathbf{x}) = \omega_i(p^{(i)}[1 : n(i)](\mathbf{x})).$$

Napomena 5.

1. *U skladu s prethodnom napomenom,* $\mathbf{p}^{(i)}$ *je jednočlan ili uređeni par pa je*

$$s(\mathbf{x}) = \neg p^{(i)}(\mathbf{x}) \quad \textit{ili} \quad s(\mathbf{x}) = p_1^{(i)}(\mathbf{x}) \circ p_2^{(i)}(\mathbf{x})$$

   za $\circ \in \{\wedge, \vee, \veebar\}$.

2. *Vektor izlaznih podataka* $\mathbf{y}$ *sastoji se od signala pa prethodna definicija vrijedi i u ovom slučaju.*

Želimo pokazati da je prethodna definicija dobra. U tu svrhu uvodimo pojam grafa $G(C) = (V, E)$ strujnog kruga C definiranog na sljedeći način:

- čvorovi grafa su signali, tj. $V = Sig(C)$,
- između čvorova $s, t \in V$ povlačimo brid točno onda kada je t logički sklop, a s je njegov ulazni podatak, tj.

$$(s, t) \in E \Leftrightarrow (\exists i \in [1 : b])(t = g_i \wedge (\exists j \in [1 : n(i)])(s = p_j^{(i)})).$$

Lema 7. *Neka je* C *strujni krug. Tada je pripadni graf* $G(C)$ *acikličan.*

Dokaz. Pretpostavimo suprotno, tj. da postoji ciklus c i neka su $g^{(c)}[1 : t]$ redom elementi ciklusa. Svaki $g_i^{(c)} \in H$ je element vektora g. Stoga, označimo njegovu poziciju u vektoru g s n_i.

Budući da je $(g_1^{(c)}, g_2^{(c)})$ brid u ciklusu, po konstrukciji grafa $G(C)$ je $g_1^{(c)}$ ulazni signal za logički sklop $g_2^{(c)}$ pa je po definiciji strujnog kruga njegova pozicija n_1 u vektoru g manja od pozicije n_2. Induktivno zaključujemo da je $n_i < n_j$ za sve $i < j$ pa specijalno vrijedi i $n_1 < n_t$. No, budući da postoji i brid $(g_t^{(c)}, g_1^{(c)})$, vrijedi $n_t < n_1$. Sada imamo $n_1 < n_t < n_1$ što je kontradikcija. Dakle, $G(C)$ je acikličan. □

Sada možemo dokazati propoziciju o dobroj definiranosti Booleovog izraza signala s.

Propozicija 1. *Booleovi izrazi* $s(\mathbf{x})$ *su dobro definirani za sve* $s \in Sig(C)$ *strujnog kruga* C.

Dokaz. Budući da je graf $G(C)$ aciklički, svaki čvor ima dubinu pa tvrdnju možemo dokazati koristeći metodu matematičke indukcije po dubini signala s u pripadnom grafu $G(C)$.

- Baza: Za $n = 0$ je s ulaz ili konstanta pa je $s(\mathbf{x})$ po Definiciji 12 dobro definiran.
- Pretpostavka: Neka tvrdnja vrijedi i za signale na dubini $n \leq k$.
- Korak: Dokažimo da je i za signal $s = (\omega, p[1 : t])$ na dubini $n = k + 1$ Booleov izraz $s(\mathbf{x})$ dobro definiran. Naime, signali vektora $\mathbf{p}$ su ulazni signali od s pa su na dubini manjoj od n. Po pretpostavci Booleovi izrazi $p_i(\mathbf{x})$ su dobro definirani za sve $i \in [1 : t]$. Stoga, Booleov izraz $s(\mathbf{x}) = \omega(p[1 : t](\mathbf{x}))$ se dobije direktnom primjenom Definicije 12 pa je dobro definiran.

□

Definicija 13. *Strujnom krugu C pridružujemo vektor Booleovih izraza $C(\mathbf{x})$ definiran s $C(\mathbf{x}) = \mathbf{y}(\mathbf{x})$.*

Primjer 6. *Zapišimo strujni krug iz Primjera 5 kao Booleov izraz.*

Rješenje. *Treba odrediti $\mathbf{y}(\mathbf{x})$. Budući da je $\mathbf{y} = v$, kao rezultat dobit ćemo Booleov izraz.*

$$\begin{aligned}\mathbf{y}(\mathbf{x}) = v(\mathbf{x}) &= t(\mathbf{x}) \vee u(\mathbf{x}) = (\pi_1(\mathbf{x}) \wedge r(\mathbf{x})) \vee (s(\mathbf{x}) \wedge \pi_2(\mathbf{x})) \\ &= (x_1 \wedge \neg\pi_2(\mathbf{x})) \vee (\neg\pi_1(\mathbf{x}) \wedge x_2) \\ &= (x_1 \wedge \neg x_2) \vee (\neg x_1 \wedge x_2).\end{aligned}$$

U prethodnom primjeru smo vidjeli da se strujni krug može pretvoriti u odgovarajući Booleov izraz. To vrijedi i općenito prema Propoziciji 1. Sljedeći teorem pokazuje da vrijedi i obratno te daje način na koji se iz Booleovog izraza može konstruirati strujni krug.

Teorem 1. *Neka je $e(x)$, $x = x[1:m]$ Booleov izraz. Tada postoji strujni krug C s ulaznim signalima $\mathbf{x} = \mathbf{x}[1:m]$ i izlaznim signalom $y \in Sig(C)$ takav da je*

$$e = y(\mathbf{x}).$$

Dokaz. Dokažimo tvrdnju indukcijom po strukturi Booleovog izraza, tj. po broju konstanti, varijabli i operatora.

- Baza: Za $n = 1$ je e varijabla x_1 ili konstanta $b \in \mathbb{B}$. Bez smanjenja općenitosti pretpostavimo da je $e = x_1$. Definirajmo strujni krug C kao $C = (H, x_1, x_1)$. Doista, za $\mathbf{x} \in \mathbb{B}$ je

$$y(\mathbf{x}) = \pi_1(\mathbf{x}) = x_1 = e.$$

- Pretpostavka: Pretpostavimo da tvrdnja vrijedi za Booleov izraz veličine $n \leq k$.

- Korak: Dokažimo tvrdnju za $n = k + 1$. Imamo dva slučaja:

  - Ako je $e = \neg e'$, tada po pretpostavci indukcije, budući da je Booleov izraz e' veličine k, postoji strujni krug $C' = (H', \mathbf{x}, y')$ takav da je

$$e' = y'(\mathbf{x}).$$

    Definirajmo strujni krug $C = (H, \mathbf{x}, y)$ na sljedeći način:

    * $g = (\neg, y')$,
    * $H = H' \cup \{g\}$,
    * $y = g$.

    Uočimo da je

$$y(\mathbf{x}) = g(\mathbf{x}) = \neg y'(\mathbf{x}) = \neg e' = e.$$

– Ako je $e = e' \circ e''$ za $\circ \in \{\wedge, \vee, \veebar\}$, onda, slično kao gore, postoje strujni krugovi C' i C'' sa izlaznim signalima y' i y'' takvi da vrijedi

$$e' = y'(\mathbf{x}) \quad \text{i} \quad e'' = y''(\mathbf{x}).$$

U tom slučaju lako definiramo strujni krug C:

$$y(\mathbf{x}) = y'(\mathbf{x}) \circ y''(\mathbf{x}) = e' \circ e'' = e.$$

□

Ovim je pokazano da su Booleovi izrazi i strujni krugovi sa jednim izlazom ekvivalentni. Može se pokazati i općenitije: da strujnom krugu od m izlaza odgovara vektor Booleovih izraza duljine m.

U Primjeru 6 je dobiven Booleov izraz koji odgovara danom strujnom krugu. No, lako se uoči da se taj Booleov izraz može zapisati na ekvivalentne načine, primjerice,

$$\mathbf{y}(\mathbf{x}) \equiv x_1 \veebar x_2.$$

Iako su ova dva Booleova izraza ekvivalentna, ipak su drugačiji. Očito je da su i pripadni strujni krugovi različiti: strujni krug pridružen ovom ekvivalentnom zapisu je puno jednostavniji. Stoga, želimo doći do nekog općenitijeg pojma koji će biti jedinstven, a koji će opisivati oba Booleova izraza. Odgovor na to daje sljedeći teorem.

Teorem 2. *Za svaku funkciju $f \in \mathcal{B}_n$ postoji Booleov izraz e tako da vrijedi*

$$f(x) \equiv e.$$

Dokaz. Neka je $a = a[1:n] \in \mathbb{B}^n$ i $x = x[1:n]$ vektor varijabli. Definiramo monom kao

$$m(a) = \bigwedge_{i=1}^{n} x_i^{a_i},$$

pri čemu izraz $x_i^{a_i}$ zovemo literal. Tada,

$$\begin{aligned} m(a) = 1 &\Leftrightarrow (\forall i \in [1:n])(x_i^{a_i} = 1) && \text{Lema 6, tvrdnja 2} \\ &\Leftrightarrow (\forall i \in [1:n])(x_i = a_i) && \text{Lema 6, tvrdnja 1} \\ &\Leftrightarrow x = a. \end{aligned}$$

Stoga, imamo

$$m(a) = 1 \Leftrightarrow x = a. \tag{2.1}$$

Nadalje, definiramo $S(f)$ kao skup svih onih argumenata a za koje f poprima vrijednost 1:

$$S(f) = \{a \in \mathbb{B}^n \colon f(a) = 1\}.$$

Ukoliko je $S(f) = \emptyset$, onda je $f(x) = 0$ te je $e = 0$ pa smo gotovi. Inače stavimo

$$e = \bigvee_{a \in S(f)} m(a).$$

Tada

$$\begin{aligned} e = 1 &\Leftrightarrow (\exists a \in S(f))(m(a) = 1) && \text{Lema 6, tvrdnja 3} \\ &\Leftrightarrow (\exists a \in S(f))(a = x) && \text{Jednadžba (2.1)} \\ &\Leftrightarrow x \in S(f) \\ &\Leftrightarrow f(x) = 1. \end{aligned}$$

Stoga, primjenom Propozicije 5 zaključujemo da je $f(x) \equiv e$. □

Booleov izraz e konstruiran u prethodnom teoremu zove se disjunktivna normalna forma (DNF) funkcije f. Jasno je da postoje i drugi Booleovi izrazi ekvivalentni Booleovoj funkciji. Zbog toga ćemo pojmove vezane uz strujne krugove definirati preko funkcija.

2.2 HDL

Uz standardni grafički prikaz strujnih krugova pomoću simbola za logičke sklopove i odgovarajućih bridova između njih, postoji i posebni jezik, tzv. HDL (eng. Hardware Description Language). To je jezik koji služi za opisivanje strukture i ponašanja strujnih krugova. Ovaj jezik omogućuje jednostavnu provjeru rada strujnog kruga prije nego se krene sa njegovom konkretnom izradom, a također pomaže u optimizaciji brzine strujnog kruga, potrošnje energije i ukupne cijene.

Osnovni elementi HDL-a su jednostavni čipovi (And, Or, Not, Xor) koji implementiraju logiku odgovarajućih logičkih sklopova. U HDL-u svaki od ovih osnovnih čipova prima i vraća podatke preko argumenata koji se uvijek moraju navesti prilikom korištenja čipa. Budući da svi gore navedeni logički sklopovi vraćaju jedan bit, naziv izlaznog argumenta je out. Nazivi ulaznih argumenata ovise o broju argumenata pojedinog čipa pa je za čip *Not* naziv ulaznog argumenta in, dok su argumenti za ostale čipove a i b.

Vlastiti čipovi koje implementiramo također se sastoje od ulaznih i izlaznih argumenata i možemo ih imenovati po volji. Definicija vlastitog čipa sastoji se od zaglavlja i implementacije. U zaglavlju se pomoću ključne riječi CHIP deklarira čip (navodi se njegovo ime), a s ključnim riječima IN i OUT se definiraju njegovi ulazni i izlazni argumenti. U implementaciji se navode drugi čipovi i međusobno se povezuju njihove ulazne i izlazne vrijednosti.

Primjer 7. *Zapis strujnog kruga iz primjera 5 dan je ispod uz malo drugačije oznake.*

```
1  CHIP Xor {
2      IN a, b;
3      OUT out;
4
5      PARTS:
6      // na = ¬a
7      // nb = ¬b
8      Not(in = a, out = na);
9      Not(in = b, out = nb);
10
11     // ta = a ∧ nb
12     // tb = b ∧ na
```

```
13      And(a = a, b = nb, out = ta);
14      And(a = b, b = na, out = tb);
15
16      // out = ta ∨ tb
17      Or(a = ta, b = tb, out = out);
18  }
19
```

Program 2.1: Primjer strujnog kruga zapisanog u HDL-u.

Naziv čipa je Xor, njegove ulazni argumenti su a i b, dok je izlazni argument out. U prve dvije linije implementacije koristi se logički sklop Not *za izračun negacije vrijednosti bitova a i b i sprema u privremene varijable na i nb. Zatim se te varijable, zajedno s ulaznim a i b, koriste u sljedeća dva* And *logička sklopa koji daju novi par privremenih varijabli ta i tb koje konačno u posljednjem* Or *sklopu daju konačnu vrijednost out.*

Ovako zapisan kôd iz prethodnog primjera može se pokrenuti i testirati programom *Hardware-Simulator.*

2.3 Neki osnovni strujni krugovi

Neki od osnovnih strujnih krugova su:

- multiplekser,
- ispitivač nule,
- dekoder,
- zbrajalo.

U nastavku su ukratko opisani. Kasnije će se ovi strujni krugovi koristiti u izradi složenijih, kao što je primjerice ALU.

2.3.1 Multiplekser

Definicija 14. *Multiplekser je funkcija $mux \in \mathcal{B}_3$ definirana s*

$$mux(x, y, s) = \begin{cases} x, & s = 0 \\ y, & s = 1 \end{cases}.$$

Na temelju prethodne definicije želimo konstruirati strujni krug koji će se ponašati na takav način, tj. želimo strujni krug koji će na neki način simulirati if granjanje u programiranju. Dakle, ovisno o vrijednosti ulaza s, ovaj strujni krug kao izlaz treba vraćati ili x ili y. Krenimo od tablice istinitosti za mux koja direktno slijedi iz definicije.

Odredimo sada DNF po Teoremu 2. Najprije uočimo da je skup $S(mux)$, koji predstavlja sve argumente za koje mux poprima vrijednost 1, jednak:

$$S(mux) = \{(0, 1, 1), (1, 0, 0), (1, 1, 0), (1, 1, 1)\}.$$

x	0	0	0	0	1	1	1	1
y	0	0	1	1	0	0	1	1
s	0	1	0	1	0	1	0	1
mux	0	0	0	1	1	0	1	1

Tablica 2.1: Tablica istinitosti za multiplekser.

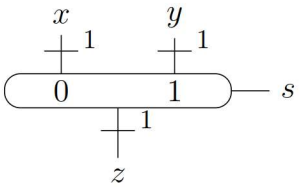
Po definiciji DNF, uz varijable (x, y, s), dobivamo

$$\begin{aligned} e &= \bigvee_{a \in S(mux)} m(a) \\ &= m(0,1,1) \vee m(1,0,0) \vee m(1,1,0) \vee m(1,1,1) \\ &= (x^0 \wedge y^1 \wedge s^1) \vee (x^1 \wedge y^0 \wedge s^0) \vee (x^1 \wedge y^1 \wedge s^0) \vee (x^1 \wedge y^1 \wedge s^1) \\ &= (\bar{x} \wedge y \wedge s) \vee (x \wedge \bar{y} \wedge \bar{s}) \vee (x \wedge y \wedge \bar{s}) \vee (x \wedge y \wedge s). \end{aligned}$$

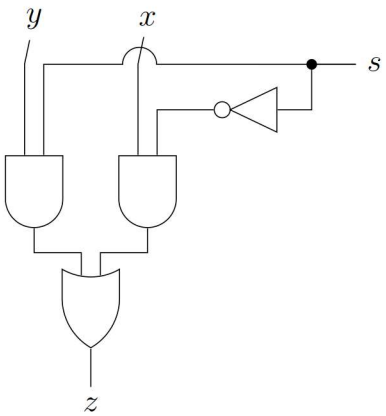
Uočavamo da dobivena DNF sadrži relativno puno logičkih operatora što bi rezultiralo velikim brojem logičkih sklopova. Ako se dobiveni izraz pokuša pojednostaviti, dobije se

$$e \equiv (y \wedge s) \vee (x \wedge \bar{s})$$

što sadrži samo 4 operatora pa je i implementacija jednostavnija. Strujni krug dan je na Slici 2.2b, a simbol koji će se koristiti u kasnijim implementacijama na Slici 2.2a.



(a) Simbol multipleksera.



(b) Implementacija multipleksera.

Slika 2.2: Multiplekser

Ostali navedeni strujni krugovi mogu se na sličan način konstruirati pa su u nastavku dane samo definicije.

2.3.2 Ispitivač nule

Ispitivač nule provjerava je li dani vektor $a[n-1:0]$ zapravo vektor nula 0^n. Formalno, to je funkcija $zero \in \mathcal{B}_n$ definirana s

$$zero(a[n-1:0]) = \begin{cases} 1, & a = 0^n \\ 0, & a \neq 0^n \end{cases}.$$

Sljedeća propozicija daje jedan mogući način na koji se iz $zero$ može konstruirati strujni krug.

Propozicija 2.

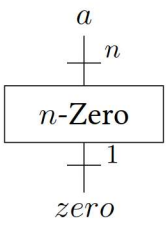
$$zero(a[n-1:0]) \equiv \neg \bigvee_{i=0}^{n-1} a_i.$$

Dokaz.

$$\begin{aligned} zero(a[n-1:0]) = 1 &\Leftrightarrow a[n-1:0] = 0^n \\ &\Leftrightarrow \bigvee_{i=0}^{n-1} a_i = 0 \\ &\Leftrightarrow \neg \bigvee_{i=0}^{n-1} a_i = 1. \end{aligned}$$

Po Lemi 5 slijedi ekvivalencija. □

Simbol ispitivača nule je na Slici 2.3.



Slika 2.3: Simbol n-bitnog ispitivača nule.

2.3.3 Dekoder

Dekoder za dani binarni vektor $x[n-1:0]$ vraća binarni vektor koji sadrži nule na svim mjestima osim na poziciji $\langle x \rangle$. Budući da je $\langle x \rangle \in [0 : 2^n - 1]$, zaključujemo da se radi o vektoru duljine 2^n. Formalno, dekoder je funkcija $dec \in \mathcal{B}_{n,2^n}$ definirana s

$$dec_i(x[n-1:0]) = \begin{cases} 1, & i = \langle x \rangle \\ 0, & i \neq \langle x \rangle \end{cases}.$$

Definirajmo rekurzivnu funkciju $rdec \in \mathcal{B}_{n,2^n}$ na sljedeći način:

$$\begin{aligned} n = 1 \quad & rdec(x_0) = x_0 \bar{x}_0 \\ n > 1 \quad & rdec[2^t \cdot i + j](x[n-1:0]) = rdec_i(x[n-1:t]) \wedge rdec_j(x[t-1:0]) \end{aligned}$$

za sve parove $(i, j) \in [0 : 2^{n-t} - 1] \times [0 : 2^t - 1]$ gdje je $t = \lceil \frac{n}{2} \rceil$.

Propozicija 3. *Neka je $rdec$ prethodno definirana funkcija. Tada za svaki $x \in \mathbb{B}^n$ vrijedi*

$$rdec(x) = dec(x).$$

Dokaz. Dokaz ćemo provesti korištenjem metode matematičke indukcije po n.

- Baza: Za $n = 1$ ćemo raspisati kako izgledaju ove dvije funkcije:

$$dec(0) = 01 \qquad rdec(0) = 0\bar{0} = 01$$
$$dec(1) = 10 \qquad rdec(1) = 1\bar{1} = 10$$

  pa je u ovom slučaju $dec = rdec$.

- Pretpostavka: Pretpostavimo da za $n = k$ vrijedi tvrdnja.

- Korak: Dokažimo tvrdnju za $n = k + 1$. Naime, po definiciji funkcije $rdec$,

$$rdec[2^t \cdot i + j](x[k:0]) = 1 \Leftrightarrow rdec_i(x[k:t]) = 1 \wedge rdec_j(x[t-1:0]) = 1.$$

  Po pretpostavci indukcije, prethodno vrijedi ako i samo ako je

$$dec_i(x[k:t]) = 1 \wedge dec_j(x[t-1:0]) = 1,$$

  a po definiciji funkcije dec ako i samo ako

$$\langle x[k:t] \rangle = i \wedge \langle x[t-1:0] \rangle = j$$

  što je prema Lemi 2 ekvivalentno s

$$\langle x[k:0] \rangle = \langle x[k:t]x[t-1:0] \rangle = 2^t \cdot i + j.$$

  To znači da je $dec[2^t \cdot i + j](x[k:0]) = 1$ pa zaključujemo, po Lemi 5, da je $dec = rdec$.

□

2.3.4 Zbrajanje

Zbrajanje je osnovna matematička operacija koja se u računalu implementira pomoću zbrajala, a koristi se za zbrajanje dvaju binarnih brojeva. Pretpostavit ćemo da se radi o binarnim brojevima bez predznaka.

Želimo konstruirati strujni krug koji će zbrajati bitove. Rezultat zbrajanja dva bita će generalno biti binarni vektor duljine 2, a najveći takav binarni broj je 11_2. No, zbrajanjem samo dva bita ne može se dobiti taj broj. Iz tog razloga uvodimo pojam potpuno zbrajalo koje zbraja tri bita kao funkcija $fa \in \mathcal{B}_{3,2}$ definirana s

$$fa(a, b, c) = bin_2(a + b + c).$$

Budući da je $fa(a, b, c) \in \mathbb{B}^2$, možemo pisati $fa(a, b, c) = c's$ za $c', s \in \mathbb{B}$. Stoga je

$$\langle c's \rangle = a + b + c. \tag{2.2}$$

U sljedećem je primjeru potpuno zbrajalo zapisano u obliku DNF.

a	0	0	0	0	1	1	1	1
b	0	0	1	1	0	0	1	1
c	0	1	0	1	0	1	0	1
c'	0	0	0	1	0	1	1	1
s	0	1	1	0	1	0	0	1

Tablica 2.2: Zbrajanje binarnih znamenki a i b sa znamenkom prijenosa c.

Primjer 8. *Zapišimo funkcije fa_1 i fa_0 u obliku DNF.*
Na temelju Tablice 2.2 istinosnih vrijednosti funkcija fa_1 i fa_0 slijedi:

$$fa_1(a,b,c) \equiv (\bar{a} \wedge b \wedge c) \vee (a \wedge \bar{b} \wedge c) \vee (a \wedge b \wedge \bar{c}) \vee (a \wedge b \wedge c)$$
$$fa_0(a,b,c) \equiv (\bar{a} \wedge \bar{b} \wedge c) \vee (\bar{a} \wedge b \wedge \bar{c}) \vee (a \wedge \bar{b} \wedge \bar{c}) \vee (a \wedge b \wedge c).$$

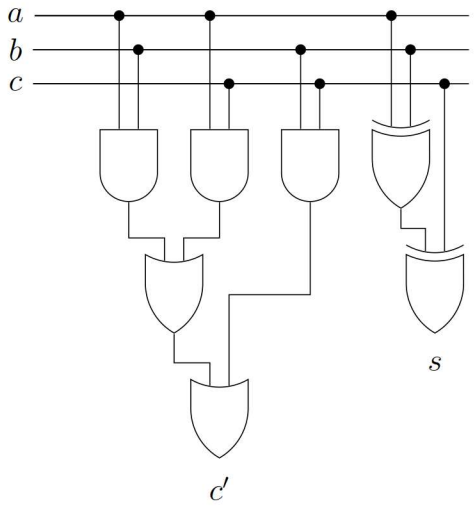
Dobivene DNF se lako pojednostave korištenjem pravila Booleove algebre. Dobije se

$$fa_1(a,b,c) \equiv (a \wedge b) \vee (b \wedge c) \vee (a \wedge c)$$
$$fa_2(a,b,c) \equiv a \veebar b \veebar c.$$

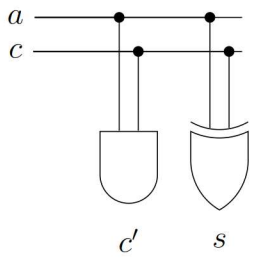
Prikaz strujnog kruga za potpuno zbrajalo dan je na Slici 2.4a. Kada je ulaz b jednak nuli, tada jednakost (2.2) postaje

$$\langle c's \rangle = a + c$$

i takvo zbrajalo zovemo poluzbrajalo. Definira se kao funkcija $ha \in \mathcal{B}_{2,2}$ s $ha(a,c) = bin_2(a+c)$. Prikaz strujnog kruga za poluzbrajalo dan je na slici 2.4b.



(a) Potpuno zbrajalo



(b) Poluzbrajalo

Slika 2.4: Implementacija zbrajala.

Paralelno zbrajalo

U nastavku želimo konstruirati zbrajalo koje će zbrajati binarne brojeve od n bitova. Slično kao i gore, lako se zaključi da zbrajanjem dva n-bitna broja generalno dobijemo $n+1$ bitni broj, ali najviše $\langle 1^n 0\rangle = 2^{n+1} - 2$, dok je najveći broj koji se može dobiti od $n+1$ bitova jednak $2^{n+1} - 1$:

$$\begin{aligned} S &= \langle a[n-1:0]\rangle + \langle b[n-1:0]\rangle \\ &\leq 2^n - 1 + 2^n - 1 && \text{Napomena 1} \\ &= 2^{n+1} - 2. \end{aligned}$$

Zbog toga, kao i u prethodnom slučaju zbrajanja 2 bita, uvodimo i treći bit c_0 koji ćemo zvati znamenka prijenosa. Dakle, $S \in B_{n+1}$ pa je paralelno zbrajalo funkcija $pa\colon \mathbb{B}^n \times \mathbb{B}^n \times \mathbb{B} \to \mathbb{B}^{n+1}$ definirana s

$$pa(a[n-1:0], b[n-1:0], c_0) = bin_{n+1}(\langle a\rangle + \langle b\rangle + c_0). \tag{2.3}$$

No, ovakav zapis nije pogodan za implementaciju. Zbog toga ćemo zbrajanje provesti na uobičajen način, tj. po komponentama od pozicije 0 do pozicije $n-1$ korištenjem potpunog zbrajala. U tu svrhu označimo s c_{i+1} znamenku prijenosa s pozicije i na poziciju $i+1$, a međurezultate (c_{i+1}, s_i) ćemo dobiti koristeći potpuno zbrajalo:

$$\langle c_{i+1} s_i\rangle = a_i + b_i + c_i. \tag{2.4}$$

Uočimo da uz ove oznake $c_n s[n-1:0]$ predstavlja rezultat zbrajanja.

Želimo pokazati da je $pa(a, b, c_0) = c_n s$, a to ćemo napraviti dokazujući ekvivalentnu jednakost u sljedećem teoremu.

Teorem 3. *Neka su $a, b \in \mathbb{B}^n$ i $c_0 \in \mathbb{B}$. Nadalje, neka su $c_n \in \mathbb{B}$ i $s \in \mathbb{B}^n$ izračunati prema prethodno opisanom algoritmu. Tada je*

$$\langle c_n s[n-1:0]\rangle = \langle a[n-1:0]\rangle + \langle b[n-1:0]\rangle + c_0.$$

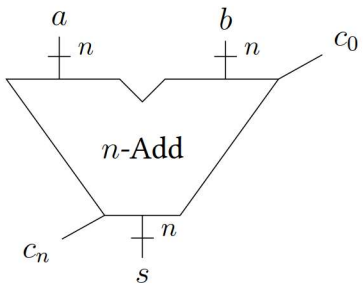
Dokaz. Dokažimo metodom matematičke indukcije po n.

- Baza: Za $n = 1$ imamo bitove $a = a_0$, $b = b_0$ i c_0 te c_1 i $s = s_0$ za koje vrijedi
$$\langle a\rangle + \langle b\rangle + c_0 = a_0 + b_0 + c_0 = \langle c_1 s_0\rangle = \langle c_1 s\rangle.$$
- Pretpostavka: Pretpostavimo da tvrdnja vrijedi za $n = k$.
- Korak: Dokažimo tvrdnju za $n = k+1$.

$$\begin{aligned} &\langle a[k:0]\rangle + \langle b[k:0]\rangle + c_0 \\ &= (a_k + b_k)\cdot 2^k + \langle a[k-1:0]\rangle + \langle b[k-1:0]\rangle + c_0 && \text{Lema 2} \\ &= (a_k + b_k)\cdot 2^k + \langle c_k s[k-1:0]\rangle && \text{pretpostavka indukcije} \\ &= (a_k + b_k + c_k)\cdot 2^k + \langle s[k-1:0]\rangle && \text{Lema 2} \\ &= \langle c_{k+1} s_k\rangle \cdot 2^k + \langle s[k-1:0]\rangle && \text{Jednakost (2.4)} \\ &= \langle c_{k+1} s[k:0]\rangle && \text{Lema 2} \end{aligned}$$

□

Za paralelno zbrajalo koristit ćemo oznaku 2.5.



Slika 2.5: Simbol paralelnog zbrajala.

2.3.5 Oduzimanje

Teorem 4. *Neka su* $a, b \in \mathbb{B}^n$. *Tada vrijedi*

$$[a] - [b] = [a] + [\bar{b}] + 1.$$

Dokaz. Dokaz slijedi direktno iz pete jednakosti Leme 4. □

Teorem 5. *Neka su* $a, b \in \mathbb{B}^n$. *Tada*

$$\langle a \rangle - \langle b \rangle \equiv \langle a \rangle + \langle \bar{b} \rangle + 1 \pmod{2^n}.$$

Ako je dodatno $\langle a \rangle - \langle b \rangle \geq 0$, *tada*

$$\langle a \rangle - \langle b \rangle = (\langle a \rangle + \langle \bar{b} \rangle + 1) \bmod 2^n.$$

Dokaz. Iz prethodnog teorema i druge jednakosti Leme 4 slijedi

$$\begin{aligned} [a] - [b] &\equiv \langle a \rangle - \langle b \rangle \pmod{2^n} \\ [a] - [b] = [a] + [\bar{b}] + 1 &\equiv \langle a \rangle + \langle \bar{b} \rangle + 1 \pmod{2^n}, \end{aligned}$$

pa je

$$\langle a \rangle - \langle b \rangle \equiv \langle a \rangle + \langle \bar{b} \rangle + 1 \pmod{2^n}.$$

Uz $\langle a \rangle - \langle b \rangle \geq 0$ vrijedi

$$\langle a \rangle - \langle b \rangle \in B_n = [0 : 2^n - 1].$$

Po Lemi 1 slijedi

$$\langle a \rangle - \langle b \rangle = (\langle a \rangle + \langle \bar{b} \rangle + 1) \bmod 2^n.$$

□

Ovaj teorem i korolar govore o oduzimanju binarnih brojeva. No, računalo radi s binarnim vektorima pa stoga uvodimo sljedeću definiciju.

Definicija 15. *Neka su* $a, b \in \mathbb{B}^n$. *Definiramo zbrajanje, odnosno oduzimanje, binarnih vektora* a *i* b *na sljedeći način:*

$$\begin{aligned} a +_n b &= bin_n((\langle a \rangle + \langle b \rangle) \bmod 2^n) \\ a -_n b &= bin_n((\langle a \rangle - \langle b \rangle) \bmod 2^n). \end{aligned}$$

Da je prethodna definicija dobra, bez obzira o tome uzimamo li vektore a i b kao brojeve s predznakom ili bez, pokazuje sljedeći teorem.

Teorem 6. *Neka su $a, b \in \mathbb{B}^n$. Tada vrijede sljedeće jednakosti:*

$$\begin{aligned}
\langle a +_n b\rangle &= (\langle a\rangle + \langle b\rangle) \bmod 2^n \\
\langle a -_n b\rangle &= (\langle a\rangle - \langle b\rangle) \bmod 2^n \\
[a +_n b] &= ([a] + [b]) \operatorname{tmod} 2^n \\
[a -_n b] &= ([a] - [b]) \operatorname{tmod} 2^n.
\end{aligned}$$

Dokaz. Prve dvije jednakosti slijede odmah iz definicije. Preostale dvije dokazujemo korištenjem Leme 4:

$$\begin{aligned}
[a \pm_n b] &\equiv \langle a \pm_n b\rangle \pmod{2^n} \\
&\equiv \langle a\rangle \pm \langle b\rangle \pmod{2^n} \\
&\equiv [a] \pm [b] \pmod{2^n}.
\end{aligned}$$

Budući da je $[a \pm_n b] \in T_n$, zaključujemo

$$[a \pm_n b] = ([a] \pm [b]) \operatorname{tmod} 2^n.$$

□

U nastavku slijedi implementacija strujnog kruga koji vrši obje operacije: i zbrajanje i oduzimanje brojeva.

2.3.6 Strujni krug za aritmetičke operacije

Neka su $a, b \in \mathbb{B}^n$ te $u, sub \in \mathbb{B}$ gdje u označava radi li se o operaciji s brojevima bez predznaka ($u = 1$) ili sa predznakom ($u = 0$), a sub označava je li to operacija oduzimanja ili ne. Definirajmo egzaktan rezultat $S \in \mathbb{Z}$ kao

$$S = \begin{cases} [a] + [b], & (u, sub) = 00 \\ [a] - [b], & (u, sub) = 01 \\ \langle a\rangle + \langle b\rangle, & (u, sub) = 10 \\ \langle a\rangle - \langle b\rangle, & (u, sub) = 11. \end{cases}$$

Neka je $s \in \mathbb{B}^n$ rezultat operacije koji će ovaj strujni krug vratiti. Definirajmo ga na sljedeći način:

$$\begin{aligned}
s &= a +_n b, \quad \text{za } sub = 0 \\
s &= a -_n b, \quad \text{za } sub = 1
\end{aligned}.$$

Uočimo da po Teoremu 6 odmah slijedi

$$\begin{aligned}
[s] &= S \operatorname{tmod} 2^n, \quad \text{za } u = 0 \\
\langle s\rangle &= S \bmod 2^n, \quad \text{za } u = 1
\end{aligned}.$$

Sljedeća propozicija daje način konstrukcije rezultata s.

Propozicija 4.

$$s = pa(a, b \veebar sub, sub)$$

Dokaz. Uočimo da za argument $b \veebar sub$, koji ćemo radi jednostavnosti označiti s d, vrijedi

$$d = b \veebar sub = \begin{cases} b, & sub = 0 \\ \bar{b}, & sub = 1 \end{cases}.$$

Nadalje, koristeći definiciju paralelnog zbrajala (2.3) i Lemu 3 imamo

$$\langle s \rangle = (\langle a \rangle + \langle d \rangle + \langle sub \rangle) \bmod 2^n.$$

Naime, $s \in \mathbb{B}^n$, a rezultat $\langle a \rangle + \langle d \rangle + \langle sub \rangle \in B_{n+1}$ pa zbog toga koristimo modulo. Nadalje, raspišimo prethodni izraz ovisno o vrijednosti sub bita:

$$\langle s \rangle = \left(\begin{cases} \langle a \rangle + \langle b \rangle, & sub = 0 \\ \langle a \rangle + \langle \bar{b} \rangle + 1, & sub = 1 \end{cases} \right) \bmod 2^n.$$

Sada iskoristimo Teorem 5 i dobijemo:

$$\begin{aligned} \langle s \rangle &= \left(\begin{cases} \langle a \rangle + \langle b \rangle, & sub = 0 \\ \langle a \rangle - \langle b \rangle, & sub = 1 \end{cases} \right) \bmod 2^n \\ &= \begin{cases} (\langle a \rangle + \langle b \rangle) \bmod 2^n, & sub = 0 \\ (\langle a \rangle - \langle b \rangle) \bmod 2^n, & sub = 1 \end{cases}. \end{aligned}$$

I konačno, primjenom bin_n na obje strane jednakosti dobivamo

$$s = \begin{cases} a +_n b, & sub = 0 \\ a -_n b, & sub = 1 \end{cases}.$$

□

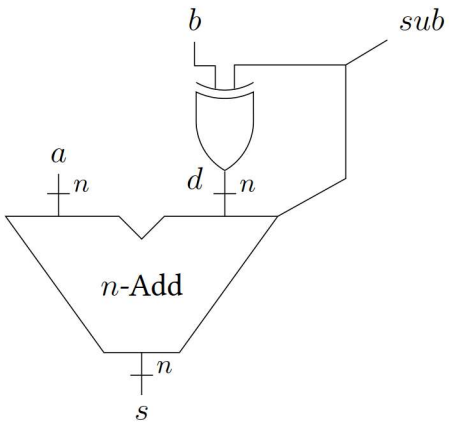
Na Slici 2.6 je dana implementacija ovog strujnog kruga.

Prethodno opisani strujni krug daje dio funkcionalnosti jednog puno većeg strujnog kruga, tzv. aritmetičko-logičke jedinice. Ovaj strujni krug omogućuje izvođenje različitih aritmetičkih i logičkih operacija. Zbog potencijalne kompleksnosti implementacije ovakvog strujnog kruga i zasebnog implementiranja svake od operacija, u nastavku je dan jednostavniji pristup, pomno dizajniran u [2].

2.4 Aritmetičko-logička jedinica

Aritmetičko-logička jedinica al je funkcija $(a, b, af) \mapsto (alures, zr, ng)$ gdje su:

- $a, b \in \mathbb{B}^n$, n paran, vektori na kojima će se raditi operacije,
- $af \in \mathbb{B}^6$ argument koji govori koju operaciju treba izvesti na a i b,



Slika 2.6: Implementacija zbrajanja i oduzimanja.

- $alures \in \mathbb{B}^n$ rezultat,
- $zr, ng \in \mathbb{B}$ dodatni bitovi koji govore je li rezultat nula ili negativan broj.

Vektor af ćemo promatrati kao $af = (za, na, zb, nb, f, no)$, a odgovarajuće bitove ćemo zvati kontrolni bitovi pri čemu je značenje ovih bitova sljedeće:

- za i zb: koriste se za postavljanje vektora a i b na 0,
- na i nb: koriste se za negiranje vektora a i b,
- f: u ovisnosti o ovom bitu odabire se operacija između zbrajanja i konjunkcije,
- no: u ovisnosti o ovom bitu konačni se rezultat negira.

Popis kontrolnih bitova zajedno s operacijama dan je u Tablici 2.3. Dokaz da kontrolni bitovi uistinu daju navedenu operaciju dan je u Dodatku A.

Ovako definiranu aritmetičko-logičku jedinicu ćemo implementirati u HDL-u (vidi Dodatak A). Rezultat $alures$ će se implementirati direktno korištenjem navedene Tablice 2.3. Za implementaciju bita zr koristit ćemo ispitivač nule, a za ng ćemo samo provjeravati je li MSB[1] konačnog rezultata nula ili jedinica:

$$\begin{aligned} ng &= \begin{cases} 1, & alures[n-1] = 1 \\ 0, & alures[n-1] = 0 \end{cases} \\ &= alures[n-1]. \end{aligned}$$

[1]Most significat bit (prvi bit binarnog zapisa nekog broja), za razliku od Least significant bit koji je zadnji bit. Primjerice, u broju 1100111**0** je **1** njegov MSB, dok je **0** LSB.

za	na	zb	nb	f	no	$alures$
1	0	1	0	1	0	0
1	1	1	1	1	1	1
1	1	1	0	1	0	-1
0	0	1	1	0	0	a
1	1	0	0	0	0	b
0	0	1	1	1	1	$-a$
1	1	0	0	1	1	$-b$
0	1	1	1	1	1	$a+1$
1	1	0	1	1	1	$b+1$
0	0	1	1	1	0	$a-1$
1	1	0	0	1	0	$b-1$
0	0	0	0	1	0	$a+b$
0	1	0	0	1	1	$a-b$
0	0	0	1	1	1	$b-a$
0	0	1	1	0	1	$\bar{a}$
1	1	0	0	0	1	$\bar{b}$
0	0	0	0	0	0	$a \wedge b$
0	1	0	1	0	1	$a \vee b$

Tablica 2.3: Popis kontrolnih ulaza i odgovarajućih operacija.

Poglavlje 3

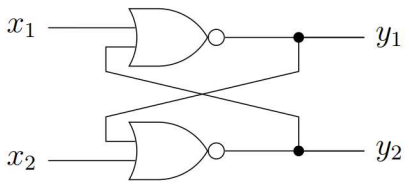
Sekvencijalna logika

Do sada smo u strujnim krugovima koristili samo kombinacijsku logiku. Korištenjem takve logike ne može se implementirati spremanje podataka jer u njoj ne postoji ovisnost o vremenu. To možemo postići korištenjem sekvencijalne logike. Ova vrsta logike omogućava propagiranje stanja strujnih krugova kroz vrijeme. Dakle, sekvencijalni strujni krugovi ovise i o prethodnim stanjima. U nastavku ćemo najprije uvesti najjednostavniji sekvencijalni strujni krug: element za pohranu jednog bita podatka.

3.1 Sekvencijalni strujni krugovi

3.1.1 Asinkroni sekvencijalni strujni krugovi

Najjednostavniji element za spremanje podataka je bistabil. Radi se o strujnom krugu prikazanom na slici 3.1. Odmah uočavamo da ovo po definiciji nije logički strujni krug zato što pripadni graf ovog strujnog kruga sadrži cikluse.



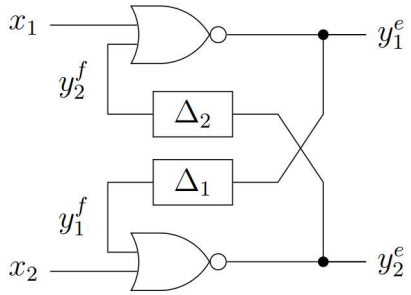
Slika 3.1: Implementacija asinkronog bistabila.

U nastavku ćemo analizirati rad ovog strujnog kruga. Najprije uočavamo da se izlazne vrijednosti y_1 i y_2 ponovno koriste kao ulazne vrijednosti za logičke sklopove NOR. U tom smislu zovemo ih povratne vrijednosti (eng. feedback values). Kako bismo shvatili taj pojam u strujnim krugovima, potrebno je objasniti i pojam vrijeme propagacije (eng. propagation delay). Naime, radi se o tome da svakom logičkom sklopu i vodiču treba određeno vrijeme za procesiranje i prijenos podataka. Drugim riječima, ako se u trenutku t_1 promijenila ulazna vrijednost za logički sklop NOR, tada će odgovarajuća izlazna vrijednosti biti dostupna u trenutku t_2 pri čemu je $\Delta = t_2 - t_1 > 0$. Dakako, radi se o vrlo malenim vrijednostima Δ, gotovo beznačajnim kod kombinacijske logike. No, kod sekvencijalne logike imaju veliku važnost koju ćemo kasnije i uvidjeti.

Radi lakše daljne analize, vrijeme propagacije ćemo prikazati posebnim elementom u strujnom krugu. Zbog toga možemo pretpostaviti da su vremena propagacija logičkih sklopova i vodiča jednaki nula. Također, definirat ćemo po dvije nove varijable za izlazne i povratne vrijednosti:

- y_1^e i y_2^e koje odgovaraju vrijednostima dobivenim iz NOR logičkih sklopova i zovemo ih ekscitacijske varijable (eng. excitation variable),
- y_1^f i y_2^f koje odgovaraju povratnim vrijednostima i zovemo ih povratne varijable (eng. feedback variable).

Na taj način gore opisan strujni krug može se prikazati kao na slici 3.2.



Slika 3.2: Transformacija asinkronog bistabila.

Odnos između povratnih i ekscitacijskih varijabli opisan je sljedećom jednakošću:

$$y_i^f(t+\delta_i) = y_i^e(t), \quad i \in \{1,2\}$$

ili, uz $\delta = \max\{\delta_1, \delta_2\}$,

$$y_i^f(t+\delta) = y_i^e(t).$$

Definicija 16. *Sekvencijalni strujni krug je uređena petorka* $(H, \mathbf{x}, \mathbf{y}^e, \mathbf{y}^f, \mathbf{z})$ *gdje je*

- $\mathbf{x} = \mathbf{x}[1:n]$ *vektor ulaznih signala,*
- $H = g([1:b])$, $b \in \mathbb{N}$, *skup logičkih sklopova,*
- $\mathbf{y}^e = \mathbf{y}^e[1:m] \in Sig(S)^m$ *vektor ekscitacijskih varijabli,*
- $\mathbf{y}^f = \mathbf{y}^f[1:m]$ *vektor povratnih varijabli,*
- $\mathbf{z} = \mathbf{z}[1:l] \in Sig(S)^l$ *vektor izlaznih signala.*

Vektor $\mathbf{y}^f$ *zovemo trenutno stanje strujnog kruga, a* $\mathbf{y}^e$ *sljedeće stanje.*

Primjer 9. *Definirajmo formalno strujni krug sa slike 3.2. To je* $S = (H, \mathbf{x}, \mathbf{y}^e, \mathbf{y}^f, \mathbf{z})$ *gdje je*

- $\mathbf{x} = (x_1, x_2)$,
- $H = \{a, b\}$, $a = (\downarrow, (\pi_1, y_2^f))$, $b = (\downarrow, (\pi_2, y_1^f))$,
- $y_1^e = a$, $y_2^e = b$,

- $z_1 = a$, $z_2 = b$.

Analizirajmo sada istinosnu tablicu asinkronog bistabila uzevši u obzir i vremena propagacija δ_1 i δ_2. Primjerice, neka je $\mathbf{x} = 11$ i $\mathbf{y}^f = 00$. Lako se vidi da je tada i $\mathbf{y}^e = 00$. Pretpostavimo da smo promijenili x_2, tj. $\mathbf{x} = 10$. Tada kao sljedeće stanje strujnog kruga imamo

$$y_1^e(t_0) = x_1 \downarrow y_2^f(t_0) = \overline{x_1 \vee y_2^f(t_0)} = \overline{1 \vee 0} = 0$$
$$y_2^e(t_0) = x_2 \downarrow y_1^f(t_0) = \overline{x_2 \vee y_1^f(t_0)} = \overline{0 \vee 0} = 1.$$

Uočimo da dobivene vrijednosti predstavljaju ujedno i izlazne signale strujnog kruga ($\mathbf{z} = 01$) što općenito ne mora uvijek biti tako.

Nakon što prođe vrijeme propagacije δ_1, varijabla y_1^f će poprimiti vrijednost varijable y_1^e te isto i za y_2^f. To znači da će nakon $\delta = \max\{\delta_1, \delta_2\}$ trenutno stanje biti $\mathbf{y}^f = 01$. Budući da je

$$\mathbf{y}^f(t_0 + \delta) = 01 \neq 00 = \mathbf{y}^f(t_0),$$

povratne varijable će utjecati na sljedeće stanje i dobit ćemo

$$y_1^e(t_0 + \delta) = \overline{x_1 \vee y_2^f(t_0 + \delta)} = \overline{1 \vee 1} = 0$$
$$y_2^e(t_0 + \delta) = \overline{x_2 \vee y_1^f(t_0 + \delta)} = \overline{0 \vee 0} = 1.$$

Uočimo da će sada nakon vremena δ povratne varijable ostati nepromijenjene. Zbog toga će i ekscitacijske varijable ostati nepromijenjene pa strujni krug neće prijeći u neko drugo stanje.

Uvedimo sada pojam stanja.

Definicija 17. *Neka je S sekvencijalni strujni krug. Ukupno stanje s strujnog kruga S je vektor*

$$s = y^f[1 : m]x[1 : n].$$

Primjerice, u prethodnom primjeru strujni krug je prošao kroz tri stanja: $0011 \to 0010 \to 0110$, i nakon toga ostao u stanju 0110. Takvo stanje nazivamo stabilno stanje.

Definicija 18. *Za stanje $s = y^f[1 : m]x[1 : n]$ sekvencijalnog strujnog kruga S kažemo da je stabilno onda kada za sve $i \in [1 : m]$ vrijedi:*

$$y_i^f = y_i^e.$$

Dvije važne pretpostavke koje se koriste kod ovakvih strujnih krugova su pretpostavke na ulazne podatke:

1. isključivo se jedan ulazni podatak može promijeniti, a ne oba istovremeno,
2. promjena ulaznih podataka nije dopuštena sve dok strujni krug ne završi u stabilnom stanju.

Zbog prethodne pretpostavke na promjenu ulaznih podataka, u raspisu tablice svih stanja često ćemo koristiti tzv. Gray code poredak. Radi se o poretku binarnih brojeva u kojem se dva susjedna binarna broja razlikuju isključivo u jednom bitu. Zbog toga će se ovakav poredak koristiti i u tabličnom prikazu svih stanja strujnih krugova. Tablica svih stanja za prethodni primjer dana je u 3.1.

Zaokružena stanja u prethodnoj tablici su stabilna stanja. Radi jednostavnosti, svako stanje je označeno slovom. Primjerice, u prethodnom primjeru smo prošli kroz stanja $\textcircled{c} \to d \to \textcircled{h}$.

$y_1^f y_2^f$ \ x_1x_2	00	01	11	10
00	$11^{\,a}$	$10^{\,b}$	$\textcircled{00}^{\,c}$	$01^{\,d}$
01	$\textcircled{01}^{\,e}$	$00^{\,f}$	$00^{\,g}$	$\textcircled{01}^{\,h}$
11	$00^{\,i}$	$00^{\,j}$	$00^{\,k}$	$00^{\,l}$
10	$\textcircled{10}^{\,m}$	$\textcircled{10}^{\,n}$	$00^{\,o}$	$00^{\,p}$

$y_1^e y_2^e$

Tablica 3.1: Ekscitacijska tablica za bistabil.

Primjer 10. *Pretpostavimo da su ulazni podaci* $\mathbf{x} = 00$ *i* $\mathbf{y}^f = 01$. *U tom slučaju je* $\mathbf{y}^e = 01$ *i nalazimo se u stanju* ⓔ. *Pretpostavimo da smo promijenili ulazni podatak na* $\mathbf{x} = 01$. *Zbog toga prelazimo u stanje* f *i budući da je sada* $\mathbf{y}^e = 00 \neq 01 = \mathbf{y}^f$, *ovo stanje je nestabilno. Nakon vremena propagacije* $\delta = \max\{\delta_1, \delta_2\}$, *vrijedit će* $\mathbf{y}^f = \mathbf{y}^e = 00$ *i prijeći ćemo u stanje* b. *U tom slučaju, nove ekscitacijske varijable su* $\mathbf{y}^e = 10$ *i ponovno se nalazimo u nestabilnom stanju. Dakle, nakon* δ *imamo* $\mathbf{y}^f = 10$ *i prelazimo u stanje* ⓝ *što je ujedno i stabilno stanje jer je sada* $\mathbf{y}^f = \mathbf{y}^e$. *Ukratko, ovaj put možemo opisati na sljedeći način:* $\textcircled{e} \to f \to b \to \textcircled{n}$.

Pogledajmo sada što bi se dogodilo kada bismo postupili u suprotnosti sa pretpostavkom 1, tj. kada bi se oba ulazna podatka istovremeno promijenila.

Primjer 11. *Pretpostavimo da smo u stanju* ⓒ *te da želimo promijeniti ulazne podatke na* $\mathbf{x} = 00$. *Budući da to nije moguće napraviti potpuno istovremeno (zbog različitih vremena propagacije), jedna operacija će se uvijek obaviti prije od druge. Dakle, moguće je sljedeće:*

- *prijelaz iz* $\mathbf{x} = 11$ *u* $\mathbf{x} = 00$ *preko* $\mathbf{x} = 01$. *U tom slučaju imamo:*

$$\textcircled{c} \to b \to \textcircled{n} \to \textcircled{m},$$

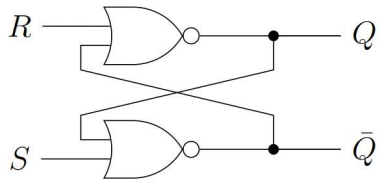
- *prijelaz iz* $\mathbf{x} = 11$ *u* $\mathbf{x} = 00$ *preko* $\mathbf{x} = 10$. *U tom slučaju imamo:*

$$\textcircled{c} \to d \to \textcircled{h} \to \textcircled{e}.$$

Drugim riječima, dobili smo dva različita stanja. Prema tome, u slučaju postupanja suprotno od gornjih pravila, završit ćemo u tzv. utrci u kojoj ne možemo unaprijed znati što će se dogoditi.

Kako bismo na neki način uklonili pretpostavku 1, ulazni podatak $\mathbf{x} = 11$ ćemo zabraniti. U tom slučaju kod stabilnih stanja uočavamo lijepo svojstvo: $y_1^e = \bar{y}_2^e$. Lako se može provjeriti da preostale ulazne podatke $\{00, 01, 10\}$ možemo mijenjati na koji god način i da će rezultat uvijek biti jedinstven. Varijablama x_1 i x_2 možemo pridružiti i simbolična imena: R umjesto x_1 (eng. reset) i S umjesto x_2 (eng. set). Budući da su ekscitacijske i povratne varijable jedna drugoj komplement, promatrat ćemo

samo jedan par, i to y_1^e i y_1^f. Trenutno stanje strujnog kruga označit ćemo s Q (y_1^f), a sljedeće s Q^* (y_1^e). Zbog ovakvih naziva, često se ovaj bistabil zove i asinkroni SR bistabil (eng. SR latch).

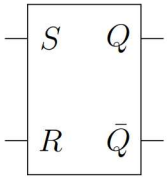


Slika 3.3: Prikaz asinkronog SR bistabila.

Tablica stanja je dana u tablici 3.2, a na slici 3.4 je dan simbol koji ćemo koristiti za asinkroni SR bistabil.

S	R	Q^*
0	0	Q
0	1	0
1	0	1
1	1	Nevažeće

Tablica 3.2: Stanja za asinkroni SR bistabil.



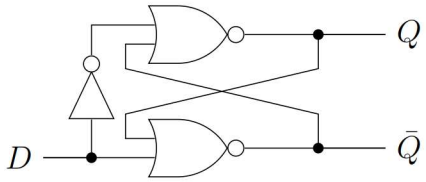
Slika 3.4: Simbol asinkronog SR bistabila.

Do sada smo isključivo govorili o stanjima strujnog kruga no isprva nas je zanimalo spremanje podataka. Stanje Q^* zapravo predstavlja i izlaznu vrijednost ovog strujnog kruga i vrijednost koja je spremljena u asinkronom SR bistabilu. Prema tome,

- $S = 1$ ($R = 0$): 1 će se spremiti u asinkroni SR bistabil,
- $R = 1$ ($S = 0$): 0 će se spremiti u asinkroni SR bistabil,
- $S = 0$, $R = 0$: spremljena vrijednost se neće promijeniti.

Uočavamo da s ovakvom terminologijom situacija $S = 1$, $R = 1$ nema smisla.

Kako bismo zabranili ulazni podatak $\mathbf{x} = 11$, možemo umjesto dva ulazna podatka koristiti samo jedan kao što je prikazano na slici 3.5. Takav bistabil zove se asinkroni D bistabil. Istinosna tablica za asinkroni D bistabil je dana u 3.3.



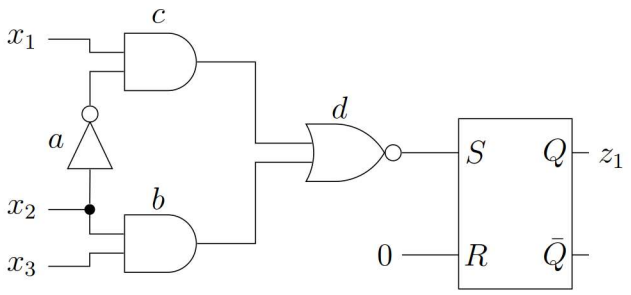
Slika 3.5: Implementacija asinkronog D bistabila.

D	Q^*
0	0
1	1

Tablica 3.3: Stanja za asinkroni D bistabil.

Kod prethodno opisanih strujnih krugova protok vremena je bio kontinuiran. Zbog toga, čim se dogodi promjena na ulaznim varijablama, strujni krug odmah prelazi u drugo stanje. U nekim situacijama takvo ponašanje nije poželjno. Pogledajmo to na sljedećem primjeru.

Primjer 12. *Neka je dan strujni krug kao na slici 3.6. Radi se o strujnom krugu koji se sastoji od kombinacijske logike pomoću koje je određen S, ulazni signal za asinkroni SR bistabil.*



Slika 3.6: Primjer sekvencijalnog strujnog kruga.

Pretpostavimo da su ulazne vrijednosti sljedeće: $x_1 = x_2 = x_3 = 1$. *Lako se vidi da je*

$$d = \overline{(x_1 \wedge \bar{x}_2) \vee (x_2 \wedge x_3)} = 0.$$

Pretpostavit ćemo da je signal za R konstantan i iznosi 0. *Zbog toga, asinkroni SR bistabil neće promijeniti trenutnu vrijednost koja je spremljena. Pretpostavimo da je* $z_1 = 0$. *U nastavku ćemo promotriti što će se dogoditi kada se* x_2 *promijeni na* 0. *Jasno je, obzirom na prethodnu jednakost, da će biti* $d = 0$ *i zbog toga se ništa neće promijeniti sa vrijednosti spremljenom u asinkronom SR bistabilu. Dakle, očekujemo da bude* $z_1 = 0$.

Na slici 3.7 se može vidjeti vremenski slijed promjena nastalih zbog postavljanja $x_2 = 0$. *Tamna boja označava logički 1, a svjetla 0. Na početku su vrijednosti postavljene kao što je i pretpostavljeno u ovom primjeru. Nakon toga se* x_2 *mijenja u* 0. *Zbog vremena propagacije koje imaju vodič i logički sklopovi, sklop* a *(tj.* $\bar{x}_2$*) tek nakon vremena* δ *vraća vrijednost* 1. *Primjera radi možemo pretpostaviti da je otprilike isto vrijeme trebalo da logički sklop* b *vrati* 0. *Nakon toga, novo izračunata vrijednost* 1 *iz sklopa* a *ulazi u sklop* c *te nakon još dodatnih* δ *sklop* c *vraća* 1. *Za to vrijeme vrijednost* 0 *iz sklopa* b *ulazi u sklop* d. *Budući da nova vrijednost iz sklopa* c *nije još uspjela doći, a stara vrijednost je* 0, *rezultat koji će* d *dati je* 1. *Nakon još dodatnog vremena* δ *će i nova vrijednost od* c *dospjeti do sklopa* d *i promijeniti ga natrag na* 0.

Slika 3.7: Primjer grafa koji prikazuje promjenu signala kroz vrijeme.

Uočavamo problem: kombinacijska logika će u konačnici dati točan rezultat, ali joj za to treba određeno vrijeme. S druge strane, asinkroni SR bistabil je osjetljiv na promjene što znači da, ukoliko je S bio dovoljno dugo postavljen na 1, postoji mogućnost da asinkroni SR bistabil spremi 1, a to nije ono što smo očekivali.

Zbog prethodno opisanog problema, ovakvi strujni krugovi se rijetko koriste. Uzrok problema ovakvih strujnih krugova je činjenica da promjene na ulaznim podacima odmah generiraju nove izlazne vrijednosti pa zbog toga i zbog vremena propagacije može doći do krivih rezultata. Ovakve strujne krugove zovemo strujni krugovi ovisni o vrijednosti ulaznog bita (eng. level-triggered circuits). Teoretski je moguće izgraditi cijeli sustav isključivo od asinkrone sekvencijalne logike, no najčešće se asinkrona logika koristi u nekim manjim strujnim krugovima gdje je brzina bitna (npr. kod asinkronih ulaznih podataka na bistabilima 3.11 ili kod implementacije brojača, vidi [5]).

Kako bi se zaobišao prethodno opisani problem, uvedeni su strujni krugovi kod kojih postoji ograničenje na promjene.

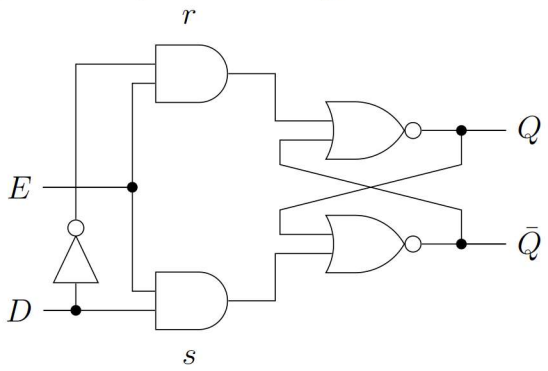
3.1.2 Sinkroni sekvencijalni strujni krugovi

Sinkroni sekvencijalni strujni krugovi su sekvencijalni strujni krugovi kod kojih je vrijeme diskretizirano. To je postignuto uvođenjem takta (eng. clock). Radi se o oscilatoru koji neprestano alternira između 0 i 1 u jednakim vremenskim razmacima. Ti vremenski razmaci moraju biti dovoljno dugi kako bi svaki logički sklop imao dovoljno vremena dok se njegov izlazni podatak ne pojavi u stabilnom stanju. To osigurava da se prethodno opisani problemi više ne pojavljuju. Zbog toga se sinkroni sekvencijalni strujni krugovi koriste u implementaciji mnogih uređaja, između ostalog i današnjih računala.

Pogledajmo sliku 3.8 koja prikazuje asinkroni D bistabil s ulaznim podatkom E (eng. enable) koji predstavlja ograničenje na spremanje podataka. Taj podatak najčešće ima ulogu takta.

E	D	Q^*
0	0	Q
0	1	Q
1	0	0
1	1	1

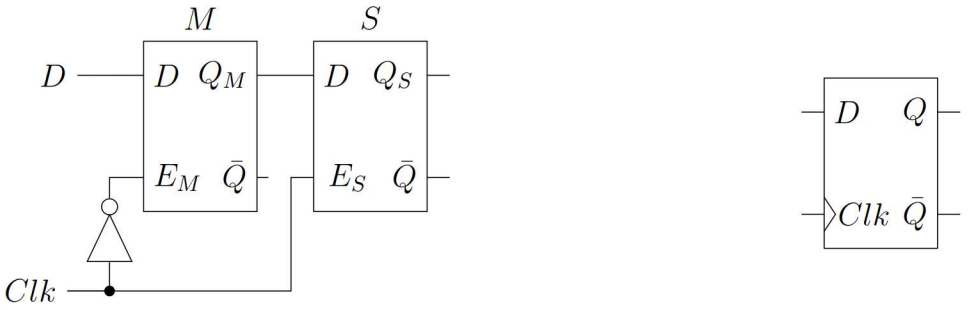
Tablica 3.4: Stanja za asinkroni D bistabil s ograničenjem.



Slika 3.8: Implementacija asinkronog D bistabila s ograničenjem.

Očito je $r = E \wedge \bar{D}$ i $s = E \wedge D$. Zbog toga, za $E = 0$ je $r = s = 0$ i u tom slučaju će vrijednost spremljena u asinkronom D bistabilu s ograničenjem ostati nepromijenjena. Za $E = 1$ imamo $r = \bar{D}$ i $s = D$ i u tom slučaju za $D = 0$ će se vrijednost u asinkronom D bistabilu s ograničenjem resetirati, a za $D = 1$ postaviti.

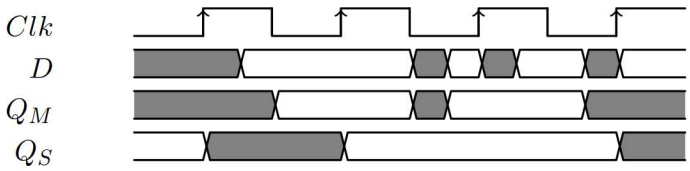
I ovaj bistabil je ovisan o vrijednosti ulaznog bita. Naime, dok je $E = 1$, sve promjene na D će direktno utjecati na rezultat. Mi bismo željeli strujni krug kod kojeg će se podatak spremati isključivo pri promijeni takta. Takvi strujni krugovi zovu se strujni krugovi ovisni o promjeni ulaznog bita (eng. edge-triggered circuits). Postoje dvije vrste takvih strujnih krugova: oni koji spremaju prilikom promjene takta sa 0 na 1, tzv. pozitivn (eng. positive-edge-triggered circuits), i oni koji spremaju prilikom promjene sa 1 na 0, tzv. negativni (eng. negative-edge-triggered). Jedan od strujnih krugova koji ćemo promatrati je pozitivno-sinkroni D bistabil ovisan o promijeni ulaznog bita (eng. positive-edge-triggered D flip-flop) prikazan na slici 3.9.



Slika 3.9: Implementacija i simbol pozitivno-sinkronog D bistabila ovisnog o promijeni ulaznog bita.

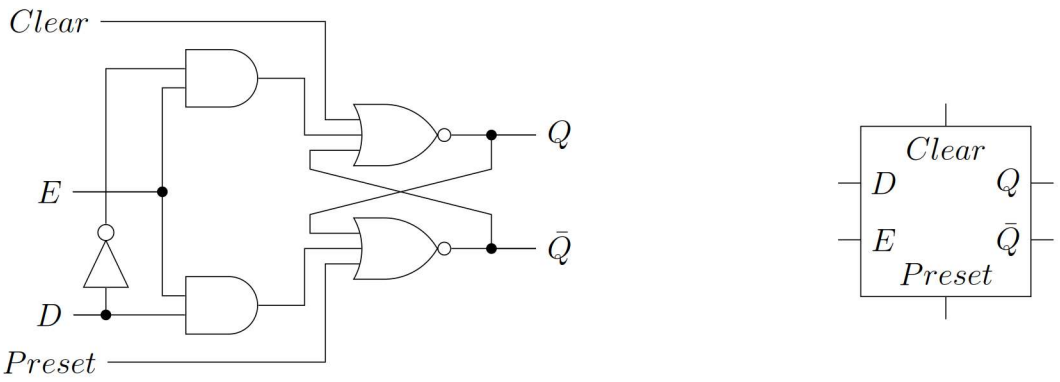
Kao što možemo vidjeti, sinkroni D bistabila sastoji se od dva asinkrona D bistabila s ograničenjem. Prvi se zove glavni (eng. master), a drugi sporedni (eng. slave). Pogledajmo primjer rada ovog bistabila na sljedećem primjeru.

Primjer 13. *Neka je vrijednost spremljena u sinkronom D bistabilu* 0 *i pretpostavimo da želimo spremiti* 1. *Kada je* $Clk = 0$ *postavljanjem* $D = 1$ *dolazi do spremanja u glavni bistabil jer ulazni podatak glavnog bistabila je* $E_M = \overline{Clk} = 1$. *Budući da je* $E_S = 0$, *vrijednost* Q_M *se sada neće spremiti u sporedni bistabil. U idućem taktu je* $Clk = 1$ *te je* $E_S = 1$ *pa se* $Q_M = 1$ *sada sprema u sporedni bistabil. Uočimo da daljne promjene na podatku* D *neće utjecati na trenutno spremljenu vrijednost u sporednom bistabilu. Iz toga vidimo da ovaj sinkroni bistabil uistinu reagira isključivo prilikom promjene takta sa* 0 *na* 1.



Slika 3.10: Primjer rada sinkronog D bistabila.

Često puta će u strujnim krugovima biti potrebna inicijalizacija bistabila na neku poznatu vrijednost neovisno o vrijednosti takta. To se može postići uvođenjem asinkronih ulaznih podataka $Preset$ i $Clear$. Implementacija za asinkroni D bistabil dana je na slici 3.11. Sinkroni D bistabil s asinkronim ulazima implementira se odgovarajućom zamjenom asinkronog bistabila.



Slika 3.11: Implementacija i simbol asinkronog D bistabila s asinkronim ulazima.

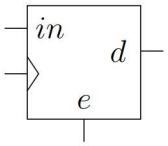
3.2 Primjeri strujnih krugova s taktom

U nastavku ćemo promatrati sekvencijalne strujne krugove koji koriste takt i koji sadrže elemente za spremanje podataka. Taj element ćemo općenito zvati registar. To je uređena četvorka $r = (in, ce, d, e)$ gdje je

- in ulazni podatak registra,
- ce ulaz za takt,
- d trenutno spremljena vrijednost u registru koja je ujedno i izlazni podatak registra,
- e ulazni podatak o kojem ovisi hoće li se nova vrijednost spremiti u registar.

Od navedenih ulaznih podataka, prva tri su nužna za svaki registar, a zadnji je opcionalan (u određenim slučajevima se ne koristi). Registar također može imati i ulazne podatke $Clear$ i $Preset$.

Registar se može implementirati direktno pomoću sinkronog bistabila s dodatnim strujnim krugom za e podatak. Simbol koji ćemo koristiti za registar dan je na slici 3.12.

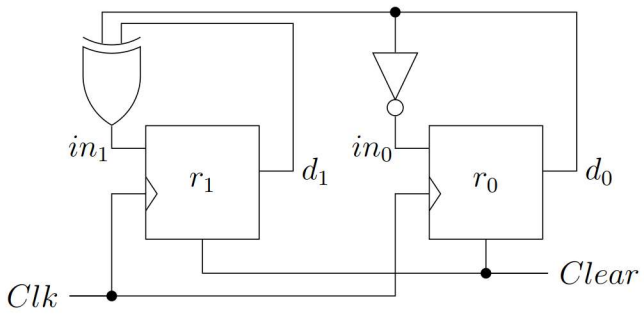


Slika 3.12: Simbol registra.

Registar od n bitova $R = (in[n-1:0], ce, d[n-1:0], e)$ se lako implementira korištenjem n registara od jednog bita $r_i = (in_i, ce, d_i, e)$.

U sljedećem primjeru koristi se registar sa $Clear$ ulaznim podatkom.

Primjer 14. *Na Slici 3.13 dan je strujni krug s taktom.*



Slika 3.13: Primjer strujnog kruga s taktom.

Uočimo da pomoću $Clear$ *ulaznog podatka možemo postići* $d_1 = d_0 = 0$. *Tada je* $in_1 = 0$, $in_0 = 1$ *te prilikom prve promjene takta sa* 0 *na* 1 *će se vrijednosti* in_1 *i* in_0 *spremiti u registre, tj.* $\mathbf{d} = 01$. *Nakon toga ćemo imati* $\mathbf{in} = 10$ *sve dok ne dođe do druge promjene takta s* 0 *na* 1 *nakon koje ćemo u registrima imati vrijednost* $\mathbf{d} = 10$. *Tablica koja prikazuje ponašanje ovog strujnog kruga dana je ispod.*

Clk	0	1	2	3	4	5	
(d_1, d_0)	00	01	10	11	00	01	$\cdots$
(in_1, in_0)	01	10	11	00	01	10	

Uočavamo da je $\mathbf{in} = \mathbf{d} +_4 1$. *Zaista, prema 2.4 prethodno zbrajanje možemo zapisati i na sljedeći način:*

$$\begin{array}{cc} {\scriptstyle c_1} & \\ d_1 & d_0 \\ & 1 \quad + \\ \hline c_2 & s_1 \ s_0 \end{array}$$

Preostaje nam odrediti vrijednosti bitova s_0, s_1, c_1, c_2. *Korištenjem jednakosti iz Primjera 8 uočavamo da je*

$$(c_1, s_0) = d_0 + 1 = (d_0 \wedge 1, d_0 \veebar 1) = (d_0, \neg d_0).$$

Analognim postupkom zaključi se da je

$$(c_2, s_1) = c_1 + d_1 = d_0 + d_1 = (d_0 \wedge d_1, d_0 \veebar d_1).$$

Dakle,

$$\begin{aligned} in_1 &= d_0 \veebar d_1 \\ in_0 &= \neg d_0. \end{aligned}$$

a to uistinu i odgovara strujnom krugu.

3.2.1 RAM

RAM je memorija u računalu koja se koristi za privremenu pohranu podataka i programa koje računalo u tom trenutku izvodi. Dakle, podaci koji već postoje u RAM-u se iz njega mogu dohvaćati, a novi podaci spremati. Ova memorija pohranjuje podatke sve dok se ne prekine napajanje električnom strujom. U trenutku prekida napajanja (npr. prilikom gašenja računala), svi podaci koji su u tom trenutku bili spremljeni u RAM-u se brišu.

Model RAM-a koji ćemo koristiti sastoji se od:

- n-bitnog ulaznog podatka x,
- a-bitne ulazne adrese w,
- signal za pisanje e,
- n-bitni izlaz z.

Svaki registar unutar RAM-a ima svoju adresu. Prema tome, u implementaciji RAM-a koristi se 2^a registara R_i od n bita. Pomoću te adrese možemo pročitati trenutnu vrijednost u određenom registru ili spremiti novu. Za to nam služi ulaz e registra R_i (koji ćemo sada označiti s e_i) koji sada ovisi i o adresi i definiramo ga na sljedeći način:

$$e_i = e \wedge dec_i(w)$$

gdje je $dec\colon \mathbb{B}^a \to \mathbb{B}^{2^a}$ dekoder. Izlazni podatak memorije možemo dobiti na sljedeći način:

$$z = \bigvee_{i=0}^{2^a-1} (z_i \wedge dec_i(w))$$

gdje je z_i izlazni podatak registra R_i. Doista, po definiciji dekodera, iz prethodne jednakosti dobivamo

$$\begin{aligned} z &= \bigvee_{i=0}^{2^a-1} \begin{cases} z_i, & i = \langle w \rangle \\ 0^n, & \text{inače} \end{cases} \\ &= z_{\langle w \rangle} \end{aligned}$$

Dakle, RAM će uvijek vratiti vrijednost spremljenu u registru koji odgovara adresi w. Implementacija RAM-a u HDL-u je dana u Dodatku A.

3.2.2 ROM

ROM je memorija iz koje se podaci mogu samo dohvaćati. Jedna od primjena je pohrana tzv. firmware programa. To je poseban program koji omogućuje kontrolu nad sklopovljem nekog uređaja. Primjerice, u računalu je to BIOS, program koji se prvi pokreće prilikom uključivanja računala, a inicijalizira sklopovlje računala te pokreće operacijski sustav.

ROM se sastoji isključivo od a-bitne ulazne adrese w pa je stoga implementacija slična implementaciji RAM-a, samo što umjesto registara ROM sadrži konstantne vrijednosti. Budući da se podaci samo čitaju, ROM nema ulazni podatak i signal za pisanje pa je to zapravo strujni krug kombinacijske logike.

3.2.3 Kombinacija RAM-a i ROM-a

Često je poželjno implementirati manji dio memorije kao ROM, a ostatak kao RAM. To je tzv. von Neumannova arhitektura računala u kojoj je program za inicijalno pokretanje računala spremljen u ROM-u, a podaci i drugi programi spremljeni u RAM-u. Dakle, prilikom pokretanja računala pokreće se BIOS koji inicijalizira sklopovlje računala te učitava operacijski sustav u RAM, koji se potom pokrene te koristi podatke iz RAM-a i učitava druge programe u RAM.

Dodatak A

ALU

A.1 Dokaz ispravnosti kontrolnih bitova i operacija

Ovdje ćemo predstaviti dokaz operacija danih Tablicom 2.3. Svaki kontrolni vektor možemo pretvoriti u odgovarajući logički izraz. Primjerice, za operaciju $a+1$ kontrolni je vektor $(za, na, zb, nb, f, no) = 011111$. U smislu značenja pojedinih kontrolnih bitova, zaključujemo sljedeće:

- $za = 0$: ulazni podatak a se neće postaviti na nulvektor (a),
- $na = 1$: rezultirajući podatak a će se negirati ($\bar{a}$),
- $zb = 1$: ulazni podatak b će se postaviti na nulvektor ($\mathbf{0}$),
- $nb = 1$: rezultirajući podatak b će se negirati ($\bar{\mathbf{0}}$),
- $f = 1$: prethodno dobiveni a i b će se zbrojiti ($\bar{a} + \bar{\mathbf{0}}$),
- $no = 1$: konačni rezultat će se negirati ($\neg(\bar{a} + \bar{\mathbf{0}})$).

Analogno postupimo sa svim kontrolnim bitovima. Preglednosti radi, operacije ćemo razdvojiti po funkciji koja se koristi: konjunkcija ($f = 0$) i zbrajanje ($f = 1$).

A.1.1 Operacije koje koriste konjunkciju

Popis operacija i pripadnih izraza dan je u Tablici A.1. Dokazi za ove izraze su trivijalni pa su prepušteni čitatelju.

A.1.2 Operacije koje koriste zbrajanje

Popis operacija dan je Tablicom A.2. Prvi izraz je trivijalan, a za ostale ćemo prvo dokazati dvije leme.

Lema 8. *Neka su $a, b \in \mathbb{B}^n$. Tada vrijedi*

$$[a \pm_n b] \equiv [a] \pm [b] \pmod{2^n}.$$

za	na	zb	nb	f	no	$izraz$	$rezultat$
0	0	1	1	0	0	$a \wedge \bar{\mathbf{0}}$	a
1	1	0	0	0	0	$\bar{\mathbf{0}} \wedge b$	b
0	0	1	1	0	1	$\bar{a} \wedge \bar{\mathbf{0}}$	$\bar{a}$
1	1	0	0	0	1	$\bar{\mathbf{0}} \wedge \bar{b}$	$\bar{b}$
0	0	0	0	0	0	$a \wedge b$	$a \wedge b$
0	1	0	1	0	1	$\neg(\bar{a} \wedge \bar{b})$	$a \vee b$

Tablica A.1: Tablica kontrolnih bitova i izraza za $f = 0$.

Dokaz. Dokazujemo korištenjem Leme 4:

$$\begin{aligned}[a \pm_n b] &\equiv \langle a \pm_n b \rangle \pmod{2^n} \\ &\equiv \langle a \rangle \pm \langle b \rangle \pmod{2^n} \\ &\equiv [a] \pm [b] \pmod{2^n}.\end{aligned}$$

□

Lema 9. *Neka su $a, b \in \mathbb{B}^n$. Tada vrijedi*

$$[\neg(\bar{a} +_n \bar{b})] \equiv [a] + [b] + 1 \pmod{2^n}.$$

Dokaz.

$$\begin{aligned}[\neg(\bar{a} +_n \bar{b})] &\equiv -1 - [\bar{a} +_n \bar{b}] \pmod{2^n} && \text{Lema 4} \\ &\equiv -1 - ([\bar{a}] + [\bar{b}]) \pmod{2^n} && \text{Lema 8} \\ &\equiv -1 - (-1 - [a]) - (-1 - [b]) \pmod{2^n} && \text{Lema 4} \\ &\equiv [a] + [b] + 1 \pmod{2^n}.\end{aligned}$$

□

Napomena 6. *Uočimo da se i drugi oblici zbrajanja mogu dokazati preko prethodne leme:*

- $[\neg(\bar{a} +_n b)] \equiv [a] + [\bar{b}] + 1 \pmod{2^n} \equiv [a] - [b] \pmod{2^n}$,
- $[\bar{a} +_n b] \equiv [\neg(\neg(\bar{a} +_n b))] \pmod{2^n} \equiv -1 - [\neg(\bar{a} +_n b)] \pmod{2^n} \equiv -1 - ([a] - [b]) \pmod{2^n}$

Uz pomoć prethodne leme i napomene lako se dokažu svi izrazi iz navedene tablice. Primjerice,

$$\begin{aligned}[\neg(a +_n \bar{b})] &\equiv [\neg(\bar{b} +_n a)] \pmod{2^n} \\ &\equiv [b] - [a] \pmod{2^n} \\ &\equiv [b - a] \pmod{2^n}.\end{aligned}$$

za	na	zb	nb	f	no	$izraz$	$rezultat$
1	0	1	0	1	0	$\mathbf{0}+\mathbf{0}$	0
1	1	1	1	1	1	$\neg(\bar{\mathbf{0}}+\bar{\mathbf{0}})$	1
1	1	1	0	1	0	$\bar{\mathbf{0}}+\mathbf{0}$	-1
0	0	1	1	1	1	$\neg(a+\bar{\mathbf{0}})$	$-a$
1	1	0	0	1	1	$\neg(\bar{\mathbf{0}}+b)$	$-b$
0	1	1	1	1	1	$\neg(\bar{a}+\bar{\mathbf{0}})$	$a+1$
1	1	0	1	1	1	$\neg(\bar{\mathbf{0}}+\bar{b})$	$b+1$
0	0	1	1	1	0	$a+\bar{\mathbf{0}}$	$a-1$
1	1	0	0	1	0	$\bar{\mathbf{0}}+b$	$b-1$
0	0	0	0	1	0	$a+b$	$a+b$
0	1	0	0	1	1	$\neg(\bar{a}+b)$	$a-b$
0	0	0	1	1	1	$\neg(a+\bar{b})$	$b-a$

Tablica A.2: Tablica kontrolnih bitova i izraza za $f = 1$.

A.2 Implementacija ALU-a

```
CHIP ALU {
    IN a[16], b[16], za, na, zb, nb, f, no;
    OUT out[16], zr, ng;

    PARTS:
    // PRE-SETTING
    // ta = za ? 0 : a;
    // pa = na ? !ta : ta;
    MUX16(a = a, b[0..15] = false, s = za, out = ta);
    Not16(in = ta, out = nta);
    MUX16(a = ta, b = nta, s = na, out = pa);

    // tb = zb ? 0 : b;
    // pb = nb ? !tb : tb;
    MUX16(a = b, b[0..15] = false, s = zb, out = tb);
    Not16(in = tb, out = ntb);
    MUX16(a = tb, b = ntb, s = nb, out = pb);

    // COMPUTING
```

```
20      // comp = f ? x+y : x&y;
21      PA(a = pa, b = pb, c0 = false, s = aPb);
22      And16(a = pa, b = pb, out = aAb);
23      MUX16(a = aAb, b = aPb, s = f, out = comp);
24
25      // POST-SETTING
26      // out = no ? !comp : comp;
27      Not16(in = comp, out = ncomp);
28      MUX16(a = comp, b = ncomp, s = no, out = out, out = res, out[15]
         = ng);
29
30      // ADDITIONAL INFO
31      // zr = out == 0 ? true : false
32      // ng = out < 0 ? true : false
33      Zero16(in = res, out = zr);
34  }
35
```

Program A.1: Implementacija ALU-a

A.3 Implementacija RAM-a

```
1   CHIP RAM8 {
2       IN x[16], e, w[3];
3       OUT z[16];
4
5       PARTS:
6       // e_i = 1 ⇔ i = ⟨w⟩
7       Dec3(
8           x = w,
9           out[0] = e0,
10          out[1] = e1,
11          out[2] = e2,
12          out[3] = e3,
13          out[4] = e4,
14          out[5] = e5,
15          out[6] = e6,
16          out[7] = e7
17      );
18
19      And(a = e0, b = e, out = te0);
20      And(a = e1, b = e, out = te1);
21      And(a = e2, b = e, out = te2);
22      And(a = e3, b = e, out = te3);
```

```
23      And(a = e4, b = e, out = te4);
24      And(a = e5, b = e, out = te5);
25      And(a = e6, b = e, out = te6);
26      And(a = e7, b = e, out = te7);
27
28      Register(in = x, e = te0, out = r0);
29      Register(in = x, e = te1, out = r1);
30      Register(in = x, e = te2, out = r2);
31      Register(in = x, e = te3, out = r3);
32      Register(in = x, e = te4, out = r4);
33      Register(in = x, e = te5, out = r5);
34      Register(in = x, e = te6, out = r6);
35      Register(in = x, e = te7, out = r7);
36
37      And1By16(a = e0, b = r0, out = re0);
38      And1By16(a = e1, b = r1, out = re1);
39      And1By16(a = e2, b = r2, out = re2);
40      And1By16(a = e3, b = r3, out = re3);
41      And1By16(a = e4, b = r4, out = re4);
42      And1By16(a = e5, b = r5, out = re5);
43      And1By16(a = e6, b = r6, out = re6);
44      And1By16(a = e7, b = r7, out = re7);
45
46      // z = ⋁_{i=0}^{7}(r_i ∧ e_i)
47      Or16(a = re0, b = re1, out = d1);
48      Or16(a = re2, b = re3, out = d2);
49      Or16(a = re4, b = re5, out = d3);
50      Or16(a = re6, b = re7, out = d4);
51
52      Or16(a = d1, b = d2, out = dd1);
53      Or16(a = d3, b = d4, out = dd2);
54
55      Or16(a = dd1, b = dd2, out = z);
56  }
57
```

Program A.2: Implementacija RAM-a

Bibliografija

[1] C. Baumann, W. Paul, S. Schmaltz, *System Architecture as an Ordinary Engineering Discipline*, 2013

[2] N. Nisan, S. Schocken, *Building a Modern Computer From First Principles: From Nand to Tetris*: `https://www.nand2tetris.org/`

[3] J. Cavanagh, *Sequential Logic: Analysis and Synthesis*, CRC Press, Florida, 2007.

[4] HDL kôd: `https://github.com/imilicic/computer-circuits-hdl`

[5] E. Hwang, *Digital Logic and Microprocessor Design*, Cengage Learning, Inc., Mason, 2005.

[6] T. Ndjountche, *Digital Electronics*, ISTE Ltd., London, 2016.

[7] I. Wegener, *The Complexity of Boolean Functions*, John Wiley & Sons Ltd., New York, 1987.

Sažetak

U radu su opisane vrste računalne logike: kombinacijska i sekvencijalna. Kombinacijska logika odnosi se na logičke strujne krugove kojima izlazne vrijednosti ovise isključivo o trenutnim ulaznim vrijednostima. Ovakvi strujni krugovi mogu se poistovjetiti s Booleovim izrazima. Sekvencijalna logika je poopćenje kombinacijske: uz sve elemente kombinacijske logike još dodatno ima i element za pohranu podataka. Može se reći i da je to logika kod koje izlazne vrijednosti strujnih krugova ovise i o prijašnjim ulaznim vrijednostima. Sekvencijalna logika može se podijeliti na asinkronu i sinkronu. Kod asinkrone logike, elementi za spremanje podataka odmah reagiraju na svaku promjenu ulaznih podataka. To u određenim situacijama može dovesti do krivih rezultata što je ujedno i glavni nedostatak ove logike. Sinkrona logika je odgovor na ovaj problem, ali je dosta sporija od asinkrone.

Ključne riječi

Logički sklop, logički strujni krug, Booleovi izrazi, Booleove funkcije, zbrajalo, aritmetičko-logička jedinica, asinkroni SR bistabil, asinkroni D bistabil, sinkroni D bistabil, registar, memorija, RAM, ROM

Abstract

In this work I explain two different types of computer logic: combinational and sequential. Combinational logic refers to logic circuits whose outputs depend only on current input values. Such logic circuits can be identified with Boolean expression. Sequential logic is generalization of combinational logic: it contains all the elements of combinational logic and has a special memory element. We can also say that sequential logic is logic whose outputs depend also on previous input values. Sequential logic can be divided into asynchronous and synchronous. Memory elements in asynchronous logic immediatelly react to any input change which in certain cases can give us wrong results. This is the main disadvantage of asynchronous logic. This problem is fixed in synchronous logic but, because of that, this logic is much slower than asynchronous logic.

Key words

Logic gate, logic circuit, Boolean expressions, Boolean functions, adder, arithmetic-logic unit, SR latch, D latch, D flip-flop, register, memory, RAM, ROM

Životopis

Rođen sam 7. 9. 1994. u Vinkovcima. Godine 2008. sam završio Osnovnu školu Ivan Goran Kovačić u Štitaru nakon čega sam upisao Gimnaziju Županja, prirodoslovno-matematički smjer. Tijekom srednje škole sudjelovao sam, između ostalog, i na natjecanjima iz matematike na kojem sam 2013. osvojio prvo mjesto na županijskoj razini. Godine 2014. sam upisao Sveučilišni nastavnički studij matematike i informatike na Odjelu za matematiku Sveučilišta J. J. Strossmayera u Osijeku, a godine 2015. se prebacujem na Preddiplomski studij matematike na istom odjelu. Studij sam završio 2016. godine s temom završenog rada Arhitektura računala pod mentorstvom izv. prof. dr. sc. Domagoja Matijevića. Iste godine sam upisao Sveučilišni diplomski studij na navedenom odjelu, smjer Matematika i računarstvo. Tijekom srpnja 2017. godine obavljam stručnu praksu u tvrtki Mono d.o.o, a od listopada 2018. radim u toj tvrtki kao student na poziciji developera.