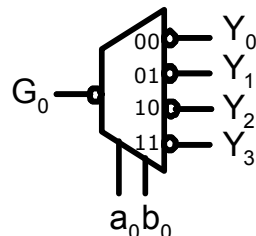


RAZVOJ DIGITALNIH SISTEMOV

Izpit 17. 09. 2014

- Realizirajte funkcijo $f^3 = V(1, 2, 4, 7)$ z redundantnimi mintermi pri $V_x(0, 3, 6)$ z enim TTL dekodrom 74139. Dekoder 74139 ima vhod za omogočenje elementa (G) in izhode Y_0, Y_1, Y_2, Y_3 v negativni logiki. Njegovo delovanje povzema spodnja tabela:

G_0	a_0	b_0	Y_0	Y_1	Y_2	Y_3
1	X	X	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0

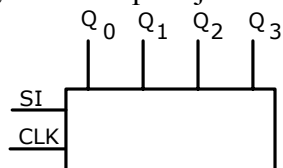


- Uporabite ROM vezje za realizacijo funkcij:

$$g_1 = x_1 + \overline{x_2} \cdot \overline{x_3} \quad g_2 = \overline{x_1} \cdot x_3 + x_1 \cdot x_2$$

ROM vezje ima 3 naslovne spremenljivke in 2 bitno vsebino. Narišite shemo ROM vezja in v shemi označite programirane povezave oz. 'varovalke' s piko (●).

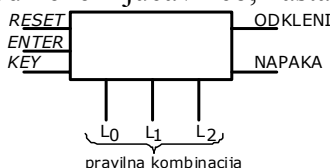
- Sestavite 4-bitni pomikalni register s T-celicami in izbiralniki 2/1. Register ima zaporedni vhod SI (ang. serial input), in vzporedni izhod (Q_0, Q_1, Q_2, Q_3). Uporabite poimenovanje signalov na spodnji sliki.



- Narišite diagram stanj Moore-ov avtomat končnih stanj, ki deluje kot 3-bitna sekvenčna ključavnica. Ključavnica ima tipko za ponastavitev ($RESET$), ki postavlja ključavnico v začetno stanje, tipko ($ENTER$) za vnos nastavitvene kombinacije in preklopnik (KEY) za vnos enega bita kombinacije. Izhod ključavnice je ($ODKLENI$), ki postane '1' ko uporabnik vnese pravilno kombinacijo in izhod ($NAPAKA$), ki postane '1' če je vnesena kombinacija napačna. Uporabo ključavnice povzema spodnje zaporedje:

- 1.) pritisnemo $RESET$
- 2.) s preklopnikom KEY nastavimo bit kombinacije odklepanja
- 3.) pritisnemo $ENTER$
- 4.) izvede se primerjava i-tega bita ($KEY=L_i$)
- 5.) dvakrat ponovimo korake 2 – 4
- 6.) če je vnesena 3-bitna kombinacija pravilna, se postavi $ODKLENI='1'$, sicer se postavi $NAPAKA='1'$.
- 7.) ponoven pritisk na $RESET$ nas vrne na korak 1.

Pravilno kombinacijo, ki odklene ključavnico, nastavljamo z biti L_0, L_1 in L_2 .



Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

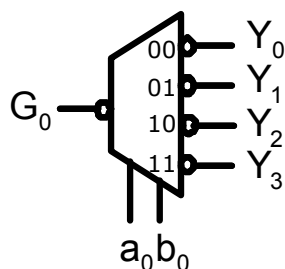
Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete (VSŠ, UNI).

Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>

Rešitev 1. naloge:

Delovanje dekoderja 74139¹ povzema spodnja tabela:

G_0	a_0	b_0	Y_0	Y_1	Y_2	Y_3
1	X	X	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0



Funkcijo f narišemo v Veitch–ev diagram, da si jo lažje predstavljamo:

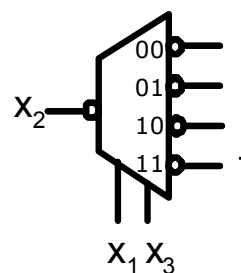
	x_1			
x_2	X	1	X	1
	1	0	1	X
	x_3			

Čim imamo na voljo dekodirnik z ENABLE vhomom v negativni logiki preverimo ali obstaja spremenljivka v osnovni ali negirani obliki, pri kateri so vsa polja enaka '1' vključno z redundancami X.

V zgornjem Veitch–evem diagramu je to spremenljivka x_2 : Namreč, če je $x_2=1$, potem lahko vse redundance izberemo tako, da bo $f=1$ za vse vrednosti $x_2=1$. Ko določimo spremenljivko za omogočenje elementa (G), opazujemo samo preostali del Veitch–evega diagrama. Spodnje 4 vrednosti diagrama narišemo v novem Veitch–evem diagramu 2 spremenljivk.

	x_1	
x_3	0	1
	1	X

Dekoder ima aktivno nizke izhode, zato iz nastalega diagrama realiziramo *negacijo* funkcije f zato v Veitch–evem diagramu združujemo ničle, kar nastopa samo v primeru ko sta $x_1=1$ in $x_3=1$.



¹<http://www.alldatasheet.com/view.jsp?Searchword=74HC139>

Rešitev 2. naloge:

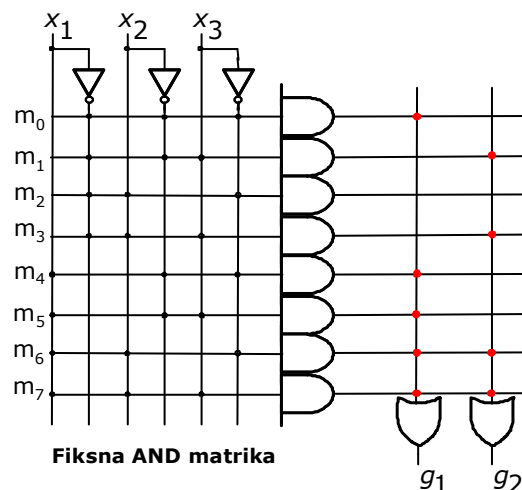
Če se funkcije ne nahajajo v popolni disjunktivni normalni obliki (PDNO), jih prevedemo v to obliko z uporabo pravil Boole-ove algebre. Funkcijo lahko tudi izpišemo v Veitch-ev diagram in izpišemo številke mintermov, kjer je funkcija enaka '1'.

$$\begin{aligned} g_1(x_1, x_2, x_3) &= x_1 + \overline{x_2} \cdot \overline{x_3} = x_1 \cdot (\overline{x_2} \cdot \overline{x_3} + x_2 \cdot \overline{x_3} + \overline{x_2} \cdot x_3 + x_2 \cdot x_3) + (\overline{x_1} + x_1) \cdot \overline{x_2} \cdot \overline{x_3} \\ g_1(x_1, x_2, x_3) &= x_1 \cdot \overline{x_2} \cdot \overline{x_3} + x_1 \cdot x_2 \cdot \overline{x_3} + x_1 \cdot \overline{x_2} \cdot x_3 + x_1 \cdot x_2 \cdot x_3 + \overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3} + \overline{x_1} \cdot x_2 \cdot \overline{x_3} \\ g_1(x_1, x_2, x_3) &= x_1 \cdot \overline{x_2} \cdot \overline{x_3} + x_1 \cdot x_2 \cdot \overline{x_3} + x_1 \cdot \overline{x_2} \cdot x_3 + x_1 \cdot x_2 \cdot x_3 + \overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3} \\ g_1(x_1, x_2, x_3) &= V(4, 6, 5, 7, 0) \end{aligned}$$

Podobno storimo še za preostale funkcije:

$$\begin{aligned} g_2 &= \overline{x_1} \cdot x_3 + x_1 \cdot x_2 = \overline{x_1} \cdot (\overline{x_2} + x_2) \cdot x_3 + x_1 \cdot x_2 \cdot (\overline{x_3} + x_3) \\ g_2(x_1, x_2, x_3) &= \overline{x_1} \cdot \overline{x_2} \cdot x_3 + \overline{x_1} \cdot x_2 \cdot x_3 + x_1 \cdot x_2 \cdot \overline{x_3} + x_1 \cdot x_2 \cdot x_3 \\ g_2(x_1, x_2, x_3) &= V(1, 3, 6, 7) \end{aligned}$$

PDNO je najprimernejša oblika za realizacijo z ROM, ker je matrika AND fiksna. Programirane vrednosti AND matrike predstavljajo vse minterme funkcije treh spremenljivk ($x_1 \ x_2 \ x_3$) od m_0 do m_7 . Številka minterma določa naslov lokacije ROM pomnilnika.



Narišemo celotno vezje ROM strukture in vstavimo pike (•) v OR matriki tam, kjer želimo programirati določeno spremenljivko v členu PDNO.

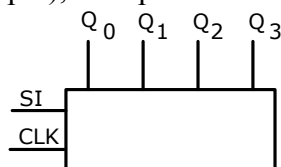
Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete (VŠŠ, UNI).

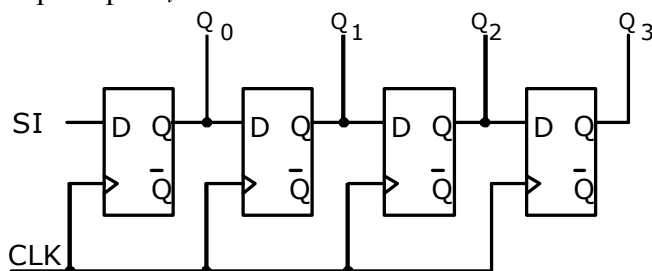
Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>

Rešitev 3. naloge:

Sestavite 4-bitni pomikalni register s T-celicami in izbiralniki 2/1. Register ima zaporedni vhod SI (ang. serial input), in vzporedni izhod (Q_0, Q_1, Q_2, Q_3)



Zaporedno-vzporedni (SIPO) pomikalni register, realiziran s pomočjo D-FF, je veriga kaskadno vezanih D-FF, v kateri je izhod prejšnjega flip-flopa Q_{i-1} vezan na vhod naslednjega flip-flopa D_i .



Če želimo pomikalni register sestaviti iz T-FF in 2/1 izbiralnikov, moramo pravzaprav realizirati celico D-FF s pomočjo T-FF in 2/1 izbiralnikov. V ta namen zapišemo tabelo D-FF, pri kateri dodamo izhodni stolpec T vhoda.

D	$Q(t)$	$Q(t+1)$	T
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0

XOR vrata moramo realizirati s pomočjo 2/1 izbiralnikov, zato zapišemo enačbo XOR funkcije:

$$f = x \oplus y = \bar{x} \cdot y + x \cdot \bar{y}$$

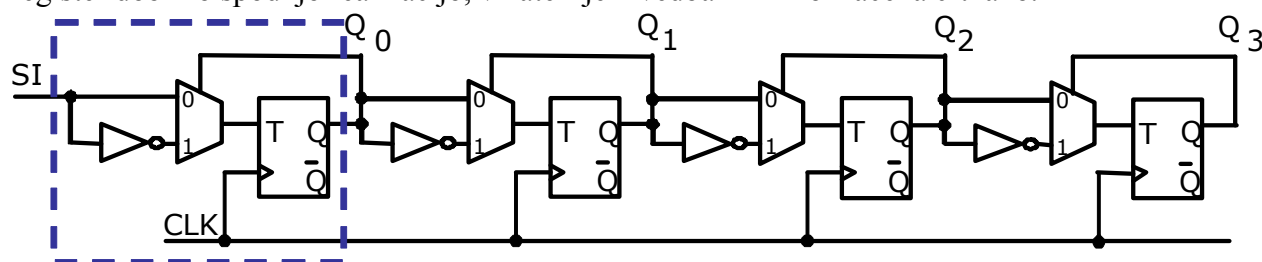
Iz tabele sledi, da je T vhod XOR operacija $Q(t)$ in vhoda D-FF, ki ga realiziramo.

Funkcijo f realiziramo z izbiralnikom tako, da naredimo razvoj po spremenljivki x in dobimo:

$$Q(t+1) = Q(t) \oplus D$$

x	f
0	y
1	y'

Če nastali D-FF iz T-FF in 2/1 izbiralnika sestavimo skupaj v 4-bitni pomikalni register dobimo spodnjo realizacijo, v kateri je izvedba D-FF označena črtkano.



Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete (VSŠ, UNI).

Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>

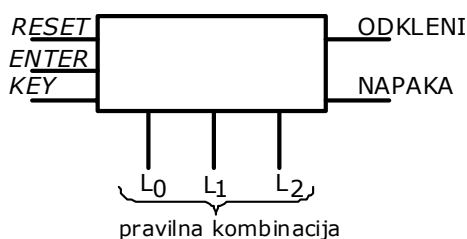
Rešitev 4. naloge:

Narišite diagram stanj Moore-ov avtomat končnih stanj, ki deluje kot 3-bitna sekvenčna ključavnica. Ključavnica ima tipko za ponastavitev (*RESET*), ki postavlja ključavnico v začetno stanje, tipko (*ENTER*) za vnos nastavljene kombinacije in preklopnik (*KEY*) za vnos enega bita kombinacije. Izhod ključavnice je (*ODKLENI*), ki postane '1' ko uporabnik vnese pravilno kombinacijo in izhod (*NAPAKA*), ki postane '1' če je vnesena kombinacija napačna.

Uporabo ključavnice povzema spodnje zaporedje:

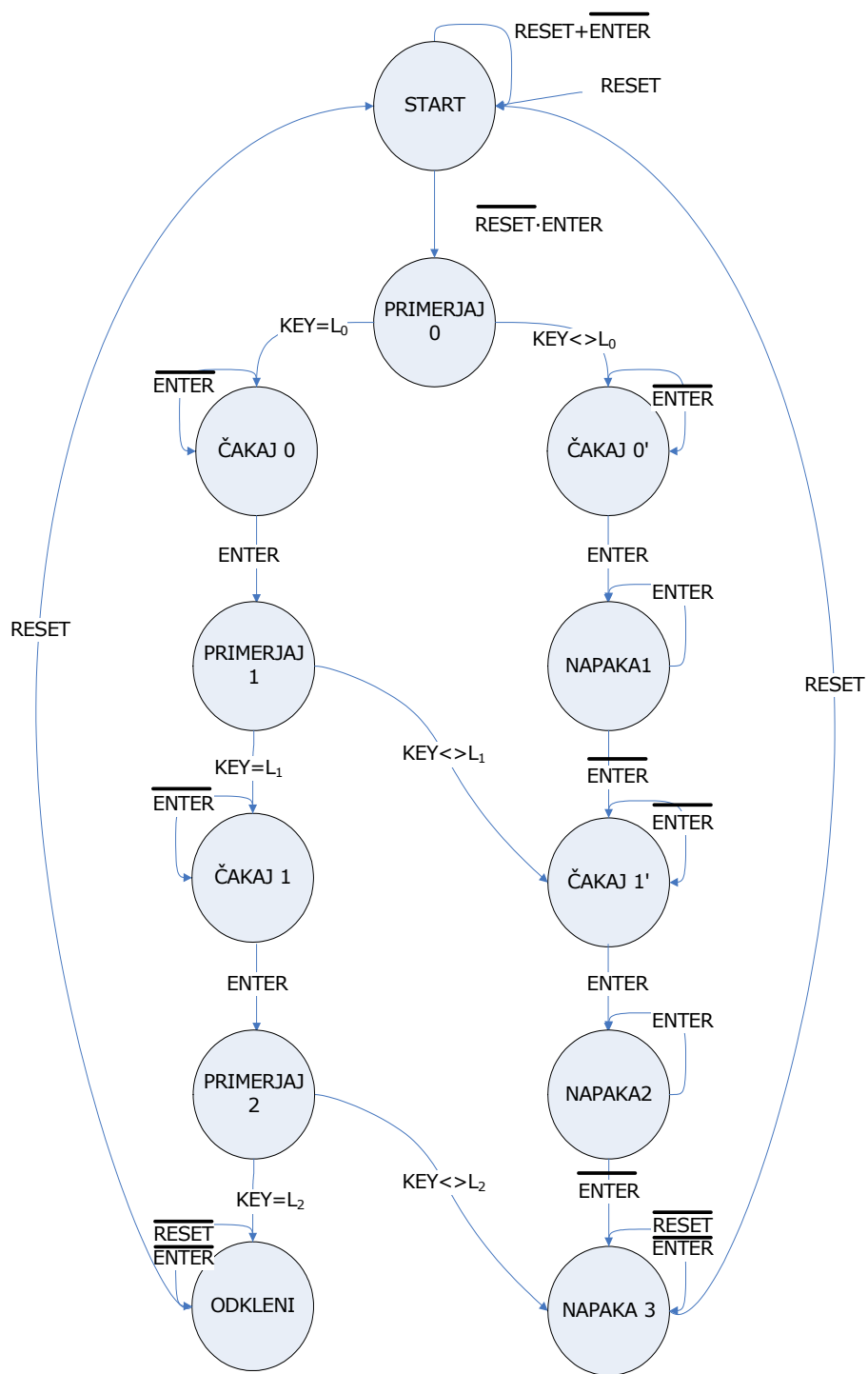
- 1.) pritisnemo *RESET*
- 2.) s preklopnikom *KEY* nastavimo bit kombinacije odklepanja
- 3.) pritisnemo *ENTER*
- 4.) izvede se primerjava i -tega bita ($KEY = L_i$)
- 5.) dvakrat ponovimo korake 2 – 4
- 6.) če je vnesena 3-bitna kombinacija pravilna, se postavi *ODKLENI*='1', sicer se postavi *NAPAKA*='1'.
- 7.) ponoven pritisk na *RESET* nas vrne na korak 1.

Pravilno kombinacijo, ki odklene ključavnico, nastavljamo z biti L_0 , L_1 in L_2 .



Ključavnica se ob vklopu ali ob ponastavitvi (*RESET*) nahaja v stanju *START*. V tem stanju uporabnik nastavi preklopnik *KEY* v stanje prvega bita kombinacije ('0' ali '1'). Iz tega stanja lahko pride v stanje *PRIMERJAJ 0* samo ob pogoju, da *RESET* ni pritisnjen in da je *ENTER* pritisnjen. V tem stanju je lahko vnesena kombinacija pravilna ($KEY = L_0$) ali nepravilna ($KEY \neq L_0$). Če je pravilna, potem preide v stanje *ČAKAJ 0*, v katerem čaka da uporabnik spusti tipko *ENTER*. Uporabnik v tem stanju nastavi drugi bit kombinacije (*KEY*) in ponovno pritisne *ENTER*. Avtomat preide v stanje *PRIMERJAJ 1*, od koder sta zopet dve možnosti. Omenjeni postopek se lahko ponavlja za več bitov kombinacije. Bistveno je, da pred vsako primerjavo postavimo stanje čakanja, v katerem čakamo, da uporabnik spusti tipko *ENTER*, nato nastavi bit kombinacije in šele nato preide v stanje nove primerjave ob ponovnem pritisku na *ENTER*. V zadnjem stanju primerjave avtomat preide v stanje *ODKLENI*.

Če se uporabnik pri vnašanju zmoti, preide avtomat v sekvenco stanj napake, po kateri mora vnesti še dve mesti kode (lahko samo dvakrat pritisne *ENTER*) in šele nato preide v stanje *NAPAKA 3*, v katerem se postavi izhod *NAPAKA*.



Čas pisanja je 60 minut. Vsaka naloga je vredna 10 točk.

Na list z rešitvami se podpišite in napišite še vpisno številko ter kateri predmet pišete (VSŠ, UNI).

Rezultati bodo objavljeni na: <https://estudent.fri.uni-lj.si>