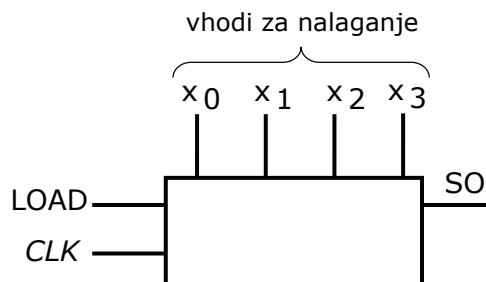


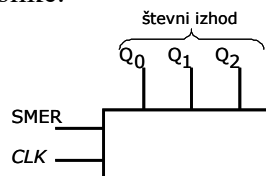
RAZVOJ DIGITALNIH SISTEMOV

2. kolokvij
17.1.2025

1. Realizirajte RS-flip flop z uporabo T flip flopa in logičnih vrat.
2. Narišite vezje 4-bitnega pomikalnega registra z D flip flopi in kombinacijskimi logičnimi gradniki. Register ima zaporedni izhod *SO* (ang. serial output) in vzporedni vhod (x_0, x_1, x_2, x_3) ter krmilni signal *LOAD*. Ko je *LOAD*='1', se vsebina vzporedno naloži v register, sicer register zaporedno pomika vsebino proti zaporednemu izhodu *SO*. Ob pomikanju se na izpraznjena mesta registra vpisujejo '0'.



3. Prikažite sintezo sinhronnega dvosmernega 3-bitnega števca z uporabo T flip flopov: Zapišite tabelo prehajanja stanj in določite enačbe flip flopov. Števec ima vhod *SMER*, ki določa smer štetja: Če je *SMER*='0', števec šteje naraščajoče, sicer padajoče. Imena signalov so razvidna iz spodnje slike.



4. Načrtajte diagram prehajanja stanj Mooreovega avtomata, ki deluje kot krmilje za kavni avtomat. Kava stane 15 centov, plačujemo pa lahko s kovancema za 5 in 10 centov.

Krmilje ima:

- vhod *5cent*, ki postane '1', ko uporabnik vrže v avtomat kovanec za 5 centov,
- vhod *10cent* ki postane '1', ko uporabnik vrže v avtomat kovanec za 10 centov,
- izhod *p*, ki postane '1', ko uporabnik vrže v avtomat skupno 15 centov.

Avtomat ne vrača drobiža in se ob detekciji plačila 15 centov ne vrača nazaj v začetno stanje, ampak ostane v končnem stanju. Vnos dveh kovancev naenkrat ni mogoč.

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. Rezultati bodo objavljeni na domači strani predmeta.

Rešitev 1. naloge

Za vezje RS-FF narišemo pravilnostno tabelo, pri čemer na vhodni strani zberemo vhode R, S in trenutno stanje Q(t), na izhodni pa naslednje stanje Q(t+1). RS-FF opravlja tri funkcije (HOLD, RESET, SET) glede na kombinacijo vhodnih signalov, medtem ko T-FF opravlja samo dve (HOLD, INVERT).

S	R	Q(t)	Q(t+1)	T	funkcija RS	funkcija T
0	0	0	0	0	HOLD	HOLD
0	0	1	1	0	HOLD	HOLD
0	1	0	0	0	RESET	HOLD
0	1	1	0	1	RESET	INVERT
1	0	0	1	1	SET	INVERT
1	0	1	1	0	SET	HOLD
1	1	0	X	X	/	/
1	1	1	X	X	/	/

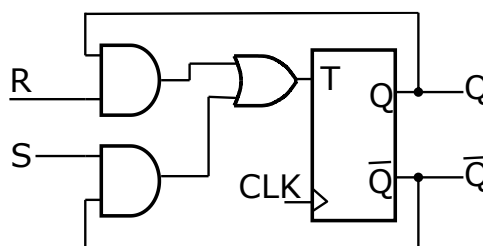
Iz tabele narišemo Veitchev diagram za vhod T v odvisnosti od vhodov R, S in trenutnega stanja Q(t).

$$T = S \cdot \overline{Q(t)} + R \cdot Q(t)$$

Vezje narišemo.

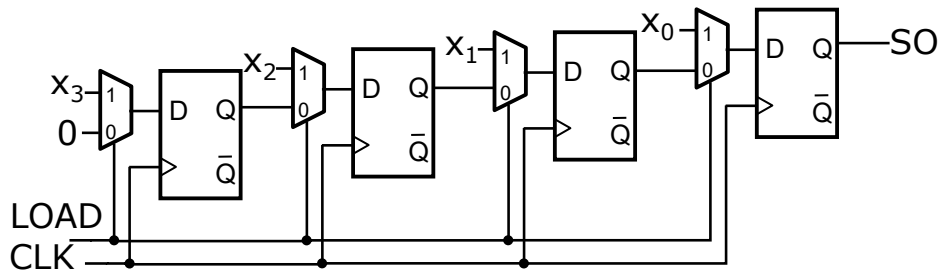
T:

	S			
R	X	X	1	0
	1	0	0	0
	Q(t)			

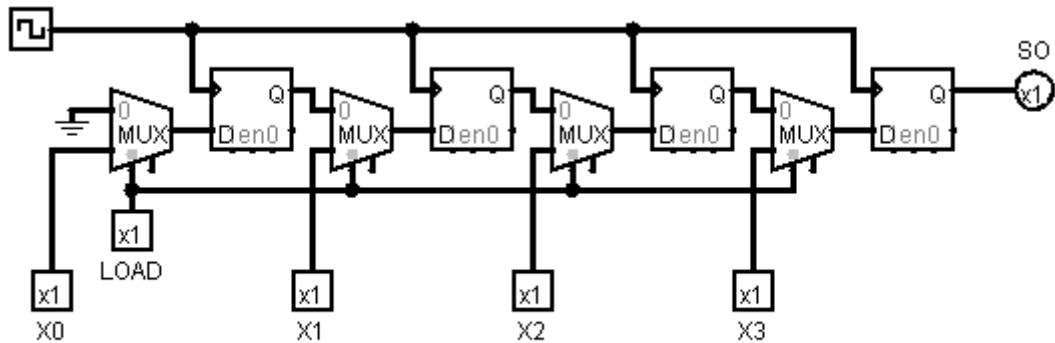


Rešitev 2. naloge:

Vzporedno – zaporedni (PISO) pomikalni register, ki ima možnost vzporednega nalaganja in pomikanja najbolj enostavno realiziramo s pomočjo 4-bitne vezave univerzalnih logičnih modulov ULM. Register ima dve funkciji: nalaganje in pomikanje, torej ima ULM multiplekser 2/1, ki je povezan na vhod D-FF.



Enaka rešitev je prikazana na spodnji sliki.



Neposredna izvedba PISO registra z D-FF

Vezje se nahaja v Logisim predlogah rešenih nalog na domači strani predmeta:
Logisim\shift_reg\shift_reg_4bit_PISO_dff.circ

Rešitev 3. naloge:

Postopek sinteze zahteva, da zapišemo tabelo prehajanja stanj števca:

SME R	Q 2	Q 1	Q 0	Q 2	Q 1	Q 0	T 2	T 1	T 0
0	0	0	0	0	0	1	0	0	1
0	0	0	1	0	1	0	0	1	1
0	0	1	0	0	1	1	0	0	1
0	0	1	1	1	0	0	1	1	1
0	1	0	0	1	0	1	0	0	1
0	1	0	1	1	1	0	0	1	1
0	1	1	0	1	1	1	0	0	1
0	1	1	1	0	0	0	1	1	1
1	0	0	0	1	1	1	1	1	1
1	0	0	1	0	0	0	0	0	1
1	0	1	0	0	0	1	0	1	1
1	0	1	1	0	1	0	0	0	1
1	1	0	0	0	1	1	1	1	1
1	1	0	1	1	0	0	0	0	1
1	1	1	0	1	0	1	0	1	1
1	1	1	1	1	1	0	0	0	1

Normalna analiza bi zahtevala, da narišemo Veitch–eve diagrame za štiri spremenljivke za vsak vhod T–FF, vendar ker so T–FF po svoji naravi primerni za realizacijo števecv, so praviloma njihove vhodne enačbe zelo enostavne. Iz tabele prehajanja stanj števca določimo enačbe T–FF:

Iz stolpca T₀ se vidi, da je T₀=**1**'. Kot zanimivost omenimo dejstvo, ki se ponavlja pri večini sinhronih števecv: Če namesto '1' na T₀ vodimo nek zunanji signal, nam to omogoča štetje števca (ENABLE oz. EN).

Iz stolpca T₁ se vidi, da se ponavlja vzorec **01**, če je SMER='0' in **10**, če je SMER='1'.

SMER	T ₁
0	Q ₀
1	Q ₀ '

kar lahko kratko zapišemo kot:

$$T_1 = \text{SMER} \cdot \overline{Q_0} + \overline{\text{SMER}} \cdot Q_0 = \text{SMER} \oplus Q_0$$

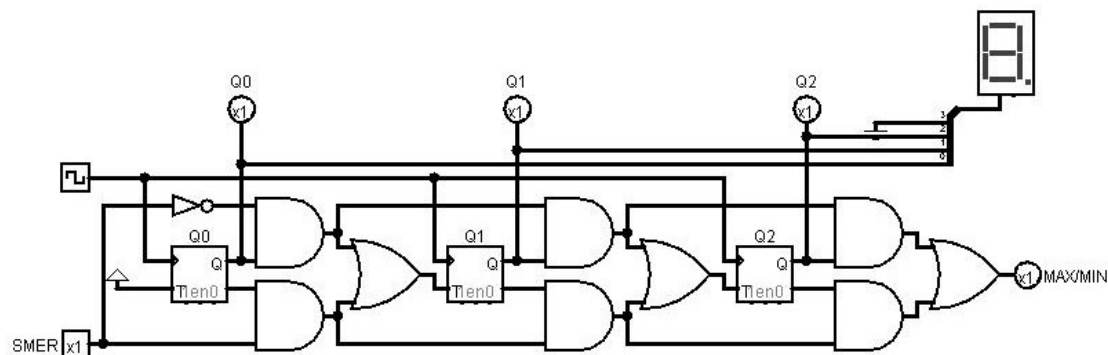
Za T₂ se da enostavno ugotoviti realizacijo iz Veitch–evega diagrama:

	SMER				
Q ₂	1	0	0	0	Q ₀
	0	0	1	0	
	0	0	1	0	
	1	0	0	0	
Q ₁					

$$T_2 = \text{SMER} \cdot \overline{Q_1} \cdot \overline{Q_0} + \overline{\text{SMER}} \cdot Q_1 \cdot Q_0$$

V enačbi za T₂ poiščemo podobnosti z enačbo za T₁: Enačba za T₁ vsebuje konjunkciji SMER·Q₀' in $\overline{\text{SMER}} \cdot Q_0$, ki sta vsebovani tudi v enačbi za T₂, kar nam dodatno poenostavi realizacijo števca. Obenem nam taka realizacija nakazuje osnovno strukturo, ki jo lahko s ponavljanjem razširimo v večbitni dvosmerni sinhroni števec.

Primer podobnega vezja 4-bitnega dvojiškega dvosmernega števca, ki ima še vzporedno nalaganje je 74191¹. Če boste primerjali našo realizacijo in realizacijo v podatkovnem listu, boste opazili, da je v dejanski realizaciji 74191 precej več večvhodnih AND vrat: Delno je razlog za to v dodani logiki za vzporedno nalaganje, delno pa tudi zato, da zagotovimo enakomerno zakasnitev med posameznimi stopnjami števca.



Ko narišemo vezje dvosmernega števca, zelo spominja na združitev sinhronnega števca za štetje navzgor in sinhronnega števca za štetje navzdol: Če bi števec vseboval samo zgornja AND vrata (vezanih neposredno na T vhod – brez OR) bi bil to števec navzgor, če pa samo spodnja AND bi bil števec navzdol. Signal SMER določa katera AND vrata so omogočena:

- zgornja AND vrata, ko je SMER='0' – štejemo naraščajoče,
- spodnja AND vrata, ko je SMER='1' – štejemo padajoče.

Zahteve naloge so s tem izpolnjene. V nadaljevanju bomo pokazali še dodajanje ostalih krmilnih signalov (EN - omogočanje štetja, RCO, MAX/MIN, LOAD - vzporedno nalaganje).

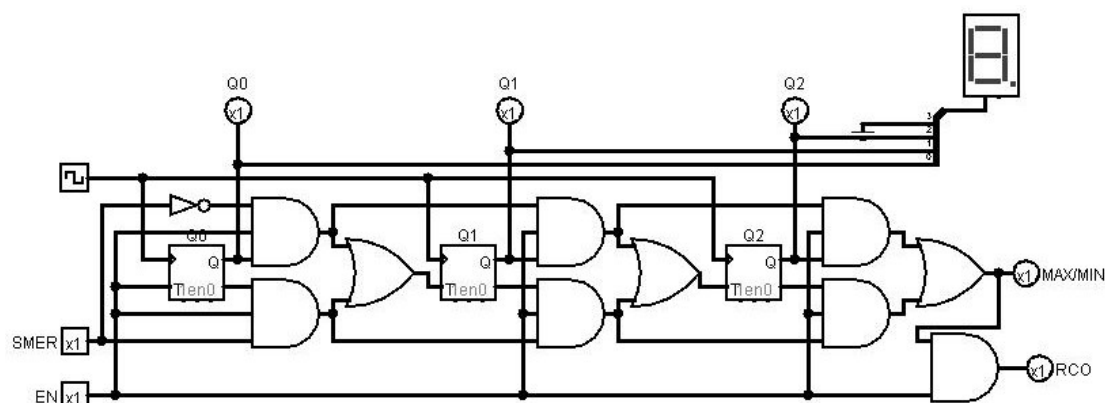
Signal EN je signal za omogočanje štetja – dokler je EN='0' se vsebina vseh T–FF ne spreminja, ampak ohranja trenutno stanje. Iz sheme števca je razviden tudi način razširitve števca na večje število bitov.

Pri tovrstnih števcih želimo realizirati tudi signal za proženje naslednjih stopenj števca RCO (ang.. ripple carry out) poimenovan včasih tudi TC (ang. terminal count), oz. tudi RC (ang. ripple clock). RCO je signal, ki postane '1', ko števec preide iz najvišjega stanja (v našem primeru "111") v stanje "000" pri štetju navzgor in ko preide iz najnižjega stanja "000" v najvišje stanje ("111") pri štetju navzdol:

SMER	RCO
0	$Q_0 \cdot Q_1 \cdot Q_2$
1	$Q_0' \cdot Q_1' \cdot Q_2'$

Tak signal uporabljamo pri realizaciji večbitnih števcov tako, da izdelane 3 bitne števce vezemo kaskadno – torej da signal RCO vezemo na EN signal naslednjega vezja. Za realizacijo takega signala bi narisali enako kombinacijo AND in OR vrat še na izhodu Q_2 , kot kaže naslednja slika:

¹ <http://www.alldatasheet.com/view.jsp?Searchword=74191>



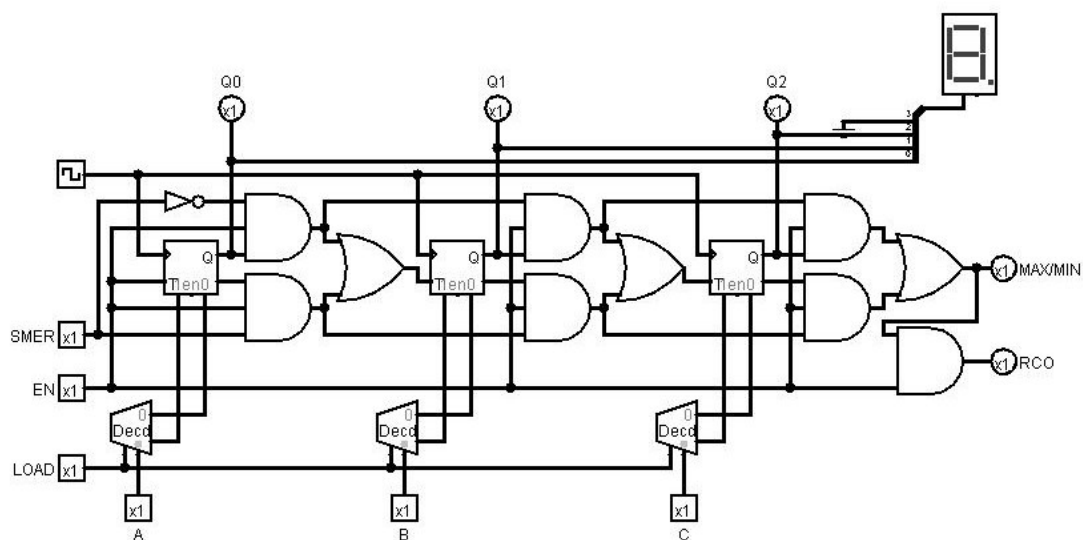
Vezje se nahaja v Logisim predlogah rešenih nalog na domači strani predmeta:
Logisim\counter\counter_up_down_3_bit_using_T_FF_with_enable.circ

Večina števcov je realizirana v 4-bitni zasnovi, tako da glede na vrednost RCO signala ločimo dve skupini števcov:

- desetiški (BCD) števci, katerih RCO se postavi na '1' takrat, ko števec preide iz stanja "1001" v "0000" in
- dvojiški (binarni), katerih RCO se postavi na '1' takrat, ko števec preide iz stanja "1111" v "0000". Več o delovanju RCO najdete v opisu delovanja števcov 74161².

Števču dodamo še signal za asinhrono vzporedno nalaganje (LOAD), ki v aktivnem stanju (LOAD='1') asinhrono nastavi vrednosti T-FF na vrednosti vhodov za nalaganje C, B, A. V števcu 74191³ je izveden z enostavnim dekodeerjem, ki postavi ustrezni vhod za asinhrono postavljanje T-FF (ang. preset) na '1', če je vrednost vhoda za nalaganje '1'. Če je vrednost vhoda za nalaganje '0', potem se '1' postavi na asinhroni vhod za brisanje T-FF (ang. clear).

Vezje se nahaja v Logisim predlogah rešenih nalog na domači strani predmeta:
Logisim\counter\counter_up_down_3_bit_using_T_FF_with_enable_with_load.circ



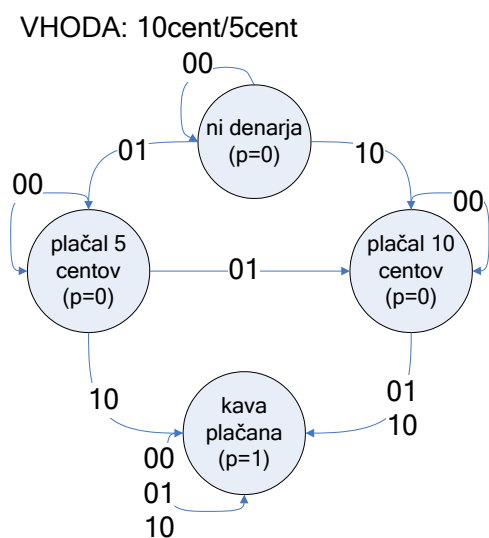
² <http://www.alldatasheet.com/view.jsp?Searchword=74161>

³ <http://www.alldatasheet.com/view.jsp?Searchword=74191>

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. Rezultati bodo objavljeni na domači strani predmeta.

Rešitev 4. naloge:

Moore-ova realizacija avtomata končnih stanj. Opis diagrama stanj:



Na začetku se nahajamo v stanju "*ni denarja*", v katerem je izhod $p=0$. Vhoda v avtomat sta dva: 10cent in 5cent, kar na diagramu kodiramo kot 10cent/5cent. Mehanizem za vnos kovancev preprečuje hkraten vnos dveh kovancev, torej je kombinacija (10cent/5cent=11) nemogoča, zato bo avtomat od tu lahko prešel v poljubno stanje (X). Če uporabnik ni vrgel denarja v avtomat (10cent/5cent=00), potem ostaja v stanju "*ni denarja*". Če uporabnik vrže v avtomat 5 centov (10cent/5cent=01), potem preide v stanje "*plačal 5 centov*". Če uporabnik vrže v avtomat 10 centov

(10cent/5cent=10), potem preide v stanje "*plačal 10 centov*". Ne glede na to koliko je vrgel bo izhod v teh dveh stanjih enak $p=0$, ker še ni plačal celotne cene kave. Če smo v stanju "*plačal 5 centov*" in uporabnik vrže v avtomat 10 centov (10cent/5cent=10), potem preide v stanje "*kava plačana*", kjer postavimo izhod ($p=1$). Stanje "*kava plačana*" je končno in tam tudi ostanemo za vse možne kombinacije. Če smo v stanju "*plačal 10 centov*" in uporabnik vrže v avtomat 5 ali 10 centov (10cent/5cent=10 oz. 01), potem podobno preidemo v stanje "*kava plačana*", kjer postavimo izhod ($p=1$).