

RAZVOJ DIGITALNIH SISTEMOV

2. kolokvij
16. 1. 2014

1. Uporabite ROM vezje za realizacijo naslednjih funkcij:

$$g_1 = x_1 + \overline{x_2} \cdot \overline{x_3} \quad g_2 = \overline{x_1} \cdot x_3 + x_1 \cdot x_2 \quad g_3 = \overline{x_2} \cdot \overline{x_3} + x_1 \cdot x_2 \quad g_4 = \overline{x_2} \cdot x_3 + x_1$$

ROM vezje ima 3 vhodne spremenljivke in 4 bitno vsebino. Povezave oz. 'varovalke' označite s piko (●). Slika ROM je na hrbtni strani.

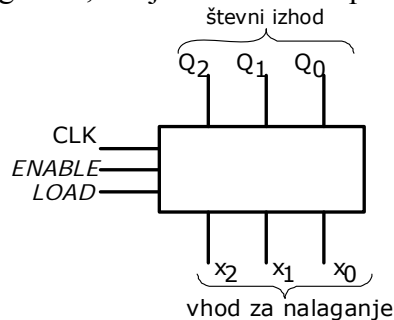
2. Narišite vezje 2-bitnega SIPO (ang. Serial In - Parallel Out) pomikalnega registra z JK-FF in logičnimi vrati. Z izdelanim registrom realizirajte sekvenčno vezje z vhodom x in izhodom y , ki postavi $y=1$, kadar sta dva zaporedna vhoda enaka in različna od enega prej, torej $x(t) = x(t-1) \neq x(t-2)$. Sicer je $y(t) = 0$. V začetnem stanju pri $t=0$ predpostavite, da je vsebina pomikalnega registra enaka '0'.



3. Prikažite sintezo 3-bitnega sinhronnega števca navzgor z omogočanjem štetja (*ENABLE*) in vzporednim nalaganjem (*LOAD*) z D flip-flopi, izbiralniki 2/1 in logičnimi vrati. Logika vseh krmilnih signalov je pozitivna.

S tovrstnim števcem realizirajte števec, ki šteje 2, 3, 4, 5, 2, 3, 4, 5 ...

Uporabite poimenovanje signalov, kot je narisano na spodnji sliki.



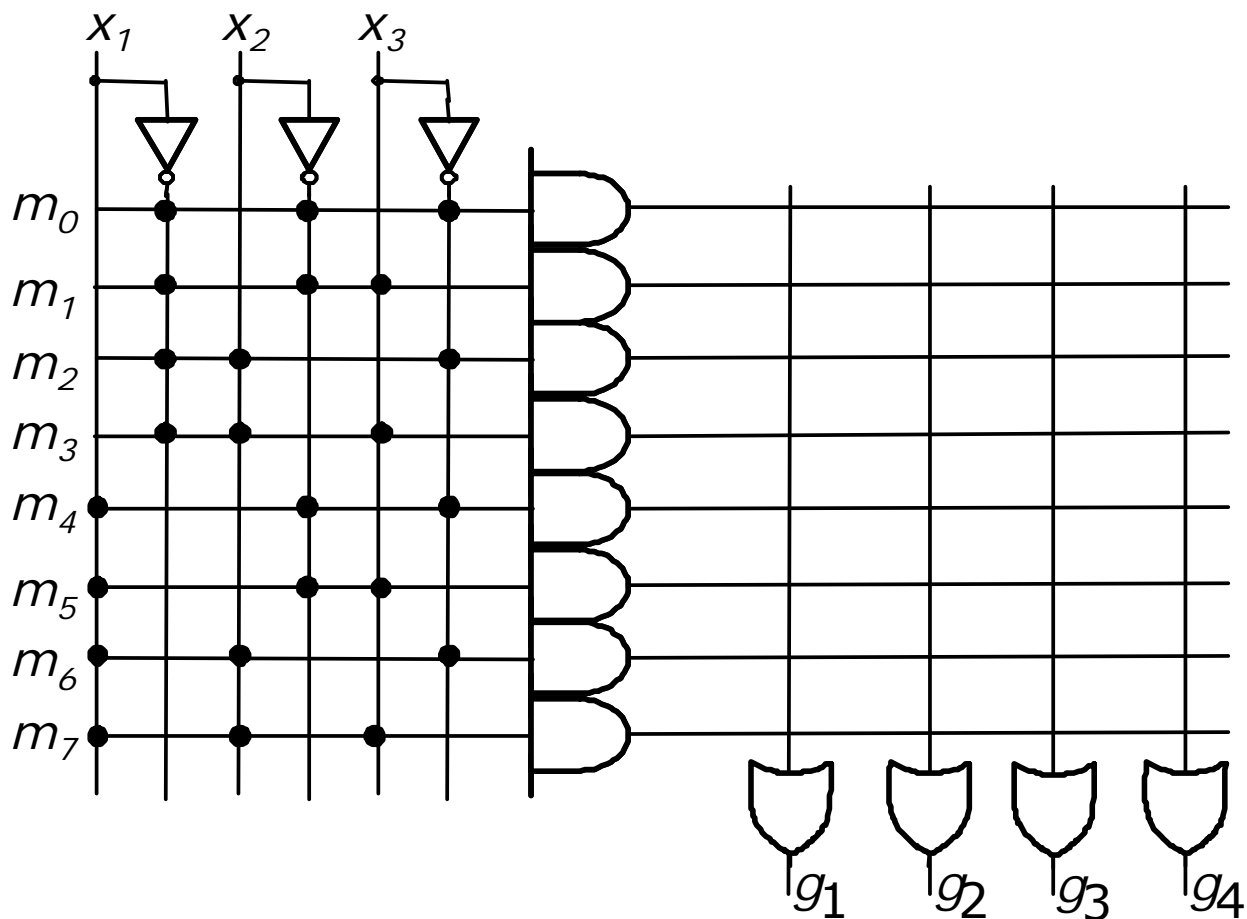
4. Načrtajte tabelo prehajanja stanj Moore-ovega avtomata, ki deluje kot krmilje za kavni avtomat. Kava stane 15 centov, plačujemo pa lahko s kovancema za 5 in 10 centov.

Krmilje ima:

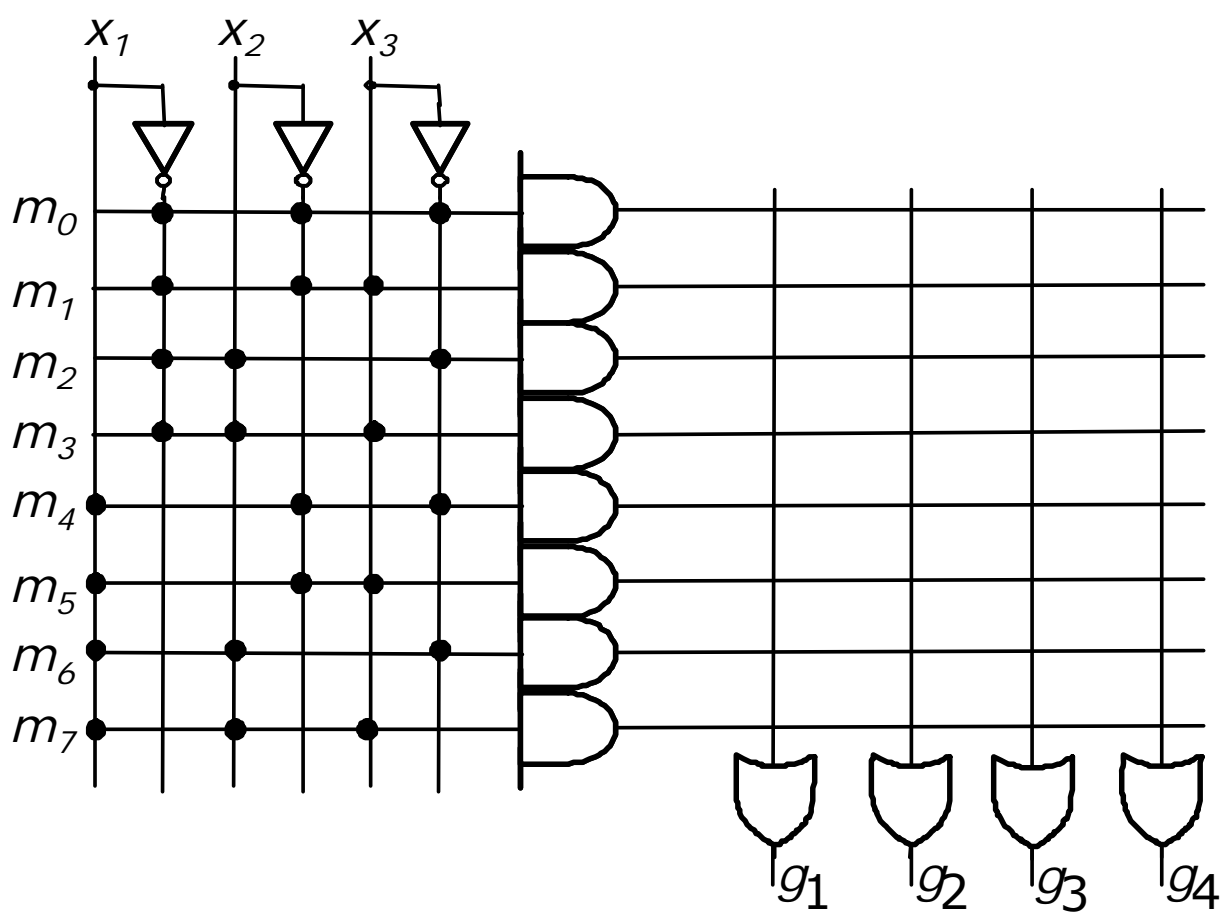
- vhod *5cent*, ki postane '1', ko uporabnik vrže v avtomat kovanec za 5 centov,
- vhod *10cent* ki postane '1', ko uporabnik vrže v avtomat kovanec za 10 centov,
- izhod *p*, ki postane '1', ko uporabnik vrže v avtomat skupno 15 centov.

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 <https://studij.fe.uni-lj.si/>



Če se zmotite, prečrtajte napačno shemo in uporabite drugo shemo!



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 <https://studij.fe.uni-lj.si/>

DEVELOPMENT OF DIGITAL SYSTEMS

Midterm Examination
16. 1. 2014

1. Implement following functions using an imaginary ROM circuit:

$$g_1 = x_1 + \overline{x_2} \cdot \overline{x_3} \quad g_2 = \overline{x_1} \cdot x_3 + x_1 \cdot x_2 \quad g_3 = \overline{x_2} \cdot \overline{x_3} + x_1 \cdot x_2 \quad g_4 = \overline{x_2} \cdot x_3 + x_1$$

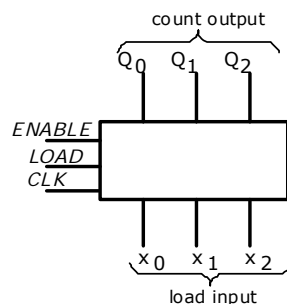
Imaginary ROM circuit has 3 inputs and 4 bit contents as shown on the exam back page. Program the functions by marking programming points with (●) symbol in the OR matrix.

2. Draw the circuit diagram of a two bit SIPO (serial in – parallel out) shift register using JK type flip-flops and logic gates. The register has a serial input (SI) and parallel output (Q_0, Q_1, Q_2).

Design a sequential circuit with an input x and output y using this shift register. The output will become $y='1'$ when two consecutive inputs $x(t), x(t-1)$ are equal, and different from previous input $x(t-2)$: $x(t) = x(t-1) \neq x(t-2)$. Otherwise y equals '0'. Assume that initially the shift register is empty (its entire contents reads '0').



5. Synthesize a 3 bit binary up counter with count enable signal (ENABLE) and parallel load capability (LOAD) using D type flip-flops, multiplexers 2/1 and logic gates: Draw the state transition table, determine D flip-flop input equations and draw the resulting counter circuit using signal names in the figure below. All control signals have positive logic. Using this counter, design a counter which will count in repeating sequence 2, 3, 4, 5, 2, 3, 4, 5

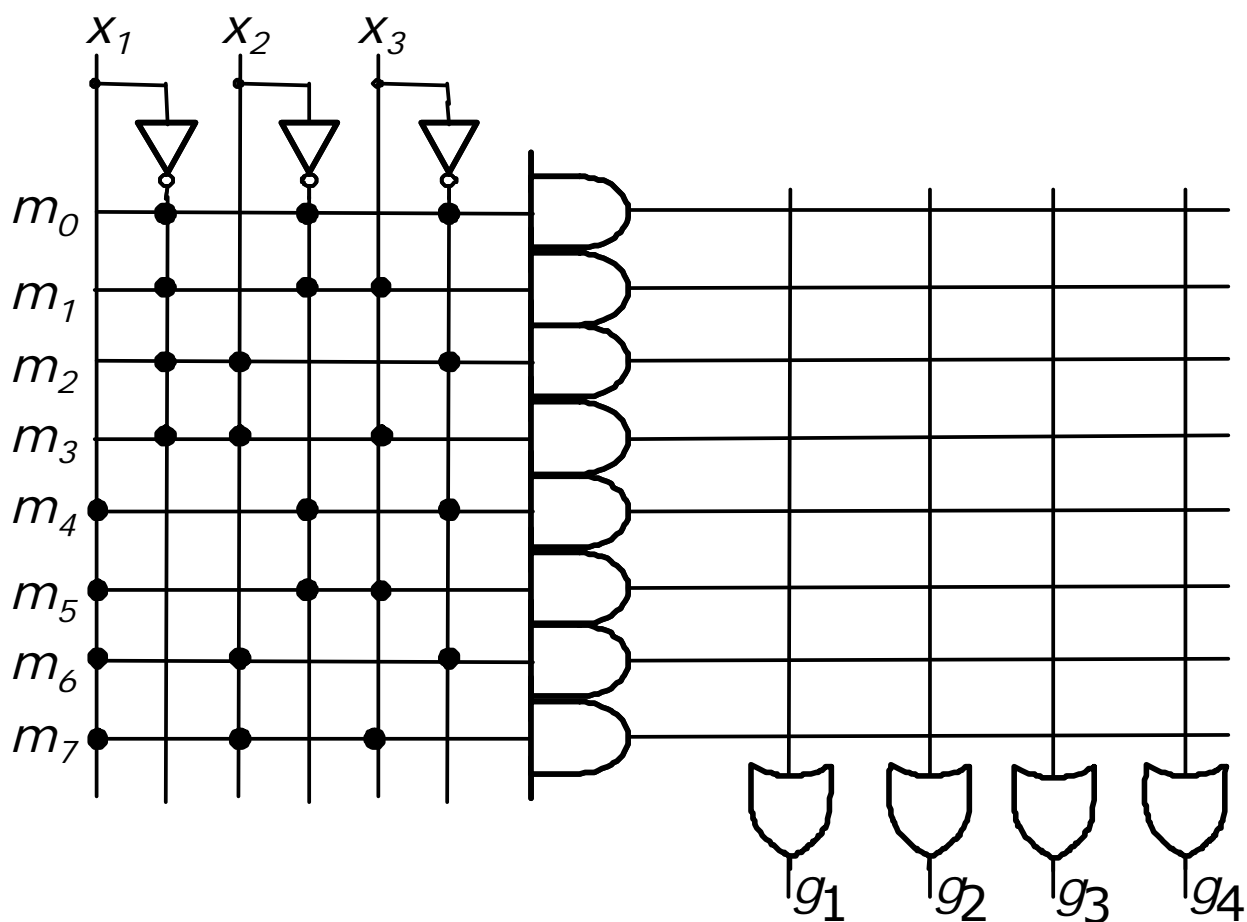


3. Draw a state transition table of a Moore type finite state machine, which controls the operation of a coffee vending machine. The price of coffee is 15 cents. We can pay using 5 or 10 cent coins. The vending machine control circuit has:

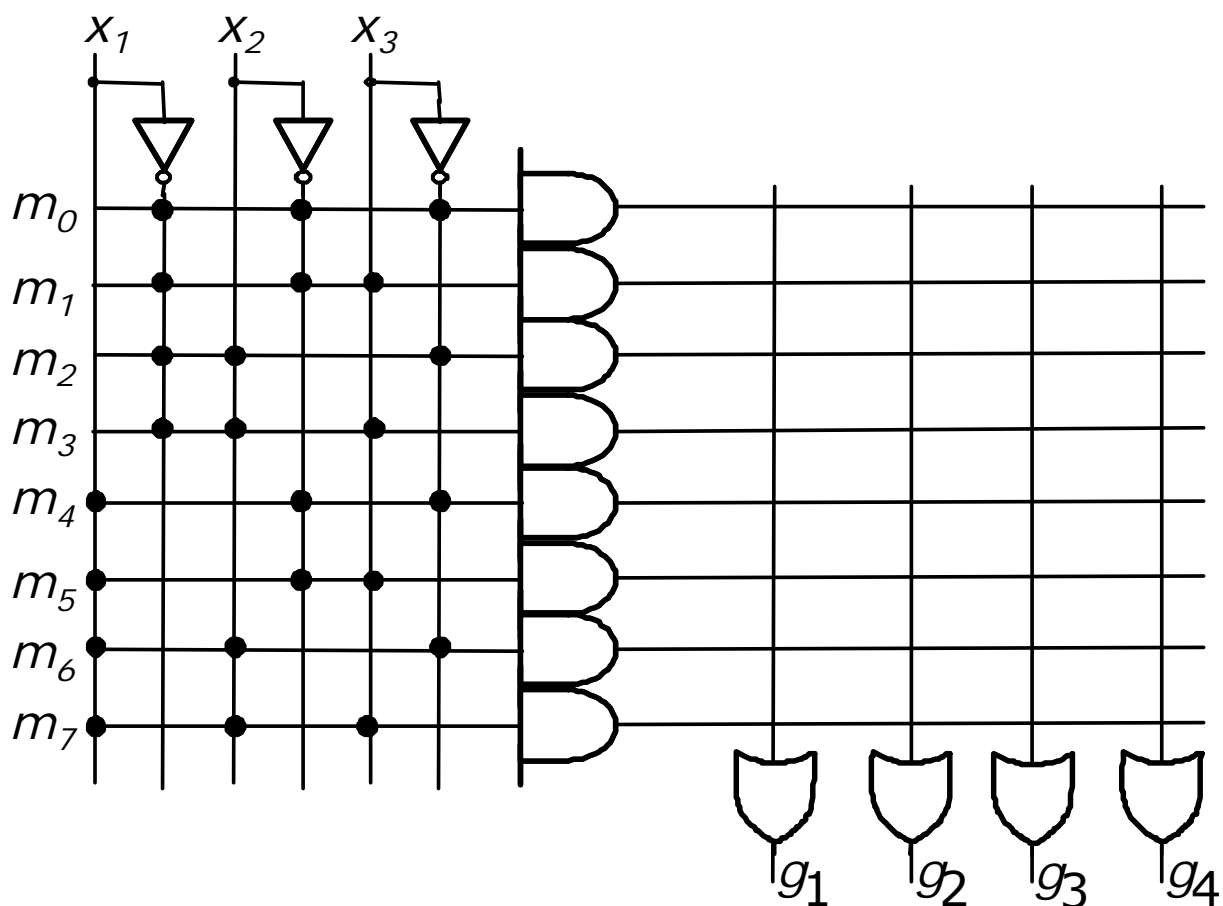
- input *5cent*, which becomes '1', when user inserts a 5 cent coin into machine,
- input *10cent*, which becomes '1', when user inserts a 10 cent coin into machine,
- output *p*, which becomes '1', when user inserts a total sum of 15 cents.

The machine does not return any money and stays in the final state upon payment of 15 cents. Simultaneous insertion of two coins is not possible.

Examination duration is 60 minutes. Each assignment is worth 10 points. Please write your enrollment number and course name on the answer sheet. Grades will be announced on <https://studij.fe.uni-lj.si/>



Upon error, use the other scheme!



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 <https://studij.fe.uni-lj.si/>

Rešitev 1. 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'.

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

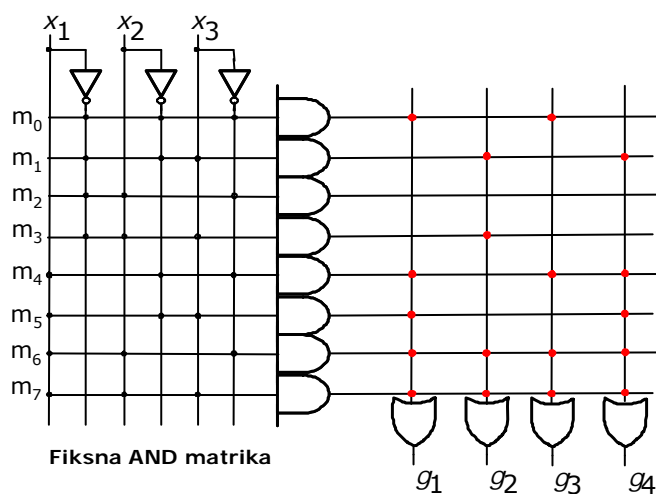
Podobno storimo še za preostale funkcije:

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

$$\begin{aligned}g_3 &= \overline{x_2} \cdot \overline{x_3} + x_1 \cdot x_2 = \\g_3(x_1, x_2, x_3) &= (\overline{x_1} + x_1) \cdot \overline{x_2} \cdot \overline{x_3} + x_1 \cdot x_2 \cdot (\overline{x_3} + x_3) = \\g_3(x_1, x_2, x_3) &= \overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3} + x_1 \cdot \overline{x_2} \cdot \overline{x_3} + x_1 \cdot x_2 \cdot \overline{x_3} + x_1 \cdot x_2 \cdot x_3 \\g_3(x_1, x_2, x_3) &= V(0, 4, 6, 7)\end{aligned}$$

$$\begin{aligned}g_4(x_1, x_2, x_3) &= \overline{x_2} \cdot x_3 + x_1 \\g_4(x_1, x_2, x_3) &= (\overline{x_1} + x_1) \cdot \overline{x_2} \cdot x_3 + x_1 \cdot (\overline{x_2} \cdot \overline{x_3} + x_2 \cdot \overline{x_3} + \overline{x_2} \cdot x_3 + x_2 \cdot x_3) \\g_4(x_1, x_2, x_3) &= (\overline{x_1} + x_1) \cdot \overline{x_2} \cdot x_3 + x_1 \cdot (\overline{x_2} \cdot \overline{x_3} + x_2 \cdot \overline{x_3} + \overline{x_2} \cdot x_3 + x_2 \cdot x_3) \\g_4(x_1, x_2, x_3) &= \overline{x_1} \cdot \overline{x_2} \cdot x_3 + x_1 \cdot \overline{x_2} \cdot \overline{x_3} + x_1 \cdot \overline{x_2} \cdot x_3 + x_1 \cdot x_2 \cdot \overline{x_3} + x_1 \cdot \overline{x_2} \cdot x_3 + x_1 \cdot x_2 \cdot x_3 \\g_4(x_1, x_2, x_3) &= V(1, 5, 4, 6, 7)\end{aligned}$$

PDNO je najprimernejša oblika za realizacijo z ROM, ker je matrika AND fiksna. Programirane vrednosti AND matrike predstavljajo vse minterme funkcije treh spremenljivk ($x_1x_2x_3$) 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. Rezultati bodo objavljeni na <https://studij.fe.uni-lj.si>

Vsebino ROM pomnilnika običajno podajamo s tabelo v kateri programirane povezave (●) v OR matriki pišemo kot '1' na danem mestu vsebine.

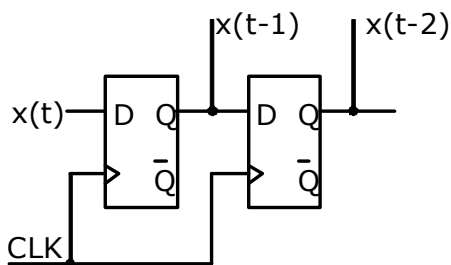
<i>Minterm</i> $g_i(x_1x_2x_3)$	<i>Naslov lokacije</i> $x_1x_2x_3$	<i>Vsebina</i> <i>lokacije</i> $g_1g_2g_3g_4$
m_0	000_2	1010_2
m_1	001_2	0101_2
m_2	010_2	0000_2
m_3	011_2	0100_2
m_4	100_2	1011_2
m_5	101_2	1001_2
m_6	110_2	1111_2
m_7	111_2	1111_2

Realni ROM elementi imajo 8 bitne podatke, zloge ali včasih oktete (ang. *byte*, *octet*). Vsebina ROM elementov se podaja v datoteki, ki jo nato programiramo z posebnim inštrumentom (ROM programatorjem).

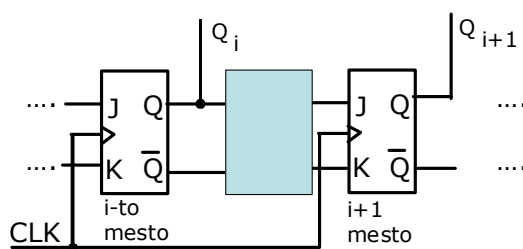
Najenostavnejši način podajanja zapisa vsebine ROM elementa je v surovi dvojiški obliki (ang. *raw binary file*) v kateri si 8 bitni podatki sledijo zapisani v dvojiški obliki. Kompleksnejša zapisa podajanja vsebine ROM s tabelo sta INTEL šestnajstiška oblika (ang. [*Intel hex record*](#)) in Motorola SREC oblika (ang. [*Motorola S record*](#)).

Rešitev 2. naloge:

Pomnjenje preteklih vrednosti vhoda x lahko izvedemo s pomočjo pomikalnega registra.

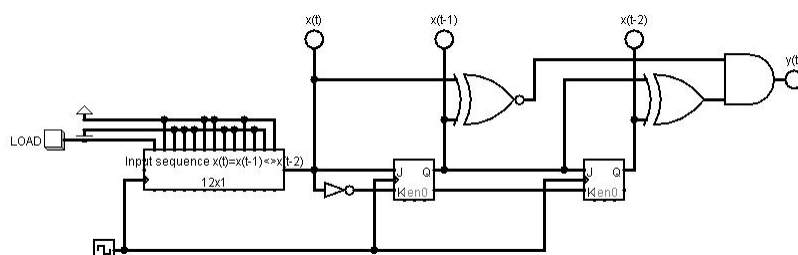


Zapomniti si moramo trenutno vrednost $x(t)$ ter še dve prejšnji $x(t-1)$ in $x(t-2)$. Za pomnjenje dveh preteklih vrednosti bomo načrtali 2-bitni pomikalni register. Zapišemo tabelo prehajanja stanj z i -tega mesta na $(i+1)$ mesto. Enačba izhoda za $(i+1)$ mesto pomikalnega registra je enačba D-FF ($Q_{i+1}(t+1) = D_{i+1}$), saj je pomikalni register običajno izveden kot veriga D-FF, pri katerih je izhod prejšnjega FF vezan na vhod naslednjega. Ena možnost realizacije je realizacija D-FF z uporabo JK-FF. Ta možnost je potratna, zato raje zapišemo vzbujalno tabelo za opazovano mesto in določimo vhode glede na spremembo stanja ($Q_{i+1}(t) \rightarrow Q_{i+1}(t+1)$) na $(i+1)$ mestu.



Trenutna vsebina registra		Vsebinska registra ob pomiku	Vhoda FF na mestu $i+1$		Pomen stanja
$Q_i(t)$	$Q_{i+1}(t)$	$Q_{i+1}(t+1)$	J_{i+1}	K_{i+1}	
0	0	0	0	X	RESET/HOLD
0	1	0	X	1	INVERT/RESET
1	0	1	1	X	INVERT/SET
1	1	1	X	0	SET/HOLD

Če v dobljenih vseh za mesto $(i+1)$ izberemo primerne redundance, dobimo vhoda $J_{i+1} = Q_i(t)$ in $K_{i+1} = Q_i(t)'$. To velja tudi za vhodno mesto, torej bomo na vhodni JK-FF vezali vhod $J_0 = x(t)$ in $K_0 = x(t)'$. Primerjavo neenakosti mest izvedemo z XOR vrati ($x(t) \oplus x(t-1)$), medtem ko primerjavo enakosti izvedemo z XNOR vrati ($x(t-1) \oplus x(t-2)$). Veljati morata oba pogoja, kar izvedemo z AND vrati.



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 <https://studij.fe.uni-lj.si>

Vezje generatorja je v predlogah avditornih vaj na domači strani predmeta:
Logisim\shift_reg\sequence_detector_x(t)_is_x(t_1)_is_not_x(t_2).circ

Seveda poleg take rešitve vedno obstaja klasični pristop reševanja s teorijo avtomatov, vendar smo pri tem precej omejeni. Čim bi morali primerjati dlje v zgodovino npr. $x(t-4)$, $x(t-5)$... klasična teorija avtomatov hitro pripelje do minimizacije v Veitch–evih diagramih 5, 6 ... spremenljivk, kar je dolgotrajen proces.

<i>Vhod in trenutno stanje</i>			<i>Naslednje stanje (ob pomiku)</i>		<i>Vhodi JK–FF</i>			
x	$Q_1(t)$	$Q_2(t)$	$Q_1(t+1)$	$Q_2(t+1)$	J_1	K_1	J_2	K_2
0	0	0	0	0	0	X	0	X
0	0	1	0	0	0	X	X	1
0	1	0	0	1	X	1	1	X
0	1	1	0	1	X	1	X	0
1	0	0	1	0	1	X	0	X
1	0	1	1	0	1	X	X	1
1	1	0	1	1	X	0	1	X
1	1	1	1	1	X	0	X	0

Če primerno izberemo redundance, niti ni potrebno risati Veitch–evih diagramov, ampak lahko dobimo enako rešitev kot pri realizaciji s pomočjo pomikalnega registra:

$$J_1 = x \quad K_1 = \bar{x}$$

$$J_2 = Q_1(t) \quad K_2 = \overline{Q_1(t)}$$

Primerjavo na izhodu tako dobljenega registra (x , Q_1 , Q_2) v splošnem izvedemo s primerjalnikom enakosti (ang. equality comparator), realiziranega z XOR oz. z XNOR vrati.

Rešitev 3. naloge:

trenutno stanje			naslednje stanje			D-FF		
Q ₂	Q ₁	Q ₀	Q ₂	Q ₁	Q ₀	D ₂	D ₁	D ₀
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	0
0	1	0	0	1	1	0	1	1
0	1	1	1	0	0	1	0	0
1	0	0	1	0	1	1	0	1
1	0	1	1	1	0	1	1	0
1	1	0	1	1	1	1	1	1
1	1	1	0	0	0	0	0	0

Podobno za D₂ narišemo Veitchev diagram

D_2 :

	Q_2			
Q_1	1	0	1	0
	1	1	0	0
	Q_0			

Za D₂ sledi:

$$D_2 = Q_2 Q_1' + Q_2 Q_0' + Q_2' Q_1 Q_0$$

Iz tabele prehajanja stanj števca določimo enačbe D-FF:

Za D₀ se iz tabele vidi $D_0 = Q_0'$

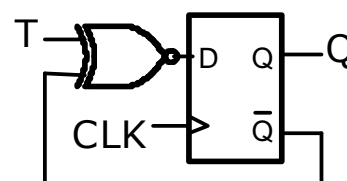
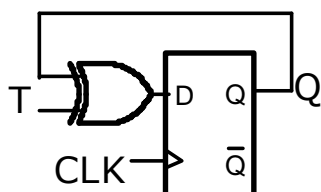
Za D₁ narišemo Veitchev diagram:

D_1 :

	Q_2			
Q_1	1	0	0	1
	0	1	1	0
	Q_0			

$$D_1 = Q_0 \oplus Q_1$$

Če enačbo $D_0 = Q_0'$ zapišemo kot $D_0 = 1 \oplus Q_0$ in jo narišemo v vezju, smo pravzaprav realizirali T-FF s pomočjo D-FF, kot kaže levi del spodnje slike:

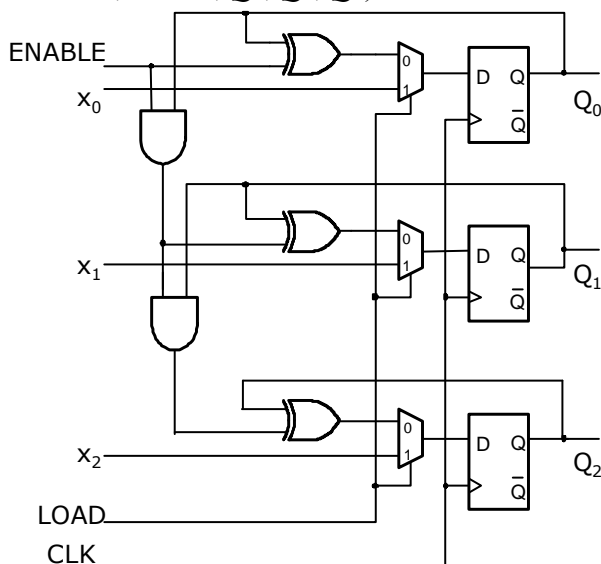


Slika 1: Realizacija T-FF s pomočjo D-FF in XOR vrat (levo) in XNOR vrat (desno)

Naloga pravi, da moramo izdelati števec, ki ima vhod za omogočanje štetja (*ENABLE*): Če prvemu "T-FF" (D-FF z XOR vrati) postavimo vhod $T_0 = 0'$ namesto $T_0 = 1'$, vsi FF ne bodo štel, ampak bodo ohranjali stanje. Torej, če na vhod T_0 postavimo zunanji signal *ENABLE*, števec ne bo štel, ampak ohranjal stanje, če bo *ENABLE* = '0'. V verigi sinhronnega števca so namreč vsi FF vezani tako, da so odvisni od prvega FF. Če stanje ohranja prvi, ga bodo tudi vsi ostali.

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 <https://studij.fe.uni-lj.si>

Za realizacijo signala za vzporedno nalaganje pa izkoristimo osnovno lastnost D–FF (pomnjenje). To storimo tako, da na vhod vsakega D–FF postavimo 2/1 izbiralnik, s katerim določimo, ali se bo dana informacija vpisala s števnege vhoda ali preko zunanjih priključkov. Do iste realizacije bi prišli, če bi v osnovni analizi upoštevali ta dva krmilna signala – analiza je veliko bolj zapletena, saj vsebuje Veitcheve diagrame za 5 spremenljivk ($ENABLE$, $LOAD$, Q_2 , Q_1 , Q_0).



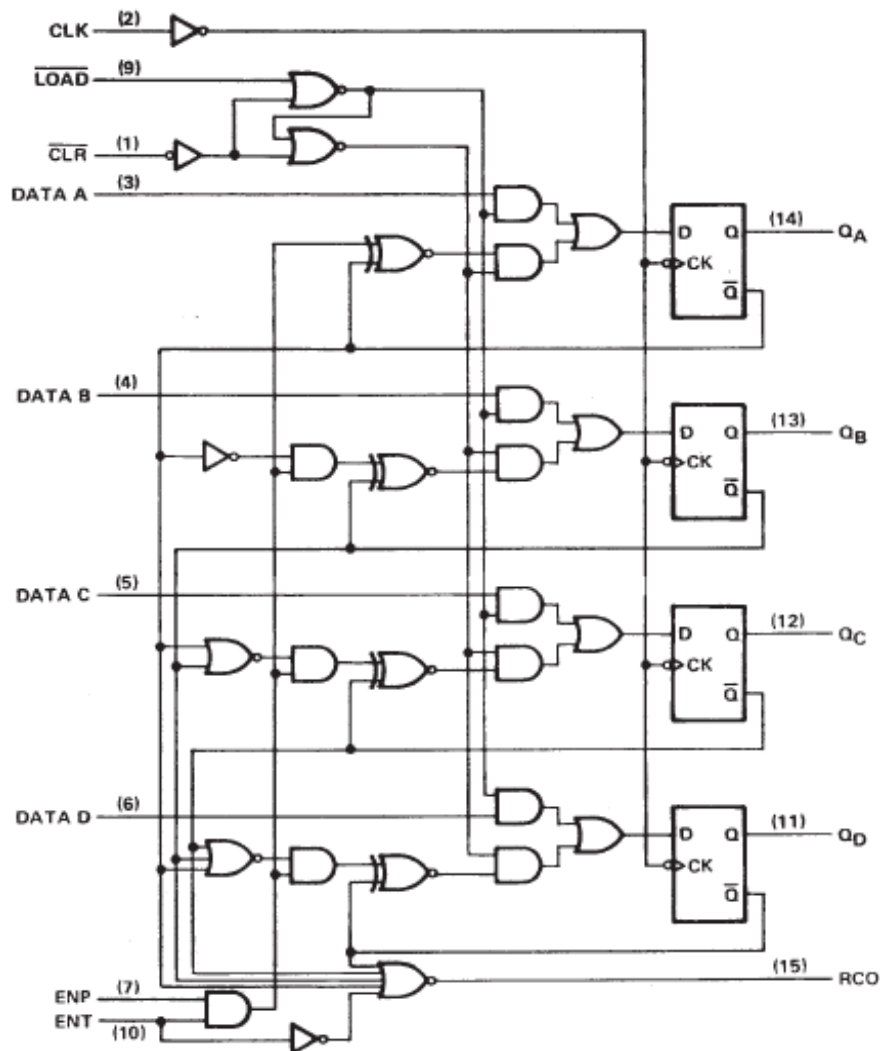
Slika 2: Sinhroni števec z vzporednim nalaganjem (LOAD) in omogočanjem štetja (ENABLE) (3-bitna izvedba).

Če želimo z nastalim števcem šteti naraščajoče 2, 3, 4, 5 ..., moramo ko števec pride do stanja 5 ($Q_2Q_1Q_0=101_2$) števec postaviti nazaj na stanje v stanje 2 ($Q_2Q_1Q_0=x_2x_1x_0=010_2$), torej signal $LOAD$ dekodiramo s pomočjo 2-vhodnih AND vrat. Pomembno pri tem je, da se pri dekodiranju zavedamo, da se $Q_2='1'$ in $Q_0='1'$ v števeni sekvenci pojavlja samo enkrat – če bi se večkrat, bi morali dekodirati tudi Q_1 . Pri tovrstnih števcih ponavadi uporabljamo še en signal $RESET$, s katerim postavimo števec v začetno stanje, kar dosežemo tako, da na vhod D–FF za brisanje asinhrono priključimo signal $RESET$.

Nastali strukturi števca bi lahko na isti način dodali še četrti bit. Tako bi dobili podobno strukturo kot je 4-bitni sinhroni števec z vzporednim nalaganjem 74163¹. Pri spodnji realizaciji tega vezja so uporabljeni D–FF, proženi na negativni rob signala ure (CLK). Štetje omogočimo s signaloma ENP , ENT (ang. enable parallel, enable transfer). Štetje je izvedeno tako, da se D–FF spremenijo v T–FF, kar dosežemo s pomočjo XNOR vrat, ki imajo en vhod vezan na števeni signal ($ENT=ENP='1'$), drug vhod pa na izhod Q' FF. AND vrata pred XNOR zagotavljajo prenehanje štetja, če velja $ENT \cdot ENP \neq '1'$. Na vhodu D–FF je vezan enostaven izbiralnik iz dveh AND vrat, vezanih na OR vrata. Ta izbiralnik določa, ali bo števec štel, ali bo vzporedno naložil vrednost. Zgornja AND vrata izbiralnika so krmiljena s signalom $LOAD \cdot CLR'$, spodnja pa z $LOAD' \cdot CLR'$. Čim velja, da je $CLR'='1'$ in $LOAD'='0'$, se v D–FF asinhrono naloži vsebina na vseh Q_D Q_C Q_B $Q_A=DATA_D$ $DATA_C$ $DATA_B$ $DATA_A$, medtem ko dokler velja $LOAD'='1'$ in $CLR'='1'$, bo števec štel navzgor. Če je $CLR'='0'$, je na obeh vseh AND vrat izbiralnika '0' in stanje vseh D–FF se asinhrono postavi na $Q_DQ_CQ_BQ_A="0000"$. Števeni izhodi so Q_D Q_C Q_B Q_A .

¹ <http://www.alldatasheet.com/view.jsp?Searchword=74163>

Izhod *RCO* (*ripple carry out*) se postavi na '1', ko je števec v stanju "1111" ob tem, da je $ENT='1'$. Signal *RCO* je izveden z NOR vrati na vhode katerih so priključeni negirani izhodi D-FF, kar je po De Morgan-ovem teoremu ekvivalent AND vratom.

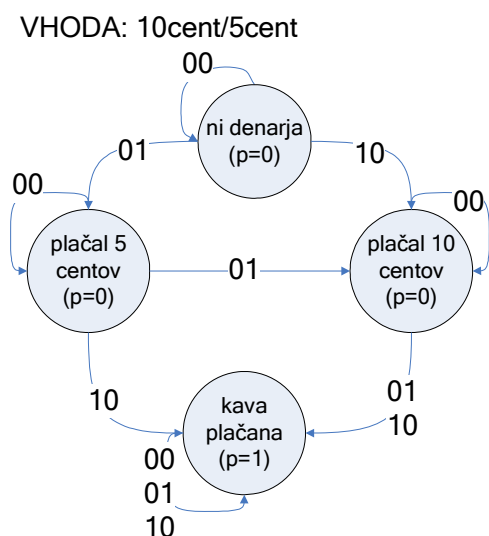


Slika 3: Struktura 4-bitnega MSI sinhronnega števca z vzporednim nalaganjem (74163).

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 ($p=1$).

Naredimo tabelo prehajanja stanj:

trenutno stanje	10cent	5cent	naslednje stanje	izhod p
ni denarja	0	0	ni denarja	0
ni denarja	0	1	plačal 5 centov	0
ni denarja	1	0	plačal 10 centov	0
ni denarja	1	1	X	X
plačal 5 centov	0	0	plačal 5 centov	0
plačal 5 centov	0	1	plačal 10 centov	0
plačal 5 centov	1	0	kava plačana	0
plačal 5 centov	1	1	X	X
plačal 10 centov	0	0	plačal 10 centov	0
plačal 10 centov	0	1	kava plačana	0
plačal 10 centov	1	0	kava plačana	0
plačal 10 centov	1	1	X	X
kava plačana	0	0	kava plačana	1
kava plačana	0	1	kava plačana	1
kava plačana	1	0	kava plačana	1
kava plačana	1	1	X	X

Izberemo kodiranje stanj:

stanje	Q_1	Q_0
ni denarja	0	0
plačal 5 centov	0	1
plačal 10 centov	1	0
kava plačana	1	1

Nad tabelo prehajanja stanj uporabimo predlagano kodiranje stanj:

Q_1	Q_0	10cent	5cent	Q_1	Q_0	izhod P
0	0	0	0	0	0	0
0	0	0	1	0	1	0
0	0	1	0	1	0	0
0	0	1	1	X	X	X
0	1	0	0	0	1	0
0	1	0	1	1	0	0
0	1	1	0	1	1	0
0	1	1	1	X	X	X
1	0	0	0	1	0	0
1	0	0	1	1	1	0
1	0	1	0	1	1	0
1	0	1	1	X	X	X
1	1	0	0	1	1	1
1	1	0	1	1	1	1
1	1	1	0	1	1	1
1	1	1	1	X	X	X

Naloga zahteva realizacijo z D–FF:

t				t+1				
Q_1	Q_0	10cent	5cent	Q_1	Q_0	D_1	D_0	izhod P
0	0	0	0	0	0	0	0	0
0	0	0	1	0	1	0	1	0
0	0	1	0	1	0	1	0	0
0	0	1	1	X	X	X	X	X
0	1	0	0	0	1	0	1	0
0	1	0	1	1	0	1	0	0
0	1	1	0	1	1	1	1	0
0	1	1	1	X	X	X	X	X
1	0	0	0	1	0	1	0	0
1	0	0	1	1	1	1	1	0
1	0	1	0	1	1	1	1	0
1	0	1	1	X	X	X	X	X
1	1	0	0	1	1	1	1	1

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 <http://estudent.fri.uni-lj.si/fe.html>

1	1	0	1	1	1	1	1	1
1	1	1	0	1	1	1	1	1
1	1	1	1	X	X	X	X	X

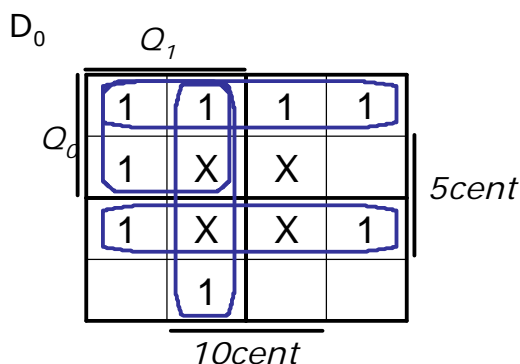
Iz dobljene tabele narišemo Veitch–eve diagrame za oba D–FF in izhod p:

$$D_0 = V(1, 4, 6, 9, 10, 12-14) \text{ in } Vx(3, 7, 11, 15)$$

$$D_1 = V(2, 5, 6, 8, 9, 10, 12-14) \text{ in } Vx(3, 7, 11, 15)$$

$$p = V(12-14) \text{ in } Vx(3, 7, 11, 15)$$

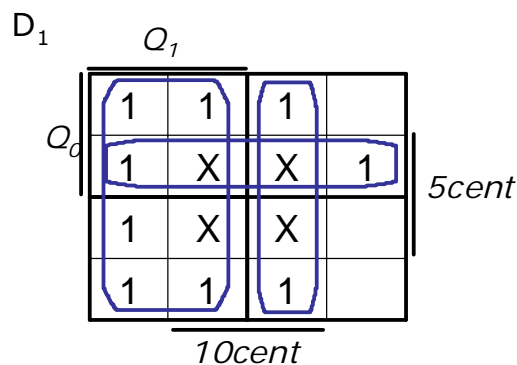
Veitch–ev diagram za D_0 :



$$D_0 = Q_1 \cdot Q_0 + \overline{5cent} \cdot Q_0 + 10cent \cdot Q_1 + 5cent \cdot \overline{Q_0}$$

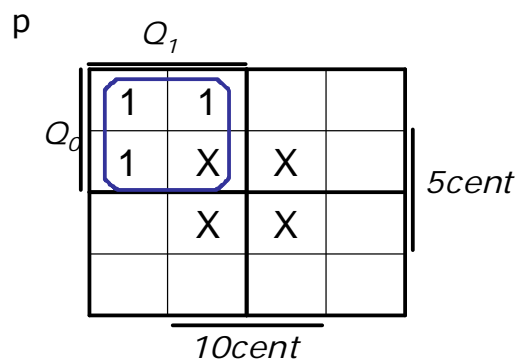
$$D_0 = Q_1 \cdot (Q_0 + 10cent) + 5cent \oplus Q_0$$

Veitch–ev diagram za D_1 :



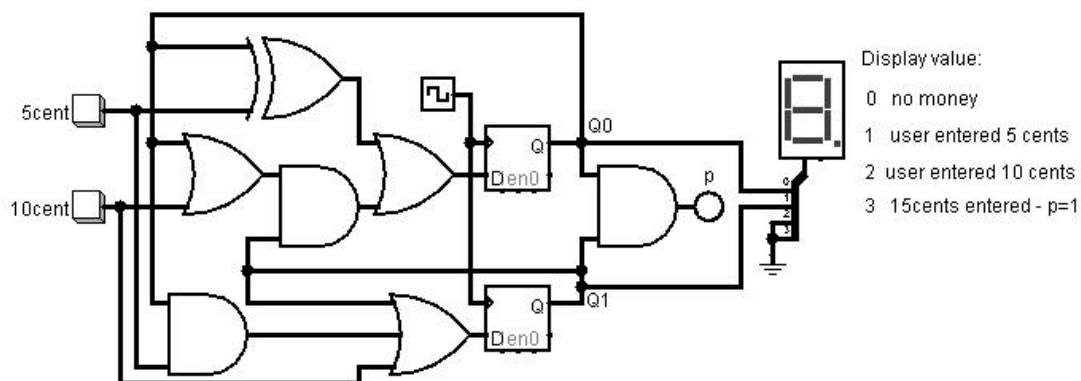
$$D_1 = Q_1 + 5cent \cdot Q_0 + 10cent$$

Veitch–ev diagram za izhod p:



$$p = Q_1 \cdot Q_0$$

Enačbo za izhod p bi lahko napisali tudi samo s sklepanjem, saj se izhod p postavi samo, ko je avtomat v stanju "kava plačana", ki ima kodo $Q_1Q_0="11"$ – torej ko bosta Q_1 in Q_0 enaka '1', bo izhod $p=1$.



Vezje se nahaja v Logisim predlogah rešenih nalog na domači strani predmeta: `Logisim\fsm\Vending_machine_d_ff_5_10_cents_price_15cents.circ`