

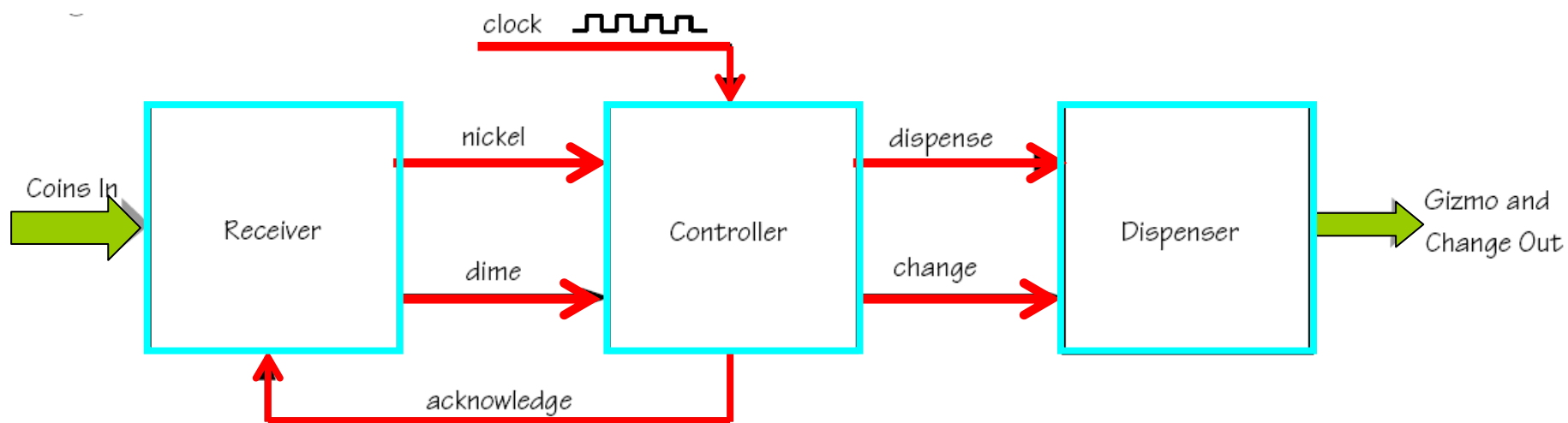
VTŠ: Osnovi računarske tehnike

Dijagrami, tabele i mašine stanja

mr. Veličković Zoran
April, 2010.

Dijagrami, tabele i mašine stanja (1)

- ❑ Zamislite mašinu koja radi na metalne novčiće od 5 (nickel) i 10 (dime) centi.
- ❑ Kada se ubace metalni novčići čiji je iznos 15 centi puni se željeni napitak (gizmo).
- ❑ Osnovni blokovi ove mašine su prijemnik (*receiver*) novca, kontroler (*controller*) koji nadgleda čitavo funkcionisanje i dozer napitka (*dispenser*) .



Dijagrami, tabele i mašine stanja (2)

- ❑ Veze na dijagramu sa imenima nickel, dime, dispense, change, i acknowledge predstavljaju digitalne signale koji mogu imati vrednosti 0 i 1.
- ❑ Novčići od 5 i 10 centi se mogu ubacivati proizvoljnim redom jedan po jedan.
- ❑ Kada su novčići ubačeni, prijemnik određuje tip novčića i generiše - postavlja signal nickel ili dime na logičku 1.
- ❑ Čitava operacija je sinhronizovana clock signalom.
- ❑ Na rastuću ivicu clock-a, kontroler ispituje ulaze nickel i dime da vidi da li je novčić ubačen.

Dijagrami, tabele i mašine stanja (3)

- ❑ Kontroler vodi računa o ubačenom iznosu i određuje koja je sledeća akcija koja će biti obavljena.
- ❑ Svaki put kada kontroler ispituje ulaze on šalje signal potvrde (acknowledge) nazad prijemniku.
- ❑ Ovi signali potvrde informišu prijemnik da su novčići uzeti u obzir dok prijemnik odgovara resetovanjem signala nickel i dime na 0 i čeka sledeći novčić.
- ❑ Kada kontroler zaključi da je ubačeno dovoljno novca, informiše dozer da se napitak natoči i promeni stanje signala dispense i change na 1.

Dijagram stanja kontrolera

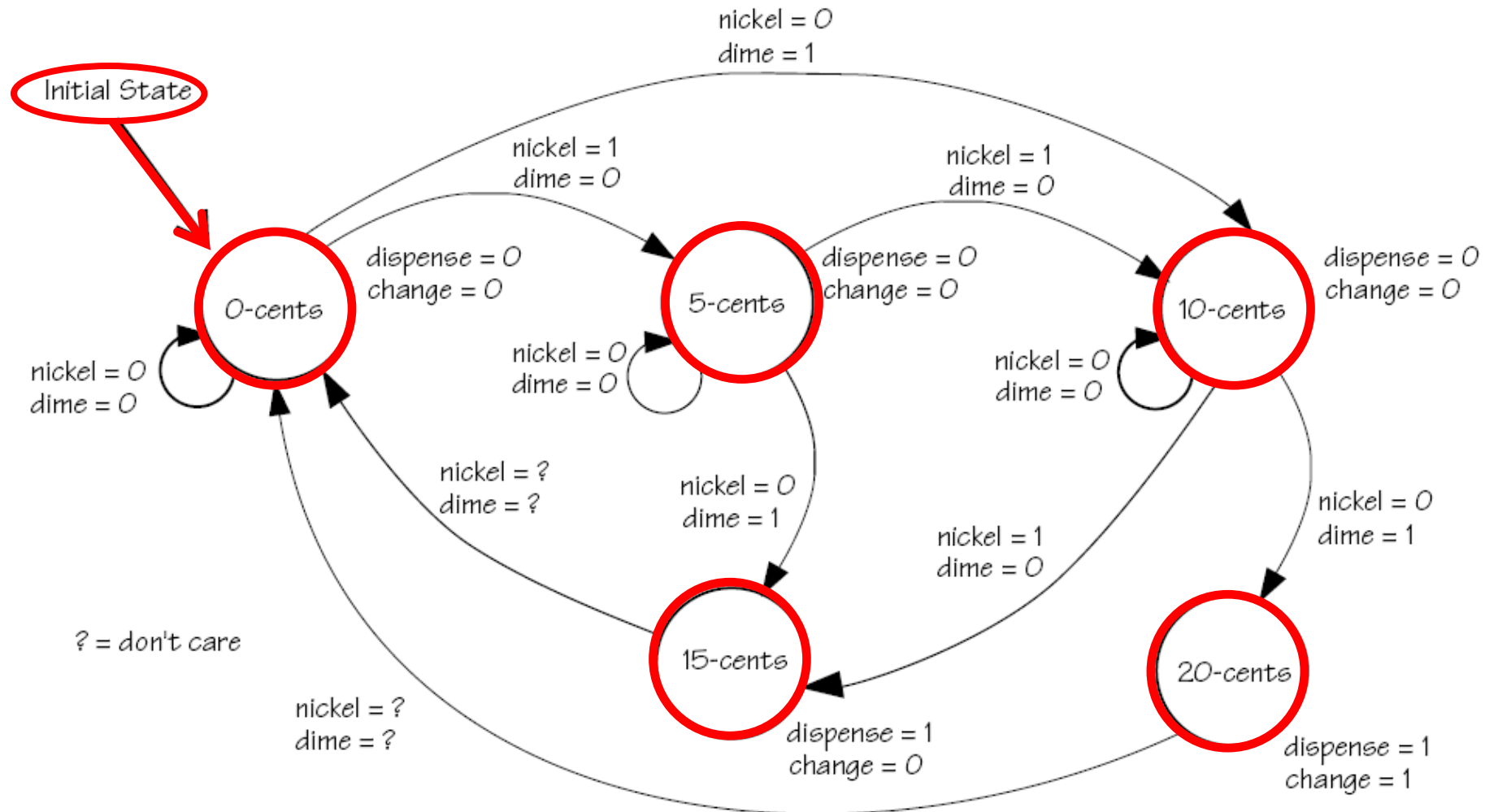


Tabela stanja kontrolera

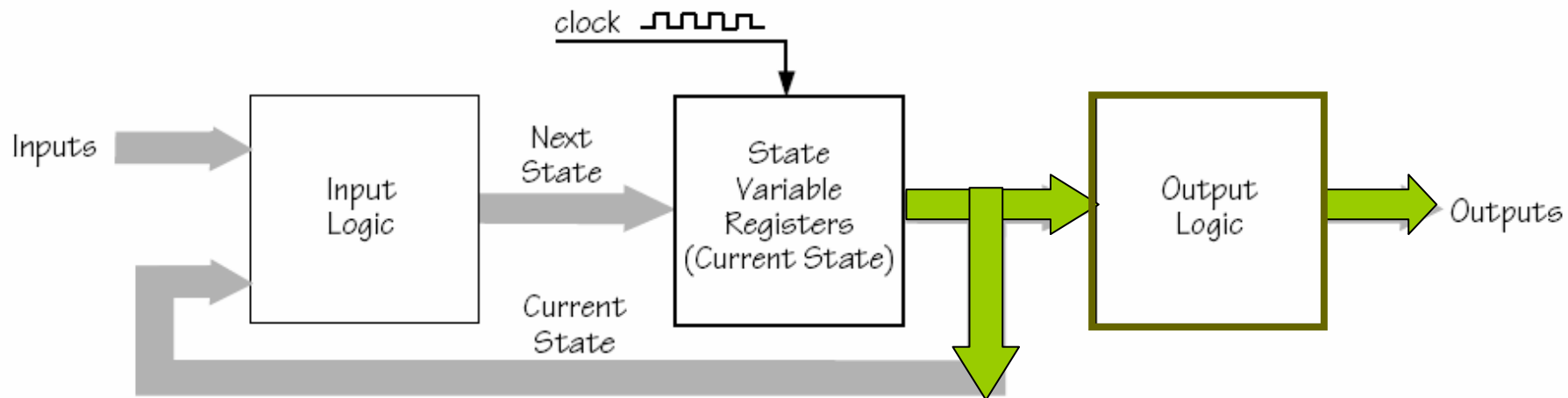
Current state	clock	nickel	dime	dispense change	Next state	
0-cents	↑	0	0	0	0	0-cents
0-cents	↑	1	0	0	0	5-cents
0-cents	↑	0	1	0	0	10-cents
5-cents	↑	0	0	0	0	5-cents
5-cents	↑	1	0	0	0	10-cents
5-cents	↑	0	1	0	0	15-cents
10-cents	↑	0	0	0	0	10-cents
10-cents	↑	1	0	0	0	15-cents
10-cents	↑	0	1	0	0	20-cents
15-cents	↑	?	?	1	0	0-cents
20-cents	↑	?	?	1	1	0-cents

Mašine stanja (1)

- ❑ Implementacija funkcija u kontroleru naziva se **mašina stanja** (*state machine*).
- ❑ Ako je broj stanja konačan realizacija se naziva **mašina sa konačnim brojem stanja** (*finite state machine* - **FSM**).
- ❑ FSM-a se realizuje pomoću **skupa registara** poznatih pod nazivom **promenljive stanja** (*state variables*).
- ❑ Svako stanje (0-centa, 5-centa, 10-centa) je označeno kao jedinstveni binarni uzorak (patern) sačinjen od **0** i **1**.
- ❑ Uzorak (patern) reprezentuje **trenutno stanje** (*current state*) koji je smešten u **promenljive stanja** (*state variables*).

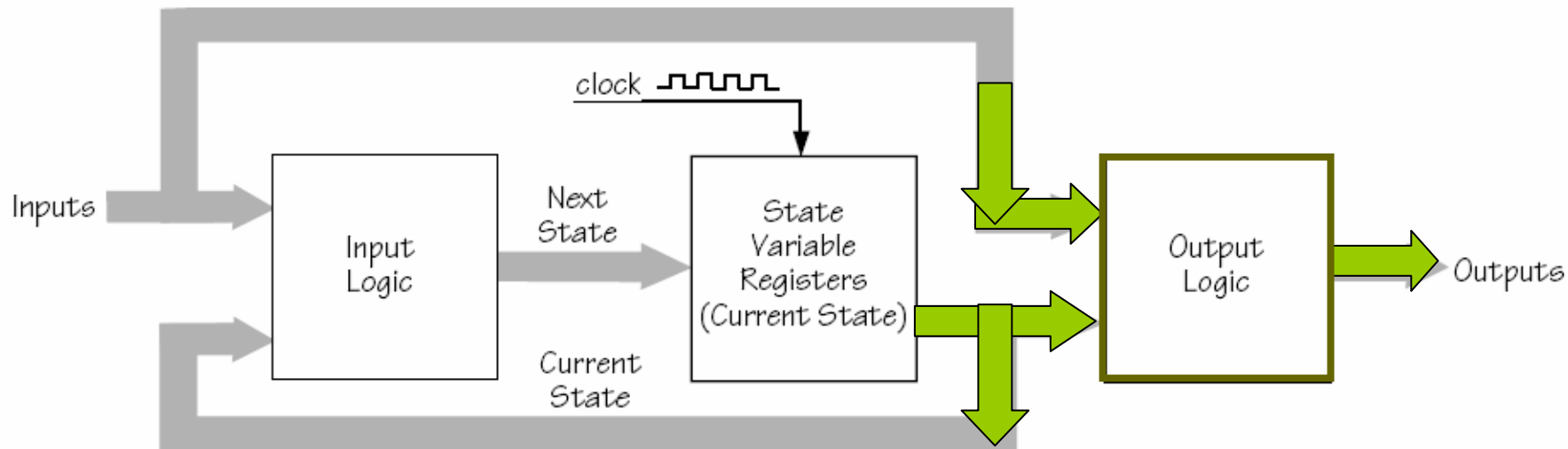
Murova mašina stanja

- ❑ Kod Murove mašine izlazi su **jedino** izvedeni iz vrednosti koje imaju promenljive stanja.
- ❑ Funkcija kontrolera je zapravo klasa primera Moore mašine.



Melay-eva mašina stanja

- Kod Mealy-eve mašine izlazi mogu biti generisani kombinacijom vrednosti iz promenljivih stanja i jednog ili više ulaza.



Mašine stanja (2)

- ❑ Kod obe realizacije ulazna logika se sastoji od primitivnih logičkih kola AND, NAND, OR, and NOR.
- ❑ Ova logička kola formiraju vrednosti ulaza i **zajedno** sa vrednošću trenutnog stanja određuju sledeće stanje (*next state*).
- ❑ Vrednost next uzorka će biti postavljeno na sledećoj rastućoj ivici clock-a.
- ❑ Izlazna logika (output logic) se takođe sastoji od standardnih primitivnih logičkih kola koja formiraju odgovarajuće vrednosti na izlazima iz trenutnog stanja zapamćenog u promenljivama stanja.

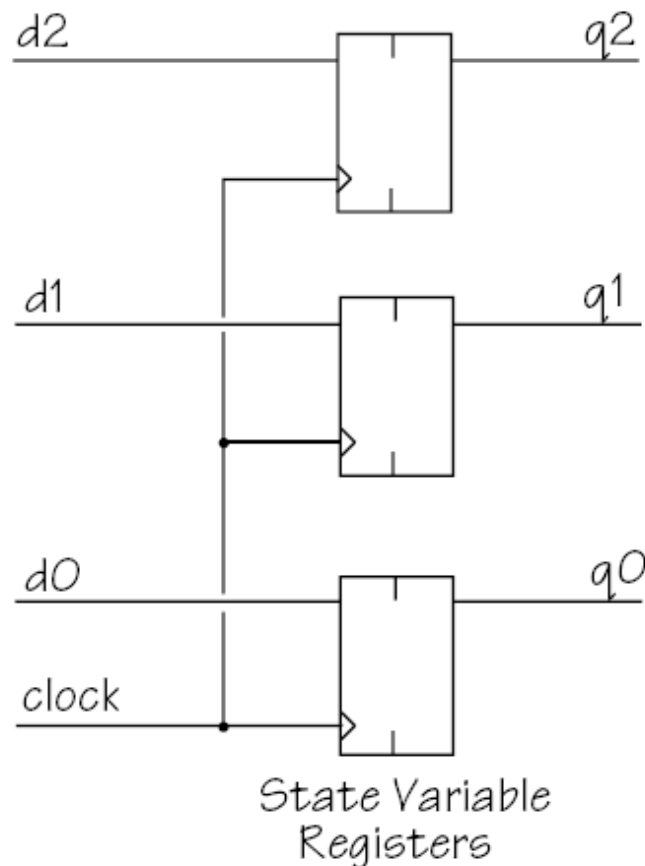
Označavanje stanja (1)

- ❑ Ključna stvar kod realizacije mašina je određivanje (označavanje) stanja mašina koja odgovaraju procesima prema kojima su i definisana i označena binarnim brojevima koji će biti zapamćeni u promenljivama stanja.
- ❑ Pogledaj tabelu stanja kontrolera, kako kodirati pojedina stanja mašine?
- ❑ Označavanje stanja zahteva minimalni broj registara i naziva se **binarno kodovanje** (*binary encoding*).
- ❑ Svaki registar može imati samo jedan binarni broj, tako da se može obeležiti samo sa 0 ili 1.

Označavanje stanja (2)

- ❑ Dva registra se mogu predstaviti kao četiri binarne vrednosti (00, 01, 10, and 11).
- ❑ Tri registra mogu imati osam binarnih vrednosti (000, 001, 010, 011, 100, 101, 110, and 111), itd.
- ❑ Naša mašina se sastoji od **5** jedinsvenih stanja tako da zahtevaju minimalno **tri** promenljive registra stanja.
- ❑ Izaberite optimalno kodovanje i zamenite kodove stanja u tablici istinitosti.
- ❑ Dopunite nekorišćene kodove sa (?) ili (*), kako bi se mogla izvršiti optimizacija Karnuovim mapama.
- ❑ Izvršite optimizaciju funkcija d2, d1, d0, dispense i change.

Označavanje stanja (3)



Current State			State Assignments
q2	q1	q0	
0	0	0	10-cents
0	0	1	15-cents
0	1	0	—
0	1	1	20-cents
1	0	0	0-cents
1	0	1	5-cents
1	1	0	—
1	1	1	—
Next State			State Assignments
d2	d1	d0	

Tablica istinitosti (1)

Staro stanje

Novo stanje

q2	q1	q0	nick	dime	disp	chan	d2	d1	d0
1	0	0	0	0	0	0	1	0	0
1	0	0	1	0	0	0	1	0	1
1	0	0	0	1	0	0	0	0	0
1	0	1	0	0	0	0	1	0	1
1	0	1	1	0	0	0	0	0	0
1	0	1	0	1	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	1
0	0	0	0	1	0	0	0	1	1
0	0	1	?	?	1	0	1	0	0
0	1	1	?	?	1	1	1	0	0

Tablica istinitosti (2)

- ❑ Pogledajte funkciju dispense, ima samo dve jedinice i to u susednim ćelijama, te se može jednostavno optimizovati: $\sim q_2$ & q_0 .
- ❑ Druga jednostavna optimizacija je funkcija change q_1 .
- ❑ Za ostale funkcije se mora kreirati Karnuova mapa sa 5 promenljivih.
- ❑ Ova mapa se svodi na dvostruku Karnuovu mapu sa definisanom jednom od promenljivih.

Primer: Karnaugh mape, 5-prom.

$A = 0$					$A = 1$				
BC	DE				BC	DE			
	00	01	11	10		00	01	11	10
00	0	1	3	2	00	16	17	19	18
01	4	5	7	6	01	20	21	23	22
11	12	13	15	14	11	28	29	31	30
10	8	9	11	10	10	24	25	27	26

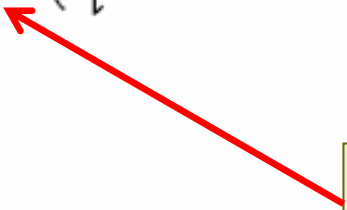
$$\bar{B} \cdot \bar{E} \cdot \bar{A} \mid B \cdot \bar{D} \cdot E \cdot \bar{A}$$

$A = 0$					$A = 1$				
BC	DE				BC	DE			
	00	01	11	10		00	01	11	10
00	1			1	00				
01	1			1	01		1	1	
11			1		11		1	1	
10			1		10		1		

$$C \cdot E \cdot A \mid B \cdot \bar{D} \cdot E \cdot A$$

$$f = A' B' E' + B D' E + A C E$$

Označavanje stanja (4)

$$\begin{aligned}d0 &= (\overline{q0} \ \& \ \overline{q2} \ \& \ dime) \vee (q0 \ \& \ q2 \ \& \ \overline{nickel}) \vee (\overline{q0} \ \& \ nickel) \\d1 &= (\overline{q0} \ \& \ \overline{q2} \ \& \ dime) \\d2 &= (q0 \ \& \ \overline{q2}) \vee (q2 \ \& \ \overline{nickel} \ \& \ \overline{dime}) \vee (\overline{q0} \ \& \ q2 \ \& \ \overline{dime}) \\dispense &= (q0 \ \& \ \overline{q2}) \\change &= (q1)\end{aligned}$$


Koristi se više puta a računa samo jednom

- ❑ Domaći: Koliko promenljivih će se pojaviti u Karnuovim mapama?
- ❑ Nacrtati odgovarajuće Karnoove mape za slučaj naše mašine. Proveriti dobijene rezultate!

Latching

- ❑ Razmotriti problem prvog uključivanja.
- ❑ U kom stanju će se sistem “probuditi”?
- ❑ Koji se problemi mogu javiti kod ovakve realizacije?
- ❑ Šta se dešava ako se kontroler nađe u nekom stanju koje se ne koristi (latching).
- ❑ Kako se može izaći iz njega?
- ❑ Rešenje 1: Unapred definisati u koje se stanje prelazi ako se desi latching.
- ❑ Šta ako se desi nestanak napajanja?
- ❑ Rešenje 2: Formira se poseban signal koji dovodi (puni) registre u zapamćenost stanje.