Proiect Individual
Aparat Alba-Neagra

Student: Ulmeanu Vlad-Adrian, grupa 315CA
Asistent: Popescu Teodor

Cuprins

1 Tema Proiectului **2**
1.1 Temă 2
1.2 Detalii de implementare 2

2 Mod de implementare **3**
2.1 Realizare schemă bloc 3
2.2 Explicarea semnalelor din schema bloc 4
2.3 Explicarea ieșirilor din schema bloc 4
2.4 Funcționarea aparatului 5

3 Organigrama aparatului **6**

4 Spațiul stărilor **7**

5 Tabelul tranzițiilor **8**

6 Diagramele Karnaugh pentru stările următoare și ieșiri **9**

7 Implementare Master **14**

8 Anexa 1: Randomizator **16**

9 Anexa 2: Algoritm calculare tabel spațiu stări **22**

10 Anexa 3: Calcul câștiguri **24**

11 Referințe **24**

1 Tema Proiectului

1.1 Temă

Tema aleasă prezintă realizarea unui aparat de Alba-Neagra, capabil de a simula activitățile unui joc real echivalent. Acesta presupune existența a trei păhărele, exact unul dintre ele conținând un disc mic. În urma amestecării păhărelelor într-un mod aleator de altă persoană, jucătorul trebuie sa ghicească paharul sub care se află discul.

Pentru implementare, în timpul unui joc, jucătorul nu va ști în niciun moment de timp sub care dintre cele trei păhărele (0, 1 sau 2) se află discul câștigător, dar va trebui să facă o alegere (0, 1 sau 2), care va fi comparată obiectiv cu răspunsul corect, ales aleator anterior.

Pentru a complica jocul, există și un mod special de a-l juca: jucătorul face o alegere, care dacă se dovedește a fi corectă, mai este supusă altei încercări. Se recalculează locul discului câștigător, însă alegerea jucătorului nu se modifică. Acum, pentru a fi plătit, jucătorul trebuie să fi ales **greșit** paharul câștigător.

1.2 Detalii de implementare

Acest proiect urmează detaliile de implementare impuse pentru proiectul individual al cursului de Proiectare Logică:

- implementare pe adrese pe 4 biți (Q_3, Q_2, Q_1, Q_0, unde Q_3 este cel mai semnificativ bit)
- Q_0 este implementat folosind un CBB de tip D și un MUX 16:1 (aici, Q_3, Q_2, Q_1, Q_0 sunt variabile de selecție)
- Q_1 este implementat folosind un CBB de tip D și un MUX 8:1 (Q_3, Q_1, Q_0 variabile de selecție)
- Q_2 folosind un CBB de tip JK, având J implementat printr-un MUX 2:1 (Q_1 variabilă de selecție) și K printr-un MUX 4:1 (Q_3 și Q_1 variabile de selecție).
- Q_3 folosind un CBB de tip JK, având J implementat cu porți de tip NAND și K cu cu porți de tip NOR.
- ieșirile circuitului sunt implementate folosind un decodificator 4:16, având ieșirile active pe 0.

2 Mod de implementare

2.1 Realizare schemă bloc

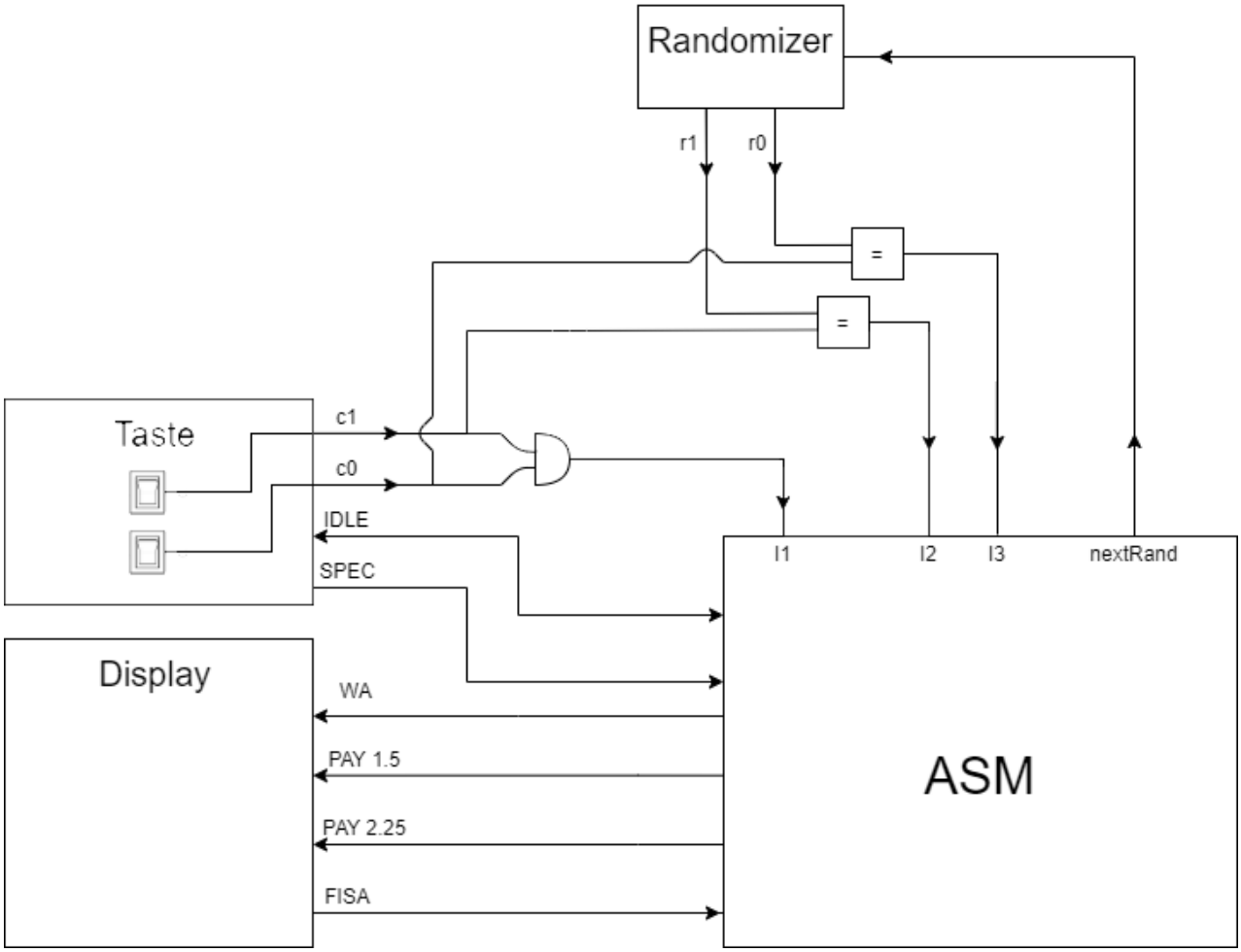


Fig. 1: Schemă bloc

Unde operatorul "=" prezent în figura 1 se poate descrie astfel:

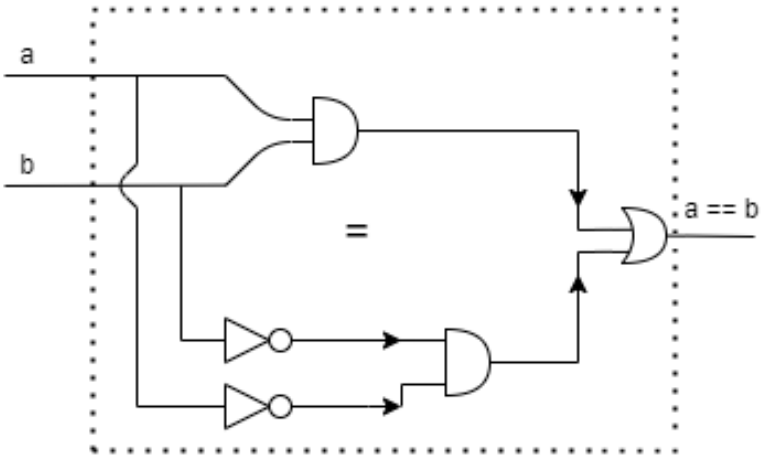


Fig. 2: Submodul egalitate

2.2 Explicarea semnalelor din schema bloc

- **FISA** - pentru a începe jocul, jucătorul trebuie să introducă de fiecare dată o unitate monetară. **FISA** este 1 doar imediat după ce a fost introdusă unitatea monetară, permițând algoritmului să continue.
- **c1**, **c0** - cele două alegeri (choice) ale jucătorului, reprezentate prin două switch-uri. Se pot observa 4 variante posibile de alegere. Cum avem nevoie doar de 3, înseamnă ca una (**c1** == 1 și **c0** == 1) este interzisă. În acest caz, jucătorul trebuie să introducă un nou input.
- **IDLE** - este considerat 1 dacă a fost introdus un input de către jucător și trebuie verificat mai departe. Dacă este 0, înseamnă că încă se așteaptă introducerea lui **c1** sau **c0**.
- **SPEC** - este considerat 1 dacă jucătorul a decis sa joace în modul "special" descris înainte. Dacă este 0, atunci se va juca varianta clasică, cu o singură ghicire.
- **r1**, **r0** - cele două alegeri aleatoare și corecte ale aparatului. Odată introdusă alegerea jucătorului, **r1** și **r0** vor fi comparate cu **c1** și **c0** (prin intermediul intrărilor **I2**, **I3**) pentru a putea decide un rezultat.
- **I1** - este 1 dacă input-ul jucătorului este interzis și 0 altfel.
- **I2** - este 1 dacă prima "cifră" (**c1**) a jucătorului este identică cu prima "cifră" a aparatului (**r1**).
- **I3** - este 1 dacă a doua "cifră" (**c0**) a jucătorului este identică cu a doua "cifră" a aparatului (**r0**).

2.3 Explicarea ieșirilor din schema bloc

- **nextRand** - când consideră că este necesar, algoritmul va trimite un semnal către modulul extern "Randomizer" prin această ieșire, dorind să primească înapoi un număr "proaspăt" aleator, care poate fi 0, 1 sau 2, codificat prin intrările r1 și r0. Implementarea modulului "Randomizer" poate fi găsită în anexa 1.
- **WA** - (Wrong Answer) această ieșire este 1 atunci când jucătorul pierde ori jocul clasic, ori cel special. Se afișează un mesaj corespunzător pe modulul extern "Display".
- **PAY 1.5** - această ieșire este 1 doar atunci când jucătorul a câștigat jocul clasic. Acesta este răsplătit cu o sumă egală cu 150% din valoarea costului de intrare.
- **PAY 2.25** - această ieșire este 1 doar atunci când jucătorul a câștigat jocul special. Acesta este răsplătit cu o sumă egală cu 225% din valoarea costului de intrare. Explicațiile pentru valorile câștigurilor apar în anexa 3.

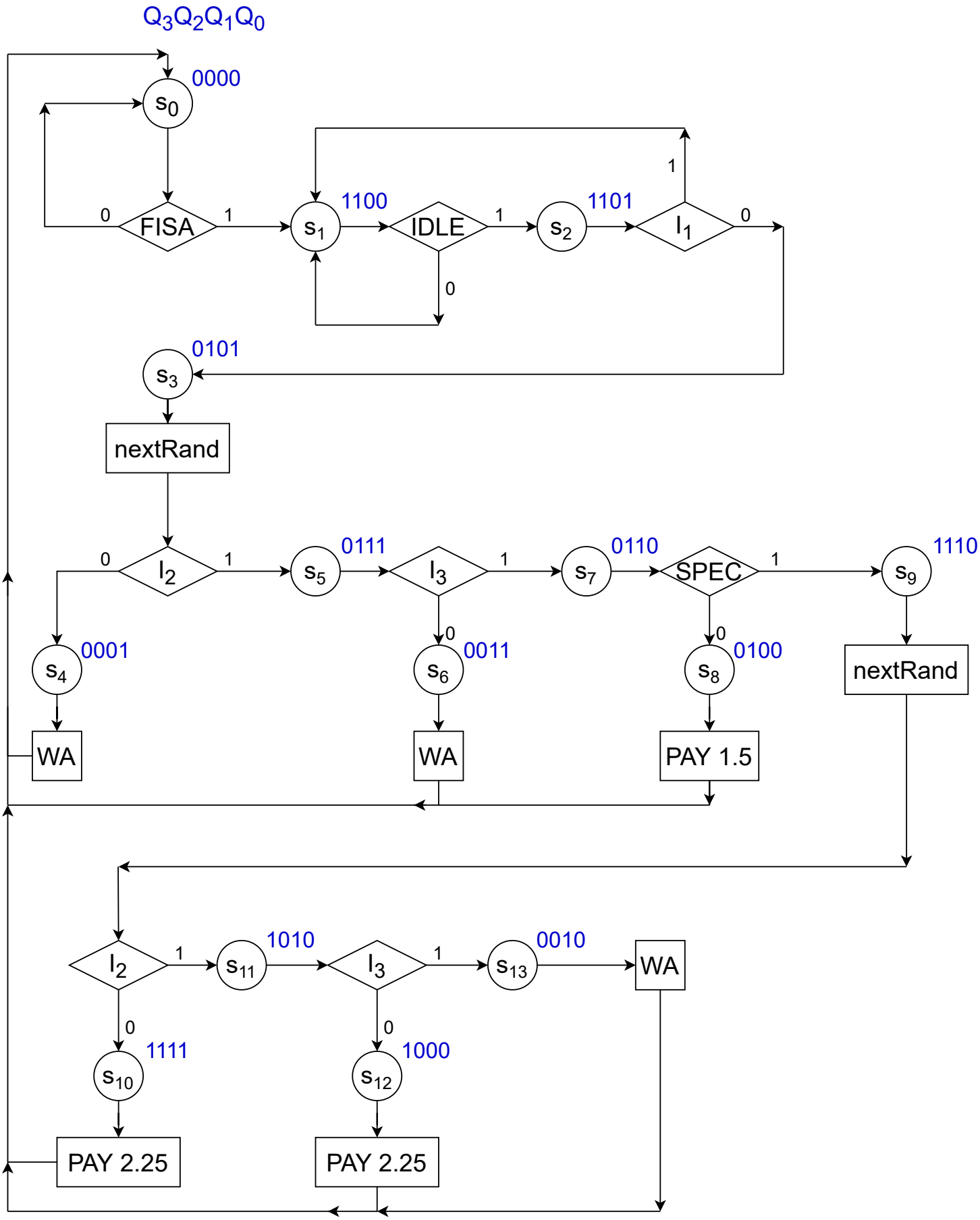
2.4 Funcționarea aparatului

Aparatul se va afla într-un mod pasiv pana la introducerea unei unități monetare într-o parte din modulul “Display”. După aceea, se va aștepta introducerea unei alegeri din partea jucătorului. Se va verifica corectitudinea alegerii, iar în caz contrar se va solicita o nouă alegere.

Se va genera un nou număr aleator, iar acesta se va compara cu alegerea jucătorului. Dacă este diferit, se va genera output-ul “**WA**” și se va reveni în starea pasivă. Altfel, dacă modul special nu a fost selectat, se va genera output-ul “**PAY 1.5**” și se va reveni în starea pasivă.

Altfel, intrând în modul special, mai generăm încă o dată un număr aleator și îl comparăm cu alegerea neschimbată a jucătorului. Dacă este diferită, generăm output-ul “**PAY 2.25**”, iar dacă este identică generăm output-ul “**WA**”; în orice caz, vom reveni în starea pasivă.

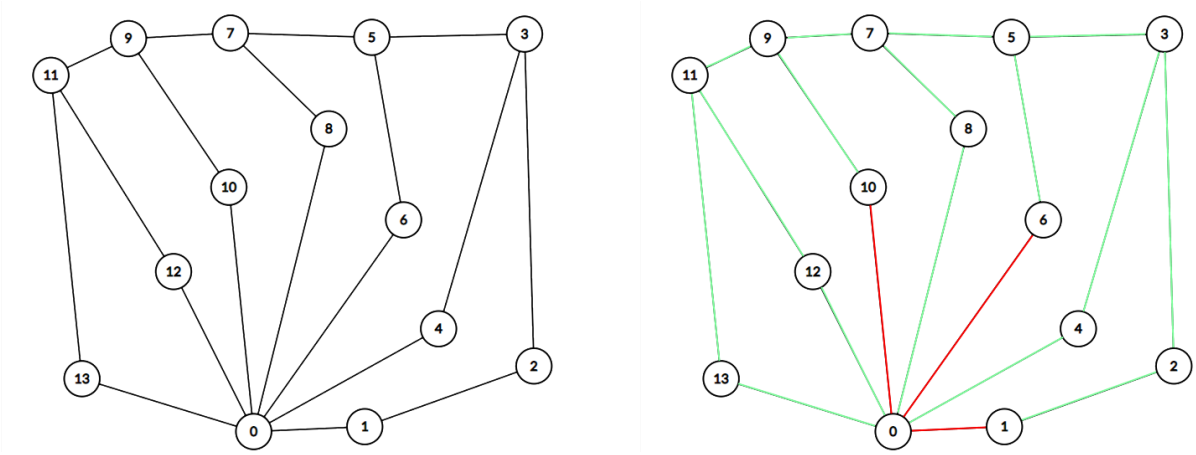
3 Organigrama aparatului



4 Spațiul stărilor

Pentru a completa tabelul cu spațiul stărilor, am creat mai întâi un graf cu muchii bidirecționale, cu numărul de noduri egal cu numărul de stări din organigramă. Între două noduri există o muchie in graf dacă stările lor echivalente sunt conectate în organigramă.

Așadar, rezultă următorul graf (stânga):



(a) stânga (b) dreapta

Observăm că singurul nod cu grad mai mare decat 4 din graf este nodul 0, care are gradul 7. Așadar, în cel mai bun caz, vor exista exact trei muchii ale căror stări nu vor putea fi adiacente în tabelul de spațiu al stărilor.

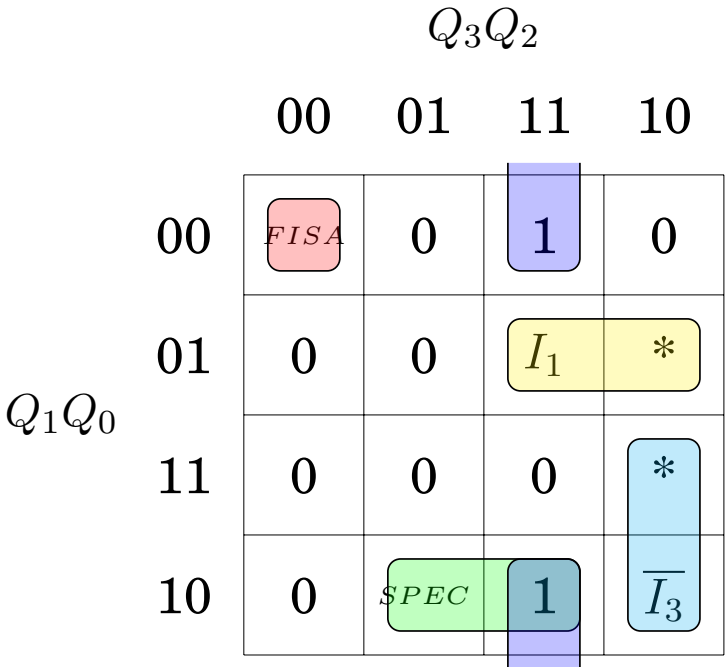
Folosind codul scris de mine prezentat în anexa 2, am reușit să generez un tabel de spațiu al stărilor optim pentru organigrama actuală, în care doar stările s_0-s_{10}, s_0-s_6, s_0-s_1 (graf dreapta) nu sunt adiacente în tabel:

Q_1Q_0 \ Q_3Q_2	00	01	11	10
00	s_0	s_8	s_1	s_{12}
01	s_4	s_3	s_2	*
11	s_6	s_5	s_{10}	*
10	s_{13}	s_7	s_9	s_{11}

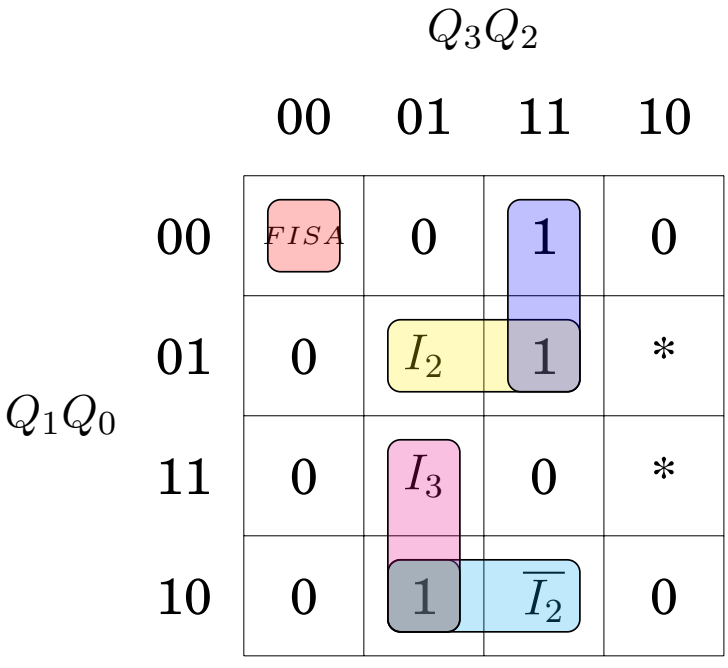
5 Tabelul tranzițiilor

	Q_3^t	Q_2^t	Q_1^t	Q_0^t	Q_3^{t+1}	Q_2^{t+1}	Q_1^{t+1}	Q_0^{t+1}	J_3	K_3	J_2	K_2	D_1	D_0	next Rand	WA	PAY 1.5	PAY 2.25
s_0	0	0	0	0	FISA	FISA	0	0	FISA	*	FISA	*	0	0	0	0	0	0
s_4	0	0	0	1	0	0	0	0	0	*	0	*	0	0	0	1	0	0
s_{13}	0	0	1	0	0	0	0	0	0	*	0	*	0	0	0	1	0	0
s_6	0	0	1	1	0	0	0	0	0	*	0	*	0	0	0	1	0	0
s_8	0	1	0	0	0	0	0	0	0	*	*	1	0	0	0	0	1	0
s_3	0	1	0	1	0	I_2	I_2	1	0	*	*	$!I_2$	I_2	1	1	0	0	0
s_7	0	1	1	0	SPEC	1	SPEC	0	SPEC	*	*	0	SPEC	0	0	0	0	0
s_5	0	1	1	1	0	I_3	1	$!I_3$	0	*	*	$!I_3$	1	$!I_3$	0	0	0	0
s_{12}	1	0	0	0	0	0	0	0	*	1	0	*	0	0	0	0	0	1
*	1	0	0	1	*	*	*	*	*	*	*	*	*	*	*	*	*	*
s_{11}	1	0	1	0	$!I_3$	0	I_3	0	*	I_3	0	*	I_3	0	0	0	0	0
*	1	0	1	1	*	*	*	*	*	*	*	*	*	*	*	*	*	*
s_1	1	1	0	0	1	1	0	IDLE	*	0	*	0	0	IDLE	0	0	0	0
s_2	1	1	0	1	I_1	1	0	$!I_1$	*	$!I_1$	*	0	0	$!I_1$	0	0	0	0
s_9	1	1	1	0	1	$!I_2$	1	$!I_2$	*	0	*	I_2	1	$!I_2$	1	0	0	0
s_{10}	1	1	1	1	0	0	0	0	*	1	*	1	0	0	0	0	0	1

6 Diagramele Karnaugh pentru stările următoare și ieșiri



$$Q_3^{t+1} = Q_3Q_2\overline{Q}_0 + Q_3\overline{Q}_1Q_0I_1 + Q_3\overline{Q}_2Q_1\overline{I}_3 + \overline{Q}_3\overline{Q}_2\overline{Q}_1\overline{Q}_0 \cdot FISA + Q_2Q_1\overline{Q}_0 \cdot SPEC$$



$$Q_2^{t+1} = \overline{Q}_3Q_2Q_1\overline{Q}_0 + Q_3Q_2\overline{Q}_1 + \overline{Q}_3Q_2Q_1I_3 + Q_2Q_1\overline{Q}_0\overline{I}_2 + Q_2\overline{Q}_1Q_0I_2 + \overline{Q}_3\overline{Q}_2\overline{Q}_1\overline{Q}_0 \cdot FISA$$

Q_1Q_0 \ Q_3Q_2	00	01	11	10
00	0	0	0	0
01	0	I_2	0	*
11	0	1	0	*
10	0	$SPEC$	1	I_3

$$Q_1^{t+1} = \overline{Q}_3 Q_2 Q_1 Q_0 + Q_3 Q_2 Q_1 \overline{Q}_0 + \overline{Q}_3 Q_2 Q_1 \cdot SPEC + Q_3 Q_2 Q_0 I_2 + Q_3 Q_1 \overline{Q}_0 I_3$$

Q_1Q_0 \ Q_3Q_2	00	01	11	10
00	0	0	$IDLE$	0
01	0	1	$\overline{I_1}$	*
11	0	$\overline{I_3}$	0	*
10	0	0	$\overline{I_2}$	0

$$Q_0^{t+1} = \overline{Q}_3 Q_2 \overline{Q}_1 Q_0 + Q_2 \overline{Q}_1 Q_0 \overline{I}_1 + \overline{Q}_3 Q_2 Q_0 \overline{I}_3 + Q_3 Q_2 Q_1 \overline{Q}_0 \overline{I}_2 + Q_3 Q_2 \overline{Q}_1 \overline{Q}_0 \cdot IDLE$$

Q_1Q_0 \ Q_3Q_2	00	01	11	10
00	$FISA$	0	*	*
01	0	0	*	*
11	0	0	*	*
10	0	$SPEC$	*	*

$$J_3 = Q_2Q_1\overline{Q_0} \cdot SPEC + \overline{Q_2}\,\overline{Q_1}\,\overline{Q_0} \cdot FISA = \overline{\overline{Q_2Q_1\overline{Q_0}}\,SPEC} \cdot \overline{\overline{Q_2}\,\overline{Q_1}\,\overline{Q_0}\,FISA}$$

Q_1Q_0 \ Q_3Q_2	00	01	11	10
00	*	*	0	1
01	*	*	$\overline{I_1}$	*
11	*	*	1	*
10	*	*	0	I_3

$$K_3 = Q_1Q_0 + \overline{Q_2}\,\overline{Q_1} + Q_0\,\overline{I_1} + \overline{Q_2}I_3 = \overline{\overline{\overline{Q_1} + \overline{Q_0}} + \overline{\overline{Q_2} + \overline{Q_1}} + \overline{\overline{Q_0} + I_1} + \overline{Q_2 + \overline{I_3}}}$$

Q_1Q_0 \ Q_3Q_2	00	01	11	10
00	$FISA$	*	*	0
01	0	*	*	*
11	0	*	*	*
10	0	*	*	0

$Q_1 = 0 \rightarrow J_2 = \overline{Q}3\overline{Q}0 \cdot FISA$ (dreptunghi 2x4 sus)
$Q_1 = 1 \rightarrow J_2 = (*)0$ (dreptunghi 2x4 jos)

Q_1Q_0 \ Q_3Q_2	00	01	11	10
00	*	1	0	*
01	*	$\overline{I_2}$	0	*
11	*	$\overline{I_3}$	1	*
10	*	0	I_2	*

$Q_3 = 0, Q_1 = 0 \rightarrow K_2 = \overline{Q}_0 + \overline{I}_2$ (pătrat 2x2 stânga-sus)
$Q_3 = 0, Q_1 = 1 \rightarrow K_2 = Q_0\overline{I}_3$ (pătrat 2x2 stânga-jos)
$Q_3 = 1, Q_1 = 0 \rightarrow K_2 = (*)0$ (pătrat 2x2 dreapta-sus)
$Q_3 = 1, Q_1 = 1 \rightarrow K_2 = Q_0 + I_2$ (pătrat 2x2 dreapta-jos)

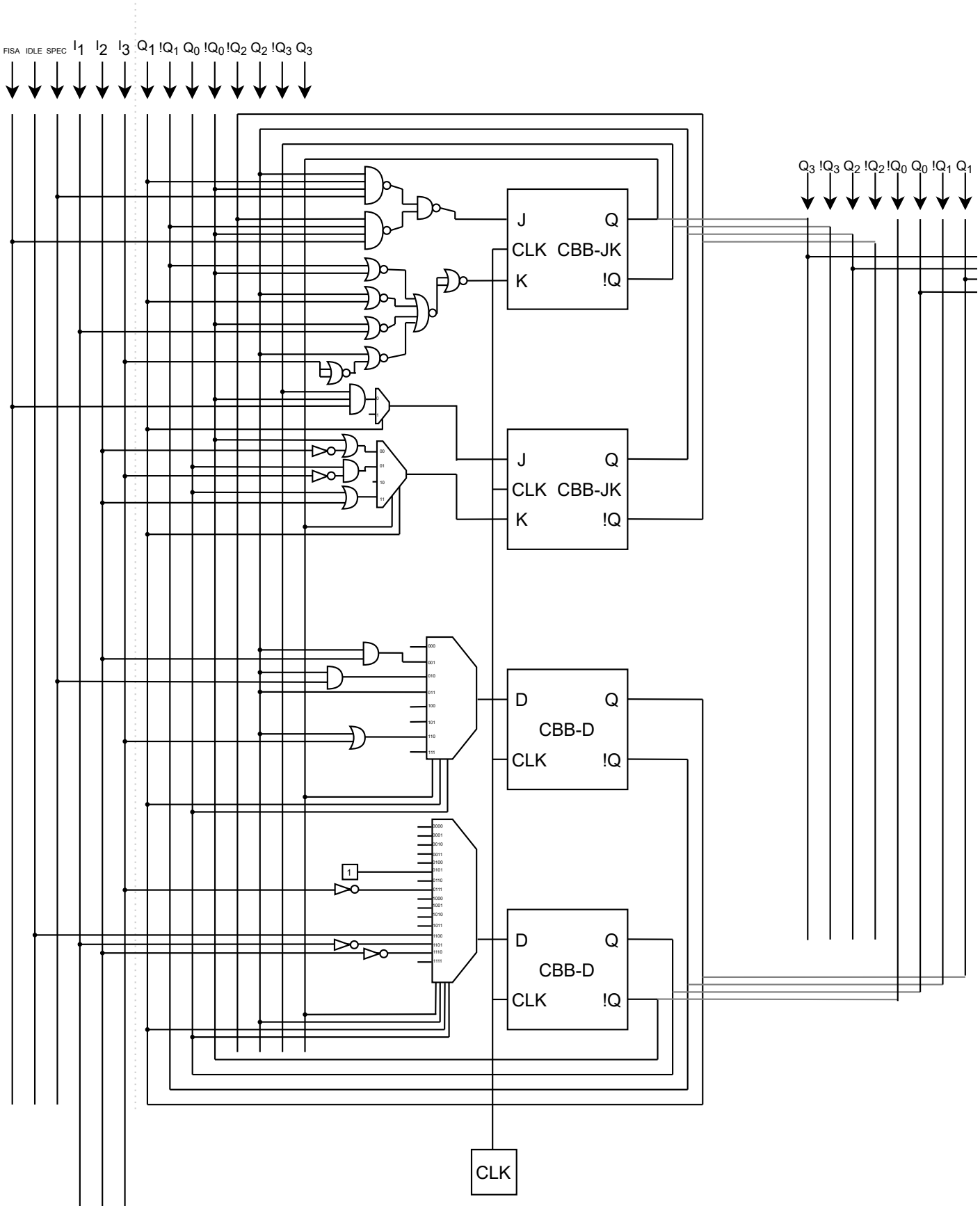
Q_1Q_0 \ Q_3Q_2	00	01	11	10
00	0	0	0	0
01	0	I_2	0	*
11	0	1	0	*
10	0	$SPEC$	1	I_3

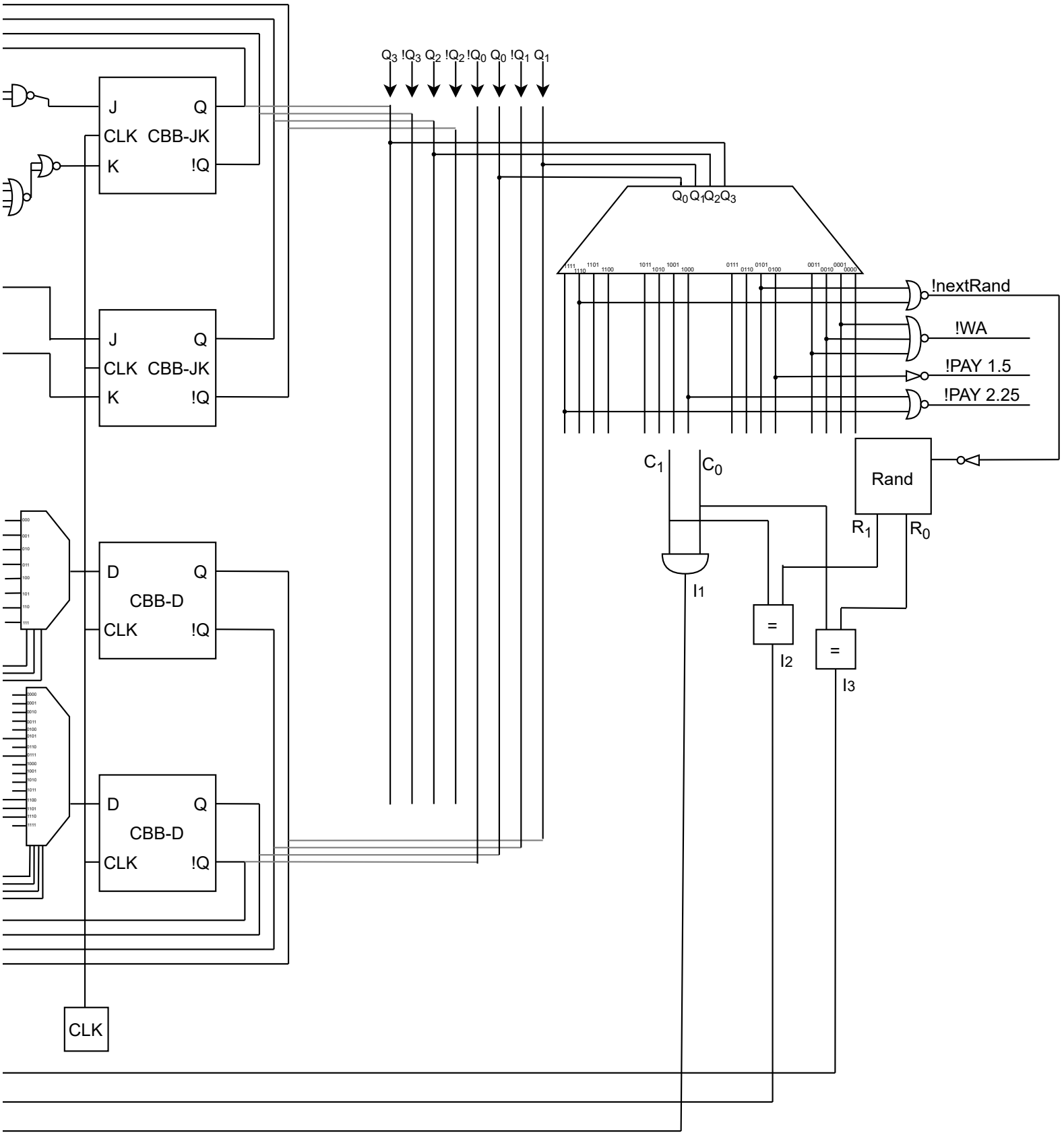
$Q_3 = 0, Q_1 = 0, Q_0 = 0 \rightarrow D_1 = 0$ (dreptunghi 1x2, primul din stânga)
$Q_3 = 0, Q_1 = 0, Q_0 = 1 \rightarrow D_1 = Q_2 I_2$ (dreptunghi 1x2, al doilea din stânga)
$Q_3 = 0, Q_1 = 1, Q_0 = 0 \rightarrow D_1 = Q_2 \cdot SPEC$ (dreptunghi 1x2, al patrulea din stânga)
$Q_3 = 0, Q_1 = 1, Q_0 = 1 \rightarrow D_1 = Q_2$ (dreptunghi 1x2, al treilea din stânga)
$Q_3 = 1, Q_1 = 0, Q_0 = 0 \rightarrow D_1 = 0$ (dreptunghi 1x2, primul din dreapta)
$Q_3 = 1, Q_1 = 0, Q_0 = 1 \rightarrow D_1 = (*)0$ (dreptunghi 1x2, al doilea din dreapta)
$Q_3 = 1, Q_1 = 1, Q_0 = 0 \rightarrow D_1 = Q_2 + I_3$ (dreptunghi 1x2, al patrulea din dreapta)
$Q_3 = 1, Q_1 = 1, Q_0 = 1 \rightarrow D_1 = (*)0$ (dreptunghi 1x2, al treilea din dreapta)

Q_1Q_0 \ Q_3Q_2	00	01	11	10
00	0	0	$IDLE$	0
01	0	1	$\overline{I_1}$	*
11	0	$\overline{I_3}$	0	*
10	0	0	$\overline{I_2}$	0

MUX $16:1$ **cu variabilele de selecție** Q_3, Q_2, Q_1, Q_0 **pentru alegerea lui** D_0.

7 Implementare Master





.svg-ul pentru implementarea circuitului se poate obține aici: cutt.ly/implementare-master-svg.

8 Anexa 1: Randomizator

Pentru acest proiect, am avut nevoie de un generator de numere aleatoare din mulțimea $0, 1, 2$. Deoarece am folosit adrese pe 4 biți până acum, am fost inspirat să construiesc un generator de perioadă 16. Unul dintre cele mai simple generatoare de implementat este **generatorul liniar congruențial**. Acesta se folosește de următoarele valori:

- *seed*, sau valoarea veche a generatorului;
- a, un multiplicator, număr natural, care este de obicei mare;
- b, un număr natural, de obicei cu cateva ordine de mărime mai mic decât a;
- *mod*, un număr mai mare decat a sau b.

Valoarea nouă a lui seed se obține după formula: $(a \cdot seed + b) \,\%\, mod$.

De obicei, vrem $gcd(seed, mod) = 1$. Putem sa luăm *mod* prim pentru a garanta acest lucru. În practică, b este foarte mic, iar *mod* este luat ca o putere mare a lui 2, astfel transformând operația costisitoare de modulo într-o shiftare pe biți.

În cazul de față, deoarece perioada este foarte mică, nu am luat *mod* prim. Codul în Python care generează primele 16 numere modulo 3 din secvența cu $a = 32984, b = 127163, mod = 5757557$ este:

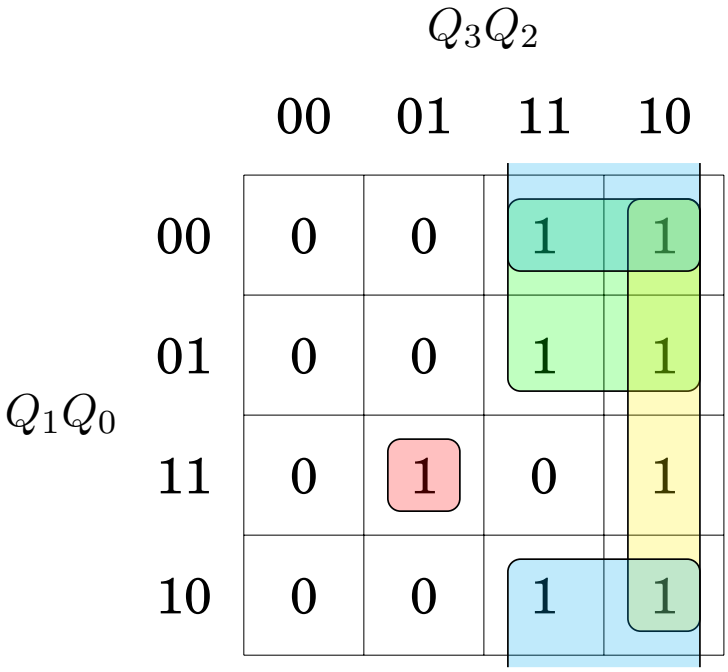
```
import math

def lcg():
    a = 32984
    b = 127163
    mod = 5757557
    seed = 0
    while True:
        seed = (seed * a + b) % mod
        yield seed

i = 1
for seed in lcg():
    print(seed % 3)
    i += 1
    if (i >= 16):
        break

#2212121001022201
```

Primele 16 numere ale secvenței sunt: $2, 2, 1, 2, 1, 2, 1, 0, 0, 1, 0, 2, 2, 2, 0, 1$.

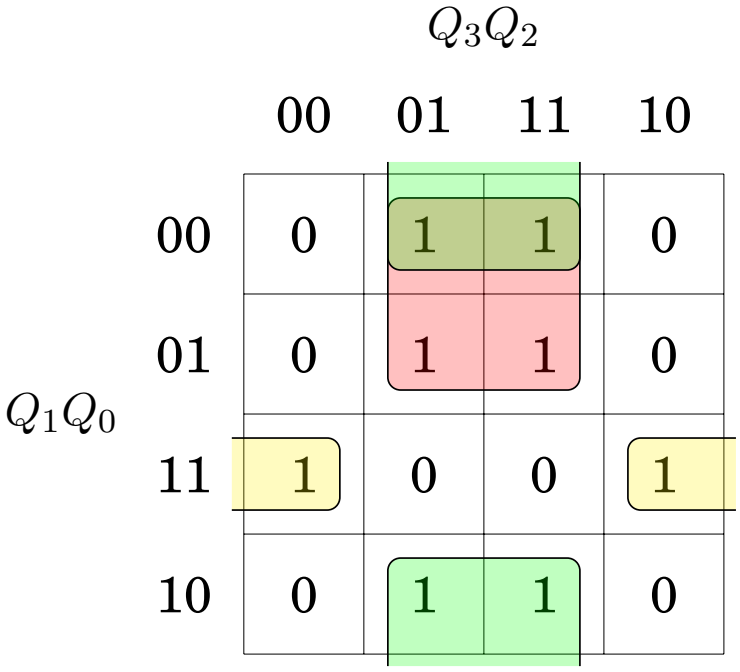
Am ales să implementez modulul “Random” cu 4 CBB-uri de tip D. Mai jos se găsește tabelul tranzițiilor. Ultima coloană reprezintă ieșirea dorită în starea de pe prima coloană. Coloanele R_1, R_0 reprezintă ieșirile propriu-zise, atribuite numărului aleator dorit.

	Q_3^t	Q_2^t	Q_1^t	Q_0^t	Q_3^{t+1}	Q_2^{t+1}	Q_1^{t+1}	Q_0^{t+1}	D_3	D_2	D_1	D_0	R_1	R_0	R
s_0	0	0	0	0	0	0	0	1	0	0	0	1	1	0	2
s_1	0	0	0	1	0	0	1	0	0	0	1	0	1	0	2
s_2	0	0	1	0	0	0	1	1	0	0	1	1	0	1	1
s_3	0	0	1	1	0	1	0	0	0	1	0	0	1	0	2
s_4	0	1	0	0	0	1	0	1	0	1	0	1	0	1	1
s_5	0	1	0	1	0	1	1	0	0	1	1	0	1	0	2
s_6	0	1	1	0	0	1	1	1	0	1	1	1	0	1	1
s_7	0	1	1	1	1	0	0	0	1	0	0	0	0	0	0
s_8	1	0	0	0	1	0	0	1	1	0	0	1	0	0	0
s_9	1	0	0	1	1	0	1	0	1	0	1	0	0	1	1
s_{10}	1	0	1	0	1	0	1	1	1	0	1	1	0	0	0
s_{11}	1	0	1	1	1	1	0	0	1	1	0	0	1	0	2
s_{12}	1	1	0	0	1	1	0	1	1	1	0	1	1	0	2
s_{13}	1	1	0	1	1	1	1	0	1	1	1	0	1	0	2
s_{14}	1	1	1	0	1	1	1	1	1	1	1	1	0	0	0
s_{15}	1	1	1	1	0	0	0	0	0	0	0	0	0	1	1

Diagramele Karnaugh pentru stările următoare și ieșiri:



$$Q_3^{t+1} = D_3^t = \overline{Q}_3 Q_2 Q_1 Q_0 + Q_3 \overline{Q}_2 + Q_3 \overline{Q}_1 + Q_3 \overline{Q}_0$$



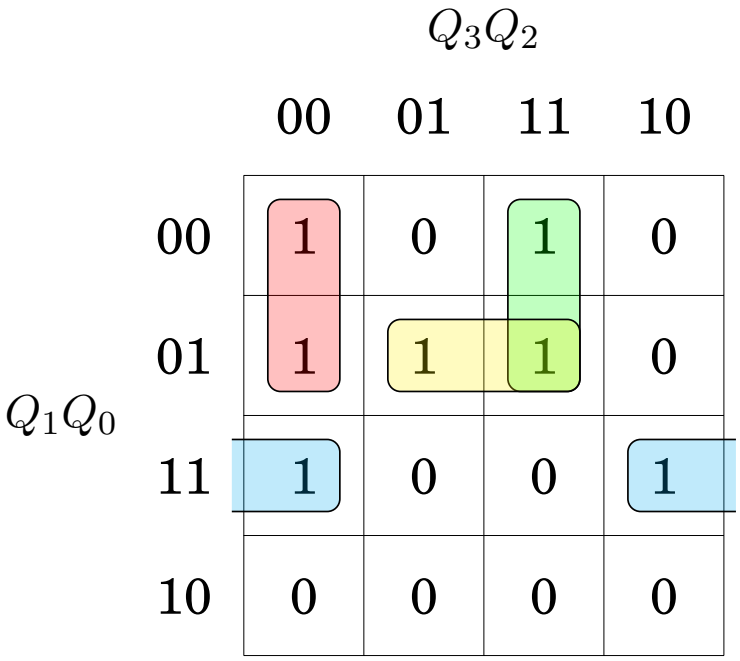
$$Q_2^{t+1} = D_2^t = \overline{Q}_2 Q_1 Q_0 + Q_2 \overline{Q}_1 + Q_2 \overline{Q}_0$$

Q_1Q_0 \ Q_3Q_2	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	0	0	0	0
10	1	1	1	1

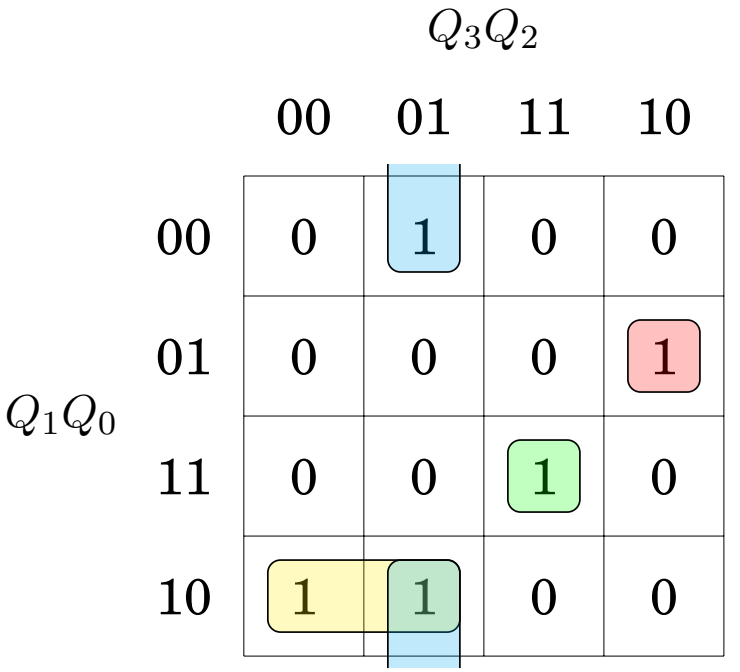
$$Q_1^{t+1} = D_1^t = \overline{Q}_1 Q_0 + Q_1 \overline{Q}_0 (Q_1 XOR Q_0)$$

Q_1Q_0 \ Q_3Q_2	00	01	11	10
00	1	1	1	1
01	0	0	0	0
11	0	0	0	0
10	1	1	1	1

$$Q_0^{t+1} = D_0^t = \overline{Q}_0$$

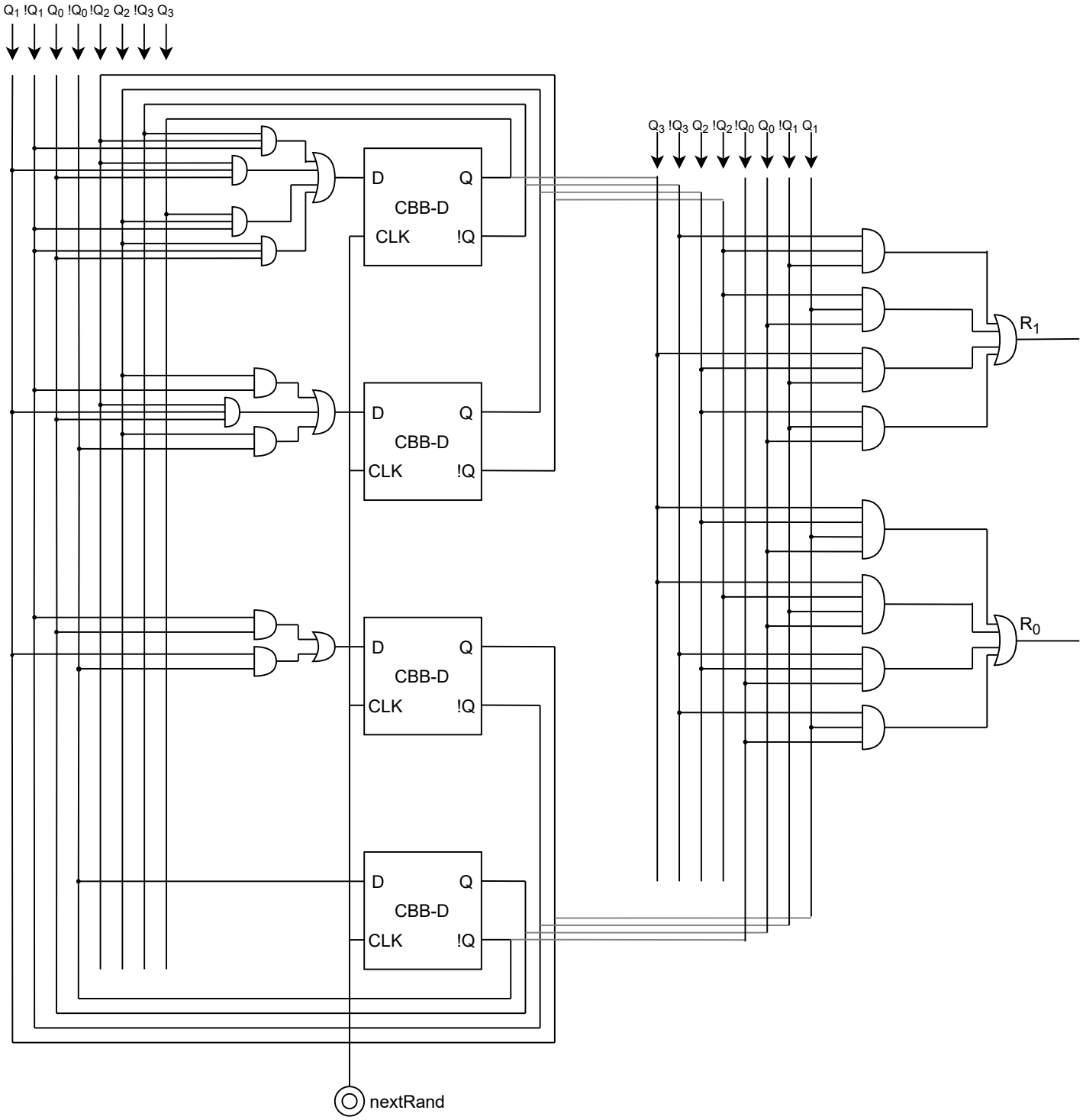


$$R_1 = \overline{Q}_3\overline{Q}_2\overline{Q}_1 + \overline{Q}_2Q_1Q_0 + Q_3Q_2\overline{Q}_1 + Q_2\overline{Q}_1Q_0$$



$$R_0 = Q_3Q_2Q_1Q_0 + Q_3\overline{Q}_2\overline{Q}_1Q_0 + \overline{Q}_3Q_2\overline{Q}_0 + \overline{Q}_3Q_1\overline{Q}_0$$

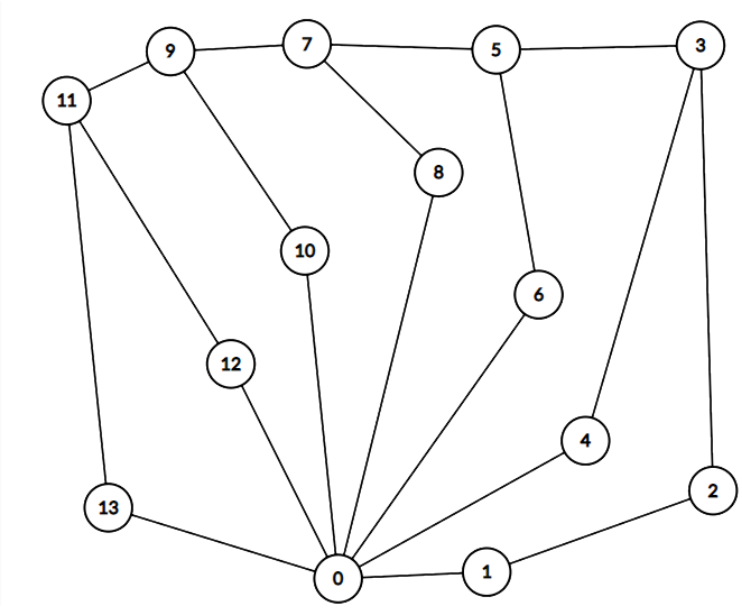
Implementarea circuitului:



Implementarea in logic.ly se poate găsi aici: cutt.ly/nextrand-logicly. .svg-ul se poate găsi aici: cutt.ly/nextrand-svg.

Trebuie să amintim că din cauza lipsei unei stări intermediare între ieșirea “nextRand” din organigramă și blocul de decizie care se folosește de aceste variabile, nu putem spune cu exactitate când vor fi generate cele două cifre noi aleatoare(înainte de decizie sau după aceasta). Așadar, avem parte de un caracter mai impredictibil.

9 Anexa 2: Algoritm calculare tabel spațiu stări



Privind pentru prima oară graful de deasupra, am încercat sa completez manual tabelul de stări, însă am rămas cu 6 muchii din 19 pe dinafară. Cum numărul de muchii totale este relativ mare, ar fi greu pentru cineva sa scadă sub 6 muchii nepuse în tabel, darămite să găsească o soluție optimă, cu doar 3 muchii rămase.

Așadar, am încercat sa scriu inițial un cod care "amestecă" aleator stările în tabel și numără câte muchii din graf se regăsesc în el. Această implementare este destul de costisitoare, dar reușește să găsească un tabel cu doar 5 muchii rămase după un milion de simulări (deci deja mai bine decat de mână).

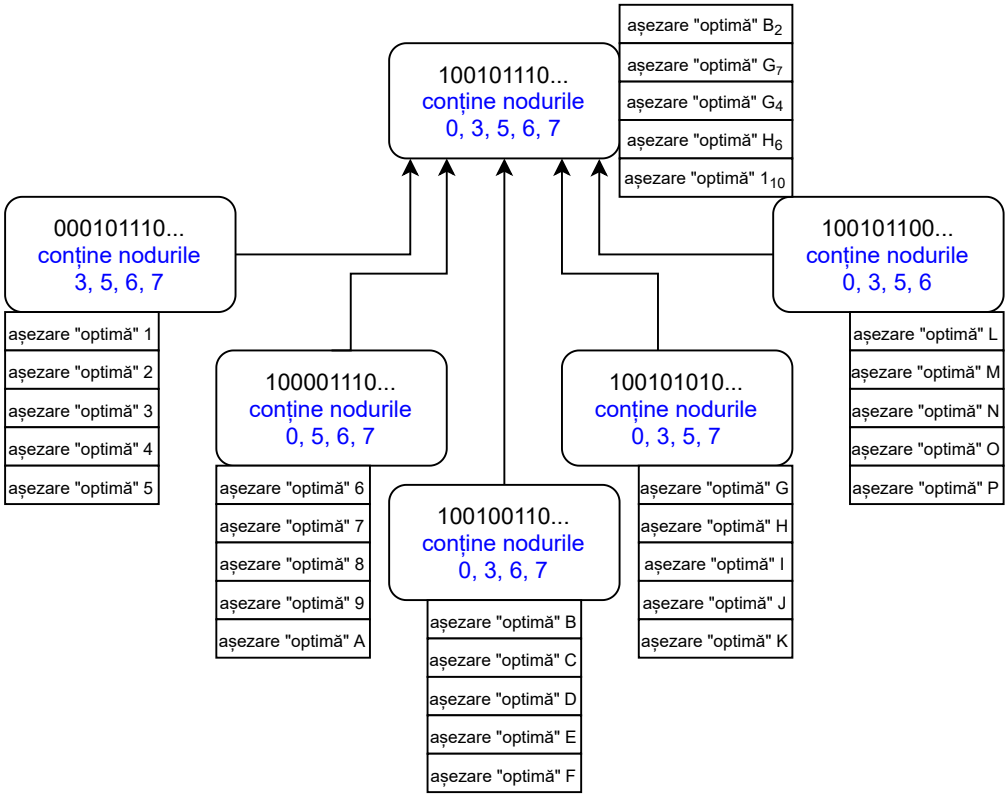
Q_1Q_0 \ Q_3Q_2	00	01	11	10
00	*	s_7	s_5	s_6
01	s_{12}	s_8	s_3	*
11	s_{11}	s_{13}	s_2	s_1
10	s_4	s_9	s_{10}	s_0

(fără starea s_0 translatată în $(0,0,0,0)$)

Implementarea se poate găsi aici: https://github.com/vlad-ulmeanu01/KMap_Adjacent_States/tree/main/Random_Fill.

Următorul algoritm euristic a reușit să găsească o așezare a stărilor în tabel cu doar 3 muchii rămase:

- Cum avem 14 noduri în graf, atribuim fiecărui număr n de la 0 la $2^{14} - 1$ inclusiv câte o submulțime de stări.
- Dacă bitul i ($i \in \{0, 1, .., 13\}$) se regăsește în descompunerea numărului n în baza 2, atunci a i-a stare aparține submulțimii numărului n.
- Calculăm pentru fiecare număr n cele mai bune 5 tabele de stări care conțin doar stările din submulțimea numărului n. Considerăm un tabel mai bun decat altul dacă are mai puține muchii lipsă din graf.
- Considerăm un număr "fiu" al altui număr(ambele numere din $\{0, 1, ..., 2^{14} - 1\}$ dacă numărul "fiu" se poate obține eliminând un bit de 1 din scrierea în baza 2 al celuilalt număr ("tatăl").
- Pentru a calcula cele mai bune 5 tabele de stări pentru un număr, îl considerăm "tată" și îi luăm toți "fiii", ale căror tabele au fost deja calculate (de mai sus rezultă că valoarea unui fiu este strict mai mică decât a tatălui, iar noi "rezolvăm" numerele în ordine crescătoare). Concatenăm toate tabelele generate din toți fiii (pentru fiecare așezare a unui fiu încercăm să punem starea asociată bitului lipsă în toate pozițiile goale din tabel, astfel obținând așezări posibile pentru "tată"), le sortăm după numărul de muchii lipsă din graf și le reținem doar pe primele 5.



Mai sus avem o ilustrare pentru numărul 233 și cei 5 fii ai săi.

Există și câteva observații:

- La început pornim doar cu numărul 0 și așezarea vidă.
- Răspunsul "optim" este prima așezare a numărului $2^{14} - 1$.
- Deoarece tăiem constant din așezări pentru numere, este posibil ca răspunsul să nu fie optim. Dacă dorim un rezultat mai apropiat de cel bun, putem păstra mai multe așezări pentru fiecare număr, astfel scazand posibilitatea de a "tăia" din greșeală răspunsul corect. Pentru graful meu, am păstrat 5 așezări și am avut noroc, ajungând la cel mai bun rezultat. Trebuie amintit timpul mare de rulare aferent abordării exponențiale - în cazul de față se apropie deja de 3 minute.
- Graful de față ar putea fi denumit "graf-ciob", având în vedere că orice ciclu trebuie să conțină un anume nod, aici nodul 0.
- Complexitatea este $O(2^n \cdot (n \cdot \frac{n}{2} \cdot n \cdot (laturatabel^2 + log_2(nrmuchii)))) + \frac{n}{2} \cdot n \cdot nrstari \cdot log_2(\frac{n}{2} \cdot n \cdot nrstari))$, care pentru $n = 16, nrmuchii = 19, nrstari = 5, laturatabel = 4$ depășește 3 miliarde de operații.

Această implementare se poate găsi aici: https://github.com/vlad-ulmeanu01/KMap_Adjacent_States/tree/main/Exponential_DP.

10 Anexa 3: Calcul câștiguri

În teorie, într-un joc clasic, jucătorul are o probabilitate egală cu $\frac{1}{3}$ de a câștiga. Așadar, are nevoie în medie de $\frac{1}{\frac{1}{3}}$ jocuri pentru a câștiga. Vom stabili suma pe care o vom da jucătorului ca fiind 50% din valoarea medie în cazul de față: vom plăti jucătorul cu $\frac{3}{2} = 1.5$ unități monetare, adică 150% din suma necesară începerii unui joc.

Pentru jocul special, jucătorul câștigă cu o probabilitate de $\frac{1}{3} \cdot \frac{2}{3} = \frac{2}{9}$, adică are nevoie în medie de $\frac{1}{\frac{2}{9}} = \frac{9}{2} = 4.5$ jocuri pentru a câștiga. Așadar, vom plăti jucătorul cu $\frac{4.5}{2} = 2.25$ unități monetare, adică 225% din suma necesară începerii unui joc.

11 Referințe

- Poza de pe copertă este luată de aici: https://cutt.ly/poza-alba-neagra.
- Aplicația folosită pentru desenele grafurilor este https://csacademy.com/app/graph_editor/.
- Pentru desene am folosit https://app.diagrams.net.
- Pentru o parte din ecuații am folosit https://latex.codecogs.com/eqneditor/editor.php
- Pentru a trece proiectul in LaTeX am beneficiat de suportul lui tata (bineînțeles) și https://tex.stackexchange.com/.
- Mulțumiri lui Teodor pentru feedback și ajutor!
- Proiectul scris în Google Docs se poate găsi aici: https://drive.google.com/file/d/1Paj5xer8VRAaooRjjjy_q_FiClobVzmR/view?usp=sharing.