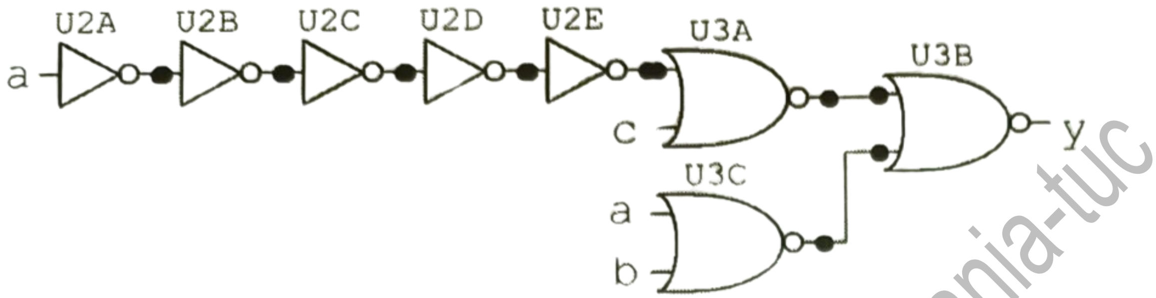
Treść zadań.

Zadanie 1

Zrealizuj układ przedstawiony na rysunku. Wyjście układu podłącz do wejścia zegarowego licznika synchronicznego.



Sprawdź, czy układ działa poprawnie. Jeżeli nie to pokaż sytuacje, w których układ działa niepoprawnie. Sprawdź także czy można wtedy coś poprawić.

Zadanie 2

Zaprojektować na przerzutnikach sr asynchroniczny układ przedstawiony w postaci tablicy przejść/wyjść.

ab	00	01	11	10	LP
0			0	3	10
1	1	1	2	3	00
2		1	2		01
3	1		0	3	00

a) Przypisać następujące kody

0 – 00, 1 – 01, 2 – 11, 3 – 10

Sprawdź, czy układ działa poprawnie. Jeżeli nie, to narysuj jego graf pracy. Sprawdź także czy można przy takim kodowaniu coś poprawić.

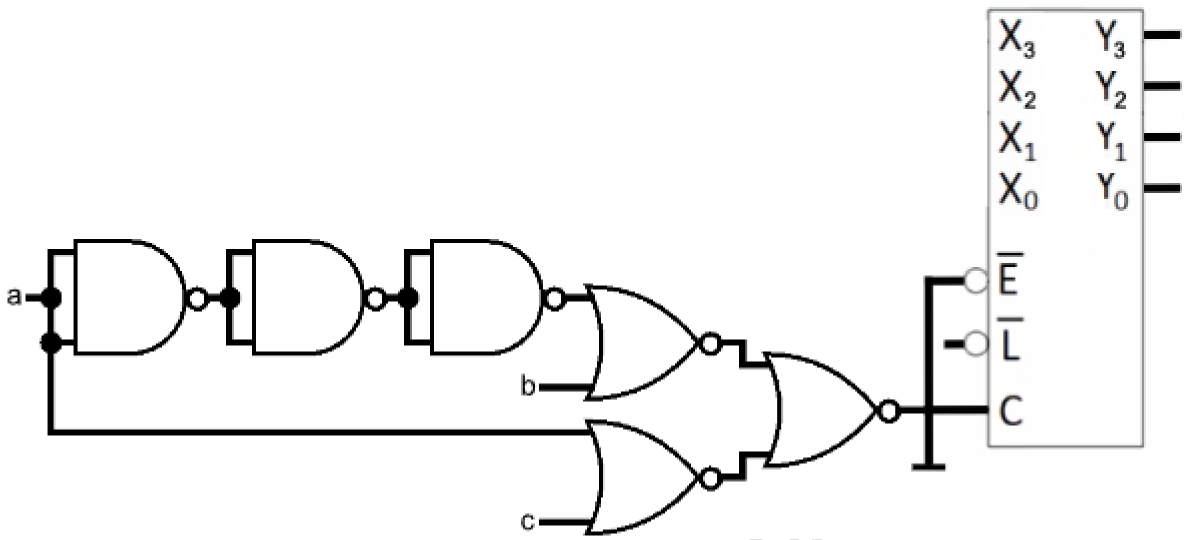
b) Przypisać następujące kody

0 – 00, 1 – 01, 2 – 10, 3 – 11

Sprawdź, czy układ działa poprawnie. Jeżeli nie, to narysuj jego graf pracy. Sprawdź także czy można przy takim kodowaniu coś poprawić.

Zadanie 1

Zbudowaliśmy zadany układ i podłączyliśmy jego wyjście do wejścia zegarowego licznika synchronicznego:



Na postawie układu stworzyliśmy funkcję wyjściową:

$y = \overline{\overline{\overline{a} + c} + \overline{a + b}} = (\overline{a} + c)(a + b)$

Siatka Karnaugha:

ab \ c	0	1
00	0	0
01	1	1
11	0	1
10	0	1

y

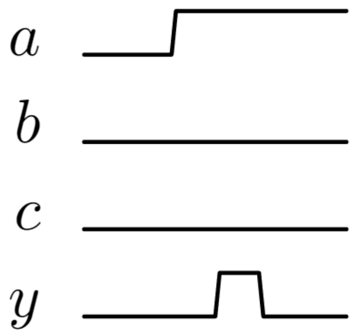
Na postawie zmian na wyjściach licznika doszliśmy do wniosku, że w układzie występuje hazard statyczny w warunkach niedziałania przy zmianie wejść:

abc: $000 \rightarrow 100$

y: $0 \rightarrow 1 \rightarrow 0$

Hazard jest zauważalny ze względu na zwiększony czas propagacji sygnału a, który musi przejść przez 3 bramki NAND (działające w tym przypadku jak bramki NOT) zanim dojdzie do właściwej bramki NOR. Nie podłączaliśmy 5 aż bramek szeregowo, ponieważ licznik zliczał przekłamania wynikające z hazardu już przy 3 bramkach.

Wykres czasowy:

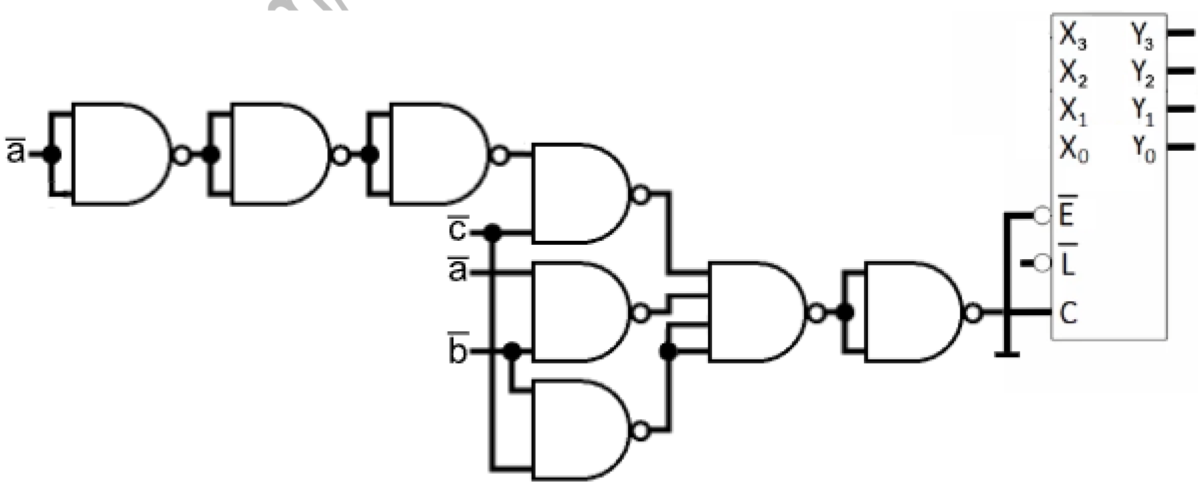


W pozostałych sytuacjach układ działał poprawnie. Aby poprawić działanie układu dodaliśmy grupę antyhazardową $(\mathrm{b}+\mathrm{c})$. Po naszej poprawcę HSn nie występował.

Użyliśmy bramki NAND, ponieważ na naszym stanowisku nie była dostępna 4-wejściowa bramka NOR.

$$y = (\bar{a}+c)(a+b)(\mathrm{b}+\mathrm{c}) = \overline{\overline{(\bar{a}+c)}}\;\overline{\overline{(a+b)}}\;\overline{\overline{(\mathrm{b}+\mathrm{c})}} = \overline{a\bar{c}}\;\overline{\bar{a}\bar{b}}\;\overline{\bar{b}\bar{c}} = \overline{\overline{\overline{a\bar{c}}\;\overline{\bar{a}\bar{b}}\;\overline{\bar{b}\bar{c}}\;1}}$$

Schemat układu:



Zadanie 2

a)

Siatka przejść:

ab

q_1q_2	00	01	11	10
00	--	--	00	10
01	01	01	11	10
11	--	01	11	--
10	01	--	00	10

Q_1Q_2

ab

q_1q_2	00	01	11	10
00	-	-	0	**1**
01	0	0	**1**	**1**
11	-	**0**	1	-
10	**0**	-	**0**	1

Q_1

$$s_1 = a\overline{b} + q_2a$$
$$r_1 = \overline{a} + b\overline{q_2}$$

$$\overline{s_1} = \overline{a\overline{b} + q_2a} = \overline{a\overline{b}}\,\overline{q_2a} = \overline{\overline{\overline{a\overline{b}}\,\overline{q_2a}}}$$
$$\overline{r_1} = \overline{\overline{a} + b\overline{q_2}} = \overline{ab\overline{q_2}} = \overline{\overline{a1\cdot\overline{b\overline{q_2}}}}$$

ab

q_1q_2	00	01	11	10
00	-	-	0	0
01	1	1	1	**0**
11	-	1	1	-
10	**1**	-	0	0

Q_2

$$s_2 = \overline{a}$$
$$r_2 = a\overline{b}$$

$$\overline{s_2} = a$$
$$\overline{r_2} = \overline{a\overline{b}}$$

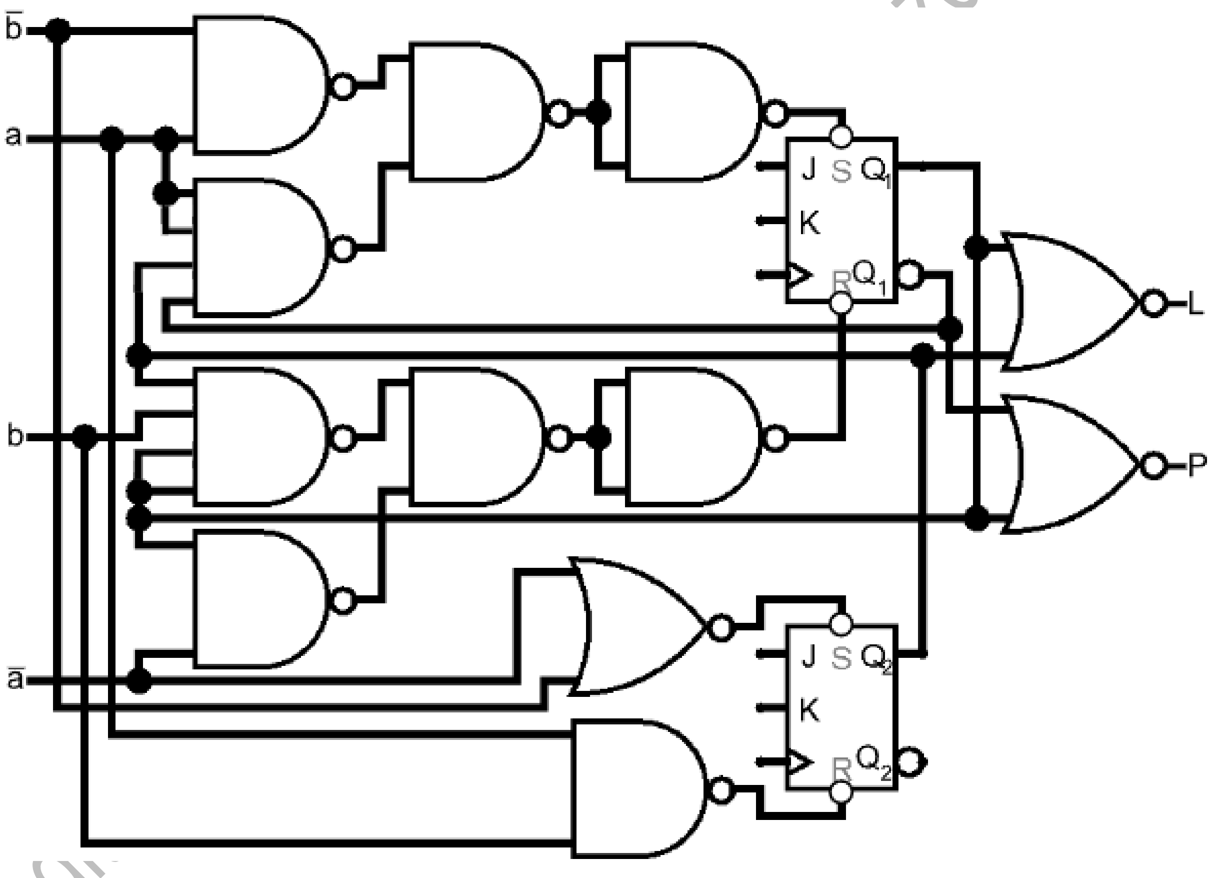
Siatka Karnaugha dla wyjść:

q_1 \ q_2	0	1
0	10	00
1	00	01

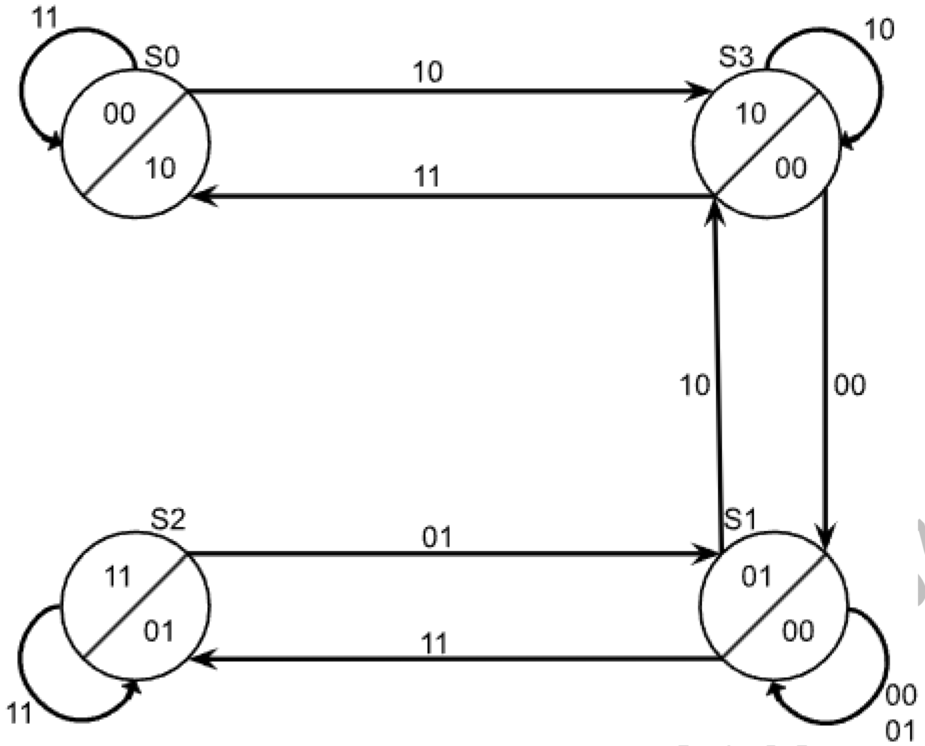
LP

$$L = \overline{q_1}\,\overline{q_2} = \overline{\overline{\overline{q_1}\,\overline{q_2}}} = \overline{q_1 + q_2}$$
$$P = q_1 q_2 = \overline{\overline{q_1 q_2}} = \overline{\overline{q_1} + \overline{q_2}}$$

Schemat układu:



Graf pracy:



Podczas laboratorium myśleliśmy, że problemem w układzie jest to, że w stanie pierwszym przy zmianie wejść $ab = 00$ na $ab = 11$ w zależności i od tego czy zmieni się najpierw a czy b układ trafi od stanu 2 lub stanu 0.

Jeżeli:

$ab: 00 \to 01 \to 11$

to:

$S: 1 \to 1 \to 2$

Jeżeli:

$ab: 00 \to 10 \to 11$

to:

$S: 1 \to 3 \to 0$

Jednak nie jest błąd układu, tylko jego funkcjonalność zgodna z tablicą przejść.

Po narysowaniu grafu pracy układu zauważyliśmy, że wyścig krytyczny może wystąpić przy zmianie stanu 1 na stan 3 lub stanu 3 na stan 1, ponieważ wtedy zmieniają się dwa bity stanów jednocześnie.

$S_1 \to S_3$

$q_1q_2: 01 \to 10$

$S_3 \rightarrow S_1$

$q_1 q_2 \colon 10 \rightarrow 01$

Jeżeli któryś z bitów adresowych zmieniłby się pierwszy, to układ trafiłby do stanu 0 lub stanu 2.

Nie udało nam się zaobserwować błędów w działaniu układu przy przejściach między stanem 1 i stanem 3 podczas laboratorium. Jeśli jednak pojawiłyby się takie problemy, konieczne byłoby wprowadzenie dodatkowych przejść w siatce przejść, aby układ zawsze działał poprawnie.

Skorygowana tablica przejść z podkreśloną korektą:

		ab				
	$q_1 q_2$	00	01	11	10	
S_0	00	$\underline{01}$	--	00	10	
S_1	01	01	01	11	10	
S_2	11	$\underline{01}$	01	11	$\underline{10}$	
S_3	10	01	--	00	10	$Q_1 Q_2$

Wtedy nawet w przypadku niechcianej zmiany na stan 0 lub stan 2 układ ostatecznie trafiłby do oczekiwanego stanu. Należałoby odpowiednio zmienić funkcje wejść dla przerzutników typu sr. W naszym przypadku nie musieliśmy wprowadzać żadnych zmian, ponieważ projektując układ w ramach optymalizacji wykorzystaliśmy stany nieokreślone, dobierając grupy dla warunków działania w taki sposób, że skonstruowany układ został już skorygowany.

b)

Siatka przejść:

q_1q_2 \ ab	00	01	11	10
00	--	--	00	11
01	01	01	10	11
10	--	01	10	--
11	01	--	00	11

Q_1Q_2

q_1q_2 \ ab	00	01	11	10
00	--	--	00	11
01	01	01	10	11
11	01	--	00	11
10	--	01	10	--

Q_1Q_2

Aby poprawnie wyznaczyć funkcje należy zamienić miejscami stan 2 i 3, żeby kody stanów ułożone były zgodnie z kodem Graya

q_1q_2 \ ab	00	01	11	10
00	-	-	0	**1**
01	0	0	**1**	**1**
11	**0**	-	**0**	1
10	-	**0**	1	-

Q_1

$$s_1 = a\overline{b} + \overline{q_1}q_2a$$
$$r_1 = q_1\overline{a} + q_1q_2b$$

$$\overline{s_1} = \overline{a\overline{b} + \overline{q_1}q_2a} = \overline{a\overline{b}}\;\overline{\overline{q_1}q_2a} = \overline{\overline{\overline{a\overline{b}}\;\overline{\overline{q_1}q_2a1}}}$$
$$\overline{r_1} = \overline{q_1\overline{a} + q_1q_2b} = \overline{q_1\overline{a}}\;\overline{q_1q_2b} = \overline{\overline{\overline{q_1\overline{a}}\;\overline{q_1q_2b1}}}$$

q_1q_2 \ ab	00	01	11	10
00	-	-	0	**1**
01	1	1	**0**	1
11	1	-	**0**	1
10	-	**1**	0	-

Q_2

$$s_2 = \overline{a} + \overline{b}$$
$$r_2 = ab$$

$$\overline{s_2} = \overline{\overline{a} + \overline{b}}$$
$$\overline{r_2} = \overline{ab}$$

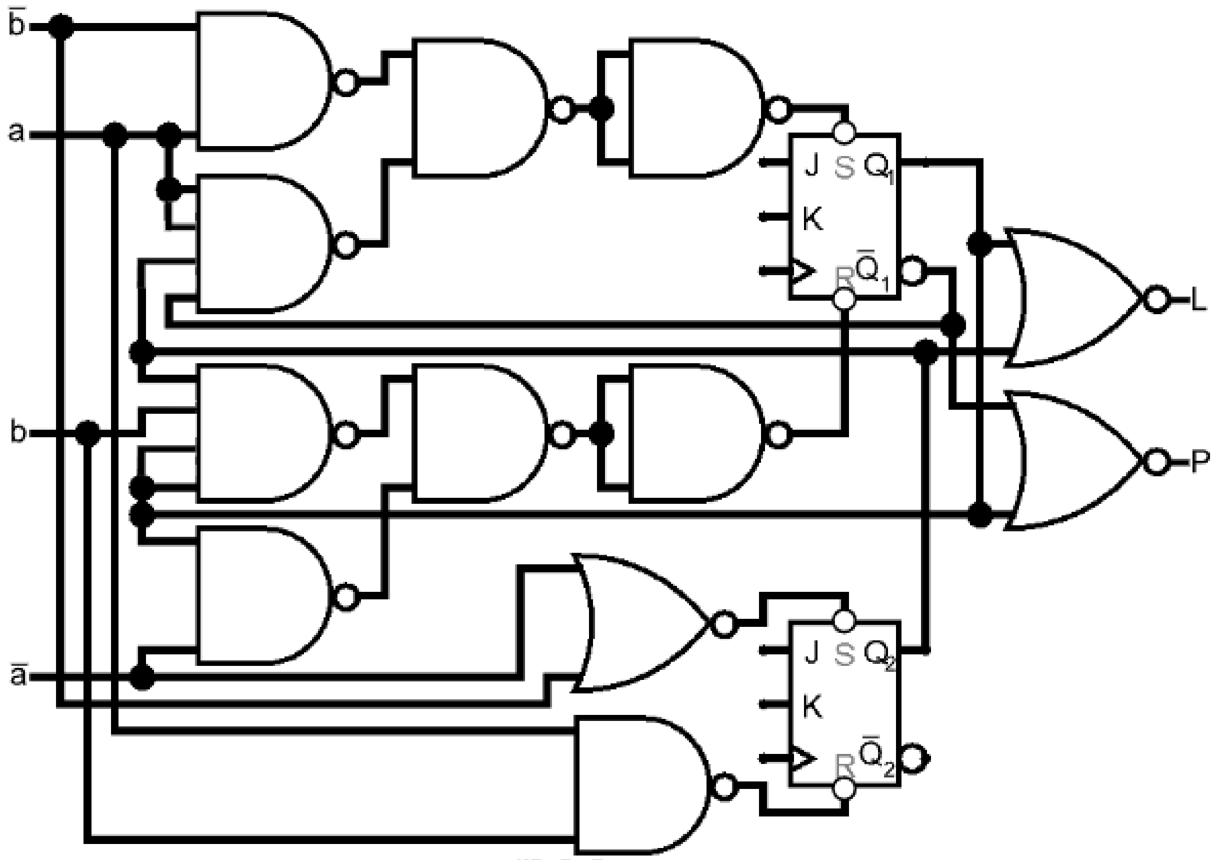
Siatka Karnaugha dla wyjść:

q_1 \ q_2	0	1
0	10	00
1	01	00

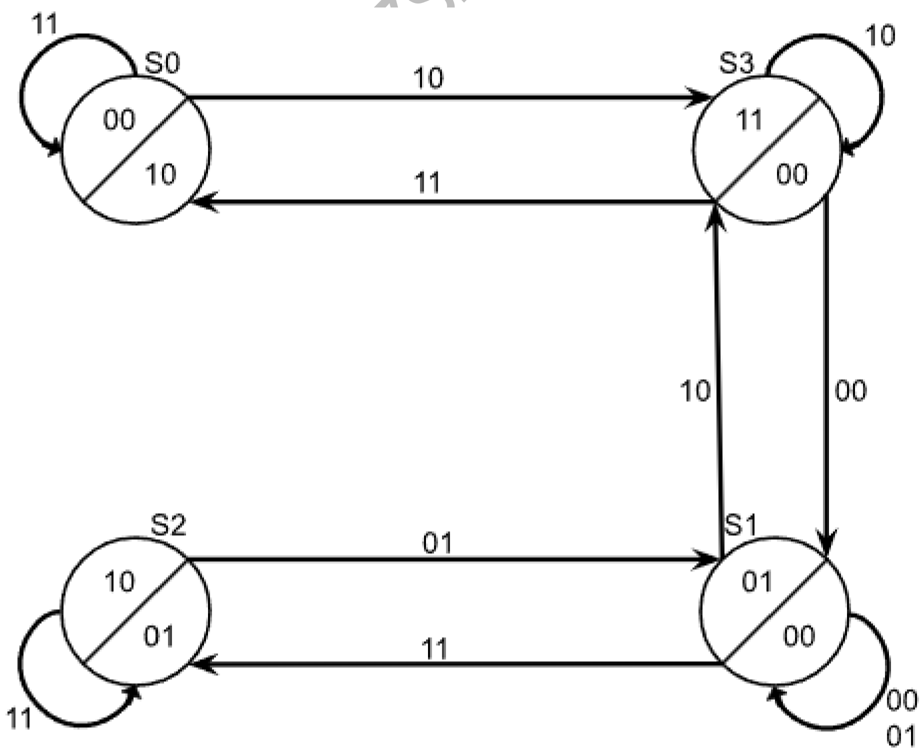
LP

$$L = \overline{q_1}\,\overline{q_2} = \overline{\overline{\overline{q_1}\,\overline{q_2}}} = \overline{q_1 + q_2}$$
$$P = q_1\overline{q_2} = \overline{\overline{q_1\overline{q_2}}} = \overline{\overline{q_1} + q_2}$$

Schemat układu:



Graf pracy:



Tego układu nie zdążyliśmy przetestować na laboratorium. Na podstawie grafu pracy można zauważyć przejścia, w których mógłby wystąpić wyścig krytyczny:

$S_0 \rightarrow S_3$

q_1q_2: $00 \rightarrow 11$

$S_3 \rightarrow S_0$

q_1q_2: $00 \rightarrow 11$

$S_1 \rightarrow S_2$

q_1q_2: $01 \rightarrow 10$

$S_2 \rightarrow S_1$

q_1q_2: $01 \rightarrow 10$

ab

	q_1q_2	00	01	11	10	
S_0	00	--	$\underline{01}$	00	11	
S_1	01	01	01	10	11	
S_3	11	01	$\underline{01}$	00	11	
S_2	10	--	01	10	$\underline{11}$	Q_1Q_2

Można zapobiec wyścigom przy przejściach:

$S_0 \rightarrow S_3$

$S_2 \rightarrow S_1$

Nie można zapobiec wyścigom bez zmiany kodowania przy przejściach:

$S_3 \rightarrow S_0$

$S_1 \rightarrow S_2$

Tak samo jak w podpunkcie a) nie można już nic poprawić w układzie, jeżeli przy projektowaniu wykorzystało się stany nieokreślone do dobrania najprostszych grup w warunkach działania.

Wnioski

Podczas laboratorium zbudowaliśmy, uruchomiliśmy i przetestowaliśmy opisane układy z wyjątkiem układu w podpunkcie b) w zadaniu 2. Działały tak jak opisaliśmy w poszczególnych zadaniach. Czas propagacji sygnałów może wpływać na działanie układów cyfrowych. Hazardy i wyścigi krytyczne są ważnymi aspektami wpływającymi na stabilność układów. Aby zapobiec hazardowi można wprowadzić gry antyhazardowe. Aby zapobiec wyścigom w asynchronicznych układach sekwencyjnych można ustalić kodowanie w ten sposób, aby w przejściach między stanami zmieniał się tylko jeden bit jednocześnie. Jeżeli nie da się w taki sposób ustalić kodowania, można wprowadzić stany pośrednie.