Projektowanie systemów cyfrowych

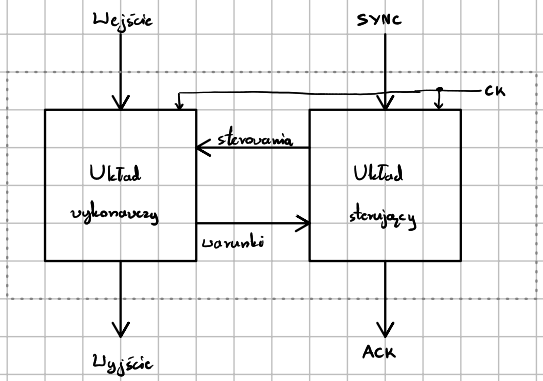
Układ wykonawczy (operacyjny)
- wykonuje określone operacje
- obsługuje wejście/wyjście

Układ sterujący
- kontroluje kolejność wykonywania operacji
- warunki jako wejścia
- sterowania jako wyjścia

Może być zrealizowany jako
- zwykły automat
- rozdzielacz sterujący

} mniej wygodne

- układ mikroprogramowany w pamięci ROM



Układy komunikują się sygnałami sterowań i warunków

Układy są taktowane wspólnym zegarem
Można (albo trzeba) zastosować zegar dwufazowy
- jeden z układów otrzymuje zanegowany sygnał zegarowy
- układ sterowania efektywnie otrzymuje sygnał opóźniony o pół taktu
- wymusza kolejność między układami

Jest wymagany w automatach Moore'a, gdzie w komórce warunkowej sprawdza się warunek ustawiony w ostatniej komórce operacyjnej

Sieć działań algorytmu

Schemat blokowy, składa się z klatek operacyjnych i warunkowych, kroków Start i Stop.

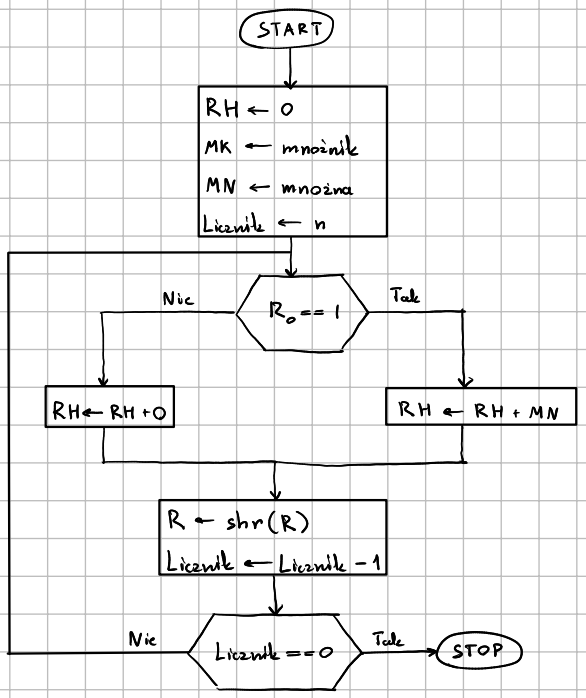
Klatka warunkowa
- warunkowe rozgałęzienie

Klatka operacyjna
- synchroniczne operacje
- wszystkie operacje wykonują się w 1 takcie zegara

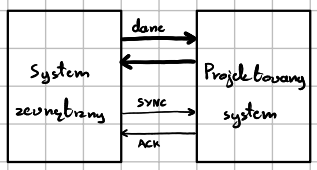
Sieć nie zawiera operacji kombinacyjnych, tylko synchroniczne
(np. nie dodawanie ale zapisanie wyniku dodawania do rejestru)

Zamiast sprawdzać kilka warunków na raz, można ustawić pomiędzy sprawdzeniu klatkę operacyjną, która nie wykonuje żadnej operacji (NOP - no operation)

Przykładowa sieć działań dla algorytmu mnożenia



Przeplot (handshaking)

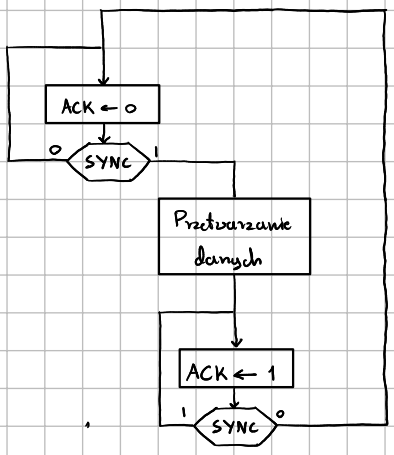


Prosty protokół komunikacyjny, umożliwia komunikację systemu z otoczeniem
zapewnia poprawność odczytywanych i zapisywanych danych

SYNC - synchronize - jest gotowe nowe wejście
ACK - acknowledge - jest gotowe wyjście

Może być zrealizowany synchronicznie lub asynchronicznie

Konieczny przy każdej próbie odczytu wyjść
Odczyt szeregowy - pętla przeplotu przy odczycie każdej danej



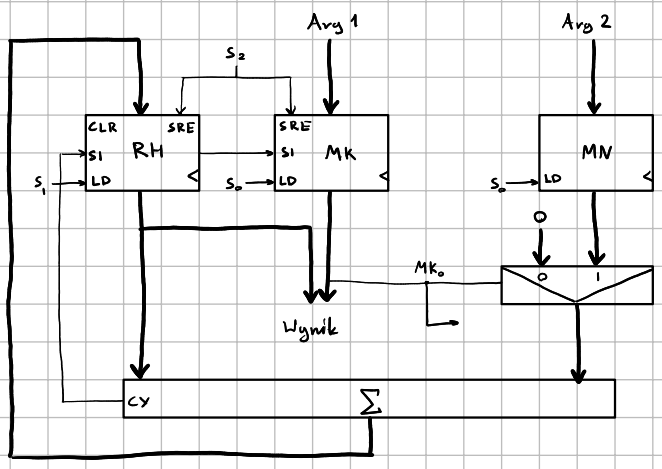
Produkowanie sygnału ACK
np. przerzutnik asynchroniczny $\overline{RS}$
na podstawie sygnałów sterowania
synchroniczne lub asynchroniczne

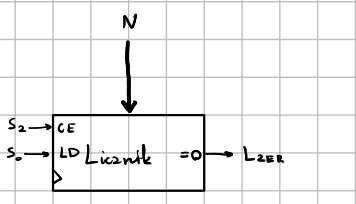
Zewnętrzny	Oczekiwanie	Gotowe dane na wyjściu		Odebrany wynik	Oczekiwanie
Projektowany	Oczekiwanie	Przetwarzanie danych	Gotowy wynik		Oczekiwanie

Projekt układu wykonawczego dla algorytmu mnożenia

- Rejestr na wynik i mnożnik R
- Rejestr na mnożną
- Licznik pętli
- Sumator - kombinacyjny
- Multiplekser - rozgałęzienia warunkowe

Trzeba ustalić szerokość szyn danych



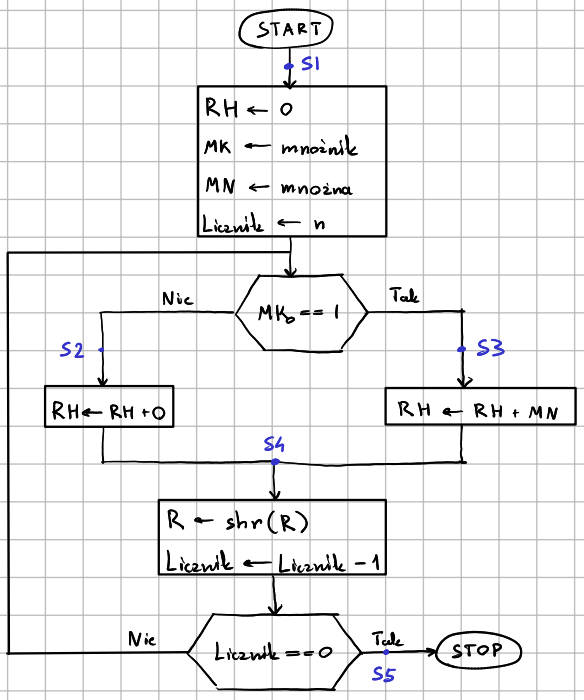


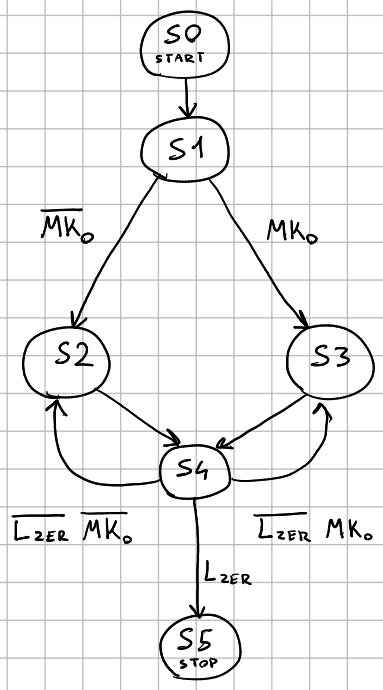
MK_0, L_{ZER} - warunki dla układu sterowania
S_0, S_1, S_2 - sygnały od układu sterowania
SRE - shift right enable
CE - count enable

Realizacja układu sterowania

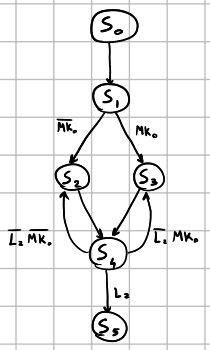
Sieć działań → graf automatu → tabela automatu

Stworzenie grafu automatu na podstawie sieci działań
- ustawić stan przed każdą klatką operacyjną





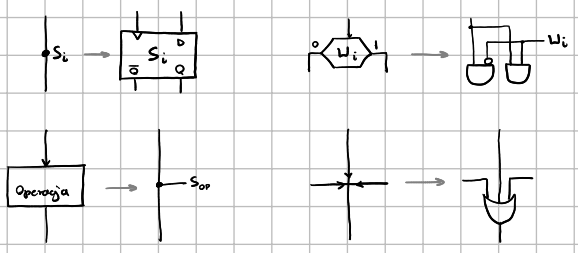
Stworzenie tabeli na podstawie grafu



S \ $L_2 MK_0$	00	01	11	10	S_2	S_1	S_0	Wyjście
0	1	1	1	1	0	0	0	
1	2	3	3	2	0	0	1	Ładowanie MK, MN ; licznika, zerowanie RH
2	4	4	4	4	0	0	0	Ładowanie RH+0 do RH
3	4	4	4	4	0	1	0	Ładowanie RH+MN do RH
4	2	3	5	5	1	0	0	Przesunięcie R w prawo, dekrementacja licznika
5	5	5	5	5	0	0	0	

Realizacja jako zwykły automat
- Każda zmiana w projekcie wymaga zaprojektowania automatu od nowa
- Niepraktyczne, niewygodne rozwiązanie

Realizacja jako rozdzielacz sterujący
- Zakodowanie stanów w kodzie 1 z n
- Stworzenie schematu na podstawie sieci działań według schematu
- Tylko 1 z przerzutników D zawiera wartość 1 w każdym momencie
- Musi zostać zainicjowany w odpowiednim stanie po przywróceniu zasilania

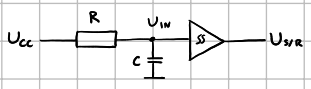


Układ inicjacji z bramką Schmitta

Zapewnia, że układ przyjmie określony stan przy włączeniu zasilania

Układ RC zapewnia stopniowe narastanie napięcia.

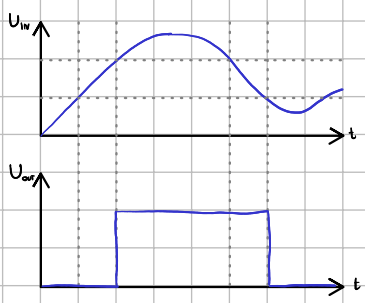
Uzyskane opóźnienie zostawia czas na ustawienie przerzutników w pożądany stan

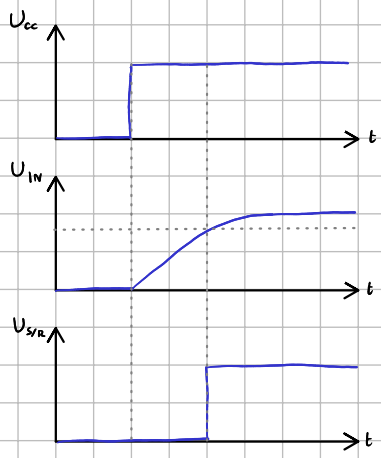


Najpierw wszystkie przerzutniki się zerują.

Po czasie, do pierwszego przerzutnika zaczyna się 1.

Bramka Schmitta - histereza





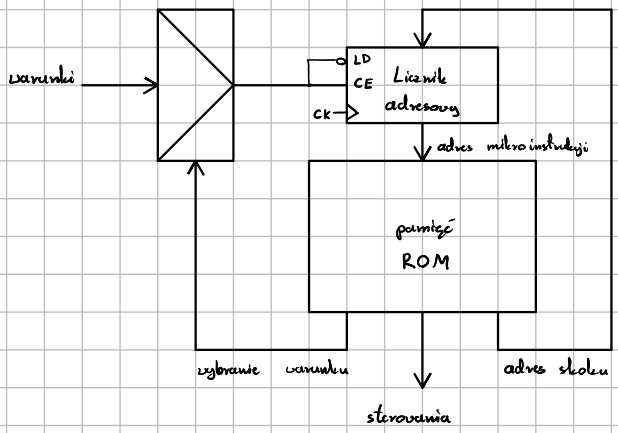
Mikroprogramowany układ sterujący

realizuje automat synchroniczny

Pamięć ROM zawiera mikroinstrukcje mikroprogramu.

Mikroinstrukcja składa się z

- kodu warunku
- wyjścia - sygnałów sterowania dla układu wykonawczego
- adresu skoku warunkowego



Multiplekser

- wybiera za pomocą kodu sygnał warunku z układu wykonawczego
- wyjście steruje licznikiem

Licznik adresowy

- przechowuje adres wykonywanej mikroinstrukcji
- warunkowo albo się inkrementuje - następna instrukcja albo wczytuje adres - skok warunkowy

Zapisanie mikroprogramu na podstawie tabeli automatu

SYNC, L_2, MK_0 STAN	000	001	010	011	100	101	110	111	S_3	S_2	S_1	S_0
1	1	1	1	1	2	2	2	2	0	0	0	1
2	4	3	4	3	4	3	4	3	0	0	0	0
3	4	4	4	4	4	4	4	4	0	0	1	0
4	2	2	5	5	2	2	5	5	0	1	0	0
5	1	1	1	1	5	5	5	5	1	0	0	0

1. 0001; if SYNC then goto 2 else goto 1
2. 0000; if MK_0 then goto 3 else goto 4
3. 0010; if TRUE then goto 4 else goto -
4. 0100; if L_2 then goto 5 else goto 2
5. 1000; if SYNC then goto 5 else goto 1

Budowa układu mikroprogramowanego wymaga instrukcji typu

X. wyjście; if warunek then następna instrukcja else goto adres skoku

Trzeba zmodyfikować instrukcję

5. 1000; if SYNC then goto 5 else goto 1

↓

5. 1000; if $\overline{SYNC}$ then goto 6 else goto 5
6. 0000; if FALSE then goto - else goto 1

w klasycznej wersji może być tylko 1 skok

Mikroprogram w pamięci ROM

Warunkom i adresom trzeba przypisać kody

oddzielne dla warunków X i not X

1. 0001; if SYNC then goto 2 else goto 1
2. 0000; if MK_0 then goto 3 else goto 4
3. 0010; if TRUE then goto 4 else goto -
4. 0100; if L_2 then goto 5 else goto 2
5. 1000; if $\overline{SYNC}$ then goto 6 else goto 5
6. 0000; if FALSE then goto - else goto 1

Warunek	Kod
SYNC	000
not SYNC	001
MK_0	010
L_2	011
TRUE	100
FALSE	101

Adres	Kod
1	000
2	001
3	010
4	011
5	100
6	101

Zawartość ROM

adres instrukcji			warunki			operacje				adres skoku		
			w_2	w_1	w_0	s_3	s_2	s_1	s_0	a_2	a_1	a_0
0	0	0	0	0	0	0	0	0	1	0	0	0
0	0	1	0	1	0	0	0	0	0	0	1	1
0	1	0	1	0	0	0	0	1	0	-	-	-
0	1	1	0	1	1	0	1	0	0	0	0	1
1	0	0	0	0	1	1	0	0	0	1	0	0
1	0	1	1	0	1	0	0	0	0	0	0	0

Modyfikacje klasycznego układu

Dla specyficznych potrzeb projektu można wprowadzić modyfikacje do klasycznego układu

Można np. wprowadzić kompresję mikroinstrukcji w celu zaoszczędzenia pamięci

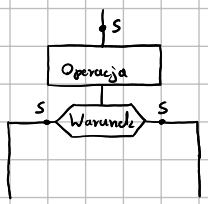
Przykłady

- multiplekser adresowy - więcej możliwości warunkowych skoków
- układ modyfikacji adresu } dekodery do skompresowanych
- układ modyfikacji sterowań } adresów sterowań

Realizacja automatów Moore'a i Mealy'ego

Automat Moore'a

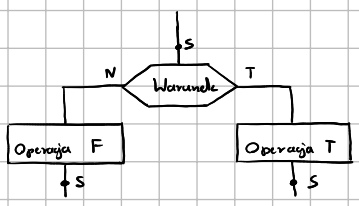
- stan przed klatką operacyjną
- pętla musi zawierać stan



Adres: operacja; if warunek
then goto adres + 1
else goto adres skoku

Automat Mealy'ego

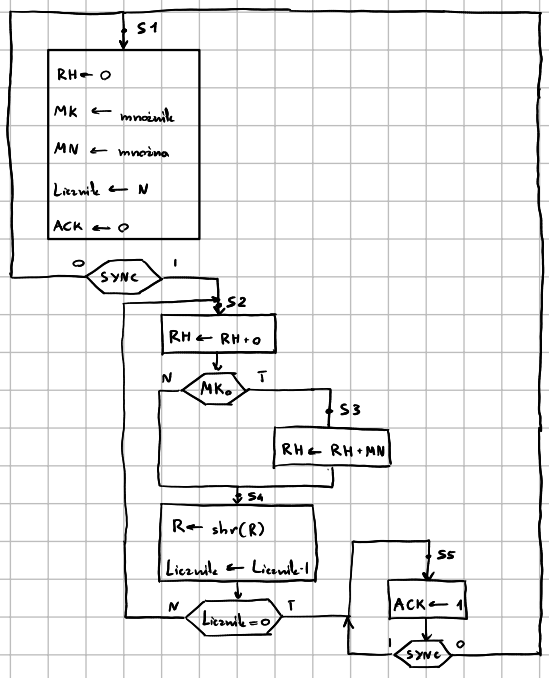
- stan przed klatką warunkową
- klatki operacyjne muszą być rozdzielone stanem



Adres: if warunek
then operacja T; goto adres + 1
else operacja F; goto adres skoku

Układ mnożący w NKB, z przeplotem

realizacja



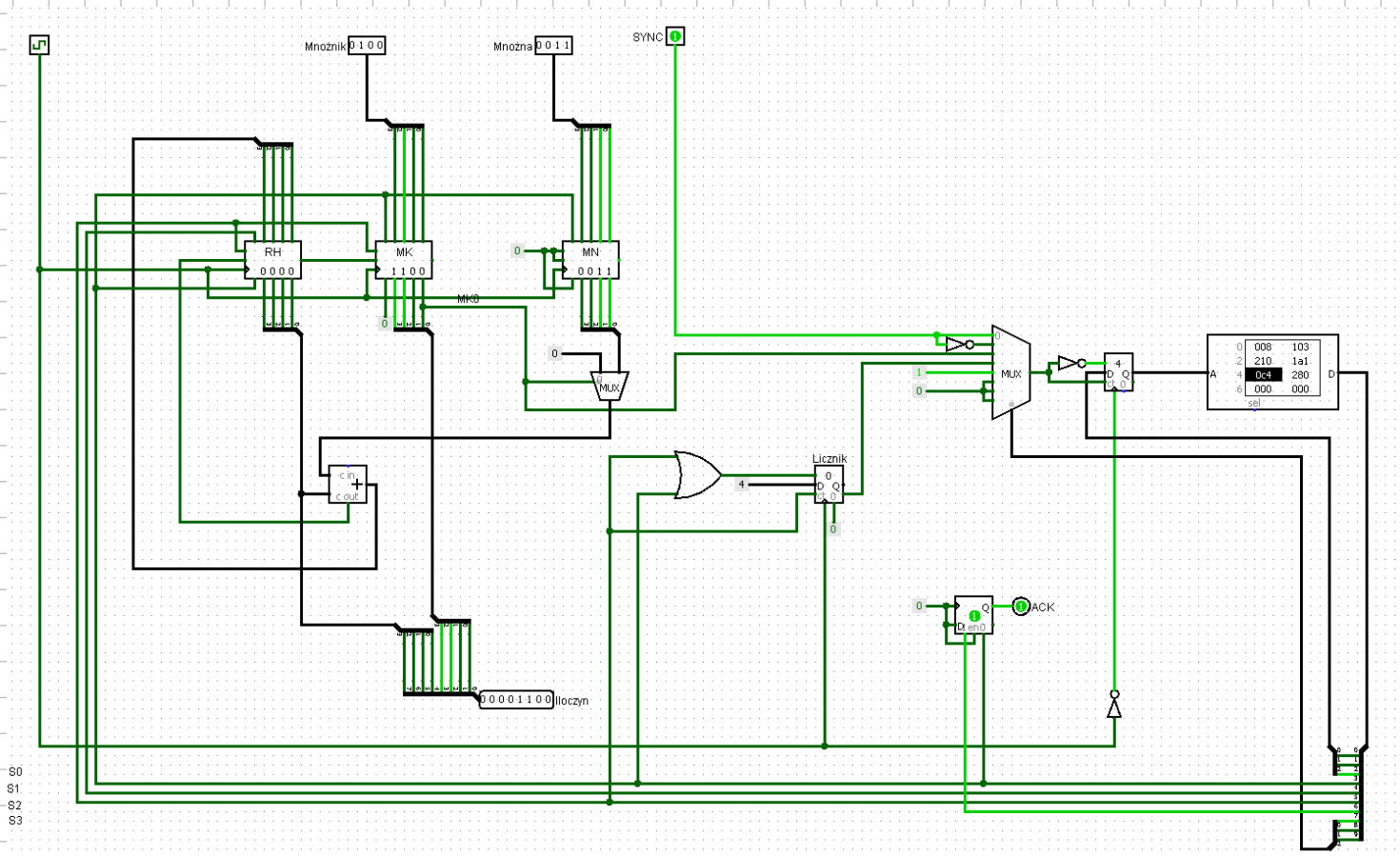
STAN	SYNC, L2, MK0 = 000	001	010	011	100	101	110	111	S3	S2	S1	S0
1	1	1	1	1	2	2	2	2	0	0	0	1
2	4	3	4	3	4	3	4	3	0	0	0	0
3	4	4	4	4	4	4	4	4	0	0	1	0
4	2	2	5	5	2	2	5	5	0	1	0	0
5	1	1	1	1	5	5	5	5	1	0	0	0

1. 0001; if SYNC then goto 2 else goto 1
2. 0000; if MK_0 then goto 3 else goto 4
3. 0010; if TRUE then goto 4 else goto -
4. 0100; if L_2 then goto 5 else goto 2
5. 1000; if $\overline{SYNC}$ then goto 6 else goto 5
6. 0000; if FALSE then goto - else goto 1

adres instrukcji			warunki W_2	W_1	W_0	operacje S_3	S_2	S_1	S_0	adres skoku a_2	a_1	a_0	HEX
0	0	0	0	0	0	0	0	0	1	0	0	0	0x008
0	0	1	0	1	0	0	0	0	0	0	1	1	0x103
0	1	0	1	0	0	0	0	1	0	-	-	-	0x210
0	1	1	0	1	1	0	1	0	0	0	0	1	0x1A1
1	0	0	0	0	1	1	0	0	0	1	0	0	0x0C4
1	0	1	1	0	1	0	0	0	0	0	0	0	0x280

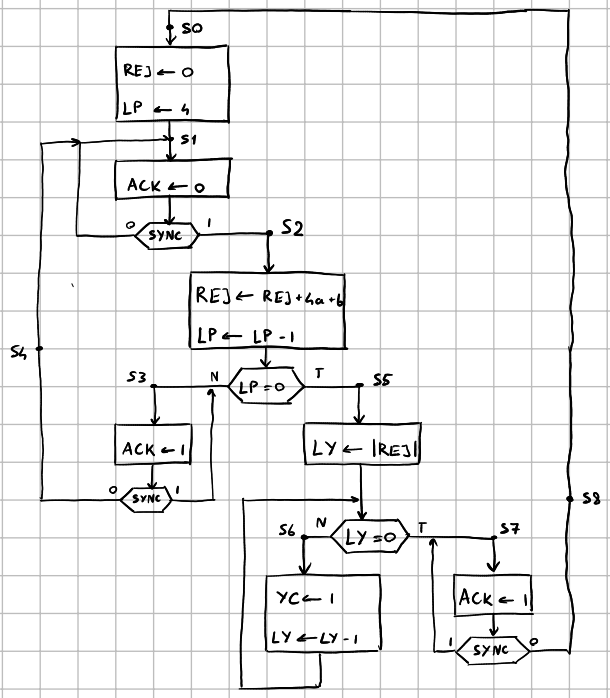
Warunek	Kod
SYNC	000
not SYNC	001
MK_0	010
L_2	011
TRUE	100
FALSE	101

Adres	Kod
1	000
2	001
3	010
4	011
5	100
6	101



Projekt układu

- dostaje na wyjściu szeregowo 4 pary liczb U2 (a_i, b_i)
- oblicza wartość $Y = \sum_{i=1}^{4} (4a_i + b_i)$
- przepuszcza na wyjście $|Y|$ impulsów zegara
- wykonuje przeplot

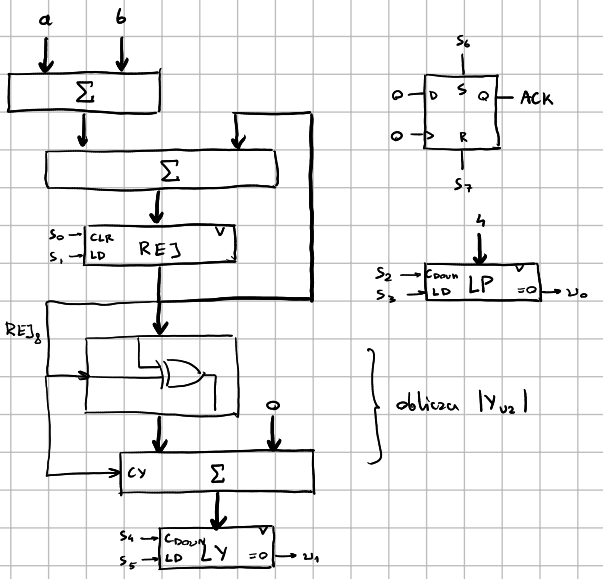


0. REJ ← 0; LP ← 4;
1. ACK ← 0;
2. REJ ← REJ + 4a + b; LP ← LP - 1;
3. ACK ← 1;
4. NOP
5. LY ← |REJ|
6. YC ← 1; LY ← LY - 1;
7. ACK ← 1;
8. NOP

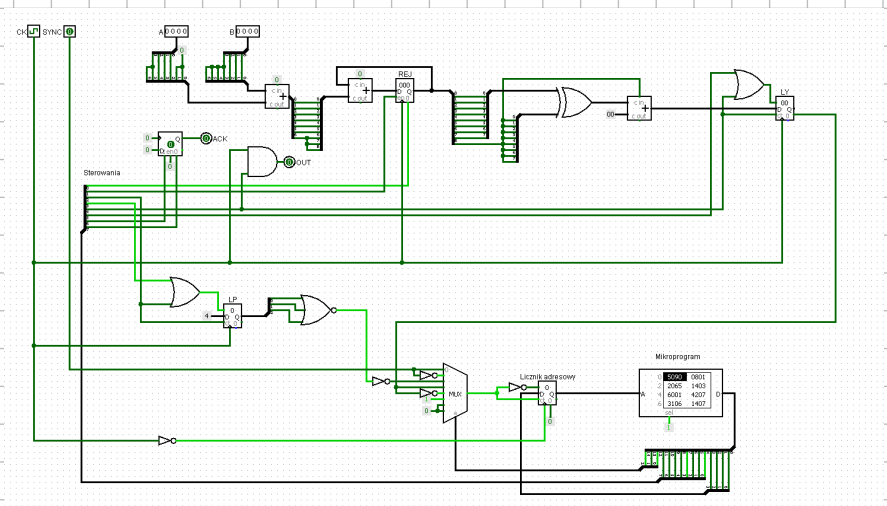
if TRUE	goto 1	else	-
if SYNC	goto 2	else	goto 1
if not LP=0	goto 3	else	goto 5
if not SYNC	goto 4	else	goto 3
if FALSE	-	else	goto 1
if not LY=0	goto 6	else	goto 7
if LY=0	goto 7	else	goto 6
if not SYNC	goto 8	else	goto 7
if False	-	else	goto 0

Warunek	Kod
SYNC	000
not SYNC	001
not W_0	010
W_1	011
not W_1	100
TRUE	101
FALSE	110

Adres	Kod
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000



numer	u_2	u_1	u_0	s_7	s_6	s_5	s_4	s_3	s_2	s_1	s_0	a_3	a_2	a_1	a_0	HEX
0000	1	0	1	0	0	0	0	1	0	0	1	0	0	0	0	0x5090
0001	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0x0801
0010	0	1	0	0	0	0	0	0	1	1	0	0	1	0	1	0x2065
0011	0	0	1	0	1	0	0	0	0	0	0	0	0	1	1	0x1403
0100	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0x6001
0101	1	0	0	0	0	1	0	0	0	0	0	0	1	1	1	0x4207
0110	0	1	1	0	0	0	1	0	0	0	0	0	1	1	0	0x3106
0111	0	0	1	0	1	0	0	0	0	0	0	0	1	1	1	0x1407
1000	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0x6000



bity liczb U2 są rozszerzone
mnożenie przez przesunięcie bitowe