

Projektowanie systemów cyfrowych

Układ wykonawczy (operacyjny)

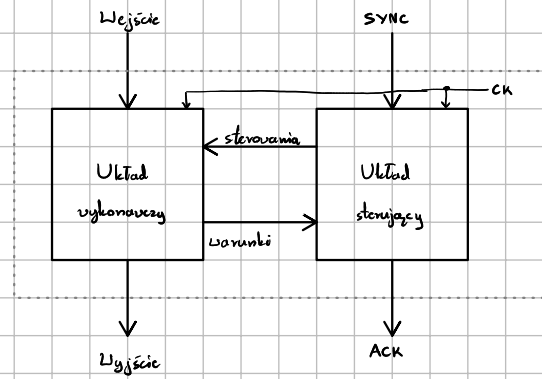
- wykonuje określone operacje
- obsługuje wejście/wyjście

Układ sterujący

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

Może być zrealizowany jako

- zwykły automat
 - rozdzielacz sterujący
 - układ mikroprogramowany w pamięci ROM
- } mniej wygodne



Układy komunikują się sygnałami sterowania i warunków

Układy są taktowane wspólnym zegarem

Można (albo trzeba) zastosować zegar dwufazowy

- jeden z układów otrzymuje zamegowany 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 warunki ustawione w ostatniej komórce operacyjnej

Siec działani algorytmu

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

Klatka warunkowa

- warunkowe rozgałęzienie

Klatka operacyjna

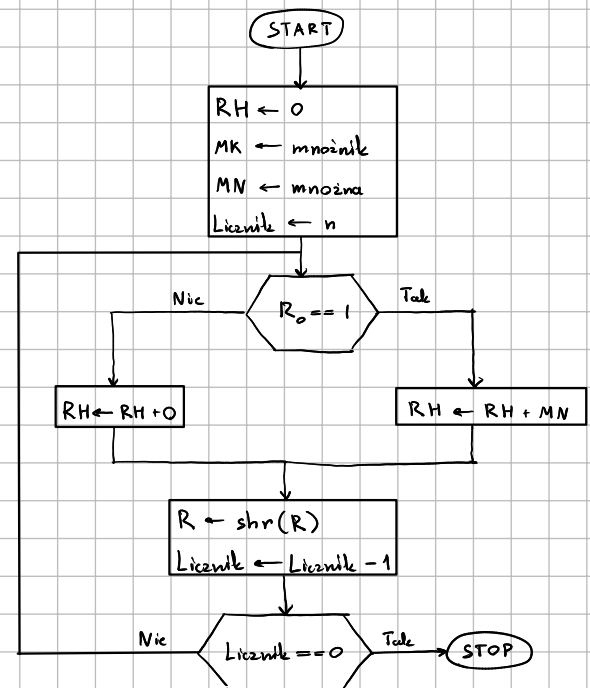
- synchronizacja operacji
- wszystkie operacje wykonują się w 1 taktie zegara

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

Zamiast sprawdzać kilka warunków na raz, można ustawić

pośrodku sprawdzenia klatek operacyjnych, 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 - synchronizacja - jest gotowe nowe wejście

ACK - acknowledge - jest gotowe wyjście

Może być zrealizowany synchronicznie lub asynchronicznie

Konieczny przy każdej próbie odczytu wejście

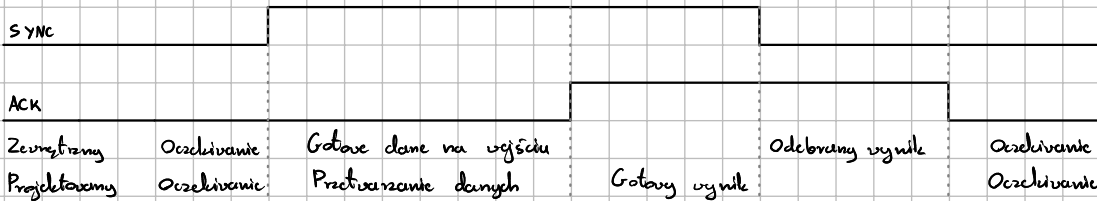
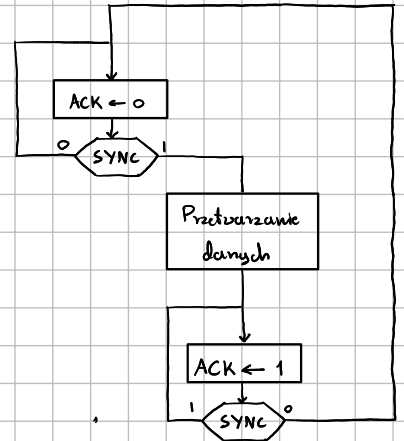
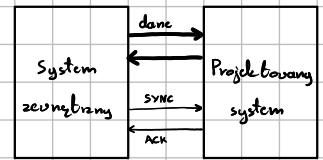
Odczyt szeregowy - pętla przeplotu przy odczycie każdej danej

Produkowanie sygnału ACK

np. przerzutnik asynchroniczny \overline{RS}

na podstawie sygnałów sterowania

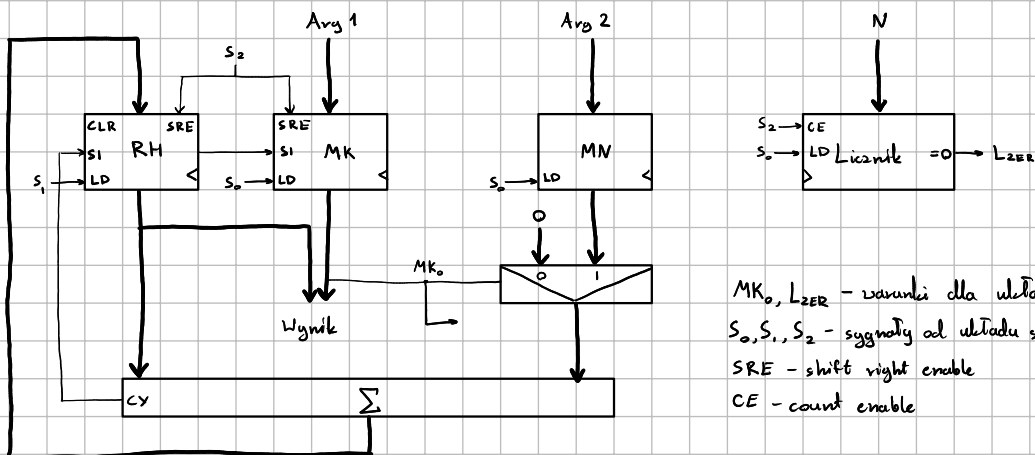
synchroniczne lub asynchroniczne



Projekt układu wykonawczego dla algorytmu mnożenia

- Rejestr na wynik i mnożnik R
- Rejestr na mnożną
- Licznik pętli
- Sumator - kombinacyjny
- Multiplexer - rozgłaszanie warunków

Trzeba ustalić szerokość szyn danych



$MK_0, LZER$ - warunki dla układu sterowania

S_0, S_1, S_2 - sygnały od układu sterowania

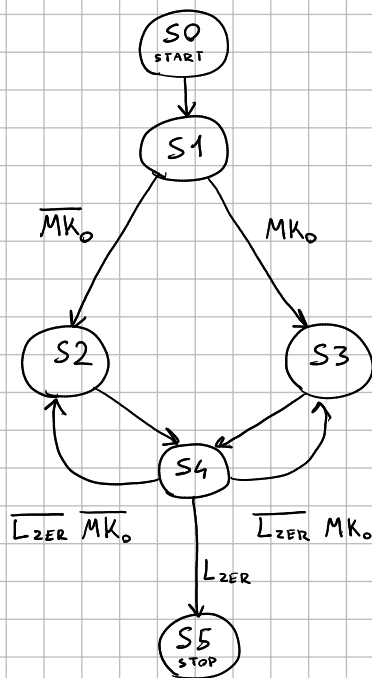
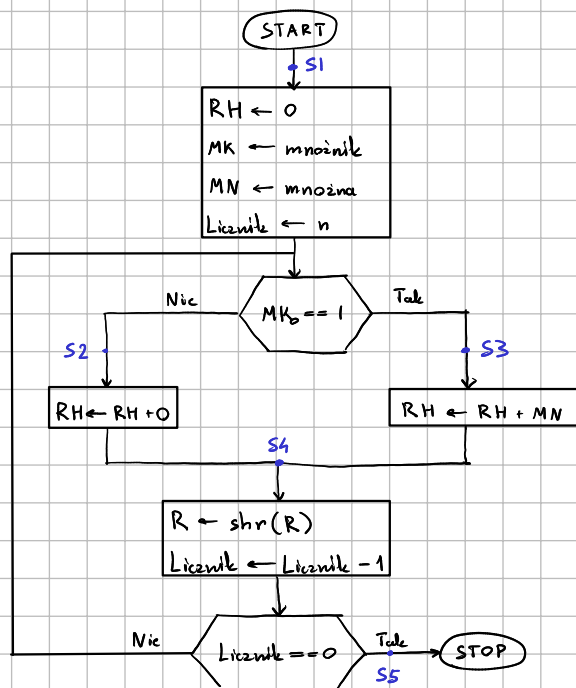
SRE - shift right enable

CE - count enable

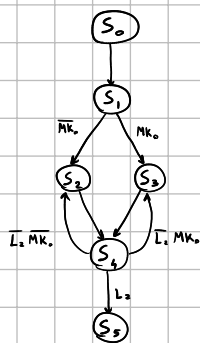
Realizacja układu sterowania

Ścieżki działań → graf automatu → tabela automatu

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



Stworzenie tabeli na podstawie grafu



L.MK S	00	01	11	10	S₂	S₁	S₀	Wyjście
0	1	1	1	1	0	0	0	
1	2	3	3	2	0	0	1	Ładowanie MK, MN i 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, decrementacja 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

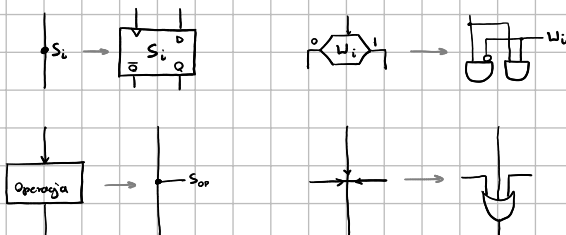
Realizacja jako rozdzielacz sterujący

Zakodowanie stanów w kodzie 12n

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

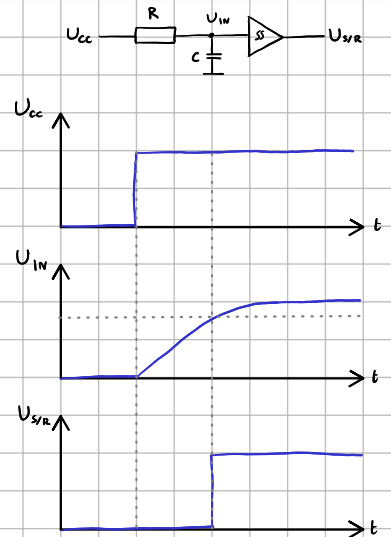
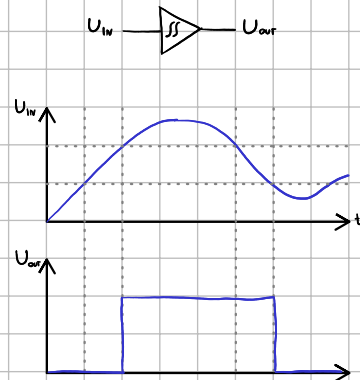
Układ RC zapewnia stopniowe narastanie napięcia.

Użytkownik opóźnienie zostawia czas na ustawienie przerzutników w pożądanym stanie

Najpierw wszystkie przerzutniki się zerują.

Po czasie, do pierwszego przerzutnika załaduje się 1

Bramka Schmitta - histereza



Mikroprogramowany układ sterujący realizuje automat synchroniczny

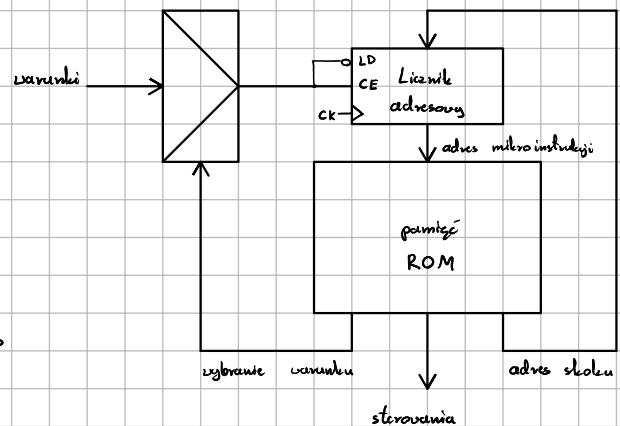
Pamięć ROM zawiera mikroinstrukcje mikroprogramu.

Mikroinstrukcja składa się z

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

Multiplexer

- 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 czytuje 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 MK₀ then goto 3 else goto 4
3. 0010; if TRUE then goto 4 else goto -
4. 0100; if L₂ 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 $\overline{\text{SYNC}}$ then goto 6 else goto 5

6. 0000; if FALSE then goto - else goto 1

w klasycznej wersji może być tylko 1 skok

Mikroprogram u 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 MK₀ then goto 3 else goto 4
3. 0010; if TRUE then goto 4 else goto -
4. 0100; if L₂ then goto 5 else goto 2
5. 1000; if 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 w ₂ w ₁ w ₀	operacje s ₃ s ₂ s ₁ s ₀	adres skoku 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 u celu zaoszczędzenia pamięci

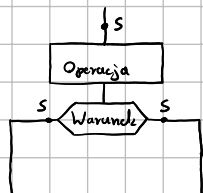
Przykłady

- multiplexer 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

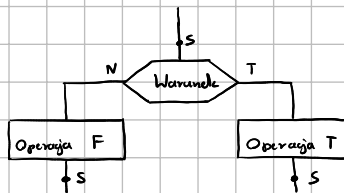
- stan przed klatką operacyjną
- pętla musi zawierać stan



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

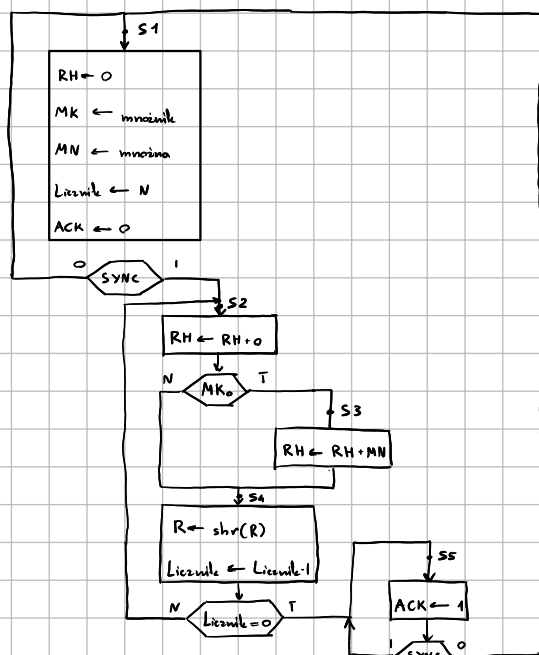
Automat Mealy'ego

- 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 2 NKB, 2 przepłotem realizacja



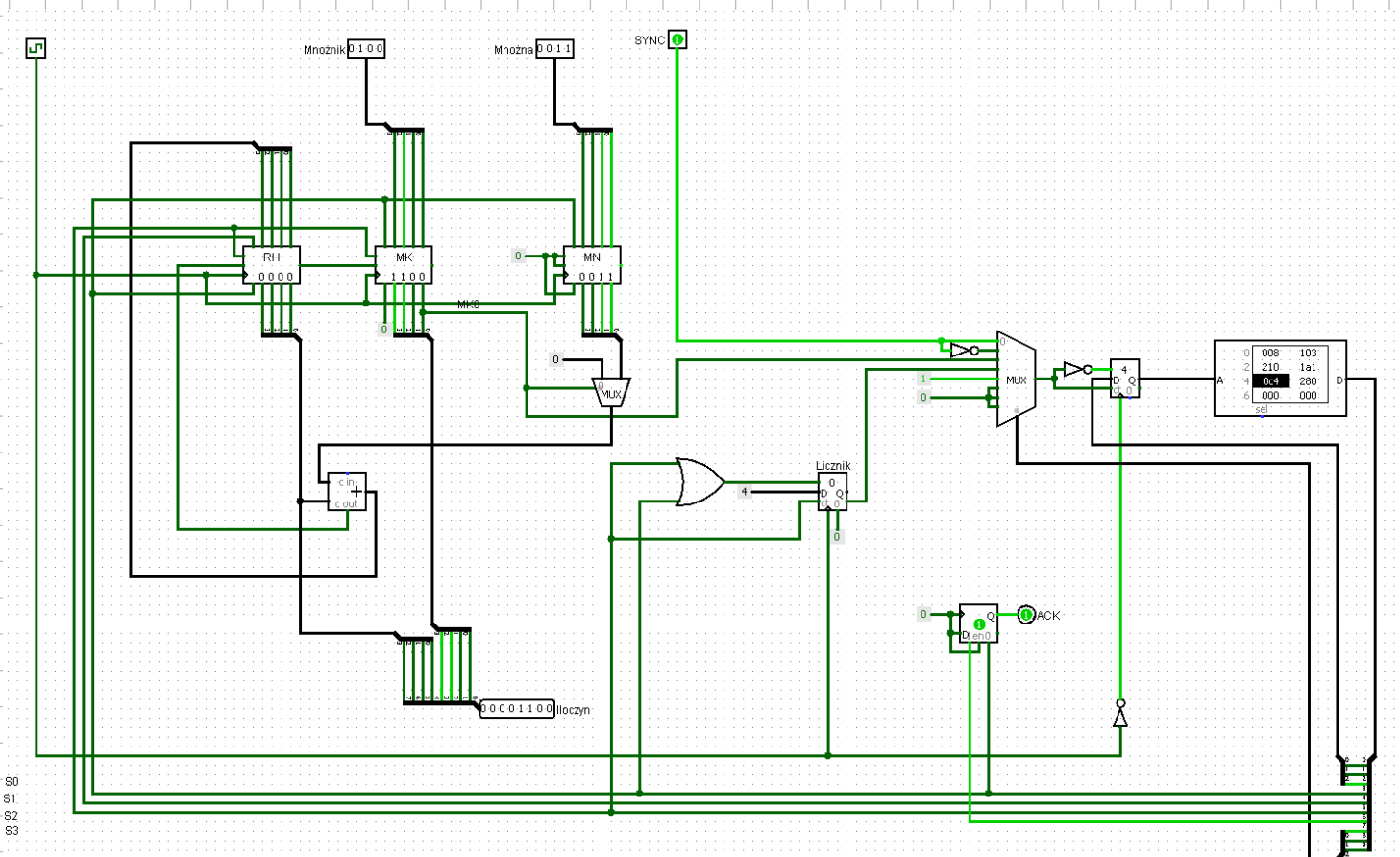
SYNC, L ₂ , MK ₀	000	001	010	011	100	101	110	111	S ₃	S ₂	S ₁	S ₀
1	1	1	1	1	2	2	2	2	0	0	0	1
2	4	3	4	3	4	3	4	3	0	0	0	0
3	4	4	4	4	4	4	4	4	0	0	1	0
4	2	2	5	5	2	2	5	5	0	1	0	0
5	1	1	1	1	5	5	5	5	1	0	0	0

- 0001; if SYNC then goto 2 else goto 1
- 0000; if MK₀ then goto 3 else goto 4
- 0010; if TRUE then goto 4 else goto -
- 0100; if L₂ then goto 5 else goto 2
- 1000; if SYNC then goto 6 else goto 5
- 0000; if FALSE then goto - else goto 1

adres	warunki	operacje	adres skłonu	HEX
instrukcji	U ₂ U ₁ U 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	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	0x230

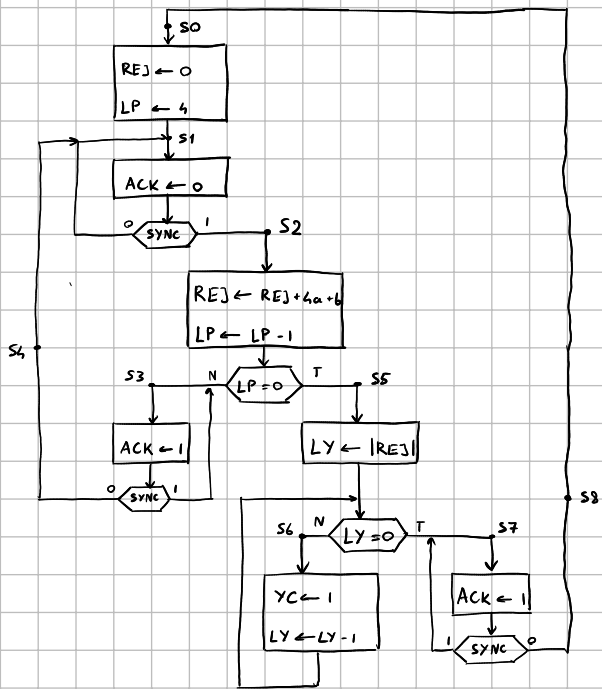
Warunek	Kod
SYNC	000
not SYNC	001
MK ₀	010
L ₂	011
TRUE	100
FALSE	101

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



Projekt uktaden

- dostaje na wyjściu szeregowo 4 pary liczb w U2 (a_i, b_i)
- oblicza wartości $Y = \sum_{i=1}^4 (4a_i \cdot b_i)$
- przypisuje na wyjściu 141 impulsów zegara
- wykonuje przeliczenia



0. REJ=0; LP=4;

1. $ACK \leftarrow 0$;

2. $REJ \leftarrow REJ + 4a + b$; $LP \leftarrow LP - 1$.

3. ACK $\leftarrow 1$;

4. NOP

5. $LY \leftarrow |RE|$

6. $YC \leftarrow 1$; $LY \leftarrow LY - 1$;

```

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
```

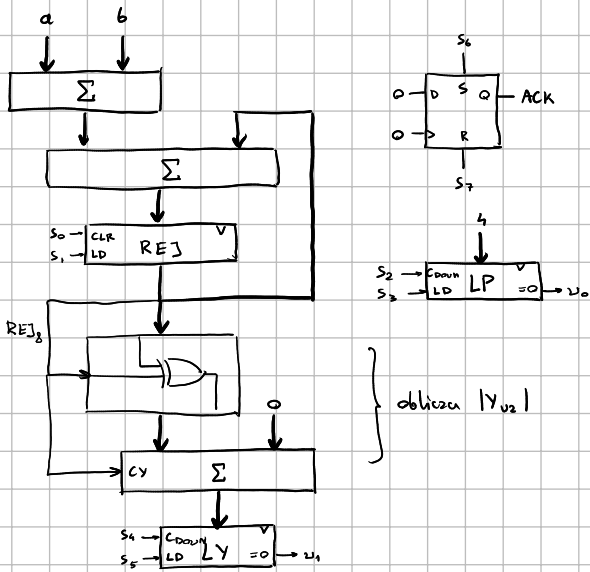
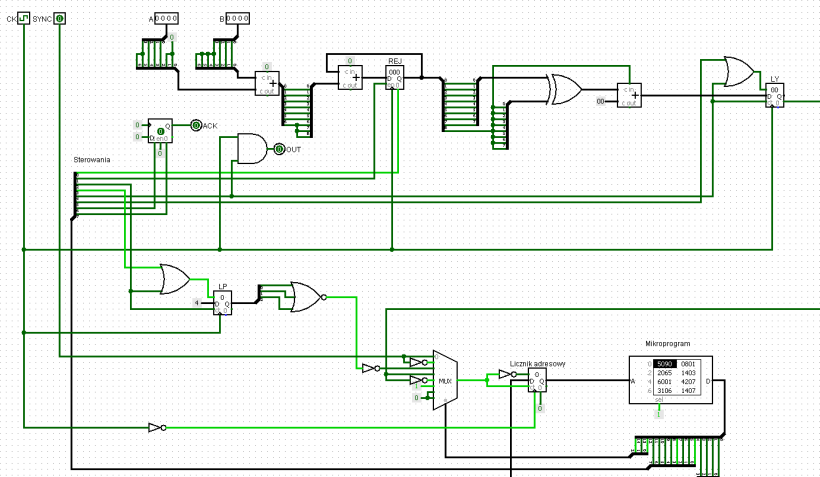
```
if LY=0 goto 7 else goto 6
```

```
if not src goto 3 else goto 7
```

```
if False - else goto 0
```

Warumule	Kod
SYNC	000
not SYNC	001
not Wo	010
W _i	011
not W _i	100
TRUE	101
FALSE	110

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

[illegible]

bity liczb U2 są rozszerzone
mnożenie przez przesunięcie bitowe