POLITECHNIKA ŚLĄSKA

WYDZIAŁ AUTOMATYKI, ELEKTRONIKI I INFORMATYKI

KIERUNEK INFORMATYKA

Projekt inżynierski

Aplikacja do edycji i wspomagania rozwiązywania zadań metodą TKŁ

Autor: Leszek Komorowski

Kierujący pracą: dr inż. Adam Opara

Gliwice, Styczeń 2022

2.3.3 Synteza układów sekwencyjnych

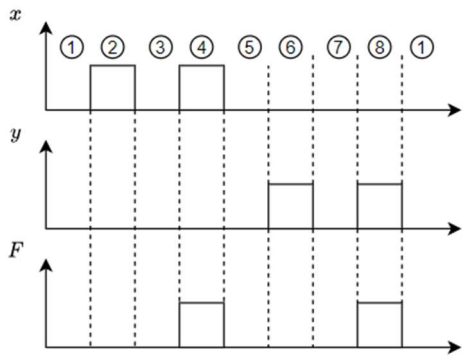
Nieco bardziej skomplikowanym procesem w stosunku do syntezy układów kombinacyjnych, jest synteza układów sekwencyjnych. Przede wszystkim przed przystąpieniem do tego etapu, konieczne jest sprawdzenie czy zadany układ jest układem sekwencyjnym. Polega ono na wykryciu czy istnieje przynajmniej jeden taki stan wejść, dla którego otrzymuje się różne stany wyjść. Gdy to sprawdzenie zostanie przeprowadzone, można rozpocząć syntezę. W przypadku układów asynchronicznych wyróżnia się dwie metody: metodę Huffmana (siatek programu) oraz metodę Siwińskiego (metoda tablic kolejności łączeń). Natomiast dla układów synchronicznych z reguły stosuje się tylko metodę Huffmana. W tym podrozdziale zostanie omówiona metoda siatek programu, natomiast metodzie TKŁ poświęcony zostanie osobny podrozdział.

Metoda Huffmana rozpoczyna się poprawnym opisem działania układu. Najczęściej jest on przedstawiony słownie, przebiegami czasowymi lub grafem przejść. Algorytm ten składa się z kilku etapów:

- Tworzenie pierwotnej tablicy programu
- Redukcja pierwotnej tablicy programu
- Kodowanie wierszy zredukowanej tablicy programu
- Określenie funkcji przejść elementów pamięci
- Określenie funkcji wyjść

Sposób realizacji niektórych z etapów zależy od wybranego rodzaju automatu – Moore'a lub Mealy'ego. Rezultatem procesu syntezy mogą być dwa schematy logiczne – jeden, gdy do realizacji bloku pamięciowego wykorzystane są przerzutniki oraz drugi, gdy wykorzystane są tylko bramki logiczne.

Pierwszy etap, którym jest tworzenie pierwotnej tablicy programu bazuje na opisie działania programu, np. przebiegu czasowym lub grafie przejść. Na rys. 16 przedstawiono przebieg gotowy pod kątem syntezy metodą Huffmana, gdzie poszczególne stany wewnętrzne układu zostały wyszczególnione liniami przerywanymi, a ich numeracja znajduje się w kółkach zawartych w danej wydzielonej części.



Rysunek 16. Przebieg czasowy uzupełniony o oznaczenia stanów

Pierwotna tablica składa się z wierszy będących kolejnymi stanami wewnętrznymi układu, kolumn z sygnałami wejściowymi zakodowanymi kodem Gray'a oraz dodatkową kolumną przechowującą stan wyjść przy danym stanie wewnętrznym. Stan stabilny w tablicy oznacza się symbolem stanu wewnątrz koła, a przejście do kolejnego stanu (stan niestabilny) oznacza się po prostu symbolem stanu. Brak przejścia dla danego stanu wejść oznacza się znakiem „-". Siatkę wypełnia się stanami stabilnymi wynikającymi z pracy układu, a następnie podaje się stany niestabilne, zaznaczając je w tym samym wierszu co poprzedni stan stabilny.

Poprawnie stworzona siatka programu dla przebiegów z rys. 16, zobrazowana jest na rys 17.

Stan \ xy	00	01	11	10	F
1	①	-	-	2	0
2	3	-	-	②	0
3	③	-	-	4	0
4	5	-	-	④	1
5	⑤	6	-	-	0
6	7	⑥	-	-	0
7	⑦	8	-	-	0
8	1	⑧	-	-	1

Rysunek 17. Pierwotna siatka programu

Dla tak przygotowanej tablicy programu, można przejść do etapu redukcji, który składa się z dwóch części: redukcji stanów równoważnych i pseudorównoważnych oraz redukcji stanów zgodnych.

Stanami równoważnymi i pseudorównoważnymi nazywa się te stany, dla których spełnione są następujące warunki:

- Oba stany mają zgodne wejścia
- Wyjścia są niesprzeczne (w przypadku, gdy stan wyjść jednego ze stanów jest nieokreślony, taki stan jest stanem pseudorównoważnym)
- Niesprzeczne przejścia – kolumny zawierają te same indeksy stanów niestabilnych, zawierają znak „-” lub w przypadku, gdy występuje równoważność warunkowa, czyli występują dwa indeksy stanów równoważnych

Wynikiem redukcji jest uzyskanie wiersza, w którym znajdują się nałożone wartości. Ponadto w innych wierszach, indeksy zredukowanych stanów zastępuje się stanem reprezentującym je po redukcji [6].

Z tablicy na rys. 17 można zaobserwować kilka kombinacji par stanów równoważnych. Taką parą są stany: 1-5, 3-7. Ponieważ spełniają one wszystkie warunki, można je zredukować. Rys. 18 przedstawia tablicę po wykonaniu redukcji.

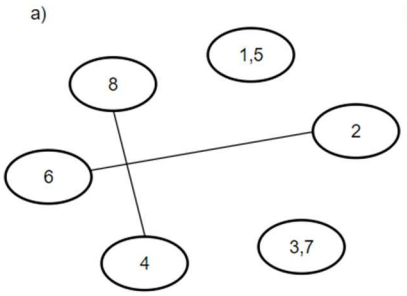
Stan \ *xy*	00	01	11	10	*F*
1,5	①	6	-	2	0
2	3	-	-	②	0
3,7	③	8	-	4	0
4	1	-	-	④	1
6	3	⑥	-	-	0
8	1	⑧	-	-	1

Rysunek 18. Zredukowana pierwotna siatka programu

Stanem zgodnym nazywa się taki stan, gdzie nie występują żadne sprzeczności indeksów. Ponadto, wybór stanów zgodnych zależy od wybranej realizacji. W przypadku, gdy wybraną realizacją jest automat Moore'a to oba stany muszą mieć zgodne wyjścia, dla automatu Mealy'ego warunek ten nie jest konieczny.

Przy redukcji stanów zgodnych, pożytecznym jest stworzenie wykresu redukcyjnego. Na wykresie tym linią ciągłą oznacza się stany, które można połączyć w realizacji dla automatu Moore'a. Linią przerywaną natomiast zaznacza się stany, których redukcja możliwa jest dla automatu Mealy'ego.

W przypadku rozważanego przykładu wykres redukcyjny dla obu realizacji będzie taki sam. Ze względu na zgodność wyjść oraz brak kolizji indeksów można połączyć stany 2-6 oraz 4-8. Wykres(a) oraz uzyskaną w ten sposób siatkę(b) przedstawia rys. 19.



b)

$Stan$	xy 00	01	11	10	F
1,5	①	6	-	2	0
2,6	3	⑥	-	②	0
3,7	③	8	-	4	0
4,8	1	⑧	-	④	1

Rysunek 19. a) wykres redukcyjny b) zredukowana w pełni siatka programu

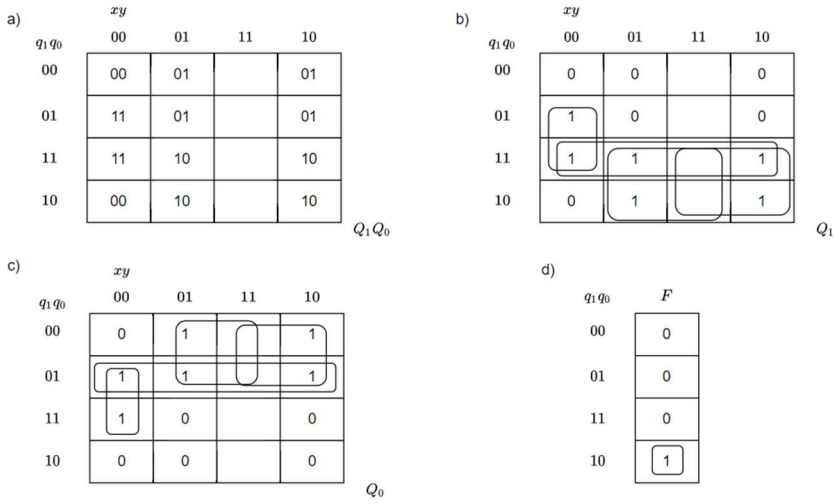
Tak zredukowaną siatkę, można zakodować. W zależności od liczby wierszy, wymagana jest różna liczba sygnałów stanu, którymi każdy wiersz będzie zakodowany. Jeżeli liczba wierszy zawiera się w przedziale $(2^{n-1}, 2^n >$ to wystarczy użyć dokładnie n sygnałów kodujących.

Kodowanie służy rozróżnieniu wierszy, ale w przypadku gdy zostanie ono wykonane błędnie, przyczynić się to może do powstania wyścigu w układzie. Temu zjawisku można zapobiec poprzez takie kodowanie, by zmiany stanów były sąsiednie logiczne. W przypadku rozważanego przykładu zjawisko wyścigu nie wystąpi, gdy zakoduje się wiersze odpowiednio stanami: 00, 01, 11, 10.

Na podstawie zakodowanej siatki programu, możliwe jest wyznaczenie funkcji przejść, czyli wyrażeń opisujących elementy pamięci oraz wyznaczenie funkcji wyjść, czyli wyrażeń będących wyjściem bloku kombinacyjnego. Siatki przejść wypełnia się podając dla stanów stabilnych kod danego wiersza, natomiast dla stanów niestabilnych kod wiersza, w którym znajduje się kolejny stan stabilny.

Określanie funkcji wyjść dla automatu Moore'a jest stosunkowo proste. Wymaga stworzenia tablicy zależności sygnałów wewnętrznych i wartości wyjścia. Gdyby automat realizowany był jako automat Mealy'ego, konieczne jest także uwzględnienie wpływu sygnałów wejściowych, dlatego w przypadku zredukowanej siatki dla układu Mealy'ego w każdym wierszu podaje się także wartości wyjść w danych stanach.

Ponieważ zadany przykład nie posiadał wierszy redukowanych dla struktury Mealy'ego, stworzone zostały siatki przejść i wyjść dla układu Moore'a. Rys. 20 przedstawia zakodowaną siatkę przejść dla obu sygnałów stanu (a), siatkę przejść dla stanu Q_1(b), siatkę przejść dla stanu Q_2(c) oraz tablicę zależności dla wyjścia F(d).



Rysunek 20.
a) siatka przejść dla sygnałów Q1,Q0 b) siatka przejść sygnału Q1
c) siatka przejść sygnału Q0 d) tablica zależności wyjścia F

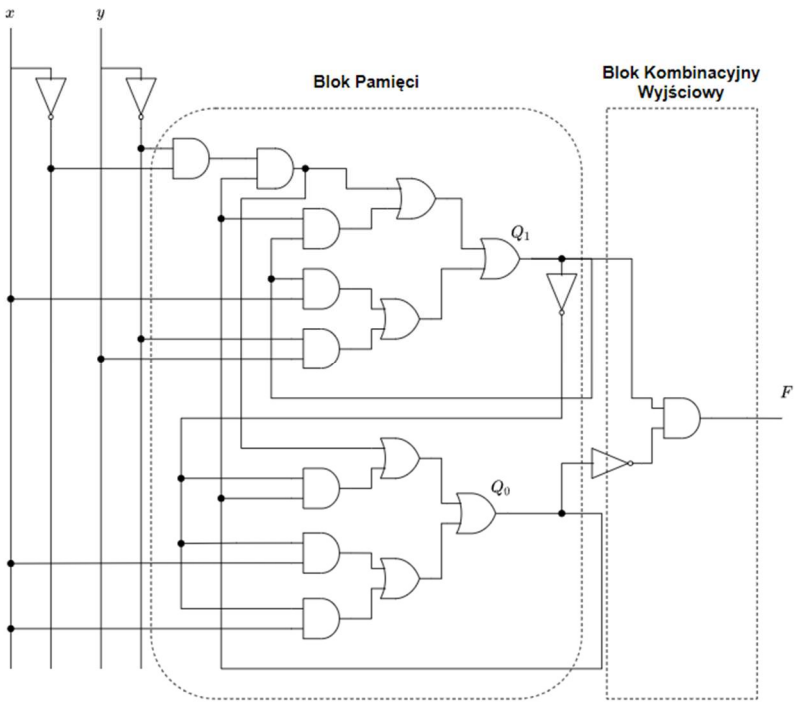
Z siatek z rys. 20 można wywnioskować wyrażenia strukturalne dla sygnałów stanu wewnętrznego oraz funkcji F. Na siatkach b) oraz c) zaznaczono grupy jedynek, na podstawie których można uzyskać wyrażenia sygnałów wewnętrznych. Jak można słusznie zauważyć, dla obu sygnałów tak utworzone wyrażenia są wolne od hazardu, gdyż wszystkie składniki jedynki są częścią co najmniej dwóch wspólnych grup. Zadany w przykładzie układ można zatem zrealizować przy pomocy wyrażeń:

$$F = q_1\overline{q_0} \quad (24)$$

$$Q_1 = q_0\bar{x}\bar{y} + q_1q_0 + q_1x + q_1y \quad (25)$$

$$Q_0 = q_0\bar{x}\bar{y} + \overline{q_1}q_0 + \overline{q_1}x + \overline{q_1}y \quad (26)$$

Realizacja takiego układu, przy użyciu podstawowego zestawu funkcjonalnie pełnego zaprezentowana została na rys. 21.



Rysunek 21. Schemat uzyskanego układu