# Wielopoziomowa synteza układów logicznych

Dwupoziomowa synteza sprowadza się do realizacji, w których pierwszy poziom tworzą bramki AND, a drugi bramki OR. Cała struktura układu jest opisana formułą typu: suma iloczynów. Są to więc struktury naturalnie przystosowane do realizacji na bramkach AND, OR, NOT, NAND, a w konsekwencji również wygodne do implementacji w strukturach PLA i PAL powszechnie stosowanych w technice układów PLD.

W realizacjach *Semi Custom, z*e względu na powierzchnie, jakie zajmują realizowane układy kombinacyjne w całej strukturze modułu scalonego, oszczędniejsze są tzw. **Struktury wielopoziomowe**. Są one uzyskiwane metodami transformacji sumo-iloczynowych wyrażeń boolowskich na wielopoziomowe wyrażenia silnie sfaktoryzowane (rozłożone na czynniki). Dzięki takim implementacjom wielopoziomowych sieci bramek logicznych, zwiększa się swoboda projektowania, którą można wykorzystać zarówno podczas optymalizacji powierzchni oraz opóźnienia, jak i w celu spełnienia szczególnych wymagań, np. różnych ograniczeń czasowych w poszczególnych częściach układu.

Z tego powodu wielopoziomowe sieci logiczne są chętniej stosowane niż implementacje dwupoziomowe, takie jak np. układy logiki programowalnej. Procedury syntezy wielopoziomowej, stanowią dzisiaj podstawę wielu komputerowych narzędzi syntezy i optymalizacji układów cyfrowych, pomimo tego, iż wiedza na temat metod projektowania wielopoziomowych układów logicznych nie osiągnęła jeszcze poziomu porównywalnego z wiedzą o metodach optymalizacji logiki dwupoziomowej. Samo modelowanie i optymalizacja takich sieci logicznych są dużo trudniejsze.

## Sieć logiczna

Strukturę wielopoziomowego układu logicznego na poziomie połączeń między bramkami można opisać w postaci **sieci logicznej. S**kłada się z modułów, reprezentujących porty wejścia/wyjścia i bramki logiczne, oraz połączeń między nimi.

Można przyjąć, że sieci logiczne są sieciami *funkcji lokalnych*, gdzie sieć jest opisem struktury a funkcje lokalne opisują lokalne zachowanie układu.

Można ją również przedstawić w postaci grafu. Każdemu wierzchołkowi odpowiada funkcja lokalna.

Optymalizacja sieci

Główne cele optymalizacji kombinacyjnych układów logicznych to zmniejszenie **powierzchni i opóźnienia.** Porównując implementacje układów logicznych dwu- i wielopoziomowych, można zauważyć różnice w stosunkach powierzchni do opóźnienia.

W przypadku dwupoziomowej implementacji układu reprezentowanego przez sumę iloczynów, powierzchnia oraz opóźnienie są proporcjonalne do rozmiaru pokrycia, a więc poszukiwanie minimalnego pokrycia sprowadza się do minimalizacji zarówno powierzchni jak i opóźnienia.

W przypadku wielopoziomowych układów w ogólnym przypadku implementacje o minimalnej powierzchni nie mają najmniejszego opóźnienia i vice versa (rys). Tak więc poszukiwanie kompromisu pomiędzy powierzchnią a opóźnieniem jest szczególnie ważnym zadaniem.

Modelowanie powierzchni oraz opóźnienia

OPÓŹNIENIE

Optymalizacja zależności czasowych polega na minimalizacji opóźnień najwolniejszej ścieżki, nazywanej **ścieżką krytyczną.** Z modelowaniem są związane dwa zagadnienia: obliczanie opóźnień propagacji w każdym wierzchołku oraz obliczenie opóźnienia ścieżki danych.

W przypadku niektórych sieci, opóźnienia propagacji można łatwo oszacować gdyż, zależność obciążenia wyjść bramek od opóźnienia jest zawarta w opisie biblioteki. W przeciwnym razie należy oszacować opóźnienia bramek wirtualnych, będących implementacjami funkcji lokalnych. Ogólnie przyjmuje się, że na każdym poziomie występuje opóźnienie jednostkowe

Najprościej opóźnienia ścieżki danych można obliczyć przybliżając sumę opóźnień propagacji występujących we wszystkich wierzchołkach ścieżki.

Przykład:
Czasy gotowości danych pierwotnych wejść wynoszą ta = 0 i tb = 10
Opóźnienia propagacji wierzchołków wewnętrznych d = ….

POWIERZCHNIA

Powierzchnia wielopoziomowej sieci logicznej to powierzchnia zajmowana przez bramki oraz połączenia. Jeżeli bramki są przyporządkowane komórkom z biblioteki, wtedy znana jest powierzchnia każdej z nich. W innym przypadku trzeba szacować powierzchnię *bramek wirtualnych*, będących implementacjami funkcji logicznych.

Często powierzchnię szacuje się na podstawie liczby literałów L(f) w rozłożonym na czynniki wyrażeniu reprezentującym funkcję.

*Przykład:*

$$L((a + b)ca') = 4 \qquad L((a + b + cd)(a' + b')) = 6$$

Można twierdzić, że w nowoczesnych technologiach VLSI liczba literałów jest proporcjonalna do powierzchni układu (powierzchni płatka krzemu). Niestety nie jest to kryterium najlepsze do pełnej oceny implementacji.

Z tego powodu synteza wielopoziomowa posługuje się bogatym zbiorem operacji umożliwiających iteracyjną transformację sieci logicznej, pierwotnie zadanej w postaci minimalnego boolowskiego wyrażenia algebraicznego. Te operacje to: **dekompozycja, ekstrakcja, faktoryzacja i substytucja.**

Do najważniejszych należą operacje faktoryzacji i dekompozycji wyrażeń boolowskich. Operacje te transformują pojedyncze wyrażenia boolowskie na zbiór kilku nowych, niezależnych wyrażeń

## DEKOMPOZYCJA

Dekompozycja transformuje pojedyncze wyrażenia boolowskie na zbiór kilku nowych, niezależnych wyrażeń.

*Rozważmy funkcję:*

$$f = abc + abd + a'c'd' + b'c'd'$$

*która po dekompozycji przyjmie następującą postać:*

$$f = gh + g'h'$$

*gdzie:*

$$g = ab$$
$$h = c + d$$

Liczba literałów w wyrażeniu (minimalnym) dla funkcji f wynosi 12. Jego bezpośrednia realizacja wymaga 9 bramek (uwzględniając inwertory).

Liczba literałów w nowych wyrażeniach g, h oraz f wynosi 8 i w rezultacie uzyskujemy prostszą implementację.

## EKSTRAKCJA (WYDZIELANIE)

Ekstrakcja jest odpowiednikiem dekompozycji dla zespołów funkcji. Jej zadaniem jest identyfikacja wspólnych wyrażeń (podfunkcji) w zbiorach funkcji, w celu ich jednokrotnej realizacji, co zwykle prowadzi do uproszczenia struktury.

*Przykład:*
*Dokonujemy ekstrakcji w sieci logicznej, opisanej wyrażeniami:*

$$f = (a + b)cd + e$$
$$g = (a + b)e'$$
$$h = cde$$

*Operacja ekstrakcji powinna rozpoznać podfunkcje*: $p = a + b$ *oraz* $r = cd$
*Uwzględniając wyznaczone podfunkcje, pierwotną sieć logiczną możemy przedstawić w postaci:*

$$f = pr + e$$
$$g = pe'$$
$$h = re$$

Jak widać, pierwotny zbiór funkcji zawierający 11 literałów może być zrealizowany na 8 bramkach. Zbiór funkcji f, g, h po ekstrakcji w dalszym ciągu zawiera 11 literałów, ale jego realizacja wymaga już tylko 7 bramek.

## FAKTORYZACJA (UPRASZCZANIE)

Faktoryzację traktuje się jako oddzielną operację, często wykorzystywaną w innych (np. w ekstrakcji). Przekształca dwupoziomowe wyrażenie boolowskie w wielopoziomowe, nie uwzględniające podfunkcji pośrednich.

*Rozważmy funkcję f, daną w następującej sumo-iloczynowej postaci:*

$$f = ac + ad + bc + bd + e$$

*Pierwotne 9 literałów tego wyrażenia operacja faktoryzacji redukuje do 5:*

$$f = (a + b)(c + d) + e$$

**SUBSTYTUCJA**

Czyli PODSTAWIENIE funkcji *g* do funkcji *f* polega na przedstawieniu *f* z uwzględnieniem *g*.

*Przykład:*

*jeśli :* f = a + bc oraz g = a + b

*to* f *może być przedstawione wyrażeniem* g *jak oddzielny literał:*

f = a + bc = g(a + c)

Istnieją jeszcze inne metody syntezy, lecz te omówione do tej pory należą do najczęściej stosowanych w optymalizacji wielopoziomowej. Przekształcenia logiczne wpływają zarówno na powierzchnię jak i szybkość działania sieci, ponieważ modyfikują liczbę literałów, funkcje lokalne oraz zależności między nimi.
Podobnie jak w przypadku optymalizacji dwupoziomowej, na sieci wykonuje się iteracyjne przekształcenia logiczne. Powierzchnię lub opóźnienie sieci uznaje się za optymalne, jeśli żadna z dostępnych operacji nie poprawia rozwiązania (odpowiedniej miary jakości układu).

ALGORYTMY OPTYMALIZACJI WIELOPOZIOMOWEJ

Wszystkie metody optymalizacji wielopoziomowej polegają na stopniowym doskonaleniu poprzez wykonywanie przekształceń. Istnieje wiele odmian tych metod, różniących się sposobami wybierania i wykonywania operacji. Podstawowa klasyfikacja polega na podziale na **metody algorytmiczne oraz metody sterowane regułami.**

Metody algorytmiczne wymagają zdefiniowania algorytmu każdego typu przekształcenia. Algorytm ten wykrywa czy i gdzie można wykonać przekształcenie, jak i kończy swoje działanie, gdy żadne przekształcenie danego typu nie poprawia rozwiązania. Zaleta metod algorytmicznych polega na systematycznym wykonywaniu tych operacji i w rezultacie można oczekiwać, że stosując jeden lub więcej algorytmów, otrzyma się sieć o pewnych właściwościach. Każdy algorytm można traktować jak operator wykonujący zbiór przekształceń.

Optymalizacja oparta na regułach polega na wykonywaniu przekształceń różnych typów zgodnie z regułami naśladującymi postępowanie człowieka projektującego układ. Baza danych zawiera zbiór par reprezentujących najczęściej spotykane fragmenty sieci. Systemy oparte na regułach wykrywają sytuacje, w których podsieć odpowiadająca pierwszemu elementowi pary można zastąpić równoważną jej podsiecią optymalną.

## Model algebraiczny

Polega na reprezentowaniu lokalnych funkcji logicznych przy użyciu wyrażeń algebraicznych, czyli wielo-liniowych wielomianów zmiennych sieciowych o jednostkowych współczynnikach.
*Przekształcenia algebraiczne* polegają na przekształceniu wyrażeń zgodnie z regułami algebry wielomianów, bez wykorzystania szczególnych właściwości algebry Boole'a. W algebrze uwzględnia się tylko prawa rozłączności tzn.

$$a * (b + c) = ab + ac$$
$$a + (b * c) \text{ nie= } (a + b)(a + c)$$

nie definiuje się dopełnień. W rezultacie nie można korzystać z takich właściwości jak m.in: prawa de Morgana lub tożsamości a + a' = 1, aa' = 0

*pojęcia*

kostka – zbiór literałów takich, że:
x e c ➔ x' nie e c

np. a + ac nie jest wyrażeniem algebraicznym
a + a'b jest wyrażeniem algebraicznym

W optymalizacji wielopoziomowej ważną rolę odgrywa dzielenie wyrażeń algebraicznych.

## Definicja dzielenia algebraicznego:

Niech $\{f_{dzielna}, f_{dzielnik}, f_{iloraz}, f_{reszta}\}$ będą wyrażeniami algebraicznymi. O wyrażeniu, $f_{dzielnik}$ mówi się, że jest dzielnikiem algebraicznym wyrażenia $f_{dzielna}$ jeśli $f_{dzielna} = f_{dzielnik} * f_{iloraz} + f_{reszta}$ i $f_{dzielnik} * f_{iloraz} \text{ nie= } 0$.

Przykłady:

Niech dzielna = ac + ad + bc + bd + e i dzielnik = a + b wtedy w wyniku dzielenia otrzymuje się iloraz c + d oraz resztę „e" ponieważ, dzielna = (a + b)(c + d) + e. nośnikami wyrażeń są zbiory rozłączne sup(dzielnik) = {a,b} i sup(iloraz) = {c,d} .

Niech fi = a + bc oraz fj = a + b. Wyrażenie fj nie jest uważane za algebraiczny dzielnik wyrażenia fi mimo że fi = fj * fk dla fk = a + c , ponieważ sup(fj) = (a,b) i fk={a,c) mają część wspólną.
Rdzenie

Iloraz z dzielenia algebraicznego wyrażenia f przez kostkę c (f|c) jest nazywany **rdzeniem** k funkcji f, jeśli iloraz ten zawiera co najmniej dwie kostki nie mające żadnego wspólnego literału. Kostka reprezentujące ten podzielnik jest nazywana **ko-rdzeniem**.
Jeśli rdzeń nie ma innych rdzeni oprócz siebie, to taki rdzeń nazywany jest rdzeniem stopnia 0. Rdzeń jest nazywany stopnia n, jeśli ma co najmniej jeden rdzeń stopnia (n-1), natomiast nie ma żadnego (za wyjątkiem siebie) rdzenia stopnia n lub wyższego.

Przykład:

f = a'c'd + a'bc + abd + abc' + bcd

iloraz f przez kostkę a: f|a = bd + bc'
ale skoro literał b jest wspólny dla obu kostek to f|a nie jest rdzeniem.
Natomiast:
f|c' = a'd + ab

w tym przypadku kostki tworzą rdzeń, gdyż nie ma wspólnego literału. Ko-rdzeniem dla f|c' jest c'

Ko-rdzeń danego rdzenia nie jest wyznaczany jednoznacznie. Np. funkcja

f= ad + bd + ac + bc

ma rdzeń a + b, przy czym jest on uzyskiwany za pośrednictwem ko-rdzeni c oraz d.

Jednym z ważniejszych zastosowań rdzeni jest możliwość wyznaczania wspólnych części w różnych wyrażeniach boolowskich. Sytuacja taka ma miejsce wtedy, gdy rdzenie różnych wyrażeń mają przecięcie składające się z więcej niż jednej kostki.

*Przykład:*

Mamy dwa wyrażenia boolowskie:
f1 = abc + ac'g + b'df + cde
f2 = a'bd' + bce' + b'de + c'e'g

Obliczając przecięcia rdzeni z f1 i f2 można wyznaczyć wspólne składniki tych rdzeni. Na przykład rdzeń *ab + de* funkcji *f1* przecina się z *de + g'* funkcji *f2* a częścią wspólną jest kostka *de*.
f1 i f2 mają wspólny rdzeń *bc + c'g* odpowiadający różnym ko-rdzeniom *a* oraz *e'*.
Obliczanie przecięcia rdzeni (czyli wyznaczanie wspólnych podwyrażeń), których rezultatem jest zmniejszenie liczby literałów, należy do tzw. Problemu pokrycia prostokątnego.

Prostokąt {R(1,6), C(1,2)} reprezentuje przecięcie rdzeni odpowiadających wierszom 1 oraz 6. Przecięcie obu tych rdzeni generuje wspólne wyrażenie *bc + c'g*. Prostokąt rozciągnięty na więcej niejedną kolumnę identyfikuje przecięcie rdzeni złożone z więcej niż jednej kostki.
Zbiór prostokątów tworzy pokrycie prostokątne macierzy, jeśli każda liczba całkowita jest pokryta przez co najmniej jeden prostokąt danego zbioru.
Pokrycie macierzy ko-rdzeni jest następujące:
{R(1,6), C(1,2)} , {R(3), C(5,6)}, {R(4), C(7)}, {R(5), C(4, 9)}

A więc funkcje realizujące to pokrycie są opisane wyrażeniami:

f1 = aX + dY
f2 = e'X + b'Z + a'bd'
gdzie:
X = bc + c'g
Y = b'f + ce
Z = de + g'

Pierwotne wyrażenia funkcji miały L(f1,f2) = 26
Po operacji L(f1, f2) = 22

Opracował:
*Michał Wikar*
*III rok EiT*
*AGH, 2011*