Robert CZERWIŃSKI, Dariusz KANIA
POLITECHNIKA ŚLĄSKA, INSTYTUT ELEKTRONIKI

Synteza logiczna układów sekwencyjnych realizowanych w strukturach CPLD opisanych za pomocą języka VHDL

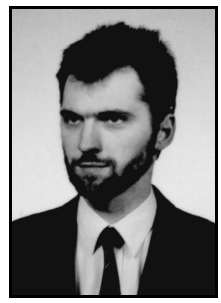
Dr inż. Robert CZERWIŃSKI

Studia na Wydziale Automatyki, Elektroniki i Informatyki Politechniki Śląskiej ukończył w 2001 roku. Na tymże wydziale obronił pracę doktorską w 2006 roku. Obecnie jest adiunktem w Instytucie Elektroniki Politechniki Śląskiej. Jego zainteresowania naukowe skupiają się wokół projektowania układów cyfrowych z wykorzystaniem układów logiki programowalnej oraz systemów mikroprocesorowych.

e-mail: robert.czerwinski@polsl.pl

Dr hab. inż. Dariusz KANIA

Ukończył studia na Wydziale Automatyki, Elektroniki i Informatyki Politechniki Śląskiej. Pracę doktorską obronił w 1995, habilitacyjną w 2004r. Jest adiunktem w Instytucie Elektroniki Politechniki Śląskiej. Jego zainteresowania naukowe koncentrują się wokół programowalnych układów i systemów cyfrowych.



e-mail: Dariusz.Kania@polsl.pl

Streszczenie

W artykule przedstawiono problem kodowania stanów wewnętrznych automatów sekwencyjnych ukierunkowany na realizację układu w strukturze matrycowej typu PAL. Opracowano sposób uwzględniania elementów dwupoziomowej minimalizacji oraz elementów dopasowania już na etapie kodowania stanów wewnętrznych. Sporo miejsca poświęcono problemowi opisu automatu w języku opisu sprzętu VHDL pod kątem efektywnego przeprowadzenia syntezy logicznej w systemie Quartus II. Skuteczność metod kodowania i opracowanego opisu potwierdzają uzyskane wyniki eksperymentów.

Słowa kluczowe: automaty sekwencyjne, kodowanie stanów, CPLD, VHDL.

Logic synthesis of sequential automata implemented in CPLDs, and described in VHDL

Abstract

The paper concerns the problem of state assignment for finite state machines (FSM), targeting at PAL-based CPLDs implementations. The main feature of a PAL-based cell is a limited number of product terms (k AND-gates) that are connected to a single sum (OR-gate). Methods, that do not take into account this limited number of product terms in the process of state assignment, usually lead to multi-cell and multi-level structures. To make allowance for number of product terms the elements of two-level minimization and elements of technology mapping must be taken into account in the process of state assignment. This is possible thanks to Primary and Secondary Merging Conditions and Implicants Distribution Table. The problem of the sequential automata VHDL design is also considered. The VHDL design description of the FSM for Quartus II is proposed. Experimental results consider the efficiency of the proposed methods.

Keywords: FSM, state assignment, VHDL, CPLD.

1. Wprowadzenie

Podstawowym etapem syntezy logicznej automatów sekwencyjnych jest proces kodowania stanów wewnętrznych. Istotna trudność wynika z faktu, iż dobór słów kodowych ma wpływ jednocześnie na część wejściową i wyjściową implikantów wielowyjściowych. Proces kodowania ma niebagatelny wpływ na liczbę elementów pamięci i złożoność bloków kombinacyjnych realizujących funkcję wzbudzeń przerzutników i bloku realizującego funkcję wyjść. Większość narzędzi wspomagających projektowanie układów sekwencyjnych opartych jest na klasycznej metodzie, która po przeprowadzeniu kodowania stanów wewnętrznych zawiera etap dwupoziomowej minimalizacji jednowyjściowych funkcji. Następnie przeprowadzane jest dopasowanie technologiczne struktury układu sekwencyjnego do struktury występujących w układach programowalnych bloków logicznych.

Rdzeń większości współczesnych układów CPLD stanowi struktura AND-OR typu PAL o liczbie iloczynów nie przekraczającej 5. Efektywne wykorzystanie iloczynów komórki typu PAL wymaga uwzględnienia efektów dwupoziomowej minimalizacji oraz elementów dopasowania technologicznego już na etapie kodowania stanów wewnętrznych automatów sekwencyjnych. Oprócz sposobów kodowania stanów praca porusza niezwykle istotne zagadnienie projektowania układów cyfrowych, mianowicie sformułowanie odpowiedniego opisu wyników kodowania w języku VHDL.

Prezentowane w niniejszym artykule podstawy teoretyczne oraz algorytmy kodowania stanów szerzej przedstawiono w pracach [1-3]. Ujęto tam również przeprowadzone eksperymenty wraz z analizą porównawczą wyników z wynikami uzyskanymi za pomocą systemów JEDI i NOVA [4, 5].

2. Metody kodowania

Niech wagą stanu η^S nazywa się liczbę przejść do danego stanu S w tablicy przejść-wyjść. Niech rzędem kodu μ nazywa się liczbę jedynek słowa kodowego. Niech C_μ oznacza kostkę rzędu μ. Niech implikantem symbolicznym będzie (X, S, S^+, Y), gdzie X jest wektorem wejściowym automatu, S jest stanem aktualnym automatu, S^+ jest stanem następnym automatu, zaś Y jest wektorem wyjściowym. Niech K oznacza minimalną liczbę bitów słowa kodowego niezbędną do zakodowania stanów automatu, zaś kodowanie stanu S niech będzie oznaczone jako $\varepsilon(S)$.

Ponieważ przejście do stanu następnego automatu jest reprezentowane przez tyle implikantów wielowyjściowych ile wynosi waga tego stanu, stąd **stanom o wyższych wagach należy przyporządkować słowa kodowe o mniejszej liczbie jedynek (minimalnego rzędu)**. Stan o najwyższej wadze powinien mieć przyporządkowane słowo kodowe zerowego rzędu. W wyniku takiego przyporządkowania żadna ze składowych funkcji przejść automatu nie będzie miała implikantów odpowiadających przejściu do tego stanu.

Zdefiniowano pierwotne i wtórne warunki sklejenia oraz tablicę rozkładu implikantów umożliwiającą wprowadzenie w proces kodowania elementów dopasowania projektowanego układu do wykorzystywanej struktury programowalnej.

Pierwotnymi warunkami sklejenia (PWS) nazywa się relację pomiędzy dwoma implikantami symbolicznymi (X, S_p, S_i, Y_a) i (X, S_r, S_i, Y_b) taką, że cześć wejściową tworzą dwa stany S_p i S_r oraz taki sam wektor wyjściowy X, a część wyjściową tworzy stan S_i. Stanom S_p i S_r przypisuje się wówczas słowa kodowe, które różnią się na jednej pozycji. Analiza pierwotnych warunków sklejenia jest możliwa przed procesem kodowania.

Klasyczna metoda realizacji funkcji $f: B^n \to B^m$ w strukturach CPLD typu PAL związana jest z realizacją składowych funk-

cji $f_i : B^n \to B$ dla $i = m-1,\ldots,0$, gdzie: n jest liczbą wejść układu sekwencyjnego, a m liczbą wyjść układu sekwencyjnego. Kostka odpowiadająca części wejściowej jest implikantem funkcji $f_i : B^n \to B$ $(i = m-1,\ldots,0)$ dla i-tego składnika części wyjściowej równego 1. Druga możliwość pokrycia pary implikantów jednym implikantem wynika z klasycznej metody realizacji funkcji w strukturach CPLD. Jeżeli realizowane jest przejście z dwóch różnych stanów S_i i S_j dla wektora X, którym przypisano kody różniące się na jednej pozycji, to możliwe jest pokrycie tych implikantów jednym. Podobnie, jeżeli realizowane jest przejście ze stanu S_i dla wektorów wejściowych X_u i X_w, które różnią się na jednej pozycji oraz implikanty są implikantami tej samej funkcji, to również możliwe jest pokrycie tych implikantów jednym.

Wtórny Warunek Sklejenia (WWS) powstaje, gdy istnieją dwa przejścia ze stanów S_p i S_r do stanów S_a i S_b dla wektora wejściowego X, przy czym implikanty należą do tej samej funkcji przejścia δ_i (kody przyporządkowane stanom S_a i S_b mają wspólną jedynkę).

W celu dopasowania liczby implikantów do liczby iloczynów komórki typu PAL, konieczne jest wyznaczanie liczby implikatów składowej funkcji w trakcie kodowania stanów. **Tablicą Rozkładu Implikantów** (TRI) nazywa się tablicę, w której kolejne wiersze odpowiadają wagom zakodowanych stanów. Wagi η^S zapisywane są odpowiednio w kolumnach o indeksach odpowiadających bitom, dla których w kodzie stanu występuje 1. Spełnienie pierwotnych lub wtórnych warunków sklejenia zaznaczane jest w odpowiedniej kolumnie tablicy przez „-1".

Na początku procesu kodowania ukierunkowanego na **m**inimalizację **l**iczby **k**omórek (mlk) wyznaczana jest liczba bitów K słowa kodowego, wybierany jest poziom aktywności każdego wyjścia bloku wyjść [6] oraz wyznaczane są PWS. Pierwszym kodowanym stanem jest stan o najwyższej wadze (i najmniejszej liczbie PWS), któremu przyporządkowuje się zerowe słowo kodowe. Następnie, rząd jest inkrementowany i znów wybierany jest stan o największej wadze. Jeżeli stanów jest więcej niż jeden, to wybiera się stan spełniający kolejne warunki: może spełnić najwięcej PWS, może spełnić najwięcej WWS. Po wybraniu stanu, któremu zostanie przyporządkowane słowo kodowe następuje wybór tego słowa. Jeżeli wszystkie słowa kodowe rzędu μ zostały wyczerpane, to w kolejnych krokach wykorzystywane są słowa kodowe $\mu+1$ rzędu. Jeżeli słów kodowych rzędu μ jest więcej niż jedno, to należy wybrać słowo kodowe, dla którego wartości σ^δ, σ^λ i η^δ po uwzględnieniu wszystkich warunków sklejenia są minimalne. Wartości σ^δ, σ^λ i η^δ oznaczają odpowiednio liczbę komórek bloku przejść, liczbę komórek bloku wyjść i liczbę implikantów funkcji przejść.

```
KODOWANIE_mlk()
{
    Wyznacz_K();
    Wyb_akt_wy();     /* wybór poziomu aktywności wyjść */
    Wyznacz_PWS();
    μ = 0;     /* rząd zerowy */
    Szuk_S_max_η^S_min_PWS();     /* największa waga, najmniej PWS */
    ϵ(S) = ℂ_μ;     /* kodowanie */
    μ = 1;     /* ustaw rząd na 1 */
    while(nie wszystkie_stany)     /* dopóki nie są zakodowane wszystkie stany */
    {
        Szuk_S_max_η^S_max_PWS_max_WWS();
        if(licz_ℂ_μ == 0) μ++;     /* kostka rzędu μ nie istnieje to zwiększ rząd */
        Szuk_ℂ_μ_min_σ^δ_σ^λ_η^δ();     /* szuk. ℂ_μ o min. wart. σ^δ, σ^λ i η^δ */
        ϵ(S) = ℂ_μ;     /* przypisanie wyszukanej kostki */
        Odsw_TRI();     /* odświeżenie TRI, dodanie nowych wierszy */
        Wyznacz_nowe_WWS();
        Usun_PWS_WWS();     /* usuń spełn. i niemożliwe do spełn. war. */
    }
}
```

Rys. 1. Algorytm kodowania ukierunkowany na minimalizację liczby komórek typu PAL
Fig. 1. State assignment agorithm for the number of the PAL-cells minimization

Po zakodowaniu stanu następuje dopisanie nowych wierszy do TRI, wyznaczenie nowych WWS oraz usunięcie warunków, które już nie mogą zostać spełnione. Dotyczy to zarówno PWS, jak i WWS. Proces kodowania powtarzany jest do momentu zakodowania wszystkich stanów. Algorytm kodowania *mlk* został przedstawiony na rysunku 1.

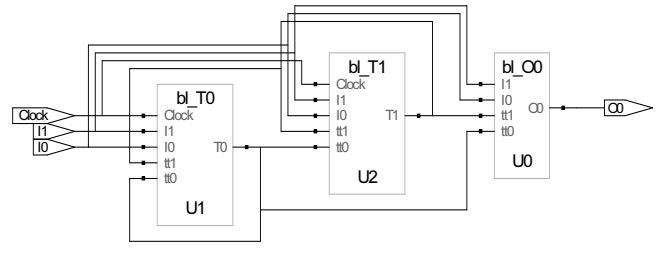
Algorytm kodowania ukierunkowany na **m**inimalizację **l**iczby **w**arstw logicznych (mlw) ma wbudowany mechanizm kontroli liczby warstw. Przed procesem kodowania wyznaczana jest minimalna liczba warstw, dla której zawsze możliwa jest realizacja automatu. W trakcie procesu kodowania sprawdzana jest aktualna liczba warstw na podstawie TRI. Jeżeli wyznaczona przed kodowaniem liczba warstw została przekroczona, to zwiększana jest liczba bitów słowa kodowego i proces kodowania prowadzony jest dalej. Precyzyjny opis algorytmu *mlw* można znaleźć w pracach [1-3].

3. Metody opisu automatów

Najwygodniejszą i najpopularniejszą formą opisu automatu w języku VHDL jest opis behawioralny [7]. Istotną cechą tego typu opisu jest uniezależnienie go od szczegółów ostatecznej realizacji oraz możliwość zastosowania typu wyliczeniowego, czyli przedstawienie stanów w postaci symbolicznej. Zastosowanie opisu behawioralnego umożliwia łatwe przenoszenie projektu na różne platformy sprzętowe. Narzędzie syntezy jest wówczas odpowiedzialne tylko za umiejętne wykorzystanie zasobów układu programowalnego. Zastosowanie opisu behawioralnego ze stanami opisanymi w sposób symboliczny zapewnia dodatkowy stopień swobody w procesie syntezy.

Doświadczenia zdobyte podczas projektowania układów cyfrowych wskazują, że nieodpowiedni opis projektowanego układu w języku opisu sprzętu prowadzi często do rozwiązań dalekich od oczekiwań pod względem zajmowanej powierzchni, czy szybkości działania. W trakcie wykonywania wstępnych eksperymentów okazało się, że synteza automatów modelowanych behawioralnie często prowadziła do bardzo nieefektywnych rozwiązań. Dla przykładu można podać wyniki syntezy automatu testowego *lion*. Synteza tego automatu opisanego behawioralnie, przeprowadzona za pomocą systemu Quartus II, doprowadziła do realizacji wykorzystującej: 12 komórek – stany opisane symbolicznie, 10 komórek – zdefiniowane słowa kodowe. Odpowiedni dobór słów kodowych oraz analiza TRI pozwoliła na znalezienie rozwiązania wykorzystującego tylko 3 komórki PAL.

W trakcie poszukiwania odpowiedniego opisu automatu w języku opisu sprzętu VHDL okazało się, że spodziewane wyniki daje synteza automatów opisanych na poziomie międzyrejestrowym (RTL), z komponentami opisanymi w postaci równań logicznych. Pomysł opiera się na zdefiniowaniu komponentów opisujących funkcję wzbudzeń wraz z przerzutnikiem dla każdej składowej funkcji przejść oddzielnie. Ponadto oddzielnie opisywane są również bloki kombinacyjne związane z każdym wyjściem. Układ sekwencyjny jest opisany strukturalnie. W opisie tym wykorzystuje się poprzednio utworzone komponenty. Na rysunku 2 przedstawiono strukturę (RTL) automatu testowego *lion*.



Rys. 2. Proponowana struktura modelu RTL dla automatu testowego *lion*
Fig. 2. The proposed RTL view of the *lion* automaton

Zaproponowany format opisu przykładowego komponentu opisującego składową funkcję przejść (bl_T0) przedstawiono na rysunku 3.

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY bl_T0 IS
  PORT (Clock : IN STD_LOGIC; I1, I0, tt1, tt0 : IN STD_LOGIC; T0 :
        OUT STD_LOGIC);
END bl_T0;
ARCHITECTURE rtl OF bl_T0 IS
BEGIN
  PROCESS (Clock)
  BEGIN
    if(Clock'event and Clock='1') then
      T0 <= (tt0 and tt1 and not I1) or (tt0 and not tt1 and not I0) or
              (not tt1 and I0 and I1) or (tt1 and I0 and not I1);
  END if;
  END PROCESS;
END;
```

Rys. 3. Funkcyjny format opisu komponentu *bl_T0* automatu testowego *lion*
Fig. 3. Functional model of the component *bl_T0* of the *lion* automaton

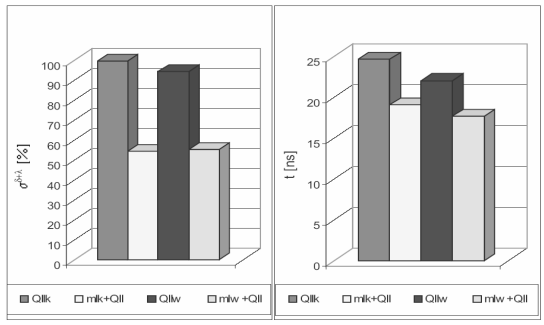
4. Wyniki eksperymentów

W celu porównania zaproponowanej metody kodowania i opisu z metodami zaimplementowanymi w systemach komercyjnych przeprowadzono syntezę 15 automatów testowych (AT) [8] wykorzystując system Quartus II oraz opracowany w Instytucie Elektroniki Politechniki Śląskiej prototypowy program do kodowania stanów wewnętrznych [2]. Po pierwsze, syntezie poddano każdy automat opisany behawioralnie w sposób symboliczny. Rozpatrywane AT poddano również procesowi kodowania stanów za pomocą prototypowego programu, a wyniki syntezy opisano w języku VHDL w sposób przedstawiony w rozdziale 3. Następnie tak opisane automaty poddano procesowi syntezy za pomocą systemu Quartus II. Syntezę przeprowadzono dwoma metodami, w których optymalizowano powierzchnię lub szybkość. W celu przedstawienia wyników, przyjęto następujące oznaczenia:

- QIIk – synteza automatu, opisanego w sposób symboliczny; optymalizacja powierzchni,
- QIIw – synteza automatu, opisanego w sposób symboliczny; optymalizacji szybkości,
- mlk+QII – kodowanie *mlk* + synteza Quartus II (optymalizacja powierzchni),
- mlw+QII – kodowanie *mlw* + synteza Quartus II (optymalizacja szybkości),

Podsumowanie wyników eksperymentów przedstawiono w postaci wykresu kolumnowego na rysunku 4, gdzie: $\sigma^{\delta+\lambda}$ oznacza liczbę komórek użytych do realizacji automatu. Za 100% przyjęto wyniki uzyskane metodą QIIk. Pozostałe kolumny przedstawiają wyniki w odniesieniu do metody QIIk. Na drugim wykresie przedstawiono wartość średnią czasu, który jest definiowany jako opóźnienie od narastającego zbocza na wejściu zegarowym do odpowiedzi układu na wyjściu w najdłuższej ścieżce (ang. „clock to output delay") wszystkich automatów.

Analiza wyników przedstawionych na wykresie kolumnowym pokazuje, że synteza automatów ukierunkowana na optymalizację szybkości QIIw doprowadziła również do mniejszej liczby komórek w stosunku do metody ukierunkowanej na optymalizację powierzchni QIIk. Wyniki uzyskane po syntezie automatów poddanych uprzednio kodowaniu stanów metodami *mlk* i *mlw* nie mają tej niekorzystnej cechy, a przede wszystkim prowadzą do rozwiązań wykorzystujących znacznie mniejszą liczbę komórek. Łączna liczba komórek została zredukowana o blisko połowę dla obydwu zaproponowanych metod kodowania. Średni czas *t* został zredukowany o blisko 20%. Widać więc, że zaproponowany opis automatu wykorzystujący wynik odpowiednio prowadzonego procesu kodowania może doprowadzić do znaczącej poprawy efektywności procesu syntezy systemów komercyjnych.



Rys. 4. Wyniki syntezy przeprowadzonej w systemie Quartus II
Fig. 4. Experimental results of the synthesis performed in Quartus II

5. Podsumowanie

W artykule przedstawiono dwa problemy syntezy automatów sekwencyjnych. Niezwykle istotnym etapem syntezy automatów sekwencyjnych jest proces kodowania stanów. Okazało się jednak, że nawet odpowiedni sposób kodowania może prowadzić do bardzo złych wyników z powodu wykonywania ostatnich etapów syntezy w narzędziach komercyjnych. Dopiero odpowiedni opis układu w języku opisu sprzętu VHDL, dostosowany do narzędzia syntezy oraz wykorzystywanego układu programowalnego, umożliwia przeprowadzenie efektywnej syntezy za pomocą narzędzi firmowych (np. Quartus II). Wykorzystanie narzędzi komercyjnych pozwala natychmiast wykonywać syntezę układów sekwencyjnych dla szerokiej gamy układów CPLD. Dzięki temu, przedstawione w artykule metody wspomagania projektowania układów cyfrowych realizowanych w strukturach programowalnych mogą być natychmiast wykorzystywane w projektowej praktyce inżynierskiej.

6. Literatura

[1] R. Czerwiński, D. Kania: State assignment for PAL-based CPLDs, Proceedings of Euromicro Symposium on Digital System Design, 2005, pp. 127-134
[2] R. Czerwiński: Kodowanie stanów automatów sekwencyjnych dla matrycowych struktur programowalnych typu PAL, rozprawa doktorska, Politechnika Śląska, Gliwice 2006
[3] R. Czerwiński, D. Kania, J. Kulisz: FSMs state encoding targeting at logic level minimization, Bulletin of the Polish Academy of Sciences, Vol.54, No.4, 2006, pp. 479-487
[4] T. Villa, A. Sangiovanni-Vincentelli: NOVA: State assignment for finite state machines for optimal two-level logic implementation, IEEE Trans. on Computer-Aided Design, vol. 9, pp. 905–924, 1990.
[5] E. Sentovich, K. Singh, L.Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P. Stephan, R. Brayton, A. Sangiovanni-Vincentelli: SIS: A System for Sequential Circuit Synthesis, Electronics Research Laboratory Memorandum, UC, Berkeley, 1992
[6] R. Czerwinski, D. Kania: Metody kodowania stanów automatów sekwencyjnych oparte na wyborze aktywności wyjść, RUC'2003, Szczecin, 2003, ss. 9–16.
[7] K. Skahill: VHDL for programmable logic, Addison-Wesley Publ. 1996 (tł. "Język VHDL. Projektowanie programowalnych układów logicznych", WNT, Warszawa, 2001)
[8] MCNC, LGSynth'91 benchmarks, Collaborative Benchmarking Laboratory, Department of Computer Science at North Carolina State University, http:// www.cbl.ncsu.edu/.

Artykuł recenzowany