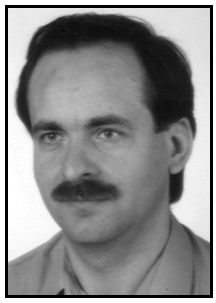
Stanisław DENIZIAK [1], **Mariusz WIŚNIEWSKI** [2]
[1] POLITECHNIKA KRAKOWSKA, KATEDRA INFORMATYKI TECHNICZNEJ
[2] POLITECHNIKA ŚWIĘTOKRZYSKA, KATEDRA INFORMATYKI

Dekompozycja funkcjonalna z wbudowanym kodowaniem wejść dla układów FPGA opartych o komórki LUT

Dr hab. inż. Stanisław DENIZIAK



Ukończył studia na Wydziale Elektroniki Politechniki Warszawskiej, obronił pracę doktorską w 1994r, a habilitacyjną w 2006r. Jest profesorem nadzwyczajnym w Katedrze Informatyki Technicznej Politechniki Krakowskiej oraz profesorem wizytującym w Katedrze Informatyki Politechniki Świętokrzyskiej. Jego zainteresowania naukowe to metody szybkiego prototypowania systemów informatycznych, projektowanie systemów wbudowanych, testowanie i diagnostyka systemów cyfrowych.

e-mail: pedenizi@cyf-kr.edu.pl

Mgr inż. Mariusz WIŚNIEWSKI

Ukończył studia na Wydziale Elektrotechniki, Automatyki i Informatyki Politechniki Świętokrzyskiej. gdzie od 2000 roku pracuje jako asystent w Katedrze Informatyki. Jego zainteresowania obejmują zagadnienia związane z wykorzystaniem rozwiązań informatycznych w syntezie układów cyfrowych, a także różne zagadnienia związane z inżynierią programowania.

e-mail: kinmw@eden.tu.kielce.pl

Streszczenie

W pracy przedstawiona jest metoda symbolicznej dekompozycji funkcji z wielowartościowymi wejściami. Poprzez zastosowanie funkcjonalnej dekompozycji symbolicznej, proces kodowania binarnego wartości wejść jest zintegrowany z dekompozycją. Algorytmy optymalizacji stosowane w metodzie mają na celu minimalizację kosztu implementacji funkcji w układach FPGA. Wyniki wykonanych eksperymentów wykazują dużą efektywność opracowanej metody, dla większości benchmarków uzyskano znacznie lepsze wyniki niż w dotychczas stosowanych metodach.

Słowa kluczowe: dekompozycja funkcji symbolicznych, FPGA, synteza logiczna.

An Integrated Input Encoding and Symbolic Functional Decomposition for LUT-Based FPGAs

Abstract

In this paper a method for decomposition of functions with multi-valued inputs is presented. Decomposition is performed simultaneously with encoding of symbolic values. In this way an impact of input encoding on decomposition efficiency is taken into consideration during optimization. The goal of our method is to find encoding that maximally simplifies functional decomposition. Input encoding is built in balanced decomposing strategy based on parallel end serial functional decompositions. Experimental results showed that the presented method significantly reduces the cost of FPGA implementations for most evaluated benchmarks.

Keywords: functional decomposition, FPGA, logic synthesis.

1. Wstęp

Dekompozycja funkcji binarnych jest jednym z podstawowych problemów w syntezie systemów cyfrowych implementowanych w układach FPGA [1]. Najbardziej obiecującą metodą dla układów FPGA opartych o komórki LUT jest dekompozycja funkcjonalna [2]. Funkcje systemów cyfrowych często są specyfikowane jako funkcje symboliczne, które na drodze kodowania przekształcane są na funkcje binarne. W tradycyjnym podejściu kodowanie zmiennych symbolicznych wykonywane jest przed dekompozycją, możliwa jest też dekompozycja funkcji symbolicznych [3] a następnie kodowanie uzyskanych funkcji. Jednak najlepsze wyniki uzyskuje się łącząc kodowanie i dekompozycję w jeden proces [10, 11].

Jednym z problemów syntezy symbolicznej jest kodowanie wejść. Typowe podejście do kodowania wejść bazuje na poszukiwaniu kodów spełniających ograniczenia wejściowe wynikające ze stosowania minimalizacji symbolicznej [4, 5]. Jednakże wszystkie te metody zostały opracowane na potrzeby minimalizacji logiki dwupoziomowej.

W przypadku faktoryzacji zmiennej symbolicznej [6] lub dekompozycji symbolicznej [7] kodowanie wykonuje się po dekompozycji, zatem rozdzielanie bitów kodu nie jest tu uwzględniane. Jedno z pierwszych rozwiązań dotyczących kodowania wejść w układach FPGA zostało zaprezentowane w [8]. Dekompozycja szeregowa została sprecyzowana jako problem kodowania wejść. Jednakże proponowana metoda została wbudowana w klasyczne metody dekompozycji, które nie są efektywne dla układów FPGA. Podobne rozwiązanie można znaleźć w [9]. Najbardziej obiecującą metodą syntezy funkcji wielowartościowych jest symboliczna dekompozycja funkcjonalna [11]. Zostało pokazane, że ta metoda znacząco zmniejsza koszt implementacji automatów w układach FPGA [10, 11].

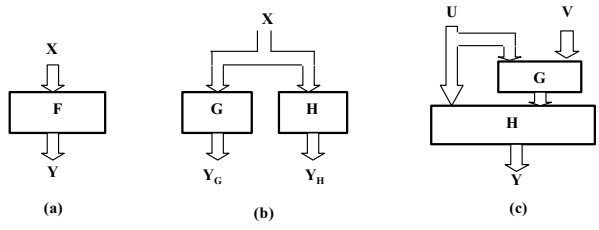
Niniejsza praca prezentuje nową metodę dekompozycji funkcji integrującą kodowanie z dekompozycją wejść wielowartościowych. Celem naszej metody jest znalezienie sposobu kodowania upraszczającego dekompozycję funkcjonalną. Kodowanie wejść jest wbudowane w dekompozycje równoległą i szeregową [2]. Uzyskane wyniki pokazują, że metoda znacząco zmniejsza koszt implementacji układów w urządzeniach FPGA.

Następny rozdział zawiera podstawy teoretyczne dekompozycji funkcjonalnej i kodowania wejść. Rozdział 3 przedstawia metodę kodowania wejść. Wyniki eksperymentów znajdują się w rozdziale 4. Artykuł kończy podsumowanie.

2. Wiadomości podstawowe

2.1. Dekompozycja funkcjonalna

Niech F będzie funkcją wielowyjściową (rys. 1a). Funkcja może zostać poddana dekompozycji równoległej lub szeregowej. Dekompozycja równoległa przedstawia funkcję F za pomocą funkcji G i H, rozdzielających wyjścia funkcji F (rys. 1b). Ideą dekompozycji szeregowej jest przedstawienie funkcji $F(X)$ poprzez funkcje G i H, tak że $F=H(U,G(U,V))$, gdzie $U \cup V=X$ (rys. 1c).



Rys. 1. Funkcja pierwotna (a), po dekompozycji równoległej (b) i szeregowej (c)
Fig. 1. Primal function (a), after parallel decomposition (b), after serial decomposition (c)

Niech funkcja F ma n_F wejść i m_F wyjść. Załóżmy, że taka funkcja będzie zaimplementowana w urządzeniu FPGA zawierającym komórki LUT z n_L wejściami i m_L wyjściami. Spodziewany koszt implementacji (*EIC*) funkcji F wyrażamy następująco:

$$EIC(F) = \left\lceil \frac{n_F}{n_L} \right\rceil * \left\lceil \frac{m_F}{m_L} \right\rceil \quad (1)$$

Optymalizacja powinna minimalizować *EIC*, zatem najlepsza dekompozycja jest wtedy, gdy suma *EIC(G) + EIC(H)* jest najmniejsza.

2.2. Kodowanie wejść

Niech funkcja F posiada wejścia binarne $X=\{x_1,...,x_n\}$, symboliczne $S=\{s_1,...,s_k\}$ i binarne wyjścia $Y=\{y_1,...,y_m\}$. Ponieważ optymalna dekompozycja może wymagać rozdzielenia kodowanych zmiennych, wszystkie wejścia symboliczne powinny zostać zakodowane w zbiorze binarnym $Q=\{q_1, ... ,q_p\}$ przed dekompozycją. Ponadto kodowanie powinno gwarantować znalezienie najlepszej dekompozycji, tj. mającej najmniejsze *EIC(G)+EIC(H)*.

Problem kodowania wejść może być przedstawiony przy pomocy rachunku nakryć [1] uogólnionego na funkcje wielowartościowe [7]:

Obserwacja 1. Kodowanie wejścia symbolicznego jest optymalne dla dekompozycji równoległej, jeśli istnieją podzbiory wyjść Y_G,Y_H i podzbiory wejść V_1,V_2 takie, że: $Y_G \cup Y_H = Y$, $Y_G \cap Y_H = \phi$, i $V_1, V_2 \subseteq X \cup Q$, oraz $\beta_{V_1} \le \beta_{Y_G}$, $\beta_{V_2} \le \beta_{Y_H}$ i *EIC(G)+EIC(H)* jest najmniejsze, gdzie:

$$\beta_{V_1} = \prod_{x_k \in V_1} \beta_{x_k} * \prod_{q_j \in V_1} \beta_{q_j} \qquad \beta_{V_2} = \prod_{x_k \in V_2} \beta_{x_k} * \prod_{q_j \in V_2} \beta_{q_j}$$
$$\beta_{Y_G} = \prod_{y_k \in Y_G} \beta_{y_k} \qquad \beta_{Y_H} = \prod_{y_k \in Y_H} \beta_{y_k} \quad (2)$$

Obserwacja 2. Kodowanie wejścia symbolicznego jest optymalne dla dekompozycji szeregowej, jeśli istnieją zbiory $V=\{x_{v1},..., x_{v2}, q_1,...,q_{p1}\}$ i $U=\{x_{u1},..., x_{u2}, q_{p2},...,q_p\}$, oraz nakrycie β_g, takie że: $\beta_V \le \beta_g$ i $\beta_U * \beta_g \le \beta_F$ oraz *EIC(G)+EIC(H)* jest najmniejsze, gdzie:

$$\beta_V = \prod_{x_k \in V} \beta_{x_k} * \prod_{q_j \in V} \beta_{q_j} \qquad \beta_U = \prod_{x_k \in U} \beta_{x_k} * \prod_{q_j \in U} \beta_{q_j} \quad (3)$$

3. Dekompozycja z kodowaniem wejść

3.1. Kodowanie wejść upraszczające dekompozycje równoległą

Szukanie najlepszej dekompozycji równoległej wymaga następujących kroków:

```
procedure parallel_decomposition(F)
{
  find_sets_YG_and_YH;
  find_decomposition_for_set_YG(G);
  find_decomposition_for_set_YH(H, QG);
  check_for_disjoint_coding_possibility(QG, QH);
}
```

W pierwszej kolejności wyznaczane są zbiory wyjść związane z funkcjami G i H. Z powodu wielu możliwości, zastosowano heurystyki minimalizujące przestrzeń poszukiwań. Dla wybranych zbiorów wykonuje się dekompozycje, z których wybierana jest najlepsza.

Głównym zadaniem w dekompozycji dla zbioru Y_G jest znalezienie częściowego zakodowania zmiennej symbolicznej S_i. W tym celu należy znaleźć nakrycie β_{Q_G} , składające się z bloków nakrycia β_{S_i} , takie, aby zachodziła zależność $\beta_{X_G} * \beta_{Q_G} \le \beta_{Y_G}$ (gdzie $X_G \subseteq X$ i $Q_G \subseteq Q$). Sposób wyznaczania nakrycia β_{Q_G} przedstawia następująca procedura:

```
procedure find_decomposition_for_set_YG(G)
{  repeat
     max_QG_blk := null;
     for każdego bloku β_Si do {
       A := wybranie następnego bloku z β_Si;
       for każdego bloku B z β_Si innego niż A do {
         QG_block := A + B;
         if QG_block ≤ β_YG then A := QG_block;
       }
       if ilość bloków w A > max_QG_blk then max_QG_blk := A;
     }
     zapamiętanie max_QG_blk jako bloku nakrycia β_QG;
     usuniecie użytego bloku z β_Si;
   until nie ma bloków w β_Si;
}
```

Dekompozycja funkcji ze zbioru Y_G jest wykonywana w dwóch krokach:

- Algorytm poszukuje maksymalnej ilości bloków nakrycia β_{S_i} , które można połączyć. W tym celu każdy blok z nakrycia β_{S_i} jest sumowany z pozostałymi, jednocześnie sprawdzana jest zależność $\beta_{X_G} * \beta_{Q_G} \le \beta_{Y_G}$. Jeśli jest spełniona, to otrzymany blok jest zapamiętywany i sprawdzeniu podlega następny bloku z nakrycia β_{S_i} .
- Po sprawdzeniu wszystkich bloków, największy jest zapamiętywany w nakryciu β_{Q_G} .

Dekompozycja funkcji ze zbioru Y_H jest bardziej złożona. Także w tym przypadku należy znaleźć nakrycie β_{Q_H} , jednak muszą być spełnione dwa warunki: $\beta_{X_H} * \beta_{Q_H} \le \beta_{Y_H}$ i $\beta_{Q_G} * \beta_{Q_H} = \beta_{S_i}$. Następujący pseudokod obrazuje ten algorytm:

```
procedure find_decomposition_for_set_YH(H, QG')
{  repeat
     max_QH_blk := null;
     for każdego bloku β_Si do {
       A := wybranie następnego bloku z β_Si;
       for każdego bloku B z β_Si innego niż A do {
         QH_block := A + B;
         if β_QG * QH_block ≤ β_Si and QH_block ≤ β_YH then A := QH_block;
       }
       if ilość bloków w bloku A > max_QH_blk then max_QH_blk := A;
     }
     zapamiętanie bloku max_QH_blk w nakryciu β_QH;
     usunięcie użytego bloku z β_Si;
   until nie ma bloków w β_Si;
}
```

Dekompozycja Y_H wykonywana jest w dwóch krokach:

- Algorytm poszukuje maksymalnej ilości bloków nakrycia β_{S_i} , które można połączyć. Każdy blok z nakrycia β_{S_i} jest sumowany z pozostałymi, przy czym sprawdzane są zależności

$\beta_{X_H} * \beta_{Q_H} \leq \beta_{Y_H}$ oraz $\beta_{Q_G} * \beta_{Q_H} = \beta_{S_i}$. Jeśli zachodzą, to otrzymany blok jest zapamiętywany i sprawdzany jest następny blok z nakrycia β_{S_i}.

- Po sprawdzeniu wszystkich bloków, największy jest zapamiętywany w nakryciu β_{Q_H}.

Dekompozycja równoległa dla przykładowej funkcji z tabeli 1 przebiega następująco. Dla $Y_G=\{y_2,y_3\}$ i $Y_H=\{y_1\}$ wyznaczono $\beta_{Y_G} = \{\overline{1,2,6};\overline{5,10};\overline{4,9};\overline{3,7,8}\}$ i $\beta_{Y_H} = \{\overline{1,2,3,4,10};\overline{5,6,7,8,9}\}$. Następnie algorytm poszukuje najlepszego nakrycia β_{Q_G}. Dla $X_G=\{x_1,x_2\}$ znaleziono nakrycie $\beta_{Q_G} = \{\overline{4,9};\overline{5,10};\overline{1,2,3};\overline{6,7,8}\}$. Ponieważ $\beta_{X_G} * \beta_{Q_G} \leq \beta_{Y_G}$ zatem $G=f(X_G,Q_G)$ i zmienną Q_G można zakodować na 2 bitach, a funkcja G będzie wymagała jednej komórki LUT o 4 wejściach. Podobnie dla funkcji H, najlepsza dekompozycja jest dla $X_H=\{\}$ i $\beta_{Q_H} = \{\overline{1,2,3,4,10};\overline{5,6,7,8,9}\}$. Końcowy wynik dekompozycji został zamieszczony w tabeli 2.

Tab. 1. Funkcja F
Tab. 1. Function F

	x1	x2	x3	y1	y2	y3
1	0	-	S0	0	0	0
2	-	0	S0	0	0	0
3	1	1	S0	0	1	1
4	-	-	S1	0	1	0
5	-	-	S2	1	0	1
6	1	0	S3	1	0	0
7	0	-	S3	1	1	1
8	-	1	S3	1	1	1
9	-	-	S4	1	1	0
10	-	-	S5	0	0	1

Tab. 2. Funkcja F po dekompozycji równoległej
Tab. 2. Function F after parallel decomposition

G						H	
x1	x2	q0	q1	y2	y3	q2	y1
0	-	1	0	0	0	0	0
1	1	1	0	1	1	1	1
-	-	0	0	1	0		
-	-	0	1	0	1		
1	0	1	1	0	0		
0	-	1	1	1	1		

3.2. Kodowanie wejść upraszczające dekompozycje szeregową

Głównym problem w dekompozycji szeregowej jest znalezienie najlepszej reprezentacji nakryć β_{V_Q}, β_{U_Q} oraz funkcji G. Algorytm składa się z dwóch procedur: *determine_sets_UV* i *find_serial_decomp*. Następujący pseudokod przedstawia główne kroki procedury *determine_sets_UV*:

```
procedure determine_sets_UV(X, F)
{  for wszystkich podzbiorów zbioru X do {
     U, V := rozdzielenie X na dwa rozłączne podzbiory;
     A := β_{V_I} * β_{S_i};
     B := β_{U_I} * β_{S_i};
     if A * B ≤ β_F then dodanie UV do listy;
   } }
```

Na początku wybierane są dwa rozłączne zbiory U i V. Następnie sprawdzane jest istnienie dekompozycji szeregowej. Jeżeli zależność $A*B \leq \beta_F$ jest spełniona, to dekompozycja istnieje i zbiory U i V są zapamiętywane. Po wyznaczeniu właściwych zbiorów U i V można przejść do wykonania dekompozycji szeregowej. Algorytm jest następujący:

```
procedure find_serial_decomp(U, V, F)
{  wybranie wartości początkowej dla β_{V_Q};
   wybranie wartości początkowej dla β_{U_Q};
   for każdego bloku z β_{S_i} do {
      block_B_X := następny blok z β_{S_i};
      A := obecny β_{V_Q} □block_B_X;
      B_G_tmp := A * β_{V_I};
      for każdego bloku z β_{U_Q} do {
         B := następny blok z β_{U_Q} □block_B_X;
         if A * B ≤ β_{S_i} then {
            β_{U_Q} := β_{U_Q} □block_B_X;
            find_function_G(A, β_{U_Q});
            zapamiętanie częściowego β_{U_Q} i β_G;
         } }
      znalezienie najlepszego częściowego β_{U_Q} i β_G;
      usunięcie użytego bloku z β_{S_i};
   } }
```

Algorytm próbuje znaleźć najlepszą funkcję G i najlepsze nakrycia β_{V_Q} i β_{U_Q}. Powyższe elementy są ściśle ze sobą związane i będą wyznaczane jednocześnie. Dodatkowo muszą być spełnione zależności: $\beta_{V_I} * \beta_{V_Q} \leq \beta_G$, $\beta_{U_I} * \beta_{U_Q} * \beta_G \leq \beta_F$ i $\beta_{V_Q} * \beta_{U_Q} = \beta_{S_i}$. Ostatnie równanie gwarantuje, że wszystkie stany zmiennej symboliczne będą zakodowane poprzez różne kody. Algorytm jest wykonywany w dwóch krokach:

- Na początku wyznaczane są bloki nakrycia β_{V_Q}. Celem tego jest znalezienie bloków, dla których minimalna ilość bloków nakrycia β_G nie jest większa niż 2^{O_G} (gdzie O_G jest spodziewaną ilością wyjść funkcji G) oraz ilość bloków w nakryciu β_{U_Q} jest najmniejsza.
- W każdym przebiegu pierwszego kroku wybierane są najlepsze nakrycia β_{V_Q} i β_{U_Q}. Dla tych nakryć generowana jest funkcja G mająca najmniejszą ilość wyjść.

Po sprawdzeniu wszystkich bloków nakrycia β_{S_i} wybrane zostaną najlepsze nakrycia β_{V_Q} i β_{U_Q}. Następnie największy blok jest zapamiętywany jako blok β_{U_Q}.

Tab. 3. Funkcja F po dekompozycji szeregowej
Tab. 3. Function F after serial decomposition

G					H						
x1	q0	q1	q2	g	x2	g	q3	q4	y1	y2	y3
0	0	0	0	0	-	0	0	0	0	0	0
-	0	0	0	-	1	1	0	0	0	1	1
-	0	0	1	0	-	0	0	1	0	1	0
-	0	1	0	1	-	1	0	1	1	0	1
1	1	0	1	0	0	0	1	1	1	0	0
0	1	0	1	1	-	1	1	1	1	1	1
-	0	1	1	0	-	0	1	0	1	1	0
-	1	0	0	1	-	1	1	0	0	0	1

Dla podzbiorów $V_I=\{x_1\}$ i $U_I=\{x_2\}$ funkcji F z tabeli 1 wyznaczono następujące optymalne nakrycia $\beta_{V_Q}=\{\overline{1,2,3};\overline{4};\overline{5};\overline{9};\overline{10};\overline{6,7,8}\}$ i $\beta_G=\{\overline{1,2,4,6,8,9};\overline{2,3,5,7,8,10}\}$. Funkcja G ma jedno wyjście, a zmienna V_Q wymaga do zakodowania 3 bitów (1 komórka LUT o 4 wejściach). Wyznaczone nakrycie wymaga $\beta_{U_Q}=\{\overline{1,2,3};\overline{4,5};\overline{9,10};\overline{6,7,8}\}$ do zakodowania 2 bitów. Wynik dekompozycji przedstawiony jest w tabeli 3.

3.3. Zrównoważone kodowanie wielopoziomowe i dekompozycja

Omówione procedury wykonujące dekompozycje są powtarzane dla funkcji G i H aż do uzyskania funkcji mających nie więcej wejść niż mają komórki LUT w układzie FPGA. Zaobserwowano, że stosowanie obu rodzajów dekompozycji powinno być zrównoważone, zależnie od celu optymalizacji[2]. Dekompozycja szeregowa jest bardziej efektywna w optymalizacji wykorzystania powierzchni, podczas, gdy dekompozycja równoległa jest użyteczniejsza w optymalizacji układu pod względem szybkości. W prezentowanym rozwiązaniu stosowano dekompozycje zgodnie z następującymi zasadami:

1. z dekompozycji równoległych i szeregowych wybierana jest ta z najmniejszym EIC,
2. jeśli najlepsza dekompozycja równoległa i szeregowa mają równe EIC, to: jeśli $|O| > |I| + \lceil \log_2 S_i \rceil$ to wybierana jest dekompozycja równoległa, w przeciwnym razie wybrana zostanie dekompozycja szeregowa.

Celem optymalizacji jest minimalizacja liczby komórek LUT, jednak można także zmniejszyć ilości poziomów układu poprzez zwiększenie wagi dekompozycji równoległej.

4. Wyniki wykonanych eksperymentów

Wydajność prezentowanej metody została zweryfikowana przy pomocy benchmarków MCNC. Ponieważ wszystkie benchmarki mają wejścia binarne, wybrane wejścia zostały zgrupowane i zakodowane symboliczne. W tabeli 4 zostały przedstawione uzyskane przez nas wyniki oraz rezultaty wykonania innych, dobrze znanych programów syntezy dla urządzeń FPGA[2]. Druga kolumna w tabeli 4 zawiera liczbę wejść binarnych, jakie zostały zgrupowane w jedno wejście wielowartościowe oraz ostateczną długość uzyskanego kodu. Wyniki w następnych kolumnach oznaczają odpowiednio ilość komórek LUT potrzebna do implementacji (pierwsza liczba) oraz ilość poziomów logicznych (liczba po znaku /). Ponieważ w każdym przypadku kodowanie zwiększyło ilość wejść można przypuszczać, że minimalna długość kodu nie zawsze jest optymalna w dekompozycji funkcjonalnej.

Tab. 4. Wyniki eksperymentów
Tab. 4. Experimental results

Nazwa	Wej/ kod	Nasza metoda	DEMAIN	BDD	Mispga-d	Chortle-d	FlowMap	ASYL
5px1	7/9	9/1	9/2	19/2	21/2	29/4	25/3	13/3
9sym	9/10	4/3	5/3	6/3	10/2	20/3	13/3	4/2
clip	9/11	5/2	16/2	-	54/4	-	-	46/4
f51m	8/9	8/1	10/2	81/3	23/4	65/5	-	16/3
misex1	8/12	7/1	8/2	18/2	17/2	25/3	25/3	13/2
rd73	7/10	4/2	5/2	8/2	8/2	52/4	-	8/2
rd84	8/12	5/2	7/2	13/3	13/3	61/4	43/4	14/3
sao2	10/14	5/2	18/3	26/4	45/5	58/5	-	36/3

Bardziej interesujące jest porównanie pomiędzy prezentowaną metodą a programem DEMAIN[2]. Nasze podejście wykorzystuje podobną strategię dekompozycji, a różni się jedynie sposobem kodowania. Widać z tego, że stosowanie dekompozycji wraz z kodowaniem znacząco zmniejsza koszt implementacji dla większości układów.

5. Podsumowanie

W artykule została przedstawiona wydajna metoda dekompozycji funkcji z wejściami wielowartościowymi. Wyniki eksperymentów pokazują, że metoda jest znacznie bardziej efektywna niż inne, znane z literatury, dla większości benchmarków zostały uzyskane najtańsze implementacje w urządzeniach FPGA.

Zostało także pokazane, że kodowanie binarne wejść wielowartościowych ma olbrzymie znaczenie dla jakości dekompozycji. Prezentowana metoda wykonuje kodowanie po dekompozycji symbolicznej, eliminując tą zależność. Ponadto zaobserwowano, że stosowanie kodów o minimalnej długości nie gwarantuje znalezienia najlepszej dekompozycji.

Następne prace będą dotyczyły poszukiwania nowych strategii dekompozycji dla nowoczesnych układów FPGA zbudowanych z różnych typów komórek LUT, jak również rozszerzenie metody na kodowanie wejść/wyjść oraz kodowanie stanów.

6. Literatura

[1] J.A.Brzozowski, T.Łuba, "Decomposition of Boolean Functions Specified by Cubes", Journal of Mult.-Valued Logic & Soft Computing, vol.9, 2003, pp.377-417.

[2] M. Nowicka, T. Łuba, M. Rawski, "FPGA-Based Decomposition of Boolean Functions. Algorithms and Implementation", Proc. of the 6th Int. Conf. on Advanced Computer Systems, 1999, pp.501-509.

[3] R.K.Brayton, S.P. Khatri, "Multi-valued Logic Synthesis", Proc. of the Int. Conference on VLSI Design, 1999, pp.196-205

[4] P.Ashar, S.Devadas, A.R.Newton, "Sequential Logic Synthesis", Kluwer Academic Pub., Norwell, 1992

[5] M.Martinez, M.J.Avedillo, J.M.Quintana, J.L.Huertas, "COPAS: A New Algorithm for the Partial Input Encoding Problem", VLSI Design, Vol. 14 (2), 2002, pp.171-181.

[6] S.Malik, L.Lavagno, R.K.Brayton, A.Sangiovanni-Vincentelli, "Symbolic minimization of multilevel logic and the input encoding problem", IEEE Transactions on CAD, vol.11, no.7, 1992, pp.825–843.

[7] J.A.Brzozowski, J.J.Lou, "Blanket algebra for multiple-valued function decomposition", Proc. of the International Workshop on Formal Languages and Computer Systems 1997, in. Algebraic Engineering, C.L.Nehaniv and M.Ito,eds.World Scientific, 1999, pp.262-276.

[8] R.Murgai, R.K.Brayton, A. Sangiovanni-Vincentelli, "Optimum Functional Decomposition Using Encoding", Proc. of the DAC, 1994, pp. 408-414.

[9] V.Muthukumar, "An Improved Input-Output Encoding Approach for Functional Decomposition", Proc. of the Euromicro DSD, 2001.

[10] S.Deniziak, „Kodowanie stanów dla układów FPD o architekturze opartej o komórki LUT", VII Krajowa Konferencja Reprogramowalne Układy Cyfrowe, 2004, str.19-26.

[11] M.Rawski, H.Selvaraj, T.Łuba, P.Szotkowski, "Application of symbolic functional decomposition concept in FSM implementation targeting FPGA devices", Proc. of the 6th Intl Conf. on Computational Intelligence and Multimedia Applications,2005, pp.153-158.

Artykuł recenzowany