Mirosław MOŚCICKI
POLITECHNIKA SZCZECIŃSKA, WYDZIAŁ INFORMATYKI

Algorytm generowania równań boolowskich dla operatorów relacji języka VHDL

Mgr inż. Mirosław MOŚCICKI

Mgr inż. Mirosław Mościcki jest absolwentem Wydziału Informatyki Politechniki Szczecińskiej (2000). Obecnie pracuje na stanowisku asystenta w Katedrze Technik Programowania. Jest specjalistą w zakresie tworzenia oraz testowania oprogramowania. Od 1997 roku uczestniczył w wielu projektach informatycznych jako programista, projektant oraz tester.

e-mail: mmoscicki@wi.ps.pl

Streszczenie

W artykule zaprezentowano sposób generowania równań boolowskich dla operacji porównania języka VHDL. W języku VHDL istnieje 6 operatorów relacji: =, /=, <, <=, >, >=, które pozwalają na stwierdzenie, czy pomiędzy operandami zachodzi określona relacja. Operandy muszą być tego samego typu, natomiast wynik jest zawsze typu BOOLEAN. W artykule zaprezentowano algorytm użyty dla wszystkich operatorów relacji. W przypadku gdy operandy są typu tablicowego algorytmy generowania równań boolowskich dla operatorów relacji są dość mocno rozbudowanie i z tego względu właśnie te algorytmy zostały szczegółowo przedstawione. Pokazano praktyczne zastosowanie opisanego algorytmu.

Słowa kluczowe: język VHDL, równania boolowskie, operacje porównania.

Boolean equations generation algorithm for relational operators in VHDL language

Abstract

In this paper is proposed and described a Boolean Equation generation algorithm for relational operators in VHDL language. There are 6 relational operators: =, /=, <, <=, >, >=. Relational operators, compare two operands of the same base type and return a BOOLEAN value. IEEE VHDL defines the equality (=) and inequality (/=) operators for all types. Two operands are equal if they represent the same value. For array and record types, IEEE VHDL compares corresponding elements of the operands. IEEE VHDL defines the ordering operators (<, <=, >, and >=) for all enumerated types, integer types, and one-dimensional arrays of enumeration or integer types. If the two arrays have different lengths and the shorter array matches the first part of the longer array, the shorter one is ordered before the longer. Thus, the bit vector 101 is less than 101000. Arrays are compared from left to right, regardless of their index ranges (to or downto). There are shown practical application of the algorithm.

Keywords: VHDL language, Boolean equations, relational operators.

1. Wstęp

Od kilku lat w Katedrze Technik Programowania Wydziału Informatyki Politechniki Szczecińskiej realizowany jest projekt [1], którego celem jest stworzenie kompilatora dokonującego konwersji pliku zawierającego program napisany w języku VHDL na równania boolowskie. Równania boolowskie są bardzo dobrym materiałem wyjściowym do dalszej obróbki ponieważ jest to forma matematyczna. Na ich podstawie można stworzyć układy cyfrowe realizujące określone zadania, lub poddać je minimalizacji. Równania boolowskie mogą być wykorzystywane przy produkcji układów FPGA [2].

Język VHDL służy do projektowania cyfrowych układów logicznych. W niniejszym opracowaniu zostanie omówiony problem generowania równań boolowskich dla operatorów relacji.

2. Algorytmy generowania równań boolowskich

W języku VHDL zostało zdefiniowanych sześć następujących operatorów relacji: : =, /=, <, <=, >, >= [3]. Operandy zawsze muszą być tego samego typu, natomiast wynik jest zawsze typu BOOLEAN. Operatory równości (=) oraz nierówności (/=) zostały zdefiniowane dla wszystkich dostępnych w języku VHDL typów. W przypadku pozostałych relacji operandy mogą być typu skalarnego lub tablicowego o elementach typu dyskretnego. Operator równości zwraca wartość logiczną TRUE tylko wtedy, gdy oba operandy posiadają jednakowe wartości, natomiast wartość FALSE, kiedy wartości są różne. Dla różnych typów operandów relacji obowiązują określone reguły ich porównania. W przypadku typu skalarnego, wartości skalarne operandów są sobie równe jedynie wtedy, gdy ich wartości są takie same. Dwie wartości typu złożonego są równe wtedy, gdy każda wartość elementu lewego operandu jest taka sama, jak odpowiadająca mu wartość elementu w prawym operandzie i odwrotnie. Operatory porządkowe: mniejszości (<), mniejszości lub równości (<=), większości (>), większości lub równości (>=), zwracają wartość logiczną TRUE wtedy, gdy spełniony jest warunek w danej relacji, w przeciwnym przypadku zwracana jest wartość FALSE. Przykładowo dla operatora większości (>) lewy operand jest większy od prawego kiedy spełnione są warunki:

1. Dla wartości typu skalarnego warunek jest spełniony, kiedy wartość lewego operandu jest większa od wartości prawego.
2. Dla typu tablicowego o elementach dyskretnych relacja większości jest spełniona w następujących przypadkach:
   - Lewy operand jest tablicą niepustą a prawy jest tablicą pustą, lub
   - Oba operandy są niepustymi tablicami oraz pierwszy element lewej tablicy jest większy od odpowiadającego mu prawego elementu tablicy, lub
   - Pierwszy element lewej tablicy jest równy odpowiadającemu mu elementowi prawej tablicy, natomiast drugi element jest większy od odpowiadającego mu drugiemu elementu prawej tablicy, lub
   - Kolejnych n (n>1) elementów lewej tablicy jest równych kolejnym n elementom prawej tablicy, natomiast n+1 element lewej jest większy od n+1 odpowiadającego mu elementu prawej tablicy.

W równaniach boolowskich powstających w procesie kompilacji wykorzystujemy tylko trzy podstawowe operacje: or (|), and (&), not (!) [4]. Związane to jest z pierwotnym zastosowaniem generowanych równań. Ich głównym przeznaczeniem była symulacja i weryfikacja na komputerze wieloprocesorowym opracowanym specjalnie do tego zadania. Jednostka taka składała się z wielkiej liczby procesorów zdolnych do wykonywania bardzo prostych operacji. Z tego też względu w istniejącym formacie wyjściowym nie można korzystać z podprogramów. Taki format został przyjęty na życzenie firmy ALDEC dla której początkowo był opracowywany kompilator.

Przed przystąpieniem do generowania równań boolowskich program źródłowy w języku VHDL musi zostać poddany analizie leksykalnej, syntaktycznej oraz semantycznej [5, 6].

Podczas procesu kompilacji w sytuacji gdy pojawi się wywołanie operatora relacji, wywoływana jest odpowiednia funkcja generatora kodu. Funkcja ta ma za zadanie wygenerowanie, w zależności od argumentów wejściowych, odpowiednich rów-

nań w postaci tekstowej. W generatorze kodu występuje 6 różnych funkcji:

```
char **_inequ(OPRND *op1,OPRND *op2)
char **_equ(OPRND *_op1, OPRND *_op2)
char **_more(OPRND *_op1,OPRND *_op2)
char **_less(OPRND *_op1,OPRND *_op2)
char **_eless(OPRND *op1,OPRND *op2)
char **_emore(OPRND *op1,OPRND *op2)
```

Każda z funkcji posiada dwa argumenty. Wskazują one na strukturę *Variable* (przykład 1), która zawiera informację o typie zmiennej biorącej udział w operacji logicznej. Uzyskane równania boolowskie są zwracane jako parametr wyjściowy funkcji.

Równania boolowskie dla wszystkich operatorów są generowane według określonego algorytmu, zmienia się tylko szablon na podstawie którego generowane są równania.

Algorytmy wykorzystywane w procesie generowania równań boolowskich dla operatorów języka VHDL są stosunkowo proste podczas operacji równości i nierówności, a bardziej skompilowane dla pozostałych operacji. W przypadku operatora równości i nierówności wystarczy wykorzystać funkcję & oraz ! na wszystkich bitach porównywanych operandów. Przykład 2 pokazuje w jaki sposób generowane są równania dla operatora „=".

Przykład 2. Operator =
Example 2. Operator =

```
entity test is
   port (
          a,b: in BIT_vector(0 to 4);
          z: out boolean
   );
end test;

architecture arch of test is
begin
z<=  (b = a);
end arch;
```

Wygenerowane równania boolowskie:

```
z=((((((b(0)&a(0))|(!b(0)&!a(0)))&((b(1)&a(1))|(!b(1)&!a(1))))&
   ((b(2)&a(2))|(!b(2)&!a(2))))&((b(3)&a(3))|(!b(3)&!a(3))))&
                ((b(4)&a(4))|(!b(4)&!a(4))));
```

W przypadku porównania typów jednobitowych proces generowania równań boolowskich jak i wygenerowane równania są bardzo proste. Zostanie to pokazane na przykładzie operatora „większe lub równe"

Przykład 4. Operator >=
Example 4. Operator >=

```
entity test is
   port (
      a,b: in bit;
      z: out boolean
   );
end test;

architecture test5 of test is
begin
z<=(a >= b);
end test5;
```

Wygenerowane równania boolowskie:

```
z=!(!a&b);
```

W przypadku operatora >= wartość TRUE ('1') będzie zwrócona wtedy, gdy oba operandy są sobie równe oraz gdy pierwszy operand jest większy od drugiego, w przeciwnym razie wynik przyjmie wartość FALSE ('0'). Przeanalizujmy uzyskane równania dla wszystkich możliwych kombinacji bitów:

Tab. 1. Operator >=
Tab. 1. Operator >=

A	B	!A	!A&B	!(!A&B)
0	0	1	0	1
0	1	1	1	0
1	0	0	0	1
1	1	0	0	1

Jak widać na podstawie powyższej tabeli uzyskane równania są prawidłowe.

Stopień skomplikowania wygenerowanych równań boolowskich rośnie wraz ze złożonością operandów.

3. Podsumowanie

Bardzo istotną sprawą podczas generowania równań boolowskich jest opracowanie szybkich i skutecznych algorytmów dla operatorów relacji. Proces skomplikowania równań dla operatorów relacji rośnie wraz z długością poszczególnych operandów. W przypadku operandów jednobitowych sprawa jest bardzo prosta, komplikuje się natomiast dla operandów wielobitowych (integer, różnego rodzaju tablice itp.). Poprawność, oraz optymalność wygenerowanych równań dla poszczególnych operatorów relacji ma kluczowy wpływ na jakość całego kompilatora. Wynika to z częstotliwości występowania operacji porównania w programach tworzonych w języku. Zastosowany w generatorze kodu algorytm został przetestowany pod względem poprawności dla kilkuset różnych przykładów, przy uwzględnieniu maksymalnie wielu różnych wartości operandów. Uzyskane wyniki pozwalają stwierdzić, że generowane równania dla operatorów relacji są prawidłowe. W dalszych pracach trzeba się będzie skupić na optymalności generowanych równań, ponieważ tylko szybkie algorytmy umożliwią przemysłowe zastosowanie kompilatora powstającego w Katedrze Technik Programowania Wydziału Informatyki Politechniki Szczecińskiej.

4. Literatura

[1] Bielecki W., Hayduke S., Drążkowski R., Liersz M., Radziewicz M., Błaszyński P.: Organizacja kompilatora do syntezy układów logicznych z syntezowalnego podzbioru języka VHDL, Materiały IV Sesji Naukowej Informatyki, INFORMA, Szczecin 1999.

[2] Sołdek J., Miejsce układów reprogramowalnych w informatyce, Materiały I Krajowej Konferencji Naukowej. Reprogramowalne układy cyfrowe.

[3] Wrona W., VHDL język opisu i projektowania układów cyfrowych, Wydawnictwo pracowni komputerowej Jacka Skalmierskiego, Gliwice 1998.

[4] FPGA Express, VHDL Reference Manual, 1997.

[5] Błaszyński P, Drążkowski R.: Organizacja analizatora semantycznego kompilatora języka VHDL do syntezy układów logicznych, P., R., Materiały III krajowej konferencji naukowej RUC'2000, INFORMA, Szczecin 2000.

[6] Błaszyński P.: Generacja i wyszukiwanie wartości semantycznych w kompilatorze języka VHDL służącym do generacji równań boolowskich, Materiały V Sesji Naukowej Informatyki, INFORMA, Szczecin 2000.

Artykuł recenzowany