О несинтезируемых конструкциях языка VHDL

Петр Бибило (г. Минск, Беларусь)

В статье изучаются моделируемые, но несинтезируемые конструкции языка VHDL для двух широко распространённых синтезаторов логических схем – синтезатора XST, входящего в состав системы Xilinx ISE сквозного проектирования программируемых логических интегральных схем фирмы Xilinx, и синтезатора Leonardo фирмы Mentor Graphics. Статья предназначена для специалистов, занимающихся проектированием цифровых систем и знакомых с основными элементами языка VHDL.

Язык VHDL широко распространился в качестве языка проектирования цифровых систем. Этому способствовали его использование в различных промышленных системах проектирования и стандартизация языковых версий. В настоящее время основным действующим стандартом является стандарт ANSI/IEEE Std 1076-1993, появившийся в 1993 г. и более известный как VHDL'93. Именно этот стандарт поддерживается в настоящее время в основных промышленных системах проектирования. В таких системах можно как проводить моделирование исходных спецификаций, так и синтезировать логические схемы по корректным VHDL-программам. Однако синтез логических схем возможен не для всех конструкций языка VHDL. Конструкции языка, которые поддерживаются при синтезе, образуют *синтезируемое* подмножество языка. Иногда синтезируемое подмножество называют также *синтезабельным* [1] подмножеством. И хотя имеется стандарт IEEE Std 1076.6-1999 на синтезируемость конструкций уровня RTL, при практическом использовании различных синтезаторов следует учитывать следующее важное обстоятельство – различные синтезаторы «понимают» синтезируемое подмножество языка VHDL поразному.

Рассмотрим два достаточно распространённых синтезатора логических схем: синтезатор XST и синтезатор LeonardoSpectrum, далее называемый Leonardo. Синтезатор XST входит в состав системы Xilinx ISE сквозного проектирования программируемых логических интегральных схем (ПЛИС) фирмы Xilinx. Синтезатор Leonardo (фирма Mentor Graphics) позволяет синтезировать как схемы ПЛИС, так и логические схемы в заданной пользователем библиотеке проектирования. Такие схемы входят в состав заказных СБИС (ASIC). Заметим, что Leonardo может быть использован в качестве инструмента синтеза и в системе Xilinx ISE. На практике зачастую приходится использовать оба синтезатора, например, в случае перехода от ПЛИС к заказной СБИС либо наоборот, когда нужно реализовать заказную схему в составе ПЛИС. Если исходные VHDL-программы написаны без учёта использования двух синтезаторов, то в некоторых достаточно «тонких» ситуациях могут проявиться различия в определении синтезируемых конструкций и проект, синтезированный с помощью одного синтезатора, может не синтезироваться другим синтезатором.

Целью данной статьи является изучение и сравнение множеств моделируемых, но несинтезируемых конструкций языка VHDL для XST (версия 8.1i) и для Leonardo (версия 2005a.82). Предполагается, что читатель знаком с основами программирования на VHDL.

Приведём пример моделируемой, но несинтезируемой VHDL-программы:

```
-- VHDL-код, несинтезируемый в
Leonardo и XST

entity example_1_shared is
port(
x1 : in integer;
y1, y2 : out integer);
end example_1_shared;
architecture beh of
example_1_shared is
shared variable COUNT : Integer;
begin
p1: process (x1)
begin
COUNT := 1;
y1 <= x1 + COUNT;
end process;
p2: process (x1)
begin
COUNT := 3;
y2 <= x1 + COUNT;
end process;
end beh;
```

Разделяемая (`shared`) переменная `COUNT` используется в двух процессах `p1`, `p2`, однако при синтезе появляется противоречие, связанное с употреблением этой переменной. Естественно, что если в каждом процессе использовать свою локальную переменную, например, `COUNT1` – для процесса `p1`, `COUNT2` – для процесса `p2`, то описание будет синтезируемым. Так вот, далее в статье будут рассматриваться корректные VHDL-программы (VHDL-конструкции), которые являются моделируемыми, но не синтезируемыми. Будем называть их запрещёнными (для синтеза, но не для моделирования) VHDL-конструкциями. Запрещёнными VHDL-конструкциями в синтезаторах XST и Leonardo могут быть некоторые типы данных и операторы. Запреты на синтез могут быть связаны также с определёнными ограничениями на использование операторов, отсутствие поддержки при синтезе может быть связано также с превышением размерностей данных и т.д.

Перечислим конструкции, которые *не поддерживаются при синтезе* обоими синтезаторами XST и Leonardo. Попутно будем фиксировать найденные экспериментальным путём различия в понимании (опреде-

лении) синтезаторами запрещённых конструкций. Заметим, что большое число примеров синтезируемых VHDL-конструкций для Leonardo можно найти в [2].

Операции над типом real не поддерживаются при синтезе

Отсутствие поддержки понимается следующим образом: если проектировщик употребил в VHDL-коде тип real, то синтезаторы выдадут сообщение об ошибке (тип real нельзя декларировать) и схема не будет построена:

```
-- Несинтезируемый VHDL-код в
Leonardo и XST

entity exp1_real is
port(
b1 : in real range 2.0 to 3.14;
 -- ошибка (тип real не поддер-
живается при синтезе)
b2 : in real range -2.0 to 4.77;
b3 : out real range -4.0 to
8.00);
end exp1_real;
architecture str1 of exp1_real
is
begin
b3 <= b1 + b2;
end str1;
```

Операции над файлами не поддерживаются при синтезе

Данные операции используют ключевые слова VHDL: file (файл), acsess – (доступ).

Распределители (allocators) не поддерживаются при синтезе

Эта конструкция комментариев не требует.

Атрибуты, определённые пользователем

Синтезатор XST не поддерживает такие атрибуты, а синтезатор Leonardo – поддерживает. Пример атрибута two_length, определённого пользователем, приведён в листинге:

```
-- VHDL-код, синтезируемый в
Leonardo
-- VHDL-код, несинтезируемый в
XST

entity attr is
port( a : in bit_vector (0 to
5);
 b : out integer); end;
architecture beh of attr is
attribute two_length : integer;
attribute two_length of a : sig-
nal is (a'length) * 2;
begin
b <= a'two_length;
end;
```

В результате синтеза представленной VHDL-программы Leonardo получает схемную реализацию константы 12 (удвоенная длина массива). Логическая схема состоит из элементов, реализующих логический нуль и логическую единицу.

Сигналы, декларированные в пакете

Синтезатор Leonardo *не поддерживает* такие декларации (пример в листинге ниже), синтезатор XST – *поддерживает*. В листинге приведено структурное описание схемы, внутренний сигнал C1 которой декларирован в пакете pack:

```
-- VHDL-код, несинтезируемый в
Leonardo
-- VHDL-код, синтезируемый в XST

package pack is
signal C1 : bit; -- декларация
сигнала C1
end pack;

package body pack is
end pack;
Library work;
use work.pack.all;
entity adder_2 is
port (a1, b1, a2,b2 : in BIT;
 c2,s2,s1 : out BIT);
end adder_2;
architecture struct_1 of adder_2
is
component
add1 port (b1,b2: in BIT;
 c1,s1: out BIT);
end component;
component add2 port(c1, a1,a2:in
BIT;
 c2,s2:out BIT);
end component;
-- signal c1:BIT := '1'; -- сигнал
декларирован в пакете pack
begin
circ1: add1 port map (b1, b2,
c1, s1);
circ2: add2 port map (c1, a1,
a2, c2, s2);
end struct_1;
```

Заметим, что если внести в описание строку, являющуюся комментарием (в VHDL каждая строка комментария начинается с двух символов «–»), и не использовать пакет pack, то и для Leonardo описание станет синтезируемым. Очевидно, что оба синтезатора потребуют функциональные описания компонент add1, add2 для полноты проекта.

Ключевое слово after игнорируется при синтезе

Это справедливо для обоих синтезаторов. Игнорируется – это значит – пропускается, как будто этого слова вовсе и нет. Поэтому синтезаторы не гарантируют требуемую задержку синтезированной схемы (задержка определяется алгоритмической моделью). Синтезаторы ориентированы на реализацию некоторых функций (алгоритмов), которые соответствуют событиям, связанным между собой причинно-следственными связями в физическом времени, схемная реализация функций может привести к поведению во времени, отличающемуся от поведения во времени алгоритмической модели.

Например, оператор

```
Y <= X1 or X2 or X3 after 30 ns;
```

в процессе синтеза заменяется трёхвходовым дизъюнктором, который имеет задержку, например, 3 нс. Это реальная задержка логического элемента из технологической библиотеки. Таким же логическим элементом будет заменен оператор

```
Y <= X1 or X2 or X3 after 1 ns;
```

Даже в этом простейшем примере видно, что поведение во времени исходной алгоритмической модели будет отличаться от поведения логической схемы.

Ограничения на инициализацию значений

Игнорируются инициальные значения сигналов в разделе декларации сигналов архитектурного тела. Игнорируются инициальные значения портов OUTPUT, INOUT в интерфейсе объекта проекта, т.е. в entity. В операторе процесса игнорируются инициальные значения переменных в разделе деклараций переменных.

Ограничения использования оператора цикла loop

Циклы поддерживаются, если цикловые переменные ограничены константами.

Ограничения на использование оператора wait

Выражение в условии `until` оператора wait должно определять передний или задний фронт. Многократные операторы `wait` допускаются в операторе процесса, однако условия ожидания должны быть одинаковыми [2]. Для XST оператор `wait` не может быть внутри цикла (оператор цикла может быть в процессе, функции, процедуре), для Leonardo – может.

Ограничения на атрибуты выделения фронтов сигналов

Атрибуты `'event`, `'stable` могут быть использованы только для того, чтобы определить передний или задний фронты сигналов. Например:

- `clk'event and clk='1'` (задание переднего фронта сигнала `clk`);
- `clk'event and clk='0'` (задание заднего фронта сигнала `clk`);
- `not clk'stable and clk='0'` (задание заднего фронта сигнала `clk`).

Выражения, задающие фронты сигналов, могут быть использованы только как условия. Например:

- `If (clk'event and clk='1') then` (условие в операторе `if`);
- `Wait until ((not clk'stable) and clk='0')` (условие в операторе wait);
- `Block (clk'event and clk='1')` (охранное выражение в операторе блока).

В условиях (выражениях), где есть синхросигнал `clk`, не допускается употребление других сигналов. Например, конструкция

```
If (clk'event and clk='1' and
rst ='0') then
```

является запрещённой как для Leonardo, так и для XST.

В приведённом ниже примере синтезатор Leonardo синтезирует схему (триггер и логический элемент), а синтезатор XST выдаёт сообщение об ошибке:

```
-- VHDL-код, синтезируемый в
Leonardo
-- VHDL-код, несинтезируемый в
XST
library IEEE;
use IEEE.std_logic_1164.all;
entity event is
 port (CLK, D : in std_logic;
 Q1 : out std_logic;
 Q2 : out boolean);
end;
architecture RTL of event is
begin
 process (CLK)
 begin
 if (CLK'event and CLK = '1')
then -- триггер
 Q1 <= D;
 end if;
Q2 <= (CLK'event and CLK = '0'); --
инвертор
 end process;
end RTL;
```

Если удалить из листинга строку, отмеченную жирным шрифтом, то XST строит схему – D-триггер.

В следующем примере синтезатор XST не синтезирует логическую схему, Leonardo – синтезирует:

```
-- VHDL-код, синтезируемый в
Leonardo
-- VHDL-код, несинтезируемый в
XST

entity while_loop is
port (inp_sig : in bit_vector (3
downto 0);
ena : in bit;
result : out bit_vector (3 down-
to 0));
end while_loop;
architecture example of
while_loop is
begin
process
variable i : integer;
begin
i := 0;
while (i<4) loop
 result (i) <= ena and
inp_sig(i);
i := i + 1;
end loop;
wait until (inp_sig = "1010");
end process;
end example;
```

Всё дело в выделенном жирным шрифтом условии (`inp_sig = "1010"`) ожидания в операторе wait: для синтезатора XST не допускается использования векторов в условиях для `wait`. Если использовать однобитный сигнал, то XST схему синтезирует. В синтезаторе XST оператор wait может использоваться в следующем виде:

```
wait [on clock_signal] until
[clock_signal'event | not
clock_signal'stable) and]
clock_signal= {'0'|'1'};
```

Квадратные скобки означают необязательное выражение, вертикальная черта служит для обозначения альтернативных случаев.

Ограничения для операций {/, rem, mod}

В выражениях

```
c <= x / y; -- операция деления,
c – целая часть частного
d <= x rem y; -- операция деле-
ния, d – остаток
e <= x mod y; -- операция деле-
ния по модулю
```

значение y должно быть $2^N, N = 0,1,2,\ldots$. Например, следующие выражения:

```
c <= a /(-2); --error
d <= b rem (3); --error
e <= x mod (5); --error
```

рассматриваются как ошибки, и схемы не синтезируются.

Ограничения оператора ** (возведение в степень)

Оператор

```
c <= a**(2); --error
```

приводит к ошибке в XST и в Leonardo. В обоих синтезаторах только число 2 можно возводить в степень:

```
c <= 2**(N); -- good
```

где N – целое положительное число типа `integer`.

В Leonardo можно схемно реализовать оператор `c <= 2**(N);`, если $0 \le N \le 30$. Чтобы работать в Leonardo с большей размерностью N типа `integer`, требуется подключение пакетов, например пакета `NUMERIC_STD`. Все описанные в пакете `NUMERIC_STD` функции поддерживаются при синтезе, кроме функций `{/, rem, mod}`, которые синтезируются только в случаях, когда делитель равен 2^N.

В отличие от Leonardo, синтезатор XST реализует оператор:

```
c <= 2**(N);
```

для $N > 30$.

Охраняемый блок

Пример показывает различия в синтезируемых конструкциях Leonardo и XST: в синтезаторе Leonardo схема синтезируется, в XST выдаётся сообщение, что охраняемые сигналы не поддерживаются при синтезе, и схема не строится:

```
-- VHDL-код, синтезируемый в
Leonardo
-- VHDL-код, несинтезируемый в
XST

entity guard is
port(
x1, x2, x3, x4, x5, x6 : in bit;
v, w : out bit);
end guard;
architecture str of guard is
begin
p1: block ((x5 and x6 )='1')
begin
v <= guarded (x1 or x2);
w <= guarded (x3 xor x4);
end block;
end str;
```

Назначение сигнала

В XST поддерживаются операторы назначения сигнала с ключевым словом `transport` и не поддерживаются со словом `guarded` (охраняемый). В Leonardo таких ограничений нет.

Сигналы типа register, bus

В XST *не поддерживаются* типы `register`, `bus`, а в Leonardo поддерживаются. Если декларировать сигналы такого типа, то синтезатор XST выдаёт следующее сообщение: «`Bus or register unsupported in signal declaration.`»

Ключевое слово disconnect не поддерживается

В таблицу сведены результаты сравнения синтезаторов: знак «–» соответствует запрещённой для синтеза конструкции; знак «+» соответствует синтезируемой конструкции; знак «#» указывает на различное понимание синтезируемости конструкции в сравниваемых синтезаторах. В общем оказывается, что Leonardo имеет меньше запрещённых VHDL-конструкций, исключение составляет только операция возведения в степень 2^N, для которой XST не имеет ограничения $N \le 30$. Поэтому проекты, синтезированные в XST, имеют шанс быть синтезированными в Leonardo без дополнительных изменений, но не наоборот. Однако это не абсолютный вывод, опытные проектировщики, возможно, встречались и с другими ситуациями несовместимости синтезаторов. Синтезаторы быстро совершенствуются, и для следующих версий ситуация может быть уже другой.

Результаты сравнения синтезаторов

VHDL-конструкция	Синтезатор XST	Синтезатор Leonardo
real	–	–
file, accsess	–	–
allocator	–	–
disconnect	–	–
guarded	–	+
register	–	+
bus	–	+
linkage	–	+
group	–	+
wait	#	#
x/a; x rem a; x mod a; (a = 2^N, N = 0,1, 2, …)	+	+
2**N	#	#
a**(N)	–	–
Декларация сигнала в пакете	+	–
Атрибут, определённый пользователем	–	+
after	Игнорируется	Игнорируется

Для получения синтезируемых проектов, повышения их качества и возможности использования обоих синтезаторов – XST и Leonardo – можно рекомендовать следующее.

- Не используйте в VHDL-проектах задание времени в «явном» виде, т.е. следует избегать конструкций вида `wait for`;
- Переключайте состояния проектируемых систем по изменениям фронтов и уровней сигналов;
- Не пользуйтесь конструкциями, которые не синтезируются (различаются в понимании) в обоих синтезаторах. Пользуйтесь теми конструкциями, которые понимаются одинаково в обоих синтезаторах, тем более что таких конструкций подавляющее большинство;
- Для унификации проектов пользуйтесь пакетами из библиотеки IEEE;
- Особенно осторожно следует обращаться с оператором `wait` ожидания и выражениями для условий ожидания. Именно эти конструкции приводят к последовательностным схемам, т.е. схемам с триггерами, и именно здесь есть важные отличия XST и Leonardo. В выражениях с сигналом `clk` должен участвовать только один сигнал `clk`, а не несколько сигналов, – это довольно типичная ошибка;
- Для описания комбинационной логики не используйте охраняемые сигналы и блоки – они ведут к триггерам;
- Чем ближе стиль VHDL-программ к стилю *data flow* [1, 2], тем лучшие результаты синтеза в обоих синтезаторах – имеется в виду сложность и быстродействие логических схем;
- Совместимость синтезаторов XST и Leonardo может быть обеспечена на уровне RTL-описаний, например, проект синтезируется Leonardo, затем с помощью команды `unmap` осуществляется переход к RTL-описанию, которое воспринимается синтезатором XST. Напомним, что RTL-описания, получаемые Leonardo после unmap, содержат только логические операторы и операторы `port map` структурного описания.

Литература

1. *Поляков А.К.* Языки VHDL и VERILOG в проектировании цифровой аппаратуры. М.: СОЛОН-Пресс, 2003.
2. *Бибило П.Н.* Синтез логических схем с использованием языка VHDL. М.: Солон-Р, 2002.