**ИНФОРМАТИКА**

**2004** **январь-март** **№ 1**

*ЛОГИЧЕСКОЕ ПРОЕКТИРОВАНИЕ*

*УДК 519.7*

**П.Н. Бибило**

# ОПИСАНИЕ ПАРАЛЛЕЛЬНЫХ И СЕКВЕНЦИАЛЬНЫХ АВТОМАТОВ НА ЯЗЫКЕ VHDL

*Предлагаются синтезируемые VHDL-модели параллельных автоматов, представленных на языке ПРАЛУ, и секвенциальных автоматов. Синтезируемыми являются такие VHDL-описания, по которым возможно автоматическое построение логических схем в заданных базисах логических элементов.*

## Введение

Для описания поведения цифровых систем разработан ряд моделей, одной из которых является модель параллельного автомата [1]. Достоинством данной модели является удобство исходного описания поведения цифровой системы и простота схемной реализации в виде программируемой логической матрицы (ПЛМ) с памятью в виде регистра RS-триггеров. Модель секвенциального автомата является промежуточной при схемной реализации параллельных автоматов, представленных на языке ПРАЛУ. Язык ПРАЛУ предназначен для описания параллельных алгоритмов логического управления. Его характерными особенностями являются логическая стройность, простота, компактность получаемых описаний, использование двоичных (булевых) переменных для входных и выходных переменных устройства управления, алгоритм функционирования которого задан на языке ПРАЛУ. Полное описание языка ПРАЛУ содержится в работе [1]. Язык VHDL является международным стандартом для систем автоматизации проектирования и предназначен для спецификации, моделирования и синтеза [2] цифровых систем на основе заказных и программируемых пользователями сверхбольших интегральных схем (СБИС). Переход от моделей параллельного и секвенциального автоматов к VHDL-моделям представляет практический интерес. В настоящей работе предлагаются VHDL-модели символьных ПРАЛУ-описаний параллельных автоматов и матричных описаний секвенциальных автоматов. Предлагаемые алгоритмические VHDL-модели автоматов являются синтезируемыми для системы синтеза (синтезатора) LeonardoSpectrum [2]. Данный синтезатор позволяет проводить синтез схем, реализуемых как на программируемых логических схемах типа FPGA (Field-Programmable Gate Arrays) и CPLD (Complex Programmable Logic Devices), так и в составе заказных СБИС. Синтезируемыми являются такие VHDL-описания, по которым возможно автоматическое построение логических схем в заданных технологических базисах.

## 1. VHDL-модель параллельного автомата

Модель параллельного автомата позволяет учитывать параллелизм логического управления. В работе [1] для описания функционирования параллельного автомата используется язык ПРАЛУ. Будем считать исходным ПРАЛУ-описание параллельного автомата, состоящее из *элементарных цепочек*. Согласно [1] *элементарной* называется цепочка вида

$$\mu_i : -k_i' \rightarrow k_i'' \rightarrow \nu_I , \qquad (1)$$

в которой операция $-k_i'$ или $\rightarrow k_i''$ (или обе вместе) может отсутствовать. В общем случае элементарная цепочка состоит из четырех частей:

$\mu_i$ – множество начальных меток цепочки;

$-k_i'$ – операция ожидания события $k_i'$ ;

$\rightarrow k_i''$ – операция действия;

$\nu_i$ – множество заключительных меток цепочки.

В формуле (1) двоеточие служит разделителем, а стрелка перед $\nu_i$ играет роль операции внесения элементов в текущее множество запуска цепочек. Множество M, называемое *множеством запуска* цепочек, получается в результате объединения множеств начальных и заключительных меток всех цепочек. Сначала поясним, что представляют собой операции ожидания и действия, а затем – каким образом выполняется алгоритм в целом, т. е. как выполняются цепочки, как они взаимодействуют друг с другом и какова роль в этом множества M.

В записи (1) $k_i'$, $k_i''$ представляют собой элементарные конъюнкции булевых переменных. Конъюнкции $k_i'$ образуются из литералов булевых переменных множества X, конъюнкции $k_i''$ образуются из литералов переменных множества Y. Если конъюнкция $k_i'$ ($k_i''$) отсутствует в (1), то предполагается, что она тождественно равна 1. Операция $-k_i'$ представляет собой операцию ожидания события $k_i'$. Выполнение этой операции сводится к ожиданию события, когда переменные, входящие в конъюнкцию $k_i'$, примут значения, обращающие $k_i'$ в единицу. Операция действия $\rightarrow k_i''$ означает присвоение таких значений переменным конъюнкции $k_i''$, при которых $k_i''$ обращается в единицу.

Цепочка (1) срабатывает, если в текущем множестве запуска содержится множество $\mu_i$ и если выполнилась операция ожидания $-k_i'$. Срабатывание цепочки заключается в немедленном удалении множества $\mu_i$ из текущего множества запуска, после чего выполняется операция действия $\rightarrow k_i''$, а затем немедленно добавляются элементы множества $\nu_i$ в текущее множество запуска. В начале работы алгоритма в текущее множество запуска включается метка 1, алгоритм заканчивает работу, если множество запуска содержит заключительную метку. В процессе работы алгоритма некоторые цепочки могут выполняться одновременно (параллельно), поэтому такой формализм позволяет описывать параллельные алгоритмы логического управления. В работе [1] показано, что множество элементарных цепочек представляет собой параллельный автомат. При этом множество таких цепочек должно удовлетворять определенным требованиям, например, цепочки $i$, $j$, имеющие одинаковые множества начальных меток, должны иметь ортогональные конъюнкции в операциях ожидания. Другие требования к корректности исходных ПРАЛУ-описаний изложены в [1]. Будем предполагать, что исходное ПРАЛУ-описание является корректным.

Рассмотрим вопрос о преобразовании ПРАЛУ-описаний в VHDL-коды. Естественно, сначала нужно, если потребуется, изменить идентификаторы языка ПРАЛУ в соответствии с требованиями языка VHDL. После этого преобразование ПРАЛУ-описаний в VHDL-коды может быть полностью автоматизировано. Покажем, как это может быть сделано. Рассмотрим пример ПРАЛУ-описания [3], в котором X = { $x_1, x_2$ }, Y = { $y_1, y_2$ }:

$$
\begin{aligned}
&1: -x_1x_2 \rightarrow y_1\bar{y}_2 \rightarrow 10\\
&10: -\bar{x}_2 \rightarrow 2.3.4\\
&3.5: -x_2 \rightarrow 8\\
&4: -\bar{x}_1 \rightarrow \bar{y}_1 \rightarrow 7\\
&4: -x_1 \rightarrow y_2 \rightarrow 9\\
&7: -\bar{x}_2 \rightarrow 9\\
&6.8.9: \rightarrow y_2 \rightarrow 11\\
&11: -x_1 \rightarrow 1\\
&2: \rightarrow \bar{y}_1 \rightarrow 5.6
\end{aligned}
$$

Предлагается представлять VHDL-описание параллельного автомата в виде двух взаимодействующих процессов (оператор `process`). В первом процессе (`process p1`) в текущем такте $t_i$ определяются элементарные цепочки, которые работают параллельно, и следующие

(для следующего такта $t_{i+1}$ дискретного времени) значения переменных $y_j$ действия и значения переменных множества запуска цепочек. Подготавливаемые значения переменных действия обозначаются через $n\_y_j$, а подготавливаемые значения переменных множества запуска – через $n\_z_i$. Предполагается, что каждая из цепочек срабатывает за один такт дискретного времени.

Во втором процессе (`process p2`) осуществляется смена такта по переднему фронту сигнала синхронизации CLK. Для этого используется оператор `if (CLK='1' AND CLK'event) THEN` нахождения переднего фронта сигнала CLK. Установка автомата в начальное состояние (переменные действия имеют значение 0, а в множество запуска включается только переменная `z1`) осуществляется по единичному значению сигнала `rst` разрешения, который имеет более высокий приоритет по отношению к другим входным сигналам. На момент смены такта значения сигналов, соответствующих переменным ожидания, должны быть установившимися. Если в текущем такте ни одна из элементарных цепочек не срабатывает, то множество запуска цепочек для следующего такта не изменяется.

***Листинг 1*. VHDL-модель параллельного автомата PA**

```
LIBRARY work;
USE work.wire.all;

entity PA is
port (clk, rst, x1, x2 : in bit;
      y1, y2 : out RESOLVED_BIT);
end;
ARCHITECTURE BEHAVIOR OF PA IS
signal z1, z2, z3, z4, z5, z6, z7, z8, z9, z10, z11 :  RESOLVED_BIT;
signal n_z1, n_z2, n_z3, n_z4, n_z5, n_z6,
 n_z7, n_z8, n_z9, n_z10, n_z11 :  RESOLVED_BIT;
signal n_y1, n_y2 :  RESOLVED_BIT;

begin

p1 : PROCESS (z1, z2, z3, z4, z5, z6, z7, z8, z9, z10,z11, x1, x2)
begin
 n_z1 <= z1; n_z2 <= z2; n_z3 <= z3;
 n_z4 <= z4; n_z5 <= z5; n_z6 <= z6; n_z7 <= z7;
 n_z8 <= z8; n_z9 <= z9; n_z10 <= z10; n_z11 <= z11;
 n_y1 <= '0'; n_y2 <= '0';

if  ( (z1 and not x1 and x2)= '1')  then                  -- line 1
n_z1<= '0';n_y1<= '1'; n_y2<= '0'; n_z10 <= '1';
end if;
if ((z10 and not x2) = '1')  then                         -- line 2
n_z10 <=  '0'; n_z2 <=  '1'; n_z3 <= '1'; n_z4 <=  '1';
end if;
if ((z3 and z5 and x2) = '1')  then                       -- line 3
n_z3 <=  '0'; n_z5 <=  '0'; n_z8 <= '1';
end if;
if ((z4 and  not  x1) = '1') then                         -- line 4
n_z4 <= '0'; n_y1 <= '0'; n_z7 <= '1';
end if;
if  ((z4 and  x1) = '1') then                             -- line 5
n_z4 <= '0'; n_y2 <= '1'; n_z9 <= '1';
end if;
if  ((z7 and not x2)= '1') then                           -- line 6
n_z7 <= '0'; n_z9 <= '1';
end if;
```

```
if  ((z6 and z8 and z9) = '1') then                          -- line 7
n_z6 <= '0'; n_z8 <= '0'; n_z9 <= '0'; n_y2 <= '0'; n_z11 <= '1';
end if;
if  ((z11 and x1) = '1')  then                               -- line 8
n_z11 <= '0'; n_z1 <= '1';
end if;
if (z2 = '1')  then                                          -- line 9
n_z2 <= '0' ; n_y1 <= '0'; n_z5 <= '1'; n_z6 <= '1';
end if;
END PROCESS p1;
p2: PROCESS (CLK, rst)
      BEGIN
if (rst = '1') then
y1 <= '0'; y2 <= '0';                                -- initial state
z1 <= '1'; z2 <= '0'; z3 <= '0';
z4 <= '0'; z5 <= '0'; z6 <= '0'; z7 <= '0';
z8 <= '0'; z9 <= '0'; z10 <= '0'; z11 <= '0';
      elsif (rst ='0')  then
         if (CLK='1' AND CLK'event) THEN
            if (n_z1 or z2 or n_z3 or n_z4 or n_z5 or n_z6
                 or n_z7 or n_z8 or n_z9 or n_z10 or n_z11) = '0'
                 then  null;
              else
                y1 <= n_y1;  y2 <= n_y2;
 z1 <= n_z1; z2 <= n_z2; z3 <= n_z3; z4 <= n_z4; z5 <= n_z5;z6 <= n_z6;
 z7 <= n_z7; z8 <= n_z8; z9 <= n_z9; z10 <= n_z10; z11 <= n_z11;
            end if;
         end if;
end if;
END PROCESS p2;
END BEHAVIOR;
```

В листинге 1 комментарии начинаются с двух дефисов и продолжаются до конца строки. Так как выходные и внутренние сигналы автомата РА назначаются из различных источников, то введен тип `resolved_bit`, который определяется через соответствующую разрешающую функцию, помещенную в пакет `wire` (листинг 2).

***Листинг 2*. Пакет `wire` с разрешающей функцией `RES_FUNC`**

```
package wire is
function RES_FUNC(DATA: in bit_vector) return bit;
subtype RESOLVED_BIT is RES_FUNC bit;
end;
package body wire is
  function RES_FUNC(DATA: in bit_vector) return bit is
begin
       for I in DATA'range loop
            if DATA(I) = '1' then
  return '1';
            end if;
       end loop;
  return '0';
  end;
end;
```

### 2. VHDL-модель секвенциального автомата

Секвенциальный автомат является моделью цифровой системы, функционирующей в дискретном времени, и состоит из множества $S$ секвенций $s_i$. Каждая секвенция $s_i$ имеет форму

$f_i \vdash k_i$, где $f_i$ – булева функция входных и внутренних переменных, $k_i$ – элементарная конъюнкция внутренних и выходных переменных. Каждая секвенция $f_i \vdash k_i$ описывает определенное требование к поведению цифровой системы: если в некоторый момент времени $f_i$ принимает значение 1, то непосредственно вслед за этим (в следующем такте дискретного времени) $k_i$ также принимает значение 1. При этом значения всех переменных в $k_i$ определяются однозначно.

Рассмотрим простой секвенциальный автомат SEKV, полученный из параллельного автомата PA с помощью программ системы [4]. Автомат SEKV задается девятью секвенциями:

$$\bar{x}_1 x_2 z_0 z_1 \vdash \bar{z}_0 \bar{z}_4 y_1 \bar{y}_2,$$
$$\bar{x}_2 \bar{z}_0 z_1 z_4 \vdash \bar{z}_1 \bar{z}_2,$$
$$x_2 z_0 \bar{z}_1 \bar{z}_2 \vdash z_2,$$
$$\bar{x}_1 \bar{z}_1 \bar{z}_4 \vdash \bar{z}_3 z_4 \bar{y}_1,$$
$$x_1 \bar{z}_1 \bar{z}_4 \vdash z_3 z_4 y_2.$$
$$\bar{x}_2 \bar{z}_1 \bar{z}_3 z_4 \vdash z_3,$$
$$z_0 \bar{z}_1 z_2 z_3 z_4 \vdash \bar{z}_0 z_1 \bar{y}_2,$$
$$x_1 \bar{z}_0 z_1 z_4 \vdash z_0,$$
$$\bar{z}_0 \bar{z}_1 \vdash z_0 \bar{z}_2 \bar{y}_2.$$

Переменные $x_1$, $x_2$ являются входными; $z_0$, $z_1$, $z_2$, $z_3$, $z_4$ – внутренними; $y_1$, $y_2$ – выходными для автомата SEKV. Секвенциальный автомат является простым, так как функции $f_i$ представляются элементарными конъюнкциями входных и внутренних переменных. Будем интерпретировать данную систему секвенций как *инерционный* секвенциальный автомат. В инерционном секвенциальном автомате предполагается следующее [5]: если во всех элементарных конъюнкциях $k_i$ , соответствующих функциям $f_i$, принимающим в текущий момент времени значение 1, отсутствует символ некоторой внутренней переменной, то считается, что эта переменная сохраняет свое значение, а если отсутствует символ выходной переменной, то считается, что она принимает значение 0. В системах автоматизированного проектирования секвенциальный автомат представляется парой троичных матриц с одинаковым числом строк: первая троичная матрица задает левые части секвенций, вторая – правые части секвенций. В листинге 3 приведено матричное задание секвенциального автомата в системе LOCON.

***Листинг 3*. Модель секвенциального автомата SEKV в системе LOCON**

```
TITLE SEKV
FORMAT  SA
AUTHOR Bibilo
DATE 06/10/04
PROJECT LOCON_VHDL
DCL_PIN
EXT
INP
x1 x2
OUT
y1 y2
INTER
Z0 Z1 Z2 Z3 Z4
END_PIN
FUNCTION
```

```
0111---    0---010
-001--0    -00----
-1100--    --1----
0--0--0    ---010-
1--0--0    ---11-1
-0-0-01    ---1---
--10111    01----0
1-01--1    1------
--00---    1-0---0
END_FUNCTION
END_SEKV
```

Предлагается использовать стиль *data flow* (поток данных) для VHDL-модели простого инерционного автомата, заданного парой троичных матриц. Стиль *data flow* характеризуется тем, что используются операторы процессов, назначения сигналов, логические операторы. Значения сигналов определяются из выражений и передаются один другому операторами назначения сигналов – образуется поток данных. Такой стиль описаний наиболее целесообразно использовать в качестве исходных описаний, по которым автоматически синтезируются логические схемы. Заметим, что модель параллельного автомата также представляется в стиле *data flow*.

В предлагаемой VHDL-модели секвенциального автомата используются два взаимосвязанных процесса (оператор `process`). В первом процессе в текущем такте $t_i$ проводится анализ входных и внутренних переменных секвенций, выбираются «срабатывающие» секвенции и формируются будущие значения внутренних и выходных переменных, т. е. значения внутренних и выходных переменных для следующего такта $t_{i+1}$ дискретного времени. Второй процесс устанавливает подготовленные будущие значения в качестве текущих значений в такте $t_{i+1}$. Смена такта осуществляется по переднему фронту сигнала синхронизации, который так же, как и сигнал установки автомата в начальное состояние, неявно присутствует как в модели параллельного, так и в модели секвенциального автоматов.

***Листинг 4*. VHDL-модель секвенциального автомата SEKV**

```
LIBRARY work;
USE work.wire.all;

ENTITY SEKV IS
      PORT (CLK, rst, x1, x2 : in bit;
            y1, y2 : out RESOLVED_BIT );
END;

ARCHITECTURE BEHAVIOR OF SEKV IS
SIGNAL  z0, z1, z2, z3, z4: RESOLVED_BIT := '1';
SIGNAL  n_z0, n_z1, n_z2, n_z3, n_z4 : RESOLVED_BIT;
SIGNAL n_y1, n_y2 : resolved_bit ;

BEGIN
p1: PROCESS (x1, x2, z0, z1, z2, z3, z4)
      BEGIN
n_z0 <= z0;  n_z1 <= z1; n_z2 <= z2; n_z3 <= z3; n_z4 <= z4;
n_y1 <= '0'; n_y2 <= '0';

if ( not x1  and x2 and  z0 and  z1) = '1' then                -- line 1
n_z0 <= '0'; n_z4 <= '0'; n_y1 <= '1'; n_y2 <= '0';
end if;
if (not x2 and not z0 and  z1 and not z4)='1' then             -- line 2
n_z1 <= '0';n_z2 <= '0';
end if;
```

```
if (x2 and z0 and  not z1 and not z2) ='1' then                    -- line 3
n_z2 <= '1';
end if;
if (not x1 and not z1 and not z4) = '1' then                       -- line 4
n_z3 <= '0'; n_z4 <= '1'; n_y1 <= '0';
end if;
if (x1 and not z1 and not z4 )='1'  then                           -- line 5
n_z3 <= '1'; n_z4 <= '1'; n_y2 <= '1';
end if;
if (not x2 and  not z1 and not z3  and z4) = '1' then              -- line 6
n_z3 <= '1';
end if;
if (z0 and not z1 and z2 and z3 and  z4 ) = '1' then               -- line 7
n_z0 <= '0'; n_z1 <= '1'; n_y2 <= '0';
end if;
if (x1 and not z0 and  z1 and z4) = '1' then                       -- line 8
n_z0 <= '1';
end if;
if (not z0 and not z1) = '1' then                                  -- line 9
n_z0 <= '1'; n_z2 <= '0';  n_y1 <= '0';
end if;
END PROCESS p1;
p2: PROCESS (CLK, rst)
      BEGIN
if (rst = '1') then
y1 <= '0'; y2 <= '0';                                    -- initial state
z0 <= '1';  z1 <= '1'; z2 <= '1'; z3 <= '1'; z4 <= '1';
      elsif (rst ='0' ) then
            if ( CLK='1' AND CLK'event) THEN
  y1 <= n_y1; y2 <= n_y2;
  z0 <= n_z0; z1 <= n_z1; z2 <= n_z2; z3 <= n_z3; z4 <= n_z4;
            end if;
end if;
END PROCESS p2;
END BEHAVIOR;
```

Как и в VHDL-модели параллельного автомата PA, в VHDL-модели секвенциального автомата SEKV используется тот же оператор нахождения переднего фронта сигнала синхронизации CLK. Установка автомата в начальное состояние (значения всех выходных равны нулю, внутренних переменных – единице) осуществляется по единичному значению сигнала rst разрешения, который имеет более высокий приоритет по отношению к другим входным сигналам. Во время текущего такта входные сигналы могут изменяться (это не вызывает изменения выходных сигналов в текущем такте), однако на момент смены такта входные сигналы должны быть установившимися.

По VHDL-моделям параллельного (PA) и секвенциального (SEKV) автоматов были автоматически построены с помощью синтезатора LeonardoSpectrum логические схемы в базисе логических элементов [2, с. 159], входящих в библиотеку проектирования СБИС на основе базовых матричных кристаллов. Согласно методике, представленной в работе [5], была построена логическая схема ПЛМ с элементами памяти в виде RS-триггеров, реализующая рассматриваемый секвенциальный автомат. Модель данной схемы была написана на языке VHDL. Моделирование показало эквивалентность поведения во времени всех моделей: исходных алгоритмических моделей параллельного и секвенциального автоматов и трех структурных моделей логических схем.

**Заключение**

Предложенные VHDL-модели ПРАЛУ-описаний параллельных автоматов и секвенциальных автоматов могут быть построены программно из соответствующих моделей автоматов,

используемых в системе LOCON [4], и предназначены для создания интегрированной среды проектирования LeonardoSpectrum–LOCON.

Исследования проведены при частичной поддержке Международного научно-технического центра (проект В-986).

**Список литературы**

1. Закревский А.Д. Параллельные алгоритмы логического управления. – Мн.: Ин-т техн. кибернетики НАН Беларуси, 1999. – 202 с.

2. Бибило П.Н. Синтез логических схем с использованием языка VHDL. – М.: Солон-Р, 2002. – 384 с.

3. Ковалев А.В., Поттосин Ю.В. К декомпозиции параллельных алгоритмов логического управления // Автоматика и вычислительная техника. – 1988. – № 1. – С. 8–13.

4. Романов В.И. Алгоритмическое проектирование в системе LOCON // Логическое проектирование: сб. науч. тр. Вып. 5. – Мн.: Ин-т техн. кибернетики НАН Беларуси, 2000. – С. 137–146.

5. Закревский А.Д. Логический синтез каскадных схем. – М.: Наука, 1981. – 416 с.

**Поступила 17.01.05**

*Объединенный институт проблем*
*информатики НАН Беларуси,*
*Минск, Сурганова, 6*
*e-mail: bibilo@newman.bas-net.by*

**P.N. Bibilo**

## VHDL-DESCRIPTIONS OF PARALLEL AND SEQUENT AUTOMATA

The synthesized VHDL descriptions of parallel and sequent automata are developed. VHDL description is considered as synthesized if LeonardoSpectrum synthesizer could build a logic circuit in the given basis of logic gates according to this description.