II Konferencja Naukowa KNWS'05
"Informatyka- sztuka czy rzemiosło"
15-18 czerwca 2005, Złotniki Lubańskie

THE METHODS FOR TRANSFORMATION OF PARALLEL AND SEQUENT AUTOMATA INTO VHDL-DESCRIPTIONS

Pyotr Bibilo

United Institute of Informatics Problems, National Academy of Sciences of Belarus
Surganov str., 6. 220012 Minsk, Belarus

e-mail: bibilo@newman.bas-net.by

SUMMARY

The developed methods for transformation of parallel and sequent automata allow one to obtain VHDL descriptions in the synthesized subset of the language that LeonardoSpectrum synthesizer works with.

1. INTRODUCTION

To describe the behavior of digital systems a number of languages (models) are developed. One of such models is the parallel automaton [1]. The advantage of this model is convenience of the initial description of behavior of a digital system and simplicity of its circuit implementation in the form of programmable logical array (PLA) with the memory as a RS-flip-flop register. The model of sequent automaton is intermediate one when a parallel automaton described in PRALU language is implemented in a circuit. The PRALU language is intended to describe parallel algorithms of logical control. The logical “orderliness”, simplicity, compactness of descriptions, binary (Boolean) variables using as input and output ones of the control device whose algorithm is given in the language are characteristic features of PRALU language. The whole description of PRALU language is in [1]. VHDL is an international standard in CAD systems and intended for specification, simulation and synthesis of digital systems based on very large-scale integrated (VLSI) circuits that are custom made and programmable by the user [2]. The transition from the models of parallel and sequent automata to VHDL models is of practical interest. In this paper, methods for transforming symbolic PRALU descriptions of parallel automata and matrix descriptions of sequent automata into algorithmic synthesized VHDL descriptions for LeonardoSpectrum synthesizer [2] are suggested. This synthesizer allows one to synthesize circuits implemented both in programmable logic circuits of FPGA (Field-Programmable Gate Arrays) or CPLD (Complex Programmable Logic Devices) kind and as parts of custom made VLSI circuits.

VHDL description is synthesized if automatic construction of logic circuits in given technological bases is possible.

2. PARALLEL AUTOMATON

Using in the practice of designing digital systems the languages and formal models allowing one to describe parallelism and asynchronous working of logical control is widening. One of such models is model of *parallel automaton*. In [1] PRALU language is used to describe the functioning of a parallel automaton. We take PRALU description of parallel automaton consisting of elementary chains as the initial one. According to [1] we call *elementary* a chain of the following form:

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

where operation $-k_i'$ or $\rightarrow k_i''$ or both may be absent. In general case, an elementary chain consists of four parts:

μ_i is the set of the initial marks of the chain;
$-k_i'$ is the operation of waiting of event k_i';
$\rightarrow k_i''$ is the act operation;
ν_i is the set of the final marks of the chain.

The colon in formula (1) is a spacer, and the arrow before ν_i is the symbol of the operation of inserting elements to the current chain start set. Set M called *chain start set* is obtained by union of sets of initial and final marks of all chains. First, we explain what are the wait and act operations. Then, we explain how an algorithm is executed as a whole, i.e. how chains are executed, how they interact, and what is their place in set M.

In formula (1), k_i' and k_i'' are elementary conjunctions of Boolean variables. Conjunctions k_i' are constituted of letters of Boolean variables from set X, and conjunctions k_i'' are constituted of letters of Boolean variables from set Y. If conjunction k_i' (k_i'') is absent in (1), it is supposed to be equal to 1 identically. Operation $-k_i'$ is a wait operation that waits for event k_i', that means waiting for the event when the variables in conjunction k_i' take values that reduce k_i' to 1. Act operation $\rightarrow k_i''$ means assigning the values to the variables of conjunction k_i'' that reduce it to 1.

The chain (1) starts if set μ_i is in the current start set and wait operation $-k_i'$ is fulfilled. The starting of the chain consists of immediate removal of set μ_i from the current start set that is followed by fulfilling the act operation $\rightarrow k_i''$. Then, the elements of set ν_i are added to the current start set immediately. At the beginning of the algorithm execution, the mark 1 is introduced into the current start set. The algorithm execution ends when the start set contain the final mark. When the algorithm is executing some chains can be fulfilled simultaneously

(in parallel), therefore such a formalism allows one to describe parallel logical control algorithms. A set of elementary chains is shown in [1] to be a parallel automaton. At that, a set of such chains must satisfy certain requirements, e.g. chains *i* and *j* that have the same set of initial marks must have orthogonal conjunctions in the wait operations. Other requirements to correctness of initial PRALU descriptions are in [1]. Assume the initial PRALU description to be correct.

Let us consider the problem of transforming PRALU descriptions into VHDL codes. It is natural that first, if necessary, one must change the identifiers of PRALU language according to the requirements of VHDL. Then, the transformation of PRALU descriptions into VHDL codes can be automated completely. Now we show how can it be done. Let us consider the example of PRALU description from [3] where $X = \{ x_1, x_2 \}$, $Y = \{ y_1, y_2 \}$:

$1: -x_1 x_2 \to y_1 \bar{y}_2 \to 10$
$10: -\bar{x}_2 \to 2.3.4$
$3.5: -x_2 \to 8$
$4: -\bar{x}_1 \to \bar{y}_1 \to 7$
$4: -x_1 \to y_2 \to 9$
$7: -\bar{x}_2 \to 9$
$6.8.9: \to y_2 \to 11$
$11: -x_1 \to 1$
$2: \to \bar{y}_1 \to 5.6$

We suggest to represent VHDL description of a parallel automaton (Listing 1) as two interacting processes (operator `process`). The first process (`process p1`) defines in the current time step t_i the elementary chains that work in parallel and defines the next (for the next time step t_{i+1} of the discrete time) values of variables y_j and the values of variables of the chain start set. The prepared values of the act variables are denoted as n_y_j and the prepared values of the variables of the start set as n_z_i. Every chain is assumed to be fulfilled during one time step of discrete time (if it can do it).

The second process (`process p2`) changes a time step by the front of the synchronization signal, CLK. To do it operator `if (CLK='1' and CLK'event) THEN` of finding the front CLK. The automaton is set to the initial state (the act variables are equal to 0 and the start set has only variable `z1`) by value 1 of signal `rst` of permission that has the higher priority in comparison with other input signals. At the moment of changing the time step, the signal values that correspond to the wait variables must be established. If no elementary chain starts in the current time step, the chain start set does not change for the next time step.

Listing 1. VHDL model of parallel automaton, PA.

```
LIBRARY work;
USE work.wire.all;
entity PA is
port (clk, rst, x1, x2 : in bit;
   y1, y2 : inout 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;
```

The commentaries in Listing 1 begin with two hyphens and continue up to the end of a line. As the output signals (and inner ones) of automaton PA are assigned from different sources, type `resolved_bit` is introduced that is defined by the corresponding resolution function that is placed in package `wire` (Listing 2). As the values of signals `y1` and `y2` transmitted in the architecture body (process `p1)` to inner signals `n_y1`, `n_y2`, the mode `inout` is used for signals `y1` and `y2`.

Listing 2. Package wire with resolution function 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;
```

3. SEQUENT AUTOMATON

The sequent automaton is a model of a digital system functioning in discrete time and consist of set S of sequents s_i. Every sequent s_i is in form $f_i \vdash k_i$ where f_i is Boolean function of input and inner variables, k_i is an elementary conjunction of inner and output variables. Every sequent f_i |- k_i describes a certain requirement to the behavior of the digital system: if f_i takes value 1 at some moment, k_i takes 1 as well directly after it (at the next time step of discrete time). At that, the values of all variables of k_i are defined unambiguously.

Let us consider the simple sequent automaton obtained from a parallel automaton by the programs of the system from [4]. It is given by nine sequents:

$\bar{x}_1 x_2 z_0 z_1 \vdash \bar{z}_0 \bar{z}_4 y_1 \bar{y}_2$,

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

Variables x_1 and x_2 are input ones, z_0, z_1, z_2, z_3, z_4 inner and y_1, y_2 output for the considered automaton. The sequent automaton is simple one as the functions f_i are represented by elementary conjunctions of input and output variables. We interpret the given sequent system as inertial sequent automaton and admit the following [5]: if the symbol of some inner variable is absent in all elementary conjunctions k_i corresponding to functions f_i that are equal to 1 at a current moment, then this variable is considered to keep its value; if the symbol of some output variable is absent, the variable is equal to 0. In CAD systems, a sequent automaton is given by a pair of ternary matrices with the same number of rows. One of the matrices gives the left parts of the sequents and the other matrix the right parts.

A method is suggested to transform a simple inertial automaton given by a pair of ternary matrices into a VHDL description represented in *data flow* style. Two interconnected processes (operator `process`) are used in the suggested VHDL model of sequent automaton. One of the processes analyzes input and inner variables of the sequents at the current time step t_i, selects the starting sequents and forms the next values of inner and output variables for the next time step t_{i+1} of discrete time. The other process sets the prepared next values as the current values at time step t_{i+1}. The time steps change at the front of the synchronization signal that as well as the signal setting the automaton into initial state is present implicitly both in the model of parallel automaton and in the model of sequent automaton.

Listing 3. VHDL model of sequent automaton, 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;
```

In VHDL model of sequent automaton, SEKV, as in VHDL model of parallel automaton, PA, the same operator finding the front of the synchronization signal, CLK, is used. The automaton is set to the initial set (the values of all input and output variables are equal to 0) by value 1 of signal `rst` of permission that has the higher priority in comparison with other input signals. During the current time step, the input signals may change (it does not cause the change of output signals at the current time step) but the input signals must be set at the moment of changing a time step.

Using the VHDL models of parallel and sequent automata, logic circuits in the basis of the logic elements [2, page 159] from library of VLSI design based on the base matrix crystals have been constructed by LeonardoSpectrum synthesizer. According to the technique described in [5] a logic circuit of PLA with memory elements as RS-flip-flops has been constructed that implements the considered sequent automaton. The model of the given circuit is described in VHDL. The simulation showed the equivalence of the temporal behavior of all models, i.e. initial algorithmic models of parallel and sequent automata and three structure models of logic circuits.

4. CONCLUSION

The suggested methods for transformation of PRALU descriptions of parallel and sequent automata are easily implemented in computer programs and intended to create the integrated design environment LeonardoSpectrum-LOCON [4]. The author acknowledges Yu.V. Pottosin for useful discussions on this work.

The research is supported by International Scientific and Technology Center (Project B-986).

REFERENCES

[1] A. D. Zakrevskij: *Parallel Algorithms for Logical Control*, Institute of Engineering Cybernetics of NAC of Belarus, Minsk 1999, 202 p. (in Russian)

[2] P. N. Bibilo: *Synthesis of Logical Circuits Using VHDL*, Solon-R, Moscow, 2002, 384 p. (in Russian)

[3] A. V. Kovalyov, Yu. V. Pottosin: *On decomposition of parallel logical control algorithms.* Avtomatgika i vychislitelnaya tekhnika, 1988, № 1, pp. 8–13

[4] V. I. Romanov: *Algorithmic design in LOCON system*, Logical design, Issue 5, Institute of Engineering Cybernetics of NAC of Belarus, Minsk, 2000, pp.137-146 (in Russian)

[5] A. D. Zakrevskij: *Logical Synthesis of Cascade Networks*, Nauka, Moscow, 1981, 416 p. (in Russian)