The International Workshop
on Discrete-Event System Design, DESDes'01,
June 27÷29, 2001; Przytok near Zielona Góra, Poland

PETRI NET MODELS OF VHDL CONTROL STATEMENTS

Ewa IDZIKOWSKA

Dpt. of Control, Robotics and Computer Science, Technical University of Poznań,
pl. M. Skłodowskiej-Curie 5, 60-965 Poznań, POLAND,
Idzikowska@sk-kari.put.poznan.pl

Abstract. *Hardware description languages are used to model logical systems on the behavioural level. The model describes the system as a set of interconnected processes, which can be executed in parallel and represents two aspects of the process – computation and control. Control can be modelled by Petri nets. Such a model is very useful to analyse, which state can be reached starting from a given system state. Petri net models of all VHDL control statements are shown in this paper. These models are used to generate automatically control flow model.*

Key Words. *Hardware Description Language, Petri nets, control flow*

1. INTRODUCTION

In the VLSI area a structured design process is required. In response to this need hardware description languages (HDL) are developed. VHDL is a language for describing digital electronic systems. It is designed to fill a number of needs in the design process. Firstly, it allows description of the structure of a design, that is how it is decompose into sub-designs and how these sub-designs are interconnected. Secondly, it allows the specification of the function of designs using familiar programming language forms. Thirdly, as a result, it allows a design to be simulated before being manufactured, so that designers can quickly compare alternatives and test for correctness without the delay and expense of hardware prototyping. VHDL models are also used in the process of generating test and silicon compilation.

2. CONCURENCY

The hardware description languages must have a mechanism for modelling signals flow through the circuit. In VHDL (VHSIC Hardware Description Language) this requirement is handled by the process construct. Each process represents a block of logic and all processes execute in parallel. The process construct represents the method by which concurrent activities (parallel signal flow) in digital circuits are modelled.

The VHDL model represents two separate aspects of the process - computation and control, but the model does not explicitly distinguish between control and data. We have to find out ourselves this different semantic. The control part of circuit usually has much less states than

the data part thus it is possible and feasible to derive the control structure This structure can be represented by using Petri nets. It is interesting to analyse, which state can be reached starting from a given system state.

3. CONTROL FLOW

As mentioned above, the VHDL control information can be represented using Petri nets. The sequence of instructions, the flow of information and the order of computation performance can be modelled by means of Petri nets.

The control flow model is derived from the VHDL source code in the following way:

- places represent VHDL code statements,
- transitions represent actions - execution of code statements from its pre-places.

Control flow can be modelled with using Conditional Petri Nets with Time [3].

Def. Control flow model

The Petri-net representation of the control flow in VHDL model, CFPN, is a digraph derived from the VHDL source code, with mapping of VHDL code statements to the places. Transitions represent execution of these statements.

CFPN = (P, T, F, M_o, D, C), where:

P = { p_1, p_2,..., p_n} - a finite set of places,
T = { t_1, t_2,..., t_m} - a finite set of transitions,
M_o = { m_1, m_2,..., m_n} - an initial marking,
D = { d_1, d_2,..., d_m} - a finite set of time intervals associated with transitions,
C = { c_1, c_2,...., c_m} - a finite set of conditions associated with transitions,
F - a control flow relation.

Petri net model of the control flow is automatically derived from the VHDL source program. To do it, it's necessary to find all VHDL control statements and control variables. In order to do it, an index *S* is assigned to each statement. The Petri net place, which represents statement S, has the same index. The next chapters describe Petri net models of VHDL control statements.

4. PROCESSES AND THE WAIT STATEMENT

The primary unit of behavioural description in VHDL is a process. It is a sequential body of code which can be activated in response to changes in state. When more than one process is activated at the same time, they execute concurrently. The process is specified as follows [2]:

PROCESS*(sensitivity_list)*
process declarative part
begin
sequence of statements (1)
END PROCESS (2)

The process is activated initially during the initialisation phase of simulation. It executes all of the sequential, and then repeats, starting again with the first statement. The process may suspend itself by executing a *wait* statement. This is of the form:

WAIT ON *sensitivity_list* UNTIL *condition* FOR *time _expression*; (1)

The *sensitivity_list* of the *wait* statement and the list in the header of the process statement specify a set of signals to which the process is sensitive while it is suspended. When an event

occurs on any of these signals, it means the value of the signal changes, the process resumes and evaluates the condition. If it is true or if the condition is omitted, execution proceeds with the next statement. Otherwise the process resuspends. The *time_expression* indicates the maximum time for which the process will wait. If it is omitted, the process may wait indefinitely.

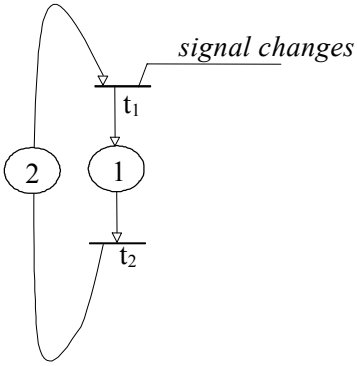

Fig.1. Petri net model of a process.

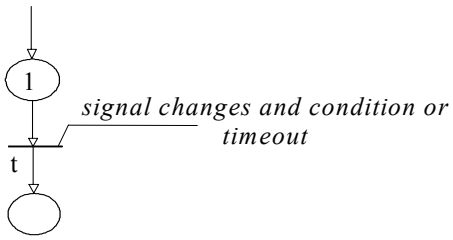

Fig.2. Petri net model of a Wait statement.

A Petri nets model of a process is shown at the Fig. 1. Transition t_1 is the conditional transition and represents an input to the process. It can be fired, when a token is in the place *2*, and the value at least one of the signal from *sensitivity_list* changes. A token in the place 2 indicates, that the process is suspended. After firing t_1 process is resumed (a token in the place *1*) and is active until transition t_2 is fired. Then the process will be suspended again.

Fig. 2 shows Petri net model of *Wait* statement. This statement suspends process. The process is resumed after firing transition *t*.

5. PETRI NET MODEL OF CONTROL STATEMENTS

In this section Petri net models of all control statements are shown. These models are used in the process of automatic control flow model derivation.

There are some control statements in VHDL:

- *IF-THEN-ELSEIF-ELSE*
- *CASE*
- *LOOP*
- *EXIT*
- *NEXT*

5.1. Conditional statement

The full form of the conditional IF statement is as follows [1]:

IF *condition_1* THEN	(1
sequence of statements	(2)
ELSEIF *condition_2* THEN	(3)
sequence of statements	(4)
ELSE	
sequence of statements	(5)
END IF;	(6)

The *elseif* and *else* clauses are optional. Petri net model of IF statement is shown at the Fig.3.

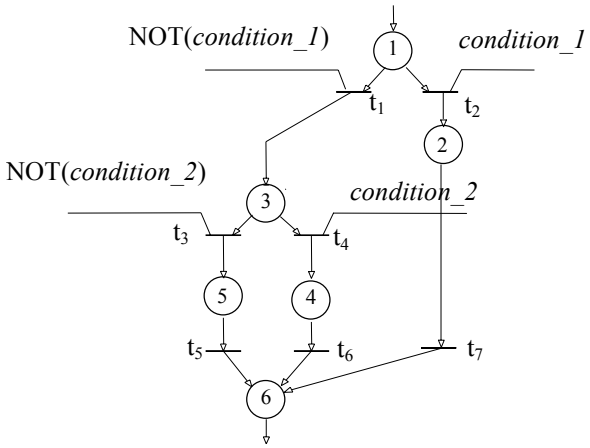

Fig.3. Petri net model of the IF statement.

Transitions t_1, t_2, t_3 and t_4 are conditional. It means, that the each of this transition may be fired if it is enabled and if its condition is *TRUE*.

Transitions t_5, t_6 and t_7 are the time transitions. Each time attributed to these transitions represents delay associated with realization of statements *5, 4* and 7 respectively.

5.2. CASE statement

The *case* statement performs decoding based on the value of a control expression and then executes a selected statement or group of statements. The full form of this statement is as follows:

CASE *n* IS (1)

WHEN *expression_1*⇒*statement_1*; (2)

WHEN *expression_2*⇒ *statement_2*; (3)

…

…

WHEN *expression_n*⇒ *statement_n*; (n+1)

END CASE; (n+2)

The Petri net model of this statement is shown at Fig.4. Transitions t_1, t_2, t_n are conditional transitions and can be fired, if the value of control expression is equal *n*.

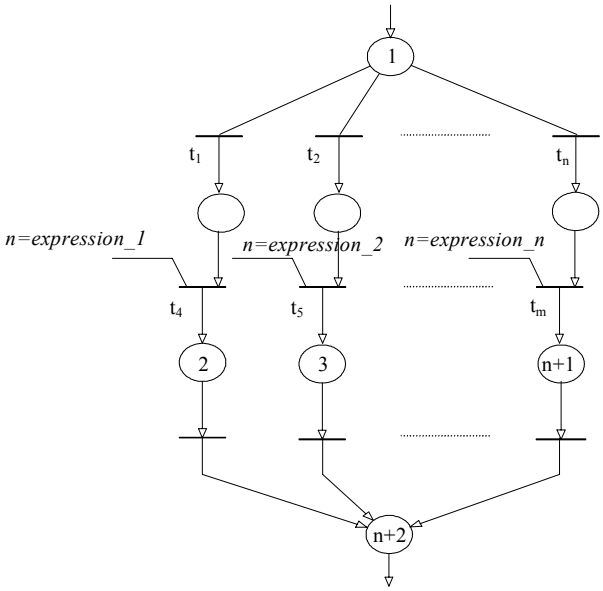

Fig.4. Petri net model of the CASE statement.

5.3. Loop statements

VHDL has a basic *loop* statement, which can be augmented to the usual forms *while* and *for* loops seen in other programming languages. The *for* iteration scheme allows a specified number of iterations. The loop parameter *l* declares an object, which takes on successive values from the range [*wp, wk*] for each iteration of the loop [1].

FOR *l* IN *wp* TO *wk* LOOP (1)

sequence of statements (2)

END LOOP; (3)

The *while* iteration scheme allows a test condition to be evaluated before each iteration. The iteration only proceeds if the test (*l*<= *wk)* evaluates to *TRUE*.

loop label: WHILE *condition* LOOP (1)

sequence of statements (2)

END LOOP *loop label*; (3)

Petri net models of these two loop statements are shown at Fig. 5 and 6 respectively.

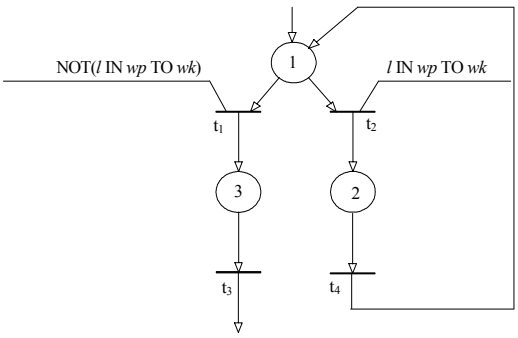

Fig.5. Petri net model of the FOR…LOOP statement.

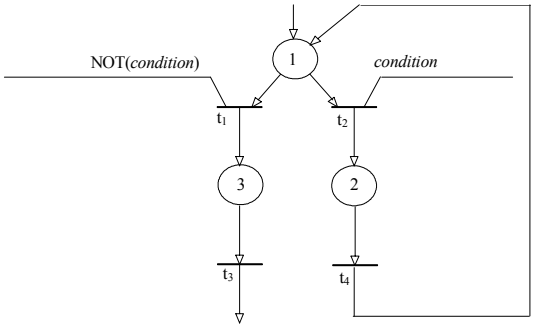

Fig.6. Petri net model of the WHILE…LOOP statement.

There are two additional statements, which can be used inside a loop to modify the basic pattern of iteration. The *next* statement terminates execution of the current iteration and starts the subsequent iteration. The *exit* statement terminates execution of the current iteration and terminates the loop.

loop label: WHILE *condition_1* LOOP (1)

sequence of statements; (2)

NEXT *loop label* WHEN *condition_2* (3)

sequence of statements; (4)

END LOOP *loop statement*; (5)

The Petri net model of this statement is shown at Fig.7.

The loop statement with exit is presented below and its Petri net model is shown at Fig. 8.

LOOP (1)
sequence of statements; (2)

EXIT WHEN *condition*; (3)

sequence of statements; (4)

END LOOP; (5)

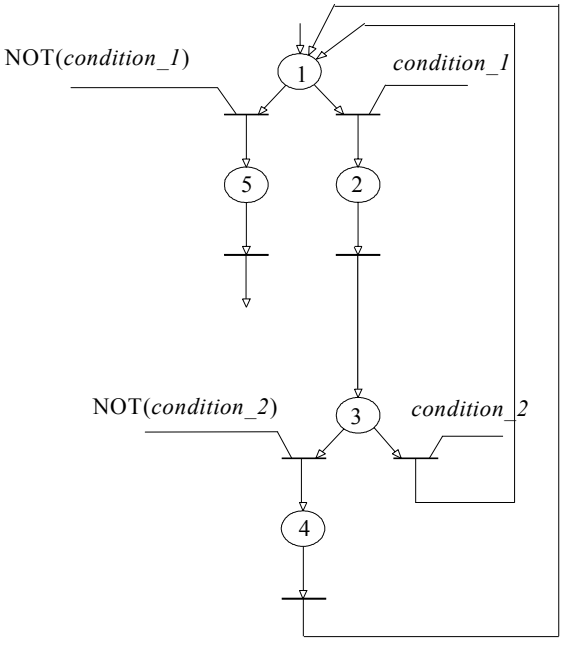

Fig.7. Petri net model of the WHILE…LOOP with NEXT statement.

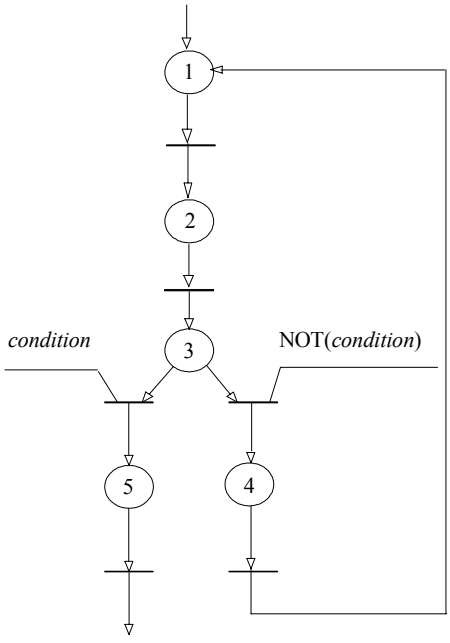

Fig.8. Petri net model of the LOOP with EXIT statement.

6. SUMMARY

In this paper Petri net models of VHDL control statements are presented. These models are used to generate automatically of control flow model. The control flow dictates the partial ordering of data flow in VHDL model and represents the conditions under which the processes are activated. Petri net model of VHDL control flow is very useful in the process of generating tests for design verification.

REFERENCES

[1] Armstrong J.R., *Chip-level modeling with VHDL*. Prentice Hall, Englewood Cliffs, New Jersey,1989

[2] Ashenden P.J., *The VHDL Cookbook*, Dept. Computer Science University of Adelaide, South Australia 1990

[3] E. Idzikowska, Generation Of Validation Tests From Behavioural Description In Vhdl *Proc. of Sixth Annual Advanced Technology Workshop ATW98, Ajaccio, France 1998*