Capacitive Values Extension for std_logic_1164

Masamichi Kawarabayashi
NEC Corporation
1753 Shimonumabe, Nakahara-ku
Kawasaki, Kanagawa 211-8666 Japan
kaba@lsi.nec.co.jp

Naohiro Kobayashi
Mitsubishi Electric Corporation
4-1 Muzuhara
Itami, Hyogo 664-8641 Japan
kobayas1@lsi.melco.co.jp

Abstract

This paper proposes an extension of IEEE Std 1164 (MVL-9) for modeling Intellectual Properties (IP's) including dynamic circuits. The new logic values to be added to IEEE Std 1164 are three capacitive values: C (capacitive unknown), D (discharge), and P (precharge), which are essential for precise modeling of delay time and power dissipation with consideration of charge decay. Proposed standard logic value package is described, then the behavior of realistic circuits is illustrated. Finally, the impact of this extension on the current design environments and EDA tools is discussed.

1 Introduction

IEEE Std 1164[2] (MVL-9) declares nine logic values, i.e. 'U', 'X', '0', '1', 'Z', 'W', 'L', 'H' and '-', as "std_ulogic" type. These values would be complete enough to model most of static circuits. From this reason, MVL-9 is adopted in the VITAL[3] libraries to model gate level circuits. As a number of Intellectual Properties[5][6][7], e.g. IP's, described in VHDL[1] are recently supplied, more precise modeling of these variety of circuits is required.

However, when the behaviors of dynamic circuits are to be modeled using VHDL, it is difficult to model them precisely with the existing MVL-9. For example, in the logic circuits fabricated using quarter-micron technology, delay time and power dissipation calculated using MVL-9 are usually larger than those of real circuits. These inaccuracies in delay time and power dissipation estimation will become fatal problems when IP's with the dynamic circuits are modeled in VHDL and VITAL. One of the practical solutions to this problem is to add three capacitive values to current IEEE Std 1164, which yield more accurate estimations of delay time and power dissipation of these circuits.

The organization of this paper is as follows. In section 2, the static features of these new capacitive values are discussed. In section 3, the idea of charge delay is described. In section 4, the advantages of our proposal are shown. In section 5, the impacts of our proposal on the current design environments and EDA tools are summarized. Finally, the possible implementation of the extended resolution functions is discussed in section 6.

2 Static specification of proposed value set

2.1 Declaration of value set

The standard type "std_ulogic" is declared as an enumeration type of nine logic values, 'U', 'X', '0', '1', 'Z', 'W', 'L', 'H', and '-', in the standard package of IEEE Std 1164. In the current standard, 'X', '0' and '1' have **strong** strength, and represent the state of objects with low impedance drivers. Then, 'W', 'L' and 'H' have **weak** strength, which are weaker than **strong** strength, and represent the state of pull-up or pull-down through resistance. Finally, 'Z' has the **weakest** strength, and represents the state of objects without any drivers. When CMOS dynamic circuits are designed, the behaviors of **charge** and **discharge** are essential for accurate modeling. However, these behaviors can not be accurately modeled in MVL-9.

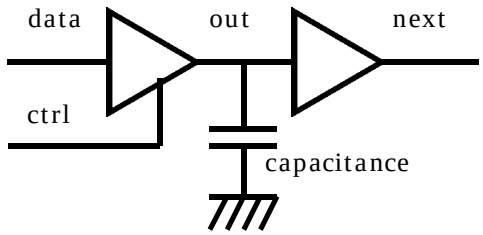

Figure 1: Example for the circuit of capacitive value

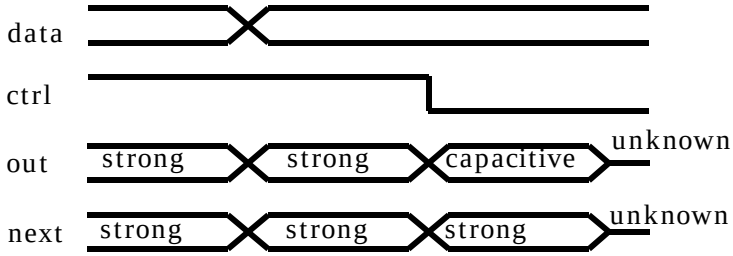

Figure 2: Example for the waveform of capacitive value

The behavior of charge/discharge mechanism in a dynamic circuit can be explained as shown in Figures 1 and 2. When the signal "ctrl" becomes '1', signal "out" will be driven by a logic value that has strong strength. Then charge or discharge will be performed on the capacitance. When "ctrl" becomes '0', "out" will be separated from any drivers, then the electrical charge will be preserved in the capacitance for a while. At this time "out" has enough strength to drive the signal "next", then the value of signal "next" will be fixed with a certain logical value. However, the value of signal "out" will become a "high impedance" due to the leakage current, then the signal "next" will reach the "unknown" state.

In short when the state of a signal changes from a strong value to 'Z', some intermediate states with **capacitive** strength should be considered to model the CMOS dynamic circuits more accurately. This strength should be distinguished from 'Z' strength. Provided these capacitive values were not used, when "ctrl" becomes '0', the values of signals "out" and "next" will immediately become 'Z' and unknown 'X', respectively. This behavior does not reflect the actual physical phenomenon exactly. Thus, logical values with capacitive strength are indispensable as well as strength of strong, weak and high impedance to perform more accurate modeling, especially in the deep sub-micron designs.

Therefore, we propose to add three new values i.e. 'C', 'D', and 'P', to represent the **capacitive** strengths, where 'C' (capacitive unknown) represents "capacitive with unknown charge" (positive or negative charge); 'D' (discharge) represents capacitive with the existence of negative charge; then 'P' (precharge) represents capacitive with the existence of positive charge. Proposed logic values with these capacitive values are summarized in Table 1.

2.2 Declaration of the strength between values

The signals with capacitive strengths can not preserve certain value for a long time due to the leakage current. It is clear that these strengths are weaker than the strengths through resisters. Therefore, the strengths of 'C', 'D' and 'P' should be placed between weak and high impedance,

Value / Strength	Low	High	Unknown
Initialize	U		
Strong	0	1	X
Weak	L	H	W
Capacitive	D	P	C
High Impedance	Z		

Table 1: Strengths and values

```
TYPE std_ulogic IS
 ( 'U',  -- Uninitialized
   'X',  -- Forcing  Unknown
   '0',  -- Forcing  0
   '1',  -- Forcing  1
   'Z',  -- High Impedance
   'W',  -- Weak    Unknown
   'L',  -- Weak    0
   'H',  -- Weak    1
   '-',  -- Don't care
   'C',  -- Capacitive unknown
   'D',  -- Discharge 0
   'P'   -- Precharge 1
   );
```

Figure 3: Extended declaration of std_ulogic type

```
CONSTANT cvt_to_ux01 :
   logic_ux01_table := (
       'U',  -- 'U'
       'X',  -- 'X'
       '0',  -- '0'
       '1',  -- '1'
       'X',  -- 'Z'
       'X',  -- 'W'
       '0',  -- 'L'
       '1',  -- 'H'
       'X',  -- '-'
       'X',  -- 'C'
       '0',  -- 'D'
       '1'   -- 'P'
        );
```

Figure 4: Extended declaration of "cvt_to_ux01" constant

which are weaker than 'U', 'X', '0', '1', 'W', 'L', 'H' and '-', but stronger than 'Z'.

Two types of extensions can be considered. One is to declare a new type which has the different name from "std_ulogic", the other is to use the same type with additional 3 values. We select the latter one for users' convenience in this paper. These capacitive values are placed at the right end of the "std_ulogic" declaration to preserve the compatibility as much as possible with existing IEEE Std 1164. The new "std_ulogic", called MVL-12 is declared as shown in Figure 3.

2.3 Declarations of logical operations and type conversion functions

Extensions of logical operations and type conversion functions are needed to treat capacitive values in MVL-12. Extended type conversion functions are designed as follows.

(1) To_UX01

'C': The output is assigned to 'X' since it has an unknown value as 'X'.
'D': The output is assigned to '0' since it has a low value as '0'.
'P': The output is assigned to '1' since it has a high value as '1'.

(2) "cvt_to_ux01"

The constant declaration of "cvt_to_ux01" is extended and shown in Figure 4.

(3) To_bit

'C': The output is assigned to '0', which is the initial value of "bit" type, since it has an unknown value and is treated similarly to the other values except for '0' and '1'.
'D': The output is assigned to '0' since it has a low value as '0'.
'P': The output is assigned to '1' since it has a high value as '1'.

(4) To_X01

'C': The output is assigned to 'X' since it has an unknown value and is treated similarly to the other values except for '0' and '1'.
'D': The output is assigned to '0' since it has a low value as '0'.
'P': The output is assigned to '1' since it has a high value as '1'.

(5) To_X01Z

'C': The output is assigned to 'X' since it has an unknown value, and is treated similarly to the other values except for '0' and '1', and is distinguished with 'Z'.
'D': The output is assigned to '0' since it has a low value as '0'.
'P': The output is assigned to '1' since it has a high value as '1'.

(6) Is_X

'C': The output is assigned to "TRUE" since it has an unknown value, is treated similarly to the other values except for '0' and '1'.

'D': The output is assigned to "FALSE" since it has a low value as '0'.

'P': The output is assigned to "FALSE" since it has a high value as '1'.

According to the discussion on the strength relation between values in section 2.2, the capacitive values are weaker than strong values or weak values, but stronger than the high impedance value. Therefore, the resolution function "resolve" can be extended as in the "resolution_table" shown in Figure 5. Similarly, "and_table", which is used in the logical operation "and", is shown in Figure 6.

3 Dynamic specification of proposed value set

In this section we consider the dynamic behavior of the circuit, which uses the value set in MVL-12, and discuss the implementation of charge decay.

3.1 Concept of charge decay

The static specifications of the value set in MVL-12 were discussed in the previous section. Though the capacitive values 'C', 'D' and 'P' were added into the existing MVL-9, only the extension of the value set is not sufficient in order to handle the capacitive values and to model the behavior of dynamic circuits. The transition time for the capacitance should be considered.

```
CONSTANT resolution_table : stdlogic_table := (
--      ---------------------------------------------------------------------
--      |  U    X    0    1    Z    W    L    H    -    C    D    P        |  |
--      ---------------------------------------------------------------------
        ( 'U', 'U', 'U', 'U', 'U', 'U', 'U', 'U', 'U', 'U', 'U', 'U' ), -- | U |
        ( 'U', 'X', 'X', 'X', 'X', 'X', 'X', 'X', 'X', 'X', 'X', 'X' ), -- | X |
        ( 'U', 'X', '0', 'X', '0', '0', '0', '0', 'X', '0', '0', '0' ), -- | 0 |
        ( 'U', 'X', 'X', '1', '1', '1', '1', '1', 'X', '1', '1', '1' ), -- | 1 |
        ( 'U', 'X', '0', '1', 'Z', 'W', 'L', 'H', 'X', 'C', 'D', 'P' ), -- | Z |
        ( 'U', 'X', '0', '1', 'W', 'W', 'W', 'W', 'X', 'W', 'W', 'W' ), -- | W |
        ( 'U', 'X', '0', '1', 'L', 'W', 'L', 'W', 'X', 'L', 'L', 'L' ), -- | L |
        ( 'U', 'X', '0', '1', 'H', 'W', 'W', 'H', 'X', 'H', 'H', 'H' ), -- | H |
        ( 'U', 'X', 'X', 'X', 'X', 'X', 'X', 'X', 'X', 'X', 'X', 'X' ), -- | - |
        ( 'U', 'X', '0', '1', 'C', 'W', 'L', 'H', 'X', 'C', 'C', 'C' ), -- | C |
        ( 'U', 'X', '0', '1', 'D', 'W', 'L', 'H', 'X', 'C', 'D', 'C' ), -- | D |
        ( 'U', 'X', '0', '1', 'P', 'W', 'L', 'H', 'X', 'C', 'C', 'P' )  -- | P |
);
```

Figure 5: Extended declaration of "resolution_table" constant

```
CONSTANT and_table : stdlogic_table := (
--      ---------------------------------------------------------------------
--      |  U    X    0    1    Z    W    L    H    -    C    D    P        |  |
--      ---------------------------------------------------------------------
        ( 'U', 'U', '0', 'U', 'U', 'U', '0', 'U', 'U', 'U', '0', 'U' ),  --| U |
        ( 'U', 'X', '0', 'X', 'X', 'X', '0', 'X', 'X', 'X', '0', 'X' ),  --| X |
        ( '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0' ),  --| 0 |
        ( 'U', 'X', '0', '1', 'X', 'X', '0', '1', 'X', 'X', '0', '1' ),  --| 1 |
        ( 'U', 'X', '0', 'X', 'X', 'X', '0', 'X', 'X', 'X', '0', 'X' ),  --| Z |
        ( 'U', 'X', '0', 'X', 'X', 'X', '0', 'X', 'X', 'X', '0', 'X' ),  --| W |
        ( '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0' ),  --| L |
        ( 'U', 'X', '0', '1', 'X', 'X', '0', '1', 'X', 'X', '0', '1' ),  --| H |
        ( 'U', 'X', '0', 'X', 'X', 'X', '0', 'X', 'X', 'X', '0', 'X' ),  --| - |
        ( 'U', 'X', '0', 'X', 'X', 'X', '0', 'X', 'X', 'X', '0', 'X' ),  --| C |
        ( '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0', '0' ),  --| D |
        ( 'U', 'X', '0', '1', 'X', 'X', '0', '1', 'X', 'X', '0', '1' )   --| P |
);
```

Figure 6: Extended declaration of "and_table" constant

When a signal with a strong value, is cut off from a driver, the stored charge is gradually lost because of the leakage current. After a certain time, the signal becomes the high-impedance state. Because of the attenuation of charge, which is called “charge decay”, the value of the signal may change even if the MOS transistors in the circuit do not switch. Therefore, in order to determine the change in values with the capacitance, we have to model not only the change of driver's value, but also the charge decay. The former means the external factor, and the latter means the internal factor.

3.2 How to describe charge decay

For dynamic circuits, we consider the resolution mechanism of the value, where two or more drivers drive one signal. This situation should be handled by the resolution function in VHDL. However, when considering the capacitance of the signal, the mechanism of the resolution function in VHDL is not appropriate. Because only the external factor can be taken into account for the resolution function, but the internal factor of charge decay cannot be considered.

A special entity within the existing VHDL specifications is introduced to simulate the behavior of charge decay. The behavior of “capacitive” entity is shown in Figure 7. Input and output ports are shared, and the output value is one of four values ‘Z’, ‘C’, ‘D’ and ‘P’ due to the input value. One signal is declared to preserve the charged previous value. The behavior of charge decay is described on the line of "when others". When the signal is not driven, the charged previous value is preserve for “delay” time, which is defined as a generic parameter. Then, output value become high impedance. Consequently the signal resolution mechanism with capacitance elements can be described using this special entity.

A circuit example is shown in Figure 8 to simulate capacitance on the signal with the special entity.

4 Effects of capacitive value extension

After the capacitive values ‘C’, ‘D’ and ‘P’ are added to the MVL-9, the accuracy of the gate-level simulation can be improved. It was impossible to express the intermediate state between the high-impedance ‘Z’ and the other values when we used MVL-9. Therefore, when signal transitions occur in the intermediate state described above, the transitions cannot be analyzed precisely by logic simulators. For the real circuits, the following three types of transition delay time are not the same, while it is impossible to distinguish mutually using MVL-9.

```
entity capacitance is
    generic (delay : time := 100ns);
    port (Tbus  : inout std_ulogic
    );
end capacitance;

architecture RTL of capacitance is
    signal Data : std_ulogic := 'Z';
begin
    process(Tbus)
    begin
        case Tbus is
           when 'X'|'W' => Data <= 'C';
           when '0'|'L' => Data <= 'D';
           when '1'|'H' => Data <= 'P';
           when others  => Data <= Data, 'Z' after delay;
        end case;
        Tbus  <= Data;
    end process;
end RTL;
```

Figure 7: Entity for capacitance

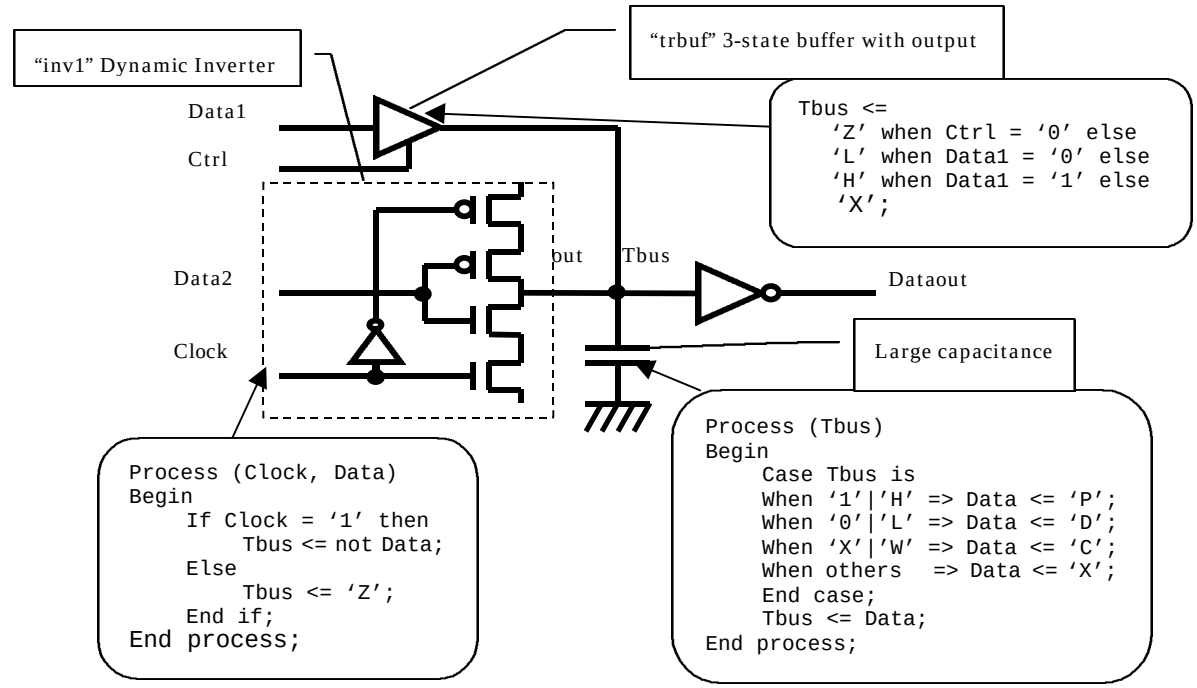

Figure 8: Example for the modeling of 'C', 'D' and 'P'.

(1) Signal transition delay from the intermediate state between 'X' and 'Z' to '1'.
(2) Signal transition delay from the intermediate state between '0' and 'Z' to '1'.
(3) Signal transition delay from the intermediate state between '1' and 'Z' to '1'.

The individual delay model cannot be assigned to each case when using MVL-9. Therefore, when verifying the timing of circuits using MVL-9, the signal transition delay of the circuit is always defined as the worst case of above three values.

Recently, high-performance and high-speed LSIs have been developed, and tight timing constraints have been assigned to logic circuits. Precise delay models are required to design such circuits. When we use MVL-12 proposed here, the individual delay value can be assigned to each signal transition delay in the circuit. For the intermediate state in the (1), (2) and (3) can be expressed by the value 'C', 'D' and 'P', respectively. The accuracy of the delay time can be improved by introducing MVL-12, and models for precise timing can be realized. It is indispensable to design high performance LSIs.

Similarly, the effectiveness of MVL-12 for the power dissipation is discussed. The rapid growth

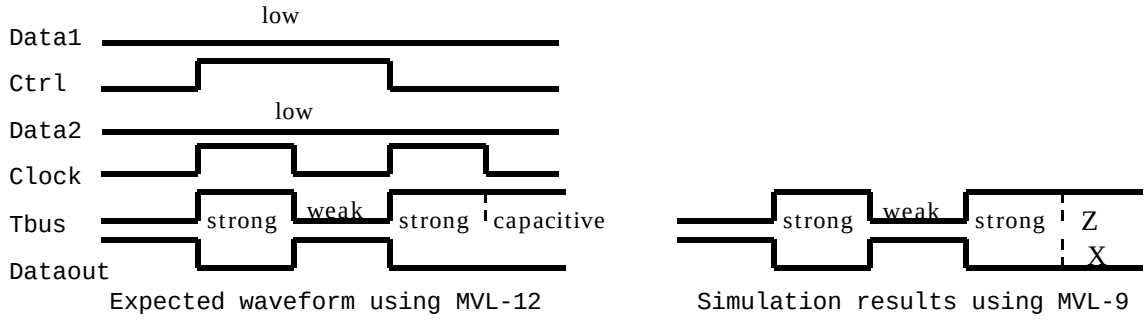

Figure 9: Waveforms for the example in Figure 8

of the market for portable electric appliances and the heat problem for LSI packages cause great interest in low-power designs. As for the power dissipation, only one power model is assigned to three types of the signal transition shown in the above example when using MVL-9 model. In order to calculate precise power dissipation for the low-power design, the detailed power models are required. The individual power models can be assigned to three types of the signal transitions when using MVL-12 for designing low-power LSIs.

An example using MVL-12 is shown in Figure 8. “trbuf”, “inv1” and “Tbus” are modeled in VHDL. The VHDL description for “Tbus” is expressed with the capacitive values. The model for “Tbus” cannot be represented with MVL-9. Figure 9 shows the input waveforms of the signals “Data1”, “Ctrl”, “Data2” and “Clock”, and the waveforms of the signals “Tbus” and “Dataout” obtained by the logic simulation. The expected simulation result with MVL-12 is shown on the left side of Figure 9, and the simulation result with the MVL-9 is shown on the right.

We assumed that the output value of “trbuf” is equal to ‘Z’. In case of MVL-12, when the output value of “inv1” changes from ‘1’ to ‘Z’, the value of “Tbus” changes from ‘1’ to ‘P’, then changes to ‘Z’ after “delay” time. Meanwhile, in case of MVL-9, the value of “Tbus” changes from ‘1’ to ‘Z’ immediately, and the intermediate state between ‘1’ and ‘Z’ cannot be considered. Consequently, more precise modeling can be done with MVL-12 when designing the 3-state circuit structure.

Under the quarter-micron technologies, in proportion as the increase of the wire resistance for the unit wire length, the wire delay has become dominant. Therefore, precise delay simulation or analysis is essential for long wire signals. Because the capacitive values are introduced and delay models of signals can be defined precisely, the precise analysis can be realized.

Power models can be also defined precisely. It is appropriate to use the precise models for designing low-power and high-speed LSIs. Recently, the IP reuse techniques[5][6][7] have become more popular. When reusing IPs with 3-state signal interfaces, the capacitive values can be used more effectively.

5 Impacts on the current design environments

In this section we discuss the impact on the current design environments, e.g. VHDL language specification, standards and EDA tools.

5.1 VHDL language specification

Some declarations in the standard library of IEEE Std 1164 should be modified because of the extension of value set. All modifications of the declarations for the basic operators and the type conversion functions are within the scope of the VHDL language specification. The resolution mechanism of values on the signal is essential in order to represent the charge decay. Within the scope of the VHDL language specification, it can be realized by introducing the special entity. Consequently, there are no impacts on VHDL language specification when the extended value set is introduced. However, we can investigate to extend the specification of the resolution mechanism in VHDL. One of the further extension is discussed in section 6.

5.2 Standard library

Various functions in the STD_LOGIC_1164 package are re-declared owing to the re-declaration of “std_ulogic” type. Note that the name of library, package, types and functions are not necessary to be changed. Consequently, there are few possibilities to change the designers’ descriptions when the MVL-12 package is referred.

5.3 VITAL

When MVL-12 is applied to the VITAL[3] libraries, it is indispensable to modify the assignments of output values for instances, e.g. switches, 3-state elements, and MOS transistors, in which charge decay can be handled, and 'C', 'D' and 'P' should be effectively used. Though the impacts for VITAL specification might be relatively large, it should be accepted if the precise modeling is requested for deep sub-micron designs.

5.4 SDF

The transition patterns should be added because the three additional values are introduced. The current delay calculator, which can calculate the delay time from one value to another value, should be improved. After the charge decay is considered in those EDA tools and SDF, it will be easy to design dynamic circuits.

5.5 EDA tools

The impacts for EDA tools are various on account of the mechanisms to handle the "std_logic" type. It may be minimized for EDA tools because only the static specification, i.e. value set and some functions, are modified.

The new value set may cause errors in some EDA tools, e.g. simulators, because they can not recognize new values. This problem may be solved after the re-compilation of the packages. However, it is preferable to optimize the EDA tools for the new value set if the best environments for designers are provided. For instance, the waveform viewers for the logic simulators have to help designers to identify the additional values from the waveforms. Moreover, it will be requested to improve the EDA tools which achieves some accelerations for a standard library, e.g. VITAL simulators. Though it seems to be necessary to improve the logic synthesis tools to recognize the added values, the modification can be minimized for those tools because the added values are able to be handled as well as the other meta-logical values[4].

6 Further discussion for language extensions

In section 3, we discussed that it is impossible to describe the behavior of charge decay by the resolution function within the current language specifications of IEEE Std 1076. Therefore, we

```
impure function charged(driver: in std_ulogic_vector) return std_ulogic is
    alias previous: std_ulogic is charged'resolved_signal;
   signal resolved: std_ulogic;
   variable tmp: std_ulogic:= 'Z';
begin
   -- resolve all inputs by static analysis (first stage)
   for i in driver'range loop
      tmp := resolution_table(tmp, driver(i));
   end loop;
   -- consider charge decay behavior when Hi-Z (second stage)
   if tmp = 'Z' then
      case previous is
         when 'X'|'W' => resolved <= 'C', 'Z' after previous'decay_time;
         when '0'|'L' => resolved <= 'D', 'Z' after previous'decay_time;
         when '1'|'H' => resolved <= 'P', 'Z' after previous'decay_time;
        when others  => resolved <= 'Z';
      end case;
   else
      resolved <= tmp;
   end if;
   return(resolved);
end;
```

Figure 10: Extended VHDL description for charge decay

consider what type of extensions for the resolution functions are required. As well as section 3, we assume the resolution mechanism of the values, where two or more drivers drive one signal. One example description out of various extensions is shown in Figure 10.

Though the resolution function in IEEE Std 1076-1993 shall be pure function, the extended resolution function “charged” is impure function. It has an attribute “resolved_signal”. This attribute is attached to the signal which preserves the previous resolved value. Each reference of the function should correspond to each resolution signal individually. In addition, the signal has a attribute “decay_time”, which represents the decay time. “previous” is the signal which can refer the previous value. In the function body, an intermediate variable “tmp” is calculated statically from the values of the “driver”. The behavior of charge decay is considered. Therefore, if “tmp” is high impedance, one of capacitive values is preserved in “resolved” for “decay_time” before assigning 'Z'.

In summary, two types of new features are required if we realize the solution.

- Resolved signals shall be referred to in the resolution function.
- Signal assignment statement shall be described in the resolution function.

7 Conclusion

We proposed that three capacitive values should be added into the standard values in IEEE Std 1164. The behavior of the realistic circuits is illustrated when the standard package is extended. This extension will be essential for the model of dynamic circuits under the quarter-micron technology. In order to design IPs with the dynamic circuits effectively using the standard package of IEEE Std 1164, it shall be extended in near future.

Acknowledgements

This paper is the effort of VHDL Project Group/EDA Technical Committee/EIAJ in 1997. We appreciate the collaboration with Mr. T. Kowatari, Mr. N. Fujiike, Mr. H. Sasaki, Mr. M. Yokoyama, Mr. K. Makino, Mr. T. Kitahara, Mr. S. Sekiguchi, Mr. S. Katayama, Mr. K. Matsuzaki, Mr. M. Mizuno and Ms. K. Fuse. We also appreciate Prof. M. Imai and Mr. S. Kojima for the technical supports.

References

[1] “IEEE Standard VHDL Language Reference Manual”, Institute of Electrical and Electronics Engineers Inc., IEEE Std 1076-1993

[2] “IEEE Standard Multivalue Logic System for VHDL Model Interoperability (Std_logic_1164)”, Institute of Electrical and Electronics Engineers Inc., IEEE Std 1164-1993

[3] “IEEE Standard for VITAL Application-Specific Integrated Circuit (ASIC) Modeling Specification”, Institute of Electrical and Electronics Engineers Inc., IEEE Std 1076.4-1995

[4] “IEEE Standard VHDL Synthesis Packages”, Institute of Electrical and Electronics Engineers Inc., IEEE Std 1076.3-1997

[5] R. Glover, D. Fairbairn, L. Cooke, S. Schulz, T. Inoue, R. Raghavan, J.L. Bories, W. Rhines “Panel: Challenges in Worldwide IP Reuse”, in Proc. of 34th Design Automation Conference, pp401 1997

[6] E.F. Girczyc, S. Carlson, “Increasing Design Quality and Engineering Productivity through Design Re-use”, in Proc. of 30th Design Automation Conference, pp48 1993

[7] A.A. Jerraya, H. Ding, P. Kission, M. Rahmouni, “Behavioral Synthesis and Component Reuse with VHDL”, Kluwer Academic Publishers, pp18 1997