Extension of VHDL to support multiple-byte characters

Kiyoshi Makino
Seiko Instruments Inc.
6-41-6 Kameido, Koto-ku
Tokyo 136-8512 Japan
kmakino@sii.co.jp

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

Abstract

Written Japanese is comprised of many kinds of characters. Whereas one-byte is sufficient for the Roman alphabet, two-byte are required to support written Japanese. It should be noted that written Chinese require three-byte. The scope of this paper is not restricted to written Japanese because we should consider the implementation of the standard which covers the major languages using the multiple-byte characters.

Currently, VHDL does not support multiple-byte characters. This has proven to be a major impediment to the productivity for electronics designers in Japan, and possibly in other Asian countries. In this paper, we briefly describe the problem, give a short background of the character set required to support Japanese and other multiple-byte characters language, and propose a required change in the IEEE Std 1076.

1. Introduction

VHDL[1] have recently become very popular to design the logical circuits all over the world. We, the Electronic Industries Association of Japan (EIAJ), have been working for the standardization activities with the Design Automation Sub-Committee (DASC) in IEEE. In the meanwhile, the VHDL and other related standards have evolved and improved. As the number of designers using VHDL increases so does the need to support the local language though designers have asked EDA tool vendors to enhance their products to support local language, it have often been rejected unfortunately.

There are several reasons for this lack of responses on the part of the EDA tool vendors. First, the vast majority of VHDL based tools available today are developed and enhanced in western countries which use the Roman alphabet. Hence, the developers do not know what is necessary to implement a multiple-byte characters. Secondly, this is related to the first issue, since they do not know how to handle multiple-byte characters[2] in their tools, they overestimate the complexity involved. Practically the required change can be minimal. Thirdly, their software must be compliant with the IEEE Std 1076, and since the Language Reference Manual (LRM) does not spell out the support of multiple-byte characters, they are reluctant to deviate from compliance.

If the LRM were amended to support for multiple-byte characters, all three issues mentioned above would be adequately addressed. A clear standard with implementation guidelines would remove the mystery shrouding multiple-byte characters, and inclusion in the standard would ensure implementation.

2. Support of multiple-byte characters within VHDL

There are three levels of support for multiple-byte characters possible within VHDL. They are:

a) VHDL Identifier
b) CHARACTER/STRING literal
c) Comment text

```
a) signal 同期信号 : std_logic;
b) assert false report "信号処理を間違えています";
c) Y <= not A;  -- 日本語のコメント
```

Figure 1 shows an example of VHDL descriptions with Japanese characters

Figure 1 shows an example of VHDL descriptions with Japanese characters.

First, there is absolutely no need to support multiple-byte characters for VHDL identifiers. Some designers request Japanese text as type of *CHARACTER*. But defining Japanese text as *CHARACTER* type is not easy. Current *CHARACTER* type in *STANDARD* package is defined as enumeration type of each character. Because the number of characters in Japanese or Chinese are huge (6000 - 34000), it is not sufficient way to define a new character type as enumeration type. One possible way to define such new character type in LRM is to refer other character set standard defined by ISO etc. Each Asian country has its own character set standards, and these are always changing. Therefore it is not easy to make a consensus to choose one character set. Though,we should request to support a new character type for multiple-byte characters, but because of its technical difficulty, we have to consider it as a future work.

However, the comment texts are vitally important for designers. VHDL is considered as a specification language and comments embedded within the VHDL description are required to explain the expected behavior. This becomes even more vital as the VHDL description is reused in a system-on-chip design environment.

We think that multiple-byte characters as VHDL comments are the most important issue and could be resolved in short term.

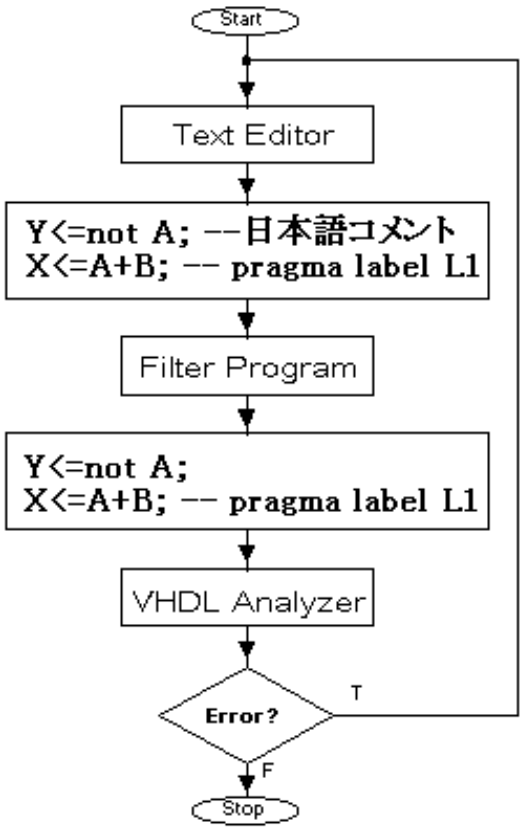

Figure 2: Traditional filtering procedure

3. Workaround employed today

There are three steps to support the comments using the multiple-byte characters in the available EDA tools.

a) Officially support multiple-byte character comments
b) Unofficially support multiple-byte character comments
c) Do not support multiple-byte character comments

VHDL analyzer of type a) are few and far between, and not offered by the major vendors. VHDL analyzer of types b) and c) require a complicated procedure to guarantee that it works correctly.

This process is used for both types of tools, even though some tools "unofficially" support multiple-byte character comments. The unofficial support often results in unexpected errors and problems, therefore b) and c) are

handled identically. The process is as follows: (See Figure 2.)

1. Write the VHDL with embedded multiple-byte code comments
2. Pass the VHDL description through a "filter" to remove or preserve the comments
3. Compile the VHDL code
4. Collect error encountered
5. Repeat from step #1 above.

This process is required to be repeated for each syntax error, modification, or enhancement of the VHDL description. This is time consuming and a potential source of unintentional errors.

4. Overview of multiple-byte character set

"Character set" and "encoding method" are individual issues. "character set" means just a group of characters. "encoding method" means how each character is represented in storage element of the computer. Therefore, different encoding methods may be adopted to represent a single character set. ASCII has both meanings of "character set" and also "encoding".

4.1 Character set

There are many character set standards in JAPAN and other Asian countries, e.g. JIS X 0208-1978, JIS X 0208-1983, JIS X 0208-1990 and JIS X 0212-1990. The number of characters in standard is not stable. New characters appear, and old characters disappear for each re-standardization. Furthermore, there will probably be more standards of character sets in the near future. Therefore, it is impossible to select one character set standard to be supported as a comment text in VHDL.

4.2 Encoding method

Four popular encoding methods exist.

a) JIS ... Modal
b) Shift-JIS ... Non-Modal
c) EUC ... Non-Modal
d) Unicode ... Fixed-width

Figure 3 shows how ASCII characters and Japanese characters are mixed in a same text by each encoding methods.

JIS encoding is modal. In Figure 3-a, <KANJI-IN> and <KANJI-OUT> are escape sequence to change the mode. In this method, actual Japanese character code use only 7-bit range, thus it might conflict with normal ASCII code if the tool does not correctly handle the escape sequence. As JIS encoding method use only 7-bit, it is used for Internet mailing system.

On the other hand, Shift-JIS use 8-bit code, and it is non-modal. This Shift-JIS encoding method has been used for Japanese version of DOS, and Windows environment. (See Figure 3-b)

EUC(Extended UNIX Code) is also non-modal encoding, and it is generally used for UNIX operating system. While Shift-JIS support two-byte code characters only, EUC can support three-byte code characters, therefore EUC can also be used for Chinese characters. (See Figure 3-c)

Unicode (ISO-10646) defines character set standards and the encoding method. Each character always has two-byte(16-bit) fixed-width code.

a) JIS

a	b	c	<ESC>	$	B	日	本	語	<ESC>	(	J	a	b	c
61	62	63	1B	24	42	46-7C	4B-5C	38-6C	1B	28	4A	61	62	63
			<KANJI-IN>								<KANJI-OUT>			

b) Shift-JIS

a	b	c	日	本	語	a	b	c
61	62	63	93-FA	96-7B	8C-EA	61	62	63

c) EUC

a	b	c	日	本	語	a	b	c
61	62	63	C6-FC	CB-DC	B8-EC	61	62	63

d) Unicode

a	b	c	日	本	語	a	b	c
00-61	00-62	00-63	65-E5	67-2C	8A-9E	00-61	00-62	00-63

Figure 3: Japanese encoding methods

However unfortunately, Unicode is not popular at least in Japan. Because there are no EDA tools to handle Unicode practically, we think it is too early to support Unicode.

Basically, we think both EUC and Shift-JIS encoding should be handled in the programming language e.g. C,VHDL. There are few requirements to handle JIS or Unicode now.

4.3 Range used for each encoding method

Each encoding method has individual range of encoding shown in Figure 4.

The character set of ISO 8859-1 is available in VHDL93, therefore we can use any single byte character between 0x21 - 0x7E and 0xA1 - 0xFF. But we can not use 0x80 - 0xA0 specified in Shift-JIS encoding, or 0x8E,0x8F in EUC. These characters exceed the specification of VHDL93.

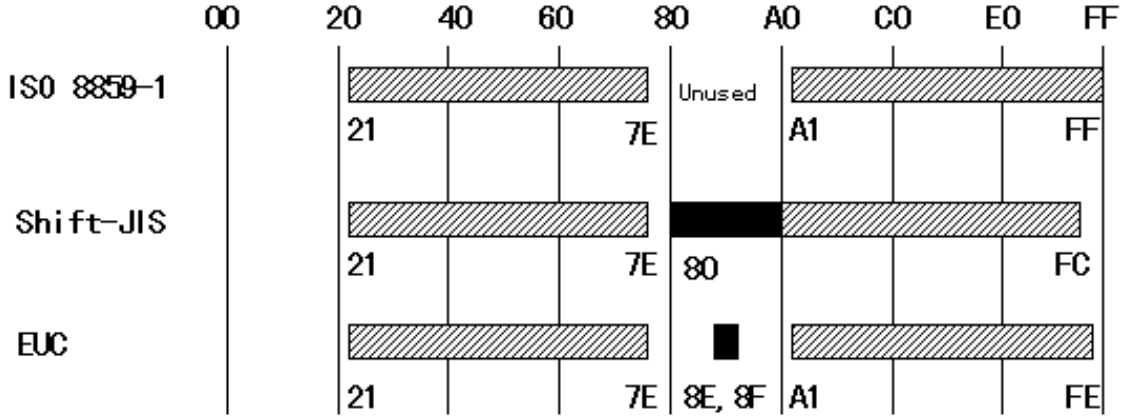

Figure 4: Encoding range

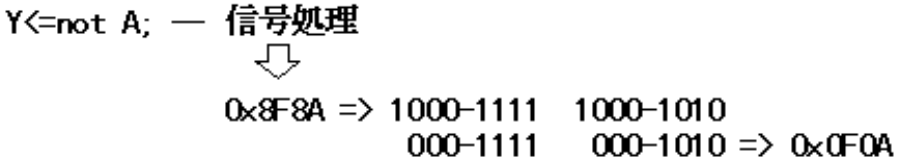

Figure 5: Example of misinterpretation

ISO 8859-1 does not define characters between 0x80 - 0xA0. If the tool treat them as 7-bit character (not in 8-bit), then 0x80 - 0xA0 become 0x00 - 0x20. These characters often cause problem.

For instance, when one of the Japanese characters in comments is 0x8F8A shown in Figure 5, it is out of range of ISO 8859-1. If a tool handles it as 7-bit code instead of 8-bit code, 0x8A become 0x0A, and the poor tool misunderstand it as <end of line> (0x0A). Though some existing EDA tools still have those problems EDA vendors sometimes didn't improve it even customer asked them.

5. Required Change for VHDL LRM

In the IEEE Std 1076-1993, There are 2 sections related to character set and comments. Section 13.1 defines character set in VHDL text as ISO 8-bit character set [ISO 8859-1:1987(E)]. But ISO 8859-1 is not sufficient, so we have to expand it to include 0x80 to 0xA0 character also. Section 13.8 defines comment. We think this section should explicitly define that it allow any character sequence of character in ISO 8859-1 and 0x80 to 0xA0 character from two adjacent hyphens until end of line. As we discussed earlier, we can not choose exact character set. Therefore LRM should define the range of character which appear at the comment text.

6. Conclusions

We presented the reason to support multiple-byte characters in VHDL. Especially the comments written in multiple-byte characters are essential for Asian designers. Moreover it is not difficult to implement to handle them. Consequently the comment in multiple-byte characters should be supported in the next version of VHDL.

Acknowledgments

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. T. Kitahara, Mr. S. Sekiguchi, Mr. S. Katayama, Mr. K. Matsuzaki, Mr. M. Mizuno, Ms. K. Fuse, Mr. H. Imai and Mr. M. Sameshima. 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] Ken Lunde, "Understanding Japanese Information Processing", O Reilly Associates, Inc. 1993