Intel-Hex to VHDL Converter
HEX2VHD

User manual

Project acronym:	HEX2VHD
Main author:	Aitzol Zuloaga
Date:	05/02/2012

Ingeniarien Goi Eskola
Escuela Superior de Ingenieros
Bilbao

eman ta zabal zazu
Universidad del País Vasco
Euskal Herriko Unibertsitatea

DOCUMENT CHANGE CONTROL

Date	Author	Description of Change
30/10/2011	Aitzol Zuloaga	Initial document

CONTENTS

1 ABSTRACT

The HEX2VHD is an application to extract information from an Intel-Hex Format File and carry it to a user developed VHDL description. In the default package, contains a standard VHDL description of ROM memory to be used as template for ROM creation from an Intel-Hex Format File. It can be used to create program memories for processor cores from third party compiler results. Also, it can be used for Verilog descriptions but it had not been tested.

2 ABOUT HEX2VHD

The HEX2VHD conversion tool is an application to extract information from a file with Intel-Hex Format and carry it to a user developed VHDL description. The Intel-Hex Format is widely used file format to exchange the binary information to program memories for processors and other applications.

Nowadays, the systems are implemented in FPGA devices, and memories are frequently implemented inside devices as a part of the whole system. The development of these systems is done using Hardware Description Languages (HDL) like VHDL or Verilog. But many times, the development of the information that will be used in a system memory is done by other applications as compilers and constant generators. One of the most common output file of these systems is the Intel-Hex Format.

One of the problems to translate from Intel-Hex Format to an HDL is the variety of ways to describe a ROM in such languages. By this reason the HEX2VHD uses an HDL template as a guide to generate the final HDL file.

With these constrains, the HEX2VHD requires to work with four (4) files:

- The **Intel-Hex file** that contains the binary information to be translated to the final HDL file.
- The **Template file** that contains the user HDL description of the ROM to be used in the system.
- The **Target file** that will be the final output file in HDL form with the information obtained of Intel-Hex file
- The **Configuration file** that contains the keywords used in the Template file to be translated by the program to obtain the Target file.

2.1 Intel-Hex format file

The Intel-Hex is a file format that contains binary information to be stored in a memory. This binary information is formatted in a text file using some readable ASCII characters. These characters are, the colon ':', the numbers ('0' to '9'), the characters to represent the hexadecimals numbers ('A' to 'F') and the line-end characters (CR and LF).

The information is fragmented in "records" or "text lines". Each record starts with a colon ':' and ends with line-end characters (CR and LF). Any other information in record is expressed with pairs of hexadecimal digits ('0' to '9', 'A' to 'F').

Each record is divided in 6 parts:

- **Start code,** one character, the colon ':'.
- **Byte count**, two hex digits, the number (N) of bytes (hex digit pairs) in the data field
- **Address**, four hex digits, the 16-bit address in big-endian format of the beginning of the memory position for the data.
- **Record type**, two hex digits, 00 to 05, the type of the data field.
- **Data**, a sequence of N bytes of the data, represented by 2N hex digits.
- **Checksum**, two hex digits, the checksum of the N bytes of the data.

There are six record types in an Intel-Hex file:

- **00, data record**, contains data and 16-bit address.
- **01, End Of File record**, used to mark the final of the file. It is used once per file in the last line of the file. Usually the complete record is ':00000001FF'.
- **02, Extended Segment Address Record**, segment-base address (two hex digit pairs in big endian order). Used when 16 bits are not enough to specify the address of the memory position. HEX2VHD support this record only for a 0000 value for segment address.
- **03, Start Segment Address Record,** specifies the initial content of the CS:IP registers for 80x86 processors. Not supported by HEX2VHD.
- **04, Extended Linear Address Record**, allows fully 32 bit addressing (up to 4GiB). Not supported by HEX2VHD.
- **05, Start Linear Address Record,** specifies the initial content of the EIP register for 80x86 and higher processors. Not supported by HEX2VHD.

The HEX2VHD tools is intended for small the memories used inside FPGAs, in this base it only supports address in the range of 64KiB.

2.2 Template file

The Fig. 1 shows a typical VHDL description of a small (16 x 8) ROM memory. The lines that follows the "`constant rom : rom_type:=(`" contains the values of data to be placed in each position of the ROM memory.

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity rom16x8 IS
  port (
    signal adr    :in  std_logic_vector(3 DOWNTO 0);
    signal dat    :out std_logic_vector(7 DOWNTO 0));
end rom16x8;

architecture behavioral of rom16x8 is
  type rom_type is array (0 to 15) of std_logic_vector (7 downto 0);
  constant rom : rom_type:=(
    X"A0",
    X"20",
    X"10",
    X"B2",
    X"F8",
    X"03",
    X"2E",
    X"75",
    X"03",
    X"B8",
    X"93",
    X"CD",
    X"AC",
    X"A3",
    X"B2",
    X"20");
begin

  dat_o <= rom(conv_integer(adr));

end behavioral;
```

Fig. 1. Typical VHDL description of a ROM memory block.

The objective of HEX2VHD is to take the information directly of the hex file and put it in the correct places in the HDL description. As it said, to do this, HEX2VHD requires a template file with special key-words correctly placed to be substituted by the information obtained of the hex file. All keywords used in the template file must be enclosed curly brackets (‘{‘ ‘}’) as shows Fig. 2. The keywords must be defined in the Configuration file as it is explained in the following section.

```
------------------------------------------------------------
--  {NAME}
--  Date: {DATE}
------------------------------------------------------------

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity rom16x8 IS
  port (
    signal adr    :in  std_logic_vector(3 DOWNTO 0);
    signal dat    :out std_logic_vector(7 DOWNTO 0));
end rom16x8;

architecture behavioral of rom16x8 is
  type rom_type is array (0 to 15) of std_logic_vector (7 downto 0);
constant rom : rom_type:=(
  X"{INIT_00}",
  X"{INIT_01}",
  X"{INIT_02}",
  X"{INIT_03}",
  X"{INIT_04}",
  X"{INIT_05}",
  X"{INIT_06}",
  X"{INIT_07}",
  X"{INIT_08}",
  X"{INIT_09}",
  X"{INIT_0A}",
  X"{INIT_0B}",
  X"{INIT_0C}",
  X"{INIT_0D}",
  X"{INIT_0E}",
  X"{INIT_0F}");
begin

  dat_o <= rom(conv_integer(adr));

end behavioral;
```

Fig. 2. VHDL template file of a ROM memory block.

2.3 Configuration file

The Configuration file contains all the keywords that HEX2VHDL will substitute in the template file to obtain the Target HDL file.

The Configuration file is a text file that contains one keyword on each line (Fig. 3). In its basic format, each keyword represents a “position” of memory in the same order that was found in the Configuration file. The file shown in the Fig. 3 uses INIT_00 for the position 0, INIT_01 for the position 1, and so on. In this basic format, each position represents a byte.

```
INIT_00
INIT_01
INIT_02
INIT_03
INIT_04
INIT_05
INIT_06
INIT_07
INIT_08
INIT_09
INIT_0A
INIT_0B
INIT_0C
INIT_0D
INIT_0E
INIT_0F
```

Fig. 3. Basic Configuration file for HEX2VHD.

But the configuration file can be more complex and useful. For HEX2VHD there are 3 types of keywords:

- The data keyword.
- The user constant keyword.
- The configuration parameter keyword.

The previously explained basic configuration file contains exclusively *data keywords*. Each of these keywords represents a position of the memory and their substitution value is obtained from Intel-Hex file.

The *user constant keyword* allows the definition of user defined strings (Fig. 4) that will be substituted in the template file. These keywords are followed by a comma ‘,’ and the substitution string.

```
NAME,Receiver
DATE,JAN 2011
INIT_00
INIT_01
INIT_02
INIT_03
INIT_04
INIT_05
INIT_06
INIT_07
INIT_08
INIT_09
INIT_0A
INIT_0B
INIT_0C
INIT_0D
INIT_0E
INIT_0F
```

Fig. 4. Configuration file for HEX2VHD using *user constant keywords*.

The *configuration parameter keyword* has a similar writing form of the *user constant keyword* but it uses a set of reserved words to define some operational parameters of the HEX2VHD tool (Fig. 5). These keywords are the following:

- **RADIX**: defines the numerical representation of the data in the Target file. Its values can be **BIN** (binary) or **HEX** (Hexadecimal). The default form is hexadecimal. It is important to remark that VHDL doesn't allow hexadecimal representation of data for std_logic_vector with sizes non-multiples of 4. In these cases binary form must be used.
- **BITS**: defines the number of bits used to represent values in the HDL Target File. This keyword it is more useful when you use binary radix format.
- **NIBBLES**: defines the number of nibbles used to represent values in the HDL Target File. This keyword it is more useful when you use hexadecimal radix format.

```
RADIX,HEX
BITS,8
NAME,Receiver
DATE,JAN 2011
INIT_00
INIT_01
INIT_02
INIT_03
INIT_04
INIT_05
INIT_06
INIT_07
INIT_08
INIT_09
INIT_0A
INIT_0B
INIT_0C
INIT_0D
INIT_0E
INIT_0F
```

Fig. 5. Configuration file for HEX2VHD using *configuration parameter keywords.*

2.4 HDL Target file

The HDL Target file is created by the HEX2VHD tool. It is a copy of the Template file with all keywords enclosed by curly brackets substituted (Fig. 6).

```
-------------------------------------------------------------
--  Receiver
--  Date: JAN 2011
-------------------------------------------------------------

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity rom16x8 IS
  port (
    signal adr    :in  std_logic_vector(3 DOWNTO 0);
    signal dat    :out std_logic_vector(7 DOWNTO 0));
end rom16x8;

architecture behavioral of rom16x8 is
  type rom_type is array (0 to 15) of std_logic_vector (7 downto 0);
constant rom : rom_type:=(
  X"FF",
  X"FF",
  X"20",
  X"12",
  X"D0",
  X"06",
  X"20",
  X"12",
  X"20",
  X"12",
  X"20",
  X"12",
  X"20",
  X"12",
  X"20",
  X"12");
begin

  dat_o <= rom(conv_integer(adr));

end behavioral;
```

Fig. 6. Final VHDL description of a ROM memory block.

3 USING HEX2VHD

The HEX2VHD is designed to be used as a command-line program. The user can associate it to an *Integrated-Development-Environment* (IDE). The usage format as a command-line program is:

HEX2VHD <hex_file> <template_file> [<target_file> [<config_file>]]

The first parameter <hex_file> is the name of the Intel-Hex file with its extension (normally .hex).

The second parameter <template_file> is the name of the HDL file with its extension.

The third parameter <target_file> is the name of the Target file with its extension (normally .vhd). This parameter is optional, if it is omitted, the target file will have the same name as hex file but with .vhd extension. Also, if it is omitted, the next parameter must be omitted too.

The fourth parameter <config_file> is the name of the Configuration file with his extension. This parameter is optional, if it is omitted, the "hex2vhd.txt" file will be used.

In the present version of HEX2VHD, the program and all involved files must be located in the same directory.

3.1 Errors and Warnings

The error and warning messages displayed by the HEX2VHD program are the following ones:

`Incorrect number of parameters. Usage:` `hex2vhd <hex_file> <template_file> [<target_file> [<config_file>]]`
This error is displayed when the program was invoked with no parameters or only one parameter. The minimum number of parameters is two: <hex_file> <template_file>.

`Hex file doesn't exists`
This error is displayed when the indicated Intel-Hex file can not be found.

`Template file doesn't exists`
This error is displayed when the indicated Template file can not be found.

`Configuration file doesn't exists`
This error is displayed when neither the indicated Configuration file nor the default Configuration file can be found.

`Warning: Start address not supported in hex-file`
This warning is displayed when the Intel-Hex file contains a type 03 record. The record will be ignored.

`Warning: Record not supported in hex-file`
This warning is displayed when the Intel-Hex file contains a type 04 or type 05 record. The record will be ignored.

`Warning: keyword not found`
This error is displayed when a keyword used in the Template file was not defined in the Configuration file. The current place in the Target file will be erased and the warning message will be written in the Target file as a VHDL comment.

4 BIBLIOGRAPHY

[1] Intel Corporation. “Hexadecimal Object File Format Specification”. January 1988.

[2] Xilinx Incorporated. “Using Block RAM in Spartan-3 Generation FPGAs”. XAPP463. March 2005.