VHDL Made Easy

By BorjaLive (B0vE)

VME Parser Engine v1.2.x

Índice

Descripción

VHDLME o, para abreviar, VME es un lenguaje de descripción de hardware intermedio con una sintaxis orientada a scripts.

El código en VME debe ser traducido primero a VHDL para después ser compilado. Las herramientas ayudas sintácticas están disponibles para Notepad++ y el compilador se distribuye en dos versiones, con y sin interfaz gráfica.

Esta idea surgió a partir de haber observado por primera vez VHDL. En comparación con otros lenguajes de programación convencionales, aparenta una tipología demasiado estricta y una sintaxis enrevesada, compleja y, en algunos momentos, inconsistente; teniendo en cuenta que se trata, no de un lenguaje propiamente, sino de una notación como lo podría ser JSON.

Dos personas llegaron a esta conclusión.

VME viene a simplificar los lenguajes de descripción de hardware aportando una mayor flexibilidad y legibilidad al código.

Actualmente se encuentra escrito en AutoIt3, un lenguaje de scripting para Windows; nada aconsejable para proyectos de esta índole. Decisiones del creador.

Notas de la v1.2.x

Este manual de usuario está escrito para la versión 1.2 LTS y, por tanto, puede ser incompatible con funciones futuras del lenguaje. Bastantes cambios que rompen códigos escritos para la 1.1 LTS han sido introducidos.

Ya puedes dejar de leer, el resto de esta página es relleno.

La versión del lenguaje viene definida por el número de iteración del motor. Dado que VME sigue el modelo de aproximaciones sucesivas, nunca existirá una versión definitiva ni una versión concreta; sino un conjunto discreto de estados caracterizados por el numero de veces que se ha reiniciado el desarrollo (X), numero de veces que se han hecho cambios de diseño (Y), número de veces que el creador pensó que había solucionado todos los errores conocidos (Z). Dando como resultado X.Y.Z A diciembre de 2018 la versión más reciente es la 1.1.1, pero se espera que antes de 2019 se lance la 1.1.2 y 1.2 Beta.

Las versiones Beta no reciben actualizaciones, mientras que las LTS tampoco las reciben. La denominación Beta o LTS ha sido escogida para evitar llegar nunca a una Release.

Sintaxis

Inspirada en C y adaptada a las peculiaridades de VHDL.

VME no distingue mayúsculas y minúsculas. No utiliza ‘;’ para cerrar líneas. La comprobación de igualdad y la asignación son iguales “=”

Parámetros de par ser

Son llamadas directas al indentador de líneas, estas no se ven reflejadas directamente como instrucciones (aunque pueden variar la salida).

Hay una división entre las que simplemente definen variables globales en el proceso de compilación, y las que realizan funciones más avanzadas. Empezando por las primeras, son:

Para pasar datos se usa el siguiente formato

```
Parametro Valor1 Valor2 … ValorN
```

Entity

Define el nombre que tomará la entidad en el código VHDL. Por defecto es “noname”

Architecture

Define el nombre que tomara la arquitectura en el código VHDL. Por defecto es “noname_architecture”

Enginer

Define el nombre del creador del código, saldrá en la cabecera del archivo generado. Por defecto es el nombre del usuario en el ordenador.

Description

Define la descripción que saldrá en la cabecera del archivo generado. Por defecto está vacía.

Los dos siguientes parámetros se usan para crear tipos extra de variables.

TypeDef

Renombra un tipo de dato.

```
TypeDef TipoDeDato NuevoNombre
```

State

Crea variables con un valor identificativo

```
State TipoDeDato Estado1 Estado2 … EstadoN
```

Define

¿Conoces el #Define de C? pues es más de lo mismo. Busca en todo el documento la palabra original y la remplaza por la otra.

```
Define Busqueda Remplazo
```

Secciones

Se definen con limitadores de inicio y fin; adicionalmente pueden incluir un modificador en el limitador de apertura.
Su uso no es obligatorio; no obstante, no está permitido alternar el uso de una sección en un mismo fichero. Es decir, si usas una sección para delimitar cierta porción de código que comparte la misma función, no podrás escribir ese mismo tipo de sentencia fuera de las secciones.

Var … VarEnd

No acepta modificadores.

Se utiliza para declarar las variables. Estas pueden ser puertos o señales. Más sobre variables en PAG X.

Logic [MOD] … LogicEnd

Acepta los modificadores “parallel” y “sequential”

Contiene todo tipo de sentencias excepto declaración de variables. Al usarse sin modificadores, admite tanto código paralelo como secuencial, al agregar los modificadores estos restringen las sentencias al tipo que definen.

Nota: Si no se usan secciones, el compilador ignorará todas las líneas que no pueda parsear; si se usan, y la línea está dentro de las secciones, pero no puede parsearla, lo mostrará como un error.

Implement … ImplementEnd

No acepta modificadores

Obligatoria para asignar los componentes de la FPGA con los puertos de la entidad. NOTA: Solo se utilizarán si se le pide al compilador que genere las restricciones.

Declaración de variable

Los arrays se declaran como el tipo de dato que contienen acompañado de su tamaño pasado como argumentos. Los argumentos son opcionales.

```
TipoDeDato[Argumentos] Nombre
```

Asignaciones

Sintácticamente se realizan igual que en la mayoría de lenguajes de alto nivel.

```
Variable = Expresion
```

Conversiones

Simulan las conversiones explicitas de C

```
VariableConvertida = (Tipo) VariableAconvertir
```

Los tipos a convertir son:

(integer)

(binary[x])

Dónde 'x' es el tamaño del nuevo array.

Funciones

Las funciones de VME son una simplificación de las funciones basadas en módulos de apoyo conocidos. El compilador sabe que realizan las funciones, gracias a ello puede remplazarlas con expresiones incluidas en IEEE para no depender de otras librerías. Su sintaxis es:

```
Función Argumento1 Argumento2 … ArgumentoN
```

Las funciones permitidas son las siguientes:

LCDF: ANDz, ORz, NANDz, NORz, XORz, XNORz

Dónde 'z' es el número de entradas de la puerta lógica. La última variable siempre será la salida.

Estructuras concurrentes

VME cuenta con las cuatro estructuras que surgen de: asignar directamente un valor o ejecutar una expresión y comprobar una igualdad o una expresión.

NOTA: En VHDL estas estructuras exigen que el último caso involucre a todas las otras opciones, lo que es lo mismo, sea un "Else". VME no es tan rígido, si detecta que una estructura es ambigua, automáticamente asignará al último caso la condición "ELSE" y lo notificará con un Warning.

SetIf (When … Else)

Realiza una asignación múltiple evaluando expresiones.

```
Set F
Valor1 If Expresion1
Valor2 If Expresion2
Valor3 Else
```

ValorX puede ser una variable o un literal.

Es una estructura paralela.

SetSwitch (With … Case)

Realiza una asignación múltiple evaluando igualdades.

```
Set F Switch Variable
Case Expresion1a|Expresion1b: Valor1
Case Expresion2: Valor2
Else: Valor3
```

ValorX puede ser una variable o un literal.

Es una estructura paralela.

NOTA: VHDL no permite realizar esta estructura para comprobar cadenas, si VME detecta esto lo convertirá a un SetIf y lo mostrará con un Warning.

IfThen (If … Then)

Realiza diferentes instrucciones evaluando expresiones.

```
If Expresion1 Then Instruccion1
Else If Expresion 2 Then Instruccion2
Else Instruccion3
[EndIf]
```

Es una estructura secuencial.

Si “ExpresiónN” es “clock” esta se remplazará por el código necesario para comprobar que el reloj está en un flanco de subida.

SwitchCase (Switch … Case)

Realiza diferentes instrucciones evaluando igualdades.

```
Switch Variable
Case Valor1a|Valor1b: Instrucción1
Case Valor2: Instrucción2
Else: Instruccion3
[EndSwitch]
```

Es una estructura secuencial.

Si se acompaña la cabecera con “Set” y el nombre de una variable, se asumirá que la operación deseada es asignar un múltiple valor a la misma variable, por ello “InstrucciónN” deberán ser posibles valores para la variable que precede a “Set”

? (If … Then)

El operador ternario tiene una tradución directa a su omólogo en estructura IfThen. Puede usarse para esclarecer el documento.

```
Variable = Expresion ? Valor1 : Valor2
```

Es una estructura secuencial.

Recibe el mismo tratamiento que:

```
If Expresion Then Variable = Valor1
Else Variable = Valor2
[EndIf]
```

Remarks: “EndIf” y “EndSwitch” están ahí, pero no son necesarias.

Estructuras cíclicas

ForNext (For)

Ejecuta repetidamente un bloque de estructuras. Define un contador, no modificable, para contar los ciclos.

```
For contador In Numero
        Instrucciones …
Next
```

```
For contador In Variable
        Instrucciones …
Next
```

Si el límite se define mediante una variable, esta deberá ser un vector y se iterará hasta el número elementos de la variable.

Operadores

Para la asignación se usa '=' el sentido es de derecha a izquierda.

Para comprobar la igualdad también se usa '='.

Los operadores lógicos son "And", "Or" y "Not", "Nand", "Nor", "Xor", "Xnor". Estos operadores son equivalentes a su versión como funciones implícitas de VME.

Para concatenar binarios, generalmente en una cadena, se usa '&'.

Implementación

Para asignar elementos IO de una FPGA a las entradas y salidas.

LED?

Diodos led, el rango va desde LED1 hasta LED8.

LEVER?

Interruptores SW, desde LEVER1 hasta LEVER4.

LCD?

Control de caracteres en el display LCD, desde LCD1 hasta LCD7.

BTN?

Pulsadores, desde BTN1 hasta BTN4.

JACK?!

Conectores J1 y J2, de HEADER1A a HEADER1D y HEADER2A a HEADER2D.

STICK? y STICC

Botones rotatorios, desde STICK1 a STICK2 y STICC.

Elemento Variable

Guía práctica

Uso del compilador

El parser de VME es un freeware, open source, bajo licencia GNU GPL. Por tanto, el código fuente puede ser descargado libremente y su distribución o modificación se regulan bajo Copyleft.

Cualquiera con unos conocimientos mínimos de Autoit3 o con experiencia en la programación podría implementar el parser en una interfaz personalizada; pero si estas buscando una forma sencilla de compilar un ".vme" sin quebrarte la cabeza, ya he desarrollado dos versiones.

Línea de comandos: el binario puede ser llamado desde un símbolo del sistema. Los parámetros del compilador se pasan como argumentos.

Interfaz Gráfica: Tan sencillo como seleccionar la fuente y pulsar el botón grande que dice "Compilar"

Parámetros del compilador

NoHeader

Desactiva la cabecera por defecto que se escribe en el archivo de salida.
En línea de comandos es "--noHeader"

LibStrict

Fuerza el uso excluyente de las librerías IEEE, si se usaran elementos de otras librerías, estos se implementaran de otra forma.
En línea de comandos es "--libStrict"

Verbose

Muestra información detallada de las actividades del compilador. En línea de comandos es “--verbose” o “-v”

Silent

Evita crear cuadros de dialogo emergente. No mostrará avisos y, si lo necesita, tomara decisiones por si mismo.
En línea de comandos es “--silent” o “-s”

Implement

El parser generará las líneas necesarias para montar el circuito en una FPGA Xilinx Spartan 3E. Se requiere especificar los elementos IO. En línea de comandos es “--implement”

Bypass

El parser evitará comprobar expresiones, identificadores, funciones y otros campos que; de no tener una sintaxis correcta, podrían causar excepciones incontroladas en VME PE o errores sintácticos en el archivo VHDL final. En líneas de comandos es “--bypass” o “-f”

IDE

Actualmente VME no cuenta con IDE propio ni se espera que lo tenga.

Se recomienda usar Notepadd++ con el lenguaje definido por usuario proporcionado en la carpeta de recursos.

Si hay algún desarrollador frontend interesado, se aceptan propuestas.

La línea del error

Aunque VME puede detectar la línea en la cual se encuentra el error, lo cierto es que el compilador funciona de una forma tan interdependiente y destartalada, que los errores pueden ser descubiertos a la hora de la escritura y, al estar sintetizado el documento, a veces es imposible dar con la línea exacta.

Por lo general, si el error se encuentra en la cabecera de una estructura, se culpará a la última línea de la estructura.

Por si acaso, no te fíes demasiado de la línea que indique.

Errores y versiones

Bugs conocidos

Errores tan recurrentes que merecen nombre propio

Derrape: Cuando se recorren las líneas de un archivo y se encuentra una estructura de varias líneas, el bucle que avanza dentro de la estructura a veces no comprueba si es el final del archivo. Como resultado; si una estructura de varias líneas acaba justo en el final del documento, arrayOutOfBoundsException al canto. Derrapa.

Tracking Lost: Cuando las funciones de estandarización se pasan estandarizando cierto texto y acaban por simplificarlo demasiado. Suele darse al eliminar espacios o interiores.

Reporte de errores

El desarrollador está abierto a mensajes que informen de posibles errores en el código del compilador. (Así al menos parece que hay personas interesadas en VME y el desarrollador no piensa que su trabajo es un vano)

NOTA: En la versión actual, 1.2.0, prácticamente se descubren nuevos errores y bugs con cada nuevo archivo que se prueba. Es de esperar que queden pequeños trucos sintácticos que provocan el fallo del compilador y no son detectados a tiempo.

Control de versiones

En el repositorio de GitHub siempre estará la versión más reciente del código fuente y los últimos binarios disponibles.

En la página de SourceForge solo se publican las releases más estables.

Los binarios se distribuyen compilados para arquitecturas de 32 y 64 bits, con y sin compresión UPX.

Teóricamente VME puede ser ejecutado desde Windows 95 hasta en Windows 10. No se esperan portes a otros sistemas operativos.

Historial de cambios

1.2.0 ak 1.2 LTS

+Agregado el operador ternario '?', asignaciones SetIfElse en una sola linea.

+Agregado soporte para modificadores de variable. I.E.: "Const" para "Constant" (Ya estaba, pero ahora es util)

+Agregadas dos instrucciones nuevas, "Sleep" y "Wait", las dos son sequential only.

+Agregado nuevo parametro para el parser. "Define" remplaza una cadena por otra en todas las lineas.

+Agregado nuevo parametro para el parser. "TypeDef" permite ponerle renombrar un tipo de variable. (Nativo en VHDL)

+Agregado nuevo parametro para el parser. "State" crear estados autonumerados. (Nativo en VHDL)

+Agregado soporte para insertar parametros del parser. ("Define","TypeDef","Entity","Architecture","Enginer","Description")

*Se ha cambiado la sintaxis de la instrucción SwitchCase.

*Cambiada la sintaxis del SetIf y SetSwitch

*Ahora la instrucciones SetIf y SwitchCase son recursiva. Sus instrucciones condicionadas pueden ser cualquier otra instrucción.

*Ahora la cabezera escribe automaticamente el nombre del ingeniero, basado en el usuario del ordenador.

*Ahora la expresión "Clock" es válida en instrucciones secuenciales y será remplazada por el uso correcto del reloj.

*Corregidos errores de cierres inesperados con la Interfaz gráfica.

*El elemento de implementación "HEADER" ha sido renombrado a "JACK"

META: *Se ha intentado simular encapsulacion, de algun tipo, para ayudar a los desarrolladores que quieran crear su propio script de compilacion.

META: *Ahora las definiciones de ENTITY y ARCHITECTURE se hacen mediante parametros del parser.

-Eliminado el modificador "Set" en el "SwitchCase"

-Se descarta el soporte para módulos y el sistema de Includes.

-Se han eliminado las notas de versión referentes a la 1.0 y 1.1 de este documento.