IBIS-AMI DLL/SO File Checking

16 Feb 2016

Currently the only check for IBIS-AMI DLL/SO files is E4616 “Code file %s not found. It was defined in [Algorithmic Model] for Model %s”. The following checks will be added:

### DLL/SO Machine Architecture

The DLL/SO file must be suitable for the machine architecture and address bit width designated by first argument of the Executable subparameter. It is acceptable to check only those files of the machine architecture and address bit width of the currently running IBISCHK6 or later executable. It should be possible to accomplish this check by loading the file and reading its symbol table.

An ERROR message must be given where a DLL/SO file not matching the stated machine architecture or the stated bit width is referenced. Example: “Code file %s is not loadable as a %d bit %s object” (filename, bits, OS).

### DLL/SO Function Existence

Each DLL/SO file must contain code symbols for exported functions in one of four possible combinations:

1. Case 1: Executable model file has AMI\_Init, AMI\_GetWave and AMI\_Close. (IBIS 5.0 and above)
2. Case 2: Executable model file has AMI\_Init and AMI\_Close. (IBIS 5.0 and above)
3. Case 3: Executable model file has AMI\_Resolve, AMI\_Resolve\_Close, AMI\_Init, AMI\_GetWave and AMI\_Close. (IBIS 6.0 and above)
4. Case 4: Executable model file has AMI\_Resolve, AMI\_Resolve\_Close, AMI\_Init and AMI\_Close. (IBIS 6.0 and above)

These cases can be checked according to the following rules, which require parsing the AMI file associated with each DLL/SO file:

1. DLL/SO file must export AMI\_Init() and AMI\_Close() functions in symbol table. Example ERROR messages:  
   “Code file %s does not contain required AMI\_Init() function”  
   “Code file %s does not contain required AMI\_Close() function”
2. If the corresponding AMI file contains GetWave\_Exists=True, DLL/SO file must export AMI\_GetWave() function in symbol table. Example ERROR message:  
   “Code file %s does not contain AMI\_GetWave() function, required because GetWave\_Exists=True in AMI file %s”
3. If corresponding AMI file contains Resolve\_Exists=True, DLL/SO file must export the AMI\_Resolve() and AMI\_Resolve\_Close() functions in its symbol table. Example ERROR messages:  
   “Code file %s does not contain AMI\_Resolve() function, required because Resolve\_Exists=True in AMI file %s”  
   “Code file %s does not contain AMI\_Resolve() function, required because Resolve\_Exists=True in AMI file %s”

Failure to conform to any of the above must result in an ERROR message. It is acceptable to check only those files of the machine architecture and address bit width of the currently running IBISCHK6 executable. It will be necessary to examine the AMI files associated with all IBIS file Executable, Executable\_TX, and Executable\_RX lines within an [Algorithmic Model] section to find the AMI file(s) associated with each DLL/SO and determine the status of GetWave\_Exists and Resolve\_Exists.

Checks for the presence of exported code symbols can be implemented simply by using the same code used by EDA tools to find symbols in the DLL/SO files. For example, dlopen() and dlsym() on Linux, and LoadLibrary() and GetProcAddress() on Windows.

A Linux example code fragment:

#include <dlfcn.h>  
…  
 char \*msg = dlerror(); // To reset.  
 void \*dll = dlopen( file\_name, RTLD\_NOW );  
 msg = dlerror();  
 void \*func = dlsym( dll, func\_name );  
 msg = dlerror();

A Windows example code fragment:

#include <windows.h>  
…  
 void \*dll = LoadLibrary ( file\_name );  
 void \*func = GetProcAddress((HMODULE)dll, func\_name );

### Function Call Testing

Once loading of the DLL/SO files has been accomplished the next step is to give each one a quick test. The test sequence is as follows:

1. AMI\_Init is called with test data:

long AMI\_Init (

double \*impulse\_matrix, hard-coded array of values

long number\_of\_rows, calculated for 4 UI

long aggressors, 1 aggressor or Max\_Aggressors

double sample\_interval, calculated for 32 and 128 samples/bit

double bit\_time, calculated for 6.25Gbps

char \*AMI\_parameters\_in, String created from AMI parameters

char \*\*AMI\_parameters\_out, Pointer to pointer, will be set by DLL

void \*\*AMI\_memory\_handle, Pointer to pointer, will be set by DLL  
char \*\*msg Pointer to pointer, will be set by DLL

)

The IBIS Open Forum will provide data for impulse\_matrix. The number\_of\_rows

### Platform Information

For IBIS files containing one or more [Algorithmic Model] keywords, some means must be provided to summarize Executable subparameter DLL/SO files and their compatibility with the running IBISCHK6 O/S and/or bit width. This can be reported at the end of execution, and does not need to list the handling of each unique DLL/SO file. This will also serve to report the platform sets supported by the models. For example:

NOTE: Status of [Algorithmic Model] Executables for Windows 64:  
 ventura\_tx\_32.dll Windows 32: NOT CHECKED  
 ventura\_tx\_64.dll Windows 64: 0 ERRORS, 0 WARNINGS  
 ventura\_tx\_32.so Linux 32: NOT CHECKED  
 ventura\_tx\_64.so Linux 64: NOT CHECKED  
NOTE: This IBISCHK6 executable supports Windows 64 bit only.

The above is a sample report for IBISCHK6 running on Windows 64. The unique set of DLL/SO files found in the IBIS file are listed once each, there is no need to print this list for each [Model].