Non-Member Contribution to Study Group 12

Source: Signalogic, Inc.
Title: G.191 (aka STL, or Software Tools Library) Upgrade for Server Platforms
Contact: Jeff Brower, Signalogic, jbrower@signalogic.com

Abstract The ITU STL (Software Tools Library) BASOP modules, originally conceived in the 1990s, should be upgraded to meet thread-safe and malware resilient operation required by modern server platforms and software architecture.

Background

The ITU STL (Software Tools Library) BASOP modules were written in the 1990s to provide a uniform set of fixed-point arithmetic "basic operators" (e.g. add, multiply, shift) that supported varying data widths, saturation arithmetic, possible overflow, possible carry, and minimal error handling. In those years codecs were typically implemented on special-purpose DSP (digital signal processor) devices with limited, if any, exposure to third-party software and networks. Nearly 30 years later, codecs tend to be implemented on general-purpose x86 and Arm CPUs running in complex software environments with numerous local and Internet network connections – and, not surprisingly, production requirements have changed substantially. Coding techniques that were common in those years, such as global variables and application termination upon error, are no longer acceptable. Now the need is for robust, error and malware resilient, thread-safe operation suitable for production applications that typically run on server platforms (e.g. private or public cloud).

Additional, detailed background explanation is provided in SG12-C-0170, submitted by Fraunhofer HHI & IIS and Telefon AB - LM Ericsson [7].

Proposal

To meet the current need, several STL BASOP module modifications are proposed, as listed below.

Global Variable Handling (Thread-safe Operation)

1) The following preprocessor macros should be defined:

NO_BASOPS_OVERFLOW_GLOBAL_VAR to remove references to the Overflow global variable

NO_BASOPS_CARRY_GLOBAL_VAR to remove references to the Carry global variable

2) Similar to the recommendation in [6] and [7] by Fraunhofer IIS and Telefon AB-LM Ericsson, the following functions should be replicated with an "_ovf" suffix and accepting an Overflow parameter:

add_ovf, sub_ovf, shl_ovf, L_mult_ovf, L_add_ovf, L_sub_ovf, L_shl_ovf, L_add_c_ovf, L_sub_c_ovf, L_sat_ovf

For example, add_ovf would be defined:

```
Word16 add_ovf (Word16 var1, Word16 var2, Word16* Overflow);
```

and updated codecs would (i) set NO_BASOPS_OVERFLOW_GLOBAL_VAR and (ii) call add_ovf() as necessary in places where their code checks for add result overflow, passing an &Overflow parameter and checking its result

3) As described in [6] and [7] by Fraunhofer IIS and Telefon AB-LM Ericsson, asserts should be added to existing functions to let codec developers know when overflow and/or carry is occurring and the new _ovf() variant should possibly be used. Such asserts should be controlled by ENABLE_BASOPS_OVERFLOW_ASSERT and ENABLE_BASOPS_CARRY_ASSERT preprocessor macros

Error Handling

The following preprocessor macros should be defined:

1) NO_BASOPS_EXIT to avoid exit() and abort() calls

2) ENABLE_BASOPS_ERROR_DISPLAY to enable detailed error message display in error conditions, for example

```
fprintf(stderr, "Division error in div_s in basop32, var1 = %d var2 = %d \n", var1, var2);
```

   indicating (i) the function call and module in which the error occurred, and (ii) all parameters relevant to the error

3) ENABLE_BASOPS_ERROR_LOGGING to enable logging to file of error messages

4) ENABLE_BASOPS_ERROR_USER_FUNC, which if defined, enables a call to a user-defined function that allows codec vendors and users to define application specific error handling behavior. For example, to handle a division-by-zero error the application might set the function result to the maximum possible data width value and allow the function to continue. Currently the STL error messages are minimal, not indicating which function caused the error, and inflexible, forcing a total application exit

Backwards Compatibility

#defines should be set in the Makefiles of updated codecs (or in an appropriate header file, for example “options.h” present in the EVS codec). If not set, then older codec versions using the updated STL will continue to function as-is, with no changes. This provides full backwards compatibility with older codec versions

Fine-Grained Control

Fine-grained control is preferred for development and debug/test purposes. For example, the Overflow global variable can be removed separately from the Carry global variable

Current Work Status and Next Steps

On the STL Github site [2], the Signalogic pull-request [5] includes the above changes, with the following work still needed:

1) Polarity of preprocessor macros should be reversed to meet the backwards compatibility requirement. For example, USE_BASOPS_OVERFLOW_GLOBAL_VAR should be changed to NO_BASOPS_OVERFLOW_GLOBAL_VAR

2) The functions add, L_add_c, L_sub_c, and L_sat need _ovf equivalents

3) Asserts as described in 4) under Global Variable Handling above should be added

4) Application specific error handling that occurs upon error and avoids application exit should be improved, as mentioned in 3) under Error Handling above. The current pull-request has some error continuation behavior, but not a user-defined function option

All STL updates should be made available on the OpenITU Github site in the form of Pull Requests.

IVAS Support

STL changes should be take into account new codec development, in particular IVAS [3] [4].

Preprocessor Macros Notes

1) "Preprocessor macros" as used here is interchangeable with "#defines". They may be implemented as #define statements in C code and/or "-D" fields in the Makefile gcc command line

2) Additional preprocessor macros may be needed, after group discussion and feedback. For example, the Signalogic Github pull-request [5] includes EXCLUDE_BASOPS_NOT_USED, which is used to exclude basop functions not used in the EVS codec. Possibly a similar mechanism should be created that is flexible and usable by other codecs

References

[1] Recommendation ITU-T G.191 (03/2023), *Software tools for speech and audio coding standardization*

[2] OpenITU Github project, https://github.com/openitu/STL

[3] 3GPP Forge project, https://forge.3gpp.org/rep/ivas-codec-pc/ivas-codec, *IVAS Codec Public Collaboration*

[4] 3GPP TS 26.258, Codec for Immersive Voice and Audio Services - C code (floating point)

[5] Signalogic Pull Request, https://github.com/openitu/STL/pull/171, *threadsafe basop32 and enh40 files, with control over global variables and exit/abort*

[6] Fraunhofer IIS Pull Request, https://github.com/openitu/STL/pull/173, *Remove global flags under BASOP_NOGLOB*

[7] “G.191 Basic Operator Improvements”, ITU SG12-C-0170 Contribution, Stefan Döhla Fraunhofer HHI & IIS, Erik Norvell Telefon AB-LM Ericsson.