SILOS/Harmony における Verilog (PLI)インタフェースの使用 (Windowsの場合)

はじめに

IEEE 1364 Verilog標準には多数の標準システムタスクとシステム関数が用意されていますが、カスタマイズされたシステムタスクやシステム関数をベンダから提供されて使用したり、PLI (Programming Language Interface)ルーチンを使用して設計者が独自に作成したりしなければならないこともあります。IEEE 1364 Verilog標準で提供されているPLIルーチンには、TFルーチン、ACCルーチン、およびVPIルーチンがあります。本稿は、これらPLIルーチンの使用法の詳細については触れませんが、ユーザ定義のシステムタスクやシステム関数をSILOS/Harmonyで作成し使用するのに役立つ、PLIライブラリの作成フローを解説します。

次の例は、ユーザ定義システムタスクの命名から、Cルーチンとしてのインプリメント、コンパイル、PLIライブラリへのリンク、Verilogコードからの呼び出しまでの流れを分かりやすく示しています。図1に、SILOS/Harmonyにおいてユーザ定義のシステムタスク/システム関数を作成し呼び出すまでのフローの概要を示します。

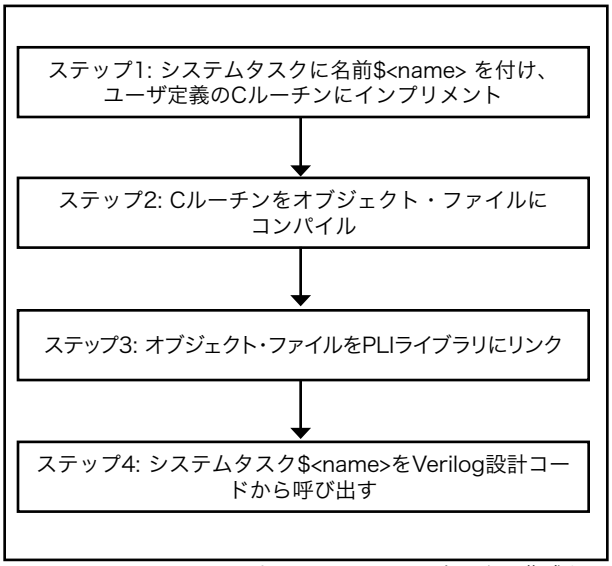

図1: SILOS/HarmonyにおけるPLIシステムタスクの作成から呼び出しまでのフロー

PLIライブラリの作成方法とWindowsでのシミュレーションの例

Step 1: ユーザ定義Cルーチンの命名とインプリメント

設計の目的に応じて、設計者が独自に作成したユーザ定義システムタスクを命名し、PLIルーチンを使ってCソース・コードにインプリメントすることができます。SILOS/Harmonyは、ユーザ定義のコードとシミュレータ間のインタフェースとして、TFルーチンおよびACCルーチンをサポートしています。これらのPLIルーチンについての詳細は、IEEEから入手可能な『IEEE 1364 Verilog HDL仕様』を参照してください。

以下に、tf_putp()ルーチンを使用したユーザ定義システム関数のCソース・コード例を示します。

```
/*////////////////////////////////////////////
////////////////
//
// title: C code for tf_putp() test to write a
// single simple value
//
// This PLI C code uses tf_putp() to write a
// simple value to a Verilog system
// task/function argument (parameters).
//
////////////////////////////////////////////////
////////////////*/

#ifdef _WIN32
#define DllExport __declspec(dllexport)
#else
#define DllExport
#endif

#include <stdio.h>
#include "veriuser.h"

DllExport int putp_01_calltf();

char *veriuser_version_str = "";

int (*endofcompile_routines[])() =
```

```
{
    /*** my_eoc_routine, ***/
    0 /*** final entry must be 0 ***/
};

bool err_intercept(level,facility,code)
int level; char *facility; char *code;
{ return(true); }

DllExport s_tfcell veriusertfs[] = {
  {usertask,     0, 0, 0, putp_01_calltf,  0,
"$putp_test", 1},
  {0}
};


/* Use tf_putp() with a single task/function
argument */
DllExport int putp_01_calltf()
{
  int val = 1;
  io_printf("PLI Code:   tf_putp(1, val) is
writing a value of %x (hex)\n",
    val);
  tf_putp(1, val);
  return(0);
}
```

上記のCソース・コードにおいて、$putp_testは、Verilogコードから呼び出し可能なユーザ定義システムタスクです。putp_01_calltf()は、tf_putp()ルーチン(PLIルーチン)をインプリメントするためのCルーチンです。$putp_testとputp_01_calltf()は、veriusertfsテーブルを介して実行時にリンクされます。

次のCソース・コード例のveriusertfsテーブルは、s_tfcell構造内にエントリを持ちます。

```
s_tfcell veriusertfs[] = {
  {usertask,     0, 0, 0, putp_01_
  calltf,  0, "$putp_test", 1},
  {0}
};
```

veriusertfsテーブルの各フィールドについての簡単な説明が、veriuserstfs.hファイル内にあります。

```
/* VERILOG user tasks and functions C header
file */
typedef struct t_tfcell
{
    short type;         /* either usertask or
userfunction */
    short data;         /* parameter for the
following routines */
    int (*checktf)();   /* routine for checking
parameters */
    int (*sizetf)();    /* for providing size
of function return value */
    int (*calltf)();    /* routine called dur-
ing simulation */
    int (*misctf)();    /* miscellaneous rou-
tine (see below) */
    char *tfname;       /* the name of the sys-
tem task or function */
    int forwref;       /* indicates special pa-
rameters allowed */
    char *tfveritool;   /* Which Veritool owns
the task */
    char *tferrmessage; /* An optional special
case error message
                           which will be printed
if the task is skipped */

    /* these components are for system usage
only */
    int hash;
    struct t_tfcell *left_p;
    struct t_tfcell *right_p;
    char *namecell_p;
    int warning_printed;  /* Flag is set when
skipping warning is printed */
} s_tfcell, *p_tfcell;
```

注記: 上記のCソース・コードでは、エクスポートされた関数とデータを出力DLLファイルに追加するため、マクロDllExportを定義しています。DllExport宣言を使用しない場合、.defファイルを別途用意し、これらのエクスポート名を定義する必要があります。詳細については、Microsoft社のリンカに関するドキュメントを参照してください。

ステップ2: Cソース・コードのコンパイル、オブジェクト・ファイル(*.obj)の作成

お使いのマシンに最新バージョンのMicrosoft　C/C++コンパイラがインストールされていること、および、コンパイラとリンカがコマンド・ウィンドウで実行できる環境が設定されていることを確認してください。

上記のCソース・コードをコンパイルするには、ユーザ定義のCソース・コード・ファイルに加えて、3つのヘッダ・ファイル(acc_user.h、ext_user.h、およびveriuser.h)が必要です。これらのファイルを入手するにはシルバコ・ジャパン(E-mail: jsupport@silvaco.com)へお問い合わせください。

また、最新版のSILOS dllインポート・ライブラリ (libsilosdll.lib)を作業ディレクトリにコピーする必要があります。このファイルは、SILOSまたはHarmonyのWindowsインストール・ディレクトリ（例: C:/sedatools/lib/silos/4.10.39.R/x86-nt)にあります。

コマンド・ウィンドウを開き、作業ディレクトリに移動します。次のコマンドを入力すると、Cソース・コードがコンパイルされ、オブジェクト・ファイル(*.obj)が作成されます。

```
cl /ML /c myfile.c
```

場合によっては、その他のコンパイラ・フラグを追加する必要があります。ソース・コードには、そのソース・コードに含まれるユーザ定義システム関数に関するエントリを持つ、veriusertfsテーブルが1つ含まれている必要があります。

ステップ3: PLIライブラリへのリンク

次のコマンドを使用すると、mypli.dllという名前のPLIライブラリが作成されます。

```
link /dll libsilosdll.lib myfile.obj
```

ステップ4: Verilogコードからのシステムタスク$putp_test()の呼び出しとシミュレーション

.dllファイルを作成した後、!ploadコマンドを使用してSILOS/Harmonyに読み込ませる.dllファイルを指定します。!ploadコマンドは、モジュールの境界内にない限り、どのVerilogインプット・ファイルの中にも置くことができます。ユーザ定義システムタスク$ tf_putp()を使用したVerilogファイルの例を以下に示します。

```
////////////////////////////////////////////////
/////////////////
//
// title: testbench for tf_putp() test to write
// a simple value
//
// This test writes a value to a scalar reg
// data type
//
////////////////////////////////////////////////
/////////////////
`timescale 1ns/1ns

/*
 * "silos_ms" is automatically defined by
SILOS-X
 * on the Windows platform
 */
`ifdef silos_ms
!pliload myfile.dll
`else
!pliload myfile.so
`endif

module test;

  reg r1;

  initial
    begin
      #1
      $display("\nTest Bench: executing
\"$putp_test(r1);\" ");
      $display("Test Bench: expect tfarg 1 to
receive 1 (hex)");
      $putp_test(r1);
      #1
      $display("Test Bench: tfarg 1 received %h
(hex)", r1);

      #1 $display("\n\n--------End of Test
Bench--------");
      $finish;
    end
endmodule
```

シミュレーション結果

WindowsマシンにおいてSILOSをバッチ・モードで実行した際のシミュレーション結果を以下に示します。

```
E:\example\pli_appNote> cl /ML /c myfile.c

E:\example\pli_appNote> link /dll libsilos-
dll.lib myfile.obj

E:\example\pli_appNote>   c:\sedatools\exe\
silosx -b pli01.v


E:\TestFiles\pli_appNote>l:\exe\silos.exe -b
pli01.v

           SILOS-X  Version 4.10.62.R

Copyright (c) 2004 - 2009 Simucad Design Auto-
mation, All rights reserved.

Copyright (c) 1984 - 2004 Silvaco Data Systems,
All rights reserved.

                  4701 Patrick Henry Drive
             Santa Clara, California, 95054,
U.S.A.
                (408)-567-1000  Fax: (408)-496-
6080
              Web Site:   "www.simucad.com"

Reading "pli01.v"
!pliload myfile.dll
```

```
Highest level modules (that have been auto-
instantiated):
                test
        2 total devices.
        Linking ...

        1 nets total: 0 saved and 0 monitored.
        0 registers total: 0 saved.
        Done.

Test Bench: executing "$putp_test(r1);"
Test Bench: expect tfarg 1 to receive 1 (hex)
PLI Code:   tf_putp(1, val) is writing a value
of 1 (hex)
Test Bench: tfarg 1 received 1 (hex)


--------End of Test Bench--------
$finish in file "E:\TestFiles\pli_appNote\
pli01.v" at line 30

        0 State changes on observable nets.

        Simulation stopped at the end of time
0.003us.
    No errors

E:\TestFiles\pli_appNote>
```

まとめ

IEEE-1364 Verilog標準にはVerilog言語を拡張する幅広いPLIルーチンが提供されているため、設計者は設計の目的に応じてオリジナルのシステムタスクやシステム関数を作成できます。本稿では、WindowsプラットフォームのSILOS/HarmonyにおいてPLIライブラリを作成し、ユーザ定義のシステムタスクを標準のVerilogコードから呼び出すために使用したフローを解説しました。

参考文献

[1] Verilog® HDL: A Guide to Digital Design and Synthesis, Second Edition.

[2] IEEE Standard Verilog® Hardware Description Language, IEEE Std 1364-2001.