# 原理

使用SV的dpi接口实现联合仿真。

## 应用场景与相关工具

此方法适用于算法模块仿真，模块中不应当有时序控制交互等功能。

C语言编译工具：gcc（Linux系统自带；windows系统可以使用cygwin）

RTL仿真工具：VCS

## RTL端实现

在TB中声明，C语言的函数（function），注意不支持task，即c语言侧不带消耗仿真时间。

program automatic ci\_verify(input reg clk);

import "DPI-C" function void ci\_rm\_wrapper

(

input shortint PBMod,

input shortint FECMod,

input shortint StepSize,

input shortint Offset,

input shortint Col\_Rotate\_Mod,

input shortint Shift,

input bit [15:0] sv\_ci\_din[],

input bit [15:0] sv\_ci\_dou[],

input bit [15:0] sv\_cdi\_din[],

input bit [15:0] sv\_cdi\_dou[]

);

//.......

task task\_ci\_sim\_item;

//...............

ci\_rm\_wrapper(

ci\_param\_u.PBMod,

ci\_param\_u.FECMod,

ci\_param\_u.StepSize,

ci\_param\_u.Offset,

ci\_param\_u.Col\_Rotate\_Mod,

ci\_param\_u.Shift,

ci\_seq\_item\_u.ci\_din,

ci\_seq\_item\_u.ci\_dou,

ci\_seq\_item\_u.cdi\_din,

ci\_seq\_item\_u.cdi\_dou

);

//............

endtask

initial begin

//..........

task\_ci\_sim\_item();

//...........

end

endprogram

## C语言侧

C语言应当兼容gcc。

建议C模型的函数接口设置为，参数（结构体），输入（数组，或变量），输出（数组，或变量）。

C模型应当包含系统定义头文件、算法函数，主函数（测试算法函数正确性）。

C语言如要提供编译后的.o文件，则需保证编译环境为CentOS 6.5+ 系统；其它环境编译的文件无法使用。

### 系统定义头文件

#ifndef \_TOP\_H\_

#define \_TOP\_H\_

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#include <string.h>

#include <time.h>

typedef char sWord16\_c; //16bits char

typedef short sWord16;

typedef unsigned short uWord16;

typedef long sWord32;

typedef unsigned long uWord32;

typedef float Word32\_f; //16bits float IEEE754

typedef double Word64\_f; //64bits double IEEE754

typedef \_\_int64 sWord40; //40bits 有符号长整型

typedef unsigned \_\_int64 uWord40; //40bits 无符号长整型

typedef \_\_int64 sWord64; //64bits 有符号长整型

typedef unsigned \_\_int64 uWord64; //64bits 无符号长整型

#define MAX16 32767

#define MIN16 -32768

#define UMAX16 65535

#define MAX32 (sWord32)0x7fffffff

#define MIN32 (sWord32)0x80000000

#define UMAX32 4294967295

#define MAX40 (sWord40)0x7fffffffff

//#define MIN40 (sWord40)0x8000000000

#define MIN40 (sWord40)((sWord64)0x8000000000000000>>24)

#define UMAX40 1099511627775

#define MAX64 (sWord64)0x7fffffffffffffff

#define MIN64 (sWord64)0x8000000000000000

#define UMAX64 18446744073709551615

#define SHIFTL(data, len) ((sWord40)(data)\*(sWord40)pow(2,(len)))

#define SHIFTR(data, len) ((sWord40)(data)/(sWord40)pow(2,(len)))

typedef struct

{

uWord16 PBMod; // PB块大小

uWord16 FECMod; // Turbo编码模式

uWord16 StepSize; // 信道交织跳转步长

uWord16 OffSet; // 信道交织起始偏置

uWord16 Col\_Rotate\_Mod; // 信道交织列旋转的模式,0:1~8为一组循环;1:1~10为一组循环;2:1~12为一组循环;3:1~14为一组循环

uWord16 Shift; // 信道交织半字节移位方向

uWord16 PBlen; // PB块长度

} STRUCT\_ChanelIntlvPara;

// =============================== TX ==============================

void LME3460\_Channel\_Interleaver(uWord16 \*Input,uWord16 \*Output, STRUCT\_ChanelIntlvPara ChanelIntlvPara);

// ================================ RX =============================

void LME3460\_Channel\_Deinterleaver(sWord16 \*Input,sWord16 \*Output, STRUCT\_ChanelIntlvPara ChanelIntlvPara);

#endif

### 算法模型

### 自测用主函数

#include "Sys\_Top.h"

void main()

{

uWord16 i = 0;

uWord16 CI\_Input[8320] = {0}, CI\_Output[8320] = {0};

sWord16 CDI\_Input[8320] = {0}, CDI\_Output[8320] = {0};

STRUCT\_ChanelIntlvPara ChanelIntlvPara;

ChanelIntlvPara.PBMod = 0;

ChanelIntlvPara.FECMod = 0;

ChanelIntlvPara.OffSet = 16;

ChanelIntlvPara.StepSize = 4;

ChanelIntlvPara.Shift = 0;

ChanelIntlvPara.Col\_Rotate\_Mod = 0;

for (i = 0; i < 256; i++)

{

CI\_Input[i] = i;

}

LME3460\_Channel\_Interleaver(CI\_Input,CI\_Output,ChanelIntlvPara);

for (i = 0; i < 256; i++)

{

CDI\_Input[i] = CI\_Output[i];

}

LME3460\_Channel\_Deinterleaver(CDI\_Input,CDI\_Output,ChanelIntlvPara);

}

### Gcc使用

使用第一个命令可以生成可执行文件 chi\_run.out ，运行该文件则可以知道算法函数是否在gcc编译后正确运行。

在Centos 6.5+ 上使用第二个命令生成的chi.out文件可以在RTL仿真平台使用

gcc -Wall sys\_Top.h LME3460\_Channel\_Interleaver.c LME3460\_Channel\_Deinterleaver main.c -o chi\_run.out

gcc -c sys\_Top.h LME3460\_Channel\_Interleaver.c LME3460\_Channel\_Deinterleaver -o chi.out

## Wrapper

RTL人员在调用c模型时，需要使用sv自带的dpi方法包裹c模型。

变量可以直接赋值，数组需要使用sv的dpi方法，获得指针之后才进行赋值。

#include "Sys\_Top.h"

#include "svdpi.h"

void ci\_rm\_wrapper(

const uWord16 PBMod,

const uWord16 FECMod,

//.......

const svOpenArrayHandle sv\_ci\_din,

svOpenArrayHandle sv\_ci\_dou,

const svOpenArrayHandle sv\_cdi\_din,

svOpenArrayHandle sv\_cdi\_dou,

)

{

STRUCT\_ChanelIntlvPara ci\_rm\_param;

int i,j;

uWord16 c\_ci\_din[8320],c\_ci\_dou[8320];

uWord16 \*a, \*b;

ci\_rm\_param.PBMod = PBMod;

ci\_rm\_param.FECMod = FECMod;

//......

for(i=0; i<8320; i++)

{

if(i < dlen)

{

a = (uWord16 \*) svGetArrElemPtr(sv\_ci\_din,i);

c\_ci\_din[i] = \*a;

}

else

c\_ci\_din[i] = 0;

}

LME3460\_Channel\_Interleaver(c\_ci\_din,c\_ci\_dou, ci\_rm\_param);

for(i=0; i<dlen; i++)

{

b = (uWord16 \*) svGetArrElemPtr(sv\_ci\_dou,i);

\* b = c\_ci\_dou[i];

}

//...

}

## VCS设定

Vcs仿真时添加选项 +vpi +sverlog

添加c源文件（不包括main.c），或者编译后的chi.out