CLA 算法案例开发手册

Revision History

Draft Date	Revision No.	Description
2022/09/10	V1.1	1. 内容勘误。
2021/03/04	V1.0	1. 初始版本。

目　　录

前　言

（裸机）CLA 算法案例位于产品资料“4-软件资料\Demo\Algorithm-demos\”路径下。案例目录说明如下表，其中 bin 目录存放程序可执行文件，project 目录目录存放案例工程源文件。

表 1

目录	说明
bin	程序可执行文件 xxx_ram.out，用于加载至 DSP 片内 RAM 程序可执行文件 xxx_flash.out，用于固化至 DSP 片内 FLASH
project	工程源码

本文档案例程序默认使用 DSP 为 TMS320F28377D 的核心板，通过 TL-XDS200 仿真器加载运行进行操作效果演示。

1 cla_divide 案例

1.1 案例说明

案例功能：演示 CLA(Control Law Accelerators)核心的使用方法。

程序定义分子分母变量，并不断修改分子分母大小，由 CPU1 核心唤醒 CLA 核心对分子分母共进行 64 次除法运算，然后通过 CCS 读取程序变量值，以校验 CLA 除法运算结果的正确性。

1.2 案例测试

请加载程序到 CPU1 核心运行。然后点击 CCS 的"View -> Expressions"，在弹出的 Expressions 窗口点击"Add new expression"依次新建 g_pass、g_fail、g_div_val 和 g_div_expected 程序变量。

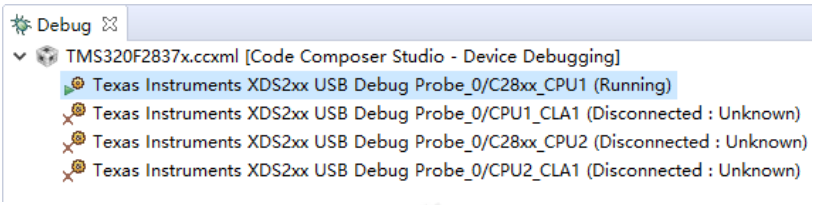

图 1

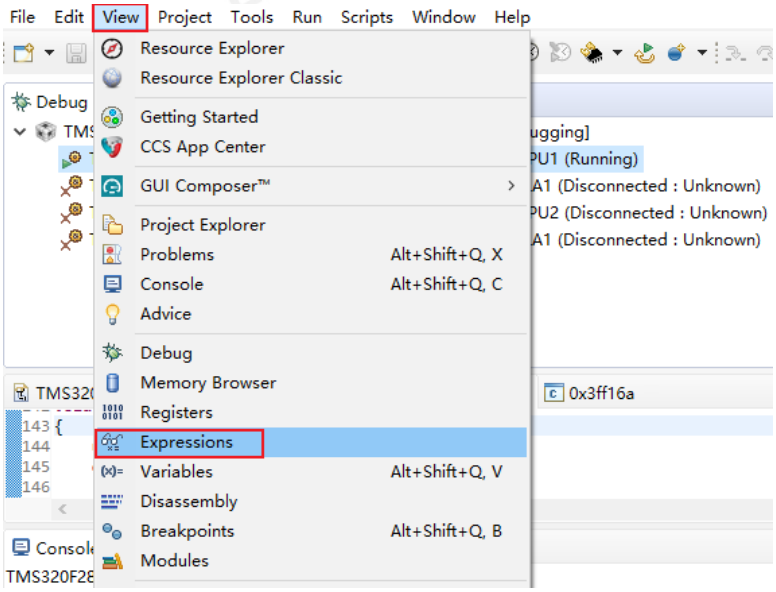

图 2

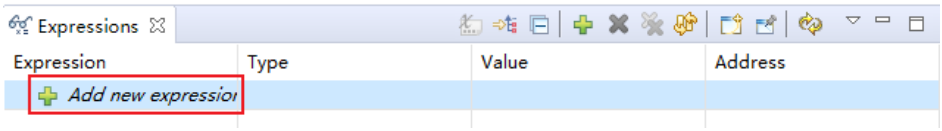

图 3

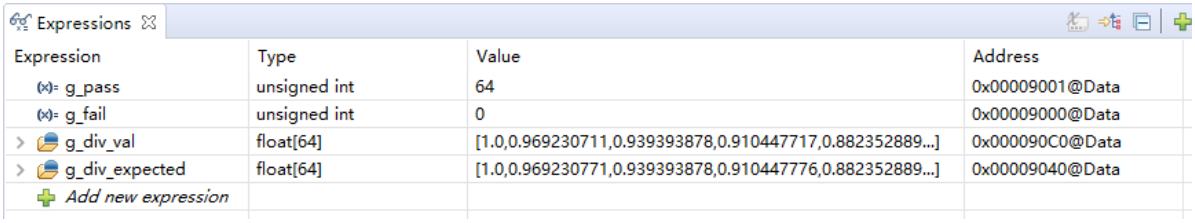

图 4

表 2

参数变量	解析
g_pass	除法运算校验成功次数
g_fail	除法运算校验失败次数
g_div_val	CLA 除法运算结果
g_div_expected	程序预设的除法运算正确值，用于校验 CLA 除法运算结果是否正确

g_div_val 与 g_div_expected 数值一致，说明 CLA 除法运算结果正确。

1.3 关键代码

（1） 定义循环次数 BUFFER_SIZE 为 64，且定义基于 BUFFER_SIZE 的除法运算正确值 g_div_expected[BUFFER_SIZE]。

```
/* buffer size */
#define BUFFER_SIZE     64

/* global buffer */
float g_div_val[BUFFER_SIZE];
float g_div_expected[BUFFER_SIZE]={
    1, 0.9692308, 0.9393939, 0.9104478, 0.8823529,
    0.8550724, 0.8285714, 0.8028169, 0.7777778, 0.7534246,
    0.7297297, 0.7066666, 0.6842105, 0.6623377, 0.6410257,
    0.6202531, 0.6000000, 0.5802469, 0.5609756, 0.5421687,
    0.5238096, 0.5058824, 0.4883721, 0.4712644, 0.4545455,
    0.4382023, 0.4222222, 0.4065934, 0.3913043, 0.3763441,
    0.3617021, 0.3473684, 0.3333333, 0.3195876, 0.3061225,
    0.2929293, 0.2800000, 0.2673267, 0.2549020, 0.2427184,
    0.2307692, 0.2190476, 0.2075472, 0.1962617, 0.1851852,
    0.1743119, 0.1636364, 0.1531532, 0.1428571, 0.1327434,
    0.1228070, 0.1130435, 0.1034483, 0.09401710, 0.08474576,
    0.07563026, 0.06666667, 0.05785124, 0.04918033, 0.04065040,
    0.03225806, 0.02400000, 0.01587302, 0.007874016
};
```

图 5

（2） 在 CLA_initCpu1Cla1 函数中，注册 Task1 的中断服务函数为 Cla1Task1，该函数在 divide.cla 文件中已定义。当程序调用 Cla1ForceTask1andWait 函数时，将调用 Cla1Task1 进行运算。

```
/* CLA_initCpu1Cla1 - Setup CLA task vectors and set PIE table ISR handles */
void CLA_initCpu1Cla1(void)
{
    /*
     * Compute all CLA task vectors
     * On Type-1 CLAs the MVECT registers accept full 16-bit task addresses as
     * opposed to offsets used on older Type-0 CLAs
     */
    EALLOW;
    Cla1Regs.MVECT1 = (uint16_t)(&Cla1Task1);

    /*
     * Enable the IACK instruction to start a task on CLA in software
     * for all  8 CLA tasks. Also, globally enable all 8 tasks (or a
     * subset of tasks) by writing to their respective bits in the
     * MIER register
     */
    Cla1Regs.MCTL.bit.IACKE = 1;
    Cla1Regs.MIER.all = 0x00FF;

    /*
     * Configure the vectors for the end-of-task interrupt
     */
    PieVectTable.CLA1_1_INT = &cla1Isr1;

    /* Enable CLA interrupts at the group and subgroup levels */
    PieCtrlRegs.PIEIER11.all = 0xFFFF;
    IER |= (M_INT11);
}
```

图 6

（3） 在 Cla1Task1 函数中读取全局变量 Num 和 Den，进行除法运算，并将结果保存至

全局变量 Res 中。

```
#include "cla_divide.h"

/* Task 1 : Divide */
__interrupt void Cla1Task1(void)
{
    Res = Num / Den;
}
```

图 7 src\divide.cla

（4） 在 main 函数中，分别进行设备和 CLA 初始化。完成初始化后，将进行循环除法运算。初始化分子分母变量后，将唤醒 CLA 核心进行除法运算。BUFFER_SIZE = 64，则除法运算循环为：64/64、63/65、62/66、61/67…，分子循环减 1，分母循环加 1，直至循环结束。定义的 g_div_expected 数值通过预先计算得到，用于与 CLA 除法运算结果进行校验。

```
void main(void)
{
    unsigned char i = 0;
    double abs = 0.0;

    /* device initialization */
    Device_init();                          设置初始化

    /* Configure CLA memory sections */
    CLA_configClaMemory();                  配置CLA内存空间

    /* Initialize CLA1 task vectors and end of task interrupts */
    CLA_initCpu1Cla1();                     配置CLA任务和中断服务函数

    for(i = 0; i < BUFFER_SIZE; i++) {
        Num = (float)(BUFFER_SIZE - i);     初始化分子分母
        Den = (float)(BUFFER_SIZE + i);

        /* Wakeup the Task and wait it done */
        Cla1ForceTask1andWait();            唤醒CLA任务，并等待完成

        /* Get and Verify the data */
        g_div_val[i] = Res;

        /* Get abs value */
        abs = fabs(g_div_expected[i]-g_div_val[i]);   校验CLA计算正确性
        if(abs < 0.01) {
            g_pass ++;
        } else {
            g_fail ++;
        }
    }
}
```

图 8

更多帮助

销售邮箱：sales@tronlong.com

技术邮箱：support@tronlong.com

创龙总机：020-8998-6280

技术热线：020-3893-9734

创龙官网：www.tronlong.com

技术论坛：www.51ele.net

官方商城：https://tronlong.tmall.com