1. Edma transmission problem：
According to the edma\_test.c use case, a block transmission is implemented. If the source address and destination address are both in MSMC, the data transmission result is correct. Configure cmd file, source address in DDR3, while using #pragma DATA\_SECTION to set the destination address to MSMC, data transmission error, the code is as follows, perhaps the example understanding is wrong, hope to give advice according to edma\_test.c to achieve a block transmission;

The code is as follows:

#include <stdio.h>

#include <c6x.h>

#include <ti/csl/csl\_edma3.h>

#include <ti/csl/csl\_edma3Aux.h>

#include <ti/csl/csl\_cacheAux.h>

#include <stdlib.h>

float srcBuff[1024];

#pragma DATA\_SECTION(dstBuff,".dst")

 float dstBuff[1024];

static Bool Verify\_Transfer

(

 float \*srcBuff,

 float \*dstBuff

)

{

 Uint16 key;

 /\* Invalidate the cache before verification \*/

 /\* Disable Interrupts \*/

 key = \_disable\_interrupts();

 CACHE\_invL1d ((void \*)srcBuff, 1024\*sizeof(float), CACHE\_WAIT);

 CACHE\_invL2 ((void \*)srcBuff, 1024\*sizeof(float), CACHE\_WAIT);

 CACHE\_invL1d ((void \*)dstBuff, 1024\*sizeof(float), CACHE\_WAIT);

 CACHE\_invL2 ((void \*)dstBuff, 1024\*sizeof(float), CACHE\_WAIT);

 \_mfence();

 /\* Re-enable Interrupts. \*/

 \_restore\_interrupts(key);

 return TRUE;

}

static Int32 edma\_ping\_pong\_xfer\_gbl\_region (Int32 instNum, Uint8 channelNum, float \*srcBuff1, float \*dstBuff1)

{

 CSL\_Edma3Handle hModule;

 CSL\_Edma3Obj edmaObj;

 CSL\_Edma3ParamHandle htrans;

 CSL\_Edma3ChannelObj chObj;

 CSL\_Edma3CmdIntr regionIntr;

 CSL\_Edma3ChannelHandle hChannel;

 CSL\_Edma3ParamSetup myParamSetup;

 CSL\_Edma3Context context;

 CSL\_Edma3ChannelAttr chAttr;

 CSL\_Status status;

 int loopIndex;

 /\* Start the EDMA PING-PONG test over the Global Region. \*/

 printf ("Debug: Testing EDMA(%d) Ping-Pong Test (Global) Region for Channel %d...\n", instNum, channelNum);

 /\* Initialize data \*/

 for (loopIndex = 0; loopIndex < 1024; loopIndex++)

 {

 srcBuff1[loopIndex] = loopIndex;

 dstBuff1[loopIndex] = 1;

 }

 /\* Module Initialization \*/

 CSL\_edma3Init(&context);

 /\* Open the EDMA Module using the provided instance number \*/

 hModule = CSL\_edma3Open(&edmaObj, instNum, NULL, &status);

 /\* Channel open \*/

 chAttr.regionNum = CSL\_EDMA3\_REGION\_GLOBAL;

 chAttr.chaNum = channelNum;

 hChannel = CSL\_edma3ChannelOpen(&chObj, instNum, &chAttr, &status);

 /\* For first EDMA instance there are only 2 TCs and 2 event queues

 \* Modify the channel default queue setup from 0 to 1

 \*/

 CSL\_edma3HwChannelSetupQue(hChannel,CSL\_EDMA3\_QUE\_1);

 /\* Map the DMA Channel to PARAM Block 2. \*/

 //CSL\_edma3MapDMAChannelToParamBlock (hModule, channelNum, 2);

 CSL\_edma3MapDMAChannelToParamBlock (hModule, channelNum, 1);

 /\* Obtain a handle to parameter set 2 \*/

 // hParamPing = CSL\_edma3GetParamHandle(hChannel, 2, &status);

 htrans = CSL\_edma3GetParamHandle(hChannel, 1, &status);

 myParamSetup.option = CSL\_EDMA3\_OPT\_MAKE(CSL\_EDMA3\_ITCCH\_DIS, \

 CSL\_EDMA3\_TCCH\_DIS, \

 CSL\_EDMA3\_ITCINT\_DIS, \

 CSL\_EDMA3\_TCINT\_EN, \

 0, CSL\_EDMA3\_TCC\_NORMAL,\

 CSL\_EDMA3\_FIFOWIDTH\_NONE, \

 CSL\_EDMA3\_STATIC\_EN, \

 CSL\_EDMA3\_SYNC\_A, \

 CSL\_EDMA3\_ADDRMODE\_INCR, \

 CSL\_EDMA3\_ADDRMODE\_INCR );

 myParamSetup.srcAddr = (Uint32)srcBuff1;

 myParamSetup.aCntbCnt = CSL\_EDMA3\_CNT\_MAKE(4\*1024,1);

 myParamSetup.dstAddr = (Uint32)dstBuff1;

 myParamSetup.srcDstBidx = CSL\_EDMA3\_BIDX\_MAKE(4\*1024,4\*1024);

 myParamSetup.linkBcntrld= CSL\_EDMA3\_LINKBCNTRLD\_MAKE(0xffff,0);

 myParamSetup.srcDstCidx = CSL\_EDMA3\_CIDX\_MAKE(4\*1024,4\*1024);

 myParamSetup.cCnt = 1;

 /\* htrans setup \*/

 CSL\_edma3ParamSetup(htrans,&myParamSetup);

 /\* Interrupt enable (Bits 0-1) for the global region interrupts \*/

 regionIntr.region = CSL\_EDMA3\_REGION\_GLOBAL;

 regionIntr.intr = 0x1;

 regionIntr.intrh = 0x0000;

 CSL\_edma3HwControl(hModule,CSL\_EDMA3\_CMD\_INTR\_ENABLE,&regionIntr);

 /\* Trigger channel \*/

 CSL\_edma3HwChannelControl(hChannel,CSL\_EDMA3\_CMD\_CHANNEL\_SET,NULL);

 regionIntr.region = CSL\_EDMA3\_REGION\_GLOBAL;

 regionIntr.intr = 0;

 regionIntr.intrh = 0;

 /\* Poll on IPR bit 0 \*/

 do {

 CSL\_edma3GetHwStatus(hModule,CSL\_EDMA3\_QUERY\_INTRPEND,&regionIntr);

 } while (!(regionIntr.intr & 0x1));

 /\* Clear the pending bit \*/

 CSL\_edma3HwControl(hModule,CSL\_EDMA3\_CMD\_INTRPEND\_CLEAR,&regionIntr);

 /\* Check transfer by comparing the source and destination buffers \*/

 Verify\_Transfer( srcBuff1, dstBuff1);

 /\* Close channel \*/

 CSL\_edma3ChannelClose(hChannel);

 /\* Close EDMA module \*/

 CSL\_edma3Close(hModule);

 /\* The test passed. \*/

 return 0;

}

int main(){

 int loopIndex;

// float\* srcBuff;

// srcBuff = (float \*) malloc(1024 \* sizeof(float));

// dstBuff = (float \*) malloc(n \* sizeof(float));

 edma\_ping\_pong\_xfer\_gbl\_region(0,2,srcBuff,dstBuff);

 for (loopIndex = 0; loopIndex < 1024; loopIndex++)

 {

 printf("src1=%d ",srcBuff[loopIndex]);

 printf("dst1=%d \n",dstBuff[loopIndex]);

 }

}

1. Refer to the example of emda\_test.c, set PaRAM as follows, where set ACNT to 256, BCNT to 1, CCNT to 1, using A transfer mode, according to the understanding of the manual，srcBidx=dtsBidx=ACNT,scrCidx=dstCidx=ACNT,But that's not how it's set up in the example.

myParamSetup.option = CSL\_EDMA3\_OPT\_MAKE(CSL\_EDMA3\_ITCCH\_DIS, \
CSL\_EDMA3\_TCCH\_DIS, \
CSL\_EDMA3\_ITCINT\_DIS, \
CSL\_EDMA3\_TCINT\_EN, \
0, CSL\_EDMA3\_TCC\_NORMAL,\
CSL\_EDMA3\_FIFOWIDTH\_NONE, \
CSL\_EDMA3\_STATIC\_DIS, \
CSL\_EDMA3\_SYNC\_A, \
CSL\_EDMA3\_ADDRMODE\_INCR, \
CSL\_EDMA3\_ADDRMODE\_INCR );
myParamSetup.srcAddr = (Uint32)srcBuff1;
myParamSetup.aCntbCnt = CSL\_EDMA3\_CNT\_MAKE(256,1);
myParamSetup.dstAddr = (Uint32)dstBuff1;
myParamSetup.srcDstBidx = CSL\_EDMA3\_BIDX\_MAKE(1,1);
myParamSetup.linkBcntrld= CSL\_EDMA3\_LINKBCNTRLD\_MAKE(hParamPong,0);
myParamSetup.srcDstCidx = CSL\_EDMA3\_CIDX\_MAKE(0,1);
myParamSetup.cCnt = 1;