YUNTU 云途半导体

LINFlexD UART模式使用DMA传输异常数据解决方案

YUNTU AUTOMOTIVE SEMICONDUCTOR

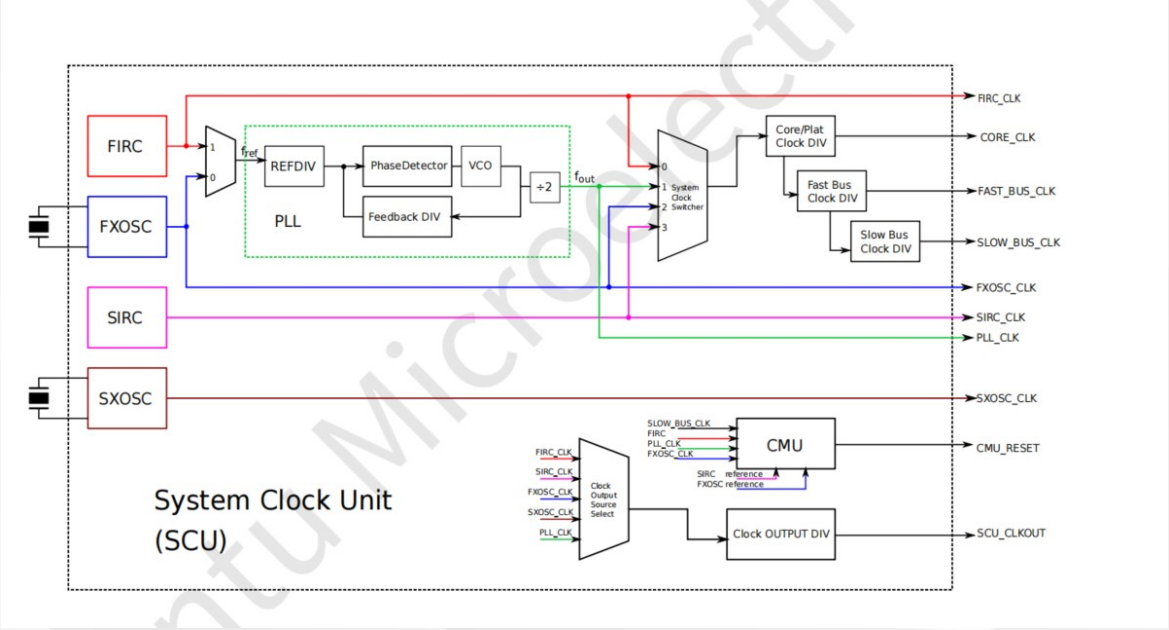
真车规 高可靠 高性能

问题分析:

ME0：

LINFlexD UART模式在使用DMA进行数据传输时，如果LINFlexD模块时钟与内核时钟（系统时钟）不同源或者频率过高，可能出现实际传输的数据与预期发送的数据不一致。

原因为DMA信号过长，导致LINFlexD传输之后，DMA信号没有消除。LINFlexD进行了多次传输。



ME0时钟树

No.	Module Name	Offset	Bus Clock	Bus Clock Enable	Peripheral Functional Clock	Software Reset	Additional Clock
0	-	0x000	-	-	-	-	-
1	-	0x004	-	-	-	-	-
2	-	0x008	-	-	-	-	-
3	-	0x00C	-	-	-	-	-
4	-	0x010	-	-	-	-	-
5	-	0x014	-	-	-	-	-
6	-	0x018	-	-	-	-	-
7	-	0x01C	-	-	-	-	-
8	DMA	0x020	CORE_CLK	OFF	-	YES	-
9	-	0x024	-	-	-	-	-
10	-	0x028	-	-	-	-	-
11	-	0x02C	-	-	-	-	-
12	-	0x030	-	-	-	-	-

Table 13.1 continued from previous page

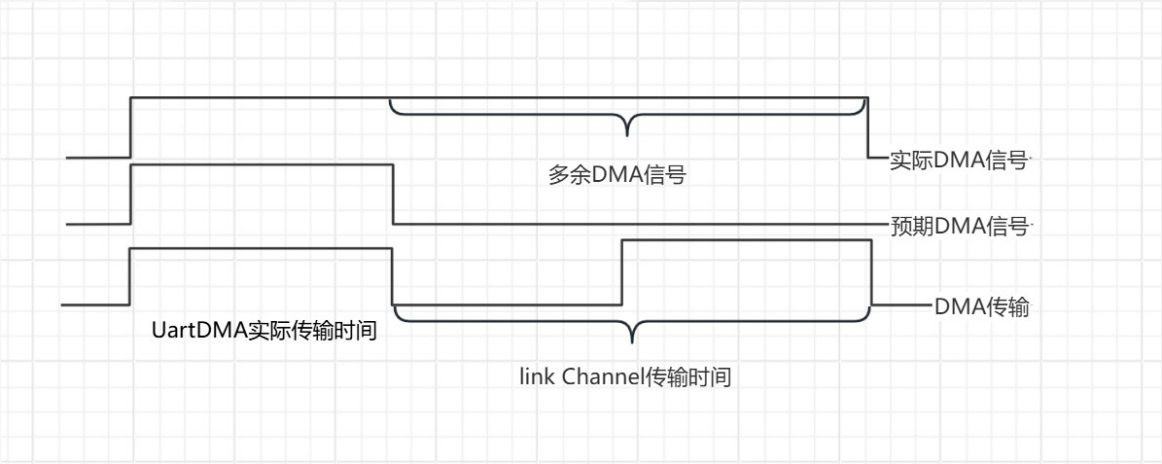
No.	Module Name	Offset	Bus Clock	Bus Clock Enable	Peripheral Functional Clock	Software Reset	Additional Clock
24	-	0x060	-	-	-	-	-
25	-	0x064	-	-	-	-	-
26	-	0x068	-	-	-	-	-
27	LINFlexD0	0x06C	SLOW_BUS_CLK	OFF	-	YES	FAST_BUS_CLK
28	LINFlexD1	0x070	SLOW_BUS_CLK	OFF	-	YES	FAST_BUS_CLK
29	LINFlexD2	0x074	SLOW_BUS_CLK	OFF	-	YES	FAST_BUS_CLK
30	LINFlexD3	0x078	SLOW_BUS_CLK	OFF	-	YES	FAST_BUS_CLK
31	LINFlexD4	0x07C	SLOW_BUS_CLK	OFF	-	YES	FAST_BUS_CLK
32	LINFlexD5	0x080	SLOW_BUS_CLK	OFF	-	YES	FAST_BUS_CLK
33	-	0x084	-	-	-	-	-
34	-	0x088	-	-	-	-	-
35	I2C0	0x08C	SLOW_BUS_CLK	OFF	YES	YES	-
36	I2C1	0x090	SLOW_BUS_CLK	OFF	YES	YES	-
37	I2C2	0x094	SLOW_BUS_CLK	OFF	YES	YES	-

问题修改方案:

DMA具有channellink的功能，即当前DMA传输完成后，可以触发link使能的DMA通道进行传输，且link传输次数为单次。

修改方案为：

1. UART每次进行DMA传输时，均link一次最后一个DMA通道，以此来抵消掉多余DMA信号带来的影响。
2. 将最后一个DMA通道配置为CIM->DIEINFO的寄存器之间的传输通道。



9.3.1.22 DMA CTS_TCNT_LKEN Register

Offset: 416h + n * 20h

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	LKEN			LKCH				TCNT								
W																
Reset	u	0	0	u	u	u	u	u	u	u	u	u	u	u	u	u

Table 9.26: DMA CTS_TCNT_LKEN Register Description

Field	Function
15 LKEN	**Link Enable** 0b - Channel to channel linking is disabled, refer LKDIS for register information 1b - Channel to channel linking is enabled.
12 - 9	**Link Channel**

Field	Function
LKCH	DMA next trigger channel index after current dma trigger loop is done(one trigger)
8 - 0 TCNT	**Trigger Count** Trigger counter decrease 1 for every DMA trigger, when TCNT is 0, channel done flag set. ***NOTE***. If new trigger happens and hardware request is enabled, TCNT will reload TCNT and serve new trigger request.

Specific modification content (Register)：

Step1:Configure the max DMA channel attribute during initialization

1. Enable DMA the transfer loop mapping feature

```
DMA0->CTRL = DMA_CTRL_LOEN_MASK;
```

2. Set source and destination addresses （CIM）

```
DMA0->CTS[15].SADDR = (uint32_t)&CIM->DIEINFO;
DMA0->CTS[15].DADDR = (uint32_t)&CIM->DIEINFO;
```

3. Set transfer size for 4B

```
DMA0->CTS[15].TCR = (uint16_t) (DMA_CTS_TCR_SSIZE(DMA_TRANSFER_SIZE_4B) |
DMA_CTS_TCR_DSIZE(DMA_TRANSFER_SIZE_4B));
```

4.Set the total number of bytes to be transferred

```
DMA0->CTS[themaxchannel].BCNT.LODIS = DMA_CTS_BCNT_LODIS_BCNT(0x4U);
```

5.Set the max channel iteration count to 1 (single block mode)

```
regValTemp = DMA0->CTS[themaxchannel].TCNTRV;
regValTemp &= (uint16_t) ~(DMA_CTS_TCNT_LKDIS_TCNT_MASK);
regValTemp |= (uint16_t) DMA_CTS_TCNT_LKDIS_TCNT(1);
DMA0->CTS[themaxchannel].TCNTRV = regValTemp;
DMA0->CTS[themaxchannel].TCNT.LKDIS = regValTemp;
```

6.Disables the channel interrupt requests.(Prevent interrupt be enable from elsewhere)

```
regValTemp = DMA0->CTS[themaxchannel].CSR;
regValTemp &= (uint16_t) ~(DMA_CTS_CSR_TDINT_MASK);
regValTemp |= (uint16_t) DMA_CTS_CSR_TDINT(0U);
DMA0->CTS[themaxchannel].CSR = regValTemp;
```

7. Enable the DMA channel

E010002: [DMA] DMA REQEN register modify write may interfere with other channel(s)

```
/* Halt DMA before changing the REQEN register */
INT_SYS_DisableIRQGlobal;
base->CTRL|=1<<17u;
int32_t timeout=50;
/* Wait DMA active done */
while ((base->CTRL&DMA_CTRL_ACTIVE_MASK) !=0)
{
   timeout--;
   if (timeout<=0)
   {
      break;
   }
}
base->REQEN|= (0x01UL<<channel);
/* Resume DMA */
base->CTRL&=~(1<<17u);
INT_SYS_EnableIRQGlobal();
```

Specific modification content (Register)：

Step2.:When sending/receiving data using LINFlexD module in UART mode, link its DMA channel to the max channel.

1. Enable Uart Tx/Rx DMA channel Trigger Count with Link, Link Uart Tx/Rx DMA Channel and themaxchannel.

```
regValTemp=DMA0->CTS[sourcechannel].TCNTRV;
regValTemp&= (uint16_t) ~(DMA_CTS_TCNT_LKEN_LKCH_MASK);
regValTemp &= (uint16_t) ~(DMA_CTS_TCNT_LKEN_LKEN_MASK);
regValTemp|= (uint16_t) DMA_CTS_TCNT_LKEN_LKEN(1);
regValTemp|= (uint16_t) DMA_CTS_TCNT_LKEN_LKCH(themaxchannel);
DMA0->CTS[sourcechannel].TCNTRV=regValTemp;
DMA0->CTS[sourcechannel].TCNT.LKEN=regValTemp;
```

Note:

sourcechannel : sourcechannel is the DMA channel used by the customer when transmitting data in the UART DMA mode.
themaxchannel: themaxchannel is the DMA channel used to clear extra requests.

Specific modification content (SDK):

Step1:Add the following code to DMA_INIT

（Configure the max DMA channel attribute during）

```
const dma_channel_config_tdma_dummy_config= {
.virtChnConfig=FEATURE_DMA_VIRTUAL_CHANNELS-1U,
.source=DMA_REQ_DISABLED,
.callback=NULL,
.callbackParam=NULL,
};
dma_chn_state_t dma_dummyconfig_State;

/* Dummy channel to clear extra request */
DMA_DRV_ChannelInit(&dma_dummyconfig_State,&dma_dummy_config);
DMA_DRV_ConfigSingleBlockTransfer(FEATURE_DMA_VIRTUAL_CHANNELS-1U, DMA_TRANSFER_PERIPH2PERIPH,
(uint32_t)&CIM->DIEINFO, (uint32_t)&CIM->DIEINFO, DMA_TRANSFER_SIZE_4B, 4);
DMA_DRV_ConfigureInterrupt(FEATURE_DMA_VIRTUAL_CHANNELS-1U, DMA_CHN_MAJOR_LOOP_INT, false);
DMA_DRV_StartChannel(FEATURE_DMA_VIRTUAL_CHANNELS-1U);
```

Step2:Add the following code to DMA_DRV_ConfigMultiBlockTransfer

（When sending/receiving data using LINFlexD module in UART mode, link its DMA channel to the max channel.）

```
if(s_virtEdmaState->maxChannelForChLinkState==true)
{
  /* Set the final channel loop link to assist in DMA transmission */
  DMA_CTSSetChannelLoopLink(dmaRegBase, dmaChannel, FEATURE_DMA_VIRTUAL_CHANNELS-1U, true);
}
else
{
  /* No need for the final channel set loop link. */
}
```

Reference revision example(Register)：

dma_driver.c

```
platform > drivers > src > dma > C dma_driver.c > DMA_DRV_PushConfigToReg(uint8_t, const dma_transfer_config_t *)
                        uint32_t chnCount)
    if ((chnStateArray != NULL) && (chnConfigArray != NULL))
        for (index = 0U; index < chnCount; index++)
        {
            chnInitStatus = DMA_DRV_ChannelInit(chnStateArray[index], chnConfigArray[index]);
            if (chnInitStatus != STATUS_SUCCESS)
            {
                dmaStatus = chnInitStatus;
            }
        }
    }

        //1. Enable DMA the transfer loop mapping feature
        uint32_t regValTemp;
        DMA0->CTRL = DMA_CTRL_LOEN_MASK;
        //2. Set source and destination addresses (CIM)
        DMA0->CTS[15].SADDR = (uint32_t)&CIM->DIEINFO;
        DMA0->CTS[15].DADDR = (uint32_t)&CIM->DIEINFO;
        //3. Set transfer size for 4B
        DMA0->CTS[15].TCR = (uint16_t) (DMA_CTS_TCR_SSIZE(DMA_TRANSFER_SIZE_4B) | DMA_CTS_TCR_DSIZE(DMA_TRANSFER_SIZE_4B));

        //4.Set the total number of bytes to be transferred
        DMA0->CTS[15].BCNT.LODIS = DMA_CTS_BCNT_LODIS_BCNT(0x4U);

        //5.Set the max channel iteration count to 1 (single block mode)
        regValTemp = DMA0->CTS[15].TCNTRV;
        regValTemp &= (uint16_t) ~(DMA_CTS_TCNT_LKDIS_TCNT_MASK);
        regValTemp |= (uint16_t) DMA_CTS_TCNT_LKDIS_TCNT(1);
        DMA0->CTS[15].TCNTRV = regValTemp;
        DMA0->CTS[15].TCNT.LKDIS = regValTemp;
        //6.Disables the channel interrupt requests.(Prevent interrupt be enable from  elsewhere)
        regValTemp = DMA0->CTS[15].CSR;
        regValTemp &= (uint16_t) ~(DMA_CTS_CSR_TDINT_MASK);
        regValTemp |= (uint16_t) DMA_CTS_CSR_TDINT(0U);
        DMA0->CTS[15].CSR = regValTemp;
        DMA0->REQEN|= (0x01UL<<15);

问题  输出  调试控制台  终端  端口  ROBOT DOCUMENTATION  ROBOT OUTPUT
[main] 正在生成文件夹: c:/Users/ytmicro.2DZ72T3/Desktop/YCT_DEMO/TEST/ME0/UART_DMA/test/build
```

```
    DEV_ASSERT((srcAddr % transferOffset) == 0U);
    DEV_ASSERT((destAddr % transferOffset) == 0U);
#endif /* (CUSTOM_DEVASSERT) || defined (DEV_ERROR_DETECT) */

    /* Configure the transfer for one data block */
    retStatus = DMA_DRV_ConfigSingleBlockTransfer(virtualChannel, type, srcAddr, destAddr, transferSize, blockSize);

    if (retStatus == STATUS_SUCCESS)
    {
        DMA_Type *dmaRegBase = s_dmaBase[dmaInstance];

        /* Set the number of data blocks */
        DMA_CTSSetTriggerCount(dmaRegBase, dmaChannel, blockCount);

        /* Enable/disable requests upon completion */
        DMA_CTSSetDisableDmaRequestAfterCTSDoneCmd(dmaRegBase, dmaChannel, disableReqOnCompletion);


        /* Set the final channel loop link to assist in DMA transmission */
        uint32_t regValTemp;
        regValTemp = DMA0->CTS[dmaChannel].TCNTRV;
        regValTemp &= (uint16_t) ~(DMA_CTS_TCNT_LKEN_LKCH_MASK);
        regValTemp &= (uint16_t) ~(DMA_CTS_TCNT_LKEN_LKEN_MASK);
        regValTemp |= (uint16_t) DMA_CTS_TCNT_LKEN_LKCH(15);
        regValTemp |= (uint16_t) DMA_CTS_TCNT_LKEN_LKEN(1);
        DMA0->CTS[dmaChannel].TCNTRV = regValTemp;
        DMA0->CTS[dmaChannel].TCNT.LKEN = regValTemp;

    }

    return retStatus;
}
```

云途
YUNTU

THANKS

高品质　高性能　高可靠性　高一致性

业务合作：business@ytmicro.com
苏州办公室：苏州市高新区长江路211号天都大厦北楼39楼
上海办公室：上海市浦东新区祥科路58号A栋610