IDMA descriptor linked-list
IDMA descriptor 1:
10040, 20, 0x39f32180
0x7f85d91000, 0x3913b000, 0x4179e0, 0x390d5000
IDMA descriptor 2:
10000, 20, 0x39f32540
0x7f85d90000, 0x3a37b000, 0x41cd90, 0x3913b000
IDMA descriptor 3:
10003, 2e, 0x39f321c0
(nil), (nil), 0x41cef0, 0x3a37b000

启动传输后

JBIG IDMA Reg Dump:
cfg = 0x10500
status = 0x11b
line_width = 0
int_en = 0x3f
int_pend = 0
int_ack = 0
desc_write = 0
desc_read = 0x390d5000
xfer_length = 0xf32120
xfer_addr = 0x50
ctrl_word = 0x10040
reset = 0

---

analysis

desc_read = 0x390d5000
IDMA正在处理IDMA descriptor 1

xfer_length = 0xf32120
这个长度非法 (整个data length 为0x20)

xfer_addr = 0x50
这address肯定非法

发现一个bug

```
 typedef struct jbig_idma_descriptor_s
 {
     volatile uint32_t control;
-    volatile uint32_t *dataBuffer;
+    volatile uint32_t dataBuffer;
     volatile uint32_t length;
 //    volatile struct jbigDescriptor * next_desc_phys_addr;
 //    struct jbigDescriptor * next_desc_virt_addr;
 //    void * context;            ///< context pointer to allow recovery of callback and dma tracking info
 //    void* hw_addr;             ///< physical HW address of this descriptor;
 //                               ///needed to free with MEM_FREE_UNCACHED and needed when starting DMA
-} jbig_idma_descriptor_t;
+} __attribute__ ((packed)) jbig_idma_descriptor_t;

 typedef struct jbig_odma_descriptor_s
 {
     volatile uint32_t control;
     volatile uint32_t length;
-    volatile uint32_t *dataBuffer;
+    volatile uint32_t dataBuffer;
 //    volatile struct jbig_odma_descriptor_s* next_desc_phys_addr;
 //    struct jbig_odma_descriptor_s* next_desc_virt_addr;
 //    void* context;             ///< context pointer to allow recovery of callback and dma tracking info
 //    void* hw_addr;             ///< physical HW address of this descriptor;
 //                               ///needed to free with MEM_FREE_UNCACHED and needed when starting DMA

-} jbig_odma_descriptor_t;
```

在正常工作的32-bit environment上

IDMA descriptor 1:
10040, 20, 0x200f28c0
0xb6791000, 0x3003000, 0x16978, 0x501e000
IDMA descriptor 2:
10000, 20, 0x20163380
0xb6790000, 0x3008000, 0x18f90, 0x3003000
IDMA descriptor 3:
10003, 2e, 0x20198840
(nil), (nil), 0x19058, 0x3008000

当传输结束后
JBIG IDMA Reg Dump:
cfg = 0x10500

status = 0x13
line_width = 0
int_en = 0x3f
int_pend = 0
int_ack = 0
desc_write = 0
desc_read = 0x3008000
xfer_length = 0
xfer_addr = 0x20198860
ctrl_word = 0x10003
reset = 0

analysis
desc_read = 0x3008000
desc_read指向最后一个dma descriptor
xfer_length = 0
所有data已经都传输完毕，所以xfer_length为0

但有个疑问
xfer_addr = 0x20198860　???
IDMA descriptor 3
10003, 2e, 0x20198840
即data buffer的物理地址为 `0x20198840` ，长度为0x2e
即地址范围为[0x20198840, 0x2019886e),

cfg = 0x10500,即IDMA的BURST_LEN = 00, 4 words, sizeof(word) = ?
0x2019886e - xfer_addr = 0x2019886e - 0x20198860 = 0xe = 14
这怎么理解呢???

The root cause of the bug

original code

```
typedef struct jbig_idma_descriptor_s
{
    volatile uint32_t control;
    volatile uint32_t *dataBuffer;
    volatile uint32_t length;
//    volatile struct jbigDescriptor * next_desc_phys_addr;
//    struct jbigDescriptor * next_desc_virt_addr;
//    void * context;            ///< context pointer to allow recovery of c
allback and dma tracking info
//    void* hw_addr;             ///< physical HW address of this descriptor
;
//                               ///needed to free with MEM_FREE_UNCACHED an
d needed when starting DMA
} jbig_idma_descriptor_t;

typedef struct jbig_odma_descriptor_s
{
    volatile uint32_t control;
    volatile uint32_t length;
    volatile uint32_t *dataBuffer;
//    volatile struct jbig_odma_descriptor_s* next_desc_phys_addr;
//    struct jbig_odma_descriptor_s* next_desc_virt_addr;
//    void* context;             ///< context pointer to allow recovery of c
allback and dma tracking info
//    void* hw_addr;             ///< physical HW address of this descriptor
;
//                               ///needed to free with MEM_FREE_UNCACHED an
d needed when starting DMA

} jbig_odma_descriptor_t;

typedef struct jbig_descriptor_s
{
    // The following union contains the three words of either the idma
    // or odma descriptor.
    union
    {
        jbig_idma_descriptor_t idma;
        jbig_odma_descriptor_t odma;
    };

    volatile struct jbig_descriptor_s * next_desc_phys_addr;
    struct jbig_descriptor_s* next_desc_virt_addr;
    void* context;             ///< context pointer to allow recovery of cal
lback and dma tracking info
    void* hw_addr;             ///< physical HW address of this descriptor;
                               ///needed to free with MEM_FREE_UNCACHED and
needed when starting DMA
} __attribute__ ((aligned (32))) jbig_descriptor_t;
```

fixed code

```
typedef struct jbig_idma_descriptor_s
{
    volatile uint32_t control;
    volatile uint32_t dataBuffer;
    volatile uint32_t length;
//    volatile struct jbigDescriptor * next_desc_phys_addr;
//    struct jbigDescriptor * next_desc_virt_addr;
//    void * context;            ///< context pointer to allow recovery of callback and dma tracking info
//    void* hw_addr;             ///< physical HW address of this descriptor;
//                               ///needed to free with MEM_FREE_UNCACHED and needed when starting DMA
} __attribute__ ((__packed__)) jbig_idma_descriptor_t;

typedef struct jbig_odma_descriptor_s
{
    volatile uint32_t control;
    volatile uint32_t length;
    volatile uint32_t dataBuffer;
//    volatile struct jbig_odma_descriptor_s* next_desc_phys_addr;
//    struct jbig_odma_descriptor_s* next_desc_virt_addr;
//    void* context;             ///< context pointer to allow recovery of callback and dma tracking info
//    void* hw_addr;             ///< physical HW address of this descriptor;
//                               ///needed to free with MEM_FREE_UNCACHED and needed when starting DMA

} __attribute__ ((__packed__))  jbig_odma_descriptor_t;

typedef struct jbig_descriptor_s
{
    // The following union contains the three words of either the idma
    // or odma descriptor.
    union
    {
        jbig_idma_descriptor_t idma;
        jbig_odma_descriptor_t odma;
    };

    volatile uint32_t next_desc_phys_addr;      // The field also belongs to hardware [IO]DMA descriptor

    struct jbig_descriptor_s* next_desc_virt_addr;
    void* context;             ///< context pointer to allow recovery of callback and dma tracking info
    void* hw_addr;             ///< physical HW address of this descriptor;
                               ///needed to free with MEM_FREE_UNCACHED and needed when starting DMA
} __attribute__ ((aligned (32))) jbig_descriptor_t;
```

问题在与jbig_descriptor_t中的如下fields

```
    union
    {
        jbig_idma_descriptor_t idma;
        jbig_odma_descriptor_t odma;
    };

    volatile struct jbig_descriptor_s * next_desc_phys_addr;
```

这些fields是完全与jbig silcon block中[IO]DMA descriptop的layout完全对应的。但在code中

```
volatile uint32_t *dataBuffer;
volatile struct jbig_descriptor_s * next_desc_phys_addr;
```

都被定义成pointer，在32-bit environment下是4 bytes，没有什么问题(与hardware layout match)，
但在64-bit environment下，pointer变成了9 bytes，就与hardware layout mismatch,自然jbig block没法处理了。