DMA 项目文档

组长：段江飞 组员：资威、张丽玮

一、 项目内容

DMA 是计算机中不可缺少的一个部分，其主要实现内存中的数据与 CPU 中数据的交互。支持从内存中取出数据传入 CPU 中，也支持从 CPU 中取出数据传入内存中。DMA 中需要有 2 个数据的缓冲区 BUF1 和 BUF2，工作在不同的状态。例如：传输的方向为内存->CPU 时，BUF1 在接收内存传入的数据，BUF2 在向 CPU 输出数据；只有当 BUF1 满了，BUF2 空了之后，BUF1 才和 BUF2 交换，交换之后，BUF2 接收内存传入的数据，BUF1 向 CPU 输出数据；如此交替下去，实现流水。

二、 整体设计

DMA 与 CPU 之间可进行双向数据传输，其基本过程相似，设计时利用复位信号来控制数据传输的方向，每次复位数据传输方向发生改变，初始化的数据传输方向为从 MEM 到 CPU 方向。

数据传输过程类似于两个 FIFO 的联结，设计时参考 FIFO 的设计，利用三段式状态机实现功能。

DMA 的两个 BUF 在传输过程中要实现交换，根据 BUF1 和 BUF2 的缓冲区空、缓冲区满的状态将状态机分为 6 个状态（利用独热码），实现单方向数据传输时的流水作业。

整体程序分为四段，第一段是时序逻辑，实现次态到现态的迁移；第二段是组合逻辑，实现各个状态的转换；第三段是时序逻辑，用来实现 DMA 的主体功能；第四段是组合逻辑，用来辅助控制 DMA 的功能和状态。

三、 模块接口与参数

1. 接口

Input 与 Output 是参照实验要求的接口，未进行更改，接口及功能如下。

(1)Input 接口

clk：时钟信号，时序逻辑取其上升沿触发（1bit）；

rst_n：复位信号，在为 0 时进行复位操作（1bit）；

mem_to_dma_valid：mem 传入的数据是否有效（1bit）；

mem_to_dma_enable：mem 是否准备好接受数据（1bit）；

cpu_to_dma_valid：cpu 传入的数据是否有效（1bit）；

cpu_to_dma_enable：cpu 是否准备好接受数据（1bit）；

mem_data_out：mem 传出的数据（4bit）在 valid 和 enable 同时为 1 的时候才会被 dma 读入；

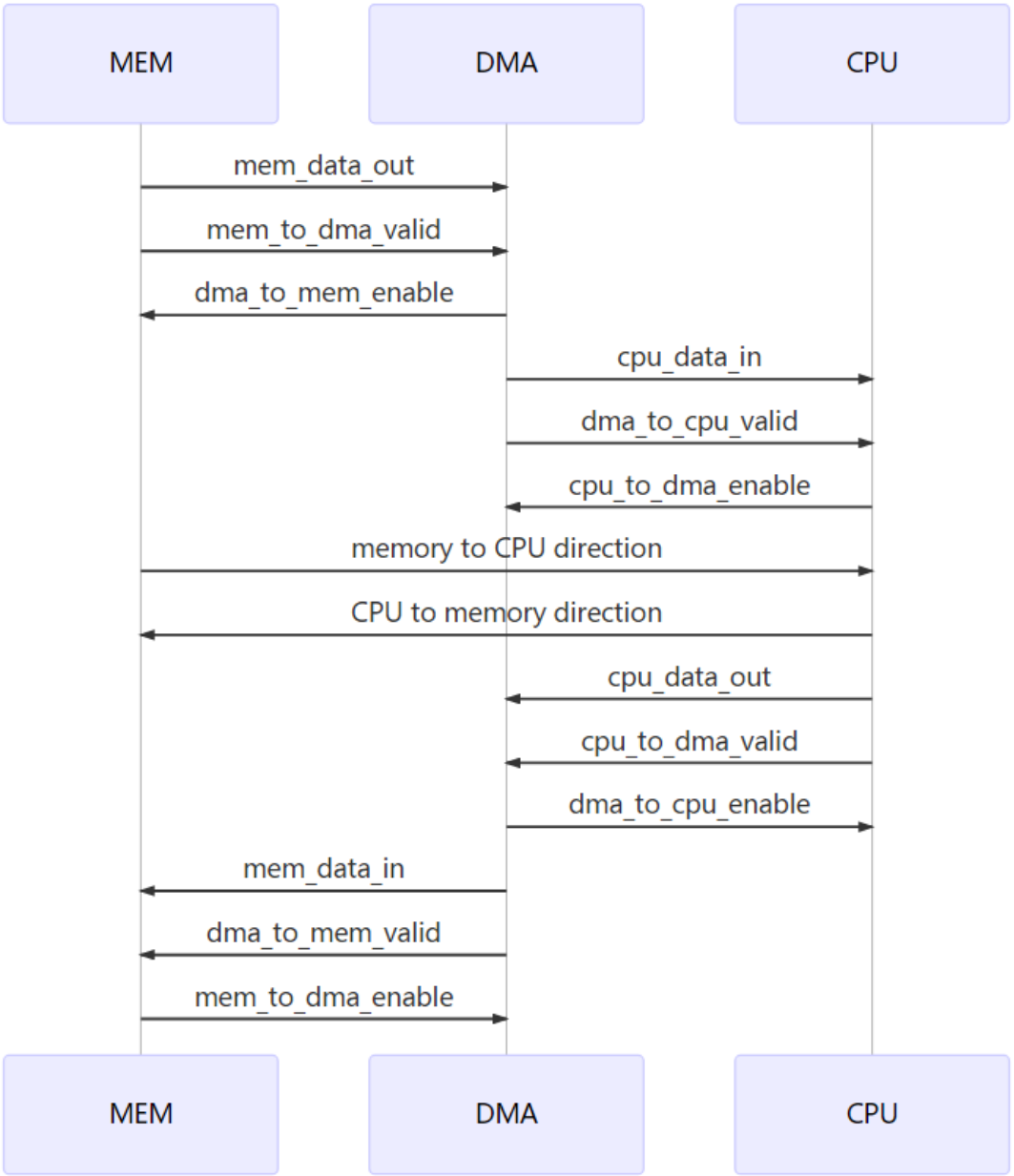
cpu_data_out：cpu 传出的数据（8bit）在 valid 和 enable 同时为 1 的时候才会被 dma 读入；

（2）Output

dma_to_mem_valid：dma 传入的数据是否有效（1bit）；

dma_to_mem_enable：dma 是否准备好接受数据（1bit）；

dma_to_cpu_valid：dma 传入的数据是否有效（1bit）；
dma_to_cpu_enable：dma 是否准备好接受数据（1bit）；
mem_date_in：dma 传入 mem 的数据，只有在 valid 和 enable 同时为 1 的时候才会被读入（4bit）；
cpu_data_in：dma 传入 cpu 的数据，只有在 valid 和 enable 同时为 1 的时候才会被读入（8bit）；



2. 状态

现态和次态表示使用 currentstate（当前状态 6bit）和 nextstate（下一状态 6bit）；
利用独热码和 BUF1、BUF2 的状态划分 6 个状态，前三个状态是在向 buf1 中写入数据，在 buf2 中读取数据。BUF 的空满状态根据有效数据判断，有效数据

是指已经写入 dma 但是还有被读取的数据。

前三个状态是向 buf1 写入数据，从 buf2 中读取数据

S1:buf1 未满（需要写入数据），buf2 未空（需要读出数据）；

S2:buf1 满，buf2 未空；

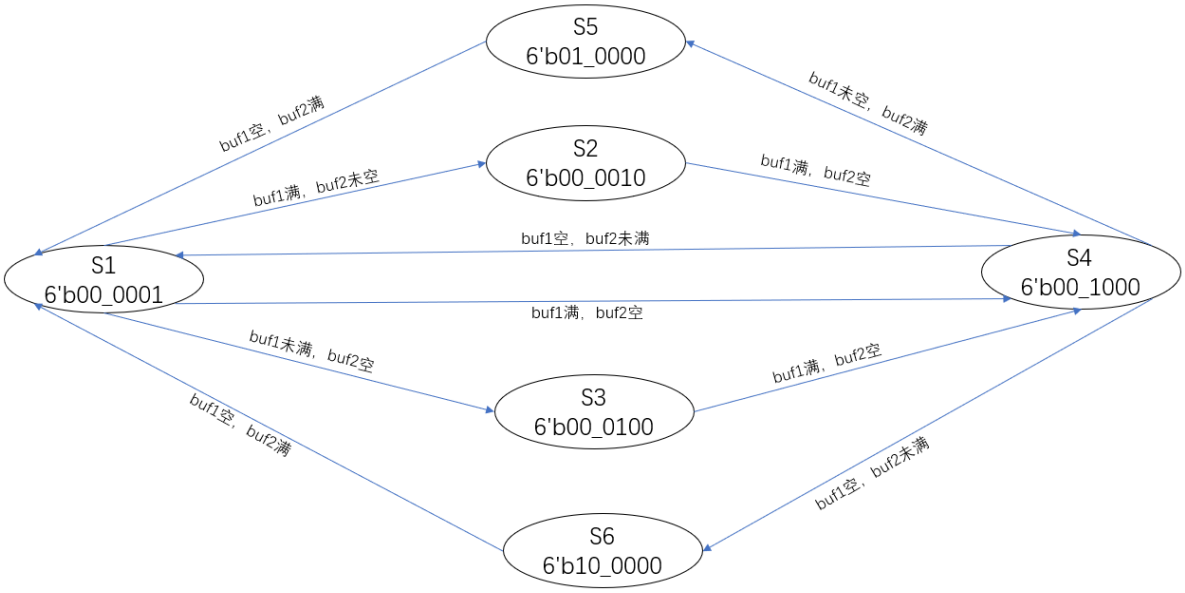
S3:buf1 未满，buf2 空；

后三个状态是在向 buf2 中写入数据，从 buf1 中读取数据。

S4:buf1 未空，buf2 未满；

S5:buf1 未空，buf2 满；

S6：buf1 空，buf2 未满；



3. 其他参数

传输方向

data_flow_direction:控制数据传输方向，为 0 时从 CPU 到 MEM，为 1 时从 MEM 到 CPU；每次复位时，data_flow_direction 信号取反

DMA 缓冲区

buf1[7:0][7:0]:缓存区 1（$8 \cdot 8bit = 64bit$）；

buf2[7:0][7:0]:缓存区 2（$8 \cdot 8bit = 64bit$）；

BUF 读出、写入指针

buf1 和 buf2 的读取指针，指示 MEM 或 CPU 读取的位置，每次增到 8 时自动舍去高位变为 0。

buf1_read_ptr:用来负责 buf1 输出工作的指针（3bit）；

buf2_read_ptr:同上，负责 buf2（3bit）；

buf1 和 buf2 的写入指针，指示 MEM 或 CPU 写入的位置，每次增到 8 时自动舍去

高位变为 0。

buf1_write_ptr:负责 buf1 写入 dma 的操作的指针（3bit）；

buf2_write_ptr:同上，负责 buf2（3bit）；

MEM 的数据是 4bit 传输，buf1 和 buf2 的每个寄存器是 8bit，需要有一个信号来标志读写的是高位还是低位。

buf1_write_low:为 1 代表 buf1 在写入时写的是低位，每次操作后取反（1bit）；

buf1_read_low:同上，从 buf1 读取数据时用（1bit）；

buf2_write_low:同上，buf2 写入数据时用（1bit）；

buf2_read_low:同上，buf2 读取数据时用（1bit）；

计数

buf1 和 buf2 中的数据个数（按 4bit 计）要进行计数，便于输入输出和状态转换

counter_buf1:用来计数 buf1 中的有效数据（5bit）；

counter_buf2:同上，计数 buf2 中的有效数据（5bit）；

辅助功能

DMA 是两个同步的 FIFO，要实现数据传输过程中 buf1 和 buf2 的瞬间翻转并接受输入和输出功能，设置了下面接个参数来及时控制信号的改变，避免数据传输的延迟。

buf1willfull:当 buf1 在填数据的时候用，当 buf1 在这个时钟上升沿录入了最后的数据，这个值就设为 1，用来帮助转换状态（去除状态转换的那一格的延迟）（1bit）；

buf2willempty:当 buf2 在弹出数据的时候用的，弹出最后一个数据的时钟上升沿时置为 1（1bit）；

buf1willempty:同上（1bit）；

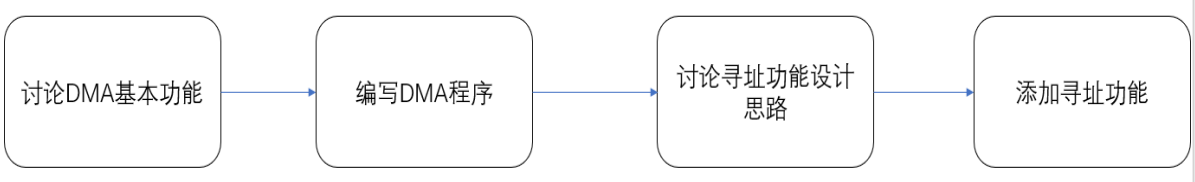
buf2willfull:同上（1bit）；

cpu_in_finish:在 mem 到 cpu 的方向，如果读入已经读完了，但是还没有输出完就会进入 S2 或者 S5，这时候最后一个数据一进入 dma 的输出端口状态就会改变但是最后一个值还没有被 cpu 读进去，等最后一个值被录进去了，这个值就会为 1，从而把 valid 拉低（1bit）；

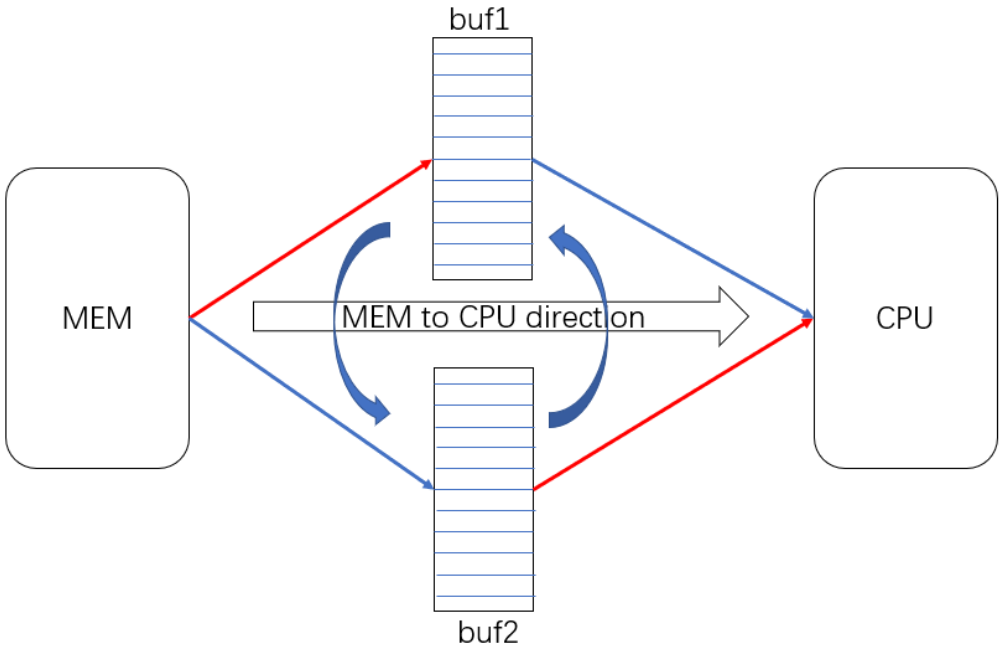
mem_in_finish:同上，用于 cpu 到 mem 的方向（1bit）；

四、 流程图

小组合作过程



DMA 设计



以 MEM 到 CPU 方向为例，初始化时 buf1 和 buf2 均为空，进行红色箭头方向传输，buf2 不输出，buf1 满了之后 buf1 和 buf2 进行调转，进行蓝线方向输出。

五、 实现基本功能

本项目的基本功能部分实现了一个残缺版的 DMA，只有当一个 buf 数据存满并且另外一个 buf 完全输出时，两个 buf 的功能才会转换。

六、 亮点

1. 每次复位时改变数据的传输方向，实现手动控制 MEM 到 CPU 和 CPU 到 MEM 方向的数据传输。
2. 利用三段式状态机实现 DMA 的功能，将组合逻辑和时序逻辑分离开，更方便程序的优化和维护，更符合思维习惯。
3. 利用独热码，实现状态的转换，容错性高。
4. 实现无延迟状态转换，简单来说就是一个缓冲区满时，瞬间交换状态并同时输出，不会出现延迟，为了实现这个功能我们对最初版本的 DMA 程序做了比较大的修改。

   比如说，在 mem 到 cpu 的方向，最初版本的 DMA 从 S3 到 S4，当 buf1 满了之后还会等一个时钟上升沿到来才会把 dma_to_cpu_valid 拉高，然后把 buf1 的最初的数据输出。新版本的 DMA 在 buf1 填满的那个时钟上升沿就能把 dma_to_sth_valid 拉高，并且把 buf1 的最初的数据放在 cpu_data_in 里等待 cpu 的读取，这个功能是使用 buf1willfull、buf1willempty、buf2willfull、buf2willempty 控制状态跳变来实现的。比如还是从 mem 到 cpu 的方向，在 S3 状态，buf1 在一个时钟上升沿写入了最后的数据，那么就把 buf1willfull 拉高，然后在第二块的状态转换的组合逻辑中是有

buf1willfull 的，它的拉高会让 nextstate 瞬间变成 S4，而第四段的组合逻辑再根据状态 S1 将 dma_to_cpu_valid 瞬间拉高，那么如果下个上升沿 cpu_to_dma_enable 正好也是 1(因为 cpu_to_dma_enable 为了保证鲁棒性，在 testbench 里面设的是随机值)，就可以直接让 cpu 取到数据了。

5. DMA 能够实现把 MEM 或者 CPU 输入的数据完全读出再进行下一步操作。这是在实现无延迟是加入的变量 cpu_in_finish 和 mem_in_finish 实现的，因为我们的 dma 的状态是没有延迟的，组合逻辑也是用的 nextstate 的，所以当写入 dma 已经结束但是还没有被 mem（或者 cpu）读取干净，就会在把最后一个数据放入 mem_data_in(或者 cpu_data_in)的时候状态就会有所改变，但是 mem 和 cpu 是在时钟上升沿才会读取 mem_data_in 和 cpu_data_in 的数据，所以我要等最后一个数据被 mem 或者 cpu 读取之后再将 valid 置 0，那么这个 dma 的功能就完善了。
6. 程序的鲁棒性好。

附加功能——寻址的实现

一、功能简述

DMA 主要实现的是内存中数据和 CPU 中数据的交互，而在实际操作中，CPU 从内存中取数据需要传递一个信号告诉内存要取得数据的地址和长度，这就是寻址操作。附加功能中实现的寻址功能是简化版本，由于内存没有地址信息，我们所用的有效信息就是 CPU 传递的所取得数据长度信息。

寻址功能依靠 burst 进行传输，每次 burst 的长度限制为 256bit，根据用户输入的所需数据的字节取数据，如果长度超过了 burst 的上限，就分多个 burst 进行，每个 burst 过程，数据超过 BUF 的上限时进行多次 BUF 调换传输。

二、整体设计

根据已经实现的 DMA 功能，在其中加入输入接口 Length，随机以表示用户的需求，两个和地址有关的参数 valid 和 enable，分别随即表示长度的有效性和此时是否能读入用户需求的长度。之后运用对输入和输出数据的分别统计，来表示此时是否超出 burst，当输入已达到上限值而输出未到时，及时跳转状态，停止输入，而等待输出结束后进入下一轮输入输出。

三、增添的接口与参数

1、接口

Input 与 Output 是参照实验要求的接口，未进行更改，接口及功能如下。

(1) Input 接口

length：用户所需数据长度，在 testbench 中进行随机读取（11bit）；

address_valid：length 传入的数据是否有效（1bit）；

（2）Output 接口

address_enable：此时 mem 是否可以接受 cpu 传输的长度数据（1bit）；

2、其他参数

address_in_enable：表示此时是否可以从内存向 dma 输入数据（1bit）

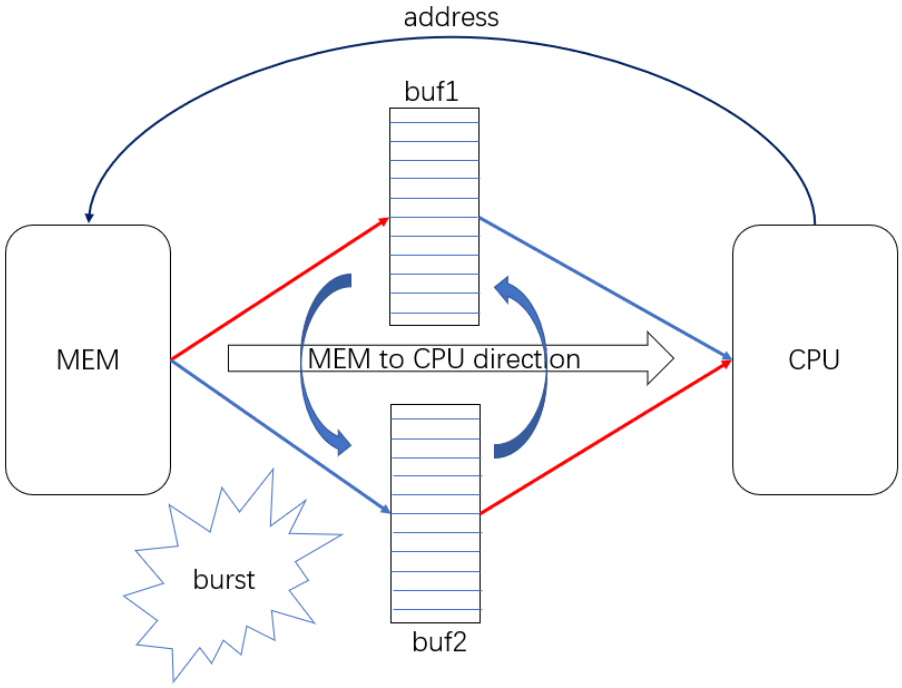
counter_address_in：统计已经从内存输入 dma 的数据数（11bit）

counter_address_out：统计已经从 dma 输出到 cpu 的数据数（11bit）

return_S3：当其为 1 的时候回到 S3（1bit）

burst_will_finish：表示 burst 是否将要取满（1bit）

四、流程图



五、程序设计思路

1、首先随机读入 length 和 address_valid，当 address_valid 和 address_enable 均为 1 时表示此时 length 可以读入，用户需求有效并且内存可以满足此需求。这时将 counter_address_in 的值置为 0，而 counter_address_out 的值置为 length/4（由于 length 表示的是 bit，而 counter 表示的是位）。

2、在第三段时序逻辑中，加入对于 counter_address_in 和 counter_address_out 的判断，若输入和输出都已经完成，则 address_enable 将被置为 1，而 address_in_enable 也将被置为 1；否则，若 counter_addreass_in 输入完成，但是 counter_address_out 不为 0，即输出未完成，address_in_enable 将被置为 0。同样，在第三段时序逻辑中，每一次有效数据的输入，counter_buf+1 时，counter_address_in 将为同时+1，而 counter_buf-2 时，counter_address_out 将为同时-2。

3、在第二段组合逻辑实现状态变换中，加入了新的判断条件，即当输入已经完成而输出未完成时，将为跳转为 S2 或 S5 来进行只输出不输入的操作。还有当一次 length 操作完成时，将会回到 S3，通过 return_S3 参数控制

五、亮点

可以实现较长 length 的有效稳定输入，实现指定长度数据的有效输出，并且同样保证了无延迟输入输出。

编程缺点

程序设计风格和硬件设计不是特别符合，习惯于 C 语言的软件设计，编码风格有待改进。

Contribution			
原版 DMA	改进版 DMA	寻址功能	项目文档
段江飞	资 威	资 威，张丽玮	段江飞，资 威，张丽玮
设计思路		段江飞，资 威，张丽玮	