多通道数据整形器设计

多通道数据整形器 MCDF（Multi-Channel Data Formatter），将多个通道的上行(umlink)数据经过内部的 FIFO 以数据包(data packet)的形式送出。其结构如图 1 所示。

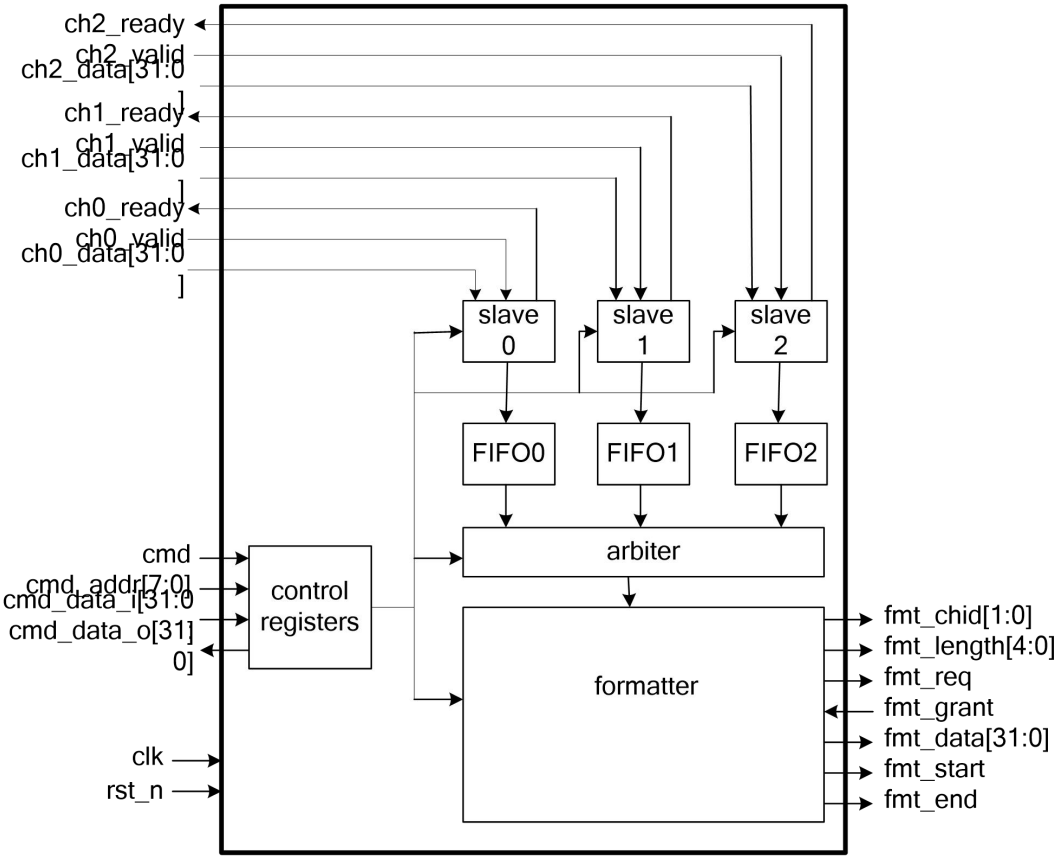

图 1 MCDF 的设计结构(fmt_length 应该是 6 位，cmd_addr 只需要 6 位)

1、 MCDF 外部接口

1.1 系统信号接口

- clk ： 时钟信号
- rst_n： 复位信号，低电平有效

1.2 通道从端(slave)接口 （x=0, 1, 2）

- chx_data[31:0]： 通道数据输入
- chx_valid ： 通道数据有效指示信号，高电平有效
- chx_ready ： 通道数据接收信号，高电平表示接收成功

1.3 整形器(formatter)接口信号

- fmt_chid[1:0] ： 整形数据包的通道 ID 号
- fmt_length[4:0]： 整形数据包长度信号
- fmt_req ： 整形数据包发送请求

- fmt_grant ： 整形数据包被允许发送的接收指示
- fmt_data[31:0]： 数据输出信号
- fmt_start ： 数据包起始指示信号
- fmt_end ： 数据包结束指示信号

1.4 控制寄存器(control register)接口

- cmd[1:0] ： 寄存器读写命令
- cmd_addr[7:0] ： 寄存器地址
- cmd_data_i[31:0] ： 寄存器写入数据
- cmd_data_o[31:0] ： 寄存器读出数据

2、 MCDF 各模块接口信号及时序

2.1 通道从端(slave)

一个通道从端包括图 1 中的 slaveX 及 FIFOX（X=0，1，2）。其中 FIFO 深度为 64。

通道从端从外部接口接收数据，当接收到一个完整的数据包（到底多大？怎么知道 fmt 要多大？默认 32, 由 2.3.3 中所述的通道 X 的控制寄存器 bit[5:3]决定）后，则向 arbiter 发出发送请求。若请求信号得到响应，则开始发送，直至整个数据包全部发送完成后，再根据情况确定是否发出发送数据请求（是否数据多余 32）。

（一直能写，满了不 ready）

2.1.1 通道从端接口信号

通道从端接口信号如下表所示同，包括系统信号、MCDF 外部接口信号、register 模块接口信号、arbiter 模块接口信号。

信号名称	方向	功能	备注
clk_i	input	Clock input	系统信号
rstn_i	input	low level effective reset	
chx_data_i[31:0]	input	Data input	外部接口
chx_valid_i	input	Data is valid From outside	
chx_ready_o	output	Ready to accept data	
slvx_en_i	input	Write enable To FIFO	register 接口
margin_o[5:0]	output	Data margin	
slvx_data_o[31:0]	output	Data Output to arbiter	Arbiter

slvx_val_o	output	data valid to Arbiter	
slvx_req_o	output	required send data to arbiter	
a2sx_ack_i	input	Read acknowledge	

2.1.2 通道从端接口时序

如图 2 所示。当 chx_valid 信号为高时，表示写入数据有效。此周期 chx_data 应给出要写入的数据。若该时钟周期 chx_ready 为高，则表示已经将数据写入。若此时钟周期 chx_ready 信号为低，则表示数据还未写入，需要等待 chx_ready 为高时才将数据写入。

注意：后面其它模块的 xx_valid 及 xx_ready 时序均与图 2 中相同。

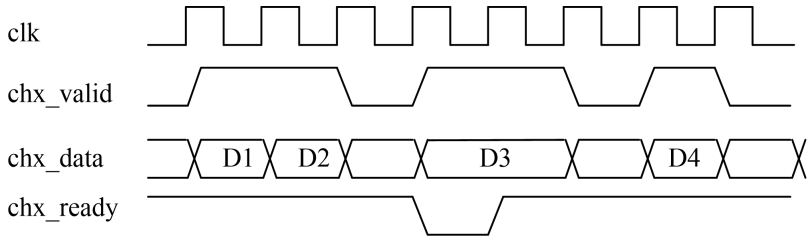

图 2 通道从端接口时序

2.2 arbiter

2.2.1 接口信号

如果 formatter 的发送数据请求信号 slv0_val_i（应该是 f2a_rd _req_i？）为高，则 arbiter 根据 slave0_FIFO 的发送请求信号 slv0_req_i、slave1_FIFO 发送请求信号 slv1_req_i、slave2_FIFO 发送请求信号 slv2_req_i，按优先级确定响应通道的发送请求，并根据通道的编号（id=0, 1, 2）X 产生以下信号送到 formatter：

a2f_id_o = X(通道编号)

a2f_data_o = slvX_data_i

a2f_pkglen_sel_o = slvX_pkglen_i

a2f_val_o = slvX_val_i

将 formatter 的响应信号送往对应的 slave 通道：

a2sX_ack_o = f2a_ack_i

如果各通道的优先级相同，则按通道编号从低到高轮流发送。

arbiter 模块的接口信号如下：

信号名称	方向	功能	备注
clk_i	input	Clock input	系统信号
rstn_i	input	low level effective reset	
slv0_prio_i[1:0]	input	slave 0 priority	控制寄存器的信号
slv0_pkglen_i[2:0]	input	slave 0 package length	
slv1_prio_i[1:0]	input	slave 1 priority	
slv1_pkglen_i[2:0]	input	slave 1 package length	
slv2_prio_i[1:0]	input	slave 2 priority	
slv2_pkglen_i[2:0]	input	slave 2 package length	
slv0_data_i[31:0]	input	slave 0 data	slave 接口
slv0_req_i	input	slave 0 required	
slv0_val_i	input	slave 0 data valid	
a2s0_ack_o	output	slave 0 Read acknowledge	
slv1_data_i[31:0]	input	slave 1 data	
slv1_req_i	input	slave 1 required	
slv1_val_i	input	slave 1 data valid	
a2s1_ack_o	output	slave 1 Read acknowledge	
slv2_data_i[31:0]	input	slave 2 data	
slv2_req_i	input	slave 3 required	
slv2_val_i	input	slave 3 data valid	
a2s2_ack_o	output	slave 2 Read acknowledge	
f2a_id(rd)_req_i	input	formatter read required	formatter 接口
f2a_ack_i	input	formatter acknowledge	
a2f_val_o	output	arbiter data valid	
a2f_id_o[1:0]	output	slave id(0,1,2)	
a2f_data_o[31:0]	output	data output to formatter	
a2f_pkglen_sel_o[2:0]	output	data package length	

2.3 整形器(formatter)

2.3.1 整形器接口信号

信号名称	方向	功能	备注
clk_i	input	Clock input	系统信号
rstn_i	input	low level effective reset	
f2a_ack_o	output	formatter acknowledge	arbiter 接口
fmt_id(rd)_req_o	output	formatter read required	
a2f_val_i	input	arbiter data valid	
a2f_id_i[1:0]	input	slave id(0,1,2)	
a2f_data_i[31:0]	input	data input from arbiter	
pkglen_sel_i[2:0]	input	data package length	
fmt_grant_i	input	Read acknowledge	外部接口
fmt_chid_o[1:0]	output	slave id(0,1,2)	
fmt_length_o[5:0]	output	data package length	
fmt_req_o	output	data output required	
fmt_data_o[31:0]	output	data output	
fmt_start_o	output	first data indicate	
fmt_end_o	output	last data indicate	

2.2.2 整形器接口时序

如图 3 所示。整形器按照数据包的形式发送数据。数据包的长度可以是 4、8、16 和 32。整形器必须完整的发送某一个通道的数据包后，才可以准备发送下一个数据包。在发送数据包期间，fmt_chid 和 fmt_length 应保持不变，直到数据包发送结束。

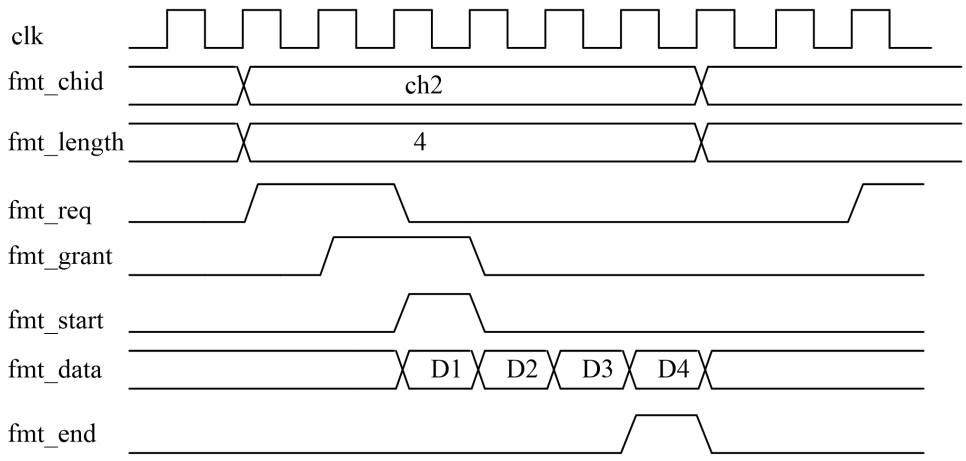

图 3 整形器接口时序

整形器准备发送数据包时，先将 fmt_req 置高，等待接收端的 fmt_grant 信号。当 fmt_grant 信号变为高时，则在下一个周期将:

（1） fmt_req 置低

（2） fmt_start 输出一个时钟周期的高电平脉冲

（3） fmt_data 开始送出第一个数据

数据开始发送后应连接发送，中间不允许有空闲周期，直到发送完所有数据。在发送最后一个数据时，fmt_end 信号应置高并保持一个时钟周期。

相邻的数据包之间应至少有一个时钟周期的空闲，即 fmt_end 变为低电平后，至少经过一个时钟周期 fmt_req 才可以再次置高。

2.3 控制寄存器(control register)

2.3.1 接口时序

控制寄存器接口时序如图 4 所示。在每个时钟周期根据 cmd 命令完成指定操作。当 cmd 为写指令时，将数据 cmd_data_i 写入 cmd_addr 指定的寄存器中。当 cmd 为读指令时，从 cmd_addr 指定的寄存器中的数据读出，并在下一个时钟周期送到 cmd_data_o 端口。当 cmd 为其它指令时不进行任何操作。

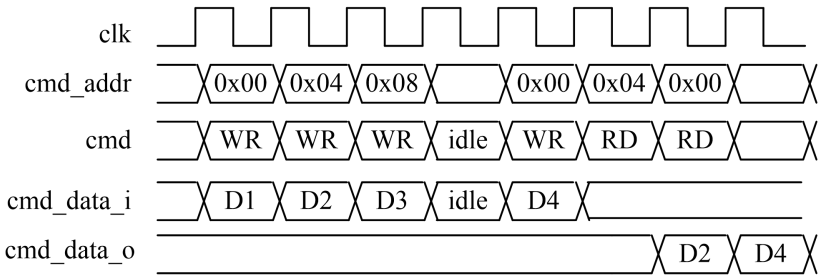

图 4 控制寄存器接口时序

2.3.2 接口信号

信号名称	方向	功能	备注
clk_i	input	Clock input	系统信号
rstn_i	input	low level effective reset	
cmd_i[1:0]	input	read or write control	外部接口
cmd_addr_i[5:0]	input	command address	
cmd_data_i[31:0]	input	command data input	
cmd_data_o[31:0]	output	command data output	
slv0_margin_i[5:0]	input	slave0 channel margin	slave 接口
slv0_en_o	output	slave 0 enable	
slv1_margin_i[5:0]	input	slave 1 channel margin	
slv0_en_o	output	slave 1 enable	
slv2_margin_i[5:0]	input	slave 2 channel margin	
slv0_en_o	output	slave 2 enable	
slv0_pkglen_o[2:0]	output	slave 0 package lenth	arbiter 接口
slv0_prio_o[1:0]	output	slave 0 priority	
slv1_pkglen_o[2:0]	output	slave 1 package lenth	
slv1_prio_o[1:0]	output	slave 1 priority	
slv2_pkglen_o[2:0]	output	slave 2 package lenth	
Slv_1_prio_o[1:0]	output	slave 2 priority	

2.3.3 控制寄存器模块中的寄存器

- cmd_addr 地址 0x00：通道 slave0 的控制寄存器，32bit，可读写，位定义为：

- bit[0] ：通道使能信号。1 为打开，0 为关闭。复位值为 1。
- bit[2:1] ：优先级。0 为最高，3 为最低。复位值为 3。
- bit[5:3] ：数据包长度。0 对应长度 4，1 对应 8，2 对应 16，3 对应 32。其它数值均暂时对应 32。复位值为 0。
- bit[31:6] ：保留位，不能写入。复位值为 0。

- cmd_addr 地址 0x04：通道 slave1 的控制寄存器，32bit，可读写，位定义同 slave0。
- cmd_addr 地址 0x08：通道 slave2 的控制寄存器，32bit，可读写，位定义同 slave0。
- cmd_addr 地址 0x12（不是 0x0c 吗？）：通道 slave0 的状态寄存器，32bit，只读，位定义为：
  - bit[7: 0] ：从端 FIFO0 的可写余量，实时同步 FIFO0 的余量，复位值为 FIFO 深度值。
  - bit[31:8] ：保留位，复位值为 0。
- cmd_addr 地址 0x16：通道 slave1 的状态寄存器，位定义同 slave0。
- cmd_addr 地址 0x20：通道 slave2 的状态寄存器，位定义同 sl