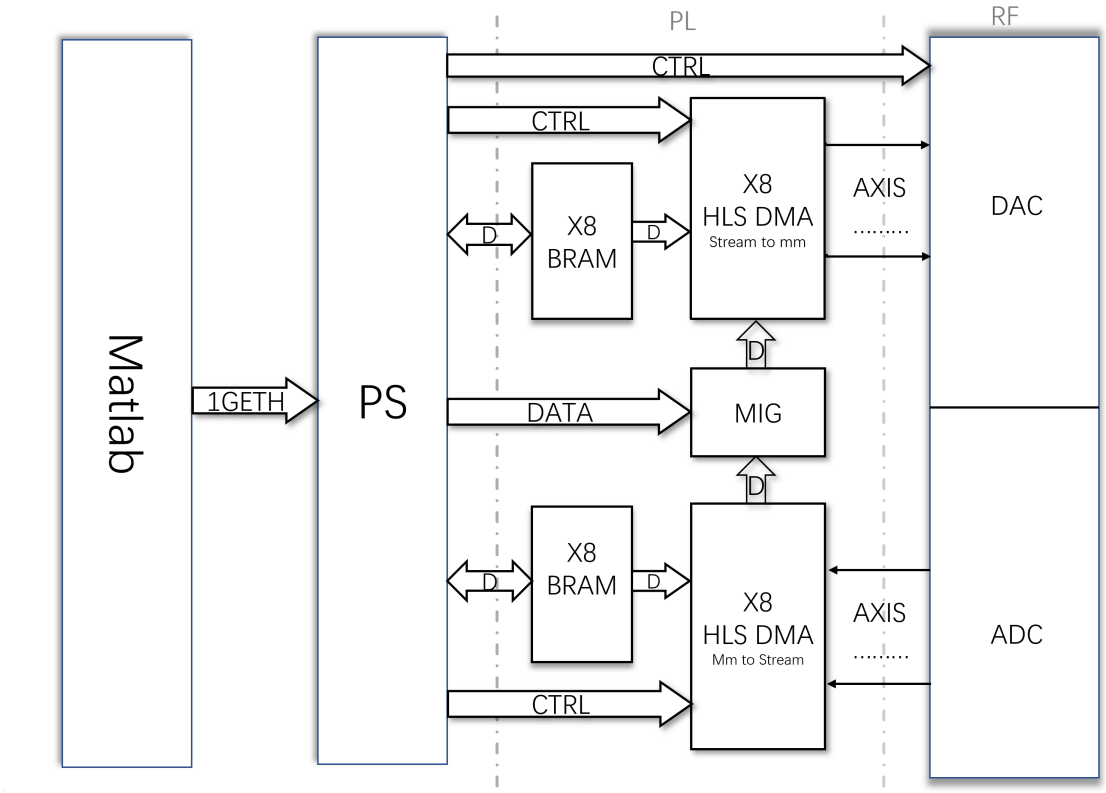
RFSoc Matlab联合工具说明

一.简介

RFSoc Matlab联合工具主要是实现Matlab发送和接收DAC和ADC的数据的功能。此工具支持八路DAC和ADC同时发送和接受数据[1]，且DAC数据支持同步输出。

二.框架图



三.模块说明

- Matlab：Matlab主要用于产生和接受DAC和ADC数据和控制HLS DMA。
- PS：PS主要是在程序启动时初始化时钟和启动LWIP。LWIP负责数据的解包和搬移。
- HLS_DMA：HLS_DMA负责将DAC数据从BRAM或者PL DDR4的对应地址中输出到DAC的AXIS接口，从ADC的AXIS获得数据然后搬移到对应地址的BRAM或者PL DDR4中，DAC的HLS_DMA在启动多路时是同步输出的。ADC的HLS_DMA的启动同样是同步的。
- BRAM：BRAM是存放ADC和DAC的数据。一共有16个BRAM支持八路ADC和DAC同时开启。
- PL_DDR4：PL_DDR4是为了DAC和ADC的大数据量存储。

四.功能说明

- DAC的波形数据由Matlab产生，通过千兆以太网将数据搬移PL端的BRAM或DDR4中（其中数据大小必须是256的整倍数[2]）,Ps端会使用PS DMA将从LWIP接收的数据搬移到PL端的BRAM或DDR4中，由于DDR4和总线带宽问题（带宽为8GB/s左右），在八路DAC全开的情况下请使用BRAM作为DAC数据的存储。HLS_DMA（Stream_to_mm)的启动也是由Matlab来控制，Matlab将每个HLS_DMA（每一路HLS_DAM对饮一个DAC）数据搬移地址和数据搬移长度配置完成后，统一启动DMA。
- ADC通过HLS_DMA（Mm_to_Stream）搬移到PL端的BRAM或者DDR4中，Matlab通过千兆以太网将数据读取并显示。HLS_DMA（Mm_to_Stream）同样是通过Matlab来控制，控制方式同DAC

一样。HLS_DMA接受ADC数据时可以在接收到所设的的长度后会将`done`寄存器置1，Matlab可以通过轮询`done`寄存器来判断数据时候接受完成。ADC和DAC一样由于带宽限制所以在启动八路时最好使用BRAM进行存储。

五.寄存器表

地址	SIZE(BYTE)	功能 说明
0x00_B000_0000	64K	DAC BRAM0 此BRAM仅支持 DAC HLS_DMA访问
0x00_B001_0000	64K	DAC BRAM1 此BRAM仅支持 DAC HLS_DMA访问
0x00_B002_0000	64K	DAC BRAM2 此BRAM仅支持 DAC HLS_DMA访问
0x00_B003_0000	64K	DAC BRAM3 此BRAM仅支持 DAC HLS_DMA访问
0x00_B00C_0000	64K	DAC BRAM4 此BRAM仅支持 DAC HLS_DMA访问
0x00_B00D_0000	64K	DAC BRAM5 此BRAM仅支持 DAC HLS_DMA访问
0x00_B00E_0000	64K	DAC BRAM6 此BRAM仅支持 DAC HLS_DMA访问
0x00_B00F_0000	64K	DAC BRAM7 此BRAM仅支持 DAC HLS_DMA访问
0x48_000F_0000	4G	PL DDR4 支持 DAC 和 ADC HLS_DMA的同时访问
0x00_B004_0000	64K	ADC BRAM7 此BRAM仅支持 ADC HLS_DMA访问
0x00_B005_0000	64K	ADC BRAM7 此BRAM仅支持 ADC HLS_DMA访问
0x00_B006_0000	64K	ADC BRAM7 此BRAM仅支持 ADC HLS_DMA访问
0x00_B007_0000	64K	ADC BRAM7 此BRAM仅支持 ADC HLS_DMA访问
0x00_B008_0000	64K	ADC BRAM7 此BRAM仅支持 ADC HLS_DMA访问
0x00_B009_0000	64K	ADC BRAM7 此BRAM仅支持 ADC HLS_DMA访问
0x00_B00A_0000	64K	ADC BRAM7 此BRAM仅支持 ADC HLS_DMA访问
0x00_B00B_0000	64K	ADC BRAM7 此BRAM仅支持 ADC HLS_DMA访问
0x00_A000_0000	---	DMA控制寄存器

- DMA控制寄存器表

名称	偏移量	SIZE(BIT)	功能
DMA控制	0x08	32	选通控制每路ADC和DAC HLS_DMA读取写入地址和长度

BIT	15-9	8	7	6	5	4	3	2	1	0
说明	-	1：数据锁存 0:不做操作	DAC7	DAC6	DAC5	DAC4	DAC3	DAC2	DAC1	DAC0

BIT	31-25	24	23	22	21	20	19	18	17	16
说明	-	1：数据锁存 0:不做操作	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0

名称	偏移量	读写	SIZE(BIT)	功能
DAC_DMA高地址位	0x0C	WO	32	DAC DMA读取数据的高地址位，通过DMA控制器来选择配置那一路DAC DMA
DAC_DMA低地址位	0x10	WO	32	DAC DMA读取数据的高地址位，通过DMA控制器来选择配置那一路DAC DMA
DAC_DMA读取长度	0x14	WO	32	DAC DMA读取数据长度，通过DMA控制器来选择配置那一路DAC DMA，实际读取长度是 寄存器的值*256bit
ADC_DMA高地址位	0x18	WO	32	ADC DMA写入数据的高地址位，通过DMA控制器来选择配置那一路ADC DMA
ADC_DMA低地址位	0x1C	WO	32	ADC DMA写入数据的高地址位，通过DMA控制器来选择配置那一路ADC DMA
ADC_DMA读取长度	0x20	WO	32	ADC DMA写入数据长度，通过DMA控制器来选择配置那一路ADC DMA，实际写入长度是 寄存器的值*256bit
DAC_DMA选通位	0x24	WO	8	数字0-7 对应 DAC0 - DAC7 ， 此寄存器是选择需要启动的DAC_DMA。
ADC_DMA选通位	0x28	WO	8	数字0-7 对应 ADC0 - ADC7 ， 此寄存器是选择需要启动的ADC_DMA。
DAC_DMA循环启动	0x40	WO	1	设置为1时所选通的DAC DMA循环启动（循环启动时数据不会因为再启动中断中断），设置为0则会在当前数据搬移后停止。
DAC_DMA单次启动	0x44	WO	1	设置为1时会启动一次DAC DMA。
ADC_DMA循环启动	0x80	WO	1	设置为1时所选通的ADC DMA循环启动（循环启动时数据不会因为再启动中断中断），设置为0则会在当前数据搬移后停止。
ADC_DMA单次启动	0x84	WO	1	设置为1时会启动一次ADC DMA。
DAC0 done信号	0xC0	RO	1	DAC0 DMA执行完毕信号，读取后置0
DAC1 done信号	0xC4	RO	1	DAC1 DMA执行完毕信号，读取后置0
DAC2 done信号	0xC8	RO	1	DAC2 DMA执行完毕信号，读取后置0
DAC3 done信号	0xCC	RO	1	DAC3 DMA执行完毕信号，读取后置0
DAC4 done信号	0xD0	RO	1	DAC4 DMA执行完毕信号，读取后置0

名称	偏移量	读写	SIZE(BIT)	功能
DAC5 done信号	0xD4	RO	1	DAC5 DMA执行完毕信号，读取后置0
DAC6 done信号	0xD8	RO	1	DAC6 DMA执行完毕信号，读取后置0
DAC7 done信号	0xDC	RO	1	DAC7 DMA执行完毕信号，读取后置0
DAC all done信号	0xC0	RO	8	DAC0-DAC7 DMA执行完毕信号，和DAC0 done - DAC7 done信号关联
ADC0 done信号	0xC0	RO	1	ADC0 DMA执行完毕信号，读取后置0
ADC1 done信号	0xC4	RO	1	ADC1 DMA执行完毕信号，读取后置0
ADC2 done信号	0xC8	RO	1	ADC2 DMA执行完毕信号，读取后置0
ADC3 done信号	0xCC	RO	1	ADC3 DMA执行完毕信号，读取后置0
ADC4 done信号	0xD0	RO	1	ADC4 DMA执行完毕信号，读取后置0
ADC5 done信号	0xD4	RO	1	ADC5 DMA执行完毕信号，读取后置0
ADC6 done信号	0xD8	RO	1	ADC6 DMA执行完毕信号，读取后置0
ADC7 done信号	0xDC	RO	1	ADC7 DMA执行完毕信号，读取后置0
ADC all done信号	0xC0	RO	8	ADC0-ADC7 DMA执行完毕信号，和DAC0 done - DAC7 done信号关联

六.Matlab函数说明

1. **波形数据的写入**

```
WriteMemoryToDDR(IP, addr, data);
```

- IP：RFSoc的IP地址，IP地址默认是192.168.2.10
- addr：数据要写入的物理地址
- data：波形数据

2. **波形数据读出**

```
data = ReadMemoryFromDDR(IP, addr, size);
```

- IP：RFSoc的IP地址，IP地址默认是192.168.2.10
- addr：数据读取的物理地址
- size：数据读取的长度，单位：BYTE

3. **寄存器写入**

```
SetDiscreteReg(IP,[addr,value]);
```

- IP：RFSoc的IP地址，IP地址默认是192.168.2.10
- addr：寄存器的物理地址
- value：寄存器的值

4. **寄存器读取**

```
value = GetDiscreteReg(IP,addr);
```

- IP：RFSoc的IP地址，IP地址默认是192.168.2.10
- addr：寄存器的物理地址
- value：寄存器的值

5. **配置DMA**

```
ConfigDMA(IP,'DAC0',addr_high,addr_low,size);
```

- IP：RFSoc的IP地址，IP地址默认是192.168.2.10
- 'DAC0'：这是输入想要控制的DAC或者ADC，例如：‘DAC1' 'ADC1'
- addr_high：DMA读写的高地址位
- addr_low：DMA读写的低地址位
- size：DMA读写的长度，值是数据长度➗32（单位：BYTE）