

图2以A实部运算为例，解释本次流水线过程。

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Describe** | **Name** | **Describe** |
| a\_re | A的实部 | b\_re\_c\_re | b\_re\*c\_re. |
| a\_re\_mid | a\_re\*2^13 | a\_re\_addend | b\_re\_c\_re-b\_im\_c\_im. |
| ar1’ | a\_re\_mid延一拍 | outa\_re | ar1’+a\_re\_addend |
| b\_re、c\_re | B、C的实部 |  |  |

算法流程关键问题：

**1、蝶形运算时，a\_re\_mid需不需要延迟一拍?**

答：由于a\_re\_addend相对于b\_re\_c\_re是需要延迟一拍的，因为需要完成一次减法操作。如果a\_re\_mid不延迟一拍的话，ar’\_d1就不是和ar1’完成减法，而是与ar2’完成减法。

**2、假设输入256个点，则需要消耗几个Cycle?**

答：理论上，从检测到ar1、br1、cr1数据，到输出outa\_re需要2个Cycle.则计算256个点，需要消耗258个Cycle.

**3、上述的clk\_rd控制什么?**

答：clk\_rd为RAM的读出控制时钟，在clk\_rd时钟上升沿检测到地址后，输出相应的A、B数据。RAM\_Ctrl模块输出的旋转因子C时钟是与clk保持同步的。为了B、C两数据能够保持对齐，并完成运算，此时需要进行跨时钟域同步。clk\_rd是clk时钟的反相。

**4、bug：**

1）读写时钟时，出现的问题：

不能读L+1级的数据时，正好还在写L级的数据。可以边算L级，写入L级数据。因此每级算完之后，需要等待3个时钟（乘法器一开始的3个流水线延时时钟）。对于9级计算，则需要27个时钟。但是优点是，不需要额外的RAM结构。