neon和vfp测试

首先在双核通信之前cpu0和cpu1都注册一个软中断处理函数，在这个中断处理函数中点亮led。当0核裸机程序做完0核相关初始化后唤醒1核然后进入一个循环。1核启动后也需要相关的初始化，然后发送软中断通知0核，接着也和0核一样进入一个循环，在这个循环里两个核之间一直进行交互通信。 **以下是各种情况较为详细的步骤和对比：**

一、neon双字整数加减法运算测试

1. cpu0收到中断后会触发irq异常，最后会进入之前注册的中断处理函数，然后将cpu0控制的led点亮，接着计算0x4000次的双字neon整数加法和0x4000次的双字neon整数减法，然后将cpu0控制的led熄灭，然后触发相应的软中断给cpu1。
2. cpu1收到中断后会触发irq异常，最后会进入之前注册的中断处理函数，然后将cpu1控制的led点亮，接着计算0x8000次的单字普通整数加法和0x8000次的单字普通整数减法，然后将cpu1控制的led熄灭，然后触发相应的软中断给cpu0。
3. 重新回到第1步

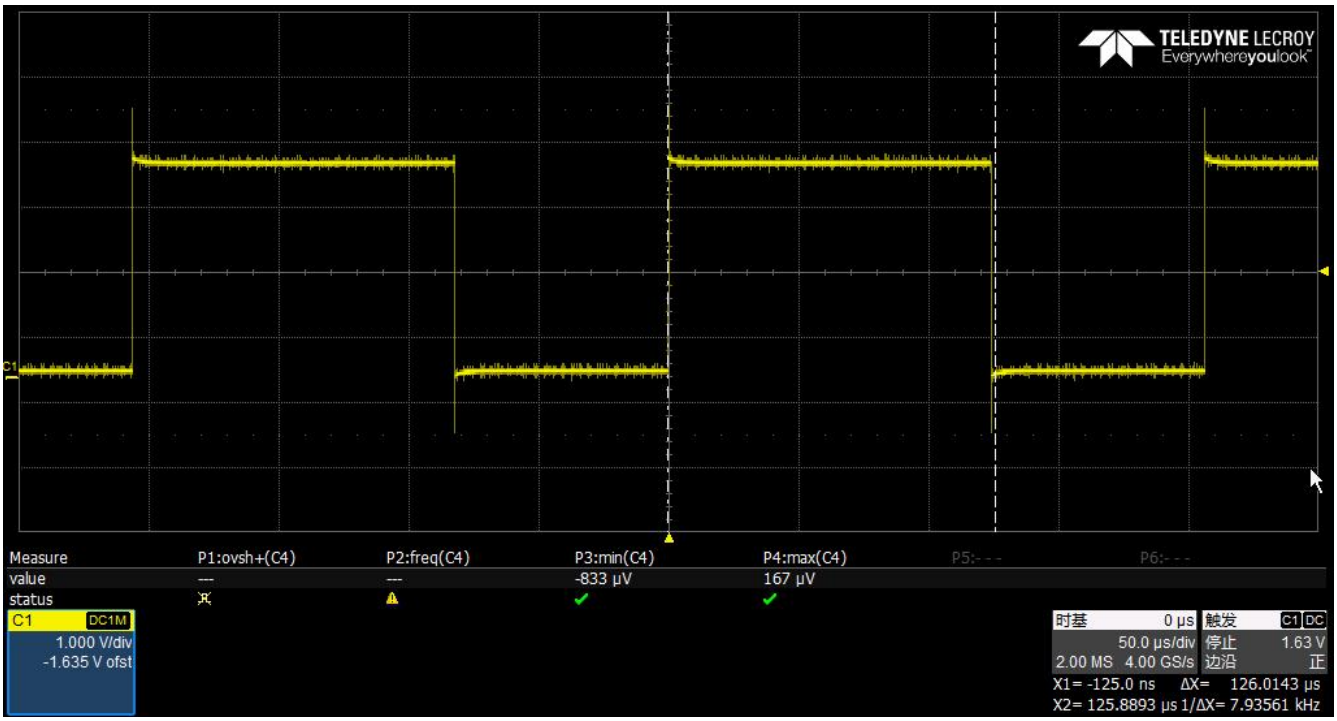

图1 普通整数加减法运算

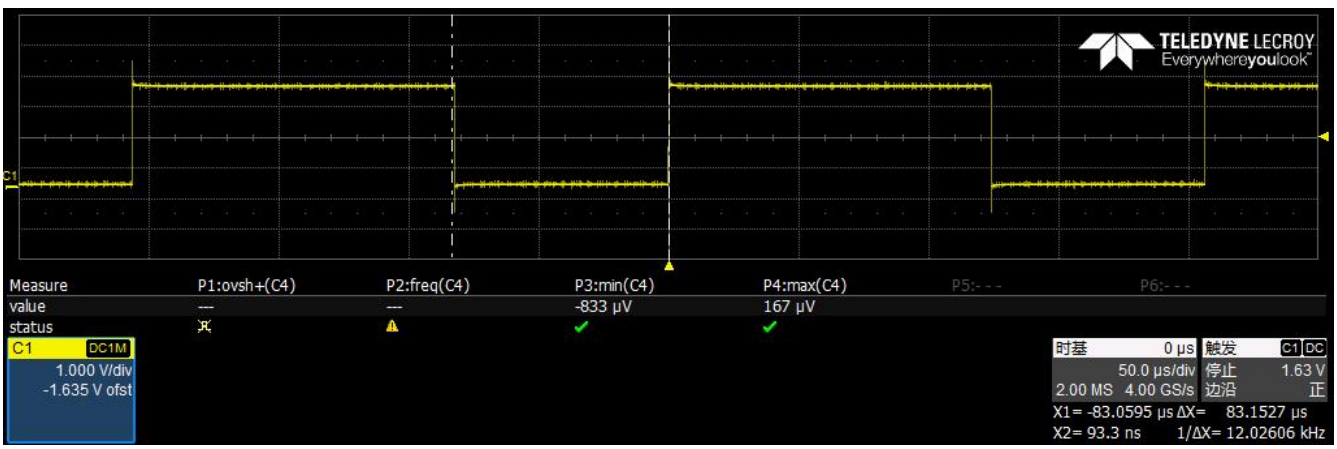

图2 neon双字整数加减法运算

从以上测试结果看出neon双字整数加减法运算速度是普通单字整数加减法运算速度的126/83=1.5倍

二、neon四字整数加减法运算测试

1. cpu0收到中断后会触发irq异常，最后会进入之前注册的中断处理函数，然后将cpu0控制的led点亮，接着计算0x2000次的四字neon整数加法和0x2000次的四字neon整数减法，然后将cpu0控制的led熄灭，然后触发相应的软中断给cpu1。
2. cpu1收到中断后会触发irq异常，最后会进入之前注册的中断处理函数，然后将cpu1控制的led点亮，接着计算0x8000次的单字普通整数加法和0x8000次的单字普通整数减法，然后将cpu1控制的led熄灭，然后触发相应的软中断给cpu0。
3. 重新回到第1步

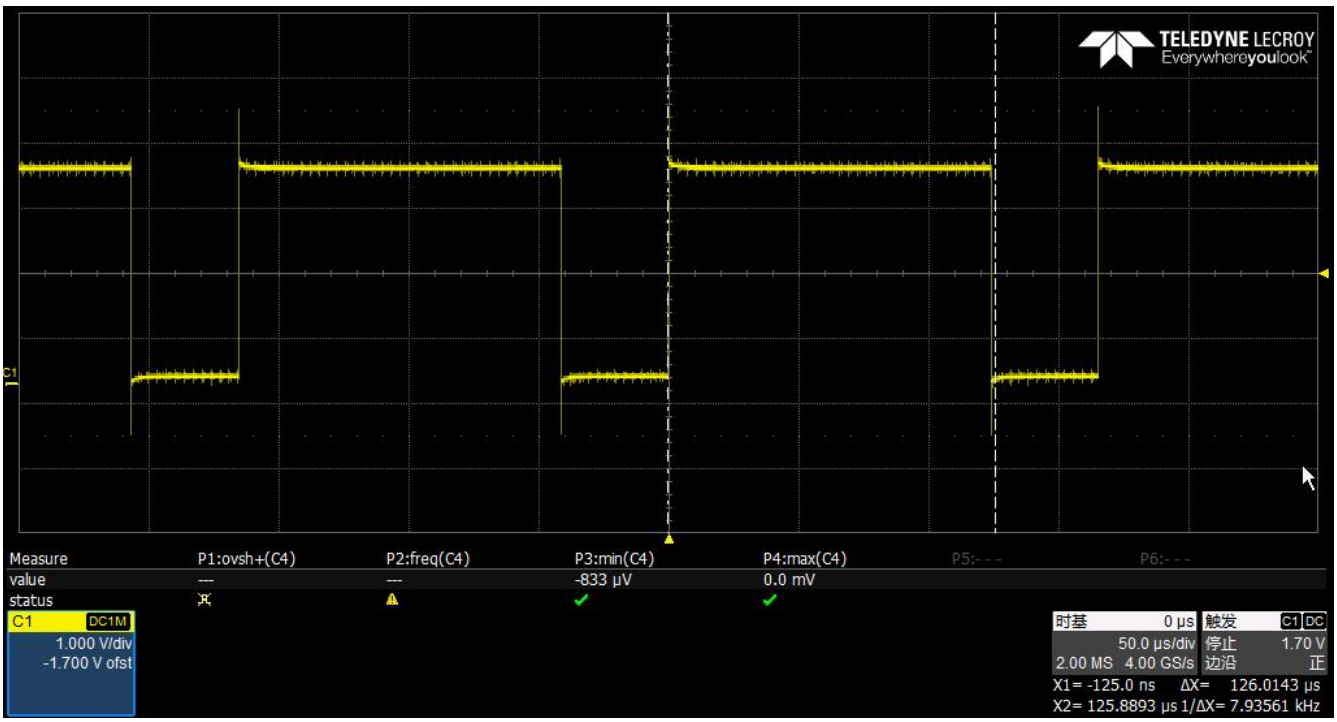

图3 普通整数加减法运算

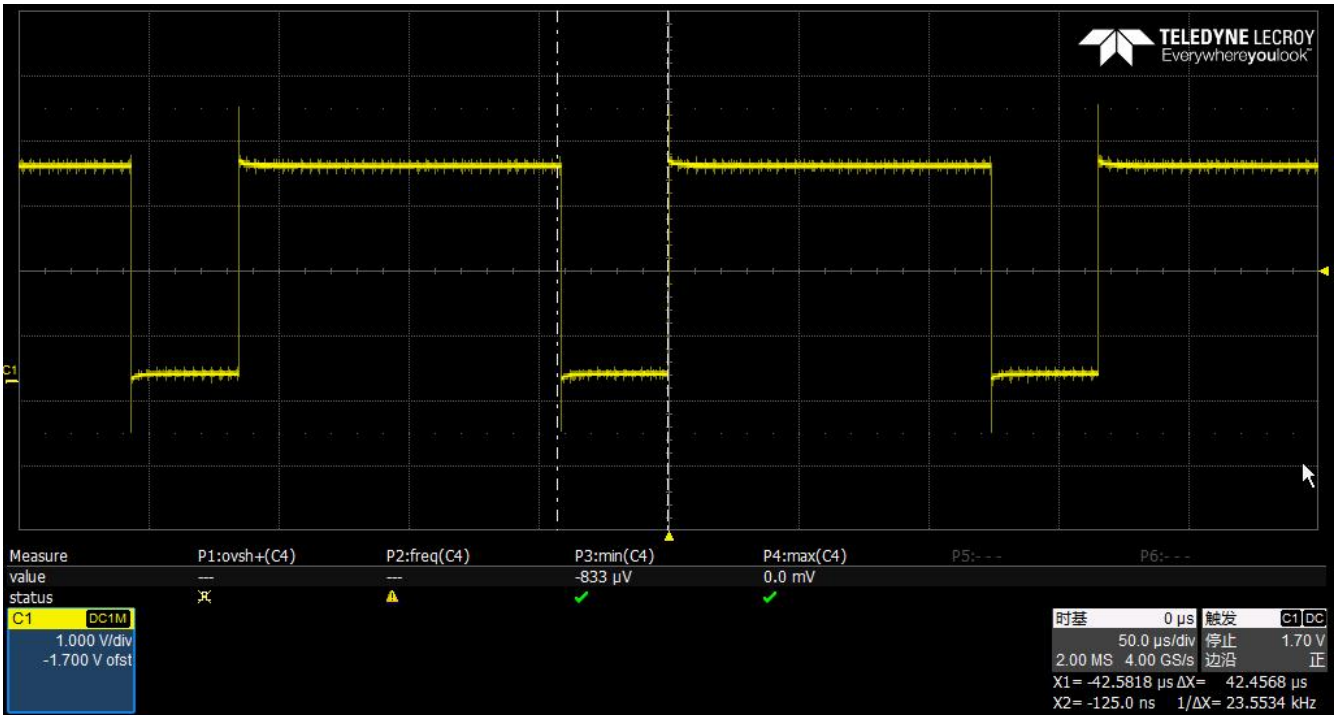

图4 neon四字整数加减法运算

从以上测试结果看出neon双字整数加减法运算速度是普通单字整数加减法运算速度的126/42=3倍

三、vfp浮点乘除法运算测试

1. cpu0收到中断后会触发irq异常，最后会进入之前注册的中断处理函数，然后将cpu0控制的led点亮，接着计算0x8000次的单字vfp浮点乘法和0x8000次的单字vfp浮点除法，然后将cpu0控制的led熄灭，然后触发相应的软中断给cpu1。
2. cpu1收到中断后会触发irq异常，最后会进入之前注册的中断处理函数，然后将cpu1控制的led点亮，接着计算0x8000次的单字普通浮点乘法和0x8000次的单字普通浮点除法，然后将cpu1控制的led熄灭，然后触发相应的软中断给cpu0。
3. 重新回到第1步

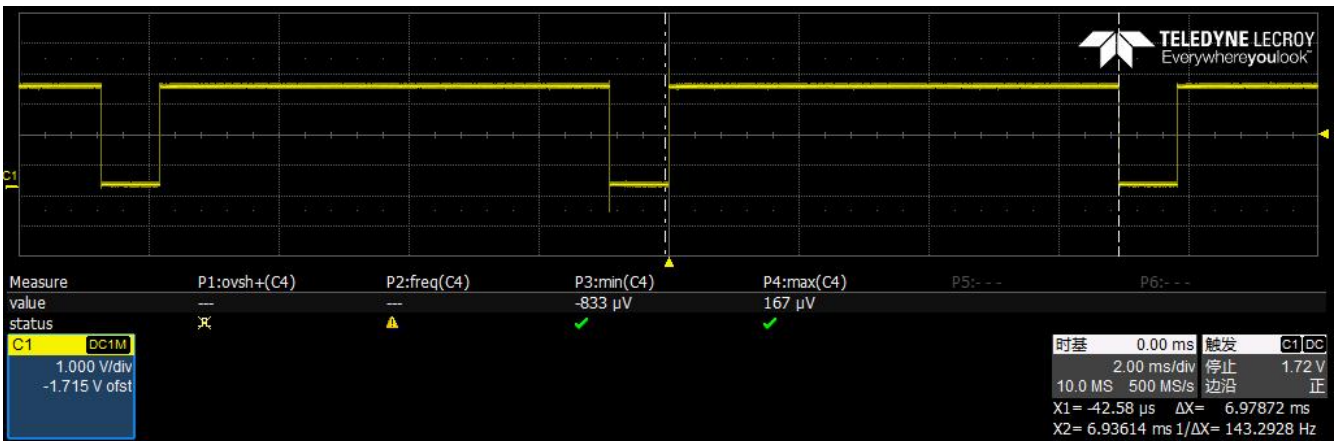

图5 普通整数加减法运算

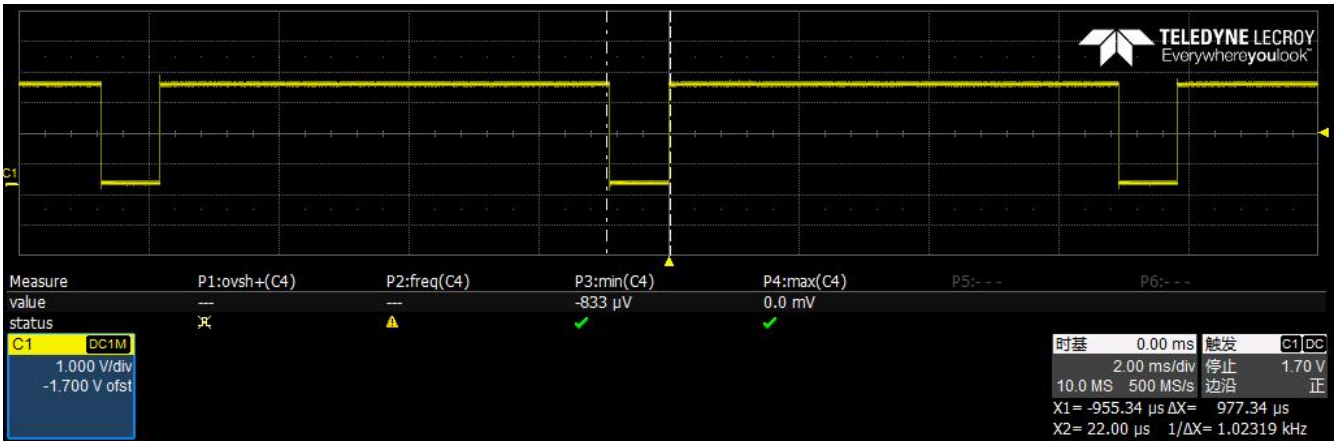

图6 普通整数加减法运算

从以上测试结果看出neon双字整数加减法运算速度是普通单字整数加减法运算速度的

6.97/0.97=7倍