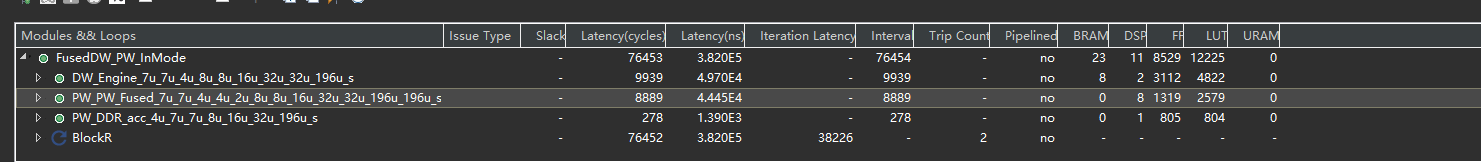
# 三核心

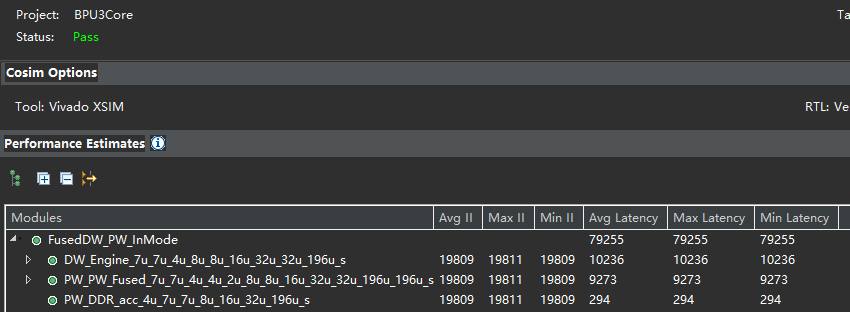
## 1-软件版本

5-23

Csim通过

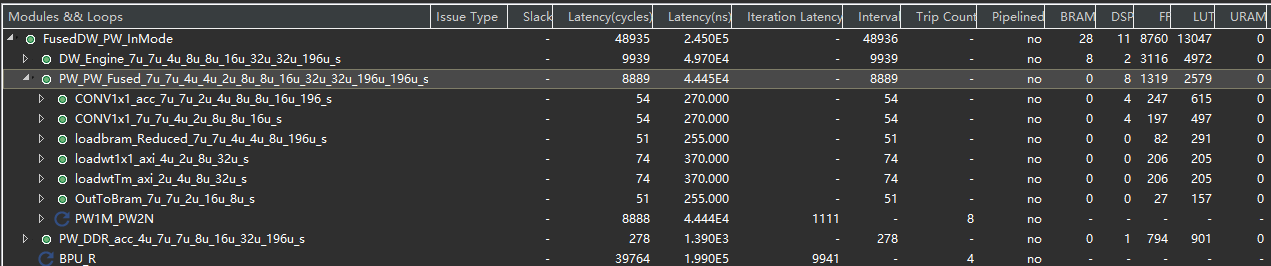
综合：

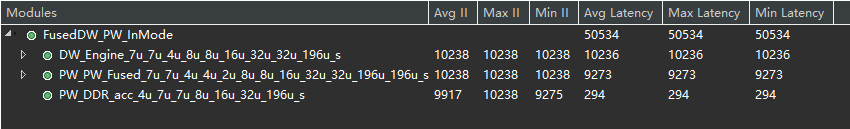


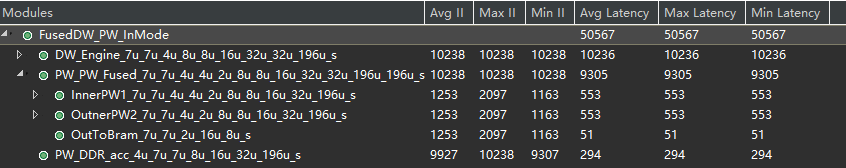


## 2-优化版本

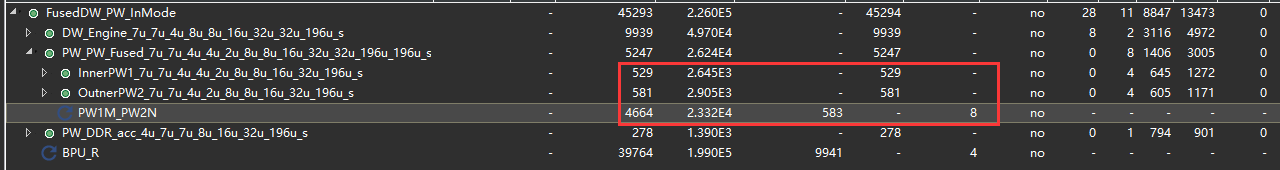
### 2-1外层双缓冲



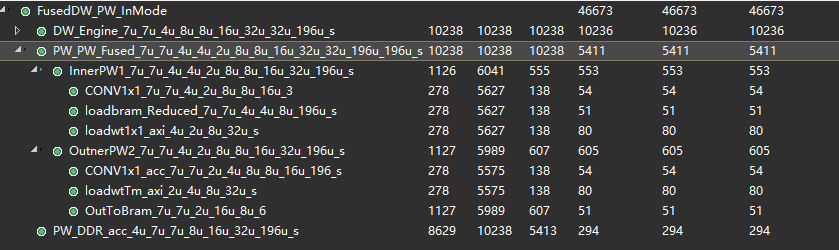




### 2-2 PW\_PW外层双缓冲

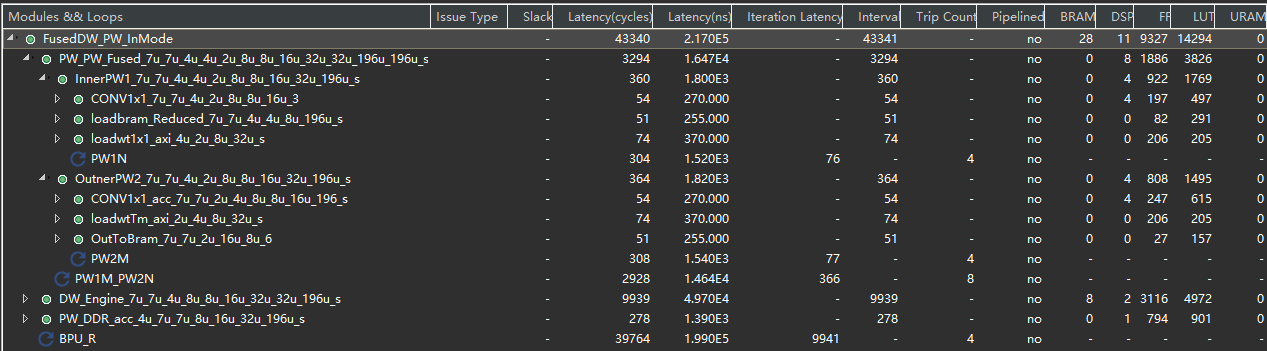


CtoRTL

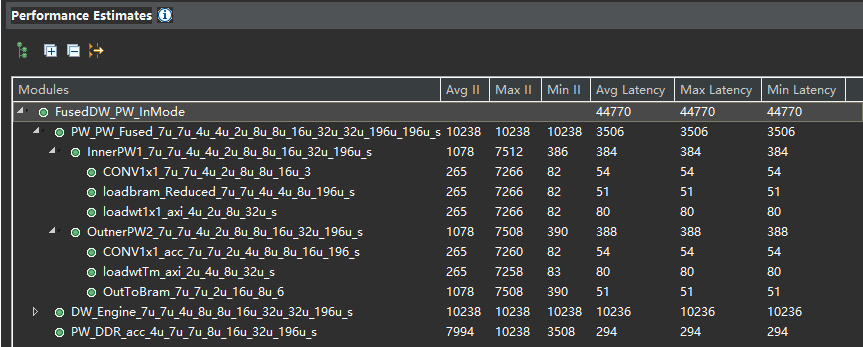


### 2-3 全部双缓冲

综合：



CtoRTL



## 3-仿真记录

整除情况：DwTn=8,PwTn=4。这里可以看出**preload weight buffer**还是很重要的！

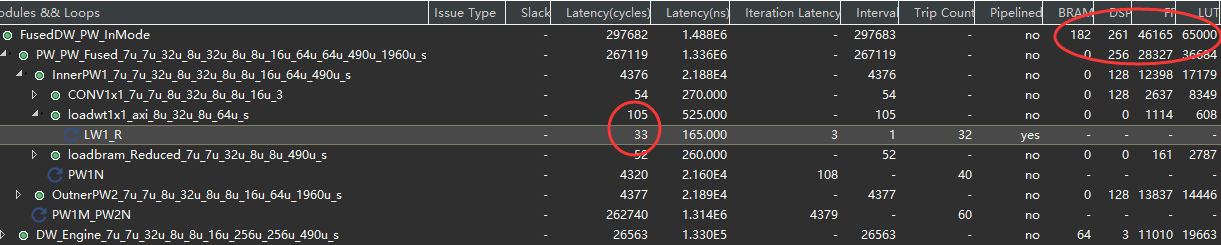
这里的CtoRTL是还没有preload的情况

|  |  |  |
| --- | --- | --- |
|  | CoRTL cycles  带宽是根据端口确定的，后者带宽根据硬件参数确定 | Modeling |
| 14\*14\*16-> 14\*14\*16(e=1)  DwTn,PwTn,Tm=8,4,2 | 25443 | 28720.0带宽[1.0,1.0, 1.0, 1.0, 4.0] |
| 14\*14\*16-> 14\*14\*16(e=6)  DwTn,PwTn,Tm=8,4,2 | 81367 | 83647.0 |
| 14\*14\*24-> 14\*14\*24(e=1)  DwTn,PwTn,Tm=8,4,2 | 38962 | 42622 |
| 14\*14\*32-> 14\*14\*32(e=1)  DwTn,PwTn,Tm=8,4,2 | 59279 | 54724.0 |
| 14\*14\*32-> 14\*14\*32(e=1)  DwTn,PwTn,Tm=8,8,2 | Csim正确cosim失败 |  |
| 14\*14\*16-> 14\*14\*16(e=1)  DwTn,PwTn,Tm=8,8,2 | 24167 |  |
| 14\*14\*32-> 14\*14\*32(e=1)  DwTn,PwTn,Tm=16,4,4 | 33044 |  |
| 28\*28\*32->28\*28\*32（e=6）  DwTn,PwTn,Tm=8,4,2 | 1118788, 未preload造成-》961609 | 934733.0 |
| 28\*28\*32->28\*28\*32（e=6）  DwTn,PwTn,Tm=16,8,4 | 324632 | 313651 |
| 7\*7\*320 ->7\*7\*320(e=6)  DwTn,PwTn,Tm= 32,32,8 | 257609 未preload造成 | 221467 |
| 7\*7\*320 ->7\*7\*320(e=6)  DwTn,PwTn,Tm= 32,8,32 | 312510 未preload，preload优化后170808 | 163587 |
| 7\*7\*32 ->7\*7\*32(e=2)  DwTn,PwTn,Tm= 8,8,8 | 14377 CtoRTL | 15021 |
| 14\*14\*128 ->14\*14\*128(e=6)  DwTn,PwTn,Tm= 32,8,32 | 综合：129472 cosim 116399 | 114130 |

## 4 优化

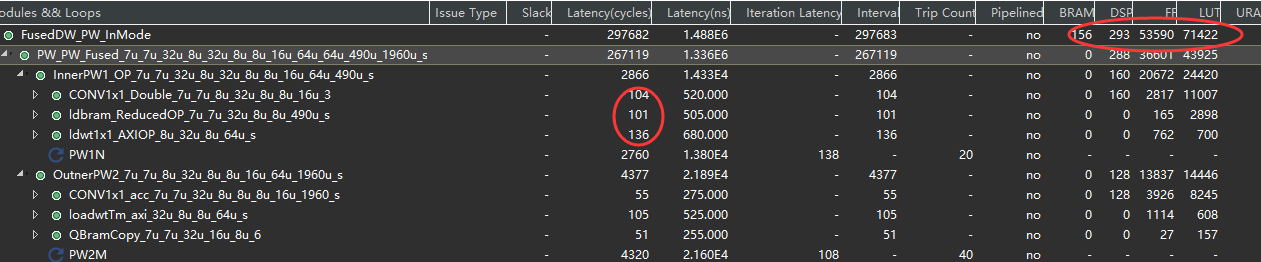
### 4-1 权重预取

未优化前

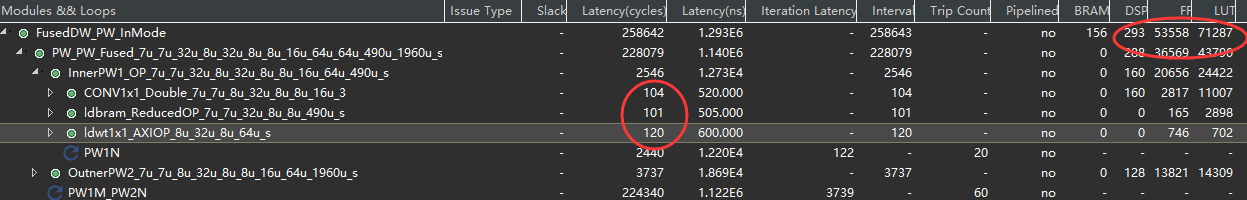


先对inner\_PW1的权重预加载一倍，目前直接对wt[Tm][2\*Tn]与Inbuf[2\*Tn][Tr][Tc]进行并行展开。应该比较消耗资源。

可以看出这个setup时延位75左右。此时top中latency设置为64，可以改为48.都是比较接近实际的。

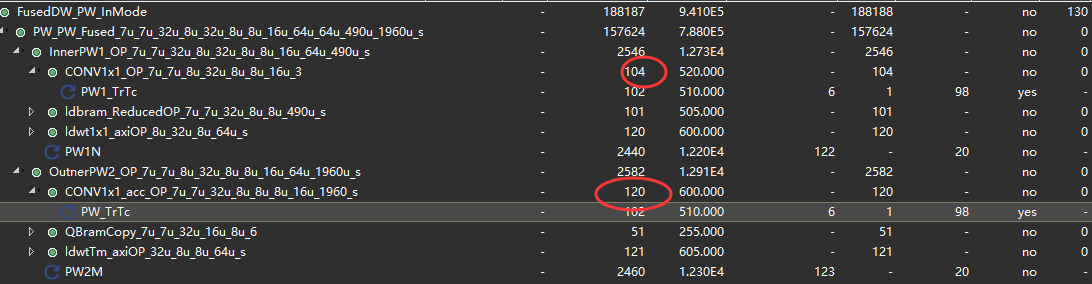


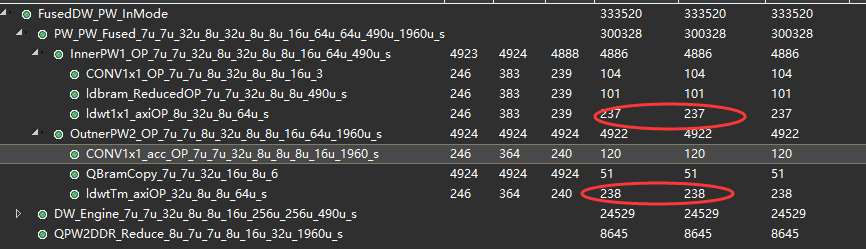
Top函数接口改为48之后减少了16cycles。



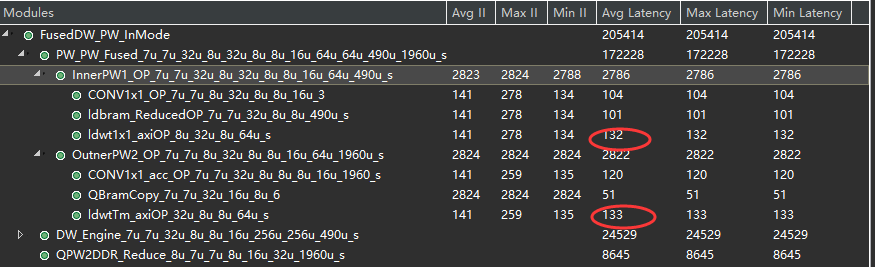
计划改成wt[2][Tm][Tn]，inbuf[2][Tn][Tr][Tc]

Outer 优化后

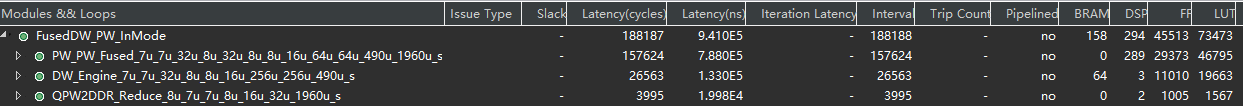




Setup时间并没有共用，outstanding改为64

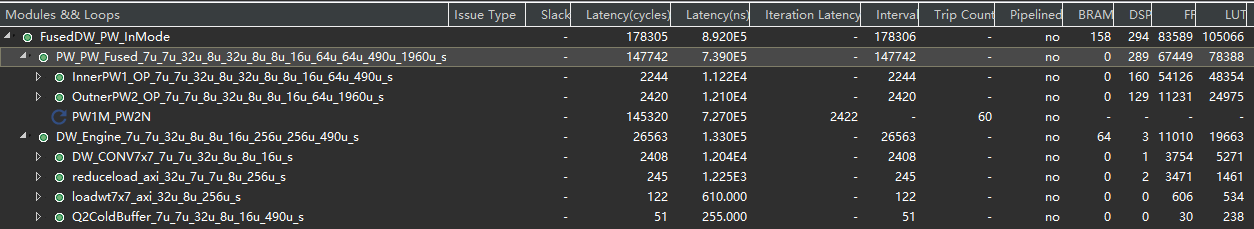


Outstanding = 96后：cotoRTL不变。综合后的资源消耗如下

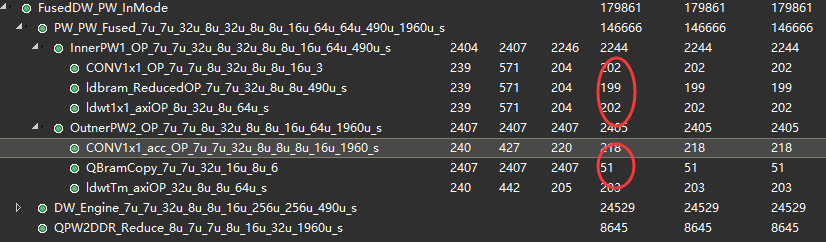


Preload=3 or 4可以进一步减少访存延迟？

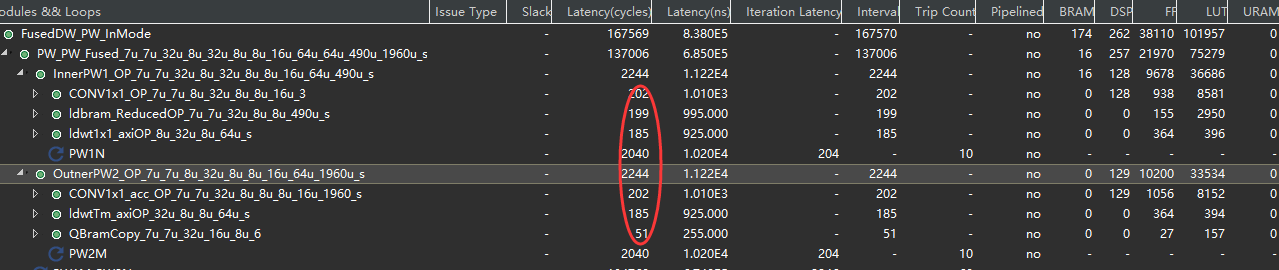
PRELOAD=4



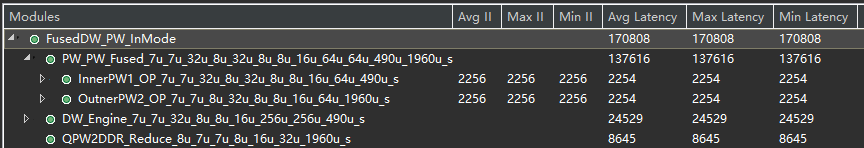
CoRTL



计算时延218，partition维度出错导致，修改后



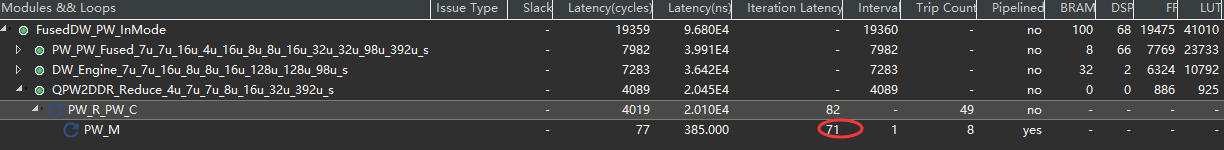
如果能用一个变量代替的就用一个变量代替，即便变量等零时也别使用零替换

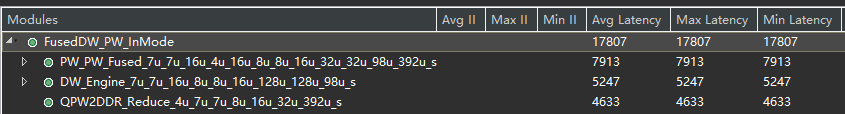


### 4-2添加分支

优化前：

综合报告

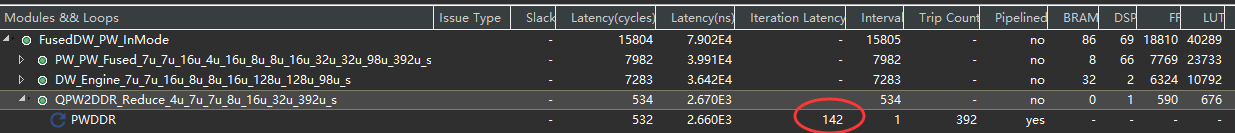




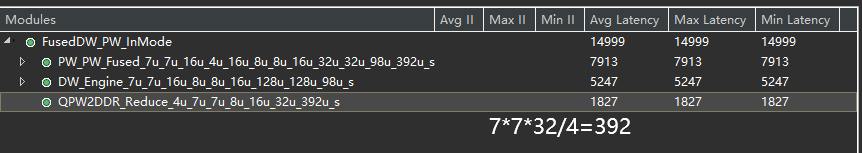
QPW2DDR\_Reduce**循环展开优化后**

7\*7\*32->7\*7\*32 e6, DwTn,PwTn,Tm = 16,4,16

综合报告：



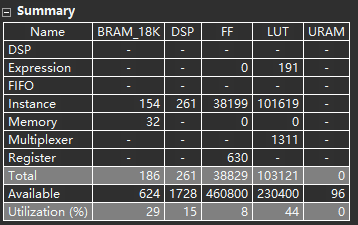
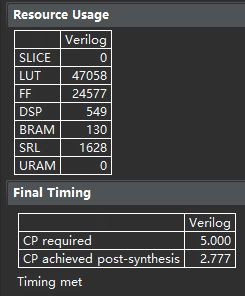
RTL报告



## 5 上板实测

|  |  |  |
| --- | --- | --- |
| 14\*14\*128 ->14\*14\*128(e=6)  DwTn,PwTn,Tm= 32,8,32 | 综合：129472 cosim 116399 | 114130 |

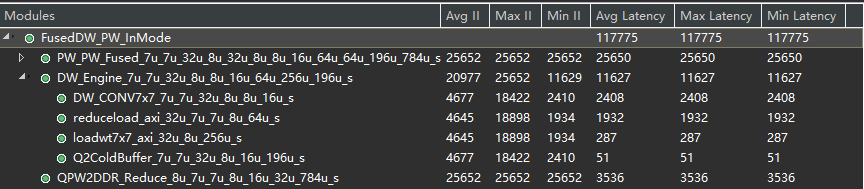
资源消耗情况

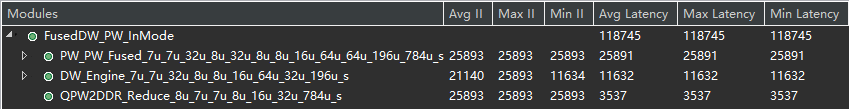
板载性能为cosim 的 83%

### 5-1接口改写

上板子实测版本，按照并行度给的带宽，这里修改为DSE的实际带宽，上板性能应该会下降一些。

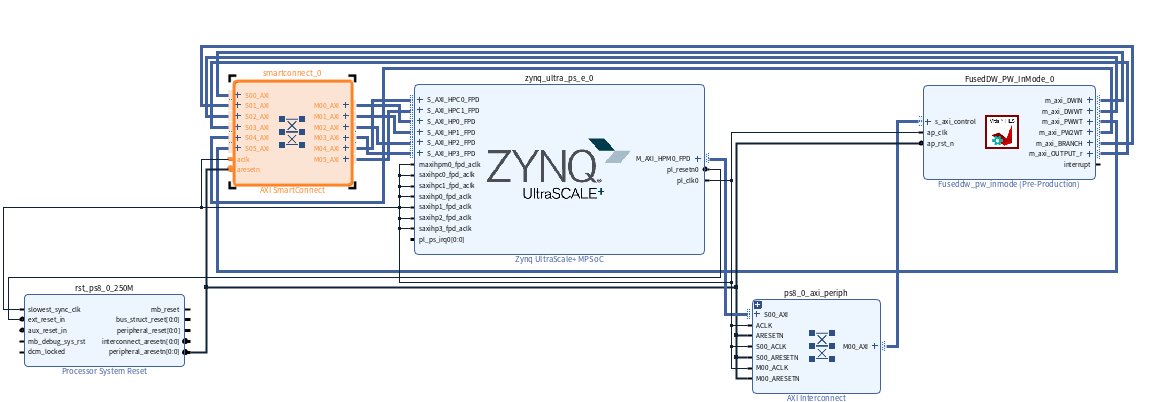


接口8-4-8-8-4，**时钟频率约束到250Mhz**后



理论cycles : 118745左右 T=4ns. 预计474980 ns = 474us

共用一个axi-smart 桥



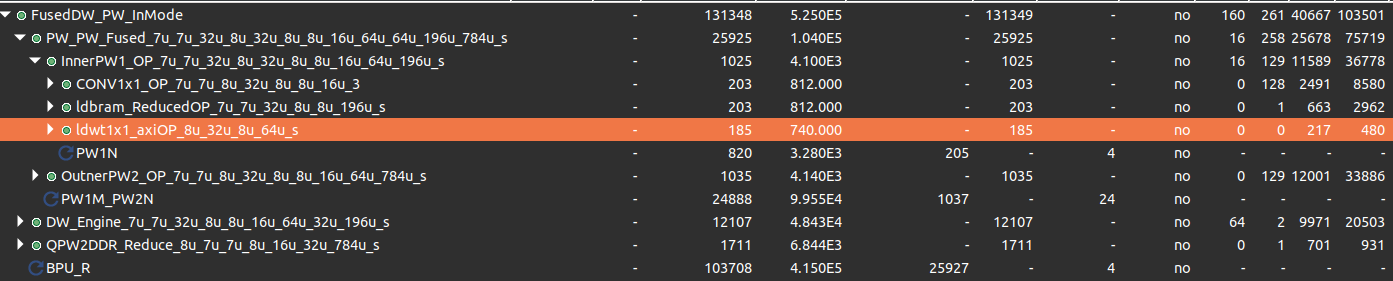
7030us

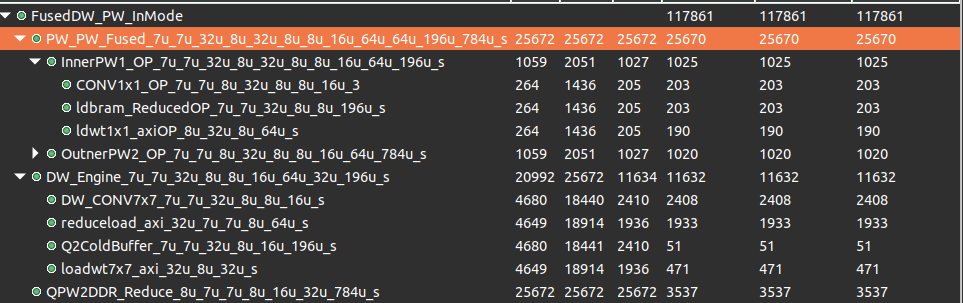
多个桥：

7016 us

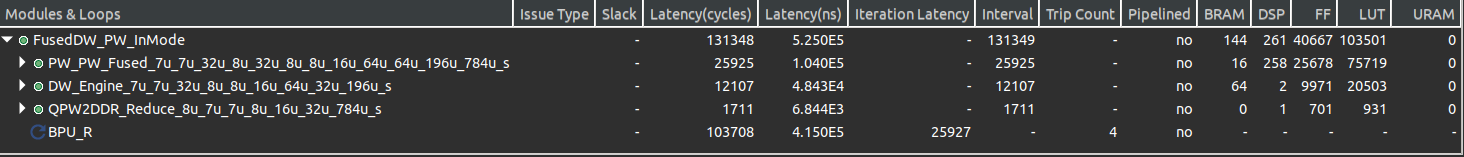
相较于之前的6\_27版本，仅仅修改了dw\_in 和dw\_wt的带宽。应该是brust未被识别

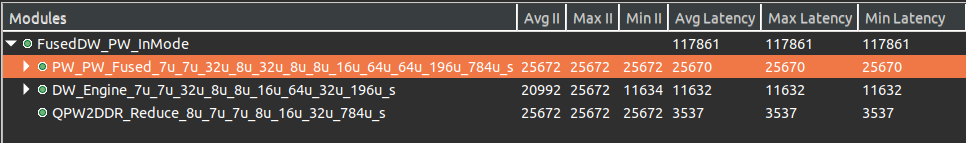
将pw1,pw2中wt加载部分都是用cnt代替。理论上更容易被识别

权重延迟好了一些

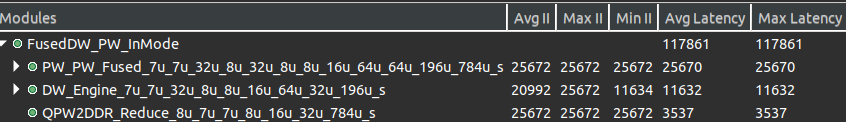
这里将outstanding缩小为64进行尝试

BRAM使用上，综合报告中BRAM使用减少了

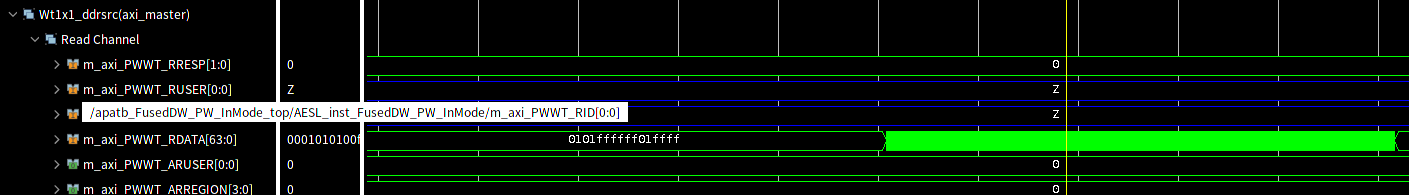
cosim无变化



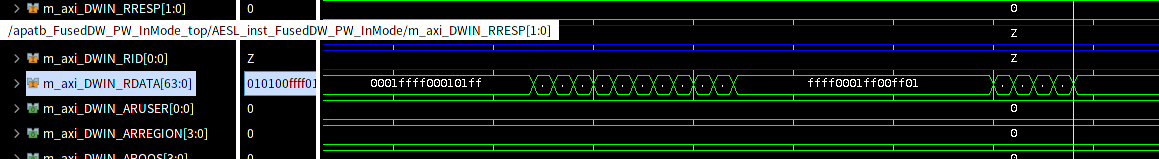
outstanding 缩小为32，进行尝试，cosim仍无变化，节约了资源



wt7x7,wt1x1权重brust length可以设置为64，配合outstanding=32已经很优秀

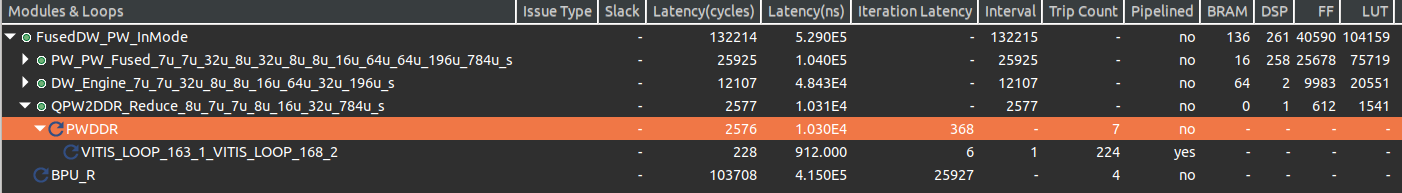


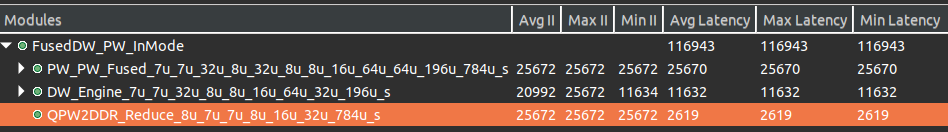
问题出现在DWaxi\_in这里,仅仅连续读16数据由于outstanding。综合:751cycles, cosim:1933 cycles

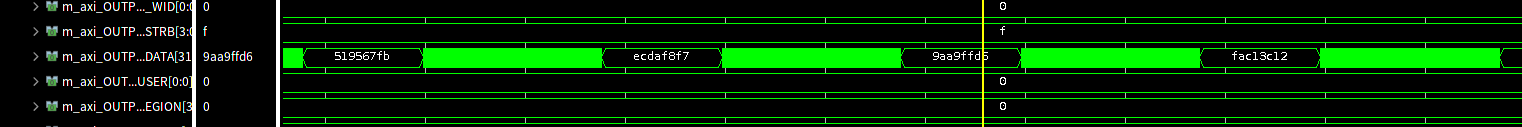


pw2ddr中，似乎也没被识别为brust,目前的length=32,和设定的outstanding一致.综合:1711， cosim：3537

修改后，dw\_in没有变化。pw2\_out虽然综合报告上性能差了一些，但是cosim有提升(初始版本的综合性能比较理想)





仿真波形上看，brust已经启动，长度为224cycles(32\*7)，中间的间隔大概160cycles, 共有6个setup latency.可见loop不合并会导致多个setup延迟即便有outstanding，也又多次