**Хлыстун Дмитрий, группа P41193**

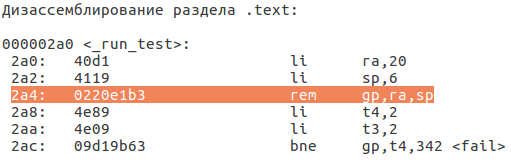
**Lab SCR1 pipeline**

|  |  |  |  |
| --- | --- | --- | --- |
| Вариант | Команда | Arch #1 | Arch #2 |
| 8 | REM | RVIMC | RVIC |

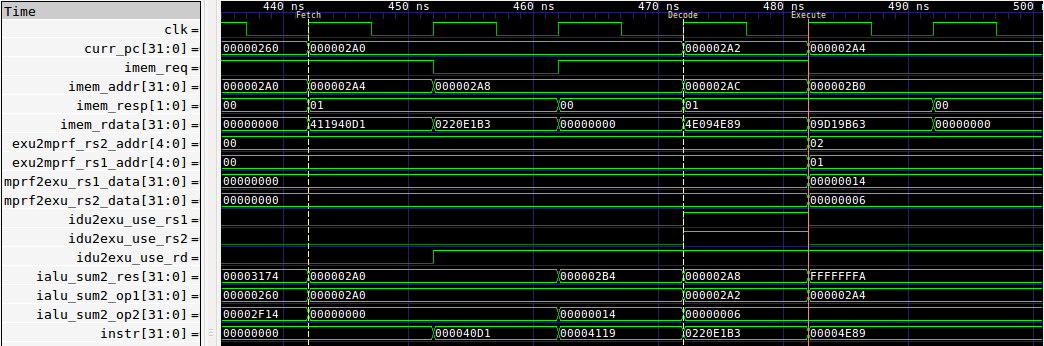
**1 Часть**

Для разборки был выбран файл с тестом «rem.S».

Отрывок из dump-файла, где используется команда rem:

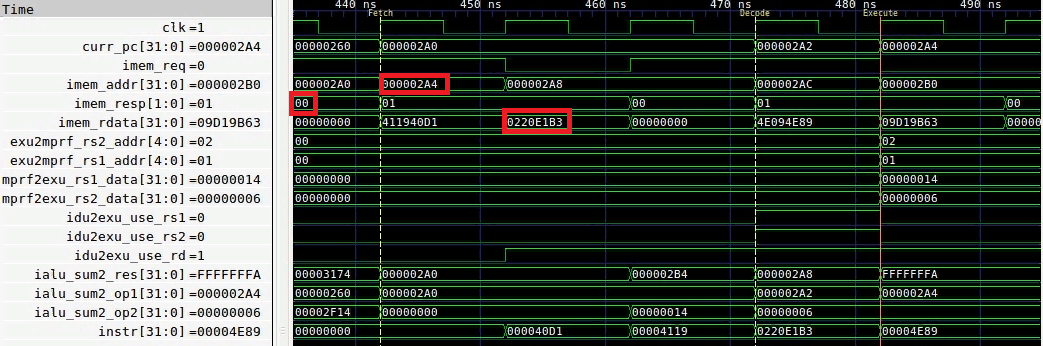


Результирующая wave-form:

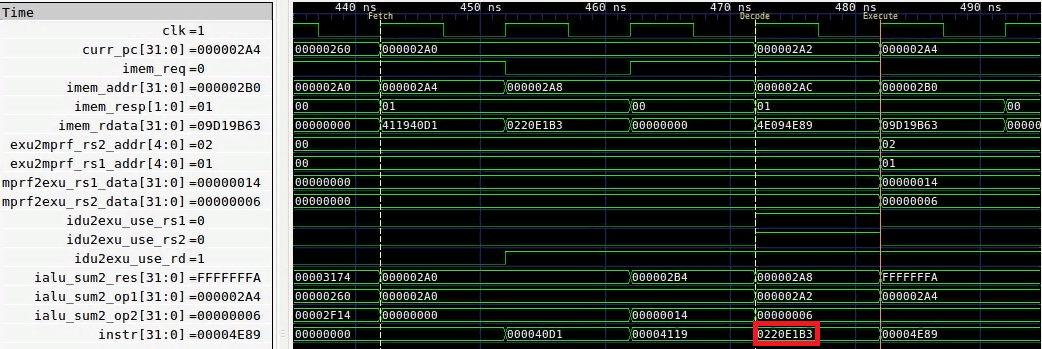


Описание сигналов:

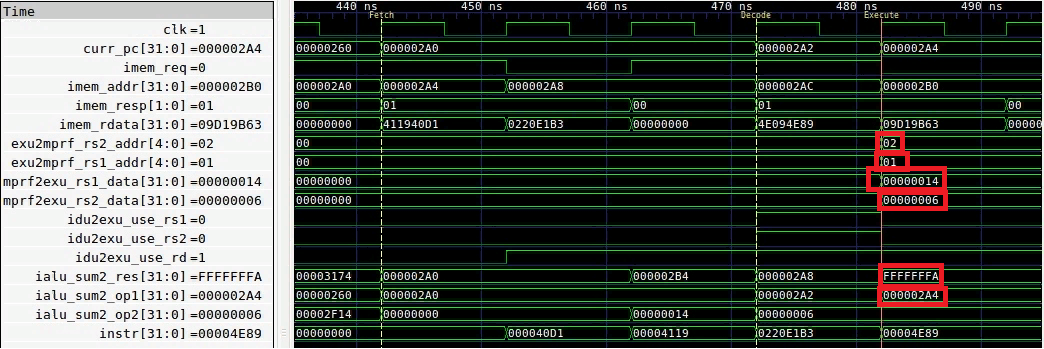
* clk - тактовый импульс
* curr\_pc - текущее значение счетчика команд, соответствует стадии Execution (000002A4)
* набор сигналов для Instruction Fetch:
  + imem\_req - запрос от процессора в память инструкций.
  + imem\_addr - адрес запроса памяти инструкций (000002A4)
  + imem\_resp - ответ памяти инструкций (00)
  + imem\_rdata - данные чтения памяти инструкций (0220Е1В3)



* набор сигналов для Decode:
  + instr – закодированная инструкция (0220Е1В3)
  + idu2exu\_use\_rs1 – сигнал о том, что в команде есть rs1 (0)
  + idu2exu\_use\_rs2 – сигнал о том, что в команде есть rs2 (0)

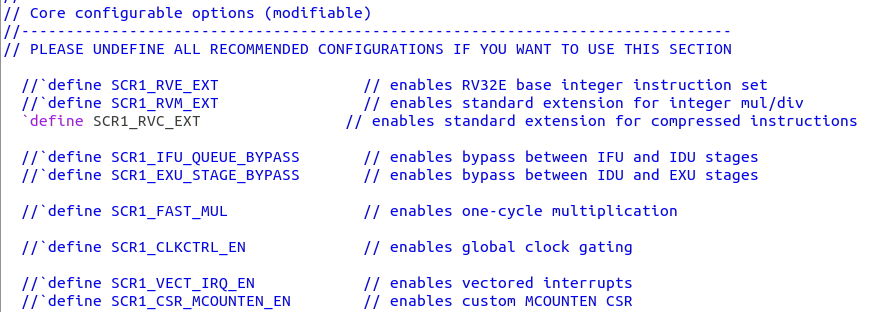


* Набор сигналов для Execute:
  + exu2mprf\_rs2\_addr – номер регистра в котором лежит адрес для записи в память. (02), x2 = sp
  + exu2mprf\_rs2\_data – значение для записи в регистровый файл (00000006).
  + exu2mprf\_rs1\_addr – номер регистра в котором лежит адрес. (01) x1;
  + exu2mprf\_rs1\_data – значение регистра x1 (00000014);
  + ialu\_sum2\_op1 – первый операнд АЛУ (000002A4) Это значение регистра x2(sp);
  + ialu\_sum2\_op2 – второй операнд АЛУ (FFFFFFFA) Это смещение;
  + ialu\_sum2\_res – результат АЛУ.

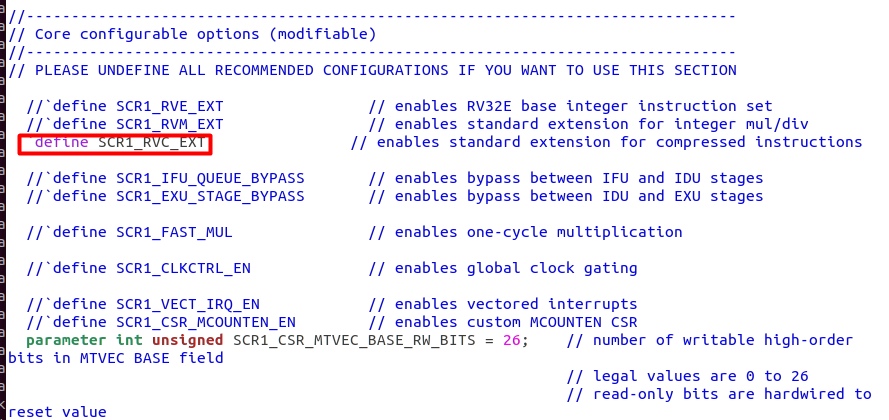


**2 Часть**

Были запущены два теста-бенчмарка Coremark и Dhrystone для двух архитектур по заданию: RVIMC и RVIC. Для этого был изменён файл «scr1\_arch\_description.svh»:

RVIMC 

RVIC



Вызов: make run\_verilator\_wf BUS=AHB ARCH=IC IPIC=0

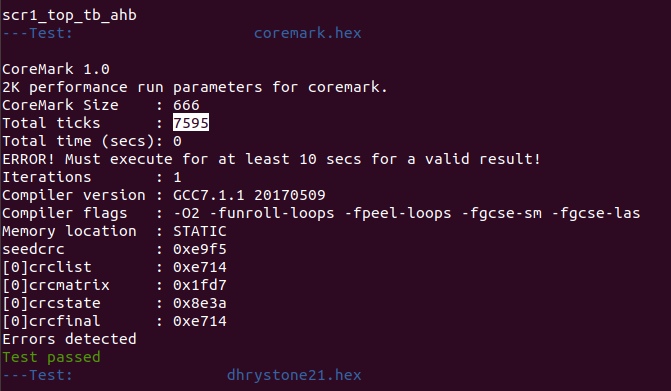
Были получены следующее результаты:

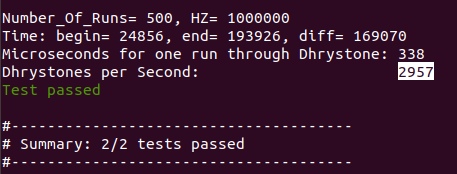
|  |  |  |
| --- | --- | --- |
| Тест | RVIМC | RVIC |
| Coremark (Total ticks) | 7595 Total ticks | 11397 Total ticks |
| Coremark (size memory) | 75,5Кбайт | 78,6 Кбайт |
| Dhrystone (per Seconds) | 2957 per Seconds | 2922 per Seconds |
| Dhrystone (size memory) | 41,6 Кбайт | 45,3 Кбайт |

При выборе разных архитектур, для Dhrystone, количество итераций в секунду изменилось на 35, а размер занимаемой памяти – на 3,7 Кбайт. Для Coremark количество итераций в секунду изменяется на 3802, при это размер занимаемой памяти увеличился.

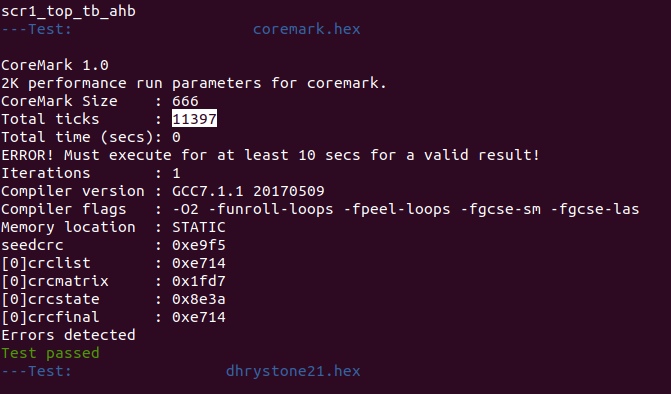
Из полученных результатов можно сделать вывод, что Coremark занимает почти в два раза больше памяти, чем тест Dhrystone, но при этом количество итераций в секунду выше, по сравнению с Dhrystone.

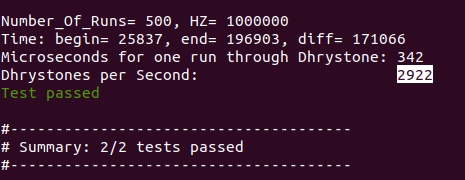
Скриншоты тест-бенчмарков Coremark и Dhrystone для ARCH = RVIМC





Скриншоты тест-бенчмарков Coremark и Dhrystone для ARCH = RVIC

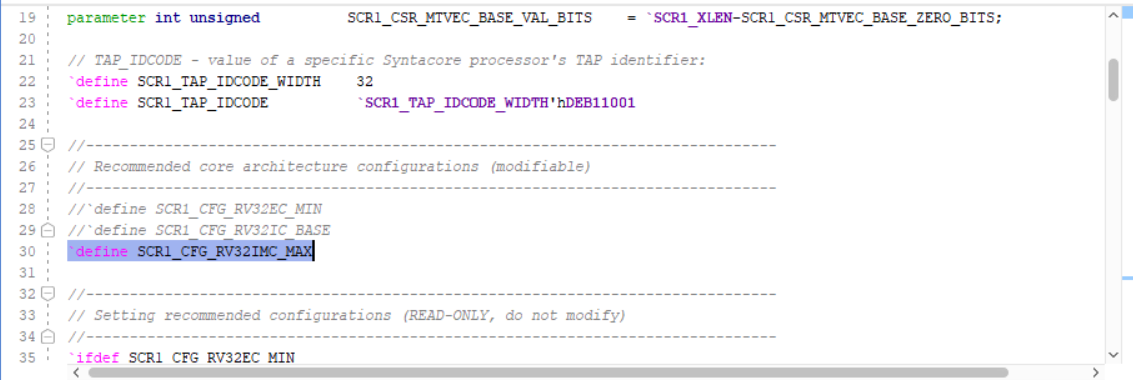




**3 часть**

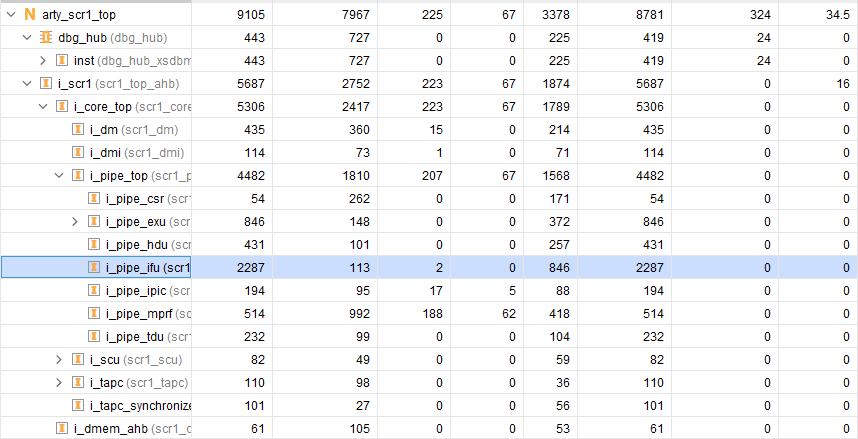
Запуск c конфигурацией IMC\_MAX.

Конфигурация была выбрана в файле scr1\_arch\_description.svh

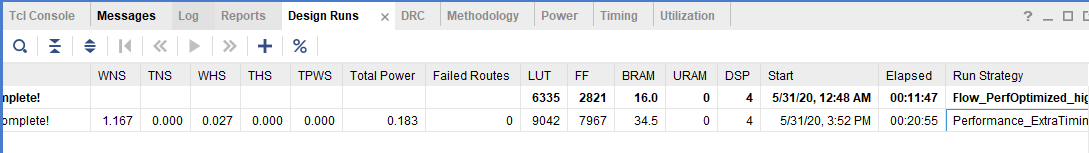


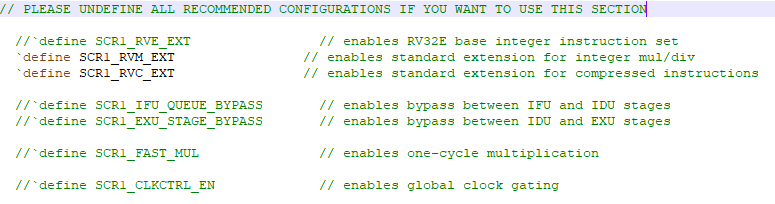
Fmax = 33.(3) MHz

1. Report Utilization. Модуль ядра, занимающий наибольшую площадь на кристалле: i\_pipe\_ifu = 2287.



1. Значение WNS = 1.167 c Perfomance\_ExtraTimingOpt.

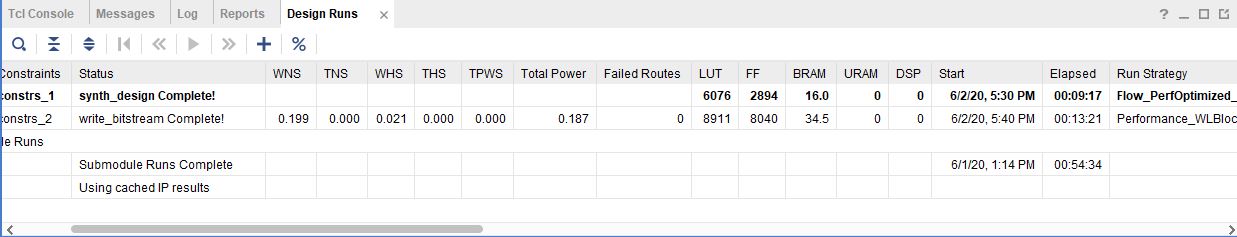


В файле scr1\_arch\_description.svh по варианту оставил **RVIМC**. 

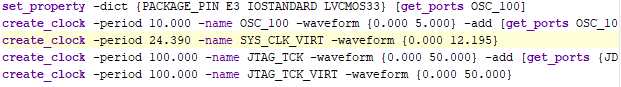
Определение Fmax:

Изначально поставил частоту F = 33.333 MHz.

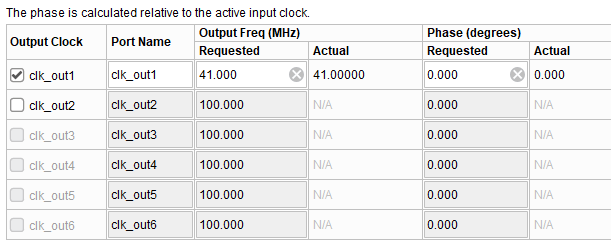
WNS получилось 0.199.



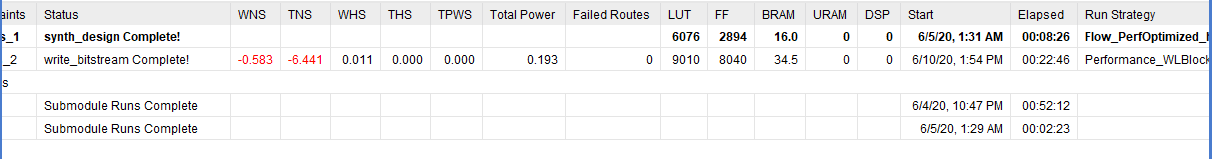
Я решил попробовать оптимизировать и уменьшить значение WNS. Поставив SYS\_CLK\_VIRT 24.390 ns.



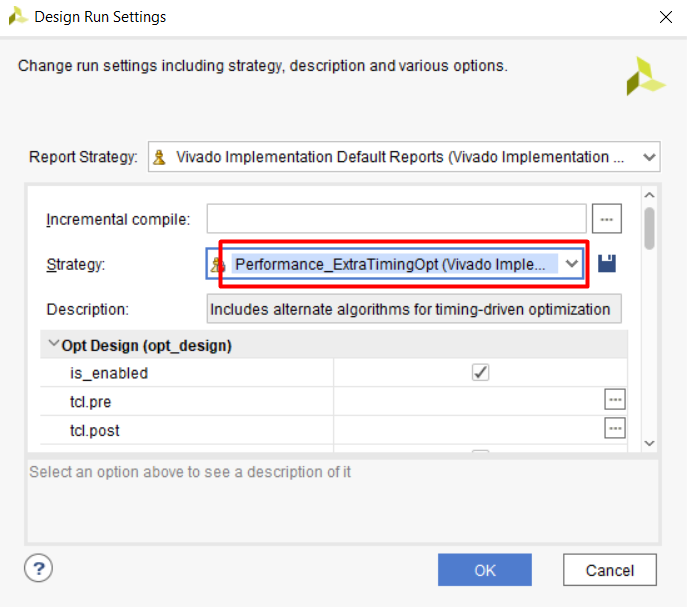
Следовательно, F равно (1/24.390)\*1000 = 41 MHz.



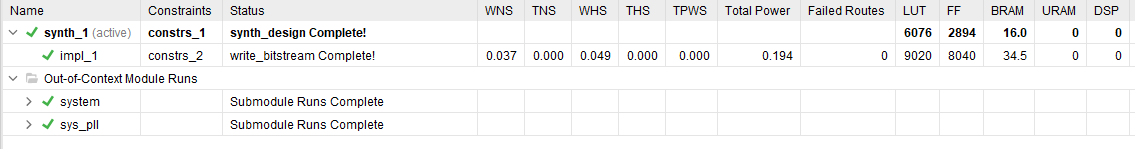
WNS получилось -0.583.



Использовал стратегию Perfomance\_ExtraTimingOpt для того, чтобы WNS было положительным.

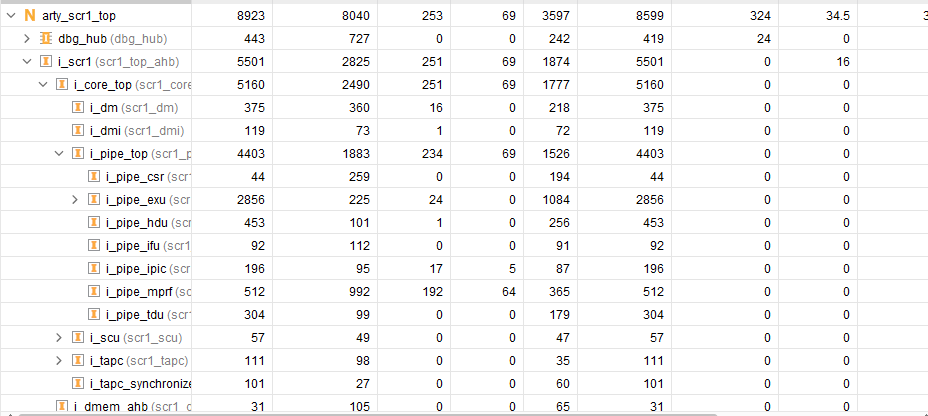


С помощью Perfomance\_ExtraTimingOpt удалось добиться WNS = 0.037.

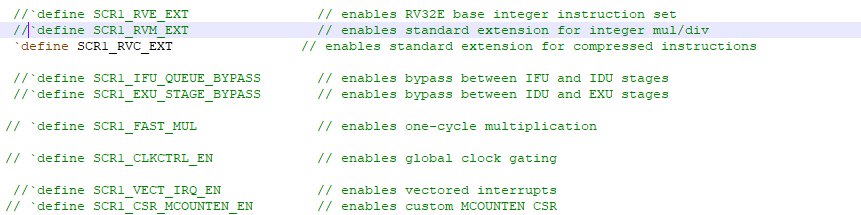


Следовательно, для RVМIC Fmax = 41 MHz.

После успешного прохождения Generate Bitstream, был запущен Report Utilization и определил модуль ядра, занимающий наибольшую площадь на кристалле i\_pipe\_exu = 2856.

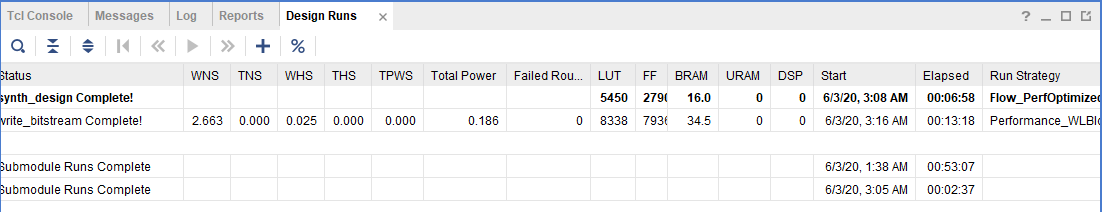


В файле scr1\_arch\_description.svh по варианту поставил **RVIC.**

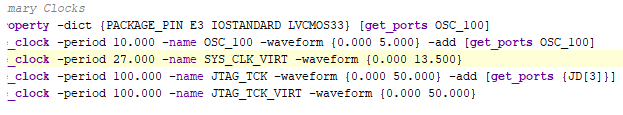


Изначально поставил частоту F = 33.333 MHz/ 30 ns.

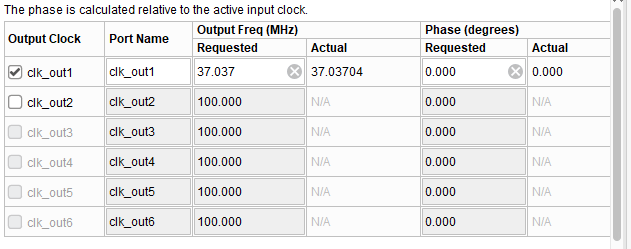
WNS получилось 2.663.



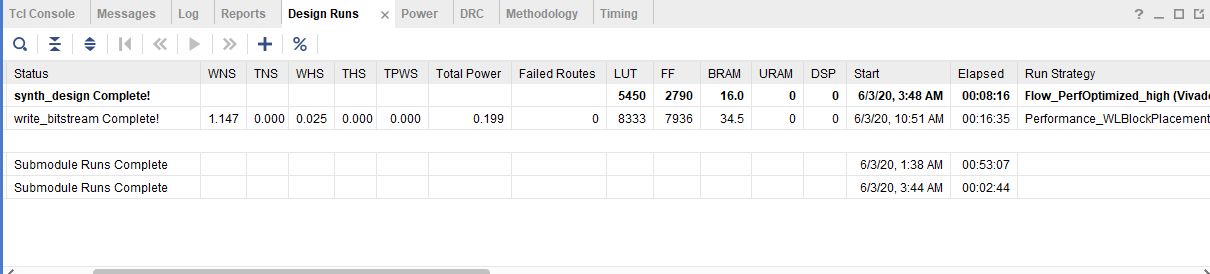
Исходя из этого значения, я понял, что у меня есть запас 3 ns и поэтому уменьшил период на 3 ns (период SYS\_CLK\_VIRT = 27 ns).



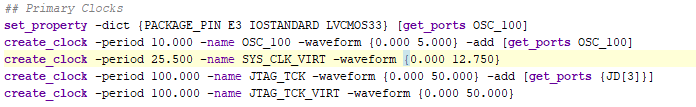
Следовательно, F равно (1/27)\*1000 = 37.037 MHz.



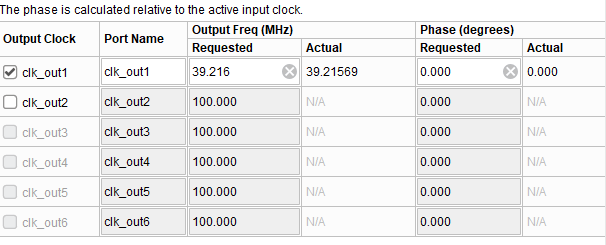
Получил WNS = 1.147 ns



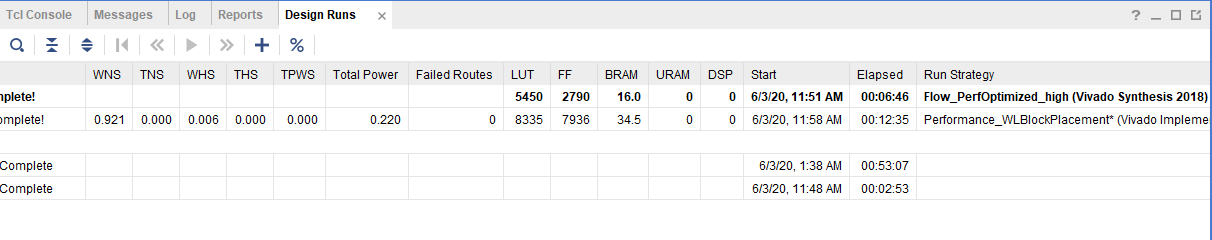
WNS = 1.147 ns можно сделать вывод, что запаса по времени удержания и установки с избытком. Можно попробовать уменьшить период SYS\_CLK\_VIRT на 1.5 ns. Следовательно, период будет равен 25.5 ns.



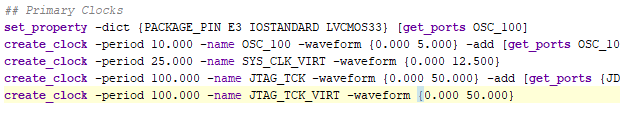
Следовательно, F равно (1/25.5)\*1000 = 39,216 MHz.



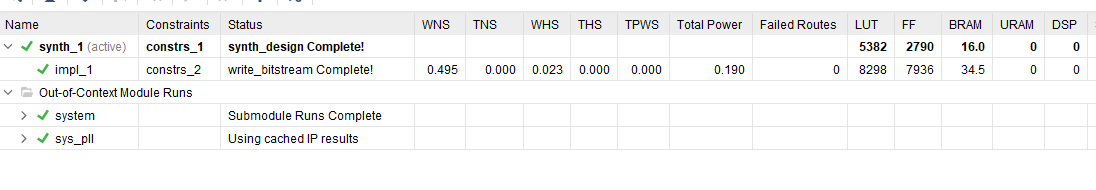
Получилось WNS = 0.921



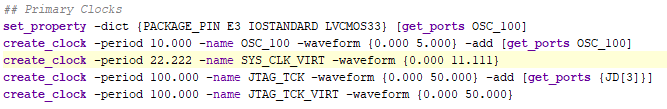
Я решил попробовать оптимизировать и уменьшить значение WNS. Поставив SYS\_CLK\_VIRT 25 ns, а F равно (1/20) \*1000 = 40,000 MHz.



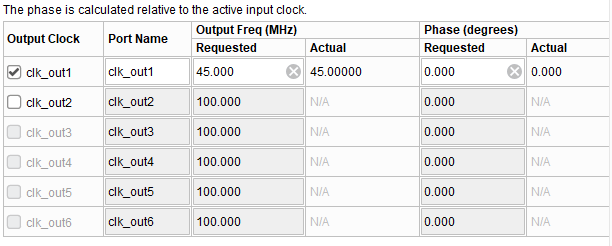
И получил WNS = 0.495



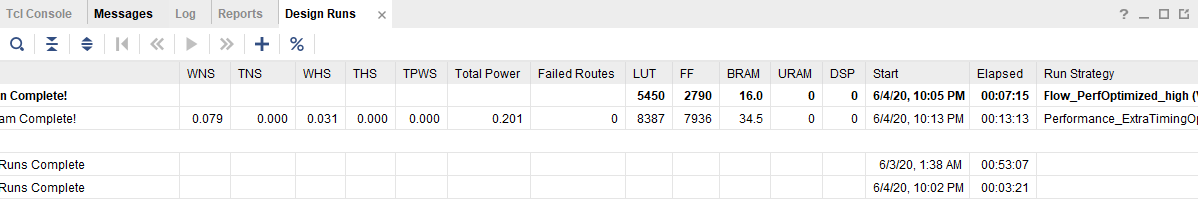
Я решил попробовать оптимизировать и уменьшить значение WNS. Поставив SYS\_CLK\_VIRT 22.222 ns.



F равно (1/22.222) \*1000 = 45,000 MHz.

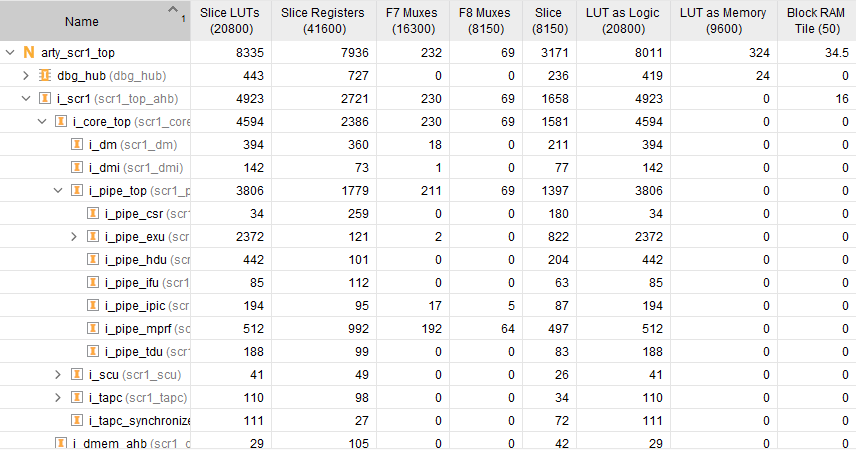


Для того чтобы WNS приблизился к 0 решил использовать стратегию Perfomance\_ExtraTimingOpt. И получил WNS = 0.079



Следовательно, для RVIC Fmax = 45 MHz.

Определил модуль ядра, занимающий наибольшую площадь на кристалле i\_pipe\_exu = 2372.



Сравнение занимаемой площади на кристалле в зависимости от архитектур по варианту:

|  |  |  |  |
| --- | --- | --- | --- |
| Arch | LUT (Look Up Table) | FF (Flip-Flop) | Fmax |
| RVIМC | 9020 | 8040 | 41 MHz |
| RVIC | 8387 | 7936 | 45 MHz |
| IMC\_MAX | 9042 | 7967 | 33.333 MHz |