Министерство науки и высшего образования Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

“НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО”

Факультет ПИиКТ

ОТЧЁТ

По лабораторной работе №4

По предмету: Цифровая схемотехника

Вариант: 1

Студент:

Степанов М. А.

Группа P33301

Преподаватель:

Салонина Екатерина Александровна

Санкт-Петербург

2023

Описание задания

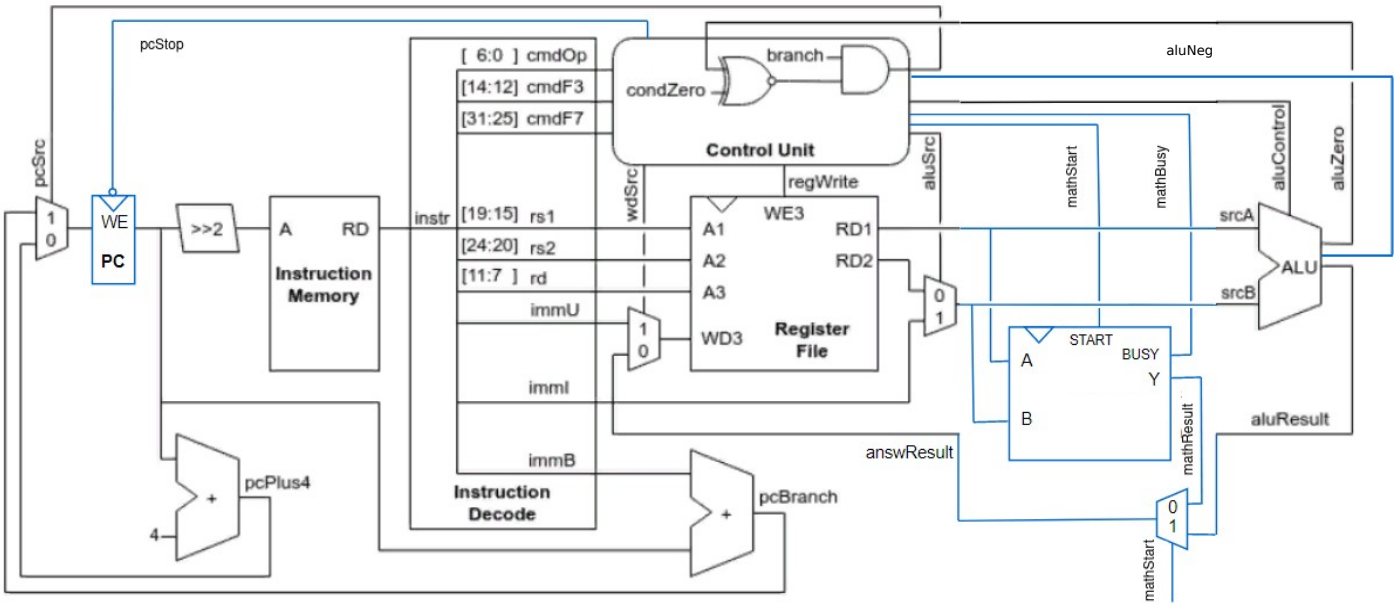
В лабораторной работе вам предлагается разобраться во внутреннем устройстве простейшего процессорного ядра архитектуры RISC-V. Результатом изучения микроархитектуры процессорного ядра и системы команд RISC-V станут ваши функциональные и нефункциональные модификации ядра.

Основное задание:

1. Расширить систему команд процессора двумя новыми командами, в соответствии с вашим вариантом: **BLT** и **FUN**;

2. Подготовить тестовое окружение системного уровня и убедиться в корректности вашей реализации путём запуска симуляционных тестов.

Микроархитектурная схема



Модификации:

1. Регистр **PC** теперь с разрешением записи, за счет чего реализуется многотактовая команда: когда команда требует выполнения в несколько тактов, из Control Unit подается сигнал **pcStop**, который на входе в регистр инвертируется и запрещает наращивание счетчика;

2. Параллельно с АЛУ вставлен блок вычисления математической функции **Function Calculator**, в котором и реализуется логика команды FUN;

3. **srcA (rd1)** и **srcB** теперь идут не только в АЛУ, но и во входы A и B модуля Function Calculator, но записываются в его внутренние регистры лишь по фронту тактового сигнала и активной линии **mathStart**;

4. **mathBusy** – выходная линия математического блока, которая активна ровно на период расчетов – по ней определяется момент, когда можно снимать результат;

5. Расширен тракт результата вычислений, теперь aluResult не попадает в регистровый файл напрямую, а выбирается через мультиплексор (управ. сигнал **mathStart**)

6. Добавлен флаг в АЛУ — флаг отрицательного числа.

Описание алгоритмов функционирования добавленных инструкций

1) ***BLT***. Инструкция перехода, однотактовая. Совершает переход на указанная метка, если значение первого регистра меньше, чем значение второго регистра. Для этой команды был добавлен провод, по которому передается сигнал указывающий на негативный результат в АЛУ.

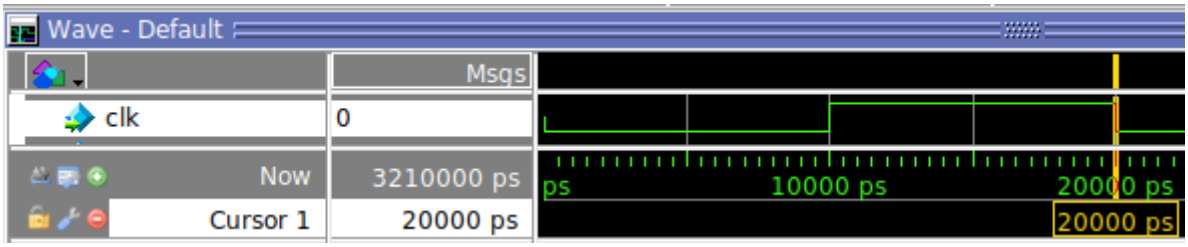
   Для тестирования команды была написана программа, которая выполняет несколько загрузок чисел в регистры, при этом между загрузками вставлены переходы BLT, некоторые должны совершить переход, а некоторые нет. Затем по временной диаграмме в массиве регистров смотрим, какие переходы произошли, а какие нет.

2) ***FUN.*** Инструкция многотактовая. Как только команда декодировалась, Control Unit останавливает счетчик команд путем выставления сигнала psStop, и вместе с открытием вентилей нужных регистров выключает сигнал mathStart в математическом блоке. Блок начинает вычисления и устанавливает сигнал mathBusy в активное значение, и во время записи результата сигнал mathStart снова подается и mathBusy сбрасывается.

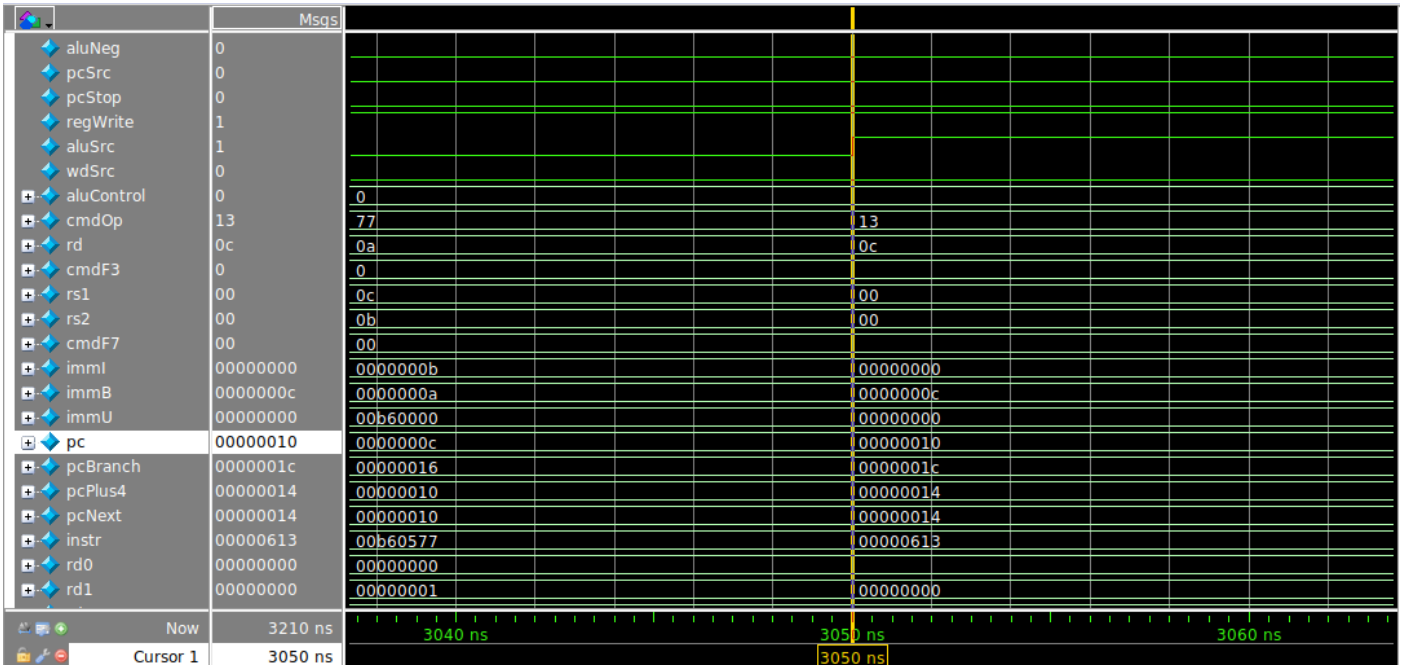
   Для тестирования просто загружаем значения в регистры вызываем команду и смотрим результат по временной диаграмме.

Временны́е диаграммы для подсчета кол-ва тактов

Для самой быстрой инструкции (любая кроме HYP): 20 ns // 1 такт



Для самой медленной инструкции (HYP): 2920 ns // 146 такта



Временны́е параметры проекта

Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Logic %	Net %	Requirement
Unconstrained Paths (1)												
(none) (10)												
Path 1	∞	1	1	1	sm_top/f0/data_reg[0]/C	sm_top/f0/q_reg[0]/D	0.288	0.147	0.141	51.1	48.9	-∞
Path 2	∞	1	1	1	sm_top/f0/data_reg[1]/C	sm_top/f0/q_reg[1]/D	0.288	0.147	0.141	51.1	48.9	-∞
Path 3	∞	1	1	1	sm_top/f0/data_reg[2]/C	sm_top/f0/q_reg[2]/D	0.288	0.147	0.141	51.1	48.9	-∞
Path 4	∞	1	1	1	sm_top/f0/data_reg[3]/C	sm_top/f0/q_reg[3]/D	0.288	0.147	0.141	51.1	48.9	-∞
Path 5	∞	1	1	1	sm_top/f1/data_reg[0]/C	sm_top/f1/q_reg[0]/D	0.288	0.147	0.141	51.1	48.9	-∞
Path 6	∞	1	1	1	sm_top/sm_cpu/..._reg_reg[0]/C	sm_top/sm_cpu/calc/m/a_reg[0]/D	0.288	0.147	0.141	51.1	48.9	-∞
Path 7	∞	1	1	1	sm_top/sm_cpu/..._reg_reg[1]/C	sm_top/sm_cpu/calc/m/a_reg[1]/D	0.288	0.147	0.141	51.1	48.9	-∞
Path 8	∞	1	1	1	sm_top/sm_cpu/..._reg_reg[2]/C	sm_top/sm_cpu/calc/m/a_reg[2]/D	0.288	0.147	0.141	51.1	48.9	-∞
Path 9	∞	1	1	1	sm_top/sm_cpu/..._reg_reg[3]/C	sm_top/sm_cpu/calc/m/a_reg[3]/D	0.288	0.147	0.141	51.1	48.9	-∞
Path 10	∞	1	1	1	sm_top/sm_cpu/..._reg_reg[4]/C	sm_top/sm_cpu/calc/m/a_reg[4]/D	0.288	0.147	0.141	51.1	48.9	-∞

Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Logic %	Net %	Requirement
Unconstrained Paths (1)												
(none) (10)												
Path 11	∞	5	5	24	sm_top/f2/q_reg_rep[0]/C	HEX1[0]	6.731	3.710	3.021	55.1	44.9	∞
Path 12	∞	5	5	24	sm_top/f2/q_reg_rep[0]/C	HEX1[2]	6.731	3.710	3.021	55.1	44.9	∞
Path 13	∞	5	5	24	sm_top/f2/q_reg_rep[0]/C	HEX1[3]	6.731	3.710	3.021	55.1	44.9	∞
Path 14	∞	5	5	24	sm_top/f2/q_reg_rep[0]/C	HEX1[4]	6.731	3.710	3.021	55.1	44.9	∞
Path 15	∞	5	5	24	sm_top/f2/q_reg_rep[0]/C	HEX1[5]	6.731	3.710	3.021	55.1	44.9	∞
Path 16	∞	5	5	24	sm_top/f2/q_reg_rep[0]/C	HEX1[6]	6.731	3.710	3.021	55.1	44.9	∞
Path 17	∞	5	5	24	sm_top/f2/q_reg_rep[0]/C	HEX2[1]	6.730	3.709	3.021	55.1	44.9	∞
Path 18	∞	5	5	24	sm_top/f2/q_reg_rep[0]/C	HEX2[5]	6.730	3.709	3.021	55.1	44.9	∞
Path 19	∞	5	5	24	sm_top/f2/q_reg_rep[0]/C	HEX5[5]	6.727	3.709	3.018	55.1	44.9	∞
Path 20	∞	5	5	24	sm_top/f2/q_reg_rep[0]/C	HEX5[6]	6.727	3.709	3.018	55.1	44.9	∞

Выводы

1) С одной стороной модификация проекта позволяет добавлять многотактовые команды при помощи остановки pc, с другой стороны данная возможность не слишком необходима для архитектуры RISC-V, так как добавление каждой многотактовой программы потребует либо записи промежуточных значений в регистрах, либо добавление новых функциональных блоков, которые значительно усложнят микроархитектуру, а также сам функциональный блок вычисляет довольно специфичную функцию необходимую лишь в узконаправленных задачах.

2) Альтернативное решение для флагов АЛУ — стоило бы составить комбинаторную схему, поддерживающую все возможные переходы с простыми логическими операторами, так как данное нововведение практически не усложнит микроархитектуру, но даст более широкие возможности, чем использование лишь перехода BLT.

Альтернативное решение для функции — данную задачу проще решить на программном уровне и использовать составленную программу как функцию. Если же микроархитектура разрабатывается для узконаправленных задач (вроде построение лучей света и пр.), то удобнее было бы добавить операцию извлечения кубического корня и завести функциональный блок с заранее подсчитанными значениями для входного параметра, что позволило бы асинхронно извлекать ответ и выполнять данную операцию за один такт.

3) С учетом уже имеющегося работоспособного блока вычисления функции, моё решение позволяет сэкономить на времени разработки. Более того, разработка блока с заранее подготовленными значениями функции занимало бы много физических ресурсов на плате, так как состояло бы из тысяч элементов (для 32 бит), для увеличения же моего решения до 32 бит требуется лишь изменить входной параметр схемы.

4) Где взять временные диаграммы после синтеза? Зачем может понадобиться вычислять f = 3*a + 2*cbrt(b)?