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: 6

Студент:

Казаченко Р. О.

Группа P33301

Преподаватель:

Салонина Екатерина Александровна

Санкт-Петербург

2023

Описание задания

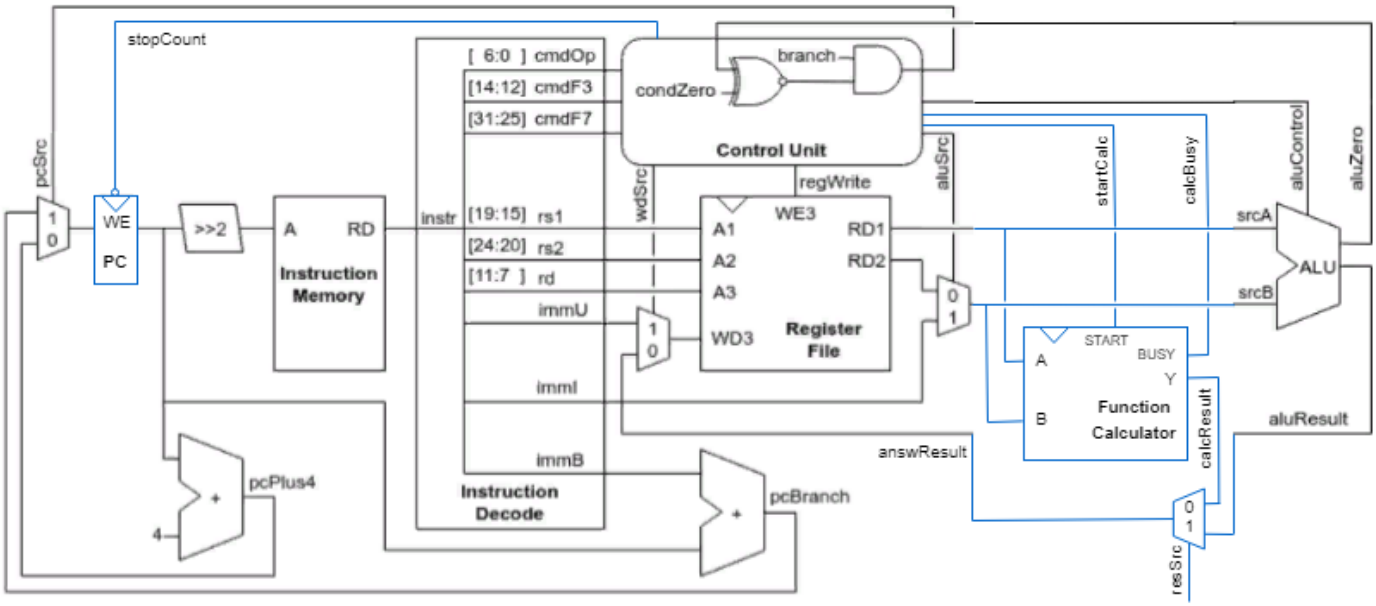
В лабораторной работе вам предлагается разобраться во внутреннем устройстве простейшего процессорного ядра архитектуры RISC-V. Результатом изучения микроархитектуры процессорного ядра и системы команд RISC-V станут ваши функциональные и нефункциональные модификации ядра.

Основное задание:

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

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 **stopCount**, 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 HYP;

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 **startCalc**;

4. **calcBusy** – 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 **resSrc**) в **answResult**, чтобы у нас была возможность записать в регистровый файл и результат вычислений математического блока (**calcResult**).

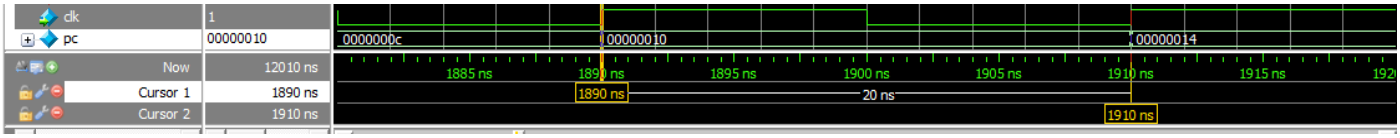
Описание алгоритмов функционирования добавленных инструкций

1) ***XORI***. Инструкция однотактовая и полностью повторяет алгоритм выполнения инструкции *ADDI*, за исключением операции в АЛУ. Команда декодируется, из регистрового файла достается значение нужного регистра, константа из кода команды расширяет свой знак и всё это попадает в АЛУ. Там применяется операция побитового XOR и результат помещается в указанный регистр в регистровом файле.

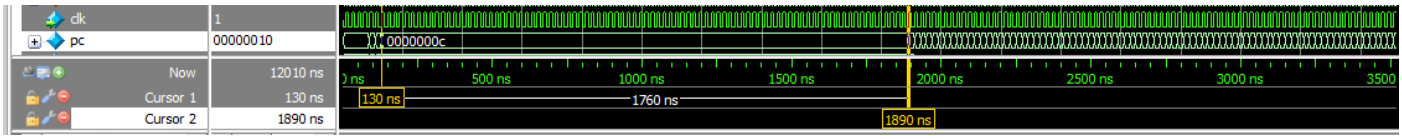
2) ***HYP.*** Инструкция многотактовая. Как только команда декодировалась, Control Unit останавливает счетчик команд путем выставления сигнала stopCount, и вместе с открытием вентилей нужных регистров подает сигнал startCalc в математический блок. Блок начинает вычисления и устанавливает сигнал calcBusy в активное значение, и только по спаду этого сигнала мы можем записать результат вычислений в нужный регистр, предварительно выбрав правильный источник результата resSrc.

Временны́е диаграммы для подсчета кол-ва тактов

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



Для самой медленной инструкции (HYP): 1760 ns // 88 тактов



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 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	∞

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	-∞

Выводы

После модификаций улучшилась расширяемость процессора в плане добавления команд: благодаря сигналу stopCount можно добавлять многотактовые команды. Однако внедрение самого математического блока получилось не самым выгодным образом: одно из преимуществ перехода процессора к многотактовому – избавиться от большого количества АЛУ – не получило реализации в моей схеме, а наоборот, в функциональном блоке добавились новые сумматоры и умножители. По-хорошему, нужно перерабатывать всю схему и оставить на ней из всех сумматоров, умножителей и т. п. лишь один АЛУ, однако это очень трудоёмкий процесс, который потребует переписывания алгоритмов работы всех команд. Но накладные финансовые расходы на это в разы выше, чем моя реализация, учитывая то, что всё это необходимо всего для одной команды. Если многотактовых команд изначально планировалось бы больше, то логичнее было бы в корне переработать архитектуру процессора, как это было сказано выше.