|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации Мытищинский филиал**  **Федерального государственного бюджетного образовательного учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МФ МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ *Космический .*

КАФЕДРА *Прикладной математики, информатики и вычислительной техники*

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

***К ПРОЕКТУ МАКЕТИРОВАНИЯ***

***Учебного CISC процессора***

Студент *К3-63Б*  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_** *Зотов Е. А ы*

(Группа) (Подпись,дата) (Фамилия.И.О.)

Руководитель проекта **\_\_\_\_\_\_\_\_\_\_\_\_** *Ефремов Н.В ы*

(Подпись, дата) (Фамилия.И.О.)

*2025 г.*

Содержание

[Введение 3](#_Toc185949573)

1. Файлы проекта [4](#_Toc185949575)

2. Назначения входов и выходов 5

3. Схема тактирования 8

4. Схема остановы 9

5. Просмотр и редактирование содержимого памяти в реальном времени 11

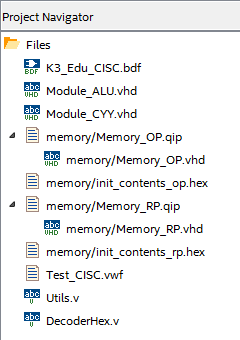
[6. Тестирование и отладка процессора 14](#_Toc185949582)

**Введение**

Проект макетирования K3\_EDU\_CISC (К3 Учебный CISC процессор) – проект реализации учебного процессора в кристалле ПЛИС стенда Altera DE2-115 и для проведения натурных испытаний. Он основан на файлах проекта CYY\_final.

1. **Файлы проекта**

Файлы проекта были переименованы и структурированы для лучшего понимания структуры проекта:



*Рис. NUM. Файлы проекта*

|  |  |  |
| --- | --- | --- |
| **Название в CYY\_final** | **В этом проекте** | **Пояснение** |
| - | K3\_Edu\_CISC.bdf | Представление процессора в виде блок-диаграммы, файл верхнего уровня |
| ctrl\_un\_BO.vhd | Module\_ALU.vhd | Операционное устройство для выполнения умножения и сложения (потенциально АЛУ)  Модуль переименован в Module\_ALU, в остальном файл перенесен без изменений |
| CYY\_final.vhd | Module\_CYY.vhd | Центральное устройство управления ЦУУ  Модуль переименован в Module\_CYY, убраны декларации модулей ОП, РП, АЛУ, раскомментированы входы и выходы для подключения этих модулей в блок-диаграмме |
| Module\_OP.vhd | Memory\_OP.vhd | 1-Port и 2-Port RAM, сгенерированы с помощью MegaWizard (Подробнее в главе «Просмотр и редактирование содержимого памяти») |
| RP2.vhd | Memory\_RP.vhd |
| - | Viewer\_RP.vhd |
| OP\_init.mif | init\_contents\_op.hex | Файл инициализации ОП, перенесен без изменений |
| file\_init\_RP.mif | init\_contents\_rp.hex | Файл инициализации РП, перенесен без изменений |
| - | Utils.v | Описание bus\_comparator, clock\_divider, debounce\_circuit |
| - | DecoderHex.v | Шестнадцатеричный декодер для 7-сегментного индикатора (Decoder\_Hex) |

*Таблица 0. Файлы проекта и изменения*

1. **Назначения входов и выходов**

В ходе разработки макета процессора, следующие назначения переключателей, кнопок и индикаторов показались наиболее удобными:

|  |  |  |
| --- | --- | --- |
| **Input** | **Название линии** | **Пояснение** |
| CLOCK\_50 | SIGNAL\_CRYSTAL\_CLOCK | 50 MHz clock |
| SW[17] | SIGNAL\_CLOCK\_SOURCE\_SELECT | Выбор источника тактирования:  0 – с генератора  1 – тактовая кнопка |
| SW[16] | SIGNAL\_BREAK\_ENABLE\_ADDR\_MATCH | Разрешить останову по достижении заданного адреса |
| SW[15] | SIGNAL\_BREAK\_ENABLE\_COMMAND\_END | Разрешить останову по окончании выполнения команды |
| SW[14..7] | BUS\_BREAKPOINT\_ADDR[7..0] | Установка адреса остановы |
| SW[6..0] | - | *Не используется* |
| KEY[3] | SIGNAL\_CLOCK\_MANUAL\* | Кнопка тактирования |
| KEY[2] | SIGNAL\_BREAK\_IGNORE\* | Разрешение тактирования (выход из остановы) |
| KEY[1] | - \* | *Не используется* |
| KEY[0] | SIGNAL\_RESET\* | Сброс (то же самое, что и set) |

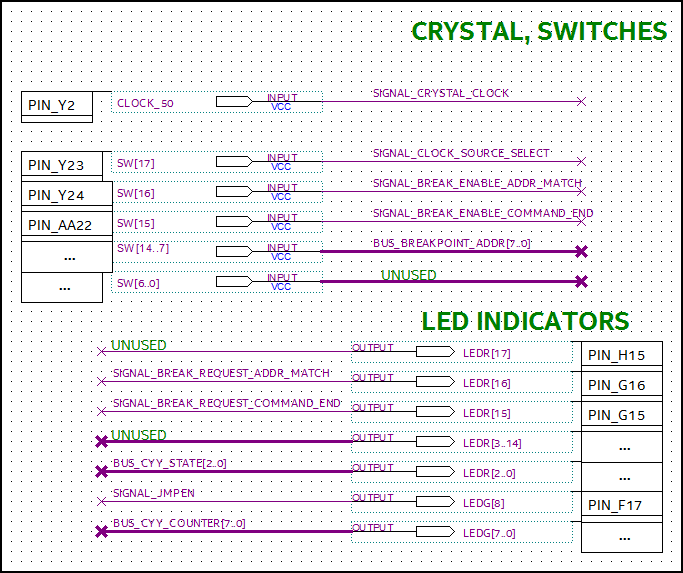
*Таблица 1. Назначения входных сигналов*

*\* - Примечание: KEY[3] .. KEY[0] соединены с линией не напрямую, а через схему подавления дребезга. Также сигналы инвертированы для работы по схеме ACTIVE-HIGH, т.е. выдают лог. единицу когда кнопка нажата (см. Рис. «Назначения кнопок и схема подавления дребезга»)*

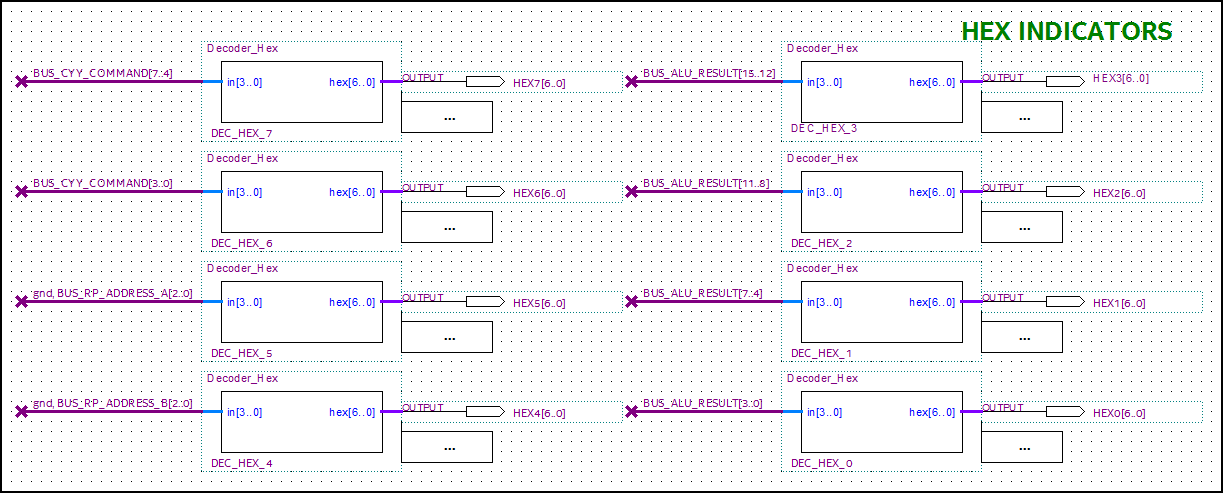
|  |  |  |
| --- | --- | --- |
| **Output** | **Название линии** | **Пояснение** |
| LEDR[17] | - | *Не используется* |
| LEDR[16] | SIGNAL\_BREAK\_REQUEST\_ADDR\_MATCH | Процессор остановлен по достижении заданного адреса |
| LEDR[15] | SIGNAL\_BREAK\_REQUEST\_COMMAND\_END | Процессор остановлен по окончании выполнения команды |
| LEDR[3..14] | - | *Не используется* |
| LEDR[2..0] | BUS\_CYY\_STATE[2..0] | Вывод состояния ЦУУ |
| LEDG[8] | SIGNAL\_JMPEN | Разрешение условного перехода (загорается только при ПР=10) |
| LEDG[7..0] | BUS\_CYY\_COUNTER[7..0] | Вывод счетчика команд ЦУУ |
| HEX 7..6 | BUS\_CYY\_COMMAND[7..0] | Вывод выполняемой команды |
| HEX 5 | gnd, BUS\_RP\_ADDRESS\_A[2..0] | Вывод адреса РП A |
| HEX 4 | gnd, BUS\_RP\_ADDRESS\_B[2..0] | Вывод адреса РП B |
| HEX 3..0 | BUS\_ALU\_RESULT[15..0] | Вывод результата АЛУ |

*Таблица 2. Назначения выходных сигналов*

В блок-диаграмме проекта все входы и выходы собраны вместе для возможности быстрого редактирования. Это позволяет, например, быстро перенести проект на стенд Altera DE-70:

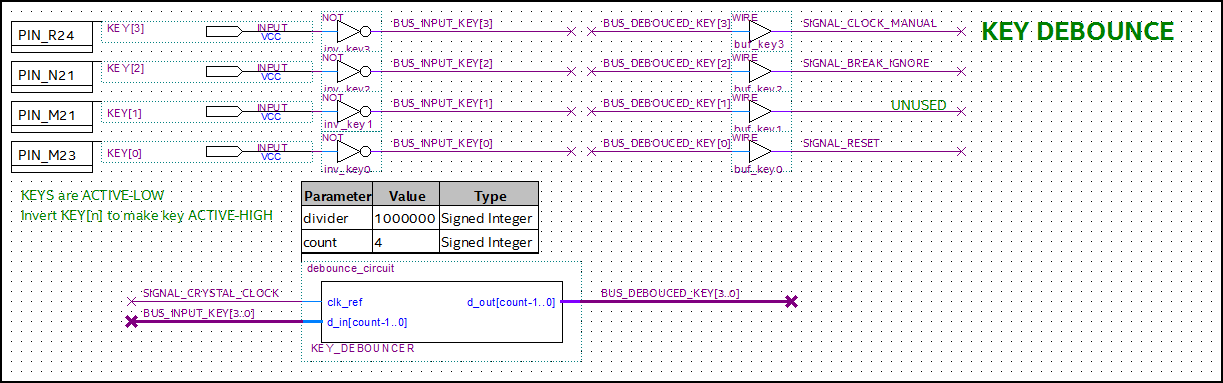


*Рис. NUM. Назначения переключателей и светодиодов*



*Рис. NUM. Назначения HEX индикаторов*

Важно! Чтобы не привязываться к коду (прямой, обратный, дополнительный) индикаторы выводят значения в шестнадцатеричном формате. Так, студенты должны уметь быстро переводить двоичный формат в шестнадцатеричный и обратно. Также есть вариант предложить студенту написать свой декодер для преобразования в десятичный формат и код, соответствующий варианту его проекта.



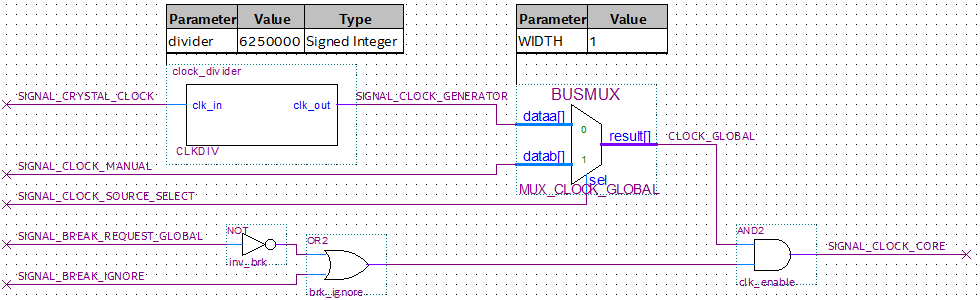
*Рис. NUM. Назначения кнопок и схема подавления дребезга*

1. **Схема тактирования**

Схема тактирования обеспечивает возможность тактировать процессор как вручную с помощью кнопки, так и автоматически от внутреннего генератора. Выбор источника тактирования осуществляется с помощью переключателя (см. Таблицу 1), расположенного на плате стенда.

Частота внутреннего генератора – 50 МГц. Чтобы наблюдать за постепенной работой процессора в автоматическом режиме частоту требуется существенно занизить. Для этого используется компонент clock\_divider, который представляет собой счетчик с настраиваемым значением переполнения. При переполнении сигнал clk\_out инвертируется. Таким образом, при значении переполнения 6 250 000 выходная частота составит 50 000 000 / 6 250 000 / 2 = 4 Гц.

В схеме тактирования также предусмотрена возможность запрета тактирования процессора. Так, для реализации возможности задавать точки остановы нужно лишь запретить тактирование, а для продолжения работы разрешить (Подробнее в главе «*Схема остановы»*).



*Рис. NUM. Схема тактирования процессора*

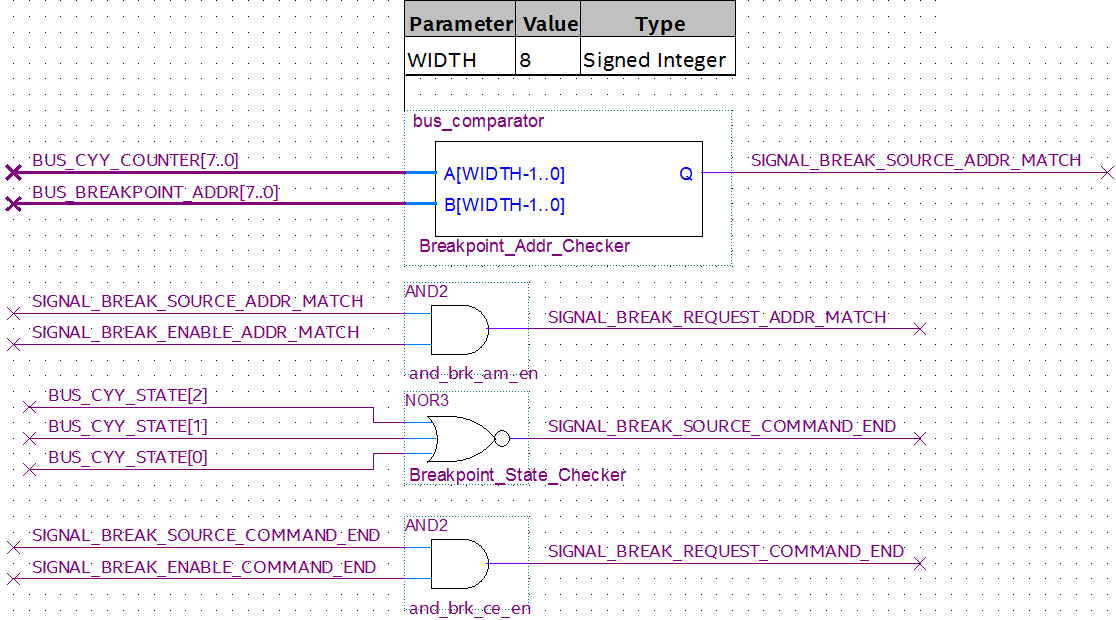
1. **Схема остановы**

Схема остановы обеспечивает возможность программной или аппаратной приостановки работы процессора. Основная задача данной схемы — генерация сигнала запрета тактирования процессора при поступлении одного или нескольких сигналов запроса остановки, обозначенных на схеме как *SIGNAL\_BREAK\_REQUEST\_\*название запроса остановы\**.

Сигнал BREAK\_REQUEST\_GLOBAL может формироваться от следующих стандартных источников:

* COMMAND\_END — остановка процессора по завершении выполнения текущей команды. Используется, например, для покомандного выполнения программы.
* ADDR\_MATCH — остановка при достижении заранее заданного пользователем адреса. Применяется для реализации точек остановы (breakpoints).

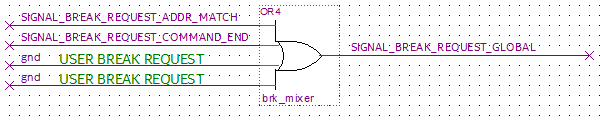
Каждый из этих источников может быть включён или отключён с помощью соответствующих переключателей (см. Таблицу 1), расположенных на плате стенда. Это позволяет гибко управлять условиями остановки при отладке программы.



*Рис. NUM. Логика формирования запросов ADDR\_MATCH и COMMAND\_END*

**Расширяемость схемы**

Схема остановы может быть легко дополнена, пользовательскими сигналами остановки. Для этого у элемента brk\_mixer (логического ИЛИ) нужно заменить линию сигнала USER\_BREAK\_REQUEST своей линией запроса:



*Рис. NUM. Использование лог. ИЛИ для остановы от любого запроса*

Это даёт возможность расширять функциональность схемы, добавляя новые условия для приостановки процессора, такие как:

* Совпадение наблюдаемых флагов, значений и т.п.
* Команда HALT

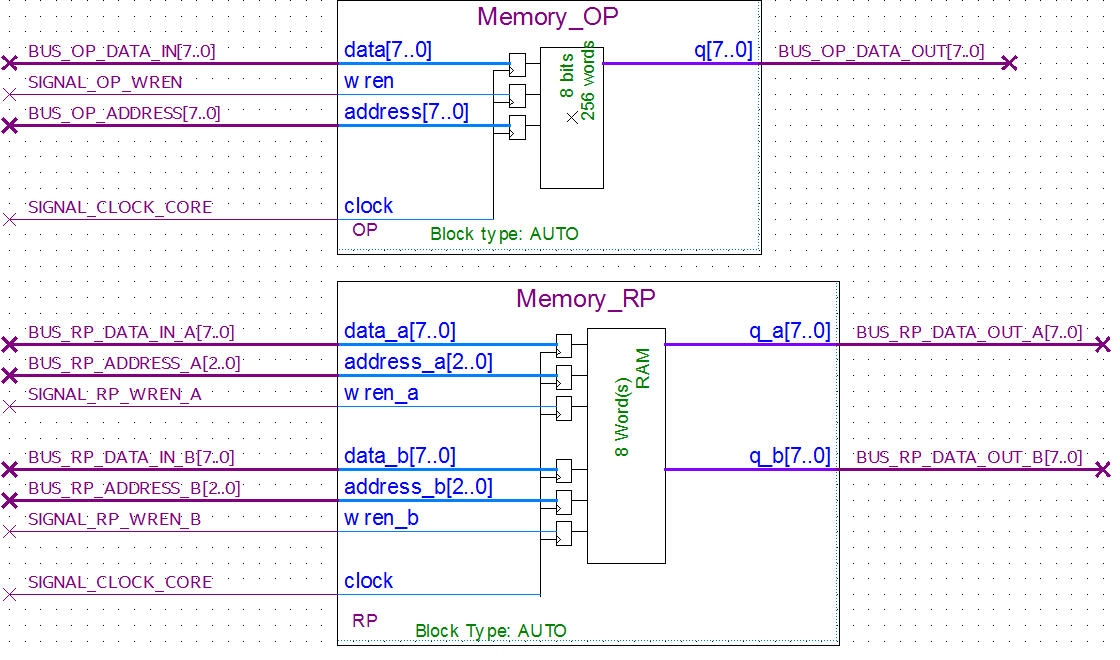
**Выход из остановы**

За выход из остановы отвечает линия BREAK\_IGNORE, которая подключена к кнопке на стенде (см. Таблицу 1). При ее нажатии сигнал запрета тактирования игнорируется, что разрешает дальнейшее тактирование процессора (см. Рис. *«Схема тактирования процессора»*). Таким образом, процессор может продолжить выполнение команд.

Важно! Стоит заметить, что кнопка продолжения выполнения программы только разрешает тактирование процессора. Поэтому если процессор тактируется вручную, просто нажатие кнопки продолжения не приведет к изменениям. Важно **удерживать кнопку продолжения во время подачи тактов** до тех пор, пока процессор не выйдет из остановы.

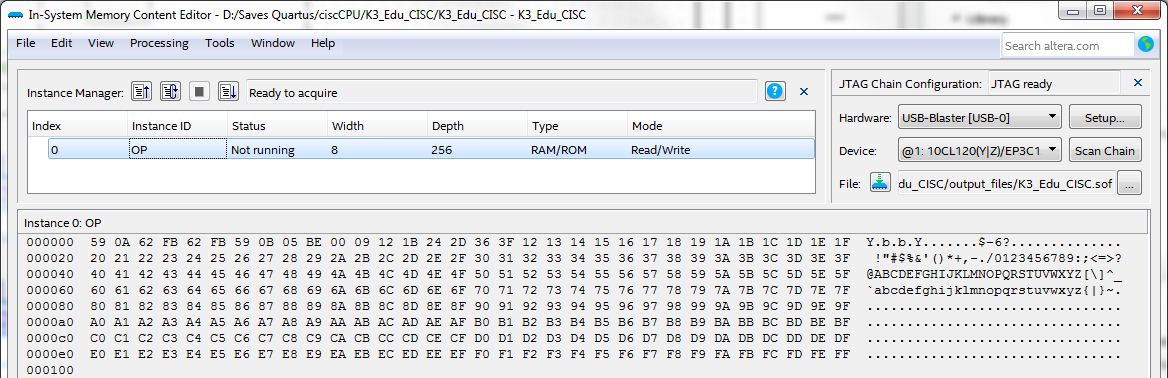
1. **Просмотр и редактирование содержимого памяти в реальном времени**

В качестве Оперативной памяти (ОП) и Регистровой памяти (РП) используются блоки IP-памяти, предоставляемые программой Quartus. Главными преимуществами использования этих блоков является возможность настраивать параметры памяти с помощью диалогового окна MegaWizard, а также возможность просмотра и редактирования содержимого однопортовой памяти с помощью *In-System Memory Content Editor*.



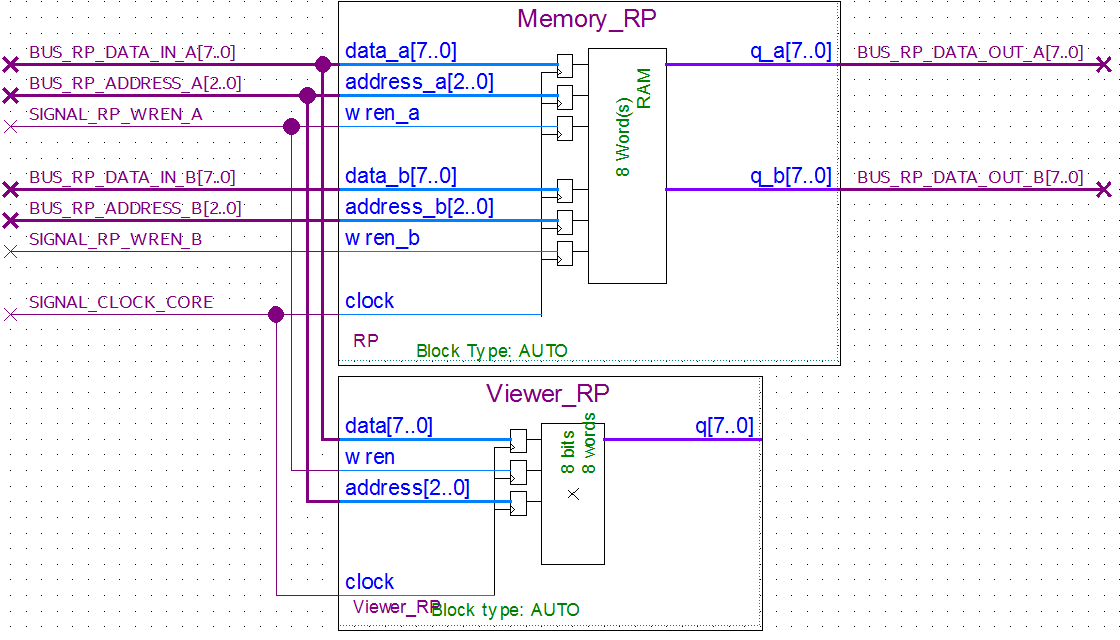
*Рис. NUM. Блоки IP-Памяти: 1-Port ОП и 2-Port РП*

**Механизм работы**

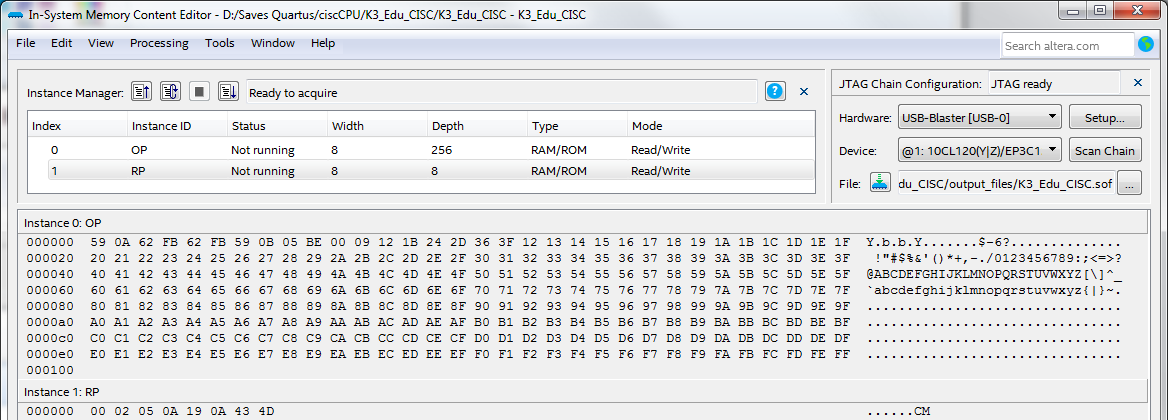
Для просмотра и редактирования содержимого без внесения изменений в схему используется следующий механизм: однопортовая память во время компиляции преобразуется в двухпортовую, второй порт используется для просмотра и редактирования содержимого памяти через JTAG:

*Рис. NUM. Просмотр ОП в In-System Memory Content Editor*

К сожалению, подобным образом не получится просматривать и редактировать содержимое регистровой памяти, так как второй порт используется процессором. Однако поскольку запись в регистровую память осуществляется только через порт A, то мы можем дублировать блок регистровой памяти однопортовым блоком памяти с такими же параметрами и файлом инициализации. В проекте такой блок называется Viewer \_RP:



*Рис. NUM. Блок Viewer \_RP дублирует Memory\_RP*

Теперь есть возможность просматривать ОП и РП:

*Рис. NUM. Просмотр ОП и РП в In-System Memory Content Editor*

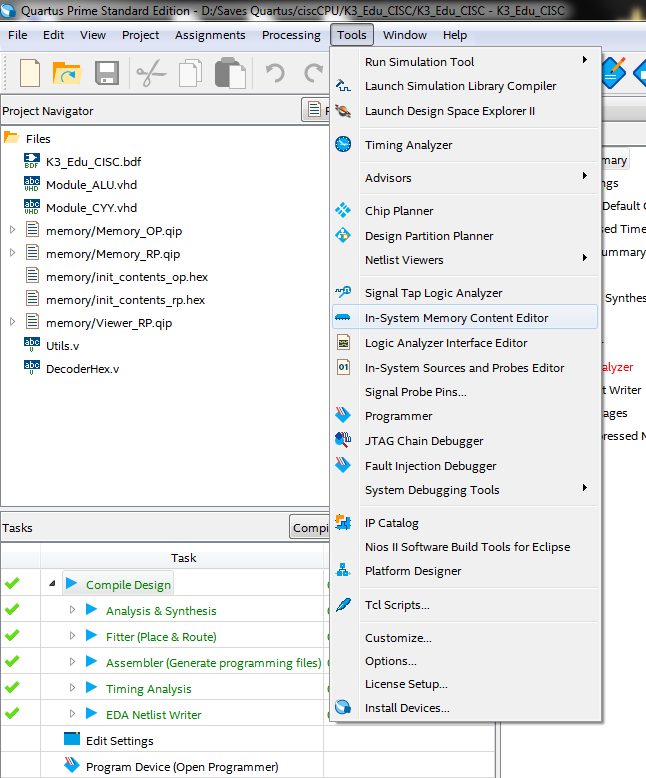
Важно! Редактирование содержимого RP приведет только к изменению содержимого Viewer\_RP, а значения Memory\_RP не поменяются! OP можно редактировать, так как для OP используется блок однопортовой памяти.

1. **Загрузка в стенд и работа с процессором**

**Загрузка в стенд**

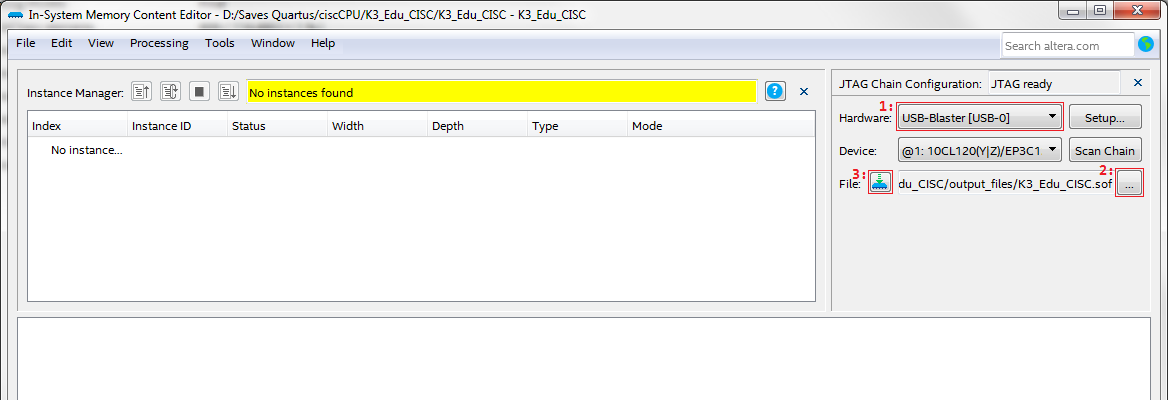
Для загрузки проекта в стенд необходимо выполнить следующие шаги:

1. Компилируем проект
2. Заходим в In-System Memory Content Editor:



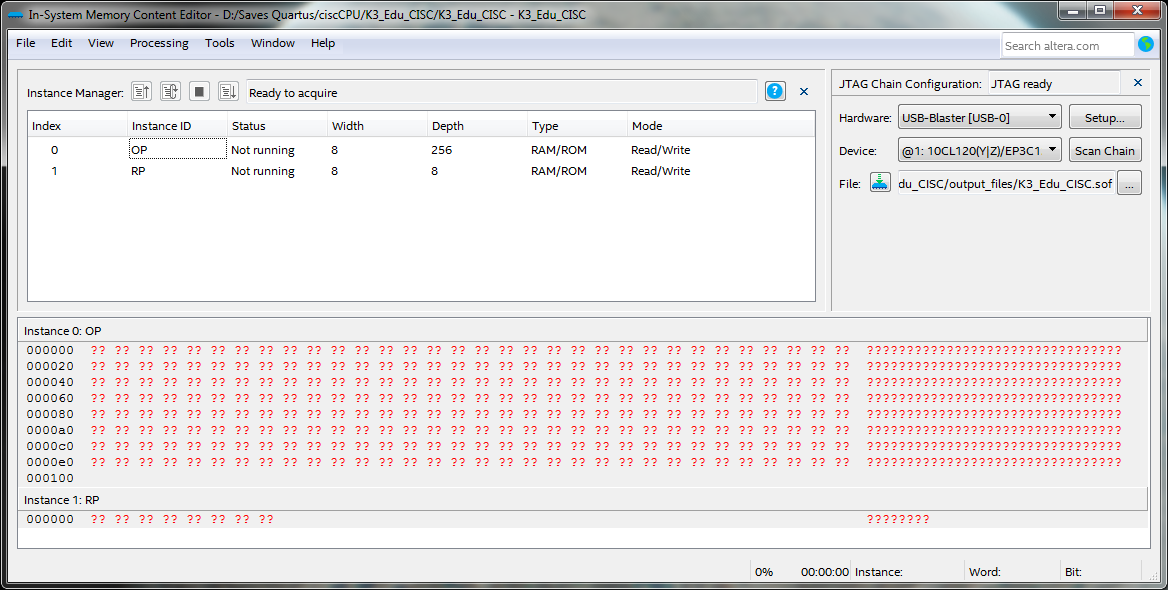
*Рис. NUM. Меню Tools, In-System Memory Content Editor*

1. В разделе Hardware выбираем USB-Blaster
2. В разделе File выбираем output\_files/K3\_Edu\_CISC.sof
3. Нажимаем Download



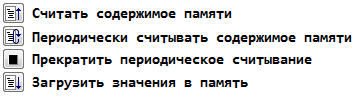
*Рис. NUM. Выбор USB-Blaster, .sof файл проекта, загрузка в стенд*

После загрузки проекта в стенд можем наблюдать памяти OP и RP:

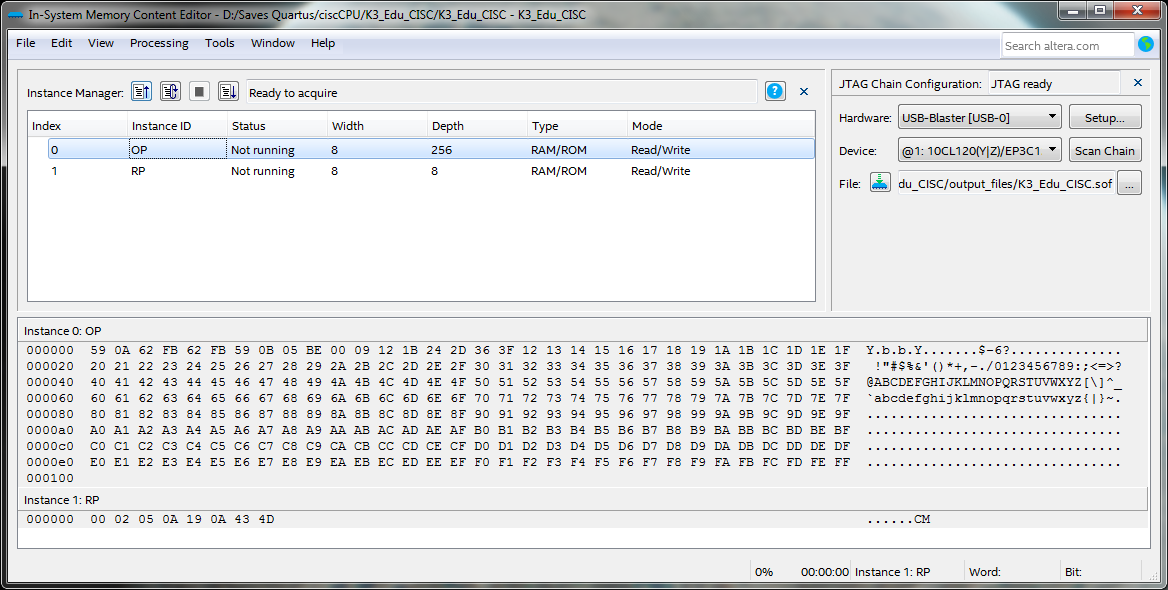


*Рис. NUM. OP и RP в In-System Memory Content Editor*

Для работы с памятью нам доступны следующие действия:



*Рис. NUM. Пояснения для кнопок работы с памятью*



*Рис. NUM. Отображение содержимого OP и RP после считывания*

**Работа с процессором**

Перед проверкой корректности работы напомним список команд, с которыми работает процессор, формат и результат работы каждой команды:

|  |  |  |  |
| --- | --- | --- | --- |
| **КОП** | **Формат** | **Тип команды** | **Пояснение** |
| 00 | R1, R2 | Регистр-регистр | Сложение |
| 01 | R1, R2, Offset[7..0] | Память-регистр | Умножение |
| 10 | Offset[5..0] | Переход | Условный переход |
| 11 | - | - | *Не используется* |

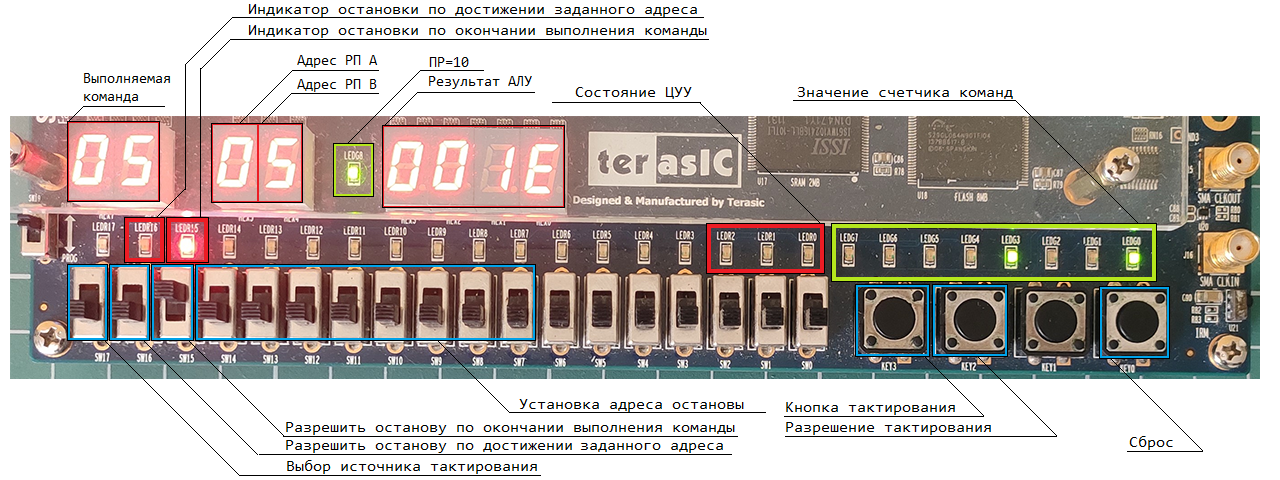
*Таблица 3. Команды учебного процессора*

|  |  |  |
| --- | --- | --- |
| **Пример** | **Разметка бит** | **Результат** |
| sum RA, RB | 00 AAA BBB | RA += RB |
| mul X(RA), RB | 01 AAA BBB XXXXXXXX | OP[RA + X] \*= RB |
| jmp X | 10 XXXXXX | СК += X , если результат пред. операции > 0 (ПР=10) |

*Таблица 4. Формат (разметка) команд учебного процессора*

После загрузки процессора в стенд:

1. Устанавливаем разрешение остановы по окончании выполнения команды
2. Выбираем тактирование от генератора
3. Сбрасываем процессор: Для этого зажимаем кнопки *«Сброс»* и *«Разрешение тактирования»*
4. Нажимая кнопку *«Разрешение тактирования»* выполняем программу в покомандном режиме, после выполнения каждой команды проверяем содержимое памяти ОП и РП, значения индикаторов и светодиодов. *На этом этапе рекомендуется параллельно сверяться с диаграммой функционального моделирования процессора из проекта CYY\_final*



*Рис. NUM. Расположение и назначение переключателей, кнопок и индикаторов на стенде Altera DE-115*

Убеждаемся в корректной работе процессора, проверяем:

* Корректно ли процессор декодирует команды
* Корректно ли процессор вынимает адреса РП
* Корректно ли проводит считывание/запись данных в память
* Корректно ли проводит вычисления в АЛУ