**Федеральное государственное автономное образовательное учреждение высшего образования**

**"Национальный исследовательский университет**

**"Высшая школа экономики"**

**Московский институт электроники и математики им. Тихонова**

Департамент компьютерной инженерии

**Курс «Инструментальные средства и технология программирования»**

**ОТЧЕТ**

**о выполнении практической работы № 2.1**

тема работы: «Разработка конвейрного умножителя в среде Altera Quartus II»

Выполнили:

Студенты группы БИВ-155:

Пчелкин Д. А.

Иванов А. А.

Принял

к.т.н., ассистент. МИЭМ НИУ ВШЭ

Романова И.И.

Москва, 2018

# Задание

Выполнить действия, описанные в практической работе 2.1. – Часть 1, 2, 3. Оформить отчет, который должен включать: титульный лист, введение и постановку задачи, тему работы, описание всех этапов выполнения проекта, скриншоты (рисунки с подрисуночными подписями) ключевых моментов, выводы.

# Выполнение работы

## 1 Создание проекта

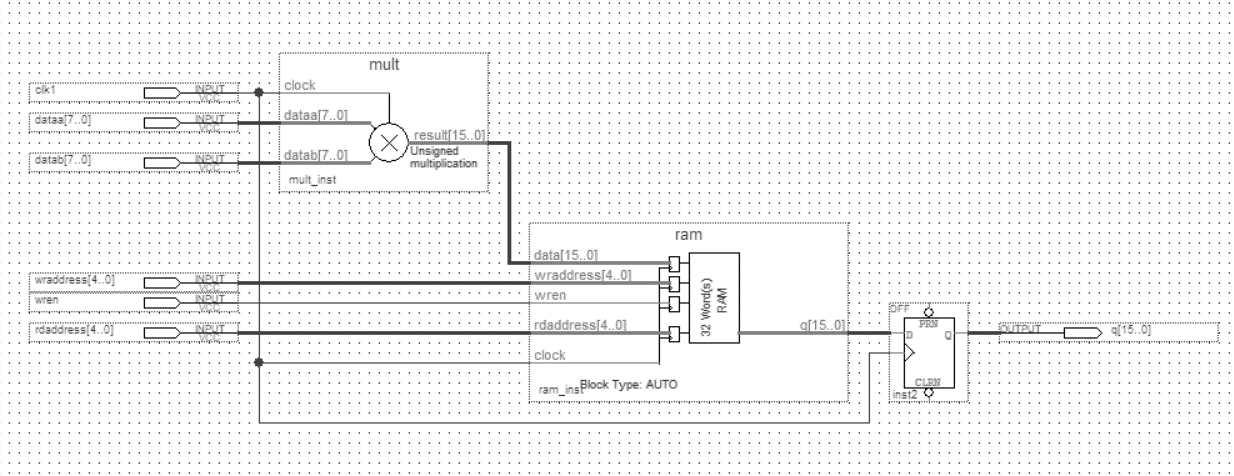


Рисунок 1. Схема полученного конвейерного умножителя

В схеме, представленной на рис. 1, изображен конвейерный умножитель, состоящий из умножителя 8x8 и блока оперативной памяти (RAM) 32x16. Для каждого из элементов предусмотрены по три входа от внешнего источника. На умножитель подается сигнал clock и входные восьмибитные данные, результат сохраняется в блоке памяти. Затем данные читаются из блока оперативной памяти, используя независимую шину адреса для чтения. Схема реализована с использованием мегафункций, созданных с помощь. Утилиты Altera Quartus II MegaWizard® Plug-in Manager.

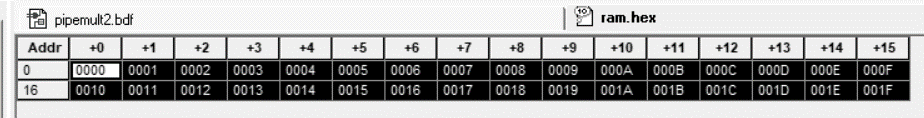


Рисунок 2. Содержимое файла для инициализации памяти

На рис. 2 представлены значения, хранящиеся в памяти при начальной инициализации. Память содержит 32 16-ти разрядных слова. На изображении значения представлены в шестнадцатеричном (HEX) представлении. Диапазон значений составляет промежуток от 0 до 31 с шагом 1. Файл создан с помощью редактора Altera Quartus II Memory Editor.

Анализ созданного проекта проводится с помощью инструмента Altera Quartus II Analysis & Elaboration.

## 2 Моделирование проекта

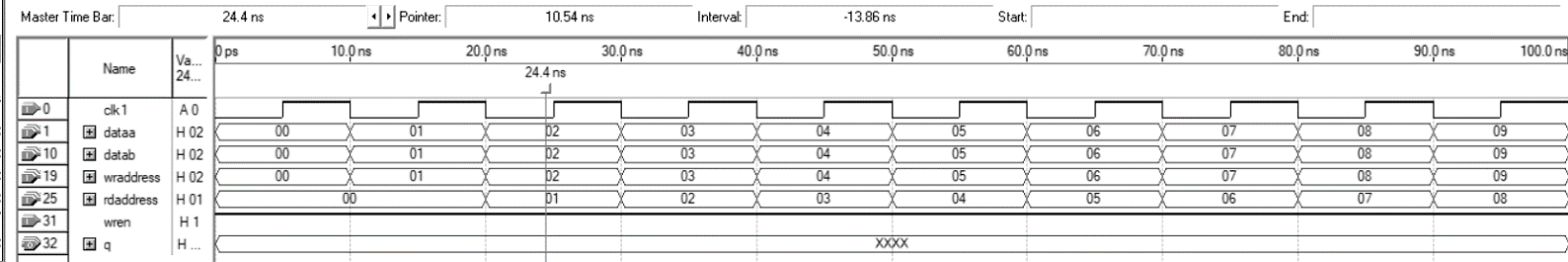


Рисунок 3. Значения сигналов, записанные в файл pipemult.cvwf

На рис. 3 представлены значения входных сигналов схемы, которые будут использованы для последующего функционального моделирования работы конвейерного умножителя. Для проверки корректности работы полученной схемы требуется задать сигналы: clk1, dataa, datab, wraddress, rdaddress, wren. Результат работы схемы будет выведен через выход q. Моделирование будет проводиться в течение 100 нс. Для сигнала clk задается период 10 нс, сигналы dataa, datab, rraddress и wraddress увеличиваются на 1, каждые 10 нс. Для сигнала rraddress вводится задаржка в 10 нс. На wren подается высокий уровень сигнала, q неопределен до завершения моделирования.

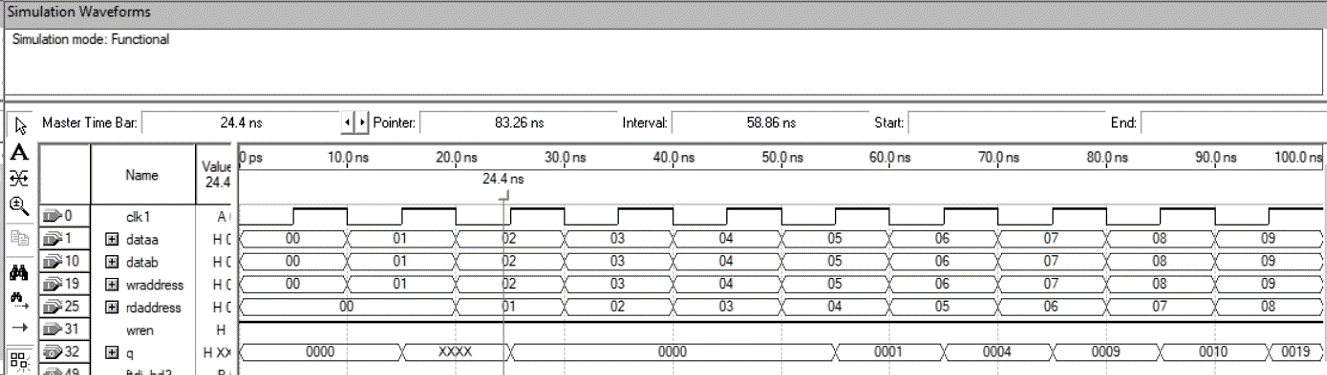


Рисунок 4. Результат моделирования

На рис. 4 представлен результат функционального моделирования работы полученной схемы. По переднему фронту тактового сигнала происходит последовательно умножение ‑ запись в ram ‑ вывод. Из-за того, что все операции идут последовательно, на первом тактовом сигнале происходит умножение, на втором ‑ запись в память, на третьем ‑ вывод. То есть первые два тактовых сигнала не дадут результата умножения и лишь на третьем будет вывод.

## 3 Анализ полученных результатов

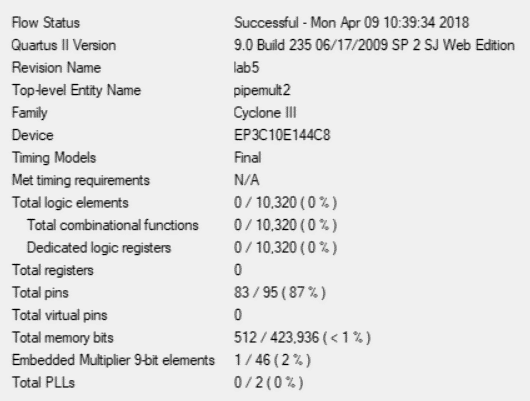


Рисунок 5. Отчет компилятора

На рис. 5 изображены результаты компиляции проекта. В том числе указаны:

1. Число используемых логических элементов: 0.
2. Число используемых пинов: 83/95.
3. Число битов используемой памяти: 512 бит.
4. Число используемых встроенных 9-битовых умножителей: 1/46.

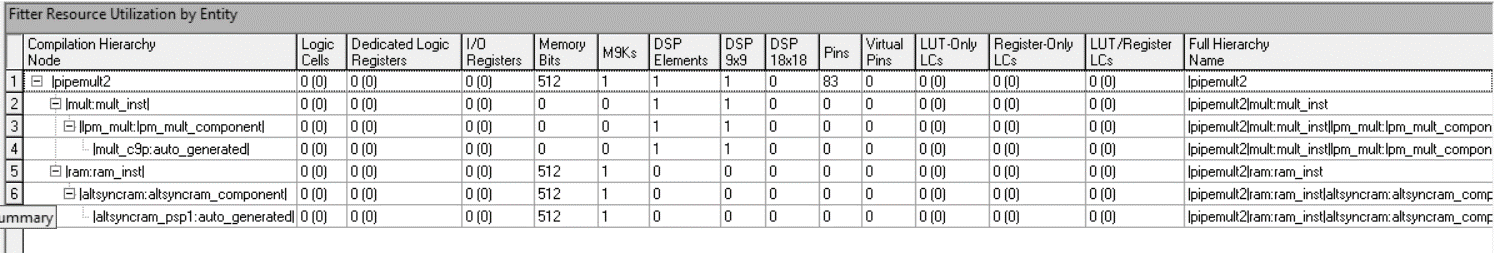


Рисунок 6. Используемые ресурсы

На рис. 6 представлены ресурсы, используемые для реализации мегафункций mult и ram.

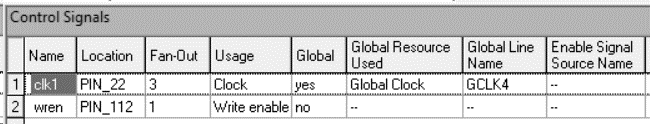


Рисунок 7. Раздел Control Signals отчета компилятора

На рис. 7 представлена информация о входных сигналах.

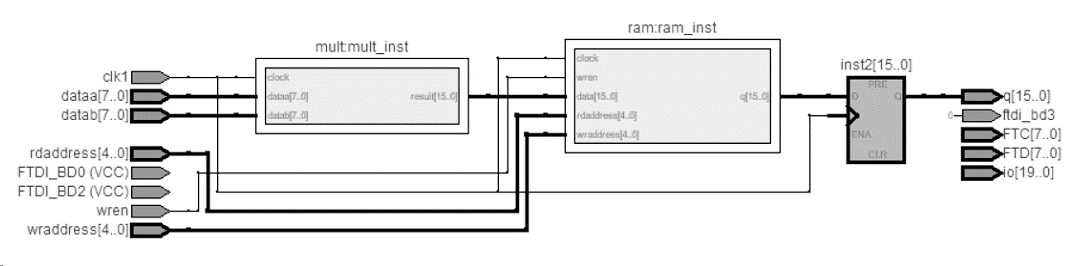


Рисунок 8. Отображение логической реализации проекта

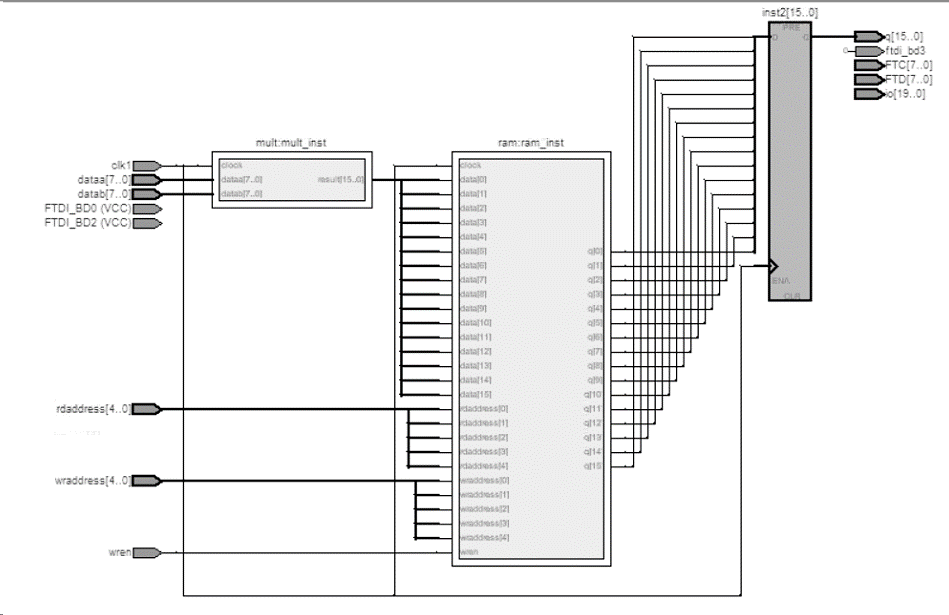


Рисунок 9. Развернутое отображение логической реализации проекта

На рис. 8-9 изображены графические представления логической реализации проекта с указанием контактов ввода/вывода, блоков mult и ram, а также выходных регистров в сокращенном и развернутом виде, соответственно.

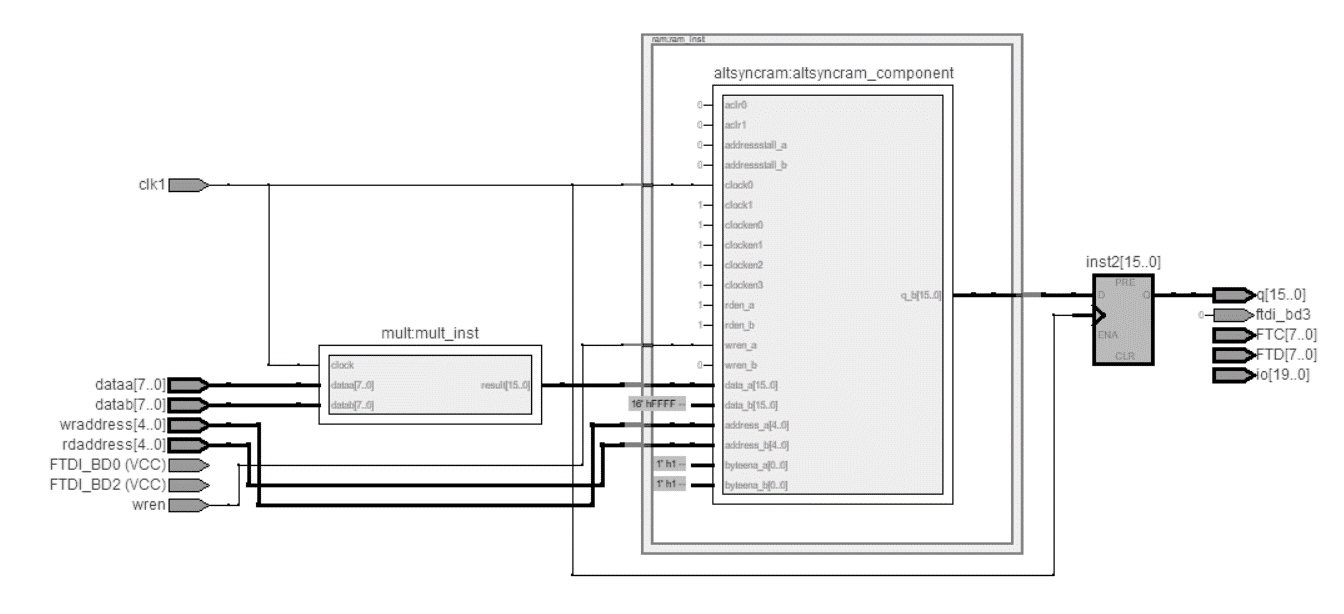


Рисунок 10. Отображение логической реализации проекта с демонстрацией содержимого блока памяти

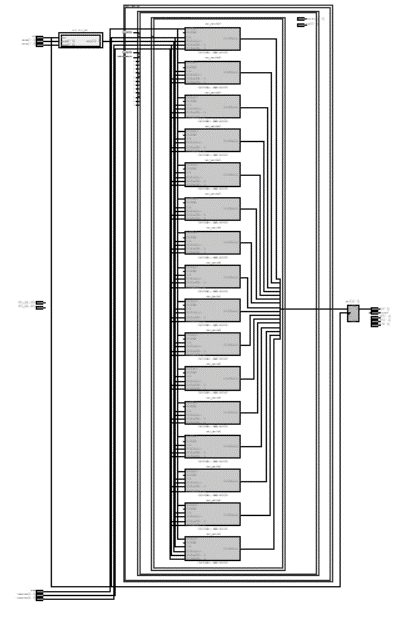


Рисунок 11. Нижний уровень иерархии для блока памяти

На рис. 10-11 изображена детализация реализаций блока памяти. На нижнем уровне иерархии данного блока видно, что он состоит из 16 функциональных однобитных элементов RAM размера 32.

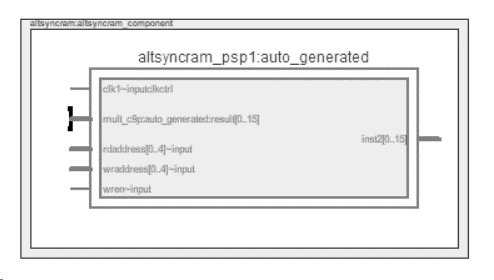


Рисунок 12. Фактическая реализация блока памяти

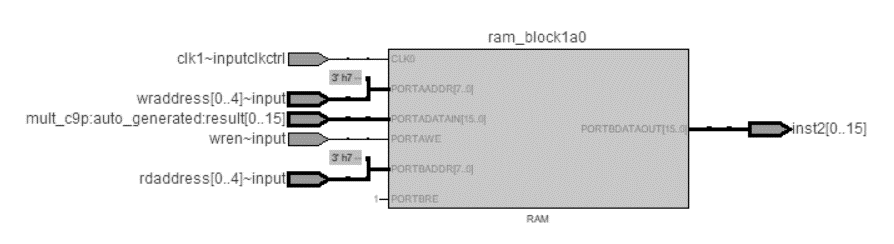


Рисунок 13. Нижний уровень иерархии фактической реализации блока памяти

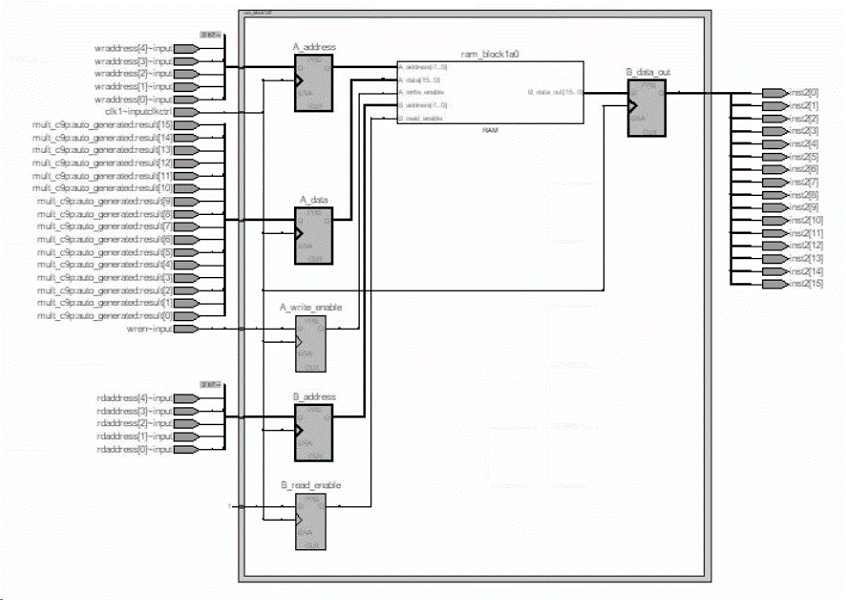


Рисунок 14. Подробный анализ способа реализации блока памяти

На рис. 12-14 представлены детали фактической реализации блока памяти, полученные с помощью утилиты Altera Quartus II Technology Map Viewer. Из рисунков выше видно, что в отличие от отображаемых в RTL Viewer 16 однобитных функциональных блоков памяти, фактически используемый ресурс микросхемы ‑ единственный блок встроенной памяти. В процессе компиляции выходные регистры были помещены в блок памяти, чтобы улучшить его производительность и снизить количество используемых логических ресурсов в проекте. Такой способ оптимизации называется упаковкой регистров, и трассировщик использует его для уменьшения логических ресурсов микросхемы.

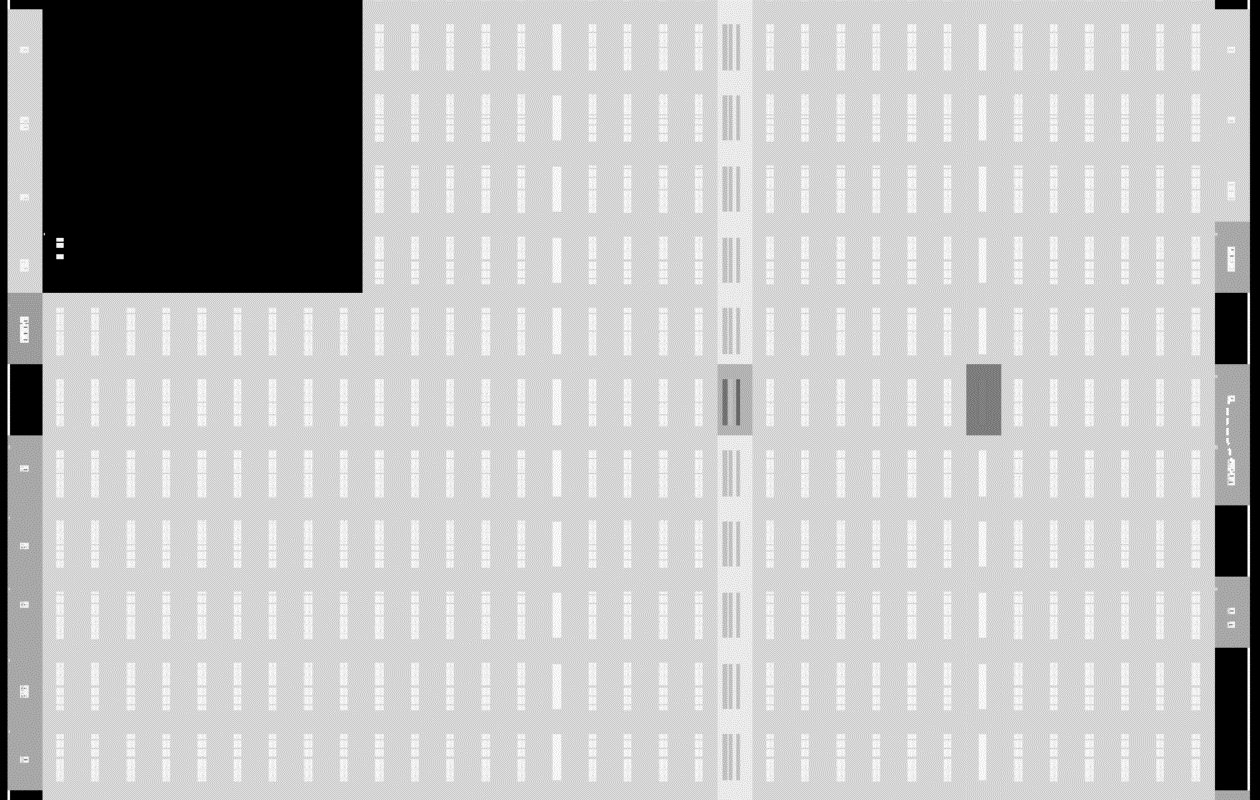


Рисунок 15. Физическое расположение конвейерного умножителя на кристалле

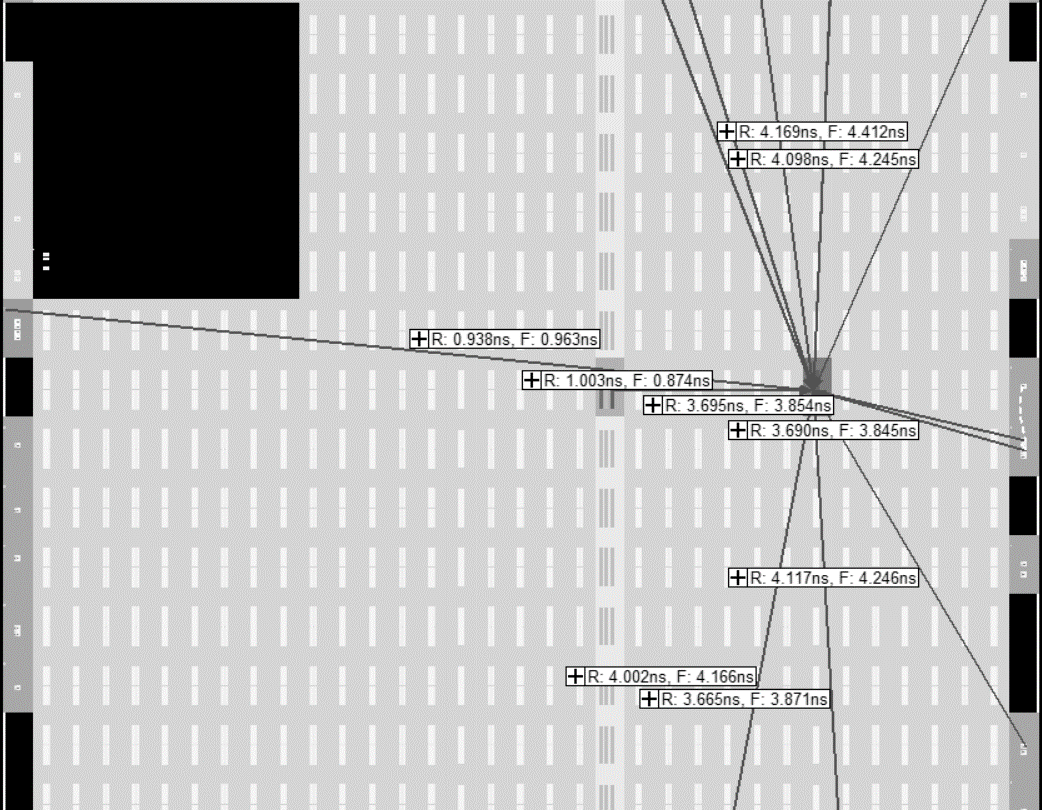


Рисунок 16. Входные связи для блока памяти

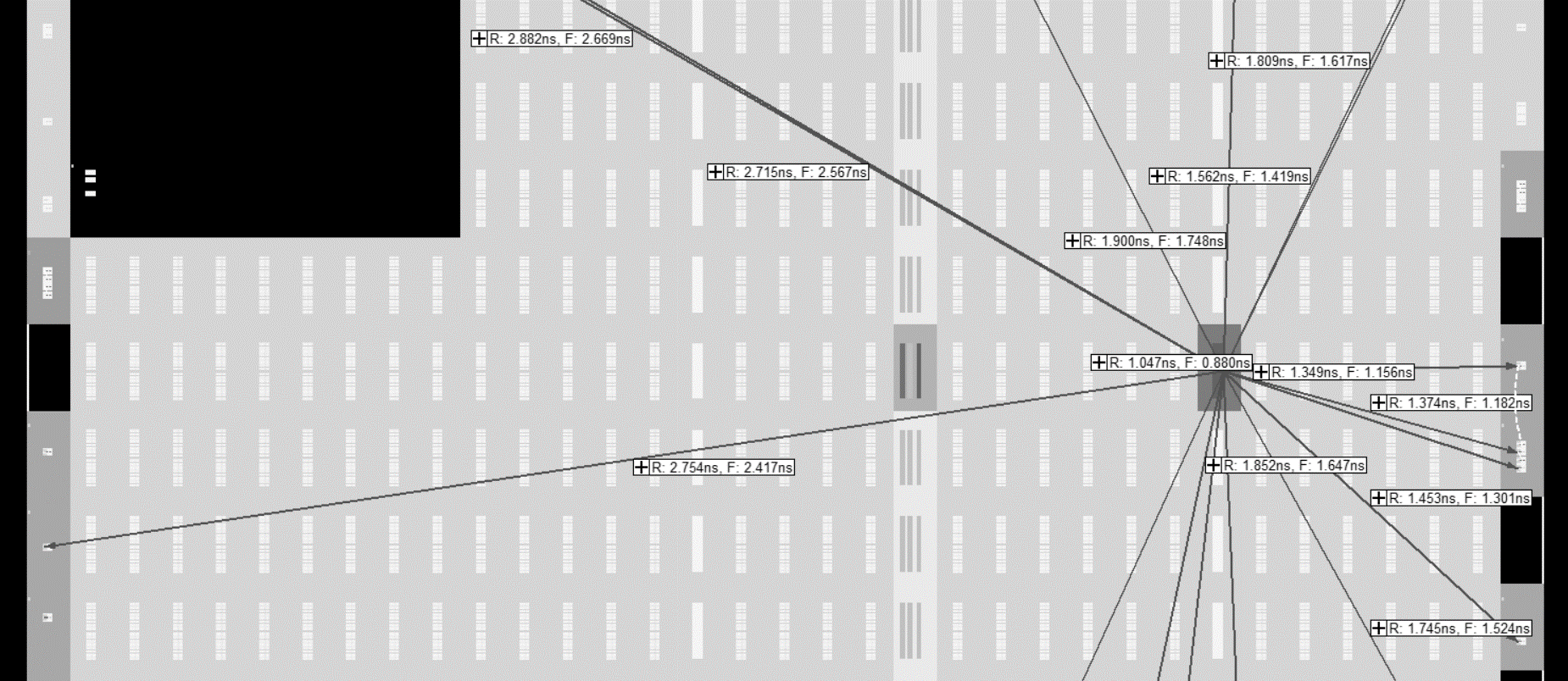


Рисунок 17. Выходные связи для блока памяти

На рис. 15-17 изображено физическое расположение разработанной системы на кристалле, а также входные и выходные связи. Изображения получены с помощью Chip Planner.

# Выводы

В ходе выполнения данной работы производилась работа в ПО Altera Quartus II. В результате были изучены: схема конвейерного умножителя, создание мегафункций с помощью утилиты MegaWizard® Plug-in Manager (умножитель 8х8, 32x16 RAM), создание НЕХ файлов с помощью редактора Memory Editor, анализ логической реализации проекта с помощью RTL Viewer, проверка физической реализации блоков с помощью Technology Map Viewer.