1. **Задание**

Разработать собственное устройство с использованием мегафункций и блоков памяти. Создать проект и реализовать заданное устройство. Выполнить моделирование. Синтезировать устройство с разными настройками по оптимизации ресурсов или скорости, сравнить полученные результаты. Необходимо выполнить все этапы разработки устройства, выполненные в работах 2.1 и 2.2. Выполнить программирование отладочной платы (прототипирование) и продемонстрировать работу устройства.

Реализовать умножитель, входные данные A и B, 6 бит. По разрешающему сигналу (нажатие кнопки) данные со входов умножаются между собой и сохраняются в регистре, после чего подаются на выход схемы. Предусмотреть сброс данных в накопителе (нажатие кнопки).

1. **Выполнение работы**
   1. С помощью утилиты MegaWizard Plug-in Manager создадим умножитель 6\*6 и 32\*12 RAM для хранения результатов. Реализуем счетчик и входной модуль (приложения 1 и 2). Соберем итоговую схему и назначим входные и выходные пины (рис. 1).

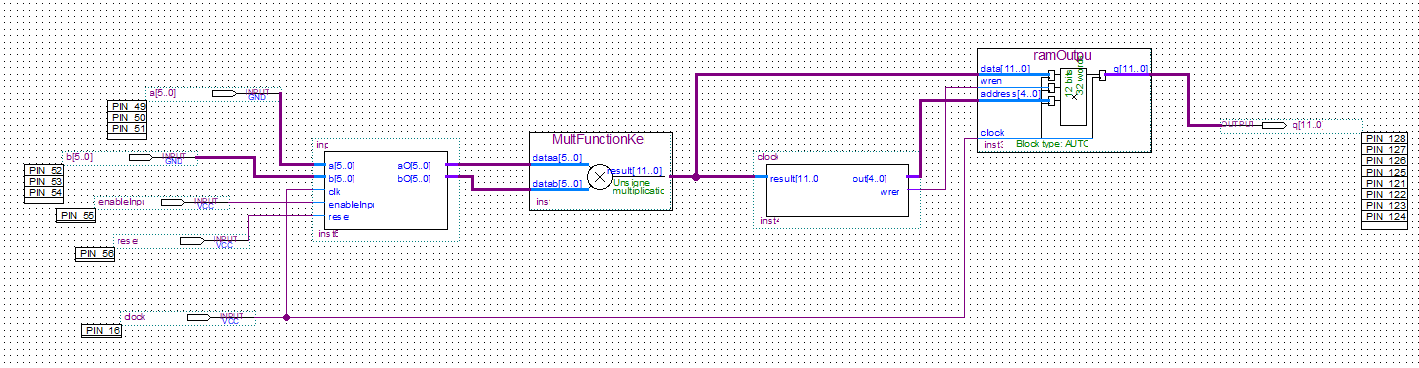


Рисунок 1. Итоговая схема проекта и назначения входных и выходных пинов

* 1. Выполним прототипирование на плате, для этого воспользуемся удаленным сервером с ПЛИС. Подадим для умножения значения 111100 и 111010, после чего подадим разрешающий сигнал. Результатом должно являться число 1101 1001 1000, следовательно, схема работает правильно (рис. 2).

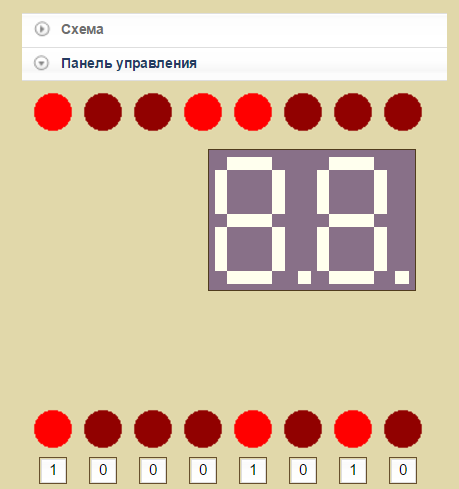


Рисунок 2. Умножения чисел 111100 и 111010. В результате представлены последние 8 разрядов

* 1. Выполним сброс и перемножим числа 111000 и 111000. Результатом должно являться число 1100 0100 0000, что соответствует действительности (рис. 3)



Рисунок 3. Умножения чисел 111000 и 111000. В результате представлены последние 8 разрядов.

* 1. Выполним команду Start Analysis & Elaboration для проверки правильности подключений файлов.
  2. Выполним подготовку к моделированию и само моделирование. Результаты представлены на рисунке 4.

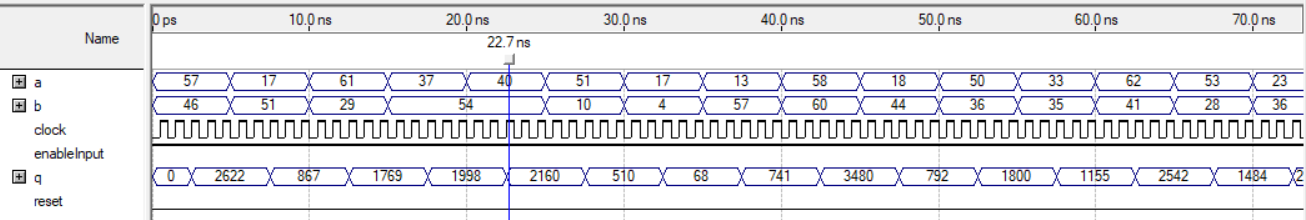


Рисунок 4. Результаты моделирования проекта

* 1. Проанализируем отчет о компиляции проекта. Для этого скомпилируем проект и откроем Compilation Report, затем выпишем полученные значения из вкладки Flow Summary окна Compilation Report в таблицу 1. Остальные данные приведены на рисунках 5 и 6.

Таблица 1.

**Flow Summary окна Compilation Report**

|  |  |
| --- | --- |
| **Параметр** | **Значение** |
| total logic elements | 18/12480 |
| total memory bits | 384 / 419328 |
| embedded multiplier 9-bit elements | 1 / 96 |
| total pins | 27 / 367 |

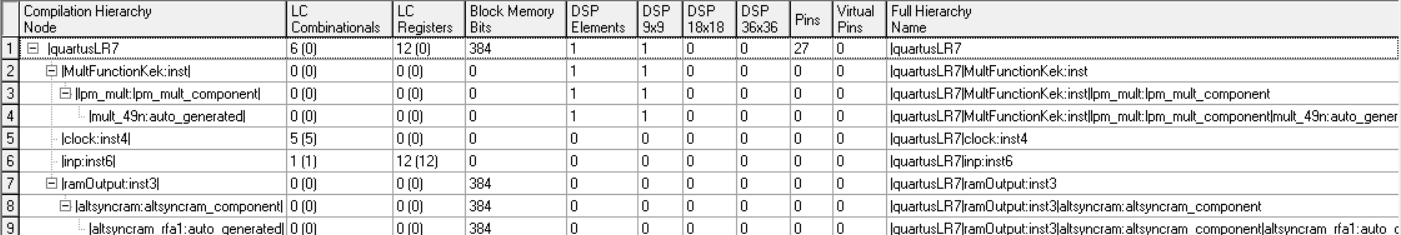


Рисунок 5. Таблица Resource Utilization by Entity об используемых ресурсах

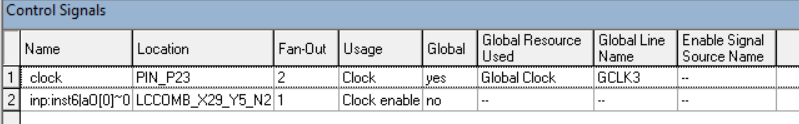
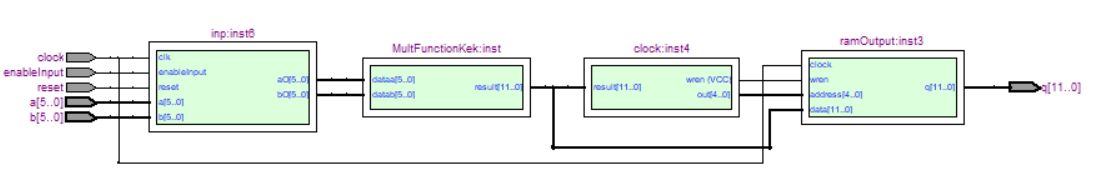
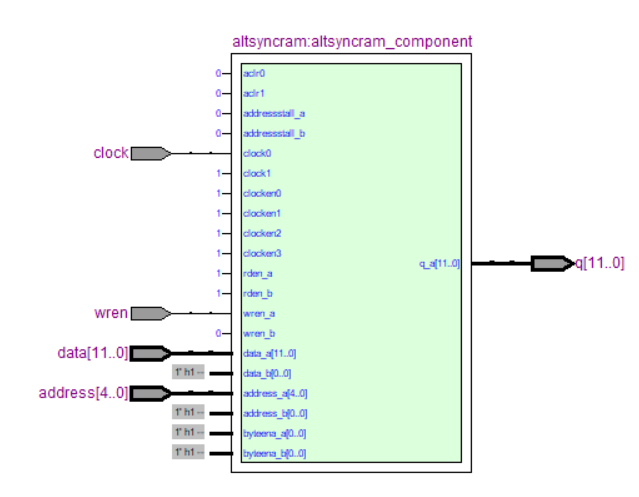
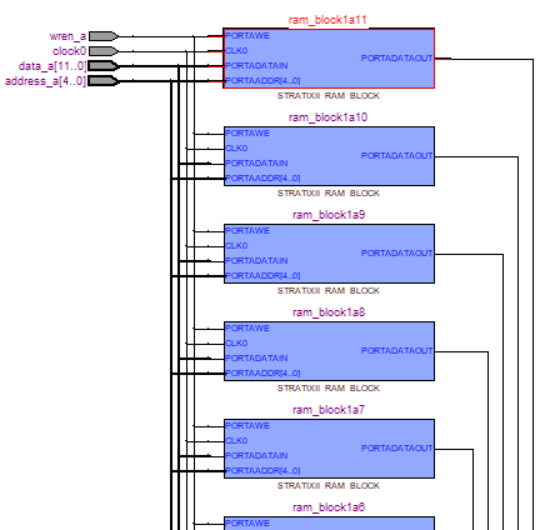


Рисунок 6. Информация об управляющих сигналах из раздела Control Signals

* 1. Проанализируем логическую реализацию проекта с помощью утилиты RTL Viewer. Результаты приведены на рисунках 7-9.







* 1. Проверим физическую реализацию блока RAM с помощью Technology Map Viewer (рис 10). В отличие от отображаемых в RTL Viewer нескольких однобитных функциональных блоков памяти, здесь отображается фактически используемый ресурс микросхемы – единственный блок встроенной памяти.

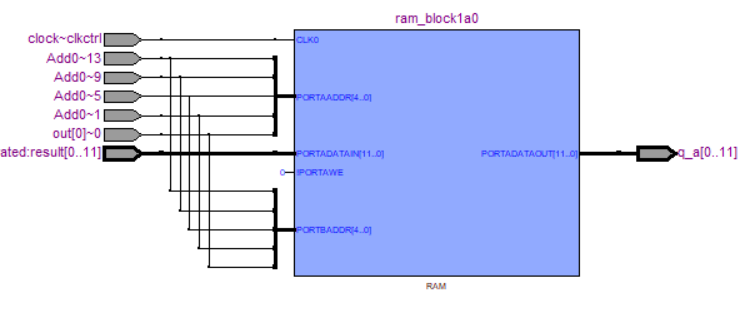


Рисунок 10. Скриншот фактической реализации блока оперативной памяти

* 1. Проверим связи блока ram с помощью редактора топологии кристалла Chip Planner. Выберем Generate Fan-In Connections и Generate Fan-Out Connections. Результаты приведены на рисунке 8.

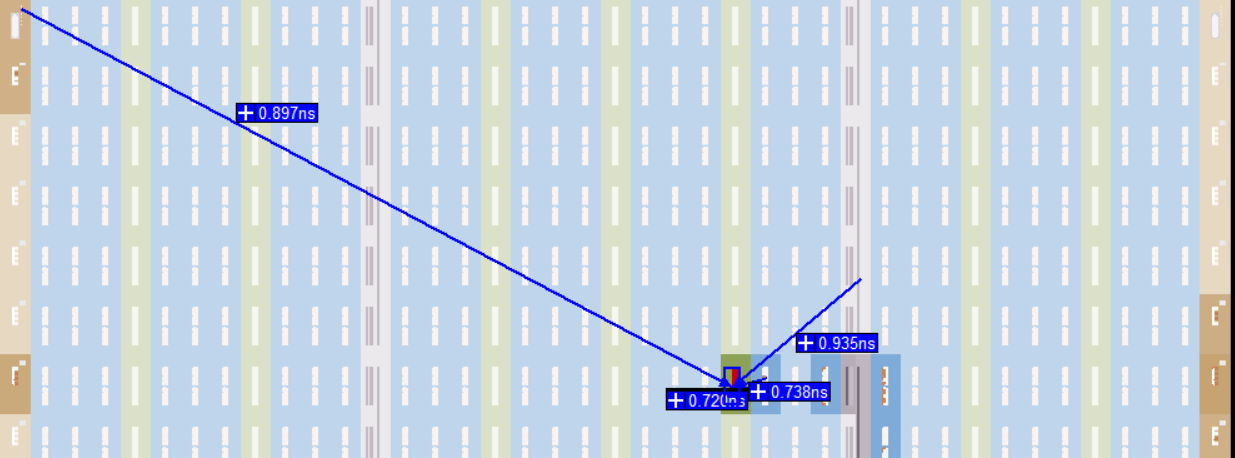


Рисунок 11. Входные сигналы в блок оперативной памяти в окне Chip Planner

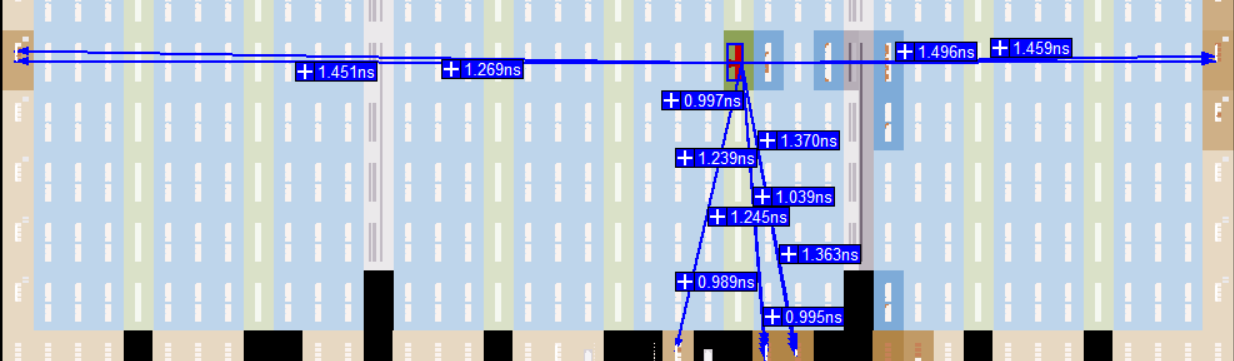


Рисунок 12. Входные сигналы из блока оперативной памяти в окне Chip Planner

* 1. Перейдем к выполнению второй части лабораторной работы. Создадим новую версию проекта для сохранения новых параметров (рис. 13).

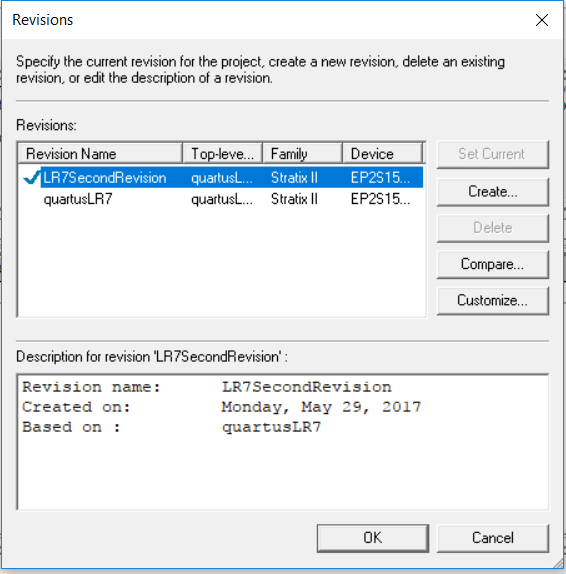


Рисунок 13. Создание новой версии проекта

* 1. С помощью редактора назначений реализуем умножитель на логических элементах (рис. 2).

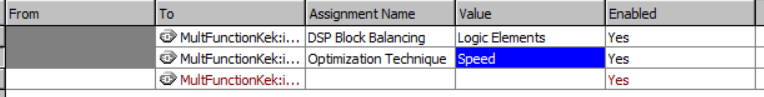


Рисунок 14. Назначения умножителя на логических элементах

* 1. Выполним повторную компиляцию проекта. Повторно проанализируем информацию об используемых ресурсах (табл. 1, рис. 3, рис. 4).

Таблица 2.

**Flow Summary окна Compilation Report**

|  |  |
| --- | --- |
| **Параметр** | **Значение** |
| total logic elements | 40/12480 |
| total memory bits | 384 / 419328 |
| embedded multiplier 9-bit elements | 0 / 96 |
| total pins | 27 / 367 |

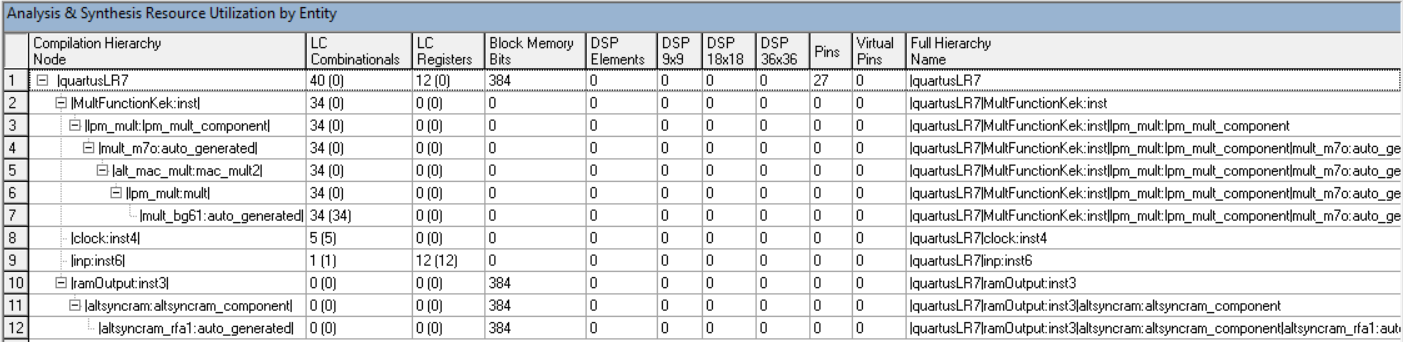


Рисунок 15. Таблица Resource Utilization by Entity об используемых ресурсах

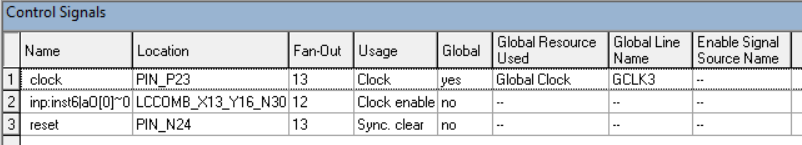


Рисунок 16. Информация об управляющих сигналах из раздела Control Signals

Сравним две ревизии проекта с помощью кнопки Compare. Можно сделать вывод, что при данной реализации модуля умножителя, используются только логические элементы, но не встроенный модуль умножителя.

* 1. С помощью Pin Planner выполним назначение контактов ввода-вывода к различным банкам и установим значения напряжения в 1.8В и 2.5В.

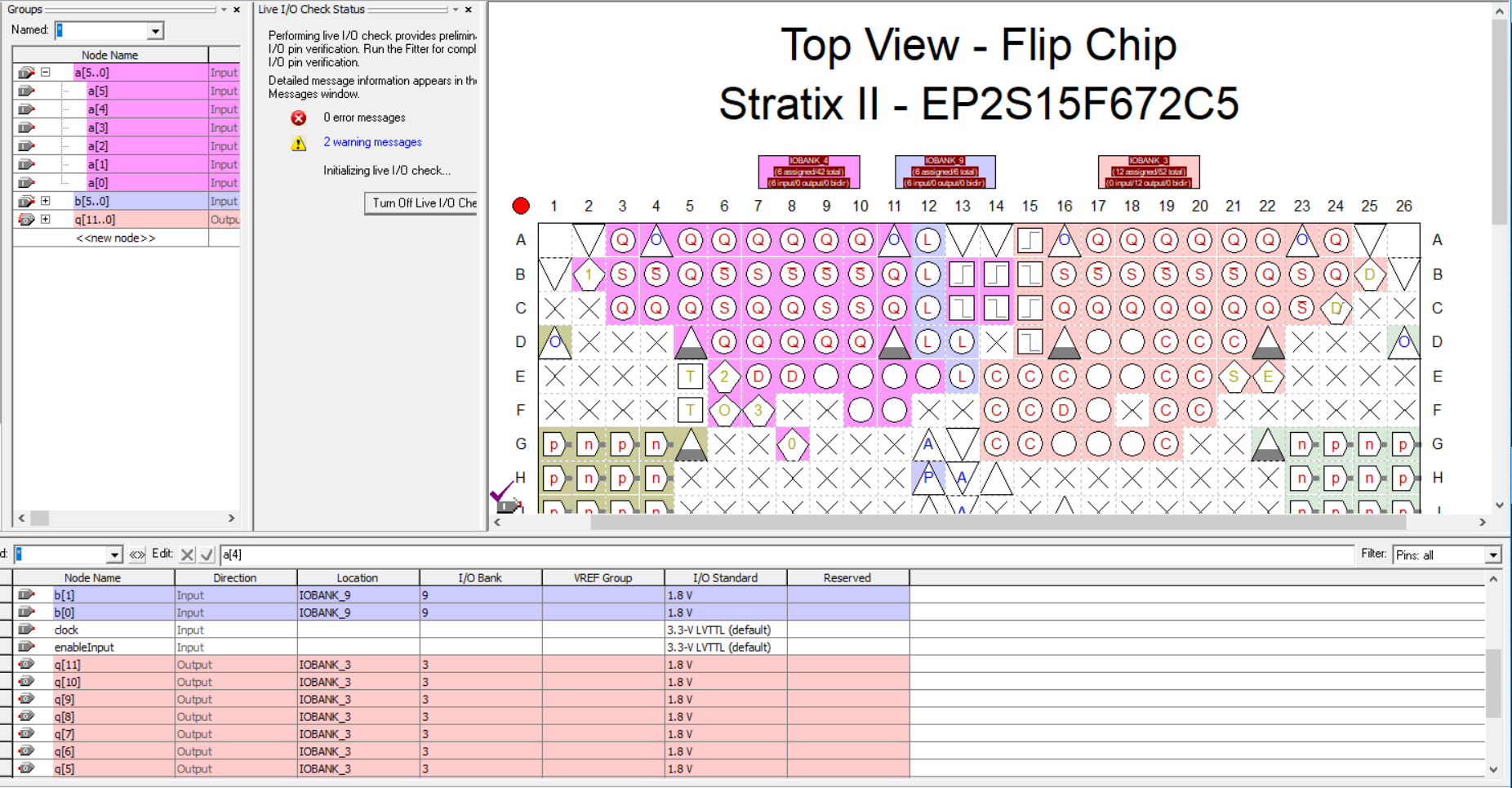


Рисунок 17. Результаты назначения контактов ввода-вывода

* 1. Сохраним назначенные контакты ввода-вывода с помощью утилиты Back Annotate.
  2. Выполним перенос настроек из текущей версии проекта в предыдущую. Для этого воспользуемся экспортом назначений текущей версии проекта в файл формата .CSV, а затем импортом значений из этого файла в исходную версию проекта. Проверим правильность переноса значений.

1. **Выводы**

В результате выполнения данной лабораторной работы был изучен и реализован конвейерный умножитель, изучена работа с утилитами MegaWizard Plug-in Manager, RTL Viewer, Chip Planner, а также разобраны некоторые вкладки отчета о компиляции проекта. Кроме того, были изучены следующие возможности: создание нескольких версий одного проекта, выбор варианта реализации устройств – не только с помощью встроенных модулей, но и логических элементов, сравнение различных версий проекта, назначение контактов ввода-вывода, изменение напряжения, подаваемого по умолчанию, перенос назначений контактов в другую версию проекта.