**[Архитектура ПЛИС (FPGA)](https://marsohod.org/11-blog/265-fpga)**

Создано 20 Январь 2014

Автор: nckm

Просмотров: 49169

[FPGA](https://marsohod.org/index.php/ourblog/11-blog/265-fpga) – это сокращение от английского словосочетания Field Programmable Gate Array.

[ПЛИС](https://marsohod.org/index.php/ourblog/11-blog/265-fpga) – это сокращение от словосочетания «Программируемая Логическая Интегральная Схема». Слово ПЛИС встречается в русскоязычных документациях и описаниях вместо слова FPGA. Далее по тексту в основном будет использоваться этот термин - ПЛИС.  
   
ПЛИС и FPGA – это аббревиатуры, обозначающие один и тот же класс электронных компонентов, микросхем. Это микросхемы, применяемые для создания собственной структуры цифровых интегральных схем.

Логика работы ПЛИС определяется не на фабрике изготовителем микросхемы, а путем дополнительного программирования (в полевых условиях, field-programmable) с помощью специальных средств: программаторов и программного обеспечения.

Микросхемы ПЛИС – это не микропроцессоры, в которых пользовательская программа выполняется последовательно, команда за командой. В ПЛИС реализуется именно электронная схема, состоящая из логики и триггеров.

Проект для ПЛИС может быть разработан, например, в виде принципиальной схемы. Еще существуют специальные языки описания аппаратуры типа [Verilog](https://marsohod.org/index.php/verilog) или VHDL.

В любом случае, и графическое и текстовое описание проекта реализует цифровую электронную схему, которая в конечном счете будет «встроена» в ПЛИС.

Обычно, сама микросхема ПЛИС состоит из:

* конфигурируемых логических блоков, реализующих требуемую логическую функцию;
* программируемых электронных связей между конфигурируемыми логическими блоками;
* программируемых блоков ввода/вывода, обеспечивающих связь внешнего вывода микросхемы с внутренней логикой.

Строго говоря это не полный список. В современных ПЛИС часто бывают встроены дополнительно блоки памяти, блоки DSP или умножители, PLL и другие компоненты. Здесь, в этой статье я их рассматривать не буду.

Разработчик проекта для ПЛИС обычно абстрагируется от внутреннего устройства конкретной микросхемы. Он просто описывает желаемую логику работы «своей» будещей микросхемы в виде схемы или текста на Verilog/ VHDL. Компилятор, зная внутреннее устройство ПЛИС сам пытается разместить требуемую схему по имеющимся конфигурируемым логическим блокам и пытается соединить эти блоки с помощью имеющихся программируемых электронных связей. В общем случае размещение и трассировка связей между логическими блоками в ПЛИС остается за компилятором.

**Классификация ПЛИС по типу хранения конфигурации.**

SRAM-Based.  
Это одна из самых распространенных разновидностей ПЛИС. Конфигурация ПЛИС хранится ячейках статической памяти, изготовленной по стандартной технологии CMOS.   
Достоинство этой технологии – возможность многократного перепрограммирования ПЛИС. Недостатки – не самое высокое быстродействие, после включения питания прошивку нужно вновь загружать. Значит на плате должен еще стоять загрузчик, специальная микросхема FLASH или микроконтроллер – все это удорожает конечное изделие.

Flash-based.  
В таких микросхемах хранение конфигурации происходит во внутренней FLASH памяти или памяти типа EEPROM. Такие ПЛИС лучше тем, что при выключении питания прошивка не пропадает. После подачи питания микросхема опять готова к работе. Однако, у этого типа ПЛИС есть и свои недостатки. Реализация FLASH памяти внутри CMOS микросхемы – это не очень просто. Требуется совместить два разных техпроцесса для производства таких микросхем. Значит они получаются дороже. Кроме того, такие микросхемы, как правило, имеют ограниченное количество циклов перезаписи конфигурации.

Antifuse.  
Специальная технология по которой выполняются однократно программируемые ПЛИС. Программирование такой ПЛИС заключается в расплавлении в нужных местах чипа специальных перемычек для образования нужной схемы.  
Недостаток – собственно программировать / прошивать чип можно только один раз. После этого исправить уже ничего нельзя. Сам процесс прошивки довольно не быстрый. Зато есть масса достоинств у таких ПЛИС: они довольно быстрые (могут работать на больших частотах), меньше подвержены сбоям при радиации – все из-за того, что конфигурация получается в виде перемычек, а не в виде дополнительной логики, как у SRAM-based.

**Конфигурируемые логические блоки.**

В документации компании Альтера встречается выражение Logic Array Block (LAB) – массив логики. У компании Xilinx в микросхемах ПЛИС есть примерно такие же блоки - Configurable Logic Block (CLB). Конфигурируемый логический блок – это базовый элемент в ПЛИС, в нем может быть выполнена какая-то простая логическая функция или реализовано хранение результата вычисления в регистрах (триггерах).

Сложность и структура конфигурируемого логического блока (CLB) определяется производителем.

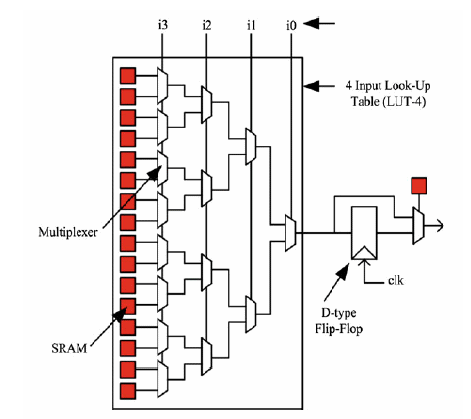
Теоретически, конфигурируемый логический блок может быть, например, очень простым, просто как отдельный транзистор. Или он может быть очень сложным, как целый процессор. Это крайние точки реализации.

В первом случае потребуется огромное число программируемых связей, чтобы потом из отдельных транзисторов собрать требуемую схему. Во втором случае связей может нужно и не так много, но теряется гибкость проектирования пользовательской схемы.

Именно поэтому конфигурируемый блок обычно представляет из себя что-то среднее: он обычно достаточно сложен, чтобы можно было бы зашить туда некоторую функцию, но и довольно мал, чтобы разместить множество таких блоков внутри ПЛИС и чтобы была возможность связать их в единую схему.

Таким образом, выбор структуры конфигурируемого логического блока производителем ПЛИС – это всегда поиск компромиса по площади кристалла, по быстродействию, энергопотреблению и так далее.

Конфигурируемый логический блок может состоять из одного или нескольких базовых логических элементов. В англоязычной литературе это Basic Logic Element (BLE) или просто Logic Element  (LE). В ПЛИС обычно используются так называемые LUT-based базовые логические элементы. Что-то вроде этого:

  
Рис. 1. Пример традиционного базового логического элемента.

LUT – это Look-Up Table, таблица преобразования. Например, на Рис.1 показан четырехбитный LUT в составе базового логического блока. Здесь четырехбитному числу на входе логической функции ставится в соответствие однобитный результат. Красные квадратики на Рис. 1 обозначают программируемый элемент, регистр – это та память, где хранится прошивка для ПЛИС. Видно, что для конфигурации 4-х битного LUT требуется 16 конфигурационных регистра. Содержимое этих регистров определяют логическую функцию, реализованную внутри  базового логического элемента.   
  
Еще один конфигурационный регистр (на Рис. 1 это одиночный красный квадратик справа) определяет нужно ли на выход базового логического элемента выдавать прямо значение с LUT или нужно выдать зафиксированное в D-триггере значение с LUT. Фиксация и хранение данных в цифровых схемах нужна практически в любом проекте.

Примерно такой логический элемент использовался в моем [экспериментальном проекте «ПЛИС внутри ПЛИС»](https://marsohod.org/projects/plata1/166-minicpld).

Рассматривая Рис. 1 как пример традиционного базового логического элемента понимаешь какая избыточность заложена внутрь современного кристалла ПЛИС (SRAM-based). Ведь в самом деле, конфигурационные регистры (красные квадратики) прямо не доступны для использования в цифровом проекте. Они только служат для формирования пользовательской функции. Для одного D-триггера в пользовательском проекте требуется более 16 (иногда много больше) триггеров для хранения конфигурации ПЛИС.

На самом деле базовый логический элемент в разных ПЛИС оказывается гораздо сложнее, чем показано на Рис. 1. Ниже есть некоторые примеры из документации на разные типы ПЛИС.

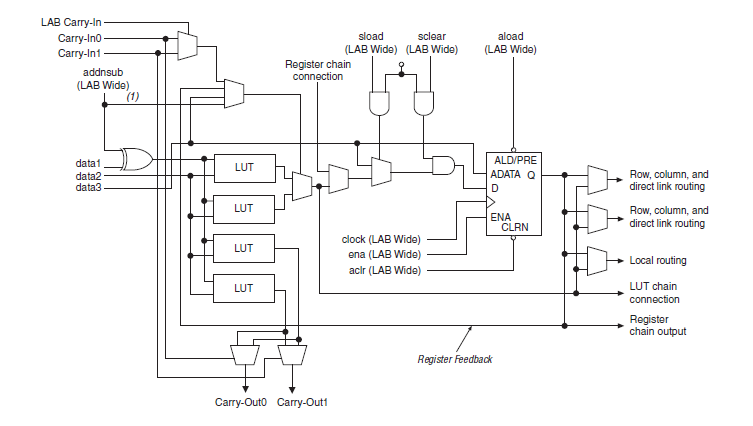


Рис. 2. Базовый логический элемент CPLD MAX II компании Альтера.

Здесь хорошо видны LUT и D-Триггер хранения результата. Ниже, на Рис. 3 представлен базовый элемент Cyclone III.

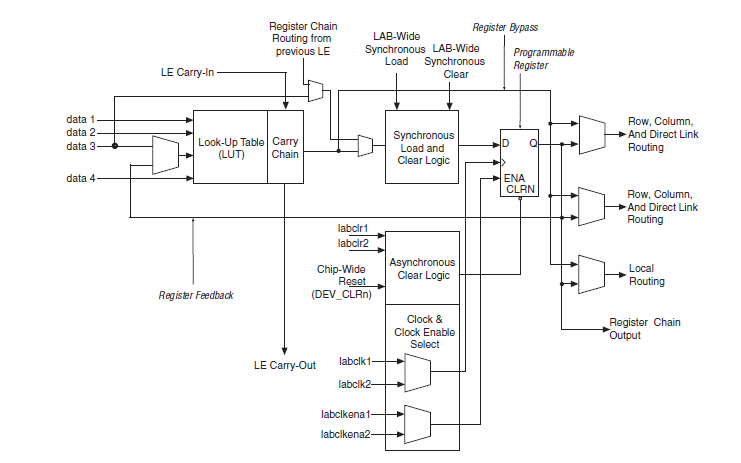


Рис. 3. Базовый логический элемент FPGA Cyclone III компании Альтера.

В микросхемах Альтеры в одном LAB может содержаться 10-16 LE.

В микросхемах компании Xilinx Virtex-6 базовый логический элемент – это так называемый Slice. В одном CLB всего два Slice. Зато один Slice – это довольно сложное устройство:

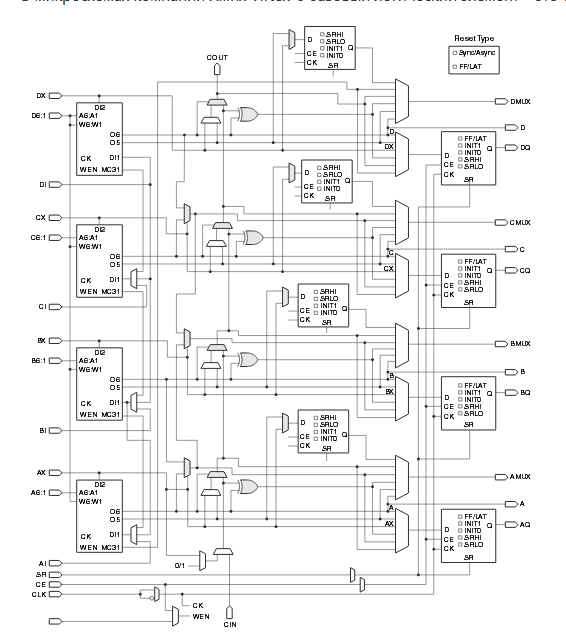


Рис. 4. Базовый элемент Xilinx Virtex-6 Slice.

В одном CLB Virtex-6 имеется 8 LUT и 16 D-Триггеров и еще кое-что плюс к этому. Вот так все сложно.

Другая крайность – микросхемы FPGA компании Microsemi (бывшая Actel).  
Например, в микросхемах серии 40MX базовый логический элемент выглядит вот так:

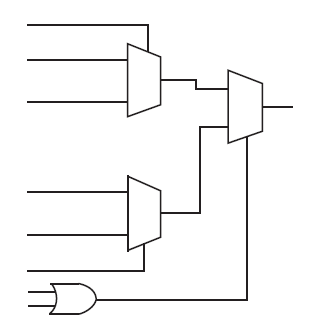


Рис. 5. Logic Module of Microsemi 40MX serie.

Восемь входов и один выход.  
Здесь нет ни Look-Up Table, ни даже D-Триггера. Триггера, как и остальная логика, формируются где нужно из вот таких крошечных кирпичиков – Logic Module.

Почему у разных компаний получилась такая большая разница в реализации базового логического элемента? Видимо в микросхемах Microsemi связь между базовыми блоками обходится гораздо дешевле: серия 40MX является однократно программируемой. В ней межблочные связи «проплавляются» между соединяющими дорожками и позже не могут быть изменены. Нет никаких регистров для временного хранения прошивки. Здесь нет программируемых переключателей, мультиплексоров, как в FPGA других типов. Ну микросхемы компании Microsemi - это несколько особый случай. Это технология называется antifuse – для производства таких микросхем используется модифицированный техпроцесс CMOS с дополнительными слоями для организации межблочных связей.

**Программируемые связи между логическими блоками.**

Чтобы в ПЛИС заработала нужная нам цифровая схема мало того, что нужно сконфигурировать имеющиеся логические блоки особым образом, еще нужно создать, запрограммировать связи между логическими блоками.

Для этого в ПЛИС имеются специальные конфигурируемые коммутаторы.

В англоязычной документации встречаются следующие термины: FPGA Routing Architecture и Programmable Routing Interconnect. Это все об этом, о программируемых связях между логическими блоками.

Известно две основных методики построения ПЛИС по типу архитектуры связей: островная и иерархическая.

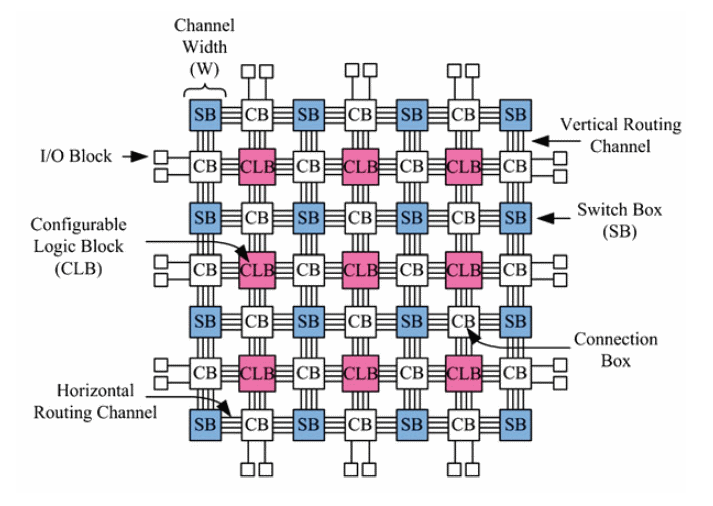


Рис. 6. Островная ПЛИС.

Островная ПЛИС называется так потому, что конфигурируемые блоки все равны между собой и находятся, как острова в океане, между узлами коммутации и линиями связи.

Здесь, на Рис. 6 обозначаются CB – Connection Box и SB – Switch Box. В сущности это программируемые мультиплексоры, подключающие тот или иной CLB к другому CLB через цепочки проводов в ПЛИС.

Это island-style FPGA или mesh-based FPGA. Типичный пример таких микросхем – это серии Altera Cyclone и Stratix.

Второй известный тип ПЛИС – это иерархические ПЛИС. Здесь идет расчет на то, что в схеме всегда есть участки которые взаимодействуют друг с другом более тесно, чем с отдаленными модулями проекта.

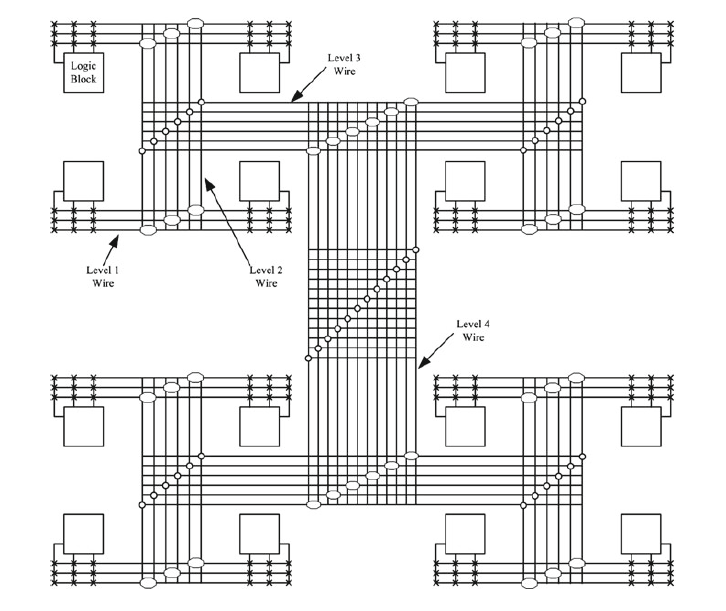


Рис. 7. Иерархическая ПЛИС.

Здесь близлежащие CLB соединить довольно просто, нужно не много коммутаторов и получающиеся связи работают быстро. Вот если нужен более крупный блок вычислителей, то сигнал должен выйти на более высокий уровень иерархии и потом зайти вглубь в соседнюю «комнату».

Нельзя сказать, что это существенно хуже, чем island-style. Просто каждый метод имеет свои плюсы и минусы.

Типичные представители иерархических ПЛИС – это микросхемы компании Альтера серии Flex10K, APEX.

**Программное обеспечение для проектирования ПЛИС.**

Программное обеспечение для проектирования ПЛИС, а именно компилятор (синтезатор логики и фиттер и ассемблер) – это, возможно, самая сложная часть всей ПЛИС технологии.

Компилятор должен проанализировать пользовательский проект (схемы и текстовые описания на Verilog HDL или VHDL ) и сгенерировать нетлист (netlist) – список всех элементов схемы и связи между ними. Netlist должен быть оптимизирован – логические функции нужно минимизировать, возможные дублированные регистры нужно удалить.

Затем компилятор должен вместить всю логику из netlist в имеющуюся архитектуру ПЛИС. Это делает фиттер (fitter). Он размещает логические элементы и выполняет трассировку связей между ними (процесс place and route). Сложность состоит в том, что один и тот же проект может быть размещен в ПЛИС разными способами и этих способов миллионы. Некоторое размещение и трассировка оказываются лучше, другие хуже. Главный критерий качества полученной системы – максимальная частота, на которой сможет работать проект при данном размещении элементов и при данной трассировке связей. Здесь оказывает влияние длина связей между логическими блоками и количество программируемых коммутаторов между ними.

Компилятор, зная архитектуру ПЛИС по результатам работы дополнительно выдает отчет о времени прохождении сигналов от регистра до регистра. Эта информация часто бывает полезной для разработчика высокопроизводительных систем. Разработчик для ПЛИС имеет возможность давать некоторые советы компилятору где, в каком месте кристалла лучше разместить тот или иной модуль проекта.

Выбирая для своего проекта, для своей платы конкретную микросхемы ПЛИС разработчик в некоторой мере попадает в зависимость от производителя этой ПЛИС, так как должен в работе пользоваться программным обеспечением от этого же производителя.

Программное обеспечение компании Альтера: [Quartus II](https://marsohod.org/index.php/aquartus2).  
ПО Xilinx для проектирования для ПЛИС: ISE Suite, Vivaldo Design Suite.  
ПО компании Microsemi: Libero IDE, Libero SoC.

Возможно, программное обеспечение, компиляторы для ПЛИС – это важнейшая составляющая интеллектуальной собственности компаний производителей ПЛИС.