|  |  |
| --- | --- |
| ДИСЦИЛИНА | **Схемотехника устройств компьютерных систем Часть 2** |
|  |  |
| ИНСТИТУТ | **ИТ** |
| КАФЕДРА | **вычислительной техники** |
|  |  |
| ВИД УЧЕБНОГО | **Лекция** |
| МАТЕРИАЛА |  |
| ПРЕПОДАВАТЕЛЬ | **Тарасов И.Е.** |
|  |  |
| СЕМЕСТР | 3 |
|  |  |

3. Программируемые логические интегральные схемы

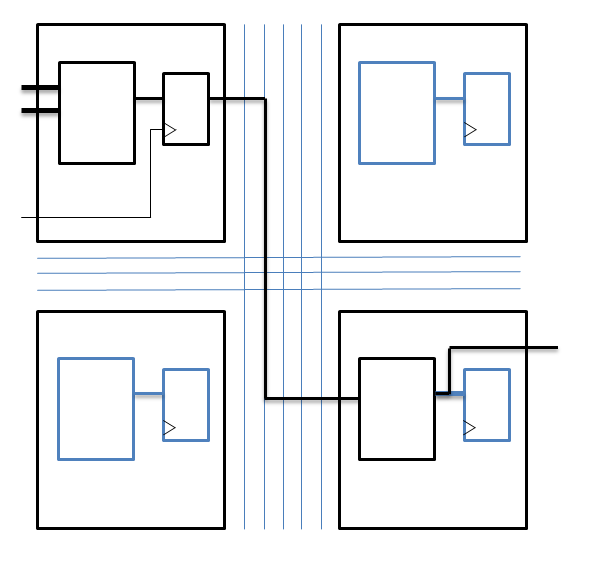
3.1. Основные сведения

Программируемые логические интегральные схемы (ПЛИС) – отдельный подкласс микросхем, представляющих собой набор цифровых компонентов, соединения между которыми могут программироваться уже после ее изготовления. Это позволяет создавать в ПЛИС практически произвольные цифровые схемы, что делает ее удобной для создания макетов будущих микросхем с целью их проверки. Кроме того, для небольших партий изделий, где не подходят универсальные процессоры, оригинальная схема, реализованная в ПЛИС, может оказаться эффективной с точки зрения производительности и цены.

Общим термином для ПЛИС является PLD – Programmable Logic Device. Этот термин отражает саму возможность программирования соединений между компонентами микросхемы. В зависимости от типа компонентов и соединений ПЛИС могут иметь различные архитектуры. Например, одной из первых является SPLD – Simple Programmable Logic Device. Ее развитием стала CPLD – Complex Programmable Logic Devices, которая до сих пор применяется. Более массовая и известная архитектура – FPGA (Field-Programmable Gate Array). Этот термин нуждается в пояснении. Прежде всего, gate array – это, по сути, «микросхема» (дословно – «массив вентилей»). Понятие field-programmable дословно переводится как «программируемый в поле». Термин in-the-field, от которого образовано field programmable, обозначает устройства, которые программируются не на предприятии-изготовителе, а пользователем (т.е. «в поле», за пределами завода).

Архитектура FPGA подразумевает размещение матрицы логических ячеек, между которыми проложены соединительные линии («трассировочные»). Параметры самих ячеек могут настраиваться, подключение к трассировочным линиям также управляется. Переключение трассировочных линий и настройка режима работы ячеек производится путем загрузки в FPGA специального конфигурационного файла. Обычно формат этого файла закрыт производителем во избежание восстановления схемы по имеющемуся или прочитанному из FPGA файлу.

Схема взаимодействия логических ячеек в ПЛИС с архитектурой FPGA показана на рис. 3.1.

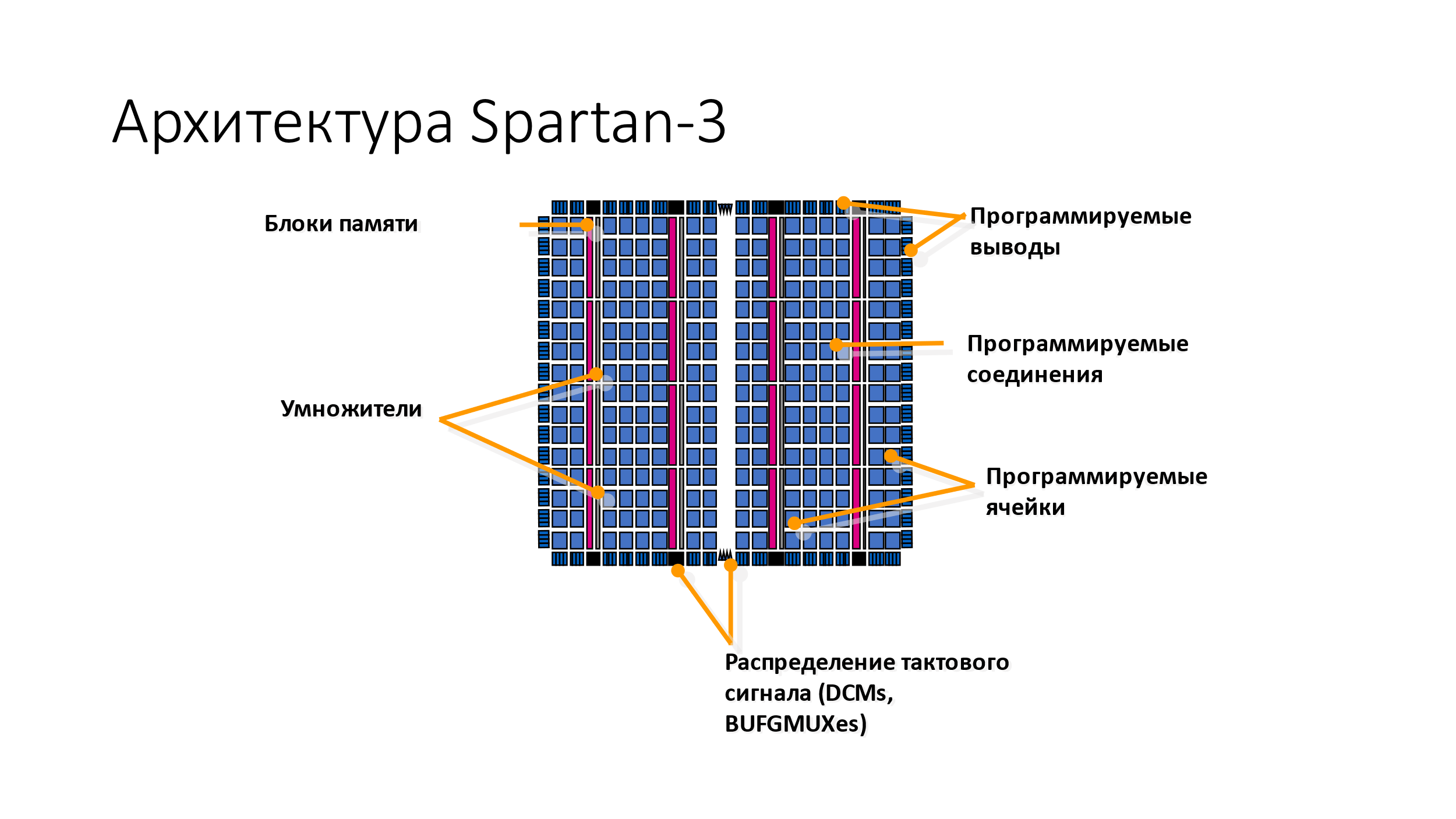


*Рисунок 3.1 Схема взаимодействия логических ячеек в ПЛИС с архитектурой FPGA*

На рис. 3.1 видно, что выход одной из ячеек (слева сверху) подключен к одной из вертикальных трассировочных линий и идет вниз, подключаясь к входу правой нижней ячейки. Во второй ячейке не используется триггер, ее выходом является выход блока комбинационной логики.

Управление ячейками и трассировочными линиями крайне редко производится вручную. Создание файла с конфигурационными данными для загрузки в FPGA выполняется в САПР в автоматическом режиме.

Логические ячейки и трассировочные линии – не единственные компоненты FPGA. Пример простой ПЛИС серии Spartan-3 производства Xilinx показан на рис. 3.2.



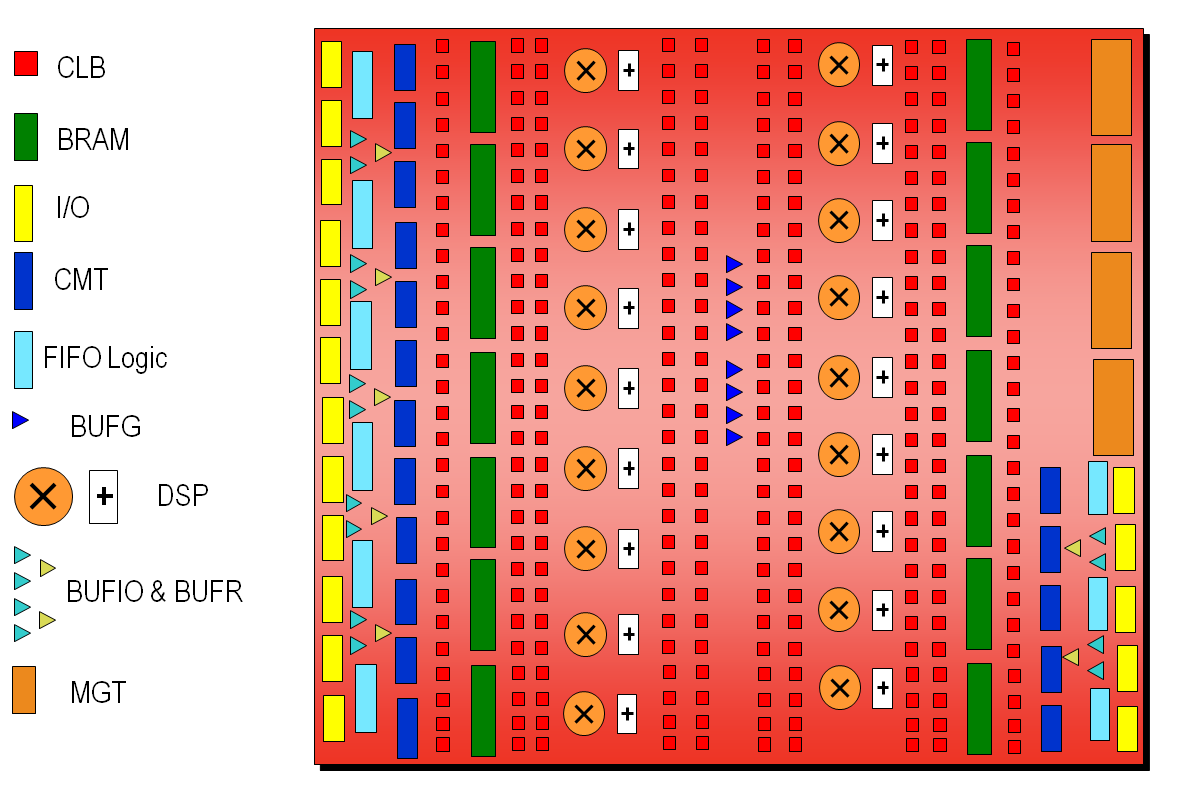
*Рисунок 3.2 Архитектура ПЛИС Xilinx с архитектурой FPGA серии Spartan-3*

Важнейшим дополнением к матрице ячеек и трассировочных линий являются внешние выводы. Они также являются настраиваемыми (программируемыми), поскольку заранее неясно, будет ли конкретный вывод входом или выходом. Кроме направления передачи данных, у выводов FPGA можно также настраивать реализуемые электрические интерфейсы. Не все выводы FPGA являются программируемыми, часть выводов используется для подключения питания, загрузки конфигурации, а в более сложных FPGA существуют специальные выводы с фиксированными функциями.

На рис. 3.2 также видны примеры дополнительных компонентов, которые часто являются основой для построения эффективных схем в ПЛИС. В Spartan-3 это блоки памяти и блоки умножения независимых операндов. Такие блоки можно было бы реализовать и путем соединения логических ячеек, но в этом случае возможности ячеек будут использованы нерационально. Память, реализованная в виде компактного блока, займет существенно меньше места, чем память такого же объема, реализованная в виде настроенных и соединенных логических ячеек. Аналогично, умножение вида a\*b можно сделать и путем соединения логических ячеек, но готовый аппаратный модуль займет меньше места и будет работать быстрее. К моменту появления серии Spartan-3 стало понятно, что память и блоки умножения требуются во многих проектах, и если добавить к FPGA аппаратные (непрограммируемые) блоки, производительность и возможности существенно вырастут.

Отдельно показаны компоненты для формирования тактовых сигналов и их распределения по кристаллу FPGA. Эти компоненты относятся не к «ускоряющим работу», а к обязательным для применения.

На рис. 3.3 показана архитектура ПЛИС Xilinx серии Kintex-7. Сравнение с рис. 3.2 показывает, что дальнейшее развитие архитектуры FPGA привело к появлению большего количества аппаратных блоков и увеличению их разнообразия.

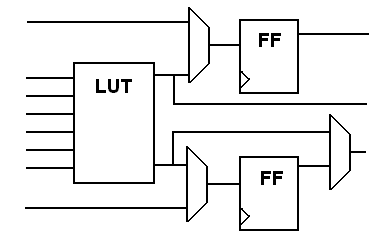


*Рисунок 3.3 Архитектура ПЛИС серии Xilinx Kintex-7*

Новым по сравнению со Spartan-3 является компонент MGT (Multi-Gigabit Transceiver). Это блок, который способен работать с высокоскоростными («мультигигабитными») сигналами, которые передаются по дифференциальным парам. Примерами интерфейсов, которым требуются MGT, являются PCI Express, Serial ATA (SATA), 10G Ethernet и множество интерфейсов со скоростями передачи данных 1-10 Гбит/с и более (современные интерфейсы достигают скоростей 116 Гбит/с). Такой сигнал невозможно обработать на базе цифровых компонентов, а тем более путем соединения логических ячеек, которые заведомо работают медленнее из-за более длинных и сложных трассировочных линий. Поэтому блок, который может полностью выполнить прием и передачу сигнала для высокоскоростной линии, помещается на кристалл ПЛИС и подключается к трассировочным линиям с помощью параллельных интерфейсов. Блоки MGT появились в FPGA в период возникновения потребности в магистральном сетевом оборудовании. Например, магистральный сетевой коммутатор может оперировать десятками оптоволоконных линий, для которых необходимо произвести маршрутизацию пакетов. ПЛИС большого логического объема с блоками MGT широко используются для разработки сетевого оборудования, которое требуется в относительно небольших количествах (поэтому разработка специальной микросхемы экономически не оправдана).

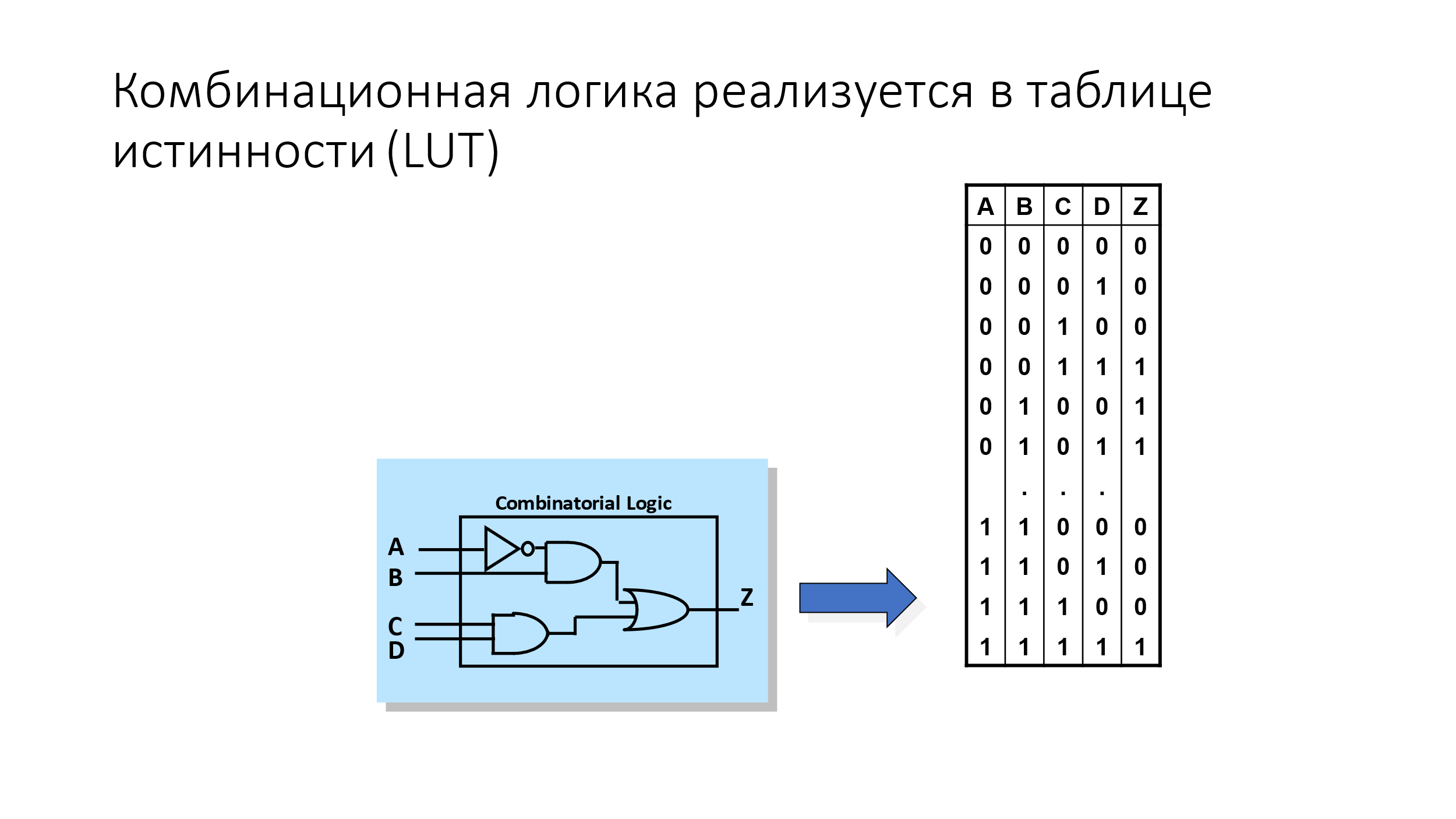
3.2 Архитектура логической ячейки

Понятие «логическая ячейка» (logic cell) для FPGA подразумевает комбинацию из логического генератора и триггера. Пример структурной схемы логической ячейки FPGA (соответствующей Kintex-7 и более поздним) показан на рис. 3.4.



*Рисунок 3.4 Структурная схема логической ячейки FPGA*

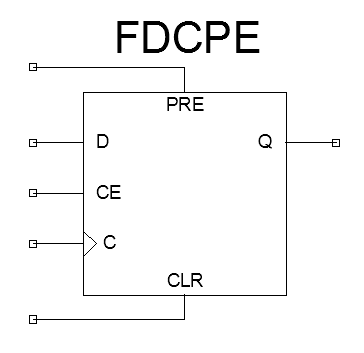
Генератор произвольных логических функций обозначен на рисунке как LUT (Look-Up Table, «таблица истинности»). В действительности он не содержит внутри логические элементы, а представляет собой обычную память. Входы блока являются адресными линиями этой памяти, а выход – это выход данных. Если составить желаемую таблицу истинности для такого блока и записать ее в память, то при работе FPGA подача комбинации входных сигналов будет выбирать соответствующую ячейку памяти. Можно представлять, что внутри блока размещена соответствующая схема из логических элементов, однако это просто память с N входами, в которую записаны нужные реакции на каждую входную комбинацию, как показано на рис. 3.5.



*Рисунок 3.5 Принцип реализации комбинационных схем на базе таблицы истинности*

Количество входов в LUT существенно определяет, схемы какой сложности можно реализовать. Долгое время промышленным стандартом была 4-входовая LUT. Во многих случаях схему выражают в количестве «эквивалентных логических ячеек», под которыми понимают набор из 4-входовой LUT и триггера. На рис. 3.4 можно видеть, что современные FPGA используют 6-входовые LUT и два триггера.

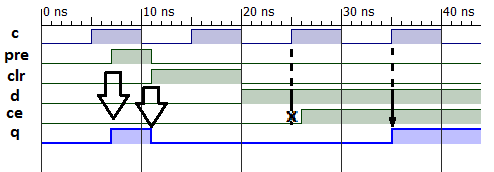
Вторым компонентом ячейки является триггер. Его условное графическое обозначение показано на рис. 3.6, а временные диаграммы, иллюстрирующие работу – на рис. 3.7. Показанное условное графическое изображение отражает представление САПР



*Рисунок 3.6 Условное графическое изображение триггера в составе ячейки FPGA*

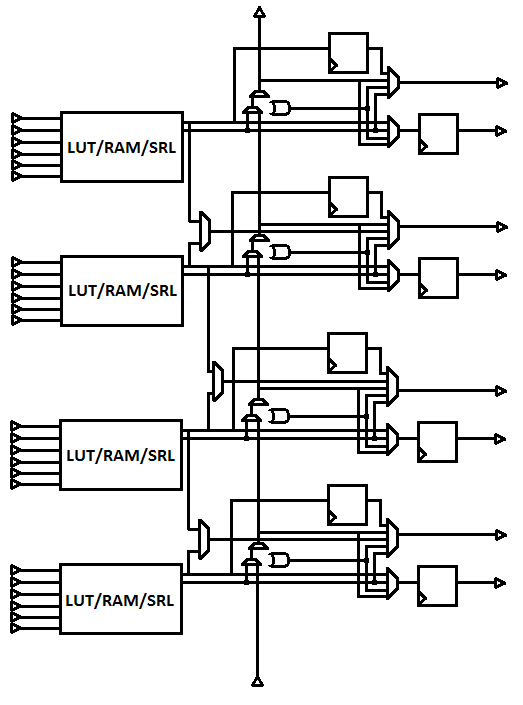
На рис. 3.7 показаны основные сценарии работы триггера. Входы CLR (очистка) и PRE (предустановка) являются асинхронными, т.е. появление сигнала логической единицы на них приводит к немедленному изменению состояния триггера. Сигнал CLR приводит к сбросу (переходу выхода в 0), а PRE – к переходу выхода в 1. Современные FPGA обычно позволяют физически реализовать только один из этих сигналов (кроме того, асинхронный сброс или установка триггера настоятельно не рекомендуется).

Если ни один из сигналов CLR, PRE неактивен, то триггер записывает состояние входа данных D в момент появления фронта тактового сигнала. Однако дополнительный сигнал CE (Clock Enable) позволяет разрешать или запрещать реакцию триггера на тактовый сигнал. На рис. 3.7 видно, что в момент времени 25 нс на входе C был фронт тактового сигнала, однако в этот момент на входе CE был 0, поэтому триггер не отреагировал на фронт и не изменил свое состояние. В момент времени 35 нс сигнал CE был в состоянии 1 («счет разрешен»), поэтому выход триггера Q записал уровень 1, который был в этот момент на входе D.



*Рисунок 3.7 Временные диаграммы работы триггера*

Логические ячейки объединяют в FPGA в более крупные структуры – секции (slice) и конфигурируемые логические блоки (Configurable Logic Block, CLB). На рис. 3.8 показана структура логической секции FPGA Xilinx серии 7. В секции размещены 4 логические ячейки и несколько дополнительных компонентов. Например, каждая пара логических генераторов объединена мультиплексором, а два мультиплексора объединены третьим. Это упрощает реализацию логических функций с большим количеством входов. Другим компонентом, который относится к секции, а не какой-то отдельной ячейке, является цепь ускоренного переноса (fast carry chain). На рисунке она видна как вертикальная цепочка вентилей. Назначением этой цепочки является передача между разрядами бита переноса при реализации операций сложения и вычитания. Ее роль подобна аппаратным блокам. В принципе, можно реализовать блок вычисления переноса и на базе LUT, но это приведет к нерациональному использованию ресурсов ПЛИС. В то же время вычисление бита переноса занимает немного места, а требуется достаточно часто. Поэтому такие цепочки добавлены в секции FPGA и используются при построении сумматоров и вычитателей по мере необходимости.

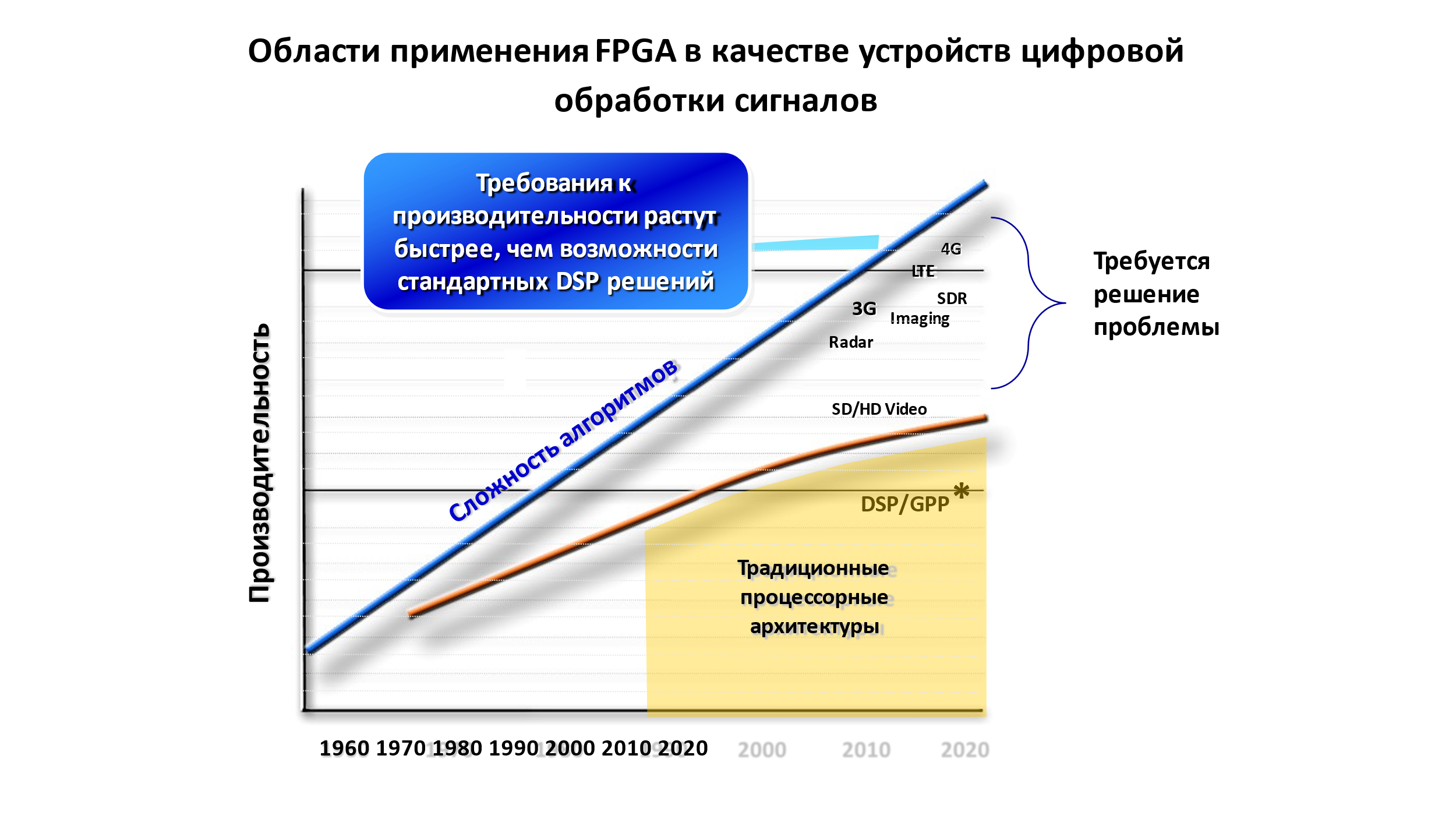


*Рисунок 3.8 Структура логической секции ПЛИС FPGA (Xilinx, серия 7 и последующие)*

3.3 Дополнительные компоненты FPGA

Рассмотренные выше дополнительные компоненты FPGA активно используются в тех проектах, где FPGA являются именно серийной компонентной базой, а не макетом будущего изделия. Обычно ПЛИС по сравнению с микроконтроллерами или процессорами проигрывает по ряду характеристик. Если сравнить универсальную микросхему (ПЛИС) и специализированный процессор, изготовленный по сопоставимому технологическому процессу, будет видно, что ПЛИС оказывается медленнее и дороже. Это является следствием того, что схема в ПЛИС создается на базе конфигурируемых логических ячеек, которые работают в определенном режиме, а значит, используются не полностью. Частота работы схемы в ПЛИС меньше, потому что соединения между ячейками выполняются с помощью конфигурируемых трассировочных линий, а не металлических проводников, проводимых строго между точками, которые необходимо соединить.

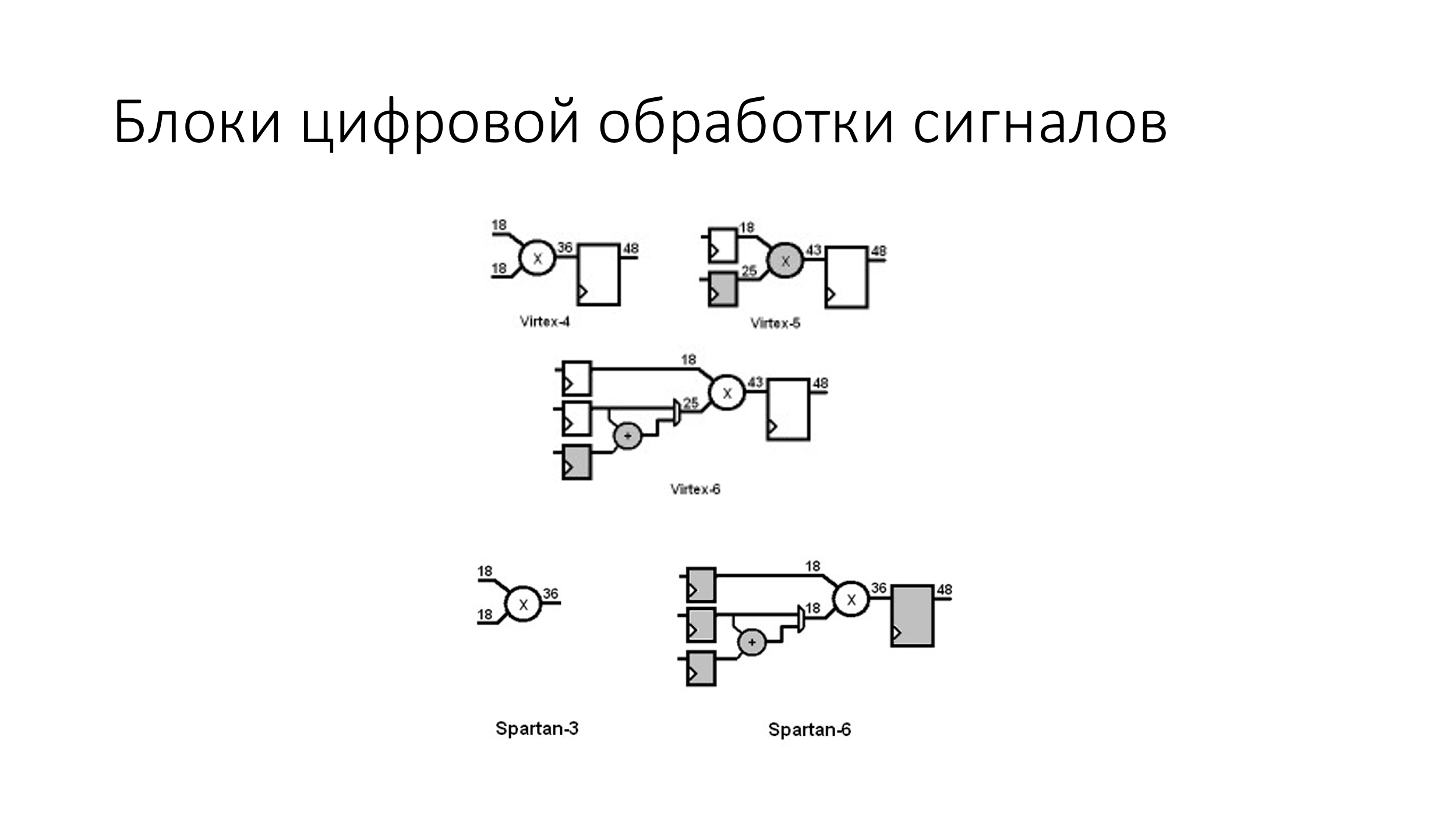
Вместе с тем, возможность создания схемы с нестандартной архитектурой является сильной стороной ПЛИС. На рис. 3.8 показана иллюстрация к проблеме создания систем цифровой обработки сигналов. Такие технологии, как беспроводные сети новых поколений (4G, 5G), обработка изображений высокого разрешения, видеопотоков и ряд других, требуют все возрастающей производительности специального подкласса вычислений, которые обозначаются как «цифровая обработка сигналов» (ЦОС). Это достаточно обширный класс алгоритмов, однако с точки зрения вычислительной техники в качестве ключевой характеристики указывают операцию умножения с накоплением (Multiply and Accumulate, MAC). Эта операция вида s = s + k\*x требует блока умножения, который сложнее в реализации, чем остальные базовые арифметические операции. Обычно разместить множество блоков умножения в процессоре общего назначения оказывается нерационально, потому что нет гарантии, что все программы будут их эффективно использовать. На практике производители микроконтроллеров воздерживаются от поддержки параллельного выполнения множества операций умножения с накоплением, потому что нет гарантии, что эти ресурсы существенно повысят производительность программ.



*Рисунок 3.8 Иллюстрация к областям применения ПЛИС в задачах цифровой обработки сигналов*

Вместе с этим для ПЛИС такая проблема стоит не так остро. Размещенные в ПЛИС блоки, аппаратно выполняющие умножение с накоплением, можно соединить требуемым образом с помощью настраиваемых трассировочных линий. Блоки могут быть распределены между несколькими схемами, объединены, отключены и т.д.

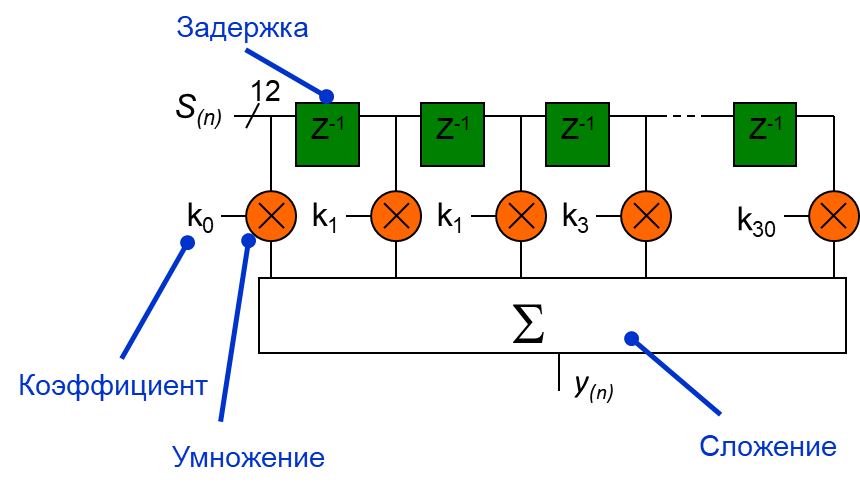
Эволюция блоков цифровой обработки сигналов показана на рис. 3.9.



*Рисунок 3.9 Блоки цифровой обработки сигналов в ПЛИС Xilinx*

В простых ПЛИС Spartan-3 блок цифровой обработки представляет собой просто аппаратный умножитель. Он может перемножить два 18-разрядных операнда, давая на выходе 36-разрядное произведение. Преимуществом такого блока является его компактность по сравнению с такой же схемой, реализованной на базе логических ячеек. Кроме того, ячейки, соединяемые конфигурируемыми трассировочными линиями, дают существенно большую задержку при вычислении произведения, поэтому умножитель, реализующий только одну операцию, оказался быстрее, меньше по площади и энергопотреблению. Однако практическое применение ПЛИС быстро показало, что операция умножения часто сопровождается операцией накопления (сложения с аккумулятором). Аккумулятор можно реализовать на базе ячеек, но он сразу же снижает тактовую частоту. Поэтому первым же шагом в развитии блока цифровой обработки сигналов стало добавление аппаратного аккумулятора, способного работать на такой же высокой частоте, что и умножитель. С учетом того, что сложение двух 36-разрядных чисел может дать 37 разрядов, для аккумулятора требуется разрядность больше, чем выход умножителя. Первое поколение блоков цифровой обработки сигналов в серии Virtex-4 имело 18-разрядные операнды для умножения и 48-разрядный аккумулятор для накопления суммы. Иногда используется запись вида 18\*18=48, которая показывает, что перемножаются два 18-разрядных операнда, а аккумулятор имеет 48 разрядов.

Одним из самых известных алгоритмов цифровой обработки сигналов является фильтр с конечной импульсной характеристикой (КИХ-фильтр, также FIR – Finite Impulse Response). Его структурная схема показана на рис. 3.10.



*Рисунок 3.10 Структурная схема цифрового фильтра с конечной импульсной характеристикой*

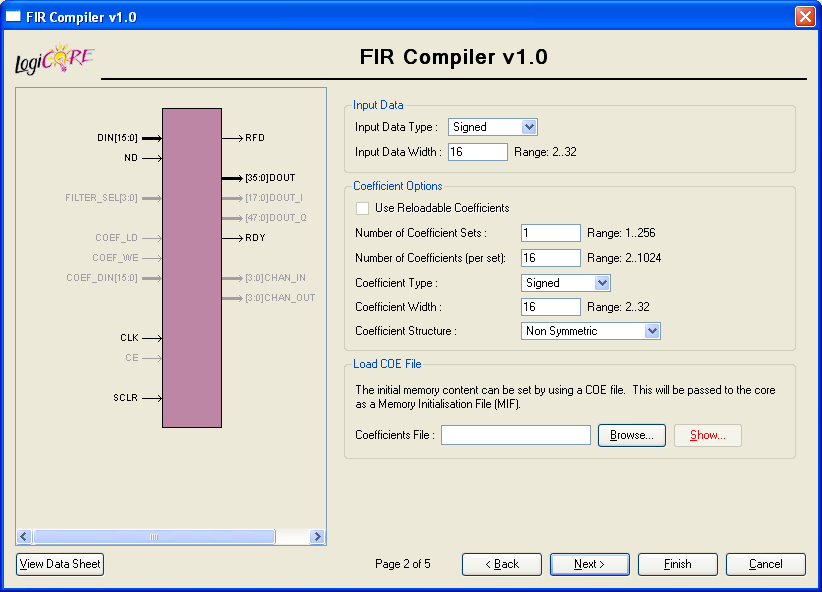
Фильтр реализует вычисление вида:

Величины xi – подаваемые на вход фильтра значения входного сигнала. Они также называются «отсчетами» (sample). Коэффициенты k являются постоянными и определяют поведение фильтра. Количество коэффициентов в фильтре называется *порядком фильтра* (filter order).

Увеличение порядка фильтра позволяет проводить более сложные преобразования с входным сигналом. Поэтому фильтры с большим количеством коэффициентов (а следовательно, и операций умножения) представляют практический интерес для обработки сигналов. При этом не слишком эффективно использовать для такой операции процессоры общего назначения. Для вычисления выходного значения фильтра потребуется выполнять операции умножения n последних отсчетов на соответствующие коэффициенты, т.е. организовать цикл на n итераций. Это пропорционально снизит производительность процессора, даже если одна итерация выполняется быстро.

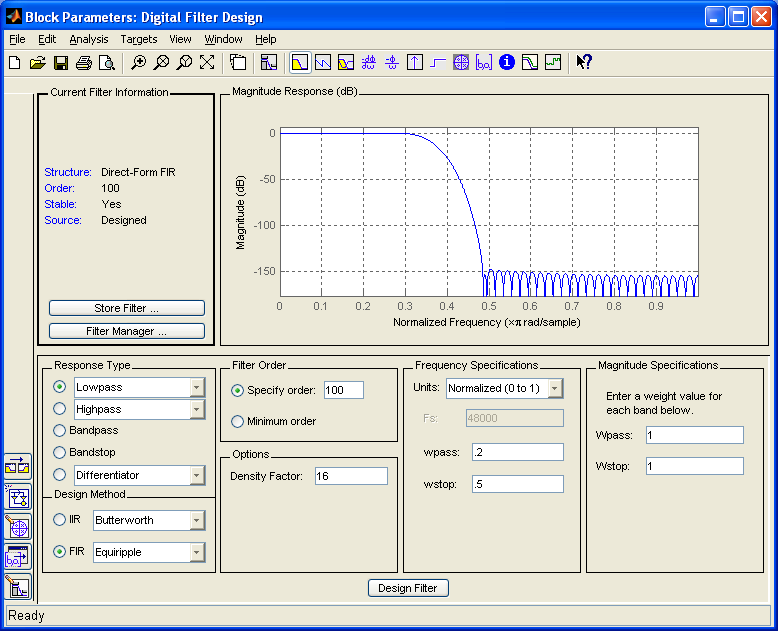
Поскольку все операции для вычисления выходного значения фильтра можно выполнять одновременно, это делает FPGA удобной аппаратной платформой для цифровой фильтрации. Если необходимо реализовать фильтр с n коэффициентами, можно использовать n аппаратных блоков, выполняющих умножение с накоплением. При этом тактовая частота такого фильтра остается высокой.

Цифровые фильтры являются широко распространенными вычислительными модулями, поэтому их разработка часто автоматизируется в САПР. На рис. 3.11 показан генератор IP-ядра КИХ-фильтра, который генерирует RTL-представление такого компонента, основываясь на таких входных параметрах, как порядок фильтры, разрядность коэффициентов и т.п.



*Рисунок 3.11 Генератор IP-ядер цифровых фильтров в составе САПР ПЛИС*

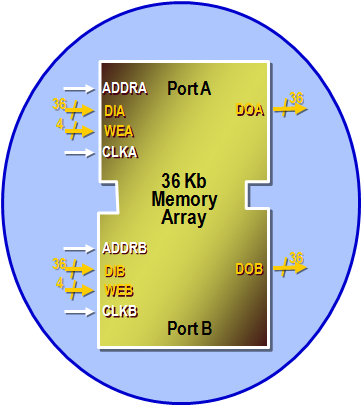
Определение коэффициентов фильтра – отдельное крупное направление в цифровое обработке сигналов. Кроме обширного теоретического материала, в этой области существует большое количество программного обеспечения, выполняющего расчеты коэффициентов фильтра на основе их желаемых характеристик. Пример такого инструмента показан на рис. 3.12.



*Рисунок 3.12 Создание цифрового фильтра в программном пакете Matlab*

Другим типом аппаратного блока, который существенно улучшает характеристики проекта в ПЛИС по сравнению с конфигурируемыми ячейками, является блочная память (Block RAM, BRAM). Несмотря на то, что в ячейках FPGA имеются триггеры, которые могут хранить данные, остальная площадь ячейки используется для других целей (исключением является специальный режим распределенной памяти, доступный для FPGA Xilinx). Если описать большой блок памяти с расчетом на его реализацию в ячейках, FPGA будет использована малоэффективно. Если бы внутри был блок обычной памяти, он бы занял существенно меньше места при том же объеме в битах. Именно с этой целью в FPGA добавляют блоки статической памяти, которые занимают существенно меньше места по сравнению с ячейками, которые могли бы хранить такой же объем данных.

Графическое представление блока памяти в FPGA Xilinx показано на рис. 3.13. Это не единственный возможный вариант блока памяти, однако именно в этом виде он реализуется в нескольких семействах FPGA. Другие производители используют похожие решения.

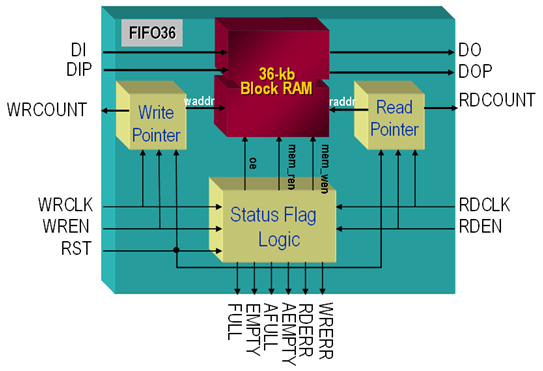


*Рисунок 3.13 Блок статической памяти в ПЛИС Xilinx*

Блок памяти реализует двупортовую статическую синхронную память. Термин «статическая» означает, что такая память сохраняет данные при наличии напряжения питания, и не требует специальных циклов обновления данных («регенерации»), как в динамической памяти. Термин «синхронная» отражает тот факт, что все операции с памятью происходят по фронту тактового сигнала. Альтернативой является асинхронный интерфейс, сигналы которого позволяют производить запись или чтение без привязки к специальным моментам времени. Однако синхронный интерфейс соответствует тенденциям цифровой электроники, рассмотренным в гл. 2, где синхронные схемы являются предпочтительными из-за большей предсказуемости временных характеристик.

Под термином «двупортовая» (dual-port) подразумевается наличие двух независимых наборов данных (портов), которые позволяют проводить с блоком памяти две операции одновременно. Двупортовую память также подразделяют на true dual-port («истинно двупортовую») и simple dual-port (также pseudo dual-port, «псевдо двупортовая»). Отличием является то, что в «псевдо двупортовой памяти» только один порт является универсальным, а второй может производить только чтение данных. Для «истинно двупортовой» памяти таких ограничений нет, и оба интерфейса могут использоваться для чтения или записи в любых сочетаниях. Если производится одновременная запись в одну и ту же ячейку памяти, результат неопределен. Такие ситуации должны отслеживаться на уровне схемы или общей архитектуры системы.

В некоторых случаях вместе с блоками памяти реализуют и дополнительные схемы, облегчающие построение часто используемых цифровых модулей. На рис. 3.14 показана схема компонента FIFO, реализуемого в ряде серий FPGA.



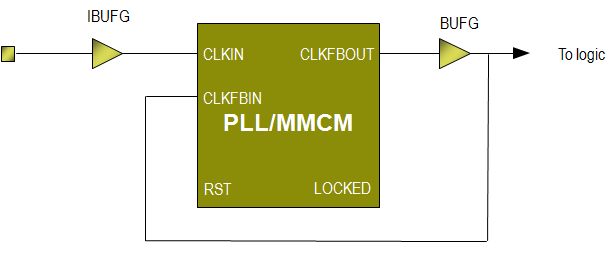
*Рисунок 3.14 Компонент FIFO на базе блока статической памяти*

Компонент FIFO (First In, First Out), называемый также «очередью», представляет собой массив данных, запись в который производится с одного «конца», а чтение – с другого. Это удобное представление того, что происходит с данными, но в действительности пересылка данных в процессе записи и чтения не происходит. Вместо этого при чтении и записи изменяются значения указателей на «голову» и «хвост» активных данных в FIFO. Для упрощения реализации таких схем указатели уже встроены в специализированный контроллер, который может быть использован для реализации компонента FIFO, если он требуется в проекте.

Следующий компонент относится к обязательным для установки в проект. Это генератор тактовых сигналов. Его назначением является подстройка фазы выходного тактового сигнала относительно входного (опорного). При работе микросхемы задержка распространения сигналов зависит от сочетания параметров, которое обозначается аббревиатурой PVT (Process, Voltage, Temperature). Эта аббревиатура отражает вариации технологического процесса (т.е. разброс параметров от одного кристалла к другому), колебания напряжения питания и колебания температуры. Все это приводит к изменению фазы тактового сигнала, что негативно отражается на стабильности работы цифровой схемы.

Для коррекции фазы тактового сигнала в процессе работы микросхемы используются компоненты тактовых генераторов. Это аппаратные блоки, которые сравнивают фазу входного тактового сигнала CLKin и сигнала обратной связи CLKfbin (от feedback – «обратная связь»), который проложен по кристаллу микросхемы так, чтобы наиболее показательно отражать сдвиг фазы. Тактовый генератор может быть построен по одному из подходов: Delay Locked Loop (DLL), Phase Locked Loop (PLL), или может использоваться их сочетанию (блок MMCM - Mixed Mode Clock Manager).

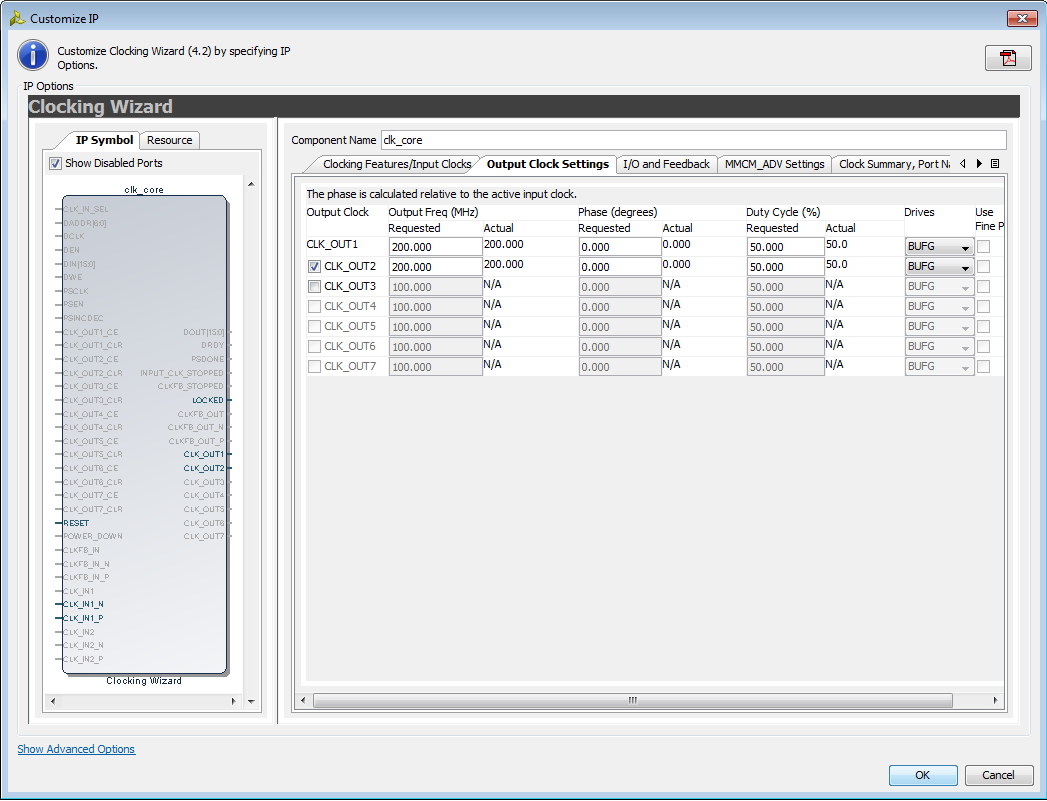
Применение генератора тактовых сигналов в ПЛИС показано на рис. 3.15.



*Рисунок 3.15 Применение генератора тактовых сигналов в ПЛИС*

В простейшем случае выходная частота такого генератора равна входной. Может показаться, что он не обязателен в проекте, тем более что такая схема, по сути, не имеет своего RTL-описания (ее можно представить как clk\_out <= clk\_in). Однако необходимость подстраивать фазу тактового сигнала в процессе работы делает блок PLL или MMCM практически обязательным, если схему предполагается загружать в реальную ПЛИС. Поведенческое моделирование можно выполнить и без установки в проект такого блока, потому что при этом используются идеализированные модели сигналов и компонентов, и блок тактового генератора полезной работы в модели не выполняет.

Установку тактового генератора в проект можно выполнить с помощью вспомогательных инструментов САПР. На рис. 3.16 показан генератор IP-ядра генератора тактового сигнала. Он позволяет указать значение входной тактовой частоты и настроить требуемые значения выходных частот.



*Рисунок 3.16 Генератор IP-ядра генератора тактовых сигналов в САПР ПЛИС*

В современных блоках PLL и MMCM выходная частота не обязана совпадать с входной. В генераторах имеется возможность установить каждый из нескольких выходов (4-6) в индивидуальное значение частоты, определяемое формулой вида:

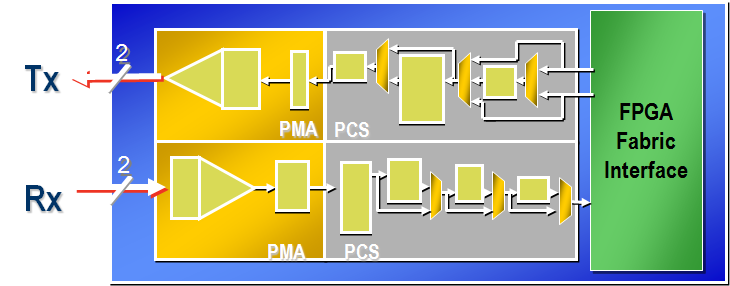
F = Fin\*M/D

где M, D – целые коэффициенты (например, 1-32 или 1-64 для различных моделей генераторов).

Таким образом, можно установить в проекте частоту, равную удвоенной или половинной входной, а также, например, 31/32 от величины входной частоты.

Применение тактового генератора настоятельно рекомендуется для проектов в ПЛИС. Отказ от него приводит к непредсказуемым нарушениям работы проекта, которые могут проявляться при определенных условиях, что дополнительно затрудняет не только их исправление, но и выявление.

Еще одной разновидностью аппаратного блока является MGT (Multi-Gigabit Transceiver), или «высокоскоростной последовательный приемопередатчик». Архитектура такого блока показана на рис. 3.17.



*Рисунок 3.17 Архитектура блока MGT в FPGA*

Блок существенно отличается от обычного цифрового вывода. Вместо единственного проводника сигналы передаются по дифференциальной паре. Это разновидность электрического интерфейса, в которой уровень 0 или 1 определяется не абсолютным значением напряжения, а разностью («*дифференциалом»*) напряжений между двумя линиями. Такой подход позволяет существенно увеличивать частоту передаваемых данных и увеличить устойчивость к помехам, т.к. помехи действуют одновременно на два проложенных рядом проводника, а разность напряжений остается такой же. Можно представить две величины, к которым добавили одно и то же значение. Независимо от добавляемого значения разность между величинами останется такой же.

Для дифференциальных интерфейсов существует множество вариантов скорости передачи данных и форматов передаваемых пакетов. Нижняя граница находится приблизительно на уровне 600 Мбит/с, а верхняя граница зависит от типа блока MGT и может составлять 3,125, 6,5, 10, 11,3, 28, 32, 36 Гбит/с. Верхняя граница скорости передачи постоянно изменяется по мере выхода новых серий элементной базы. Некоторые разновидности блоков MGT способны обеспечивать скорость передачи данных 116 Гбит/с.

Дифференциальная пара не может быть подключена непосредственно к цифровым сигналам внутри ПЛИС, поскольку цифровые элементы не могут работать на такой высокой тактовой частоте, чтобы обеспечить требуемый поток данных. Поэтому блок MGT использует подключение к ячейкам ПЛИС по параллельному интерфейсу и реализует аппаратно два уровня классической 7-уровневой сетевой модели OSI. Это уровни PMA (Physical Media Attachment) и PCS (Physical Code Sublayer). Они относятся к 2-му и 3-му уровня модели OSI (первый уровень – это непосредственно носитель данных, т.е. медные проводники, оптоволокно и т.д.). Таким образом, со стороны схемы в ПЛИС необходимо обеспечить поток данных (например, разрядностью 32 бита), а блок MGT сформирует требуемые сигналы для передачи по дифференциальной паре. Необходимо отметить, что такое подключение требует реализации соответствующего протокола обмена, т.е. ПЛИС не может просто передать 0 или 1 с помощью блока MGT в другую микросхему. Обмен данными с помощью MGT происходит с помощью пакетов. В самом блоке нет ограничений на формат пакета, поскольку в модели OSI это формируется на более высоких уровнях. Практическое использование MGT обычно подразумевает использование генераторов IP-ядер и библиотечных компонентов. В целом работа с этими блоками требует соответствующего опыта, поскольку даже простая демонстрация передачи пакетов требует большого объема RTL-описаний и программного кода.

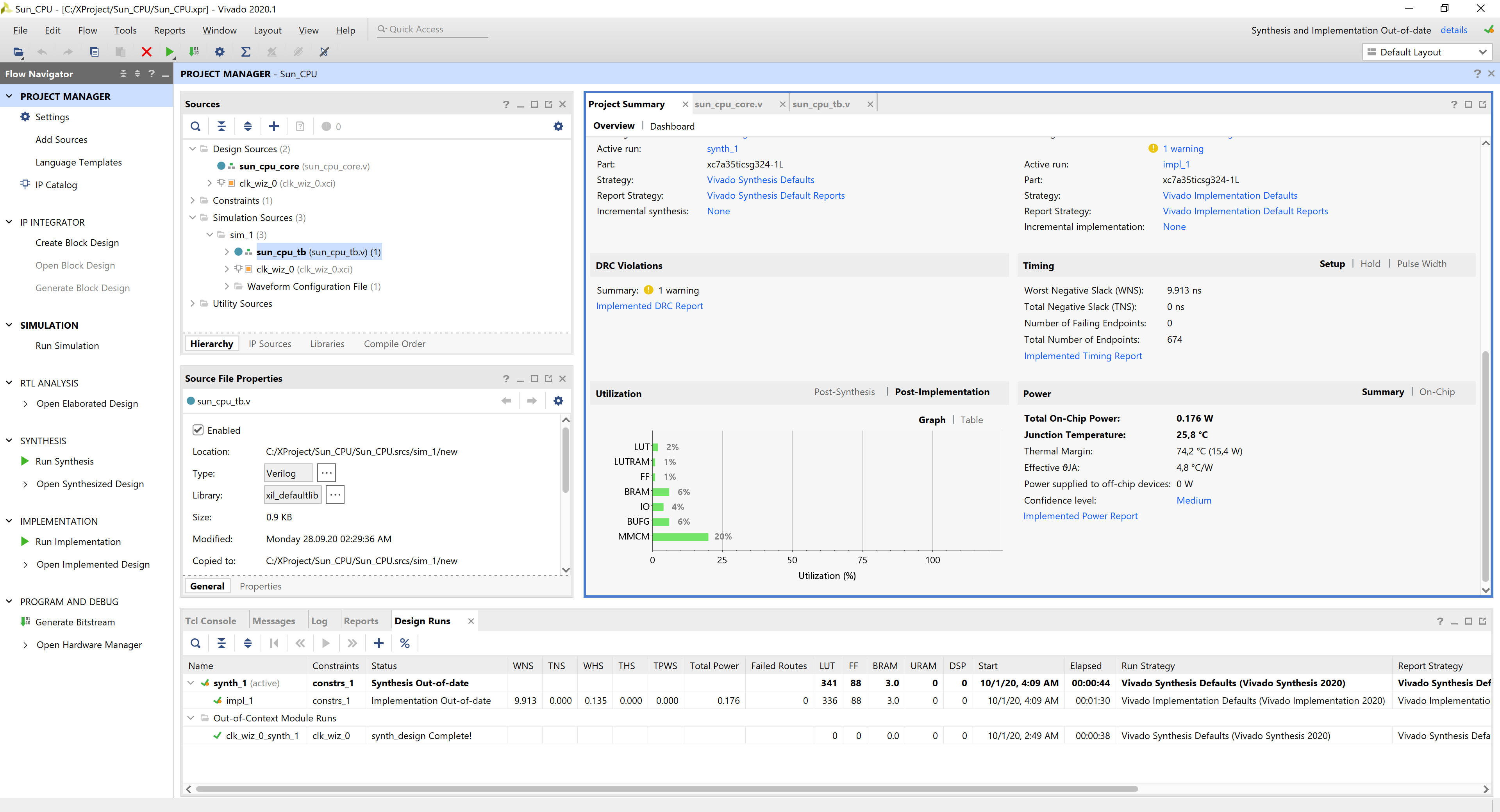
Блоки MGT используются при реализации таких интерфейсов, как PCI Express, Serial ATA (SATA), 10 G Ethernet и многих других. Практическое проектирование таких систем является предметом отдельного изучения. Оно осложняется тем, что при передаче высокочастотных сигналов на их параметры влияют многие факторы, в том числе геометрические характеристики проводников на печатных платах. Для нормальной передачи сигнала с частотой 1 ГГц и выше недостаточно просто соединить контакты на печатной плате, поскольку емкость и индуктивность дорожки будет искажать форму высокочастотных сигналов. Разработка печатных плат для высокочастотных сигналов требует применения специальных САПР, и несмотря на то, что блоки MGT имеются в ПЛИС, их работоспособность во многом определяется качеством проектирования и изготовления печатной платы.

Широкое применение блоков MGT в ПЛИС обусловлено быстрым развитием проводных и беспроводных сетей. Например, магистральные коммутаторы в оптоволоконных сетях могут быть реализованы на базе ПЛИС с десятками блоков MGT, маршрутизация пакетов между которыми обеспечивается цифровыми схемами, реализуемыми с помощью логических ячеек.

3.4 Инструменты разработки для FPGA

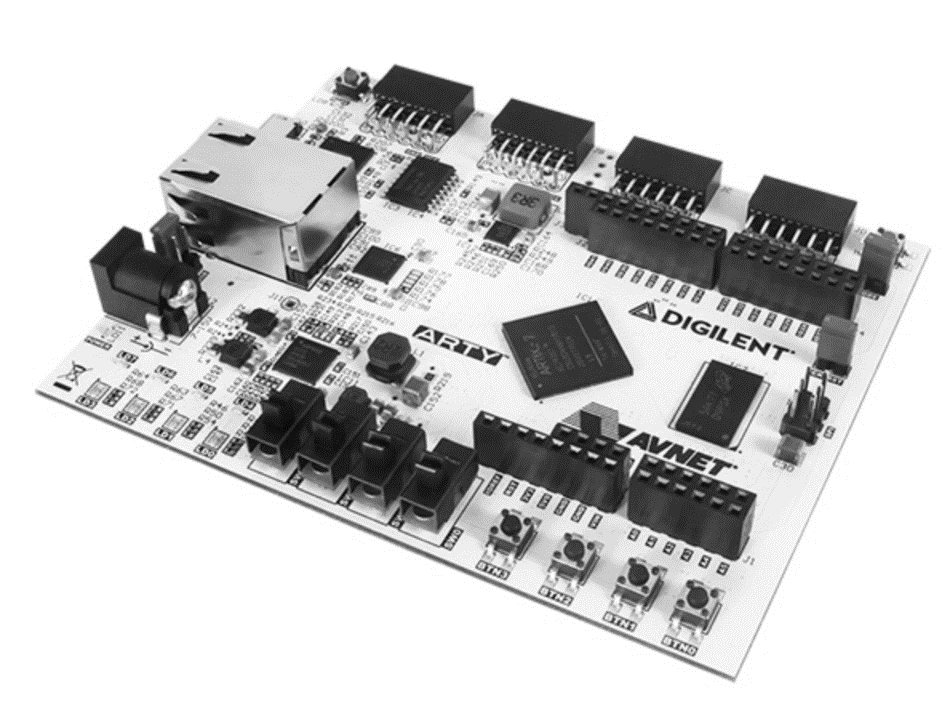
Разработку схемы на базе ПЛИС можно провести с помощью систем автоматизированного проектирования (САПР). Особенностью ПЛИС является закрытый производителем формат файла, описывающего конфигурацию проекта, поэтому по крайней мере завершающие стадии разработки должны выполняться с помощью программных продуктов, поставляемых производителем. Ранние стадии (ввод проекта, моделирование и синтез) доступны и для САПР других компаний.

Производители ПЛИС обычно обеспечивают САПР, поддерживающие полный цикл разработки, от ввода проекта до загрузки получаемого файла в ПЛИС. Внешний вид окна САПР Xilinx Vivado показан на рис. 3.18.



*Рисунок 3.18 Внешний вид окна САПР ПЛИС Xilinx Vivado*

Для быстрого освоения элементной базы производители микросхем часто выпускают «отладочные платы». Это платы, на которых кроме основной микросхемы установлены вспомогательные компоненты, интерфейс для программирования и наиболее показательные периферийные устройства. Назначением такой платы является обучение и предоставление разработчикам образца подключения часто используемых внешних устройств. Отладочные платы могут выпускаться как самим производителем микросхем, так и другими компаниями. Внешний вид отладочной платы для ПЛИС показан на рис. 3.19. Показанный вариант компоновки является не единственным. Отладочные платы могут иметь интерфейс PCI Express и устанавливаться в настольный ПК, или наоборот, размещаться в небольших габаритах.



*Рисунок 3.19 Пример отладочной платы на базе ПЛИС*

Сочетание компьютера с установленной САПР ПЛИС и отладочной платы позволяет создать цифровую схему и загрузить макет устройства в ПЛИС без привлечения дополнительного оборудования и инструментов. Упрощенный маршрут проектирования для ПЛИС показан на рис. 3.20.



*Рисунок 3.20 Упрощенный маршрут проектирования для ПЛИС*

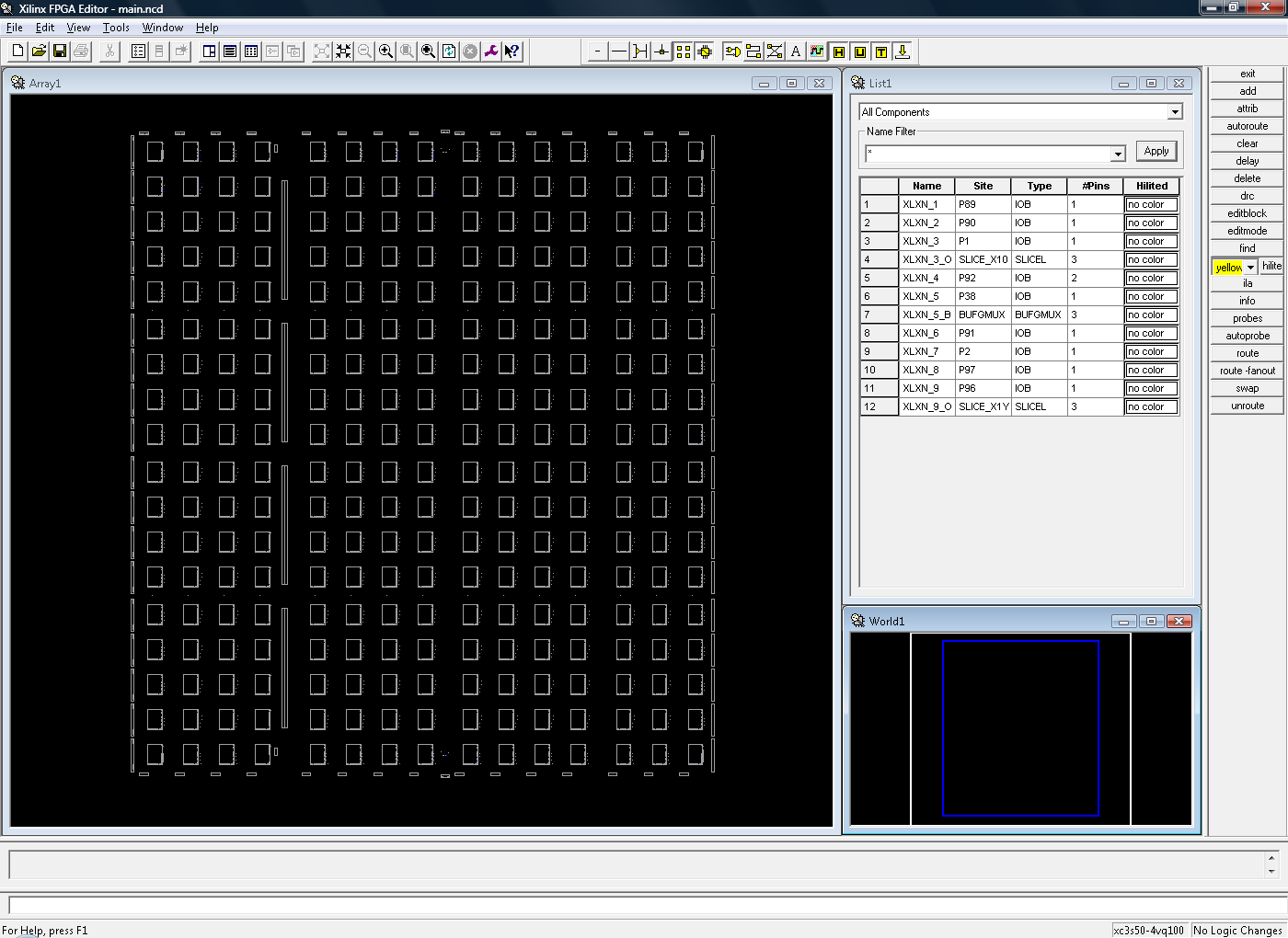
В качестве минимального набора входных данных требуется файл верхнего уровня (top level module), описывающий схему проекта, а также файл проектных ограничений (constraints).

Первый этап, синтез (synthesize), состоит в построении схемы соединений на основе исходного файла на языке описания аппаратуры. Полученный список связей (netlist) является достаточно абстрактным и не привязывает элементы схемы к конкретным компонентам ПЛИС. Кроме того, внешние выводы проекта следует привязать к выводам конкретной микросхемы, которая будет использоваться. Эту информацию невозможно привести в тексте на языке описания аппаратуры, поэтому в САПР добавляется специальный файл проектных ограничений, где указано, например, что сигнал «a\_in» должен быть связан с выводом ПЛИС «A1». Поскольку каждая модель ПЛИС имеет собственные варианты исполнения, проектные ограничения являются специфическими для каждой ПЛИС, отладочной платы и проекта в целом.

Имея список связей и сопровождающий его перечень проектных ограничений, САПР может приступить ко второму крупному этапу – implementation. В этот этап включены шаги по определению, какой элемент ПЛИС будет выполнять конкретные действия и как будут проведены соединительные линии между ними. Наиболее длительное действие здесь – «размещение и трассировка» (Place and Route). Для сложных проектов может быть потрачено очень большое время (часы и даже дни) на получение приемлемого результата, поскольку при большом объеме ПЛИС существует очень много вариантов размещения элементов схемы в отдельных ячейках и проведения связей между ними.

Часто в проектные ограничения включают также требования к времени распространения сигналов. В этом случае установка компонентов в противоположные углы ПЛИС имеет существенный риск превысить допустимые времена задержек. Очевидно, что для достижения высокой тактовой частоты ячейки ПЛИС, которые должны быть соединены, следует поставить рядом. Однако в сложных проектах часто появляются разветвленные линии связей между множеством ячеек, поэтому добиться компактной расстановки схемы часто бывает невозможно. Именно поэтому этап implementation занимает длительное время, а повышение тактовой частоты проекта в ПЛИС требует определенной квалификации разработчика.

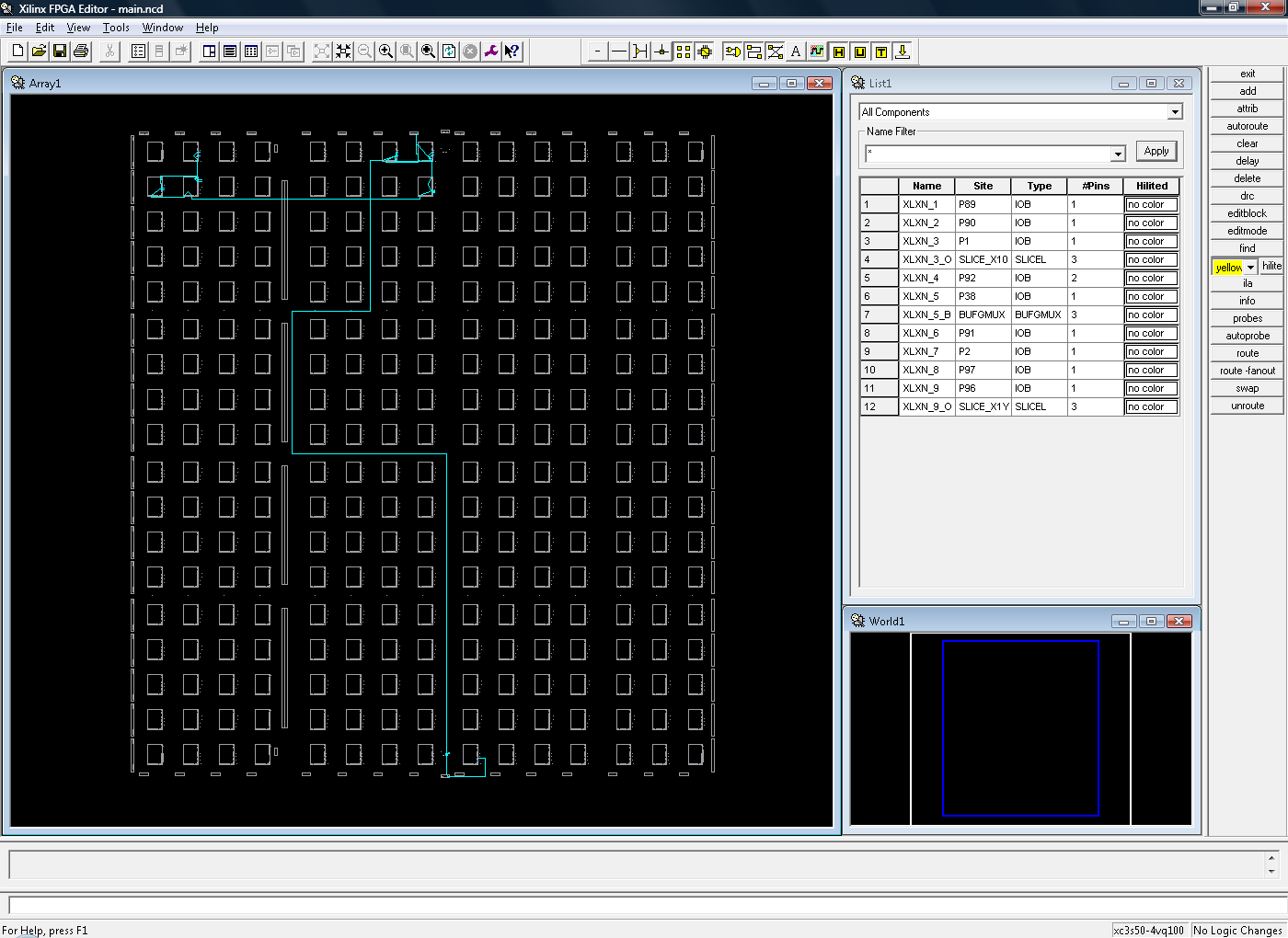
На рис. 3.21 показан внешний вид внутреннего инструмента САПР ПЛИС – редактора топологии. Этот редактор не является необходимым при разработке проекта, однако позволяет показать структуру ПЛИС и режимы работы ячеек.



*Рисунок 3.21 Внешний вид ПЛИС в редакторе топологии*

На рис. 3.22 показан фрагмент редактора топологии после выполнения трассировки проекта. Видны добавленные трассировочные линии, показанные голубым цветом.

Можно отметить, что показанная ПЛИС относится к устаревшему семейству Spartan-3 и имеет минимальный логический объем. Даже в этом случае микросхема содержит несколько сотен логических ячеек и проектирование путем их ручного соединения – крайне непродуктивный путь. Изучение соединений является вспомогательным инструментом и обычно применяется на уровне крупных блоков проекта для оптимизации их взаимной расстановки.



*Рисунок 3.22 Внешний вид ПЛИС в редакторе топологии с выполненной трассировкой проекта*

Средства загрузки конфигурации ПЛИС обычно являются частью САПР. Большинство отладочных плат уже содержат схемы для загрузки конфигурации, подключаемые к порту USB компьютера, однако сами по себе ПЛИС не имеют такой возможности. Поэтому при выборе отладочных плат для проекта необходимо проверять, требуется ли для работы с ними отдельный внешний программатор.

3.5 Выводы по главе

Программируемые логические интегральные схемы (ПЛИС) – удобный и практически безальтернативный способ получения работоспособного макета цифровой микросхемы. Поскольку ПЛИС хранит конфигурацию в статической энергозависимой памяти, она имеет неограниченный ресурс перезагрузки конфигурации и может быть использована для создания макетов схожим образом с обычной компиляцией программ. САПР ПЛИС поддерживают сквозной маршрут проектирования – от ввода схемы до загрузки полученной конфигурации в микросхему.

В составе ПЛИС с архитектурой FPGA находятся конфигурируемые логические ячейки, способные при соединении представить практически любой цифровой узел. Кроме ячеек, в FPGA добавляют аппаратные компоненты, которые не конфигурируются, однако реализуют часто востребованные в практических проектах схемы. К таким компонентам относятся блоки статической памяти, блоки «умножение с накоплением» и блоки MGT (высокоскоростных последовательных приемопередатчиков). На базе этих компонентов часто создают серийные изделия, в которых ПЛИС используются не как макеты будущих микросхем, а как элемент конечного продукта, который не планируется для замены. Это становится возможным потому, что десятки (а в больших ПЛИС сотни и тысячи) аппаратных компонентов обеспечивают высокую производительность ПЛИС в задачах цифровой обработки сигналов и сетевом оборудовании.

При разработке схем на базе ПЛИС необходимо следовать тем же рекомендациям, которые характерны для обычных цифровых микросхем. Например, синхронные схемы следует использовать и в проектах ПЛИС. При превышении определенного размера проекта (обычно сотни тысяч логических ячеек) следует использовать подход GALS. Проблема «темного кремния» для ПЛИС пока не стоит остро, поскольку ячейки ПЛИС сами по себе являются достаточно «разреженными» и ресурсы кристалла не используются на 100%. Поэтому производители обычно могут обеспечить допустимый уровень тепловыделения и ПЛИС могут охлаждаться распространенными несложными системами охлаждения.

Контрольные вопросы:

1. Что такое ПЛИС? Что является отличительной чертой архитектуры FPGA?

2. Какие основные компоненты присутствуют в ПЛИС с архитектурой FPGA?

3. Как изменяется номенклатура и удельный вес аппаратных компонентов в составе ПЛИС FPGA?

4. Какие основные компоненты имеет логическая ячейка FPGA?