

Что такое ПЛИС (FPGA)?

Программируемая Логическая
Интегральная Схема
(Field Programmable Gate Array)

Профессор Мелентьев О.Г.

«Микросхема-конструктор» – полупроводниковый кристалл с цифровыми компонентами без жестко зафиксированных металлических соединений, формирующих конкретную схему. Соединениями в FPGA можно управлять путем замыкания транзисторных ключей, размещенных на том же кристалле.

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

Проект для ПЛИС может быть разработан в виде принципиальной схемы или написан на языках описания аппаратуры типа Verilog или VHDL.

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

Архитектура ПЛИС.

Микросхема ПЛИС состоит из:

конфигурируемых логических блоков (базовых логических элементов), реализующих требуемую логическую функцию;

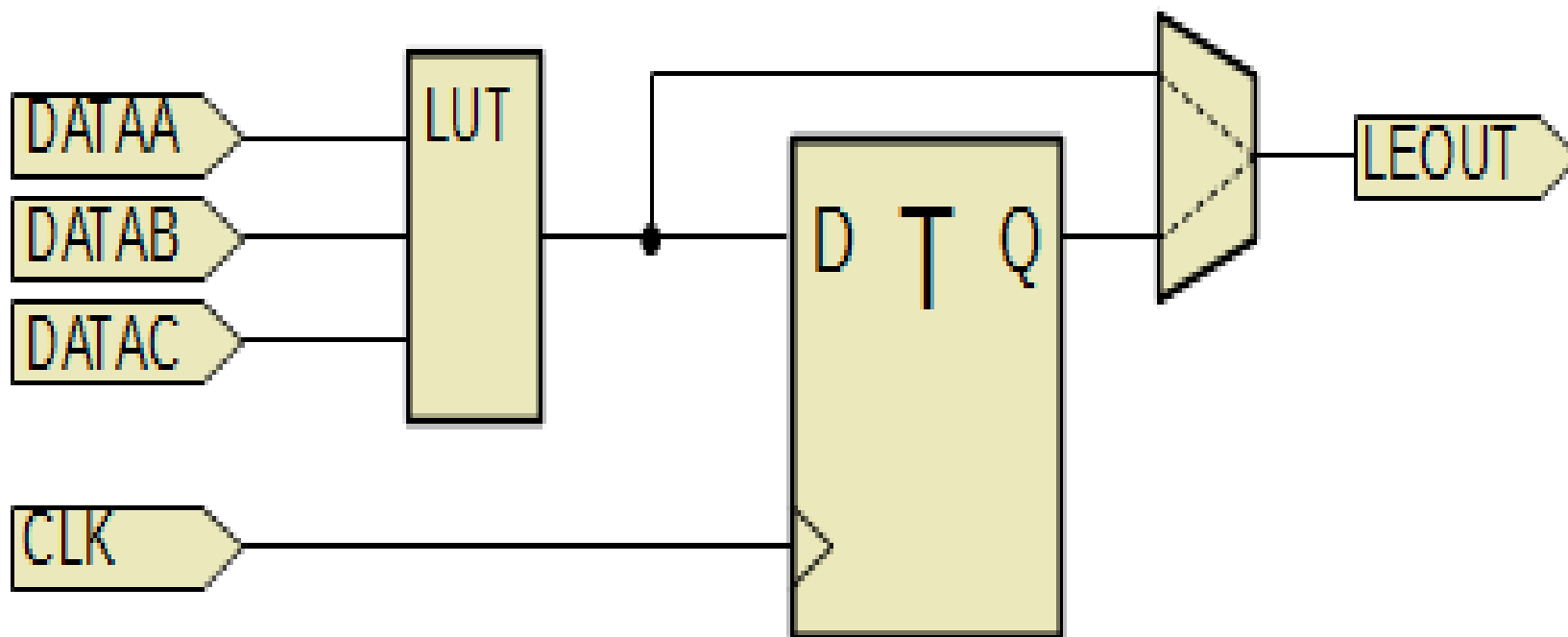
программируемых электронных связей между конфигурируемыми логическими блоками;

программируемых блоков ввода/вывода, обеспечивающих связь внешнего вывода микросхемы с внутренней логикой.

В современных ПЛИС часто бывают встроены дополнительно блоки памяти, блоки DSP или умножители, PLL и другие компоненты.

Обобщенная структура логического элемента (LE) ПЛИС

В ПЛИС в качестве простейшего логического элемента используют соединение программируемого комбинационного устройства и D-триггера



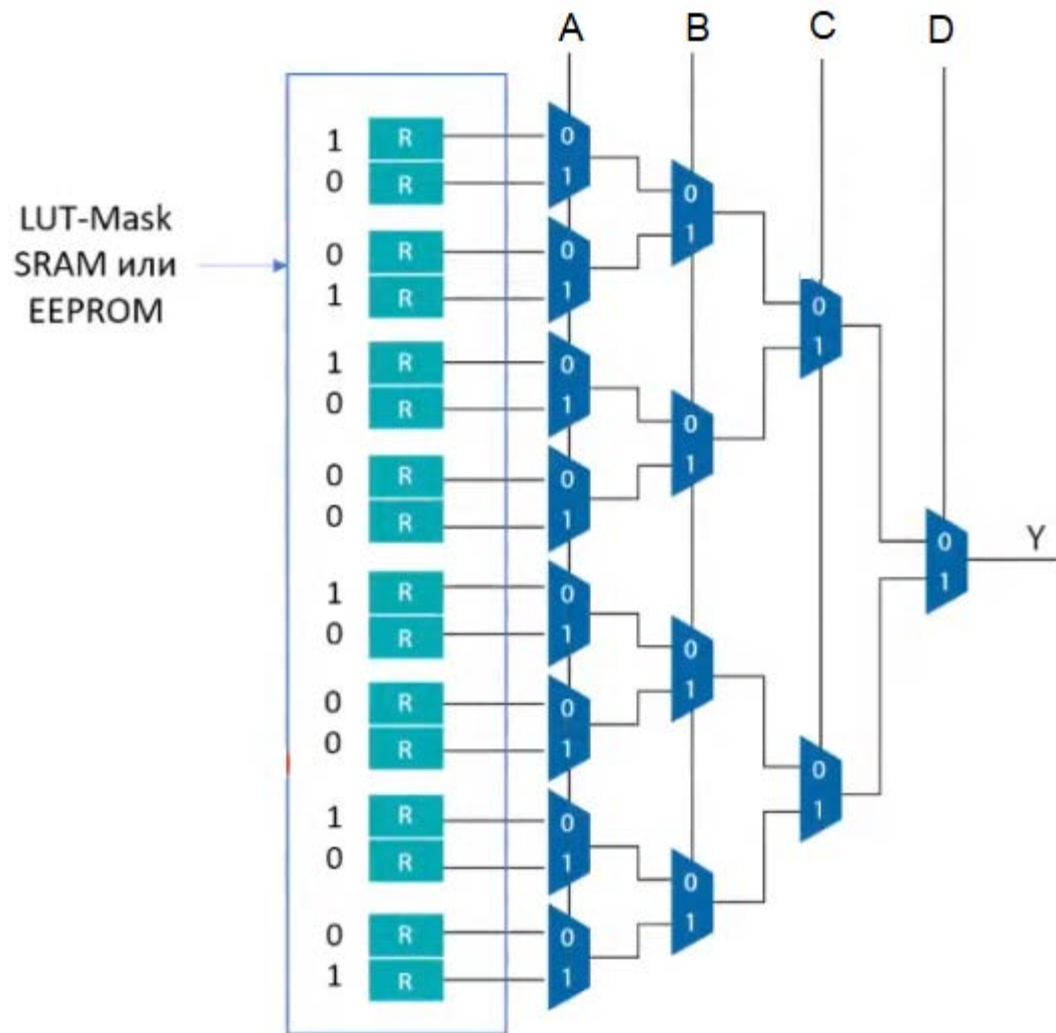
DATAA(B,C) – логические входы, LEOUT выход,
CLK – вход тактовых импульсов

Устройство LUT(Look up Table)

Справочная таблица или таблица истинности

Метод реализации функции, в котором вычисление заменяется поиском по готовой таблице решений.

Метод позволяет реализовать любую логическую функцию в виде памяти SRAM, где адрес – это аргумент, а содержимое ячейки – значение.



$$Y = \bar{A}\bar{B} + ABC\bar{D} + ABCD$$

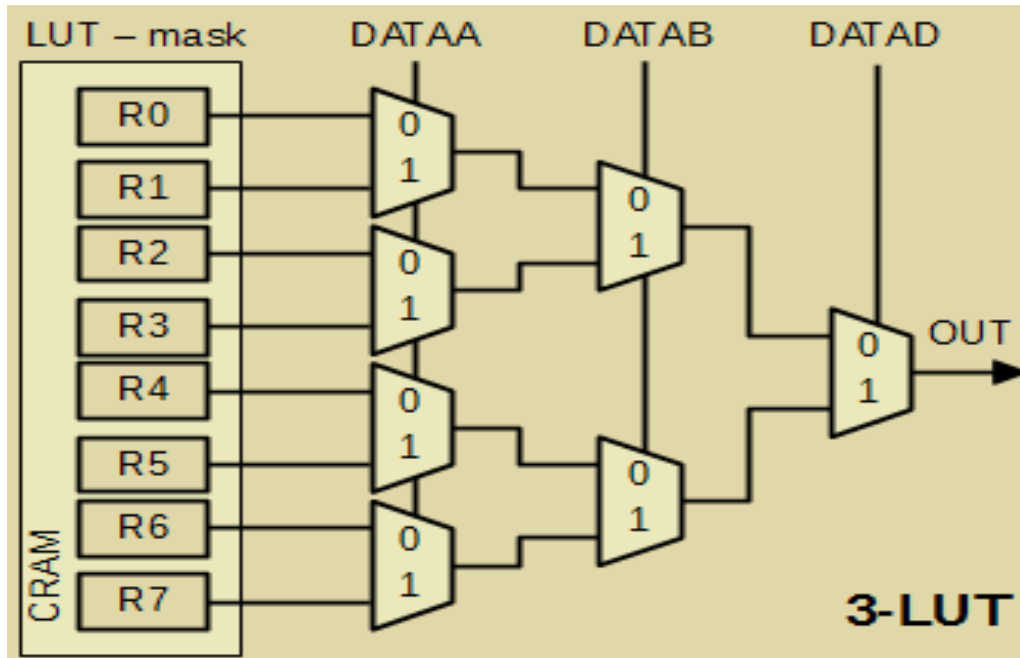
Если от ячейки требуется работа в качестве только комбинационного устройства, то выходной мультиплексор коммутирует выход элемента LUT на выход всей ячейки,

если выход должен быть регистровым, то сигнал с LUT защелкивается по сигналу синхронизации в D-триггер, выход которого через мультиплексор соединяется с LEOUT.

В ПЛИС для конфигурации используется оперативная память **CRAM** (Configuration RAM). Эта память распределена по всему кристаллу, значения, записанные в нее, управляют внутренним коммутационным полем, определяя структуру синтезируемого цифрового устройства

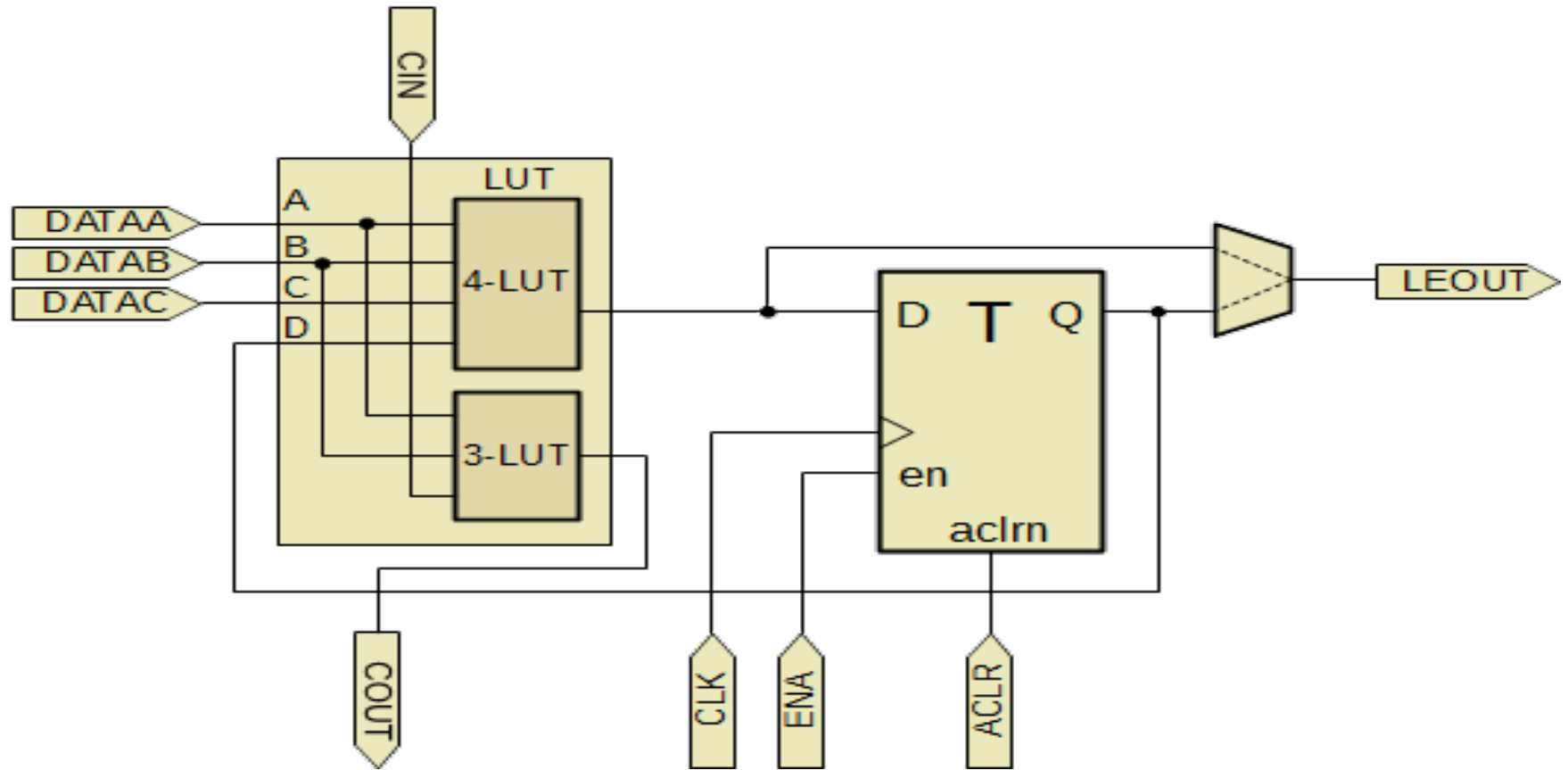
Управляющий вход мультиплексора (на рисунке не показан) подключен к соответствующему биту конфигурационной памяти CRAM.

Мультиплексорами управляют сигналы входных портов для построения k-входовой LUT (k-LUT), которая реализует любую логическую функцию из k переменных, требуется 2^k бит SRAM и $2^k - 1$ мультиплексоров



Такой подход позволяет достаточно точно спрогнозировать время прохождения сигнала, и оно не будет зависеть от реализуемой логической функции. Эта особенность делает возможным временной анализ схемы.

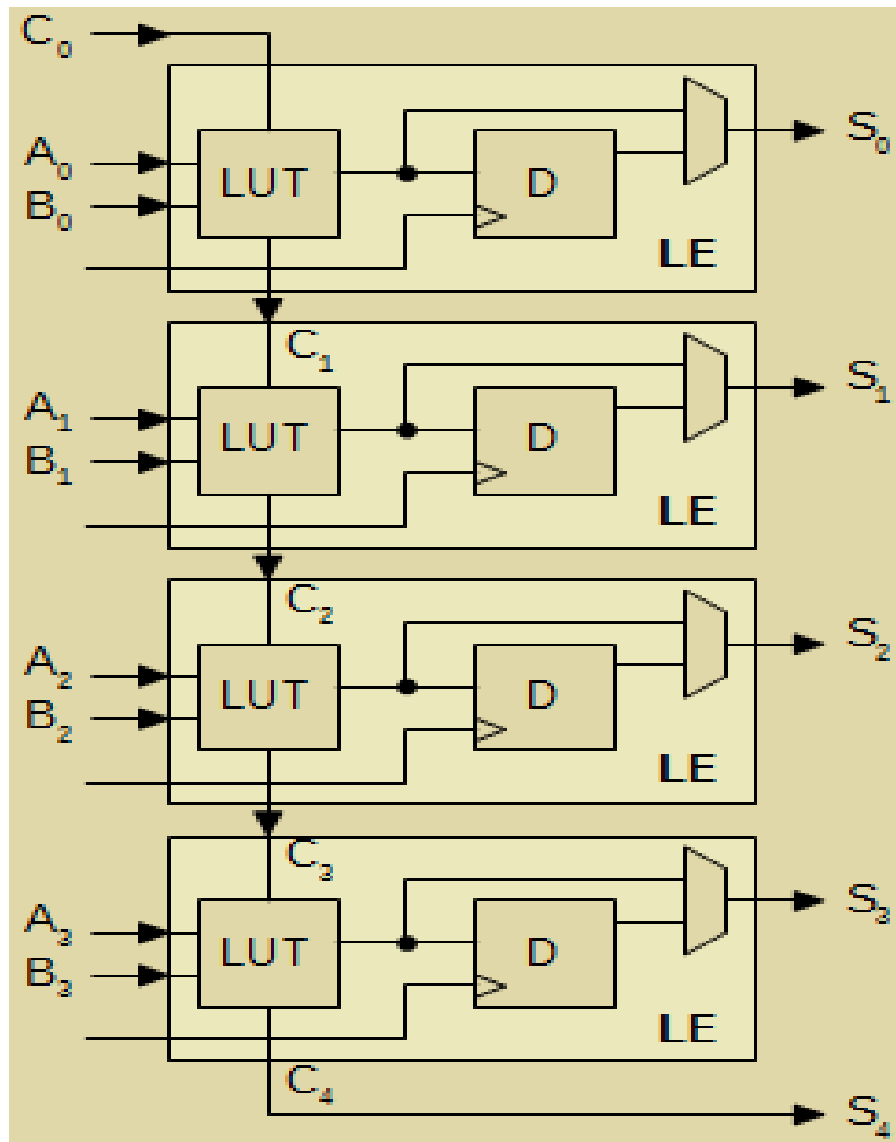
Структура логического элемента (LE) с каналом переноса



Для реализации всех известных триггеров и сумматоров в LE добавляют:

- обратную связь с выхода D-триггера;
- вход разрешения ENA ("Enable" – "Включить")
- вход асинхронного сброса ACLR ("Asynchronous Clear");
- вход и выход переноса (CIN, COUT)

Четырехразрядный сумматор



При разработке базовой логической ячейки решались две задачи:

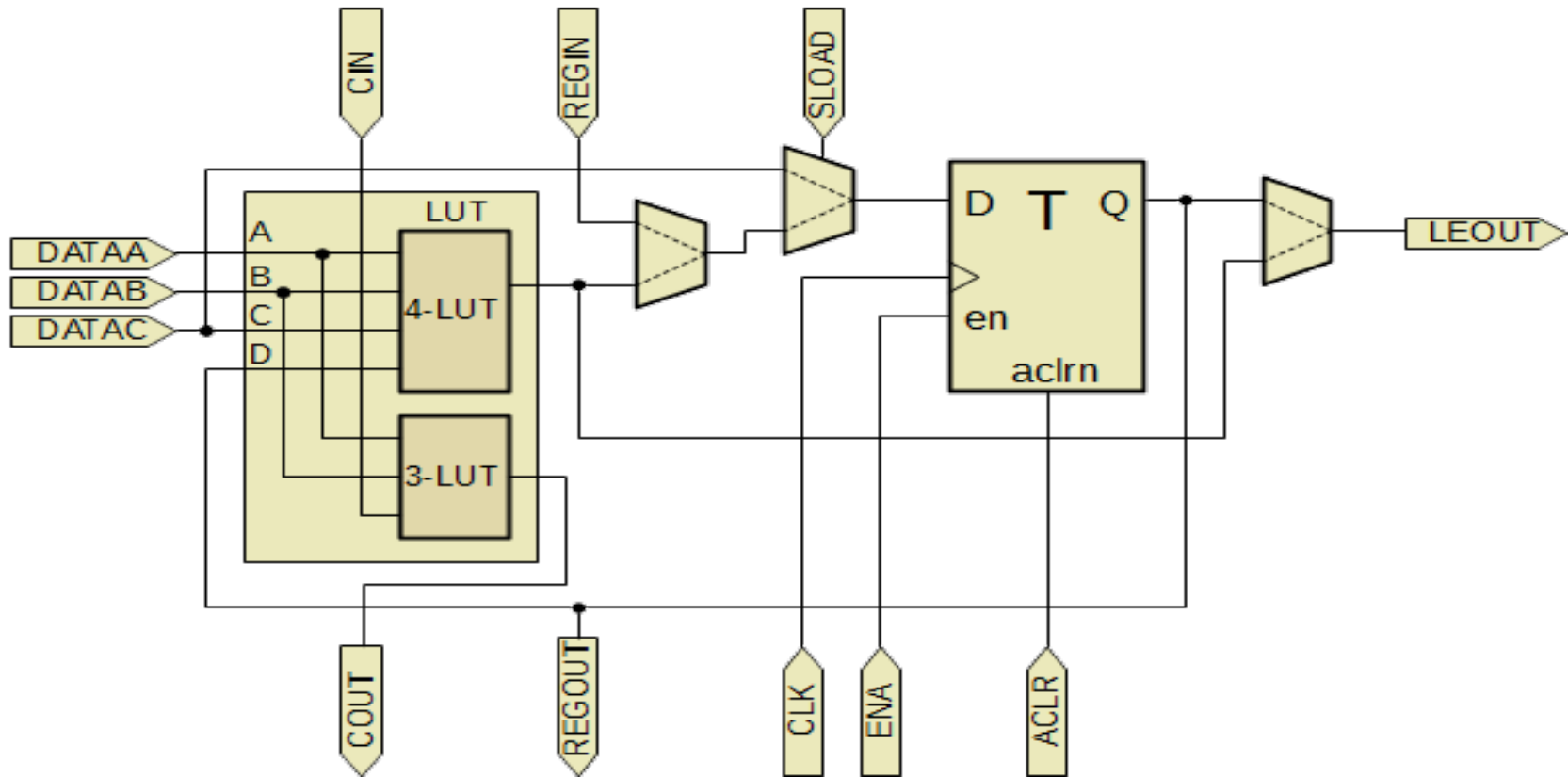
во-первых, синтезируемые устройства должны обладать максимальным быстродействием,
во-вторых, использование ресурсов должно быть, как можно более полным

Дополнение к схеме, даёт возможность использовать комбинационное устройство и триггер элемента отдельно для синтеза независимых модулей.

Мультиплексор на входе триггера позволит выбирать источник сигнала: либо с входа DATAC, либо с выхода LUT.

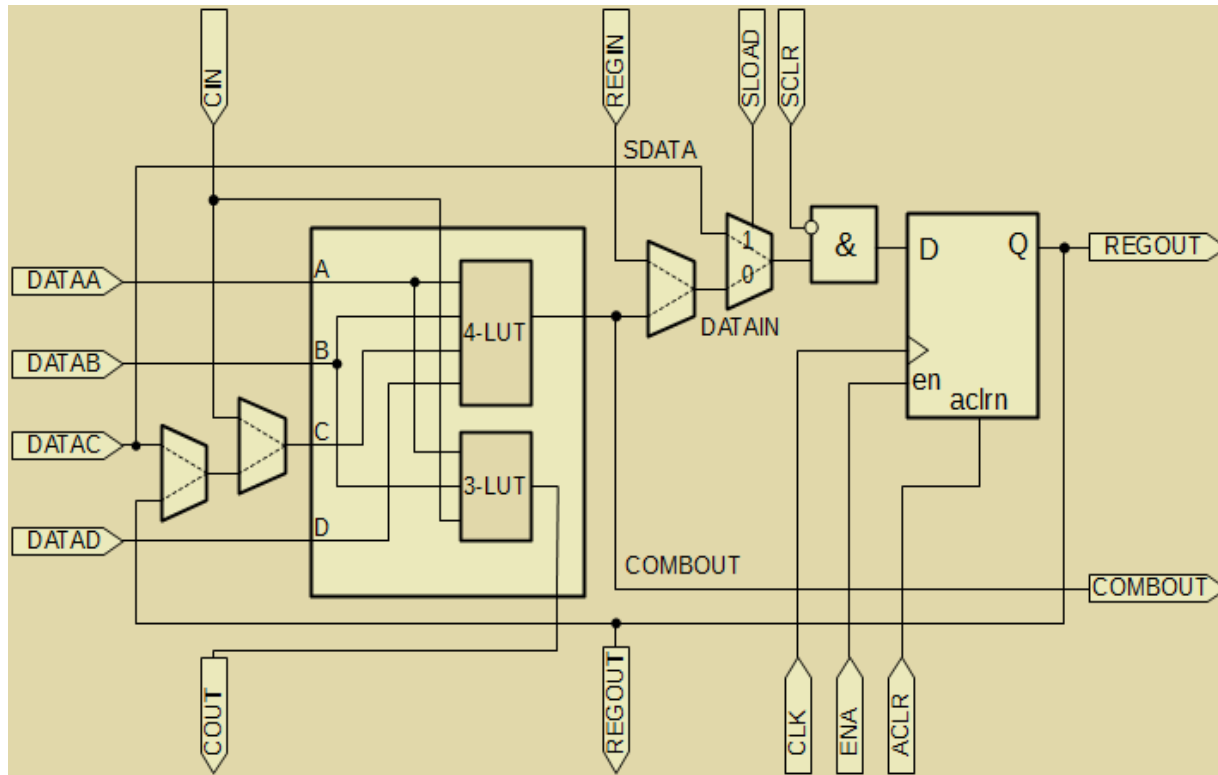
Появляется возможность организовать дополнительный канал соединения триггеров соседних LE для увеличения быстродействия при построении последовательных регистров.

Структура логического элемента (LE) с возможностью разделения LUT и триггера



Вход REGIN и выход REGOUT образуют выделенный канал для соединения триггеров, вход SLOAD ("Synchronous Loading" – "Синхронная загрузка") управляет выбором источника сигнала для входа триггера.

Логический элемент (LE) Cyclone IV



Добавился сигнал синхронной очистки SCLR ("Synchronous Clear"). На входе DATAC мультиплексор выбирает источник сигнала, благодаря чему LUT может реализовывать логическую функцию четырех переменных, кроме того, в качестве переменной может быть использован флаг переноса или выход собственного триггера

Выходные мультиплексоры LE имеют более сложную структуру.

Логические элементы LE объединяются в логические блоки LAB (Logic array blocks).

В Cyclone IV каждый LAB содержит:

- 16 логических ячеек;

- сигналы управления LAB;

- цепи флага переноса LE;

- цепи каскадного объединения регистров;

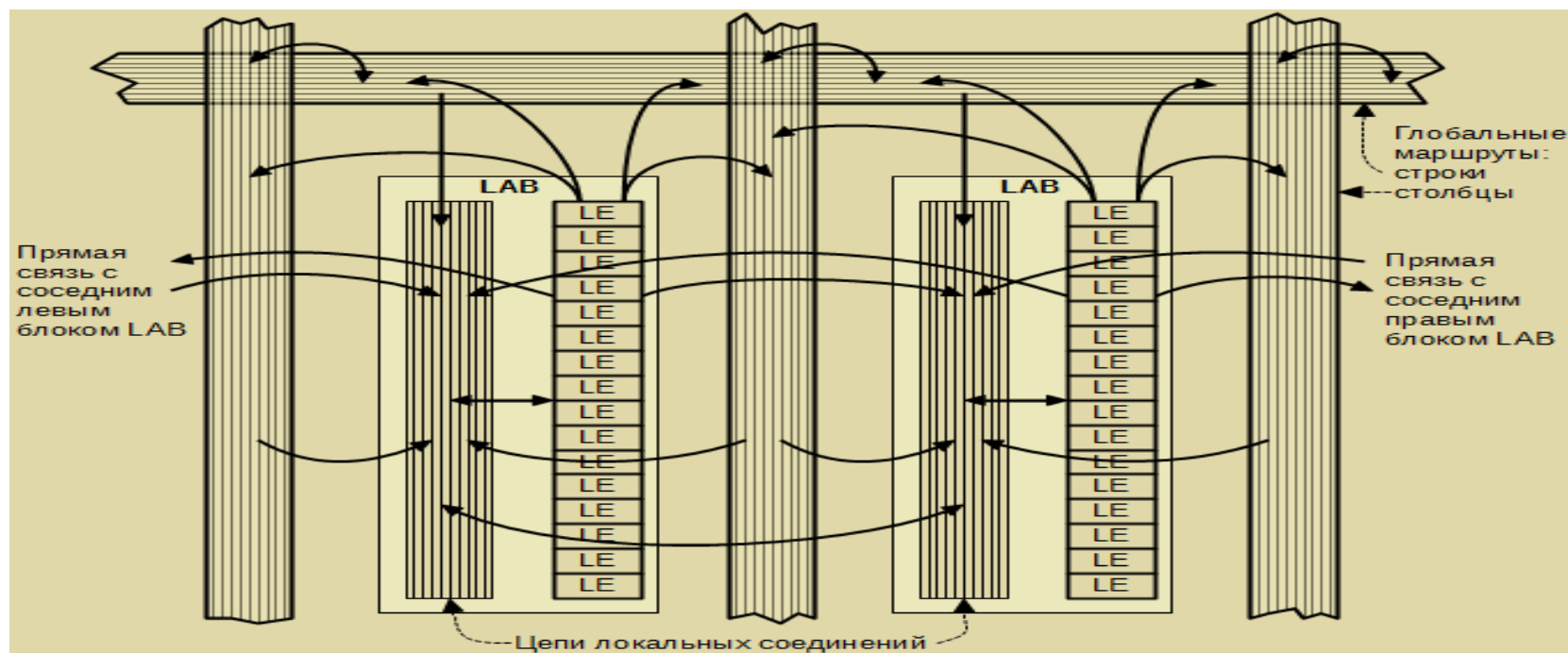
- цепи локальных соединений.

Цепи локальных соединений передают сигналы между ячейками LE в одном LAB.

Цепи объединения регистров соединяют выход регистра одного LE с входами регистров прилегающих ячеек LE.

Компилятор Quartus II размещает связанную логику в LAB или в соседних LAB, позволяя использовать локальные цепи связи и связи регистров для увеличения производительности и эффективности размещения.

Структура соединений LAB в коммутационном поле ПЛИС



Каждый LE имеет три выхода, которые обеспечивают соединение с коммутационным полем ПЛИС. Эти выходы поступают на строки и столбцы глобальных соединительных трасс и на маршруты локальных соединений.

Как и в базовом логическом элементе, LUT или триггер могут независимо управлять этими выходами.

Для управления всеми ячейками LE в пределах одного LAB одновременно, в логический блок встроена специальная логика и выделены особые линии – каналы управления. По таким каналам распространяются широковещательные (в пределах одного LAB) сигналы управления.

Архитектура позволяет одновременно использовать до восьми управляющих сигналов:

два тактовых сигнала (labclk1 и labclk2);

два сигнала разрешения (labclkena1 и labclkena2);

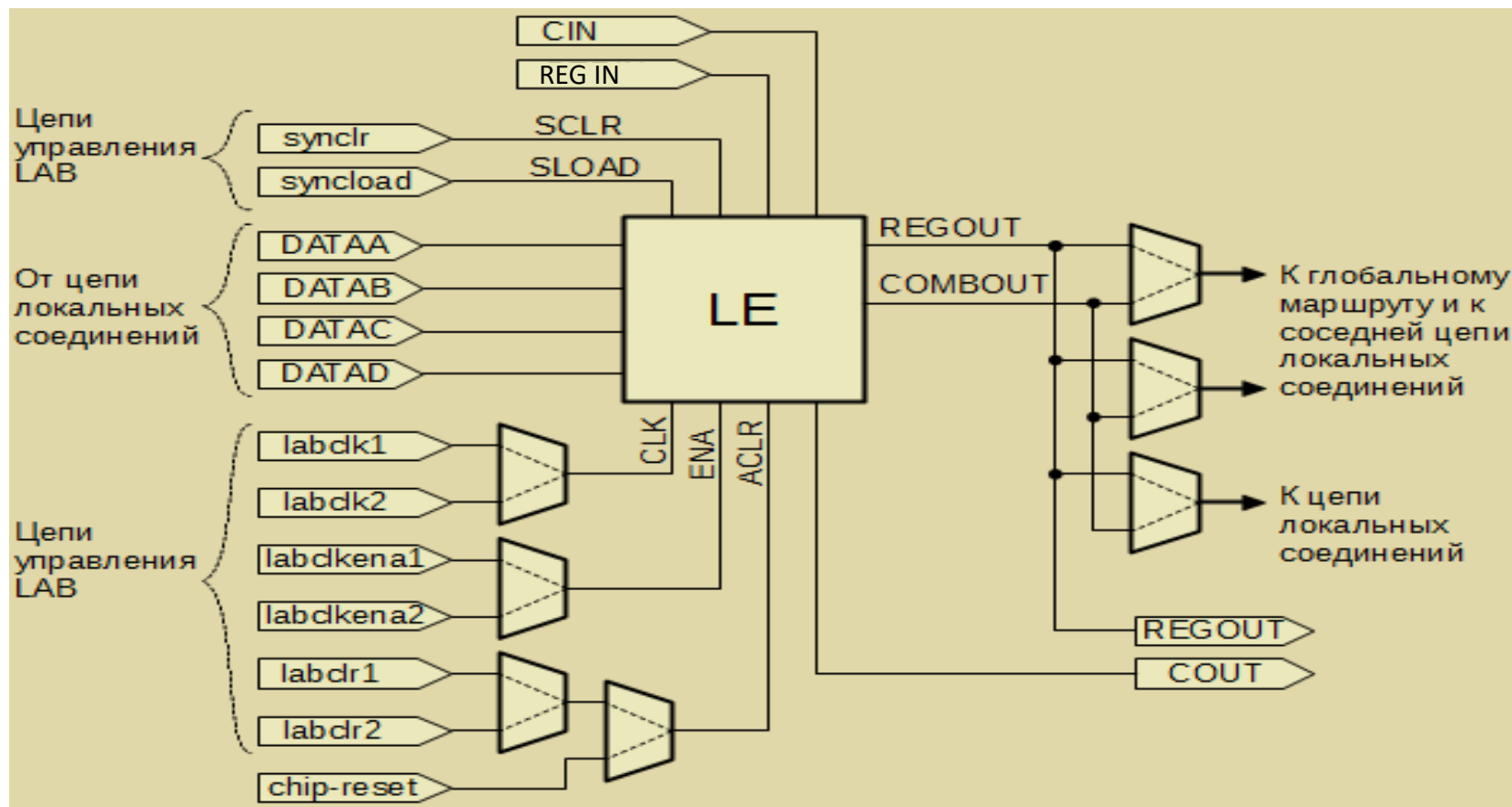
два сигнала асинхронного сброса (labclr1 и labclr2);

сигнал синхронного сброса/очистки (syncclr);

сигнал синхронной загрузки (syncload).

Сигналы синхронной загрузки и сброса удобно использовать для синтеза различных счетчиков и регистров. Эти сигналы оказывают воздействие на все триггеры LE в пределах одного LAB.

Структура взаимодействия LE с сигналами управления



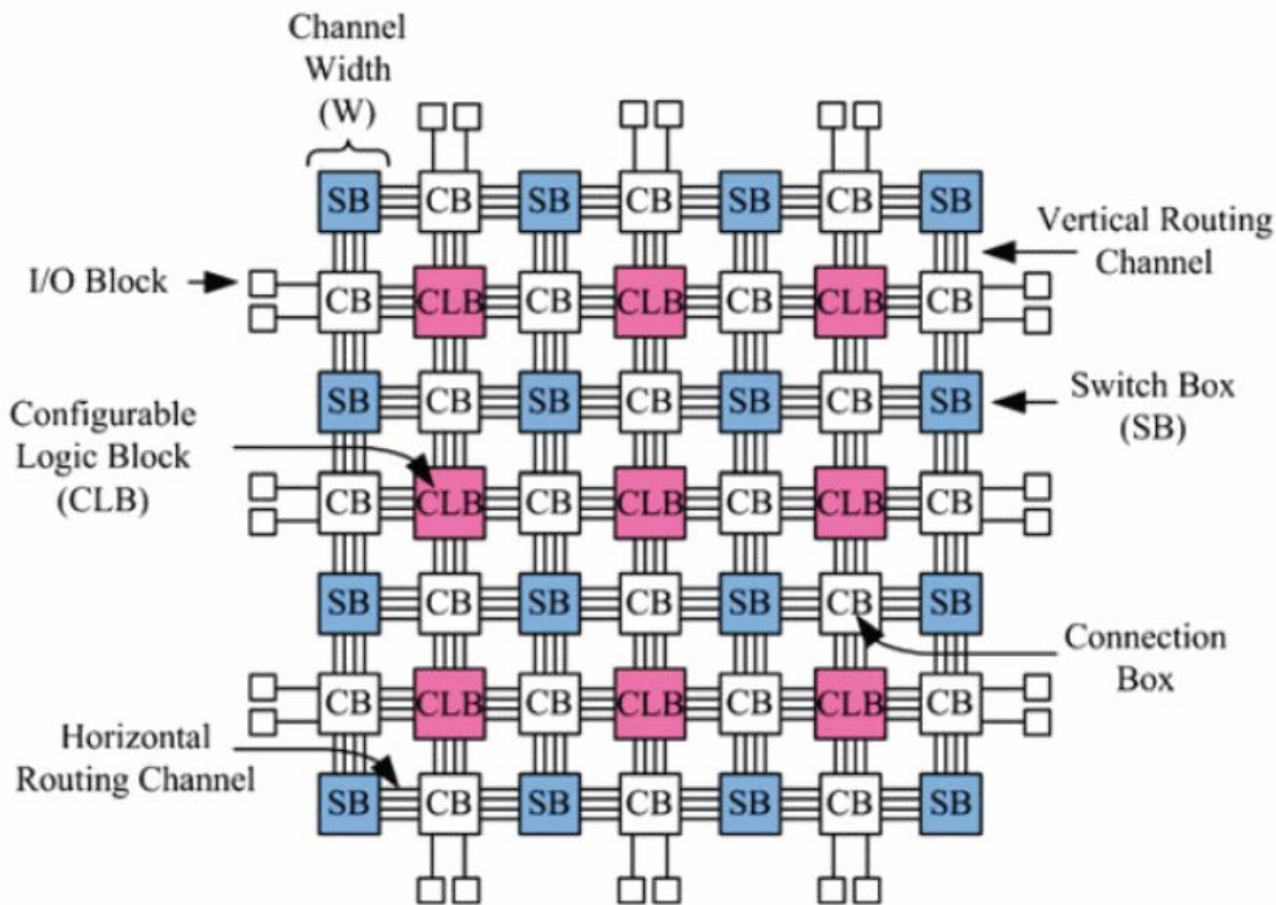
Чтобы в ПЛИС заработала нужная нам цифровая схема необходимо сконфигурировать имеющиеся логические блоки соответствующим образом и запрограммировать связи между логическими блоками.

Для этого в ПЛИС имеются специальные конфигурируемые коммутаторы.

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

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

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.

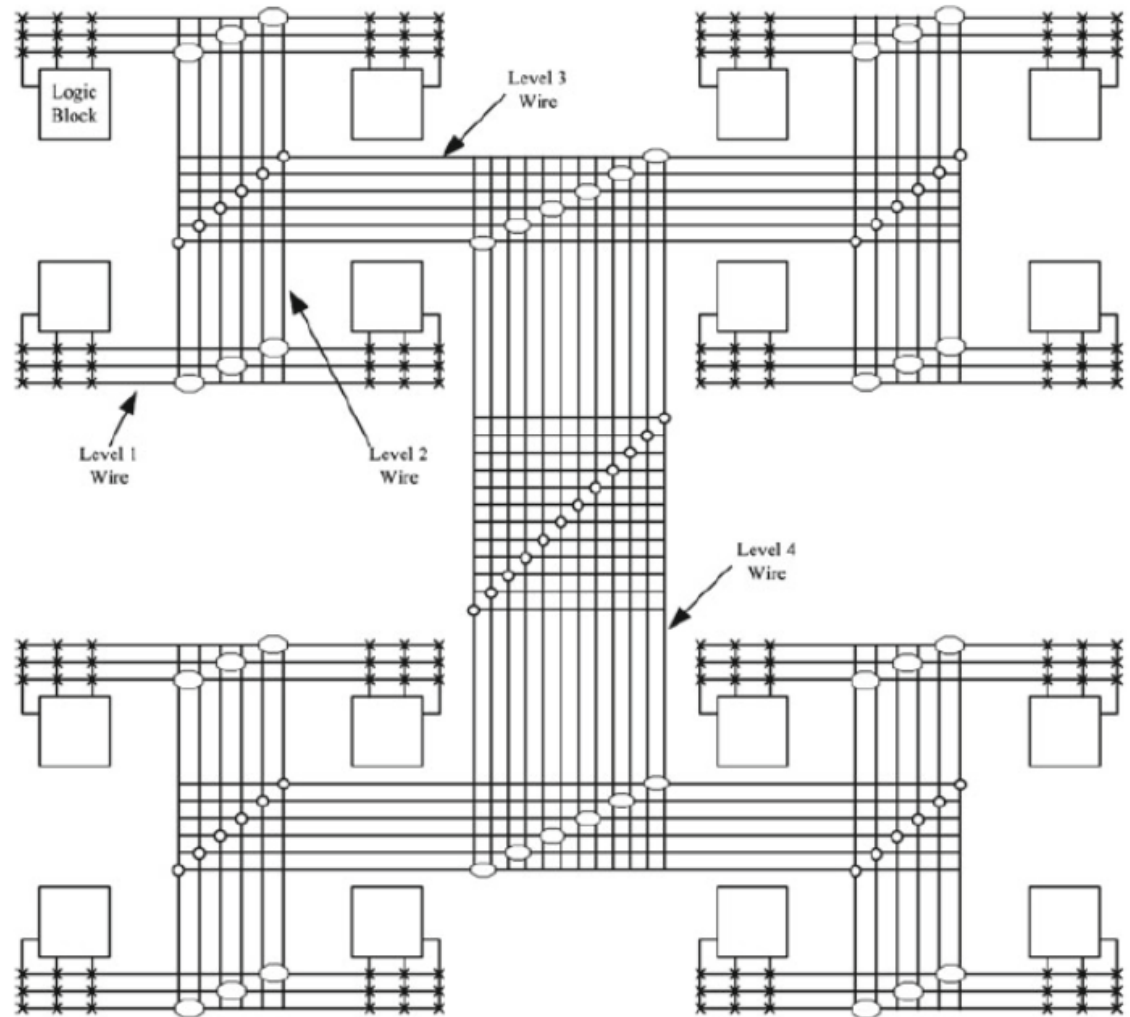


Типичный пример таких микросхем – это серии Altera Cyclone и Stratix.

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

Здесь близлежащие CLB соединить довольно просто, не нужно много коммутаторов и получающиеся связи работают быстро.

Если нужен более крупный блок вычислителей, то сигнал должен выйти на более высокий уровень иерархии и потом зайти вглубь в соседнюю «комнату».



Программное обеспечение для проектирования ПЛИС.

Программное обеспечение для проектирования ПЛИС, а именно компилятор (синтезатор логики, фиттер и ассемблер)

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 удалить.

Затем компилятор должен вместиť 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.

Программное обеспечение компании Альтера: Quartus II.
ПО Xilinx для проектирования для ПЛИС: ISE Suite, Vivaldo Design Suite.

Компиляция проекта в Quartus

Quartus Prime Lite Edition - D:/Rabota_2022/3_Projects/reseiver_fm_analog/receiver_analog - receiver_analog

File Edit View Project Assignments Processing Tools Window Help

Search altera.com

receiver_analog

Project Navigator Hierarchy

Entity:Instance Logic Cell

MAX 10: 10M50SAE144C8GES

receiver_analog 10055 (3)

Tasks Compilation

Task

Compile Design

Analysis & Synthesis

Compilation Report - receiver_analog

Table of Contents

- Flow Summary
- Flow Settings
- Flow Non-Default Global Settings
- Flow Elapsed Time
- Flow OS Summary
- Flow Log
- Analysis & Synthesis
- Fitter
- Assembler
- Power Analyzer
- Flow Messages
- Flow Suppressed Messages

Flow Summary

<<Filter>>

- Flow Status
- Quartus Prime Version
- Revision Name
- Top-level Entity Name
- Family
- Device
- Timing Models
- Total logic elements
- Total registers
- Total pins

IP Catalog

Installed IP

- Project Directory
 - No Selection Available
- Library
 - Basic Functions
 - DSP
 - Interface Protocols
 - Memory Interfaces and Controllers
 - Processors and Peripherals
 - University Program

+ Add...

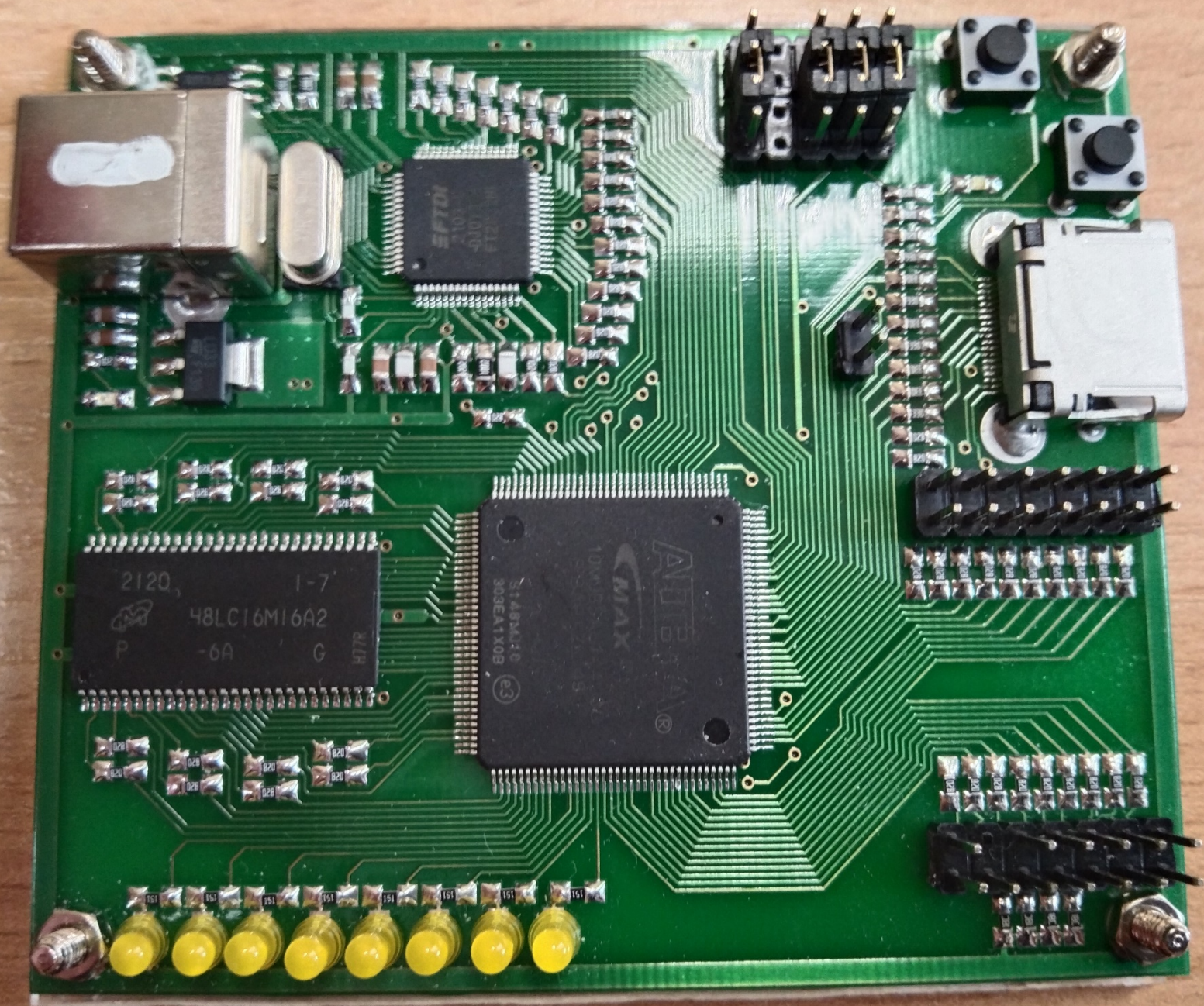
All Find... Find Next

Messages

Type	ID	Message
Info	332114	Report Metastability: Found 8 synchronizer chains.
Info	332102	Design is not fully constrained for setup requirements
Info	332102	Design is not fully constrained for hold requirements
Info		Quartus Prime Timing Analyzer was successful. 0 errors, 6 warnings
Info	293000	Quartus Prime Full Compilation was successful. 0 errors, 40 warnings
Info		*****
Info		Running Quartus Prime Netlist Viewers Preprocess
Info		Command: quartus_npp receiver_analog -c receiver_analog --netlist_type=sgate
Warning	18236	Number of processors has not been specified which may cause overloading on shared machines. Set the global assignment NUM_PARALLEL_PROCESSORS
Info		Quartus Prime Netlist Viewers Preprocess was successful. 0 errors, 1 warning

System (28) Processing (433)

100% 00:00:01



Основные этапы разработки прототипа

Спецификация. Разработка технического задания

Математическое моделирование

Разработка RTL дизайна (микроархитектура)

Разработка тестового окружения. Симуляция. Верификация.

Синтез: Компиляция.

Конфигурация ПЛИС

Имплементация в прототип

Основные этапы разработки прототипа

