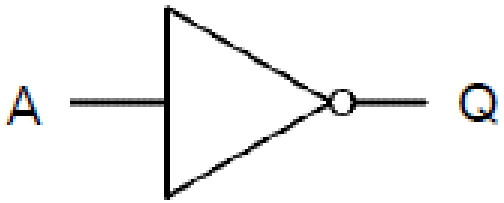
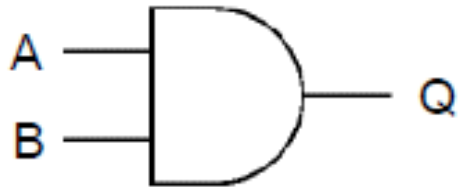
FPGA

Преподаватели: Эпштейн Леонид Борисович

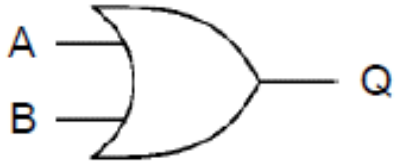
Логические элементы



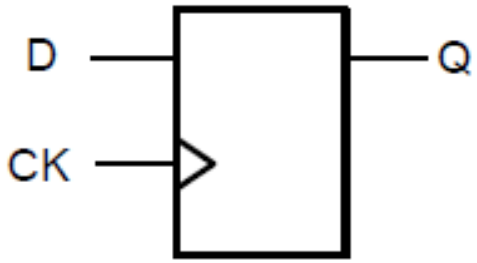
NOT



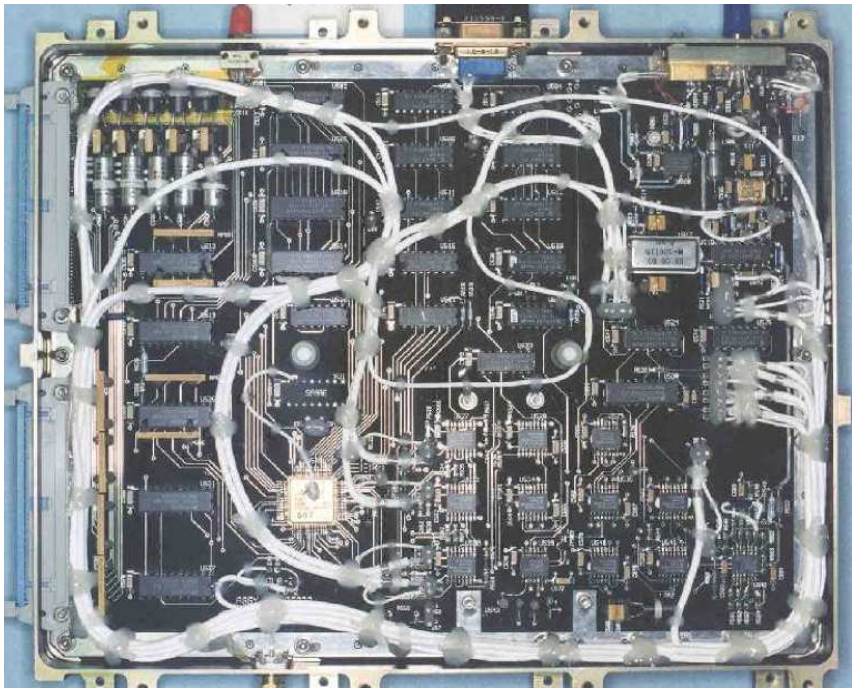
AND

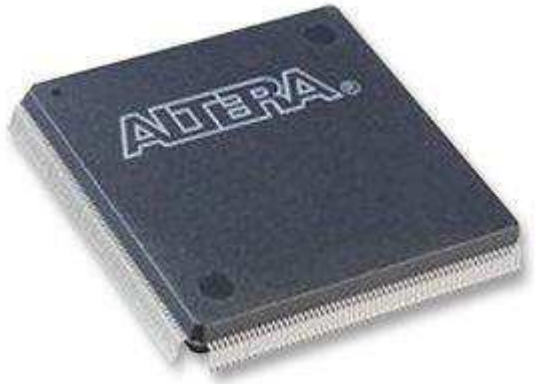


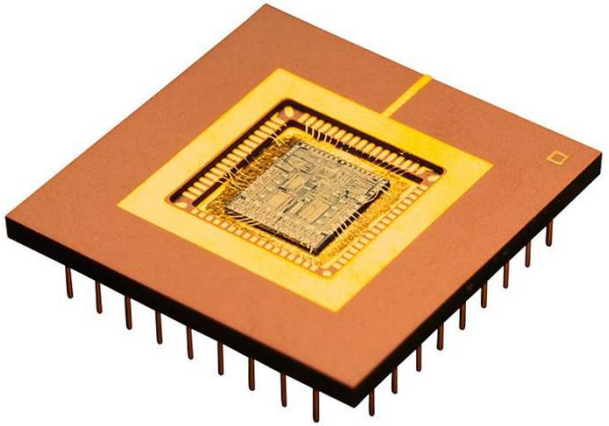
OR



FPGA







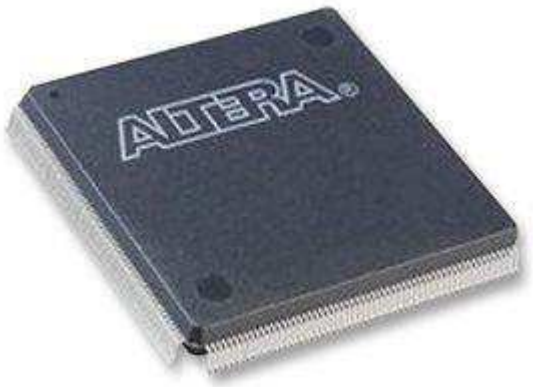
ASIC

Программируемая логическая интегральная схема (ПЛИС)

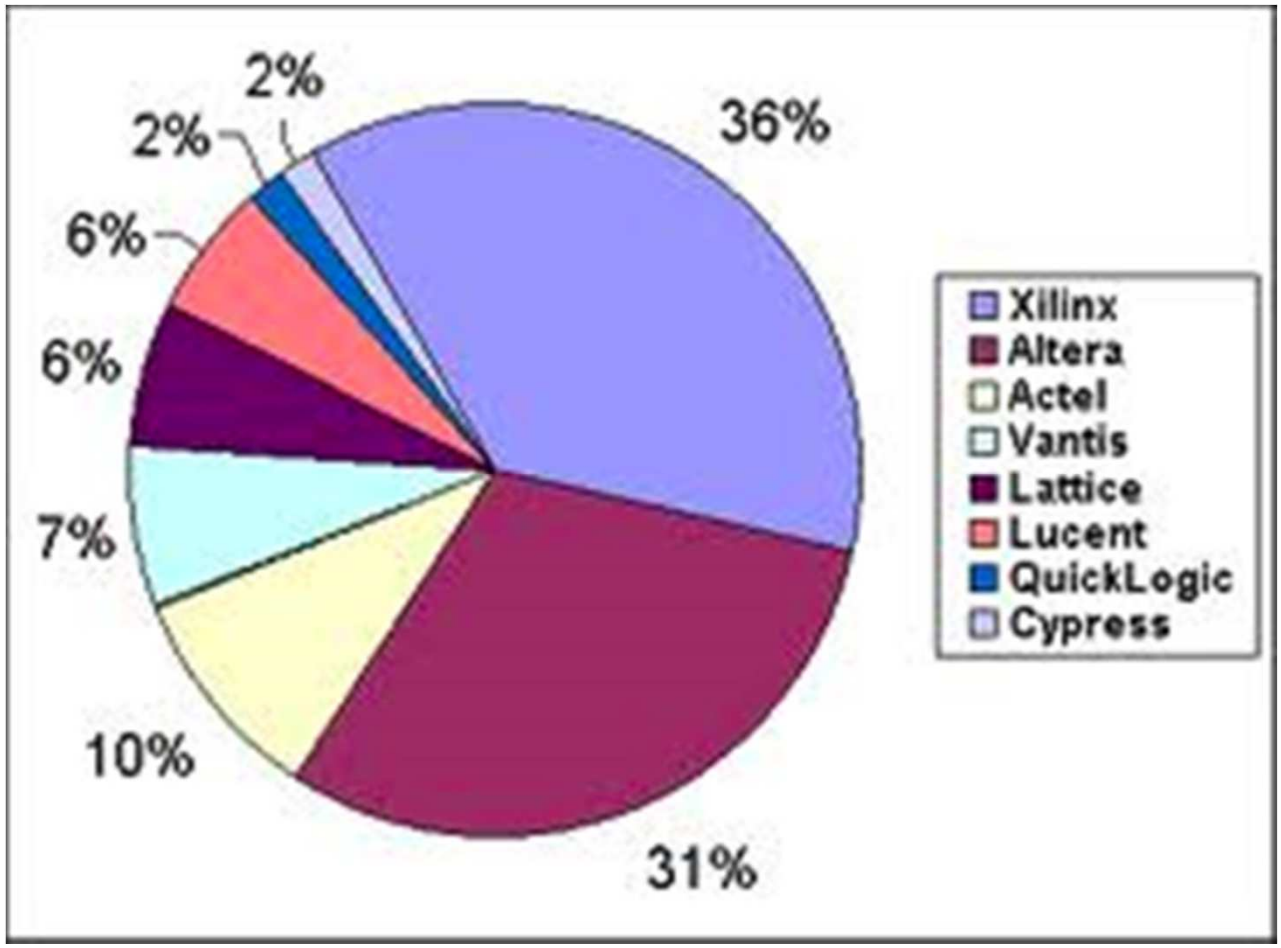
Полупроводниковое устройство, которое может быть сконфигурировано разработчиком после изготовления.

FPGA могут быть модифицированы практически в любой момент в процессе их использования.

Они состоят из конфигурируемых логических блоков (логические вентили или gates). Принципиальное отличие FPGA от DSP состоит в том, что и функции блоков, и конфигурация соединений между ними могут меняться с помощью специальных сигналов, посылаемых схеме.



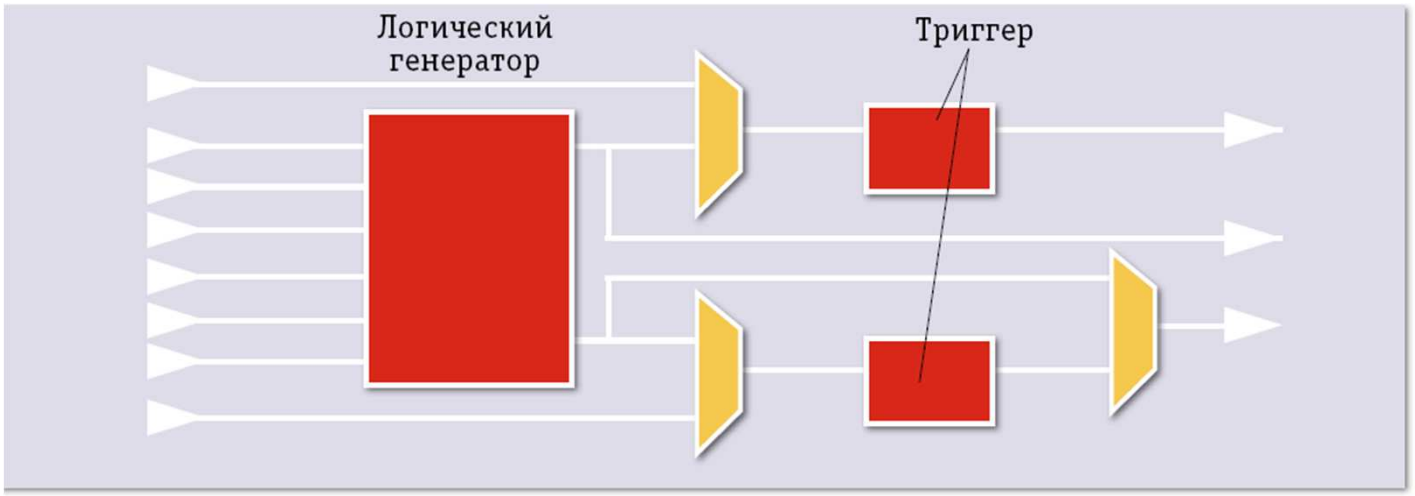
Производители FPGA



Внутренняя структура

FPGA включают в себя три главных программируемых элемента: нескоммутированные программируемые логические блоки (ПЛБ), блоки ввода-вывода (БВВ) и внутренние связи. ПЛБ являются функциональными элементами для построения логики пользователя, БВВ обеспечивают связь между контактами корпуса и внутренними сигнальными линиями.
Программируемые ресурсы внутренних связей обеспечивают управление путями соединения входов и выходов ПЛБ и блоков ввода-вывода на соответствующие сети.

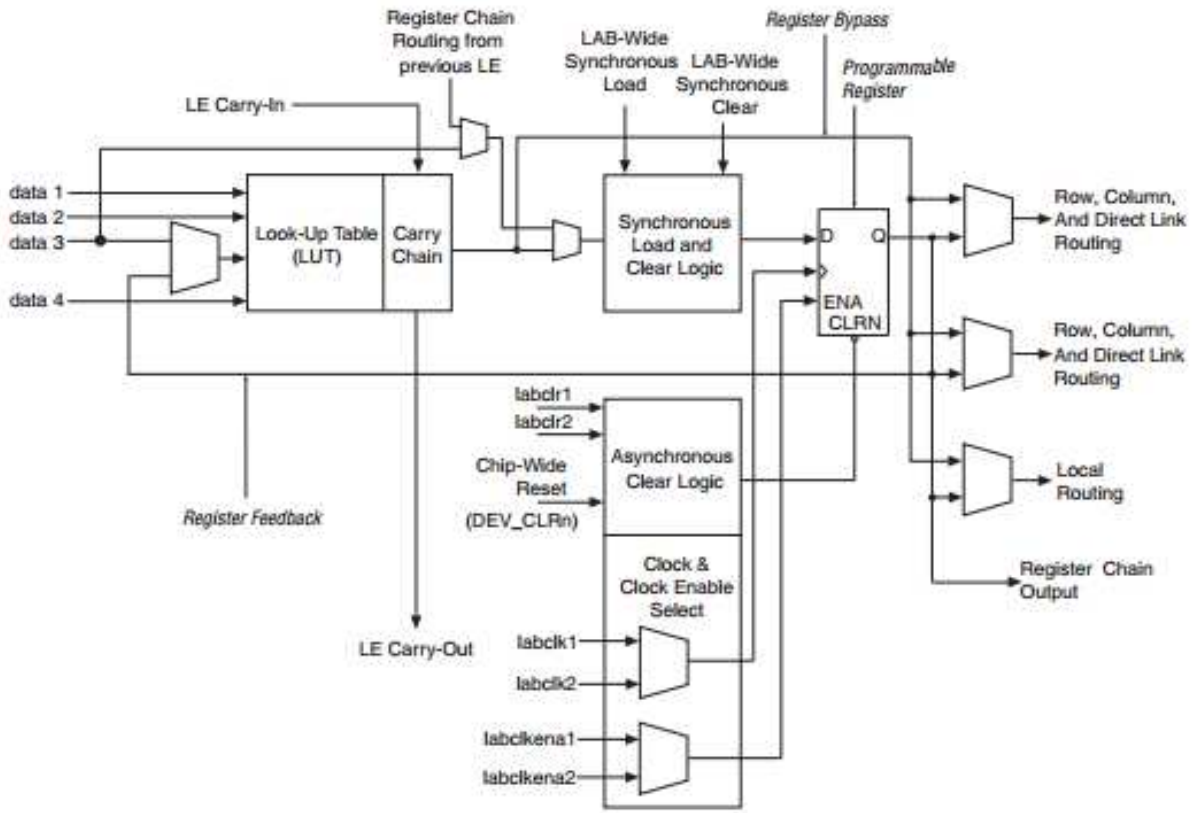
Логический блок FPGA



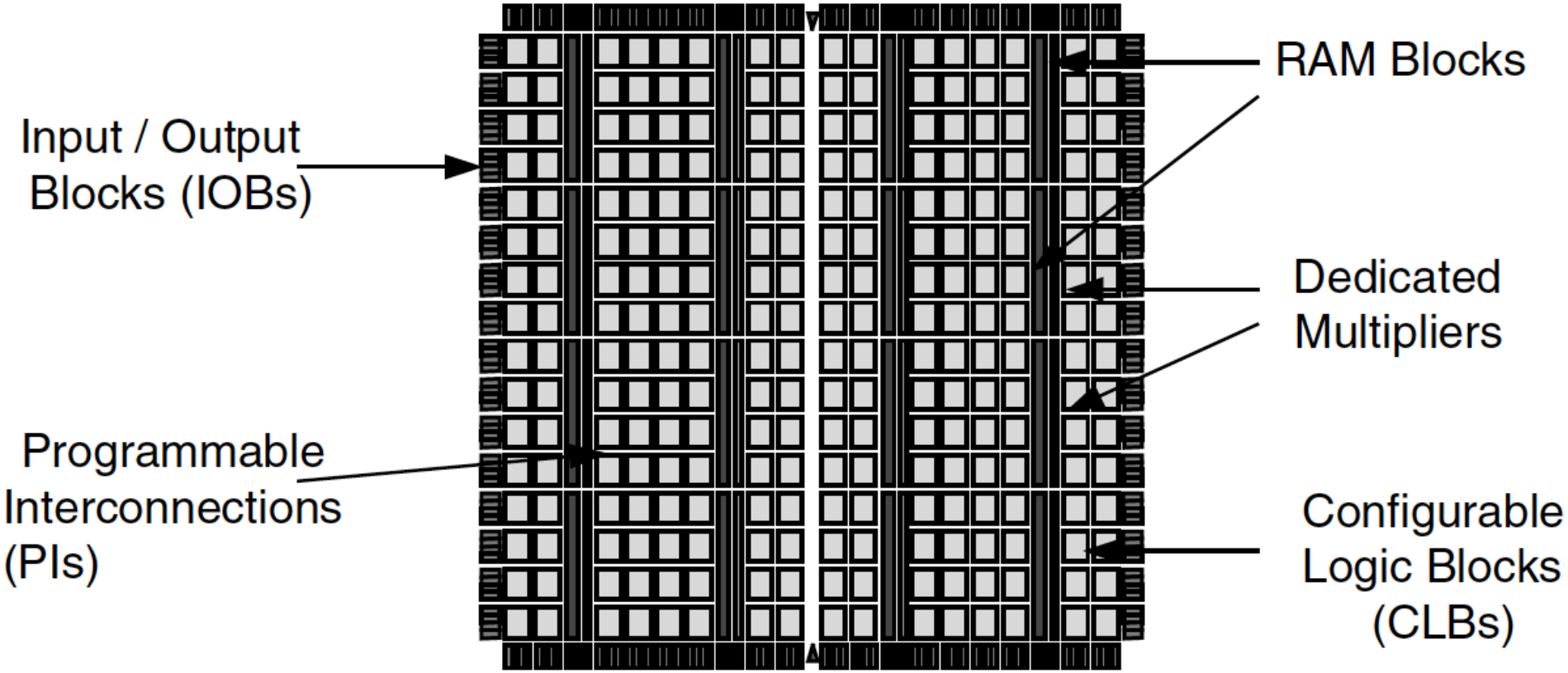
Базовый элемент FPGA – логическая ячейка, содержащая логический генератор и триггеры. Логический генератор FPGA – блок статической памяти, заполненной таблицей выходных значений. Для шести входов возможны 64 комбинации входных сигналов, таким образом, для реализации логического генератора требуется 64 бит памяти. Получается, что внутри FPGA расположены не базовые логические элементы И, ИЛИ, НЕ и т. п., а таблицы значений, с помощью которых можно имитировать любую комбинацию этих элементов. Выходы логических генераторов могут быть записаны в триггер, а могут быть и пущены в обход его, что позволяет при наличии достаточного количества логических ячеек-«кубиков» построить цифровую схему практически любой сложности.

Структура элементарной логической ячейки ALTERA Cyclone III

Figure 2–1. Cyclone III Device Family LEs



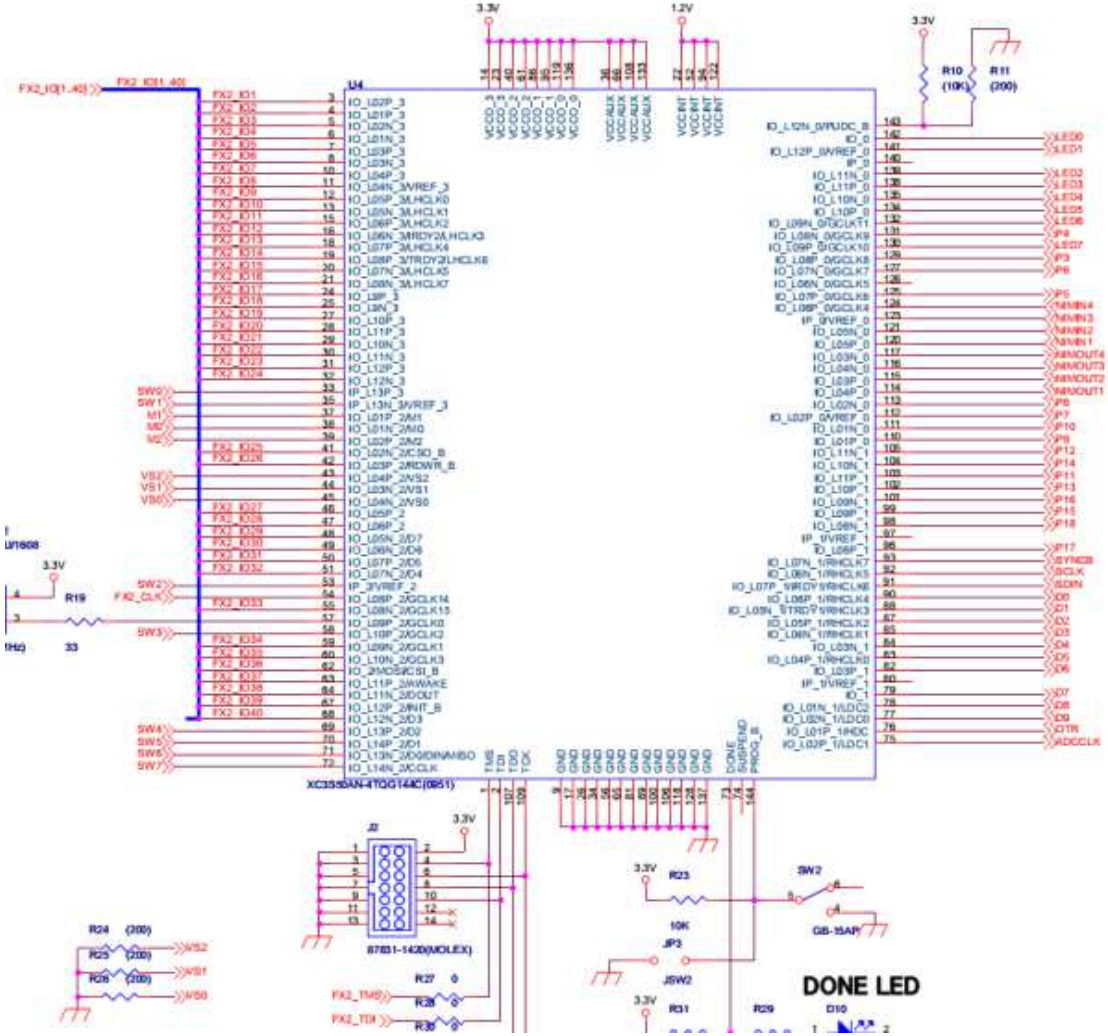
Архитектура FPGA



FPGA

- Configurable logic blocks (CLBs), containing combinational logic and register resources
- Input/output blocks (IOBs), interface between the FPGA and the outside world
- Programmable interconnections (PIs)
- RAM blocks
- Hardware multiplier
- Clock Manager
- Other resources: three-state buffers, global clock buffers, boundary scan logic, processors, SATA controller, RAM controller, PCI-Express controller, Ethernet controller, DSP blocks, and so on

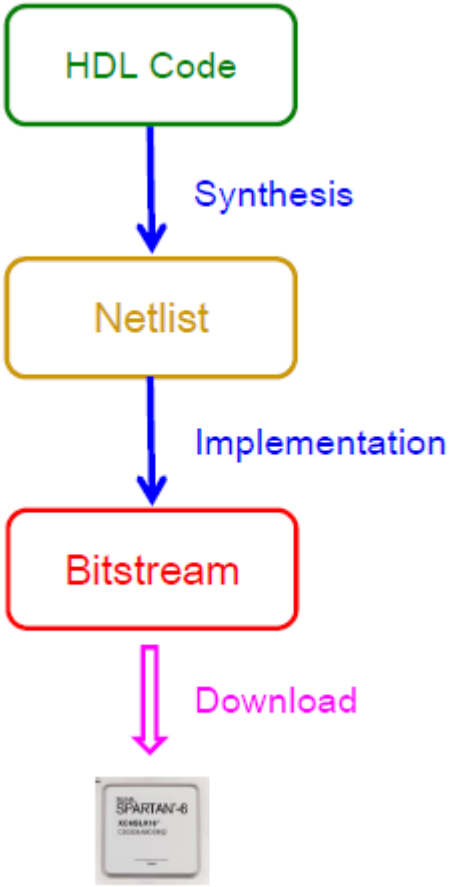
Принципиальная схема



Как конфигурировать FPGA

- Verilog
- VHDL
- MATLAB
- FlexRIO

- Altera: Quartus II
- Xilinx: ISE, Vivado



САПР и разработка HDL кода

Основные инструменты проектирования на сегодняшний день – это низкоуровневые языки описания аппаратуры (Hardware Description Language, HDL), описывающие не порядок действий при вычислениях, а список соединений компонентов (VHDL, Verilog).

Существуют библиотеки готовых компонентов – IP-ядер (Intellectual Property – "интеллектуальная собственность"), представляющих собой уже разработанные описания на HDL, сопровождаемые проектными ограничениями – указаниями по оптимальному размещению компонентов на кристалле. Ориентация на IP-ядра позволяет быстро собрать проект из «крупных кубиков» – цифровой фильтр или декодер требуют только настройки в графическом интерфейсе САПР, а не разработки с нуля.

Также используется пакет Matlab, который с помощью модулей, добавляемых САПР, способен сгенерировать HDL из блок-схемы.

Пример: Verilog

counter8.v

```
module counter8(
    input clk,
    input reset,
    output reg [7:0] q
    );

always @(posedge clk or posedge reset) begin
    if( reset ) begin
       q <= 0;
    end
    else begin
       q <= q + 1;
    end
end

endmodule
```

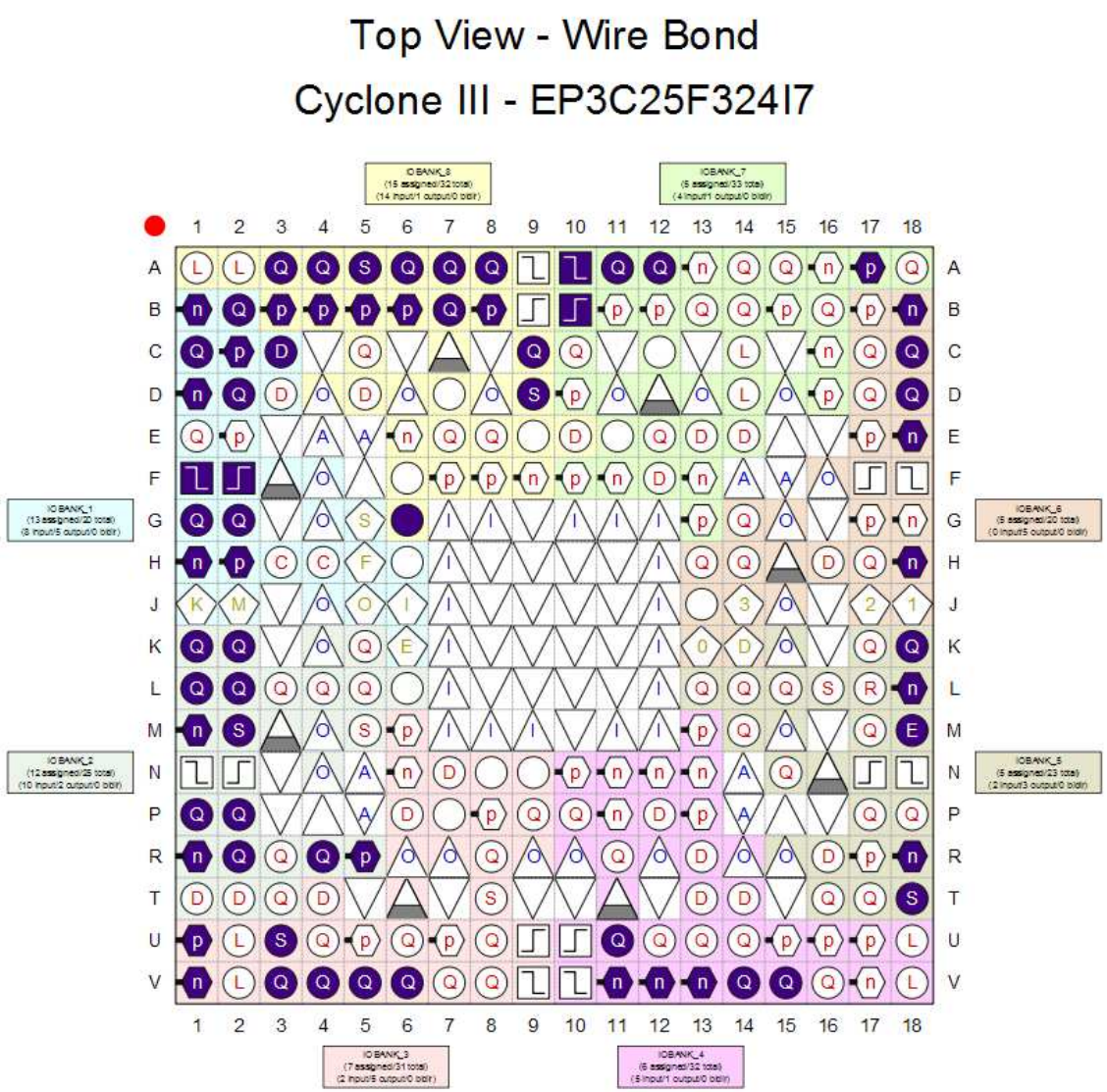
"q" is the register.
Need to declare as "reg".

Sensitivity list: the behavior below happenes only when the condition in @() satisfies.

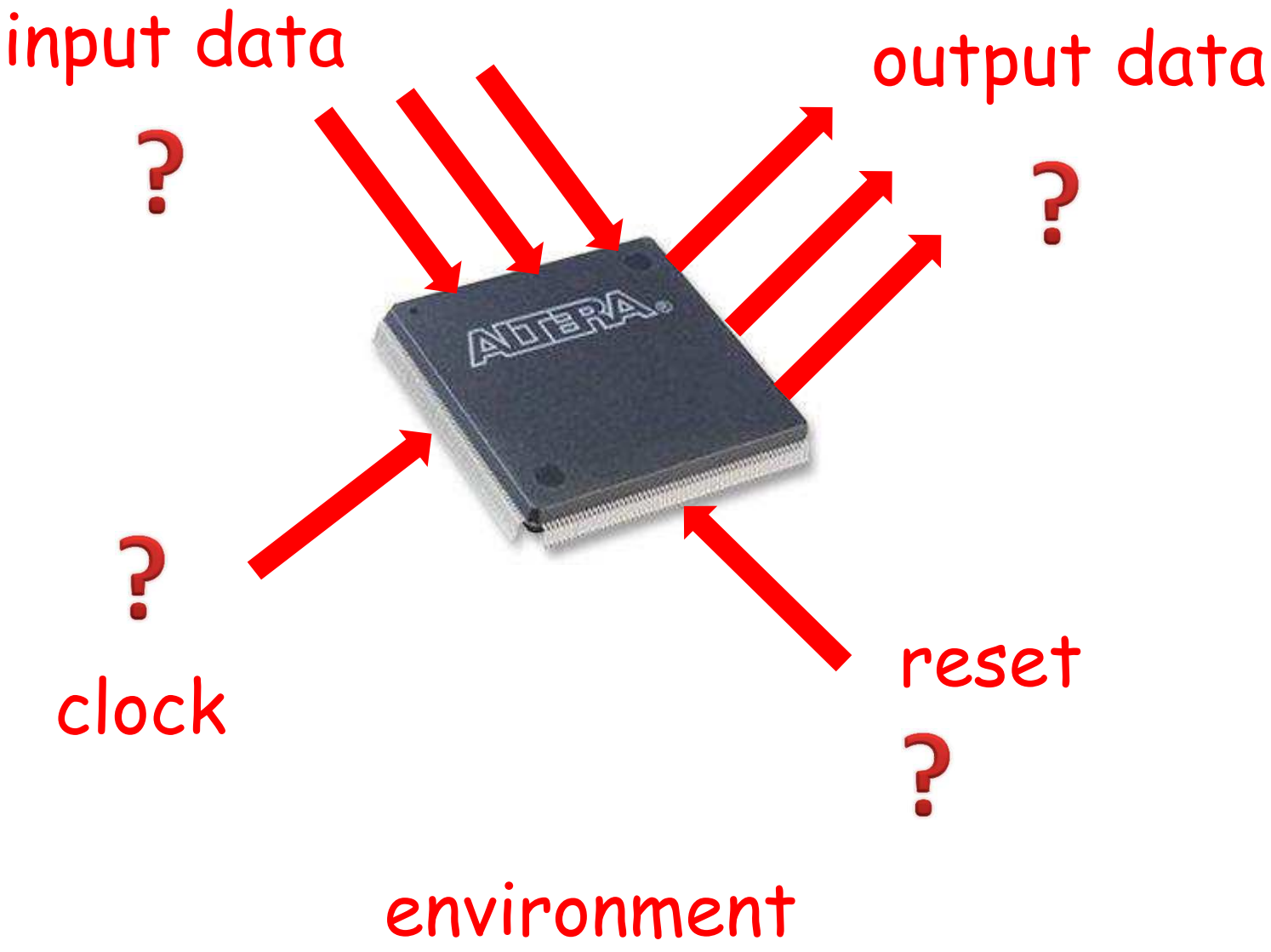
Asynchronous reset.
"q" is set to 0 when reset is high.

This happens at the positive edge of "clk".

Распределение сигналов по выходным ножкам



Testbench



Testbench

counter8.v

```
module counter8(
    input clk,
    input reset,
    output reg [7:0] q
    );

always @(posedge clk or
posedge reset) begin
    if( reset ) begin
        q <= 0;
    end
    else begin
        q <= q + 1;
    end
end

endmodule
```

sim_counter8.v

```
`timescale 1ns/1ns

module sim_counter8;

reg         clk, reset;
wire [7:0]  q;

parameter STEP = 20;

counter8 u0( clk, reset, q );

initial begin
    reset = 1'b1;
    #15 reset = 1'b0;
end

// continue to the next page
```

unit in simulation

counter8 module inside (like a function call in C)

describe the behavior of reset signal (high for 0-15ns; low later).

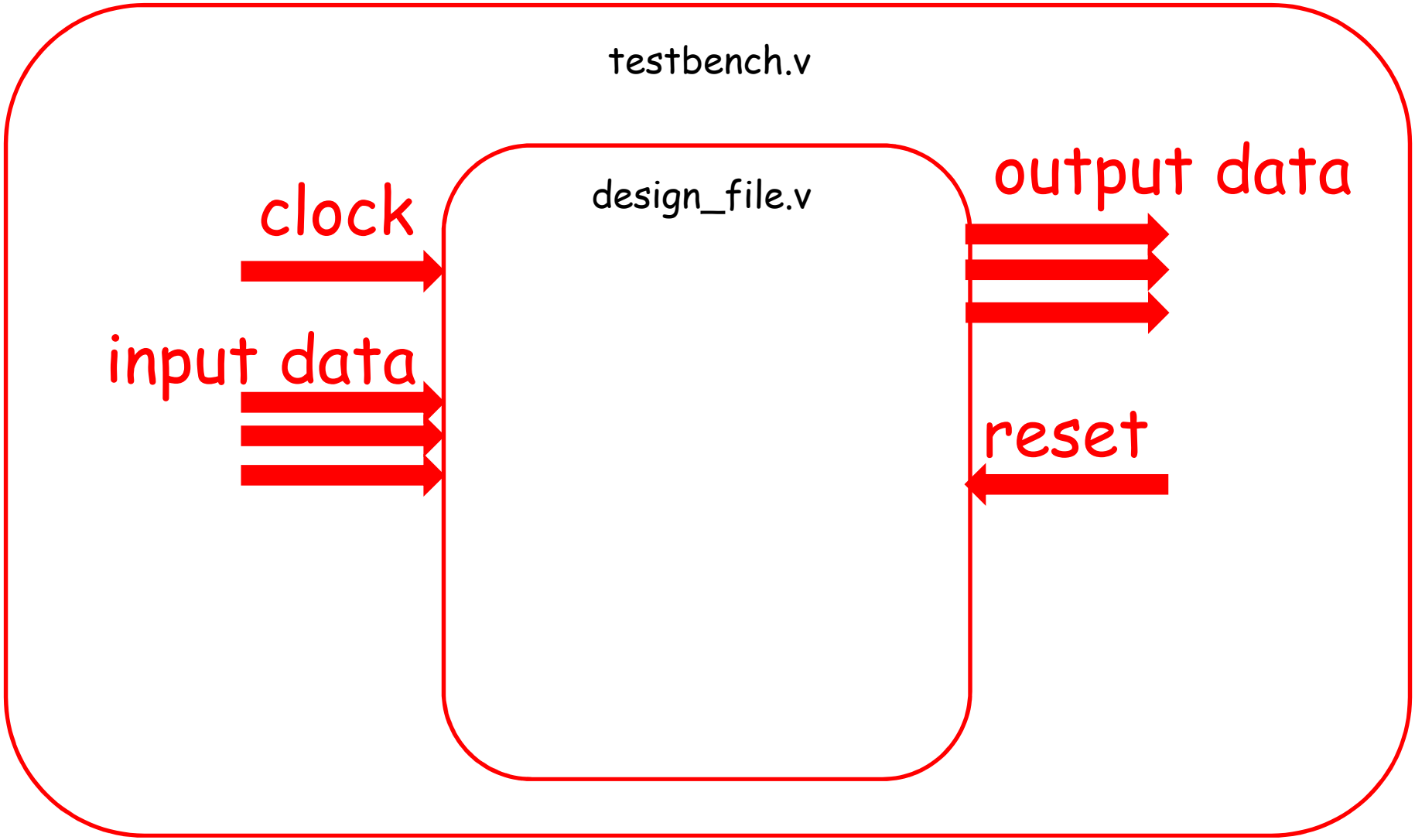
```
initial clk = 1'b0;
always #(STEP/2) clk = ~clk;

initial begin
    #(30*STEP) $stop;
end

endmodule
```

A little bit more complicated:

- clk = 0 (low) at t = 0.
- At t = 10ns (= STEP/2), clk is flipped (i.e. clk = 1).
- At t = 20ns, clk is flipped again ("always" means the behavior inside is repeated).

Stop at 600ns.

Testbench


testbench.v
design_file.v
clock
input data
output data
reset