Лекция 1
Программируемые логические интегральные схемы

Осень 2016

План лекции

- Программируемые логические интегральные схемы (ПЛИС).
- Устройство ПЛИС DE0-Nano компании Altera.
- Уровни абстракции при проектировании ПЛИС
- Маршрут проектирования ПЛИС
- Языки описания схем
- Базовые концепции языка Verilog

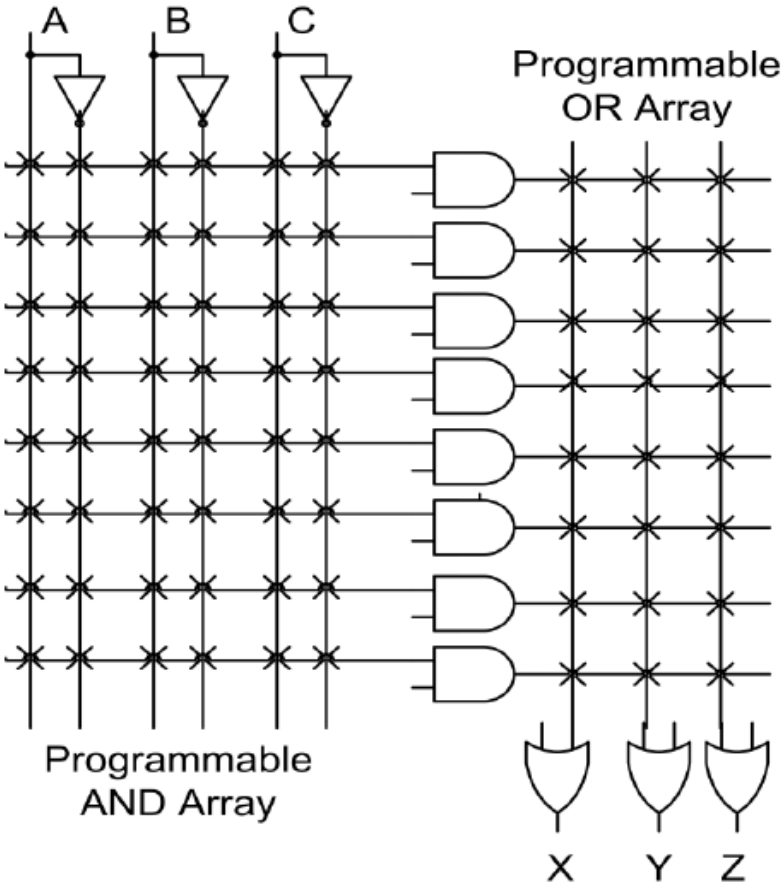
Программируемая логическая интегральная схема

- Программируемая интегральная схема (ПЛИС, англ. Programmable logic device, PLD) – электронный компонент, используемый для создания цифровых интегральных схем, логика работы которого не определяется при изготовлении, а задаётся посредством программирования (проектирования).

- Для программирования используются программатор и IDE (среда разработки), позволяющие задать желаемую структуру цифрового устройства в виде принципиальной электрической схемы или набора модулей на специальных языках описания аппаратуры: Verilog, VHDL, и др.

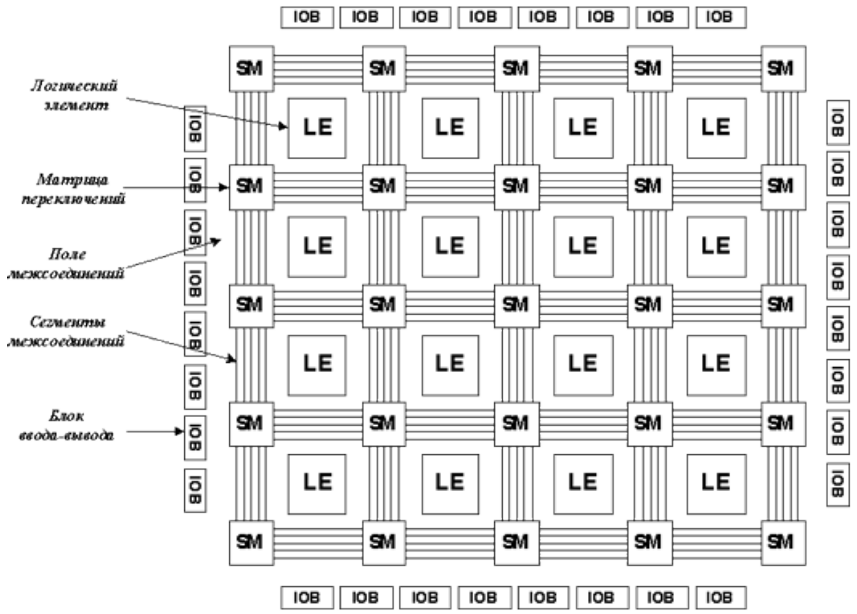
Классификация ПЛИС

- Основные типы ПЛИС:
  - Programmable array logic (PAL)
  - Gate array logic (GAL)
  - Complex programmable logic device (CPLD)
  - Field-programmable gate array (FPGA)
- ПЛИС различают по технологии производства и возможности перепрограммирования

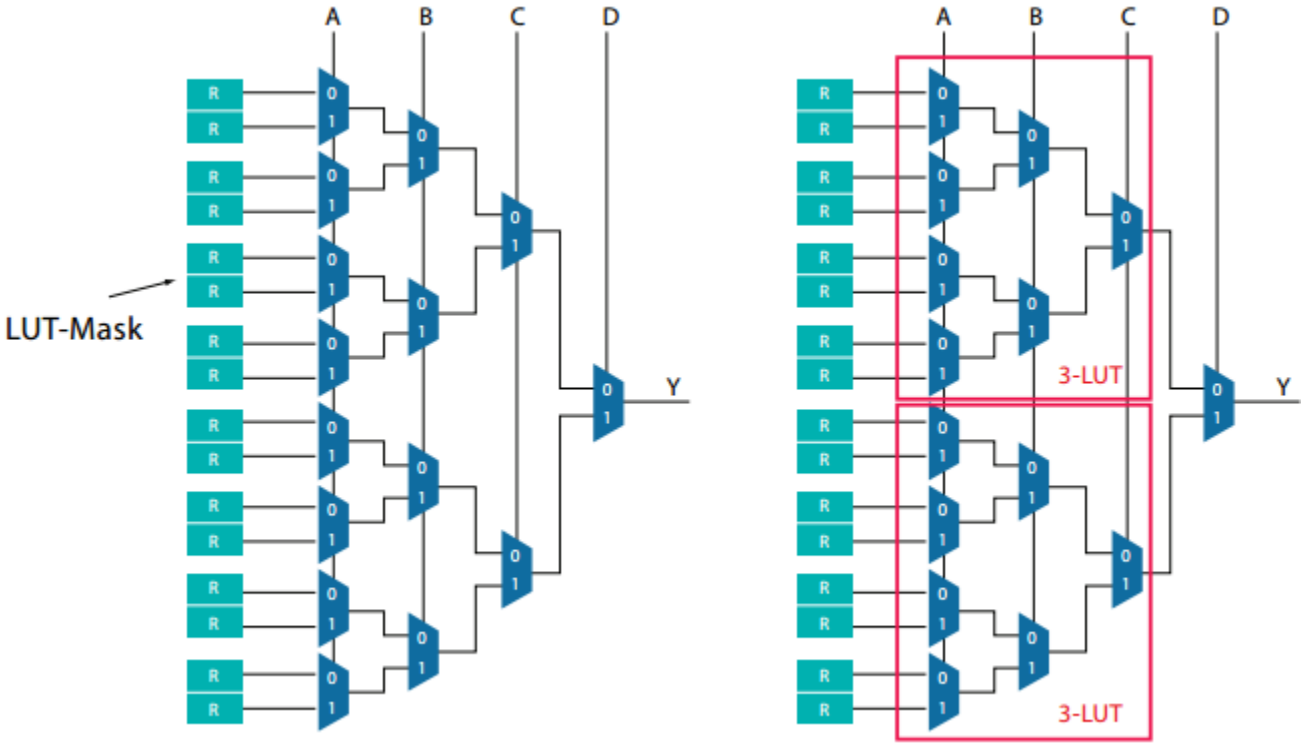


Матричные ПЛИС (FPGA)

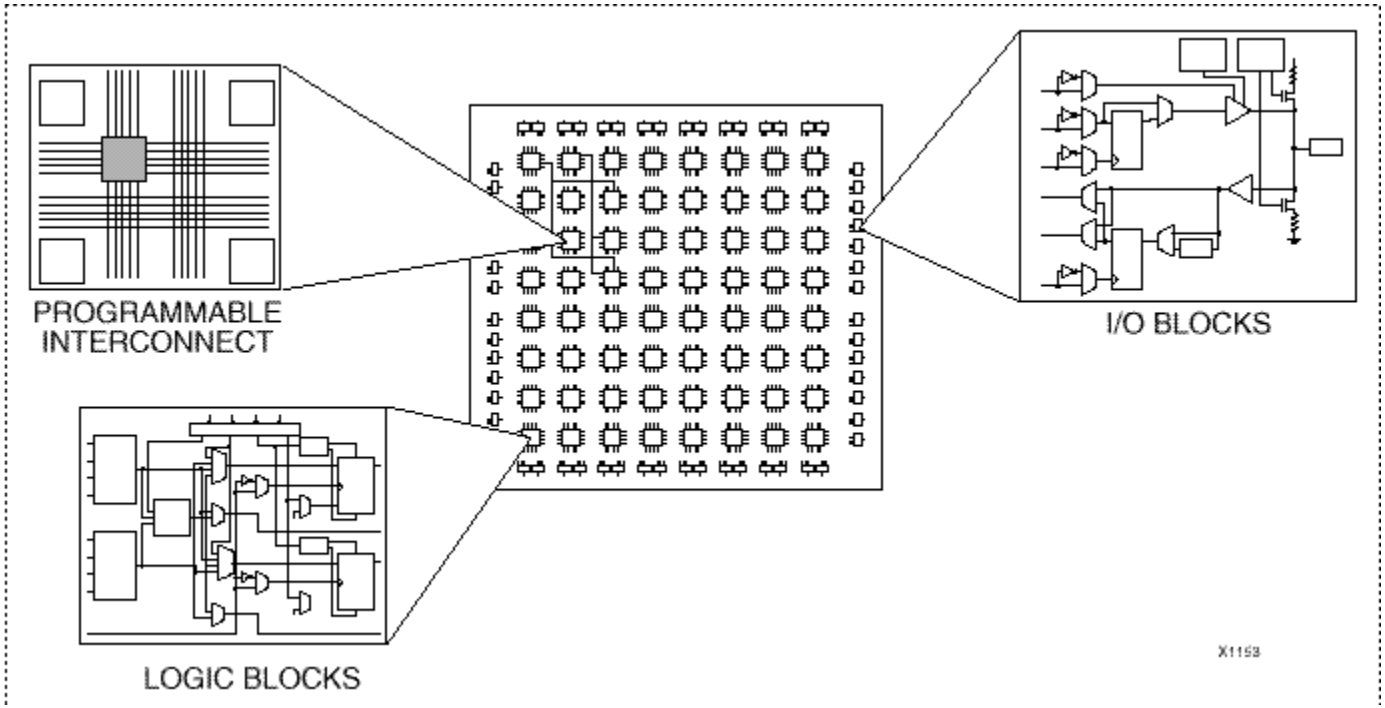
- Программируемая пользователем вентильная матрица (FPGA) – специальный вид ПЛИС, который может быть перепрограммирован пользователем.
- Основные программируемые компоненты:
  - программируемые логические блоки;
  - блоки ввода и вывода;
  - Коммутация между элементами.



Программируемые логические блоки

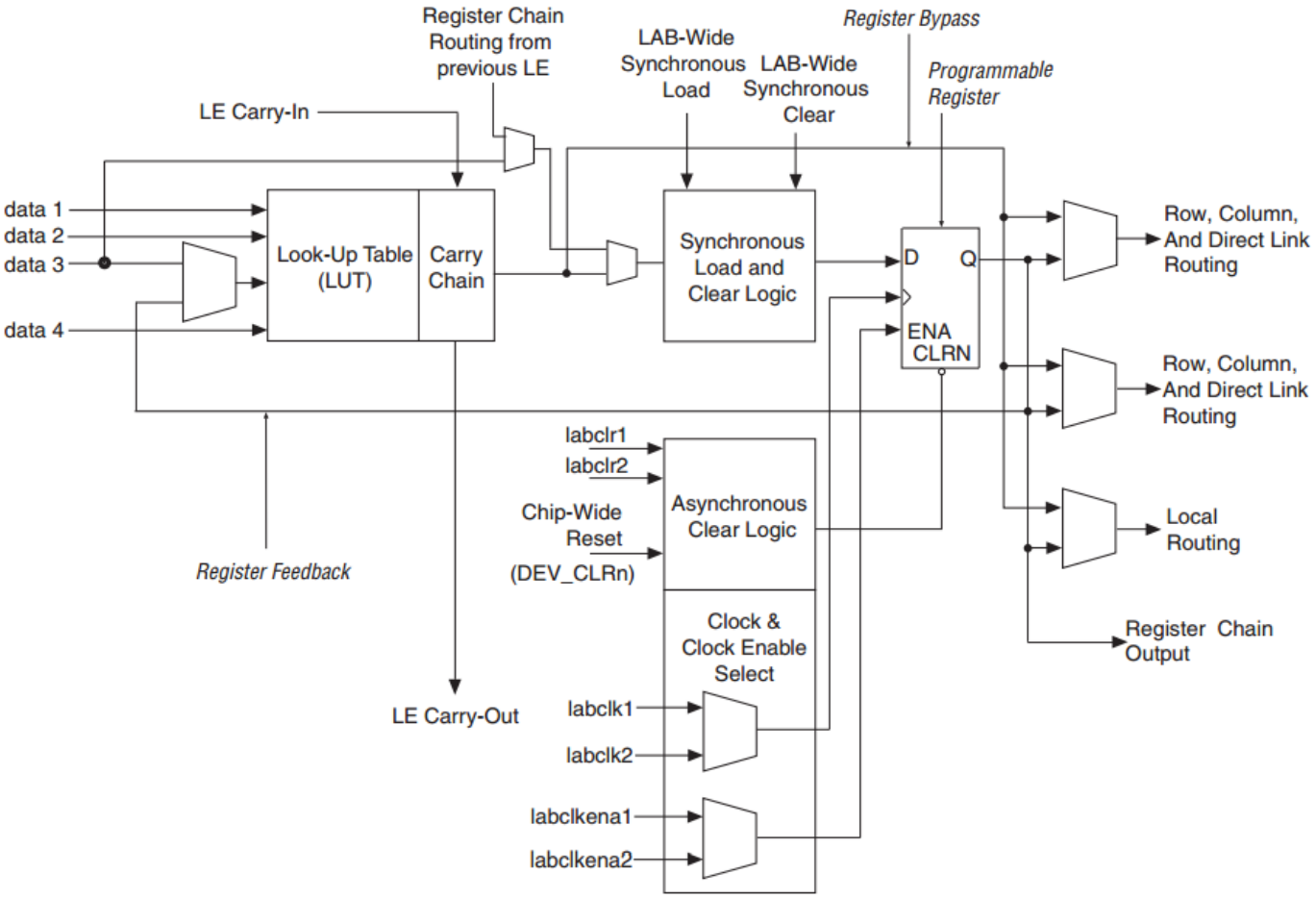


Топология матричных ПЛИС

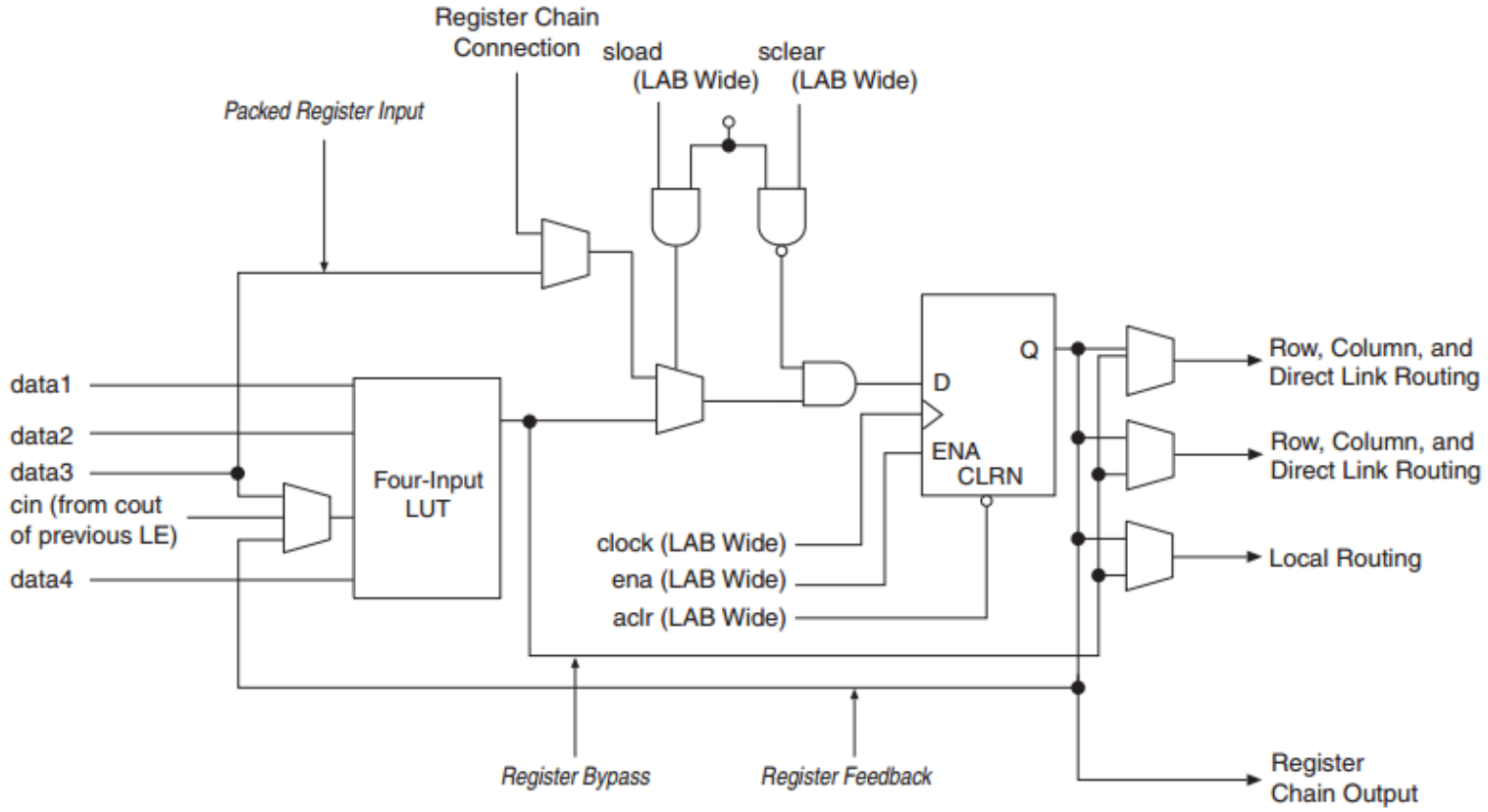


Устройство ПЛИС DE0-Nano

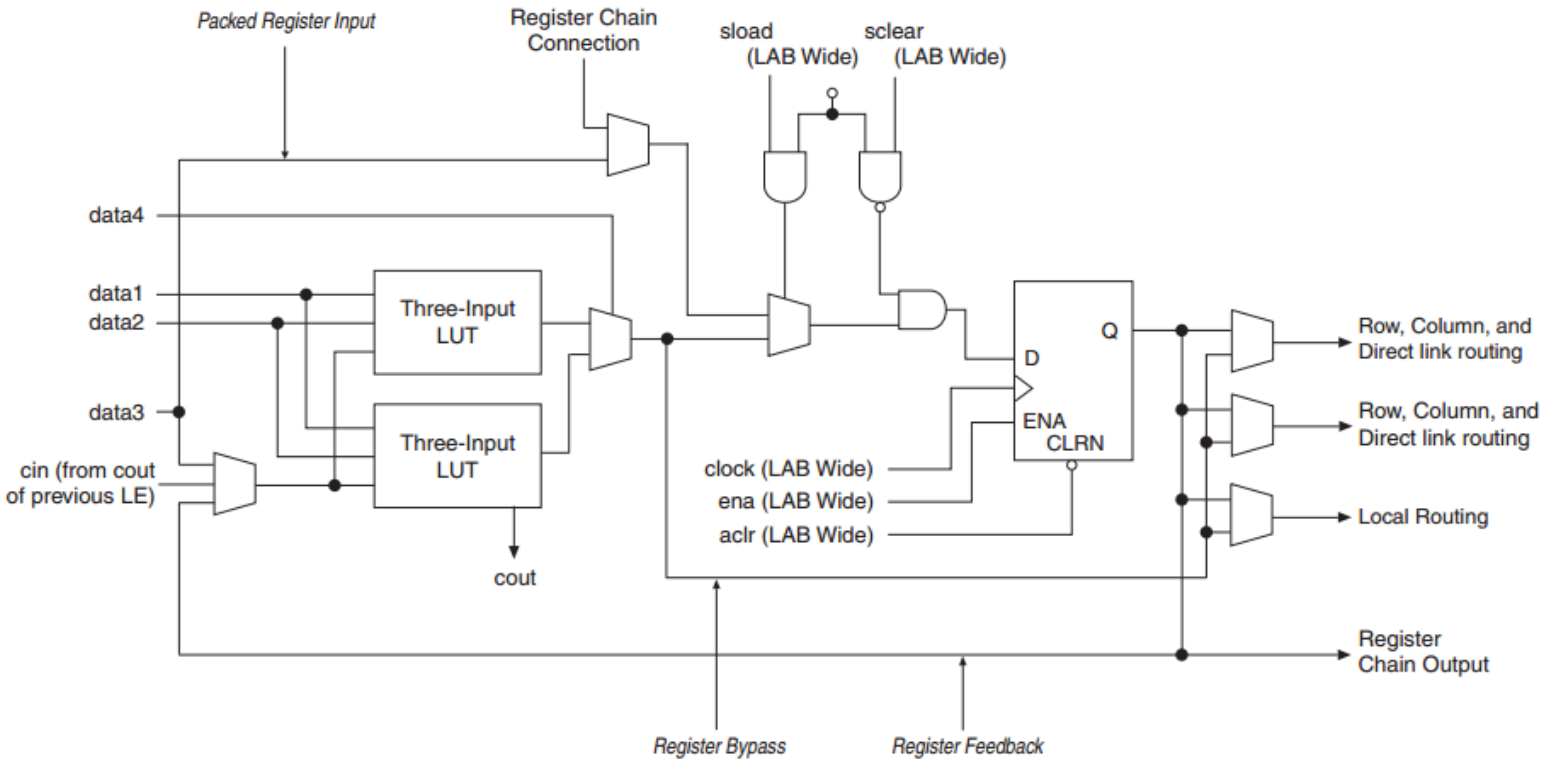
Устройство логических блоков



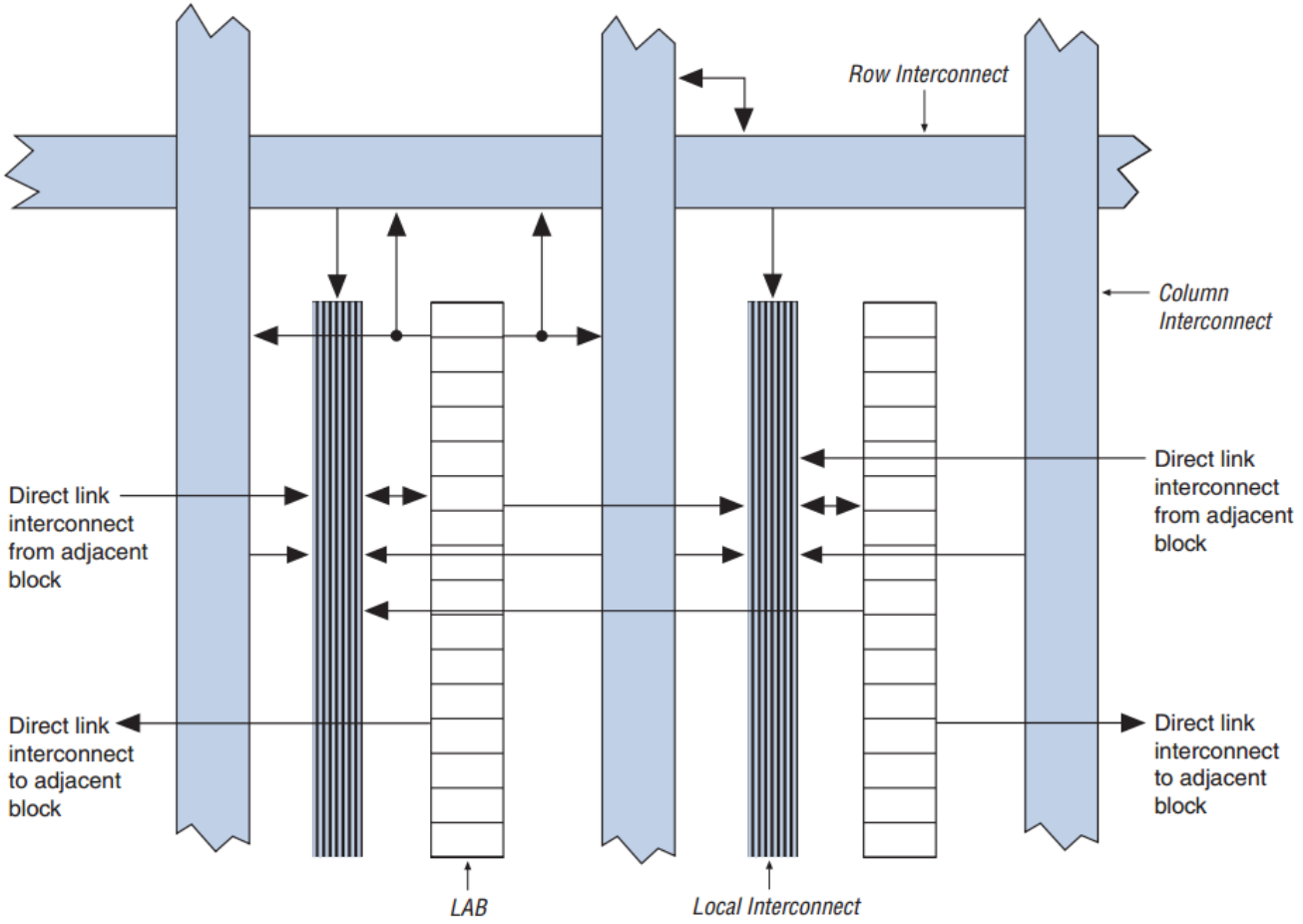
Логический блок в комбинационном режиме



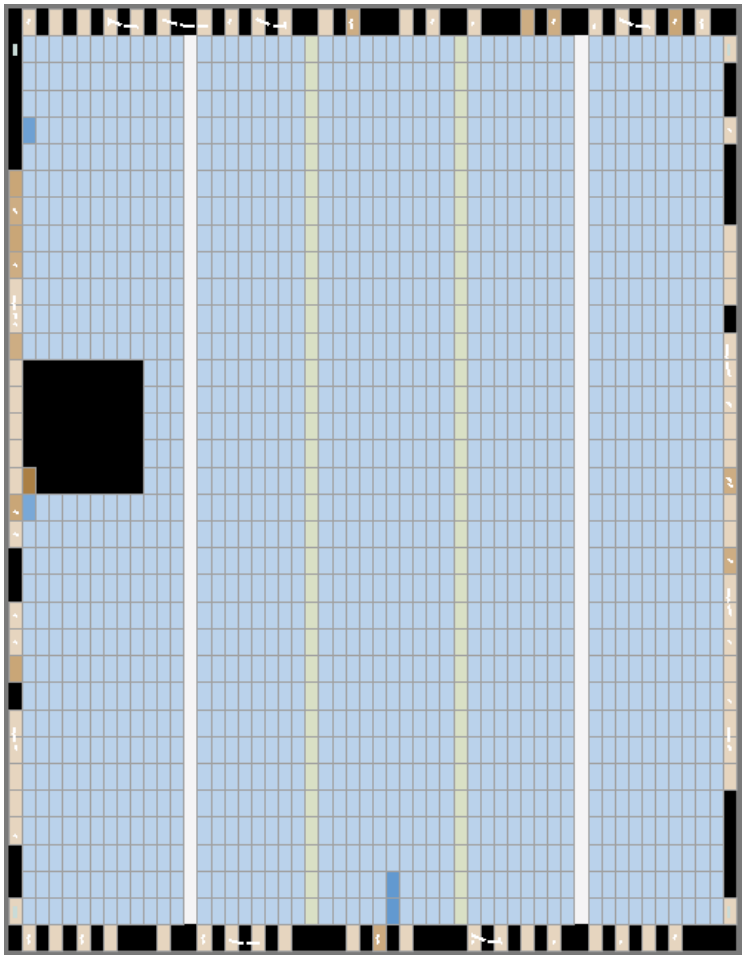
Логический блок в арифметическом режиме



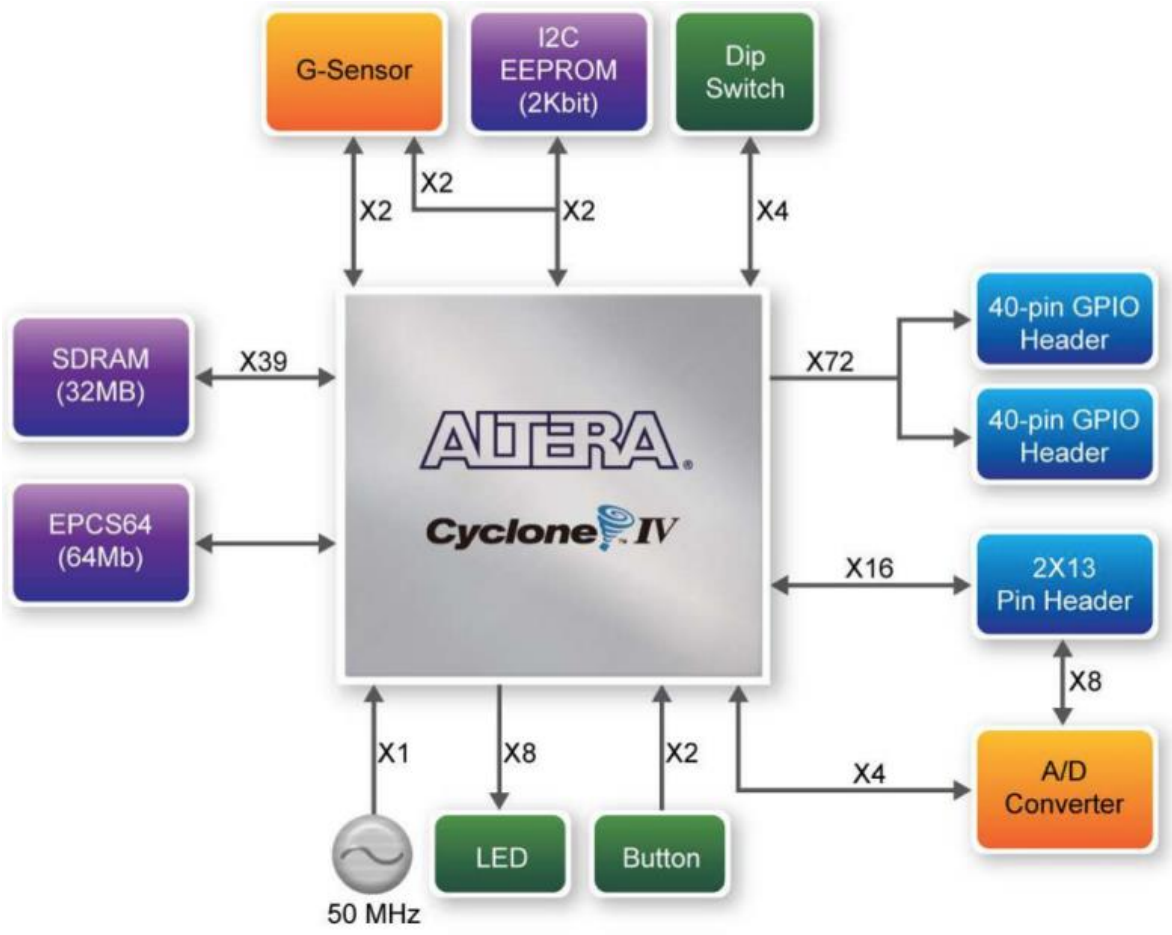
Массивы логических блоков и трассировочные ресурсы ПЛИС



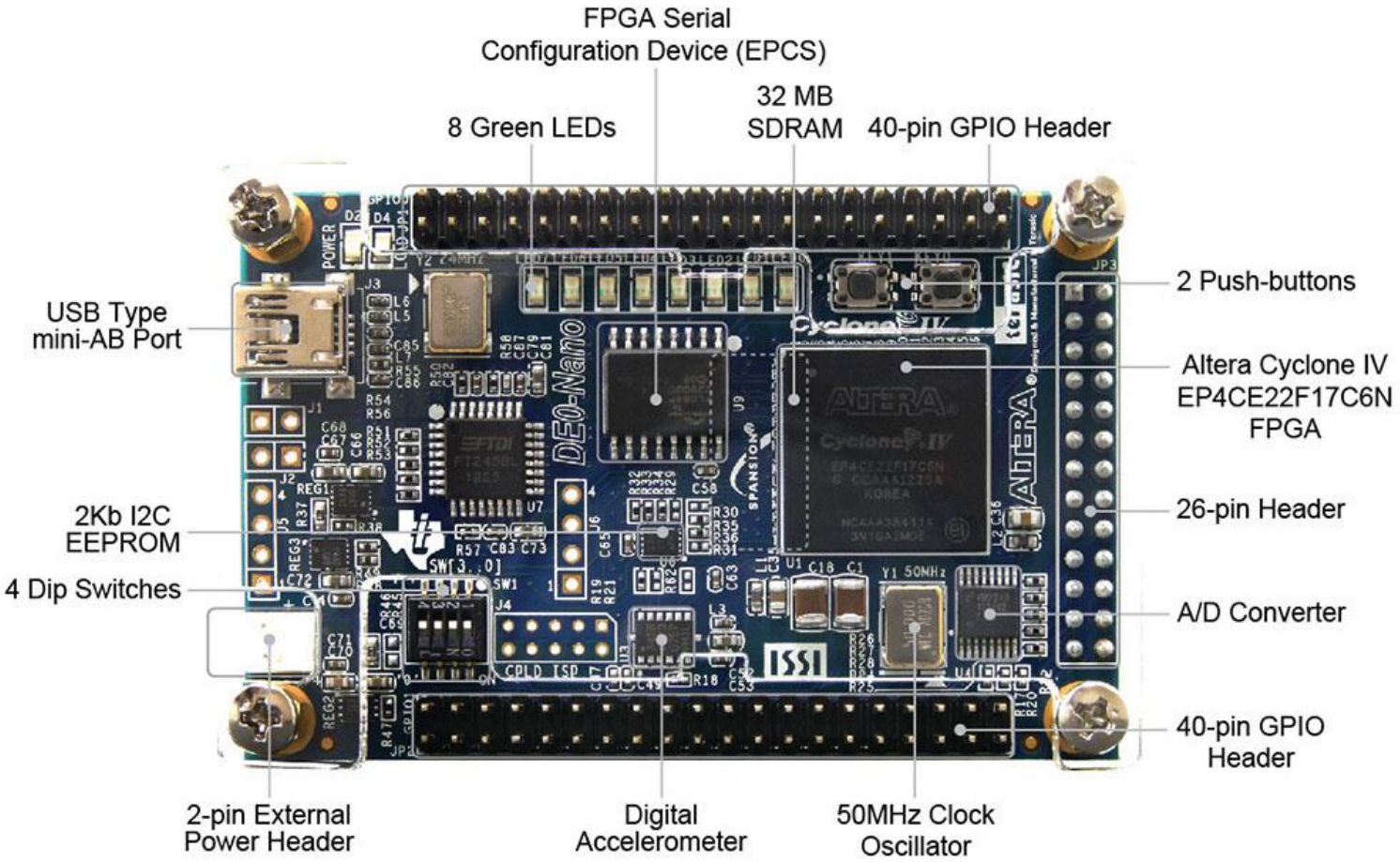
Структура блоков DE0-Nano



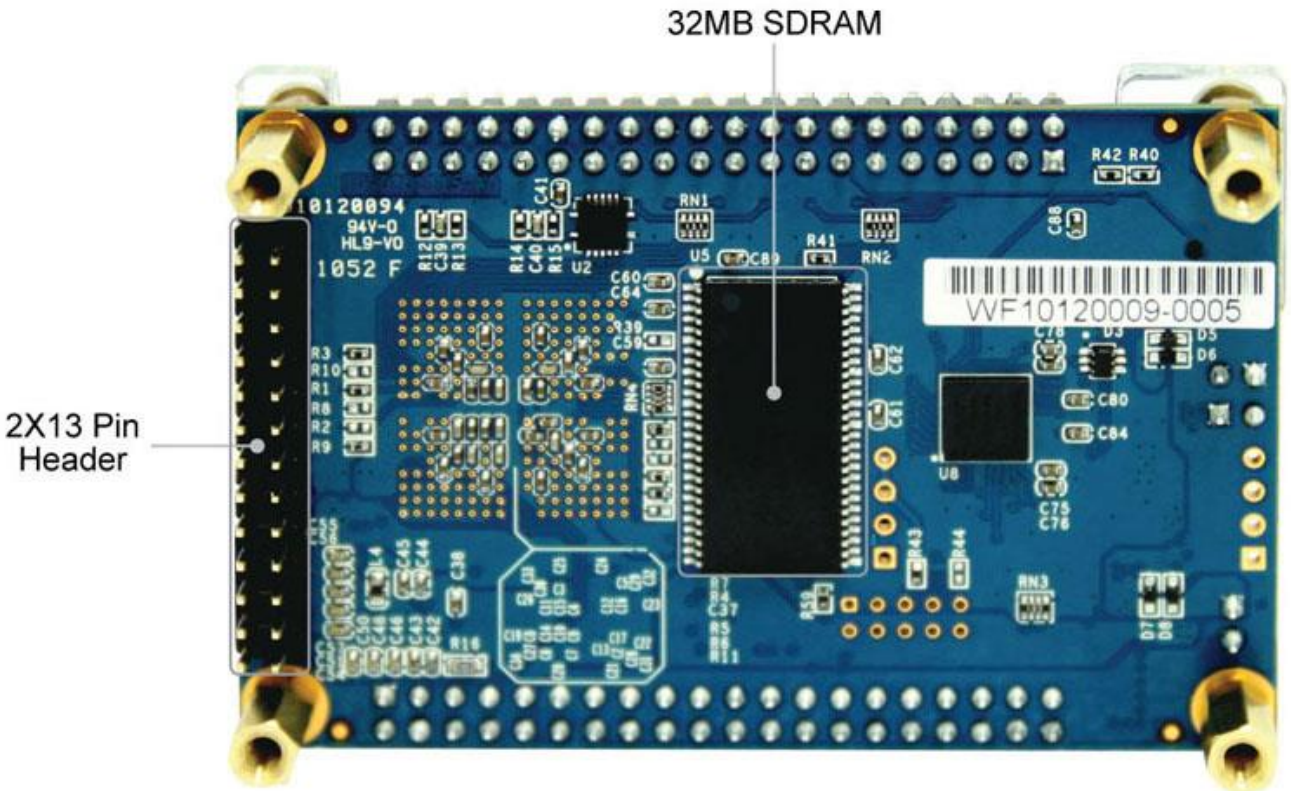
Периферия DE0-Nano



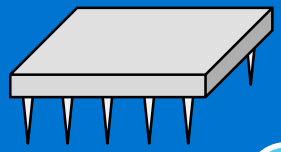
Периферия DE0-Nano



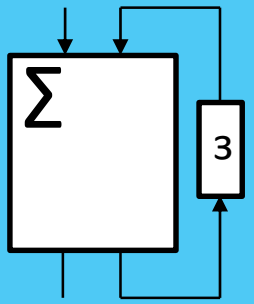
Периферия DE0-Nano



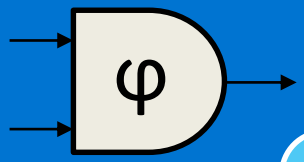
Уровни абстракции при проектировании цифровых СБИС



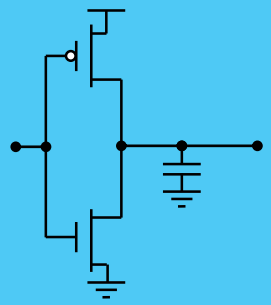
Системный уровень



Автоматный (поведенческий) уровень

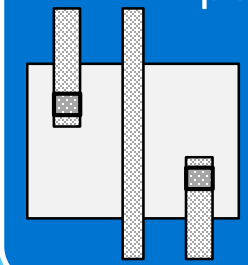


Логический (схемный) уровень

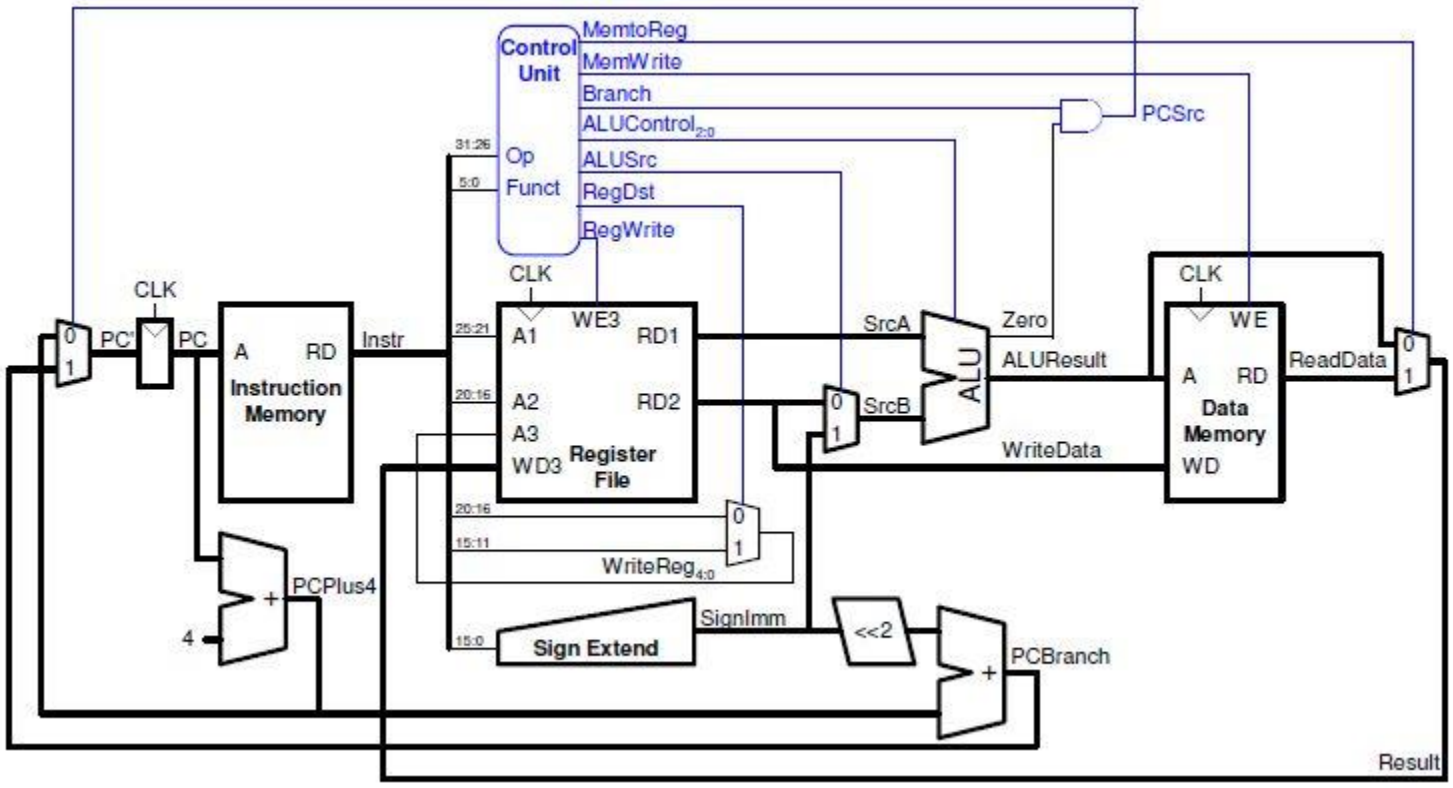


Транзисторный уровень

Уровень топологии



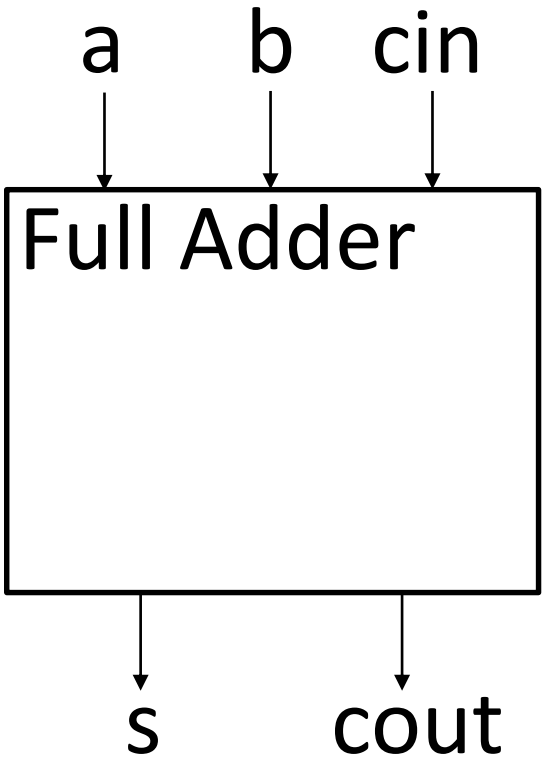
Системный уровень



Системный уровень

- Моделирование системы взаимодействующих процессов/сигналов
- Подходы к моделированию:
  - взаимодействие систем/компонент системы
  - система команд(instruction set simulation)
  - микроархитектура
  - использование языков описания аппаратуры (Verilog, SystemVerilog, SystemC)

Поведенческий уровень

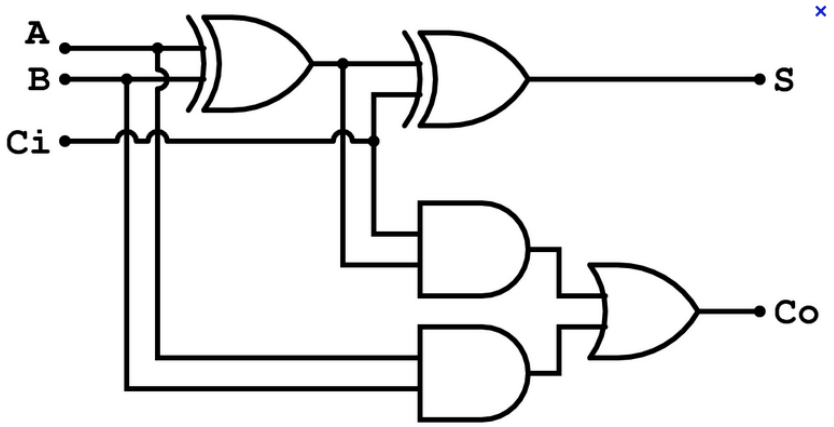



```
`timescale 1ns / 1ps
module FullAdder (
        input a,
        input b,
        input cin,
        output s,
        output cout );

      assign {cout,s} = a + b + cin;

endmodule
```

Поведенческий уровень

- Моделирование поведения/функционирования процесса/сигнала
- Подходы к моделированию:
  - использование языков описания аппаратуры (Verilog, VHDL)
  - register-transfer level (RTL)
  - использование автоматов и других математических моделей

Логический (схемный) уровень

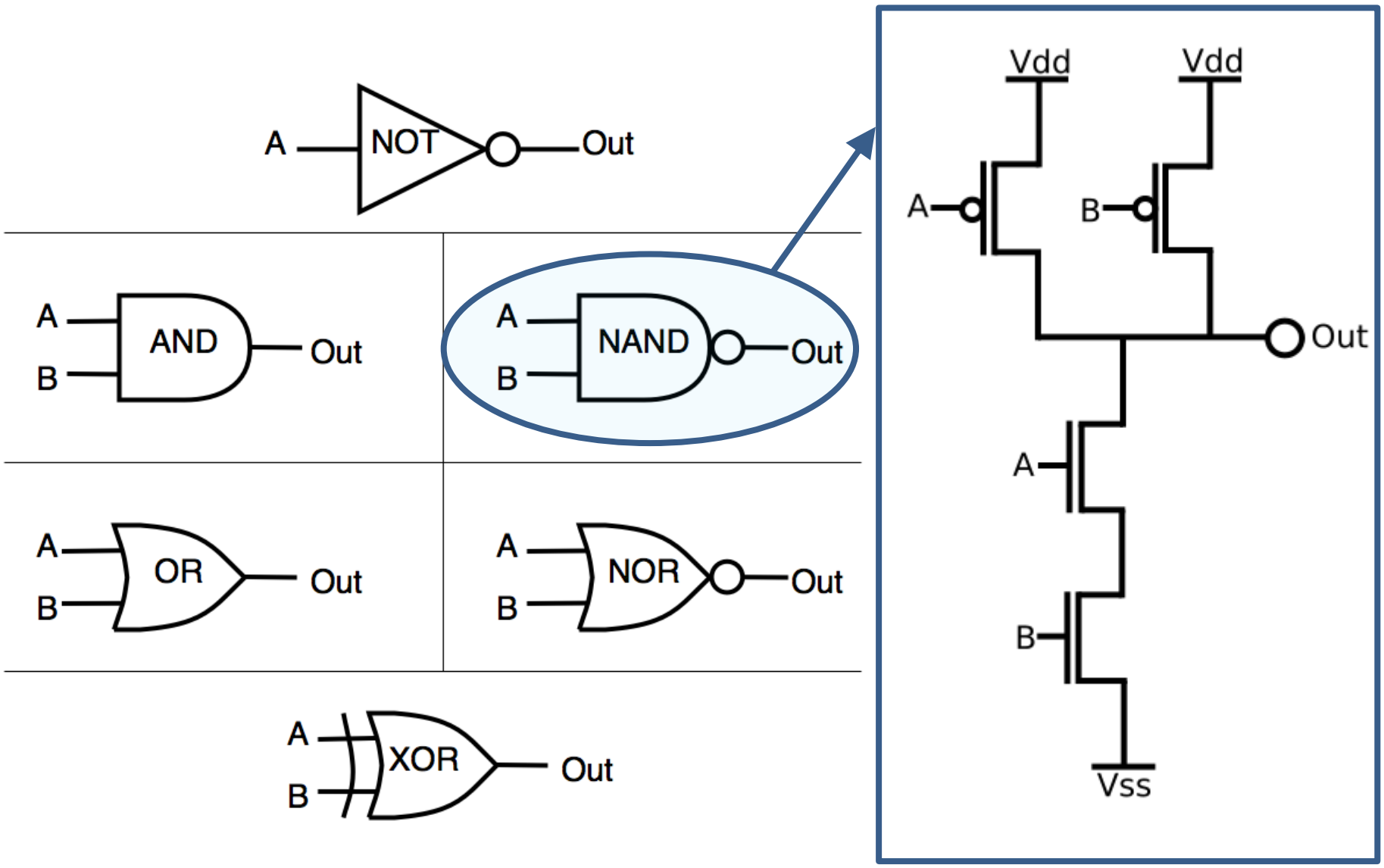



```
`timescale 1ns / 1ps
module FullAdder (
        input a,
        input b,
        input cin,
        output s,
        output cout );

        // wires (from ands to or)
        wire w1, w2, w3;

        // carry-out circuitry
        and( w1, a, b );
        and( w2, a, cin );
        and( w3, b, cin );
        or( cout, w1, w2, w3 );

        // sum
        xor( s, a, b, cin );

endmodule
```

Логический (схемный) уровень

- Моделирование структуры и основных элементов блока, реализующего заданный процесс/сигнал
- Подходы к моделированию:
  - использование языков описания аппаратуры (Verilog, VHDL)
  - netlist, gate-level design
  - математические модели схем
    - схемы из функциональных элементов (СФЭ) и их обобщения
    - And-Inverter Graphs (AIG)
    - Binary Decision Diagrams (BDD)
    - и др.

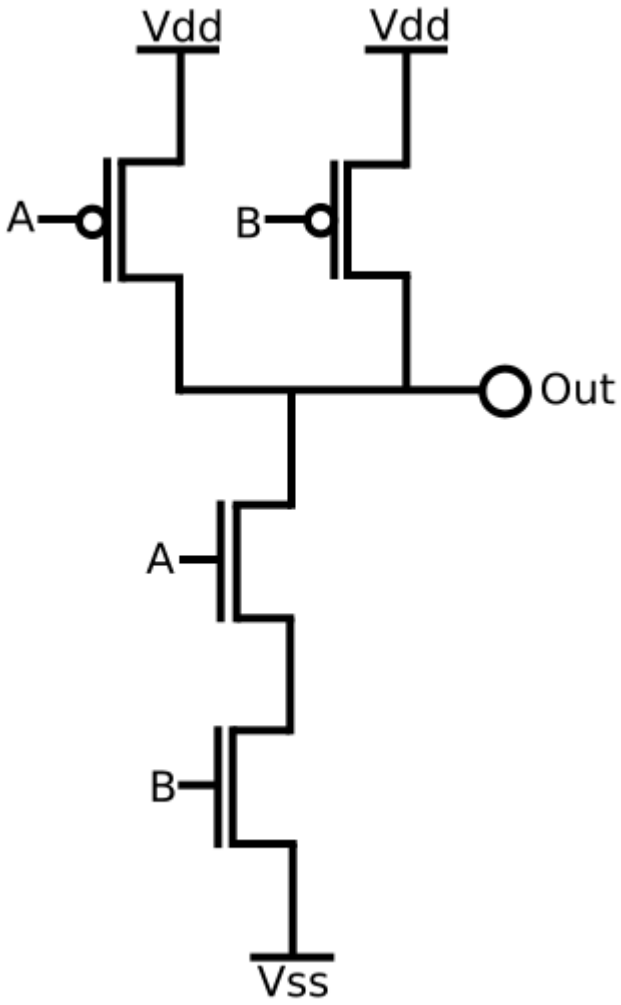
Транзисторный уровень

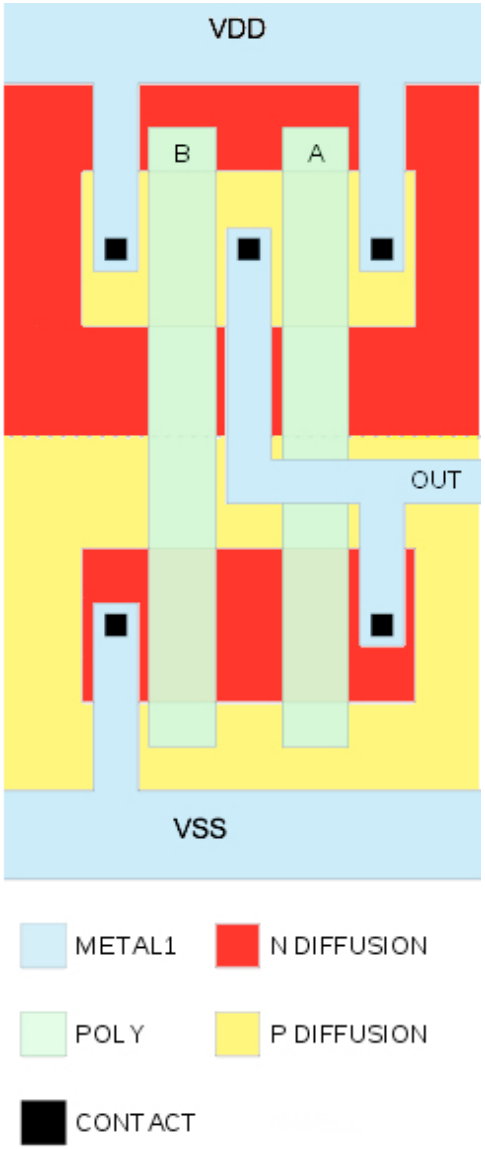


Транзисторный уровень

- Моделирование структуры основных логических элементов интегральной схемы
- Определение/оценка основных физических характеристик логических элементов (размер, задержка, энергопотребление и др.)
- Подходы к моделированию:
  - использование различных транзисторных моделей схем
  - имитационной моделирование
  - SPICE моделирование

Уровень топологии

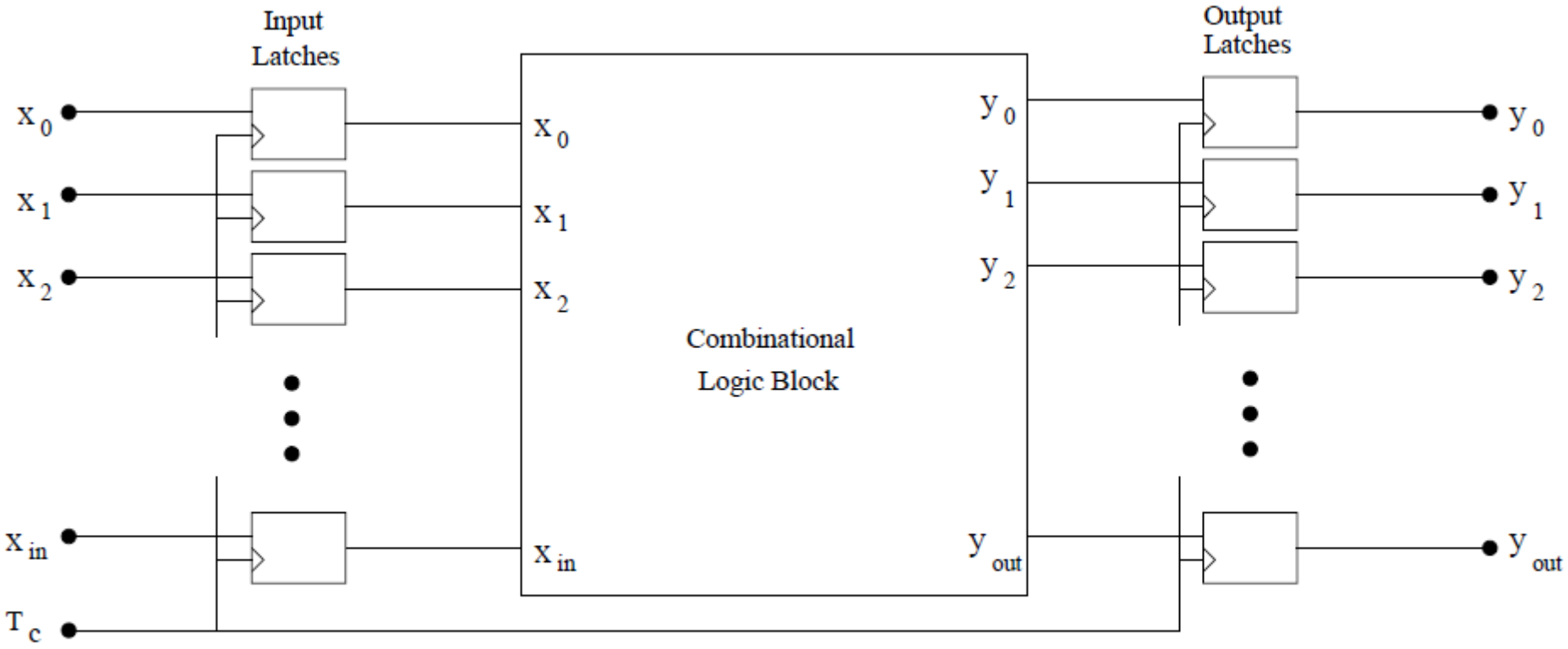




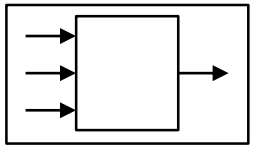
Уровень топологии

- Моделирование топологии (структуры и геометрии всех слоев) проектируемого устройства
- Основные задачи:
  - Design Rule Check (DRC)
  - Layout vs Schematics (LVS)
  - Оптимизация топологии и повышение выхода годных
    - Optical Proximity Correction(OPC)
    - Double/Triple patterning

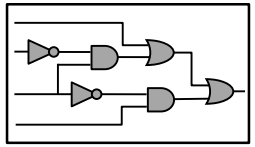
Комбинационная и последовательная логика

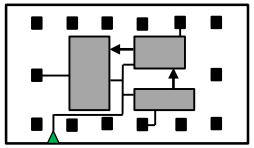


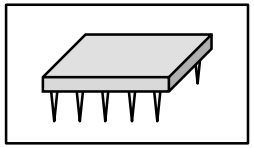
Упрощенный маршрут проектирования



```
ENTITY test is
port a: in bit;
end ENTITY test;
```







Спецификация системы

↓

Проектирование архитектуры

↓

Функциональное проектирование

↓

Логическое проектирование

↓

Физическое проектирование

↓

Программирование ПЛИС

Языки описания аппаратного обеспечения

- Hardware Description Language (HDL) – язык описания структуры и функционирования аппаратного обеспечения цифровой системы на различных уровнях абстракции.
- Основные HDL:
  - Verilog
  - SystemVerilog
  - VHDL
  - SystemC
  - …
- http://en.wikipedia.org/wiki/Hardware_description_language

Стили разработки аппаратного обеспечения

- Снизу-вверх
  - Сначала отдельно проектируются базовые элементы, а потом из простых элементов проектируются (собираются) более сложные элементы.
- Сверху-вниз
  - Сначала проектируются системы более высокого уровня, а потом рекурсивно проектируются их компоненты более низкого уровня.

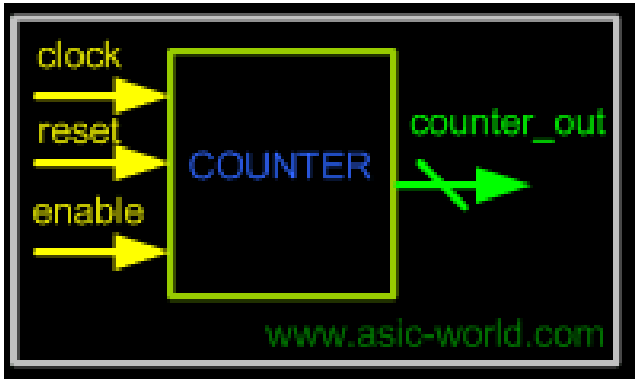
Основные уровни абстракции аппаратного обеспечения

- Поведенческий уровень (Behavioral Level)
  - Описание устройства при помощи набора последовательных алгоритмов, работающих одновременно и согласованно.
- Уровень регистровых передач (Register-Transfer Level)
  - Описание устройства при помощи заданного набора операций преобразования значений, хранящихся в регистрах. По сути устройство описывается в виде иерархии взаимосвязанных автоматов.
- Схемный (функциональный) уровень (Gate Level)
  - Описание устройства при помощи схемы из функциональных элементов в заданной библиотеке элементов.

Язык Verilog – Hello world!

```
1  module hello_world;
2    initial begin
3      $display ("Hello world!");
4      #10 $finish;
5    end
6  endmodule
```

Пример моделирования на языке Verilog – 4-х битовый счетчик



- Входы
  - Clock – вход тактового генератора
  - Reset – сброс значения счетчика (сброс при «1»)
  - Enable – включение счетчика (при «1» счетчик работает)
- Выходы
  - Counter_out – значение счетчика
- Функция
  - Счетчик увеличивается на 1-цу за каждый «такт» тактового генератора

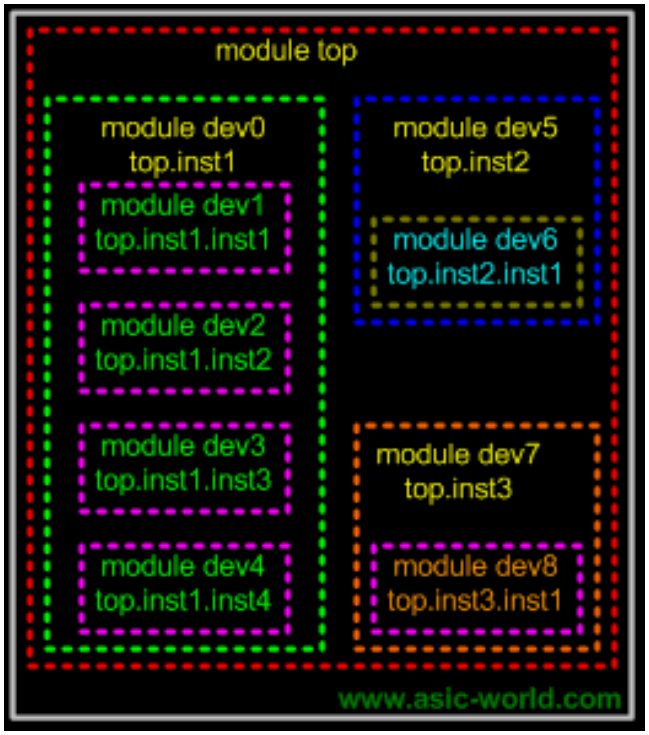
Пример моделирования на языке Verilog – 4-х битовый счетчик

```
1  module counter (clock, reset, enable, counter_out);
2
3    input clock;
4    input reset;
5    input enable;
6
7    output [3:0] counter_out;
8
9    wire clock;
10   wire reset;
11   wire enable;
12
13   reg [3:0] counter_out;
14
```

Пример моделирования на языке Verilog – 4-х битовый счетчик

```
15    always @ (posedge clock)
16      begin: COUNTER
17        if (reset == 1'b1) begin
18          counter_out <= #1 4'b0000;
19        end
20        else if (enable == 1'b1) begin
21          counter_out <= #1 counter_out + 1;
22        end
23      end
24  endmodule
```

Язык Verilog - модули



- Основные строительные блоки аппаратного описания на языке Verilog
- Описание представляет собой вложенную иерархию модулей
- Инстанцирование модулей

Язык Verilog – объявление и инстанцирование модулей

```
1  module device(device_output, input_1, input_2);
2    ...
3  endmodule
4
5  module another_device();
6    wire wire_1, wire_2;
7    reg register;
8    ...
9    device device_insance(register, wire_1, wire_2);
10   ...
11 endmodule
```

Язык Verilog - сети

- Сети позволяют хранить и передавать сигналы между модулями.
- Основные типы сетей:
  - Провода (wire)
  - Регистры (reg)
- Провода могут иметь ветвление
- Провода можно объединять в шины

Язык Verilog – сети

```
1  module counter (clock, reset, enable, counter_out);
2
3    input clock;
4    input reset;
5    input enable;
6
7    output [3:0] counter_out;
8
9    wire clock;
10   wire reset;
11   wire enable;
12
13   reg [3:0] counter_out;
14
```

Язык Verilog - порты

- Порты задают входы и выходы модулей
  - input, output, inout
- Связывают модули с сетями
- Все модули имеют порты (кроме модулей верхнего уровня иерархии)
- Явная (by name) и неявная (by order) «привязка» портов

Язык Verilog – объявление портов

```
1  module bit_adder(in_1, in_2, carry_in, out, carry_out);
2    ...
3  endmodule
4
5  module adder(a, b, c);
6    input [1:0] a, b;
7    output [2:0] c;
8
9    wire [1:0] a, b;
10   wire [2:0] c;
11   wire t_0, t_1;
12   ...
13   assign t_0 = 0;
14   bit_adder add_0 (
15     .in_1 (a[0]),
16     .in_2 (b[0]),
17     .carry_in (t_0),
18     .out (c[0]),
19     .carry_out (t_1)
20   );
21
22   bit_adder add_1 (a[1], b[1], t_1, c[1],c[2]);
23   ...
24 endmodule
```