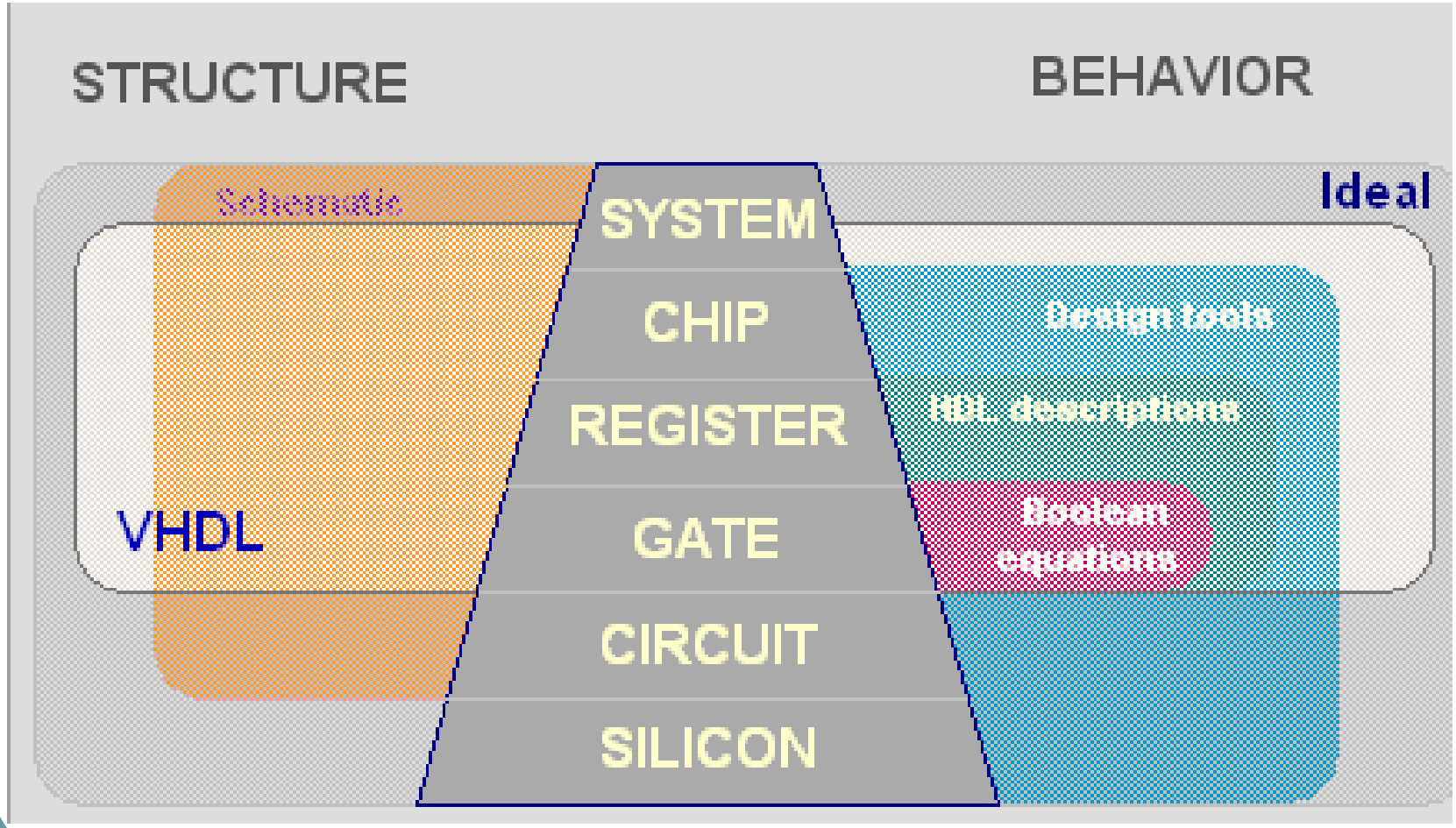
Языки HDL (Hardware Description Language)

- Проблемы традиционного дизайна:
  - несоответствие формы в которой ставится задача (обычно задается поведение) и схемы проекта (сеть взаимосвязанных компонентов)
  - «ручной» перевод описания проекта в набор логических выражений и схематику
  - работа со сложными (тысячи и более компонентов) проектами
- Наиболее используемые HDL-языки: VHDL, Verilog, Abel
- Very High Speed Integrated Circuit (VHSIC) Hardware Description Language (VHDL) предложен DoD USA в начале 80-х
  - Первые симуляторы появились в начале 90-х
  - ПО для ПК – в середине 90-х
  - Основное применение сейчас – PLD, FPGA

VHDL описания охватывают сразу несколько уровней проекта



Проектирование программируемой логики в OrCAD

Ввод проекта: схематики и VHDL-файлов

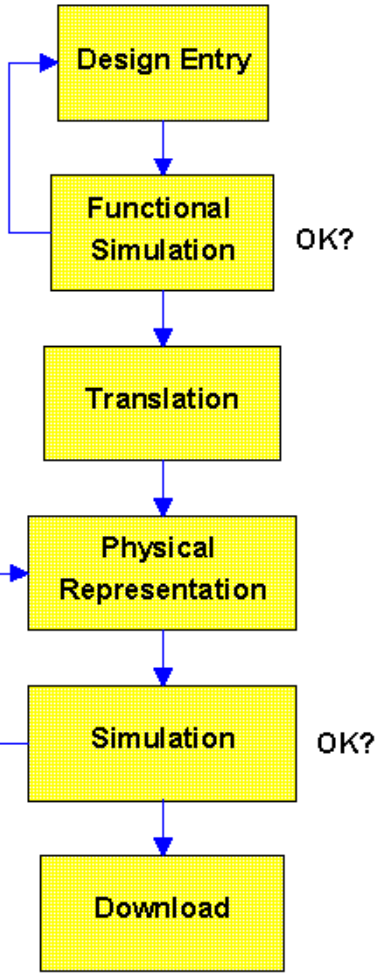
Поведение соответствует проектному

Перевод в конкретную технологию, оптимизация

Размещение с помощью программ Place&Route

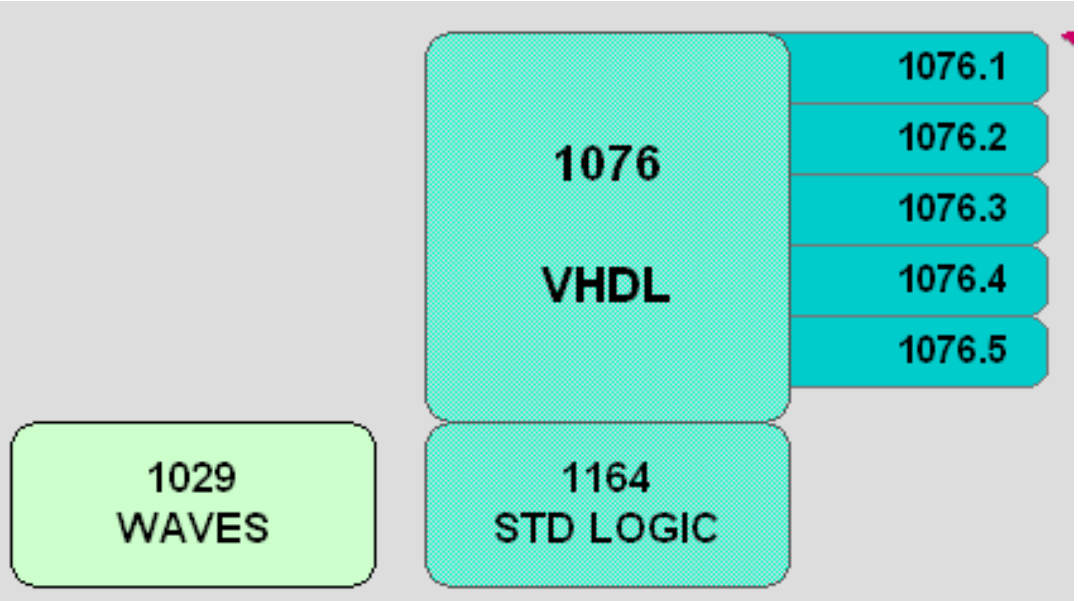
Проверка работоспособности «размещенного» проекта

«Прошивка» PLD



Стандартизация VHDL с 1987 г.

- ANSI-IEEE Std 1076-1987 (aka VHDL'87)
- ANSI-IEEE Std 1076-1993 (aka VHDL'93)
- ANSI-IEEE Std 1076.1-1999
- IEEE P1076.6/D1.12 Draft Std for VHDL Register Transfer Level Synthesis, 1998 – синтезируемое подмножество VHDL



Пакеты (packages)

- Предопределенные пакеты
  - STANDARD - стандартные определения и объявления языка VHDL (используется по-умолчанию)
  - TEXTIO – объявления основных операций над текстом
  - STD_LOGIC_1164 – часто используемые расширения языка VHDL при проектировании цифровых систем
- Существуют и библиотеки помимо IEEE, сторонних организаций, например, разработчиков инструментария для VHDL
- Для использования пакет объявляется перед entity
  - library IEEE;
  - use IEEE.Std_Logic_1164.all;

  или
  - library Std;
  - use Std.TextIO.all;

Типы данных VHDL, синтезируемые Express/Leonardo

Standard types	Class
natural, positive	Subtype of integer
string	Array of character
bit_vector	Array of bit

Std_logic_1164 types	Class
std_ulogic	Enumeration
std_logic	Subtype of std_ulogic
std_ulogic_vector	Array of std_ulogic
std_logic_vector	Array of std_logic

Numeric_std types	Class
signed	Unconstrained array of bit
unsigned	Unconstrained array of bit

Описание системы на VHDL

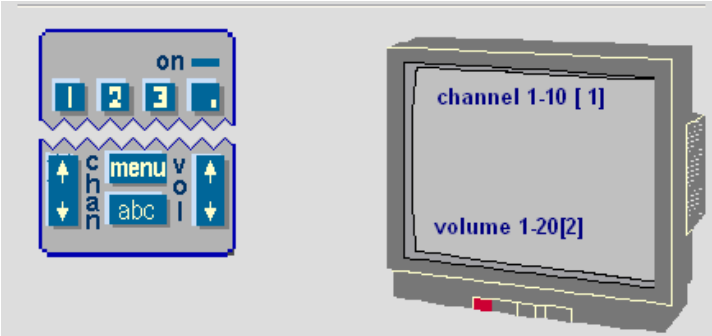
- Описание системы на VHDL реализуется в двух основных частях:
  - интерфейс – описание взаимодействия между системой и ее окружением
  - архитектура – описание поведения (функциональности) системы
- Кроме того, часто используются внешние модули, т.н. пакеты (packages)

Описание интерфейса


```
entity Eight_bit_register is

generic ( LENGTH = 8
          fmax   = 50 MHz

        );                                   parameters

port    ( D_IN   eight-bit input
          D_OUT  eight-bit output
          CLK    one-bit input
        );                                   connections

end entity Eight_bit_register;
```

Функциональное описание системы

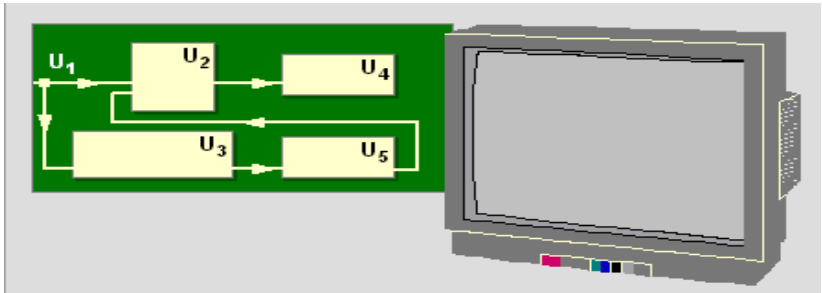
Поведенческое описание
(что делает?)




```
entity TVset is
   port  (
                VOLUME_UP : one-bit input
         );
end entity TVset;
```

```
architecture TV2000 of TVset is

...    if VOLUME_UP then increase the sound volume;

end architecture TV2000;
```

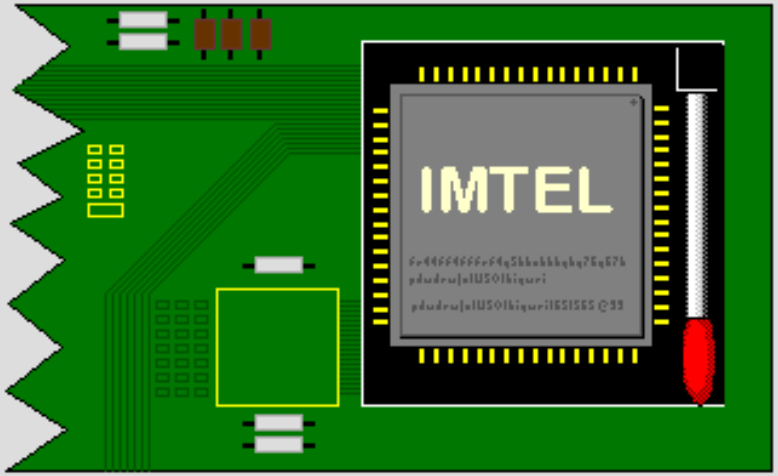
Структурное описание
(как реализовано?)




```
entity TVset is
   port  (

         );
end entity TVset;
```

```
architecture TV2000 of TVset is

...        U1, U5 -> U2 -> U4

end architecture TV2000;
```

Одному интерфейсу может соответствовать несколько архитектур



AMC

CEDAR

IMTEL

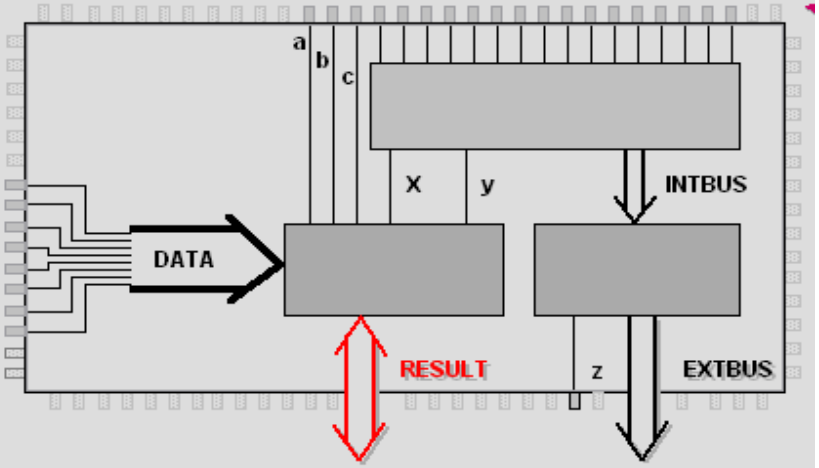
LYRIX

entity SIXTIUM **is**
…
end entity SIXTIUM;

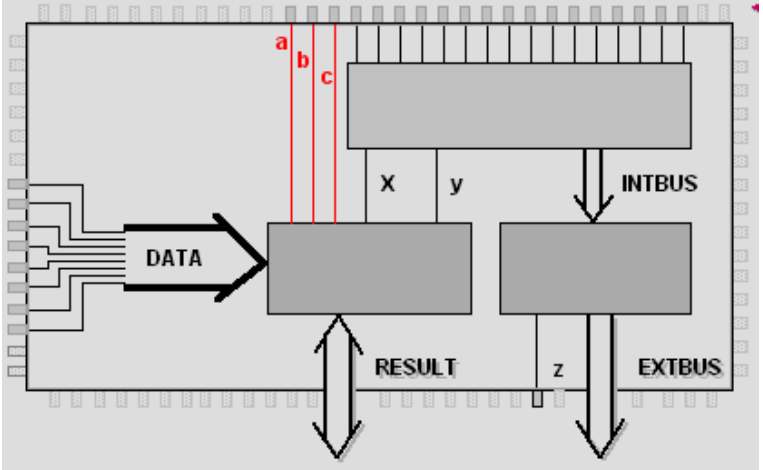
architecture IMTEL **of** SIXTIUM **is**
…
end IMTEL;

A new processor is needed. Which should I choose?

Сигналы в VHDL

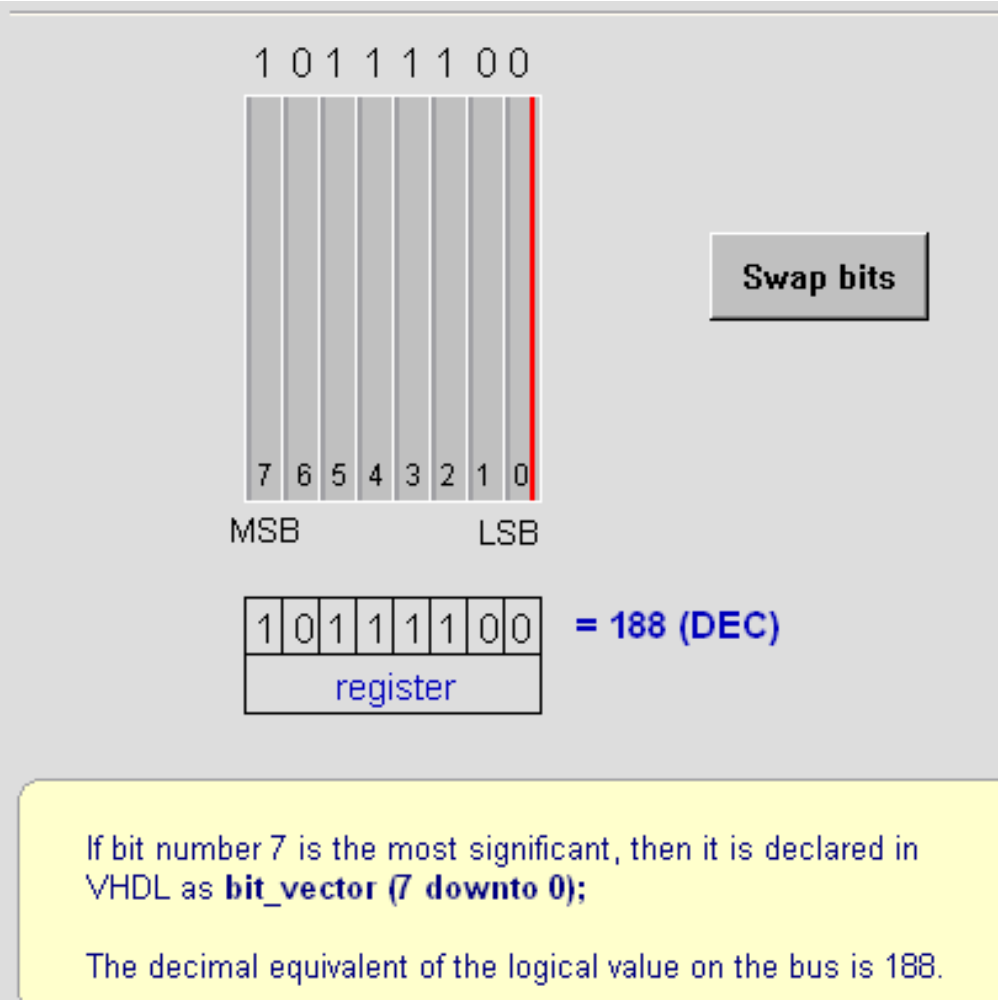


VHDL type of a signal: `bit_vector (bus)`



VHDL type of a signal: `bit (wire)`

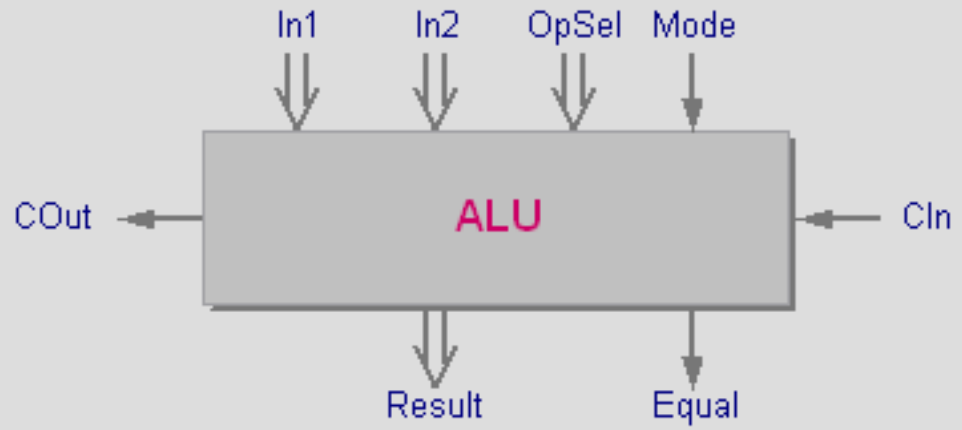
Объявления шин (0-LSB, 7-MSB)



Определение интерфейса. Заголовок-идентификатор Entity

```
entity ALU is
  port ( A,B  :  in bit_vector(7 downto 0); -- comment
         Mod  :  in bit_vector(3 downto 0);
         C    :  out bit_vector(7 downto 0));
end entity ALU;
```

Динамические каналы обмена между блоком и окружением - port




```
entity ALU is
   port(
      In1    :in bit_vector(3 downto 0);  -- first operand
      In2    :in bit_vector(3 downto 0);  -- second operand
      OpSel  :in bit_vector(3 downto 0);  -- operation select
      CIn    :in bit;                     -- carry in
      Mode   :in bit;                     -- mode arithm/logic
      Result :out bit_vector(3 downto 0); -- operation result
      COut   : out bit;                   -- carry out
      Equal  : out bit                    -- In1 = In2 when 1
        );
end entity ALU;
```

Статические каналы обмена между блоком и окружением - generic

description

Generic as size of an object

generic declaration

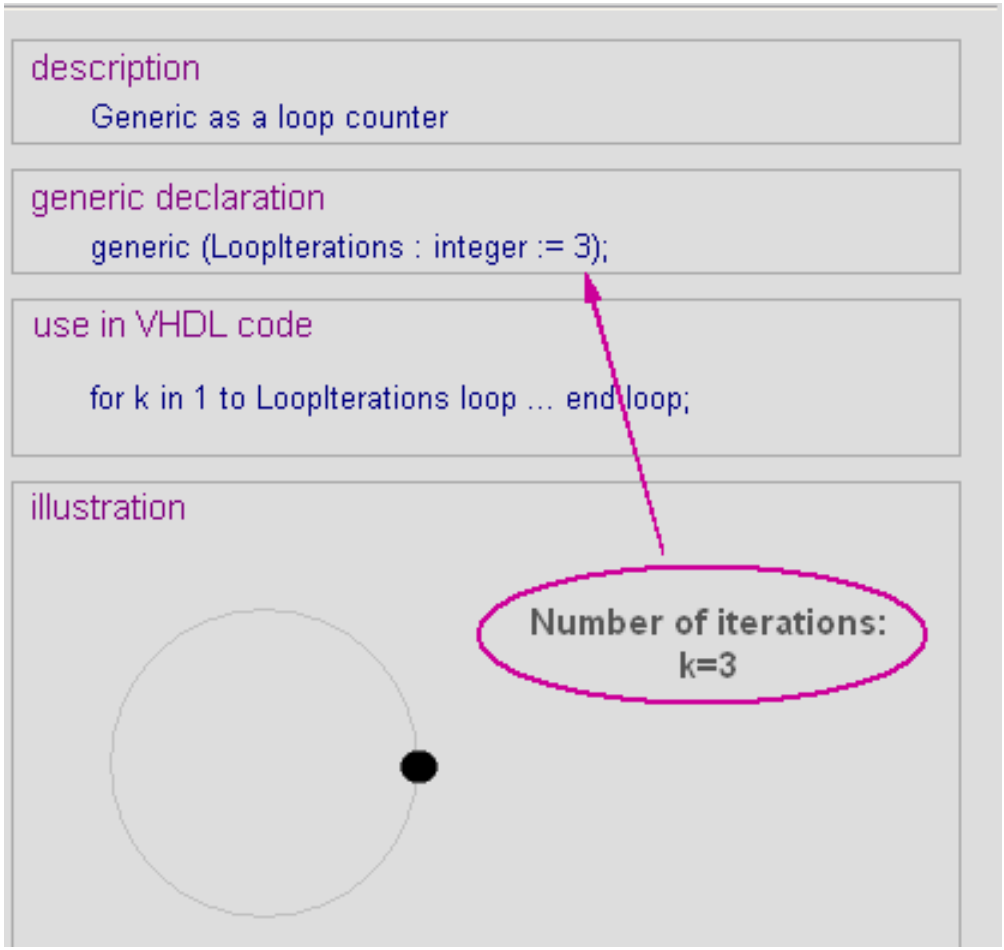
```
generic (BusWidth : integer := 8);
```

use in VHDL code

```
port( DataBus : inout bit_vector(BusWidth-1 downto 0));
```

illustration

Использование generic



Описание поведения системы

- Типы данных (строгая типизация, использование разных типов в одной операции - невозможно). Типы определяются в пакете, процессе, архитектурном теле.
  - Скалярные, диапазоны
    - целый: type byte_int is range 0 to 255, пакет STANDARD определяет bit, integer, positive …
    - вещественный: type voltage is range 0.0 to 10.0
    - физические, например, time
    - нумерованные (перечислимые): type logic_level is (low, high);
  - Составной
    - Массивы (type BIT_VECTOR is array (NATURAL range <>)of bit; затем, при применении: BIT_VECTOR (0 to 3))
    - Записи (type instr is record
      - Mnemonic: string;
      - Code: bit_vector (3 downto 0);
      - ExeCycles: Integer;
      - end record;)

Выражения и операторы

- операторы: логические, арифметические, отношений
- задание значений сигналов, примеры:
  - x<=y<=z;
  - a<=b or c;
  - k<='1'; (апострофы используются для типов bit и character)
  - m<="0101"; (кавычки используются для типов string и bit_vector)
  - n<=m&k; (слияние двух значений типа bit_vector)
- Задержки: инерционные и транспортные (при отсутствии задержки, изменение сигнала произойдет в очередной цикл симулятора (дельта-задержкой) после изменения причинных сигналов)
  - out1<=Input after 3 s;
  - out2<=transport Input after 3 s;

Элементы языка VHDL

- Лексические элементы:
  - идентификатор: Abc1
  - разделитель
  - ключевое (зарезервированное) слово: entity, if, then
  - литерал (десятичный, базовый, символьный, строковый, строка бит): 123, 1E2,123.4, ,1.2e-3; 2#1111_1100#, 10#252#; ’A’, ’a’; “abcdefg”; B“1010_1010_1010”
  - комментарий
- Лексические элементы разделяются:
  - разделителями: & ( ) * + ? - . / : ; < = > |
  - концами строк
  - знаками форматирования
  - составными разделителями: => ** := >=
- Пример: A <= B and C;

Подтипы

- Подтип – тип с ограничением:
- subtype имя_подтипа is базовый_тип [ограничение]

- Пример:

```
Type big_integer is range 0 to 2000;
Subtype small_integer is big_integer range 0 to 15
```

- Независимое определение типов big_ и small_ как диапазонов 0-2000, 0-15 привело бы к ошибке, например, при присвоении.

Объявления (декларации)

- Декларация константы:
  - `constant список_констант: тип [:=значение]`
- Декларация переменной:
  - `variable список_переменных: тип [:=значение]`
- Декларация сигнала:
  - `signal список_сигналов: тип [:=значение]`
- Декларация компонента:
  - `component имя компонента`
    - `generic (список параметров)`
    - `port (список портов)`
    - `end component`

Атрибуты объектов языка VHDL

- Предопределенные (predefined)
  - CLOCK’**event** – тип BOOLEAN со значением TRUE, когда сигнал CLOCK изменился
- Пользовательские (user-defined)

```
Signal vect : bit_vector (0 to 5);
Attribute two_length : integer;
Attribute two_length of vect :
   Signal is  (vect’length)*2;
…
C <= vect’two_length;
```

Имена

- Имена используются для декларированных объектов. Бывают:
  - Простые: CLOCK, COUNTER
  - Индексированные (массив): INSTR(13)
  - Интервал имен: INSTR (7 downto 0)
  - Выборка имен: DATE1.month

Поведенческое описание с использованием процессов

```
name: process (sensitivity list)
  declarations
begin
  sequential statements
end process name;


MUX2TO1: process (A,B,SEL)
  constant High : Bit := '1';
begin
  Y <= A;
  if (SEL='1') then Y <= B;
  end if;
end process MUX2TO1;
```

Приостановка и возобновление процессов

- wait for type_expression
- wait until condition
- wait on sensitivity_list
- complex wait

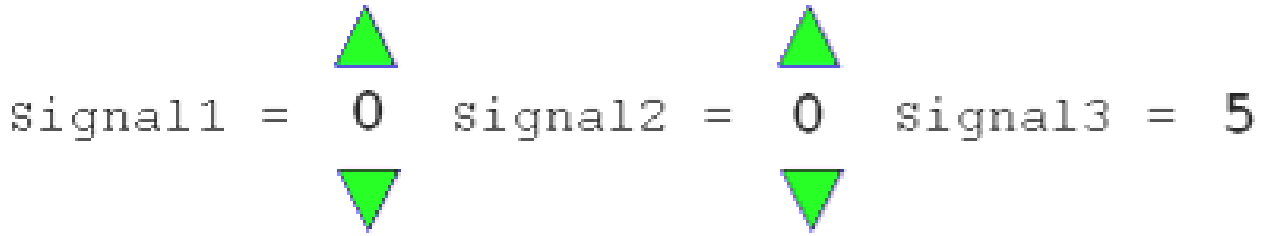
“сложный (complex)” вариант оператора ожидания (wait) содержит сочетание двух или трех различных его форм:

```
wait on Data until Clk ='1';
wait until Clk='1' for 10 ns;
```

Sensitivity list процесса

- Правила использования сигналов в процессах
  - сигналы объявляются вне процесса
  - присвоение значений сигналу происходит в состоянии приостановки от т.н. драйвера (источника) сигнала
  - только последнее присвоение внутри процесса действует

Обновление значений сигналов, входящих в sensitivity list процесса



Signal changed Value

```
process (Signal1,Signal2,Signal3)
begin
   SomeStatement1;
   Signal3 <= Signal1 + 5;
   SomeStatement3;
   SomeStatement4;
   SomeStatement5;
end process;
```

Присвоение значений сигналам в процессах

```
process(C,D)
begin
   A <= 2;
   B <= A + C;
   A <= D + 1;
   E <= A * 2;
end process;
```

```
A = 3        A <= 3
B = 2        B <= 2
C = 1
D = 2
E = 2        E <= 2
```

1. Исходные значения всех сигналов 1
2. Значение сигнала D изменилось с 1 на 2
3. После выполнения операций A=3, B=2, E=2
4. Процесс приостановился и новые значения присвоились сигналам A, B, E

Использование переменных в процессах

```
process(C,D)
variable Av, Bv, Ev : integer := 0;
begin
   Av := 2;
   Bv := Av + C;
   Av := D + 1;
   Ev := Av * 2;
   A <= Av;
   B <= Bv;
   E <= Ev;
end process;
```

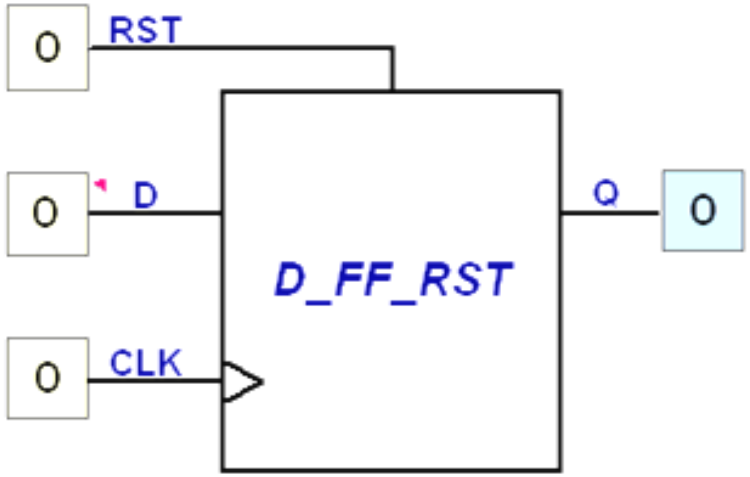
1. Исходные значения всех сигналов 1
2. Значение сигнала D изменилось с 1 на 2
3. Переменные Av=Bv=Ev=0
4. После выполнения операций A=3, B=3, E=6
5. Процесс приостановился и новые значения присвоились сигналам A, B, E

```
A = 3      A <= 3          Av = 3
B = 3      B <= 3          Bv = 3
C = 1
D = 2
E = 6      E <= 6          Ev = 6
```

Сигналы и переменные в процессах

	сигналы	переменные
объявление	вне процесса	внутри процесса
изменение значения	во время приостановки процесса	немедленно
задержки	допустимы инерционные и транспортные задержки	невозможны

Условные конструкции в процессах

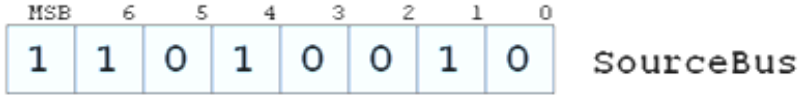



```
D_FF_RST : process (D, CLK, RST)
  begin
    if RST = 1
      then Q <= '0';
        elsif rising_edge(CLK)
          then
            Q <= D;
    end if;
  end process D_FF_RST;
```

Циклы в процессах

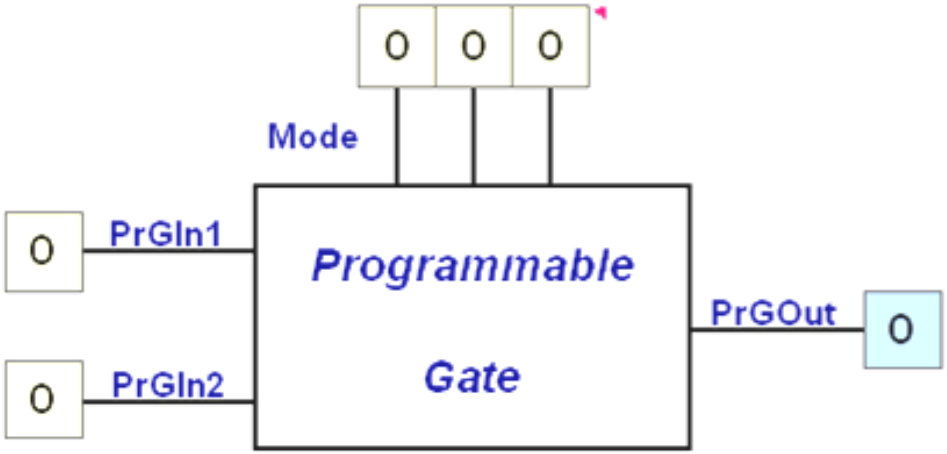
```
process
  variable Count : integer := 0;
  begin
    wait until Clk='1';
    while Level = '1' loop
      Count := Count + 1;
      wait until Clk = '0';
    end loop;
  end process;
```


```
constant MSB : natural := 7;
subtype BusRange is natural range MSB downto 0;
signal SourceBus : bit_vector(BusRange);

ReverseBus: process (SourceBus)
  variable ResultBus : bit_vector(BusRange);
  begin
    for Cntr in BusRange loop
      ResultBus(MSB - Cntr) := SourceBus(Cntr);
    end loop;
  end process ReverseBus;
```



The *Cntr* loop counter is not declared anywhere but this is legal in VHDL because its range is specified with a subtype defined earlier. It could also be explicitly written as "*for Cntr in MSB downto 0*" loop.

Выбор (case) в процессах




```
ProgrGate: process (Mode, PrGIn1, PrGIn2)
 begin
   case Mode is
     when "000"  => PrGOut <= PrGIn1 and PrGIn2;
     when "001"  => PrGOut <= PrGIn1 or PrGIn2;
     when "010"  => PrGOut <= PrGIn1 nand PrGIn2;
     when "011"  => PrGOut <= PrGIn1 nor PrGIn2;
     when "100"  => PrGOut <= not PrGIn1;
     when "101"  => PrGOut <= not PrGIn2;
     when others => PrGOut <= '0';
   end case;
end process ProgrGate;
```

Тесты (TestBench)

```
…
process begin
   wait for 100 ns;
   if (unsigned(A)=15) then
      B <= std_logic_vector(unsigned(B) + 1);
   end if;
   A <= std_logic_vector(unsigned(A) + 1);
end process;
…
```

Программируемые логические ИС

- Программируемые логические матрицы (ПЛМ) – программируемые матрицы “И” и “ИЛИ”. FPLA (Field Programmable Logic Array) и FPLS (Field Programmable Logic Sequensers)
- Программируемая матричная логика (PAL — *Programmable Array Logic*) имеет программируемую матрицу ”И” и фиксированную матрицу “ИЛИ”
- программируемая макрологика содержат единственную программируемую матрицу “И-НЕ” или “ИЛИ-НЕ”

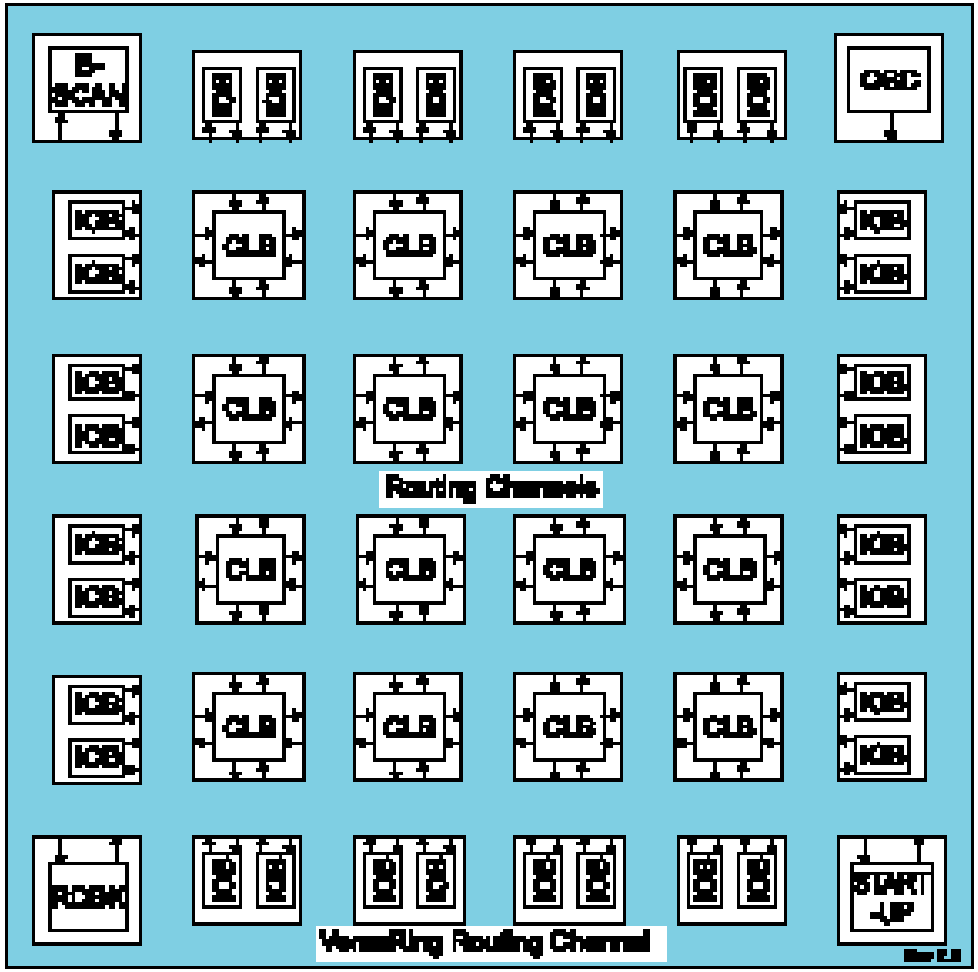
Развитие ПЛИС

- Программируемые коммутируемые матричные блоки (ПКМБ) — это ПЛИС, содержащие несколько матричных логических блоков (МЛБ), объединённых коммутационной матрицей.
- Каждый МЛБ представляет собой структуру типа ПМЛ, то есть программируемую матрицу “И”, фиксированную матрицу “ИЛИ” и макроячейки.
- ПЛИС типа ПКМБ, как правило, имеют высокую степень интеграции (до 10000 эквивалентных вентилей, до 256 макроячеек).
- В зарубежной литературе они получили название *Complex Programmable Logic Devices* (CPLD).

Современные ПЛИС

- Современные ПЛИС – сеть из десятков тысяч реконфигурируемых логических блоков (configurable logic blocks - CLB), функциональность и межсоединения которых определяются содержимым статического ОЗУ или электрически-перепрограммируемым ПЗУ.
- Программируемые вентильные матрицы (ПВМ), состоящие из логических блоков (ЛБ) и коммутирующих путей — программируемых матриц соединений. Логические блоки таких ПЛИС состоят из одного или нескольких относительно простых логических элементов, в основе которых лежит таблица перекодировки (ТП, Look-up table — LUT), программируемый мультиплексор, D-триггер, а также цепи управления.

Field Programmable Gate Array (FPGA)



Характеристики ПЛИС семейства APEX20K фирмы ALTERA

	EP20K100	EP20K160	EP20K200	EP20K300	EP20K400	EP20K600	EP20K1000
Максимальное число эквивалентных вентилей	263 000	404 000	526 000	728 000	1 052 000	1 537 000	2 670 000
Число логических элементов	4 160	6 400	8 320	11 520	16 640	24 320	42 240
Встроенные блоки памяти	26	40	52	72	104	152	264
Максимальный объем памяти, бит	53 248	81 920	106 496	147 456	212 992	311 296	540 672
Число макроячеек	416	640	832	1 152	1 664	2 432	4 224
Число выводов пользователя	252	320	382	420	502	620	780

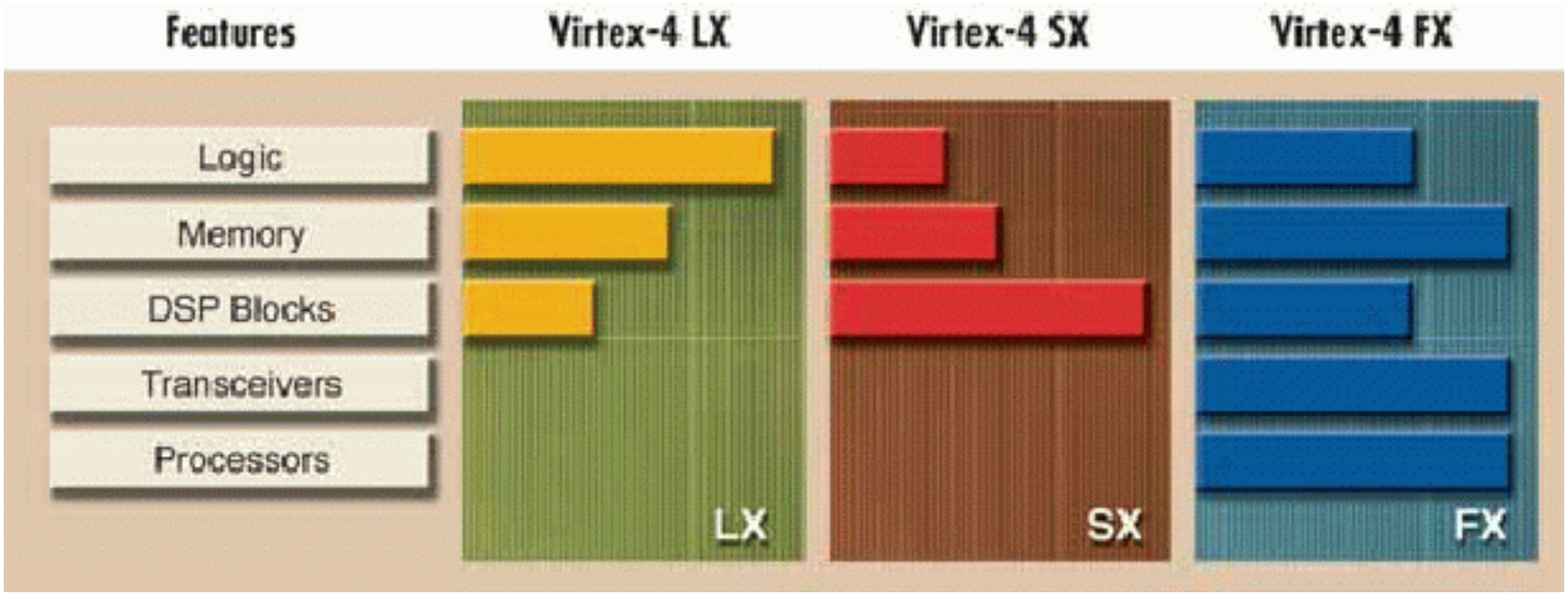
Фирма Altera Corporation, (101 Innovation Drive, San Jose, CA 95134, USA, www.altera.com) была основана в июне 1983 года.

Характеристики ПЛИС семейства Virtex фирмы XILINX

	XCV50	XCV100	XCV150	XCV200	XCV300	XCV400	XCV600	XCV800	XCV1000
Максимальное число эквивалентных вентилей	57 906	108 904	164 674	236 666	322 970	468 252	661 111	888 439	1 124 022
Число логических элементов	1 728	2 700	3 888	5 292	6 912	10 800	15 552	21 168	27 648
Максимальный объем памяти, бит	24 576	38 400	55 296	75 264	98 304	153 600	221 184	301 056	393 216
Число выводов пользователя	180	180	260	284	316	404	512	512	512

Компания Xilinx Inc. (2100 Logic Drive, San Jose, CA 95124-3400, USA, www.xilinx.com) была основана в феврале 1984

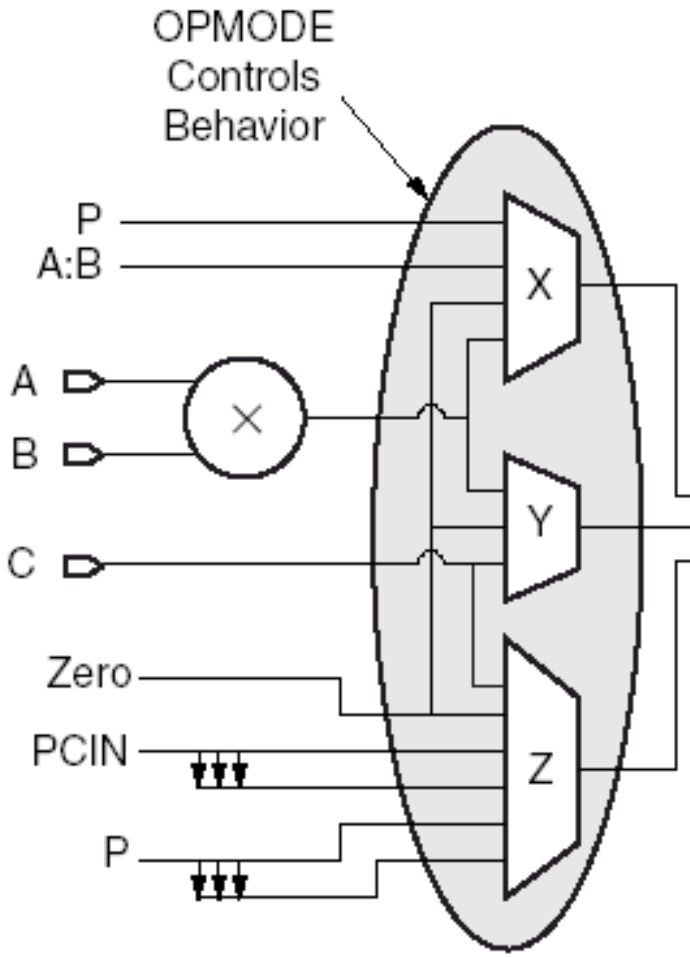
Три платформы ПЛИС Virtex-4



Производительность Virtex-4

	Virtex-4 -10	FPGA-90 nm Slow	Virtex-4 -11	FPGA-90 nm Middle	Virtex-4 -12	FPGA-90 nm Fast
Multiplier (9 x 9 or 18 x 18)	400 MHz	268 MHz	450 MHz	322 MHz	500 MHz	370 MHz
16-tap FIR filter with 8-bit data	400 MHz	165 MHz	450 MHz	203 MHz	500 MHz	237 MHz
16K x 32-bit RAM	527 MHz	306 MHz	610 MHz	326 MHz	643 MHz	370 MHz
16K x 64-bit RAM	311 MHz	195 MHz	311 MHz	226 MHz	335 MHz	272 MHz
4K x 144-bit RAM	400 MHz	289 MHz	450 MHz	309 MHz	500 MHz	350 MHz
64-bit Adder	205 MHz	173 MHz	221 MHz	208 MHz	245 MHz	239 MHz
48-bit Magnitude Compare	335 MHz	170 MHz	364 MHz	205 MHz	401 MHz	238 MHz
High-Speed Serial I/O	6.25 Gb/s	Not Applicable	10.3 Gb/s	Not Applicable	11 Gb/s	Not Applicable

Блок ЦОС DSP48-slice (упрощенная модель), XtremeDSP



Состав:

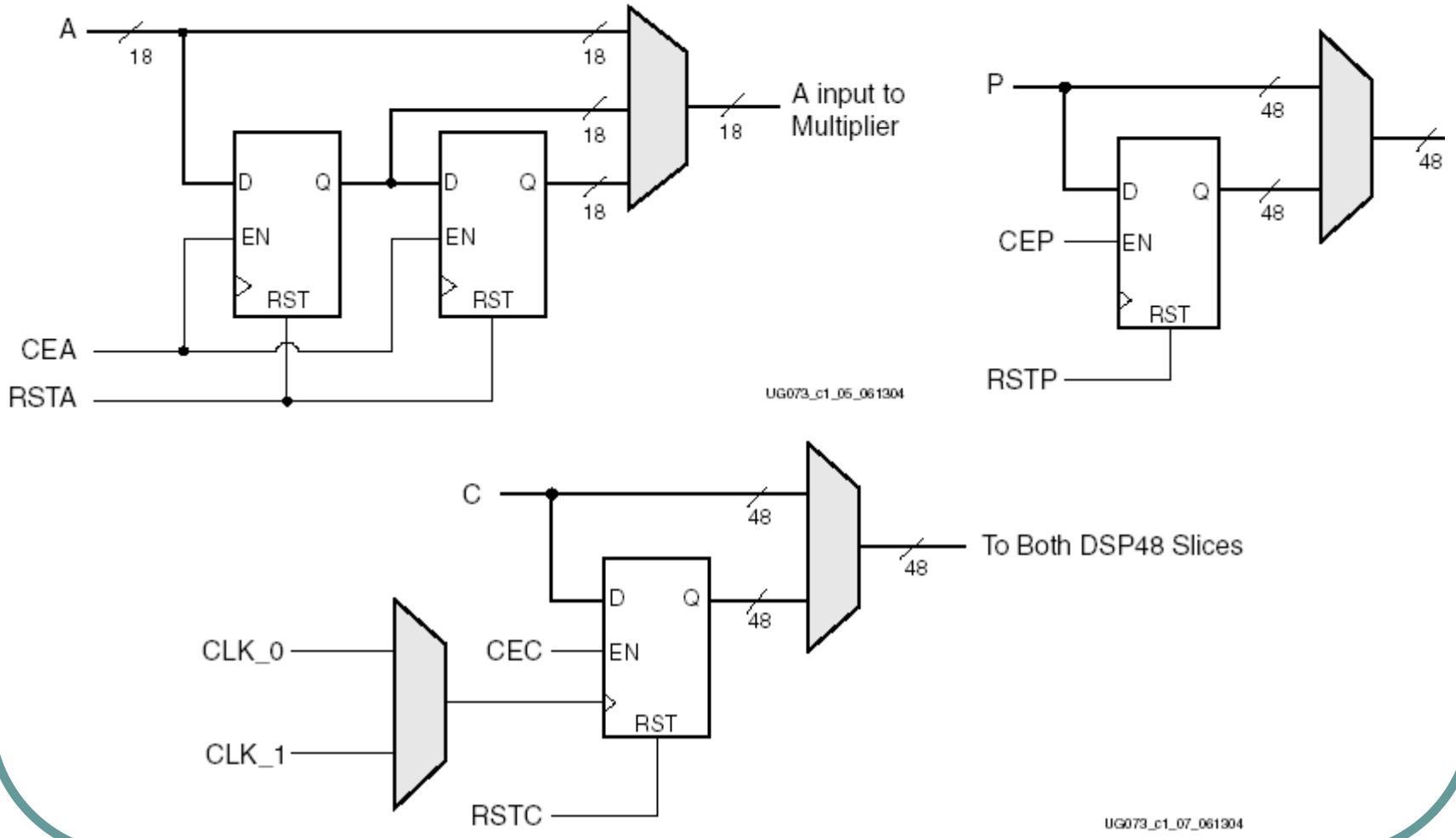
18-битный умножитель,

три 48-битных мультиплексора,

трех-операндный 48-битный сумматор

Поступление операндов – 500 млн/сек

Управление входными операндами и выходом DSP-slice



Последовательная реализация КИХ фильтров на DSP-slice

- Последовательная реализация
- Fs=CLK/N
- Для 96-звенного фильтра – 4.5 MSPS

$$y_n = \sum_{i=0}^{N-1} x_{n-i}\, h_i$$

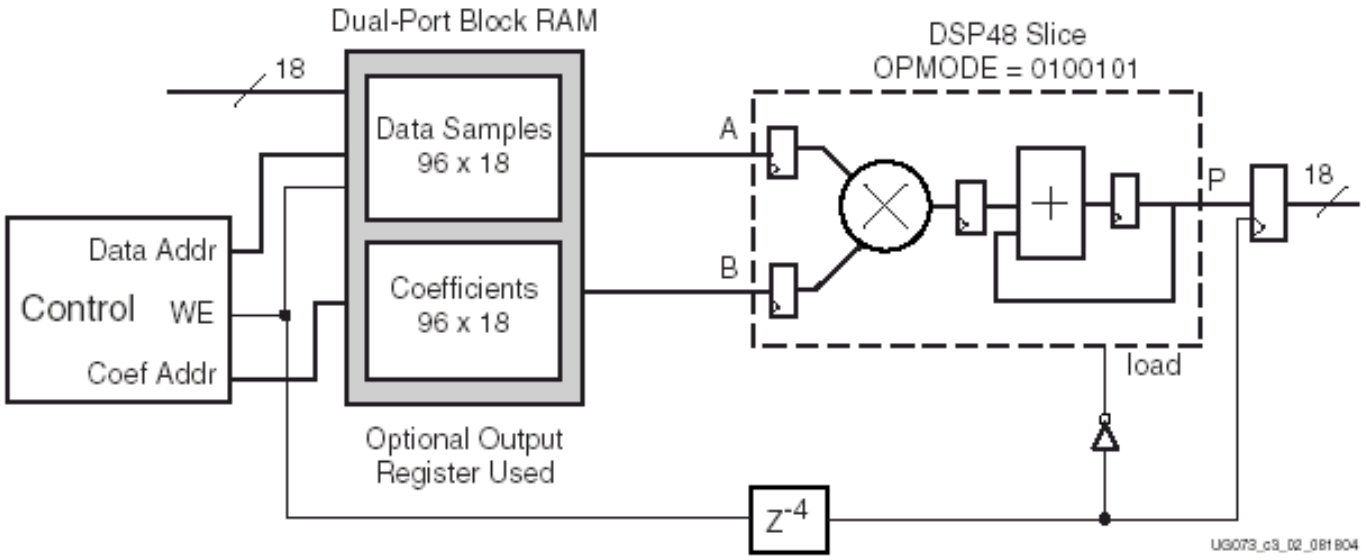
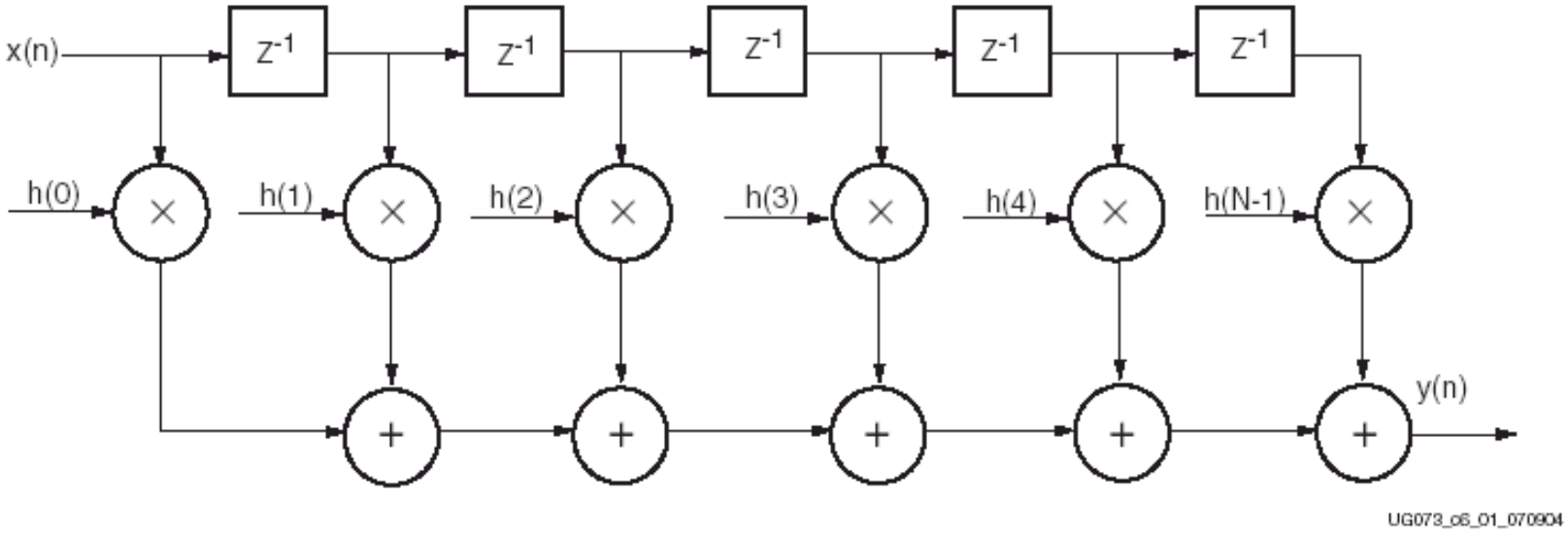
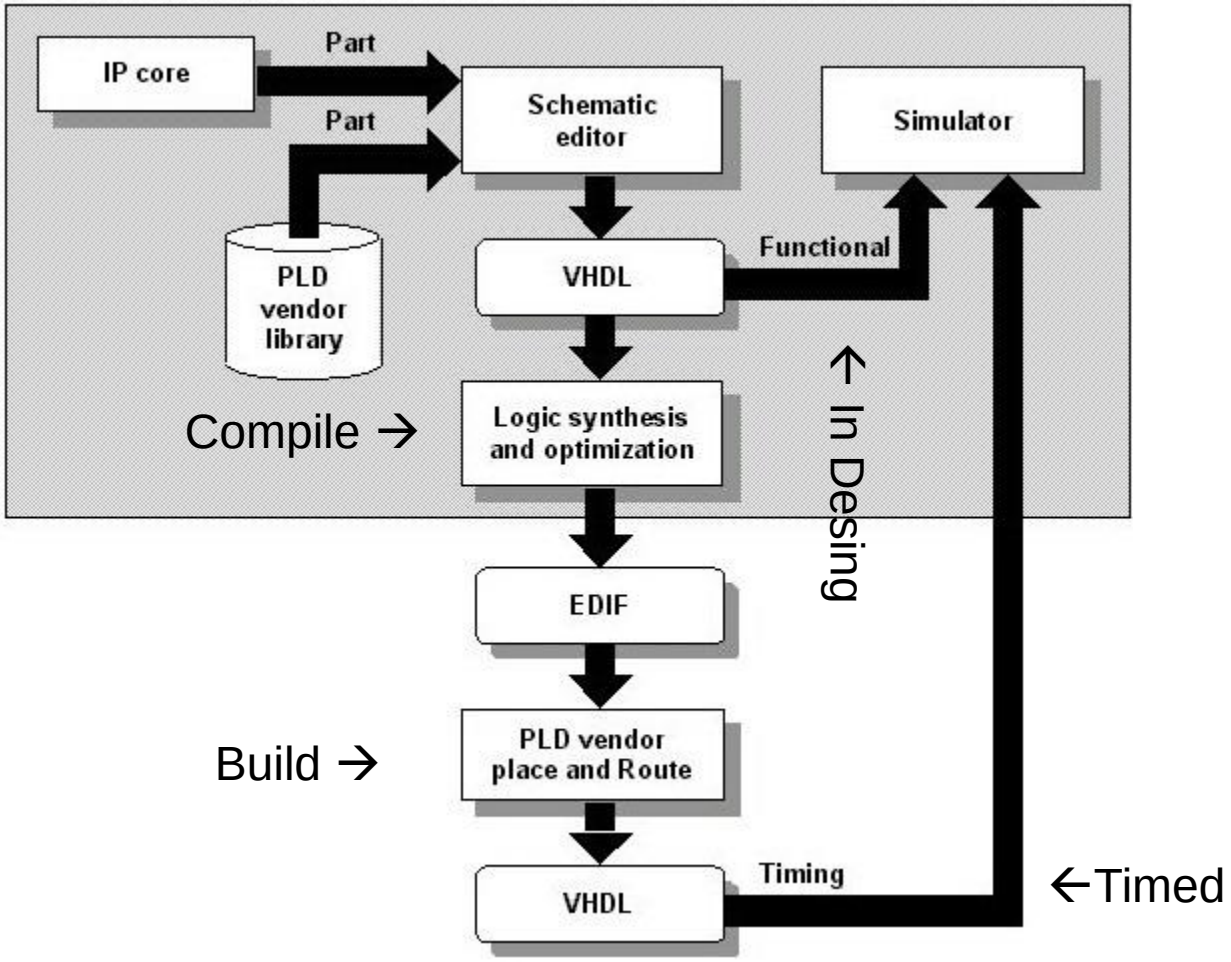


Figure 3-1: **Single-Multiplier MACC FIR Filter**


x(n)
Z-1
Z-1
Z-1
Z-1
Z-1
h(0)
h(1)
h(2)
h(3)
h(4)
h(N-1)
y(n)
UG073_c6_01_070904

Автоматический синтез логической схемы, основные этапы

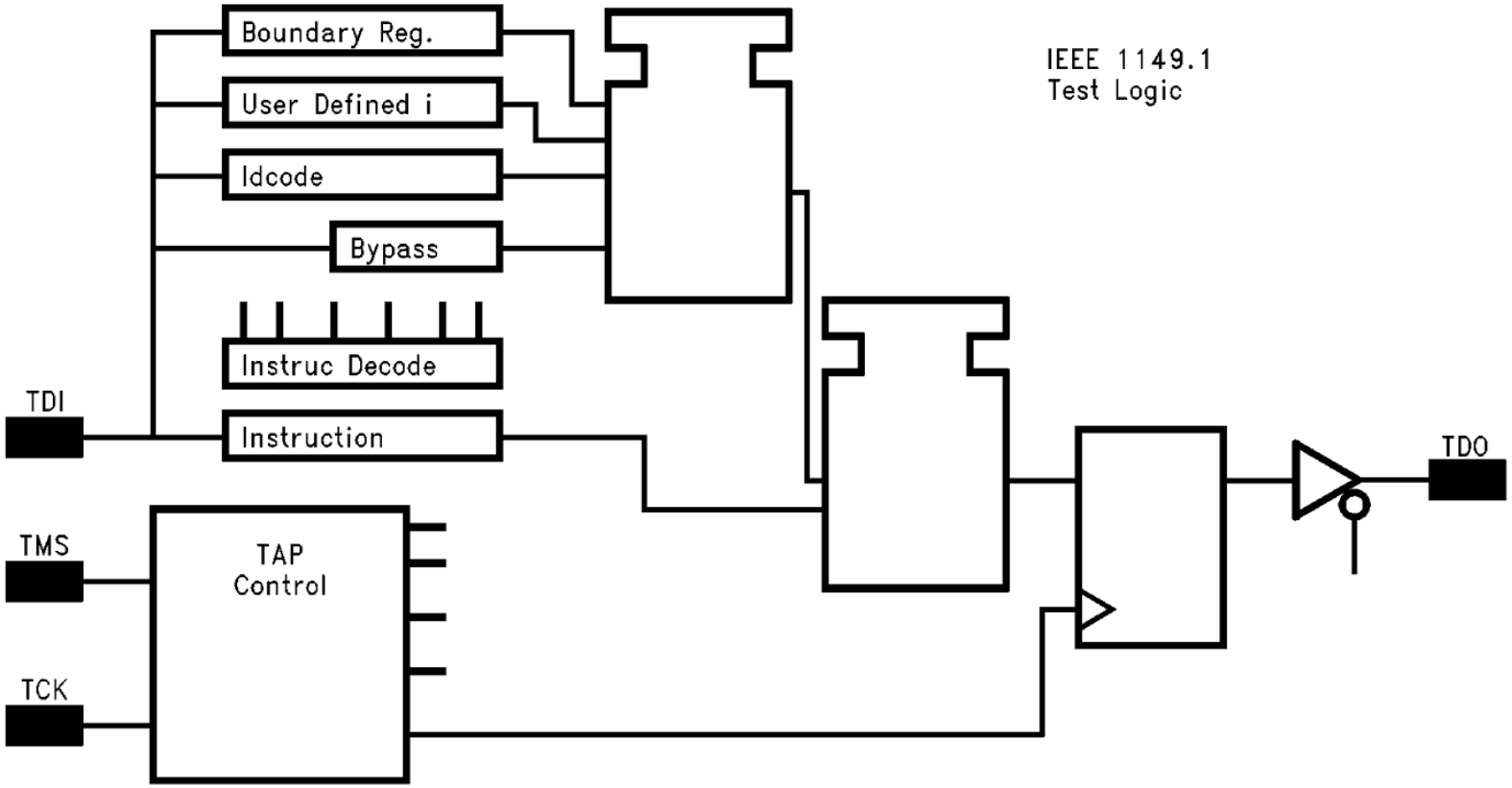
ПЛИС-проект в OrCAD, последовательность



IEEE-STD-IEEE P1149.1 (JTAG)

- IEEE-STD-1149.1- стандарт группы производителей электроники Joint Test Action Group (JTAG)
- IEEE-1149.1 - это стандарт «Standard Test Access Port and Boundary Scan Architecture»
- Часто под JTAG подразумевают 4-х проводной интерфейс этого стандарта
- 1149.1-2001 - IEEE
- Группа JTAG сформирована в 1985 для разработки методов тестирования печатных плат
- http://www.jtag.com

Базовая архитектура 1149.1



- TCK - синхросигнал системы тестирования TDI - входные последовательные данные
- TMS – выбор режима тестирования TDO - выходные последовательные данные
- TRST – сброс TAP-контроллера в начальное состояние (test-logic-reset state)

Литература

- Бибило П. Н. Синтез логических схем с использованием языка VHDL – М.: Солон-Р, 2002.-384 с.
- http://www.elsevierdirect.com/companions/9780123705976/errata/15~Chapter%2010%20BSCAN%201500.pdf