МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Юго-Западный государственный университет»
(ЮЗГУ)

Кафедра космического приборостроения и систем связи

УТВЕРЖДАЮ
Проректор по учебной работе
_____ О.Г.Локтионова
«15» 12 2017 г.

Использование языка VHDL для проектирования элементов программируемой логики

Методические указания по выполнению лабораторной работы
для студентов, обучающихся по направлению подготовки 11.03.03

Курск 2017

УДК 681.5

Составитель А.А. Гримов

Рецензент

Доктор технических наук, профессор А.Ф. Рыбочкин

Использование языка VHDL для проектирования элементов программируемой логики: методические указания по выполнению лабораторной работы по дисциплине «Информационные технологии конструирования электронных средств» / Юго-Зап. гос. ун-т.; сост.: А. А. Гримов. Курск, 2017. 26 с.: ил. 12, табл. 1.

Содержатся теоретические сведения о использовании языка описания аппаратуры VHDL для проектирования элементов программируемой логики, выполнение проектных операций и синтаксисе языка.

Указывается порядок выполнения лабораторной работы.

Текст печатается в авторской редакции

Подписано в печать . Формат 60×84 1/16.
Усл. печ. л. ___. Уч.-изд. л. ___. Тираж 100 экз. Заказ . Бесплатно.

Юго-Западный государственный университет.
305040, г. Курск, ул. 50 лет Октября, 94.

ОГЛАВЛЕНИЕ

1. ЦЕЛЬ РАБОТЫ

Целью работы является изучение основ языка описания аппаратуры VHDL и его применения при проектировании элементов программируемой логики.

2. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

В теоретической части рассматриваются этапы проектирования с использованием VHDL, приведены основные синтаксические конструкции языка и примеры их применения.

Что такое VHDL?

1. Промышленный стандарт IEEE для описания аппаратных средств.
2. Язык высокого уровня для моделирования и синтеза цифровых схем.

Терминология

HDL – язык описания аппаратных средств является языком, предназначенным для моделирования и синтеза схем

Behavior Modeling – моделирование поведения. Компонент описывается в качестве отклика его выходов на входные воздействия

Structural Modeling – моделирование структуры. Компонент описывается как соединения между более простыми компонентами или примитивами (компонентами или примитивами низкого уровня)

– Register Transfer Level (RTL) – тип моделирования поведения для целей синтеза, при котором осуществляется привязка модели к структуре регистров программируемого кристалла при синтезе схемы. При этом:
  - Учитывается предполагаемая элементная база или используются элементарные компоненты
  - Проект должен уложиться в рамки ограничений, накладываемых возможностями программируемого логического кристалла.

– **Synthesis** – синтез, трансляция HDL в схему и оптимизация полученной схемы

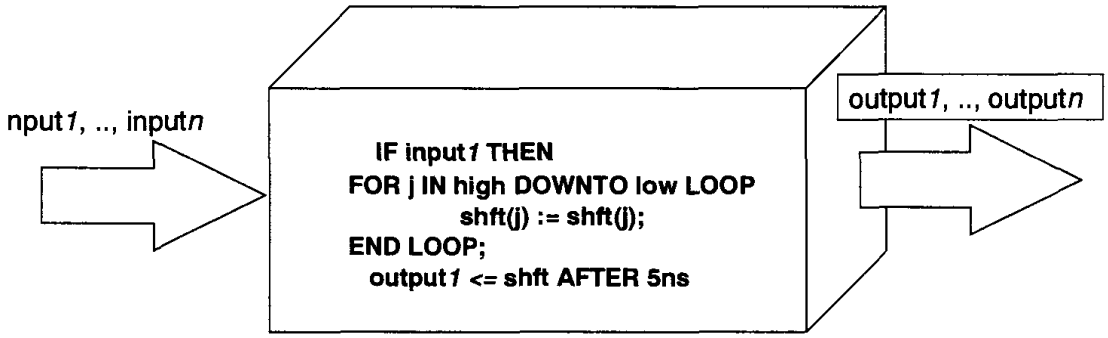
– **RTL Synthesis** – процесс трансляции RTL-модели схемы в структуру, оптимизированную для внедрения в выбранный тип программируемого кристалла

Behavior Modeling - моделирование поведения

Описывается только функционирование схемы, сведения о структуре не приводятся

Отсутствует привязка к специфическим аппаратным средствам

Используется как для моделирования, так и для синтеза схем

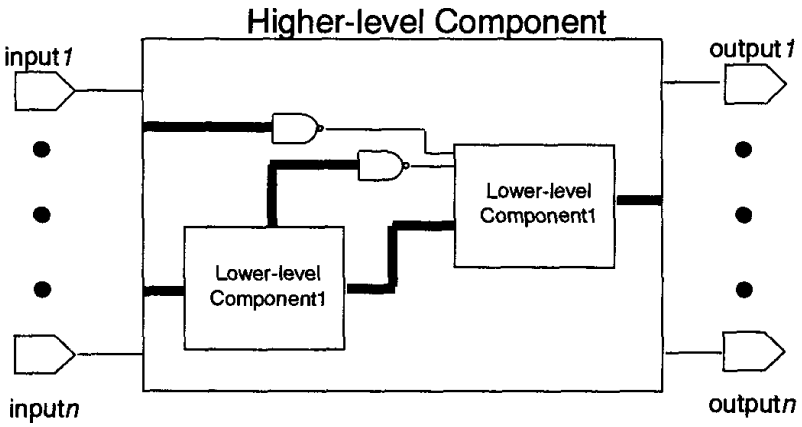


Structural Modeling - моделирование структуры

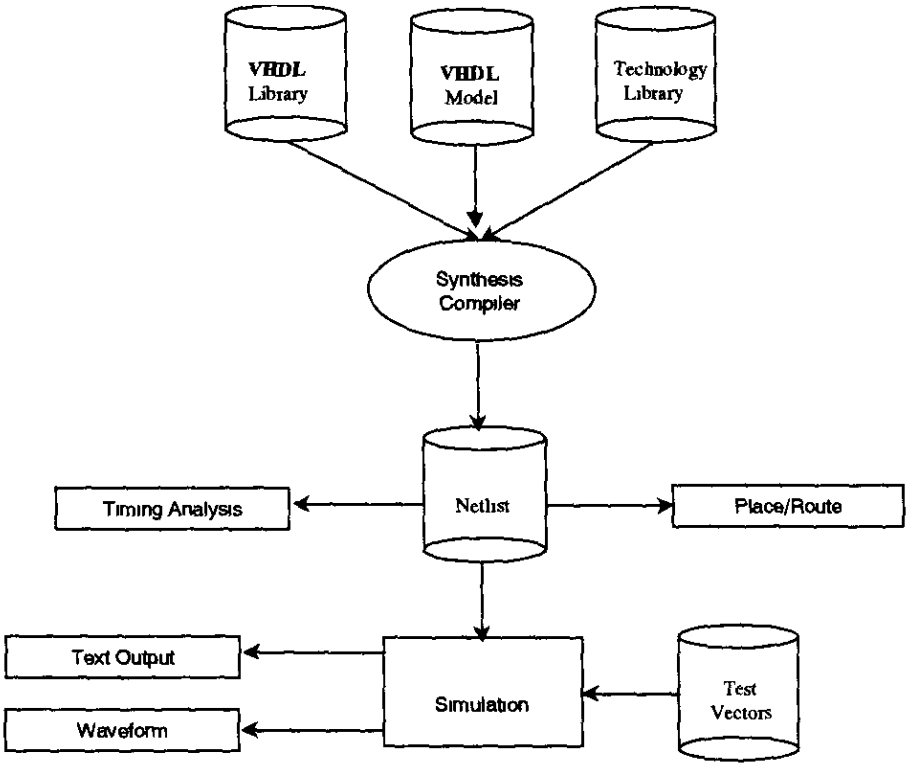
Описание функционирования и структуры схемы

Ориентация на специфические схемотехнические средства

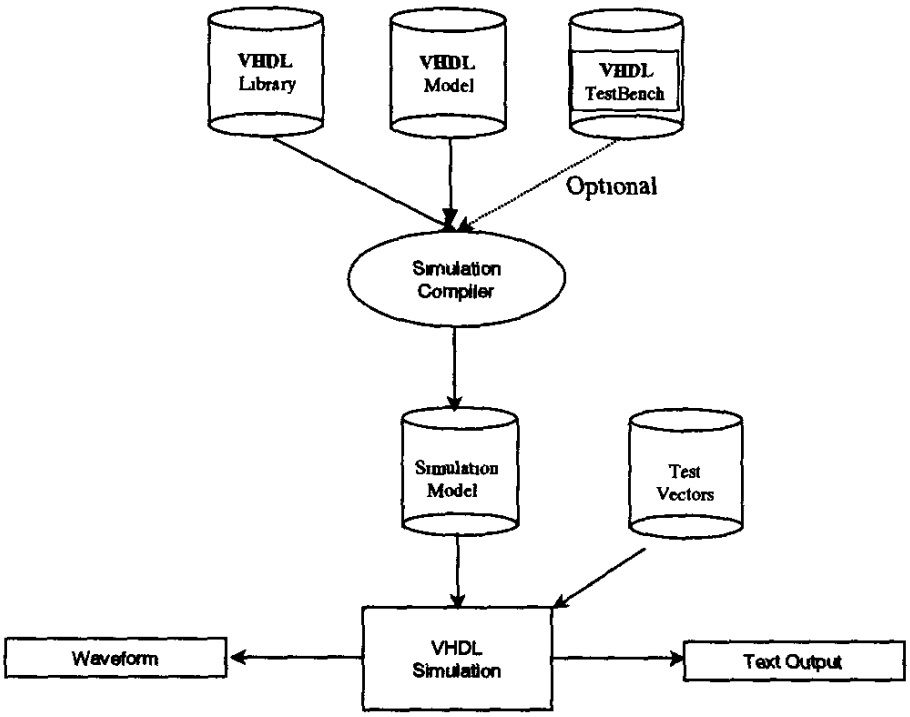
Используется как для моделирования, так и для синтеза схем



Typical Synthesis Design Flow – Порядок синтеза схемы



Typical Simulation Design Flow – Порядок моделирования схемы



Основы VHDL

– Имеется два набора языковых конструкций:
  - Синтез
  - Моделирование
– Язык VHDL построен на базе ключевых слов.
– В большинстве случаев язык НЕ РАЗЛИЧАЕТ прописные и строчные буквы.
– Выражения VHDL завершаются символом «точка с запятой» **;**
– VHDL не чувствителен к пробелам. Они используются для улучшения читаемости текста.
– Комментарии в VHDL начинаются с двух стоящих рядом дефисов и занимают остаток строки
– Модели VHDL могут быть:
  - Behavioral (поведение)
  - Structural (структура)
  - Mixed (смешанный тип)

Элементы проекта VHDL (Design Units)

Элементами проекта VHDL являются:

– Entity
  - Используется для определения интерфейса модели, т.е. модели с точки зрения ее окружения. Пример: условное обозначение мультиплексора.
– Architecture
  - Используется для определения функционирования модели. Пример: внутренняя схема мультиплексора.
– Configuration
  - Используется для указания связи между Architecture и Entity
– Package
  - Содержит набор сведений, к которым могут обращаться модели VHDL. Пример: Library (библиотеки)

- Состоит из двух частей: объявлений Package Declaration и тела пакета Package Body.

Entity

Entity Declaration – объявление Entity

```
ENTITY <entity_name> IS
        Generic Declarations
        Port Declarations
END <entity_name>; (1076-1987 version)
END ENTITY <entity_name> ; ( 1076-1993 version)
```

– Аналогия: символ компонента.

– <entity_name> может быть любым буквенно-цифровым именем

– Generic Declarations – общие объявления, используются для передачи информации в модель

– Port Declarations – объявления портов
  - Используются для описания входов и выходов, например, выводов компонентов.

Entity: Port Declarations

```
ENTITY <entity_name> IS
        Generic Declarations
        Port ( signal clk : in bit;
                --Note: signal is assumed and is not required
                        q : out bit
             );
END <entity_name>; (1076-1987 version)
END ENTITY <entity_name> ; ( 1076-1993 version)
```

Структура: <class> object_name : <mode> <type> ;

– <class> : что можно делать с объектом

– Object_name: identifier – идентификатор (символическое имя)

– <mode>: directional - направленность
  - **In** (вход) **out** (выход)
  - **Inout** (двунаправленный) **buffer** (выход с внутренней обратной связью)

– <type> : Что может содержаться в объекте

Architecture

Ключевые аспекты архитектуры:

– Аналогия: принципиальная схема

– Описывает функционирование и задержки времени в модели

– Должна быть привязана к **ENTITY**

– **ENTITY** может иметь несколько вариантов architecture

– Выражения в описании Architecture обрабатываются одновременно (concurrently) –параллельные процессы (Processes)

– Стили Architecture
  - Behavioral (поведенческий): Как функционирует проект
  - RTL: Проекты описываются в терминах регистров
  - Функциональный: без указания временных параметров
  - Structural (структурный): список связей (Netiist)
  - Уровень вентилей/компонентов (Gate/Component Level)
  - Hybrid (смешанный): комбинация указанных выше стилей

Объявление Architecture

```
ARCHITECTURE <identifier> OF <entity_identifier> IS
--architecture declaration section (list does not include all)
      signal temp : integer := 1;  -- Signal Declarations  =1 is default value optional
      constant load : boolean := true;  --Constant Declarations
      type states is ( S1, S2, S3, S4) ;   --Type Declarations
      --Component Declarations discussed later
      --Subtype Declarations
      --Attribute Declarations
      --Attribute Specifications
      --Subprogram Declarations
      --Subprogram body
BEGIN
    Process Statements
    Concurrent Procedural calls
    Concurrent Signal assignment
    Component instantiation statements
    Generate Statements
END <architecture identifier> ; (1076-1987 version)
END ARCHITECTURE; (1076-1993 version)
```

VHDL Basic Modeling Structure – базовая структура модели

ENTITY *entity_name* **IS**
generics
port declarations
END *entity_name;*

ARCHITECTURE *arch_name* **OF** *entity_name* **IS**
enumerated data types
internal signal declarations
component declarations
BEGIN
signal assignment statements
process statements
component instantiations
END *arch_name;*

Packages - пакеты

Packages - обычный способ хранения и использования информации в пределах всей модели.

Packages состоит из:

– Package Declaration – объявление пакета (Required)
  - Type declarations – объявления типов
  - Subprograms declarations – объявления подпрограмм
– Package Body –тело пакета (Optional)
  - Subprogram definitions – определения подпрограмм

VHDL имеет два встроенных набора Packages:
- Standard
- TEXTIO

Libraries - библиотеки

Содержит пакет или набор пакетов.

Resource Libraries – библиотеки ресурсов:
- Standard Package
- IEEE developed packages

Any library of design units that are referenced in a design

• Working Library – рабочая библиотека
  - Библиотека, в которую помещается результат компиляции модуля.

Подключение библиотеки

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY cmpl_sig IS
PORT ( a, b, sel : IN std_logic;
            x, y, z : OUT std_logic;
END cmpl_sig;
ARCHITECTURE logic OF cmpl_sig IS
BEGIN
              -- simple signal assignment
            x <= (a AND NOT sel) OR (b AND sel);
              -- conditional signal assignment
            y <= a WHEN sel='0' ELSE
                 b;
              -- selected signal assignment
            WITH sel SELECT
                        z <= a WHEN '0',
                             b WHEN '1',
                            '0' WHEN OTHERS;
END logic;
CONFIGURATION cmpl_sig_conf OF cmpl_sig IS
            FOR logic
            END FOR;
END cmpl_sig_conf;
```

Форматы объявлений:

LIBRARY <name>, <name> ;

Name – символическое имя, определяемое средствами инструментальногопрограммного пакета.

WORK, STD – встроенные библиотеки, они не требуют объявления

USE lib_name.pack_name.object;

ALL – ключевое слово, подключает все элементы библиотеки

Размещение выражений Library/Use в начале проекта дает возможность ссылки на ее компоненты в пределах всего проекта.

Библиотека LIBRARY STD

Содержит следующие пакеты:

• **standard** (Types: Bit, Boolean, Integer, Real, Time. Все операторные функции для поддержки типов данных)
• **textio** (File operations)

Является встроенной библиотекой
• Не требует объявления в проекте VHDL

Предопределенные в Standard Package типы данных

Type **BIT**
- 2-х значная логика ('0','1')

signal a_temp: bit;
- **BIT_VECTOR** array of bits

signal temp1: bit_vector(3 **downto** 0);

signal temp2: bit_vector(0 **to** 3);

Type **BOOLEAN**
- (false, true)

Type **Integer**

Положительные и отрицательные целые десятичные числа.

signal int_tmp : integer; - 32 bit number

signal int_tmp1 : integer **range** 0 to 255; -8 bit number

Примечание: Standard package имеет и другие типы

Библиотека LIBRARY IEEE;

Содержит следующие пакеты:
• **std_logic_1164** (std_logic types & related functions)
• **std_logic_arith** (арифметические функции)
• **std_logic_signed** (signed арифметические функции)
• **std_logic_unsigned** (unsigned арифметические функции)

Типы, определенные в std_logic_1164 Package

Type STD_LOGIC
- 9 – значная логика (**'U', 'X', '0', '1', 'Z', 'W', 'L', 'H','-'**)
  - **'W, 'L, 'H'** – «мягкие» величины (не поддерживаются при синтезе)
  - **'X' –** неопределенное значение

- **'Z'** - (not 'z') «третье» состояние
- **'-'** Don't Care

- Разрешение конфликтов на шине: разрешает конфликты при наличии нескольких задатчиков для одного сигнала

Type STD_ULOGIC

- 9 – значная логика, как в STD_LOGIC
- Нет механизма разрешения конфликтов на шине: не поддерживается возможность наличия нескольких задатчиков для одного сигнала;
- Возникновение ошибки в случае появления такой ситуации.

Основы описания архитектуры

Содержание раздела:

– Основы концепции и использование сигналов
  - Signal Assignments – назначение сигналов
  - Concurrent Signal Assignment statements – операторы параллельного присвоения значений сигналов
  - Signal Delays – задержки сигналов

– Processes - процессы
  - Implied -неявные
  - Explicit – явно определенные

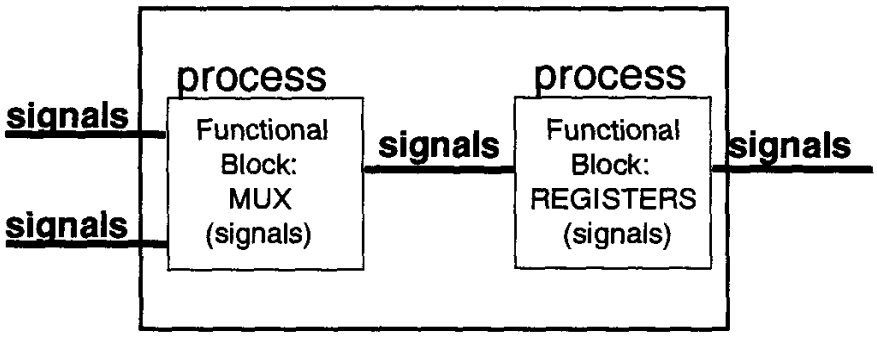
– Основы концепции и использование переменных

– Sequential Statement – последовательно выполняемые операторы
  - If-Then
  - Case
  - Loops

Сигналы

Сигналы имитируют физическую связь (провода), которые осуществляют контакт между процессами/функциями (модулями устройства).

- Сигналы могут быть объявлены в **Packages, Entity** и **Architecture**

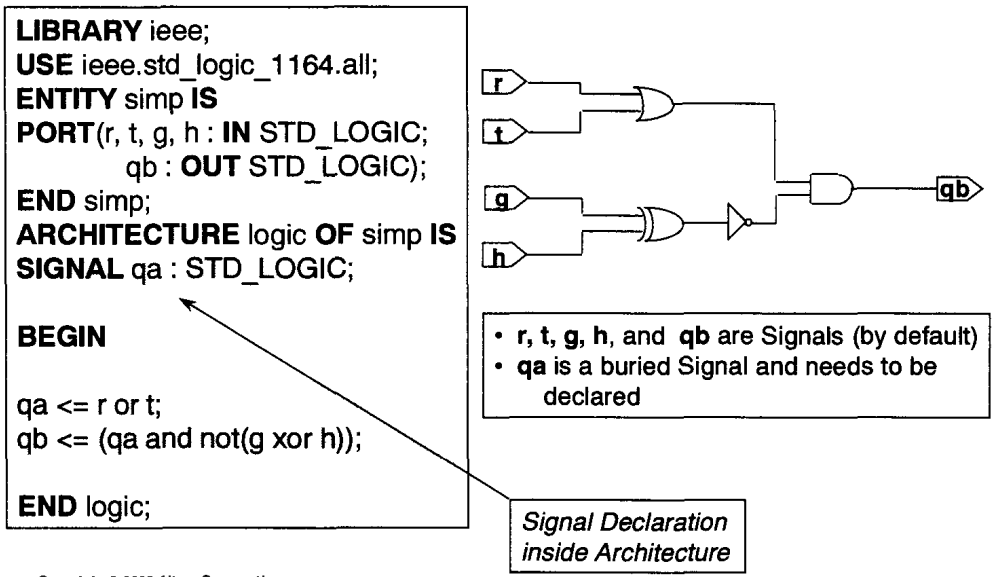


Присвоение сигналам значений

SIGNAL temp : STD_LOGIC_VECTOR (7 downto 0);

- All bits:
  temp <= "10101010";
  temp <= x"AA" ; (1076-1993)
- Single bit:
  temp(7) <= '1';
- Bit-slicing:
  temp (7 downto 4) <= "1010";
- Single-bit: single-quote (')
- Multi-bit: double-quote (")

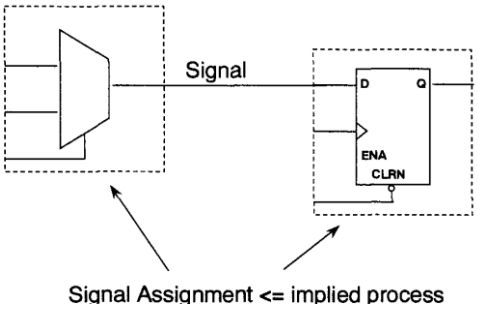
Использование сигналов для передачи информации



Определение значения сигнала

- Для определения значения сигнала используется запись: <=

Назначение сигнала подразумевает процесс (функцию), в которые этот сигнал будет имплементирован при синтезе схемы

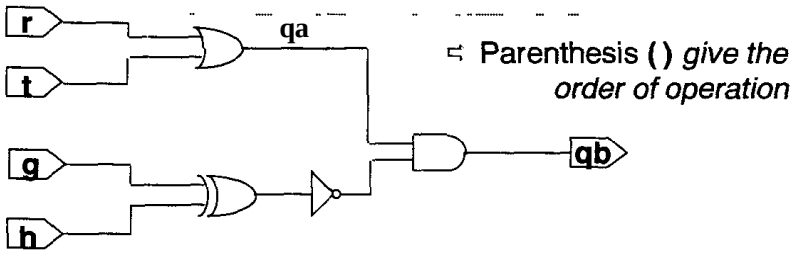


Варианты назначения сигналов

Три варианта назначения сигнала:

Простое назначение - Simple Signal Assignment
Условное назначение - Conditional Signal Assignment
Назначение по выбору - Selected Signal Assignment

Простое назначение сигнала



Формат:

<signal_name> <= <expression>;

Пример: введем в модель вспомогательный сигнал **qa**

qa<= r or t;
qb<= (qa and not (g xor h)); ⇒ implied process

Для описания процесса используются операторы VHDL

Операторы VHDL

Тип оператора	**Обозначение оператора**
Логические	**and or nand nor xor xnor**
Отношения	= /= < <= > >=
Сложение/вычитание	+ - **&**
Присвоение знака	+ -

Умножение/деление	*** / mod rem**
Другие	**** abs not**

VHDL определяет функции **Arithmetic & Boolean functions** только для встроенных типов данных, определенных в ***Standard*** **package**

Арифметические операторы такие как +, -, <, >, <=, >= определены только для типа INTEGER

Логические операторы такие как AND, OR, NOT определены только для типа BIT

Библиотекой VHDL по умолчанию (implicit library, built-in) является **Library STD.**

Типы, определенные в **Standard** package:

BIT, BOOLEAN, INTEGER

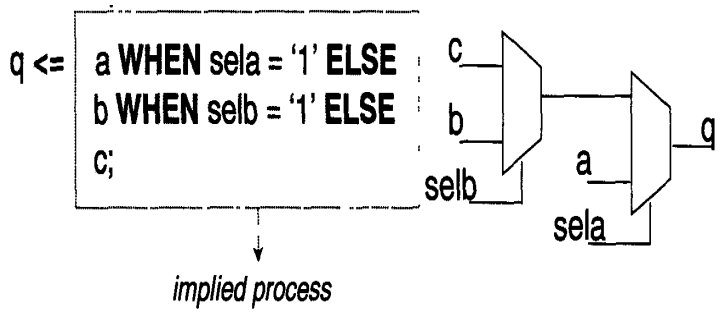
Примечание: элементы в этой библиотеки можно не объявлять, они являются встроенными.

Определение сигнала по условию - Conditional Signal Assignment

```
<signal_name> <= <signal/value> when <condition1> else
                 <signal/value> when <condition2> else
                                  .
                                  .
                 <signal/value> when <condition3> else
                 <signal/value>;
```

• Формат

```
q <= a WHEN sela = '1' ELSE
     b WHEN selb = '1' ELSE
     c;
```



• Пример:

Определение сигнала с помощью выбора варианта - Selected Signal Assignments

- Должны быть рассмотрены **BCE** возможные варианты
- Выражение **WHEN OTHERS** охватывает все другие варианты помимо явно перечисленных

Пример:

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY cmpl_sig IS
PORT ( a, b, sel : IN STD_LOGIC;
       z : OUT STD_LOGIC;
END cmpl_sig;

ARCHITECTURE logic OF cmpl_sig IS
BEGIN
        -- selected signal assignment
      WITH sel SELECT
            z <= a WHEN '0',
                 b WHEN '1',
                 '0' WHEN OTHERS;
END logic;
```

sel has a ***STD_LOGIC*** data type

- *What are the values for a **STD_LOGIC** data type*
- *Answer:* **{'0','1','X','Z'}**
- *Therefore, is the **WHEN OTHERS** clause necessary?*
- *Answer:* **YES**

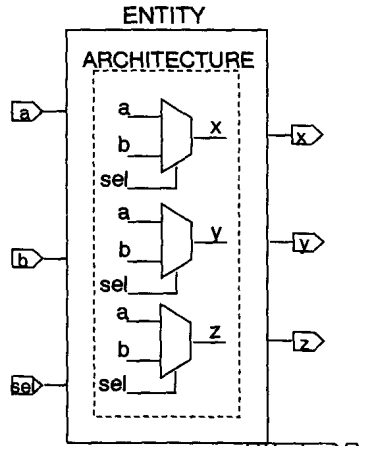
Пример модели VHDL • параллельное присвоение сигналов

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY cmpl_sig IS
PORT ( a, b, sel : IN STD_LOGIC;
       x, y, z : OUT STD_LOGIC;
END cmpl_sig;

ARCHITECTURE logic OF cmpl_sig IS
BEGIN
        -- simple signal assignment
      x <= (a AND NOT sel) OR (b AND sel);
        -- conditional signal assignment
      y <= a WHEN sel='0' ELSE
           b;
        -- selected signal assignment
      WITH sel SELECT
            z <= a WHEN '0',
                 b WHEN '1',
                 '0' WHEN OTHERS;
END logic;
```

• The signal assignments execute in parallel, and therefore the order we list the statements should not affect the outcome



Процессы

Выражение Explicit Process (явно определенный процесс)

```
--  Explicit Process Statement
PROCESS (sensitivity_list)
   Constant Declarations
   Type Declarations
   Variable Declarations
        BEGIN
        -- Sequential statement #1;
        -- ........
        -- Sequential statement #N ;
        END PROCESS;
```

– Process может подразумевать:
  - *Implied processes* – неявные *процессы*
  - *Explicit processes* – явно *определенные процессы*
– Неявный процесс состоит из:
  - Паралельных операторов определения значений сигналов
  - Выражений компонентов - Component statements
  - Чувствительность процесса отображена в правой части выражения
– Явный процесс:
  - Параллельный оператор
  - Состоит только из последовательностных операторов.

Выполнение выражения Process

```
PROCESS (a,b)
  BEGIN
    --sequential statements
  END PROCESS;

PROCESS
    BEGIN
       -- sequential statements
    WAIT ON (a,b) ;
     END PROCESS;
```

Выражение **Process** выполняется бесконечно, пока не будет прервано оператором **WAIT** или списком чувствительности (**Sensitivity List**)
Список чувствительности неявно подразумевает оператор **WAIT** в конце процесса
Process может иметь несколько операторов **WAIT**
Process не может иметь одновременно **Sensitivity List** и оператор **WAIT**

Примечание: при синтезе логических схем на **WAIT** и **Sensitivity List** накладываются дополнительные ограничения

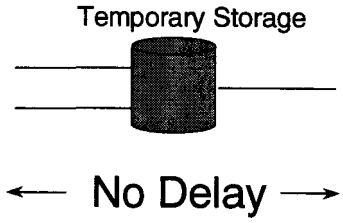
Объявление переменных

- Переменные объявляются внутри процесса **Process**
- Значения переменных присваиваются с помощью символов **:=**

Объявление переменных:

VARIABLE <*name*> : <*DATA_TYPE*> := <*value*>;
VARIABLE temp : STD_LOGIC_VECTOR (7 downto 0);

Присвоение переменной значения обрабатывается незамедлительно – задержки не возникает



Присвоение значений переменным

VARIABLE temp : STD_LOGIC_VECTOR (7 downto 0);

- All bits:
  temp := "10101010";
  temp := x"AA"; (1076-1993)
- Single bit:
  temp(7) := '1'
- Bit-slicing:
  temp (7 downto 4) := "1010";
- Single-bit: single-quote (')
- Multi-bit: double-quote (")

Пример

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY var IS
PORT     (a, b : IN  STD_LOGIC;
          y : OUT  STD_LOGIC);
END var;

ARCHITECTURE logic OF var IS
BEGIN

PROCESS (a, b)
        VARIABLE  c  :  STD_LOGIC;
BEGIN
c := a AND b;

y <= c;

END PROCESS;

END logic;
```

Variable declaration

Variable assignment

***Variable** is assigned to a **Signal** to synthesize to a piece of hardware*

Последовательные операторы

К последовательным операторам относятся
- Оператор **IF-THEN**
- Оператор **CASE**
- Оператор организации цикла **Loop**

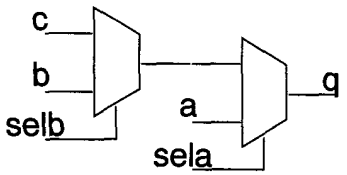
Оператор If-Then

■ Format:

```
IF <condition1> THEN
        {sequence of statement(s)}
ELSIF <condition2> THEN
        {sequence of statement(s)}
                .
                .
ELSE
        {sequence of statement(s)}
END IF;
```

■ Example:

```
PROCESS(sela, selb, a, b, c)
BEGIN
   IF sela='1' THEN
        q <= a;
   ELSIF selb='1' THEN
        q <= b;
   ELSE
        q <= c;
    END IF;
END PROCESS;
```

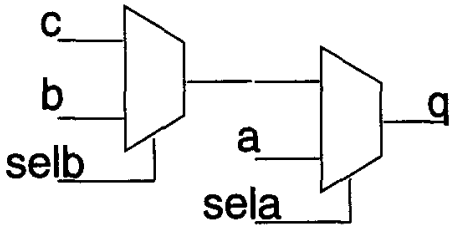


Условия проверяются в порядке записи, сверху вниз. Приоритет при выполнении.
- Первое условие, имеющее значение **true**, вызывает на исполнение соответствующую последовательность операторов.
- Если все условия имеют значение **false**, выполняется последовательность операторов в блоке "ELSE".

Аналогично установке значения сигналов.

Implied Process

```
q <= a WHEN sela = '1' ELSE
     b WHEN selb = '1' ELSE
     c;
```



Explicit Process

```
PROCESS(sela, selb, a, b, c)
BEGIN
   IF sela='1' THEN
        q <= a;
   ELSIF selb='1' THEN
        q <= b;
   ELSE
        q <= c;
    END IF;
END PROCESS;
```

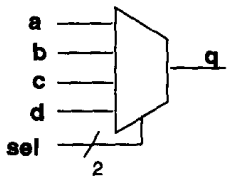
Оператор Case

■ Format:

```
CASE {expression} IS
        WHEN <condition1> =>
                {sequence of statements}
        WHEN <condition2> =>
                {sequence of statements}
                .
                .
        WHEN OTHERS =>  -- (optional)
                {sequence of statements}
        END CASE;
```

■ Example:

```
PROCESS(sel, a, b, c, d)
BEGIN
   CASE sel IS
        WHEN "00" =>
                q <= a;
        WHEN "01" =>
                q <= b;
        WHEN "10" =>
                q <= c;
        WHEN OTHERS =>
                q <= d;
   END CASE;
END PROCESS;
```



Условия обрабатываются одновременно. Приоритет отсутствует.

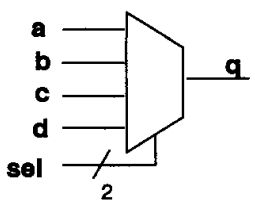
- Должны быть рассмотрены **ВСЕ** возможные варианты условий

Блок **WHEN OTHERS** обрабатывает все оставшиеся варианты условий, которые явно не описаны в операторах **Case.**

Аналогично установке значения сигналов.

Implied Process

```
WITH sel SELECT
        q <= a WHEN "00",
             b WHEN "01",
             c WHEN "10",
             d WHEN OTHERS;
```



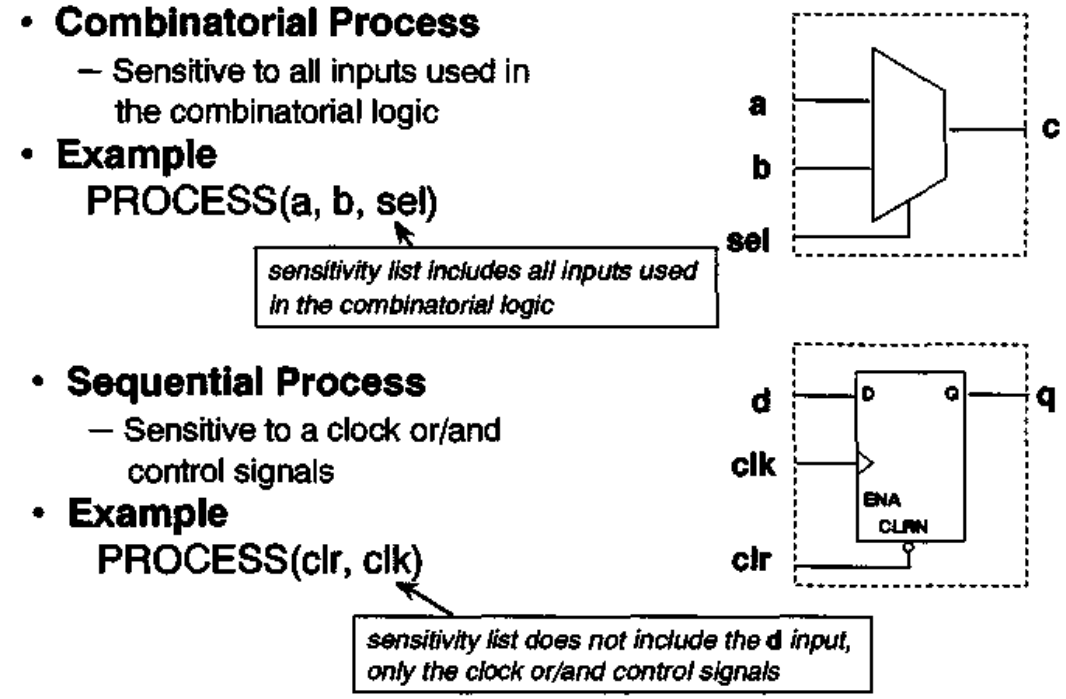
Explicit Process

```
PROCESS(sel, a, b, c, d)
BEGIN
   CASE sel IS
        WHEN "00" =>
                q <= a;
        WHEN "01" =>
                q <= b;
        WHEN "10" =>
                q <= c;
        WHEN OTHERS =>
                q <= d;
   END CASE;
END PROCESS;
```

Последовательные циклы

– Бесконечный цикл - Повторяется бесконечно, пока не будет выполнен оператор **EXIT**	[loop_label]**LOOP** --последовательно выполняемое выражение **EXIT** loop_label; **END LOOP;**
– Цикл **While** - Проверка условия в конце цикла	**WHILE** <condition> **LOOP** -- последовательно выполняемое выражение **END LOOP;**
– Цикл **FOR** - Цикл с фиксированным числом проходов	**FOR** <identifier> **IN** <range> **LOOP** -- последовательно выполняемое выражение **END LOOP;**

Два типа выражений Process



3. ПОДГОТОВКА К РАБОТЕ

1. Изучить методические указания к данной работе.
2. Освоить работу с пакетом в лаборатории кафедры.

4. РАБОЧЕЕ ЗАДАНИЕ

1. Опишите D-триггер с использованием выражения clk'event and clk='1', с использованием оператора WAIT и с использованием выражения rising_edge . Проведите синтез и моделирование с использованием командного файла. Текст описания и результаты моделирования включите в отчет.

2. Опишите D-триггер с асинхронным сбросом. Проведите синтез и моделирование с использованием командного файла. Текст описания и результаты моделирования включите в отчет.

3. С использованием цикла FOR опишите устройство сдвига на 4 разряда данных на входной 18-разрядной шине. Проведите синтез и моделирование с использованием командного файла.Текст описания и результаты моделирования включите в отчет.

4. Опишите сторожевой таймер, служащий для сброса микроконтроллеров в случае программного сбоя, если некоторое действие (обращение к сторожевому таймеру) не выполняется в течение заранее установленного промежутка времени. Счетчик состояний таймера st должен иметь возможность счета от 1 до 100, если только на очередном такте не будет активен сигнал wd_reset. По достижении максимального значения дальнейшее увеличение значения счетчика должно прекращаться и асинхронно (вне блока process) должен вырабатываться сигнал watchdog, сообщающий о срабатывании таймера. Проведите синтез и моделирование с использованием командного файла.Текст описания и результаты моделирования включите в отчет.

5. Опишите на VHDL контроллер асинхронного обмена на примере создания контроллера ISA (для 8 – разрядной шины).

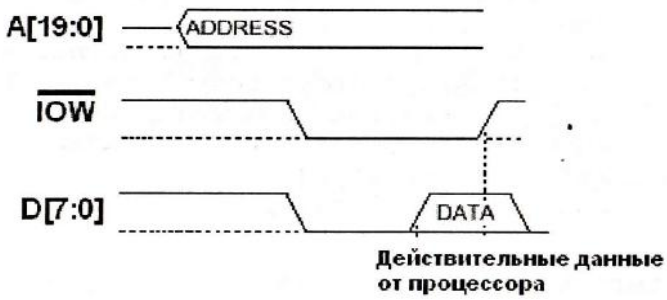
Основные сигналы шины ISA приведены в таблице (символом # обозначены сигналы, имеющие активный низкий уровень).

Таблица 1. Основные сигналы шины ISA

Сигнал	*Значение*
A[19:0]	Шина адреса
AEN	Разрешение адреса. Этот сигнал выдается контроллером ПДП и указывает, что идет выполнение цикла прямого доступа к памяти. Этот сигнал служит для блокировки декодирования адреса, чтобы адрес, используемый в ПДП, не был случайно использован как адрес устройства ввода-вывода
D[7:0]	Двунаправленная шина данных
MEMR#	Чтение памяти
MEMW#	Запись в память
IOR#	Чтение устройства ввода-вывода
IOW#	Запись в устройство ввода-вывода

Этих сигналов вполне достаточно для реализации простейшей карты расширения. Временные диаграммы циклов чтения и записи представлены на рисунках:

Временная диаграмма чтения устройства ввода-вывода



Временная диаграмма записи в устройство ввода-вывода

В шине ISA реализуется простейший асинхронный протокол обмена. Хотя системный тактовый сигнал и выведен на разъем, он не используется для управления протоколом обмена, а действительные моменты передачи данных определяются перепадами уровня на линиях ***IOR#*** и ***IOW#.***

Чтение происходит следующим образом: попадание адреса в адресное пространство карты (при активном ***AEN)*** и активный сигнал ***IOR#*** должны приводить к подключению внешних линий карты к шине данных. В остальное время выходной регистр карты ввода-вывода должен находиться в Z-состоянии.

При записи данных в карту используется подобный же подход. По положительному перепаду сигнала *10W#* данные с линий *D[7:0]* записываются в один из регистров карты, но только в том случае, если установленный адрес соответствует этому регистру.

Стандарты на подключение устройств к шине ISA выделяли для плат диапазон адресов ввода-вывода 300H-31FH. 8-разрядные карты ISA декодируют только 10 младших разрядов адреса. Реализуйте устройство, обеспечивающее запись в порты ***wa*** и ***wb,*** адресуемые по 300H и 301H соответственно, и чтение из портов ***ra*** и ***rb,*** адресуемых так же.

Проведите синтез и моделирование с использованием командного файла.Текст описания и результаты моделирования включите в отчет.

Выполните выданные преподавателем индивидуальные задания. Текст описания и результаты моделирования включите в отчет.

5. КОНТРОЛЬНЫЕ ВОПРОСЫ.

1. Что такое VHDL?
2. Понятие Behavior Modeling - моделирование поведения
3. Понятие Structural Modeling - моделирование структуры
4. Нарисуйте блок-схему порядка синтеза схемы (Typical Synthesis Design Flow)

5. Нарисуйте блок-схему порядка моделирования схемы (Typical Simulation Design Flow)
6. Назовите основные элементы проекта VHDL.
7. Что такое Entity? Синтаксис записи.
8. Что такое Architecture?
9. Объявление Architecture.Синтаксис записи.
10. Приведите базовую структуру модели на VHDL
11. Подключение библиотеки
12. Какие типы, определены в std_logic_1164 Package?
13. Что такое сигнал?
14. Как присвоить сигналам значения?
15. Варианты назначения сигналов.
16. Простое назначение сигнала
17. Операторы VHDL.
18. Как сделать определение сигнала по условию?
19. Как определить сигнал с помощью выбора варианта?
20. Что такое Process (явный и неявный)?
21. Как выполняется выражение Process?
22. Как определяются задержки сигналов?
23. Что такое событие в моделировании?
24. Основные термины моделирования процессов.
25. Как объявляются переменные?
26. Как присваиваются значения переменным?
27. Оператор If-Then в явном и неявном процессе.
28. Оператор Case в явном и неявном процессе.
29. Последовательные циклы Sequential LOOPS
30. Два типа выражений Process