

Приложения

1. Форма задания синтаксических конструкций языка VHDL

Для задания синтаксических конструкций языка VHDL используются так называемые формы Бэкуса—Наура, служащие для описания грамматик формальных языков. Для определения синтаксической конструкции используется выражение

$$\langle \text{синтаксическая конструкция} \rangle ::= \langle \text{определение} \rangle$$

Используемые фигурные скобки служат для обозначения повторения выражения, заключенного в них. Выражение может повторяться k раз ($k = 0, 1, 2, \dots$). При $k = 0$ выражение отсутствует.

Вертикальная черта служит для обозначения альтернативных случаев. Например,

$$\text{letter_or_digit} ::= \text{letter} \mid \text{digit},$$

т. е. синтаксическая конструкция символ_или_число определяется как символ (*letter*) или цифра (*digit*).

В случае, например,

$$\text{choices} ::= \text{choice} \{ \mid \text{choice} \},$$

когда в фигурных скобках стоит то же выражение (с символом «вертикальная черта»), это означает повторение того же выражения. В данном примере повторяется выражение «*choice*».

Квадратные скобки служат для обозначения необязательного выражения или необязательного слова. Например, запись

$$\text{return_statement} ::= \text{return}[\text{expression}];$$

эквивалентна записи

$$\text{return_statement} ::= \text{return}; \mid \text{return}[\text{expression}];$$

Если элемент синтаксической конструкции начинается с курсива, то выделяемая курсивом часть конструкции несет смысловую нагрузку (семантическую информацию). Выделяемое курсивом слово может не приниматься во внимание.

Например, элементы *type_name* и *subtype_name* могут рассматриваться просто как *name* (имя). В случае *type_name* подчеркивается, что это имя типа, в случае *subtype_name*, что это имя подтипа.

2. Синтаксис языка VHDL'93

Синтаксические конструкции, приводимые в данном приложении, можно найти по следующему адресу сети Internet:
<http://tech-www.informatik.uni-hamburg.de/vhdl/tools/vhdl-93.bnf>

abstract_literal ::= decimal_literal | based_literal

access_type_definition ::= access subtype_indication

actual_designator ::=
 expression
 | *signal_name*
 | *variable_name*
 | *file_name*
 | **open**

actual_parameter_part ::= parameter_association_list

actual_part ::=
 actual_designator
 | *function_name* (actual_designator)
 | *type_mark* (actual_designator)

adding_operator ::= + | - | &

aggregate ::=
 (element_association { , element_association })

alias_declaration ::=
 alias *alias_designator* [: subtype_indication] **is** *name* [signature];

alias_designator ::= identifier | character_literal | operator_symbol

allocator ::=
 new subtype_indication
 | **new** qualified_expression

```
architecture_body ::=
    architecture identifier of entity_name is
        architecture_declarative_part
    begin
        architecture_statement_part
    end [ architecture ] [ architecture_simple_name ] ;

architecture_declarative_part ::=
    { block_declarative_item }

architecture_statement_part ::=
    { concurrent_statement }

array_type_definition ::=
    unconstrained_array_definition | constrained_array_definition

assertion ::=
    assert condition
        [ report expression ]
        [ severity expression ]

assertion_statement ::= [ label : ] assertion ;

association_element ::=
    [ formal_part => ] actual_part

association_list ::=
    association_element { , association_element }

attribute_declaration ::=
    attribute identifier : type_mark ;

attribute_designator ::= attribute_simple_name
attribute_name ::=
    prefix [ signature ] ' attribute_designator [ ( expression ) ]
```

attribute_specification ::=
 attribute attribute_designator **of** entity_specification **is** expression ;

base ::= integer

base_specifier ::= B | O | X

base_unit_declaration ::= identifier ;

based_integer ::=
 extended_digit { [underline] extended_digit }

based_literal ::=
 base # based_integer [. based_integer] # [exponent]

basic_character ::=
 basic_graphic_character | format_effector

basic_graphic_character ::=
 upper_case_letter | digit | special_character | space_character

basic_identifier ::=
 letter { [underline] letter_or_digit }

binding_indication ::=
 [use_entity_aspect]
 [generic_map_aspect]
 [port_map_aspect]

bit_string_literal ::= base_specifier "bit_value"

bit_value ::= extended_digit { [underline] extended_digit }

block_configuration ::=
 for block_specification
 { use_clause }
 { configuration_item }
 end for ;

```
block_declarative_item ::=
    subprogram_declaration
    | subprogram_body
    | type_declaration
    | subtype_declaration
    | constant_declaration
    | signal_declaration
    | shared_variable_declaration
    | file_declaration
    | alias_declaration
    | component_declaration
    | attribute_declaration
    | attribute_specification
    | configuration_specification
    | disconnection_specification
    | use_clause
    | group_template_declaration
    | group_declaration

block_declarative_part ::=
    { block_declarative_item }

block_header ::=
    [ generic_clause
    [ generic_map_aspect ; ] ]
    [ port_clause
    [ port_map_aspect ; ] ]

block_specification ::=
    architecture_name
    | block_statement_label
    | generate_statement_label [ ( index_specification ) ]

block_statement ::=
    block_label :
        block [ ( guard_expression ) ] [ is ]
            block_header
            block_declarative_part
```

```
begin
    block_statement_part
end block [ block_label ] ;
```

```
block_statement_part ::=
    { concurrent_statement }
```

```
case_statement ::=
    [ case_label : ]
    case expression is
        case_statement_alternative
        { case_statement_alternative }
    end case [ case_label ] ;
```

```
case_statement_alternative ::=
    when choices =>
        sequence_of_statements
```

```
character_literal ::= ' graphic_character '
```

```
choice ::=
    simple_expression
    | discrete_range
    | element_simple_name
    | others
```

```
choices ::= choice { | choice }
component_configuration ::=
    for component_specification
        [ binding_indication ; ]
        [ block_configuration ]
    end for ;
```

```
component_declaration ::=
    component identifier [ is ]
        [ local_generic_clause ]
        [ local_port_clause ]
    end component [ component_simple_name ] ;
```

```
component_instantiation_statement ::=
    instantiation_label :
        instantiated_unit
            [ generic_map_aspect ]
            [ port_map_aspect ] ;

component_specification ::=
    instantiation_list : component_name

composite_type_definition ::=
    array_type_definition
    | record_type_definition

concurrent_assertion_statement ::=
    [ label : ] [ postponed ] assertion ;

concurrent_procedure_call_statement ::=
    [ label : ] [ postponed ] procedure_call ;

concurrent_signal_assignment_statement ::=
    [ label : ] [ postponed ] conditional_signal_assignment
    | [ label : ] [ postponed ] selected_signal_assignment

concurrent_statement ::=
    block_statement
    | process_statement
    | concurrent_procedure_call_statement
    | concurrent_assertion_statement
    | concurrent_signal_assignment_statement
    | component_instantiation_statement
    | generate_statement

condition ::= boolean_expression

condition_clause ::= until condition

conditional_signal_assignment ::=
    target <= options conditional_waveforms ;
```

```
conditional_waveforms ::=
    { waveform when condition else }
    waveform [ when condition ]

configuration_declaration ::=
    configuration identifier of entity_name is
        configuration_declarative_part
        block_configuration
    end [ configuration ] [ configuration_simple_name ] ;

configuration_declarative_item ::=
    use_clause
    | attribute_specification
    | group_declaration

configuration_declarative_part ::=
    { configuration_declarative_item }

configuration_item ::=
    block_configuration
    | component_configuration
configuration_specification ::=
    for component_specification binding_indication ;

constant_declaration ::=
    constant identifier_list : subtype_indication [ := expression ] ;

constrained_array_definition ::=
    array index_constraint of element_subtype_indication

constraint ::=
    range_constraint
    | index_constraint

context_clause ::= { context_item }
```

context_item ::=
 library_clause
 | use_clause

decimal_literal ::= integer [. integer] [exponent]

declaration ::=
 type_declaration
 | subtype_declaration
 | object_declaration
 | interface_declaration
 | alias_declaration
 | attribute_declaration
 | component_declaration
 | group_template_declaration
 | group_declaration
 | entity_declaration
 | configuration_declaration
 | subprogram_declaration
 | package_declaration

delay_mechanism ::=
 transport
 | [**reject** *time_expression*] **inertial**

design_file ::= design_unit { design_unit }

design_unit ::= context_clause library_unit

designator ::= identifier | operator_symbol

direction ::= **to** | **downto**

disconnection_specification ::=
 disconnect guarded_signal_specification **after** *time_expression* ;

discrete_range ::= *discrete_subtype_indication* | range

```

element_association ::=
    [ choices => ] expression

element_declaration ::=
    identifier_list : element_subtype_definition ;

element_subtype_definition ::= subtype_indication

entity_aspect ::=
    entity entity_name [ ( architecture_identifier ) ]
    | configuration configuration_name | open

entity_class ::=
    entity      | architecture | configuration
    | procedure | function      | package
    | type      | subtype      | constant
    | signal   | variable   | component
    | label    | literal      | units
    | group    | file

entity_class_entry ::= entity_class [ <> ]

entity_class_entry_list ::=
    entity_class_entry { , entity_class_entry }

entity_declaration ::=
    entity identifier is
        entity_header
        entity_declarative_part
    [ begin
        entity_statement_part ]
    end [ entity ] [ entity_simple_name ] ;

entity_declarative_item ::=
    subprogram_declaration
    | subprogram_body
    | type_declaration
    | subtype_declaration
    | constant_declaration

```

- | *signal_declaration*
- | *shared_variable_declaration*
- | *file_declaration*
- | *alias_declaration*
- | *attribute_declaration*
- | *attribute_specification*
- | *disconnection_specification*
- | *use_clause*
- | *group_template_declaration*
- | *group_declaration*

entity_declarative_part ::=
 { *entity_declarative_item* }

entity_designator ::= *entity_tag* [*signature*]

entity_header ::=
 [*formal_generic_clause*]
 [*formal_port_clause*]

entity_name_list ::=
 entity_designator { , *entity_designator* }
 | **others**
 | **all**

entity_specification ::=
 entity_name_list : *entity_class*

entity_statement ::=
 concurrent_assertion_statement
 | *passive_concurrent_procedure_call_statement*
 | *passive_process_statement*

entity_statement_part ::=
 { *entity_statement* }

entity_tag ::= *simple_name* | *character_literal* | *operator_symbol*

enumeration_literal ::= identifier | character_literal

enumeration_type_definition ::=
 (enumeration_literal { , enumeration_literal })

exit_statement ::=
 [label :] **exit** [loop_label] [**when** condition] ;

exponent ::= E [+] integer | E - integer

expression ::=
 relation { **and** relation }
 | relation { **or** relation }
 | relation { **xor** relation }
 | relation [**nand** relation]
 | relation [**nor** relation]
 | relation { **xnor** relation }

extended_digit ::= digit | letter

extended_identifier ::=
 \ graphic_character { graphic_character } \

factor ::=
 primary [** primary]
 | **abs** primary
 | **not** primary

file_declaration ::=
 file identifier_list : subtype_indication file_open_information] ;

file_logical_name ::= *string_expression*

file_open_information ::=
 [**open** *file_open_kind_expression*] **is** file_logical_name

file_type_definition ::=
 file of type_mark

`floating_type_definition ::= range_constraint`

`formal_designator ::=`
 `generic_name`
 | `port_name`
 | `parameter_name`

`formal_parameter_list ::= parameter_interface_list`

`formal_part ::=`
 `formal_designator`
 | `function_name (formal_designator)`
 | `type_mark (formal_designator)`

`full_type_declaration ::=`
 `type identifier is type_definition ;`

`function_call ::=`
 `function_name [(actual_parameter_part)]`

`generate_statement ::=`
 `generate_label :`
 `generation_scheme generate`
 `[{ block_declarative_item }`
 `begin]`
 `{ concurrent_statement }`
 `end generate [generate_label] ;`

`generation_scheme ::=`
 `for generate_parameter_specification`
 | `if condition`

`generic_clause ::=`
 `generic (generic_list) ;`

`generic_list ::= generic_interface_list`

generic_map_aspect ::=
 generic map (*generic_association_list*)

graphic_character ::=
 basic_graphic_character | **lower_case_letter** |
other_special_character

group_constituent ::= **name** | **character_literal**

group_constituent_list ::= **group_constituent** { , **group_constituent** }

group_template_declaration ::=
 group identifier is (*entity_class_entry_list*) ;

group_declaration ::=
 group identifier : *group_template_name*
(*group_constituent_list*);
guarded_signal_specification ::=
 guarded_signal_list : **type_mark**

identifier ::=
 basic_identifier | **extended_identifier**

identifier_list ::= **identifier** { , **identifier** }

if_statement ::=
 [*if_label* :]
 if condition then
 sequence_of_statements
 { **elsif condition then**
 sequence_of_statements }
 [**else**
 sequence_of_statements]
 end if [*if_label*] ;

incomplete_type_declaration ::= **type identifier** ;

`index_constraint ::= (discrete_range { , discrete_range })`

`index_specification ::=`
 `discrete_range`
 `| static_expression`

`index_subtype_definition ::= type_mark range <`

`indexed_name ::= prefix (expression { , expression })`

`instantiated_unit ::=`
 `[component] component_name`
 `| entity entity_name [(architecture_identifier)]`
 `| configuration configuration_name`

`instantiation_list ::=`
 `instantiation_label { , instantiation_label }`
 `| others | all`

`integer ::= digit { [underline] digit }`

`integer_type_definition ::= range_constraint`

`interface_constant_declaration ::=`
 `[constant] identifier_list : [in] subtype_indication [:=`
 `static_expression]`

`interface_declaration ::=`
 `interface_constant_declaration`
 `| interface_signal_declaration`
 `| interface_variable_declaration`
 `| interface_file_declaration`

`interface_element ::= interface_declaration`

`interface_file_declaration ::=`
 `file identifier_list : subtype_indication`

```
interface_list ::=
    interface_element { ; interface_element }

interface_signal_declaration ::=
    [ signal ] identifier_list : [ mode ] subtype_indication [ bus ] [ :=
    static_expression ]

interface_variable_declaration ::=
    [ variable ] identifier_list : [ mode ] subtype_indication [ :=
    static_expression ]

iteration_scheme ::=
    while condition
    | for loop_parameter_specification
label ::= identifier

letter ::= upper_case_letter | lower_case_letter

letter_or_digit ::= letter | digit

library_clause ::= library logical_name_list ;

library_unit ::=
    primary_unit
    | secondary_unit

literal ::=
    numeric_literal
    | enumeration_literal
    | string_literal
    | bit_string_literal
    | null

logical_name ::= identifier

logical_name_list ::= logical_name { , logical_name }
```

logical_operator ::= **and** | **or** | **nand** | **nor** | **xor** | **xnor**

loop_statement ::=
 [*loop_label* :]
 [iteration_scheme] **loop**
 sequence_of_statements
 end loop [*loop_label*] ;

miscellaneous_operator ::= ****** | **abs** | **not**

mode ::= **in** | **out** | **inout** | **buffer** | **linkage**

multiplying_operator ::= ***** | **/** | **mod** | **rem**

name ::=
 simple_name
 | operator_symbol
 | selected_name
 | indexed_name
 | slice_name
 | attribute_name

next_statement ::=
 [label :] **next** [*loop_label*] [**when condition**] ;

null_statement ::= [label :] **null** ;

numeric_literal ::=
 abstract_literal
 | physical_literal

object_declaration ::=
 constant_declaration
 | signal_declaration
 | variable_declaration
 | file_declaration

operator_symbol ::= string_literal

options ::= [**guarded**] [delay_mechanism]

package_body ::=
 package body *package_simple_name* **is**
 package_body_declarative_part
 end [package body] [package_simple_name] ;

package_body_declarative_item ::=
 subprogram_declaration
 | subprogram_body
 | type_declaration
 | subtype_declaration
 | constant_declaration
 | *shared*_variable_declaration
 | file_declaration
 | alias_declaration
 | use_clause
 | group_template_declaration
 | group_declaration

package_body_declarative_part ::=
 { package_body_declarative_item }

package_declaration ::=
 package identifier is
 package_declarative_part
 end [package] [package_simple_name] ;

package_declarative_item ::=
 subprogram_declaration
 | type_declaration
 | subtype_declaration
 | constant_declaration
 | signal_declaration
 | *shared*_variable_declaration

```
| file_declaration  
| alias_declaration  
| component_declaration  
| attribute_declaration  
| attribute_specification  
| disconnection_specification  
| use_clause  
| group_template_declaration  
| group_declaration
```

```
package_declarative_part ::=  
    { package_declarative_item }
```

```
parameter_specification ::=  
    identifier in discrete_range
```

```
physical_literal ::= [ abstract_literal ] unit_name
```

```
physical_type_definition ::=  
    range_constraint  
        units  
            base_unit_declaration  
            { secondary_unit_declaration }  
        end units [ physical_type_simple_name ]
```

```
port_clause ::=  
    port ( port_list ) ;
```

```
port_list ::= port_interface_list
```

```
port_map_aspect ::=  
    port map ( port_association_list )
```

```
prefix ::=  
    name  
    | function_call
```

primary ::=

- name
- | literal
- | aggregate
- | function_call
- | qualified_expression
- | type_conversion
- | allocator
- | (expression)

primary_unit ::=

- entity_declaration
- | configuration_declaration
- | package_declaration

procedure_call ::= *procedure_name* [(actual_parameter_part)]

procedure_call_statement ::=

[label :] procedure_call ;

process_declarative_item ::=

- subprogram_declaration
- | subprogram_body
- | type_declaration
- | subtype_declaration
- | constant_declaration
- | variable_declaration
- | file_declaration
- | alias_declaration
- | attribute_declaration
- | attribute_specification
- | use_clause
- | group_template_declaration
- | group_declaration

process_declarative_part ::=

{ process_declarative_item }

```
process_statement ::=
    [ process_label : ]
        [ postponed ] process [ ( sensitivity_list ) ] [ is ]
            process_declarative_part
        begin
            process_statement_part
        end [ postponed ] process [ process_label ] ;
```

```
process_statement_part ::=
    { sequential_statement }
```

```
qualified_expression ::=
    type_mark ' ( expression )
    | type_mark ' aggregate
```

```
range ::=
    range_attribute_name
    | simple_expression direction simple_expression
```

```
range_constraint ::= range range
```

```
record_type_definition ::=
    record
        element_declaration
        { element_declaration }
    end record [ record_type_simple_name ]
```

```
relation ::=
    shift_expression [ relational_operator shift_expression ]
```

```
relational_operator ::= = | /= | < | <= | > | >=
```

```
report_statement ::=
    [ label : ]
        report expression
        [ severity expression ] ;
```

return_statement ::=
 [**label** :] **return** [**expression**] ;

scalar_type_definition ::=
 enumeration_type_definition | **integer_type_definition**
 | **floating_type_definition** | **physical_type_definition**

secondary_unit ::=
 architecture_body
 | **package_body**

secondary_unit_declaration ::= identifier = physical_literal ;

selected_name ::= prefix . suffix

selected_signal_assignment ::=
 with expression select
 target <= **options** **selected_waveforms** ;

selected_waveforms ::=
 { **waveform when choices** , }
 waveform when choices

sensitivity_clause ::= on sensitivity_list

sensitivity_list ::= signal_name { , signal_name }

sequence_of_statements ::=
 { **sequential_statement** }

sequential_statement ::=
 wait_statement
 | **assertion_statement**
 | **report_statement**
 | **signal_assignment_statement**
 | **variable_assignment_statement**
 | **procedure_call_statement**

```
| if_statement
| case_statement
| loop_statement
| next_statement
| exit_statement
| return_statement
| null_statement
shift_expression ::=
    simple_expression [ shift_operator simple_expression ]

shift_operator ::= sll | srl | sla | sra | rol | ror

sign ::= + | -

signal_assignment_statement ::=
    [ label : ] target <= [ delay_mechanism ] waveform ;

signal_declaration ::=
    signal identifier_list : subtype_indication [ signal_kind ] [ := ex-
pression ] ;

signal_kind ::= register | bus

signal_list ::=
    signal_name { , signal_name }
    | others
    | all

signature ::= [ [ type_mark { , type_mark } ] [ return type_mark ] ]

simple_expression ::=
    [ sign ] term { adding_operator term }

simple_name ::= identifier

slice_name ::= prefix ( discrete_range )

string_literal ::= "{ graphic_character }"
```

```
subprogram_body ::=
    subprogram_specification is subprogram_declarative_part
    begin
        subprogram_statement_part
    end [ subprogram_kind ] [ designator ] ;
subprogram_declaration ::=
    subprogram_specification ;

subprogram_declarative_item ::=
    subprogram_declaration
  | subprogram_body
  | type_declaration
  | subtype_declaration
  | constant_declaration
  | variable_declaration
  | file_declaration
  | alias_declaration
  | attribute_declaration
  | attribute_specification
  | use_clause
  | group_template_declaration
  | group_declaration

subprogram_declarative_part ::=
    { subprogram_declarative_item }

subprogram_kind ::= procedure | function

subprogram_specification ::=
    procedure designator [ ( formal_parameter_list ) ]
  | [ pure | impure ] function designator [(formal_parameter_list)]
    return type_mark

subprogram_statement_part ::=
    { sequential_statement }

subtype_declaration ::=
    subtype identifier is subtype_indication ;
```

```
subtype_indication ::=
    [ resolution_function_name ] type_mark [ constraint ]

suffix ::=
    simple_name
    | character_literal
    | operator_symbol
    | all

target ::=
    name
    | aggregate

term ::= factor { multiplying_operator factor }

timeout_clause ::= for time_expression

type_conversion ::= type_mark ( expression )

type_declaration ::=
    full_type_declaration
    | incomplete_type_declaration

type_definition ::=
    scalar_type_definition
    | composite_type_definition
    | access_type_definition
    | file_type_definition

type_mark ::=
    type_name
    | subtype_name

unconstrained_array_definition ::=
    array ( index_subtype_definition { , index_subtype_definition } )
    of element_subtype_indication

use_clause ::=
    use selected_name { , selected_name } ;
```

variable_assignment_statement ::=
 [**label** :] **target** := **expression** ;

variable_declaration ::=
 [**shared**] **variable** **identifier_list** : **subtype_indication** [:= **expression**] ;

wait_statement ::=
 [**label** :] **wait** [**sensitivity_clause**] [**condition_clause**] [**time-out_clause**] ;

waveform ::=
 waveform_element { , **waveform_element** }
 | **unaffected**

waveform_element ::=
 value_expression [**after** **time_expression**]
 | **null** [**after** *time_expression*]

3. Пакет STANDARD

package standard is**type boolean is** (false, true);**type bit is** ('0', '1');**type character is** (
nul, soh, stx, etx, eot, enq, ack, bel,
bs, ht, lf, vt, ff, cr, so, si,
dle, dc1, dc2, dc3, dc4, nak, syn, etb,
can, em, sub, esc, fsp, gsp, rsp, usp,
' ', '!', '"', '#', '\$', '%', '&', "'",
'(', ')', '*', '+', ',', '-', '.', '/',
'0', '1', '2', '3', '4', '5', '6', '7',
'8', '9', ':', ';', '<', '=', '>', '?',
'@', 'A', 'B', 'C', 'D', 'E', 'F', 'G',
'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O',
'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W',
'X', 'Y', 'Z', '[', '\', ']', '^', '_',
' ', 'a', 'b', 'c', 'd', 'e', 'f', 'g',
'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o',
'p', 'q', 'r', 's', 't', 'u', 'v', 'w',
'x', 'y', 'z', '{', '|', '}', '~', del);**type severity_level is** (note, warning, error, failure);**type integer is range** -2147483647 to 2147483647;**type real is range** -1.0E38 to 1.0E38;**type time is range** -2147483647 to 2147483647**units**

fs;

ps = 1000 fs;

ns = 1000 ps;

us = 1000 ns;

ms = 1000 us;

sec = 1000 ms;

min = 60 sec;

hr = 60 min;

end units;

function now return time;
subtype natural is integer range 0 to integer'high;
subtype positive is integer range 1 to integer'high;
type string is array (positive range \diamond) of character;
type bit_vector is array (natural range \diamond) of bit;

end standard.

4. Пакет STD_LOGIC_1164

```

package STD_LOGIC_1164 is
  type std_ulogic is ( 'U', -- Uninitialized
                      'X', -- Forcing Unknown
                      '0', -- Forcing 0
                      '1', -- Forcing 1
                      'Z', -- High Impedance
                      'W', -- Weak Unknown
                      'L', -- Weak 0
                      'H', -- Weak 1
                      '-'); -- Don't care
  type std_ulogic_vector is array ( NATURAL RANGE <> ) of
std_ulogic;
  function resolved ( s : std_ulogic_vector ) RETURN std_ulogic;
  subtype UX01 is resolved std_ulogic RANGE 'U' TO '1';
  subtype std_logic is resolved std_ulogic;
  type std_logic_vector is array ( NATURAL RANGE <> ) of std_logic;
  function "and" ( l : std_ulogic; r : std_ulogic ) RETURN UX01;
  function "and" ( l, r : std_logic_vector ) RETURN std_logic_vector;
  function "and" ( l, r : std_ulogic_vector ) RETURN std_ulogic_vector;
end STD_LOGIC_1164;

package body STD_LOGIC_1164 is
  type stdlogic_1d is array (std_ulogic) of std_ulogic;
  type stdlogic_table is array (std_ulogic, std_ulogic) of std_ulogic;

  constant resolution_table : stdlogic_table := (
    -- -----
    -- | U   X   0   1   Z   W   L   H   -   | |
    -- -----
    ( 'U', 'U', 'U', 'U', 'U', 'U', 'U', 'U', 'U', 'U' ), -- | U |
    ( 'U', 'X', 'X', 'X', 'X', 'X', 'X', 'X', 'X', 'X' ), -- | X |
    ( 'U', 'X', '0', 'X', '0', '0', '0', '0', 'X', 'X' ), -- | 0 |
    ( 'U', 'X', 'X', '1', '1', '1', '1', '1', 'X', 'X' ), -- | 1 |
    ( 'U', 'X', '0', '1', 'Z', 'W', 'L', 'H', 'X', 'X' ), -- | Z |
    ( 'U', 'X', '0', '1', 'W', 'W', 'W', 'W', 'X', 'X' ), -- | W |
    ( 'U', 'X', '0', '1', 'L', 'W', 'L', 'W', 'X', 'X' ), -- | L |
    ( 'U', 'X', '0', '1', 'H', 'W', 'W', 'H', 'X', 'X' ), -- | H |
    ( 'U', 'X', 'X', 'X', 'X', 'X', 'X', 'X', 'X', 'X' ) -- | - |
  );

  function resolved ( s : std_ulogic_vector ) RETURN std_ulogic IS

```

```

variable result : std_ulogic := 'Z'; -- weakest state default
begin
  if (s'LENGTH = 1) then
    RETURN s(s'LOW);
  else
    for i in s'RANGE loop
      result := resolution_table(result, s(i));
    end loop;
  end if;
  RETURN result;
end resolved;

constant and_table : stdlogic_table := (
  -----
  -- | U   X   0   1   Z   W   L   H   -   | |
  -----
  ( 'U', 'U', '0', '1', 'Z', 'W', 'L', 'H', '0', '0' ), -- | U |
  ( 'U', 'X', '0', 'X', 'X', 'X', '0', 'X', 'X', 'X' ), -- | X |
  ( '0', '0', '0', '0', '0', '0', '0', '0', '0', '0' ), -- | 0 |
  ( 'U', 'X', '0', '1', 'X', 'X', '0', '1', 'X', 'X' ), -- | 1 |
  ( 'U', 'X', '0', 'X', 'X', 'X', '0', 'X', 'X', 'X' ), -- | Z |
  ( 'U', 'X', '0', 'X', 'X', 'X', '0', 'X', 'X', 'X' ), -- | W |
  ( '0', '0', '0', '0', '0', '0', '0', '0', '0', '0' ), -- | L |
  ( 'U', 'X', '0', '1', 'X', 'X', '0', '1', 'X', 'X' ), -- | H |
  ( 'U', 'X', '0', 'X', 'X', 'X', '0', 'X', 'X', 'X' ) -- | - |
);

function "and" ( l : std_ulogic; r : std_ulogic ) RETURN UX01 is
begin
  RETURN (and_table(l, r));
end "and";

function "and" ( l, r : std_logic_vector ) RETURN std_logic_vector is
  alias lv : std_logic_vector ( 1 TO l'LENGTH ) is l;
  alias rv : std_logic_vector ( 1 TO r'LENGTH ) is r;
  variable result : std_logic_vector ( 1 TO l'LENGTH );
begin
  if ( l'LENGTH /= r'LENGTH ) then
    assert FALSE report
      "arguments of 'and' operator are not of the same length"
    severity FAILURE;

```

```
else
  for i in result'RANGE loop
    result(i) := and_table (lv(i), rv(i));
  end loop;
end if;
RETURN result;
end "and";
```

```
function "and" ( l, r : std_ulogic_vector )
RETURN std_ulogic_vector is
alias lv : std_ulogic_vector ( 1 TO l'LENGTH ) is l;
alias rv : std_ulogic_vector ( 1 TO r'LENGTH ) is r;
variable result : std_ulogic_vector ( 1 TO l'LENGTH );
begin
if ( l'LENGTH /= r'LENGTH ) then
  assert FALSE report
    "arguments of 'and' operator are not of the same length"
  severity FAILURE;
else
  for i in result'RANGE loop
    result(i) := and_table (lv(i), rv(i));
  end loop;
end if;
RETURN result;
end "and";
end STD_LOGIC_1164;
```

Представляем Вам наши лучшие книги:



URSS

Математическая логика

Клини С. Математическая логика.

Бахтияров К. И. Логика с точки зрения информатики.

Гамов Г., Стерн М. Занимательные задачи.

Колмогоров А. Н., Драгалин А. Г. Математическая логика.

Драгалин А. Г. Конструктивная теория доказательств и нестандартный анализ.

Перминов В. Я. Развитие представлений о надежности математического доказательства.

Петров Ю. А. Логические проблемы абстракций бесконечности и осуществимости.

Бирюков Б. В., Тростников В. Н. Жар холодных чисел и пафос бесстрастной логики.

Математическое моделирование и оптимизация

Тарасевич Ю. Ю. Математическое и компьютерное моделирование.

Тарасевич Ю. Ю. Перколяция: теория, приложения, алгоритмы.

Плохотников К. Э. Математическое моделирование и вычислительный эксперимент.

Мышкис А. Д. Элементы теории математических моделей.

Блехман И. И., Мышкис А. Д., Пановко Я. Г. Прикладная математика.

Гастев Ю. А. Гомоморфизмы и модели (логико-алгебраич. аспекты моделирования).

Киселева И. А. Коммерческие банки: модели и информационные технологии.

Софиева Ю. Н., Циблин А. М. Введение в задачи и методы условной оптимизации.

Галеев Э. М. Оптимизация: теория, примеры, задачи.

Ковалев М. М. Дискретная оптимизация (целочисленное программирование).

Ковалев М. М. Матронды в дискретной оптимизации.

Балакришнан А. Введение в теорию оптимизации в гильбертовом пространстве.

Прикладная лингвистика

Хомский Н., Миллер Дж. Введение в формальный анализ естественных языков.

Венцов А. В., Касевич В. Б. Проблемы восприятия речи.

Потапова Р. К. Речь: коммуникация, информация, кибернетика.

Потапова Р. К. Тайны современного Кентавра. Речевое взаимодействие.

Потапова Р. К. Новые информационные технологии и лингвистика.

Потапова Р. К. Речевое управление роботом.

Потапов В. В. Динамика и статика речевого ритма.

Гладкий А. В. Синтаксические структуры естественного языка.

Серия «Науки об искусственном»

Финн В. К. Интеллектуальные системы и общество.

Шамис А. Л. Поведение, восприятие, мышление.

Саймон Г. Науки об искусственном.

Арбиб М. Метафорический мозг.

Тарасов В. Б. От многоагентных систем к интеллектуальным организациям.

Гаазе-Рапопорт М. Г., Поспелов Д. А. От амёбы до робота: модели поведения.

Попов Э. В. Общение с ЭВМ на естественном языке.

Редько В. Г. (ред.) От моделей поведения к искусственному интеллекту.

Варшавский В. И., Поспелов Д. А. Оркестр играет без дирижера: Размышление об эволюции некоторых технических систем и управление ими.

Представляем Вам наши лучшие книги:



URSS

Серия «Синергетика: от прошлого к будущему»

- Пенроуз Р.* **НОВЫЙ УМ КОРОЛЯ.** Пер. с англ.
Хакен Г. **Информация и самоорганизация.** Пер. с англ.
Редько В. Г. **Эволюция, нейронные сети, интеллект.**
Чернавский Д. С. **Синергетика и информация (динамическая теория информации).**
Безручко Б. П. и др. **Путь в синергетику. Экскурс в десяти лекциях.**
Данилов Ю. А. **Лекции по нелинейной динамике. Элементарное введение.**
Трубецков Д. И. **Введение в синергетику. В 2 кн.: Колебания и волны; Хаос и структуры.**
Климонтович Ю. Л. **Турбулентное движение и структура хаоса.**
Князева Е. Н., Курдюмов С. П. **Основания синергетики. Кн. 1, 2.**
Арнольд В. И. **Теория катастроф.**
Гуц А. К., Фролова Ю. В. **Математические методы в социологии.**
Малинецкий Г. Г. **Математические основы синергетики.**
Малинецкий Г. Г., Потапов А. Б. **Нелинейная динамика и хаос: основные понятия.**
Малинецкий Г. Г., Потапов А. Б., Подлазов А. В. **Нелинейная динамика.**
Капица С. П., Курдюмов С. П., Малинецкий Г. Г. **Синергетика и прогнозы будущего.**
Малинецкий Г. Г. (ред.) **Будущее России в зеркале синергетики.**
Быков В. И. **Моделирование критических явлений в химической кинетике.**
Чумаченко Е. Н. и др. **Сверхпластичность: материалы, теория, технологии.**
Баранцев Р. Г. **Синергетика в современном естествознании.**
Баранцев Р. Г. и др. **Асимптотическая математика и синергетика.**
Турчин П. В. **Историческая динамика. На пути к теоретической истории.**
Котов Ю. Б. **Новые математические подходы к задачам медицинской диагностики.**
Гельфанд И. М. и др. **Очерки о совместной работе математиков и врачей.**
Пригожин И. **Неравновесная статистическая механика.**
Пригожин И. **От существующего к возникающему.**
Пригожин И., Стенгерс И. **Время. Хаос. Квант. К решению парадокса времени.**
Пригожин И., Стенгерс И. **Порядок из хаоса. Новый диалог человека с природой.**
Пригожин И., Николис Г. **Познание сложного. Введение.**
Пригожин И., Гленсдорф П. **Термодинамическая теория структуры, устойчивости и флуктуаций.**
Суздальев И. П. **Нанотехнология: физико-химия нанокластеров, наноструктур и наноматериалов.**

Тел./факс:

**(495) 135-42-46,
(495) 135-42-16,**

E-mail:

URSS@URSS.ru

http://URSS.ru

Наши книги можно приобрести в магазинах:

- «Библио-Глобус» (м. Лубянка, ул. Мясницкая, 6. Тел. (495) 625-2457)
 «Московский дом книги» (м. Арбатская, ул. Новый Арбат, 8. Тел. (495) 203-8242)
 «Молодая гвардия» (м. Полянка, ул. Б. Полянка, 28. Тел. (495) 238-5001, 780-3370)
 «Дом научно-технической книги» (Ленинский пр-т, 40. Тел. (495) 137-6019)
 «Дом книги на Ладонской» (м. Бауманская, ул. Ладонская, 8, стр. 1. Тел. 267-0302)
 «Гнозис» (м. Университет, 1 гум. корпус МГУ, комн. 141. Тел. (495) 939-4713)
 «У Кентавра» (РГТУ) (м. Новослободская, ул. Чайнова, 15. Тел. (499) 973-4301)
 «СПб. дом книги» (Невский пр., 28. Тел. (812) 311-3954)

Уважаемые читатели! Уважаемые авторы!

Наше издательство специализируется на выпуске научной и учебной литературы, в том числе монографий, журналов, трудов ученых Российской академии наук, научно-исследовательских институтов и учебных заведений. Мы предлагаем авторам свои услуги на выгодных экономических условиях. При этом мы берем на себя всю работу по подготовке издания — от набора, редактирования и верстки до тиражирования и распространения.



URSS

Среди вышедших и готовящихся к изданию книг мы предлагаем Вам следующие:

- Росс Эшби У.* Введение в кибернетику.
Бир С. Мозг фирмы.
Бир С. Кибернетика и менеджмент.
Бир С. Наука управления.
Ворожцов А. В. Путь в современную информатику.
Кронрод А. С. Беседы о программировании.
Гуц А. К. Комплексный анализ и кибернетика.
Закревский А. Д. Параллельные алгоритмы логического управления.
Закревский А. Д. Логические уравнения.
Закревский А. Д. Логика распознавания.
Поваров Г. Н. Ампер и кибернетика.
Магазов С. С. Лекции и практические занятия по технологии баз данных.
Бриллюэн Л. Научная неопределенность и информация.
Карабутов Н. Н. Адаптивная идентификация систем: информационный синтез.
Григорьев В. А. Человек как неформальная интеллектуальная система.
Цыгичко В. Н. Прогнозирование социально-экономических процессов.
Федулов А. А., Федулов Ю. Г., Цыгичко В. Н. Введение в теорию статистически ненадежных решений.
Лефевр В. А., Смолян Г. Л. Алгебра конфликта.
Болотов А. А., Гаишков С. Б., Фролов А. Б., Часовских А. А. Элементарное введение в эллиптическую криптографию: Алгебраические и алгоритмические основы.
Болотов А. А., Гаишков С. Б., Фролов А. Б. Элементарное введение в эллиптическую криптографию: Протоколы криптографии на эллиптических кривых.
Полевой Д. В. Таблицы в системах обработки документов.
Емельянов С. В. (гл. ред.) Информационные технологии и вычислительные системы.
Афанасьев А. П. Продолжение траекторий в оптимальном управлении.
Афанасьев А. П. Динамические системы управления с неполной информацией.
Арлазаров В. Л. (ред.) Организационное управление и искусственный интеллект.
Лапин Н. И. (ред.) Социальная информатика: основания, методы, перспективы.
Черешкин Д. С. (ред.) Проблемы кибербезопасности информационного общества.
Вайдлих В. Социодинамика: системный подход к математическому моделированию социальных наук.
Каплун А. Б., Морозов Е. М., Олферьева М. А. ANSYS в руках инженера.

По всем вопросам Вы можете обратиться к нам:
 тел./факс (495) 135-42-16, 135-42-46
 или электронной почтой URSS@URSS.ru
 Полный каталог изданий представлен
 в интернет-магазине: <http://URSS.ru>

Научная и учебная
литература

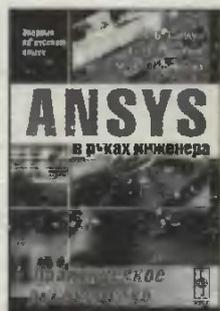


Доктор технических наук, профессор, автор 240 научных работ, в том числе 9 монографий. Его основные работы относятся к теории дискретных устройств и автоматизации проектирования дискретных устройств и цифровых сверхбольших интегральных схем, применению методов искусственного интеллекта в системах автоматизированного проектирования (САПР).

По его мнению, успешное развитие микроэлектроники связано с внедрением в практику новых маршрутов проектирования на основе языков высокого уровня, таких как VHDL, созданием новых эффективных параллельных методов и архитектур для решения вычислительных и оптимизационных задач, использованием методов искусственного интеллекта. При этом он руководствуется тем, что разработка отечественных САПР цифровых схем должна вестись с учетом их интеграции с зарубежными САПР.

Автор уделяет большое внимание подготовке студентов и специалистов-разработчиков, читает курсы лекций в Белорусском государственном университете информатики и радиоэлектроники, является инициатором создания русскоязычного Интернет-сайта по языку VHDL.

Наше издательство предлагает следующие книги:



63471 Библио. Наука. Минск. 71
Изд. 3



Цена 229.00 ISBN 63471

71

НАУЧНАЯ И УЧЕБНАЯ ЛИТЕРАТУРА

Тел./факс: 7 (495) 135-42-16
Тел./факс: 7 (495) 135-42-46



URSS

E-mail: URSS@URSS.ru
Каталог изданий в Интернете: <http://URSS.ru>