Programovací jazyk VHDL

Na začátku přidat knihovní podporu:

```
library IEEE;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
```

1. Datové typy

Proměnné mohou být uvnitř PROCESS, FUNCTION nebo PROCEDURE.

VARIABLE jméno : typ (:= hotnota);
Přiřazení: proměnná := hodnota; signál <= hodnota;

- boolean – True, False
- integer – celá čísla v rozsahu -2^{32}–2^{32};
  Jiné číselné soustavy vyjádřeny pomocí:

  základ soustavy # hodnota #

  Př.: `42` (desítkově), `2#101010#`, `16#2A#`
  Pro lepší čitelnost možnost oddělovat „_“:

  ```
  B"1010_1111_1100", 16#A_F_C#
  ```

  Zadání rozsahu datového typu:

  ```
  variable state : integer range 0 to 4;
  ```

- natural – celá čísla v rozsahy 0–2^32
- real – `2.75`, `2#10.11#`, `16#2.C#`
- character – `'A'`, `'H'`
- string – `"ahoj vole!"`
- bit – `'0'`, `'1'`
- bit_vector – `B"0111011"` (bin.), `X"3B"` (hexa.), `7"73"` (oct.)
- std_ulogc, std_ulogic_vector
- std_logic, std_logic_vector – mají definovanou rezoluční funkci při setkání dvou hodnot na jednom drátu
  Devítihodnotová logika:

  ```
  'U', 'X', '0', '1', 'Z', 'W', 'L', 'H', '-'
  ```

1.1. Uživatelsky definovaný datový typ

```
TYPE bit IS ('0', '1')
TYPE my_integer IS RANGE -32 TO 32;
```

1.2. Výčtový typ

```
TYPE color IS (red, green, blue);
```

1.3. Pole

TYPE název IS ARRAY (specifikace) OF datový typ;

```
TYPE matrix IS ARRAY (0 TO 2)
     OF std_logic_vector(7 downto 0);

matrix(0)(7 DOWNTO 3) <= "1010";
matrix <= ((others => '0'),
           (others =>'0'),
           "11110000");
```

1.4. Signály

SIGNAL jméno : typ (:= počáteční hodnota);

```
SIGNAL bus : std_logic_vector(3 downto 0)
                                    := "0101";
SIGNAL bus : std_logic_vector(3 downto 0)
                          := ('0','1','0','1');
```

1.5. Agregační operátory

my_vector : bit_vector (3 down 0);

- my_vector(3) <= '1';
  my_vector(2) <= A and B;
  my_vector(1) <= '0';
  my_vector(0) <= A xor B;
  Možno napsat nahuštěně:
  my_vector('1', A and B, '0', A xor B);

- my_vector'('0', '1', '0', '0'); – jinak:
  my_vector'(3 => '1', others => '0');

1.6. Atributy

1.6.1. Atributy polí a vektorů

```
signal c : in bit_vector(7 downto 0) := "10101010";
                std_logic_vector
```

- c'LOW = 0 – nejnižší index pole,
- c'HIGH = 7 – nejvyšší index pole
- c'LEFT = 7 – nejlevější index pole
- c'RIGHT = 0 – nejpravější index pole
- c'RANGE = (7 downto 0) – rozsah vektoru
- c'LENGTH = 8 – vrátí velikost vektoru
- c'REVERSE_RANGE = (0 to 7) – vrátí rozsah vektoru v obráceném pořadí

1.6.2. Atributy signálů

- s'EVENT – pravda pokud se signál mění
- s'STABLE – pravda pokud se signál nemění
- s'ACTIVE – pravda pokud s='1'

1.6.3. Uživatelské atributy

Deklarace: ATRIBUTE název: typ;
Specifikace: ATRIBUTE název OF cíl: třída IS hodnota;

```
ATRIBUTE number_of_inputs : INTEGER;
ATRIBUTE number_of_inputs OF xor3: SIGNAL IS 3;

inputs <= xor3'number_of_inputs; -- vrátí hodnotu 3
```

1.7. Konstanty

Mohou být definovány v deklarační části ENTITY, ARCHITEKTURE nebo PACKAGE.
CONSTANT název : typ := hodnota;

1.8. Struktura:

TYPE název IS RECORD
a_1 : typ (:= hodnota);
...
a_n : typ (:= hodnota);
END RECORD;

1.9. Konverze datových typů

Konverzní funkce se nacházejí v balíku

```
use ieee.std_logic_arith.all;
```

- conv_integer(hodnota)
- conv_unsigned(hodnota)
- conv_signed(hodnota, bitů)
- conv_std_logic_vector(hodnota, bitů)

2. Priority operátorů

	Stejná
Nejvyšší	**, abs, not
↓	*, /, mod, rem
↓	unární + a −
↓	+, −, & (zřetězení)
↓	sll, srl, sla, sra, rol, ror,
↓	=, /=, <, <=, >, >=
Nejnižší	and, or, nand, nor, xor, xnor

3. Definice entity a jejího chování

ENTITY název IS
  GENERIC (název parametru : typ := hodnota);
  PORT (
    $port_1$: chování typ;
    ...
    $port_n$: chování typ);
END název;

Není-li u parametrů zadána hodnota, nutno zadat při vložení komponenty pomocí GENERIC MAP.

Chování portů: IN, OUT, INOUT, BUFFER.

3.1. Vnitřní popis entity

ARCHITECTURE definuje chování, entita může mít víc popisů.

ARCHITECTURE název OF entita IS
  deklarační část pro komponenty, signály,...
BEGIN
  paralelní prostředí...
END název;

4. Stavové příkazy

(label:) **PROCESS** (citlivostní seznam)
  variable název : typ (rozsah) (:= počáteční hodnota);
BEGIN
  WAIT UNTIL ...; −− místo citl. seznamu
  ...sekvenční kód ...
END PROCESS;

(label:) **WHILE** podmínka LOOP
  ...
END LOOP;

(label:) **IF** podmínka THEN
  ...
  ELSIF podmínka THEN
    ...
  ELSE
    ...
END IF;

(label:) **CASE** výraz IS
  WHEN $volba_1$ => ...;
  WHEN $volba_2$ => ...;
−− chování pro ostatní volby
  WHEN OTHERS => ...;
END CASE;

(label:) **FOR** i IN rozsah LOOP
  ...
END LOOP;

rozsah = a TO b, b DOWNTO a, ...

(label:) **LOOP**
  ...
  EXIT label WHEN podmínka;
END LOOP;

- **EXIT** LABEL (WHEN podmínka); – bez labelu ukončí nejbližší uzavírající cyklus
- **NEXT** LABEL (WHEN podmínka); – skočí na další iteraci cyklu

5. Paralelní prostředí

label: **FOR** i IN rozsah **GENERATE**
  paralelní příkaz;
END GENERATE;

label: **IF** podmínka **GENERATE**
  paralelní příkaz;
END GENERATE;

5.1. Paralelní stavové příkazy

- WHEN (analogie IF):

  signál <= $výraz_1$ when $volby_1$ else,
    $výraz_2$ when $volby_2$ else,
    ...
    $výraz_n$;

- WITH (analogie CASE):

  WITH testovaný výraz SELECT
  signál <= $výraz_1$ WHEN $volby_1$,
    $výraz_2$ WHEN $volby_2$,
    ...
    $výraz_n$ WHEN OTHERS;
    (UNAFFECTED WHEN OTHERS;)

6. Strukturní popis

Deklarace komponenty (v deklarační části architektury):

COMPONENT název
  (GENERIC ...;) −− parametry
  PORT (...); −− porty
END COMPONENT;

Zapojení:

LABEL: název komponenty
  GENERIC MAP (parametr => hodnota) −− aktuální param
  PORT MAP (mapování,...); −− popis struktury

Mapování:

1. Port komponenty => port nadřazené entity
2. Port komponenty => signál architektury
3. Poziční mapování, nadřazené signály v pořadí portů entity

BLOCK slouží k seskupování paralelních příkazů, může obsahovat lokální deklarace.

label: BLOCK (strážený signál)
  deklarace
BEGIN
  ...
END BLOCK label;

7. Vlastní knihovny

```
USE work.název.all;
```

PACKAGE název IS
  deklarace...
END název;
PACKAGE BODY název IS
  popis funkcí a procedur...
END;

8. Procedury a funkce

Mohou být deklarovány uvnitř dekl. části ENTITY, ARCHITECTURY a PACKAGE.

FUNCTION název (parametr : typ; ...) RETURN typ IS
  deklarace
BEGIN
  sekvenční výrazy
END název;

PROCEDURE název (parametr : typ; ...) IS
BEGIN
  sekvenční výrazy
END PROCEDURE;

Parametry procedury mohou být IN, OUT nebo INOUT.

9. Zpoždění pro simulace

- wait for *čas* ns/us/ms/sec; wait on *signál* (místo citlivostního seznamu v procesu); wait until *výraz*;
- y <= NOT (a AND b) AFTER 10 ns; – definuje setrvačné zpoždění
- y <= TRANSPORT b AFTER 10 ns; – definuje transportní zpoždění