Jazyky HDL a VHDL

Prvé jazyky HDL (CDL, ISP, AHPL)

- na verifikáciu architektúry návrhu
- neumožňovali modelovať návrhy s vysokým stupňom presnosti
- nepresný časový model
- jazykové konštrukcie implikujú konkrétnu štruktúru technických prostriedkov

Novšie jazyky (Verilog, UDL/I, VHDL)

- univerzálnejší model času
- nevedú na konkrétnu štruktúru technických prostriedkov
- hlavné využitie: na *dokumentáciu* a *modelovanie* návrhu
- podporené simulátorom (náhrada prototypu)

VHDL (VHSIC Hardware Description Language)

VHSIC (Very High Spead Integrated Circuits)

1983 – 85 (Intermetrics, IBM a Texas Instrumets)

1987 VHDL oficiálne uznaný za štandard (IEEE 1076-1987).

Modelovacie schopnosti VHDL

Úroveň abstrakcie	Reprezentácia správania sa	Stavebné prvky štruktúry
systémová	špecifikácia výkonnosti	počítač, disk, radar...
integrovaných obvodov	algoritmus	procesor, RAM, ROM, UART, paralelný port ...
registrov	tok údajov	register, ALU, počítadlo, MUX, ROM ...
hradiel	Boolovské rovnice	AND, OR, XOR, FF
obvodová	diferenciálne rovnice	tranzistor, R, L, C
polovodičov	žiadna	geometrické obrazce

Výhody jazyka VHDL

Je všeobecne prístupný

Podporuje rôzne návrhové metodológie a technológie:

Je nezávislý na technológii

Poskytujú široké opisné možnosti

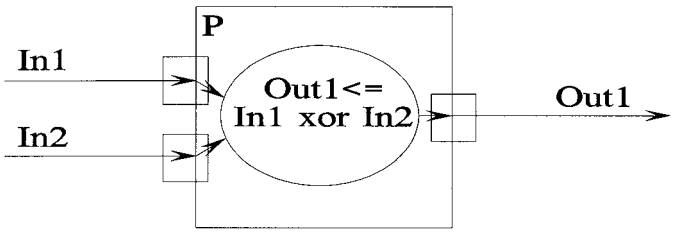
Umožňuje nezávislý návrh subsystémov

Podporuje návrh rozsiahlych systémov a opätovné použitie návrhu

Model číslicového zariadenia

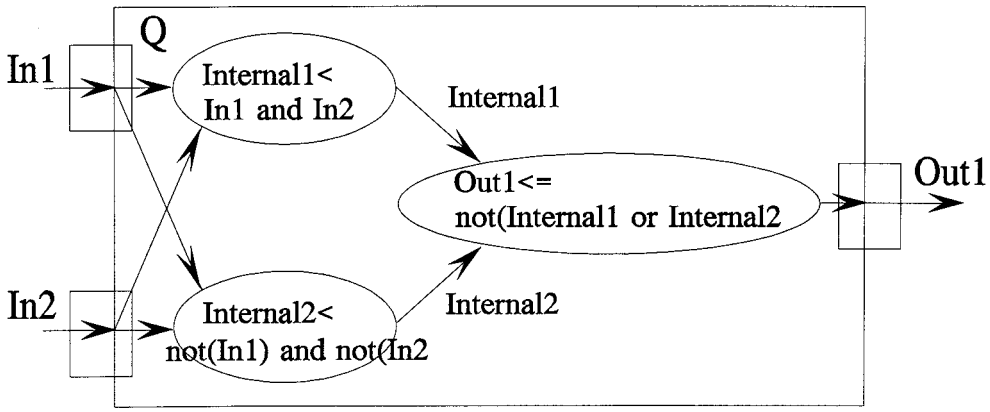
VHDL model pozostáva z 3 nezávislých modelov: **modelu správania sa**, **modelu časovania** a **modelu štruktúry**.

Model správania sa



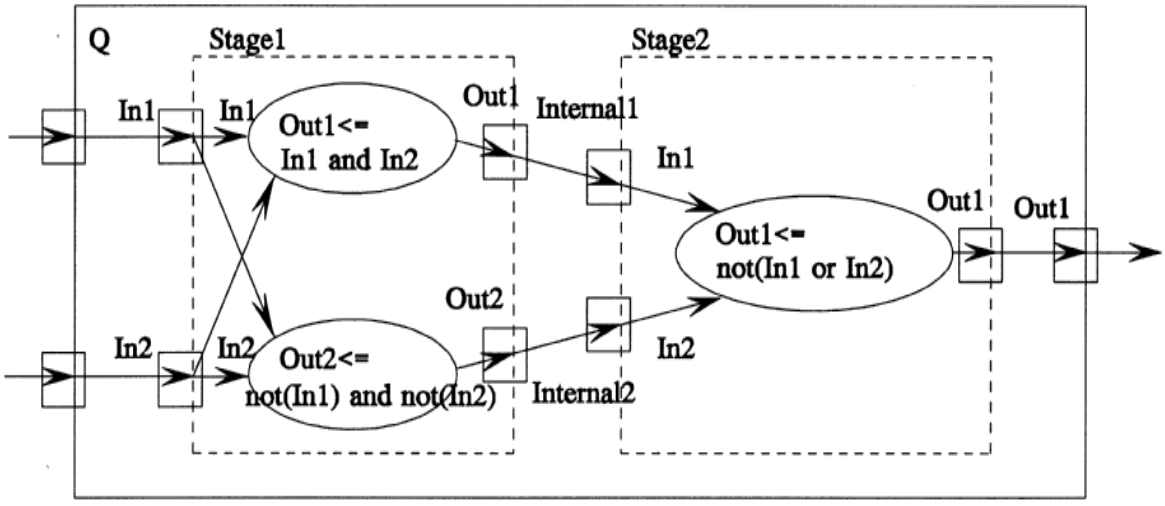
Model správania sa diskrétneho systému P vo VHDL:

```
-- návrhová entita:
-- deklarácia entity (entity declaration)
entity Obvod_XOR is
  port (In1:in Bit; In2:in Bit;
        Out1:out Bit);
end Obvod_XOR;

-- telo architektúry (architecture body)
architecture Spravanie_P of Obvod_XOR is
begin
  Out1 <= In1 xor In2;
end Spravanie_P;
```




```
-- deklarácia entity (entity declaration)
entity Obvod_XOR is
  port (In1:in Bit; In2:in Bit;
        Out1:out Bit);
end Obvod_XOR;

-- telo architektúry (architecture body)
architecture Spravanie_Q of Obvod_XOR is
  signal Internal1, Internal2: Bit;
begin
    Internal1 <= In1 and In2;
    Internal2 <= not(In1) and not(In2);
    Out1 <= not (Internal1 or Internal2);
end Spravanie_Q;
```

Model štruktúry

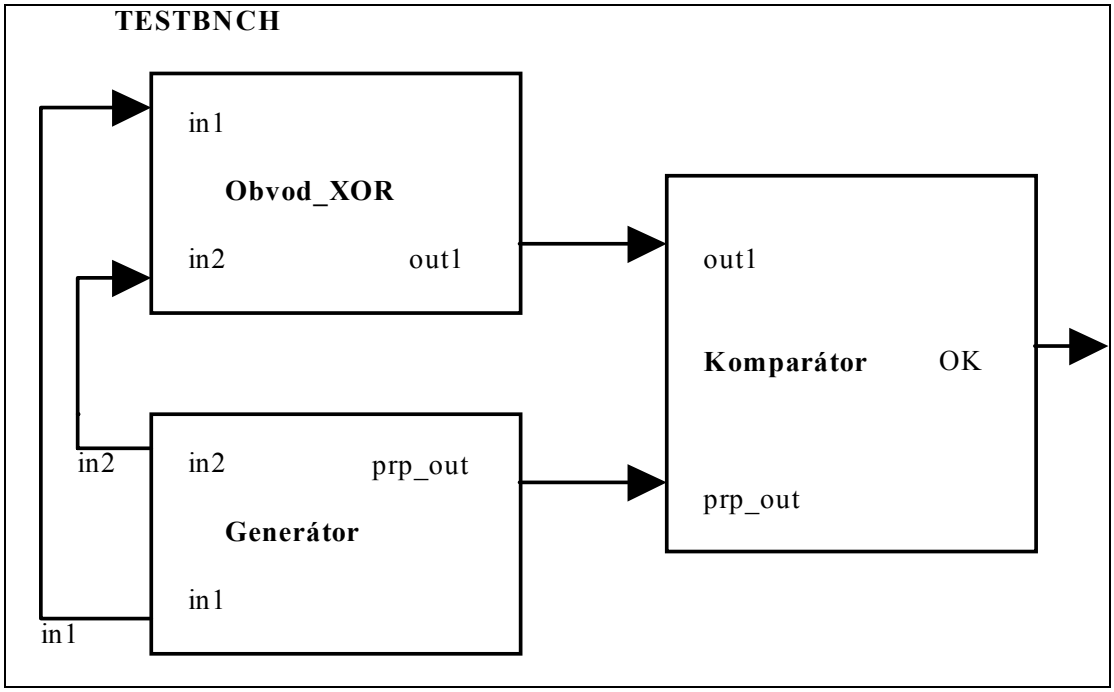



```
-- návrhová entita podsystému Stage_1
entity Stage_1 is
  port ( In1:in Bit; In2:in Bit;
       Out1:out Bit; Out2:out Bit );
end Stage_1;
architecture Spravanie of Stage_1 is
begin
    Out1 <= In1 and In2;
    Out2 <= not(In1) and not(In2);
```

```
end Spravanie;

-- návrhová entita podsystému Stage_2
entity Stage_2 is
  port ( In1:in Bit; In2:in Bit;
       Out1:out Bit );
end Stage_2;
architecture Spravanie of Stage_2 is
begin
    Out1 <= not (In1 or In2);
end Spravanie;

-- návrhová entita diskrétneho systému Q
entity Obvod_XOR is
  port (In1:in Bit; In2:in Bit;
        Out1:out Bit);
end Obvod_XOR;
architecture Struktura_Q of Obvod_XOR is
  --deklarácia vnútorných signálov
  signal Internal1, Internal2: Bit;
  --lokálna deklarácia komponentov
  component Stage_1
  port ( In1:in Bit; In2:in Bit;
       Out1:out Bit; Out2:out Bit );
  end component;
  component Stage_2
  port (In1:Bit; In2:Bit; Out1:out Bit);
  end component;
begin
  --príkazy vytvorenia inštancií komponentov
  S1:Stage_1
```

```
   port map ( In1=>In1, In2=>In2, Out1=>Internal1, Out2=>Internal2  );
  S2:Stage_2
   port map ( In2=>Internal1, In1=>Internal2, Out1=>Out1  );
end Struktura_Q;
```

Testovací modul



Testovacia entita pre obvod XOR:

```
entity TESTBNCH is
end TESTBNCH;

architecture stimulus of TESTBNCH is
component Obvod_XOR is
   port ( in1 : in BIT;      -- in1
          in2 : in BIT;      -- in2
          out1 : out BIT);
   end component;
for DUT: Obvod_XOR use entity Work.Obvod_XOR(Spravanie_Q);
signal   in1, in2, out1: BIT :='0';
begin
   DUT: Obvod_XOR port map (in1, in2, out1);
   STIMULUS1:   in1 <= '1' after 5 ns, '0' after 20 ns;
   STIMULUS2:   in2 <= '1' after 15 ns, '0' after 30 ns;
end stimulus;
```

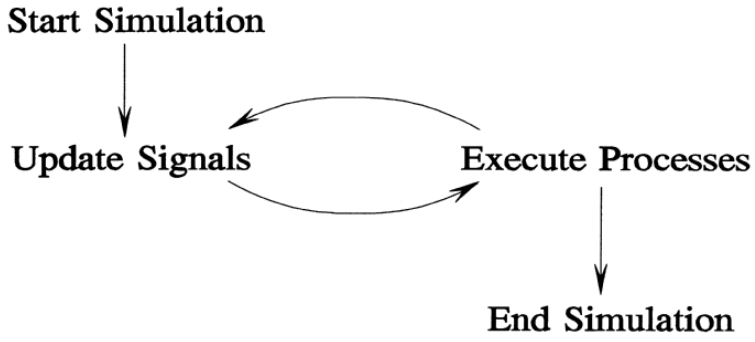
Model času

spúšťacie signály (triggering signals) ➔ **podnet-reakcia**

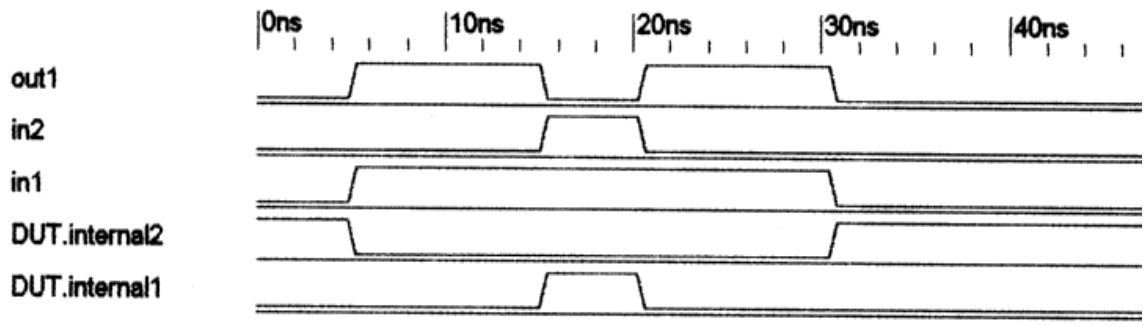
transakcia - s daným **oneskorením**

súbor transakcií pre daný signál – ***ovládač***

Štandard VHDL definuje dvojstupňový model času nazývaný ***simulačným cyklom*** (simulation cycle).



9/23/1997 13:45 C:\acc-eda\WaveView\obvod_xor.tim



	Procesy:	**Ovládače:**
	Internal1 <= In1 **and** In2;	o_Internal1
	Internal2 <= **not**(In1) **and not**(In2);	o_Internal2
	Out1 <= **not** (Internal1 **or** Internal2);	o_Out1
STIMULUS1:	in1 <= '1' **after** 5 ns, '0' **after** 20 ns;	o_in1
STIMULUS2:	in2 <= '1' **after** 15 ns, '0' **after** 30 ns;	o_in2

sim. čas	in1	in2	out1	internal1	internal2	o_Internal1	o_Internal2	o_Out1	o_in1	o_in2
0 ns	'0'	'0'	'0'	'0'	'0'	('0', δ)	('1', δ)	('1', δ)	('1', 5) ('0', 20)	('1', 15) ('0', 30)
+ δ	-	-	'1'	'0'	'1'			('0', δ)		
+ δ	-	-	'0'	-	-					
5 ns	'1'	-	-	-	-	('0', δ)	('0', δ)			
+ δ	-	-	-	'0'	'0'			('1', δ)		
+ δ	-	-	'1'	-	-					
15 ns	-	'1'	-	-	-	('1', δ)	('0', δ)			
+ δ	-	-	-	'1'	'0'			('0', δ)		
+ δ	-	-	'0'	-	-					
20 ns	'0'	-	-	-	-	('0', δ)	('0', δ)			
+ δ	-	-	-	'0'	'0'			('1', δ)		
+ δ	-	-	'1'	-	-					
30 ns	-	'0'	-	-	-	('0', δ)	('1', δ)			
+ δ	-	-	-	'0'	'1'			('0', δ)		
+ δ	-	-	'0'	-	-					

Údajové typy a objekty

Objekty

Objekt je prostriedok, ktorý môže uchovávať hodnotu (napr.signál, port a podobne). Vo VHDL existujú 3 triedy objektov:

- signály,
- premenné a
- konštanty.

Konštanta – jej hodnota sa špecifikuje v čase deklarácie (pomocou literálu) a počas simulácie sa nemení

Premenná – jej hodnota sa môže v čase simulácie viac krát meniť

- hodnotu nadobúda okamžite po vykonamí príkazu priradenia
- môže byť deklarovaná iba v príkaze process a v podprograme

Signál – jeho hodnota sa môže v čase simulácie viac krát meniť

- reprezentuje hodnotu na skutočnej údajovej linke obvodu, preto sa jeho hodnota nikdy nemení okamžite po vykonaní príkazu priradenia, ale najskôr na začiatku nasledujúceho simulačného času
- je možné naplánovať niekoľko zmien hodnoty signálu jedným príkazom priradenia

Najdôležitejšie pravidlá typovej kompatibility:

- Typ výrazu priradeného nejakému objektu musí byť zhodný s typom tohto objektu.
- Typy operandov väčšiny preddefinovaných operátorov musia byť totožné.
- Typ aktuálneho objektu musí byť zhodný s typom príslušného formálneho objektu.

Príklady deklarácií objektov:

constant ROM_size: Integer:=16#FFFF#;
variable In_fetch: Boolean;
signal Enable: Bit;
signal CLK, CLEAR: Bit:='0';
variable Address: Integer **range** 0 **to** ROM_size;

Modifikácia Backus-Naurovej Formy:

- Symbol "::=" definuje nahradenie ľavej strany vytvárajúceho pravidla pravou stranou. Na ľavej strane vytvárajúceho pravidla je syntaktická kategória, na pravej nahrádzajúce pravidlo.
- Zvislá čiara oddeľuje *alternatívne* prvky na pravej strane vytvárajúceho pravidla.
- Hranaté zátvorky uzatvárajú *voliteľné* prvky na pravej strane vytvárajúceho pravidla.
- Krútené zátvorky uzatvárajú *opakujúci sa* prvok, alebo prvky na pravej strane vytvárajúceho pravidla.

Syntax deklarácie objektu vyzerá nasledovne:

object_declaration ::=
object_class identifier_list: subtype_indication [:= expression] ;
|**signal** identifier_list: subtype_indication signal_kind [:= expression] ;

object_class ::= **constant|variable|signal**
identifier_list ::= identifier {, identifier}
subtype_indication ::= *type*_name [constraint]
|*subtype*_name [constraint]
signal_kind ::= **bus|register**

Údajové typy

Údajový typ určuje hodnoty, ktoré môže objekt tohto typu nadobúdať.
Podtyp – je typ s obmedzením rozsahu. Hodnota patrí do daného podtypu, ak je legitímnou hodnotou typu a spĺňa obmedzenie.
Bázový typ – je typ, od ktorého bol podtyp odvodený

V jazyku VHDL sú definované 4 triedy údajových typov:
1) **skalárne** – ich hodnoty sa nedajú rozdeliť na atomické
– hodnoty sa dajú usporiadať
základné | a) **vymenovaný** (enumeration) | diskrétne
b) **celočíselný** (integer) | numerické
c) **reálny** (floating point) | numerické
d) **fyzikálny** | numerické

2) **zložené** – hodnoty sa dajú rozdeliť na atomické
a) **polia**
b) **záznamy**

3) **typy prístupu**
4) **typy súborov**

Syntax deklarácie typu:

```
type_declaration    ::=   type identifier is type_definition;
type_definition     ::=   scalar_type_definition
                          |composite_type_definition
                          |access_type_definition
                          |file_type_definition
scalar_type_definition ::=      integer_or_floating_type_def
                          |enumeration_type_definition
                          |physical_type_definition
integer_or_floating_type_def ::= range_constraint
range_constraint    ::=   range range_specification
range_specification       ::=   range_attribute_name
                          |expression direction expression
direction ::= to | downto
enumeration_type_def ::= (enumeration_literal {, enumeration_literal})
enumeration_literal  ::=  identifier
                          | character_literal
physical_type_definition ::=    range_constraint
                                units
                                      primary_unit_declaration
                                        {secondary_unit_declaration}
```

end units [*physical_type*_simple_name]
primary_unit_declaration::= idetifier
secondary_unit_declaration ::= idetifier = physical_literal;

Príklady deklarácií skalárnych údajových typov:
Deklarácie celočíselného typu
type Byte **is range** -128 **to** 127;
type Bit_position **is range** 7 **downto** 0;
type Decimal **is range** -1E9 **to** 1E9;

Deklarácie reálneho typu
type Frac **is range** -1+0.1E-1 **to** 1-0.1E10;

Deklarácie vymenovaného typu

type Two_level_logic **is** ('0','1') ;

type Three_level_logic **is** ('0','1','Z') ;

type Four_level_logic **is** ('X','0','1','Z');

Deklarácie fyzikálneho typu

type Resistance **is range** 1 **to**10E9

units

ohm; -- the base unit

kohm = 1000 ohm; -- the sec. unit

end units ;

Preddefinované údajové typy:

-- Integer
-- VHDL umožňuje každej implementácii definovať vlastný
-- rozsah, ale rozsah by mal byť aspoň
-- -(2**31-1) to (2**31-1)
type Integer **is range** -2147483648 to 2147483647;

-- Real
-- VHDL umožňuje každej implementácii definovať vlastný
-- rozsah, ale rozsah by mal byť aspoň -1E38 to 1E38
-- a presnosť najmenej 6 desatinných miest

```
type Real is range
     -16#0.7FFF_FF8#E+32 to 16#0.7FFF_FF8#E+32

-- Boolean
type Boolean is (False, True);

-- Bit
type Bit is ('0', '1');

-- Severity_level
type Severity_level is (Note, Warning, Error, Failure);

-- Character
-- 128 ASCII znakov; biele znaky sú
-- reprezentované identifikátormi
type Character is (...BEL, BS,...
                ...' ', '!',..
                ...'a', 'b',...);


-- Time
-- Skutočný rozsah typu Time sa môže líšiť v závislosti od implementácie
-- minimálne -(2**31-1) do (2**31-1) femtosekúnd.
type Time is range -(2**31-1) to (2**31-1)
  units
    fs;-- femtosecond
    ps=1000 fs; -- picosecond
    ns=1000 ps; -- nanosecond
    us=1000 ns; -- mikrosecond
    ms=1000 us; -- milisecond
    sek=1000 ms; --second
    min=60sec; --minute
    hr=60min; -- hour
  end units
```

Literály

VHDL má 7 druhov literálov: *celočíselné*, *reálne*, *znakové*, *reťazcové*, *literály bitových reťazcov, fyzikálne literály a vymenované literály*.

Reálne literály a celočíselné literály :

```
2               -- celočíselný literál so základom 10
0               -- celočíselný literál so základom 10
7754236         -- celočíselný literál so základom 10
10E4            -- celočíselný literál so základom 10
16#D2#          -- celočíselný literál so základom 16 (HEX)
8#720#          -- celočíselný literál so základom 8 (OCTAL)
2#11010010#         -- celočíselný literál so základom 2 (BINARY)
1.0             -- reálny literál so základom 10
0.0             -- reálny literál so základom 10
65971.33333         -- reálny literál so základom 10
65_971.33_333 -- reálny literál so základom 10
8#43.6#e+4  -- reálny literál so základom 8
43.6E-4         -- reálny literál so základom 10
```

Znakové literály sú jednoduché ASCII znaky uzatvorené v apostrofoch.

```
'A'
'#'
```

Reťazcové literály sú sekvencie ASCII znakov uzatvorené v úvodzovkách.

```
"ABC"
"!@#$%*()_+|"
"double "" quote"
```

Literál bitových reťazcov

```
B"11111111"
B"1111_1111"
X"fF"
X"c3"
O"70"
```

Fyzikálny literál

```
15 ft
10 kohm
2.3 sec
```