10. Principy VHDL

Jazyk VHDL byl navržen a optimalizován pro popis struktury a chování elektronických systémů. Je převážně používán jako jazyk pro popis digitálních obvodů. Odlišuje se od tradičních jazyků:

- umožňuje modelovat souběžné děje - systém popsán pomocí paralelně pracujících komponent
- specifické datové objekty a typy - signály, sběrnice, vícehodnotová logika, fyzikální datové typy
- koncept času - schopnost modelovat elektrické vlastnosti (zpoždění, parametry hradel)

VHDL sa používá pro simulaci (validace, verifikace), syntézu (popis hardware - FPGA a ASIC) a specifikaci (obecný programovací jazyk). Pouze podmnožina VHDL může být syntetizována, to je zpravidla dáno konkrétním syntézním nástrojem. VHDL je navrženo tak, aby simulace byla efektivní.

Modelovaný systém je chápan jako sada vzájemně komunikujících komponent. Kompletní systém je modelován pomocí hierarchie komponent. Ve VHDL je každá komponenta modelována pomocí dvojice **entity** a **architecture**. Jednotlivé části popisu jsou separovatelné, jedna entita může mít několik architektur.

Entita

Popisuje rozhraní mezi komponentou a okolím. Rozhraní komponenty se skládá ze signálů rozhraní a seznamu generických parametrů, sloužících k předávání dalších parametrů do entity. Signály rozhraní mohou být podle směru šíření dat v módu:

- IN
- OUT (hodnota nemůže být uvnitř komponenty čtena
- INOUT (obousměrný port, sběrnice)
- BUFFER (podobně jako OUT, hodnotu však lze uvnitř komponenty číst)

Architektura

Popisuje chování a/nebo strukturu komponenty a je vždy svázána s entitou. Každá komponenta může být popsána na úrovni struktury (strukturní popis), chování (behaviorální popis) nebo datových toků (dataflow popis). Způsoby popisu je možné libovolně kombinovat.
Deklarační část je vyhrazena pro pro deklaraci signálů, konstant, typů nebo funkcí použitých uvnitř architektury.
Sekce paralelních příkazů může obsahovat instance komponent a procesy propojené pomocí signálů. Nezáleží na pořadí příkazů!

Strukturní (nebo také strukturální) popis znamená, že si nejdříve nakreslíme schéma zapojení obvodu a posléze toto schéma zapíšeme pomocí konstrukcí daného HDL. Tedy popíšeme jednotlivé komponenty obvodu a jejich propojení pomocí vodičů. Komponenty mohou být rovněž dekomponovány na subkomponenty a samostatně popsány. Lze takto definovat hierarchii komponent. Tento styl popisu má tu výhodu, že je velice blízký finální obvodové realizaci, návrhář má do jisté míry pod kontrolou proces syntézy a časování.

Behaviorální popis (nebo také popis chování) využívá faktu, že chování obvodu můžeme popsat algoritmem. Architektura se může skládat z více procesů, které mezi sebou komunikukí prostřednictvím signálů. Používáme konstrukce běžné v programovacích jazycích (cykly, procedury, funkce apod.). Při tomto popisu neuvažujeme obvodové detaily, tvorbu zapojení obvodu necháváme na procesu syntézy, který však nemusí být nutně průchodný. Pro některé programové konstrukce totiž neexistuje obvodový ekvivalent (např. pro rekurzi). Tento způsob s výhodou použijeme, pokud nám stačí obvody pouze simulovat a finální implementace není důležitá.

Data-flow popis je asi nejjednodušším způsobem, jak popisovat hardware, protože přímo vychází z logických rovnic. Modeluje datové závislosti. Zkráceně zapisuje chování pomocí paralelních příkazů uvnitř architektury. Jeho možnosti jsou však omezené.

Datové objekty

VHDL obsahuje konstrukce pro deklaraci datových objektů:

- constant - hodnota určitého typu, jenž je během simulace (syntézy) neměnná
- variable - typicky se používá k uchování mezihodnoty, deklaraci je možné provést pouze uvnitř procesu, funkce nebo procedury, hodnota **je** aktualizována ihned po vykonání příkazu přiřazení :=
- signal - signál realizuje spojení mezi jednotlivými komponentami systému, jedná se o jediný komunikační prostředek mezi jednotlivými paralelně běžícími procesy, hodnota **není** aktualizována ihned jakmile je vykonán příkaz přiřazení <=

Datové typy

VHDL je silně typovaný jazyk. Datové typy lze rozdělit na:

- skalární (boolean, integer, character, string, bit, …)
- kompozitní datový typ array a record (bit_vector, string, …)
- datový typ file
- datový typ acces (pointer)

Běžně používané typy jsou: **std_logic, std_logic_vector(X to/downto Y), integer**

Process

Proces může popisovat chování celé komponenty nebo pouze její části. Příkazy jsou vyhodnoceny **sekvenčně** a proto je důležité jejich pořadí.

```
[label:] process [(sensitivity list)]
      deklarační část
begin
      část sekvenčních příkazů
end process [label];
```

- Sensitivity list – seznam signálů; kdykoliv se změní signál ze sensitivity listu, je proces spuštěn. Sensitivity list slouží pouze pro simulaci (zrychlení simulace), syntézní nástroje jej ignorují (provádějí analýzu kódu). Častá chyba – v simulaci obvod funguje ve skutečnosti nikoliv.
- Sekvenční příkazy – sekvence příkazů popisující chování dané komponenty nebo její části. Přiřazení (<=) je vykonáno až po ukončení procesu (přirozené nebo voláním wait)

Proměnné versus signály

```
architecture VAR of EXAMPLE is
  signal CLK, RESULT: integer :=0;
begin

 process(CLK)
    variable variable1: integer :=1;
    variable variable2: integer :=2;
    variable variable3: integer :=3;
 begin
  if CLK'event and CLK='1' then
    variable1 := variable2;
    variable2 := variable1 + variable3;
    variable3 := variable2;
    RESULT <= variable1 + variable2 +
              variable3;
  end if;
 end process;

end architecture;
```

```
architecture SIG of EXAMPLE is
  signal CLK, RESULT: integer := 0;
  signal signal1: integer :=1;
  signal signal2: integer :=2;
  signal signal3: integer :=3;
begin

  process(CLK)
  begin
   if CLK'event and CLK='1' then
      signal1 <= signal2;
      signal2 <= signal1 + signal3;
      signal3 <= signal2;
      RESULT  <= signal1 + signal2 +
                 signal3;
   end if;
  end process;

end architecure;
```

V prvním případě budou výsledky 12, 25, …; v druhém 6, 8, …

Kombinační obvod

Kombinační logické obvody jsou takové logické obvody, ve kterých stavy na výstupech závisí pouze na okamžitých kombinacích vstupních proměnných a nezávisí na jejich předchozích hodnotách, s výjimkou krátkého přechodového děje. Kombinační logické obvody nemají žádnou paměť předchozích stavů, takže jedné kombinaci vstupních proměnných odpovídá právě jediná výstupní kombinace funkčních hodnot.

Sekvenční obvod

Sekvenční obvod se skládá ze dvou částí – kombinační a paměťové. Abychom mohli určit hodnotu výstupní proměnné, je potřeba u sekvenčních obvodů sledovat kromě vstupních proměnných ještě i jeho vnitřní proměnné – vnitřní stav. Jsou to proměnné, které jsou uchovány v paměťových členech. Existence vnitřních proměnných způsobuje, že stejné hodnoty vstupních proměnných přivedené na vstup obvodu, nevyvolávají vždy stejnou odezvu na výstupu obvodu.