KATEDRA INŻYNIERII KOMPUTEROWEJ

ModelSim® - VHDL

1. **Zakres tematyczny ćwiczenia:**
   - parametryzacja kodu z generacją opisu strukturalnego,
   - budowa testbench'a VHDL z automatyczną weryfikacją wyników symulacji.

2. **Parametryzacja działania i parametryzacja struktury**

   Budowa modeli parametrycznych w języku VHDL możliwa jest dzięki mechanizmowi osadzania wartości generycznych w interfejsie projektowanego urządzenia. Składnia polecenia `generic`, przedstawiona poniżej, zawiera identyfikator nadawany przez użytkownika, deklarację typu i opcjonalne przypisanie wartości domyślnej.

```
entity_declaration ⇐
        entity identifier is
            [ generic ( generic_interface_list ) ; ]
            [ port ( port_interface_list ) ; ]
        end [ entity ] [ identifier ] ;

     generic_interface_list ⇐
    ( identifier { , … } : subtype_indication [ := expression ] )
    { ; … }
```

   Przy osadzaniu w projekcie wcześniej zdefiniowanych komponentów, można dokonać mapowania wartości generycznych na takich samych zasadach, jak przy mapowaniu portów. Jeśli przy osadzeniu komponentu nie wystąpi operacja mapowanie parametru, przyjmowana jest wartość domyślna (Listing 2).

   Listing 1. Deklaracja modelu z parametryzacją działania

```
entity and2 is
      generic ( Tpd : time := 1 ns );
      port ( a, b : in bit; y : out bit );
end entity and2;

architecture simple of and2 is
begin
      and2_function:    y <= a and b after Tpd;
end architecture simple;
```

   Listing 2. Osadzenie modelu z parametryzacją działania

```
gate1 : entity work.and2(simple)
generic map ( Tpd => 2 ns )
port map ( a => sig1, b => sig2, y => sig_out );

gate2 : entity work.and2(simple)
-- Tpd = 1 ns (wartość domyślna)
port map ( a => a1, b => b1, y => sig1 );
```

Parametryzację struktury urządzenia można osiągnąć wykorzystując `generic` i instrukcje współbieżne generacji (`for generate`, `if generate`).

```
gen-label: for loop-index in loop-range generate
      concurrent statements ;
end generate ;

gen-label: if boolean_exp generate
      concurrent statements ;
end generate ;
```

Listing 3. Deklaracja modelu z parametryzacją struktury (sumator N-bitowy)

```
entity adderN is
      generic (N: integer:=4);
      port( ai:   in std_logic_vector(N-1 downto 0);
            bi:   in std_logic_vector(N-1 downto 0);
            ci:   in std_logic;
            so:   out std_logic_vector(N-1 downto 0);
            co:   out std_logic);
      end adderN;

architecture generacja of adderN is
      signal carry: std_logic_vector(N downto 0):=(others=>'0');
      begin
            carry(0) <= ci;
            co <= carry(N);
            s_gen: for i in (N-1) downto 0 generate
                  sumator:entity work.fullA(mix)
                        port map( ai => ai(i),
                                  bi => bi(i),
                                  ci => carry(i),
                                  so => so(i),
                                  co => carry(i+1));
            end generate;
 end generacja;
```

3. Przygotowanie projektu

[!] Polecenie: utwórz nowy projekt o nazwie <nr_indeksu_5> w katalogu CADHDL na dysku wskazanym przez prowadzącego (jeśli katalog nie istnieje utwórz go); pozostaw domyślną nazwę dla biblioteki roboczej;

[!] Polecenie: pobierz z serwera kursu plik ‘sum3.vhd’ (model badany) i zapisz w dowolnym katalogu tymczasowym; dodaj powyższy plik do projektu z opcją ‘Copy to project directory’ i domyślnym typem źródła;

4. Symulacja z automatyczną weryfikacją wyników przetwarzania

[!Z] Polecenie: Stwórz testbench dla sumatora N-bitowego.
Wymagania dla środowiska symulacji:

- parametryzacja kodu (możliwość symulacji dla różnych N);
- symulacja wszystkich możliwych kombinacji sygnałów wejściowych (ai, bi, ci).

Kod testbench’a przedstaw prowadzącemu.

[!Z] Polecenie: napisz makro(a) do kompilacji i symulacji układu sumatora 1-bitowego i 64-bitowego;

[!] Polecenie: dokonaj symulacji urządzenia dla obu powyższych przypadków;
- wyniki przedstaw prowadzącemu;
- zwróć uwagę na rozmiar plików .wlf dla obu przypadków.

[!Z] Zadanie:
Napisz parametryzowany testbench dla sumatora N-bitowego **z autoweryfikacją (w procesie lub procedurze)**, z podaniem informacji o czasach delta oraz zapisem wyników symulacji do pliku tekstowego (rozwiązanie jest rozwinięciem modelu utworzonego w punkcie 4).

[1] ModelSim®SE Reference Manual, Software Version 6.4a, Mentor Graphics 2008.