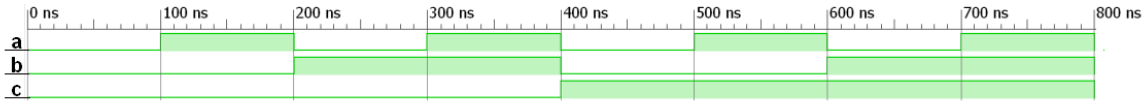
SYMULACJA W VHDL – UZUPEŁNIENIE DO INSTRUKCJI

Załóżmy iż utworzyliśmy schemat realizujący funkcję we=a*b+/c. Ponieważ jest to funkcja 3 zmiennych to należy sprawdzić wszystkie 8 kombinacji bitów (000, 001,..,111). Należy utworzyć zatem odpowiedni ciąg pobudzeń. Ponieważ pobudzenia są umiejscowione na osi czasu zatem należy należałoby wygenerować następujący przebieg czasowy.



Rys.1. Pobudzenia w skali czasu

Oczywiście czas trwania pojedynczej kombinacji sygnałów - pobudzenia może być inny niż 100ns jak przedstawiono na rysunku.

Wygenerowanie takiego przebiegu można zrealizować na wiele sposobów. W wyniku działania Project Nawigatora zostanie wygenerowany szablon pliku symulacji (Test Bench). W nim, w części zaznaczonej na czerwono należy wpisać przebieg symulacji.

```
-- Vhdl test bench created from schematic
--  D:\JAREK\DYDAKTYK\UkladyCyfrowe\testy\funkcja3zm\fun.sch - Thu Mar 03 17:56:53 2011
-- 1) This testbench template has been automatically generated using types
-- std_logic and std_logic_vector for the ports of the unit under test.
-- Xilinx recommends that these types always be used for the top-level
-- I/O of a design in order to guarantee that the testbench will bind
-- correctly to the timing (post-route) simulation model.
-- 2) To use this template as your testbench, change the filename to any
-- name of your choice with the extension .vhd, and use the "Source->Add"
-- menu in Project Navigator to import the testbench. Then
-- edit the user defined section below, adding code to generate the
-- stimulus for your design.
--
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;
LIBRARY UNISIM;
USE UNISIM.Vcomponents.ALL;
ENTITY fun_fun_sch_tb IS
END fun_fun_sch_tb;
ARCHITECTURE behavioral OF fun_fun_sch_tb IS

   COMPONENT fun
   PORT( a :       IN      STD_LOGIC;
         b       :       IN      STD_LOGIC;
         c       :       IN      STD_LOGIC;
         wy      :       OUT     STD_LOGIC);
   END COMPONENT;

   SIGNAL a        :       STD_LOGIC;
   SIGNAL b        :       STD_LOGIC;
   SIGNAL c        :       STD_LOGIC;
   SIGNAL wy       :       STD_LOGIC;

BEGIN

   UUT: fun PORT MAP(
          a => a,
          b => b,
          c => c,
          wy => wy
   );

-- *** Test Bench - User Defined Section ***
   tb : PROCESS
   BEGIN
      --WAIT; -- will wait forever
   END PROCESS;
-- *** End Test Bench - User Defined Section ***
END;
```

Można zauważyć, iż w utworzonym pliku pojawiają się wszystkie wejścia i wyjścia wprowadzone w schemacie. Podstawową konstrukcja do wykonania symulacji jest proces, którego budowa jest następująca:

```
nazwa_procesu : PROCESS
BEGIN
  - - tutaj zawartość procesu
END PROCESS
```

Proces zaczyna działanie w punkcie 0 na osi czasu. Wewnątrz procesu umieszczamy instrukcje sterujące przebiegiem czasowym procesu. Stosuje się dwa typy instrukcji:

- wprowadzenie pobudzenia dla wejścia,
- zatrzymanie działania procesu na określony czas.

Podstawową instrukcją jest wprowadzenia wartości pobudzenia. Wygląda ona następująca:

```
nazwa_wejscia <= wartość_logiczna ;
```

wartości logiczne 0 lub 1 podajemy w pojedynczym cudzysłowiu. Np.

```
a <= '1';
b <= '0';
```

Budowa instrukcji zatrzymania procesu na określony czas wygląda następująco:

```
WAIT FOR XXX jednostka;
```

gdzie:

jednostka – ps, ns, us, ms,s (jednostki czasu)

XXX – wartość

Dla ułatwienia założymy, iż proces wykonuje wszystkie instrukcje po kolei aż do końca, poczym zaczyna wszystko od początku. Można założyć, iż instrukcje wprowadzania pobudzenia mają zerowy czas trwania, co oznacza iż wykonanie następujących instrukcji jest jednoczesne mimo ich różnej kolejności.

a<='0';
b<='1';

Instrukcja WAIT powoduje zatrzymanie procesu na XXX jednostek czasu, co oznacza, iż wprowadzone wcześniej pobudzenia będą na symulatorze widoczne przez XXX jednostek czasu. Po czym proces przechodzi do wykonywania następnych instrukcji.

Poniżej przedstawiono podanie kolejnych symulacji dla naszego przykładu. (dwie kreski oznaczają iż wszytko poza nimi do końca linii jest komentarzem)

Sposób 1:

```
-- *** Test Bench - User Defined Section ***
   tb : PROCESS
   BEGIN
      a <= '0';  b <= '0';   c <= '0';
      WAIT FOR 100 ns;

      a <= '1';  b <= '0';   c <= '0';
      WAIT FOR 100 ns;

      a <= '0';  b <= '1';   c <= '0';
```

```
         WAIT FOR 100 ns;

         a <= '1';  b <= '1';   c <= '0';
         WAIT FOR 100 ns;

         a <= '0';  b <= '0';   c <= '1';
         WAIT FOR 100 ns;

         a <= '1';  b <= '0';   c <= '1';
         WAIT FOR 100 ns;

         a <= '0';  b <= '1';   c <= '1';
         WAIT FOR 100 ns;

         a <= '1';  b <= '1';   c <= '1';
         WAIT FOR 100 ns;
      END PROCESS;
   -- *** End Test Bench - User Defined Section ***
```

Przedstawiony sposób jest nadmiarowy, ponieważ jeśli to samo pobudzenie nie zmienia się po instrukcji WAIT, to nie trzeba je powtarzać.

```
-- *** Test Bench - User Defined Section ***
   tb : PROCESS
   BEGIN
      a <= '0';  b <= '0';   c <= '0';
      WAIT FOR 100 ns;

      a <= '1';
      WAIT FOR 100 ns;

      a <= '0';  b <= '1';
      WAIT FOR 100 ns;

      a <= '1';
      WAIT FOR 100 ns;

      a <= '0';  b <= '0';   c <= '1';
      WAIT FOR 100 ns;

       a <= '1';
      WAIT FOR 100 ns;

      a <= '0';  b <= '1';
      WAIT FOR 100 ns;

      a <= '1';
      WAIT FOR 100 ns;
   END PROCESS;
-- *** End Test Bench - User Defined Section ***
```

Sposób 2:

Jeśli przypatrzymy się rysunkowi pobudzeń, wówczas możemy zauważyć iż sygnał wejściowy a jest fala prostokątna o okresie 200ns, b o okresie 400ns, a c o okresie 800ns.

Istniej możliwość utworzenia w symulacji pray procesów i one wszystkie startują w punkcie zero na osi czasu. Wykorzystamy tą możliwość i napiszemy symulację z wykorzystaniem 3 procesów – każdy osobny dla każdej zmiennej. Każdy proces będzie tworzyła fale prostokątna o zadanym okresie.

```
-- *** Test Bench - User Defined Section ***
tb_a : PROCESS
   BEGIN
      a <= '0';
      WAIT FOR 100 ns;
      a <= '1';
      WAIT FOR 100 ns;
   END PROCESS;

tb_b : PROCESS
   BEGIN
      b <= '0';
      WAIT FOR 200 ns;
      b <= '1';
      WAIT FOR 200 ns;
   END PROCESS;

tb_c : PROCESS
   BEGIN
      c <= '0';
      WAIT FOR 400 ns;
      c <= '1';
      WAIT FOR 400 ns;
   END PROCESS;

-- *** End Test Bench - User Defined Section ***
```