Podstawy Układów Programowalnych FPGA
Symulacja i testbench. Dzielnik częstotliwości.

Dariusz Tefelski
dariusz.tefelski@pw.edu.pl

Politechnika Warszawska,
Wydział Fizyki

Przypomnienie struktury programu w VHDL

- Na początku programu należy dołączyć bibliotekę:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
```

- Następnie określa się pojedynczą jednostkę (*entity*), w którym definiowane są wyprowadzenia układu FPGA. W poniższym przykładzie mamy 2 wejścia *a* i *b* oraz jedno wyjście *y*, a jednostka została nazwana *simple_logic*:

```
entity simple_logic is
Port ( a : in  STD_LOGIC;
       b : in  STD_LOGIC;
       y : out  STD_LOGIC);
end simple_logic;
```

- Tryby portów:
  - **in** - sygnał wejściowy,
  - **out** - sygnał wyjściowy,
  - **inout** - sygnał wejściowy i wyjściowy,
  - **buffer** - port wyjściowy z możliwością odczytu.

Typy danych

- Każde wyjście musi mieć określony typ.
- Dostępne typy danych:
  - BIT, BIT_VECTOR
  - **STD_LOGIC**, STD_LOGIC_VECTOR,
  - STD_ULOGIC, STD_ULOGIC_VECTOR,
  - INTEGER, BOOLEAN,
  - TYP WYLICZENIOWY
- Typ STD_LOGIC czyli **standard logic** to ogólny typ zalecany dla wejść/wyjść układów FPGA. Przyjmuje następujące wartości:
  - U - niezainicjowany,
  - X - stan nieokreślony,
  - 0 oraz 1 - logiczne 0 bądź logiczna 1,
  - Z - stan wysokiej impedancji
  - W - sygnał słaby, nie wiadomo czy 0 czy 1,
  - L - słaby sygnał, prawdopodobnie 0,
  - H - słaby sygnał, prawdopodobnie 1,
  - '-' - stan dowolny.

Typy danych

- W niektórych przypadkach konieczne jest stworzenie wejścia/wyjścia kilku bitowego np:

```
entity simple_logic is
    Port (
        -- zestaw 8 diod reprezuntujacych jeden bajt
        LED :  out STD_LOGIC_VECTOR(7 DOWNTO 0)
    );
end simple_logic;
```

- STD_LOGIC_VECTOR to tablica obiektów STD_LOGIC.

Obiekty w VHDL

- Wyróżniamy trzy podstawowe klasy obiektów:
  1. *signal* - sygnał,
  2. *variable* - zmienne, można ich używać tylko w procesach,
  3. *constant* - stałe.
- Stałe tworzymy w następujący sposób:

```
constant nazwa_stalej : typ := wartosc;
```

np:

```
constant czas : integer := 100;
```

- Sygnały posiadają typ i wartość, która może się zmieniać w określony sposób i w określonym czasie.

```
signal nazwa_sygnalu : typ := wartosc;
```

np:

```
signal enable : std_logic := '0';
```

- Wartości zmiennych są przypisywane natychmiast a sygnałów zgodnie z regułami np. z pewnym opóźnieniem.

Przypomnienie struktury programu w VHDL

- W kolejnym kroku określa się architekturę:

```
architecture nazwa_architekturyl of nazwa_jednostki is
begin
-- instrukcje przypisania wartości do sygnałów,
-- procesy,
-- komponenty
end Behavioral;
```

np:

```
architecture Behavioral of simple_logic is
begin
    y <= a and b; -- funkcja logiczna AND
end Behavioral;
```

- Sygnały definiuje się pomiędzy **architecture** a **begin**.

Sekwencyjny opis układu w VHDL

- Do tej pory tworzyliśmy programy, które przypisywały wartości do danych wyjść. Takie instrukcje są instrukcjami równoległymi.
- W języku VHDL możemy tworzyć programy również sekwencyjnie ale tylko w procesach bądź funkcjach, wykorzystując np. strukturę **if**:

```
architecture Sequential of priority is
begin
    process (a) is
    begin
        if a(3)='1' then --sprawdzenie 3 bitu sygnalu a
            y<='1'
        else
            y<='0'
       end if;
   end process;
end architecture Sequential;
```

- Instrukcja procesu (**process**) ma listę czułości z sygnałem **a**. Proces jest tylko wtedy uruchamiany gdy jakikolwiek sygnał wymieniony w liście czułości zmieni swoją wartość.

Testbench - jednostki testowe

- Gdy układ staje się coraz bardziej skomplikowany to możemy sprawdzić poprawność jego działania wykorzystując symulacje.
- Aby zrozumieć zasadę symulacji, sprawdźmy program z poprzednich zajęć:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity simple_logic is
    Port (
          a : in std_logic_vector(1 downto 0);
          y : out  STD_LOGIC);
end simple_logic;

architecture Behavioral of simple_logic is
begin
    y <= a(1) and a(0);
end Behavioral;
```

Testbench - jednostki testowe

- Aby przetestować poprawność działania programu, musimy sprawdzić jak zachowuje się wyjście **y** w zależności o wszystkich możliwych stanów na wejściach **a(1), a(0)** zgodnie z tabelę prawdy:

a(1)	**a(0**	**y**
0	0	0
0	1	0
1	0	0
1	1	1

Testbench - jednostki testowe

- Proces generowania symulacji jest opisany w instrukcji do laboratorium. Wygenerowany kod wygląda następująco:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY test_bench IS
END test_bench;

ARCHITECTURE behavior OF test_bench IS
-- Component Declaration for the Unit Under Test (UUT)
COMPONENT podstawowe_bramki
    PORT(
        a : in std_logic_vector(1 downto 0);
        y : out std_logic;
);
END COMPONENT;
```

- Blok ENTITY będzie pusty a to co było oryginalnie zadeklarowane wewnątrz ENTITY $\rightarrow$ PORT będzie teraz umieszczone w komponencie.

Kontynuacja poprzedniego programu

- Następnie następuje deklaracja sygnałów wejściowych i wyjściowych:

```
--Inputs
signal a : std_logic_vector(1 downto 0) := (others => '0');
--Outputs
signal y : std_logic  :=  '0';
```

- A w kolejnym kroku następuje określenie okresu zegara (wartość domyślna to 10 ns czyli f=100 MHz):

```
constant period : time := 10 ns;
```

- Potem tworzy się mapa połączeń sygnałów jednostki testowej z testowanym komponentem:

```
BEGIN
-- Instantiate the Unit Under Test (UUT)
uut: simple_logic PORT MAP (
    a => a,
    y => y
);
```

Kontynuacja poprzedniego programu

- Ostatni element to kod symulacji zawarty w procesie: **stim_proc**

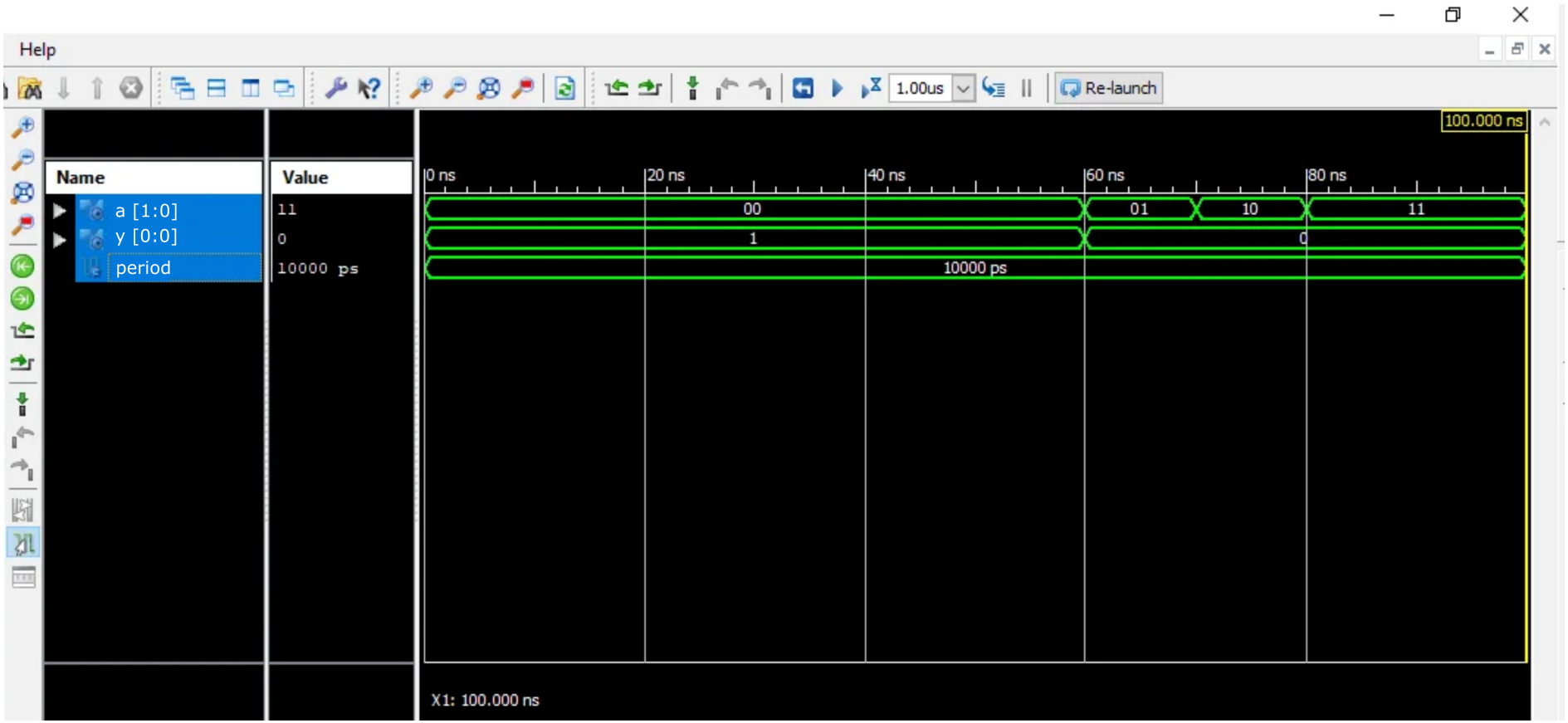
```
-- Stimulus process
stim_proc: process
begin
    wait for 50 ns;
    a(0) <= '0';
    a(1) <= '0';
    wait for period;
    a(0) <= '1';
    a(1) <= '0';
    wait for period;
    a(0) <= '0';
    a(1) <= '1';
    wait for period;
    a(0) <= '1';
    a(1) <= '1';
wait;
end process;
END;
```

Przedrostki czasu

Przyrostek w VHDL	Nazwa	Mnożnik
fs	femtosekunda	10^{-15} s
ps	pikosekunda	10^{-12} s
ns	nanosekunda	10^{-9} s
us	mikrosekunda	10^{-6} s
ms	milisekunda	10^{-3} s
sec	sekunda	1 s
min	minuta	60 s
hr	godzina	3600 s

Źródło obrazka: https://forbot.pl/

Rezultat symulacji



Dziękuję za uwagę!

Prezentacja powstała w oparciu o materiały z:

- "Projektowanie układów cyfrowych z wykorzystaniem języka VHDL"
- http://www.cs.put.poznan.pl/
- `https://forbot.pl/`