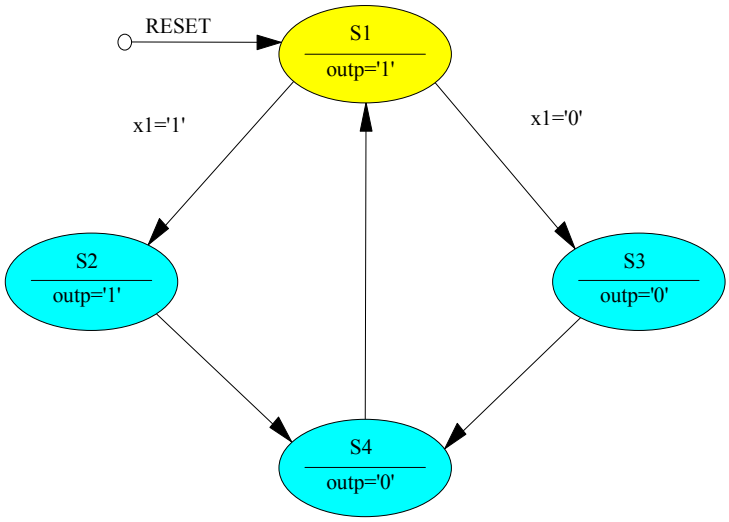
KATEDRA INŻYNIERII KOMPUTEROWEJ

ModelSim® - VHDL

1. **Zakres tematyczny ćwiczenia:**
   - symulacja modeli z wykorzystaniem testbench'a VHDL,
   - modelowanie automatów skończonych w języku VHDL.

2. **Sposoby modelowania automatów skończonych w języku VHDL**
   Większość narzędzi do syntezy zawiera specjalne szablony przeznaczone do opisu automatów skończonych (FSM). Podstawowym sposobem opisu automatu w języku VHDL jest użycie instrukcji `process`. Poniżej przedstawione zostały przykłady konstrukcji automatu z użyciem szablonu 1, 2 i 3-procesowego [2]. Zwróć uwagę na zastosowanie typu wyliczeniowego do definiowania stanów automatu!



Rys.1. Diagram stanu analizowanego automatu.

Listing 1. Opis 1-procesowy [2]

```
entity fsm_1 is
    port ( clk, reset, x1 : IN std_logic;
           outp           : OUT std_logic);
end entity;

architecture beh1 of fsm_1 is
    type state_type is (s1,s2,s3,s4);
    signal state: state_type ;
begin

    process (clk,reset)
    begin
        if (reset ='1') then
```

```
            state <=s1;
            outp<='1';
        elsif (clk='1' and clk'event) then
            case state is
                when s1 =>  if x1='1' then
                                state <= s2;
                                outp <= '1';
                            else
                                state <= s3;
                                outp <= '0';
                            end if;
                when s2 => state <= s4; outp <= '0';
                when s3 => state <= s4; outp <= '0';
                when s4 => state <= s1; outp <= '1';
            end case;
        end if;
    end process;
end beh1;
```

Listing 2. Opis 2-procesowy [2]

```
entity fsm_2 is
    port ( clk, reset, x1 : IN std_logic;
           outp           : OUT std_logic);
end entity;

architecture beh1 of fsm_2 is
    type state_type is (s1,s2,s3,s4);
    signal state: state_type ;
begin

    process1: process (clk,reset)
    begin
        if (reset ='1') then state <=s1;
        elsif (clk='1' and clk'Event) then
            case state is
                when s1 =>  if x1='1' then
                                state <= s2;
                            else
                                state <= s3;
                            end if;
                when s2 => state <= s4;
                when s3 => state <= s4;
                when s4 => state <= s1;
            end case;
        end if;
    end process process1;
    process2 : process (state)
    begin
        case state is
            when s1 => outp <= '1';
            when s2 => outp <= '1';
            when s3 => outp <= '0';
            when s4 => outp <= '0';
        end case;
    end process process2;
end beh1;
```

Listing 3. Opis 3-procesowy [2]

```
entity fsm_3 is
    port ( clk, reset, x1 : IN std_logic;
           outp           : OUT std_logic);
end entity;

architecture beh1 of fsm_3 is
    type state_type is (s1,s2,s3,s4);
    signal state, next_state: state_type ;
begin
```

```
    process1: process (clk,reset)
    begin
        if (reset ='1') then
            state <=s1;
        elsif (clk='1' and clk'Event) then
            state <= next_state;
        end if;
    end process process1;
    process2 : process (state, x1)
    begin
        case state is
            when s1 =>  if x1='1' then
                            next_state <= s2;
                        else
                            next_state <= s3;
                        end if;
            when s2 => next_state <= s4;
            when s3 => next_state <= s4;
            when s4 => next_state <= s1;
        end case;
    end process process2;
    process3 : process (state)
    begin
        case state is
            when s1 => outp <= '1';
            when s2 => outp <= '1';
            when s3 => outp <= '0';
            when s4 => outp <= '0';
        end case;
    end process process3;
end beh1;
```

3. Przygotowanie projektu

[!] Polecenie: utwórz nowy projekt o nazwie <nr_indeksu_3> 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 pliki 'automat_struct.vhd' (model automatu), 'automat_struct_tb.vhd' (testbench) oraz makro 'automat_struct_tb.do'. Zapisz pliki w dowolnym katalogu tymczasowym; dodaj powyższe pliki do projektu z opcją 'Copy to project directory' i domyślnym typem źródła;

Interfejs wzorcowego modelu `automat(struct)`:

```
clk: in std_logic;
rst: in std_logic;
display: out std_logic_vector(3 downto 0);
```

Działanie modelu `automat(struct)`:

generator numeru indeksu (5 cyfr kodowanych NKB),
po wyświetleniu numeru indeksu generowany znacznik końca sekwencji (F),
reset synchroniczny;

[!] Polecenie: uruchom symulację modelu wzorcowego korzystając z makra .do (testbench VHDL); zwróć uwagę na znaczenie opcji zastosowanych w poleceniu `vsim` makra symulacji;

[!] Polecenie: zapisz wyniki symulacji do pliku .wlf (polecenie `dataset`).

4. Modelowanie automatu

[!Z] Polecenie:
- stwórz **nową architekturę** dla istniejącej jednostki projektowej `automat`; działanie nowej architektury powinno odpowiadać działaniu układu symulowanego w punkcie poprzednim; **zastosuj opis 1-procesowy**
- przygotuj **makro** kompilacji/symulacji/porównania;
- dokonaj symulacji nowozdefiniowanej architektury korzystając z testbench'a VHDL; dokonaj porównania (funkcja `compare`) z zestawem danych z punktu 3; **wyniki porównania przedstaw prowadzącemu**;

[!Z] Polecenie:
- stwórz **nową architekturę** dla istniejącej jednostki projektowej `automat`; działanie nowej architektury powinno odpowiadać działaniu układu symulowanego w punkcie poprzednim; **zastosuj opis 2-procesowy**
- przygotuj **makro** kompilacji/symulacji/porównania;
- dokonaj symulacji nowozdefiniowanej architektury korzystając z testbench'a VHDL; dokonaj porównania (funkcja `compare`) z zestawem danych z punktu 3; **wyniki porównania przedstaw prowadzącemu**;

[!Z] Polecenie:
- stwórz **nową architekturę** dla istniejącej jednostki projektowej `automat`; działanie nowej architektury powinno odpowiadać działaniu układu symulowanego w punkcie poprzednim; **zastosuj opis 3-procesowy**
- przygotuj **makro** kompilacji/symulacji/porównania;
- dokonaj symulacji nowozdefiniowanej architektury korzystając z testbench'a VHDL; dokonaj porównania (funkcja `compare`) z zestawem danych z punktu 3; **wyniki porównania przedstaw prowadzącemu**;

[1] ModelSim®SE Reference Manual, Software Version 6.4a, Mentor Graphics 2008.
[2] Synthesis and Simulation Design Guide, ISE Version 9.2i, Xilinx Corporation 2007.