# Układy cyfrowe i systemy wbudowane

*Laboratorium 6 –moduły PS2_Kbd oraz RS232_Tx*

*sprawozdanie*

Prowadzący: dr inż. Jarosław Sugier

# 1.Treść zadań laboratoryjnych i ich realizacja.

## 1.1. Zadania laboratoryjne

W ramach zajęć laboratoryjnych mieliśmy dostosować stworzony na poprzednich zajęciach zamek szyfrowy otwierany 4-literowym kodem „KKJK” (inicjały członków grupy), tak aby współpracował z modułami PS2_Kbd oraz RS232_Tx. W ramach zadania dodatkowego, mieliśmy stworzyć 4-bitowy licznik otwarć zamka.

## 1.2.Obsługa PS2_Kbd

Przejście z obsługi klawiatury PS2_Rx na PS2_Kbd nie było skomplikowane – PS2_Kbd działa w sposób bardziej intuicyjny – zamiast odbierać 3 znaki, odbieramy jeden i czekamy na przejście sygnału F0 w stan 0, która sygnalizuje puszczenie guzika. Korzystanie z takiego układu pozwala na znaczące uproszczenie maszyny stanów w stosunku do wersji z poprzedniego ćwiczenia.

## 1.3.Maszyna stanów

Do opisania układu stworzyliśmy następujące stany: z0, z1, z2, z3, z4. Stan z0 to stan początkowy – użytkownik nie wprowadził jeszcze żadnego poprawnego znaku. Pozostałe stany oznaczają kolejne wczytane znaki.

W związku ze specyfiką działania modułu PS2_Kbd zmiany stanów zachodzą tylko wtedy, kiedy sygnał DO_Rdy = ‘1’ oraz F0 = ‘0’.

Kod odpowiadający za aktualizowanie stanów nie zmienił się w stosunku do poprzedniego ćwiczenia. Zaktualizowany został tylko kod odpowiadający za logikę przejść między stanami.

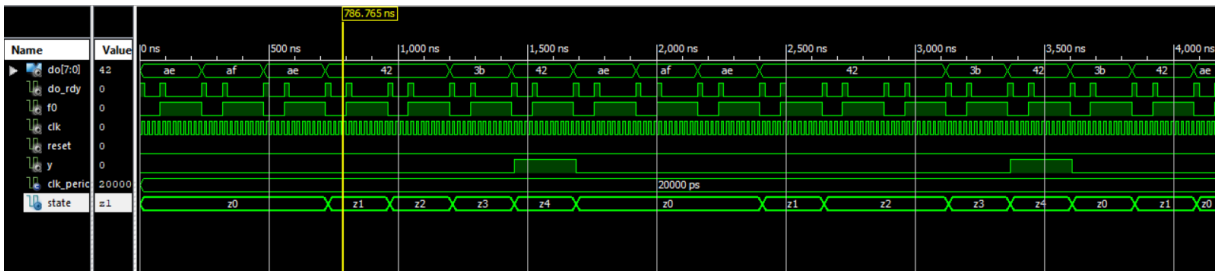
```
54
55      stateLogic : process(state, DO, F0, DO_Rdy)
56      begin
57         next_state <= state; -- by default
58         if DO_rdy = '1' and F0 = '0' then
59            case state is
60               when z0 =>
61                  if DO = K then
62                     next_state <= z1;
63                  end if;
64               when z1 =>
65                  if DO = K then
66                     next_state <= z2;
67                  else
68                     next_state <= z0;
69                  end if;
70               when z2 =>
71                  if DO = J then
72                     next_state <= z3;
73                  elsif DO = K then
74                     next_state <= z2;
75                  else
76                     next_state <= z0;
77                  end if;
78               when z3 =>
79                  if DO = K then
80                     next_state <= z4;
81                  else
82                     next_state <= z0;
83                  end if;
84               when z4 =>
85                  if DO = K then
86                     next_state <= z1;
87                  else
88                     next_state <= z0;
89                  end if;
90            end case;
91         end if;
92      end process stateLogic;
93
94   Y<= '1' when state = z4 else '0';
95
```

Testowanie odbywało się z wykorzystaniem sekwencji: ***KKJK***KKKJKJK*** gdzie * to dowolny symbol nie będący częścią sekwencji otwierającej zamek. Do wszystkich zadań wykonanych w ramach omawianych zajęć wykorzystany został ten sam testbench. Input2 to tablica zawierająca kolejne znaki sekwencji.

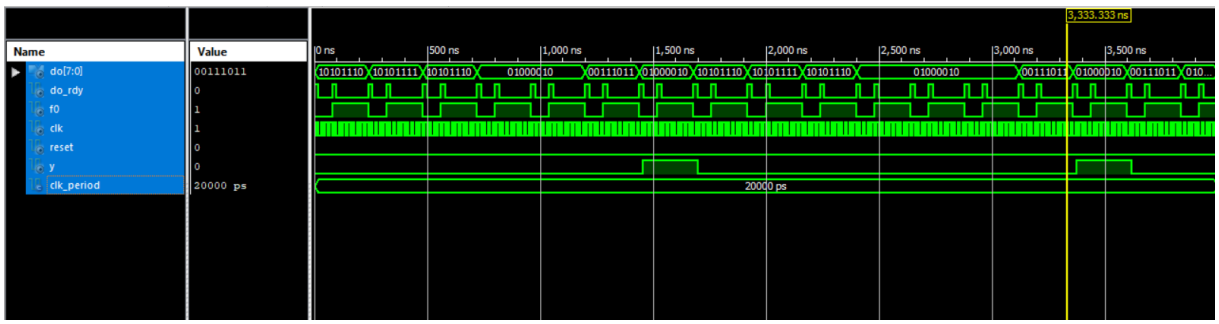
```
106
107     -- Stimulus process
108     stim_proc: process
109        begin
110
111        for i in 19 downto 0 loop
112           F0 <= '0';
113           DO <= input2(i);
114           DO_Rdy <= '1';
115           wait for CLK_period;
116
117           DO_rdy <= '0';
118           wait for 3*CLK_period;
119
120           F0 <= '1';
121           DO_rdy <= '1';
122           wait for CLK_period;
123
124           DO_rdy <= '0';
125           wait for 7*CLK_period;
126
127
128        end loop;
129        wait;
130        -- insert stimulus here
131
132        wait;
133     end process;
```

Dla zadania pierwszego wykonaliśmy symulację behawioralną (w celu zaprezentowania przejść między stanami) oraz symulację post-route. Zaprezentowane wyniki są analogiczne do tych z poprzedniego laboratorium: zamek otwiera się w oczekiwanych momentach.

Wyniki symulacji behawioralnej:




Wyniki symulacji post-route:



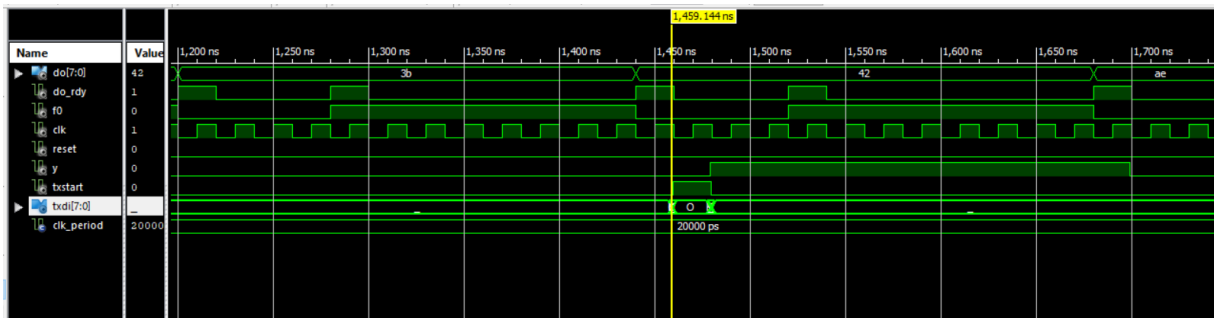
## 1.4.Obsługa RS232_Tx i zmiany w maszynie stanów.

Układ RS232_Tx to moduł, który obsługuje nadawanie bajtów do portu RS232. W normalnych warunkach, do tego portu podłączony zostałby wyświetlacz. W celu wysłania bajtu danych, należy go podać do układu RS232_Tx na wejście TxDI.

Podczas laboratorium za zadanie mieliśmy stworzyć dodatkowy, jednotaktowy stan, podczas którego na port RS232 wysłany zostałby bajt ‘O’. Bajt ‘O’ miał być wysyłany po wprowadzeniu ostatniej poprawnej litery do otwarcia zamku. Realizując zadanie rozbiliśmy wcześniej istniejący stan ‘4’ na stany ‘4A’ i ‘4B’. Stan ‘4A’ to stan jednotaktowy, a stan ‘4B’, to dawny stan ‘4’ – końcowy stan maszyny, po otwarciu zamka. Za ustawienie TxDi odpowiada proces setTxDi. W kodzie można też zauważyć elsif state = ‘z4A’, które nigdy nie zostanie wywołane. Ten fragment kodu nie jest jednak błędem, ponieważ w języku VHDL instrukcja case musi opisywać wszystkie opcje, musieliśmy więc dokonać sprawdzenia dla stanu ‘z4A’ lub skorzystać z instruckji ‘others’

```
81              when z3 =>
82                 if DO = K then
83                    next_state <= z4A;
84                 else
85                    next_state <= z0;
86                 end if;
87              when z4A =>
88                 next_state <= z4B;
89              when z4B =>
90                 if DO = K then
91                    next_state <= z1;
92                 else
93                    next_state <= z0;
94                 end if;
95           end case;
96        elsif state = z4A then
97           next_state <= z4B;
98        end if;
99     end process stateLogic;
100
101    setTxDi : process(state)
102    begin
103       if state = z4A then
104          TxDi <= "01001111";
105       else
106          TxDi <= "00000000";
107       end if;
108    end process setTxDi;
```

Fragment symulacji post-route prezentujący jednotaktowy sygnał:

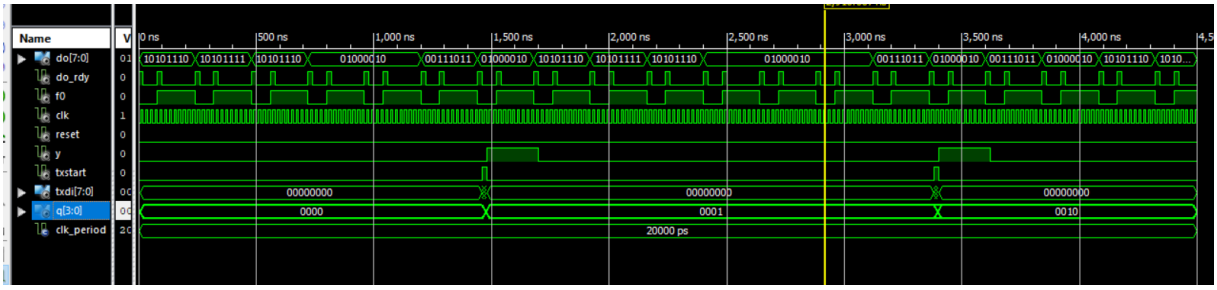



## 1.5.Licznik otwarć

Polecenie nie precyzowało rodzaju licznika, zdecydowaliśmy się na licznik modulo 8. Tmp to zmienna 4-bitowa typu unsigned. Schemat licznika bazowaliśmy na slajdach z wykładu. Podczas zajęć nie udało nam się dokończyć tego zadania. Po zajęciach znaleźliśmy błąd – zmienna odpowiadająca za licznik była niezainicjalizowana.

```
111     counter : process ( Clk, state)
112     begin
113        if rising_edge( Clk ) and state = z4A then
114           if tmp = "1011" then
115           tmp <= "0000";
116           else
117           tmp <= tmp + 1;
118           end if;
119        end if;
120     end process counter;
121
122  Y<= '1' when state = z4B else '0';
123  TxStart <= '1' when state = z4A else '0';
124  Q<= STD_LOGIC_VECTOR(tmp);
125
```

Wyniki symulacji post-route:




## 2.Podsumowanie

Układ PS2_Kbd jest dużo bardziej intuicyjny w obsłudze niż PS2_Rx. Korzystanie z niego pozwala na duże uproszczenie logiki programu.  Praca z układem RS232_Tx również okazała się nieskomplikowana.