Edge detector

Solution

```
library ieee;
use ieee.std_logic_1164.all;
entity edge_detector1 is
   port(
      clk, reset: in std_logic;
      strobe: in std_logic;
      p1: out std_logic
   );
end edge_detector1;

architecture moore_arch of edge_detector1 is
   type state_type is (zero, edge, one);
   signal state_reg, state_next: state_type;
begin
   -- state register
   process(clk,reset)
   begin
      if (reset='1') then
         state_reg <= zero;
```

Edge detector

Solution

```
     elsif (clk'event and clk='1') then
        state_reg <= state_next;
     end if;
  end process;
  -- next-state logic
  process(state_reg,strobe)
  begin
     case state_reg is
        when zero=>
           if strobe= '1' then
              state_next <= edge;
           else
              state_next <= zero;
           end if;
        when edge =>
           if strobe= '1' then
              state_next <= one;
           else
              state_next <= zero;
```

Edge detector

Solution

```
            end if;
         when one =>
            if strobe= '1' then
               state_next <= one;
            else
               state_next <= zero;
            end if;
      end case;
   end process;
   -- Moore output logic
   p1 <= '1' when state_reg=edge else
         '0';
end moore_arch;
```

Edge detector

Solution

- □ Machine d’états Moore sans la logique de sortie

```
architecture clever_assign_buf_arch of edge_detector1 is
   constant zero: std_logic_vector(1 downto 0):= "00";
   constant edge: std_logic_vector(1 downto 0):= "10";
   constant one: std_logic_vector(1 downto 0) := "01";
   signal state_reg,state_next: std_logic_vector(1 downto 0);
begin
   -- state register
   process(clk,reset)
   begin
      if (reset=’1’) then
         state_reg <= zero;
      elsif (clk’event and clk=’1’) then
         state_reg <= state_next;
      end if;
   end process;
   -- next-state logic
   process(state_reg,strobe)
```

Edge detector

Solution

```
begin
   case state_reg is
      when zero=>
         if strobe= '1' then
            state_next <= edge;
         else
            state_next <= zero;
         end if;
      when edge =>
         if strobe= '1' then
            state_next <= one;
         else
            state_next <= zero;
         end if;
      when others =>
         if strobe= '1' then
            state_next <= one;
         else
            state_next <= zero;
```

Edge detector

Solution

```
            end if;
      end case;
   end process;
   -- Moore output logic
   p1 <= state_reg(1);
end clever_assign_buf_arch;
```

Edge detector

Solution

- □ Machine d’états Moore avec la logique *look-ahead*

```
architecture look_ahead_arch of edge_detector1 is
   type state_type is (zero, edge, one);
   signal state_reg, state_next: state_type;
   signal p1_reg, p1_next: std_logic;
begin
   -- state register
   process(clk,reset)
   begin
      if (reset=’1’) then
         state_reg <= zero;
      elsif (clk’event and clk=’1’) then
         state_reg <= state_next;
      end if;
   end process;
   -- output buffer
   process(clk,reset)
   begin
```

Edge detector

Solution

```
      if (reset=’1’) then
         p1_reg <= ’0’;
      elsif (clk’event and clk=’1’) then
         p1_reg <= p1_next;
      end if;
   end process;
   -- next-state logic
   process(state_reg,strobe)
   begin
      case state_reg is
         when zero=>
            if strobe= ’1’ then
               state_next <= edge;
            else
               state_next <= zero;
            end if;
         when edge =>
            if strobe= ’1’ then
               state_next <= one;
```

Edge detector

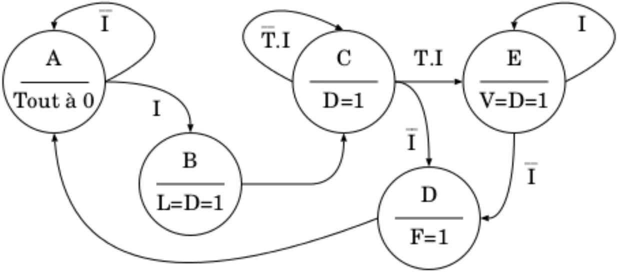
Solution

```
            else
               state_next <= zero;
            end if;
         when one =>
            if strobe= '1' then
               state_next <= one;
            else
               state_next <= zero;
            end if;
      end case;
   end process;
   -- look-ahead output logic
   p1_next <= '1' when state_next=edge else
              '0';
   -- output
   p1 <= p1_reg;
end look_ahead_arch;
```

Exemple d'utilisation sur la carte

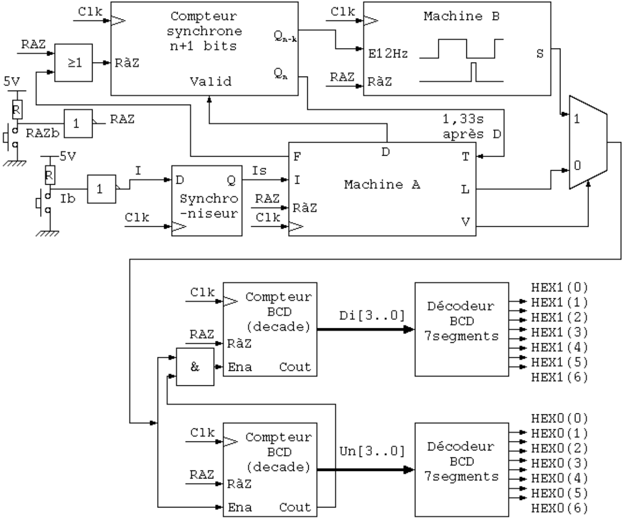
Bouton poussoir automatique

On souhaite réaliser un système de comptage par bouton poussoir qui devient automatique (à 24Hz : division par 2^{21} de l'horloge à 50MHz) si le bouton est maintenu plus de 1,34s (division par 2^{26} de l'horloge à 50MHz)



Exemple d'utilisation sur la carte

Bouton poussoir automatique


Clk
Compteur synchrone n+1 bits
Q_{n-k}
Q_n
RàZ
Valid
RAZ
≥1
5V
R
RAZb
1
RAZ
Clk
Machine B
E12Hz
S
RAZ
RàZ
1,33s après D
5V
R
Ib
1
I
D
Q
Synchro -niseur
Clk
Is
RAZ
Clk
F
I
RàZ
D
Machine A
T
L
V
1
0
Clk
Compteur BCD (decade)
RAZ
RàZ
&
Ena
Cout
Di[3..0]
Décodeur BCD 7segments
HEX1(0)
HEX1(1)
HEX1(2)
HEX1(3)
HEX1(4)
HEX1(5)
HEX1(6)
Clk
Compteur BCD (decade)
RAZ
RàZ
Ena
Cout
Un[3..0]
Décodeur BCD 7segments
HEX0(0)
HEX0(1)
HEX0(2)
HEX0(3)
HEX0(4)
HEX0(5)
HEX0(6)