Machine d'états

Exemples

- Coder en VHDL un détecteur de front montant en utilisant :
  - une machine de Moore
  - une machine de Mealy

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;
          end if;
```

Edge detector

Solution

```
            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;
         end if;
```

Edge detector

Solution

```
      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;
          else
```

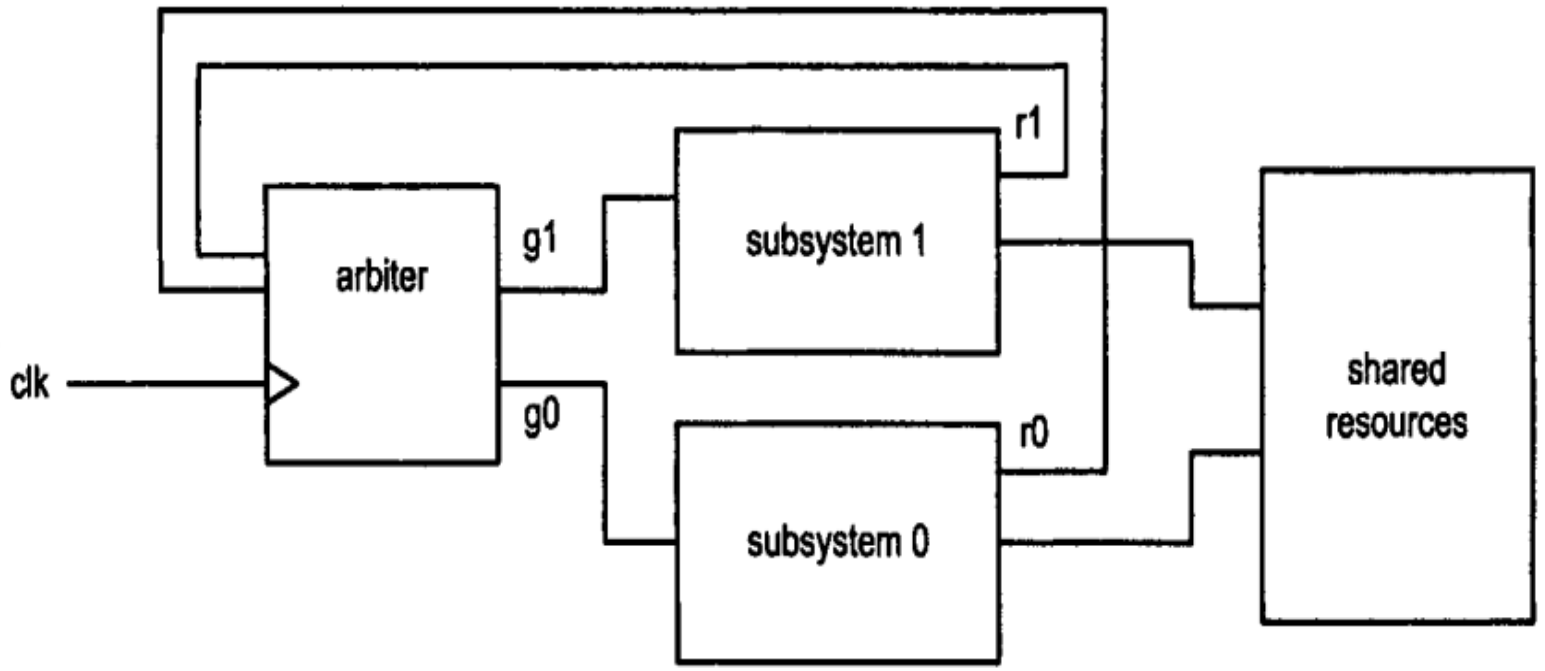
Edge detector

Solution

```
                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;
```

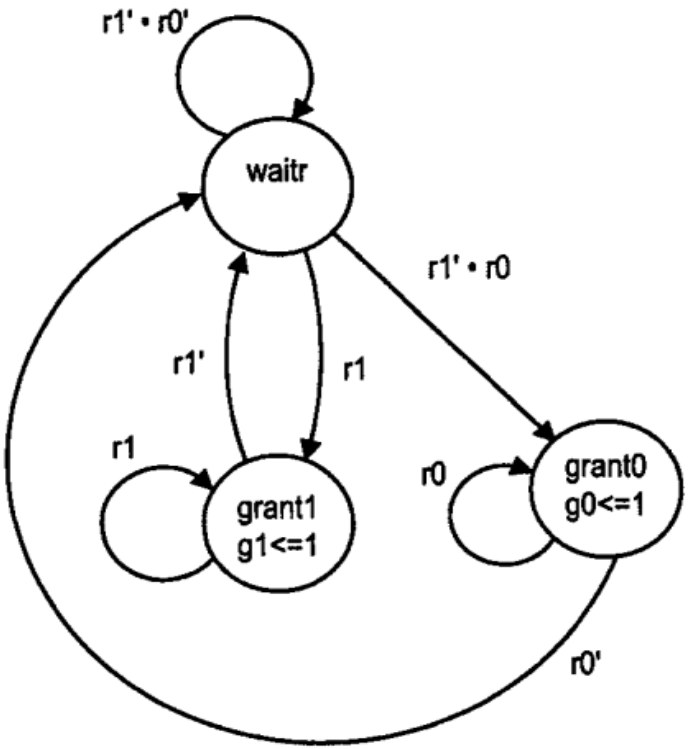
Arbitre

Introduction



Arbitre

Introduction



Arbitre

Solution

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
entity arbiter2 is
   port(
      clk: in std_logic;
      reset: in std_logic;
      r: in std_logic_vector(1 downto 0);
      g: out std_logic_vector(1 downto 0)
   );
end arbiter2;

architecture fixed_prio_arch of arbiter2 is
   type mc_state_type is (waitr, grant1, grant0);
   signal state_reg, state_next: mc_state_type;
begin
   -- state register
   process(clk,reset)
   begin
```

Arbitre

Solution

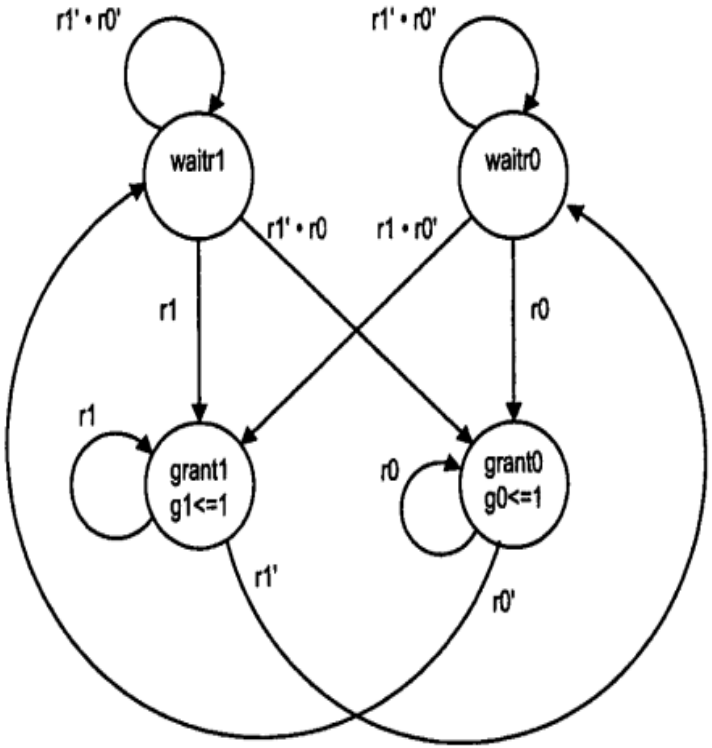
```
    if (reset='1') then
       state_reg <= waitr;
    elsif (clk'event and clk='1') then
       state_reg <= state_next;
    end if;
end process;
-- next-state and output logic
process(state_reg,r)
begin
    g <= "00";     -- default values
    case state_reg is
       when waitr =>
          if r(1)='1' then
             state_next <= grant1;
          elsif r(0)='1' then
             state_next <= grant0;
          else
             state_next <= waitr;
          end if;
       when grant1 =>
```

Arbitre

Solution

```
            if (r(1)='1') then
               state_next <= grant1;
            else
               state_next <= waitr;
            end if;
            g(1) <= '1';
         when grant0 =>
            if (r(0)='1') then
               state_next <= grant0;
            else
               state_next <= waitr;
            end if;
            g(0) <= '1';
      end case;
   end process;
end fixed_prio_arch;
```

Arbitre 1


r1' • r0'
r1' • r0'
waitr1
waitr0
r1' • r0
r1 • r0'
r1
r0
r1
r0
grant1
g1<=1
grant0
g0<=1
r1'
r0'

Arbitre 1

Solution

```
architecture rotated_prio_arch of arbiter2 is
   type mc_state_type is (waitr1, waitr0, grant1, grant0);
   signal state_reg, state_next: mc_state_type;
begin
   -- state register
   process(clk,reset)
   begin
      if (reset='1') then
         state_reg <= waitr1;
      elsif (clk'event and clk='1') then
         state_reg <= state_next;
      end if;
   end process;
   -- next-state and output logic
   process(state_reg,r)
   begin
      g <= "00";     -- default values
      case state_reg is
         when waitr1 =>
```

Arbitre 1

Solution

```
            if r(1)='1' then
               state_next <= grant1;
            elsif r(0)='1' then
               state_next <= grant0;
            else
               state_next <= waitr1;
            end if;
         when waitr0 =>
            if r(0)='1' then
               state_next <= grant0;
            elsif r(1)='1' then
               state_next <= grant1;
            else
               state_next <= waitr0;
            end if;
         when grant1 =>
            if (r(1)='1') then
               state_next <= grant1;
            else
               state_next <= waitr0;
```

Arbitre 1

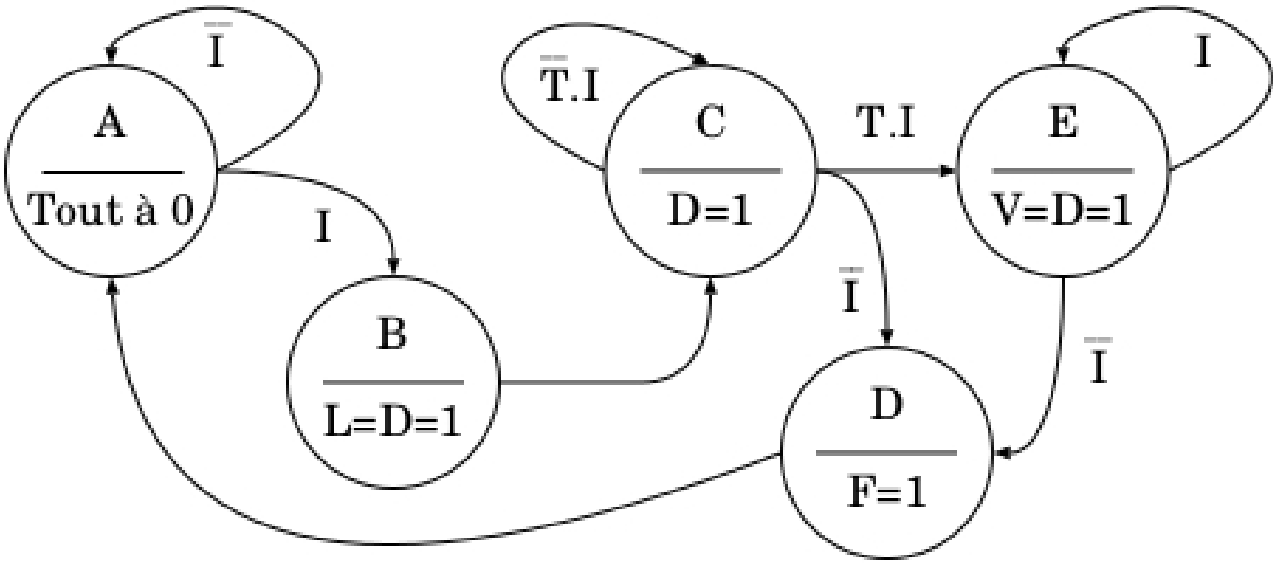
Solution

```
            end if;
            g(1) <= '1';
         when grant0 =>
            if (r(0)='1') then
               state_next <= grant0;
            else
               state_next <= waitr1;
            end if;
            g(0) <= '1';
      end case;
   end process;
end rotated_prio_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