

Solution seance 3

# Exo 1 Q1

ARCHITECTURE Archi1exo4 OF exo4 IS

BEGIN

process (adr)

begin

case adr is

when "00" => s<= a;

when "01" => s<= b;


when "10" => s<= c;

when others => s<= d;

end case;

end process;

END Archi1exo4;



X7seg

Process (sw)

Begin

Case sevenseg is

when x"0" => sevenseg <= "1000000";

when x"1" => sevenseg <= "1111001";

.....

Q3a

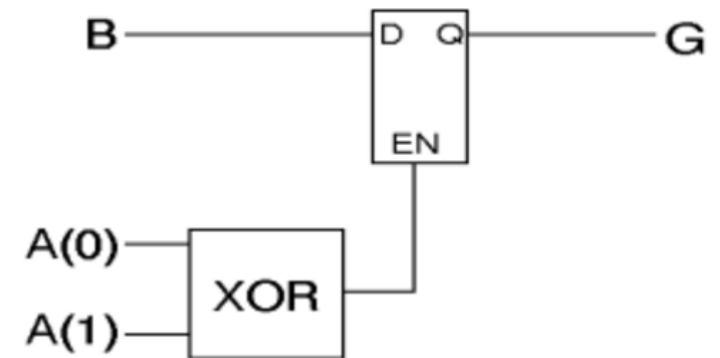
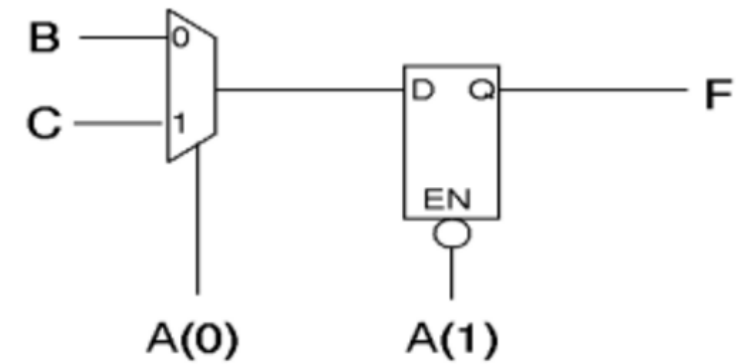
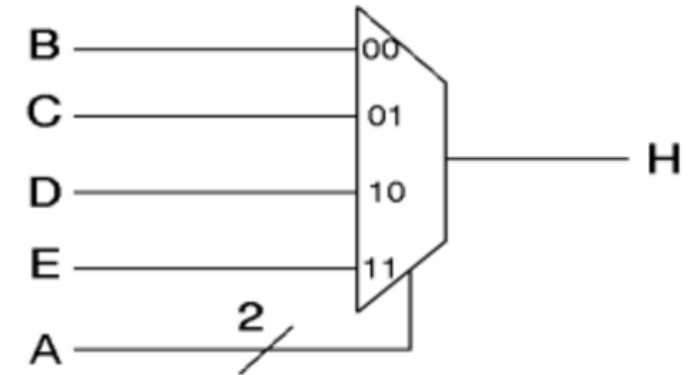
Seul H est complet


F ne change que si  $a(1) = 0$

G ne change que si  $a = 01$  ou  $10$

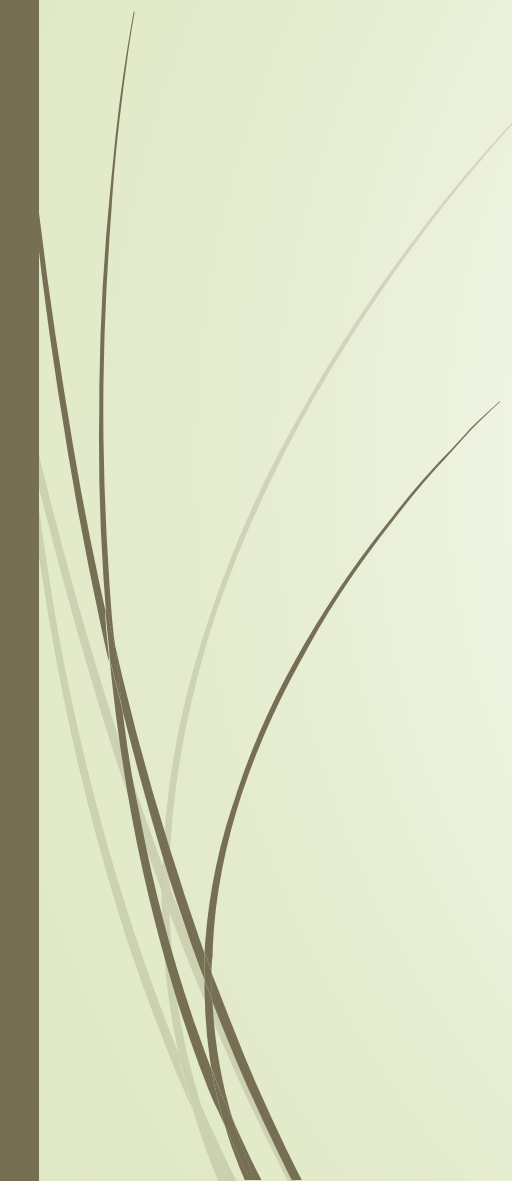
= Xor

Le circuit représenté par ce programme contient les éléments suivants :





Q3b



- Le signal H est issu d'un multiplexeur à 4 entrées.
- Le signal F est mis à jour seulement lorsque A(1) vaut '0', et, prend alors une valeur dépendant de A(0).
- Quant au signal G, il est seulement mis à jour lorsque A(0) et A(1) sont différents.

## Code avec 3 process + 2 latch

architecture synthesizable of test is

signal Fmux,ADiff : std\_logic;

begin

ADiff <= A(0) xor A(1);

mux0: process (A, B, C, D, E)

begin

case A is

when "00" => H <= B;

when "01" => H <= C;

when "10" => H <= D;

when "11" => H <= E;

when others => null;

end case;

end process;

mux1: process (A, B, C)

begin

if A(0) = '0' then

Fmux <= B;

else

Fmux <= C;

end if;

end process;

latch0: process (A, Fmux)

begin

if A(1) = '0' then

F <= Fmux;

end if;

end process;

latch1 : process (ADiff, B)

begin

if ADiff = '1' then

G <= B;

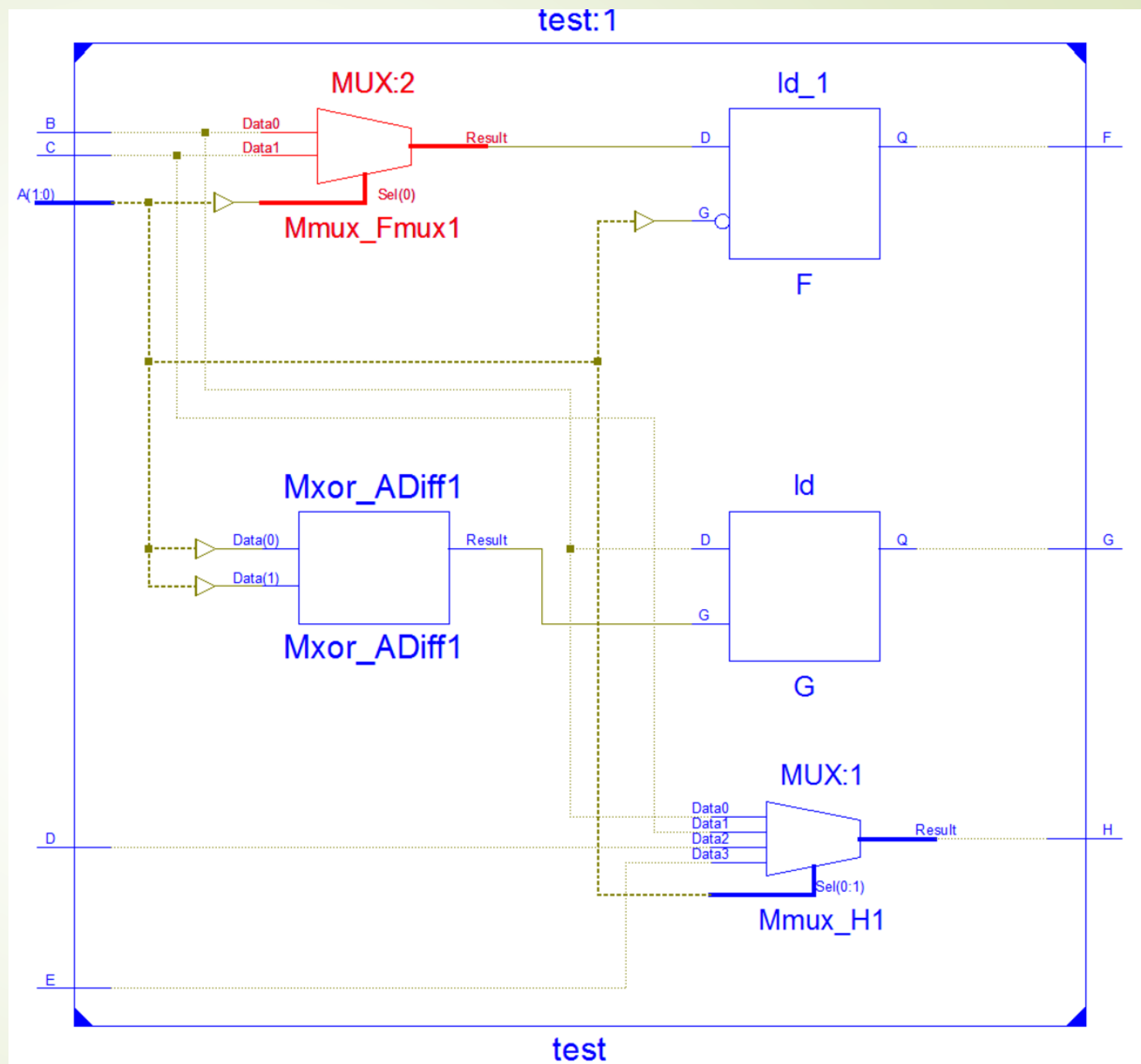
end if;


end process;

end synthesizable;

Résultat ISE

Même dessin!





## Exo 2 qA

Exactement le même résultat avec ISE mais plus propre car on affecte les signaux de sortie seulement à la fin du process.. Plus facile à lire car les signaux ne change pas de valeur avant la fin!!!!

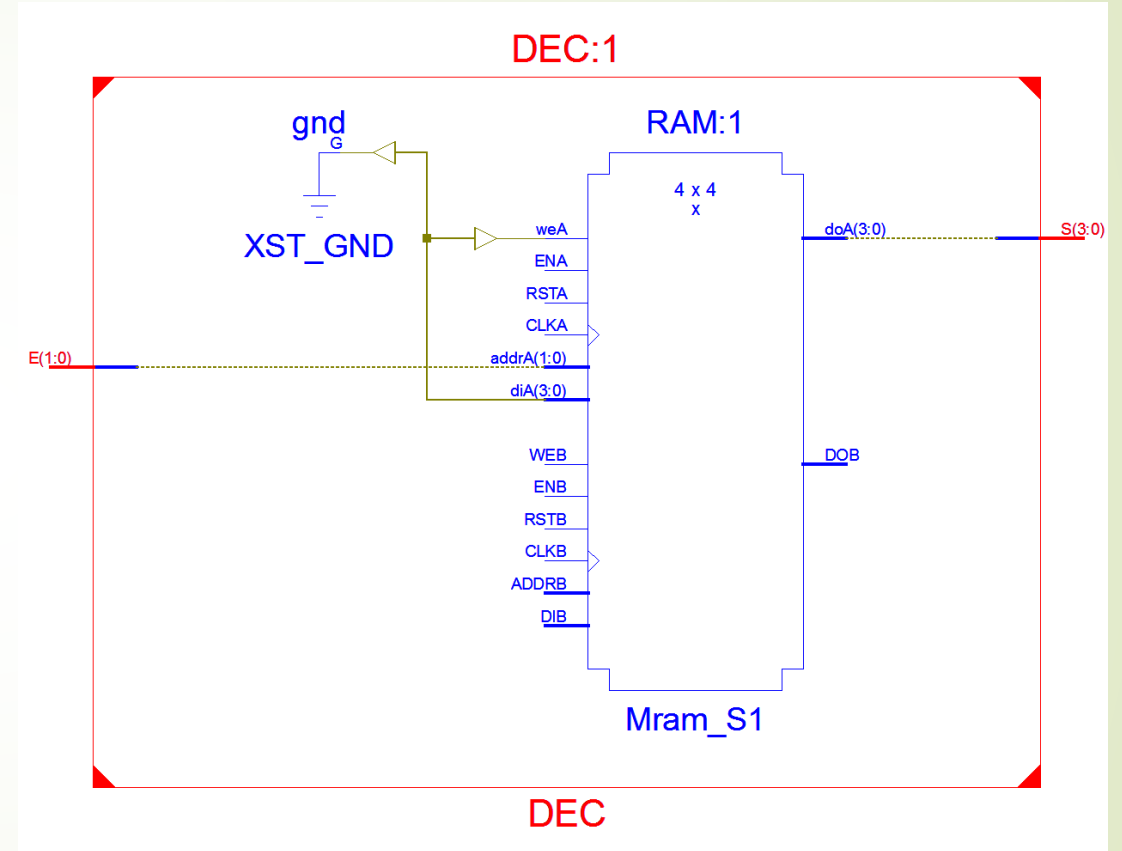
```
process (adr)
variable sout : std_logic;
begin
    case adr is
        when "00" => sout:= a;
        when "01" => sout:= b;
        when "10" => sout:= c;
        when others => sout:= d;
    end case;
    s<= sout;
end process;
```





B2

```
process (E)
begin
  S<= "0000";
  case E is
    when "00" => S(0) <= '1';
    when "01" => S(1) <= '1';
    when "10" => S(2) <= '1';
    when "11" => S(3) <= '1';
    when others => null;
  end case;
end process;
```





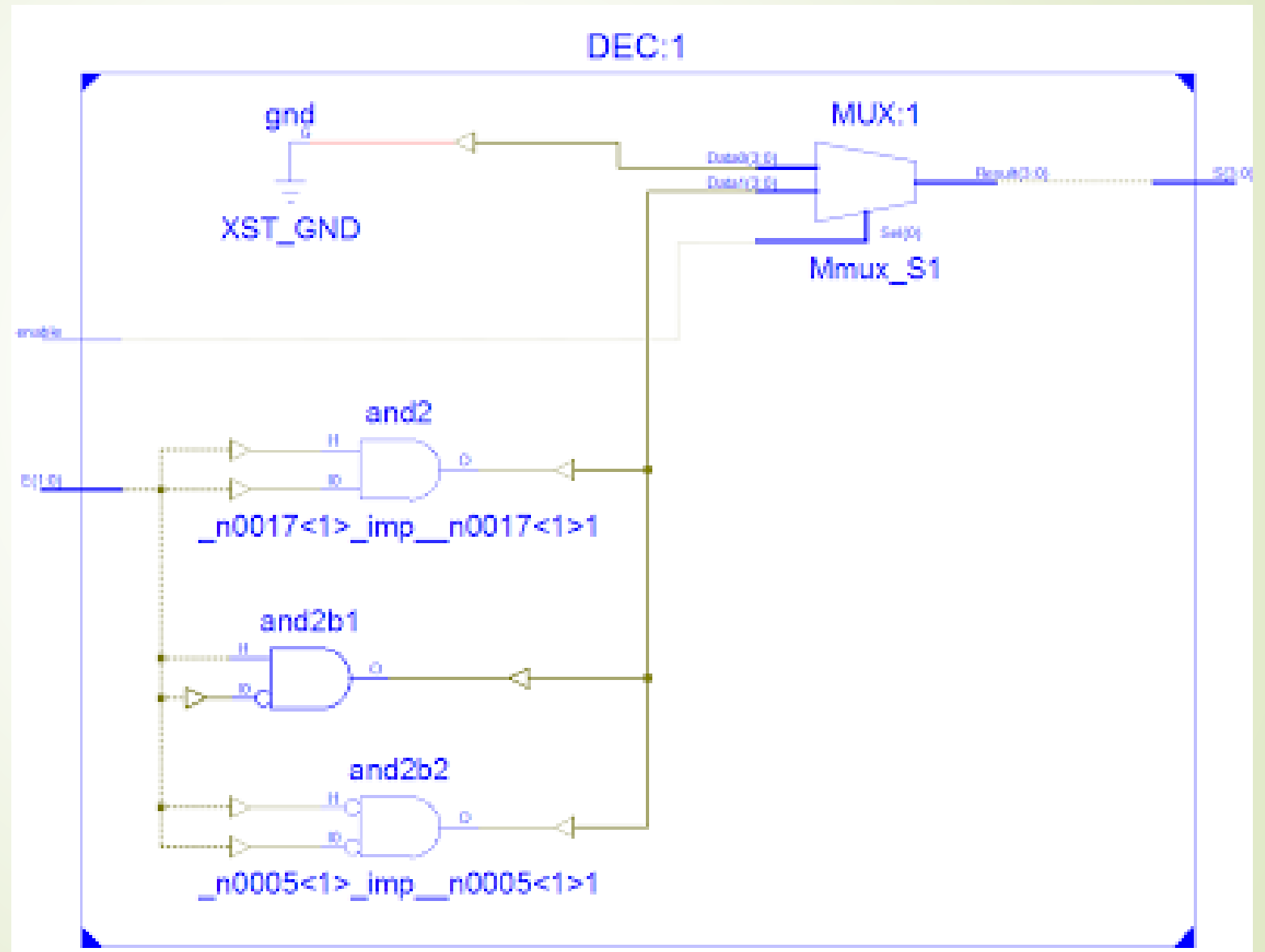
Q3

```
library ieee;  
use ieee.std_logic_1164.all;
```

```
entity DEC is port(  
    E: in std_logic_vector(1 downto 0);  
    enable : std_logic;  
    S: out std_logic_vector(3 downto 0));  
end;
```

```
architecture FLOT_MUX of DEC is  
begin  
    process (E,enable)  
    begin  
        S<= "0000";  
        if enable='1' then  
            case E is  
                when "00" => S(0) <= '1';  
                when "01" => S(1) <= '1';  
                when "10" => S(2) <= '1';  
                when "11" => S(3) <= '1';  
                when others => null;  
            end case;  
        end if;  
    end process;  
end;
```

ISE



Q4  
 architecture FLOT\_MUX of DEC is  
 signal sel : std\_logic\_vector(2 downto 0);  
 begin  
 sel <= enable & E;  
 with sel select  
 S <= "0001" when "100",  
 "0010" when "101",  
 "0100" when "110",  
 "1000" when "111",  
 "0000" when others;  
 end;

- Ici c'est une RAM 2 fois plus grande pour les 4 others!

