

Solution seance 1 et 2

Exo 1 Q1

- Trois switch de droite pour A B et Cin
- Deux led pour S et Cout

➤ Fichier UCF :

NET A LOC = T10; // switch) droite sur la NEXYS 3

NET B LOC = T9; // switch suivant

NET Cin LOC = V9; // troisième switch

NET S LOC = U16; // LED à droite

NET Scout LOC = V16; // LED suivante

Q2

Port entrée

A(0:3)

B(0:3)

Port sortie

S(0:3)

Cout

Il faut

4 switch pour A, 4 switch pour B

4 led pour S et une pour Cout

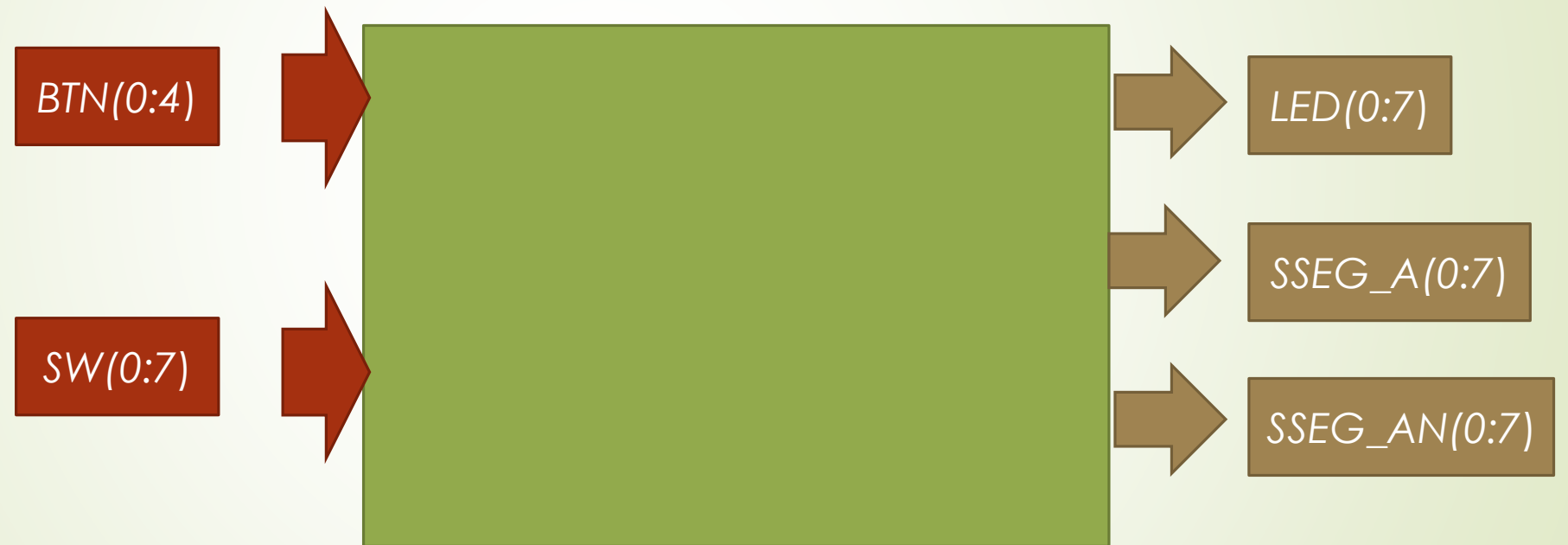
Afin de rendre les chose 'lisible' on a intérêt à mettre les 4 A dans l'ordre sur 4 switch de droite, puis 4 switch pour B

Idem pour les sortie 4 leds de droite ds l'ordre pour S puis Cout

La lecture est directe. Mais c'est un choix libre!!!!

- Net A<0> LOC=T10 ;
Net A<1> LOC=T9 ;
Net A<2> LOC=V9 ;
Net A<3> LOC=M8 ;
Net B<0> LOC=N8 ;
Net B<1> LOC=U8 ;
Net B<2> LOC=V8 ;
Net B<3> LOC=T5 ;
- Net Cout LOC=M11 ;
Net S<3> LOC=V15 ;
Net S<2> LOC=U15 ;
Net S<1> LOC=V16 ;
Net S<0> LOC=U16 ;


Exo 2 Q1





Question 2

- connecter le même switch à deux entrées différentes dans le circuit (ex les 2 entrées d'une porte OU)
 - OUI on duplique le signal
- connecter deux sorties du circuit à une même LED ?
 - NON un seul signal 0 ou 1
- connecter deux LEDs à une même sortie du circuit ?
 - OUI on voit deux fois la même chose



Q3

- On doit saisir par tranches de 8 bits
- Après chaque saisie il faut presser un BTN
- A chaque étape on décale de 8 à gauche la valeur déjà saisie et on 'ajoute' les nouveaux 8 bits
- Il faut utiliser une FSM pour ordonner les traitements
 - 3 etats pour 16 bits
 - Wait ,btn, btn
 - 4 pour 32 bits
 - Wait, btn, btn, btn, btn

Prepa TP

- Le plus simple c'est une table de vérité pour chaque segment en fonction de la valeur en entrée sur 4 bits, **attention ici actif à l'état 1, il fut inverser toutes les sorties sur la Nexys**

BINARY STATE	INPUTS				OUTPUTS							DISPLAY
	A3	A2	A1	A0	a	b	c	d	e	f	g	
0	L	L	L	L	H	H	H	H	H	H	L	0
1	L	L	L	H	L	H	H	L	L	L	L	1
2	L	L	H	L	H	H	L	H	H	L	H	2
3	L	L	H	H	H	H	H	H	L	L	H	3
4	L	H	L	L	L	H	H	L	L	H	H	4
5	L	H	L	H	H	L	H	H	L	H	H	5
6	L	H	H	L	H	L	H	H	H	H	H	6
7	L	H	H	H	H	H	H	L	L	L	L	7
8	H	L	L	L	H	H	H	H	H	H	H	8
9	H	L	L	H	H	H	H	L	L	H	H	9
10	H	L	H	L	H	H	H	L	H	H	H	A
11	H	L	H	H	L	L	H	H	H	H	H	b
12	H	H	L	L	H	L	L	H	H	H	L	c
13	H	H	L	H	L	H	H	H	H	L	H	d
14	H	H	H	L	H	L	L	H	H	H	H	e
15	H	H	H	H	H	L	L	L	H	H	H	F



Suite Q1

- On connecte les 4 switch sur les entrées du circuit obtenu
- On connecte les sorties sur les 7 segments de l'afficheur S7segs
- Il ne reste plus qu'à choisir le ou les afficheurs en plaçant une valeur sur les 4 bits de sélection aff
- UCF cf Q1 exo 2 , rendre les signaux qu'il faut avec les bons noms....

Q2

- Peut-on afficher les 8 bits à la fois avec notre afficheur ?
 - NON un seul afficheur à la fois, sinon on affiche la même chose sur les afficheurs sélectionnés
- Sinon combien de phases sont nécessaires ?
 - 2 phases, soit un btn pour passer du premier au deuxième , soit une clock qui le fait automatiquement
- Comment peut on tromper l'oeil du lecteur afin de lui faire voir les deux digits "en même temps" ?
 - Il faut une clock assez rapide pour tromper l'œil; 50Hz par exemple. E cinema c'était 24 images/seconde
- Dans ce cas, qu'est ce qui changerait dans le fichier UCF par rapport à la question 1 ?
 - Il faut prendre les 8 switches et une clock, il faudra multiplexer les 2 x 4 entrée sw[0:3) et SW(4:7) en fonction de la clock et en même temps changer le AFF pour changer d'afficheur



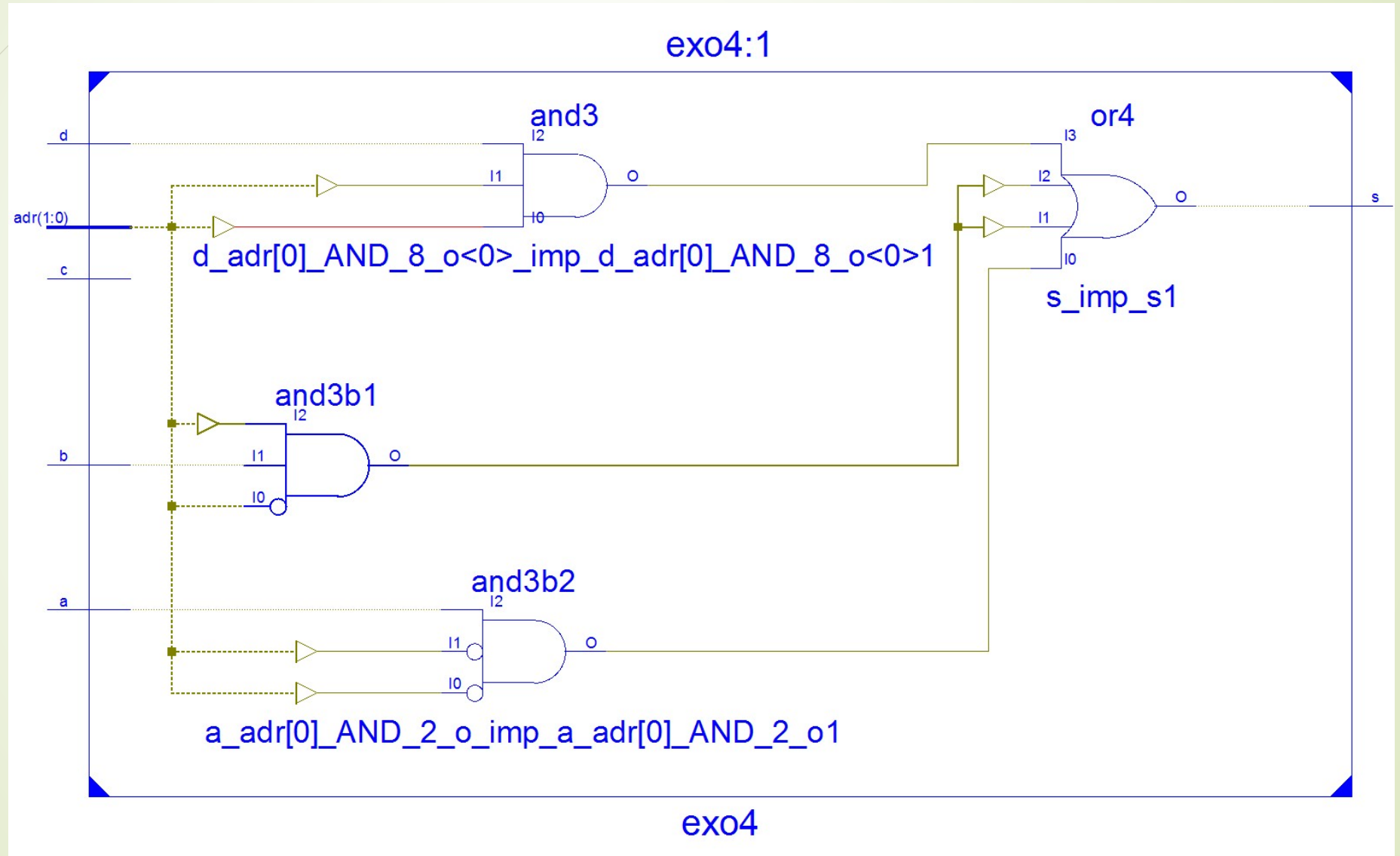
Solution séance 2

Exo 1

entity exo1 is
port (
a : in bit_vector (4 downto 0);
s : out bit_vector (2 downto 0));
end exo1 ;

architecture aexo1 of exo1 is
begin
with s select
s <= "101" when "01001",
 "011" when "10010",
 "001" when others ;
end aexo1;

Exo 2



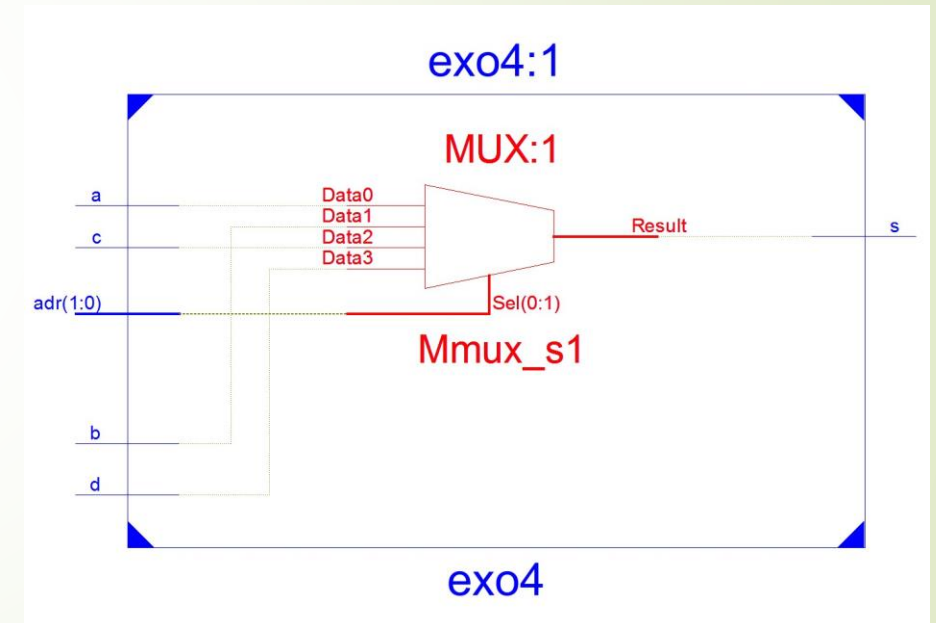


fonctionnement

- C'est un mutliplexeur sur sélecteur $\text{adr}(1:0)$

Avec with

- ARCHITECTURE Archi1exo4 OF exo4 IS
- BEGIN
- with adr select
- $s \leq$ a when "00",
- b when "01",
- c when "10",
- d when others;
- END Archi1exo4;



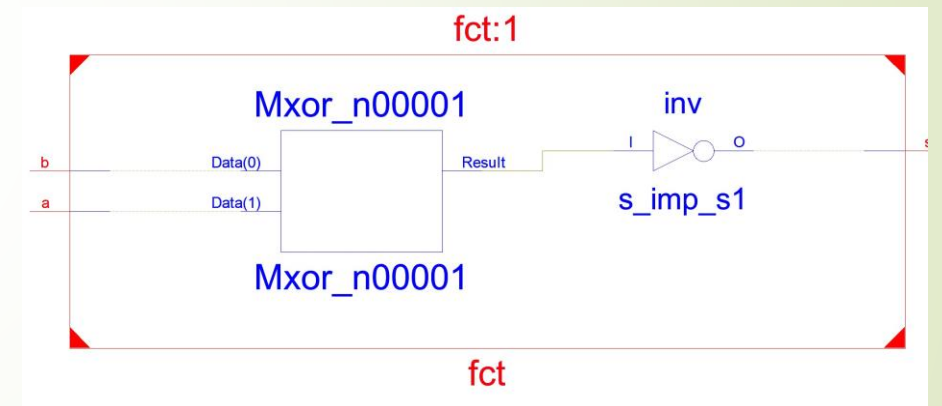
Avec when

```
➤ ARCHITECTURE Archi1exo4 OF  
  exo4 IS  
  
➤ BEGIN  
  
➤     s <=  a when adr ="00" else  
➤           b when adr ="01" else  
➤           c when adr ="10" else  
➤           d ;  
  
➤ END Archi1exo4;
```

➤ Même circuit obtenu par ISE

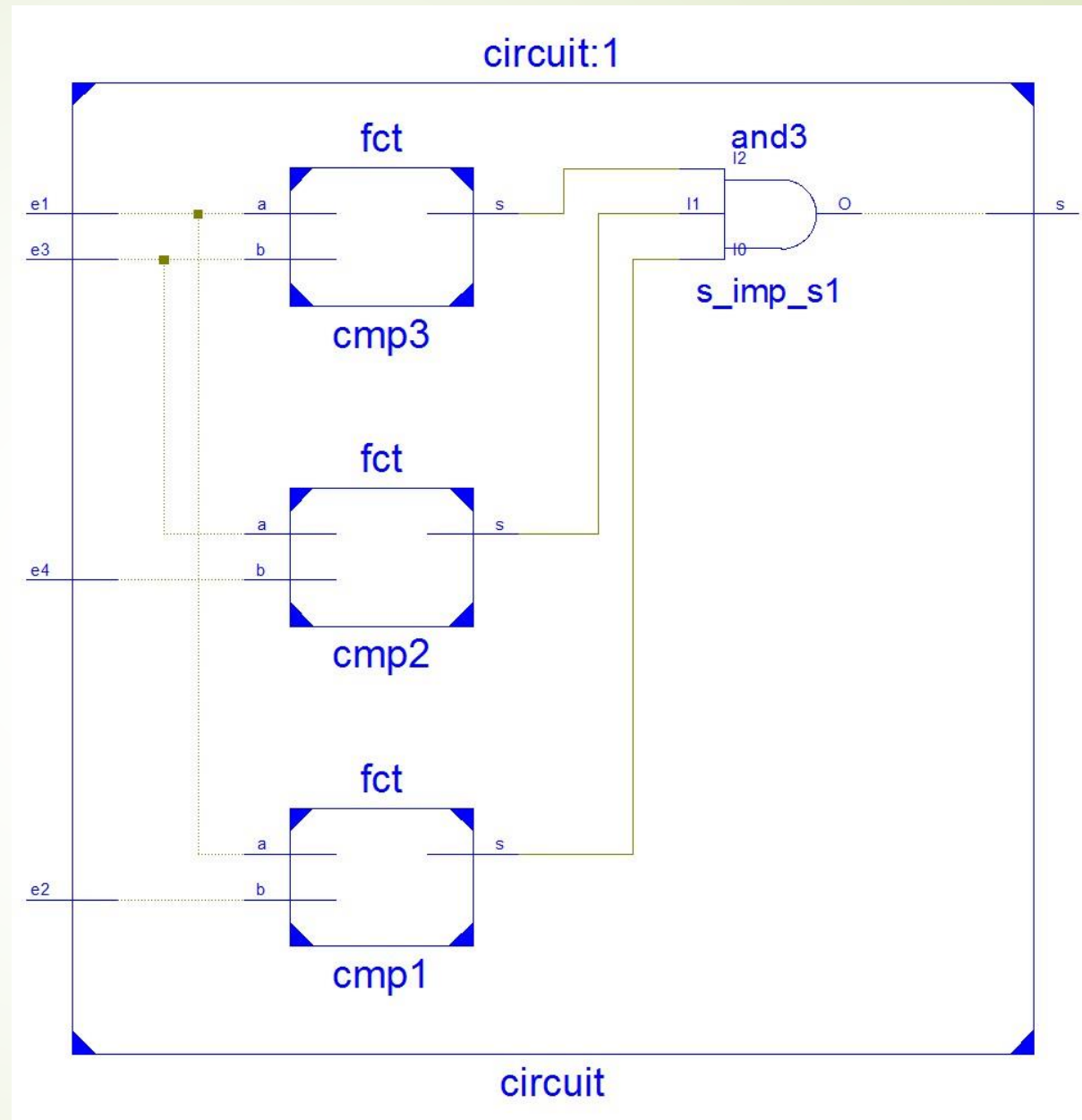
Exo 43

- Soit comparateur 1 si egal 0 sinon
- Soit un NXOR
- Il manque les biblio

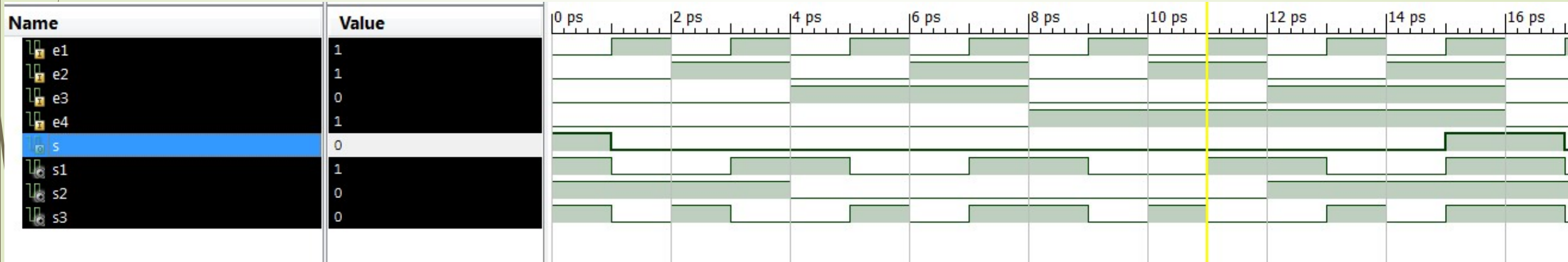


Exo 3 Q2

= 1 si les 4 entrées
sont égales

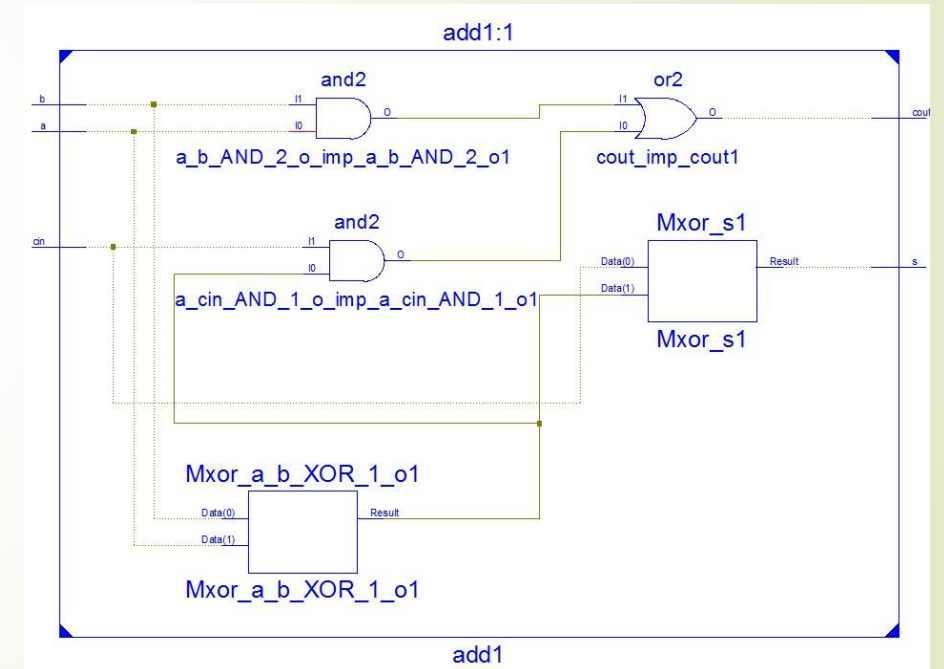


Par Isim



Exo 4

- entity add1 is
- Port (a : in STD_LOGIC;
- b : in STD_LOGIC;
- cin : in STD_LOGIC;
- s : out STD_LOGIC;
- cout : out STD_LOGIC);
- end add1;
- architecture Behavioral of add1 is
- begin
- s<= (a xor b) xor cin;
- cout<= (((a xor b) and cin) or (a and b));
- end Behavioral;



Add 4 x 1

```
entity add4x1 is
  Port ( a : in STD_LOGIC_VECTOR (3 downto 0);
        b : in STD_LOGIC_VECTOR (3 downto 0);
        s : out STD_LOGIC_VECTOR (3 downto 0);
        cin : in STD_LOGIC;
        cout : out STD_LOGIC);
end add4x1;

architecture Behavioral of add4x1 is
  COMPONENT add1
  PORT(  a : IN std_logic;
        b : IN std_logic;
        cin : IN std_logic;
        s : OUT std_logic;
        cout : OUT std_logic );
  END COMPONENT;
```

```
  signal c1, c2, c3 : std_logic;
begin
  Inst0: add1 PORT MAP(
    a => a(0), b => b(0), cin => cin, s => s(0), cout => c1);
  Inst1: add1 PORT MAP(
    a => a(1), b => b(1), cin => c1, s => s(1), cout => c2);
  Inst2: add1 PORT MAP(
    a => a(2), b => b(2), cin => c2, s => s(2), cout => c3);
  Inst3: add1 PORT MAP(
    a => a(3), b => b(3), cin => c3, s => s(3), cout => cout);
end Behavioral;
```