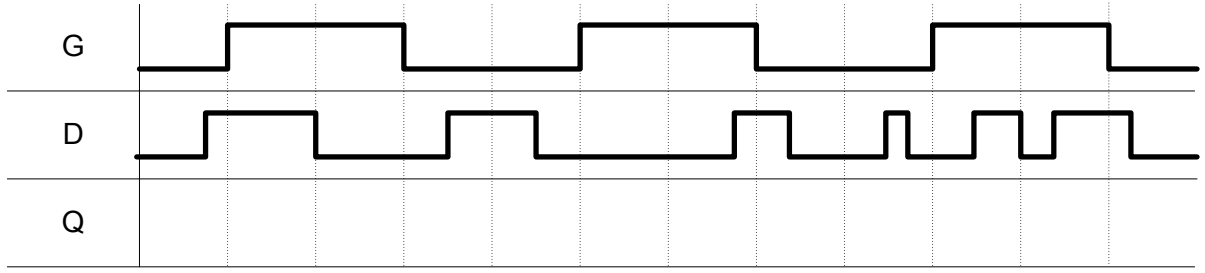
VHDL pour circuits séquentiels

Sujets de ce thème

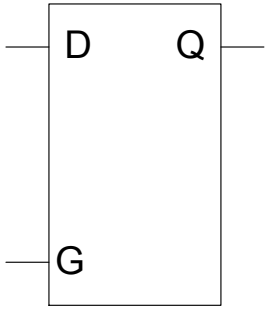
- Code VHDL du loquet.
- Code VHDL de la bascule: fronts d'horloge et initialisation.
- Code VHDL correspondant à un circuit séquentiel.

Loquet D: code VHDL

- Un loquet D peut-être modélisé en VHDL par un énoncé `if-then` à l'intérieur d'un processus.
- Le processus doit avoir dans sa liste de sensitivité le signal de contrôle *G* et le signal de donnée *D*.
- Le signal *D* est assigné à la sortie *Q* quand le signal de contrôle *G* est actif (mode transparent)




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

entity loquetD is
   port (
      G : in STD_LOGIC; -- contrôle
      D : in STD_LOGIC;  -- donnée
      Q : out STD_LOGIC
   );
end loquetD;

architecture loquetD of loquetD is
Begin

   process(G, D) is
   begin
      if (G = '1') then
         Q <= D;
--    else  -- implicite, infère élément à mémoire
--          -- ne pas changer Q
      end if;
   end process;

end loquetD;
```



Patron de code spécial reconnu par les synthétiseurs pour signifier un loquet.

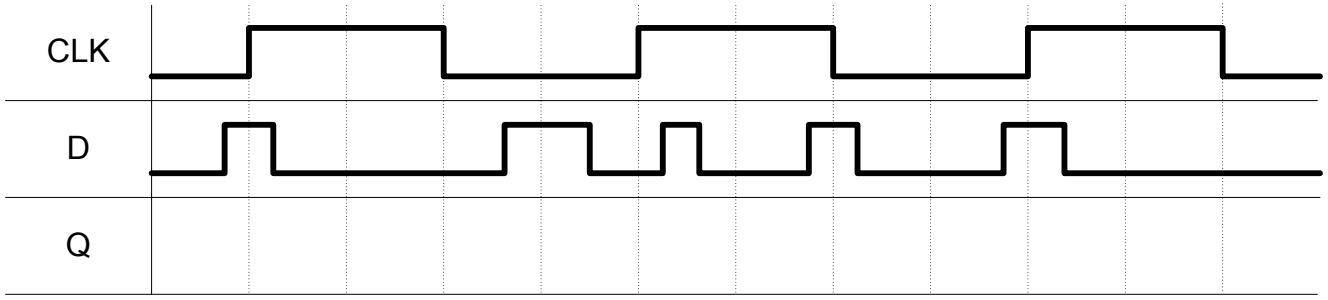
Exemple: un loquet ou pas?

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

entity mystere1 is
   port (a, b, c: in std_logic;
      s : in std_logic_vector (1 downto 0);
      o : out std_logic);
end mystere1;

architecture archi of mystere1 is
begin

   process (a, b, c, s)
   begin

      if (s = "00") then o <= a;
         elsif (s = "01") then o <= b;
         elsif (s = "10") then o <= c;
      end if;

   end process;

end archi;
```

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

entity mystere2 is
   port (a, b, c: in std_logic;
      s : in std_logic_vector (1 downto 0);
      o : out std_logic);
end mystere2;

architecture archi of mystere2 is
begin

   process (a, b, c, s)
   begin

      if (s = "00") then o <= a;
         elsif (s = "01") then o <= b;
         elsif (s = "10") then o <= c;
         else o <= c;
      end if;

   end process;

end archi;
```

Bascule D: code VHDL

- Pour modéliser une bascule, il est nécessaire de pouvoir décrire le fait que le changement d'état se produit sur une *transition* d'un signal d'horloge et non sur sa *valeur*.
- Pour ce faire, on peut utiliser les attributs d'événement (*event attribute*) définis en VHDL.
- L'exemple démontre l'utilisation de l'attribut `event` sur le signal CLK, dénoté par `CLK'event`. La condition CLK = '1' dénote alors un front montant.




```
library ieee;
use ieee.std_logic_1164.all;
entity basculeD is
   port (
      CLK : in STD_LOGIC; -- horloge
      D : in STD_LOGIC;  -- entrée
      Q : out STD_LOGIC -- sortie
   );
end basculeD;

architecture basculeD of basculeD is
begin

   process(CLK) is
   begin
      if (CLK = '1' and CLK'event) then
         Q <= D;
      end if;
   end process;

end basculeD;
```

Patron de code spécial reconnu par les synthétiseurs pour signifier une bascule

Bascule D: code VHDL - spécification du front d'horloge désiré

- Le package `std_logic_1164` contient aussi deux fonctions qui combinent ces conditions, `rising_edge()` et `falling_edge()`. Ces deux fonctions retournent des valeurs booléennes.

comportement désiré	option 1	option 2
front montant	CLK'event and CLK = '1'	rising_edge(CLK)
front descendant	CLK'event and CLK = '0'	falling_edge(CLK)

Bascule D: code VHDL - deux types de signaux d'initialisation

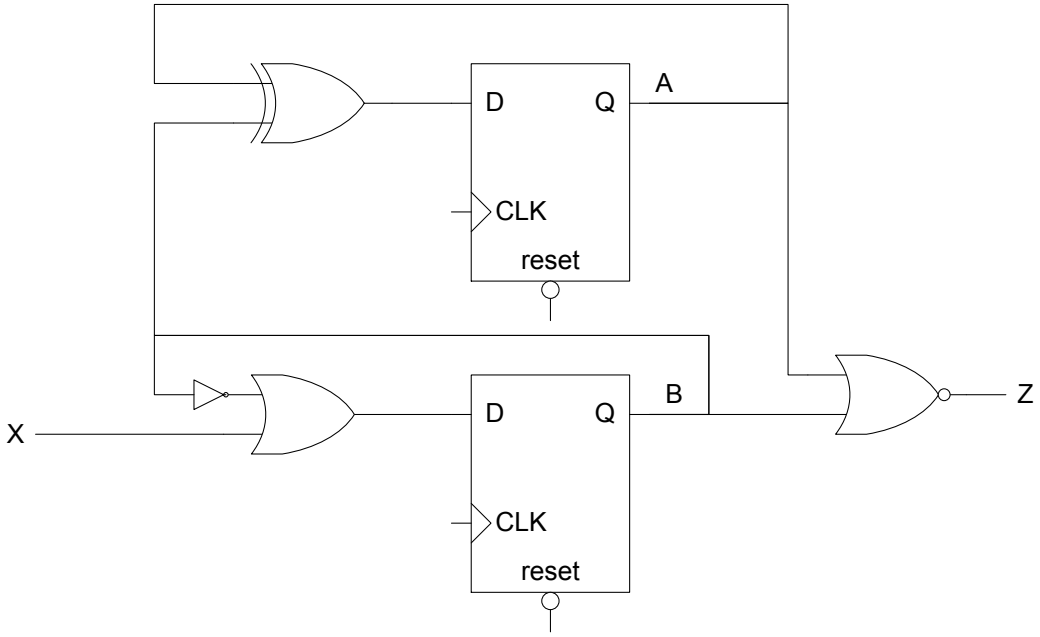
```
library IEEE;
use IEEE.std_logic_1164.all;

entity basculeDRA is
   port (
      reset : in STD_LOGIC;
      CLK : in STD_LOGIC;
      D : in STD_LOGIC;
      Q : out STD_LOGIC
   );
end basculeDRA;

architecture basculeDRasynch of basculeDRA is
begin

   process(CLK, reset) is
   begin
      if (reset = '0') then
         Q <= '0';
      elsif (rising_edge(CLK)) then
         Q <= D;
      end if;
   end process;

end basculeDRasynchA;
```

Initialisation asynchrone – indépendante de l'horloge

```
library IEEE;
use IEEE.std_logic_1164.all;

entity basculeDRS is
   port (
      reset : in STD_LOGIC;
      CLK : in STD_LOGIC;
      D : in STD_LOGIC;
      Q : out STD_LOGIC
   );
end basculeDRS;

architecture basculeDRsynch of basculeDRS is
begin

   process(CLK, reset) is
   begin
      if (rising_edge(CLK)) then
         if (reset = '0') then
            Q <= '0';
         else
            Q <= D;
         end if;
      end if;
   end process;

end basculeDRsynch;
```

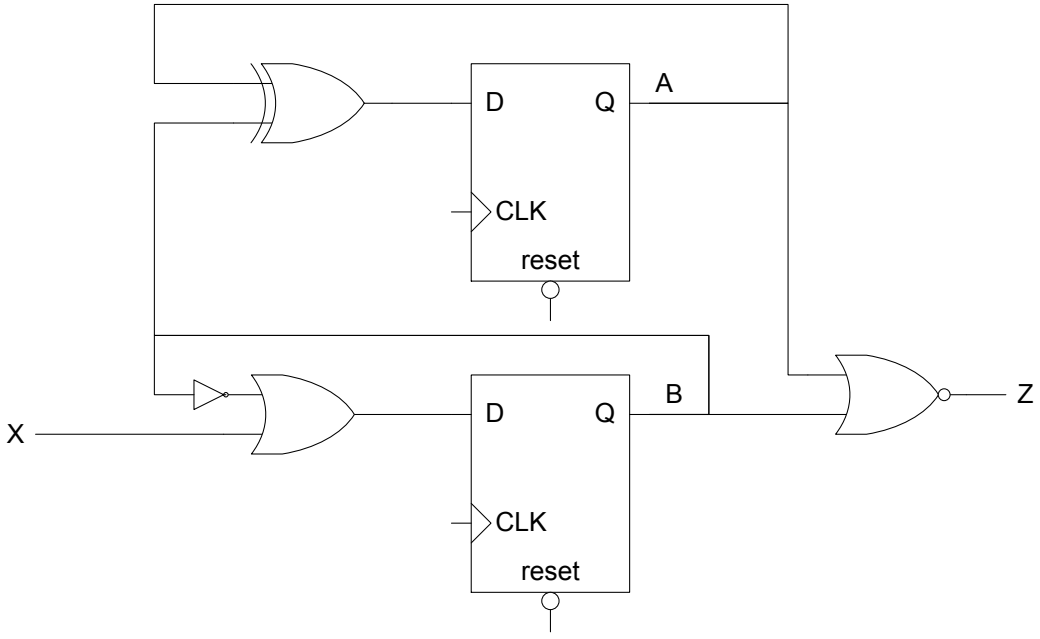
Initialisation synchrone avec l'horloge

Exemple: décrire le circuit en VHDL



1. Définir l'entité et ses ports.
2. Définir l'architecture et déclarer des signaux pour les bascules.
3. Modéliser les bascules:
   a. Forme générale
   b. Signaux de réinitialisation
   c. Équation des entrées des bascules
4. Modéliser la sortie

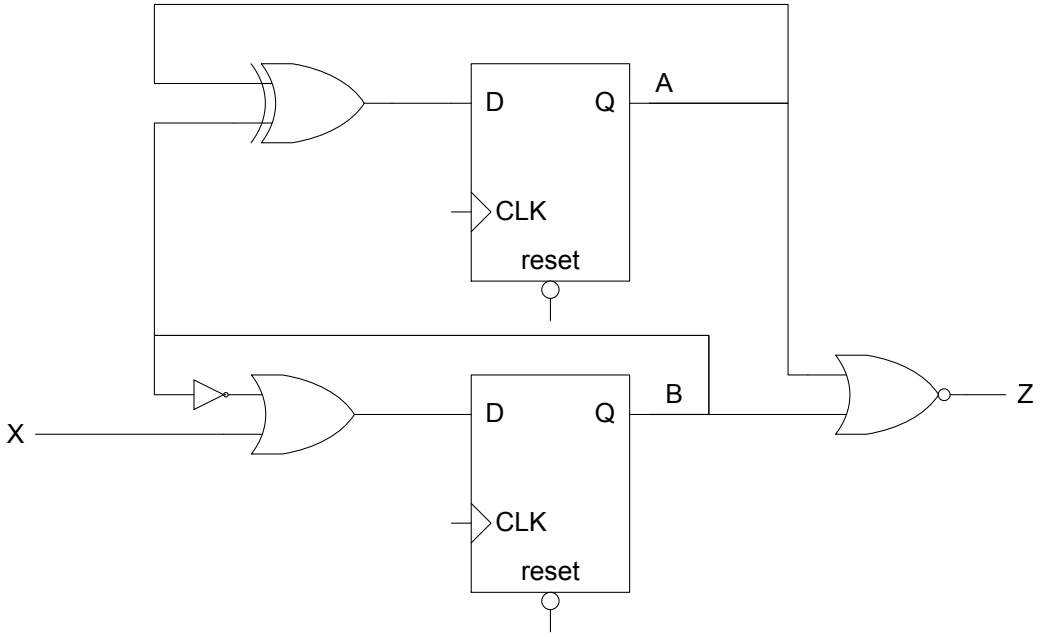
Exemple: décrire le circuit en VHDL (1)




```
library IEEE;
use IEEE.std_logic_1164.all;

entity cctsequentielex1 is
   port (
      reset, CLK, X : in STD_LOGIC;
      Z : out STD_LOGIC
   );
end cctsequentielex1;
```

1. Définir l'entité et ses ports.
2. Définir l'architecture et déclarer des signaux pour les bascules.
3. Modéliser les bascules:
   a. Forme générale
   b. Signaux de réinitialisation
   c. Équation des entrées des bascules
4. Modéliser la sortie

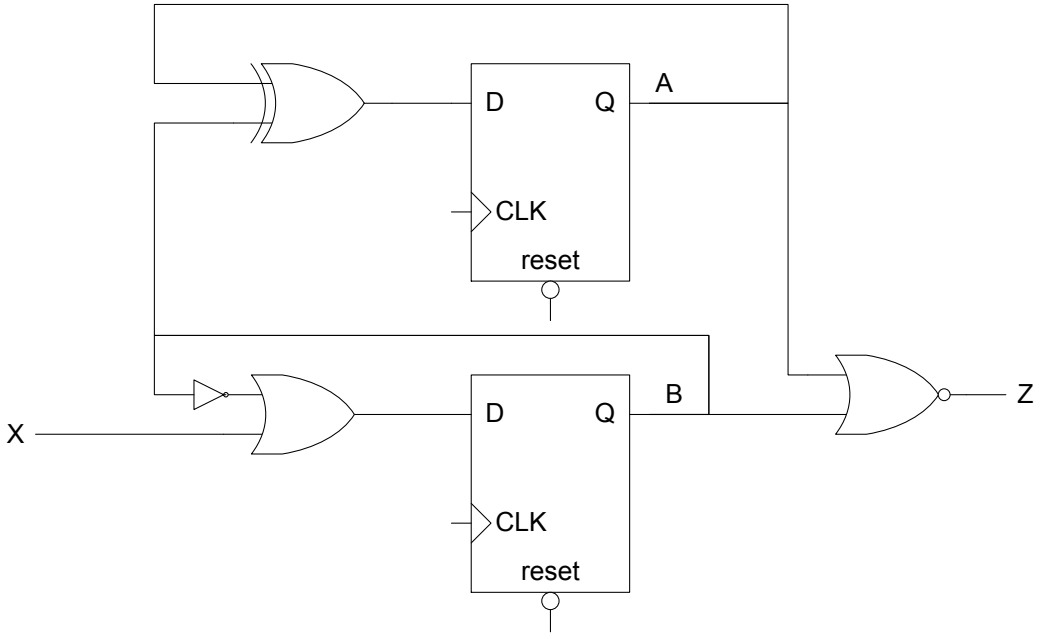
Exemple: décrire le circuit en VHDL (2)



1. Définir l'entité et ses ports.
2. Définir l'architecture et déclarer des signaux pour les bascules.
3. Modéliser les bascules:
   a. Forme générale
   b. Signaux de réinitialisation
   c. Équation des entrées des bascules
4. Modéliser la sortie

```
library IEEE;
use IEEE.std_logic_1164.all;

entity cctsequentielex1 is
   port (
      reset, CLK, X : in STD_LOGIC;
      Z : out STD_LOGIC
   );
end cctsequentielex1;

architecture arch1 of cctsequentielex1 is
signal A, B : STD_LOGIC;
begin









end arch1;
```

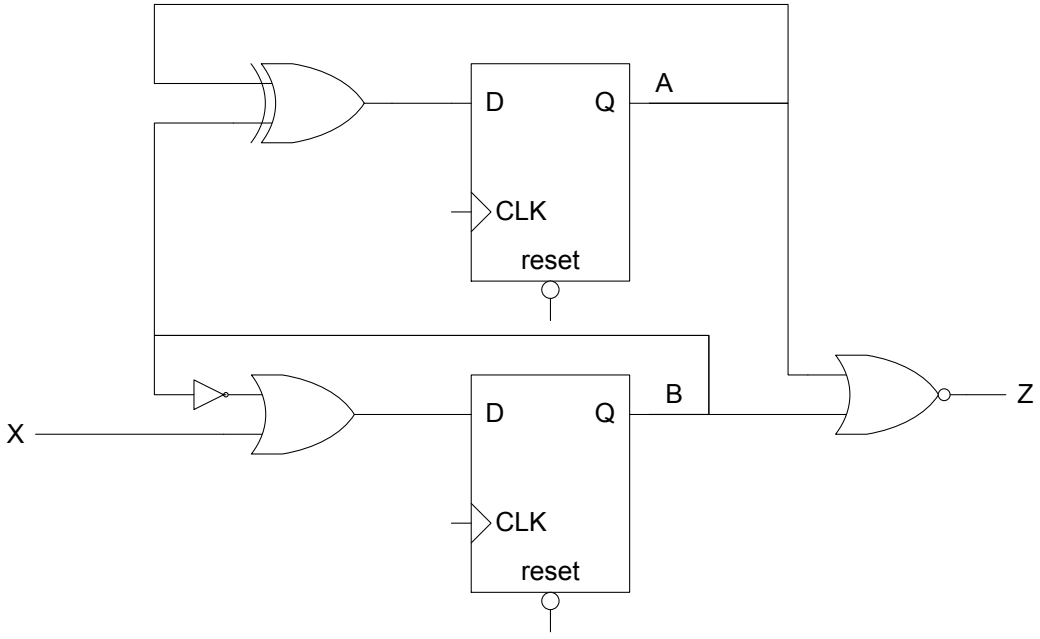
Exemple: décrire le circuit en VHDL (3)



1. Définir l'entité et ses ports.
2. Définir l'architecture et déclarer des signaux pour les bascules.
3. Modéliser les bascules:
   a. Forme générale
   b. Signaux de réinitialisation
   c. Équation des entrées des bascules
4. Modéliser la sortie

```
library IEEE;
use IEEE.std_logic_1164.all;

entity cctsequentielex1 is
   port (
      reset, CLK, X : in STD_LOGIC;
      Z : out STD_LOGIC
   );
end cctsequentielex1;

architecture arch1 of cctsequentielex1 is
signal A, B : STD_LOGIC;
begin

   process(CLK, reset) is
   begin
       if (reset = '0') then
         A <= '0';
         B <= '0';
       elsif (rising_edge(CLK)) then
         A <=
         B <=
       end if;
   end process;




end arch1;
```

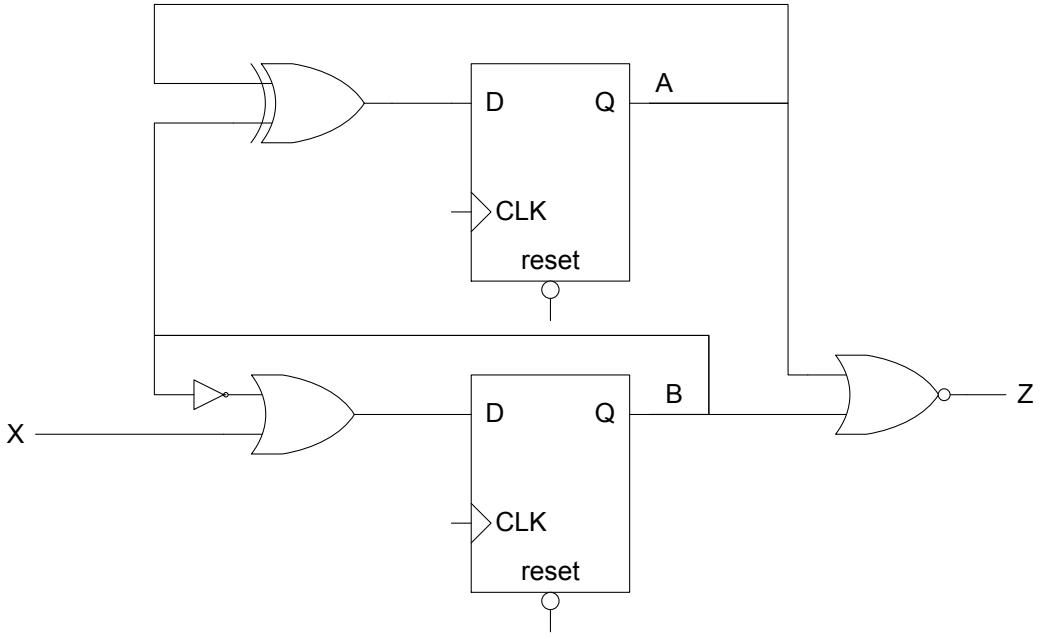
Exemple: décrire le circuit en VHDL (4)



1. Définir l'entité et ses ports.
2. Définir l'architecture et déclarer des signaux pour les bascules.
3. Modéliser les bascules:
   a. Forme générale
   b. Signaux de réinitialisation
   c. Équation des entrées des bascules
4. Modéliser la sortie

```
library IEEE;
use IEEE.std_logic_1164.all;

entity cctsequentielex1 is
   port (
      reset, CLK, X : in STD_LOGIC;
      Z : out STD_LOGIC
   );
end cctsequentielex1;

architecture arch1 of cctsequentielex1 is
signal A, B : STD_LOGIC;
begin

   process(CLK, reset) is
   begin
       if (reset = '0') then
         A <= '0';
         B <= '0';
       elsif (rising_edge(CLK)) then
         A <= A xor B;
         B <= x or not(B);
       end if;
   end process;



end arch1;
```

Exemple: décrire le circuit en VHDL (5)

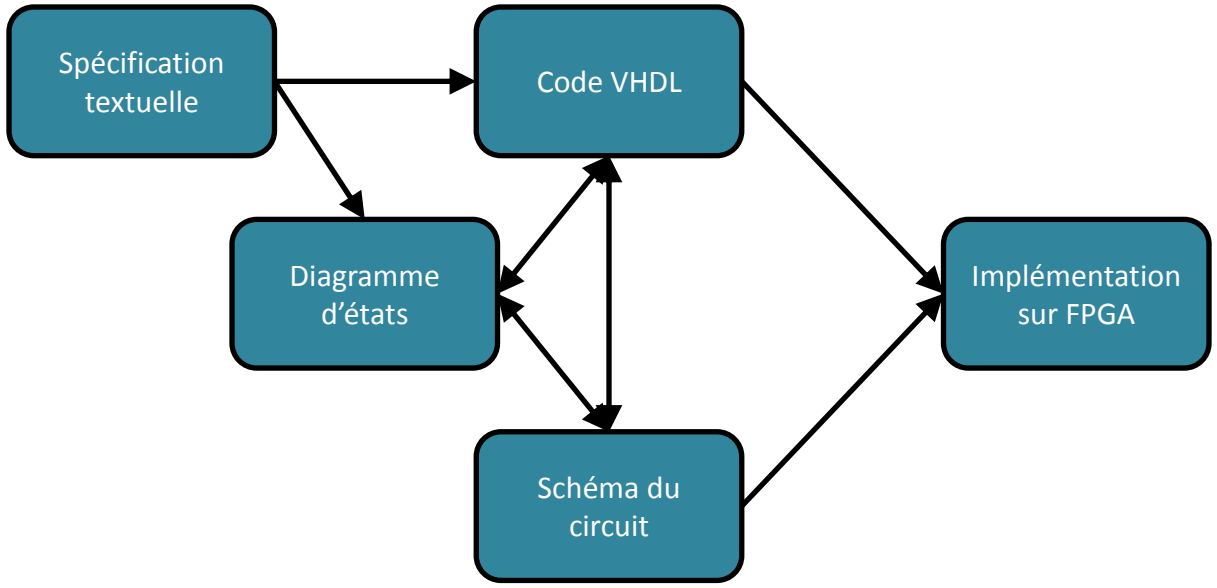


1. Définir l'entité et ses ports.
2. Définir l'architecture et déclarer des signaux pour les bascules.
3. Modéliser les bascules:
   a. Forme générale
   b. Signaux de réinitialisation
   c. Équation des entrées des bascules
4. Modéliser la sortie

```
library IEEE;
use IEEE.std_logic_1164.all;

entity cctsequentielex1 is
   port (
      reset, CLK, X : in STD_LOGIC;
      Z : out STD_LOGIC
   );
end cctsequentielex1;

architecture arch1 of cctsequentielex1 is
signal A, B : STD_LOGIC;
begin

   process(CLK, reset) is
   begin
       if (reset = '0') then
         A <= '0';
         B <= '0';
       elsif (rising_edge(CLK)) then
         A <= A xor B;
         B <= x or not(B);
       end if;
   end process;

   z <= not(A or B);

end arch1;
```

Vous devriez maintenant être capable de …

- Donner le code VHDL pour modéliser un loquet et une bascule, incluant le signal de réinitialisation. (B3)
- Identifier dans un code VHDL si un loquet, une bascule ou aucun des deux n'est modélisé. (B3)
- Donner le code VHDL correspondant au schéma d'un circuit séquentiel. (B3)



Code	Niveau (http://fr.wikipedia.org/wiki/Taxonomie_de_Bloom)
B1	Connaissance - mémoriser de l'information.
B2	Compréhension – interpréter l'information.
B3	Application – confronter les connaissances à des cas pratiques simples.
B4	Analyse – décomposer un problème, cas pratiques plus complexes.
B5	Synthèse – expression personnelle, cas pratiques plus complexes.