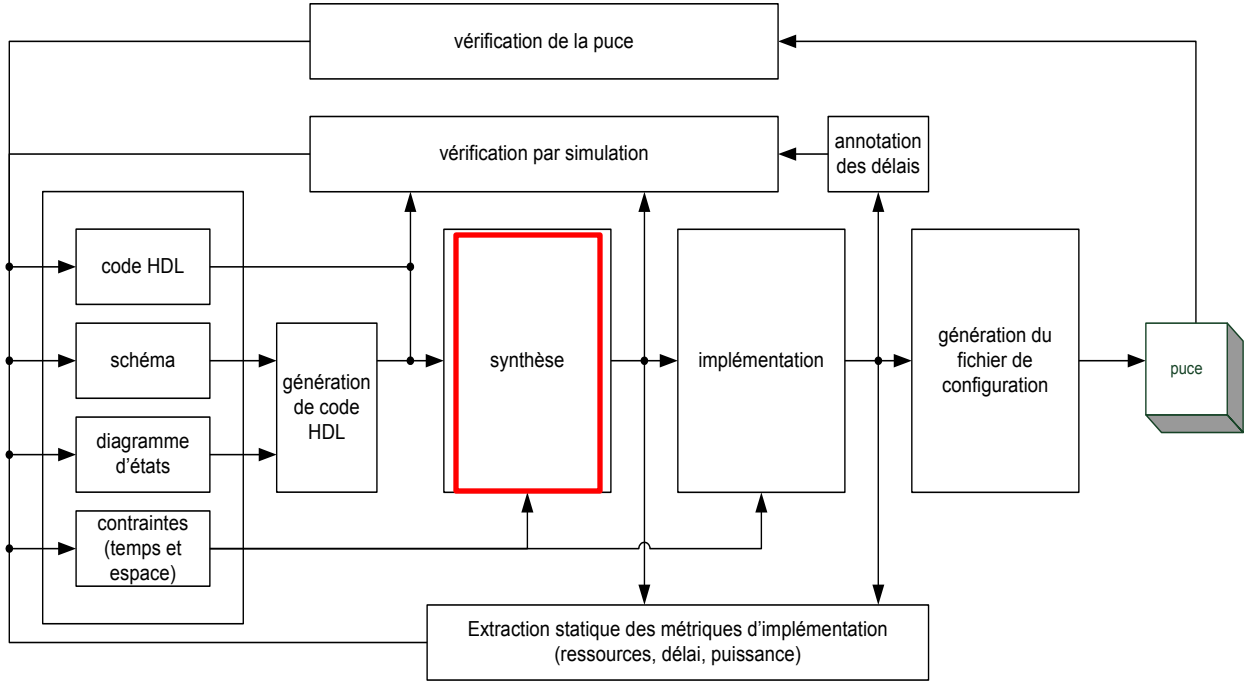
Synthèse d'un circuit séquentiel

Sujets de ce thème

- Reconnaître des bascules et des loquets dans du code VHDL.
- Étapes de synthèse.
- Schéma correspondant au code VHDL.
- Implémenter un circuit séquentiel sur un FPGA.
- Bonnes pratiques de modélisation de circuits séquentiels pour FPGA.

La synthèse d'un modèle VHDL

- La synthèse du code VHDL est effectuée par un synthétiseur.
- Le processus de synthèse peut être décomposé et effectué en plusieurs passes. Ce processus est très complexe sauf pour les circuits les plus simples.
- Le produit du synthétiseur est communément appelé «liste des interconnexions» (*netlist*), qui inclut:
  - la liste des composantes de base utilisées; et,
  - les liens qui les relient.
- Après le processus de synthèse, il est possible d'obtenir un estimé de la performance et des ressources utilisées par le circuit.



Reconnaître les loquets et bascules - rappel

- Les bascules et les loquets sont reconnus par des patrons de code spécifiques.
- Un loquet est reconnu par un énoncé `if-then` où toutes les conditions ne sont pas couvertes.
- Une bascule est reconnue par l'assignation d'un signal ou d'une variable à l'intérieur d'un processus, à l'intérieur d'une condition de transition sur un signal d'horloge.

```
process(G, D) is
begin
   if (G = '1') then
      Q <= D;
   end if;
end process;
```

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

Étapes de la synthèse d'un modèle séquentiel décrit en VHDL

1. Identifier les entrées et sorties du circuit.
2. Identifier les éléments à mémoire.
3. Pour chaque élément à mémoire, trouver l'équation booléenne de son entrée. Réduire l'équation si nécessaire. Identifier les entrées spéciales: horloge et initialisation.
4. Pour chaque sortie, trouver l'équation booléenne ou la table de vérité correspondante. Réduire l'équation si nécessaire.
5. Si la cible est un FPGA, découper les équations selon la taille des tables de correspondance disponibles.
6. Implémentation: choisir des ressources spécifiques sur la puce pour les ports et les fonctions logiques.

```
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;
```

Exemple: donner le schéma correspondant à ce module

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

entity cctsequentielex4 is
   port (
      clk, reset : in STD_LOGIC;
      A, B, C: in STD_LOGIC;
      U, V, W, X, Y, Z : out STD_LOGIC
   );
end cctsequentielex4;

architecture arch of cctsequentielex4 is
begin

   U <= B xor C;

   process(A, B)
   begin
      V <= A and B;
   end process;
```

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

   process(A, C) is
   begin
      if (A = '0') then
         Y <= C nor B;
         Z <= not(B);
      else
         Z <= B nand C;
      end if;
   end process;

end arch;
```

Exemple: donner le schéma correspondant à ce module (1)

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

entity cctsequentielex4 is
   port (
      clk, reset : in STD_LOGIC;
      A, B, C: in STD_LOGIC;
      U, V, W, X, Y, Z : out STD_LOGIC
   );
end cctsequentielex4;

architecture arch of cctsequentielex4 is
begin

   U <= B xor C;

   process(A, B)
   begin
      V <= A and B;
   end process;

   ...

end arch;
```

Exemple: donner le schéma correspondant à ce module (2)

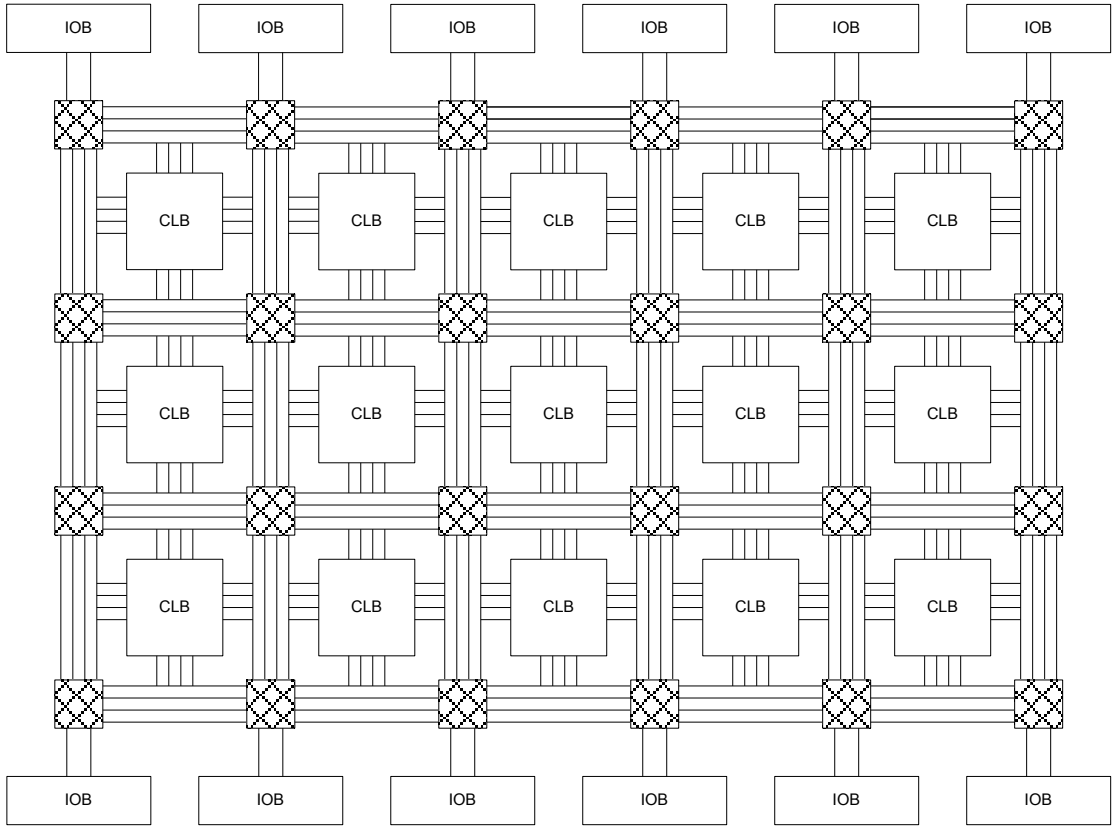
```
library IEEE;
use IEEE.std_logic_1164.all;

entity cctsequentielex4 is
   port (
      clk, reset : in STD_LOGIC;
      A, B, C: in STD_LOGIC;
      U, V, W, X, Y, Z : out STD_LOGIC
   );
end cctsequentielex4;

architecture arch of cctsequentielex4 is
begin

   ...

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

   ...

end arch;
```

Exemple: donner le schéma correspondant à ce module (3)

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

entity cctsequentielex4 is
   port (
      clk, reset : in STD_LOGIC;
      A, B, C: in STD_LOGIC;
      U, V, W, X, Y, Z : out STD_LOGIC
   );
end cctsequentielex4;

architecture arch of cctsequentielex4 is
begin

   ...

   process(A, C) is
   begin
      if (A = '0') then
         Y <= C nor B;
         Z <= not(B);
      else
         Z <= B nand C;
      end if;
   end process;


end arch;
```

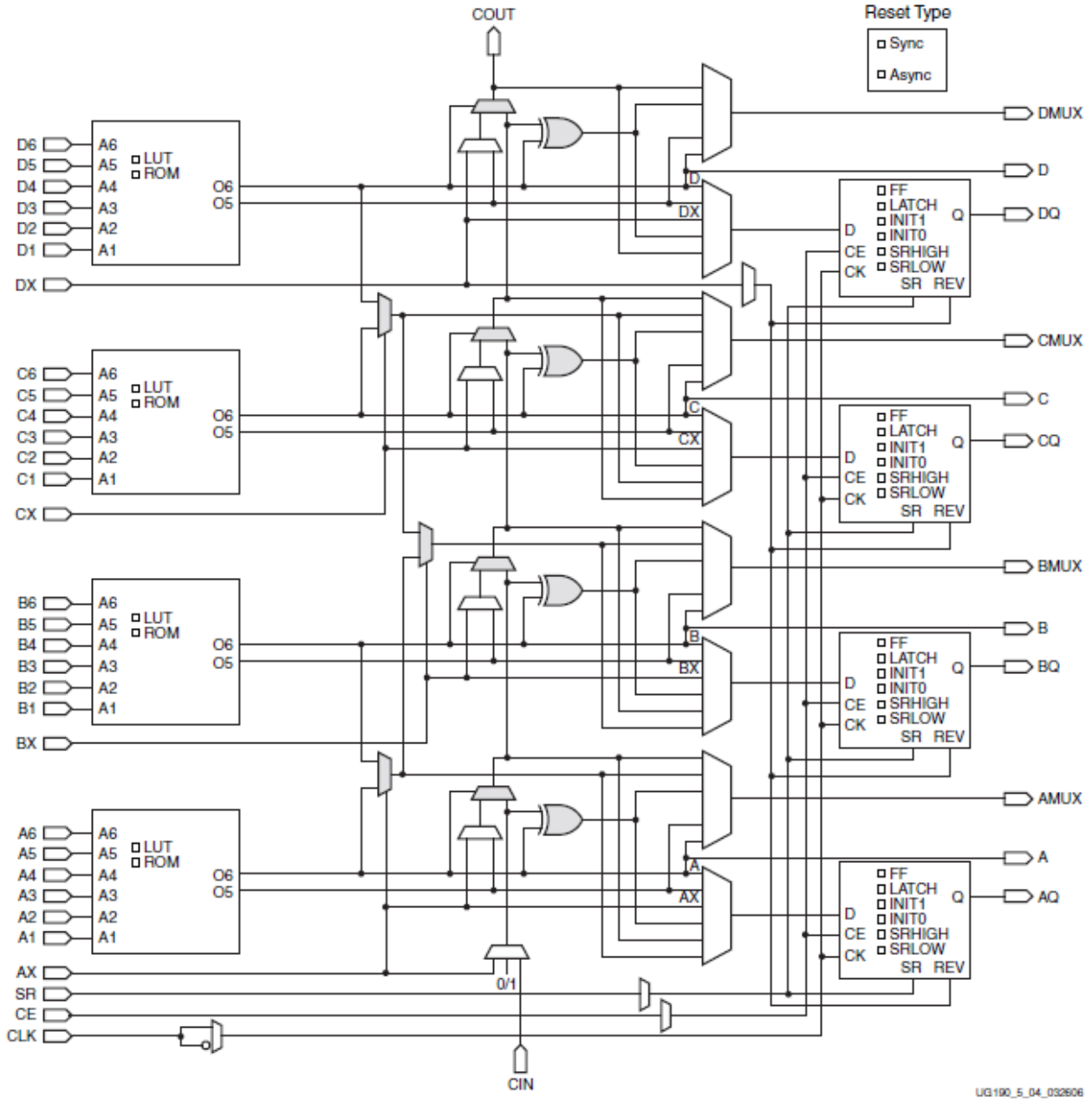
Rappel: architecture d'un FPGA

- Un FPGA est composé à la base de :
  - un réseau de blocs de logique programmable (Configurable Logic Block – CLB);
  - un réseau d'interconnexions programmables entre les blocs; et,
  - des blocs d'entrée et de sortie avec le monde extérieur (Input/Output Block – IOB).
- Dans la figure, on a:
  - 12 IOBs, 3 × 5 CLBs
- Le FPGA XC5VLX50TFFG1136C a plutôt:
  - 480 IOBs, 120 × 30 CLBs



Rappel: tranche d'un FPGA Virtex 5

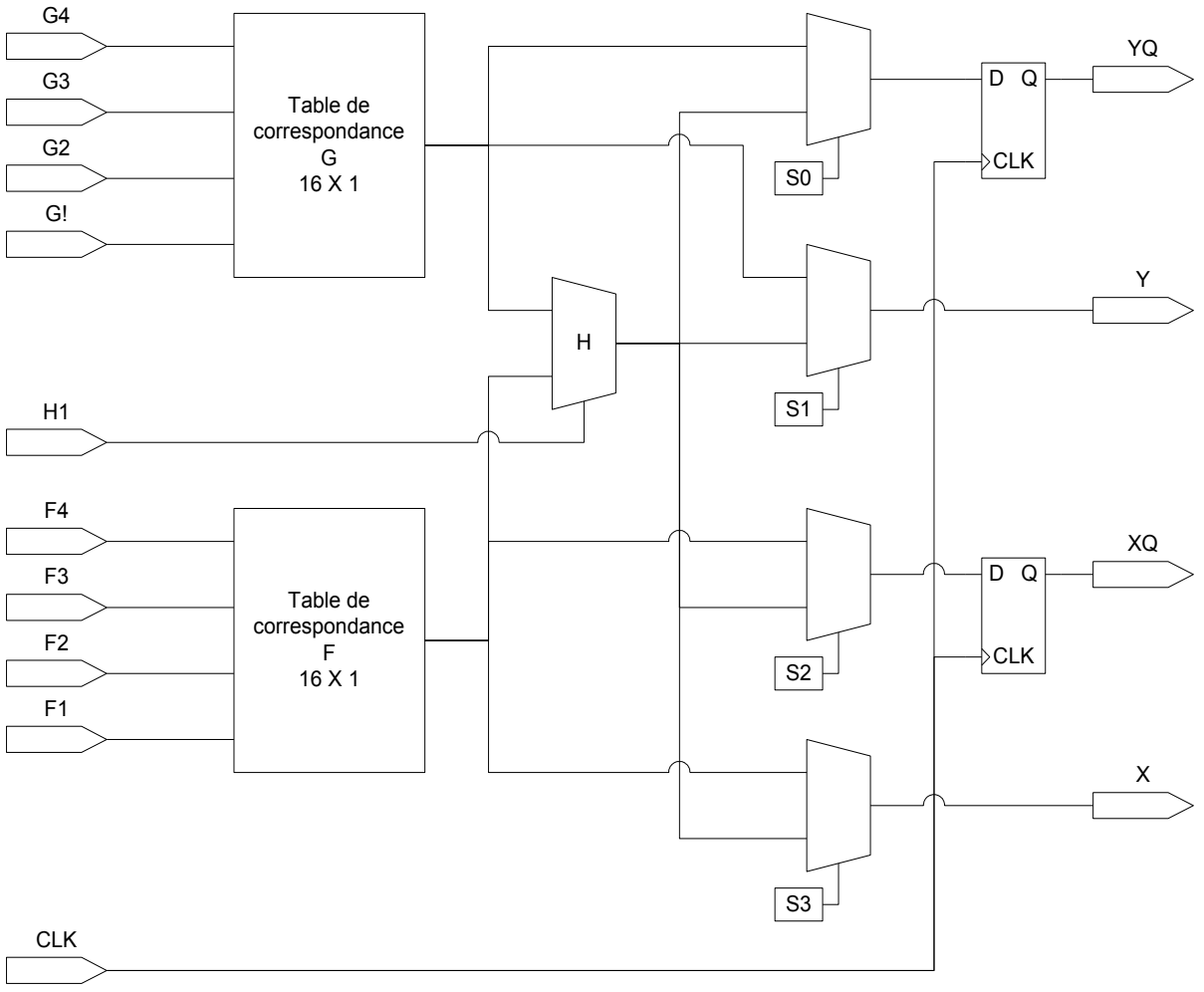
- Une tranche comprend:
  - Quatre tables de correspondance (Look-up Table – LUT) à 6 entrées, pouvant être programmées comme:
    - fonction logique
    - mémoire ROM
  - Quatre éléments à mémoire: bascule ou loquet.
  - Des multiplexeurs pour router les signaux.
  - Des portes logiques pour l'addition rapide.



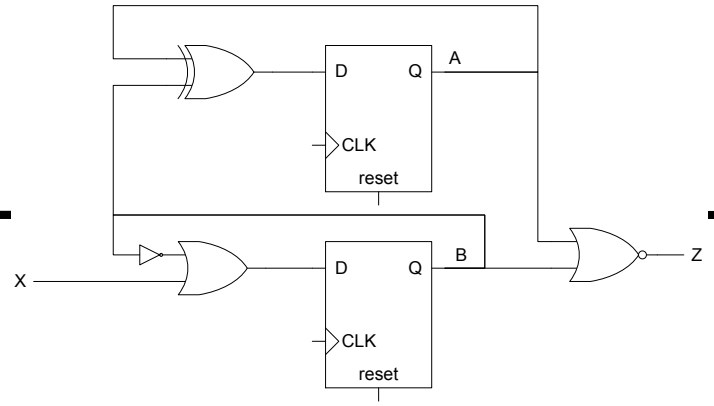
Xilinx inc., Virtex-5 FPGA User Guide (ug190 v. 5.4), March 2012

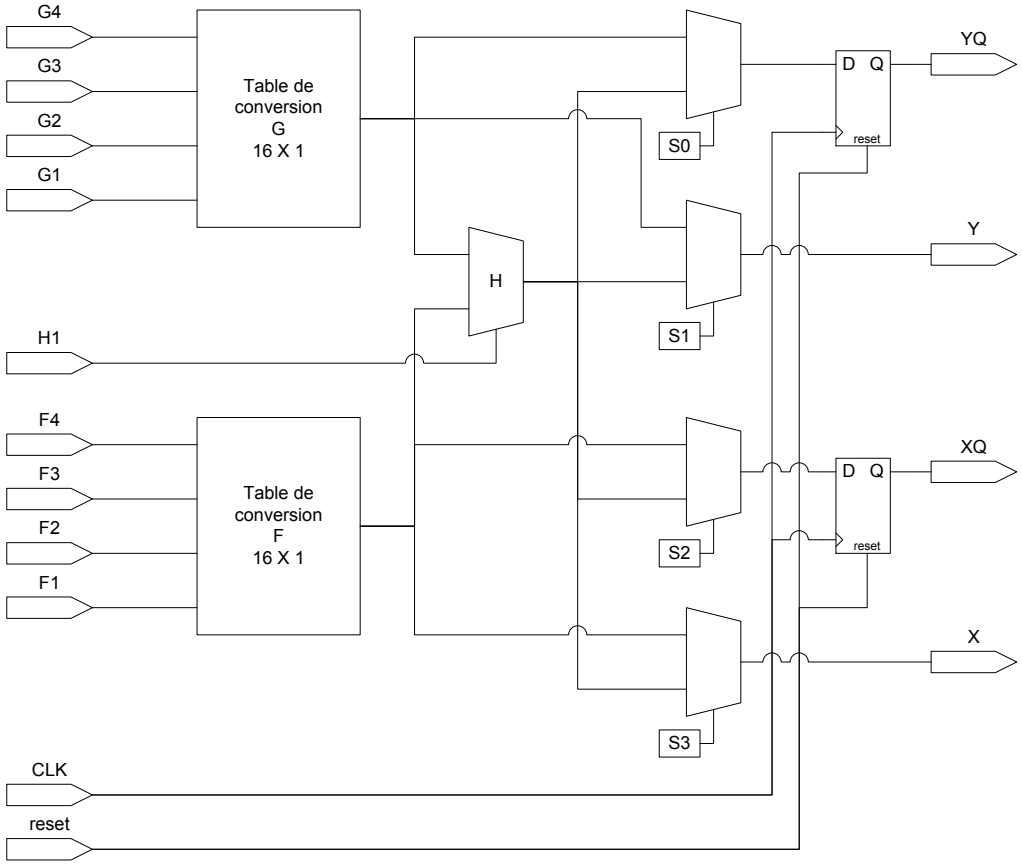
Modèle simplifié d'une tranche (~ une tranche du Virtex 2 Pro)

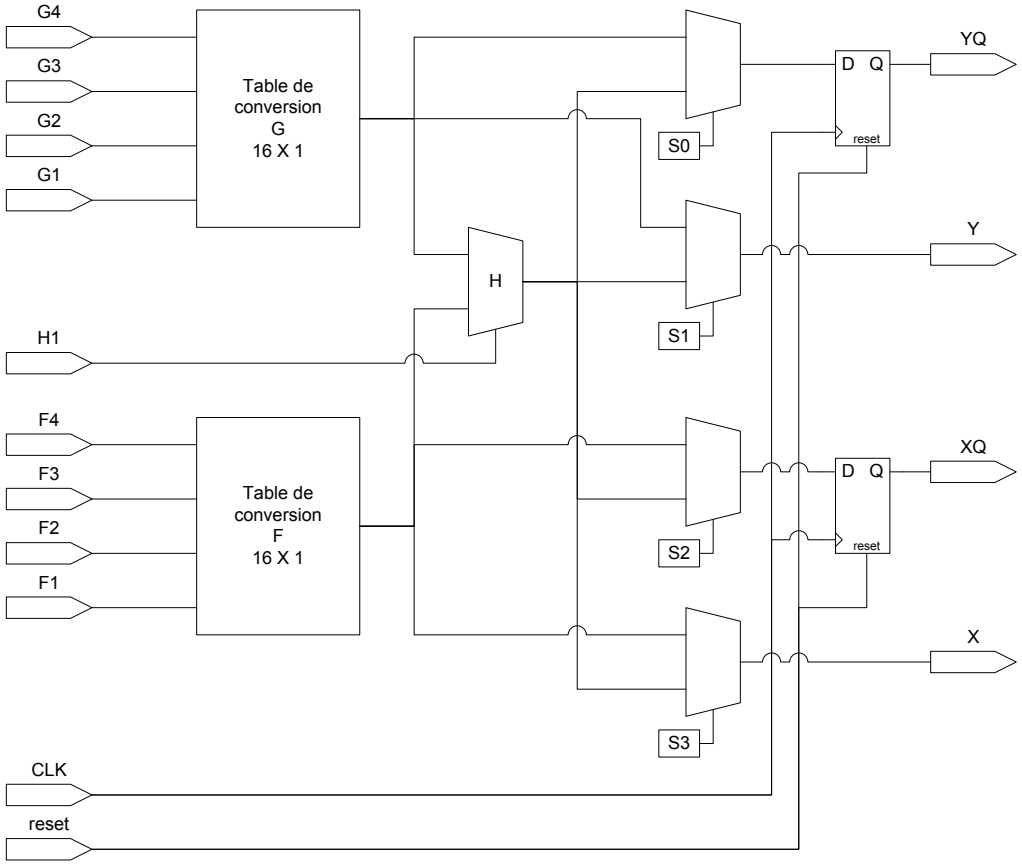
- Deux tables de correspondance à 4 entrées:
  - fonction logique
  - mémoire RAM
  - mémoire ROM
  - décalage
- Deux éléments à mémoire, bascule ou loquet.
- Des multiplexeurs pour router les signaux.



Exemple: Implémenter le circuit suivant sur un FPGA







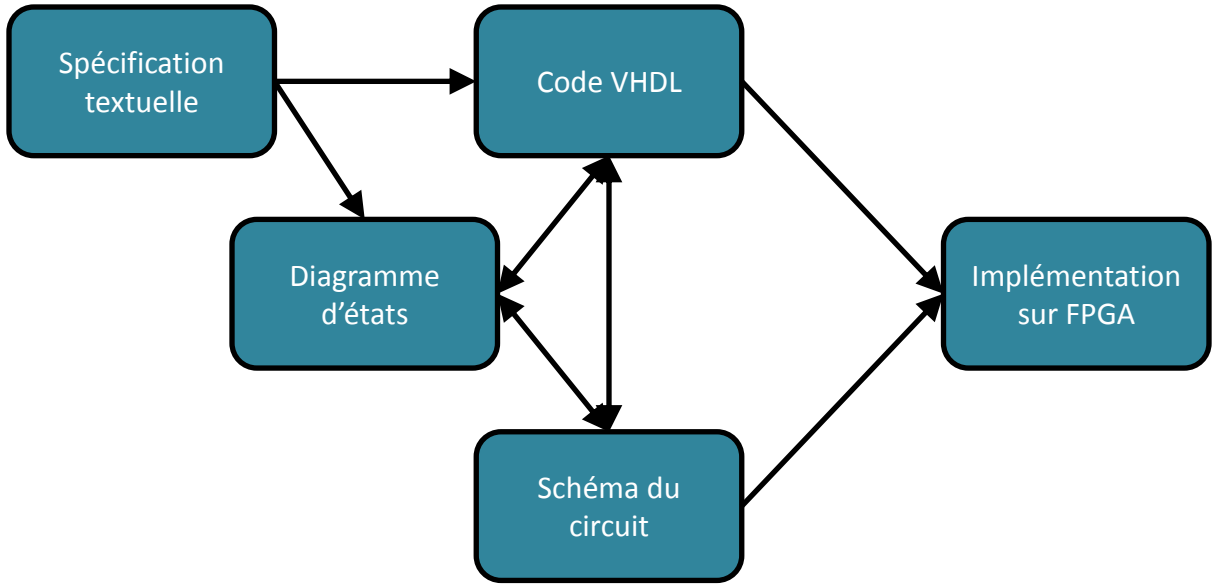
Trois bonnes pratiques à respecter avec les FPGA

1. Utiliser des bascules, éviter les loquets.
   - Les éléments à mémoire d'un FPGA peuvent implémenter une bascule ou un loquet: il n'y a pas de différence de coût.
   - Le désavantage du loquet est son mode transparent.
   - Pour le reste du cours, nous allons utiliser exclusivement des bascules.
2. Le signal d'horloge `CLK` est spécial:
   - En principe, on mène toutes les bascules avec le même signal d'horloge `CLK`.
   - On ne fait pas d'opérations logiques sur le signal `CLK`.
3. Le signal de réinitialisation `reset` est spécial:
   - Les circuits séquentiels doivent pouvoir être placés dans un état de départ connu.
   - En principe, on mène toutes les bascules avec le même signal de réinitialisation.
   - Certaines bascules peuvent être initialisées à '0', d'autres à '1'.
   - On ne fait pas d'opérations logiques sur le signal de réinitialisation.

On ne fait pas d'opérations logiques sur les signaux d'horloge et de réinitialisation!

Vous devriez maintenant être capable de …

- Expliquer comment on peut effectuer la synthèse d'un module séquentiel décrit en VHDL. (B2)
- Donner le schéma correspondant au code VHDL d'un circuit séquentiel. (B3)
- Effectuer la synthèse d'un module séquentiel décrit en VHDL et montrer son implémentation sur FPGA. (B3)
- Expliquer et appliquer les bonnes pratiques de la modélisation de circuits séquentiels pour FPGA. (B2, 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.