RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE
MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE
UNIVERSITÉ AKLI MOHAND OULHADJ BOUIRA

FACULTÉ DES SCIENCES ET DES SCIENCES APPLIQUEES
DEPARTEMENT DE GENIE ELECTRIQUE

Electronique Numériques Avancées : FPGA,VHDL

Cours présenté par :
M. Arezki FEKIK

Dans le cadre de la formation **Master 1 Electroniques des Systèmes Embarqués**

Introduction générale

- **VHDL** est l'acronyme de **VHSIC HDL** (***V**ery **H**igh **S**peed **I**ntegrated **C**ircuit **H**ardware **D**escription **L**anguage*), c'est un langage de description matérielle qui a été créé dans les années 1980 à la demande du département de la défense américaine (**DOD**).
- La première version du **VHDL** accessible au public a été publiée en 1985, et a fait l'objet d'une norme internationale en **1986** par l'institut des ingénieurs électriciens et électroniciens (**IEEE**).
- De nos jours, le langage **VHDL** devient un outil indispensable pour la conception des systèmes électroniques intégrés, il est proposé par la grande majorité des sociétés de développement et la commercialisation d'**ASIC** et d'**FPGA** telle que la société américaine **Xilinx**.
- Avec un langage de description matérielle et un **FPGA** (***F**ield **P**rogrammable **G**ate **A**rray*), un concepteur peut développer rapidement et simuler un circuit numérique sophistiqué, de l'implémenter sur une carte de prototypage, et de vérifier son fonctionnement.

PLAN DE COURS

- Chapitre. I: Concepts de base du langage VHDL
- Chapitre. II: Objets et types de données
- Chapitre. III: Différentes descriptions d'une architecture
- Chapitre. IV: Modélisation des circuits séquentiels
- Chapitre. V: Exemple De FPGA : La Famille Spartan-6

Modélisation des circuit séquentiels par VHDL

Chapitre. IV: La modélisation des circuits séquentiels

- **Introduction :**

- ✓ Les bascules sont les éléments de base des circuits séquentiels.
- ✓ Dans ce chapitre, nous allons d'abord décrire en VHDL ces bascules, puis on modélisera des circuits séquentiels complexes.
- ✓ Nous rappelons que dans les circuits séquentiels, les sorties ne dépendent pas que des entrées, mais aussi de la valeur de l'état de sortie précédente.

Chapitre. IV: La modélisation des circuits séquentiels

- **Les opérations synchrones et asynchrones**
  Les circuits séquentiels peuvent être classés en deux catégories :
  - Les circuits séquentiels synchrones.
  - Les circuits séquentiels asynchrones.

En VHDL, la fonction des fronts d'une horloge (nommée par exemple ***CLK***) est :

- Pour les fronts montants :
  CLK'event and CLK='1' ou bien : **rising_edge(CLK);**

- - Pour les fronts descendants :
  CLK'event and CLK = '0' ou bien **falling_edge(CLK);**

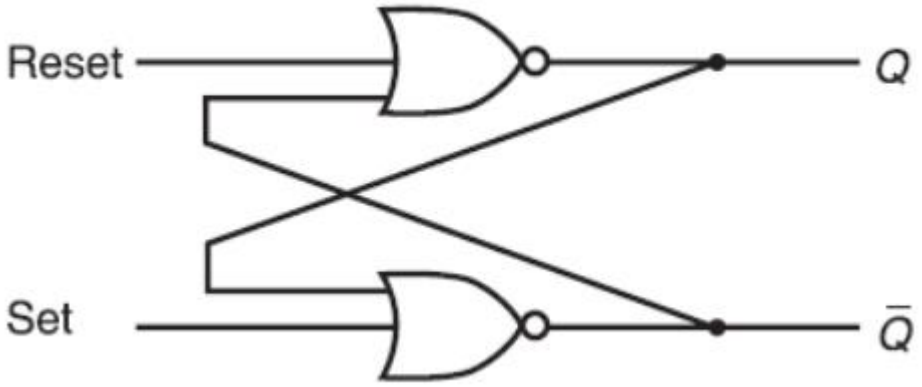
Chapitre. IV: La modélisation des circuits séquentiels

- **Les Bascules:**
- Les bascules sont les éléments de base de la logique séquentielle, un peu comme l'étaient les portes logiques en logique combinatoire. Nous verrons par la suite qu'elles permettent de réaliser de nombreux systèmes (compteurs, registres, mémoires …) d'où leur importance.

La bascule R-S:

C'est une bascule à deux entrées R (**RESET**), S (**SET**) et deux sorties complémentaires **Q** et $\overline{\mathbf{Q}}$

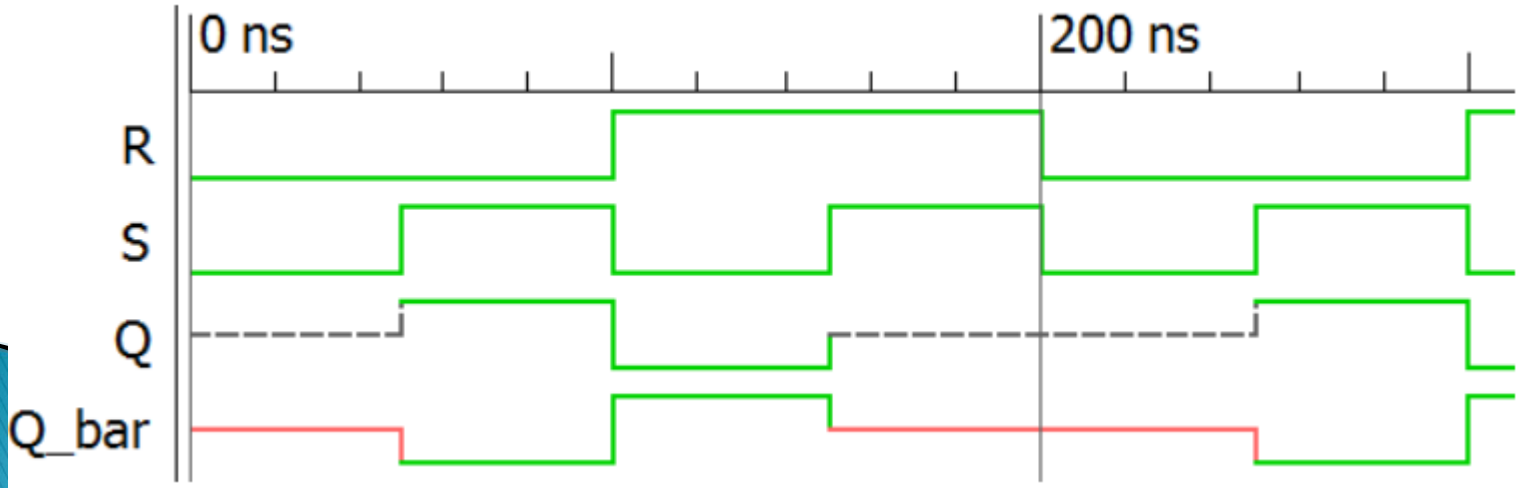
R	S	Q(t)	$\overline{Q}(t+1)$
0	0	Q	$\overline{Q}$
0	1	1	0
1	0	0	1
1	1	ETAT IND	ETAT IND



Bascule R-S formé avec des portes logiques NOR

Chapitre. IV: La modélisation des circuits séquentiels

```
 1  library ieee;
 2  use ieee.std_logic_1164.all;
 3
 4  entity RS is
 5     port(R, S : in std_logic;
 6          Q, Q_bar : out std_logic);
 7  end RS;
 8
 9  architecture basc of RS is
10     signal SIG : std_logic;
11     signal RS : std_logic_vector(1 downto 0);
12  begin
13     RS(1) <= R;
14     RS(0) <= S;
15     with RS select
16        SIG <= SIG when "00",
17               '1' when "01",
18               '0' when "10",
19               '-' when others;
20     Q <= SIG;
21     Q_bar <= not SIG;
22  end basc;
```



Chapitre. IV: La modélisation des circuits séquentiels

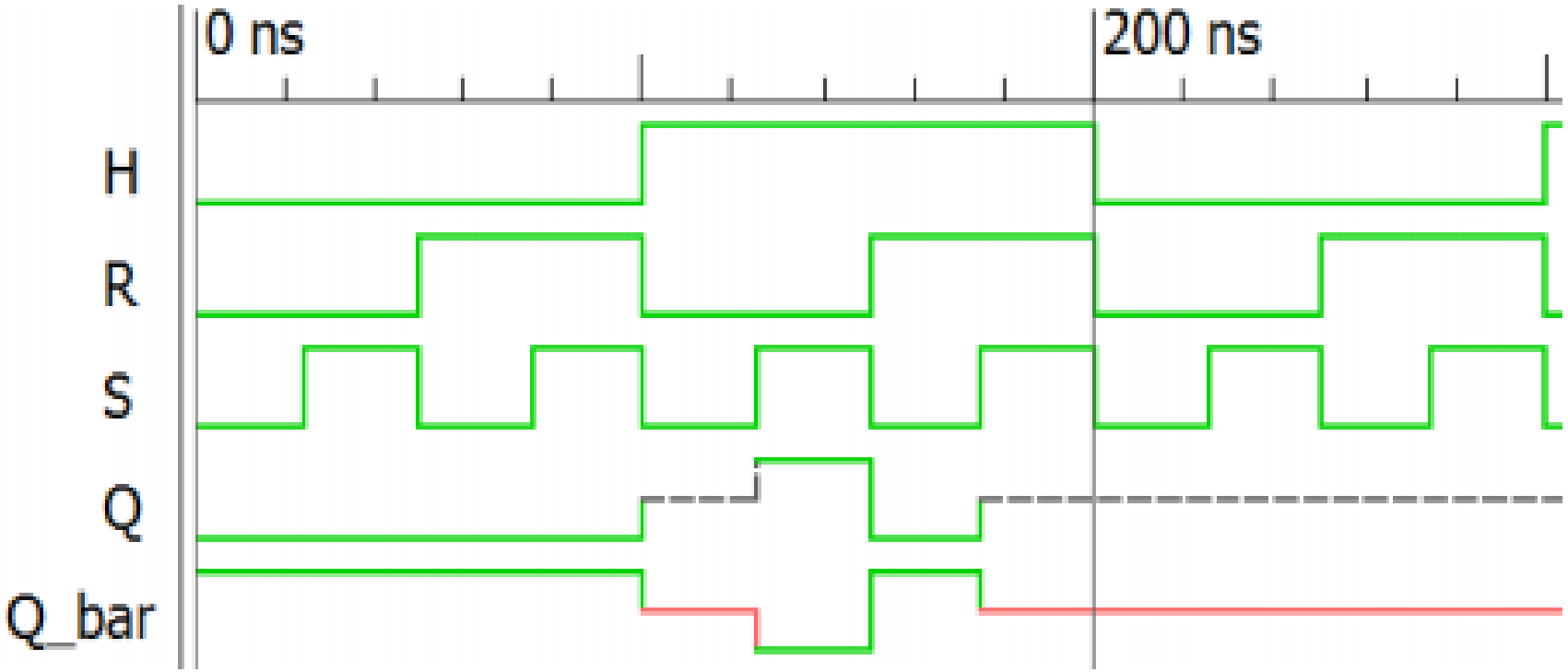
- **Bascule RSH:**
- C'est une bascule R-S avec une troisième entrée qui sert d'activation

H	R	S	Q(t)	$\overline{Q}$ (t+1)
0	0	0	Q	$\overline{Q}$
0	0	1	Q	$\overline{Q}$
0	1	0	Q	$\overline{Q}$
0	1	1	Q	$\overline{Q}$
1	0	0	Q	$\overline{Q}$
1	0	1	1	0
1	1	0	0	1
1	1	1	*Etat indéterminé*	*Etat indéterminé*

```
 1  library ieee;
 2  use ieee.std_logic_1164.all;
 3
 4  entity RSH is
 5     port(R, S, H : in std_logic;
 6          Q, Q_bar : out std_logic);
 7  end RSH;
 8
 9  architecture basc of RSH is
10        signal SIG : std_logic := '0';
11        signal RS : std_logic_vector (1 downto 0);
12  begin
13     RS(1) <= R;
14     RS(0) <= S;
15     process(RS, SIG, H)
16     begin
17        if (H = '1') then
18           case RS is
19              when "00" => SIG <= SIG;
20              when "01" => SIG <= '1';
21              when "10" => SIG <= '0';
22              when others => SIG <= '-';
23           end case;
24        else
25           SIG <= SIG;
26        end if;
27        Q <= SIG;
28        Q_bar <= not SIG;
29     end process;
30  end basc;
```

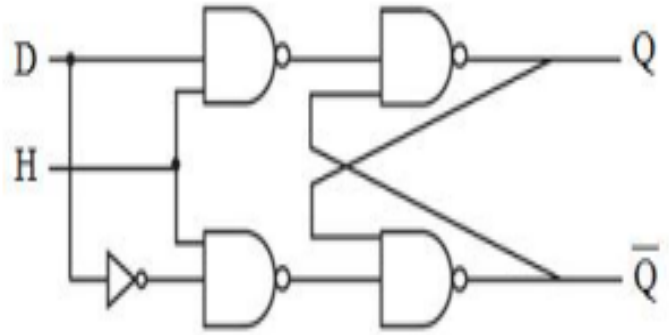
Chapitre. IV: La modélisation des circuits séquentiels

Résultat de la simulation du bascule RSH



Chapitre. IV: La modélisation des circuits séquentiels

- **Bascule D :**
  Un **bascule R-S** peut être converti en un autre **bascule**, connu sous le nom de **bascule D**, en ajoutant un inverseur

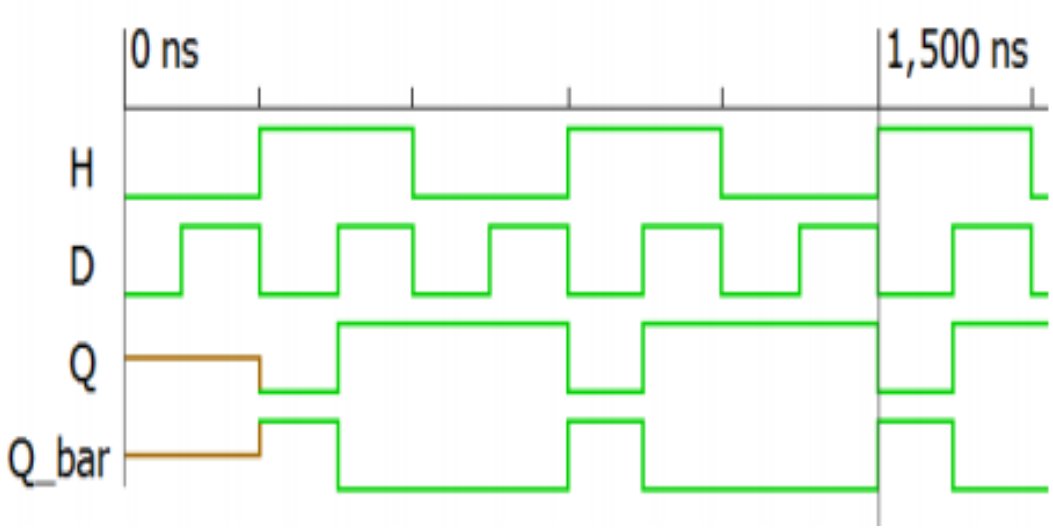


H	D	Q(t+1)	$\overline{Q}$ (t+1)
0	-	Q	$\overline{Q}$
1	0	0	1
1	1	1	0

Chapitre. IV: La modélisation des circuits séquentiels

La description en VHDL du **bascule D** est donnée par le **listing** suivant

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3
4  entity D is
5     port ( D, H : in std_logic;
6            Q, Q_bar : out std_logic);
7  end D;
8
9  architecture LATCH of D is
10    signal SIG : std_logic;
11 begin
12    process(H, D)
13    begin
14       if (H = '1') then
15          SIG <= D;
16       else
17          SIG <= SIG;
18       end if;
19    end process;
20    Q <= SIG;
21    Q_bar <= not SIG;
22 end LATCH;
```



Chapitre. IV: La modélisation des circuits séquentiels

- **Bascule J-K :**
  la bascule J-K a les fonctions similaires à la bascule R-S, avec l'entrée ***J*** qui est équivalente à l'entrée ***S*** (mise à '**1**'), et l'entrée ***K*** qui est équivalente à l'entrée ***R*** (remise à '0'). Contrairement au bascule R-S, la bascule J-K ne possède pas des combinaisons d'entrée qui donneront un état indéterminé.
  Prenons comme exemple une bascule J-K actif sur front descendant.

  Si les entrées **J** et **K** sont à **'1'**, la sortie prend l'inverse de l'état précédent.

Chapitre. IV: La modélisation des circuits séquentiels

La description en VHDL de la bascule **J-K** est donnée par le **listing Suivant**

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3
4  entity JK is
5     port ( J, K, Clk, Set, Reset : in std_logic;
6            Q, Q_bar : out std_logic);
7  end JK;
8
9  architecture FFJK of JK is
10    signal SIG : std_logic;
11    signal JK : std_logic_vector (1 downto 0);
12 begin
13    JK(1) <= J;
14    JK(0) <= K;
```

Chapitre. IV: La modélisation des circuits séquentiels

```
15      process(Clk, Set, Reset)
16      begin
17          if (Reset = '0' and Set = '0') then
18              SIG <= '0';
19          elsif (Reset = '0' and Set = '1') then
20              Sig <= '0';
21          elsif (Reset = '1' and Set = '0') then
22              Sig <= '1';
23          elsif (Reset = '1' and Set = '1') then
24              if (Clk'event and Clk = '0') then
25                  case JK is
26                      when "00" => SIG <= SIG;
27                      when "01" => SIG <= '0';
28                      when "10" => SIG <= '1';
29                      when "11" => SIG <= not SIG;
30                      when others => SIG <= '-';
31                  end case;
32              else
33                  SIG <= SIG;
34              end if;
35          end if;
36      end process;
37      Q <= SIG;
38      Q_bar <= not SIG;
39  end FFJK;
```

Chapitre. IV: La modélisation des circuits séquentiels

- **Les Compteurs :**

  Ils sont très utilisés dans les descriptions **VHDL**. L'écriture d'un compteur peut être très simple comme très compliquée. Ils font appels aux ***process***.

```
Library ieee;
Use ieee.std_logic_1164.all;
Use ieee.numeric_std.all;
Use ieee.std_logic_unsigned.all;

entity CMP4BITS is
PORT (
      CLOCK : in std_logic;
      Q     : inout std_logic_vector(3 downto 0));
end CMP4BITS;

architecture DESCRIPTION of CMP4BITS is
begin
  process (CLOCK)
  begin
    if (CLOCK ='1' and CLOCK'event) then
      Q <= Q + 1;
    end if;
  end process;
end DESCRIPTION;
```