POLYTECHNIQUE
MONTRÉAL

INF1500
Logique des systèmes numériques

Laboratoire 5

Soumis par:
Mohamad Awad - 2034584
Lourhmati Oussama - 2081643
Groupe 05

Le 10 décembre 2020

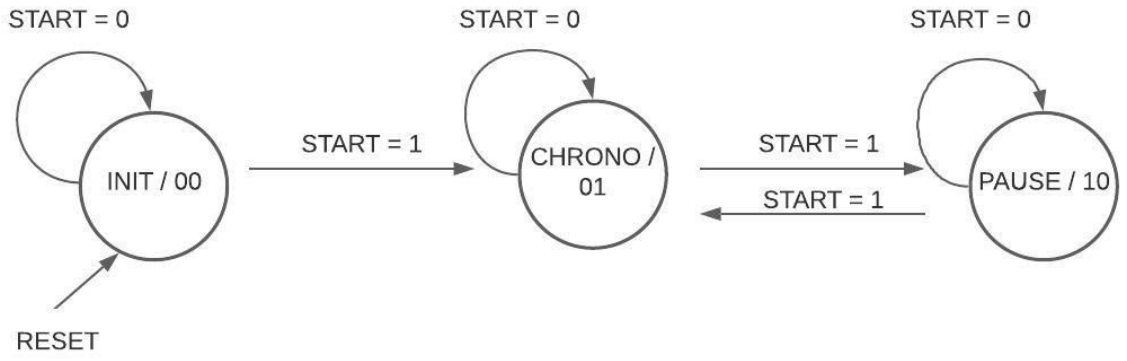
Introduction

Dans ce laboratoire, nous avons conçu deux machines à états en code VHDL, et nous les avons combinées pour obtenir un chronomètre comme système final. Nous décrivons dans ce qui suit, les deux machines, leurs diagrammes d'états et par la suite, dans la vérification du système, nous avons vérifié les fonctionnements de chaque module ainsi que le bon fonctionnement du système final.

1- Description du système

Chronomètre

- État INIT (initial): 00 (COUNT est à 0).
- État CHRONO (en cours): 01 (COUNT incrémente périodiquement).
- État PAUSE (en pause): 10 (COUNT est constant).



Le diagramme ci-dessus indique que le chronomètre est une machine à états de type Moore. Les entrées du chronomètre sont trois signaux :

1. Une horloge CLK.
2. Un signal START qui est synchronisé, il est vérifié à chaque front montant de l'horloge, et qui est généré par un générateur de pulsion.
3. Un signal Reset qui est asynchrone et qui permet de faire une réinitialisation du système à tout moment. Il est généré par un click sur un bouton RESET_S du système.

Les sorties du chronomètre sont :

1. State, un std_logic_vector(1 downto 0) qui indique l'état courant.

2. Le compte du chronomètre, std_logic_vector (31 downto 0).

Comme on peut le voir dans le diagramme:

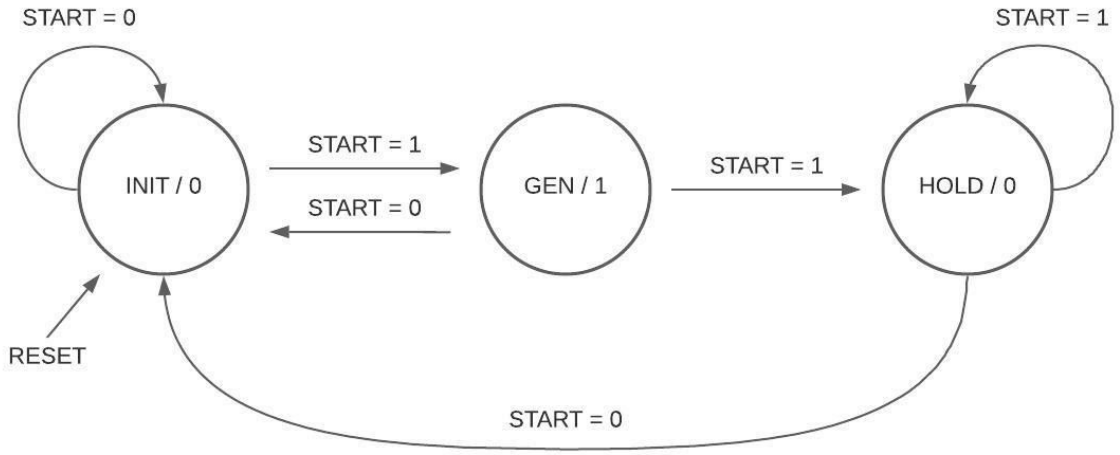
- Un click sur le bouton start , déclenche le chronomètre (INIT → CHRONO) .
- Un deuxième click sur le bouton start va pauser le compte. (CHRONO → PAUSE).
- Un click sur start lorsque le chronomètre est en pause va continuer le compte. (PAUSE → CHRONO).
- Un Reset va réinitialiser le système à tout moment. (→ INIT).

Ci dessous, est le code VHDL du chronomètre, on fait référence au fichier (CHRONO.vhd) dans le projet vivado.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.numeric_std.all;

entity CHRONO_VHDL is
    port(
        CLK: in STD_LOGIC;
        RESET: in STD_LOGIC;
        START: in STD_LOGIC;
        COUNT: out STD_LOGIC_VECTOR(31 downto 0);
        STATE: out STD_LOGIC_VECTOR(1 downto 0)
    );
end CHRONO_VHDL;


architecture CHRONOBehavioral of CHRONO_VHDL is
    type STATE_TYPE is (INIT, CHRONO ,PAUSE);
    signal CURRENT_STATE, NEXT_STATE : STATE_TYPE;
    signal TMP : STD_LOGIC_VECTOR(31 downto 0) := (others => '0');


begin
    -- Partie séquentielle
  process (CLK, RESET)
  begin
  if (RESET ='1') then
  CURRENT_STATE <= INIT;
  elsif (CLK'event and CLK ='1') then
```

```
CURRENT_STATE <= NEXT_STATE;
end if;
end process;

     -- Partie combinatoire
process (CURRENT_STATE, START, CLK)
   begin
    case CURRENT_STATE is

      when INIT =>
          if (reset ='1') then next_state<= init; end if;
          COUNT <= (others => '0');
          TMP <= X"00000001";
          STATE <= "00";
          if (START = '1' and RESET = '0') then
             NEXT_STATE <= CHRONO;
          end if;

      when CHRONO =>
          if (reset ='1') then next_state<= init; end if;
          COUNT<=TMP;
          if (CLK'event and CLK= '0') then
              TMP <= STD_LOGIC_VECTOR(UNSIGNED(TMP)+1);
          end if;
          STATE <= "01";
          if (START = '1' and CLK'event and CLK='0' and RESET ='0' ) then
              NEXT_STATE <= PAUSE;
          end if;

      when PAUSE =>
          if (reset ='1') then next_state<= init; end if;
          COUNT<=STD_LOGIC_VECTOR(UNSIGNED(TMP)-1);
          STATE <= "10";
          if (START = '1' and CLK'event and CLK ='0' and RESET ='0') then
              NEXT_STATE <= CHRONO;
          end if;

    end case;  end process; end CHRONOBehavioral;
```

Générateur de pulsion



Le diagramme ci-dessus indique que le générateur de pulsion est une machine de type Moore. La sortie de la machine va dépendre de l'état courant.

Cette machine permet de générer une pulsion unique qui commence au prochain front montant dès qu'on presse le bouton start, et va durer une période entière d'horloge.Ceci permet de donner une bonne forme et synchronisation aux signaux entrés par le click du bouton et qui sont généralement pas en synchronisation avec le clk et de longue durée par rapport à la période de celle-ci.

Les entrées :

1. CLK , l'horloge, qui est la même pour le chronomètre.
2. START, qui est synchrone et vérifié chaque front montant, il est généré par le click sur le bouton du système
3. Le signal RESET qui est le même dans le chronomètre et qui réinitialise le générateur et tout le système.

La sortie : S , qui est la pulsion unique générée une seule fois après chaque click sur le bouton start et qui joue le rôle du signal start du chronomètre.

Comme le montre le diagramme :

- Lorsque le générateur est sur son état INIT, un click sur le bouton start permet d'aller à l'état GEN.

- L'état GEN dure exactement une période de la clock, parce que dans les deux cas si Start est à 1 ou 0 , la machine passe à un autre état et ce, au prochain front montant de l'horloge. Donc la pulsion va être générée comme il le faut.
- Si Start est à 1 ça veut dire que le bouton est toujours pressé (press and hold). donc la machine va sur l'état HOLD, S est donc 0.
- Une fois Start est à 0, ça veut dire que le bouton n'est plus pressé. Donc la machine revient à son état initial , elle attend le prochain clic du bouton.
- À tout moment , Reset permet de réinitialiser la machine.

Ci dessous, est le code VHDL du générateur de pulsion, on fait référence au fichier (PULSE_GEN.vhd) dans le projet vivado.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity PULSE_GEN is
    port(
    CLK , START , RESET :in std_logic;
    S: out std_logic
);
end PULSE_GEN;

architecture PULSE_GENBehavioral of PULSE_GEN is
    type STATE_TYPE is (INIT, GEN , HOLD);
    signal CURRENT_STATE, NEXT_STATE : STATE_TYPE;

begin

 process (CLK, RESET)
     begin
      if(RESET ='1') then
           CURRENT_STATE <= INIT;
      elsif (CLK'event and CLK ='1') then
           CURRENT_STATE <= NEXT_STATE;
      end if;
 end process;


 process (CURRENT_STATE,START)
     begin
        case CURRENT_STATE is

            when INIT =>
```

```
            S<='0';
            if (START = '1' and RESET ='0') then
               NEXT_STATE <= GEN ;
            end if;

            when GEN =>
            S<='1';
            if (RESET ='1') then
               NEXT_STATE <= INIT;
            else
               if START='0' then
                   NEXT_STATE <= INIT;
               else
                   NEXT_STATE <= HOLD;
               end if;
            end if;

            when HOLD =>
            if (RESET ='1') then NEXT_STATE <= INIT;
            else
                S<='0';
                if START ='0' then
                    NEXT_STATE <= INIT;
                end if;
            end if;

        end case ;
  end process;

end PULSE_GENBehavioral;
```

System

On combine les deux machines dans une description VHDL structurale pour obtenir le système final . On fait référence au fichier (System.vhd) dans le projet vivado .

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity SYSTEM is
     port(
        CLK_S, START_S, RESET_S: in STD_LOGIC;
```

```
        COUNT_S: out STD_LOGIC_VECTOR(31 downto 0);
        STATE_S: out STD_LOGIC_VECTOR(1 downto 0)
    );
end SYSTEM;

architecture SYSTEMBehavioral of SYSTEM is

    component PULSE_GEN is
    port(
        CLK , START , RESET :in std_logic;
        S: out std_logic
    );
    end component;

    component CHRONO_VHDL is
    port(
        CLK, START, RESET: in STD_LOGIC;
        COUNT: out STD_LOGIC_VECTOR(31 downto 0);
        STATE: out STD_LOGIC_VECTOR(1 downto 0)
    );
    end component;

    signal TMP_COUT0: STD_LOGIC;

begin
    U0 :
    PULSE_GEN port map(
    CLK => CLK_S, START => START_S, RESET => RESET_S, S => TMP_COUT0
    );

    U1:
    CHRONO_VHDL port map(
    START => TMP_COUT0 , CLK => CLK_S, RESET => RESET_S, COUNT => COUNT_S, STATE =>
STATE_S
    );

end SYSTEMBehavioral;
```

2- Vérification du système

Pour la vérification du système de ce laboratoire .

- Nous n'utilisons de *force clock* ,sauf pour l'horloge CLK ,qui a une période de 10 ns.

- Les simulations fonctionnent en appliquant des séries de *force constant* durant principalement de 5 ns à 10 ns (ces *force constant* représentent un click sur un bouton)
    - surtout pour la simulation du CHRONO puisqu'il faut tenir compte de chaque front montant et descendant de l'horloge qui a une période de 10 ns (100 MHz).
- Pour le PULSE_GEN et le système global, les *force constant* pourraient durer plus longtemps. (un press and hold sur le bouton qui ne cause pas de problème pour pulse_gen)
- Pour le bouton RESET, qui est **asynchrone** par rapport à l'horloge, la longueur et la forme du signal n'est pas importante.

En effet, un *force clock* pour start et reset , veut dire plusieurs click des boutons du système sur un grand intervalle de temps, ce qui est inutile dans l'analyse du comportement.

Module PULSE_GEN

Pour la vérification du module PULSE_GEN, nous procédons par quelques simulations pour différentes situations: on clique le bouton start plusieurs fois et de différentes manières pour vérifier que les pulsions ont la même forme. et on clique reset aussi dans plusieurs situation pour s'assurer d'une réinitialisation complète.

- Dans la simulation de la Figure 2.0, il y a génération d'un signal unique et répond aux 2 problèmes décrits dans l'énoncé du laboratoire, et qui permet la détection de n'importe quel signal START (même s'il ne coïncide avec aucun front du clk)
- Dans la Figure 2.1, quel que soit l'état de la machine, RESET remet la machine à son état initial : current state et next_state son à INIT, il peut couper même la pulsion avant d'être générée complètement
- Les deux figures valident les états de la machine décrits au début du rapport et convient au diagramme d'états.

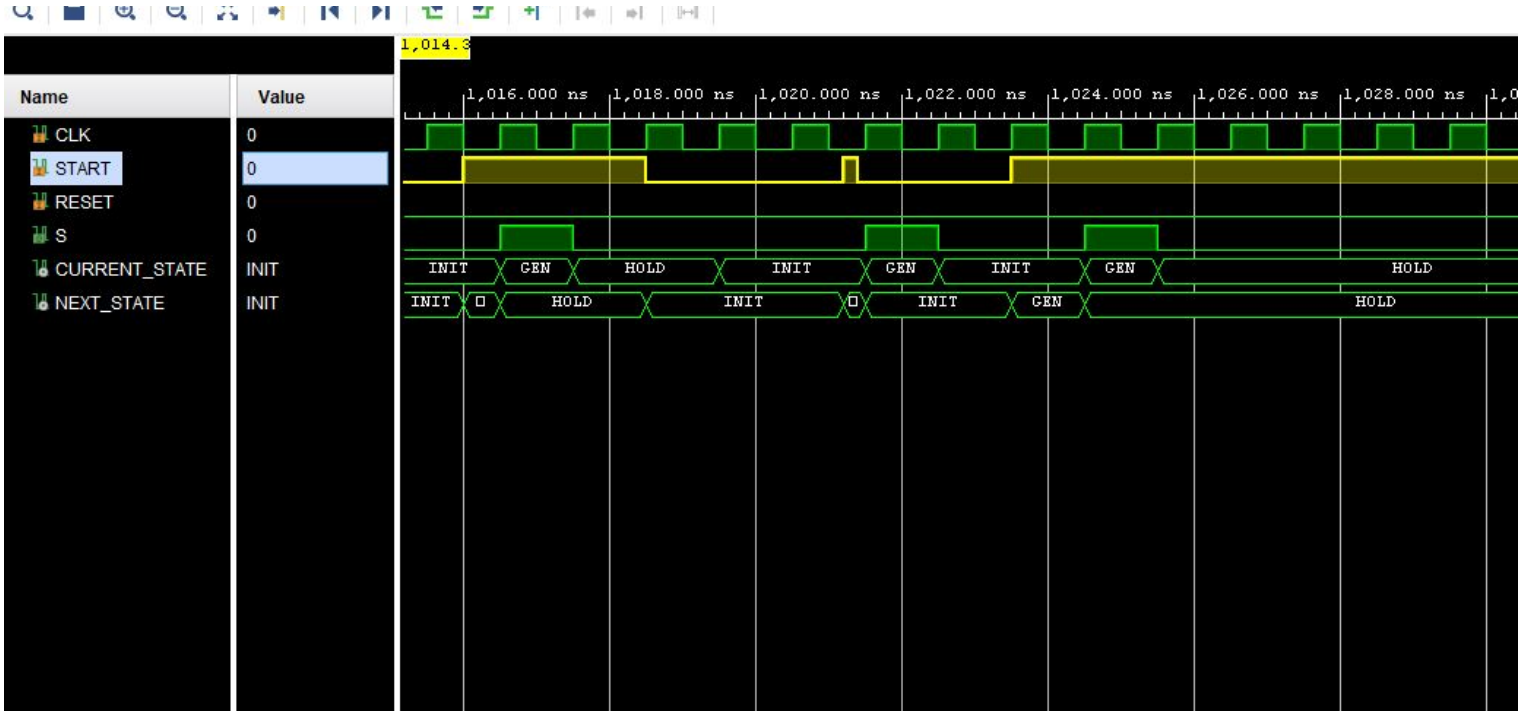


Figure 2.0 : PULSE_GEN

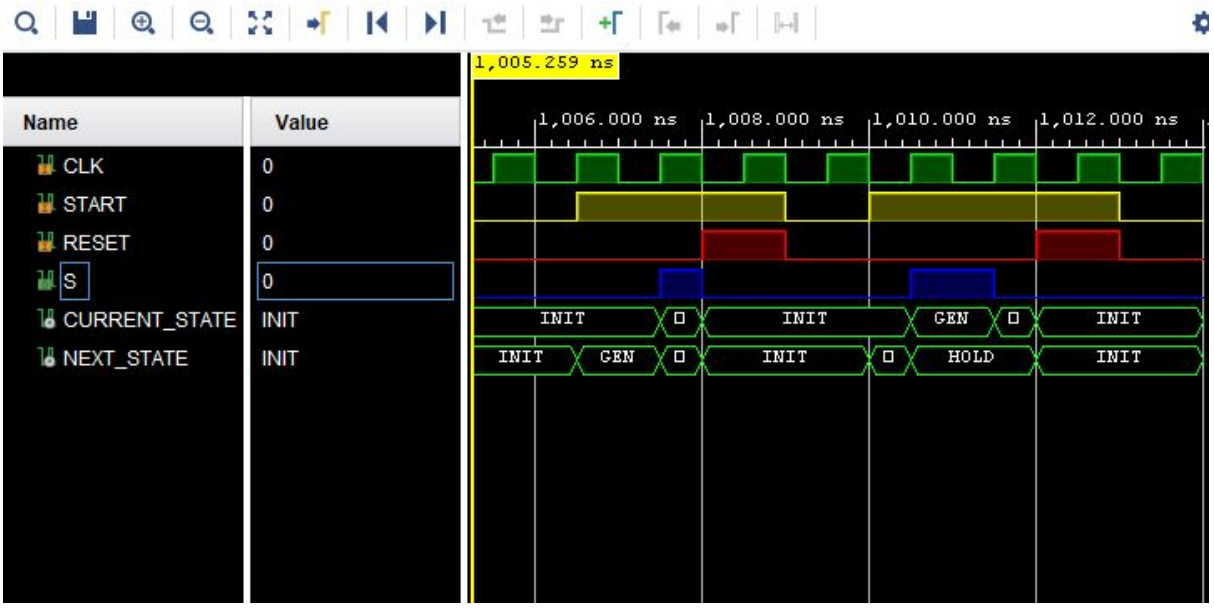


Figure 2.1 : RESET dans PULSE_GEN

Module CHRONO

Pour la vérification du CHRONO, le signal START dans la simulation doit être identique au signal généré par PULSE_GEN. Sinon, on pourrait avoir des problèmes et des ambiguïtés, avec des transitions entre les états inattendues puisque le chronomètre est conçu **juste** pour recevoir des signaux de forme unique: des signaux générés par PULSE_GEN.

- On envoie le signal start trois fois sans interruption par reset pour vérifier le bon comportement des états, puis on envoie des signaux reset à des endroits différents pour vérifier une réinitialisation complète
- Pour le COUNT
    - une fois dans l'état CHRONO, il prend la valeur de TMP. Puisque la valeur de TMP incrémente à chaque front descendant, la valeur de COUNT incrémente alors au prochain front montant.
    - Dès qu'on est à l'état PAUSE, COUNT prend la valeur TMP - 1, pour s'arrêter à sa dernière valeur, puisque TMP est toujours en avance de 1 par rapport à COUNT.
    - Revenu dans l'état CHRONO, COUNT prendra directement la valeur de TMP qui est en avance par rapport à sa valeur actuelle de 1 (le COUNT continue instantanément quand on est dans l'état CHRONO)
- Dans l'état INIT, TMP vaut 1. De cette manière, lorsqu'on est dans l'état CHRONO, le COUNT prend 1, et débute immédiatement.
- **Dans la Figure 2.2**, nous passons de l'état INIT à CHRONO (START =1), et par la suite de CHRONO à PAUSE lorsque le bouton START est à 1 une deuxième fois.
- **Dans la Figure 2.3,** nous passons de l'état PAUSE vers l'état CHRONO lorsque l'utilisateur appuie encore une fois sur le bouton START (START = 1).
- **Dans la Figure 2.4**, Le bouton RESET, à tout moment , quelque soit la longueur du signal et même si le bouton start est cliqué, va mettre l'état de la machine à INIT ainsi que son prochain état d'une façon **asynchrone.**

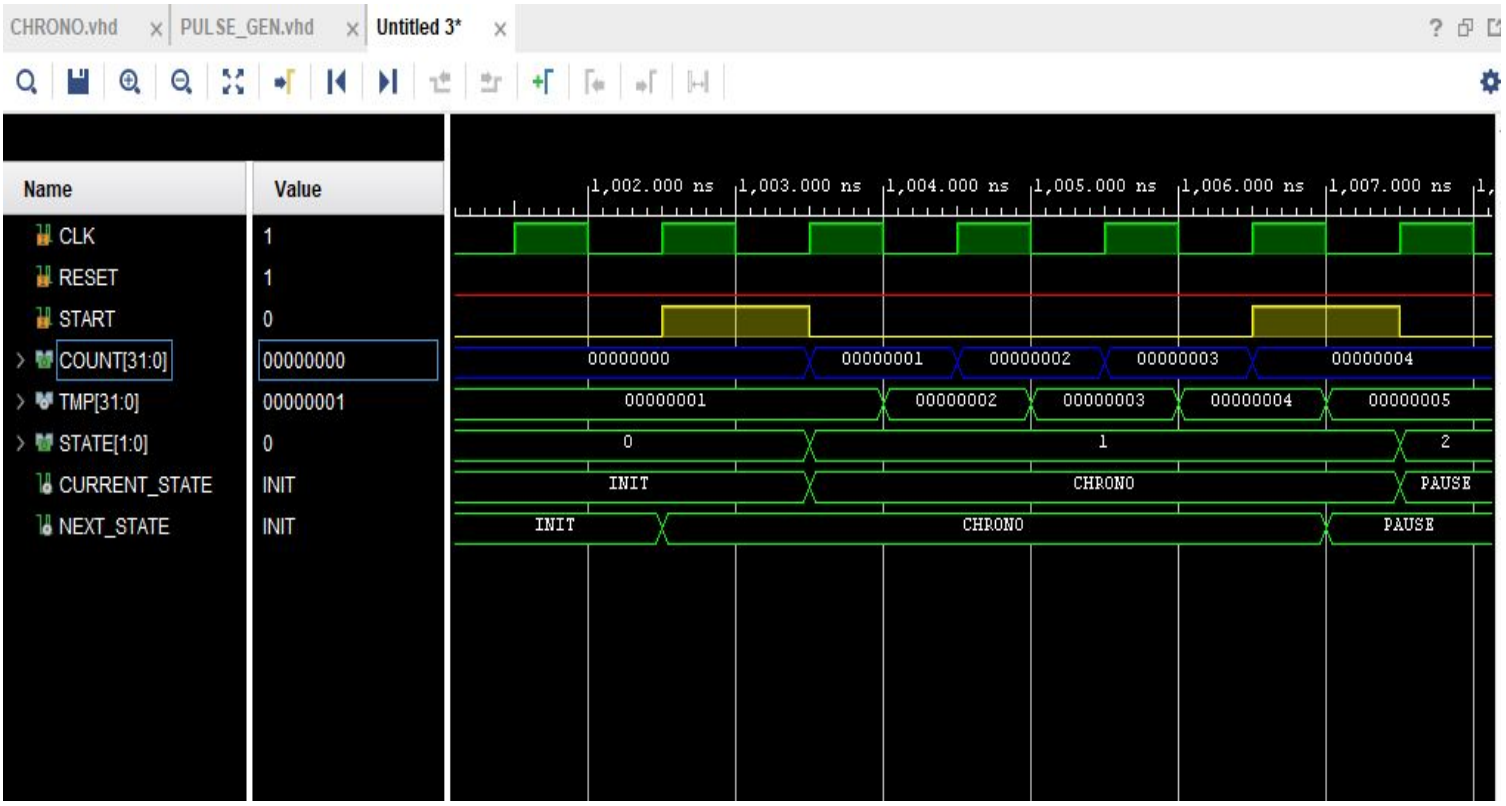


Figure 2.2: INIT → CHRONO → PAUSE

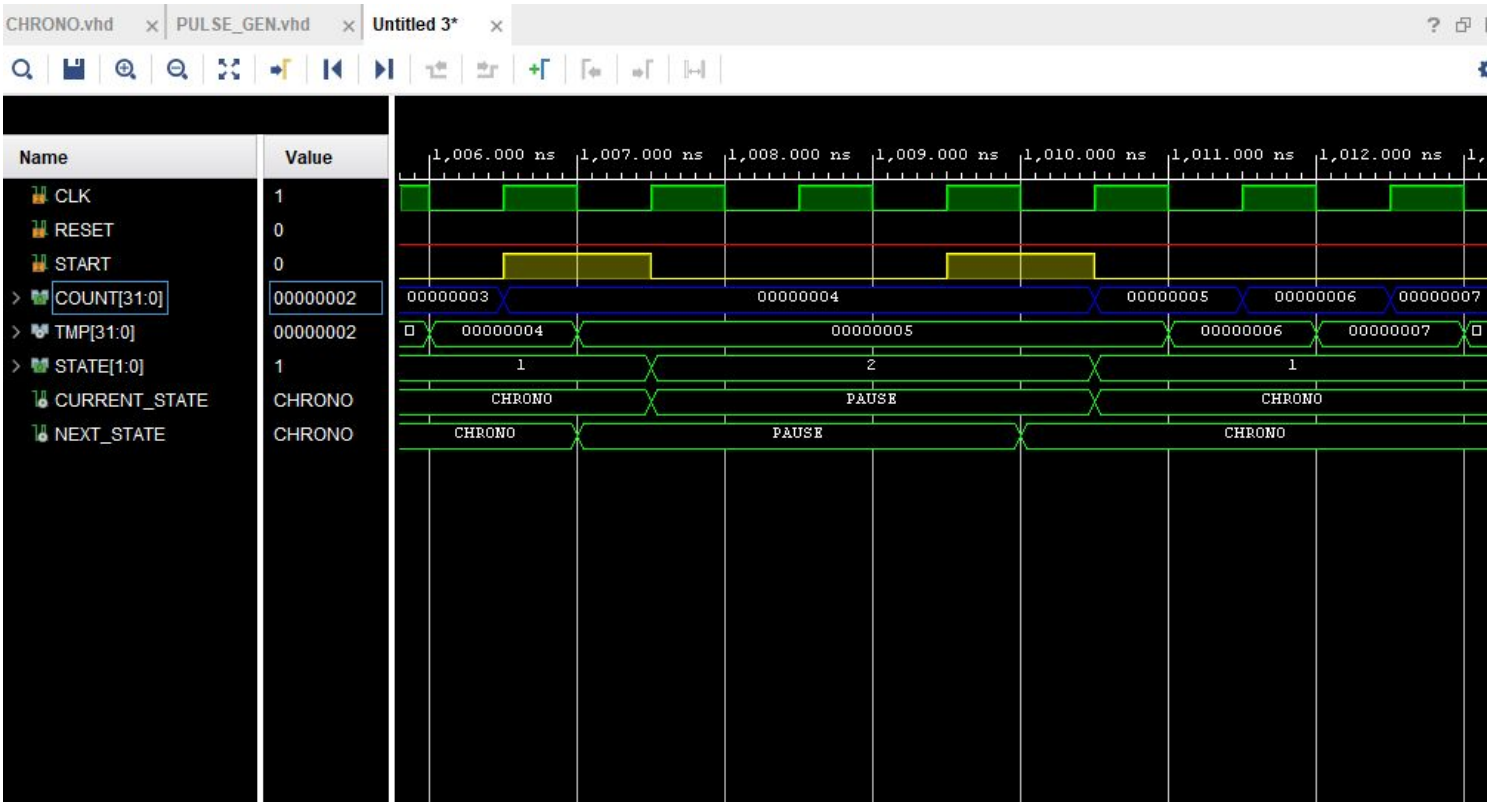


Figure 2.3: PAUSE → CHRONO

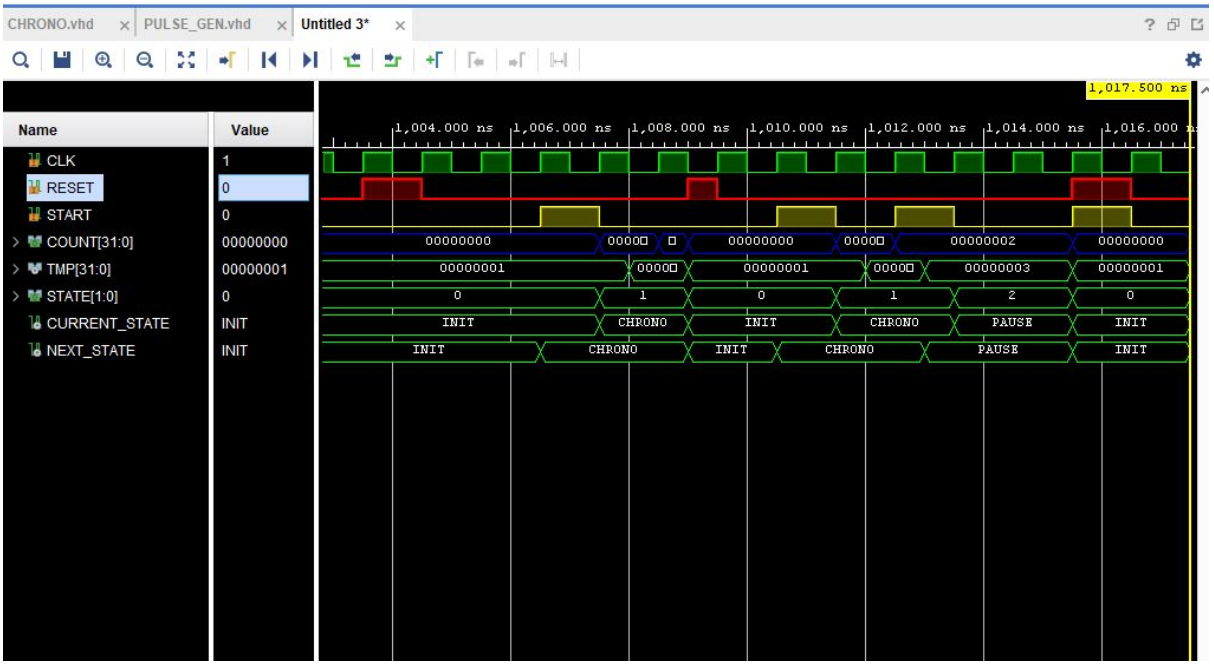


Figure 2.4 : CHRONO → RESET

Module SYSTEM (système global)

Pour la vérification du système global, nous avons divisé la simulation en plusieurs parties pour simplifier la vision des captures d'écran. Il faut dans tous les cas que le signal start soit de forme quelconque , pour s'assurer du fonctionnement du générateur de pulsion

- **Dans la Figure 2.5**, nous pouvons voir plusieurs transitions entre les états: quelle que soit la synchronisation (assurée ou non) et la longueur du signal START, il y aura une transition, d'un état à un autre :
    - **N.B:**Le système réagit deux fronts montants après un signal START, car on a un front montant pour générer le pulse dans PULSE_GEN, et un front montant pour réagir dans le chronomètre, et faire une transition d'état. Nous avons donc un total de deux fronts montants.
- **Dans la Figure 2.6**, quelles que soient les formes du signal et même si START est cliqué en même temps, le RESET va toujours faire une réinitialisation du système d'une façon asynchrone (il a la priorité).
- **Dans la Figure 2.7-2.8**: Cette simulation a la même stratégie de test que la simulation du module chronomètre. une vérification du fonctionnement des états

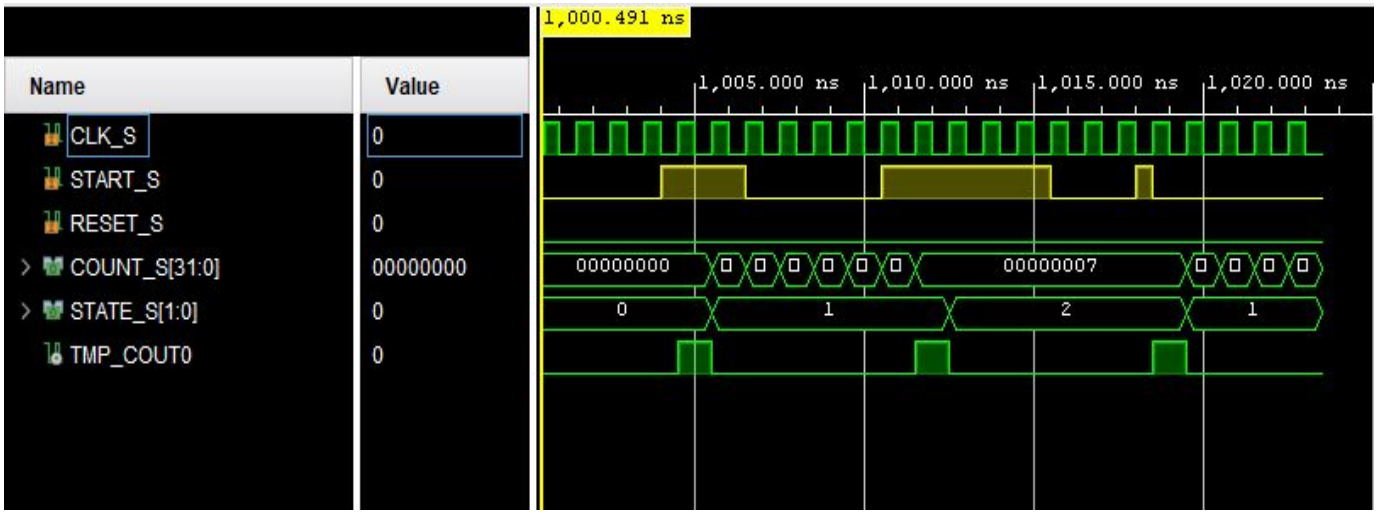


Figure 2.5 : Transition entre états

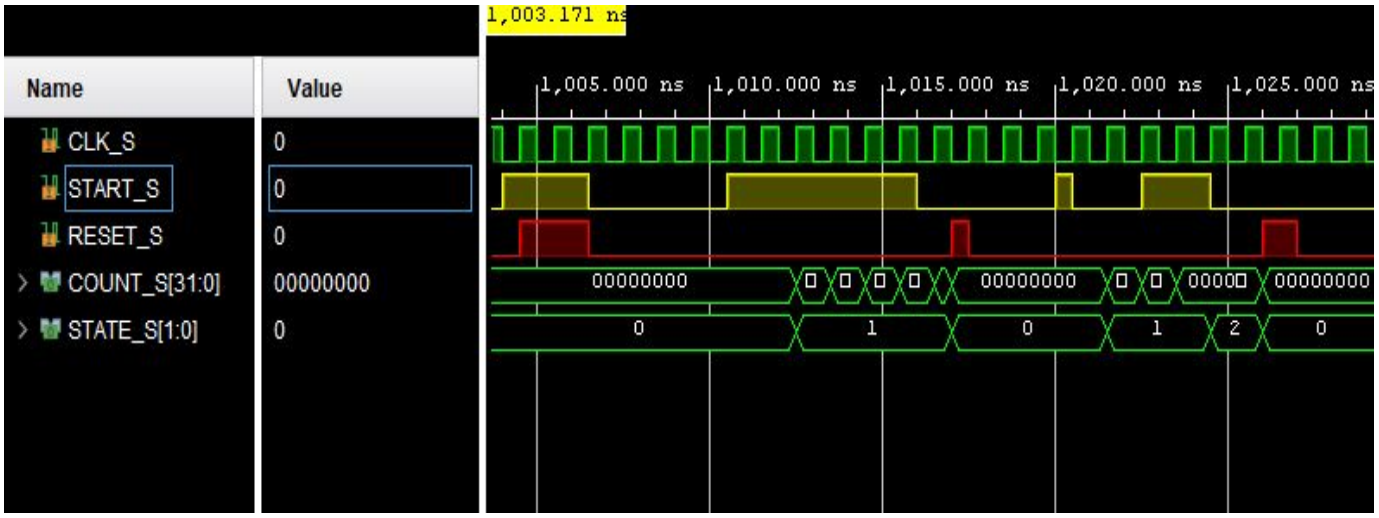


Figure 2.6 : RESET (système global)

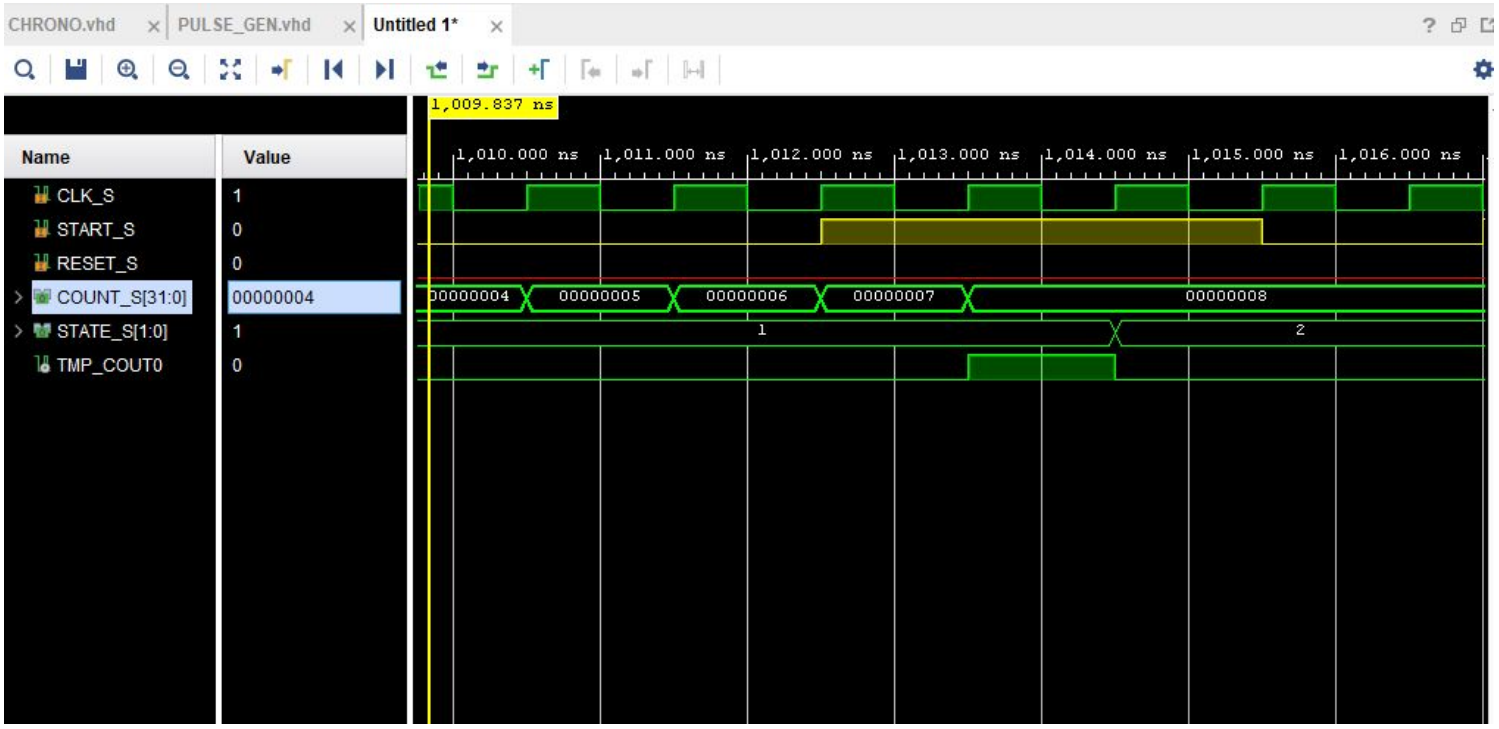


Figure 2.7: CHRONO → PAUSE

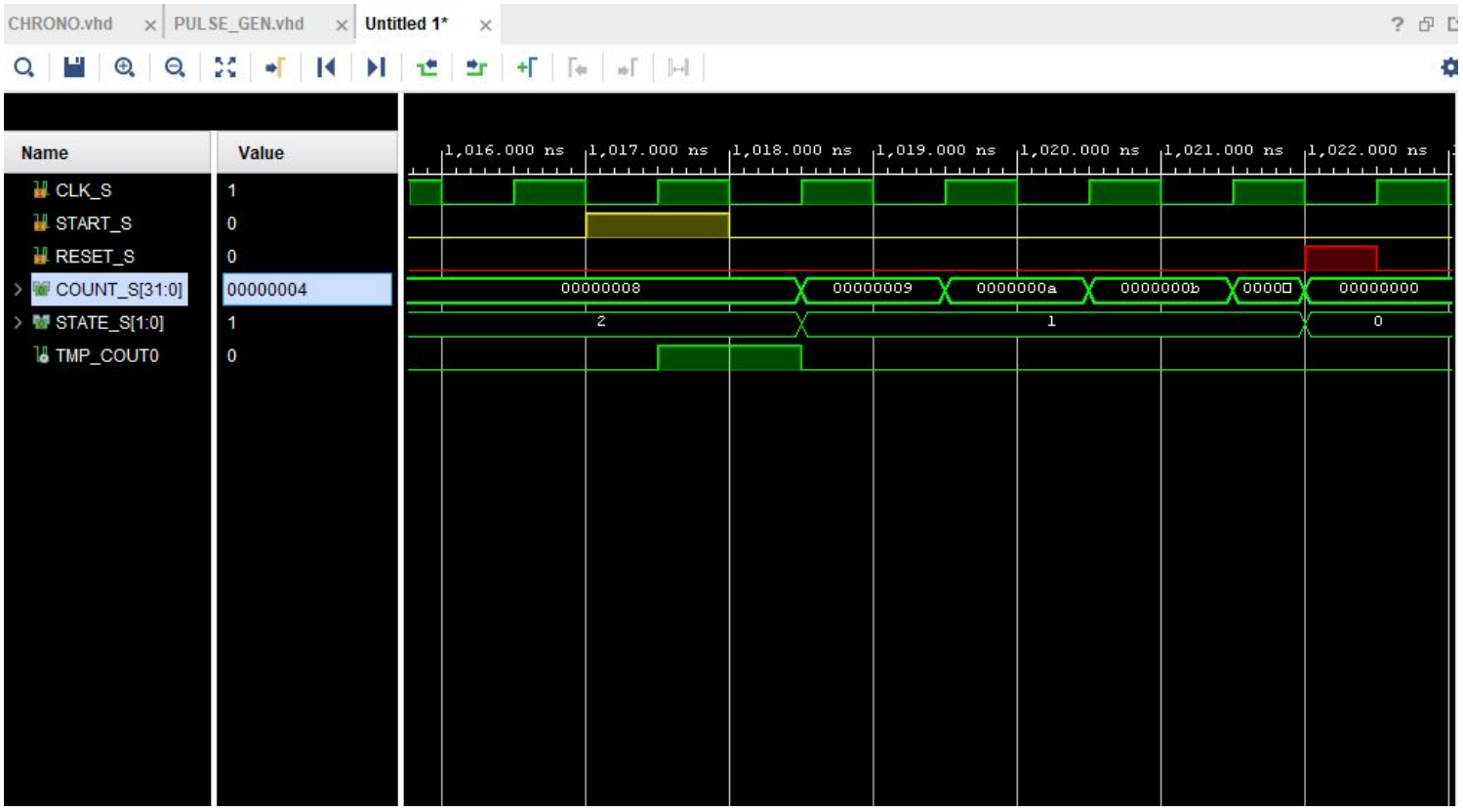


Figure 2.8 : PAUSE → CHRONO → INIT

3- Réponses aux questions

Question 1: Vos machines à état sont-elles une machine de Moore ou de Mealy ? Justifiez en quoi c'est l'un ou l'autre.

Nos machines à état sont des machines de **Moore** car, la valeur des sorties COUNT et STATE dépendent seulement de l'état courant du système. En effet, dans notre circuit, les entrées vont modifier l'état courant, mais ne vont pas directement influencer les sorties. Les sorties seront déterminées selon l'état, tel que décrit dans la description du système ainsi que dans nos codes et nos diagrammes d'état.

Question 2: Expliquez pourquoi vous avez choisi Mealy ou Moore.

La raison principale de choisir **Moore** c'est que le système fait un COUNT qui incrémente périodiquement (à chaque front montant du CLK) .Donc une machine de Mealy avec des entrées asynchrones (à part le RESET) va certainement avoir des sorties ayant des durées fortement en relation avec la période du CLK, et qui changeront rapidement par un changement d'état. Ce type de comportement est réalisable avec une Machine Moore qui est, de notre point de vue, plus simple à concevoir.

De plus, pour le PULSE_GEN, le signal doit être généré au prochain front montant, pour éviter des problèmes conséquents du cas où START n'est pas synchronisé avec CLK . Une machine de Mealy pourrait aussi peut être générer un pulse sans attendre le prochain front montant, mais seulement si START est synchronisé avec CLK ce qui a une probabilité négligeable étant donné que ce sont des utilisateurs qui pressent sur le bouton.

Pour le chronomètre, le seul avantage qu'aura sa réalisation en **Mealy** est de réduire le décalage entre la pression du bouton et la réaction du système. Ce décalage serait de 1 seul front montant du CLK, au lieu de 2. Mais on choisi de rester sur Moore pour les raisons suivantes :

- Moore est plus simple à concevoir, vous trouverez dans la dernière page du rapport un code VHDL du chronomètre en Mealy que nous avons fait.
    - Pour obtenir le comportement optimal attendu (un chronomètre qui réagit directement avec le click sur le bouton START, **sans attendre le front montant du CLK**, nous avons dû utiliser **"start'event"** dans l'état PAUSE.
    - Code en général plus compliqué.
- Le but du laboratoire n'est pas d'avoir un comportement optimal . Pour un vrai chronomètre, une erreur de deux fronts montant du CLK (20 ns) est négligeable devant une incrémentation qui se fait à chaque seconde (100 millions front montants).

Question 3: Quelle est la valeur maximale de comptage offerte par le chronomètre selon les consignes données ?

Selon l'énoncé, la sortie COUNT est sur 32 bits, donc la valeur maximale de comptage offerte par ce chronomètre est sur 32 bits. Le maximum en binaire. correspond au 32 bits à 1 : *11111111111111111111111111111111*. Ce qui est équivalent à **4 294 967 295** en binaire (2^32 -1). Donc la valeur maximale de comptage offerte par le chronomètre est de **4 294 967 295**.

Conclusion

Pour conclure, notre système (chronomètre) est composé de 2 modules, le module PULSE_GEN qui va générer une pulsion unique et qui sera relié au module CHRONO. Le module CHRONO va lui se comporter comme un chronomètre (tel que décrit dans la description du système). Pour concevoir les différents modules, nous avons utilisé uniquement du code VHDL (process), et une description structurale pour le système final. Par la suite, pour valider le système nous avons fait une multitude de simulations pour chaque module et pour différentes situations, tel qu'expliqué dans la vérification du système. Bref, de cette manière, il est clair que le système fonctionne correctement et qu'il a le comportement désiré.

```
--Partie combinatoire de l'architecture Mealy de l'entity CHRONO_VHDL (Non utilisée dans le laboratoire)
  process (CURRENT_STATE, START, CLK)
     begin
      case CURRENT_STATE is
    when INIT =>
        STATE <= "00";
        if (reset ='1') then next_state<= init; end if;
            COUNT <= (others => '0');
        if (START = '1' and RESET = '0') then
            COUNT<= X"00000001";
            TMP <= X"00000002";
            NEXT_STATE <= CHRONO;
        end if;
     when CHRONO =>
        STATE <= "01";
        if (reset ='1') then next_state<= init; end if;
            COUNT<=TMP;
        if (CLK'event and CLK= '0' and start = '0') then
            TMP <= STD_LOGIC_VECTOR(UNSIGNED(TMP)+1);
        end if;
        if (START = '1') then
            COUNT<=STD_LOGIC_VECTOR(UNSIGNED(TMP)-1);
            if (CLK'event and CLK='0' and RESET ='0' ) then
            NEXT_STATE <= PAUSE;
            end if;
        end if;
     when PAUSE =>
         STATE <= "10";
        if (reset ='1') then next_state<= init; end if;
            if (CLK'event and CLK= '0' and start = '0') then
                COUNT<= STD_LOGIC_VECTOR(UNSIGNED(TMP)-1);
            end if;
        if (start ='1' and start'event)  then
                COUNT <= TMP ;
                TMP <= STD_LOGIC_VECTOR(UNSIGNED(TMP)+1);
       elsif (START = '1' and clk = '0' and clk'event ) then
                NEXT_STATE <= CHRONO;
       end if; end case; end process; end Mealy;
```