HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

Reconfigurable & embedded
Digital Systems

Détecteur de clic/double clic

Départements : TIC

Unité d'enseignement CSN

Auteurs : **Rafael Dousse**

Professeur : **Etienne Messerli**
Assistant : **Anthony Jaccard**

Classe : **SysLog1-A ou -B**
Salle de labo : **A07 ou A09**

Date : **28.05.2024**

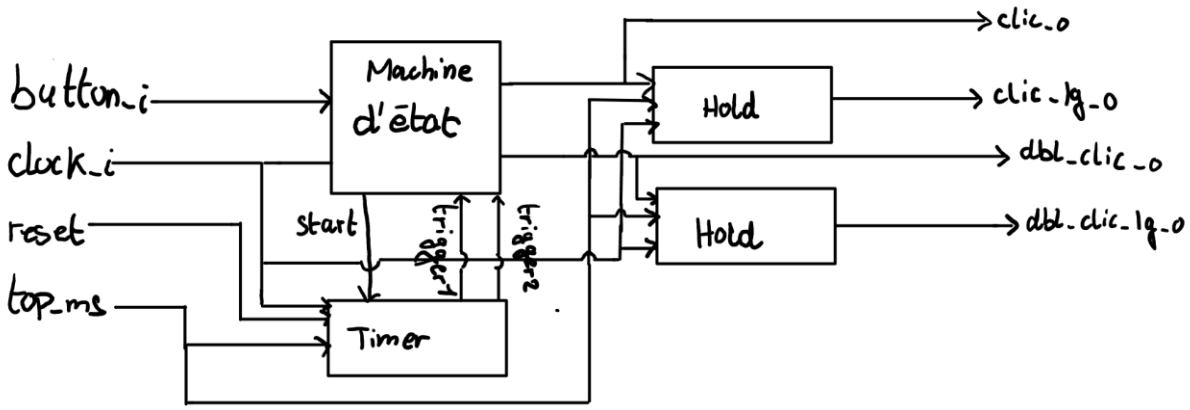
1 Introduction

Le but de ce laboratoire est d'implémenter un système qui détecte un simple clic ou un double clic tout en respectant certaines contraintes. Pour ce faire nous devons réaliser une machine d'état qui va me permettre de représenter chaque état pas lesquels on doit passer pour détecter un clic ou un double clic. Il est aussi nécessaire de réutiliser le concept de timer que nous avions développer lors du labo précédent.

Pour décrire le système, je vais décrire la conception de base avec les schéma fait main qui doit représenter ma machine complête puis je vais expliquer comment le code a été fait et finalement la synthéthisation et l autre…. -> a réecrire

2 Analyse et conception

Pour faire fonctionner la détection d'un clic/double clic il faut deux élèment. Une machine d'état et un timer pour calculer le temps. Un autre élément est utilisé qui est un bloc hold qui sert à maintenir une led allumé pendant plus d'un coup de clock (1 demi seconde) mais celui-là nous est déjà fourni. Le schéma général de à quoi devrait représenter le système est le suivant :



2.1 Machine d'état

Pour cette machine d'état j'ai choisi de faire 8 états pour représenter mon système. Les entrées du système qui permettent de passer d'un état à l'autre sont :
Reset, trigger1, triggers et top_ms.

Les sorties du système sont :
Double_clic, Clic et start_délai.

Les états sont les suivants :

- ATTENTE : C'est l'état de base dans lequel se trouve le système, dans lequel il se retrouve à la fin des états ou dans lequel il se retrouve si on appuie sur reset.
  La sorite start délai reste à 1 et passe a 0 quand elle passe à l'état appuyer. Elle va indiquer au timer de se lancer.

- APPUYER : C'est l'état après l'appui d'un bouton = 1 tout en s'assurant que les trigers soient aussi à 0 pour s'assurer de pas revenir dans cet état si on fait un appui long et qu'on se retrouve en attente avant de revenir sur appuyer. La condition de transition est synchronisée avec le signal top_ms.
  Si le trigger1 est activé, cela veut dire qu'on a attendu trop longtemps et on retourne à l'état d'attente.

- CLIC : La transitions d'état appuyer à clic se fait par le passage du bouton de 1 à 0. La sortie start délai passe à 1 et à 0 dans l'état suivant pour relancer le timer.

- ATTENTE_DBLCLK : Cet état nous permet de savoir si un clic est validé ou si on va devoir attendre un double clic. Si trigger2 est activé alors on va à l'état CONFIRMATION_CLIC si le bouton est activé avant le trigger2 alors c'est dans l'état APPUYER_DBLCLCK que l'on va.

- CONFIRMATION_CLIC : Cet état signifie que le trigger2 a été activé et qu'un unique clic a été enregistrer. La sortie clic est activé et après cet état, on retourne à l'état d'attente.

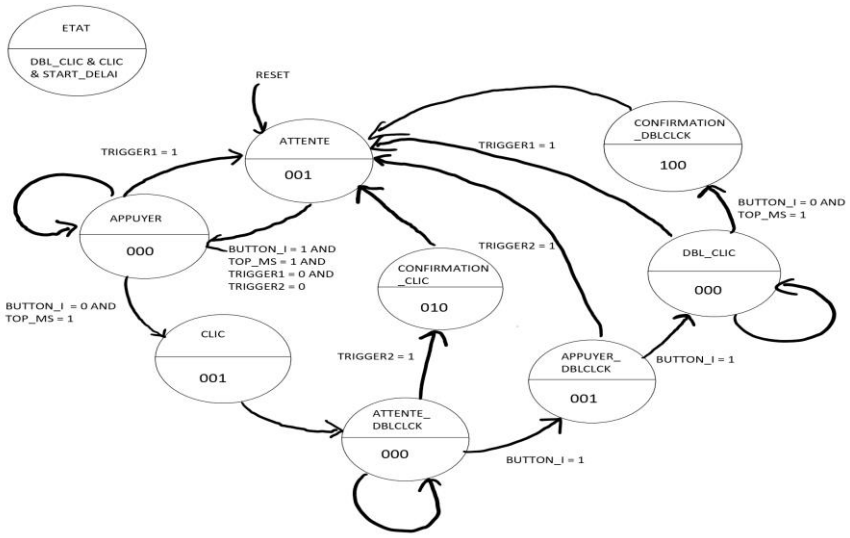
- APPUYER_DBLCLK : Si on se retrouve dans cet état, cela signifie que l'on a appuyer sur le bouton à temps et que on va devoir voir si on a un double clique. La sortie start_delai passe à 1 à nouveau.

- DBL_CLIC : Cet état atteint de savoir si on relâche le bouton dans les temps. Si trigger1 est à 1 alors on repasse dans l'état d'attente et si le bouton passe à 0 avant et que top_ms est à 1 alors on à bien un double clique.

- CONFIRMATION_DBLCLK : Cet état nous permet juste de mettre la sortie double clique à 0 et de retourner à l'état initiale d'attente.

Ce modèle contient des états qui sont assez similaires pour un appuie et double appuie et il aurait surement pu être possible de réduire ce nombre d'états en utilisant les mêmes états pour le clic et le double clic en rajoutant des conditions de transition.

Voici le schéma de la machine d'état :



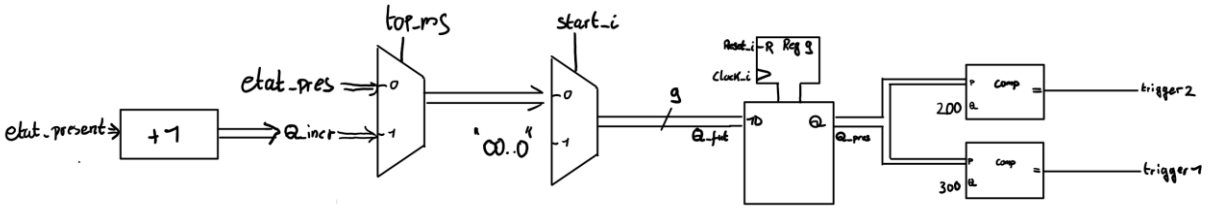
2.2 Timer

Le timer est plus simple à faire que celui du précédent laboratoire. Elle prent comme entrée le déclencheur *start*, le *top_ms* qui permet d'incrémenter l'état_présent à la fréquence d'1KHz. Il y a aussi comme entrée le *reset* et la *clock,* qui elle va à une fréquence de 1MHz. Finalement, il y a les 2 sorties trigger, qui s'active avec des comparateurs. Comme la fréquence de la clock est à 1MHz, on utilise le top_ms pour faire les calculs. Donc 1KHz donne 1ms. Il faut 200 coups de top_ms pour 200ms et 300 coup de top_ms pour avoir 300ms.
Les blocs qui régissent le système sont deux comparateurs, un registre, deux multiplexeurs et un additionneur.
L'état futur est dépendant de deux variables. Le start est prioritaire. Il permet de charger la valeur de 0 s'il est à 1 et s'il est à 0 alors c'est au tour du top_ms de donnée la valeur à l'état futur. Si top_ms est à 1, alors cela veut dire l'on doit incrémenter notre état_futur mais s'il est à 0 alors on garde la valeur de notre état présent. C'est comme ça qu'on peut calculer le délai d'activation des triggers. De plus, il n'y a pas besoin de soucier de remettre à 0 l'état présent si on atteint 300 et de mettre un multiplexeur en plus. Cela est assuré par le fait que si on atteint 200 ou 300 et que l'on change d'état (dans notre machine d'état) à l'état présent, alors le start_délai va être à 1 tant qu'on est dans l'état d'ATTENTE ce qui fait que dès qu'on appuie sur un bouton, il va repasser à 0 et charger la valeur de 0 dans notre état futur.
Voici la table des fonctions synchrones :

Start_i	top_ms	etat_pres	etat_fut	fonction
1	–	–	0	Load
0	0	Q	Q	Hold
0	1	Q	Q+1	incr

L'unique fonction asynchrone du système est le reset.

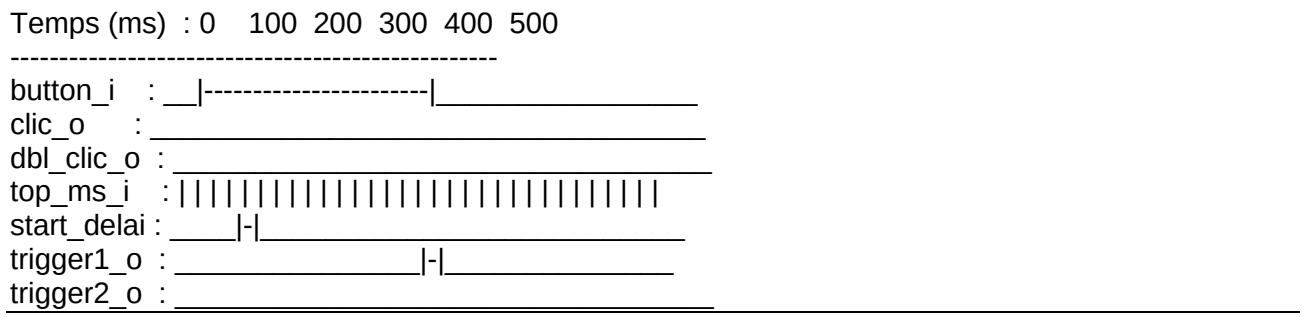
Et voici le schéma du système :



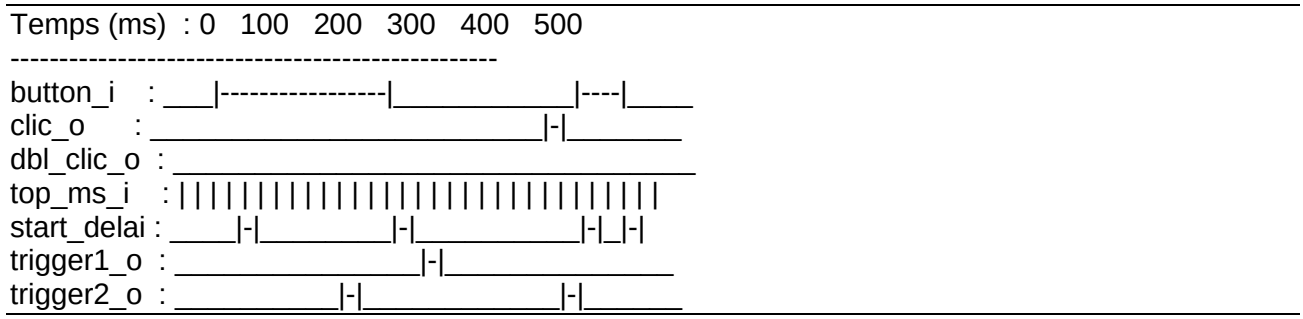
2.3 Chronogramme

Voici une représentation approximative de ce à quoi devrait ressemblé les chronogrammes lors d'un clic/double clic manqué ainsi que d'un cli/double clic capturé.

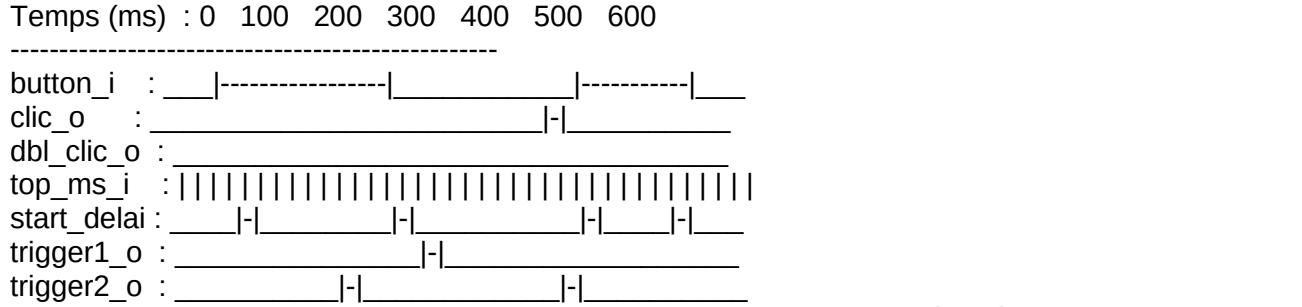
Clic manqué :



Clic detecté:

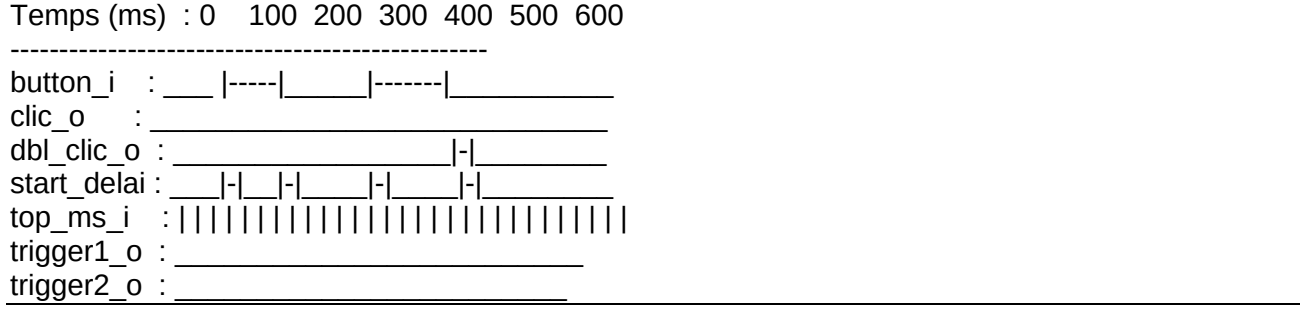


Double clic Manqué



Ici, c'est le même que pour le clique car si on double clique n'est pas détecté alors c'est un clic. Le délai entre les deux cliques de boutons est supérieur à 200 ms

Double clic détecté



3 Réalisation et implantation

Pour le code VHDL, j'ai appris trop tard (pas fait attention non plus à la donné) qu'il était nécessaire de faire un fichier séparer uniquement pour la machine d'état. J'ai donc tout fait dans le fichier det_clic_dblclic_top et le timer dans son fichier timer.

3.1 Code VHDL du timer

Voila comment j'ai fait le code du vhdl avec le multiplexer pour mes états futur, le process pour le registre et les sorties trigger :

```
entity timer is
    generic (
        T1_g : natural range 1 to 1023 := 2;
        T2_g : natural range 1 to 1023 := 3 );
    port (
        clock_i    : in  std_logic;
        reset_i    : in  std_logic;
        start_i    : in  std_logic;
        top_ms_i   : in  std_logic;
        trigger1_o : out std_logic;
        trigger2_o : out std_logic
        );
end timer;

architecture comport of timer is

  ----------
  -- TODO --
  ----------

  -- Declaration des signaux internes
  signal etat_fut : unsigned(10 downto 0);
  signal etat_pres : unsigned(10 downto 0);
  signal Q_incr : unsigned(10 downto 0);


begin


  ----------
  -- TODO --
  ----------


  -- Multiplexeur pour etat_fut

    etat_fut <= (others => '0' ) when start_i = '1' else
    Q_incr when top_ms_i = '1'  else
    etat_pres;

  process(clock_i, reset_i)
    begin
      if reset_i = '0' then
        etat_pres <= (others => '0');
```

```
    elsif rising_edge(clock_i) then
      etat_pres <= etat_fut;
    end if;
end process;

-- Incrémentation de l'état
Q_incr <= etat_pres + 1;


-- Detection des triggers
-- trigger1 pour le simple clic donc est actif quand T1 = T1_g est atteint
trigger1_o <= '1' when etat_pres =  to_unsigned(T1_g, etat_pres'length)  else '0';
-- trigger2 pour le double clic donc est actif quand T2 = T2_g est atteint
trigger2_o <= '1' when etat_pres = to_unsigned(T2_g, etat_pres'length) else '0';


end comport;
```

3.2 **Code VHDL du det_clic_dblclic**

Pour la déclaration des états, je l'ai fait en déclarant un type appelé *state_type*. La déclaration des types de cette manière est plus simple à utiliser mais elle représente un inconvénient. C'est qu'on ne sait pas le code que va donner le synthétiseur. Il est mieux de déclarer des constantes, afin que le code soit plus portable et que l'on puisse donnée un code nous même et qu'il soit pareil à chaque fois. Pour ce laboratoire, le professeur m'a dit que ça allait de laisser avec l'enum mais de montrer le code fait par le synthétiseur que voici :

	Name	CONFIRMATION_DBLCLCK	DBL_CLIC	APPUYER_DBLCLCK	CONFIRMATION_CLIC	ATTENTE_DBLCLCK	CLIC	APPUYER	ATTENTE
1	ATTENTE	0	0	0	0	0	0	0	0
2	APPUYER	0	0	0	0	0	0	1	1
3	CLIC	0	0	0	0	0	1	0	1
4	ATTENTE_DBLCLCK	0	0	0	0	1	0	0	1
5	CONFIRMATION_CLIC	0	0	0	1	0	0	0	1
6	APPUYER_DBLCLCK	0	0	1	0	0	0	0	1

Transitions / Encoding

Voici la déclaration des états :

```
type state_type is
   (ATTENTE,
    APPUYER,
    CLIC,
    ATTENTE_DBLCLCK,
    CONFIRMATION_CLIC,
    APPUYER_DBLCLCK,
    DBL_CLIC,
    CONFIRMATION_DBLCLCK);
```

Ensuite, j'ai instancié trois composants. Un timer qui prends les constantes de temps de 200 et 300 déclaré dans un fichier externe. Il y a aussi deux composant de maintient pour allumer la led de 0.5 secondes. Et après il y a deux process. Un pour la machine d'état et un autre pour le registre qui va s'occuper de faire les changements d'état en fonction de la clock ou revenir à l'état initial si le reset est activé. Le code est à retrouver à la fin du rapport.

4 Simulation

La simulation m'a posé beaucoup de problème pour savoir exactement quel étaient les problèmes. J'ai pu résoudre la plupart de problème mais le manque de temps et le surplus de labos a fait que je n'ai pas pu résoudre tous les problèmes qui sont apparu. La plupart des détections se font mais il y a trois doubles cliques qui ne sont pas détecté. J'ai l'impression que le code est correct mais il y a surement un souci ou deux qui font que la synchronisation se fait peut-être -à un coup de clock trop tard ou peut être qu'il manque un état auquel je n'ai pas pensé. Je mets quand même 3 images des deux premiers appuie qui montre que le délai à été dépassé pour la double clique et que donc c'est bien un coup de clique qui est détecté.

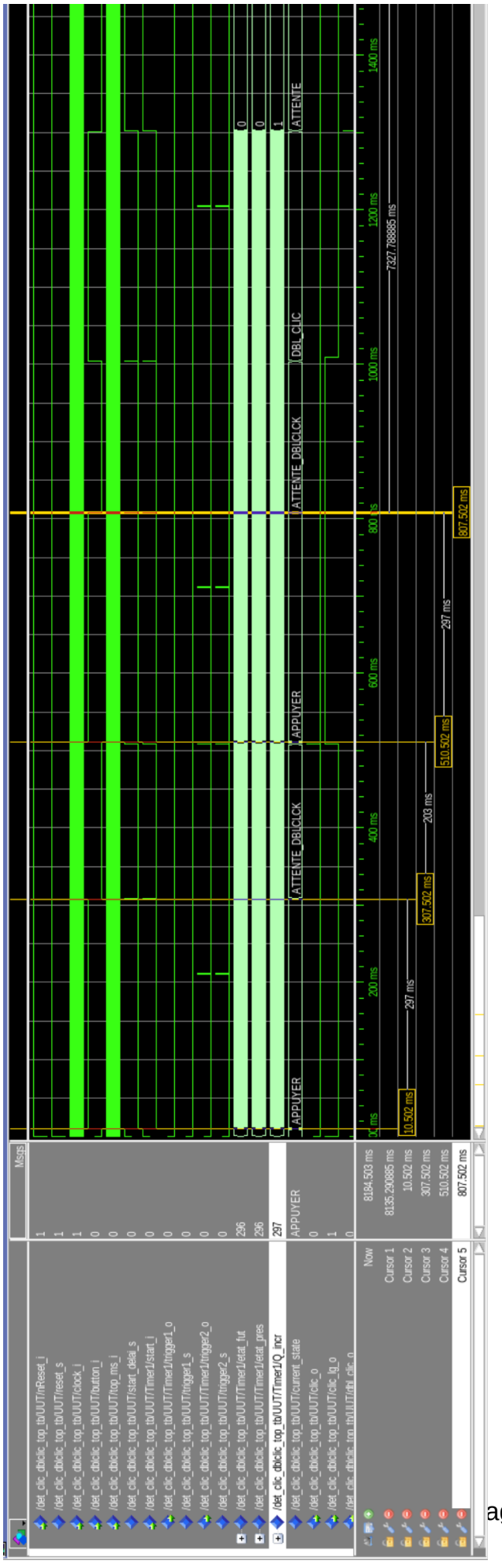
Ce que l'on peut voir sur les 2 derniers chronogrammes est le temps pris par le premier bouton qui est de 297ms donc inférieur à 300 donc ça peut être un clique. Ensuite le délai entre le passage à 0 du bouton et le passage à 1 du second clique sur le bouton est de 202ms donc plus grand que 200ms. Ça veut dire que le trigger est activé et que on a bien un seul clique et non pas un double clique. On a du mal à le voir sur l'image mais le clic long est à 1 ce qui montre qu'il est bien détecté. Les trois chronogrammes sont sur les deux pages suivantes pour les voir en grand.

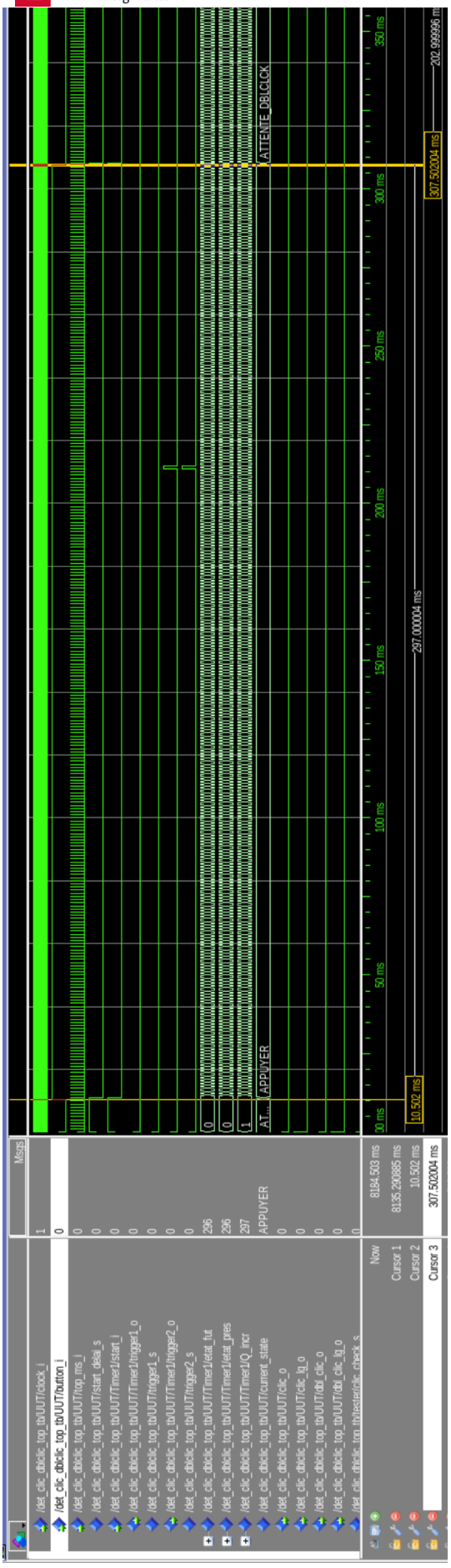
Logique et fmax

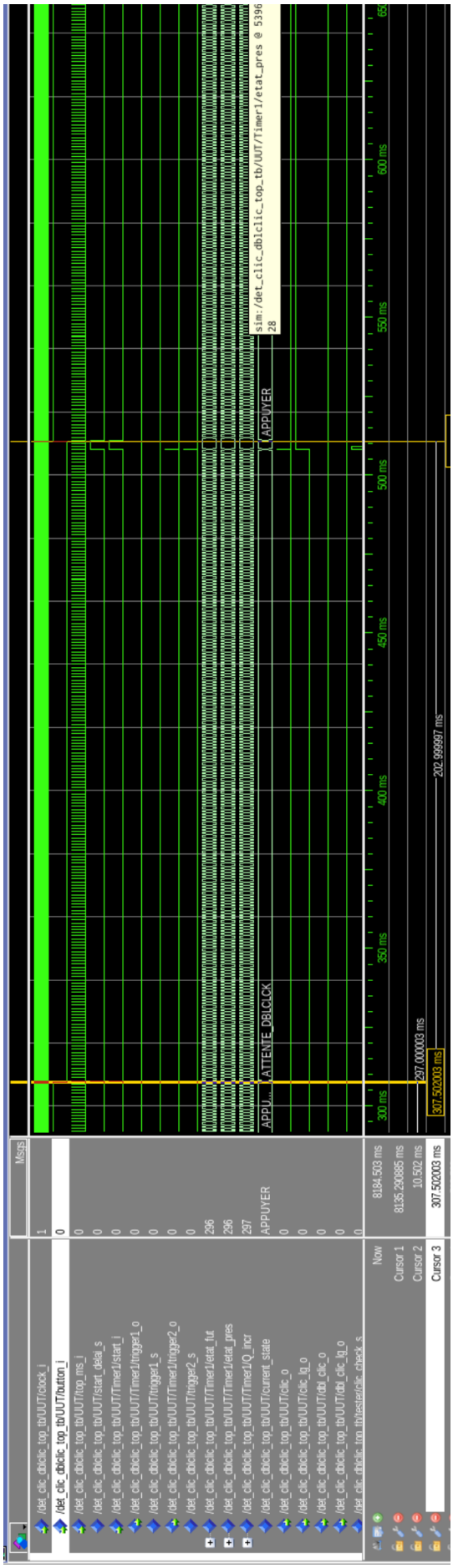
Le nombre total d'éléments logique est de 79. La fréquence max calculé par le synthétiseur est résumé dans quartus. Elle indique :

Fmax: 54.18MHz, Restricted Fmax : 54.18MHz, Clock Name: clock_i
Fmax: 60.82MHz, Restricted Fmax : 60.82MHz, Clock Name: button_i

Pour calculer le Fmax, le synthétiseur va anlyser le chemin le plus long que va devoir parcours les données entre deux éléments séquentiels, calculer les retards potentiel des composants. Il connait aussi le délai de propagation ce qui lui permet de faire les calculs des délais et des chemins les plus longs et de trouver la fréquence maximale.


/det_clic_dbclic_top_tb/UUT/nReset_i
/det_clic_dbclic_top_tb/UUT/reset_s
/det_clic_dbclic_top_tb/UUT/clock_i
/det_clic_dbclic_top_tb/UUT/button_i
/det_clic_dbclic_top_tb/UUT/top_ms_i
/det_clic_dbclic_top_tb/UUT/start_delai_s
/det_clic_dbclic_top_tb/UUT/timer1/start_i
/det_clic_dbclic_top_tb/UUT/timer1/trigger1_o
/det_clic_dbclic_top_tb/UUT/trigger1_s
/det_clic_dbclic_top_tb/UUT/timer1/trigger2_o
/det_clic_dbclic_top_tb/UUT/trigger2_s
/det_clic_dbclic_top_tb/UUT/timer1/etat_fut
/det_clic_dbclic_top_tb/UUT/timer1/etat_pres
/det_clic_dbclic_top_tb/UUT/timer1/Q_incr
/det_clic_dbclic_top_tb/UUT/current_state
/det_clic_dbclic_top_tb/UUT/clic_o
/det_clic_dbclic_top_tb/UUT/clic_lg_o
Msgs
Now
Cursor 1
Cursor 2
Cursor 3
Cursor 4
Cursor 5
APPUYER
ATTENTE_DBLCLCK
DBL_CLIC
ATTENTE
10.502 ms
297 ms
807.502 ms
203 ms
510.502 ms
7327.788885 ms


/det_clic_dbclic_top_tb/UUT/clock_i
/det_clic_dbclic_top_tb/UUT/button_i
/det_clic_dbclic_top_tb/UUT/top_ms_i
/det_clic_dbclic_top_tb/UUT/start_delai_s
/det_clic_dbclic_top_tb/UUT/Timer1/start_i
/det_clic_dbclic_top_tb/UUT/Timer1/trigger1_o
/det_clic_dbclic_top_tb/UUT/trigger1_s
/det_clic_dbclic_top_tb/UUT/Timer1/trigger2_o
/det_clic_dbclic_top_tb/UUT/trigger2_s
/det_clic_dbclic_top_tb/UUT/Timer1/etat_fut
/det_clic_dbclic_top_tb/UUT/Timer1/etat_pres
/det_clic_dbclic_top_tb/UUT/Timer1/Q_incr
/det_clic_dbclic_top_tb/UUT/current_state
/det_clic_dbclic_top_tb/UUT/clic_o
/det_clic_dbclic_top_tb/UUT/clic_lg_o
/det_clic_dbclic_top_tb/UUT/dbl_clic_o
/det_clic_dbclic_top_tb/UUT/dbl_clic_lg_o
/det_clic_dbclic_top_tb/testericlic_check_s
Msgs
Now
Cursor 1
Cursor 2
Cursor 3
APPUYER
ATTENTE_DBLCLICK
297.0000004 ms
10.502 ms
307.5020004 ms

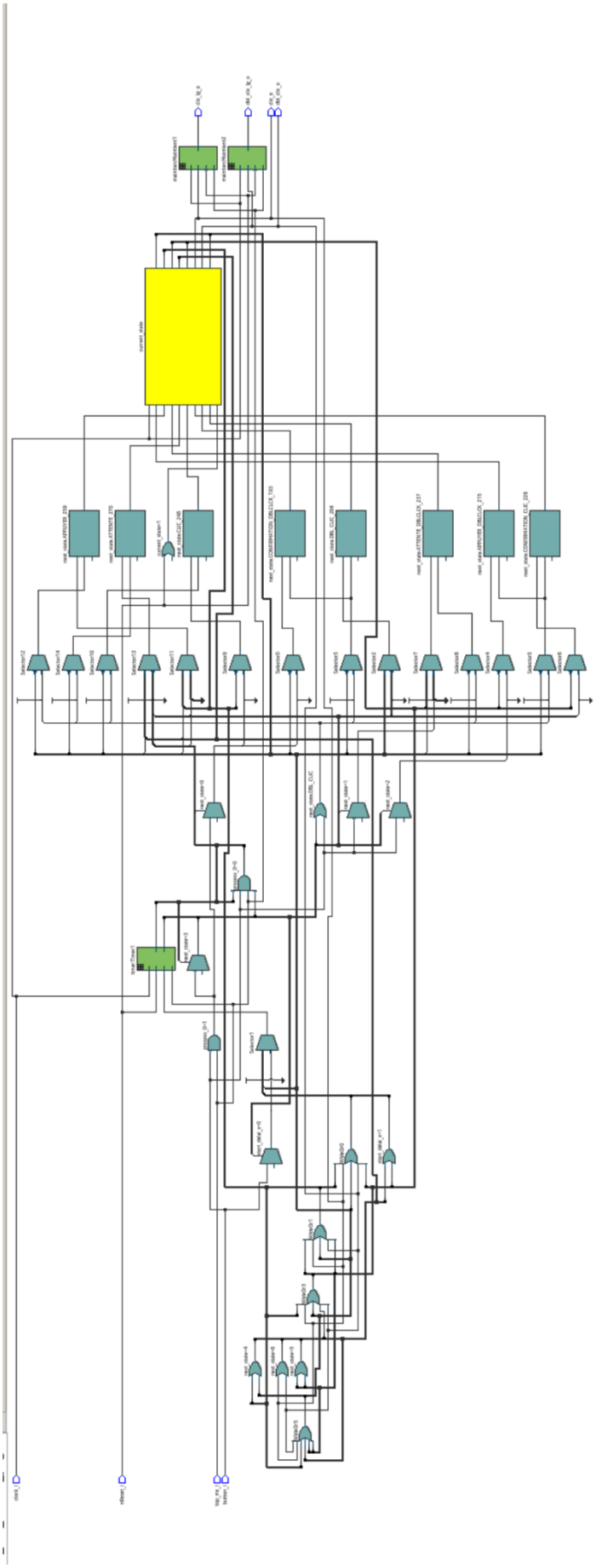

/det_clic_dbclic_top_tb/UUT/clock_i
/det_clic_dbclic_top_tb/UUT/button_i
/det_clic_dbclic_top_tb/UUT/top_ms_i
/det_clic_dbclic_top_tb/UUT/start_delai_s
/det_clic_dbclic_top_tb/UUT/Timer1/start_i
/det_clic_dbclic_top_tb/UUT/Timer1/trigger1_o
/det_clic_dbclic_top_tb/UUT/trigger1_s
/det_clic_dbclic_top_tb/UUT/Timer1/trigger2_o
/det_clic_dbclic_top_tb/UUT/trigger2_s
/det_clic_dbclic_top_tb/UUT/Timer1/etat_fut
/det_clic_dbclic_top_tb/UUT/Timer1/etat_pres
/det_clic_dbclic_top_tb/UUT/Timer1/Q_incr
/det_clic_dbclic_top_tb/UUT/current_state
/det_clic_dbclic_top_tb/UUT/clic_o
/det_clic_dbclic_top_tb/UUT/clic_lg_o
/det_clic_dbclic_top_tb/UUT/dbl_clic_o
/det_clic_dbclic_top_tb/UUT/dbl_clic_lg_o
/det_clic_dbclic_top_tb/testericlic_check_s
sim:/det_clic_dbclic_top_tb/UUT/Timer1/etat_pres @ 5396
28
Msgs
Now
Cursor 1
Cursor 2
Cursor 3
APPUYER
ATTENTE_DBLCLICK
297.0000003 ms
307.5020003 ms
202.999997 ms

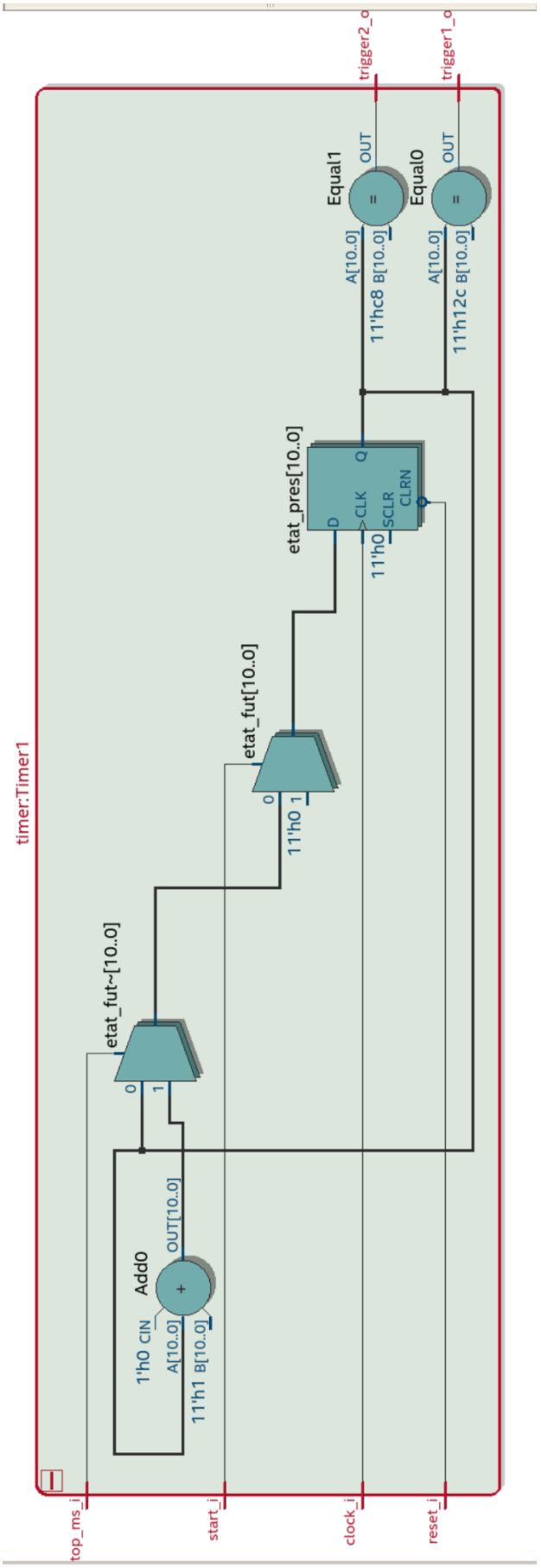
Voici les logs qui montre que la plupart des tests passe mes pas tous :

```
# vsim -voptargs=""+acc"" work.det_clic_dblclic_top_tb
# Start time: 21:48:05 on May 28,2024
# ** Note: (vsim-3813) Design is being optimized due to module recompilation...
# ** Note: (vopt-143) Recognized 1 FSM in architecture body "det_clic_dblclic_top(struct)".
# Loading std.standard
# Loading std.textio(body)
# Loading ieee.std_logic_1164(body)
# Loading ieee.numeric_std(body)
# Loading work.det_clic_dblclic_pkg
# Loading work.det_clic_dblclic_top_tb(struct)#1
# Loading work.det_clic_dblclic_top(struct)#1
# Loading work.timer(comport)#1
# Loading work.maintien(comport)#1
# Loading work.det_clic_dblclic_top_tester(tester)#1
 run -all
# ** Warning: NUMERIC_STD."=": metavalue detected, returning FALSE
#    Time: 0 ns  Iteration: 0  Instance: /det_clic_dblclic_top_tb/UUT/Maintient2
# ** Warning: NUMERIC_STD."=": metavalue detected, returning FALSE
#    Time: 0 ns  Iteration: 0  Instance: /det_clic_dblclic_top_tb/UUT/Maintient1
# ** Warning: NUMERIC_STD."=": metavalue detected, returning FALSE
#    Time: 0 ns  Iteration: 0  Instance: /det_clic_dblclic_top_tb/UUT/Timer1
# ** Warning: NUMERIC_STD."=": metavalue detected, returning FALSE
#    Time: 0 ns  Iteration: 0  Instance: /det_clic_dblclic_top_tb/UUT/Timer1
# ** Note: génération d'un clic
#    Time: 10502 us  Iteration: 0  Instance: /det_clic_dblclic_top_tb/tester
# ** Note: génération d'un double-clic
#    Time: 510502 us  Iteration: 0  Instance: /det_clic_dblclic_top_tb/tester
# ** Note: génération d'un clic
#    Time: 1505502 us  Iteration: 0  Instance: /det_clic_dblclic_top_tb/tester
# ** Note: génération d'un clic
#    Time: 2014502 us  Iteration: 0  Instance: /det_clic_dblclic_top_tb/tester
# ** Note: Pas de détection de clic
#    Time: 2525502 us  Iteration: 0  Instance: /det_clic_dblclic_top_tb/tester
# ** Note: Pas de détection de clic
#    Time: 3027502 us  Iteration: 0  Instance: /det_clic_dblclic_top_tb/tester
# ** Note: génération d'un double-clic
#    Time: 3680502 us  Iteration: 0  Instance: /det_clic_dblclic_top_tb/tester
# ** Error: Erreur lors d'un double-clic, code de l'erreur: pdblc0
#    Time: 4482002 us Started: 3680002 us  Scope:
/det_clic_dblclic_top_tb/tester/assert__pdblclick File:
../src_tb/det_clic_dblclic_top_tester.vhd Line: 364
# ** Note: Pas de détection de double-clic
#    Time: 4680502 us  Iteration: 0  Instance: /det_clic_dblclic_top_tb/tester
# ** Error: Erreur lors d'un clic, code de l'erreur: pc0
#    Time: 4683002 us Started: 4178002 us  Scope:
/det_clic_dblclic_top_tb/tester/assert__pclick File: ../src_tb/det_clic_dblclic_top_tester.vhd
Line: 334
```

```
# ** Note: génération d'un double-clic
#    Time: 5682502 us  Iteration: 0  Instance: /det_clic_dblclic_top_tb/tester
# ** Error: Erreur lors d'un double-clic, code de l'erreur: pdblc0
#    Time: 6484002 us Started: 5682002 us  Scope:
/det_clic_dblclic_top_tb/tester/assert__pdblclick File:
../src_tb/det_clic_dblclic_top_tester.vhd Line: 364
# ** Note: génération d'un clic
#    Time: 6682502 us  Iteration: 0  Instance: /det_clic_dblclic_top_tb/tester
# ** Error: Erreur lors d'un clic, code de l'erreur: pc0
#    Time: 6685002 us Started: 6180002 us  Scope:
/det_clic_dblclic_top_tb/tester/assert__pclick File: ../src_tb/det_clic_dblclic_top_tester.vhd
Line: 334
# ** Note: génération d'un double-clic
#    Time: 7183502 us  Iteration: 0  Instance: /det_clic_dblclic_top_tb/tester
# ** Error: Erreur lors d'un double-clic, code de l'erreur: pdblc0
#    Time: 7986002 us Started: 7184002 us  Scope:
/det_clic_dblclic_top_tb/tester/assert__pdblclick File:
../src_tb/det_clic_dblclic_top_tester.vhd Line: 364
# 8184502000 ns  >> fin de la simulation
#               >> Vérifiez dans le log si il y des erreurs!
```

5 Vues RTL




timer:Timer1
trigger2_o
trigger1_o
Equal1
Equal0
OUT
A[10..0]
B[10..0]
11'hc8
11'h12c
etat_pres[10..0]
D
Q
CLK
SCLR
CLRN
11'h0
etat_fut[10..0]
11'h0 1
etat_fut~[10..0]
Add0
OUT[10..0]
CIN
11'h1
top_ms_i
start_i
clock_i
reset_i

6 Intégration sur la carte

Malheureusement, je n'ai pas eu le temps de tester l'intégration sur la carte. J'ai eu beaucoup de problème à trouver les bons états et la bonne manière de faire. Et même dans l'état actuel du code. J'ai encore trois doubles cliques qui ne sont pas identifier et je ne comprends toujours pas pourquoi. Je n'ai donc pas pu faire valider mon travail par l'assistant.

7 Conclusion

Ce laboratoire a permis d'implémenter un système de détection de clique/double clique en implémentant un système qui utilise une machine d'état et un timer. Ce n'est pas quelque chose de simple à réaliser afin qu'il soit compétemment fonctionnel. Il faut pouvoir penser à toutes les différentes possibilités d'états que peut avoir le système et c'est compliqué dans un système séquentielle full synchrone. J'en ai malheureusement fait l'expérience. La surcharge de travail que nous avons en ce moment fait que mon système à quelque lacune qui fait que tous les cas ne sont pas forcément détectés et je n'ai pu trouver les erreurs restantes malgré une conception de la machine qui me parait cohérente.

Date : 20 Mai 2024

Noms des étudiants : Rafael Dousse

8 Annexes ….

Code du fichier vhdl *det_dblclic_top*

```
--------------------------------------------------------------------------------
-- HEIG-VD, Haute Ecole d'Ingenierie et de Gestion du canton de Vaud
-- Institut REDS, Reconfigurable & Embedded Digital Systems
--
-- Fichier      : det_clic_dblclic_top.vhd
-- Auteur       : Etienne Messerli, le 05.05.2016
--
-- Description  : Detection d'un clic et double clic
--                Projet repris du labo Det_Clic_DblClic 2012
--
-- Utilise      : Labo SysLog2 2016
--| Modifications |------------------------------------------------------------
-- Ver   Date        Qui        Description
-- 1.0   20.11.2020  EMI   Ajout generique pour timer et maintien
--
--------------------------------------------------------------------------------

library ieee;
    use ieee.std_logic_1164.all;
    use ieee.numeric_std.all;

use work.det_clic_dblclic_pkg.all;

entity det_clic_dblclic_top is


    generic (T1_g      : natural range 1 to 1023 := 4;
             T2_g      : natural range 1 to 1023 := 6;
             T_HOLD    : natural range 1 to 1023 := 2
             );
    port(clock_i       : in  std_logic;  --horloge systeme 1MHz
         nReset_i      : in  std_logic;  --reset asynchrone
         button_i      : in  std_logic;
         top_ms_i      : in  std_logic;
         clic_o        : out std_logic;
         dbl_clic_o    : out std_logic;
         clic_lg_o     : out std_logic;
         dbl_clic_lg_o : out std_logic
         );
end det_clic_dblclic_top;

architecture struct of det_clic_dblclic_top is

  -- Internal signal declarations
  signal reset_s       : std_logic;
```

```
----------
-- TODO --
----------
-- Enum pour les états
type state_type is
(ATTENTE,
 APPUYER,
 CLIC,
 ATTENTE_DBLCLCK,
 CONFIRMATION_CLIC,
 APPUYER_DBLCLCK,
 DBL_CLIC,
 CONFIRMATION_DBLCLCK);

-- Signaux pour les états
signal current_state, next_state : state_type;
-- Signaux pour les triggers
signal trigger1_s, trigger2_s : STD_LOGIC;
-- Signal pour le clic simple
signal clic_s, dbl_clic_s : std_logic;
-- Signal sortie clics long
signal clic_lg_s, dbl_clic_lg_s : std_logic;
-- Signal pour le timer
signal start_delai_s : std_logic;

signal reset_hold : std_logic;


-- Component declarations


----------
-- TODO --
----------

component timer is
    generic (T1_g : natural range 1 to 1023 := 4;
             T2_g : natural range 1 to 1023 := 6
            );
    Port ( clock_i    : in  std_logic;
           reset_i    : in  std_logic;
           start_i    : in  std_logic;
           top_ms_i   : in  std_logic;
           trigger1_o : out std_logic;
           trigger2_o : out std_logic
    );
end component;
```

```
    component maintien
        generic (T_HOLD : natural range 1 to 1023 := 2
                );
        port (clock_i    : in  std_logic;
              reset_i    : in  std_logic;
              pulse_i    : in  std_logic;
              top_ms_i   : in  std_logic;
              p_hold_o   : out std_logic
              );
    end component;
    for all : maintien use entity work.maintien;



begin

    -- Reset
    reset_s <= nReset_i;
    reset_hold <= not nReset_i;

    -- Timer
    Timer1 : timer
    generic map (T1_g => T1_c, T2_g => T2_c)
    Port map (
        clock_i => clock_i,
        reset_i => nReset_i,
        start_i => start_delai_s,
        top_ms_i => top_ms_i,
        trigger1_o => trigger1_s,
        trigger2_o => trigger2_s
    );
    -- Maintient pour le clic simple
    Maintient1 : maintien
    generic map (T_HOLD => T_HOLD_C)
    Port map (
        clock_i => clock_i,
        reset_i => reset_hold,
        pulse_i => clic_s,
        top_ms_i => top_ms_i,
        p_hold_o => clic_lg_s
    );
    -- Maintient pour le double clic
    Maintient2 : maintien
    generic map (T_HOLD => T_HOLD_C)
    Port map (
        clock_i => clock_i,
        reset_i => reset_hold,
        pulse_i => dbl_clic_s,
        top_ms_i => top_ms_i,
        p_hold_o => dbl_clic_lg_s
```

```
);


-- Gestion des états
process(current_state, button_i, trigger1_s, trigger2_s, top_ms_i)
begin
    -- Initialisations des sorties a 0
    clic_s <= '0';
    dbl_clic_s <= '0';
    start_delai_s <= '0';

    -- Cas de la machine à états
    case current_state is
        when ATTENTE =>
            start_delai_s <= '1';
            if button_i = '1' and top_ms_i = '1' and trigger1_s = '0' and trigger2_s = '0' then
                next_state <= APPUYER;
            else
                next_state <= ATTENTE;
            end if;

        when APPUYER =>
            if trigger1_s = '1' then
                next_state <= ATTENTE;
            elsif button_i = '0'and top_ms_i ='1'then
                next_state <= CLIC;

            else
                next_state <= APPUYER;
            end if;

        when CLIC =>
            start_delai_s <= '1';
            next_state <= ATTENTE_DBLCLCK;

        when ATTENTE_DBLCLCK =>
            if trigger2_s = '1' then
                next_state <= CONFIRMATION_CLIC;
            elsif button_i = '1' then
                next_state <= APPUYER_DBLCLCK;
            else
                next_state <= ATTENTE_DBLCLCK;
            end if;

        when CONFIRMATION_CLIC =>
            clic_s <= '1';
            next_state <= ATTENTE;

        when APPUYER_DBLCLCK =>
```

```
            if trigger2_s = '1' then
                next_state <= ATTENTE;
                elsif button_i = '1'  then
                    start_delai_s <= '1';
                    next_state <= DBL_CLIC;
                end if;

            when DBL_CLIC =>
                if trigger1_s = '1' then
                    next_state <= ATTENTE;
                elsif button_i = '0' and top_ms_i = '1' then
                    next_state <= CONFIRMATION_DBLCLCK;
                else
                    next_state <= DBL_CLIC;
                end if;

            when CONFIRMATION_DBLCLCK =>
                dbl_clic_s <= '1';
                next_state <= ATTENTE;

            when others =>
                next_state <= ATTENTE;
        end case;
    end process;


    -- Machine à états pour gérer les transitions
    process(clock_i, reset_s)
    begin
        if reset_s = '0' then
            current_state <= ATTENTE;
        elsif rising_edge(clock_i) then
            current_state <= next_state;
        end if;
    end process;

    clic_lg_o <= clic_lg_s;
    dbl_clic_lg_o <= dbl_clic_lg_s;
    clic_o <= clic_s;
    dbl_clic_o <= dbl_clic_s;


end struct;
```