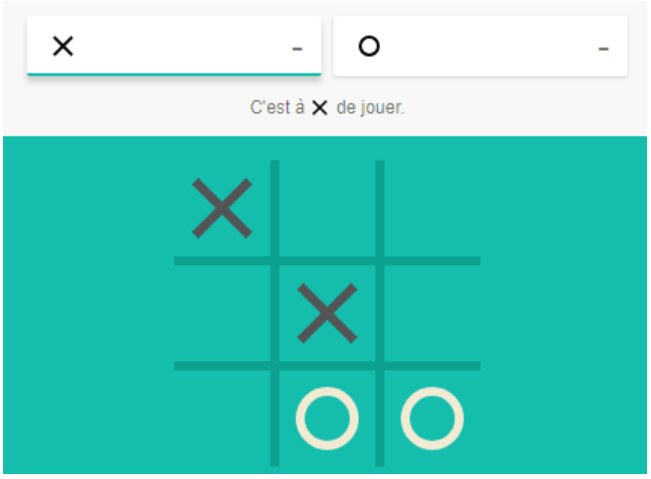
Rapport de Projet : Tic Tac Toe

Grégoire Roumache - Alexandre Radoux
Joachim Paquay - Thomas Chavet

Avril 2018

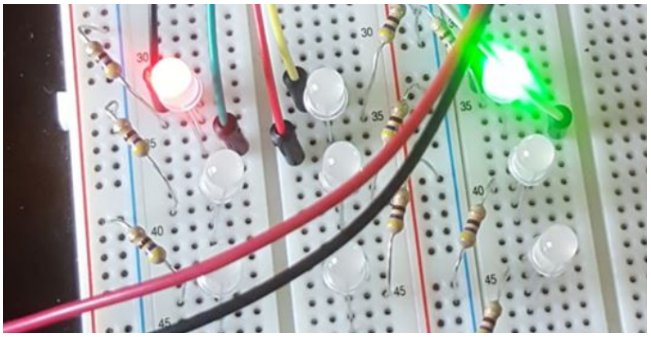
1 Introduction

Nous avons décidé de faire le jeu tic tac toe (aussi appelé jeu du morpion) pour notre projet de Digital Electronics. Les deux joueurs doivent essayer d'aligner leur symbole respectif pour gagner sur une grille de 9 cases.

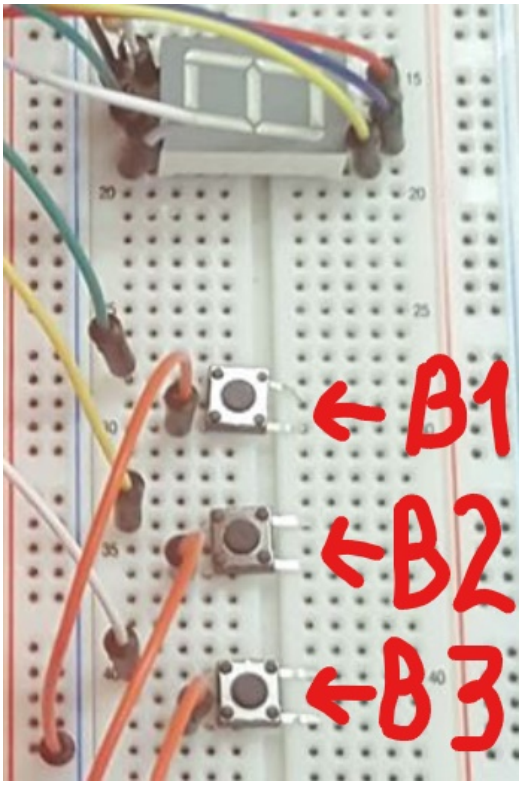


2 Courte Présentation des Résultats Pratiques

Pour adapter ce jeu à une forme électronique, nous avons utiliser des LED pour représenter chaque case et on utilise des couleurs au lieu des croix et des cercles.



Pour savoir qui doit jouer, nous avons mis une LED supplémentaire qui indique si c'est au joueur rouge ou au vert de jouer. Ensuite, le joueur peut allumer la LED qu'il a sélectionné en appuyant sur le bouton B2.



Mais comment sélectionner cette LED ? En fait, en se servant du bouton B1, on peut incrémenter un compteur qui va de 0 à 8 et représente toutes les LED. Pour savoir où en est ce compteur, nous avons placé un afficheur 7-segments qui affiche la valeur de ce compteur.

Numérotation des LED :

0	**1**	**2**
3	**4**	**5**
6	**7**	**8**

Quant au bouton B3, il sert à lancer une nouvelle partie en cas d'égalité. Mais dans le cas d'une victoire, toutes les LED s'allument de la même couleur (celle du gagnant) pour montrer qui a gagné.

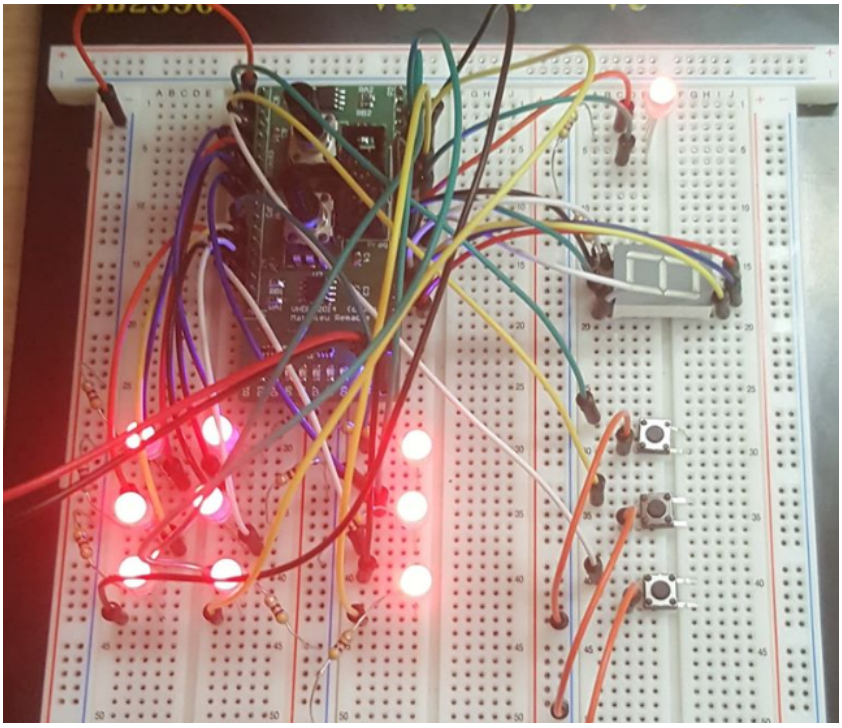
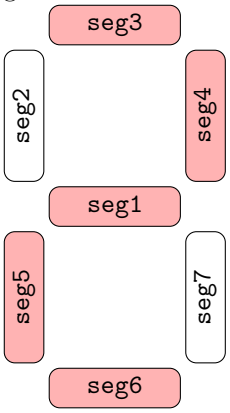


FIGURE 1 – Exemple : Victoire Rouge

3 Explication des Variables du Code VHDL

1. Comme entrées du système, nous avons :
   — Les 3 boutons `button0`, `button1` et `button3` qui correspondent respectivement aux boutons surnommés B1, B2 et B3 dans la section du rapport ci-dessus.
   — L'horloge `clk` qui sert à filtrer les signaux venant des boutons, c-à-d qu'on veut éviter que lorsqu'on appuie sur le bouton pour incrémenter le compteur (ex : compteur = 5 ⟶ compteur = 6) les rebonds n'augmente le compteur d'une valeur aléatoire (ex : compteur = 5 ⟶ compteur = 9).
2. Les sorties du systèmes sont :
   — `ledTurnGreen` & `ledTurnRed`, elles désignent quel joueur doit jouer en allumant la LED en haut à gauche de la figure 1.
   — Les vecteurs `ledGreen` & `ledRed` qui sont de taille 9 et donne leur couleur aux LED en bas à gauche de la figure 1.
   — `seg1`, `seg2`, ... , `seg7` qui allument les segments de l'afficheur 7-segments.



3. Les variables internes au système sont :
   — Les vecteurs `ledVert` & `ledRouge` qui sont associés aux vecteurs `ledGreen` & `ledRed` qui, eux, sont en sortie.

— `TourRouge` qui vaut 1 si c'est au tour du joueur rouge de jouer et qui vaut 0 si c'est au joueur vert de jouer.
— `VictoireRouge` qui vaut 1 lorsque le joueur rouge gagne et son homologue `VictoireVert` qui vaut évidemment 1 en cas de victoire du joueur vert.
— La variable `ledSelect` qui fait office de compteur pour sélectionner la LED que le joueur veut allumer.
— Les variables `SUPERcounter0`, `SUPERcounter1`, `SUPERcounter2` qui servent à éviter les rebonds. Il y en a un pour chaque bouton et elles sont expliquées plus en détail dans la section suivante.

4 Dispositif anti-rebonds

Ci-dessous se trouve le code utilisé pour empêcher les rebonds de créer des problèmes avec le compteur qui sert à sélectionner les LED.

```
1  -- Utilisation du button0 pour selectionner la led
2      if (button0 = '0') then
3        SUPERCOUNTER0 <= SUPERcounter0 + 1;
4      else
5        SUPERcounter0 <= 0;
6      end if;
7
8      if (SUPERcounter0 = 250) then
9
10       ledSelect <= ledSelect + 1;
11       if (ledSelect = 9) then
12         ledSelect <= 0;
13       end if;
14
15     end if;
```

À l'origine, nous avions remarqué un problème : lorsque le joueur voulait choisir la LED à allumer, il ne pouvait pas la sélectionner correctement car le compteur augmentait de manière aléatoire... Pour résoudre ce problème, nous avons utilisé l'horloge interne de la CPLD (`clk`). À chaque fois que la condition `rising_edge( clk )` est vraie, le code va vérifier si le joueur appuie sur le boutton B1 (= variable interne `button0`) ce qui fait que `button0` = '0'. Si c'est le cas, alors la variable `SUPERcounter0` est incrémentée, et lorsque `SUPERcounter0` vaut 250, on estime qu'il n'est pas possible que ça soit un hasard dû aux rebonds mais que l'utilisateur veut vraiment augmenter le compteur (c-à-d incrémenter `ledSelect`).

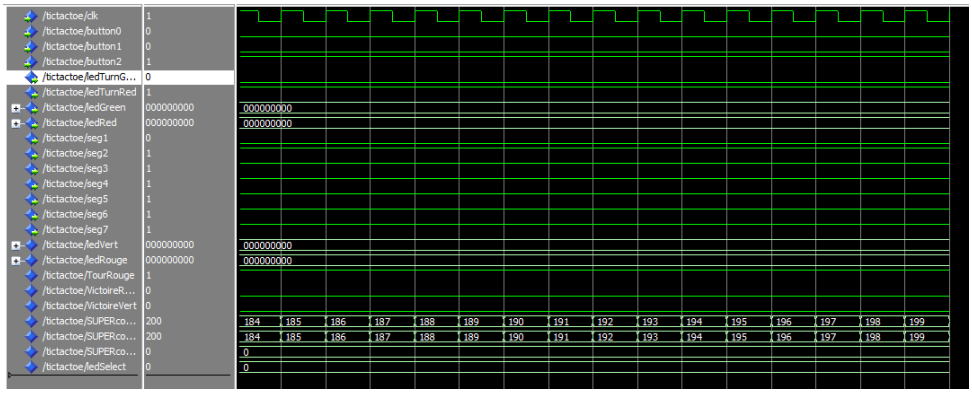
Si la variable `SUPERcounter0` n'atteint pas la valeur 250 et que le bouton prend la valeur 1 alors, le programme "comprends" que c'est dû à un/des rebonds et qu'il ne faut en fait pas augmenter le compteur. La variable `SUPERcounter0` est donc remise à 0.

Le seul inconvénient de cette approche est qu'il faut appuyer un certain temps sur un bouton pour que le programme le prenne en compte. Ce temps d'attente est réglable grâce au potentiomètre présent sur la CPLD. Dans notre cas, le temps d'attente va d'environ 0.5 sec à 5 sec.

L'avantage de cette approche est la très faible probabilité que les rebonds enclenchent l'activation d'un bouton. En effet, puisqu'il faut que le compteur (`SUPERcounter0`) augmente jusqu'à 250, il faudrait que `button0` prenne la valeur '1' 250 fois d'affilée. Or, il y a 50% de chance qu'elle prenne cette valeur (observation expérimentale) et donc, Il y a une chance sur $2^{250} \simeq 10^{75}$ que cela arrive.

5 Tests

Voici un des tests que nous avons effectués.



Annexes

A. Code VHDL de Tic Tac Toe

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3
4  entity TicTacToe is
5    port
6    (
7      -- Input ports
8      clk   : in std_logic;
9
10     button0 : in std_logic;
11     button1 : in std_logic;
12     button2 : in std_logic;
13
14     -- Output ports
15     ledTurnGreen : buffer std_logic;
16     ledTurnRed   : buffer std_logic;
17
18     ledGreen : out std_logic_vector (8 downto 0);
19     ledRed : out std_logic_vector (8 downto 0);
20
21     seg1  : buffer std_logic := '0'; -- seg = segment display
22     seg2  : buffer std_logic := '1';
23     seg3  : buffer std_logic := '1';
24     seg4  : buffer std_logic := '1';
25     seg5  : buffer std_logic := '1';
26     seg6  : buffer std_logic := '1';
27     seg7  : buffer std_logic := '1'
28
29   );
30 end entity TicTacToe;
31
32
33
34 architecture TTT_arch of TicTacToe is
35
36   signal ledVert   : std_logic_vector (8 downto 0) := "000000000";
37   signal ledRouge : std_logic_vector (8 downto 0) := "000000000";
```

```
38
39    signal TourRouge : std_logic := '1';
40
41    signal VictoireRouge : std_logic := '0';
42    signal VictoireVert : std_logic := '0';
43
44    -- Signaux anti-rebond (1 pour chaque bouton)
45    signal SUPERcounter0 : integer range 0 to 500 := 0;
46    signal SUPERcounter1 : integer range 0 to 500 := 0;
47    signal SUPERcounter2 : integer range 0 to 500 := 0;
48
49    signal ledSelect : integer range 0 to 9 := 0;
50
51
52
53
54
55  begin -- architecture TTT_arch
56
57    process(clk)
58
59    begin -- process
60
61      -- Assignement des signaux aux sorties
62      ledGreen  <= ledVert;
63      ledRed    <= ledRouge;
64      ledTurnRed  <= TourRouge;
65      ledTurnGreen <= not(TourRouge);
66
67
68      if( rising_edge(clk)) then
69
70
71
72
73    -- Utilisation du button0 pour selectionner la led
74        if(button0 = '0') then
75          SUPERCOUNTER0 <= SUPERcounter0 + 1;
76        else
77          SUPERcounter0 <= 0;
78        end if;
79
80        if(SUPERcounter0 = 250) then
81
82          ledSelect <= ledSelect + 1;
83          if(ledSelect = 9) then
84            ledSelect <= 0;
85          end if;
86
87        end if;
88
89
90
91    -- Utilisation du button1 pour confirmer la led
92        if(button1 = '0') then
93          SUPERcounter1 <= SUPERcounter1 + 1;
94        else
95          SUPERcounter1 <= 0;
96        end if;
97
98        if(SUPERcounter1 = 250) then
```

```
99
100         if (ledSelect = 0 and ledVert(0) = '0' and ledRouge(0) = '0') then
101           if (TourRouge = '1') then
102             TourRouge  <= '0';
103             ledRouge(0) <= '1';
104           else
105             TourRouge  <= '1';
106             ledVert(0)  <= '1';
107           end if;
108
109         elsif (ledSelect = 1 and ledVert(1) = '0' and ledRouge(1) = '0') then
110           if (TourRouge = '1') then
111             TourRouge  <= '0';
112             ledRouge(1) <= '1';
113           else
114             TourRouge  <= '1';
115             ledVert(1)  <= '1';
116           end if;
117
118         elsif (ledSelect = 2 and ledVert(2) = '0' and ledRouge(2) = '0') then
119           if (TourRouge = '1') then
120             TourRouge  <= '0';
121             ledRouge(2) <= '1';
122           else
123             TourRouge  <= '1';
124             ledVert(2)  <= '1';
125           end if;
126
127         elsif (ledSelect = 3 and ledVert(3) = '0' and ledRouge(3) = '0') then
128           if (TourRouge = '1') then
129             TourRouge  <= '0';
130             ledRouge(3) <= '1';
131           else
132             TourRouge  <= '1';
133             ledVert(3)  <= '1';
134           end if;
135
136         elsif (ledSelect = 4 and ledVert(4) = '0' and ledRouge(4) = '0') then
137           if (TourRouge = '1') then
138             TourRouge  <= '0';
139             ledRouge(4) <= '1';
140           else
141             TourRouge  <= '1';
142             ledVert(4)  <= '1';
143           end if;
144
145         elsif (ledSelect = 5 and ledVert(5) = '0' and ledRouge(5) = '0') then
146           if (TourRouge = '1') then
147             TourRouge  <= '0';
148             ledRouge(5) <= '1';
149           else
150             TourRouge  <= '1';
151             ledVert(5)  <= '1';
152           end if;
153
154         elsif (ledSelect = 6 and ledVert(6) = '0' and ledRouge(6) = '0') then
155           if (TourRouge = '1') then
156             TourRouge  <= '0';
157             ledRouge(6) <= '1';
158           else
159             TourRouge  <= '1';
```

```
160             ledVert(6)  <= '1';
161           end if;
162
163         elsif (ledSelect = 7 and ledVert(7) = '0' and ledRouge(7) = '0') then
164           if (TourRouge = '1') then
165             TourRouge  <= '0';
166             ledRouge(7) <= '1';
167           else
168             TourRouge  <= '1';
169             ledVert(7)  <= '1';
170           end if;
171
172         elsif (ledSelect = 8 and ledVert(8) = '0' and ledRouge(8) = '0') then
173           if (TourRouge = '1') then
174             TourRouge  <= '0';
175             ledRouge(8) <= '1';
176           else
177             TourRouge  <= '1';
178             ledVert(8)  <= '1';
179           end if;
180
181       end if;
182
183     end if;
184
185
186
187   -- Utilisation du button2 pour commencer une nouvelle partie
188     if (button2 = '0') then
189       SUPERcounter2 <= SUPERcounter2 + 1;
190     else
191       SUPERcounter2 <= 0;
192     end if;
193
194     if (SUPERcounter2 = 250) then
195
196       TourRouge <= '1';
197       ledSelect <= 0;
198
199       VictoireRouge <= '0';
200       VictoireVert <= '0';
201
202       for index in 0 to 8 loop
203         ledVert(index)  <= '0';
204         ledRouge(index) <= '0';
205       end loop;
206
207     end if;
208
209
210
211
212
213   -- Check si 3 led de meme couleur sont alignees
214     -- horizontal VERT
215       if  (ledVert(0) = '1' AND ledVert(1) = '1' AND ledVert(2) = '1') then
216         VictoireVert <= '1';
217       elsif  (ledVert(3) = '1' AND ledVert(4) = '1' AND ledVert(5) = '1') then
218         VictoireVert <= '1';
219       elsif  (ledVert(6) = '1' AND ledVert(7) = '1' AND ledVert(8) = '1') then
220         VictoireVert <= '1';
```

```
221       -- vertical VERT
222         elsif (ledVert(0) = '1' AND ledVert(3) = '1' AND ledVert(6) = '1') then
223           VictoireVert <= '1';
224         elsif (ledVert(1) = '1' AND ledVert(4) = '1' AND ledVert(7) = '1') then
225           VictoireVert <= '1';
226         elsif (ledVert(2) = '1' AND ledVert(5) = '1' AND ledVert(8) = '1') then
227           VictoireVert <= '1';
228       -- diagonal VERT
229         elsif (ledVert(0) = '1' AND ledVert(4) = '1' AND ledVert(8) = '1') then
230           VictoireVert <= '1';
231         elsif (ledVert(2) = '1' AND ledVert(4) = '1' AND ledVert(6) = '1') then
232           VictoireVert <= '1';
233         end if;
234
235       -- horizontal ROUGE
236         if (ledRouge(0) = '1' AND ledRouge(1) = '1' AND ledRouge(2) = '1') then
237           VictoireRouge <= '1';
238         elsif (ledRouge(3) = '1' AND ledRouge(4) = '1' AND ledRouge(5) = '1') then
239           VictoireRouge <= '1';
240         elsif (ledRouge(6) = '1' AND ledRouge(7) = '1' AND ledRouge(8) = '1') then
241           VictoireRouge <= '1';
242       -- vertical ROUGE
243         elsif (ledRouge(0) = '1' AND ledRouge(3) = '1' AND ledRouge(6) = '1') then
244           VictoireRouge <= '1';
245         elsif (ledRouge(1) = '1' AND ledRouge(4) = '1' AND ledRouge(7) = '1') then
246           VictoireRouge <= '1';
247         elsif (ledRouge(2) = '1' AND ledRouge(5) = '1' AND ledRouge(8) = '1') then
248           VictoireRouge <= '1';
249       -- diagonal ROUGE
250         elsif (ledRouge(0) = '1' AND ledRouge(4) = '1' AND ledRouge(8) = '1') then
251           VictoireRouge <= '1';
252         elsif (ledRouge(2) = '1' AND ledRouge(4) = '1' AND ledRouge(6) = '1') then
253           VictoireRouge <= '1';
254         end if;
255
256
257
258
259     -- Allume l'afficheur 7-segements (en fonction du compteur)
260     if (ledSelect = 0) then
261       seg1 <= '0';
262       seg2 <= '1';
263       seg3 <= '1';
264       seg4 <= '1';
265       seg5 <= '1';
266       seg6 <= '1';
267       seg7 <= '1';
268
269     elsif (ledSelect = 1) then
270       seg1 <= '0';
271       seg2 <= '0';
272       seg3 <= '0';
273       seg4 <= '1';
274       seg5 <= '0';
275       seg6 <= '0';
276       seg7 <= '1';
277
278     elsif (ledSelect = 2) then
279       seg1 <= '1';
280       seg2 <= '0';
281       seg3 <= '1';
```

```
282       seg4 <= '1';
283       seg5 <= '1';
284       seg6 <= '1';
285       seg7 <= '0';
286
287     elsif (ledSelect = 3) then
288       seg1 <= '1';
289       seg2 <= '0';
290       seg3 <= '1';
291       seg4 <= '1';
292       seg5 <= '0';
293       seg6 <= '1';
294       seg7 <= '1';
295
296     elsif (ledSelect = 4) then
297       seg1 <= '1';
298       seg2 <= '1';
299       seg3 <= '0';
300       seg4 <= '1';
301       seg5 <= '0';
302       seg6 <= '0';
303       seg7 <= '1';
304
305     elsif (ledSelect = 5) then
306       seg1 <= '1';
307       seg2 <= '1';
308       seg3 <= '1';
309       seg4 <= '0';
310       seg5 <= '0';
311       seg6 <= '1';
312       seg7 <= '1';
313
314     elsif (ledselect = 6) then
315       seg1 <= '1';
316       seg2 <= '1';
317       seg3 <= '1';
318       seg4 <= '0';
319       seg5 <= '1';
320       seg6 <= '1';
321       seg7 <= '1';
322
323     elsif (ledSelect = 7) then
324       seg1 <= '0';
325       seg2 <= '0';
326       seg3 <= '1';
327       seg4 <= '1';
328       seg5 <= '0';
329       seg6 <= '0';
330       seg7 <= '1';
331
332     elsif (ledSelect = 8) then
333       seg1 <= '1';
334       seg2 <= '1';
335       seg3 <= '1';
336       seg4 <= '1';
337       seg5 <= '1';
338       seg6 <= '1';
339       seg7 <= '1';
340   end if;
341
342
```

```
343
344
345
346     -- En cas de victoire
347     if(VictoireVert = '1') then
348
349       seg1 <= '0';
350       seg2 <= '0';
351       seg3 <= '0';
352       seg4 <= '0';
353       seg5 <= '0';
354       seg6 <= '0';
355       seg7 <= '0';
356
357       for index in 0 to 8 loop
358         ledVert(index)  <= '1';
359         ledRouge(index) <= '0';
360       end loop;
361
362       TourRouge <= '0';
363
364     end if;
365
366     if(VictoireRouge = '1') then
367
368       seg1 <= '0';
369       seg2 <= '0';
370       seg3 <= '0';
371       seg4 <= '0';
372       seg5 <= '0';
373       seg6 <= '0';
374       seg7 <= '0';
375
376       for index in 0 to 8 loop
377         ledVert(index)  <= '0';
378         ledRouge(index) <= '1';
379       end loop;
380
381       TourRouge <= '1';
382
383     end if;
384
385
386
387       end if; -- end if ( rising_edge( clk ) )
388     end process;
389   end architecture TTT_arch;
```