

UNIVERSITÉ DE BRETAGNE OCCIDENTALE

MASTER 2 INFORMATIQUE
DÉPARTEMENT INFORMATIQUE

2020/2021

SYSTÈME ON-CHIP

Détection de dépassement de temps d'exécution

Auteur :
William PENSEC

Auteur :
Timothé LANNUZEL

22 janvier 2021



Sommaire

I Introduction

L'objectif de ce projet est de concevoir en VHDL un moniteur de temps d'exécution de tâches sur un processeur. En effet, sur un système temps réel, il est très important que les contraintes de temps soient respectées afin d'éviter tout problèmes. Le composant doit suivre l'exécution de chaque tâche et envoyer un signal d'interruption au processeur si l'une d'entre elles dépasse son échéance. La capacité maximale d'une tâche s'appelle le Worst Case Execution Time (WCET). En connaissant cette valeur, on sait si le processeur peut gérer le système ou s'il est nécessaire de le changer pour quelque chose de plus performant.

II Conception VHDL

Le projet s'est découpé en plusieurs étapes qui ont été de créer d'abord les différents modules qui composent le système puis de créer les fichiers de tests (testbench) de ces modules. La seconde étape est de regrouper ces modules afin de créer une IP sous Vivado qui pourra être utilisée ailleurs. Cette IP sera composée du CPU, d'une mémoire, de compteurs et d'un composant permettant la communication avec le CPU par l'AXI.

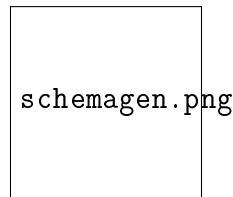


FIGURE 1 – Architecture générale

II.1 Chronomètre

L'image ??, à la page ??, représente le fonctionnement de manière schématique du module chronomètre-décompteur. Le code de cette partie est disponible dans l'archive ou sinon voir le code ?? à la page ??. Le chronomètre est lié à une horloge sur front montant `rising_edge(clk)`. Cela permet de contrôler les opérations un front sur deux pour aller un peu plus lentement. Autrement, il y a un port de démarrage/arrêt du chronomètre `startStop` qui permet comme son nom l'indique de démarrer ou stopper le module ; mais également un port afin de mettre en pause et de reprendre le timer `suspendResume`. Nous avons inclus un port de chargement `load` et de reset `reset` permettant de charger la valeur d'initialisation (valeur qui correspond à la durée du timer par exemple `<10>` périodes d'horloge) ou au contraire de mettre à 0 le timer de la tâche en cours.

Enfin, le dernier port qui est celui qui nous int resse le plus est celui du `wcet`. Ce port est donc un tableau de 16 bits. C'est dans ce tableau que l'on va enregistrer la valeur du `Worst Case Execution Time (WCET)`. C'est cette valeur qui sera charg e par le port `load` en m moire et c'est cette valeur qui servira   d compter le temps avant d'envoyer si besoin l'interruption au processeur si le `WCET` arrive   0 dans le timer.

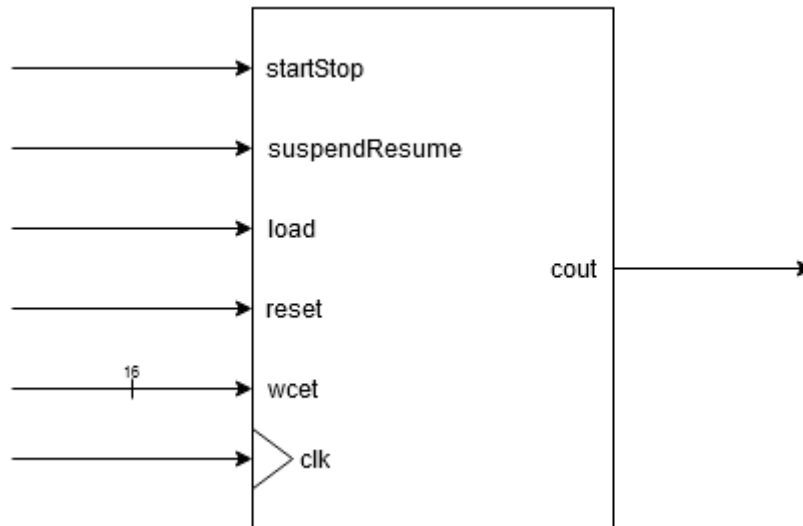


FIGURE 2 – Bloc chronom tre

II.2 TestBench Chronom tre

Le code du test bench est le code ??   la page ?. Il s'articule de la mani re suivante : tout d'abord comme d'habitude nous appelons le composant   qui il fait r f rence, c'est   dire le chronom tre. Puis, on cr e les signaux n cessaires pour assigner des valeurs aux ports du composant. Ensuite dans l'architecture comportementale du composant test, on affecte des valeurs aux signaux. Nous avons d cid  de faire une horloge avec une p riode de 1 ns afin d'avoir quelque chose de rapide. La valeur `startStop_ch` est initialis e   0 et passe   1 apr s 5 ns c'est   dire que le chronom tre ne d marrera qu'apr s 5 ns d'ex cution de programme, cette valeur a  t  mise seulement dans un but de test mais en soit doit  tre initialis e   1 lors de la cr ation du chronom tre. La valeur `load_ch` correspondant au chargement du `WCET` en m moire ; il est initialis    1, c'est   dire qu'on charge en m moire la donn e d s qu'elle est disponible puis on passe cette valeur   0 car on d sire arr ter la mise en m moire de la valeur afin de passer   la d cr mentation. La valeur du `reset` est laiss    0 car nous n'en avons pas

besoin du tout mais si on passe cette valeur   1 alors la donn  e est mise   0 comme convenu ! Enfin, la valeur du `wcet_ch` est initialis  e   7.

Un exemple d’ex  cution est propos     l’image ??   la page ?. On distingue sur l’image toutes les  tapes cit  es dans le paragraphe pr  c  dent.

II.3 Moniteur de t  ches

II.4 TestBench Moniteur de t  ches

III R  sultats

III.1 Chronom  tre

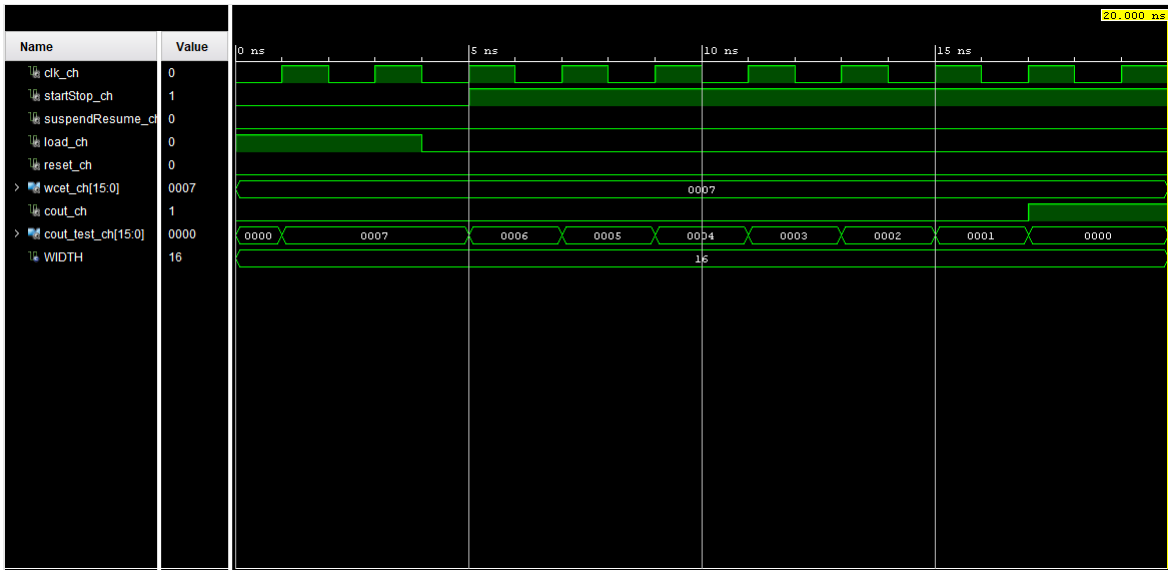


FIGURE 3 – Testbench chronom  tre

III.2 Moniteur de tâches

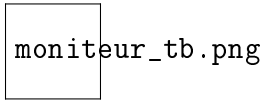


FIGURE 4 – Testbench moniteur de tâches

IV Code

IV.1 Chronomètre

```
1
2 — Engineer: Timothé LANNUZEL & William PENSEC
3 — Create Date: 03.01.2020 16:01:00
4 — Module Name: chronometer – Behavioral
5 — Project Name: Détection de dépassement de temps d'exécution
6 — Revision: 1.2
7
8
9
10 library IEEE;
11 use IEEE.STD_LOGIC_1164.ALL;
12 use IEEE.STD_LOGIC_UNSIGNED.ALL;
13 use IEEE.NUMERIC_STD.ALL;
14
15 entity chronometer is
16     generic(
17         WIDTH : integer := 16
18     );
19
20     Port (
21         clk : in std_logic;
22         startStop : in std_logic;
23         suspendResume : in std_logic;
24         load : in std_logic;
25         reset : in std_logic;
26         wcet : in std_logic_vector(WIDTH - 1 downto 0);
27         cout : out std_logic;
28         cout_test : out std_logic_vector(WIDTH - 1 downto 0)
29     );
30 end chronometer;
31
```

```

32 architecture Behavioral of chronometer is
33     signal curr_value : std_logic_vector(WIDTH - 1 downto 0) := (
34         others => '0');
35     begin
36         cout_test <= curr_value;
37         compteur : process(clk)
38         begin
39             if rising_edge(clk) then
40                 if load = '1' then
41                     curr_value <= wcet;
42                 elsif reset = '1' then
43                     curr_value <= (others => '0');
44                 elsif startStop = '1' and suspendResume = '0' and
45                     unsigned(curr_value) /= 0 then — 1 start | 0 resume
46                     curr_value <= std_logic_vector(unsigned(curr_value) -
47                         1);
48                 end if;
49             end if;
50         end process;
51         test : process(curr_value)
52         begin
53             if startStop = '1' and unsigned(curr_value) = 0 then
54                 cout <= '1';
55             else
56                 cout <= '0';
57             end if;
58             if reset = '1' then
59                 cout <= '0';
60             elsif load = '1' then
61                 cout <= '0';
62             end if;
63         end process;
64     end Behavioral;
65 end Behavioral;

```

Listing 1 – Chronom tre

IV.2 TestBench Chronom tre

```

1
2 — Engineer: Timoth  LANNUZEL & William PENSEC
3 — Create Date: 05.01.2020 17:21:00
4 — Module Name: chronometer_tb – Behavioral
5 — Project Name: D tection de d passement de temps d'ex cution

```

```

6  — Revision: 1.0
7  —
8
9
10 library IEEE;
11 use IEEE.STD_LOGIC_1164.ALL;
12 use IEEE.STD_LOGIC_UNSIGNED.ALL;
13 use IEEE.NUMERIC_STD.ALL;
14
15 entity chronometer_tb is
16 — Port ( );
17 end chronometer_tb;
18
19 architecture Behavioral of chronometer_tb is
20     component chronometer is
21         generic(
22             WIDTH : integer := 16
23         );
24         Port(
25             clk : in std_logic;
26             startStop : in std_logic;
27             suspendResume : in std_logic;
28             load : in std_logic;
29             reset : in std_logic;
30             wcet : in std_logic_vector(WIDTH - 1 downto 0);
31             cout : out std_logic;
32             cout_test : out std_logic_vector(WIDTH - 1 downto 0)
33         );
34     end component chronometer;
35
36     constant WIDTH : integer := 16;
37     signal clk_ch : std_logic := '0';
38     signal startStop_ch : std_logic;
39     signal suspendResume_ch : std_logic;
40     signal load_ch : std_logic;
41     signal reset_ch : std_logic;
42     signal wcet_ch : std_logic_vector(WIDTH - 1 downto 0);
43     signal cout_ch : std_logic;
44     signal cout_test_ch : std_logic_vector(WIDTH - 1 downto 0);
45
46 begin
47     clk_ch <= not clk_ch after 1 ns;
48     startStop_ch <= '0', '1' after 5 ns;
49     suspendResume_ch <= '0'; — always active
50     load_ch <= '1', '0' after 4 ns; — 0 to stop data's loading
51     reset_ch <= '0'; — never reseted
52     wcet_ch <= std_logic_vector(to_unsigned(7, 16));
53
54     iut : entity work.chronometer(Behavioral)

```



```
55     Port map(  
56         clk => clk_ch ,  
57         startStop => startStop_ch ,  
58         suspendResume => suspendResume_ch ,  
59         load => load_ch ,  
60         reset => reset_ch ,  
61         wcet => wcet_ch ,  
62         cout => cout_ch ,  
63         cout_test => cout_test_ch  
64     );  
65 end Behavioral;
```

Listing 2 – TestBench chronomètre

IV.3 Moniteur de tâches

```
1  
2 — Engineer: Timothé LANNUZEL & William PENSEC  
3 — Create Date: 10.12.2020 16:22:12  
4 — Module Name: taskMonitor – Behavioral  
5 — Project Name: Détection de dépassement de temps d'exécution  
6 — Revision: 1.2  
7  
8  
9  
10 library IEEE;  
11 use IEEE.STD_LOGIC_1164.ALL;  
12 use IEEE.STD_LOGIC_UNSIGNED.ALL;  
13 use IEEE.NUMERIC_STD.ALL;  
14  
15 entity taskMonitor is  
16     Port (  
17         clk_in : in std_logic;  
18         reset_in : in std_logic;  
19  
20         id_task : in std_logic_vector(3 downto 0);  
21         wcet_task : in std_logic_vector(15 downto 0);  
22         mess_task : in std_logic_vector(3 downto 0);  
23  
24         counter_interrupt : out std_logic;  
25         counter_interrupt_test : out std_logic_vector(16 - 1 downto 0)  
26     );  
27 end taskMonitor;  
28  
29 architecture Behavioral of taskMonitor is  
30  
31
```

```

32  -----Signal-----
33  -----
34  --signaux pour les chronos
35  signal sigstartStop : std_logic_vector(3 downto 0);
36  signal sigsuspendResume : std_logic_vector(3 downto 0);
37  signal sigload : std_logic_vector(3 downto 0);
38  type register_array is array ( 3 downto 0 ) of std_logic_vector(
15  15 downto 0 );
39  signal sigtaskwcet : register_array;
40
41  signal curChrono: std_logic_vector(3 downto 0) := (others => '0')
; -- pour connaitre sur quelle chrono on est
42  signal interrupt_timer : std_logic_vector(3 downto 0);
43
44  signal curcounter_interrupt_test : register_array;
45  signal currTaskId : integer :=0;
46  signal sigReset : std_logic_vector(3 downto 0);
47
48  -----
49  -----Components-----
50  -----
51  component chronometer
52  generic(
53      WIDTH : integer := 16
54  );
55  Port (
56      clk : in std_logic;
57      startStop : in std_logic;
58      suspendResume : in std_logic;
59      load : in std_logic;
60      reset : in std_logic;
61      wcet : in std_logic_vector(WIDTH - 1 downto 0);
62      cout : out std_logic;
63      cout_test : out std_logic_vector(WIDTH - 1 downto 0)
64  );
65  end component;
66
67  begin
68
69  -----Port map-----
70  -----
71  generate_chrono : for i in 0 to 3 generate
72      instChrono : entity work.chronometer(Behavioral)
73      port map(
74          clk => clk_in,
75          startStop => sigstartStop(i),
76          suspendResume => sigsuspendResume(i),
77          load => sigload(i),
78          reset => sigReset(i),

```

```

79         wcet => sigtaskwcet(i),
80         cout => interrupt_timer(i),
81         cout_test => curcounter_interrupt_test(i)
82     );
83 end generate;
84
85
86 tache : process(clk_in)
87 begin
88     --Changement du WCET
89     if reset_in = '1' then
90         loop1 : for i in 0 to 3 LOOP
91             sigstartStop(i) <= '0';    --STOP
92             sigload(i) <= '0';
93             sigsuspendResume(i) <= '0';
94             sigReset(i) <= '1';
95         END LOOP loop1;
96     else
97         case conv_integer ( mess_task ) is    --Changement
de message
98             when 0 =>
99                 sigstartStop(currTaskId) <= '0';    --STOP
100                 sigload(currTaskId) <= '0';
101                 sigsuspendResume(currTaskId) <= '0';
102                 sigReset(currTaskId) <= '0';
103             when 1 =>
104                 sigstartStop(currTaskId) <= '1';    --START
105                 sigload(currTaskId) <= '0';
106             when 2 =>
107                 sigtaskwcet(currTaskId) <= wcet_task;
108                 sigload(currTaskId) <= '1';    --LOAD
109             when 3 =>
110                 sigsuspendResume(currTaskId) <= '1';--SUSPEND
111             when 4 =>
112                 sigsuspendResume(currTaskId) <= '0';--RESUME
113             when 5 =>
114                 sigReset(currTaskId) <= '1';    --RESET
115                 sigstartStop(currTaskId) <= '0';
116                 sigload(currTaskId) <= '0';
117                 sigsuspendResume(currTaskId) <= '0';
118             when others =>
119                 --ne rien faire
120         end case;
121     end if;
122     if reset_in = '1' then
123         counter_interrupt <= '0';
124     elsif interrupt_timer(currTaskId) = '1' then
125         counter_interrupt <= '1';
126     else

```

```
127         counter_interrupt <= '0';
128     end if;
129     counter_interrupt_test <= curcounter_interrupt_test(currTaskId)
;
130 end process;
131
132 tache2 : process(id_task)
133 begin
134     currTaskId <= to_integer (unsigned(id_task));
135
136 end process;
137
138 end Behavioral;
```

Listing 3 – Moniteur de tâches

IV.4 TestBench Moniteur de tâches

```
1
2 — Engineer: Timothé LANNUZEL & William PENSEC
3 — Create Date: 19.01.2021 12:09:40
4 — Module Name: taskMonitor_tb – Behavioral
5 — Project Name: Détection de dépassement de temps d'exécution
6 — Revision: 1.2
7
8
9
10 library IEEE;
11 use IEEE.STD_LOGIC_1164.ALL;
12 use IEEE.STD_LOGIC_UNSIGNED.ALL;
13 use IEEE.NUMERIC_STD.ALL;
14
15 entity taskMonitor_tb is
16 — Port ( );
17 end taskMonitor_tb;
18
19 architecture Behavioral of taskMonitor_tb is
20     component taskMonitor is
21         Port (
22             clk_in : in std_logic;
23             reset_in : in std_logic;
24
25             id_task : in std_logic_vector(3 downto 0);
26             wcet_task : in std_logic_vector(15 downto 0);
27             mess_task : in std_logic_vector(3 downto 0);
28
29             counter_interrupt : out std_logic;
```

```

30  RAJOUTER
31      counter_interrupt_test: out std_logic_vector(16 - 1 downto
    0)
32  RAJOUTER
33
34  );
35  end component taskMonitor;
36
37  signal clk : std_logic := '0';
38  signal reset : std_logic;
39  signal id : std_logic_vector(3 downto 0);
40  signal wcet : std_logic_vector(15 downto 0);
41  signal message : std_logic_vector(3 downto 0);
42  signal compteur : std_logic;
43  RAJOUTER
44  signal compteur_test: std_logic_vector(15 downto 0);
45  RAJOUTER
46
47  begin
48      clk <= not clk after 1 ns;
49      reset <= '0', '1' after 29 ns;
50      id <= std_logic_vector(to_unsigned(0,4));
51      tache 0
52          --std_logic_vector(to_unsigned(1,4)) after 7 ns ;
53      tache 1
54          wcet <= std_logic_vector(to_unsigned(3, 16)),
55          tache 0: 3
56          std_logic_vector(to_unsigned(5, 16)) after 7 ns;
57      tache 1: 5
58          message <= std_logic_vector(to_unsigned(0,4)),
59          stop
60          std_logic_vector(to_unsigned(2,4)) after 2 ns,
61      load
62          std_logic_vector(to_unsigned(1,4)) after 3 ns;
63      start
64          -- std_logic_vector(to_unsigned(3,4)) after 5 ns,
65      suspend
66          --std_logic_vector(to_unsigned(0,4)) after 7 ns,
67      stop
68          --std_logic_vector(to_unsigned(2,4)) after 8 ns,
69      load
70          --std_logic_vector(to_unsigned(1,4)) after 9 ns ;
71      start
72
73  iut : entity work.taskMonitor(Behavioral)
74  Port map(
75      clk_in => clk,
76      reset_in => reset,
77      id_task => id,

```

```
67         wcet_task => wcet ,
68         mess_task => message ,
69         counter_interrupt => compteur ,
70 RAJOUTER
71         counter_interrupt_test =>compteur_test
72 RAJOUTER
73
74     );
75
76 end Behavioral;
```

Listing 4 – TestBench moniteur de tâches

V Continuité du projet