

Implantation d'un pic dans un FPGA

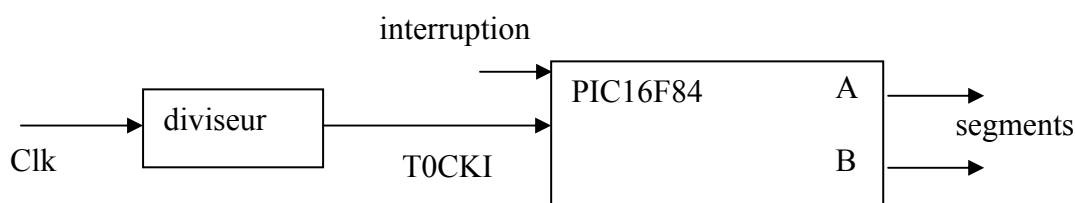
Le but du TP est d'implanter un PIC16F84 dans un FPGA. L'objectif est d'acquérir une sensibilisation au problème du Co-Design. Essayer de cerner les difficultés d'un tel développement mais aussi l'énorme avantage qui peut en résulter.

1 Présentation du TP

Afin de tester le plus complètement qui soit le modèle VHDL du PIC, nous considérerons une application très simple de **chenillard à diodes** dont les temporisations sont effectuées par **interruption timer**.

1.1 Description VHDL

Le circuit d'application est donc le PIC auquel on se contente d'ajouter un diviseur d'horloge relié à l'entrée T0CKI. Le timer sera programmé en mode compteur. Le fichier *pic.vhd* décrit au plus haut niveau de hiérarchie cette application.



```

LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;
USE IEEE.numeric_std.ALL;

ENTITY pic IS
  GENERIC (
    simulation : boolean := false); -- change les constantes de temps
  -- réduit la durée du diviseur en simulation
  PORT (
    clk      : IN  std_logic;    -- horloge systeme
    reset_n  : IN  std_logic;    -- reset general
    int      : IN  std_logic;    -- demande interruption
    port_a   : INOUT std_logic_vector(7 DOWNTO 0);
    port_b   : INOUT std_logic_vector(7 DOWNTO 0) );
END pic;
  
```

```

ARCHITECTURE structure OF pic IS

COMPONENT P16F84 -- http://www.opencores.org
  GENERIC(
    SyncReset : boolean := true);
  PORT(
    Clk : IN std_logic;
    Reset_n : IN std_logic;
    T0CKI : IN std_logic;
    INT : IN std_logic;
    Port_A : INOUT std_logic_vector(7 DOWNTO 0);
    Port_B : INOUT std_logic_vector(7 DOWNTO 0)
  );
END COMPONENT;

SIGNAL t0cki : std_logic;          -- entrée externe du timer

BEGIN -- structure

P1 : P16F84 PORT MAP (
  clk  => clk,
  reset_n => reset_n,
  T0CKI => t0cki,
  int  => int,
  port_a => port_a,
  port_b => port_b);

-- l'horloge de la carte étant trop rapide, il faut la diviser
-----
diviseur: PROCESS
  CONSTANT rapport : natural := 749; -- si 50 Mhz alors 15 us
  CONSTANT rapport_s : natural := 9; -- pour simu
  VARIABLE c : natural RANGE 0 TO rapport;
BEGIN -- PROCESS diviseur
  WAIT UNTIL rising_edge(clk);
  t0cki <= '0';
  IF simulation THEN
    IF c < rapport_s THEN          -- simulation
      c := c + 1;
    ELSE
      c := 0;
      t0cki <= '1';
    END IF;
  ELSE
    IF c < rapport THEN          -- synthèse
      c := c + 1;
    ELSE
      c := 0;
      t0cki <= '1';
    END IF;
  END IF;
END PROCESS diviseur;

END structure;

```

1.2 Programme d'application

Afin de pouvoir simplifier la vision du code que l'on aura au cours de la simulation, on a écrit le programme en assembleur. Cependant, pour la compréhension rapide du programme, voici son équivalent en langage C (chemin `..\Tp_picvhdl\prog\chen_sim.c`).

```
#include <pic1684.h>
void interrupt decalage(void);

main(void)
{
    TRISA = 0xFF; // 8 entrées pour A
    TRISB = 0x00; // 8 sorties pour B
    OPTION = 0x20; // prescaler 2 , mode compteur
    INTCON = 0xA0; // autorise l'interruption timer
    PORTB = 0xFE; // une seule diode allumée
    TMR0 = 0xFE ; // le minimum pour la simulation, le maximum autrement
    while(1)
    {
        // on ne fait rien que recopier sur 2 segments la valeur de SW1
        if((PORTA & 0x01) == 1) PORTA = 0x06;
    }
}

void interrupt decalage(void)
{
    TMR0 = 0xFE; // le minimum
    PORTB = (PORTB << 1) | 1 ;
    if (PORTB == 0xFF) PORTB = 0xFE;
    TOIF = 0; // acquittement interruption
}
```

Après avoir initialisé correctement le PIC en particulier son **timer** et l'interruption associée, le programme principal est une **boucle vide**. L'**interruption** qui survient à intervalles réguliers sert à décaler et afficher une parmi les 8 diodes électroluminescentes.

1.3 Matériel utilisé

Pour des raisons de disponibilité mais aussi de différence avec les cartes Xilinx déjà utilisées en projet VHDL, la carte de test sera la carte de développement NIOS-STRATIX1S10 d'ALTERA. Cette carte contient les éléments suivants :

- ❑ Un circuit FPGA Stratix EP1S10F780C6
- ❑ 1MB de SRAM
- ❑ 32MB de SDRAM
- ❑ 8MB de Flash
- ❑ Interface Ethernet
- ❑ Un connecteur de port série et son translator de niveau
- ❑ Un connecteur JTAG (port parallèle) pour les fonctions de configurations et de debug matériel
- ❑ Un oscillateur, circuit d'horloge 50 MHz
- ❑ 2 afficheurs 7 segments, 8 leds et 4 boutons poussoirs

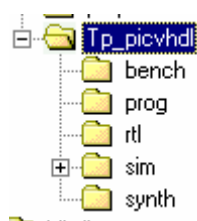


On utilisera ici le FPGA et ses entrées-sorties simples : un bouton poussoir et dix segments parmi les seize des deux afficheurs.

2 Première implantation du projet

2.1 Arborescence du projet.

Recopier chez-vous le répertoire `d:\Tp_picvhdl` et tout ce qu'il contient. L'arborescence du projet est alors la suivante :



- ❑ `prog` contient les fichiers liés au programme assembleur
- ❑ `rtl` contient l'ensemble des fichiers synthétisables
- ❑ `bench` contient les fichiers de commande pour modelsim et le fichier `test_pic.vhd` de stimuli
- ❑ `synth` contient le fichier de commandes pour le projet.
- ❑ `sim` est le répertoire de travail en simulation

2.2 Génération de la ROM

Le fichier source correct pour la simulation est *chen_sim.vhd*. Après compilation (déjà faite) il s'appelle *chen_sim.hex*. La génération de la rom est très simple.

- Ouvrir une **fenêtre DOS**
- Ce placer dans le répertoire : `d:\users\grxx\Tp_picvhdl\prog`
- Executer : **hex2rom chen_sim.hex ROM84 10114s > rom84_sim.vhd**
- Executer : **hex2rom chen.hex ROM84 10114s > rom84.vhd**

Cette dernière génération rétablira au niveau du projet des valeurs réalistes de temporisation et servira lors de la synthèse.

2.3 Quartus

Quartus est l'outil de CAO d'Altera pour ses familles de circuits à haute densité d'intégration typiquement APEX et STRATIX. Il est toujours obligatoire pour la phase purement technologique c'est à dire placement-routage et génération du fichier de configuration. Il est donc possible pour concevoir un circuit, d'avoir en tête des outils généralistes comme **hdl designer** et **Precision** et de n'utiliser Quartus que pour cette étape finale.

D'un autre côté, Quartus est aussi un outil complet permettant de gérer un projet, de saisir de la schématique ou du VHDL, d'effectuer la synthèse Vhdl, Verilog ou Hdla (langage ancien d'ALTERA), éditer un Floorplan, placer et router les cellules.

Pour des raisons de simplicité, nous fixerons ainsi notre méthodologie :

- Projet entièrement décrit en VHDL
- Création d'un projet dans Quartus
- Synthèse VHDL par Quartus
- Placement-routage et génération du fichier de configuration (*.sof)

2.3.1 Préliminaires

- S'assurer tout d'abord d'avoir bien le **fichier rom84.vhd à jour**.
- Recopier *rom84.vhd* sous `d : \users\grxx\Tp_picvhdl \rtl`
- Invoquer Quartus :

Programmes > SoPC > Altera > Quartus II

2.3.2 Création du projet

File > New Project Wizard

- Page1 Directory : `D: \users\grxx\Tp_picvhdl`
 - Nom du projet : pic
 - Nom de l'entité au sommet : pic
- Page2 Prendre tous les fichiers sources *.vhd du répertoire *rtl*. Attention !! l'ordre est à priori indifférent sauf pour *ppx_pack.vhd* qu'il faut placer en premier.
- Page3 Rien
- Page4 Family : Stratix
- Page5 Target Device : EP1S10F780C6
- Page 6 Vérification
- Page7 Finish

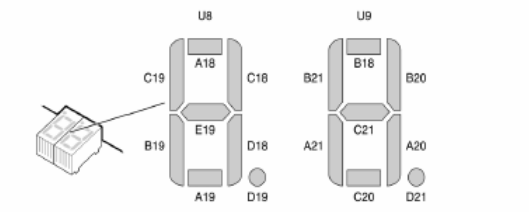
2.3.3 Options de compilation

La liste et l'affectation des broches sera réalisée par un fichier de commande :

View > Auxiliary Windows > Tcl Console

Exécuter dans cette fenêtre : **source synth/stratix_pin_assign.tcl**

Figure 16. Dual-Digit Display



Vérifier que les paramètres ainsi affectés sont corrects.

Assignement > Device...

Chip and Devices > Assign pins

Puis, si tout est ok

Chip and Devices > devices & pin options

unused pins : as input tri_states

Assignement > Timing Settings...

70 MHz pour **Clock Settings** default required Fmax

2.3.4 Compilation

Processing > Start compilation

En fin de compilation, analyser le rapport de compilation (l'analyse détaillée sera faite plus tard).

1. Combien d'éléments logiques ont été utilisés ?
2. Quelle est la fréquence maximum de fonctionnement ?

2.3.5 Programmation

On utilisera la programmation par le port parallèle du PC et le câble ByteBlaster

Tools > Programmer

- Sélectionner **Program/configure**
- Le fichier **pic.sof** est le fichier de configuration du FPGA
- Start** (observez la carte de test)

Il n'est pas nécessaire de sauver la chaîne de configuration ensuite.

2.3.6 Analyse

3. Quelle anomalie constate t-on sur la maquette ?

3 Simulation du niveau RTL

C'est le fichier d : \users\grxx.\Tp_picvhdl\bench\test_pic qui fournit les stimuli pour la simulation. En simulation, il est très pénalisant d'avoir de grandes boucles de temporisation, cela allonge le temps de simulation sans apporter d'information majeure quant à la fonctionnalité. Pour cette raison, on utilisera pour la simulation programme d'application légèrement modifié, toutes les temporisations étant réduites au minimum.

- ❑ Rapport de division de l'horloge passant de 750 à 10 (*pic.vhd*)
- ❑ Initialisation du timer passant de 00 à FF (fichier *chen_sim.vhd*)
- ❑ Rapport de prescaler passant de 256 à 2 (fichier *chen_sim.vhd*)

3.1 Simulation

On commence par invoquer *modelsim* :

Programs > SoPC > Modelsim

- ❑ **File > New Project**
- ❑ Project Name : *pic* ; Project Location : *D: \ users\grxx\Tp_picvhdl\sim*
- ❑ Default Library Name: *work*

La simulation est alors simplissime puisqu'il suffit d'exécuter (**Tools > Execute Macro...**) les 2 macros suivantes (se trouvant dans **bench**)

- ❑ *Compile.do*
- ❑ *Simu.do*
- ❑ *Run 20 us*

Cependant pour bien analyser le déroulement de cette simulation, il est absolument nécessaire de faire la relation entre le code exécuté et le code binaire que l'on observe. On se sert pour cela du fichier *chen_sim.lst* fourni par l'outil de compilation MPLAB

MPASM 03.20.07 Released

CHEN_SIM.ASM 3-5-2003 16:45:22

PAGE 1

LOC OBJECT CODE LINE SOURCE TEXT
VALUE

```

00001
00002
00003 LIST p=16F84 ; Définition de processeur
00004 #include <p16F84.inc> ; Définitions de variables
00001 LIST
00002 ; P16F84.INC Standard Header File, Version 2.00 Microchip Technology, Inc.
00136 LIST
00005
2007 3FF2 00006 __CONFIG __CP_OFF & _WDT_OFF & _PWRTE_ON & _HS_OSC
00007
00008 ; '__CONFIG' précise les paramètres encodés dans le processeur au moment de
00009 ; la programmation du processeur. Les définitions sont dans le fichier include.
00010 ; Voici les valeurs et leurs définitions :
00011
00012 ; __CP_ON Code protection ON : impossible de relire
00013 ; __CP_OFF Code protection OFF

```

```

00014 ;   _PWRTE_ON   Timer reset sur power on en service
00015 ;   _PWRTE_OFF  Timer reset hors-service
00016 ;   _WDT_ON     Watch-dog en service
00017 ;   _WDT_OFF   Watch-dog hors service
00018 ;   _LP_OSC    Oscillateur quartz basse vitesse
00019 ;   _XT_OSC    Oscillateur quartz moyenne vitesse
00020 ;   _HS_OSC    Oscillateur quartz grande vitesse
00021 ;   _RC_OSC    Oscillateur à réseau RC
00022
00023 ;*****
00024 ;               ASSIGNATIONS               *
00025 ;*****
00026
000000F9   00027 TRISA_VAL EQU H'F9' ; PORT A en entrée sauf 1 et 2
00000000   00028 TRISB_VAL EQU H'00' ; PORTB en sortie
000000A0   00029 INTCON_VAL EQU H'A0' ; Validation des interruptions pour le TIMER0
00000020   00030 OPTION_VAL EQU H'20' ; TIMER 0 en mode compteur avec prescaler : 2
000000FE   00031 INIT_TIMER EQU H'FE' ; valeur faible pour la simulation
000000FE   00032 INIT_CHEN EQU H'FE' ; 1 diode allumee
00033 ;*****
00034 ;               DEFINE               *
00035 ;*****
00036
00037
00038 #DEFINE BOUTON PORTA,0 ; bouton-poussoir
00039 #DEFINE SEG1 PORTA,1 ; segment du milieu
00040 #DEFINE SEG2 PORTA,2 ; segment du milieu
00041
00042
00043 ;*****
00044 ;               MACRO               *
00045 ;*****
00046
00047 BANK0 macro
00048     bcf STATUS , RP0 ; passer banque0
00049     endm
00050
00051 BANK1 macro
00052     bsf STATUS , RP0 ; passer banque1
00053     endm
00054
00055 ;*****
00056 ;               DECLARATIONS DE VARIABLES               *
00057 ;*****
00058
00059     CBLOCK 0x00C ; début de la zone variables
00060
0000000C   00061     w_temp : 1 ; Sauvegarde du registre W
0000000D   00062     status_temp : 1 ; Sauvegarde du registre STATUS
00063
00064     ENDC ; Fin de la zone
00065
00066 ;*****
00067 ;               DEMARRAGE SUR RESET               *
00068 ;*****
00069
0000   00070     org 0x00 ; Adresse de départ après reset
0000 2850 00071     goto init ; Adresse 0: initialiser
00072
00073 ;*****

```

```

00074 ;          INITIALISATIONS          *
00075 ;*****
00076
0050 00077      org 0x50
0050 00078      init
00079
00080      BANK1          ; sélectionner banque 1
0050 1683      M      bsf STATUS, RP0 ; passer banque 1
0051 30F9 00081      movlw TRISA_VAL ; charger valeur
0052 0085 00082      movwf TRISA ; configurer le port A
0053 3000 00083      movlw TRISB_VAL ; charger valeur
0054 0086 00084      movwf TRISB ; configurer le port B
0055 3020 00085      movlw OPTION_VAL ; charger valeur
0056 0081 00086      movwf OPTION_REG ; configurer le TIMER0
00087
00088      BANK0          ; repasser banque 0
0057 1283      M      bcf STATUS, RP0 ; passer banque 0
0058 30A0 00089      movlw INTCON_VAL ; charger valeur
0059 008B 00090      movwf INTCON ; configurer les interruptions
005A 30FE 00091      movlw INIT_CHEN
005B 0086 00092      movwf PORTB
005C 30FE 00093      movlw INIT_TIMER ; valeur de réglage du timer
005D 0081 00094      movwf TMR0
00095
00096 ;*****
00097 ;          PROGRAMME PRINCIPAL          *
00098 ;*****
00099
005E 00100      start
005E 1805 00101      btfsc BOUTON ; recopie du bouton sur le segment milieu
005F 2864 00102      goto cest1
0060 2861 00103      goto cest0
0061 1085 00104      cest0 bcf SEG1
0062 1105 00105      bcf SEG2
0063 285E 00106      goto start
0064 1485 00107      cest1 bsf SEG1
0065 1505 00108      bsf SEG2
0066 285E 00109      goto start
0067 285E 00110      goto start
00111
00112 ;*****
00113 ;          ROUTINE INTERRUPTION          *
00114 ;*****
00115
00116          ;sauvegarder registres
00117          ;-----
0004 00118      org 0x004 ; adresse d'interruption
0004 008C 00119      movwf w_temp ; sauver registre W
0005 0E03 00120      swapf STATUS,w ; swap status avec résultat dans w
0006 008D 00121      movwf status_temp ; sauver status swappé
00122
00123
00124          ;traitement de l'interruption
00125          ;-----
00126
0007 30FE 00127      movlw INIT_TIMER ; valeur de réglage du timer
0008 0081 00128      movwf TMR0
0009 0D86 00129      rlf PORTB,f ; décaler à gauche
000A 1406 00130      bsf PORTB,0 ; un 1 à droite
000B 1C03 00131      btfss STATUS,0 ; fin des 8 décalages

```

```

000C 1006    00132    bcf      PORTB,0    ; reinitialise à FE
           00133
           00134          ; restaurer registres
           00135          ;-----
000D        00136    restorerereg
000D 0E0D    00137    swapf   status_temp,w  ; swap ancien status, résultat dans w
000E 0083    00138    movwf  STATUS        ; restaurer status
000F 0E8C    00139    swapf  w_temp,f      ; Inversion L et H de l'ancien W
           00140          ; sans modifier Z
0010 0E0C    00141    swapf  w_temp,w      ; Réinversion de L et H dans W
           00142          ; W restauré sans modifier status
0011 110B    00143    bcf    INTCON,T0IF   ; effacer flag interrupt timer
0012 0009    00144    retfie          ; return from interrupt
           00145
           00146
           00147
           00148
00149    END          ; directive fin de programme

```

3.2 Analyse de la simulation

On s'attachera à observer le déroulement d'une instruction, la vitesse de prise en compte de l'interruption, le détail de l'instruction *retfie* etc..

4. Les instructions s'exécutent-elles en un cycle ?
5. En vous plaçant après le Reset à l'adresse 050, expliquer le séquençage d'une instruction (combien de cycles ?)
6. Expliquer la séquence FC → FD en 2220 ns
7. Calculer le nombre de cycles entre 2 impulsions T0CKI ?
8. Expliquer ce résultat
9. Mesurer le nombre de cycles entre 2 interruptions.
10. Expliquer ce résultat par un calcul simple
11. Par un calcul identique, quelle est alors la valeur de la temporisation sur la maquette ?
12. Quelle est la durée du programme d'interruption en simulation ?
13. Combien y a-t-il d'instructions dans la routine d'interruption ?
14. Expliquer la discordance sur les deux derniers résultats
15. Analyser le mouvement dans la pile au moment de l'interruption
16. Quelle est l'anomalie ?
17. A quel signal erroné peut-on imputer cette erreur ?

4 Deuxième implantation du projet

L'erreur constatée lors de la première implantation, analysée par la simulation conduit à modifier le fichier *ppx_pcs.vhd* (voir fichier *rtl/new_ppx_pcs.vhd*).

Si l'on recompile ensuite le projet, on peut constater que tout fonctionne correctement.

18. Par quelle solution logicielle peut-on cependant obtenir un chenillard correct ?
Écriture, compilation, simulation, implantation.