الجمهورية الجزائرية الديمقراطية الشعبية
وزارة التعليم العالي والبحث العلمي

UNIVERSITE BADJI MOKHTAR - ANNABA
BADJI MOKHTAR – ANNABA UNIVERSITY

جامعة باجي مختار – عنابـــــــــة

Faculté : TECHNOLOGIE
Département : ELECTRONIQUE
Domaine : SCIENCES ET TECHNIQUES
Filière : TELECOMMUNICATIONS
Spécialité : SYSTME ET TELECOMMUNICATION

Mémoire
Présenté en vue de l'obtention du Diplôme de Master

Thème :

Régulation de température : Etude et simulation sur FPGA

Présenté par : *Mellouki Chaima & Bouacha Aicha Beya*

Encadrant : *Mokhnach Azzouz* *Grade: Professeur* *Université: Badji mokhtar-annaba*

Jury de Soutenance :

Taibi Mahmoud	Prof	Badji mokhtar-annaba	Président
Mokhnache Azzouz	M.C.B	Badji mokhtar-annaba	Encadrant
Messadeg Djemil	Prof	Badji mokhtar-annaba	Examinateur

Année Universitaire : 2021/2022

ملخص

كما يوحي الاسم، فإن جهاز التحكم في درجة الحرارة هو أداة للتحكم في درجة الحرارة. يأخذ جهاز التحكم في درجة الحرارة مدخلاً من مستشعر درجة الحرارة، وله مخرج متصل بعنصر تحكم، مثل السخان أو المروحة.

في هذه الذاكرة، نقدم حلقة تنظيم، والتي من درجة حرارة ونقطة ضبط ستولد MLI متناسبًا مع الفرق بين درجة الحرارة ونقطة الضبط ويتم تنفيذها على FPGA، المخصصة للتحكم في الأنظمة الديناميكية وتنظيمها. تحقيقا لهذه الغاية، سوف نعرض من خلال فصول هذه الرسالة المراحل المختلفة التي اتبعناها لتطوير الأجزاء الأربعة الأساسية لنظام التحكم أو التنظيم، في هذه الحالة:

الجزء التنظيمي وجزء الاستحواذ وجزء العرض، الجزء MLI.

تم تنفيذ التنفيذ على لوحةFPGA، على لوحة VIRTEX6 من عائلة VIRTEX .

بيئة التطوير هي مجموعة ISE14.7 من Xilinx .

Abstarct

As the name suggests, a temperature controller is an instrument for controlling temperature. The temperature controller takes an input from a temperature sensor, and has an output that is connected to a control element, such as a heater or ventilator.

In this Brief, we present a control loop, which from a temperature and a setpoint will generate an MLI proportional to the difference between temperature and setpoint implemented on FPGA, dedicated to the control and regulation of dynamic systems. To this end, we will show through the chapters of this brief the various steps that we have followed to develop the four essential parts of a control and regulation system, as the case may be: the part of regulation, the scan portion and the display portion, the MLI portion.

The implementation on an FPGA board was carried out on a VIRTEX6 board from the VIRTEX family.

The development environment is the ISE14.7 suite from Xilinx.

Resumé

Comme son nom l'indique, un régulateur de température est un instrument pour contrôler la température. Le régulateur de température prend une entrée provenant d'un capteur de température, et a une sortie qui est reliée à un élément de commande, tel qu'un dispositif de chauffage ou de ventilation.

Dans ce Mémoire, nous présentons une boucle de régulation, qui à partir d'une température et d'une consigne va générer un MLI proportionnel à l'écart entre température et consigne implémenter sur FPGA, dédiée au contrôle et à la régulation des systèmes dynamiques. A cet effet, nous allons montrer à travers les chapitres de ce mémoire les différentes étapes que nous avons suivies pour développer les quatres parties essentielles d'un système de contrôle et de régulation, en l'occurrence : la partie de régulation, la partie d'acquisition et la partie d'affichage, la partie MLI.

L'implémentation sur une carte FPGA a été réalisé sur une carte VIRTEX6 de la famille VIRTEX.

L'environnement de développement est la suite ISE14.7 de Xilinx.

Mots clés : Régulation - FPGA – Implémentation – Simulation – MLI.

Dédicace

Je dédie ce mémoire :

A mon papa « Cherif » le père et l'ami, mon exemple éternel, mon soutien moral et source de joie, celui qui s'est toujours sacrifié pour me voir réussir, que Dieu te garde toujours avec moi, je te remercie pour les valeurs nobles et pour ton éducation.

A ma mère « Noura » la lumière de mes jours, la source de mes efforts, la flamme de mon cœur, ma vie et mon bonheur, qui m'a comblé de son amour, qui m'a toujours soutenue et conseillée, merci merci beaucoup.

A mes frères « Abderrahmane » « Amine » et « Kamel », sans oublier le sucre de ma vie ma petite « Céline », je dédie ce travail dont le grand plaisir leurs revient en premier lieu pour leurs conseils, aides, et encouragements.

A celui que j'aime beaucoup et qui m'a soutenue tout au long de ce projet « Zinou », et bien sur ma meilleur « Khaoula », j'aurais besoin de toute une vie pour vous remercier et vous décrire combien je vous aime.

A toute ma famille, et mes amis, à mon binôme « Bouchra ».

Aux personnes qui m'ont toujours aidé et encouragé, qui étaient toujours à mes côtés, et qui m'ont accompagné durant mon chemin d'études supérieures, merci à tous d'être dans ma vie.

Mellouki Chaima

Dédicace

Je dédie ce mémoire :

A ma mère « Hakima » l'étoile filante qui brille dans mon univers, qui m'a soutenu et encouragé durant ces années d'études, qu'elle trouve ici le témoignage de ma profonde reconnaissance, je te remercie pour son assistance et sa présence dans ma vie.

A mon père « Abdelhak », qui m'a toujours soutenu et conseillée, merci beaucoup.

A ma sœur « Rahma » et mes frères « Riad » et « Mohammed », qui ont partagé avec moi tous les moments d'émotion lors de la réalisation de ce projet, ils m'ont chaleureusement supporté et encouragé tout au long de mon parcours.

A tous mes proches « Abdelghani », « Nour », « Zina », « Jiji », « Khaoula » et « Alia », merci pour leurs conseils et encagements.

A toute la famille, a mon binôme « Chaima », a tous ceux qui ont contribué de près ou de loin pour que ce projet soit possible, je vous dis merci.

Bouacha Aicha Beya

REMERCIEMENTS

Tout d'abord, nous remercions Dieu « Allah » le tout puissant, de nous avoir donné la santé et la volonté d'entamer et de terminer ce projet.

Nous tenons à remercier notre encadreur Mr. Mokhnache Azzouz pour le grand honneur qu'i nous a fait en nous proposant le sujet de ce mémoire de fin d'étude. Nous avons eu l'honneur et le privilège de travailler sous son assistance et de profiter de ses qualités humaines, professionnelles et de sa grande expérience, il nous guidé tout au long de ce travail.

L'élaboration avec amabilité et dynamisme le caractérisant. Que ce modeste travail puisse satisfaire nos examinateurs, pour qu'ils nous ont prodigué, ainsi que pour le savoir qu'ils nous ont inculqué.

Nous remercions tous nos enseignants de l'université d'Annaba.

Nos remercions vont également aux membres de jury d'avoir accepté de juger notre travail.

Nous remercions sincèrement nos familles, en particulier nos parents, qui ont été obtenez de l'aide pendant que nous étudions. Puissent-ils trouver ici le fruit de la patience et du soutien ils nous font affronter tous les moments difficiles.

Nous tenons également à remercier nos collègues pour leur aide et leur conseils précieux.

LISTE DES ACRONYMES

A:

ASIC: Application-specific integrated circuit.

B :

BO : Boucle Ouverte

BF : Boucle Fermée.

C :

CAN : Convertisseur Analogique-Numérique.

CNA : Convertisseur Numérique-Analogique.

D:

DSP: Digital Signal Processor.

F:

FPGA: Field Programmable Gate Array.

H:

HDL: Hard ware description language.

I :

ISE : Signifie l'environnement de Synthèse Intégré.

P :

PID : Proportionnelle Intégral et Dérivée.

PD : Proportionnelle Dérivée.

PI : proportionnelle Intégral.

T:

TOR: Tout ou rien.

V:

VHDL: Very High-Speed Integrated Circuit Hard ware Description Language.

LISTE DES FIGURES

Figure 1. 1: Réponse d'un procédé asservi un échelon de consigne. 3
Figure 1. 2:Réponse d'un procédé régulé à un échelon de perturbation. 4
Figure 1. 3: Schéma de principe du la régulation. 5
Figure 1. 4: système en BO……. 6
Figure 1. 5: Système en BF. 6
Figure 1. 6: Action discontinue. 7
Figure 1. 7: Schéma bloc de la boucle de réglage. 8
Figure 2.1 : Schéma général du régulateur proposé……11
Figure 2.2 : Entité acquisition……..……..…13
Figure 2.3 : Entité affichage……..…14
Figure 2.4 : Entité acquisition / affichage……..15
Figure 2.5 : Schéma régulateur……..16
Figure 2.6 : Schéma programmateur……..18
Figure 2.7 : Schéma de fonctionnement de la partie qui génère le mode de travail du régulateur…18
Figure 2.8 : Schéma d'un bloc de quatre boutons……19
Figure 2.9 : Schéma de compteur (REG10)……..20
Figure 2.10 : Schéma de compteur / décompteur……..20
Figure 3.1 : Structure de base d'un module sous VHDL……25
Figure 3.2 : Syntaxe déclarative de l'entité……25
Figure 3.3 : Syntaxe déclarative de l'architecture……..26
Figure 3.4 : Structure d'un programme sous VHDL……..27
Figure 3.5 : Carte XC6VLX75T-2F488……29
Figure 3.6 : Résultats " No errors"……..33
Figure 3.7 : Les ressources utilisés "1er partie"……...34
Figure 3.8 : Les ressources utilisés "2éme partie"……..35
Figure 3.9 : Le composant de régulation complet "généré par ISE14.7"……36
Figure 3.10 : Reg_Temp " généré par ISE14.7"……37
Figure 4.1 : Mli_tb.vhdl……39
Figure 4.2 : vérification MLI-TB……...40
Figure 4.3 : Mode_nor_pro_tb.vhdl……...41
Figure 4.4 : Sys_aff_tb.vhdl……..42
Figure 4.5 : ModelSIM : MLI à 1……..43
Figure 4.6 : MLI à zéro égalité temp consigne……..44
Figure 4.7 : MLI à 1 simulateur ISIM intégré……45

TABLE DES MATIERES

ملخص.....2
Abstract 2
Résumé 3
Dédicace 4
Remerciement.....6
Liste des acronymes.....7
Liste des figure.....8
Introduction Générale 1

Chapitre 1 : Présentation générale de la régulation.

1.1 Introduction générale sur la régulation : **3**
1.2 L'ASSERVISSEMENT : **3**
1.3 Régulation : **4**
1.3.1 OBJECTIFS DE LA REGULATION :..... 4
1.3.2 Principe général de la régulation : 5
1.3.3 TYPOLOGIE DE LA REGULATION :..... 5
1.3.4 Régulation tout ou RIEN (TOR) :..... 7
1.4 Etude de régulateur industriel PID : **8**
1.4.1 Régulation analogique :..... 8
1.4.2 Régulation numérique :..... 8
1.5 Conclusion : **9**

Chapitre 2 : Description détaillée sur schéma proposé.

2 SCHEMA PROPOSE : **11**
2.1 PRINCIPE DU SCHEMA : **11**
2.2 LES DIFFERENTS BLOCS ET ENTITES **13**
2.2.1 ENTITE ACQUISITION :..... 13
2.2.2 ENTITE AFFICHAGE 14
2.2.3 ENTITE ACQUISITION AFFICHAGE..... 15
2.2.4 Bloc de régulation 15
2.3 Fonctionnement du programmateur **17**
2.4 Conclusion **21**

Table des matières

Chapitre 3 : Implémentation sur FPGA.

3 IMPLEMENTATION FPGA : 23

3.1 Introduction : 23
3.2 Architecture d'un FPGA : 23
3.3 Le langage VHDL : 23
3.3.1 La Simulation et la synthèse : 24
3.3.2 Portabilité : 24
3.3.3 Une construction hiérarchique : 24
3.3.4 Une description fonctionnelle : 24
3.3.5 Structure d'un programme VHDL (LE couple entité, architecture): 25
3.3.5.1 Entity ou Entité: 25
3.3.5.2 Architecture : 25
3.3.6 Les test benchs : 27

3.4 choix environnement et carte : 27
3.4.1 Présentation de la suite logicielle xilinx : 28
3.4.2 Présentation et choix du logiciel XILINX ISE 14.7: 28
3.4.3 Choix de la Famille Virtex : 28
3.4.4 Choix de la carte XC6VLX75T-2F484 : 29

3.5 Implémentation : 29
3.5.1 Compilation des codes VHDL 32

3.6 Conclusion 37

Chapitre 4 : Simulations et Résultats.

4 SIMULATION : 39

4.1 Résultats et conclusion sur la simulation: 45

Conclusion générale **46**
Bibliographie **47**

Introduction générale

Dans la grande majorité des cas, la régulation dans toutes ses formes est présente dans les procédés industriels et autres. Elle vise entre autres à améliorer un certain nombre de paramètres, entre autres on peut citer la sécurité, les économies d'énergie et le respect de l'environnement. Donc une place centrale est donnée à la régulation dans tous les domaines tant industriels que d'autres domaines. Ceci vise en particulier à rester près des conditions optimales de fonctionnement, souvent définies par les cahiers de charges [1].
Le contrôle en boucle fermée, régulation ou asservissement, est le plus populaire parmi la plupart des fabricants. [2].
Dans la cadre de notre projet de fin d'études, nous nous sommes proposés de réaliser une boucle de régulation applicable à de très nombreux processus. Le cahier des charges nous impose de traiter cette boucle.
Nous proposons :

- Une implémentation FPGA sur une carte Vitex6XC6VLX75T de la société Xilinx

Ce mémoire se compose de quatre chapitres, répartis comme suit :

- **Le 1er Chapitre** donne une présentation générale de la régulation déclinée dans ses différentes variantes pour situer notre étude par rapport à ce qui existe.

- **Le 2éme Chapitre** donne une description détaillée du schéma proposé et expose les principes sur lesquels s'appuient le régulateur mis en œuvre dans ce mémoire.

- **Le 3éme Chapitre** détaille l'implémentation FPGA de ce régulateur. On y expose le choix de la carte FPGA choisie. C'est une VIRTEX6 (XC6VLX75T du package : F484 Speed : -2) de Xilinx ainsi que l'environnement de développement Xilinx ISE 14.7.

Les motifs de ces choix sont explicités.

- **Le 4éme Chapitre** Tous les programmes VHDL écrits pour implémenter et simuler notre régulateur sont décrits dans leurs aspects fonctionnels.

- . Enfin, nous terminons par une conclusion et quelques remarques.

Chapitre 1 :

Présentation générale sur la régulation

1.1 INTRODUCTION GENERALE SUR LA REGULATION :

La régulation est une technique qui consiste le plus souvent à maintenir une grandeur physique ou physico-chimique à une valeur imposée appelée consigne. Ces grandeurs peuvent être de nature très diverses, telles que : température, pression, niveau, débit, densité, etc… [4]
La régulation consiste à annuler l'écart entre la valeur mesurée (grandeur à règles) et la consigne. Par exemple le niveau d'un liquide dans un bac est régit sur le débit d'entrée et le débit de sortie, ceci constitue un système réglé. Le niveau réalisé est la grandeur réglée et le niveau désiré est la valeur de consigne. [4]
Dans un procédé industriel, chaque boucle de régulation a pour objectif de maintenir une grandeur physique égale à la consigne, quelles que soient les variations des grandeurs perturbatrices, à l'aide d'un actionneur agissant sur une grandeur réglant. Si on cherche à atteindre une consigne, on parlera sur l'asservissement, si on veut éliminer des perturbations pour qu'une valeur reste constante, on parlera sur la régulation.
Dans ce chapitre, nous allons donner quelques rappels sur l'asservissement et la régulation

1.2 L'ASSERVISSEMENT :

L'asservissement est un système dont l'objet principal est d'atteindre le plus possible sa valeur de consigne notée C et de la maintenir constante (Y(t) la valeur mesurée de système qu'est présentée en bleu dans le figure1-1), quelles que soient les perturbations externes. Le principe général est de comparer la consigne et l'état du système de manière à le corriger efficacement. On parle également de système commandé en boucle fermée.
La figure 1-1 illustre la réponse d'un procédé asservi à un échelon de consigne [3].

Figure 1. 1: Réponse d'un procédé asservi un échelon de consigne.

❖ **EXEMPLES :**

- ✓ Asservissement de température : obtention d'un profil de température en fonction du temps dans un four de traitement thermique.
- ✓ Asservissement d'un débit d'air par rapport à un débit de gaz afin d'obtenir une combustion idéale.

1.3 REGULATION :

La régulation est une partie de la science technique appelée automatique, c'est l'ensemble des techniques qui permettent de contrôler une grandeur physique, sans intervention humaine, pour la maintenir à une valeur donnée, appelée consigne notée Y. Cette dernière traduisant l'objectif désiré du procédé, est constante et les grandeurs perturbatrices influencent fortement sur la grandeur à maitriser notée y(t) [1].

La réponse d'un procédé régulé à un échelon de perturbation est représentée dans la figure 1-2.[3]

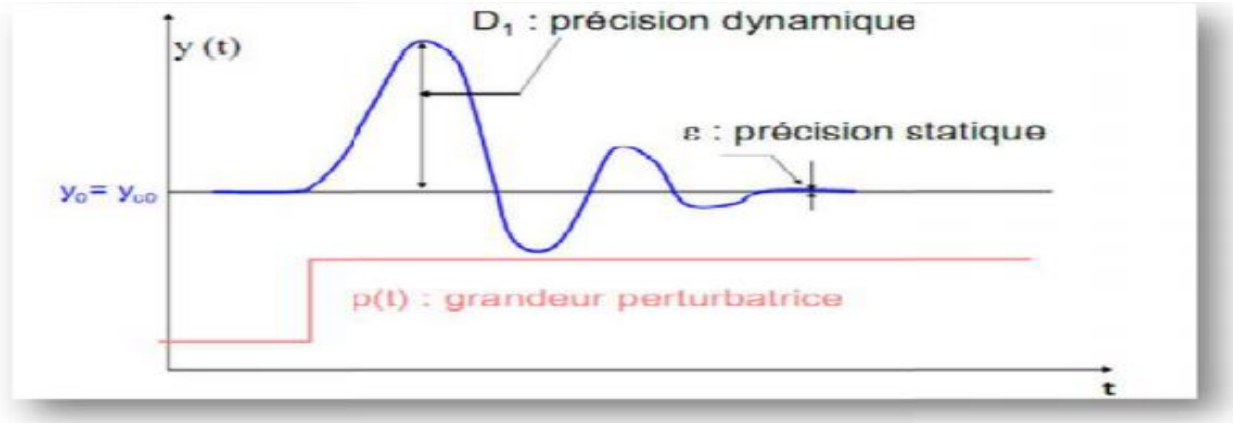


Figure 1. 2:Réponse d'un procédé régulé à un échelon de perturbation.

❖ **EXEMPLES :** [5]

- ✓ Régulation de température dans un local subissant les variations climatiques.
- ✓ Régulation de niveau dans un réservoir dépendant de plusieurs débits d'alimentation et de soutirage.
- ✓ Régulation de Ph de rejet d'eau destinés à être déversés dans une rivière.

1.3.1 OBJECTIFS DE LA REGULATION :

L'objectif d'une régulation ou d'un asservissement est d'assurer le fonctionnement d'un procédé selon les critères prédéfinis par un cahier des charges. Les aspects de sécurité des personnes et des installations doivent à être pris en considération, tout comme l'énergie et le respect de l'environnement. Un cahier des charges définit des critères qualitatifs à imposer qui se traduisent souvent par des critères

qualitatifs, comme la stabilité, la précision, la rapidité ou de lois d'évolution [1]. Voici quelques exemples d'objectifs qualitatifs : [2]

- ✓ Obtenir une combustion air-gaz correcte dans un bruleur.
- ✓ Maintenir une qualité constante d'un mélange de produits.
- ✓ Obtenir un débit de fluide constant dans une conduite en fonction des besoins.
- ✓ Faire évoluer une température d'un four selon un profil déterminé.

1.3.2 PRINCIPE GENERAL DE LA REGULATION :

Dans la plupart des appareils et installations industrielles, tertiaires et mêmes domestiques, il est nécessaire de maintenir des grandeurs physiques à des valeurs déterminées, en débit des variations externes ou internes influant sur ces grandeurs. Par exemple, le niveau d'un réservoir d'eau, la température d'une étuve, le débit d'une conduite de gaz, étant de nature variable, doivent donc être réglés par des actions convenables sur le processus considéré. Si les perturbations affectant la grandeur à contrôler sont lentes ou négligeables, un simple réglage (appelé boucle ouverte) permet d'obtenir et de maintenir la valeur souhaité (par exemple : action sur un robinet d'eau). Dans la majorité des cas, ce type de réglage n'est pas suffisant, parce que trop grossier ou instable.
Il faut alors comparer, en permanence, la valeur mesurée de la grandeur réglée à celle que l'on souhaite obtenir et agir en conséquence sur la grandeur d'action, dite grandeur réglant. [5]
La figure ci-dessous illustre le principe de la régulation.

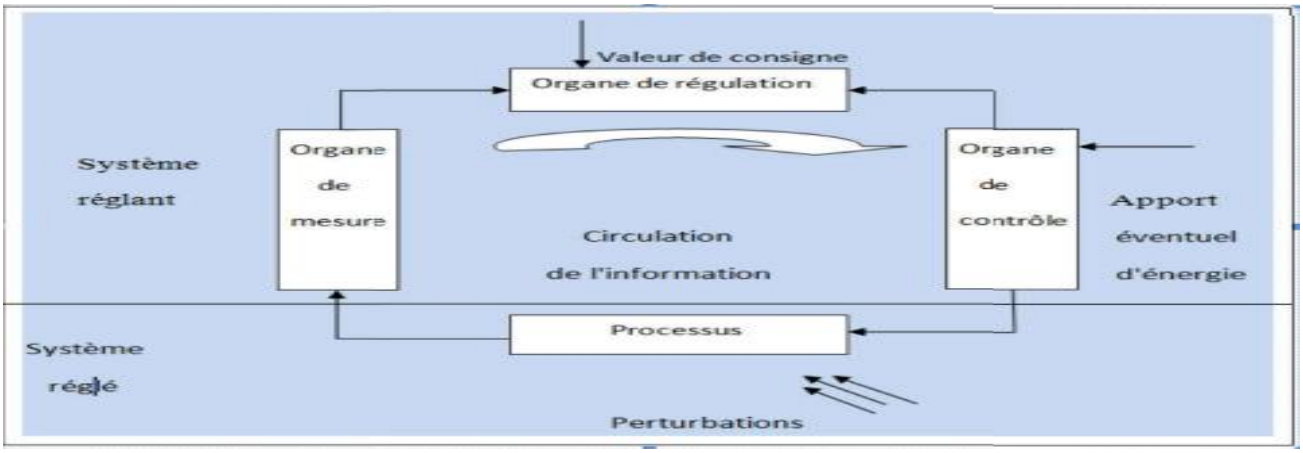


Figure 1. 3: Schéma de principe du la régulation.

1.3.3 TYPOLOGIE DE LA REGULATION :

Il y'a deux types de boucles de régulation : régulation en boucle ouverte et en boucle fermée.

a –Régulation en boucle ouverte :
Un système en boucle ouverte est un système qui ne comporte pas de contre-réaction (feedback) entre

la sortie et l'entrée, qui n'implique aucune information sur la sortie. Classiquement, il est composé d'un correcteur, actionneur et un processus. Comme elle montre la figure suivante : [6]

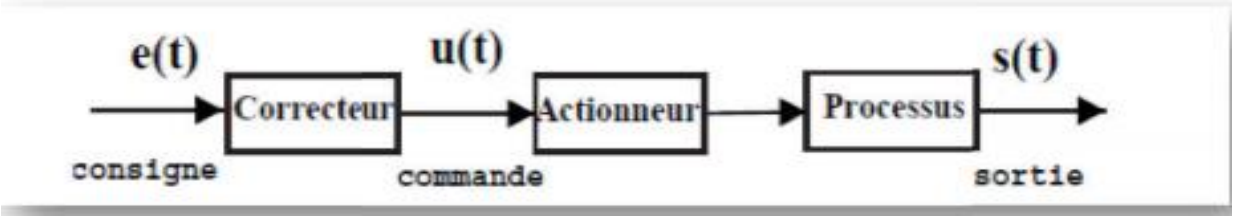


Figure 1. 4: système en BO.

Le système en boucle ouverte contient des avantages et des inconvénients parmi eux :

- **Les avantages :** [5]
  - ✓ Permet d'anticiper les phénomènes et d'obtenir ses temps de réponse très courts.
  - ✓ Il n'y a pas d'oscillation à craindre (car il s'agit d'un système dynamiquement stable).
  - ✓ C'est la seule solution envisageable lorsqu'il n'y a pas de contrôle final possible

- **Les inconvénients :** [7]
  - ✓ Correction impossible : n'ayant aucune information sur la sortie, l'opérateur ne peut élaborer aucune stratégie d'ajustement pour obtenir la sortie désirée.
  - ✓ Sensibilité aux perturbations peut, à un moment donné, affecter la sortie. L'opérateur ne pourra corrige cette situation.

b- Régulation en boucle fermée :

La grandeur réglant exerce une influence sur la grandeur réglée, pour la maintenir dans des limites définies malgré les perturbations. La figure ci-dessous illustre un système en BF [3].

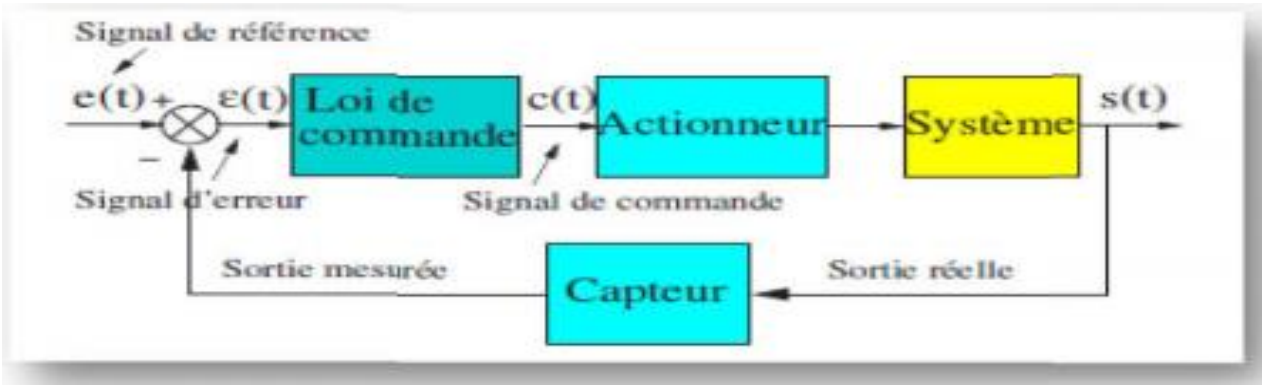


Figure 1. 5: Système en BF.

Comme celle en boucle ouverte, la boucle fermée contient aussi un côté positif et un côté négatif :

- **Côté positif :** [5]
  - ✓ Une bonne partie des facteurs perturbateurs sont automatiquement compenses par la contre-réaction à travers le procédé.
  - ✓ Il n'est pas nécessaire de connaitre avec précision les lois, le comportement des différents composants des allures statistique et dynamiques des divers phénomènes rencontre soit utile pour le choix des composants.
- **Côté négatif :** [5]
  - ✓ Il faut citer le fait que la précision et la fidélité de la régulation dépend de la fidélité et de la précision sur les valeurs mesurées et sur la consigne.
  - ✓ Le comportement dynamique de la boucle dépend des caractéristiques des différents comportements de la boucle, et notamment du processus.
  - ✓ Pour que la régulation envoie une commande à l'organe de contrôle, il faut que les perturbations ou les éventuelles variations de la valeur de consigne se manifestent sur la sortie du processus.

1.3.4 REGULATION TOUT OU RIEN (TOR) :

Ce dispositif est utilisé surtout pour les réglages de température ou de niveau, c'est-à-dire pour des processus à forte inertie. La régulation TOR se caractérise par une action sur l'organe de régulation et ne peut fonctionner qu'à 100% ou 0%. L'action du régulateur peut se présenter comme un contacte ouvert ou fermé, aussi comme un signal de 0V ou bien 24V pour commander une électrovanne [4] [8].

La figure ci-dessous présente une action discontinue d'un régulateur TOR. [4]

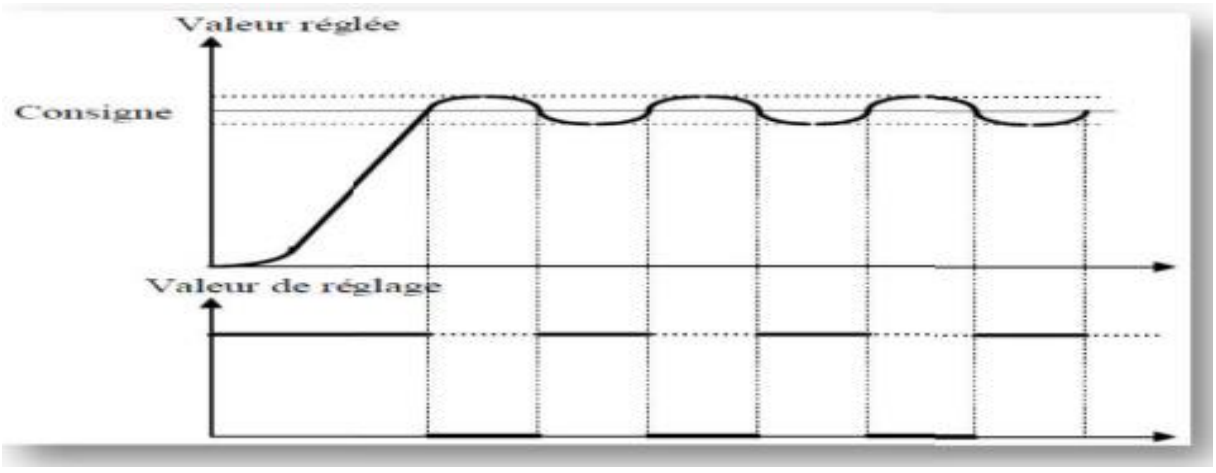


Figure 1. 6: Action discontinue.

1.4 ETUDE DE REGULATEUR INDUSTRIEL PID :

Le régulateur industriel est un appareil qui a pour rôle essentiel de contrôle le procédé, c'est-à-dire de garantir les comportements dynamique et statique du procédé conformes au cahier des charges défini.

Ceci est réalisé en réglant et ajustant les paramètres de sa fonction de transfert au procédé à contrôles. [1]

La figure ci-dessous illustre le schéma bloc de la boucle de réglage [9]

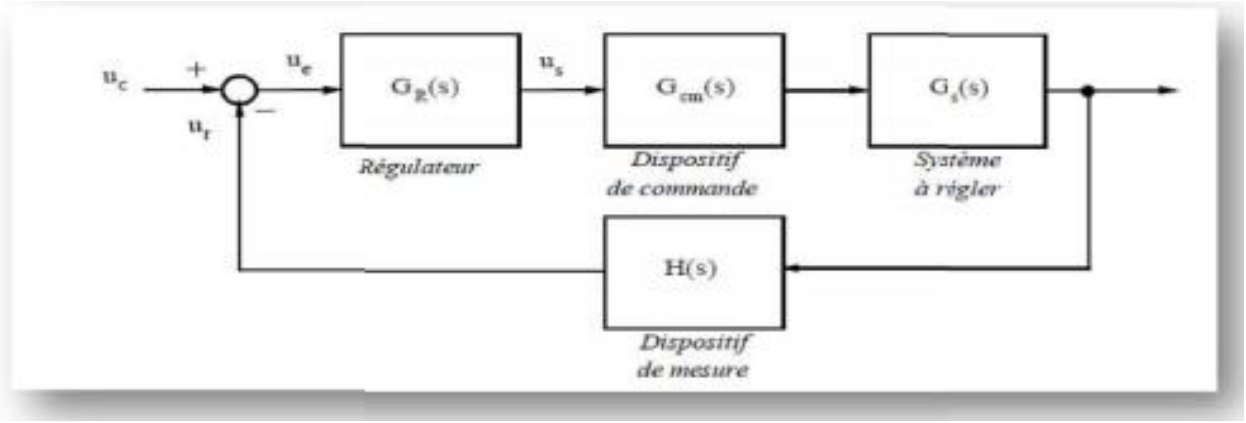


Figure 1. 7: Schéma bloc de la boucle de réglage.

Il existe deux types de régulations industrielles :

- Régulation analogique.
- Régulation numérique.

1.4.1 REGULATION ANALOGIQUE :

C'est un système qui fonctionne à partir d'un signal continu spécifique sous forme de tension, nécessaires à la réalisation des régulateurs analogique. Les valeurs des grandeurs physiques constituant les signaux analogiques doivent être représentés par des nombres.

1.4.2 REGULATION NUMERIQUE :

Le principe de la régulation numérique est identique à celui de la commande analogique. Dans les deux cas, la structure du système bouclé reste la même, mais les signaux nécessaires au traitement de l'information changent, à cet effet, Un des dispositifs électroniques qui permettent de transformer les signaux analogiques au numérique et vise versa. Ceci est illustré comme suit :

- Le signal mesuré échantillonné et quantifié est obtenu par un convertisseur analogique/numérique.
- Le signal réglant fourni sous la forme d'une suite de nombres est obtenu par un convertisseur numérique/analogique.

1.5 CONCLUSION :

Nous avons vu dans ce premier chapitre les notions d'asservissement et de régulation avec ses deux types (régulation en boucle ouverte et régulation en boucle fermée). Aussi on a vu le principe de fonctionnement de la régulation industriel et ses types de boucles (régulation analogique et régulation numérique).

Chapitre 2 :

Description détaillée du schéma proposé

2 SCHEMA PROPOSE :

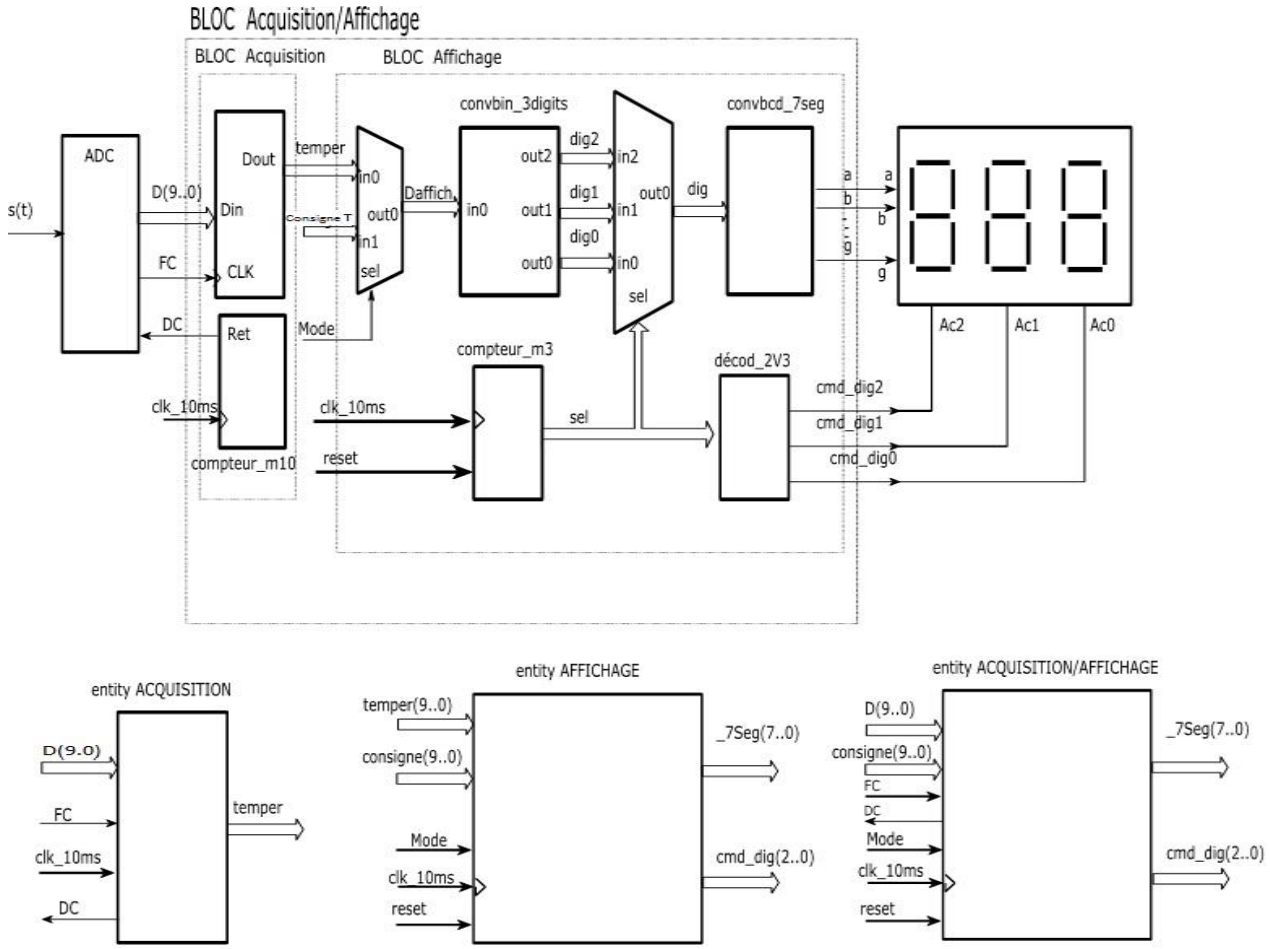


Figure 2. 1: Schéma général du régulateur proposé.

2.1 PRINCIPE DU SCHEMA :

Le schéma du régulateur est composé de quatre blocs principaux réalisant chacun une tâche spécifique :

- ❖ Un bloc d'acquisition.
- ❖ Un bloc d'affichage.
- ❖ Un bloc MLI.
- ❖ Un bloc de programmation.

Nous allons décrire dans le détail le fonctionnement de ce régulateur.

Comme présenté figure 2.1, Les deux premiers blocs sont associés dans un seul bloc appelé bloc acquisition / affichage.

Les liaisons entre les deux blocs sont clairement indiquées dans la figure 2.1.

Le schéma de cheminement du signal est ci-dessous exposé :

Le signal s(t) venant du capteur de température est appliqué à l'entrée du convertisseur, Selon la précision voulue (ici 10 bits), la sortie est injectée sur l'entrée (Din) du bloc acquisition qui est constituée entre autres d'un registre tampon et d'un compteur (compteur_m10). Sur le front montant du signal d'horloge (CLK), le signal est transféré sur la sortie (Dout). Ceci est destiné à maintenir stable la valeur de la température tout le temps du traitement.
Le compteur (compteur m_10) fonctionne en permanence et donne un ordre de début de conversion (DC) au bout d'un temps déterminé (correspondant ici à 100 ms, c'est-à-dire 10ms *10). En fin de comptage une retenue est générée et c'est cette retenue qui va servir de signal de début de conversion, pour donner naissance à un nouveau cycle d'acquisition.
L'entrée du bloc d'affichage est constituée d'un sélecteur à deux entrées (In0 et In1) et d'une sortie (Daffich ou Out0).
Les deux entrées reçoivent respectivement le signal de température (sur In0) et la consigne (sur In1).
Il convient de signaler que le signal mode appliqué sur l'entrée Sel du sélecteur est celui qui permet de choisir les données à afficher (Température ou consigne).
Selon le mode de fonctionnement (normal ou programmation) le bloc d'affichage traitera et affichera alternativement la température ou la consigne.
En effet, deux modes de travail existent :

- Le mode de programmation qui permet de choisir et fixer la consigne de température.
- Le mode normal de fonctionnement qui affiche la température en permanence.
- La sortie du sélecteur est dirigé vers un bloc de conversion binaire-trois digits.
- L'entrée In0 est convertie en en trois digits (Out0, Out1, Out2) qui sont dirigés vers les entrées du sélecteur. Les signaux sur les sorties Out0, Out1, Out2 renommés en dig0, dig1, dig2 (on est à la sortie du convertisseur convbin_3digits) sont appliqués aux entrées In0, In1, In2 du sélecteur. La sortie choisie est déterminée par le signal sel appliquée sur l'entrée sel correspondante.

- Un compteur modulo 3 (compteur_m3) voit sa sortie dirigée vers le sélecteur et appliquée à l'entrée sel.
- Le choix est fait et une des entrées est dirigée vers Out0 du sélecteur. Le signal est alors appliqué (dig0) à l'entrée du convertisseur BCD 7 segments (ConvBCD_7Seg). Les sorties du ConvBCD_7Seg sont prêts à être appliquées à l'afficheur adéquat.
- Les digits (dig0, dig1, dig2) sont respectivement destinés à être affichés sur les afficheurs Ac0, Ac1, Ac2.

- Ce choix est fait par l'utilisation de la sortie sel du compteur_m3. De manière simultanée (rappelons qu'il est appliqué à l'entrée du sélecteur de digits), il est appliqué au de-cod_2v3 dont les sorties cmd_dig0, cmd_dig1, cmd_dig2 sont liées ,décodées , vont commander les afficheurs correspondants, c'est-à-dire dans l'ordre Ac0, Ac1, Ac2.
- Ainsi on assure que la bonne donnée est affichée sur le bon afficheur.

2.2 LES DIFFERENTS BLOCS ET ENTITES

- Comme précédemment indiqué, le découpage fonctionnel est clairement indiqué dans le schéma de la figure 2.1.
- Pour plus de clarté, nous allons montrer les différents blocs séparément. Cette présentation détaillée convient à ce que nous allons faire :
- Ecrire le code VHDL qui décrit les différents blocs.
- Implémenter le schéma.
- Simuler le schéma.

2.2.1 ENTITE ACQUISITION :

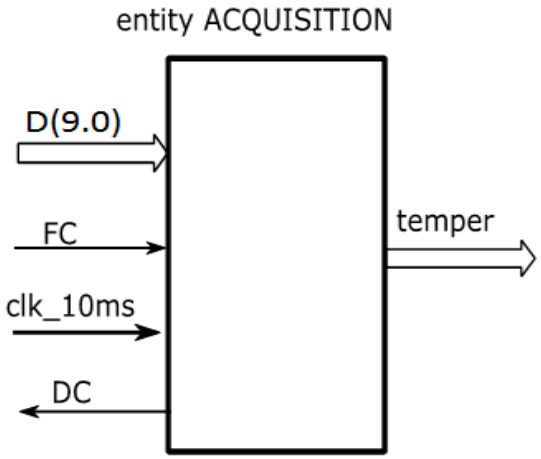


Figure 2. 2: entité acquisition.

Comme illustré dans ce schéma, le bloc acquisition a pour entrées

- D (9,0) température.
- FC (signal de fin de conversion).
- Clk_10ms (horloge de 10ms).
- DC (signal de début de conversion).

Elle a pour sortie

- La température.

2.2.2 ENTITE AFFICHAGE

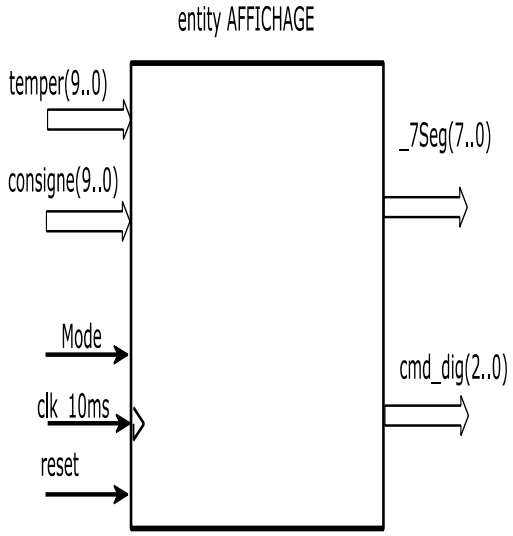


Figure 2. 3: entité affichage.

L'entité affichage a pour entrées :

- Température (9..0) : La température.
- Consigne (9..0) : La consigne.
- Mode : Le mode (programmation ou normal).
- Clk_10ms : Horloge 10 ms.
- Reset : Remise à zéro.

L'entité affichage a pour sorties :

- -7Seg (7..0) : code 7 segments pour l'affichage.
- Cmd_dig (2..0) : code pour le choix de l'afficheur.

Encore une fois, cette présentation se prête bien à une description VHDL.

2.2.3 ENTITE ACQUISITION AFFICHAGE

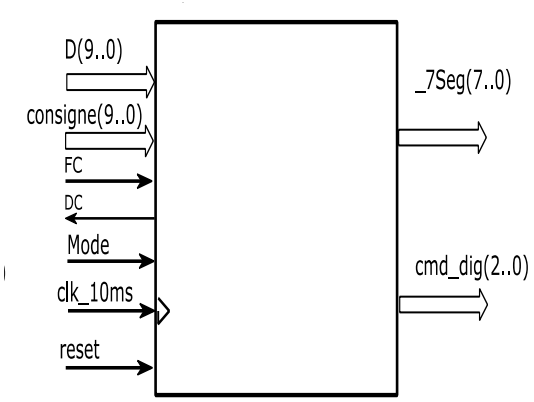


Figure 2. 4: Entité acquisition/ affichage.

L'entité acquisition / affichage a pour entrées :

- Température (9..0) : La température.
- Consigne (9..0) : La consigne.
- FC : Fin de conversion.
- DC : Début de conversion.
- Mode : Le mode (programmation ou normal).
- Clk_10ms : Horloge 10 ms.
- Reset : Remise à zéro.

L'entité acquisition affichage a pour sorties :

- -7Seg(7..0) : code 7 segments pour l'affichage.
- Cmd_dig(2..0) : code pour le choix de l'afficheur.

Encore une fois, cette présentation se prête bien à une description VHDL.

2.2.4 BLOC DE REGULATION

La régulation llaz

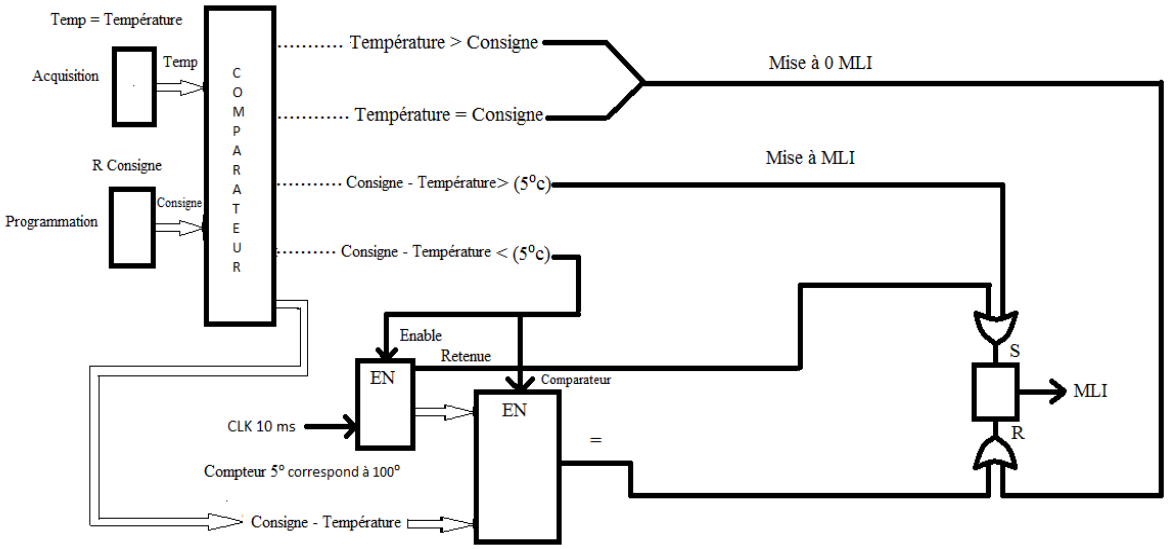


Figure 2. 5 : Schéma régulateur.

La régulation se fait selon le schéma suivant :
L'entrée du régulateur est un comparateur qui reçoit sur ses entrées :

- La température
- La consigne

Les résultats possibles de cette comparaison sont les suivants :

1. La température est supérieur à la consigne
2. La température est égale à la consigne
3. La température est inférieure à la consigne

Le comportement du système est alors le suivant :
Il convient de signaler que l'action principale consiste à générer un signal MLI.

Ce signal MLI est mis à :

- 1 : pour mettre en marche l'élément de chauffage et donc corriger l'écart entre la température et la consigne jusqu'à zéro.
- 0 : Pour mettre à l'arrêt l'élément de chauffage.

Il est clair que les cas :

1- La température est supérieur à la consigne.
2- La température est égale à la consigne.

nécessitent la mise à zéro du MLI, dans la mesure où on laisse tomber la température sans action dans le cas 1 et on ne fait rien dans le cas 2.

Le cas 3 (La température est inférieure à la consigne) se subdivise en deux cas :

1. Consigne – température > 5 degrés
2. Consigne – température < 5 degrés
   - Le cas 1 entraine la mise à 1 du MLI pendant tout le cycle (Chauffage-arrêt), car on estime que l'écart est encore important.
   - Le cas 2 entraine aussi la mise à 1 du MLI mais sur une partie du cycle. Une comparaison est effectuée entre la sortie du comparateur et un compteur (correpondant à 5 degrés), Si le cycle est complet, il ya génération d'une retenue qi sera utilisée pour mettre à 1 le MLI.

   En cas d'égalité , il faut alors mettre le MLI à zéro sur la bascule SR et le cycle recommence. La comparaison entre température et consigne va être éffectuée de nouveau. Comme l'écart s'est réduit entre temps, cette fois çi, le MLI va être moins long et ainsi de suite jusqu'à l'égalité température et consigne. Les signaux ENABLE vont disparaitre et on est dans le cas de l'égalité entre consigne et température. Ce cas provoque la mise à zéro du MLI puisu'on retombe dans le cas température = consigne..

La boucle est alors complète.

2.3 FONCTIONNEMENT DU PROGRAMMATEUR

Le programmateur est présenté dans le schéma suivant 2.6.

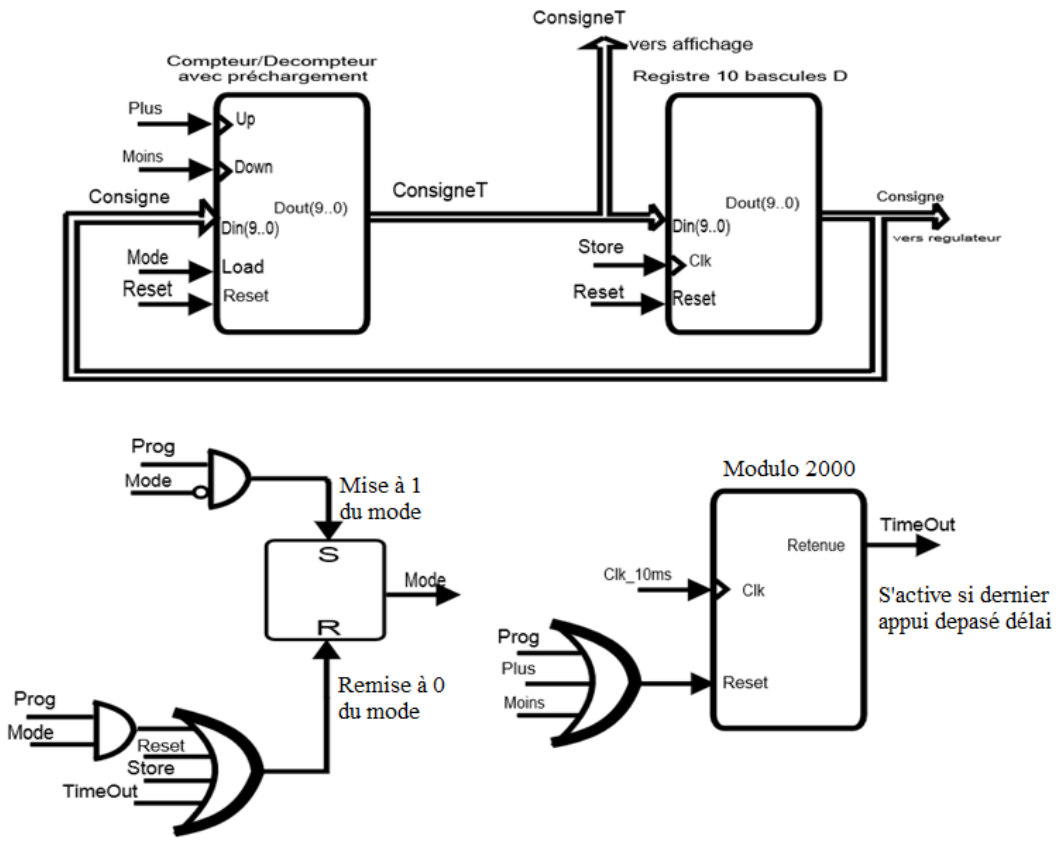


Figure 2. 6: Schéma programmateur.

Le fonctionnement du programmateur est le suivant :

On va d'abord expliquer le foncctionnement de la partie qui génère le mode de travail du régulateur.

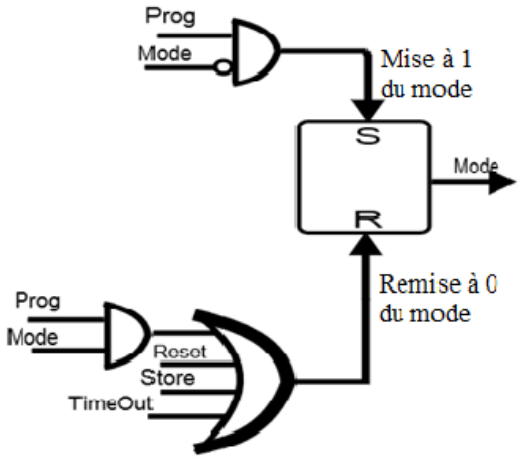


Figure 2. 7: Schéma de fonctionnement de la partie qui génère le mode de travail du régulateur.

Rappelons que deux modes de fonctionnement sont possibles :

- Le mode programmation qui permet de fixer la consige
- Le mode de fonctionnement normal de régulation

Un bloc de quatre boutons :

- Programmation
- Plus
- Moins
- Store

permet de procéder à la programmation de la consigne.

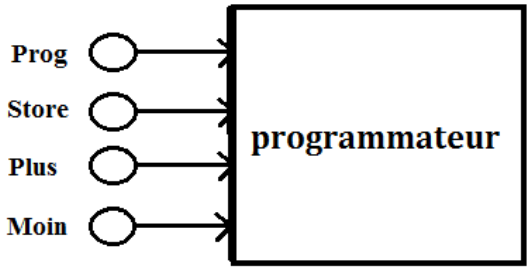


Figure 2. 8: Schéma d'un bloc de quatre boutons.

La séquence de programmation se passe ainsi :

1. Un appui sur le bouton programmation permet d'entrer dans le mode programmation ou d'en sortir
2. Sur le front montat du signal prog, si le mode était à zéro, il passe à 1(c'est-à-dire que du mode normal on passe en mode programmation). La porte ET du haut du schéma 2.6 illustre ce fonctionnement.
   La sortie de la porte ET met le mode à 1 sur l'entrée SET de la bscule SR
3. La port ET du bas du schéma 2.6 à sa sortie (prog = 1 et mode = 0) à zéro sort donc de la commande.

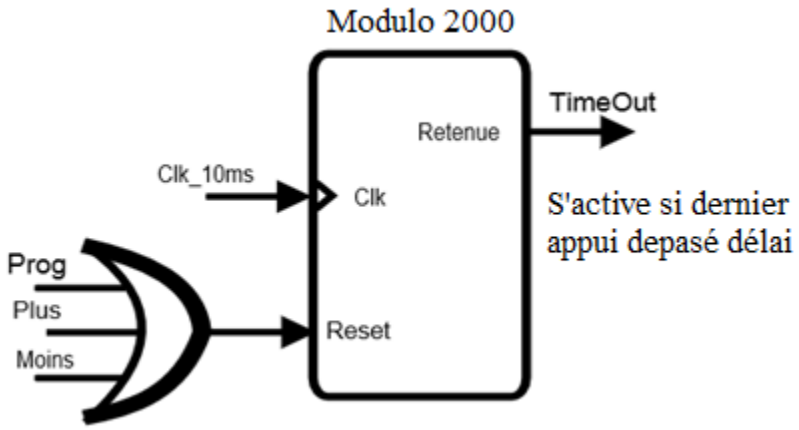


Figure 2. 9: Schéma de compteur (REG10).

- ✓ Le compteur (REG10) de la figure 2.9 est un compteur de TIMEOUT.
- ✓ Le timeout est une durée de temps (ici 2000 ms pour l'exemple) au bout de laquelle, on estime que la reprogrammation est nécessaire pour une raison quelconque.
- ✓ On voit clairement sur le schémaque l'entrée CLK reçoit l'horlage. Le décompte se fait normalement. En bout de comptage, une retenue est généré et sert de de signal de timeout.
- ✓ Si l'un des boutons (PROG, PLUS, MOINS) est appuyé, un RESET remet le compteur à zéro (un reset est nécessaire car cela voudra que l'opérateur est entrain de régler la consigne normalemen et donc il faut recommencer un décomptage du cycle timeout) .

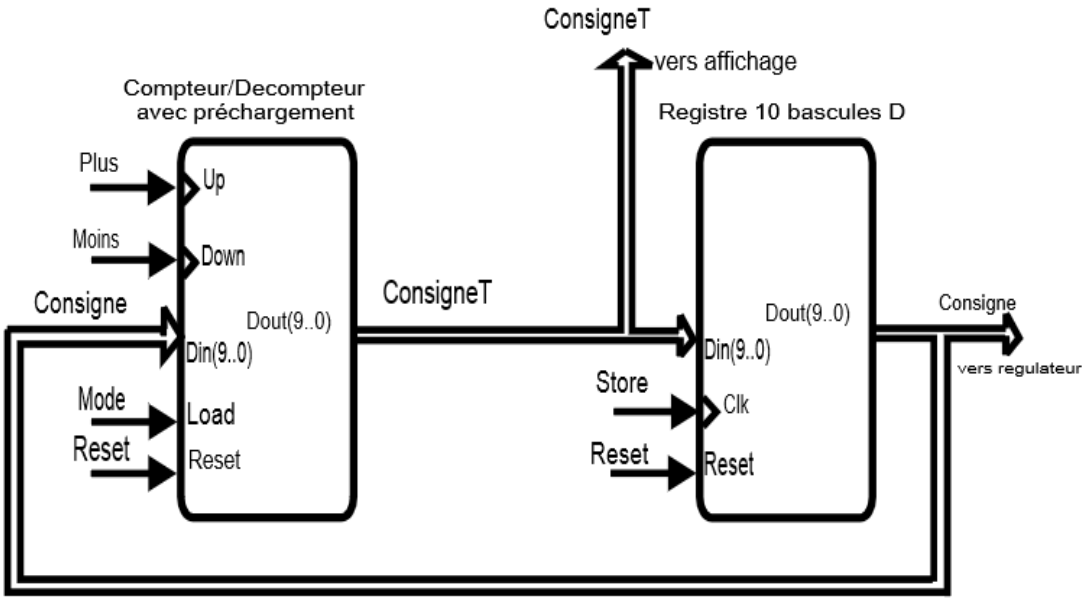


Figure 2. 10: Schéma de compteur/décompteur.

4. Le front montant du mode 1 (sur l'entrée LOAD) charge le compteur / décompteur avec la valeur de la consigne.
5. Le front montant du mode 1 sélectionne l'affichage de la consigne.
6. Le bouton PLUS augmente la valeur de la consigne temporarire consigneT.
7. Le bouton MOINS diminue la valeur de l consigne temporaire consigneT .
8. Le bouton STORE mémorise la consigne temporaire (consigneT) dans la consigne et sort du mode de programmation.
9. Si le mode = 1 et front montatnt de PROG , on sort du mode programmation sans mémoriser.

2.4 CONCLUSION

Voici donc un aperçu du fonctionnement de description détaillée du schéma générale de régulateur proposé et ses différents blocs/entités, et le schéma de régulateur, aussi on a vu une description détaillée du schéma de programmateur.

Chapitre 3 :

Implémentation sur FPGA

3 IMPLEMENTATION FPGA :

3.1 INTRODUCTION :

Un FPGA (Field Programmable Gate Array) est un composant électronique constitué de milliers voir de millions de transistors connectés ensembles pour réaliser des fonctions logiques. Des fonctions logiques simples peuvent être réalisées telles que des soustractions ou additions. Des fonctions plus complexes peuvent être réalisées telles que le filtrage numérique du signal ou détection d'erreurs et correction. Aérospatial, aviation, automobile, radar, missiles, ordinateur… ne sont que quelques exemples des domaines ayant recours aux FPGAs.

- ✓ Les FPGA sont donc des réseaux logiques programmables sur site.

À l'aide de blocs logiques préconstruits et de ressources de routage programmables, c'est un circuit configurable afin de mettre en œuvre des fonctionnalités matérielles personnalisées, sans avoir jamais besoin d'utiliser de moyens matériels. Il suffit de développer des tâches de traitement numérique par la technologie des circuits reprogrammables.

- ✓ Les FPGA sont de ce fait une alternative innovante par rapport aux autres possibilités de contrôles conventionnels (ASIC, DSP, microprocesseur …).

Le traitement embarqué à l'aide de FPGA est devenu une partie intégrante d'un nombre croissant d'applications telles que les applications réseau industrielles et vidéo, ainsi que les systèmes de contrôle en boucle fermée sur les marchés industriels et de l'aérospatiale et de la défense. La valeur combinée des performances de traitement parallèle, de la réutilisation de la conception, de l'atténuation des risques de conception et de l'obsolescence, et de la réduction fonds (rendus encore plus attrayants par la marche continue vers des nœuds de processus de plus en plus petits) a alimenté cette tendance à une conclusion apparemment inévitable : le FPGA devient la plate-forme de choix pour la conception embarquée future sur ces marchés.

Il était donc tout à fait logique que nous prenions la décision d'implémenter notre régulateur sur une carte FPGA dont nous justifierons le choix par la suite.

3.2 ARCHITECTURE D'UN FPGA :

Structurés sous forme de matrices, les FPGA sont composés d'éléments logiques de base, constitués de portes logiques, présentes physiquement sur le circuit. Ces portes sont reliées par un ensemble d'interconnexions modifiables : d'où l'aspect programmable du circuit.

3.3 LE LANGAGE VHDL :

Le VHDL est un langage de description matériel HDL. Il répond à tous les critères établis pour un langage HDL.

Les plus importants points forts de ce langage sont :

3.3.1 LA SIMULATION ET LA SYNTHESE :

Le VHDL permet d'avoir des lignes de code pouvant être simulé (simulation fonctionnelle) dans le but de vérifier si le code obéit correctement à la fonction souhaitée, mais pour parvenir à une maquette réalisable ce n'est pas suffisant. D'où la nécessité de pouvoir synthétiser le même code. Lors de la synthèse, le compilateur traduit le premier code en un autre équivalent mais à un niveau d'abstraction plus bas. À ce niveau de synthèse, le compilateur traduit le code de haut niveau en portes logiques ceci en fonction du circuit logique programmable qui est ciblé. Le fichier synthétisé lui aussi, se doit d'être simulé (simulation événementielle) et vérifier si le compilateur (synthétiseur) est resté fidèle aux exigences de départ. Puis le fichier est compilé une deuxième fois pour aboutir au circuit logique qui sera implémenté. Cette fois encore, une troisième simulation (facultative) (simulation temporelle) pour être certain que le circuit est resté inchangé.

3.3.2 PORTABILITE :

Avec deux objectifs, portable vis-à-vis du circuit logique programmable, c'est-à- dire peut-être implémenté sur l'importe quel circuit logique programmable à condition d'avoir la capacité logique requise. Portable vis-à-vis du compilateur, pouvoir passer d'un compilateur à un autre et obtenir un même circuit en fin de processus.

3.3.3 UNE CONSTRUCTION HIERARCHIQUE :

Cette approche permet de simplifier la conception puisque les taches peuvent être divisées pour aboutir au point le plus simple possible puis faire un assemblage des blocs.

3.3.4 UNE DESCRIPTION FONCTIONNELLE :

Complémentaire de la précédente, la vision fonctionnelle apporte la puissance des langages de programmation. Tout algorithme est la description interne d'un bloc situé quelque part dans la hiérarchie du schéma complet. La vision structurelle est concurrente, la vision algorithmique est séquentielle, au sens informatique du terme [11]. Un programma VHDL doit être compris comme l'assemblage en parallèle des taches indépendantes qui s'exécutent concurremment.

3.3.5 STRUCTURE D'UN PROGRAMME VHDL (LE COUPLE ENTITE, ARCHITECTURE):

Un opérateur élémentaire, un circuit intégré, une carte électronique ou un système complet, est complètement défini par des signaux d'entrées et de sorties et par la fonction réalisée de façon interne [11]. Les concepteurs du VHDL ont adopté l'approche suivante : l'importe quel système est considéré comme une boite noire. Cette boite noire a des entrées et des sorties. Ils ont appelé la boîte noire « ENTITY ». L'importe quel système électronique effectue des opérations sur le signal d'entré pour donner le résultat du traitement en sortie. Ces opérations sont le contenu de la boîte noire ce contenu est appelé « ARCHITECTURE ».

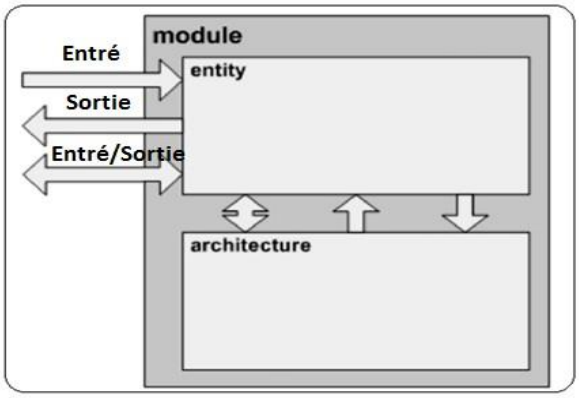


Figure 3. 1 : structure de base d'un module sous VHDL. [12]

3.3.5.1 Entity ou Entité:

Dans la partie déclarative de l'entité le circuit est décrit comment il est vu par l'extérieur ceci à travers les entrées, sorties et entrées/sorties.

```
Entity <entity name> Is Port(
   <signal name   : <signal direction> <data type>);
End <entity name>;
```

Figure 3. 2: Syntaxe déclarative de l'entité.

3.3.5.2 Architecture :

L'architecture décrit le comportement que doit avoir le circuit ou les opérations qu'il doit effectuer. Une architecture se doit toujours d'être attachée à une entité

(Ils vont de pair).

```
Architecture <architecture name> Of <entity name> Is
<Define signals and constants>
Begin
End <architecture name>;
```

Figure 3. 3: Syntaxe déclarative de l'architecture.

C'est dans cette section que le programme est rédigé. Un programme comporte essentiellement les éléments suivants : les signaux internes, opérateurs logiques (synchrone ou concurrent), les process. La description d'une architecture peut prendre trois formes :

Description comportementale : Ce type de description spécifie le comportement du composant ou du circuit à réaliser au moyen d'instructions séquentielles ou sous forme de flot de données (constituant un process).

Description structurelle : Dans ce type de description, les interconnexions des composants préalablement décrits sont énoncées. Cette description est la transcription directe d'un schéma.

Description mixte : Elle regroupe les deux descriptions décrites précédemment. À chaque entité peut être associée à une ou plusieurs architectures mais au moment de l'exécution (simulation, synthèse…) seulement une architecture et une seule est utilisée.

```
Library IEEE;
Use IEEE.std_logic_1164.all;

Entity <entity name> Is Port
  (<list of ports or design inputs and outputs>);
End <entity name>;

Architecture <architecture name> Of <entity name> Is
  <in this section define signals and constants>

Signal <signal name>          : Data Type;

Begin
<concurrent statements>

<process name>: Process (sensitivity list)
Begin
  <sequential statements>
End;

End <architecture name>;
```

Figure 3. 4 : Structure d'un programme sous VHDL.

3.3.6 LES TEST BENCHS :

La plupart des simulateurs logiques permettent de créer des stimuli pour vérifier le comportement d'un modèle. Souvent un langage de commande donne à l'utilisateur la possibilité d'automatiser, par la création de fichiers contenant des instructions de ce langage, l'enchaînement des opérations nécessaires aux tests. Le langage VHDL est un langage de modélisation et de simulation. Il contient tous les éléments nécessaires à la création de stimuli et surtout à l'exploitation des résultats.

Pour élaborer un TESTBENCH, il faut établir la liste des cas la plus complète que possible. Les stimuli sont les entrées appliquées au programme via le simulateur pour imiter le comportement des vraies entrées. Après exécution du simulateur, les sorties sont observées si elles sont comme prévu par le programme et ceci sans aucun risque sur le matériel (FPGA…).

3.4 CHOIX ENVIRONNEMENT ET CARTE :

Ce choix tient compte de deux facteurs :

- Le choix de la plateforme de développement
- Le choix de la carte cible

Deux grandes familles de fournisseurs de cartes FPGA dominent le marché des FPGA : Quartus (Altera, puis Intel) et Xilinx.

3.4.1 PRESENTATION DE LA SUITE LOGICIELLE XILINX :

Design Suite ISE (ISE signifie l'environnement de synthèse intégré) est un outil logiciel conçu pour la conception de HDL, l'analyse temporelle, et autres. Il a été abandonné en faveur de Vivado Design Suite depuis 2012. Ce produit est cependant suffisant pour implémenter notre régulateur. Cela explique que nous ayons choisi ce produit comme plate-forme de développement

3.4.2 PRESENTATION ET CHOIX DU LOGICIEL XILINX ISE 14.7:

La description de circuits numériques sous forme de schémas logiques, de machines à états finis ou en langages de description matériel (VHDL, Verilog, …) est faite sur des plates-formes.

Les étapes traditionnellement suivies sont :

- la compilation, la simulation comportementale,
- la synthèse, le placement routage et l'implémentation,
- la simulation temporelle et l'analyse de timing,
- la programmation sur les circuits programmables de Xilinx (CPLD et FPGA).

Le logiciel Xilinx ISE 14.7 est un logiciel de description, de simulation, et de programmation de circuits et systèmes numériques sur des composant programmables.

La suite ISE permet de suivre les différentes étapes précitées de façon simple et intuitive. Elle permet donc d'implémenter notre régulateur de façon complète.

Notre choix s'est donc porté sur elle.

3.4.3 CHOIX DE LA FAMILLE VIRTEX :

Virtex est la famille phare de produits FPGA développés par Xilinx . Les autres gammes de produits actuelles incluent Kintex (milieu de gamme) et Artix (à bas prix), chacune comprenant des configurations et des modèles optimisés pour différentes applications. En outre, Xilinx propose la série à faible coût Spartan, qui continue d'être mise à jour et approche de la production en utilisant la même architecture sous-jacente et le même nœud de processus que les plus grands appareils de la série 7.

Les FPGA Virtex sont généralement programmés dans des langages de description de matériel tels que VHDL ou Verilog , à l'aide du logiciel informatique Xilinx ISE ou Vivado Design Suite .

Les produits Xilinx FPGA ont été reconnus par EE Times, EDN et d'autres pour leur innovation et leur impact sur le marché.

3.4.4 CHOIX DE LA CARTE XC6VLX75T-2F484 :

Figure 3. 5: carte XC6VLX75T-2F488.

3.5 IMPLEMENTATION :

Tout le travail effectué dans ce mémoire consiste principalement à écrire les programmes VHDL qui décrivent le fonctionnement des différentes parties telles que décrites. Des programmes testbenchs ont été écrits pour tester les différents composants (components) et l'ensemble components

Ces programmes décrivent le fonctionnement des trois parties principales :

- reg_temp.vhdl :

C'est le module principal qui lance le programme de régulation.
Au départ il effectue une instanciation des modules MLIx, sys_affi, mode_nor_pro, et la séquence suivante de code VHDL :

temper<=Dadc when fc='1';

mode_nor_pro1: mode_nor_pro port

map(prog=>prog,plus=>plus,moins=>moins,store=>store,clk=>clk,consigne=>con

signe,mode=>mode,reset=>reset,consigneT=>consigneT);

sys_aff1: sys_aff port map(temper=> temper,consigneT=>consigneT,clk=>clk,mode=>mode,re

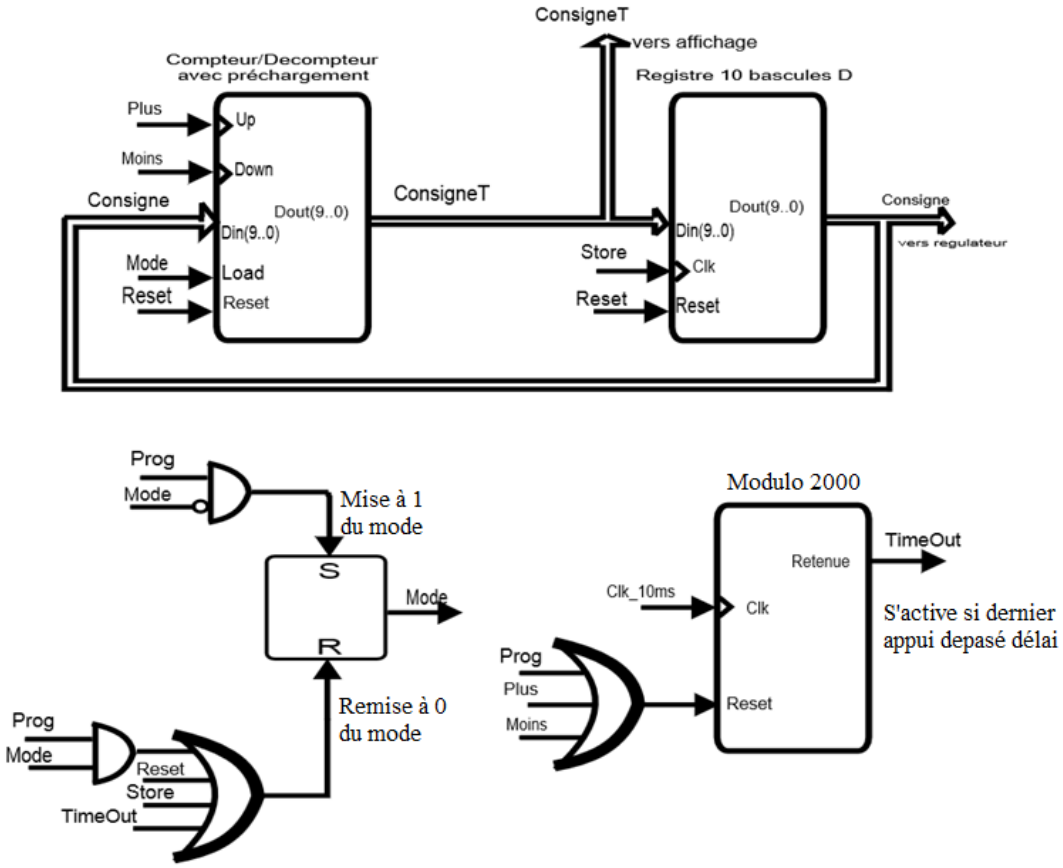
set=>reset,code_7seg=>code_7seg,cmd_dig=>cmd_dig);

MLIx1: MLIx port map(temper=> temper,consigne=>consigne,clk=>clk,mli=>mli,reset=>reset); régle le fonctionnement de l'ensemble.

Le code de départ << temper<=Dadc when fc='1' >> charge la sortie Dadc du convertisseur dans la variable temper (température) et le cycle commence.

On voit clairement que les instances mode_nor_pro1, sys_aff1, MLIx1 sont mappées convenablement et le système fonctionne parfaitement.

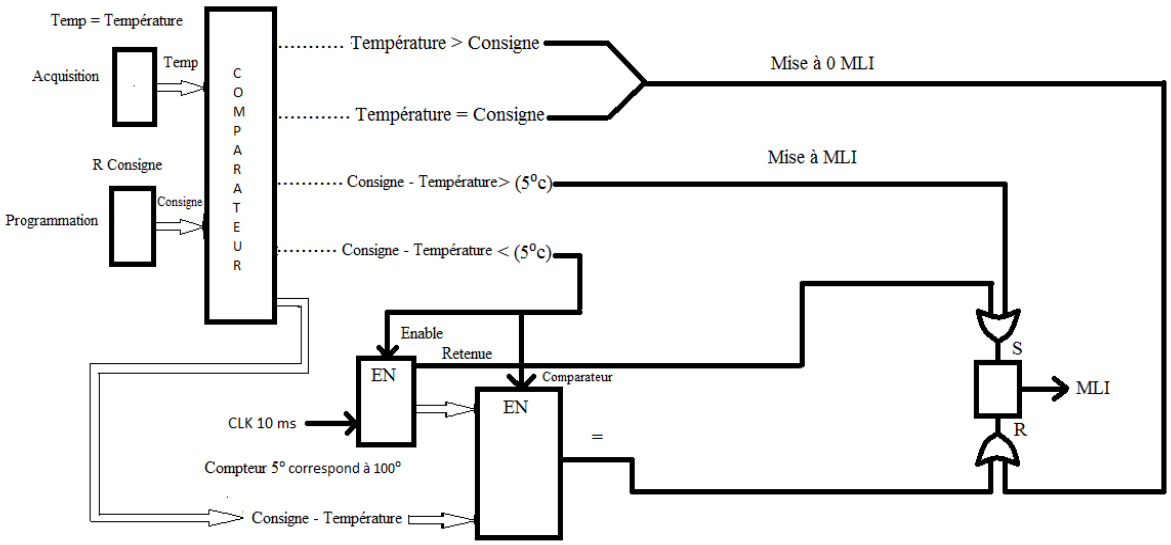
- Mode_prog.vhd :



Le détail de fonctionnement du programmateur a été longuement expliqué dans la section correspondante du **chapitre 2** intitulé « Description détaillée du schéma proposé » **Figure 2.6**.

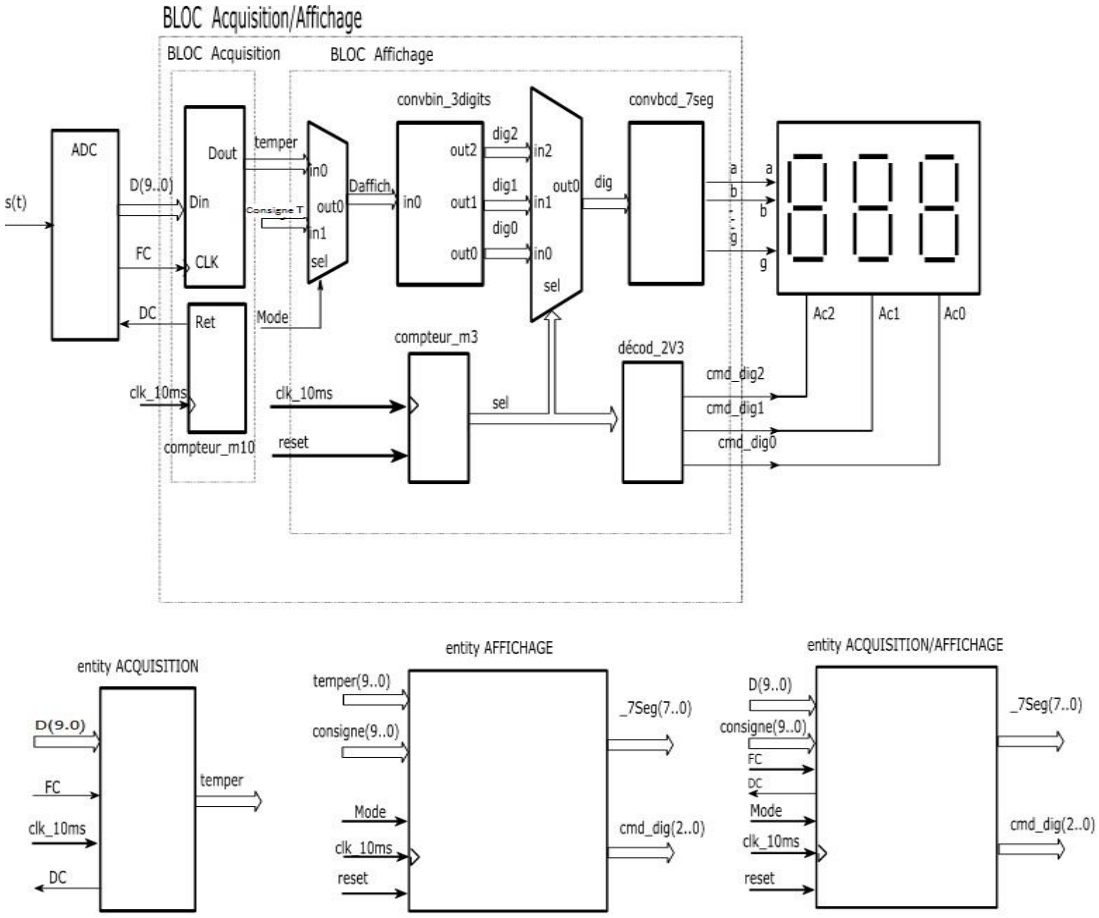
- MLI.vhd :

Le détail du fonctionnement du module a été longuement expliqué dans la section correspondante du **chapitre 2** intitulé « Description détaillée du schéma proposé » **Figure 2.5**.



- Exemple **MLI** dans différentes configurations

➢ sys_aff.vhd :



Le détail du fonctionnement du module a été longuement expliqué dans dans la section correspondante du **chapitre 2** intitulé « Description détaillée du schéma proposé » **Figure 2.1**.

3.5.1 COMPILATION DES CODES VHDL

Comme ci-joint indiqué dans les différentes captures d'écran, les différents programmes ont été vérifiés, compilés et implémentés dans toutes les étapes dans Xilinx ISE 14.7.

Les résultats sont affichés :

- La partie implémentation indique les programmes membres du projet régulateur temp (notre mémoire).
- On y voit clairement le succès de la compilation, indiqué sur le panel gauche (signes OK verts) et le panel droit (No Errors)

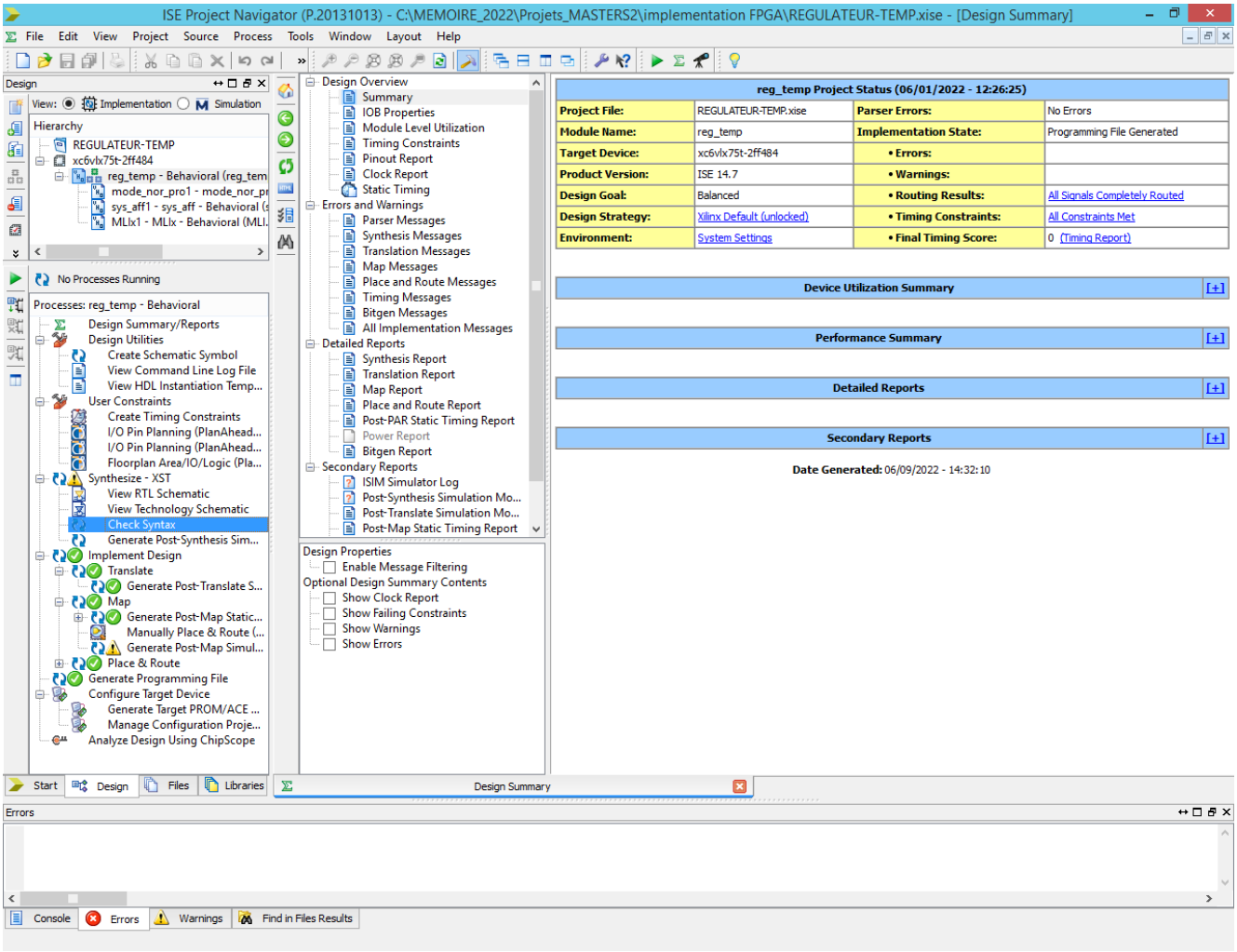


Figure 3. 6 : Résultats "No errors".

Les deux captures suivantes donnent les ressources utilisés (première partie puis deuxième partie)

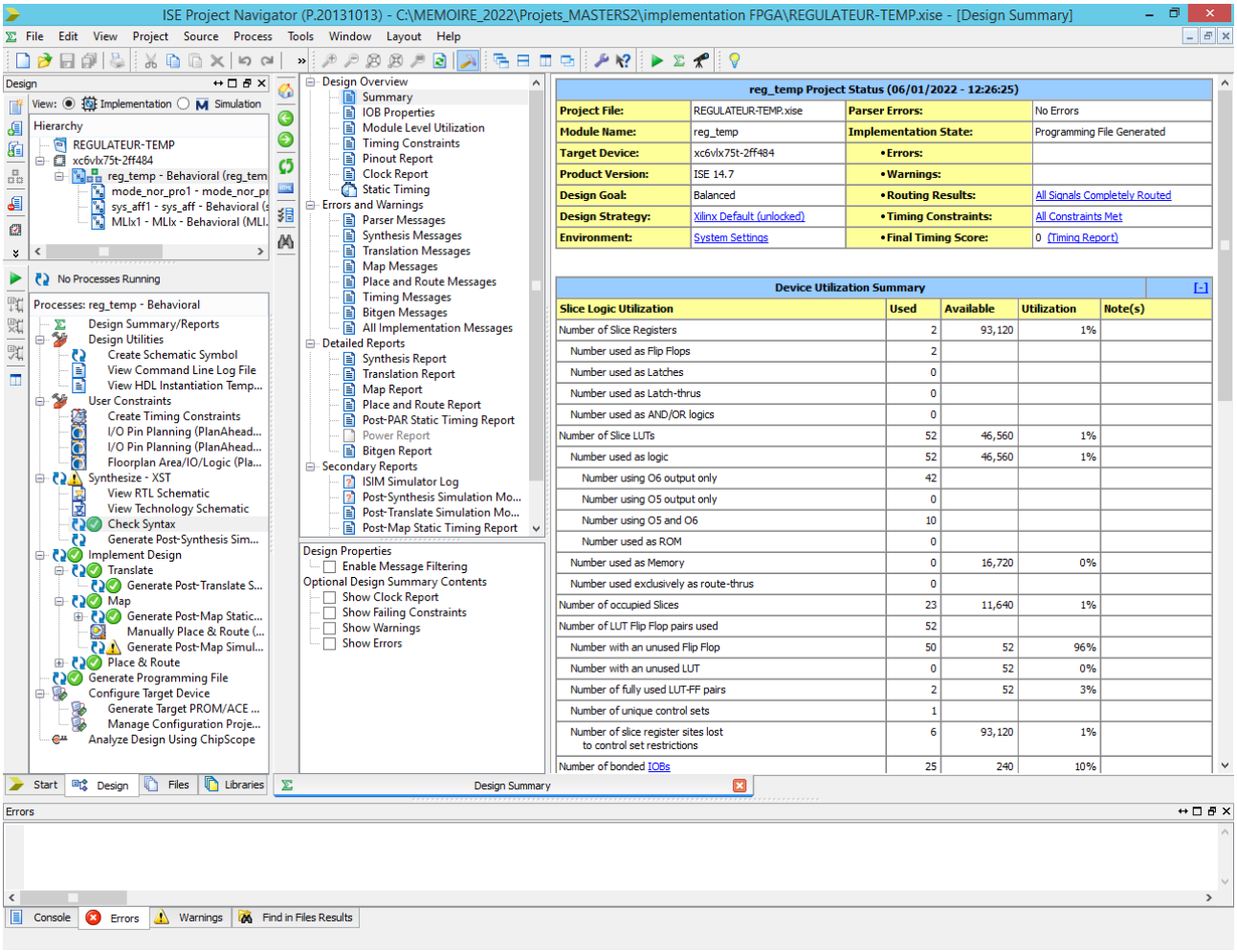
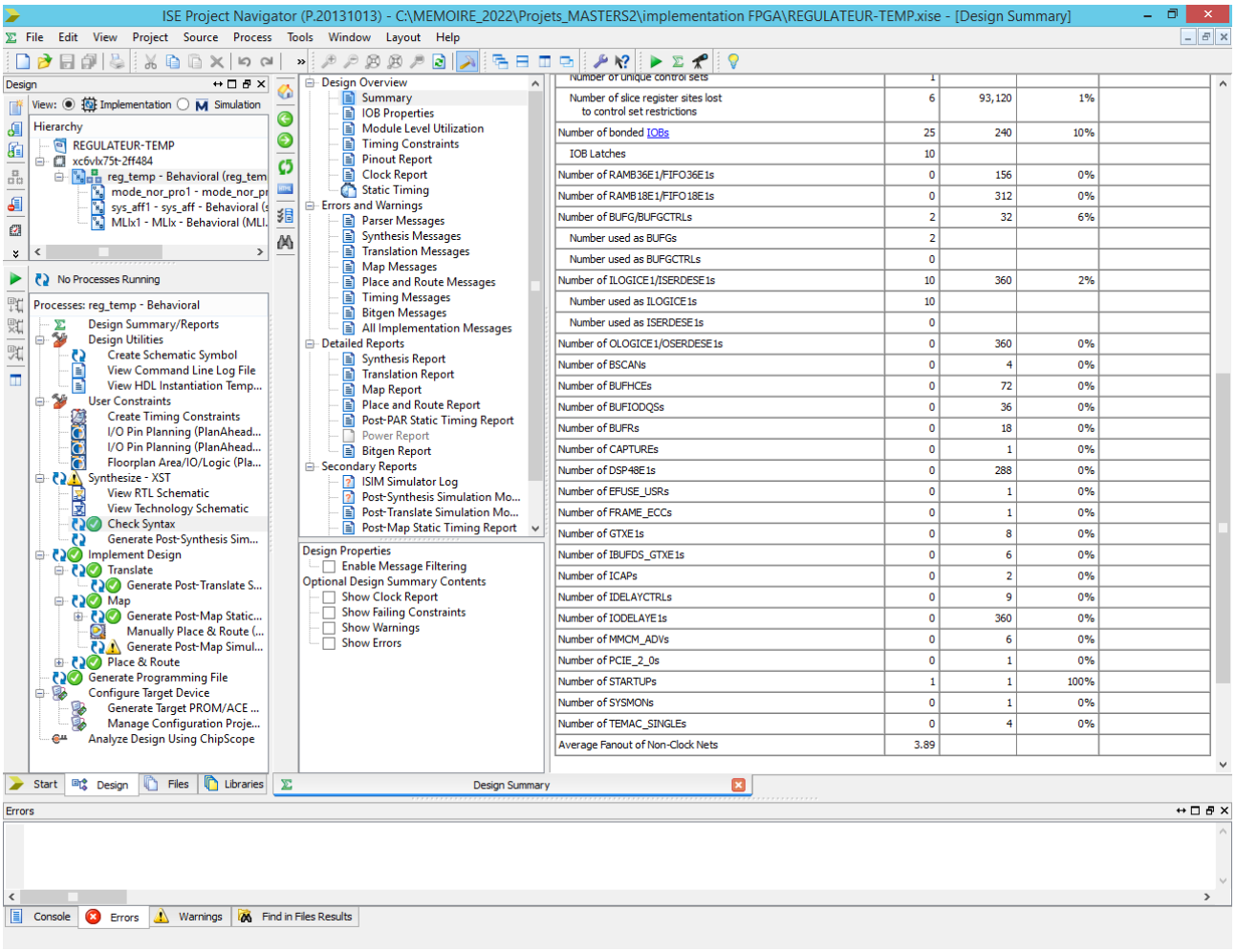


Figure 3. 7 : Les ressources utilisés "1er partie".



Number of unique control sets	1			
Number of slice register sites lost to control set restrictions	6	93,120	1%	
Number of bonded IOBs	25	240	10%	
IOB Latches	10			
Number of RAMB36E1/FIFO36E1s	0	156	0%	
Number of RAMB18E1/FIFO18E1s	0	312	0%	
Number of BUFG/BUFGCTRLs	2	32	6%	
Number used as BUFGs	2			
Number used as BUFGCTRLs	0			
Number of ILOGICE1/ISERDESE1s	10	360	2%	
Number used as ILOGICE1s	10			
Number used as ISERDESE1s	0			
Number of OLOGICE1/OSERDESE1s	0	360	0%	
Number of BSCANs	0	4	0%	
Number of BUFHCEs	0	72	0%	
Number of BUFIODQSs	0	36	0%	
Number of BUFRs	0	18	0%	
Number of CAPTUREs	0	1	0%	
Number of DSP48E1s	0	288	0%	
Number of EFUSE_USRs	0	1	0%	
Number of FRAME_ECCs	0	1	0%	
Number of GTXE1s	0	8	0%	
Number of IBUFDS_GTXE1s	0	6	0%	
Number of ICAPs	0	2	0%	
Number of IDELAYCTRLs	0	9	0%	
Number of IODELAYE1s	0	360	0%	
Number of MMCM_ADVs	0	6	0%	
Number of PCIE_2_0s	0	1	0%	
Number of STARTUPs	1	1	100%	
Number of SYSMONs	0	1	0%	
Number of TEMAC_SINGLEs	0	4	0%	
Average Fanout of Non-Clock Nets	3.89			

Figure 3. 8 : Les ressources utilisés "2éme partie".

La capture d'écran suivante montre le composant de régulation complet, généré par ISE 14.7, selon les codes VHDL et les instanciations réalisées.

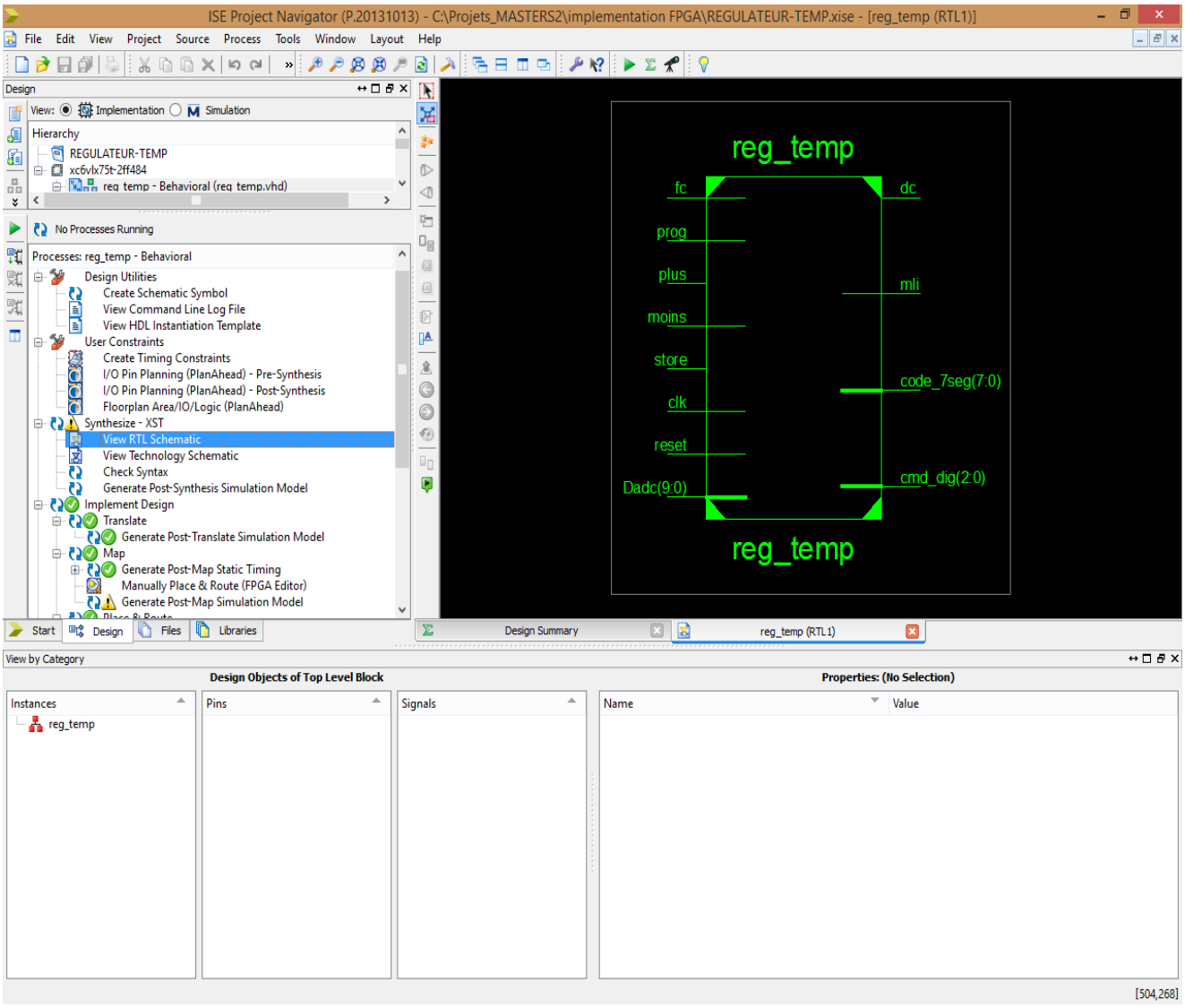


Figure 3. 9 : Le composant de régulation complet "généré par ISE 14.7".

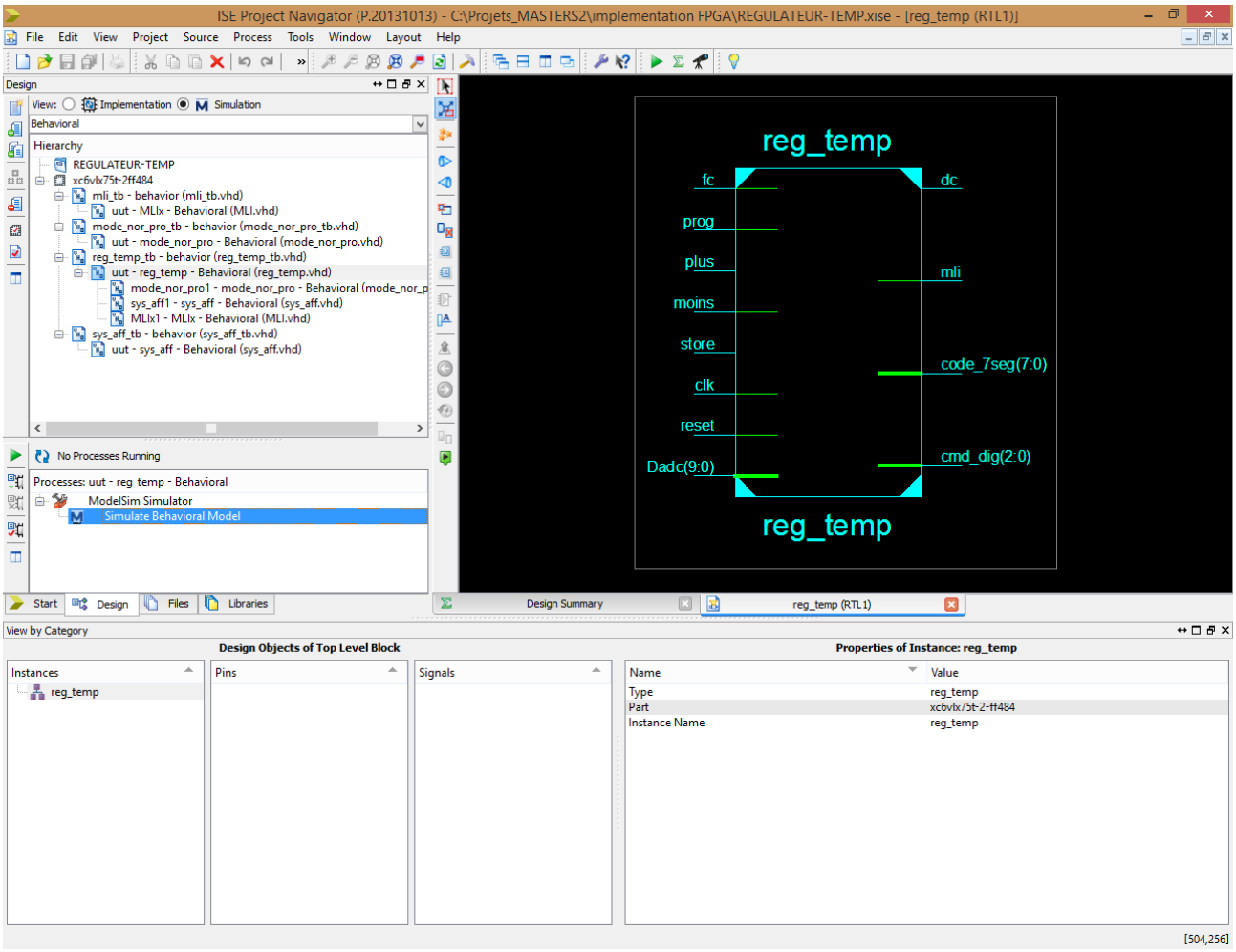


Figure 3. 10 : Reg_Temp (généré par ISE 14.7).

3.6 CONCLUSION

Dans ce chapitre, Les VHDL écrits répondent parfaitement à notre conception. Ils ont été compilés et synthétisés avec succès, ce qui atteste que notre travail a abouti parfaitement et complètement.

Chapitre 4 :

Simulations et Résultats

4 SIMULATION :

Des programmes de testbenchs destinés à simuler le fonctionnement des quatre parties principales ont été également écrits :

- reg_temp_tb.vhd
- mode_nor_pro_tb.vhd
- mli_tb.vhd
- sys_aff_tb.vhd

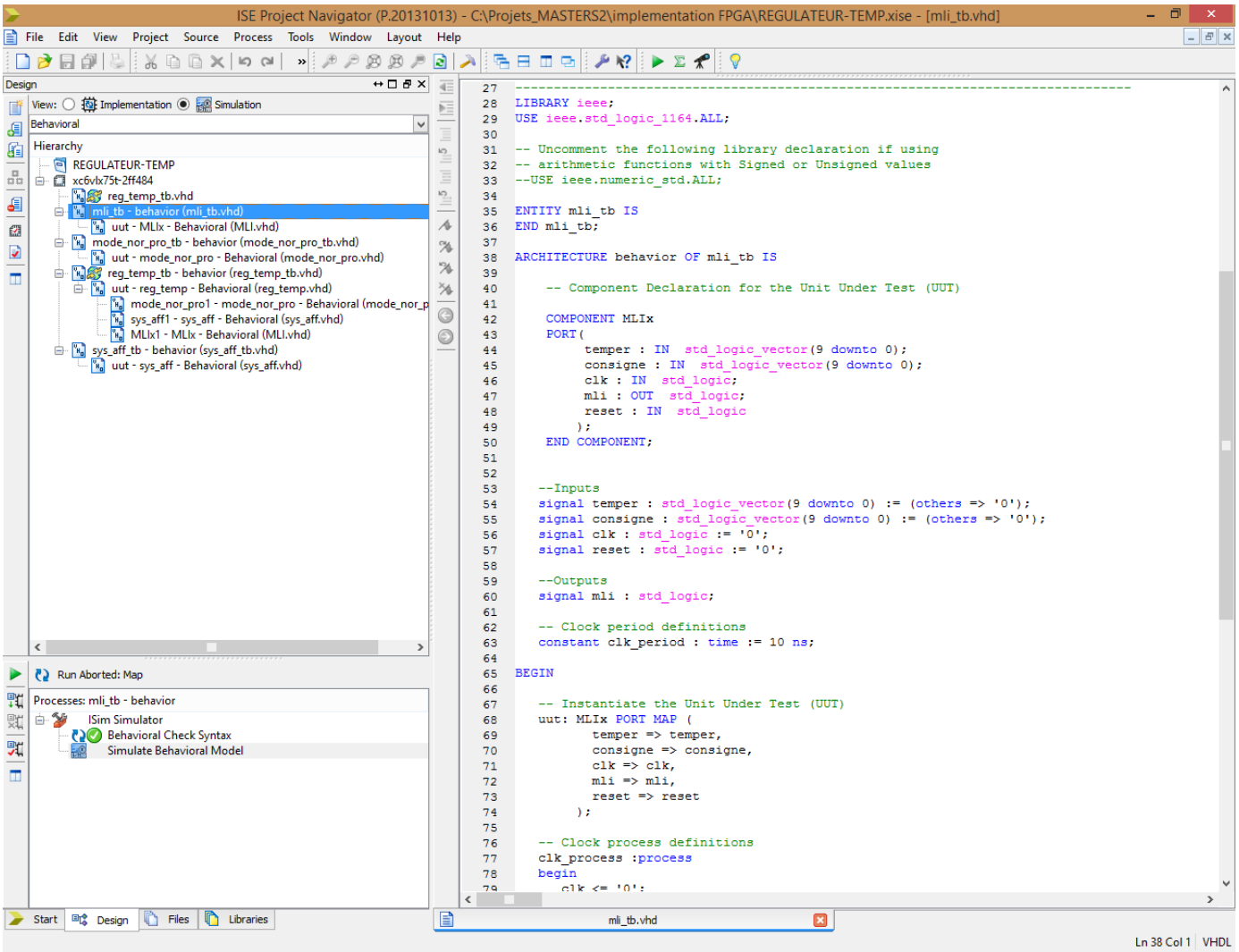


Figure 4. 1 : mli_tb.vhdl.

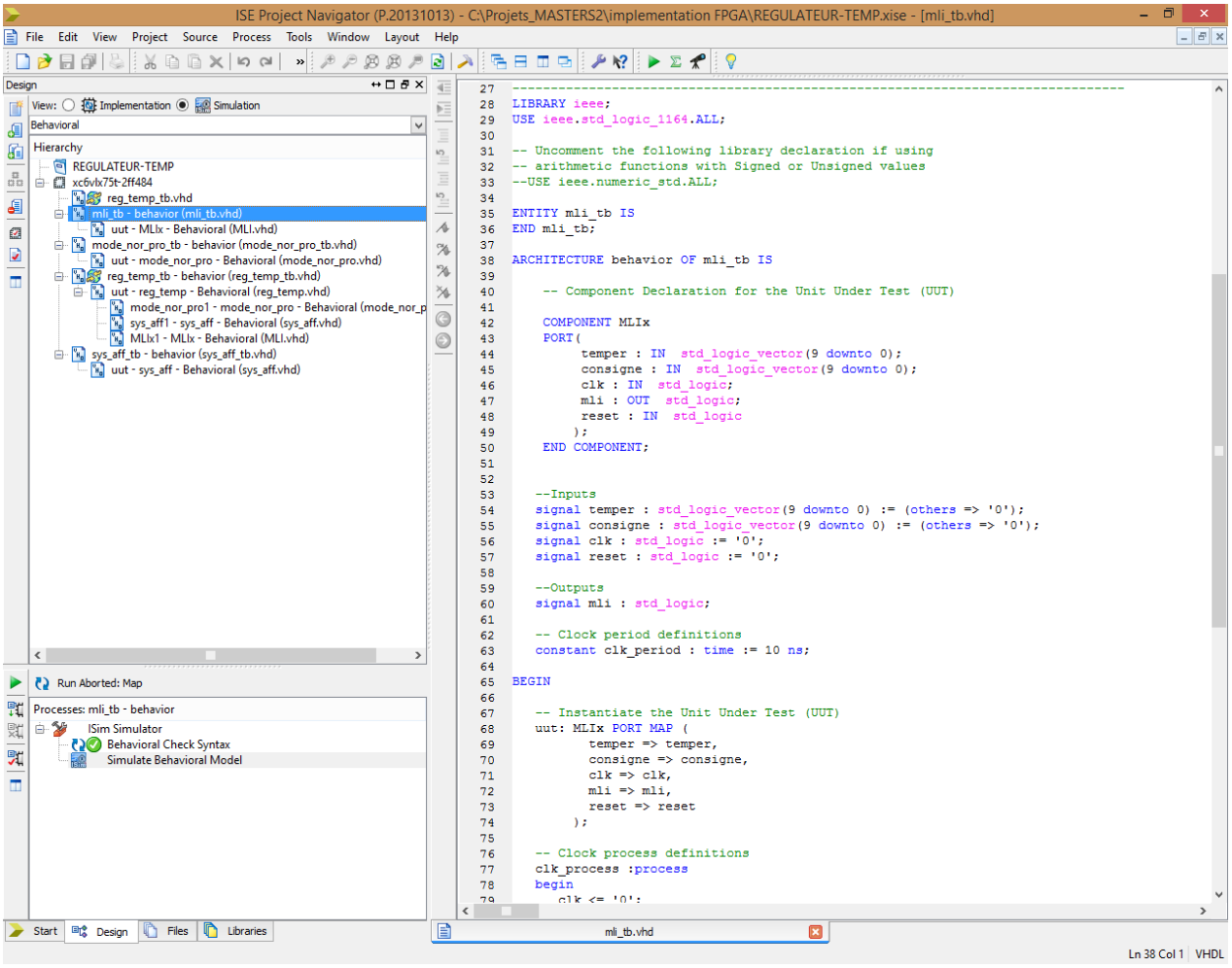
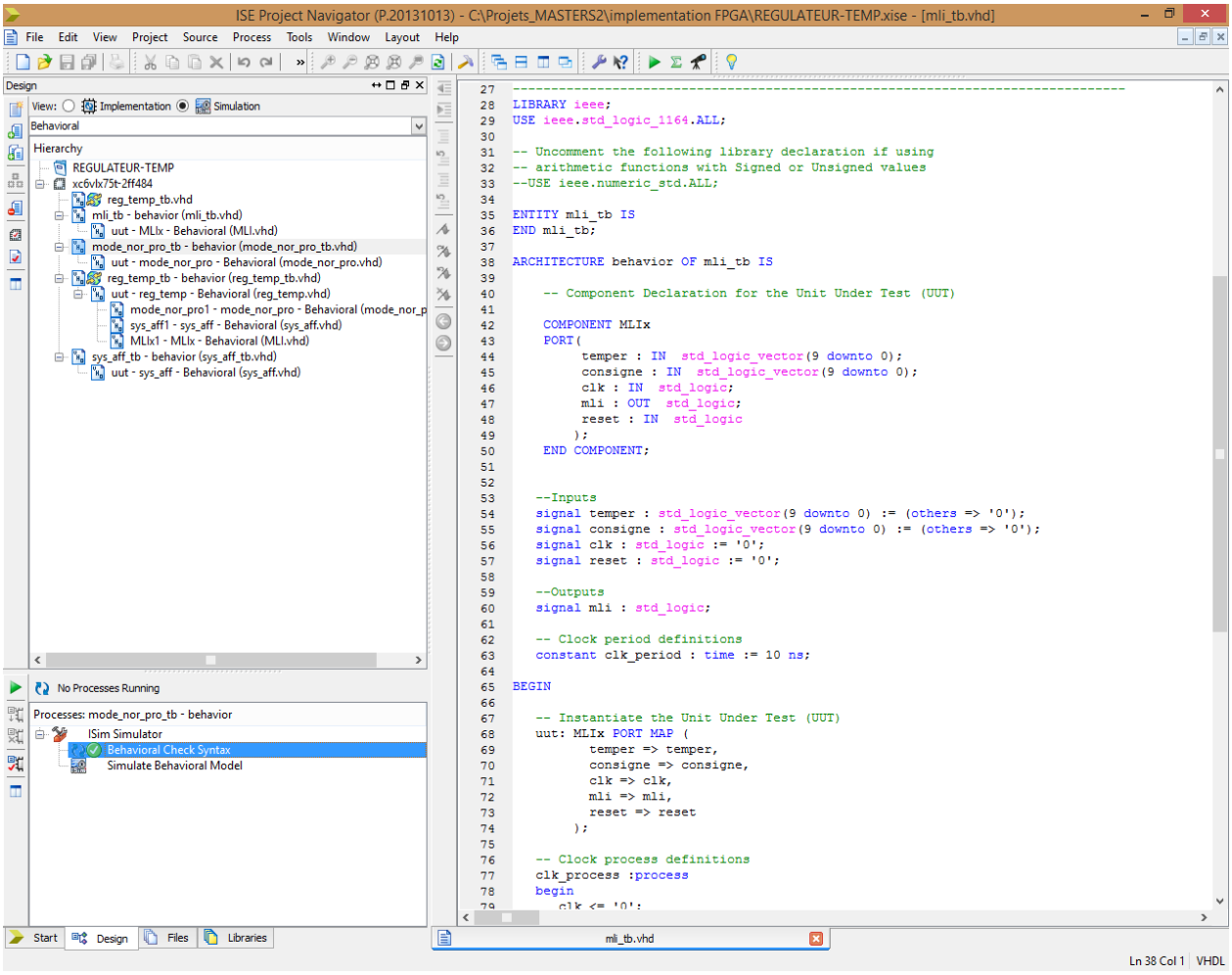


Figure 4. 2 : Vérification MLI-TB.




```
27  --------------------------------------------------------------------------------
28  LIBRARY ieee;
29  USE ieee.std_logic_1164.ALL;
30
31  -- Uncomment the following library declaration if using
32  -- arithmetic functions with Signed or Unsigned values
33  --USE ieee.numeric_std.ALL;
34
35  ENTITY mli_tb IS
36  END mli_tb;
37
38  ARCHITECTURE behavior OF mli_tb IS
39
40      -- Component Declaration for the Unit Under Test (UUT)
41
42      COMPONENT MLIx
43      PORT(
44           temper : IN  std_logic_vector(9 downto 0);
45           consigne : IN  std_logic_vector(9 downto 0);
46           clk : IN  std_logic;
47           mli : OUT  std_logic;
48           reset : IN  std_logic
49          );
50      END COMPONENT;
51
52
53     --Inputs
54     signal temper : std_logic_vector(9 downto 0) := (others => '0');
55     signal consigne : std_logic_vector(9 downto 0) := (others => '0');
56     signal clk : std_logic := '0';
57     signal reset : std_logic := '0';
58
59     --Outputs
60     signal mli : std_logic;
61
62     -- Clock period definitions
63     constant clk_period : time := 10 ns;
64
65  BEGIN
66
67     -- Instantiate the Unit Under Test (UUT)
68     uut: MLIx PORT MAP (
69            temper => temper,
70            consigne => consigne,
71            clk => clk,
72            mli => mli,
73            reset => reset
74          );
75
76     -- Clock process definitions
77     clk_process :process
78     begin
79        clk <= '0';
```

Figure 4. 3 : Mode_nor_pro_tb.vhdL.

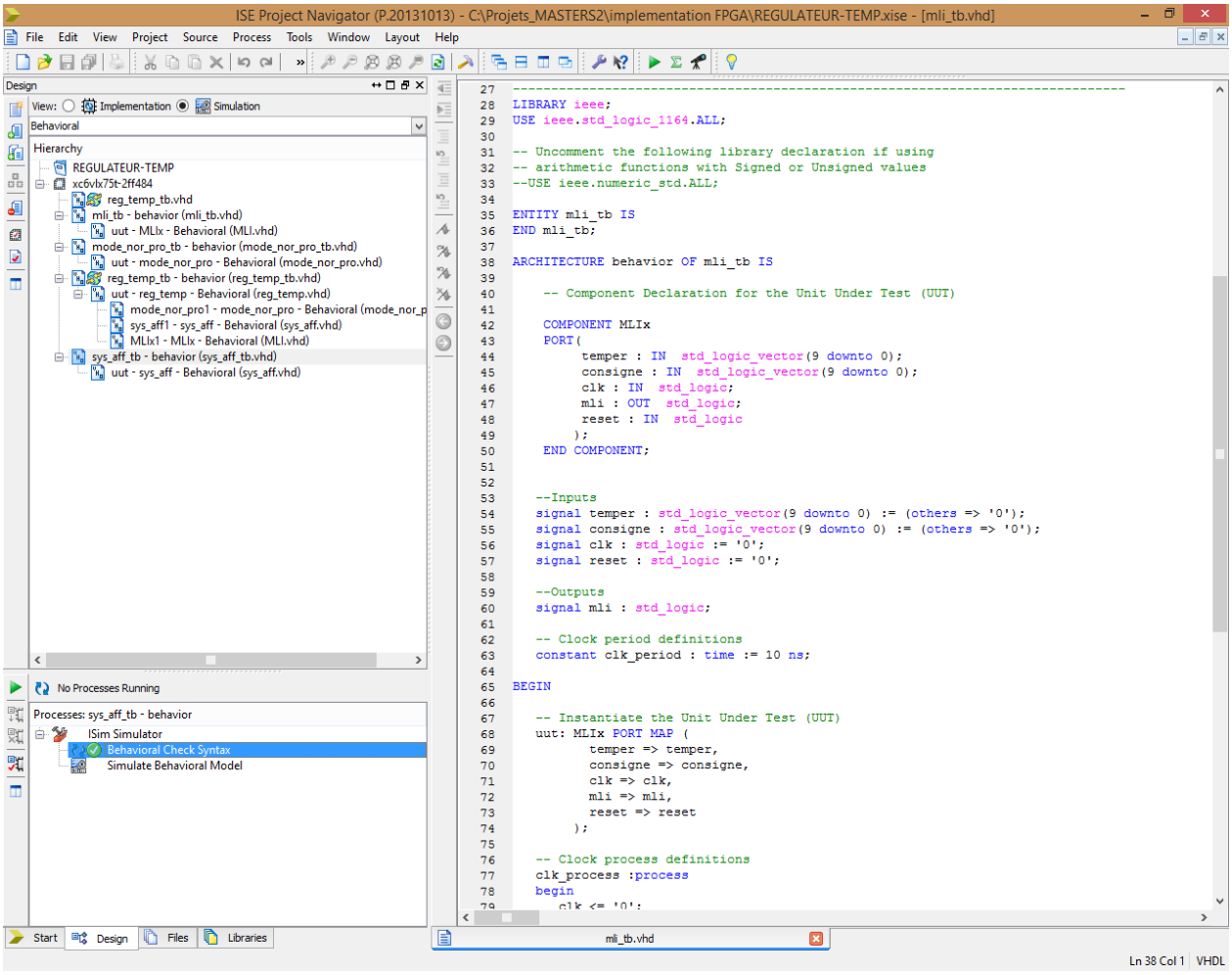


Figure 4. 4 : sys_aff_tb.vhdl.

Ces programmes ont consisté principalement à définir les signaux et les attentes entre autres :

- ❖ Une horloge (10 ms).
- ❖ Une consigne (20°c).
- ❖ Des températures de 10°c,15°c,16°c,17°c,20°c,21°c.
- ❖ Attendre des délais suffisants avant de lancer les différents cycles de régulation correspondants aux différentes températures.
- ❖ Le choix des valeurs de la température vise à voir la durée du MLI pour les différents cas inférieurs à la consigne. Le cas (21°c) est là pour montrer la mise à zéro du MLI.

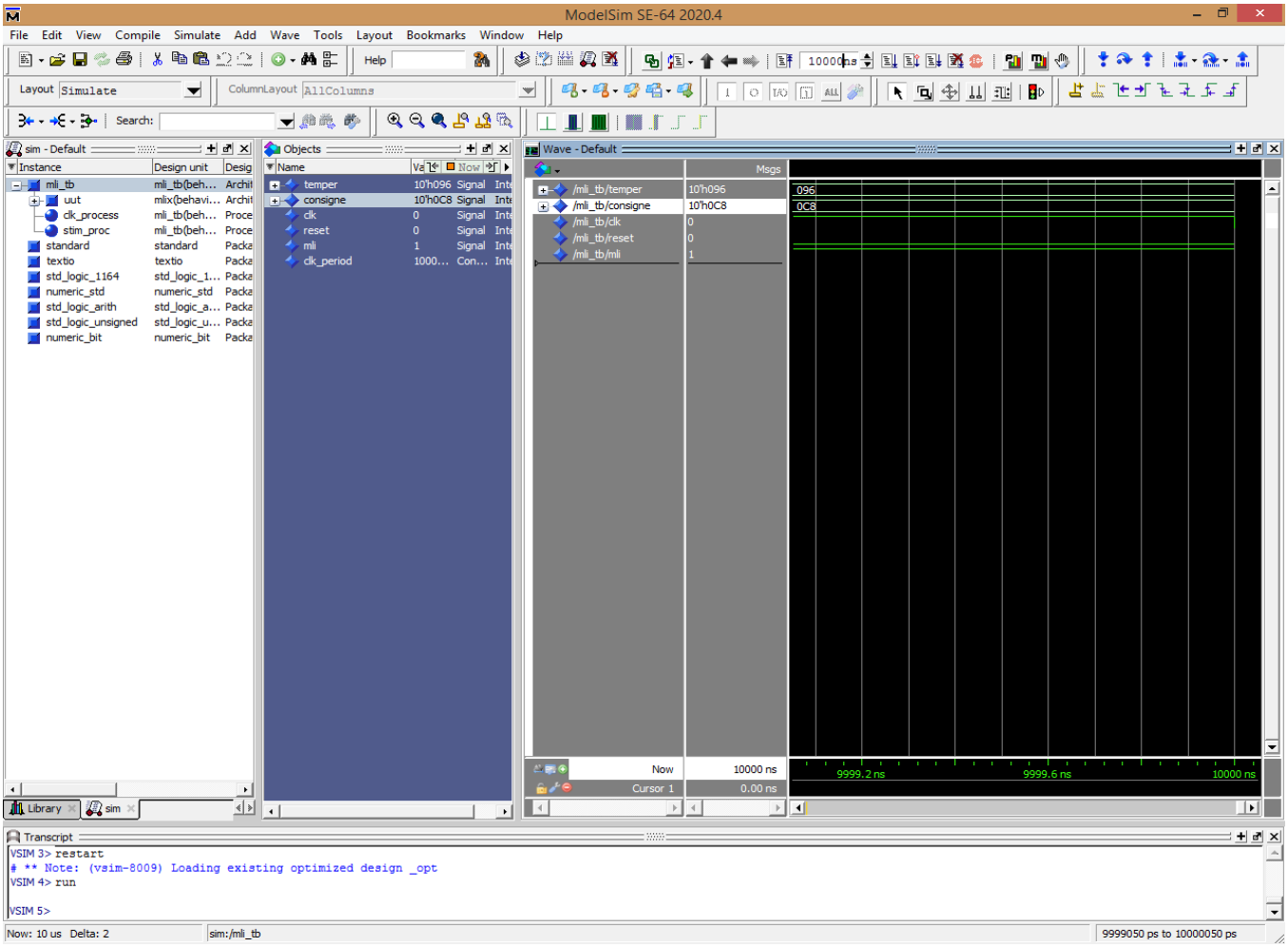


Figure 4. 5 : ModelSIM : MLI à 1.

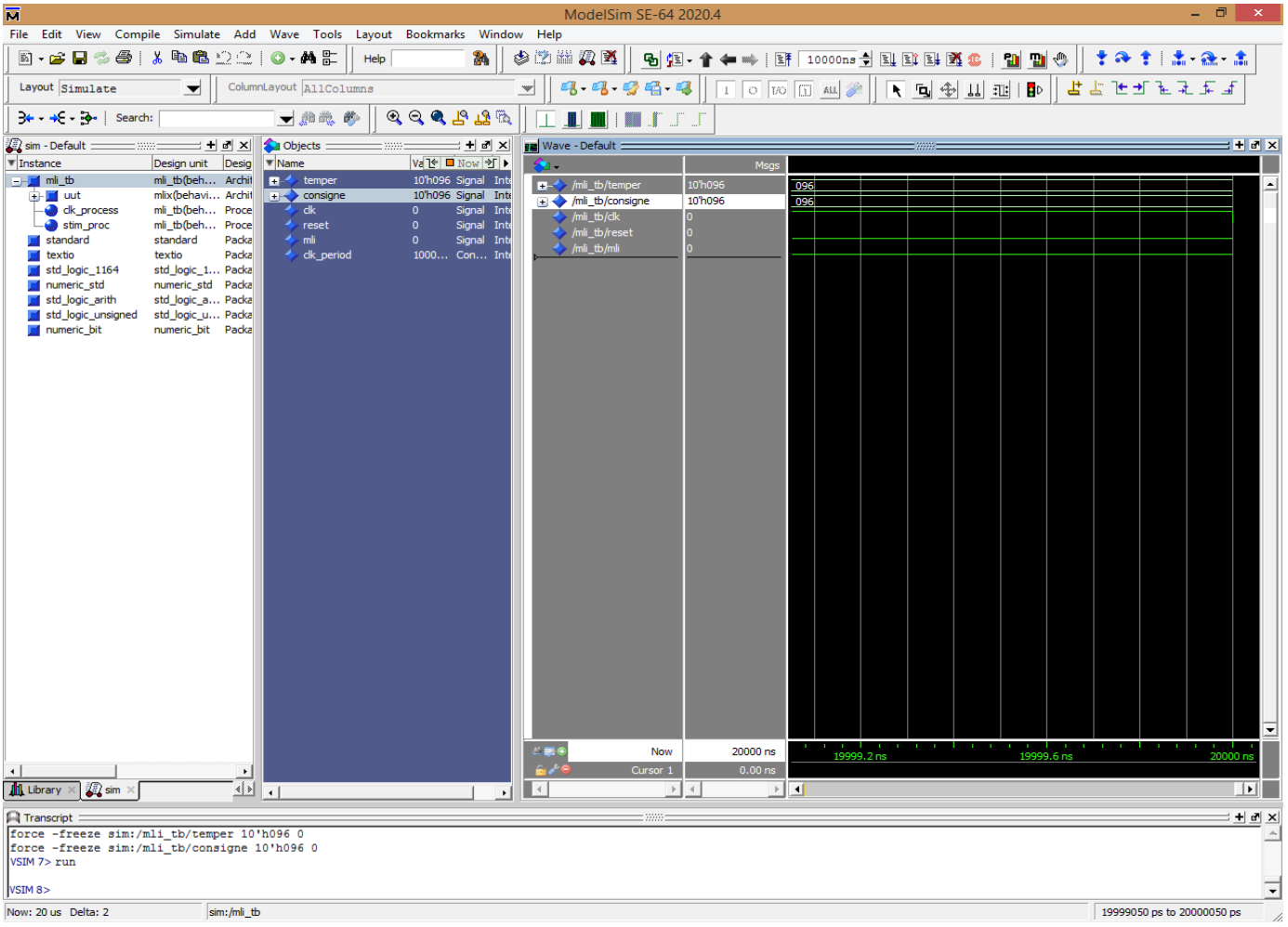


Figure 4. 6 : MLI à zéro égalité temp consigne.

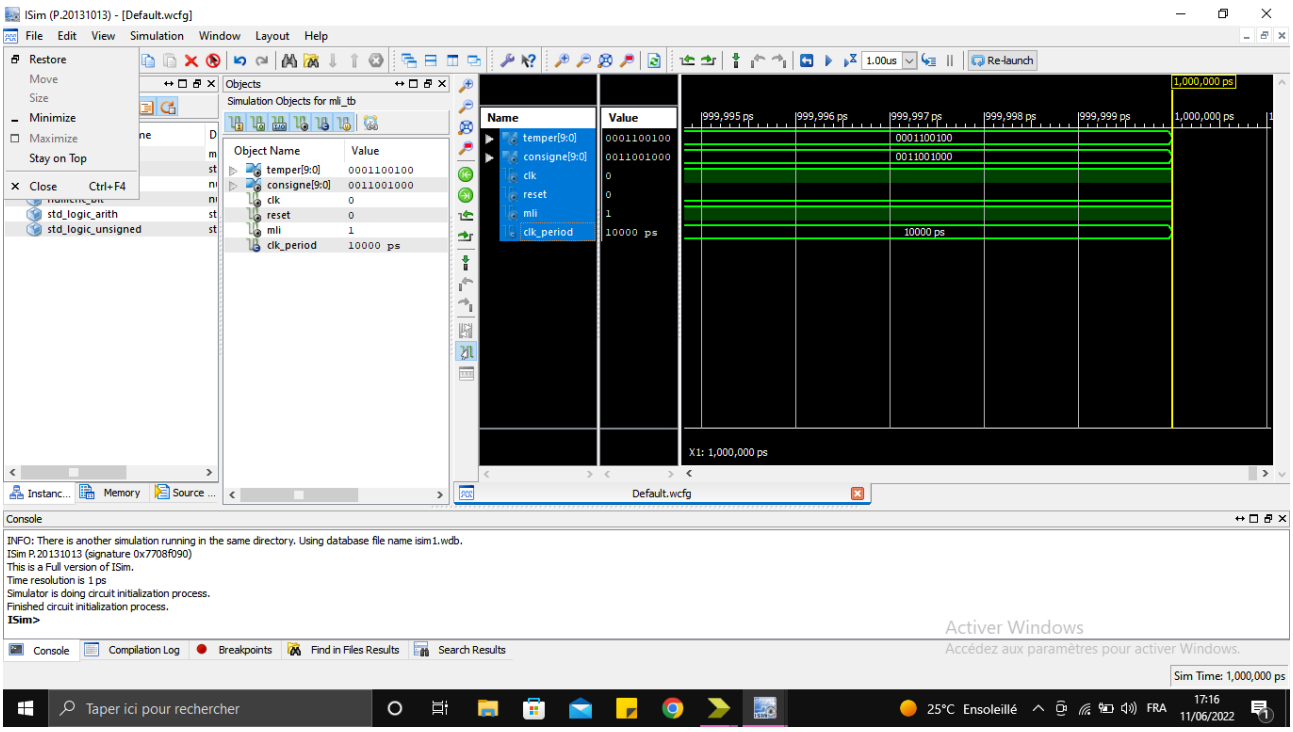


Figure 4. 7 : MLI à 1 simulateur ISIM intégré.

4.1 RESULTATS ET CONCLUSION SUR LA SIMULATION:

Les résultats de simulation obtenus valident la conception et l'implémentation. Ils attestent que le composant REG_TEMP est prêt à l'emploi comme composant complet et autonome.

Le simulateur MODELSIM 2020. qui est la référence en la matière a donné pour notre régulateur des résultats satisfaisants, prouvant que la conception est cohérente et la programmation VHDL bien menée de bout en bout.

Nous avons pourtant utilisé ISIM, le simulateur intégré de ISE 14.4 pour rester dans le même environnement.

Les deux simulateurs donnent des résultats corrects et cohérents entre eux, ce qui permet de conclure à la robustesse du code généré dans les testbenchs.

Conclusion générale

Ce travail est accompli de bout en bout avec succès.

Nous avions pour objectif d'étudier un régulateur, de valider sa conception par un environnement professionnel et de le simuler pour valider son fonctionnement en tenant compte des impératifs de timing qui sont importants dans l'aspect temps réel.

L'objectif est atteint, ce qui fait de ce travail un effort ayant réussi.

Nous signalons toutefois les perspectives possibles qui peuvent servir à améliorer cd travail et à le compléter.

Parmi cela, signalons que le MLI, pris en charge, peut être utilisé pour compléter le schéma par une action de régulation PID ou autre. L'effort à faire en ce sens ne met pas en cause notre travail, puisqu'il s'agit d'un complément et non d'une reformulation de schéma.
De plus, comme vu dans l'implémentation, La carte VIRTEX choisie est très riche en ressources et ces ressources sont assez faiblement utilisées (1% à 10% en général).

Tout travail exigeant des ressources supplémentaires, pourra se faire sans changer de carte, car les ressources non utilisées sont immenses.

BIBIOGRAPHIE

[1] P.PROUVOT, « Instrumentation régulation en 30 fichiers », édition Dunod, Paris, France, 2010.

[2] D.LEQUESNE, « la régulation PID analogique et numérique », édition Eyrolles, France, 2006.

[3] S.Sekhsokh et K.Oukili « Etude d'une boucle de régulation de niveau », mémoire de master en génie des procédés, école supérieure de technologie _ FFS _ 2010/2011.

[4] J.Albet, « banc de régulation », Manuel technique, ENSIACET A7, T.P 2éme année, janvier 2007.

[5] M.Kareb et L.Gadi, « Etude et implémentation d'une commande PID en utilisant le PIC18F2550 pour le contrôle de vitesse d'un MCC », mémoire de master en automatique, UMMTO, Tizi-Ouzou,2016.

[6] E.Magarotto, « cours de régulation », département de génie chimique et procédés, université de Caen, France, version septembre 2004.

[7] C.Bessieres, « Système asservis analogique et échantillonnés », cours, France.

[8] W.Aous et G.Sedaoui « conception et réalisation d'un régulateur PID numérique à base d'un microcontrôleur PICI16F877A », mémoire de master en électronique industrielle, UMMTO, Tizi-Ouzou, 2014.

[9] M.Correvon « Systèmes électroniques », cours les régulateurs standards, chapitre7, institut d'automatisation industrielle, Suisse Occidentale, Suisse.

[10] https://www.techno-science.net/definition/6725.html

[11] WEBER, J. (2000). Le language VHDL. PARIS: DUNOD.

[12] KAFIG, W. (2011). VHDL 101. OXFORD: ELSEVIER.

BIBLIOGRAPHIE