

École Polytechnique de l'Université de Tours  
64, Avenue Jean Portalis  
37200 TOURS, FRANCE  
Tél. +33 (0)2 47 36 14 14  
Fax +33 (0)2 47 36 14 22  
[www.polytech.univ-tours.fr](http://www.polytech.univ-tours.fr)

**Spécialité Informatique Industrielle**  
**4<sup>ème</sup> année**  
**2014-2015**

**Compte rendu TP3**

**CoDesign**

Apprenti :  
**Thomas GACHE, Thibault ARTUS**  
[thomas.gache@yahoo.fr](mailto:thomas.gache@yahoo.fr), [theskullmachine@gmail.com](mailto:theskullmachine@gmail.com)

Tuteur :  
**Alexis ROLLAND**  
[alexis.rolland@univ-tours.fr](mailto:alexis.rolland@univ-tours.fr)  
Polytech'Tours



# Table des matières

---

<b>1</b>	<b>Présentation</b>	<b>5</b>
1.1	Ressources nécessaires . . . . .	5
1.2	Présentation des outils utilisés . . . . .	5
1.3	Présentation du mode d'emploi . . . . .	5
<b>2</b>	<b>Mise en place du projet Quartus</b>	<b>6</b>
<b>3</b>	<b>Mise en place du projet Eclipse NBT</b>	<b>9</b>
3.1	Validation du fonctionnement hardware . . . . .	9
3.2	Mise en place des bibliothèques contenant les fonctions I2C . . . . .	9

# Table des figures

---

2.1	Partie Hardware	6
2.2	Paramètres Clock Source	6
2.3	Sélection Nios II	7
2.4	Paramètres RAM	7
2.5	Paramètres des vecteurs d'interruption	7
2.6	Paramètres Timer	8
2.7	Ajouts des fichiers de synthèse	8
3.1	Arborescence des projets	9

# CHAPITRE 1

## Présentation

---

### 1.1 Ressources nécessaires

- Ordinateur avec les outils Altera Quartus II (13.1) et Eclipse Nios II software Build Tools for Eclipse
- Carte DE2-115
- Ressources documentaires Altera

### 1.2 Présentation des outils utilisés

Le module attaché à Quartus II permettant de concevoir la partie matérielle du design :

- **QSYS** : Outil destiné à préparer de manière graphique le processeur (au sens coeur de processeur plus périphériques et ressources nécessaires). Il permet de des réglages fins et de réaliser réellement un processeur sur mesure.

Une fois le processeur synthétisé et téléchargé dans le FPGA, il reste à passer à la partie software, c'est à dire développer et implanter le programme dans ce processeur.

Le module attaché à Eclipse permettant de concevoir la partie logicielle du design :

- **NIOS II Software Build Tools** : Quartus intègre un module logiciel basé sur l'EDI Eclipse permettant de gérer des programmes complexes, de télécharger et déverminer ceux-ci sur une cible NIOS II.

### 1.3 Présentation du mode d'emploi

L'objectif de ce mode d'emploi est de mettre en oeuvre une mémoire I2C externe pour stockage de données non volatile lié au processeur NIOS II et/ou à la carte DE2-115.

# CHAPITRE 2

## Mise en place du projet Quartus

Les répertoires de travail sont hiérarchisés de la façon suivante :

On a lancé le module logiciel **QSYS** permettant la programmation hardware de notre FPGA.  
La partie hardware est implémentée comme ci-dessous :

Use	Connections	Name	Description	Export	Clock	Base	End	IRQ
<input checked="" type="checkbox"/>		ClockSource	Clock Source		exported			
		clk_in	Clock Input	clk				
		clk_in_reset	Reset Input	reset				
		clk	Clock Output	Double-click to export	ClockSource			
		clk_reset	Reset Output	Double-click to export				
<input checked="" type="checkbox"/>		MainCPU	Nios II Processor					
		clk	Clock Input	Double-click to export	ClockSource			
		reset_n	Reset Input	Double-click to export	[clk]			
		data_master	Avalon Memory Mapped Master	Double-click to export	[clk]			
		instruction_master	Avalon Memory Mapped Master	Double-click to export	[clk]			
		jtag_debug_module_r...	Reset Output	Double-click to export	[clk]			
		jtag_debug_module	Avalon Memory Mapped Slave	Double-click to export	[clk]			
		custom_instruction_m...	Custom Instruction Master	Double-click to export	[clk]			
<input checked="" type="checkbox"/>		onchipRAM	On-Chip Memory (RAM or ROM)					
		clk1	Clock Input	Double-click to export	ClockSource			
		s1	Avalon Memory Mapped Slave	Double-click to export	[clk1]			
		reset1	Reset Input	Double-click to export	[clk1]			
<input checked="" type="checkbox"/>		jtag_uart	JTAG UART					
		clk	Clock Input	Double-click to export	ClockSource			
		reset	Reset Input	Double-click to export	[clk]			
		avalon_jtag_slave	Avalon Memory Mapped Slave	Double-click to export	[clk]			
<input checked="" type="checkbox"/>		SystemTimer	Interval Timer					
		clk	Clock Input	Double-click to export	ClockSource			
		reset	Reset Input	Double-click to export	[clk]			
		s1	Avalon Memory Mapped Slave	Double-click to export	[clk]			
<input checked="" type="checkbox"/>		I2CMaster	I2C Master (opencores.org)					
		clock	Clock Input	Double-click to export	ClockSource			
		clock_reset	Reset Input	Double-click to export	[clock]			
		export	Conduit	i2c_opencores_0_export	[clock]			
		avalon_slave_0	Avalon Memory Mapped Slave	Double-click to export	[clock]			

FIGURE 2.1 – Partie Hardware

On a créé nos module dans l'ordre suivant :

### 1. Clock Source

L'horloge source est paramétré comme ci dessous :

**Parameters**

Clock frequency:  Hz

☒ Clock frequency is known

Reset synchronous edges:

FIGURE 2.2 – Paramètres Clock Source

De plus, on a exporté les signaux clk et reset pour que ces signaux soient paramétrables au niveau software.

## 2. Nios II Processor

On a sélectionné le Nios II comme ci-dessous :



**Select a Nios II Core**

Nios II Core:

☒ Nios II/e

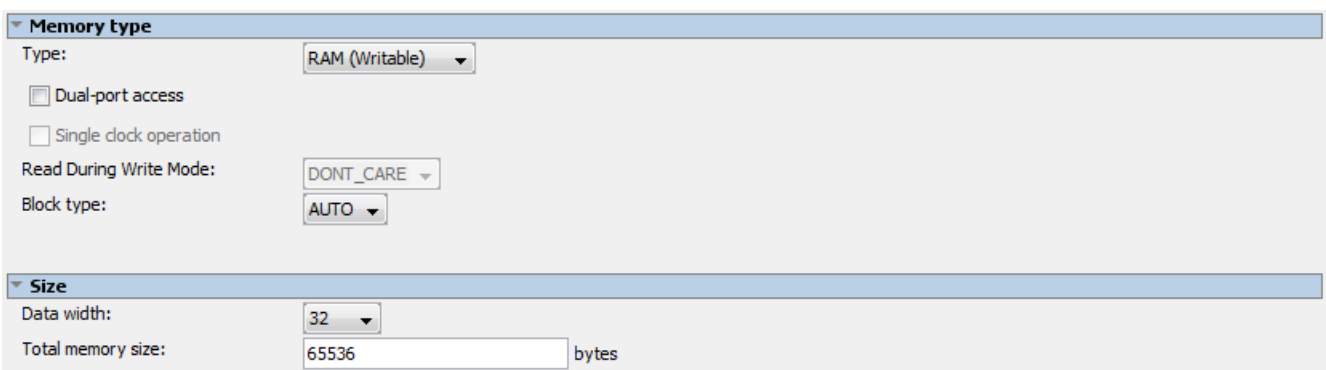
☐ Nios II/s

☐ Nios II/f

FIGURE 2.3 – Sélection Nios II

## 3. On-Chip Memory (RAM)

On a sélectionné un module de 64 Ko de RAM comme paramétré ci-dessous :



**Memory type**

Type: RAM (Writable)

☐ Dual-port access

☐ Single clock operation

Read During Write Mode: DONT\_CARE

Block type: AUTO

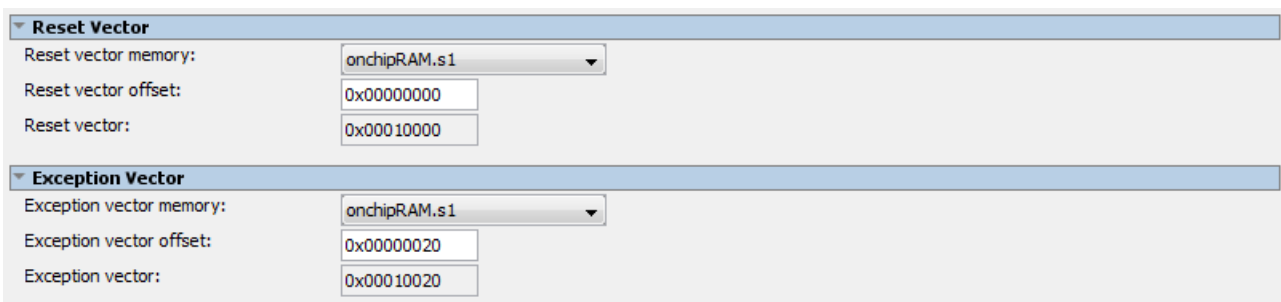
**Size**

Data width: 32

Total memory size: 65536 bytes

FIGURE 2.4 – Paramètres RAM

De plus, il faut revenir dans les paramètres du Nios II pour paramétrer les vecteurs d'interruption :



**Reset Vector**

Reset vector memory: onchipRAM.s1

Reset vector offset: 0x00000000

Reset vector: 0x00010000

**Exception Vector**

Exception vector memory: onchipRAM.s1

Exception vector offset: 0x00000020

Exception vector: 0x00010020

FIGURE 2.5 – Paramètres des vecteurs d'interruption

## 4. JTAG UART

Création d'un JTAG UART pour gérer l'implémentation du code software et le debug.

## 5. Interval Timer

Création de l'IP core d'un timer permettant la gestion du temps du protocole I2C :

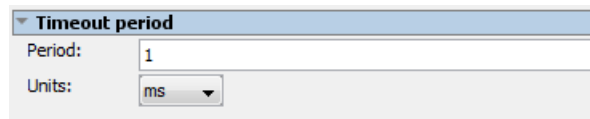


FIGURE 2.6 – Paramètres Timer

## 6. I2C Master

L'IP core d'un maître I2C n'étant pas présent de base dans QSYS, on a pris un opencore créé et mis à jour par Richard Herveille. <http://opencores.org/project,i2c>

Il faut faire une *Analysis & Synthesis* des fichiers Verilog dans un nouveau projet Quartus pour en vérifier l'intégrité. Ensuite, on a créé un nouveau composant I2C Master en ajoutant les fichiers Verilog de l'opencore téléchargé précédemment.

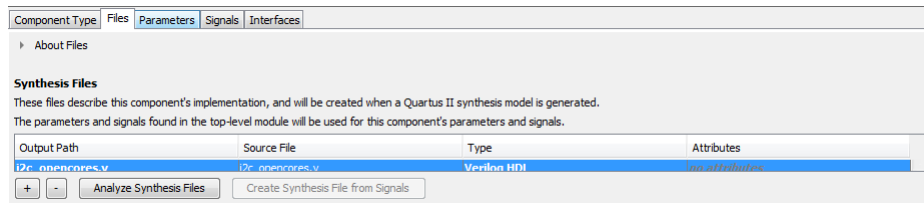


FIGURE 2.7 – Ajouts des fichiers de synthèse

A la suite de l'ajout du module, on exporte la connexion *export*.

Il faut après vérification du tout, faire une génération pour créer les fichiers HDL permettant la compilation de l'ensemble des modules dans le *Top Level*.

Enfin, il faut créer un autre projet Quartus dans un nouveau répertoire pour compiler l'ensemble des modules précédemment générés sur QSYS pour en faire le *Top Level* d'intégration. Il suffit ensuite de programmer la carte.



# Mise en place du projet Eclipse NBT

---

### 3.1 Validation du fonctionnement hardware

Après avoir réussi l'implémentation du programme dans la carte. Il faut lancer le plugin **Nios II Software Build Tools for Eclipse** et créer un nouveau projet *HelloWorld.c* qui fera apparaître un projet BSP et le projet contenant le *HelloWorld.c*. Une fois les deux projets créés, il suffit de générer le projet BSP via son éditeur, puis le compiler. Ensuite compiler le projet où se trouve le fichier *HelloWorld.c*. Il devrait apparaître dans la console *Hello from Nios II!*.

### 3.2 Mise en place des bibliothèques contenant les fonctions I2C

Pour utiliser les fonctions présentes dans le paquetage de l'opencore maître I2C, il suffit d'inclure dans le projet les bibliothèques *system.h* et *i2c\_opencores.h*. Voici l'arborescence obtenue.

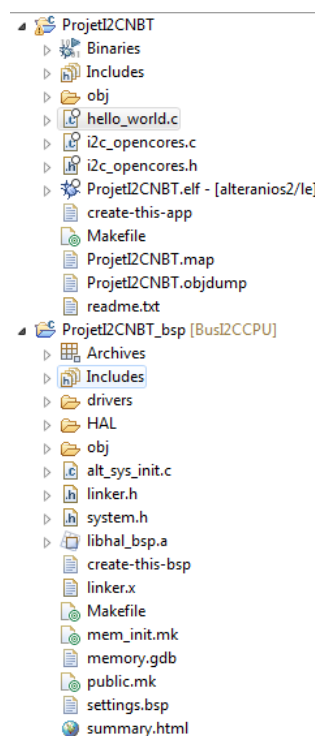


FIGURE 3.1 – Arborescence des projets

Un programme de test d'une mémoire I2C externe est présente aussi dans le paquetage, nommée *I2C\_tests.c* qu'il est possible d'utiliser pour effectuer ses propres tests.





# CoDesign

---

Spécialité Informatique Industrielle  
4<sup>ème</sup> année  
2014-2015

Compte rendu TP3

**Résumé:**

---

Mots clefs: Apprenti :  
**Thomas GACHE, Thibault ARTUS**  
[thomas.gache@yahoo.fr](mailto:thomas.gache@yahoo.fr), [theskullmachine@gmail.com](mailto:theskullmachine@gmail.com)

Tuteur :  
**Alexis ROLLAND**  
[alexis.rolland@univ-tours.fr](mailto:alexis.rolland@univ-tours.fr)  
Polytech'Tours