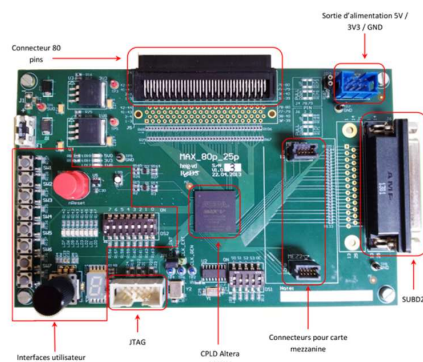


# Additionneurs en VHDL

## CONCEPTION DE SYSTÈMES NUMÉRIQUES (CSN)



Auteur : Spinelli Isaia  
Prof : Etienne Messerli  
Ing : Sébastien Masle  
Date : 22.10.2019  
Salle : A09 – HEIG-VD  
Classe : CSN

## Table des matières

Introduction.....	- 2 -
Encodeur de priorité à 4 entrées.....	Erreur ! Signet non défini.
Spécifications.....	Erreur ! Signet non défini.
Description textuelle .....	Erreur ! Signet non défini.
Description en VHDL.....	Erreur ! Signet non défini.
Simulation.....	Erreur ! Signet non défini.
Synthèse .....	Erreur ! Signet non défini.
Encodeur de priorité à 16 entrées .....	Erreur ! Signet non défini.
Analyse .....	Erreur ! Signet non défini.
Description en VHDL.....	Erreur ! Signet non défini.
Simulation automatique.....	Erreur ! Signet non défini.
Synthèse .....	Erreur ! Signet non défini.
Intégration.....	Erreur ! Signet non défini.
Test de l'intégration .....	Erreur ! Signet non défini.
Conclusion .....	Erreur ! Signet non défini.
Difficultés rencontrées .....	Erreur ! Signet non défini.
Compétences acquises .....	Erreur ! Signet non défini.
Résultats obtenus.....	Erreur ! Signet non défini.

## Introduction

Description de différentes versions de l'addition en VHDL par étapes, soit :

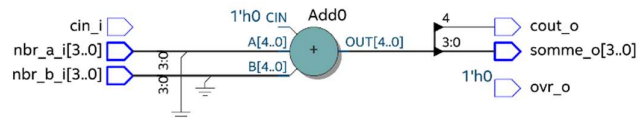
- Sans dépassement
- Avec carry
- Avec carry et overflow

## Additionneur 4 bits avec carry (in et out)

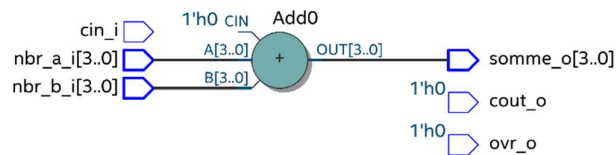
Le code est en annexe.

Suivis des vues RTL :

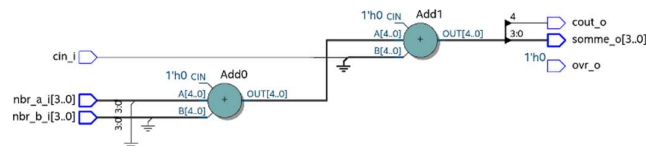
Vue RTL étape a : additionneur de deux vecteurs 4 bits sans report



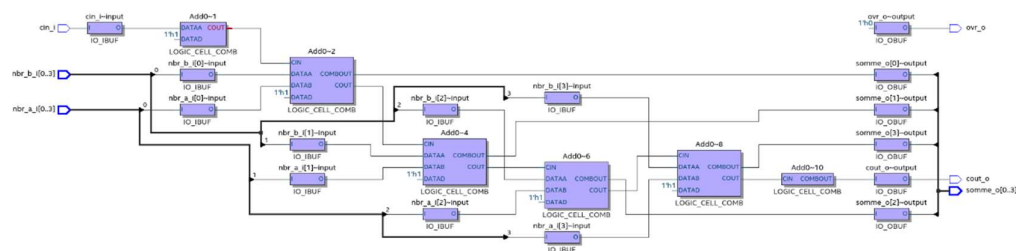
Vue RTL étape b : ajout du carry out



Vue RTL étape c : ajout du carry in



Vue Technologique de l'étape c :



Remarque : On peut voir dans la vue RTL pour l'ajout du carry in, qu'il a besoin d'ajouter un additionneur pour utiliser le CIN mais dans la vue Technologique on constate qu'il utilise seulement 4 additionneurs 1 bits.

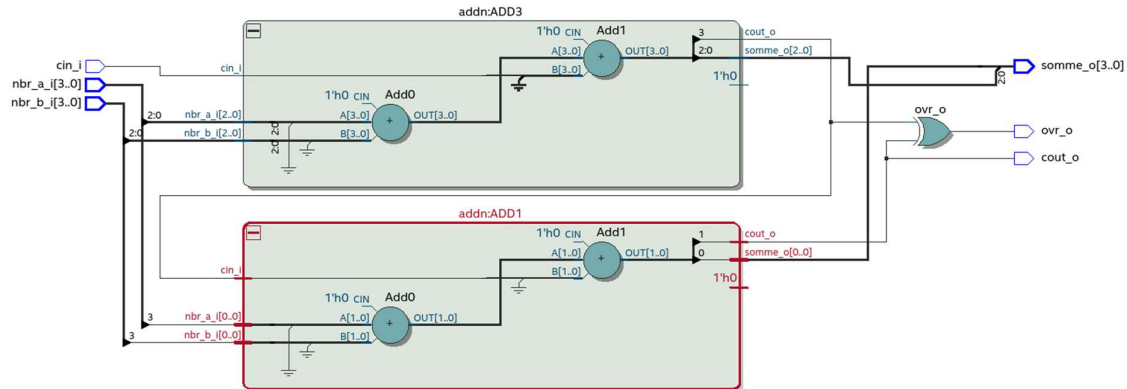
## Additionneur générique N bits avec carry (in et out)

Le code est en annexe.

## Additionneur 4 bits avec carry (in et out) et overflow

Le code est en annexe.

Voici la vue Technologique qui donne une bonne vue d'ensemble de la logique de l'additionneur avec overflow :



Voici le log de la simulation automatique de l'add4\_full :

```

Transcript
# -- Compiling architecture test_bench_full of add4_tb
# -- Loading entity add4
# ** Warning: ../src_tb/add4_full_tb.vhd(59): (vcom-1236) Shared variables must be of a protected type.
# End time: 09:55:07 on Oct 22,2019, Elapsed time: 0:00:00
# Errors: 0, Warnings: 1
# End time: 09:55:07 on Oct 22,2019, Elapsed time: 0:00:13
# Errors: 0, Warnings: 4
# vsim -novopt work.add4_tb
# Start time: 09:55:07 on Oct 22,2019
# ** Warning: (vsim-8891) All optimizations are turned off because the -novopt switch is in effect. This will cause
# are using this switch to preserve visibility for Debug or PLI features please see the User's Manual section on Prese
# Loading std.standard
# Refreshing /cours REDS/CSN/SpinelliIsaia/Labo_3_additionneurs/add4/comp/work.add4_tb(test_bench_full)
# Loading std.textio(body)
# Loading ieee.std_logic_1164(body)
# Loading ieee.numeric_std(body)
# Loading work.add4_tb(test_bench_full)
# Refreshing /cours REDS/CSN/SpinelliIsaia/Labo_3_additionneurs/add4/comp/work.add4(flot_don)
# Loading work.add4(flot_don)
# Refreshing /cours REDS/CSN/SpinelliIsaia/Labo_3_additionneurs/add4/comp/work.addn(flot_don)
# Loading work.addn(flot_don)
VSIM(pause)> run -all
# ** Note: Début de la simulation
# Time: 0 ns Iteration: 0 Instance: /add4_tb
# ** Note: Nombre d'erreurs détectées = 0
# Time: 51200 ns Iteration: 0 Instance: /add4_tb
# ** Note: Fin de la simulation
# Time: 51200 ns Iteration: 0 Instance: /add4_tb

```

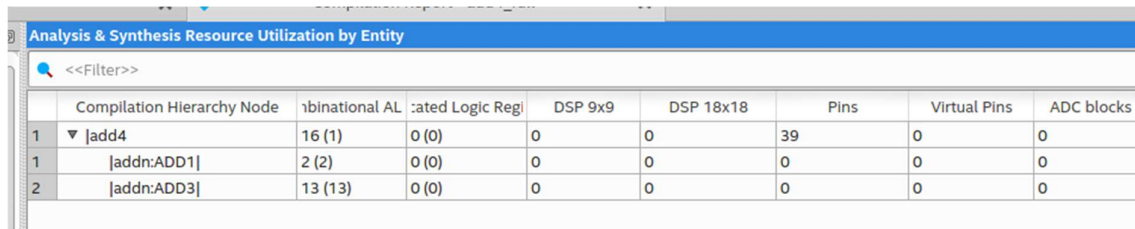
Voici la logique avec l'additionneur complet 4 bits :

Analysis & Synthesis Resource Utilization by Entity								
<<Filter>>								
	Compilation Hierarchy Node	Combinational AL	latched Logic Regl	DSP 9x9	DSP 18x18	Pins	Virtual Pins	ADC block
1	▼ [add4]	8 (1)	0 (0)	0	0	15	0	0
1	[addn:ADD1]	2 (2)	0 (0)	0	0	0	0	0
2	[addn:ADD3]	5 (5)	0 (0)	0	0	0	0	0

## Additionneur générique N bits (in et out) et overflow

Le code est en annexe.

Voici la quantité logique de l'additionneur générique avec overflow pour (12 bits)



	Compilation Hierarchy Node	Combinational AL	Static Logic Regl	DSP 9x9	DSP 18x18	Pins	Virtual Pins	ADC blocks
1	▼ [add4]	16 (1)	0 (0)	0	0	39	0	0
1	[addn:ADD1]	2 (2)	0 (0)	0	0	0	0	0
2	[addn:ADD3]	13 (13)	0 (0)	0	0	0	0	0

Remarque : On peut voir que l'additionneur ADD1 a bien évidemment la même quantité de logique et que le ADD3 augmente de façon linéaire N+1 quantité logique N étant le nombre de bit.

Date : 22.10.19

Nom de l'étudiant : Spinelli Isaia