

Systèmes embarqués automobile et aéronautique

**Master d'Université Spécialisé (MUS) en
Ingénierie des Systèmes Embarqués (ISE)**

Chapitre 6: Les SoC (System on Chip) pour les applications embarquées avancées

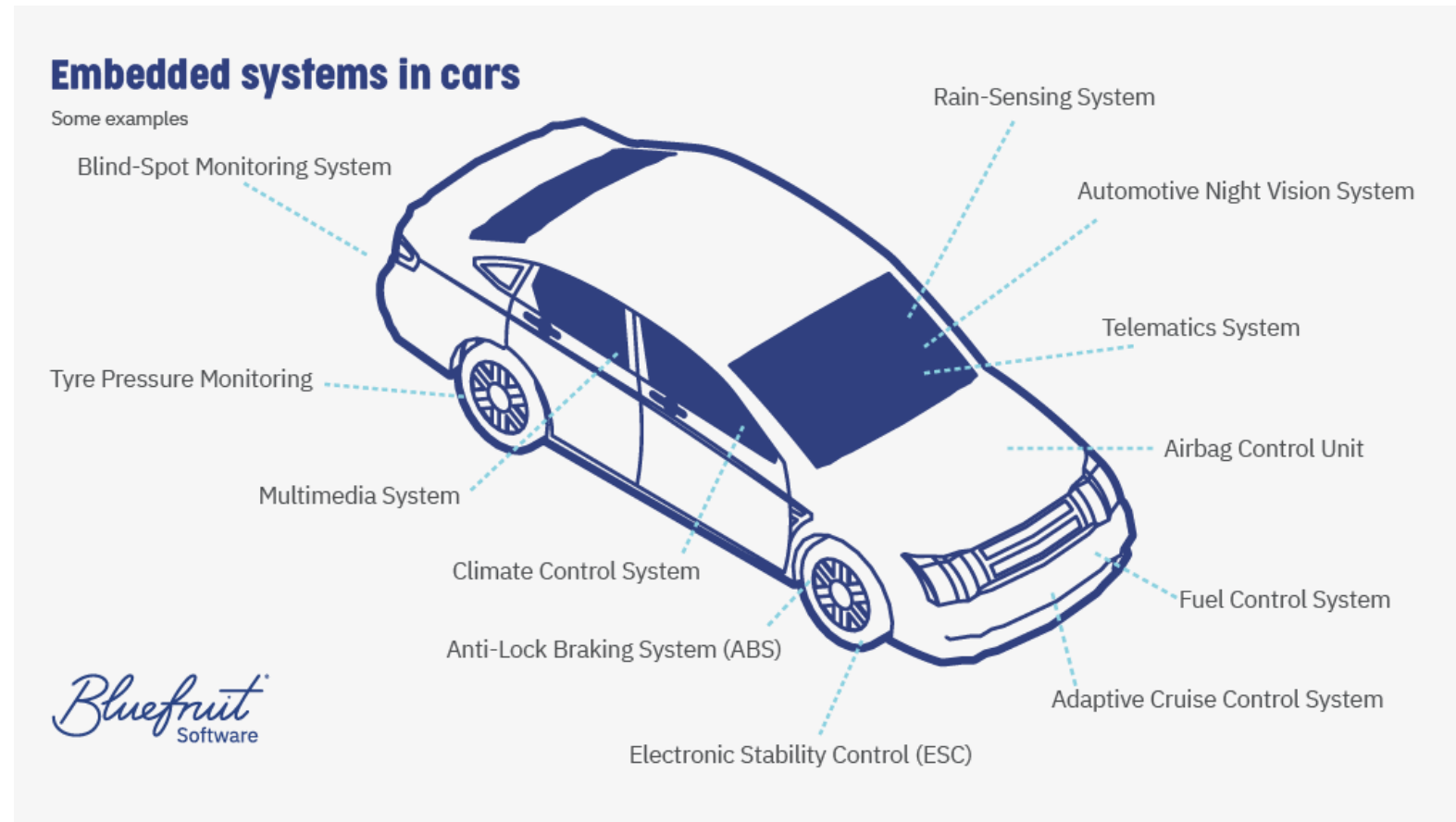
Application: Implémentation d'un SoC Dual Core

Année universitaire 2024/2025

Dr. Ing. LASSIOUI Abdellah

Introduction:

Les véhicules modernes intègrent plusieurs dizaines, voire centaines de capteurs pour surveiller et contrôler divers systèmes tels que le moteur, la sécurité, l'aide à la conduite et le confort. Ces capteurs collectent en temps réel des données sur la vitesse, la température, la pression, les obstacles environnants et bien plus encore. Leur intégration croissante reflète l'évolution vers des voitures plus intelligentes et autonomes

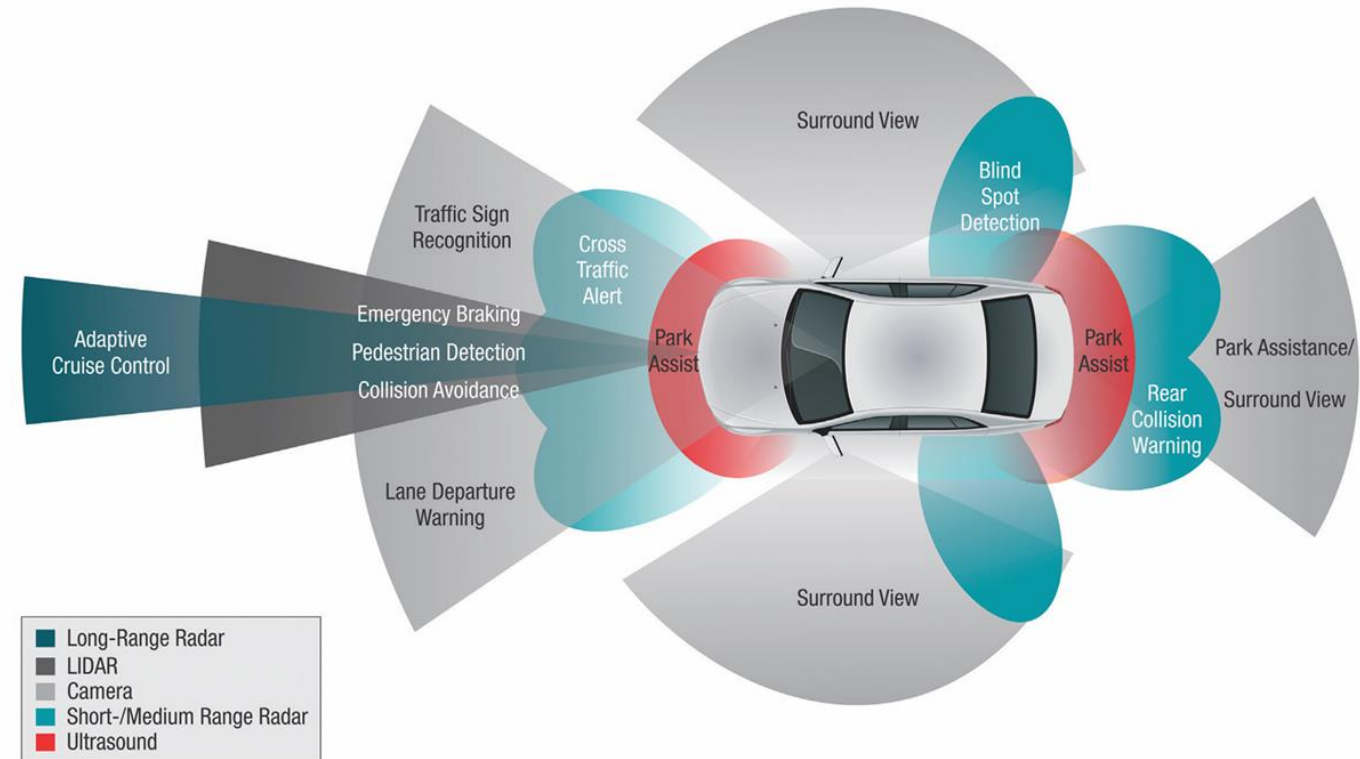


<https://bluefruit.co.uk/processes/what-embedded-systems-in-car/>

Complexité des véhicules modernes:

Exemple de la conduite autonome

La conduite autonome repose sur une forte intégration de contrôleurs embarqués, traitant en temps réel des données issues de capteurs, caméras et lidars. Ces contrôleurs exécutent des algorithmes avancés pour la perception, la planification de trajectoire et la prise de décision. Cette centralisation et l'optimisation du traitement garantissent une réponse rapide et fiable, essentielle pour la sécurité et l'efficacité des véhicules autonomes.



Source :

*Advanced Driver Assistance (ADAS) Solutions Guide
(Texas Instrument)*

Limitations des contrôleurs classiques et migration vers les SoC

Les **microcontrôleurs traditionnels**, limités en puissance de calcul et en traitement parallèle, peinent à gérer ces tâches en temps réel.

Pour répondre à ces exigences croissantes, les SoC (**System-on-Chip**) offrent des architectures **multi-cœurs** et des **accélérateurs dédiés**, optimisant ainsi les performances des systèmes embarqués.

Définition d'un SoC

Un **SoC (System on Chip)** est un circuit intégré qui regroupe **plusieurs composants** d'un système informatique complet **sur une seule puce**.

Ces composants peuvent inclure :

- **Un ou plusieurs processeur (CPU)**
- **Mémoire (RAM, ROM)**
- **Unités de traitement spécialisées (GPU, DSP pour le traitement du signal, etc.)**
- **Interfaces de communication (UART, SPI, I2C, Ethernet, USB, etc.)**
- **Blocs analogiques (convertisseurs ADC/DAC, régulateurs de tension, etc.)**
- **Contrôleurs de périphériques (écran, capteurs, stockage, etc.)**

Caractéristiques d'un SoC

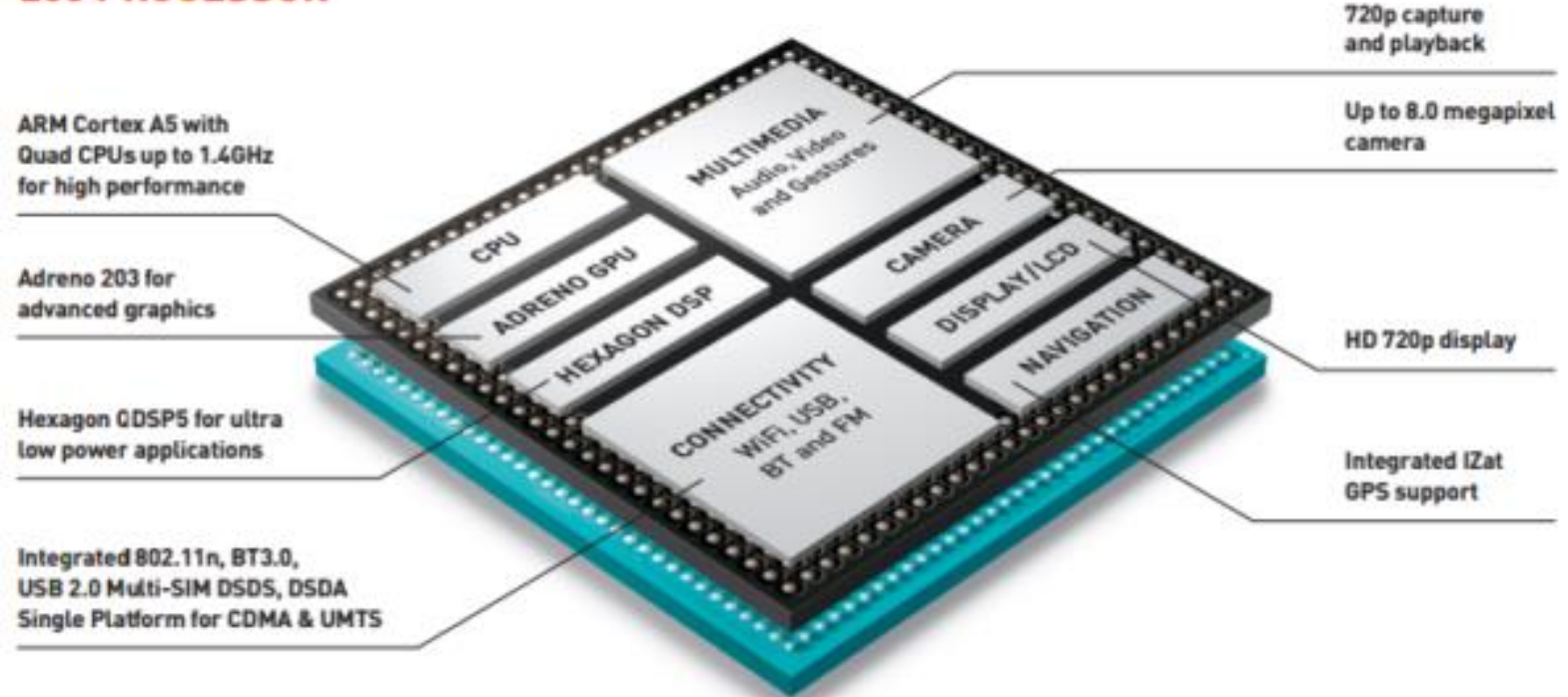
- ❑ **Intégration élevée** : Tout ou presque est intégré sur une seule puce, ce qui réduit la taille globale du système et les coûts de production.
- ❑ **Faible consommation d'énergie** : Optimisé pour des applications embarquées, le SoC est souvent conçu pour consommer très peu d'énergie.
- ❑ **Haute performance** : Les composants intégrés communiquent directement sur la même puce, réduisant les temps de latence.

Exemple d'un SoC: Qualcomm Snapdragon

Un SoC comme le **Snapdragon** de Qualcomm dans les smartphones intègre :

- ❑ Un **processeur principal** (CPU) pour les calculs.
- ❑ Un **GPU** pour le rendu graphique.
- ❑ Un **DSP**
- ❑ Un contrôleur de connectivité (WiFi, BT, USB,...)
- ❑ Des **contrôleurs** pour les périphériques comme caméras, écrans, etc.

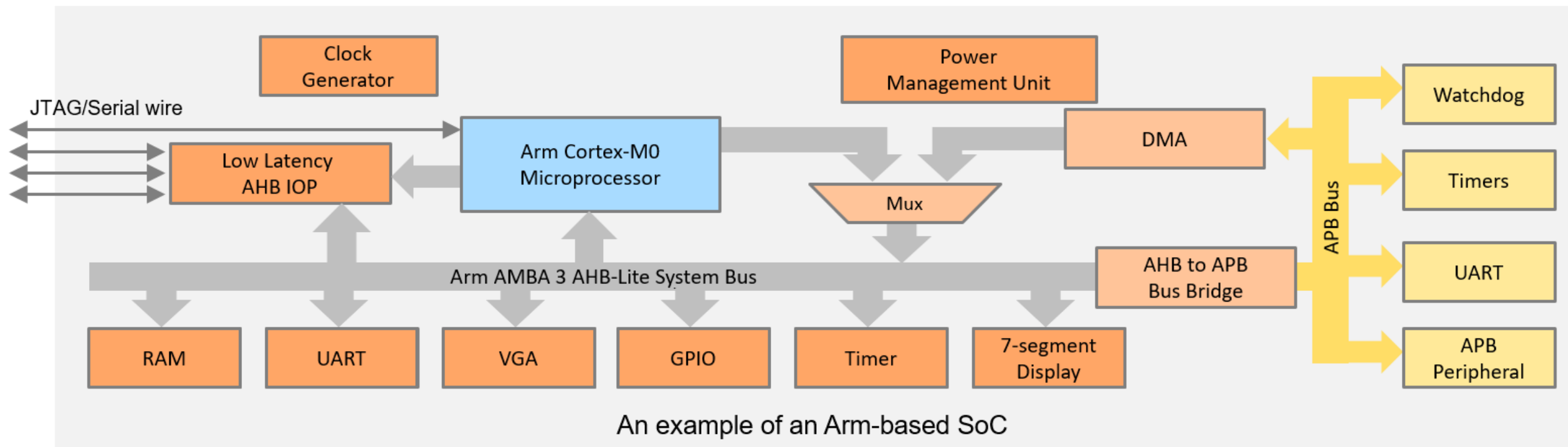
200 PROCESSOR



Source: <https://techawarness.com/what-is/what-is-qualcomms-snapdragon-soc/>

Exemple d'un SoC: Le processeur ARM (Advanced RISC Machines)

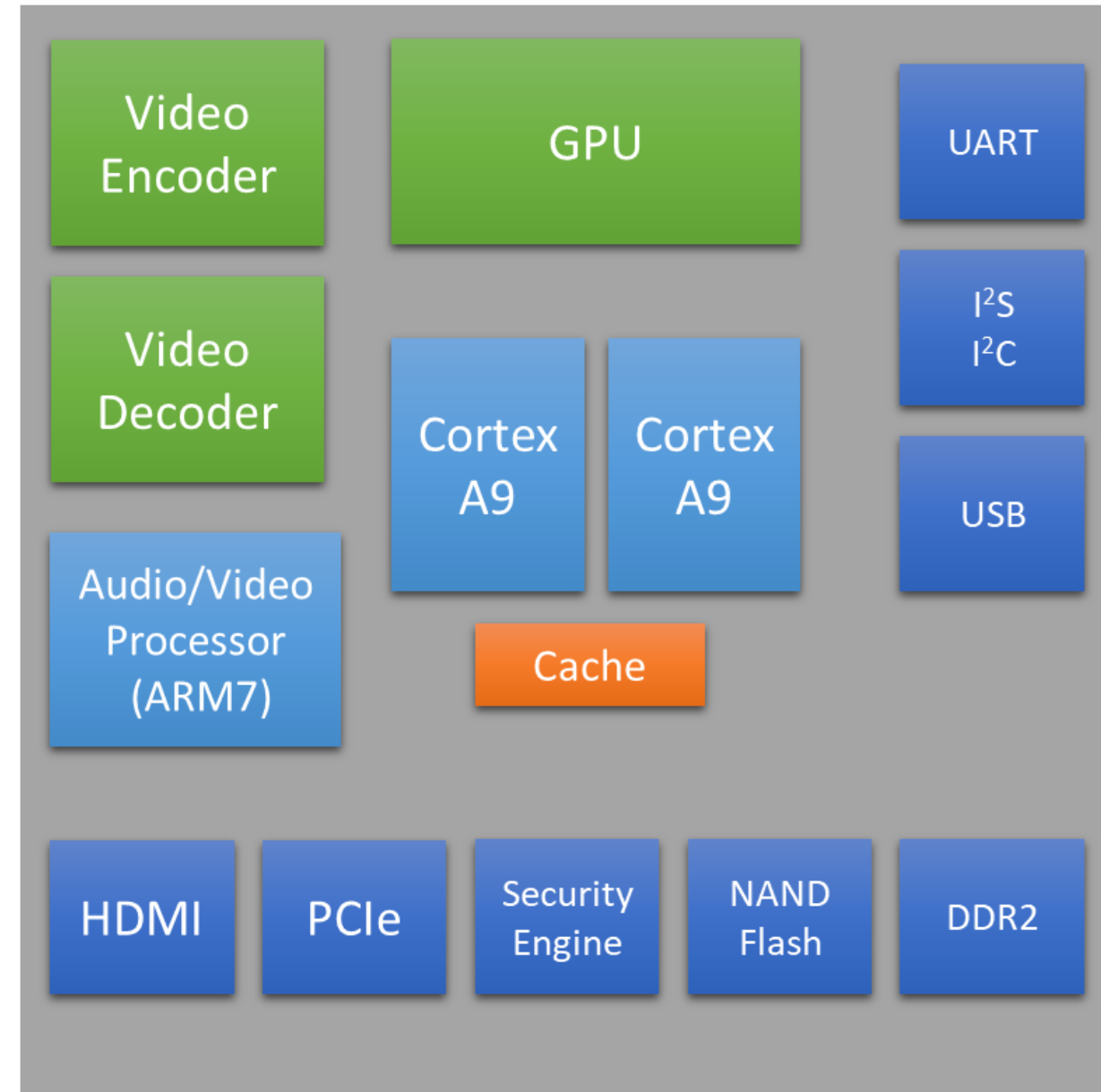
Un SoC Arm de base se compose généralement d'un processeur Arm, tel que le **Cortex-M0**, d'un bus AMBA (Advanced Microcontroller Bus Architecture), d'un ou plusieurs périphériques (UART, VGA, GPIO, TIMERS...), d'un générateur d'horloge, d'une unité de gestion d'alimentation, AHB (Advanced High Performance Bus), APB (Advanced Peripheral Bus)...



Exemple d'un SoC: La carte NVIDIA Tegra 2

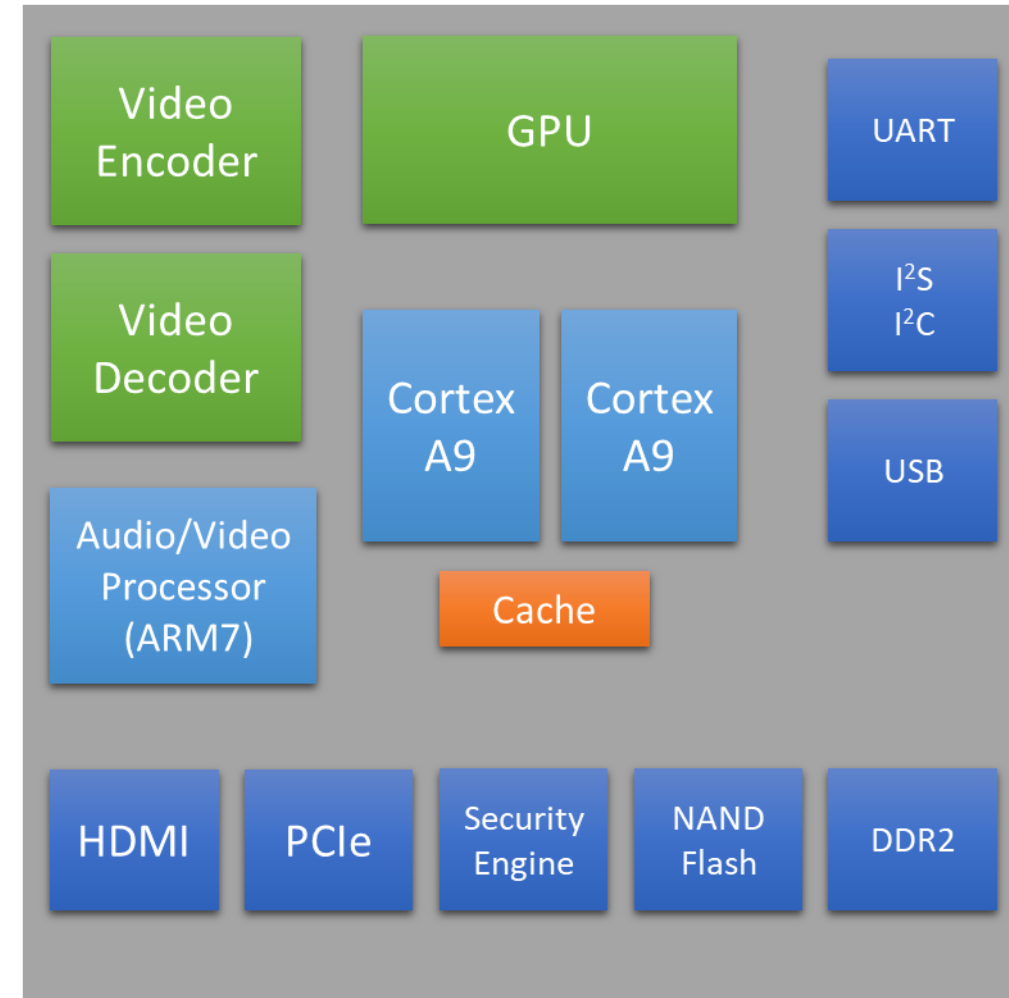
La fonction principale du Tegra 2 est de fournir un système intégré complet permettant :

- ❑ **Calcul efficace** (double cœur ARM Cortex-A9).
- ❑ **Traitement graphique avancé** (GPU ULP GeForce).
- ❑ **Support multimédia** (lecture/encodage vidéo HD 1080p).
- ❑ **Optimisation de la consommation d'énergie** pour une utilisation dans les systèmes embarqués.



Exemple d'un SoC: La carte NVIDIA Tegra 2

Designer	NVIDIA
Year	2010
Processor	Arm Cortex-A9 (dual-core)
Frequency	Up to 1.2 GHz
Memory	1 GB 667 MHz LP-DDR2
Graphics	ULP GeForce
Process	40 nm
Package	12 × 12 mm (package on package)



Le NIOS II Embedded Processor

Le **Nios II** est un processeur **softcore** développé par Intel (anciennement Altera), conçu pour être intégré dans des systèmes sur puce (SoC) basés sur FPGA. Il offre une grande flexibilité en permettant aux concepteurs d'adapter l'architecture selon les besoins spécifiques de l'application.



Le NIOS II Embedded Processor

Caractéristiques principales du Nios II dans les SoC

1.Architecture configurable : Contrairement aux **microcontrôleurs classiques**, le Nios II peut être **personnalisé en termes de mémoire, d'unités de calcul et de périphériques**.

2.Intégration avec FPGA : Fonctionne sur les FPGA Intel (ex-Altera), permettant une adaptation rapide aux exigences des systèmes embarqués.

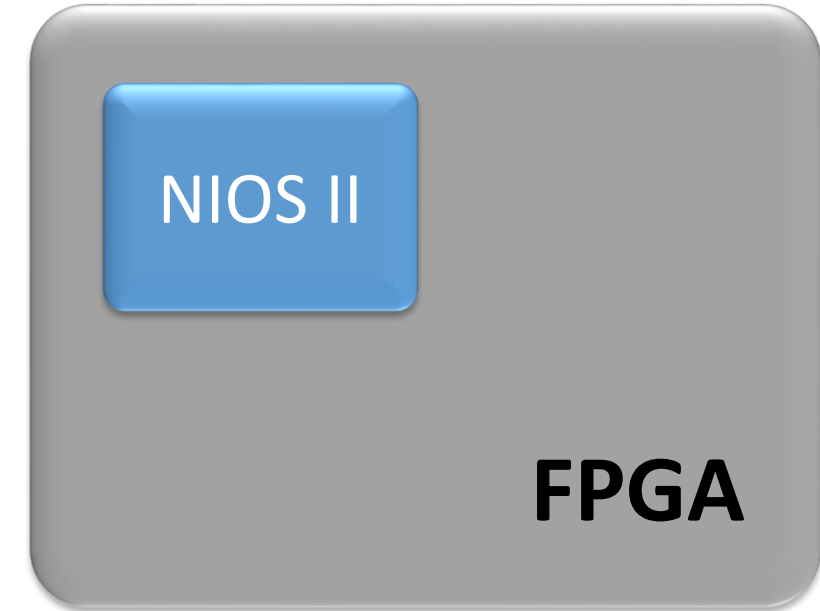
3.Optimisation des performances : Disponible en plusieurs variantes (économique, standard, performant) selon le besoin en puissance de calcul.

4.Applications : **Utilisé dans l'automobile**, les télécommunications, le contrôle industriel et les systèmes embarqués avancés.



Le NIOS II Embedded Processor

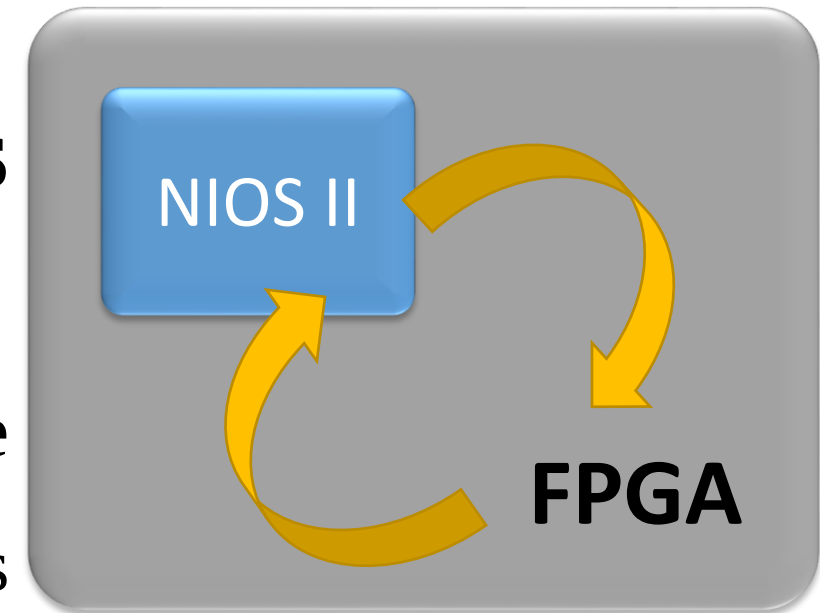
- ❑ **Famille: Un processeur de 32Bits**
- ❑ **Performances : 300 MIPS**
- ❑ **Pourcentage des CLB occupé : 25% (Cyclone III FPGA)**
- ❑ **Contrairement au processeurs fixes (ARM), NIOS est configurable (possibilité d'ajouter des périphériques en fonction du besoin)**



Le NIOS II Embedded Processor

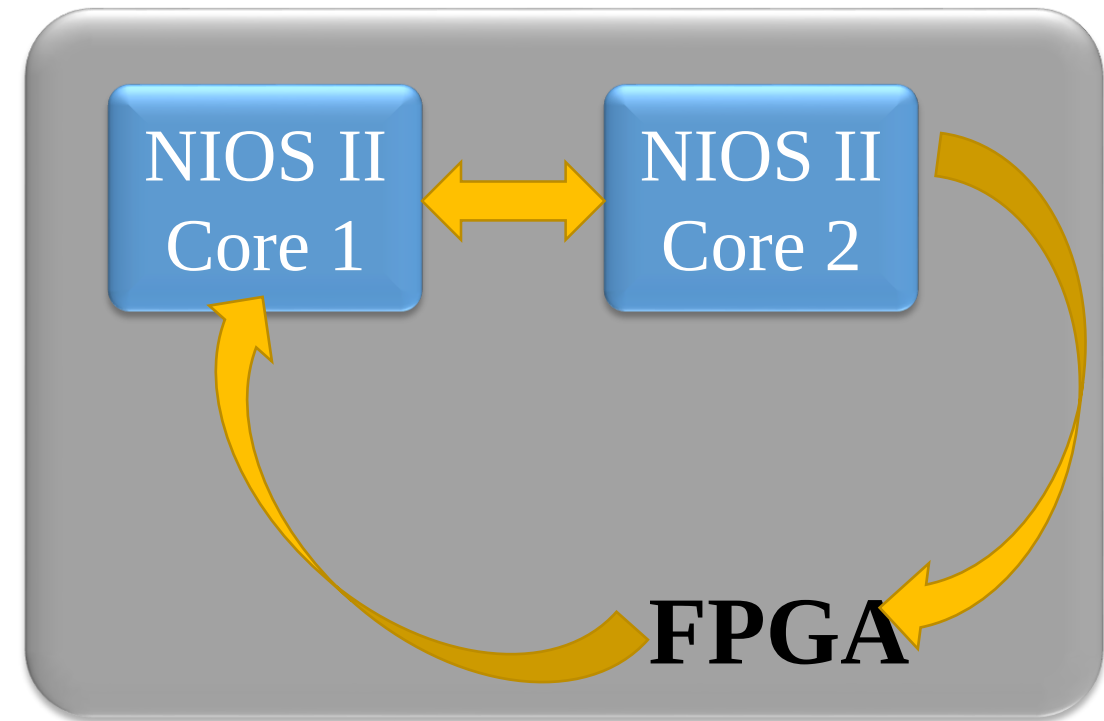
❑ Coexistence avec les circuits FPGA:

- ❑ Combinaison d'un traitement logiciel (via NIOS II) et d'un traitement matériel (via les CLBs)
- ❑ Le processeur NIOS II peut travailler en parallèle avec des blocs matériels personnalisés pour des tâches critiques en temps réel ou des traitements de signaux rapides.



Le NIOS II Embedded Processor

- ❑ **Plusieurs Nios II** peuvent être ajoutés dans un même FPGA, en fonction de la taille et des ressources disponibles sur le circuit FPGA (l'espace logique, la mémoire, les entrées/sorties, etc.).
- ❑ **Parallélisme et multiprocessing:** Utiliser plusieurs processeurs Nios permet de créer des systèmes parallèles, où chaque processeur peut gérer des tâches distinctes



Le NIOS II Embedded Processor

- ❑ Il existe trois variantes du processeur NIOS II
- ❑ NIOS II/f Core (**Pipeline 6, Hrdware Multiplier 1 cycle, Logic Elements 1800**)
- ❑ NIOS II/s Core (**Pipeline 5, Hrdware Multiplier 3 cycle, Logic Elements 1200**)
- ❑ NIOS II/e Core (**Pipeline None, Hrdware Multiplier xx, Logic Elements 600**)

Le NIOS II Embedded Processor

❑ Notion du pipeline

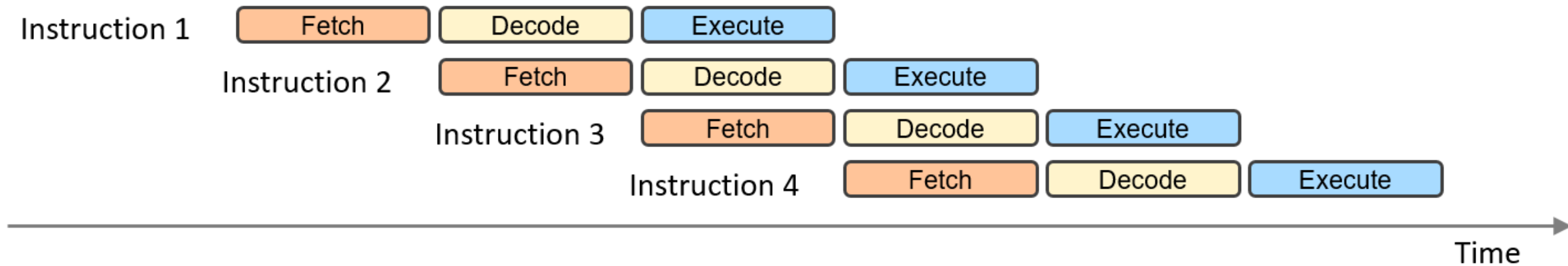
Le pipeline décrit l'enchaînement d'exécution d'instructions dans un processeur :

- 1.IF (Instruction Fetch)** : L'instruction est récupérée de la mémoire.
- 2.ID (Instruction Decode)** : L'instruction est décodée pour déterminer quelles opérations doivent être effectuées et quels registres ou mémoires doivent être accédés.
- 3.EX (Execute)** : L'opération est exécutée (par exemple, une addition ou un accès mémoire).
- 4.MEM (Memory Access)** : Si nécessaire, l'accès à la mémoire est effectué (lecture ou écriture).
- 5.WB (Write Back)** : Le résultat est écrit dans les registres ou dans la mémoire.

Le NIOS II Embedded Processor

❑ Notion du pipeline

Ce diagramme illustre les principes de fonctionnement d'un pipeline à trois étapes. L'essence de cette architecture est qu'elle permet de traiter jusqu'à trois instructions simultanément. Par exemple, lorsque la première instruction est en cours d'exécution, le processeur peut décoder la deuxième instruction et récupérer la troisième instruction en même temps, ce qui contribue à améliorer le débit du pipeline du processeur, ce qui se traduit par de meilleures performances.



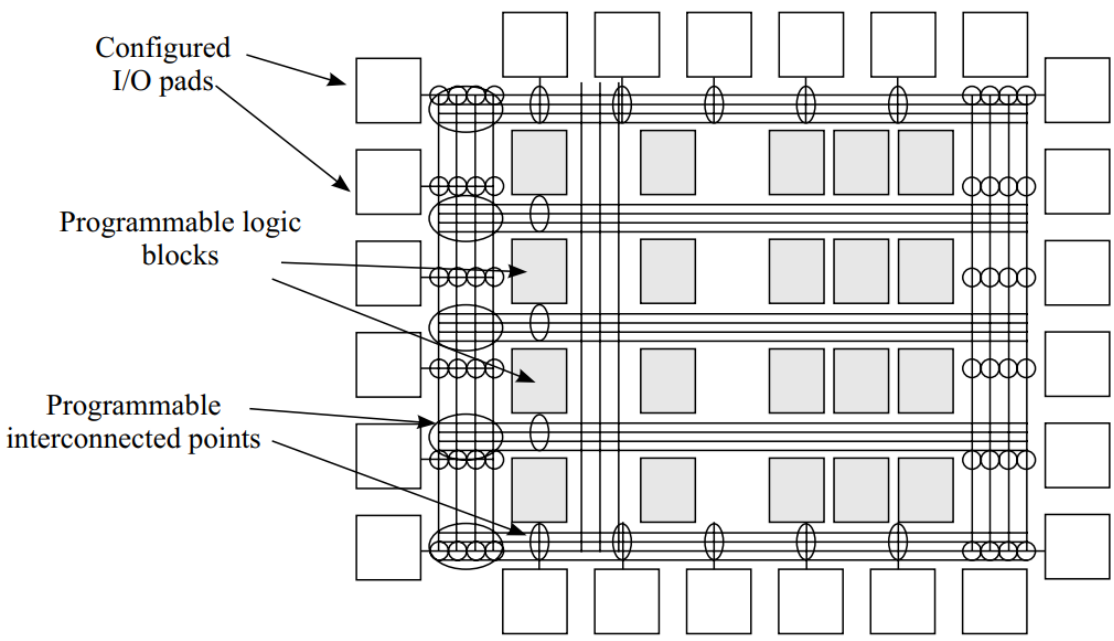
Le NIOS II Embedded Processor

❑ Notion du pipeline: exemple d'une pipeline à 6 cycles

Cycle 1	Cycle 2	Cycle 3	Cycle 4	Cycle 5	Cycle 6
IF (I1)	IF (I2)	IF (I3)	IF (I4)	IF (I5)	IF (I6)
	ID (I1)	ID (I2)	ID (I3)	ID (I4)	ID (I5)
		EX (I1)	EX (I2)	EX (I3)	EX (I4)
			MEM (I1)	MEM (I2)	MEM (I3)
				WB (I1)	WB (I2)
					RET (I1)

Le NIOS II Embedded Processor

- ❑ Sur un circuit FPGA on peut implémenter plusieurs Microprocesseur NIOS II
- ❑ NIOS II/f Core: **Logic Elements 1800**



Exemple: Cyclone V FPGA device family (source Intel)

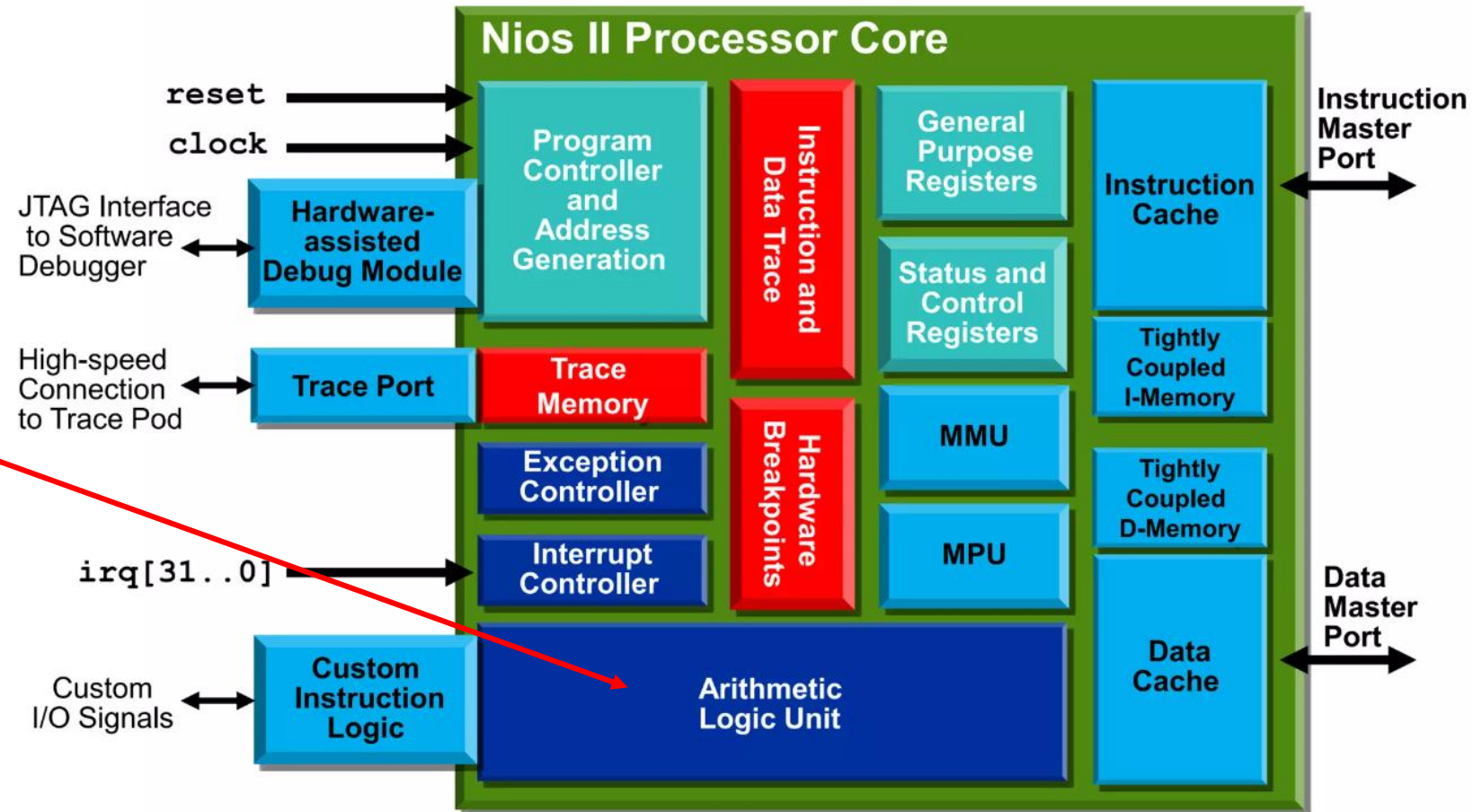
Density	Ordering Part Number (OPN)
25K LE	5CSEBA2U19I7LN
	5CSEBA2U23I7LN
	5CSXFC2C6U23I7LN
40K LE	5CSEBA4U19I7LN
	5CSEBA4U23I7LN
	5CSXFC4C6U23I7LN
85K LE	5CSEBA5U19I7LN
	5CSEBA5U23I7LN
	5CSXC5C6U23I7LN
110K LE	5CSEBA6U19I7LN
	5CSEBA6U23I7LN
	5CSXFC6C6U23I7LN

Le NIOS II Embedded Processor architecture

□ ALU: une unité

performant les opérations

arithmétiques et logiques



Source: Intel (Altera Nios II Architecture)

Le NIOS II Embedded Processor architecture

□ MPU, MMU (Memory

Protection Unit et Management

Unit): elles assurent la

protection et le management de

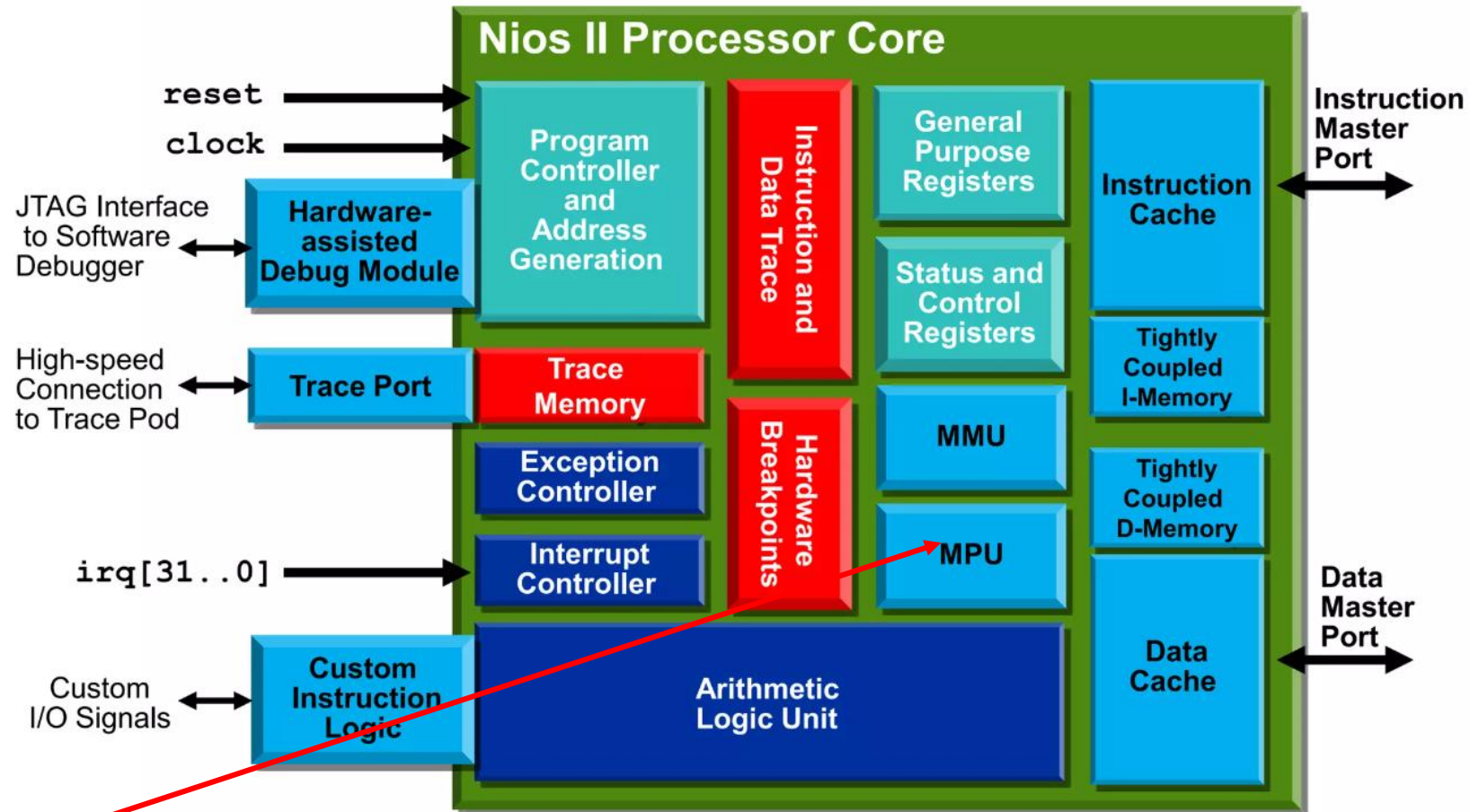
la mémoire. (Divise la mémoire

en plusieurs régions. Pour

chaque région on définit des

droits d'accès spécifiques

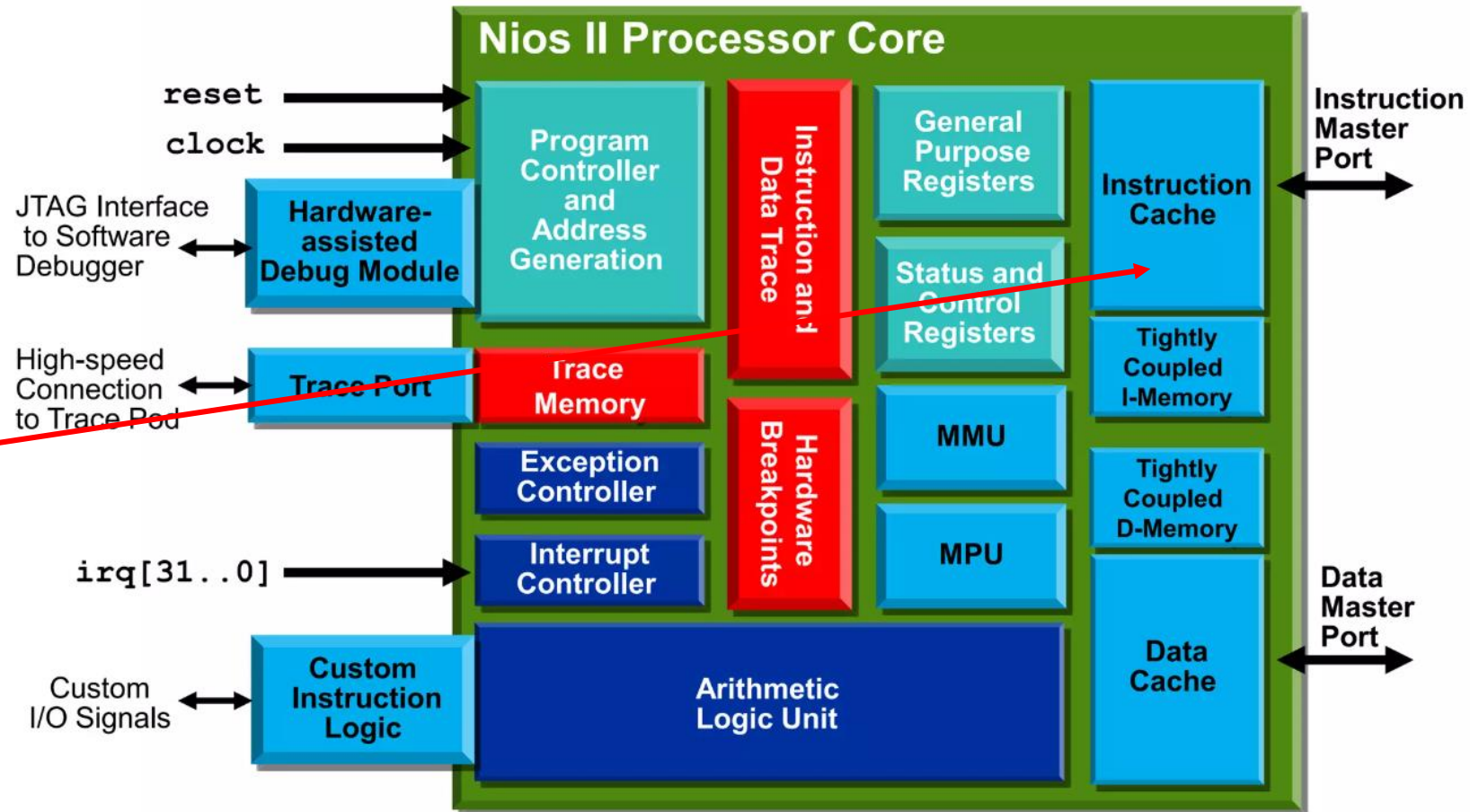
(lecture, écriture)).



Source: Intel (Altera Nios II Architecture)

Le NIOS II Embedded Processor architecture

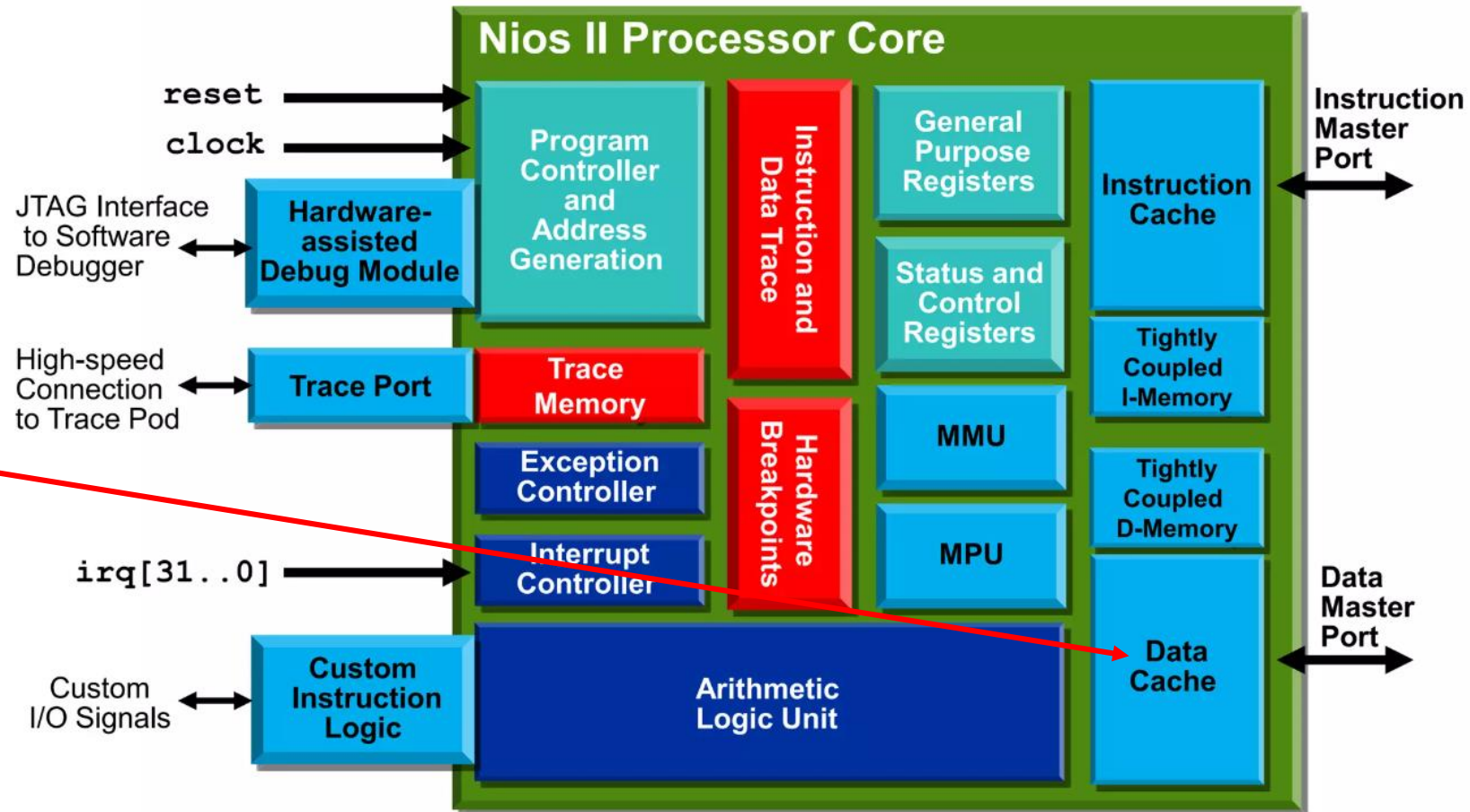
- ❑ **Instruction Cache:** il s'agit d'une mémoire rapide située entre le processeur et la mémoire principale
- ❑ Elle stocke temporairement les instructions qui sont couramment utilisées pour accélérer l'exécution et optimiser le pipeline



Source: Intel (Altera Nios II Architecture)

Le NIOS II Embedded Processor architecture

- ❑ **Data Cache:** il s'agit d'une mémoire rapide qui stocke temporairement les données lues ou écrites dans la mémoire principale
- ❑ **Objectif:** réduire le temps d'accès aux données

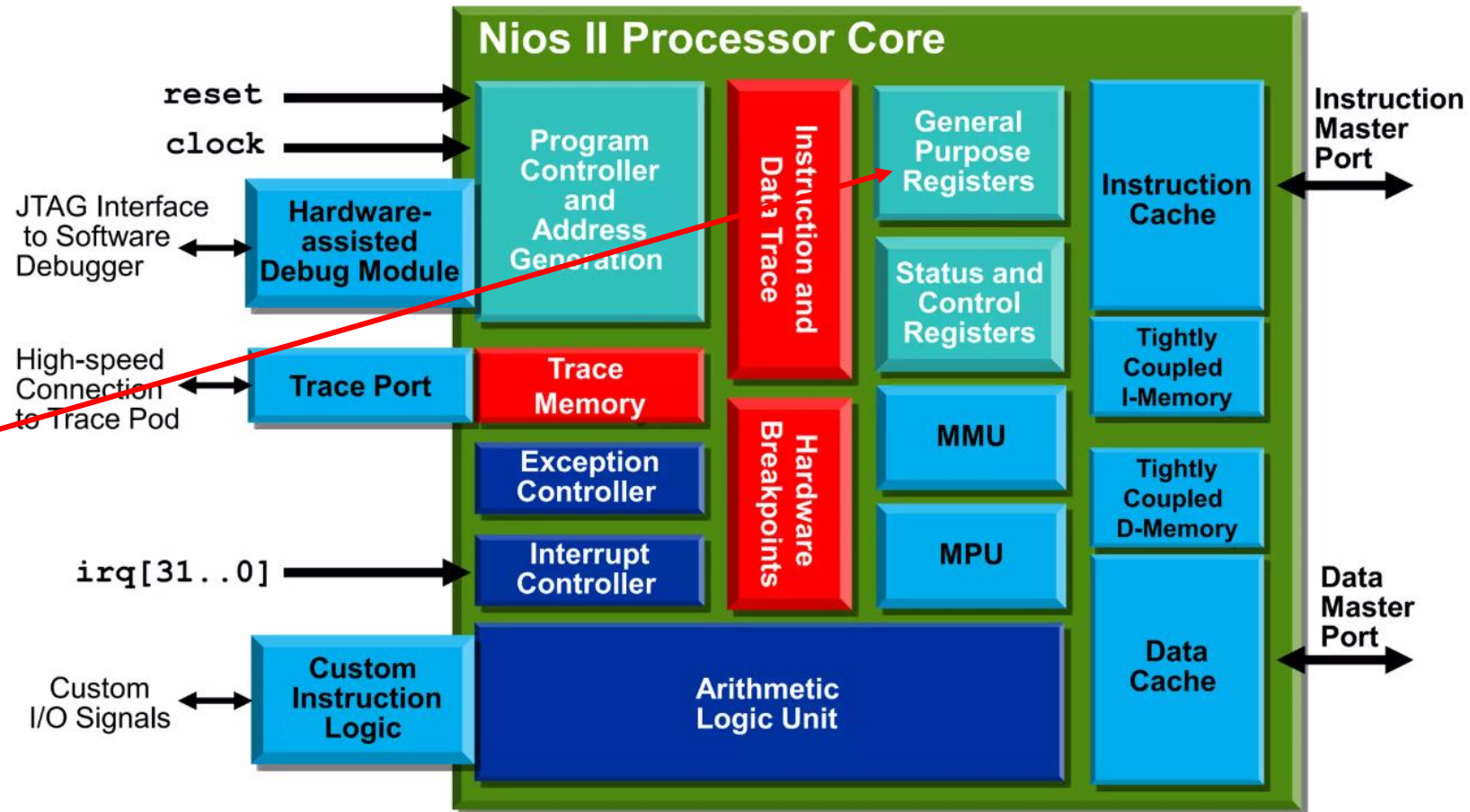


Source: Intel (Altera Nios II Architecture)

Le NIOS II Embedded Processor architecture

❑ Status and control registres:

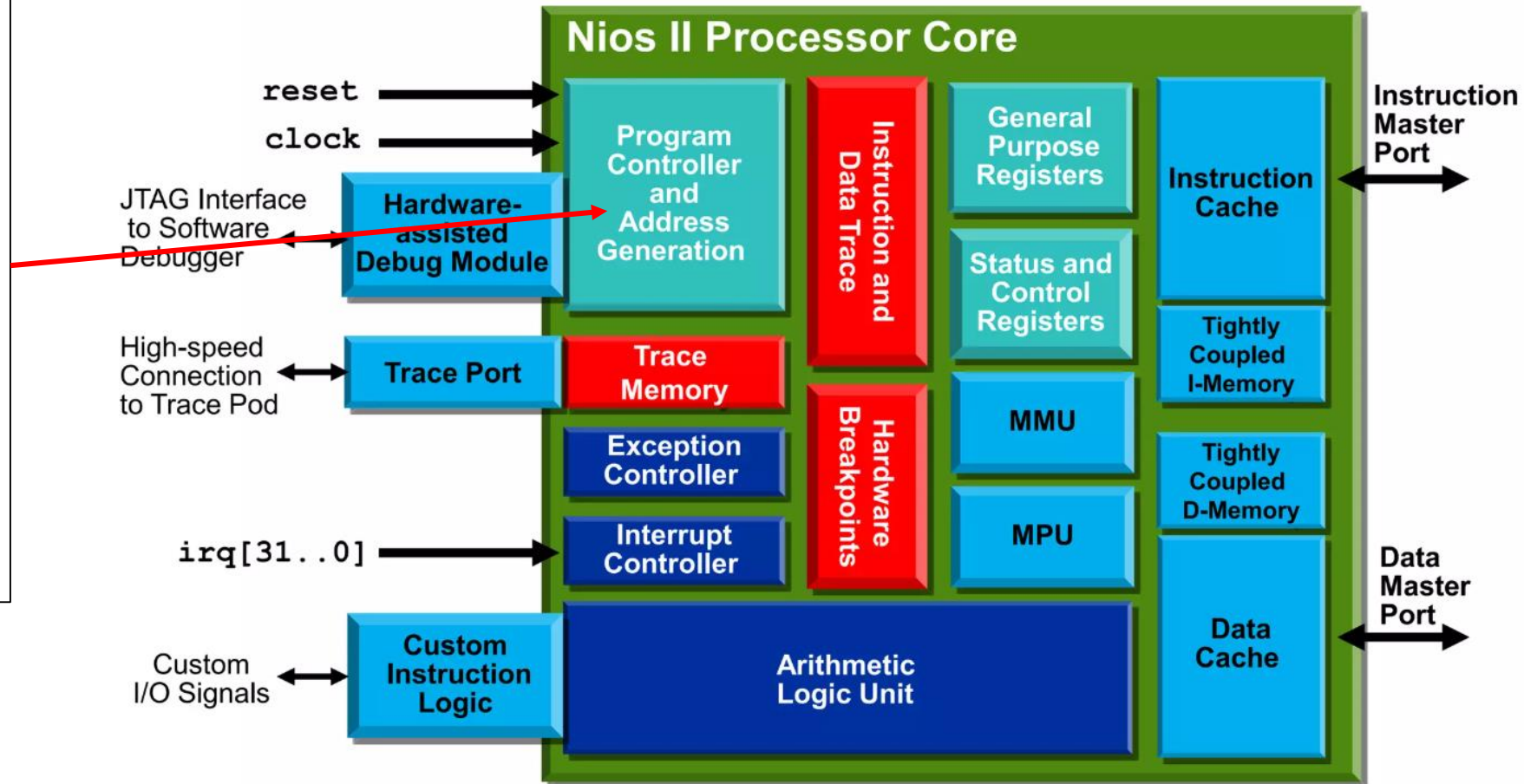
ces registres servent dans les opérations arithmétiques et logiques (32 registres généraux (R0 à R31) dans le NIOS 2)



Source: Intel (Altera Nios II Architecture)

Le NIOS II Embedded Processor architecture

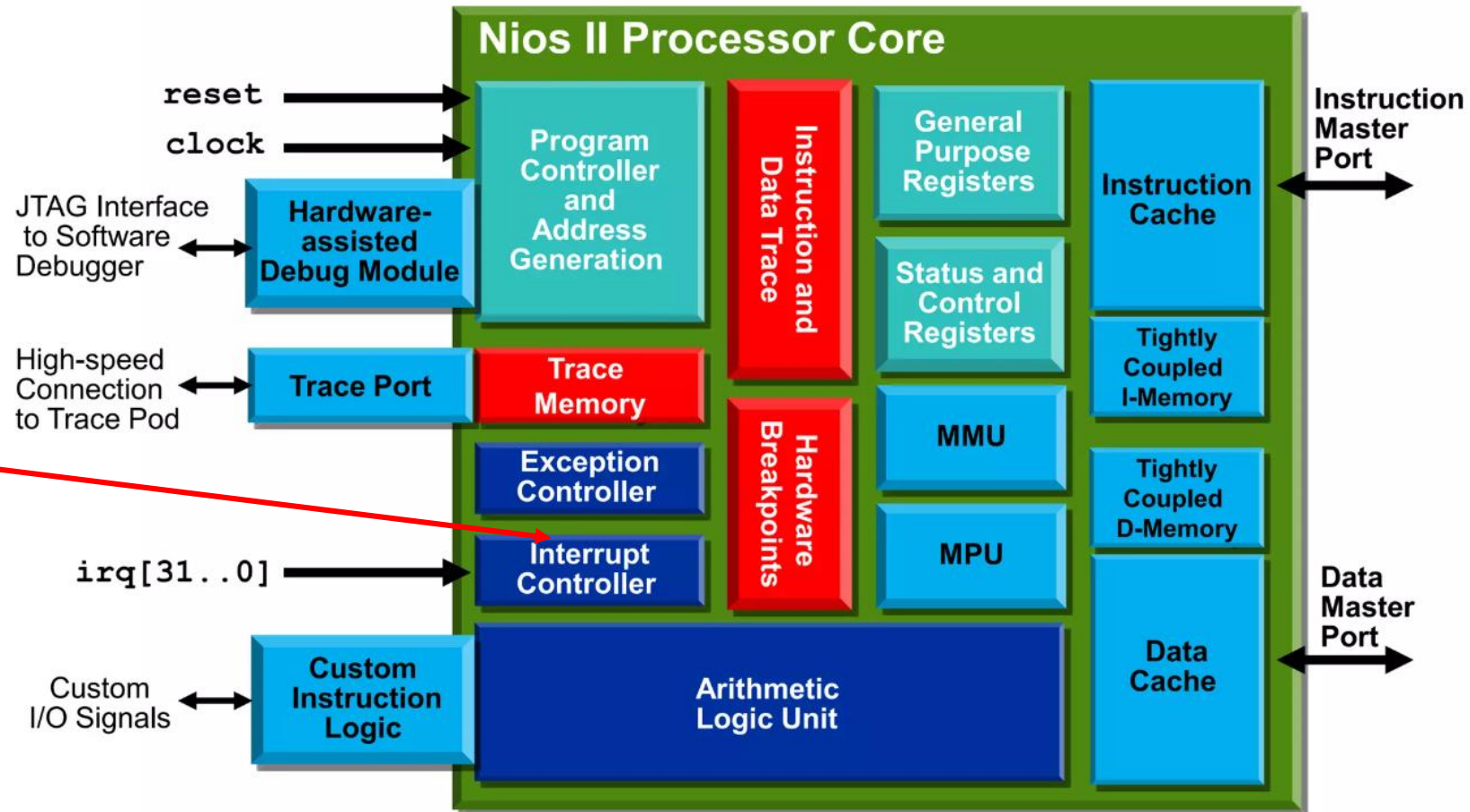
❑ **Program controller and address generation :**
responsable de l'exécution séquentielle des instructions et de la gestion du flux du programme.



Source: Intel (Altera Nios II Architecture)

Le NIOS II Embedded Processor architecture

❑ **Exception controller and interrupt controller:** sont des mécanismes essentiels pour gérer les événements imprévus ou asynchrones qui surviennent pendant l'exécution d'un programme (32 interruptions sont présent en charge)

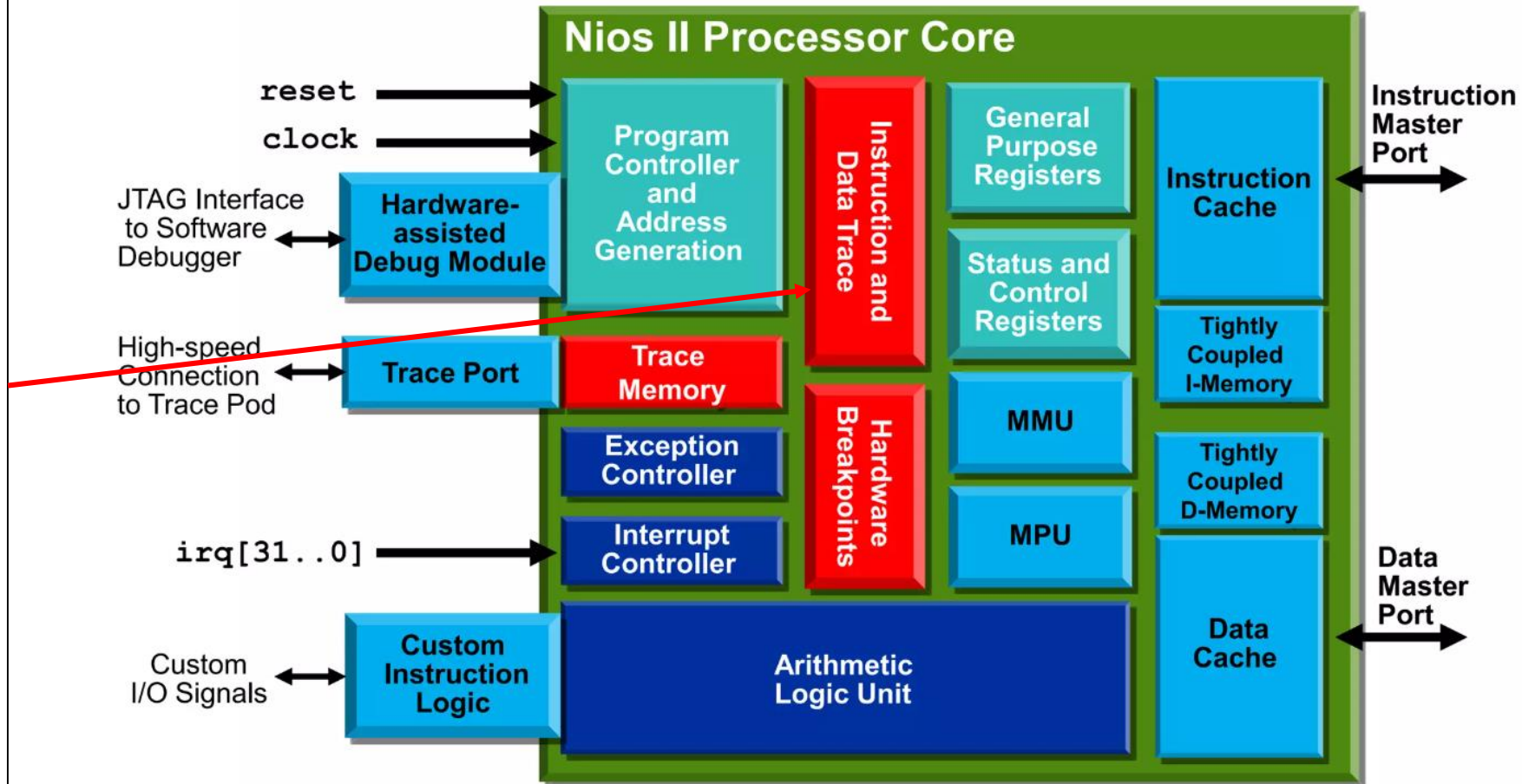


Source: Intel (Altera Nios II Architecture)

Le NIOS II Embedded Processor architecture

❑ Instruction and data trace:

sont des mécanismes qui permettent de stocker les instructions exécutées et les données obtenues dans une mémoire (**trace memory**). Cela permet de suivre le flux d'exécution du programme



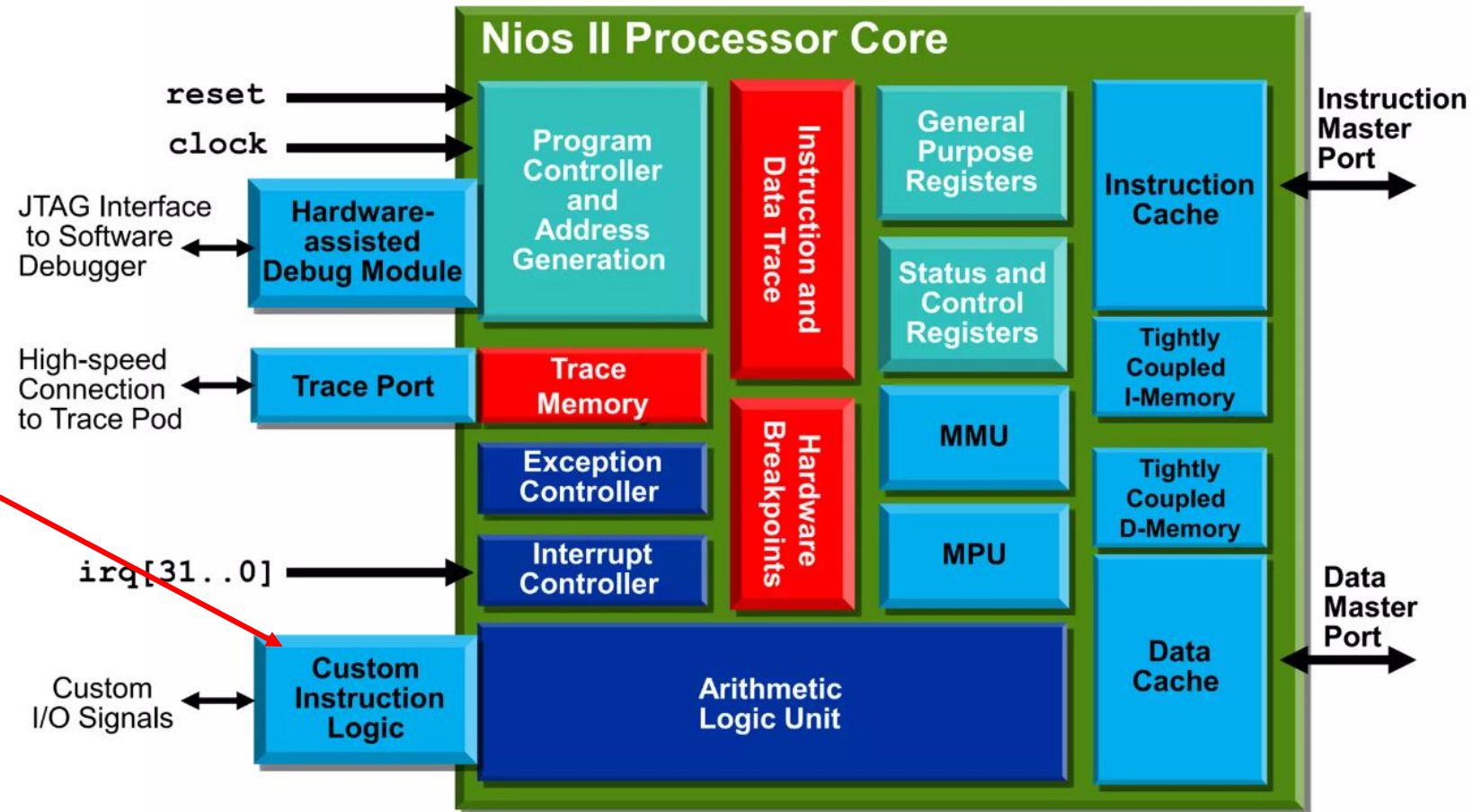
Source: Intel (Altera Nios II Architecture)

Le NIOS II Embedded Processor architecture

❑ Custom instruction logique:

sont fonctionnalités qui permettent d'ajouter des instructions spécifiques personnalisées (autres que les autres 53 instructions proposées par Intel) pour accélérer l'exécution

❑ (Exemple XOR avec décalage)

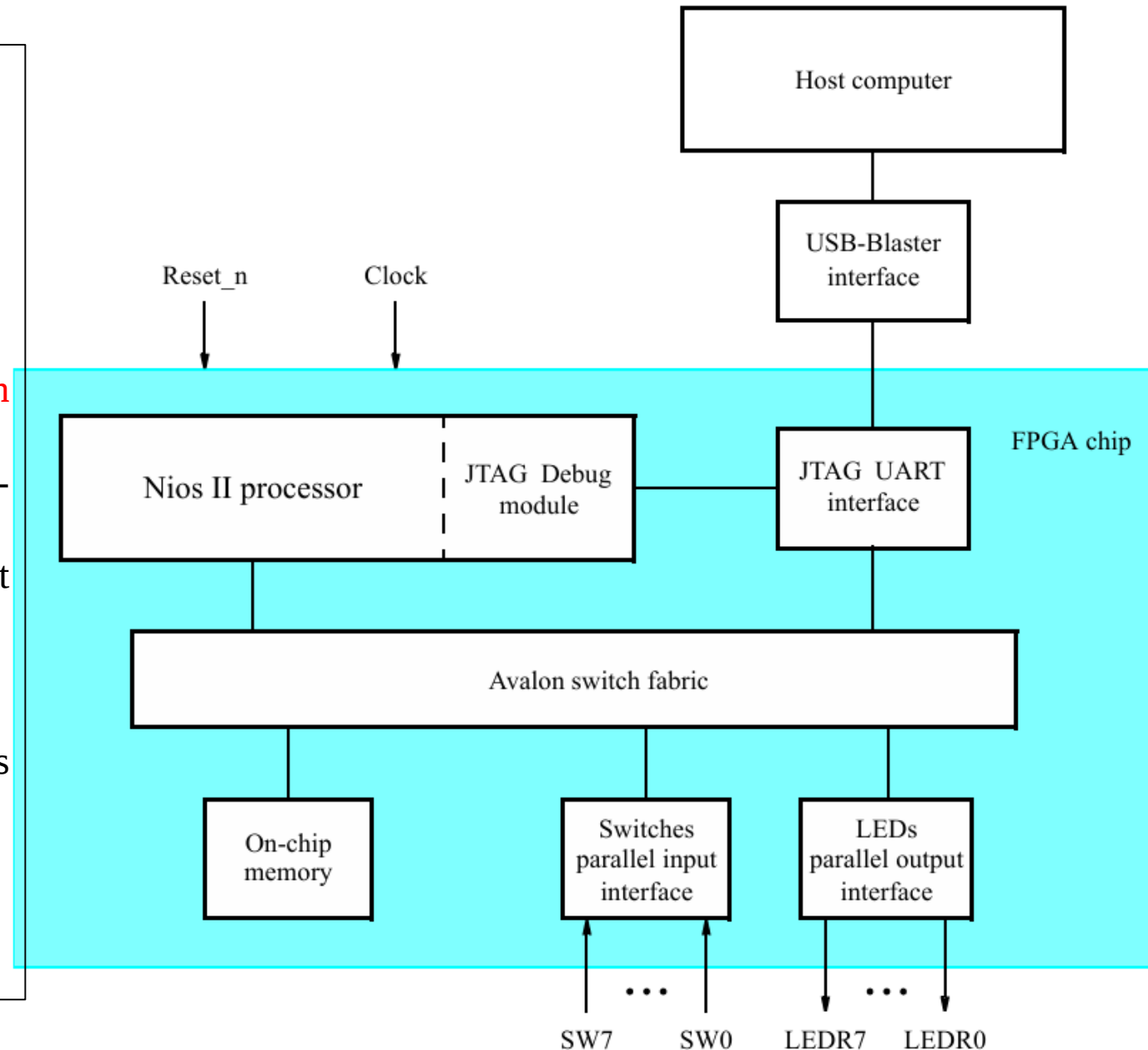


Source: Intel (Altera Nios II Architecture)

Application:
Conception et implémentation d'un SoC
sur une carte FPGA (DE2-115)

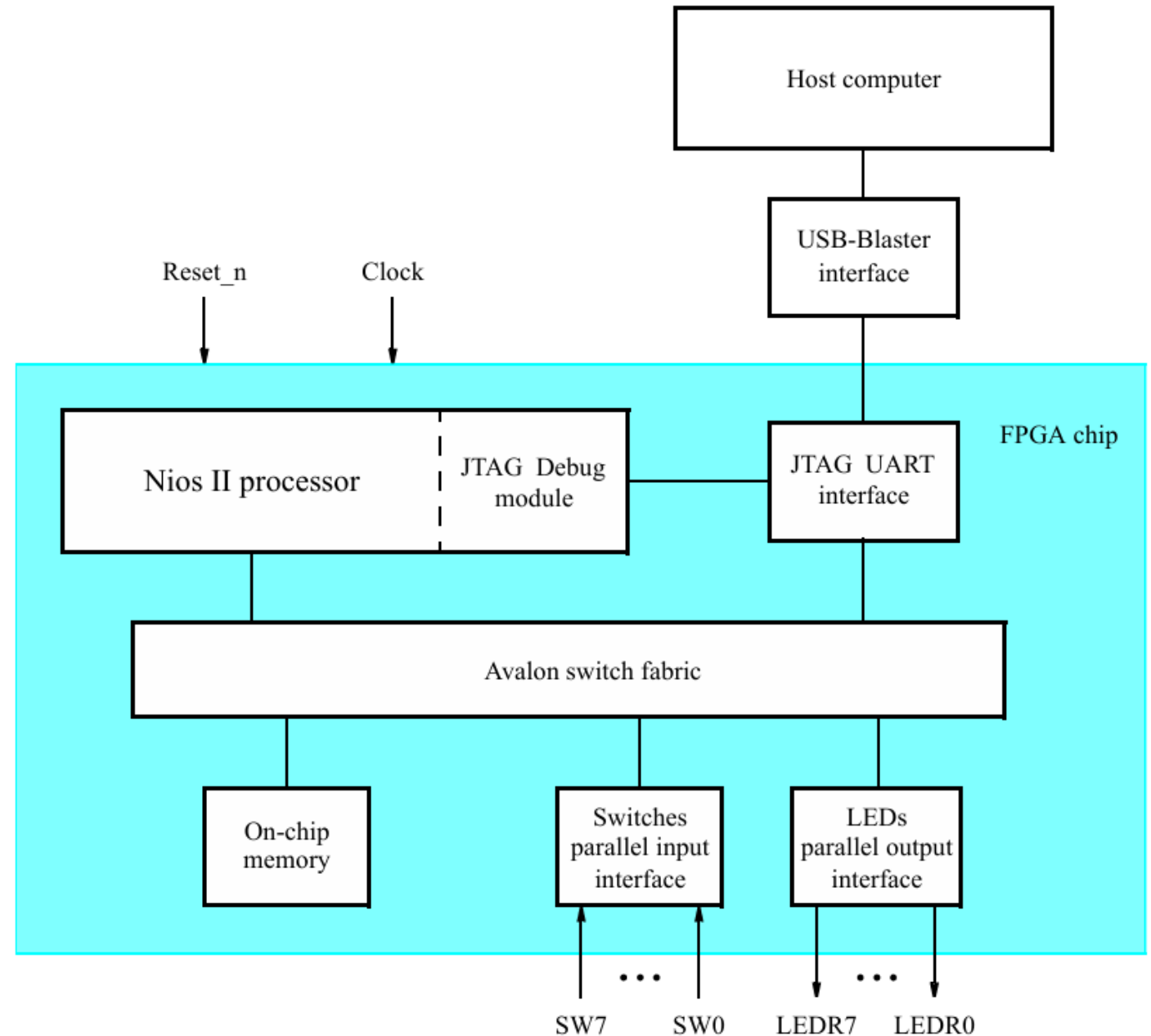
Description du SoC

- ❑ Le SoC est constitué:
- ❑ D'un Processeur Soft Nios II
- ❑ D'une mémoire de type RAM
- ❑ D'un système d'interconnexion appelé **Avalon Switch fabric** (un bus de communication flexible (Master-Slave) propre à Intel (Altera) permettant d'interconnecter plusieurs périphériques
- ❑ Des **Switches** d'entrée (inputs) et des **LEDs** de sorties (output)
- ❑ Voir slide suivant....



Description du SoC

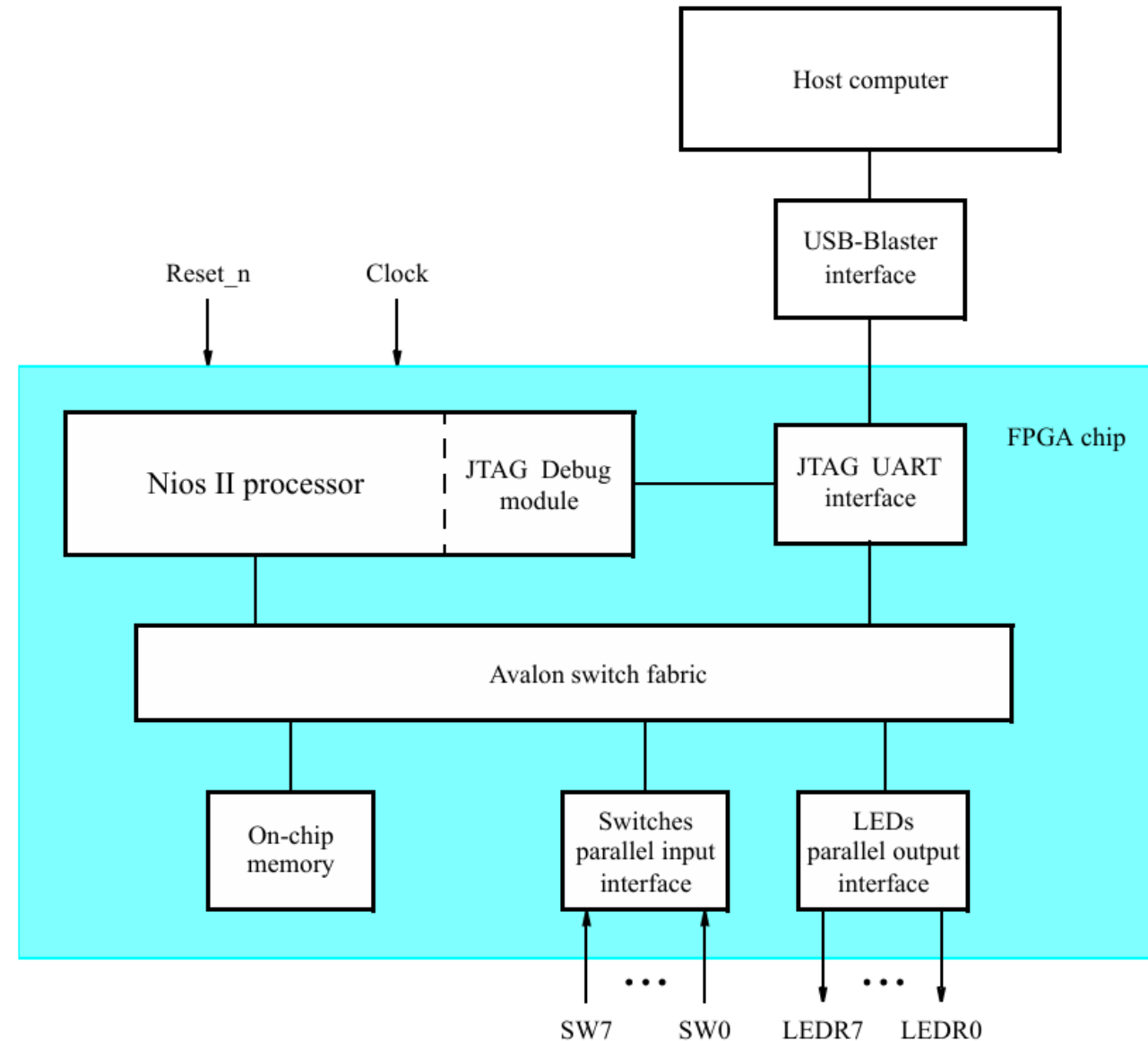
- ❑ Le SoC est constitué:
- ❑ D'une **interface JTAG UART** permettant de connecter la carte FPGA à l'ordinateur
- ❑ Un module **JTAG Debug Module** permettant d'interconnecter le processeur Nios II avec le PC pour des tâches de débogage



Description du fonctionnement SoC

Huit commutateurs sur la carte DE2-SoC, **SW7- SW0**, sont utilisés pour allumer ou éteindre huit LED, **LEDR7 - 0**.

Pour obtenir l'opération souhaitée, le motif à huit bits correspondant à l'état des commutateurs doit être envoyé au port de sortie pour activer les LED. Cela sera fait en faisant exécuter au processeur Nios II un **programme stocké dans la mémoire intégrée**. Un fonctionnement continu est nécessaire, de sorte que lorsque les commutateurs sont basculés, les lumières changent en conséquence.

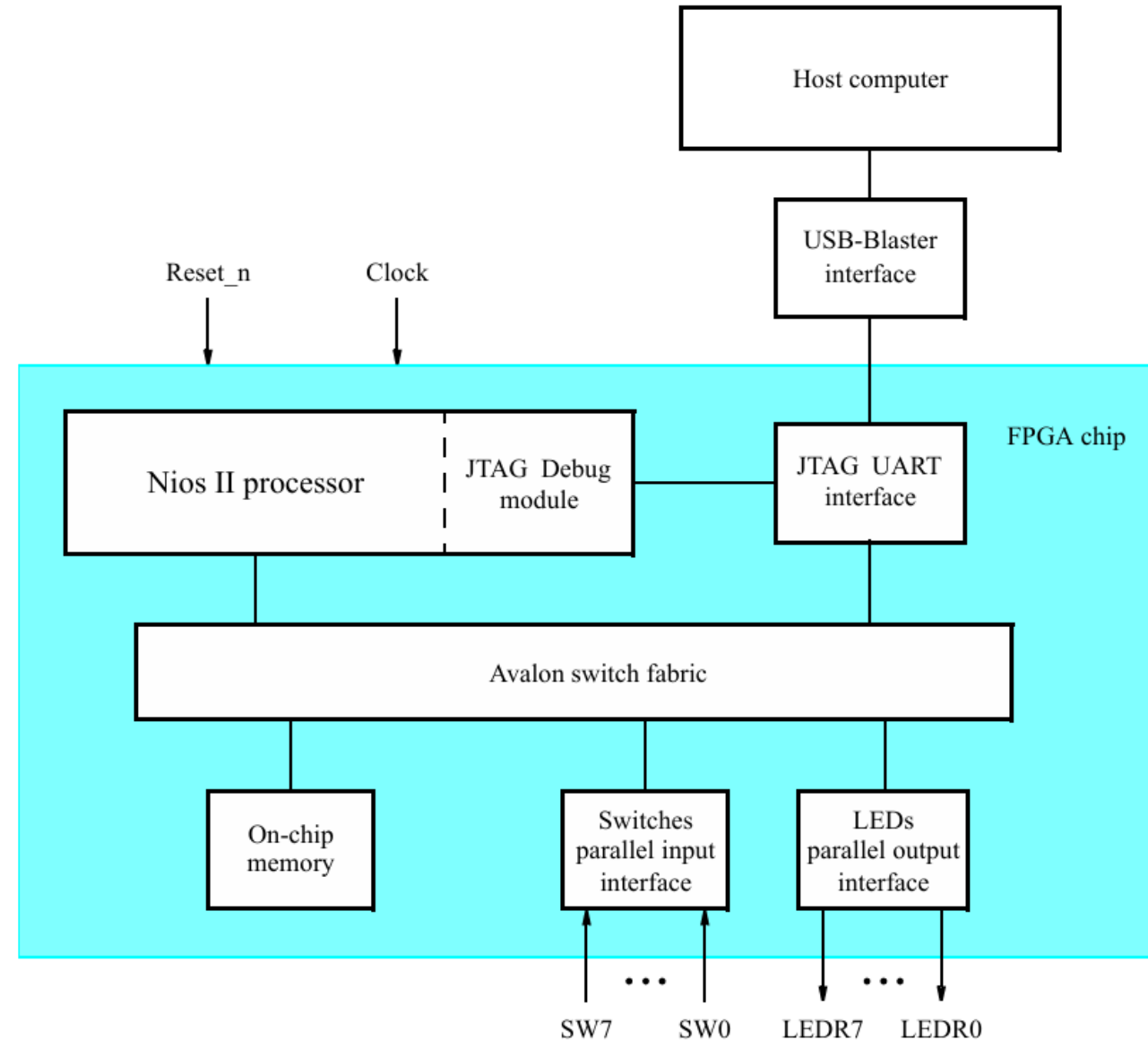


Implémentation du SoC

Pour implémenter le SoC on va utiliser l'outil **Platform Designer de Intel**

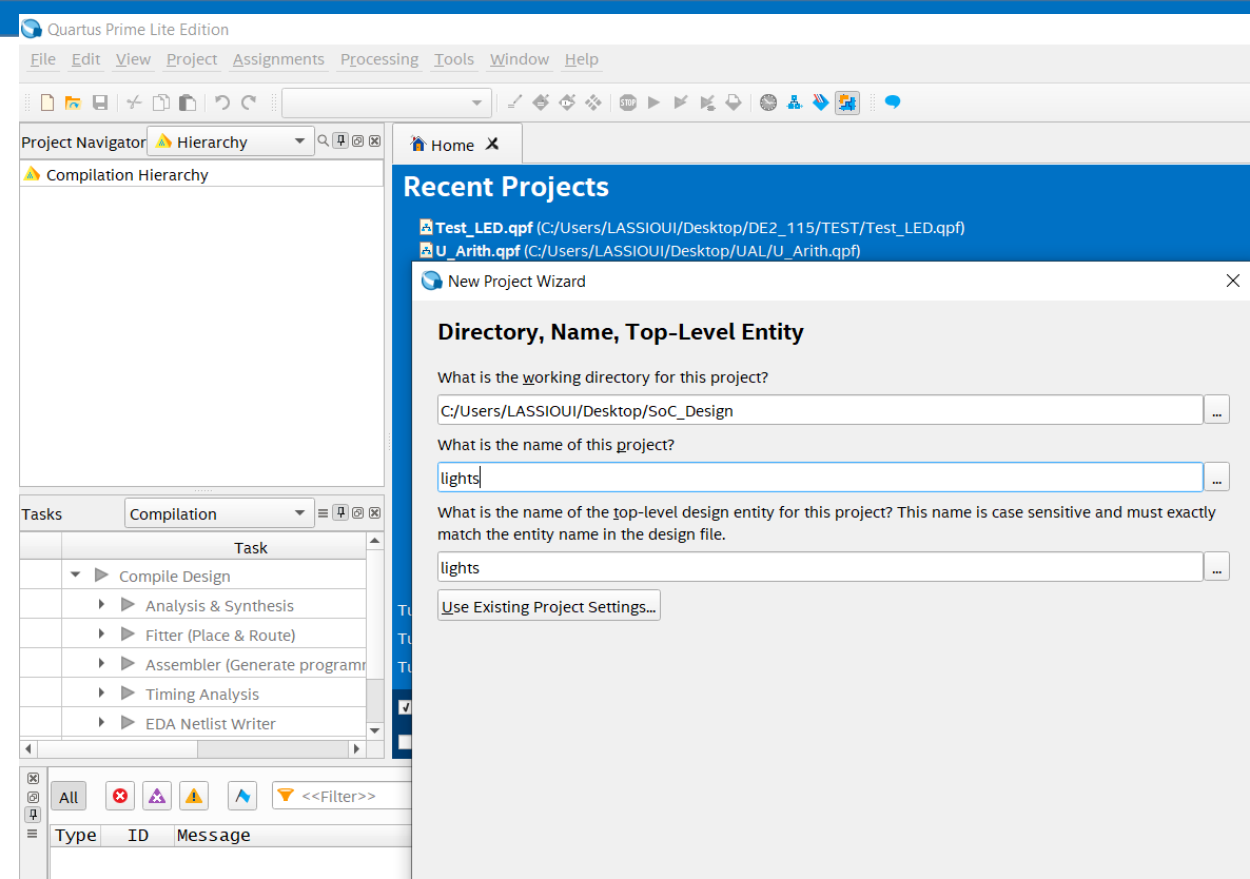
Via la plateforme, il faut instancier les éléments suivants:

- ☐ Le processeur Nios II
- ☐ Une mémoire de 4K-Byte
- ☐ Deux interfaces parallèles d'entrées/sorties
- ☐ Un module JTAG de communication avec le PC



Implémentation du SoC

- Étape 1: Création d'un projet dans Quartus Prime
- Étape 2: Choix de la famille du circuit FPGA disponible sur le DE2-115 board

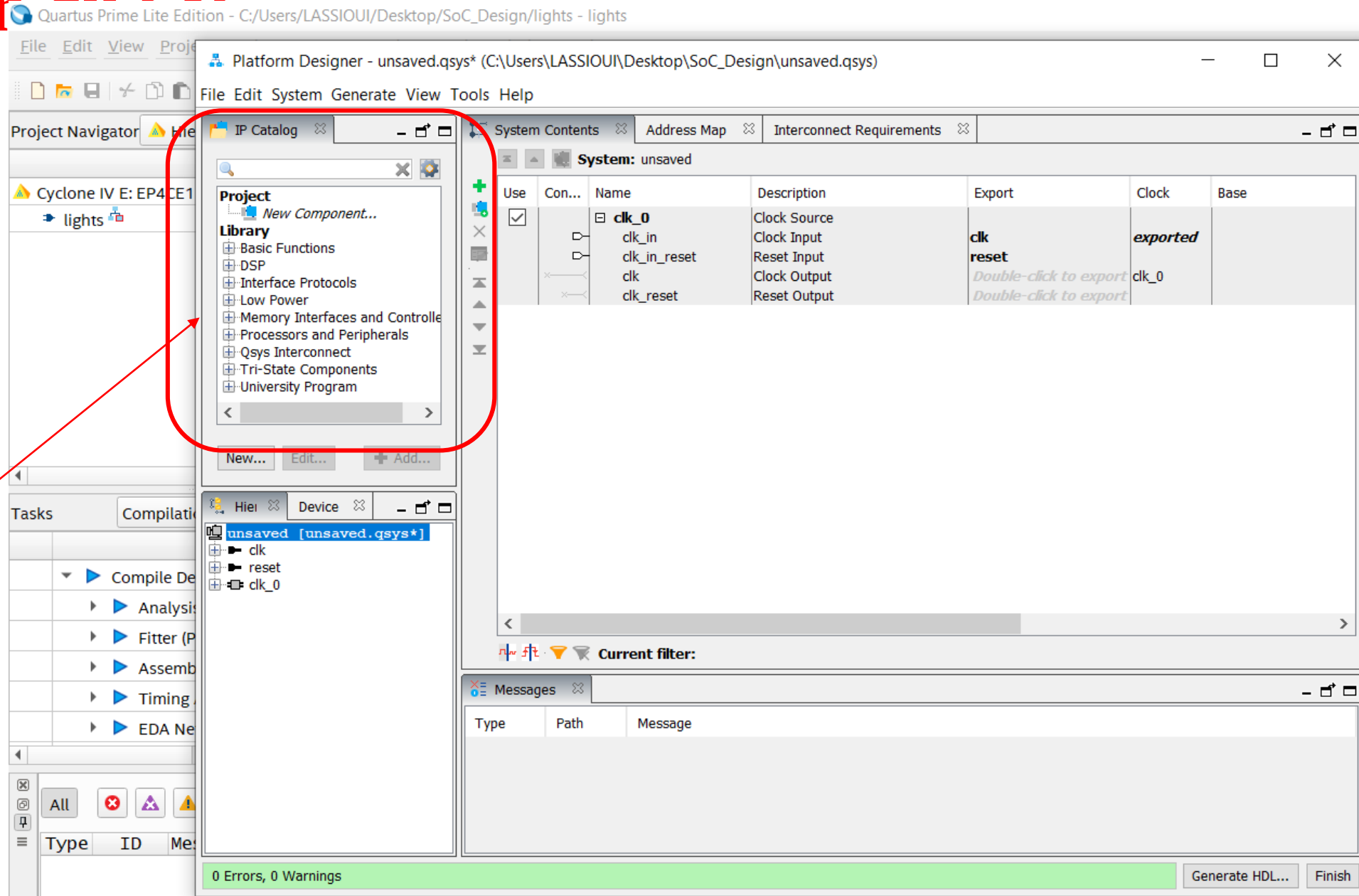


EP4CE115F23C8L	1.0V	114480	281	281	3981312	532	4
EP4CE115F23C9L	1.0V	114480	281	281	3981312	532	4
EP4CE115F23I7	1.2V	114480	281	281	3981312	532	4
EP4CE115F23I8L	1.0V	114480	281	281	3981312	532	4
EP4CE115F29C7	1.2V	114480	529	529	3981312	532	4
EP4CE115F29C8	1.2V	114480	529	529	3981312	532	4
EP4CE115F29C8L	1.0V	114480	529	529	3981312	532	4
EP4CE115F29C9L	1.0V	114480	529	529	3981312	532	4
EP4CE115F29I7	1.2V	114480	529	529	3981312	532	4

Implémentation

Étape 3: Aller vers
Tools et lancer l'outil
Platform Designer

La fenêtre IP Catalog
va nous permettre
d'ajouter les blocs IP
à notre conception



Implémentation du SoC

L'exécution du SoC est cadencé par un signal d'horloge **clk_0**. Dans notre cas, on va utiliser le signal d'horloge fourni par l'oscillateur interne de la carte DE2-115 (50MHz)

The screenshot displays the Platform Designer interface for a project named 'Platform Designer - unsaved.qsys*'. The main window is divided into several panes:

- IP Catalog:** Shows a tree view of components under 'Library', including Basic Functions, DSP, Interface Protocols, Low Power, Memory Interfaces and Controllers, Processors and Peripherals, Qsys Interconnect, Tri-State Components, and University Program.
- Hier:** Shows a hierarchical view of the design, with 'unsaved [unsaved.qsys*]' containing 'clk', 'reset', and 'clk_0'.
- System:** A table listing components in the system. The 'clk_0' component is selected, showing its connections and description.
- Parameters:** A detailed view of the 'Clock Source' component. It shows the 'Clock frequency' set to 50000000 Hz and the 'Clock frequency is known' checkbox checked. A red arrow points from the text '(50MHz)' in the left pane to this frequency value.
- Messages:** A pane at the bottom showing any messages or errors.

The status bar at the bottom indicates '0 Errors, 0 Warnings' and provides buttons for 'Generate HDL...' and 'Finish'.

Implémentation du SoC

Étape 4: Ajouter le
Processeur Nios II

Choisir la version Nios II/e (on peut
l'utiliser sans licence)

Le processeur possède deux entrées
Reset and Interrupt (quand l'une de
ces entrées est activée, le processeur
commence l'exécution à des
adresses spécifiques de la mémoire)

Edit Module - Platform Designer - unsaved.qsys* (C:\Users\LASSIOU\Desktop\SoC_Design\unsaved.qsys)

File Edit System Generate View Tools Help

IP Catalog

Project

Library

- Basic Functions
- DSP
- Interface Protocols
- Low Power
- Memory Interfaces and Controllers
- Processors and Peripherals
 - Co-Processors
 - Embedded Processors
 - Nios II Processor**
 - Nios V/c Compact Microcontroller Intel FPGA IP
 - Nios V/g General Purpose Processor Intel FPGA IP
 - Nios V/m Microcontroller Intel FPGA IP
 - Hard Processor Components
 - Hard Processor Systems
 - Inter-Process Communication
- Peripherals
- Qsys Interconnect
- Tri-State Components
- University Devices

New... Edit... Add...

Hierarchy Device Family

unsaved [unsaved.qsys*]

- clk
- reset
- clk_0
- nios2_gen2_0**
- Connections

System Contents

System: unsaved Path: nios2_gen2_0

Use	Connections	Name	Description	Export
☒	clk_0	clk_0	Clock Source	clk
☒	clk_in	clk_in	Clock Input	reset
☒	clk_in_reset	clk_in_reset	Reset Input	reset
☒	clk	clk	Clock Output	clk
☒	clk_reset	clk_reset	Reset Output	reset
☒	clk	clk	Clock Input	clk
☒	reset	reset	Reset Input	reset
☒	data_master	data_master	Avalon Memory Mapped Master	data_master
☒	instruction_master	instruction_master	Avalon Memory Mapped Master	instruction_master

Nios II Processor - nios2_gen2_0

Nios II Processor
altera_nios2_gen2

Block Diagram

Show signals

clk reset irq debug_mem_slave

clock reset interrupt avalon nios_custom_instruction

Select an Implementation

Nios II Core: ☒ Nios II/e ☐ Nios II/f

	Nios II/e	Nios II/f
Summary	Resource-optimized 32-bit RISC	Performance-optimized 32-bit RISC
Features	JTAG Debug ECC RAM Protection	JTAG Debug Hardware Multiply/Divide Instruction/Data Caches Tightly-Coupled Masters ECC RAM Protection External Interrupt Controller Shadow Register Sets MPU MMU

Error: nios2_gen2_0: Reset slave is not specified. Please select the reset slave
Error: nios2_gen2_0: Exception slave is not specified. Please select the exception slave
Warning: nios2_gen2_0: Nios II cores are not recommended for new projects and are subject to removal in a future release. Nios V cores are the recommended replacement as applicable.

Cancel Finish

Implémentation du SoC

Étape 5: Ajouter la mémoire (basic functions : On-Chip memory

Assurer que le bus est de 32 bits et la taille est de 4K-Bytes

La mémoire est de type RAM

The screenshot displays the Intel Quartus II Platform Designer interface for configuring an On-Chip Memory (RAM or ROM) Intel FPGA IP. The main window is titled "On-Chip Memory (RAM or ROM) Intel FPGA IP - onchip_memory2_0".

Left Panel (Project Hierarchy): Shows the project structure. The "Library" tab is active, displaying various IP blocks. The "On-Chip Memory" category is expanded, and "On-Chip Memory (RAM or ROM) Intel FPGA IP" is selected. A red box highlights this selection, with a red arrow pointing to the configuration window.

Right Panel (Configuration): Shows the configuration options for the selected IP block.

- Block Diagram:** Displays a block diagram of the memory block. It shows three inputs: "clk1" (clock), "s1" (avalon), and "reset1" (reset). The block is labeled "onchip_memory2_0" and "altera_avalon_onchip_memory2".
- Memory type:** The "Type" is set to "RAM (Writable)". Other options include "Dual-port access" (unchecked), "Single clock operation" (unchecked), "Read During Write Mode" (set to "DONT_CARE"), and "Block type" (set to "AUTO").
- Size:** The "Slave S1 Data width" is set to "32". The "Total memory size" is set to "4096" bytes. The "Minimize memory block usage (may impact fmax)" option is unchecked.
- Read latency:** The "Slave s1 Latency" and "Slave s2 Latency" are both set to "1".
- ROM/RAM Memory Protection:** The "Reset Request" is set to "Enabled".
- ECC Parameter:** The "Extend the data width to support ECC bits" is set to "Disabled".
- Memory initialization:** The "Initialize memory content" checkbox is checked.

Bottom Panel (Hierarchy): Shows the project hierarchy. The "Device Family" tab is active, displaying the hierarchy of the project. The "onchip_memory2_0" block is highlighted in the hierarchy.

Implémentation du SoC

Observer que la mémoire et le processeur ont été ajouté à la conception mais aucune connexion n'existe entre les deux

(C:\Users\LASSIOUI\Desktop\SoC_Design\unsaved.qsys)

File Help

System Contents Address Map Interconnect Requirements

System: unsaved Path: nios2_gen2_0

Use	Connections	Name	Description	Export	Clock	Base
☒		clk_0	Clock Source			
		clk_in	Clock Input	clk	exported	
		clk_in_reset	Reset Input	reset		
		clk	Clock Output	Double-click to export	clk_0	
☒		clk_reset	Reset Output	Double-click to export		
		nios2_gen2_0	Nios II Processor			
		clk	Clock Input	Double-click to export	unconnected	
		reset	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]	
		irq	Interrupt Receiver	Double-click to export	[clk]	
		debug_reset_requ...	Reset Output	Double-click to export	[clk]	
		debug_mem_slave	Avalon Memory Mapped Slave	Double-click to export	[clk]	
☒		onchip_memory2_0	On-Chip Memory (RAM or ROM)...			
		clk1	Clock Input	Double-click to export	unconnected	
		s1	Avalon Memory Mapped Slave	Double-click to export	[clk1]	
		reset1	Reset Input	Double-click to export	[clk1]	

PGA IP

ared Memory FIFO
eduler
)

OM) Intel FPGA IP
interface Protocol Intel
Ilel Interface Protocol In
Interface Protocol Intel

+ Add...

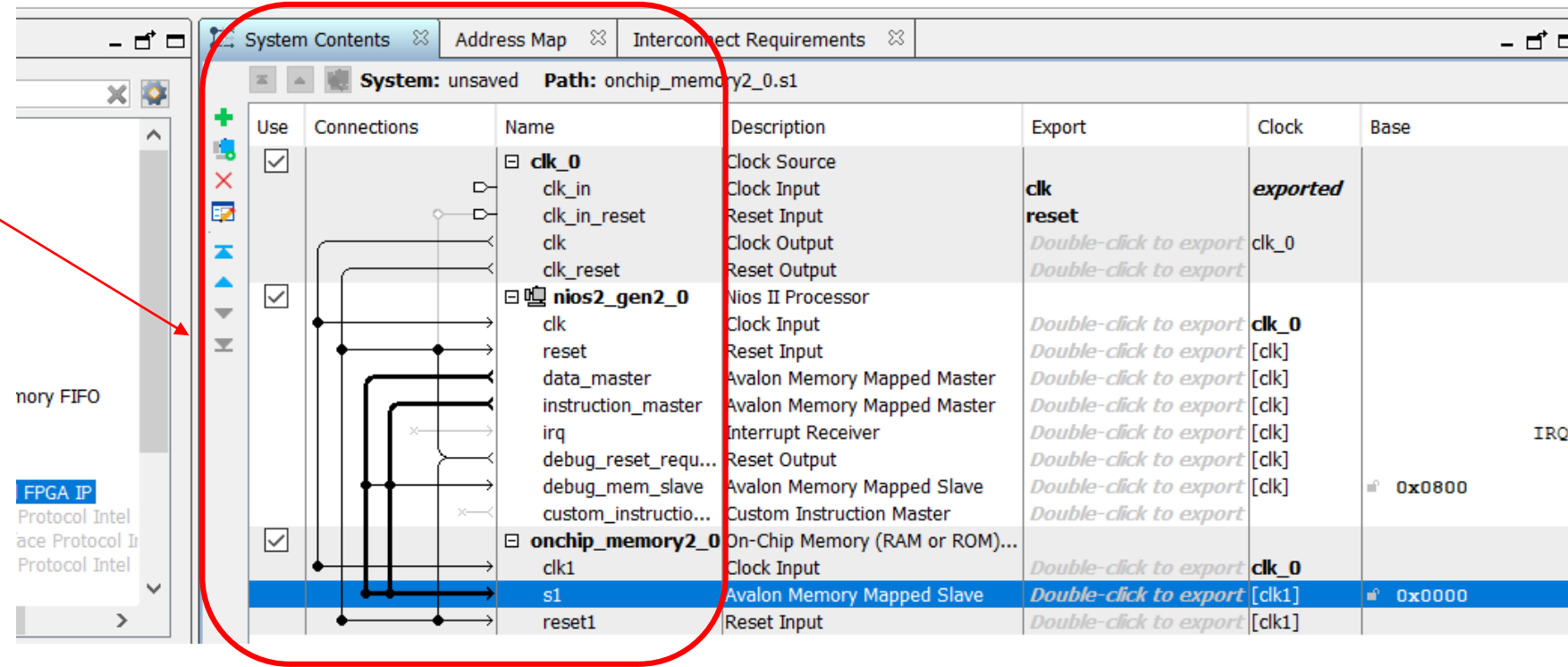
Les connexions à faire:

- ☐ Clk out vers clk_in du processeur et de la mémoire
- ☐ Reset out (clk and Jtag) vers Reset in Mémoire et Nios II
- ☐ S1 de la mémoire vers Data_Master et Instruction_master du Nios II

Implémentation du SoC

La figure suivante illustre les connexions effectuées

s:\LASSIOUI\Desktop\SoC_Design\unsaved.qsys)



Les connexions à faire:

- ❑ Clk out vers clk_in du processeur et de la mémoire
- ❑ Reset out (clk and Jtag) vers Reset in Mémoire et Nios II
- ❑ S1 de la mémoire vers Data_Master et Instruction_master du Nios II

Implémentation du SoC

Étape 6: Ajout des interfaces

de communications

Choisir la direction input et la taille 8

De la même façon, ajouter le périphérique d'interface de sortie

Edit Module - Platform Designer - unsaved.qsys* (C:\Users\LASSIOUI\Desktop\SoC_Design\unsaved.qsys)
File Edit System Generate View Tools Help

The screenshot shows the Altera Platform Designer interface. On the left, the 'IP Catalog' pane is open, displaying a tree of IP blocks. The 'PIO (Parallel I/O) Intel FPGA IP' is selected and highlighted in blue. A red arrow points from the text 'Choisir la direction input et la taille 8' to this selection. The main window displays the configuration for the 'PIO (Parallel I/O) Intel FPGA IP'. The 'Block Diagram' tab is active, showing a block labeled 'pio_0' with four input signals: 'clk', 'reset', 's1', and 'external_connection'. The 'Basic Settings' tab is also visible, showing the 'Width (1-32 bits)' set to 8 and the 'Direction' set to 'Input'. The 'Output Register' and 'Edge capture register' tabs are also visible.

Implémentation du SoC

Platform Designer - unsaved.qsys* (C:\Users\LASSIOUI\Desktop\SoC_Design\unsaved.qsys)

File Edit System Generate View Tools Help

Use	Connections	Name	Description	Export	Clock	Base	End
☒		clk_0	Clock Source				
		clk_in	Clock Input	clk	exported		
		clk_in_reset	Reset Input	reset			
		clk	Clock Output	clk_0			
		clk_reset	Reset Output	reset			
☒		nios2_gen2_0	Nios II Processor				
		clk	Clock Input	clk_0			
		reset	Reset Input	reset			
		data_master	Avalon Memory Mapped Master	data_master			
		instruction_master	Avalon Memory Mapped Master	instruction_master			
		irq	Interrupt Receiver	irq			IRQ 0
		debug_reset_requ...	Reset Output	debug_reset_request			
		debug_mem_slave	Avalon Memory Mapped Slave	debug_mem_slave			
		custom_instructio...	Custom Instruction Master	custom_instruction_master			
☒		onchip_memory2_0	On-Chip Memory (RAM or ROM)...				
		clk1	Clock Input	clk_0			
		s1	Avalon Memory Mapped Slave	s1		0x0000	0x0fff
		reset1	Reset Input	reset			
☒		pio_0	PIO (Parallel I/O) Intel FPGA IP				
		clk	Clock Input	clk_0			
		reset	Reset Input	reset			
		s1	Avalon Memory Mapped Slave	s1		0x0000	0x000f
		external_connection	Conduit	external_connection			
☒		pio_1	PIO (Parallel I/O) Intel FPGA IP				
		clk	Clock Input	clk_0			
		reset	Reset Input	reset			
		s1	Avalon Memory Mapped Slave	s1		0x0000	0x000f
		external_connection	Conduit	external_connection			

Les connexions à faire:

- ☐ Clk out vers clk_in PIO
- ☐ Reset out (clk and Jtag) vers Reset in du PIO
- ☐ S1 du PIO vers Data_Master du Nios II

Implémentation du SoC

Étape 7: Ajout de l'interface JTAG

Edit Module - Platform Designer - unsaved.qsys* (C:\Users\LASSIOUI\Desktop\SoC_Design\unsaved.qsys)

File Edit System Generate View Tools Help

IP Catalog System Contents Address Map Interconnect Requirements

Interface Protocols

- Audio & Video
- Ethernet
- JESD
- PCI Express
- Serial
 - 16550 Compatible UART Intel FPGA IP
 - Avalon I2C (Master) Intel FPGA IP
 - Avalon-ST Serial Peripheral Interface Intel FPGA IP
 - eSPI to LPC bridge Intel FPGA IP
 - Intel eSPI slave
 - JTAG UART Intel FPGA IP**
 - Lightweight UART (RS-232 Serial Port) Intel FPGA IP
 - SPI (4 Wire Serial) Intel FPGA IP
 - UART (RS-232 Serial Port) Intel FPGA IP
- SerialLite
- Transceiver PHY
- Transceiver PLL
- Low Power

JTAG UART Intel FPGA IP - jtag_uart_0

JTAG UART Intel FPGA IP
altera_avalon_jtag_uart

Block Diagram

☐ Show signals

clk reset avalon_jtag_slave clock reset avalon interrupt

Write FIFO (Data from Avalon to JTAG)

Buffer depth (bytes): 64

IRQ threshold: 8

☐ Construct using registers instead of memory blocks

Read FIFO (Data from JTAG to Avalon)

Buffer depth (bytes): 64

IRQ threshold: 8

☐ Construct using registers instead of memory blocks

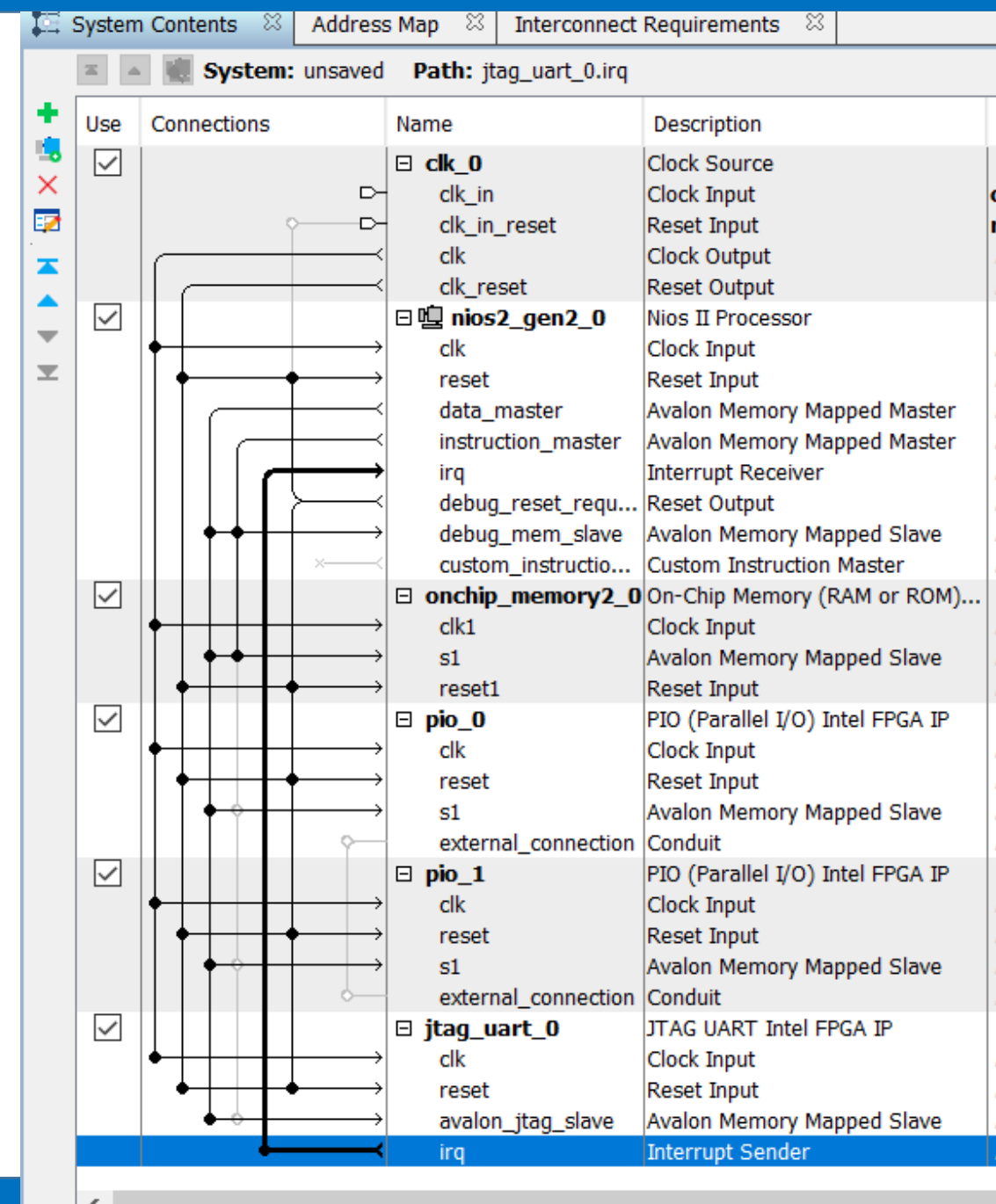
Implémentation du SoC

Étape 8: Ajouter les connexions suivantes:

Clk out vers clk Jtag

Reset out (clk et nios) vers reset JTAG

Irq (interrupt requets) vers Irq Nios II



Implémentation du SoC

Étape 8: l'objectif de cette étape

est de spécifier une adresse de
base de la mémoire

Cliquer sur On-chip memory
puis choisir l'adresse de base

suivante: **0x00000000**

Puis allez vers **System -> Assign**

Base Adresses

iers\LASSIOU\Desktop\SoC_Design\unsaved.qsys

elp

System: unsaved Path: onchip_memory2_0.s1

Use	Connections	Name	Description	Export	Clock	Base	End	IRQ	Tags
☒		clk_0	Clock Source	clk	exported				
☒		clk_in	Clock Input	Double-click to export	clk_0				
☒		clk_in_reset	Reset Input	Double-click to export					
☒		clk	Clock Output	Double-click to export					
☒		clk_reset	Reset Output	Double-click to export					
☒		nios2_gen2_0	Nios II Processor						
☒		clk	Clock Input	Double-click to export	clk_0				
☒		reset	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]				
☒		irq	Interrupt Receiver	Double-click to export	[clk]			IRQ 0	IRQ 31
☒		debug_reset_requ...	Reset Output	Double-click to export	[clk]				
☒		debug_mem_slave	Avalon Memory Mapped Slave	Double-click to export	[clk]				
☒		custom_instructio...	Custom Instruction Master	Double-click to export	[clk]				
☒		onchip_memory2_0	On-Chip Memory (RAM or ROM)...						
☒		clk1	Clock Input	Double-click to export	clk_0				
☒		s1	Avalon Memory Mapped Slave	Double-click to export	[clk1]	0x0000	0x0fff		
☒		reset1	Reset Input	Double-click to export	[clk1]				
☒		switches	PIO (Parallel I/O) Intel FPGA IP						
☒		clk	Clock Input	Double-click to export	clk_0				
☒		reset	Reset Input	Double-click to export	[clk]				
☒		s1	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x2010	0x201f		
☒		external_connection	Conduit	Double-click to export					
☒		LEDs	PIO (Parallel I/O) Intel FPGA IP						
☒		clk	Clock Input	Double-click to export	clk_0				
☒		reset	Reset Input	Double-click to export	[clk]				
☒		s1	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x2000	0x200f		
☒		external_connection	Conduit	Double-click to export					
☒		jtag_uart_0	JTAG UART Intel FPGA IP						
☒		clk	Clock Input	Double-click to export	clk_0				
☒		reset	Reset Input	Double-click to export	[clk]				
☒		avalon_jtag_slave	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x2020	0x2027		
☒		irq	Interrupt Sender	Double-click to export	[clk]				

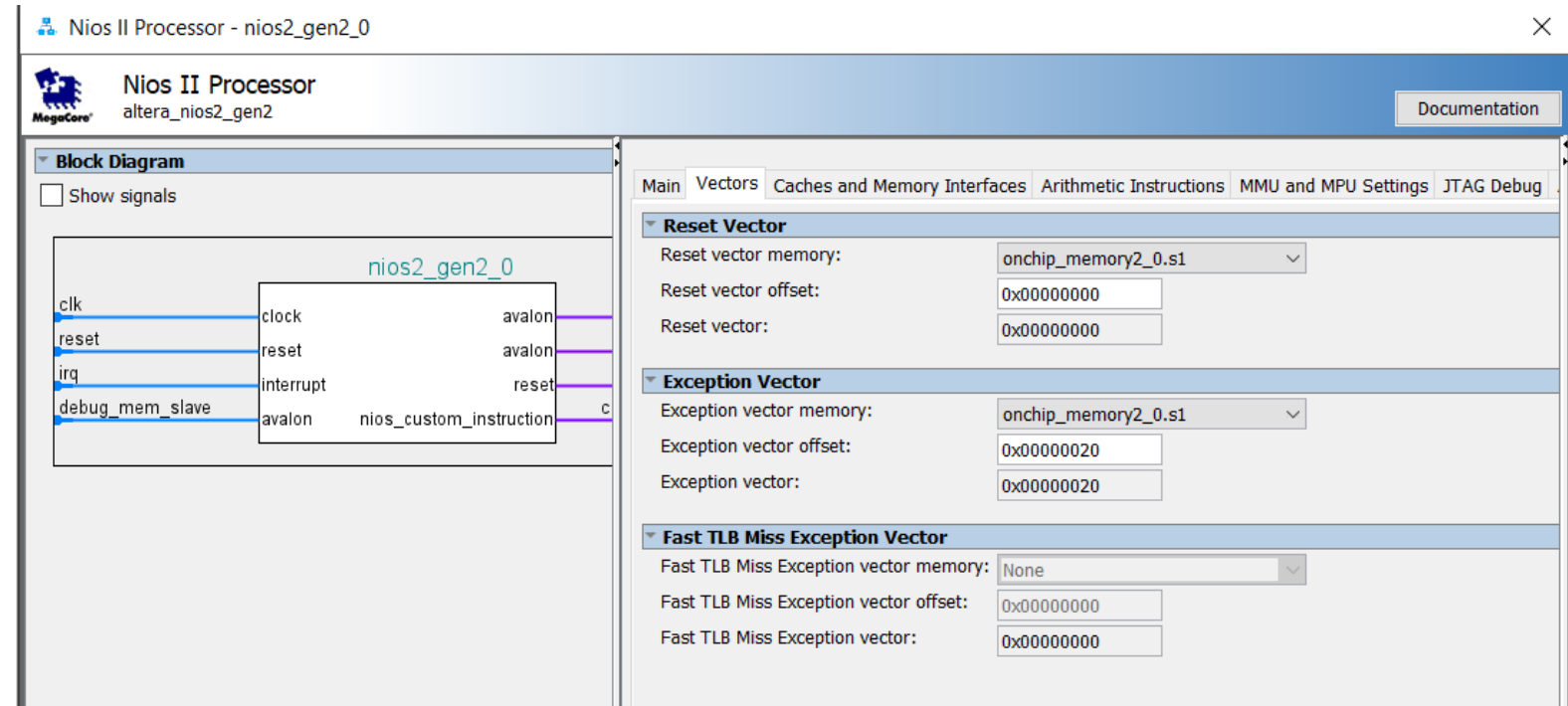
Messages

Type	Path	Message
2 Errors		
✖	unsaved.nios2_gen2_0	Reset slave is not specified. Please select the reset slave
✖	unsaved.nios2_gen2_0	Exception slave is not specified. Please select the exception slave

Implémentation du SoC

Étape 9: Spécifier la mémoire On-Chip Memory comme étant la mémoire de base utilisée par le NIOS II lors des opérations de Reset et d'interruption

Cliquez droit sur Nios II -> Edit
-> **Vectors** comme le montre la figure suivante



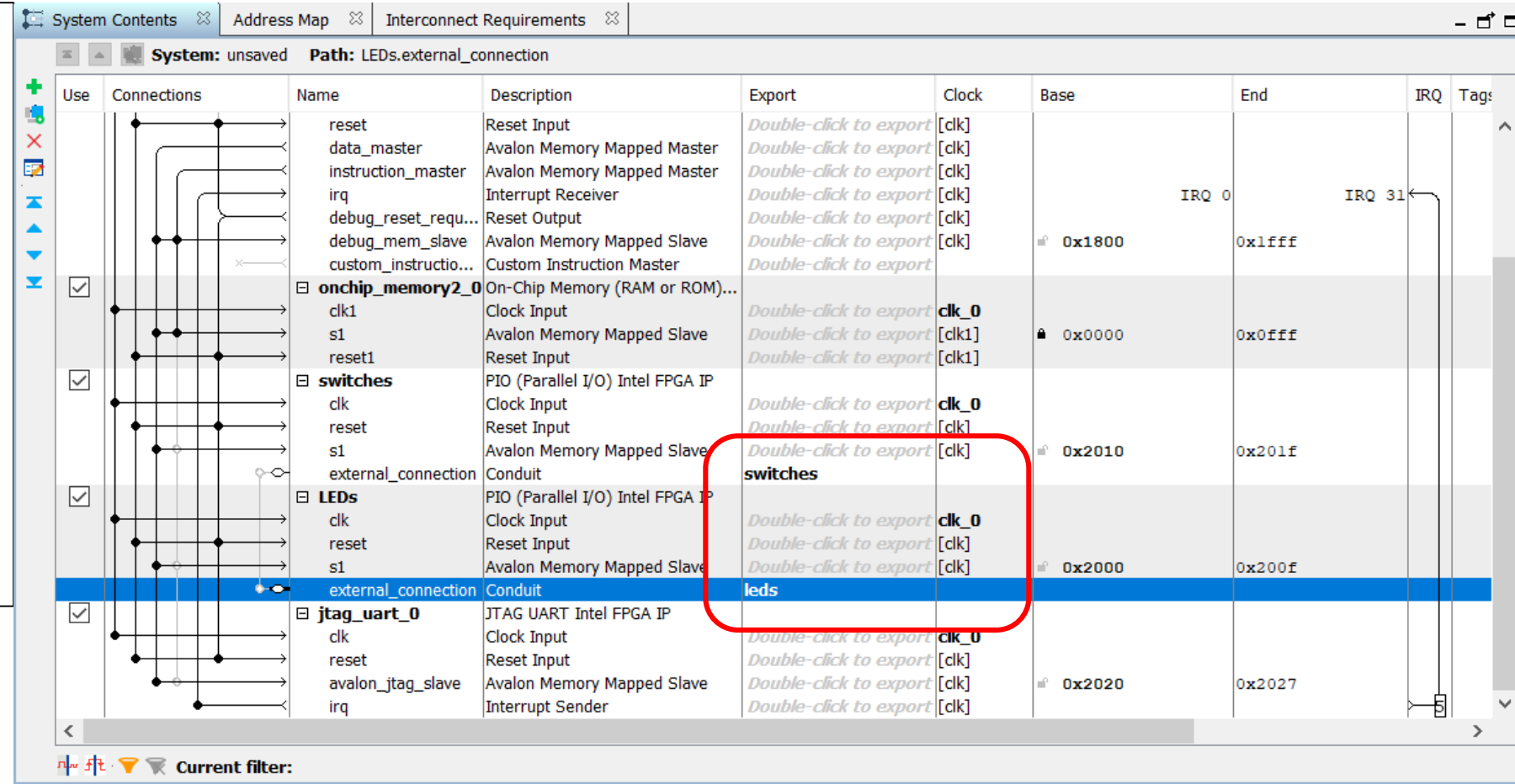
À ce stade, tous les messages d'erreurs disparaîtront

Implémentation du SoC

Étape 10: Spécifier les connexions avec les périphériques externes

Cliquez sur LEDs puis double-clic sur export (nommer le LEDS)

Même chose pour les switches



System Contents | Address Map | Interconnect Requirements

System: unsaved Path: LEDs.external_connection

Use	Connections	Name	Description	Export	Clock	Base	End	IRQ	Tag
		reset	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]				
		irq	Interrupt Receiver	Double-click to export	[clk]				
		debug_reset_requ...	Reset Output	Double-click to export	[clk]				
		debug_mem_slave	Avalon Memory Mapped Slave	Double-click to export	[clk]				
		custom_instructio...	Custom Instruction Master	Double-click to export	[clk]				
		onchip_memory2_0	On-Chip Memory (RAM or ROM)...	Double-click to export	clk_0				
		clk1	Clock Input	Double-click to export	[clk1]				
		s1	Avalon Memory Mapped Slave	Double-click to export	[clk1]				
		reset1	Reset Input	Double-click to export	[clk1]				
		switches	PIO (Parallel I/O) Intel FPGA IP	Double-click to export	clk_0				
		clk	Clock Input	Double-click to export	[clk]				
		reset	Reset Input	Double-click to export	[clk]				
		s1	Avalon Memory Mapped Slave	Double-click to export	[clk]				
		external_connection	Conduit	Double-click to export	[clk]				
		LEDs	PIO (Parallel I/O) Intel FPGA IP	Double-click to export	clk_0				
		clk	Clock Input	Double-click to export	[clk]				
		reset	Reset Input	Double-click to export	[clk]				
		s1	Avalon Memory Mapped Slave	Double-click to export	[clk]				
		external_connection	Conduit	Double-click to export	[clk]				
		jtag_uart_0	JTAG UART Intel FPGA IP	Double-click to export	clk_0				
		clk	Clock Input	Double-click to export	[clk]				
		reset	Reset Input	Double-click to export	[clk]				
		avalon_jtag_slave	Avalon Memory Mapped Slave	Double-click to export	[clk]				
		irq	Interrupt Sender	Double-click to export	[clk]				

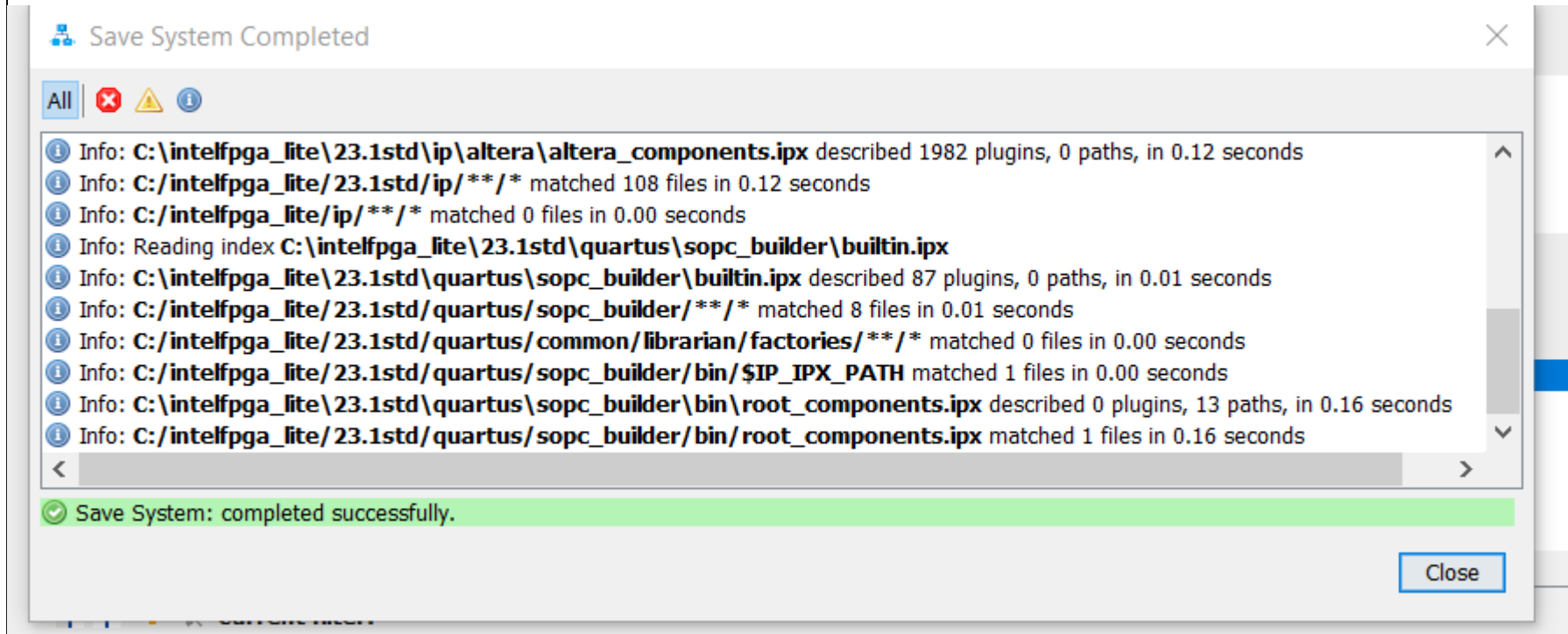
Current filter:

Implémentation du SoC

Étape 11: enregistrer le système
sous le nom `nios_system` dans le
même répertoire du projet

La figure suivante doit être
générée

Sinon, il faut revoir les étapes
précédentes

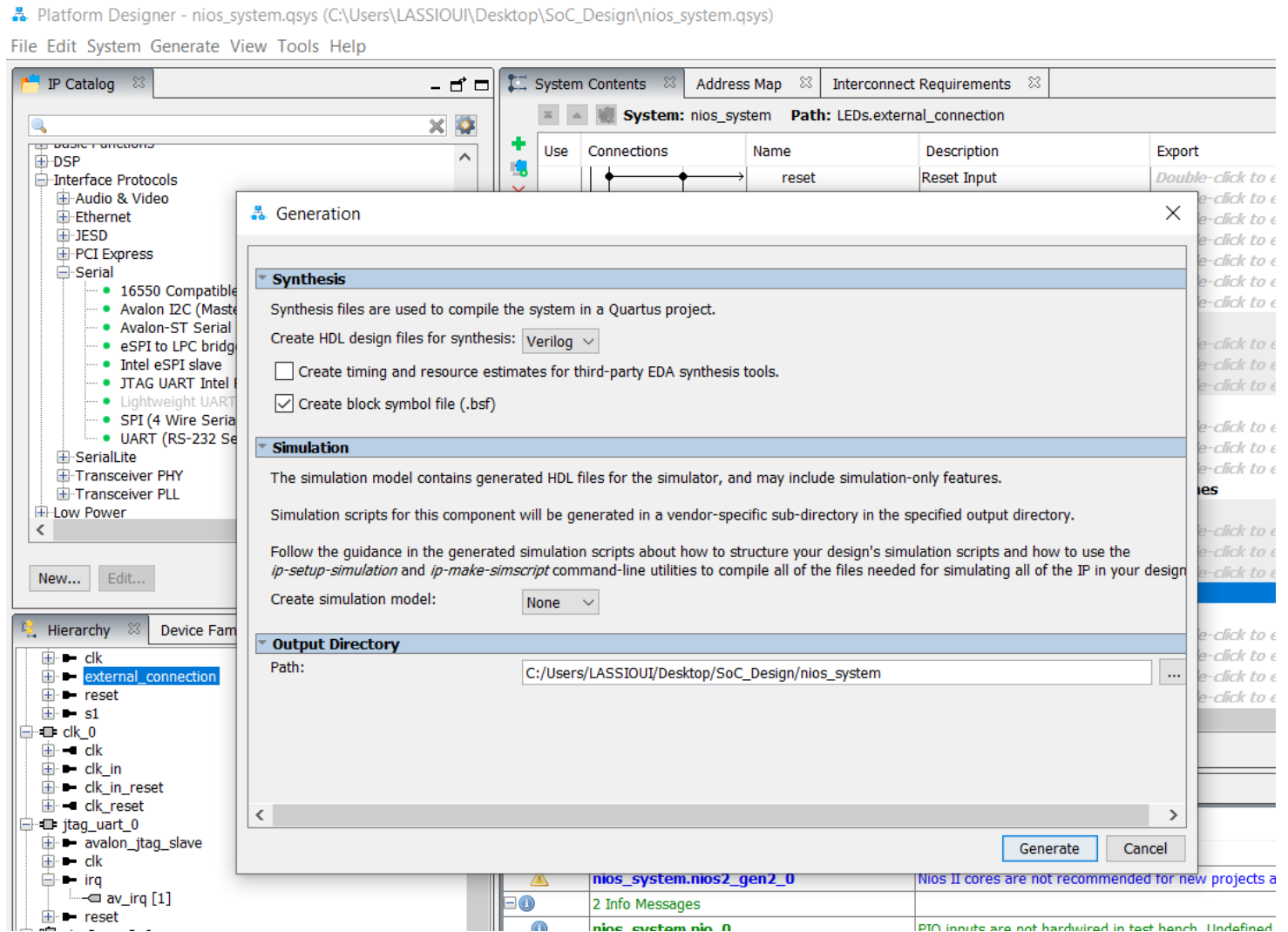


Implémentation du SoC

Étape 12: Génération du code

HDL du SoC

Allez vers Generate -> HDL



Implémentation du SoC

Intégrer le code HDL généré dans un fichier VHDL

Implémentation d

Le code généré est
très large, ce qui nous
interesse est de savoir
les noms du module
pour l'instancier
dans le code VHDL

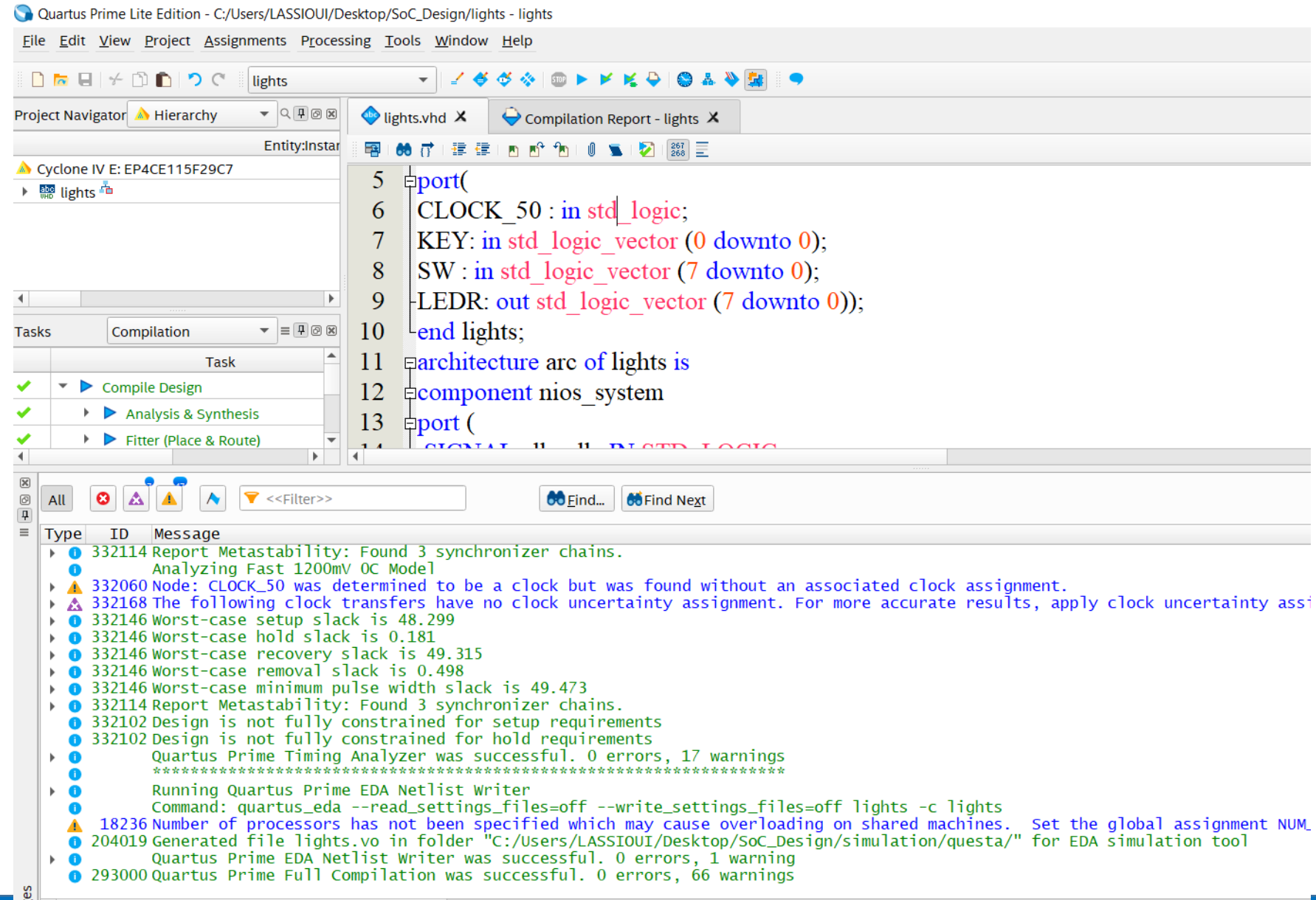
```
C:\Users\LASSIOUI\Desktop\SoC_Design\nios_system\synthesis\nios_system.v - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?

and_gate.vhd x and_gate_tb.vhd x new 1 x new 2 x new 3 x new 4 x new 5 x new 6 x nios_system.v x

1 // nios_system.v
2
3 // Generated using ACDS version 23.1 991
4
5 `timescale 1 ps / 1 ps
6 module nios_system (
7     input wire      clk_clk,           //      clk.clk
8     output wire [7:0] leds_export,     //      leds.export
9     input wire      reset_reset_n,    //      reset.reset_n
10    input wire [7:0] switches_export  //      switches.export
11 );
12
13    wire [31:0] nios2_gen2_0_data_master_readdata;
14    wire      nios2_gen2_0_data_master_waitrequest;
15    wire      nios2_gen2_0_data_master_debugaccess;
16    wire [13:0] nios2_gen2_0_data_master_address;
17    wire [3:0] nios2_gen2_0_data_master_byteenable;
18    wire      nios2_gen2_0_data_master_read;
19    wire      nios2_gen2_0_data_master_write;
20    wire [31:0] nios2_gen2_0_data_master_writedata;
21    wire [31:0] nios2_gen2_0_instruction_master_readdata;
22    wire      nios2_gen2_0_instruction_master_waitrequest;
23    wire [12:0] nios2_gen2_0_instruction_master_address;
24    wire      nios2_gen2_0_instruction_master_read;
25    wire      mm_interconnect_0_jtag_uart_0_avalon_jtag_slave_chi
26    wire [31:0] mm_interconnect_0_itag_uart_0_avalon_itag_slave_rea
```

Implémentation du SoC

Le code généré est très large, ce qui nous intéresse est de savoir les noms du module pour l'instancier dans le code VHDL



Quartus Prime Lite Edition - C:/Users/LASSIOUI/Desktop/SoC_Design/lights - lights

File Edit View Project Assignments Processing Tools Window Help

lights

Project Navigator Hierarchy Entity:Instanciation

Cyclone IV E: EP4CE115F29C7

lights

Tasks Compilation

Task

Compile Design

Analysis & Synthesis

Fitter (Place & Route)

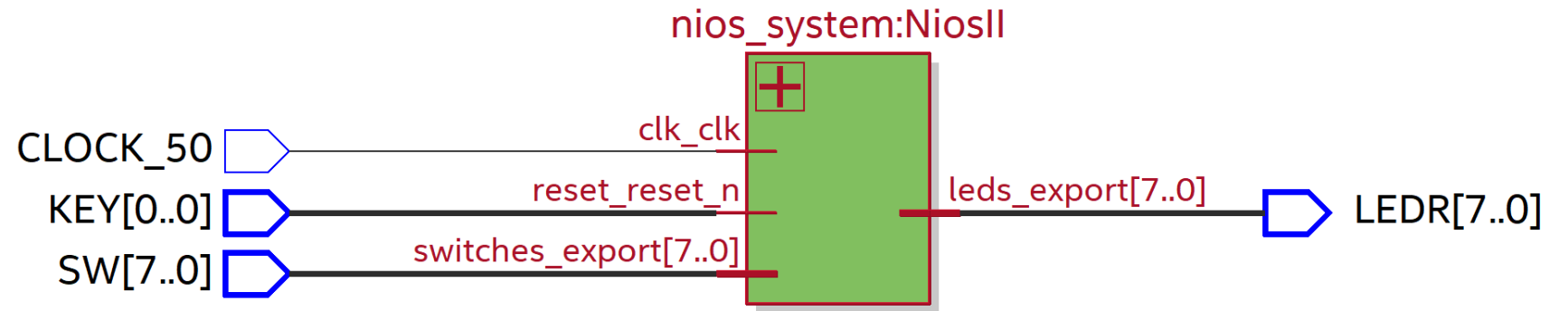
```
5 port(  
6   CLOCK_50 : in std_logic;  
7   KEY: in std_logic_vector (0 downto 0);  
8   SW : in std_logic_vector (7 downto 0);  
9   LEDR: out std_logic_vector (7 downto 0));  
10 end lights;  
11 architecture arc of lights is  
12 component nios_system  
13 port (  
14   SIGNAL_11 : in std_logic;
```

Type ID Message

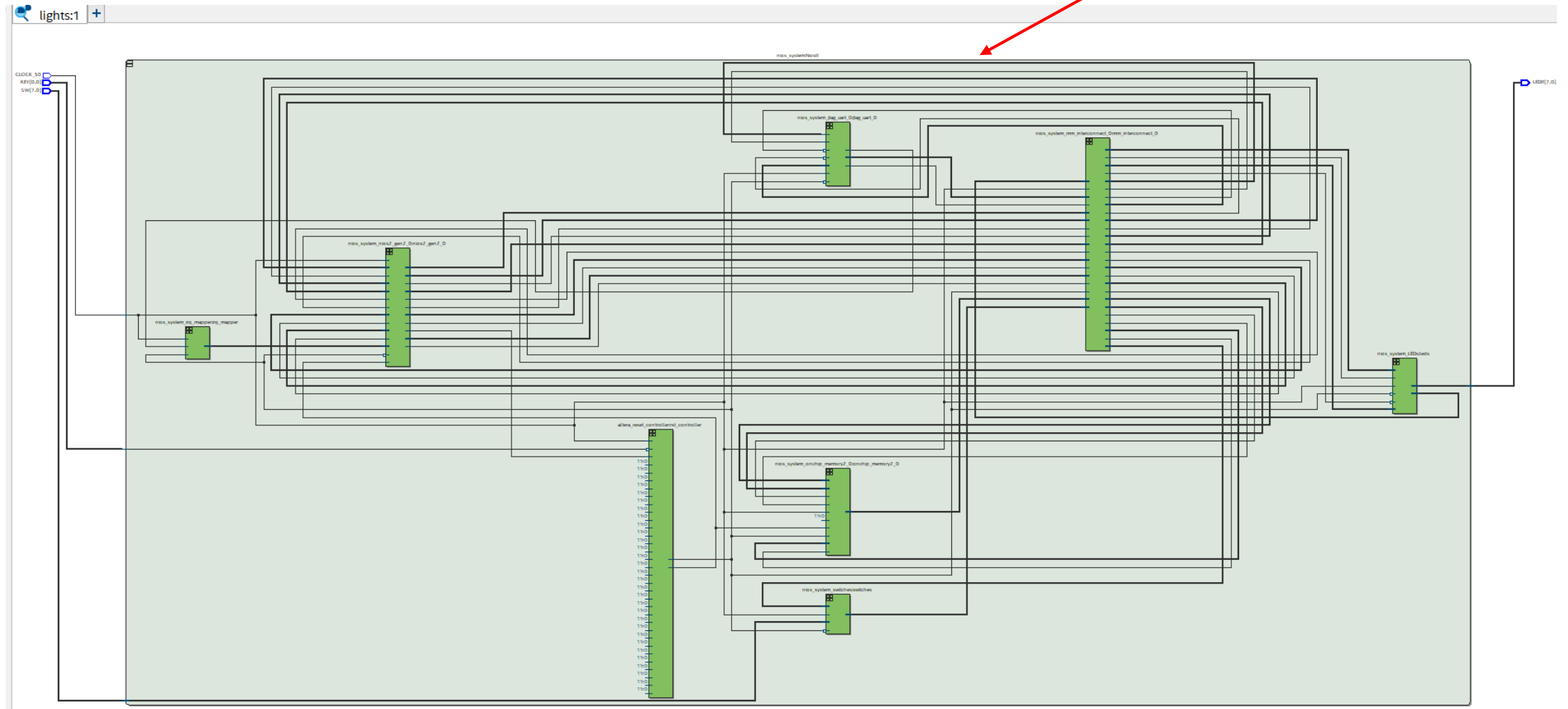
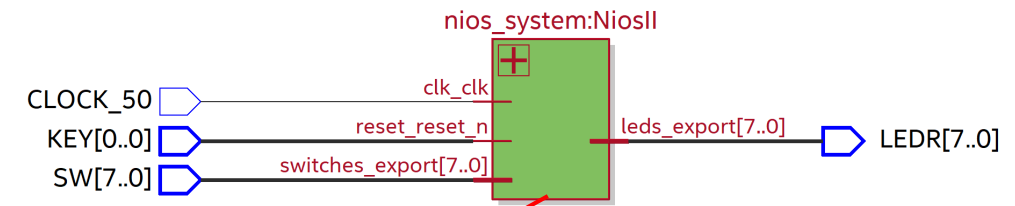
- 332114 Report Metastability: Found 3 synchronizer chains.
- 332060 Node: CLOCK_50 was determined to be a clock but was found without an associated clock assignment.
- 332168 The following clock transfers have no clock uncertainty assignment. For more accurate results, apply clock uncertainty assignment.
- 332146 Worst-case setup slack is 48.299
- 332146 Worst-case hold slack is 0.181
- 332146 Worst-case recovery slack is 49.315
- 332146 Worst-case removal slack is 0.498
- 332146 Worst-case minimum pulse width slack is 49.473
- 332114 Report Metastability: Found 3 synchronizer chains.
- 332102 Design is not fully constrained for setup requirements
- 332102 Design is not fully constrained for hold requirements
- Quartus Prime Timing Analyzer was successful. 0 errors, 17 warnings
- Running Quartus Prime EDA Netlist Writer
- Command: quartus_edu --read_settings_files=off --write_settings_files=off lights -c lights
- 18236 Number of processors has not been specified which may cause overloading on shared machines. Set the global assignment NUM
- 204019 Generated file lights.vo in folder "C:/Users/LASSIOUI/Desktop/SoC_Design/simulation/questa/" for EDA simulation tool
- Quartus Prime EDA Netlist Writer was successful. 0 errors, 1 warning
- 293000 Quartus Prime Full Compilation was successful. 0 errors, 66 warnings

Implémentation du SoC

Après la compilation,
le système généré est
illustré dans la figure
suivante



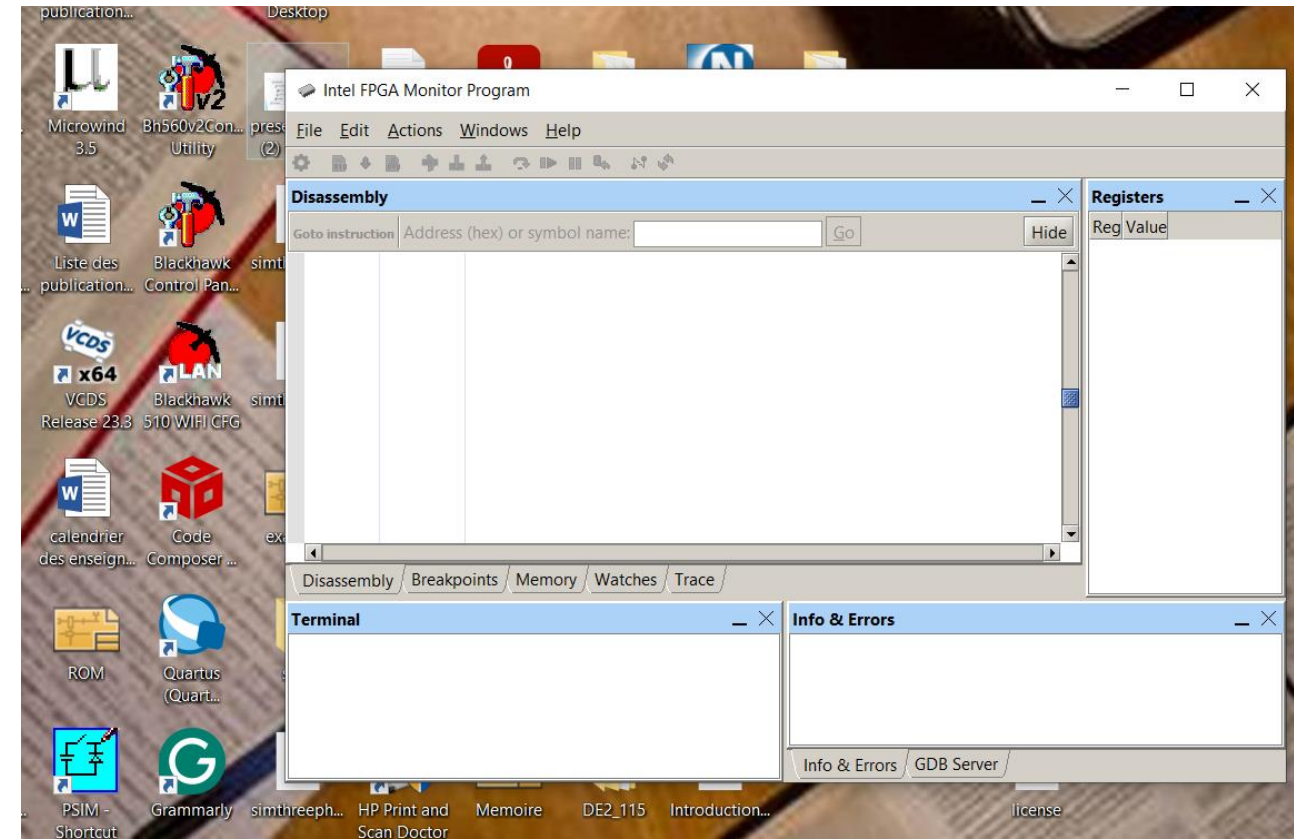
Implémentation du SoC



Implémentation du SoC

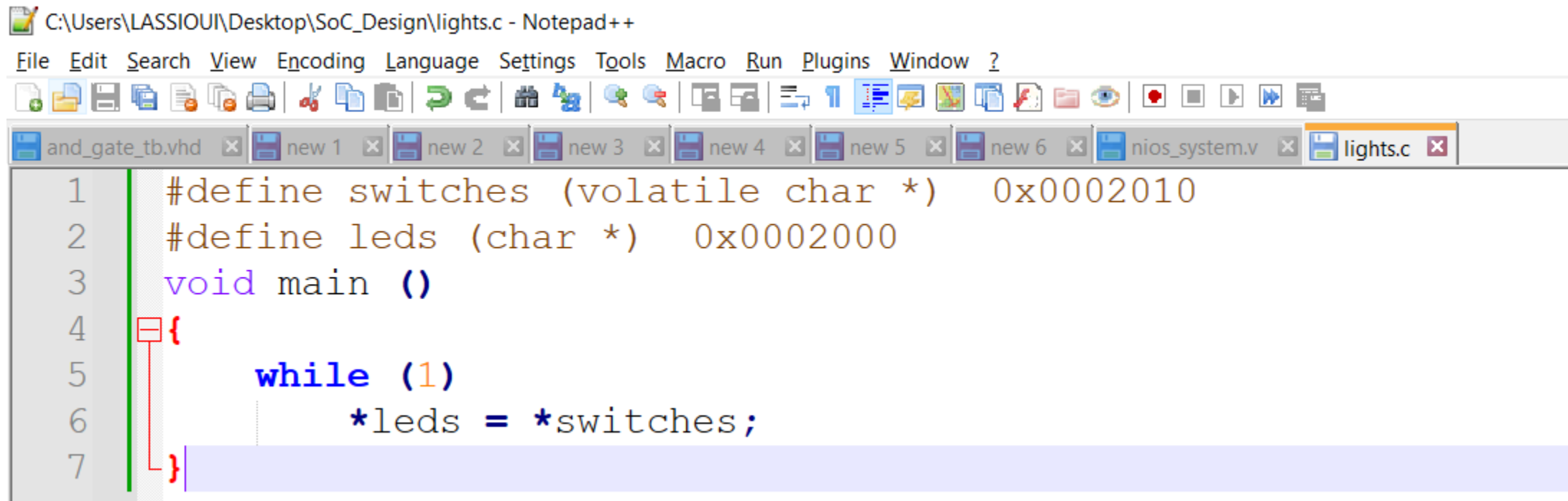
FPGA Monitor program:

Dans ce qui suit on va utiliser le logiciel FPGA Monitor Program pour charger la configuration dans le circuit FPGA et aussi pour programmer le Processeur Nios II



Implémentation du SoC

Créer un fichier lights.c dans notepad++ et enregistrer le dans le répertoire de travail



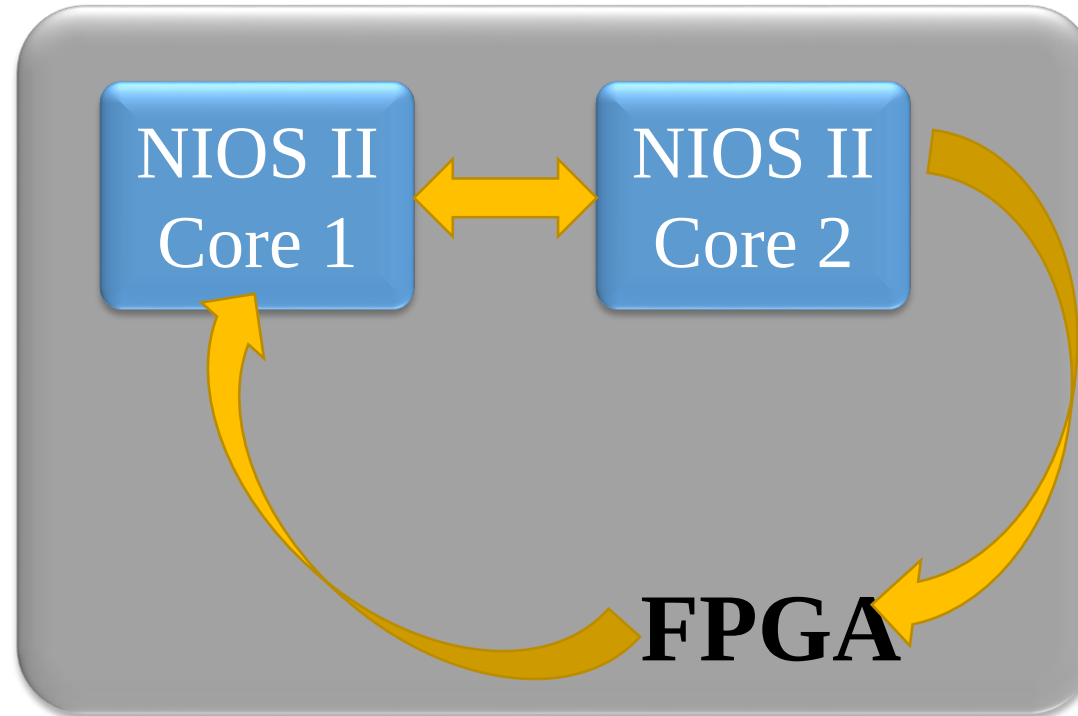
The screenshot shows the Notepad++ application window with the title bar "C:\Users\LASSIOUI\Desktop\SoC_Design\lights.c - Notepad++". The menu bar includes File, Edit, Search, View, Encoding, Language, Settings, Tools, Macro, Run, Plugins, and Window. The toolbar contains various icons for file operations and editing. The tab bar shows several open files: and_gate_tb.vhd, new 1, new 2, new 3, new 4, new 5, new 6, nios_system.v, and lights.c. The main text area displays the following C code:

```
1  #define switches (volatile char *) 0x0002010
2  #define leds (char *) 0x0002000
3  void main ()
4  {
5      while (1)
6          *leds = *switches;
7  }
```

Implémentation du SoC

Charger le fichier dans le circuit FPGA et vérifier son fonctionnement.

Application II: Implémentation d'un SoC Dual Core

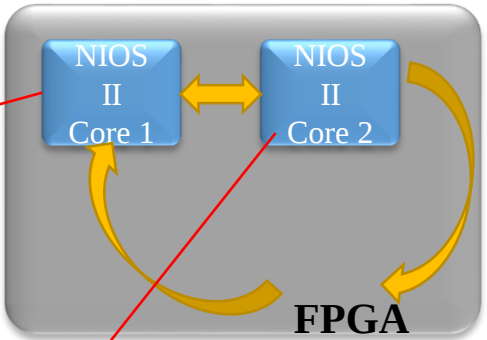


Application II: Implémentation d'un SoC Dual Core

sys (C:\Users\LASSIOUI\Desktop\Folder1\Test_SoC\Processor_1.qsys)

ate View Tools Help

Use	Connections	Name	Description	Export
☒		clk_0	Clock Source	
		clk_in	Clock Input	clk
		clk_in_reset	Reset Input	
		clk	Clock Output	
		clk_reset	Reset Output	
☒		Processor_1	Nios II (Classic) Processor	
		clk	Clock Input	
		reset_n	Reset Input	
		data_master	Avalon Memory Mapped Master	
		instruction_master	Avalon Memory Mapped Master	
		d_irq	Interrupt Receiver	
		jtag_debug_modul...	Reset Output	
		jtag_debug_module	Avalon Memory Mapped Slave	
		custom_instructio...	Custom Instruction Master	
☒		JTAG	JTAG UART	
		clk	Clock Input	
		reset	Reset Input	
		avalon_jtag_slave	Avalon Memory Mapped Slave	
		irq	Interrupt Sender	
☒		RAM	On-Chip Memory (RAM or ROM)	
		clk1	Clock Input	
		s1	Avalon Memory Mapped Slave	
		reset1	Reset Input	
☒		SWITCH	PIO (Parallel I/O)	
		clk	Clock Input	
		reset	Reset Input	
		s1	Avalon Memory Mapped Slave	
		external_connection	Conduit	
☒		LED	PIO (Parallel I/O)	
		clk	Clock Input	
		reset	Reset Input	
		s1	Avalon Memory Mapped Slave	



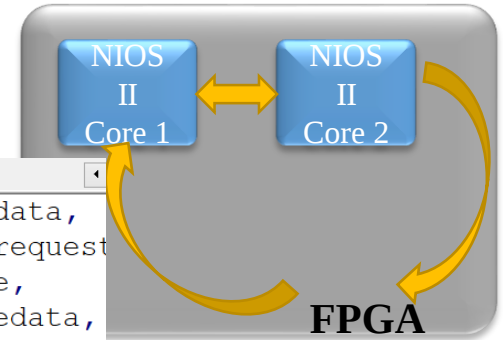
.ASSIOUI\Desktop\Folder1\Test_SoC\Processor_2.qsys)

ls Help

Use	Connections	Name	Description	Export	Clock	Base
☒		clk_0	Clock Source		exported	
		clk_in	Clock Input	clk		
		clk_in_reset	Reset Input			
		clk	Clock Output			
		clk_reset	Reset Output			
☒		Processor_2	Nios II (Classic) Processor			
		clk	Clock Input		clk_0	
		reset_n	Reset Input		[clk]	
		data_master	Avalon Memory Mapped Master		[clk]	
		instruction_master	Avalon Memory Mapped Master		[clk]	
		d_irq	Interrupt Receiver		[clk]	
		jtag_debug_modul...	Reset Output		[clk]	
		jtag_debug_module	Avalon Memory Mapped Slave		[clk]	
		custom_instructio...	Custom Instruction Master		[clk]	0x0002_00
☒		RAM_2	On-Chip Memory (RAM or ROM)			
		clk1	Clock Input		clk_0	
		s1	Avalon Memory Mapped Slave		[clk1]	0x0001_00
		reset1	Reset Input		[clk1]	
☒		INPUT_2	PIO (Parallel I/O)			
		clk	Clock Input		clk_0	
		reset	Reset Input		[clk]	
		s1	Avalon Memory Mapped Slave		[clk]	0x0002_10
		external_connection	Conduit		inputs_2	
☒		OUTPUT_2	PIO (Parallel I/O)			
		clk	Clock Input		clk_0	
		reset	Reset Input		[clk]	
		s1	Avalon Memory Mapped Slave		[clk]	0x0002_10
		external_connection	Conduit		output_2	
☒		JTAG_2	JTAG UART			
		clk	Clock Input		clk_0	

Application II: Implémentation d'un SoC Dual Core

Un extrait du code vhdl généré par Qsys



```
Processor_1_inst.vhd x new 8 x AND_GATE.vhd x AND_GATE_TB.vhd x test.php x test_db.php x Processor_2_inst.vhd x Processor_2.vhd x
337 jtag_debug_module_readdata => mm_interconnect_0_processor_2_jtag_debug_module_readdata,
338 jtag_debug_module_waitrequest => mm_interconnect_0_processor_2_jtag_debug_module_waitrequest,
339 jtag_debug_module_write => mm_interconnect_0_processor_2_jtag_debug_module_write,
340 jtag_debug_module_writedata => mm_interconnect_0_processor_2_jtag_debug_module_writedata,
341 no_ci_readra => open
342 );
343
344 ram_2 : component Processor_2_RAM_2
345 port map (
346     clk => clk_clk, -- clk1.clk
347     address => mm_interconnect_0_ram_2_s1_address, -- s1.address
348     clken => mm_interconnect_0_ram_2_s1_clken, -- .clken
349     chipselect => mm_interconnect_0_ram_2_s1_chipselect, -- .chipselect
350     write => mm_interconnect_0_ram_2_s1_write, -- .write
351     readdata => mm_interconnect_0_ram_2_s1_readdata, -- .readdata
352     writedata => mm_interconnect_0_ram_2_s1_writedata, -- .writedata
353     byteenable => mm_interconnect_0_ram_2_s1_byteenable, -- .byteenable
354     reset => rst_controller_reset_out_reset, -- reset1.reset
355     reset_req => rst_controller_reset_out_reset_req, -- .reset_req
356     freeze => '0' -- (terminated)
357 );
358
359 mm_interconnect_0 : component Processor_2_mm_interconnect_0
360 port map (
361     clk_0_clk_clk => clk_clk,
362     Processor_2_reset_n_reset_bridge_in_reset_reset => rst_controller_reset_out_reset,
363     Processor_2_data_master_address => processor_2_data_master_address,
```

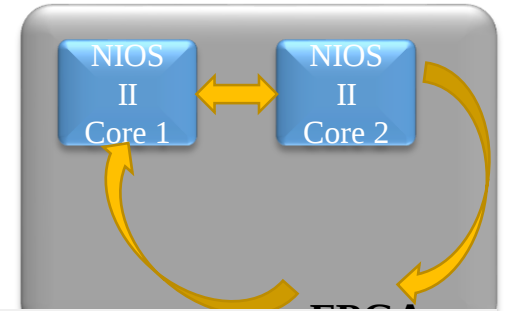
Application II: Implémentation d'un SoC Dual Core

C:/Users/LASSIOUI/Desktop/Folder1/Test_SoC/Test_SoC - Test_SoC

Assignments Processing Tools Window Help

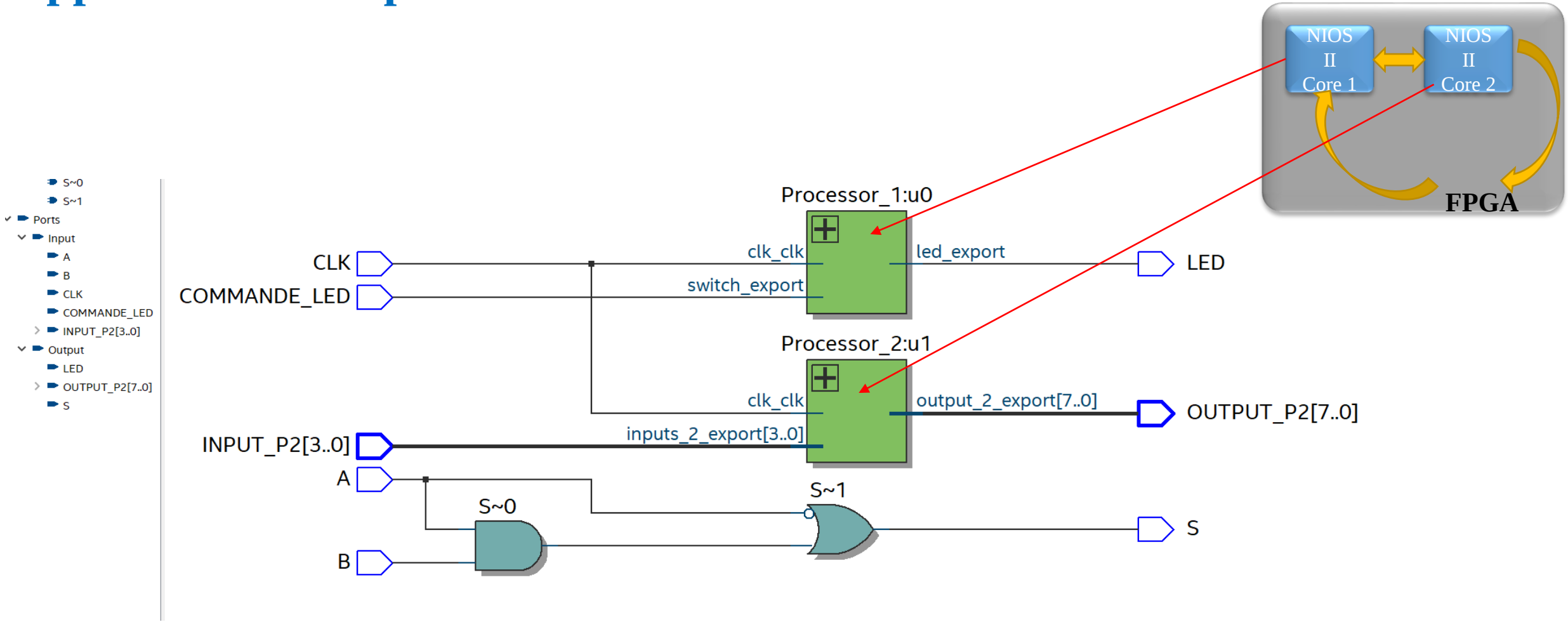
```
Test_SoC
Test_SoC.vhd
Compilation Report - Test_SoC

1 library ieee;
2 use ieee.std_logic_1164.all;
3 entity Test_SoC is
4 port (
5 INPUT_P2 : IN std_logic_vector(3 downto 0);
6 OUTPUT_P2 : OUT std_logic_vector(7 downto 0) ;
7 A, B: in std_logic;
8 S: out std_logic;
9 CLK: in std_logic;
10 COMMANDE_LED: in std_logic;
11 LED: out std_logic);
12 end entity;
13 architecture arc of Test_SoC is
14
15 -----
16 component Processor_1 is
17 port (
18 clk_clk : in std_logic := 'X'; -- clk
19 switch_export : in std_logic := 'X'; -- export
20 led_export : out std_logic -- export
21 );
22 end component Processor_1;
23
```



```
23
24 -----
25 component Processor_2 is
26 port (
27 clk_clk : in std_logic := 'X';
28 inputs_2_export : in std_logic_vector(3 downto 0) := (others:
29 output_2_export : out std_logic_vector(7 downto 0)
30 );
31 end component Processor_2;
32 -----
33 begin
34 u0 : component Processor_1
35 port map (
36 clk_clk => CLK, -- clk.clk
37 switch_export => COMMANDE_LED, -- switch.export
38 led_export => LED -- led.export
39 );
40 -----
41
42 u1 : component Processor_2
43 port map (
44 clk_clk => CLK, -- clk.clk
45 inputs_2_export => INPUT_P2, -- inputs_2.export
46 output_2_export => OUTPUT_P2 -- output_2.export
47 );
48 -----
49
50 S <= (A and B) or not(A);
51
52 end architecture;
53
```

Application II: Implémentation d'un SoC Dual Core



Application II: Implémentation d'un SoC Dual Core

Programme en langage C du processeur N1

Ce code permet de contrôler l'état de la led en utilisant le switch

Nios II - Processor_1/hello_world_small.c - Eclipse

File Edit Source Refactor Navigate Search Project Nios II Run Window Help

```
/* "Small Hello World" example. */
#include "system.h"
#include "stdio.h"
#include "altera_avalon_pio_regs.h"

int main()
{
    printf("Hello from Nios II!\n");

    /* Event loop never exits. */
    while (1)
    {
        IOWR_ALTERA_AVALON_PIO_DATA(LED_BASE, IORD_ALTERA_AVALON_PIO_DATA(SWITCH_BASE));
    }

    //return 0;
}
```

