Architecture des systèmes embarqués
C2 : Processeur NIOS II

Yann DOUZE

Yann.douze@sorbonne-universite.fr

Qu'est ce qu'un « Soft » processeur ?

- Un processeur décrit dans un langage HDL (VHDL, Verilog) et qui peut être implémenté dans un FPGA
- On peut implémenter plusieurs processeurs en parallèle sur le même FPGA (MPCore : Multi Processeur Core)
- On peut utiliser le reste des ressources du FPGA pour accélérer ou customiser le processeur.
- Egalement appelé « Softcore »

Pourquoi utiliser un « Soft » processeur ?

- **Haut niveau de réutilisation (Design Reuse)**
- **Réduction du risque d'obsolescence**
- **Simplifie la mise à jour ou les modifications**
- **Davantage d'options de mise en œuvre**
- **Plus faible latence entre le processeur et les composants du FPGA.**

Outils de Intel-Altera

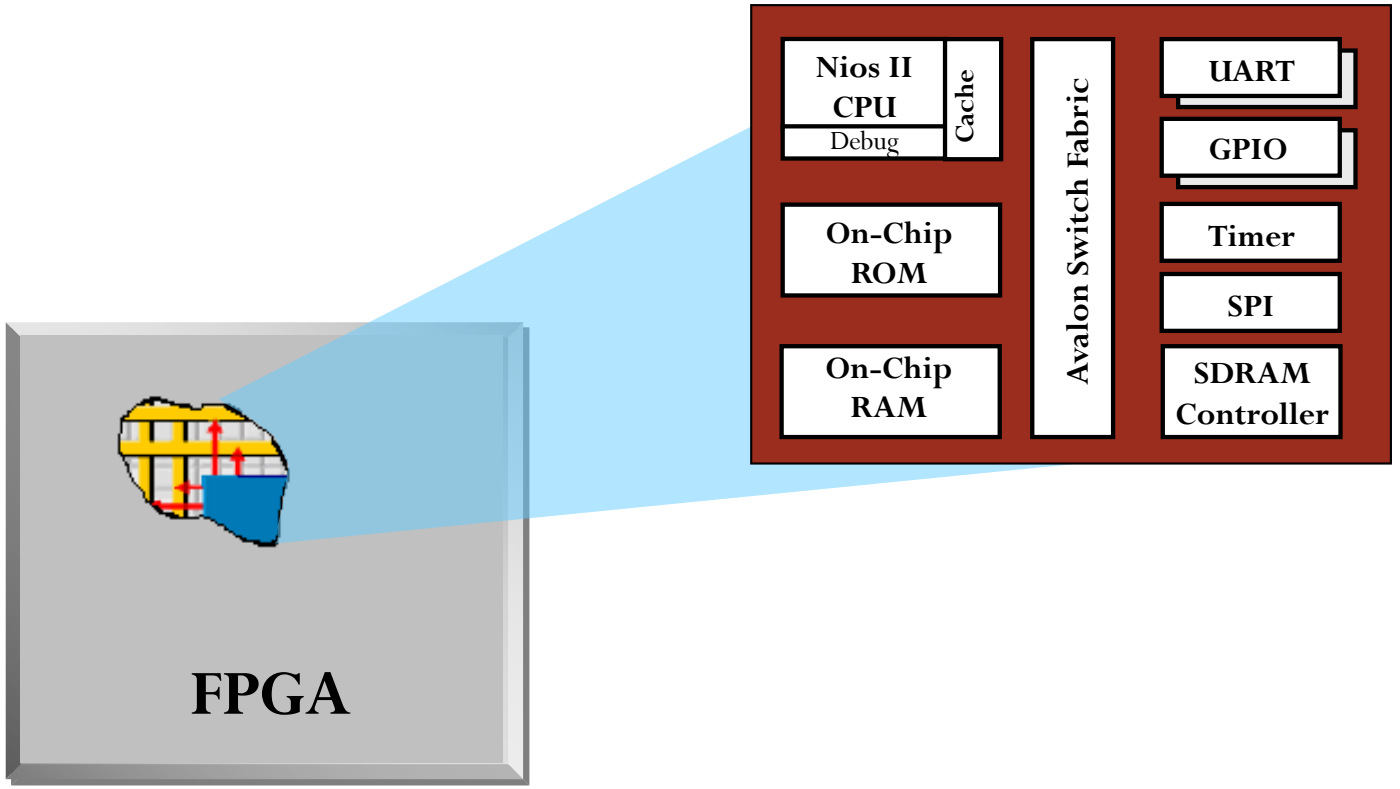
- **SOPC : System On Programmable Chip**
- **Objectifs:**
  - **Réduire les coûts, la consommation et la complexité des systèmes embarqués**
  - **Améliorer la flexibilité par l'ajoût de périphérique sur la même puce**
  - **Garder le même principe qu'un microcontrôleur (Processeur + périphérique au sein d'un même circuit)**

NIOS II : un processeur configurable

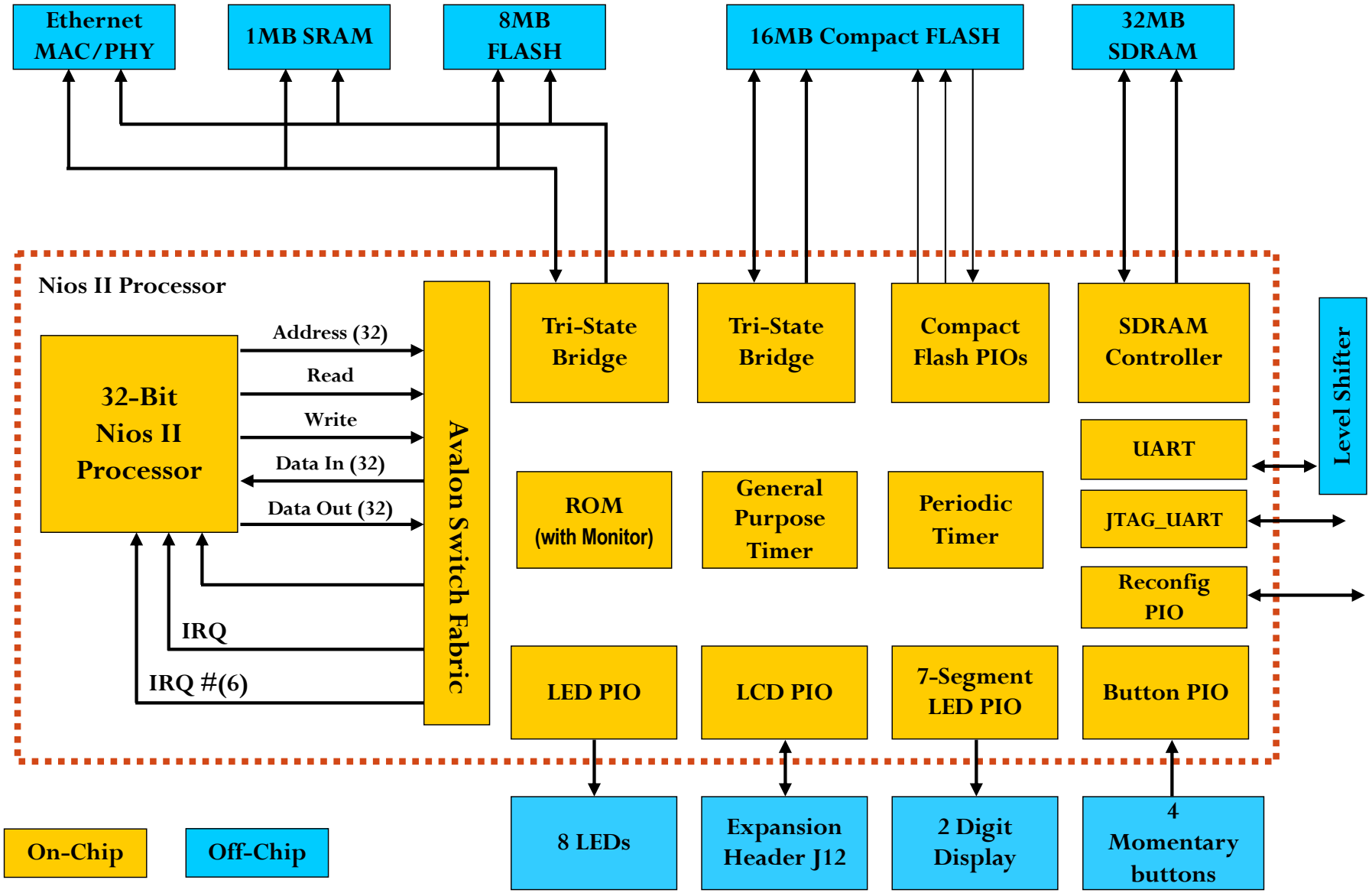
- **Un microcontrôleur possède une architecture figée**
- le processeur NIOS-II dispose d'une architecture configurable
  - on ajoute des éléments internes en fonction des besoins du processeur et de l'application.
  - ajout flexible de périphériques externes et du mapping mémoire.
- Les périphériques standards
  - Intel-Altera délivre un ensemble de périphériques (timers, interface série, gpio, SDRAM controllers,…)
- Les périphériques « custom » peuvent être développés et intégrés au processeur.
  - Ceci est recommandé pour les fonctions gourmandes en cycles CPU (-> hardware)

What is Nios II?

- Altera's Second Generation Soft-Core 32 Bit RISC Microprocessor
  - Nios II Plus All Peripherals Written In HDL
  - Can Be Targeted For All Altera FPGAs
  - Synthesis Using Quartus II Integrated Synthesis



Standard Reference Design Block Diagram



Nios II Versions

- **Nios II Processor Comes In Three ISA Compatible Versions**

– **FAST: Optimized for Speed**

– **STANDARD: Balanced for Speed and Size**

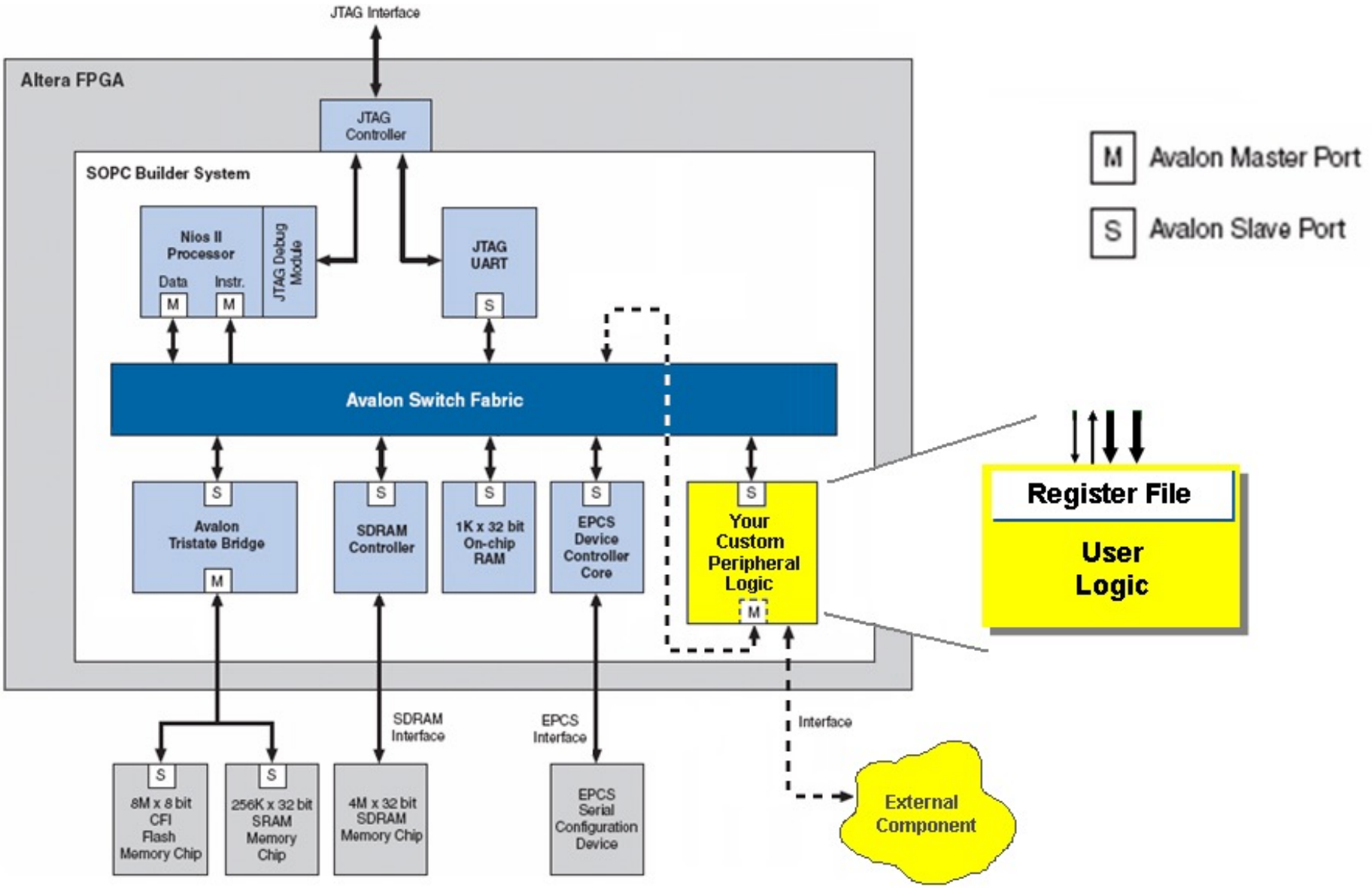
– **ECONOMY: Optimized for Size**

- **Software**
  - Le code compilé (binary) est compatible
  - pas de changement logiciel quand le CPU change

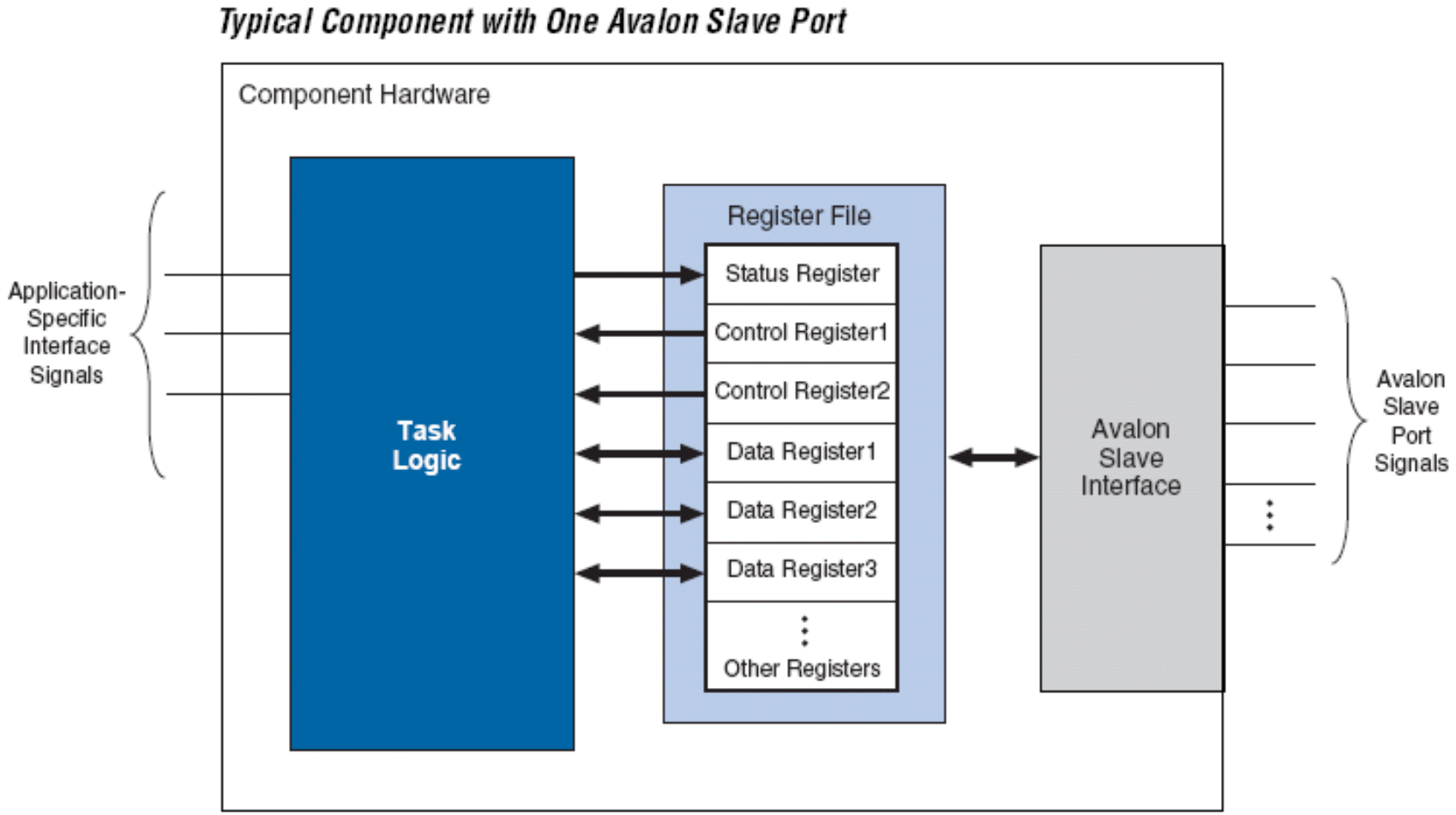
Binary Compatibility / Flexible Performance

	Nios II /f Fast	Nios II /s Standard	Nios II /e Economy
Pipeline	6 Stage	5 Stage	None
H/W Multiplier & Barrel Shifter	1 Cycle	3 Cycle	Emulated In Software
Branch Prediction	Dynamic	Static	None
Instruction Cache	Configurable	Configurable	None
Data Cache	Configurable	None	None
Logic Usage (Logic Elements)	1400 - 1800	1200 – 1400	600 – 700
Custom Instructions	Up to 256		

Custom Peripheral Integration

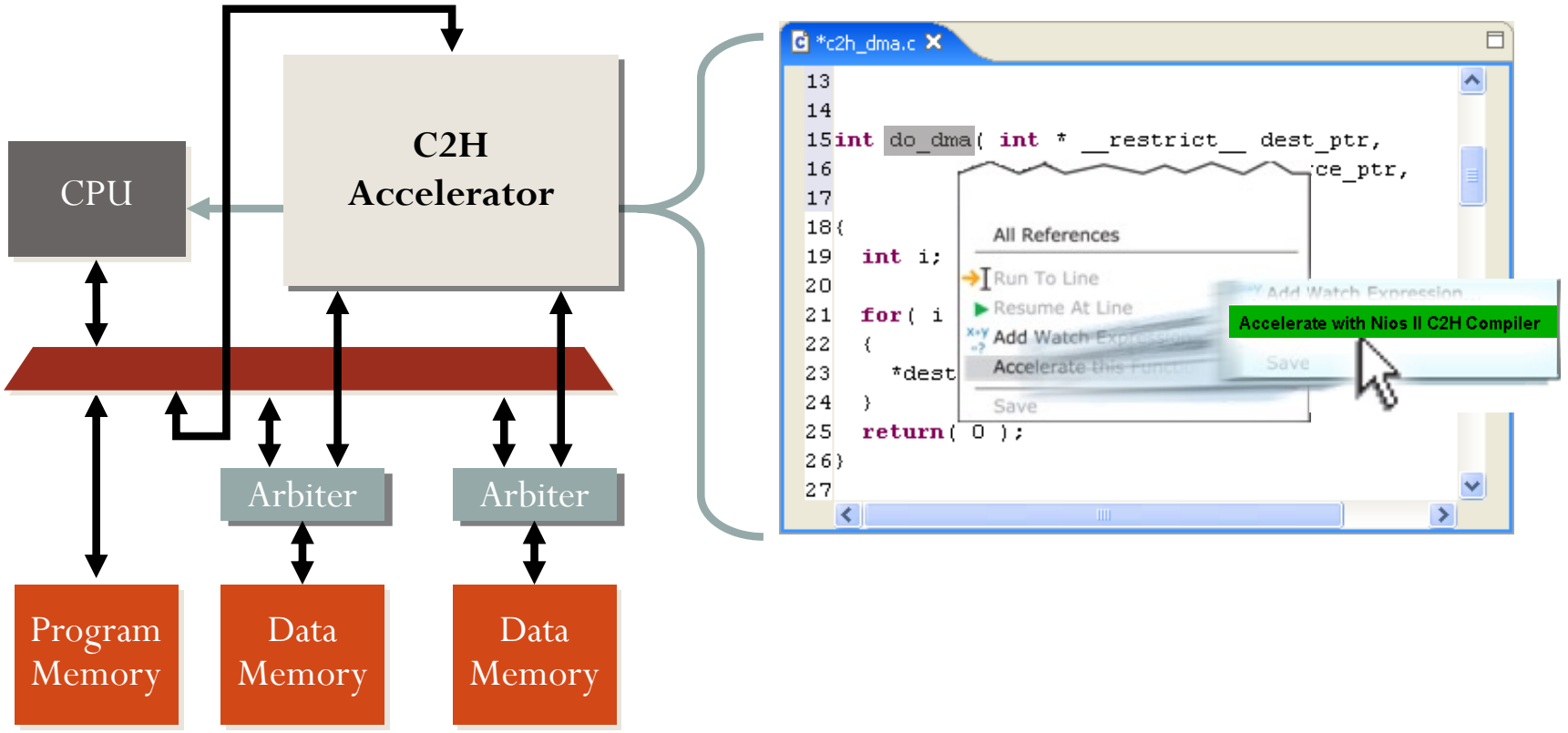


Custom Peripheral Structure



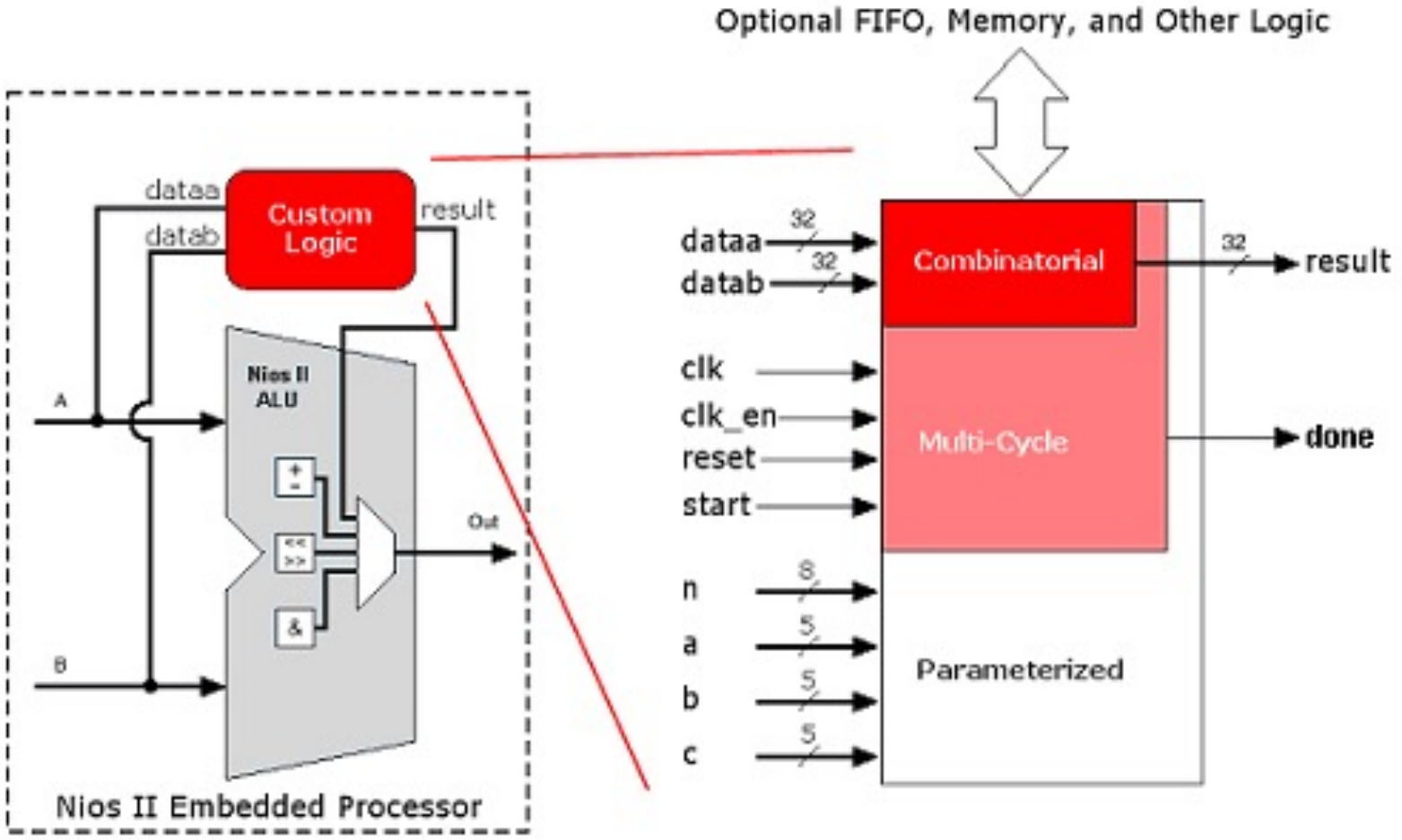
C2H Hardware Accelerator Block

- *Custom Peripheral* automatically generated from ANSI C function and integrated into Avalon Switch Fabric



Nios II Custom Instructions

- Augment Nios II Instruction Set.
- A custom instruction written in HDL or schematic format can easily be added in parallel with the Nios II ALU, providing extended capabilities to the processor.

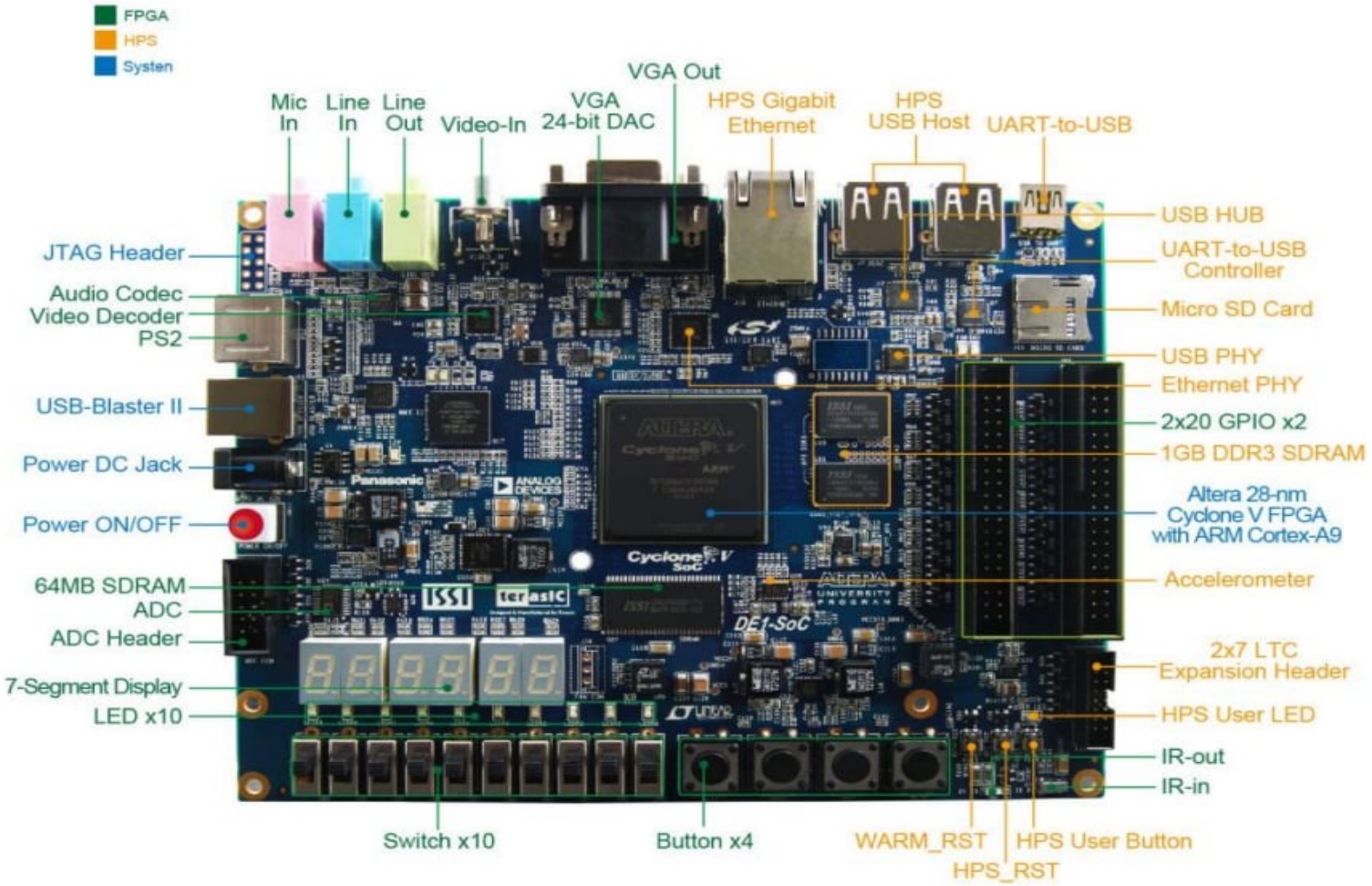


Génération du code NIOS

- «Platform Designer» d'Altera, outil graphique permettant la configuration système des périphériques et du processeur.
- Possibilité d'import/export de code VHDL de périphérique custom
- Génération automatique de code HDL pour l'instanciation du processeur dans le système

Travaux pratiques

Carte DE1-SoC



Avantages de la carte DE1-SoC

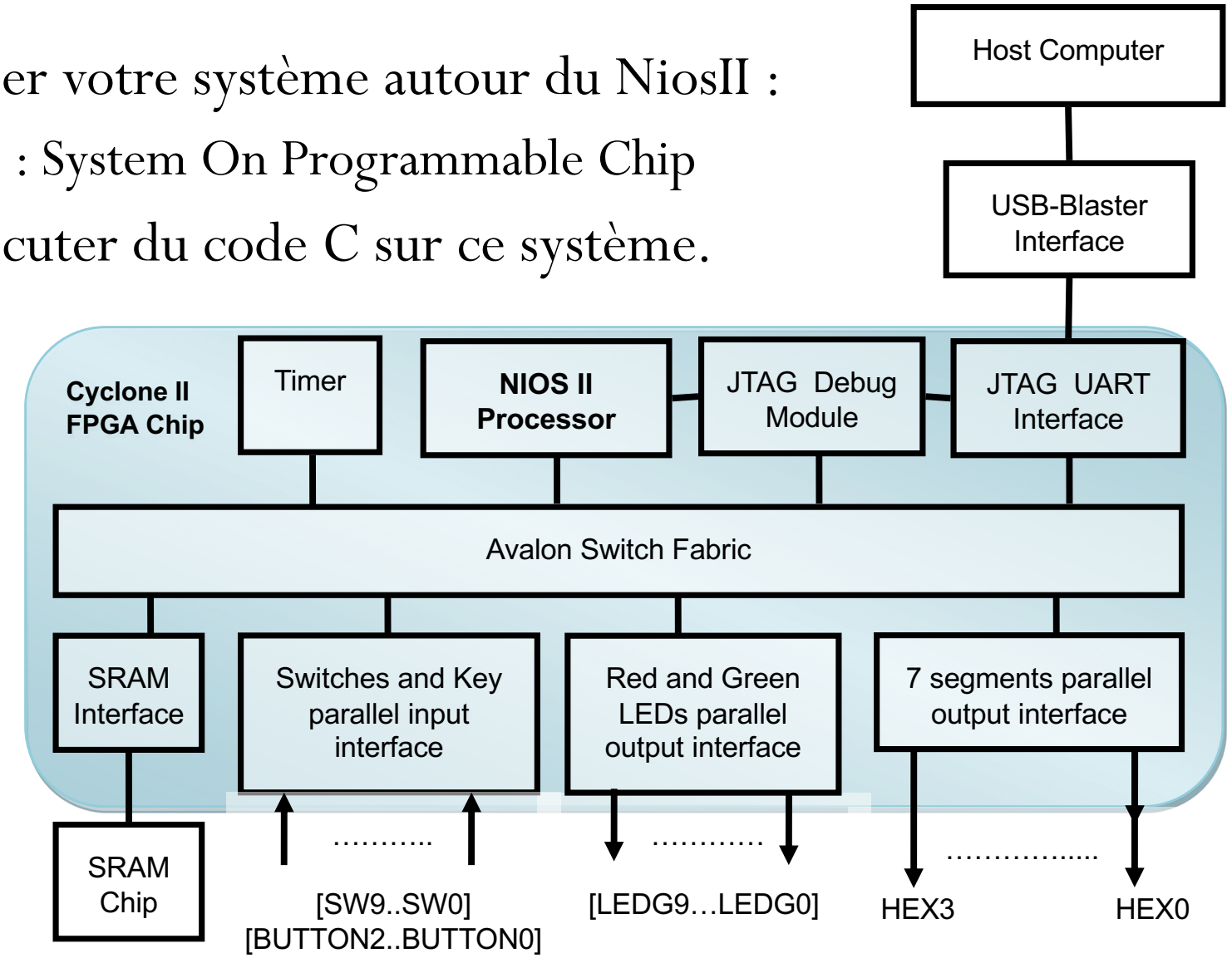
- Carte pédagogique très complète.
- Pas très chère (prix < 300 $)
- Différent espace mémoire : SRAM, SDRAM
- Beaucoup de périphériques : audio, vidéo, USB, Ethernet, carte SD, IrDA, RS232, etc…
- Très bien documenté avec beaucoup d'exemples de conception.
- Permet d'étudier la partie hardware d'un design autant que la partie software.
- Lien : https://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&No=836

Objectif du TP1

1. Créer votre système autour du NiosII :

   SOPC : System On Programmable Chip

2. Exécuter du code C sur ce système.



Travaux pratiques : TP1

- Dans cette exercice, nous utiliserons les outils d'Alterapour générer un environnement NIOS II et dérouler du Soft C.
- L'exercice se décompose en 5 parties:
  - Partie 1: Création de votre système NIOS II
  - Partie 2: Exécuter du code C sur ce système
  - Partie 3: Utiliser la couche d'abstraction HAL
  - Partie 4: Exercices
  - Partie 5: Création d'un jeu de PONG

TP1 : Partie 1 (Hardware)

- **Suivre les indications chapitre «partie 1» du document «DE1_SoC_TP1_v18.pdf»**
  - **Etape 1 à 3 :** Mis en place de votre environnement TP1
    - Dans le répertoire **…/tp_train/** copier le fichier «**tp1.zip**»
    - Dézipper le répertoire: «**tp1.zip**» clic droit et **Extractto here(extraire ici)**
  - **Etape 4 à 14** : constituer le système en ajoutant les périphériques
  - **Etape 15 à 19**: Finaliser les interconnections des périphériques
    - Connecter les périphériques aux horloge et reset système
    - Connecter les périphériques au bus données master du Nios
    - Affecter les adresses et les numéros d'interruptions aux périphériques
  - **Etape 20 à 25**: finaliser le FPGA
    - Génération du HDL dans PlatformDesigner
    - Instanciation et récupération du composant «**DE1_SC_Basic_Nios2_SoC.vhd**» dans le projet quartus
    - Synthèse et routage du FPGA
    - Affectation des pins avec l'outil pin planner de quartus
    - Routage du FPGA
  - **Etapes 26 à 29**: Chargement du binaire sur la carte

TP1 : Partie 2 (Software)

- **Suivre les indications chapitre «partie 2» dans le document «DE1_SoC_TP1_v18»**
  - **Etape 1 à 5**: Création d'un projet software (BSP + Code objet)
    - Un «BoardSupport Package» est un ensemble de logiciels utilisés pour démarrer et faire fonctionner un système embarqué.
    - Compilation du code
  - **Etape 6**: téléchargement du code sur la carte et test
    - Les «segments» de l'afficheur de droite évoluent
    - Les «segments» des autres afficheurs sont tous allumés
    - Les «LEDS» en face des «switchs» s'allument quand ceux-ci sont hauts
  - **Etape 7 à 13**: Tester le debugger
    - Positionner des points d'arrêts
    - Visualiser les valeurs de variables
    - Faire du pas a pas
  - **Etape 14 à 15**: Arrêt et retour à la fenêtre nios

Lire et écrire dans les périphériques du Nios II
Méthode utilisé dans le TP1 partie 2

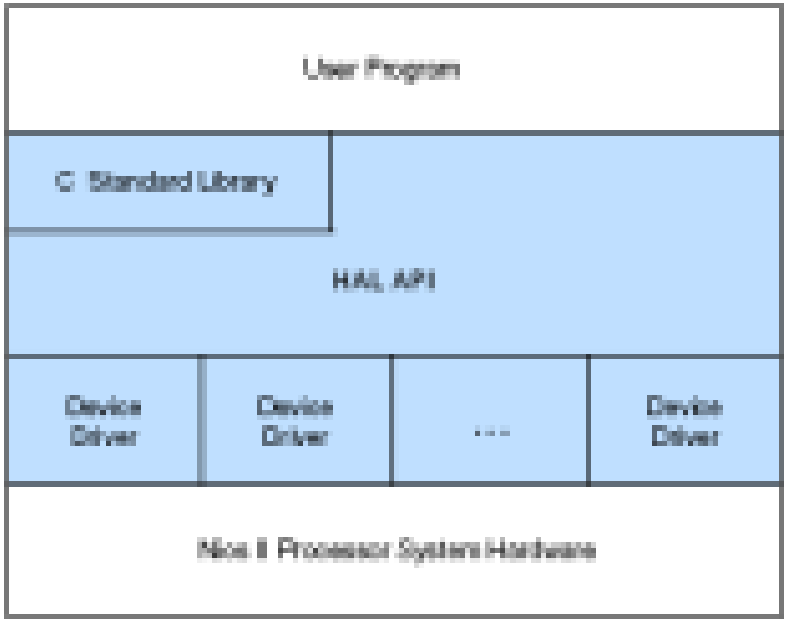
- Actuellement dans la partie 2 du TP1 on utilise une adresse définie par un pointeur auquel on a donné une valeur en dure.
  - **Pour lire** la valeur dans le périphérique «Switch» on utilise :
    - volatile int * SW_switch_ptr= (int *) 0x000810A0;// SW slider switch address
    - …
    - SW_value= *(SW_switch_ptr); // read the SW slider switch values
  - **Pour écrire** dans un registre «LEDR» :
    - volatile int * red_LED_ptr= (int *) 0x00081090;// red LED address
    - …
    - *(red_LED_ptr) = SW_value;// light up the red LEDs

Hardware Abstraction Layer (HAL)

- Cette couche logicielle fournie par «Altera» isole les applications «software» des modifications «Hardware»
- Les applications sont indépendantes car la HAL extrait des informations de systèmes tels que:
  - GPIO
  - UART, JTAG-UART, LCD display
  - Flash memory devices
  - Timer devices
  - DMA controller core
  - Ethernet mac/phycontroller
  - ….

Hardware Abstraction Layer (HAL)

- HAL library generation:
  1. «SOPC Builder» génère un système hardware
  2. NiosII IDE» génère une librairie HAL customisée pour correspondre au système hardware.
- Un changement de configuration Hardware est propagé automatiquement dans la configuration HAL des drivers.
- NIOS II est programmé en C



Lire et écrire dans les périphériques du nios II

Utilisation de la HAL

- Pour lire la valeur d'un registre dans le périphérique SWITCH:
  - **SW = IORD_ALT_UP_PARALLEL_PORT_DATA(SW_BASE);**
- Pour écrire dans un registre du périphérique LEDR :
  - **IOWR_ALT_UP_PARALLEL_PORT_DATA(LEDR_BASE, sw);**

- **LER_BASE** et **SW_BASE** correspondent aux adresses de base des périphériques LEDR (LEDS rouges) et SWITCH (interrupteurs).
  - Voir le fichier **system.h** dans :
    - **NiosII_training_project_bsp -> Debug -> System Description – System.h**

Lire et écrire dans les périphériques du nios II

Utilisation de la HAL

- Lorsque un périphérique comporte plusieurs registres, il faut ajouter un offset à l'adresse de BASE afin d'écrire dans les différents registres du périphérique.
- Par exemple :

```
IOWR_ALT_UP_PARALLEL_PORT_DATA(SEVEN_SEG_BASE + 0, SevenSeg[0] );
IOWR_ALT_UP_PARALLEL_PORT_DATA(SEVEN_SEG_BASE + 4, SevenSeg[1] );
IOWR_ALT_UP_PARALLEL_PORT_DATA(SEVEN_SEG_BASE + 8, SevenSeg[2] );
```

TP1 : Partie 3 (utilisation drivers de la HAL)

- **Suivre les indications chapitre «partie 3» dans le document «DE1_SoC_TP1_v18»**
  - Ouvrir Quartus et le projet «DE1_SoC_ADC.qpf»
    - Charger le binaire sur la carte d'évaluation
  - Lancer PlatformDesignerà partir de quartus
  - Lancer eclipseà partir de PlatformDesigner
    - Ouvrir et modifier le code afin d'utiliser la HAL d'altera
    - Compiler le code
      - le répertoire **niosII_training_project,** ensuite avec le clic droit sélectionnez **BuildProject**
    - Télécharger le programme sur la cible
  - le répertoire **niosII_training_project,** avec le clic droit sélectionnez **Run As -> NiosII Hardware**

TP1: Partie 4 (Exercices à effectuer)

- **Suivre les indications chapitre «partie 4» dans le document «DE1_SoC_TP1_v18»**
  - Exercice 1: Utilisation en C de la fonction «Printf»
    - Réalisation d'un chenillard sur les LED rouge
    - Affichage en continu sur la console niosde la valeur des switch SW
  - Exercice 2: Utilisation de la fonction «scanf»
    - Commande d'affichage des LED
    - Commande d'affichage d'une valeur entrée au clavier sur l'un des afficheurs 7 segments
  - Exercice 3: Utilisation des poussoirs KEY pour contrôler une vitesse
    - Réaliser une chenillard qui défile de droite a gauche
    - Contrôler la vitesse de défilement grâce aux poussoirs KEY(1) et KEY(2). Pour cela, remplacer la valeur de tempo de 200 (dernière ligne du While) par une variable «speed» initialisée a 200
  - Exercice 4: Affichage de la vitesse sur les afficheurs 7 segments
    - Créer un compteur BCD représentant l'incrément ou le décrément de la vitesse
    - Afficher en permanence sa valeur sur les afficheurs 7 segments

TP1: Partie 5 (Jeu de PONG)

- **Suivre les indications chapitre «partie 5» dans le document «DE1_SoC_TP1_v18 »**
- –Exercice 1: Partie de Pong
- •Réalisation d'un chenillard sur les LED rouge avec changement de sens si KEY correspondante est appuyée au bon moment.
- •Affichage en continu sur la console niosde la valeur des switch SW
- –Exercice 2: Accélération de la vitesse de défilement.
- •La vitesse est réinitialisée grâce a la variable «speed» à 100
- •Incrément de la vitesse tant qu'aucun des joueurs n'a perdu
- –Exercice 3: Affichage de la vitesse sur les 7 segments
- •Réaliser un incrémenter / décrémenter BCD
- •Contrôler la vitesse de défilement grâce aux poussoirs KEY(1) et KEY(2)
- •Créer un compteur BCD représentant l'incrément ou le décrément de la vitesse