Projets Industriels 2009 - 2010

PI16 – TRIXELL – Soutenance Finale 28/06/2010

Portage d'application médicale sous Linux et interface Wifi

Introduction

Plan de la présentation

- 1 – Objectifs
- 2 – Spécification
- 3 – Développement
- 4 – Déroulement du projet et perspectives

Objectifs

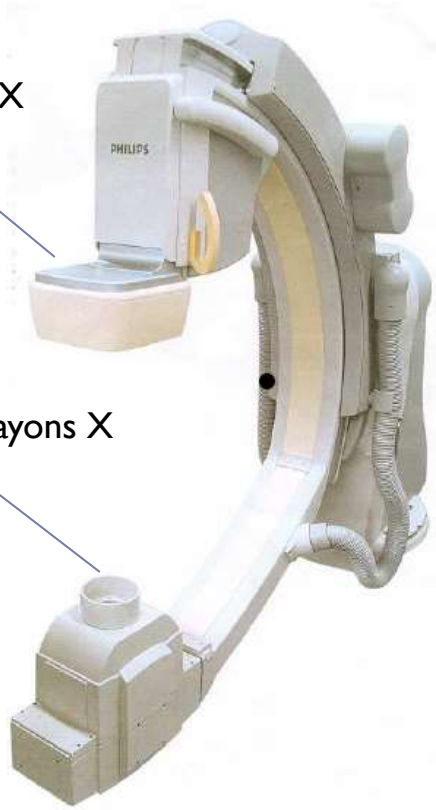
Présentation de TRIXELL

- Leader mondial des capteurs médicaux numériques à rayons X
- Joint venture
  - THALES
  - SIEMENS
  - PHILIPS
- Ressources humaines
  - Plus de 400 employés
- Siège à Moirans
- CA 175 M€ en 2008
- Stratégie d'innovation

Système radiographique

Capteur rayons X

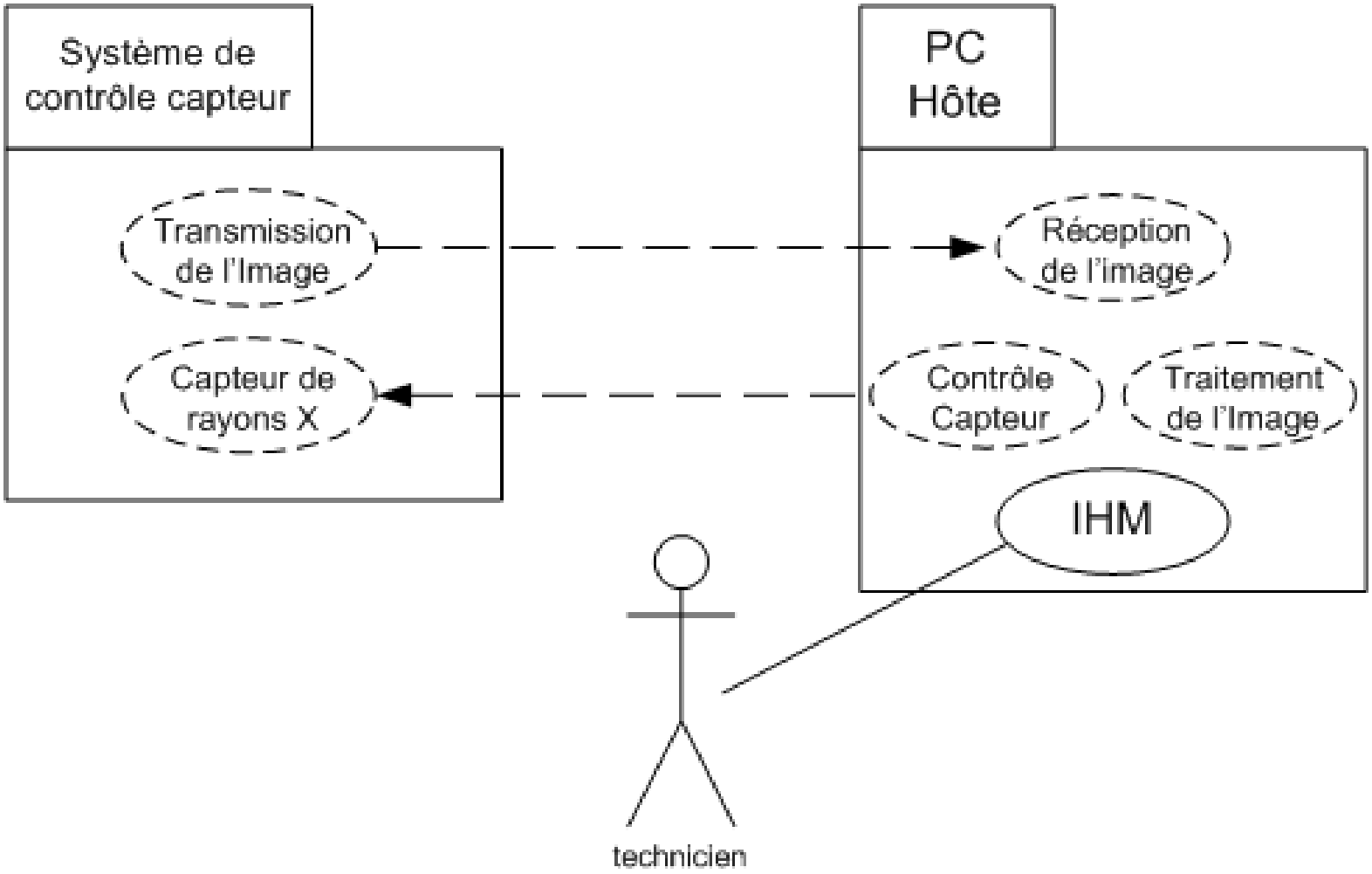
Générateur de rayons X



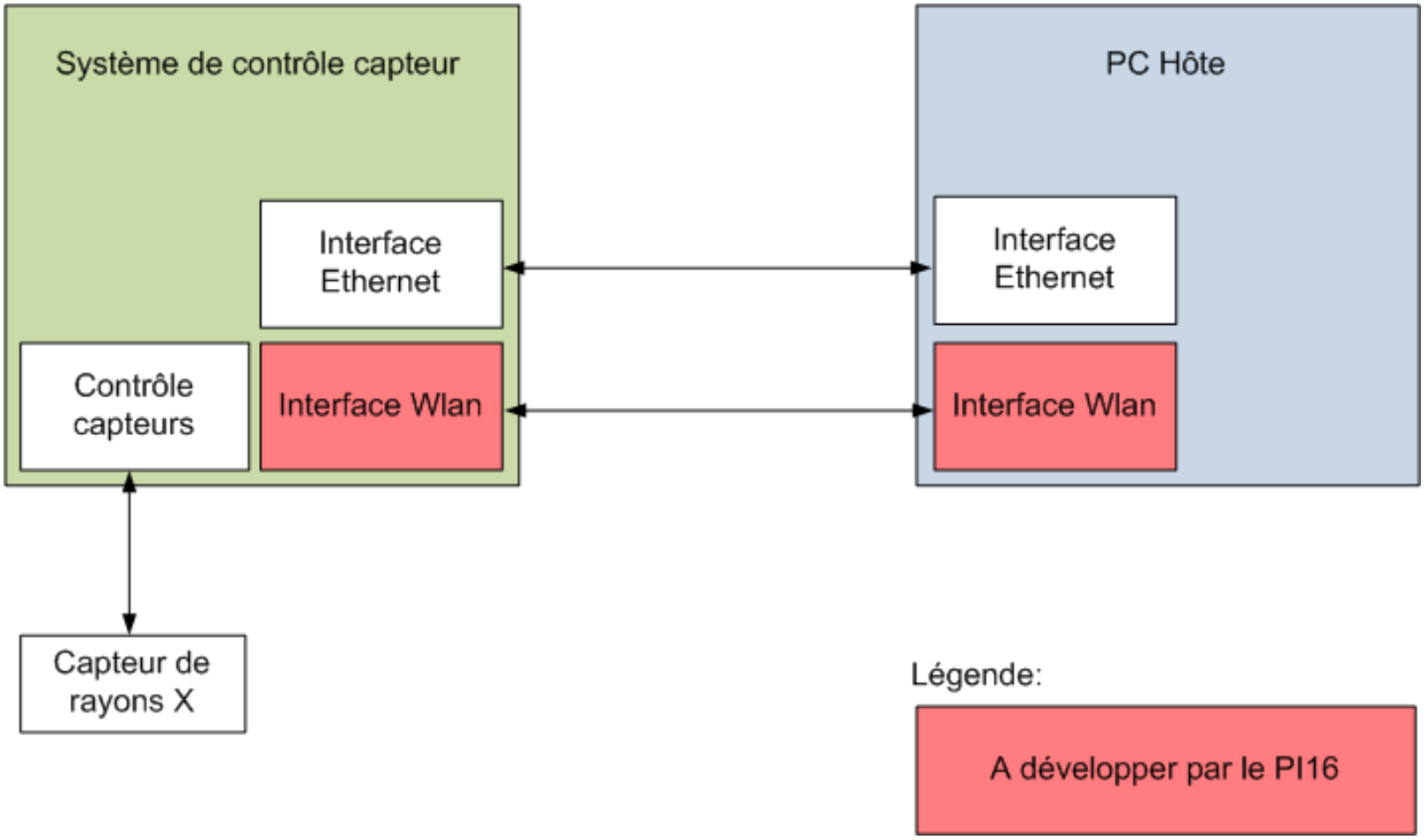
C-arm

Pourquoi le « sans fil » ?

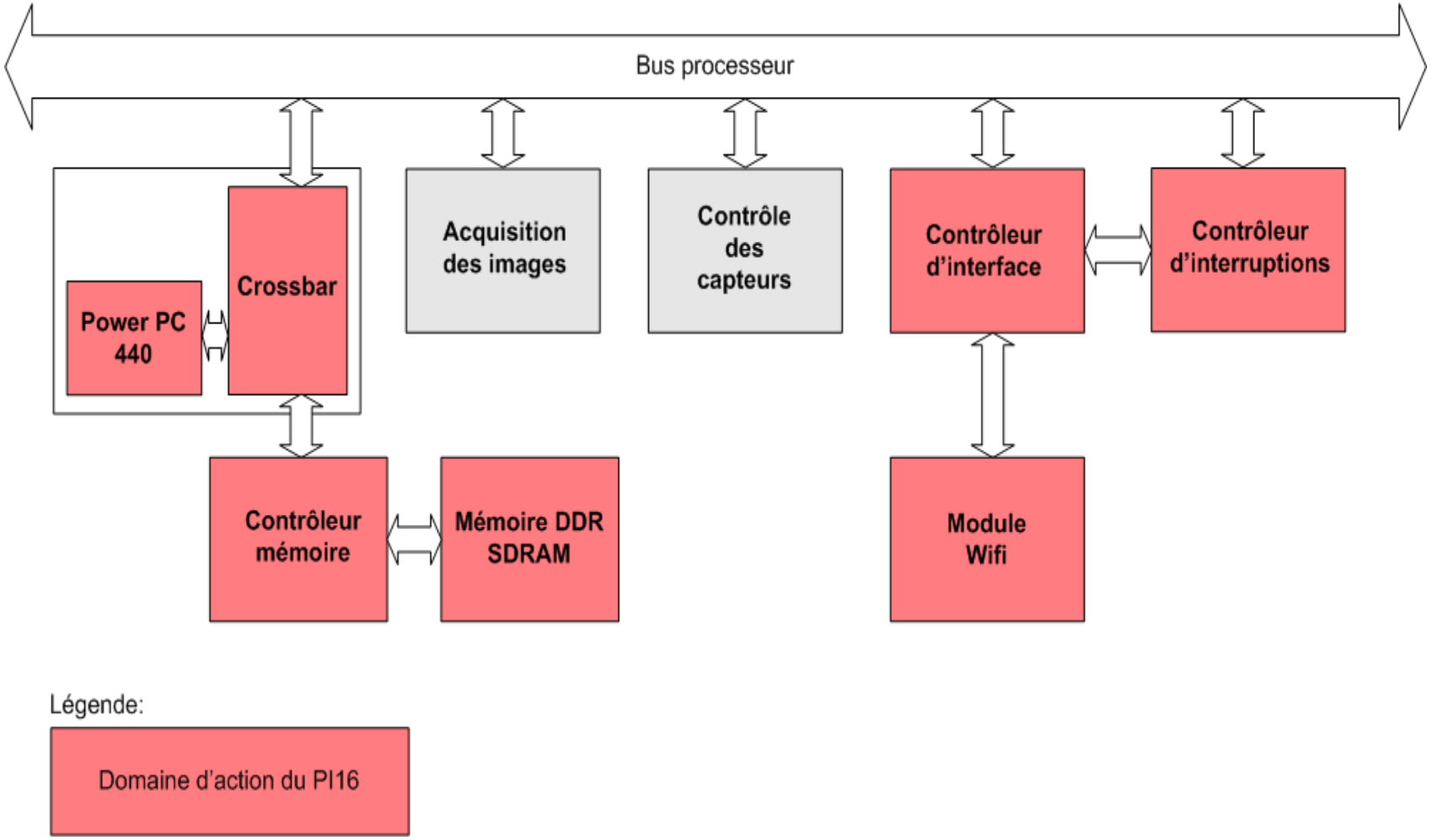
Fonctionnement du dispositif



Architecture du capteur



1 – Architecture du système de contrôle capteur



Les exigences

- Une interface sans-fil.
- Transmettre une image issue du capteur en moins de 5 secondes.
- Solution pérenne.
- Utiliser un OS lowcost.

Les attentes du projet

- 1 – Analyse des différentes interfaces des différents modules Wifi du commerce

- 2 – Intégration matérielle sur la plateforme Trixell.

- 3 – Intégration logicielle.

Livrables

- Etat de l'art des modules Wifi compatibles avec les FPGA Xilinx.
- Documents de veille technologique présentant les avantages et les inconvénients des solutions envisagées.
- Démonstrateurs implémentant les solutions retenues.
- Dossier technique et manuel d'utilisation des prototypes.
- Cahier de recettes des tests de performance et de stabilité des différentes solutions Wifi

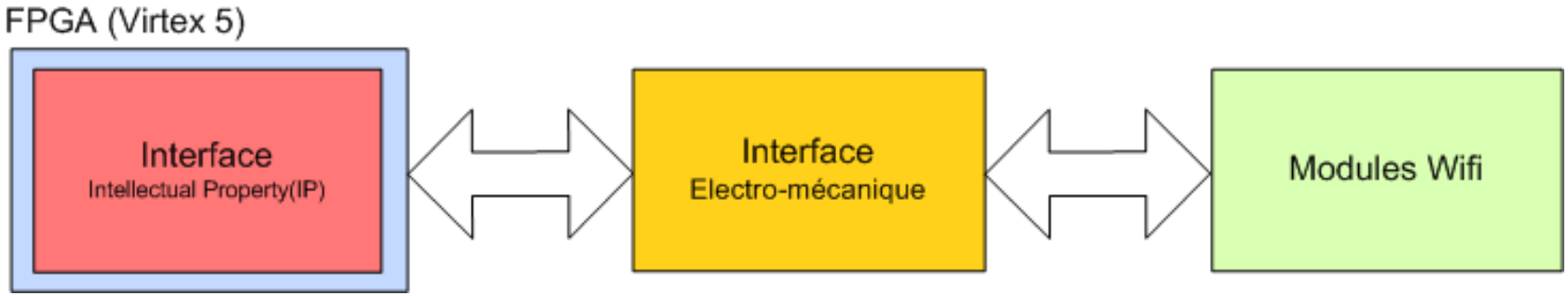
Découpage du projet

- But : Ajouter la fonctionnalité Wifi
  - Etablir les critères de choix pour l'interface
    - Choisir l'interface
  - Etablir les critères de choix pour le module
    - Choisir le module
  - Intégration matérielle l'interface et le module à la plateforme TRIXELL
    - Création / Achats des éventuelles cartes d'adaptation
  - Intégration de l'interface au niveau du FPGA
    - Développement de Wrapper si nécessaire
  - Mise en place des composants logiciels
  - Test de la fonctionnalité

Spécification

Comment intégrer le Wifi ?

- Intégration
  - IP
  - Interface matérielle
  - Modules Wifi

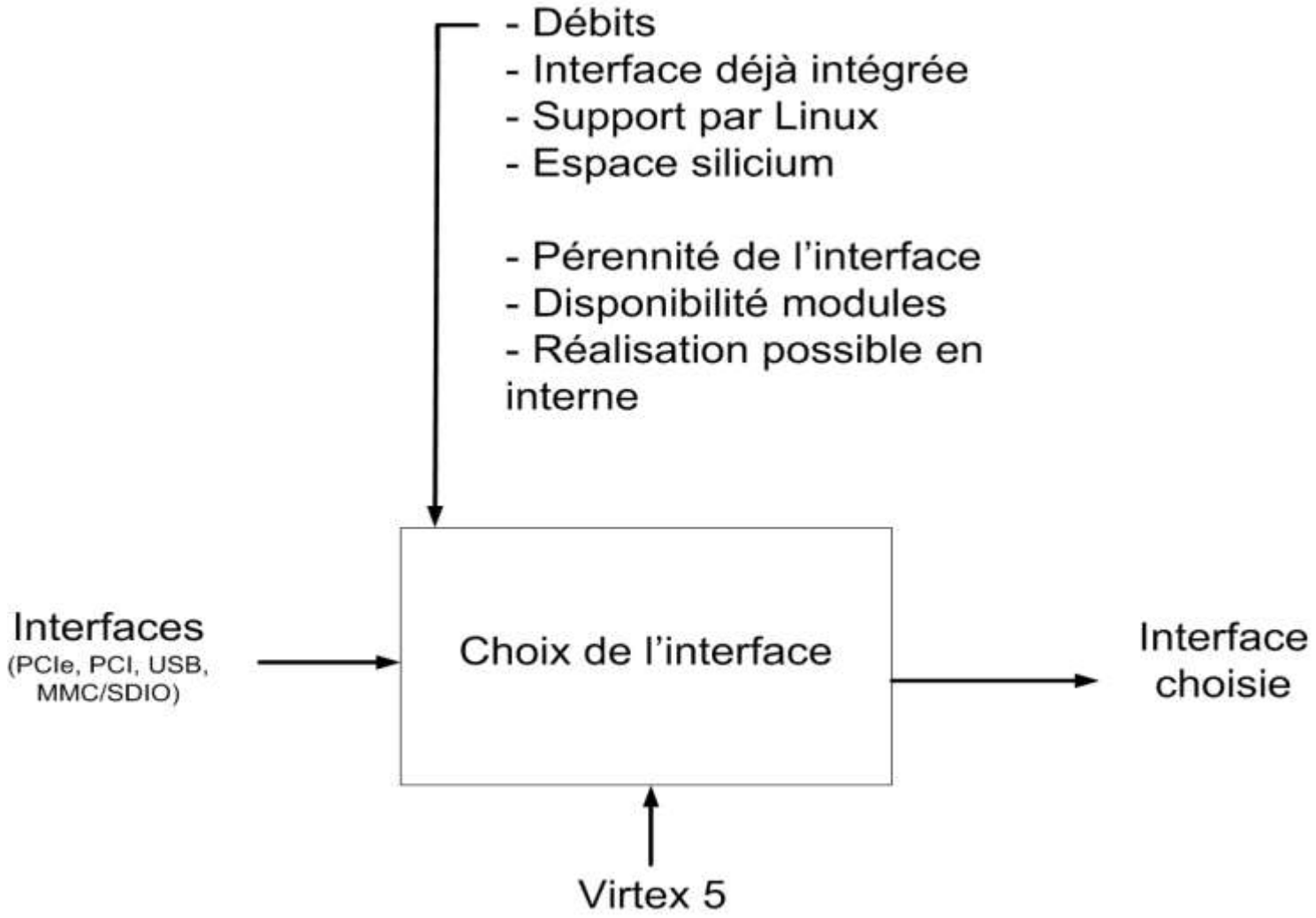


Carte de développement

- Carte de développement ML507 Xilinx
- FPGA Virtex5 (XC5VFX70T)
- Entrées sorties
  - Ethernet Gigabit
  - Lane PCIe 1x
- Ordre de prix
  - FPGA : 1000$
  - ML507 : 1500$

Xilinx ML507

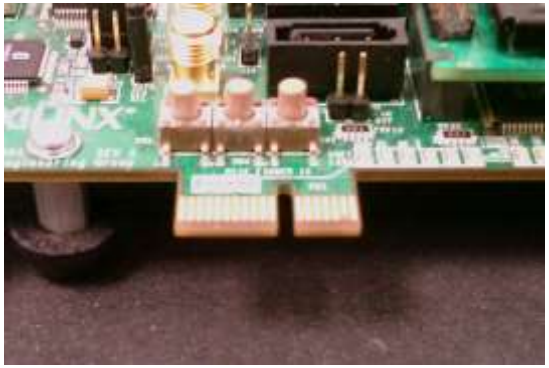
Critères de choix des interfaces



Interfaces retenues

- Etude préalable: PCIe, PCI, USB 2.0, SDIO/MMC

- Interface
  - USB 2.0
  - PCIe



Critères de choix des modules Wifi

- Chipset intégrant 802.11a et 802.11g.
- Version industrielle sans packaging
- Possibilité de connecter une antenne
- Le débit
- Support du cryptage
- Pérennité et simplicité d'approvisionnement
- Support des drivers Linux

Modules retenus

- Modules Wifi
  - USB 2.0
    - Unex DNUR-83

    - Unex DNUA-81

  - PCIe
    - Unex DNXA-92

Choix des IPs

- USB : Tableau comparatif des IP disponibles
  - Architecture
  - Coûts
  - Encombrement (Slices, Flipflop)

- PCIe : Choix restreint
  - Partenariat Xilinx : PLDA / Northwest Logic
  - Langage utilisé

- Discussion avec John Willams (PDG de PETALOGIX) sur l'intégration sous Linux des drivers

Développement

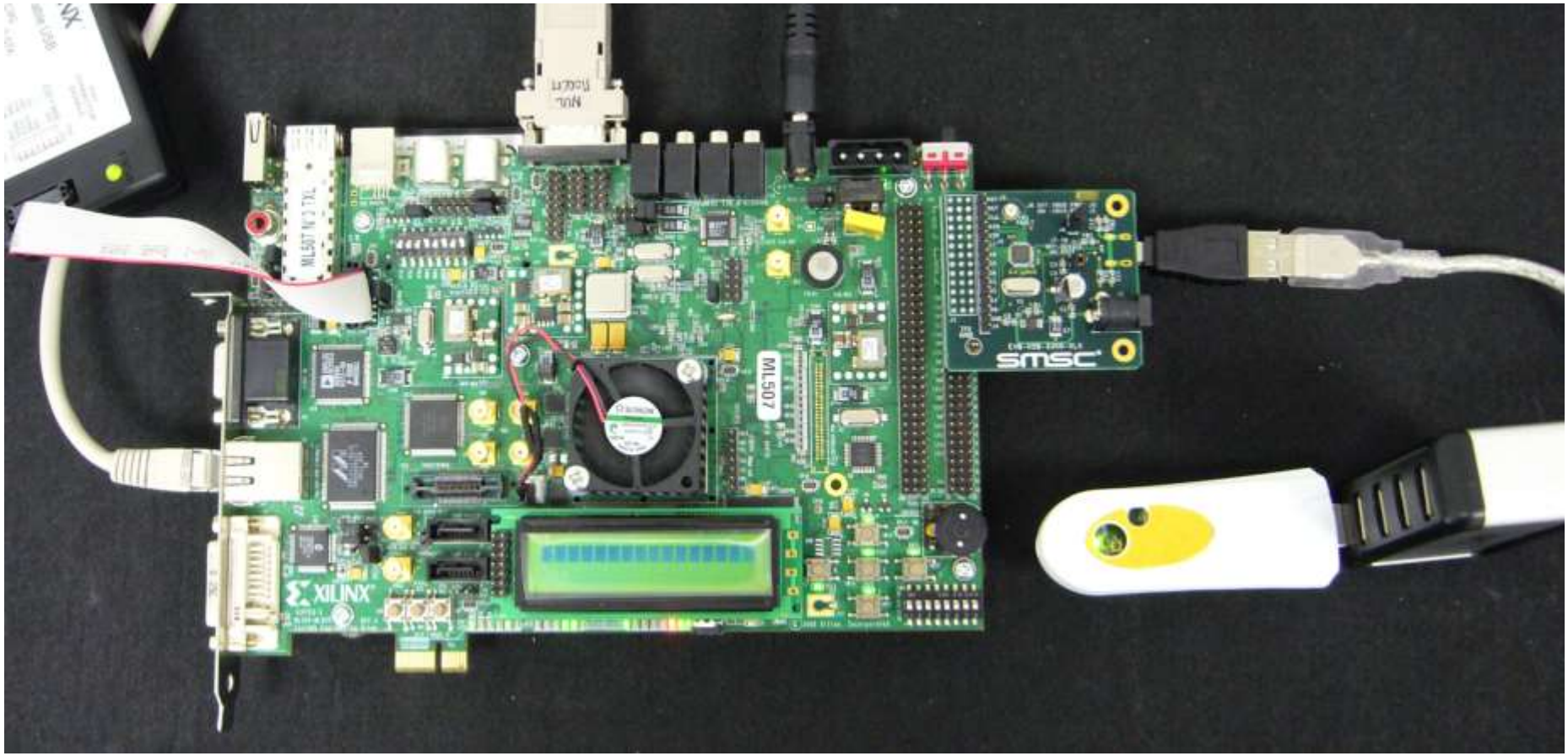
Chargement de l'OS

- Travail effectué
  - Choix d'un noyau permettant de piloter une interface WIFI
    - Etude comparative des noyaux Li
  - Compilation du noyau Linux
  - Faire tourner Linux
    - Chargement de l via JTAG et CompactFlash
  - Chargement du système de fichiers Linux
    - Via NFS
    - Via la CompactFlash

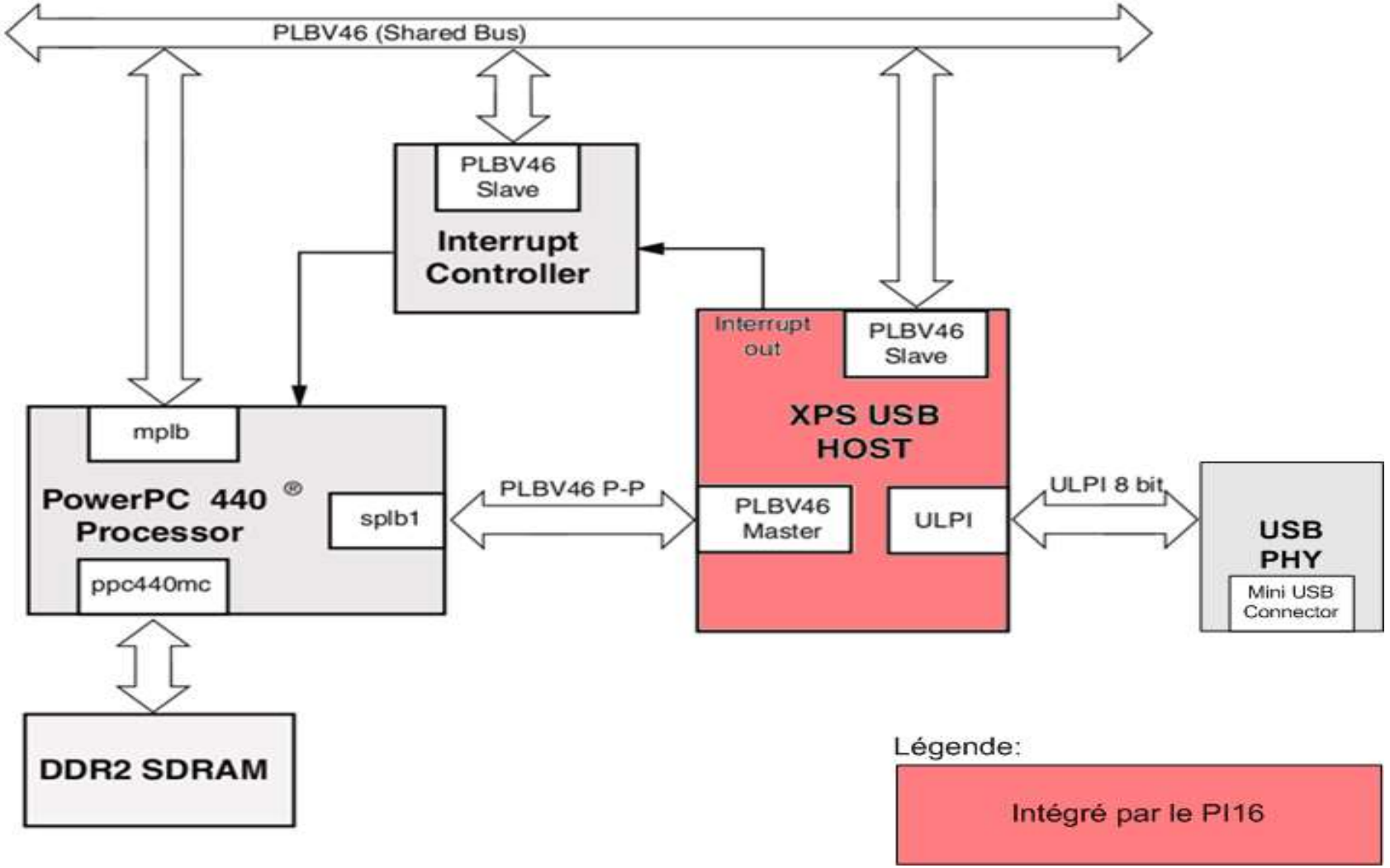
USB - Intégration des modules USB

- I – Intégration au niveau du FPGA
- 2 – Intégration logicielle
- 3 – Matériel nécessaire
- 4 – Résultats obtenus
- 5 – Problèmes rencontrés
- 6 – Bilan

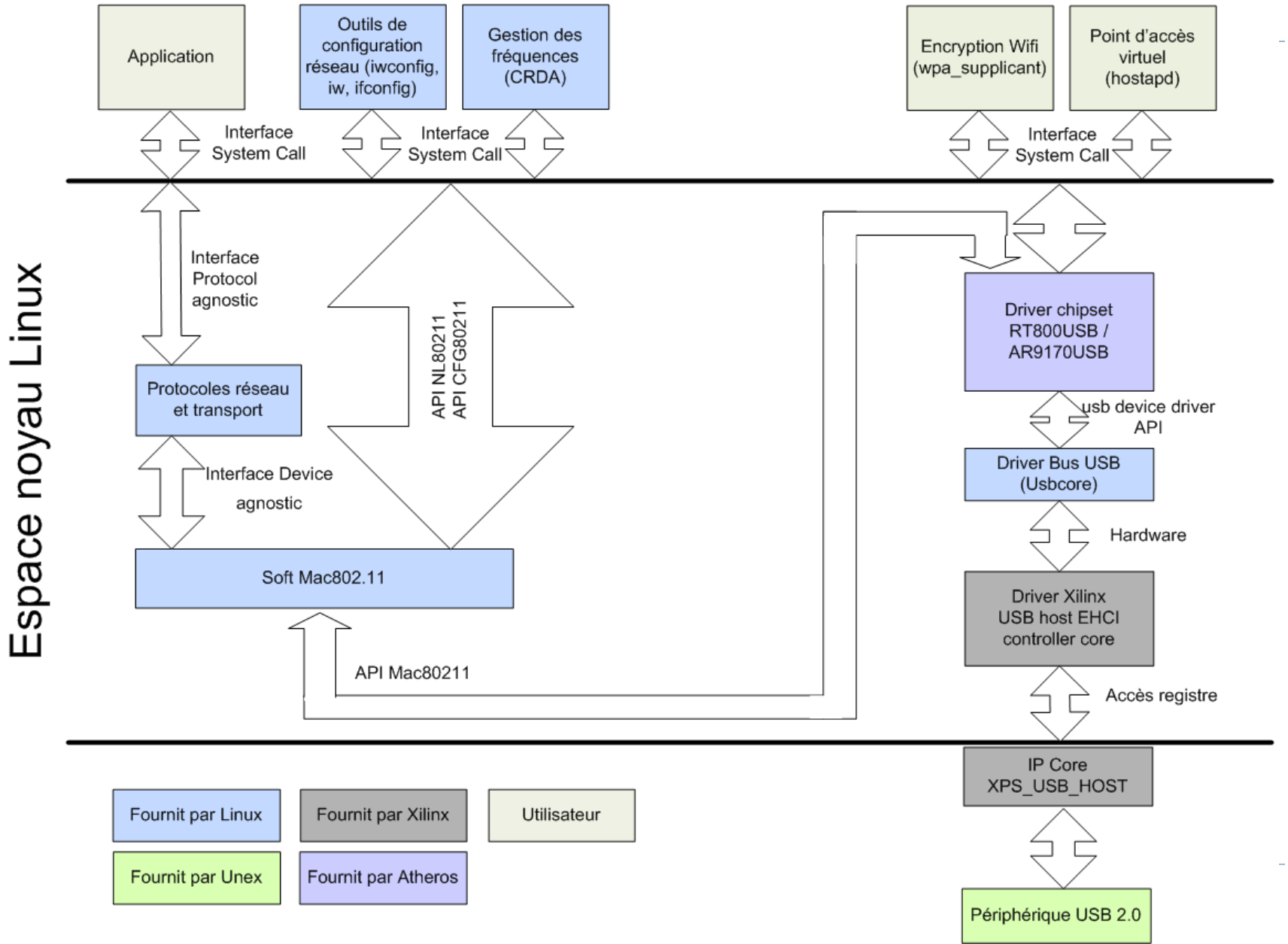
USB - Plateforme



USB – Intégration au SOC FPGA

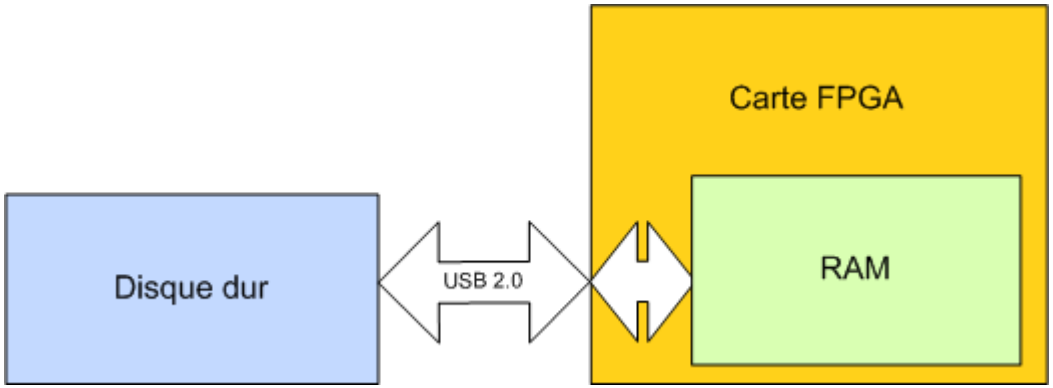


USB – Architecture logicielle



USB – Test de l'interface

- Evaluation de l'interface USB
  - Test de débit avec un périphérique de stockage de masse
    - Valeurs moyenne 20 Mo/s
    - Varie entre 18 Mo/s et 23 Mo/s
  - Transfert de donné vers un disque dur puis vérification de leur intégrité



USB – Tests des modules Wifi

- Transfert d'une l'image (3000x3000x16bits) (18 Mo) depuis le prototype vers un PC

  - Liaison point à point
  - Puissance d'émission limitée à 10mW
  - Vérification des données après le transfert
  - Différents protocoles (FTP, UFTP, TFTP)
- Qualification du lien (Netperf)
- Ecriture de scripts
  - Tests d'endurance
  - Mise en place facilitée

USB – Résultats - Transfert d'image via FTP

	Cannal 11 (2462 MHz)			Cannal 46 (5230 MHz)		
Débit en Mo/sec	**Antenne FOUET noire**	**Antenne PATCH**	**Antenne FOUET blanche**	**Antenne FOUET noire**	**Antenne PATCH**	**Antenne FOUET blanche**
Débit maximum	1,15	1,11	1,02	0,45	0,42	0,45
Débit minimal	0,88	0,42	0,81	0,35	0,40	0,36
Débit moyen	1,05	0,81	0,97	0,39	0,41	0,43

USB – Résultats

- Variation importantes en fonction de la bande de fréquence
- Bande 2.4GHz
  - Exemple : avec un module commercial (D-Link)
  - 1,73 Mo/s sous Linux
  - 0,50 Mo/s sous Windows
  - Débits suffisants
- Bande 5GHz
  - Débits non stables
    - Environnement de test inadapté
    - Liaison point à point expérimentale
  - Débits insuffisants

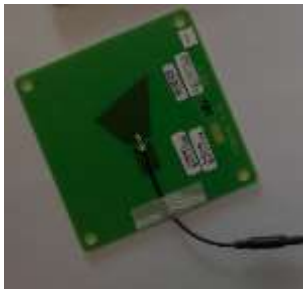
USB – Problèmes rencontrés

- Alimentation électrique des modules USB
- Performances des antennes
- Limitations du mode point à point (Ad Hoc)
  - Problèmes d'interconnexion entre Linux et Windows
  - Utilisation de la bande 5Ghz non implémentée dans la plus-part des drivers
  - WPA2 expérimental
- Linux
  - Documentation insuffisante
  - Support fournit par la communauté

USB – Caractérisation des antennes

- Transmission sans-fil : Besoin d'antennes adaptées
  - Utilisation des ressources du RFTLab

- Exemples
  - 1) Antenne Patch
    - 2,53GHz (-32,05dB) / 5,03GHz (-23,40dB)



  - 2) Antenne FOUET noire
    - 2,40GHz (-27,57dB)

USB – Bilan technique

- Maquette USB fonctionnelle et autonome.
- Débits de l'interface USB comparables.
- Taux de transfert des modules Wifi.
  - Résultats corrects dans le prototype de TRIXELL dans une des configurations de test.
- Possibilité d'intégration dans le prototype de TRIXELL.

USB – Bilan

- Choix de l'IP 03/02/2010 ✓
- Acquisition de la carte SMSC. 17/03/2010 ✓
- Intégration de l'IP dans le FPGA. 23/03/2010 ✓
- Chargement du driver de l'IP dans le noyau. 24/03/2010 ✓
- Activation du support des périphériques de stockage de masse. 26/03/2010 ✓
- Essais de lecture/écriture sur une clé USB. 02/04/2010 ✓
- Activation du driver Wifi et de la Soft Mac802.11. 07/04/2010 ✓
- Tests de performance et stabilité. 26/04/2010 ✓
- Portage sur CompactFlash 26/05/2010 ✓
- Vérification de la documentation et des procédures 01/06/2010 ✓
- Démonstration chez TRIXELL 10/06/2010 ✓
- Finalisation du transfert technologique 23/06/2010 ✓

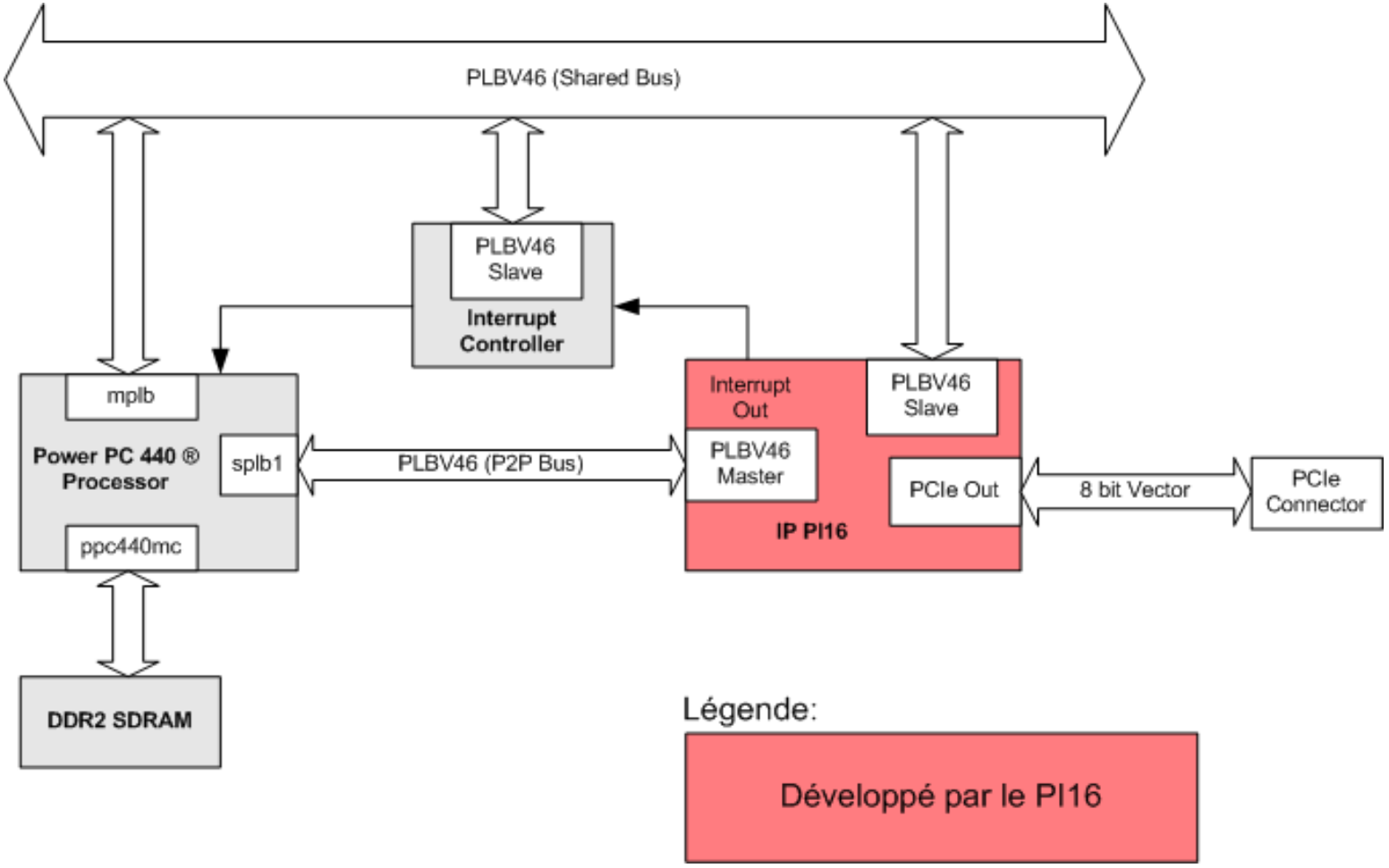
USB – Tutoriels

- Documentation technique
  - USB_MANTEC_01: Mise en place du hardware et programmation du FPGA
  - USB_MANTEC_02 : Mise en place des composants logiciels (Linux,drivers Wifi, logiciels de test)
  - USB_MANTEC_03 : Installation du logiciel sur carte CompactFlash pour rendre la plateforme autonome
- Manuel utilisateur
  - USB_MANUSER : Mise en route de la maquette et opérations de base
- Fiche de tests
  - Description de l'environnement de test
  - Fonctionnement des scripts de test

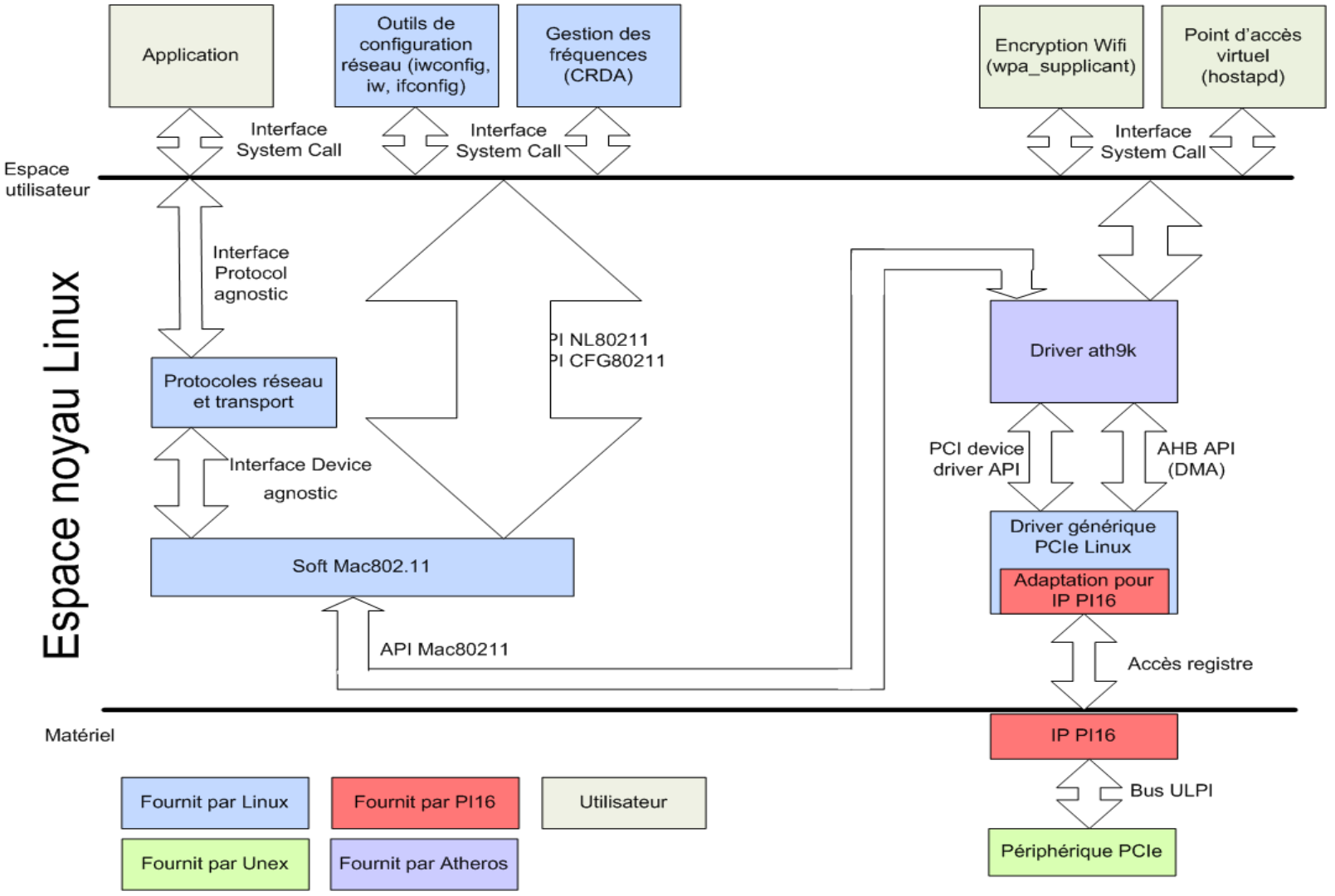
PCIe – Intégration des modules PCIe

- 1 – Intégration au niveau du FPGA
- 2 – Intégration logicielle
- 3 – Matériel nécessaire
- 4 – Résultats obtenus
- 5 – Problèmes rencontrés
- 6 – Bilan

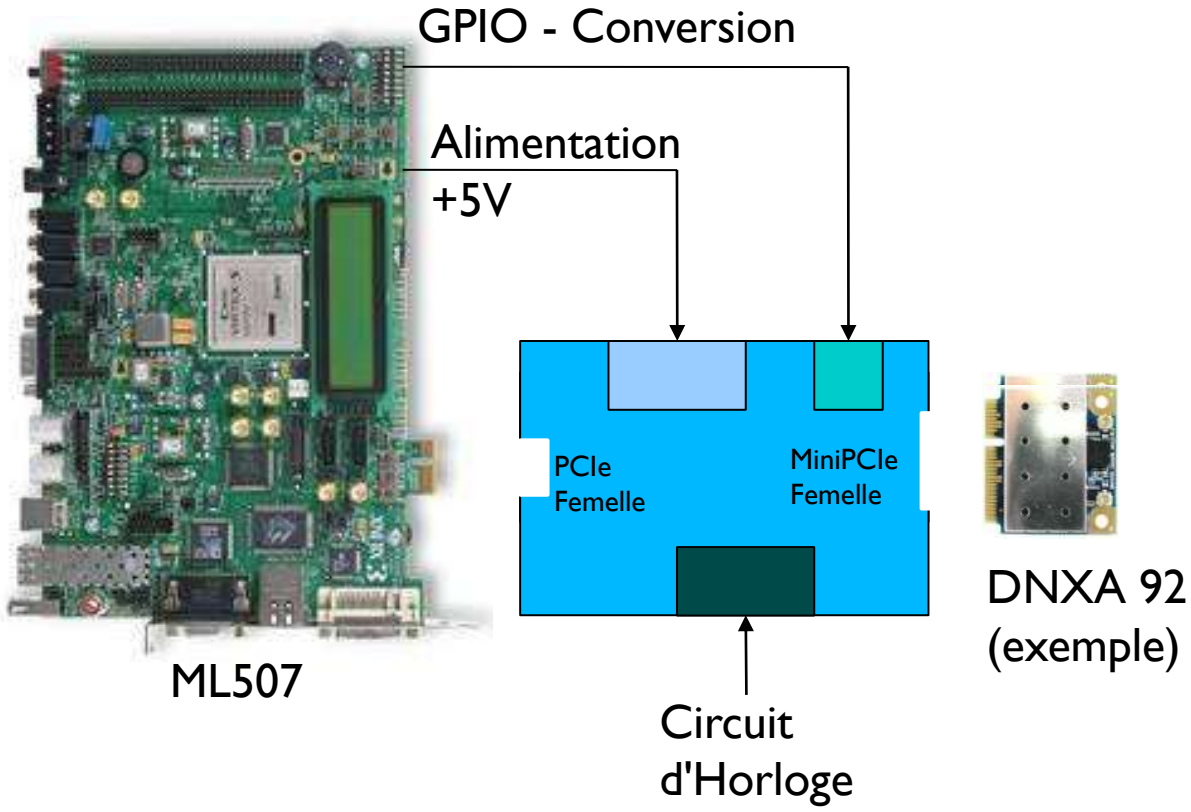
PCIe – Intégration au SOC FPGA



PCIe – Architecture logicielle

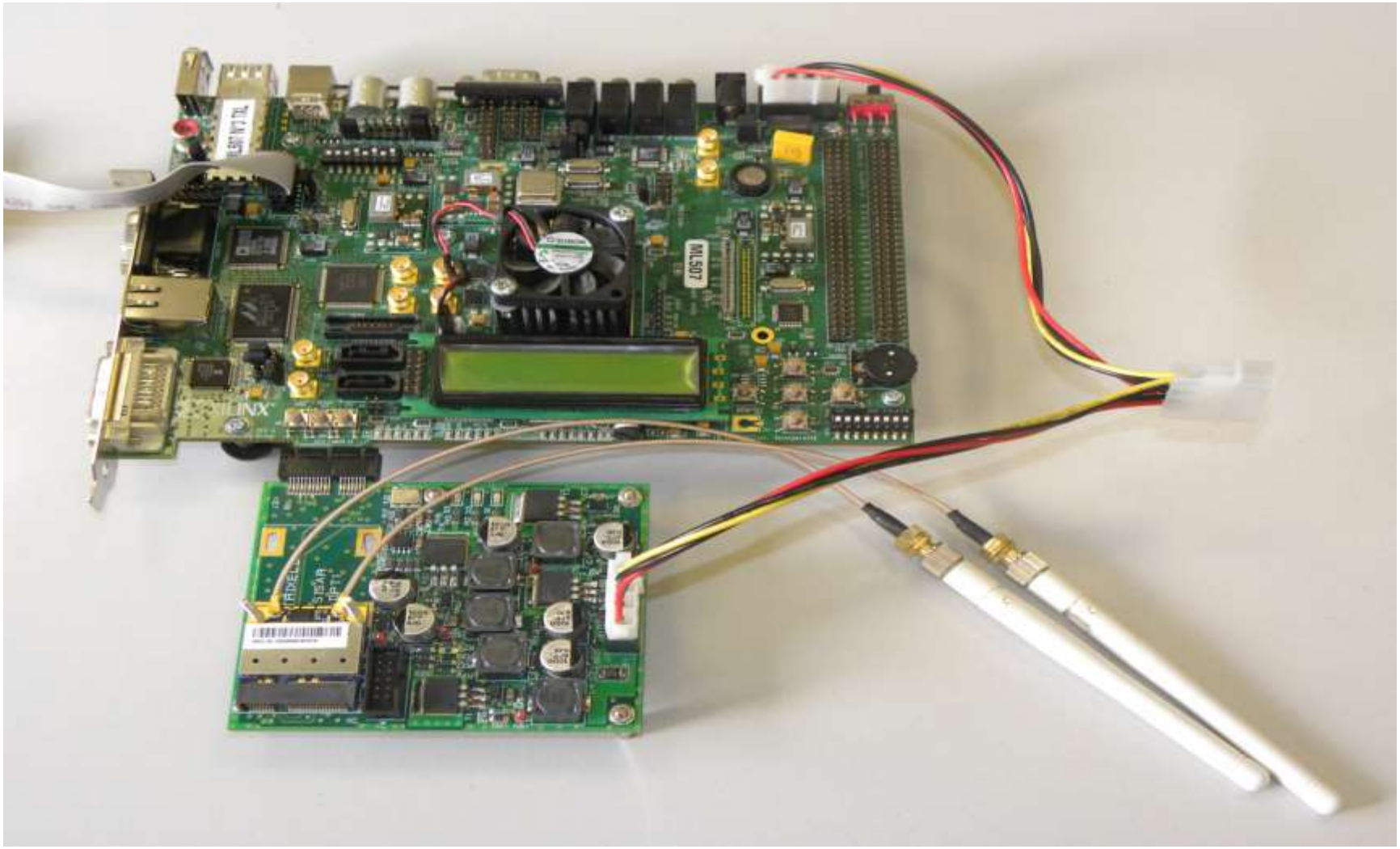


PCIe – Carte d'adaptation 1 – « ADPT1 »

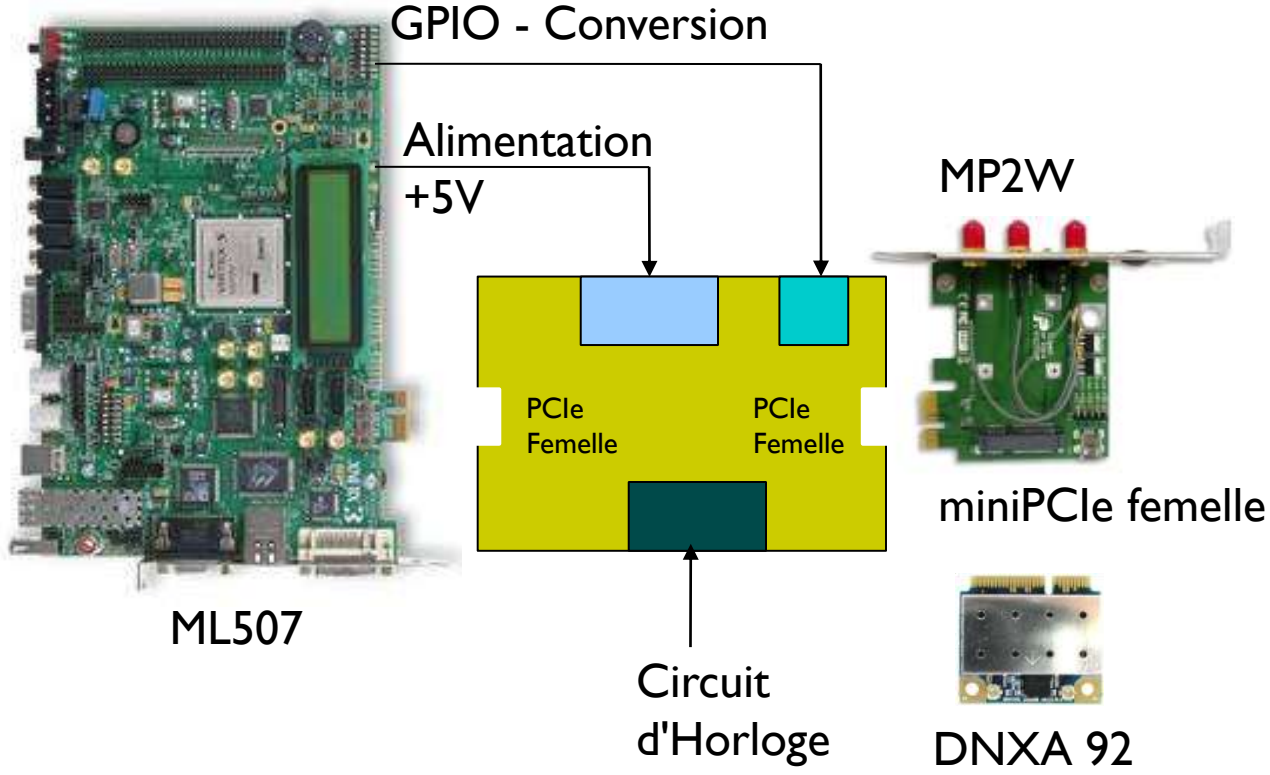


- Schématique réalisée 12/03/2010 ✓
- Production terminée 14/04/2010 ✓

PCIe – Carte d'adaptation 1 – « ADPT1 »

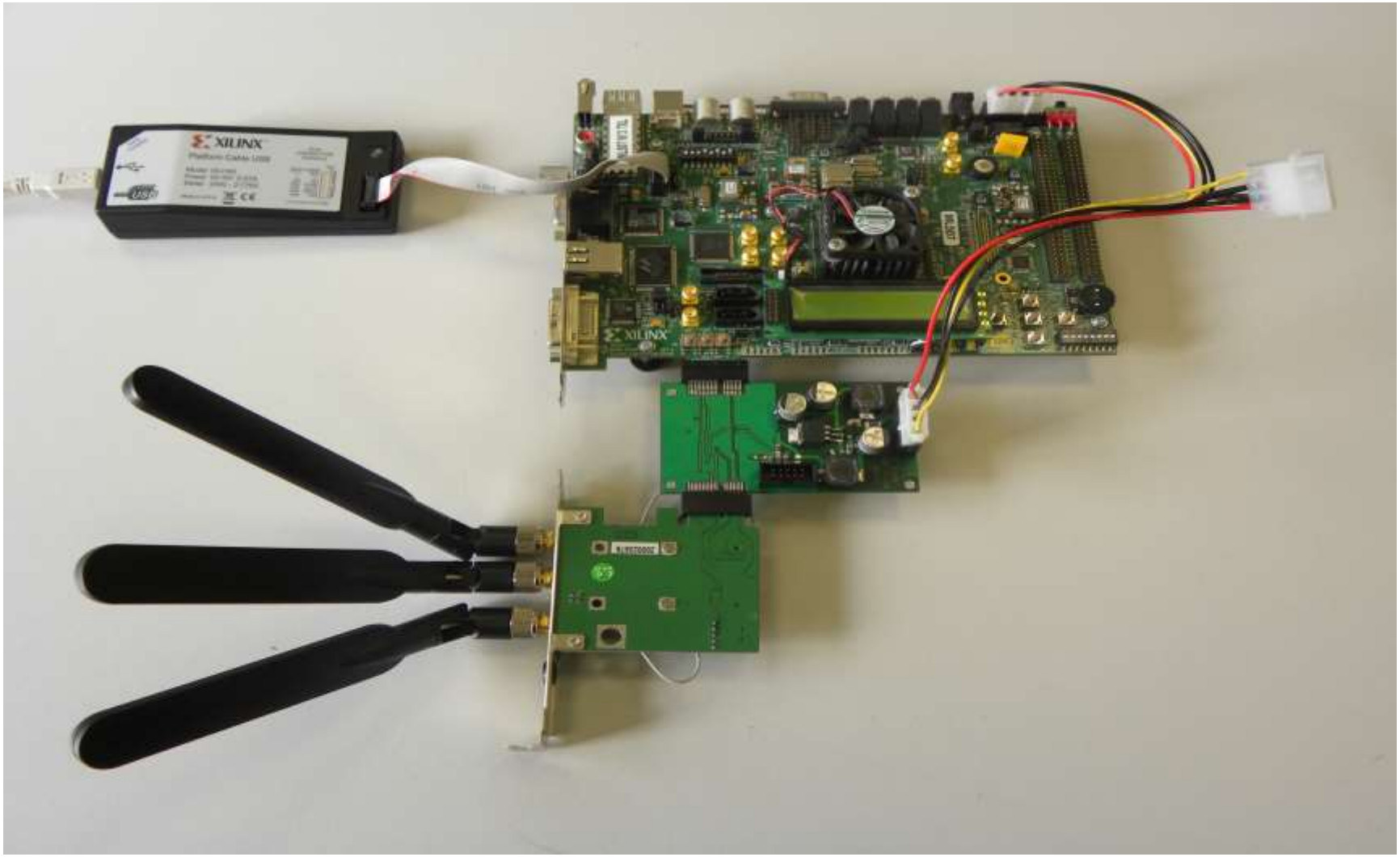


PCIe – Carte d'adaptation 2 – « ADPT2 »



- Schématique réalisée 14/04/2010 ✓
- Routage terminé 07/05/2010 ✓
- Câblage achevé 02/06/2010 ✓

PCIe – Carte d'adaptation 2 – « ADPT2 »



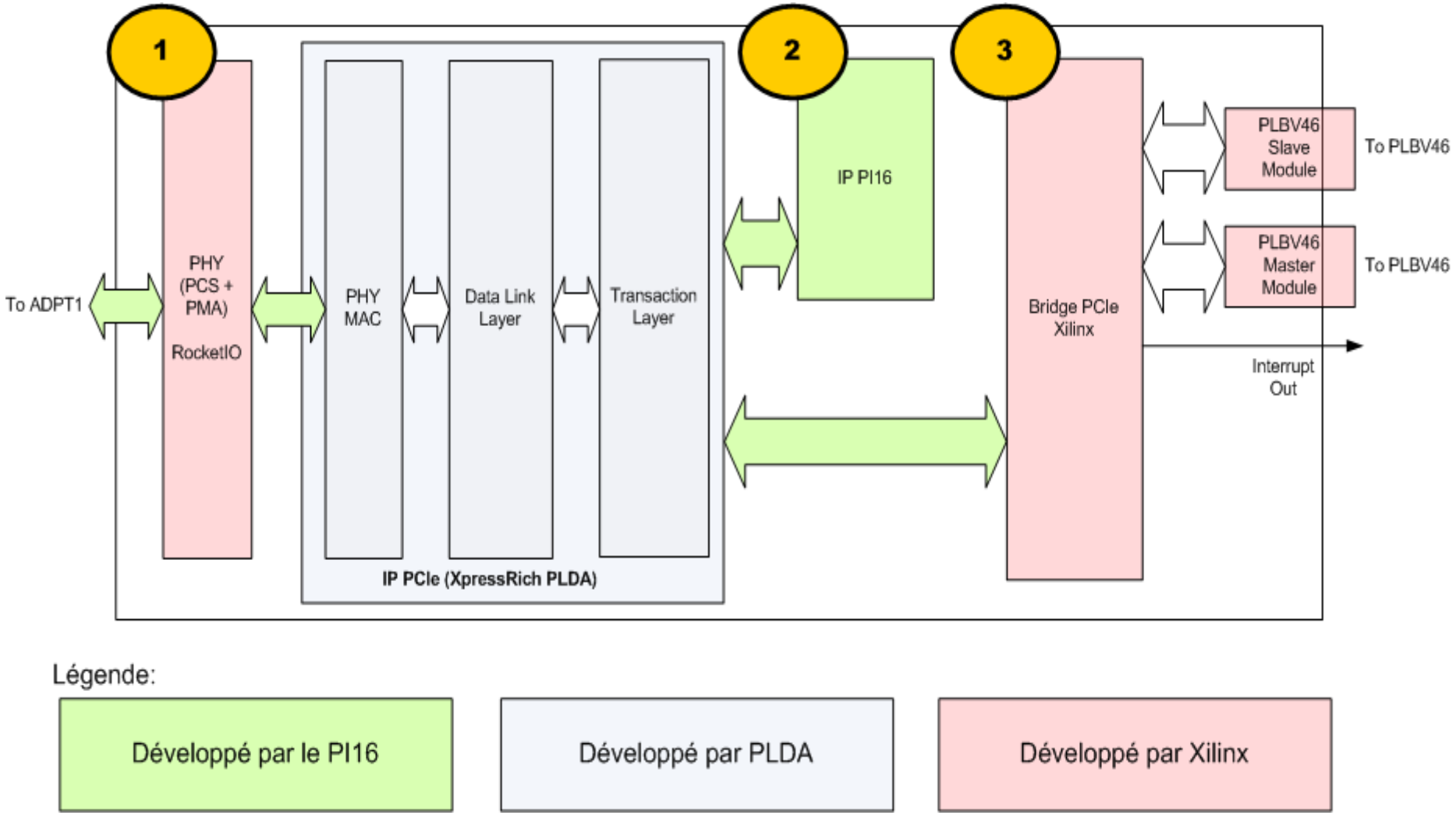
PCIe – Conception des cartes

- Travail avec un sous-traitant
  - Sésame PCB
    4 rue du père Louvat - 38500 VOIRON

  - Réunion pour s'assurer des bonnes contraintes de placement
  - Routage et PCB

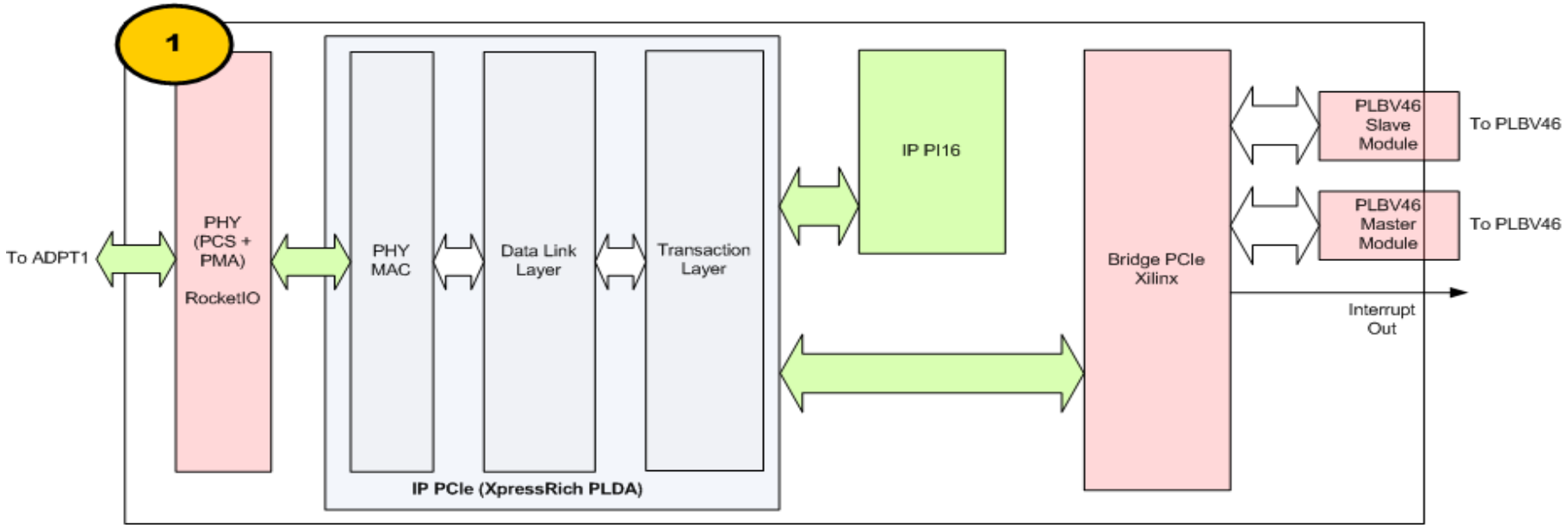
- Carte couplée avec des paires différentielles à haute fréquence et oscillateur 100 Mhz
  - Acquisition d'un savoir-faire technologique
  - Retranscrit pour la ADPT2 et la carte de bouclage MiniPCIe

PCIe – Architecture FPGA de la solution



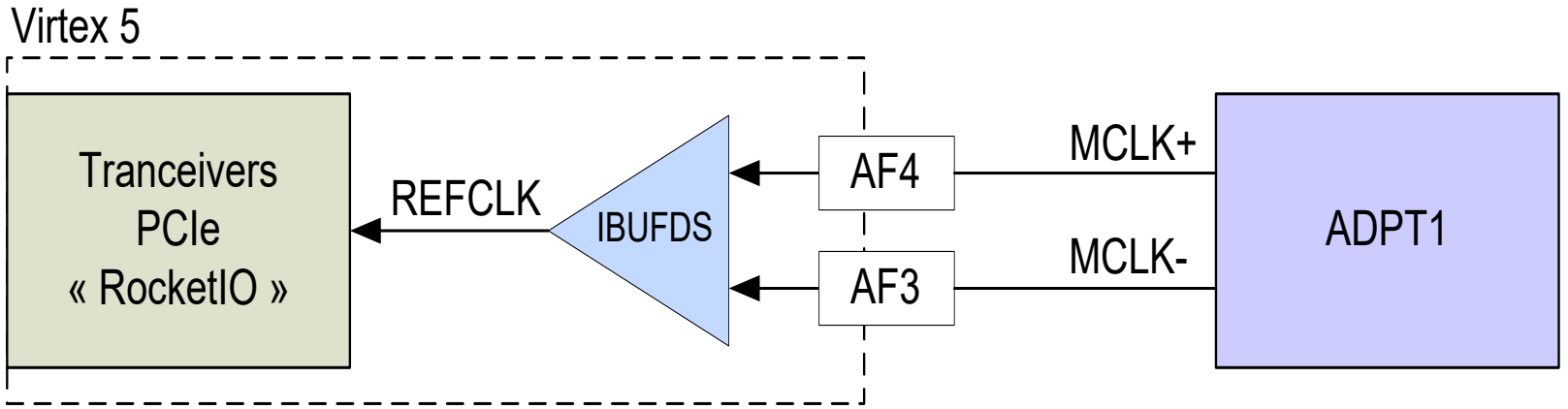
Partie 1

Les transceivers PCIe

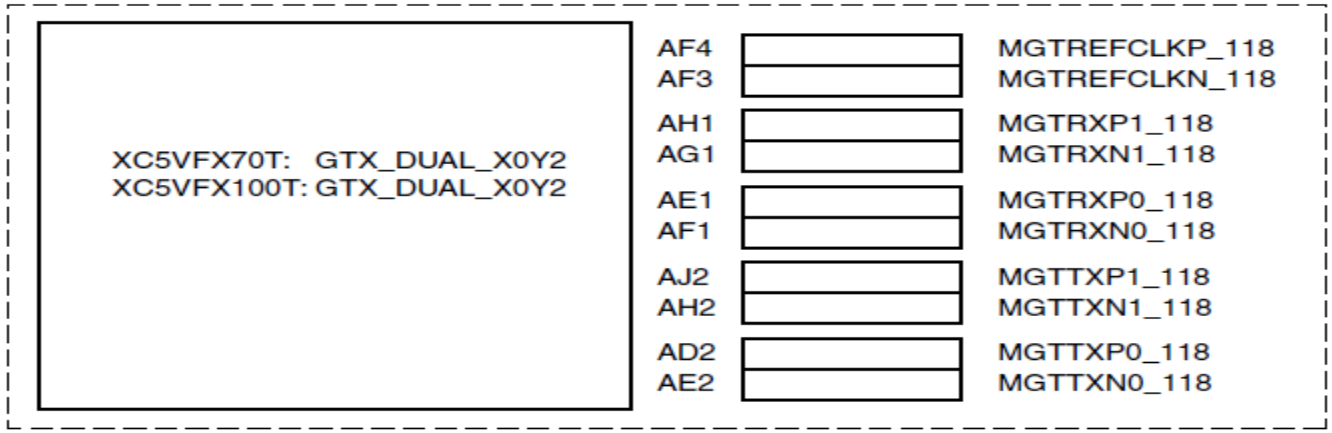


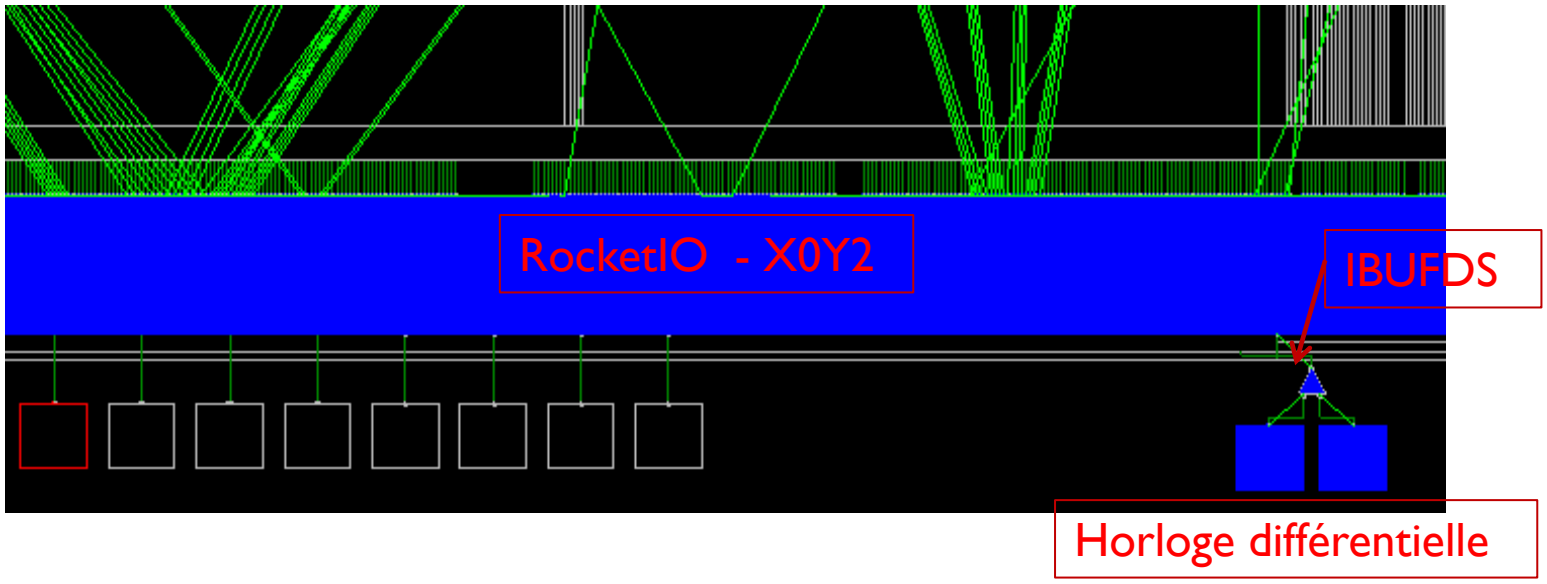
PCIe – Connexion aux pin matériels PCIe

- Fonction des RocketIO
- Recherche des fichiers de contraintes
- Connexion de l'horloge différentielle à la ML507



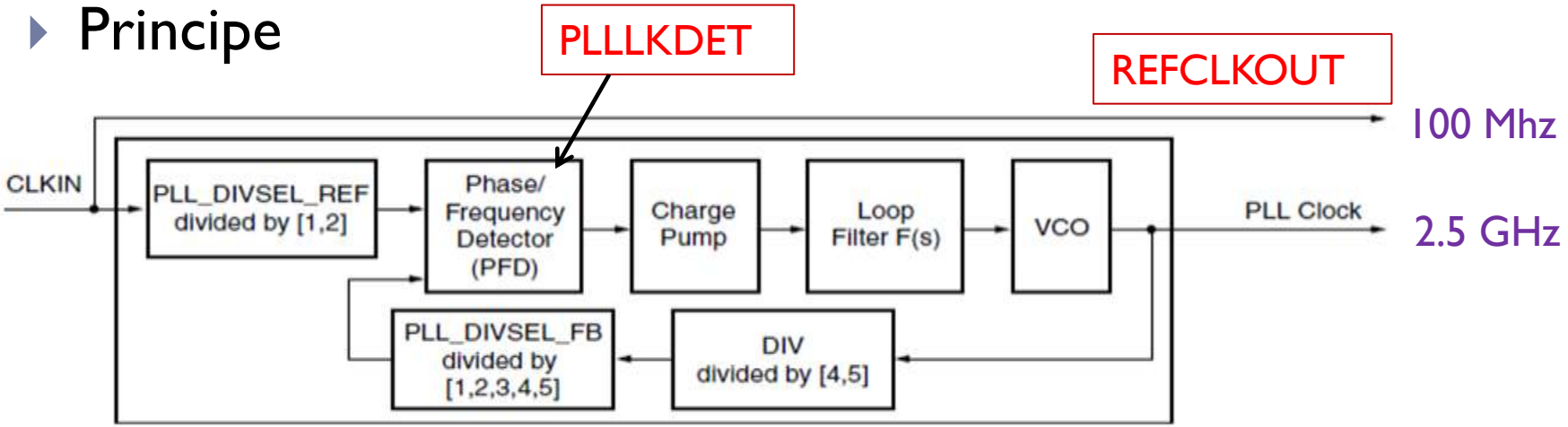
PCIe – Connexion des pins PCIe aux RocketIO



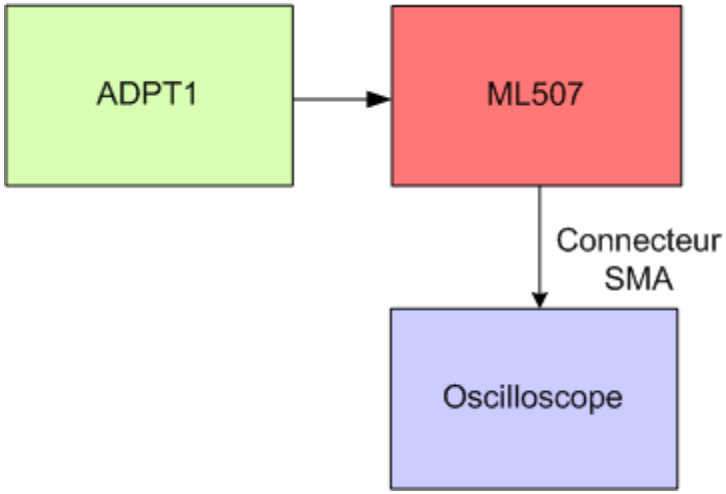


PCIe – Test de l'oscillateur externe

- Principe



- Test

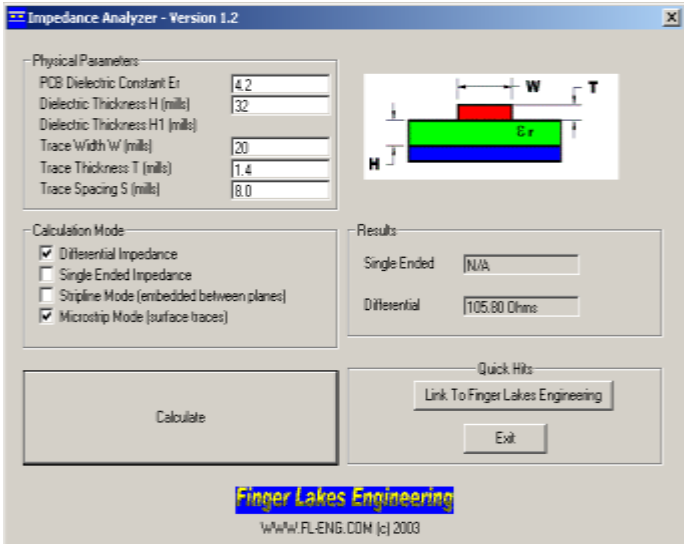


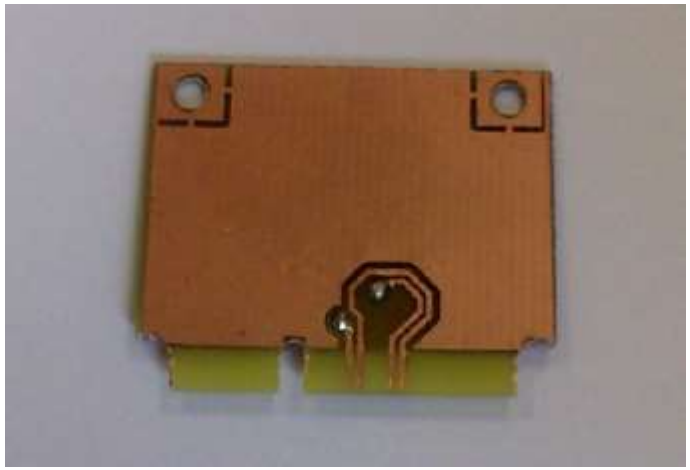
PCIe – Carte de bouclage MiniPCIe

- Caractéristiques
  - Classe 4 sur 2 couches
  - Epaisseur : 0.8 mm
  - Distance entre les pistes: 8 mils
  - Largeur des pistes: 20 mils
  - Paires différentielles: 100 Ω

@ 2.5 GHz

- Routage réalisé 21/05/2010 ✓
- Fin de la fabrication 26/05/2010 ✓

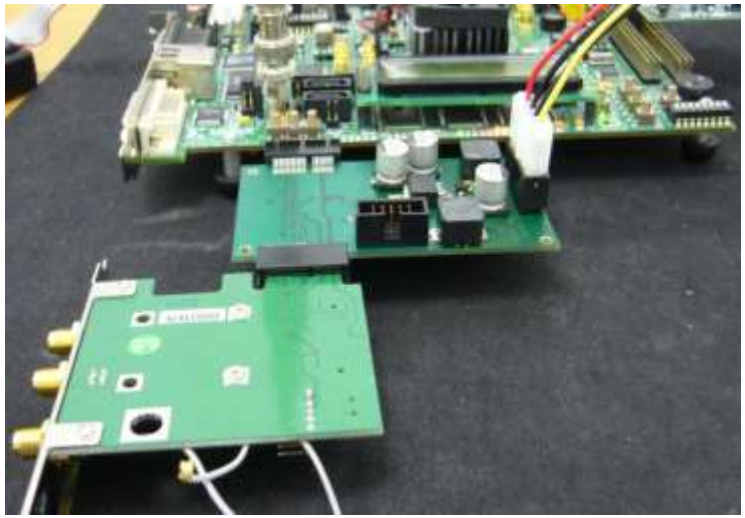




PCIe – Tests IBERT

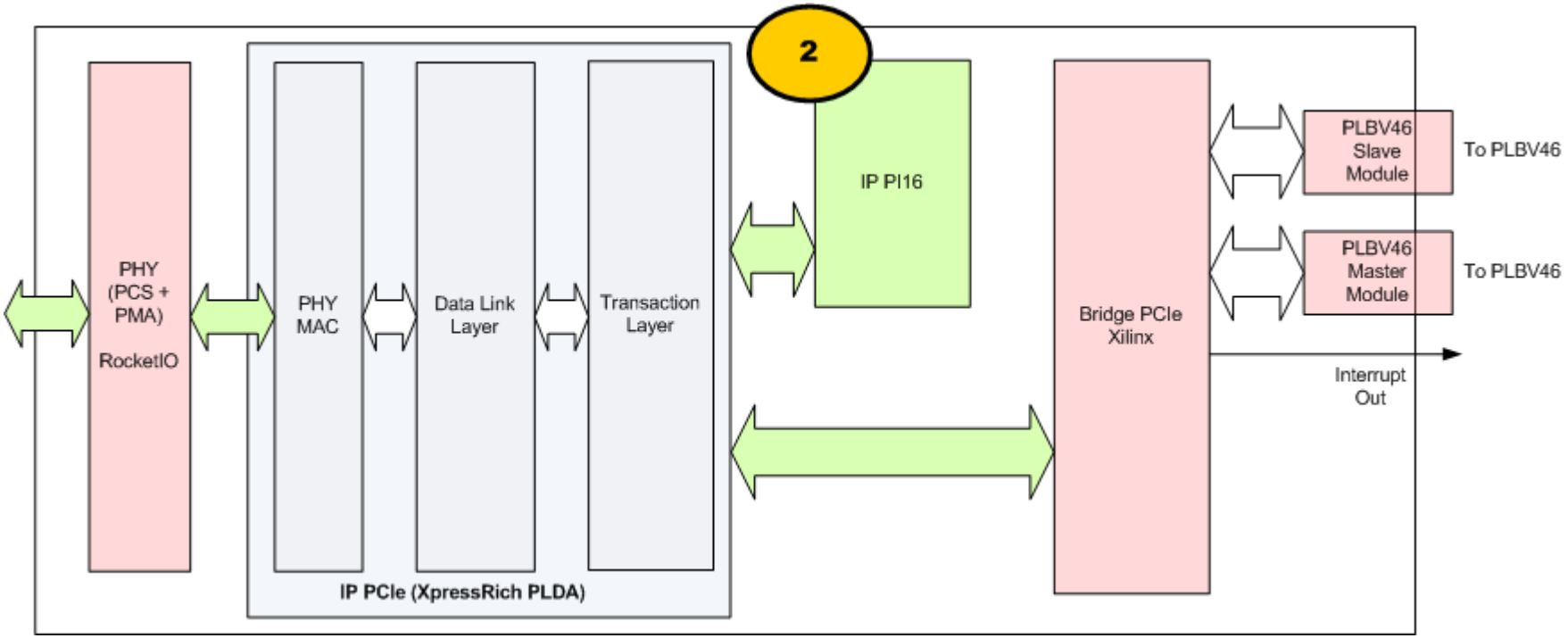
- Principe
- Carte de Loopback
- Validation des cartes ADPT1/ADPT2
- Fiche de test

BERT Settings		
TX/RX Data Pattern	PRBS 7-bit	PRBS 7-bit
RX Bit Error Ratio	6,444E-014	6,444E-014
RX Line Rate	2,500 Gbps	2,500 Gbps
RX Received Bit Count	1,552E013	1,552E013
RX Bit Error Count	0,000E000	0,000E000
BERT Reset	Reset	Reset



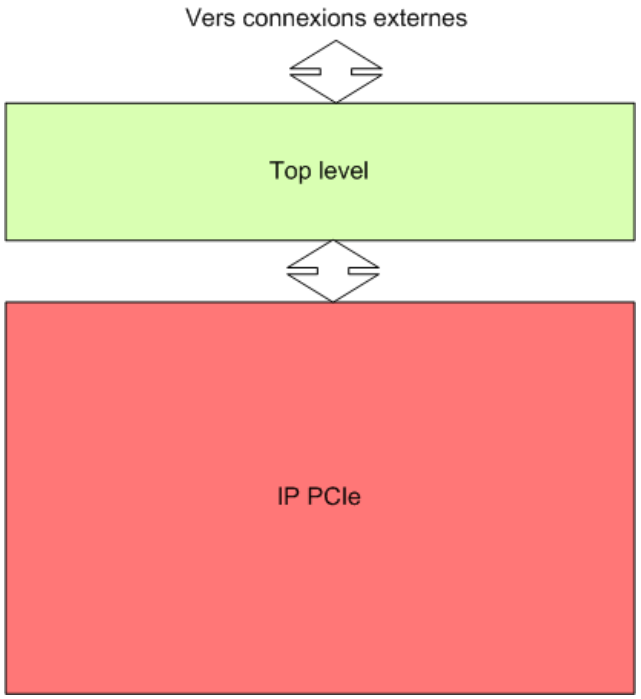
Partie 2

L'IP PI16



PCIe – Configuration de l'IP PLDA

- Configuration du Top level
- Besoins de Trixell
- Production d'un tutoriel



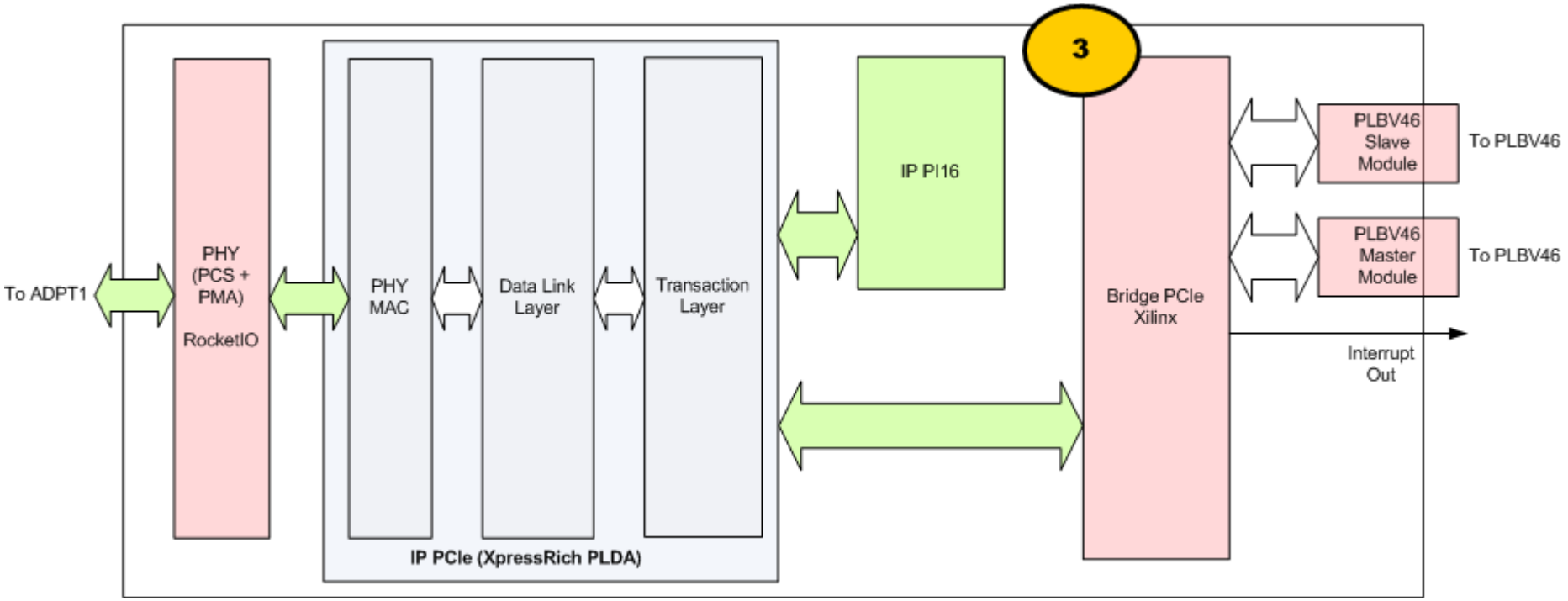
PCIe – Modules

- Clock
  - DCM 125 Mhz
- Reset
  - Spécification
  - Testbench
- Registre d'état de l'IP PCIe
  - Spécification
  - Testbench
- Registres de configuration PCIe
  - Spécification
  - Testbench
- Moyens de debug
  - Sorties SMA
  - Oscilloscope in situ

ChipScope Pro

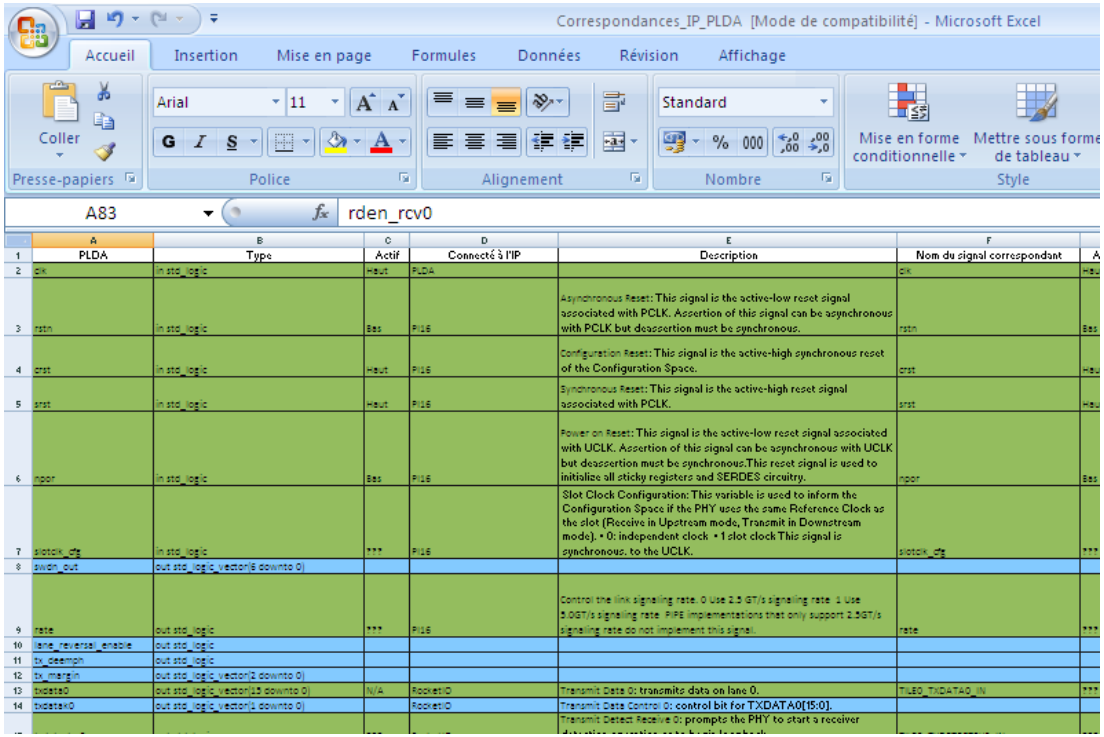
Partie 3

Jonction de l'IP PLDA avec le bridge Xilinx



PCIe – Correspondances IP PLDA avec Bridge Xilinx

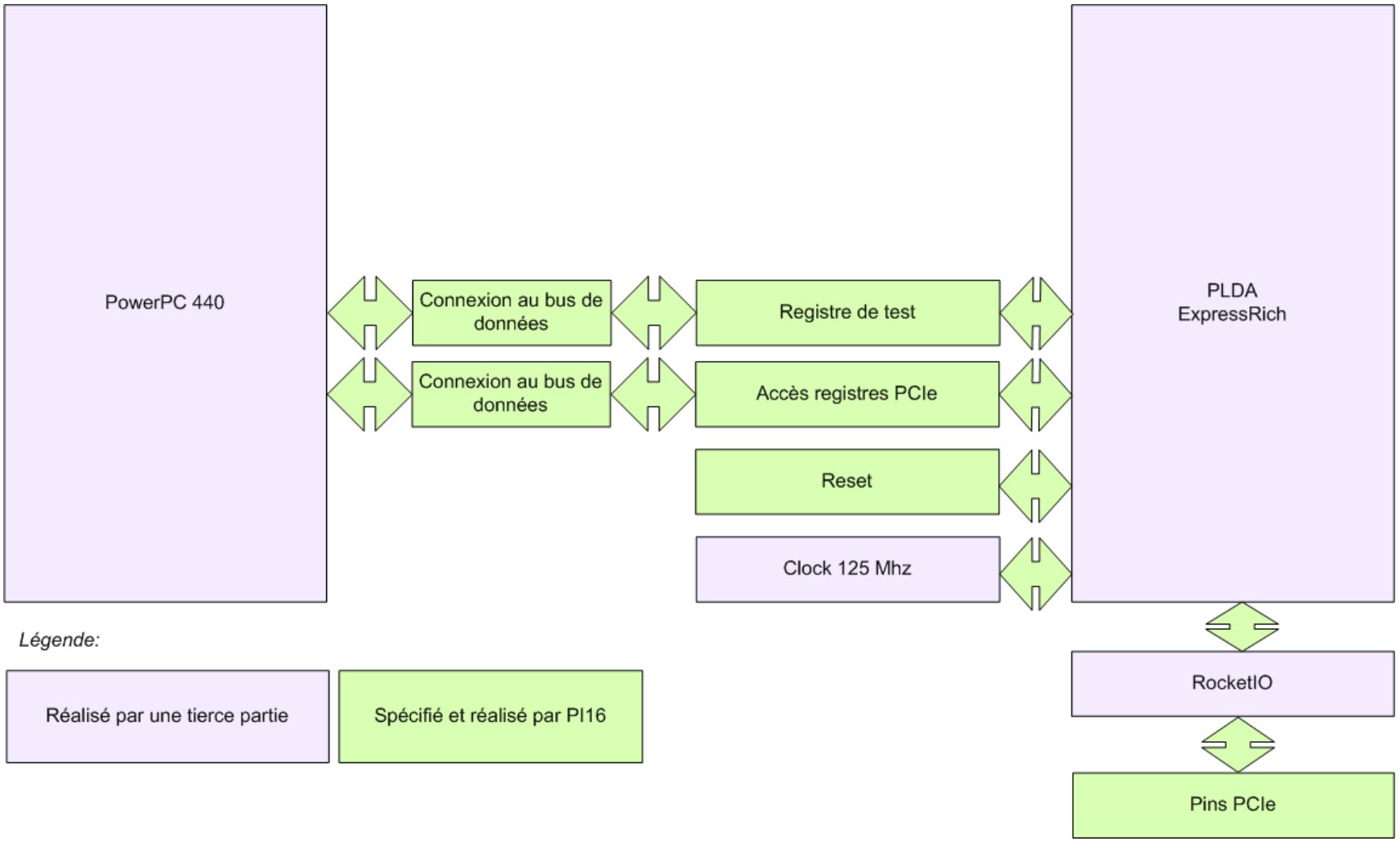
- Tableau
  - Signaux équivalents
  - Modifications nécessaires



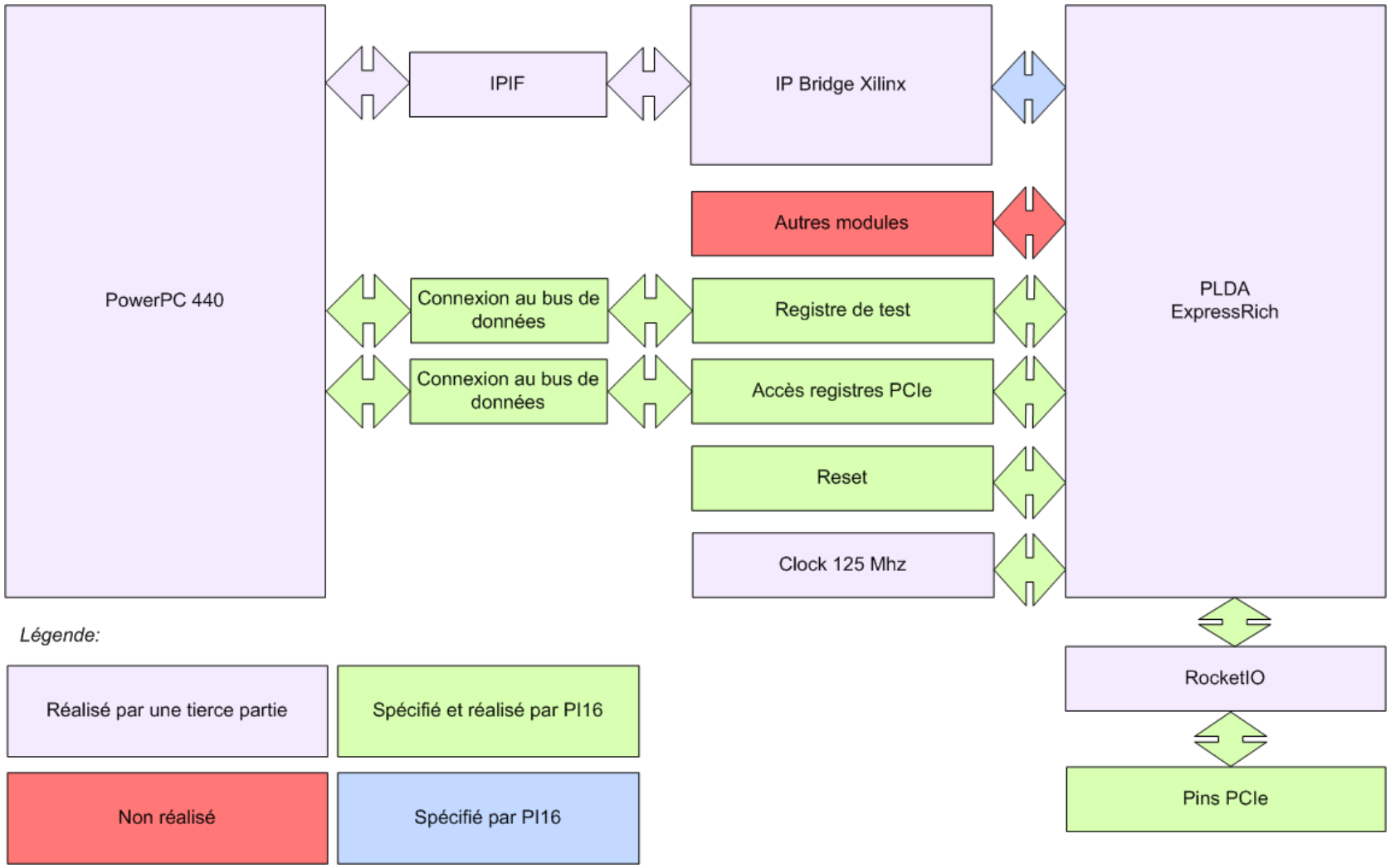
	A	B	C	D	E	F
1	PLDA	Type	Actif	Connecté à l'IP	Description	Nom du signal correspondant
2	clk	in std_logic	Haut	PLDA		clk
3	rstn	in std_logic	Bas	P116	Asynchronous Reset: This signal is the active-low reset signal associated with PCLK. Assertion of this signal can be asynchronous with PCLK but deassertion must be synchronous.	rstn
4	crst	in std_logic	Haut	P116	Configuration Reset: This signal is the active-high synchronous reset of the Configuration Space.	crst
5	srst	in std_logic	Haut	P116	Synchronous Reset: This signal is the active-high reset signal associated with PCLK.	srst
6	npor	in std_logic	Bas	P116	Power on Reset: This signal is the active-low reset signal associated with UCLK. Assertion of this signal can be asynchronous with UCLK but deassertion must be synchronous.This reset signal is used to initialize all sticky registers and SERDES circuitry.	npor
7	slotclk_cfg	in std_logic	***	P116	Slot Clock Configuration: This variable is used to inform the Configuration Space if the PHY uses the same Reference Clock as the slot (Receive in Upstream mode, Transmit in Downstream mode). • 0: independent clock • 1 slot clock This signal is synchronous. to the UCLK.	slotclk_cfg
8	swdn_out	out std_logic_vector(6 downto 0)				
9	rate	out std_logic	***	P116	Control the link signaling rate. 0 Use 2.5 GT/s signaling rate 1 Use 5.0GT/s signaling rate PIPE implementations that only support 2.5GT/s signaling rate do not implement this signal.	rate
10	lane_reversal_enable	out std_logic				
11	tx_deemph	out std_logic				
12	tx_margin	out std_logic_vector(2 downto 0)				
13	txdata0	out std_logic_vector(15 downto 0)	N/A	RocketIO	Transmit Data 0: transmits data on lane 0.	TILE0_TXDATA0_IN
14	txdatak0	out std_logic_vector(1 downto 0)		RocketIO	Transmit Data Control 0: control bit for TXDATA0[15:0].	
					Transmit Detect Receive 0: prompts the PHY to start a receiver	

Résultats

PCIe – Résultats – Architecture de test



PCIe – Résultats – Architecture finale



PCIe – Problèmes rencontrés

- Réactivité et implication du support technique des partenaires technologiques

- Matériels
  - Réajustements pour suivre la norme MiniPCIe dans les cartes d'adaptation

- Suite logicielle
  - Apprentissage d'outils nouveaux : ISE/EDK
  - Problèmes de synthèse avec XST
  - Problèmes d'intégration ISE/EDK

- Technique
  - Manque de documentation de l'IP PCIe

PCIe – Tutoriels

- Les plus importants
  - TU09 explique comment produire le wrapper pour les RocketIO à partir de Coregen
  - TU10 explique les paramètres qu'il faut utiliser pour générer correctement le top level de l'IP PCIe PLDA.
  - TU11 explique comment synthétiser l'IP du PI16 dans ISE
  - TU12 explique comment test les RocketIO de la carte ML507
  - TU13 explique comment synthétiser l'IP du PI16 dans EDK

PCIe – Bilan

Hardware

• Acquisition IP PLDA	17/03/2010	✓
• Recenser les fonctionnalités du PCIe nécessaires	10/04/2010	✓
• Configuration Wizard PLDA, tutoriel	26/04/2010	✓
• Connexion RocketIO à l'interface PIPE PDLA, tutoriel	19/05/2010	✓
• Tests IBERT (ADPT1/ADPT2/Loopback)	31/05/2010	✓
• Création des modules	21/05/2010	✓
• Accès aux registres de l'IP PCIe	19/05/2010	80%
• Correspondance IP PLDA / Bridge Xilinx	24/05/2010	✓
• Connexion de l'IP PLDA à l'IP Xilinx	Non traité	

Software

• IP PCIe : activation du driver PCIe Linux	Non traité
• Wifi : activation de la pile 802.11 de Linux	Non traité
• Wifi : activation du driver ath9k	Non traité
• Tests de performances et de fiabilité	Non traité

Déroulement du projet et perspectives

Apports personnels

- Gestion d'un projet
  - Réunions : Call conference, reportings
  - Organisation : Planning, Comptes rendu, Deadlines
  - Appels de fournisseurs étrangers (Taiwan)
- Prise en main de la suite de développement FPGA Xilinx
- Création d'IP en langage VHDL
- Développement drivers sous Linux
- Interface Wifi

Transfert de technologie

- Réalisation d'une base complète de tutoriels
- Réalisation d'une base de données sous SVN avec l'intégralité technique du projet
  - Contient l'intégralité du travail sur le projet
- Installation de la plateforme sur le site de l'entreprise

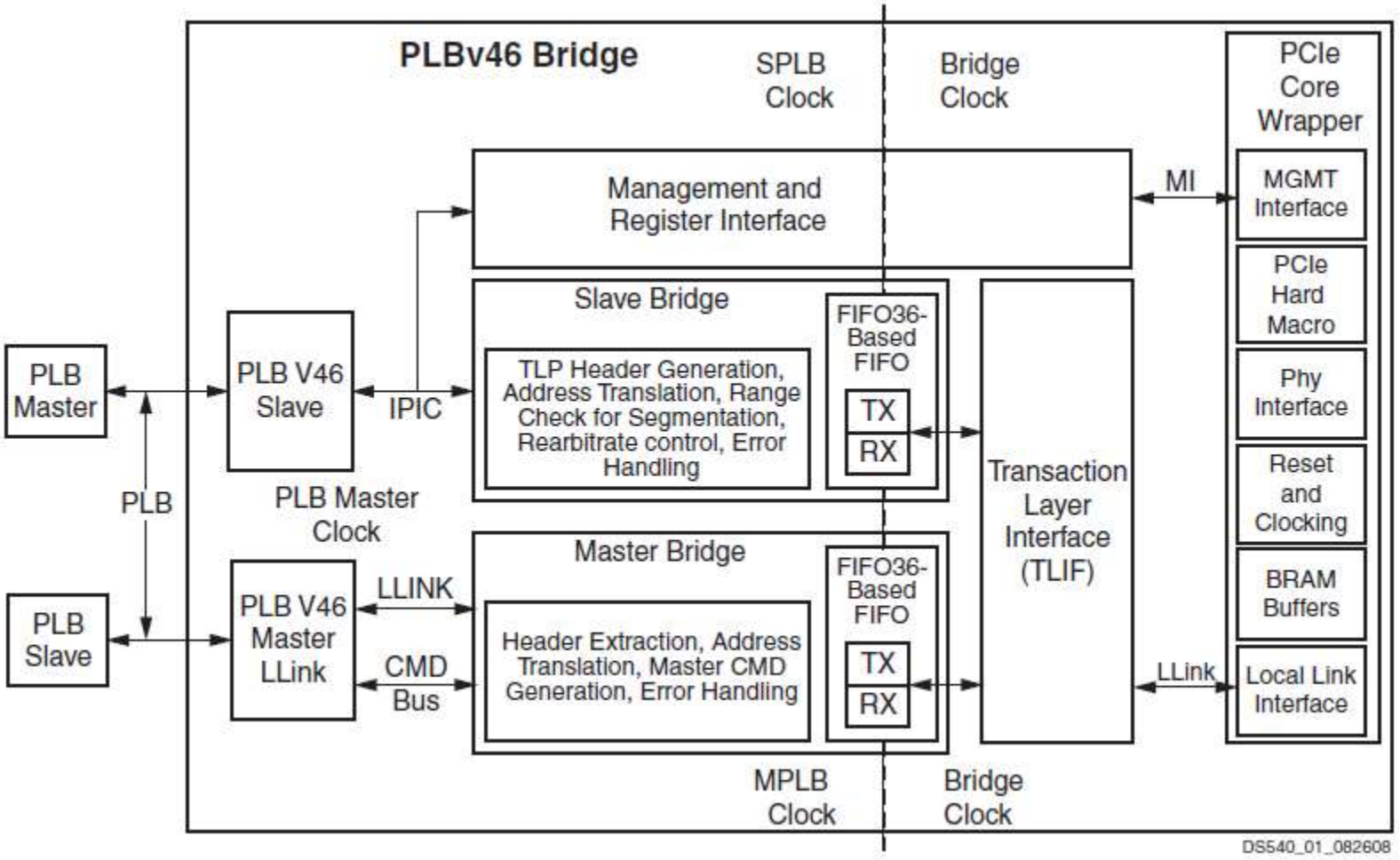
Suite du projet

- Intégration d'un module Wifi par Trixell
  - Solution USB retenue par Trixell sur son prochain prototype.
- Optimisation des débits sur le module USB
- Finalisation de l'intégration de la solution PCIe
  - Modification du Bridge Xilinx.
  - Attente du driver PCIe Linux en cours développement.

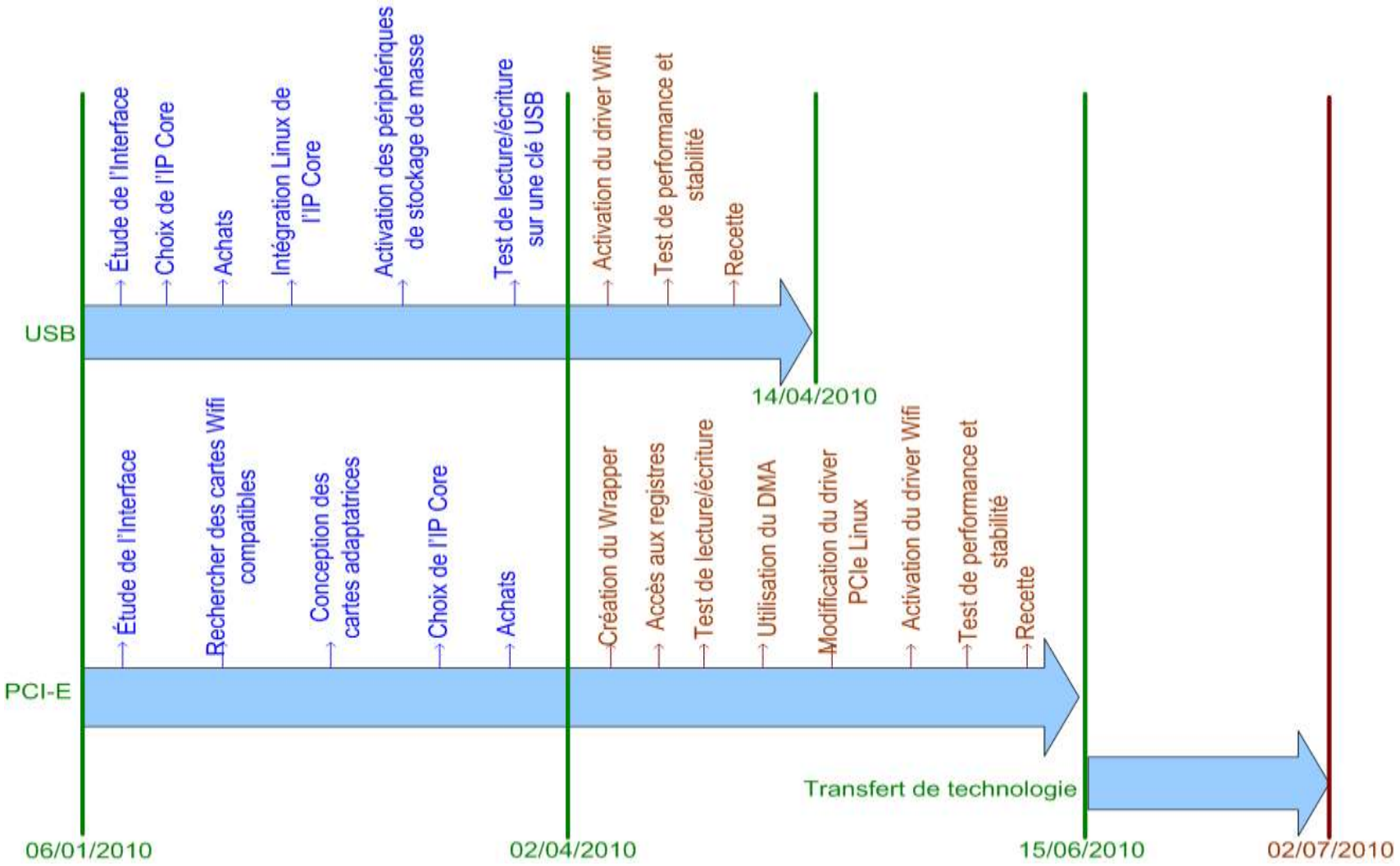
Conclusion

Merci de votre attention.

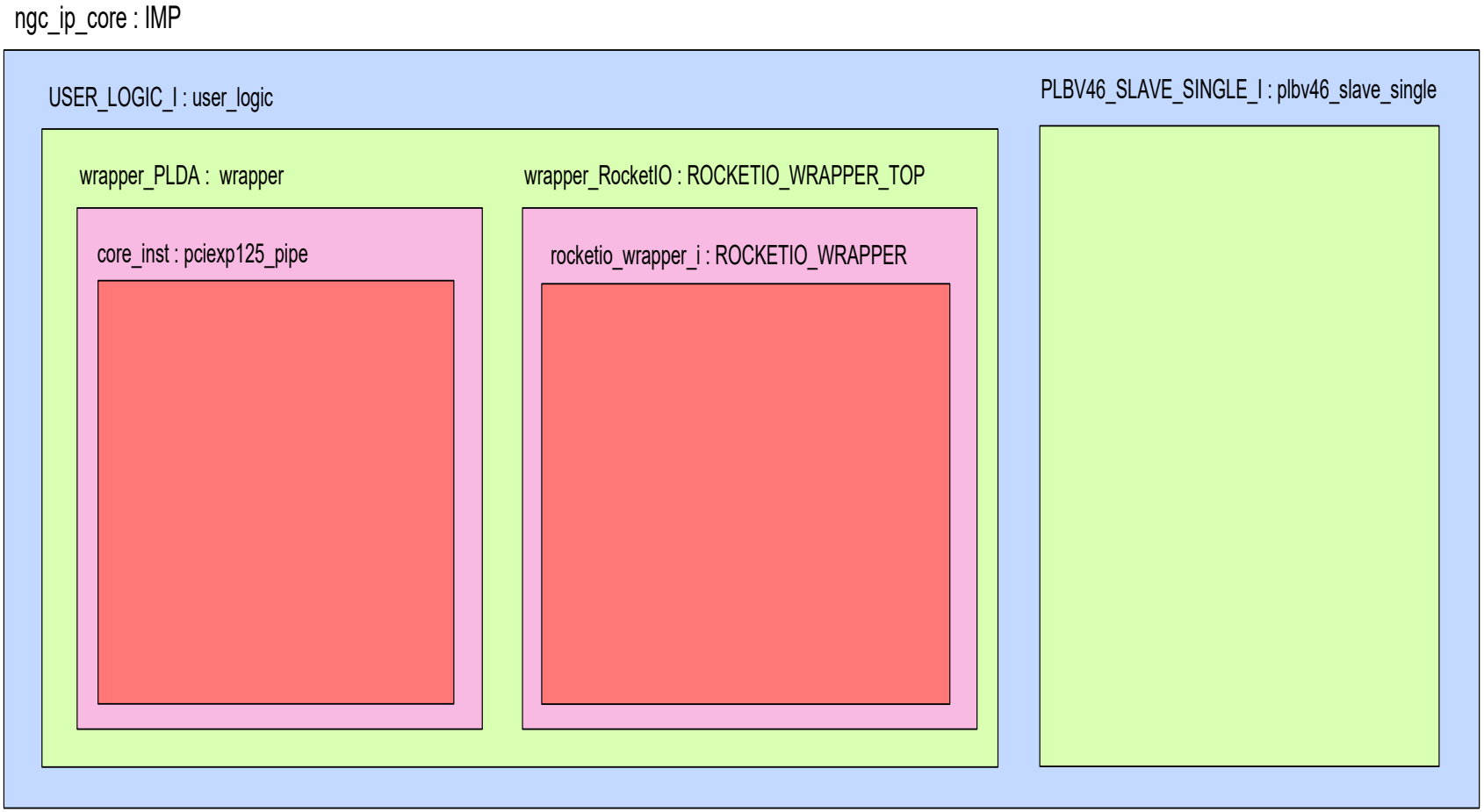
2 – PCIe – Xilinx / PLDA



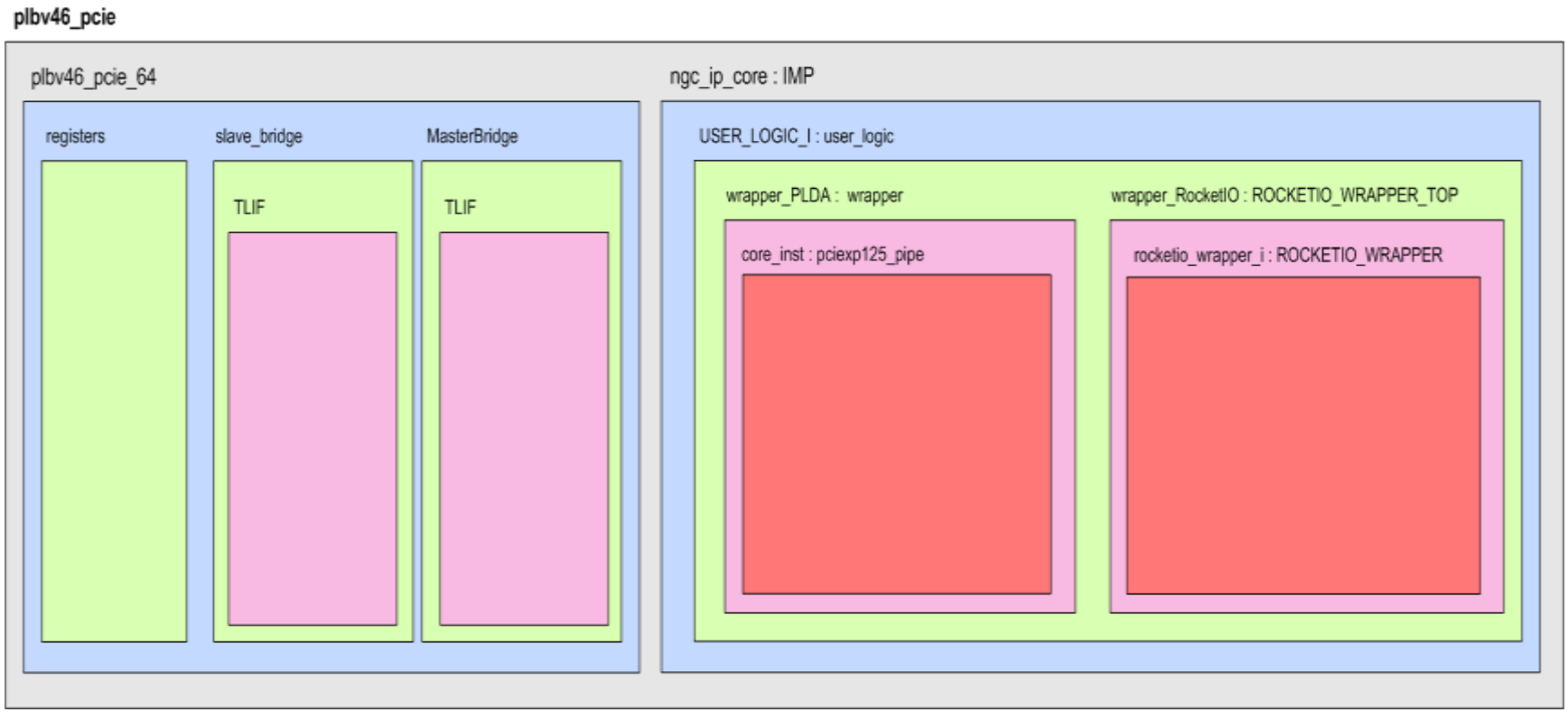
4 –Planning


USB
Étude de l'Interface
Choix de l'IP Core
Achats
Intégration Linux de l'IP Core
Activation des périphériques de stockage de masse
Test de lecture/écriture sur une clé USB
Activation du driver Wifi
Test de performance et stabilité
Recette
14/04/2010
PCI-E
Étude de l'Interface
Rechercher des cartes Wifi compatibles
Conception des cartes adaptatrices
Choix de l'IP Core
Achats
Création du Wrapper
Accès aux registres
Test de lecture/écriture
Utilisation du DMA
Modification du driver PCIe Linux
Activation du driver Wifi
Test de performance et stabilité
Recette
Transfert de technologie
06/01/2010
02/04/2010
15/06/2010
02/07/2010

2 – Architecture du wrapper PI16

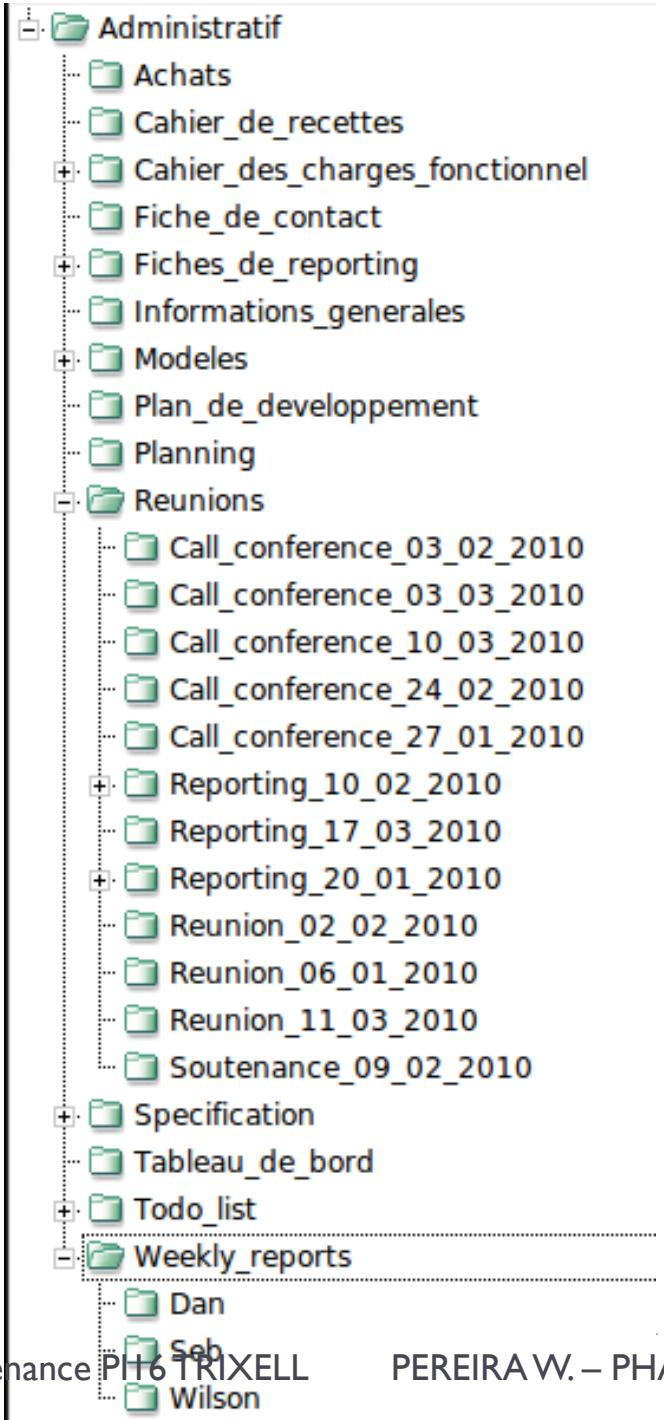


2 – Architecture du wrapper PTR6+EBridge Xilinx



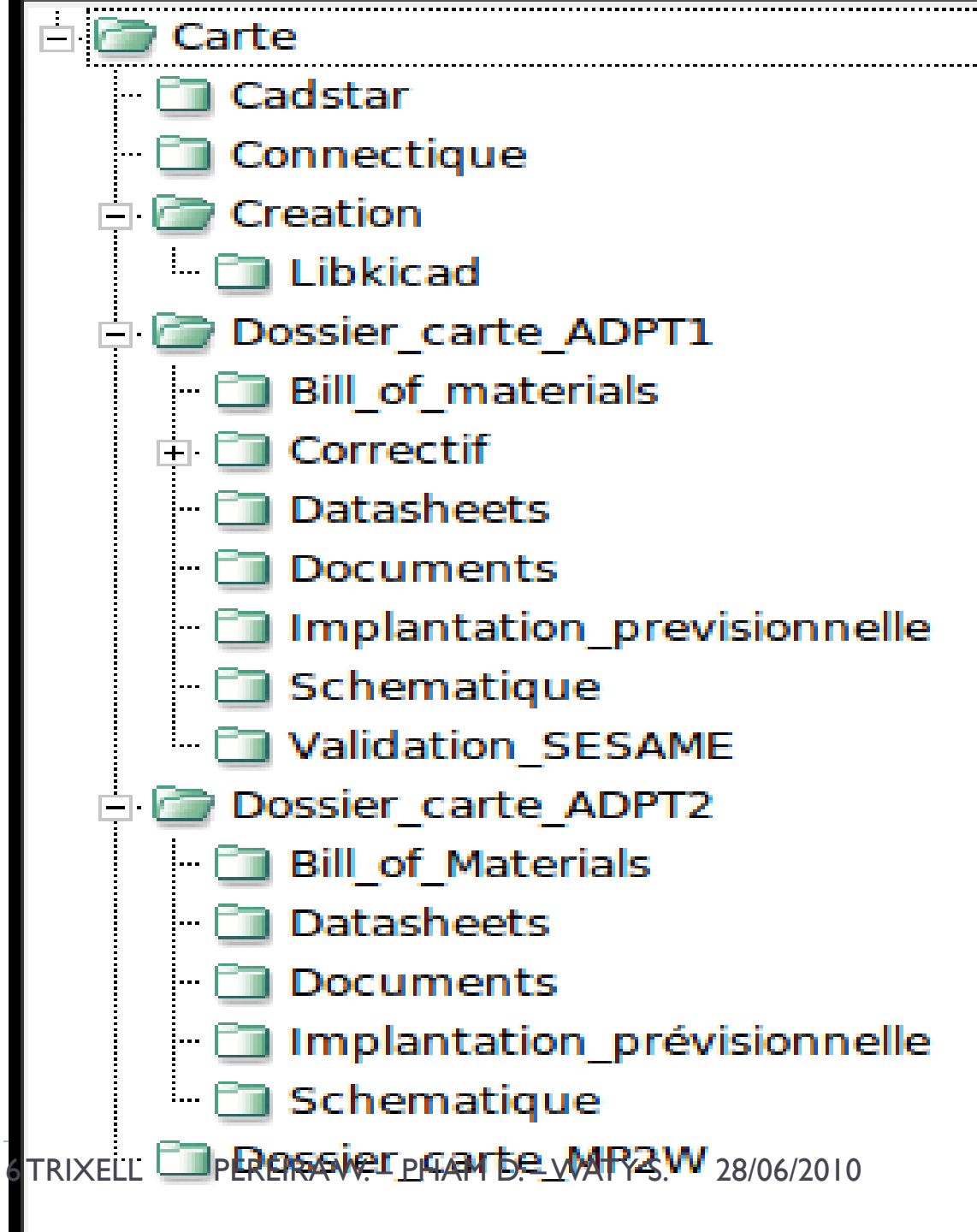
Organisation des dossiers informatiques

- Les documents « Administratif » :



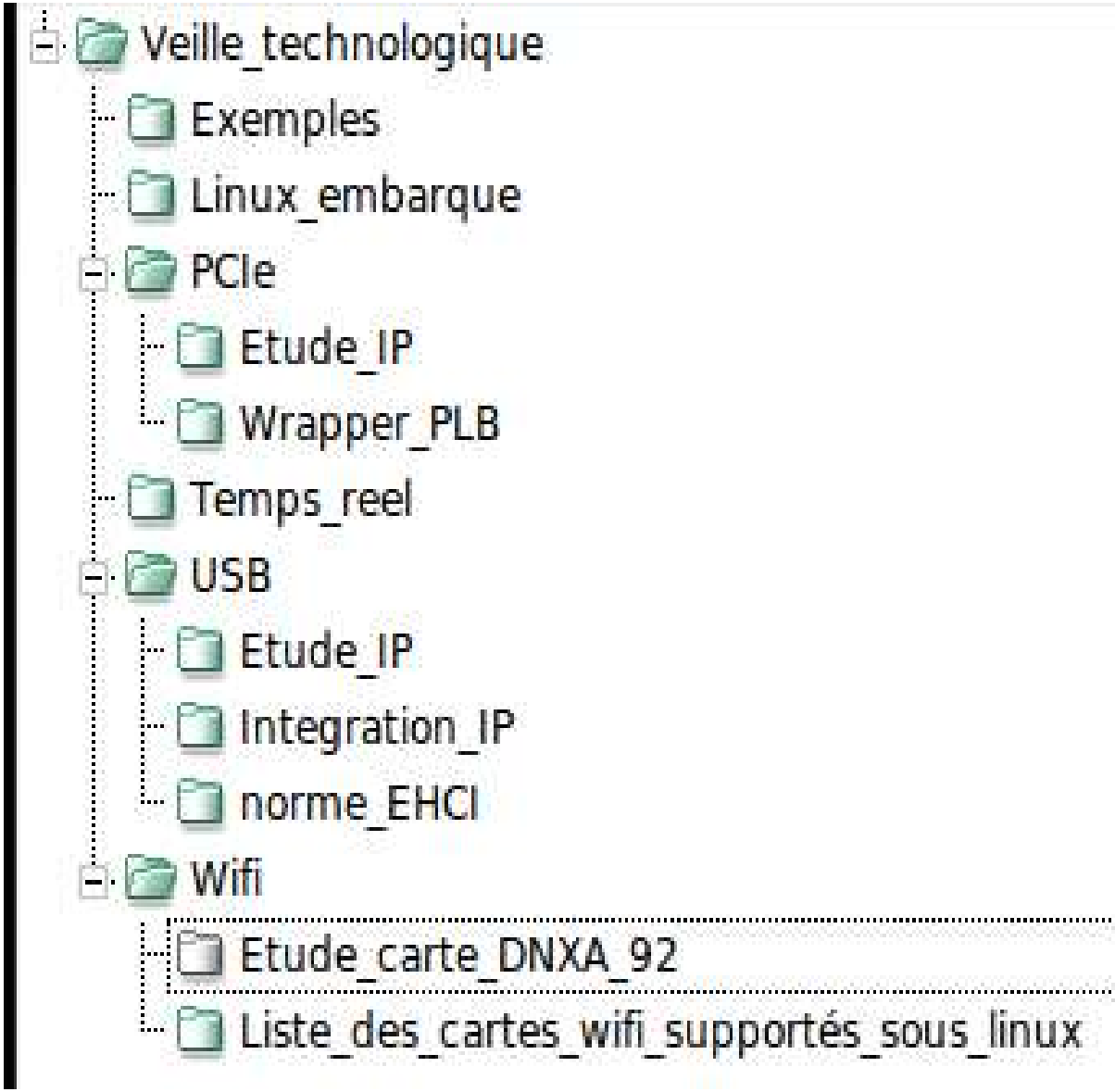
Organisation des dossiers informatiques

- Les documents concernant la conception de carte :



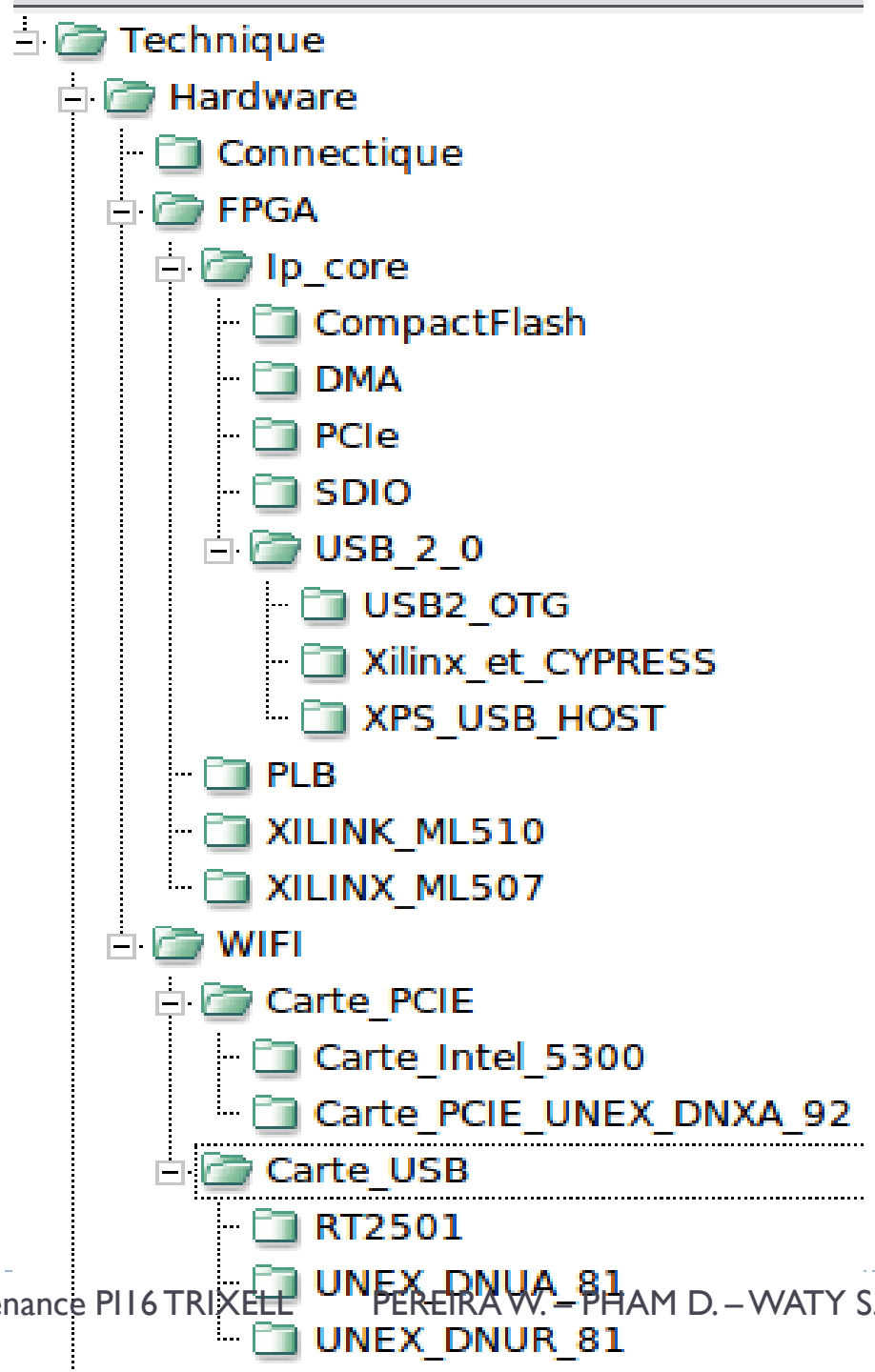
Organisation des dossiers informatiques

- Les documents de veille technologique :



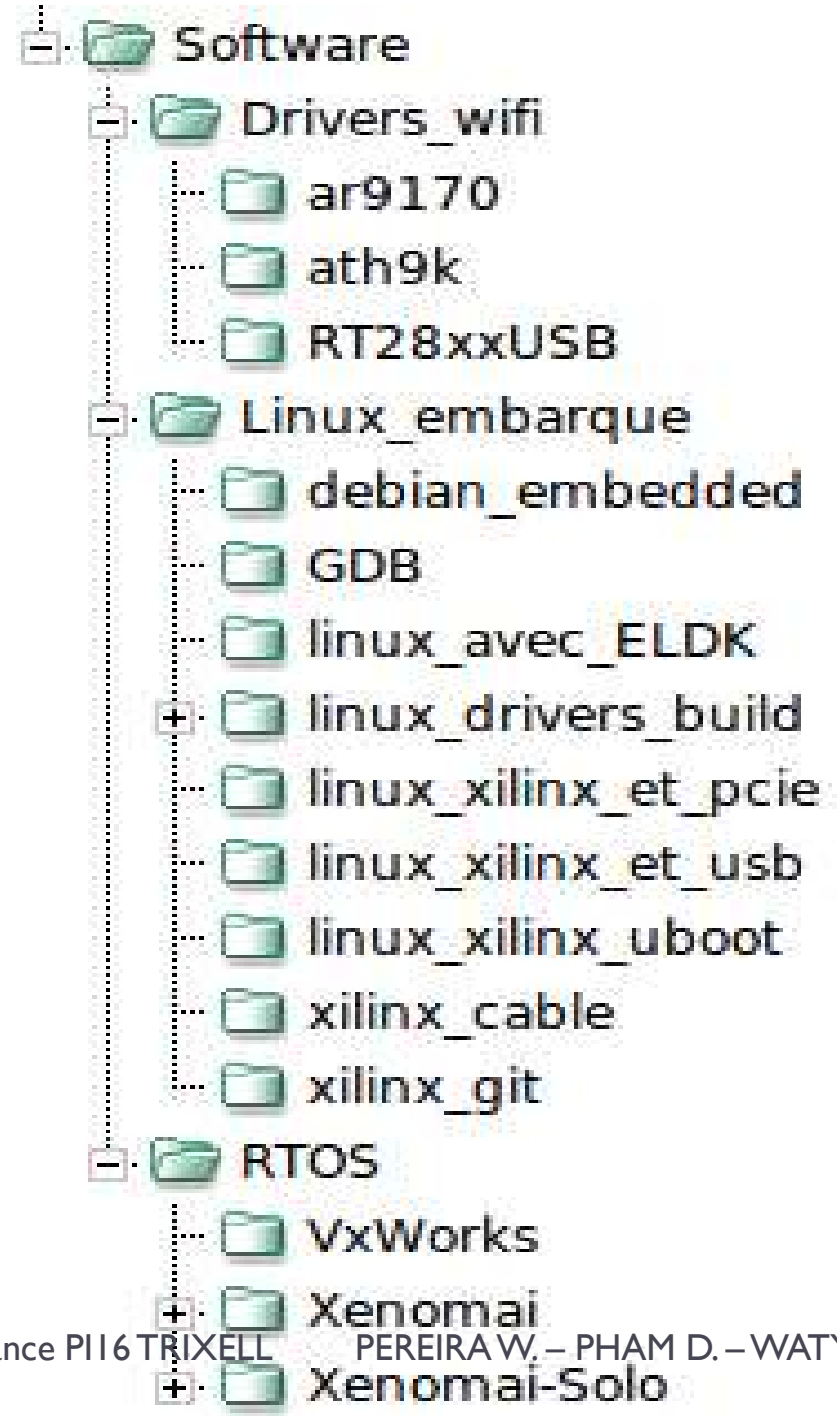
Organisation des dossiers informatiques

- Les documents de techniques :

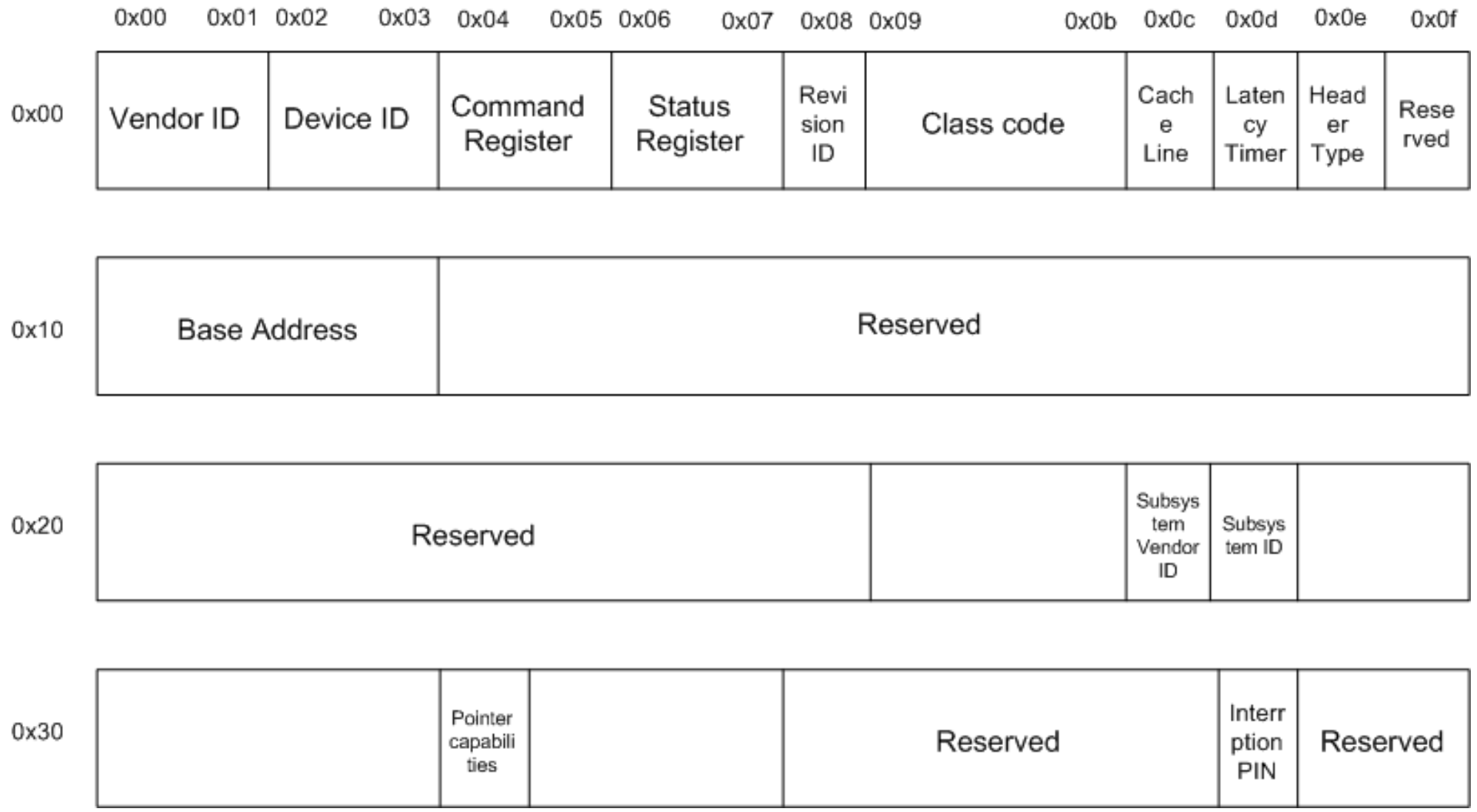


Organisation des dossiers informatiques

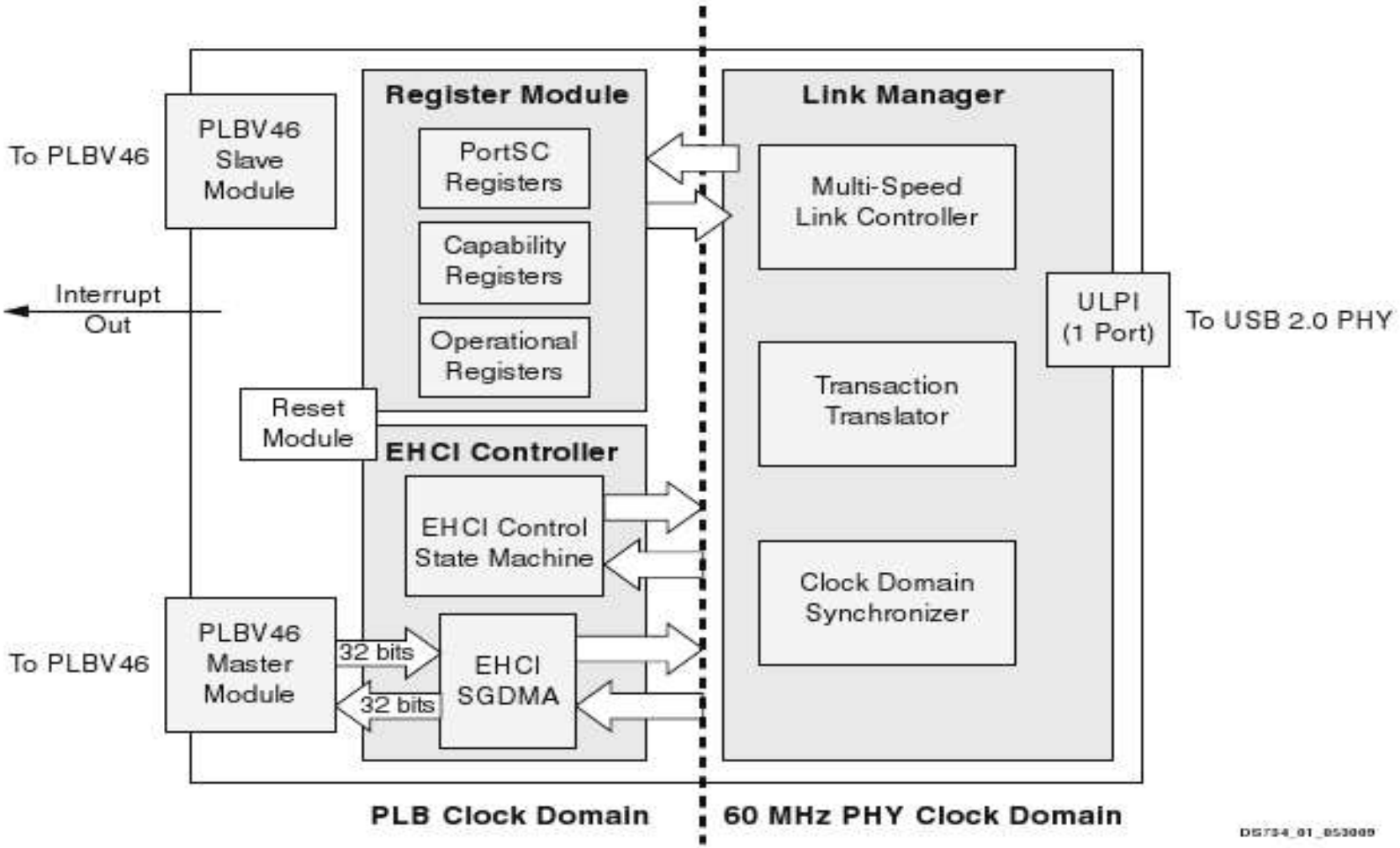
- Les documents de techniques :



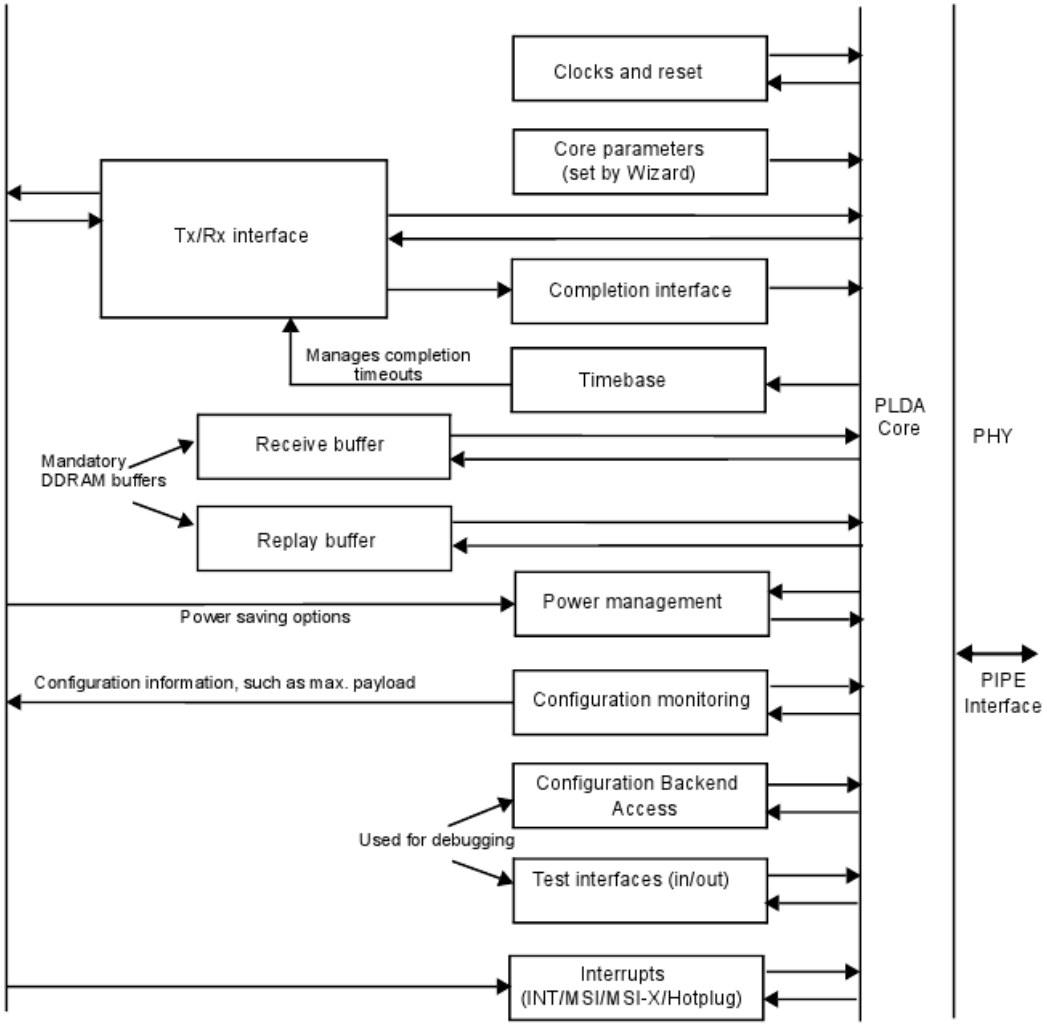
DNXA92 – Interface PCIe



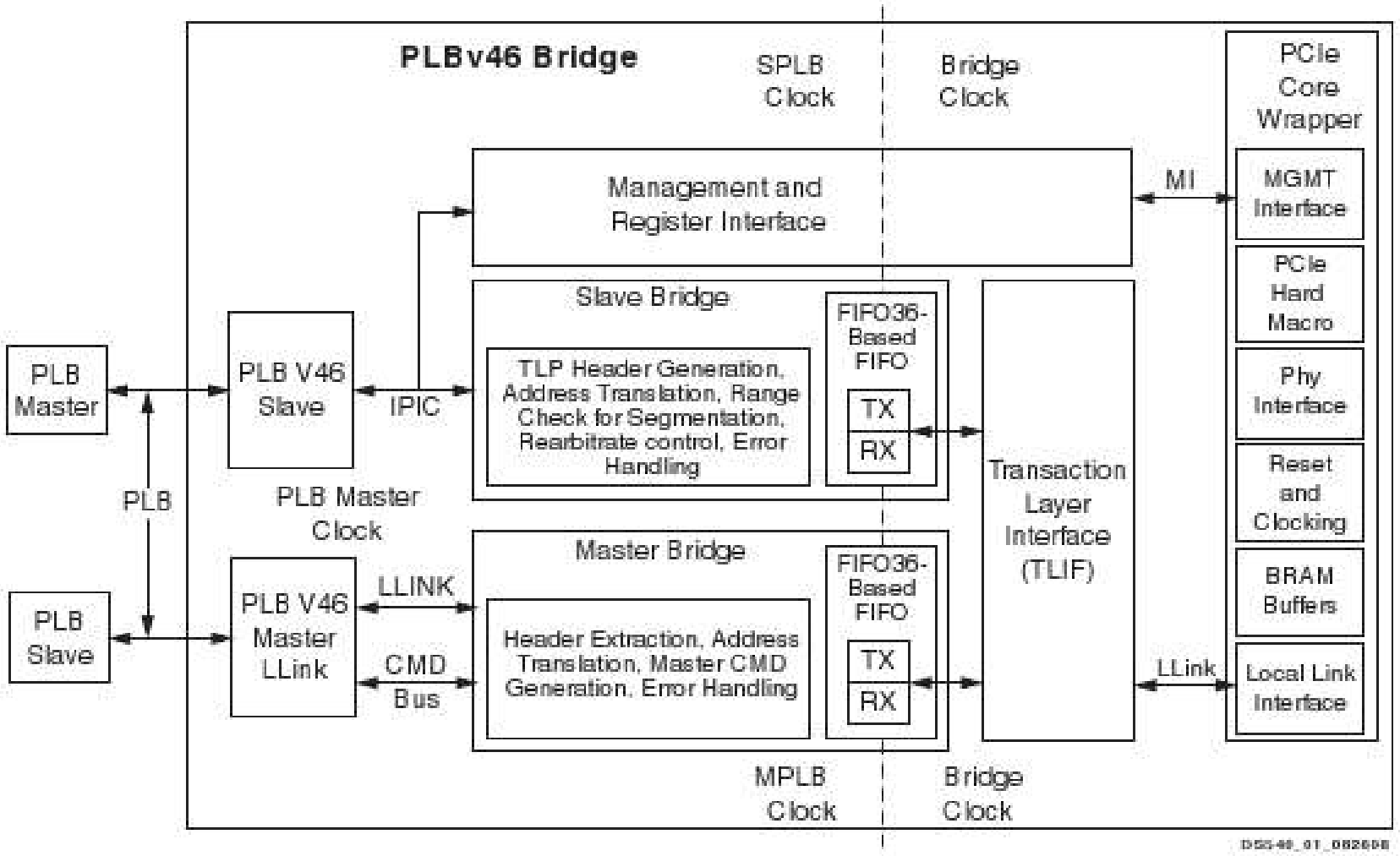
IP XPS_USB_HOST



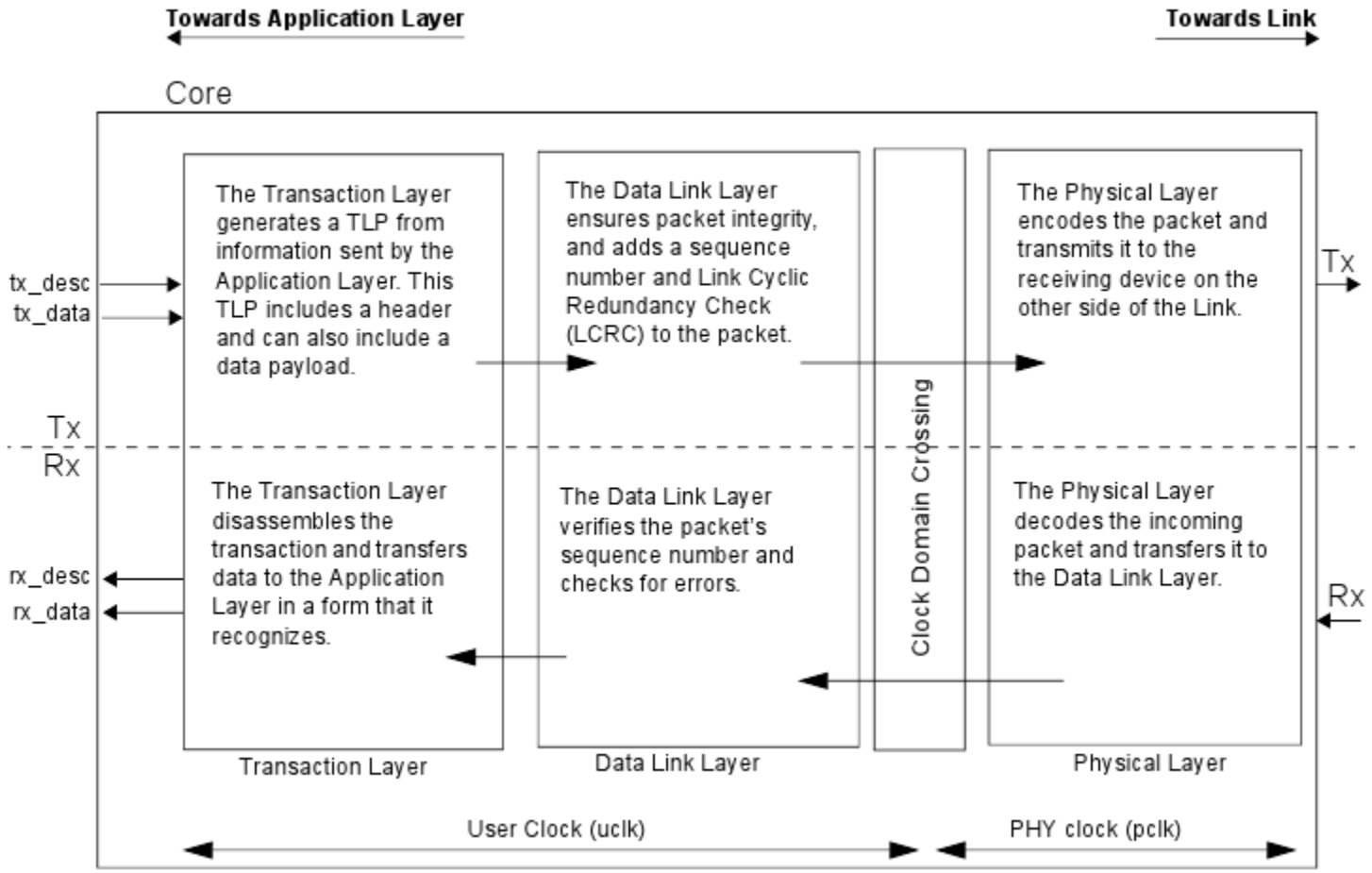
Modules du Wrapper



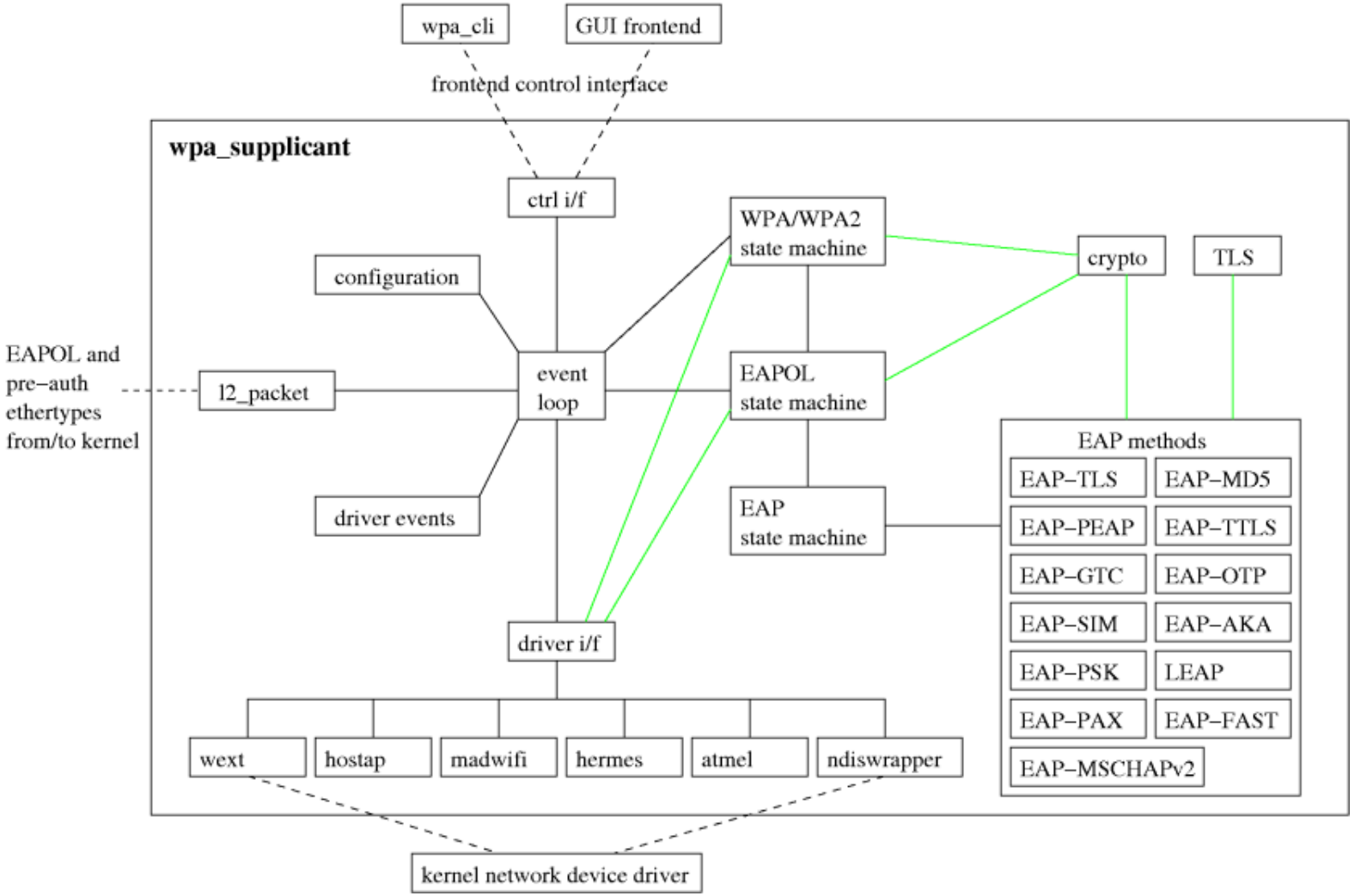
Bridge PLB - PCIe



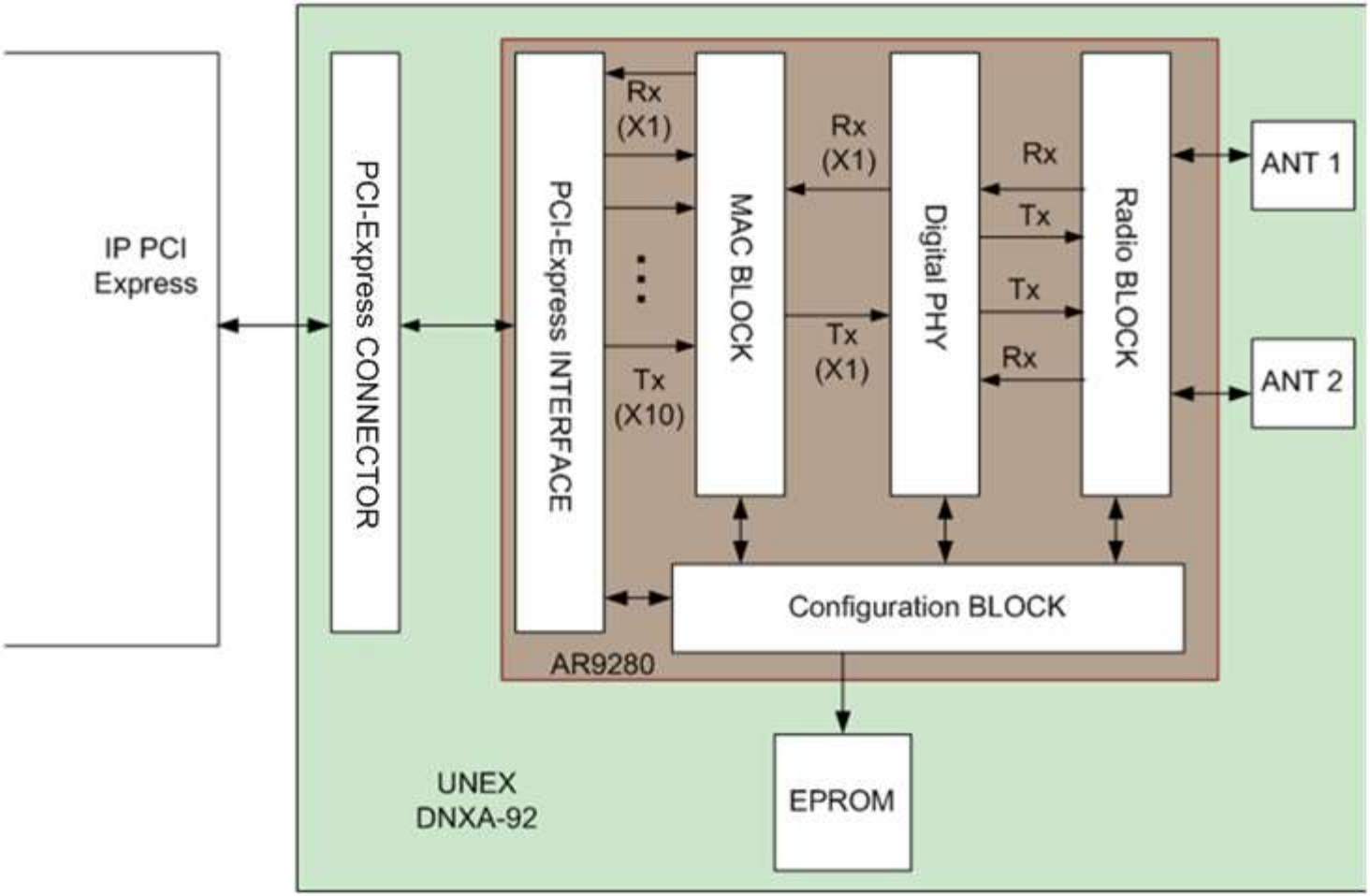
Architecture IP PCIe



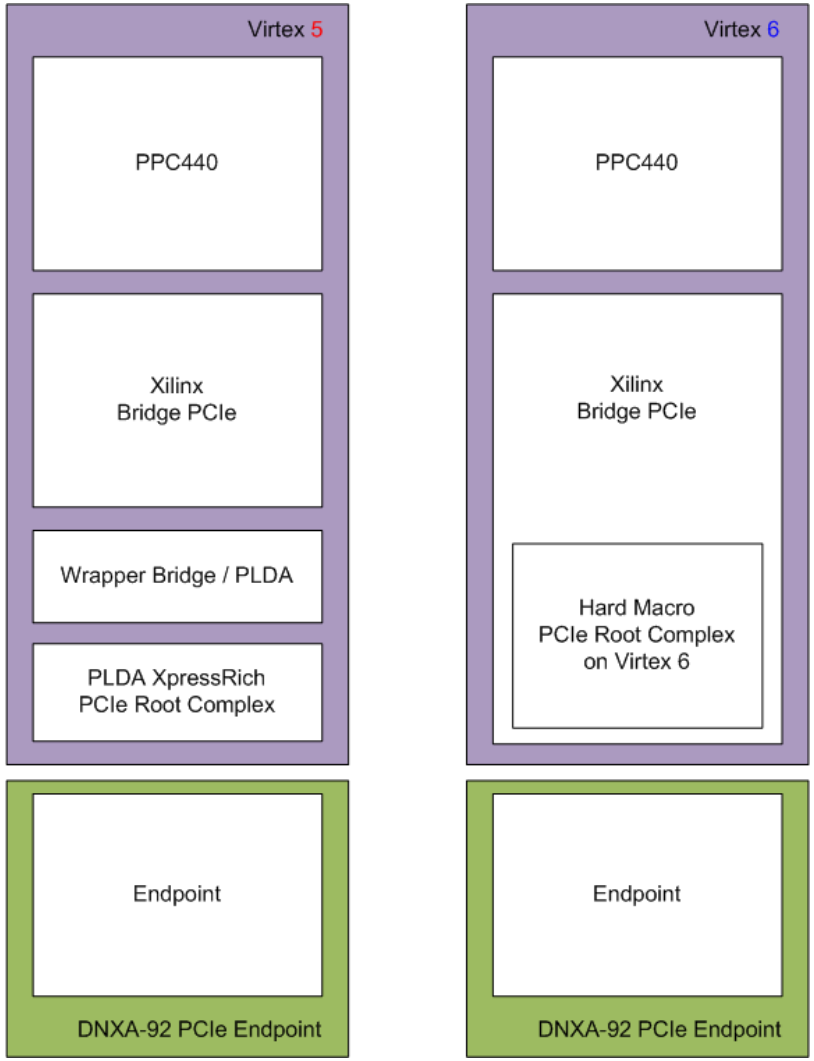
Encryptage


wpa_cli
GUI frontend
frontend control interface
wpa_supplicant
ctrl i/f
configuration
WPA/WPA2
state machine
crypto
TLS
EAPOL and
pre-auth
ethertypes
from/to kernel
l2_packet
event
loop
EAPOL
state machine
EAP methods
EAP-TLS
EAP-MD5
EAP-PEAP
EAP-TTLS
EAP-GTC
EAP-OTP
EAP-SIM
EAP-AKA
EAP-PSK
LEAP
EAP-PAX
EAP-FAST
EAP-MSCHAPv2
driver events
EAP
state machine
driver i/f
wext
hostap
madwifi
hermes
atmel
ndiswrapper
kernel network device driver

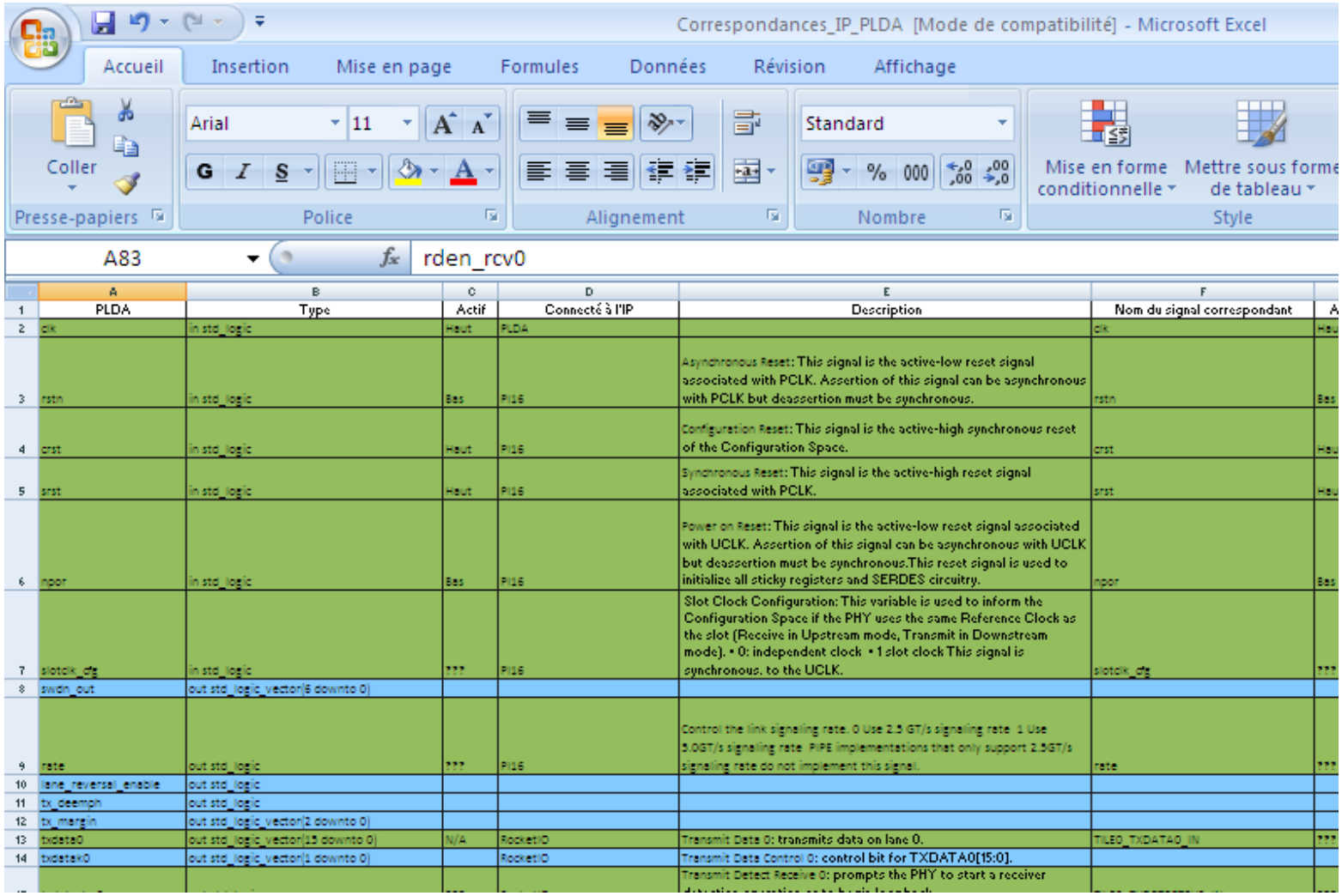
3 – PCIe 2 - Module Wifi PCIe – DNXA-92



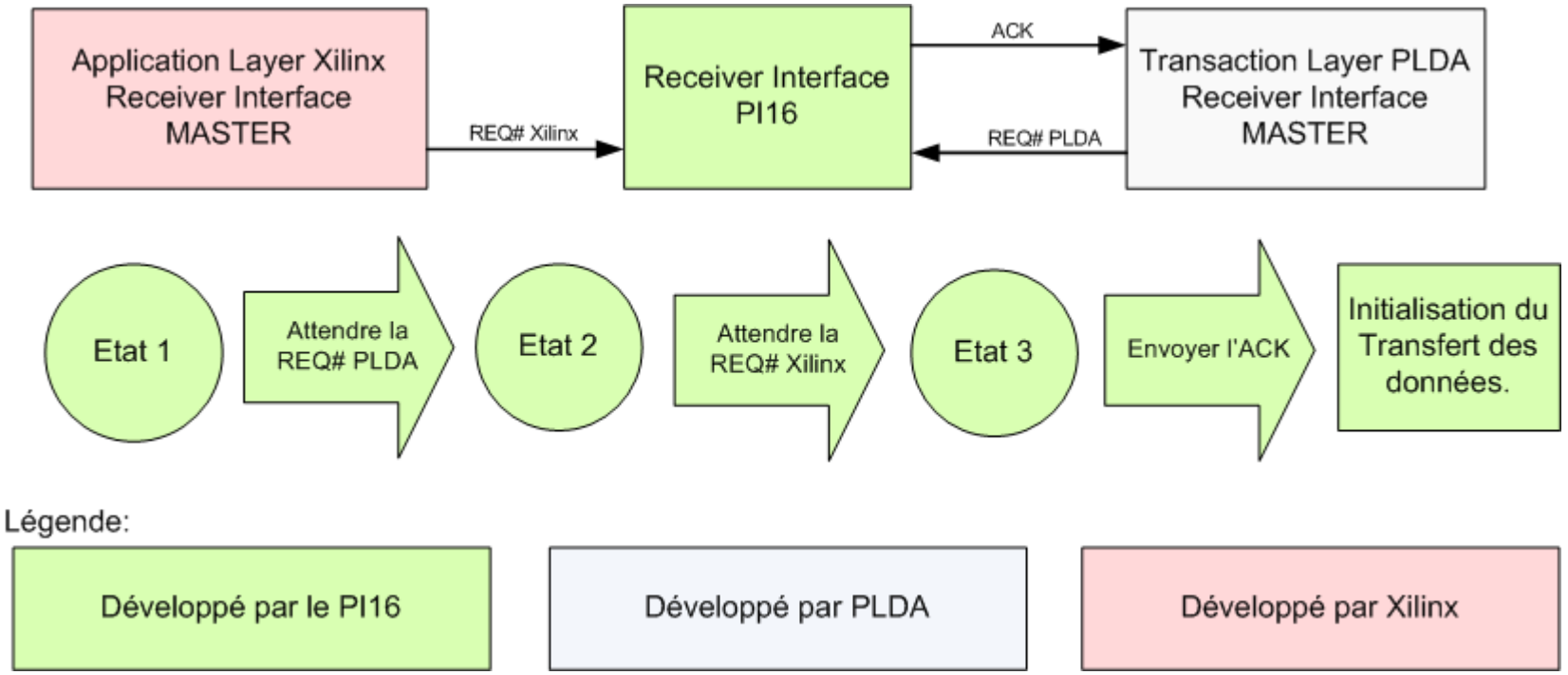
Virtex 5 vs Virtex 6



Correspondances IP PLDA et Bridge Xilinx



	A	B	C	D	E	F
1	PLDA	Type	Actif	Connecté à l'IP	Description	Nom du signal correspondant
2	clk	in std_logic	Haut	PLDA		clk
3	rstn	in std_logic	Bas	PI16	Asynchronous Reset: This signal is the active-low reset signal associated with PCLK. Assertion of this signal can be asynchronous with PCLK but deassertion must be synchronous.	rstn
4	crst	in std_logic	Haut	PI16	Configuration Reset: This signal is the active-high synchronous reset of the Configuration Space.	crst
5	srst	in std_logic	Haut	PI16	Synchronous Reset: This signal is the active-high reset signal associated with PCLK.	srst
6	npor	in std_logic	Bas	PI16	Power on Reset: This signal is the active-low reset signal associated with UCLK. Assertion of this signal can be asynchronous with UCLK but deassertion must be synchronous.This reset signal is used to initialize all sticky registers and SERDES circuitry.	npor
7	slotclk_cfg	in std_logic	???	PI16	Slot Clock Configuration: This variable is used to inform the Configuration Space if the PHY uses the same Reference Clock as the slot (Receive in Upstream mode, Transmit in Downstream mode). • 0: independent clock • 1 slot clock This signal is synchronous. to the UCLK.	slotclk_cfg
8	swdn_out	out std_logic_vector(6 downto 0)				
9	rate	out std_logic	???	PI16	Control the link signaling rate. 0 Use 2.5 GT/s signaling rate 1 Use 5.0GT/s signaling rate PIPE implementations that only support 2.5GT/s signaling rate do not implement this signal.	rate
10	lane_reversal_enable	out std_logic				
11	tx_deemph	out std_logic				
12	tx_margin	out std_logic_vector(2 downto 0)				
13	txdata0	out std_logic_vector(15 downto 0)	N/A	RocketIO	Transmit Data 0: transmits data on lane 0.	TILE0_TXDATA0_IN
14	txdatak0	out std_logic_vector(1 downto 0)		RocketIO	Transmit Data Control 0: control bit for TXDATA0[15:0].	
					Transmit Detect Receive 0: prompts the PHY to start a receiver	


Application Layer Xilinx
Receiver Interface
MASTER
REQ# Xilinx
Receiver Interface
PI16
ACK
REQ# PLDA
Transaction Layer PLDA
Receiver Interface
MASTER
Etat 1
Attendre la
REQ# PLDA
Etat 2
Attendre la
REQ# Xilinx
Etat 3
Envoyer l'ACK
Initialisation du
Transfert des
données.
Légende:
Développé par le PI16
Développé par PLDA
Développé par Xilinx