Une Plate-Forme Légère Reconfigurable Partiellement à Distance

Pierre Bomel, Guy Gogniat, Jean-Philippe Diguet

RenPar'18 / SympA'2008 / CFSE'6
Fribourg, Suisse, du 11 au 13 février 2008

Une Plate-Forme Légère Reconfigurable Partiellement à Distance

Pierre Bomel, Guy Gogniat, Jean-Philippe Diguet

Université de Bretagne Sud,
LESTER, CNRS FRE 2734,
Centre de Recherche,
56321 Lorient Cedex
(pierre.bomel, guy.gogniat, jean-philippe.diguet)@univ-ubs.fr

---

Résumé
Nous présentons une architecture matérielle et logicielle ainsi qu'une implémentation de protocole réseau destinées à la reconfiguration dynamique partielle de FPGA dans le but d'obtenir des systèmes "ultra-légers reconfigurables à distance". Ces systèmes sont destinés aux applications embarquées ayant très peu de ressources matérielles et tirant profit d'architectures dédiées. Ils sont fondés sur un simple FPGA associé à son contrôleur réseau, et ne requièrent pas de mémoires externes pour stocker le code de l'exécutif temps réel, ni les tampons mémoires des protocoles de communication. Nos mesures prouvent que nos systèmes ultra-légers atteignent des débits de reconfiguration au moins dix fois plus rapides que les meilleurs actuels.

Mots-clés : Reconfiguration partielle, serveur de bistreams, FPGA, Ethernet, liaison de données

---

1. Introduction

Les FPGA (Field Programmable Gate Array) sont des circuits reconfigurables qui permettent de concevoir à la demande des systèmes contenant toutes les fonctionnalités requises par une application donnée sans devoir développer un silicium spécifique. Un seul et même FPGA pour de nombreuses applications permet de répondre à deux problèmes majeurs de la conception de circuits : 1) l'explosion des coûts de fabrication de tout nouveau silicium liée à l'évolution des technologies des semi-conducteurs et 2) la possibilité d'upgrader et de corriger aussi bien au niveau matériel que logiciel sur site, donc de prolonger la durée de vie des équipements. Certains modèles de FPGA intègrent des blocs durs spécialisés de type processeurs, mémoires, DSP et interfaces de communication rapides. Ils apportent une extrême flexibilité aussi bien matérielle que logicielle, à grain fin comme à gros grain. Leur reconfiguration dynamique complète permet d'implanter diverses architectures dans le temps et accroît leur taux de réutilisation. Enfin, l'évolution des FPGA est telle qu'à présent les circuits Virtex de la société Xilinx possèdent la propriété de pouvoir être reconfigurés partiellement de façon externe (exo-reconfiguration) ou interne (endo-reconfiguration). La reconfiguration dynamique partielle (RDP) nécessite néanmoins que les diverses configurations partielles (fichiers communément nommés "bitstreams") soient stockées quelque part à l'extérieur du circuit. La logique voudrait que ce lieu de stockage soit accessible par plusieurs FPGA de façon à mutualiser les ressources de stockage. A ce jour les travaux exploitent majoritairement des mémoires locales de type FLASH assistées, dans le meilleur des cas pour réduire la taille de la mémoire, par un décompresseur matériel [1]. On assiste donc à la migration de mm^2 de silicium des FPGA vers des mémoires FLASH. Le taux d'utilisation de ces dernières est extrêmement faible, de l'ordre de 10^{-9}, car elles ne sont sollicitées que lors de la remise à zéro du système. Enfin, bien que le coût de ces mémoires soit évidemment inférieur à celui des FPGA, le bilan en terme de réduction de surface silicium, de surface du circuit imprimé, de consommation et de moyenne des temps de bon fonctionnement est négatif.

Face à cette contradiction technique, nous proposons un modèle de gestion efficace de la RDP qui exploite la tendance forte des systèmes embarqués à être reliés à un réseau de communication d'un ma-

nière ou d'une autre. De plus, l'évolution des réseaux locaux de communication est telle que des dizaines, voire des centaines de Mbit/s, sont maintenant disponibles à des coûts très faibles et avec des taux d'erreurs quasiment nuls. Notre modèle repose donc sur l'accès, via un réseau local, à un serveur distant de bitstreams au niveau liaison de données (niveau 2) de la pile OSI. Le serveur se trouvant sur le même réseau local que le système ultra-léger nous n'avons pas besoin d'une fonction de routage. La couche IP n'est donc pas indispensable. L'accès partagé via un réseau de communication standard tel que Ethernet, Wifi, ou CAN, à une unité externe de stockage de bitstreams permet ainsi à un nombre quelconque de FPGA de télécharger dynamiquement des bistreams. Le stockage sur disque dur des bitstreams remplace avantageusement les nombreuses mémoires FLASH locales grâce à son immense capacité et son faible coût. Cette délocalisation du stockage des bitstreams rend possible la conception de systèmes ultra-légers reconfigurables partiellement.

Dans ce papier nous présentons dans quelles conditions et avec quelles performances obtenues cette hypothèse nous permet de construire des systèmes RDP ultra-légers. Les résultats expérimentaux que nous obtenons prouvent que nous obtenons des débits au moins dix fois plus rapides que les meilleurs actuels tout en ayant une empreinte mémoire si faible que l'usage de mémoires externes pour stocker l'exécutif temps réel et les tampons des protocoles de communication n'est plus indispensable.

Nous passons en revue en section 2 les travaux relatifs à la RDP via un réseau de communication standard. Dans la section 3 nous présentons notre contribution en terme de matériel et de logiciel de base embarqué et proposons un protocole de communication adapté aux contraintes de la RDP. Dans la section 4 nous décrivons nos expériences et présentons nos résultats de mesures concernant la RDP sur la base d'IP de traitement du signal représentatives de la complexité des systèmes visés. Enfin, en section 5, nous concluons et présentons les extensions que nous envisageons d'apporter à nos travaux dans un avenir proche.

2. Travaux relatifs au sujet

Dans cette section nous présentons les travaux relatifs à la RDP via un réseau de communication standard de type Ethernet. A ce jour, il semble qu'aucun autre réseau n'ait été utilisé à cette fin. Pourtant le réseau Wifi pour la bureautique nomade et le réseau CAN pour l'automobile nous semblent être d'excellents candidats alternatifs.

La communauté de la RDP s'accorde sur le fait que, dans les domaines applicatifs ayant de fortes contraintes temps-réel, la durée de la RDP est un des aspects critiques de sa mise en oeuvre. Si elle n'est pas suffisamment brève, l'intérêt de la RDP pour bâtir des systèmes performants est compromis. Walder [2] confirme cela dans le domaine du "wearable computing" et conclut que le temps de RDP peut être négligé dans son modèle d'ordonnancement en ligne si le temps de chargement est négligeable par rapport au temps de calcul des applications. L'extrême rapidité de la RDP est donc une hypothèse forte qui conditionne de nombreux travaux. Les chercheurs investiguent deux approches pour optimiser le temps de RDP. Ce sont la réduction de la taille des bistreams et l'accélération du débit du chargement. La première approche repose sur les outils et les méthodologies de conception de circuits "hors ligne" (flots de conception Module Based et Difference Based de Xilinx [3]) alors que la deuxième est une approche de type logiciel embarqué donc "en ligne". Dans ce papier, nous nous concentrons sur l'endo-reconfiguration dynamique partielle tout en reconnaissant que la première approche est bien évidement nécessaire et complémentaire. A l'issue de ces travaux, nous proposons un démonstrateur constitué d'une couche logicielle minimale de bas niveau qui abstrait l'accès aux ressources matérielles de RDP et met à la disposition des couches logicielles supérieures la gestion complète, transparente et efficace de la fonction de RDP. Par rapport aux derniers modèles en couches d'environnements d'exécution proposés par Kettelhoit [4] et Dittmann [5] notre implantation se situe respectivement au niveau des couches "Configuration layer" et "Execution Environment". De petite taille (quelques centaines de lignes), écrite en C et en VHDL avec des outils standards, cette implantation a pour vocation d'être aisément intégrable dans tout environnement.

La reconfiguration partielle des FPGA de Xilinx passe par la maîtrise du port de configuration nommé ICAP (Internal Configuration Access Port). Les Virtex2 PRO et Virtex4 VFX contiennent ce port et de un à deux coeurs de PPC405 (bientôt les Virtex5 intègreront aussi un coeur de PPC). L'ICAP est accessible à tout processeur, y compris le Microblaze synthétisé, via le composant HWICAP qui l'encapsule et lui associe une interface avec le bus OPB pour un coût de 150 slices et une BRAM. Le débit crête de reconfiguration annoncé est d'un octet par cycle d'horloge soit 100 Mo/s pour un système qui fonctionne à 100 MHz. Etant donné que les systèmes peuvent avoir des fréquences d'horloge différentes, nous choisissons comme unité de mesure le nombre de bits par seconde et par MHz. Ainsi l'ICAP a-t-il une bande passante annoncée de 1 Mo/s.MHz (1 Mega-octet par seconde et par MHz), soit 8 Mbit/s.MHz (8 Megabits par seconde et par MHz).

Claus [6] considère que, pour des applications vidéo temps-réel telles que l'assistance à la conduite de véhicules, la taille des bistreams est de l'ordre de 300 Ko. La nature adaptative des traitements implique de changer dynamiquement d'algorithme sans perdre aucune image (640x480 pixels, VGA, noir et blanc). Dans ces conditions il accepte que le temps de RDP ne dépasse pas 1/8ème du temps calcul alloué à chaque image. Pour un débit de 25 images/s, soit 40 ms par image, cela représente un délai de 5 ms. Transmettre 300 Ko en 5 ms fixe la contrainte de débit à 60 Mo/s. L'ICAP suffit donc. La plateforme repose sur un Virtex2 dans lequel un des PPC405 exécute le code logiciel qui gère la RDP. Le texte de l'article laisse penser que Claus n'a pas obtenu un système fonctionnel le jour de la publication puisque la gestion de l'ICAP est annoncée comme faisant partie de travaux futurs. Cela illustre néanmoins la faisabilité de systèmes temps-réel complexes, à condition que la RDP soit suffisamment rapide.

Non strictement destinée à la RDP, la note d'application XAPP433 [7] de Xilinx décrit un système construit autour d'un Virtex4 FX12 à 100 MHz. Il contient un Microblaze qui exécute le code d'un serveur HTTP qui permet le téléchargement de fichiers via un réseau Ethernet à 100 Mbit/s. La pile de protocoles utilisée est la pile lwIP [8] de Dunkels et l'OS est XMK de Xilinx. Une mémoire externe de 64 Mo, très probablement sous-utilisée, est nécessaire pour stocker les tampons mémoire de lwIP. Le débit annoncé est de 500 Ko/s, soit 40 Kbit/s.MHz . Ce débit est 200 fois moindre que le débit de l'ICAP.

Lagger [9] propose le système ROPES (reconfigurable Object for Pervasive Systems) dédié à l'accélération de fonctions de cryptographie (RC4, DES, triple DES). Il est construit autour d'un Virtex2 1000 à 27 MHz. Le processeur est un Microblaze qui exécute le code de uClinux et télécharge les bitstreams via Ethernet via les protocoles HTTP ou FTP au dessus de la pile TCP/IP/Ethernet. Pour des bistreams de tailles d'environ 70 Ko les temps de chargement sont de 2380 ms pour HTTP et de 1200 ms pour FTP. Le débit obtenu est de l'ordre de 30 à 60 Ko/s, soit d'au maximum 17 Kbit/s.MHz .

Enfin, Williams [10] propose d'utiliser uClinux comme plate-forme de RDP. Pour cela , il a développé un pilote de périphérique en mode caractère au dessus de l'ICAP. Ce pilote permet de charger le contenu des bistreams d'une provenance quelconque, et en particulier à partir d'un système de fichiers distant monté via NFS. La plate-forme matérielle est de type Virtex2. Le processeur est un Microblaze qui exécute le code de uClinux et télécharge les bistreams via la pile de protocoles NFS/UDP/IP/Ethernet. Williams reconnaît que le niveau de service offert par uClinux a un prix en terme de performances qu'il accepte, sans être plus précis. Il ne cite aucune mesure de performances. Pour obtenir une référence dans un tel contexte, nous avons choisi de faire quelques mesures de transfert de fichiers avec FTP d'un serveur distant vers un Virtex2 PRO. Nous avons utilisé pour cela un PC sous Linux (DELL PWS 360, Pentium 4, 3 GHz, 2 Go de mémoire) et notre carte XUP. Le FPGA concerné est un Virtex2 PRO contenant deux PPC405. Un des PPC405 exécute le code de Linux Montavista. Aucune des deux machines n'était chargée par un calcul. Le fichier était de grande taille (20 Mo) et localisé sur le disque dur du portable. Enfin, les deux machines étaient connectées au même HUB Ethernet 100 Mbit/s. Nous avons obtenu des performances comprises entre 200 Ko/s et 400 Ko/s dans ces conditions extrêmement favorables, soit un maximum de 32 Kbit/s.MHz .

3. Contribution

L'état de l'art qui précède nous permet de constater que la configuration "Microblaze + Linux + TCP/IP" domine. Mais elle est aussi source d'inefficacité car les mesures prouvent que les débits obtenus sont au moins deux ordres de grandeur inférieurs au débit crête de l'ICAP, et que les quantités de mémoires externes requises sont de l'ordre de plusieurs dizaines de Mo. Bien que le débit d'un réseau ne puisse rivaliser avec celui d'une mémoire, un tel écart de débit et une telle consommation de mémoires externes nous semble excessifs pour le service fourni. Lors de nos expériences nous avons identifié trois causes principales d'inefficacité. Tout d'abord Linux et la pile TCP/IP ne peuvent fonctionner sans une grande mémoire externe pour stocker le code du noyau ainsi que tous les tampons requis par les protocoles. Ensuite parce que la nature des protocoles employés (destinés dans les années 80 à des liaisons de données moins rapides et moins fiables que maintenant) est telle que des débits maximaux de seulement quelques centaines de Ko/s peuvent être atteints. Enfin parce que nos mesures comparatives de calcul intensif à l'aide de FFT prouvent que le processeur Microblaze est environ 4 fois moins rapide que le PPC405 à la même fréquence. Taille mémoire excessive, débit médiocre du à des protocoles inadaptés et processeur inefficace sont les principaux défauts que notre modèle de système ultra-léger tente de corriger.
Dans cette section nous présentons notre contribution en termes de modèles d'architectures matérielle, logicielle et de protocole. Nous décrivons en détail les points essentiels qui améliorent les performances.

3.1. Architecture matérielle

L'architecture matérielle (Fig. 1) que nous proposons repose sur un Virtex2 PRO 30 cadencé à 100 MHz. Un coeur de PPC405 exécute le code de l'exécutif de téléchargement dynamique. Les applications visées sont du domaine des systèmes adaptatifs ultra-légers, du type traitement d'images pour aide au pilotage de véhicules, robotique mobile, ou chargement dynamique de protocoles pour la radio-logicielle multi-standards. Nous considérons que les IP chargées communiquent avec le reste du système via un sous-ensemble fixe des pads. Le FPGA représente alors un sous-ensemble de composants électroniques dont les fonctions peuvent être très rapidement modifiées par téléchargement. La communication des IP avec le PPC405 et ou l'environnement du FPGA sortent du cadre de cet article. Cependant, elle peut être assurée par la création de wrappers PLB ou OPB autour des IP reconfigurables, via des bus macros ou via les pads connectés à un crossbar externe, tout comme dans la machine ESM [11].

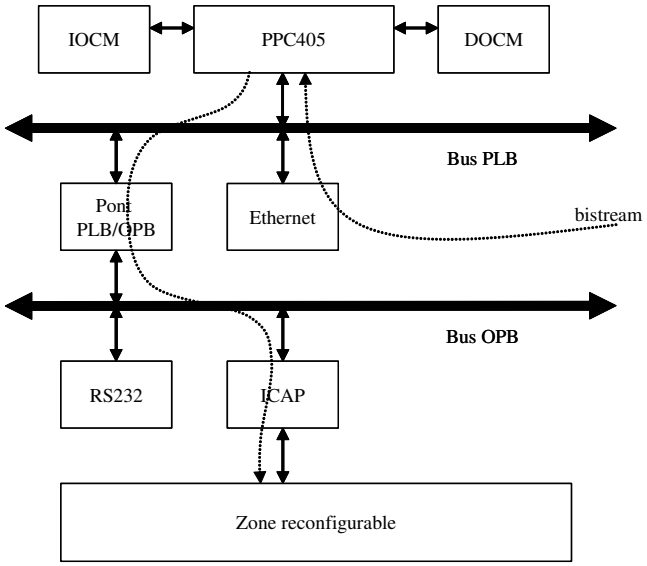

FIG. 1 – Architecture matérielle - Chemin suivi par le bistream du contrôleur Ethernet jusqu'à l'ICAP

Avec l'outil EDK nous avons spécifié un système qui contient un PPC405 et les périphériques strictement nécessaires. Le PPC405 ayant une architecture Harward, nous lui avons adjoint deux mémoires pour stocker le code et les données. Ce sont la mémoire programme ou IOCM (Instruction On Chip Memory) et données ou DOCM (Data On Chip memory). Le PPC405 communique avec les périphériques via deux bus reliés par un pont. Ce sont le bus PLB (Processor Local Bus) pour les périphériques "rapides" et le bus OPB (On Chip Peripheral Bus) pour les périphériques "lents". Le contrôleur Ethernet PHY est connecté sur le bus PLB. La liaison série, pour l'instrumentation et les traces, est connectée au bus OPB. Enfin L'ICAP pilote le contenu des zones reconfigurables. Il est d'ailleurs regrettable qu'il ne soit pas accessible via le bus PLB car seule la version OPB est disponible. Le contrôleur Ethernet est un LXT972A d'Intel. L'exo-reconfiguration complète ou partielle du FPGA peut être faite via le port JTAG et l'endo-reconfiguration via le port ICAP.

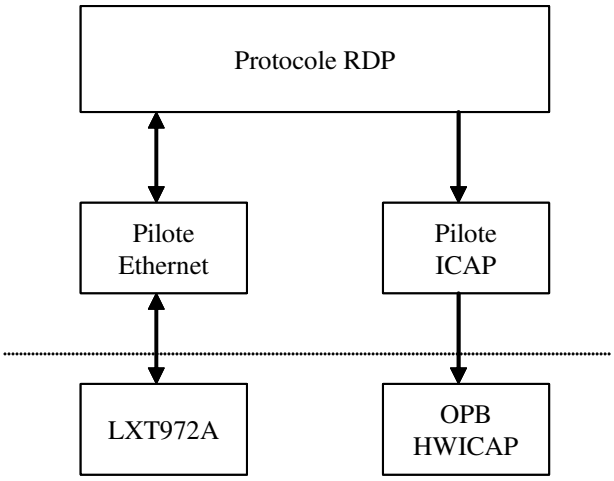

FIG. 2 – Architecture logicielle - Pile de protocoles pour la RDP

3.2. Architecture logicielle

L'architecture logicielle (Fig. 2) est composée de trois parties qui sont le pilote de l'ICAP, le pilote du contrôleur Ethernet et enfin le traitement du protocole de RDP. Le but recherché est d'éliminer au maximum l'empilement de couches logicielles, donc de travailler au niveau le plus bas de la pile OSI c'est à dire le niveau 2 "liaison de donnée". La nature du protocole RDP est une des sources de gain en performance. Un module de mesure du temps et l'accès à la liaison série via la bibliothèque d'entrées-sorties fournie par Xilinx sont aussi utilisés mais non représentés sur la figure car leur usage est marginal. Cet exécutif établit un pipeline de données aussi efficace que possible entre le contrôleur Ethernet et l'ICAP. Nous cherchons à tendre vers l'optimum de 100 Mbit/s qui est le débit maximal du contrôleur Ethernet.

Pour implanter le chargement de l'ICAP et la communication via Ethernet nous avons conçu un système de type producteur-consommateur entre le contrôleur Ethernet et l'ICAP. Ainsi, un tampon circulaire de paquets est-il rempli de façon asynchrone par le gestionnaire d'interruption (handler) du contrôleur Ethernet et vidé au plus vite par le protocole de RPD. Le protocole de RDP s'exécute en tache de fond alors que les gestionnaire d'interruptions sont prioritaires. Pour permetter ce découplage des taches, cette réception de paquets est faite par rafales de taille au maximum égale à la moitié de la capacité du buffer. Le protocole RDP s'exécute en parallèle (entre deux interruptions) et transfère les paquets reçus du tampon dans l'ICAP avant de lancer la reconfiguration partielle. Le dimensionnement du tampon intermédiaire est un point critique qui permet de faire fonctionner simultanément la réception des paquets et la reconfiguration via l'ICAP. Le nombre maximal de paquets d'une rafale dépend des res-

sources mémoires disponibles et le protocole de RDP proposé supporte des configurations mémoires différentes et même variables dans le temps afin d'adapter le débit aux ressources disponibles à l'instant du chargement. L'objectif est bien sur d'allouer le tampon de taille minimale qui assure le débit maximal.

3.3. Liaison de données avec Ethernet

Ethernet, crée par R. Metcalfe au Xerox Parc dans les années 1970, est devenu un ensemble de technologies de communication pour réseaux locaux entre ordinateurs. De type CSMA-CD, il repose sur l'envoi de paquets de données sur un medium partagé et sur la détection de collisions en cas d'émissions simultanées. L'insertion de HUBs pour simplifier le câblage et améliorer la qualité de service, transforme le réseau en une succession de liaisons point-à-point interconnectées par des équipements. Dans ces conditions, deux équipements reliées au même HUB communiquent au travers d'une liaison physique quasiment non partagée. C'est dans ce contexte que nous avons testé le débit maximal atteignable au niveau liaison de données entre notre PC portable et la carte XUP. Nous avons donc programmé sur le PC une application qui envoi des paquets le plus vite possible à destination de la carte XUP, sans aucun protocole de contrôle de flux, ni de détection d'erreurs. Pour nos essais de transmission de paquets Ethernet nous avons utilisé les trames standards telles que spécifiées par la norme 802.3. Le champ EtherType contient la longueur du paquet de données et ne spécifie pas un protocole de niveau supérieur. L'accès direct au niveau MAC se fait sous Linux à l'aide des sockets de type "raw". Ce test montre qu'un débit limite de 60 Mbit/s lié aux performances du PC est atteignable. L'absence d'erreurs de transmission pendant des semaines de test prouve que le niveau de qualité de la liaison de donnée est si élevé qu'il n'y a aucune raison d'implanter dans ce contexte un mécanisme de correction d'erreurs sophistiqué.

3.4. Protocole adaptatif de RDP

Notre protocole se situe au niveau 2 de la pile OSI et implémente une liaison de données avec détection d'erreurs et contrôle de flux. L'adaptativité de ce protocole correspond à la faculté qu'il a de s'adapter aux ressources mémoires disponibles sur la cible FPGA. En cas d'erreur de transmission, la reconfiguration dynamique est instantanément arrêtée après signalisation à l'émetteur de l'erreur. Pour cela, le contrôleur Ethernet détecte tout paquet mal transmis et la numérotation en séquence de 1 à N des paquets permet de détecter tout paquet manquant, dupliqué, ou déplacé dans le flux. L'extrême rareté des erreurs de transmission nous a conduit à choisir une stratégie de rupture immédiate de la communication au niveau bistream plutôt que paquet.

Le serveur de bistream, un pentium4 à 2 GHz, possède plus de puissance de calcul qu'un PPC405 à 100 MHz. Il est donc indispensable de mettre en oeuvre un mécanisme de contrôle de flux pour réguler le flux des paquets. Tout mécanisme de contrôle de flux consiste à envoyer de l'information au serveur. Cette rétroaction suspend le pipeline de transmission de donnée. Il est alors nécessaire d'envoyer aussi peu de paquets de contrôle de flux au serveur que possible. Nous avons choisi d'implanter un système d'acquitement positif tous les P paquets, P étant déterminé par le module "protocole RDP" en fonction des ressources mémoires disponibles à l'instant du chargement.

Le protocole peut être utilisé dans deux modes différents. En mode "maître" ou "auto-reconfiguration" (cadre du haut Fig. 3) le système décide du moment de la reconfiguration et transmet au serveur l'identité du bistream (un nom de fichier dans une arborescence). En mode "esclave" il reçoit directement le fichier sans en connaître l'identité. Le mode "maître" est optionnel.

En début de transmission le serveur envoie le nombre total de paquets N qui seront transmis et la cible répond avec la valeur de P. En début de transmission, et après chaque acquitement positif, le serveur envoie P paquets en rafale puis attend l'acquitement suivant. L'automate de gauche (Fig. 3) décrit le comportement du serveur et celui de droite celui du système cible. Immédiatement après l'identification (mode maître) du bistream, la transmission est constituée de N/P rafales de P paquets jusqu'au Nième paquet, qui termine la session de chargement. En cas de détection d'erreur ou de remise à zéro matérielle le système revient à sa position d'attente du nombre N du protocole. Un "chien de garde" permet aussi de détecter la disparition subite de l'une des extrémités et replace le serveur et/ou le système dans leurs positions d'attente respectives.

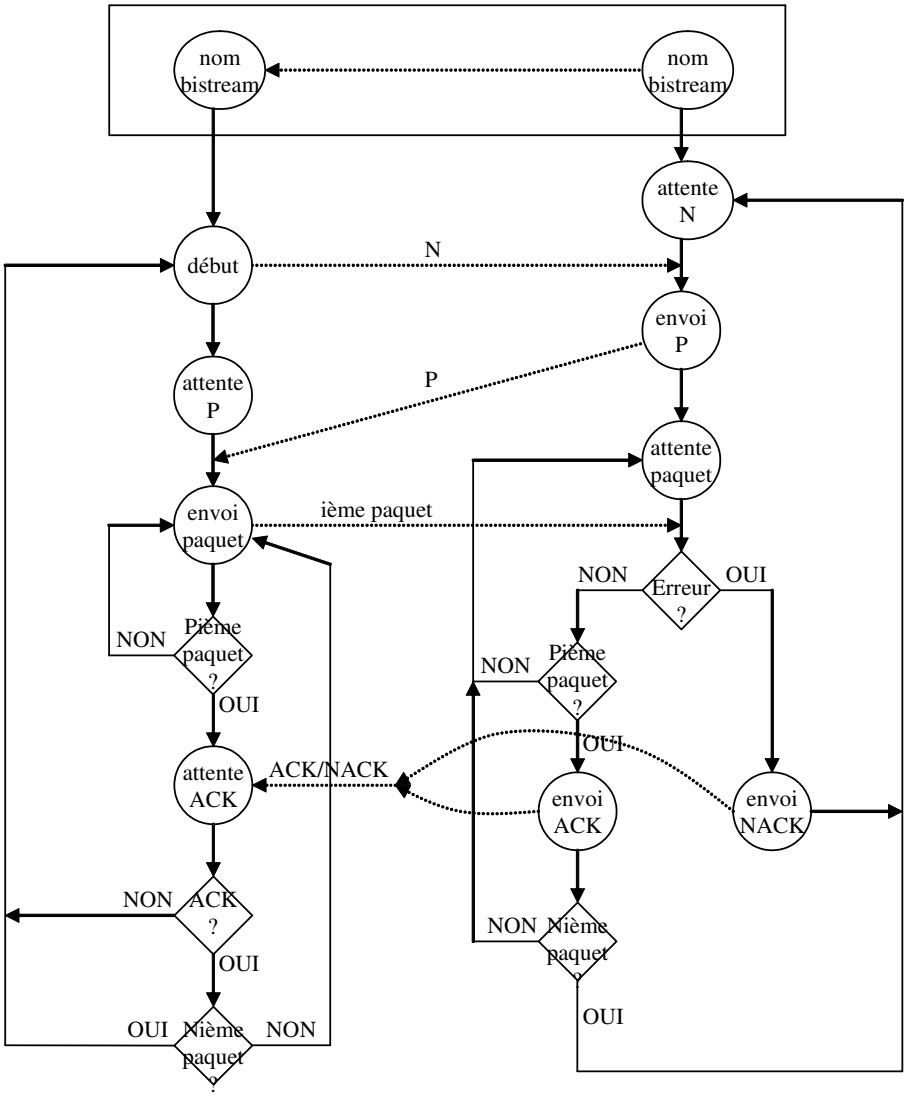

FIG. 3 – Protocole de RDP

4. Expériences et mesures

Nos mesures reposent sur le chargement répétitif d'IPs de cryptographie de type DES, et triple DES dont les tailles de bistreams sont de l'ordre de 60 Ko et 200 Ko. Les mesures obtenues dépendent, comme on peut s'y attendre, de la quantité de mémoire allouée au protocole de RDP pour le stockage des paquets, donc de la valeur de P. La figure 4 présente les résultats expérimentaux obtenus. Les deux courbes du haut (en traits pleins et avec des tirets) représentent respectivement les débits obtenus pour des bistreams de taille 60K et 200K. On remarque dans les deux cas, qu'à partir d'une taille de 3 paquets Ethernet (P=3) un débit maximal situé entre 375 et 400 Kbit/s.MHz est atteint. Les courbes en pointillés représentent les débits moyens obtenus par Xilinx, Lagger et probablement Williams. Notre protocole de RDP ne nécessite donc que très peu de mémoire pour stocker les 3 paquets Ethernet du buffer tout en procurant un débit plus proche du débit crête de notre réseau Ethernet. Dans ces conditions le rapport entre le débit de reconfiguration effectif et celui de l'ICAP n'est plus que d'un ordre de grandeur au lieu de deux.

Enfin, notre système complet réside dans une mémoire de 32 Ko pour les données et 40 Ko pour le code exécutable. Cela nous permet d'affirmer que notre système est non seulement plus efficace en débit mais qu'il est aussi "ultra-léger" du point de vue des mémoires.

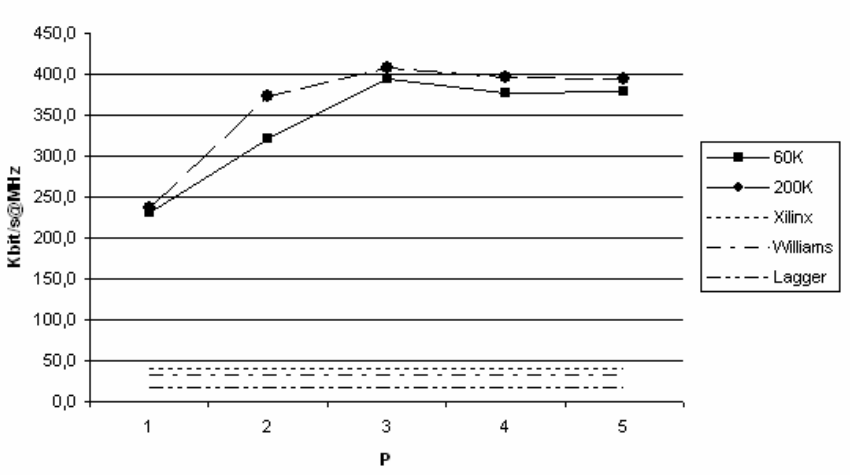

FIG. 4 – Mesures de débit en fonction de P

5. Conclusion et extensions

Par rapport aux travaux de Claus, Xilinx, Lagger et Williams, le débit de chargement que nous avons atteint est au moins d'un ordre de grandeur supérieur au meilleur d'entre eux. Le tableau 1 présente les débit Ethernet exprimées en Kbit/s.MHz ainsi que la taille de l'empreinte mémoire requise par chacune des solutions.

	Lagger[9]	Williams[10]	Xilinx[7]	RDP[auteurs]
Débit Kbit/s.MHz	17	32	40	375-400
Mémoire (octets)	$>$ 1M	$>$ 1M	$>$ 1M	$<$ 100K

TAB. 1 – débits comparatifs

Les extensions possibles consistent à optimiser le protocole RDP en substituant un DMA aux transferts de données faits par le PPC405 afin de tendre vers la bande passante maximale d'Ethernet. Ce DMA aura aussi comme avantage de soulager le processeur du transfert de donnée et donc de permettre une mise en oeuvre de la RDP à l'aide de processeurs moins puissants. Une reprise de LwIP pour en dériver une version "standard" intégrant le protocole de RDP est aussi envisagée. Enfin, une version totalement matérielle du protocole RDP peut s'avérer nécessaire lorsque nous porterons le protocole RDP au dessus de Wifi, CAN ou envisagerons d'utiliser des FPGA sans coeurs de PPC405 mais exploitant des coeurs de Microblaze synthétisés.

Bibliographie

1. "Real-time Configuration Code Decompression for Dynamic FPGA Self-Reconfiguration", Michael Hübner, Michael Ullmann, Florian Weissel, Jürgen Becker, Proceedings of the 18th International Parallel and Distributed Processing Symposium (IPDPS04), 2004.
2. "Online Scheduling for Block-partitioned Reconfigurable Devices", Herbert Walder, Marco Platzner, Proceedings of Design, Automation and Test in Europe (DATE'03), 2003.
3. "Two Flows for Partial Reconfiguration : Module Based or Difference Based", Xilinx, XAPP290, sept. 2004
4. "A Layer Model for Systematically Designing Dynamically Reconfigurable Systems", Boris Kettelhoit, Mario Porrmann, Proceedings of International Conference on Field Programmable Logic and Applications (FPL'06), 2006.
5. "Hard Real-Time Reconfiguration Port Scheduling", Florian Dittmann, Stefan Frank, Proceedings of Design, Automation and Test in Europe (DATE'07), 2007.
6. "Using Partial-Run-Time Reconfigurable Hardware to accelerate Video Processing in Driver Assistance System", Christopher Claus, Johannes Zeppenfeld, Florian Muller, Walter Stechele, Proceedings of Design, Automation and Test in Europe (DATE'07), 2007.
7. "Web Server design using MicroBlaze Soft Processor", Xilinx, XAPP433, 13 octobre 2006.
8. "lwIP", Adam Dunkels, Computer and Networks Architectures (CNA), Swedish Institute of Computer Science, http ://www.sics.se/adam/lwip/.
9. Self-Reconfigurable Pervasive Platform For Cryptographic Application", Arnaud Lagger, Andres Upegui, Eduardo Sanchez, Proceedings of International Conference on Field Programmable Logic and Applications (FPL'06), 2006.
10. "Embedded Linux as a platform for dynamically self-reconfiguring systems-on-chip", John Williams, Neil Bergmann, Proceedings of the 2004 International Conference on Engineering of Reconfigurable Systems and Algorithms (ERSA'04), ISBN : 1-932415-42-4, 2004.
11. "The Erlangen Slot Machine : Increasing Flexibility in FPGA-Based Reconfigurable Platforms", C. Bobda, M. Majer, A. Ahmadinia, T. Haller, A. Linarth, J. Teich, Journal of VLSI Signal Processing Systems, Volume 47, Issue 1 (April 2007), Pages : 15 - 31, ISSN :0922-5773.