THESE DE DOCTORAT DE L'UNIVERSITE PARIS VI

Spécialité :

INFORMATIQUE

Présentée par :

M. Alexandru STOICA

pour obtenir le titre de DOCTEUR de L'UNIVERSITE PARIS VI

Sujet de la thèse :

ETUDE D'UNE ARCHITECTURE DE COMMUTATEUR ATM HAUT DEBIT AVEC RESPECT DE LA QUALITE DE SERVICE

soutenue le 3 mai 2002,

devant le jury composé de :

M. Alain GREINER	Directeur de Thèse
Mme. Caroline FAYET	Rapporteur
M. François ANCEAU	Rapporteur
M. Serge FDIDA	Examinateur
M. Jean Pierre GAUTHIER	Examinateur
M. Michel LEMONIER	Examinateur

Remerciements

Je teins à remercier chacun des membres du jury, pour l'intérêt qu'ils ont accordé à mon activité de recherche. Je remercie à mon directeur de thèse, Monsieur professeur Alain Greiner, pour avoir accepté la responsabilité de l'encadrement de ma thèse, pour sa disponibilité, pour ses conseils et ses critiques. Je veux exprimer ma reconnaissance à Monsieur Jean Pierre Gauthier pour avoir accepté de partager avec moi sa compétence et pour avoir soutenu notre projet vis à vis de CS Telecom. Je remercie aux dirigeants du Groupe Compagnie des Signaux et à Monsieur Michel Lemonier en particulier pour l'initiation du projet et pour le financement de mon activité de recherche. Je suis reconnaissant à mon pays, la Roumanie, aux professeurs de l'Institut Polytechnique de Bucarest ou j'ai été formé en tant qu'ingénieur. Je remercie également à toute l'équipe du laboratoire LIP6-ASIM et particulièrement à Roseline Chotin. Je voudrais pas oublier de remercier à mes parents Paul et Maria et ma femme Madalina pour leurs encouragement et leur soutien moral.

Résumé

La garantie de la qualité de service dans les réseaux haut débit est devenu dans ces dernières années un enjeu majeur de l'industrie des télécommunication. Notre recherche porte sur les architectures d'équipements ATM et vise la conception de commutateurs possédant une valence élevée (bande passante et nombre de connexions) et qui garantissent une qualité de service supérieure.

Le mémoire est divisé en deux parties :

Une première partie étudie comment améliorer une architecture existante de commutateur ATM, construite autour du bus à réservation de bande passante (et utilisant donc une technique de commutation de circuit). Cette architecture permet de fournir des garanties absolues en terme de débit, et l'étude cherche à augmenter la bande passante et le nombre de connexions de cette architecture qui possède de réels atouts en ce qui concerne la qualité de service.

La deuxième partie du mémoire étudie une architecture de commutateur dont le dispositif de commutation est une matrice utilisant la commutation de paquets (c'est à dire sans réservation de bande passante). Dans ce type d'architecture, la cause principale de la dégradation de la qualité de service est le goulot d'étranglement représenté par la matrice de commutation de paquets. Les matrices de commutation de paquets ne sont pas à priori conçues pour gérer des priorités. La performance en termes de qualité de services d'une architecture de ce type dépend essentiellement des stratégies utilisées pour l'ordonnancement des files d'attente situées au niveau des ports d'entrée. L'étude propose une architecture de commutation basée sur l'envoi de notifications qui sert à réaliser un triage du trafic entrant en vue d'une amélioration de la qualité de service globale de l'architecture. Les performances d'une architecture générique à commutation de paquets utilisant cette technique ont été évaluées à l'aide d'un simulateur.

Abstract

The guarantee of the quality of services of the high throughput networks became in the recent years a major objective of the telecommunications industry. Our research is oriented on ATM switching architectures and aims the conception of switches characterized by an elevated valence (throughput and number of connections) and which guarantee a superior quality of services.

The thesis is divided in two parts:

In the first part we have studied how to improve an existing switching ATM architecture, constructed around a throughput reservation bus (and using therefore the technique of circuit commutation). This architecture allows to provide absolute guarantees in term of throughput, and we looked forward to increase the throughput and the number of connections of this architecture who possesses real assets concerning the quality of services.

The second part of the thesis studies an architecture in which the switching device use the packets commutation principle (meaning therefore that throughput reservation isn't used). Concerning this type of architecture, the main reason of the deterioration of the quality of services is the bottleneck represented by the packets commutation matrix. The matrixes using packets commutation are priori not conceived to manage priorities. The performance in terms of quality of services of this type of architecture depends essentially of the scheduling strategies used for the queues situated at the input ports. We suggest a switching architecture based on the consignment of notifications used to sort the incoming traffic in order to improve the global quality of service of this architecture. The performances of a generic switching architecture with packet commutation using this technique was evaluated by simulation.

Sommaire

Liste des figures

Liste des figures

Liste des tableaux

Chapitre 1. Problématique

Ce chapitre a pour but de définir les objectifs de la recherche et de poser les problèmes que l'étude aura à résoudre.

1.1 Introduction

Les architectes des réseaux de télécommunication haut débit posent de plus en plus souvent le problème de la qualité de service des connexions définies à travers un réseau. L'ATM (*Asynchronous Transfer Mode*) est le protocole le plus utilisé pour les RNIS (Réseaux Numériques avec Intégration de Services), protocole qui permet le transport d'une large variété de services. Notre recherche porte sur l'architecture interne des commutateurs de réseaux qui utilisent le protocole ATM et vise deux objectifs:

- la réalisation de commutateurs possédant une valence élevée (ce qui impose une large bande passante, et le support d'un grand nombre de connexions) ;
- il est souhaité que cette augmentation de la valence ne dégrade pas les garanties de service.

1.2 Le contexte : la nécessité de garantir la qualité de service

1.2.1 Le concept de qualité de service

Le trafic à travers un réseau comporte plusieurs types de données. La concurrence de ces types de données pose des problèmes sur la priorité à attribuer à chaque catégorie. Les applications multimédia sont généralement gourmandes en termes de débit et de latence en raison des exigences sur la qualité de l'image et du son. En même temps, pour certains transferts de données « classiques », une latence réduite est demandé.

Du point de vue de l'utilisateur, le réseau doit lui fournir des services selon les caractéristiques prévues dans son contrat. La notion de qualité de services ou QoS (*Quality of Service*) définit la capacité d'un réseau à fournir des services respectant les caractéristiques spécifiées par le contrat de chaque client du réseau. La QoS doit répondre aux besoins et aux exigences du client, quelque soit la technologie utilisée pour l'acheminent des données : Frame Relay, ATM, Ethernet ou IP. Des concepts comme la priorité de service, les classes de service ou finalement la qualité de service sont issus de la nécessité de transporter des données caractérisées par des contraintes différentes à l'aide d'un réseau unique.

Le développement des techniques de transport numérique de l'information vidéo, audio et le trafic croissant de données de toute nature exigent de la part de la recherche et de l'industrie de télécommunication de développer des architectures qui soient capables de garantir une qualité de service pour des débits de plus en plus élevés.

1.2.2 Quelques paramètres de la qualité de service

La QoS comporte une série de paramètres qui permettent de caractériser les garanties qui peuvent être fournies à chaque flux (ou connexion) transitant par un réseau de télécommunication. Le trafic est différencié jusqu'au niveau connexion. Chaque connexion peut demander une bande passante minimale. L'absence d'un service minimum peut provoquer chez les utilisateurs le « gel » de la navigation sur Internet.

Un paramètre spécifique aux applications temps-réel et interactives est le temps de transmission. Ce paramètre se décompose dans un délai fixe imposé par le temps de propagation dans les équipements, les faisceaux électriques ou optiques et un délai variable nommé « gigue » issu des congestions instantanées lorsque plusieurs flux de données sollicitent au même moment le même port de sortie. Un temps de transmission supérieur à quelques centaines de millisecondes rend une conversation téléphonique désagréable. Des valeurs maximales doivent être garanties afin de pouvoir offrir des services interactifs. Un bon contrôle de la gigue est souhaité dans les applications de type vidéo à la demande. Une gigue élevée provoque des interruptions dans la continuité de l'image et du son d'une

application multimédia. La gigue peut être maîtrisée dans les équipements de télécommunication par la mise en place des jeux de files d'attente.

Un autre paramètre important est lié à la perte de données. La perte de données est générée par la saturation du réseau et le remplissage des tampons présents dans les commutateurs. Pour les application temps réel, la perte de données provoque une altération du son et/ou de l'image, la retransmission ne permettant pas de corriger cette dégradation. Pour les applications non temps réel, la perte de données se traduit par des temps de transfert accrus à cause des retransmissions opérées automatiquement par les couches hautes du protocole. Dans les deux cas le pourcentage de pertes de données doit être minimisé.

Cet étude se concentre sur l'aspect débit de la qualité de service.

1.2.3 La garantie et le coût de la qualité de service

Le concept de qualité de service est global pour un réseau de télécommunication. Le terme utilisé est celui de qualité de service de bout en bout (*End to End QoS*). Trois niveaux de qualité de service sont offerts par un réseau hétérogène:

- le niveau « meilleur effort » (*Best Effort*), correspond à une absence de QoS et n'offre aucune garantie ;
- le niveau « service différentié » fournit une faible QoS (*soft QoS*) et offre des garanties statistiques moyennes pour les débits et les taux de pertes ;
- le niveau « garantie de service » fournit une forte QoS (*hard QoS)* et réserve les ressources du réseau pour des catégories spécifiques de trafic.

Le fait qu'un client choisisse un niveau de qualité de service plutôt qu'un autre est décidé par la nature de son application et de ses moyens.

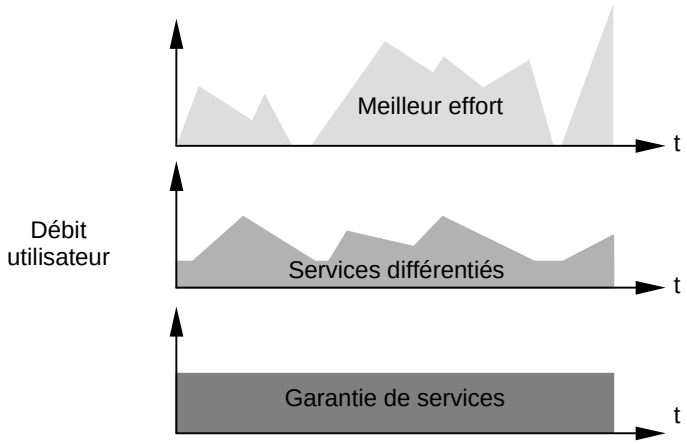

Figure 1.1 - Niveaux de qualité de services offerts par un réseau hétérogène

Les coûts d'implémentation de chaque niveau de service sont sensiblement différents et peuvent varier avec un facteur 10 entre le meilleur effort et la garantie de service. La cause principale de cette différence est liée à la nécessité de réservation des ressources réseau pour garantir une bande passante minimale aux clients. Dans le cas des services différenties ces mêmes ressources sont sur-réservées pour plusieurs clients (*overbooking*).

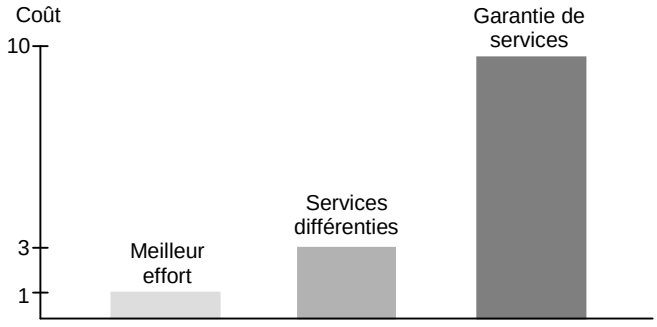

Figure 1.2 - Coûts relatifs des niveaux de qualité de services

1.2.4 L'impact de la QoS sur l'architecture matérielle

La QoS est présente à plusieurs niveaux dans un réseau :

- au niveau global, dans le management du réseau, dans les fonctions de signalisation entre les éléments du réseau et dans les fonctions de contrôle et de comptabilité du trafic

- au niveau des commutateurs dans les techniques de contrôle des files d'attente, dans l'ordonnancement et dans le mise en forme (*shaping*) du trafic.

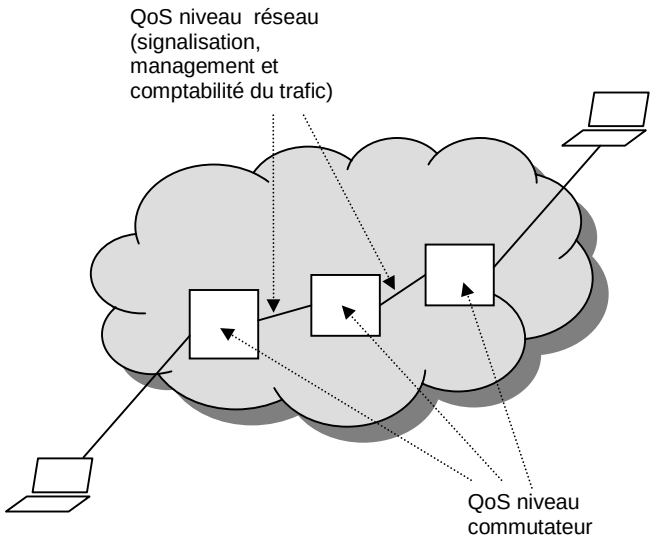

Figure 1.3 - La qualité de service à différents niveaux

Au niveau des commutateurs, la QoS a un fort impact sur l'architecture matérielle. Généralement la QoS demande l'allocation de ressources matérielles. Par exemple, pour faire face à la contention qui apparaît lorsque plusieurs flux entrants doivent sortir par le même port, les ports d'entrée peuvent être équipés d'une simple FIFO, ne faisant pas la distinction entre les différents flux entrant sur ce port. Le fait de posséder un jeux complexe de files d'attente rend le nœud capable de gérer plusieurs niveaux de priorités et d'allouer de la bande passante en fonction des contrats de QoS.

1.3 Objectifs contradictoires: bande passante élevée et garantie de QoS

La croissance des volumes de données transférés imposent aux équipements de fournir une bande passante de plus en plus élevée et, simultanément, les utilisateurs réclament une meilleure qualité de service. Ces deux exigences sont contradictoires : un équipement conçu pour supporter une bande passante élevée doit gérer un nombre important de connexions simultanées (et donc un nombre important de files d'attente : c'est à dire une file d'attente par

flux). Ces deux paramètres (bande passante totale du commutateur et nombre maximal de connexions) sont susceptibles d'évoluer au cours de la vie d'un équipement et il est donc nécessaire que les commutateurs soient extensibles de façon modulaire (c'est la propriété d'extensibilité par ajout de nouveaux modules). Cette exigence d'extensibilité est contradictoire avec une gestion centralisée de files d'attente. De plus, un mécanisme de gestion centralisée des files d'attente introduit un goulot d'étranglement qui peut pénaliser la performance globale. Inversement, un mécanisme de gestion décentralisée des files d'attente rend beaucoup plus difficile la garantie de la qualité de service.

Les architectures de commutateurs qui utilisent une technique de commutation de circuit permettent plus facilement de garantir la qualité de service par rapport aux architectures basées sur la commutation de paquets parce que les premières réalisent en pratique une réservation de bande passante.

Le mécanisme de réservation de bande passante associée à la commutation de circuit a deux inconvénients :

- il limite l'extensibilité des commutateurs, car il suppose généralement un contrôle centralisé, difficilement compatible avec une approche modulaire.
- il a pour conséquence une sous-utilisation globale de la bande passante disponible, lorsqu'un utilisateur n'exploite pas la bande passante réservée.

1.4 Conclusion

L'objectif fixé est de concevoir un équipement possédant une valence élevée (bande passante et nombre de connexions) et garantissant la qualité de service. Cet objectif principal est divisé en deux axes de recherche :

- un axe qui vise un débit important pour une architecture à commutation de circuit et qui permet une qualité de service élevée,
- un axe qui vise la mise à niveau de la qualité de service d'une architecture à commutation de paquets, architecture qui permet d'avoir de débits importants.

1.5 Structure du mémoire

Le mémoire est organisé de la manière suivante :

- le deuxième chapitre analyse quelques exemples d'architectures de commutation prenant en compte les aspects de la contention interne,
- le troisième chapitre présente les principes d'une architecture conçue comme un cluster de sous-systèmes liés par des liaisons haut débit bidirectionnelles,
- le quatrième chapitre étudie la faisabilité de l'architecture cluster en décrivant son composant de base,
- le cinquième chapitre propose une architecture utilisant une matrice de commutation,
- le sixième chapitre décrit un simulateur générique pour un commutateur utilisant une matrice à commutation de paquets,
- le septième chapitre présente et analyse les résultats des simulations effectuées dans le but d'évaluer la performance de l'architecture utilisant une matrice de commutation.

Chapitre 2. Etat de l'art des architectures de commutation ATM

Ce chapitre présente l'architecture générale d'un commutateur ATM et quelques architectures représentatives. Les avantages et les inconvénients de chaque technique de commutation et l'impact sur la qualité de service seront mises en évidence.

2.1 Introduction

La commutation ATM se différencie par rapport aux autres protocoles par le classement des données transférées en plusieurs types de trafic.

Une des particularités du protocole ATM est que les cellules sont transportées d'une manière orientée vers l'établissement de connexions et de canaux virtuels de communication entre la source et la destination. Le format des cellules ATM est défini par le standard ATM. Conformément à ce standard, l'information de l'utilisateur est transférée en utilisant des paquets de dimension fixe, nommés cellules à cinquante-trois octets. L'en-tête de la cellule occupe cinq octets. Les quarante-huit octets restants sont réservés pour l'information utilisateur. En revanche les standards ne définissent pas les techniques de commutation. Chaque commutateur ATM utilise ses propres principes architecturaux. De nombreuses directions de recherche ont été ouvertes pour explorer les diverses possibilités de commutation en vue de l'assurance de la qualité de service.

Le fait de transporter plusieurs types de trafic, chacun avec un débit, une latence et un taux de perte propre, augmente la complexité des algorithmes d'ordonnancement. Plusieurs alternatives de conception des architectures ATM existent également en logiciel et en matériel. Tout logiciel ATM est séparé en une partie destinée à piloter le matériel (*hardware-dependend*) et une partie indépendante de l'architecture. Une conception modulaire du logiciel et du matériel facilite l'évolution d'une architecture.

Quelques architectures matérielles représentatives seront présentées. Chaque technique de commutation a ses mérites et ses inconvénients. Les critères de performances visent le débit,

le nombre de connexions gérées, l'extensibilité, les problèmes de latence et de gigue, et la tolérance aux erreurs. L'analyse de ces techniques permettra de tirer à la fin de ce chapitre quelques conclusions sur la commutation ATM.

2.2 Les fonctions de commutation

Un commutateur ATM contient un ensemble de ports d'entrée et de ports de sortie, à travers lesquels il est connecté à des terminaux ou à d'autres commutateurs. Un système ATM, à part la fonction de commutation, accomplit et intègre une fonction de management. Un commutateur dispose aussi d'interfaces destinées à l'échange des informations de contrôle et de management. Conformément aux standards, un système ATM doit supporter un ensemble de règles de contrôle de trafic. Une description des fonctions liées à la commutation et des règles imposées au contrôle de trafic est nécessaire.

2.2.1 La couche (le plan) utilisateur (*User Plane*)

La fonction principale d'un commutateur ATM est de transférer les données utilisateur d'un port d'entrée vers un port de sortie. Généralement le commutateur exécute des traitements sur l'en-tête des cellules. Les octets utilisateur sont transportés de manière transparente sauf s'il s'agit des cellules OAM (*Operation And Maintenance*) ou si l'équipement effectue des opérations de cryptage/décryptage sur les données utilisateur. Après l'entrée d'une cellule dans le commutateur, l'information contenue dans les champs VPI (*Virtual Path Identifier*) et VCI (*Virtual Channel Identifier*) est utilisée pour router la cellule vers le port de sortie correspondant. Ce processus se déroule à travers trois blocs fonctionnels: le module d'entrée, le module de routage et le module de sortie. Un module de routage constitué par plusieurs éléments devient une matrice de routage.

2.2.2 La couche de contrôle (*Control Plane*)

Cette couche remplit les fonctions liées à l'établissement et au contrôle des connexions VP/VC. Contrairement aux cellules de données de l'utilisateur, le contenu des cellules de contrôle n'est pas transparent au réseau. Le commutateur identifie et traite le contenu des cellules de contrôle (signalisation), et en produit lui-même. Le Contrôleur d'Admission de Connexion (*Connection Admission Control* - CAC) remplit les principales fonctions de signalisation. L'information de signalisation peut ou non passer à travers le commutateur ou être échangée par l'intermédiaire d'un réseau de signalisation.

Le système de commutation doit supporter le contrôle d'admission des cellules, des paramètres utilisateur UPC (*User Parameter Control*), des paramètres réseau NPC (*Network Parameter Control*) et de la congestion. Les fonctions UPC/NPC sont généralement effectuées par les modules d'entrée et les fonctions de contrôle de congestion sont gérés par le *Switch Manager*. Le *Switch Manager* gère aussi l'ordonnancement et l'élimination des cellules dans les files d'attente.

2.2.3 La couche de management

La couche de management est chargée de surveiller et de contrôler le réseau pour assurer une opération correcte et efficace. Les fonctions de cette couche peuvent être divisées en fonctions de management des erreurs, de management de la performance, de configuration, de sécurité du trafic et de management du trafic. Ces fonctions sont effectuées par un bloc fonctionnel nommé *Switch Manager*. Le *Switch Manager* est responsable du déroulement des procédures OAM. Les cellules OAM sont reconnues et traitées par le commutateur ATM et en même temps le commutateur génère lui-même des cellules OAM pour transmettre des informations aux autres commutateurs.

2.3 Architecture générique d'un commutateur

La présentation de l'architecture tient compte de la fonctionnalité. Le commutateur est constitué par des modules d'entrée IM, des modules de sortie OM, une matrice de commutation (*Cell Switching Fabric*), un contrôleur d'admission de connections CAC et un *Switch Manager* SM.

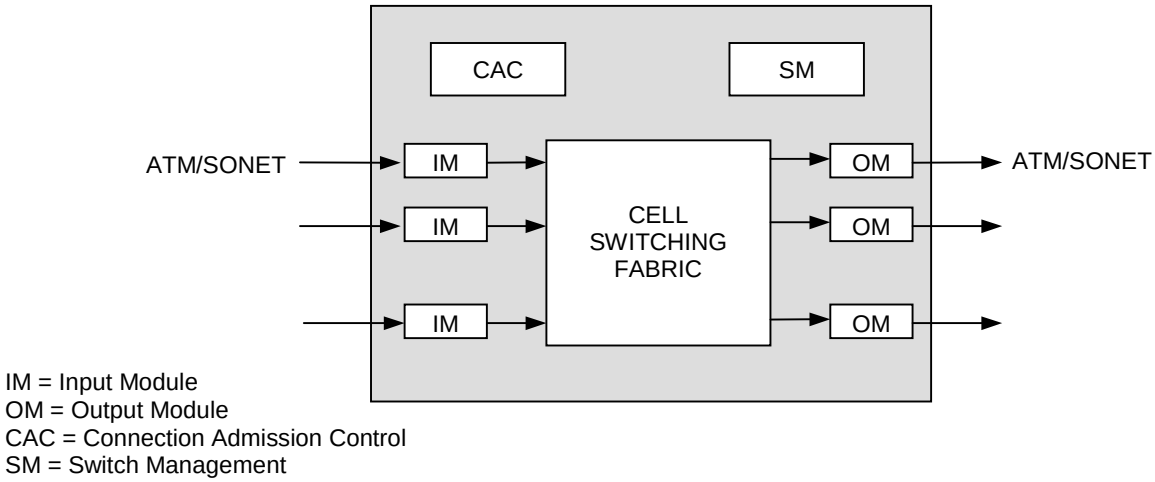

Figure 2.1 - Model générique d'un commutateur

Les blocs fonctionnels n'ont pas de délimitations matérielles très précises.

2.3.1. Modules d'entrée

Le module d'entrée effectue une analyse du signal et extrait le flux de cellules ATM. Les opérations de base sont la conversion et la récupération du signal, le traitement de l'en-tête SONET et la dé-sérialisation de la cellule. Pour chaque cellule ATM les fonctions suivantes sont exécutées:

- le contrôle de l'erreur de l'en-tête en utilisant le champ HEC (*Header Error Control*) ;
- la détermination du port destination en fonction des étiquettes VPI/VCI ;
- le transfert des cellules de signalisation à CAC et des cellules OAM au *Switch Manager* ;
- UPC/ UNC pour chaque VPC/ VCC ;

- l’addition d'une étiquette interne contenant l'information d’acheminement à travers l’équipement.

2.3.2 Modules de sortie

Les modules de sortie préparent la cellule ATM pour la transmission physique. Les opérations suivantes sont exécutées :
- l’enlèvement et le traitement de l'étiquette interne ;
- la translation possible des valeurs VPI/VCI ;
- la génération du champ HEC ;
- l’insertion de cellules de CAC et du *Switch Manager* dans le flux de cellules ATM ;
- la génération de l’en-tête SONET et la transformation des cellules dans un flux SONET ;
- la conversion du flux numérique en signal optique.

2.3.3 Le contrôleur de l’admission de connexions (CAC)

Le CAC établit, modifie et détermine les paramètres de chaque connexion: le chemin virtuel VP (*Virtual Path*) et le canal virtuel VC (*Virtual Channel*). Le CAC est responsable :
- des protocoles de signalisation de haut niveau ;
- de l’appel des fonctions AAL (*ATM Adaptation Layer*) pour interpréter et générer des cellules de signalisation ;
- de l’interface avec le réseau de signalisation ;
- de la négociation de contrats de trafic avec les utilisateurs qui demandent des nouveaux VPC et VCC ;
- de la renégociation avec les utilisateurs du changement de VPC/VCC déjà établis ;
- de l’allocation de ressources du commutateur pour les de VPC/VCC, et de la sélection du trajet ;
- des décisions d’admission ou de rejet des VPC/VCC ;
- de la génération des paramètres UPC/NPC ;

Le CAC peut être centralisé. Dans ce cas, l'unité de traitement reçoit les cellules de signalisation des modules d'entrée, les interprète, et produit des décisions d'admission et d'allocation des ressources pour toutes les connexions du commutateur. La fonction CAC peut être distribuée à travers les modules d'entrée. Cette solution demande un effort de développement plus important à cause de la nécessité de gérer la communication entre les CACs. Le contrôle distribué permet en revanche de gérer un large nombre de connexions, le contrôle étant distribué entre plusieurs unités parallèles. Cette distribution rend l'équipement plus modulable. Les commutateurs ATM de Hitachi et NEC ont des modules d'entrée possédant chacun leurs propre CAC. Certaines fonctions CAC sont distribuées dans les modules de sortie ce qui permet l'encapsulage des informations de contrôle des couches supérieures dans les cellules de signalisation.

2.3.4 Le *Switch Manager*

Le management d'un commutateur couvre un large spectre d'activités. Le *switch manager* gère l'OAM au niveau ATM et au niveau physique, la configuration des éléments du commutateur, la sécurité, le management du trafic, l'interface avec les systèmes d'exploitation et l'utilisation des ressources. La complexité des fonctions de management varie entre minimale et complexe. Un niveau minimal comprend quelques taches de base comme le management de la tolérance aux erreurs, le management de la performance, le management de la configuration, la comptabilité du trafic et le management de la sécurité et du trafic.

Un *switch manager* centralisé doit être capable de faire face à une rafale de demandes. Les fonctions de management peuvent être distribuées à travers les modules d'entrée et de sortie. Dans ce cas il est nécessaire de mettre en place un réseau de communication entre les différentes unités.

2.3.5 La matrice de commutation (*switch fabric*)

Le *switch fabric* est essentiellement responsable de l'acheminement des cellules de données, des cellules de signalisation et des cellules OAM entre les modules d'entrée et de sortie. Le *switch fabric* doit remplir aussi d'autres fonctions comme celle de tampon de mémorisation de cellules, monitorisation de la contention, concentration du trafic, multiplexage et ordonnancement de cellules selon les priorités. Un service de redondance des données permet de gérer la tolérance aux erreurs. Le *switch fabric* doit être capable de faire de la diffusion (*multicast* et *broadcast*).

2.3.2.1 La concentration, l'expansion et le multiplexage

Le trafic a besoin d'une concentration dans le port d'entrée pour une meilleure utilisation de la bande passante. La concentration cumule les débits de différentes connexions et réalise un flux caractérisé par un débit comparable avec le débit du port d'entrée du *switch fabric*. La proportion de la concentration est corrélée avec les caractéristiques des connexions gérées et nécessite parfois une reconfiguration dynamique. La concentration permet de réaliser un routage et une fonction de tampon de mémorisation sur des plans multiples et la duplication du trafic pour une meilleure tolérance aux erreurs. Les modules de sortie réalisent l'opération inverse, celle d'expansion du trafic.

2.3.2.2 Le routage et la fonction de tampon de mémorisation

Le routage et la fonction de tampon de mémorisation sont les fonctionnalités les plus importantes du *switch fabric*. Le module d'entrée attache une en-tête à chaque trame, et le *switch fabric* route la trame vers le module de sortie selon une table de routage prédéfinie. Il est possible que plusieurs trames demandent le même module de sortie. Pour faire face à la contention les cellules sont organisées dans des files d'attente. Il existe une multitude de stratégies de routage et de fonctions de tampon de mémorisation. Les plus complexes impliquent un degré important de parallélisme et de contrôle distribué au niveau matériel. Afin d'examiner ces différentes stratégies, quelques critères de performance peuvent être identifiés:

- la capacité totale de commutation,
- les délais de transit et la gigue,
- le degré d'utilisation exprimé par le débit total réalisé rapporté à la capacité maximale du commutateur,

- la tolérance aux erreurs,
- la capacité globale des tampons de mémorisation,
- la complexité de l'implémentation

2.4 Exemples d'architectures de commutation

Une des classifications possibles des architectures de commutation ATM a été proposée par Chen et Liu [1]. Les quatre catégories ont été différenciées en fonction de la conception matérielle de la matrice de commutation :
- architectures à mémoire partagée,
- architectures avec un bus partagé,
- architectures à interconnexion complète,
- architectures utilisant un réseau multi-étages.

L'étude considère un commutateur avec N ports d'entrée et N ports de sortie fonctionnant à la même vitesse.

2.4.1 L'approche mémoire partagée

La structure d'un commutateur à mémoire partagée est présentée dans la figure 2.2. Les cellules entrant en flux série sont converties en un flux parallèle et écrites d'une manière séquentielle dans une mémoire double port. Un contrôleur d'accès décide l'adresse d'écriture des cellules en mémoire selon l'en-tête de chaque trame. Cette approche est utilisable seulement pour les protocoles de communication avec de trames de taille fixe comme l'ATM, chaque trame occupant une zone de mémoire bien définie.

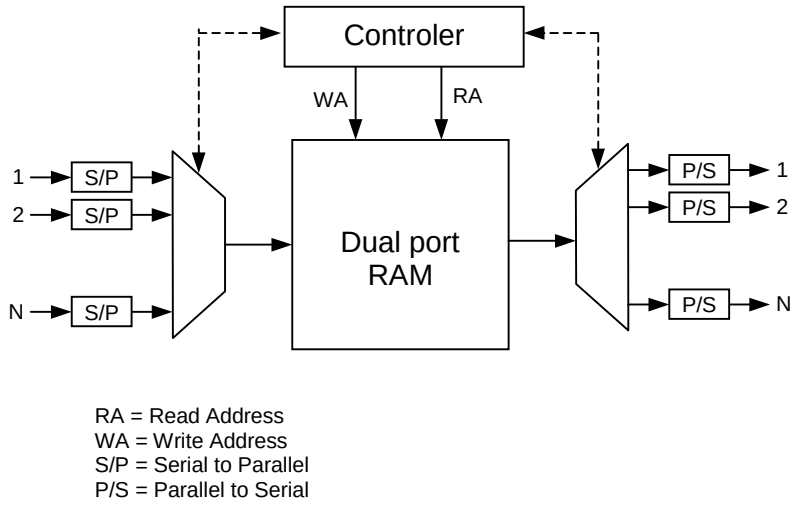

Figure 2.2 - Architecture de type mémoire partagée

L'approche est équivalente à une fonction de mémorisation en sortie. La capacité de mémorisation est commune à toutes les connexions. Avec une telle architecture, 100% de la capacité mémoire peut être utilisée quand le système est lourdement chargé. Si les mémoires tampon sont distribuées, l'utilisation est toujours inférieure à 100%. Le commutateur Prelude réalisé par CNET est un de premiers prototypes de cette technique [2]. D'autres exemples sont GCNS-2000 de AT&T et Hitachi [3].

Le principal inconvénient de cette architecture est la limitation du débit total. La bande passante totale est réduite au débit supporté par la mémoire partagée. Le débit supporté par la mémoire doit être N fois supérieur au débit des ports d'entrée. La mémoire possède un seul port d'écriture et les cellules sont écrites une après l'autre. Le contrôleur est centralisé et il doit traiter les en-têtes de cellules pour que la bande passante du port d'écriture de la mémoire soit entièrement utilisée. En conséquence la QoS est difficilement gérable, la complexité des algorithmes d'ordonnancement étant en corrélation avec le nombre de classes de priorités. La diffusion vers plusieurs ports de sortie est aussi mise en difficulté. Un autre inconvénient de la centralisation du contrôle est la faible modularité.

2.4.2 L'approche bus partagé

Les cellules sont transférées entre les ports de sortie et d'entrée à l'aide d'un bus. L'exemple le plus connu est celui d'un multiplexage temporel ou TDM (*Time Division Multiplexing*). Une telle structure est présentée dans la figure 2.3. Après la transformation du flux série de cellules en un flux parallèle, une en-tête contenant l'indice du port de sortie est attachée à chaque cellule. Les cellules sont diffusées vers tous les récepteurs possibles sur le bus TDM. Les ports d'entrée se partage entre eux le contrôle du bus d'une manière tournante. Les ports de sortie réalisent un filtrage de l'adresse contenue dans l'en-tête système et les cellules qui lui sont destinées sont transférées vers les ports de sortie.

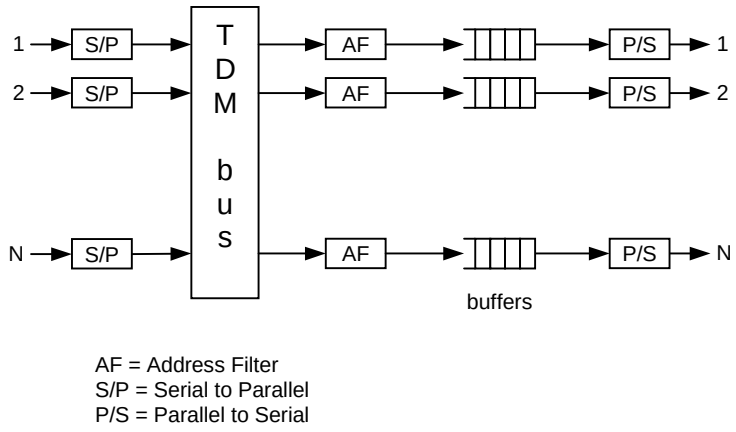

Figure 2.3 - Architecture de type bus partagé

Les ports de sortie sont modulaires ce qui facilite l'implémentation des filtres d'adresse et des mémoires tampon. Cette architecture permet facilement la diffusion de la même cellule vers plusieurs ports de sortie. Plusieurs commutateurs utilisent cette architecture: le commutateur d'IBM PARIS (*Packetized Automated Routing Integrated System*), plaNET, le système de NEC ATOM (*ATM Output Buffer Modular Switch*), et ForeRunner ASX-100 de Fore Systems [4]. SCPS (*Synchronous Composite Packet Switching*) et SafeCom4000 sont d'autres exemple d'architecture à bus partagé [6].

Le désavantage majeur des architectures à bus partagé est la limitation de la bande passante. Comme dans le cas précèdent, le débit du bus doit être au moins N fois supérieur au débit des ports. Les filtres d'adresses opèrent à la vitesse du bus partagé. L'extensibilité est

physiquement limitée. Par rapport à l'architecture précédente, les mémoires tampons de sortie ne sont pas partagées ce qui demande une quantité plus importante de mémoire tampon pour une bande passante équivalente. Le fait que la bande passante peut être facilement distribuée aux ports permet d'offrir un bon niveau de qualité de service.

2.4.3 Architecture de type complètement connectée ou *crossbar*

L'idée de cette approche est de créer un chemin pour chacune des N^2 paires d'entrée-sortie par l'utilisation de N^2 points d'interconnexion. Une fois la cellule arrivée dans un port d'entrée, une en-tête avec l'identificateur du port de sortie est attachée et la cellule diffusée vers tous les ports de sortie. Les filtres d'adresses sélectionnent les cellules qui lui sont destinées. L'architecture est représentée dans la figure 2.4.

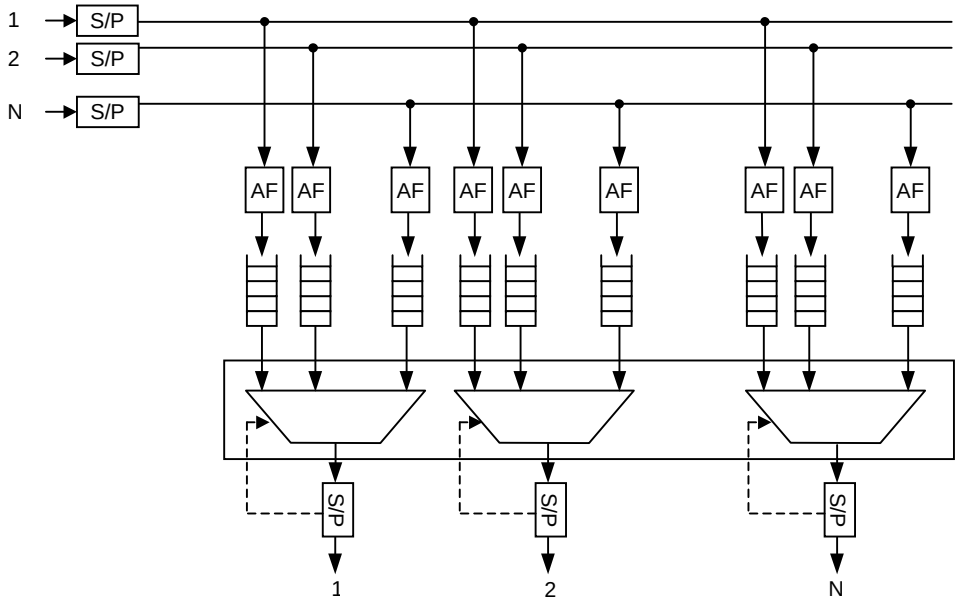

Figure 2.4 - Architecture de type « complètement connectée »

Le principal inconvénient de cette architecture est son coût puisqu'il y a N^2 files d'attente et N^2 filtres d'adresse. Une autre limitation est la faible extensibilité à cause de la complexité matérielle. A part ces inconvénients, cette architecture a beaucoup d'avantages. La diffusion est naturelle, c'est le concept même de l'architecture. La fonction de tampon de mémorisation

a lieu en sortie et chaque port d'entrée a sa propre mémoire tampon. Les mémoires tampons comme les filtres d'adresses fonctionnent à la vitesse du lien.

Quelques exemples de commutateurs qui implémentent cette architecture : *Bus Matrix Switch* de Fujitsu et GTE Government System de SPANet. AT&T a développé un prototype de commutateur nommé Knockout [6].

2.4.4 Approche de type réseau multi-étages

Un autre exemple de matrice de commutation sont les réseaux MIN (*Multistage Interconnection Networks*) qui ont été développés pour réduire le nombre de liens d'interconnexion. Les MIN sont des structures plus proches des arbres.

2.4.4.1 Les réseaux *banyan*

Un des types de réseaux MIN le plus connu est le réseau *banyan*. Un réseau *banyan* est constitué d'une interconnexion d'éléments de commutation qui sont des petits *crossbar* n×n. Un *crossbar* à deux entrées et deux sorties nécessite un seul bit de contrôle pour acheminer la cellule vers le port supérieur ou inférieur. De cette façon à chaque étage du réseau correspond un bit de l'adresse du port de sortie. Si dans un réseau MIN, l'adresse contenue dans l'en-tête de la trame spécifie un chemin unique vers le port de sortie il s'agit d'un auto-routage. La figure 2.5 représente un élément de base, un réseau *banyan* avec 4 ports et un réseau *banyan* avec 8 ports.

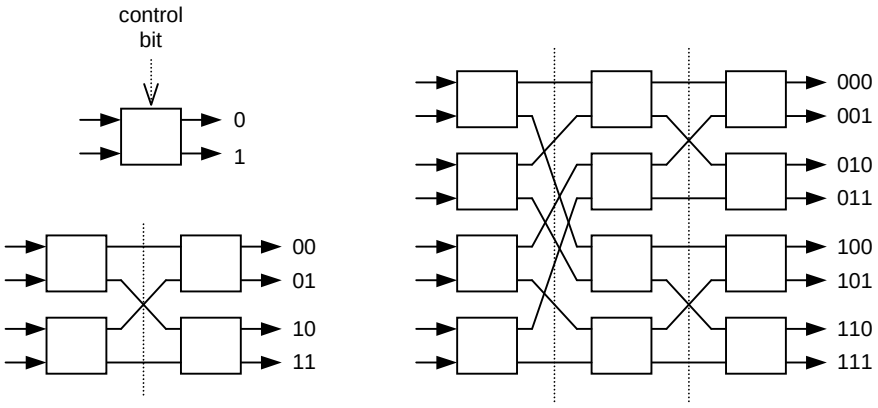

Figure 2.5 - Elément de base, réseau *banyan* avec 4 ports, réseau *banyan* avec 8 ports

Le nombre d'étages du réseau est le logarithme à base 2 du nombre de ports. La construction d'un réseau *banyan* N×N est simple. L'architecture est facilement extensible. Le routage des cellules est effectué par des éléments de base et tout le système opère à la même fréquence. L'adresse du port de sortie spécifie complètement le chemin à travers le réseau *banyan* et plusieurs cellules peuvent être acheminées d'une manière parallèle ce qui permet d'avoir une bande passante élevée. Le commutateur ATM Data Networks 1000 d'Alcatel utilise un réseau *banyan.*

2.4.4.2 Le risque de contention dans le réseaux *banyan*

Le désavantage majeur des réseaux *banyan* est le risque élevé de contention. Un réseau *banyan* avec N entrées et N sorties possède moins de N^2 points d'interconnexion. Si les en-têtes de deux cellules se trouvent sur les deux entrées d'un élément de base et sollicitent la même sortie, une des cellules passe vers l'étage suivant et l'autre est bloquée. La bande passante globale est donc supérieure à celle d'un bus unique mais inférieure à celle d'un *crossbar*. La seule solution est d'utiliser de la mémoire tampon dimensionnée pour faire face à ce risque de contention.

2.4.4.3 Réseaux MIN avec des chemins multiples

A part les architectures de type *banyan* qui sont de MIN à chemin unique, il existe d'autres types de réseaux MIN avec des chemins multiples comme les réseaux *Benes* et *Clos* ou les réseaux *banyan* en plans parallèles. Le fait d'avoir plusieurs alternatives de routage pour la même paire d'entrée-sortie permet une distribution du trafic sur des chemins parallèles ce qui minimise le nombre de conflits internes. Dans le cas de chemins multiples, les cellules de la même connexion peuvent arriver à la destination avec des délais différents. Ceci implique l'utilisation d'un mécanisme de séquencement du trafic. Widjaja et Leon-Garcia [7] ont proposé une architecture qui utilise de MIN à chemins multiples.

2.5 Conclusions. Performances et limites

La présentation de diverses approches architecturales permet de tirer quelques conclusions pour mieux comprendre les performances et les limites d'un système de commutation.

2.5.1 La contention interne, matrices de commutation bloquantes et non-bloquantes

Une matrice de commutation est non-bloquante s'il ne peut y avoir de contention sur les ports d'entrée dans l'hypothèse où le débit des ports de sortie est non borné. Les architectures de type division spatiale sont toujours bloquantes. Un système qui utilise un bus TDM qui fonctionne N fois plus vite que les N ports est une architecture non-bloquante. Les architectures à mémoire partagée sont aussi non-bloquantes si la fréquence de la mémoire est N fois supérieure à la fréquence des N ports. Un *crossbar* complet est non-bloquant. Les réseaux MIN sont des architecture bloquantes. Les architectures non-bloquantes permettent le fonctionnement au débit maximal tandis que les architectures bloquantes n'arrivent jamais à fonctionner au débit maximal. Pour réaliser un débit proche du débit maximal entrant, une architecture de ce type doit avoir la fréquence de la matrice de commutation supérieure à la fréquence des ports. Onvural et Raif [4] ont démontré qu'un réseau *banyan* à N ports fonctionnant à une fréquence de racine carré de N fois supérieure à la fréquence des ports peut prévenir tout blocage interne.

2.5.2 Utilisation de mémoires tampon

L'utilisation de mémoires tampon est indispensable dans toute architecture de commutation. Il existe quatre méthodes d'utilisation de ces mémoires tampon.

2.5.2.1 Utilisation des mémoires tampon en entrée de la matrice de commutation

L'utilisation des mémoires tampons en entrée a un inconvénient majeur. Si la cellule en tête de la FIFO est bloquée à cause de l'occupation du port de sortie par une autre cellule, toutes les autres cellules sont bloquées même si leur port de sortie est disponible (l'effet *head of the line*). Le problème peut être évité en classant les files d'attente selon leur port de destination et en utilisant un mécanisme d'ordonnancement informé sur les sortie bloquées.

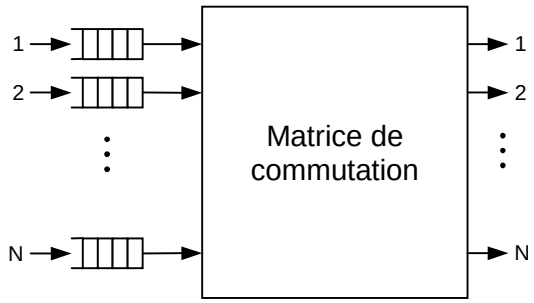

Figure 2.6 - Utilisation de mémoires tampon en entrée

2.5.2.2 Utilisation des mémoires tampon en sortie

La méthode est utilisée à la sortie des matrices de commutation non-bloquantes. Les tampons de mémorisation de sortie réalisent une adaptation du flux de cellules irrégulier venant du bus au débit des liens de sortie. Le tampons de mémorisation de sortie doivent opérer à la fréquence du bus ou du *crossbar*.

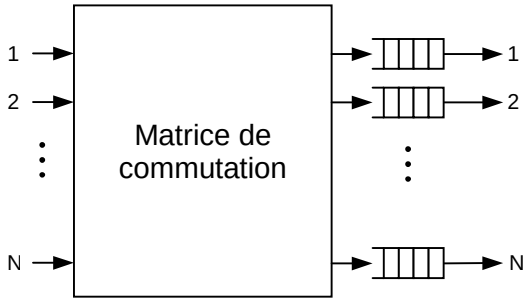

Figure 2.7 - Utilisation de mémoires tampon en sortie

2.5.2.3 Utilisation des mémoires tampon à l'intérieur d'une matrice de commutation

Dans un réseau multi-étages, la mémoire tampon est distribuée à travers les éléments de base de la matrice. Dans un réseau *banyan*, si deux cellules arrivant sur deux entrées distinctes sollicitent la même sortie d'un élément de base, une de cellules est stockée dans la FIFO locale et l'autre est transférée vers l'étage suivant. Le système de tampons de mémorisation distribués peut être utilisé pour l'implémentation d'un mécanisme de rétropropagation de la contention où les FIFO d'un étage du réseau banyan sont informées par un signal de retour qu'elles doivent retenir les cellules. La rétropropagation peut attendre les ports d'entrée et les cellules sont arrêtées dans les FIFO d'entrée. Le phénomène *head of the line* peut aussi

apparaître dans les FIFO distribuées. Un autre désavantage est l'introduction d'un délais variable pour la traversée de la matrice.

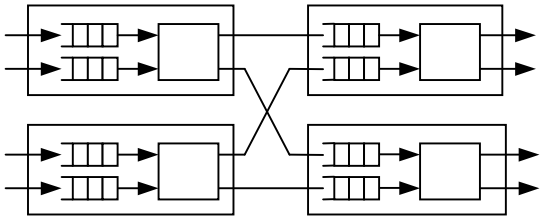

Figure 2.8 - Utilisation de mémoires tampon en interne

2.5.2.4 Utilisation de tampons de mémorisation extérieurs de recirculation

La technique est applicable aux *crossbars*. Si deux cellules sollicitent la même sortie, l'une est stockée dans un tampon de mémorisation externe dont la sortie est reinjectée dans le *crossbar*. Cette technique est utilisée par le commutateur Starlite de AT&T [8]. Cette technique demande un contrôle compliqué pour rétablir l'ordre séquentiel des cellules.

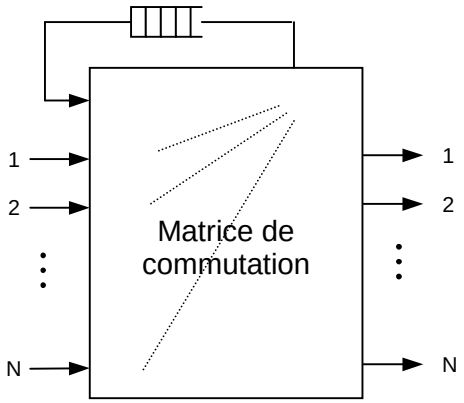

Figure 2.9 - Utilisation de tampons de mémorisation de recirculation

2.5.3 Problèmes d'extensibilité

Pour les architectures à mémoire partagée et les architectures à bus partagé, la limitation la plus importante est la ressource partagée. Les architectures de ce type réalisent un compromis entre le nombre de ports et la vitesse de ces ports. Pour augmenter la bande passante et le nombre de ports les architectures de type bus partagée et mémoire partagé nécessitent une

augmentation de la vitesse du bus ou de la fréquence de la mémoire. La croissance quadratique du matériel utilisé par un *crossbar* limite l'extensibilité. Les architectures de type réseau multi-étages sont en général plus facilement extensibles. Les deux facteurs de limitation pour les systèmes de commutation utilisant des réseaux multi-étages sont la densité d'intégration des circuits et la quantité de mémoire tampon utilisée.

Chapitre 3. Architecture de commutation ATM respectant le principe de la commutation de circuit

Le point de départ de cette étude est le commutateur SafeCom4000 développé par CS Telecom. Le but de notre étude a été de réutiliser les cartes et les circuits déjà développés. Nous proposons une nouvelle architecture nommée SafeCom4000X qui est un cluster de commutateurs connectés par des liens série haut débit. Au début de ce chapitre seront présentées les caractéristiques du SafeCom4000. La deuxième partie du chapitre, qui constitue notre contribution, présente l'architecture du SafeCom4000X. Cet équipement est modulable et peut atteindre 4,8Gb/s. Jusqu'à 1,8 Gb/s, un niveau de qualité de services comparable à celui de SafeCom4000 est maintenu.

3.1 Problématique issue des limitations du SafeCom4000

Ce paragraphe décrit la problématique posée par l'évolution architecturale du SafeCom4000.

3.1.1 Présentation générale de l'équipement SafeCom4000

L'équipement SafeCom4000 est un commutateur multiservice (ATM, FrameRelay, IP, CES) [5]. Il peut fonctionner comme commutateur dans un réseau utilisateur ou comme équipement d'accès opérateur. La QoS du SafeCom4000 est basée sur la réservation de bande passante. L'architecture possède d'importants atouts en ce qui concerne la qualité de service mais la bande passante globale est limitée par le principe architectural utilisé. Le défi est de concevoir une nouvelle architecture avec une valence supérieure en nombre de ports, (de façon à augmenter à la fois la bande passante et le nombre de connexions) et de garantir le même niveau de qualité de service.

La commutation et le traitement interne des données est réalisée à l'aide d'un bus « propriétaire » nommé bus ATM. Le protocole de ce bus utilise la méthode TDM (*Time*

Division Multiplexing). Les ports de sortie ne possèdent pas de capacité de mémorisation et ils se comportent comme des pipelines. L'ordonnancement des cellules est effectué par plusieurs ordonnanceurs placés dans une configuration parallèle sur le bus partagé. Les ordonnanceurs sont activés d'une manière séquentielle, chacun devenant maître du bus selon un ordre préétabli. Cette répartition de bande passante est réalisée par l'association d'un certain nombre de plages temporelles (*time slots*). Au niveau connexion, la qualité de service est gouvernée par des algorithmes d'ordonnancement capables de prendre en compte les contraintes définies pour chaque connexion. Toutes les fonctions ATM et le management du débit sont implémentés dans un composant ASIC nommé FACE (*Full ATM Core Engine*)[9].

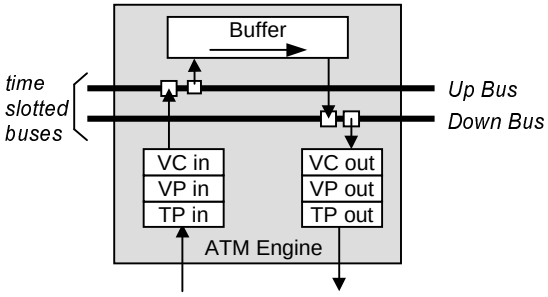

Figure 3.1 - Architecture générale du circuit FACE

L'architecture générale du circuit FACE comporte trois blocs. Les blocs d'entrée (*In* sur la figure 3.1) et de sortie (*Out* sur le figure 3.1) réalisent les fonctions ATM aux niveaux TP (*Trunk Path*), VP (*Virtual Path*) et VC (*Virtual Channel*). Le bloc tampon de mémorisation (*Buffer* sur la figure 3.1) contient les files d'attente. Toute cellule entrant par le bloc *In* passe par le bloc *Buffer* avant d'être transmise vers un bloc *Out*. La communication entre les blocs est réalisé à l'aide du bus ATM, accessible de l'extérieur et qui permet l'échange de cellules ATM avec d'autres circuits FACE. Plusieurs circuits FACE connectés sur le bus ATM permettent de réaliser un commutateur complexe. Le bus ATM est décomposé en deux parties, selon le types de transferts: il existe un bus *Up* qui set à réaliser les transferts d'un bloc d'entrée *In*(i) vers un bloc *Buffer*(k) et le bus *Down* qui permet les transferts d'un bloc *Buffer*(k) vers un bloc *Out*(j). Chacun de ces bus a une largeur de 32 bits et une fréquence de 22 MHz. Le débit utile maximal d'un bus (en termes de cellules ATM) est de 622 Mb/s. C'est également le débit maximal du commutateur, puisque les deux bus sont utilisés « en série ». L'architecture du cœur de l'équipement SafeCom4000 est composé de 4 circuits FACE connectés sur le bus ATM.

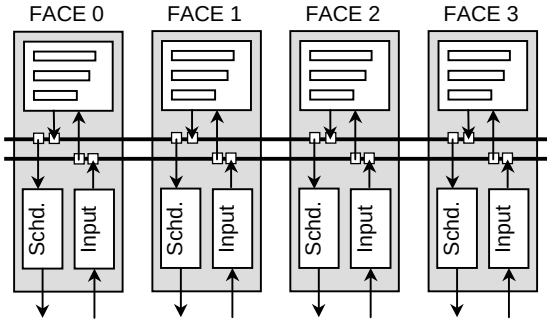

Figure 3.2 - Architecture du cœur du commutateur SafeCom4000

Les blocs *In*(i) sont responsables des transferts sur le bus *Up*. Pour transférer les cellules ATM d'un port *In*(i) vers un *Buffer*(k), le port *In*(i) émet une requête sur un bus de contrôle pendant la plage temporelle (*time slot*) correspondante. Le transfert est fait sans acquittement mais bloc *Buffer*(k) émet après chaque requête un message de rétro-propagation de la contention (*back-pressure*) contenant une information sur la longueur de la file d'attente correspondant au flux auquel appartient la cellule. Le protocole correspondant à l'injection d'une cellule est présenté dans la figure 3.3. Notons que l'équipement SafeCom4000 n'effectue que des transferts locaux (d'un bloc *In*(i) vers un bloc *Buffer*(i) situé sur la même puce), mais que rien n'empêche en principe un bloc *In*(i) d'effectuer une requête vers un bloc *Buffer*(k) situé sur une autre puce, durant la plage temporelle qui lui est attribuée.

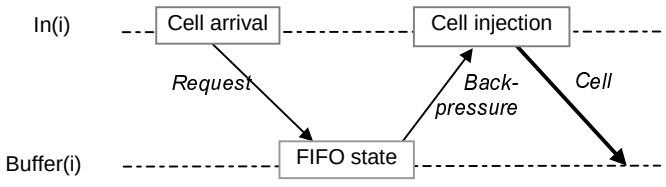

Figure 3.3 - Protocole correspondant à l'injection d'une cellule dans le tampon

Les blocs *Out*(j) sont responsables des transferts sur le bus output. Les ordonnanceurs sont situés dans les blocs *Out*(j). A chaque injection d'une cellule dans un bloc *Buffer*(k), une notification contenant l'identificateur de la connexion à laquelle la cellule appartient est envoyée vers tous les ordonnanceurs du commutateur. Chaque ordonnanceur j connaît donc la disponibilité des cellules pour chacun des flux qui doivent sortir par le bloc *Out*(j). A partir de ces informations, et des contrat de qualité de service qui ont été négociés pour chacune des connexions, l'ordonnanceur j choisit une connexion x et adresse la requête vers le tampon

Buffer(k) correspondant pendant la plage temporelle qui lui est allouée. Le tampon *Buffer*(k) émet la cellule placée dans la tête de la file d'attente x choisie sur le bus *Down*. Le protocole correspondant à l'extraction d'une cellule est présenté dans la figure 3.4.

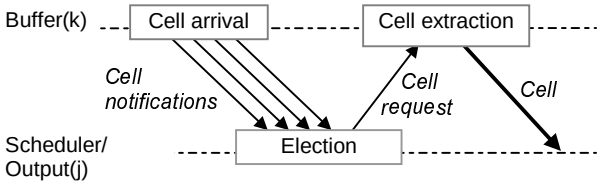

Figure 3.4 - Protocole correspondant à l'extraction d'une cellule du tampon de mémorisation

L'avantage principal d'une architecture de type TDM est de pouvoir garantir une bande passante à chaque port et de permettre au manager du commutateur de garantir la QoS pour chaque connexion.

3.1.2 Evolution du SafeCom4000 : augmentation de la bande passante avec le maintien de la qualité de services

SafeCom4000 reste un peu faible en termes de débit dans sa catégorie des équipements. Le problème posé initialement est d'augmenter la bande passante totale et de maintenir la même qualité de service. Etant donné qu'il s'agit d'un projet industriel, le coût d'opération doit être minimal. Le bus ATM se trouve en fond de panier de l'équipement. Le problème posé est de réutiliser sans modifications majeures la mécanique de l'équipement ainsi que les cartes existantes et éventuellement de rajouter de nouvelles cartes qui permettent de réaliser un cluster de commutateurs. Il reste à définir la méthode de connexion entre les sous-ensembles et à concevoir un protocole de communication entre les éléments du cluster.

3.2 Architecture du commutateur SafeCom4000

3.2.1 Types de cartes

L'architecture de l'équipement SafeCom4000 est hiérarchique et les cartes sont connectées à un bus d'interconnexion avec un débit en rapport avec le débit de la carte elle-même. Deux niveaux d'interconnexion ont été définis : UTOPIA bus à 155Mb/s et ATM bus à 622 Mb/s. Les bus de connexion sont disponibles en fond de panier ou sont internes à une carte. Certaines ressources comme le CPU ou le cœur ATM sont centralisées. Le circuit FACE intègre les fonctions ATM et permet de concentrer les accès des cartes faible débits à travers des bus UTOPIA disposées en fond de panier. L'architecture est modulable et plusieurs configurations peuvent être conçues à partir de quatre types de cartes : CPU, SUPATM, ATM et service.

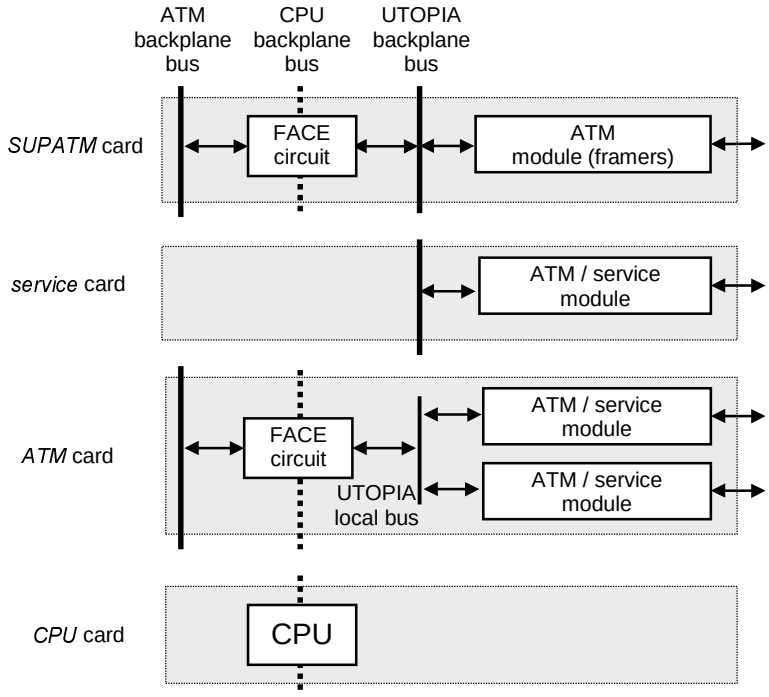

Figure 3.5 - Les quatre types de cartes

3.2.2 Le circuit FACE

Le circuit FACE (*Full ATM Core Engine*) est composé de deux blocs fonctionnels ISAF (*Integrated Switching ATM Functions*) et ICAF (*Integrated Control ATM Functions*). Le bloc ICAF exécute les fonctions ATM sur un flux ATM bidirectionnel provenant d'un bus Utopia et pouvant aller jusqu'au 155Mb/s. La partie réception du ICAF exécute les fonction liées à l'interface avec le bus Utopia, le démultiplexage VP/VC, F4 et UPC. La partie émission exécute les fonctions liées à l'interface Utopia, la mise en forme VP du trafic (VP *shaping*), le multiplexage VC et F4. Le module est pourvu d'une mémoire SSRAM pour stocker l'information de contexte sur les connexions gérées. Le bloc ISAF exécute le management des files d'attente et les fonctions de mémoire tampon et de translation d'en-tête. Il peut exister jusqu'à 1024 files d'attente. ISAF possède une mémoire SSRAM pour stoker les paramètres des connexions et une mémoire SDRAM pour les cellules. Le module IOBUS permet l'interconnexion en interne les blocs ISAF et ICAF et l'interconnexion externe via le bus ATM avec des modules appartenant aux autres circuits FACE. L'interface processeur a 32 bits et permet des accès singuliers ou en pipeline. Plusieurs processeur sont compatibles avec le protocole du bus processeur du circuit FACE: 6840/6860, MPC860 PowerQuick et 8960HD.

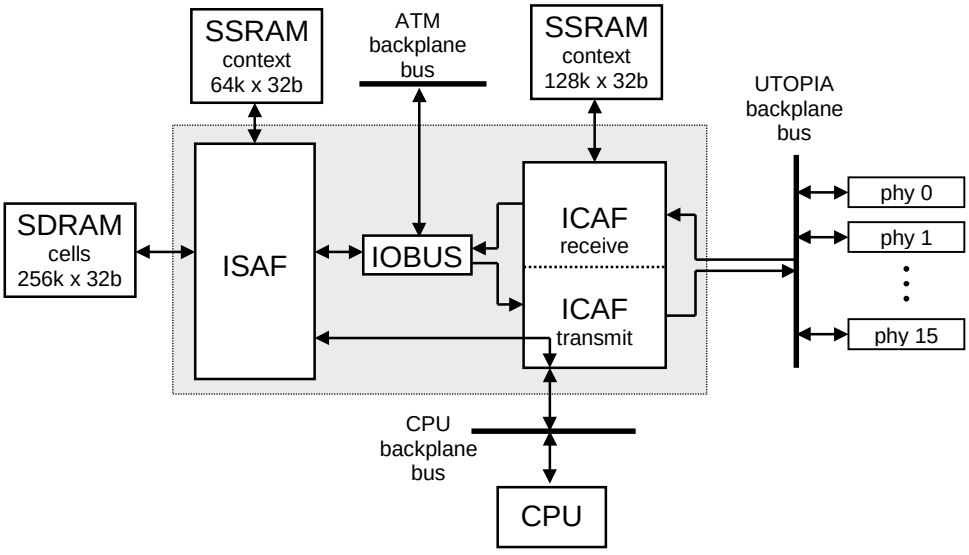

Figure 3.6 - Circuit FACE : blocs fonctionnels et interfaces

Le circuit FACE peut à lui seul fonctionner comme un mini-commutateur. Le circuit FACE utilise le principe de la mémoire partagée et réalise la commutation physique entre les ports du bus Utopia.

3.2.3 Le bus ATM

Le bus ATM, nommé aussi le « bus cellule » assure des transferts bi-directionnels de cellules ATM entre les cartes de type ATM et SUPATM. Les cartes ATM réalisent une adaptation d'un service quelconque (Frame Relay, SMDS, 2Mb/s tramé etc...) d'abord au bus Utopia et puis au bus ATM. Les cartes SUPATM adaptent les interfaces ATM (155Mb/s SDH, 34Mb/s PDH etc...) à ce même bus. Utilisant deux chemins de données ayant chacun 32 bits, le bus réalise en même temps deux types de transfert: un transfert nommé « montant » d'un ICAF vers un ISAF distant, sur le chemin *Up* ou UDATA et un transfert nommé « descendant » d'un ISAF vers un ICAF distant sur le chemin *Down* ou DDATA. Pour les transferts montants, la source et la destination sont uniques. Dans le sens descendant, le bus ATM permet la diffusion. Les informations de contrôle sont bien sûr différentes selon le sens. Elles se trouvent multiplexées sur deux bus ADDR (*ADDRess*) et BR (*Bus Request*). ADDR est un bus d'adressage sur lequel sont émises les requêtes de transferts. Chaque circuit FACE dispose d'une ligne BR écoutée par tous les autres circuits.

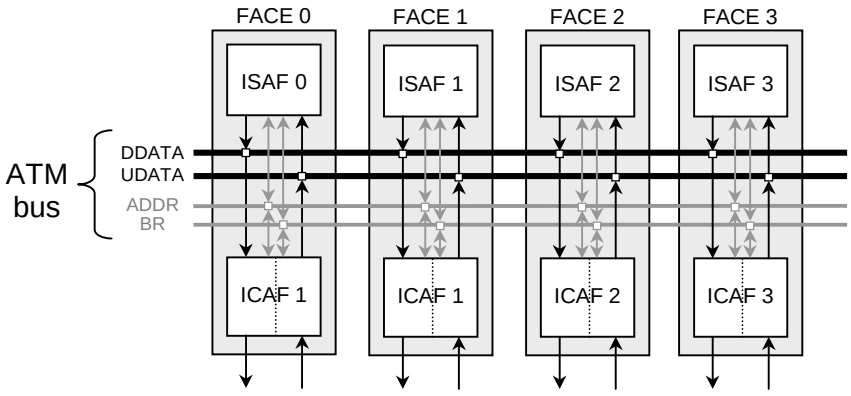

Figure 3.7 - Connexion au bus ATM

Un répartiteur de plages temporelles est décentralisé dans chaque carte réseau et configuré de façon qu'à un instant donné, un seul ICAF initialise un transfert montant et un transfert

descendant. Cette allocation est fixe et indépendante de l'état des cartes SUPATM, elle dépend de la bande passante nécessaire à chaque carte réseau. En conséquence, un cycle de transfert non utilisé par un ICAF est perdu sans possibilité de récupération par une autre carte.

3.2.4 Le principe de commutation du SafeCom4000

SafeCom4000 utilise le principe de commutation du bus partagé. Plusieurs circuits FACE disposés sur le bus ATM réalisent l'échange de cellules selon le principe du bus partagé. Le circuit FACE peut à lui seul fonctionner comme un mini-commutateur. Il utilise le principe de la mémoire partagée et réalise la commutation physique entre les ports du bus Utopia.

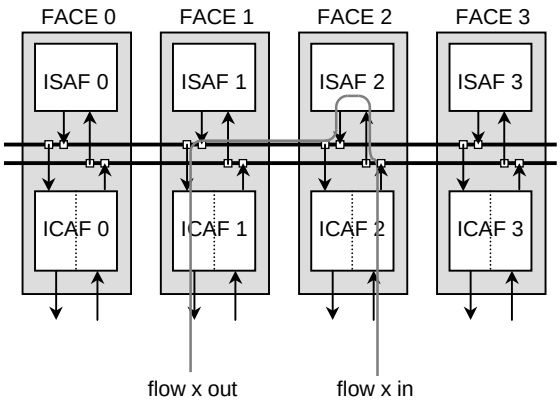

Figure 3.8 - Principe de commutation de flux utilisateurs du SafeCom4000

La commutation physique est opéré sur le bus ATM *Up* à la sortie du block ISAF. La figure 3.8 présente le parcours d'une cellule appartenant à un flux x. La cellule est reçue par l'ICAF 2, envoyée à l'ISAF 2 et transmise à travers le bus ATM à l'ICAF 1.

3.2.5 L'allocation du débit sur le bus ATM et le mécanisme de QoS

Le signaux du bus ATM sont synchrones avec une référence d'horloge et avec deux répartiteurs de cycle. Ces signaux sont émis par une carte de synchronisation connectée sur le bus ATM. Un des répartiteurs de cycles nommé NTS (*New Time Slot*) délimite des plages

temporelles de 16 cycles. A chaque plage temporelle correspondent les transferts de deux cellules une dans le sens montant et une dans le sens descendant. L'autre répartiteur de cycles, le signal LTS (*Last Time Slot*) délimite des groupes de 64 plages temporelles. Le circuits FACE possède, également un compteur de cycles et un compteur de plages temporelles. La valeur du compteur de plages temporelles est incrémentée d'une unité à chaque occurrence du signal NTS. A un instant donné, grâce à la synchronisation avec le signal LTS la valeur de ce compteur est la même dans tout les circuits FACE.

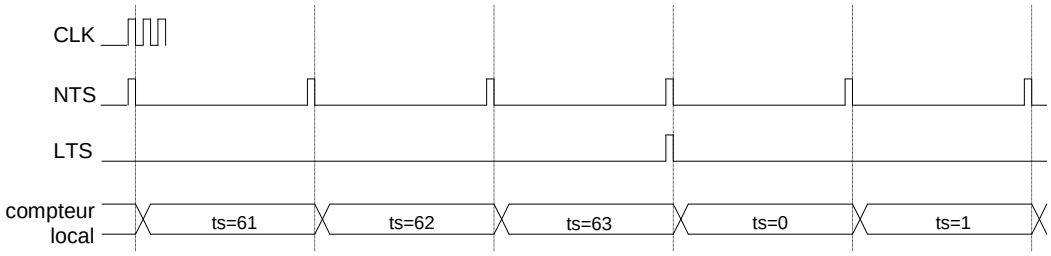

Figure 3.9 - Synchronisation du bus ATM

Chaque carte SUPATM se voit allouer une liste de plages temporelles c'est à dire un ensemble de valeurs entre 0 et 63, les listes de différentes cartes étant disjointes. A chaque nouvelle plage temporelle, la carte SUPATM qui possède dans sa liste la valeur du compteur de plages temporelles est autorisée à utiliser le bus ATM. Il est possible que des plages temporelles ne soient pas allouées et qu'elles ne se trouvent dans aucune des listes, ce qui est équivalent à une partie de la bande passante non utilisée. Il existe une correspondance entre le nombre de plages temporelles et le pourcentage du débit alloué. A chaque plage temporelle corresponds 1/64 du débit maximum du bus soit 9,37 Mb/s par sens. Pour des raisons de cohérence du pipeline, deux numéros consécutifs sur la liste des plages temporelles allouées à une carte réseau doivent être espacées d'au moins 4 unités. Par exemple, pour allouer à une carte ATM/SDH un débit de 150Mb/s, la liste suivante lui est fournie : 0+n, 4+n, 8+n, 12+n, 16+n, 20+n, 24+n, 28+n, 32+n, 36+n, 40+n, 44+n, 48+n, 52+n, 56+n, 60+n.

3.2.6 Implémentation de la QoS, mécanismes de transfert

3.2.6.1 Les transferts dans le sens montant

Sur un bus classique, la carte maître adresse directement la carte esclave et l'invite à opérer un transfert. Le protocole du bus ATM est un peu plus complexe dans le sens montant, dans la mesure où la carte esclave est désignée par la carte maître après un arbitrage :

- ts=n : la partie réception d'un ICAF, qui s'est vu allouer une plage temporaire par son répartiteur de cycle, émet une requête de transfert montant nommée UREQ (*Up REQest*) sur le bus ADDR. Cette requête contient un identificateur de groupement nommé GRPI (*GRoup IDentifier*). GRPI représente en effet un identificateur de connexion et il est associé d'une manière unique avec un VCI et un VPI d'un conduit ATM. L'ordre d'émission de requêtes est fonction des contrats de QoS et permet de garantir la bande passante de chaque conduit ATM.
- ts=n+1 : les tampons ISAFs répondent sur le bus BR avec un message COB (*Cell On Board*) si ils possèdent au moins une cellule appartenant à la connexion demandée.
- ts=n+2, t= n+3 : l'ICAF collecte ces résultats et élit alors une ISAF par priorité tournante en émettant alors l'identificateur de l'ISAF sur le bus de contrôle ADDR.
- ts=n+4, t= n+5 : l'ISAF désignée envoie sa cellule sur le bus données UDATA.

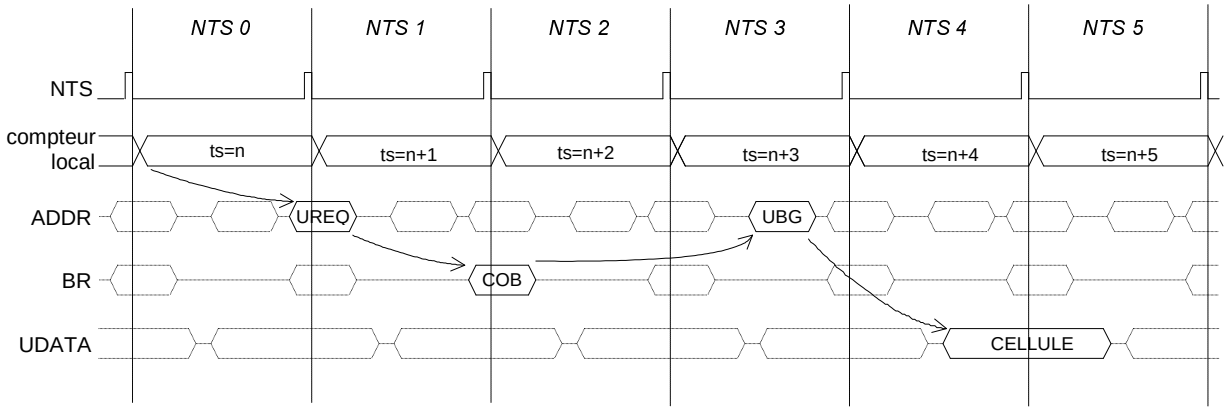

Figure 3.10 - Protocole de transfert dans le sens montant

Le séquencement d'un tel algorithme cellule par cellule est incompatible avec des débits élevés, il est donc nécessaire d'utiliser un pipeline. Chaque étage du pipeline est responsable d'une étape du protocole et donc d'une fonction et tous les étages travaillent simultanément sur des requêtes différentes. Le dernier étage conclut par le transfert effectif de la cellule

ATM sur le chemin de données UDATA. Le débit nominal du bus ATM est d'une cellule par plage temporelle dans chaque sens.

3.2.6.2 Les transferts dans le sens descendant

Dans le sens descendant, l'adressage d'un ISAF est classique. L'initiateur du transfert est toujours le module ICAF. La demande d'envoi de la cellule ne nécessite pas un acquittement.

- ts=n : par l'intermédiaire de sa table d'allocation, un ICAF se voit allouer une plage temporelle et se prépare à émettre une requête descendante ;
- ts=n+1 : la requête descendante nommée DREQ est émise sur le bus ADDR. La requête contient l'identificateur de l'ISAF source de la cellule. L'identification est faite par son numéro ou par un bit dans un champ de bits dans le cas de la diffusion.
- ts=n+2 : Il existe un mécanisme de rétro-propagation qui sert à informer les ICAF sur l'état des files d'attente. L'information de rétro-propagation sur trois bits est contenue dans un message nommé DBP (*Down Backpressure*) et représente le niveau de remplissage des files. Cette information sert au pipeline montant de l'ICAF ainsi qu'au pipeline descendant.
- ts=n+3 : La carte services reçoit la cellule sur le bus de données DDATA.

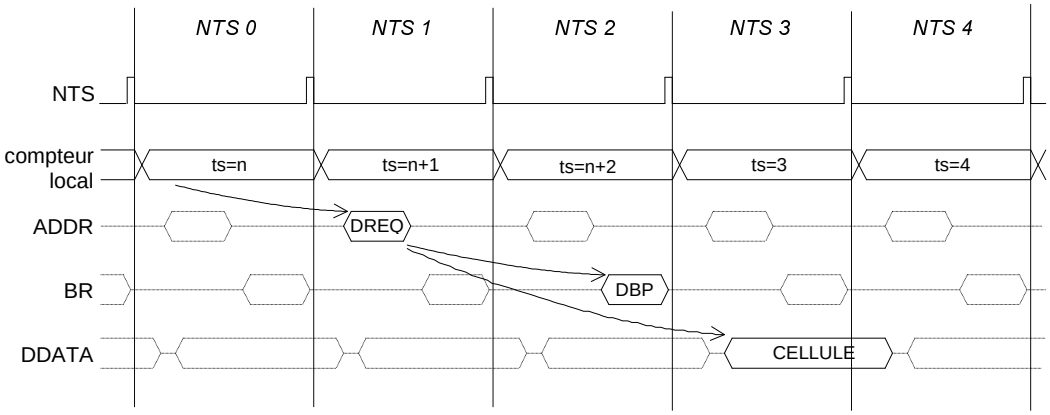

Figure 3.11 - Protocole de transfert dans le sens descendant

Il existe aussi un autre échange d'information sur le bus ATM. L'ISAF a la possibilité d'émettre des "notifications" vers les ICAF partie montante. Une notification signale l'arrivée d'une nouvelle cellule disponible dans une file d'attente d'émission de l'ISAF. La notification est caractérisée par l'identificateur de la file d'attente. Elle est asynchrone dans la

mesure où elle peut être émise dans n'importe quelle plage temporelle. Cette notification est diffusée vers toutes les ICAF sur le bus de contrôle BR. En fait, un seul ICAF est intéressé par la file d'attente et sa notification. L'ICAF peut, selon son gré et ses performances, moduler les émissions du GRPI lié à la file d'attente en fonction de ces notifications de cellules.

3.3 Architecture proposée pour le SafeCom4000X

Cette partie présente l'architecture que nous proposons. L'étude a été poursuivie tenant compte du matériel existant (composant FACE, architecture du bus ATM). L'innovation consiste en la mise en place d'un nouveau principe de commutation qui se rajoute au principe existant.

3.3.1 Le principe de commutation du SafeCom4000X

L'architecture SafeCom4000X est conçue comme un cluster de plusieurs équipements SafeCom4000. Une liaison haut débit bidirectionnelle HSL (*High Speed Link*) est utilisée entre n'importe quelle paire de sous-systèmes. Dés le début se pose le problème de l'allocation des flux qui passent d'un sous-système à l'autre. Un sous-système doit être capable de faire la distinction entre les cellules appartenant aux flux qui restent internes et qui seront traitées selon le principe de commutation local et les cellules des flux ayant comme destination un port d'un autre sous-système. Dans ce but un module spécial nommé RISAF (*Remote ISAF*) est prévu pour faire face au trafic intra-cluster. RISAF contient les files d'attente de toutes les connexions unidirectionnelles d'une paire (sous-système A vers sous-système B). RISAF gère uniquement les files d'attente des connexions de niveau « cluster » tandis que les ISAF locaux gèrent uniquement les connexions « locales ». Tenant compte du fait que RISAF doit faire face à un trafic beaucoup plus important sa capacité est double par rapport au ISAF. Il est capable de gérer jusqu'à 4096 connections groupées dans 2048 files d'attente. La mémoire de contexte est une SSRAM de 128K x 32 bits et la mémoire de cellules est une SDRAM de 512K x 32 bits. Le principe de commutation de flux utilisateur étendu au niveau cluster est présenté dans la figure 3.12.

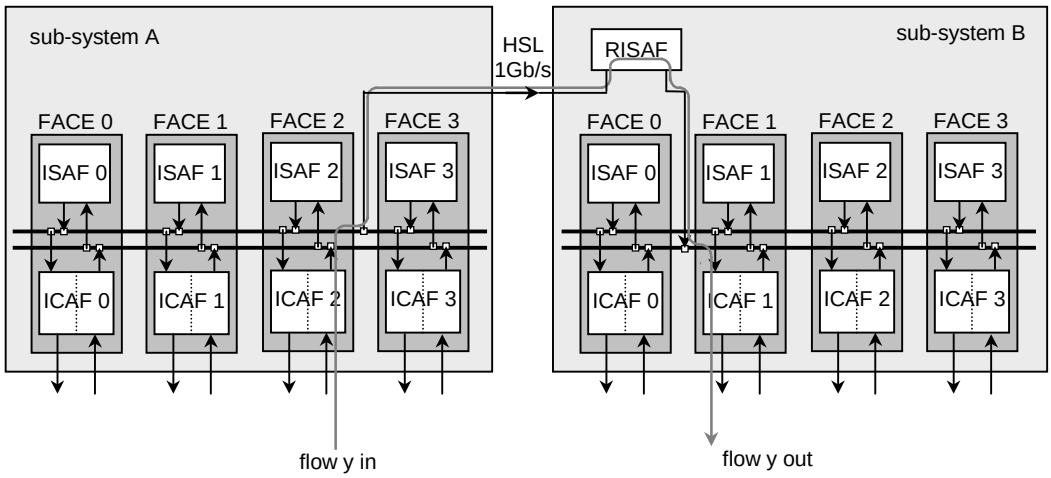

Figure 3.12 - Principe de commutation de flux utilisateur du SafeCom4000X

RISAF est disposé dans le sous-ensemble destination et sera connecté directement au bus ATM Up. Les cellules d'un ICAF au lieu d'être envoyées vers un ISAF local seront envoyées vers un RISAF connecté à un autre sous-ensemble. Un nouveau module nommé F2F (*FACE to FACE*) joue le rôle de connecteur entre les sous-ensembles. F2F contient principalement un module RISAF (Remonte ISAF), les drivers d'un médium physique HSL qui réalise l'interconnexion entre les sous-ensembles et toute la logique qui lui permet de se connecter sur le bus ATM. Tenant compte de l'en-tête système, le débit réel sur le médium physique qui relie deux sous-ensembles sera d'environ 800 Mb/s au lieu de 660 Mb/s qui est le débit total maximal du bus ATM. Pour avoir des connexions bidirectionnelles les circuits F2F doivent être montés tête-bêche.

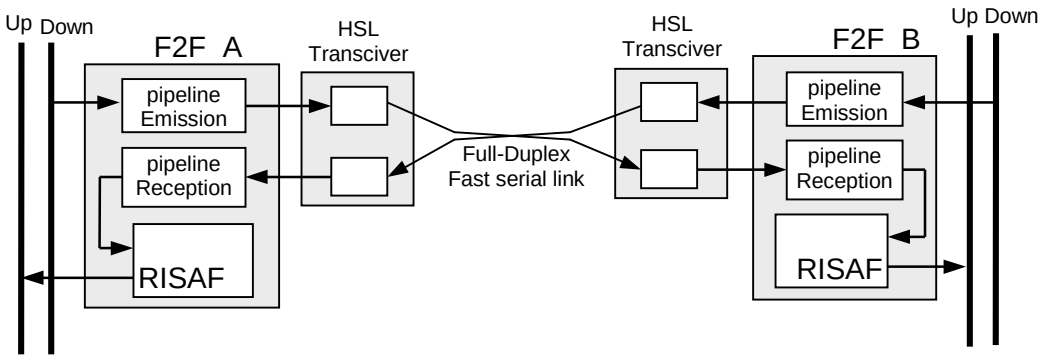

Figure 3.13 - Circuits F2F montés tête-bêche

L'échange des cellules entre deux sous-systèmes est fait en utilisant un transfert descendant vers un RISAF d'un sous-système distant. Dans le sens descendant, le pipeline émission

convertit les requêtes de cellules descendantes et les cellules correspondantes dans des trames HSL destinés à être envoyés via le lien HSL au RISAF du sous-système distant.

L'architecture du SafeCom4000X continue la logique des niveaux hiérarchiques et un troisième niveau d'interconnexion est rajouté. Ce troisième niveau nommé HSL *(High Speed Link*) succède le bus ATM et le bus UTOPIA. Le plan utilisateur (*user plane*) ATM de SafeCom4000 devient un sous-domaine du SafeCom4000X.

3.3.2 Topologie du SafeCom4000X utilisant des liaisons point à point

SafeCom4000X est un maillage de plusieurs sous-ensembles. La limitation du nombre de sous-ensembles est donnée par la valeur maximale du débit instantané entrant dans un sous-système.

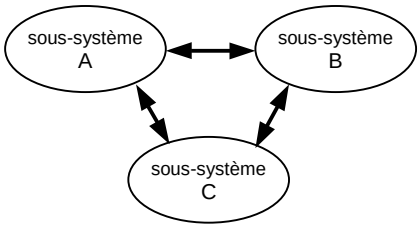

Figure 3.14 - Topologie à trois sous-systèmes 1,8Gb/s

Une configuration avec trois sous-systèmes est réalisable à l'aide des liaisons point à point. Cette configuration utilise deux circuits F2F et quatre circuits FACE par sous-système.

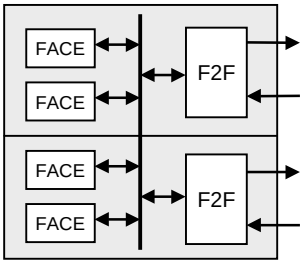

Figure 3.15 - Architecture du sous-système

Les modifications matérielles nécessaires pour l'implémentation de la nouvelle architecture sont limitées aux modifications du fond de panier et à la conception d'une nouvelle carte,

nommée SUPATMX, carte qui dispose de deux circuits FACE et d'un circuit F2F. La structure de carte SUPATMX est présentée dans la figure 3.16.

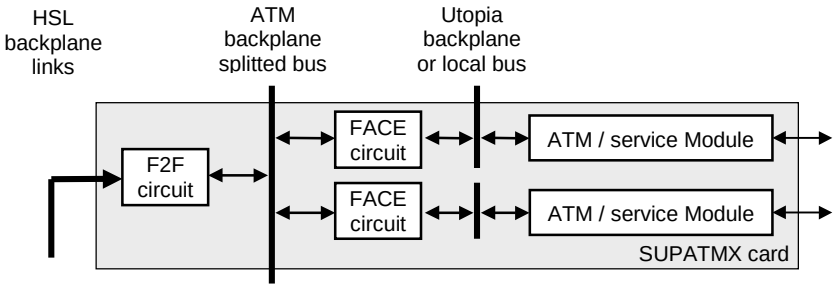

Figure 3.16 - Carte SUATMX

La topologie du SafeCom4000X avec 6 slots (1,8 Gb/s) est présentée par la figure 3.17.

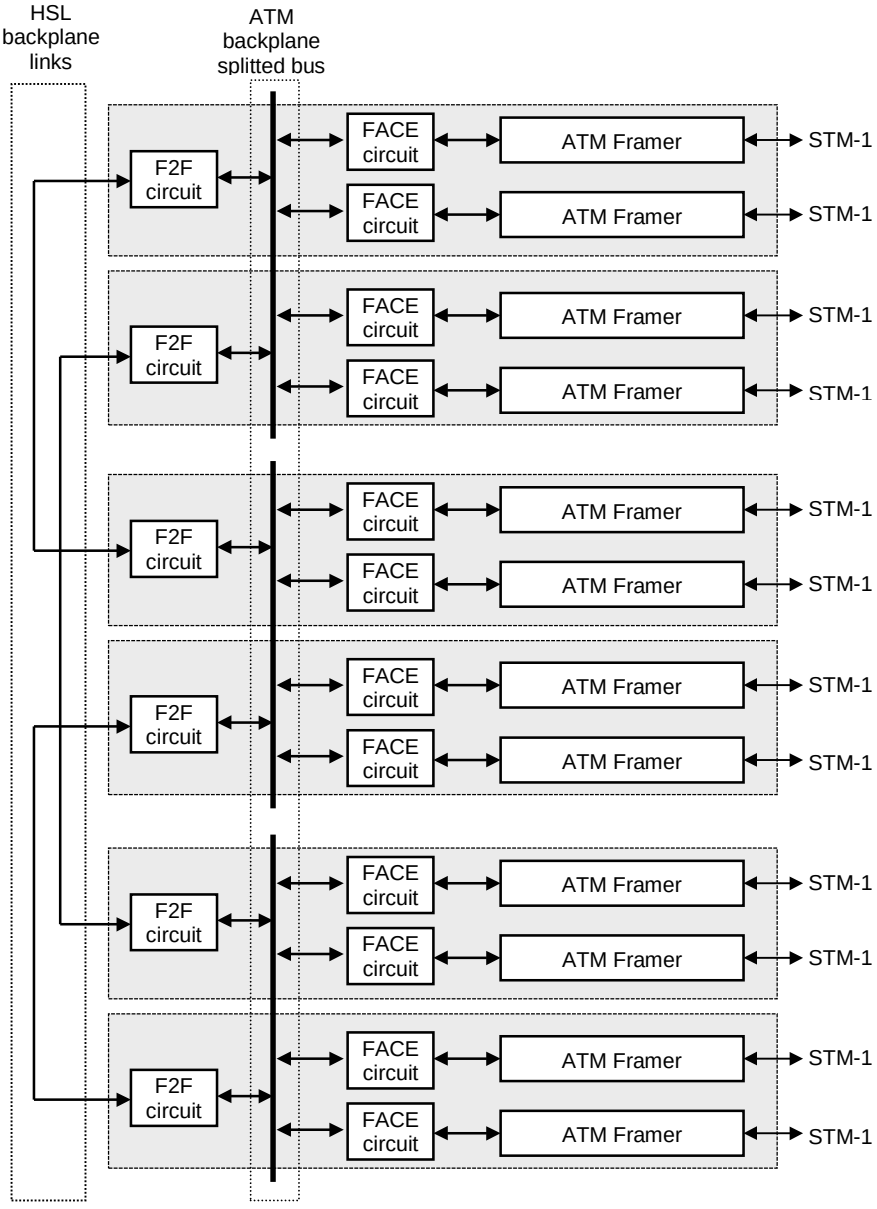

Figure 3.17 - Topologie du commutateur SafeCom4000X en configuration 1,8 Gb/s

3.3.3 Topologie du SafeCom4000X utilisant de liaisons point à multipoint. Problème de la qualité de service

En principe pour chaque liaison entre deux équipements, une paire de circuits F2F est nécessaire. La solution pour utiliser un nombre réduit de circuits F2F consiste en la réalisation de liaisons point à multipoint. Les flux entrants des autres sous-systèmes doivent donc être concentrés par un composant nommé Merger. Dans ce cas les messages doivent contenir des identificateurs de la source et de la destination.

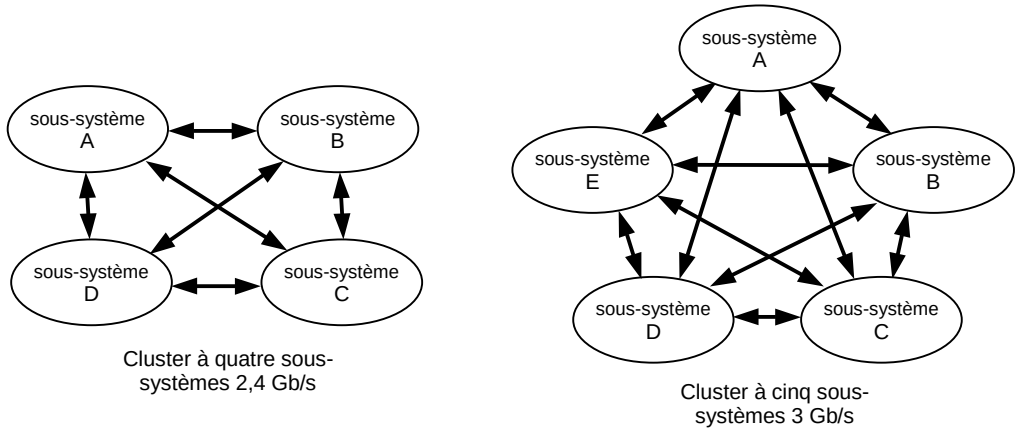

Figure 3.18 - Topologie à quatre et cinq sous-systèmes

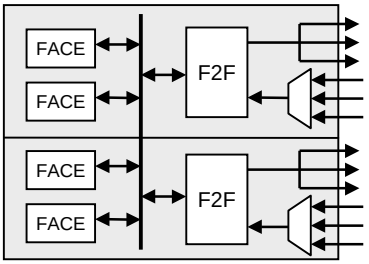

Figure 3.19 - Architecture du sous-système en version point à multipoint

Le volume de données issues d'un sous-système atteint une valeur maximale égale au débit maximal du cluster. La valeur moyenne du débit entrant ne dépasse pas la capacité du circuit F2F mais la concentration de plusieurs flux donne lieu à des débits instantanés très élevés. En conséquence, le concentrateur doit disposer d'une capacité de mémorisation. La difficulté de la réalisation d'une architecture point à multipoint consiste dans la mise en place de ce

tampon de cellules dans le composant Merger, tampon qui doit être capable de supporter un débit égal au débit maximal d'un cluster multiplié par le nombre de liens concentrés. Le passage de messages par cette mémoire tampon peut produire une dégradation de la QoS. L'architecture point à multipoint ne permet donc pas de garantir la même qualité de service que l'architecture SafeCom4000.

3.4 La montée en fonctionnalité du SafeCom4000X

3.4.1 Montée en fonctionnalités de niveau VC

L'architecture initiale est basée sur le composant FACE. Les blocs ICAF traitent les niveaux ATM PHY VP et VC. ISAF traite de l'interconnexion des terminaisons VP ou VC. Les blocs ISAF et ICAF s'interconnectent via le bus ATM disponible à l'extérieur de FACE et sur le fond de panier.

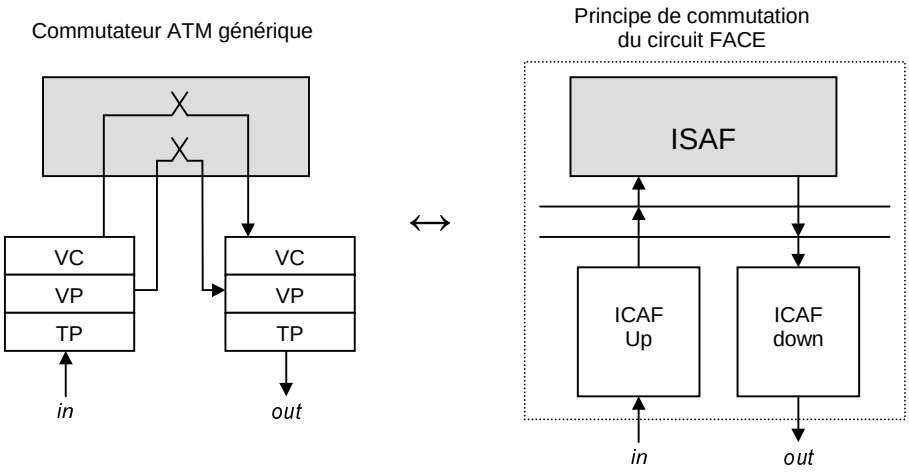

Figure 3.20 - Principe de commutation ATM du SafeCom4000

Le principe du SafeCom4000X permet d'ajouter des fonctions de niveau VC. En fait le traitement de la couche VC est reporté à l'extérieur de FACE dans le circuit F2F. L'ISAF et le niveau VC de FACE sont toujours utilisables simultanément à F2F. F2F augmente donc les capacités en y ajoutant des fonctionnalités au niveau VC. Deux blocs seront insérés dans les pipelines émission et réception du F2F. Le bloc d'extension VC Down placé dans la partie émission du circuit F2F permettra d'avoir des capacités accrues sur: le demultiplexage VC,

l'OAM F5 et l'UPC. Le bloc d'extension VC Up placé dans la partie réception du F2F augmente les capacités sur le *shaping* VC, le *scheduling* VC, l'OAM F5, les comptages de cellules VC (acceptées, refusées) et l'EPD.

Il existe différents moyens pour échanger des données entre un ICAF *Down* et un ICAF *Up* situés dans le même sous-système ou dans des sous-systèmes distants. Le schéma de la figure 3.21 donne les chemins pris par les flux de données. Le SafeCom4000X ajoute de nouveaux modes de transfert intra-cluster entre les blocs *Down* et *Up* sans en supprimer les modes de transfert locaux.

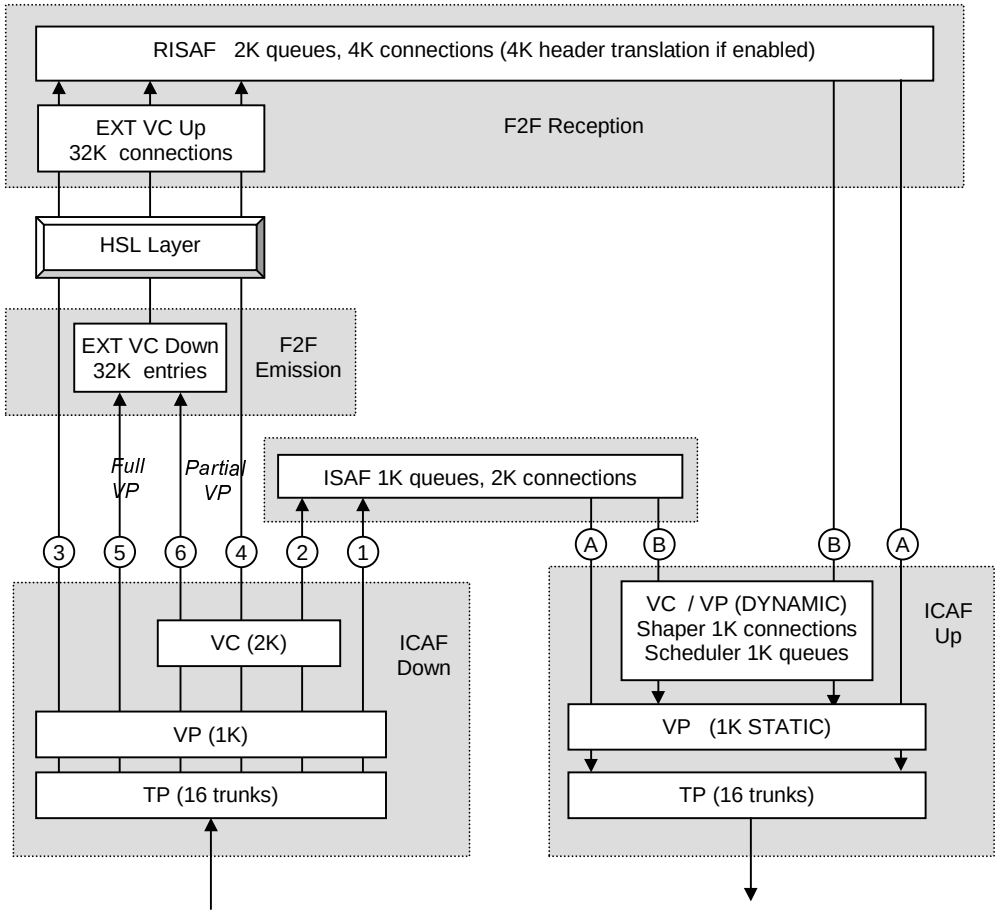

Figure 3.21 - Principe de commutation ATM du SafeCom4000X

Voici le résumé des types de flux qui peuvent être gérés dans le sens descendant :

1 - local *cross-connect* VP : le flux VP est émis vers un ISAF local ;

2 - local *cross-connect* VC : le flux VC est émis vers un ISAF local, la fonction EPD est réalisée par le circuit FACE ;

3 - cluster *cross-connect* VP : le flux VP est émis vers un RISAF distant ;
4 - cluster *cross-connect* VC : le flux VC est émis vers un RISAF distant, la fonction EPD est réalisée par le circuit F2F ;
5 - cluster *cross-connect* VC : le flux VP est émis vers EXT VC qui décode la totalité des canaux VC (*full VC*) contenus dans un tuyau VP. 32K VC peuvent être décodés ;
6 - cluster *cross-connect* VC : le flux VC non reconnu par le bloc VC de FACE est émis vers EXT VC qui peut décoder une partie (*partial VC*) des canaux VC contenus dans le VP, l'autre partie restant à la charge de FACE. 32K VC peuvent être décodés.

Le résumé des modes de transfert dans le sens montant :
A - *cross-connect* VP : le flux VP est émis de l'ISAF ou RISAF vers ICAF *Up*
B - *cross-connect* VC : le flux VC est émis de l'ISAF ou RISAF vers ICAF *Up*

Dans le mode de transmission 5 (*full VP*), le circuit FACE décode la couche VP exactement comme pour une *cross-connection* VP. F2F se charge de décoder la totalité des VC et de traiter la couche VC. Le fait que F2F décode la totalité des VC d'un VP implique que les fonctions VC de FACE ne sont pas utilisables sur l'ensemble des VC: F5, UPC, ABR *Virtual Destination*. Par contre l'UPC niveau VP et le F4 sont opérationnelles bien que les *cross-connections* soient de niveau VC.

Dans le mode de transmission 6 (partial VP), FACE décode les VC qui ne sont pas reconnus par FACE. F2F se charge de traiter la partie complémentaire des VC reconnus par FACE dans un VP donné. FACE « demande » à F2F de traiter les cellules qu'il ne reconnaît pas lui-même. Le fait que F2F décode une partie des VC d'un VP implique que les fonctions VC de FACE restent utilisables sur les VC décodés par FACE: F5, UPC, ABR *Virtual Destination*. Bien entendu, les VC traités par F2F ne peuvent pas se voir appliquer ces fonctions.

Le bloc EXT VC out s'applique toujours à la totalité des connexions y compris celles qui ne sont pas passées par EXT VC in. Cette caractéristique est utile pour la fonction EPD, réalisée obligatoirement par EXT VC out sur une connexion inter-clusters.

En sortie, la mise en forme du trafic (*shaping*) est réalisée par FACE et non par F2F. En conséquence le nombre de connexions mises en forme n'est pas augmenté dans la nouvelle architecture. Le bloc EXT VC permet essentiellement d'accroître le nombre de connexions

commutées dans la mesure où ces connexions sont seulement ordonnancées par le circuit FACE via des files d'attente partagées. Une connexion locale à un sous-système peut passer par F2F pour profiter de l'extension VC. Un re-bouclage interne à F2F réalise cette fonction sans que la connexion transite sur un lien physique HSL. Cette caractéristique est intéressante pour le SafeCom4000 lorsqu'une extension du nombre de connexions gérées est souhaitée.

3.5 La couche physique HSL

La couche physique HSL supporte deux modes de transport des données : cluster de type point à point sans concentration et cluster de type point à multipoint avec concentration.

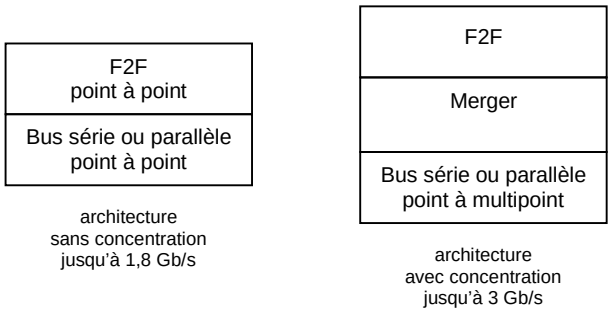

Figure 3.22 - Types d'architectures

Dans le cas d'une l'architecture sans concentration, un émetteur diffuse les cellules à un sous-système distant au travers d'un lien série point à point. Cette méthode limite le nombre de sous-systèmes à trois parce que le nombre de circuits F2F par sous-système devient trop important au delà de 3 sous-systèmes. De plus, la redondance n'est pas assurée. L'identification d'une destination est implicite parce que le lien est point à point.

Dans le cas d'une architecture avec concentration, un émetteur diffuse les cellules à l'ensemble des clusters distants au travers de liens séries point à multipoint mais supportant la même information. F2F constitue une source vers plusieurs destinataires et une destination de plusieurs sources. La fonction de concentration est faite par le circuit Merger.

La trame HSL est composée de deux messages, un message cellule et un message DBP qui contient l'information de rétro-propagation. Le format de la trame HSL est présenté dans l'annexe A.2.8.

Le sous-système émetteur émet une trame HSL pendant chaque plage temporelle c'est à dire toutes les 640 ns. La trame peut contenir des messages valides ou non. La durée d'émission du message est de 63 cycles de 50 Mhz ce qui représente 630ns. La trame est précédée d'une période de repos de 10ns. Cette période de repos est vue comme un caractère « idle » par la couche physique.

Le sous-système récepteur reçoit périodiquement une trame toutes les 640ns. Cependant, la période de repos de 10ns peut être réduite ou augmentée par la couche physique pour des raisons de plésiochronisme. Le récepteur se cadre en permanence sur les trames entrantes grâce à un caractère de synchronisation. Un code détecteur des erreurs est vérifié en permanence. Toute erreur entraîne l'écartement de la trame et est reportée dans un compteur de 16 bits sans dépassement. Le compteur peut être lu par le CPU pour évaluer la fiabilité de la couche physique. La lecture de ce compteur le remet à zéro.

Le médium HSL possède un service de redondance exposé dans l'annexe A.2.9.

3.6 Conclusions

L'architecture SafeCom4000X représente une évolution importante (en débit et en nombre de connexions gérées) par rapport à l'architecture initiale. L'avantage économique principal est que l'infrastructure initiale est réutilisée et que l'investissement est réduit au développement d'une seule carte et d'un seul composant VLSI et à quelques modifications du fond de panier. En vue d'une implémentation industrielle l'architecture bénéficie du dépôt d'un brevet [10].

Chapitre 4. Architecture du circuit F2F

Le chapitre présente l'architecture du circuit F2F. La fonctionnalité principale du circuit F2F est de relier entre eux deux bus ATM. Une liaison bidirectionnelles d'un débit de 660 Mb/s est crée entre les sous-systèmes organisés autour de ces deux bus ATM.

4.1 Généralités

Le service rendu par le circuit F2F aux couches supérieures de contrôle et management est similaire à celui rendu par le bloc ISAF du circuit FACE. La différence principale est que F2F est couplé sur des bus ATM différents en entrée et en sortie. Des fonctions supplémentaires au niveau VC y sont intégrées. Ces fonctions pallient les carences du circuit FACE sur certains points comme le nombre de connexions gérées. Les fonctions supplémentaires sont rassemblées dans un module interne à F2F et appelé EXT VC (Extensions VC). Le module EXT VC possède sa propre mémoire de contexte et a une conception évolutive, conception qui permet l'adjonction de nouvelles fonctionnalités pour prendre en compte les dernières évolutions de l'ATM. Le circuit F2F est réalisé en technologie FPGA [11].

4.1.1 Fonctionnalité générale du circuit

La réalisation du circuit F2F pose un certain nombre de problèmes issus principalement de la difficulté de l'adaptation du protocole du bus ATM au mécanisme de transfert bidirectionnel. La figure 4.1 présente l'interconnexion entre un sous-système A et un sous-système B, fonction rendue par l'utilisation de deux circuits F2F montés tête-bêche dans une configuration sans concentration.

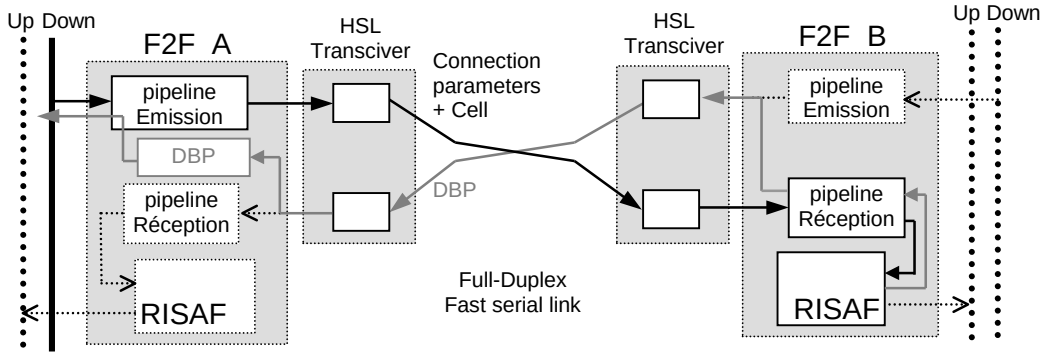

Figure 4.1 - Interconnexion entre sous-systèmes, transit de l'information DBP par le lien HSL

La partie émission du circuit F2F se trouve dans le sous-système source. Nommé F2F Down son rôle principal est de convertir les requêtes descendantes du bus ATM et les cellules associées en un flux parallèle unique facile à transporter sur un fond de panier. Les messages contiennent une cellule ATM préfixée par les paramètres de connexions et ceux de la file d'attente de destination.

Une fois le message arrivé dans la partie réception du circuit F2F B, les paramètres nécessaires à la génération d'une requête descendante sont extraits et la requête est envoyée vers le tampon de cellules RISAF qui répond à cette requête avec l'information de rétro-propagation. Revenant au processus d'émission du message, le protocole descendant du bus ATM, prévoit que l'ISAF récepteur doit envoyer une information liée à la taille de la file d'attente venant de recevoir la cellule courante. Cette information appelée DBP (*Down Back Pressure*) est utile au bloc ICAF du circuit FACE. Conformément au protocole du bus ATM, l'information DBP doit être renvoyé sur le bus ATM quelques centaines de nanosecondes après la requête initiale d'émission de la cellule. Le transit par le lien HSL ne permet pas de fournir si rapidement l'information DBP parce que la file d'attente est déportée dans le RISAF distant. Cette émission relative au taux de remplissage de file d'attente est donc impossible. La solution consiste en l'utilisation d'une mémoire qui stocke l'information DBP dans la partie émission. Le chemin gris de la figure 4.1 présente l'acheminement de l'information DBP à partir de RISAF à travers la liaison HSL vers la mémoire DBP et par l'intermédiaire d'une lecture de cette mémoire vers le bus ATM. Le mécanisme de mémorisation de la DBP produit une erreur d'une unité sur l'information sur la dimension des files d'attente parce que dès l'émission d'une cellule i, il devient possible d'avoir le taux de

remplissage correspondant à la cellule i-1. Cette erreur n'affecte pas dans une grande mesure l'information DBP qui est codée sur trois bits. F2F est prévu pour fonctionner dans des configurations avec un maximum de 7 sous-systèmes distants. La mémoire DBP doit donc gérer les taux de remplissage de 7 RISAF distants chacun pouvant avoir jusqu'à 2048 files d'attente. F2F dispose donc d'une mémoire interne de 48Kb (8 x 2048 x 3 bits).

La partie réception du circuit F2F se trouve dans le sous-système destination. Nommée F2F Up, sa fonction principale est de convertir un flux parallèle unique issu du fond de panier en des requêtes du bus ATM avec ses cellules correspondantes. Le protocole de transfert descendant est reproduit à l'intérieur du circuit F2F pour permettre de s'interfacer avec le bloc RISAF. A la partie réception revient également la charge de récupérer l'information DBP fournie par le RISAF et de l'envoyer via le lien HSL au sous-système émetteur.

4.1.2 F2F en mode de fonctionnement avec rebouclage

Le circuit F2F ne permet pas uniquement d'augmenter le débit mais grâce aux fonctionnalités EXT VC, il offre aussi la possibilité d'accroître le nombre de connexions commutées. Il peut être utilisé uniquement pour cette fonctionnalité et dans ce cas il fonctionne avec un rebouclage interne. Une seule carte SUATMX avec un module F2F suffit pour servir la totalité des cartes ATM de l'équipement.

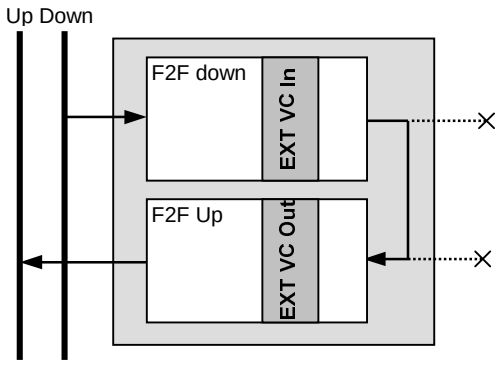

Figure 4.2 - F2F en mode de fonctionnement avec rebouclage interne

En configuration avec rebouclage interne, F2F ne gère plus le mécanisme de rétro-propagation décrit précédemment. Les fonctions ATM qui utilisent la rétro-propagation seront

supprimées car elles n'auront pas l'information nécessaire pour limiter leur production de cellules.

4.2 Description fonctionnelle du circuit F2F

La figure 4.3 donne une représentation fonctionnelle du circuit F2F pour mettre en évidence les blocs, leurs interconnexions et les interfaces système.

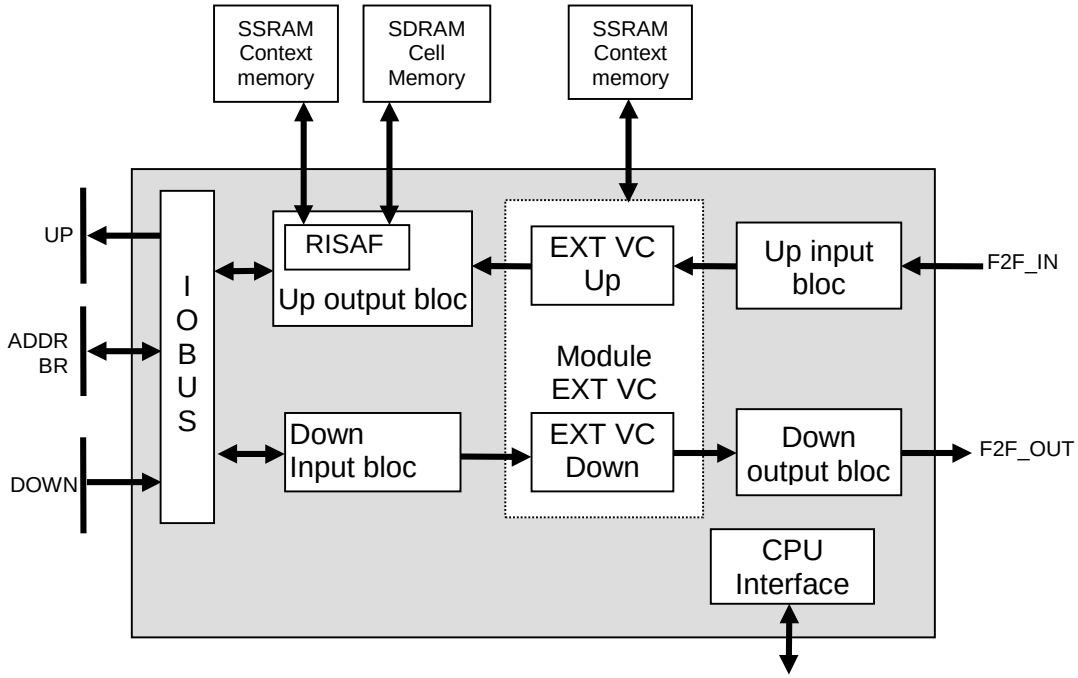

Figure 4.3 - Blocs fonctionnels de F2F

4.2.1 Le transit du bus ATM vers le lien HSL

Le bloc d'entrée, DOWN IN, assure l'interfaçage avec la partie descendante du bus ATM. Les requêtes descendantes DREQ arrivent par l'intermédiaire du bus de contrôle ADDR. Le format de la requête est décrit dans l'annexe A.2.1. L'adressage à travers le bus ATM est fait à l'aide d'un identificateur. Conformément au protocole du bus ATM, F2F répond à toutes les requêtes avec l'information DBP sur sa propre ligne du bus de contrôle BR. La cellule est ensuite envoyée sur le bus ATM. Le rôle du bloc d'entrée est donc de préfixer la cellule avec

un numéro de connexion et un numéro de file d'attente, informations récupérées de la requête descendante. Le format de la trame à la sortie de ce bloc est décrite dans l'annexe A.2.2. Une autre information importante, récupérée de la requête, est un bit nommée DEMUX qui valide la fonction d'extension VC. La trame ainsi crée est ensuite émise vers le bloc suivant, EXT VC.

Par rapport à l'architecture système du SafeCom4000, le circuit F2F permet l'extension du nombre de connexions commutées. Le nombre de connexions est fixé à 32K d'unidirectionnelles et donc à 16K de bidirectionnelles. Le traitement est divisé en deux parties : l'une réalisée sur le chemin d'émission et l'autre sur le chemin de réception. La partie EXT VC émission exécute le décodage VP/VC, décodage activé selon le bit DEMUX. La fonction UPC n'est pour l'instant pas intégrée dans le circuit, elle représentera une évolution ultérieure et sera activée par un bit d'activation dans le contexte du VC.

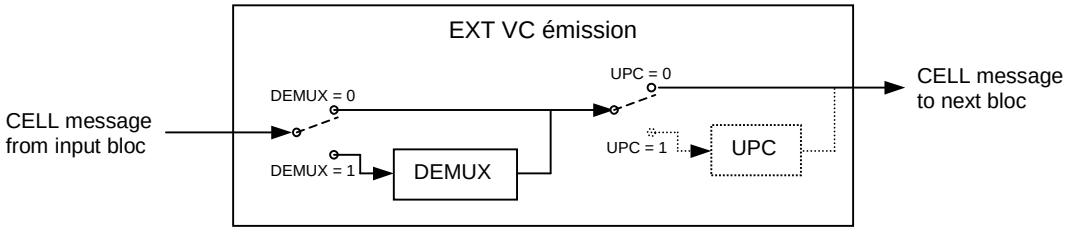

Figure 4.4 - Bloc EXT VC émission

L'algorithme de décodage EXT VC émission est détaillé dans l'annexe A.2.4.

Le bloc de sortie du pipeline descendant est chargé de former et d'envoyer la trame HSL. La trame contient en effet deux messages : le message CELL et le message DBP. Si ni le message CELL, ni le message DBP ne sont actifs, une trame vide est transmise une plage temporelle sur deux pour maintenir la liaison HSL.

4.2.2 Le transit du lien HSL vers le tampon RISAF

Le bloc entrant de la partie réception est chargé de synchroniser les trames entrantes et de les mettre au format interne de F2F. Le relais est pris par le bloc EXT VC UP qui effectue le comptage des cellules commutées et des cellules perdues par saturation. Cette évaluation du taux de pertes de cellules est effectuée pour assurer la compatibilité avec une architecture

avec concertation, architecture qui présente le risque de perte de cellules. Deux fonctions optionnelles sont aussi implémentées: la translation d'en-tête VPI/VCI et la fonction EPD (*Early Packet Discard*).

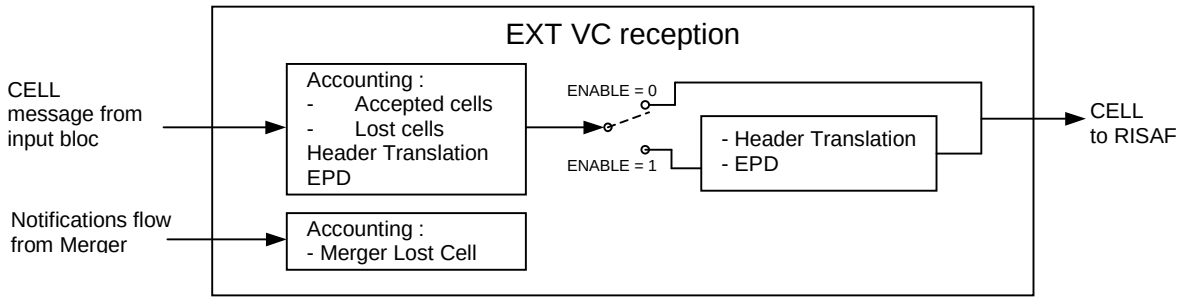

Figure 4.5 - Bloc EXT VC réception

La fonction de translation d'en-tête et la fonction EPD sont activées à chaque occurrence d'une cellule par des bits d'activation configurés par le processeur dans la mémoire de contexte. Ces fonctions sont indépendantes de la valeur du bit DEMUX. Pour une connexion donnée, l'activité EXT VC émission n'implique pas l'activité de EXT VC réception. A noter que le comptage et la translation peuvent être réalisées par RISAF. Par contre l'EPD n'étant pas opérationnel dans RISAF, il doit être fait par EXT VC.

EXT VC dispose d'une mémoire privée SSRAM qui contient les paramètres de contexte des connexions et de démultiplexage. La mémoire est utilisée par les blocs EXT VC émission et EXT VC réception.

4.2.3 Fonctionnalités du bloc RISAF

RISAF fait partie intégrante du conduit de réception du circuit F2F. RISAF s'inteface avec EXT VC en utilisant le protocole descendant du bus ATM à l'intérieur du circuit et avec le bus ATM local. RISAF accepte des requêtes de transferts descendants provenant d'un sous-système distant et des requêtes de transferts montants de la part de toutes les ICAFs connectés au bus ATM du sous-système local.

RISAF est un gestionnaire de files d'attente. L'architecture du bloc RISAF est inspirée de l'architecture du bloc ISAF du circuit FACE. RISAF supporte un débit de 660 Mb/s en entrée et en sortie. La capacité maximale du tampon est de 64K de cellules et il peut gérer et translater 4K de connexions par rapport à la capacité de 32K de cellules / 2K de connexions de ISAF. Le bloc RISAF découple le routage physique du routage logique. Le routage physique consiste en l'insertion et l'extraction des cellules à partir des files d'attente et le routage à travers le bus ATM. Le routage logique consiste en la translation d'en-tête VPI/VCI. En pratique, le découplage se traduit par le fait que plusieurs connexions VPI/VCI différentes peuvent partager la même file d'attente. RISAF peut donc fonctionner en complémentarité avec EXT VC : le routage physique est toujours confié à RISAF mais le routage logique, qui consiste dans la translation d'en-tête VPI/VCI, peut être réalisé par EXT VC. Le bloc RISAF est informé par l'intermédiaire d'un signal que les translations VP et VC de certains cellules ont été réalisées par EXT VC. Le fait que l'extension d'adresse EXT VC décode 32K de connexions entraîne obligatoirement, dans la majorité des cas, une translation exécutée par EXT VC au lieu de RISAF sauf pour les cellules destinées à la diffusion.

4.2.4 Interfaces du circuit F2F

F2F s'interface directement avec le bus ATM présent sur le fond de panier. La structure et le fonctionnement du bus ATM ont été décrits dans le troisième chapitre, paragraphe 3.2. L'interface HSL consiste en deux bus parallèles de 16 bit de large synchronisés à 50 Mhz. La synchronisation niveau trame est opérée par 2 signaux de début de trame. La définition de cette interface est donnée dans l'annexe A.2.10.

RISAF utilise une mémoire SSRAM de 8Mb cadencée à 50Mhz pour stocker les informations de contexte relatives au files d'attente. Les files d'attente utilisent une mémoire SDRAM de 32Mb cadencée à 50Mhz. Une autre mémoire de contexte est nécessaire pour le traitement EXT VC. Les détails d'implémentation sont donnés dans l'annexe A.2.10.

Le circuit F2F est capable de fonctionner dans des architectures avec concentration des flux de type point à multi-point. Le circuit peut donc se connecter au maximum à 7 sources différentes. Pour pouvoir comptabiliser les pertes de cellules due à la contention, F2F

s'interface avec le circuit de concentration Merger par l'intermèdiaire de 8 lignes série cadencées à 50Mhz. Ces lignes sont indépendantes et les notifications de perte cellules de Merger vers F2F sont envoyées de manière asynchrone par rapport au cadrage des plages temporelles, à l'aide d'un signal série. Chaque ligne série correspond à un débit de 660 Mb/s ce qui permet de comptabiliser toutes les éventuelles pertes de cellule venant d'une source distante. La concentration des flux peut nuire aussi à l'information de rétro-propagation DBP. Pour l'éviter, le circuit Merger peut directement envoyer à F2F l'information DBP etle numéro de la file d'attente à laquelle elle fait référence par l'intermédiaire des autre 8 liens séries asynchrones. La fréquence de notification de chaque ligne DBP correspond à un flux cellule de 660Mb/s. Les interfaces de Merger Lost Cell et BDP sont détaillées dans l'annexe A.2.10. L'interface CPU est classique et peut traiter des transferts simples de type mot. Les détails sont présentés dans l'annexe A.2.10.

Le document [12] présente en détail la fonctionnalité du circuit.

4.3 Architecture du circuit F2F - présentation structurelle

Le circuit F2F contient deux pipelines : un pipeline émission nommé *DOWN* et un pipeline montant nommé *UP*. Chaque étage du pipeline a une durée égale à une plage temporelle et opère sur une seule cellule ATM.

4.3.1 Synchronisation du circuit F2F

Pour pouvoir se cadrer sur les cellules transférées sur le bus ATM, les étages du pipeline sont synchrones avec les plages temporelles du bus ATM. Le bus ATM externe et le bus CPU sont synchronisés à une horloge de 25Mhz. La fréquence de fonctionnement du circuit F2F est donc double par rapport à la fréquence du bus ATM ce qui permet de disposer d'un nombre double de cycles par étage du pipeline. Un horloge système à 50Mhz est généré à l'intérieur du circuit FPGA à l'aide d'un boucle DLL. L'horloge à 25Mhz du fond de panier est uniquement utilisée à l'intérieur du circuit comme signal de validation.

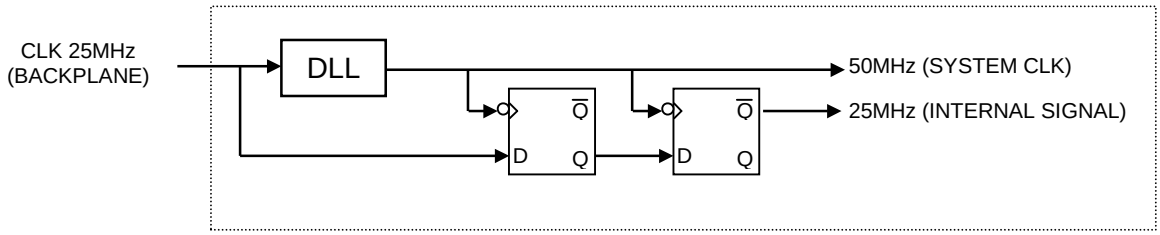

Figure 4.6 - Horloges système du circuit F2F

Toutes les entrées et les sorties du circuit F2F sont échantillonnées sur le front montant du signal CLK50. Les options de synthèse du circuit utilisent les bascules des modules IOB du FPGA Virtex ce qui permet d'assurer les mêmes temps de *setup* et de *hold* quelque soit l'entrée ou la sortie considérée.

Les plages temporelles du bus ATM ont 16 cycles et les étages du pipeline disposent chacun de 32 cycles. Une cellule peut arriver du bus ATM vers le circuit F2F tous les 16 cycles d'horloge en mots de 32 bits, à une fréquence de 25Mhz soit une toutes les 640ns. La même cadence cellule se retrouve du circuit F2F vers le bus ATM (pipeline réception).

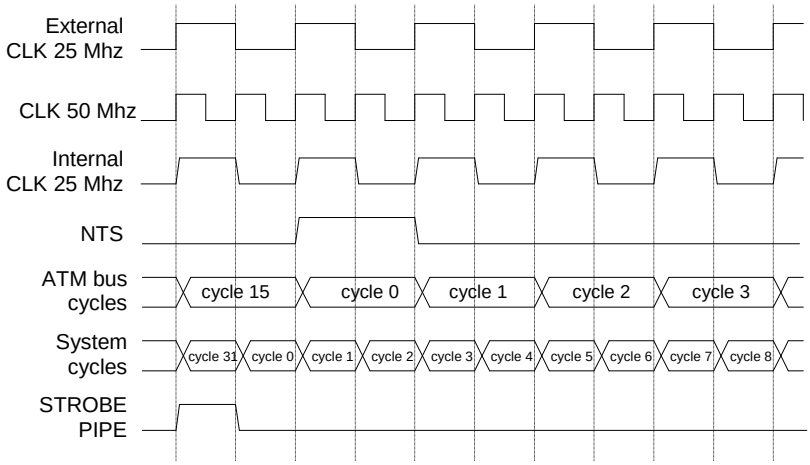

Figure 4.7 - Signaux de synchronisation

Le bus ATM possède un signal nommé NTS (*New Time Slot*) qui définit la synchronisation au niveau cellule. Ce signal est actif tout les 16 coups d'horloge. L'horloge interne ayant une fréquence double, le nombre de cycles disponibles pour traiter une cellule est aussi double, la numérotation des cycles se fait de 0 à 31. Les transferts d'un bloc à un autre sont référencés

par rapport à cette numérotation et sont constants dans le temps. Les exceptions à cette règle doivent être réduites aux événements asynchrones comme les accès CPU et les lignes en série. Par défaut et dans la mesure du possible, les cellules et les autres informations de contrôle sont transférées d'un bloc à l'autre au cycle 31. Cette synchronisation des étages du pipeline est assurée par le signal STROBE PIPE. Le temps écoulé entre deux occurrences de STROBE PIPE à donc la même durée qu'une plage temporelle du bus ATM. STROBE PIPE est actif au cycle 31 et valide les transferts. Les signaux de contrôle sont donc repris par le bloc suivant au cycle 0. Au regard des débits écoulés par F2F, une cellule rentre et une autre sort à chaque plage temporelle. Par contre le temps de traitement d'une cellule prends plusieurs plages temporelles.

4.3.2 Organisation de l'architecture

Le circuit F2F est réalisé en technologie FPGA Xilinx Virtex. L'utilisation de la famille de composants Virtex de Xilinx offre, à part la compatibilité avec les autres circuits de l'équipement et avec les caractéristiques du fond de panier, la possibilité d'utiliser des mémoires double port. Cette utilisation permet un nombre élevé d'accès ce qui rend possible l'implémentation des pipelines de F2F autour de ces blocs mémoire. Les étages du pipeline effectuent les opérations sur la cellule qui les concerne en utilisant un pointeur vers la zone de mémoire où se trouvent leurs données respectives. Le pipeline émission est organisé autour d'un bloc nommé DOWN RAM et le pipeline réception autour d'une mémoire nommé UP RAM. DOWN RAM et UP RAM ont la même structure. Elle peuvent stocker 8 trames et elles ont donc la dimension 8 x 512 bits = 4Kb. Chacun des deux port est un port de lecture/écriture. Un des ports est multiplexé temporairement entre deux sources, de sorte que 4 entités puissent accéder à la mémoire. Le partage de la mémoire est effectué cycle à cycle.

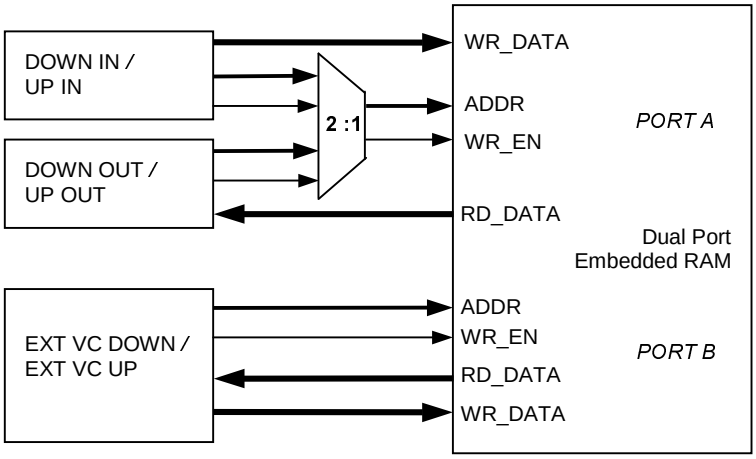

Figure 4.8 - Accès à la mémoire double port

Pour les deux pipelines, les blocs IN et OUT partagent le même port de la mémoire. Le bloc IN utilise les cycles impaires du port A pour écrire les paramètres et la cellule récupérés du bus ATM, tandis que les cycles paires sont affectés à DOWN OUT. Le port B est entièrement attribué au bloc EXT VC. Les étages du pipeline utilisent un pointeur sur 3 bits nommé CELL PTR. CELL VAL est un signal de validation de la cellule. CELL PTR et CELL VAL caractérisent chacune des 8 trames stockées dans la mémoire. Ces deux arguments sont donc passés d'un étage du pipeline à un autre. L'adresse physique des mots de 32 bits de la cellule est constituée par la concaténation du pointeur CELL PTR et d'un *offset* de 4 bits. La figure 4.9 est une représentation structurelle de l'architecture du circuit F2F organisée autour des mémoires UP RAM et DOWN RAM.

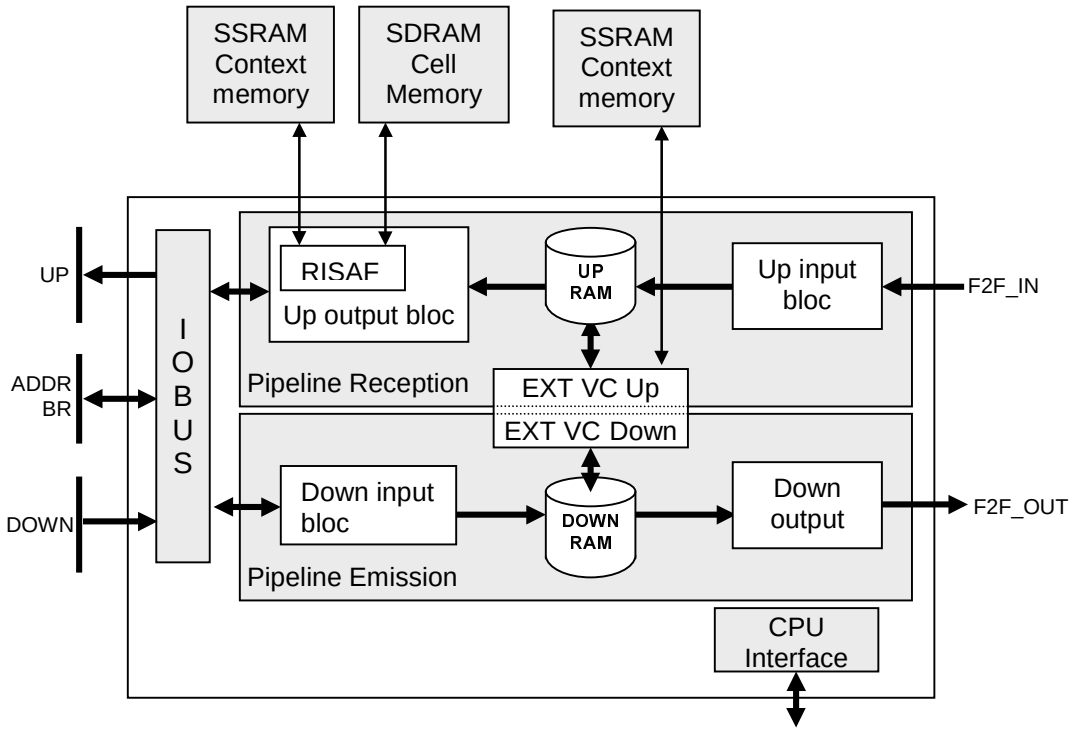

Figure 4.9 - Architecture du circuit F2F - présentation structurelle

4.3.3 Le pipeline émission

La figure 4.10 présente les 8 étages du pipeline d'émission, leur distribution dans les blocs fonctionnels et les zones de mémoire DOWN RAM sur lesquels ils effectuent des opérations.

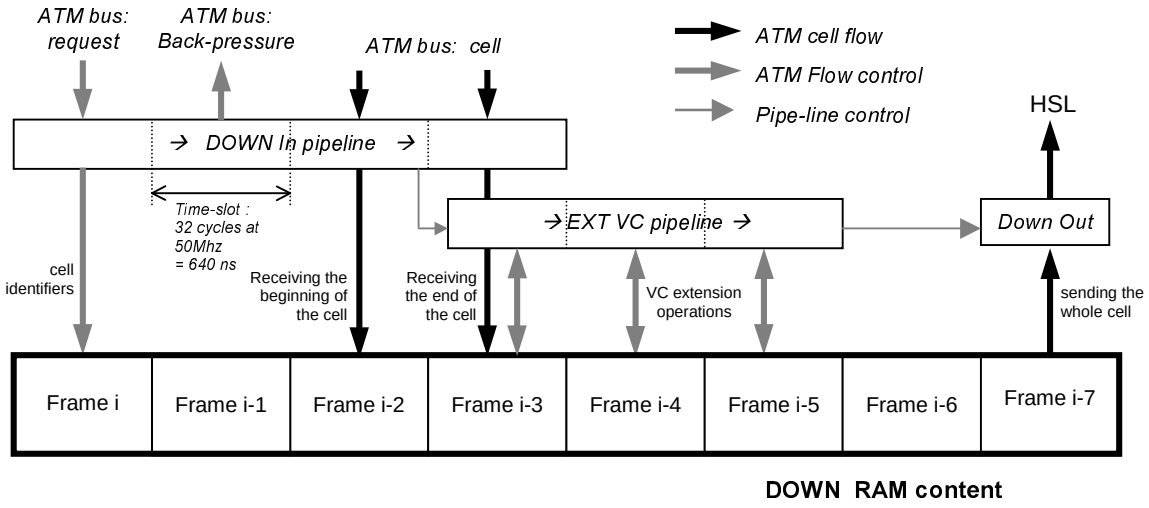

Figure 4.10 - Pipeline émission

Le fonctionnement du pipeline émission est basé sur un compteur en boucle nommé DLCC (*Down Local Cell Counter*), qui est incrémenté à chaque occurrence du signal STROBE PIPE. La valeur de DLCC varie de 0 à 7 et définit le pointeur CELL PTR appartenant au premier étage du pipeline. Le signal de validation CELL VAL indique la validité de la cellule. A chaque occurrence de STROBE PIPE les informations de contrôle du pipeline, c'est à dire la valeur du pointeur et la validité de la trame, sont passées simultanément de chaque étage du pipeline à l'étage suivant.

Le bloc d'entrée nommé DOWN IN dispose d'une logique de reconnaissance des trames qui lui sont destinées. Cette reconnaissance est faite à partir d'un identificateur contenu dans la requête descendante DREQ, qui précède la cellule. Le CPU consigne dans un registre du circuit cet identificateur. A chaque circuit destinataire correspond également un bit dans un champ de bits de la requête. La présence des valeurs binaires vraies dans ce champ de bits est prioritaire sur l'information codée dans le champ identificateur. Ce mécanisme parallèle d'adressage est mis en place pour permettre la diffusion d'une cellule vers plusieurs destinataires. Le numéro de file d'attente et l'identificateur de connexion sont récupérés de la requête descendante et écrits dans la page de mémoire correspondante au pointeur courrant en début de la trame HSL. Compte tenu du protocole du bus ATM, l'information DBP, stockée dans DBP RAM, est retournée sur le bus BR. Le bloc d'entrée est concerné seulement par la lecture de l'information DBP. Cette information est actualisée périodiquement par le bloc UP IN qui la reçoit via le lien HSL. Le CPU a la possibilité de remettre à zéro l'information DBP. Les étages 2 et 3 du pipeline stockent les mots de 32 bits de la cellule associée dans DOWN RAM au fur et à mesure que les mots arrivent, c'est à dire un mot de 32 bits tous les 2 coups d'horloge cadencée à 50Mhz. L'adresse physique d'écriture est composée à partir du pointeur courant. Lorsque le bloc d'entrée a terminé les opérations sur le message et à l'occurrence du signal STROBE PIPE, l'information de contrôle du pipeline est transférée au bloc EXT VC. Le traitement EXT VC décrit dans l'annexe A.2.4 est lui-même pipeliné. Une parties des cycles d'accès à la mémoire DOWN RAM étant non-utilisés, la conception du bloc EXT VC permet une évolution sans impact sur les blocs périphériques.

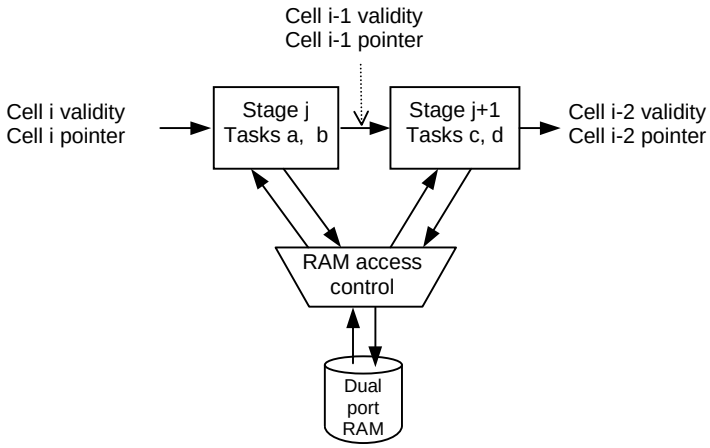

Figure 4.11 - Accès multiplexé des étages du pipeline

Les informations transitant entre le bloc EXT VC et les blocs d'entrée et de sortie sont limitées aux informations de contrôle du pipeline c'est-à-dire le signal de validité CELL VAL et le pointeur CELL PTR.

Le bloc de sortie nommé DOWN OUT prend le couple (CELL VAL, CELL PTR) à l'occurrence du signal STROBE PIPE. Le message est alors lu de DOWN RAM par mot de 16 bits et envoyé au fur et à mesure par le port de sortie F2F OUT. Le message DBP provenant de UP OUT est introduit dans la trame. Un code détecteur des erreurs est calculé à partir de tous les octets de la trame et inséré à la fin. Si ni le message CELL ni le message DBP ne sont actifs, une trame vide est transmise une plage temporelle sur deux. Les trames vides permettent le maintien du témoin de continuité prouvant que la ligne physique n'est pas coupée. D'autre part le repos dans l'émission de trames vides permet aux drivers HSL d'émettre des caractères IDLE et donc de se re-synchroniser sur ceux-ci au cas où la synchronisation HSL de bas niveau serait perdue.

4.3.4 Le pipeline réception

Le pipeline émission et le pipeline réception possèdent le même nombre d'étages. La figure 4.12 présente la distribution des étages dans les blocs fonctionnels et les zones de la mémoire UP RAM sur lesquels ils effectuent des opérations.

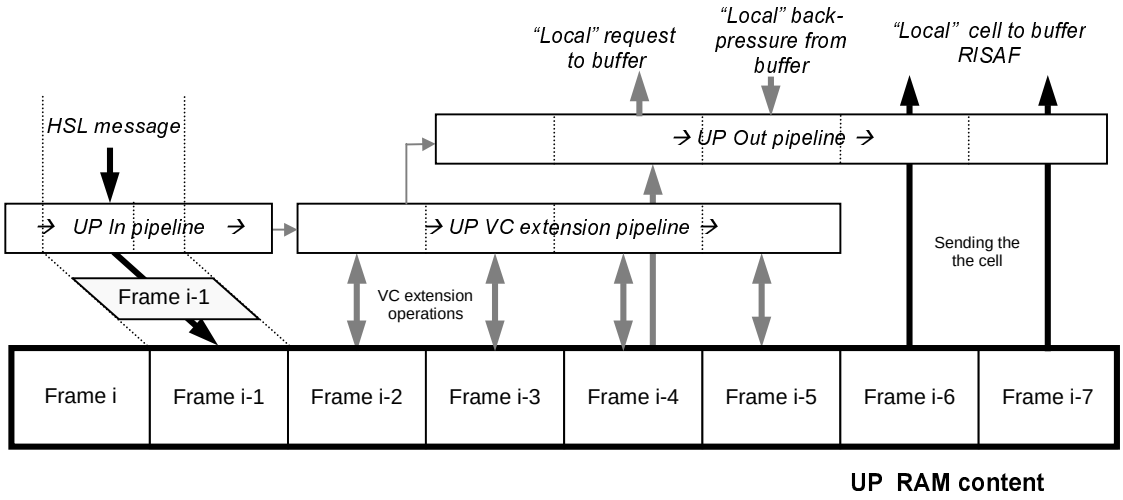

Figure 4.12 - Pipeline réception

La synchronisation des étages des pipelines réception et émission est identique. Un compteur en boucle nommé ULCC (*Up Local Cell Counter*) est incrémenté à chaque occurrence du signal STROBE PIPE. La valeur de ce compteur donne le pointeur CELL PTR du premier étage de pipeline.

Le bloc d'entrée, UP IN reçoit les trames provenant du lien HSL, les convertit et les écrit en mot de 32 bits dans la mémoire UP RAM. Une fonction importante du bloc Up In est de réaliser un cadrage des messages reçus de l'interface HSL avec les plages temporelles du sous-système récepteur. La synchronisation de trames au niveau horloge est assurée par le composant d'interface HSL. Au niveau plage temporelle, le circuit F2F ne possède aucune information sur le retard en nombre de cycles du premier mot de la trame par rapport au signal STROBE PIPE. Pour cette raison le bloc d'entrée possède deux étages de pipeline pour la réception de la cellule. La synchronisation de ces deux étages de pipeline est particulière car pendant les deux étages l'écriture de la trame est effectuée en utilisant le même pointeur, celui de la trame i-1. Le bloc d'entrée fait la validation de la trame écrite en mémoire en fonction du résultat du calcul du code détecteur des erreurs qui se trouve en fin de trame. Les trames dont le code détecteur est erroné, sont éliminées.

Le traitement de l'extension VC s'étend sur quatre étages du pipeline et l'émission de la requête et de la cellule s'étends sur cinq étages de pipeline. Ceci est possible parce que l'émission de la requête vers le tampon RISAF n'est pas affecté par les taches effectuées sur le champs VP et VC de la cellule. Le traitement EPD (*Early Packet Discard*) est

complètement effectué dans le premier étage du pipeline du bloc EXT VC parce qu'il conditionne l'émission de la requête. Les opérations de comptage de cellules sont faites par EXT VC. Les informations transitant entre les étages du pipeline sont limitées au pointeur et à la validité de la cellule. La mise en invalidité d'une cellule par la fonction EPD se traduit par la remise à zéro du signal CELL VAL. Le bloc de sortie effectue la lecture des paramètres d'identification de la file d'attente et de connexion pour reconstituer une requête DREQ et l'envoyer vers le module interne RISAF. La cellule est ensuite lue par mot de 32 bits et envoyée en interne vers le tampon RISAF en utilisant le protocole de communication du bus ATM. L'information DBP retournée par RISAF est introduite par le bloc DOWN OUT dans la trame HSL courante ce qui conclut l'algorithme du pipeline réception.

4.3.5 Le bloc CPU

Le bloc CPU permet l'accès par un processeur aux ressources interne de F2F. Ce contrôle de ressources est réalisé à l'aide d'un ensemble de registres distribués dans les blocs F2F. L'interface CPU est classique, cadencée à 25Mhz. Il existe tout même un bus CPU interne cadencé à 50 Mhz, bus qui permet l'accès aux registres des différents blocs. Certains blocs n'ont pas accès au bus CPU interne mais quelques informations sont fournies directement par les registres du bloc CPU. Parmi ces informations, l'option de rebouclage est contrôlée par un bit d'un des registres du bloc CPU. Quand cette option est activée, l'entrée de réception UP IN est connectée en interne à la sortie F2F OUT. Il existe aussi un bit SOFTWARE RESET qui permet au processeur de réinitialiser le circuit.

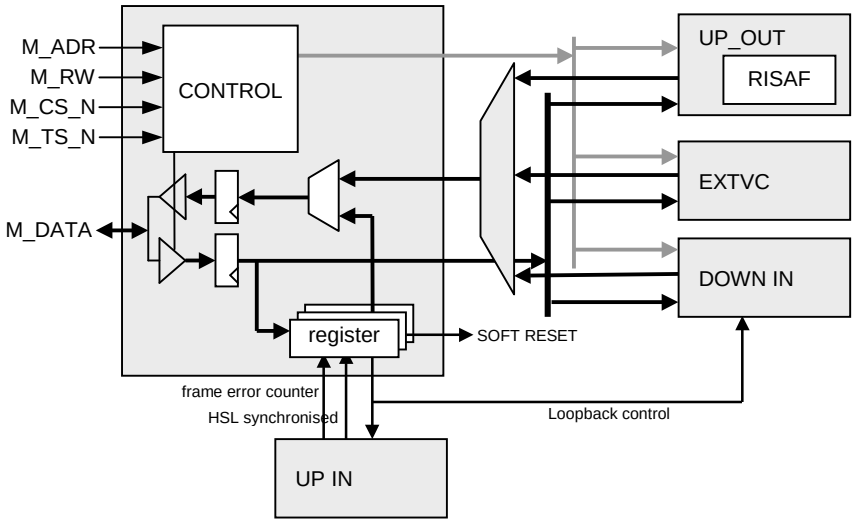

Figure 4.13 - Bloc CPU

4.3.6 L'accès à la mémoire EXT VC

Trois blocs accèdent à la mémoire EXT VC: EXT VC émission, EXT VC réception et CPU. Les accès sont multiplexés d'une manière cyclique et fixés par un séquenceur synchronisé avec le compteur de cycles. Tout ces signaux passent par un NON-OU logique. Les blocs fournissent un état logique '0' lorsque le cycle mémoire ne leur appartient pas. Les signaux de contrôle actifs à l'état bas sont fournis par les blocs à l'entrée du NON-OU en logique positive.

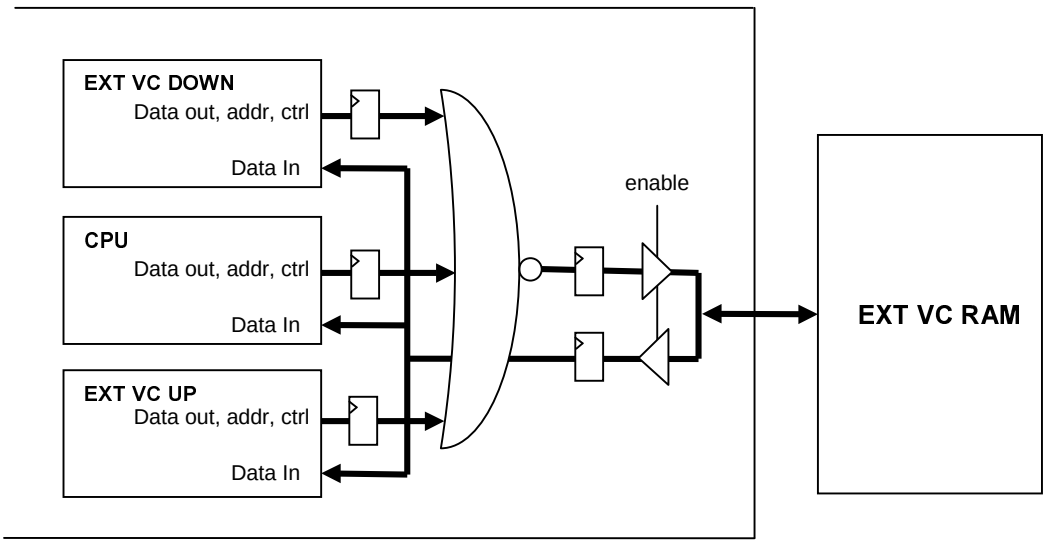

Figure 4.14 - Accès partagé de la mémoire EXT VC

Le document [13] donne plus de détails sur l'architecture matérielle du circuit.

4.4 Conclusions

Le chapitre présente l'architecture matérielle du composant F2F que nous avons développée. En vue de la synthèse matérielle, l'architecture du circuit est décrite en langage VHDL. Les outils de développement FPGA Express et Xilinx Alliance ont été utilisés pour les simulations, la synthèse et la configuration du FPGA. L'architecture profite des facilités des blocs de mémoire SelectRAM et des modules IOB (Input Ouput Buffer) offertes par la famille de composants Vitex de Xilinx [14]. Le circuit utilise 3385 CLB (Cell Logic Block), 20 blocs SelectRAM et deux circuits DLL (Delay-Locked Loop). La fréquence maximale de fonctionnement du circuit est de 56 MHz.

Chapitre 5. Architecture de commutateur extensible utilisant une matrice à commutation de paquets

Les chapitres 3 et 4 ont montré comment une architecture de commutateur utilisant une technique de bus partagé et reposant sur le principe de la commutation de circuit peut être étendue - sans dégradation des garanties de service - pour augmenter le nombre de connexions. Mais cette étude montre clairement que les possibilités d'extension restent limitées par le principe même du bus partagé.

Ce chapitre, s'intéresse à une architecture de commutateur organisée autour d'une matrice de commutation de type réseau multi-étages utilisant le principe de la commutation de paquets. Ce type d'architecture est - par principe - beaucoup plus modulaire et possède donc de bonnes caractéristiques d'extensibilité, mais elle complique fortement le problème de la garantie de services, puisqu'il n'y a plus de réservation explicite de bande passante. L'étude fait donc l'analyse des différentes techniques permettant de garantir un certain niveau de qualité de services.

Tout ce chapitre, considère l'hypothèse que toute l'intelligence du commutateur est située à la périphérie de la matrice, c'est à dire dans les contrôleurs des ports d'entrée et de sortie. En particulier, tous les algorithmes de gestion de la QoS sont réalisés au niveau des ports. La matrice de commutation est considère donc comme une "boîte noire" qui fournit pour seule garantie un acheminement des paquets sans perte.

5.1 La qualité de services dans les matrices à commutation de paquets

Comme il a été montré au chapitre 2, les matrices de commutation de paquets permettent de réaliser des architectures extensibles avec des débits élevés. L'utilisation simpliste d'une matrice à commutation de paquets repose sur le scénario suivant : les en-têtes des paquets entrants sont analysés au niveau de chaque port d'entrée. Le contrôleur du port d'entrée ajoute,

en tête du paquet, une étiquette désignant le port de sortie correspondant (cette étiquette est obtenue par recherche associative dans des tables dont le contenu a été défini lors de l'acceptation des connexions). Les paquets sont alors injectés dans la matrice, sans aucun contrôle du trafic injecté, et la matrice se charge de leur acheminement vers le port de sortie visé. Les paquets arrivant sur un port de sortie sont débarrassés de l'étiquette qui a servi à traverser la matrice par le contrôleur du port de sortie et émis dans l'ordre où ils arrivent.

Cette approche simpliste est évidemment inefficace pour deux raisons : Dans un réseau multi-étages, le débit effectif commuté par la matrice est toujours inférieur au débit maximal (qui est égal à la somme des débits maximaux des ports d'entrée), à cause des contentions dans le réseau. La contention est causée par un débit instantané concentré vers un port de sortie supérieur à sa capacité. Si plusieurs paquets sollicitent le même port de sortie il y a forcément un arbitrage qui bloque un des paquets et implicitement tous les autres paquets qui proviennent du même port d'entrée. Les mécanismes de rétropropagation de la contention (*backpressure*) permettent d'éviter les pertes de données dans la matrice, mais ne permettent pas d'éviter le phénomène d'engorgement de la matrice si un trafic trop important est injecté par rapport à ce que la matrice peut accepter.

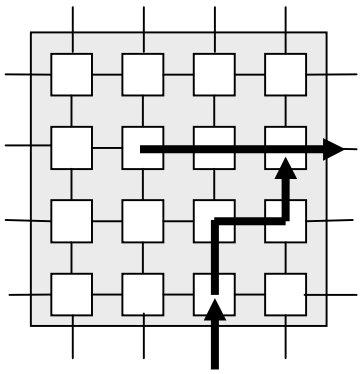

Figure 5.1 - Congestion en sortie d'une matrice de routage

Par ailleurs, dans l'approche simpliste présentée ci-dessus, tous les flux sont traités de la même façon, et il n'y a aucune gestion de la qualité de service.

Dans une approche plus réaliste, il faut évidemment prévoir des capacités de mémorisation en entrée et en sortie (en plus des tampons distribués dans la matrice elle même), et des politiques différenciées pour traiter les flux qui ont négocié des contrats de QoS différents.

- chaque port d'entrée gère un ensemble de files d'attente en entrée. Il y a une file d'attente par flux entrant sur ce port (i.e. par connexion). Pour chaque port d'entrée, un ordonnanceur local décide dans quelle file d'attente consommer une cellule, et à quel moment injecter la cellule dans la matrice : il n'est pas forcément optimal d'injecter le traffic maximal dans la matrice, car il faut éviter le phénomène d'engorgement.
- chaque port de sortie gère un ensemble de files d'attente en sortie. Il y a une file d'attente par flux sortant sur ce port (i.e. par connexion). Pour chaque port de sortie, un ordonnanceur local décide dans quelle file d'attente consommer la cellule à émettre. Pour ce qui concerne les ports de sortie, le but est d'assurer toujours le débit maximal, pour rentabiliser les liens physiques entre commutateurs.

Tout le problème de la qualité de service est donc concentré dans les stratégies de choix mis en œuvre par les ordonnanceurs en entrée et en sortie. C'est l'étude de ces stratégies qui fait l'objet de ce chapitre.

5.2 Modèle de l'architecture

5.2.1 Définition des flux à travers la matrice

Le rôle du contrôleur d'admission de connexions est d'accepter un certain nombre de flux à travers la matrice. Le mécanisme d'admission n'est pas discuté ici. Chaque flux F_{ijm} est identifié par un numéro de port d'entrée i, un numéro de port de sortie j, et un indice m permettant de différencier plusieurs flux qui ont la même source et la même destination. Selon les contrats de qualité de services, chaque flux est caractérisé par un débit minimal, une latence maximale et un taux de perte maximal. Si le nombre maximal de flux définis entre toute paire de ports entrée-sortie est M et si la matrice a N ports, il existe au plus N×N×M flux. Il faut remarquer la possibilité d'existence des flux rebouclés de type F_{iim}, qui entrent et sortent par le même port. Une des conditions imposée au trafic est que la somme des débits moyens des flux sortants par un port j ne doit pas dépasser le débit maximal du port j.

$$\forall \quad j, \quad \sum_{i,m} D_{ijm} \leq D_{\max}$$

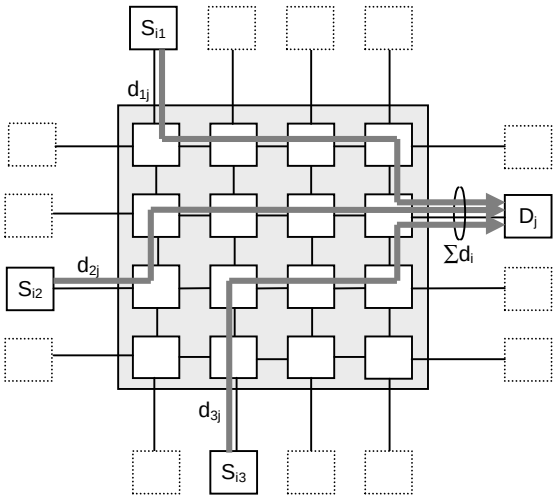

Figure 5.2 - Concentration de flux

5.2.2 Enoncé du problème

L'organisation fonctionnelle générique d'un commutateur ATM est reprise dans la figure 5.3. Dans cette analyse il a été faite l'hypothèse que la matrice de commutation ne fournit aucun service de gestion de priorités. Cela signifie que la garantie de la qualité de services reste entièrement à la charge des modules d'entrée et de sortie. Pour satisfaire les contrats de QoS qui ont été négociés pour chacun des flux, le commutateur doit - principalement - assurer un débit minimal D_{ijm} à chaque flux F_{ijm}.

On considère que cette fonction est entièrement assurée par les ordonnanceurs de sortie : en effet chaque ordonnanceur de sortie j dispose d'un certain nombre de files d'attentes (une file pour chacun des flux sortant par le port j), et décide de consommer un paquet dans une file d'attente plutôt que dans une autre en fonction des contrats de QoS qui ont été négociés pour chaque flux F_{ijm}. Ces contrats peuvent être très variables, le but n'est pas de proposer une politique particulière pour les ordonnanceurs de sortie. Il est considéré au contraire que la politique d'ordonnancement en sortie est une donnée du problème.

Le problème posé est alors le suivant : comment faire en sorte que la matrice de commutation ne perturbe pas la politique d'ordonnancement des ports de sortie? En d'autres termes,

comment faire en sorte que la file F_{ijm} du port de sortie j ne soit jamais vide lorsque l'ordonnanceur du port j décide de consommer une cellule dans cette file.

Il faut donc définir une politique d'ordonnancement du trafic en entrée ayant comme but d'alimenter en priorité les files d'attente de sortie qui ont tendance à se vider. Pour prendre leurs décisions, les ordonnanceurs d'entrée doivent donc utiliser des informations sur l'état des files d'attente des ports de sortie, et l'objectif des algorithmes qui gouvernent le fonctionnement des ordonanceurs d'entrée est de faire en sorte que les N×M files d'attente destination ne soient jamais vides.

En résumé, l'idée est de considérer que le contrôle de la qualité de service est entièrement réalisé par les politiques d'ordonnancement mises en œuvre par les ordonnanceurs de sortie. Ces politiques sont à priori quelconques, et inconnues des ordonnanceurs d'entrée. L'étude cherche donc à définir un algorithme d'ordonnancement en entrée en utilisant pour seule information l'état des files d'attente des ports de sortie.

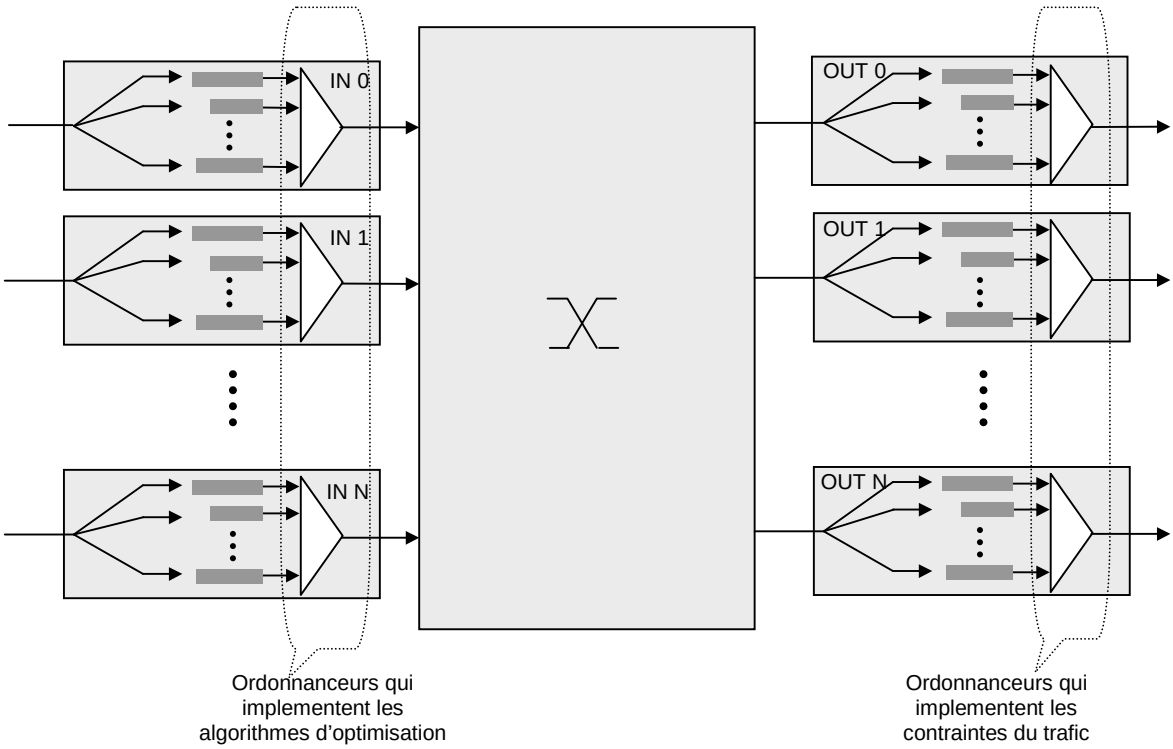

Figure 5.3 - Architecture de commutation extensible utilisant une matrice à commutation de paquets

5.2.3 Algorithme d'ordonnancement en entrée utilisant un système de notifications

Remarquons tout d'abord qu'il y a une correspondance biunivoque entre l'ensemble des files d'attente de tous les ports d'entrée, et l'ensemble des files d'attente de tous les ports de sortie, puisqu'il y a une file d'attente en entrée et une file d'attente en sortie pour chaque flux F_{ijm}. Chaque ordonnanceur d'un port de sortie j qui consomme un paquet dans une file d'attente destination F_{ijm} dispose d'un mécanisme de notification qui lui permet de signaler cette consommation à l'ordonnanceur du port d'entrée correspondant i. L'extraction d'un paquet d'une file d'attente destination doit être compensée au plus tôt par l'émission d'un paquet de la file d'attente source avec laquelle elle est associée d'une manière biunivoque.
Les notifications portent une information de priorité qui dépend du niveau de remplissage de la file d'attente destination. Le niveau de priorité P_i contenu dans une notification est évalué dynamiquement par le bloc de sortie suivant le taux de remplissage de la FIFO en sortie. La notification de priorité la plus élevée correspond à une tentative de consommation dans une FIFO vide ou presque vide.

i port source (destination de la notification)	**j** port destination (source de la notification)	**m** numéro sous-flux	**P_i** niveau de priorité

Figure 5.4 - Format de la notification

Il existe deux solutions pour acheminer les notifications aux ordonnanceurs des ports d'entrée.

Une première solution consiste à utiliser la même matrice pour l'acheminement des paquets de données et des paquets de notification. La bande passante nécessaire à l'acheminement des notifications est nettement inférieure à la capacité de la matrice de commutation puisque la longueur d'une notification est de 4 octets, alors que la longueur d'une cellule ATM avec en-tête est typiquement de 60 octets. La matrice doit être capable dans ce cas de traiter des paquets de taille variable, mais la charge provoquée par les transferts de notifications reste faible (environ 6%) devant celle des paquets de données utiles.

Une autre solution consiste à utiliser un système de commutation spécial pour l'acheminement des notifications. Le commutateur de notification à le même nombre de ports que le commutateur de données mais nécessite une bande passante nettement inférieure.

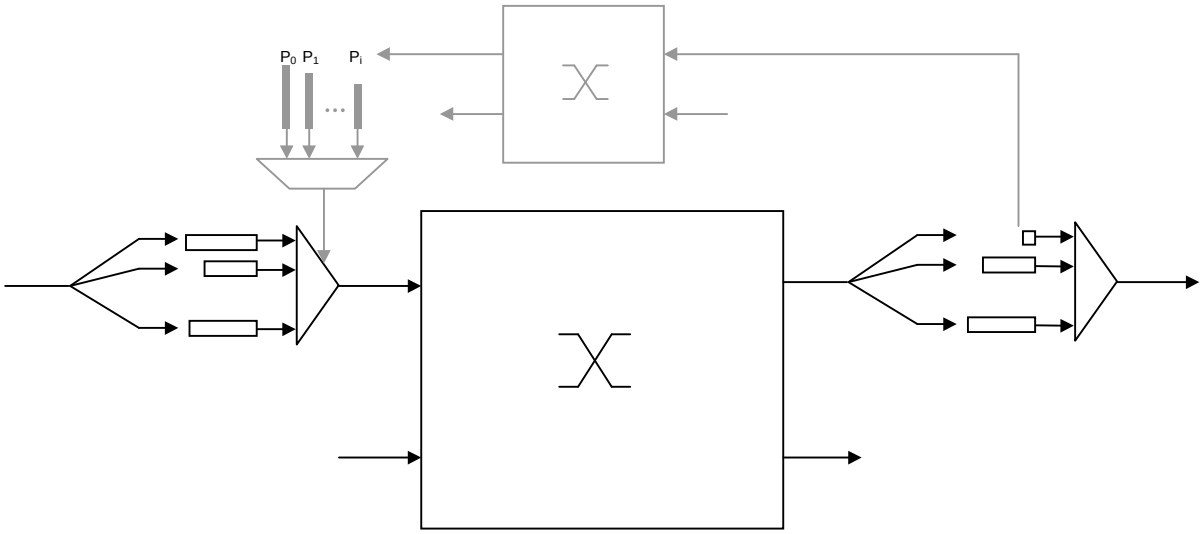

Figure 5.5 - Système d'acheminement de notifications

Une fois acheminée, la notification est déposée dans une file d'attente correspondante à son niveau de priorité P_i. L'ordonnanceur de sortie utilise une après l'autre les notifications situées en tête de la file d'attente de notification la plus prioritaire. L'information contenue dans la notification est utilisée par l'ordonnanceur pour choisir dans quelle file d'attente source consommer le prochain paquet.

Dans le cas où aucun paquet n'est présent dans la file d'attente de la source, la notification est détruite pour permettre de traiter les notifications suivantes.

Si une notification de priorité P_0 est reçue et la file d'attente indique en entrée par la priorité est vide, c'est une situation anormale, et le CAC est informé.

Dans les matrices multi-étages, le risque de contention est important. Les approches qui cherchent a améliorer le comportement face à la contention impliquent une coordination de l'émission de paquets. Cette coordination, nécessaire pour prendre en compte les échéances des autres modules dans le but d'éviter la contention, à besoin d'un mécanisme de communication entre les modules. Ce mécanisme dépend du nombre de ports et il n'est donc pas extensible. L'architecture qui fait l'objet de l'étude a comme objectif principal une valence élevée (débit et extensibilité) et pour ce raison n'est pas axée sur l'amélioration du comportement face à la contention.

5.2.3 Le format de paquet

Le format du paquet transmis au sein du système de commutation contient dans l'en-tête les champs suivants :

- indice du port source i ;
- indice du port de destination j ;
- numéro de sous-flux m ;

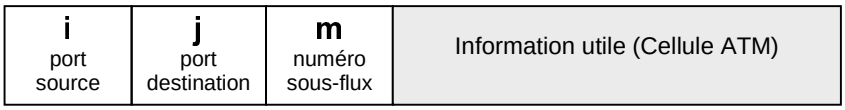

Figure 5.6 - Le format de paquet

Les indices i, j, m identifient complètement le flux F_{ijm}.

5.2.4 Limitations de l'architecture

Le système de notification décrit ici est efficace pour la commutation de paquets de taille fixe comme les cellules ATM. Pour les paquets de taille fixe, le calcul du taux de remplissage d'une file d'attente est simple, et une cellule consommée peut être directement compensée par une nouvelle cellule. Si les paquets ont une taille variable, le calcul du taux de remplissage est plus délicat, et il est plus difficile d'évaluer combien de paquets doivent être réémis pour un paquet consommé.

Un paramètre important de la QoS est le taux de perte de cellules. Dans notre architecture les endroits sensibles à la perte de cellules par saturation sont les files d'attente d'entrée et de sortie. Le paramètre qui permet d'ajuster ce taux est la taille des capacités de mémorisation.

5.3 Conclusion

Un point important de la logique de cette architecture le fait que le mécanisme de compensation à l'aide de notifications utilise comme seule information le taux de remplissage de la file d'attente de sortie correspondante. Les contrats de qualité de service restent entièrement à la charge des ordonnateurs de sortie. Le fait qui permet l'acquittement complet de ces contrats est qu'aucune des files d'attente de sortie soit jamais vide. Le bon fonctionnement de ces ordonnanceurs repose donc sur une information booléenne qui indique si dans la file sélectionnée il y a au moins une cellule. Un paramètre important de l'architecture est le nombre de priorités des notifications. Si les notifications ne sont pas départagées (une seule priorité) cella signifie que toute consommation d'une cellule est aussitôt remplacée par l'injection d'une autre. Le classement de notifications (en deux ou plusieurs priorités) signifie que les FIFO déficitaires sont alimentées en priorité.

Chapitre 6. Définition d'un simulateur générique pour un commutateur utilisant une matrice à commutation de paquets

Pour évaluer de façon quantitative les performances de l'algorithme proposé au chapitre 5, un simulateur générique de commutateur a été développé. Ce chapitre présente l'architecture logicielle et les modèles utilisés par ce simulateur destiné à évaluer au niveau système les performances d'une architecture utilisant une matrice de routage à commutation de paquets.

La généricité du simulateur se situe à plusieurs niveaux : le simulateur permet à l'utilisateur de définir les dimensions du réseau (nombre de ports, et taille des mémoires tampon), les caractéristiques du trafic (nombre de flux, débits et cadence de rafales). Enfin, l'utilisateur peut redéfinir les algorithmes d'ordonnancement et les fonctions d'instrumentation.

6.1 Modèle général du système

Le simulateur reprend l'architecture classique d'un commutateur avec un élément central de commutation entouré par des modules d'entrée et de sortie. Les modules d'entrée et de sortie ont une structure identique. Chaque module contient un ensemble de files d'attente et dispose d'une fonction d'ordonnancement. L'interface du simulateur permet de redéfinir facilement les algorithmes d'ordonnancement en entrée et en sortie de la matrice de commutation. Le routage dans la matrice est effectué selon une étiquette contenue dans l'en-tête du paquet.

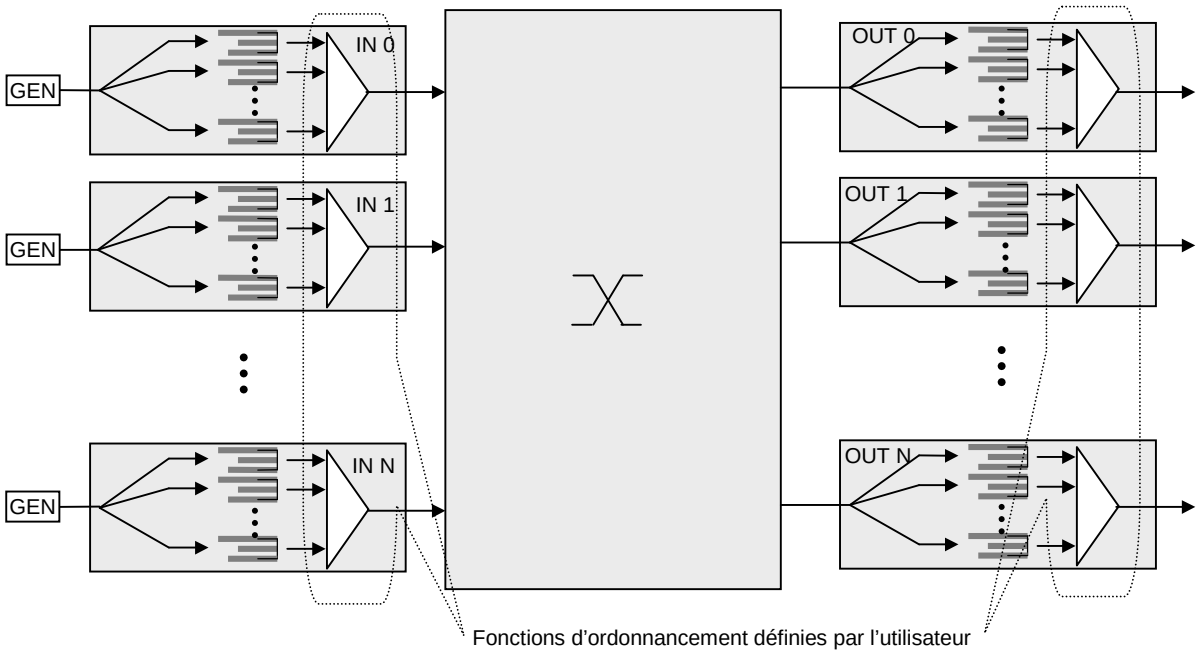

Figure 6.1 - Modèle général du système

A l'aide des deux paramètres N et M l'utilisateur définit la dimension du réseau et le nombre maximal de flux qui peuvent le traverser. N représente le nombre de ports. M représente le nombre maximal de flux qui peuvent être définis entre une paire de ports (i, j) . Il y a donc au plus N×N×M flux à travers le réseau et chaque ordonnanceur d'entrée ou de sortie gère au plus N×M flux. A l'aide d'une fonction d'initialisation, l'utilisateur a la possibilité de définir les flux qui existent (c'est à dire ceux qui ont été acceptés par le CAC) et les caractéristiques de chaque flux : le débit et l'intensité des rafales.

6.2 Modèles des éléments du système

6.2.1 Modèle de la matrice de routage

La matrice de routage modélise un réseau multi-étage "générique" de commutation de paquets. Le modèle ne cherche pas à décrire précisément la structure de ce réseau, ni le comportement fin de la matrice de routage. Le modèle considère que la matrice possède un mécanisme de contrôle de flux bas niveau garantissant une transmission sans pertes, et dispose d'une quantité limitée de mémoire distribuée sur les différents étages du réseau.

Le modèle de matrice de routage utilisé est celui d'un crossbar complet. Le modèle considère donc que la principale source de contention dans la matrice est la contention "en sortie", lorsque deux paquets provenant de deux entrées distinctes doivent emprunter la même sortie. Cette approximation n'est pas très éloignée de la réalité parce que ce sont principalement les contentions en sortie générées par un débit instantané supérieur à la capacité du port de sortie qui affectent la performance de la matrice de commutation. La capacité de mémorisation globale distribuée dans la matrice est modélisé par des FIFO. Il n'y a pas une FIFO par flux F_{ijm}, mais une FIFO par couple de ports (i,j), car chaque FIFO concentre les capacités de mémorisation distribuées au long des chemin possibles entre une entrée i et une sortie j. Le simulateur permet à l'utilisateur de définir la capacité de ces FIFO, pour représenter les capacités de mémorisation distribuées dans une matrice particulière.

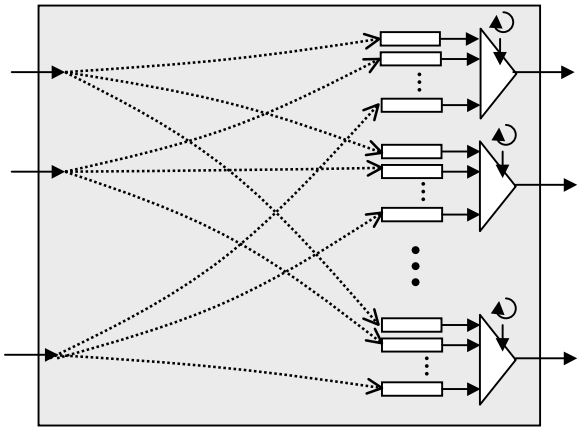

Figure 6.2 - Modélisation de la matrice de routage

Les ordonnanceurs de la matrice ne sont pas paramètrables. Ils fonctionnent selon un système de priorité tournante pour garantir l'absence de famine.

6.2.2 Modèle des files d'attente et des paquets

Les FIFO sont représentées par des vecteurs de pointeurs. Chaque pointeur non-nul pointe sur un descripteur de paquet qui contient les caractéristiques du paquet. Pour ce qui concerne les données transportées, le seul paramètre important est la longueur du paquet, qui est utilisée pour calculer les temps de transfert d'une FIFO à une autre. En plus de la longueur, chaque

paquet transporte l'indice du port source i, l'indice du port de destination j, l'indice de différenciation de flux m, nécessaires pour l'acheminement et la date d'émission nécessaire pour l'évaluation du temps de transit dans le système. La longueur du paquet est paramétrable. Le simulateur dispose d'un générateur de paquets. L'utilisateur peut demander la génération de paquets de longueur fixe ou peut demander la génération aléatoire de paquets de longueur variable, avec une distribution uniforme entre une longueur minimale et une longueur maximale. La date d'émission représente le nombre de cycles écoulés depuis le début de la simulation jusqu'à l'écriture du paquet dans la FIFO d'entrée.

6.2.3 Modèle de nœuds

Considérant l'ensemble constitué par les ports d'entrée, la matrice et les ports de sortie, est obtenue une structure régulière en trois couches dont chaque contient des files d'attente, une fonction d'arbitrage et une fonction de routage. Chaque ensemble (files d'attente / ordonnanceur / routeur) est regroupé dans une entité gouvernée par une machine à états. Le modèle d'un nœud est présenté dans la figure 6.3.

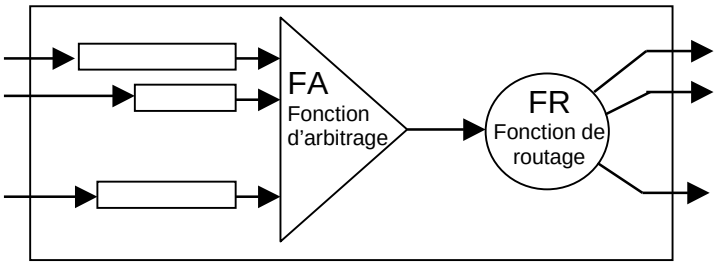

Figure 6.3 - Détail d'un nœud réseau

La première couche correspond aux modules d'entrée, la seconde couche aux modules matrice, et la troisième couche aux modules de sortie. La figure 6.4 présente une architecture à quatre ports (N=4), avec un seul flux définit par paire de ports (M=1). Dans chaque couche le routage est effectué d'une manière différente :

- nœuds d'entrée : routage selon l'indice j du port destination,
- nœuds matrice : routage selon l'indice i du port de source,
- nœuds de sortie : pas de routage.

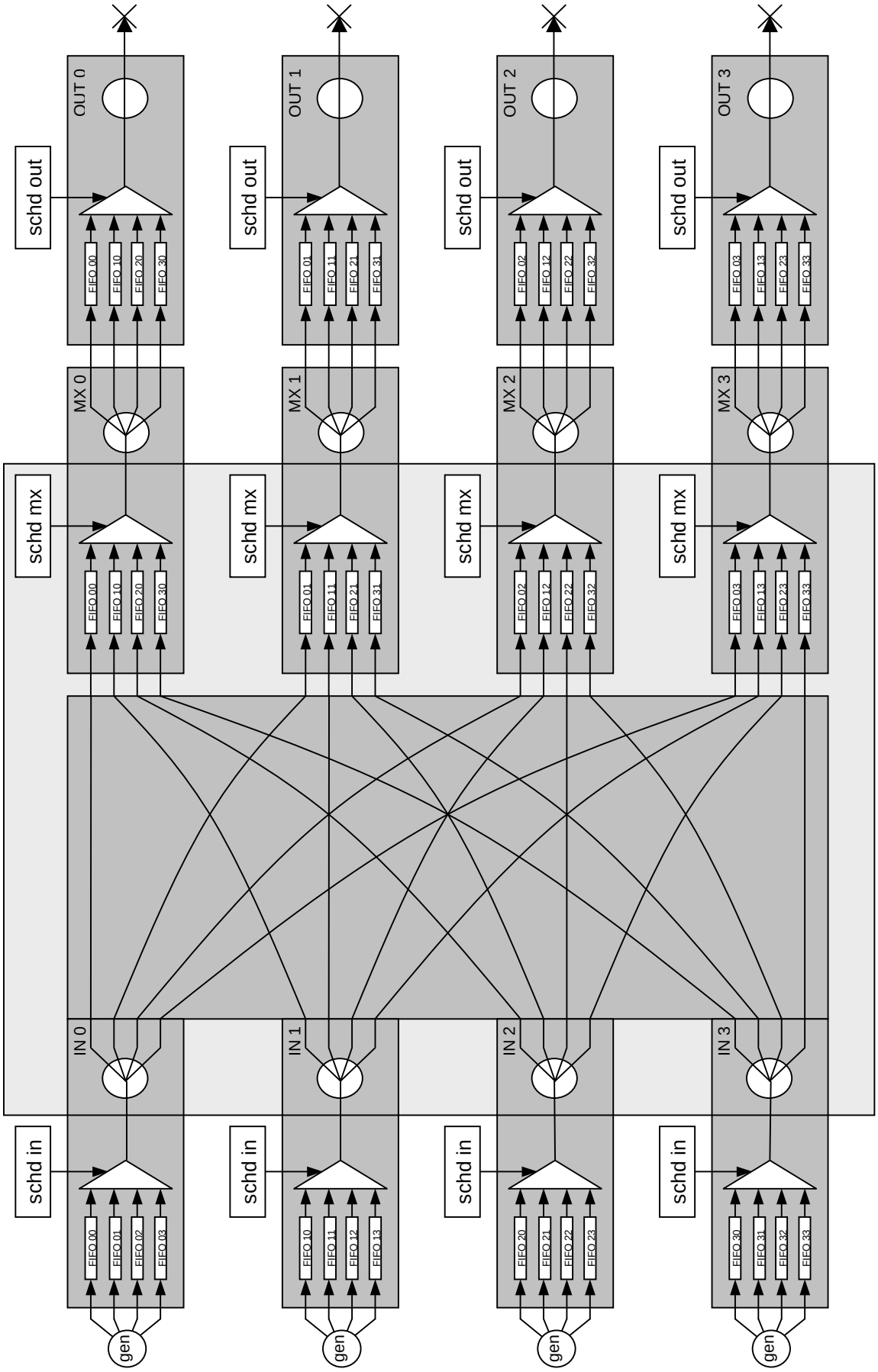

Figure 6.4 - Modèle de l'architecture pour un système à quatre ports (N=4), un flux par paire de ports (M=1)

6.2.4 Modélisation des transferts

Nous avons choisi la simulation événementielle pour des raisons de performances, et cela implique quelques précisions sur la modélisation des transferts d'un paquet entre deux FIFO.

Le principe de la simulation événementielle suppose que l'état de toutes les FIFO n'est pas explicitement décrit à tous les cycles. La simulation décrit les changements d'états des FIFO lorsqu'il se produit un "événement".

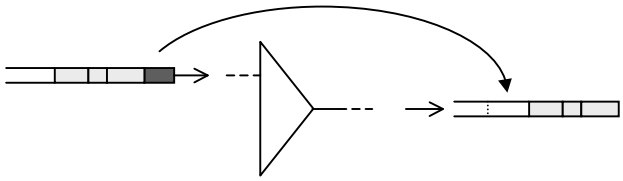

Figure 6.5 - Modélisation événementielle du transfert d'un paquet

Les transferts entre FIFO sont de type « store & forward », c'est à dire que le récepteur d'un paquet n'accorde la permission d'écriture que si le récepteur dispose d'un espace mémoire supérieur ou égal à la longueur du paquet. La permission d'écriture dans la FIFO destination est donc accordée au niveau paquet.

Pour gérer la progression des paquets dans le commutateur tout en respectant le contrôle de flux, un ordonnanceur peut être dans un des trois états suivants :

- IDLE : toutes les FIFO source gérées par l'ordonnanceur sont vides.
- BUSY : l'ordonnanceur a sélectionné une FIFO source, et il est occupé à transférer un paquet de la FIFO source vers la FIFO destination.
- WAIT : il y a au moins une FIFO source non vide qui a été sélectionnée par l'ordonnanceur, mais il n'y a pas assez de place dans la FIFO destination.

Le simulateur événementiel réalise tout transfert à l'aide de deux événements : un événement d'extraction (POP) de la FIFO source, un événement d'insertion (PUSH) dans la FIFO destination. La position temporelle des événements est calculée en fonction de la longueur du

paquet. Pendant toute la durée d'un transfert, la machine d'état décrivant l'ordonnanceur de la FIFO source se trouve soit dans un état BUSY si le transfert est en cours soit dans un état WAIT si un transfert est bloqué par manque de place.

Dans le cas des transferts "store & forward", le transfert a une durée connue qui ne dépend que de la longueur du paquet, et pendant le transfert, la machine à états se trouve dans l'état BUSY. Les états WAIT issus du fait qu'il n'y pas suffisamment de place dans les FIFO destination apparaissent avant le transfert. Le transfert commence dès qu'une place égale à la longueur du paquet est libérée dans la FIFO destination ce qui se traduit par un enchaînement d'un état WAIT et d'un état BUSY. Dans la figure 6.5, la durée de l'état WAIT est t_w, la durée de l'état BUSY représente le temps effectif de transfert t_{tr} et la durée totale du transfert est $t_w + t_{tr}$.

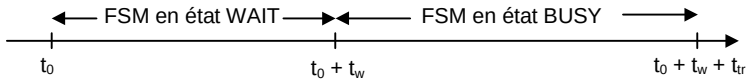

Figure 6.6 - Transfert « store & forward »

Dans l'intervalle ($t_0 + t_w$, $t_0 + t_w + t_{tr}$), il existe une non-concordance entre la place réelle dans les FIFO et celle indiquée par le vecteur d'état des FIFO. Cependant, pour garantir un comportement « store & forward », l'état des deux FIFO source et destination ne doit être mis à jour qu'à la fin du transfert, c'est à dire $t_0 + t_w + t_{tr.}$

Le débit maximal offert par une matrice de commutation représente la somme des débits de chacune des entrées de la matrice. Si les débits maximaux des ports sont égaux, le débit total est le produit du nombre de ports par le débit maximum d'un port.

$$D_{\max - matrice} = N \times D_{\max - port}$$

En réalité, le débit effectif de la matrice est toujours inférieur au débit maximal à cause des contentions internes dans la matrice. Une des possibilités pour augmenter le débit effectif est d'augmenter le débit maximal de la matrice. Ce résultat peut être obtenu en augmentant la fréquence de fonctionnement de la matrice par rapport à celle des ports d'entrée/sortie. Le simulateur permet de définir la valeur du paramètre mx_ck. L'accélération de la matrice est obtenue en variant la durée de l'état BUSY à l'aide de ce paramètre : la durée du transfert est $t_w \times$ mx_ck.

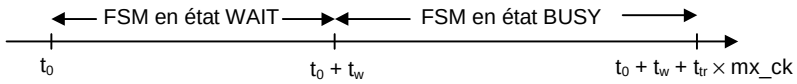

Figure 6.7 – Accélération de transferts

6.3 Fonctionnement du simulateur

6.3.1 L'architecture logicielle

L'architecture logicielle est modulaire. Un fichier de configuration qui peut être facilement redéfini par l'utilisateur sert à paramétrer le simulateur.

Le but du simulateur est de tester différents algorithmes d'ordonnancement. Par ailleurs, l'utilisateur du simulateur doit pouvoir redéfinir facilement les différents paramètres qu'il souhaite mesurer au cours d'une session de simulation. Pour pouvoir les modifier facilement, les fonctions d'ordonnancement et les fonctions d'instrumentation (qui permettent d'analyser les résultats) sont regroupées dans un module séparé et peuvent être compilées séparément.

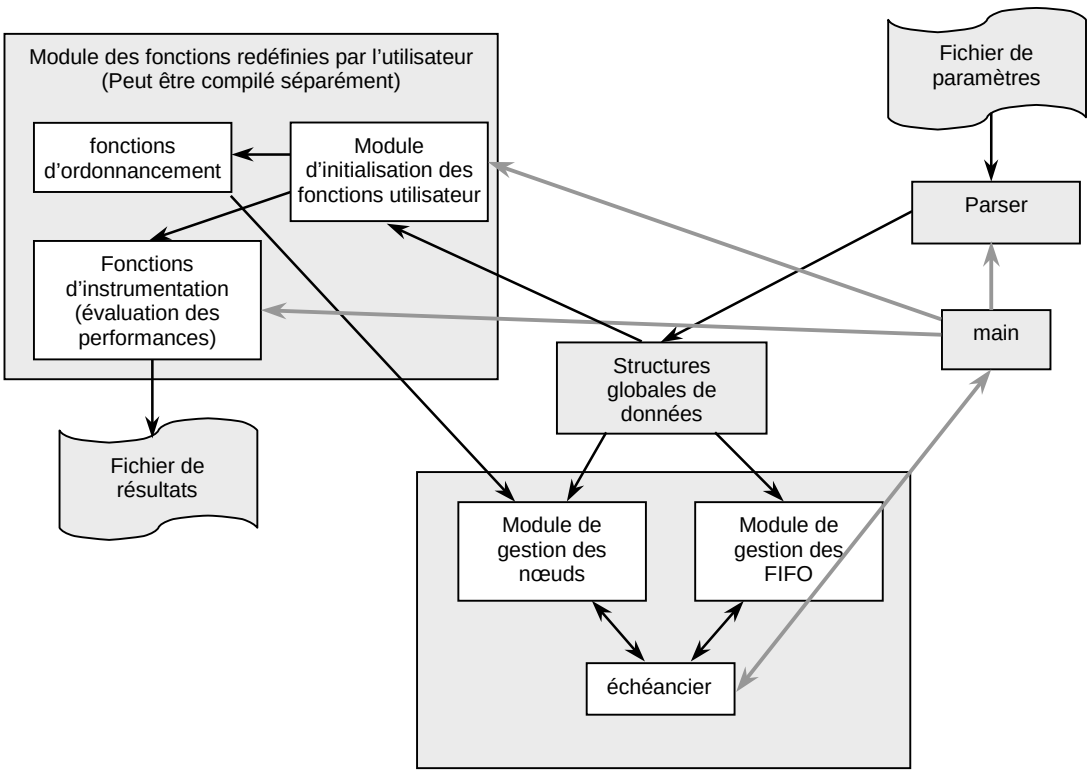

Figure 6.8 - Schéma du simulateur

L'interface entre les fonctions utilisateur et le moteur du simulateur est facilitée par un fichier qui définit les structures globales de données. Le fichier, nommé datasrtuct.h est présenté dans l'annexe A.3.5. Les structures globales de données sont créées par les fonctions d'initialisation et à l'aide des paramètres extraits du fichier de configuration.

Parmi les structures globales des données, la plus importante est celle de files d'attente. Pour être accédées facilement par les fonctions d'ordonnancement, les FIFO sont organisées dans trois tableaux: un tableau tridimensionnel (N×N×M) pour les FIFO d'entrée, un tableau bidimensionnel (N×N) pour les FIFO de la matrice et un tableau tridimensionnel (N×N×M) pour les FIFO de sortie.
Une autre structure globale est le tableau des contraintes de trafic. Pour chaque flux sont précisés le débit moyen et la longueur moyenne d'une rafale qui caractérise l'agressivité de chaque flux.

6.3.2 Le déroulement de la simulation

La simulation comporte trois phases : l'initialisation, l'exécution et la génération de résultats. L'initialisation est faite à partir des données contenues dans le fichier de paramètres nommé simfi.cfg. La structure du fichier est présentée en annexe A.3.2. Sont initialisées les structures de données et les variables globales simulateur. Quelques éléments sont fixés pendant la phase d'initialisation:

- la duré de la simulation;
- la dimension de la matrice N;
- l'indice M de différenciation de flux;
- les tailles maximales des FIFO de la matrice et des FIFO sortie, seule la taille des FIFO d'entrée n'est pas bornée.

L'information booléenne qui indique l'existence d'un flux est contenue dans un tableau tridimensionnel N×N×M. Le tableau, nommé $flux_{ijm}$ peut être remplit aléatoirement en respectant un pourcentage de flux vides donné dans le fichier de paramètres. Un autre tableau tridimensionnel D_{ijm} définit les débits moyens garantis (c'est à dire les contrats de QoS) de

chaque flux. Le tableau peut être généré automatiquement en définissant un débit maximal et un débit minimal pour chaque flux. Ce deux paramètres se trouvent aussi dans le fichier de configuration. Pendant la création du tableau, est vérifiée l'inégalité mathématique qui assure un comportement sans blocage à long terme en sortie de la matrice de routage:

$$\forall \quad j, \quad \sum_{i,m} \frac{D_{ijm}}{D_{\max\text{-}port}} \leq 1 \quad (D_{\max\text{-}port} = 1)$$

En plus du débit moyen D_{ijm}, un second paramètre N_{ijm} permet de caractériser chaque flux, en définissant le caractère plus ou moins régulier de ce débit, c'est à dire l'intensité des rafales. Les éléments du tableau N_{ijm} sont des entiers qui représentent le nombre moyen de paquets contenus dans une rafale. Ici encore, il est possible de générer automatiquement le contenu de ce tableau, à l'aide d'un paramètre du fichier de configuration qui indique le nombre maximal de paquets d'une rafale. Pour faciliter l'interprétation des résultats sont fournis en sortie des fichiers contenant les éléments de ces trois tableaux qui décrivent le trafic.

L'enchaînement des phases de la simulation est contrôlé par le module principal (main). Si les fonctions d'ordonnancement peuvent être définies par l'utilisateur, les fonctions de routage font partie de la structure du simulateur et elles ne sont pas paramétrables. En entrée de la matrice le routage est fait selon l'indice du port de destination, en sortie selon l'indice de la file destination. Les fonctions d'ordonnancement sont appelées seulement si au moins une des FIFO n'est pas vide et elles doivent retourner chaque fois une FIFO non-vide. Si les fonctions d'ordonnancement fournies par l'utilisateur ne respectent pas ce principe, la simulation est arrêtée et un message d'erreur est généré.

Au moment où le nombre de cycles spécifié par le fichier de configuration est atteint, la simulation est arrêtée et des fichiers de résultats sont générés selon les procédures écrites dans les fonctions d'instrumentation accessibles à l'utilisateur. Les fonctions d'instrumentation ont accès à une série de données qui sont enregistrées pendant la simulation. Outre les débits moyens effectivement réalisés pour chaque flux, il est possible de mesurer les valeurs moyennes et maximales des temps de transit, le nombre et la durée de contention dans la matrice pour chaque flux. Enfin, il est possible de compter le nombre d'événements de type essai de lecture d'une FIFO vide effectué par les ordonnanceurs de sortie dans le but d'évaluer les performances des algorithmes d'ordonnancement.

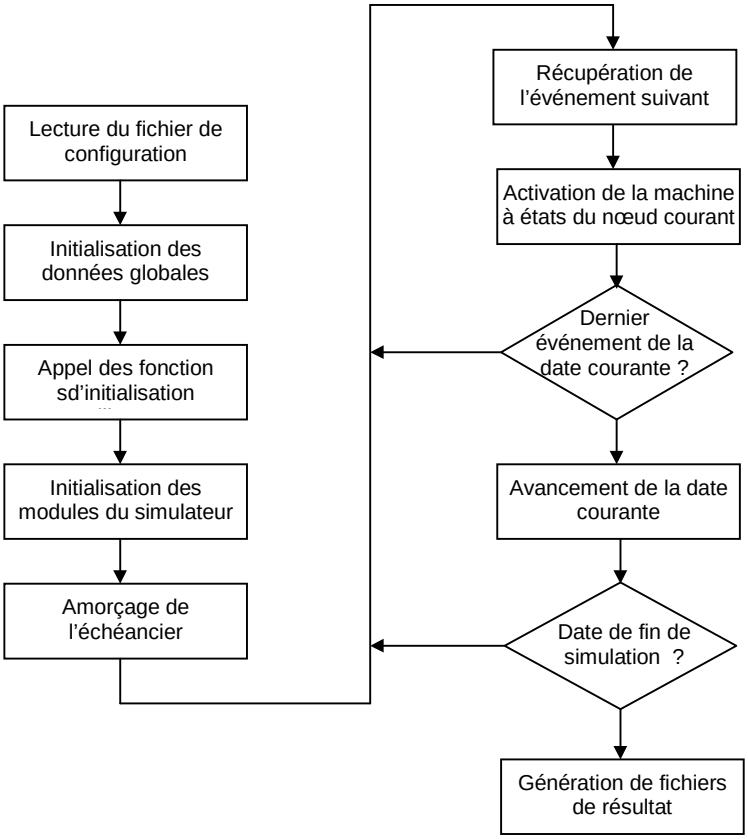

Figure 6.9 - Diagramme d'activité du simulateur

Chapitre 7. Evaluation de la performance d'une architecture de commutateur extensible utilisant une matrice à commutation de paquets

Le chapitre présente les résultats des simulations effectuées dans le but d'évaluer l'architecture de commutateur décrite dans le chapitre 5. En première partie sont détaillés les objectifs de la simulation, la configuration du simulateur et la méthode d'implémentation des contraintes de trafic. La deuxième partie présente et analyse les résultats expérimentaux.

7.1 Objectifs de la simulation

L'objectif est de vérifier si l'architecture à base de notifications proposée dans le chapitre 5 permet de garantir une satisfaction de contrats de QoS à 100% et de déterminer de façon quantitative les conditions qui permettent d'obtenir cette garantie. La méthode expérimentale utilisée pour évaluer la performance du commutateur consiste à définir différents indicateurs et à mesurer leurs valeurs lorsque le commutateur est soumis à différentes charges de travail.

Le premier indicateur consiste à cumuler les violations des contrats de débit pour chacun des flux, et de les rapporter au contrat global de débit. L'insuffisance de débit pour un flux représente la différence entre le contrat de débit D_{ijm} et le débit moyen $D_{ijm\text{-}réel}$ effectivement réalisé pour le flux. Le débit total contractuel est défini par :

$$D_{contrat} = \sum_{i,j,m} D_{ijm-contrat}$$

Le cumul des insuffisances de débit des flux déficitaires est défini par :

$$D_{insuff.} = \sum_{i,j,m} (D_{ijm-contrat} - D_{ijm-réel})\Big|_{D_{ijm-contrat} > D_{ijm-réél}}$$

Le taux de réalisation des contrats de débit est donc :

$$QoS = 1 - \frac{D_{insuff.}}{D_{conttrat}}$$

Par ailleurs, le chapitre 5 conclut qu'une information sur la présence ou l'absence de paquets dans les files d'attente des modules de sortie sélectionnées donne aussi une information utile sur la performance en termes de qualité de services de l'architecture. Par conséquent, un second indicateur de performance est le pourcentage de sélections d'une FIFO vide effectuées par les ordonnanceurs de sortie (par rapport au nombre total de sélections).

Un troisième indicateur permet d'évaluer le degré de contention de la matrice de commutation. Le blocage des ports d'entrée de la matrice est causé par le remplissage des FIFO qui modélisent les capacités de mémorisation de la matrice. Le pourcentage de cycles passés dans l'état WAIT pour les ordonnanceurs des ports d'entrée est un bon indicateur de la contention de la matrice. La mesure représente une moyenne par port de sortie.

Les simulations ont pour objectif principal d'évaluer dans quelle mesure l'architecture est capable de desservir les contrats de QoS implémentés par les ordonnanceurs de sortie. Néanmoins, la latence moyenne et la latence maximale sont deux informations intéressantes. La latence moyenne représente le nombre moyen de cycles nécessaires à un paquet pour traverser le système à partir de son injection par le nœud d'entrée jusqu'à son extraction de la FIFO de sortie.

7.2 Modélisation du trafic

L'expérimentation cherche à évaluer les perturbations apportées par la matrice de commutation sur les contrats mis en œuvre par les ordonnanceurs de sortie. Les caractéristiques du trafic en entrée ne sont pas modélisées, les FIFO d'entrée sont considérées comme toujours alimentées. Ce choix est motivé par le fait que les perturbations sur le trafic entrant produisent une dégradation de la qualité de service de bout en bout (*end to end QoS)*. Cette dégradation n'est pas causée par le commutateur en cause mais par des éléments de réseau qui se trouvent en amont.

Les caractéristiques du trafic en sortie sont définies à l'aide des paramètres du fichier de configuration. Il est possible de définir le nombre de flux, leur débits moyens souhaités, et l'intensité des rafales de chaque flux, grâce à trois tableaux de dimensions N×N×M:

- un avec des valeurs booléennes définissant l'existence du flux F_{ijm} ;
- un définissant les débits moyens D_{ijm} pour chaque flux;
- un contenant les longueurs moyennes N_{ijm} des rafales pour chaque flux.

Les éléments du tableau des débits D_{ijm} respectent la condition :

$$\forall j, \quad \sum_{i,m} \frac{D_{ijm}}{D_{\max - port}} \leq 1$$

Pour simplifier, les débits sont normalisés, c'est à dire que le débit maximal d'un port est le débit unité : $D_{port} = 1$. Les débits D_{ijm} sont donc compris entre 0 et 1.

Le paramètre « Charge » du fichier de configuration représente la somme des débits de tous les flux ($\sum D_{ijm}$) rapportée à la capacité maximale de commutation (N).

$$Charge = \frac{\sum_{i,j,m} D_{ijm}}{N}$$

Le paramètre « Charge » doit être considéré comme un objectif, puisque les débits D_{ijm} sont eux-mêmes des objectifs à atteindre.

Conformément aux principes architecturaux énoncés dans le paragraphe 5.2.2 le trafic en sortie est entièrement défini par les ordonnanceurs de sortie. La fonction de sélection d'une FIFO, qui définit le comportement des ordonnanceurs de sortie, accepte comme argument l'indice du port de sortie associé j, et retourne les indices i et m de la file d'attente d'où la prochain paquet sera extrait. Le sélection de la file est effectuée en fonction des contrats de débit D_{ijm} et des caractéristiques de rafales N_{ijm} de chacun des flux. Le paramètre N_{ijm}, qui permet de régler la longueur d'une rafale, caractérise un trafic irrégulier en sortie. Si la fonction d'ordonnacement sélectionne une FIFO correspondant à un flux dont le paramètre N_{ijm} est supérieur à 1, il y aura N_{ijm} sélections consécutives de la même FIFO. Si ce nombre de sélections est atteint ou si la FIFO devient vide, une nouvelle rafale est initialisée pour un autre flux. Chaque fois qu'un paquet est transmis, les débits réels réalisés sont évalués:

$$D_{ijm-réel} = \frac{Nombre\ de\ mots\ transmis\ pour\ le\ flux\ F_{ijm}}{Date\ courante\ de\ simulation}$$

Les débits réels sont comparés aux débits D_{ijm} souhaités, et le flux qui est le plus déficitaire au moment de l'appel de la fonction de sélection est choisi. Si la FIFO choisie est vide, le compteur d'échecs est incrémenté et la sélection se poursuit jusqu'au moment où une file non vide est trouvée.

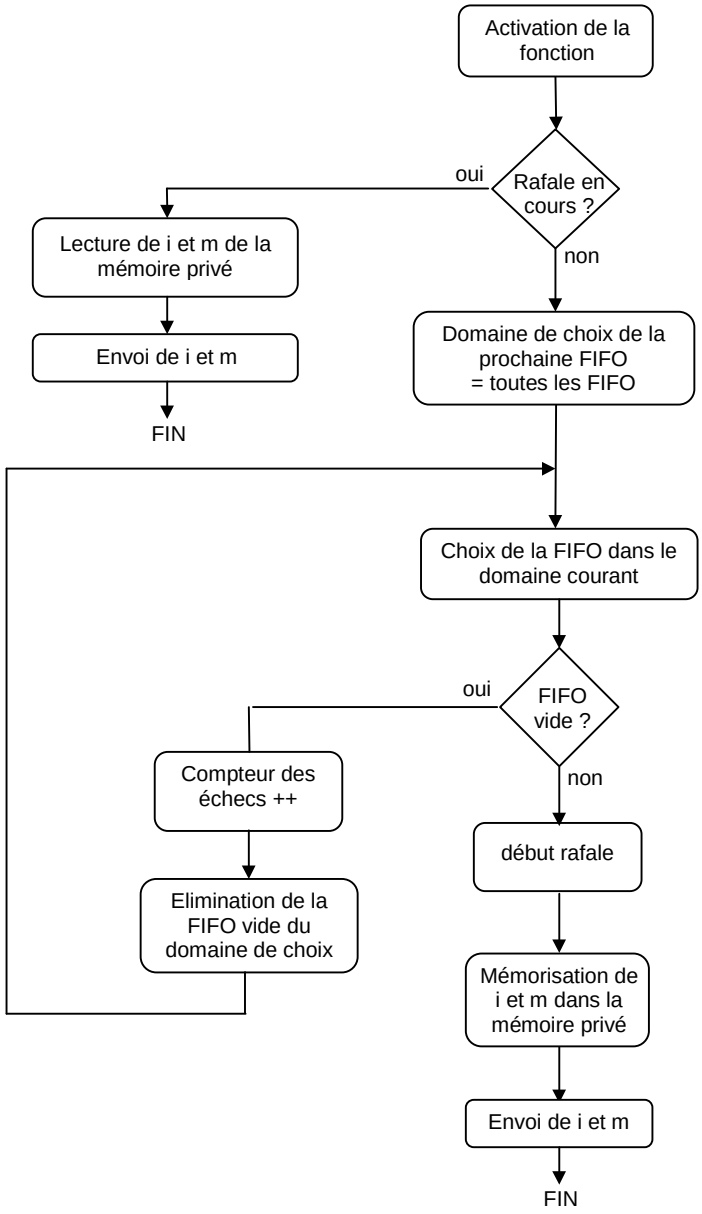

Figure 7.1 - Algorithme de la fonction d'ordonnancement en sortie

7.3 Mesures de performances

7.3.1 Le coefficient d'accélération de la matrice

Comme expliqué dans le chapitre 5, le débit maximal offert par la matrice de commutation est en principe égal à la somme des débits de chacune des entrées de la matrice. Mais le débit effectif de la matrice est toujours inférieur au débit maximal (à cause des phénomènes de contention dans la matrice). Il y a deux possibilités pour faire face à cette difficulté :

- La première solution est de diminuer la charge de la matrice (c'est à dire la somme des débits acceptés Dijm), mais cette solution aboutit à une sous-utilisation systématique des liens physiques en entrée et en sortie du commutateur, ce qui n'est pas souhaitable.
- Pour éviter cette sous-utilisation du réseau, il faut que le débit maximal de la matrice soit supérieur à la somme des débits des ports d'entrée ou de sortie. En pratique, ce résultat peut être obtenu en faisant fonctionner la matrice à une fréquence supérieure à celle des ports d'entrée/sortie. Ce coefficient d'accélération de la matrice (le *speed-up*) est un paramètre important pour atteindre l'objectif de 100% de satisfaction des contrats de QoS, mais c'est un facteur de coût pour le commutateur, et il est donc souhaitable que ce facteur soit aussi proche que possible de l'unité.

Les conditions de simulation sont précisées dans le fichiers de paramètre de configuration (simfi.cfg) ci-dessous :

- dimension de la matrice (Nombre de ports) N = 8 ;
- nombre maximal de sous-flux par paire de ports entrée-sortie M = 100 ;
- durée de la simulation (en nombre de cycles) EOS = 5 000 000;
- taille maximale des FIFO de la matrice (en nombre de mots) dms_max_mx= 64 ;
- indice de multiplication de la fréquence des ports d'entrée in_ck = 1 ;
- coefficient de multiplication de la fréquence de la matrice mx_ck = paramètre variable ;
- indice de multiplication de la fréquence des ports d'entrée out_ck = 1 ;
- taille maximale des FIFO des ports de sortie (en nombre de mots) dms_max _out= 2048 ;
- taille des paquets (en nombre de mots) pkt_size = 60;
- pourcentage de flux non initialisés flux0_prc = 10 ;
- rapport entre le débit maximal et minimal d'un flux Dmaxmin = 8;
- longueur maximale d'une rafale max_brst_leng = 8;
- charge global de la matrice Charge = paramètre variable ;
- Nombre de priorités utilisées par l'algorithme prio_range = 4;

Dans les simulations ci-dessous, deux paramètres varient: mx_ck et Charge. Les tableaux suivants donnent les résultats expérimentaux obtenus en faisant varier le coefficient

d'accélération de la matrice de 5% d'une mesure à l'autre. La série d'expérimentations est faite pour 5 valeurs du paramètre Charge : 90%, 80%, 70%, 60% et 50%.

Signification des colonnes des tableaux:

- « Charge effective » appliquée sur la matrice représente ($\sum D_{ijm\text{-}réel}$) / N . L'évaluation est faite à la fin de la simulation;
- « Accélération de la matrice» représente le paramètre mx_ck ;
- « QoS » représente le taux de réalisation de contrats de trafic le paramètre défini dans le paragraphe 7.1 ;
- « Taux de réalisation du contrat du flux le plus déficitaire » est le rapport $D_{ijm\text{-}réel}/D_{ijm}$ du flux pour lequel la différence $D_{ijm} - D_{ijm\text{-}réel}$ a la valeur maximale ;
- « Taux de sélection de FIFO vides » représente le rapport entre le nombre de sélection de FIFO vides et le nombre total d'appels de la fonction d'ordonnancement de sortie ;
- « Taux de contentions dans la matrice » représente la durée cumulée des états WAIT des ordonnanceurs d'entrée rapportée à la durée de la simulation.

Résultats expérimentaux pour une charge à 90% de la capacité maximale de commutation:

Charge effective	Accélé-ration de la matrice	QoS	Taux de réalisation du contrat du flux le plus déficitaire	Taux de sélections de FIFO vides	Taux de contentions dans la matrice	Latence moyenne	Latence maximale
91,0471%	1	89,2822%	92,4536%	32,2040%	7,5659%	154728	835786
93,3030%	1,05	91,6043%	89,3562%	29,0940%	8,3950%	163000	813710
96,3506%	1,1	94,9191%	85,4879%	24,4570%	7,4456%	193510	873406
96,3483%	1,15	97,0120%	94,6846%	20,5332%	7,2571%	211493	859625
96,3609%	1,2	97,3702%	92,4562%	18,5071%	6,8978%	220062	898966
95,4906%	1,25	98,3263%	95,5316%	17,4001%	7,0426%	225473	889488
95,4714%	1,3	99,1325%	92,8436%	15,8430%	6,6767%	233666	919767
96,3584%	1,35	99,4083%	89,4062%	14,4369%	6,8424%	229489	899882
96,3488%	1,4	99,8147%	99,9430%	4,1779%	6,2378%	250820	909982
96,3532%	1,45	99,4818%	96,7982%	1,4631%	5,9383%	256619	890086
96,3588%	1,5	99,7625%	97,9630%	2,7540%	5,9756%	255599	929710
95,4927%	1,55	99,6616%	97,9030%	0,9026%	5,7897%	264431	948726
95,4776%	1,6	99,6732%	97,3390%	0,9707%	5,5879%	268318	929958
96,3530%	1,65	99,8522%	96,7408%	0,9908%	5,5931%	266018	879986
95,5049%	1,7	99,7643%	97,9350%	0,8518%	4,8259%	276376	939831
95,4925%	1,75	99,7167%	99,1430%	0,7028%	4,7830%	282076	960010
96,3429%	1,8	100%	100%	0,6794%	4,1960%	287120	930080

Tableau 7.1 - Résultats expérimentaux pour un taux de charge de 90%

Résultats pour une charge à 80% de la capacité maximale de commutation:

Charge effective	Accélération de la matrice	QoS	Taux de réalisation du contrat du flux le plus déficitaire	Taux de sélections de FIFO vides	Taux de contentions	Latence moyenne	Latence maximale
90,6716%	1	95,5425%	94,4436%	26,9773%	7,6187%	176322	940643
91,3573%	1,05	97,4663%	98,3972%	22,2329%	7,7166%	204645	996905
92,1490%	1,1	98,2977%	95,8479%	21,2827%	7,3823%	209952	998582
91,3642%	1,15	99,1080%	96,9462%	16,0743%	7,4773%	224619	998291
91,3805%	1,2	99,8274%	97,6782%	12,5438%	6,9974%	237966	996964
91,3797%	1,25	99,8477%	99,8735%	12,0443%	7,0477%	232773	999610
91,3613%	1,3	99,9988%	97,9318%	4,2421%	6,7177%	247466	1000053
91,3753%	1,35	99,9801%	98,9192%	6,3469%	6,5923%	249786	996082
91,3856%	1,4	100%	100%	1,0700%	6,2562%	262466	996538
91,3739%	1,45	99,9988%	98,4867%	0,8273%	5,8371%	269596	1000017
91,3656%	1,5	100%	100%	0,8165%	5,7350%	270521	999796
91,3542%	1,55	100%	100%	0,7810%	5,2725%	281702	1000040
91,3561%	1,6	100%	100%	0,7838%	5,2398%	280988	1000033
91,3988%	1,65	100%	100%	0,7420%	5,0727%	285947	999927

Tableau 7.2 - Résultats expérimentaux pour un taux de charge de 80%

Résultats pour une charge à 70% de la capacité maximale de commutation:

Charge effective	Accélération de la matrice	QoS	Taux de réalisation du contrat du flux le plus déficitaire	Taux de sélections de FIFO vides	Taux de contentions dans la matrice	Latence moyenne	Latence maximale
86,8923%	1	99,6130%	97,3632%	19,2415%	7,6200%	218836	1000036
87,5993%	1,05	99,7981%	94,3792%	15,5363%	7,6972%	226047	999077
87,5777%	1,1	99,9839%	98,7992%	11,7002%	7,5206%	238012	993840
88,3208%	1,15	99,9899%	97,9643%	9,7438%	7,4386%	242347	999411
87,5749%	1,2	99,9987%	93,9873%	8,4500%	7,2601%	247951	999742
87,5652%	1,25	100%	100%	6,8939%	7,3191%	249001	989254
87,6045%	1,3	100%	100%	1,1759%	6,7544%	262126	1000007
87,5549%	1,35	100%	100%	0,9193%	6,8393%	265963	983592

Tableau 7.3 - Résultats expérimentaux pour un taux de charge de 70%

Les résultats pour une charge à 60% de la capacité maximale de commutation:

Charge effective	Accélération de la matrice	QoS	Taux de réalisation du contrat du flux le plus déficitaire	Taux de sélections de FIFO vides	Taux de contentions dans la matrice	Latence moyenne	Latence maximale
83,4540%	1	99,9622%	95,3392%	16,7605%	7,8533%	241970	999237
83,4776%	1,05	99,9975%	97,7529%	11,8163%	7,9559%	230119	989753

84,1047%	1,1	100%	100%	6,4891%	7,4586%	255731	999936
83,4817%	1,15	100%	100%	1,0047%	6,9229%	276179	992686
84,0905%	1,20	100%	100%	0,9045%	6,8642%	272039	999607

Tableau 7.4 - Résultats expérimentaux pour un taux de charge de 60%

Les résultats pour une charge à 50% de la capacité maximale de commutation:

Charge effective	Accélération de la matrice	QoS	Taux de réalisation du contrat du flux le plus déficitaire	Taux de sélections de FIFO vides	Taux de contentions dans la matrice	Latence moyenne	Latence maximale
80,9257%	1	100%	100%	8,4042%	7,7134%	249439	995723
80,2607%	1,05	100%	100%	6,6472%	7,9909%	249352	993227
80,8778%	1,1	100%	100%	3,0971%	7,7377%	265695	998246
80,3080%	1,15	100%	100%	1,4304%	7,5041%	267434	954292

Tableau 7.5 - Résultats expérimentaux pour un taux de charge de 50%

Les données expérimentales sont rassemblés dans le graphique d'évaluation de l'accélération de la matrice nécessaire pour atteindre un niveau de qualité de service de 100%:

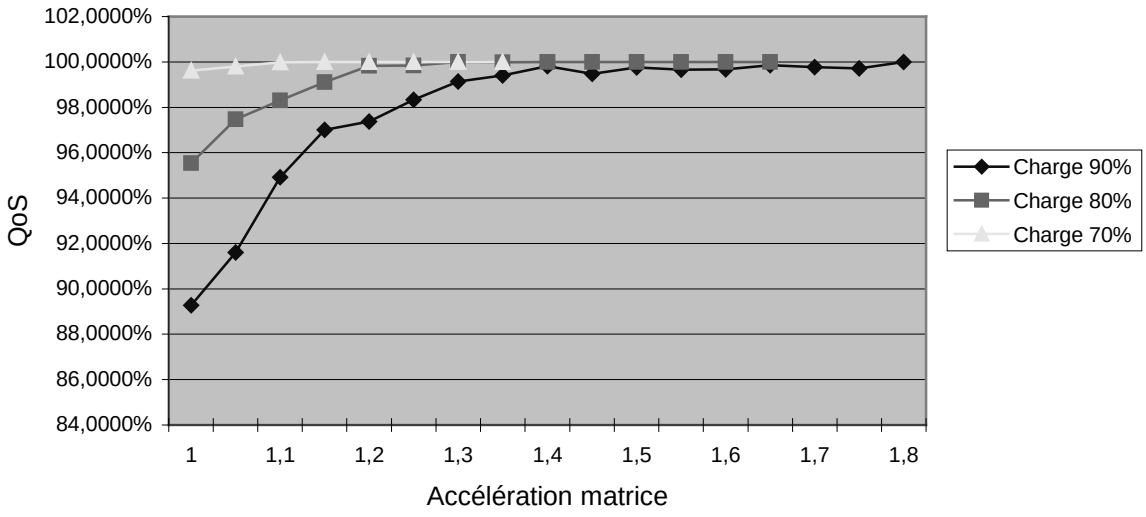

Figure 7.2 - Evaluation de l'accélération de la matrice

Pour une charge à 90%, il faut une accélération de 1.8 pour obtenir un taux de satisfaction de 100%. Pour une charge de 80%, le taux de satisfaction de 100% est atteint à partir d'une accélération de 1,5. Pour une charge de 70% cette valeur baisse à 1,25 et pour une charge de

60% à 1,1. Pour des charges inférieures à 50% la matrice n'a pas besoin de fonctionner à une fréquence supérieure à la fréquence des modules d'entrée et de sortie.

Le graphique suivant vise à évaluer la charge maximale supportée par le réseau à une accélération donnée. Les courbes sont tracées pour des accélérations différentes et permettent de voir à partir de quelle charge la QoS souffre de dégradations.

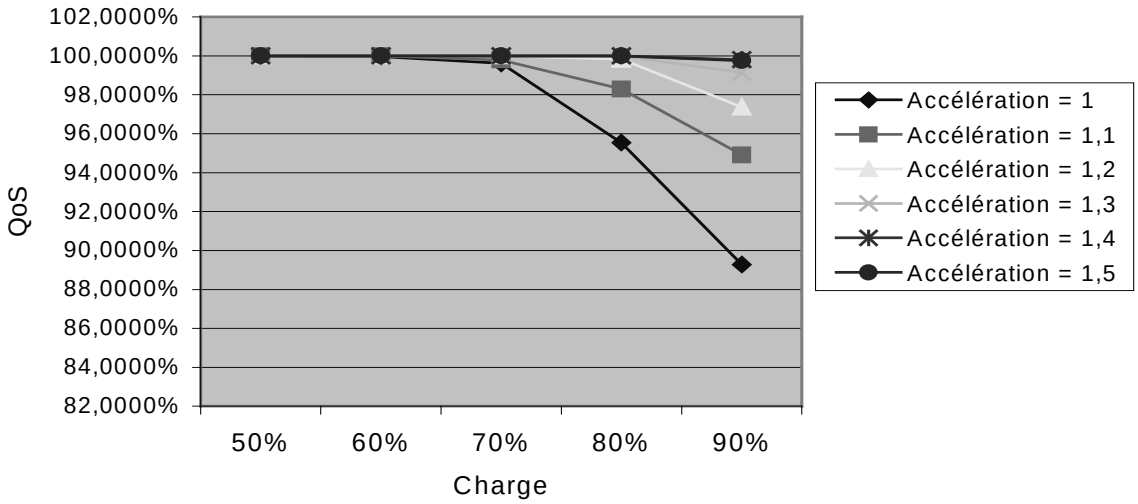

Figure 7.3 - Evaluation de l'accélération de la matrice

Le graphique suivant évalue le pourcentage de sélections de FIFO vides par rapport au nombre total d'appels aux fonctions d'ordonnancement en sortie. Ce taux est moins significatif que le taux de satisfaction des contrats de débit, mais il donne lui aussi une image sur la performance de l'architecture.

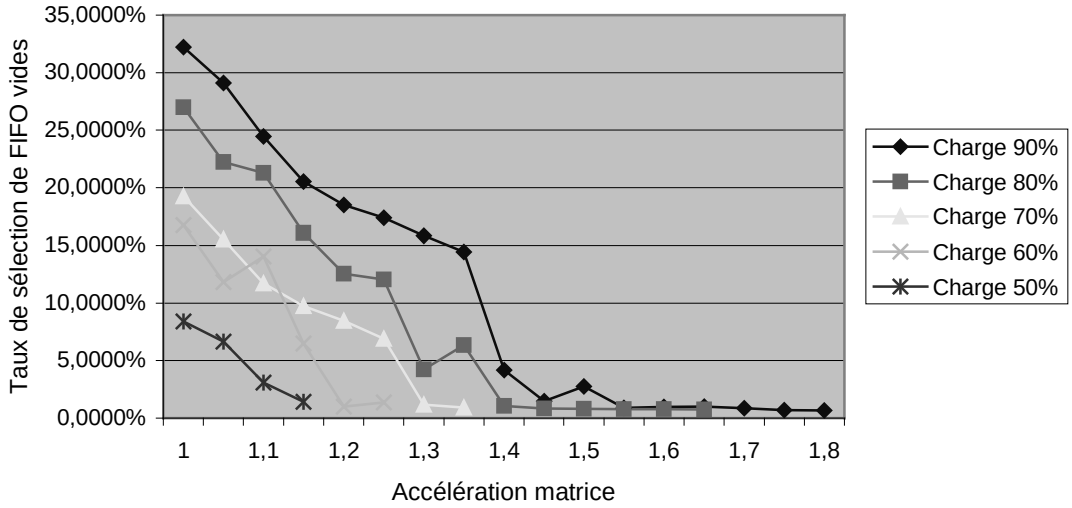

Figure 7.4 - Taux de sélection de FIFO vides en fonction de l'accélération de le matrice

Le taux de sélection des FIFO vides est inversement proportionnel au taux de réalisation de contrats de trafic. Le taux dépend de la charge de la matrice. Il faut remarquer qu'un taux de sélection de FIFO vides de 0% n'est jamais atteint.

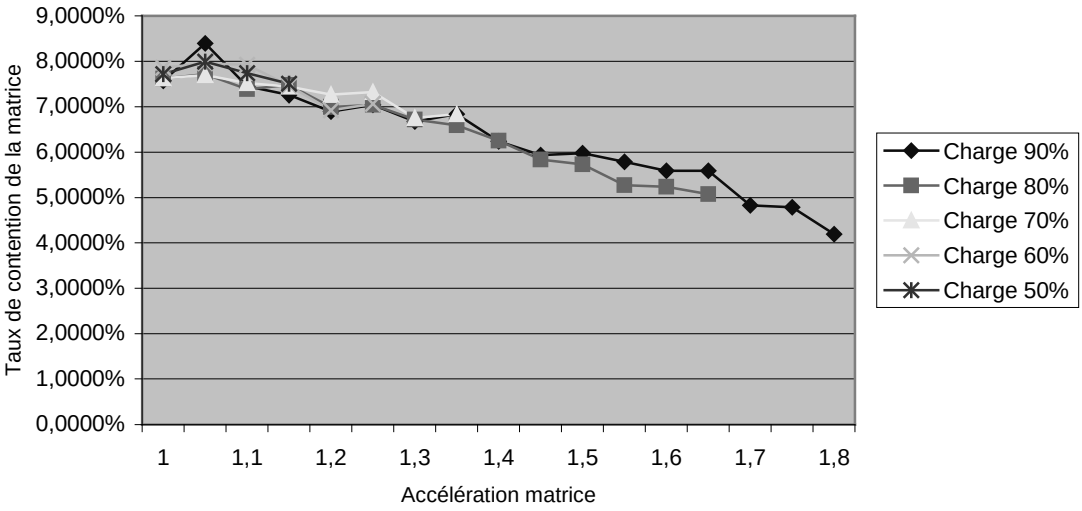

Figure 7.5 - Taux de contentions de la matrice de routage en fonction de l'accélération de la matrice

Le taux de contention de la matrice présente une décroissance linéaire par rapport à l'accélération de la matrice ce qui est un résultat logique. Ce taux ne semble pas dépendre du paramètre Charge. Ce résultat peut être interprété en remarquant que la charge effective correspondant à la somme des débits effectivement réalisés (c'est à dire $\sum D_{ijm\text{-}réel} / N$) est toujours proche de 90%, quelle que soit la valeur du paramètre Charge qui est une valeur « objectif ».

7.3.2 Influence du nombre des niveaux de priorité sur la performance de l'architecture

L'expérimentation suivante cherche à évaluer les performances de l'architecture en fonction du nombre de niveaux de priorité des notifications. L'évaluation est faite pour différentes valeurs du coefficient d'accélération de la matrice. Les conditions de la simulation sont celles du paragraphe précédent. Dans toutes les simulation une charge de 80% est considérée. Les paramètres qui sont modifiés d'une simulation à l'autre sont l'accélération de la matrice et le nombre de priorités.

Accélération de la matrice	Nombre de priorités	Charge effective	QoS	Taux de réalisation du contrat du flux le plus déficitaire	Taux de sélections de FIFO vides	Taux de contention	Latence moyenne	Latence maximale
1,5	1	91,5989%	92,4418%	86,4536%	31,3471%	8,4084%	123991	592741
1,5	2	91,5862%	100,0000%	100,0000%	2,1144%	5,5184%	198314	975907
1,5	4	91,3656%	100,0000%	100,0000%	0,8165%	5,7350%	194161	997443
1,5	8	91,5788%	100,0000%	100,0000%	1,8845%	5,6802%	180403	930917
1,5	16	91,5881%	100,0000%	100,0000%	2,1160%	5,5693%	182553	940367
1,5	32	91,5851%	100,0000%	100,0000%	0,2039%	5,6965%	182348	950526
1,25	1	90,7513%	91,5542%	87,4672%	32,2321%	8,7627%	75854	507839
1,25	2	91,5806%	99,9159%	97,2493%	10,4441%	6,7634%	245379	997132
1,25	4	91,6148%	99,9240%	98,9403%	11,9564%	6,8230%	239870	893462
1,25	8	91,5825%	99,9679%	97,4899%	9,9681%	6,7700%	240328	994149
1,25	16	91,5958%	99,9573%	99,6352%	8,9783%	6,8887%	245031	995470
1,25	32	90,8997%	99,9702%	98,8032%	8,7715%	6,9119%	246099	992336
1	1	91,3243%	88,0932%	92,9422%	35,6082%	8,4731%	196592	980033
1	2	90,8790%	96,3276%	96,8379%	24,4174%	7,7630%	273818	965439
1	4	91,5862%	96,2933%	99,9239%	24,7906%	7,5508%	275997	984323
1	8	90,5808%	95,0094%	98,9344%	26,3799%	7,5924%	275997	998328
1	16	90,6594%	95,6753%	99,9831%	26,2411%	7,6718%	275210	994121
1	32	89,9908%	95,0105%	97,5764%	25,9684%	7,8801%	270729	993051

Tableau 7.6 Résultats expérimentaux concernant l'influence du nombre de priorités et de l'accélération de la matrice sur la performance

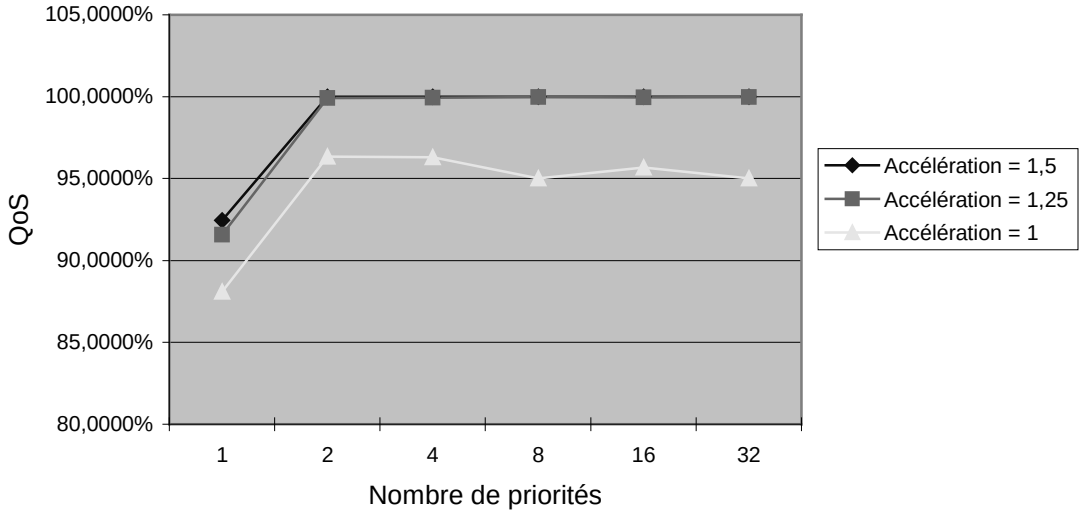

Figure 7.6 - Performance de l'architecture en fonction du nombre de priorités et de l'accélération

Les simulations montrent que les performances sont moins bonnes si toutes les notifications ont la même priorité. Il n'est pas nécessaire d'utiliser plus de 2 niveaux de priorité pour

améliorer considérablement la performance en termes de QoS. Avec 2 niveaux et pour une accélération de 1,25 les contrats sont satisfait à plus de 99,9%.

7.3.3 Evaluation de l'extensibilité de l'architecture

L'expérimentation vise à évaluer la mesure dans laquelle l'architecture est extensible. L'architecture est extensible si une augmentation du nombre de ports ne dégrade pas les performances. Le nombre de ports est augmenté en conservant les mêmes caractéristiques du trafic par port. D'une expérimentation à l'autre, la dispersion de débits, l'intensité de rafales et le nombre de flux par port sont maintenus constants. Le nombre total de flux reste proportionnel au nombre de ports.

Conditions de simulation :

- dimension de la matrice N = paramètre variable ;
- nombre maximal de sous-flux par paire de ports entrée-sortie M = paramètre variable;
- durée de la simulation EOS = 5 millions de cycles;
- taille maximale des FIFO matrice dms_max_mx= 64 mots;
- indice de multiplication de la fréquence des ports d'entrée in_ck = 1 ;
- indice de multiplication de la fréquence de la matrice mx_ck = 1.5 ;
- indice de multiplication de la fréquence des ports d'entrée out_ck = 1 ;
- taille maximale des FIFO sortie dms_max _out= 2048 mots;
- taille des paquets pkt_size = 60 mots ;
- pourcentage de flux non initialisés flux0_prc = 0.10 ;
- rapport entre le débit maximal et minimal d'un flux Dmaxmin = 8;
- Longueur maximale d'une rafale max_brst_leng = 8;
- charge globale de la matrice Charge = 0.80 ;
- Nombre de priorités utilisées par l'algorithme prio_range = 4;

Tableau d'évaluation de l'influence du nombre de ports :

Nombre de ports	Nombre total de flux	Charge effective	QoS	Taux de réalisation du contrat du flux le plus déficitaire	Taux de sélections de FIFO vides	Taux de contention matrice	Latence moyenne	Latence maximale
2	367	90,8968%	100%	100%	0,0349%	22,2147%	117453	577872
4	720	91,6380%	100%	100%	0,1899%	11,3816%	96472	654599
8	1429	90,5298%	100%	100%	2,1519%	5,0335%	93278	568327
16	2754	91,2135%	99,5552%	98,2421%	16,5101%	2,3968%	77265	549573
32	5505	90,5392%	98,8446%	97,6327%	29,3655%	1,1110%	61532	593332
64	11520	90,5392%	98,2743%	97,2137%	36,5937%	0,9822%	63287	578293

Tableau 7.7 - Résultats expérimentaux concernant l'influence du nombre de ports sur la performance

L'expérimentation prouve que l'architecture est extensible et que l'influence du nombre de ports sur la qualité de service est faible. Jusqu'à un nombre de 16 ports, une qualité de service proche de 100% est garantie. Entre 32 et 64 ports les défauts de qualité de service sont en dessous de 2%.

7.3.4 Evaluation de l'influence du nombre de flux sur la performance de l'architecture

L'expérimentation antérieure prouve une légère détérioration de la qualité de services pour un nombre élevé de ports. Il est possible de faire l'hypothèse que le nombre de flux définis à travers la matrice contribue à cette légère détérioration. La simulation suivante évalue l'influence de ce paramètre sur la performance de l'architecture.

Conditions de simulation :

- dimension de la matrice N = 8 ;
- nombre maximal de sous-flux par paire de ports entrée-sortie M = paramètre variable;
- durée de la simulation EOS = 5 millions de cycles;
- taille maximale des FIFO matrice dms_max_mx= 64 mots;
- indice de multiplication de la fréquence des ports d'entrée in_ck = 1 ;
- indice de multiplication de la fréquence de la matrice mx_ck = 1.5 ;
- indice de multiplication de la fréquence des ports d'entrée out_ck = 1 ;
- taille maximale des FIFO sortie dms_max _out= 2048 mots;
- taille des paquets pkt_size = 60 mots ;
- pourcentage de flux non initialisés flux0_prc = 0.10 ;
- rapport entre le débit maximal et minimal d'un flux Dmaxmin = 8;
- Longueur maximale d'une rafale max_brst_leng = 8;
- charge globale de la matrice Charge = 0.80 ;
- Nombre de priorités utilisées par l'algorithme prio_range = 4;

D'une simulation à l'autre le seul paramètre qui varie est le nombre de flux par port. Tableau d'évaluation de l'influence du nombre de flux :

Nombre de flux par paire de ports	Nombre total de flux	Charge effective	QoS	Taux de réalisation du contrat du flux le plus déficitaire	Taux de sélections de FIFO vides	Taux de contention	Latence moyenne	Latence maximale
10	573	91,6087%	99,9719%	99,8993%	19,1150%	4,5344%	18948	320717
40	2299	91,5795%	100%	100%	1,7755%	5,2129%	24544	822845
70	4044	91,5939%	100%	100%	1,9513%	5,6794%	37745	992319
100	5767	91,3656%	100%	100%	1,9777%	5,7350%	38192	998184
130	7532	91,5824%	99,7898%	99,9898%	2,1616%	5,5204%	45946	999060
160	9196	91,6089%	99,7088%	98,8345%	2,1613%	5,1530%	50598	1000060
190	10953	91,6141%	98,8619%	99,6199%	1,9173%	4,5635%	95813	999945
220	12702	91,5755%	98,1501%	98,5975%	1,9779%	4,2940%	99660	1000004

250	14418	91,5904%	97,1202%	98,1965%	1,9710%	4,2574%	132801	999991
280	16158	91,5785%	95,6056%	98,7069%	2,2464%	3,9970%	143630	999970
310	17842	91,5710%	94,4204%	99,5064%	2,1025%	4,0473%	215592	993340
340	19603	91,6112%	93,3442%	98,5365%	2,2251%	3,9751%	277705	1000103
370	21311	91,5821%	92,5073%	97,7343%	2,0711%	3,0614%	241334	1000007

Tableau 7.8 - Résultats expérimentaux concernant l'influence du nombre de flux sur la performance

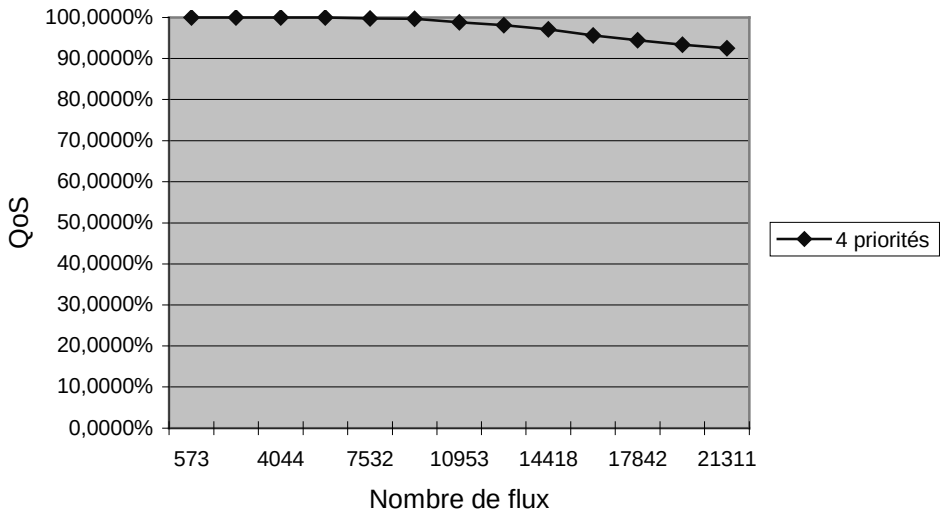

Figure 7.7 - Performances de l'architecture en fonction du nombre de flux

A charge totale constante, une augmentation du nombre de flux résulte dans une légère dégradation des performances. Pour pouvoir garantir les contrats de qualité de services il faut augmenter l'accélération de la matrice ou baisser la charge du réseau.

7.3.5 Comparaison des stratégies d'ordonnancement

D'autres stratégies d'ordonnancement que le mécanisme de notification proposé dans le chapitre 5 avaient été envisagées. Les expérimentations suivantes comparent trois stratégies d'ordonnancement pour les ports d'entrée :

- stratégie qui utilise le système de notifications ;
- stratégie qui approvisionne la FIFO destination la plus déficitaire ;

- stratégie aléatoire (pas d'optimisation).

L'algorithme "FIFO la plus déficitaire" consiste, pour chaque port de sortie, à demander l'alimentation de la FIFO de sortie la moins pleine. Plus précisément, chaque fois que l'ordonnanceur du port de sortie j consomme un paquet dans une FIFO, il envoie une notification au port d'entrée i correspondant à la FIFO la moins pleine du port j. Toutes les notifications arrivant sur le port i sont stockées dans une unique file de notifications (quelle que soit la provenance), et traitées par l'ordonnanceur du port i dans l'ordre d'arrivée. Si plusieurs FIFO du port j sont vides, la FIFO à alimenter est choisie au hasard parmi les FIFO vides.

Conditions de simulation :

- dimension de la matrice N = 8 ;
- nombre maximal de sous flux par paire de ports entrée-sortie M = 100;
- durée de la simulation EOS = 5 millions de cycles;
- taille maximale des FIFO matrice dms_max_mx= 64 mots;
- indice de multiplication de la fréquence des ports d'entrée in_ck = 1 ;
- indice de multiplication de la fréquence de la matrice mx_ck = 1.5 ;
- indice de multiplication de la fréquence des ports d'entrée out_ck = 1 ;
- taille maximale des FIFO sortie dms_max _out= 2048 mots;
- taille des paquets pkt_size = 60 mots ;
- pourcentage des flux non initialisés flux0_prc = 0.10 ;
- rapport entre le débit maximal et minimal d'un flux Dmaxmin = paramètre variable;
- Longueur maximale d'une rafale max_brst_leng = paramètre variable;
- charge globale de la matrice Charge = 0.80 ;
- Nombre de priorités utilisées par l'algorithme prio_range = 4

Les performances des ces stratégies d'ordonnancement seront évaluées pour des caractéristiques de trafic différentes. Un paramètre qui caractérise la dispersion du trafic est Dmaxmin. Le paramètre représente le rapport entre les valeurs maximale et minimale du débit d'un flux. Un autre paramètre qui caractérise le trafic est la valeur maximale de rafales max_brst_leng. Pour une première série de mesures la valeur du paramètre Dmaxmin est positionnée à 8 et celle de la dimension maximale des rafales à 8.

Architecture	Charge effective	QoS	Taux de réalisation du contrat du flux le plus déficitaire	Taux de sélections de FIFO vides	Taux de contention	Latence moyenne	Latence maximale
Notifications	91,7378%	100,0000%	100,0000%	0,8147%	5,5048%	277364	998133
FIFO déficitaire	88,3749%	87,9379%	73,6725%	55,4755%	5,8359%	64658	324542
Aléatoire	91,2859%	84,9292%	63,3618%	50,1684%	5,1058%	131361	839508

Tableau 7.9 - Résultats expérimentaux concernant la comparaison des architectures pour Dmaxmin=8 et max_brst_leng=8

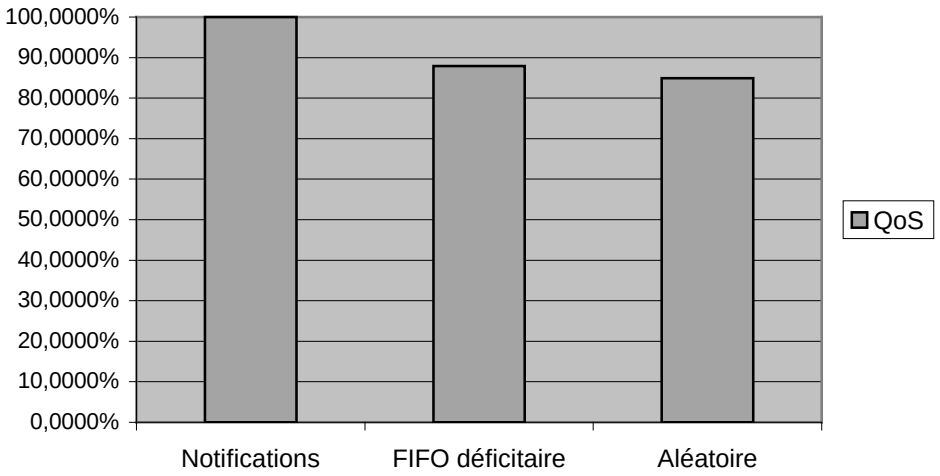

Figure 7.8 - Taux de réalisation de contrats de trafic en fonction de l'architecture

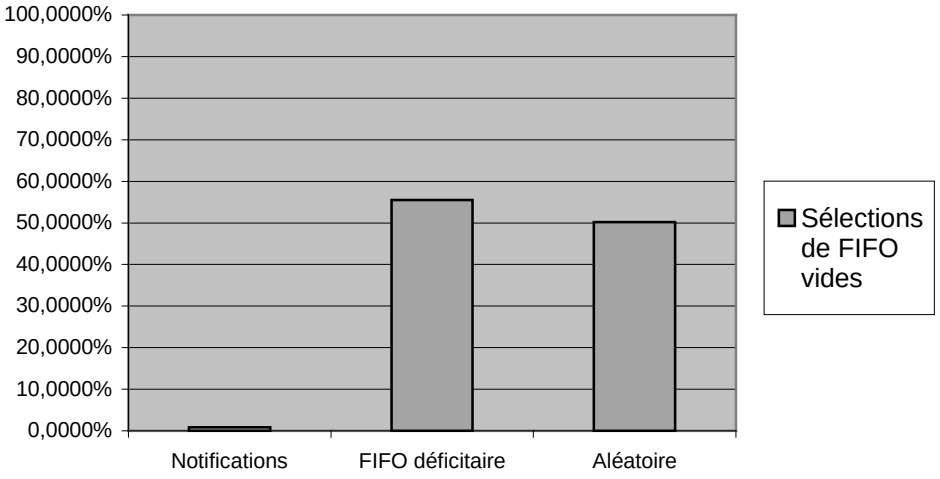

Figure 7.9 - Taux de sélections de FIFO vides en fonction de l'architecture

Dans ces conditions, la stratégie "notifications" présente de meilleures performances que la stratégie "FIFO déficitaire". Ce résultat peut être expliqué par le fait que la stratégie "FIFO déficitaire" est inefficace dans le cas d'un trafic avec un comportement irrégulier et beaucoup de rafales. Les mesures suivantes sont faites pour des caractéristiques de trafic plus régulières.

Architecture	Charge effective	QoS	Taux de réalisation du contrat du flux le plus déficitaire	Taux de sélections de FIFO vides	Taux de contention	Latence moyenne	Latence maximale
Notifications	91,5978%	100,0000%	100,0000%	1,2354%	5,3437%	149676	827379
FIFO déficitaire	89,3441%	98,8152%	96,3462%	3,3843%	8,0602%	255432	884369
Aléatoire	91,3609%	85,2616%	75,4562%	51,3984%	5,4526%	143267	823452

Tableau 7.10 - Résultats expérimentaux concernant la comparaison des architectures pour Dmaxmin=1 et max_brst_leng=8

Architecture	Charge effective	QoS	Taux de réalisation du contrat du flux le plus déficitaire	Taux de sélections de FIFO vides	Taux de contention	Latence moyenne	Latence maximale
Notifications	93,0081%	100,0000%	100,0000%	0,0000%	5,1066%	278973	980032
FIFO déficitaire	88,2214%	99,9516%	99,0857%	0,0000%	8,8336%	292832	997020
Aléatoire	91,6986%	85,6166%	70,3655%	52,8039%	5,0456%	163432	802215

Tableau 7.11 - Résultats expérimentaux concernant la comparaison des architectures pour Dmaxmin=8 et max_brst_leng=1

Architecture	Charge effective	QoS	Taux de réalisation du contrat du flux le plus déficitaire	Taux de sélections de FIFO vides	Taux de contention	Latence moyenne	Latence maximale
Notifications	93,0129%	100,0000%	100,0000%	0,0000%	4,9259%	271356	596867
FIFO déficitaire	89,7913%	100,0000%	100,0000%	0,0000%	9,6688%	228896	459371
Aléatoire	91,3680%	86,0937%	73,1356%	49,8825%	4,5521%	188208	762509

Tableau 7.12 - Résultats expérimentaux concernant la comparaison des architectures pour Dmaxmin=1 et max_brst_leng=1

Les résultats des simulations sont rassemblés dans des graphiques comparatives.

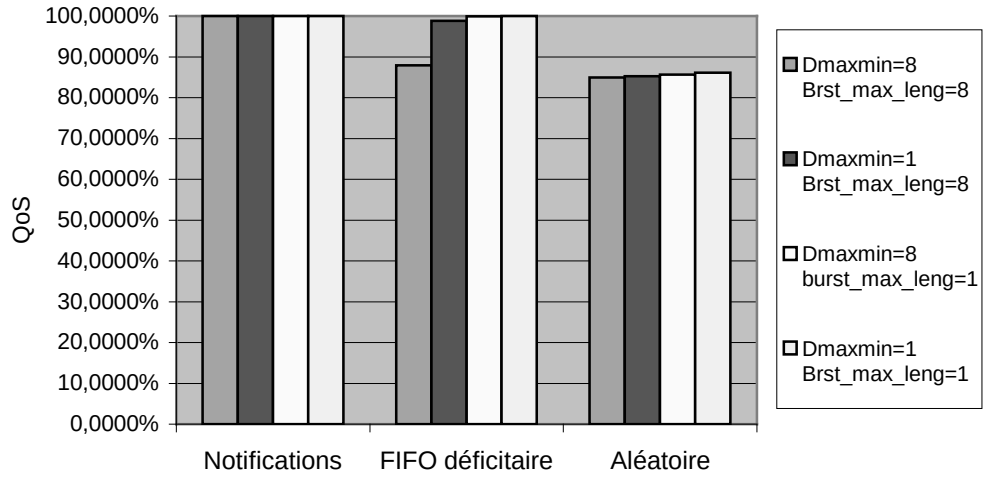

Figure 7.10 - Taux de réalisation de contrats de trafic en fonction de l'architecture et des caractéristiques du trafic

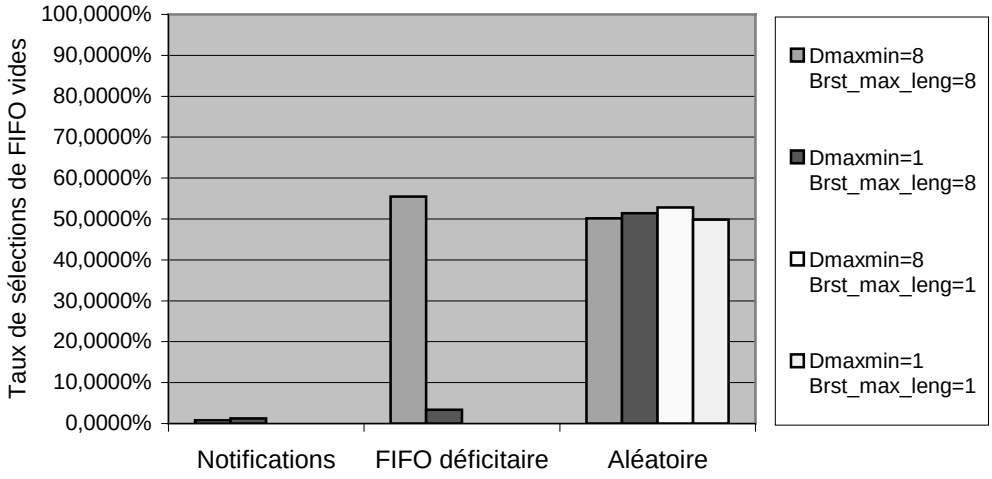

Figure 7.11 - Taux de sélections de FIFO vides en fonction de l'architecture et des caractéristiques du trafic

Les simulations prouvent qu'une des causes des faibles performances de la stratégie « FIFO déficitaire » est le comportement irrégulier du trafic d'entrée. C'est un argument en faveur du mécanisme des notifications qui conserve une bonne performance en matière de qualité de service indépendamment des caractéristiques du trafic.

7.4 Conclusions

Les mesures de performance ont visé l'évaluation d'une stratégie d'ordonnancement en entrée à base de notifications. Les simulations prouvent que l'architecture offre des garanties de qualité de service élevées, qu'elle est extensible et que les performances peuvent être maintenues dans des conditions de trafic avec un comportement irrégulier.

Conclusions et perspectives

Conclusions

Le but de la recherche était de définir une architecture de commutateur ATM haut débit avec respect de la qualité de service. La problématique a montré que l'impact de la qualité de service sur l'architecture matérielle est substantiel et que le coût matériel dépend considérablement du niveau de qualité de services fournit par l'architecture. Nous avons aussi montré que pour une bande passante élevée il est difficile de garantir un niveau élevé de QoS.

L'analyse de quelques exemples d'architectures de commutation a révélé que la source la plus importante de dégradation de la qualité des services est la contention interne. L'objectif que nous avons fixé a été de concevoir un équipement de communication utilisant le protocole ATM qui possède un valence élevée et qui en même temps permet de garantir la qualité des services de chaque connexion établie à travers le système. Dans ce but la recherche a été orientée sur deux axes :

- un axe qui vise un débit important pour une architecture à commutation de circuit ;
- un axe qui vise le mise a niveau de la qualité de service d'une architecture à commutation de paquets.

Dans la première partie nous avons étudié les possibilités d'exploitation des performances en termes de qualité de service offertes par une architecture basée sur un bus à réservation de bande passante. Nous avons pris comme point de départ de l'étude le commutateur SafeCom4000 organisé autour du bus partage ATM. Nous avons énoncé les principes d'une nouvelle architecture SafeCom4000X qui reprend le principe du bus partagé dans le but de réaliser un cluster de sous-systèmes liés par des liaisons haut débit bidirectionnelles. Nous avons démontré la faisabilité de l'architecture proposée en décrivant en détail le composant F2F. La fonction principale du circuit est de relier entre eux deux bus ATM. Jusqu'à un nombre de trois sous-systèmes, ce qui représente un débit total commuté de 1,8 Gb/s, on conclut que l'architecture SafeCom4000X fournit un niveau de QoS équivalent à celui de l'équipement SafeCom4000. Une architecture de commutateur utilisant la technique du bus

partagé et reposant sur le principe de commutation de circuit peut-être étendue sans dégradation des garanties de service mais cette extension reste limitée.

Dans la deuxième partie nous nous sommes intéressés à une architecture organisée autour d'une matrice de commutation de type réseau multi-étages, en principe beaucoup plus modulaire et qui possède des bonnes caractéristiques d'extensibilité. Le fait qu'il n'y a plus de réservation explicite de bande passante complique la garantie de service. La qualité de service est fortement détérioré par la contention interne.

Nous avons montré comment la performance d'une matrice de commutation multi-étages peut être améliorée par la mise en place de stratégies d'ordonnancement du trafic entrant. Le trafic entrant est trié à l'aide des notifications qui servent à compenser les consommations des cellules dans les files d'attente de sortie. Dans notre vision de l'architecture, les contrats de qualité de service restent entièrement à la charge des ordonnanceurs de sortie. Les notifications sont classées par priorités ce qui signifie que les FIFO déficitaires sont alimentées en priorité. Le mécanisme de compensation que nous avons mis au point utilise comme seule information le taux de remplissage de la file d'attente de sortie correspondante. Le fait qui permet l'acquittement complet de ces contrats est qu'aucune des files d'attente de sortie n'est jamais vide.

La validation du principe architectural a nécessité la mise en place d'un simulateur générique pour un commutateur utilisant une matrice à commutation de paquets. L'utilité du simulateur est d'évaluer de façon qualitative les performances de l'algorithme proposé. Le simulateur est caractérisé par un dégréé de généralité qui consiste dans le fait qu'il permet à l'utilisateur de définir les dimensions du réseau et les caractéristiques du trafic. Nous avons prévu un paramètre qui permet de modifier l'accélération de la matrice dans le but de pouvoir augmenter le débit maximal offert par la matrice de commutation.

Les simulations que nous avons effectuées ont prouvé que l'architecture utilisant des notifications permet de garantir une satisfaction de contrats de QoS à 100% dans des conditions qui ont été déterminées de façon quantitative. Nous pouvons conclure que même pour des charges de trafic importantes il est possible de garantir une QoS de 100% utilisant une accélération de la matrice facilement réalisable. Les résultats expérimentaux donnent aux utilisateurs la possibilité de dimensionner le réseau en fonction de la performance souhaitée.

Nous avons également prouvé que l'objectif de concevoir une architecture extensible a été atteint. Nous avons effectué une étude comparative de plusieurs stratégies d'ordonnancement qui a montré que la stratégie de notification reste la plus performante indépendamment des caractéristiques du trafic.

Perspectives

L'étude d'une architecture basée sur un bus à réservation de bande passante a été menée dans le but d'une implémentation matérielle de la plate-forme 1,8Gb/s SafeCom4000X. Cette étude n'a pas été finalisée à cause des difficultés financières de l'entreprise qui a soutenu le projet. Une des perspectives serait de faire aboutir ce projet.

En ce qui concerne l'architecture basée sur la commutation de paquets, les simulations présentent la latence comme un résultat de contraintes portant sur les débits minimaux qui doivent être garantis pour les différents flux. La méthode utilisée pour la définition des caractéristiques du trafic modélise le caractère plus ou moins agressif des rafales de chaque flux mais ne prend pas en compte les contraintes sur la latence ou sur les taux de pertes. Notre étude montre à quelles conditions et à quel coût il est possible de garantir des débits minimaux pour les différents flux (ce qui constitue qu'une partie de la QoS) avec une architecture de commutateur utilisant une matrice à commutation de paquets. Cette architecture a en effet le mérite d'être réellement extensible. Il reste cependant à analyser comment le mécanisme de notifications se comporte lorsque les contraintes de latence et de taux de pertes sont prises en compte.

Annexes

A.1 Le principe de management du SafeCom4000X

A.1.1 Le gestionnaire ASC (*ATM Software Component*)

SafeCom4000X est un cluster mais il doit être géré comme équipement unique. La méthode de couplage permet de modifier seulement le plan utilisateur de l'équipement, l'interface entre le plan utilisateur et les autre plans reste la même et les modifications ne portent pas sur les autres plans. L'ASC (*ATM Software Component*), le gestionnaire du plan utilisateur prends en charge le changement matériel pour masquer vis-à-vis des autres plans l'aspect cluster. L'ASC est situé sur la carte SUPATMX et il accède à l'ensemble des fonctions matérielles des sous-systèmes par l'intermédiaire du IOBUS disponible en fond de panier. L'IOBUS est unique à tous les circuits FACE et F2F. Des informations relatives à la topologie sont disponibles en fond de panier pour renseigner l'ASC sur la manière d'interconnecter les cartes en fonction des versions du fond de panier. La figure 3.22 présente le principe de management du SafeCom4000.

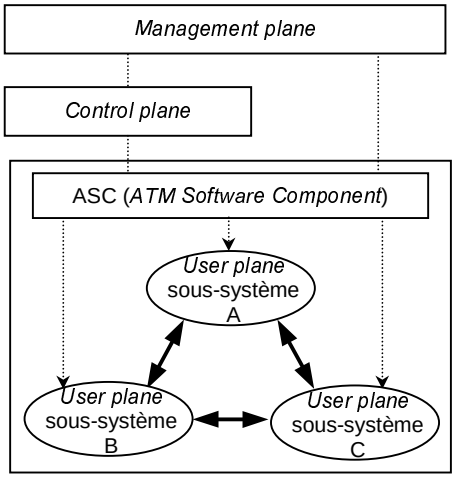

Figure A.1 - Cluster d'équipements : le plan utilisateur masque l'aspect cluster

Le système de décodage d'en-tête ATM utilisé par l'extension VC nécessite de connaître à priori la largeur de champ VCI qui devra être décodé. En plus le système nécessite la mise en place d'une hiérarchie d'identificateurs au niveau cluster et au niveau sous-système. En d'autres termes, le pilotage logiciel de l'architecture matérielle par l'intermédiaire de l'ASC nécessite l'affectation de différentes variables supplémentaires. D'un autre coté, l'aspect cluster n'apporte pas de dépendances entre les sous-systèmes. Par exemple, l'identificateur d'un circuit FACE dans un sous-système donné peut prendre la même valeur pour un autre FACE dans un autre cluster.

Deux types d'identifications sont définis: l'identification des entités (ICAF, RISAF, ISAF, sous-système) et l'identification des connexions au travers des différents blocs.

A.1.2 Identification des entités dans l'équipement

La figure A.2 représente le couplage entre 2 sous-systèmes et l'identification de blocs et de sous-systèmes. Il existe des identificateurs locaux à un sous-système et des identificateur de niveau cluster.

La partie IACF du composant FACE, possède une liste de plages temporelles pendant lesquelles le bloc prend le contrôle du bus ATM et initie des transferts par l'émission de requêtes. Les destinataires de ces requêtes sont les blocs ISAF de circuits FACE et les blocs RISAF de circuits F2F. L'adressage est fait à l'aide d'un identificateur nommé IDCS. Chaque ISAF possède son propre identificateur qui désigne aussi la ligne BR sur laquelle le bloc émet des notifications. Les RISAF occupent deux IDCS consécutifs (un numéro pair, et le numéro impair suivant). Cette restriction est issue du fait que RISAF utilise deux lignes BR pour écouler des notifications de cellules qu'il est impossible d'écouler sur une ligne BR unique. Une notification prend deux plages temporelles ce qui signifie qu'une ligne BR supporte des notifications d'un débit équivalent de 330Mb/s. RISAF utilise deux lignes BR et implicitement occupe deux IDCS consécutifs pour pouvoir écouler justement un débit de 660 Mb/s. L'adressage d'un RISAF est fait donc à l'aide d'un IDCS pair pour initier des transferts montants. A l'aide de ce même IDCS sont initiés aussi les transferts descendants destinés aux sous-systèmes distants.

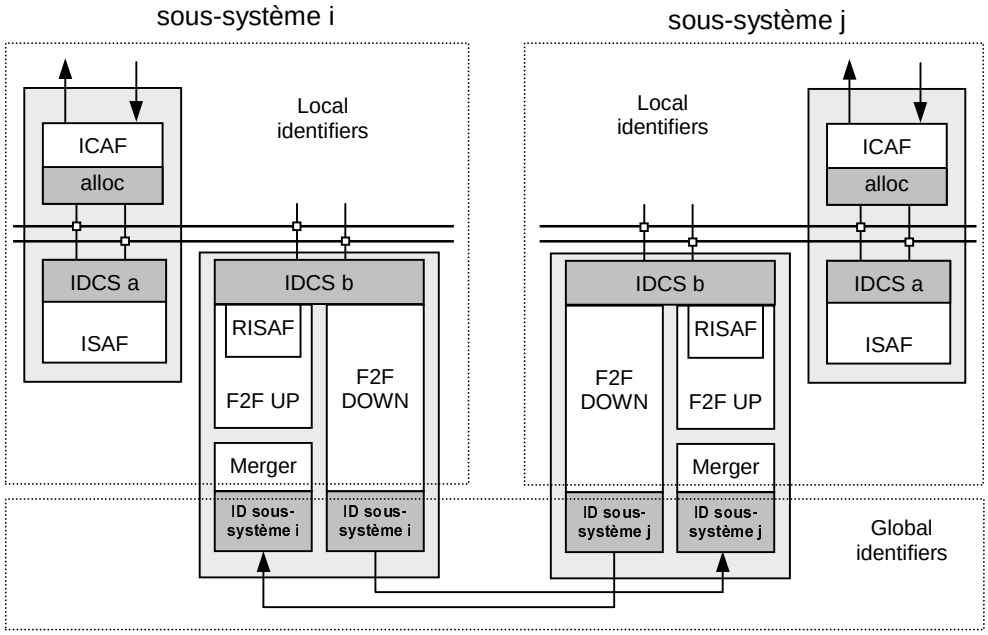

Figure A.2 - Identification de blocs

Chaque sous-système est indépendant et les affectations d'identificateurs IDCS dans un sous-système ne sont pas liées aux affectations prises dans les autres sous-systèmes. En ce qui concerne la distribution des identificateurs globaux nécessaires pour réaliser des architectures avec concentration, ceux-ci prennent des valeurs entre 0 et 7. Cette valeur est inscrite dans un registre d'identification du circuit F2F.

A.1.3 Identification des connexions à travers l'équipement

Le trace des connexions donne une vision statique sur l'affectation des ressources effectuée par l'ASC. La trace des connexions donne aussi une vision dynamique sur les traitements effectués sur une cellule à chaque étape dans l'équipement.

En ce qui concerne l'extension VC, l'équipement a deux modes de fonctionnement: l'extension VC invalidée et l'extension VC validée. Cette validation est dynamique, cellule à cellule, en fonction d'un bit d'information DEMUX positionné par le bloc ICAF à l'initialisation de chaque transfert.

A.1.3.1 Extension d'adresses VC invalidée (DEMUX = 0)

Le schéma de la figure A.3 présente les ressources réservées lors de l'établissement d'une connexion de niveau VC ou de niveau VP quand l'extension d'adresses VC est invalidée.

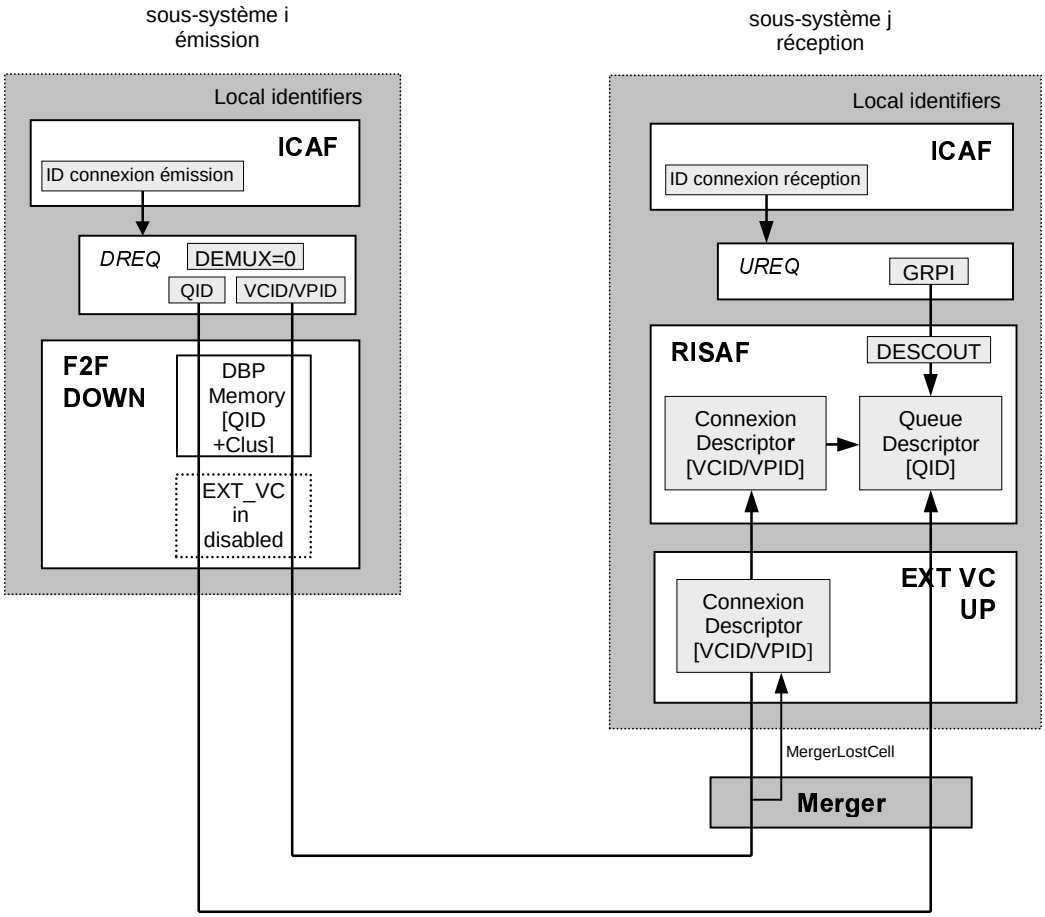

Figure A.3 - Trace de connexions dans le cas extension d'adresses VC invalidée

Lors de l'ouverture d'une connexion α, un descripteur de connexion VCID/VPID[j] est choisi dans EXT VC UP dans l'ensemble des descripteurs disponibles dans le F2F du sous-système destination ($F2F_i$réception). Le descripteur et l'identificateur de la file d'attente sont présents dans la requête descendante et font partie du message envoyé sur les liens HSL. Le descripteur $F2F_j$(réception).EXT_VC.Connection_descriptor[j] est donc associé à la connexion α. De la même manière, un descripteur $F2F_{i(}$(émission).RISAF.Connection_descriptor[j] est associée à cette même connexion α.

La catégorie de services peut être de type file dédiée ou file partagée. Dans le cas de la file dédiée, l'identificateur de file « k » $F2F_j$(réception).RISAF.Queue_identifier[k] est associé de manière unique et bi-univoque à cette même connexion α. Dans le cas de la file partagée l'identificateur de file « k » $F2F_j$(réception).RISAF.Queue_identifier[k] est associée à cette connexion α, et éventuellement associé à d'autres connexions partageant cette même file. Pour des raisons de simplicité uniquement le cas de la file partagé est traité (DEMUX=1).

La diffusion de la cellule vers plusieurs tampons RISAF distants peut être activée en positionnant plusieurs bits dans le champ de bits destination de la requête descendante. La réservation des descripteurs $F2F_{i...n}$(émission).VC_EXT.Connection_descriptor[j], $F2F_{i...n}$(émission).RISAF.Connection_descriptor[j], $F2F_{i...n}$(réception).RISAF.Queue_identifier[k] doit être possible. Lors de l'ouverture de connexion, la problématique consiste à trouver des index j et k tels qu'ils soient disponibles pour les différents F2Fs/RISAFs impliqués dans la diffusion.

Les descripteurs $F2F_i$(réception).VC_EXT.Connection_descriptor[j] et $F2F_i$(réception.RISAF.Connection_descriptor[j] portent des informations redondantes comme le comptage de cellules et la translation d'en-tête. En pratique, le comptage et la translation peuvent se faire indifféremment dans EXT_VC ou RISAF. Par contre, l'option de marquage EFCI est disponible seulement dans RISAF et non dans EXT_VC. Inversement, le compteur de cellules perdues par Merger dans les architectures avec concentration est disponible uniquement dans EXT_VC.

Bloc	Opérations
ICAF down	La cellule est manipulée à l'aide d'un identificateur de connexion « ID conn down » Le contexte DREQ fournit ses nouveaux identificateurs : VCID (si connexion niveau VC) ou VPID (si connexion niveau VP) et QID (Queue Identifier). Le bit DEMUX est positionné à 0. L'information DBP retournée par le bus ATM permet d'être Virtual Destination en ABR (QID<2K)
F2F down	L'information DBP du bus ATM est renvoyé en temps-réel vers ICAF down. La cellule est placée dans une trame HSL et les différents champs (VCID/VPID, QID, DEST_FID) remplis à partir des informations DREQ. Le champ SRC ID (Cluster ID / Board ID) est lu localement. La cellule est émise sur HSL.
Merger	Merger filtre les cellules pour le F2F local à partir du champ DEST_FID. Pas de traitement sur identificateurs logiques sauf si la cellule est perdue par saturation de Merger, une notification de perte (paramètre : VPID/VCID) est alors envoyée vers F2F up (VC_EXT out).
VC_EXT F2F up	La cellule est émise vers RISAF. Si RISAF répond par une saturation alors le compteur de cellules perdues est incrémenté sinon le compteur de cellules acceptées est incrémenté. L'EPD est géré si l'option est validée. De manière asynchrone, les notifications de cellules perdues dans Merger sont traduites par un incrément des compteurs de cellules perdues par Merger. VPID/VCID sert d'index pour ces opérations.

	NOTA : La translation est inhibée à cet étage.
RISAF in	RISAF met dans la file d'attente « QID » la cellule, VCID/VPID est attaché à la cellule. Les compteurs de cellules perdues/acceptées indexés par VCID/VPID sont mis à jour. Une notification « ID conn up » est émise vers ICAF up.
RISAF out	Suite à la réception d'un GRPI indiquant la présence d'une cellule dans la file d'attente, le processus d'émission est lancé : Le contexte DESCOUT[GRPI] est lu et les paramètres , QID et « j » mémorisés. La cellule en tête de file « QID » est extraite. Le champ VCID/VPID attaché à la cellule est lu. Le contexte de la connexion pointée par VCID/VPID permet de lire une variable d'indirection IP qui est additionnée à « j ». le résultat de l'addition pointe sur la nouvelle en-tête à donner à la cellule (nouveau champ VPI/VCI). NOTA : j=0 en cas d'unicast et J = 0…n-1 en cas de diffusion vers n sorties.
ICAF up	L'ICAF up reçoit les différentes notifications de RISAF in et les introduit dans son échéancier. Le échéancier élit une connexion « ID conn up » et lit le contexte UREQ contenant le champ GRPI lequel est émis sur le bus ATM. ICAF reçoit la cellule et exécute ses traitements selon le contexte de connexion indexé par « ID conn up ».

Tableau A.1 - Trace du transfert d'une cellule dans le cas extension d'adresses VC invalidée

A.1.3.2 Extension d'adresses VC validée (DEMUX = 1)

Le schéma de la figure A.4 présente les ressources réservées lors de l'établissement d'une connexion de niveau VC uniquement. En effet, le décodage des connexions au travers de l'extension VC est valable uniquement au niveau VC, les capacités de FACE au niveau VP étant suffisantes.

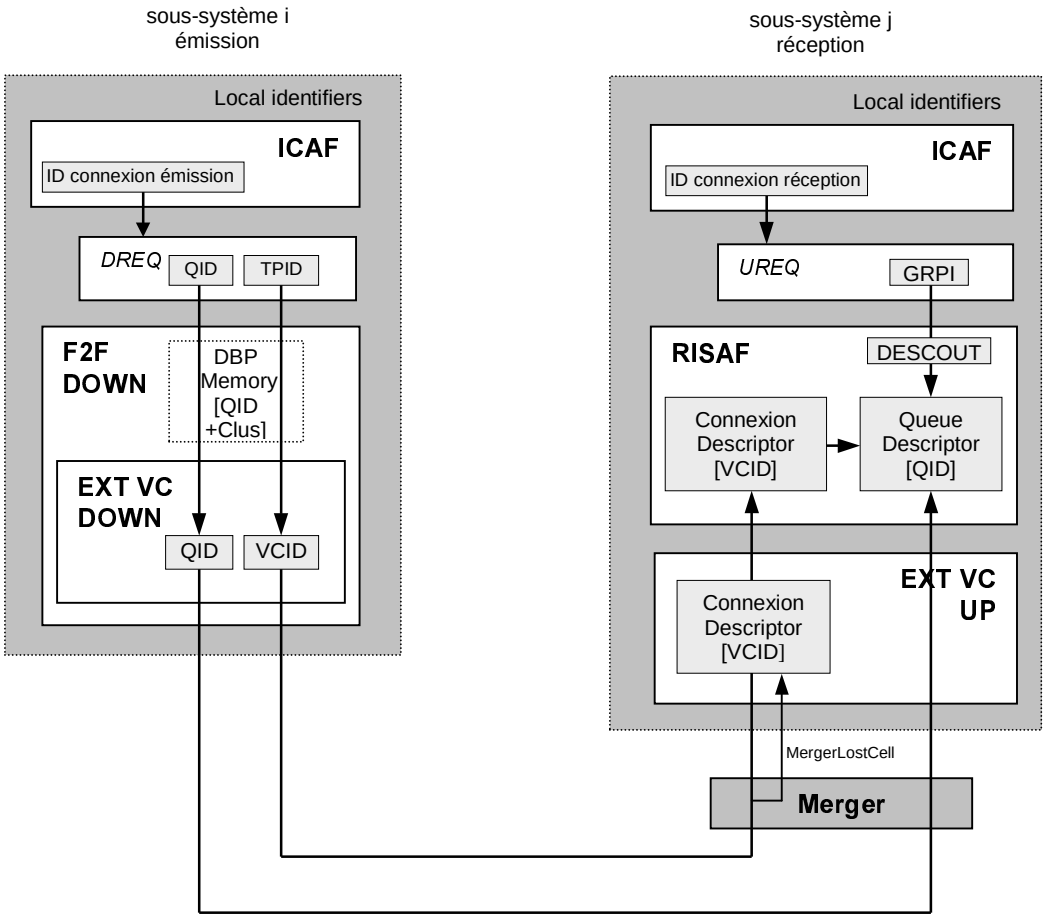

Figure A.4 - Trace de connexions dans le cas extension d'adresses VC validée

Lors de l'ouverture d'une connexion α, un descripteur de connexion VCID[j] d'EXT VC UP est choisi dans l'ensemble des descripteurs de F2F disponibles sur le sous-système de destination $F2F_i$(réception). Le descripteur et l'identificateur de la file d'attente sont présents dans la requête descendante et font partie du message envoyé sur les liens HSL. Le descripteur $F2F_i$,(réception).VC_EXT.Connection_descriptor[j] est donc associé à la connexion α. Le descripteur $F2F_i$(réception).RISAF.Connection_descriptor[j] n'est pas associé à cette même connexion α. Les connexions traitées avec DEMUX=1 pointent dans RISAF vers un des deux descripteurs communs $F2F_i$(réception).RISAF.Connection_descriptor[j=0] ou $F2F_i$(réception).RISAF.Connection_descriptor[j=1], qui correspondent à une absence de translation et respectivement à l'option de marquage EFCI inactivée (j=0) ou activée (j=1). La catégorie de service peut être de type file partagée. Dans ce cas le descripteur de file « k »

$F2F_i$(réception).RISAF.Queue_descriptor[k]) est associé à cette connexion α, mais éventuellement associé à d'autres connexions partageant cette même file.

Les connexions de niveau VP traitées avec DEMUX=1 ne peuvent pas être diffusées, (pas de *cross-connect* VP).

Blocs	Opérations
ICAF down	La cellule n'est pas reconnue par FACE, elle se voit donc affecter un identificateur commun avec les autres cellules non reconnues dans le même conduit physique. Le contexte DREQ fournit ses nouveaux identificateurs : TPID (Transport path IDentifier). Le champ QID est invalide. Le bit DEMUX est positionné à 1.
F2F down	DEMUX = 1 provoque le décodage du triplet (TPID, VPI, VCI). Un identificateur VCID est fourni en retour si une connexion VC est reconnue. Dans le cas contraire un compteur de cellules « unallocated » est incrémenté par conduit physique TPID si le VP n'existe pas ou par VP si le VC n'existe pas. Grâce à VCID, EXT_VC extrait de ses tables les valeurs QID, DEST_FID et autres variables et les met dans le message CELL de la trame HSL.. Le champ SRC ID est lu localement. La cellule est alors émise sur HSL.
Merger	Merger filtre les cellules pour le F2F local à partir du champ DEST_FID. Pas de traitement sur identificateurs logiques sauf si la cellule est perdue par saturation de Merger, une notification de perte (variable : VCID) est alors envoyée vers F2F up (VC_EXT out).
F2F up VC_EXT	La cellule est émise vers RISAF. Si RISAF répond par une saturation alors le compteur de cellules perdues est incrémenté sinon le compteur de cellules acceptées est incrémenté. L'EPD est géré si l'option est validée. La translation d'en-tête VPI/VCI est opérée à cet étage. Selon l'option de marquage EFCI, le champ VCID est écrasé et remplacé par les valeurs '0' ou '1'. (De manière asynchrone, les notifications de cellules perdues dans Merger sont traduites par un incrément des compteurs de cellules perdues par Merger. VCID sert d'index pour ces opérations.)
RISAF in	RISAF met la cellule dans la file d'attente « QID », avec VCID (0 ou 1) attaché à la cellule. Les compteurs de cellules perdues/acceptées indexés par VCID sont mis à jour (ces compteurs ne sont pas exploités). Une notification « ID conn up » est émise vers ICAF up selon « QID ».
RISAF out	Suite à la réception d'un GRPI indiquant la présence d'une cellule présente dans la file d'attente, le processus d'emission est lancé : Le contexte DESCOUT[GRPI] est lu et les paramètres , QID et « j* » mémorisés. La cellule en tête de file « QID » est extraite. Le champ VCID/VPID attaché à la cellule est lu. Le contexte de la connexion pointée par VCID permet de lire une variable d'indirection IP qui est additionnée à « j ». le résultat de l'addition pointe sur la nouvelle en-tête à donner à la cellule (nouveau champ VPI/VCI). La translation de cette nouvelle en-tête est inhibée. Par contre, l'option de marquage EFCI est prise en compte (selon le VCID d'origine 0 ou 1) * j est toujours égal à 0 car le multicast n'est pas géré dans ce mode.
ICAF up	L'ICAF up reçoit les différentes notifications de RISAF in et les introduit dans son échéancier. Le échéancier élit une connexion « ID conn up » et lit le contexte UREQ contenant le champ GRPI lequel est émis sur le bus ATM. ICAF reçoit la cellule et exécute ses traitements selon le contexte de connexion indexé par « ID conn up ».

Tableau A.2 - Trace du transfert d'une cellule dans le cas extension d'adresses VC validée

A.2 Annexes à l'architecture du circuit F2F

A.2.1 Format de la requête descendante DREQ

Name	Bit 7	bit 6	bit 5	Bit 4	bit 3	bit 2	bit 1	bit 0
DREQ0	-	-	-	DEMUX	-	-	-	-
DREQ1 INFO[7 :0]	Connection Identifier LSB (VCID/TPID)							
DREQ2 SERV[7 :0]	Queue Identifier LSB (QID)							
DREQ3 INFO[15 :8]	CLASS [2 :0]			-	Connection Identifier MSB (VCID/TPID)			
DREQ4 SERV[15 :8]	-				Queue Identifier MSB (QID)			
DREQ5 CS[7 :0]	DEST_FID[7 :0]							

Tableau A.3 - Format de la requête descendante DREQ

NB : Les champs grisés sont lus par le bloc DOWN IN et reportés dans le message cellule.

A.2.2 Format du message cellule à la sortie du bloc DOWN IN

Bits	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
BYTE 0	PAD = 00h							
1	SYNC = 5Ah							
2	CELL Val	CLASS[2 :0]			DEMUX	SRC_ID[2 :0]		
3	DEST_FID[7 :0]							
4	Queue Identifier LSB (QID)							
5	0	0	0	0	0	Queue Identifier MSB (QID)		
6	Connection Identifier LSB (VCID / TPID)							
7	0	0	0	0	Connection Identifier MSB (VCID/TPID)			
8-59	CELL 0-51							
60-63	00h							

Tableau A.4 - Format du message cellule à la sortie du bloc DOWN IN

NB : Les champs grisés en foncé sont chargés par le bloc DOWN IN et proviennent de la requête descendante DREQ. Les champs grisés en clair sont chargés par DOWN IN et proviennent de données constantes ou locales (SYNC, SRC ID).

A.2.3 Format du message cellule à la sortie du bloc EXT VC émission

Bits	bit 7	bit 6	Bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
BYTE 0	PAD = 00 h							
1	SYNC 5A h							

2	CELL Val	CLASS[2 :0]			DEMUX	SRC ID
3	DEST_FID					
4	Queue Identifier LSB (QID)					
5	0	0	0	0	0	Queue Identifier MSB (QID)
6	Connection Identifier LSB (VCID)					
7	0	Connection Identifier MSB (VCID)				
8-59	CELL 0-51					
60-63						

Tableau A.5 - Format du message cellule à la sortie du bloc EXT VC emission

NB : Les champs modifiés par EXT VC émission DOWN si DEMUX=1 sont représentés en grisé.

A.2.4 L'algorithme de décodage EXT VC émission (TPID,VPI,VCI)

EXT VC in est activé sur commande de la variable booléenne DEMUX présente dans le message CELL. Si DEMUX = 0, le bloc est transparent. Le décodage d'une connexion VC s'effectue de la manière suivante : L'information TPID (Transport Path Identifier) provenant de FACE (ICAF DOWN: champ INFO[15 :0]) représente le conduit ATM du sous-système SafeCom4000.

1) Première étape : identification du VP

TPID sert alors d'index dans les tables TP_BASE et TP_BOUND. TP_BASE[TPID] pointe à la base d'une zone linéaire CONNECTION_ID où sont rangés les identificateurs de connexion de niveau VP (VPID). Le contenu de CONNECTION_ID[TP_BASE[TPID]+VPI] donne l'identificateur de connexion VPID. La comparaison du champ VPI de la cellule avec TP_BOUND renseigne si la valeur VPI « sort » de la plage VP décodée. Cette plage démarre toujours à VCI=0. Bien entendu des VPI non décodés peuvent se trouver à l'intérieur de la plage. Le champ VPI contient 12 bits (NNI). Le champ VPI[11 :8] est positionné à « 0 » en UNI (VPI sur 8 bits).

2) Deuxième étape : identification du VC

VPID sert d'index dans les tables VP_BASE et VP_BOUND. VP_BASE[VPID] pointe à la base d'une zone linéaire « CONNECTION_ID » où sont rangés les identificateurs de connexion de niveau VC (VCID). Le contenu CONNECTION_ID[VP_BASE[VPID]+VCI] donne l'identificateur final de connexion VCID. La comparaison du champ VCI de la cellule

avec VP_BOUND renseigne si la valeur VCI « sort » de la plage VC décodée. Cette plage démarre toujours à VCI=0. Bien entendu des VCI non décodés peuvent se trouver à l'intérieur de la plage.

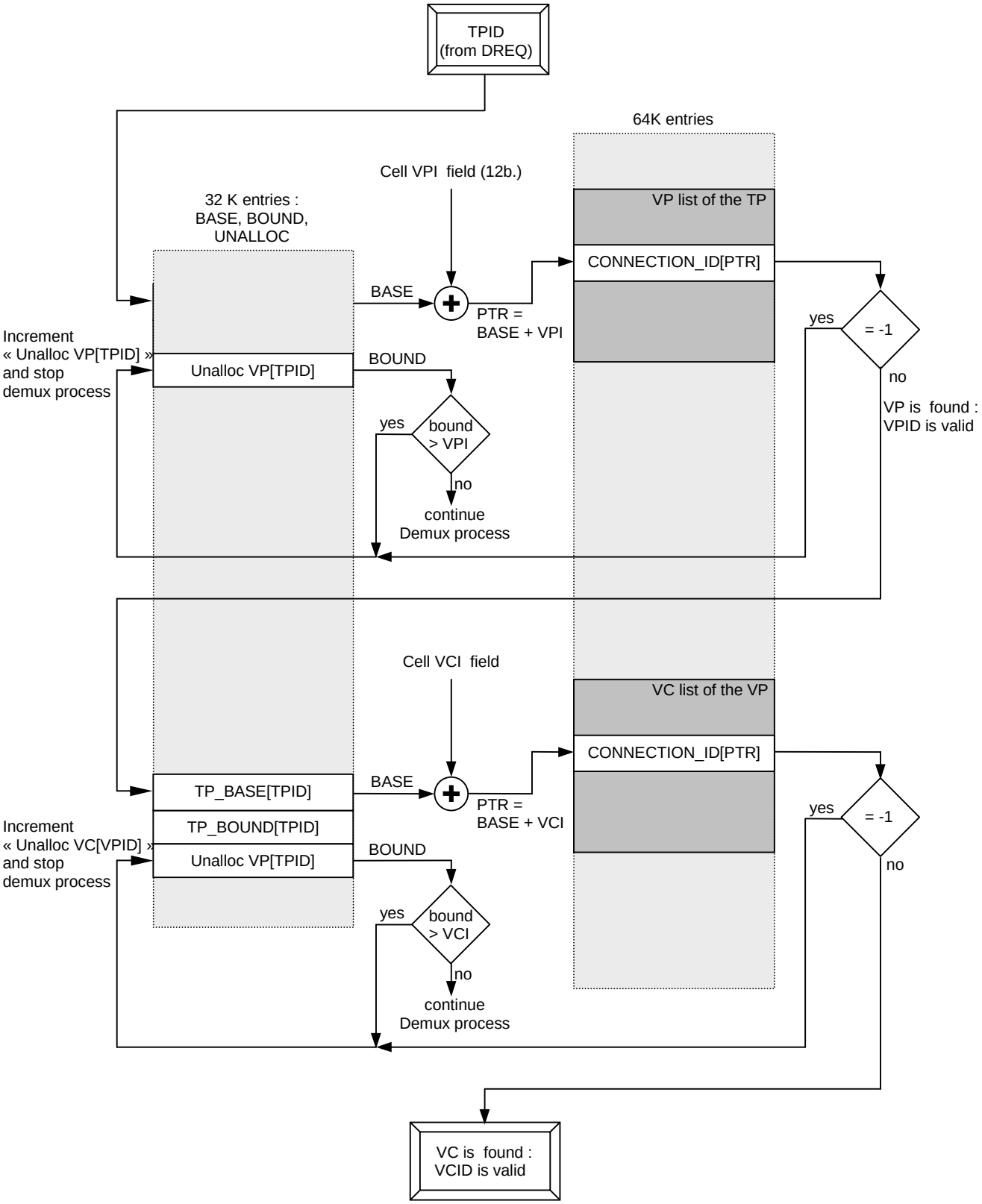

Figure A.5 - Algorithme de décodage EXT VC émission

Pour des raisons d'optimisation de la place mémoire, les structures de niveau TP et VP sont partagées dans une même table et leurs noms de structures deviennent génériques : BASE et BOUND. La table CONNECTION_ID contient les identificateurs VPID et VCID. De plus, la table est divisée en 2 sous-champs CONNECTION_ID_EVEN et CONNECTION_ID_ODD selon que l'index est pair ou impair. 32K de connexions VC peuvent être validées parmi un champ possible de 64K diminué de la place occupée par le niveau VP. La somme des plages VP et VC est de 64K, approximativement, la moitié des entrées connections ID sont occupées dans le pire cas (32K de connexions valides). La difficulté de l'algorithme est la gestion de des différentes zones linéaires stockées dans « CONNECTION_ID ». Les valeurs TPID et VCID, les seules visibles de l'extérieur du bloc ne sont pas affectées par la gestion de la table CONNECTION_ID.

L'algorithme fonctionnel de traitement d'une cellule dans EXT_VC in a besoin de 7 accès à la SSRAM (6 lectures et une écriture) :

```
Début
Si CELL.DEMUX==0 /* test activité du bloc */ Alors
       Fin de traitement.
Finsi.

/* ***************************** recherche du VP identifier : VPID ********************** */
Bound = EXT_VC.BOUND[TPID] /* /* SRAM : 1 READ */
Si CELL.VPI > Bound Alors
       EXT_VC.UNALLOC[TPID]++  /* VPI hors plage,  SRAM : 1 READ/ 1 WRITE */
       CELL.cell_val = 0
       Fin de traitement.
Finsi.
Base = EXT_VC.BASE[TPID] /* lecture RAM externe, mot déjà lu, SRAM : 0 */
PTR = Base + CELL.VPI
VPID = EXT_VC.CONNECTION_ID|PTR]    /* SRAM : 1 READ  */
Si VPID == -1 Alors
       EXT_VC.UNALLOC[TPID]++ /* valeur de VPI non attribuée , mot déjà lu/écrit, SRAM : 0 */
       CELL.cell_val = 0
       Fin de traitement.
Finsi.

/* ************************** recherche du VC identifier : VCID ************************** */
Bound = EXT_VC.BOUND[VPID] /* /* SRAM : 1 READ */
Si CELL.VCI > Bound Alors
       EXT_VC.UNALLOC[VPID]++  /* VCI hors plage, mot déjà lu/écrit SRAM : 0 **/
       CELL.cell_val = 0
       Fin de traitement.
Finsi.
Base = EXT_VC.BASE[VPID] /* lecture RAM externe, mot déjà lu/écrit SRAM : 0 */
PTR = Base + CELL.VCI
VCID = EXT_VC.CONNECTION_ID|PTR]    /* SRAM : 1 READ  */
Si VCID == -1 Alors
       EXT_VC.UNALLOC[VPID]++ /* valeur de VCI non attribuée , mot déjà lu/écrit SRAM : 0 */
       CELL.cell_val = 0
       Fin de traitement.
Finsi.

/* ********************************** Traitement VC  ********************************** */
CELL.cell_val = 1  /* cellule valide */
/* mise à jour des champs du message CELL */
```

```
CELL.CLASS = EXT_VC.CLASS
CELL.DEST_FID = EXT_VC.DEST_FID[VCID] /* SRAM : 1 READ */
CELL.CONNECTION_IDENTIFIER = VCID
CELL.QUEUE_IDENTIFIER = EXT_VC.QUEUE_IDENTIFIER[VCID] /* SRAM : 0 */
/* ************************************************************************** */
Fin de traitement
```

A.2.5 Format du message cellule généré à la sortie du bloc EXT VC réception

Bits	7	6	5	4	3	2	1	0
BYTE 0	PAD							
1	SYNC 5Ah							
2	CELL_val	CLASS[2 :0]			DEMUX	SRC_ID		
3	DEST_FID							
4	Queue Identifier LSB (QID)							
5	0	0	0	0	0	Queue Identifier MSB (QID)		
6	Connection Identifier LSB (VCID) = 0 / 1							
7	0	Connection Identifier MSB (VCID)=0						
8	CELL 0 (VPI MSB)							
9	CELL 1 (VPI LSB)				CELL 1 (VCI MSB)			
10	CELL 2 (VCI)							
11	CELL 3 (VCI LSB)				CELL 3 (PTI+CLP)			
12-59	CELL 4-51							
50-63	0							

Tableau A.6 - Format du message cellule généré à la sortie du bloc EXT VC réception

NB: Les champs modifiés par EXT VC UP (si la translation d'en-tête est effectuée) sont représentés en grisé.

A.2.6 L'algorithme de décodage EXT VC réception

NOTA : Les valeurs basses de « Queue Identifier » et « Connection Identifier » sont à réserver aux connexions ne sollicitant pas les fonctionnalités EPD et translation d'en-tête.

L'algorithme fonctionnel de traitement d'une cellule dans EXT VC out de 7 accès à la SSRAM: 4 lectures et 3 écritures :

```
Début de traitement
VCID = CELL.Connection Identifier
/* ****************************** EPD processing ************************************* */
epd_enable,epd_first,reset_clp = EXT_VC.flags_out[VCID]     /* SRAM : 1 READ */
epd_discard = EXT_VC.epd_discard_state[VCID]     /* SRAM : 0 READ */
Si (discard_epd == 1) ET (CELL.PTI <>  « Last cell») ET (epd_first==0) ET (CELL.PTI ==user
cell) ET (epd_enable==1) Alors
/* premiere et derniere cellules de trame AAL5 jamais détruites, OAM & RM transparents */
```

```
        EXT_VC.Lost_CELL[VCID] ++  /* comptage cellules perdues, SRAM : 0 */
        Cell_process = 0
Sinon
        Cell_process = 1
Finsi
Si CELL.PTI = « Last cell » ET (CELL.PTI ==user cell) Alors
        epd_first_next = 1   /* la prochaine cellule sera la première d'une trame */
        discard_epd == 0   /* reset écartement des cellules par epd **** */
Sinon
        epd_first = 0   /* reset discard by EPD */
finsi
EXT_VC.flags_out[VCID]     = ,epd_enable,epd_first_next,reset_clp   /* epd_first_next écrit en
epd_first,  SRAM : 1  WRITE */
EXT_VC.epd_discard_state[VCID]     = discard_epd   /* SRAM : 0  WRITE */
Si Cell_process == 1 Alors
        /* ***************************** emission de cellule ***************************** */
        Emettre la cellule vers RISAF.  Récupérer l'information DBP.
        /* ***************************** positionnement temoin epd discard ***************** */
        Si epd_first == 1  /* test de premiere cellule */ Alors
                Si (DBP == « seuil EPD dépassé ('110' or '111') ») Alors
                        discard_epd == 1
                sinon
                        discard_epd == 0
                finsi
        finsi
        EXT_VC.epd_discard_state[VCID] = discard_epd   /* SRAM : 1  WRITE */
finsi

/ ***************************  comptage et translation cellule ************************** */
Si Cell_process == 1 Alors
        Si DBP == « cellule détruite par saturation » Alors     /* SRAM : 1 READ/ 1 WRITE */
                EXT_VC.Lost_CELL[VCID] ++  /* comptage cellules perdues */
        Sinon
                EXT_VC.Accepted_CELL[VCID] ++  /* comptage cellules acceptées */
        Finsi
        vp,vp12,vc, tvp, tvc, efci_option = EXT_VC.NewVPVC[VCID].   /* SRAM :  1 READ */
        Si tvp <> 0 ou  tvc <>  0  /* test si translation locale */ Alors
                /* destruction  de l'identicateur de connexion pour RISAF*/
                Si efci_option == 0 Alors
                        CELL.Connection Identifier = 0  /* entrée « non translatante » avec
                                                          efci_option=0  de RISAF */
                sinon
                        CELL.Connection Identifier = 1   /* entrée « non translatante » avec
                                                           efci_option =1  de RISAF */

                finsi
        finsi
        Si tvp == 1 /* translation VP */ alors
                Si vp12 == 1   /* VPI 12 bits */ alors
                        CELL.CELL.VPI[11 :0] = vp
                sinon
                        CELL.CELL.VPI[7 :0] = vp
                Finsi
        Finsi
        Si tvc == 1 /* translation VC */ Alors
                CELL.CELL.VCI[15 :0] = vc
        Finsi
        Si  reset_clp=1  /* reset CLP */
        Alors
                CELL.CELL.CLP = 0
        Finsi
finsi
Fin de traitement
```

L'algorithme fonctionnel de traitement des notifications de pertes de cellules à cause de la concentration dans le Merger : 14 accès à la SSRAM (7 lectures et 7 écritures) :

```
Début de traitement
Pour i allant de 0 à 6 /* scrutation des canaux HSL */
        Si notification IDVC[i] de perte cellule sur la ligne MergerLostCell[i] Alors
                EXT_VC.MergerLostCell [IDVC[i]] ++    /* SRAM : 1 READ / 1  WRITE */
        Finsi
```

```
Fin boucle
Fin de traitement
```

A.2.7 Structure de la mémoire EXT VC

La mémoire EXT VC est remplie d'une table de structure de base indexée par une adresse de base de 15 bits ce qui représente une variable sur un domaine d'entrées de 0 à 32K. La structure de base comporte 16 mots de 32 bits (32Kx16x32= 16 Mb). L'index (entre crochets [ ]) est multiple en fonction des champs adressés.

$WORD_{32}$	D31:24	D23:16	D15:8	D7:0
0	BOUND [TPID or VPID]		BASE[TPID or VPID]	
1	UNALLOC(TPID or VPID]			
2	CONNECTION_ID_odd [PTR]		CONNECTION_ID_even [PTR]	
3	0 CLASS[VCID]	DEST_FID[VCID]	Queue Identifier [VCID]	
4	flags in[VCID]		Discard_state[VCID]	flags out [VCID]
5	New VPVC [VCID]			
6	Accepted CELL [VCID]			
7	F2F Lost CELL [VCID]			
8	Merger Lost CELL[VCID]			
9	Reserved			
10	Reserved			
11	Reserved			
12	Reserved			
13	Reserved			
14	Reserved			
15	Reserved			

Tableau A.7 - Structure de base de la mémoire EXT VC

NB : En gris clair sont représentées, les structures de données manipulées par EXT VC émission et en gris foncé, les structures de données manipulées aujourd'hui par EXT VC réception.

Définition des champs des variables de la mémoire EXT VC :

W0	bits	bit num	Function	Reset
BOUND	16	31 :16	Bound of the VPI or VCI field Indexed by TPID or VPID	0
BASE	16	15 :0	Offset of the CONNECTION_ID field Indexed by TPID or VPID	0

W1	bits	bit num	Function	Reset
UNALLOC	32	31 :0	Unrecognized cells Indexed by TPID (unkown VPI) or VPID (unkown VCI)	0

W2	bits	bit num	Function	Reset
CONNECTION_ID	16	31 :16	Contains new base pointer of VP or VCID Indexed by odd ptr	Not valid

_odd				
CONNECTION_ID _even	16	15 :0	Contains new base pointer of VP or VCID Indexed by even ptr	not valid

W3	bits	bit num	Function	Reset
Reserved	5	31 :27	-	
CLASS	3	26 :24	Class of the connexion (VC)Indexed by VCID	0
DEST_FID	8	23 :16	Destination cluster field Indexed by VCID	0
Queue_Identifier (QI)	16	15 :0	Destination queue identifierIndexed by VCID	0

W4	bits	bit num	Function	Reset
Flags_in	5	31 :16	Options dedicated to VC_EXT in Indexed by VCID	0
Reserved	7	15 :9	Reserved	0
Discard_state	1	8	Discard state by EPD Indexed by VCID /VPID	0
Reserved	5	7 :3	Reserved	
Reset_CLP	6	7 :2	Reset the CLP bit when set to 1 (useful for ABR VS/VD)	0
Epd_enable	1	1	EPD function enable (configuration: set by CPU) Indexed by VCID/VPID	0
Epd_first	1	0	Epd « first cell of frame » flag (set by F2F) Indexed by VCID / VPID	0

W5 : NewVPVC	bits	bit num	Function Indexed by VCID /VPID	Reset
EF	1	31	If set, F2F can modify cell's EFCI bit	0
TVP	1	30	VPI field translation	0
TVP12	1	29	VPI field is 12 bit wide (else 8 bit wide)	0
TVC	1	28	VCI field translation	0
VP	12	27 :16	New VPI field	0
VC	16	15 :0	New VPI field	0

W6	bits	bit num	Function	Reset
Accepted CELL	32	31 :0	Accepted CELLs by RISAF Indexed by VCID / VPID	0

W7	bits	bit num	Function	Reset
F2F Lost CELL	32	31 :0	Rejected CELLs by RISAF Indexed by VCID/ VPID	0

W8	bits	bit num	Function	Reset
Merger Lost CELL	32	31 :0	Rejected CELLs by merger Indexed by VCID/VPID	0

W15 :9	bits	bit num	Function	Reset
Reserved	32	31 :0	Reserved	0

Tableau A.8 - Définition des champs des variables de la mémoire EXT VC

A.2.8 Format de la trame HSL

Cas d'un message cellule valide et DBP valide

Bits	7	6	5	4	3	2	1	0
BYTE 0	PAD = 00h							
1	SYNC = 5Ah							
2	cell_val=1	CLASS			DEMUX	SRC ID		
3	DEST_FID (Bit field)							
4	Queue Identifier (QID LSB)							
5	Queue Identifier (QID MSB)							
6	Connection Identifier (VCID LSB)							

7	Connection Identifier (VCID MSB)			
8-59	CELL 0-51			
60	dbp_val=1	DBP	0	SRC ID
61	Queue Identifier (QID LSB)			
62	DEST_ID	Queue Identifier (QID MSB)		
63	/BIP 8			

Tableau A.9 - Format d'une trame HSL avec un message cellule validé et un message DBP validé

NB : Les champs modifiés par le bloc DOWN OUT sont représentés en grisé.

Cas d'un message avec cellule invalide et DBP invalide

Bits	7	6	5	4	3	2	1	0
BYTE 0	PAD							
1	SYNC 5Ah							
2	cell_val=0	CLASS=0			DEMUX=0	SRC ID=0		
3	DEST_FID (Bit field)=0							
4	Queue Identifier (QID LSB)=0							
5	Queue Identifier (QID MSB)=0							
6	Connection Identifier (VCID LSB)=0							
7	Connection Identifier (VCID MSB)=0							
8-59	CELL 0-51=0							
60	dbp_val=0	DBP=0			0	SRC ID=0		
61	Queue Identifier (QID LSB)=0							
62	DEST_ID=0				Queue Identifier (QID MSB)=0			
63	/BIP 8 (NOT 5Ah = A5h)							

Tableau A.10 - Format d'une trame HSL avec un message cellule invalidé et un message DBP invalidé

NB :

1) Un message CELL peut être valide tandis que DBP est invalide et vice-versa.

2) Les champs modifiés par DOWN OUT sont représentés en grisé et les champs modifiés par les autres blocs sont représentés en grisé foncé.

Le premier octet de la trame nommé PAD correspond à un code de repos qui permet de régler les problèmes de plésiochronisme entre les horloges source et destination dans le cas d'utilisation de lien série HSL. Il n'a pas de signification et peut être non transmis. Le deuxième caractère SYNC est un code de synchronisation permettant au module de réception de « cadrer » la trame. Chacun des messages est validé par les bits de validation CELL VAL et DPB VAL. Si un de bits est inhibé cela signifie que le message correspondant (CELL ou DBP) n'est pas valide auquel cas le message est rempli de '0' sur la totalité de ses champs afin de ne pas contrarier la synchronisation sur SYNC. Le dernier caractère BIP8 est un code

de contrôle qui permet de vérifier la validité de la trame et le bon fonctionnement du médium de commutation. Le codage BIP8 porte sur la totalité de la trame sans le champ PAD.

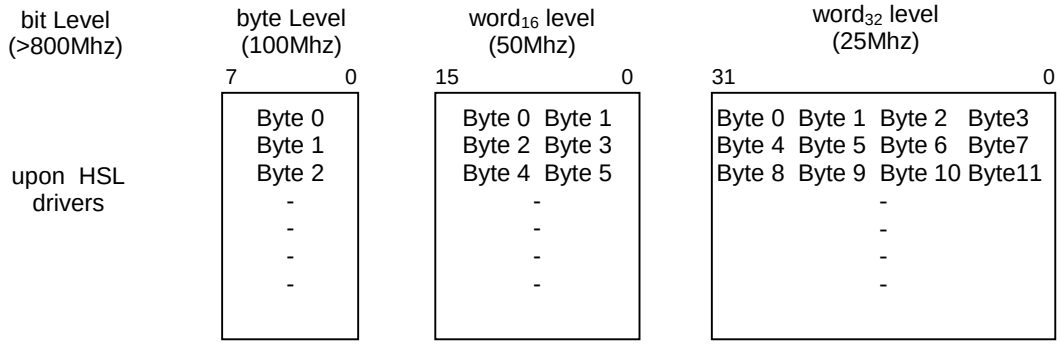

Figure A.6 - Format de transmission

A.2.9 Le service de redondance HSL

Les circuits comme le VSC7216 de la société VITESSE présente la possibilité d'émettre et de recevoir de trames sur deux liens différents pour une même unité parallèle/série. Deux sous-systèmes clusters peuvent alors être connectés comme représentés dans la figure A.7.

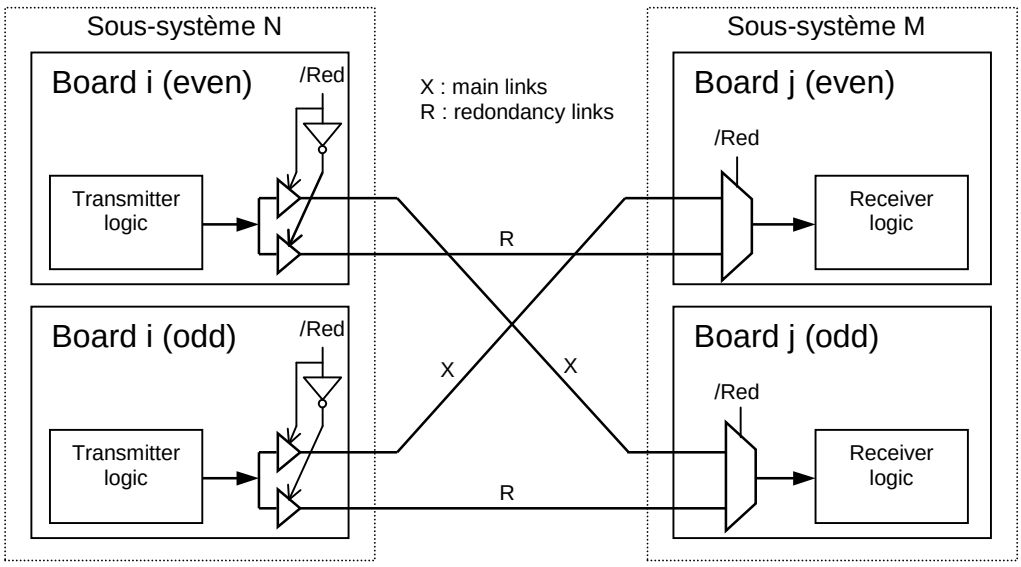

Figure A.7 - Service de redondance

La notion de lien principal et de lien de redondance n'est pas importante: Toute carte *i* d'un sous-système *N* peut maintenant être reliée à une autre carte soit paire *j* soit impaire *j+1* d'un autre sous-système *M*. Deux clusters peuvent être symboliquement connectés selon 2 modes : soit connexion directe, soit connexion croisée. A noter que le mode choisi concerne deux clusters (M, N) pris deux à deux, le mode choisi sur les clusters (M, K) est indépendant.

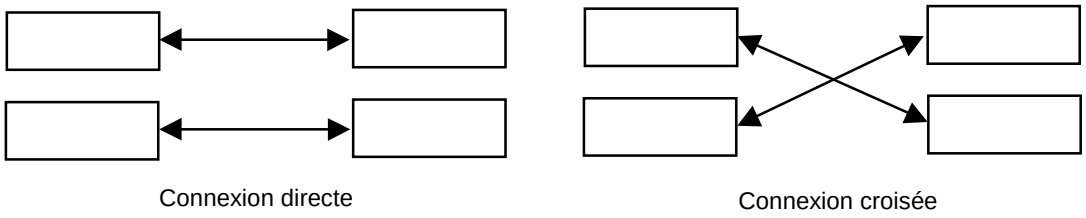

Figure A.8 - Modes de connexion directe et croisée

La capacité de pouvoir croiser les liens permet d'obtenir toutes les possibilités d'interconnexions et de pouvoir masquer à l'utilisateur l'architecture interne du SafeCom4000X.

A.2.10 Interfaces du circuit F2F

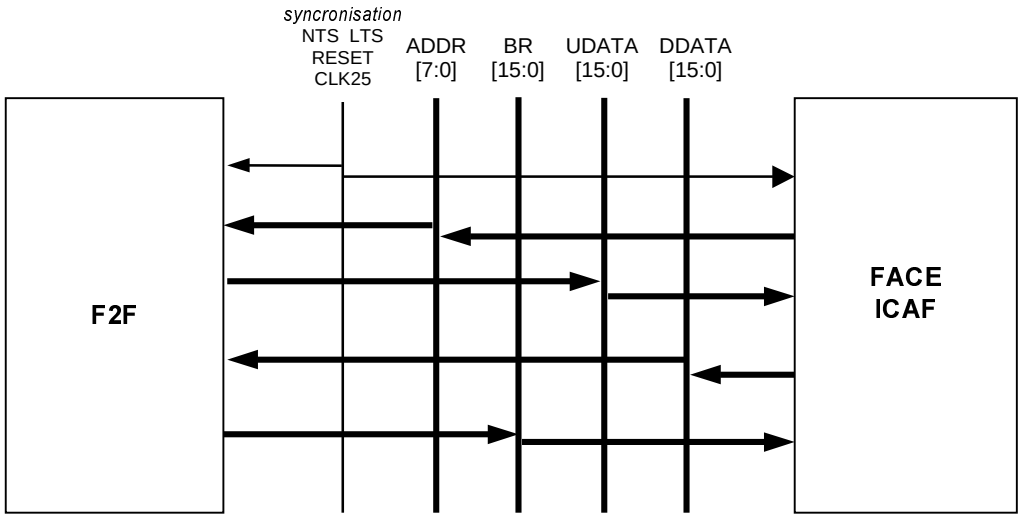

Figure A.9 - Interface avec le bus ATM

Cycles:	b15	b0	b1	b2	B3	b4	b5	b6	b7	b8	b9	b10	b11	b12	b13	b14
NTS	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
LTS	0	0/1	0	0	0	0	0	0	0	X	0	0	0	X	0	0
ADDR [0:7]	UREQ 1 TSn	UREQ 2 TSn	DREQ 0 TSn	DREQ 1 TSn	DREQ 2 TSn	DREQ 3 TSn	DREQ 4 TSn	0	0	BG TSn-2	0	0	DREQ 5 TSn	DREQ 6 TSn	0	0
BR [0:15]	UCOB 0 TSn-1	UCOB 1 TSn-1	UCOB 2 TSn-1	NOTIF tsn:19 tsn+1:9 (async)	NOTIF tsn:18 tsn+1:8 (async)	NOTIF Tsn:17 Tsn+1: 7 (async)	NOTIF Tsn:16 Tsn+1: 6 (async)	NOTIF Tsn:15 tsn+1:5 (async)	NOTIF tsn:14 tsn+1:4 (async)	NOTIF tsn:13 tsn+1:3 (async)	NOTIF tsn:12 tsn+1:2 (async)	NOTIF tsn:11 tsn+1:1 (async)	NOTIF tsn:10 tsn+1:0 (async)	DBP2 TSn-1	DBP1 TSn-1	DBP0 TSn-1
UDATA [0:31]	D8 TSn-4	D9 TSn-4	D10 TSn-4	D11 TSn-4	D12 TSn-4	D13 TSn-4	0	0	D0 TSn-3	D1 TSn-3	D2 TSn-3	D3 TSn-3	D4 TSn-3	D5 TSn-3	D6 TSn-3	D7 TSn-3
DDATA [0:31]	D12 TSn-3	D13 TSn-3	0	0	D0 TSn-2	D1 TSn-2	D2 TSn-2	D3 TSn-2	D4 TSn-2	D5 TSn-2	D6 TSn-2	D7 TSn-2	D8 TSn-2	D9 TSn-2	D10 TSn-2	D11 TSn-2

Tableau A.11 - Assignation de cycles pour les signaux du bus ATM

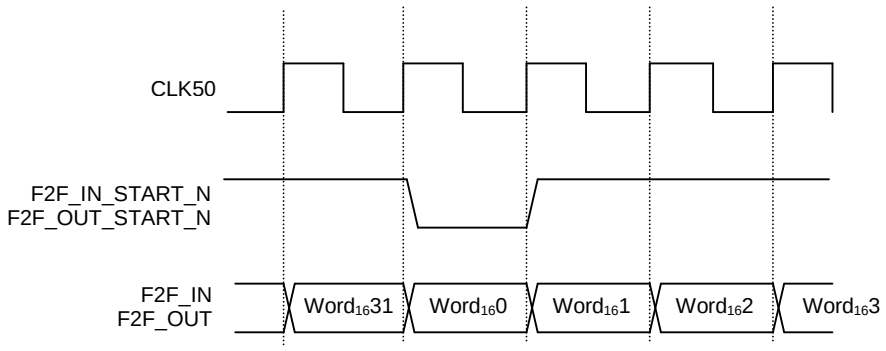

Figure A.10 - Interface HSL

F2F_xx_START_N	$Word_{16}$	F2F_xx [15 :8]	F2F_xx[7 :0]
0	0	PAD	SYNC
1	1	Byte 2	Byte 3
1	2	Byte 4	Byte 5
1	I	Byte 2*i	Byte 2*i+1
1	31	Byte 62	BIP8

Tableau A.12 - Synchronisation de la trame HSL

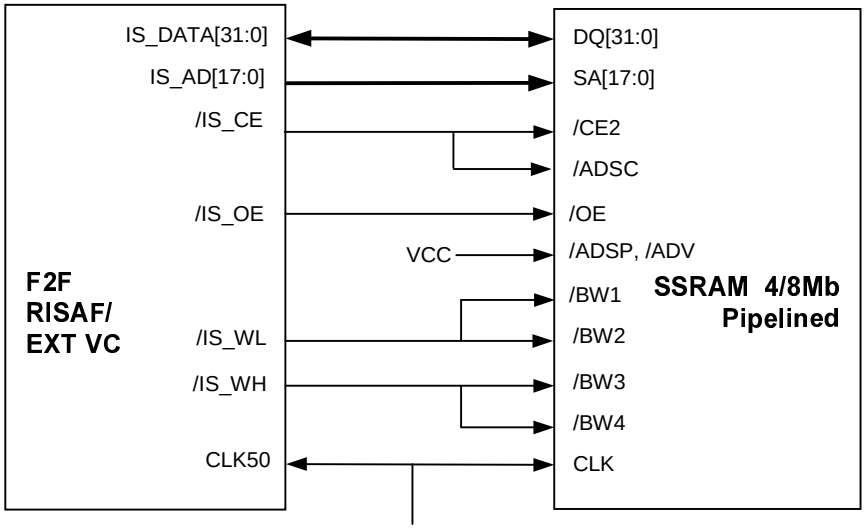

Figure A.11 - Interface SSRAM (les mémoires de contexte RISAF et EXT VC sont identiques)

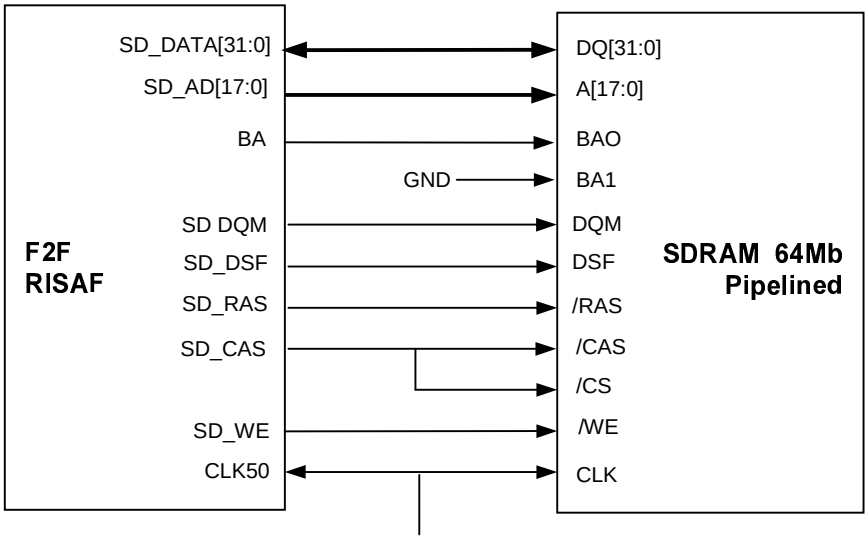

Figure A.12 - Interface SDRAM, mémoire de cellules RISAF

..... IDLE	IDLE	START =0	VCID[0]	VCID[1]	……	VCID[15]	STOP =1	IDLE	IDLE .....

Figure A.13 - Interface Merger Lost Cell

.... IDLE	START =0	DBP[0]	DBP[1]	DBP[2]	QID[0.....10]	DESTID [0]	DESTID [1]	DESTID [2]	STOP =1	IDLE ....

Figure A.14 - Interface DBP

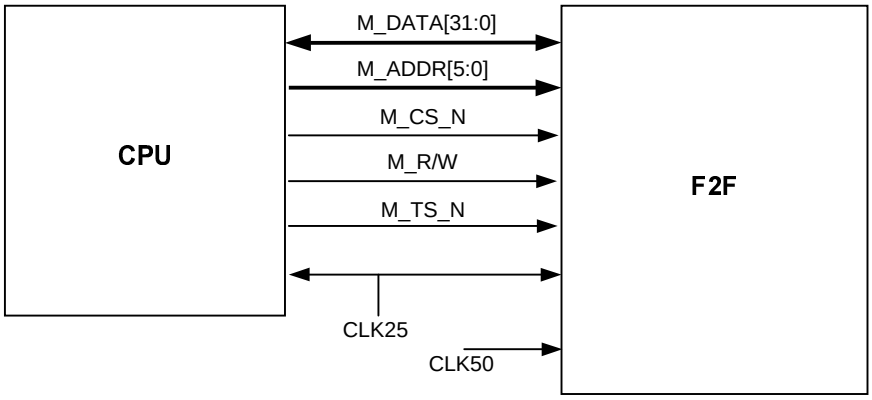

Figure A.15 - Interface CPU

Signal	Type	Largeur	Niveau RESET	Fonction	Type (default LV-TL)
Signaux de synchronisation					
CLK25	in	1	-	Clock system 25Mhz (Enable)	
CLK50	in	1	-	Clock system 50Mhz	
RESET_N	in	1	-	Reset module, active low	
TOTAL		3			
Signaux de l'interface ATM bus					
NTS	in	1	-	New Time Slot	
LTS	in	1	-	Last Time Slot	
BR	out	16	Hi-Z	Bus Request Input	PCI
UDATA	out	32	Hi-Z	Up Data Input	PCI
DDATA	in	32	-	Down Data Input	PCI
ADDR	in	8	-	Address Input	PCI
TOTAL		90			
SSRAM RAM RISAF					
IS_DATA	I/O	32	0	SSRAM Data	
IS_AD	out	18	0	SSRAM Address(IS_AD17 for future use)	
IS_CE_N	out	1	1	SSRAM Chip Enable	
IS_WL_N	out	1	1	SSRAM Write Enable Low	
IS_WH_N	out	1	1	SSRAM Write Enable high	
IS_OE_N	out	1	1	SSRAM output Enable	
TOTAL		54			
Signaux de l'interface SDRAM					
SD_DATA	I/O	32	0	SDRAM Data Out	
SD_AD	out	10	0	SDRAM Address	
SD_BA	out	1	1	SDRAM Bank Select	
SD_WE_N	out	1	1	SDRAM Read Write	
SD_RAS_N	out	1	1	SDRAM Row Address Select	
SD_CAS_N	out	1	1	SDRAM Col. Address Select	
SD_DSF	out	1	1	SDRAM Fonction Speciale	

SD_DQM	out	1	1	SDRAM Fonction Speciale	
SD_CS_N	out	1	1	SDRAM Chip Select	
SD_CKE	out	1	1	SDRAM Clock Enable	
TOTAL		50			
Signaux Merger					
F2F_IN	in	16	-	RX Data	
F2F_IN_START_N	in	1	-	RX Data frame start	
F2F_OUT	out	16	0	TX Data	
F2F_OUT_START_N	out	1	1	TX Data frame start	
MergerLostcell	in	8	-	Merger Lost Cell notifications from Merger	
DBP	in	8	-	Down Backpressure message from Merger	
TOTAL		50			
Signaux de l'interface CPU					
M_DATA	I/O	32	Hi-Z	Data to/from CPU	
M_ADDR	in	6	-	Address from CPU	
M_CS_N	in	1	-	Chip Select (can be common with Merger)	
M_R/W	in	1	-	Read if 1 / Write if 0	
M_TS_N	in	1	-	Transfer Start if 0	
TOTAL		41			
SRAM ZBT EXT_VC					
X_DATA	I/O	32	0	SSRAM Data	
X_AD	out	19	0	SSRAM Address	
X_CE_N	out	1	1	SSRAM Chip Enable	
X_WR_N	out	1	1	SSRAM Write Enable	
X_BW0_N	out	1	1	SSRAM Write byte Enable (D7 :0)	
X_BW1_N	out	1	1	SSRAM Write Byte Enable (D15 :8)	
X_BW2_N	out	1	1	SSRAM Write Byte Enable (D24 :16)	
X_BW3_N	out	1	1	SSRAM Write Byte Enable (D31 :24)	
X_OE_N	out	1	1	SSRAM output Enable	
TOTAL		58			
TOTAL I/O		346			

Tableau A.13 - Résumé des entrées et sorties du circuit F2F

A.3 Annexes du modèle du simulateur

A.3.1 Structures de données

Pour être accédées facilement par les fonctions d'ordonnancement, les FIFO sont organisées en trois tableaux: un tableau tridimensionnel pour les FIFO d'entrée, un tableau bidimensionnel pour les FIFO de la matrice et un tableau tridimensionnel pour les FIFO de sortie. Les indices d'une FIFO sont:

- i: indice de port d'entrée (valeurs : de 0 à N-1) ;
- j: indice correspondant au port de sortie (valeurs : de 0 à N-1) ;
- m: indice de flux (valeurs de 0 à M-1) (pour les FIFO d'entrée et de sortie uniquement).

Les dimensions des files d'attente de la matrice et des modules de sortie sont stockées dans deux tableaux l_mx_{ijm} et l_out_{ijm}. Le nombre de paquets des files d'attente de la matrice et des modules de sortie sont stockés dans deux tableaux n_mx_{ijm} et n_out_{ijm}. Un tableau c_out_{ijm} contient le nombre de caractères extraits pour chaque flux F_{ijm} depuis le début de la simulation et un tableau t_out_{ijm} contient le nombre de paquets extraits pour chaque flux F_{ijm} depuis le début de la simulation.

A.3.2 Le fichier de paramètres simfi.cfg

L'initialisation commence par la lecture d'un fichier de configuration contenant des informations concernant la dimension du commutateur, les caractéristiques du trafic, la durée de la simulation, etc. La syntaxe du fichier de configuration :

Caractéristiques de la simulation :

```
# Date de fin de simulation
EOS               = 5000000;
# Préremplissage des FIFO de sortie :
prefill_mode      = 1;
```

Dimensions du réseau :

```
# Nombre de ports :
dms_N             = 8;
# Nombre de flux par port :
nbr_max_M         = 10;
# Taille maximale des FIFO sortie :
dms_max_fifo_out  = 1024;
# Taille maximale des FIFO matrice :
dms_max_fifo_mx   = 64;
# Diviseurs d'horloge
out_ck            = 1;
in_ck             = 1;
mx_ck             = 1;
```

Caractéristiques du trafic :

```
# Taille des paquets constante :
pkt_size          = 60;
# Taille de paquets variable :
#pkt_max_size      = 128;
#pkt_min_size      = 4;
# pourcentage de flux existants :
flux0_prc         = 5;
```

```
# rapport entre le débit maximum et minimum :
Dmaxmin           = 8;
# Longueur maximale d'une rafale :
max_brst_leng     = 1;
# charge globale de la matrice:
Charge            = 0.80;
```

Paramètres de l'architecture:

```
# Nombre de priorités :
prio_range        = 4;
```

Après le nombre de cycles définit par EOS, un évènement fin_de_simulation provoque la génération du fichier de résultats et l'arrêt de la simulation.
Les paramètres liés à la dimension du réseau, nombre de ports et de sous flux, permettent d'initialiser la matrice et les files d'attentes.
Le paramètre prefill_mode permet de remplir aléatoirement les FIFO de sortie avec des paquets possédant une date d'entrée inférieure à 0. L'utilisation de cette option permet d'éviter d'avoir des FIFO vides en début de simulation ce qui peut dégrader les résultats.

Les caractéristiques du trafic établies par les fonctions utilisateur. A partir de flux0_prc, une fonction utilisateur init_flux() génère le tableau tridimensionnel $flux0_{ijm}$ qui a comme éléments les valeurs binaires 1 pour les flux qui existent et 0 pour ceux non définis. Un fichier init_sim.logF est généré. A partir de Dmaxmin, et dms_max_brst une fonction utilisateur init_sim() génère deux tableaux tridimensionnels : D_{ijm} qui a comme éléments les contrats de débits et N_{ijm} qui contient la taille maximale des rafales pour chaque flux. Deux fichiers init_sim.logD et init_sim.logN sont générés. La taille des paquets peut être constante ou variable. Si pkt_max_size et pkt_min_size sont mis en commentaire avec un # une taille constante définie par pkt_size est prise en compte. Tout ces paramètres sont accessibles aux fonctions utilisateur par l'intermédiaire du fichier datastruct.h.

A.3.4 L'interface entre le simulateur et les fonctions utilisateur

Pour pouvoir être appelées par le simulateur, les fonctions utilisateur (génération de résultats) doivent respecter une certaine interface.

La syntaxe de la fonction d'ordonnancement d'entrée de cette fonction est la suivante: int schd_in (int i, int *j, int *m). Cette fonction peut lire grâce au paramètre i l'indice du port d'entrée. Elle doit fournir comme résultat deux entiers j et m représentant les indices du flux à sélectionner, j étant l'indice du port destination et m l'indice du sous-flux. La fonction doit renvoyer 0 si aucune erreur ne se produit pendant la sélection, un entier différent de 0 sinon.

La fonction d'ordonnancement matrice définit le comportement des ordonnanceurs de la matrice. La syntaxe de cette fonction est la suivante: int schd_mx (int j, int *i). L'indice j représente le port de sortie et le résultat i doit indiquer le numéro du port d'entrée sélectionné. Cette fonction est appelée seulement s'il existe des FIFO non vides. Une condition supplémentaire imposée : elle doit pointer toujours vers une FIFO non vide. La fonction doit renvoyer 0 si aucune erreur ne se produit pendant la sélection, un entier différent de 0 sinon. Telle quelle est définie elle fonctionne selon un algorithme *round-robin* pour donner une priorité égale à tout port d'entrée.

La fonction d'ordonnancement sortie définit le comportement des ordonnanceurs des ports de sortie. La syntaxe de cette fonction est la suivante: int schd_in (int j, int *i, int *m). j identifie le port de sortie correspondant. Les résultats i et m doivent représenter les indices du flux à sélectionner, i étant l'indice du port d'entrée et m l'indice du sous-flux. La fonction doit envoyer toujours des indices représentant une FIFO non vide. La fonction doit renvoyer 0 si aucune erreur ne se produit pendant la sélection, un entier différent de 0 sinon.

A.3.5 Description du fichier datastruct.h

Les fonctions ont accès à un fichier datastruct.h qui contient les structures de données communes. Ces variables communes permettent aux fonction de connaître quels sont les flux définis et à schd_out d'assurer les gestion des débits programmés et des rafales. Par l'intermédiaire de datastruct.h le simulateur met à la disposition de l'utilisateur des données qui lui permettent de générer les résultats de la simulation.
Les fonctions ont également accès à des structures privées qui sont définies par l'utilisateur. Ces structures privées permettent la gestion de l'historique, du *round-robin*.

```
extern int date;
```

Le paramètre permet l'accès à la date courante de la simulation.

Dans le fichier datastruct.h se retrouvent une série de paramètres provenant du simfi.cfg :

```
/* End Of Simulation */
extern int EOS;
/* Number of Input/Output ports: */
extern int dms_N;
/* Maximum number of fluxes = dms_N squared: */
extern int dms_N2;
/* Maximum number of sous-fluxes in a flux: */
extern int nbr_max_M;
/* priority range: */
extern int prio_range;
/* Maximumc matrix FIFO length: */
extern int dms_max_fifo_mx;
/* Maximumc output FIFO length: */
extern int dms_max_fifo_out;
/* Maximum burst length: */
extern int max_brst_leng;
/* Pourcentage flux nuls */
extern int flux0_prc;
/* Maximum port throughput */
extern int Dmax_port;
/* Maximum/minimum flux throughput: */
extern int Dmaxmin;
# charge globale de la matrice:
extern int Charge            = 0.80;


/* clock factor */
extern float out_ck;
extern float mx_ck;
extern float in_ck;

/* def. size for packts */
extern int pkt_size;
extern int pkt_mode;
extern int pkt_min_size;
extern int pkt_max_size;

extern int prefill;

/* contrat D / maximum D pourcentage : */
extern float D_Dmax_p;
```

D_Dmax_p est un paramètre produit par init_sim qui indique le rapport entre le débit programmé et le débit maximal qui peut être définit. D_Dmax_p doit avoir une valeur proche de la valeur du paramètre Charge.

Les fonctions suivantes permettent aux ordonnanceurs d'avoir accès à la taille en nombre de caractères ou en nombre de paquets des différentes FIFO.

La fonction extern int fifo_mx_size_chars (int i, int j) renvoie la taille en mots d'une FIFO de la matrice et la fonction extern int fifo_mx_size_frames (int i, int j) la taille en nombre de paquets. i représente l'indice du port de sortie et j l'indice du port d'entrée.

La fonction extern int fifo_out_size_chars (int i, int j, int m) renvoie la taille en mots d'une FIFO de sortie et la fonction extern int fifo_out_size_frames (int i, int j, int m) renvoie la taille en nombre de paquets d'une FIFO de sortie. i représente l'indice du port de sortie, j l'indice du port d'entrée.

Les fonctions init_flux et init_sim permettent à l'utilisateur de décider du processus d'initialisation. Ce fonctions génèrent les tableaux des flux, des débits et des intensités de rafales.

```
extern int flux0_read (int i, int j, int m);
/* flux ijm throughput: */
extern int Dijm_read (int i, int j, int m);
/* burst caracteristsics for ijm flux: */
extern int Nijm_read (int i, int j, int m);
```

Ces fonctions permettent d'avoir accès aux éléments des tableaux des flux, des débits et des intensités de rafales.

```
/* duration matrix FIFO contenetion: */
extern int t_wait_mx_read (int i, int j);
/* number of matrix FIFO contenetion: */
extern int n_wait_mx_read (int i, int j);
/* duration output FIFO contenetion: */
extern int t_wait_out_read (int i, int j, int m);
/* number of output FIFO contenetion: */
extern int n_wait_out_read (int i, int j, int m);

/* medium transit delay: */
extern int t_med_read (int i, int j, int m);
/* maximum transit delay: */
extern int t_max_read (int i, int j, int m);
```

Ces fonctions permettent à la fonction utilisateur instr() d'extraire une série de paramètres statistiques nécessaires pour l'évaluation des performances de la simulation.

Bibliographie

[1] Chen Thomas M., Liu Stephen, "ATM Switching System", *Artech House, Incorported*, Boston, (1995), pp. 81-233.

[2] M. Devault, J. Cochennec and M.Servel, "The Prelude ATM experiment: assessments and future prospects", *IEEE J. Selected Areas Communications*, (1988), pp. 1527-1537.

[3] H. Kuwahara, N. Endo, M. Ogino and T. Kozaki, "Shared buffer memory switch for ATM exchange", *Proceedings of International Conference on Communications*, Boston, (1989), pp. 4.4.1-4.4.5.

[4] Onvural, Raif O., "Asynchronus transfer mode networks : performance issues", *Artech House Incorporated*, Boston, (1994), pp. 207-252.

[5] Gauthier Jean Pierre, "The SC4000's management of ATM Trafic", *CS Telecom white papers*, http://www.cstelecom.com, (1999).

[6] H.S. Heh, M.G. Hluchyj and A. S. Acampora, "The Knockout switch: a simple, modular architecture for high-performance packet switching", *IEEE Selected Areas Communications*, SAC-5 18 (1987), pp. 1274-1283.

[7] Widjaja Indra, Leon-Garcia Alberto, "Helical Switch : A multipath ATM switch which preserve cell sequence", *IEEE Transactions & Communications* v42 N 8, (1994), pp. 2618-2629.

[8] Robertazzi, Thomas G., "Performance evaluation of high speed switching fabrics and networks : ATM, broadband ISDN, and MAN technology", *New York IEEE Press*, (1996).

[9] Gauthier Jean Pierre, "FACE, Full ATM Core Engine", *CS Telecom white papers*, http://www.cstelecom.com, (1999).

[10] Stoica Alexandru, Greiner Alain, Gauthier Jean-Pierre, "Système de commutation de données, notamment de données de type ATM", *Demande de brevet INPI No 00/15071* (2000).

[11] Stoica Alexandru, Greiner Alain, Gauthier Jean-Pierre, "A FPGA solution for ATM switching architecture " *Proceedings of International Conference DCIS 2001*, Porto, Portugal (2001).

[12] Stoica Alexandru, Gauthier Jean-Pierre, "ATM Core 1 SC4000X DS ATM operation F2F functional specification" *CS Telecom internal document*, (2000).

[13] Stoica Alexandru, Gauthier Jean-Pierre, "ATM Core 1 SC4000X SSC ATM operation F2F FPGA *CS Telecom internal document*, (2000).

[14] "Virtex™ Field Programmable Gate Arrays", *Xilinx Virtex Series FPGA*, http://www.xilinx.com, (2000).

[15] Fayet Caroline, "Architectures de Commutateurs ATM", *Habilitation à diriger des recherches*, Université de Versailles – St. Quentin en Yvelines (1997).

[16] F. A. Tobagi and C. Kwok, "Fast packet switching architectures for broadband intergrated services digital networks", *Proceedings IEEE 78*, (1990), pp. 133-167.

[17] H. Ahmadi and W.E. Denzel, "A survey of modern high performance switching techniques", *IEEE Selected Areas Communications*, 7 (7) (1989) 1091-1103.

[18] Black, Uyses D., "ATM : Foundation for Broadband Networks", *Prentice Hall, Incorporated*, (1995), pp. 181-202.

[19] Tubtiang Arnon "Un commutateur ATM pour le réseau numérique à intégration de services large bande", *thèse de doctorat de l'Université Paris VI*, (1993).

[20] Reibaldi Vincent "Conception et réalisation d'un routeur de paquets a haute performances", *thèse de doctorat de l'Université Paris VI*, (1997).

[21] A. Engel, B. Mobasser, "New standards in relation to the Broadband ISDN", *Electrical communication, vol. 64 No. 2/3,* (1990).

[22] A.R. Tripath, G.J. Lipovsk, " Packet Switching in Banyan Networks " , *Ann. Symp. Comp. Architect.,* (1979).

[23] D.M. Dias and J.R. Jump, " Analysis and Simulation of Buffered Delta Networks ", IEEE trans on Computers, Vol c-30, No. 10, (1981).

[24] Pattavina Achille, "Switching Theory : Architectures and Performance in Broadband Atm Networks", *John Wiley & Sons*, (1998).

[25] Kilkki Kalevi, "Differentiated Services for the Internet", *New Riders Publishing*, ISBN: 1578701325, (1998).

[26] Ferguson Paul, Huston Geoff, "Quality of Service: Delivering QoS on the Internet and in Corporate Networks", *John Wiley & Sons*, ISBN: 0471243582, (1998).

[27] Black Darryl P., "Building Switched Networks: Multilayer Switching, Qos, IP Multicast, Network Policy, and Service Level Agreements", *Addison-Wesley Pub Co*, ISBN: 0201379538, (1999).

[28] Giroux Natalie, Ganti Sudhakar, "Quality of Service for ATM Networks", *Prentice Hall* PTR, ISBN: 0130953873, (1999).

[29] McDysan David E., Spohn Darren L., "ATM Theory and Applications", *McGraw-Hill Professional Publishing*, ISBN: 0070453462, (1998).

[30] Schormans J. A., "Introduction to ATM/IP Design and Performance with Applications Analysis Software", *John Wiley & Sons*, ISBN: 047149187X, (2001).

[31] Black Uyless D., "QOS In Wide Area Networks", *Prentice Hall PTR* , ISBN: 0130264970, (2000).

[32] Gluzani Mohsen, Rays Ammars, "Designing ATM Switching Networks", *McGraw-Hill Professional Publishing*, ISBN: 0070252173, (1999).

[33] Cole Robert G., Ramaswamy Ravi, "Wide-Area Data Network Performance Engineering", *Artech House*, ISBN: 0890065691, (1999).

[34] Nassar Daniel, "Network Performance Baselining", *Macmillan Technical Publishing*, ISBN: 1578702402, (2000).

[35] Hassan Mahbub, Atiquzzaman Mohammed, "Performance of TCP/IP Over ATM Networks" *Artech House*, ISBN: 1580530370, (2000).

[36] McDysan David, "QoS and Traffic Management in IP and ATM Networks", *McGraw-Hill Professional Publishing*, ISBN: 0071349596, (1999).

[37] Ginsburg David, "Atm : Solutions for Enterprise Internetworking", *Addison-Wesley Pub Co*, ISBN: 0201877015, (2000).